

Rockchip RKMedia Development Guide

文件标识: RK-KF-YF-382

发布版本: V1.1.1

日期: 2020-12-08

文件密级: ☐绝密 ☐秘密 ☐内部资料 ☒公开

免责声明

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

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

商标声明

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

本文档可能提及的其他所有注册商标或商标, 由其各自拥有者所有。

版权所有 © 2020 瑞芯微电子股份有限公司

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

瑞芯微电子股份有限公司

Rockchip Electronics Co., Ltd.

地址: 福建省福州市铜盘路软件园A区18号

网址: www.rock-chips.com

客户服务电话: +86-4007-700-590

客户服务传真: +86-591-83951833

客户服务邮箱: fae@rock-chips.com

前言

概述

本文主要描述了RKMedia 媒体开发参考。

产品版本

芯片名称	内核版本
RK1126/RK1109	Linux V4.19

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
V0.0.1	范立创 / 余永镇	2020-08-31	初始版本
V1.0.0	林刘迪铭	2020-09-03	增加数据类型和错误码，关联链接
V1.1.0	范立创 / 陈茂森	2020-11-18	<p>1.增加API说明：</p> <ul style="list-style-type: none"> (1) RK_MPI_MB_CreateBuffer (2) RK_MPI_MB_CreateImageBuffer (3) RK_MPI_MB_CreateAudioBuffer (4) RK_MPI_MB_ConvertToImgBuffer (5) RK_MPI_MB_ConvertToAudBuffer (6) RK_MPI_MB_SetSzie (7) RK_MPI_MB_SetTimestamp (8) RK_MPI_MB_GetFlag (9) RK_MPI_MB_GetTsvcLevel (10) RK_MPI_MB_IsViFrame (11) RK_MPI_MB_GetImageInfo (12) RK_MPI_MB_BeginCPUAccess (13) RK_MPI_MB_EndCPUAccess (14) RK_MPI_VI_StartRegionLuma (15) RK_MPI_VI_StopRegionLuma (16) RK_MPI_VENC_GetVencChnAttr (17) RK_MPI_VENC_SetVencChnAttr (18) RK_MPI_VENC_CreateJpegLightChn (19) RK_MPI_VENC_GetRcParam (20) RK_MPI_VENC_SetResolution (21) RK_MPI_VENC_GetRoiAttr (22) RK_MPI_VENC_RGN_SetPaletteId (23) RK_MPI_VENC_GetFd (24) RK_MPI_VENC_QueryStatus (25) RK_MPI_ALGO_MD_EnableSwitch (26) RK_MPI_ALGO_OD_EnableSwitch (27) RK_MPI_VO_SetChnAttr (28) RK_MPI_AI_StartStream (29) RK_MPI_AO_QueryChnStat (30) RK_MPI_AO_ClearChnBuf (31) RK_MPI_SYS_StartGetMediaBuffer (32) RK_MPI_SYS_StopGetMediaBuffer (33) RK_MPI_VDEC_CreateChn (34) RK_MPI_VDEC_DestroyChn <p>2.修改API: RK_MPI_VO_CreateChn</p> <p>3.新增示例说明板块</p>
V1.1.1	陈茂森	2020-12-08	修正VI工作模式个数错误

目录

Rockchip RKMedia Development Guide

1. 系统概述
 - 1.1 概述
 - 1.2 系统架构
 - 1.3 系统资源数目表
2. 系统控制
 - 2.1 概述
 - 2.2 功能描述
 - 2.2.1 系统绑定
 - 2.3 API参考
 - 2.3.1 RK_MPI_SYS_Init
 - 2.3.2 RK_MPI_SYS_DumpChn
 - 2.3.3 RK_MPI_SYS_Bind
 - 2.3.4 RK_MPI_SYS_UnBind
 - 2.3.5 RK_MPI_SYS_RegisterEventCb
 - 2.3.6 RK_MPI_SYS_RegisterOutCb
 - 2.3.7 RK_MPI_SYS_SendMediaBuffer
 - 2.3.8 RK_MPI_SYS_StartGetMediaBuffer
 - 2.3.9 RK_MPI_SYS_StopGetMediaBuffer
 - 2.3.10 RK_MPI_SYS_GetMediaBuffer
 - 2.3.11 RK_MPI_MB_CreateBuffer
 - 2.3.12 RK_MPI_MB_CreateImageBuffer
 - 2.3.13 RK_MPI_MB_CreateAudioBuffer
 - 2.3.14 RK_MPI_MB_ConvertToImgBuffer
 - 2.3.15 RK_MPI_MB_ConvertToAudBuffer
 - 2.3.16 RK_MPI_MB_ReleaseBuffer
 - 2.3.17 RK_MPI_MB_GetPtr
 - 2.3.18 RK_MPI_MB_GetFD
 - 2.3.19 RK_MPI_MB_GetSize
 - 2.3.20 RK_MPI_MB_SetSzie
 - 2.3.21 RK_MPI_MB_GetModeID
 - 2.3.22 RK_MPI_MB_GetChannelID
 - 2.3.23 RK_MPI_MB_GetTimestamp
 - 2.3.24 RK_MPI_MB_SetTimestamp
 - 2.3.25 RK_MPI_MB_GetFlag
 - 2.3.26 RK_MPI_MB_GetTsvcLevel
 - 2.3.27 RK_MPI_MB_IsViFrame
 - 2.3.28 RK_MPI_MB_GetImageInfo
 - 2.3.29 RK_MPI_MB_BeginCPUAccess
 - 2.3.30 RK_MPI_MB_EndCPUAccess
 - 2.4 数据类型
 - 2.4.1 基本数据类型
 - 2.4.1.1 公共数据类型
 - 2.4.1.2 IMAGE_TYPE_E
 - 2.4.1.3 CODEC_TYPE_E
 - 2.4.1.4 MOD_ID_E
 - 2.4.1.5 Sample_Format_E
 - 2.4.1.6 RECT_S
 - 2.4.2 系统控制数据类型
 - 2.4.2.1 MPP_CHN_S
 - 2.4.2.2 EventCbFunc
 - 2.4.2.3 MEDIA_BUFFER
 - 2.4.2.4 OutCbFunc
 - 2.4.2.5 MB_IMAGE_INFO_S
 - 2.5 错误码

3. 视频输入

3.1 概述

3.2 功能描述

3.2.1 VI通路初始化

3.2.2 VI视频节点

3.2.3 VI工作模式

3.3 API参考

3.3.1 RK_MPI_VI_EnableChn

3.3.2 RK_MPI_VI_DisableChn

3.3.3 RK_MPI_VI_SetChnAttr

3.3.4 RK_MPI_VI_StartRegionLuma

3.3.5 RK_MPI_VI_StopRegionLuma

3.3.6 RK_MPI_VI_GetChnRegionLuma

3.3.7 RK_MPI_VI_StartStream

3.4 数据类型

3.4.1 VI_MAX_CHN_NUM

3.4.2 VI_PIPE

3.4.3 VI_CHN

3.4.4 VI_CHN_ATTR_S

3.4.5 VIDEO_REGION_INFO_S

3.5 错误码

4. 视频编码

4.1 概述

4.2 功能描述

4.2.1 数据流程图

4.2.2 码率控制

4.2.3 GOP Mode

4.2.4 感兴趣区域(ROI)

4.2.5 旋转(Rotation)

4.3 API参考

4.3.1 RK_MPI_VENC_CreateChn

4.3.2 RK_MPI_VENC_GetVencChnAttr

4.3.3 RK_MPI_VENC_SetVencChnAttr

4.3.4 RK_MPI_VENC_CreateJpegLightChn

4.3.5 RK_MPI_VENC_DestroyChn

4.3.6 RK_MPI_VENC_GetRcParam

4.3.7 RK_MPI_VENC_SetRcParam

4.3.8 RK_MPI_VENC_SetRcMode

4.3.9 RK_MPI_VENC_SetRcQuality

4.3.10 RK_MPI_VENC_SetBitrate

4.3.11 RK_MPI_VENC_RequestIDR

4.3.12 RK_MPI_VENC_SetFps

4.3.13 RK_MPI_VENC_SetGop

4.3.14 RK_MPI_VENC_SetAvcProfile

4.3.15 RK_MPI_VENC_InsertUserData

4.3.16 RK_MPI_VENC_SetResolution

4.3.17 RK_MPI_VENC_GetRoiAttr

4.3.18 RK_MPI_VENC_SetRoiAttr

4.3.19 RK_MPI_VENC_SetGopMode

4.3.20 RK_MPI_VENC_RGN_Init

4.3.21 RK_MPI_VENC_RGN_SetBitMap

4.3.22 RK_MPI_VENC_RGN_SetCover

4.3.23 RK_MPI_VENC_RGN_SetPaletteId

4.3.24 RK_MPI_VENC_SetJpegParam

4.3.25 RK_MPI_VENC_StartRecvFrame

4.3.26 RK_MPI_VENC_GetFd

4.3.27 RK_MPI_VENC_QueryStatus

4.4 数据类型

- 4.4.1 VENC_MAX_CHN_NUM
 - 4.4.2 VENC_CHN
 - 4.4.3 VENC_ATTR_JPEG_S
 - 4.4.4 VENC_ATTR_MJPEG_S
 - 4.4.5 VENC_ATTR_H264_S
 - 4.4.6 VENC_ATTR_H265_S
 - 4.4.7 VENC_ATTR_S
 - 4.4.8 VENC_MJPEG_CBR_S
 - 4.4.9 VENC_MJPEG_VBR_S
 - 4.4.10 VENC_H264_CBR_S
 - 4.4.11 VENC_H264_VBR_S
 - 4.4.12 VENC_H265_CBR_S
 - 4.4.13 VENC_H265_VBR_S
 - 4.4.14 VENC_RC_MODE_E
 - 4.4.15 VENC_RC_ATTR_S
 - 4.4.16 VENC_GOP_MODE_E
 - 4.4.17 VENC_GOP_ATTR_S
 - 4.4.18 VENC_CHN_ATTR_S
 - 4.4.19 VENC_PARAM_MJPEG_S
 - 4.4.20 VENC_PARAM_H264_S
 - 4.4.21 VENC_PARAM_H265_S
 - 4.4.22 VENC_RC_PARAM_S
 - 4.4.23 VENC_RC_QUALITY_E
 - 4.4.24 VENC_ROI_ATTR_S
 - 4.4.25 OSD_REGION_ID_E
 - 4.4.26 OSD_REGION_INFO_S
 - 4.4.27 OSD_PIXEL_FORMAT_E
 - 4.4.28 VENC_COLOR_TBL_S
 - 4.4.29 OSD_COLOR_PALETTE_BUF_S
 - 4.4.30 BITMAP_S
 - 4.4.31 COVER_INFO_S
 - 4.4.32 VENC_RECV_PIC_PARAM_S
 - 4.4.33 VENC_JPEG_PARAM_S
 - 4.4.34 VENC_RESOLUTION_PARAM_S
 - 4.4.35 VENC_CHN_STATUS_S
- 4.5 错误码
- 5. 视频解码
 - 5.1 概述
 - 5.2 API参考
 - 5.2.1 RK_MPI_VDEC_CreateChn
 - 5.2.2 RK_MPI_VDEC_DestroyChn
 - 5.3 数据类型
 - 5.3.1 VDEC_MAX_CHN_NUM
 - 5.3.2 VDEC_CHN
 - 5.3.3 VDEC_CHN_ATTR_S
 - 5.3.4 VIDEO_MODE_E
 - 5.3.5 VIDEO_DECODEC_MODE_E
 - 5.3.6 VDEC_ATTR_VIDEO_S
- 6. 移动侦测
 - 6.1 概述
 - 6.2 功能描述
 - 6.3 API参考
 - 6.3.1 RK_MPI_ALGO_MD_CreateChn
 - 6.3.2 RK_MPI_ALGO_MD_DestroyChn
 - 6.3.3 RK_MPI_ALGO_MD_EnableSwitch
 - 6.4 数据类型
 - 6.4.1 ALGO_MD_MAX_CHN_NUM
 - 6.4.2 ALGO_MD_ROI_RET_MAX

- 6.4.3 ALGO_MD_CHN
 - 6.4.4 ALGO_MD_ATTR_S
- 6.5 错误码
- 7. 遮挡侦测
 - 7.1 概述
 - 7.2 功能描述
 - 7.3 API参考
 - 7.3.1 RK_MPI_ALGO_OD_CreateChn
 - 7.3.2 RK_MPI_ALGO_OD_DestroyChn
 - 7.3.3 RK_MPI_ALGO_OD_EnableSwitch
 - 7.4 数据类型
 - 7.4.1 ALGO_OD_MAX_CHN_NUM
 - 7.4.2 ALGO_OD_ROI_RET_MAX
 - 7.4.3 ALGO_OD_CHN
 - 7.4.4 ALGO_OD_ATTR_S
 - 7.5 错误码
- 8. 视频输出
 - 8.1 概述
 - 8.2 功能描述
 - 8.3 API参考
 - 8.3.1 RK_MPI_VO_CreateChn
 - 8.3.2 RK_MPI_VO_GetChnAttr
 - 8.3.3 RK_MPI_VO_SetChnAttr
 - 8.3.4 RK_MPI_VO_DestroyChn
 - 8.4 数据类型
 - 8.4.1 VO_MAX_CHN_NUM
 - 8.4.2 VO_CHN
 - 8.4.3 VO_CHN_ATTR_S
 - 8.5 错误码
- 9. 音频
 - 9.1 概述
 - 9.2 功能描述
 - 9.2.1 音频输入输出
 - 9.2.2 音频编解码
 - 9.2.3 音频算法
 - 9.3 API参考
 - 9.3.1 音频输入
 - 9.3.1.1 RK_MPI_AI_EnableChn
 - 9.3.1.2 RK_MPI_AI_DisableChn
 - 9.3.1.3 RK_MPI_AI_SetChnAttr
 - 9.3.1.4 RK_MPI_AI_SetVolume
 - 9.3.1.5 RK_MPI_AI_GetVolume
 - 9.3.1.6 RK_MPI_AI_SetTalkVqeAttr
 - 9.3.1.7 RK_MPI_AI_GetTalkVqeAttr
 - 9.3.1.8 RK_MPI_AI_SetRecordVqeAttr
 - 9.3.1.9 RK_MPI_AI_GetRecordVqeAttr
 - 9.3.1.10 RK_MPI_AI_EnableVqe
 - 9.3.1.11 RK_MPI_AI_DisableVqe
 - 9.3.1.12 RK_MPI_AI_StartStream
 - 9.3.2 音频输出
 - 9.3.2.1 RK_MPI_AO_EnableChn
 - 9.3.2.2 RK_MPI_AO_DisableChn
 - 9.3.2.3 RK_MPI_AO_SetChnAttr
 - 9.3.2.4 RK_MPI_AO_SetVolume
 - 9.3.2.5 RK_MPI_AO_GetVolume
 - 9.3.2.6 RK_MPI_AO_SetVqeAttr
 - 9.3.2.7 RK_MPI_AO_GetVqeAttr
 - 9.3.2.8 RK_MPI_AO_EnableVqe

9.3.2.9	RK_MPI_AO_DisableVqe
9.3.2.10	RK_MPI_AO_QueryChnStat
9.3.2.11	RK_MPI_AO_ClearChnBuf
9.3.3	音频编码
9.3.3.1	RK_MPI_AENC_CreateChn
9.3.3.2	RK_MPI_AENC_DestroyChn
9.3.4	音频解码
9.3.4.1	RK_MPI_ADEC_CreateChn
9.3.4.2	RK_MPI_ADEC_DestroyChn
9.4	数据类型
9.4.1	音频输入
9.4.1.1	AI_MAX_CHN_NUM
9.4.1.2	AI_CHN
9.4.1.3	AI_CHN_ATTR_S
9.4.1.4	AI_TALKVQE_CONFIG_S
9.4.1.5	AI_RECORDVQE_CONFIG_S
9.4.2	音频输出
9.4.2.1	AO_MAX_CHN_NUM
9.4.2.2	AO_CHN
9.4.2.3	AO_CHN_ATTR_S
9.4.2.4	AO_VQE_CONFIG_S
9.4.2.5	AO_CHN_STATE_S
9.4.3	音频编码
9.4.3.1	AENC_MAX_CHN_NUM
9.4.3.2	AENC_CHN
9.4.3.3	AENC_ATTR_AAC_S
9.4.3.4	AENC_ATTR_MP2_S
9.4.3.5	AENC_ATTR_G711A_S
9.4.3.6	AENC_ATTR_G711U_S
9.4.3.7	AENC_ATTR_G726_S
9.4.3.8	AENC_CHN_ATTR_S
9.4.4	音频解码
9.4.4.1	ADEC_MAX_CHN_NUM
9.4.4.2	ADEC_CHN
9.4.4.3	ADEC_ATTR_AAC_S
9.4.4.4	ADEC_ATTR_MP2_S
9.4.4.5	ADEC_ATTR_G711A_S
9.4.4.6	ADEC_ATTR_G711U_S
9.4.4.7	ADEC_ATTR_G726_S
9.4.4.8	ADEC_CHN_ATTR_S
9.5	错误码
9.5.1	音频输入错误码
9.5.2	音频输出错误码
9.5.3	音频编码错误码
9.5.4	音频解码错误码
10.	RGA
10.1	概述
10.2	功能描述
10.3	API参考
10.3.1	RK_MPI_RGA_CreateChn
10.3.2	RK_MPI_RGA_DestroyChn
10.4	数据类型
10.4.1	RGA_MAX_CHN_NUM
10.4.2	RGA_CHN
10.4.3	RGA_INFO_S
10.4.4	RGA_ATTR_S
10.5	错误码
11.	注意事项

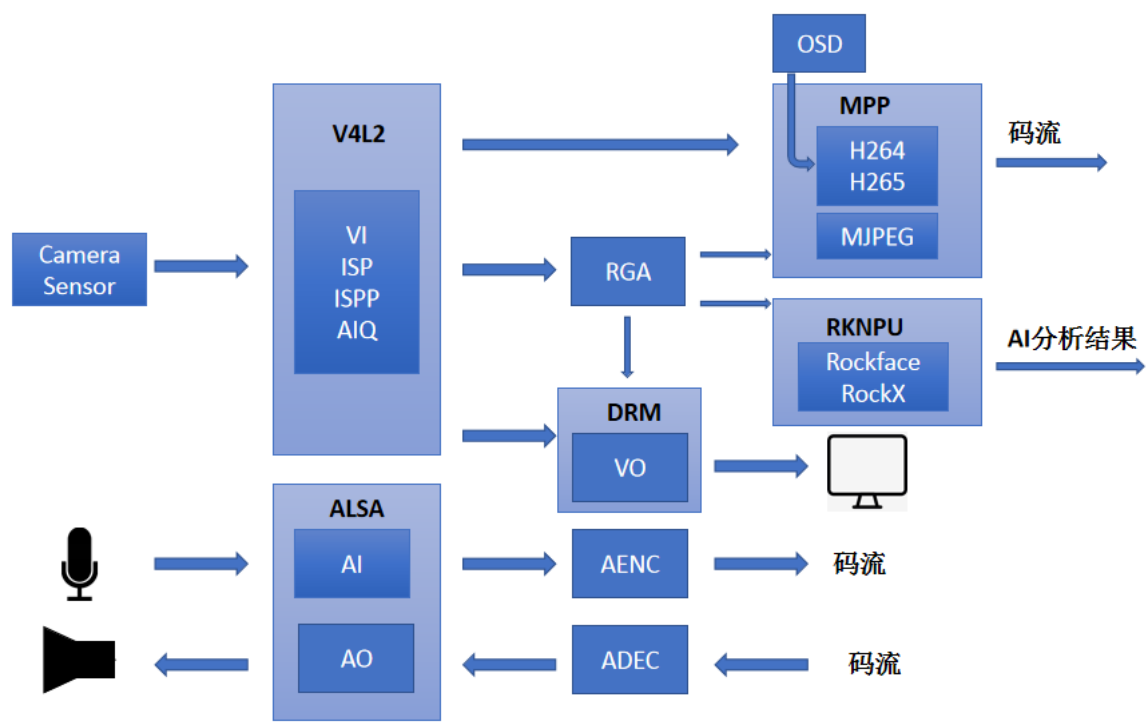
- 11.1 通道析构顺序
- 11.2 参数初始化
- 12. Proc调试信息说明
 - 12.1 VI
 - 12.2 VENC
- 13. LOG调试等级说明
- 14. 示例
 - 14.1 音频类示例
 - 14.1.1 rkmedia_ai_test
 - 14.1.2 rkmedia_ai_aenc_test
 - 14.1.3 rkmedia_ao_test
 - 14.1.4 rkmedia_adec_ao_test
 - 14.1.5 rkmedia_audio_test
 - 14.2 基础视频类示例
 - 14.2.1 rkmedia_vi_get_frame_test
 - 14.2.2 rkmedia_vi_luma_only_mode_test
 - 14.2.3 rkmedia_vi_multi_bind_test
 - 14.2.4 rkmedia_vi_venc_test
 - 14.2.5 rkmedia_vi_vo_test
 - 14.2.6 rkmedia_venc_avbr_test
 - 14.2.7 rkmedia_venc_jpeg_light_test
 - 14.2.8 rkmedia_venc_jpeg_test
 - 14.2.9 rkmedia_venc_local_file_test
 - 14.2.10 rkmedia_venc_smartp_test
 - 14.2.11 rkmedia_main_stream_with_jpeg_test
 - 14.2.12 rkmedia_vdec_test
 - 14.2.13 rkmedia_vo_display_test
 - 14.2.14 rkmedia_vo_scale_test
 - 14.3 RGA&OSD相关示例
 - 14.3.1 rkmedia_venc_cover_test
 - 14.3.2 rkmedia_venc_mjpeg_test
 - 14.3.3 rkmedia_venc_osd_test
 - 14.3.4 rkmedia_venc_roi_osd_test
 - 14.3.5 rkmedia_rga_api_test
 - 14.3.6 rkmedia_rga_crop_venc_test
 - 14.3.7 rkmedia_rga_osd_test
 - 14.4 ISP示例
 - 14.4.1 rkmedia_isp_test
 - 14.4.2 rkmedia_vi_double_cameras_test
 - 14.4.3 rkmedia_vi_double_cameras_switch_test
 - 14.5 智能分析使用示例
 - 14.5.1 rkmedia_vi_md_test
 - 14.5.2 rkmedia_vi_od_test
 - 14.5.3 rkmedia_vi_rknn_venc_rtsp_test
 - 14.5.4 rkmedia_vi_rockx_venc_rtsp_test

1. 系统概述

1.1 概述

RKMedia提供了一种媒体处理方案，可支持应用软件快速开发。RKMedia在各模块基础API上做进一步封装，简化了应用开发难度。该平台支持以下功能：VI(输入视频捕获)、VENC(H.265/H.264/JPEG/MJPEG 编码)、VDEC(H.265/H.264/JPEG、MJPEG 解码)、VO(视频输出显示)、RGA视频处理（包括旋转、缩放、裁剪）、AI(音频采集)、AO（音频输出）、AENC（音频编码）、ADEC（音频解码）、MD（移动侦测）、OD（遮挡侦测）。

1.2 系统架构



1.3 系统资源数目表

模块名称	通道数量
VI	4
VENC	16
VDEC	16
AI	1
AO	1
AENC	16
ADEC	16
MD	4
OD	4
RGA	16
VO	2

2. 系统控制

2.1 概述

系统控制基本系统的初始化工作，同时负责完成各个模块的初始化、反初始化以及管理各个业务模块的绑定关系、提供当前系统版本、系统日志管理。

2.2 功能描述

2.2.1 系统绑定

RKMedia提供系统绑定接口（[RK_MPL_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（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。目前支持的绑定关系如[表 2-1](#)所示。

表2-1 RKMedia支持的绑定关系

数据源	数据接受者
VI	VO/RGA/VENC/MD/OD
VDEC	VO/RGA/VENC/MD/OD
RGA	VO/VENC/MD/OD
AI	AO/AENC
ADEC	AO

2.3 API参考

2.3.1 RK_MPI_SYS_Init

【描述】

初始化系统。

【语法】

RK_S32 RK_MPI_SYS_Init();

【参数】

无。

【返回值】

返回值	描述
0	成功。
非0	失败，其值参见 错误码 。

【需求】

头文件：rkmedia_api.h

库文件：libeasymedia.so

【注意】

无。

【举例】

无。

【相关主题】

无。

2.3.2 RK_MPI_SYS_DumpChn

【描述】

打印通道信息。

【语法】

RK_VOID RK_MPI_SYS_DumpChn([MOD_ID_E](#) enModId);

【参数】

参数名称	描述	输入/输出
enModId	模块号。	输入

【返回值】

无

【需求】

头文件：rkmedia_api.h

库文件：libeasymedia.so

【注意】

无。

【举例】

无。

【相关主题】

无。

2.3.3 RK_MPI_SYS_Bind

【描述】

数据源到数据接收者绑定接口。

【语法】

RK_S32 RK_MPI_SYS_Bind(const [MPP_CHN_S](#) *pstSrcChn,const [MPP_CHN_S](#) *pstDestChn);

【参数】

参数名称	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，其值参见 错误码 。

【需求】

头文件：rkmedia_api.h

库文件：libeasymedia.so

【注意】

系统目前支持的绑定关系，请参见[表2-1](#)。

在释放被绑定的通道前，需先通过[RK_MPI_SYS_UnBind](#)进行解绑。

【举例】

无。

【相关主题】

[RK_MPI_SYS_UnBind](#)

2.3.4 RK_MPI_SYS_UnBind

【描述】

数据源到数据接收者解绑定接口。

【语法】

RK_MPI_SYS_UnBind(const [MPP_CHN_S](#) *pstSrcChn,const [MPP_CHN_S](#) *pstDestChn);

【参数】

参数名称	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，其值参见 错误码 。

【需求】

头文件：rkmedia_api.h

库文件：libeasymedia.so

【注意】

无。

【举例】

无。

【相关主题】

[RK_MPI_SYS_Bind](#)

2.3.5 RK_MPI_SYS_RegisterEventCb

【描述】

注册事件回调，比如移动侦测事件。

【语法】

RK_S32 RK_MPI_SYS_RegisterEventCb(const [MPP_CHN_S](#) *pstChn, [EventCbFunc](#) cb);

【参数】

参数名称	描述	输入/输出
pstChn	指定通道指针。	输入
cb	事件回调函数。	输出

【返回值】

返回值	描述
0	成功。
非0	失败，其值参见 错误码 。

【需求】

头文件：rkmedia_api.h

库文件：libeasymedia.so

【注意】

无。

【举例】

[rkmedia_vi_md_test](#)。

【相关主题】

无。

2.3.6 RK_MPI_SYS_RegisterOutCb

【描述】

注册数据输出回调。

【描述】

注册数据输出回调。与[RK_MPI_SYS_GetMediaBuffer](#)相比，无需缓存buffer等待用户索取，因此更节省内存。

【语法】

RK_S32 RK_MPI_SYS_RegisterOutCb(const [MPP_CHN_S](#) *pstChn, [OutCbFunc](#) cb);

【参数】

参数名称	描述	输入/输出
pstChn	指定通道指针。	输入
cb	数据输出回调函数。	输出

【返回值】

返回值	描述
0	成功。
非0	失败，其值参见 错误码 。

【需求】

头文件：rkmedia_api.h

库文件：libeasymedia.so

【注意】

回调函数不能处理耗时操作，否则对应通道数据流将被阻塞。

【举例】

[rkmedia_vi_venc_test](#)。

【相关主题】

无。

2.3.7 RK_MPI_SYS_SendMediaBuffer

【描述】

向指定通道输入数据，比如将本地yuv文件送入编码器编码。

【语法】

RK_S32 RK_MPI_SYS_SendMediaBuffer([MOD_ID_E](#) enModID, RK_S32 s32ChnID, [MEDIA_BUFFER](#) buffer);

【参数】

参数名称	描述	输入/输出
enModID	模块号。	输入
s32ChnID	通道号。	输入
buffer	缓冲区。	输入

【返回值】

返回值	描述
0	成功。
非0	失败，其值参见 错误码 。

【需求】

头文件：rkmedia_api.h

库文件：libeasymedia.so

【注意】

无。

【举例】

[rkmedia vo_display_test](#)。

【相关主题】

[RK_MPI_SYS_GetMediaBuffer](#)

2.3.8 RK_MPI_SYS_StartGetMediaBuffer

【描述】

启用接收缓冲区。启用接收缓冲区后，即使通道Bind，也能通过[RK_MPI_SYS_GetMediaBuffer](#)获取MediaBuffer。

【语法】

RK_S32 RK_MPI_SYS_StartGetMediaBuffer([MOD_ID_E](#) enModID, RK_S32 s32ChnID);

【参数】

参数名称	描述	输入/输出
enModID	模块号。	输入
s32ChnID	通道号。	输入

【返回值】

返回值	描述
0	成功。
非0	失败，其值参见 错误码 。

【需求】

头文件：rkmedia_api.h

库文件：libeasymedia.so

【注意】

启用接收缓冲区后，要及时调用[RK_MPI_SYS_GetMediaBuffer](#)取走MediaBuffer，否则会提示丢包警告。

【举例】

无。

【相关主题】

[RK_MPI_SYS_GetMediaBuffer](#)

[RK_MPI_SYS_StopGetMediaBuffer](#)

2.3.9 RK_MPI_SYS_StopGetMediaBuffer

【描述】

关闭接收缓冲区，并清理缓冲区现有MediaBuffer。

【语法】

[MEDIA_BUFFER](#) RK_MPI_SYS_StopGetMediaBuffer([MOD_ID_E](#) enModID, RK_S32 s32ChnID);

【参数】

参数名称	描述	输入/输出
enModID	模块号。	输入
s32ChnID	通道号。	输入

【返回值】

返回值	描述
0	成功。
非0	失败，其值参见 错误码 。

【需求】

头文件：rkmedia_api.h

库文件：libeasymedia.so

【注意】

调用该接口后，若再次调用[RK_MPI_SYS_GetMediaBuffer](#)获取数据，则接收缓冲区再次被打开。

【举例】

无。

【相关主题】

[RK_MPI_SYS_StartGetMediaBuffer](#)

[RK_MPI_SYS_GetMediaBuffer](#)

2.3.10 RK_MPI_SYS_GetMediaBuffer

【描述】

从指定通道中获取数据。

【语法】

[MEDIA_BUFFER](#) RK_MPI_SYS_GetMediaBuffer([MOD_ID_E](#) enModID, RK_S32 s32ChnID, RK_S32 s32MilliSec);

【参数】

参数名称	描述	输入/输出
enModID	模块号。	输入
s32ChnID	通道号。	输入
s32MilliSec	阻塞等待时间。	输入

【返回值】

返回值类型	描述
MEDIA_BUFFER	缓冲区指针。

【需求】

头文件：rkmedia_api.h

库文件：libeasymedia.so

【注意】

该接口会自动触发[RK_MPI_SYS_StartGetMediaBuffer](#)。

【举例】

[rkmedia_vi_get_frame_test](#)。

【相关主题】

[RK_MPI_SYS_StartGetMediaBuffer](#)

[RK_MPI_SYS_StopGetMediaBuffer](#)

2.3.11 RK_MPI_MB_CreateBuffer

【描述】

创建普通缓冲区。

【语法】

[MEDIA_BUFFER](#) RK_MPI_MB_CreateBuffer(RK_U32 u32Size, RK_BOOL boolHardWare, RK_U8 u8Flag);

参数名称	描述	输入/输出
u32Size	创建缓冲区大小。	输入
boolHardWare	是否创建硬件类型Buffer（DMA Buffer）。	输入
u8Flag	硬件类型Buffer附加标记，取值： 0：开辟带缓存类型的硬件Buffer MB_FLAG_NOCACHED：开辟不带缓存类型的硬件Buffer	输入

【返回值】

返回值类型	描述
MEDIA_BUFFER	缓冲区指针。

【需求】

头文件：rkmedia_buffer.h

库文件：libeasymedia.so

【注意】

普通缓冲区不携带图像信息结构体。

【举例】

[rkmedia_venc_local_file_test](#)。

【相关主题】

无。

2.3.12 RK_MPI_MB_CreateImageBuffer

【描述】

创建图像缓冲区。与普通缓冲区相比，携带图像信息结构体。在图像处理过程中，推荐使用该方法获取缓冲区。

【语法】

[MEDIA_BUFFER](#) RK_MPI_MB_CreateImageBuffer([MB_IMAGE_INFO_S](#) *pstImageInfo, RK_BOOL boolHardWare, RK_U8 u8Flag);

参数名称	描述	输入/输出
pstImageInfo	图像缓冲区信息结构体。	输入
boolHardWare	是否创建硬件类型Buffer（DMA Buffer）。	输入
u8Flag	硬件类型Buffer附加标记，取值： 0：开辟带缓存类型的硬件Buffer MB_FLAG_NOCACHED：开辟不带缓存类型的硬件Buffer	输入

【返回值】

返回值类型	描述
MEDIA_BUFFER	缓冲区指针。

【需求】

头文件：rkmedia_buffer.h

库文件：libeasymedia.so

【注意】

无。

【举例】

[rkmedia_venc_local_file_test](#)。

【相关主题】

无。

2.3.13 RK_MPI_MB_CreateAudioBuffer

【描述】

创建音频缓冲区。

【语法】

[MEDIA_BUFFER](#) RK_MPI_MB_CreateAudioBuffer(RK_U32 u32BufferSize, RK_BOOL boolHardWare);

参数名称	描述	输入/输出
u32BufferSize	缓冲区大小。	输入
boolHardWare	是否创建硬件类型Buffer（DMA Buffer）。	输入

【返回值】

返回值类型	描述
MEDIA_BUFFER	缓冲区指针。

【需求】

头文件：rkmedia_buffer.h

库文件：libeasymedia.so

【注意】

无。

【举例】

[rkmedia_adec_ao_test](#)。

【相关主题】

无。

2.3.14 RK_MPI_MB_ConvertToImgBuffer

【描述】

将普通缓冲区转换为图像缓冲区。

【语法】

[MEDIA_BUFFER](#) RK_MPI_MB_ConvertToImgBuffer([MEDIA_BUFFER](#) mb, [MB_IMAGE_INFO_S](#) *pstImageInfo);

参数名称	描述	输入/输出
mb	普通缓冲区指针。	输入
pstImageInfo	图像缓冲区信息结构体。	输入

【返回值】

返回值类型	描述
MEDIA_BUFFER	缓冲区指针。

【需求】

头文件：rkmedia_buffer.h

库文件：libeasymedia.so

【注意】

无。

【举例】

无。

【相关主题】

无。

2.3.15 RK_MPI_MB_ConvertToAudBuffer

【描述】

将普通缓冲区转换为音频缓冲区。

【语法】

[MEDIA_BUFFER](#) RK_MPI_MB_ConvertToAudBuffer([MEDIA_BUFFER](#) mb);

参数名称	描述	输入/输出
mb	普通缓冲区指针。	输入

【返回值】

返回值类型	描述
MEDIA_BUFFER	缓冲区指针。

【需求】

头文件：rkmedia_buffer.h

库文件：libeasymedia.so

【注意】

无。

【举例】

无。

【相关主题】

无。

2.3.16 RK_MPI_MB_ReleaseBuffer

【描述】

释放缓冲区。

【语法】

RK_S32 RK_MPI_MB_ReleaseBuffer([MEDIA_BUFFER](#) mb);

【参数】

参数名称	描述	输入/输出
mb	缓冲区指针。	输入

【返回值】

返回值	描述
0	成功。
非0	失败，其值参见 错误码 。

【需求】

头文件：rkmedia_buffer.h

库文件：libeasymedia.so

【注意】

无。

【举例】

[rkmedia_venc_local_file_test](#)。

【相关主题】

[RK_MPI_SYS_GetMediaBuffer](#)

2.3.17 RK_MPI_MB_GetPtr

【描述】

从指定的[MEDIA_BUFFER](#)中获取缓冲区数据指针。

【语法】

void *RK_MPI_MB_GetPtr([MEDIA_BUFFER](#) mb);

【参数】

参数名称	描述	输入/输出
mb	缓冲区指针。	输入

【返回值】

返回值类型	描述
void *	缓冲区数据指针。

【需求】

头文件：rkmedia_buffer.h

库文件：libeasymedia.so

【注意】

无。

【举例】

[rkmedia_venc_local_file_test](#)。

【相关主题】

无。

2.3.18 RK_MPI_MB_GetFD

【描述】

从指定的[MEDIA_BUFFER](#)中获取文件描述符。

【语法】

```
int RK_MPI_MB_GetFD(MEDIA\_BUFFER mb);
```

【参数】

参数名称	描述	输入/输出
mb	缓冲区指针。	输入

【返回值】

返回值类型	描述
int	文件描述符。

【需求】

头文件：rkmedia_buffer.h

库文件：libeasymedia.so

【注意】

无。

【举例】

[rkmedia_venc_local_file_test](#)。

【相关主题】

无。

2.3.19 RK_MPI_MB_GetSize

【描述】

从指定的[MEDIA_BUFFER](#)中获取缓冲区数据大小。

【语法】

```
size_t RK_MPI_MB_GetSize(MEDIA\_BUFFER mb);
```

【参数】

参数名称	描述	输入/输出
mb	缓冲区。	输入

【返回值】

返回值类型	描述
size_t	缓冲区数据大小。

【需求】

头文件：rkmedia_buffer.h

库文件：libeasymedia.so

【注意】

无。

【举例】

[rkmedia_venc_local_file_test](#)。

【相关主题】

无。

2.3.20 RK_MPI_MB_SetSzie

【描述】

设置指定的[MEDIA_BUFFER](#)数据大小。

【语法】

RK_S32 RK_MPI_MB_SetSzie([MEDIA_BUFFER](#) mb, RK_U32 size);

【参数】

参数名称	描述	输入/输出
mb	缓冲区指针。	输入
size	缓冲区数据大小	输入

【返回值】

返回值	描述
0	成功。
非0	失败，其值参见 错误码 。

【需求】

头文件：rkmedia_buffer.h

库文件: libeasymedia.so

【注意】

在操作缓冲区，并改变其大小后需使用此函数设置缓冲区数据大小，否则[RK_MPI_MB_GetSize](#)将无法获取到正确的缓冲区数据大小。

【举例】

无。

【相关主题】

[RK_MPI_MB_GetSize](#)。

2.3.21 RK_MPI_MB_GetModelID

【描述】

从指定的[MEDIA_BUFFER](#)中获取模块ID。

【语法】

[MOD_ID_E](#) RK_MPI_MB_GetModelID([MEDIA_BUFFER](#) mb);

【参数】

参数名称	描述	输入/输出
mb	缓冲区。	输入

【返回值】

返回值类型	描述
MOD_ID_E	模块ID。

【需求】

头文件: rkmedia_buffer.h

库文件: libeasymedia.so

【注意】

无。

【举例】

[rkmedia_vi_multi_bind_test](#)。

【相关主题】

无。

2.3.22 RK_MPI_MB_GetChannelID

【描述】

从指定的[MEDIA_BUFFER](#)中获取通道ID。

【语法】

RK_S16 RK_MPI_MB_GetChannelID([MEDIA_BUFFER](#) mb);

【参数】

参数名称	描述	输入/输出
mb	缓冲区。	输入

【返回值】

返回值类型	描述
RK_S16	通道ID。

【需求】

头文件：rkmedia_buffer.h

库文件：libeasymedia.so

【注意】

无。

【举例】

[rkmedia vi multi bind test](#)。

【相关主题】

无。

2.3.23 RK_MPI_MB_GetTimestamp

【描述】

从指定的[MEDIA_BUFFER](#)中获取时间戳。

【语法】

RK_U64 RK_MPI_MB_GetTimestamp([MEDIA_BUFFER](#) mb);

【参数】

参数名称	描述	输入/输出
mb	缓冲区。	输入

【返回值】

返回值类型	描述
RK_U64	时间戳。

【需求】

头文件：rkmedia_buffer.h

库文件：libeasymedia.so

【注意】

无。

【举例】

[rkmedia_vi_rockx_venc_rtsp_test](#)。

【相关主题】

[RK_MPI_MB_SetTimestamp](#)。

2.3.24 RK_MPI_MB_SetTimestamp

【描述】

设置指定的[MEDIA_BUFFER](#)的时间戳。

【语法】

RK_S32 RK_MPI_MB_SetTimestamp([MEDIA_BUFFER](#) mb, RK_U64 timestamp);

【参数】

参数名称	描述	输入/输出
mb	缓冲区。	输入
timestamp	时间戳	输入

【返回值】

返回值	描述
0	成功。
非0	失败，其值参见 错误码 。

【需求】

头文件：rkmedia_buffer.h

库文件：libeasymedia.so

【注意】

直接调用librga库（相对RGA Channel）处理MEDIA_BUFFER时，需要用户手动处理时间戳属性。比如：

```
// 调用librga api处理MediaBuffer0，输出的MediaBuffer1的时间戳属性将丢失
// MediaBuffer0 --> RGA Crop --> MediaBuffer1
// 需要调用该接口手动拷贝MediaBuffer0的时间戳
RK_MPI_MB_SetTimestamp(MediaBuffer1, RK_MPI_MB_GetTimestamp(MediaBuffer0));
```

【举例】

[rkmedia_vi_rockx_venc_rtsp_test](#)。

【相关主题】

[RK_MPI_MB_GetTimestamp](#)。

2.3.25 RK_MPI_MB_GetFlag

【描述】

从指定的[MEDIA_BUFFER](#)中获取Flag。Flag用于标记该Buffer的特殊属性，比如帧类型：I帧、P帧等。

【语法】

RK_S32 RK_MPI_MB_GetFlag([MEDIA_BUFFER](#) mb);

【参数】

参数名称	描述	输入/输出
mb	缓冲区。	输入

【返回值】

返回值类型	描述
RK_S32	硬件类型Buffer附加标记： 0：开辟带缓存类型的硬件Buffer MB_FLAG_NOCACHED：开辟不带缓存类型的硬件Buffer

【需求】

头文件：rkmedia_buffer.h

库文件：libeasymedia.so

【注意】

无。

【举例】

[rkmedia_venc_smartp_test](#)。

【相关主题】

无。

2.3.26 RK_MPI_MB_GetTsvcLevel

【描述】

从指定的[MEDIA_BUFFER](#)中获取TSVC等级。TSVC表示时间维度上的SVC，启用TSVC功能后，码流将分三个Level：L0、L1、L2。高等级的编码帧需要依赖低等级编码帧才能解码（如L2依赖L1、L0；L1依赖L0），而低等级的变编码帧可独立解码（L0可独立解码，L1、L0可独立于L2进行解码）。比如60fps的视频，如果只解码L1、L0，则会获得一个30fps的视频，如果只解码L0，则获得一个15fps的视频。

【语法】

RK_S32 RK_MPI_MB_GetTsvcLevel([MEDIA_BUFFER](#) mb);

【参数】

参数名称	描述	输入/输出
mb	缓冲区。	输入

【返回值】

返回值类型	描述
RK_S32	TSVC等级。

【需求】

头文件：rkmedia_buffer.h

库文件：libeasymedia.so

【注意】

若编码器没开启TSVC/SMARTP模式，该接口返回值无效。

【举例】

[rkmedia_venc_smartp_test](#)。

【相关主题】

无。

2.3.27 RK_MPI_MB_IsViFrame

【描述】

判断指定的[MEDIA_BUFFER](#)是否为VirtualIntra帧（虚拟I帧）。

【语法】

RK_BOOL RK_MPI_MB_IsViFrame([MEDIA_BUFFER](#) mb);

【参数】

参数名称	描述	输入/输出
mb	缓冲区。	输入

【返回值】

返回值类型	描述
RK_BOOL	是否为VI帧。

【需求】

头文件：rkmedia_buffer.h

库文件：libeasymedia.so

【注意】

仅在smartp模式下有效。

【举例】

[rkmedia_venc_smartp_test](#)。

【相关主题】

无。

2.3.28 RK_MPI_MB_GetImageInfo

【描述】

从指定的图像缓冲区[MEDIA_BUFFER](#)中获取图像信息。

【语法】

RK_S32 RK_MPI_MB_GetImageInfo([MEDIA_BUFFER](#) mb, [MB_IMAGE_INFO_S](#) *pstImageInfo);

【参数】

参数名称	描述	输入/输出
mb	缓冲区指针。	输入
pstImageInfo	缓冲区图像信息结构体指针。	输出

【返回值】

返回值	描述
0	成功。
非0	失败，其值参见 错误码 。

【需求】

头文件：rkmedia_buffer.h

库文件：libeasymedia.so

【注意】

仅图像缓冲区可使用该函数获取信息。

【举例】

[rkmedia_vi_get_frame_test](#)。

【相关主题】

无。

2.3.29 RK_MPI_MB_BeginCPUAccess

【描述】

解决CPU与硬件模块（如：ENCODER）操作同一个[MEDIA_BUFFER](#)产生的同步问题。

【语法】

RK_S32 RK_MPI_MB_BeginCPUAccess([MEDIA_BUFFER](#) mb, RK_BOOL bReadonly);

【参数】

参数名称	描述	输入/输出
mb	缓冲区指针。	输入
bReadonly	是否只读。	输入

【返回值】

返回值	描述
0	成功。
非0	失败，其值参见 错误码 。

【需求】

头文件：rkmedia_buffer.h

库文件：libeasymedia.so

【注意】

需与[RK_MPI_MB_EndCPUAccess](#)一同使用。

【举例】

比如CPU填充Buffer后送给VENC编码的场景，使用[RK_MPI_MB_CreateImageBuffer](#)接口开辟的MediaBuffer默认是带Cache（缓存）的（可通过flag=MB_FLAG_NOCACHED开辟NoCache类型的MediaBuffer），因此CPU写Buffer后会有部分数据留在缓存中没能及时同步到内存（DDR）中，此时立即送入VENC编码，会导致画面有异常（比如间断性绿色条短线）。使用该接口就是保障CPU操作Buffer之后，缓存被立即刷新到内存。

```

MEDIA_BUFFER mb;
RK_MPI_MB_BeginCPUAccess(mb, RK_FALSE);
// CPU fill data to mb.
memset(RK_MPI_MB_GetPtr(mb), 'F', size);
RK_MPI_MB_EndCPUAccess(mb, RK_FALSE);
// Send mb to VENC
RK_MPI_SYS_SendMediaBuffer(RK_ID_VENC, 0, mb);

```

【相关主题】

无。

2.3.30 RK_MPI_MB_EndCPUAccess

【描述】

解决CPU与硬件模块（如：ENCODER）操作同一个[MEDIA_BUFFER](#)产生的同步问题。

【语法】

RK_S32 RK_MPI_MB_EndCPUAccess([MEDIA_BUFFER](#) mb, RK_BOOL bReadonly);

【参数】

参数名称	描述	输入/输出
mb	缓冲区指针。	输入
bReadonly	是否只读。	输入

【返回值】

返回值	描述
0	成功。
非0	失败，其值参见 错误码 。

【需求】

头文件：rkmedia_buffer.h

库文件：libasymedia.so

【注意】

需与[RK_MPI_MB_BeginCPUAccess](#)一同使用。

【举例】

无。

【相关主题】

无。

2.4 数据类型

2.4.1 基本数据类型

2.4.1.1 公共数据类型

【说明】

基本数据类型定义。

【定义】

```
typedef unsigned char RK_U8;
typedef unsigned short RK_U16;
typedef unsigned int RK_U32;

typedef signed char RK_S8;
typedef short RK_S16;
typedef int RK_S32;

typedef unsigned long RK_UL;
typedef signed long RK_SL;

typedef float RK_FLOAT;
typedef double RK_DOUBLE;

#ifndef _M_IX86
typedef unsigned long long RK_U64;
typedef long long RK_S64;
#else
typedef unsigned __int64 RK_U64;
typedef __int64 RK_S64;
#endif

typedef char RK_CHAR;
#define RK_VOID void

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 MAX_FILE_PATH_LEN    256
```

2.4.1.2 IMAGE_TYPE_E

【说明】

定义图像格式枚举类型。

【定义】

```
typedef enum rk_IMAGE_TYPE_E {  
    IMAGE_TYPE_UNKNOW = 0,  
    IMAGE_TYPE_GRAY8,  
    IMAGE_TYPE_GRAY16,  
    IMAGE_TYPE_YUV420P,  
    IMAGE_TYPE_NV12,  
    IMAGE_TYPE_NV21,  
    IMAGE_TYPE_YV12,  
    IMAGE_TYPE_FBC2,  
    IMAGE_TYPE_FBC0,  
    IMAGE_TYPE_YUV422P,  
    IMAGE_TYPE_NV16,  
    IMAGE_TYPE_NV61,  
    IMAGE_TYPE_YV16,  
    IMAGE_TYPE_YUYV422,  
    IMAGE_TYPE_UYVY422,  
    IMAGE_TYPE_RGB332,  
    IMAGE_TYPE_RGB565,  
    IMAGE_TYPE_BGR565,  
    IMAGE_TYPE_RGB888,  
    IMAGE_TYPE_BGR888,  
    IMAGE_TYPE_ARGB8888,  
    IMAGE_TYPE_ABGR8888,  
    IMAGE_TYPE_JPEG,  
  
    IMAGE_TYPE_BUTT  
} IMAGE_TYPE_E;
```

2.4.1.3 CODEC_TYPE_E

【说明】

定义编解码格式枚举类型。

【定义】

```
typedef enum rk_CODEC_TYPE_E {  
    RK_CODEC_TYPE_NONE = -1,  
    // Audio  
    RK_CODEC_TYPE_AAC,  
    RK_CODEC_TYPE_MP2,  
    RK_CODEC_TYPE_VORBIS,  
    RK_CODEC_TYPE_G711A,  
    RK_CODEC_TYPE_G711U,  
    RK_CODEC_TYPE_G726,  
    // Video  
    RK_CODEC_TYPE_H264,  
    RK_CODEC_TYPE_H265,
```

```
RK_CODEC_TYPE_JPEG,  
RK_CODEC_TYPE_MJPEG,  
RK_CODEC_TYPE_NB  
} CODEC_TYPE_E;
```

2.4.1.4 MOD_ID_E

【说明】

定义模块 ID 枚举类型。

【定义】

```
typedef enum rkMOD_ID_E {  
    RK_ID_UNKNOW = 0,  
    RK_ID_VB,  
    RK_ID_SYS,  
    RK_ID_VDEC,  
    RK_ID_VENC,  
    RK_ID_H264E,  
    RK_ID_JPEGE,  
    RK_ID_H265E,  
    RK_ID_VO,  
    RK_ID_VI,  
    RK_ID_AIO,  
    RK_ID_AI,  
    RK_ID_AO,  
    RK_ID_AENC,  
    RK_ID_ADEC,  
    RK_ID_ALGO_MD,  
    RK_ID_ALGO_OD,  
    RK_ID_RGA,  
  
    RK_ID_BUTT,  
} MOD_ID_E;
```

2.4.1.5 Sample_Format_E

【说明】

定义采样格式枚举类型。

【定义】

```
typedef enum rkSample_Format_E {  
    RK_SAMPLE_FMT_NONE = -1,  
    RK_SAMPLE_FMT_U8,  
    RK_SAMPLE_FMT_S16,  
    RK_SAMPLE_FMT_S32,  
    RK_SAMPLE_FMT_FLT,  
    RK_SAMPLE_FMT_U8P,  
    RK_SAMPLE_FMT_S16P,  
    RK_SAMPLE_FMT_S32P,  
    RK_SAMPLE_FMT_FLTP,  
    RK_SAMPLE_FMT_G711A,
```

```
RK_SAMPLE_FMT_G711U,  
RK_SAMPLE_FMT_NB  
} Sample_Format_E;
```

2.4.1.6 RECT_S

【说明】

定义区域属性结构体。

【定义】

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

【成员】

成员名称	描述
s32X	区域的X轴坐标
s32Y	区域的Y轴坐标
u32Width	区域的宽度
u32Height	区域的高度

【注意事项】

无。

【相关数据类型及接口】

无。

2.4.2 系统控制数据类型

系统控制相关数据类型定义如下：

[MPP_CHN_S](#)：定义模块设备通道结构体。

[EventCbFunc](#)：事件回调函数指针。

[MEDIA_BUFFER](#)：数据缓冲区指针。

[OutCbFunc](#)：数据输出回调函数指针。

[MB_IMAGE_INFO_S](#)：图像信息结构体。

2.4.2.1 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。

2.4.2.2 EventCbFunc

【说明】

事件回调函数指针。

【定义】

```
typedef enum rkEVENT_TYPE_E {
    RK_EVENT_ERR = 0,
    RK_EVENT_MD, // Algo::Move detection event.
    RK_EVENT_OD, // Algo::Occlusion detection event.
    RK_EVNET_BUT
} EVENT_TYPE_E;

typedef struct rkMD_EVENT_S {
    RK_U16 u16Cnt;
    RK_U32 u32Width;
    RK_U32 u32Height;
    RECT_S stRects[4096];
} MD_EVENT_S;

typedef struct rkOD_EVENT_S {
    RK_U16 u16Cnt;
    RK_U32 u32Width;
    RK_U32 u32Height;
    RECT_S stRects[10];
    RK_U16 u16Occlusion[10];
} OD_EVENT_S;

typedef struct rkEVENT_S {
```

```
EVENT_TYPE_E type;
MOD_ID_E mode_id;
union {
    MD_EVENT_S md_event;
    OD_EVENT_S stOdEvent;
};
} EVENT_S;

typedef void (*EventCbFunc) (EVENT_S *event);
```

【成员】

成员名称	描述
type	事件类型。
mode_id	模块号。
md_event	移动侦测事件。
stOdEvent	遮挡侦测事件。

2.4.2.3 MEDIA_BUFFER

【说明】

数据缓冲区指针。

【定义】

```
typedef void *MEDIA_BUFFER;
```

【相关数据类型及接口】

[OutCbFunc](#)

2.4.2.4 OutCbFunc

【说明】

数据输出回调函数指针。

【定义】

```
typedef void (*OutCbFunc) (MEDIA_BUFFER mb);
```

【相关数据类型及接口】

[MEDIA_BUFFER](#)

2.4.2.5 MB_IMAGE_INFO_S

【说明】

图像信息结构体。

【定义】

```
typedef struct rkMB_IMAGE_INFO {
    RK_U32 u32Width;
    RK_U32 u32Height;
    RK_U32 u32VerStride;
    RK_U32 u32HorStride;
    IMAGE_TYPE_E enImgType;
} MB_IMAGE_INFO_S;
```

【成员】

成员名称	描述
u32Width	宽度。
u32Height	高度。
u32HorStride	虚宽。
u32VerStride	虚高。
enImgType	图像格式类型。

【相关数据类型及接口】

[IMAGE_TYPE_E](#)

2.5 错误码

系统控制错误码如[表2-2](#)所示：

表2-2 系统控制 API 错误码

错误代码	宏定义	描述
1	RK_ERR_SYS_NULL_PTR	空指针错误
2	RK_ERR_SYS_NOTREADY	系统控制属性未配置
3	RK_ERR_SYS_NOT_PERM	操作不允许
4	RK_ERR_SYS_NOMEM	分配内存失败，如系统内存不足
5	RK_ERR_SYS_ILLEGAL_PARAM	参数设置无效
6	RK_ERR_SYS_BUSY	系统忙
7	RK_ERR_SYS_NOT_SUPPORT	不支持的功能

3. 视频输入

3.1 概述

视频输入（VideoInput，简称VI）用于读取Camera数据。该模块是对标准V4L2接口的封装，依赖Linux V4L2驱动架构。ISP/ISPP/VICAP驱动通过V4L2架构向用户层提供文件节点（如：/dev/video0），VI通过操作文件节点实现参数配置视频帧的读取等操作。

3.2 功能描述

3.2.1 VI通路初始化

针对rv1126/rv1109平台，需要调用rkaiq接口初始化硬件通路。使用方法可以参考external/rkmedia/examples/common/sample_common.h中的接口。

rkaiq与rkmedia VI接口联合使用说明：

1、限制条件：

如果所有VI Channel都关闭，那么需要重新进行通路初始化。多VI Channel场景下，如果只关闭部分VI Channel，则不需要重新初始化通路，如果所有VI Channel都关闭，则需要重新初始化。单VI Channel场景下，关闭VI Channel就需要调用ISP Stop逻辑关闭ISP通路；打开VI则需要再次初始化VI通路。

2、通路初始化与反初始化的接口调用顺序如下

初始化：

- 1) ISP Init // 对应rk_aiq_uapi_sysctl_init
- 2) ISP Run // 对应rk_aiq_uapi_sysctl_prepare & rk_aiq_uapi_sysctl_start
- 3) VI Enable（single/multi）

反初始化：

- 1) VI disable（single/multi）
- 2) ISP Stop // 对应rk_aiq_uapi_sysctl_stop & rk_aiq_uapi_sysctl_deinit

3.2.2 VI视频节点

VI的创建需要指定视频节点（VideoNode），比如“/dev/video0”。每个视频节点对应一路视频流。单摄像头能提供多种分辨率的视频流，比如RV1126/RV1109平台的ISPP可同时提供4种分辨率视频流，也是因为ISPP驱动向用户层提供了4个视频节点。

对于带有RKISP的平台（如：RV1126/RV1109），每个接入ISPP的Camera都将向用户提供4个视频节点，如下表所示。以“rkispp_”开头的名称是驱动提供的一种别名机制，在VI内部会翻译为对应的/dev/videoX节点，使用者只需使用这4个固定名称，就能获取不同分辨率的视频流。

表3-1 ISPP视频节点（RV1126/RV1109芯片）

ISPP 节点名称	最大宽度	缩放倍数	支持的输出格式
rkispp_m_bypass	Sensor最大宽度	不支持	NV12/NV16/YUYV/ FBC0/FBC2
rkispp_scale0	3264	1-8倍	NV12/NV16/YUYV
rkispp_scale1	1280	2-8倍	NV12/NV16/YUYV
rkispp_scale2	1280	2-8倍	NV12/NV16/YUYV

注：rkispp_m_bypass不支持缩放，分辨率仅能保持sensor最大分辨率。rkispp_scale0分辨率超过2K之后，需要使用NV16格式。

3.2.3 VI工作模式

VI有两种工作模式，如下表所示

模式名称	宏定义名称	功能说明
正常模式	VI_WORK_MODE_NORMAL	相对于“亮度模式”，该模式下正常读取Camera数据并发给后级。
亮度模式	VI_WORK_MODE_LUMA_ONLY	亮度模式下，VI仅用于亮度统计。此时VI模块无法通过回调函数或者RK_MPI_SYS_GetMediaBuffer获取数据。

3.3 API参考

3.3.1 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】

参数名称	描述	输入/输出
ViPipe	VI 管道号。	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，其值参见 错误码 。

【需求】

头文件：rkmedia_api.h

库文件：libeasymedia.so

【注意】

无。

【举例】

[rkmedia_vi_get_frame_test](#)。

【相关主题】

[RK_MPI_VI_DisableChn](#)

3.3.2 RK_MPI_VI_DisableChn

【描述】

关闭VI通道。

【语法】

RK_S32 RK_MPI_VI_DisableChn([VI_PIPE](#) ViPipe, [VI_CHN](#) ViChn);

【参数】

参数名称	描述	输入/输出
ViPipe	VI 管道号。	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，其值参见 错误码 。

【需求】

头文件：rkmedia_api.h

库文件：libeasymedia.so

【注意】

无。

【举例】

[rkmedia_vi_get_frame_test](#)。

【相关主题】

[RK_MPI_VI_EnableChn](#)

3.3.3 RK_MPI_VI_SetChnAttr

【描述】

设置VI通道属性。

【语法】

RK_MPI_VI_SetChnAttr([VI_PIPE](#) ViPipe, [VI_CHN](#) ViChn, const [VI_CHN_ATTR_S](#) *pstChnAttr);

【参数】

参数名称	描述	输入/输出
ViPipe	VI 管道号。	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，其值参见 错误码 。

【需求】

头文件：rkmedia_api.h

库文件：libeasymedia.so

【注意】

若被设置的通道已通过[RK_MPI_SYS_Bind](#)与其他通道绑定，则需在使用该函数设置前通过[RK_MPI_SYS_UnBind](#)进行解绑。

若被设置的通道已使用[RK_MPI_VI_EnableChn](#)使能，则需在使用该函数设置前通过[RK_MPI_VI_DisableChn](#) 关闭该通道。

【举例】

[rkmedia_vi_get_frame_test](#)。

【相关主题】

无。

3.3.4 RK_MPI_VI_StartRegionLuma

【描述】

开启VI亮度统计。

【语法】

RK_S32 RK_MPI_VI_StartRegionLuma([VI_CHN](#) ViChn);

【参数】

参数名称	描述	输入/输出
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，其值参见 错误码 。

【需求】

头文件：rkmedia_api.h

库文件：libeasymedia.so

【注意】

仅当[VI工作模式](#)设置[VI_WORK_MODE_LUMA_ONLY](#)为时生效。在该模式下，缓冲区个数（[u32BufCnt](#)）需要大于等于3。

【举例】

无。

【相关主题】

无。

3.3.5 RK_MPI_VI_StopRegionLuma

【描述】

停止VI亮度统计。

【语法】

RK_S32 RK_MPI_VI_StopRegionLuma([VI_CHN](#) ViChn);

【参数】

参数名称	描述	输入/输出
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，其值参见 错误码 。

【需求】

头文件：rkmedia_api.h

库文件：libeasymedia.so

【注意】

仅当[VI工作模式](#)设置[VI_WORK_MODE_LUMA_ONLY](#)为时生效。

【举例】

无。

【相关主题】

无。

3.3.6 RK_MPI_VI_GetChnRegionLuma

【描述】

获取区域亮度信息。可以用于VENC OSD反色。

【语法】

RK_S32 RK_MPI_VI_GetChnRegionLuma([VI_PIPE](#) ViPipe, [VI_CHN](#) ViChn, const [VIDEO_REGION_INFO_S](#) *pstRegionInfo, RK_U64 *pu64LumaData, RK_S32 s32MilliSec);

【参数】

参数名称	描述	输入/输出
ViPipe	VI 管道号。	输入
ViChn	VI 通道号。取值范围：[0, VI_MAX_CHN_NUM)	输入
pstRegionInfo	区域信息。其中 pstRegionInfo->pstRegion 为统计区域的区域属性，即起始位置、宽、高；pstRegionInfo->u32RegionNum 为统计区域的个数。	输入
pu64LumaData	接收区域亮度和统计信息的内存指针，该内存大小应该大于或等于 sizeof(RK_U64)×pstRegionInfo->u32RegionNum。	输出
s32MilliSec	超时参数 s32MilliSec：-1 表示阻塞模式；0 表示非阻塞模式；大于 0 表示超时模式，超时时间的单位为毫秒（ms）。	输入

【返回值】

返回值	描述
0	成功。
非0	失败，其值参见 错误码 。

【需求】

头文件：rkmedia_api.h

库文件：libeasymedia.so

【注意】

该接口不支持FBC0/FBC2压缩格式。

【举例】

[rkmedia_vi_luma_only_mode_test](#)。

【相关主题】

无。

3.3.7 RK_MPI_VI_StartStream

【描述】

启动视频流。

【语法】

RK_S32 RK_MPI_VI_StartStream([VI_PIPE](#) ViPipe, [VI_CHN](#) ViChn);

【参数】

参数名称	描述	输入/输出
ViPipe	VI 管道号。	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，其值参见 错误码 。

【需求】

头文件：rkmedia_api.h

库文件：libeasymedia.so

【注意】

无。

【举例】

[rkmedia_vi_get_frame_test](#)。

【相关主题】

无。

3.4 数据类型

视频输入相关数据类型定义如下：

[VI_MAX_CHN_NUM](#)：定义 VI 物理通道通道的总个数。

[VI_PIPE](#)：VI管道号。

[VI_CHN](#)：VI通道号。

[VI_CHN_ATTR_S](#)：VI 通道属性结构体指针。

[VIDEO_REGION_INFO_S](#)：定义视频区域信息。

3.4.1 VI_MAX_CHN_NUM

【说明】

定义 VI 物理通道的总个数。RV1126/RV1109平台典型场景是接入1~2个Sensor模组，每个Sensor最大能提供4个视频通道（对应ISPP的4个视频节点），因此个数最大为8个。若使实际使用场景超过当前限制（如接入三个Sensor，或者使用N4扩展板等），可自行对该宏定义进行调整。

【定义】

```
RV1109/RV1126:
#define VI_MAX_CHN_NUM 8
```

3.4.2 VI_PIPE

【说明】

VI管道号，对应Sensor个数。PIPE命名是由于Sensor后端要接入ISP/ISPP等一系列处理单元，形成一个数据PIPE（管道）。

【定义】

```
typedef RK_S32 VI_PIPE;
```

3.4.3 VI_CHN

【说明】

VI通道号。与[VI_PIPE](#)共同决定某个Camera的某路数据。

【定义】

```
typedef RK_S32 VI_CHN;
```

3.4.4 VI_CHN_ATTR_S

【说明】

VI 通道属性结构体指针。

【定义】

```
typedef char RK_CHAR;

typedef enum rkVI_CHN_WORK_MODE {
    VI_WORK_MODE_NORMAL = 0,
    // for vi single caculate luma.
    // In this mode, vi has no output,
    // and data cannot be obtained from vi.
    VI_WORK_MODE_LUMA_ONLY
} VI_CHN_WORK_MODE;

typedef enum rkVI_CHN_BUF_TYPE {
    VI_CHN_BUF_TYPE_DMA = 0, // Default
    VI_CHN_BUF_TYPE_MMAP,
} VI_CHN_BUF_TYPE;

typedef struct rkVI_CHN_ATTR_S {
    const RK_CHAR *pcVideoNode;
    RK_U32 u32Width;
    RK_U32 u32Height;
    IMAGE_TYPE_E enPixFmt;
    RK_U32 u32BufCnt; // VI capture video buffer cnt.
    // VI capture video buffer type.
    VI_CHN_BUF_TYPE enBufType;
    VI_CHN_WORK_MODE enWorkMode;
} VI_CHN_ATTR_S;
```

【成员】

成员名称	描述
pcVideoNode	video节点路径。
u32Width	video宽度。
u32Height	video高度。
enPixFmt	video格式。
u32BufCnt	VI捕获视频缓冲区计数
enWorkMode	VI通道工作模式

【注意事项】

VI_WORK_MODE_LUMA_ONLY模式，用于VI亮度统计，在此模式下VI没有输出，并且无法从VI获取数据。

【相关数据类型及接口】

[IMAGE_TYPE_E](#)

[RK_MPI_VI_SetChnAttr](#)

3.4.5 VIDEO_REGION_INFO_S

【说明】

定义视频区域信息。

【定义】

```
typedef struct rkVIDEO_REGION_INFO_S {
    RK_U32 u32RegionNum; /* count of the region */
    RECT_S *pstRegion; /* region attribute */
} VIDEO_REGION_INFO_S;
```

【成员】

成员名称	描述
u32RegionNum	视频区域个数。
pstRegion	视频区域位置信息指针。

【相关数据类型及接口】

[RECT_S](#)

[RK_MPI_VI_GetChnRegionLuma](#)

3.5 错误码

视频输入 API 错误码如表3-2所示：

表3-2 视频输入 API 错误码

错误代码	宏定义	描述
10	RK_ERR_VI_INVALID_CHNID	视频输入通道号无效
11	RK_ERR_VI_BUSY	视频输入系统忙
12	RK_ERR_VI_EXIST	视频输入通道已存在
13	RK_ERR_VI_NOT_CONFIG	视频输入未配置
14	RK_ERR_VI_TIMEOUT	视频输入超时
15	RK_ERR_VI_BUF_EMPTY	视频输入缓存为空
16	RK_ERR_VI_ILLEGAL_PARAM	视频输入参数设置无效
17	RK_ERR_VI_NOTREADY	视频输入系统未初始化

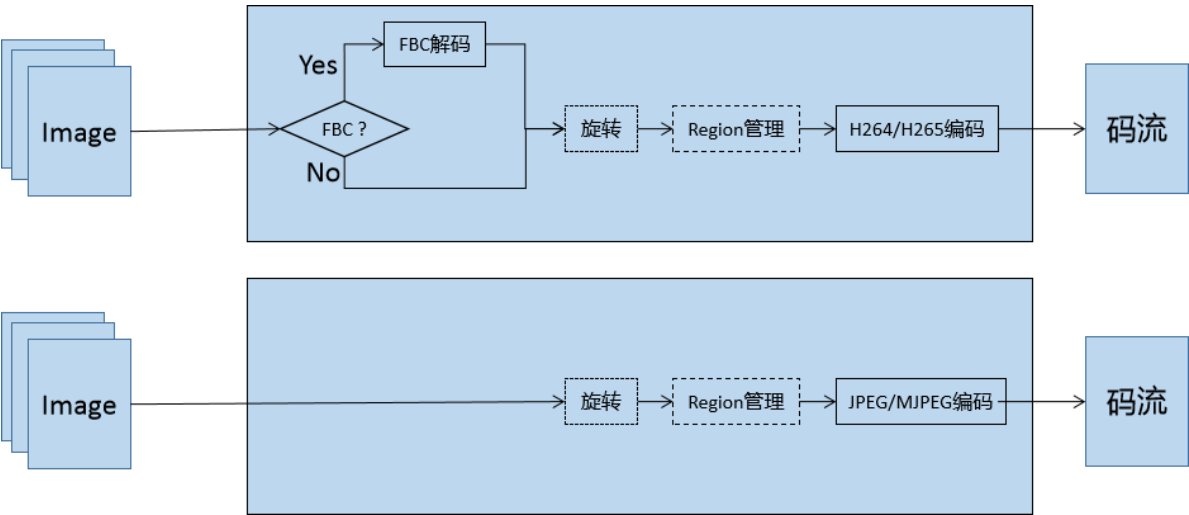
4. 视频编码

4.1 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。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。支持 H264/H265/MJPEG/JPEG编码。

4.2 功能描述

4.2.1 数据流程图



注：虚线框所述功能为可选，只有对编码器进行相应配置才会触发。

4.2.2 码率控制

编码器类型	支持码控类型
H265	CBR / VBR
H264	CBR / VBR
MJPEG	CBR / VBR

4.2.3 GOP Mode

GOP Mode用于定制参考帧的依赖关系，目前支持如下模式。注：可根据需求定制。

名称	宏定义	描述
普通模式	VENC_GOPMODE_NORMALP	最常见场景，每隔GopSize一个I帧
智能P帧模式	VENC_GOPMODE_SMARTP	每隔GopSize一个虚拟I帧，每隔BgInterval一个I帧
多层时域参考模式	VENC_GOPMODE_TSVC	编码依赖关系划分为多层，可根据RK_MPI_MB_GetTsvcLevel获取层信息，从而定制码流。 比如只播放第0层码流，可实现快速预览。

4.2.4 感兴趣区域(ROI)

通过配置编码器感兴趣区域，可实现指定区域QP的定制。比如一个对着走廊的镜头，用户真正感兴趣的是走廊中央。可通过配置ROI让走廊中央编码质量更高，图像更清晰，走廊的边框（墙体、天花板等）非感兴趣区域图像质量会偏低。通过这种方式实现保持码率基本不变情况下，突出显示用户关心区域。

每个VENC通道提供8个感兴趣区域，优先级从REGION_ID_0~REGION_ID_7递增。在多个ROI重叠的区域，其QP策略会按照优先级高的区域进行配置。

```
REGION_ID_0
REGION_ID_1
REGION_ID_2
REGION_ID_3
REGION_ID_4
REGION_ID_5
REGION_ID_6
REGION_ID_7
```

4.2.5 旋转(Rotation)

编码器支持4种类型的旋转，分别为0°，90°，180°，270°。编码器旋转目前不支持FBC格式，FBC格式的旋转则需要通过ISPP的旋转来实现。

4.3 API参考

4.3.1 RK_MPI_VENC_CreateChn

【描述】

创建编码通道。

【语法】

RK_S32 RK_MPI_VENC_CreateChn([VENC_CHN](#) VeChn, [VENC_CHN_ATTR_S](#) *stVencChnAttr);

【参数】

参数名称	描述	输入/输出
VeChn	编码通道号。取值范围：[0, VENC_MAX_CHN_NUM)。	输入
stVencChnAttr	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，其值参见 错误码 。

【需求】

头文件：rkmedia_api.h

库文件：libeasymedia.so

【注意】

与[RK_MPI_VENC_CreateJpegLightChn](#)创建的JPEG/MJPEG Light相比，该接口创建的JPEG/MJPEG 支持FBC格式、支持OSD、支持旋转、缩放；但不支持分辨率动态切换。

【举例】

[rkmedia_vi_venc_test](#)。

【相关主题】

无。

4.3.2 RK_MPI_VENC_GetVencChnAttr

【描述】

获取编码通道参数。

【语法】

```
RK_S32 RK_MPI_VENC_GetVencChnAttr(VENC_CHN VeChn, VENC_CHN_ATTR_S *stVencChnAttr);
```

【参数】

参数名称	描述	输入/输出
VeChn	编码通道号。取值范围：[0, VENC_MAX_CHN_NUM)。	输入
stVencChnAttr	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，其值参见 错误码 。

【需求】

头文件：rkmedia_api.h

库文件：libeasymedia.so

【注意】

该函数仅能获取已创建通道的参数配置。

【举例】

无。

【相关主题】

无。

4.3.3 RK_MPI_VENC_SetVencChnAttr

【描述】

设置编码通道参数。

【语法】

```
RK_S32 RK_MPI_VENC_SetVencChnAttr(VENC_CHN VeChn, VENC_CHN_ATTR_S *stVencChnAttr);
```

【参数】

参数名称	描述	输入/输出
VeChn	编码通道号。取值范围：[0, VENC_MAX_CHN_NUM)。	输入
stVencChnAttr	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，其值参见 错误码 。

【需求】

头文件：rkmedia_api.h

库文件：libeasymedia.so

【注意】

该函数仅能配置已创建通道的参数。目前支持编码复杂度（仅H264），分辨率，码率，帧率，GOP动态设置。其余配置修改，需销毁后重新创建通道。

分辨率动态配置需要保持编码器输入Buffer的宽高信息与动态配置要一致，否则将引起内存访问越界风险。该接口仅推荐在VENC不使用绑定（Bind）的场景下使用。而且要确保改变分辨率前，VENC Channel的输入buffer已经全部清空（参见[RK_MPI_VENC_QueryStatus](#)）

【举例】

无。

【相关主题】

无。

4.3.4 RK_MPI_VENC_CreateJpegLightChn

【描述】

创建轻量级的JPEG编码通道。

【语法】

RK_S32 RK_MPI_VENC_CreateJpegLightChn([VENC_CHN](#) VeChn, [VENC_CHN_ATTR_S](#) *stVencChnAttr);

【参数】

参数名称	描述	输入/输出
VeChn	编码通道号。取值范围：[0, VENC_MAX_CHN_NUM)。	输入
stVencChnAttr	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，其值参见 错误码 。

【需求】

头文件：rkmedia_api.h

库文件：libeasymedia.so

【注意】

与[RK_MPI_VENC_CreateChn](#)创建的JPEG/MJPEG相比，经过该接口创建的JPEG/MJPEG Light效率更高，支持分辨率动态切换；但不支持FBC格式、不支持OSD、不支持缩放。如果使用JPEG/MJPEG Light，可配合RGA做OSD和缩放功能。

【举例】

[rkmedia_venc_jpeg_light_test](#)。

【相关主题】

无。

4.3.5 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】

参数名称	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，其值参见 错误码 。

【需求】

头文件：rkmedia_api.h

库文件：libeasymedia.so

【注意】

无。

【举例】

[rkmedia vi venc test](#)。

【相关主题】

无。

4.3.6 RK_MPI_VENC_GetRcParam

【描述】

获取码率控制参数。

【语法】

RK_S32 RK_MPI_VENC_GetRcParam([VENC_CHN](#) VeChn, [VENC_RC_PARAM_S](#) *pstRcParam);

【参数】

参数名称	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。	输出

【返回值】

返回值	描述
0	成功。
非0	失败，其值参见 错误码 。

【需求】

头文件：rkmedia_api.h

库文件：libeasymedia.so

【注意】

无。

【举例】

无。

【相关主题】

[RK_MPI_VENC_SetRcParam](#)。

4.3.7 RK_MPI_VENC_SetRcParam

【描述】

设置码率控制参数。

【语法】

RK_S32 RK_MPI_VENC_SetRcParam([VENC_CHN](#) VeChn, const [VENC_RC_PARAM_S](#) *pstRcParam);

【参数】

参数名称	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，其值参见 错误码 。

【需求】

头文件：rkmedia_api.h

库文件：libeasymedia.so

【注意】

无。

【举例】

[rkmedia_isp_test](#)。

【相关主题】

[RK_MPI_VENC_GetRcParam](#)。

4.3.8 RK_MPI_VENC_SetRcMode

【描述】

设置码率控制模式。

【语法】

RK_S32 RK_MPI_VENC_SetRcMode([VENC_CHN](#) VeChn, [VENC_RC_MODE_E](#) RcMode);

【参数】

参数名称	描述	输入/输出
VeChn	编码通道号。取值范围：[0, VENC_MAX_CHN_NUM)。	输入
RcMode	码率控制模式。	输入

【返回值】

返回值	描述
0	成功。
非0	失败，其值参见 错误码 。

【需求】

头文件：rkmedia_api.h

库文件：libeasymedia.so

【注意】

无。

【举例】

无。

【相关主题】

无。

4.3.9 RK_MPI_VENC_SetRcQuality

【描述】

设置编码质量。用于H264 / H265编码器。

【语法】

RK_S32 RK_MPI_VENC_SetRcQuality([VENC_CHN](#) VeChn, [VENC_RC_QUALITY_E](#) RcQuality);

【参数】

参数名称	描述	输入/输出
VeChn	编码通道号。取值范围：[0, VENC_MAX_CHN_NUM)。	输入
RcQuality	编码质量。	输入

【返回值】

返回值	描述
0	成功。
非0	失败，其值参见 错误码 。

【需求】

头文件：rkmedia_api.h

库文件：libeasymedia.so

【注意】

无。

【举例】

无。

【相关主题】

无。

4.3.10 RK_MPI_VENC_SetBitrate

【描述】

设置码率。

【语法】

RK_S32 RK_MPI_VENC_SetBitrate([VENC_CHN](#) VeChn, RK_U32 u32BitRate, RK_U32 u32MinBitRate, RK_U32 u32MaxBitRate);

【参数】

参数名称	描述	输入/输出
VeChn	编码通道号。取值范围：[0, VENC_MAX_CHN_NUM)。	输入
u32BitRate	目标码率。	输入
u32MinBitRate	最小码率。单位bps	输入
u32MaxBitRate	最大码率。单位bps	输入

【返回值】

返回值	描述
0	成功。
非0	失败，其值参见 错误码 。

【需求】

头文件：rkmedia_api.h

库文件：libeasymedia.so

【注意】

无。

【举例】

无。

【相关主题】

无。

4.3.11 RK_MPI_VENC_RequestIDR

【描述】

请求IDR帧。调用该接口后，编码器立即刷新IDR帧。

【语法】

RK_S32 RK_MPI_VENC_RequestIDR([VENC_CHN](#) VeChn, RK_BOOL bInstant);

【参数】

参数名称	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，其值参见 错误码 。

【需求】

头文件：rkmedia_api.h

库文件：libeasymedia.so

【注意】

无。

【举例】

无。

【相关主题】

无。

4.3.12 RK_MPI_VENC_SetFps

【描述】

设置编码帧率。

【语法】

RK_S32 RK_MPI_VENC_SetFps([VENC_CHN](#) VeChn, RK_U8 u8OutNum, RK_U8 u8OutDen, RK_U8 u8InNum, RK_U8 u8InDen);

【参数】

参数名称	描述	输入/输出
VeChn	编码通道号。取值范围：[0, VENC_MAX_CHN_NUM)。	输入
u8OutNum	编码输出帧率分母。	输入
u8OutDen	编码输出帧率分子。	输入
u8InNum	编码输入帧率分母。	输入
u8InDen	编码输入帧率分子。	输入

【返回值】

返回值	描述
0	成功。
非0	失败，其值参见 错误码 。

【需求】

头文件：rkmedia_api.h

库文件：libeasymedia.so

【注意】

输出帧率不能大于输入帧率。

【举例】

无。

【相关主题】

无。

4.3.13 RK_MPI_VENC_SetGop

【描述】

设置GOP。用于H264 / H265编码器。

【语法】

RK_S32 RK_MPI_VENC_SetGop([VENC_CHN](#) VeChn, RK_U32 u32Gop);

【参数】

参数名称	描述	输入/输出
VeChn	编码通道号。取值范围：[0, VENC_MAX_CHN_NUM)。	输入
u32Gop	GOP。	输入

【返回值】

返回值	描述
0	成功。
非0	失败，其值参见 错误码 。

【需求】

头文件：rkmedia_api.h

库文件：libeasymedia.so

【注意】

无。

【举例】

无。

【相关主题】

无。

4.3.14 RK_MPI_VENC_SetAvcProfile

【描述】

设置 profile。用于H264 编码器。

【语法】

RK_S32 RK_MPI_VENC_SetAvcProfile([VENC_CHN](#) VeChn, RK_U32 u32Profile, RK_U32 u32Level);

【参数】

参数名称	描述	输入/输出
VeChn	编码通道号。取值范围：[0, VENC_MAX_CHN_NUM)。	输入
u32Profile	Profile IDC值。	输入
u32Level	Level IDC值。	输入

【返回值】

返回值	描述
0	成功。
非0	失败，其值参见 错误码 。

【需求】

头文件：rkmedia_api.h

库文件：libeasymedia.so

【注意】

暂时只支持u32Profile为66、77、100，分别对应Baseline、Main Profile、High Profile。

【举例】

无。

【相关主题】

无。

4.3.15 RK_MPI_VENC_InsertUserData

【描述】

插入用户数据，插入后的数据将在码流的SEI包中体现。用于H264 / H265编码器。

【语法】

RK_S32 RK_MPI_VENC_InsertUserData([VENC_CHN](#) VeChn, RK_U8 *pu8Data, RK_U32 u32Len);

【参数】

参数名称	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。	输入

【返回值】

返回值	描述
0	成功。
非0	失败，其值参见 错误码 。

【需求】

头文件：rkmedia_api.h

库文件：libeasymedia.so

【注意】

无。

【举例】

无。

【相关主题】

无。

4.3.16 RK_MPI_VENC_SetResolution

【描述】

设置VENC通道分辨率

【语法】

RK_S32 RK_MPI_VENC_SetResolution([VENC_CHN](#) VeChn, [VENC_RESOLUTION_PARAM_S](#) stResolutionParam);

【参数】

参数名称	描述	输入/输出
VeChn	编码通道号。取值范围：[0, VENC_MAX_CHN_NUM)。	输入
stResolutionParam	分辨率参数结构体。	输入

【返回值】

返回值	描述
0	成功。
非0	失败，其值参见 错误码 。

【需求】

头文件：rkmedia_api.h

库文件：libeasymedia.so

【注意】

无。

【举例】

[rkmedia_rga_crop_venc_test](#)。

【相关主题】

无。

4.3.17 RK_MPI_VENC_GetRoiAttr

【描述】

获取指定索引值的ROI配置参数。用于H264 / H265编码器。

【语法】

RK_S32 RK_MPI_VENC_GetRoiAttr([VENC_CHN](#) VeChn, [VENC_ROI_ATTR_S](#) *pstRoiAttr, RK_S32 roi_index);

【参数】

参数名称	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。	输出
roi_index	ROI区域索引值。取值范围：[0,7]。	输入

【返回值】

返回值	描述
0	成功。
非0	失败，其值参见 错误码 。

【需求】

头文件：rkmedia_api.h

库文件：libeasymedia.so

【注意】

无。

【举例】

无。

【相关主题】

[RK_MPI_VENC_SetRoiAttr](#)。

4.3.18 RK_MPI_VENC_SetRoiAttr

【描述】

设置ROI编码感兴趣区。用于H264 / H265编码器。

【语法】

RK_S32 RK_MPI_VENC_SetRoiAttr([VENC_CHN](#) VeChn,const [VENC_ROI_ATTR_S](#) *pstRoiAttr, RK_S32 region_cnt);

【参数】

参数名称	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
region_cnt	ROI区域个数。取值范围：[1,8]。	输入

【返回值】

返回值	描述
0	成功。
非0	失败，其值参见 错误码 。

【需求】

头文件：rkmedia_api.h

库文件：libeasymedia.so

【注意】

无。

【举例】

无。

【相关主题】

[RK_MPI_VENC_GetRoiAttr](#)。

4.3.19 RK_MPI_VENC_SetGopMode

【描述】

设置GopMode。用于H264 / H265编码器。

【语法】

RK_S32 RK_MPI_VENC_SetGopMode([VENC_CHN](#) VeChn, [VENC_GOP_ATTR_S](#) GopMode);

【参数】

参数名称	描述	输入/输出
VeChn	编码通道号。取值范围：[0, VENC_MAX_CHN_NUM)。	输入
GopMode	GOP属性结构体。	输入

【返回值】

返回值	描述
0	成功。
非0	失败，其值参见 错误码 。

【需求】

头文件：rkmedia_api.h

库文件：libeasymedia.so

【注意】

无。

【举例】

[rkmedia_venc_smartp_test](#)。

【相关主题】

无。

4.3.20 RK_MPI_VENC_RGN_Init

【描述】

初始化OSD。每个编码器Channel，在使用OSD模块前都需调用此函数进行初始化。

【语法】

RK_S32 RK_MPI_VENC_RGN_Init([VENC_CHN](#) VeChn, [VENC_COLOR_TBL_S](#) *stColorTbl);

【参数】

参数名称	描述	输入/输出
VeChn	编码通道号。取值范围：[0, VENC_MAX_CHN_NUM)。	输入
stColorTbl	265色调色板，仅支持ARGB8888格式，设置NULL使用默认调色板。	输入

【返回值】

返回值	描述
0	成功。
非0	失败，其值参见 错误码 。

【需求】

头文件：rkmedia_api.h

库文件：libeasymedia.so

【注意】

在调用[RK_MPI_VENC_RGN_SetBitMap](#)或[RK_MPI_VENC_RGN_SetCover](#)之前，必须先调用该接口，并且每个编码通道只能调用一次。

【举例】

无。

【相关主题】

[RK_MPI_VENC_RGN_SetBitMap](#)

[RK_MPI_VENC_RGN_SetCover](#)

4.3.21 RK_MPI_VENC_RGN_SetBitMap

【描述】

设置OSD位图。仅支持ARGB8888格式位图。

【语法】

RK_S32 RK_MPI_VENC_RGN_SetBitMap([VENC_CHN](#) VeChn, const [OSD_REGION_INFO_S](#) *pstRgnInfo, const [BITMAP_S](#) *pstBitmap);

【参数】

参数名称	描述	输入/输出
VeChn	编码通道号。取值范围：[0, VENC_MAX_CHN_NUM)。	输入
pstRgnInfo	OSD区域信息。	输入
pstBitmap	位图信息和数据。	输入

【返回值】

返回值	描述
0	成功。
非0	失败，其值参见 错误码 。

【需求】

头文件：rkmedia_api.h

库文件：libeasymedia.so

【注意】

在调用此接口之前，必须先调用[RK_MPI_VENC_RGN_Init](#)。该接口与[RK_MPI_VENC_RGN_SetCover](#)共用编码器的8个图层，参见 [OSD_REGION_INFO_S](#)。

【举例】

[rkmedia_venc_jpeg_test](#)。

【相关主题】

[RK_MPI_VENC_RGN_Init](#)

4.3.22 RK_MPI_VENC_RGN_SetCover

【描述】

设置隐私遮挡。设置ARGB8888单色遮挡，效率高于[RK_MPI_VENC_RGN_SetBitMap](#)。

【语法】

RK_S32 RK_MPI_VENC_RGN_SetCover([VENC_CHN](#) VeChn, const [OSD_REGION_INFO_S](#) *pstRgnInfo, const [COVER_INFO_S](#) *pstCoverInfo);

【参数】

参数名称	描述	输入/输出
VeChn	编码通道号。取值范围：[0, VENC_MAX_CHN_NUM)。	输入
pstRgnInfo	OSD区域信息。	输入
pstCoverInfo	隐私遮挡信息。	输入

【返回值】

返回值	描述
0	成功。
非0	失败，其值参见 错误码 。

【需求】

头文件：rkmedia_api.h

库文件：libeasymedia.so

【注意】

在调用此接口之前，必须先调用[RK_MPI_VENC_RGN_Init](#)。该接口与[RK_MPI_VENC_RGN_SetBitMap](#)共用编码器的8个图层，参见 [OSD_REGION_INFO_S](#)。

【举例】

[rkmedia_venc_cover_test](#);

【相关主题】

[RK_MPI_VENC_RGN_Init](#)

4.3.23 RK_MPI_VENC_RGN_SetPalettId

【描述】

使用调色板索引构建buffer用于OSD叠加。无需进行调色板色彩匹配，效率最高。同时使用索引构建buffer，相比ARGB8888格式buffer，内存消耗减少至1/4。

【语法】

```
RK_S32 RK_MPI_VENC_RGN_SetPalettId(VENC\_CHN VeChn, const OSD\_REGION\_INFO\_S
*pstRgnInfo,
const OSD\_COLOR\_PALETTE\_BUF\_S *pstColPalBuf);
```

【参数】

参数名称	描述	输入/输出
VeChn	编码通道号。取值范围：[0, VENC_MAX_CHN_NUM)。	输入
pstRgnInfo	OSD区域信息。	输入
pstColPalBuf	调色板索引构建的OSD Buffer。	输入

【返回值】

返回值	描述
0	成功。
非0	失败，其值参见 错误码 。

【需求】

头文件：rkmedia_api.h

库文件：libeasymedia.so

【注意】

在调用此接口之前，必须先调用[RK_MPI_VENC_RGN_Init](#)。该接口与[RK_MPI_VENC_RGN_SetBitMap](#)共用编码器的8个图层，参见 [OSD_REGION_INFO_S](#)。

【举例】

无。

【相关主题】

无。

4.3.24 RK_MPI_VENC_SetJpegParam

【描述】

设置JPEG编码参数。

【语法】

RK_S32 RK_MPI_VENC_SetJpegParam([VENC_CHN](#) VeChn, const [VENC_JPEG_PARAM_S](#) *pstJpegParam);

【参数】

参数名称	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。	输入

【返回值】

返回值	描述
0	成功。
非0	失败，其值参见 错误码 。

【需求】

头文件：rkmedia_api.h

库文件：libeasymedia.so

【注意】

无。

【举例】

[rkmedia_venc_jpeg_light_test](#)。

【相关主题】

无。

4.3.25 RK_MPI_VENC_StartRecvFrame

【描述】

设置编码器接收帧的数量。默认创建编码器将持续不断的接收VI数据，通过RK_MPI_VENC_StartRecvFrame接口可以设置接收帧数量，到达指定数目后，编码器将休眠，直至下一次调用该接口改变接收帧数目。

【语法】

RK_S32 RK_MPI_VENC_StartRecvFrame([VENC_CHN](#) VeChn, const [VENC_RECV_PIC_PARAM_S](#) *pstRecvParam);

【参数】

参数名称	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，用于指定需要接收的图像帧数。	输入

【返回值】

返回值	描述
0	成功。
非0	失败，其值参见 错误码 。

【需求】

头文件：rkmedia_api.h

库文件：libeasymedia.so

【注意】

无。

【举例】

[rkmedia_venc_jpeg_light_test](#)。

【相关主题】

无。

4.3.26 RK_MPI_VENC_GetFd

【描述】

获取编码器通道的文件描述符。

【语法】

RK_S32 RK_MPI_VENC_GetFd([VENC_CHN](#) VeChn);

【参数】

参数名称	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】

返回值类型	描述
RK_S32	文件描述符

【需求】

头文件：rkmedia_api.h

库文件：libeasymedia.so

【注意】

无。

【举例】

无。

【相关主题】

无。

4.3.27 RK_MPI_VENC_QueryStatus

【描述】

获取编码器通道状态。

【语法】

RK_S32 RK_MPI_VENC_QueryStatus([VENC_CHN](#) VeChn, [VENC_CHN_STATUS_S](#) *pstStatus);

【参数】

参数名称	描述	输入/输出
VeChn	编码通道号。取值范围：[0, VENC_MAX_CHN_NUM)。	输入
pstStatus	编码器状态结构体。	输出

【返回值】

返回值	描述
0	成功。
非0	失败，其值参见 错误码 。

【需求】

头文件：rkmedia_api.h

库文件：libeasymedia.so

【注意】

无。

【举例】

[rkmedia_venc_jpeg_light_test](#)。

【相关主题】

无。

4.4 数据类型

视频编码相关数据类型定义如下：

[VENC_MAX_CHN_NUM](#)：定义 VENC物理通道和扩展通道的总个数。

[VENC_CHN](#)：VENC通道号。

[VENC_ATTR_JPEG_S](#)：定义 JPEG 抓拍编码器属性结构体。

[VENC_ATTR_MJPEG_S](#)：定义 MJPEG 编码器属性结构体。

[VENC_ATTR_H264_S](#)：定义 H.264 编码器属性结构体。

[VENC_ATTR_H265_S](#)：定义 H.265 编码器属性结构体。

[VENC_ATTR_S](#)：定义编码器属性结构体。

[VENC_MJPEG_CBR_S](#)：定义 MJPEG 编码通道 CBR 属性结构。

[VENC_MJPEG_VBR_S](#)：定义 MJPEG 编码通道 VBR 属性结构。

[VENC_H264_CBR_S](#)：定义 H.264 编码通道 CBR 属性结构。

[VENC_H264_VBR_S](#)：定义 H.264 编码通道 VBR 属性结构。

[VENC_H265_CBR_S](#): 定义 H.265 编码通道 CBR 属性结构。

[VENC_H265_VBR_S](#): 定义 H.265 编码通道 VBR 属性结构。

[VENC_RC_MODE_E](#): 定义编码通道码率控制器模式。

[VENC_RC_ATTR_S](#): 定义编码通道码率控制器属性。

[VENC_GOP_MODE_E](#): 定义 Gop Mode 类型。

[VENC_GOP_ATTR_S](#): 定义编码器 GOP 属性结构体。

[VENC_CHN_ATTR_S](#): VENC通道属性结构体。

[VENC_PARAM_MJPEG_S](#): MJPEG 通道参数。

[VENC_PARAM_H264_S](#): H.264 通道参数。

[VENC_PARAM_H265_S](#): H.265 通道参数。

[VENC_RC_PARAM_S](#): 编码通道码率控制器的高级参数。

[VENC_RC_QUALITY_E](#): 编码质量。

[VENC_ROI_ATTR_S](#): ROI 属性结构体。

[OSD_REGION_ID_E](#): OSD区域ID枚举类型。

[OSD_REGION_INFO_S](#): OSD区域信息。

[OSD_PIXEL_FORMAT_E](#): OSD像素格式类型枚举。

[BITMAP_S](#): 位图信息和数据。

[COVER_INFO_S](#): 隐私遮挡信息。

[VENC_RECV_PIC_PARAM_S](#): 接收图像参数结构体指针，用于指定需要接收的图像帧数。

[VENC_JPEG_PARAM_S](#): JPEG 协议编码通道的高级参数。

4.4.1 VENC_MAX_CHN_NUM

【说明】

VENC物理通道和扩展通道的总个数。

【定义】

```
RV1109/RV1126:  
#define VENC_MAX_CHN_NUM 16
```

4.4.2 VENC_CHN

【说明】

VENC通道号。

【定义】

```
typedef RK_S32 VENC_CHN;
```

4.4.3 VENC_ATTR_JPEG_S

【说明】

定义 JPEG 抓拍编码器属性结构体。

【定义】

```
typedef struct rkVENC_ATTR_JPEG_S {
    RK_U32 u32ZoomWidth;    // Zoom to specified width
    RK_U32 u32ZoomHeight;   // Zoom to specified height
    RK_U32 u32ZoomVirWidth;
    RK_U32 u32ZoomVirHeight;
} VENC_ATTR_JPEG_S;
```

【成员】

成员名称	描述
u32ZoomWidth	缩放的指定宽度。
u32ZoomHeight	缩放的指定高度。
u32ZoomVirWidth	缩放的虚拟宽度。
u32ZoomVirHeight	缩放的虚拟高度。

4.4.4 VENC_ATTR_MJPEG_S

【说明】

定义 MJPEG 编码器属性结构体。

【定义】

```
typedef struct rkVENC_ATTR_MJPEG_S {
    RK_U32 u32ZoomWidth;    // Zoom to specified width
    RK_U32 u32ZoomHeight;   // Zoom to specified height
    RK_U32 u32ZoomVirWidth;
    RK_U32 u32ZoomVirHeight;
} VENC_ATTR_MJPEG_S;
```

【成员】

成员名称	描述
u32ZoomWidth	缩放的指定宽度。
u32ZoomHeight	缩放的指定高度。
u32ZoomVirWidth	缩放的虚拟宽度。
u32ZoomVirHeight	缩放的虚拟高度。

4.4.5 VENC_ATTR_H264_S

【说明】

定义 H.264 编码器属性结构体。

【定义】

```
typedef struct rkVENC_ATTR_H264_S {
    RK_U32 u32Level;
    // reserved
} VENC_ATTR_H264_S;
```

【成员】

成员名称	描述
u32Level	Profile IDC值。

4.4.6 VENC_ATTR_H265_S

【说明】

定义 H.265 编码器属性结构体。

【定义】

```
typedef struct rkVENC_ATTR_H265_S {
    // reserved
} VENC_ATTR_H265_S;
```

4.4.7 VENC_ATTR_S

【说明】

定义编码器属性结构体。

【定义】

```
typedef struct rkVENC_ATTR_S {

    CODEC_TYPE_E enType;    // RW; the type of encodec
    IMAGE_TYPE_E imageType; // the type of input image
    RK_U32 u32VirWidth;    // stride width, same to buffer_width, must greater than
                          // width, often set vir_width=(width+15)&(~15)
    RK_U32 u32VirHeight;   // stride height, same to buffer_height, must greater
                          // than height, often set vir_height=(height+15)&(~15)
    RK_U32 u32Profile;     // RW;
                          // H.264:    66: baseline; 77:MP; 100:HP;
                          // H.265:    default:Main;
                          // Jpege/MJpege: default:Baseline
    RK_BOOL bByFrame;      // reserve
    RK_U32 u32PicWidth;    // RW; width of a picture to be encoded, in pixel
```

```

RK_U32 u32PicHeight; // RW; height of a picture to be encoded, in pixel
VENC_ROTATION_E enRotation;
union {
    VENC_ATTR_H264_S stAttrH264e; // attributes of H264e
    VENC_ATTR_H265_S stAttrH265e; // attributes of H265e
    VENC_ATTR_MJPEG_S stAttrMjpege; // attributes of Mjpeg
    VENC_ATTR_JPEG_S stAttrJpege; // attributes of jpeg
};
} VENC_ATTR_S;

```

【成员】

成员名称	描述
enType	编码协议类型。
imageType	输入图像类型。
u32VirWidth	stride宽度，通常与buffer_width相同。若u32VirWidth大于buffer宽度，则必须满足16对齐。
u32VirHeight	stride高度，通常与buffer_height相同。若u32VirHeight大于buffer高度，则必须满足16对齐。
u32Profile	编码的等级。 H.264: 66: Baseline; 77:Main Profile; 100:High Profile; H.265: default:Main; Jpege/MJpege: default:Baseline
bByFrame	预留参数，暂不支持。
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。
stAttrH264e/stAttrH265e/stAttrMjpege/stAttrJpege	某种协议的编码器属性。

【相关数据类型及接口】

[VENC_ATTR_JPEG_S](#)

[VENC_ATTR_MJPEG_S](#)

[VENC_ATTR_H264_S](#)

[VENC_ATTR_H265_S](#)

[VENC_CHN_ATTR_S](#)

4.4.8 VENC_MJPEG_CBR_S

【说明】

定义 MJPEG 编码通道 CBR 属性结构。

【定义】

```
typedef struct hiVENC_MJPEG_CBR_S {
    RK_U32 u32SrcFrameRateNum;
    RK_U32 u32SrcFrameRateDen;
    RK_FR32 fr32DstFrameRateNum;
    RK_FR32 fr32DstFrameRateDen;
    RK_U32 u32BitRate; // RW; Range:[2000, 980000000]; average bitrate
} VENC_MJPEG_CBR_S;
```

【成员】

成员名称	描述
u32SrcFrameRateNum	数据源帧率分子。
u32SrcFrameRateDen	数据源帧率分母。
fr32DstFrameRateNum	目标帧率分子。
fr32DstFrameRateDen	目标帧率分母。
u32BitRate	平均比特率，取值范围：[2000, 980000000]。

4.4.9 VENC_MJPEG_VBR_S

【说明】

定义 MJPEG 编码通道 VBR 属性结构。

【定义】

```
typedef struct hiVENC_MJPEG_VBR_S {
    RK_U32 u32SrcFrameRateNum;
    RK_U32 u32SrcFrameRateDen;
    RK_FR32 fr32DstFrameRateNum;
    RK_FR32 fr32DstFrameRateDen;
    RK_U32 u32BitRate; // RW; Range:[2000, 980000000]; average bitrate
} VENC_MJPEG_VBR_S;
```

【成员】

成员名称	描述
u32SrcFrameRateNum	数据源帧率分子。
u32SrcFrameRateDen	数据源帧率分母。
fr32DstFrameRateNum	目标帧率分子。
fr32DstFrameRateDen	目标帧率分母。
u32BitRate	平均比特率，取值范围：[2000, 98000000]。

4.4.10 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; // RW; Range:[1, 65536]; the interval of I Frame.
    RK_U32 u32SrcFrameRateNum;
    RK_U32 u32SrcFrameRateDen;
    RK_FR32 fr32DstFrameRateNum;
    RK_FR32 fr32DstFrameRateDen;
    RK_U32 u32BitRate; // RW; Range:[2000, 98000000]; average bitrate
} VENC_H264_CBR_S;
```

【成员】

成员名称	描述
u32Gop	I帧间隔，取值范围：[1, 65536]。
u32SrcFrameRateNum	数据源帧率分子。
u32SrcFrameRateDen	数据源帧率分母。
fr32DstFrameRateNum	目标帧率分子。
fr32DstFrameRateDen	目标帧率分母。
u32BitRate	平均比特率，取值范围：[2000, 98000000]，单位：bps

4.4.11 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; // RW; Range:[1, 65536]; the interval of ISLICE.
    RK_U32 u32SrcFrameRateNum;
    RK_U32 u32SrcFrameRateDen;
    RK_FR32 fr32DstFrameRateNum;
    RK_FR32 fr32DstFrameRateDen;
    RK_U32 u32MaxBitRate; // RW; Range:[2000, 98000000];the max bitrate
} VENC_H264_VBR_S;
```

【成员】

成员名称	描述
u32Gop	ISLICE间隔，取值范围：[1, 65536]。
u32SrcFrameRateNum	数据源帧率分子。
u32SrcFrameRateDen	数据源帧率分母。
fr32DstFrameRateNum	目标帧率分子。
fr32DstFrameRateDen	目标帧率分母。
u32MaxBitRate	最大比特率，取值范围：[2000, 98000000]，单位：bps

4.4.12 VENC_H265_CBR_S

【说明】

定义 H.265 编码通道 CBR 属性结构。

【定义】

```
typedef struct rkVENC_H264_CBR_S VENC_H265_CBR_S;
```

【相关数据类型及接口】

[VENC_H264_CBR_S](#)

4.4.13 VENC_H265_VBR_S

【说明】

定义 H.265 编码通道 VBR 属性结构。

【定义】

```
typedef struct rkVENC_H264_VBR_S VENC_H265_VBR_S;
```

【相关数据类型及接口】

[VENC_H264_VBR_S](#)

4.4.14 VENC_RC_MODE_E

【说明】

定义编码通道码率控制器模式。

【定义】

```
typedef enum rkVENC_RC_MODE_E {  
    // H264  
    VENC_RC_MODE_H264CBR = 1,  
    VENC_RC_MODE_H264VBR,  
    VENC_RC_MODE_H264AVBR,  
    // MJPEG  
    VENC_RC_MODE_MJPEGCBR,  
    VENC_RC_MODE_MJPEGVBR,  
    // H265  
    VENC_RC_MODE_H265CBR,  
    VENC_RC_MODE_H265VBR,  
    VENC_RC_MODE_H265AVBR,  
    VENC_RC_MODE_BUTT,  
} VENC_RC_MODE_E;
```

4.4.15 VENC_RC_ATTR_S

【说明】

定义编码通道码率控制器属性。

【定义】

```
typedef struct rkVENC_RC_ATTR_S {  
    /* 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;  
    };  
} VENC_RC_ATTR_S;
```

【成员】

成员名称	描述
enRcMode	编码协议类型。
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。
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。

【相关数据类型及接口】

[VENC_MJPEG_CBR_S](#)

[VENC_MJPEG_VBR_S](#)

[VENC_H264_CBR_S](#)

[VENC_H264_VBR_S](#)

[VENC_H265_CBR_S](#)

[VENC_H265_VBR_S](#)

[VENC_RC_MODE_E](#)

4.4.16 VENC_GOP_MODE_E

【说明】

定义 Gop Mode 类型。

【定义】

```
typedef enum rkVENC_GOP_MODE_E {
    VENC_GOPMODE_NORMALP = 0,
    VENC_GOPMODE_TSVC,
    VENC_GOPMODE_SMARTP,
    VENC_GOPMODE_BUTT,
} VENC_GOP_MODE_E;
```

【注意】

具体模式说明可参考[GOP Mode](#)。

4.4.17 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	编码 GOP 大小。
s32IPQpDelta	I 帧相对 P 帧的 QP 差值。
u32BgInterval	长期参考帧的间隔。
s32ViQpDelta	虚拟 I 帧相对于普通 P 帧的 QP 差值。

【相关数据类型及接口】

[VENC_GOP_MODE_E](#)

4.4.18 VENC_CHN_ATTR_S

【说明】

VENC通道属性结构体。

【定义】

```
typedef struct rkVENC_CHN_ATTR_S {
    VENC_ATTR_S stVencAttr;    // the attribute of video encoder
    VENC_RC_ATTR_S stRcAttr;   // the attribute of rate ctrl
    VENC_GOP_ATTR_S stGopAttr; // the attribute of gop
} 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属性。

【相关数据类型及接口】

[VENC_ATTR_S](#)

[VENC_RC_ATTR_S](#)

4.4.19 VENC_PARAM_MJPEG_S

【说明】

MJPEG 通道参数。

【定义】

```
typedef struct rkVENC_PARAM_MJPEG_S {  
    // reserved  
} VENC_PARAM_MJPEG_S;
```

4.4.20 VENC_PARAM_H264_S

【说明】

H.264 通道参数。

【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  
    // RK_S32 s32MaxReEncodeTimes; /* RW; Range:[0, 3]; Range:max number  
    // of re-encode times.*/  
} VENC_PARAM_H264_S;
```

【成员】

成员名称	描述
u32StepQp	QP的step值。
u32MaxQp	QP最大值，取值范围[8, 51]。
u32MinQp	QP最小值，取值范围[0, 48]，不能大于u32MaxQp。
u32MaxIQp	I帧的QP最大值。
u32MinIQp	I帧的QP最小值。

4.4.21 VENC_PARAM_H265_S

【说明】

H.265 通道参数。

【定义】

```
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

    // RK_S32 s32MaxReEncodeTimes; /* RW; Range:[0, 3]; Range:max number
    // of re-encode times.*/
    // RK_U32 u32DeltIpQp;
} VENC_PARAM_H265_S;
```

【成员】

成员名称	描述
u32StepQp	QP的step值。
u32MaxQp	QP最大值，取值范围[8, 51]。
u32MinQp	QP最小值，取值范围[0, 48]，不能大于u32MaxQp。
u32MaxIQp	I帧的QP最大值。
u32MinIQp	I帧的QP最小值。

4.4.22 VENC_RC_PARAM_S

【说明】

编码通道码率控制器的高级参数。

【定义】

```
typedef struct rkVENC_RC_PARAM_S {
    RK_U32 s32FirstFrameStartQp; // RW; Start QP value of the first frame
    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	H.264 通道参数。
stParamH265	H.265 通道参数。
stParamMjpeg	MJPEG 通道参数。

【相关数据类型及接口】

[VENC_PARAM_H264_S](#)

[VENC_PARAM_H265_S](#)

[VENC_PARAM_MJPEG_S](#)

4.4.23 VENC_RC_QUALITY_E

【说明】

编码质量枚举类型。

【定义】

```
typedef enum rkVENC_RC_QUALITY_E {  
    VENC_RC_QUALITY_HIGHEST,  
    VENC_RC_QUALITY_HIGHER,  
    VENC_RC_QUALITY_HIGH,  
    VENC_RC_QUALITY_MEDIUM,  
    VENC_RC_QUALITY_LOW,  
    VENC_RC_QUALITY_LOWER,  
    VENC_RC_QUALITY_LOWEST,  
    VENC_RC_QUALITY_BUTT,  
} VENC_RC_QUALITY_E;
```

4.4.24 VENC_ROI_ATTR_S

【说明】

ROI 区域参数。

【定义】


```
typedef struct hiVENC_ROI_ATTR_S {
    RK_U32 u32Index; // RW; Range:[0, 7]; Index of an ROI. The system supports
                    // indexes ranging from 0 to 7
    RK_BOOL bEnable; // RW; Range:[0, 1]; Whether to enable this ROI
    RK_BOOL bAbsQp; // RW; Range:[0, 1]; QP mode of an ROI.HI_FALSE: relative
                    // QP.HI_TURE: absolute QP.
    RK_S32 s32Qp; // RW; Range:[-51, 51]; QP value,only relative mode can QP value
                  // less than 0.
    RK_BOOL bIntra; // flag of forced intra macroblock
    RECT_S stRect; // RW; Region of an ROI
} VENC_ROI_ATTR_S;
```

【成员】

成员名称	描述
u32Index	ROI索引值，取值范围[0, 7]。
bEnable	是否使能ROI。
bAbsQp	ROI的QP模式，取值范围：[0, 1]。 1： absolute QP。 0： relative QP。
s32Qp	QP值，取值范围： [-51, 51]。 只有相对模式才能使QP值小于0。
bIntra	强制帧内宏块的标志。
stRect	ROI区域。

【相关数据类型及接口】

[RECT_S](#)

4.4.25 OSD_REGION_ID_E

【说明】

OSD区域ID枚举类型。由0至7叠加优先级逐步上升，优先级越高的OSD位于更高图层。

【定义】

```
typedef enum rkOSD_REGION_ID_E {
    REGION_ID_0 = 0,
    REGION_ID_1,
    REGION_ID_2,
    REGION_ID_3,
    REGION_ID_4,
    REGION_ID_5,
    REGION_ID_6,
    REGION_ID_7
} OSD_REGION_ID_E;
```

4.4.26 OSD_REGION_INFO_S

【说明】

OSD区域信息。

【定义】

```
typedef struct rkOSD_REGION_INFO_S {
    OSD_REGION_ID_E enRegionId;
    RK_U32 u32PosX;
    RK_U32 u32PosY;
    RK_U32 u32Width;
    RK_U32 u32Height;
    RK_U8 u8Inverse;
    RK_U8 u8Enable;
} OSD_REGION_INFO_S;
```

【成员】

成员名称	描述
enRegionId	OSD区域索引值，取值范围[0, 7]。
u32PosX	OSD区域X轴坐标。
u32PosY	OSD区域Y轴坐标。
u32Width	OSD区域宽度。
u32Height	OSD区域高度。
u8Inverse	OSD区域是否反色。
u8Enable	OSD区域是否使能。

【相关数据类型及接口】

[OSD_REGION_ID_E](#)

4.4.27 OSD_PIXEL_FORMAT_E

【说明】

OSD像素格式类型枚举。

【定义】

```
typedef enum rkOSD_PIXEL_FORMAT_E {
    PIXEL_FORMAT_ARGB_8888 = 0,
    PIXEL_FORMAT_BUTT // butt of enum
} OSD_PIXEL_FORMAT_E;
```

4.4.28 VENC_COLOR_TBL_S

【说明】

调色板结构体。

【定义】

```
#define VENC_RGN_COLOR_NUM 256
typedef struct rkVENC_COLOR_TBL {
    // PixFormat: ARGB => A:bit31~bit24 R:bit23~bit16 G:bit15~bit8 B:bit7~bit0
    RK_U32 u32ArgbTbl[VENC_RGN_COLOR_NUM];
    // Enabling dichotomy will speed up the search for the color table,
    // but will sort the color table set by the user in ascending order.
    RK_BOOL bColorDichotomyEnable;
} VENC_COLOR_TBL_S;
```

【成员】

成员名称	描述
u32ArgbTbl	调色板，格式为ARGB8888，最大支持VENC_RGN_COLOR_NUM(256)个
bColorDichotomyEnable	开启二分法优化查询。

4.4.29 OSD_COLOR_PALETTE_BUF_S

【说明】

由调色板索引构建的OSD Buffer。

【定义】

```
typedef struct rkOSD_COLOR_PALETTE_BUF_S {
    RK_U32 u32Width; /* buffer's width */
    RK_U32 u32Height; /* buffer's height */
    RK_VOID *pIdBuf; /* buffer of the color palette id */
} OSD_COLOR_PALETTE_BUF_S;
```

【成员】

成员名称	描述
u32Width	Buffer宽度。
u32Height	Buffer高度
pIdBuf	由8bit调色板索引构建的Buffer指针。

【相关数据类型及接口】

无。

4.4.30 BITMAP_S

【说明】

位图信息和数据。

【定义】

```
typedef struct rkBITMAP_S {
    OSD_PIXEL_FORMAT_E enPixelFormat; /* Bitmap's pixel format */
    RK_U32 u32Width;                  /* Bitmap's width */
    RK_U32 u32Height;                 /* Bitmap's height */
    RK_VOID *pData;                   /* Address of Bitmap's data */
} BITMAP_S;
```

【成员】

成员名称	描述
enPixelFormat	位图像素格式。
u32Width	位图宽度。
u32Height	位图高度。
pData	位图数据的地址。

【相关数据类型及接口】

[OSD_PIXEL_FORMAT_E](#)

4.4.31 COVER_INFO_S

【说明】

隐私遮挡信息。

【定义】

```
typedef struct rkCOVER_INFO_S {
    OSD_PIXEL_FORMAT_E enPixelFormat; /* Bitmap's pixel format */
    RK_U32 u32Color;                  /* Covered area color */
} COVER_INFO_S;
```

【成员】

成员名称	描述
enPixelFormat	位图像素格式。
u32Color	遮挡区域颜色。

【相关数据类型及接口】

[OSD_PIXEL_FORMAT_E](#)

4.4.32 VENC_RECV_PIC_PARAM_S

【说明】

接收图像参数结构体指针，用于指定需要接收的图像帧数。

【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	需要接收的图像帧数。

4.4.33 VENC_JPEG_PARAM_S

【说明】

JPEG 协议编码通道的高级参数。

【定义】

```
typedef struct rkVENC_JPEG_PARAM_S {
    RK_U32 u32Qfactor; // 1-99
    RK_U8 u8YQt[64]; // reserve
    RK_U8 u8CbQt[64]; // reserve
    RK_U8 u8CrQt[64]; // reserve
    RK_U32 u32MCUPerECS; // reserve
} VENC_JPEG_PARAM_S;
```

【成员】

成员名称	描述
u32Qfactor	具体含义请参见 RFC2435 协议，取值范围：[1, 99]。
u8YQt	预留参数，暂未实现。
u8CbQt	预留参数，暂未实现。
u8CrQt	预留参数，暂未实现。
u32MCUPerECS	预留参数，暂未实现。

4.4.34 VENC_RESOLUTION_PARAM_S

【说明】

VENC分辨率配置结构体。

【定义】

```
typedef struct rkVENC_RESOLUTION_PARAM_S {
    RK_U32 u32Width;
    RK_U32 u32Height;
    RK_U32 u32VirWidth;
    RK_U32 u32VirHeight;
} VENC_RESOLUTION_PARAM_S;
```

【成员】

成员名称	描述
u32Width	buffer宽度
u32Height	buffer高度
u32VirWidth	stride宽度，通常与buffer_width相同。 若u32VirWidth大于buffer宽度，则必须满足16对齐。
u32VirHeight	stride高度，通常与buffer_height相同。 若u32VirHeight大于buffer高度，则必须满足16对齐。

4.4.35 VENC_CHN_STATUS_S

【说明】

编码器状态结构体。

【定义】

```
typedef struct rkVENC_CHN_STATUS_S {
    RK_U32 u32LeftFrames; // The number of unencoded frames remaining in the input buffer.
    RK_U32 u32TotalFrames; // The capacity of the input buffer.
    RK_U32 u32LeftPackets; // The number of packets remaining in the output buffer that have not been taken.
    RK_U32 u32TotalPackets; // The capacity of the output buffer.
} VENC_CHN_STATUS_S;
```

【成员】

成员名称	描述
u32LeftFrames	尚未处理帧数。
u32TotalFrames	需处理总帧数。
u32LeftPackets	已处理未输出包数。
u32TotalPackets	总输出包数。

4.5 错误码

视频编码 API 错误码如[表4-1](#)所示：

表4-1 视频编码 API 错误码

错误代码	宏定义	描述
20	RK_ERR_VENC_INVALID_CHNID	通道 ID 超出合法范围
21	RK_ERR_VENC_ILLEGAL_PARAM	参数超出合法范围
22	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
23	RK_ERR_VENC_UNEXIST	试图使用或者销毁不存在的设备、通道或者资源
24	RK_ERR_VENC_NULL_PTR	函数参数中有空指针
25	RK_ERR_VENC_NOT_CONFIG	使用前未配置
26	RK_ERR_VENC_NOT_SUPPORT	不支持的参数或者功能
27	RK_ERR_VENC_NOT_PERM	该操作不允许，如试图修改静态配置参数
28	RK_ERR_VENC_NOBUF	分配缓存失败，如申请的数据缓冲区太大
29	RK_ERR_VENC_BUF_EMPTY	缓冲区中无数据
30	RK_ERR_VENC_BUF_FULL	缓冲区中数据满
31	RK_ERR_VENC_NOTREADY	系统没有初始化或没有加载相应模块
32	RK_ERR_VENC_BUSY	VENC 系统忙

5. 视频解码

5.1 概述

VDEC 模块，即视频解码模块。本模块支持多路实时解码，且每路解码独立，支持 H264/H1265/MJPEG/JPEG 解码。

5.2 API 参考

5.2.1 RK_MPI_VDEC_CreateChn

【描述】

创建解码通道。

【语法】

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

【参数】

参数名称	描述	输入/输出
VdChn	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，其值参见 错误码 。

【需求】

头文件：rkmedia_api.h

库文件：libeasymedia.so

【注意】

无。

【举例】

[rkmedia_vdec_test](#)。

【相关主题】

无。

5.2.2 RK_MPI_VDEC_DestroyChn

【描述】

销毁解码通道。

【语法】

RK_S32 RK_MPI_VDEC_DestroyChn([VDEC_CHN](#) VdChn);

【参数】

参数名称	描述	输入/输出
VdChn	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，其值参见 错误码 。

【需求】

头文件：rkmedia_api.h

库文件：libeasymedia.so

【注意】

无。

【举例】

[rkmedia_vdec_test](#)。

【相关主题】

无。

5.3 数据类型

5.3.1 VDEC_MAX_CHN_NUM

【说明】

VDEC物理通道和扩展通道的总个数。

【定义】

```
RV1109/RV1126:
#define VDEC_MAX_CHN_NUM 16
```

5.3.2 VDEC_CHN

【说明】

VDEC通道号。

【定义】

```
typedef RK_S32 VDEC_CHN;
```

5.3.3 VDEC_CHN_ATTR_S

【说明】

解码通道属性结构体。

【定义】

```
typedef struct rkVDEC_CHN_ATTR_S {
    CODEC_TYPE_E enCodecType;           // RW; video type to be decoded
    IMAGE_TYPE_E enImageType;           // RW; image type to be outputed
    VIDEO_MODE_E enMode;                 // RW; send by stream or by frame
    VIDEO_DECODEC_MODE_E enDecodecMode; // RW; hardware or software
    union {
        VDEC_ATTR_VIDEO_S stVdecVideoAttr; // RW; structure with video
    };
} VDEC_CHN_ATTR_S;
```

【成员】

成员名称	描述
enCodecType	解码格式。
enImageType	解码完成后输出格式。
enMode	解码输入模式，支持帧/流。
enDecodecMode	解码模式，支持硬件或软件解码。
stVdecVideoAttr	解码视频属性结构体。预留属性，暂不支持。

【相关数据类型及接口】

[CODEC_TYPE_E](#)

[IMAGE_TYPE_E](#)

[VIDEO_MODE_E](#)

[VIDEO_DECODEC_MODE_E](#)

[VDEC_ATTR_VIDEO_S](#)

5.3.4 VIDEO_MODE_E

【说明】

输入模式，支持帧/流输入，

【定义】

```
typedef enum rkVIDEO_MODE_E {  
    VIDEO_MODE_STREAM = 0, // send by stream  
    VIDEO_MODE_FRAME,      // send by frame  
    VIDEO_MODE_COMPAT, // Not Support now ! 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;
```

5.3.5 VIDEO_DECODEC_MODE_E

【说明】

解码模式。

【定义】

```
typedef enum rkVIDEO_DECODEC_MODE_E {  
    VIDEO_DECODEC_SOFTWARE = 0,  
    VIDEO_DECODEC_HADRWARE,  
} VIDEO_DECODEC_MODE_E;
```

5.3.6 VDEC_ATTR_VIDEO_S

【说明】

解码视频属性结构体。预留属性，暂不支持。

【定义】

```
typedef struct rkVDEC_ATTR_VIDEO_S {  
  
} VDEC_ATTR_VIDEO_S;
```

6. 移动侦测

6.1 概述

移动侦测（MD）模块实现运动区域检测，最大支持4096个区域。

6.2 功能描述

MD算法由软件实现，输入的分辨率不宜太大，典型分辨率640x480，分辨率越大，CPU负载也高。

6.3 API参考

6.3.1 RK_MPI_ALGO_MD_CreateChn

【描述】

创建MD通道。

【语法】

RK_S32 RK_MPI_ALGO_MD_CreateChn([ALGO_MD_CHN](#) MdChn, const [ALGO_MD_ATTR_S](#) *pstChnAttr);

【参数】

参数名称	描述	输入/输出
MdChn	移动侦测通道号。取值范围：[0, ALGO_MD_MAX_CHN_NUM)。	输入
pstChnAttr	移动侦测通道属性。	输入

【返回值】

返回值	描述
0	成功。
非0	失败，其值参见 错误码 。

【需求】

头文件：rkmedia_api.h

库文件：libeasymedia.so

【注意】

无。

【举例】

[rkmedia_vi_md_test](#)。

【相关主题】

[RK_MPI_ALGO_MD_DestroyChn](#)

6.3.2 RK_MPI_ALGO_MD_DestroyChn

【描述】

销毁MD通道。

【语法】

RK_S32 RK_MPI_ALGO_MD_DestroyChn([ALGO_MD_CHN](#) MdChn);

【参数】

参数名称	描述	输入/输出
MdChn	移动侦测通道号。取值范围：[0, ALGO_MD_MAX_CHN_NUM)。	输入

【返回值】

返回值	描述
0	成功。
非0	失败，其值参见 错误码 。

【需求】

头文件：rkmedia_api.h

库文件：libeasymedia.so

【注意】

无。

【举例】

[rkmedia_vi_md_test](#)。

【相关主题】

[RK_MPI_ALGO_MD_CreateChn](#)

6.3.3 RK_MPI_ALGO_MD_EnableSwitch

【描述】

在保持MD通道开启的条件下，进行MD动态开关。

【语法】

RK_S32 RK_MPI_ALGO_MD_EnableSwitch([ALGO_MD_CHN](#) MdChn, RK_BOOL bEnable)

【参数】

参数名称	描述	输入/输出
MdChn	移动侦测通道号。取值范围：[0, ALGO_MD_MAX_CHN_NUM)。	输入
bEnable	MD开关。	输入

【返回值】

返回值	描述
0	成功。
非0	失败，其值参见 错误码 。

【需求】

头文件：rkmedia_api.h

库文件：libeasymedia.so

【注意】

无。

【举例】

[rkmedia_vi_md_test](#)。

【相关主题】

无。

6.4 数据类型

移动侦测相关数据类型定义如下：

[ALGO_MD_MAX_CHN_NUM](#)：定义移动侦测通道的最大个数。

[ALGO_MD_ROI_RET_MAX](#)：定义移动侦测每个通道的ROI区域最大个数。

[ALGO_MD_CHN](#)：移动侦测通道号。

[ALGO_MD_ATTR_S](#)：定义移动侦测通道属性结构体。

6.4.1 ALGO_MD_MAX_CHN_NUM

【说明】

定义移动侦测通道的最大个数。

【定义】

```
RV1109/RV1126:
#define ALGO_MD_MAX_CHN_NUM VI_MAX_CHN_NUM
```

【相关数据类型及接口】

[VI_MAX_CHN_NUM](#)

6.4.2 ALGO_MD_ROI_RET_MAX

【说明】

定义移动侦测每个通道的ROI区域最大个数。

【定义】

```
RV1109/RV1126:
#define ALGO_MD_ROI_RET_MAX 4096
```

6.4.3 ALGO_MD_CHN

【说明】

移动侦测通道号。

【定义】

```
typedef RK_S32 ALGO_MD_CHN;
```

6.4.4 ALGO_MD_ATTR_S

【说明】

定义移动侦测通道属性结构体。

【定义】

```
typedef struct rkALGO_MD_ATTR_S {
    IMAGE_TYPE_E imageType; // the type of input image
    RK_U32 u32Width;
    RK_U32 u32Height;
    RK_U16 u16RoiCnt; // RW; Range:[0, ALGO_MD_ROI_RET_MAX].
    RECT_S stRoiRects[ALGO_MD_ROI_RET_MAX];
    RK_U16 u16Sensitivity; // value 0(sys default) or [1 - 100].
} ALGO_MD_ATTR_S;
```

【成员】

成员名称	描述
imageType	输入图像类型。
u32Width	移动侦测区域宽度。
u32Height	移动侦测区域高度。
u16RoiCnt	ROI区域个数，取值范围：[0, ALGO_MD_ROI_RET_MAX]。
stRoiRects	ROI区域属性的结构体数组。
u16Sensitivity	移动侦测灵敏度，取值范围：[1, 100]。

【相关数据类型及接口】

[RECT_S](#)

[IMAGE_TYPE_E](#)

[ALGO_MD_ROI_RET_MAX](#)

6.5 错误码

视频编码 API 错误码如[表5-1](#)所示：

表5-1 视频编码 API 错误码

错误代 码	宏定义	描述
70	RK_ERR_ALGO_MD_INVALID_CHNID	通道 ID 超出合法范围
71	RK_ERR_ALGO_MD_BUSY	移动侦测系统忙
72	RK_ERR_ALGO_MD_EXIST	试图申请或者创建已经存在的设备、通道或者资源
73	RK_ERR_ALGO_MD_NOT_CONFIG	使用前未配置
74	RK_ERR_ALGO_MD_ILLEGAL_PARAM	参数超出合法范围

7. 遮挡侦测

7.1 概述

遮挡侦测（Occlusion Detection）模块实现遮挡报警，最大支持10个区域。

7.2 功能描述

OD算法由软件实现，输入的分辨率不宜太大，典型分辨率640x480，分辨率越大，CPU负载也高。

7.3 API参考

7.3.1 RK_MPI_ALGO_OD_CreateChn

【描述】

创建OD通道。

【语法】

RK_S32 RK_MPI_ALGO_OD_CreateChn([ALGO_OD_CHN](#) OdChn, const [ALGO_OD_ATTR_S](#) *pstChnAttr);

【参数】

参数名称	描述	输入/输出
OdChn	遮挡侦测通道号。取值范围：[0, ALGO_OD_MAX_CHN_NUM)。	输入
pstChnAttr	遮挡侦测通道属性。	输入

【返回值】

返回值	描述
0	成功。
非0	失败，其值参见 错误码 。

【需求】

头文件：rkmedia_api.h

库文件：libeasymedia.so

【注意】

无。

【举例】

[rkmedia_vi_od_test](#)。

【相关主题】

[RK_MPI_ALGO_OD_DestroyChn](#)

7.3.2 RK_MPI_ALGO_OD_DestroyChn

【描述】

销毁OD通道。

【语法】

RK_S32 RK_MPI_ALGO_OD_DestroyChn([ALGO_OD_CHN](#) OdChn);

【参数】

参数名称	描述	输入/输出
OdChn	遮挡侦测通道号。取值范围：[0, ALGO_OD_MAX_CHN_NUM)。	输入

【返回值】

返回值	描述
0	成功。
非0	失败，其值参见 错误码 。

【需求】

头文件：rkmedia_api.h

库文件：libeasymedia.so

【注意】

无。

【举例】

[rkmedia_vi_od_test](#)。

【相关主题】

[RK_MPI_ALGO_OD_CreateChn](#)

7.3.3 RK_MPI_ALGO_OD_EnableSwitch

【描述】

在保持OD通道开启的条件下，进行OD动态开关。

【语法】

RK_S32 RK_MPI_ALGO_OD_EnableSwitch([ALGO_OD_CHN](#) OdChn, RK_BOOL bEnable);

【参数】

参数名称	描述	输入/输出
OdChn	遮挡侦测通道号。取值范围：[0, ALGO_OD_MAX_CHN_NUM)。	输入
bEnable	OD开关。	输入

【返回值】

返回值	描述
0	成功。
非0	失败，其值参见 错误码 。

【需求】

头文件：rkmedia_api.h

库文件：libeasymedia.so

【注意】

无。

【举例】

[rkmedia_vi_od_test](#)。

【相关主题】

无。

7.4 数据类型

遮挡侦测相关数据类型定义如下：

[ALGO_OD_MAX_CHN_NUM](#)：定义遮挡侦测通道的最大个数。

[ALGO_OD_ROI_RET_MAX](#)：定义遮挡侦测每个通道的ROI区域最大个数。

[ALGO_OD_CHN](#)：遮挡侦测通道号。

[ALGO_OD_ATTR_S](#)：定义遮挡侦测通道属性结构体。

7.4.1 ALGO_OD_MAX_CHN_NUM

【说明】

定义遮挡侦测通道的最大个数。

【定义】

```
RV1109/RV1126:  
#define ALGO_OD_MAX_CHN_NUM VI_MAX_CHN_NUM
```

【相关数据类型及接口】

[VI_MAX_CHN_NUM](#)

7.4.2 ALGO_OD_ROI_RET_MAX

【说明】

定义遮挡侦测每个通道的ROI区域最大个数。

【定义】

```
RV1109/RV1126:  
#define ALGO_OD_ROI_RET_MAX 10
```

7.4.3 ALGO_OD_CHN

【说明】

遮挡侦测通道号。

【定义】

```
typedef RK_U32 ALGO_OD_CHN;
```

7.4.4 ALGO_OD_ATTR_S

【说明】

定义遮挡侦测通道属性结构体。

【定义】

```
typedef struct rkALGO_OD_ATTR_S {
    IMAGE_TYPE_E enImageType; // the type of input image
    RK_U32 u32Width;
    RK_U32 u32Height;
    RK_U16 u16RoiCnt; // RW; Range:[0, ALGO_OD_ROI_RET_MAX].
    RECT_S stRoiRects[ALGO_OD_ROI_RET_MAX];
    RK_U16 u16Sensitivity; // value 0(sys default) or [1 - 100].
} ALGO_OD_ATTR_S;
```

【成员】

成员名称	描述
enImageType	输入图像类型。
u32Width	遮挡侦测区域宽度。
u32Height	遮挡侦测区域高度。
u16RoiCnt	ROI区域个数，取值范围：[0, ALGO_OD_ROI_RET_MAX]。
stRoiRects	ROI区域属性的结构体数组。
u16Sensitivity	遮挡侦测灵敏度，取值范围：[1, 100]。

【相关数据类型及接口】

[RECT_S](#)

[IMAGE_TYPE_E](#)

[ALGO_OD_ROI_RET_MAX](#)

7.5 错误码

视频编码 API 错误码如[表6-1](#)所示：

表6-1 视频编码 API 错误码

错误代码	宏定义	描述
80	RK_ERR_ALGO_OD_INVALID_CHNID	通道 ID 超出合法范围
81	RK_ERR_ALGO_OD_BUSY	移动侦测系统忙
82	RK_ERR_ALGO_OD_EXIST	试图申请或者创建已经存在的设备、通道或者资源
83	RK_ERR_ALGO_OD_NOT_CONFIG	使用前未配置
84	RK_ERR_ALGO_OD_ILLEGAL_PARAM	参数超出合法范围

8. 视频输出

8.1 概述

VO模块用于视频输出管理。

8.2 功能描述

VO模块是对DRM/KMS的封装，支持多VOP以及多图层显示。

8.3 API参考

8.3.1 RK_MPI_VO_CreateChn

【描述】

创建VO通道。

【语法】

RK_S32 RK_MPI_VO_CreateChn([VO_CHN](#) VoChn, const [VO_CHN_ATTR_S](#) *pstAttr);

【参数】

参数名称	描述	输入/输出
VoChn	VO通道号。取值范围：[0, VO_MAX_CHN_NUM)。	输入
pstAttr	VO通道属性指针。	输入

【返回值】

返回值	描述
0	成功。
非0	失败，其值参见 错误码 。

【需求】

头文件：rkmedia_api.h

库文件：libeasymedia.so

【注意】

如果使用RK_MPI_SYS_SendMediaBuffer将MEDIA_BUFFER送给VO，送显后不能直接使用该MEDIA_BUFFER。否则显示内容会出现异常。这是由于VO将会根据fps不停使用刚才送入的MEDIA_BUFFER刷新VOP，如果刷新过程中改动到MEDIA_BUFFER的内容，则会出现显示异常（花屏、撕裂等现象）。可以通过使用双Buffer机制规避该问题。

VO Plane格式支持说明：

	Primary	Overlay	Curse
rv1109	RGB	NV12	/
rk3399			

【举例】

无。

【相关主题】

[RK_MPI_VO_DestroyChn](#)

8.3.2 RK_MPI_VO_GetChnAttr

【描述】

获取VO通道参数。

【语法】

RK_S32 RK_MPI_VO_GetChnAttr([VO_CHN](#) VoChn, const [VO_CHN_ATTR_S](#) *pstAttr);

【参数】

参数名称	描述	输入/输出
VoChn	VO通道号。取值范围：[0, VO_MAX_CHN_NUM)。	输入
pstAttr	VO通道属性指针。	输出

【返回值】

返回值	描述
0	成功。
非0	失败，其值参见 错误码 。

【需求】

头文件：rkmedia_api.h

库文件：libeasymedia.so

【注意】

无。

【举例】

无。

【相关主题】

[RK_MPI_VO_SetChnAttr](#)

8.3.3 RK_MPI_VO_SetChnAttr

【描述】

设置VO通道参数。

【语法】

RK_S32 RK_MPI_VO_SetChnAttr([VO_CHN](#) VoChn, const [VO_CHN_ATTR_S](#) *pstAttr);

【参数】

参数名称	描述	输入/输出
VoChn	VO通道号。取值范围：[0, VO_MAX_CHN_NUM)。	输入
pstAttr	VO通道属性指针。	输出

【返回值】

返回值	描述
0	成功。
非0	失败，其值参见 错误码 。

【需求】

头文件：rkmedia_api.h

库文件：libeasymedia.so

【注意】

无。

【举例】

无。

【相关主题】

[RK_MPI_VO_GetChnAttr](#)

8.3.4 RK_MPI_VO_DestroyChn

【描述】

销毁VO通道。

【语法】

RK_S32 RK_MPI_VO_DestroyChn([VO_CHN](#) VoChn);

【参数】

参数名称	描述	输入/输出
VoChn	VO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。
非0	失败，其值参见 错误码 。

【需求】

头文件：rkmedia_api.h

库文件：libeasymedia.so

【注意】

无。

【举例】

无。

【相关主题】

[RK_MPI_VO_CreateChn](#)

8.4 数据类型

视频输出相关数据类型定义如下：

[VO_MAX_CHN_NUM](#)：视频输出通道的最大个数。

[VO_CHN](#)：视频输出通道号。

[VO_CHN_ATTR_S](#)：视频输出属性结构体。

8.4.1 VO_MAX_CHN_NUM

【说明】

视频输出通道的最大个数。

【定义】

```
RV1109/RV1126:
#define VO_MAX_CHN_NUM 2
```

8.4.2 VO_CHN

【说明】

视频输出通道号。

【定义】

```
typedef RK_S32 VO_CHN;
```

8.4.3 VO_CHN_ATTR_S

【说明】

视频输出属性结构体。

【定义】

```
typedef enum rk_VO_PLANE_TYPE_E {
    VO_PLANE_PRIMARY = 0,
    VO_PLANE_OVERLAY,
    VO_PLANE_CURSOR,
    VO_PLANE_BUTT
} VO_PLANE_TYPE_E;

typedef struct rkVO_CHN_ATTR_S {
    const RK_CHAR *pcDevNode;
    RK_U16 u16ConIdx; // Connectors idx
    RK_U16 u16EncIdx; // Encoder idx
    RK_U16 u16CrtcIdx; // CRTC idx
    VO_PLANE_TYPE_E emPlaneType;
    IMAGE_TYPE_E enImgType;
    RK_U32 u32Width; // for select display mode. 0 for default mode.
    RK_U32 u32Height; // for select display mode. 0 for default mode.
    RK_U16 u16Fps; // for select display mode. 0 for default mode.
    RK_U16 u16Zpos;
    RECT_S stImgRect; // for input image rect. default equal mode[0]
    RECT_S stDispRect; // for vop display rect. default equal mode[0]
} VO_CHN_ATTR_S;
```

【成员】

成员名称	描述
pcDevNode	视频输出设备节点。
u16ConIdx	conn索引。用于多种显示模式指定。
u16EncIdx	编码器索引。用于多种显示模式指定。
u16CrtcIdx	CRTC索引。用于多种显示模式指定。
emPlaneType	视频输出图层类型。
enImgType	视频输出格式。
u32Width	视频输出宽度。
u32Height	视频输出高度。
u16Fps	视频输出帧率。
u16Zpos	输出图层Z轴高度。
stImgRect	输入图像尺寸参数。
stDispRect	输出图层尺寸参数。用于vop裁剪。

【相关数据类型及接口】

[IMAGE_TYPE_E](#)

[RECT_S](#)

8.5 错误码

视频输出 API 错误码如[表9-1](#)所示：

表9-1 RGA API 错误码

错误代码	宏定义	描述
110	RK_ERR_VO_INVALID_DEVID	设备 ID 超出合法范围
111	RK_ERR_VO_EXIST	试图申请或者创建已经存在的设备、通道或者资源
112	RK_ERR_VO_NOT_CONFIG	使用前未配置
113	RK_ERR_VO_TIMEOUT	视频输出超时
114	RK_ERR_VO_BUF_EMPTY	视频输出缓冲区为空
115	RK_ERR_VO_ILLEGAL_PARAM	非法参数
116	RK_ERR_VO_NOTREADY	系统未初始化

9. 音频

9.1 概述

AUDIO 模块包括音频输入、音频输出、音频编码、音频解码四个子模块。

音频输入和输出模块通过对Linux ALSA音频接口的封装，实现音频输入输出功能。

音频编码和解码模块通过对ffmpeg 音频编码器的封装实现。支持G711A/G711U/G726 /MP2。

9.2 功能描述

9.2.1 音频输入输出

音频输入AI输出AO，用于和 Audio Codec 对接，完成声音的录制和播放。RKMedia AI/AO依赖于Linux ALSA设备，不同的声卡，只要支持ALSA驱动，就可以使用AI/AO接口。AI中集成了音频算法，可通过配置开启。开启算法后，AI输出经过算法处理后的PCM数据。

9.2.2 音频编解码

音频编解码是通过对ffmpeg的封装实现，目前支持G711A/G711U/G726/MP2。

9.2.3 音频算法

目前支持对讲场景AEC算法，录音场景ANR算法。

9.3 API参考

9.3.1 音频输入

9.3.1.1 RK_MPI_AI_EnableChn

【描述】

打开AI通道。

【语法】

RK_S32 RK_MPI_AI_EnableChn([AI_CHN](#) AiChn);

【参数】

参数名称	描述	输入/输出
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，其值参见 错误码 。

【需求】

头文件：rkmedia_api.h

库文件：libeasymedia.so

【注意】

无。

【举例】

[rkmedia_ai_test](#)。

【相关主题】

[RK_MPI_AI_DisableChn](#)

9.3.1.2 RK_MPI_AI_DisableChn

【描述】

关闭AI通道。

【语法】

RK_S32 RK_MPI_AI_DisableChn([AI_CHN](#) AiChn);

【参数】

参数名称	描述	输入/输出
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，其值参见 错误码 。

【需求】

头文件：rkmedia_api.h

库文件：libeasymedia.so

【注意】

无。

【举例】

[rkmedia_ai_test](#)。

【相关主题】

[RK_MPI_AI_EnableChn](#)

9.3.1.3 RK_MPI_AI_SetChnAttr

【描述】

设置AO通道属性。

【语法】

RK_S32 RK_MPI_AI_SetChnAttr([AI_CHN](#) AiChn, const [AI_CHN_ATTR_S](#) *pstAttr);

【参数】

参数名称	描述	输入/输出
AiChn	音频输入通道号。取值范围：[0, AI_MAX_CHN_NUM)。	输入
pstAttr	AI 通道属性指针。	输入

【返回值】

返回值	描述
0	成功。
非0	失败，其值参见 错误码 。

【需求】

头文件：rkmedia_api.h

库文件：libeasymedia.so

【注意】

无。

【举例】

[rkmedia_ai_test](#)。

【相关主题】

无。

9.3.1.4 RK_MPI_AI_SetVolume

【描述】

设置音量。

【语法】

RK_S32 RK_MPI_AI_SetVolume([AI_CHN](#) AiChn, RK_S32 s32Volume);

【参数】

参数名称	描述	输入/输出
AiChn	音频输入通道号。取值范围：[0, AI_MAX_CHN_NUM)。	输入
s32Volume	音频输入通道音量大小。取值范围：[0, 100]。	输入

【返回值】

返回值	描述
0	成功。
非0	失败，其值参见 错误码 。

【需求】

头文件：rkmedia_api.h

库文件：libeasymedia.so

【注意】

无。

【举例】

无。

【相关主题】

无。

9.3.1.5 RK_MPI_AI_GetVolume

【描述】

获取音量。

【语法】

RK_S32 RK_MPI_AI_GetVolume([AI_CHN](#) AiChn, RK_S32 *ps32Volume);

【参数】

参数名称	描述	输入/输出
AiChn	音频输入通道号。取值范围：[0, AI_MAX_CHN_NUM)。	输入
ps32Volume	音频输入通道音量大小。	输出

【返回值】

返回值	描述
0	成功。
非0	失败，其值参见 错误码 。

【需求】

头文件：rkmedia_api.h

库文件：libeasymedia.so

【注意】

无。

【举例】

无。

【相关主题】

无。

9.3.1.6 RK_MPI_AI_SetTalkVqeAttr

【描述】

设置 AI 的声音质量增强功能（Talk）相关属性。

【语法】

RK_S32 RK_MPI_AI_SetTalkVqeAttr([AI_CHN](#) AiChn, [AI_TALKVQE_CONFIG_S](#)*pstVqeConfig);

【参数】

参数名称	描述	输入/输出
AiChn	音频输入通道号。取值范围：[0, AI_MAX_CHN_NUM)。	输入
pstVqeConfig	音频输入声音质量增强配置结构体指针。	输入

【返回值】

返回值	描述
0	成功。
非0	失败，其值参见 错误码 。

【需求】

头文件：rkmedia_api.h

库文件：libeasymedia.so

【注意】

无。

【举例】

[rkmedia_audio_test](#)。

【相关主题】

无。

9.3.1.7 RK_MPI_AI_GetTalkVqeAttr

【描述】

获取 AI 的声音质量增强功能（Talk）相关属性。

【语法】

RK_S32 RK_MPI_AI_GetTalkVqeAttr([AI_CHN](#) AiChn, [AI_TALKVQE_CONFIG_S](#)*pstVqeConfig);

【参数】

参数名称	描述	输入/输出
AiChn	音频输入通道号。取值范围：[0, AI_MAX_CHN_NUM)。	输入
pstVqeConfig	音频输入声音质量增强配置结构体指针。	输出

【返回值】

返回值	描述
0	成功。
非0	失败，其值参见 错误码 。

【需求】

头文件：rkmedia_api.h

库文件：libeasymedia.so

【注意】

无。

【举例】

无。

【相关主题】

无。

9.3.1.8 RK_MPI_AI_SetRecordVqeAttr

【描述】

设置 AI 的声音质量增强功能（Record）相关属性。

【语法】

RK_S32 RK_MPI_AI_SetRecordVqeAttr([AI_CHN](#) AiChn, [AI_RECORDVQE_CONFIG_S](#)*pstVqeConfig);

【参数】

参数名称	描述	输入/输出
AiChn	音频输入通道号。取值范围：[0, AI_MAX_CHN_NUM)。	输入
pstVqeConfig	音频输入声音质量增强配置结构体指针。	输入

【返回值】

返回值	描述
0	成功。
非0	失败，其值参见 错误码 。

【需求】

头文件：rkmedia_api.h

库文件：libeasymedia.so

【注意】

无。

【举例】

[rkmedia_audio_test](#)。

【相关主题】

无。

9.3.1.9 RK_MPI_AI_GetRecordVqeAttr

【描述】

获取 AI 的声音质量增强功能（Record）相关属性。

【语法】

RK_S32 RK_MPI_AI_GetRecordVqeAttr([AI_CHN](#) AiChn, [AI_RECORDVQE_CONFIG_S](#)*pstVqeConfig);

【参数】

参数名称	描述	输入/输出
AiChn	音频输入通道号。取值范围：[0, AI_MAX_CHN_NUM)。	输入
pstVqeConfig	音频输入声音质量增强配置结构体指针。	输出

【返回值】

返回值	描述
0	成功。
非0	失败，其值参见 错误码 。

【需求】

头文件：rkmedia_api.h

库文件：libeasymedia.so

【注意】

无。

【举例】

无。

【相关主题】

无。

9.3.1.10 RK_MPI_AI_EnableVqe

【描述】

使能 AI 的声音质量增强功能。

【语法】

RK_S32 RK_MPI_AI_EnableVqe([AI_CHN](#) AiChn);

【参数】

参数名称	描述	输入/输出
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，其值参见 错误码 。

【需求】

头文件：rkmedia_api.h

库文件：libeasymedia.so

【注意】

无。

【举例】

[rkmedia_audio_test](#)。

【相关主题】

无。

9.3.1.11 RK_MPI_AI_DisableVqe

【描述】

禁用 AI 的声音质量增强功能。

【语法】

RK_S32 RK_MPI_AI_DisableVqe([AI_CHN](#) AiChn);

【参数】

参数名称	描述	输入/输出
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，其值参见 错误码 。

【需求】

头文件：rkmedia_api.h

库文件：libeasymedia.so

【注意】

无。

【举例】

无。

【相关主题】

无。

9.3.1.12 RK_MPI_AI_StartStream

【描述】

启动音频流。

【语法】

RK_S32 RK_MPI_AI_StartStream([AI_CHN](#) AiChn);

【参数】

参数名称	描述	输入/输出
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，其值参见 错误码 。

【需求】

头文件：rkmedia_api.h

库文件：libeasymedia.so

【注意】

无。

【举例】

无。

【相关主题】

无。

9.3.2 音频输出

9.3.2.1 RK_MPI_AO_EnableChn

【描述】

打开AO通道。

【语法】

RK_S32 RK_MPI_AO_EnableChn([AO_CHN](#) AoChn);

【参数】

参数名称	描述	输入/输出
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，其值参见 错误码 。

【需求】

头文件：rkmedia_api.h

库文件：libeasymedia.so

【注意】

无。

【举例】

[rkmedia_ao_test](#)。

【相关主题】

[RK_MPI_AO_DisableChn](#)

9.3.2.2 RK_MPI_AO_DisableChn

【描述】

关闭AO通道。

【语法】

RK_S32 RK_MPI_AO_DisableChn([AO_CHN](#) AoChn);

【参数】

参数名称	描述	输入/输出
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，其值参见 错误码 。

【需求】

头文件：rkmedia_api.h

库文件：libeasymedia.so

【注意】

无。

【举例】

[rkmedia_ao_test](#)。

【相关主题】

[RK_MPI_AO_EnableChn](#)

9.3.2.3 RK_MPI_AO_SetChnAttr

【描述】

设置AO通道属性。

【语法】

```
RK_S32 RK_MPI_AO_SetChnAttr(AO\_CHN AoChn, const AO\_CHN\_ATTR\_S *pstAttr);
```

【参数】

参数名称	描述	输入/输出
AoChn	音频输出通道号。取值范围：[0, AO_MAX_CHN_NUM)。	输入
pstAttr	音频输出通道属性指针。	输入

【返回值】

返回值	描述
0	成功。
非0	失败，其值参见 错误码 。

【需求】

头文件：rkmedia_api.h

库文件：libeasymedia.so

【注意】

无。

【举例】

[rkmedia_ao_test](#)。

【相关主题】

无。

9.3.2.4 RK_MPI_AO_SetVolume

【描述】

设置音量。

【语法】

```
RK_S32 RK_MPI_AO_SetVolume(AO\_CHN AoChn, RK_S32 s32Volume);
```

【参数】

参数名称	描述	输入/输出
AoChn	音频输出通道号。取值范围：[0, AO_MAX_CHN_NUM)。	输入
s32Volume	音频输出通道音量大小。取值范围：[0, 100]。	输入

【返回值】

返回值	描述
0	成功。
非0	失败，其值参见 错误码 。

【需求】

头文件：rkmedia_api.h

库文件：libeasymedia.so

【注意】

无。

【举例】

无。

【相关主题】

无。

9.3.2.5 RK_MPI_AO_GetVolume

【描述】

获取音量。

【语法】

RK_S32 RK_MPI_AO_GetVolume([AO_CHN](#) AoChn, RK_S32 *ps32Volume);

【参数】

参数名称	描述	输入/输出
AoChn	音频输出通道号。取值范围：[0, AO_MAX_CHN_NUM)。	输入
ps32Volume	音频输出通道音量大小。	输出

【返回值】

返回值	描述
0	成功。
非0	失败，其值参见 错误码 。

【需求】

头文件：rkmedia_api.h

库文件：libeasymedia.so

【注意】

无。

【举例】

无。

【相关主题】

无。

9.3.2.6 RK_MPI_AO_SetVqeAttr

【描述】

设置 AO 的声音质量增强功能相关属性。

【语法】

RK_S32 RK_MPI_AO_SetVqeAttr([AO_CHN](#) AoChn, [AO_VQE_CONFIG_S](#) *pstVqeConfig);

【参数】

参数名称	描述	输入/输出
AoChn	音频输出通道号。取值范围：[0, AO_MAX_CHN_NUM)。	输入
pstVqeConfig	音频输出声音质量增强配置结构体指针。	输入

【返回值】

返回值	描述
0	成功。
非0	失败，其值参见 错误码 。

【需求】

头文件：rkmedia_api.h

库文件：libeasymedia.so

【注意】

无。

【举例】

[rkmedia_audio_test](#)。

【相关主题】

无。

9.3.2.7 RK_MPI_AO_GetVqeAttr

【描述】

获取 AO 的声音质量增强功能相关属性。

【语法】

```
RK_S32 RK_MPI_AO_GetVqeAttr(AO\_CHN AoChn, AO\_VQE\_CONFIG\_S *pstVqeConfig);
```

【参数】

参数名称	描述	输入/输出
AoChn	音频输出通道号。取值范围：[0, AO_MAX_CHN_NUM)。	输入
pstVqeConfig	音频输出声音质量增强配置结构体指针。	输出

【返回值】

返回值	描述
0	成功。
非0	失败，其值参见 错误码 。

【需求】

头文件：rkmedia_api.h

库文件：libeasymedia.so

【注意】

无。

【举例】

无。

【相关主题】

无。

9.3.2.8 RK_MPI_AO_EnableVqe

【描述】

使能 AO 的声音质量增强功能。

【语法】

```
RK_S32 RK_MPI_AO_EnableVqe(AO\_CHN AoChn);
```

【参数】

参数名称	描述	输入/输出
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，其值参见 错误码 。

【需求】

头文件：rkmedia_api.h

库文件：libeasymedia.so

【注意】

无。

【举例】

[rkmedia_audio_test](#)。

【相关主题】

无。

9.3.2.9 RK_MPI_AO_DisableVqe

【描述】

禁用 AO 的声音质量增强功能。

【语法】

RK_S32 RK_MPI_AO_DisableVqe([AO_CHN](#) AoChn);

【参数】

参数名称	描述	输入/输出
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，其值参见 错误码 。

【需求】

头文件：rkmedia_api.h

库文件：libeasymedia.so

【注意】

无。

【举例】

无。

【相关主题】

无。

9.3.2.10 RK_MPI_AO_QueryChnStat

【描述】

获取AO通道状态信息。

【语法】

```
RK_S32 RK_MPI_AO_QueryChnStat(AO\_CHN AoChn, AO\_CHN\_STATE\_S *pstStatus);
```

【参数】

参数名称	描述	输入/输出
AoChn	音频输出通道号。取值范围：[0, AO_MAX_CHN_NUM)。	输入
pstStatus	状态信息结构体指针。	输出

【返回值】

返回值	描述
0	成功。
非0	失败，其值参见 错误码 。

【需求】

头文件：rkmedia_api.h

库文件：libeasymedia.so

【注意】

无。

【举例】

无。

【相关主题】

无。

9.3.2.11 RK_MPI_AO_ClearChnBuf

【描述】

清空音频输出通道缓冲区数据。

【语法】

RK_S32 RK_MPI_AO_ClearChnBuf([AO_CHN](#) AoChn);

【参数】

参数名称	描述	输入/输出
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，其值参见 错误码 。

【需求】

头文件：rkmedia_api.h

库文件：libeasymedia.so

【注意】

无。

【举例】

无。

【相关主题】

无。

9.3.3 音频编码

9.3.3.1 RK_MPI_AENC_CreateChn

【描述】

创建音频编码通道。

【语法】

RK_S32 RK_MPI_AENC_CreateChn([AENC_CHN](#) AencChn,const [AENC_CHN_ATTR_S](#) *pstAttr);

【参数】

参数名称	描述	输入/输出
AencChn	音频编码通道号。取值范围：[0, AENC_MAX_CHN_NUM)。	输入
pstAttr	音频编码通道属性指针。	输入

【返回值】

返回值	描述
0	成功。
非0	失败，其值参见 错误码 。

【需求】

头文件：rkmedia_api.h

库文件：libeasymedia.so

【注意】

目前支持协议如[音频编解码](#)所示。

【举例】

[rkmedia_ai_aenc_test](#)。

【相关主题】

[RK_MPI_AENC_DestroyChn](#)

9.3.3.2 RK_MPI_AENC_DestroyChn

【描述】

销毁音频编码通道。

【语法】

RK_S32 RK_MPI_AENC_DestroyChn([AENC_CHN](#) AencChn);

【参数】

参数名称	描述	输入/输出
AencChn	音频编码通道号。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，其值参见 错误码 。

【需求】

头文件：rkmedia_api.h

库文件：libeasymedia.so

【注意】

无。

【举例】

[rkmedia ai aenc test](#)。

【相关主题】

[RK_MPI_AENC_CreateChn](#)

9.3.4 音频解码

9.3.4.1 RK_MPI_ADEC_CreateChn

【描述】

创建音频解码通道。

【语法】

RK_S32 RK_MPI_ADEC_CreateChn([ADEC_CHN](#) AdecChn, const [ADEC_CHN_ATTR_S](#) *pstAttr);

【参数】

参数名称	描述	输入/输出
AdecChn	音频解码通道号。取值范围：[0, ADEC_MAX_CHN_NUM)。	输入
pstAttr	音频解码通道属性指针。	输入

【返回值】

返回值	描述
0	成功。
非0	失败，其值参见 错误码 。

【需求】

头文件：rkmedia_api.h

库文件：libeasymedia.so

【注意】

目前支持协议如[音频编解码](#)所示。

【举例】

[rkmedia adec ao test](#)。

【相关主题】

[RK_MPI_ADEC_DestroyChn](#)

9.3.4.2 RK_MPI_ADEC_DestroyChn

【描述】

销毁音频解码通道。

【语法】

```
RK_S32 RK_MPI_ADEC_DestroyChn(ADEC\_CHN AdecChn);
```

【参数】

参数名称	描述	输入/输出
AdecChn	音频解码通道号。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，其值参见 错误码 。

【需求】

头文件：rkmedia_api.h

库文件：libeasymedia.so

【注意】

无。

【举例】

[rkmedia_adec_ao_test](#)。

【相关主题】

[RK_MPI_ADEC_CreateChn](#)

9.4 数据类型

9.4.1 音频输入

音频输入相关数据类型定义如下：

[AI_MAX_CHN_NUM](#)：音频输入通道的最大个数。

[AI_CHN](#)：音频输入通道号。

[AI_CHN_ATTR_S](#)：音频输入属性结构体。

[AI_TALKVQE_CONFIG_S](#)：音频输入声音质量增强（Talk）配置信息结构体。

[AI_RECORDVQE_CONFIG_S](#)：音频输入声音质量增强（Record）配置信息结构体。

9.4.1.1 AI_MAX_CHN_NUM

【说明】

音频输入通道的最大个数。

【定义】

```
RV1109/RV1126:  
#define AI_MAX_CHN_NUM 1
```

9.4.1.2 AI_CHN

【说明】

音频输入通道号。

【定义】

```
typedef RK_S32 AI_CHN;
```

9.4.1.3 AI_CHN_ATTR_S

【说明】

音频输入属性结构体。

【定义】

```
typedef enum rk_AI_LAYOUT_E {  
    AI_LAYOUT_NORMAL = 0,    /* Normal */  
    AI_LAYOUT_MIC_REF,       /* chanel layout: [mic:ref]; */  
    AI_LAYOUT_REF_MIC,       /* chanel layout: [ref:mic]; */  
    AI_LAYOUT_BUTT  
} AI_LAYOUT_E;  
  
typedef struct rkAI_CHN_ATTR_S {  
    RK_CHAR *pcAudioNode;  
    Sample_Format_E enSampleFormat;  
    RK_U32 u32Channels;  
    RK_U32 u32SampleRate;  
    RK_U32 u32NbSamples;  
    AI_LAYOUT_E enAiLayout;  
} AI_CHN_ATTR_S;
```

【成员】

成员名称	描述
pcAudioNode	音频设备节点路径。
enSampleFormat	采样格式。
u32Channels	通道数。
u32SampleRate	采样率。
u32NbSamples	每帧的采样点个数。
enAiLayout	输入布局类型

【相关数据类型及接口】

[Sample_Format_E](#)

9.4.1.4 AI_TALKVQE_CONFIG_S

【说明】

音频输入声音质量增强（Talk）配置信息结构体。

【定义】

```
#define AI_TALKVQE_MASK_AEC 0x1
#define AI_TALKVQE_MASK_ANR 0x2
#define AI_TALKVQE_MASK_AGC 0x4

typedef struct rkAI_TALKVQE_CONFIG_S {
    RK_U32 u32OpenMask;
    RK_S32 s32WorkSampleRate;
    RK_S32 s32FrameSample;
    RK_CHAR aParamFilePath[MAX_FILE_PATH_LEN];
} AI_TALKVQE_CONFIG_S;
```

【成员】

成员名称	描述
u32OpenMask	Talk Vqe 的各功能使能的 Mask 值。 目前支持AI_TALKVQE_MASK_AEC、 AI_TALKVQE_MASK_ANR、 AI_TALKVQE_MASK_AGC。
s32WorkSampleRate	工作采样频率。
s32FrameSample	采样点数目。
aParamFilePath	参数文件路径。

【相关数据类型及接口】

[MAX_FILE_PATH_LEN](#)

9.4.1.5 AI_RECORDVQE_CONFIG_S

【说明】

音频输入声音质量增强（Record）配置信息结构体。

【定义】

```
#define AI_RECORDVQE_MASK_ANR 0x1

typedef struct rkAI_RECORDVQE_CONFIG_S {
    RK_U32  u32OpenMask;
    RK_S32  s32WorkSampleRate;
    RK_S32  s32FrameSample;
    struct {
        RK_FLOAT fPostAddGain; /* post-gain 0*/
        RK_FLOAT fGmin;        /* spectral gain floor,unit:(dB),default:-30dB */
        RK_FLOAT fNoiseFactor; /* noise suppression factor,default:0.98 */
    } stAnrConfig;
} AI_RECORDVQE_CONFIG_S;
```

【成员】

成员名称	描述
u32OpenMask	Record Vqe 的各功能使能的 Mask 值。 目前支持AI_RECORDVQE_MASK_ANR。
s32WorkSampleRate	工作采样频率。
s32FrameSample	采样点数目。
stAnrConfig.fPostAddGain	ANR的post-gain。
stAnrConfig.fGmin	ANR频谱增益底限，单位为dB，默认值为-30dB。
stAnrConfig.fNoiseFactor	ANR噪音抑制系数，默认值为0.98。

9.4.2 音频输出

音频输出相关数据类型定义如下：

- [AO_MAX_CHN_NUM](#)：音频输出通道的最大个数。
- [AO_CHN](#)：音频输出通道号。
- [AO_CHN_ATTR_S](#)：音频输出属性结构体。
- [AO_VQE_CONFIG_S](#)：音频输出声音质量增强配置信息结构体。

9.4.2.1 AO_MAX_CHN_NUM

【说明】

音频输出通道的最大个数。

【定义】

```
RV1109/RV1126:
#define AO_MAX_CHN_NUM 1
```

9.4.2.2 AO_CHN

【说明】

音频输出通道号。

【定义】

```
typedef RK_S32 AO_CHN;
```

9.4.2.3 AO_CHN_ATTR_S

【说明】

音频输出属性结构体。

【定义】

```
typedef struct rkAO_CHN_ATTR_S {
    RK_CHAR *pcAudioNode;
    Sample_Format_E enSampleFormat;
    RK_U32 u32Channels;
    RK_U32 u32SampleRate;
    RK_U32 u32NbSamples;
} AO_CHN_ATTR_S;
```

【成员】

成员名称	描述
pcAudioNode	音频设备节点路径。
enSampleFormat	采样格式。
u32Channels	通道数。
u32SampleRate	采样率。
u32NbSamples	每帧的采样点个数。

【相关数据类型及接口】

[Sample_Format_E](#)

9.4.2.4 AO_VQE_CONFIG_S

【说明】

音频输出声音质量增强配置信息结构体。

【定义】

```
#define AO_VQE_MASK_ANR      0x1
#define AO_VQE_MASK_AGC      0x2

typedef struct rkAO_VQE_CONFIG_S
{
    RK_U32  u32OpenMask;
    RK_S32  s32WorkSampleRate;
    RK_S32  s32FrameSample;
    RK_CHAR  aParamFilePath[MAX_FILE_PATH_LEN];
} AO_VQE_CONFIG_S;
```

【成员】

成员名称	描述
u32OpenMask	AO Vqe 的各功能使能的 Mask值。 目前支持AO_VQE_MASK_ANR、 AO_VQE_MASK_AGC。
s32WorkSampleRate	工作采样频率。
s32FrameSample	采样点数目。
aParamFilePath	参数文件路径。

【相关数据类型及接口】

[MAX_FILE_PATH_LEN](#)

9.4.2.5 AO_CHN_STATE_S

【说明】

AO通道状态信息结构体。

【定义】

```
typedef struct rkAO_CHN_STATE_S {
    RK_U32  u32ChnTotalNum;
    RK_U32  u32ChnFreeNum;
    RK_U32  u32ChnBusyNum;
} AO_CHN_STATE_S;
```

【成员】

成员名称	描述
u32ChnTotalNum	AO总通道数。
u32ChnFreeNum	AO空闲通道数。
u32ChnBusyNum	AO已占用通道数。

【相关数据类型及接口】

无。

9.4.3 音频编码

音频编码相关数据类型定义如下：

[AENC_MAX_CHN_NUM](#)：音频编码通道的最大个数。

[AENC_CHN](#)：音频编码通道号。

[AENC_ATTR_AAC_S](#)：AAC 编码协议属性结构体。

[AENC_ATTR_MP2_S](#)：MP2 编码协议属性结构体。

[AENC_ATTR_G711A_S](#)：G.711A 编码协议属性结构体。

[AENC_ATTR_G711U_S](#)：G.711U 编码协议属性结构体。

[AENC_ATTR_G726_S](#)：G.726 编码协议属性结构体。

[AENC_CHN_ATTR_S](#)：音频编码属性结构体。

9.4.3.1 AENC_MAX_CHN_NUM

【说明】

音频编码通道的最大个数。

【定义】

```
RV1109/RV1126:
#define AENC_MAX_CHN_NUM 16
```

9.4.3.2 AENC_CHN

【说明】

音频编码通道号。

【定义】

```
typedef RK_S32 AENC_CHN;
```

9.4.3.3 AENC_ATTR_AAC_S

【说明】

AAC 编码协议属性结构体。

【定义】

```
typedef struct rkAENC_ATTR_AAC_S {
    RK_U32 u32Channels;
    RK_U32 u32SampleRate; // 96000, 88200, 64000, 48000, 44100, 32000,
                        // 24000, 22050, 16000, 12000, 11025, 8000, 7350
} AENC_ATTR_AAC_S;
```

【成员】

成员名称	描述
u32Channels	通道数。
u32SampleRate	采样率。取值范围为：96000, 88200, 64000, 48000, 44100, 32000, 24000, 22050, 16000, 12000, 11025, 8000, 7350。

9.4.3.4 AENC_ATTR_MP2_S

【说明】

MP2编码协议属性结构体。

【定义】

```
typedef struct rkAENC_ATTR_MP2_S {
    RK_U32 u32Channels;
    RK_U32 u32SampleRate; // 44100, 48000, 32000, 22050, 24000, 16000, 0
} AENC_ATTR_MP2_S;
```

【成员】

成员名称	描述
u32Channels	通道数。
u32SampleRate	采样率。取值范围为：44100, 48000, 32000, 22050, 24000, 16000, 0。

9.4.3.5 AENC_ATTR_G711A_S

【说明】

G.711A 编码协议属性结构体。

【定义】

```
typedef struct rkAENC_ATTR_G711A_S {
    RK_U32 u32Channels;
    RK_U32 u32SampleRate;
    RK_U32 u32NbSample;
} AENC_ATTR_G711A_S;
```

【成员】

成员名称	描述
u32Channels	通道数。
u32SampleRate	采样率。
u32NbSample	每帧的采样点个数。

9.4.3.6 AENC_ATTR_G711U_S

【说明】

G.711U 编码协议属性结构体。

【定义】

```
typedef struct rkAENC_ATTR_G711U_S {
    RK_U32 u32Channels;
    RK_U32 u32SampleRate;
    RK_U32 u32NbSample;
} AENC_ATTR_G711U_S;
```

【成员】

成员名称	描述
u32Channels	通道数。
u32SampleRate	采样率。
u32NbSample	每帧的采样点个数。

9.4.3.7 AENC_ATTR_G726_S

【说明】

G.726 编码协议属性结构体。

【定义】

```
typedef struct rkAENC_ATTR_G726_S {
    RK_U32 u32Channels;
    RK_U32 u32SampleRate;
} AENC_ATTR_G726_S;
```

【成员】

成员名称	描述
u32Channels	通道数。
u32SampleRate	采样率。

9.4.3.8 AENC_CHN_ATTR_S

【说明】

音频编码属性结构体。

【定义】

```
typedef struct rkAENC_CHN_ATTR_S {
    CODEC_TYPE_E enCodecType; /*payload type */
    RK_U32 u32Bitrate;
    RK_U32 u32Quality;
    union {
        AENC_ATTR_AAC_S stAencAAC;
        AENC_ATTR_MP2_S stAencMP2;
        AENC_ATTR_G711A_S stAencG711A;
        AENC_ATTR_G711U_S stAencG711U;
        AENC_ATTR_G726_S stAencG726;
    };
} AENC_CHN_ATTR_S;
```

【成员】

成员名称	描述
enCodecType	编码协议类型。
u32Bitrate	比特率。
u32Quality	编码质量。
stAencAAC/stAencMP2/stAencG711A/stAencG711U/stAencG726	相关编码协议属性结构体。

【相关数据类型及接口】

[CODEC_TYPE_E](#)

9.4.4 音频解码

音频解码相关数据类型定义如下：

[ADEC_MAX_CHN_NUM](#)：音频解码通道的最大个数。

[ADEC_CHN](#)：音频解码通道号。

[ADEC_ATTR_AAC_S](#)：AAC 解码协议属性结构体。

[ADEC_ATTR_MP2_S](#): MP2 解码协议属性结构体。

[ADEC_ATTR_G711A_S](#): G.711A 解码协议属性结构体。

[ADEC_ATTR_G711U_S](#): G.711U 解码协议属性结构体。

[ADEC_ATTR_G726_S](#): G.726 解码协议属性结构体。

[ADEC_CHN_ATTR_S](#): 音频解码属性结构体。

9.4.4.1 ADEC_MAX_CHN_NUM

【说明】

音频解码通道的最大个数。

【定义】

```
RV1109/RV1126:  
#define ADEC_MAX_CHN_NUM 16
```

9.4.4.2 ADEC_CHN

【说明】

音频解码通道号。

【定义】

```
typedef RK_S32 ADEC_CHN;
```

9.4.4.3 ADEC_ATTR_AAC_S

【说明】

AAC 解码协议属性结构体。

【定义】

```
typedef struct rkADEC_ATTR_AAC_S {  
    // reserved  
} ADEC_ATTR_AAC_S;
```

9.4.4.4 ADEC_ATTR_MP2_S

【说明】

MP2解码协议属性结构体。

【定义】

```
typedef struct rkADEC_ATTR_MP2_S {  
    // reserved  
} ADEC_ATTR_MP2_S;
```

9.4.4.5 ADEC_ATTR_G711A_S

【说明】

G.711A 解码协议属性结构体。

【定义】

```
typedef struct rkADEC_ATTR_G711A_S {  
    RK_U32 u32Channels;  
    RK_U32 u32SampleRate;  
} ADEC_ATTR_G711A_S;
```

【成员】

成员名称	描述
u32Channels	通道数。
u32SampleRate	采样率。

9.4.4.6 ADEC_ATTR_G711U_S

【说明】

G.711U 解码协议属性结构体。

【定义】

```
typedef struct rkADEC_ATTR_G711U_S {  
    RK_U32 u32Channels;  
    RK_U32 u32SampleRate;  
} ADEC_ATTR_G711U_S;
```

【成员】

成员名称	描述
u32Channels	通道数。
u32SampleRate	采样率。

9.4.4.7 ADEC_ATTR_G726_S

【说明】

G.726 解码协议属性结构体。

【定义】

```
typedef struct rkADEC_ATTR_G726_S {  
    // reserved  
} ADEC_ATTR_G726_S;
```

9.4.4.8 ADEC_CHN_ATTR_S

【说明】

音频解码属性结构体。

【定义】

```
typedef struct rkADEC_CHN_ATTR_S {  
    CODEC_TYPE_E enCodecType;  
    union {  
        ADEC_ATTR_AAC_S stAdecAAC;  
        ADEC_ATTR_MP2_S stAdecMP2;  
        ADEC_ATTR_G711A_S stAdecG711A;  
        ADEC_ATTR_G711U_S stAdecG711U;  
        ADEC_ATTR_G726_S stAdecG726;  
    };  
} ADEC_CHN_ATTR_S;
```

【成员】

成员名称	描述
enCodecType	编码协议类型。
stAdecAAC/stAdecMP2/stAdecG711A/stAdecG711U/stAdecG726	相关解码协议属性结构体。

【相关数据类型及接口】

[CODEC_TYPE_E](#)

9.5 错误码

9.5.1 音频输入错误码

音频输入 API 错误码如[表7-1](#)所示：

表7-1 音频输入 API 错误码

错误代码	宏定义	描述
40	RK_ERR_AI_INVALID_DEVID	音频输入设备号无效
41	RK_ERR_AI_BUSY	音频输入系统忙
42	RK_ERR_AI_EXIST	试图申请或者创建已经存在的设备、通道或者资源
43	RK_ERR_AI_NOTOPEN	系统未打开，尚未初始化或使能
44	RK_ERR_AI_NOT_CONFIG	使用前未配置

9.5.2 音频输出错误码

音频输出 API 错误码如[表7-2](#)所示：

表7-2 音频输出 API 错误码

错误代码	宏定义	描述
50	RK_ERR_AO_INVALID_DEVID	音频输出设备号无效
51	RK_ERR_AO_BUSY	音频输出系统未初始化
52	RK_ERR_AO_NOTREADY	试图申请或者创建已经存在的设备、通道或者资源
53	RK_ERR_AO_NOTOPEN	系统未打开，尚未初始化或使能

9.5.3 音频编码错误码

音频编码 API 错误码如[表7-3](#)所示：

表7-3 音频编码 API 错误码

错误代码	宏定义	描述
60	RK_ERR_AENC_INVALID_DEVID	音频编码设备号无效
61	RK_ERR_AENC_BUSY	音频编码系统忙
62	RK_ERR_AENC_CODEC_NOT_SUPPORT	音频编码不支持

9.5.4 音频解码错误码

音频解码 API 错误码如[表7-4](#)所示：

表7-3 音频解码 API 错误码

错误代码	宏定义	描述
100	RK_ERR_ADEC_INVALID_DEVID	音频解码设备号无效
101	RK_ERR_ADEC_BUSY	音频解码系统忙
102	RK_ERR_ADEC_CODEEC_NOT_SUPPORT	音频解码不支持

10. RGA

10.1 概述

RGA模块用于2D图像的裁剪、格式转换、缩放、旋转、图片叠加等。

10.2 功能描述

rkmedia中RGA通道仅支持格式转换、缩放、裁剪、旋转功能，图片叠加则需要单独调用librga.so库，参见docs/Linux/Multimedia/《Rockchip_Developer_Guide_Linux_RGA_CN.pdf》

10.3 API参考

10.3.1 RK_MPI_RGA_CreateChn

【描述】

创建RGA通道。

【语法】

RK_S32 RK_MPI_RGA_CreateChn([RGA_CHN](#) RgaChn, [RGA_ATTR_S](#) *pstRgaAttr);

【参数】

参数名称	描述	输入/输出
RgaChn	RGA通道号。取值范围：[0, RGA_MAX_CHN_NUM)。	输入
pstAttr	RGA通道属性指针。	输入

【返回值】

返回值	描述
0	成功。
非0	失败，其值参见 错误码 。

【需求】

头文件：rkmedia_api.h

库文件：libeasymedia.so

【注意】

无。

【举例】

[rkmedia_vi_vo_test](#)。

【相关主题】

[RK_MPI_RGA_DestroyChn](#)

10.3.2 RK_MPI_RGA_DestroyChn

【描述】

销毁RGA通道。

【语法】

RK_S32 RK_MPI_RGA_DestroyChn([RGA_CHN](#) RgaChn);

【参数】

参数名称	描述	输入/输出
RgaChn	RGA通道号。取值范围：[0, RGA_MAX_CHN_NUM)。	输入

【返回值】

返回值	描述
0	成功。
非0	失败，其值参见 错误码 。

【需求】

头文件：rkmedia_api.h

库文件：libeasymedia.so

【注意】

[rkmedia_vi_vo_test](#)。

【举例】

无。

【相关主题】

[RK_MPI_RGA_CreateChn](#)

10.4 数据类型

RGA相关数据类型定义如下：

[RGA_MAX_CHN_NUM](#)：RGA通道的最大个数。

[RGA_CHN](#)：RGA通道号。

[RGA_INFO_S](#)：RGA区域属性结构体。

[RGA_ATTR_S](#)：RGA属性结构体。

10.4.1 RGA_MAX_CHN_NUM

【说明】

RGA通道的最大个数。

【定义】

```
RV1109/RV1126:  
#define RGA_MAX_CHN_NUM 16
```

10.4.2 RGA_CHN

【说明】

RGA通道号。

【定义】

```
typedef RK_S32 RGA_CHN;
```

10.4.3 RGA_INFO_S

【说明】

RGA区域属性结构体。

【定义】

```
typedef struct rkRGA_INFO_S {  
    IMAGE_TYPE_E imgType;  
    RK_U32 u32X;  
    RK_U32 u32Y;  
    RK_U32 u32Width;  
    RK_U32 u32Height;  
    RK_U32 u32HorStride; // horizontal stride  
    RK_U32 u32VirStride; // virtual stride  
} RGA_INFO_S;
```

【成员】

成员名称	描述
imgType	图像格式类型。
u32X	RGA的X轴坐标。
u32Y	RGA的Y轴坐标。
u32Width	RGA的宽度。
u32Height	RGA的高度。
u32HorStride	虚宽。
u32VirStride	虚高。

【相关数据类型及接口】

[IMAGE_TYPE_E](#)

10.4.4 RGA_ATTR_S

【说明】

RGA属性结构体。

【定义】

```
typedef struct rkRGA_ATTR_S {
    RGA_INFO_S stImgIn;    // input image info
    RGA_INFO_S stImgOut;   // output image info
    RK_U16 u16Rotaion;     // support 0/90/180/270.
    RK_BOOL bEnBufPool;
    RK_U16 u16BufPoolCnt;
} RGA_ATTR_S;
```

【成员】

成员名称	描述
stImgIn	输入图像信息。
stImgOut	输出图像信息。
u16Rotaion	旋转角度。取值范围：0，90，180，270。
bEnBufPool	使能缓冲池。
u16BufPoolCnt	缓冲池计数。

【相关数据类型及接口】

[RGA_INFO_S](#)

10.5 错误码

RGAPI 错误码如表8-1所示：

表8-1 RGAPI 错误码

错误代码	宏定义	描述
90	RK_ERR_RGA_INVALID_CHNID	RGAPI输入设备号无效
91	RK_ERR_RGA_BUSY	RGAPI系统忙
92	RK_ERR_RGA_EXIST	试图申请或者创建已经存在的设备、通道或者资源
93	RK_ERR_RGA_NOT_CONFIG	使用前未配置
94	RK_ERR_RGA_ILLEGAL_PARAM	非法参数

11. 注意事项

11.1 通道析构顺序

需要特别注意的是 rkmedia对模块的析构顺序有特殊的要求：数据流管道中后级模块要先于前级模块销毁。比如：

VI --> RGA --> VENC

则建议析构顺序如下：

destroy VENC

destroy RGA

destroy VI

以VI为例，VI是数据产生端。其生产的buffer在数据管道销毁时可能被后级占用，从而导致VI管理的资源也被占用。再次打开就会遇到Device Busy的错误。这个问题在频繁创建销毁数据通道时有概率发生。

11.2 参数初始化

推荐使用memset将参数初始化为0，避免参数随机初始化带来的影响。

12. Proc调试信息说明

12.1 VI

当VI无数据输出时，查看下述节点信息，判断异常节点所在。

【调试信息】

```
# cif command
cat /proc/rkcif_mipi_lvds | grep "frame amount"; sleep 3; cat
/proc/rkcif_mipi_lvds | grep "frame amount"
# ouput
frame amount:1836735
frame amount:1836826

# isp command
cat /proc/rkisp* | grep Output; sleep 3; cat /proc/rkisp* | grep Output;
# output
Output      rkispp0 ON Format:FBC420 Size:2688x1520 (frame:1837606 rate:32ms)
Output      rkispp_m_bypass Format:NV12 Size:2688x1520 (frame:1837606 rate:31ms
delay:29ms)
Output      rkispp_scale0 Format:NV12 Size:1920x1080 (frame:1837606 rate:31ms
delay:29ms)
Output      rkispp_scale1 Format:NV12 Size:704x576 (frame:1837606 rate:31ms
delay:29ms)
Output      rkispp_scale2 Format:NV12 Size:1280x720 (frame:1837606 rate:31ms
delay:29ms)
Output      rkispp0 ON Format:FBC420 Size:2688x1520 (frame:1837698 rate:33ms)
Output      rkispp_m_bypass Format:NV12 Size:2688x1520 (frame:1837697 rate:32ms
delay:29ms)
Output      rkispp_scale0 Format:NV12 Size:1920x1080 (frame:1837697 rate:32ms
delay:29ms)
Output      rkispp_scale1 Format:NV12 Size:704x576 (frame:1837697 rate:32ms
delay:29ms)
Output      rkispp_scale2 Format:NV12 Size:1280x720 (frame:1837697 rate:32ms
delay:29ms)
```

【调试信息分析】

获取前后frame变化，若frame正常增加，说明该通路能正常传输数据。若frame没有变化，则可能该通路出现异常，数据堵塞。

【参数说明】

参数名	描述
frame amount/frame	输出帧数
rate	输出帧率
Format	输出格式
Size	输出帧大小

说明：未涉及参数，在RKMedia的调试中未使用。

12.2 VENC

【调试信息】

```
# command
cat /proc/mpp_service/session_summary
# output

-----
| session| device| width| height| format| fps_in| fps_out| rc_mode|
bitrate|gop_size|fps_calc| profile|
|8cdb338a| RKVENC| 2688| 1520| avc| 25| 16| vbr|
7549747| 50| 19.49| high|
-----

| session| device|
|0a1be0b6| VEPU2|
-----

| session| device| width| height| format| fps_in| fps_out| rc_mode|
bitrate|gop_size|fps_calc| profile|
|6e6fd71b| RKVENC| 704| 576| avc| 25| 25| cbr|
943718| 50| 30.60| high|
-----

| session| device| width| height| format| fps_in| fps_out| rc_mode|
bitrate|gop_size|fps_calc| profile|
|a87d7eac| RKVENC| 1920| 1080| hevc| 25| 25| cbr|
1887436| 50| 30.51| main|
```

【调试信息分析】

查看VENC各通道参数，可判断创建通道/修改通道参数是否正常。

【参数说明】

参数名	描述
width	分辨率宽
height	分辨率高
format	格式
fps_in	输入帧率
fps_out	输出帧率
rc_mode	码率控制模式
bitrate	码率
gop_size	I帧间隔
fps_calc	实际计算的帧率
profile	编码器profile

说明：未涉及参数，在RKMedia的调试中未使用。

13. LOG调试等级说明

支持动态修改当前各个模块的调试级别。

修改某个模块的调试等级使用 `echo` 命令，例如：

```
echo "venc=3" > /tmp/loglevel
```

修改所有模块的调试等级使用：

```
echo "all=3" > /tmp/loglevel
```

支持[MOD_ID_E](#)中所列模块，模块名称均为小写。

数字0~3分别对应ERROR、WARN、INFO、DEBUG四个级别。

14. 示例

以下提供功能示例，使用注意事项如下：

1. 运行示例前需保证无其他应用占用示例所用节点，如mediaserver。
2. 运行ISP相关示例时，需保证无其他ISP应用运行，如ispserver。

14.1 音频类示例

14.1.1 rkmedia_ai_test

【说明】

由设备输入音频保存至文件。

【代码路径】

external/rkmedia/examples/rkmedia_ai_test.c

【快速使用】

```
./rkmedia_ai_test
```

【选项】

选项	描述	默认值
-d	音频输入节点名	default
-r	输入采样率	16000
-c	输入通道号	2
-o	输出文件路径	/tmp/ai.pcm

【注意】

示例仅支持保存至pcm文件。格式为s16_le。

14.1.2 rkmedia_ai_aenc_test

【说明】

将设备输入的音频编码为aac文件（含adts头）。请先获得FFMPEG AAC编码器授权后，开启ffmpeg中aac编码器选项，然后再运行此示例。

【代码路径】

external/rkmedia/examples/rkmedia_ai_aenc_test.c

【快速使用】

```
./rkmedia_ai_aenc_test
```

【选项】

选项	描述	默认值
-d	音频输入节点名	default
-r	输入采样率	16000
-c	输入通道号	2
-o	输出文件路径	/tmp/aenc.aac

【注意】

格式为s16_le。

14.1.3 rkmedia_ao_test

【说明】

读取音频文件并播放。

【代码路径】

external/rkmedia/examples/rkmedia_ao_test.c

【快速使用】

```
./rkmedia_ao_test -i /tmp/ao.pcm
```

【选项】

选项	描述	默认值
-d	音频输入节点名	default
-r	输入采样率	16000
-c	输入通道号	2
-i	输入文件路径	无
-s	输出帧数	1024

【注意】

示例仅支持保存至pcm文件。格式为s16_le。

14.1.4 rkmedia_adec_ao_test

【说明】

读取aac编码文件（adts封装）解码播放。请先获得FFMPEG AAC解码器授权后，开启ffmpeg中aac解码器选项，然后再运行此示例。

【代码路径】

external/rkmedia/examples/rkmedia_adec_ao_test.c

【快速使用】

```
./rkmedia_adec_ao_test -i /tmp/aenc.aac
```

【选项】

选项	描述	默认值
-d	音频输出节点名	default
-r	输出采样率	16000
-c	输出通道号	2
-i	输入文件路径	/tmp/aac.adts

【注意】

该示例固定编码格式为aac。输出格式为s16_le。

14.1.5 rkmedia_audio_test

【说明】

音频功能演示示例。支持AI->AENC->AO循环，AI->AENC->File，File->ADEC->AO，AI输入Vqe增强后AO输出四种模式。

【代码路径】

external/rkmedia/examples/rkmedia_audio_test.c

【快速使用】

```
# AI->AENC->AO
./rkmedia_audio_test 0 16000

# AI->AENC->File
./rkmedia_audio_test 1 16000

# File->ADEC->AO
./rkmedia_audio_test 2 16000

# AI输入Vqe增强后AO输出
./rkmedia_audio_test 3 16000
```

【选项】

选项	描述	默认值
第一个输入参数	必选项， 0为AI->AENC->AO， 1为AI->AENC->File， 2为File->ADEC->AO， 3为AI输入Vqe增强后AO输出。	无
第二个输入参数	采样率。可选项： 0， 16000， 22050， 24000， 32000， 44100， 48000。	无
第三个输入参数	输出文件路径	/userdata/out.mp2

【注意】

该示例固定编码格式为aac。输出格式为s16_le。

14.2 基础视频类示例

14.2.1 rkmedia_vi_get_frame_test

【说明】

设备输入保存至文件。演示VI没有Bind时如何取视频流。

【代码路径】

external/rkmedia/examples/rkmedia_vi_get_frame_test.c

【快速使用】

```
./rkmedia_vi_get_frame_test
```

【选项】

选项	描述	默认值
-a --aiq	内置ISP功能启用， 输入该选项启用内置ISP功能， 无参数则使用默认值， 参数为aiq文件所在文件夹路径。	/oem/etc/iqfiles/
-h --height	分辨率高。	1080
-w --width	分辨率宽。	1920
-d --device_name	设备节点。	rkispp_scale0
-o --output	输出文件路径，未设置则不输出。	/tmp/1080p.nv12
-c --frame_cnt	输出帧数，设置-I不限制。 未设置输出路径不生效。	-1
-? --help	显示帮助信息。	无

【注意】

输出格式为nv12。

14.2.2 rkmedia_vi_luma_only_mode_test

【说明】

VI亮度统计示例。演示VI在没Bind时如何启用亮度模式。

【代码路径】

external/rkmedia/examples/rkmedia_vi_luma_only_mode_test.c

【快速使用】

```
./rkmedia_vi_luma_only_mode_test
```

【选项】

选项	描述	默认值
-a --aiq	内置ISP功能启用， 输入该选项启用内置ISP功能， 无参数则使用默认值， 参数为aiq文件所在文件夹路径。	/oem/etc/iqfiles/
-? --help	显示帮助选项。	无。

【注意】

无。

14.2.3 rkmedia_vi_multi_bind_test

【说明】

单VI通道绑定多VENC通道演示。

【代码路径】

external/rkmedia/examples/rkmedia_vi_multi_bind_test.c

【快速使用】

```
./rkmedia_vi_multi_bind_test
```

【选项】

选项	描述	默认值
-a --aiq	内置ISP功能启用， 输入该选项启用内置ISP功能， 无参数则使用默认值， 参数为aiq文件所在文件夹路径。	/oem/etc/iqfiles/
-? --help	显示帮助选项。	无。

【注意】

无。

14.2.4 rkmedia_vi_venc_test

【说明】

设备输入编码后保存至文件。

【代码路径】

external/rkmedia/examples/rkmedia_vi_venc_test.c

【快速使用】

```
./rkmedia_vi_venc_test -o /tmp/venc_output.h264
```

【选项】

选项	描述	默认值
-a --aiq	内置ISP功能启用， 输入该选项启用内置ISP功能， 无参数则使用默认值， 参数为aiq文件所在文件夹路径。	/oem/etc/iqfiles/
-h --height	分辨率高。	1080
-w --width	分辨率宽。	1920
-d --device_name	设备节点。	rkispp_scale0
-o --output	输出文件路径。	/tmp/venc_output.h264
-c --frame_cnt	输出帧数。	150
-e --encode	编码格式， 0制定H264编码， 1制定H265编码	0
-? --help	显示帮助选项。	无。

【注意】

无。

14.2.5 rkmedia_vi_vo_test

【说明】

VI经rkmedia内置RGA实现单屏双窗口同时播放。

【代码路径】

external/rkmedia/examples/rkmedia_vi_vo_test.c

【快速使用】

```
./rkmedia_vi_vo_test
```

【选项】

选项	描述	默认值
-a --aiq	内置ISP功能启用， 输入该选项启用内置ISP功能， 无参数则使用默认值， 参数为aiq文件所在文件夹路径。	/oem/etc/iqfiles/
-? --help	显示帮助选项。	无。

【注意】

无。

14.2.6 rkmedia_venc_avbr_test

【说明】

VENC AVBR模式演示。

【代码路径】

external/rkmedia/examples/rkmedia_venc_avbr_test.c

【快速使用】

```
./rkmedia_venc_avbr_test
```

【选项】

选项	描述	默认值
-a --aiq	内置ISP功能启用， 输入该选项启用内置ISP功能， 无参数则使用默认值， 参数为aiq文件所在文件夹路径。	/oem/etc/iqfiles/
-o --output	输出路径。	/userdata/output.h265
-? --help	显示帮助选项。	无。

【注意】

输出格式为H265。

14.2.7 rkmedia_venc_jpeg_light_test

【说明】

jpeg_light通道使用演示示例。启动后输入回车实时进行jpeg截图保存至tmp目录下，输入quit退出。

【代码路径】

external/rkmedia/examples/rkmedia_venc_jpeg_light_test.c

【快速使用】

```
./rkmedia_venc_jpeg_light_test
```

【选项】

选项	描述	默认值
-a --aiq	内置ISP功能启用， 输入该选项启用内置ISP功能， 无参数则使用默认值， 参数为aiq文件所在文件夹路径。	/oem/etc/iqfiles/
-? --help	显示帮助选项。	无。

【注意】

无。

14.2.8 rkmedia_venc_jpeg_test

【说明】

rkmedia内置RGA模块设置位图叠加，并通过jpeg通道保存。启动后输入回车实时进行jpeg截图保存至tmp目录下，输入quit退出。

【代码路径】

external/rkmedia/examples/rkmedia_venc_jpeg_test.c

【快速使用】

```
./rkmedia_venc_jpeg_test
```

【选项】

选项	描述	默认值
-a --aiq	内置ISP功能启用， 输入该选项启用内置ISP功能， 无参数则使用默认值， 参数为aiq文件所在文件夹路径。	/oem/etc/iqfiles/
-w --width	输出分辨率宽。	720
-h --height	输出分辨率高。	480
-W --Width	输入分辨率宽。	1920
-H --Height	输出分辨率高。	1080
-o --output	输出文件夹路径	/tmp/
-? --help	显示帮助选项。	无。

【注意】

输出分辨率不可超过4096*4096。

14.2.9 rkmedia_venc_local_file_test

【说明】

本地nv12文件经venc编码后保存至本地文件。

【代码路径】

external/rkmedia/examples/rkmedia_venc_local_file_test.c

【快速使用】

```
./rkmedia_venc_local_file_test
```

【选项】

选项	描述	默认值
-e --encode	编码格式， 0设定H264编码， 1设定H265编码	0
-i --input	输入路径。	/tmp/1080p.nv12
-o --output	输出路径。	/tmp/output.h264
-? --help	显示帮助选项。	无。

【注意】

无。

14.2.10 rkmedia_venc_smartp_test

【说明】

smartp模式使用示例。

【代码路径】

external/rkmedia/examples/rkmedia_venc_smartp_test.c

【快速使用】

```
./rkmedia_venc_smartp_test
```

【选项】

选项	描述	默认值
-a --aiq	内置ISP功能启用， 输入该选项启用内置ISP功能， 无参数则使用默认值， 参数为aiq文件所在文件夹路径。	/oem/etc/iqfiles/
-o --output	输出文件夹路径，文件名固定	/tmp/output.h264
-? --help	显示帮助选项。	无。

【注意】

输出格式h264。

14.2.11 rkmedia_main_stream_with_jpeg_test

【说明】

bypass节点经VI， VENC输出h265文件。输入回车可进行实时抓拍。

【代码路径】

external/rkmedia/examples/rkmedia_main_stream_with_jpeg_test.c

【快速使用】

```
./rkmedia_main_stream_with_jpeg_test
```

【选项】

选项	描述	默认值
-a --aiq	内置ISP功能启用， 输入该选项启用内置ISP功能， 无参数则使用默认值， 参数为aiq文件所在文件夹路径。	/oem/etc/iqfiles/
-o --output	输出文件夹路径，文件名固定。	/tmp/
-? --help	显示帮助选项。	无。

【注意】

无。

14.2.12 rkmedia_vdec_test

【说明】

输入文件解码后输出。

【代码路径】

external/rkmedia/examples/rkmedia_vdec_test.c

【快速使用】

```
./rkmedia_vdec_test -w 720 -h 480 -p /userdata/out.jpeg -f 0 -t JPEG
```

【选项】

选项	描述	默认值
-w	输入文件分辨率宽	1280
-h	输入文件分辨率高	720
-p	输入文件路径	无
-o	输出文件夹路径， 未指定此参数时，将输出至显示屏， 指定后将输出至文件	无
-f	解码方式，0：软件解码，1：硬件解码	1
-t	输入文件编码格式，支持JPEG，MJPEG，H264，H265	无
-c	输出帧数，设置-1取消限制	-1
-? --help	显示帮助选项。	无。

【注意】

无。

14.2.13 rkmedia_vo_display_test

【说明】

VO播放示例。

【代码路径】

external/rkmedia/examples/rkmedia_vo_display_test.c

【快速使用】

```
./rkmedia_vo_display_test
```

【选项】

选项	描述	默认值
-d	VO输出节点。	/dev/dri/card0
-t	层类型。 可选：Primary，Overlay。	Primary
-s	随机播放坐标宽高。 0：关闭； 1：开启。	0
-x	播放坐标x。	0
-y	播放坐标y。	0
-w	播放宽度。	720
-h	播放高度。	1080
-f	播放频率。	60
-z	播放层z坐标。 可选0，1，2。	0
-?	显示帮助选项。	无。

【注意】

无。

14.2.14 rkmedia_vo_scale_test

【说明】

VI输入，VO输出示例。

【代码路径】

external/rkmedia/examples/rkmedia_vo_scale_test.c

【快速使用】

```
./rkmedia_vo_scale_test
```

【选项】

选项	描述	默认值
-a --aiq	内置ISP功能启用， 输入该选项启用内置ISP功能， 无参数则使用默认值， 参数为aiq文件所在文件夹路径。	/oem/etc/iqfiles/
-? --help	显示帮助选项。	无。

【注意】

无。

14.3 RGA&OSD相关示例

14.3.1 rkmedia_venc_cover_test

【说明】

RKMedia内置RGA API隐私遮盖示例。

【代码路径】

external/rkmedia/examples/rkmedia_venc_cover_test.c

【快速使用】

```
./rkmedia_venc_cover_test
```

【选项】

选项	描述	默认值
-a --aiq	内置ISP功能启用， 输入该选项启用内置ISP功能， 无参数则使用默认值， 参数为aiq文件所在文件夹路径。	/oem/etc/iqfiles/
-o --output	输出路径。	/userdata/output.h264
-? --help	显示帮助选项。	无。

【注意】

输出格式为H264。

14.3.2 rkmedia_venc_mjpeg_test

【说明】

VI输入，经RKMedia内置RGA API设置位图遮盖，再经由VENC编码mjpeg后输出文件。

【代码路径】

external/rkmedia/examples/rkmedia_venc_mjpeg_test.c

【快速使用】

```
./rkmedia_venc_mjpeg_test
```

【选项】

选项	描述	默认值
-a --aiq	内置ISP功能启用， 输入该选项启用内置ISP功能， 无参数则使用默认值， 参数为aiq文件所在文件夹路径。	/oem/etc/iqfiles/
-m --mode	码率类型， cbr设定定码率， vbr设定变码率	vbr
-o --output	输出路径。	/tmp/test.mjpg
-? --help	显示帮助选项。	无。

【注意】

无。

14.3.3 rkmedia_venc_osd_test

【说明】

VI输入，经RKMedia内置RGA API设置随机OSD位图遮盖，再经由VENC编码mjpeg后输出文件。

【代码路径】

external/rkmedia/examples/rkmedia_venc_osd_test.c

【快速使用】

```
./rkmedia_venc_osd_test
```

【选项】

选项	描述	默认值
-a --aiq	内置ISP功能启用， 输入该选项启用内置ISP功能， 无参数则使用默认值， 参数为aiq文件所在文件夹路径。	/oem/etc/iqfiles/
-o --output	输出路径。	/tmp/output.h264
-? --help	显示帮助选项。	无。

【注意】

输出格式h264。

14.3.4 rkmedia_venc_roi_osd_test

【说明】

三路设置不同大小ROI以及OSD叠加，并输出文件。

【代码路径】

external/rkmedia/examples/rkmedia_venc_roi_osd_test.c

【快速使用】

```
./rkmedia_venc_roi_osd_test
```

【选项】

选项	描述	默认值
-a --aiq	内置ISP功能启用， 输入该选项启用内置ISP功能， 无参数则使用默认值， 参数为aiq文件所在文件夹路径。	/oem/etc/iqfiles/
-o --output	输出文件夹路径，文件名固定	/tmp/
-? --help	显示帮助选项。	无。

【注意】

输出格式h264。

14.3.5 rkmedia_rga_api_test

【说明】

VI输入，经外部RGA库裁剪，VO输出示例。

【代码路径】

external/rkmedia/examples/rkmedia_rga_api_test.c

【快速使用】

```
./rkmedia_rga_api_test
```

【选项】

选项	描述	默认值
-a --aiq	内置ISP功能启用， 输入该选项启用内置ISP功能， 无参数则使用默认值， 参数为aiq文件所在文件夹路径。	/oem/etc/iqfiles/
-H --vi_height	VI输入分辨率高。	1920。
-W --vi_width	VI输入分辨率宽。	1080。
-h --crop_height	裁剪分辨率高。	640。
-w --crop_width	裁剪分辨率宽。	640。
-x --crop_x	裁剪坐标X，VI左上角为原点。	300。
-y --crop_y	裁剪坐标Y，VI左上角为原点。	300。
-d --device_name	输入节点。	rkispp_scale0。
-? --help	显示帮助选项。	无。

【注意】

未分配/dev/dri/card0节点设备无法使用。

14.3.6 rkmedia_rga_crop_venc_test

【说明】

VI输入，经外部RGA库裁剪，VENC编码，rtsp直播输出示例。

【代码路径】

external/rkmedia/examples/rkmedia_rga_crop_venc_test.c

【快速使用】

```
./rkmedia_rga_crop_venc_test
```

【选项】

选项	描述	默认值
-a --aiq	内置ISP功能启用， 输入该选项启用内置ISP功能， 无参数则使用默认值， 参数为aiq文件所在文件夹路径。	/oem/etc/iqfiles/
-H --vi_height	VI输入分辨率高。	1920。
-W --vi_width	VI输入分辨率宽。	1080。
-h --crop_height	裁剪分辨率高。	640。
-w --crop_width	裁剪分辨率宽。	640。
-x --crop_x	裁剪坐标X，VI左上角为原点。	300。
-y --crop_y	裁剪坐标Y，VI左上角为原点。	300。
-r --rotation	VENC旋转， 支持0，90，180，270。	0。
-d --device_name	输入节点。	rkispp_scale0。
-? --help	显示帮助选项。	无。

【注意】

直播网址rtsp://<ip>/live/main_stream。

14.3.7 rkmedia_rga_osd_test

【说明】

VI输入，经外部RGA添加OSD叠加，VENC编码，jpeg输出示例。使用透明OSD层与非透明混用可实现OSD文字。

【代码路径】

external/rkmedia/examples/rkmedia_rga_osd_test.c

【快速使用】

```
./rkmedia_rga_osd_test
```

【选项】

选项	描述	默认值
-a --aiq	内置ISP功能启用， 输入该选项启用内置ISP功能， 无参数则使用默认值， 参数为aiq文件所在文件夹路径。	/oem/etc/iqfiles/
-m --mode	模式选择，0为实心长方形模式，1为画框模式。	0。
-r --raw_frame	未经RGA处理帧保存帧数。	0。
-p --process_frame	经RGA处理保存帧数。	1。
-o --output	输出jpeg文件夹路径。	/tmp/。
-? --help	显示帮助选项。	无。

【注意】

无。

14.4 ISP示例

14.4.1 rkmedia_isp_test

【说明】

ISP功能示例，启动后，根据菜单提示切换ISP设置。

【代码路径】

external/rkmedia/examples/rkmedia_isp_test.c

【快速使用】

```
./rkmedia_isp_test
```

【选项】

选项	描述	默认值
-a --aiq	指定aiq文件所在文件夹路径。	/oem/etc/iqfiles/
-? --help	显示帮助选项。	无。

【注意】

此示例运行时需保证未开启ispserver。

14.4.2 rkmedia_vi_double_cameras_test

【说明】

双目ISP示例。

【代码路径】

external/rkmedia/examples/rkmedia_vi_double_cameras_test.c

【快速使用】

```
./rkmedia_vi_double_cameras_test
```

【选项】

选项	描述	默认值
-a	指定aiq文件所在文件夹。 若未指定需在其他应用中开启ISP相关服务。	无
-W	输入视频分辨率宽。	1920
-H	输入视频分辨率高。	1080
-w	输出视频分辨率宽。	720
-h	输出视频分辨率高。	1280
-u	ui z层高度，取值范围[0, 1]	1

【注意】

无。

14.4.3 rkmedia_vi_double_cameras_switch_test

【说明】

双目ISP示例，根据输入参数，可选择VO播放通路。

【代码路径】

external/rkmedia/examples/rkmedia_vi_double_cameras_switch_test.c

【快速使用】

```
./rkmedia_vi_double_cameras_switch_test
```

【选项】

选项	描述	默认值
-a	指定aiq文件所在文件夹。 若未指定需在其他应用中开启ISP相关服务。	无
-W	输入视频分辨率宽。	1920
-H	输入视频分辨率高。	1080
-w	输出视频分辨率宽。	720
-h	输出视频分辨率高。	1280
-u	ui z层高度，取值范围[0, 1]	1
-i	输出channel id，取值范围[0, 1]	0

【注意】

此示例运行时需保证未开启ispserver。

14.5 智能分析使用示例

14.5.1 rkmedia_vi_md_test

【说明】

移动侦测功能示例。

【代码路径】

external/rkmedia/examples/rkmedia_vi_md_test.c

【快速使用】

```
./rkmedia_vi_md_test
```

【选项】

选项	描述	默认值
-a --aiq	内置ISP功能启用， 输入该选项启用内置ISP功能， 无参数则使用默认值， 参数为aiq文件所在文件夹路径。	/oem/etc/iqfiles/
-? --help	显示帮助选项。	无。

【注意】

无。

14.5.2 rkmedia_vi_od_test

【说明】

遮挡检测功能示例。

【代码路径】

external/rkmedia/examples/rkmedia_vi_od_test.c

【快速使用】

```
./rkmedia_vi_od_test
```

【选项】

选项	描述	默认值
-a --aiq	内置ISP功能启用， 输入该选项启用内置ISP功能， 无参数则使用默认值， 参数为aiq文件所在文件夹路径。	/oem/etc/iqfiles/
-? --help	显示帮助选项。	无。

【注意】

无。

14.5.3 rkmedia_vi_rknn_venc_rtsp_test

【说明】

基于rknn_api的人脸识别功能示例。

【代码路径】

external/rkmedia/examples/rkmedia_vi_rknn_venc_rtsp_test.c

【快速使用】

```
./rkmedia_vi_rknn_venc_rtsp_test
```

【选项】

选项	描述	默认值
-a --aiq	内置ISP功能启用， 输入该选项启用内置ISP功能， 无参数则使用默认值， 参数为aiq文件所在文件夹路径。	/oem/etc/iqfiles/
-c --cfg_path	rtsp配置文件。	/oem/usr/share/rtsp-nn.cfg。
-b --box_priors	box priors文件。	/oem/usr/share/rknn_model/box_priors.txt。
-l --labels_list	labels list文件。	/oem/usr/share/rknn_model/coco_labels_list.txt。
-p --ssd_path	ssd文件。	/oem/usr/share/rknn_model/ ssd_inception_v2_rv1109_rv1126.rknn。
-? --help	显示帮助选项。	无。

【注意】

默认RTSP地址：rtsp://<ip>/live/main_stream。

14.5.4 rkmedia_vi_rockx_venc_rtsp_test

【说明】

基于rockx的人脸识别以及人形功能示例。包含识别画框以及识别结果抓拍。

【代码路径】

external/rkmedia/examples/rkmedia_vi_rockx_venc_rtsp_test.c

【快速使用】

```
./rkmedia_vi_rockx_venc_rtsp_test
```

【选项】

选项	描述	默认值
-a --aiq	内置ISP功能启用， 输入该选项启用内置ISP功能， 无参数则使用默认值， 参数为aiq文件所在文件夹路径。	/oem/etc/iqfiles/
-c --cfg_path	rtsp配置文件。	/oem/usr/share/rtsp-nn.cfg
-r --runtime_path	runtime文件路径。	/usr/lib/librknn_runtime.so
-v --data_version	模型库， 0: PERSON_DETECTION_V2; 1: PERSON_DETECTION_V3; 2: FACE_DETECTION_V2; 3: FACE_DETECTION_V3_LARGE;	0
-l --limit_score	检测分数下限。	人形检测：0.45； 人脸检测V2：0.75； 人脸检测V3：0.8。
-p --photo_dirpath	识别抓拍保存路径。	/tmp/
-t --time_log	时间统计日志等级。 0: 禁用； 1: 检测耗时日志； 2: 画框耗时日志； 3: 所有日志。	0
-f --fps	ISP帧率。	30
-m --hdr_mode	HDR模式。 0: 普通模式； 1: HDR2模式。	0
-? --help	显示帮助选项。	无

【注意】

默认RTSP地址：rtsp://<ip>/live/main_stream。