RK Rockit MPI Developer Guide

文件标识：RK-SYS-MPI

发布版本：V1.1

日　　期：2021.11

文件密级：□绝密　□秘密　□内部资料　■公开

瑞芯微电子股份有限公司

Rockchip Electronics Co., Ltd.

地址：　福建省福州市铜盘路软件园A区18号

网址：　www.rock-chips.com

客户服务电话：+86-4007-700-590

客户服务传真：+86-591-83951833

客户服务邮箱：[fae@rock-chips.com]

前言

概述

本文档主要介绍MPI API和数据类型。

产品版本

芯片名称	内核版本
RV1126	4.19
RK356X	4.19
RK3588	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	2021-11-08	合并所有的MPI文档

系统控制

前言

概述

系统控制模块实现RK MPI通用功能接口，提供系统相关功能、大块物理内存管理等功能。

产品版本

芯片名称	内核版本
RK356X	4.19

读者对象

本文档（本指南）主要适用于以下工程师：

技术支持工程师

软件开发工程师

修订记录

版本号	作者	修改日期	修改说明
V0.2.0	许丽明	2021-01-19	初始版本
V0.3.0	许丽明	2021-02-07	增加接口和结构体索引，增加结构体描述
V0.4.0	方兴文	2021-03-27	增加输入流模式配置接口
V0.5.0	许丽明	2021-05-08	增加公共内存池操作接口
V0.6.0	许丽明	2021-08-18	增加图释，完善部分概念释义

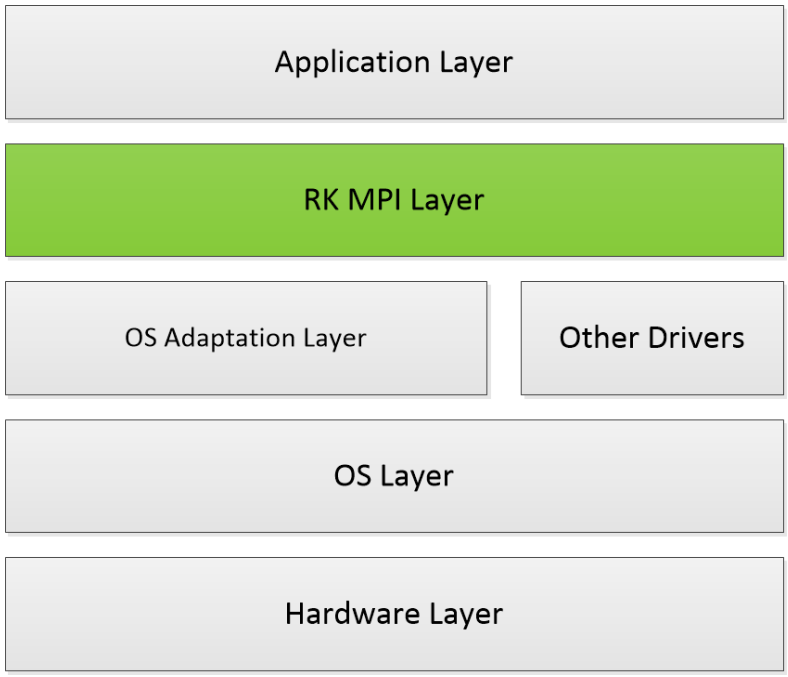
1. 目录

2. 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。

2.1 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。

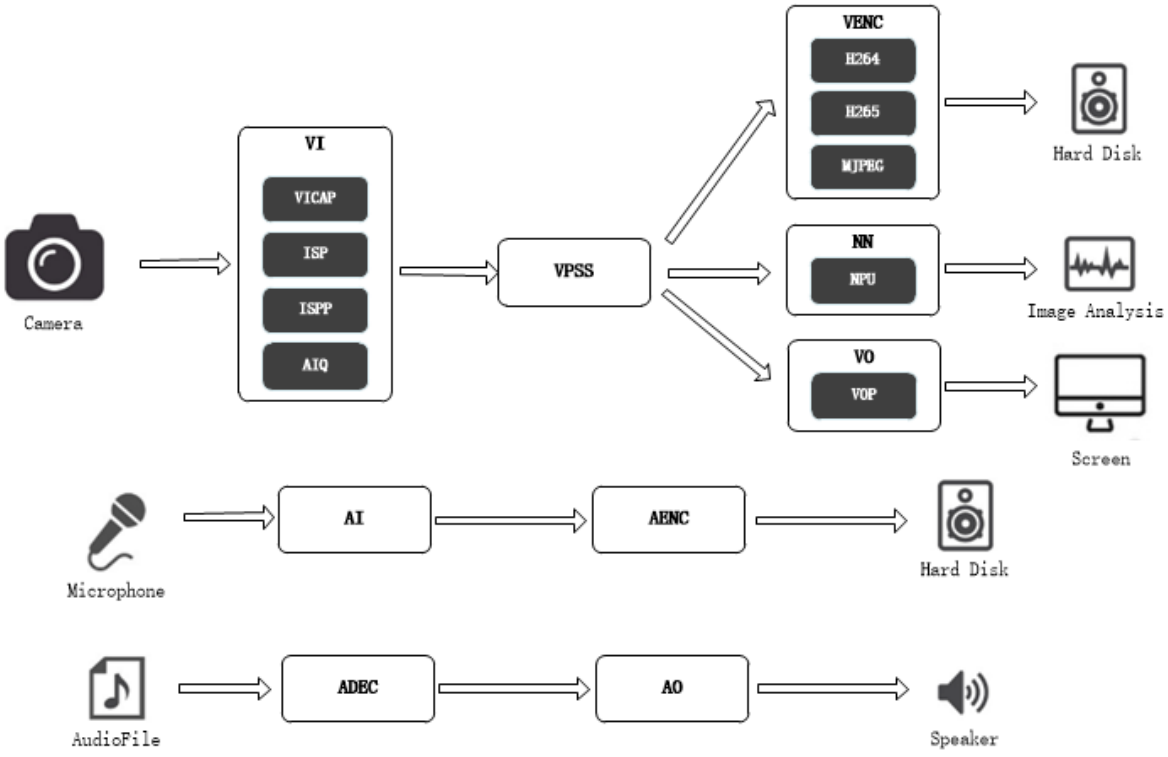
2.2 模块介绍

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。

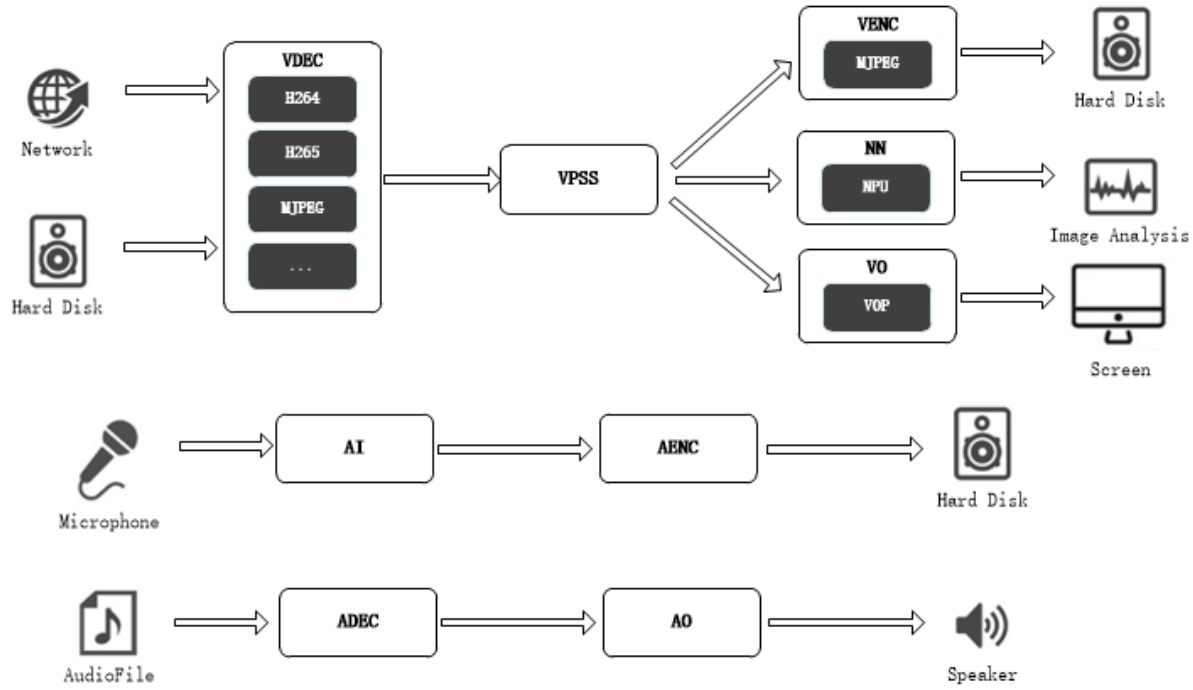
2.3 媒体处理平台典型应用

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。

2.3.1 IPC典型应用



2.3.2 NVR典型应用

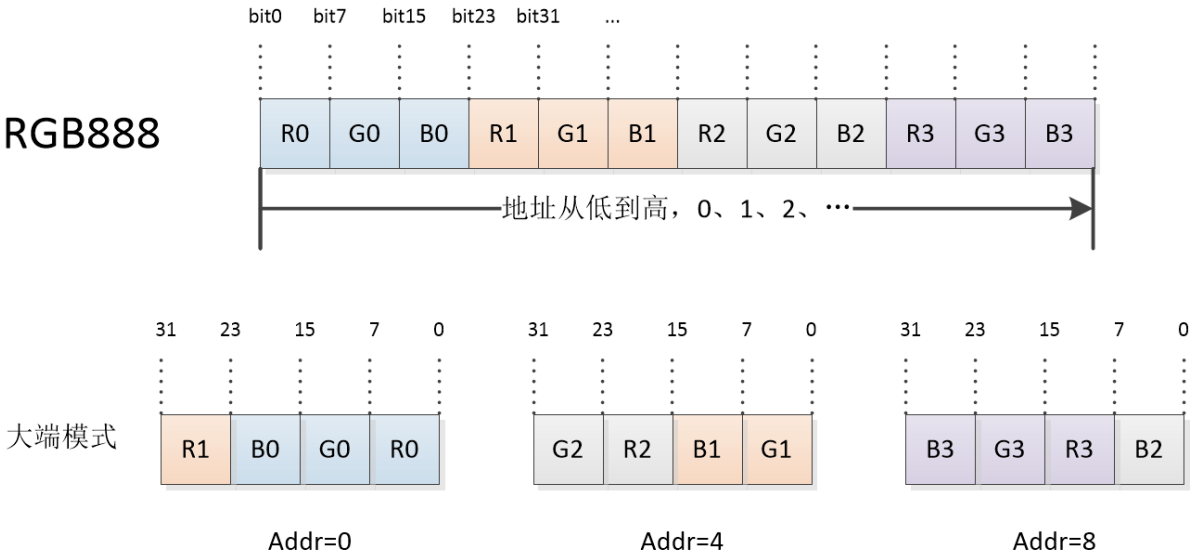


2.4 概念描述

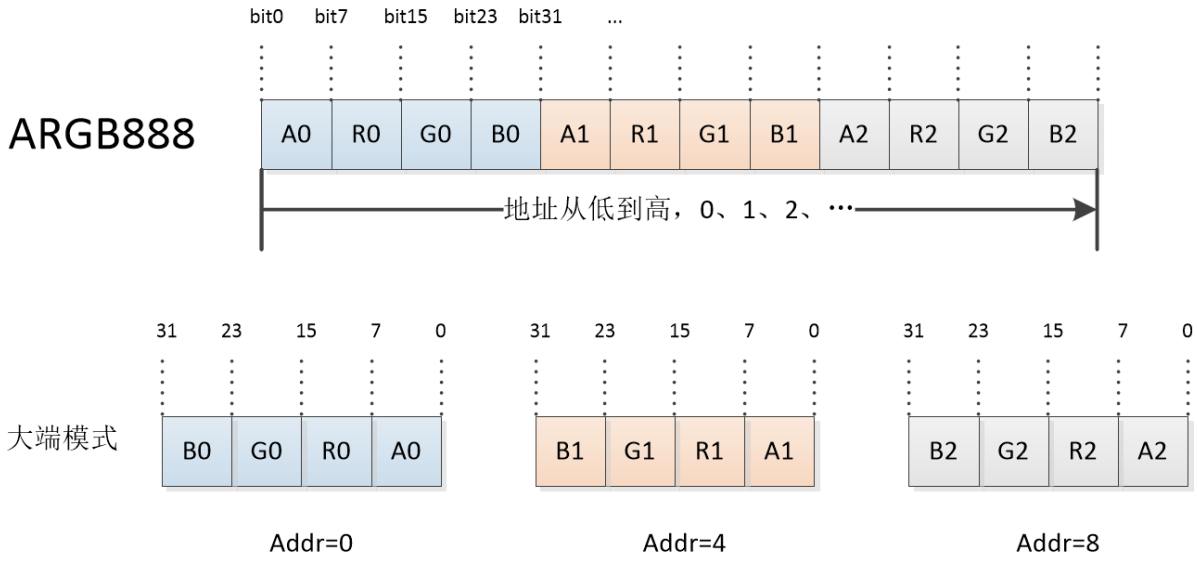
2.4.1 像素格式排布

RK MPI图像数据存储为**大端模式**，在平台图像输入和图像输出处理时需要注意这一点，否则将会出现RGB中B像素和R像素颠倒的情况出现。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）

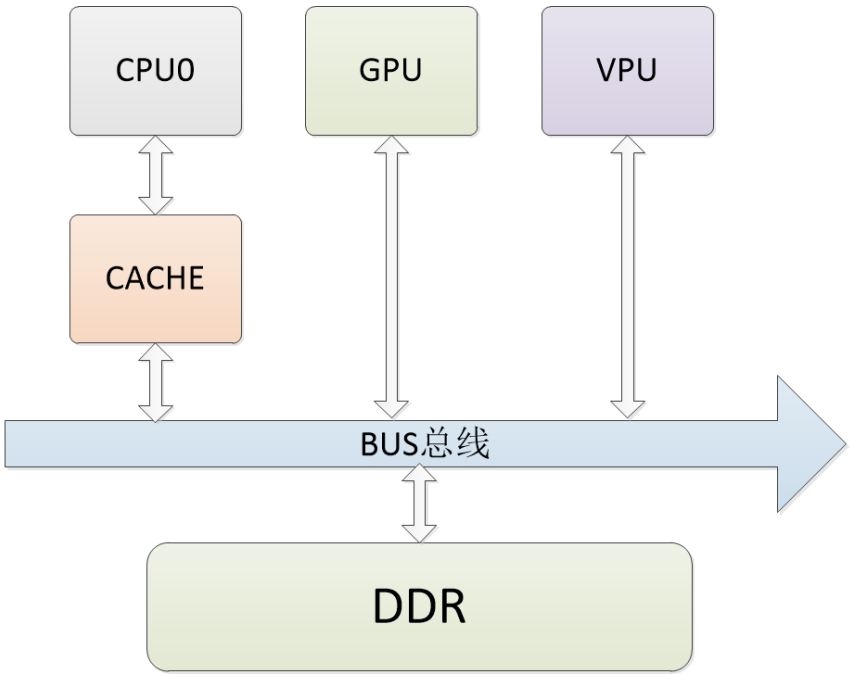


2.4.2 DMA内存与Cache的一致性

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。

3. 功能描述

3.1 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。

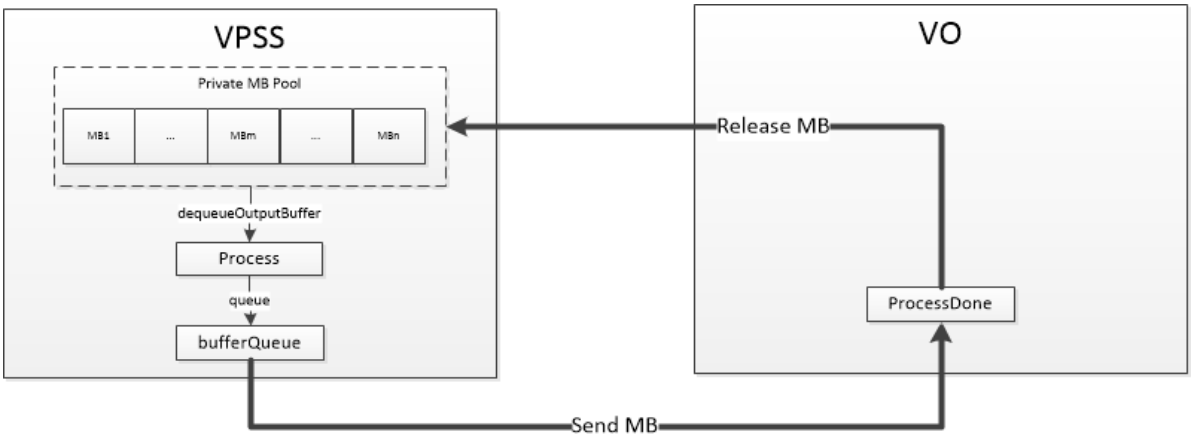
一组大小相同的内存缓存块组成一个内存缓存池。根据业务的不同，申请的缓存池的数量、缓存块的大小和数量不同。

RK提供两种内存缓存池模式：私有模式，共有模式，默认为私有模式。

3.1.1 私有模式

私有模式下，内存缓存池由通道自行申请，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。

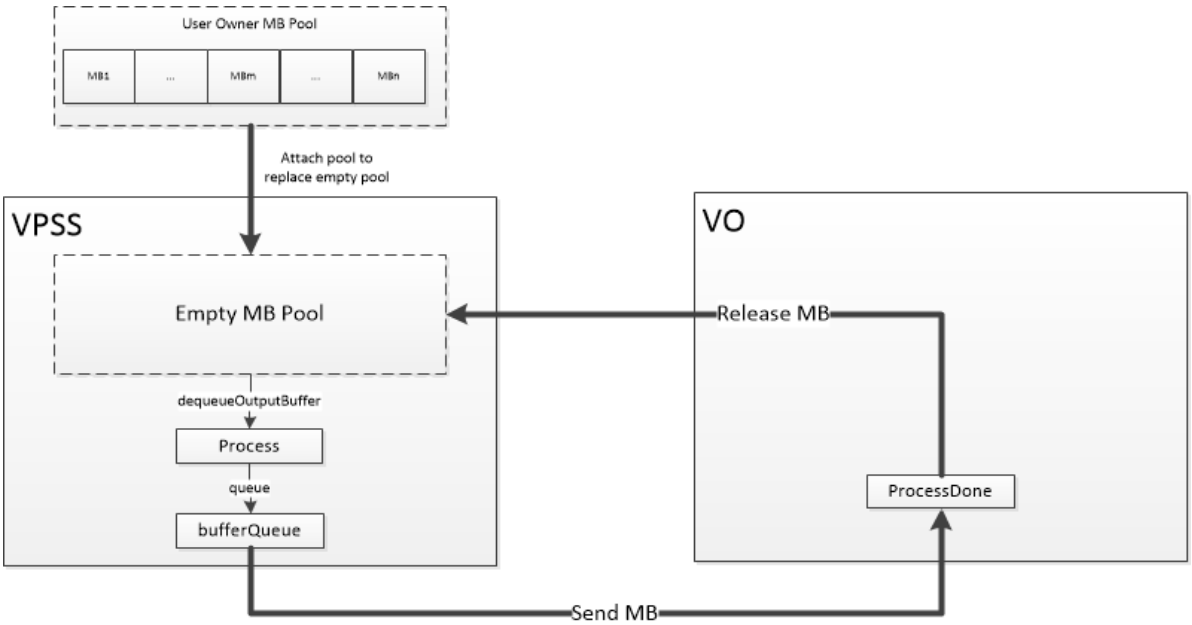
- 图2-1 内存池私有模式



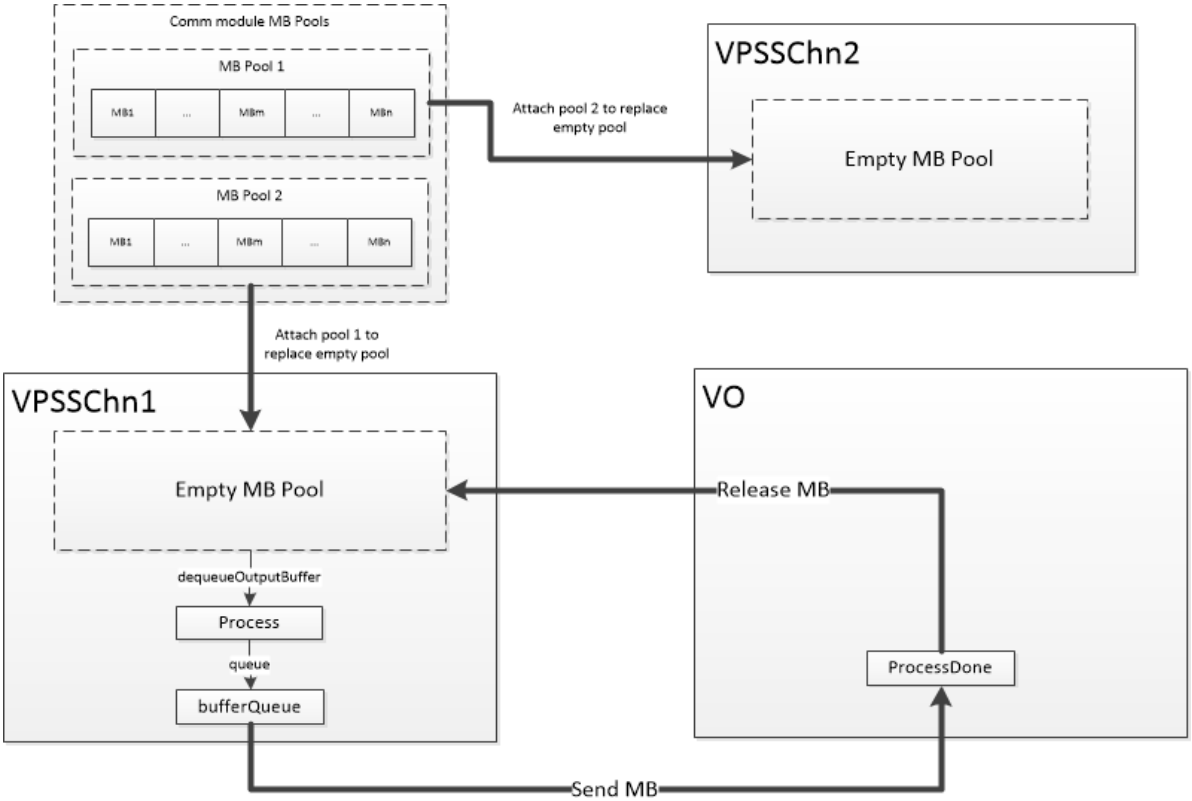
3.1.2 共有模式

共有模式下，内存池可以由用户通过模块的attach pool接口注入外部内存缓存池，也可以通过模块初始化确定所有模块分配大小、行为一致的通道使用共有的内存缓存池。

- 图2-2 内存池私有模式（用户注入）



- 图2-3 内存池共有模式（模块初始化分配）



详见：RK_MPI_MB_SetModPoolConfig接口。

3.1.3 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。

3.2 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
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

4. 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_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：获取一个缓存块的大小。
- RK_MPI_MB_VirAddr2Handle：根据虚拟地址获取对应的缓存块。
- 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_SetBufferStride：设置缓存块用于存储图像时该图像的虚宽高Stride。

4.1 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，请参见系统错误码。

【注意】

- 运行RK MPI系统前必须调用该接口进行初始化。

4.2 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，请参见系统错误码。

【注意】

- 退出RK MPI系统前必须调用该函数。

4.3 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，请参见系统错误码。

【注意】

- 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。
- 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。

4.4 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，请参见系统错误码。

【注意】

- 无

4.5 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，请参见系统错误码。

【注意】

- 无

4.6 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，请参见系统错误码。

【注意】

- 无

4.7 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，请参见系统错误码。

【注意】

- 无

4.8 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，请参见系统错误码。

【注意】

- 无

4.9 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，请参见系统错误码。

【注意】

- 无

4.10 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，请参见系统错误码。

【注意】

- 无。

4.11 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，请参见系统错误码。

【注意】

- 无

4.12 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，请参见系统错误码。

【注意】

- 无

4.13 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，请参见系统错误码。

【注意】

- 无

4.14 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，请参见系统错误码。

【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。

4.15 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，请参见系统错误码。

【注意】

- 无

4.16 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，请参见系统错误码。

【注意】

- 无

4.17 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

4.18 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，请参见系统错误码。

【注意】

- GROUP 必须已创建。

4.19 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

4.20 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，请参见系统错误码。

【注意】

- 无

4.21 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	成功。
非0	失败，请参见系统错误码。

【注意】

- 暂不支持。

4.22 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

4.23 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。

4.24 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。

4.25 RK_MPI_MB_GetSize

【描述】
获取一个缓存块的大小。

【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	缓存块的大小。

【注意】

- 无

4.26 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。

4.27 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

4.28 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，请参见系统错误码。

【注意】

- 需要在调用RK_MPI_SYS_Init前设置该属性。

4.29 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，请参见系统错误码。

【注意】

- 无

4.30 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，请参见系统错误码。

【注意】

- 无

5. 数据类型

基本数据类型定义如下：

5.1 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
```

5.2 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

5.3 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_BUTT = 9
} MB_UID_E;
```

【注意事项】
无

5.4 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_AVSPLUS,            /**< AVS+ profile=0x48 */
    RK_VIDEO_ID_AVS,                /**< 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 = 0x17000,          ///< 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   = 0x17800,
    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

5.5 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

5.6 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

5.7 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

5.8 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

5.9 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

5.10 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

5.11 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

5.12 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

5.13 MB_BLK

【说明】
定义MB内存的句柄。
【定义】

```
typedef void * MB_BLK;
```

【注意事项】
无

5.14 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

5.15 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

5.16 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

5.17 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

5.18 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

5.19 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。

5.20 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。
astCommPool	公共缓存池属性结构体。

【注意事项】
无

6. 系统错误码

6.1 系统控制系统错误码

系统控制 API 系统错误码如下所示：

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

6.2 5.2 内存缓存池系统错误码

错误代码	宏定义	描述
0xA0028003	RK_ERR_MB_ILLEGAL_PARAM	参数设置无效
0xA0028005	RK_ERR_MB_UNEXIST	内存缓存池不存在
0xA0028006	RK_ERR_MB_NULL_PTR	参数空指针错误
0xA0028009	RK_ERR_MB_NOT_PERM	操作不允许
0xA002800C	RK_ERR_MB_NOMEM	分配内存失败
0xA002800D	RK_ERR_MB_NOBUF	分配缓存失败
0xA0028010	RK_ERR_MB_NOTREADY	系统控制属性未配置
0xA0028012	RK_ERR_MB_BUSY	系统忙
0xA0028013	RK_ERR_MB_SIZE_NOT_ENOUGH	MB 块大小不够
0xA0028040	RK_ERR_MB_2MPOOLS	创建缓存池太多

内存管理

前言

概述

MMZ模块实现内存管理等功能。

产品版本

芯片名称	内核版本
RK356X	4.19

读者对象

本文档（本指南）主要适用于以下工程师：

技术支持工程师

软件开发工程师

修订记录

版本号	作者	修改日期	修改说明
V0.1.0	周弟东	2021-05-27	初始版本

1. 目录

2. 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查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_PhyAddr2Handle：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。

2.1 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。

2.2 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。

2.3 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。

2.4 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

2.5 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

2.6 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

2.7 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

2.8 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

2.9 RK_MPI_MMZ_PhyAddr2Handle

【描述】
通过物理地址查找对应的用户缓存。

【语法】

MB_BLK RK_MPI_MMZ_PhyAddr2Handle(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

2.10 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

2.11 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

2.12 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

2.13 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

2.14 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

2.15 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

2.16 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

3. 数据类型

基本数据类型定义如下：

3.1 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
```

3.2 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
```

3.3 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
```

3.4 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
```

3.5 RK_MMZ_SYNC_READONLY

【说明】
内存读操作。
【定义】

```
#define RK_MMZ_SYNC_READONLY        0x00000000
```

3.6 RK_MMZ_SYNC_WRITEONLY

【说明】
内存写操作。
【定义】

```
#define RK_MMZ_SYNC_WRITEONLY       0x00000001
```

3.7 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

前言

概述

本文档主要介绍VI的API和数据类型。

产品版本

芯片名称	内核版本
RV1126	4.19
RK356X	4.19

读者对象

本文档（本指南）主要适用于以下工程师：

技术支持工程师

软件开发工程师

修订记录

版本号	作者	修改日期	修改说明
v0.1.0	李鑫煌	2021-3-30	初始版本
v0.2.0	李鑫煌	2021-8-3	增加视频冻结功能说明
v0.3.0	李鑫煌	2021-9-7	增加功能描述说明
v0.3.1	李鑫煌	2021-9-27	修改文档格式
v0.4.0	李鑫煌	2021-10-26	完善结构体定义

1. 目录

2. 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。

3. 重要概念

- **3.0.1 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。

- **3.0.2 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 。（目前暂未实现内容，采用直通方式，同dev相同id设置即可）。

- **3.0.3 CHANNEL通道**

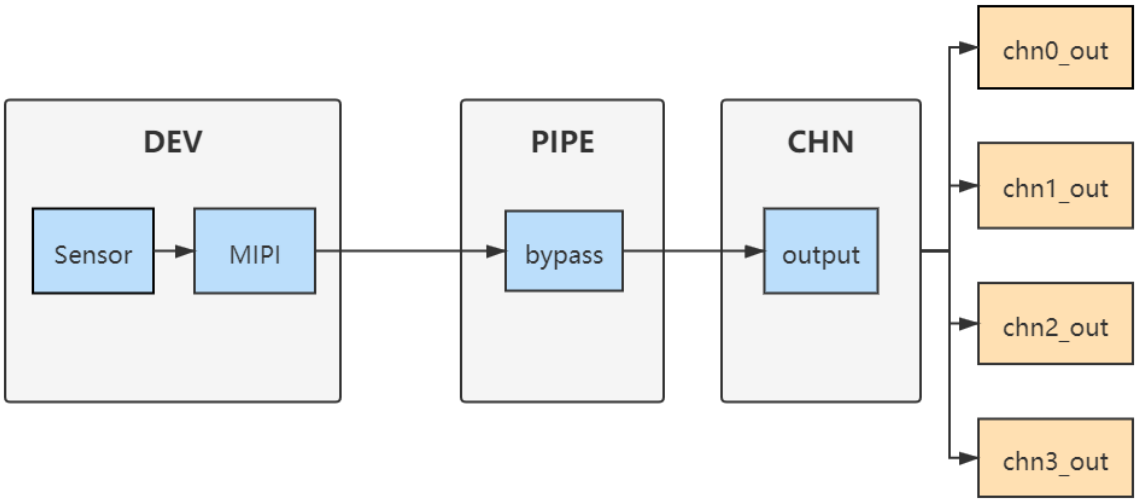
视频输入最后一级的获取通道，如dev为isp时输出有四个channel。

4. 功能描述

4.1 功能框图

VI 在软件层次上划分 3个部分，如图1-1所示。

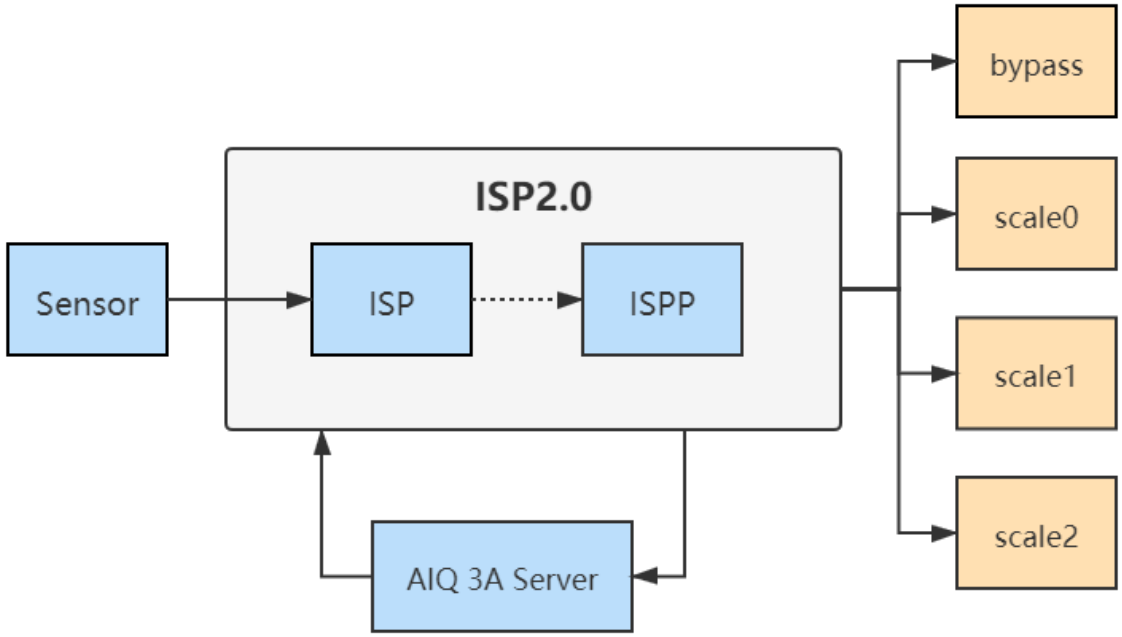
4.1.1 图1-1 VI处理流程框图



VI 从软件上划分了输入设备（dev），输入 pipe(暂未实现，采用dev直通到chn)、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	3	3	4
RK356X	3	3	4

4.1.2 图1-2 rv1126 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。

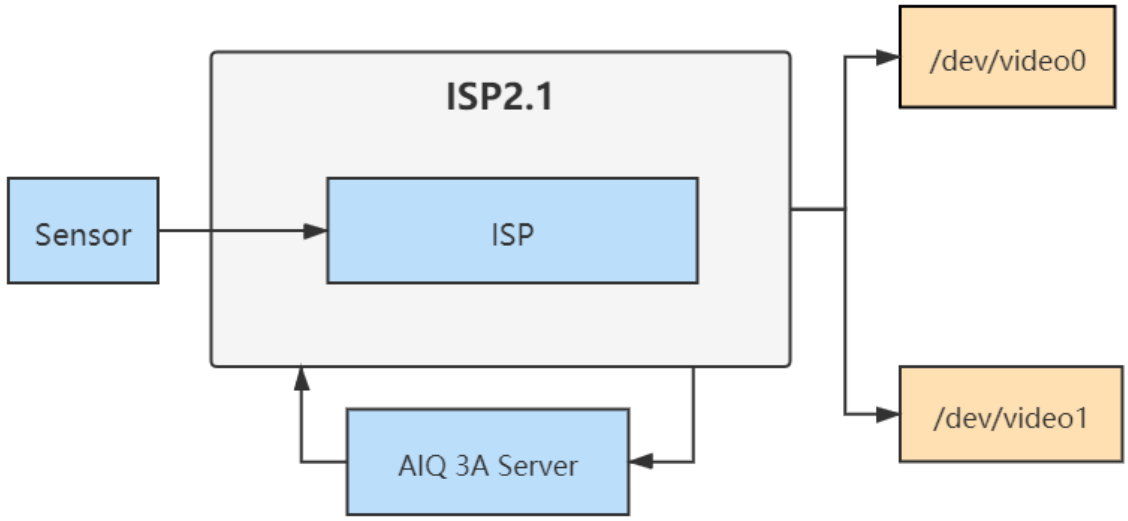
各路输出分辨率限制如下表：

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	NV12/NV16/YUYV/UYVY
rkispp_scale1 或/dev/video20	scale1	max width: 1280, 最大支持8倍缩放	NV12/NV16/YUYV/UYVY
rkispp_scale2 或/dev/video21	scale2	max width: 1280, 最大支持8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。
- 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。

4.1.3 图1-3 rk356x VI硬件处理流程框图



【注意】

- rk356x平台的aEntityName（参考 VI_ISP_OPT_S结构体定义）必须赋值。
- 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。

各路输出分辨率限制如下表：

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

5. 举例

```
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_GetDevIsEnable(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 = ctx->pipeId;
    ctx->stBindPipe.PipeId[0] = ctx->pipeId;
    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->pipeId, 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->pipeId, ctx->channelId, &ctx-
>stChnStatus);
//7.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);
}

//8. disable one chn
s32Ret = RK_MPI_VI_DisableChn(ctx->pipeId, ctx->channelId);
RK_LOGE("RK_MPI_VI_DisableChn %x", s32Ret);
//9.disable dev(will diabled 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。

6. API 参考

该功能模块为用户提供以下API:

- 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_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_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。

6.1 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。	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。

6.2 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。

6.3 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。

6.4 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。

6.5 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。	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。

6.6 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 。

6.7 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。

6.8 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。	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_INTERNAL。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采集数据bypass通路的分辨率支持。

- PIPE 必须已绑定到设备，否则会返回失败 。

6.9 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 。

6.10 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。

6.11 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 。

6.12 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, VI_FRAME_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	VI 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。

6.13 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 VI_FRAME_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	VI 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。

6.14 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。

6.15 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。

6.16 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频繁开关fd。

6.17 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

6.18 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	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

6.19 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

7. 数据类型

7.1 VI_DEV

【说明】

定义 VI 设备句柄。

【定义】

```
typedef RK_S32 VI_DEV;
```

7.2 VI_PIPE

【说明】

定义 VI PIPE。

【定义】

```
typedef RK_S32 VI_PIPE;
```

7.3 VI_CHN

【说明】

定义 VI 通道。

【定义】

```
typedef RK_S32 VI_CHN;
```

7.4 VI_MAX_DEV_NUM

【说明】

定义设备最大个数。

【定义】

```
#define VI_MAX_DEV_NUM  3
```

7.5 VI_MAX_PIPE_NUM

【说明】

定义PIPE最大个数。

【定义】

```
#define VI_MAX_PIPE_NUM  VI_MAX_DEV_NUM
```

7.6 VI_MAX_DEV_NUM

【说明】

定义VI输出设备的最大个数。

【定义】

```
#define VI_MAX_DEV_NUM 4
```

7.7 VI_DEV_ATTR_S

【说明】

定义VI设备属性结构体。（暂未使用，可以选择传入未初始化的该数据结构）

【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;
} 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）

7.8 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[VI_MAX_PHY_PIPE_NUM];                /* RW;Array of pipe ID */
} 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。

【注意事项】

- 目前PIPE传入跟DEV一样的数值即可。

  ctx->stBindPipe.u32Num = ctx->devId;
  ctx->stBindPipe.PipeId[0] = ctx->devId;

7.9 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 */
} 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]，0则获取不到输出buf。
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。 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。

7.10 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。

7.11 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

7.12 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_E;
```

【成员】

成员名称	描述
VI_ALLOC_BUF_TYPE_INTERNA	VI内部分配
VI_ALLOC_BUF_TYPE_EXTERNAL	VI外部分配

7.13 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 */
} 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	通道获取图像大小设置。

7.14 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。

7.15 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。

7.16 VI_FRAME_S

【说明】

VI 帧信息结构体。

【定义】

```
typedef struct rkVI_FRAME_S {
    MB_BLK            pMbBlk;                /* R; the mediabuf of frame */
    RK_U32            u32UniqueId;           /* R; the UniqueId of frame */
    RK_U32            u32Fd;                 /* R; the fd of frame */
    RK_U32            u32Len;                /* R; the length of frame */
    RK_S64            s64PTS;                /* R; PTS */
    RK_S32            s32Seq;                /* R; the list number of
frame*/
    PIXEL_FORMAT_E    enPixelFormat;         /* R; the pixel format */
    SIZE_S            stSize;                /* R; the frame out put size */
    COMPRESS_MODE_E   enCompressMode;        /* R; 256B Segment compress or
no compress. */
} VI_FRAME_S;
```

【成员】

成员名称	描述
pMbBlk	输出图像buf的mediabuff结构体。
u32UniqueId	输出图像buf的UniqueId。
u32Fd	输出图像buf的fd。
u32Len	输出图像的长度。
s64PTS	输出图像的时间戳。
s32Seq	输出图像的序列号。
enPixelFormat	输出图像的格式。
stSize	输出图像的大小。
enCompressMode	输出图像的压缩格式。

7.17 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  u32LostFrame;               /* RO;Lost frame count */
    RK_U32  u32VbFail;                  /* RO;Video buffer malloc failure */
    SIZE_S  stSize;                     /* RO;chn 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。
u32LostFrame	当前通道丢失帧数。
u32VbFail	当前通道获取图像失败次数。
stSize	当前通道图像大小。

7.18 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。

8. VI错误码

视频输入APIVI错误码如下所示。

错误代码	宏定义	描述
0xA0088001	RK_ERR_VI_INVALID_DEVID	VI设备号无效
0xA0088002	RK_ERR_VI_INVALID_CHNID	VI通道号无效
0xA0088003	RK_ERR_VI_INVALID_PARA	VI参数设置无效
0xA0088004	RK_ERR_VI_NOT_ENABLED	VI设备或通道没使能
0xA0088005	RK_ERR_VI_UNEXIST	不存在通道对象
0xA0088006	RK_ERR_VI_INVALID_NULL_PTR	VI空指针错误
0xA0088007	RK_ERR_VI_NOT_CONFIG	VI参数未配置
0xA0088008	RK_ERR_VI_NOT_SUPPORT	操作不允许
0xA0088009	RK_ERR_VI_NOT_PERM	无权限操作
0xA008800A	RK_ERR_VI_INVALID_PIPEID	VI PIPE号无效
0xA008800B	RK_ERR_VI_NOMEM	无内存
0xA008800E	RK_ERR_VI_BUF_EMPTY	VI输入缓冲为空
0xA008800F	RK_ERR_VI_BUF_FULL	VI输入缓存为满
0xA0088010	RK_ERR_VI_SYS_NOTREADY	系统未准备好
0xA0088012	RK_ERR_VI_BUSY	VI设备忙

视频处理子系统

前言

概述

VPSS（Video Process Sub-System）是视频处理子系统，支持的具体图像处理功能包括CROP、Scale、像素格式转换、固定角度旋转、Cover/Coverex、Mirror/Flip、压缩解压等。

产品版本

芯片名称	内核版本
RK356X	4.19
RV1109/RV1126	4.19

读者对象

本文档（本指南）主要适用于以下工程师：

技术支持工程师

软件开发工程师

修订记录

版本号	作者	修改日期	修改说明
V0.1.0	许丽明	2020-12-16	初始版本
V0.2.0	许丽明	2021-01-23	完善数据结构定义
V0.3.0	许丽明	2021-02-05	修正部分勘误，增加函数、结构体跳转
V0.3.1	许丽明	2021-06-11	修正部分勘误，增加部分接口的释义
V0.4.0	许丽明	2021-07-27	增加图释及VPSS格式支持列表
V0.4.1	金克木	2021-09-29	修正部分勘误，增加对RV1109/RV1126的支持以及部分参数的释义

1. 目录

2. 功能描述

2.1 基本概念

2.1.1 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。

2.1.2 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。

2.1.3 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。

2.1.4 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。

2.1.5 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。

2.1.6 Mirror/Flip

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

2.1.7 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。

2.1.8 Cover

视频遮挡区域，调用 VGS 对 VPSS 通道的输出图像填充纯色块。

2.1.9 Overlay

视频叠加区域，调用 VGS 对 VPSS 通道的输出图像叠加位图。

2.1.10 Mosaic

马赛克，调用 VGS 对 VPSS 通道的输出图像填充马赛克块。

2.1.11 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。

2.1.12 压缩

支持AFBC压缩。

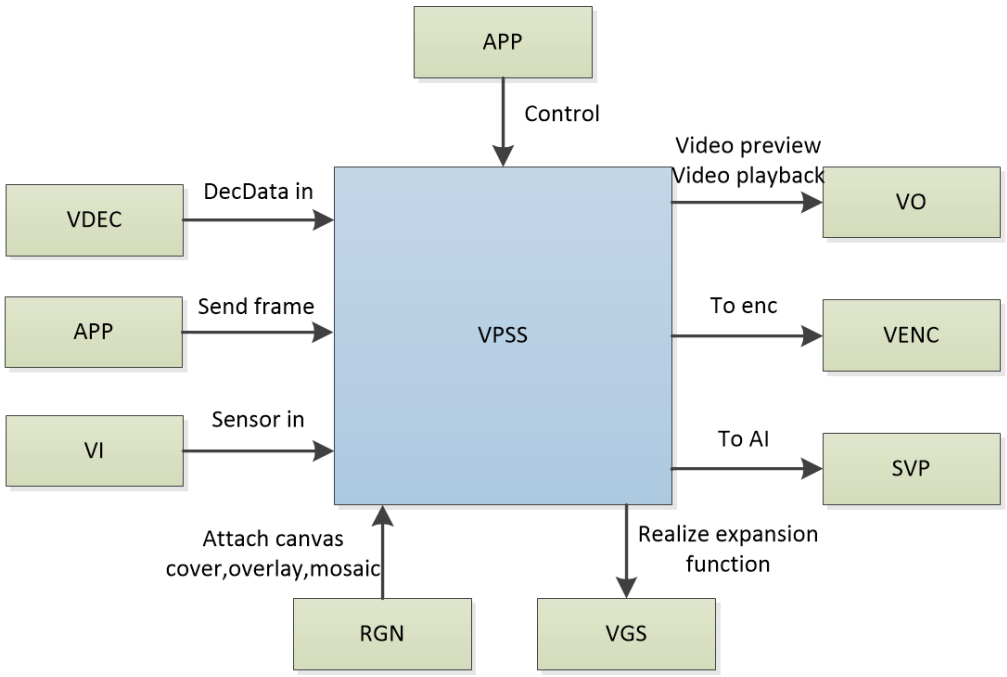
2.1.13 解压

支持AFBC解压。

2.2 应用方式

VPSS 在系统中的定位如图1-1所示。

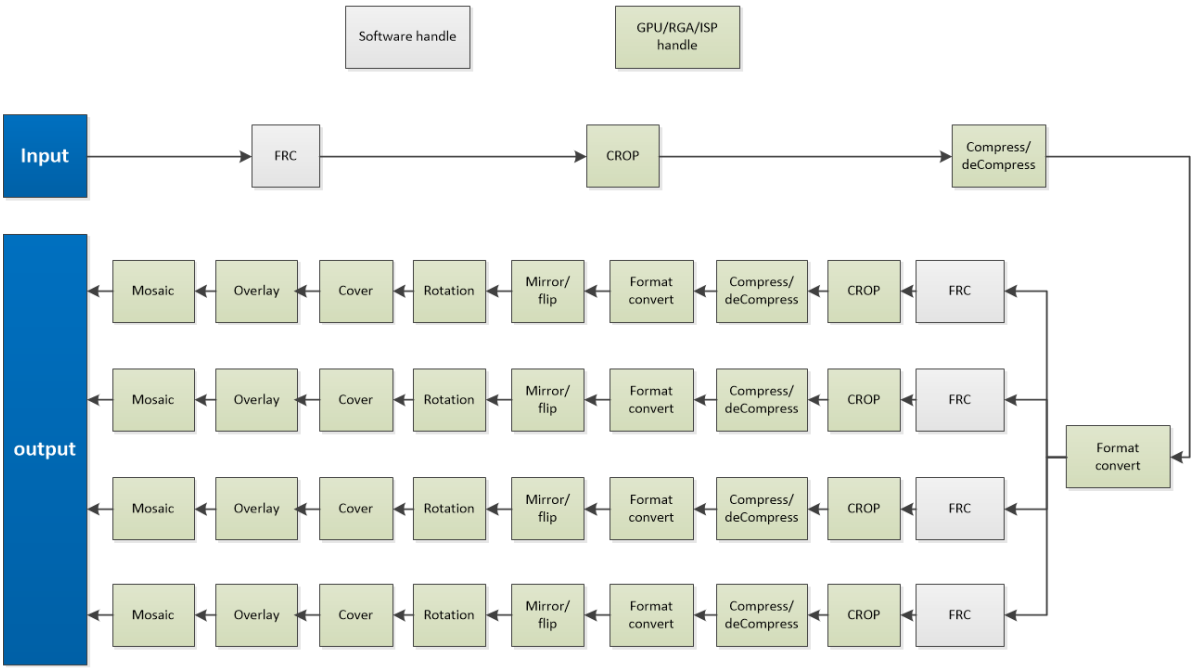
2.2.1 图1-1 VPSS上下文关系



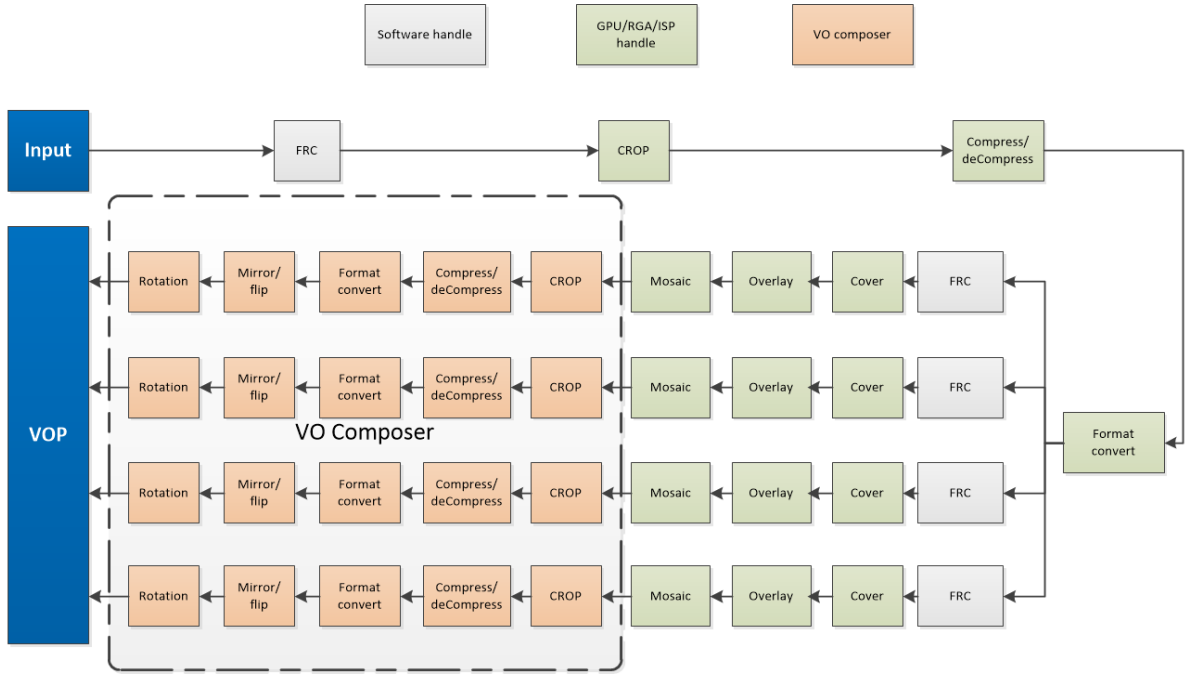
通过调用 SYS 模块的绑定接口，可与 VDEC/VI 和 VO/VENC/SVP 等模块进行绑定，其中前者为 VPSS 的输入源，后者为 VPSS 的接收者。用户可通过 VPSS MPI 接口对 Group 进行管理。每个 Group 仅可与一个输入源绑定。Group 的通道有两种工作模式：USER 和 PASSTHOUGH，两种模式间可动态切换。USER 模式下主要用于非绑定模式下，从通道手动取帧，用于AI算法、用户数据分析处理等场景，PASSTHOUGH 模式主要用于预览和回放场景下做播放控制，如电子放大。两种模式下各通道均可与多个接收者绑定。

2.3 芯片处理流程

2.3.1 RK356X VPSS user模式数据流图



2.3.2 RK356X VPSS passthrough模式数据流图



VPSS在passthrough工作模式下，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。

2.4 输入输出特性

2.4.1 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：

2.4.1.1 RK356X输出图像支持列表

数据格式	宽度对齐字节	高度对齐字节	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 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

2.4.1.2 RV1109/RV1126输出图像支持列表

数据格式	宽度对齐字节	高度对齐字节	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

2.5 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;
    }
```

3. 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_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。

3.1 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VO错误码。

【注意】

- 不支持重复创建。

3.2 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VO错误码。

【注意】

- GROUP 必须已创建。
- 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。

3.3 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VO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。

3.4 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VO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。

3.5 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VO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。

3.6 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VO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。

3.7 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VO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。

3.8 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VO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passthrough模式，并使用RK_MPI_VPSS_SetChnCrop做电子放大功能。

3.9 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VO错误码。

【注意】

- GROUP 必须已创建。

3.10 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VO错误码。

【注意】

- GROUP 必须已创建。

3.11 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VO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passthrough模式时才可获取。

3.12 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VO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。

3.13 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VO错误码。

【注意】

- GROUP 必须已创建。

3.14 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VO错误码。

【注意】

- GROUP 必须已创建。

3.15 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VO错误码。

【注意】

- GROUP 必须已创建。

3.16 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VO错误码。

【注意】

- GROUP 必须已创建。

3.17 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VO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。

3.18 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VO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。

3.19 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VO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。

3.20 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VO错误码。

【注意】

- GROUP 必须已创建。

3.21 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VO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。
- 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

3.22 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VO错误码。

【注意】

- GROUP 必须已创建。

3.23 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VO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。

3.24 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VO错误码。

【注意】

- GROUP 必须已创建。

3.25 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, 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VO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。
- 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。

3.26 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VO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。

3.27 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VO错误码。

【注意】

- 必须保证组已创建。
- 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_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。

3.28 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VO错误码。

【注意】

- 必须保证组已创建。

4. 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_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。

4.1 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】
无

4.2 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。

4.3 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】
无

4.4 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】
无

4.5 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】
无

4.6 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】
无

4.7 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】
无

4.8 VPSS_GRP

【说明】
定义 VPSS 组号。
【定义】

```
typedef RK_S32 VPSS_GRP;
```

【注意事项】
无

4.9 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。

4.10 VPSS_CHN

【说明】
定义 VPSS 通道号。
【定义】

```
typedef RK_S32 VPSS_CHN;
```

【注意事项】
无

4.11 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。

4.12 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。

4.13 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】
无

4.14 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。
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。

4.15 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;
} VPSS_CHN_ATTR_S;
```

【成员】

成员名称	描述
enChnMode	通道工作模式。
u32Width	目标图像宽度。
u32Height	目标图像高度。
enVideoFormat	目标图像视频格式。
enPixelFormat	目标图像像素格式。
enDynamicRange	目标图像动态范围。
enCompressMode	目标图像压缩模式。
stFrameRate	帧率控制信息。
bMirror	水平镜像使能。
bFlip	垂直翻转使能。
u32Depth	用户获取通道图像的队列长度。 取值范围：[0, 8]
stAspectRatio	幅形比参数。

【注意事项】

- 源帧率与目标帧率都为-1，则不进行帧率控制。
- u32Depth深度为0时，表示不保留通道图像，全部丢弃。
- u32Depth深度仅在通道非绑定模式下（无绑定后级模块）时生效。
- 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。

4.16 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Passthrough模式，auto模式弃用，工作模式的工作流程详见1.3.1 RK356X VPSS user模式数据流图和1.3.2 RK356X VPSS passthrough模式数据流图。

5. VO错误码

视频处理子系统 API VO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
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

前言

概述

VDEC 模块提供驱动视频解码硬件工作的 MPI 接口，实现视频解码功能。

产品版本

芯片名称	内核版本
RK356X	4.19

读者对象

本文档（本指南）主要适用于以下工程师：

技术支持工程师

软件开发工程师

修订记录

版本号	作者	修改日期	修改说明
V0.1.0	杨文杰	2021-01-09	初始版本
V0.2.0	杨文杰	2021-01-23	完善数据结构定义
V0.3.0	杨文杰	2021-09-09	增加芯片特性描述、补充接口说明
V1.0.0	方兴文	2021-10-02	补充并修改部分描述、整理文档格式

1. 目录

2. 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
RK356X	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

3. 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。若该虚拟内存在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。

> 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。

- 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。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。

4. 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最大解码通道数，以减少通道上下文相关的内存占用。

5. 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。

5.1 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。

5.2 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。

5.3 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（注：u32FrameBufSize大小计算复杂有内部决定，暂不开放配置）。
- 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。

5.4 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。

5.5 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。

5.6 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。

5.7 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。

5.8 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。

5.9 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。

5.10 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。

5.11 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。

5.12 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。

5.13 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。

5.14 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，然后占住当前的用户图片 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。

5.15 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。

5.16 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。

5.17 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。

5.18 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。

5.19 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_GetDisplayMode。

5.20 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。

5.21 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。
非正数值	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。

5.22 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。

5.23 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。

5.24 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），则返回错误码 RK_ERR_VDEC_NOT_SUPPORT。

【举例】
无。

【相关主题】
RK_MPI_VDEC_AttachMbPool。

6. 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。

6.1 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

6.2 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	暂不开放配置，由内部按需计算大小。	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 0，默认值为 8	静态属性。

【注意事项】

- 帧存个数分配方法：
  1 如果u32FrameBufCnt > 0，帧存个数按照u32FrameBufCnt 来分配
  2 如果u32FrameBufCnt == 0 && u32RefFrameNum > 0，帧存个数按照u32RefFrameNum + 2来分配
  3 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

【举例】
无。

【相关主题】
无。

6.3 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	参考帧的数目。 取值范围：[0, 16]，以帧为单位。 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。 -IPC码流：推荐设为 5。 -测试码流：推荐设为 16。	静态属性。
bTemporalMvpEnable	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，即需要设置 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，bTemporalMvpEnable 设置为 0。	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。

6.4 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。

6.5 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。

6.6 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 chnnel 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; refrence 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 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。

6.7 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

6.8 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

6.9 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; out put 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

6.10 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

6.11 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

6.12 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

6.13 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。

6.14 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。

6.15 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。

6.16 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。

6.17 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_PRIVATE
u32MiniBufMode	不支持。
u32ParallelMode	不支持。
stVideoModParam	不支持。

【注意事项】
无

【举例】
无。

【相关主题】
无。

6.18 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。

6.19 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	不支持。
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。
pVirAddr	图像Y和UV的地址。 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。

7. 错误码

视频解码 API 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

前言

概述

VENC 模块，即视频编码模块。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，调度 Region 模块对编码图像内容进行叠加和遮挡。

产品版本

芯片名称	内核版本
RV1109/RV1126	4.19
RK356X	4.19

读者对象

本文档（本指南）主要适用于以下工程师：

技术支持工程师

软件开发工程师

修订记录

版本号	作者	修改日期	修改说明
V0.1.0	许丽明	2021-01-09	初始版本
V0.2.0	许丽明	2021-01-23	完善数据结构定义
V0.3.0	李鑫煌	2021-09-24	完善新增功能说明及芯片支持规格信息

1. 目录

2. 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

芯片	支持的编码类别	支持的编码格式	支持的编码规格
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title编码

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_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_GetStream：获取编码码流。
- RK_MPI_VENC_ReleaseStream：释放码流缓存。
- 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_SetJpegParam：设置 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：获取编码通道的感兴趣区域编码配置。
- RK_MPI_VENC_SetRoiAttr：设置编码通道的感兴趣区域编码配置。
- 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_SetChnRotation：设置通道旋转角度。
- RK_MPI_VENC_GetChnRotation：获取通道旋转角度。
- 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_SetH264Dblk：设置H.264协议编码通道的Deblocking 类型。
- RK_MPI_VENC_GetH264Dblk：获取H.264协议编码通道的Deblocking 类型。
- 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_SetH265Dblk：设置H.265协议编码通道的Deblocking 类型。
- RK_MPI_VENC_GetH265Dblk：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。

4.1 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

4.2 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，请参见错误码。

【注意】

无

4.3 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，请参见错误码。

【注意】

无

4.4 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，请参见错误码。

【注意】

无

4.5 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，请参见错误码。

【注意】

无

4.6 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，请参见错误码。

【注意】

无

4.7 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，请参见错误码。

【注意】

无

4.8 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，请参见错误码。

【注意】

无

4.9 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，请参见错误码。

4.10 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，请参见错误码。

【注意】

无

4.11 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，请参见错误码。

【注意】

无

4.12 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，请参见错误码。

【注意】

无

4.13 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，请参见错误码。

【注意】

无

4.14 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，请参见错误码。

【注意】

无

4.15 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，请参见错误码。

【注意】

无

4.16 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，请参见错误码。

【注意】

无

4.17 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，请参见错误码。

【注意】

无

4.18 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，请参见错误码。

【注意】

无

4.19 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，请参见错误码。

【注意】

无

4.20 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，请参见错误码。

【注意】

- 关闭文件句柄需要调用RK_MPI_VENC_CloseFd。

4.21 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，请参见错误码。

【注意】

无

4.22 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，请参见错误码。

【注意】

- 如果通道未创建，则返回失败。

4.23 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，请参见错误码。

【注意】

- 如果通道未创建，则返回失败。

4.24 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。

4.25 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。

4.26 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，请参见错误码。

【注意】

- 如果通道未创建，则返回失败。

4.27 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。

4.28 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，请参见错误码。

【注意】

- 如果通道未创建或者pstSuperFrmParam为空，则返回失败。

4.29 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，请参见错误码。

【注意】

- 如果通道未创建，则返回失败。

4.30 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，请参见错误码。

【注意】

- 如果通道未创建或者enRotation为空，则返回失败。

4.31 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。

4.32 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。

4.33 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_GetH264Trans接口，获取当前编码通道的trans配置，然后再进行设置。

4.34 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。

4.35 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。

4.36 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。

4.37 RK_MPI_VENC_SetH264Dblk

【描述】

设置H.264协议编码通道的 Deblocking 类型。

【语法】

RK_S32 RK_MPI_VENC_SetH264Dblk(VENC_CHN VeChn, const 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。	输入

【返回值】

返回值	描述
0	成功。
非0	失败，请参见错误码。

【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。

4.38 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。

4.39 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。

4.40 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。

4.41 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。

4.42 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。

4.43 RK_MPI_VENC_SetH265Dblk

【描述】

设置H.265通道的Deblocking属性。

【语法】

RK_S32 RK_MPI_VENC_SetH265Dblk(VENC_CHN VeChn, const VENC_H265_DBLK_S *pstH265Dblk);

【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
pstH265Dblk	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，请参见错误码。

【注意】

- 如果通道未创建或者pstH265Dblk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_GetH265Dblk接口，获取当前编码通道的dblk配置，然后再进行设置。

4.44 RK_MPI_VENC_GetH265Dblk

【描述】

获取H.265通道的Deblocking属性。

【语法】

RK_S32 RK_MPI_VENC_GetH265Dblk(VENC_CHN VeChn, VENC_H265_DBLK_S *pstH265Dblk);

【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
pstH265Dblk	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，请参见错误码。

【注意】

- 如果通道未创建或者pstH265Dblk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。

4.45 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。

4.46 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。

4.47 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。

4.48 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。

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;
    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 stAttrMjpege;
        VENC_ATTR_JPEG_S stAttrJpege;
    };
} VENC_ATTR_S;
```

【成员】

成员名称	描述
enType	编码协议类型。H264、H265等。
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。 静态属性。
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 静态属性。
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。 静态属性。
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】

无

5.4 VENC_RC_ATTR_S

【说明】

定义编码通道码率控制器属性。

【定义】

```
    /* RW; the type of rc*/
    VENC_RC_MODE_E enRcMode;
```

```
    union {
        VENC_H264_CBR_S stH264Cbr;
        VENC_H264_VBR_S stH264Vbr;
        VENC_H264_AVBR_S stH264Avbr;

        VENC_MJPEG_CBR_S stMjpegCbr;
        VENC_MJPEG_VBR_S stMjpegVbr;

        VENC_H265_CBR_S stH265Cbr;
        VENC_H265_VBR_S stH265Vbr;
        VENC_H265_AVBR_S stH265Avbr;
    };
```

【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。
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。
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。

【注意事项】

无

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_PARAM_H264_S {
    RK_U32 u32StepQp;
    RK_U32 u32MaxQp;   // RW; Range:[8, 51];the max QP value
    RK_U32 u32MinQp;   // RW; Range:[0, 48]; the min QP value,can not be larger
than
                       // u32MaxQp
    RK_U32 u32MaxIQp;  // RW; max qp for i frame
    RK_U32 u32MinIQp;  // RW; min qp for i frame,can not be larger
                       // than u32MaxIQp
} VENC_PARAM_H264_S;
```

```
typedef struct rkVENC_PARAM_H265_S {
    RK_U32 u32StepQp;
    RK_U32 u32MaxQp;   // RW; Range:[8, 51];the max QP value
    RK_U32 u32MinQp;   // RW; Range:[0, 48];the min QP value ,can not be larger
than
                       // u32MaxQp
    RK_U32 u32MaxIQp;  // RW; max qp for i frame
    RK_U32 u32MinIQp;  // RW; min qp for i frame,can not be larger than u32MaxIQp
} VENC_PARAM_H265_S;

typedef struct rkVENC_PARAM_MJPEG_S {
    RK_U32 u32Qfactor;
} VENC_PARAM_MJPEG_S;

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
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】

无

5.6 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;
    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_S16              s16OffsetTop;
    RK_S16              s16OffsetBottom;
    RK_S16              s16OffsetLeft;
    RK_S16              s16OffsetRight;

    RK_U32              u32MaxLuminance;
    RK_U32              u32MinLuminance;

    RK_U32              u32TimeRef;
    RK_U64              u64PTS;

    RK_U64              u64PrivateData;
    RK_U32              u32FrameFlag;
    RK_BOOL             bBypassMbBlk;
} 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。
enVideoFormat	目标图像视频格式。
enPixelFormat	目标图像像素格式。
enDynamicRange	目标图像动态范围。
enCompressMode	目标图像压缩模式。
u64PTS	图像时间戳。

【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);
```

data = RK_MPI_MB_Handle2VirAddr(sFrame.stVFrame.pMbBlk);

5.7 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;
} 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。

【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

5.8 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;
} 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。

【注意事项】

请参见 VENC_H264_CBR_S注意事项

5.9 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;
} 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。

【注意事项】

请参见 VENC_H264_CBR_S注意事项

5.10 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;
} 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。

【注意事项】

请参见 VENC_H264_CBR_S注意事项

5.11 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;
} 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。

【注意事项】

请参见 VENC_H264_CBR_S注意事项

5.12 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;
} 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。

【注意事项】

请参见 VENC_H264_CBR_S注意事项

5.13 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;
} 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。

【注意事项】

请参见 VENC_H264_CBR_S注意事项

5.14 VENC_MJPEG_VBR_S

【说明】

定义 mjpeg 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;
} 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。

【注意事项】

请参见 VENC_H264_CBR_S注意事项

5.15 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_U32 u32GopSize;
    RK_S32 s32IPQpDelta;
    RK_U32 u32BgInterval;
    RK_S32 s32ViQpDelta;
} VENC_GOP_ATTR_S;
```

【成员】

成员名称	描述
enGopMode	编码 GOP 类型。
u32GopSize	GOP长度。
s32IPQpDelta	I 帧相对 P 帧的 QP 差值。
u32BgInterval	长期参考帧的间隔。 取值范围：[u32GopSize, 65536]，且必须是 u32GopSize的整数倍。
s32ViQpDelta	虚拟 I 帧相对于普通 P 帧的 QP 差值。 取值范围：[-10, 30]。

【注意事项】

无

5.16 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 ∞]

【注意事项】

无

5.17 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	Jpege 抓拍模式下指示抓拍过程是否结束。
stVencStrmInfo	编码器码流特征信息。

【注意事项】

无

5.18 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。
u32Seq	码流序列号。按帧获取帧序号；按包获取包序号。
stH624Info/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

5.19 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。
u64PTS	时间戳。单位：us。
bFrameEnd	帧结束标识。 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。
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。
stPackInfo	当前码流包数据中包含其他类型码流包数据信息。

【注意事项】

无

5.20 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	具体含义请参见 RFC2435 协议，系统默认为 70。取值范围：[1, 99]。
u8YQt	Y 量化表。取值范围：[1, 255]。（暂未使用，可不设置）
u8CbQt	Cb 量化表。取值范围：[1, 255]。（暂未使用，可不设置）
u8CrQt	Cr 量化表。取值范围：[1, 255]。（暂未使用，可不设置）
u32MCUPerECS	每个 ECS 中包含多少个 MCU，系统默认为 0，表示不划分 Ecs。（暂未使用，可不设置）

【注意事项】

无

5.21 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_TURE：绝对 QP
s32Qp	QP 值，当 QP 模式为 RK_FALSE 时，s32Qp 为 QP 偏移，s32Qp 范围[-51,51]，当 QP 模式为 RK_TRUE 时，s32Qp 为宏块 QP 值，s32Qp 范围[0,51]。
bIntra	是否为I帧
RECT_S	ROI 区域。s32X、s32Y、u32Width、u32Height 必须是 16 对齐。

【注意事项】

无

5.22 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。
stFrameRate	通道帧率控制参数。

【注意事项】

无

5.23 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】

无

5.24 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=100stSrc.u32Heigh=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.u32Heigh=1080, stDst.s32X=0,stDst.s32Y=0,stDst.u32Width=1280,stDst.u32Height=720。

【注意事项】

- 源图像大小由设置的通道参数决定
- 裁剪功能与裁剪缩放功能分别由参数stCropRect、stScaleRect决定
- 缩放倍数不能超过16倍

5.25 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

5.26 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】

无

5.27 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】

无

5.28 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

5.29 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。

5.30 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。

5.31 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。
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]。

【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]。

5.32 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】

无

5.33 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】

无

5.34 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。

5.35 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】

无

5.36 VENC_H265_DBLK_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。

5.37 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。

5.38 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。

5.39 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】

无。

6. 错误码

视频编码 API 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
0xA0048010	RK_ERR_VENC_BUF_FULL	缓冲区中数据满
0xA0048011	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

前言

概述

本文档对RK356X视频输出（VO）视频输出进行说明，并对VO相关API做介绍。

产品版本

芯片名称	内核版本
RK356X	4.19

读者对象

本文档（本指南）主要适用于以下工程师：

技术支持工程师

软件开发工程师

修订记录

版本号	作者	修改日期	修改说明
V0.1.0	许学辉 艾有军	2020-01-09	初始版本
V0.2.0	郑阳	2020-03-10	增加返回值说明
V0.2.1	许学辉	2021-04-25	增加举例说明
V0.2.2	许学辉	2021-08-03	增加UI绘制说明

1. 目录

2. 概述

视频输出（VO）模块是视频输出的封装。

3. RK356x VOP系统架构

RK3568 VOP支持3个显示输出接口，6个图层。

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

表2：RK356X 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

【备注】

* 图层只支持压缩格式，不支持非压缩格式

4. 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。

5. 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。

6. API参考

6.1 设备操作

6.1.1 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

6.1.2 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

6.1.3 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

6.1.4 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;
```

6.1.5 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。

6.1.6 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。

6.1.7 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。

6.1.8 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。

6.1.9 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。

6.1.10 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。

6.1.11 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。

6.1.12 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);
```

6.1.13 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。

6.2 Framebuffer设置

6.2.1 RK_MPI_VO_CreateGraphicsFrameBuffer

【描述】
创建图形层Framebuffer。
【语法】

```
RK_S32 RK_MPI_VO_CreateGraphicsFrameBuffer(int Width, int Hight, 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。
Hight	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  u32ImgeWidth;
    RK_U32  u32ImageHeight;
    RK_VOID *pMblk = RK_NULL;
    VO_FRAME_INFO_S stFrameInfo;
    RK_U32 u32BuffSize;

    u32ImgeWidth = 1920;
    u32ImageHeight = 1080;
    u32BuffSize = RK_MPI_VO_CreateGraphicsFrameBuffer(u32ImgeWidth,
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

6.2.2 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

6.2.3 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。

6.2.4 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

6.2.5 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

6.2.6 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。

6.3 图层操作

RK356x的VOP的图层分为视频层、图形成和鼠标层三种，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，默认绑定关系如下表。

显示输出设备	视频层	图形层	鼠标层
Dev0	Cluster0	Esmart0	Smart0
Dev1	Custer1	Esmart1	Smart1

6.3.1 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。

6.3.2 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。
- 当前发布的版本一个VoDev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。

【举例】
请参见RK_MPI_VO_EnableLayer

【相关主题】
RK_MPI_VO_UnBindLayer

6.3.3 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

6.3.4 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】

- 出于性能考虑，建议设置图层属性的显示帧率帧率为25。

【举例】
请参见RK_MPI_VO_EnableLayer的举例。

6.3.5 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。

6.3.6 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.enPixFormat = RK_FMT_YUV420SP;
    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;
```

6.3.7 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。

6.3.8 RK_MPI_VO_SetLayerPartitionMode

【描述】
设置视频层的分割模式。
【语法】

```
RK_S32 RK_MPI_VO_SetLayerPartitionMode(VO_LAYER VoLayer, VO_PART_MODE_E
enPartMode)
```

【参数】

参数名称	描述	输入/输出
VoLayer	视频输出视频层号。	输入
enPartMode	视频层处理的分割模式。	输入

【返回值】

返回值	描述
0	成功。

【备注】

- 该函数暂未支持。

6.3.9 RK_MPI_VO_GetLayerPartitionMode

【描述】
获取视频层的分割模式。
【语法】

```
RK_S32 RK_MPI_VO_GetLayerPartitionMode(VO_LAYER VoLayer, VO_PART_MODE_E
*penPartMode)
```

【参数】

参数名称	描述	输入/输出
VoDev	视频输出视频层号。	输入
penPartMode	视频层分割模式指针。	输出

【返回值】

返回值	描述
0	成功。

【备注】

- 该函数暂未支持。

6.3.10 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。

6.3.11 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。

6.3.12 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。

6.3.13 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。

6.3.14 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。

6.3.15 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

6.3.16 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;i++) {
        RK_MPI_VO_SetCursorPostion(VoLayer, x, y);
        x+=10;
        y+=10;
    }
```

6.4 通道操作

6.4.1 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] Enalbe chn failed\n", __func__);
            return RK_FAILURE;
        }
    }
```

【相关主题】
RK_MPI_DisableChn

6.4.2 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

6.4.3 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_GetChnAttr

6.4.4 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

6.4.5 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;
```

6.4.6 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

6.4.7 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。

6.4.8 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。

6.4.9 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。

6.4.10 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。

6.4.11 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 = pMblk;
    do {
        ret = RK_MPI_VO_SendFrame(VoVideoLayer, VoChn, pstVFrame, -1);
        if (ret == RT_OK) {
            break;
        } else {
            RK_MPI_VO_DestroyGraphicsFrameBuffer(pMblk);
            free(pstVFrame);
```

```
            break;
        }
    } while (1);
```

6.4.12 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

6.4.13 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

6.4.14 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_ResumeChn

6.4.15 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。

6.4.16 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

6.4.17 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。

6.4.18 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

6.4.19 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

6.4.20 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。

6.4.21 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。

6.4.22 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。

6.4.23 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);
    }
```

6.4.24 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

6.5 WBC回写操作

6.5.1 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

6.5.2 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

6.5.3 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

6.5.4 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。

6.5.5 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

6.5.6 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

6.5.7 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。

6.5.8 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

7. 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。

7.1 VO_DEV

【说明】
定义显示输出设备号。
【定义】

```
typedef RK_S32 VO_DEV;
```

7.2 VO_LAYER

【说明】
定义图层号。
【定义】

```
typedef RK_S32 VO_LAYER;
```

7.3 VO_CHN

【说明】
定义视频输出通道号。
【定义】

```
typedef RK_S32 VO_CHN;
```

7.4 VO_WBC

【说明】
定义回写通路号。
【定义】

```
typedef RK_S32 VO_WBC;
```

7.5 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
```

7.6 VO_MAX_LAYER_NUM

【说明】
定义图层最大个数。
【定义】

```
#define VO_MAX_LAYER_NUM              8
```

7.7 VO_MAX_CHN_NUM

【说明】
定义通道最大个数。
【定义】

```
#define VO_MAX_CHN_NUM              128
```

7.8 VO_MAX_WBC_NUM

【说明】
定义回写设备最大个数。
【定义】

```
#define VO_MAX_WBC_NUM              1
```

7.9 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。

7.10 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

7.11 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_FORMT_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_FORMT_AUTO	根据EDID按如下顺序选择，YCbCr444->YCbCr422->YCbCr420->RGB

7.12 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

7.13 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	Vo接口类型。
enIntfSync	Vo接口时序类型。
stSyncInfo	未使用。

7.14 VO_EDID_S

【说明】
EDID数据结构体
【定义】

```
typedef struct rk_VO_EDID_S {
    RK_BOOL             bEdidValid;
    RK_U32              u32Edidlength;
    RK_U8               u8Edid[512];
} VO_EDID_S;
```

【成员】

成员名称	描述
bEdidValid	EDID合法标志。
u32Edidlength	EDID数据长度。
u8Edid	EDID数据。

7.15 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

7.16 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);
```

7.17 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。

7.18 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。

7.19 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。

7.20 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 = 1,
    VO_LAYER_MODE_VIDEO = 2,
    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。

7.21 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 bDoubleFrame;
    RK_BOOL bClusterMode;
    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。
enPixFormat	视频层使用的像素格式。
bDoubleFrame	未使用。
bClusterMode	未使用。
enDstDynamicRange	未使用。

7.22 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 */
    RK_U32 u32BgAlpha;  /* Alpha of A = 0 in pixel format BGRA5551 */
} 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。

【备注】

- 通道数据格式为BGRA5551/RGBA5551时，需要填写u32FgAlpha和u32BgAlpha。

7.23 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。

7.24 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。

7.25 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。

7.26 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。

8. 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
0xA00b8010	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

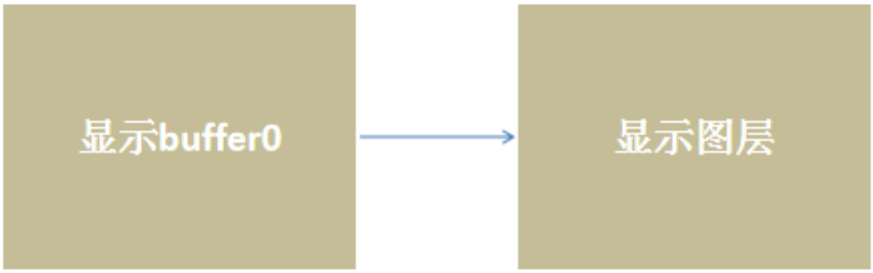
9. UI绘制说明

目前提供的API可以支持如下几种场景的UI绘制。

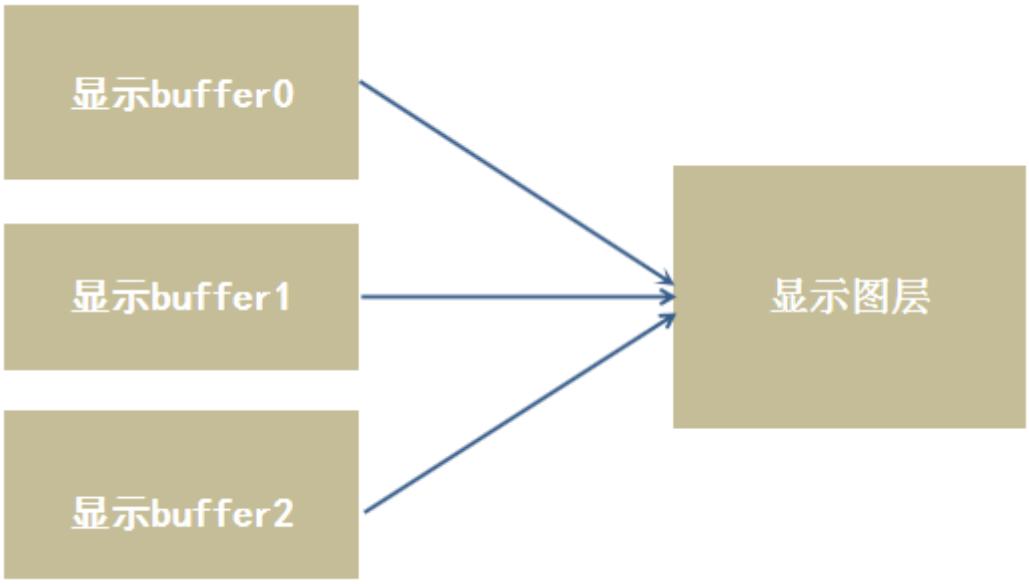
UI绘制方式	VOP图层	显示buffer模式	申请buffer方法	GPU参与合成	送VO方法	支持格式
1	UI图层	单buffer	RK_MPI_VO_CreateGraphicsFrameBuffer	否	RK_MPI_VO_SendLayerFrame	ARGB8888
2	UI图层	多buffer	RK_MPI_VO_CreateGraphicsFrameBuffer	否	RK_MPI_VO_SendLayerFrame	ARGB8888
3	UI图层	单buffer	RK_MPI_VO_GetGfxFrameBuffer	是	无需送帧函数	ARGB1555/ARGB8888
4	UI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。

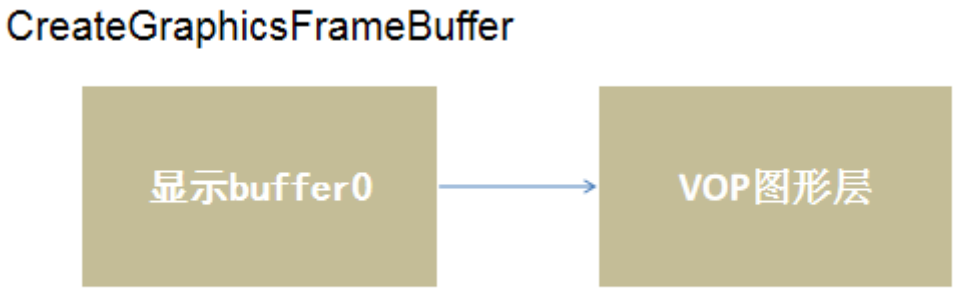


【方式1举例】

```
    VO_PUB_ATTR_S            stVoPubAttr;
    VO_VIDEO_LAYER_ATTR_S    stLayerAttr;
    VIDEO_FRAME_INFO_S       stVFrame;
    VO_LAYER VoLayer;
    VO_DEV VoDev;
    RK_VOID *pMblk = RK_NULL;
    RK_U32 s32Ret = RK_SUCCESS;
    RK_U32 u32BuffSize;

    /* alloc single buffer, and fill pMbk for UI data */
    u32BuffSize = RK_MPI_VO_CreateGraphicsFrameBuffer(1920, 1080,
RK_FMT_ARGB8888, &pMblk);
    /* 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);

    stVFrame.stVFrame.pMbBlk = pMblk;
    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。



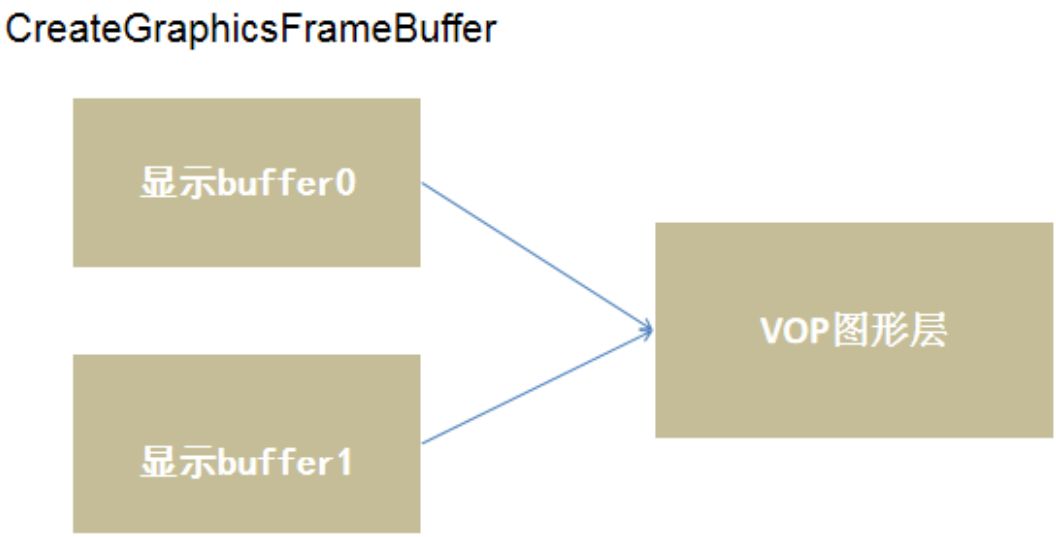
【方式2举例】

```
    VO_PUB_ATTR_S            stVoPubAttr;
    VO_VIDEO_LAYER_ATTR_S    stLayerAttr;
    VIDEO_FRAME_INFO_S       stVFrame;
    VO_LAYER VoLayer;
    VO_DEV VoDev;
    RK_VOID *pMblk = RK_NULL;
    RK_VOID *pMblk_480P = RK_NULL;
    RK_U32 s32Ret = RK_SUCCESS;
    RK_U32 u32BuffSize;
    RK_U32 count;

    /* alloc muti buffer, and fill pMbk for UI data */
    u32BuffSize = RK_MPI_VO_CreateGraphicsFrameBuffer(1920, 1080,
RK_FMT_ARGB8888, &pMblk);
    u32BuffSize = RK_MPI_VO_CreateGraphicsFrameBuffer(1920, 1080,
RK_FMT_ARGB8888, &pMblk_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 = pMblk;
            RK_MPI_VO_SendLayerFrame(VoLayer, &stVFrame);
        } else {
            stVFrame.stVFrame.pMbBlk = pMblk_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。



【方式3举例】

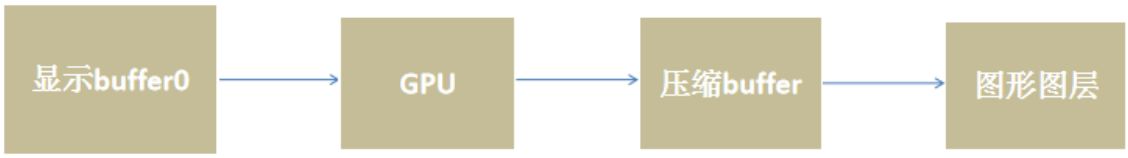
```
    VO_PUB_ATTR_S          stVoPubAttr;
    VO_FRAME_INFO_S        stVFrame;
    VO_LAYER VoLayer;
    VO_VIDEO_LAYER_ATTR_S    stLayerAttr;
    RK_U32 s32Ret = RK_SUCCESS;

    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

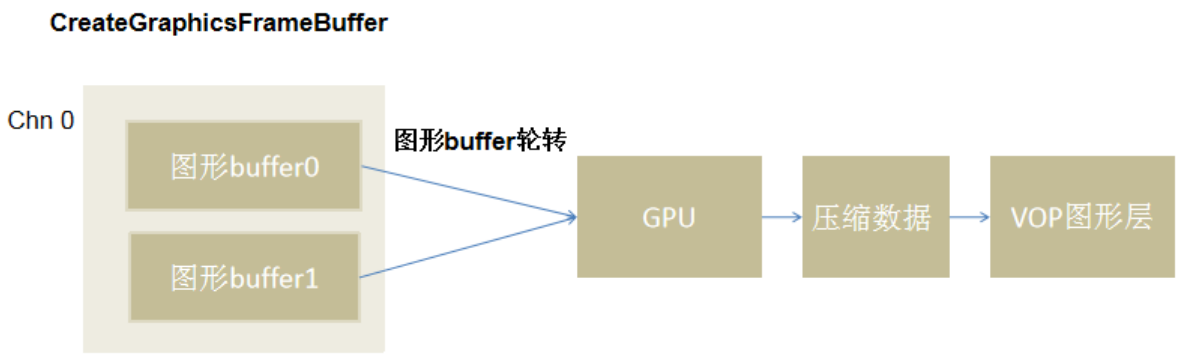


【方式4举例】

```
    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_G0 */
    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送帧函数，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。



【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;
/* Zoom to 4K */
```

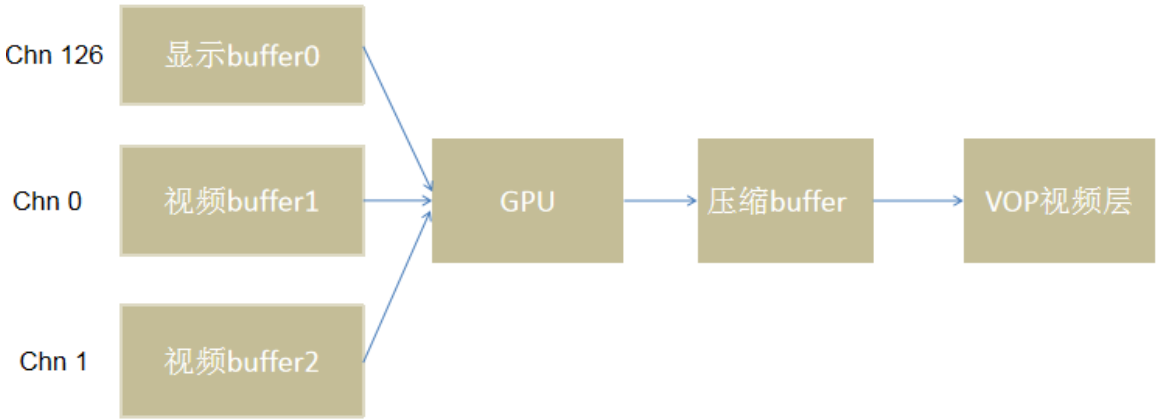
```
    RK_MPI_VO_SetChnAttr(VoLayer, 126, &stChnAttr);
    memset(stVFrame.pData, 0, stVFrame.u32Size);

    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。

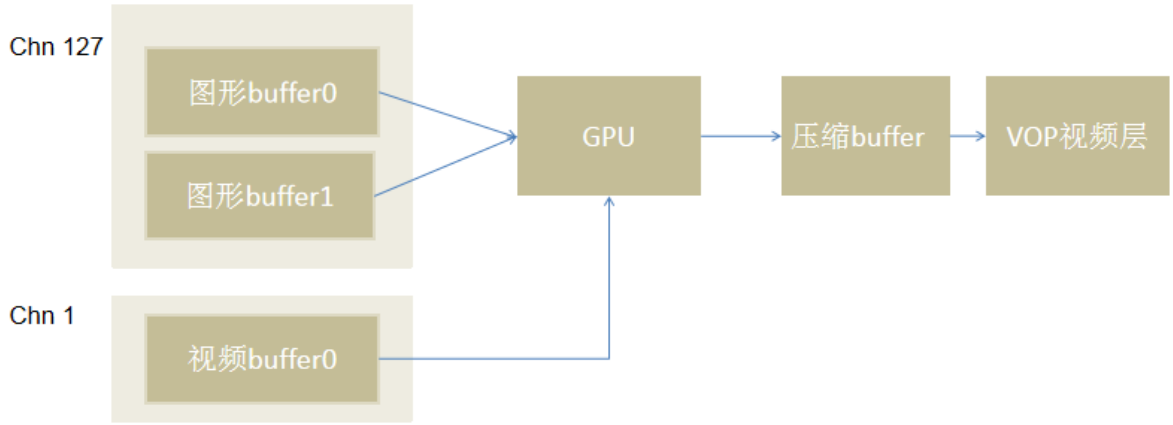
【方式6举例】

```
    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_G0 */
    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。

视频图形子系统

前言

概述

VGS（ Video Graphics Sub-System）视频图形子系统，主要是对输入的图像进行缩放、旋转、打OSD、打COVER、画线等操作。

产品版本

芯片名称	内核版本
RK356X	4.19

读者对象

本文档（本指南）主要适用于以下工程师：

技术支持工程师

软件开发工程师

修订记录

版本号	作者	修改日期	修改说明
v0.1.0	黄晓明	2021-1-4	初始版本
v0.2.0	黄晓明	2021-1-23	完善结构定义，新增马赛克接口
v0.2.1	黄晓明	2021-4-28	添加Crop接口
v1.0.0	黄晓明	2021-10-8	增加芯片规格说明、整理格式

1. 目录

2. 基本概念

2.1 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。

2.2 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。

2.3 HANDLE

任务句柄，标识一个 job。

3. 功能描述

VGS的功能有缩放、抠图、旋转、打osd、打cover、画线、打mosaic。

3.1 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。

3.2 旋转

VGS 支持对一幅图像进行0、90、180、270角度的旋转。

3.3 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。

3.4 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。

3.5 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。

3.6 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masaic操作时，输入和输出图像使用的是同一块内存。

4. 硬件规格

4.1 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。

5. 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。

5.1 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);
}
```

5.2 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

5.3 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。

5.4 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

5.5 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

5.6 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

5.7 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

5.8 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

5.9 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

5.10 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

5.11 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

5.12 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

5.13 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

5.14 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

6. 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。

6.1 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。

6.2 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

6.3 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。

6.4 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 = 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。

6.5 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, 0xFFFFFFF]。

6.6 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。

6.7 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, 0xFFFFFFF]。

6.8 VGS_COLOR_REVERT_MODE_E

【说明】

定义 VGS 上 OSD 反色模式的配置。

【定义】

```
ctypedef 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。

6.9 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。

6.10 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          enPixelFmt;
    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。

6.11 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。

6.12 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。

7. 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

前言

概述

本文档主要介绍 TDE 的 API 和数据类型。

产品版本

芯片名称	内核版本
RK356X	4.19

读者对象

本文档（本指南）主要适用于以下工程师：

技术支持工程师

软件开发工程师

修订记录

版本号	作者	修改日期	修改说明
v0.1.0	周弟东	2021-1-4	初始版本
v0.2.0	许丽明	2021-1-21	完善数据类型释义
v1.0.0	黄晓明	2021-10-8	增加芯片规格说明、整理格式

1. 目录

2. 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。

3. 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);
```

4. 硬件规格

4.1 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
输入格式为RK_FMT_YUV400SP	64	1	8
输出格式为RK_FMT_YUV400SP	4	2	8
RK_FMT_YUV420SP_10BIT RK_FMT_YUV422SP_10BIT	16	2	10
RK_FMT_RGB565 RK_FMT_RGBA5551	2	1	16
RK_FMT_BGR565 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操作区域坐标偏移均为2位对齐。
- 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。

5. 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。

5.1 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TDETDE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。

5.2 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。

5.3 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。

5.4 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_FALS：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。

5.5 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。

5.6 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_END_Job 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。

5.7 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。

5.8 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。

5.9 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

5.10 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。

5.11 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。
- 由于该操作直接将 u32FillData 填充在位图的指定区域内，如对RGBA8888 格式的图像的指定区域，根据指定颜色color进行绘制。color参数由高到低位分别是R，G，B，A，例如，红色：color = 0xff000000

5.12 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。

6. 数据类型

6.1 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

6.2 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

6.3 TDE_HANDLE

【说明】

定义job的句柄。

【定义】

```
typedef RK_S32 TDE_HANDLE;
```

【注意】

- 无

6.4 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;
} 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。

【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。

6.5 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。

6.6 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

6.7 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_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]

【注意】

- 无

7. TDE错误码

TDE APITDE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

文件标识：RK-SYS1-MPI-AUDIO

发布版本：V1.0.0

日　　期：2021.8

文件密级：□绝密　□秘密　□内部资料　■公开

瑞芯微电子股份有限公司

Rockchip Electronics Co., Ltd.

地址：　福建省福州市铜盘路软件园A区18号

网址：　www.rock-chips.com

客户服务电话：+86-4007-700-590

客户服务传真：+86-591-83951833

客户服务邮箱：fae@rock-chips.com

产品版本

芯片名称	内核版本
RK356X	4.19

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
v1.0.0	周弟东	2021-8-24	初始版本

1. 目录

2. 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。

3. 功能描述

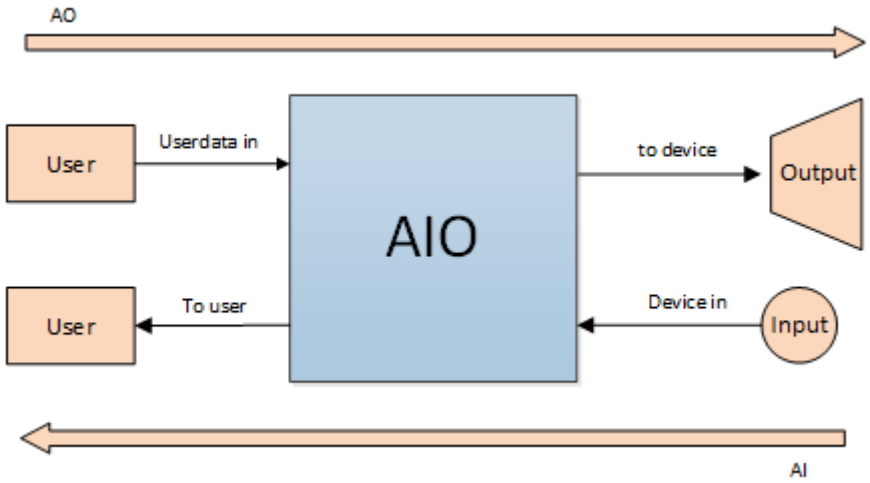
3.1 音频输入和输出

3.1.1 音频输入输出接口

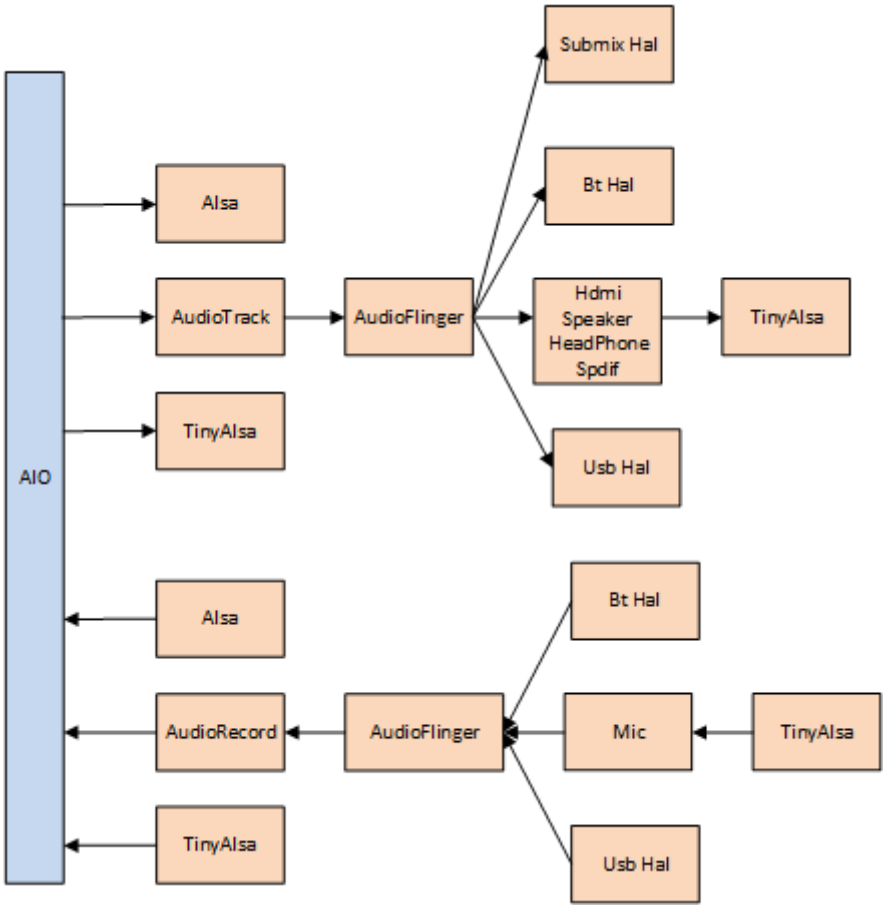
音频输入输出接口简称为AIO（Audio Input/Output）接口，用于对接音频框架Alsa（Linux平台）、Tinyalsa和AudioTrack/AudioRecord（Android平台），完成声音的录制和播放。
AIO 接口分为两种类型：只支持输入或只支持输出。

- 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。



3.1.2 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
```

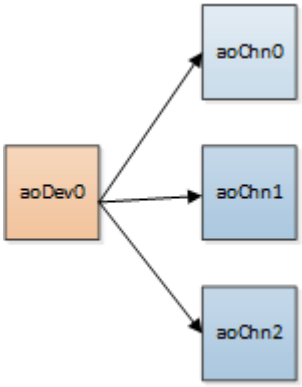
如上截图中，定义了3个声卡。序号为0的声卡为HDMI声卡，其只有放音功能；序号为1的声卡为RK809 Codec对应的声卡，其有放音和录音功能；序号为2的声卡为SPDIF声卡，其只有放音功能。

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)的序号一定一一对应的，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。

3.1.3 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。

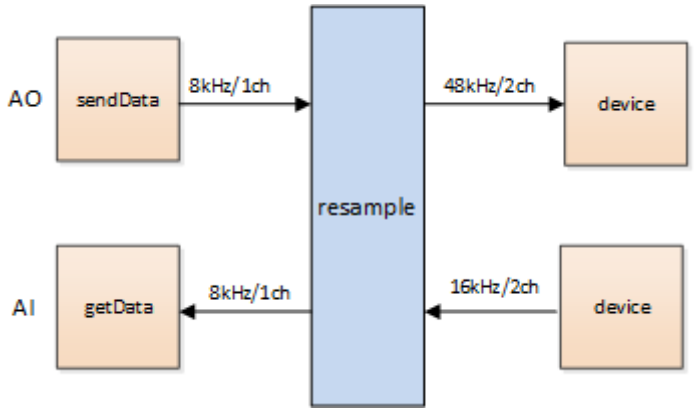


- 对AI，一个设备只支持一个通道。

3.1.4 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，且设置重采后参数为(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。



3.2 音频编码和解码

3.2.1 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。

3.2.2 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。

3.3 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。

4. API参考

4.1 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_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。

4.2 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。

4.3 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。

4.4 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。

4.5 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，其值为错误码。

【注意】

- 如果 AI 设备未启用，则返回未启用的错误码。
- 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。

4.6 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错误码。

4.7 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，其值为错误码。

【注意】

- 通道使用完成后，必须调用关闭AI通道。

4.8 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，其值为错误码。

【注意】

- 目前AI的回声抵消功能还未实现，pstAecFrm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。

4.9 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，其值为错误码。

【注意】

- 如果不需要释放回声抵消参考帧，pstAecFrm置为NULL即可。

4.10 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。

4.11 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，其值为错误码。

【注意】

- 无。

4.12 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);
```

4.13 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。

4.14 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，其值为错误码。

【注意】

- 在 AI 设备成功启用后再调用此接口。

4.15 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，其值为错误码。

【注意】

- 在 AI 设备成功启用后再调用此接口。

4.16 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，其值为错误码。

【注意】

- 清除设备属性前，需要先停止设备。

4.17 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。

4.18 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。

4.19 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_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。

4.20 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。

4.21 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。

4.22 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。

4.23 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，其值为错误码。

【注意】

- 如果 AO 设备未启用，则返回未启用的错误码。
- 禁用 AO 设备前必须先禁用设备下所有 AO 通道。

4.24 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错误码。

4.25 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，其值为错误码。

【注意】

- 通道使用完成后，必须调用此函数关闭AO通道。

4.26 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。

4.27 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);
```

4.28 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。

4.29 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。

4.30 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。

4.31 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。

4.32 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，其值为错误码。

【注意】

- 在 AO 通道成功启用后再调用此接口。

4.33 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。

4.34 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，其值为错误码。

【注意】

- 在 AO 通道成功启用后再调用此接口。

4.35 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。	输入

【返回值】

返回值	描述
0	成功。
非0	失败，其值为错误码。

【注意】

- 在 AO 通道成功启用后再调用此接口。

4.36 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，其值为错误码。

【注意】

- 在 AO 通道成功启用后再调用此接口。

4.37 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。

4.38 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，其值为错误码。

【注意】

- 在 AO 通道成功启用后再调用此接口。

4.39 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，其值为错误码。

【注意】

- 在 AO 通道成功启用后再调用此接口。

4.40 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。

4.41 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，其值为错误码。

【注意】

- 清除设备属性前，需要先停止设备。

4.42 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。

4.43 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。

4.44 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2.2.2 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。

4.45 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。

4.46 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。

4.47 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。

4.48 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。

4.49 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，请参见错误码。

【注意】

- 建议bInstant使用RK_FALSE。

4.50 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，请参见错误码。

【注意】

- 在ADEC通道成功启用后再调用此接口。

4.51 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。

4.52 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。

4.53 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。

4.54 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，请参见错误码。

【注意】

- 目前回声抵消未实现，pstAecFrm可置为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。

4.55 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。

4.56 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。

4.57 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，请参见错误码。

【注意】

- 在 AENC通道成功启用后再调用此接口。

4.58 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。

5. 数据类型

5.1 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_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。

5.2 AUDIO_DEV

【说明】
定义 AO 设备句柄。

【定义】

```
typedef RK_S32 AUDIO_DEV;
```

5.3 AO_CHN

【说明】
定义 AO 通道。

【定义】

```
typedef RK_S32 AO_CHN;
```

5.4 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
```

5.5 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
```

5.6 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。

5.7 AI_CHN

【说明】
定义 AI 通道。

【定义】

```
typedef RK_S32 AI_CHN;
```

5.8 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
```

5.9 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
```

5.10 AI_CHN_PARAM_S

【说明】
定义通道参数结构体。

【定义】

```
typedef struct rkAI_CHN_PARAM_S {
    RK_U32 u32UsrFrmDepth;
} AI_CHN_PARAM_S;
```

【成员】

成员名称	描述
u32UsrFrmDepth	音频帧缓存深度。

5.11 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
```

5.12 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
```

5.13 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];
} 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	处理完一个buffer数据所需的硬件中断次数。范围[2，300]，默认为4。
u32PtNumPerFrm	每次硬件中断处理音频数据的帧数。范围[128，256，512，1024，1536，2048]，默认为1024。
u32ChnCnt	支持的声道数目。
u8CardName[64]	声卡名字。

【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。

5.14 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 lenth per channel in frame, u32Len <= 0 mean eos */
    RK_BOOL             bBypassMbBlk;
    /* FALSE: copy, TRUE: MbBlock owned by internal */
} 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

【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。

5.15 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。

5.16 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。

5.17 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。

5.18 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 lenth, 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

5.19 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;
```

5.20 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，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。

5.21 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。

5.22 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。

5.23 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。

5.24 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。

5.25 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。
- 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。

- AUDIO_STREAM_S：定义音频码流结构体。

5.26 ADEC_CHN

【说明】
定义 ADEC 通道。

【定义】

```
typedef RK_S32 ADEC_CHN;
```

5.27 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
```

5.28 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 vaild 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。

5.29 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_U32 u32Channels;
    RK_U32 u32SampleRate;
} ADEC_ATTR_CODEC_S;
```

【成员】

成员名称	描述
u32Channels	码流声道。
u32SampleRate	码流采样率。

5.30 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;       /*decode mode*/
    ADEC_ATTR_CODEC_S  stAdecCodec;
    RK_U32             u32BufCount;
    MB_BLK             extraData;
    RK_U32             extraDataSize;
} ADEC_CHN_ATTR_S;
```

【成员】

成员名称	描述
enType	解码器codec id。
enMode	解码方式。
stAdecCodec	解码器属性参数。
u32BufCount	音频解码缓存个数。
extraData	解码器外部数据。
extraDataSize	解码器外部数据大小。

5.31 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。

5.32 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。

5.33 AENC_CHN

【说明】
定义 AENC 通道。

【定义】

```
typedef RK_S32 AENC_CHN;
```

5.34 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
```

5.35 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_U32 u32Channels;
    RK_U32 u32SampleRate;
    AUDIO_BIT_WIDTH_E enBitwidth;
} AENC_ATTR_CODEC_S;
```

【成员】

成员名称	描述
u32Channels	码流声道。
u32SampleRate	码流采样率。
enBitwidth	采样精度。

5.36 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;          /* audio codec id */
    AENC_ATTR_CODEC_S stAencCodec;     /* channel count & samplerate */
    RK_U32            u32BufCount;      /* encode buffer count */
    MB_BLK            extraData;       /* encode key parameters */
    RK_U32            extraDataSize;   /* key parameters size */
} AENC_CHN_ATTR_S;
```

【成员】

成员名称	描述
enType	解码器codec id。
stAencCodec	解码器属性参数。
u32BufCount	音频编码缓存个数。
extraData	解码器外部数据。
extraDataSize	解码器外部数据大小。

6. 错误码

6.1 音频输入错误码

音频输入API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

6.2 音频输出错误码

音频输出API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

6.3 音频解码错误码

音频解码API错误码如下所示。

错误代码	宏定义	描述
0xA00D8001	RK_ERR_ADEC_INVALID_DEVID	音频输出设备号无效
0xA00D8002	RK_ERR_ADEC_INVALID_CHNID	音频输出通道号无效
0xA00D8003	RK_ERR_ADEC_ILLEGAL_PARAM	音频输出参数设置无效
0xA00D8004	RK_ERR_ADEC_EXIST	音频输出设备或通道没使能
0xA00D8005	RK_ERR_ADEC_UNEXIST	输出空指针错误
0xA00D8006	RK_ERR_ADEC_NULL_PTR	音频输出设备属性未设置
0xA00D8007	RK_ERR_ADEC_NOT_CONFIG	操作不被支持
0xA00D8008	RK_ERR_ADEC_NOT_SUPPORT	操作不允许
0xA00D8009	RK_ERR_ADEC_NOT_PERM	系统内存不足
0xA00D800C	RK_ERR_ADEC_NOMEM	音频输出缓存不足
0xA00D800D	RK_ERR_ADEC_NOBUF	音频输出缓存为空
0xA00D800E	RK_ERR_ADEC_BUF_EMPTY	音频输出缓存为满
0xA00D800F	RK_ERR_ADEC_BUF_FULL	音频输出缓存为满
0xA00D8010	RK_ERR_ADEC_SYS_NOTREADY	音频输出系统未初始化
0xA00D8040	RK_ERR_ADEC_DECODER_ERR	音频输出系统忙
0xA00D8041	RK_ERR_ADEC_BUF_LACK	AO VQE 处理错误

6.4 音频编码错误码

音频解码API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

区域管理

前言

概述

用户一般都需要在视频中叠加 OSD 用于显示一些特定的信息（如：通道号、时间戳等），必要时还会填充色块。这些叠加在视频上的 OSD 和遮挡在视频上的色块统称为区域。REGION 模块，用于统一管理这些区域资源。

区域管理可以实现区域的创建，并叠加到视频中或对视频进行遮挡。例如，实际应用中，用户通过创建一个区域，通过RK_MPI_RGN_AttachToChn，将该区域叠加到某个通道（如 VENC 通道）中。在通道进行调度时，则会将 OSD 叠加在视频中。一个区域支持通过设置通道显示属性接口指定到多个通道中（如：多个 VENC 通道），且支持在每个通道的显示属性（如位置、层次、透明度等）都不同。

产品版本

芯片名称	内核版本
RK356X	4.19

读者对象

本文档（本指南）主要适用于以下工程师：

技术支持工程师

软件开发工程师

修订记录

版本号	作者	修改日期	修改说明
v0.1.0	许丽明	2021-02-02	初始版本
v0.2.0	许丽明	2021-07-28	增加overlay QP保护，增加部分释义
v0.3.0	许丽明	2021-07-29	增加直接绘制接口
v0.4.0	李鑫煌	2021-10-02	增加画板设置说明

1. 目录

2. 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。
  - COVER：视频遮挡区域，其中区域支持纯色块遮挡。
  - MOSAIC：马赛克遮挡区域，支持精度调节。
- 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。

2.1 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, RK_VENC_MAX_CHN_NUM - 1]
OVERLAY	VO	[0, RK_VO_MAX_LAYER_NUM - 1]	[0, RK_VO_MAX_CHN_NUM - 1]
COVER	VENC	0	[0, RK_VENC_MAX_CHN_NUM - 1]

- 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。

2.2 表1-2 RK356X region 支持的功能

支持的功能	OVERLAY	OVERLAY	COVER	COVER
支持的模块	VO	VENC	VO	VENC
像素格式	BGRA8888、BGRA5551	BGRA8888、BGRA5551、BGRA4444	RGB888	RGB888
叠加层次	支持	N/A	支持	N/A
位图填充	支持	支持	N/A	N/A
叠加透明度	支持	N/A	N/A	N/A
前景alpha范围	0~255	N/A	N/A	N/A
背景alpha范围	0~255	N/A	N/A	N/A
反色	N/A	N/A	N/A	N/A
QP保护	N/A	支持	N/A	N/A

3. 使用步骤介绍

3.1 OVERLAY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_GetCanvasInfo 获取画布信息。
- 步骤3：将位图数据写入画布信息中。
- 步骤4：调用RK_MPI_RGN_UpdateCanvas 更新画布。
- 步骤5：调用RK_MPI_RGN_AttachToChn 将画布绑定到通道特定区域上。
- 更新画布信息时重复步骤2～步骤4。
- 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。

3.2 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。

4. 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。

5. 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。

5.1 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_HANDLE_MAX)。	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。

【相关主题】

- RK_MPI_RGN_Destroy
- RK_MPI_RGN_GetAttr
- RK_MPI_RGN_SetAttr

5.2 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_HANDLE_MAX)。	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

5.3 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_HANDLE_MAX)。	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

5.4 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_HANDLE_MAX)。	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

5.5 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_HANDLE_MAX)。	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

5.6 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_HANDLE_MAX)。	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。

【相关主题】

- 无

5.7 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_HANDLE_MAX)。	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。

【相关主题】

- 无

5.8 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_HANDLE_MAX)。	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

5.9 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_HANDLE_MAX)。	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

5.10 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_HANDLE_MAX)。	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

5.11 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_HANDLE_MAX)。	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，否则将会有不必要的系统开销。
- 本接口仅支持 OVERLAY 类型的区域。

【相关主题】

- 无

6. 数据类型

6.1 RGN_MIN_WIDTH

【说明】

定义区域最小宽度。

【定义】

RK356X：

```
#define RGN_MIN_WIDTH                16
```

【注意】

- 无

6.2 RGN_MIN_HEIGHT

【说明】

定义区域最小高度。

【定义】

RK356X：

```
#define RGN_MIN_HEIGHT               16
```

【注意】

- 无

6.3 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

6.4 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

6.5 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

6.6 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

6.7 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

6.8 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

6.9 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

6.10 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

6.11 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

6.12 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

6.13 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

6.14 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

6.15 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

6.16 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

6.17 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

6.18 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

6.19 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

6.20 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

6.21 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

6.22 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

6.23 RGN_ALIGN

【说明】

定义区域对齐方式。

【定义】

RK356X：

```
#define RGN_ALIGN          16
```

【注意】

- 无

6.24 RGN_HANDLE_MAX

【说明】

定义最大的RGN句柄个数。

【定义】

```
#define RGN_HANDLE_MAX                 128
```

【注意】

- 无

6.25 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

6.26 RGN_HANDLE

【说明】

定义区域句柄。

【定义】

```
typedef RK_U32 RGN_HANDLE;
```

【注意】

- 无

6.27 RGN_TYPE_E

【说明】

定义区域类型。

【定义】

```
typedef enum rkRGN_TYPE_E {
    OVERLAY_RGN = 0,
    COVER_RGN,
    MOSAIC_RGN,
    RGN_BUTT
} RGN_TYPE_E;
```

【成员】

成员名称	描述
OVERLAY_RGN	视频叠加区域。
COVER_RGN	视频遮挡区域。
MOSAIC_RGN	MOSAIC 视频区域。

【注意】

- 无

6.28 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。

6.29 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];
} 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（alhpa/24-31bit）、红色（red/16-23bit）、绿色（green/8-15bit）、蓝色（blue/0-7bit）。

【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。

6.30 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;
} 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。

【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。（VENC不支持Alpha值设定）
- 0 表示全透明；255 表示不透明。

6.31 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RGB888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】

- 无

6.32 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

6.33 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 {
    RECT_S stRect;
    MOSAIC_BLK_SIZE_E enBlkSize;
    RK_U32 u32Layer;
} MOSAIC_CHN_ATTR_S;
```

【成员】

成员名称	描述
stRect	马赛克区域。
enBlkSize	马赛克显示类型。
u32Layer	区域层次。

【注意】

- 无

6.34 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

6.35 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;
} 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。

【注意】

- 无

6.36 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

6.37 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

6.38 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。

6.39 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

7. RGN错误码

区域管理 API RGN错误码如表 1-3所示。

7.1 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

Dump调试信息说明

前言

概述

调试信息采用额外的dumpsys 的应用来进行收集的， 可以实时反应当前系统的运行状态，dump的信息可以利用来进行问题定位和分析。

产品版本

芯片名称	内核版本
RK356X	4.19

读者对象

本文档（本指南）主要适用于以下工程师：

技术支持工程师

软件开发工程师

修订记录

版本号	作者	修改日期	修改说明
v0.1.0	黄晓明	2021-2-23	初始版本
v0.1.1	许丽明	2021-2-25	加入vpss/rgn/sys模块调试
v0.1.2	周弟东	2021-2-25	加入adec/ao/ai/aenc模块调试
v0.1.3	杨文杰	2021-2-27	加入vdec模块调试
v0.1.4	许丽明	2021-2-27	加入venc模块调试
v0.1.5	李鑫煌	2021-4-13	加入vi模块调试
v0.1.6	许丽明	2021-4-26	更新sys/vpss的打印信息 增加VPSS数据录制开关
v0.2.0	李鑫煌	2021-9-24	更新venc模块调试信息

1. 目录

运行命令

dumpsys 模块名

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VB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
all	记录所有注册模块的配置和状态信息

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. VPSS

【调试信息】

```
[root@RK356X:/userdata]# ./dumpsys vpss
-------------------------------------------------------------------------------
DUMP OF SERVICE vpss:
---------------------- vpss group attr -----------------------------
grp_id    max_w     max_h     pixel_format   dym_range src_rate  dst_rate
is_compress
0         4096      4096      image:nv12     0         25        8              Y
1         4096      4096      image:nv12     0         25        8              Y

---------------------- vpss channel attr -----------------------------
grp_id    chn_id    mode      width     height    pixel_format   is_compress
src_rate  dst_rate  depth     align     mirror    flip
0         0         PAST      704       576       image:nv12     N
-1        -1        0         16        N         N
0         1         USER      704       576       image:nv12     N
-1        -1        0         16        N         N
1         0         PAST      704       576       image:nv12     N
-1        -1        0         16        N         N
```

```
1         1         USER      704       576       image:nv12     N
-1        -1        0         16        N         N

---------------------- vpss group crop info ----------------------------
grp_id    crop_en   coor_type x         y         width     height
0         N         RATIO     0         0         0         0
1         N         RATIO     0         0         0         0

---------------------- vpss chn crop info ----------------------------
grp_id    chn_id    crop_en   coor_type x         y         width     height
0         0         Y         RATIO     250       250       500       500
0         1         N         RATIO     0         0         0         0
1         0         Y         RATIO     250       250       500       500
1         1         N         RATIO     0         0         0         0

---------------------- vpss group pic queue ----------------------------
grp_id    delay     backup    left_input_cnt left_output_cnt
0         0         Y         0              0
1         0         Y         0              0

---------------------- vpss group pic info ----------------------------
grp_id    width     height    vir_w     vir_h     pix_format     dyn_range
compress
0         0         0         0         0         image:nv12     0         N
1         0         0         0         0         image:nv12     0         N

---------------------- 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
0         0         0
0         1         0
1         0         0
1         1         0

---------------------- vpss chn work status ----------------------------
grp_id         chn_id         get_frm_cnt    get_frm_rate   rel_frm_cnt
left_input_cnt left_output_cnt
0              0              0              0.00           0              1
         0
0              1              0              0.00           0              0
         0
1              0              0              0.00           0              2
         0
1              1              0              0.00           0              0
         0

--------------------------------------------------------------------------------
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 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。
vpss channel attr VPSS模块通道属性	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。
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)。

参数模块	参数名	描述
	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。
	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]。

参数模块	参数名	描述
	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。
	left_input_cnt	待处理的组输入缓存帧数。
	left_output_cnt	待取走的组输出缓存帧数。
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 output resolution VPSS 通道输出分辨率	grp_id	GRP ID 号。有效范围：[0,VPSS_MAX_GRP_NUM)。

参数模块	参数名	描述
	chn_id	通道 ID 号。有效范围：[0,VPSS_MAX_CHN_NUM]。
	width	目标图像的宽度。
	height	目标图像的高度。
	vir_w	目标图像的虚宽。
	vir_h	目标图像的虚高。
	pix_format	目标图像的像素格式
	dyn_range	目标图像的动态范围。
	compress	目标图像压缩模式： Y：压缩； N：非压缩；
	send_ok	成功发送的图像数量。
	frm_rate	通道输出的实时帧率。
vpss chn rotation attr 通道旋转的信息。	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。有效范围：[0, 360]。
vpss chn work status 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。
	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。
	left_input_cnt	待处理的通道输入缓存帧数。
	left_output_cnt	待取走的通道输出缓存帧数。

【调试命令】

- 数据录制

  组数据录制（打开后录制**RK_MPI_VPSS_GetGrpFrame**得到的数据）

  打开组数据录制（例如打开组0的录制，并录制文件至/dev/grp_out_0.bin）

```
./dumpsys vpss open_record_grp 0 /dev/
```

关闭组数据录制（例如关闭组0的录制）

```
./dumpsys vpss close_record_grp 0
```

通道数据录制（打开后录制RK_MPI_VPSS_GetChnFrame得到的数据）

打开通道数据录制（例如打开组0通道0的录制，并录制文件至/dev/chn_out_0_0.bin）

```
./dumpsys vpss open_record_chn 0 0 /dev/
```

关闭通道数据录制（例如关闭组0通道0的录制）

```
./dumpsys vpss close_record_chn 0 0
```

4. 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

-------------------------------------------------------------------------------
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。

参数模块	参数名	描述
	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。
	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。

参数模块	参数名	描述
	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。
	blk_size	MOSAIC 精度。取值范围{8, 16, 32}
	layer	在该通道显示的层次。

5. VGS

【调试信息】

```
dumpsys vgs
-------------------------------------------------------------------------------
DUMP OF SERVICE vgs:
---------------------- module params ----------------------------
g_max_job_num       g_max_task_num
 100                 200
---------------------- recent job info1 ----------------------------
seq_no    job_hdl   state     task_num  in_size   out_size  cost_time hw_time
0         0         proced    18        6220800   5990400   5225087   0
1         0         proced    18        6220800   5990400   5061605   0
2         0         endjob    18        6220800   5990400   0         0
---------------------- recent job info2 ----------------------------
seq_no    crop      cover     mosaic    osd
0         1         1         1         1
1         1         1         1         1
2         1         1         1         1
---------------------- max waste time recent job info1 -------------------------
--
seq_no    job_hdl   state     task_num  in_size   out_size  cost_time hw_time
0         0         proced    18        6220800   5990400   5225087   0
---------------------- max waste time recent job info2 -------------------------
--
seq_no    crop      cover     mosaic    osd
```

```
0         1         1         1         1
---------------------- vgs job status -----------------------------
success   fail      cancel    all_job_num     free_num  begin_num busy_num
procing_num
2         0         0         100             0         0         1         0
---------------------- vgs task status -----------------------------
success   fail      cancel    all_task_num    free_num  busy_num
36        0         0         200             0         18
--------------------------------------------------------------------------------
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
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。
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）
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。
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

参数		描述
	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

6. ADEC

【调试信息】

```
dumpsys adec
-------------------------------------------------------------------------------
DUMP OF SERVICE adec:
---------------------- adec 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

7. AENC

【调试信息】

```
dumpsys 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

8. 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    peroid_count   peroid_size    card_name
0         48000     2              16bit          44100     stereo         16bit
         2         0              4              1024           pcm.card0
---------------------- ao dev extend status ----------------------------
ao_dev  track_mode  mute   volume
0         0         N         100
---------------------- ao chn attr -----------------------------
ao_dev    ao_chn    state     resample_open  in_rate   out_rate
0         0         start     Y              44100     48000
-------------------------------------------------------------------------------
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
音频输出设备属性	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]
	expand_flag	扩展标志
	peroid_count	处理完一个buffer数据所需的硬件中断次数
	peroid_size	每次硬件中断处理音频数据的帧数
	card_name	打开声卡名
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 Y：关闭
	volume	音量值
音频输出通道信息	ao_dev	设备号
	ao_chn	通道号
	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 Y：关闭

参数模块	参数名	描述
	in_rate	重采样时，输入数据采样率
	out_rate	重采样时，输出数据采样率

9. 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    peroid_count   peroid_size    card_name
0         16000     2              16bit          16000     stereo         16bit
         2         0              4              1024           pcm.record0
---------------------- ai dev extend status ----------------------------
ai_dev  track_mode
0         0
---------------------- ai chn attr -----------------------------
ai_dev    ai_chn    state     resample_open  in_rate   out_rate
0         0         start     Y              16000     16000
-------------------------------------------------------------------------------
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]
	expand_flag	扩展标志
	peroid_count	处理完一个buffer数据所需的硬件中断次数
	peroid_size	每次硬件中断处理音频数据的帧数
	card_name	打开声卡名
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
	mute	静音功能是否开启 Y：开启 Y：关闭
	volume	音量值
音频输入通道信息	ai_dev	设备号
	ai_chn	通道号
	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 Y：关闭

参数模块	参数名	描述
	in_rate	重采样时，输入数据采样率
	out_rate	重采样时，输出数据采样率

10. 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     1     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 stat -----------------------------
pipe  chn   width     height    enabled   lost  framerate vbfail    freeze
0     2     720       480       1         1     0         6         0

-------------------------------------------------------------------------------
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
	lost	通道丢帧数
	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

11. VO

【调试信息】

```
[root@RK356X:/]# dumpsys vo
--------------------------------------------------------------------------------
DUMP OF SERVICE vo:
```

```
-----------DEV CONFIG------------
DevId   DevEn   InfType InfSync
0       N       Unkown  Unkown
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    DispX   DispY   DispW   DispH   FrmRt
BufLens
0       N       NV12    0       0       0       0       0       0       0       0
1       N       NV12    0       0       0       0       0       0       0       0
2       Y       BGR24   1024    768     0       0       1024    768     25      3
3       N       NV12    0       0       0       0       0       0       0       0
4       N       NV12    0       0       0       0       0       0       0       0
5       Y       BGRA32  1024    768     0       0       1024    768     25      3
6       N       NV12    0       0       0       0       0       0       0       0
7       N       NV12    0       0       0       0       0       0       0       0
LAYER clu1 CHANNEL STATUS:
ChnId   Prio    X       Y       W       H       ChnFrt  FgAlpha BgAlpha Show
Pause   Step    Cache   RevCnt
0       0       0       0       512     384     25      0       0       Y       N
      N       1       9
1       1       512     0       512     384     25      0       0       Y       N
      N       1       9
2       2       0       384     512     384     25      0       0       Y       N
      N       1       9
3       3       512     384     512     384     25      0       0       Y       N
      N       1       9
LAYER esm1 CHANNEL STATUS:
ChnId   Prio    X       Y       W       H       ChnFrt  FgAlpha BgAlpha Show
Pause   Step    Cache   RevCnt
0       0       0       0       1024    768     25      128     0       Y       N
      N       1       83
1       1       0       0       1024    768     25      128     0       Y       N
      N       1       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
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
	BufLens	图层画布缓存个数
图层通道状态	ChnId	通道设备号
	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
	Show	通道是否可见：Y-可见；N-隐藏
	Pause	通道播放状态：Y-暂停；N-播放

参数模块	参数名	描述
	Step	通道是否单帧步进：Y-使能；N-关闭
	Cache	通道缓存帧数
	RevCnt	通道接收过的帧数，禁用/使能通道会清零
回写状态	WbcId	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

12. 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         0         proced    1         345600    921600    2813      0
1         0         proced    1         345600    921600    2837      0
2         0         proced    1         345600    921600    2825      0
3         0         proced    1         345600    921600    2839      0
4         0         begin     0         0         0         0         0
5         0         proced    1         345600    921600    2827      0
6         0         proced    1         345600    921600    2827      0
7         0         proced    1         345600    921600    2837      0
---------------------- recent job info2 ----------------------------
seq_no    copy      fill      resize    bitblit   rotate
0         0         0         1         0         0
1         0         0         1         0         0
2         0         0         1         0         0
3         0         0         1         0         0
4         0         0         0         0         0
5         0         0         1         0         0
6         0         0         1         0         0
```

```
7         0         0         1         0         0
---------------------- max waste time recent job info1 --------------------------
--
seq_no    job_hdl   state     task_num  in_size   out_size  cost_time hw_time
0         0         proced    1         345600    921600    5865      0
---------------------- max waste time recent job info2 --------------------------
--
seq_no    copy      fill      resize    bitblit   rotate
0         0         0         1         0         0
---------------------- tde job status -----------------------------
success   fail      cancel    all_job_num     free_num  begin_num busy_num
procing_num
68        0         0         128             0         1         0         0
---------------------- tde task status ----------------------------
success   fail      cancel    all_task_num    free_num  busy_num
68        0         0         200             0         1
-------------------------------------------------------------------------------
```

【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
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。
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）
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。
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
	procing_num	正在进行硬件处理的任务数

参数		描述
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

13. VDEC

【调试信息】

```
[root@RK356X:/userdata]# ./dumpsys vdec
-------------------------------------------------------------------------------
DUMP OF SERVICE vdec:
---------------------- MODULE PARAM -----------------------------
vdec_max_chn_num
64
---------------------- CHN COMM ATTR & PARAM ----------------------------
id              type            width           height          vir_width
vir_height      dispMode        state
0               image:h264      704             576             704             576
          1               start
1               image:h264      704             576             704             576
          1               start
2               image:h264      704             576             704             576
          1               start
---------------------- CHN VIDEO ATTR & PARAMS ----------------------------
id              compress
0               1
1               1
2               1
---------------------- 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_num
0               5086            5081            0               0
1               5084            5079            0               0
2               5085            5080            0               0
--------------------------------------------------------------------------------
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 para	vdec_max_chn_num	VDEC 支持的最大解码通道数。
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start，stop：通道stop）
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）
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
	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

14. VENC

【调试信息】

```
[root@RK356X:/userdata]# ./dumpsys venc
-------------------------------------------------------------------------------
DUMP OF SERVICE venc:
---------------------- venc module_param ----------------------------
venc_max_chn_num
16

---------------------- 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 info -----------------------------
id  start_qp  step_qp  max_qp  min_qp  max_i_qp  min_i_qp  delt_ip_qp  qfactor
0   26        8        51      10      46        24        2           0

---------------------- venc chn roi info ----------------------------
id        index     is_intra  abs_qp    qp        x         y         width
height

---------------------- venc chn dump status -----------------------------
id  in_fps    out_fps   seq     snap_set  afbc_mode
0   24.05     23.98     207     10000     0x0

---------------------- 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          0/0

---------------------- venc chn dump h264 config ----------------------------
id  intra_pred  tran_mode  chroma_qp  entropy  cabac_init  dblk_dis  dblk_a
dblk_b sao_luma  sao_cr
0   0           0          -6         CABAC    0           0         0       0
   1         1
pu_sis_en
1

---------------------- venc chn dump h265 config ----------------------------
id  cb_qp  cr_qp  scaling_list  dblk_dis  dblk_a  dblk_b

---------------------- 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                0

-------------------------------------------------------------------------------
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。
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。（8: H264 9：mjpeg 12:h264 15:jpeg）
	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 info	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。

参数		描述
	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。
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 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。
	afbc_mode	编码输入压缩模式。 0x0：无压缩; 0x100000：AFBC V1; 0x200000：AFBC V2。
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。

参数		描述
	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 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。
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。
	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。

参数		描述
	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。
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。

15. 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:
```

```
ChnId   Prio    X       Y       W       H       ChnFrt  Show    Pause   Step
Cache
0       1       0       0       480     270     25      Y       N       N       8
LAYER clu1 CHANNEL STATUS:
ChnId   Prio    X       Y       W       H       ChnFrt  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。