视频输入

文件标识：RK-SYS1-MPI-VI

发布版本：V0.1.0

日　　期：2021.3

文件密级：□绝密　□秘密　□内部资料　■公开

免责声明

商标声明

版权所有 © 2021 瑞芯微电子股份有限公司

瑞芯微电子股份有限公司

Rockchip Electronics Co., Ltd.

地址：　福建省福州市铜盘路软件园A区18号

网址：　www.rock-chips.com

客户服务电话：+86-4007-700-590

客户服务传真：+86-591-83951833

客户服务邮箱：fae@rock-chips.com

前言

概述

本文档主要介绍VI的API和数据类型。

产品版本

芯片名称	内核版本
RV1126	4.19
RK356X	4.19

读者对象

本文档（本指南）主要适用于以下工程师：

技术支持工程师

软件开发工程师

修订记录

版本号	作者	修改日期	修改说明
v0.1.0	李鑫煌	2021-3-30	初始版本

目录

基本概念

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

重要概念

- **DEV设备**

视频输入设备支持若干种时序输入，负责对时序进行解析。

- **PIPE管道**

视频输入 PIPE 绑定在设备后端，负责设备解析后的数据再处理 。（目前暂未实现内容，同dev相同id设置即可）。

- **CHANNEL通道**

视频输入最后一级的获取通道，如dev为isp时输出有四个channel。

举例

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

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

//0. get dev config status
s32Ret = RK_MPI_VI_GetDevAttr(ctx->devId, &ctx->stDevAttr);
if (s32Ret == RK_ERR_VI_NOT_CONFIG) {
    //0-1.config dev
    s32Ret = RK_MPI_VI_SetDevAttr(ctx->devId, &ctx->stDevAttr);
    if (s32Ret != RK_SUCCESS) {
        RK_LOGE("RK_MPI_VI_SetDevAttr %x", s32Ret);
        goto __FAILED1;
    }
} else {
    RK_LOGE("RK_MPI_VI_SetDevAttr already");
}
//1.get dev enable status
s32Ret = RK_MPI_VI_GetDevIsEnable(ctx->devId);
if (s32Ret != RK_SUCCESS) {
    //1-2.enable dev
    s32Ret = RK_MPI_VI_EnableDev(ctx->devId);
    if (s32Ret != RK_SUCCESS) {
        RK_LOGE("RK_MPI_VI_EnableDev %x", s32Ret);
        goto __FAILED1;
    }
    //1-3.bind dev/pipe
    ctx->stBindPipe.u32Num = ctx->pipeId;
```

```
    ctx->stBindPipe.PipeId[0] = ctx->pipeId;
    s32Ret = RK_MPI_VI_SetDevBindPipe(ctx->devId, &ctx->stBindPipe);
    if (s32Ret != RK_SUCCESS) {
        RK_LOGE("RK_MPI_VI_SetDevBindPipe %x", s32Ret);
        goto __FAILED2;
    }
} else {
    RK_LOGE("RK_MPI_VI_EnableDev already");
}
//2.config channel
ctx->stChnAttr.stSize.u32Width = ctx->width;
ctx->stChnAttr.stSize.u32Height = ctx->height;
s32Ret = RK_MPI_VI_SetChnAttr(ctx->pipeId, ctx->channelId, &ctx->stChnAttr);
if (s32Ret != RK_SUCCESS) {
    RK_LOGE("RK_MPI_VI_SetChnAttr %x", s32Ret);
    goto __FAILED2;
}
//3.enable channel
s32Ret = RK_MPI_VI_EnableChn(ctx->pipeId, ctx->channelId);
if (s32Ret != RK_SUCCESS) {
    RK_LOGE("RK_MPI_VI_EnableChn %x", s32Ret);
    goto __FAILED2;
}
//4.save debug file
if (ctx->stDebugFile.bCfg) {
    s32Ret = RK_MPI_VI_ChnSaveFile(ctx->pipeId, ctx->channelId, &ctx-
>stDebugFile);
    RK_LOGE("RK_MPI_VI_ChnSaveFile %x", s32Ret);
}
while (loopCount < ctx->loopCountSet) {
//5.get the frame
    s32Ret = RK_MPI_VI_GetChnFrame(ctx->pipeId, ctx->channelId, &ctx->stViFrame,
waitTime);
    if (s32Ret == RK_SUCCESS) {
        void *data = RK_MPI_MB_Handle2VirAddr(ctx->stViFrame.pMbBlk);
//6.get the channel status
        s32Ret = RK_MPI_VI_QueryChnStatus(ctx->pipeId, ctx->channelId, &ctx-
>stChnStatus);
//7.release the frame
        s32Ret = RK_MPI_VI_ReleaseChnFrame(ctx->pipeId, ctx->channelId, &ctx-
>stViFrame);
        if (s32Ret != RK_SUCCESS) {
            RK_LOGE("RK_MPI_VI_ReleaseChnFrame fail %x", s32Ret);
        }
        loopCount ++;
    } else {
        RK_LOGE("RK_MPI_VI_GetChnFrame timeout %x", s32Ret);
    }
    usleep(10*1000);
}

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

详细测试DEMO，请参考发布文件 : test_mpi_vi.cpp。

API 参考

该功能模块为用户提供以下API:

- RK_MPI_VI_SetDevAttr：设置 VI设备属性。
- RK_MPI_VI_GetDevAttr：获取 VI 设备属性。
- RK_MPI_VI_EnableDev：启用 VI 设备。
- RK_MPI_VI_DisableDev：禁用 VI 设备。
- RK_MPI_VI_GetDevIsEnable：获取 VI 设备是否使能。
- RK_MPI_VI_SetDevBindPipe：绑定dev跟pipe。
- RK_MPI_VI_GetDevBindPipe：获取dev绑定的pipe属性。
- RK_MPI_VI_SetChnAttr：设置VI通道属性。
- RK_MPI_VI_GetChnAttr：获取VI通道属性。
- RK_MPI_VI_EnableChn：启用VI通道。
- RK_MPI_VI_DisableChn：禁用VI通道。
- RK_MPI_VI_GetChnFrame：获取VI通道一帧数据。
- RK_MPI_VI_ReleaseChnFrame：释放VI通道一帧数据。
- RK_MPI_VI_ChnSaveFile：保存VI通道数据。
- RK_MPI_VI_QueryChnStatus：获取VI通道状态。

RK_MPI_VI_SetDevAttr

【描述】
设置VI设备参数。
【语法】

RK_S32 RK_MPI_VI_SetDevAttr(VI_DEV ViDev, const VI_DEV_ATTR_S *pstDevAttr);

【参数】

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

【返回值】

返回值	描述
0	成功。
非0	失败，其值为错误码。

【注意】

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

RK_MPI_VI_GetDevAttr

【描述】
获取VI设备属性。
【语法】

RK_S32 RK_MPI_VI_GetDevAttr(VI_DEV ViDev, VI_DEV_ATTR_S *pstDevAttr);

【参数】

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

【返回值】

返回值	描述
0	成功。
非0	失败，其值为错误码。

【注意】

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

RK_MPI_VI_EnableDev

【描述】
启用VI设备。
【语法】

RK_S32 RK_MPI_VI_EnableDev(VI_DEV ViDev);

【参数】

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

【返回值】

返回值	描述
0	成功。
非0	失败，其值为错误码。

【注意】

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

RK_MPI_VI_DisableDev

【描述】
禁用VI设备。
【语法】

RK_S32 RK_MPI_VI_DisableDev(VI_DEV ViDev);

【参数】

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

【返回值】

返回值	描述
0	成功。
非0	失败，其值为错误码。

【注意】

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

RK_MPI_VI_SetDevBindPipe

【描述】
设置 VI 设备与PIPE 的绑定关系。
【语法】

RK_S32 RK_MPI_VI_SetDevBindPipe(VI_DEV ViDev, const VI_DEV_BIND_PIPE_S *pstDevBindPipe);

【参数】

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

【返回值】

返回值	描述
0	成功。
非0	失败，其值为错误码。

【注意】

- 必须先使能 VI 设备后才能绑定PIPE 。

- 不支持动态绑定。

RK_MPI_VI_GetDevBindPipe

【描述】
获取 VI 设备与PIPE 的绑定关系。
【语法】

RK_S32 RK_MPI_VI_GetDevBindPipe(VI_DEV ViDev, VI_DEV_BIND_PIPE_S *pstDevBindPipe);

【参数】

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

【返回值】

返回值	描述
0	成功。
非0	失败，其值为错误码。

【注意】

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

RK_MPI_VI_GetDevIsEnable

【描述】
获取设备是否使能
【语法】

RK_S32 RK_MPI_VI_GetDevIsEnable(VI_DEV ViDev);

【参数】

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

【返回值】

返回值	描述
0	已经使能。
非0	未使能。

RK_MPI_VI_SetChnAttr

【描述】
设置 VI 通道属性。
【语法】

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

【参数】

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

【返回值】

返回值	描述
0	成功。
非0	失败，其值为错误码。

【注意】

通道属性的各项配置限制如下：

- 目标图像大小 stSize：

  必须配置，且大小需要在 VI 支持的范围内且不超过 VI 的缩放倍数限制。

- 通道buff类型enBufType：

  当采集数据类型为外部申请的内存时，配置为VI_ALLOC_BUF_TYPE_INTERNAL。内部申请内存时配置为VI_ALLOC_BUF_TYPE_INTERNAL。

- 采集数据选项stIspOpt，当采集的数据为isp输入或者直通时，需要配置：

  1. aEntityName：当数据类型为isp直通型时，需要配置。eg：/dev/video0
  2. stMaxSize：必须配置，isp采集数据bypass通路的分辨率支持。

- PIPE 必须已绑定到设备，否则会返回失败 。

RK_MPI_VI_GetChnAttr

【描述】
获取 VI 通道属性。
【语法】

RK_S32 RK_MPI_VI_GetChnAttr(VI_PIPE ViPipe, VI_CHN ViChn, VI_CHN_ATTR_S *pstChnAttr);

【参数】

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

【返回值】

返回值	描述
0	成功。
非0	失败，其值为错误码。

【注意】

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

RK_MPI_VI_EnableChn

【描述】
启用VI通道。
【语法】

RK_S32 RK_MPI_VI_EnableChn(VI_PIPE ViPipe, VI_CHN ViChn);

【参数】

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

【返回值】

返回值	描述
0	成功。
非0	失败，其值为错误码。

【注意】

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

RK_MPI_VI_DisableChn

【描述】
禁用 VI 通道。
【语法】

RK_S32 RK_MPI_VI_DisableChn(VI_PIPE ViPipe, VI_CHN VIChn);

【参数】

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

【返回值】

返回值	描述
0	成功。
非0	失败，其值为错误码。

【注意】

- PIPE 必须已绑定设备，否则会返回失败 。

RK_MPI_VI_GetChnFrame

【描述】
发送VI音频帧。
【语法】

RK_S32 RK_MPI_VI_GetChnFrame(VI_PIPE ViPipe, VI_CHN ViChn, VI_FRAME_S *pstFrameInfo, RK_S32 s32MilliSec);

【参数】

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

【返回值】

返回值	描述
0	成功。
非0	失败，其值为错误码。

【注意】

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

RK_MPI_VI_ReleaseChnFrame

【描述】
禁用 VI 重采样。
【语法】

RK_S32 RK_MPI_VI_ReleaseChnFrame(VI_PIPE ViPipe, VI_CHN ViChn, const VI_FRAME_S *pstFrameInfo);

【参数】

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

【返回值】

返回值	描述
0	成功。
非0	失败，其值为错误码。

【注意】

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

RK_MPI_VI_ChnSaveFile

【描述】
保存 VI 通道数据。
【语法】

RK_S32 RK_MPI_VI_ChnSaveFile(VI_PIPE ViPipe, VI_CHN ViChn, VI_SAVE_FILE_INFO_S *pstSaveFileInfo);

【参数】

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

【返回值】

返回值	描述
0	成功。
非0	失败，其值为错误码。

【注意】

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

RK_MPI_VI_QueryChnStatus

【描述】
查询 VI 通道的状态。
【语法】

RK_S32 RK_MPI_VI_QueryChnStatus(VI_PIPE ViPipe, VI_CHN ViChn, VI_CHN_STATUS_S *pstChnStatus);

【参数】

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

【返回值】

返回值	描述
0	成功。
非0	失败，其值为错误码。

【注意】

- PIPE 必须已绑定设备，否则会返回失败。

- 如果通过RK_MPI_SYS_Bind将VI绑定到了其他设备，则此接口获取不到帧率数据、丢帧数等统计数据。

数据类型

VI_DEV

【说明】
定义 VI 设备句柄。
【定义】

```
typedef RK_S32 VI_DEV;
```

VI_PIPE

【说明】
定义 VI PIPE。
【定义】

```
typedef RK_S32 VI_PIPE;
```

VI_CHN

【说明】
定义 VI 通道。
【定义】

```
typedef RK_S32 VI_CHN;
```

VI_MAX_DEV_NUM

【说明】
定义设备最大个数。
【定义】

```
#define VI_MAX_DEV_NUM  3
```

VI_MAX_PIPE_NUM

【说明】
定义PIPE最大个数。
【定义】

```
#define VI_MAX_PIPE_NUM  VI_MAX_DEV_NUM
```

VI_MAX_DEV_NUM

【说明】
定义音频输出设备的最大个数。
【定义】

```
#define VI_MAX_DEV_NUM 4
```

VI_DEV_ATTR_S

【说明】
定义音频输入输出设备属性结构体。（暂未使用，可以选择传入未初始化的该数据结构）
【定义】

```
/* The attributes of a VI device */
typedef struct rkVI_DEV_ATTR_S {
    /* RW;Interface mode */
    VI_INTF_MODE_E      enIntfMode;
    /* RW;Work mode */
    VI_WORK_MODE_E      enWorkMode;
    /* The below members must be configured in BT.601 mode or DC mode and are
invalid in other modes */
     /* RW;Input data sequence (only the YUV format is supported) */
    VI_YUV_DATA_SEQ_E   enDataSeq;
    /* RW;RGB: CSC-709 or CSC-601, PT YUV444 disable; YUV: default yuv CSC coef
PT YUV444 enable. */
    VI_DATA_TYPE_E      enInputDataType;
    /* RW;Input max size */
    SIZE_S              stMaxSize;
    /* RW;Data rate of Device */
    DATA_RATE_E         enDataRate;
} VI_DEV_ATTR_S;
```

【成员】

成员名称	描述
enIntfMode	接口模式。（暂未使用，可不设置）
enWorkMode	1、2、4 路复合工作模式。（暂未使用，可不设置）
enDataSeq	输入数据顺序 (仅支持 yuv 格式)，当 enIntfMode 为 VI_MODE_BT656 或者 VI_MODE_BT601 时取值范围为 [VI_DATA_SEQ_UYVY, VI_DATA_SEQ_YVYU]，当 enIntfMod 为 VI_MODE_BT1120_STANDARD，VI_MODE_MIPI_YUV420_NORMAL，VI_MODE_MIPI_YUV420_LEGACY，VI_MODE_MIPI_YUV422 时取值必须为 VI_DATA_SEQ_VUVU 或者 VI_DATA_SEQ_UVUV。（暂未使用，可不设置）
enInputDataType	输入数据类型，Sensor 输入一般为 RGB，AD 输入一般为YUV。（暂未使用，可不设置）
stMaxSize	捕获图像的最大宽高。（暂未使用，可不设置）
enDataRate	设备的速率。（暂未使用，可不设置）

VI_DEV_BIND_PIPE_S

【说明】
定义 VI DEV 与 PIPE 的绑定关系。
【定义】

```
/* Information of pipe binded to device */
typedef struct rkVI_DEV_BIND_PIPE_S {
    RK_U32  u32Num;                                     /* RW;Range
[1,VI_MAX_PHY_PIPE_NUM] */
    VI_PIPE PipeId[VI_MAX_PHY_PIPE_NUM];                /* RW;Array of pipe ID
*/
} VI_DEV_BIND_PIPE_S;
```

【成员】

成员名称	描述
u32Num	该 VI Dev 所绑定的 PIPE 数目，取值范围[1，VI_MAX_PIPE_NUM]。
PipeId	该 VI Dev 绑定的 PIPE 号。

【注意事项】

- 目前PIPE传入跟DEV一样的数值即可。

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

VI_CHN_ATTR_S

【说明】
定义 VI 通道属性。
【定义】

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

【成员】

成员名称	描述
stSize	获取通道图像宽高大小。
enPixelFormat	目标图像像素格式。（暂未使用，可不设置）
enDynamicRange	目标图像动态范围。（暂未使用，可不设置）
enVideoFormat	目标图像视频数据格式。（暂未使用，可不设置）
enCompressMode	目标图像压缩格式。
bMirror	Mirror 使能开关。 RK_FALSE：不使能； RK_TRUE：使能。（暂未使用，可不设置）
bFlip	Flip 使能开关。 RK_FALSE：不使能； RK_TRUE：使能。（暂未使用，可不设置）
u32Depth	用户获取图像的队列深度。（暂未使用，可不设置）
stFrameRate	帧率控制。（暂未使用，可不设置）
enAllocBufType	图像内存申请类型。 VI_ALLOC_BUF_TYPE_INTERNAL：内部申请。 VI_ALLOC_BUF_TYPE_EXTERNAL：外部申请。 当图像类型为mmap方式获取时需要设置为外部申请。
stIspOpt	获取图像为isp处理/直通数据的参数设置。

SIZE_S

【说明】
定义图像大小信息结构体。
【定义】

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

【成员】

成员名称	描述
u32Width	宽度。
u32Height	高度。

COMPRESS_MODE_E

【说明】
定义音频码流结构体。
【定义】

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

【成员】

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

VI_ALLOC_BUF_TYPE_E

【说明】
定义VI图像内存分配类型。
【定义】

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

【成员】

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

VI_ISP_OPT_S

【说明】
获取图像为isp处理/直通数据的参数设置。
【定义】

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

【成员】

成员名称	描述
u32BufCount	输出通道总的缓存块数。
u32BufSize	每个缓存块申请的缓存大小。
enCaptureType	获取图像的V4L2录制类型。
enMemoryType	获取图像的V4L2内存类型。
aEntityName	通道设备名字。
bNoUseLibV4L2	是否使用libV4L2（暂时只支持开启）。
stMaxSize	通道获取图像大小设置。

VI_V4L2_CAPTURE_TYPE

【说明】
获取图像的V4L2录制类型。
【定义】

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

【注意事项】

- 同V4L2_CAPTURE_TYPE_VIDEO_CAPTURE设置。

VI_V4L2_MEMORY_TYPE

【说明】
获取图像的V4L2内存类型
【定义】

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

【成员】

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

错误码

音频输出API错误码如下所示。

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