CV180x/CV181x 媒体软件开发指南

Version: 1.0.1.1

Release date: 2022-08-16

目录

修订记录

Revision	Date	Description
1.0.0.0	2022/6/1	Initial version
1.0.0.1	2022/6/17	Update Audio API for kernel mode
1.0.1.2	2022/8/16	Update for CV181x final release
1.0.1.3	2022/10/16	Update for CV180x
1.0.1.4	2023/7/18	Update for dual OS

1 声明

法律声明

本数据手册包含北京晶视智能科技有限公司（下称“晶视智能”）的保密信息。未经授权，禁止使用或披露本数据手册中包含的信息。如您未经授权披露全部或部分保密信息，导致晶视智能遭受任何损失或损害，您应对因之产生的损失/损害承担责任。

本文件内信息如有更改，恕不另行通知。晶视智能不对使用或依赖本文件所含信息承担任何责任。

本数据手册和本文件所含的所有信息均按“原样”提供，无任何明示、暗示、法定或其他形式的保证。晶视智能特别声明未做任何适销性、非侵权性和特定用途适用性的默示保证，亦对本数据手册所使用、包含或提供的任何第三方的软件不提供任何保证；用户同意仅向该第三方寻求与此相关的任何保证索赔。此外，晶视智能亦不对任何其根据用户规格或符合特定标准或公开讨论而制作的可交付成果承担责任。

联系我们

地址 北京市海淀区丰豪东路 9 号院中关村集成电路设计园（ICPARK）1 号楼

深圳市宝安区福海街道展城社区会展湾云岸广场 T10 栋

电话 +86-10-57590723 +86-10-57590724

邮编 100094（北京）518100（深圳）

官方网站 https://www.sophgo.com/

技术论坛 https://developer.sophgo.com/forum/index.html

2 系统概述

2.1 功能概述

2.1.1 目的

CVITEK 所提供的多媒体软件架构 (Multimedia Framework，简称 MMF)，用以缩短应用程序开发所需的时间。

此架构屏蔽了处理器端的复杂底层设计和差异，对应用程序提供统一且便捷的 MMF Programming Interface 编程接口。

MMF 包含了以下功能：ISP 影像前处理 (包含 HDR、去噪、边缘锐化等)、输入影像撷取及输出、图像几何校正、H.265/H.264/JPEG 编解码、音频撷取及输出、音频编解码等。

2.1.2 定义及缩写

- MMF(Multimedia Framework 多媒体软件架构)
- ISP(Image Signal Processor 图像信号处理器)
- VI(Video Input 影像输入)
- VPSS(Video Process Sub-System 图像处理子系统)
- VO(Video Output 影像输出)
- VDEC(Video Decoder 影像解码)
- VENC(Video Encoder 影像编码)
- ADEC(音频解码)
- AENC(音频编码)
- REGION(区域管理)

2.2 设计概述

2.2.1 系统架构

图 2.1 为 MMF 的系统架构。由下而上分别为:

- 硬件层 HW

由 CVITEK SoC 加上外围组件组成。外围组件包含 Flash、DDR、视频 Sensor、音频 AD 等。

- 驱动层 Driver

控制 HW 的驱动程序。

- 系统层 OS

基于 Linux/AliOS 的操作系统。

- 输入输出控制 Ioctl

用以控制 SDK 涵盖范围以外的组件，例如 MIPI_RX，MIPI_TX。

- 系统开发工具包 SDK

屏蔽了硬件的细节和差异，提供统一 API 以供开发。

- 应用层 Application

基于 SDK 和 ioctl，由用户开发的应用程序。

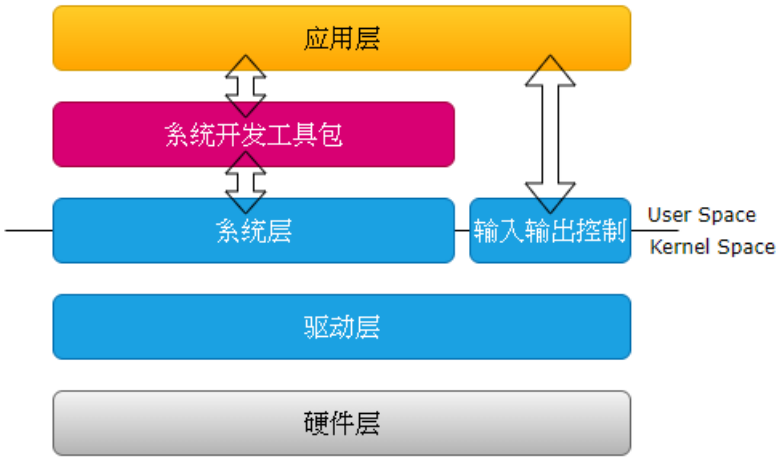


图 2.1: MMF 系统架构

图 2.2 为 CVITEK 媒体处理平台的主要内部处理流程。

其中包含多个组件;

- VI 捕捉视频图像，可对其做剪切、影像优化等处理后，再将图像数据传递给 VPSS 处理。
- VDEC 将编码后的码流译码，再将图像数据传递给 VPSS 处理。

- VPSS 接收 VI 或 VDEC 发送的图像，并可同时输出多个不同分辨率的图像，以供预览、编码或抓拍。
- VO 接收 VPSS 处理后的图像，并根据设定的时序输出到显示设备。
- REGION 可以将用户所指定的位图（Bitmap）作为 OSD 迭加到图像数据上。

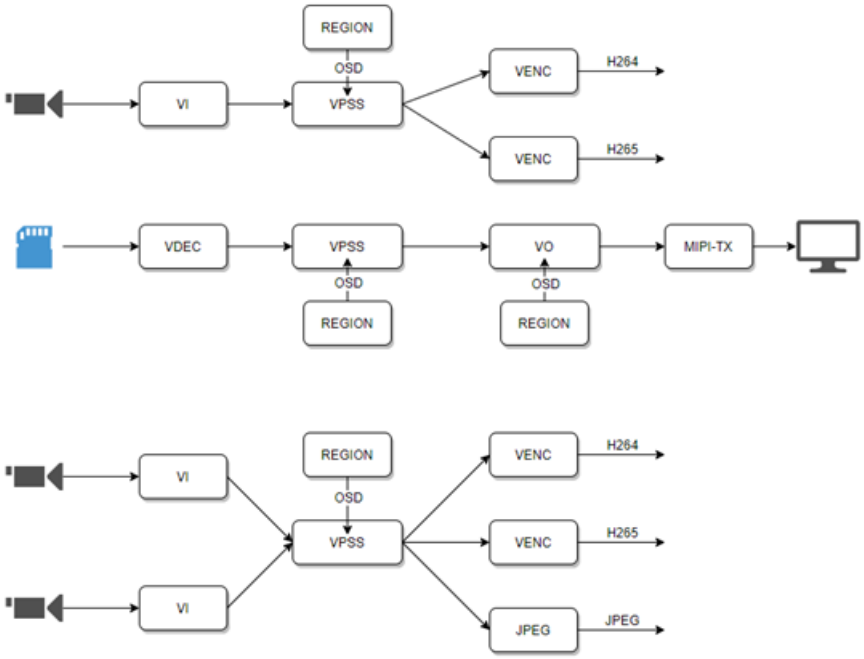


图 2.2: CVITEK 媒体处理平台的主要内部处理流程

3 系统控制

3.1 功能概述

3.1.1 目的

系统控制根据各个处理器的特性，完成硬件各个部件的复位、基本初始化工作，同时负责完成 MMF 系统各个功能模块的初始化、控制、去初始化以及管理 MMF 系统各个功能模块的工作状态、提供大块物理内存管理等功能。

应用程序启动 MMF 功能前，首先必须完成 MMF 系统初始化工作。同理，应用程序退出 MMF 功能后，也要完成 MMF 系统去初始化工作，释放资源。

3.1.2 定义及缩写

- MMF（Multimedia Framework 多媒体软件架构）
- VB（Video Buffer 影像内存区块）
- VI（Video Input 影像输入）
- VI_CAP（Video Input Capture 影像输入撷取）
- VI_PROC（Video Input Process 影像输入处理）
- VPSS（Video Process Sub-System 图像处理子系统）
- VO（Video Output 影像输出）
- VDEC（Video Decoder 影像译码）
- VENC（Video Encoder 影像编码）
- AI（Audio Input 音频输入）
- AO（Audio Output 音频输出）
- ADEC（Audio Decoder 音频解码）
- AENC（Audio Encoder 音频编码）

3.2 设计概述

3.2.1 视频内存区块池

视频内存区块池主要向各模块 (VI/VPSS/VO/VDEC/VENC/LDC…) 提供大块物理内存管理功能，负责内存的取得、分配和回收，让物理内存资源在各个媒体处理模块中分享使用，并免除不必要的复制动作。

多组大小相同、物理地址连续的区块组成一个视频区块池。必须在系统初始化之前配置公共视频区块池。根据所需功能的不同，公共区块池的数量、区块的大小和数量应对应有所增减。

视频区块池所需的数量，所需考虑的点如下：

1. 每多一个 channel 就需要增加两个 (ping-pong buffer)。
2. VO 为条件一的例外，是根据 DisplayBufLen 来决定，最小为 3。
3. 若 channel 的 u32Depth 不为 0，则需要增加 u32Depth 的数量。
4. 每增加一个 LDC 的功能 (lens distortion, rotation, etc)，就需要加一个。

在内存空间足够的情况下，可以取最大的空间来建立一个公共视频区块池即可；若是要精简内存的使用量，建议使用多个不同大小的公共视频区块池。

所有的视频处理通道都可以从公共视频区块池中获取视频区块用于保存采集的图像，如图 和表中。

1. VI 先从公共视频区块池 A 中获取视频区块 Ai 用以存放从 Sensor 所接收下来的视频数据。
2. 当 VI 完成撷取，区块 Ai 经 VI 发送给 VPSS，VPSS 通道 0 跟 1 同样从公共视频区块池 A 中获取视频区块 Aj、Ak。
3. 当 VPSS 完成工作后，输入区块 Ai 被释放回公共视频区块池，Aj 作为输出图像区块 buffer 发送给 VENC，Ak 作为输出图像区块 buffer 发送给 VO。
4. Aj 经 VENC 编码完之后释放回公共视频区块池，Ak 经 VO 显示完之后释放回公共视频区块池。

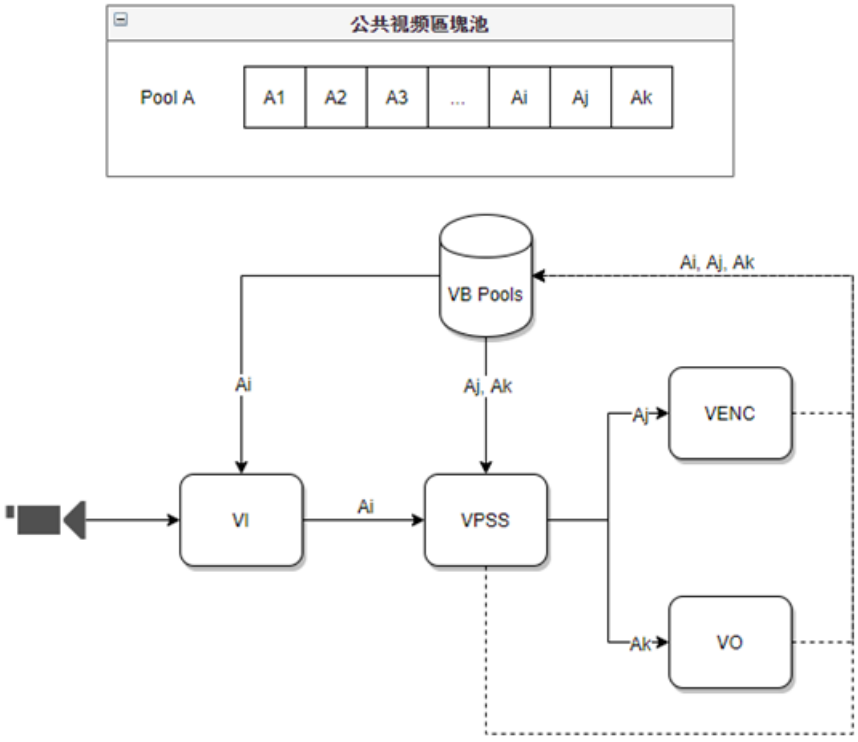


图 3.1: 公共视频区块池

表 3.1: cvi_buffer.h 中视频区块池大小计算接口简介

视频区块池大小计算接口	接口简介
COMMON_GetPicBufferConfig	一般 linear 格式的各部分数据大小
COMMON_GetPicBufferSize	一般 linear 格式的区块池

3.2.2 系统绑定

SDK 提供系统绑定接口（CVI_SYS_Bind），即通过绑定数据源和数据接收者来建立两者之间的关联关系。绑定后，数据源生成的数据将自动发送给接收者。一个数据源可以绑定多个数据接收者，若数据源未绑定，则最终会自动返回视频内存区块池。

目前支持的绑定关系如下表 所示。

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

3.2.3 VI 和 VPSS 的工作模式

VI 和 VPSS 各自的工作模式分为在线，离线模式，工作模式说明如下表 所示。

模式	VI_CAP 与 VI_PROC	VI_PROC 与 VPSS
在线模式	VI_CAP 与 VI_PROC 之间在线数据流传输，此模式下 VI_CAP 是直接把数据流送给 VI_PROC，而不会写出 RAW 数据到内存。	VI_PROC 与 VPSS 之间的在线数据流传输，此模式下 VI_PROC 是直接把数据流送给 VPSS，而不会写出 YUV 数据到内存。
离线模式	VI_CAP 写出 RAW 数据到内存，然后 VI_PROC 从内存读取 RAW 数据进行后处理。	VI_PROC 写出 YUV 数据到内存，然后 VPSS 从内存读取 YUV 数据进行后处理。

VI PIPE 可以设置成多种工作模式，情况如下：

- 第 0 个 (长曝) PIPE 可以有 4 种模式:
  - VI 在线 VPSS 离线
  - VI 在线 VPSS 在线
  - VI 离线 VPSS 离线
  - VI 离线 VPSS 在线
- 其他 PIPE，若在以下工作模式表格中，标示为”-“，表示无法独立运作输出 YUV，只能跟前一个 PIPE 在 HDR 下共同运作。

PIPE ID	0 (长曝)	1 (短曝)	2 (长曝)	3 (短曝)
模式分布 1	离线	离线	离线	离线
模式分布 2	在线	.	在线	.

3.2.4 VPSS 的工作模式

VPSS 可以有两种模式 SINGLE 或 DUAL。

处理器	模式	装置	通道数
CV181x	SINGLE	0	4
	DUAL	0 1	1 3
CV180x	SINGLE	0	3
	DUAL	0 1	1 2

- 预设为 SINGLE。
- 若非 SINGLE 时，需在 VPSS 组创建时，指定运作的 VPSS 装置。

3.2.5 Video Pipeline 的 Alignment 需求

在处理从 MEMORY 的数据时，各个处理器的模块有不同的需求。Alignment 是图像处理时每一条 line 数据的读写量，需要为此的倍数。

例如：720x480 的 YUV420 PLANAR 格式。

Y 因此为 ALIGN(720, 64) x 480 = 768 x 480 的资料量；U/V 则为 ALIGN(360, 64) x 240 = 384 x 240。

可透过 CVI_VPSS_SetChnAlign 等 API 做修改，但不可小于硬件的限制。

处理器	模块	Alignment
CV181x CV180x	VI VPSS VO	64 64 64

3.2.6 双系统消息通信

如果是双系统 SDK，需要一开始对双核通信消息初始化，建立和小核的通信后才可以调用后续的 API。双核通信消息初始化 API：CVI_MSG_Init

3.3 API 参考

MMF 系统控制实现系统初始化、系统绑定解绑、视频区块池初始化、创建视频区块池、获取 MMF 版本号、获取当前处理器的 ID 等功能。

该功能模块提供以下 API:

- CVI_MSG_Init: 双核消息通信初始化。
- CVI_MSG_Deinit: 双核消息通信去初始化。
- CVI_SYS_Init: 初始化 MMF 系统。
- CVI_SYS_Exit: 去初始化 MMF 系统。
- CVI_SYS_Bind: 数据源到数据接收者绑定。
- CVI_SYS_UnBind: 数据源到数据接收者解绑定。
- CVI_SYS_GetBindbyDest: 根据数据接收者获取绑定的数据源。
- CVI_SYS_GetBindbySrc: 根据数据源获取绑定的数据接收者。
- CVI_SYS_GetVersion: 获取当前 MMF 软件的版本号
- CVI_SYS_GetChipId: 获取当前处理器的 ID
- CVI_SYS_Mmap: 内存映射接口。
- CVI_SYS_MmapCache: 内存映射接口。
- CVI_SYS_Munmap: 内存反映射接口。
- CVI_SYS_IonAlloc: 用户分配 ION 内存。
- CVI_SYS_IonAlloc_Cached: 用户分配 ION 内存，该内存将支持 cache
- CVI_SYS_IonFlushCache: 刷新 cache 的内容到内存，并且使 cache 里的内容无效。
- CVI_SYS_IonInvalidateCache: 使 cache 里的内容无效。
- CVI_SYS_IonFree: 用户释放 ION 内存。

- CVI_SYS_SetVIVPSSMode: 设置 VI & VPSS 的工作模式
- CVI_SYS_GetVIVPSSMode: 获取 VI & VPSS 的工作模式
- CVI_SYS_SetVPSSMode: 设置 VPSS 的工作模式
- CVI_SYS_GetVPSSMode: 获取 VPSS 的工作模式
- CVI_SYS_GetModName: 获取对应 MOD_ID 的字符串句柄
- CVI_VB_SetConfig: 设置 MMF 视频区块池属性。
- CVI_VB_GetConfig: 获取 MMF 视频区块池属性。
- CVI_VB_Init: 初始化 MMF 视频区块池。
- CVI_VB_Exit: 去初始化 MMF 视频区块池。
- CVI_VB_GetBlock: 获取一个视频区块。
- CVI_VB_ReleaseBlock: 释放一个已经获取的视频区块。
- CVI_VB_PhysAddr2Handle: 通过区块的物理地址获取其句柄。
- CVI_VB_Handle2PhysAddr: 获取一个区块的物理地址。
- CVI_VB_Handle2PoolId: 获取一个区块所在区块池的 ID
- CVI_VB_InquireUserCnt: 查询区块使用计数。
- CVI_LOG_SetLevelConf: 设置日志等级。
- CVI_LOG_GetLevelConf: 获取日志等级。

3.3.1 CVI_MSG_Init

【描述】 双核消息通信初始化，仅双系统 SDK 支持。

【语法】

```
CVI_S32 CVI_MSG_Init(CVI_VOID);
```

【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：cvi_msg_client.h
- 库文件：libmsg.a

【注意】

- 在双系统中，该 API 第一个调用。
- 不能多次调用。

【举例】 无。

【相关主题】 CVI_MSG_Deinit

3.3.2 CVI_MSG_Deinit

【描述】 双核消息通信去初始化，仅双系统 SDK 支持。

【语法】

```
CVI_S32 CVI_MSG_Deinit(CVI_VOID);
```

【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：cvi_msg_client.h
- 库文件：libmsg.a

【注意】

- 在双系统中，该 API 最后一个调用。

【举例】 无。

【相关主题】 CVI_MSG_Init

3.3.3 CVI_SYS_Init

【描述】 初始化系统。包括视频输入输出、视频编解码、视频叠加区域、视频处理、音频输入输出等模块都会被初始化。

【语法】

```
CVI_S32 CVI_SYS_Init(CVI_VOID);
```

【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：cvi_sys.h, cvi_comm_sys.h
- 库文件：libsys.a

【注意】

- 由于 MMF 系统的正常运行依赖于区块池，因此需要先调用 CVI_VB_Init 初始化区块池，再初始化 MMF 系统，否则会导致功能运行异常。
- 在未调用去初始化前，如果多次初始化，仍会返回成功，但实际上系统不会对 MMF 的运行状态有任何影响。

【举例】

```
CVI_S32 s32Ret = CVI_FAILURE;

CVI_SYS_Exit();
CVI_VB_Exit();

if (pstVbConfig == NULL) {
    SAMPLE_PRT("input parameter is null, it is invaild!\n");
    return CVI_FAILURE;
}

s32Ret = CVI_VB_SetConfig(pstVbConfig);

if (s32Ret != CVI_SUCCESS) {
    SAMPLE_PRT("CVI_VB_SetConf failed!\n");
    return CVI_FAILURE;
}

s32Ret = CVI_VB_Init();

if (s32Ret != CVI_SUCCESS) {
    SAMPLE_PRT("CVI_VB_Init failed!\n");
    return CVI_FAILURE;
}

s32Ret = CVI_SYS_Init();

if (s32Ret != CVI_SUCCESS) {
    SAMPLE_PRT("CVI_SYS_Init failed!\n");
    CVI_VB_Exit();
    return CVI_FAILURE;
}
```

【相关主题】 CVI_SYS_Exit

3.3.4 CVI_SYS_Exit

【描述】 去初始化系统。包括视频输入输出、视频编解码、视频叠加区域、视频处理、音频输入输出等模块都会被销毁或禁用。

【语法】

```
CVI_S32 CVI_SYS_Exit(CVI_VOID);
```

【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：cvi_sys.h, cvi_comm_sys.h
- 库文件：libsys.a

【注意】

- 去初始化时，如果有阻塞在 MMF 上的用户进程，则去初始化会失败。如果所有阻塞在 MMF 上的调用都返回，则可以成功去初始化。

【举例】 请参考CVI_SYS_Init 的举例

【相关主题】 CVI_SYS_Init

3.3.5 CVI_SYS_Bind

【描述】 数据源到数据接收者绑定接口。

【语法】

```
CVI_S32 CVI_SYS_Bind(const MMF_CHN_S *pstSrcChn, const MMF_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：cvi_sys.h, cvi_comm_sys.h
- 库文件：libsys.a

【注意】

- 系统目前支持的绑定关系，请参见表。
- 同一个数据接收者只能绑定一个数据源。
- 绑定是指数据源和数据接收者建立关联关系。绑定后，数据源生成的数据将自动发送给接收者。
- VI 和 VDEC 作为数据源，是以通道为发送者，向其他模块发送数据，用户将设备号置为 0，SDK 不检查输入的设备号。
- VPSS 作为数据接收者时，是以设备（GROUP）为接收者，接收其他模块发来的数据，用户将通道号置为 0
- 其他情况均需指定设备号和通道号。

【举例】 无。

【相关主题】 CVI_SYS_UnBind

3.3.6 CVI_SYS_UnBind

【描述】 数据源到数据接收者解绑定接口。

【语法】

```
CVI_S32 CVI_SYS_UnBind(const MMF_CHN_S *pstSrcChn, const MMF_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：cvi_sys.h, cvi_comm_sys.h
- 库文件：libsys.a

【注意】

- pstDestChn 如果找不到绑定的源通道，则直接返回成功。如果找到了绑定的源通道但是绑定的源通道和 pstSrcChn 不匹配，则返回失败

【举例】 无。

【相关主题】 CVI_SYS_Bind

3.3.7 CVI_SYS_GetBindbyDest

【描述】 根据数据接收者获取绑定的数据源。

【语法】

```
CVI_S32 CVI_SYS_GetBindbyDest(const MMF_CHN_S *pstDestChn, MMF_
↪CHN_S *pstSrcChn);
```

【参数】

参数名称	描述	输入/输出
pstDestChn	目的通道指针	输入
pstSrcChn	源通道指针	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：cvi_sys.h, cvi_comm_sys.h
- 库文件：libsys.a

【注意】 无。

【举例】 无。

【相关主题】

- CVI_SYS_Bind
- CVI_SYS_UnBind

3.3.8 CVI_SYS_GetBindbySrc

【描述】 根据数据源获取绑定的数据接收者。

【语法】

```
CVI_S32 CVI_SYS_GetBindbySrc(const MMF_CHN_S *pstSrcChn, MMF_
↪BIND_DEST_S * pstBindDest);
```

【参数】

参数名称	描述	输入/输出
pstSrcChn	源通道指针	输入
pstBindDest	绑定的目的通道指针	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：cvi_sys.h, cvi_comm_sys.h
- 库文件：libsys.a

【注意】 无。

【举例】 无。

【相关主题】

- CVI_SYS_Bind
- CVI_SYS_UnBind

3.3.9 CVI_SYS_GetVersion

【描述】 获取当前 MMF 软件的版本号。

【语法】

```
CVI_S32 CVI_SYS_Get Version(MMF_VERSION_S *pstVersion);
```

【参数】

参数名称	描述	输入/输出
pstVersion	MMF 软件的版本号	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：cvi_sys.h, cvi_comm_sys.h
- 库文件：libsys.a

【注意】 无。

【举例】 无。

【相关主题】 无。

3.3.10 CVI_SYS_GetChipId

【描述】 获取当前处理器的 ID。

【语法】

```
CVI_S32 CVI_SYS_GetChipId(CVI_U32 *pu32ChipId);
```

【参数】

参数名称	描述	输入/输出
pu32ChipId	处理器的 ID 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：cvi_sys.h, cvi_comm_sys.h
- 库文件：libsys.a

【注意】 无。

【举例】

```
CVI_U32 u32ChipId;
CVI_SYS_GetChipId(&u32ChipId);
if (u32ChipId == CVI181x)
...
```

【相关主题】 无。

3.3.11 CVI_SYS_Mmap

【描述】 内存映射接口。

【语法】

```
void *CVI_SYS_Mmap(CVI_U64 u64PhyAddr, CVI_U32 u32Size);
```

【参数】

参数名称	描述	输入/输出
u64PhyAddr	需映像的内存单元起始地址	输入
u32Size	映射的字节数	输入

【返回值】

返回值	描述
非 0	成功
0	失败

【需求】

- 头文件：cvi_sys.h, cvi_comm_sys.h
- 库文件：libsys.a

【注意】

- 输入的地址需为合法的物理地址。
- u32Size 不能为 0

【举例】 无。

【相关主题】 CVI_SYS_Munmap

3.3.12 CVI_SYS_MmapCache

【描述】 内存映射接口。

【语法】

```
void *CVI_SYS_MmapCache(CVI_U64 u64PhyAddr, CVI_U32 u32Size);
```

【参数】

参数名称	描述	输入/输出
u64PhyAddr	需映像的内存单元起始地址	输入
u32Size	映射的字节数	输入

【返回值】

返回值	描述
非 0	成功
0	失败

【需求】

- 头文件：cvi_sys.h, cvi_comm_sys.h
- 库文件：libsys.a

【注意】

- 输入的地址需为合法的物理地址。
- u32Size 不能为 0
- 透过此 API 所取得的虚拟地址，若有跟 HW DMA 共享。在处理器存取前，需调用 invalidate；而在处理器修改后，若要让 HW DMA 读取，则需做 invalidate。
- 此 API 仅可用于映射已分配的 Cached 内存地址。例如，若某 vb pool 的 VB_POOL_CONFIG_S 中 enRemapMode 属性为 VB_REMAP_MODE_NONE，则即使透过本 API 取得虚拟地址，在读写改内存块时，也无法获得 Cached 内存带来的性能提升。

【举例】 无。

【相关主题】

- CVI_SYS_Munmap
- CVI_SYS_IonFlushCache
- CVI_SYS_IonInvalidateCache

3.3.13 CVI_SYS_Munmap

【描述】 内存反映射接口。

【语法】

```
CVI_S32 CVI_SYS_ Munmap(CVI_VOID *pVirAddr, CVI_U32 u32Size);
```

【参数】

参数名称	描述	输入/输出
pVirAddr	mmap 后返回的地址	输入
u32Size	映射的字节数	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：cvi_sys.h, cvi_comm_sys.h
- 库文件：libsys.a

【注意】 无。

【举例】 无。

【相关主题】 CVI_SYS_Mmap

3.3.14 CVI_SYS_IonAlloc

【描述】 用户分配 ION 内存。

【语法】

```
CVI_S32 CVI_SYS_IonAlloc(CVI_U64 *pu64PhyAddr, CVI_VOID **ppVirAddr,␣
↪CVI_U32 u32Len);
```

【参数】

参数名称	描述	输入/输出
pu64PhyAddr	物理地址指针	输出
ppVirAddr	虚拟地址指针的指针。若为 NULL，则不会做映射	输出
u32Len	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：cvi_sys.h, cvi_comm_sys.h
- 库文件：libsys.a

【注意】

- 透过此 API 所分配的内存，需通过 CVI_SYS_Mmap 获取其虚拟地址。

【举例】 无。

【相关主题】 CVI_SYS_IonFree

3.3.15 CVI_SYS_IonAlloc_Cached

【描述】 用户分配 ION 内存，该内存将支持 cache。

【语法】

```
CVI_S32 CVI_SYS_IonAlloc_Cached(CVI_U64 *pu64PhyAddr, CVI_VOID
↪**ppVirAddr, CVI_U32 u32Len);
```

【参数】

参数名称	描述	输入/输出
pu64PhyAddr	物理地址指针。	输出
ppVirAddr	虚拟地址指针的指针。若为 NULL，则不会做映射。	输出
u32Len	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：cvi_sys.h, cvi_comm_sys.h
- 库文件：libsys.a

【注意】

- 透过此 API 所取得的虚拟地址，若有跟 HW DMA 共享。在处理器存取前，需调用 invalidate；而在处理器修改后，若要让 HW DMA 读取，则需做 invalidate。
- 透过此 API 所分配的内存，需通过 CVI_SYS_MmapCache 获取其虚拟地址。

【举例】 无。

【相关主题】

- CVI_SYS_IonFlushCache
- CVI_SYS_IonInvalidateCache
- CVI_SYS_IonFree

3.3.16 CVI_SYS_IonFlushCache

【描述】 刷新 cache 的内容到内存，并且使 cache 里的内容无效。

【语法】

```
CVI_S32 CVI_SYS_IonFlushCache(CVI_U64 u64PhyAddr, CVI_VOID *pVirAddr,
↪ CVI_U32 u32Len);
```

【参数】

参数名称	描述	输入/输出
pu64PhyAddr	物理地址指针。	输入
pVirAddr	虚拟地址指针。	输入
u32Len	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：cvi_sys.h, cvi_comm_sys.h
- 库文件：libsys.a

【注意】

- 此接口应与CVI_SYS_IonAlloc_Cached 接口搭配使用。

【举例】 无。

【相关主题】

- CVI_SYS_IonAlloc_Cached
- CVI_SYS_IonInvalidateCache

3.3.17 CVI_SYS_IonInvalidateCache

【描述】 使 cache 里的内容无效。

【语法】

```
CVI_S32 CVI_SYS_IonInvalidateCache(CVI_U64 u64PhyAddr, CVI_VOID
↪*pVirAddr, CVI_U32 u32Len);
```

【参数】

参数名称	描述	输入/输出
pu64PhyAddr	物理地址指针。	输入
pVirAddr	虚拟地址指针。	输入
u32Len	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：cvi_sys.h, cvi_comm_sys.h
- 库文件：libsys.a

【注意】

- 此接口应与CVI_SYS_IonAlloc_Cached 接口搭配使用。

【举例】 无。

【相关主题】

- CVI_SYS_IonAlloc_Cached
- CVI_SYS_IonFlushCache

3.3.18 CVI_SYS_IonFree

【描述】 用户释放 ION 内存。

【语法】

```
CVI_S32 CVI_SYS_IonFree(CVI_U64 u64PhyAddr, CVI_VOID *pVirAddr);
```

【参数】

参数名称	描述	输入/输出
u64PhyAddr	物理地址指针。	输入
pVirAddr	虚拟地址指针。	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：cvi_sys.h, cvi_comm_sys.h
- 库文件：libsys.a

【注意】 无。

【举例】 无。

【相关主题】

- CVI_SYS_IonAlloc
- CVI_SYS_IonAlloc_Cached

3.3.19 CVI_SYS_SetVIVPSSMode

【描述】 设置 VI & VPSS 工作模式。

【语法】

```
CVI_S32 CVI_SYS_SetVIVPSSMode(const VI_VPSS_MODE_S
↪*pstVIVPSSMode);
```

【参数】

参数名称	描述	输入/输出
pstVIVPSSMode	VI，VPSS 的工作模式	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：cvi_sys.h, cvi_comm_sys.h

- 库文件：libsys.a

【注意】

- 必须在CVI_SYS_Init 后，所有的 VI PIPE 和所有的 VPSS 组创建前设置。

【举例】 无。

【相关主题】 CVI_SYS_GetVIVPSSMode

3.3.20 CVI_SYS_GetVIVPSSMode

【描述】 获取 VI & VPSS 的工作模式。

【语法】

```
CVI_S32 CVI_SYS_GetVIVPSSMode(VI_VPSS_MODE_S *pstVIVPSSMode);
```

【参数】

参数名称	描述	输入/输出
pstVIVPSSMode	VI，VPSS 的工作模式	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：cvi_sys.h, cvi_comm_sys.h
- 库文件：libsys.a

【注意】 无。

【举例】 无。

【相关主题】 CVI_SYS_SetVIVPSSMode

3.3.21 CVI_SYS_SetVPSSMode

【描述】 设置 VPSS 工作模式。

【语法】

```
CVI_S32 CVI_SYS_SetVPSSMode(VPSS_MODE_E enVPSSMode);
```

【参数】

参数名称	描述	输入/输出
enVPSSMode	VPSS 的工作模式	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：cvi_sys.h, cvi_comm_sys.h
- 库文件：libsys.a

【注意】

- 必须在 CVI_SYS_Init 后，所有的 VPSS 组创建前设置。
- 默认值为 VPSS_MODE_SINGLE。

【举例】 无。

【相关主题】 CVI_SYS_GetVPSSMode

3.3.22 CVI_SYS_GetVPSSMode

【描述】 获取 VPSS 的工作模式。

【语法】

```
VPSS_MODE_E CVI_SYS_GetVPSSMode(CVI_VOID);
```

【参数】 无。

【返回值】

返回值	描述
VPSS_MODE_E	当下 VPSS 的工作模式。

【需求】

- 头文件：cvi_sys.h, cvi_comm_sys.h
- 库文件：libsys.a

【注意】 无。

【举例】 无。

【相关主题】 CVI_SYS_SetVPSSMode

3.3.23 CVI_SYS_GetModName

【描述】 获取对应 MOD_ID 的字符串句柄。

【语法】

```
const CVI_CHAR *CVI_SYS_GetModName(MOD_ID_E id);
```

【参数】

参数名称	描述	输入/输出
id	MOD_ID_E	输入

【返回值】

返回值	描述
const CVI_CHAR *	MOD_ID 的字符串句柄
NULL	失败

【需求】

- 头文件：cvi_sys.h, cvi_comm_sys.h
- 库文件：libsys.a

【注意】 无。

【举例】 无。

【相关主题】 无。

3.3.24 CVI_VB_SetConfig

【描述】 设置 MMF 视频区块池属性。

【语法】

```
CVI_S32 CVI_VB_SetConfig(const VB_CONFIG_S *pstVbConfig);
```

【参数】

参数名称	描述	输入/输出
pstVbConfig	视频区块池属性指针	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：cvi_vb.h, cvi_comm_vb.h
- 库文件：libsys.a

【注意】

- 只能在系统处于未初始化的状态下，才可以设置区块池属性，否则会返回失败。
- video buffer 根据不同的应用场景需要不同的配置。配置规则参见视频内存区块池 。
- 公共区块池中每个区块的大小应根据当前图像像素格式以及图像是否压缩而有所不同。具体分配大小请参考VB_CONFIG_S 结构体中的描述。

【举例】 请参考CVI_SYS_Init 的举例

【相关主题】 CVI_VB_GetConfig

3.3.25 CVI_VB_GetConfig

【描述】 获取 MMF 视频区块池属性。

【语法】

```
CVI_S32 CVI_VB_GetConfig(VB_CONFIG_S *pstVbConfig);
```

【参数】

参数名称	描述	输入/输出
pstVbConfig	视频区块池属性指针	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：cvi_vb.h, cvi_comm_vb.h
- 库文件：libsys.a

【注意】

- 必须先调用CVI_VB_SetConfig 设置 MMF 视频区块池属性。

【举例】 无。

【相关主题】 CVI_VB_SetConfig

3.3.26 CVI_VB_Init

【描述】 初始化 MMF 视频区块池。

【语法】

```
CVI_S32 CVI_VB_Init(CVI_VOID);
```

【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：cvi_vb.h, cvi_comm_vb.h
- 库文件：libsys.a

【注意】

- 必须先调用 CVI_VB_SetConfig 配置区块池属性，再初始化，否则初始化会失败。
- 如果多次初始化，仍会返回成功，但实际上系统不会对 MMF 的运行状态有任何影响。

【举例】 请参考CVI_SYS_Init 的举例

【相关主题】 CVI_VB_Exit

3.3.27 CVI_VB_Exit

【描述】 去初始化 MMF 视频区块池。

【语法】

```
CVI_S32 CVI_VB_Exit(CVI_VOID);
```

【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：cvi_vb.h, cvi_comm_vb.h

- 库文件：libsys.a

【注意】

- 必须先调用CVI_SYS_Exit 去初始化 MMF 系统，再去初始化区块池，否则返回失败。
- 可以反复去初始化，不返回失败

【举例】 请参考CVI_SYS_Init 的举例

【相关主题】 CVI_VB_Init

3.3.28 CVI_VB_GetBlock

【描述】 获取一个视频区块。

【语法】

```
VB_BLK CVI_VB_GetBlock(VB_POOL Pool, CVI_U32 u32BlkSize);
```

【参数】

参数名称	描述	输入/输出
Pool	区块池 ID 号。取值范围：[0, VB_MAX_POOLS)。	输入
u32BlkSize	区块大小。取值范围：数据类型全范围，以 byte 为单位。	输入

【返回值】

返回值	描述
非 VB_INVALID_HANDLE	有效的区块句柄。
VB_INVALID_HANDLE	获取失败。

【需求】

- 头文件：cvi_vb.h, cvi_comm_vb.h
- 库文件：libsys.a

【注意】

- 用户在创建区块池之后，可以调用本接口从该区块池中来获取区块；第 2 个参数 u32BlkSize 须小于或等于创建该区块池时指定的区块大小。
- 第 1 个参数 Pool 设置为无效 ID 号 VB_INVALID_POOLID 时，会从公共区块池中获取一块最符合指定大小的区块；反之，则会从指定的 Pool 去获取一块指定大小的区块。如果指定的大小无法符合，那么将获取不到区块。
- 公共区块池主要用来存放 VPU（VI/VPSS/VO/GDC）的捕获图像。因为是多个模块共享的，因此，当对公共区块池的不当操作（如占用过多的区块）会影响整个 MMF 系统的正常运行。

【举例】 无。

【相关主题】 CVI_VB_ReleaseBlock

3.3.29 CVI_VB_ReleaseBlock

【描述】 释放一个已经获取的视频区块。

【语法】

```
CVI_S32 CVI_VB_ReleaseBlock(VB_BLK Block);
```

【参数】

参数名称	描述	输入/输出
Block	区块句柄	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：cvi_vb.h, cvi_comm_vb.h
- 库文件：libsys.a

【注意】

- 获取的区块使用后，应该调用此接口释放区块。

【举例】 无。

【相关主题】 CVI_VB_GetBlock

3.3.30 CVI_VB_CreatePool

【描述】 获取一个视频区块。

【语法】

```
VB_POOL CVI_VB_CreatePool(VB_POOL_CONFIG_S *pstVbPoolCfg);
```

【参数】

参数名称	描述	输入/输出
pstVbPoolCfg	区块池配置属性参数指针。	输入

【返回值】

返回值	描述
非 VB_INVALID_HANDLE	有效的区块池 ID 号。
VB_INVALID_HANDLE	创建区块池失败，可能是参数非法或者保留内存不够。

【需求】

- 头文件：cvi_vb.h, cvi_comm_vb.h
- 库文件：libsys.a

【注意】 无。

【举例】 无。

【相关主题】 CVI_VB_DestroyPool

3.3.31 CVI_VB_DestroyPool

【描述】 释放一个已经获取的视频区块。

【语法】

```
CVI_S32 CVI_VB_DestroyPool(VB_POOL Pool);
```

【参数】

参数名称	描述	输入/输出
Pool	区块池 ID 号。取值范围：[0, VB_MAX_POOLS)。	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：cvi_vb.h, cvi_comm_vb.h
- 库文件：libsys.a

【注意】

- 销毁一个不存在的区块池，则返回 CVI_ERR_VB_ILLEGAL_PARAM。
- 在去初始化 MMF 区块池时，所有的区块池都将被销毁，包括用户态的区块池。

- 退出 VB 池之前请确保 VB 池里的任何 VB 都没有被占用，否则无法退出。
- [0, VB_MAX_POOLS) 范围内的区块池 ID 号，包括公共区块池、模块公共区块池、模块私有区块池等的 ID 号。请确保 Pool 为CVI_VB_CreatePool 所创建的区块池的 ID 号，否则会返回失败。
- 如果当前区块池有通过CVI_VB_MmapPool 接口映像虚拟地址，则必须先通过CVI_VB_MunmapPool 接口解除映射，然后才能销毁区块池。

【举例】 无。

【相关主题】 CVI_VB_CreatePool

3.3.32 CVI_VB_PhysAddr2Handle

【描述】 由一个已经获取的区块物理地址获取句柄。

【语法】

```
VB_BLK CVI_VB_PhysAddr2Handle(CVI_U64 u64PhyAddr);
```

【参数】

参数名称	描述	输入/输出
u64PhyAddr	区块物理地址	输入

【返回值】

返回值	描述
非 VB_INVALID_HANDLE	有效的区块句柄。
VB_INVALID_HANDLE	获取失败。

【需求】

- 头文件：cvi_vb.h, cvi_comm_vb.h
- 库文件：libsys.a

【注意】

- 物理地址应该是从 MMF 视频区块池中获取的有效区块的地址。

【举例】 无。

【相关主题】 无。

3.3.33 CVI_VB_Handle2PhysAddr

【描述】 由一个已经获取的区块句柄获取物理地址。

【语法】

```
CVI_U64 CVI_VB_Handle2PhysAddr(VB_BLK Block);
```

【参数】

参数名称	描述	输入/输出
Block	区块句柄	输入

【返回值】

返回值	描述
非 0	有效的物理地址。
0	无效返回值，区块句柄非法。

【需求】

- 头文件：cvi_vb.h, cvi_comm_vb.h
- 库文件：libsys.a

【注意】

- 区块应该是从 MMF 视频区块池中获取的有效区块。

【举例】 无。

【相关主题】 无。

3.3.34 CVI_VB_Handle2PoolId

【描述】 由一个已经获取的区块句柄获取视频区块池 ID。

【语法】

```
VB_POOL CVI_VB_Handle2PoolId(VB_BLK Block);
```

【参数】

参数名称	描述	输入/输出
Block	区块句柄	输入

【返回值】

返回值	描述
非 VB_INVALID_POOLID	有效的视频区块池 ID。
VB_INVALID_POOLID	无效返回值，区块句柄非法。

【需求】

- 头文件：cvi_vb.h, cvi_comm_vb.h
- 库文件：libsys.a

【注意】

- 区块应该是从 MMF 视频区块池中获取的有效区块。

【举例】 无。

【相关主题】 无。

3.3.35 CVI_VB_InquireUserCnt

【描述】 由一个已经获取的区块句柄获取存块使用计数。

【语法】

```
CVI_S32 CVI_VB_InquireUserCnt(VB_BLK Block, CVI_U32 *pCnt);
```

【参数】

参数名称	描述	输入/输出
Block	区块句柄	输入
pCnt	区块使用计数	输出

【返回值】

返回值	描述
非负数	使用计数。
负数	无效返回值，区块句柄非法。

【需求】

- 头文件：cvi_vb.h, cvi_comm_vb.h
- 库文件：libsys.a

【注意】 无。

【举例】 无。

【相关主题】 无。

3.3.36 CVI_VB_MmapPool

【描述】 为一个视频区块池映射用户态虚拟地址。

【语法】

```
CVI_S32 CVI_VB_MmapPool(VB_POOL Pool);
```

【参数】

参数名称	描述	输入/输出
Pool	缓存池 ID 号。取值范围：[0, VB_MAX_POOLS)。	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：cvi_vb.h, cvi_comm_vb.h
- 库文件：libsys.a

【注意】

- 必须输入合法的视频区块池 ID。
- 重复映射视为成功。

【举例】 无。

【相关主题】 无。

3.3.37 CVI_VB_MunmapPool

【描述】 为一个视频区块池解除用户态映射。

【语法】

```
CVI_S32 CVI_VB_MunmapPool(VB_POOL Pool);
```

【参数】

参数名称	描述	输入/输出
Pool	区块池 ID 号。取值范围：[0, VB_MAX_POOLS)。	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：cvi_vb.h, cvi_comm_vb.h
- 库文件：libsys.a

【注意】

- 必须输入合法的视频区块池 ID。
- 视频区块池必须已经映像过，如果未映像，则直接返回 CVI_ERR_VB_NOTREADY。
- 必须先释放虚拟地址，然后再销毁区块池。

【举例】 无。

【相关主题】 无。

3.3.38 CVI_VB_GetBlockVirAddr

【描述】 获取一个视频区块池中的区块的用户态虚拟地址。

【语法】

```
CVI_S32 CVI_VB_GetBlockVirAddr(VB_POOL Pool, VB_BLK Block, void␣
↪**ppVirAddr);
```

【参数】

参数名称	描述	输入/输出
Pool	区块池 ID 号。取值范围：[0, VB_MAX_POOLS)。	输入
Block	区块句柄	输入
ppVirAddr	用户态虚拟地址。	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：cvi_vb.h, cvi_comm_vb.h

- 库文件：libsys.a

【注意】

- 必须输入合法的视频区块池 ID、合法的区块句柄。
- 视频区块池必须已经映像过，如果未映像，则直接返回 CVI_ERR_VB_NOTREADY。
- 如果物理地址不在当前 VB 池范围内，则返回 CVI_ERR_VB_ILLEGAL_PARAM。

【举例】 无。

【相关主题】 无。

3.3.39 CVI_LOG_SetLevelConf

【描述】 设置日志等级。

【语法】

```
CVI_S32 CVI_LOG_SetLevelConf(LOG_LEVEL_CONF_S *pstConf);
```

【参数】

参数名称	描述	输入/输出
pstConf	日志等级信息结构体	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：cvi_sys.h, cvi_comm_sys.h, cvi_debug.h
- 库文件：libsys.a

【注意】 无。

【举例】 无。

【相关主题】 CVI_LOG_SetLevelConf

3.3.40 CVI_LOG_GetLevelConf

【描述】 获取日志等级。

【语法】

```
CVI_S32 CVI_LOG_GetLevelConf(LOG_LEVEL_CONF_S *pstConf);
```

【参数】

参数名称	描述	输入/输出
pstConf->enModId	需要获取日志等级的模块 ID	输入
pstConf->s32Level	获取到日志等级	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：cvi_sys.h, cvi_comm_sys.h, cvi_debug
- 库文件：libsys.a

【注意】 无。

【举例】 无。

【相关主题】 CVI_LOG_SetLevelConf

3.4 数据类型

3.4.1 基础型别

```
/*----------------------------------------------
* The common data type
*----------------------------------------------
*/

typedef unsigned char          CVI_UCHAR;
typedef unsigned char          CVI_U8;
typedef unsigned short         CVI_U16;
typedef unsigned int           CVI_U32;
typedef unsigned int           CVI_HANDLE;

typedef signed char            CVI_S8;
```

(下页继续)

(续上页)

```
typedef signed char             CVI_CHAR;
typedef short                   CVI_S16;
typedef int                     CVI_S32;

typedef unsigned long           CVI_UL;
typedef signed long             CVI_SL;

typedef float                   CVI_FLOAT;
typedef double                  CVI_DOUBLE;

typedef void                    CVI_VOID;
typedef bool                    CVI_BOOL;

typedef uint64_t                CVI_U64;
typedef int64_t                 CVI_S64;

typedef size_t                CVI_SIZE_T;
/*-----------------------------------------
* const defination
*-----------------------------------------
*/

#define CVI_NULL              0L
#define CVI_SUCCESS           0
#define CVI_FAILURE           (-1)
#define CVI_FAILURE_ILLEGAL_PARAM (-2)
#define CVI_TRUE              1
#define CVI_FALSE             0
```

3.4.2 MOD_ID_E

【说明】 定义模块 ID 列举

【定义】

```
#define FOREACH_MOD(MOD) {\
    MOD(CMPI)   \
    MOD(VB)     \
    MOD(SYS)    \
    MOD(RGN)    \
    MOD(CHNL)   \
    MOD(VDEC)   \
    MOD(VPSS)   \
    MOD(VENC)   \
    MOD(H264E)  \
    MOD(JPEGE)  \
    MOD(MPEG4E) \
    MOD(H265E)  \
    MOD(JPEGD)  \
    MOD(VO)     \
    MOD(VI)     \
```

(下页继续)

(续上页)

```
    MOD(DIS)    \
    MOD(RC)     \
    MOD(AIO)    \
    MOD(AI)     \
    MOD(AO)     \
    MOD(AENC)   \
    MOD(ADEC)   \
    MOD(AUD)    \
    MOD(VPU)    \
    MOD(ISP)    \
    MOD(IVE)    \
    MOD(USER)   \
    MOD(PROC)   \
    MOD(LOG)    \
    MOD(H264D)  \
    MOD(GDC)    \
    MOD(PHOTO)  \
    MOD(FB)     \
    MOD(BUTT)   \
}

#define GENERATE_ENUM(ENUM) CVI_ID_ ## ENUM,

typedef enum _MOD_ID_E FOREACH_MOD(GENERATE_ENUM) MOD_ID_
↪E;
```

【注意事项】 无。

【相关数据类型及接口】 无。

3.4.3 VB_SOURCE_E

【说明】 定义 VB 来源选择

【定义】

```
typedef enum _VB_SOURCE_E {
    VB_SOURCE_COMMON = 0,
    VB_SOURCE_MODULE = 1,
    VB_SOURCE_PRIVATE = 2,
    VB_SOURCE_USER = 3,
    VB_SOURCE_BUTT
} VB_SOURCE_E;
```

【成员】

成员名称	描述
VB_SOURCE_COMMON	公共 VB
VB_SOURCE_MODULE	模块 VB
VB_SOURCE_PRIVATE	私有 VB
VB_SOURCE_USER	用户 VB

【注意事项】 无。

【相关数据类型及接口】 无。

3.4.4 ROTATION_E

【说明】 定义旋转角度列举

【定义】

```
typedef enum _ROTATION_E {
    ROTATION_0 = 0,
    ROTATION_90,
    ROTATION_180,
    ROTATION_270,
    ROTATION_MAX
} ROTATION_E;
```

【成员】

成员名称	描述
ROTATION_0	不旋转。
ROTATION_90	旋转 90 度。
ROTATION_180	旋转 180 度。
ROTATION_270	旋转 270 度。

【注意事项】 无。

【相关数据类型及接口】 无。

3.4.5 POINT_S

【说明】 定义坐标结构体

【定义】

```
typedef struct _POINT_S {
    CVI_S32 s32X;
    CVI_S32 s32Y;
} POINT_S;
```

【成员】

成员名称	描述
s32X	横坐标。
s32Y	纵坐标。

【注意事项】 无。

【相关数据类型及接口】 无。

3.4.6 SIZE_S

【说明】 定义大小结构体

【定义】

```
typedef struct _SIZE_S {
    CVI_U32 u32Width;
    CVI_U32 u32Height;
} SIZE_S;
```

【成员】

成员名称	描述
u32Width	宽度。
u32Height	高度。

【注意事项】 无。

【相关数据类型及接口】

- VI_DEV_ATTR_S
- VI_PIPE_STATUS_S
- VI_CHN_ATTR_S
- VI_CHN_STATUS_S
- VO_VIDEO_LAYER_ATTR_S

3.4.7 RECT_S

【说明】 定义矩形的宽度、高度和位置结构体

【定义】

```
typedef struct _RECT_S {
    CVI_S32 s32X;
    CVI_S32 s32Y;
    CVI_U32 u32Width;
    CVI_U32 u32Height;
} RECT_S;
```

【成员】

成员名称	描述
s32X	横坐标。
s32Y	纵坐标。
u32Width	宽度。
u32Height	高度。

【注意事项】 无。

【相关数据类型及接口】

- ASPECT_RATIO_S
- VI_CROP_INFO_S
- VPSS_CROP_INFO_S
- VO_CHN_ATTR_S
- VO_VIDEO_LAYER_ATTR_S

3.4.8 LDC_ATTR_S

【说明】 定义镜头畸变矫正结构体

【定义】

```
typedef struct _LDC_ATTR_S {
    bool bAspect; /* RW;Whether aspect ration  is keep */
    CVI_S32 s32XRatio; /* RW; Range: [0, 100], field angle ration of  horizontal,
↪valid when bAspect=0.*/
    CVI_S32 s32YRatio; /* RW; Range: [0, 100], field angle ration of  vertical,valid␣
↪when bAspect=0.*/
    CVI_S32 s32XYRatio; /* RW; Range: [0, 100], field angle ration of  all,valid␣
↪when bAspect=1.*/
    CVI_S32 s32CenterXOffset;
    CVI_S32 s32CenterYOffset;
    CVI_S32 s32DistortionRatio;
} LDC_ATTR_S;
```

【成员】

成员名称	描述
bAspect	畸变矫正是否水平垂直同样比例。
s32XRatio	水平视场角大小, bAspect=0 有效。[0, 100]
s32YRatio	垂直视场角大小, bAspect=0 有效。[0, 100]
s32XYRatio	整体视场角大小, bAspect=1 有效。[0, 100]
s32CenterXOffset	畸变中心点相对图像中心的水平偏移。[-511, 511]
s32CenterYOffset	畸变中心点相对图像中心的垂直偏移。[-511, 511]
s32DistortionRatio	畸变程度。[-300, 500]

【注意事项】 无。

【相关数据类型及接口】 无。

3.4.9 MMF_CHN_S

【说明】 定义模块信道结构体

【定义】

```
typedef struct _MMF_CHN_S {
    MOD_ID_E    enModId;
    CVI_S32     s32DevId;
    CVI_S32     s32ChnId;
} MMF_CHN_S;
```

【成员】

成员名称	描述
enModId	模块 ID。
s32DevId	设备 ID。在部分模块也可能是群 ID
s32ChnId	通道 ID。

【注意事项】 无。

【相关数据类型及接口】

- CVI_SYS_Bind
- CVI_SYS_UnBind
- CVI_SYS_GetBindByDest
- CVI_SYS_GetBindBySrc

3.4.10 MMF_BIND_DEST_S

【说明】 定义 MMF 系统绑定结构体

【定义】

```
typedef struct _MMF_BIND_DEST_S {
    CVI_U32   u32Num;
    MMF_CHN_S astMmfChn[BIND_DEST_MAXNUM];
} MMF_BIND_DEST_S;
```

【成员】

成员名称	描述
u32Num	绑定目的数量。
astMmfChn	绑定目的的信道结构体数组

【注意事项】 无。

【相关数据类型及接口】

- CVI_SYS_GetBindBySrc

3.4.11 MMF_VERSION_S

【说明】 定义 MMF 版本结构体

【定义】

```
#define VERSION_NAME_MAXLEN 128
typedef struct _MMF_VERSION_S {
    char version[VERSION_NAME_MAXLEN];
} MMF_VERSION_S;
```

【成员】

成员名称	描述
version	版本描述字符串。

【注意事项】 无。

【相关数据类型及接口】

- CVI_SYS_GetVersion

3.4.12 VB_CONFIG_S

【说明】 定义 MMF 系统视频区块池结构体

【定义】

```
typedef struct _VB_CONFIG_S {
    CVI_U32 u32MaxPoolCnt;
    VB_POOL_CONFIG_S astCommPool[VB_MAX_COMM_POOLS];
} VB_CONFIG_S;
```

【成员】

成员名称	描述
u32MaxPoolCnt	公共视频区块池数。
astCommPool	公共视频区块池属性。

【注意事项】 无。

【相关数据类型及接口】

- CVI_VB_SetConfig
- CVI_VB_GetConfig

3.4.13 VB_POOL_CONFIG_S

【说明】 定义 MMF 视频区块池结构体

【定义】

```
typedef struct _VB_POOL_CONFIG_S {
    CVI_U32 u32BlkSize;
    CVI_U32 u32BlkCnt;
    VB_REMAP_MODE_E enRemapMode;
    CVI_CHAR acName[MAX_VB_POOL_NAME_LEN];
} VB_POOL_CONFIG_S;
```

【成员】

成员名称	描述
u32BlkSize	视频区块大小。
u32BlkCnt	视频区块池内的区块数。
enRemapMode	区块的 memory-map 模式
acName	视频区块名称

【注意事项】

- u32BlkSize 应根据所需图像大小、格式等信息来计算。若太大，会造成无谓的内存空间浪费；若太小，各 MOD 无法取得视频区块来使用。

· 视频区块池是透过空闲内存来取得，若视频区块池大小超过内存的空闲大小，则会创建失败。

【相关数据类型及接口】

· VB_CONFIG_S

3.4.14 VI_VPSS_MODE_E

【说明】 定义 VI 和 VPSS 间的工作模式列举。

【定义】

```
typedef enum _VI_VPSS_MODE_E {
    VI_OFFLINE_VPSS_OFFLINE = 0,
    VI_OFFLINE_VPSS_ONLINE,
    VI_ONLINE_VPSS_OFFLINE,
    VI_ONLINE_VPSS_ONLINE,
    VI_VPSS_MODE_BUTT
} VI_VPSS_MODE_E;
```

【成员】

成员名称	描述
VI_OFFLINE_VPSS_OFFLINE	VI_PROC/VI_CAP 离线，VI_CAP/VPSS 离线。
VI_OFFLINE_VPSS_ONLINE	VI_PROC/VI_CAP 离线，VI_CAP/VPSS 在线。
VI_ONLINE_VPSS_OFFLINE	VI_PROC/VI_CAP 在线，VI_CAP/VPSS 离线。
VI_ONLINE_VPSS_ONLINE	VI_PROC/VI_CAP 在线，VI_CAP/VPSS 在线。

【注意事项】

· 当 VPSS_ONLINE 时，VPSS 无法分时处理，只能绑定特定的 VI PIPE 工作。

【相关数据类型及接口】 无。

3.4.15 VPSS_MODE_E

【说明】 定义 VPSS 的工作模式列举。

【定义】

```
typedef enum _VPSS_MODE_E {
    VPSS_MODE_SINGLE = 0,
    VPSS_MODE_DUAL,
    VPSS_MODE_RGNEX,
    VPSS_MODE_BUTT
} VPSS_MODE_E;
```

【成员】

成员名称	描述
VPSS_MODE_SINGLE	VPSS 以单一硬件实体的方式工作。
VPSS_MODE_DUAL	VPSS 以两个硬件实体的方式工作。
VPSS_MODE_RGNEX	VPSS 以两个硬件实体的方式工作。会用 VPSS 装置 0 做 RGN 迭图，因此只有 VPSS 装置 1 可以走在线模式。

【注意事项】

- CV181x/CV180x 不支持 VPSS_MODE_RGNEX 模式

【相关数据类型及接口】 无。

3.4.16 ASPECT_RATIO_E

【说明】 定义旋转角度列举

【定义】

```
typedef enum _ASPECT_RATIO_E {
    ASPECT_RATIO_NONE = 0,
    ASPECT_RATIO_AUTO,
    ASPECT_RATIO_MANUAL,
    ASPECT_RATIO_MAX
} ASPECT_RATIO_E;
```

【成员】

成员名称	描述
ASPECT_RATIO_NONE	不动作，满屏。
ASPECT_RATIO_AUTO	视频保持比例，自动计算视频区域。
ASPECT_RATIO_MANUAL	手动决定视频区域。

【注意事项】 无。

【相关数据类型及接口】 无。

3.4.17 ASPECT_RATIO_S

【说明】 定义画面比例结构体

【定义】

```
typedef struct _ASPECT_RATIO_S {
    ASPECT_RATIO_E enMode;
    CVI_BOOL bEnableBgColor;
    CVI_U32 u32BgColor;
    RECT_S stVideoRect;
} ASPECT_RATIO_S;
```

【成员】

成员名称	描述
enMode	画面比例列举。
bEnableBgColor	是否要让画面以外以背景色覆盖。
u32BgColor	画面比例的背景色。为 RGB888 的格式，bit[7:0] 为 B，bit[15:8] 为 G，bit[23:16] 为 R。
stVideoRect	画面比例时视频的区域，只有当 enMode 为 ASPECT_RATIO_MANUAL 时才有用

【注意事项】 无。

【相关数据类型及接口】

- ASPECT_RATIO_E

3.4.18 PIXEL_FORMAT_E

【说明】 定义像素格式枚举

【定义】

```
typedef enum _PIXEL_FORMAT_E {
    PIXEL_FORMAT_RGB_888 = 0,
    PIXEL_FORMAT_BGR_888,
    PIXEL_FORMAT_RGB_888_PLANAR,
    PIXEL_FORMAT_BGR_888_PLANAR,

    PIXEL_FORMAT_ARGB_1555, // 4,
    PIXEL_FORMAT_ARGB_4444,
    PIXEL_FORMAT_ARGB_8888,

    PIXEL_FORMAT_RGB_BAYER_8BPP, // 7,
    PIXEL_FORMAT_RGB_BAYER_10BPP,
```

(下页继续)

(续上页)

```
    PIXEL_FORMAT_RGB_BAYER_12BPP,
    PIXEL_FORMAT_RGB_BAYER_14BPP,
    PIXEL_FORMAT_RGB_BAYER_16BPP,

    PIXEL_FORMAT_YUV_PLANAR_422, // 12,
    PIXEL_FORMAT_YUV_PLANAR_420,
    PIXEL_FORMAT_YUV_PLANAR_444,
    PIXEL_FORMAT_YUV_400,

    PIXEL_FORMAT_HSV_888, // 16,
    PIXEL_FORMAT_HSV_888_PLANAR,

    PIXEL_FORMAT_NV12, // 18,
    PIXEL_FORMAT_NV21,
    PIXEL_FORMAT_NV16,
    PIXEL_FORMAT_NV61,
    PIXEL_FORMAT_YUYV,
    PIXEL_FORMAT_UYVY,
    PIXEL_FORMAT_YVYU,
    PIXEL_FORMAT_VYUY,

    PIXEL_FORMAT_FP32_C1 = 32, // 32
    PIXEL_FORMAT_FP32_C3_PLANAR,
    PIXEL_FORMAT_INT32_C1,
    PIXEL_FORMAT_INT32_C3_PLANAR,
    PIXEL_FORMAT_UINT32_C1,
    PIXEL_FORMAT_UINT32_C3_PLANAR,
    PIXEL_FORMAT_BF16_C1,
    PIXEL_FORMAT_BF16_C3_PLANAR,
    PIXEL_FORMAT_INT16_C1,
    PIXEL_FORMAT_INT16_C3_PLANAR,
    PIXEL_FORMAT_UINT16_C1,
    PIXEL_FORMAT_UINT16_C3_PLANAR,
    PIXEL_FORMAT_INT8_C1,
    PIXEL_FORMAT_INT8_C3_PLANAR,
    PIXEL_FORMAT_UINT8_C1,
    PIXEL_FORMAT_UINT8_C3_PLANAR,

    PIXEL_FORMAT_8BIT_MODE = 48, //48

    PIXEL_FORMAT_MAX
} PIXEL_FORMAT_E;
```

【成员】 无。

【注意事项】 无。

【相关数据类型及接口】 无。

3.4.19 VIDEO_FRAME_S

【说明】 定义视频图像帧信息

【定义】

```
typedef struct _VIDEO_FRAME_S {
    CVI_U32 u32Width;
    CVI_U32 u32Height;
    PIXEL_FORMAT_E enPixelFormat;
    BAYER_FORMAT_E enBayerFormat;
    VIDEO_FORMAT_E enVideoFormat;
    COMPRESS_MODE_E enCompressMode;
    DYNAMIC_RANGE_E enDynamicRange;
    COLOR_GAMUT_E enColorGamut;
    CVI_U32 u32Stride[3];

    CVI_U64 u64PhyAddr[3];
    CVI_U8 *pu8VirAddr[3];
    CVI_U32 u32Length[3];

    CVI_S16 s16OffsetTop;
    CVI_S16 s16OffsetBottom;
    CVI_S16 s16OffsetLeft;
    CVI_S16 s16OffsetRight;

    CVI_U32 u32TimeRef;
    CVI_U64 u64PTS;

    void *pPrivateData;
    CVI_U32 u32FrameFlag;
} VIDEO_FRAME_S;
```

【成员】

成员名称	描述
u32Width	图像宽度
u32Height	图像高度
enPixelFormat	图像像素格式
enBayerFormat	图像 RAW 格式
enVideoFormat	图像格式
enCompressMode	图像压缩格式
enDynamicRange	动态范围
enColorGamut	色域范围
u32Stride	图像行跨度
u64PhyAddr	物理地址
pu8VirAddr	虚拟地址
u32Length	图像空间大小 (Bytes)
s16OffsetTop	图像顶部剪裁宽度
s16OffsetBottom	图像底部剪裁宽度
s16OffsetLeft	图像左侧剪裁宽度
s16OffsetRight	图像右侧剪裁宽度
u32TimeRef	图像帧序列号
u64PTS	图像时间戳
pPrivateData	私有数据
u32FrameFlag	当前帧的标记

【注意事项】 无。

【相关数据类型及接口】 无。

3.4.20 VIDEO_FRAME_INFO_S

【说明】 定义视频帧信息

【定义】

```
typedef struct _VIDEO_FRAME_INFO_S {
    VIDEO_FRAME_S stVFrame; ///< Video frame
    CVI_U32 u32PoolId;      ///< VB pool ID
} VIDEO_FRAME_INFO_S;
```

【成员】

成员名称	描述
stVFrame	视频帧
u32PoolId	缓存池 ID

【注意事项】 无。

【相关数据类型及接口】 VIDEO_FRAME_S

3.4.21 BITMAP_S

【说明】 定义 BITMAP 信息

【定义】

```
typedef struct _BITMAP_S {
    PIXEL_FORMAT_E enPixelFormat;
    CVI_U32 u32Width;
    CVI_U32 u32Height;
    CVI_VOID * ATTRIBUTE pData;
} BITMAP_S;
```

【成员】

成员名称	描述
enPixelFormat	Bitmap 像素格式
u32Width	Bitmap 宽度
u32Height	Bitmap 高度
pData	Bitmap 数据地址

【注意事项】 无。

【相关数据类型及接口】 无。

3.5 错误码

系统控制错误码

错误代码	宏定义	描述
0xC0028003	CVI_ERR_SYS_ILLEGAL_PARAM	参数设置无效
0xC0028006	CVI_ERR_SYS_NULL_PTR	空指针
0xC0028008	CVI_ERR_SYS_NOT_SUPPORT	不支持的功能
0xC0028009	CVI_ERR_SYS_NOT_PERM	不允许的操作
0xC002800C	CVI_ERR_SYS_NOMEM	内存分配失败，如系统内存不足
0xC002800D	CVI_ERR_SYS_REMAPPING	内存映射失败
0xC0028010	CVI_ERR_SYS_NOTREADY	系统控制属性未配置
0xC0028012	CVI_ERR_SYS_BUSY	系统忙碌中

视频区块池错误码

错误代码	宏定义	描述
0xC0018003	CVI_ERR_VB_ILLEGAL_PARAM	参数设置无效
0xC0018005	CVI_ERR_VB_UNEXIST	视频区块不存在
0xC0018006	CVI_ERR_VB_NULL_PTR	空指针
0xC0018009	CVI_ERR_VB_NOT_PERM	不允许的操作
0xC001800C	CVI_ERR_VB_NOMEM	内存分配失败，如系统内存不足
0xC001800D	CVI_ERR_VB_NOBUF	内存 buff 失败
0xC0018010	CVI_ERR_VB_NOTREADY	系统控制属性未配置
0xC0018012	CVI_ERR_VB_BUSY	系统忙碌中
0xC0018013	CVI_ERR_VB_SIZE_NOT_ENOUGH	VB 块大小不够
0xC0018014	CVI_ERR_VB_INVALID	VB 句柄无效
0xC0018040	CVI_ERR_VB_2MPOOLS	创建缓存池太多

4 视频输入

视频输入（VI）模块实现的功能：通过 MIPI_RX（含 MIPI 接口、LVDS 接口和 HISPI 接口），BT.1120，BT.656，BT.601 等接口接收视频数据，实现视频数据的采集。VI 将接收到的数据存入到指定的内存区域，通过 ISP 对接收到的原始视频图像数据进行处理。

4.1 功能概述

4.1.1 目的

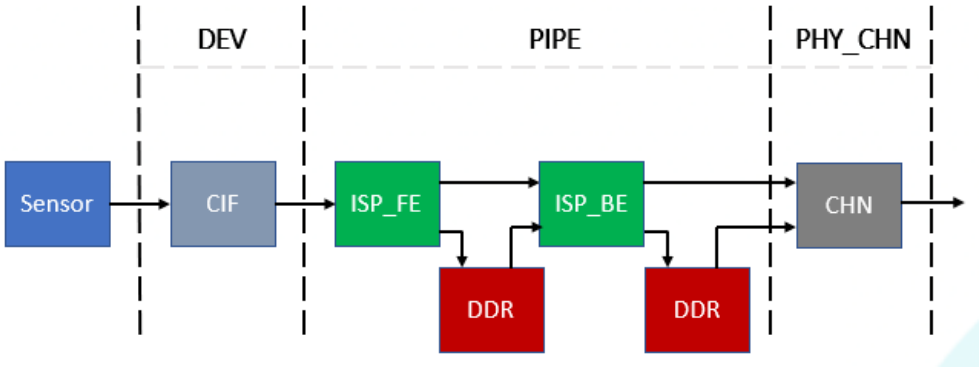
- 视频输入设备支持若干种时序输入，负责对时序进行解析。
- 视频输入 PIPE 绑定在设备后端，负责设备解析后的数据再处理。
- 视频输入负责将最终处理后的数据输出到 DDR。

4.1.2 定义及缩写

- CIF: 对接 sensor 的界面，即 MIPI_RX，BT 等接口。
- ISP_FE: 用来输出 AE，AWB，AF 等统计值相关资料到 DDR。
- ISP_BE: 对影像输入进行质量和色彩相关调整。
- CHN: 影像校正，针对影像进行裁剪、旋转或扭曲校正。
- PHY_PIPE: 物理 PIPE 绑定在设备后端，负责设备解析后的数据再处理。
- PHY_CHN: 视频物理通道负责将最终处理后的数据输出到 DDR，在真正将数据输出到 DDR 之前，它可以实现裁剪等功能。

4.2 设计概述

4.2.1 系统架构



- VI 从软件上划分了输入设备（DEV），影像撷取（ISP_FE），图像处理（ISP_BE）和影像校正（CHN）四个层级。
- Sensor 传输影像透过 CIF 的对接，将影像传递给 ISP_FE，ISP_FE 会撷取出 AE，AWB，AF 等统计值的资料，并将 RAW 文件存到 DDR 上，再透过 ISP_BE 使用在线或脱机模式，将 RAW 文件进行色域的转换以及图像的质量调整，接着将 YUV 存入 DDR，并提供后端 CHN 进行裁剪、旋转等处理。
- 目前处理器的设备的参数配置如下表所示。

表 4.1: 各处理器支持能力

处理器	DEV	PHY_PIPE	PHY_CHN	VIR_CHN	MAX resolution	FPS
CV181x	3	5	5	0	5M(2880x1620)	30
CV180x	1	1	1	0	4M(2560x1440)	30

注意： CV180x 不支持 HDR。

4.2.2 视频输入 PIPE

- VI 的 PIPE 包含了 ISP 的相关处理功能，主要是对图像数据进行流水线处理，输出 YUV 图像格式给通道。PIPE 的工作模式请参考“系统控制”章节的“VI 和 VPSS”的工作模式描述。

4.2.3 视频物理通道

- 每个物理通道具有裁剪功能，支持的典型分辨率如 2880x1620@30fps、2592x1944@30fps、2560x1440@30fps、1080p@30fps 等。

4.2.4 绑定关系

- Dev 和 CIF 绑定关系是固定的，不能动态修改绑定关系。
- Dev 和时序输入接口的约束关系如下表所示。

表 4.2: DEV 与 MIPI/SLVS 接口的绑定关系

VI DEV	MIPI	SLVS	BT.1120/BT.656/BT.601		TTL
0	0	0	0	X	X
1	1	1	1	X	X
2	X	X	X	X	2

4.3 API 参考

视频输入（VI）实现启用视频输入设备、视频输入 PIPE 创建，视频输入信道配置、Dev 绑定 MIPI 设备，PIPE 绑定 Dev 等功能。

该功能模块提供以下 API:

- CVI_VI_SetDevAttr：设置 VI 设备属性。
- CVI_VI_GetDevAttr：获取 VI 设备属性。
- CVI_VI_SetDevAttrEx：设置 VI 设备高级属性。
- CVI_VI_GetDevAttrEx：获取 VI 设备高级属性。
- CVI_VI_EnableDev：启用 VI 设备。
- CVI_VI_DisableDev：禁用 VI 设备。
- CVI_VI_SetDevBindPipe：设置 VI 设备与物理 PIPE 的绑定关系。
- CVI_VI_GetDevBindPipe：获取 VI 设备所绑定的物理 PIPE。
- CVI_VI_SetDevTimingAttr：设置自产生时序属性。
- CVI_VI_GetDevTimingAttr：获取自产生时序属性。
- CVI_VI_CreatePipe：创建一个 VI PIPE。
- CVI_VI_DestroyPipe：销毁一个 VI PIPE。
- CVI_VI_SetPipeAttr：设置 VI PIPE 的属性。
- CVI_VI_GetPipeAttr：获取 VI PIPE 的属性。
- CVI_VI_StartPipe：启用 VI PIPE。

- CVI_VI_StopPipe： 禁用 VI PIPE。
- CVI_VI_SetPipeCrop： 设置 VI 物理 PIPE 裁剪功能属性。
- CVI_VI_GetPipeCrop： 获取 VI 物理 PIPE 裁剪功能属性。
- CVI_VI_SetPipeDumpAttr： 设置 VI 物理 PIPE dump 属性。
- CVI_VI_GetPipeDumpAttr： 获取 VI 物理 PIPE dump 属性。
- CVI_VI_SetPipeFrameSource： 设置 VI PIPE 数据的来源。
- CVI_VI_GetPipeFrameSource： 获取 VI PIPE 数据的来源。
- CVI_VI_GetPipeFrame： 获取 VI 物理 PIPE 的数据。
- CVI_VI_ReleasePipeFrame： 释放 VI PIPE 的数据。
- CVI_VI_SendPipeRaw： 通过 VI PIPE 发送 RAW 数据。
- CVI_VI_QueryPipeStatus： 查看 VI PIPE 状态。
- CVI_VI_GetPipeFd： 获取 VI PIPE 文件描述符。
- CVI_VI_CloseFd： 关闭 VI 文件描述符。
- CVI_VI_AttachVbPool： 将 VI 通道绑定到某个视频缓存 VB 池中。
- CVI_VI_DetachVbPool：将 VI 通道从某个视频缓存 VB 池中解绑定。
- CVI_VI_SetChnAttr： 设置 VI 通道属性。
- CVI_VI_GetChnAttr： 获取 VI 通道属性。
- CVI_VI_EnableChn： 启用 VI 通道。
- CVI_VI_DisableChn： 禁用 VI 通道。
- CVI_VI_SetChnCrop： 设置 VI 信道裁剪功能属性。
- CVI_VI_GetChnCrop： 获取 VI 信道裁剪功能属性。
- CVI_VI_GetChnFrame： 从 VI 通道获取采集的图像。
- CVI_VI_ReleaseChnFrame： 释放一帧从 VI 通道获取的图像。
- CVI_VI_SetChnRotation： 设置 VI 通道旋转的属性。
- CVI_VI_GetChnRotation： 获取 VI 通道旋转的属性。
- CVI_VI_RegChnFlipMirrorCallBack： 注册 VI 通道翻转镜像回调函数。
- CVI_VI_UnRegChnFlipMirrorCallBack： 注销 VI 通道翻转镜像回调函数。
- CVI_VI_SetChnFlipMirror： 设置 VI 通道翻转镜像的属性。
- CVI_VI_GetChnFlipMirror： 获取 VI 通道翻转镜像的属性。
- CVI_VI_Suspend: 休眠 VI 设备。
- CVI_VI_Resume: 唤醒 VI 设备。
- CVI_VI_SetDevNum: 设置 VI 设备数量。
- CVI_VI_GetDevNum: 获取 VI 设备数量。
- CVI_VI_EnablePatt: 启用 VI pattern 模式。

- CVI_VI_SetUserPic: 设置用户输入图片，现基本不用。
- CVI_VI_EnableUserPic: 启用用户输入图片，现基本不用。
- CVI_VI_DisableUserPic: 禁用用户输入图片，现基本不用。
- CVI_VI_StartSmoothRawDump: 开始 dump 平滑 raw 图。
- CVI_VI_StopSmoothRawDump: 停止 dump 平滑 raw 图。
- CVI_VI_GetSmoothRawDump: 获取 dump 下来的平滑 raw 图。
- CVI_VI_PutSmoothRawDump: 放置 dump 下来的平滑 raw 图。
- CVI_VI_GetRgbMapLeBuf: 获取 rgbmap 长帧缓冲区。
- CVI_VI_GetRgbMapSeBuf: 获取 rgbmap 短帧缓冲区。
- CVI_VI_DumpHwRegisterToFile: 下载硬件寄存器数据到指定文件。
- CVI_VI_QueryChnStatus: 查看 VI 通道状态。
- CVI_VI_GetChnFd: 获取 VI 通道描述符。
- CVI_VI_SetChnAlign: 设置 VI 通道对齐数。
- CVI_VI_GetChnAlign: 获取 VI 通道对齐数。
- CVI_VI_RegPmCallBack: 注册 VI 设备电源管理回调函数。
- CVI_VI_UnRegPmCallBack: 注销 VI 设备电源管理回调函数。
- CVI_VI_SetTuningDis: 设置 tuning 参数。

4.3.1 CVI_VI_SetDevAttr

【描述】

设置 VI 设备属性。基本设备属性默认了部分处理器配置。

【语法】

```
CVI_S32 CVI_VI_SetDevAttr(VI_DEV ViDev, const VI_DEV_ATTR_S *pstDevAttr);
```

【参数】

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

【返回值】

返回值	描述
0	成功。
非 0	失败，请参见错误码 。

【需求】

- 头文件：cvi_vi.h, cvi_comm_vi.h
- 库文件：libvpu.a

【注意】

- 在调用前要保证 VI 设备处于禁用状态。如果 VI 设备已处于使能状态，可以使用 CVI_VI_DisableDev 来禁用设备。
- 参数 pstDevAttr 主要用来配置指定 VI 设备的视频接口模式，用于与外围 camera、sensor 或 codec 对接，支持的接口模式包括 MIPI_RX（MIPI/LVDS/HISPI）。
- 用户需要配置以下几类信息，具体属性意义参见“数据类型”部分的说明：
  - 接口模式信息：接口模式为 MIPI Rx（MIPI/LVDS/HISPI）等模式
  - 工作模式信息：1 路复合模式
  - 数据布局信息：YUV 数据输入下的数据排列
  - 数据信息：RGB、YUV 数据输入
  - 同步时序信息：垂直、水平同步信号的属性

【举例】

```
VI_DEV_ATTR_S DEV_ATTR_IMX327_2M_BASE = {
 VI_MODE_MIPI,
 VI_WORK_MODE_1Multiplex,
 VI_SCAN_PROGRESSIVE,
 {-1, -1, -1, -1},
 VI_DATA_SEQ_YUYV,

 {
 /*port_vsync   port_vsync_neg     port_hsync       port_hsync_neg       */
 VI_VSYNC_PULSE, VI_VSYNC_NEG_LOW, VI_HSYNC_VALID_SINGNAL, VI_HSYNC_
↪NEG_HIGH,
 VI_VSYNC_VALID_SIGNAL, VI_VSYNC_VALID_NEG_HIGH,

 /*hsync_hfb    hsync_act    hsync_hhb*/
 {0,1920,       0,
 /*vsync0_vhb vsync0_act vsync0_hhb*/
 0,1080,       0,
 /*vsync1_vhb vsync1_act vsync1_hhb*/
 0,0,0}
 },
 VI_DATA_TYPE_RGB,
 {1920, 1080},
 {
  WDR_MODE_NONE,
  1080
 },
 .enBayerFormat = BAYER_FORMAT_RG,
};

int main(void)
{
```

（下页继续）

(续上页)

```
  SAMPLE_SNS_TYPE_E  enSnsType   = SONY_IMX327_MIPI_2M_30FPS_12BIT;
  VI_DEV_ATTR_S      stViDevAttr;

  SAMPLE_COMM_VI_GetDevAttrBySns(enSnsType, &stViDevAttr);
  s32Ret = CVI_VI_SetDevAttr(0, &stViDevAttr);
  if (s32Ret != CVI_SUCCESS) {
    SAMPLE_PRT("CVI_VI_SetDevAttr failed with %#x\n", s32Ret);
    return s32Ret;
  }

  s32Ret = CVI_VI_EnableDev(0);
  if (s32Ret != CVI_SUCCESS) {
    SAMPLE_PRT("CVI_VI_EnableDev failed with %#x\n", s32Ret);
    return s32Ret;
  }

  CVI_VI_DisableDev(0);

  s32Ret = CVI_VI_StopPipe(ViPipe);
  if (s32Ret != CVI_SUCCESS) {
    SAMPLE_PRT("CVI_VI_StopPipe failed with %#x!\n", s32Ret);
    return s32Ret;
  }

  s32Ret = CVI_VI_DestroyPipe(ViPipe);
  if (s32Ret != CVI_SUCCESS) {
    SAMPLE_PRT("CVI_VI_DestroyPipe failed with %#x!\n", s32Ret);
    return s32Ret;
  }
}
```

【相关主题】

CVI_VI_GetDevAttr

4.3.2 CVI_VI_GetDevAttr

【描述】

获取 VI 设备属性。

【语法】

```
CVI_S32 CVI_VI_GetDevAttr(VI_DEV ViDev, VI_DEV_ATTR_S *pstDevAttr);
```

【参数】

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

【返回值】

返回值	描述
0	成功。
非 0	失败，请参见错误码 。

【需求】

- 头文件：cvi_vi.h, cvi_comm_vi.h
- 库文件：libvpu.a

【注意】

如果未设置 VI 设备属性，该接口将返回失败。

【举例】

无

【相关主题】

CVI_VI_SetDevAttr

4.3.3 CVI_VI_SetDevAttrEx

【描述】

设置 VI 设备高级属性。

【语法】

```
CVI_S32 CVI_VI_SetDevAttrEx(VI_DEV ViDev, const VI_DEV_ATTR_EX_S *pstDevAttrEx);
```

【参数】

参数名称	描述	输入/输出
ViDev	VI 设备号。取值范围：[0, VI_MAX_DEV_NUM)。	输入
pstDevAttrEx	VI 设备高级属性指针。静态属性。	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，请参见错误码 。

【需求】

- 头文件：cvi_vi.h, cvi_comm_vi.h
- 库文件：libvpu.a

【注意】

· 暂不支持此接口。

【举例】

无

【相关主题】

无

4.3.4 CVI_VI_GetDevAttrEx

【描述】

获取 VI 设备高级属性。

【语法】

```
CVI_S32 CVI_VI_GetDevAttrEx(VI_DEV ViDev, VI_DEV_ATTR_EX_S *pstDevAttrEx);
```

【参数】

参数名称	描述	输入/输出
ViDev	VI 设备号。取值范围：[0, VI_MAX_DEV_NUM)。	输入
pstDevAttrEx	VI 设备高级属性指针。静态属性。	输出

【返回值】

返回值	描述
0	成功。
非 0	失败，请参见错误码 。

【需求】

· 头文件：cvi_vi.h, cvi_comm_vi.h

· 库文件：libvpu.a

【注意】

· 暂不支持此接口。

【举例】

无

【相关主题】

无

4.3.5 CVI_VI_EnableDev

【描述】

启用 VI 设备。

【语法】

```
CVI_S32 CVI_VI_EnableDev(VI_DEV ViDev);
```

【参数】

参数名称	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，请参见错误码 。

【需求】

- 头文件：cvi_vi.h, cvi_comm_vi.h
- 库文件：libvpu.a

【注意】

- 如果未设置 VI 设备属性，该接口将返回失败。
- CV181x 支持同时启动三个 VI DEV。
- CV180x 支持同时启动一个 VI DEV。

【举例】

无

【相关主题】

CVI_VI_DisableDev

4.3.6 CVI_VI_DisableDev

【描述】

禁用 VI 设备。

【语法】

```
CVI_S32 CVI_VI_DisableDev(VI_DEV ViDev);
```

【参数】

参数名称	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，请参见错误码 。

【需求】

- 头文件：cvi_vi.h, cvi_comm_vi.h
- 库文件：libvpu.a

【注意】

- 建议先销毁所有与该 VI 设备绑定的物理 PIPE 后，再禁用 VI 设备。
- 禁用 VI 设备后将完全关闭该设备，需要重新设置属性，才能使能 VI 设备。

【举例】

无

【相关主题】

CVI_VI_EnableDev

4.3.7 CVI_VI_SetDevBindPipe

【描述】

设置 VI 设备与物理 PIPE 的绑定关系。

【语法】

```
CVI_S32 CVI_VI_SetDevBindPipe(VI_DEV ViDev, const VI_DEV_BIND_PIPE_S␣
↪*pstDevBindPipe);
```

【参数】

参数名称	描述	输入/输出
ViDev	VI 设备号。取值范围：[0, VI_MAX_DEV_NUM)。	输入/输出
pstDevBindPipe	绑定到 Dev 的物理 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，请参见错误码 。

【需求】

- 头文件：cvi_vi.h, cvi_comm_vi.h
- 库文件：libvpu.a

【注意】

- 必须先使能 VI 设备后才能绑定物理 PIPE。
- 此接口暂无使用。

【举例】

无

【相关主题】

CVI_VI_EnableDev

4.3.8 CVI_VI_GetDevBindPipe

【描述】

获取 VI 设备所绑定的物理 PIPE。

【语法】

```
CVI_S32 CVI_VI_GetDevBindPipe(VI_DEV ViDev, VI_DEV_BIND_PIPE_S *pstDevBindPipe);
```

【参数】

参数名称	描述	输入/输出
ViDev	VI 设备号。取值范围：[0, VI_MAX_DEV_NUM)。	输入/输出
pstDevBindPipe	绑定到 Dev 的物理 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，请参见错误码 。

【需求】

- 头文件：cvi_vi.h, cvi_comm_vi.h
- 库文件：libvpu.a

【注意】

- 使用本接口前，需使能设备并绑定物理 PIPE，否则返回失败。
- 此接口暂无使用。

【举例】

无

【相关主题】

CVI_VI_SetDevBindPipe

4.3.9 CVI_VI_SetDevTimingAttr

【描述】

设置自产生时序属性。

【语法】

```
CVI_S32 CVI_VI_SetDevTimingAttr(VI_DEV ViDev, const VI_DEV_TIMING_ATTR_S
↪*pstTimingAttr);
```

【参数】

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

【返回值】

返回值	描述
0	成功。
非 0	失败，请参见错误码 。

【需求】

- 头文件：cvi_vi.h, cvi_comm_vi.h
- 库文件：libvpu.a

【注意】

- 使用本接口前，需先配置 DEV 属性，并使能设备，否则返回失败。
- 使用自产生时序功能灌 RAW 时，需配置 DEV/ PIPE/ CHN 的宽高与 RAW 文件的宽高保持一致。
- 使能自产生时序后，若不灌 RAW，则无图像显示；灌 RAW 方式请参考 CVI_VI_SendPipeRaw 的描述。
- 使能自产生时序后，VI 输出帧率由配置自产生时序产生的有效帧率决定。

【举例】

无

【相关主题】

CVI_VI_EnableDev

CVI_VI_SetDevAttr

4.3.10 CVI_VI_GetDevTimingAttr

【描述】

获取自产生时序属性。

【语法】

```
CVI_S32 CVI_VI_GetDevTimingAttr(VI_DEV ViDev, VI_DEV_TIMING_ATTR_S␣
↪*pstTimingAttr);
```

【参数】

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

【返回值】

返回值	描述
0	成功。
非 0	失败，请参见错误码 。

【需求】

- 头文件：cvi_vi.h, cvi_comm_vi.h
- 库文件：libvpu.a

【注意】

- 使用本接口前，需先配置 DevTimingAttr，并使能设备，否则返回失败。

【举例】

无

【相关主题】

CVI_VI_EnableDev

CVI_VI_SetDevTimingAttr

4.3.11 CVI_VI_CreatePipe

【描述】

创建一个 VI PIPE。

【语法】

```
CVI_S32 CVI_VI_CreatePipe(VI_PIPE ViPipe, const VI_PIPE_ATTR_S *pstPipeAttr);
```

【参数】

参数名称	描述	输入/输出
ViPipe	VI PIPE 号。取值范围：[0, VI_MAX_PIPE_NUM)。	输入
pstPipeAttr	PIPE 的属性结构体指针。详见VI_PIPE_ATTR_S 说明。	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，请参见错误码 。

【需求】

- 头文件：cvi_vi.h, cvi_comm_vi.h
- 库文件：libvpu.a

【注意】

- 不支持重复创建。

【举例】

无

【相关主题】

CVI_VI_EnableDev

4.3.12 CVI_VI_DestroyPipe

【描述】

销毁一个 VI PIPE。

【语法】

```
CVI_S32 CVI_VI_DestroyPipe(VI_PIPE ViPipe);
```

【参数】

参数名称	描述	输入/输出
ViPipe	VI PIPE 号。取值范围：[0, VI_MAX_PIPE_NUM)。	输入
pstPipeAttr	PIPE 的属性结构体指针。	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，请参见错误码 。

【需求】

- 头文件：cvi_vi.h, cvi_comm_vi.h
- 库文件：libvpu.a

【注意】

- 在未创建 PIPE 或重复销毁 PIPE 时，调用本接口，将提示 PIPE 不存在。

【举例】

无

【相关主题】

CVI_VI_CreatePipe

4.3.13 CVI_VI_SetPipeAttr

【描述】

设置 VI PIPE 的属性。

【语法】

```
CVI_S32 CVI_VI_SetPipeAttr(VI_PIPE ViPipe, const VI_PIPE_ATTR_S *pstPipeAttr);
```

【参数】

参数名称	描述	输入/输出
ViPipe	VI PIPE 号。取值范围：[0, VI_MAX_PIPE_NUM)。	输入
pstPipeAttr	PIPE 的属性结构体指针。详见VI_PIPE_ATTR_S 说明	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，请参见错误码 。

【需求】

- 头文件：cvi_vi.h, cvi_comm_vi.h
- 库文件：libvpu.a

【注意】

- 使用本接口前，需先调用 CVI_VI_CreatePipe，否则提示失败。
- PIPE 属性必须合法，其中部分静态属性不可动态设置，具体请参见VI_PIPE_ATTR_S 。

【举例】

无

【相关主题】

无

4.3.14 CVI_VI_GetPipeAttr

【描述】

获取 VI PIPE 的属性。

【语法】

```
CVI_S32 CVI_VI_GetPipeAttr(VI_PIPE ViPipe, VI_PIPE_ATTR_S *pstPipeAttr);
```

【参数】

参数名称	描述	输入/输出
ViPipe	VI PIPE 号。取值范围：[0, VI_MAX_PIPE_NUM)。	输入
pstPipeAttr	PIPE 的属性结构体指针。详见VI_PIPE_ATTR_S 说明。	输出

【返回值】

返回值	描述
0	成功。
非 0	失败，请参见错误码 。

【需求】

- 头文件：cvi_vi.h, cvi_comm_vi.h
- 库文件：libvpu.a

【注意】

- 使用本接口前，需先调用CVI_VI_SetPipeAttr 。

【举例】

无

【相关主题】

CVI_VI_SetPipeAttr

4.3.15 CVI_VI_StartPipe

【描述】

启用 VI PIPE。

【语法】

```
CVI_S32 CVI_VI_StartPipe(VI_PIPE ViPipe);
```

【参数】

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

【返回值】

返回值	描述
0	成功。
非 0	失败，请参见错误码 。

【需求】

- 头文件：cvi_vi.h, cvi_comm_vi.h
- 库文件：libvpu.a

【注意】

- 使用本接口前，需先调用CVI_VI_CreatePipe 。

【举例】

无

【相关主题】

CVI_VI_CreatePipe

4.3.16 CVI_VI_StopPipe

【描述】

禁用 VI PIPE。

【语法】

```
CVI_S32 CVI_VI_StopPipe(VI_PIPE ViPipe);
```

【参数】

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

【返回值】

返回值	描述
0	成功。
非 0	失败，请参见错误码 。

【需求】

- 头文件：cvi_vi.h, cvi_comm_vi.h
- 库文件：libvpu.a

【注意】

- 使用本接口前，Pipe 必须已创建。

【举例】

无

【相关主题】

CVI_VI_CreatePipe

4.3.17 CVI_VI_SetPipeCrop

【描述】

设置 VI 物理 PIPE 裁剪功能属性。

【语法】

```
CVI_S32 CVI_VI_SetPipeCrop(VI_PIPE ViPipe, const CROP_INFO_S *pstCropInfo);
```

【参数】

参数名称	描述	输入/输出
ViPipe	VI 物理 PIPE 号。	输入
pstCropInfo	裁剪功能参数结构体指针。	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，请参见错误码 。

【需求】

- 头文件：cvi_vi.h, cvi_comm_vi.h
- 库文件：libvpu.a

【注意】

- 使用本接口前，Pipe 必须已创建。
- 本接口与CVI_VI_SetChnCrop 有同样效果。

【举例】

无

【相关主题】

无

4.3.18 CVI_VI_GetPipeCrop

【描述】

获取 VI 物理 PIPE 裁剪功能属性。

【语法】

```
CVI_S32 CVI_VI_GetPipeCrop(VI_PIPE ViPipe, CROP_INFO_S *pstCropInfo);
```

【参数】

参数名称	描述	输入/输出
ViPipe	VI 物理 PIPE 号。	输入
pstCropInfo	裁剪功能参数结构体指针。	输出

【返回值】

返回值	描述
0	成功。
非 0	失败，请参见错误码 。

【需求】

- 头文件：cvi_vi.h, cvi_comm_vi.h
- 库文件：libvpu.a

【注意】

- 使用本接口前，Pipe 必须已创建。

【举例】

无

【相关主题】

无

4.3.19 CVI_VI_SetPipeDumpAttr

【描述】

设置 VI 物理 PIPE dump 属性。

【语法】

```
CVI_S32 CVI_VI_SetPipeDumpAttr(VI_PIPE ViPipe, const VI_DUMP_ATTR_S *pstDumpAttr);
```

【参数】

参数名称	描述	输入/输出
ViPipe	物理 PIPE 号。取值范围：[0, VI_MAX_PHY_PIPE_NUM - 1)。	输入/输出
pstDumpAttr	VI 物理 PIPE dump 的属性。详见VI_DUMP_ATTR_S 说明	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，请参见错误码 。

【需求】

- 头文件：cvi_vi.h, cvi_comm_vi.h
- 库文件：libvpu.a

【注意】

- 支持 Dump 12-bit 非压缩/6-bit 压缩的 RAW 数据。
- 如果是 YUV 数据建议使用 CVI_VI_GetChnFrame。
- 暂不支持 Dump IR 数据。

【举例】

无

【相关主题】

无

4.3.20 CVI_VI_GetPipeDumpAttr

【描述】

获取 VI 物理 PIPE dump 属性。

【语法】

```
CVI_S32 CVI_VI_GetPipeDumpAttr(VI_PIPE ViPipe, VI_DUMP_ATTR_S *pstDumpAttr);
```

【参数】

参数名称	描述	输入/输出
ViPipe	物理 PIPE 号。取值范围：[0, VI_MAX_PHY_PIPE_NUM - 1)。	输入
pstDumpAttr	VI 物理 PIPE dump 的属性。详见VI_DUMP_ATTR_S 说明。	输出

【返回值】

返回值	描述
0	成功。
非 0	失败，请参见错误码 。

【需求】

- 头文件：cvi_vi.h, cvi_comm_vi.h
- 库文件：libvpu.a

【注意】

- PIPE 必须已创建。

【举例】

无

【相关主题】

CVI_VI_SetPipeDumpAttr

4.3.21 CVI_VI_SetPipeFrameSource

【描述】

设置 VI PIPE 数据的来源。

【语法】

```
CVI_S32 CVI_VI_SetPipeFrameSource(VI_PIPE ViPipe, const VI_PIPE_FRAME_SOURCE_E
↪enSource);
```

【参数】

参数名称	描述	输入/输出
ViPipe	VI PIPE 号。取值范围：[0, VI_MAX_PIPE_NUM)。	输入
enSource	PIPE 的数据来源。	输出

【返回值】

返回值	描述
0	成功。
非 0	失败，请参见错误码 。

【需求】

- 头文件：cvi_vi.h, cvi_comm_vi.h
- 库文件：libvpu.a

【注意】

- PIPE 必须已创建。

【举例】

无

【相关主题】

无

4.3.22 CVI_VI_GetPipeFrameSource

【描述】

获取 VI PIPE 数据的来源。

【语法】

```
CVI_S32 CVI_VI_GetPipeFrameSource(VI_PIPE ViPipe, VI_PIPE_FRAME_SOURCE_E
↪*penSource);
```

【参数】

参数名称	描述	输入/输出
ViPipe	VI PIPE 号。取值范围：[0, VI_MAX_PIPE_NUM)。	输入
penSource	PIPE 的数据来源。	输出

【返回值】

返回值	描述
0	成功。
非 0	失败，请参见错误码 。

【需求】

- 头文件：cvi_vi.h, cvi_comm_vi.h
- 库文件：libvpu.a

【注意】

- PIPE 必须已创建。

【举例】

无

【相关主题】

无

4.3.23 CVI_VI_GetPipeFrame

【描述】

获取 VI 物理 PIPE 的数据。

【语法】

```
CVI_S32 CVI_VI_GetPipeFrame(VI_PIPE ViPipe, VIDEO_FRAME_INFO_S *pstVideoFrame,
↪CVI_S32 s32MilliSec);
```

【参数】

参数名称	描述	输入/输出
ViPipe	物理 PIPE 号。取值范围：[0, VI_MAX_PHY_PIPE_NUM - 1)。	输入
pstVideoFrame	VI PIPE 数据信息的指针。	输出
s32MilliSec	超时参数，超时时间的单位为毫秒（ms）。	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，请参见错误码 。

【需求】

- 头文件：cvi_vi.h, cvi_comm_vi.h
- 库文件：libvpu.a

【注意】

- PIPE 必须已创建。
- 调用 CVI_VI_SetPipeDumpAttr 设置 dump 属性、使能 dump、设置 depth，否则获取不到 raw 数据。
- 超时参数 s32MilliSec，设置范围最小为 0。

【举例】

无

【相关主题】

CVI_VI_SetPipeDumpAttr

4.3.24 CVI_VI_ReleasePipeFrame

【描述】

释放 VI PIPE 的数据。

【语法】

```
CVI_S32 CVI_VI_ReleasePipeFrame(VI_PIPE ViPipe, const VIDEO_FRAME_INFO_S␣
↪*pstVideoFrame);
```

【参数】

参数名称	描述	输入/输出
ViPipe	物理 PIPE 号。取值范围：[0, VI_MAX_PHY_PIPE_NUM - 1)。	输入
pstVideoFrame	VI PIPE 数据信息的指针。	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，请参见错误码 。

【需求】

- 头文件：cvi_vi.h, cvi_comm_vi.h
- 库文件：libvpu.a

【注意】

- 本接口需与 CVI_VI_GetPipeFrame 匹配使用，GetPipeFrame 后不 Release 会导致系统内存不足，进而影响 VI 运作。
- 用户必须保证 pstVideoFrame 结构中的信息与 GetPipeFrame 时一致，否则会造成释放不成功。

【举例】

无

【相关主题】

CVI_VI_GetPipeFrame

4.3.25 CVI_VI_SendPipeRaw

【描述】

通过 VI PIPE 发送 RAW 数据。

【语法】

```
CVI_S32 CVI_VI_SendPipeRaw(CVI_U32 u32PipeNum, VI_PIPE PipeId[], const VIDEO_FRAME_
↪INFO_S *pstVideoFrame[], CVI_S32 s32MilliSec);
```

【参数】

参数名称	描述	输入/输出
u32PipeNum	Pipe 数目，固定填 1 值。	输入
PipeId	PIPE 号数组，个数为 u32PipeNum。	输入/输出
pstVideoFrame	RAW 数据信息。	输入
s32MilliSec	超时参数，大于 0 表示超时模式，超时时间的单位为毫秒（ms）。	输出

【返回值】

返回值	描述
0	成功。
非 0	失败，请参见错误码 。

【需求】

- 头文件：cvi_vi.h, cvi_comm_vi.h
- 库文件：libvpu.a

【注意】

- PIPE 必须已创建且启动。
- 送 RAW 首先需要通过CVI_VI_SetPipeFrameSource 来设置 PIPE 的数据来源。
- 支持用户自定义时序灌 RAW。
- 只支持送非压模式 RAW 数据。

【举例】

无

【相关主题】

CVI_VI_SetPipeFrameSource

4.3.26 CVI_VI_QueryPipeStatus

【描述】

查询 VI PIPE 状态。

【语法】

```
CVI_S32 CVI_VI_QueryPipeStatus(VI_PIPE ViPipe, VI_PIPE_STATUS_S *pstStatus);
```

【参数】

参数名称	描述	输入/输出
ViPipe	PIPE 号。	输入
pstStatus	PIPE 状态信息。	输入/输出

【返回值】

返回值	描述
0	成功。
非 0	失败，请参见错误码 。

【需求】

- 头文件：cvi_vi.h, cvi_comm_vi.h
- 库文件：libvpu.a

【注意】

- PIPE 必须已创建且启动。

【举例】

无

【相关主题】

无

4.3.27 CVI_VI_GetPipeFd

【描述】

获取 VI PIPE 文件描述符。

【语法】

```
CVI_S32 CVI_VI_GetPipeFd(VI_PIPE ViPipe);
```

【参数】

参数名称	描述	输入/输出
ViPipe	PIPE 号。	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，请参见错误码 。

【需求】

- 头文件：cvi_vi.h, cvi_comm_vi.h
- 库文件：libvpu.a

【注意】

- PIPE 必须已创建且启动。

【举例】

无

【相关主题】

无

4.3.28 CVI_VI_CloseFd

【描述】

关闭 VI 文件描述符。

【语法】

```
CVI_S32 CVI_VI_CloseFd(void);
```

【参数】

无

【返回值】

返回值	描述
0	成功。
非 0	失败，请参见错误码 。

【需求】

- 头文件：cvi_vi.h, cvi_comm_vi.h
- 库文件：libvpu.a

【注意】

- 暂不支持此接口。

【举例】

无

【相关主题】

无

4.3.29 CVI_VI_AttachVbPool

【描述】

将 VI 通道绑定到某个视频缓存 VB 池中。

【语法】

```
CVI_S32 CVI_VI_AttachVbPool(VI_PIPE ViPipe, VI_CHN ViChn, VB_POOL VbPool);
```

【参数】

参数名称	描述	输入/输出
ViPipe	PIPE 号。	输入
ViChn	VI 通道号。	输入
VbPool	视频缓存 VB 池信息。	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，请参见错误码 。

【需求】

- 头文件：cvi_vi.h，cvi_comm_vb.h
- 库文件：libvpu.a

【注意】

- ViPipe、ViChn 必须符合取值范围。
- VbPool 必须保证是已创建 VB 池的有效 PoolId。

【举例】

无

【相关主题】

无

4.3.30 CVI_VI_DetachVbPool

【描述】

将 VI 通道从某个视频缓存 VB 池中解绑定。

【语法】

```
CVI_S32 CVI_VI_DetachVbPool(VI_PIPE ViPipe, VI_CHN ViChn);
```

【参数】

参数名称	描述	输入/输出
ViPipe	PIPE 号。	输入
ViChn	VI 通道号。	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，请参见错误码 。

【需求】

- 头文件：cvi_vi.h
- 库文件：libvpu.a

【注意】

无

【举例】

无

【相关主题】

无

4.3.31 CVI_VI_SetChnAttr

【描述】

设置 VI 通道属性。

【语法】

```
CVI_S32 CVI_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	PIPE 号。	输入
ViChn	VI 通道号。	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，请参见错误码 。

【需求】

- 头文件：cvi_vi.h, cvi_comm_vi.h
- 库文件：libvpu.a

【注意】

- 目标图像大小 stSize：必须配置，且大小需要在 VI 支持的范围内。
- 像素格式 enPixelFormat：输出像素格式支持 NV21。
- 帧率控制 stFrameRate：不支持帧率控制。

【举例】

无

【相关主题】

无

4.3.32 CVI_VI_GetChnAttr

【描述】

获取 VI 通道属性。

【语法】

```
CVI_S32 CVI_VI_GetChnAttr(VI_PIPE ViPipe, VI_CHN ViChn, VI_CHN_ATTR_S
↪*pstChnAttr);
```

【参数】

参数名称	描述	输入/输出
ViPipe	PIPE 号。	输入
ViChn	VI 通道号。	输入
pstChnAttr	VI 通道属性结构体指针。	输出

【返回值】

返回值	描述
0	成功。
非 0	失败，请参见错误码 。

【需求】

- 头文件：cvi_vi.h, cvi_comm_vi.h
- 库文件：libvpu.a

【注意】

- PIPE 必须已创建，否则会返回失败。
- 通道属性需先设置后，才可获取。

【举例】

无

【相关主题】

无

4.3.33 CVI_VI_EnableChn

【描述】

启用 VI 通道。

【语法】

```
CVI_S32 CVI_VI_EnableChn(VI_PIPE ViPipe, VI_CHN ViChn);
```

【参数】

参数名称	描述	输入/输出
ViPipe	PIPE 号。	输入
ViChn	VI 通道号。	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，请参见错误码 。

【需求】

- 头文件：cvi_vi.h, cvi_comm_vi.h
- 库文件：libvpu.a

【注意】

- PIPE 必须已创建，否则会返回失败。
- 必须先设置通道属性。
- 不支持重复启用 VI 通道。

【举例】

无

【相关主题】

无

4.3.34 CVI_VI_DisableChn

【描述】

禁用 VI 通道。

【语法】

```
CVI_S32 CVI_VI_DisableChn(VI_PIPE ViPipe, VI_CHN ViChn);
```

【参数】

参数名称	描述	输入/输出
ViPipe	PIPE 号。	输入
ViChn	VI 通道号。	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，请参见错误码 。

【需求】

- 头文件：cvi_vi.h, cvi_comm_vi.h
- 库文件：libvpu.a

【注意】

- PIPE 必须已创建，否则会返回失败。
- 必须先设置通道属性。
- VI 通道须要先使能。

【举例】

无

【相关主题】

无

4.3.35 CVI_VI_SetChnCrop

【描述】

设置 VI 通道裁剪功能属性。

【语法】

```
CVI_S32 CVI_VI_SetChnCrop(VI_PIPE ViPipe, VI_CHN ViChn, const VI_CROP_INFO_S␣
↪*pstCropInfo);
```

【参数】

参数名称	描述	输入/输出
ViPipe	PIPE 号。	输入
ViChn	VI 通道号。	输入
pstCropInfo	裁剪功能参数结构体指针。	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，请参见错误码 。

【需求】

- 头文件：cvi_vi.h, cvi_comm_vi.h
- 库文件：libvpu.a

【注意】

- PIPE 必须已创建，否则会返回失败。
- 使用此 API 进行 Crop，会让部分画面丢失。
- 使用此 API 进行 Crop 后，VPSS 的输入 size 必须和 pstCropInfo 的 size 一样。

【举例】

无

【相关主题】

无

4.3.36 CVI_VI_GetChnCrop

【描述】

获取 VI 通道裁剪功能属性。

【语法】

```
CVI_S32 CVI_VI_GetChnCrop(VI_PIPE ViPipe, VI_CHN ViChn, VI_CROP_INFO_S
↪*pstCropInfo);
```

【参数】

参数名称	描述	输入/输出
ViPipe	PIPE 号。	输入
ViChn	VI 通道号。	输入
pstCropInfo	裁剪功能参数结构体指针	输出

【返回值】

返回值	描述
0	成功。
非 0	失败，请参见错误码 。

【需求】

- 头文件：cvi_vi.h, cvi_comm_vi.h
- 库文件：libvpu.a

【注意】

- PIPE 必须已创建，否则会返回失败。
- 必须先使用 SetChnCrop 设置属性，才能获得属性。

【举例】

无

【相关主题】

无

4.3.37 CVI_VI_GetChnFrame

【描述】

从 VI 通道获取采集的图像。

【语法】

```
CVI_S32 CVI_VI_GetChnFrame(VI_PIPE ViPipe, VI_CHN ViChn, VIDEO_FRAME_INFO_S␣
↪*pstFrameInfo, CVI_S32 s32MilliSec);
```

【参数】

参数名称	描述	输入/输出
ViPipe	PIPE 号。	输入
ViChn	VI 通道号。	输入
pstFrameInfo	VI 帧信息结构指针。	输出
s32MilliSec	超时参数，-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，请参见错误码 。

【需求】

- 头文件：cvi_vi.h, cvi_comm_vi.h
- 库文件：libvpu.a

【注意】

- PIPE 必须已创建，否则会返回失败。

- 必须先设置通道属性。
- 获取的物理地址信息来自内部使用的 VideoBuffer，因此使用完之后，必须要调用 CVI_VI_ReleaseChnFrame 接口释放其内存。
- pstFrameInfo->stVFrame.u64PhyAddr[0] 和 pstFrameInfo->stVFrame.u64PhyAddr[1]/[2] 分别指向图像的亮度分量和色度分量的物理地址。
- 如果 u32MilliSec 设为 -1 时，表示阻塞模式，程序一直等待，直到获取到图像才返回。
- 如果 u32MilliSec 大于 0 时，表示非阻塞模式，参数的单位是毫秒，指超时时间，在此时间内如果没有获取到图像，则超时返回。

【举例】

无

【相关主题】

- CVI_VI_ReleaseChnFrame
- CVI_VI_SetChnAttr

4.3.38 CVI_VI_ReleaseChnFrame

【描述】

释放一帧从 VI 通道获取的图像。

【语法】

```
CVI_S32 CVI_VI_ReleaseChnFrame(VI_PIPE ViPipe, VI_CHN ViChn, const VIDEO_FRAME_
↪INFO_S *pstFrameInfo);
```

【参数】

参数名称	描述	输入/输出
ViPipe	PIPE 号。	输入
ViChn	VI 通道号。	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，请参见错误码 。

【需求】

- 头文件：cvi_vi.h, cvi_comm_vi.h
- 库文件：libvpu.a

【注意】

- PIPE 必须已创建，否则会返回失败。
- 必须先设置通道属性。
- 此接口必须与CVI_VI_GetChnFrame 配对使用。
- 用户必须保证 pstFrameInfo 结构中的信息与获取时一致，否则可能造成释放不成功。

【举例】

无

【相关主题】

- CVI_VI_GetChnFrame
- CVI_VI_SetChnAttr

4.3.39 CVI_VI_SetChnRotation

【描述】

设置 VI 通道旋转的属性。

【语法】

```
CVI_S32 CVI_VI_SetChnRotation(VI_PIPE ViPipe, VI_CHN ViChn, const ROTATION_E
↪enRotation);
```

【参数】

参数名称	描述	输入/输出
ViPipe	VI 物理 PIPE 号。取值范围：[0, VI_MAX_PHY_PIPE_NUM - 1)。	输入
ViChn	VI 物理通道号。取值范围：[0, VI_MAX_PHY_CHN_NUM - 1)。	输入
enRotation	旋转属性。详见ROTATION_E 说明	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，请参见错误码 。

【需求】

- 头文件：cvi_vi.h, cvi_comm_vi.h
- 库文件：libvpu.a

【注意】

- 使用本接口前，需先调用 CVI_VI_CreatePipe，否则提示失败。

- 必需在设置通道属性后才能设置此属性。
- 仅支持 RGB planar，YUV420 Planar 及 YUV400 三种格式的旋转。
- 旋转后通道输出的图像内存大小可能发生变化，但从通道所获取的尺寸仍为原本用户设置值。例如 1920x1080 输入的图像，旋转 90 度后，实际输出为 1080x1920。
- VI_ONLINE_VPSS_ONLINE/ VI_OFFLINE_VPSS_ONLINE 下不支援。

【举例】

无

【相关主题】

无

4.3.40 CVI_VI_GetChnRotation

【描述】

获取 VI 通道旋转的属性。

【语法】

```
CVI_S32 CVI_VI_GetChnRotation(VI_PIPE ViPipe, VI_CHN ViChn, ROTATION_E␣
↪*penRotation);
```

【参数】

参数名称	描述	输入/输出
ViPipe	VI 物理 PIPE 号。取值范围：[0, VI_MAX_PHY_PIPE_NUM - 1)。	输入
ViChn	VI 物理通道号。取值范围：[0, VI_MAX_PHY_CHN_NUM - 1)。	输入
penRotation	旋转属性指针。详见ROTATION_E 说明	输出

【返回值】

返回值	描述
0	成功。
非 0	失败，请参见错误码 。

【需求】

- 头文件：cvi_vi.h, cvi_comm_vi.h
- 库文件：libvpu.a

【注意】

- 使用本接口前，需先调用CVI_VI_CreatePipe (ViPipe)，否则提示失败。
- VI_ONLINE_VPSS_ONLINE/ VI_OFFLINE_VPSS_ONLINE 下不支援。

【举例】

无

【相关主题】

CVI_VI_SetChnRotation

4.3.41 CVI_VI_SetChnLDCAttr

【描述】

设置 VI 通道镜头畸变矫正的属性。

【语法】

```
CVI_S32 CVI_VI_SetChnLDCAttr(VI_PIPE ViPipe, VI_CHN ViChn, const VI_LDC_ATTR_S
↪*pstLDCAttr);
```

【参数】

参数名称	描述	输入/输出
ViPipe	VI 物理 PIPE 号。取值范围：[0, VI_MAX_PHY_PIPE_NUM - 1)。	输入
ViChn	VI 物理通道号。取值范围：[0, VI_MAX_PHY_CHN_NUM - 1)。	输入
pstLDCAttr	镜头畸变矫正属性。	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，请参见错误码 。

【需求】

- 头文件：cvi_vi.h, cvi_comm_vi.h
- 库文件：libvpu.a

【注意】

- 使用本接口前，需先调用 CVI_VI_CreatePipe，否则提示失败。
- 必需在设置通道属性后才能设置此属性。
- VI_ONLINE_VPSS_ONLINE/ VI_OFFLINE_VPSS_ONLINE 下不支援。

【举例】

无

【相关主题】

CVI_VI_GetChnLDCAttr

VI_LDC_ATTR_S

4.3.42 CVI_VI_GetChnLDCAttr

【描述】

获取 VI 通道镜头畸变矫正的属性。

【语法】

```
CVI_S32 CVI_VI_GetChnLDCAttr(VI_PIPE ViPipe, VI_CHN ViChn, const VI_LDC_ATTR_S
↪*pstLDCAttr);
```

【参数】

参数名称	描述	输入/输出
ViPipe	VI 物理 PIPE 号。取值范围：[0, VI_MAX_PHY_PIPE_NUM - 1)。	输入
ViChn	VI 物理通道号。取值范围：[0, VI_MAX_PHY_CHN_NUM - 1)。	输入
pstLDCAttr	镜头畸变矫正属性。	输出

【返回值】

返回值	描述
0	成功。
非 0	失败，请参见错误码 。

【需求】

- 头文件：cvi_vi.h, cvi_comm_vi.h
- 库文件：libvpu.a

【注意】

- 使用本接口前，需先调用 CVI_VI_CreatePipe，否则提示失败。
- VI_ONLINE_VPSS_ONLINE/ VI_OFFLINE_VPSS_ONLINE 下不支援。

【举例】

无

【相关主题】

CVI_VI_SetChnLDCAttr

VI_LDC_ATTR_S

4.3.43 CVI_VI_RegChnFlipMirrorCallBack

【描述】

注册 VI 通道翻转镜像回调函数。

【语法】

```
CVI_S32 CVI_VI_RegChnFlipMirrorCallBack(VI_PIPE ViPipe, VI_DEV ViDev, void *pvData);
```

【参数】

参数名称	描述	输入/输出
ViPipe	PIPE 号。	输入
ViDev	VI 设备号。	输入
pvData	回调函数指针。	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，请参见错误码 。

【需求】

- 头文件：cvi_vi.h, cvi_comm_vi.h
- 库文件：libvpu.a

【注意】

无

【举例】

无

【相关主题】

无

4.3.44 CVI_VI_UnRegChnFlipMirrorCallBack

【描述】

注销 VI 通道翻转镜像回调函数。

【语法】

```
CVI_S32 CVI_VI_UnRegChnFlipMirrorCallBack(VI_PIPE ViPipe, VI_DEV ViDev);
```

【参数】

参数名称	描述	输入/输出
ViPipe	PIPE 号。	输入
ViDev	VI 设备号。	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，请参见错误码 。

【需求】

- 头文件：cvi_vi.h, cvi_comm_vi.h
- 库文件：libvpu.a

【注意】

无

【举例】

无

【相关主题】

无

4.3.45 CVI_VI_SetChnFlipMirror

【描述】

设置 VI 通道翻转镜像的属性。

【语法】

```
CVI_S32 CVI_VI_SetChnFlipMirror(VI_PIPE ViPipe, VI_CHN ViChn, CVI_BOOL bFlip, CVI_
↪BOOL bMirror);
```

【参数】

参数名称	描述	输入/输出
ViPipe	PIPE 号。	输入
ViChn	VI 通道号。	输入
bFlip	翻转使能开关。	输入
bMirror	镜像使能开关。	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，请参见错误码 。

【需求】

- 头文件：cvi_vi.h, cvi_comm_vi.h
- 库文件：libvpu.a

【注意】

- 使用前需先调用 CVI_VI_RegChnFlipMirrorCallBack 注册回调函数，否则提示失败。

【举例】

无

【相关主题】

- CVI_VI_RegChnFlipMirrorCallBack

4.3.46 CVI_VI_GetChnFlipMirror

【描述】

获取 VI 通道翻转镜像的属性。

【语法】

```
CVI_S32 CVI_VI_GetChnFlipMirror(VI_PIPE ViPipe, VI_CHN ViChn, CVI_BOOL *pbFlip, CVI_
↪BOOL *pbMirror);
```

【参数】

参数名称	描述	输入/输出
ViPipe	PIPE 号。	输入
ViChn	VI 通道号。	输入
pbFlip	翻转属性。	输出
pbMirror	镜像属性。	输出

【返回值】

返回值	描述
0	成功。
非 0	失败，请参见错误码 。

【需求】

- 头文件：cvi_vi.h, cvi_comm_vi.h
- 库文件：libvpu.a

【注意】

无

【举例】

无

【相关主题】

无

4.3.47 CVI_VI_Suspend

【描述】

休眠 VI 设备。

【语法】

```
CVI_S32 CVI_VI_Suspend(void);
```

【参数】

无

【返回值】

返回值	描述
0	成功。
非 0	失败，请参见错误码 。

【需求】

- 头文件：cvi_vi.h
- 库文件：libvpu.a

【注意】

无

【举例】

无

【相关主题】

CVI_VI_Resume

4.3.48 CVI_VI_Resume

【描述】

唤醒 VI 设备。

【语法】

```
CVI_S32 CVI_VI_Resume(void);
```

【参数】

无

【返回值】

返回值	描述
0	成功。
非 0	失败，请参见错误码 。

【需求】

- 头文件：cvi_vi.h
- 库文件：libvpu.a

【注意】

无

【举例】

无

【相关主题】

CVI_VI_Suspend

4.3.49 CVI_VI_SetDevNum

【描述】

设置 VI 设备数量。

【语法】

```
CVI_S32 CVI_VI_SetDevNum(CVI_U32 devNum);
```

【参数】

参数名称	描述	输入/输出
devNum	VI 设备数量。	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，请参见错误码。

【需求】

- 头文件：cvi_vi.h, cvi_comm_vi.h
- 库文件：libvi.a

【注意】

无

【举例】

无

【相关主题】

CVI_VI_GetDevNum

4.3.50 CVI_VI_GetDevNum

【描述】

获取 VI 设备数量。

【语法】

```
CVI_S32 CVI_VI_GetDevNum(CVI_U32 devNum);
```

【参数】

参数名称	描述	输入/输出
devNum	VI 设备数量。	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，请参见错误码。

【需求】

- 头文件：cvi_vi.h, cvi_comm_vi.h
- 库文件：libvi.a

【注意】

无

【举例】

无

【相关主题】

CVI_VI_SetDevNum

4.3.51 CVI_VI_EnablePatt

【描述】

启用 VI pattern 模式。

【语法】

```
CVI_S32 CVI_VI_EnablePatt(VI_PIPE ViPipe);
```

【参数】

参数名称	描述	输入/输出
ViPipe	Pipe 号。	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，请参见错误码。

【需求】

- 头文件：cvi_vi.h
- 库文件：libvi.a

【注意】

仅 DUAL OS 支持

【举例】

无

【相关主题】

无

4.3.52 CVI_VI_StartSmoothRawDump

【描述】

开始 dump 平滑 raw 图。

【语法】

```
CVI_S32 CVI_VI_StartSmoothRawDump(const VI_SMOOTH_RAW_DUMP_INFO_S
↪*pstDumpInfo);
```

【参数】

参数名称	描述	输入/输出
pstDumpInfo	vi dump 平滑 raw 图信息结构体指针	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，请参见错误码。

【需求】

- 头文件：cvi_vi.h, cvi_comm_vi.h
- 库文件：libvi.a

【注意】

无

【举例】

无

【相关主题】

CVI_VI_StopSmoothRawDump

4.3.53 CVI_VI_StopSmoothRawDump

【描述】

停止 dump 平滑 raw 图。

【语法】

```
CVI_S32 CVI_VI_StopSmoothRawDump(const VI_SMOOTH_RAW_DUMP_INFO_S
↪*pstDumpInfo);
```

【参数】

参数名称	描述	输入/输出
pstDumpInfo	vi dump 平滑 raw 图信息结构体指针	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，请参见错误码。

【需求】

- 头文件：cvi_vi.h, cvi_comm_vi.h
- 库文件：libvi.a

【注意】

无

【举例】

无

【相关主题】

CVI_VI_StartSmoothRawDump

4.3.54 CVI_VI_GetSmoothRawDump

【描述】

获取 dump 下来的平滑 raw 图。

【语法】

```
CVI_S32 CVI_VI_GetSmoothRawDump(VI_PIPE ViPipe, VIDEO_FRAME_INFO_S␣
↪*pstVideoFrame, CVI_S32 s32MilliSec);
```

【参数】

参数名称	描述	输入/输出
ViPipe	pipe 号	输入
pstVideoFrame	raw 数据信息	输出
s32MilliSec	超时参数，-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，请参见错误码。

【需求】

- 头文件：cvi_vi.h, cvi_comm_vi.h
- 库文件：libvi.a

【注意】

无

【举例】

无

【相关主题】

无

4.3.55 CVI_VI_PutSmoothRawDump

【描述】

放置 dump 下来的平滑 raw 图。

【语法】

```
CVI_S32 CVI_VI_PutSmoothRawDump(VI_PIPE ViPipe, const VIDEO_FRAME_INFO_S␣
↪*pstVideoFrame);
```

【参数】

参数名称	描述	输入/输出
ViPipe	pipe 号	输入
pstVideoFrame	raw 数据信息	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，请参见错误码。

【需求】

- 头文件：cvi_vi.h, cvi_comm_vi.h
- 库文件：libvi.a

【注意】

无

【举例】

无

【相关主题】

无

4.3.56 CVI_VI_GetRgbMapLeBuf

【描述】

获取 rgbmap 长帧缓冲区。

【语法】

```
CVI_S32 CVI_VI_GetRgbMapLeBuf(VI_PIPE ViPipe, void *pstRgbMapBuf);
```

【参数】

参数名称	描述	输入/输出
ViPipe	pipe 号	输入
pstRgbMapBuf	rgbmap 缓冲区指针	输出

【返回值】

返回值	描述
0	成功。
非 0	失败，请参见错误码。

【需求】

- 头文件：cvi_vi.h, cvi_comm_vi.h
- 库文件：libvi.a

【注意】

仅 DUAL OS 支持

【举例】

无

【相关主题】

无

4.3.57 CVI_VI_GetRgbMapSeBuf

【描述】

获取 rgbmap 短帧缓冲区。

【语法】

```
CVI_S32 CVI_VI_GetRgbMapSeBuf(VI_PIPE ViPipe, void *pstRgbMapBuf);
```

【参数】

参数名称	描述	输入/输出
ViPipe	pipe 号	输入
pstRgbMapBuf	rgbmap 缓冲区指针	输出

【返回值】

返回值	描述
0	成功。
非 0	失败，请参见错误码。

【需求】

- 头文件：cvi_vi.h, cvi_comm_vi.h
- 库文件：libvi.a

【注意】

仅 DUAL OS 支持

【举例】

无

【相关主题】

无

4.3.58 CVI_VI_DumpHwRegisterToFile

【描述】

下载硬件寄存器数据到指定文件。

【语法】

```
CVI_S32 CVI_VI_DumpHwRegisterToFile(VI_PIPE ViPipe, FILE *fp, VI_DUMP_REGISTER_
↪TABLE_S *pstRegTbl);
```

【参数】

参数名称	描述	输入/输出
ViPipe	pipe 号	输入
fp	文件描述符	输入
pstRegTbl	硬件寄存器数据指针	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，请参见错误码。

【需求】

- 头文件：cvi_vi.h, cvi_comm_vi.h
- 库文件：libvi.a

【注意】

无

【举例】

无

【相关主题】

无

4.3.59 CVI_VI_QueryChnStatus

【描述】

查看 VI 通道状态。

【语法】

```
CVI_S32 CVI_VI_QueryChnStatus(VI_PIPE ViPipe, VI_CHN ViChn, VI_CHN_STATUS_S␣
↪*pstChnStatus);
```

【参数】

参数名称	描述	输入/输出
ViPipe	pipe 号	输入
ViChn	VI 通道号	输入
pstChnStatus	VI 设备通道状态结构体指针	输出

【返回值】

返回值	描述
0	成功。
非 0	失败，请参见错误码。

【需求】

- 头文件：cvi_vi.h, cvi_comm_vi.h
- 库文件：libvi.a

【注意】

无

【举例】

无

【相关主题】

无

4.3.60 CVI_VI_GetChnFd

【描述】

获取 VI 通道描述符。

【语法】

```
S32 CVI_VI_GetChnFd(VI_PIPE ViPipe, VI_CHN ViChn);
```

【参数】

参数名称	描述	输入/输出
ViPipe	pipe 号	输入
ViChn	VI 通道号	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，请参见错误码。

【需求】

- 头文件：cvi_vi.h, cvi_comm_vi.h
- 库文件：libvi.a

【注意】

仅 DUAL OS 支持

【举例】

无

【相关主题】

无

4.3.61 CVI_VI_SetChnAlign

【描述】

设置 VI 通道对齐数。

【语法】

```
CVI_S32 CVI_VI_SetChnAlign(VI_PIPE ViPipe, VI_CHN ViChn, CVI_U32 u32Align);
```

【参数】

参数名称	描述	输入/输出
ViPipe	pipe 号	输入
ViChn	VI 通道号	输入
u32Align	VI 通道对齐数	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，请参见错误码。

【需求】

- 头文件：cvi_vi.h, cvi_comm_vi.h
- 库文件：libvi.a

【注意】

仅 DUAL OS 支持

【举例】

无

【相关主题】

CVI_VI_GetChnAlign

4.3.62 CVI_VI_GetChnAlign

【描述】

获取 VI 通道对齐数。

【语法】

```
CVI_S32 CVI_VI_GetChnAlign(VI_PIPE ViPipe, VI_CHN ViChn, CVI_U32 *pu32Align);
```

【参数】

参数名称	描述	输入/输出
ViPipe	pipe 号	输入
ViChn	VI 通道号	输入
pu32Align	VI 通道对齐数指针	输出

【返回值】

返回值	描述
0	成功。
非 0	失败，请参见错误码。

【需求】

- 头文件：cvi_vi.h, cvi_comm_vi.h
- 库文件：libvi.a

【注意】

仅 DUAL OS 支持

【举例】

无

【相关主题】

CVI_VI_SetChnAlign

4.3.63 CVI_VI_RegPmCallBack

【描述】

注册 VI 设备电源管理回调函数。

【语法】

```
CVI_S32 CVI_VI_RegPmCallBack(VI_DEV ViDev, VI_PM_OPS_S *pstPmOps, void *pvData);
```

【参数】

参数名称	描述	输入/输出
ViDev	VI 设备号	输入
pstPmOps	VI 设备电源管理操作结构体指针	输入
pvData	用户私有数据指针	输出

【返回值】

返回值	描述
0	成功。
非 0	失败，请参见错误码。

【需求】

- 头文件：cvi_vi.h, cvi_comm_vi.h
- 库文件：libvi.a

【注意】

无

【举例】

无

【相关主题】

CVI_VI_UnRegPmCallBack

4.3.64 CVI_VI_UnRegPmCallBack

【描述】

注销 VI 设备电源管理回调函数。

【语法】

```
CVI_S32 CVI_VI_UnRegPmCallBack(VI_DEV ViDev);
```

【参数】

参数名称	描述	输入/输出
ViDev	VI 设备号	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，请参见错误码。

【需求】

- 头文件：cvi_vi.h, cvi_comm_vi.h
- 库文件：libvi.a

【注意】

无

【举例】

无

【相关主题】

CVI_VI_RegPmCallBack

4.3.65 CVI_VI_SetTuningDis

【描述】

设置 tuning 参数。

【语法】

```
CVI_S32 CVI_VI_SetTuningDis(CVI_S32 ctrl, CVI_S32 fe, CVI_S32 be, CVI_S32 post);
```

【参数】

参数名称	描述	输入/输出
ctrl	控制通道参数	输入
fe	fe 控制参数	输入 be
be	be 控制参数	输入
post	post 控制参数	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，请参见错误码。

【需求】

- 头文件：cvi_vi.h, cvi_comm_vi.h
- 库文件：libvi.a

【注意】

仅 DUAL OS 支持

【举例】

无

【相关主题】

无

4.4 数据类型

视频输入相关数据类型定义如下，处理器差异的部分参见各处理器支持能力 。

- VI_MAX_DEV_NUM：定义 VI 设备的最大个数。
- VI_MAX_PHY_PIPE_NUM：定义 VI 物理 PIPE 的最大个数。
- VI_MAX_VIR_PIPE_NUM：定义 VI 虚拟 PIPE 的最大个数。
- VI_MAX_PIPE_NUM：定义 VI PIPE 的最大个数。
- VI_MAX_PHY_CHN_NUM：定义 VI 物理通道的最大个数。
- VI_MAX_CHN_NUM：定义 VI 物理信道和扩展信道的总个数。
- VI_DEV_MIN_WIDTH：VI 设备捕获图像的最小宽度。
- VI_DEV_MIN_HEIGHT：VI 设备捕获图像的最小高度。
- VI_DEV_MAX_WIDTH：VI 设备捕获图像的最大宽度。
- VI_DEV_MAX_HEIGHT：VI 设备捕获图像的最大高度。
- VI_PIPE_OFFLINE_MIN_WIDTH：VI PIPE 离线处理图像的最小宽度。
- VI_PIPE_OFFLINE_MIN_HEIGHT：VI PIPE 离线处理图像的最小高度。
- VI_PIPE_OFFLINE_MAX_WIDTH：VI PIPE 离线处理图像的最大宽度。
- VI_PIPE_OFFLINE_MAX_HEIGHT：VI PIPE 离线处理图像的最大高度。
- VI_PIPE_ONLINE_MIN_WIDTH：VI PIPE 在线处理图像的最小宽度。
- VI_PIPE_ONLINE_MIN_HEIGHT：VI PIPE 在线处理图像的最小高度。
- VI_PIPE_ONLINE_MAX_WIDTH：VI PIPE 在线处理图像的最大宽度。
- VI_PIPE_ONLINE_MAX_HEIGHT：VI PIPE 在线处理图像的最大高度。
- VI_PIPE0_MAX_WIDTH：VI PIPE0 处理图像的最大宽度。
- VI_PIPE0_MAX_HEIGHT：VI PIPE0 处理图像的最大高度。
- VI_PIPE1_MAX_WIDTH：VI PIPE1 处理图像的最大宽度。
- VI_PIPE1_MAX_HEIGHT：VI PIPE1 处理图像的最大高度。
- VI_PIPE2_MAX_WIDTH：VI PIPE2 处理图像的最大宽度。
- VI_PIPE2_MAX_HEIGHT：VI PIPE2 处理图像的最大高度。
- VI_PIPE3_MAX_WIDTH：VI PIPE3 处理图像的最大宽度。
- VI_PIPE3_MAX_HEIGHT：VI PIPE3 处理图像的最大高度。
- VI_DATA_TYPE_E：VI 输入数据类型枚举。
- VI_DEV_ATTR_S：定义视频输入设备的属性。

- VI_DEV_BIND_PIPE_S：定义 VI DEV 与 PIPE 的绑定关系。
- VI_PIPE_ATTR_S：定义 VI PIPE 属性。
- VI_DUMP_TYPE_E：枚举 dump 类型。
- VI_DUMP_ATTR_S：定义 VI PIPE dump 属性。
- VI_CHN_ATTR_S：定义 VI 通道属性。
- VI_CROP_INFO_S：定义 VI CROP 信息结构体。
- VI_DEV_TIMING_ATTR_S：自产生时序属性。
- VI_PIPE_STATUS_S：定义 VI PIPE 状态信息。
- VI_CHN_STATUS_S：定义 VI 通道状态信息。
- VI_PIPE_FRAME_SOURCE_E：定义 VI PIPE 数据的来源类型。
- VI_LDC_ATTR_S：定义 VI 镜头畸变矫正结构体。
- ROTATION_E：定义旋转角度。

4.4.1 VI_MAX_DEV_NUM

【说明】

定义 VI 设备的最大个数。

【定义】

```
#define VI_MAX_DEV_NUM 3
```

【注意事项】

无。

【相关数据类型及接口】

无。

4.4.2 VI_MAX_PHY_PIPE_NUM

【说明】

定义 VI 物理 PIPE 的最大个数。

【定义】

```
#define VI_MAX_PHY_PIPE_NUM 5
```

【注意事项】

无。

【相关数据类型及接口】

无。

4.4.3 VI_MAX_VIR_PIPE_NUM

【说明】

定义 VI 虚拟 PIPE 的最大个数。

【定义】

```
#define VI_MAX_VIR_PIPE_NUM 0
```

【注意事项】

无。

【相关数据类型及接口】

无。

4.4.4 VI_MAX_PIPE_NUM

【说明】

定义 VI PIPE 的最大个数。

【定义】

```
#define VI_MAX_PIPE_NUM (VI_MAX_PHY_PIPE_NUM + VI_MAX_VIR_PIPE_NUM)
```

【注意事项】

无。

【相关数据类型及接口】

无。

4.4.5 VI_MAX_PHY_CHN_NUM

【说明】

定义 VI 物理通道的最大个数。

【定义】

```
#define VI_MAX_PHY_CHN_NUM 3
```

【注意事项】

无。

【相关数据类型及接口】

无。

4.4.6 VI_MAX_CHN_NUM

【说明】

定义 VI 物理信道和扩展信道的总个数。

【定义】

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

【注意事项】

无。

【相关数据类型及接口】

无。

4.4.7 VI_DEV_MIN_WIDTH

【说明】

VI 设备捕获图像的最小宽度。

【定义】

```
#define VI_DEV_MIN_WIDTH 120
```

【注意事项】

无。

【相关数据类型及接口】

无。

4.4.8 VI_DEV_MIN_HEIGHT

【说明】

VI 设备捕获图像的最小高度。

【定义】

```
#define VI_DEV_MIN_HEIGHT 120
```

【注意事项】

无。

【相关数据类型及接口】

无。

4.4.9 VI_DEV_MAX_WIDTH

【说明】

VI 设备捕获图像的最大宽度。

【定义】

```
#define VI_DEV_MIN_WIDTH 4608
```

【注意事项】

无。

【相关数据类型及接口】

无。

4.4.10 VI_DEV_MAX_HEIGHT

【说明】

VI 设备捕获图像的最大高度。

【定义】

```
#define VI_DEV_MAX_HEIGHT 4608
```

【注意事项】

无。

【相关数据类型及接口】

无。

4.4.11 VI_PIPE_OFFLINE_MIN_WIDTH

【说明】

VI PIPE 离线处理图像的最小宽度。

【定义】

```
#define VI_PIPE_OFFLINE_MIN_WIDTH 120
```

【注意事项】

无。

【相关数据类型及接口】

无。

4.4.12 VI_PIPE_OFFLINE_MIN_HEIGHT

【说明】

VI PIPE 离线处理图像的最小高度。

【定义】

```
#define VI_PIPE_OFFLINE_MIN_HEIGHT 120
```

【注意事项】

无。

【相关数据类型及接口】

无。

4.4.13 VI_PIPE_OFFLINE_MAX_WIDTH

【说明】

VI PIPE 离线处理图像的最大宽度。

【定义】

```
#define VI_PIPE_OFFLINE_MAX_WIDTH 2880
```

【注意事项】

无。

【相关数据类型及接口】

无。

4.4.14 VI_PIPE_OFFLINE_MAX_HEIGHT

【说明】

VI PIPE 离线处理图像的最大高度。

【定义】

```
#define VI_PIPE_OFFLINE_MAX_HEIGHT 1944
```

【注意事项】

无。

【相关数据类型及接口】

无。

4.4.15 VI_PIPE_ONLINE_MIN_WIDTH

【说明】

VI PIPE 在线处理图像的最小宽度。

【定义】

```
#define VI_PIPE_ONLINE_MIN_WIDTH 120
```

【注意事项】

无。

【相关数据类型及接口】

无。

4.4.16 VI_PIPE_ONLINE_MIN_HEIGHT

【说明】

VI PIPE 在线处理图像的最小高度。

【定义】

```
#define VI_PIPE_ONLINE_MIN_HEIGHT 120
```

【注意事项】

无。

【相关数据类型及接口】

无。

4.4.17 VI_PIPE_ONLINE_MAX_WIDTH

【说明】

VI PIPE 在线处理图像的最大宽度。

【定义】

```
#define VI_PIPE_ONLINE_MAX_WIDTH 2880
```

【注意事项】

无。

【相关数据类型及接口】

无。

4.4.18 VI_PIPE_ONLINE_MAX_HEIGHT

【说明】

VI PIPE 在线处理图像的最大高度。

【定义】

```
#define VI_PIPE_ONLINE_MAX_HEIGHT 1944
```

【注意事项】

无。

【相关数据类型及接口】

无。

4.4.19 VI_PIPE0_MAX_WIDTH

【说明】

VI PIPE0 处理图像的最大宽度。

【定义】

```
#define VI_PIPE0_MAX_WIDTH 2880
```

【注意事项】

无。

【相关数据类型及接口】

无。

4.4.20 VI_PIPE0_MAX_HEIGHT

【说明】

VI PIPE0 处理图像的最大高度。

【定义】

```
#define VI_PIPE0_MAX_HEIGHT 1944
```

【注意事项】

无。

【相关数据类型及接口】

无。

4.4.21 VI_PIPE1_MAX_WIDTH

【说明】

VI PIPE1 处理图像的最大宽度。

【定义】

```
#define VI_PIPE1_MAX_WIDTH 2880
```

【注意事项】

无。

【相关数据类型及接口】

无。

4.4.22 VI_PIPE1_MAX_HEIGHT

【说明】

VI PIPE1 处理图像的最大高度。

【定义】

```
#define VI_PIPE1_MAX_HEIGHT 1944
```

【注意事项】

无。

【相关数据类型及接口】

无。

4.4.23 VI_PIPE2_MAX_WIDTH

【说明】

VI PIPE2 处理图像的最大宽度。

【定义】

```
#define VI_PIPE2_MAX_WIDTH 2880
```

【注意事项】

无。

【相关数据类型及接口】

无。

4.4.24 VI_PIPE2_MAX_HEIGHT

【说明】

VI PIPE2 处理图像的最大高度。

【定义】

```
#define VI_PIPE2_MAX_HEIGHT 1944
```

【注意事项】

无。

【相关数据类型及接口】

无。

4.4.25 VI_PIPE3_MAX_WIDTH

【说明】

VI PIPE3 处理图像的最大宽度。

【定义】

```
#define VI_PIPE3_MAX_WIDTH 2880
```

【注意事项】

无。

【相关数据类型及接口】

无。

4.4.26 VI_PIPE3_MAX_HEIGHT

【说明】

VI PIPE3 处理图像的最大高度。

【定义】

```
#define VI_PIPE3_MAX_HEIGHT 1944
```

【注意事项】

无。

【相关数据类型及接口】

无。

4.4.27 VI_PIPE4_MAX_WIDTH

【说明】

VI PIPE4 处理图像的最大宽度。

【定义】

```
#define VI_PIPE4_MAX_WIDTH 1920
```

【注意事项】

无。

【相关数据类型及接口】

无。

4.4.28 VI_PIPE4_MAX_HEIGHT

【说明】

VI PIPE4 处理图像的最大高度。

【定义】

```
#define VI_PIPE4_MAX_HEIGHT 1080
```

【注意事项】

无。

【相关数据类型及接口】

无。

4.4.29 VI_DATA_TYPE_E

【说明】

VI 输入数据类型枚举。

【定义】

```
typedef enum _VI_DATA_TYPE_E {
  VI_DATA_TYPE_YUV = 0,
  VI_DATA_TYPE_RGB,
  VI_DATA_TYPE_BUTT
} VI_DATA_TYPE_E;
```

【注意事项】

无。

【相关数据类型及接口】

无。

4.4.30 VI_DEV_ATTR_S

【说明】

定义 VI 设备的属性。

【定义】

```
typedef struct _VI_DEV_ATTR_S {
  VI_INTF_MODE_E enIntfMode;
  VI_WORK_MODE_E enWorkMode;
  VI_SCAN_MODE_E enScanMode;
  CVI_S32 as32AdChnId[VI_MAX_ADCHN_NUM];
  VI_YUV_DATA_SEQ_E enDataSeq;
  VI_SYNC_CFG_S stSynCfg;
  VI_DATA_TYPE_E enInputDataType;
  SIZE_S stSize;
  VI_WDR_ATTR_S stWDRAttr;
  BAYER_FORMAT_E enBayerFormat;
  CVI_U32 fbmEnable;
  CVI_U32 chn_num;
  CVI_U32 snrFps;
  CVI_U64 phy_addr;
  CVI_U32 phy_size;
  CVI_BOOL enVcWdrMode;
} VI_DEV_ATTR_S;
```

【成员】

成员名称	描述
enIntfMode	Sensor 接口模式。
enWorkMode	1、2、4 路复合工作模式。
enScanMode	输入扫描模式 (逐行、隔行)。
as32AdChnId[VI_MAX_ADCHN_NUM]	取值范围 [-1,3]，推荐统一设置为默认值-1，此参数无意义。
enDataSeq	输入数据顺序 (仅支持 yuv 格式)。
stSynCfg	同步时序配置，BT.601 模式时必须配置，其它模式时无效。
enInputDataType	输入数据类型，Sensor 输入一般为 RGB，AD 输入一般为 YUV。
stSize	VI 设备可设置要捕获图像的高宽，捕获图像的最小宽高与最大宽高： 宽 度：[VI_DEV_MIN_WIDTH, VI_DEV_MAX_WIDTH]。 高 度：[VI_DEV_MIN_HEIGHT, VI_DEV_MAX_HEIGHT]。
stWDRAttr	WDR 属性。
enBayerFormat	设备的 bayer format，当 inputDataType 为 RGB 必须设置此属性。
fbmEnable	仅 ALIOS/DUAL OS 支持，frame buffer mode: 1, slice buffer mode: 0.
chn_num	设备的通道数。
snrFps	设备的初始帧率。
phy_addr	仅 ALIOS/DUAL OS 支持，若不为 0，VI 使用用户分配的物理地址，使用前需保证物理地址的合法性.
phy_size	仅 ALIOS/DUAL OS 支持，所使用物理地址内存大小.
enVcWdrMode	仅 ALIOS/DUAL OS 支持，Sensor 使用 VC WDR mode: 1，其他: 0.

【注意事项】

- stSize 中 u32Width 必须等于实际输入图像的宽度，u32Height 必须等于实际输入图像的高度，否则会导致没有图像输出。

【相关数据类型及接口】

无。

4.4.31 VI_DEV_BIND_PIPE_S

【说明】

定义 VI DEV 与 PIPE 的绑定关系。

【定义】

```
typedef struct _VI_DEV_BIND_PIPE_S {
  CVI_U32 u32Num; /* RW;Range [1,VI_MAX_PIPE_NUM] */
  VI_PIPE PipeId[VI_MAX_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】

无。

【相关数据类型及接口】

无。

4.4.32 VI_PIPE_ATTR_S

【说明】

定义 VI PIPE 属性。

【定义】

```
typedef struct _VI_PIPE_ATTR_S {
  VI_PIPE_BYPASS_MODE_E enPipeBypassMode;
  CVI_BOOL bYuvSkip; /* RW;YUV skip enable */
  CVI_BOOL bIspBypass; /* RW;ISP bypass enable */
  CVI_U32 u32MaxW; /* RW;Range[VI_PIPE_MIN_WIDTH,VI_PIPE_MAX_WIDTH];Maximum␣
↪width */
  CVI_U32 u32MaxH; /* RW;Range[VI_PIPE_MIN_HEIGHT,VI_PIPE_MAX_HEIGHT];
↪Maximum height */
  PIXEL_FORMAT_E enPixFmt; /* RW;Pixel format */
  COMPRESS_MODE_E enCompressMode; /* RW;Compress mode.*/
  DATA_BITWIDTH_E enBitWidth; /* RW;Bit width*/
  CVI_BOOL bNrEn; /* RW;3DNR enable */
  CVI_BOOL bSharpenEn; /* RW;Sharpen enable*/
  FRAME_RATE_CTRL_S stFrameRate; /* RW;Frame rate */
  CVI_BOOL bDiscardProPic;
  CVI_BOOL bYuvBypassPath;
} VI_PIPE_ATTR_S;
```

【成员】

成员名称	描述
enPipeBypassMode	VI PIPE 的 Bypass 模式。
bYuvSkip	是否关闭下采样和 CSC。 CVI_FALSE：yuv skip 不使能。 CVI_TRUE：yuv skip 使能。
bIspBypass	ISP 是否 bypass。 CVI_FALSE：ISP 正常运行。 CVI_TRUE：ISP bypass，不运行 ISP。
u32MaxW	输入图像宽度。静态属性，创建 PIPE 时设定，不可更改。 在线模式下取值范围：[VI_PIPE_ONLINE_MIN_WIDTH, VI_PIPE_ONLINE_MAX_WIDTH]。 离线模式下取值范围：[VI_PIPE_OFFLINE_MIN_WIDTH, VI_PIPE_OFFLINE_MAX_WIDTH]。
u32MaxH	输入图像高度。 在线模式下取值范围：[VI_PIPE_ONLINE_MIN_HEIGHT, VI_PIPE_ONLINE_MAX_HEIGHT]。 离线模式下取值范围：[VI_PIPE_OFFLINE_MIN_HEIGHT, VI_PIPE_OFFLINE_MAX_HEIGHT]。
enPixFmt	像素格式。
enCompressMode	数据压缩格式。
enBitWidth	输入图像的 bit 位宽。创建 PIPE 时设定，不可更改，仅当像素格式 enPixFmt 为 YUV 像素格式时有效。
bNrEn	NR 使能开关。 CVI_FALSE：不使能。 CVI_TRUE：使能。
bSharpenEn	Sharpen 使能开关。
stFrameRate	帧率控制。
bDiscardProPic	是否丢弃长曝光的帧。
bYuvBypassPath	是否 YUV 直通模式。

【注意事项】

无。

【相关数据类型及接口】

无。

4.4.33 VI_DUMP_TYPE_E

【说明】

枚举 dump 类型。

【定义】

```
typedef enum _VI_DUMP_TYPE_E {
  VI_DUMP_TYPE_RAW = 0,
  VI_DUMP_TYPE_YUV = 1,
  VI_DUMP_TYPE_IR = 2,
  VI_DUMP_TYPE_BUTT
} VI_DUMP_TYPE_E;
```

【成员】

成员名称	描述
VI_DUMP_TYPE_RAW	Dump RAW 数据。
VI_DUMP_TYPE_YUV	Dump YUV 数据。
VI_DUMP_TYPE_IR	Dump IR 分量数据。

【注意事项】

- 不支持 Dump IR 数据
- Dump YUV 请使用 CVI_VI_GetChnFrame

【相关数据类型及接口】

无。

4.4.34 VI_DUMP_ATTR_S

【说明】

定义 PIPE dump 属性。

【定义】

```
typedef struct _VI_DUMP_ATTR_S {
  CVI_BOOL bEnable; /* RW;Whether dump is enable */
  CVI_U32 u32Depth; /* RW;Range [0,8];Depth */
  VI_DUMP_TYPE_E enDumpType;
} VI_DUMP_ATTR_S;
```

【成员】

成员名称	描述
bEnable	是否使能 dump。
u32Depth	Dump 数据的队列深度。取值范围：[0, 8]。
enDumpType	Dump 数据类型。

【注意事项】

无。

【相关数据类型及接口】

- CVI_VI_SetPipeDumpAttr
- CVI_VI_GetPipeDumpAttr

4.4.35 VI_CHN_ATTR_S

【说明】

定义 VI 通道属性。

【定义】

```
typedef struct _VI_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. */
  CVI_BOOL bMirror; /* RW;Mirror enable */
  CVI_BOOL bFlip; /* RW;Flip enable */
  CVI_U32 u32Depth; /* RW;Range [0,8];Depth */
  FRAME_RATE_CTRL_S stFrameRate; /* RW;Frame rate */
  CVI_BOOL bLVDSflow;
  CVI_U8   u8TotalChnNum;
} VI_CHN_ATTR_S;
```

【成员】

成员名称	描述
stSize	目标图像大小。 目标图像的最小宽高与最大宽高： 在线模式下高度[VI_PHYCHN_ONLINE_MIN_HEIGHT, VI_PHYCHN_ONLINE_MAX_HEIGHT]。 离线模式下高度[VI_PHYCHN_OFFLINE_MIN_HEIGHT, VI_PHYCHN_OFFLINE_MAX_HEIGHT]。 在线模式下宽度[VI_PHYCHN_ONLINE_MIN_WIDTH, VI_PHYCHN_ONLINE_MAX_WIDTH]。 离线模式下宽度[VI_PHYCHN_OFFLINE_MIN_WIDTH, VI_PHYCHN_OFFLINE_MAX_WIDTH]。
enPixelFormat	目标图像像素格式。
enDynamicRange	目标图像动态范围。
enVideoFormat	目标图像视频数据格式。
enCompressMode	目标图像压缩格式。
bMirror	Mirror 使能开关。 CVI_FALSE：不使能。 CVI_TRUE：使能。
bFlip	Flip 使能开关。 CVI_FALSE：不使能。 CVI_TRUE：使能。
u32Depth	用户获取图像的队列深度。
stFrameRate	帧率控制。源帧率取值范围：(0, 240]，以及-1。目标帧率取值范围：[-1, 240]。当源帧率为-1时，目标帧率必须为-1(不进行帧率控制)，其他情况下，目标帧率不能大于源帧率。
bLVDSflow	是否 LVDS 模式。
u8TotalChnNum	通道总数量。

【注意事项】

- 目前 VI 不支持帧率控制

【相关数据类型及接口】

- CVI_VI_SetChnAttr
- CVI_VI_GetChnAttr

4.4.36 VI_CROP_INFO_S

【说明】

定义 VI CROP 信息结构体。

【定义】

```
typedef struct _VI_CROP_INFO_S {
  CVI_BOOL bEnable; /* RW;CROP enable*/
  VI_CROP_COORDINATE_E enCropCoordinate; /* RW;Coordinate mode of the crop start point*/
  RECT_S stCropRect; /* RW;CROP rectangular*/
} VI_CROP_INFO_S;
```

【成员】

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

【注意事项】

无。

【相关数据类型及接口】

无。

4.4.37 VI_DEV_TIMING_ATTR_S

【说明】

用户自定义时序属性。

【定义】

```
typedef struct _VI_DEV_TIMING_ATTR_S {
  CVI_BOOL bEnable; /* RW;Whether enable VI generate timing */
  CVI_S32 s32FrmRate; /* RW;Generate timing Frame rate*/
} VI_DEV_TIMING_ATTR_S;
```

【成员】

成员名称	描述
bEnable	用户自定义时序使能开关。
s32FrmRate	用户自定义时序的帧率。

【注意事项】

- 当使能用户自定义时序后，若用户设置的帧率超过设备最大帧率时，系统自动以设备最大帧率为有效值。

【相关数据类型及接口】

- CVI_VI_SetDevTimingAttr

4.4.38 VI_PIPE_STATUS_S

【说明】

定义 VI PIPE 的状态信息。

【定义】

```
typedef struct _VI_PIPE_STATUS_S {
  CVI_BOOL bEnable; /* RO;Whether this pipe is enabled */
  CVI_U32 u32IntCnt; /* RO;The video frame interrupt count */
  CVI_U32 u32FrameRate; /* RO;Current frame rate */
  CVI_U32 u32LostFrame; /* RO;Lost frame count */
  CVI_U32 u32VbFail; /* RO;Video buffer malloc failure */
  SIZE_S stSize; /* RO;Current pipe output size */
} VI_PIPE_STATUS_S;
```

【成员】

成员名称	描述
bEnable	当前 PIPE 是否使能。
u32IntCnt	中断计数。
u32FrameRate	VI PIPE 的实时帧率。
u32LostFrame	丢帧计数。
u32VbFail	VB 申请失败计数。
stSize	PIPE 当前图像大小。

【注意事项】

无。

【相关数据类型及接口】

无。

4.4.39 VI_CHN_STATUS_S

【说明】

定义 VI 通道的状态信息。

【定义】

```
typedef struct _VI_CHN_STATUS_S {
  CVI_BOOL bEnable; /* RO;Whether this channel is enabled */
  CVI_U32 u32FrameRate; /* RO;current frame rate */
  CVI_U64 u64PrevTime; // latest time (us)
  CVI_U32 u32FrameNum;  //The number of Frame in one second
  CVI_U32 u32LostFrame; /* RO;Lost frame count */
  CVI_U32 u32VbFail; /* RO;Video buffer malloc failure */
  CVI_U32 u32IntCnt; /* RO;Receive frame int count */
  CVI_U32 u32RecvPic; /* RO;Receive frame count */
  CVI_U32 u32TotalMemByte; /* RO;VI buffer malloc failure */
  SIZE_S stSize; /* RO;chn output size */
} VI_CHN_STATUS_S;
```

【成员】

成员名称	描述
bEnable	当前 PIPE 是否使能。
u32FrameRate	Chn 的实时帧率。
u64PrevTime	记录前次时间，用于计算 u32FrameNum。
u32FrameNum	一秒时间内的帧数。
u32LostFrame	丢帧计数。
u32VbFail	VB 申请失败计数。
u32IntCnt	Chn 接收的帧数。
u32RecvPic	Chn 接收的帧数。
u32TotalMemByte	暂无使用。
stSize	Chn 当前图像大小。

【注意事项】

无。

【相关数据类型及接口】

无。

4.4.40 VI_PIPE_FRAME_SOURCE_E

【说明】

定义 VI PIPE 数据的来源类型。

【定义】

```
typedef enum _VI_PIPE_FRAME_SOURCE_E
{
  VI_PIPE_FRAME_SOURCE_DEV = 0, /* RW;Source from dev */
  VI_PIPE_FRAME_SOURCE_USER_FE, /* RW;User send to FE */
  VI_PIPE_FRAME_SOURCE_USER_BE, /* RW;User send to BE */
  VI_PIPE_FRAME_SOURCE_BUTT
} VI_PIPE_FRAME_SOURCE_E;
```

【成员】

成员名称	描述
VI_PIPE_FRAME_SOURCE_DEV	数据来自设备。
VI_PIPE_FRAME_SOURCE_USER_FE	数据来自用户从 FE 送进来的数据。
VI_PIPE_FRAME_SOURCE_USER_BE	数据来自用户从 BE 送进来的数据。

【注意事项】

- 不支持从 BE 进数据。

【相关数据类型及接口】

- CVI_VI_SetPipeFrameSource

4.4.41 VI_LDC_ATTR_S

【说明】

定义 VI 镜头畸变矫正结构体

【定义】

```
typedef struct _VI_LDC_ATTR_S {
  CVI_BOOL bEnable;
  LDC_ATTR_S stAttr;
} VI_LDC_ATTR_S;
```

【成员】

成员名称	描述
bEnable	LDC 使能
stAttr	LDC 设定属性

【注意事项】

无。

【相关数据类型及接口】

- LDC_ATTR_S
- CVI_VI_GetChnLDCAttr
- CVI_VI_SetChnLDCAttr

4.4.42 ROTATION_E

【说明】

定义视频旋转角度。

【定义】

```
/*Angle of rotation*/
typedef enum _ROTATION_E {
    ROTATION_0 = 0,
    ROTATION_90,
    ROTATION_180,
    ROTATION_270,
    ROTATION_XY_FLIP,
    ROTATION_MAX
} ROTATION_E;
```

【成员】

各旋转角度

【注意事项】

无

【相关数据类型及接口】

无

4.5 错误码

视频输入 API 错误码如下表所示：

表 4.3: 视频输入 API 错误码

错误代码	宏定义	描述
0xC00E8001	CVI_ERR_VI_INVALID_DEVID	设备号无效
0xC00E8002	CVI_ERR_VI_INVALID_CHNID	通道号无效
0xC00E8003	CVI_ERR_VI_INVALID_PARA	参数设置无效
0xC00E8004	CVI_ERR_VI_PIPE_EXIST	PIPE 已存在
0xC00E8005	CVI_ERR_VI_PIPE_UNEXIST	PIPE 不存在
0xC00E8006	CVI_ERR_VI_INVALID_NULL_PTR	参数空指针错误
0xC00E8007	CVI_ERR_VI_FAILED_NOTCONFIG	设备或信道属性未配置
0xC00E8008	CVI_ERR_VI_NOT_SUPPORT	操作不支持
0xC00E8009	CVI_ERR_VI_NOT_PERM	操作禁止
0xC00E800A	CVI_ERR_VI_INVALID_PIPEID	PIPE 号无效
0xC00E800C	CVI_ERR_VI_NOMEM	内存分配错误
0xC00E800E	CVI_ERR_VI_BUF_EMPTY	缓存为空
0xC00E800F	CVI_ERR_VI_BUF_FULL	缓存为满
0xC00E8010	CVI_ERR_VI_SYS_NOTREADY	系统未初始化
0xC00E8012	CVI_ERR_VI_BUSY	系统忙碌
0xC00E8040	CVI_ERR_VI_FAILED_NOTENABLE	尚未启动设备
0xC00E8041	CVI_ERR_VI_FAILED_NOTDISABLE	尚未关闭设备
0xC00E8042	CVI_ERR_VI_FAILED_CHNOTDISABLE	尚未关闭通道
0xC00E8043	CVI_ERR_VI_CFG_TIMEOUT	设置逾时
0xC00E8044	CVI_ERR_VI_NORM_UNMATCH	找无匹配号
0xC00E8045	CVI_ERR_VI_INVALID_WAYID	无效的通道号
0xC00E8046	CVI_ERR_VI_INVALID_PHYCHNID	无效的实体信道号
0xC00E8047	CVI_ERR_VI_FAILED_NOTBIND	通道未绑定
0xC00E8048	CVI_ERR_VI_FAILED_BINDED	通道绑定失败

5 视频输出

5.1 功能概述

CV180x 不支持该功能。

5.1.1 目的

VO(Video Output 视频输出) 模块从内存读取视频和图形数据，并通过相应的显示设备输出视频和图形。

5.1.2 定义及缩写

DUD（Device Ultra-High Definition 超高清设备）

DHD（Device High Definition 高清设备）

DSD（Device Standard Definition 标清设备）

VUD（Video Layer of UHD 超高清设备影像层）

VHD（Video Layer of HD 高清设备影像层）

VSD（Video Layer of SD 标清设备影像层）

G0（Graphics Layer 图形层 0）

5.2 设计概述

5.2.1 系统架构

- 显示设备

  SDK 中将超高清和高清和标清显示设备分别标示为 DUDx（3840x2160）和 DHDx（1920x1080）和 DSDx（720x576），用以表示设备的能力。其中，x 为索引号，从 0 开始取值，以此来区别第几路显示设备。

例如第 0 路高清设备标示为 DHD0，第 1 路超高清显示设备标示为 DUD1 。

超高清和高清和标清显示设备又可分别简称为 UD, HD 和 SD 设备。

CV181x 有一个高清显示设备 DHD0

- 视频层

  对于固定在每个显示设备上面对应的视频层，SDK 也采取对应显示设备的方式，以 VUDx 和 VHDx 和 VSDx 来标示。

  处理器支持显示设备，视频层和图形层的情况请参见表 5.1。

  处理器设备功能对比参考表 5.2 。

  处理器视频层功能对比如表 5.3 所示。

  设备上视频输出接口支持的最大时序见表 5.4 所示。视频层和显示设备的实际显示分辨率依赖于具体输出接口。

表 5.1: 支持显示设备，视频层和图形层

处理器名称	**子项**	**高清显示备 DHD0**	
CV183x	名称	视频层 VHD0	图形层 G0
	说明	最多支持 1 画面	
	输出接口	BT.1120/BT.656, MIPI Tx, LCD, I80	
CV182x CV181x	名称	视频层 VHD0	图形层 G0
	说明	最多支持 1 画面	
	输出接口	BT.1120/BT.656, MIPI Tx, LCD, I80, RGB	

表 5.2: 显示设备功能

处理器		**最大输出时序**	**输出接口**	**叠加拼接显示**
CV183x	DHD0	1080p@60	BT.1120/BT.656/ MIPI Tx/LCD/I80	不支持
CV182x CV181x	DHD0	1080p@60 720p@60	MIPI Tx BT.1120/BT.656/ LCD/I80/RGB	不支持

表 5.3: 视频层功能

处理器	动态绑定能力	缩放能力	支持通道数		CSC
			SINGLE 模式	MUI 模式	
CV183x	不支持	不支持	1	1	支持
CV182x CV181x	不支持	不支持	1	1	支持

表 5.4: 显示设备接口最大输出时序

处理器		大输出时序
CV183x	MIPI Tx	1080P@60
	BT.1120	1080P@60
	BT.656	1080P@30
	LCD	720P@60
CV182x CV181x	MIPI Tx	1080P@60
	BT.1120	720P@60
	BT.656	720P@30
	LCD	720P@60

- 通道

  SDK 将信道归属于视频层管理，一个视频层上可显示多个视频。每一个视频显示区域称为一个通道，视频被限制信道内，信道被限制在视频层内。对于一个视频层，其上面的通道都是独立的。同时，不同的视频层上的通道也是独立的。

- 分辨率

  分辨率主要有以下 2 种概念:

  F 设备分辨率由设备时序决定，决定该设备的输出有效像素点数。

  F 显示分辨率指在显示设备上的有效显示区域，由视频层属性中的 stDispRect 成员决定。显示分辨率小于等于设备分辨率。

- 图形层绑定

  图形层绑定是指处理器支持将特定的图形层绑定到视频层上。

  CV181x 支持 1 个图形层（G0 固定绑定到 DHD0 上，即 G0 只能与 VHD0 叠加。

- 旋转

  VO 支持对进入信道的图像做旋转的操作。也就可以先做旋转，在进入通道做后续处理。

通常作用在一些竖屏上，此时设置为旋转 90 或 270 度，可以达到满屏并保持图像比例。

- 输入和输出数据格式

  处理器 VO 支持的输入数据格式见下表 5.5 。

  表中 PIXEL FORMAT 、VIDEO FORMAT 项分别列出了处理器支持的输入像素格式、视频格式.

  表中数据格式可参考“2 系统控制”章节。

表 5.5: 处理器支持的输入数据格式

处理器		输入
CV183x	PIXEL FORMAT	YUV PLANAR 444 YUV PLANAR 422 YUV PLANAR 420 YUV 400 RGB PLANAR 888 BGR PLANAR 888 RGB Packed 888 BGR Packed 888
	VIDEO FORMAT	LINEAR
CV182xCV181x	PIXEL FORMAT	YUV PLANAR 444 YUV PLANAR 422 YUV PLANAR 420 YUV 400 NV12 NV21 NV16 NV61 YUYV YVYU UYVY VYUY RGB PLANAR 888 BGR PLANAR 888 RGB Packed 888 BGR Packed 888
	VIDEO FORMAT	LINEAR

5.3 API 参考

视频输出（VO）实现视频输出设备，视频曾，信道等功能。

该功能模块所支持的 API，以下由大而小分别从设备、视频层、通道等方面进行介绍

设备相关 API 如下：

- CVI_VO_Enable： 启用视频输出设备。
- CVI_VO_Disable： 禁用视频输出设备。
- CVI_VO_SetPubAttr： 设置视频输出设备的相关属性。
- CVI_VO_GetPubAttr： 获取视频输出设备的相关属性。
- CVI_VO_CloseFd： 关闭视频输出设备的文件句柄。

视频层相关 API 如下：

- CVI_VO_EnableVideoLayer： 启用视频层。
- CVI_VO_DisableVideoLayer： 禁用视频层。
- CVI_VO_SetVideoLayerAttr： 设置视频层的相关属性。
- CVI_VO_GetVideoLayerAttr： 获取视频层的相关属性。

通道相关 API 如下：

- CVI_VO_EnableChn： 启用视频输出通道。
- CVI_VO_DisableChn： 禁用视频输出通道。
- CVI_VO_SetChnAttr： 设置视频输出通道的相关属性。
- CVI_VO_GetChnAttr： 获取视频输出通道的相关属性。
- CVI_VO_ShowChn： 显示指定的视频输出通道。
- CVI_VO_HideChn： 隐藏指定的视频输出通道。
- CVI_VO_SetChnRotation： 设置视频输出通道的旋转属性。
- CVI_VO_GetChnRotation： 获取视频输出通道的旋转属性。
- CVI_VO_PauseChn： 暂停指定的 VO 通道的输出。
- CVI_VO_ResumeChn： 恢复指定的 VO 通道的输出。

5.3.1 CVI_VO_Enable

【描述】

启用视频输出设备。

【语法】

```
CVI_S32 CVI_VO_Enable(VO_DEV VoDev);
```

【参数】

参数名称	描述	输入/输出
VoDev	视频输出设备号。 取值范围：[0, VO_MAX_DEV_NUM]。	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，请参见错误码 。

【需求】

- 头文件：cvi_vo.h, cvi_comm_vo.h
- 库文件：libvpu.a

【注意】

- 必须在使用视频输出功能前先进行设备使能操作。
- 在调用设备使能前，必须对设备公共属性进行配置，否则返回设备未配置错误。
- 如果希望更改 VO 的时序配置，则需要先调用CVI_VO_Disable 接口，强制关闭 VO 硬件后再使能。以避免变更时序过程中出现不喜的瞬时。

【举例】

```
 CVI_S32 s32Ret = CVI_SUCCESS;

s32Ret = CVI_VO_SetPubAttr(VoDev, pstPubAttr);
if (s32Ret != CVI_SUCCESS) {
  SAMPLE_PRT("failed with %#x!\n", s32Ret);
  return CVI_FAILURE;
}

s32Ret = CVI_VO_Enable(VoDev);
if (s32Ret != CVI_SUCCESS) {
  SAMPLE_PRT("failed with %#x!\n", s32Ret);
  return CVI_FAILURE;
}
```

【相关主题】

CVI_VO_Disable

5.3.2 CVI_VO_Disable

【描述】

禁用视频输出设备。

【语法】

```
CVI_S32 CVI_VO_Disable(VO_DEV VoDev);
```

【参数】

参数名称	描述	输入/输出
VoDev	视频输出设备号。 取值范围：[0, VO_MAX_DEV_NUM]。	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，请参见错误码 。

【需求】

- 头文件：cvi_vo.h, cvi_comm_vo.h
- 库文件：libvpu.a

【注意】

- 设备禁止前必须先禁止该设备上的视频层。
- 只有在 VO 设备禁止下，才能做设置设备属性的变更。

【举例】

无

【相关主题】

CVI_VO_Enable

5.3.3 CVI_VO_SetPubAttr

【描述】

配置视频输出设备的相关属性。

【语法】

```
CVI_S32 CVI_VO_SetPubAttr(VO_DEV VoDev, const VO_PUB_ATTR_S *pstPubAttr);
```

【参数】

参数名称	描述	输入/输出
VoDev	视频输出设备号。 取值范围：[0, VO_MAX_DEV_NUM]。	输入
pstPubAttr	视频输出设备公共属性结构体指针。	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，请参见错误码 。

【需求】

- 头文件：cvi_vo.h, cvi_comm_vo.h
- 库文件：libvpu.a

【注意】

- 视频输出设备属性为静态属性，必须在执行CVI_VO_Enable 前配置。
- 各个 DEV 的使用说明详见VO_DEV 。
- 视频输出设备属性的使用说明详见VO_PUB_ATTR_S 章节。

【举例】

无

【相关主题】

CVI_VO_GetPubAttr

5.3.4 CVI_VO_GetPubAttr

【描述】

获取视频输出设备的相关属性。

【语法】

```
CVI_S32 CVI_VO_GtPubAttr(VO_DEV VoDev, VO_PUB_ATTR_S *pstPubAttr);
```

【参数】

参数名称	描述	输入/输出
VoDev	视频输出设备号。 取值范围：[0, VO_MAX_DEV_NUM]。	输入
pstPubAttr	视频输出设备公共属性结构体指针。	输出

【返回值】

返回值	描述
0	成功。
非 0	失败，请参见错误码 。

【需求】

- 头文件：cvi_vo.h, cvi_comm_vo.h
- 库文件：libvpu.a

【注意】

- 在设置设备公共属性前先获取属性，就可以只设置需要改变的配置项。

【举例】

无

【相关主题】

CVI_VO_SetPubAttr

5.3.5 CVI_VO_EnableVideoLayer

【描述】

启用视频层。

【语法】

```
CVI_S32 CVI_VO_EnableVideoLayer (VO_LAYER VoLayer);
```

【参数】

参数名称	描述	输入/输出
VoLayer	视频输出视频层号。 取值范围：[0, VO_MAX_LAYER_NUM]。	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，请参见错误码 。

【需求】

- 头文件：cvi_vo.h, cvi_comm_vo.h
- 库文件：libvpu.a

【注意】

- 视频层使能前必须保证该视频层所绑定的设备处于使能状态。

【举例】

无

【相关主题】

CVI_VO_DisableVideoLayer

5.3.6 CVI_VO_DisableVideoLayer

【描述】

禁用视频层。

【语法】

```
CVI_S32 CVI_VO_DisableVideoLayer (VO_LAYER VoLayer);
```

【参数】

参数名称	描述	输入/输出
VoLayer	视频输出视频层号。 取值范围：[0, VO_MAX_LAYER_NUM]。	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，请参见错误码 。

【需求】

- 头文件：cvi_vo.h, cvi_comm_vo.h
- 库文件：libvpu.a

【注意】

- 视频层禁止前，其上的通道必须先禁止。

【举例】

无

【相关主题】

CVI_VO_EnableVideoLayer

5.3.7 CVI_VO_SetVideoLayerAttr

【描述】

配置视频层的相关属性。

【语法】

```
CVI_S32 CVI_VO_SetVideoLayerAttr (VO_LAYER VoLayer, const VO_VIDEO_LAYER_ATTR_
↪S *pstLayerAttr);
```

【参数】

参数名称	描述	输入/输出
VoLayer	视频输出视频层号。 取值范围：[0, VO_MAX_LAYER_NUM]。	输入
pstLayerAttr	视频层属性结构体指针。	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，请参见错误码 。

【需求】

- 头文件：cvi_vo.h, cvi_comm_vo.h
- 库文件：libvpu.a

【注意】

无

【举例】

无

【相关主题】

CVI_VO_GetVideoLayerAttr

5.3.8 CVI_VO_GetVideoLayerAttr

【描述】

获取视频层的相关属性。

【语法】

```
CVI_S32 CVI_VO_GetVideoLayerAttr (VO_LAYER VoLayer, VO_VIDEO_LAYER_ATTR_S
↪*pstLayerAttr);
```

【参数】

参数名称	描述	输入/输出
VoLayer	视频输出视频层号。 取值范围：[0, VO_MAX_LAYER_NUM]。	输入
pstLayerAttr	视频层属性结构体指针。	输出

【返回值】

返回值	描述
0	成功。
非 0	失败，请参见错误码 。

【需求】

- 头文件：cvi_vo.h, cvi_comm_vo.h
- 库文件：libvpu.a

【注意】

无

【举例】

无

【相关主题】

CVI_VO_SetVideoLayerAttr

5.3.9 CVI_VO_EnableChn

【描述】

启用指定的视频输出通道。

【语法】

```
CVI_S32 CVI_VO_EnableChn(VO_LAYER VoLayer, VO_CHN VoChn);
```

【参数】

参数名称	描述	输入/输出
VoLayer	视频输出视频层号。 取值范围：[0, VO_MAX_LAYER_NUM]。	输入
VoChn	视频输出通道号。 取值范围：[0, VO_MAX_CHN_NUM]。	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，请参见错误码 。

【需求】

- 头文件：cvi_vo.h, cvi_comm_vo.h
- 库文件：libvpu.a

【注意】

- 调用前必须使能相应设备上的视频层。
- 信道使能前必须进行信道配置，否则返回信道未配置的错误。

【举例】

无

【相关主题】

CVI_VO_DisableChn

5.3.10 CVI_VO_DisableChn

【描述】

禁用视频输出通道。

【语法】

```
CVI_S32 CVI_VO_DisableChn(VO_LAYER VoLayer, VO_CHN VoChn);
```

【参数】

参数名称	描述	输入/输出
VoLayer	视频输出视频层号。 取值范围：[0, VO_MAX_LAYER_NUM]。	输入
VoChn	视频输出通道号。 取值范围：[0, VO_MAX_CHN_NUM]。	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，请参见错误码 。

【需求】

- 头文件：cvi_vo.h, cvi_comm_vo.h
- 库文件：libvpu.a

【注意】

无

【举例】

无

【相关主题】

CVI_VO_EnableChn

5.3.11 CVI_VO_SetChnAttr

【描述】

配置视频输出信道的相关属性。

【语法】

```
CVI_S32 CVI_VO_SetPubAttr(VO_LAYER VoLayer, VO_CHN VoChn, const VO_CHN_ATTR_
↪S *pstChnAttr);
```

【参数】

参数名称	描述	输入/输出
VoLayer	视频输出视频层号。 取值范围：[0, VO_MAX_LAYER_NUM]。	输入
VoChn	视频输出通道号。 取值范围：[0, VO_MAX_CHN_NUM]。	输入
pstChnAttr	视频输出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，请参见错误码 。

【需求】

- 头文件：cvi_vo.h, cvi_comm_vo.h
- 库文件：libvpu.a

【注意】

- 通道显示区域需小于视频层属性中设定的 stImageSize 大小。

【举例】

无

【相关主题】

CVI_VO_GetChnAttr

5.3.12 CVI_VO_GetChnAttr

【描述】

获取视频输出通道的相关属性。

【语法】

```
CVI_S32 CVI_VO_GtPubAttr(VO_LAYER VoLayer, VO_CHN VoChn, VO_CHN_ATTR_S␣
↪*pstChnAttr);
```

【参数】

参数名称	描述	输入/输出
VoLayer	视频输出视频层号。 取值范围：[0, VO_MAX_LAYER_NUM]。	输入
VoChn	视频输出通道号。 取值范围：[0, VO_MAX_CHN_NUM]。	输入
pstChnAttr	视频输出通道属性结构体指针。	输出

【返回值】

返回值	描述
0	成功。
非 0	失败，请参见错误码 。

【需求】

- 头文件：cvi_vo.h, cvi_comm_vo.h
- 库文件：libvpu.a

【注意】

无

【举例】

无

【相关主题】

CVI_VO_SetChnAttr

5.3.13 CVI_VO_ShowChn

【描述】

显示指定的视频输出通道。

【语法】

```
CVI_S32 CVI_VO_ShowChn(VO_LAYER VoLayer, VO_CHN VoChn);
```

【参数】

参数名称	描述	输入/输出
VoLayer	视频输出视频层号。 取值范围：[0, VO_MAX_LAYER_NUM]。	输入
VoChn	视频输出通道号。 取值范围：[0, VO_MAX_CHN_NUM]。	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，请参见错误码 。

【需求】

- 头文件：cvi_vo.h, cvi_comm_vo.h

- 库文件：libvpu.a

【注意】

- 调用前必须使能相应设备上的视频通道。
- 默认下处于显示状态。

【举例】

无

【相关主题】

CVI_VO_HideChn

5.3.14 CVI_VO_HideChn

【描述】

隐藏指定的视频输出通道。

【语法】

```
CVI_S32 CVI_VO_HideChn(VO_LAYER VoLayer, VO_CHN VoChn);
```

【参数】

参数名称	描述	输入/输出
VoLayer	视频输出视频层号。 取值范围：[0, VO_MAX_LAYER_NUM]。	输入
VoChn	视频输出通道号。 取值范围：[0, VO_MAX_CHN_NUM]。	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，请参见错误码 。

【需求】

- 头文件：cvi_vo.h, cvi_comm_vo.h
- 库文件：libvpu.a

【注意】

无

【举例】

无

【相关主题】

CVI_VO_ShowChn

5.3.15 CVI_VO_SetChnRotation

【描述】

设置 VO 通道旋转的属性。

【语法】

```
CVI_S32 CVI_VO_SetChnRotation(VO_LAYER VoLayer, VO_CHN VoChn, ROTATION_E
↪enRotation);
```

【参数】

参数名称	描述	输入/输出
VoLayer	视频输出视频层号。 取值范围：[0, VO_MAX_LAYER_NUM]。	输入
VoChn	视频输出通道号。 取值范围：[0, VO_MAX_CHN_NUM]。	输入
enRotation	旋转属性。详见 ROTATION_E 说明	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，请参见错误码 。

【需求】

- 头文件：cvi_vo.h, cvi_comm_vo.h
- 库文件：libvpu.a

【注意】

- 使用本接口前，需先调用CVI_VO_SetChnAttr，否则提示失败。
- CV183x 仅支持 RGB planar，YUV420 Planar 及 YUV400 三种格式的旋转。
- CV182x/CV181x 仅支持 NV12，NV21 及 YUV400 三种格式的旋转。
- 设置后，旋转会作用于进入信道的视频图像，要注意设置视频层和视频信道的属性大小。例如通道设定大小为 1920x1080，且旋转 90 度，因此输入给信道的图像应为 1080x1920。

【举例】

无

【相关主题】

无

5.3.16 CVI_VO_GetChnRotation

【描述】

获取 VO 通道旋转的属性。

【语法】

```
CVI_S32 CVI_VO_GetChnRotation(VO_LAYER VoLayer, VO_CHN VoChn, ROTATION_E␣
↪*penRotation);
```

【参数】

参数名称	描述	输入/输出
VoLayer	视频输出视频层号。 取值范围：[0, VO_MAX_LAYER_NUM]。	输入
VoChn	视频输出通道号。 取值范围：[0, VO_MAX_CHN_NUM]。	输入
penRotation	旋转属性指针。详见 ROTATION_E 说明	输出

【返回值】

返回值	描述
0	成功。
非 0	失败，请参见错误码 。

【需求】

- 头文件：cvi_vo.h, cvi_comm_vo.h
- 库文件：libvpu.a

【注意】

- 使用本接口前，需先调用CVI_VO_SetChnAttr，否则提示失败。

【举例】

无

【相关主题】

CVI_VO_SetChnRotation

5.3.17 CVI_VO_PauseChn

【描述】

暂停指定的 VO 通道的输出。

【语法】

```
CVI_S32 CVI_VO_PauseChn(VO_LAYER VoLayer, VO_CHN VoChn);
```

【参数】

参数名称	描述	输入/输出
VoLayer	视频输出视频层号。 取值范围：[0, VO_MAX_LAYER_NUM]。	输入
VoChn	视频输出通道号。 取值范围：[0, VO_MAX_CHN_NUM]。	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，请参见错误码 。

【需求】

- 头文件：cvi_vo.h, cvi_comm_vo.h
- 库文件：libvpu.a

【注意】

- 调用前必须使能相应设备上的视频通道。
- 允许重复暂停同一通道，不返回失败。

【举例】

无

【相关主题】

CVI_VO_ResumeChn

5.3.18 CVI_VO_ResumeChn

【描述】

恢复指定的 VO 通道的输出。

【语法】

```
CVI_S32 CVI_VO_ResumeChn(VO_LAYER VoLayer, VO_CHN VoChn);
```

【参数】

参数名称	描述	输入/输出
VoLayer	视频输出视频层号。 取值范围：[0, VO_MAX_LAYER_NUM]。	输入
VoChn	视频输出通道号。 取值范围：[0, VO_MAX_CHN_NUM]。	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，请参见错误码 。

【需求】

- 头文件：cvi_vo.h, cvi_comm_vo.h
- 库文件：libvpu.a

【注意】

- 调用前必须使能相应设备上的视频通道。
- 允许重复恢复同一通道，不返回失败。

【举例】

无

【相关主题】

CVI_VO_PauseChn

5.3.19 CVI_VO_CloseFd

【描述】

关闭视频输出设备的文件句柄。

【语法】

```
CVI_S32 CVI_VO_CloseFd(CVI_VOID);
```

【参数】

无

【返回值】

返回值	描述
0	成功。
非 0	失败，请参见错误码 。

【需求】

- 头文件：cvi_vo.h, cvi_comm_vo.h
- 库文件：libvpu.a

【注意】

此接口调用后，VO 其他 MMF 接口失效。

【举例】

无

【相关主题】

无

5.4 数据类型

5.4.1 VO_DEV

【说明】

定义设备号。

【定义】

```
typedef CVI_U32 VO_DEV;
```

【成员】

处理器	描述
CV183x	仅有一个视频输出设备。 0: DHD0 支持 LCD, MIPI TX, BT.656, BT.1120
CV182xCV181x	仅有一个视频输出设备。 0: DHD0 支持 LCD, MIPI TX, BT.656, BT.1120, RGB, I80

【注意事项】

无

【相关数据类型及接口】

无

5.4.2 VO_LAYER

【说明】

定义视频层号。

【定义】

```
typedef CVI_U32 VO_LAYER;
```

【成员】

处理器	描述
CV183x	仅有一个视频输出层。 0: VHD0
CV182xCV181x	仅有一个视频输出层。 0: VHD0

【注意事项】

无

【相关数据类型及接口】

无

5.4.3 VO_INTF_TYPE

【说明】

定义视频输出设备的接口号。

【定义】

```
#define VO_INTF_CVBS (0x01L << 0)
#define VO_INTF_YPBPR (0x01L << 1)
#define VO_INTF_VGA (0x01L << 2)
#define VO_INTF_BT656 (0x01L << 3)
#define VO_INTF_BT1120 (0x01L << 6)
#define VO_INTF_LCD (0x01L << 7)
#define VO_INTF_LCD_18BIT (0x01L << 10)
#define VO_INTF_LCD_24BIT (0x01L << 11)
#define VO_INTF_LCD_30BIT (0x01L << 12)
#define VO_INTF_MIPI (0x01L << 13)
#define VO_INTF_MIPI_SLAVE (0x01L << 14)
#define VO_INTF_HDMI (0x01L << 15)

typedef CVI_U32 VO_INTF_TYPE_E;
```

【注意事项】

无

【相关数据类型及接口】

无

5.4.4 VO_INTF_SYNC_E

【说明】

定义视频输出设备的标准时序。

【定义】

```
typedef enum _VO_INTF_SYNC_E {
  VO_OUTPUT_PAL = 0,
  VO_OUTPUT_NTSC,
  VO_OUTPUT_1080P24,
  VO_OUTPUT_1080P25,
  VO_OUTPUT_1080P30,
  VO_OUTPUT_720P50,
  VO_OUTPUT_720P60,
```

(下页继续)

(续上页)

```
  VO_OUTPUT_1080P50,
  VO_OUTPUT_1080P60,
  VO_OUTPUT_576P50,
  VO_OUTPUT_480P60
  VO_OUTPUT_800x600_60,
  VO_OUTPUT_1024x768_60,
  VO_OUTPUT_1280x1024_60,
  VO_OUTPUT_1366x768_60
  VO_OUTPUT_1440x900_60,
  VO_OUTPUT_1280x800_60,
  VO_OUTPUT_1600x1200_60,
  VO_OUTPUT_1680x1050_60
  VO_OUTPUT_1920x1200_60,
  VO_OUTPUT_640x480_60,
  VO_OUTPUT_1920x2160_30,
  VO_OUTPUT_2560x1440_30,
  VO_OUTPUT_2560x1440_60,
  VO_OUTPUT_2560x1600_60,
  VO_OUTPUT_3840x2160_24,
  VO_OUTPUT_3840x2160_25,
  VO_OUTPUT_3840x2160_30,
  VO_OUTPUT_3840x2160_50,
  VO_OUTPUT_3840x2160_60,
  VO_OUTPUT_4096x2160_24,
  VO_OUTPUT_4096x2160_25,
  VO_OUTPUT_4096x2160_30,
  VO_OUTPUT_4096x2160_50,
  VO_OUTPUT_4096x2160_60,
  VO_OUTPUT_720x1280_60, /* For MIPI DSI Tx 720 x1280 at 60 Hz */
  VO_OUTPUT_1080x1920_60, /* For MIPI DSI Tx 1080x1920 at 60 Hz */
  VO_OUTPUT_USER, /* User timing. */

  VO_OUTPUT_BUTT
} VO_INTF_SYNC_E;
```

【注意事项】

无

【相关数据类型及接口】

无

5.4.5 VO_SYNC_INFO_S

【说明】

定义设备的时序结构体

【定义】

```
typedef struct _VO_SYNC_INFO_S {
  CVI_BOOL bSynm;
  CVI_BOOL bIop;
  CVI_U16 u16FrameRate;

  CVI_U16 u16Vact;
  CVI_U16 u16Vbb;
  CVI_U16 u16Vfb;

  CVI_U16 u16Hact;
  CVI_U16 u16Hbb;
  CVI_U16 u16Hfb;

  CVI_U16 u16Hpw;
  CVI_U16 u16Vpw;

  CVI_BOOL bIdv;
  CVI_BOOL bIhs;
  CVI_BOOL bIvs;
} VO_SYNC_INFO_S;
```

【成员】

成员名称	描述
bSynm	同步讯号模式， 0: embedded ync 1: separate sync。
bIop	0: interlaced, 1:progressive。
u16FrameRate	每秒更新幅数
u16Vact	垂直影像行数
u16Vbb	垂直 back porch 行数
u16Vfb	垂直 front porch 行数
u16Hact	水平影像相数
u16Hbb	水平 back porch 相数
u16Hfb	水平 front porch 相数
u16Hpw	水平同步相数
u16Vpw	垂直同步行数
bIdv	Data valid 是否反向
bIhs	水平同步是否反向
bIvs	垂直同步是否反向

【注意事项】

无

【相关数据类型及接口】

无

5.4.6 VO_PUB_ATTR_S

【说明】

定义输出设备结构体

【定义】

```
typedef struct _VO_PUB_ATTR_S {
  CVI_U32 u32BgColor;
  VO_INTF_TYPE_E enIntfType;
  VO_INTF_SYNC_E enIntfSync;
  VO_SYNC_INFO_S stSyncInfo;
  union {
    VO_I80_CFG_S sti80Cfg;
    VO_LVDS_ATTR_S stLvdsAttr;
  };
} VO_PUB_ATTR_S;
```

【成员】

成员名称	描述
u32BgColor	背景色。 为 RGBAAA 的格式 bit[9:0] 为 B bit[19:10] 为 G bit[29:20] 为 R
enIntfType	输出设备的接口。
enIntfSync	输出设备的标准时序
stSyncInfo	输出设备的自定义时序，只有在 enIntfSync 为 VO_OUTPUT_USER 时作用
sti80Cfg	输出设备接口为 I80 时的接口属性
stLvdsAttr	输出设备接口为 LCD 时的接口属性

【注意事项】

无

【相关数据类型及接口】

无

5.4.7 VO_VIDEO_LAYER_ATTR_S

【说明】

定义视频层结构体

【定义】

```
typedef struct _VO_VIDEO_LAYER_ATTR_S {
  RECT_S stDispRect;
  SIZE_S stImageSize;
  CVI_U32 u32DispFrmRt;
  PIXEL_FORMAT_E enPixFormat;
} VO_VIDEO_LAYER_ATTR_S;
```

【成员】

成员名称	描述
stDispRect	视频层的显示范围，需小于等于设备时序
stImageSize	影像大小，若无支持 scaling，应等于 stDispRect
u32DispFrmRt	显示更新张数
enPixFormat	视频层的影像格式

【注意事项】

无

【相关数据类型及接口】

无

5.4.8 VO_CHN_ATTR_S

【说明】

定义输出信道结构体

【定义】

```
typedef struct _VO_CHN_ATTR_S {
  CVI_U32 u32Priority;
  RECT_S stRect;
} VO_CHN_ATTR_S;
```

【成员】

成员名称	描述
u32Priority	在多通道时，优先极高 (比较小) 的会在上。
stRect	信道的显示区域。

【注意事项】

无

【相关数据类型及接口】

无

5.5 错误码

错误代码	宏定义	描述
0xC00D8001	CVI_ERR_VO_INVALID_DEVID	设备 ID 不合
0xC00D8002	CVI_ERR_VO_INVALID_CHNID	通道 ID 不合
0xC00D8003	CVI_ERR_SYS_ILLEGAL_PARAM	参数设置无效
0xC00D8006	CVI_ERR_SYS_NULL_PTR	空指针
0xC00D8008	CVI_ERR_SYS_NOT_SUPPORT	不支持的功能
0xC00D8009	CVI_ERR_SYS_NOT_PERM	不允许的操作
0xC00D800C	CVI_ERR_SYS_NOMEM	内存分配失败，如系统内存不足
0xC00D8010	CVI_ERR_SYS_NOTREADY	系统控制属性未配置
0xC00D8012	CVI_ERR_VO_BUSY	系统忙碌中
0xC00D8040	CVI_ERR_VO_DEV_NOT_CONFIG	设备未设置
0xC00D8041	CVI_ERR_VO_DEV_NOT_ENABLE	设备未启用
0xC00D8042	CVI_ERR_VO_DEV_HAS_ENABLED	设备已启用
0xC00D8045	CVI_ERR_VO_VIDEO_NOT_ENABLE	视频层未启用
0xC00D8046	CVI_ERR_VO_VIDEO_NOT_DISABLE	视频层未禁止
0xC00D8047	CVI_ERR_VO_VIDEO_NOT_CONFIG	视频层未设置
0xC00D8048	CVI_ERR_VO_CHN_NOT_DISABLE	视频通道未禁止
0xC00D8049	CVI_ERR_VO_CHN_NOT_ENABLE	视频通道未启用
0xC00D804A	CVI_ERR_VO_CHN_NOT_CONFIG	视频通道未设置
0xC00D804E	CVI_ERR_VO_WAIT_TIMEOUT	等待超时
0xC00D804F	CVI_ERR_VO_INVALID_VFRAME	视频幅无效
0xC00D8050	CVI_ERR_VO_INVALID_RECT_PARA	矩阵参数无效
0xC00D8065	CVI_ERR_VO_CHN_AREA_OVERLAP	视频信道区域重迭
0xC00D8066	CVI_ERR_VO_INVALID_LAYERID	视频层 ID 不合

6 视频处理子系统

6.1 功能概述

6.1.1 目的

VPSS(Video Process Sub System) 是视频处理子系统，支持的具体图像处理功能包括 Crop、Scale 、像素格式转换、Mirror/Flip、固定角度旋转、鱼眼校正、Overlay/Overlayex 等。

6.1.2 定义及缩写

VPSS（video process sub system, 视频处理子系统）。

Grp（Group, 在 VPSS 里面表示 VPSS 的一个组）。

Chn（channel,VPSS 组的通道）。

VB（Video Buffer 影像内存区块）。

6.2 设计概述

6.2.1 系统架构

以下为 VPSS 基本概念

- GROUP

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

- CHANNEL

VPSS 组的通道。VPSS 硬件提供多个物理信道，每个信道具有缩放、裁剪等功能。它通过绑定物理信道，将物理信道输出作为自己的输入，把图像缩放成用户设置的目标分辨率输出。

- Crop

裁剪，分为 2 种：组裁剪以及物理通道裁剪

F 组裁剪，VPSS 对输入图像进行裁剪。

F 物理信道裁剪，VPSS 对各个物理通道的输出图像进行裁剪。

- 像素格式转换

支持输入输出图像的数据格式转换。

F 支持 YUV420 planar, YUV422 planar, RGB planar, RGB packed, BGR packed 互相转换。

- Scale

缩放对图像进行缩小放大。物理通道水平、垂直最小支持 32 倍缩小，最大支持 32 倍放大。

- Mirror/Flip

Mirror 即水平镜像 Flip 即上下翻转。可使用 Mirror + Flip 实现 180 旋转。

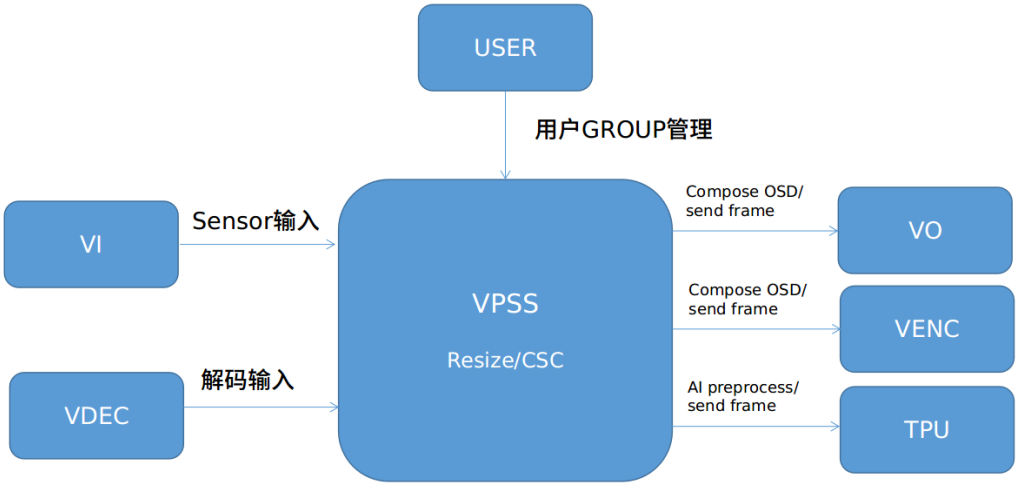
- Overlay/Overlayex

视频叠加区域，调用 RGN 对 VPSS 通道的输出图像叠加位图，支持位图格式 ARGB4444 、ARGB1555 、ARGB8888 、256 LUT(ARGB4444), Font-based。

- 固定角度旋转

调用 GDC 对 VPSS 通道的输出图像处理。支持 0 度、90 度、180 度以及 270 度固定角度的旋转功能。

Vpss 模块在系统中的位置如下图所示：



通过调用 SYS 模块提供的 binding 接口，VPSS 模块可以与 VI/VDEC 等输入模块进行绑定，将解码以及 Sensor 的数据送到 VPSS 进行处理，可以实现图像的缩放以及色彩空间的转换等功能。同时也可以通过 binding 接口，将 VPSS 模块处理之后的图像，经过 OSD 的合成送到 VO/VENC 模块，也支持做一些简单的 Deep Learning 前处理，然后将处理后的图片送到 TPU 进行 Deep Learning 运算。

表格 6- 1 VPSS 功能限制

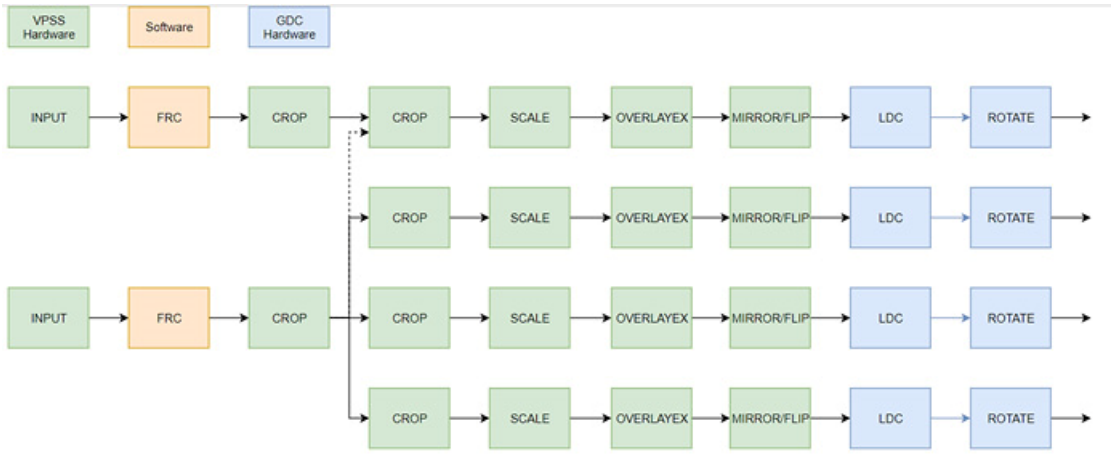
工作模式	功能		
	组裁剪	缩放	通道裁剪
VI_OFFLINE_VPSS_OFFLINE	支持	支持	支持
VI_OFFLINE_VPSS_ONLINE	不支持	支持	支持
VI_ONLINE_VPSS_OFFLINE	支持	支持	支持
VI_ONLINE_VPSS_ONLINE	不支持	支持	支持

注意: 注意：当 VPSS ONLINE 时，最多只能有两组 GRP 分别接收前端两个 sensor 来的数据，无法服务其他不同源的需求

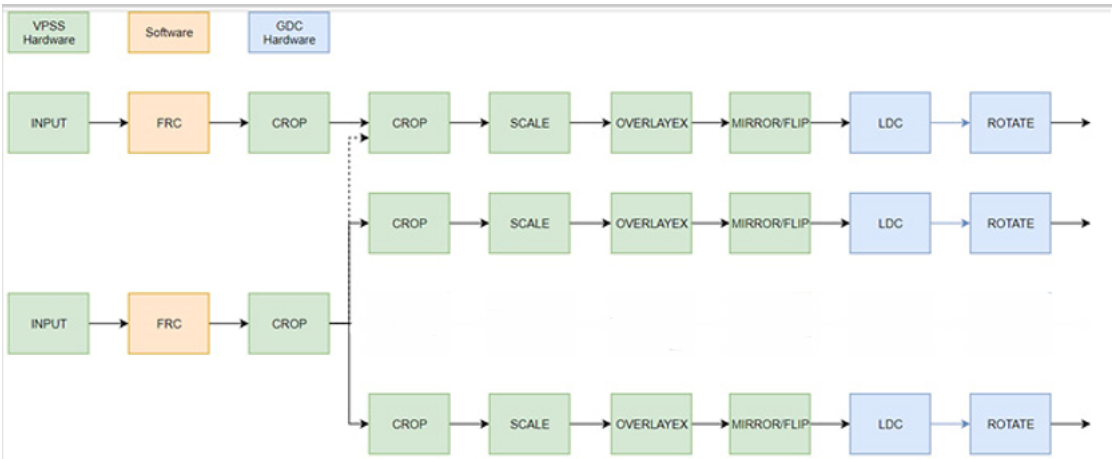
6.2.2 注意事项

VPSS 数据处理流程图如下图 6-1，有两组 INPUT。当单一 INPUT 使用时，可以有四组通道输出 (CV180x 为三组)；当两组 INPUT 使用时，则第一组 INPUT 只有一组通道输出，第二组 INPUT 只有三组通道输出 (CV180x 为两组)。

图表 6- 1 CV181x 的数据流程图



CV180x 的数据流程图



CV180x/CV181x VPSS 硬件规格

Chip	CV180x	CV181x
img_in_size	2880	2880
sc_v1_size	2880	2880
sc_v2_size	1920	1920
sc_v3_size	1280	-
sc_d_size	1920	1280

6.3 API 参考

该功能模块为用户提供以下 API:

- CVI_VPSS_CreateGrp: 创建一个 VPSS GROUP。
- CVI_VPSS_DestroyGrp: 销毁一个 VPSS GROUP。
- CVI_VPSS_GetGrpAttr: 获取 VPSS GROUP 属性。
- CVI_VPSS_SetGrpAttr: 设置 VPSS GROUP 属性。
- CVI_VPSS_StartGrp: 启用 VPSS GROUP。
- CVI_VPSS_StopGrp: 禁用 VPSS GROUP。
- CVI_VPSS_ResetGrp: 重置一个 VPSS GROUP。
- CVI_VPSS_GetGrpProcAmpCtrl: 获取一个 VPSS GROUP 的颜色控制功能描述。
- CVI_VPSS_GetGrpProcAmp: 获取一个 VPSS GROUP 的颜色控制属性。
- CVI_VPSS_SetGrpProcAmp: 设置一个 VPSS GROUP 的颜色控制属性。
- CVI_VPSS_SetGrpParamfromBin: 根据 Bin 设置一个 VPSS GROUP 的属性。
- CVI_VPSS_GetChnAttr: 获取 VPSS 通道属性。
- CVI_VPSS_SetChnAttr: 设置 VPSS 通道属性。
- CVI_VPSS_EnableChn: 启用 VPSS 通道。
- CVI_VPSS_DisableChn: 禁用 VPSS 通道
- CVI_VPSS_SetGrpCrop: 设置 VPSS GROUP CROP 功能属性。
- CVI_VPSS_GetGrpCrop: 获取 VPSS GROUP CROP 功能属性。
- CVI_VPSS_SendFrame: 用户向 VPSS 发送数据。
- CVI_VPSS_GetChnFrame: 用户从通道获取一帧处理完成的图像。
- CVI_VPSS_SendChnFrame: 用户向 VPSS 指定通道数据。

- CVI_VPSS_ReleaseChnFrame: 用户释放一帧通道处理完成的图像
- CVI_VPSS_GetGrpFrame: 用户从 VPSS 获得 Group 所属图像。
- CVI_VPSS_ReleaseGrpFrame: 用户释放该 Group 所属图像。
- CVI_VPSS_SetChnCrop: 设置 VPSS 信道裁剪功能属性。
- CVI_VPSS_GetChnCrop: 获取 VPSS 信道裁剪功能属性。
- CVI_VPSS_SetChnRotation: 设置 VPSS 通道旋转的属性。
- CVI_VPSS_GetChnRotation: 获取 VPSS 通道图像旋转属性。
- CVI_VPSS_GetChnFd: 获取 VPSS 通道对应的设备文件句柄。
- CVI_VPSS_CloseFd: 关闭 VPSS 设备通道的文件句柄。
- CVI_VPSS_SetChnBufWrapAttr: 设置低延时卷绕属性。
- CVI_VPSS_GetChnBufWrapAttr: 获取低延时卷绕属性。
- CVI_VPSS_GetWrapBufferSize: 获取所需低延时卷绕缓存大小。

6.3.1 CVI_VPSS_CreateGrp

【描述】

创建一个 VPSS GROUP

【语法】

```
CVI_S32 CVI_VPSS_CreateGrp(VPSS_GRP VpssGrp, const VPSS_GRP_ATTR_S *pstGrpAttr);
```

【参数】

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

【返回值】

返回值	描述
0	成功。
非 0	失败, 请参见错误码 。

【需求】

- 头文件：cvi_comm_vpss.h、cvi_vpss.h
- 库文件：libvpu.a

【注意】

· 使用之前要先创建 VPSS_GRP_ATTR_S

【举例】

```
VPSS_GRP_ATTR_S stVpssGrpAttr;
VPSS_CHN VpssChn = VPSS_CHN0;
CVI_BOOL abChnEnable[VPSS_MAX_PHY_CHN_NUM] = {0};
VPSS_CHN_ATTR_S astVpssChnAttr[VPSS_MAX_PHY_CHN_NUM];
CVI_S32 s32Ret = CVI_SUCCESS;
VPSS_CROP_INFO_S pstCropInfo;

stVpssGrpAttr.stFrameRate.s32SrcFrameRate = -1;
stVpssGrpAttr.stFrameRate.s32DstFrameRate = -1;
stVpssGrpAttr.enPixelFormat = PIXEL_FORMAT_YUV_PLANAR_420;
stVpssGrpAttr.u32MaxW = stSize.u32Width;
stVpssGrpAttr.u32MaxH = stSize.u32Height;

astVpssChnAttr[VpssChn].u32Width = 800;
astVpssChnAttr[VpssChn].u32Height = 600;
astVpssChnAttr[VpssChn].enChnMode = VPSS_CHN_MODE_USER;
astVpssChnAttr[VpssChn].enVideoFormat = VIDEO_FORMAT_LINEAR;
astVpssChnAttr[VpssChn].enPixelFormat = PIXEL_FORMAT_BGR_888_PLANAR;
astVpssChnAttr[VpssChn].stFrameRate.s32SrcFrameRate = 30;
astVpssChnAttr[VpssChn].stFrameRate.s32DstFrameRate = 30;
astVpssChnAttr[VpssChn].u32Depth = 0;
astVpssChnAttr[VpssChn].bMirror = CVI_FALSE;
astVpssChnAttr[VpssChn].bFlip = CVI_FALSE;
astVpssChnAttr[VpssChn].stAspectRatio.enMode = ASPECT_RATIO_NONE;

/*start vpss*/
VPSS_CHN VpssChn;
s32Ret = CVI_VPSS_CreateGrp(0, &stVpssGrpAttr);
if (s32Ret != CVI_SUCCESS) {
  SAMPLE_PRT("CVI_VPSS_CreateGrp failed with %#x!\n", s32Ret);
  return CVI_FAILURE;
}

s32Ret = CVI_VPSS_SetChnAttr(0, 0, &astVpssChnAttr[0]);
if (s32Ret != CVI_SUCCESS) {
  SAMPLE_PRT("CVI_VPSS_SetChnAttr failed with %#x\n", s32Ret);
  return CVI_FAILURE;
}

s32Ret = CVI_VPSS_EnableChn(0, 0);
if (s32Ret != CVI_SUCCESS) {
  SAMPLE_PRT("CVI_VPSS_EnableChn failed with %#x\n", s32Ret);
  return CVI_FAILURE;
}

s32Ret = CVI_VPSS_StartGrp(0);
if (s32Ret != CVI_SUCCESS) {
  SAMPLE_PRT("CVI_VPSS_StartGrp failed with %#x\n", s32Ret);
  return CVI_FAILURE;
}

s32Ret = CVI_VPSS_GetGrpCrop(0, &pstCropInfo);
```

(下页继续)

(续上页)

```
if (s32Ret != CVI_SUCCESS) {
  SAMPLE_PRT("CVI_VPSS_ GetGrpCrop failed with %#x\n", s32Ret);
  return CVI_FAILURE;
}

pstCropInfo.stCropRect.s32X = 0;
pstCropInfo.stCropRect.s32Y = 0;
pstCropInfo.stCropRect.u32Width = 600;
pstCropInfo.stCropRect.u32Height = 600;

s32Ret = CVI_VPSS_SetGrpCrop(0, &pstCropInfo);
if (s32Ret != CVI_SUCCESS) {
  SAMPLE_PRT("CVI_VPSS_ SetGrpCrop failed with %#x\n", s32Ret);
  return CVI_FAILURE;
}

s32Ret = CVI_VPSS_DisableChn(0, 0);
if (s32Ret != CVI_SUCCESS) {
  SAMPLE_PRT("failed with %#x!\n", s32Ret);
  return CVI_FAILURE;
}

s32Ret = CVI_VPSS_StopGrp(VpssGrp);
if (s32Ret != CVI_SUCCESS) {
  SAMPLE_PRT("failed with %#x!\n", s32Ret);
  return CVI_FAILURE;
}

s32Ret = CVI_VPSS_DestroyGrp(VpssGrp);
if (s32Ret != CVI_SUCCESS) {
  SAMPLE_PRT("failed with %#x!\n", s32Ret);
  return CVI_FAILURE;
}
```

【相关主题】

- CVI_VPSS_DestroyGrp
- CVI_VPSS_SetChnAttr
- CVI_VPSS_EnableChn
- CVI_VPSS_DisableChn
- CVI_VPSS_StartGrp
- CVI_VPSS_StopGrp
- CVI_VPSS_GetGrpCrop
- CVI_VPSS_SetGrpCrop

6.3.2 CVI_VPSS_DestroyGrp

【描述】

销毁一个 VPSS GROUP

【语法】

```
CVI_S32 CVI_VPSS_DestroyGrp (VPSS_GRP VpssGrp);
```

【参数】

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

【返回值】

返回值	描述
0	成功。
非 0	失败, 请参见错误码 。

【需求】

- 头文件：cvi_comm_vpss.h、cvi_vpss.h
- 库文件：libvpu.a

【注意】

- 调用此接口之前, 必须先调用 CVI_VPSS_StopGrp 禁用此 GROUP
- 该函数会释放掉该 grp 下的所有 vb_jobs

【举例】

请参见CVI_VPSS_CreateGrp

【相关主题】

CVI_VPSS_CreateGrp

6.3.3 CVI_VPSS_GetGrpAttr

【描述】

获取 VPSS GROUP 属性.

【语法】

```
CVI_S32 CVI_VPSS_GetGrpAttr (VPSS_GRP VpssGrp, VPSS_GRP_ATTR_S *pstGrpAttr);
```

【参数】

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

【返回值】

返回值	描述
0	成功。
非 0	失败, 请参见错误码 。

【需求】

- 头文件：cvi_comm_vpss.h、cvi_vpss.h
- 库文件：libvpu.a

【注意】

- Group 必须已经创建
- GROUP 属性必须合法, 其中部分静态属性不可动态设置, 具体请参见 VPSS_GRP_ATTR_S

【举例】

请参见CVI_VPSS_CreateGrp

【相关主题】

CVI_VPSS_CreateGrp

6.3.4 CVI_VPSS_SetGrpAttr

【描述】

设置 VPSS GROUP 属性.

【语法】

```
CVI_S32 CVI_VPSS_SetGrpAttr (VPSS_GRP VpssGrp, const VPSS_GRP_ATTR_S *pstGrpAttr);
```

【参数】

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

【返回值】

返回值	描述
0	成功。
非 0	失败，请参见错误码 。

【需求】

- 头文件：cvi_comm_vpss.h、cvi_vpss.h
- 库文件：libvpu.a

【注意】

- Group 必须已经创建
- GROUP 属性必须合法，其中部分静态属性不可动态设置，具体请参见 VPSS_GRP_ATTR_S

【举例】

请参见CVI_VPSS_CreateGrp

【相关主题】

CVI_VPSS_CreateGrp

6.3.5 CVI_VPSS_StartGrp

【描述】

启用 VPSS GROUP.

【语法】

```
CVI_S32 CVI_VPSS_StartGrp (VPSS_GRP VpssGrp);
```

【参数】

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

【返回值】

返回值	描述
0	成功。
非 0	失败，请参见错误码 。

【需求】

- 头文件：cvi_comm_vpss.h、cvi_vpss.h
- 库文件：libvpu.a

【注意】

- Group 必须已经创建
- 重复调用该函数设置同一个组返回成功

【举例】

请参见CVI_VPSS_CreateGrp

【相关主题】

CVI_VPSS_CreateGrp

6.3.6 CVI_VPSS_StopGrp

【描述】

禁用 VPSS GROUP.

【语法】

```
CVI_S32 CVI_VPSS_StopGrp (VPSS_GRP VpssGrp);
```

【参数】

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

【返回值】

返回值	描述
0	成功。
非 0	失败, 请参见错误码 。

【需求】

- 头文件：cvi_comm_vpss.h、cvi_vpss.h
- 库文件：libvpu.a

【注意】

- Group 必须已经创建，不然会返回失败
- 重复禁用同一 VPSS GROUP 返回成功

【举例】

请参见CVI_VPSS_CreateGrp

【相关主题】

CVI_VPSS_CreateGrp

6.3.7 CVI_VPSS_ResetGrp

【描述】

复位 VPSS GROUP.

【语法】

```
CVI_S32 CVI_VPSS_ResetGrp (VPSS_GRP VpssGrp);
```

【参数】

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

【返回值】

返回值	描述
0	成功。
非 0	失败, 请参见错误码 。

【需求】

- 头文件：cvi_comm_vpss.h、cvi_vpss.h
- 库文件：libvpu.a

【注意】

Group 必须已经创建

【举例】

请参见CVI_VPSS_CreateGrp

【相关主题】

CVI_VPSS_CreateGrp

6.3.8 CVI_VPSS_GetGrpProcAmpCtrl

【描述】

获取一个 VPSS GROUP 颜色控制功能描述.

【语法】

```
CVI_S32 CVI_VPSS_GetGrpProcAmpCtrl(VPSS_GRP VpssGrp, PROC_AMP_E type, PROC_
↪AMP_CTRL_S *ctrl);
```

【参数】

参数名称	描述	输入/输出
VpssGrp	VPSS GROUP 号。 取值范围:[0, VPSS_MAX_GRP_NUM)。	输入
type	Proc Amp(颜色控制) 种类	输入
ctrl	定义 ProcAmp 的特定，包含 min, max, step, default 等	输出

【返回值】

返回值	描述
0	成功。
非 0	失败, 请参见错误码 。

【需求】

- 头文件：cvi_comm_vpss.h、cvi_vpss.h
- 库文件：libvpu.a

【注意】

【举例】

```
PROC_AMP_CTRL_S stAmpCtrl;
CVI_S32 tmp;

 if (CVI_VPSS_GetGrpProcAmp(0, PROC_AMP_BRIGHTNESS, &tmp) != CVI_SUCCESS) {
   CVI_TRACE_LOG(CVI_DBG_ERR, "CVI_VPSS_GetGrpProcAmp NG on grp0!\n");
   return CVI_FAILURE;
 }

 if (CVI_VPSS_GetGrpProcAmpCtrl(0, PROC_AMP_BRIGHTNESS, &stAmpCtrl) != CVI_
↪SUCCESS) {
   CVI_TRACE_LOG(CVI_DBG_ERR, "CVI_VPSS_GetGrpProcAmpCtrl NG on grp0!\n");
   return CVI_FAILURE;
 }

 if (CVI_VPSS_SetGrpProcAmp(0, PROC_AMP_BRIGHTNESS, stAmpCtrl.maximum) != CVI_
↪SUCCESS) {
   CVI_TRACE_LOG(CVI_DBG_ERR, "CVI_VPSS_SetGrpProcAmp NG on grp0!\n");
   return CVI_FAILURE;
 }
```

【相关主题】

- PROC_AMP_E
- PROC_AMP_CTRL_S
- CVI_VPSS_GetGrpProcAmp
- CVI_VPSS_SetGrpProcAmp

6.3.9 CVI_VPSS_GetGrpProcAmp

【描述】

获取一个 VPSS GROUP 的颜色控制属性.

【语法】

```
CVI_S32 CVI_VPSS_GetGrpProcAmp(VPSS_GRP VpssGrp, PROC_AMP_E type, CVI_S32
↪*value);
```

【参数】

参数名称	描述	输入/输出
VpssGrp	VPSS GROUP 号。 取值范围:[0, VPSS_MAX_GRP_NUM)。	输入
type	Proc Amp(颜色控制) 种类	输入
value	ProcAmp 的设定	输出

【返回值】

返回值	描述
0	成功。
非 0	失败, 请参见错误码 。

【需求】

- 头文件：cvi_comm_vpss.h、cvi_vpss.h
- 库文件：libvpu.a

【注意】

无。

【举例】

请参见CVI_VPSS_GetGrpProcAmpCtrl

【相关主题】

- PROC_AMP_E
- PROC_AMP_CTRL_S
- CVI_VPSS_GetGrpProcAmpCtrl
- CVI_VPSS_SetGrpProcAmp

6.3.10 CVI_VPSS_SetGrpProcAmp

【描述】

设置一个 VPSS GROUP 的颜色控制属性.

【语法】

```
CVI_S32 CVI_VPSS_SetGrpProcAmp(VPSS_GRP VpssGrp, PROC_AMP_E type, const CVI_S32
↪value);
```

【参数】

参数名称	描述	输入/输出
VpssGrp	VPSS GROUP 号。 取值范围:[0, VPSS_MAX_GRP_NUM)。	输入
type	ProcAmp(颜色控制) 种类	输入
value	ProcAmp 的设定	输入

【返回值】

返回值	描述
0	成功。
非 0	失败, 请参见错误码 。

【需求】

- 头文件：cvi_comm_vpss.h、cvi_vpss.h
- 库文件：libvpu.a

【注意】

无。

【举例】

请参见CVI_VPSS_GetGrpProcAmpCtrl

【相关主题】

- PROC_AMP_E
- PROC_AMP_CTRL_S
- CVI_VPSS_GetGrpProcAmpCtrl
- CVI_VPSS_GetGrpProcAmp

6.3.11 CVI_VPSS_SetGrpParamfromBin

【描述】

根据 Bin 设置一个 VPSS GROUP 的属性.

【语法】

```
CVI_S32 CVI_VPSS_SetGrpParamfromBin(VPSS_GRP VpssGrp, CVI_U8 scene);
```

【参数】

参数名称	描述	输入/输出
VpssGrp	VPSS GROUP 号。 取值范围:[0, VPSS_MAX_GRP_NUM)。	输入
scene	所欲套用的 VPSS 输入 Bin 场景设定	输入

【返回值】

返回值	描述
0	成功。
非 0	失败, 请参见错误码 。

【需求】

- 头文件：cvi_comm_vpss.h、cvi_vpss.h
- 库文件：libvpu.a

【注意】

- 若使用新的 PQ bin，Group 必须已经创建
- 若 PQ bin 不存在，则使用默认参数
- 目前仅支持套用 ProcAmp(Brightness, Contrast, Hue, Saturation) 的设定

【举例】

```
VPSS_GRP_ATTR_S stVpssGrpAttr;
VPSS_CHN VpssChn = VPSS_CHN0;
CVI_BOOL abChnEnable[VPSS_MAX_PHY_CHN_NUM] = {0};
VPSS_CHN_ATTR_S astVpssChnAttr[VPSS_MAX_PHY_CHN_NUM];
CVI_S32 s32Ret = CVI_SUCCESS;
VPSS_CROP_INFO_S pstCropInfo;

stVpssGrpAttr.stFrameRate.s32SrcFrameRate = -1;
stVpssGrpAttr.stFrameRate.s32DstFrameRate = -1;
stVpssGrpAttr.enPixelFormat = PIXEL_FORMAT_YUV_PLANAR_420;
stVpssGrpAttr.u32MaxW = stSize.u32Width;
stVpssGrpAttr.u32MaxH = stSize.u32Height;

astVpssChnAttr[VpssChn].u32Width = 800;
astVpssChnAttr[VpssChn].u32Height = 600;
astVpssChnAttr[VpssChn].enChnMode = VPSS_CHN_MODE_USER;
```

(下页继续)

(续上页)

```
astVpssChnAttr[VpssChn].enVideoFormat = VIDEO_FORMAT_LINEAR;
astVpssChnAttr[VpssChn].enPixelFormat = PIXEL_FORMAT_BGR_888_PLANAR;
astVpssChnAttr[VpssChn].stFrameRate.s32SrcFrameRate = 30;
astVpssChnAttr[VpssChn].stFrameRate.s32DstFrameRate = 30;
astVpssChnAttr[VpssChn].u32Depth = 0;
astVpssChnAttr[VpssChn].bMirror = CVI_FALSE;
astVpssChnAttr[VpssChn].bFlip = CVI_FALSE;
astVpssChnAttr[VpssChn].stAspectRatio.enMode = ASPECT_RATIO_NONE;

/*start vpss*/
VPSS_CHN VpssChn;
s32Ret = CVI_VPSS_CreateGrp(0, &stVpssGrpAttr);
if (s32Ret != CVI_SUCCESS) {
  SAMPLE_PRT("CVI_VPSS_CreateGrp failed with %#x!\n", s32Ret);
  return CVI_FAILURE;
}

/*vpss grp0套用场景0*/
s32Ret = CVI_VPSS_SetGrpParamfromBin(0, 0);
if (s32Ret != CVI_SUCCESS) {
  SAMPLE_PRT("CVI_VPSS_SetGrpParamfromBin failed with %#x!\n", s32Ret);
  return CVI_FAILURE;
}
```

【相关主题】

- PROC_AMP_E
- PROC_AMP_CTRL_S
- CVI_VPSS_GetGrpProcAmpCtrl
- CVI_VPSS_GetGrpProcAmp

6.3.12 CVI_VPSS_GetChnAttr

【描述】

获取 VPSS 通道属性.

【语法】

```
CVI_S32 CVI_VPSS_GetChnAttr (VPSS_GRP VpssGrp, VPSS_CHN VpssChn, VPSS_CHN_
↪ATTR_S *pstChnAttr)
```

【参数】

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

【返回值】

返回值	描述
0	成功。
非 0	失败，请参见错误码 。

【需求】

- 头文件：cvi_comm_vpss.h、cvi_vpss.h
- 库文件：libvpu.a

【注意】

Group 必须已经创建

【举例】

请参见CVI_VPSS_CreateGrp

【相关主题】

CVI_VPSS_CreateGrp

6.3.13 CVI_VPSS_SetChnAttr

【描述】

设置 VPSS 通道属性.

【语法】

```
CVI_S32 CVI_VPSS_SetChnAttr (VPSS_GRP VpssGrp, VPSS_CHN VpssChn, const VPSS_CHN_
↪ATTR_S *pstChnAttr)
```

【参数】

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

【返回值】

返回值	描述
0	成功。
非 0	失败，请参见错误码 。

【需求】

- 头文件：cvi_comm_vpss.h、cvi_vpss.h
- 库文件：libvpu.a

【注意】

Group 必须已经创建

【举例】

请参见CVI_VPSS_CreateGrp

【相关主题】

CVI_VPSS_CreateGrp

6.3.14 CVI_VPSS_EnableChn

【描述】

启用 VPSS 通道.

【语法】

```
CVI_S32 CVI_VPSS_EnableChn(VPSS_GRP VpssGrp, VPSS_CHN VpssChn);
```

【参数】

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

【返回值】

返回值	描述
0	成功。
非 0	失败, 请参见错误码 。

【需求】

- 头文件：cvi_comm_vpss.h、cvi_vpss.h
- 库文件：libvpu.a

【注意】

- Group 必须已经创建
- 多次使能返回成功

【举例】

请参见CVI_VPSS_CreateGrp

【相关主题】

CVI_VPSS_CreateGrp

6.3.15 CVI_VPSS_DisableChn

【描述】

禁用 VPSS 通道.

【语法】

```
CVI_S32 CVI_VPSS_DisableChn (VPSS_GRP VpssGrp, VPSS_CHN VpssChn)
```

【参数】

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

【返回值】

返回值	描述
0	成功。
非 0	失败, 请参见错误码 。

【需求】

- 头文件：cvi_comm_vpss.h、cvi_vpss.h
- 库文件：libvpu.a

【注意】

- Group 必须已经创建
- 重复禁用返回成功

【举例】

请参见CVI_VPSS_CreateGrp

【相关主题】

CVI_VPSS_CreateGrp

6.3.16 CVI_VPSS_SetGrpCrop

【描述】

设置 VPSS CROP 功能属性.

【语法】

```
CVI_S32 CVI_VPSS_SetGrpCrop (VPSS_GRP VpssGrp, const VPSS_CROP_INFO_S
↪*pstCropInfo);
```

【参数】

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

【返回值】

返回值	描述
0	成功。
非 0	失败, 请参见错误码 。

【需求】

- 头文件：cvi_comm_vpss.h、cvi_vpss.h
- 库文件：libvpu.a

【注意】

- Group 必须已经创建
- CROP 区域尺寸不能小于 VPSS 最小尺寸，不能超过最大尺寸，否则返回失败
- Online 不支持 group 裁剪
- 开启 DIS 情况下，本设置无效

【举例】

请参见CVI_VPSS_CreateGrp

【相关主题】

CVI_VPSS_CreateGrp

6.3.17 CVI_VPSS_GetGrpCrop

【描述】

获取 VPSS CROP 功能属性.

【语法】

```
CVI_S32 CVI_VPSS_GetGrpCrop (VPSS_GRP VpssGrp, VPSS_CROP_INFO_S *pstCropInfo);
```

【参数】

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

【返回值】

返回值	描述
0	成功。
非 0	失败, 请参见错误码 。

【需求】

- 头文件：cvi_comm_vpss.h、cvi_vpss.h
- 库文件：libvpu.a

【注意】

Group 必须已经创建

【举例】

请参见CVI_VPSS_CreateGrp

【相关主题】

CVI_VPSS_CreateGrp

6.3.18 CVI_VPSS_SendFrame

【描述】

用户向 VPSS 发送数据

【语法】

```
CVI_S32 CVI_VPSS_SendFrame (VPSS_GRP VpssGrp, const VIDEO_FRAME_INFO_S␣
↪*pstVideoFrame,
```

CVI_S32 s32MilliSec);

【参数】

参数名称	描述	输入/输出
VpssGrp	VPSS GROUP 号。 取值范围:[0, VPSS_MAX_GRP_NUM)。	输入
pstVideoFrame	待发送的图像信息。具体描述请参见系统控制章节。	输入
s32MilliSec	目前版本暂未用到。	输入

【返回值】

返回值	描述
0	成功。
非 0	失败, 请参见错误码 。

【需求】

- 头文件：cvi_comm_vpss.h、cvi_vpss.h
- 库文件：libvpu.a

【注意】

- Group 必须已经创建并且已经启动，不然会返回失败

【举例】

无

【相关主题】

无

6.3.19 CVI_VPSS_GetChnFrame

【描述】

用户从通道获取一帧处理完成的图像

【语法】

```
CVI_S32 CVI_VPSS_GetChnFrame(VPSS_GRP VpssGrp, VPSS_CHN VpssChn, VIDEO_
↪FRAME_INFO_S *pstFrameInfo, CVI_S32 s32MilliSec);
```

【参数】

参数名称	描述	输入/输出
VpssGrp	VPSS GROUP 号。 取值范围:[0, VPSS_MAX_GRP_NUM)。	输入
VpssChn	VPSS 通道号 取值范围:[0, VPSS_MAX_CHN_NUM)。	输入
pstVideoFrame	待发送的图像信息。具体描述请参见“系统控制”章节。	输出
s32MilliSec	超时时间。	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，请参见错误码 。

【需求】

- 头文件：cvi_comm_vpss.h、cvi_vpss.h
- 库文件：libvpu.a

【注意】

- Group 必须已经创建
- 该函数将获得的图像信息保存在 pstvideoFrame 里面，并包含了图像的虚拟地址和物理地址

【举例】

```
VIDEO_FRAME_INFO_S pstVideoFrame;
FILE *fp;
size_t image_size;
unsigned char ptr[image_size];
int count = 0;

CVI_VPSS_GetChnFrame(0, 0, &pstVideoFrame, 5000);
image_size = pstVideoFrame.stVFrame.u32Width * pstVideoFrame.stVFrame.u32Height * 3;
pstVideoFrame.
stVFrame.u64VirAddr[i] = CVI_SYS_Mmap(pstVideoFrame -> stVFrame .u64PhyAddr[0], image_
↪size);
for (int i = 0; i < 3; i++) {
  memcpy(&ptr[count], (const CVI_VOID *)pstVideoFrame.stVFrame.u64VirAddr[i], pstVideoFrame.
↪stVFrame.u32Length[i]);
  count += pstVideoFrame.stVFrame.u32Length[i];
}
fp = fopen("/tmp/dump.bin", "w");
if (fp == CVI_NULL) {
  CVI_TRACE_VPSS(CVI_DBG_ERR, "open data file error\n");
  return CVI_FAILURE;
}
fwrite(ptr, image_size, 1, fp);
fclose(fp);
CVI_SYS_Munmap(pstVideoFrame.
```

(下页继续)

(续上页)

```
stVFrame.u64VirAddr[i], image_size);
if (CVI_VPSS_ReleaseChnFrame(0, 0, &pstVideoFrame) != 0)
  SAMPLE_PRT("CVI_VI_ReleaseChnFrame NG\n");
```

【相关主题】

CVI_VPSS_ReleaseChnFrame

6.3.20 CVI_VPSS_SendChnFrame

【描述】

用户向 VPSS 指定通道数据。可藉此来达到画面拼接。

【语法】

```
CVI_S32 CVI_VPSS_SendChnFrame (VPSS_GRP VpssGrp, VPSS_CHN VpssChn, const VIDEO_
↪FRAME_INFO_S *pstVideoFrame, CVI_S32 s32MilliSec);
```

【参数】

参数名称	描述	输入/输出
VpssGrp	VPSS GROUP 号。 取值范围:[0, VPSS_MAX_GRP_NUM)。	输入
VpssChn	VPSS 通道号 取值范围:[0, VPSS_MAX_CHN_NUM)	输入
pstVideoFrame	待发送的图像信息。具体描述请参见“系统控制”章节	输入
s32MilliSec	目前版本暂未用到	输入

【返回值】

返回值	描述
0	成功。
非 0	失败, 请参见错误码 。

【需求】

- 头文件：cvi_comm_vpss.h、cvi_vpss.h
- 库文件：libvpu.a

【注意】

- Group/Chn 必须已经创建

【举例】

无

【相关主题】

无

6.3.21 CVI_VPSS_ReleaseChnFrame

【描述】

用户释放一帧通道处理完成的图像

【语法】

```
CVI_S32 CVI_VPSS_ReleaseChnFrame (VPSS_GRP VpssGrp, VPSS_CHN VpssChn, VIDEO_
↪FRAME_INFO_S *pstFrameInfo);
```

【参数】

参数名称	描述	输入/输出
VpssGrp	VPSS GROUP 号。 取值范围:[0, VPSS_MAX_GRP_NUM)。	输入
VpssChn	VPSS 通道号 取值范围:[0, VPSS_MAX_CHN_NUM)。	输入
pstVideoFrame	待发送的图像信息。具体描述请参见“系统控制”章节。	输入

【返回值】

返回值	描述
0	成功。
非 0	失败, 请参见错误码 。

【需求】

- 头文件：cvi_comm_vpss.h、cvi_vpss.h
- 库文件：libvpu.a

【注意】

此接口要与CVI_VPSS_GetChnFrame 配合使用

【举例】

请参见CVI_VPSS_GetChnFrame

【相关主题】

CVI_VPSS_GetChnFrame

6.3.22 CVI_VPSS_GetGrpFrame

【描述】

目前该函数暂未实作

6.3.23 CVI_VPSS_ReleaseGrpFrame

【描述】

目前该函数暂未实作

6.3.24 CVI_VPSS_SetChnCrop

【描述】

设置 VPSS 信道裁剪功能属性

【语法】

```
CVI_S32 CVI_VPSS_SetChnCrop(VPSS_GRP VpssGrp, VPSS_CHN VpssChn, const VPSS_
↪CROP_INFO_S *pstCropInfo)
```

【参数】

参数名称	描述	输入/输出
VpssGrp	VPSS GROUP 号。 取值范围:[0, VPSS_MAX_GRP_NUM)。	输入
VpssChn	VPSS 通道号 取值范围:[0, VPSS_MAX_CHN_NUM)。	输入/输出
pstCropInfo	待发送的图像信息。具体描述请参见“系统控制”章节。	输入

【返回值】

返回值	描述
0	成功。
非 0	失败, 请参见错误码 。

【需求】

- 头文件：cvi_comm_vpss.h、cvi_vpss.h
- 库文件：libvpu.a

【注意】

- Group 必须已创建

- CROP 区域尺寸不能小于 VPSS 最小尺寸，不能超过最大尺寸，否则返回失败
- 该函数设置的是 vpss 输出通道的图像，而 GrpCrop 设置的是 vpss 输入的图像

【举例】

无

【相关主题】

CVI_VPSS_GetChnCrop

6.3.25 CVI_VPSS_GetChnCrop

【描述】

获取 VPSS 信道裁剪功能属性

【语法】

```
CVI_S32 CVI_VPSS_GetChnCrop(VPSS_GRP VpssGrp, VPSS_CHN VpssChn, VPSS_CROP_
↪INFO_S *pstCropInfo)
```

【参数】

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

【返回值】

返回值	描述
0	成功。
非 0	失败，请参见错误码 。

【需求】

- 头文件：cvi_comm_vpss.h、cvi_vpss.h
- 库文件：libvpu.a

【注意】

- Group 必须已创建
- 该函数获得的是 vpss 输出通道的图像裁剪信息，而CVI_VPSS_GetGrpCrop 获得的是 vpss 输入的图像裁剪信息

【举例】

无

【相关主题】

CVI_VPSS_SetChnCrop

6.3.26 CVI_VPSS_SetChnRotation

【描述】

设置 VPSS 通道旋转的属性。

【语法】

```
CVI_S32 CVI_VPSS_SetChnRotation(VPSS_GRP VpssGrp, VPSS_CHN VpssChn, ROTATION_E
↪enRotation);
```

【参数】

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

【返回值】

返回值	描述
0	成功。
非 0	失败, 请参见错误码 。

【需求】

- 头文件：cvi_vpss.h, cvi_comm_vpss.h
- 库文件：libvpu.a

【注意】

- 使用本接口前，需先调用CVI_VPSS_SetChnAttr ，否则提示失败。
- 必需在设置信道属性后才能设置此属性。
- CV181x/CV180x 不支持 180° 旋转，180° 旋转可以利用 mirror+flip 实现。
- 仅支持 NV12、NV21 及 YUV400 三种格式的旋转。
- 旋转后，信道输出的图像大小可能发生变化。
  例如 1920x1080 输入的图像，旋转 90 度后，实际输出为 1088x1920。

因为旋转需要 64 像素对齐，所以会产生无效区域，如果后端对接 vo 或者 venc，后端模块会自动裁剪有效部分，如果是 GetChnFrame, 有效区域在 VIDEO_FRAME_S 结构体 s16OffsetTop、s16OffsetBottom、s16OffsetLeft、s16OffsetRight 指定。

【举例】

无

【相关主题】

无

6.3.27 CVI_VPSS_GetChnRotation

【描述】

获取 VPSS 通道旋转的属性。

【语法】

```
CVI_S32 CVI_VPSS_GetChnRotation(VPSS_GRP VpssGrp, VPSS_CHN VpssChn, ROTATION_
↪E *penRotation);
```

【参数】

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

【返回值】

返回值	描述
0	成功。
非 0	失败, 请参见错误码 。

【需求】

- 头文件：cvi_vpss.h, cvi_comm_vpss.h
- 库文件：libvpu.a

【注意】

- 使用本接口前，需先调用CVI_VPSS_SetChnAttr ，否则提示失败。

【举例】

无

【相关主题】

CVI_VPSS_SetChnRotation

6.3.28 CVI_VPSS_SetChnLDCAttr

【描述】

设置 VPSS 通道镜头畸变矫正的属性。

【语法】

```
CVI_S32 CVI_VPSS_SetChnLDCAttr(VPSS_GRP VpssGrp, VPSS_CHN VpssChn, const VPSS_
↪LDC_ATTR_S *pstLDCAttr);
```

【参数】

参数名称	描述	输入/输出
VpssGrp	VPSS GROUP 号。 取值范围:[0, VPSS_MAX_GRP_NUM)。	输入
VpssChn	VPSS 通道号 取值范围:[0, VPSS_MAX_CHN_NUM)	输入
pstLDCAttr	镜头畸变矫正属性。	输入

【返回值】

返回值	描述
0	成功。
非 0	失败, 请参见错误码 。

【需求】

- 头文件：cvi_vpss.h, cvi_comm_vpss.h
- 库文件：libvpu.a

【注意】

- 使用本接口前，需先调用CVI_VPSS_SetChnAttr ，否则提示失败。
- 必需在设置信道属性后才能设置此属性。
- 仅支持 NV21 及 YUV400 两种格式的旋转。

【举例】

无

【相关主题】

CVI_VPSS_GetChnLDCAttr

VPSS_LDC_ATTR_S

6.3.29 CVI_VPSS_GetChnLDCAttr

【描述】

获取 VPSS 通道镜头畸变矫正的属性。

【语法】

```
CVI_S32 CVI_VPSS_GetChnLDCAttr(VPSS_GRP VpssGrp, VPSS_CHN VpssChn, VPSS_LDC_
↪ATTR_S *pstLDCAttr);
```

【参数】

参数名称	描述	输入/输出
VpssGrp	VPSS GROUP 号。 取值范围:[0, VPSS_MAX_GRP_NUM)。	输入
VpssChn	VPSS 通道号 取值范围:[0, VPSS_MAX_CHN_NUM)	输入
pstLDCAttr	镜头畸变矫正属性。	输出

【返回值】

返回值	描述
0	成功。
非 0	失败, 请参见错误码 。

【需求】

- 头文件：cvi_vpss.h, cvi_comm_vpss.h
- 库文件：libvpu.a

【注意】

- 使用本接口前，需先调用CVI_VPSS_SetChnAttr ，否则提示失败。

【举例】

无

【相关主题】

CVI_VPSS_SetChnLDCAttr

VPSS_LDC_ATTR_S

6.3.30 CVI_VPSS_GetChnFd

【描述】

获取 VPSS 通道对应的设备文件句柄。

【语法】

```
CVI_S32 CVI_VPSS_GetChnFd(VPSS_GRP VpssGrp, VPSS_CHN VpssChn);
```

【参数】

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

【返回值】

返回值	描述
正数值	成功。
非正数值	无效值。

【需求】

- 头文件：cvi_vpss.h, cvi_comm_vpss.h
- 库文件：libvpu.a

【注意】

无

【举例】

无

【相关主题】

无

6.3.31 CVI_VPSS_CloseFd

【描述】

关闭 VPSS 设备通道的文件句柄。

【语法】

```
CVI_S32 CVI_VPSS_CloseFd(CVI_VOID);
```

【参数】

无

【返回值】

返回值	描述
0	成功。
非 0	失败，请参见错误码 。

【需求】

- 头文件：cvi_vpss.h, cvi_comm_vpss.h
- 库文件：libvpu.a

【注意】

此接口调用后，VPSS 其他 MMF 接口失效。

【举例】

无

【相关主题】

CVI_VPSS_GetChnFd

6.3.32 CVI_VPSS_AttachVbPool

【描述】

将 VPSS 的通道绑定到某个视频缓存 VB 池中。

【语法】

```
CVI_S32 CVI_VPSS_AttachVbPool(VPSS_GRP VpssGrp, VPSS_CHN VpssChn, VB_POOL␣
↪hVbPool);
```

【参数】

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

【返回值】

返回值	描述
0	成功。
非 0	失败，请参见错误码 。

【需求】

- 头文件：cvi_vpss.h, cvi_comm_vpss.h
- 库文件：libvpu.a

【注意】

- Group 必须已经创建
- 重复调用该函数设置同一个组返回成功
- 当要切换当前组绑定的 VB 池时，只需再调一次接口 CVI_VPSS_AttachVbPool
- 正确配置需要绑定到的 VB 池即可。
- hVbPool 必须保证是已创建 VB 池的有效 PoolId。
- 在调用 CVI_VPSS_DetachVbPool 后，销毁创建的 VB 之前，
  需要保证 VB 没有被 VPSS 后端绑定的模块使用，
  可以通过 sleep 或清除后端模块通道缓存的方式先把 VB 都释放，再销毁缓存 VB 池。
- VB 大小根据 VPSS 通道输出图像计算，具体计算公式参考 cvi_buffer.h。

【举例】

无

【相关主题】

6.3.33 CVI_VPSS_DetachVbPool

【描述】

将 VPSS 的通道从某个视频缓存 VB 池中解绑定。

【语法】

```
CVI_S32 CVI_VPSS_DetachVbPool(VPSS_GRP VpssGrp, VPSS_CHN VpssChn);
```

【参数】

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

【返回值】

返回值	描述
0	成功。
非 0	失败，请参见错误码 。

【需求】

- 头文件：cvi_vpss.h, cvi_comm_vpss.h
- 库文件：libvpu.a

【注意】

- Group 必须已经创建

【举例】

无

【相关主题】

6.3.34 CVI_VPSS_SetChnBufWrapAttr

【描述】

设置低延时卷绕属性。

【语法】

```
CVI_S32 CVI_VPSS_SetChnBufWrapAttr(VPSS_GRP VpssGrp, VPSS_CHN VpssChn, const␣
↪VPSS_CHN_BUF_WRAP_S *pstVpssChnBufWrap);
```

【参数】

参数名称	描述	输入/输出
VpssGrp	VPSS GROUP 号。 取值范围:[0, VPSS_MAX_GRP_NUM)。	输入
VpssChn	VPSS 通道号 取值范围:[0, VPSS_MAX_CHN_NUM)	输入
pstVpssChnBufWrap	通道 buffer 卷绕属性结构体。	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，请参见错误码 。

【需求】

- 头文件：cvi_vpss.h, cvi_comm_vpss.h
- 库文件：libvpu.a

【注意】

- Group 必须已经创建
- 通道使能状态下不支持。
- 必须在已设置通道属性的前提下才能设置此接口。

- 不支持 semi-planar422 图像，不支持 Flip。
- 信道低延时卷绕开启时，通道的获取、亮度、旋转、CoverEx、OverlayEx、任意角度旋转、LDC 将无效
- 在低延时卷绕通道绑编码场景下，编码不支持软件抽帧。

【举例】

无

【相关主题】

6.3.35 CVI_VPSS_GetChnBufWrapAttr

【描述】

获取低延时卷绕属性。

【语法】

```
CVI_S32 CVI_VPSS_GetChnBufWrapAttr(VPSS_GRP VpssGrp, VPSS_CHN VpssChn, VPSS_
↪CHN_BUF_WRAP_S *pstVpssChnBufWrap);
```

【参数】

参数名称	描述	输入/输出
VpssGrp	VPSS GROUP 号。 取值范围:[0, VPSS_MAX_GRP_NUM)。	输入
VpssChn	VPSS 通道号 取值范围:[0, VPSS_MAX_CHN_NUM)。	输入
pstVpssChnBufWrap	通道 buffer 卷绕属性结构体。	输出

【返回值】

返回值	描述
0	成功。
非 0	失败，请参见错误码 。

【需求】

- 头文件：cvi_vpss.h, cvi_comm_vpss.h
- 库文件：libvpu.a

【注意】

- Group 必须已经创建

【举例】

无

【相关主题】

6.3.36 CVI_VPSS_GetWrapBufferSize

【描述】

获取低延时卷绕所需缓存大小。

【语法】

```
CVI_U32 CVI_VPSS_GetWrapBufferSize(CVI_U32 u32Width, CVI_U32 u32Height, PIXEL_
↪FORMAT_E enPixelFormat, CVI_U32 u32BufLine, CVI_U32 u32BufDepth);
```

【参数】

参数名称	描述	输入/输出
u32Width	图像宽度	输入
u32Height	图像高度	输入
enPixelFormat	图像像素格式	输入
u32BufLine	单个卷绕缓存 line 数	输入
u32BufDepth	卷绕缓存个数	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，请参见错误码 。

【需求】

- 头文件：cvi_vpss.h, cvi_comm_vpss.h
- 库文件：libvpu.a

【注意】

无

【举例】

无

【相关主题】

无

6.4 数据类型

VPSS 模块相关数据类型定义如下:

- VPSS_MAX_GRP_NUM: 定义 VPSS GROUP 的最大个数。
- VPSS_MAX_CHN_NUM: 定义 VPSS 通道的最大个数。
- VPSS_MAX_PHY_CHN_NUM: 定义 VPSS 物理通道的最大个数。
- VPSS_MIN_IMAGE_WIDTH: 定义 VPSS 图像的的最小宽度。
- VPSS_MIN_IMAGE_HEIGHT: 定义 VPSS 图像的的最小高度。
- VPSS_MAX_IMAGE_WIDTH: 定义 VPSS 图像的的最大宽度。
- VPSS_MAX_IMAGE_HEIGHT: 定义 VPSS 图像的的最大高度
- VPSS_MAX_ZOOMIN: 定义 VPSS 物理通道最大放大倍数。
- VPSS_MAX_ZOOMOUT: 定义 VPSS 物理通道的最大缩小倍数。
- VPSS_GRP: 定义 VPSS 组号。
- VPSS_CHN: 定义 VPSS 通道号。
- VPSS_CROP_COORDINATE_E: 定义 CROP 起点坐标模式。
- VPSS_CROP_INFO_S: 定义 CROP 功能所需信息。
- VPSS_GRP_ATTR_S: 定义 VPSS GROUP 属性。
- VPSS_CHN_ATTR_S: 定义 VPSS 物理通道属性。
- VPSS_MOD_PARAM_S: 通过接口设置模块参数。
- VPSS_CHN_BUF_WRAP_S: 定义 VPSS Buffer 卷绕属性。

6.4.1 VPSS_MAX_GRP_NUM

【说明】

定义 VPSS GROUP 的最大个数

【定义】

```
#define VPSS_MAX_GRP_NUM 16
```

【注意事项】

无

【相关数据类型及接口】

无

6.4.2 VPSS_MAX_CHN_NUM

【说明】

定义 VPSS 通道的最大个数

【定义】

```
#define VPSS_MAX_CHN_NUM  VPSS_MAX_PHY_CHN_NUM
```

【注意事项】

VPSS 最大信道数为物理信道数

【相关数据类型及接口】

无

6.4.3 VPSS_MAX_PHY_CHN_NUM

【说明】

定义 VPSS 物理通道的最大个数

【定义】

- cv181x

```
#define VPSS_MAX_PHY_CHN_NUM    4
```

- cv180x

```
#define VPSS_MAX_PHY_CHN_NUM    3
```

【注意事项】

无

【相关数据类型及接口】

无

6.4.4 VPSS_MIN_IMAGE_WIDTH

【说明】

定义 VPSS 图像的最小宽度

【定义】

```
#define VPSS_MIN_IMAGE_WIDTH 32
```

【注意事项】

无

【相关数据类型及接口】

无

6.4.5 VPSS_MIN_IMAGE_HEIGHT

【说明】

定义 VPSS 图像的最小高度

【定义】

```
#define VPSS_MIN_IMAGE_HEIGHT 32
```

【注意事项】

无

【相关数据类型及接口】

无

6.4.6 VPSS_MAX_IMAGE_WIDTH

【说明】

定义 VPSS 图像的最大宽度

【定义】

```
#define VPSS_MAX_IMAGE_WIDTH 2880
```

【注意事项】

无

【相关数据类型及接口】

无

6.4.7 VPSS_MAX_IMAGE_HEIGHT

【说明】

定义 VPSS 图像的最大高度

【定义】

```
#define VPSS_MAX_IMAGE_HEIGHT 2880
```

【注意事项】

无

【相关数据类型及接口】

无

6.4.8 VPSS_MAX_ZOOMIN

【说明】

定义 VPSS 物理通道最大放大倍数

【定义】

```
#define VPSS_MAX_ZOOMIN  32
```

【注意事项】

无

【相关数据类型及接口】

无

6.4.9 VPSS_MAX_ZOOMOUT

【说明】

定义 VPSS 物理通道的最大缩小倍数

【定义】

```
#define VPSS_MAX_ZOOMOUT   32
```

【注意事项】

无

【相关数据类型及接口】

无

6.4.10 VPSS_GRP

【说明】

定义 VPSS 组号。

【定义】

```
typedef CVI_U32 VPSS_GRP;
```

【注意事项】

无

【相关数据类型及接口】

无

6.4.11 VPSS_CHN

【说明】

定义 VPSS 组的管道号。

【定义】

```
typedef CVI_U32 VPSS_CHN;
```

【注意事项】

无

【相关数据类型及接口】

无

6.4.12 VPSS_ROUNDING_E

【说明】

定义 Normalize 时 rounding 的模式。

【定义】

```
typedef enum _VPSS_ROUNDING_E {
  VPSS_ROUNDING_TO_EVEN = 0,
  VPSS_ROUNDING_AWAY_FROM_ZERO,
  VPSS_ROUNDING_TRUNCATE,
  VPSS_ROUNDING_MAX,
} VPSS_ROUNDING_E;
```

【成员】

成员名称	描述
VPSS_ROUNDING_TO_EVEN	四舍五入，参考下表。
VPSS_ROUNDING_AWAY_FROM_ZERO	四舍五入，参考下表。
VPSS_ROUNDING_TRUNCATE	无条件舍去，参考下表。

【注意事项】

	TO_EVEN	AWAY_FROM_ZERO	TRUNCATE
+1.8	+2	+2	+1
+1.5			
+1.2	+1	+1	
+0.8			0
+0.5	0		
+0.2		0	
F0.2			
F0.5		F1	
F0.8	F1		
F1.2			F1
F1.5	F2	F2	
F1.8			

【相关数据类型及接口】

无

6.4.13 VPSS_CROP_COORDINATE_E

【说明】

定义 CROP 起点坐标的模式。

【定义】

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

【成员】

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

【注意事项】

VPSS_CROP_RATIO_COOR 当前版本暂不支持。

【相关数据类型及接口】

VPSS_CROP_INFO_S

6.4.14 VPSS_NORMALIZE_S

【说明】

定义 Normalize 功能所需信息。

【定义】

```
typedef struct _VPSS_NORMALIZE_S {
  CVI_BOOL bEnable;
  CVI_FLOAT factor[3];
  CVI_FLOAT mean[3];
  VPSS_ROUNDING_E rounding;
} VPSS_NORMALIZE_S;
```

【成员】

成员名称	描述
bEnable	Normalize 使能开关。
factor	正规化因子。1/8192 ~ 8191/8192 or 1/4096 ~ 8191/4096
mean	正规化方均根差。0 ~ 255
rounding	对小数部分的处理方式。

【注意事项】

1. 启用 Normalize 后的 Output 为 int8, -128 ~ 127
2. Normalize 等效 OpenCV 中的 convertTo()

   cv::convertTo(image, CV_8S, factor, -mean);
3. 根据 VPSS CHN 的 format 排列，结果会有所差异。
   若是 PIXEL_FORMAT_RGB_888_PLANAR/PIXEL_FORMAT_RGB_888 运作方式如下:
   (Pixel Value R) * factor[0] –mean[0] 在接着做 rounding。
   (Pixel Value G) * factor[1] –mean[1] 在接着做 rounding。
   (Pixel Value B) * factor[2] –mean[2] 在接着做 rounding。
   若是 PIXEL_FORMAT_BGR_888 运作方式如下:
   (Pixel Value B) * factor[0] –mean[0] 在接着做 rounding。
   (Pixel Value G) * factor[1] –mean[1] 在接着做 rounding。
   (Pixel Value R) * factor[2] –mean[2] 在接着做 rounding。

【相关数据类型及接口】

VPSS_ROUNDING_E

6.4.15 VPSS_CROP_INFO_S

【说明】

定义 CROP 功能所需信息。

【定义】

```
typedef struct _VPSS_CROP_INFO_S {
  CVI_BOOL bEnable;
  VPSS_CROP_COORDINATE_E enCropCoordinate;
  RECT_S stCropRect;
} VPSS_CROP_INFO_S;
```

【成员】

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

【注意事项】

无

【相关数据类型及接口】

VPSS_CROP_COORDINATE_E

6.4.16 VPSS_GRP_ATTR_S

【说明】

定义 VPSS GROUP 属性。

【定义】

```
typedef struct _VPSS_GRP_ATTR_S {
  CVI_U32 u32MaxW;
  CVI_U32 u32MaxH;
  PIXEL_FORMAT_E enPixelFormat;
  FRAME_RATE_CTRL_S stFrameRate;
  CVI_U8 u8VpssDev;
} VPSS_GRP_ATTR_S;
```

【成员】

成员名称	描述
u32MaxW	输入图像宽度。
u32MaxH	输入图像高度。
enPixelFormat	输入图像像素格式。
stFrameRate	组帧率。
u8VpssDev	可指定此 VPSS 组要使用哪个硬件工作。

【注意事项】

u8VpssDev 需在 CVI_SYS_SetVPSSMode() 设定为 VPSS_MODE_DUAL 或 VPSS_MODE_RGNEX 后，设定 0/1 才指定不同硬件。

【相关数据类型及接口】

- PIXEL_FORMAT_E
- CVI_VPSS_CreateGrp
- CVI_VPSS_SetGrpAttr
- CVI_VPSS_GetGrpAttr

6.4.17 VPSS_CHN_ATTR_S

【说明】

定义 VPSS 物理通道属性。

【定义】

```
typedef struct _VPSS_CHN_ATTR_S {
  CVI_U32 u32Width;
  CVI_U32 u32Height;
  VIDEO_FORMAT_E enVideoFormat;
  PIXEL_FORMAT_E enPixelFormat;
  FRAME_RATE_CTRL_S stFrameRate;
  CVI_BOOL bMirror;
  CVI_BOOL bFlip;
  CVI_U32 u32Depth;
  ASPECT_RATIO_S stAspectRatio;
  VPSS_NORMALIZE_S stNormalize;
} VPSS_CHN_ATTR_S;
```

【成员】

成员名称	描述
u32Width	目标图像宽度
u32Height	目标图像高度
enVideoFormat	目标图像视频格式。
enPixelFormat	目标图像像素格式。
stFrameRate	帧率控制信息，当且仅当 s32DstFrameRate 小于 s32SrcFrameRate 时按比例控制帧率。例如，若 s32SrcFrameRate 为 60，s32DstFrameRate 为 30，则每隔一帧丢一帧。
bMirror	水平镜像使能。
bFlip	垂直翻转使能。
u32Depth	用户获取通道图像的队列长度。静态属性。
stAspectRatio	幅形比参数。
stNormalize	Normalize 以便加速后续 TPU 的运作。

【注意事项】

无

【相关数据类型及接口】

- CVI_VPSS_SetChnAttr

6.4.18 VPSS_MOD_PARAM_S

【说明】

通过接口设置模块参数。

【定义】

```
typedef struct _VPSS_PARAM_MOD_S {
  CVI_U32 u32VpssVbSource;
  CVI_U32 u32VpssSplitNodeNum;
} VPSS_MOD_PARAM_S;
```

【成员】

成员名称	描述
u32VpssVbSource	视频区块池类型。
u32VpssSplitNodeNum	分块节点数量。

【注意事项】

该结构体属性目前版本暂未使用

【相关数据类型及接口】

6.4.19 PROC_AMP_E

【说明】

定义 VPSS PROCAMP 类别。

【定义】

```
typedef enum _PROC_AMP_E {
  PROC_AMP_BRIGHTNESS = 0,
  PROC_AMP_CONTRAST,
  PROC_AMP_SATURATION,
  PROC_AMP_HUE,
  PROC_AMP_MAX,
} PROC_AMP_E;
```

【成员】

成员名称	描述
PROC_AMP_BRIGHTNESS	亮度值。
PROC_AMP_CONTRAST	对比值。
PROC_AMP_SATURATION	色调值。
PROC_AMP_HUE	饱和度值。

【注意事项】

【相关数据类型及接口】

- PROC_AMP_CTRL_S

6.4.20 PROC_AMP_CTRL_S

【说明】

定义 VPSS PROCAMP 属性。

【定义】

```
typedef struct _PROC_AMP_CTRL_S {
  CVI_S32 minimum;
  CVI_S32 maximum;
  CVI_S32 step;
  CVI_S32 default_value;
} PROC_AMP_CTRL_S;
```

【成员】

成员名称	描述
minimum	此颜色控制功能的最小值。
maximum	此颜色控制功能的最大值。
step	此颜色控制功能的有效调整值。
default_value	此颜色控制功能的默认值。

【注意事项】

建议在开始对 VPSS ProcAmp 操作之前，先获取此些属性，以免操作失败。

【相关数据类型及接口】

- PROC_AMP_E
- CVI_VPSS_GetGrpProcAmpCtrl

6.4.21 VPSS_LDC_ATTR_S

【说明】

定义 VPSS 镜头畸变矫正结构体

【定义】

```
typedef struct _VPSS_LDC_ATTR_S {
  CVI_BOOL bEnable;
  LDC_ATTR_S stAttr;
} VPSS_LDC_ATTR_S;
```

【成员】

成员名称	描述
bEnable	LDC 使能。
stAttr	LDC 设定属性。

【注意事项】

无

【相关数据类型及接口】

- LDC_ATTR_S

6.4.22 VPSS_CHN_BUF_WRAP_S

【说明】

定义 VPSS Buffer 卷绕属性

【定义】

```
typedef struct _VPSS_CHN_BUF_WRAP_S {
  CVI_BOOL bEnable;
  CVI_U32 u32BufLine;   // 64, 128
  CVI_U32 u32WrapBufferSize;
} VPSS_CHN_BUF_WRAP_S;
```

【成员】

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

【注意事项】

- u32BufLine 需小于输出图像高度。
- u32BufLine 只有 64 / 128 两种大小。

- 使能卷绕时，u32WrapBufferSize 必须大于 0，其计算可参考代码函数 CVI_VPSS_GetWrapBufferSize。

【相关数据类型及接口】

- CVI_VPSS_SetChnBufWrapAttr
- CVI_VPSS_GetChnBufWrapAttr
- CVI_VPSS_GetWrapBufferSize

6.5 错误码

视频处理子系统 API 错误码如下表所示：

错误代码	宏定义	描述
0xC0068001	CVI_ERR_VPSS_INVALID_DEVID	VPSS GROUP 号无效
0xC0068002	CVI_ERR_VPSS_INVALID_CHNID	VPSS 通道号无效
0xC0068003	CVI_ERR_VPSS_ILLEGAL_PARAM	VPSS 参数设置无效
0xC0068004	CVI_ERR_VPSS_EXIST	VPSS GROUP 已创建
0xC0068005	CVI_ERR_VPSS_UNEXIST	VPSS GROUP 未创建
0xC0068006	CVI_ERR_VPSS_NULL_PTR	输入空指针错误
0xC0068008	CVI_ERR_VPSS_NOT_SUPPORT	操作不支持
0xC0068009	CVI_ERR_VPSS_NOT_PERM	操作不允许
0xC006800c	CVI_ERR_VPSS_NOMEM	分配内存失败
0xC006800d	CVI_ERR_VPSS_NOBUF	分配 BUF 池失败
0xC006800e	CVI_ERR_VPSS_BUF_EMPTY	图像队列为空
0xC0068010	CVI_ERR_VPSS_NOTREADY	VPSS 系统未初始化
0xC0068012	CVI_ERR_VPSS_BUSY	VPSS 系统忙

7 视频编码

7.1 功能概述

7.1.1 目的

视频编码主要提供视频及时编码，将输入讯号源影像经过标准影音编码器作压缩，降低数据量，提供应用层影音服务。

目前有支持的输入源为:

- 用户在 User Mode 自行输入的影像数据
- 视频输入（VI）模块收到的影像直接发送到编码器
- 视频输入（VI）模块收到的影像经视频处理子系统（VPSS）发送到编码器

目前支持的视讯标准为:

- JPEG
- MJPEG
- H.264
- H.265

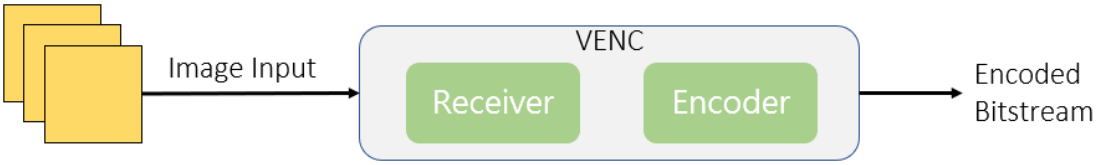
7.1.2 定义及缩写

- Encoded Bitstream 编码流
- Bitrate 比特率
- RC Rate Control 码率控制器
- Video Quality 影像质量
- QP Quantization Parameter 量化参数
- Fixed QP 固定 QP
- VBR Variable Bitrate 可变码率
- CBR Constant Bitrate 固定码率

- AVBR Adaptive Variable Bitrate 自适应可变码率
- MB Macroblock 宏块
- Frame 帧
- Frame Buffer 帧存
- Packet 包

7.2 设计概述

7.2.1 编码数据流程图



VENC 模块的输入为待编码的影像，输出为编码后的 Bitstream。VENC 含有两个子模块，Receiver 和 Encoder。

Receiver

- 接收影像的输入

Encoder

- 将收到的影像进行图片及视频编码，转换成 Encoded Bitstream，目前支持的视讯标准为
  - JPEG

    支持的影像格式为

    YUV422

    YUV420

    NV12 / NV21
  - MJPEG

    支持的影像格式和 JPEG 相同
  - H.264

    支持的影像格式为

    YUV420

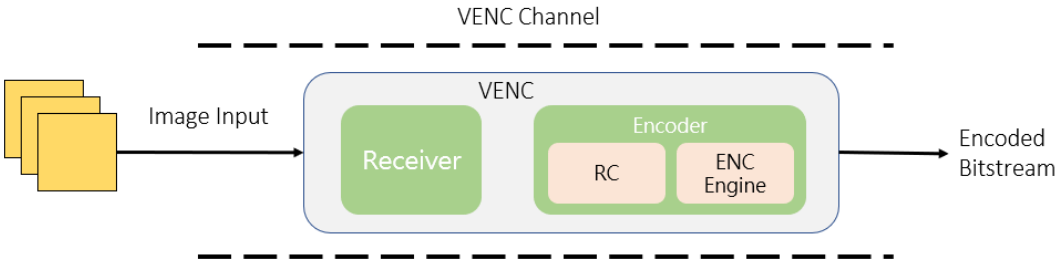
    NV12 / NV21
  - H.265

    支持的影像格式为

YUV420

NV12 / NV21

7.2.2 视频编码信道



视频编码信道为视频编码的基本操作单元，内部主要包含 VENC 相关功能，信道相关设定。系统上可以有多个视频编码信道，各信道独立运作。各信道建立后，用户可以依需求设定通道基本参数，如影像分辨率、编码 Bitrate、RC 机制等，相关的参数会对 VENC 初始化。

7.2.3 码率控制

RC 为 Encoder 中控制 Bitrate 和 Video Quality 的主要模块，主要藉由各视讯标准中的 QP 这个参数，进一步控制影像质量，当 QP 变大时，编码出来的数据量会变小，但 Video Quality 会变差；当 QP 变小时，编码出来的数据量会变大，但 Video Quality 会变好。

视频编码通常会与 RC 同时运作，如果是图片编码 (如 JPEG)，因为只有单张图片，不会需要此模块。

模式	JPEG	H.264	H.265
Fixed QP	支持	支持	支持
CBR	支持	支持	支持
VBR	不支持	支持	支持
AVBR	不支持	支持	支持

7.2.4 Fixed QP

在统计时间内，影像中的所有 MB 使用同一个 QP 值。

7.2.5 CBR

CBR (Constant Bit Rate) 固定码率，动态控制 QP，使得 Encoded Bitstream 在设定的时间区间内，维持在想要的 bitrate。

主要的参数如下：

- u32StatTime Bitrate 统计时间
  - 单位时间为秒 (second)，encoder 会在设定时间内统计 Bitrate 进而调整影像质量，符合想要的 Bitrate。
    当此值愈小时，统计区间小，影像品质变化幅度会相对比较大，
    每一张 Frame 压缩完后，对接下来的压缩有较大的影响，视讯质量比较不稳定。
    此值大时，统计区间大，每一张 Frame 压缩完后的影响程度比较小，影像变化的幅度比较小，视讯质量比较稳定。

7.2.6 VBR

VBR(Variable Bit Rate) 可变码率，允许码率在设定的统计时间内，动态调整整体 Bitrate，可使得影像质量相对稳定。

- u32MaxBitRate 最大码率
  - 设定在统计时间内，最大可使用的码率
- u32MinQp
  - 设定最小可使用的 MB QP，限制影像的最佳质量，不会使用过高的 Bitrate

7.2.7 AVBR

AVBR(Adaptive Variable Bit Rate) 自适应可变码率。依场景运动程度动态调整统计时间目标码率，码率控制内部作场景运动程度判断。场景运动程度越大，目标码率越高，而当场景偏向静止时，将自动降低目标码率。

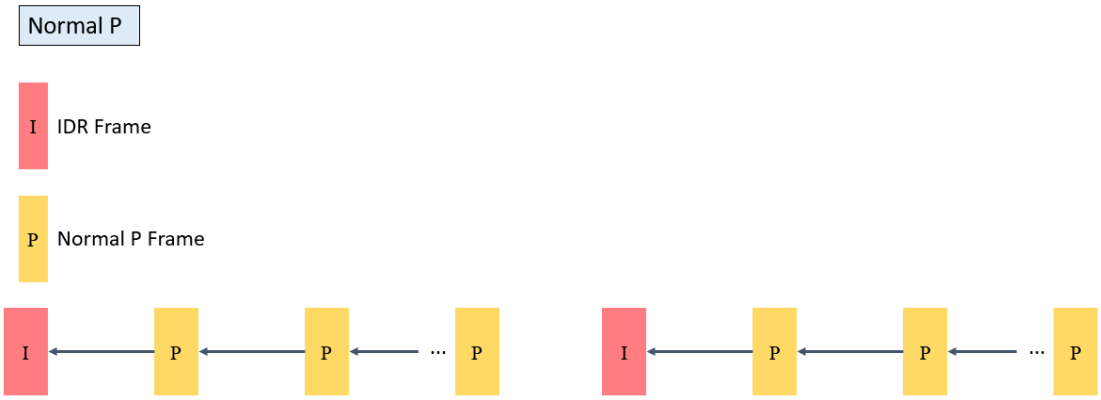
- u32MaxBitRate 最大码率
  - 设定在统计时间内，最大可使用的码率
- u32MinStillPercent
  - 场景静止状态时，目标码率与最大码率之百分比
  - MinBitrate = MaxBitrate*ChangePos*MinStillPercnet
- 根据场景运动程度，目标码率在 MinBitrate 与 MaxBitrate 间调适
- U32MaxStillQp
  - 场景完全静止时，maxQp 趋向 MaxStillQp。以保证静止场景画面质量。

7.2.8 GOP 结构

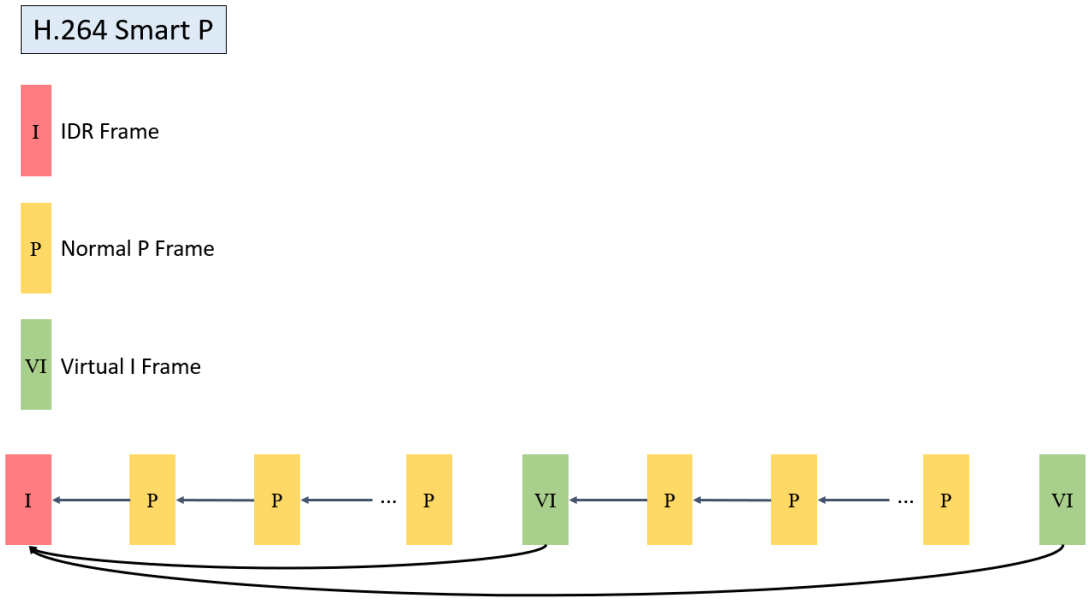
H.264 和 H.265 对于 GOP 结构支持的状况如下。

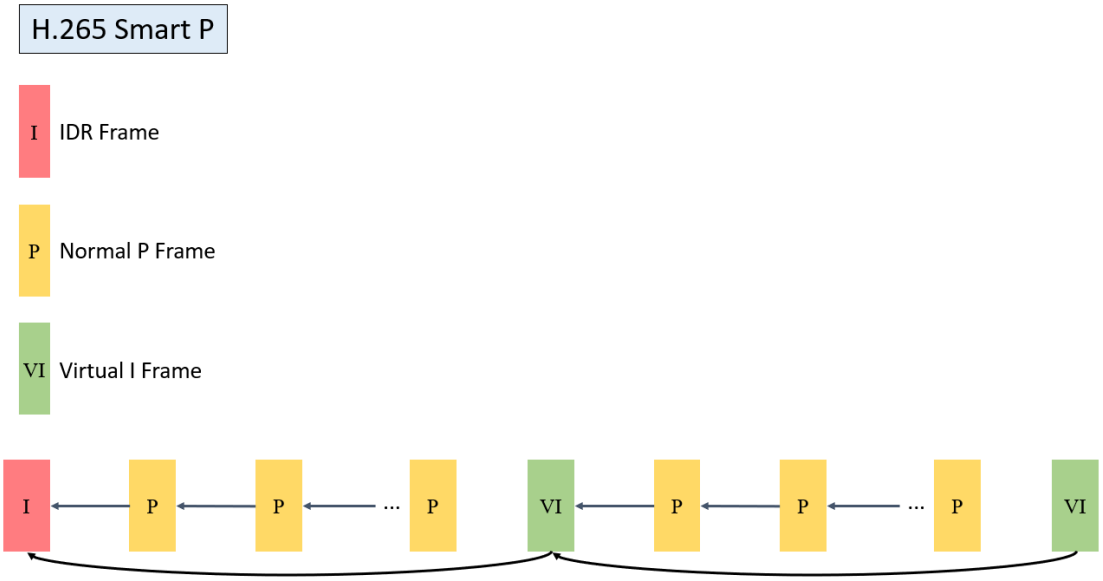
GOP 模式	H.264	H.265
NormalP	支持	支持
SmartP	支持	支持

- NormalP 的 GOP 结构如下
  - IDR 帧出现周期为 u32Gop



- SmartP 的 GOP 结构如下
  - VI 帧出现周期为 u32Gop，该帧只向前参考邻近之 IDR 帧
  - IDR 帧出现周期为 u32BgInterval，该 IDR 帧为长期参考帧

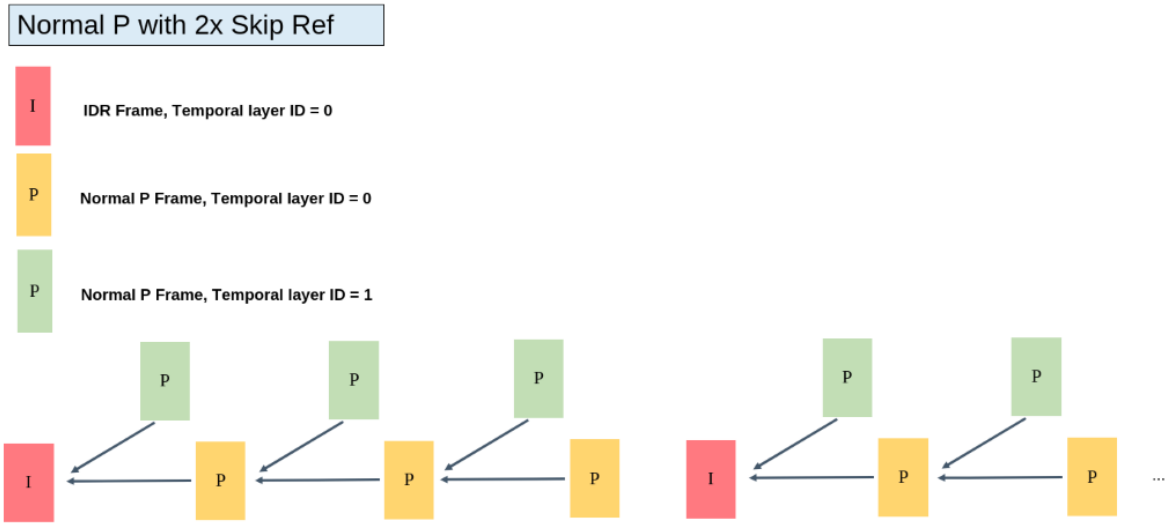




7.2.9 高级跳帧

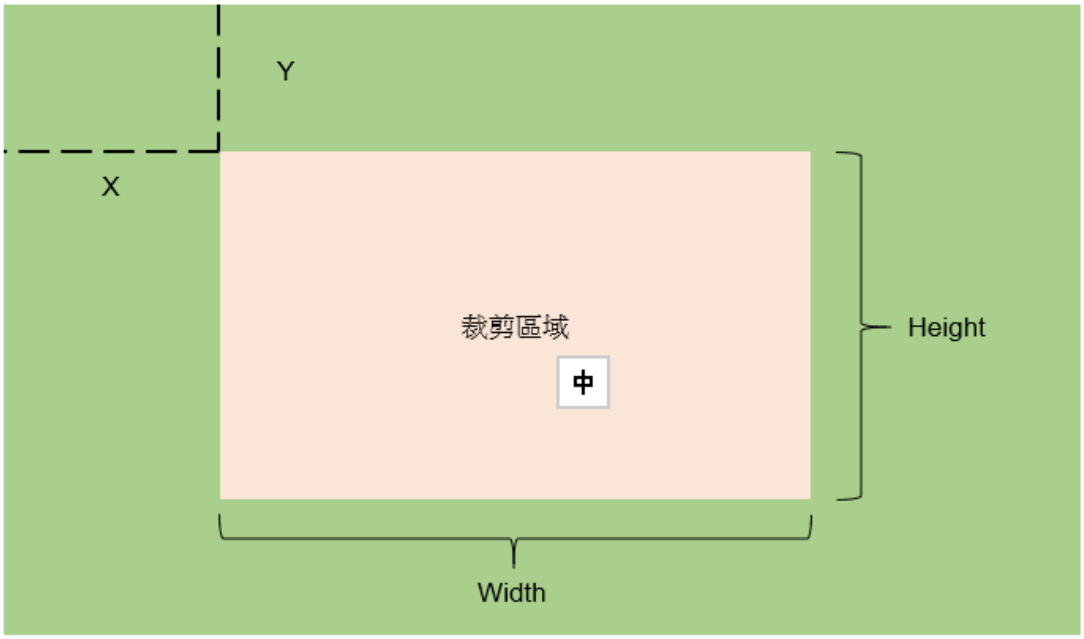
H.264 和 H.265 均支持 1 倍、2 倍、以及 4 倍跳帧参考模式。默认为 1 倍 (不跳帧) 。

· 2 倍跳帧参考模式示意如下



7.2.10 裁剪编码

对于要每一张 source frame，可以只 encode 设定的某个区域，该区域的位置用 X、Y 指定，大小仍由 width，height 指定，可参考 CVI_VENC_SetChnParam 的使用方式。



7.2.11 ROI

ROI（Region Of Interest）编码：感兴趣区域编码。

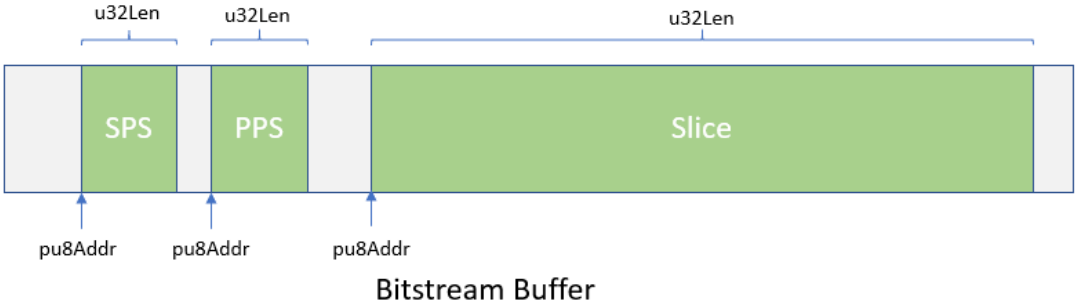
- 用户可以通过配置 ROI 区域，调适该区域的图像 Qp，实现图像中局部区域画质的差异化。
- H.264 和 H.265 均支持 8 个 ROI 设置，重复区域按照 0 ~ 7 的 ROI 索引号依次提高优先级
- ROI 区域可配置绝对 Qp 和相对 Qp 两种模式。
  - 绝对 Qp 模式：ROI 区域的 Qp 为用户设定的 Qp 值。
  - 相对 Qp 模式：ROI 区域的 Qp 为码率控制之 Qp 加上用户设定的 Qp 偏移值。
- 注意事项
  - 当码率控制模式不为 Fixed QP 模式时，ROI 区域可配置。
  - H.264 当 ROI 始能时，宏块级码率控制失效。
  - 绝对 Qp 模式因为码率控制调适宏块 QP，实际编码 QP 与设置之 QP 可能会有些差异。

7.2.12 编码码流帧配置模式

压缩后的编码码流，使用 CVI_VENC_GetStream 取得 Encoded Bitstream，相关信息如下：

- u32PackCount
  - 此张 Frame 含有多少个 NAL packet，
    例如目前压缩格式为 H.264，第一张为 I frame，此值会为 3 (含 SPS，PPS，1 Slice)，
    第二张 Frame 为 1(含 1 Slice)。

- pstPack
  - 根据 u32PackCount 的数量，会含有 pstPack[0] ~ pstPack[u32PackCount - 1] 的资料。
    如第一张 I Frame，会有 pstPack[0]，pstPack[1]，pstPack[2]
- 下图为第一张 I Frame 的 Bitstream 资料



7.2.13 多编码器并行编码

H.264 和 H.265 支持多路编码，目前默认最多为 8 路并行编码，以下为注意事项

- 编码整体效能需要低于硬件设计限制，超过会无法达到设定的 framerate
- 所需要的 Frame Buffer 会随着路数的增加成正比增加，DRAM 使用量亦随之增加

7.2.14 编码帧存计算

编码帧存（参考帧和重构帧）分配支持两种方式：PrivateVB 池方式和 UserVB 池方式。编码 PrivateVB 池方式：创建编码通道时由 VENC 创建私有 VB 池作为该通道的参考帧和重构帧 Buffer。编码 UserVB 池方式（仅 H.264/H.265 编码支持）：创建编码通道时不分配参考帧和重构帧 Buffer，而是由用户调用接口 CVI_VB_CreatePool

创建一个视频缓存 VB 池，再通过调用接口 CVI_VENC_AttachVbPool 把某个编码通道绑定到固定的视频缓存 VB 池中。两种方式可通过调用 CVI_VENC_SetModParam 接口设置相应的模块参数来选择。模块参数设置为 VB_SOURCE_PRIVATE 表示使用编码 PrivateVB 池方式；设置为 VB_SOURCE_USER 表示使用编码 UserVB 池方式。

UserVB 模式下，编码帧消耗内存最大值参考如下

	H.264	H.265
FrameSize	(align(width,32)) x (align(height,32)) x 1.5 x 2	align((width x height x 1.5), 4096) x 2

7.3 API 参考

视频编码模块主要提供视频编码通道的创建和销毁、视频编码通道的复位、开启和停止接收图像、设置和获取编码通道属性、获取和释放码流等功能。

提供以下 API:

- CVI_VENC_CreateChn：创建编码通道
- CVI_VENC_DestroyChn：销毁编码通道
- CVI_VENC_ResetChn：复位编码通道
- CVI_VENC_StartRecvFrame：开启编码通道接收输入图像
- CVI_VENC_StopRecvFrame：停止编码通道接收输入图像
- CVI_VENC_QueryStatus：查询编码通道状态
- CVI_VENC_SetChnAttr：设置编码通道的编码属性
- CVI_VENC_GetChnAttr：获取编码通道的编码属性
- CVI_VENC_GetStream：获取编码码流
- CVI_VENC_ReleaseStream：释放码流缓存
- CVI_VENC_SendFrame：支持用户发送原始图像进行编码
- CVI_VENC_GetFd：获取编码通道对应的设备文件句柄
- CVI_VENC_CloseFd：获取编码通道对应的设备文件句柄
- CVI_VENC_SetJpegParam：设置 JPEG 编码的参数集合
- CVI_VENC_GetJpegParam：获取 JPEG 编码的参数集合
- CVI_VENC_SetRcParam：设置通道码率控制高级参数
- CVI_VENC_GetRcParam：获取通道码率控制高级参数
- CVI_VENC_SetChnParam：设置 Venc 通道参数
- CVI_VENC_GetChnParam：获取 Venc 通道参数
- CVI_VENC_RequestIDR：请求 IDR 帧
- CVI_VENC_SetRoiAttr：设置编码通道的感兴趣区域编码配置
- CVI_VENC_GetRoiAttr：获取编码通道的感兴趣区域编码配置
- CVI_VENC_SetRefParam：设置 H.264/H.265 编码通道高级跳帧参考参数
- CVI_VENC_GetRefParam：获取 H.264/H.265 编码通道高级跳帧参考参数
- CVI_VENC_SetFrameLostStrategy：设置瞬时码率超出阈值时丢帧策略的配置
- CVI_VENC_GetFrameLostStrategy：获取瞬时码率超出阈值时丢帧策略的配置
- CVI_VENC_SetModParam：设置编码相关的模块参数
- CVI_VENC_GetModParam 获取编码相关的模块参数
- CVI_VENC_AttachVbPool：将编码通道绑定到某个视频缓存 VB 池中

- CVI_VENC_DetachVbPool：将编码通道从某个视频缓存 VB 池中解绑定
- CVI_VENC_GetH264Entropy：获取 H264 熵编码信息
- CVI_VENC_SetH264Entropy：设置 H264 熵编码信息
- CVI_VENC_InsertUserData：插入用户数据
- CVI_VENC_GetCuPrediction：获取 CU 预测信息
- CVI_VENC_SetCuPrediction：设置 CU 预测信息
- CVI_VENC_GetH264Trans：获取 H.264 协议编码通道的变换、量化属性
- CVI_VENC_SetH264Trans：设置 H.264 协议编码通道的变换、量化属性
- CVI_VENC_GetH265Trans：获取 H.265 协议编码通道的变换、量化属性
- CVI_VENC_SetH265Trans：获取 H.265 协议编码通道的变换、量化属性

7.3.1 CVI_VENC_CreateChn

【描述】

创建编码通道

【语法】

```
CVI_S32 CVI_VENC_CreateChn(VENC_CHN VeChn, const VENC_CHN_ATTR_S *pstAttr);
```

【参数】

参数名称	描述	输入/输出
VeChn	VENC Channel 号。	输入
pstAttr	VENC_CHN_ATTR_S 属性指针。	输入

【返回值】

返回值	描述
0	成功
非 0	失败，其值为错误码 。

【需求】

- 头文件：cvi_comm_venc.h、cvi_venc.h
- 库文件：libvenc.a

【注意】

- 通道属性分为三个结构：

```
typedef struct _VENC_CHN_ATTR_S {
 VENC_ATTR_S stVencAttr;
 VENC_RC_ATTR_S stRcAttr;
```

(下页继续)

(续上页)

```
  VENC_GOP_ATTR_S stGopAttr;
} VENC_CHN_ATTR_S;
```

- VENC_ATTR_S: 决定编码属性，此属性决定编码协议并赋值设定对应宽，高。
- VENC_RC_ATTR_S: 码率控制器属性，此属性随编码设定及码率控制模式（CBR、VBR、AVBR、FIXQP）不同对应子结构做设定。
- VENC_GOP_ATTR_S：GOP 类型属性，编码 GOP 类型 (编码单参考帧，P 帧，GOP 类型。)
- 推荐的编码宽高为：1920x1080（1080P）、1280x720（720P）。

【举例】

使用者可参阅 sample_common_venc.c 内 SAMPLE_COMM_VENC_SetChnAttr 参考对应属性指针的设定。

【相关主题】

- CVI_VENC_DestroyChn

7.3.2 CVI_VENC_DestroyChn

【描述】

销毁编码通道

【语法】

```
CVI_S32 CVI_VENC_DestroyChn (VENC_CHN VeChn);
```

【参数】

参数名称	描述	输入/输出
VeChn	VENC Channel 号。	输入

【返回值】

返回值	描述
0	成功
非 0	失败，其值为错误码 。

【需求】

- 头文件：cvi_comm_venc.h、cvi_venc.h
- 库文件：libvenc.a

【注意】

销毁并不存在的通道，将返回失败。

【举例】

可参阅 sample_common_venc.c ，销毁通道前必须停止传送 frame。

```
CVI_S32 SAMPLE_COMM_VENC_Stop(VENC_CHN VencChn)
{
  CVI_S32 s32Ret;
        //停止传送venc frame data 接收
  s32Ret = CVI_VENC_StopRecvFrame(VencChn);
  if (s32Ret != CVI_SUCCESS) {
    CVI_VENC_ERR("CVI_VENC_StopRecvPic vechn[%d] failed with %#x!\n", VencChn, s32Ret);
    return CVI_FAILURE;
  }
        //停止线程执行
  if (gs_VencTask[VencChn] != 0) {
    pthread_join(gs_VencTask[VencChn], CVI_NULL);
    CVI_VENC_SYNC("GetVencStreamProc done\n");
    gs_VencTask[VencChn] = 0;
  }
        //注销venc 通道
  s32Ret = CVI_VENC_DestroyChn(VencChn);
  if (s32Ret != CVI_SUCCESS) {
    CVI_VENC_ERR("CVI_VENC_DestroyChn vechn[%d] failed with %#x!\n", VencChn, s32Ret);
    return CVI_FAILURE;
  }
  return CVI_SUCCESS;
}
```

【相关主题】

- CVI_VENC_CreateChn

7.3.3 CVI_VENC_ResetChn

【描述】

复位通道。

【语法】

```
CVI_S32 CVI_VENC_ResetChn(VENC_CHN VeChn);
```

【参数】

参数名称	描述	输入/输出
VeChn	通道号。	输入

【返回值】

返回值	描述
0	成功
非 0	失败，其值为错误码 。

【需求】

- 头文件：cvi_comm_venc.h、cvi_venc.h
- 库文件：libvenc.a

【注意】

- Reset 并不存在的通道，返回失败 CVI_FAILURE。
- 如果一个通道没有停止接收图像而 reset 通道，则返回失败。

【举例】

- 无

7.3.4 CVI_VENC_StartRecvFrame

【描述】

开启编码通道接收输入图像，允许指定接收帧数，超出指定的帧数后自动停止接收图像。

【语法】

```
CVI_S32 CVI_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。	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，其值为错误码 。

【需求】

- 头文件：cvi_comm_venc.h、cvi_venc.h
- 库文件：libvenc.a

【注意】

- 如果通道未创建，则返回失败
- 开启编码器接收之后才能开始接收图像编码。

【举例】

- 可参照 sample_common_venc.c 内 SAMPLE_COMM_VENC_Start 函数。

```
CVI_S32 s32Ret;
VENC_RECV_PIC_PARAM_S stRecvParam;

//创建venc 通道
s32Ret = SAMPLE_COMM_VENC_Create(
    pIc, VencChn, enType, enSize, enRcMode,
    u32Profile, bRcnRefShareBuf, pstGopAttr);
if (s32Ret != CVI_SUCCESS) {
  CVI_VENC_ERR("SAMPLE_COMM_VENC_Create failed with %d\n", s32Ret);
  return CVI_FAILURE;
}
//设定venc bindmode 与否
if (pIc->bind_mode == VENC_BIND_VI) {
  VI_PIPE ViPipe = 0;
  VI_CHN ViChn = 0;
  SAMPLE_COMM_VI_Bind_VENC(ViPipe, ViChn, VencChn);
} else if (pIc->bind_mode == VENC_BIND_VPSS) {
  VPSS_GRP VpssGrp = 0;
  VPSS_CHN VpssChn = 0;
  SAMPLE_COMM_VPSS_Bind_VENC(VpssGrp, VpssChn, VencChn);
}

//指定接收frame 数，允许接收frame
stRecvParam.s32RecvPicNum = pIc->num_frames;
s32Ret = CVI_VENC_StartRecvFrame(VencChn, &stRecvParam);
if (s32Ret != CVI_SUCCESS) {
  CVI_VENC_ERR("CVI_VENC_StartRecvPic failed with %d\n", s32Ret);
  return CVI_FAILURE;
}
return CVI_SUCCESS;
```

【相关主题】

- CVI_VENC_CreateChn

7.3.5 CVI_VENC_StopRecvFrame

【描述】

停止编码通道接收输入图像。

【语法】

```
CVI_S32 CVI_VENC_StopRecvFrame(VENC_CHN VeChn);
```

【参数】

参数名称	描述	输入/输出
VeChn	编码通道号。	输入

【返回值】

返回值	描述
0	成功
非 0	失败，其值为错误码 。

【需求】

- 头文件：cvi_comm_venc.h、cvi_venc.h
- 库文件：libvenc.a

【注意】

- 通道销毁前，需调用 CVI_VENC_StopRecvFrame。
- 如通道未创建或信道已销毁，则返回失败。
- 调用此接口仅停止接收原始数据编码，码流 buffer 并不会被清除。
- 此接口用于编码通道停止接收图像来编码，在编码通道销毁或复位前必须停止接收图像。

【举例】

- 可参照 sample_common_venc.c 内 SAMPLE_COMM_VENC_Stop 函数。

【相关主题】

- CVI_VENC_DestroyChn

7.3.6 CVI_VENC_QueryStatus

【描述】

查询编码通道状态。

【语法】

```
CVI_S32 CVI_VENC_QueryStatus(VENC_CHN VeChn, VENC_CHN_STATUS_S *pstStatus);
```

【参数】

参数名称	描述	输入/输出
VeChn	VENC Channel 号。	输入
pstStatus	编码通道的状态指针。	输出

【返回值】

返回值	描述
0	成功
非 0	失败，其值为错误码 。

【需求】

- 头文件：cvi_comm_venc.h、cvi_venc.h
- 库文件：libvenc.a

【注意】

- 如果通道未创建，则返回失败。
- 在编码通道状态结构体中，u32CurPacks 表示当前帧的码流包个数。在调用CVI_VENC_GetStream 之前应确保 u32CurPacks 大于 0。

7.3.7 CVI_VENC_SetChnAttr

【描述】

设置编码通道属性。

【语法】

```
CVI_S32 CVI_VENC_SetChnAttr(VENC_CHN VeChn, const VENC_CHN_ATTR_S␣
↪*pstChnAttr);
```

【参数】

参数名称	描述	输入/输出
VeChn	VENC Channel 号。	输入
pstChnAttr	编码通道属性指针。	输入

【返回值】

返回值	描述
0	成功
非 0	失败，其值为错误码 。

【需求】

- 头文件：cvi_comm_venc.h、cvi_venc.h
- 库文件：libvenc.a

【注意】

无

【举例】

无

7.3.8 CVI_VENC_GetChnAttr

【描述】

获取编码通道属性。

【语法】

```
CVI_S32 CVI_VENC_GetChnAttr(VENC_CHN VeChn, VENC_CHN_ATTR_S *pstChnAttr);
```

【参数】

参数名称	描述	输入/输出
VeChn	VENC Channel 号。	输入
pstChnAttr	编码通道属性指针。	输出

【返回值】

返回值	描述
0	成功
非 0	失败，其值为错误码 。

【需求】

- 头文件：cvi_comm_venc.h、cvi_venc.h
- 库文件：libvenc.a

【注意】

- 获取未创建的通道的属性，返回失败。

【举例】

7.3.9 CVI_VENC_GetStream

【描述】

获取编码的码流。

【语法】

```
CVI_S32 CVI_VENC_GetStream(VENC_CHN VeChn, VENC_STREAM_S *pstStream, CVI_S32␣
↪S32MilliSec);
```

【参数】

参数名称	描述	输入/输出
VeChn	VENC Channel 号。	输入
pstFrame	码流结构体指针 VENC_STREAM_S	输出
S32MilliSec	发送图像超时时间。 取值范围：[-1, +∞) -1：阻塞。 0：非阻塞。 大于 0：超时时间。	输入

【返回值】

返回值	描述
0	成功
非 0	失败，其值为错误码 。

【需求】

- 头文件：cvi_comm_venc.h、cvi_venc.h
- 库文件：libvenc.a

【注意】

- venc 通道必须先创建，否则此函数会回传错误。
- 在非 bind mode 状况下，CVI_VENC_SendFrame 必须有调用过，否则 CVI_VENC_GetStream 将无法获得码流结构。
- CVI_VENC_GetStream 调用之前，可以搭配 CVI_VENC_GetChnAttr, CVI_VENC_QueryStatus 调用确保编码信道当下的正确性。
- 码流结构体 VENC_STREAM_S 包含 4 个部分：
  - 码流包信息指针 pstPack 指向一组 VENC_PACK_S 的内存空间，该空间由调用者分配。
  - 注意：pstPack 空间必须在调用 CVI_VENC_GetStream 前给予对应空间，否则将回传错误。

【举例】

可参照 sample_venc_lib.c 内的 SAMPLE_VENC_GetVencStreamProc:

```
while (pVencChnCtx->chnStat == CHN_STAT_START) {
  VENC_CHN_STATUS_S stStat;
  //bind mode 模式检查
  if (pVencChnCtx->chnIc.bind_mode == VENC_BIND_DISABLE) {
    CVI_S32 s32SetFrameMilliSec = 20000;
    s32Ret = cviReadSrcFrame(pVencChnCtx->pstVFrame, pVencChnCtx->fpSrc);

    s32Ret = CVI_VENC_SendFrame(VencChn, pVencChnCtx->pstFrameInfo, s32SetFrameMilliSec);
    if (s32Ret == CVI_ERR_VENC_FRC_NO_ENC) {
      continue;
    } else if (s32Ret != CVI_SUCCESS) {
      break;
```

(下页继续)

(续上页)

```
  }
 }
 //根据venc 通道状态，开始获取venc stream

 if (pVencChnCtx->chnIc.bsMode == BS_MODE_QUERY_STAT) {
  if (vencFd < 0) {
   vencFd = CVI_VENC_GetFd(vencChn);
   if (vencFd < 0) {
    SAMPLE_PRT("CVI_VENC_GetFd failed with%#x!\n", vencFd);
    break;
   }
  }

  FD_ZERO(&readFds);
  FD_SET(vencFd, &readFds);
  timeoutVal.tv_sec = 1;
  timeoutVal.tv_usec = 0;
  ret = select(vencFd + 1, &readFds, NULL, NULL, &timeoutVal);
  if (ret < 0) {
   if (errno == EINTR)
    continue;
   SAMPLE_PRT("select failed!\n");
   break;
  } else if (ret == 0) {
   SAMPLE_PRT("select timeout\n");
   continue;
  }
  VENC_CHN_ATTR_S stVencChnAttr;
  VENC_STREAM_S stStream;

  s32Ret = CVI_VENC_GetChnAttr(VencChn, &stVencChnAttr);
  if (s32Ret != CVI_SUCCESS) {
   CVI_VENC_ERR("CVI_VENC_GetChnAttr, VencChn = %d, s32Ret = %d\n", VencChn,␣
↪s32Ret);
   break;
  }
  s32Ret = CVI_VENC_QueryStatus(VencChn, &stStat);
  if (s32Ret != CVI_SUCCESS) {
   CVI_VENC_ERR("CVI_VENC_QueryStatus, Vench = %d, s32Ret = %d\n",VencChn, s32Ret);
   break;
  }
  if (!stStat.u32CurPacks) {
   CVI_VENC_ERR("u32CurPacks = NULL!\n");
   break;
  }
  //需先行赋予pstPack空间
  stStream.pstPack = (VENC_PACK_S *)malloc(sizeof(VENC_PACK_S) * stStat.u32CurPacks);
  if (stStream.pstPack == NULL) {
   CVI_VENC_ERR("malloc memory failed!\n");
   break;
  }
  s32Ret = CVI_VENC_GetStream(VencChn, &stStream, -1);
  if (s32Ret != CVI_SUCCESS) {
   CVI_VENC_ERR("CVI_VENC_GetStream, VencChn = %d, s32Ret = %d\n", VencChn,␣
↪s32Ret);
```

(下页继续)

(续上页)

```
      free(stStream.pstPack);
      stStream.pstPack = NULL;
      break;
    }
    //使用者可将取出的stStream存档或给予上层app

    s32Ret = CVI_VENC_ReleaseStream(VencChn, &stStream);
    if (s32Ret != CVI_SUCCESS) {
      CVI_VENC_ERR("CVI_VENC_ReleaseStream, s32Ret = %d\n", s32Ret);
      free(stStream.pstPack);
      stStream.pstPack = NULL;
      break;
    }
    free(stStream.pstPack);
    stStream.pstPack = NULL;
  }
}
```

7.3.10 CVI_VENC_ReleaseStream

【描述】

释放码流缓存。

【语法】

```
CVI_S32 CVI_VENC_ReleaseStream(VENC_CHN VeChn, VENC_STREAM_S *pstStream);
```

【参数】

参数名称	描述	输入/输出
VeChn	VENC Channel 号。	输入
pstStream	VENC_STREAM_S 属性指针。	输入

【返回值】

返回值	描述
0	成功
非 0	失败，其值为错误码 。

【需求】

- 头文件：cvi_comm_venc.h、cvi_venc.h
- 库文件：libvenc.a

【注意】

- 请确认 venc 通道已创建。
- 请确认 CVI_VENC_ReleaseStream 在 CVI_VENC_GetStream 之后，若提早调用 CVI_VENC_ReleaseStream，

则 CVI_VENC_GetStream 将不具备取得正确 stream 保证。

- 用户获取码流后必须及时释放已经获取的码流缓存，否则可能会导致码流 buffer 满，影响编码器编码。

【举例】

- 请参考CVI_VENC_GetStream 的举例。

7.3.11 CVI_VENC_SendFrame

【描述】

送出影像以供编码。

【语法】

```
CVI_S32 CVI_VENC_SendFrame(VENC_CHN VeChn, const VIDEO_FRAME_INFO_S␣
↪*pstFrame, CVI_S32 S32MilliSec);
```

【参数】

参数名称	描述	输入/输出
VeChn	VENC Channel 号。	输入
pstFrame	VIDEO_FRAME_INFO_S 属性指针。	输入
S32MilliSec	发送图像超时时间 输入取值范围：[-1,+∞) -1：阻塞。 0：非阻塞。 大于 0：超时时间。	输入

【返回值】

返回值	描述
0	成功
非 0	失败，其值为错误码 。

【需求】

- 头文件：cvi_comm_venc.h、cvi_venc.h
- 库文件：libvenc.a

【注意】

- 此接口支持用户发送图像至编码信道，以进行编码。
- 调用该接口发送图像，用户需要保证编码通道已创建且开启接收输入图像。

【举例】

可参照 sample_venc_lib.c‘内 ‘SAMPLE_VENC_GetVencStreamProc ：

```
while (pVencChnCtx->chnStat == CHN_STAT_START) {
  VENC_CHN_STATUS_S stStat;
  if (pVencChnCtx->chnIc.bind_mode == VENC_BIND_DISABLE) {
    CVI_S32 s32SetFrameMilliSec = 20000;
    s32Ret = cviReadSrcFrame(pVencChnCtx->pstVFrame, pVencChnCtx->fpSrc);
    if (s32Ret < 0) {
      CVI_VENC_ERR("(chn %d) cviReadSrcFrame fail\n", VencChn);
      break;
    }
    s32Ret = CVI_VENC_SendFrame(VencChn, pVencChnCtx->pstFrameInfo, s32SetFrameMilliSec);
    if (s32Ret == CVI_ERR_VENC_FRC_NO_ENC) {
      CVI_VENC_FRC("no encode\n");
      continue;
    } else if (s32Ret != CVI_SUCCESS) {
      CVI_VENC_ERR("CVI_VENC_SendFrame, VencChn = %d, s32Ret = %d\n", VencChn,␣
↪s32Ret);
      break;
    }
  }
  if (pVencChnCtx->chnIc.bsMode == BS_MODE_QUERY_STAT) {
    VENC_CHN_ATTR_S stVencChnAttr;
    VENC_STREAM_S stStream;
    s32Ret = CVI_VENC_GetChnAttr(VencChn, &stVencChnAttr);
    if (s32Ret != CVI_SUCCESS) {
      CVI_VENC_ERR("CVI_VENC_GetChnAttr, VencChn = %d, s32Ret = %d\n", VencChn,␣
↪s32Ret);
      break;
    }
    s32Ret = CVI_VENC_QueryStatus(VencChn, &stStat);
    if (s32Ret != CVI_SUCCESS) {
      CVI_VENC_ERR("CVI_VENC_QueryStatus, Vench = %d, s32Ret = %d\n", VencChn, s32Ret);
      break;
    }
    if (!stStat.u32CurPacks) {
      CVI_VENC_ERR("u32CurPacks = NULL!\n");
      break;
    }

    stStream.pstPack = (VENC_PACK_S *)malloc(sizeof(VENC_PACK_S) * stStat.u32CurPacks);
    if (stStream.pstPack == NULL) {
      CVI_VENC_ERR("malloc memory failed!\n");
      break;
    }

    s32Ret = CVI_VENC_GetStream(VencChn, &stStream, -1);
    if (s32Ret != CVI_SUCCESS) {
      CVI_VENC_ERR("CVI_VENC_GetStream, VencChn = %d, s32Ret = %d\n", VencChn,␣
↪s32Ret);
      free(stStream.pstPack);
      stStream.pstPack = NULL;
      break;
    }
    s32Ret = CVI_VENC_ReleaseStream(VencChn, &stStream);
    if (s32Ret != CVI_SUCCESS) {
      CVI_VENC_ERR("CVI_VENC_ReleaseStream, s32Ret = %d\n", s32Ret);
```

(下页继续)

(续上页)

```
      free(stStream.pstPack);
      stStream.pstPack = NULL;
      break;
    }
    free(stStream.pstPack);
    stStream.pstPack = NULL;
  }

  if (pVencChnCtx->chnIc.bind_mode) {
    if (pVencChnCtx->chnStat != pVencChnCtx->nextChnStat) {
      s32Ret = CVI_VENC_QueryStatus(VencChn, &stStat);
      if (s32Ret != CVI_SUCCESS) {
        CVI_VENC_ERR("CVI_VENC_QueryStatus, Vench = %d, s32Ret = %d\n",
            VencChn, s32Ret);
        break;
      }

      CVI_VENC_TRACE("u32LeftStreamFrames = %d\n", stStat.u32LeftStreamFrames);
      if (stStat.u32LeftStreamFrames <= 0) {
        pVencChnCtx->chnStat = CHN_STAT_STOP;
        CVI_VENC_SYNC("chnStat = CHN_STAT_STOP\n");
      }
    }
  } else {

    if (i >= pVencChnCtx->num_frames) {
      pVencChnCtx->chnStat = CHN_STAT_STOP;
    }
  }
}
```

【相关主题】

- CVI_VENC_QueryStatus
- CVI_VENC_GetStream

7.3.12 CVI_VENC_GetFd

【描述】

获取编码通道对应的设备文件句柄。

【语法】

```
CVI_S32 CVI_VENC_GetFd(VENC_CHN VeChn);
```

【参数】

参数名称	描述	输入/输出
VeChn	VENC Channel 号。	输入

【返回值】

返回值	描述
0	成功
小于或等于 0	失败，其值为错误码 。

【需求】

- 头文件：cvi_comm_venc.h、cvi_venc.h
- 库文件：libvenc.a

【注意】

- 无

【举例】

- 无

【相关主题】

- CVI_VENC_CloseFd

7.3.13 CVI_VENC_CloseFd

【描述】

取得通道 handle。

【语法】

```
CVI_S32 CVI_VENC_CloseFd(VENC_CHN VeChn);
```

【参数】

参数名称	描述	输入/输出
VeChn	VENC Channel 号。	输入

【返回值】

返回值	描述
0	成功
非 0	失败，其值为错误码 。

【需求】

- 头文件：cvi_comm_venc.h、cvi_venc.h
- 库文件：libvenc.a

【注意】

- 正在使用 select 方式监听 Stream 时，不能关闭 fd，会导致功能不正常。

【举例】

- 无

【相关主题】

- CVI_VENC_GetFd

7.3.14 CVI_VENC_SetJpegParam

【描述】

设置 JPEG 协议编码通道的高级参数。

【语法】

```
CVI_S32 CVI_VENC_SetJpegParam(VENC_CHN VeChn, const VENC_JPEG_PARAM_S
↪*pstJpegParam);
```

【参数】

参数名称	描述	输入/输出
VeChn	VENC Channel 号。	输入
pstJpegParam	JPEG 协议编码通道的高级参数集合指针。	输入

【返回值】

返回值	描述
0	成功
非 0	失败，其值为错误码 。

【需求】

- 头文件：cvi_comm_venc.h、cvi_venc.h
- 库文件：libvenc.a

【注意】

- 本接口用于设置 JPEG 协议编码通道的高级参数。
- 本接口使用于 venc 通道创建后。
- 高级参数主要参数组成
  - u32Qfactor：量化表因子范围为 [1, 99]，
    u32Qfactor 越大，量化表中的量化系数越小，得到的图像质量会更好，
    同时，编码压缩率更低。
    同理 u32Qfactor 越小，量化表中的量化系数越大，得到的图像质量会更差，
    同时，编码压缩率更高。
  - 其中 u32Qfactor=50 为预留自定义设置项，暂不支持设置
  - u32MCUPerECS：每个 Ecs 中包含多少 Mcu。
    系统模式 u32MCUPerECS = 0，表示当前帧的所有的 MCU 被编码为一个 ECS。

u32MCUPerECS 的最小值为 0，最大值不超过 (picwidth+15)&4 * (picheight+15)&4 * 2。

- 建议用户在创建通道之后，启动编码之前调用此接口，减少在编码过程中调用的次数。建议用户在调用此接口之前，先调用 CVI_VENC_GetJpegParam 接口，获取当前编码信道的 JpegParam 配置，然后再进行设置。

【举例】

- 请参照 sample_common_venc.c 内的 SAMPLE_COMM_VENC_SetJpegParam 函数：

```
CVI_S32 SAMPLE_COMM_VENC_SetJpegParam(chnInputCfg *pIc, VENC_CHN VencChn)
{
  VENC_JPEG_PARAM_S stJpegParam, *pstJpegParam = &stJpegParam;
  CVI_S32 s32Ret = CVI_SUCCESS;

  s32Ret = CVI_VENC_GetJpegParam(VencChn, pstJpegParam);
  if (s32Ret != CVI_SUCCESS) {
    CVI_VENC_ERR("CVI_VENC_GetJpegParam\n");
    return CVI_FAILURE;
  }

  pstJpegParam->u32Qfactor = pIc->quality;

  s32Ret = CVI_VENC_SetJpegParam(VencChn, pstJpegParam);
  if (s32Ret != CVI_SUCCESS) {
    CVI_VENC_ERR("CVI_VENC_SetJpegParam\n");
    return CVI_FAILURE;
  }

  return s32Ret;
}
```

7.3.15 CVI_VENC_GetJpegParam

【描述】

获取 JPEG 协议编码通道的高级参数配置。

【语法】

```
CVI_S32 CVI_VENC_GetJpegParam(VENC_CHN VeChn, VENC_JPEG_PARAM_S␣
↪*pstJpegParam);
```

【参数】

参数名称	描述	输入/输出
VeChn	VENC Channel 号。	输入
pstJpegParam	JPEG 协议编码通道的高级参数集合指针。	输出

【返回值】

返回值	描述
0	成功
非 0	失败，其值为错误码 。

【需求】

- 头文件：cvi_comm_venc.h、cvi_venc.h
- 库文件：libvenc.a

【注意】

- 本接口在编码通道创建之后，编码通道销毁之前调用。
- 建议用户在创建通道之后，启动编码之前调用此接口，减少在编码过程中调用的次数。

【举例】

- 请参见CVI_VENC_SetJpegParam 的举例。

7.3.16 CVI_VENC_SetRcParam

【描述】

设置编码通道码率控制器的高级参数。

【语法】

```
CVI_S32 CVI_VENC_SetRcParam(VENC_CHN VeChn, const VENC_RC_PARAM_S
↪*pstRcParam);
```

【参数】

参数名称	描述	输入/输出
VeChn	VENC Channel 号。	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，其值为错误码 。

【需求】

- 头文件：cvi_comm_venc.h、cvi_venc.h
- 库文件：libvenc.a

【注意】

- 本接口在编码通道创建之后，编码通道销毁之前调用。
- 此函数调用于CVI_VENC_CreateChn 之后，CVI_VENC_SetChnParam 之前。

- 建议用户先调用CVI_VENC_GetRcParam 接口，获取 RC 高级参数，然后修改相应参数，再调用本接口对高级参数进行设置。

【举例】

- 可参考 sample_common_venc.c``内 ``SAMPLE_COMM_VENC_SetRcParam 函数：

```
static CVI_S32 SAMPLE_COMM_VENC_SetRcParam(
    chnInputCfg * pIc,
    VENC_CHN VencChn)
{
  CVI_S32 s32Ret;
  VENC_RC_PARAM_S stRcParam, *pstRcParam = &stRcParam;

  s32Ret = CVI_VENC_GetRcParam(VencChn, pstRcParam);
  if (s32Ret != CVI_SUCCESS) {
    CVI_VENC_ERR("GetRcParam failed!\n");
    return CVI_FAILURE;
  }

  pstRcParam->s32FirstFrameStartQp = pIc->firstFrmstartQp;
  if (!strcmp(pIc->codec, "264") && pIc->rcMode == 0) {
    pstRcParam->stParamH264Cbr.u32MaxIQp = pIc->maxIqp;
    pstRcParam->stParamH264Cbr.u32MinIQp = pIc->minIqp;
    pstRcParam->stParamH264Cbr.u32MaxQp = pIc->maxQp;
    pstRcParam->stParamH264Cbr.u32MinQp = pIc->minQp;
  } else if (!strcmp(pIc->codec, "265") && pIc->rcMode == 0) {
    pstRcParam->stParamH265Cbr.u32MaxIQp = pIc->maxIqp;
    pstRcParam->stParamH265Cbr.u32MinIQp = pIc->minIqp;
    pstRcParam->stParamH265Cbr.u32MaxQp = pIc->maxQp;
    pstRcParam->stParamH265Cbr.u32MinQp = pIc->minQp;
  }  else if (!strcmp(pIc->codec, "264") && pIc->rcMode == 1) {
    pstRcParam->stParamH264Vbr.u32MaxIQp = pIc->maxIqp;
    pstRcParam->stParamH264Vbr.u32MinIQp = pIc->minIqp;
    pstRcParam->stParamH264Vbr.u32MaxQp = pIc->maxQp;
    pstRcParam->stParamH264Vbr.u32MinQp = pIc->minQp;
  }  else if (!strcmp(pIc->codec, "265") && pIc->rcMode == 1) {
    pstRcParam->stParamH265Vbr.u32MaxIQp = pIc->maxIqp;
    pstRcParam->stParamH265Vbr.u32MinIQp = pIc->minIqp;
    pstRcParam->stParamH265Vbr.u32MaxQp = pIc->maxQp;
    pstRcParam->stParamH265Vbr.u32MinQp = pIc->minQp;
  }

  CVI_VENC_TRACE("firstFrmstartQp = %d\n", pIc->firstFrmstartQp);

  s32Ret = CVI_VENC_SetRcParam(VencChn, pstRcParam);
  if (s32Ret != CVI_SUCCESS) {
    CVI_VENC_ERR("SetRcParam failed!\n");
    return CVI_FAILURE;
  }

  return s32Ret;
}
```

7.3.17 CVI_VENC_GetRcParam

【描述】

获取通道码率控制高级参数。

【语法】

```
CVI_S32 CVI_VENC_GetRcParam(VENC_CHN VeChn, VENC_RC_PARAM_S *pstRcParam);
```

【参数】

参数名称	描述	输入/输出
VeChn	VENC Channel 号。	输入
pstRcParam	通道码率控制参数指针	输出

【返回值】

返回值	描述
0	成功
非 0	失败，其值为错误码 。

【需求】

- 头文件：cvi_comm_venc.h、cvi_venc.h
- 库文件：libvenc.a

【注意】

- 本接口在编码通道创建之后，编码通道销毁之前调用。
- 建议用户在调用此接口之前，先调用CVI_VENC_GetChnParam 接口，获取当前信道的参数配置，然后再进行设置。

【举例】

无。

7.3.18 CVI_VENC_SetChnParam

【描述】

设置通道参数。

【语法】

```
CVI_S32 CVI_VENC_SetChnParam(VENC_CHN VeChn, const VENC_CHN_PARAM_S␣
↪*pstChnParam);
```

【参数】

参数名称	描述	输入/输出
VeChn	VENC Channel 号。	输入
pstChnParam	Venc 的通道参数。	输入

【返回值】

返回值	描述
0	成功
非 0	失败，其值为错误码 。

【需求】

- 头文件：cvi_comm_venc.h、cvi_venc.h
- 库文件：libvenc.a

【注意】

- 本接口在编码通道创建之后，编码通道销毁之前调用。
- 此函数调用于CVI_VENC_CreateChn 之后，CVI_VENC_SetChnParam 之前。
- 此函数 API 参数支持 venc crop 设定，其使用参数为：VENC_CHN_PARAM_S，stCropCfg。可设定对应的 x，y 轴及宽 (width)，高 (height)。

【举例】

```
CVI_S32 SAMPLE_COMM_VENC_SetChnParam(chnInputCfg *pIc, VENC_CHN VencChn)
{
    VENC_CHN_PARAM_S stChnParam, *pstChnParam = &stChnParam;
    CVI_S32 s32Ret = CVI_SUCCESS;

    s32Ret = CVI_VENC_GetChnParam(VencChn, pstChnParam);
    if (s32Ret != CVI_SUCCESS) {
        CVI_VENC_ERR("CVI_VENC_GetJpegParam\n");
        return CVI_FAILURE;
    }
    pstChnParam->stCropCfg.bEnable = (pIc->posX || pIc->posY);
    pstChnParam->stCropCfg.stRect.s32X = pIc->posX;
    pstChnParam->stCropCfg.stRect.s32Y = pIc->posY;
    pstChnParam->stCropCfg.stRect.u32Width = pIc->width;
    pstChnParam->stCropCfg.stRect.u32Height = pIc->height;

    s32Ret = CVI_VENC_SetChnParam(VencChn, pstChnParam);
    if (s32Ret != CVI_SUCCESS) {
        CVI_VENC_ERR("CVI_VENC_SetJpegParam\n");
        return CVI_FAILURE;
    }
    return s32Ret;
}
```

7.3.19 CVI_VENC_GetChnParam

【描述】

获取通道参数。

【语法】

```
CVI_S32 CVI_VENC_GetChnParam(VENC_CHN VeChn, VENC_CHN_PARAM_S
↪*pstChnParam);
```

【参数】

参数名称	描述	输入/输出
VeChn	VENC Channel 号。	输入
pstChnParam	Venc 的通道参数指针。	输出

【返回值】

返回值	描述
0	成功
非 0	失败，其值为错误码 。

【需求】

- 头文件：cvi_comm_venc.h、cvi_venc.h
- 库文件：libvenc.a

【注意】

- 无

【举例】

- 无

7.3.20 CVI_VENC_RequestIDR

【描述】

请求 IDR 帧。

【语法】

```
CVI_S32 CVI_VENC_RequestIDR(VENC_CHN VeChn, CVI_BOOL bInstant);
```

【参数】

参数名称	描述	输入/输出
VeChn	VENC Channel 号。	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，其值为错误码 。

【需求】

- 头文件：cvi_comm_venc.h、cvi_venc.h
- 库文件：libvenc.a

【注意】

- 如果通道未创建，则返回失败。
- 接受 IDR 帧请求后，不论 bInstant =0 与否，立即编出 IDR 帧，不受帧率控制约束。
- IDR 帧请求，只支持 H.264/H.265 编码协议。
- 此接口不受帧率控制影响，每调用一次即编出一个 IDR，调用频繁会影响码流帧率和码率的稳定。
- 在下一帧编码启动前多次接口调用只会编出一个 IDR。原帧已为 IDR 帧时不生效。
- 当高级跳帧始能时，请求 IDR 帧可能会延时生效。

【举例】

- 无

7.3.21 CVI_VENC_SetRoiAttr

【描述】

设置 H.264/H.265 通道的 ROI 属性。

【语法】

```
CVI_S32 CVI_VENC_SetRoiAttr(VENC_CHN VeChn, const VENC_ROI_ATTR_S *pstRoiAttr);
```

【参数】

参数名称	描述	输入/输出
VeChn	VENC Channel 号。	输入
pstRoiAttr	ROI 区域参数。	输入

【返回值】

返回值	描述
0	成功
非 0	失败，其值为错误码 。

【需求】

- 头文件：cvi_comm_venc.h、cvi_venc.h
- 库文件：libvenc.a

【注意】

- u32Index：支持每个通道可设置 8 个 ROI 区域，按照 0 ~ 7 索引号对 ROI 区域进行管理，u32Index 表示的用户设置 ROI 的索引号。重复区域按照 0 ~ 7 的 ROI 索引号依次提高优先级。
- bEnable：指定当前的 ROI 区域是否使能。
- bAbsQp：指定当前的 ROI 区域使用绝对 QP 或是相对 QP 模式。
- s32Qp：当 bAbsQp 为 CVI_TRUE 时，s32Qp 为对 ROI 区域设定的 Qp 值，当 bAbsQp 为 CVI_FALSE 时，s32Qp 表示对 ROI 区域内部码率控制之 Qp 加上设定的 Qp 偏移值。
- stRect：指定当前的 ROI 区域的位置坐标和区域的大小。ROI 区域必须在图像范围内。
- 系统默认没有 ROI 区域使能，用户必须在编码通道创建之后，编码通道销毁之前设置调用此接口启动 ROI。此接口在编码过程中被调用时，等到下一个帧时生效。
- 建议用户在创建通道之后，启动编码之前调用此接口，减少在编码过程中调用的次数。建议用户在调用此接口之前，先调用 CVI_VENC_GetRoiAttr 接口，获取当前信道的 ROI 配置后再进行设置。
- 设置该接口后，如果当前帧判断编码为 pskip 帧，以 pskip 帧效果优先。
- 当码率控制模式不为 Fixed QP 模式时, ROI 区域可配置。
- H.264 当 ROI 始能时, 宏块级码率控制失效。
- 绝对 Qp 模式因为码率控制调适宏块 QP, 实际编码 QP 与设置之 QP 可能会有些差异。

【举例】

- 无

7.3.22 CVI_VENC_GetRoiAttr

【描述】

获取 H.264/H.265 通道的 ROI 属性。

【语法】

```
CVI_S32 CVI_VENC_GetRoiAttr(VENC_CHN VeChn, CVI_U32 u32Index, VENC_ROI_ATTR_
↪S *pstRoiAttr);
```

【参数】

参数名称	描述	输入/输出
VeChn	VENC Channel 号。	输入
u32Index	ROI 区域索引。	输入
pstRoiAttr	ROI 区域参数。	输出

【返回值】

返回值	描述
0	成功
非 0	失败，其值为错误码 。

【需求】

- 头文件：cvi_comm_venc.h、cvi_venc.h
- 库文件：libvenc.a

【注意】

- 依照 u32Index 索引获取该 ROI 区域配置
- 用户必须在编码通道创建之后，编码通道销毁之前设置调用此接口
- 建议用户在调用CVI_VENC_SetRoiAttr 接口之前，先调用此接口，获取当前信道的 ROI 配置后再进行设置。

【举例】

- 无

7.3.23 CVI_VENC_SetRefParam

【描述】

设置 H.264/H.265 编码通道高级跳帧参考参数。

【语法】

```
CVI_S32 CVI_VENC_SetRefParam(VENC_CHN VeChn, const VENC_REF_PARAM_S␣
↪*pstRefParam);
```

【参数】

参数名称	描述	输入/输出
VeChn	VENC Channel 号。	输入
pstRefParam	H.264/H.265 编码通道高级跳帧参考参数。	输入

【返回值】

返回值	描述
0	成功
非 0	失败，其值为错误码 。

【需求】

- 头文件：cvi_comm_venc.h、cvi_venc.h
- 库文件：libvenc.a

【注意】

- H.264 和 H.265 均支持 1 倍、2 倍、以及 4 倍跳帧参考模式。默认为 1 倍跳帧参考模式 (不跳帧) 。
- 需在创建通道之后，启动编码之前调用此接口，避免在编码过程中调用。
- 1 倍跳帧参考模式设置的配置为：bEnablePred = HI_TRUE，u32Enhance = 0，u32Base = 1
- 2 倍跳帧参考模式设置的配置为：bEnablePred = HI_TRUE，u32Enhance = 1，u32Base = 1。
- 4 倍跳帧参考模式设置的配置为：bEnablePred = HI_TRUE，u32Enhance = 1，u32Base = 2
- 用户设置通道的 Gop Mode 为 NormalP 时，2 倍跳帧参考模式的 u32Gop 需为 2 的倍数；4 倍跳帧参考模式的 u32Gop 需为 4 的倍数
- 用户设置通道的 Gop Mode 为 SmartP 时，2 倍跳帧参考模式的 u32Gop 与 u32BgInterval 需为 2 的倍数；4 倍跳帧参考模式的 u32Gop 与 u32BgInterval 需为 4 的倍数

7.3.24 CVI_VENC_GetRefParam

【描述】

获取 H.264/H.265 编码通道高级跳帧参考参数。

【语法】

```
CVI_S32 CVI_VENC_GetRefParam(VENC_CHN VeChn, VENC_REF_PARAM_S *pstRefParam);
```

【参数】

参数名称	描述	输入/输出
VeChn	VENC Channel 号。	输入
pstRefParam	H.264/H.265 编码通道高级跳帧参考参数。	输出

【返回值】

返回值	描述
0	成功
非 0	失败，其值为错误码 。

【需求】

- 头文件：cvi_comm_venc.h、cvi_venc.h
- 库文件：libvenc.a

【注意】

- 无

【举例】

- 无

7.3.25 CVI_VENC_SetFrameLostStrategy

【描述】

设置编码通道瞬时码率超过阈值时丢帧策略。

【语法】

```
CVI_S32 CVI_VENC_SetFrameLostStrategy(VENC_CHN VeChn, const VENC_FRAMELOST_S␣
↪*pstFrmLostParam);
```

【参数】

参数名称	描述	输入/输出
VeChn	VENC Channel 号。	输入
pstFrmLostParam	编码通道丢帧策略的参数。	输入

【返回值】

返回值	描述
0	成功
非 0	失败，其值为错误码 。

【需求】

- 头文件：cvi_comm_venc.h、cvi_venc.h
- 库文件：libvenc.a

【注意】

- 丢帧策略只支持 H.264/H.265 编码协议。
- 丢帧策略配置由四个参数决定
  - bFrmLostOpen：当码率超过阈值时开启丢帧，确保区间码率尖峰值不会过高。
  - enFrmLostMode：仅支持编码为 PSkip 帧之丢帧方式
  - u32FrmLostBpsThr：依照系统能力设置，建议至少设置码率的 1.2 倍以上
  - u32EncFrmGaps：限制最大连续丢帧个数可使丢帧时期之画面较顺畅，区间码率尖峰值可能较高。设置为 0 时表示为不限制连续丢帧数，表示瞬时码率超出阈值允许一直丢帧，直到瞬时码率不大于阈值
- 用户可以选择性调用，默认为不开启丢帧策略
- 本接口在编码通道创建之后，编码通道销毁之前设置。
- pskip 帧仅支持 GOP 模式为 VENC_GOPMODE_NORMALP 的编码通道。如当前帧 OSD 使能且 OSD 有更新，不能编码为 Pskip 帧。
- 使能跳帧参考模式时，不建议开启丢帧策略
- 仅支持 H.264 / H.265 的 CBR / VBR 码率控制模式

【举例】

- 无

7.3.26 CVI_VENC_GetFrameLostStrategy

【描述】

设置编码通道瞬时码率超过阈值时丢帧策略。

【语法】

```
CVI_S32 CVI_VENC_GetFrameLostStrategy(VENC_CHN VeChn, VENC_FRAMELOST_S␣
↪*pstFrmLostParam);
```

【参数】

参数名称	描述	输入/输出
VeChn	VENC Channel 号。	输入
pstFrmLostParam	编码通道丢帧策略的参数。	输出

【返回值】

返回值	描述
0	成功
非 0	失败，其值为错误码 。

【需求】

- 头文件：cvi_comm_venc.h、cvi_venc.h
- 库文件：libvenc.a

【注意】

- 无

【举例】

- 无

7.3.27 CVI_VENC_SetModParam

【描述】

设置编码相关的模块参数。

【语法】

```
CVI_S32 CVI_VENC_SetModParam(const VENC_PARAM_MOD_S *pstModParam);
```

【参数】

参数名称	描述	输入/输出
pstModParam	编码模块参数指针。	输入

【返回值】

返回值	描述
0	成功
非 0	失败，其值为错误码 。

【需求】

- 头文件：cvi_comm_venc.h、cvi_venc.h
- 库文件：libvenc.a

【注意】

- 此接口可以在通道创建前后调用。
- 此接口主要用于设置对应编码 VB pool 获取方式，用户可透过结构体VENC_PARAM_MOD_S 内的 VB_SOURCE_E 类型变量设定 VB mode。

【举例】

- 无

7.3.28 CVI_VENC_GetModParam

【描述】

获取编码相关的模块参数。

【语法】

```
CVI_S32 CVI_VENC_GetModParam(VENC_PARAM_MOD_S *pstModParam);
```

【参数】

参数名称	描述	输入/输出
pstModParam	编码模块参数指针。	输出

【返回值】

返回值	描述
0	成功
非 0	失败，其值为错误码 。

【需求】

- 头文件：cvi_comm_venc.h、cvi_venc.h
- 库文件：libvenc.a

【注意】

- 通常在CVI_VENC_SetModParam 调用前搭配使用。

【举例】

- 无

7.3.29 CVI_VENC_AttachVbPool

【描述】

将编码通道绑定到某个视频缓存 VB 池中。

【语法】

```
CVI_S32 CVI_VENC_AttachVbPool(VENC_CHN VeChn, const VENC_CHN_POOL_S *pstPool);
```

【参数】

参数名称	描述	输入/输出
VeChn	通道号。	输入
pstPool	视频缓存 VB 池的 Id 号。	输入

【返回值】

返回值	描述
0	成功
非 0	失败，其值为错误码 。

【需求】

- 头文件：cvi_comm_venc.h、cvi_venc.h、cvi_comm_vb.h
- 库文件：libvenc.a

【注意】

- 必须保证通道已创建，否则会返回 CVI_FAILURE 的错误码。
- 用户必须调用接口 CVI_VB_CreatePool 创建一个视频缓存 VB 池，再通过调用接口CVI_VENC_AttachVbPool 把当前编码通道绑定到固定 PoolId 的 VB 池中。可以把多个编码通道绑定到同一个 VB 池中，但是不能把同一个编码通道绑定到多个 VB 池中。
- pstPool 必须是已创建 VB 池的有效 PoolId，包含存储 Picture 的 VB 池和存储存储 Picture 信息的 VB 池。
- 只有 H.264 / H.265 编码支持 UserVB 池方式。如果当前编码帧存分配方式使用的不是编码 UserVB 池，则返回 CVI_FAILURE。
- 用户调用此接口，必须确定透过CVI_VENC_SetModParam 设定在 VB_SOURCE_USER 模式。

【举例】

- 无

7.3.30 CVI_VENC_DetachVbPool

【描述】

将编码通道从某个视频缓存 VB 池中解绑定。

【语法】

```
CVI_S32 CVI_VENC_DetachVbPool(VENC_CHN VeChn);
```

【参数】

参数名称	描述	输入/输出
VeChn	通道号。	输入

【返回值】

返回值	描述
0	成功
非 0	失败，其值为错误码 。

【需求】

- 头文件：cvi_comm_venc.h、cvi_venc.h、cvi_comm_vb.h
- 库文件：libvenc.a

【注意】

- 必须保证通道已创建，否则会返回 CVI_FAILURE 的错误码。

【举例】

- 无

7.3.31 CVI_VENC_GetH264Entropy

【描述】

获取 H264 熵编码信息

【语法】

```
CVI_S32 CVI_VENC_GetH264Entropy(VENC_CHN VeChn, VENC_H264_ENTROPY_S␣
↪*pstH264EntropyEnc);
```

【参数】

参数名称	描述	输入/输出
VeChn	通道号	输入/输出
pstH264EntropyEnc	熵编码结构体指针	输出

【返回值】

返回值	描述
0	成功
非 0	失败，其值为错误码 。

【需求】

- 头文件：cvi_comm_venc.h、cvi_venc.h
- 库文件：libvenc.a

【注意】

- 必须保证通道已创建，否则会返回 CVI_FAILURE 的错误码。

【举例】

- 无

7.3.32 CVI_VENC_SetH264Entropy

【描述】

设置 H264 熵编码信息

【语法】

```
CVI_S32 CVI_VENC_SetH264Entropy(VENC_CHN VeChn, const VENC_H264_ENTROPY_S
↪*pstH264EntropyEnc);
```

【参数】

参数名称	描述	输入/输出
VeChn	通道号。	输入
pstH264EntropyEnc	熵编码结构体指针	输入

【返回值】

返回值	描述
0	成功
非 0	失败，其值为错误码 。

【需求】

- 头文件：cvi_comm_venc.h、cvi_venc.h
- 库文件：libvenc.a

【注意】

- 必须保证通道已创建，否则会返回 CVI_FAILURE 的错误码。

【举例】

- 无

7.3.33 CVI_VENC_InsertUserData

【描述】

插入用户数据

【语法】

```
CVI_S32 CVI_VENC_InsertUserData(VENC_CHN VeChn, CVI_U8 *pu8Data, CVI_U32 u32Len);
```

【参数】

参数名称	描述	输入/输出
VeChn	通道号。	输入
pu8Data	数据首地址	输入
u32Len	数据长度	输入

【返回值】

返回值	描述
0	成功
非 0	失败，其值为错误码 。

【需求】

- 头文件：cvi_comm_venc.h、cvi_venc.h
- 库文件：libvenc.a

【注意】

- 必须保证通道已创建，否则会返回 CVI_FAILURE 的错误码。

【举例】

- 无

7.3.34 CVI_VENC_GetCuPrediction

【描述】

获取 CU 单元的预测信息

【语法】

```
CVI_S32 CVI_VENC_GetCuPrediction(VENC_CHN VeChn, VENC_CU_PREDICTION_S␣
↪*pstCuPrediction);
```

【参数】

参数名称	描述	输入/输出
VeChn	通道号	输入
pstCuPrediction	预测信息结构体指针	输出

【返回值】

返回值	描述
0	成功
非 0	失败，其值为错误码 。

【需求】

- 头文件：cvi_comm_venc.h、cvi_venc.h
- 库文件：libvenc.a

【注意】

- 必须保证通道已创建，否则会返回 CVI_FAILURE 的错误码。
- 此接口目前仅对 H264 编码有效

【举例】

- 无

7.3.35 CVI_VENC_SetCuPrediction

【描述】

设置 CU 单元的预测信息

【语法】

```
CVI_S32 CVI_VENC_SetCuPrediction(VENC_CHN VeChn, VENC_CU_PREDICTION_S␣
↪*pstCuPrediction);
```

【参数】

参数名称	描述	输入/输出
VeChn	通道号	输入
pstCuPrediction	预测结构体指针	输入

【返回值】

返回值	描述
0	成功
非 0	失败，其值为错误码 。

【需求】

- 头文件：cvi_comm_venc.h、cvi_venc.h
- 库文件：libvenc.a

【注意】

- 必须保证通道已创建，否则会返回 CVI_FAILURE 的错误码。

- 此接口目前仅对 H264 编码有效

【举例】

无

7.3.36 CVI_VENC_GetH264Trans

【描述】

获取 H.264 协议编码通道的变换、量化属性。

【语法】

```
CVI_S32 CVI_VENC_GetH264Trans(VENC_CHN VeChn, VENC_H264_TRANS_S␣
↪*pstH264Trans);
```

【参数】

参数名称	描述	输入/输出
VeChn	通道号。	输入
pstH264Trans	H.264 协议编码通道的变换、量化参数。	输出

【返回值】

返回值	描述
0	成功
非 0	失败，其值为错误码 。

【需求】

- 头文件：cvi_comm_venc.h、cvi_venc.h
- 库文件：libvenc.a

【注意】

- 必须保证通道已创建，否则会返回 CVI_FAILURE 的错误码。
- 此接口目前仅对 H264 编码有效

【举例】

- 无

7.3.37 CVI_VENC_SetH264Trans

【描述】

设置 H.264 协议编码通道的变换、量化属性

【语法】

```
CVI_S32 CVI_VENC_SetH264Trans(VENC_CHN VeChn, const VENC_H264_TRANS_S␣
↪*pstH264Trans);
```

【参数】

参数名称	描述	输入/输出
VeChn	通道号。	输入
pstH264Trans	H.264 协议编码通道的变换、量化参数。	输出

【返回值】

返回值	描述
0	成功
非 0	失败，其值为错误码 。

【需求】

- 头文件：cvi_comm_venc.h、cvi_venc.h
- 库文件：libvenc.a

【注意】

- 必须保证通道已创建，否则会返回 CVI_FAILURE 的错误码。
- 此接口目前仅对 H264 编码有效

【举例】

- 无

7.3.38 CVI_VENC_GetH265Trans

【描述】

获取 H.265 协议编码通道的变换、量化属性

【语法】

```
CVI_S32 CVI_VENC_GetH265Trans(VENC_CHN VeChn, VENC_H265_TRANS_S␣
↪*pstH265Trans);
```

【参数】

参数名称	描述	输入/输出
VeChn	通道号。	输入
pstH265Trans	H.265 协议编码通道的变换、量化参数。	输出

【返回值】

返回值	描述
0	成功
非 0	失败，其值为错误码 。

【需求】

- 头文件：cvi_comm_venc.h、cvi_venc.h
- 库文件：libvenc.a

【注意】

- 必须保证通道已创建，否则会返回 CVI_FAILURE 的错误码。
- 此接口目前仅对 H265 编码有效

【举例】

- 无

7.3.39 CVI_VENC_SetH265Trans

【描述】

设置 H.265 协议编码通道的变换、量化属性

【语法】

```
CVI_S32 CVI_VENC_SetH265Trans(VENC_CHN VeChn, const VENC_H265_TRANS_S␣
↪*pstH265Trans);
```

【参数】

参数名称	描述	输入/输出
VeChn	通道号。	输入
pstH265Trans	H.265 协议编码通道的变换、量化参数。	输出

【返回值】

返回值	描述
0	成功
非 0	失败，其值为错误码 。

【需求】

- 头文件：cvi_comm_venc.h、cvi_venc.h

- 库文件：libvenc.a

【注意】

- 必须保证通道已创建，否则会返回 CVI_FAILURE 的错误码。
- 此接口目前仅对 H265 编码有效

【举例】

- 无

7.4 数据类型

相关数据类型、数据结构定义如下：

- VENC_MAX_CHN_NUM：定义最大通道数。
- VENC_CHN_PARAM_S：定义 Venc 通道参数结构体。
- VENC_PACK_S：定义帧码流包结构体。
- VENC_STREAM_S：定义帧码流类型结构体。
- VENC_ATTR_S：定义编码器属性结构体。
- VENC_GOP_ATTR_S：定义 Gop Mode 类型的结构体。
- VENC_GOP_NORMALP_S: 定义 NormalP 结构体
- VENC_GOP_SMARTP_S: 定义 smartP 结构体
- VENC_CHN_ATTR_S：定义编码通道属性结构体。
- VENC_RECV_PIC_PARAM_S：定义编码通道连续接收并编码的帧数结构体。
- VENC_CHN_STATUS_S：定义编码通道的状态结构体。
- VENC_JPEG_PARAM_S：定义 JPEG 编码参数集合。
- VENC_RC_ATTR_S：定义编码通道码率控制器属性。
- 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_H264_QVBR_S: 定义 H.264 编码通道 QVBR 属性结构。
- VENC_H264_FIXQP_S：定义 H.264 编码通道 Fixqp 属性结构。
- VENC_H264_QPMAP_S：定义 H.264 编码通道 QPMAP 属性结构。
- VENC_MJPEG_FIXQP_S：定义 MJPEG 编码通道 Fixqp 属性结构。
- 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_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_H265_AVBR_S：定义 H.265 编码通道 AVBR 属性结构。

- VENC_H265_QVBR_S: 定义 H.265 编码通道 QVBR 属性结构。
- VENC_H265_FIXQP_S： 定义 H.265 编码通道 Fixqp 属性结构。
- VENC_H265_QPMAP_S： 定义 H.265 编码通道 QPMAP 属性结构。
- VENC_RC_PARAM_S： 定义编码通道的码率控制高级参数。
- VENC_PARAM_H264_CBR_S: 定义 H264 CBR 高级参数
- VENC_PARAM_H264_VBR_S: 定义 H264 VBR 高级参数
- VENC_PARAM_H264_AVBR_S: 定义 H264 AVBR 高级参数
- VENC_PARAM_H265_CBR_S: 定义 H265 CBR 高级参数
- VENC_PARAM_H265_VBR_S: 定义 H265 VBR 高级参数
- VENC_PARAM_H265_AVBR_S: 定义 H265 AVBR 高级参数
- VENC_CHN_POOL_S： 定义编码通道绑定的 VB 池结构体。
- VENC_H264_ENTROPY_S： 定义 H264 熵编码结构体
- VENC_CU_PREDICTION_S: 定义 CU 预测结构体

7.4.1 VENC_MAX_CHN_NUM

【说明】

定义编码最大信道个数。

【定义】

```
#define VENC_MAX_CHN_NUM          64
```

【成员】

无。

【注意事项】

由于最大通道个数涉及到内存的分配、目前不开放扩充。

【相关数据类型及接口】

无。

7.4.2 VENC_CHN_PARAM_S

【说明】

定义 Venc 通道参数结构体。

【定义】

```
typedef struct _VENC_CHN_PARAM_S {
  CVI_BOOL bColor2Grey;
  CVI_U32 u32Priority
  CVI_U32 u32MaxStrmCnt;
  CVI_U32 u32PollWakeUpFrmCnt;
  VENC_CROP_INFO_S stCropCfg;
  VENC_FRAME_RATE_S stFrameRate;
} VENC_CHN_PARAM_S;
```

【成员】

成员名称	描述
bColor2Grey	预留，暂未使用
u32Priority	预留，暂未使用
u32MaxStrmCnt	预留，暂未使用
u32PollWakeUpFrmCnt	预留，暂未使用
stCropCfg	通道截取（Crop）参数
stFrameRate	通道帧率控制参数

【注意事项】

无。

【相关数据类型及接口】

- CVI_VENC_SetChnParam

7.4.3 VENC_PACK_S

【说明】

定义帧码流包结构体。

【定义】

```
typedef struct _VENC_PACK_S {
  CVI_U64 u64PhyAddr;
  CVI_U8 ATTRIBUTE *pu8Addr;
  CVI_U32 ATTRIBUTE u32Len;
  CVI_U64 u64PTS;
  CVI_BOOL bFrameEnd;
  VENC_DATA_TYPE_U DataType;
  CVI_U32 u32Offset;
  CVI_U32 u32DataNum;
```

（下页继续）

(续上页)

```
  VENC_PACK_INFO_S stPackInfo[8];
} VENC_PACK_S;
```

【成员】

成员名称	描述
u64PhyAddr	码流包首地址。
pu8Addr	码流包首虚拟地址。
u32Len	码流包长度。
DataType	码流类型，支持 H.264/JPEG/ H.265 协议类型的数据包。
bFrameEnd	帧结束标识。 CVI_TRUE：该码流包是该帧的最后一个包。 CVI_FALSE：该码流包不是该帧的最后一个包。
u32Offset	码流包中有效数据与码流包首地址 pu8Addr 的偏移。
u64PTS	时间戳。单位：us。
u32DataNum	当前码流包（当前包的类型由 DataType 指定）数据中包含其他类型码流包的个数。
stPackInfo[8]	当前码流包数据中包含其他类型码流包数据信息。

【注意事项】

无。

【相关数据类型及接口】

- VENC_STREAM_S

7.4.4 VENC_STREAM_S

【说明】

定义帧码流类型结构体。

【定义】

```
typedef struct _VENC_STREAM_S {
  VENC_PACK_S ATTRIBUTE *pstPack;
  CVI_U32 ATTRIBUTE u32PackCount;
  CVI_U32 u32Seq;

  union {
  VENC_STREAM_INFO_H264_S stH264Info;
  VENC_STREAM_INFO_JPEG_S stJpegInfo;
  VENC_STREAM_INFO_H265_S stH265Info;
  VENC_STREAM_INFO_PRORES_S stProresInfo;
  };
```

(下页继续)

(续上页)

```
  union {
  VENC_STREAM_ADVANCE_INFO_H264_S stAdvanceH264Info;
  VENC_STREAM_ADVANCE_INFO_JPEG_S stAdvanceJpegInfo;
  VENC_STREAM_ADVANCE_INFO_H265_S stAdvanceH265Info;
  VENC_STREAM_ADVANCE_INFO_PRORES_S stAdvanceProresInfo;
  };
} VENC_STREAM_S;
```

【成员】

成员名称	描述
pstPack	帧码流包结构。
u32PackCount	一帧码流的所有包的个数。
u32Seq	码流序列号。
stH264Info/stJpegInfo/stH265Info/stProresInfo	码流信息。
stAdvanceH264Info/stAdvanceJpegInfo/stAdvanceH265Info/stAdvanceProresInfo	码流高阶信息。

【注意事项】

无。

【相关数据类型及接口】

- VENC_PACK_S

7.4.5 VENC_GOP_ATTR_S

【说明】

定义编码器 GOP 属性结构体。

【定义】

```
typedef struct _VENC_GOP_ATTR_S {
  VENC_GOP_MODE_E enGopMode;
  union {
    VENC_GOP_NORMALP_S stNormalP;
    VENC_GOP_DUALP_S stDualP;
    VENC_GOP_SMARTP_S stSmartP;
    VENC_GOP_ADVSMARTP_S stAdvSmartP;
    VENC_GOP_BIPREDB_S stBipredB;
  };

} VENC_GOP_ATTR_S;
```

【成员】

成员名称	描述
enGopMode	编码 GOP 类型。
stNormalP	编码单参考帧 P 帧 GOP 属性结构体。
stDualP	预留，暂未使用
stSmartP	编码智能 P 帧 GOP 属性结构体。
stAdvSmartP	预留，暂未使用
stBipredB	预留，暂未使用

【注意事项】

无。

【相关数据类型及接口】

- CVI_VENC_CreateChn
- VENC_STREAM_S

7.4.6 VENC_GOP_NORMALP_S

【说明】

定义编码器 NormalP GOP 属性结构体。

【定义】

```
typedef struct _VENC_GOP_NORMALP_S {
  CVI_S32 s32IPQpDelta;
} VENC_GOP_NORMALP_S;
```

【成员】

成员名称	描述
s32IPQpDelta	I 帧和 P 帧的 QP 差值

【注意事项】

无。

【相关数据类型及接口】

- CVI_VENC_CreateChn

7.4.7 VENC_GOP_SMARTP_S

【说明】

定义编码器 SmartP GOP 属性结构体。

【定义】

```
typedef struct _VENC_GOP_SMARTP_S {
  CVI_U32 u32BgInterval;
  CVI_S32 s32BgQpDelta;
  CVI_S32 s32ViQpDelta;
} VENC_GOP_SMARTP_S;
```

【成员】

成员名称	描述
u32BgInterval	IDR 帧的间隔
s32BgQpDelta	IDR 帧和 P 帧的 QP 差值
s32ViQpDelta	Vi 帧和 P 帧的 QP 差值

【注意事项】

无。

【相关数据类型及接口】

- CVI_VENC_CreateChn

7.4.8 VENC_RECV_PIC_PARAM_S

【说明】

定义编码通道连续接收并编码的帧数结构体。

【定义】

```
typedef struct _VENC_RECV_PIC_PARAM_S {
  CVI_S32 s32RecvPicNum;
} VENC_RECV_PIC_PARAM_S;
```

【成员】

成员名称	描述
s32RecvPicNum	编码通道连续接收并编码的帧数。 范围：[-1,0) ∪ (0 ∞]

【注意事项】

无。

【相关数据类型及接口】

- CVI_VENC_StartRecvFrame

7.4.9 VENC_CHN_ATTR_S

【说明】

定义 VENC_CHN_ATTR_S 属性。

【定义】

```
typedef struct _VENC_CHN_ATTR_S {
  VENC_ATTR_S stVencAttr;
  VENC_RC_ATTR_S stRcAttr;
  VENC_GOP_ATTR_S stGopAttr;
} VENC_CHN_ATTR_S;
```

【成员】

成员名称	描述
stVencAttr	venc 属性。
stRcAtt	码率控制器属性。
stGopAttr	Gop Mode 类型的结构体。请参考上述 typedef struct _VENC_GOP_ATTR_S

【注意事项】

无。

【相关数据类型及接口】

- CVI_VENC_CreateChn

7.4.10 VENC_ATTR_S

【说明】

定义 VENC_ATTR_S 属性。

【定义】

```
typedef struct _VENC_ATTR_S {
  PAYLOAD_TYPE_E enType;
  CVI_U32 u32MaxPicWidth;
  CVI_U32 u32MaxPicHeight;
  CVI_U32 u32BufSize;
  CVI_U32 u32Profile;
  CVI_BOOL bByFrame;
  CVI_U32 u32PicWidth;
  CVI_U32 u32PicHeight;
  CVI_BOOL bSingleCore;
  CVI_BOOL bEsBufQueueEn;
  CVI_BOOL bIsoSendFrmEn;
  union {
  VENC_ATTR_H264_S stAttrH264e;
  VENC_ATTR_H265_S stAttrH265e;
```

(下页继续)

(续上页)

```
  VENC_ATTR_JPEG_S stAttrJpege;
  VENC_ATTR_PRORES_S stAttrProres;
  };
} VENC_ATTR_S;
```

【成员】

成员名称	描述
enType	payload 类型
u32MaxPicWidth	影像最大编码宽度
u32MaxPicHeight	影像最大编码高度
u32BufSize	Encoded bitstream buffer 大小
u32Profile	编码的等级
bByFrame	Encoded bitstream 收集方式 CVI_TRUE: 以帧为主 CVI_FALSE: 以封包为主
u32PicWidth	编码影像宽度
u32PicHeight	编码影像高度
bSingleCore	只使用 H264/H265
bEsBufQueueEn	使用 es buffer queue
bIsoSendFrmEn	去除 SendFrame/GetStream 必须配对使用的限制
stAttrH264e / stAttrH265e /stAttrJpege / stAttrProres	Encoder 属性

【注意事项】

打开 bIsoSendFrmEn 的同时应该打开 bEsBufQueueEn，否则会有花屏现象

双系统 bIsoSendFrmEn 必须关闭，双系统默认去除了 SendFrame/GetStream 必须配对使用的限制

【相关数据类型及接口】

无。

7.4.11 VENC_ATTR_H264_S

【说明】

定义 H264 编码属性。

【定义】

```
typedef struct _VENC_ATTR_H264_S {
  CVI_BOOL bRcnRefShareBuf;
  CVI_BOOL bSingleLumaBuf;
} VENC_ATTR_H264_S;
```

【成员】

成员名称	描述
bRcnRefShareBuf	预留
bSingleLumaBuf	一个编码通道的 FrameBuffer 使用一个 Luma Buffer

【注意事项】

无。

【相关数据类型及接口】

无。

7.4.12 VENC_ATTR_H265_S

【说明】

定义 H265 编码属性。

【定义】

```
typedef struct _VENC_ATTR_H265_S {
  CVI_BOOL bRcnRefShareBuf;
} VENC_ATTR_H265_S;
```

【成员】

成员名称	描述
bRcnRefShareBuf	预留

【注意事项】

无。

【相关数据类型及接口】

无。

7.4.13 VENC_STREAM_INFO_S

【说明】

定义 VENC_STREAM_INFO_S 属性。

【定义】

```
typedef struct _ VENC_STREAM_INFO_S {
  H265E_REF_TYPE_E enRefType;

  CVI_U32 u32PicBytesNum;
```

(下页继续)

(续上页)

```
  CVI_U32 u32PicCnt;
  CVI_U32 u32StartQp;
  CVI_U32 u32MeanQp;
  CVI_BOOL bPSkip;

  CVI_U32 u32ResidualBitNum;
  CVI_U32 u32HeadBitNum;
  CVI_U32 u32MadiVal;
  CVI_U32 u32MadpVal;
  CVI_U32 u32MseSum;
  CVI_U32 u32MseLcuCnt;
  double dPSNRVal;
} VENC_STREAM_INFO_S;
```

【成员】

成员名称	描述
u32MeanQp	目前 Frame 的平均 QP

【注意事项】

无。

【相关数据类型及接口】

无。

- CVI_VENC_QueryStatus

7.4.14 VENC_CHN_STATUS_S

【说明】

定义 VENC_CHN_STATUS_S 属性。

【定义】

```
typedef struct _VENC_CHN_STATUS_S {
  CVI_U32 u32LeftPics;
  CVI_U32 u32LeftStreamBytes;
  CVI_U32 u32LeftStreamFrames;
  CVI_U32 u32CurPacks;
  CVI_U32 u32LeftRecvPics;
  CVI_U32 u32LeftEncPics;
  CVI_BOOL bJpegSnapEnd;
  VENC_STREAM_INFO_S stVencStrmInfo;
} VENC_CHN_STATUS_S;
```

【成员】

成员名称	描述
u32LeftPics	预留
u32LeftStreamBytes	预留
u32LeftStreamFrames	VENC 剩余待接收的帧数
u32CurPacks	目前 Frame 中的 Packet 数
u32LeftRecvPics	预留
u32LeftEncPics	预留
bJpegSnapEnd	预留
stVencStrmInfo	预留

【注意事项】

无。

【相关数据类型及接口】

- CVI_VENC_QueryStatus

7.4.15 VENC_JPEG_PARAM_S

【说明】

定义 JPEG 协议编码通道高级参数结构体。

【定义】

```
typedef struct _VENC_JPEG_PARAM_S {
  CVI_U32 u32Qfactor;
  CVI_U8 u8YQt[64];
  CVI_U8 u8CbQt[64];
  CVI_U8 u8CrQt[64];
  CVI_U32 U32MCUPerECS;
} VENC_JPEG_PARAM_S;
```

【成员】

成员名称	描述
u32Qfactor	具体含义请参见 RFC2435 协议，系统默认为 0。
u8YQt	Y 量化表。(未实现)
u8CbQt	Cb 量化表。(未实现)
u8CrQt	Cr 量化表。(未实现)
u32MCUPerECS	每个 ECS 中包含多少个 MCU，系统默认为 0，表示不划分 Ecs。(未实现)

【注意事项】

无。

【相关数据类型及接口】

- CVI_VENC_SetJpegParam

- CVI_VENC_GetJpegParam

7.4.16 VENC_RC_ATTR_S

【说明】

定义编码通道码率控制器属性。

【定义】

```
typedef struct _VENC_RC_ATTR_S {
  VENC_RC_MODE_E enRcMode;
  union {
    VENC_H264_CBR_S stH264Cbr;
    VENC_H264_VBR_S stH264Vbr;
    VENC_H264_AVBR_S stH264AVbr;
    VENC_H264_QVBR_S stH264QVbr;
    VENC_H264_FIXQP_S stH264FixQp;
    VENC_H264_QPMAP_S stH264QpMap;

    VENC_MJPEG_CBR_S stMjpegCbr;
    VENC_MJPEG_VBR_S stMjpegVbr;
    VENC_MJPEG_FIXQP_S stMjpegFixQp;

    VENC_H265_CBR_S stH265Cbr;
    VENC_H265_VBR_S stH265Vbr;
    VENC_H265_AVBR_S stH265AVbr;
    VENC_H265_QVBR_S stH265QVbr;
    VENC_H265_FIXQP_S stH265FixQp;
    VENC_H265_QPMAP_S stH265QpMap;
  };
} VENC_RC_ATTR_S;
```

【成员】

成员名称	描述
enRcMode	RC 模式。
stH264Cbr	H.264 协议编码信道 Cbr 模式属性。
stH264Vbr	H.264 协议编码信道 Vbr 模式属性。
stH264AVbr	H.264 协议编码信道 AVbr 模式属性。
stH264QVbr	H.264 协议编码信道 QVbr 模式属性。
stH264CVbr	H.264 协议编码信道 CVbr 模式属性。
stH264FixQp	H.264 协议编码信道 Fixqp 模式属性。
stH264QpMap	H.264 协议编码信道 QPMAP 模式属性。
stMjpegeFixQp	Mjpeg 协议编码信道 Fixqp 模式属性。
stMjpegeCbr	Mjpeg 协议编码信道 cbr 模式属性。
stMjpegeVbr	Mjpeg 协议编码信道 vbr 模式属性
stH265Cbr	H.265 协议编码信道 Cbr 模式属性。
stH265Vbr	H.265 协议编码信道 Vbr 模式属性。
stH265AVbr	H.265 协议编码信道 AVbr 模式属性。
stH265QVbr	H.265 协议编码信道 QVbr 模式属性。
stH265CVbr	H.265 协议编码信道 CVbr 模式属性。
stH265FixQp	H.265 协议编码信道 Fixqp 模式属性。
stH265QpMap	H.265 协议编码信道 QPMAP 模式属性。

【注意事项】

无。

【相关数据类型及接口】

- CVI_VENC_CreateChn

7.4.17 VENC_H264_CBR_S

【说明】

定义 H.264 编码通道 CBR 属性结构。

【定义】

```
typedef struct _VENC_H264_CBR_S {
  CVI_U32 u32Gop;
  CVI_U32 u32StatTime;
  CVI_U32 u32SrcFrameRate
  CVI_FR32 fr32DstFrameRate;
  CVI_U32 u32BitRate;
  CVI_BOOL bVariFpsEn;
} VENC_H264_CBR_S;
```

【成员】

成员名称	描述
u32Gop	H.264 gop 值。范围：[1, 65536]。
u32StatTime	CBR 码率统计时间，以秒为单位。范围：[1, 60]。
u32SrcFrameRate	输入帧率，以 fps 为单位。
fr32DstFrameRate	编码器输出帧率，以 fps 为单位。
u32BitRate	平均 bitrate，以 kbps 为单位。
bVariFpsEn	开启 Variable FPS。开启此功能后，会同时开闭 Frame rate conversion 功能。

【注意事项】

无。

【相关数据类型及接口】

- CVI_VENC_CreateChn

7.4.18 VENC_H264_VBR_S

【说明】

定义 H.264 编码通道 VBR 属性结构。

【定义】

```
typedef struct _VENC_H264_VBR_S {
  CVI_U32 u32Gop;
  CVI_U32 u32StatTime;
  CVI_U32 u32SrcFrameRate;
  CVI_FR32 fr32DstFrameRate;
  CVI_U32 u32MaxBitRate;
  CVI_BOOL bVariFpsEn;
} VENC_H264_VBR_S;
```

【成员】

成员名称	描述
u32Gop	H.264 gop 值。范围：[1, 65536]。
u32StatTime	VBR 码率统计时间，以秒为单位。。范围：[1, 60]。
u32SrcFrameRate	输入帧率，以 fps 为单位。
fr32DstFrameRate	编码器输出帧率，以 fps 为单位。
u32MaxBitRate	编码器输出最大码率，以 kbps 为单位。
bVariFpsEn	开启 Variable FPS。开启此功能后，会同时开闭 Frame rate conversion 功能。

【注意事项】

无。

【相关数据类型及接口】

· CVI_VENC_CreateChn

7.4.19 VENC_H264_AVBR_S

【说明】

定义 H.264 编码通道 AVBR 属性结构。

【定义】

```
typedef struct _VENC_H264_AVBR_S {
  CVI_U32 u32Gop;
  CVI_U32 u32StatTime;
  CVI_U32 u32SrcFrameRate;
  CVI_FR32 fr32DstFrameRate;
  CVI_U32 u32MaxBitRate;
  CVI_BOOL bVariFpsEn;
} VENC_H264_AVBR_S;
```

【成员】

成员名称	描述
u32Gop	H.264 gop 值。
u32StatTime	AVBR 码率统计时间，以秒为单位。
u32SrcFrameRate	VI 输入帧率，以 fps 为单位。
fr32DstFrameRate	编码器输出帧率，以 fps 为单位。
u32MaxBitRate	编码器输出最大码率，以 kbps 为单位。
bVariFpsEn	开启 Variable FPS。开启此功能后，会同时开闭 Frame rate conversion 功能。

【注意事项】

无。

【相关数据类型及接口】

· CVI_VENC_CreateChn

7.4.20 VENC_H264_FIXQP_S

【说明】

定义 H.264 编码通道 Fixqp 属性结构。

【定义】

```
typedef struct _VENC_H264_FIXQP_S {
  CVI_U32 u32Gop;
  CVI_U32 u32SrcFrameRate;
  CVI_FR32 fr32DstFrameRate
  CVI_U32 u32IQp;
```

(下页继续)

(续上页)

```
  CVI_U32 u32PQp;
  CVI_U32 u32BQp;
  CVI_BOOL bVariFpsEn;
} VENC_H264_FIXQP_S;
```

【成员】

成员名称	描述
u32Gop	H.264 gop 值。范围：[1, 65536]。
u32IQp	I 帧所有宏块 Qp 值。
u32SrcFrameRate	输入帧率，以 fps 为单位。
fr32DstFrameRate	编码器输出帧率，以 fps 为单位。
u32PQp	P 帧所有宏块 Qp 值。
u32BQp	B 帧所有宏块 Qp 值。
bVariFpsEn	开启 Variable FPS。开启此功能后，会同时开闭 Frame rate conversion 功能。

【注意事项】

无。

【相关数据类型及接口】

- CVI_VENC_CreateChn

7.4.21 VENC_H264_QPMAP_S

【说明】

定义 H.264 编码通道 QPMAP 属性结构。

【定义】

```
typedef struct _VENC_H264_QPMAP_S {
  CVI_U32 u32Gop;
  CVI_U32 u32StatTime;
  CVI_U32 u32SrcFrameRate;
  CVI_FR32 fr32DstFrameRate;
  CVI_BOOL bVariFpsEn;
} VENC_H264_QPMAP_S;
```

【成员】

成员名称	描述
u32Gop	H.264 gop 值。范围：[1, 65536]。
u32StatTime	QPMAP 码率统计时间，以秒为单位。
u32SrcFrameRate	输入帧率，以 fps 为单位。
fr32DstFrameRate	编码器输出帧率，以 fps 为单位。
bVariFpsEn	开启 Variable FPS。开启此功能后，会同时开闭 Frame rate conversion 功能。

【注意事项】

无。

【相关数据类型及接口】

- CVI_VENC_CreateChn

7.4.22 VENC_MJPEG_FIXQP_S

【说明】

定义 MJPEG 编码通道 Fixqp 属性结构。

【定义】

```
typedef struct _VENC_MJPEG_FIXQP_S {
  CVI_U32 u32SrcFrameRate
  CVI_FR32 fr32DstFrameRate;
  CVI_U32 u32Qfactor;
  CVI_BOOL bVariFpsEn;
} VENC_MJPEG_FIXQP_S;
```

【成员】

成员名称	描述
u32SrcFrameRate	VI 输入帧率，以 fps 为单位。
fr32DstFrameRate	编码器输出帧率，以 fps 为单位。
u32Qfactor	MJPEG 编码的 Qfactor。
bVariFpsEn	开启 Variable FPS。开启此功能后，会同时开闭 Frame rate conversion 功能。

【注意事项】

无。

【相关数据类型及接口】

- CVI_VENC_CreateChn

7.4.23 VENC_MJPEG_CBR_S

【说明】

定义 MJPEG 编码通道 CBR 属性结构。

【定义】

```
typedef struct _VENC_MJPEG_CBR_S {
  CVI_U32 u32StatTime;
  CVI_U32 u32SrcFrameRate;
  CVI_FR32 fr32DstFrameRate;
  CVI_U32 u32BitRate;
```

(下页继续)

(续上页)

```
  CVI_BOOL bVariFpsEn;
} VENC_MJPEG_CBR_S;
```

【成员】

成员名称	描述
u32StatTime	CBR 码率统计时间，以秒为单位。
u32SrcFrameRate	编码器输入帧率，以 fps 为单位。
fr32DstFrameRate	编码器输出帧率，以 fps 为单位。
u32BitRate	平均 bitrate，以 kbps 为单位。
bVariFpsEn	开启 Variable FPS。开启此功能后，会同时开闭 Frame rate conversion 功能。

【注意事项】

无。

【相关数据类型及接口】

- CVI_VENC_CreateChn

7.4.24 VENC_H265_CBR_S

【说明】

定义 H.265 编码通道 CBR 属性结构。

【定义】

```
typedef struct _VENC_H264_CBR_S {
  CVI_U32 u32Gop;
  CVI_U32 u32StatTime
  CVI_U32 u32SrcFrameRate;
  CVI_FR32 fr32DstFrameRate;
  CVI_U32 u32BitRate;
  CVI_BOOL bVariFpsEn;
} VENC_H264_CBR_S;
typedef struct _VENC_H264_CBR_S VENC_H265_CBR_S;
```

【成员】

成员名称	描述
u32Gop	H.265 gop 值。
u32StatTime	CBR 码率统计时间，以秒为单位。
u32SrcFrameRate	VI 输入帧率，以 fps 为单位。
fr32DstFrameRate	编码器输出帧率，以 fps 为单位。
u32BitRate	平均 bitrate，以 kbps 为单位。
bVariFpsEn	开启 Variable FPS。开启此功能后，会同时开闭 Frame rate conversion 功能。

【注意事项】

无。

【相关数据类型及接口】

- CVI_VENC_CreateChn

7.4.25 VENC_H265_VBR_S

【说明】

定义 H.265 编码通道 VBR 属性结构。

【定义】

```
typedef struct _VENC_H264_VBR_S {
  CVI_U32 u32Gop;
  CVI_U32 u32StatTime;
  CVI_U32 u32SrcFrameRate;
  CVI_FR32 fr32DstFrameRate;
  CVI_U32 u32MaxBitRate;
  CVI_BOOL bVariFpsEn;
} VENC_H264_VBR_S;
typedef struct _VENC_H264_VBR_S VENC_H265_VBR_S;
```

【成员】

成员名称	描述
u32Gop	H.265 gop 值。
u32StatTime	CBR 码率统计时间，以秒为单位。
u32SrcFrameRate	VI 输入帧率，以 fps 为单位。
fr32DstFrameRate	编码器输出帧率，以 fps 为单位。
u32MaxBitRate	编码器输出最大码率，以 kbps 为单位。
bVariFpsEn	开启 Variable FPS。开启此功能后，会同时开闭 Frame rate conversion 功能。

【注意事项】

无。

【相关数据类型及接口】

- CVI_VENC_CreateChn

7.4.26 VENC_H265_AVBR_S

【说明】

定义 H.265 编码通道 AVBR 属性结构。

【定义】

```
typedef struct _VENC_H264_AVBR_S {
  CVI_U32 u32Gop;
  CVI_U32 u32StatTime;
  CVI_U32 u32SrcFrameRate;
  CVI_FR32 fr32DstFrameRate;
  CVI_U32 u32MaxBitRate;
  CVI_BOOL bVariFpsEn;
} VENC_H264_AVBR_S;
typedef struct _VENC_H264_AVBR_S VENC_H265_AVBR_S;
```

【成员】

成员名称	描述
u32Gop	H.265 gop 值。
u32StatTime	AVBR 码率统计时间，以秒为单位。
u32SrcFrameRate	VI 输入帧率，以 fps 为单位。
fr32DstFrameRate	编码器输出帧率，以 fps 为单位。
u32MaxBitRate	编码器输出最大码率，以 kbps 为单位。
bVariFpsEn	开启 Variable FPS。开启此功能后，会同时开闭 Frame rate conversion 功能。

【注意事项】

无。

【相关数据类型及接口】

- CVI_VENC_CreateChn

7.4.27 VENC_H265_FIXQP_S

【说明】

定义 H.265 编码通道 Fixqp 属性结构。

【定义】

```
typedef struct _VENC_H264_FIXQP_S {
  CVI_U32 u32Gop;
  CVI_U32 u32SrcFrameRate;
  CVI_FR32 fr32DstFrameRate;
  CVI_U32 u32IQp;
  CVI_U32 u32PQp;
  CVI_U32 u32BQp;
  CVI_BOOL bVariFpsEn;
```

(下页继续)

(续上页)

```
} VENC_H264_FIXQP_S;
typedef struct _VENC_H264_FIXQP_S VENC_H265_FIXQP_S;
```

【成员】

成员名称	描述
u32Gop	H.265 gop 值。
u32SrcFrameRate	VI 输入帧率，以 fps 为单位。
fr32DstFrameRate	编码器输出帧率，以 fps 为单位。
u32IQp	I 帧所有宏块 Qp 值。
u32PQp	P 帧所有宏块 Qp 值。
u32BQp	预留。
bVariFpsEn	开启 Variable FPS。开启此功能后，会同时开闭 Frame rate conversion 功能。

【注意事项】

无。

【相关数据类型及接口】

- CVI_VENC_CreateChn

7.4.28 VENC_H265_QPMAP_S

【说明】

定义 H.265 编码通道 QPMAP 属性结构。

【定义】

```
typedef struct _VENC_H265_QPMAP_S {
  CVI_U32 u32Gop;
  CVI_U32 u32StatTime;
  CVI_U32 u32SrcFrameRate;
  CVI_FR32 fr32DstFrameRate;
  VENC_RC_QPMAP_MODE_E enQpMapMode;
  CVI_BOOL bVariFpsEn;
} VENC_H265_QPMAP_S;
```

【成员】

成员名称	描述
u32Gop	H.265 gop 值。
u32StatTime	QPMAP 码率统计时间，以秒为单位。
u32SrcFrameRate	VI 输入帧率，以 fps 为单位。
fr32DstFrameRate	编码器输出帧率，以 fps 为单位。
enQpMapMode	CU32 或 CU64 的 QP 值的取值方式。
bVariFpsEn	开启 Variable FPS。开启此功能后，会同时开闭 Frame rate conversion 功能。

【注意事项】

无。

【相关数据类型及接口】

- CVI_VENC_CreateChn

7.4.29 VENC_RC_PARAM_S

【说明】

定义编码通道的码率控制高级参数。码率控制主参数结构。

【定义】

```
typedef struct _VENC_RC_PARAM_S {
  CVI_U32 u32ThrdI[RC_TEXTURE_THR_SIZE];
  CVI_U32 u32ThrdP[RC_TEXTURE_THR_SIZE];
  CVI_U32 u32ThrdB[RC_TEXTURE_THR_SIZE];
  CVI_U32 u32DirectionThrd;
  CVI_S32 s32FirstFrameStartQp;
  CVI_S32 s32InitialDelay;
  CVI_U32 u32ThrdLv;
  CVI_BOOL bBgEnhanceEn;
  CVI_S32 s32BgDeltaQp;
  union {
    VENC_PARAM_H264_CBR_S stParamH264Cbr;
    VENC_PARAM_H264_VBR_S stParamH264Vbr;
    VENC_PARAM_H264_AVBR_S stParamH264AVbr;
    VENC_PARAM_H264_QVBR_S stParamH264QVbr;
    VENC_PARAM_H265_CBR_S stParamH265Cbr;
    VENC_PARAM_H265_VBR_S stParamH265Vbr;
    VENC_PARAM_H265_AVBR_S stParamH265AVbr;
    VENC_PARAM_H265_QVBR_S stParamH265QVbr;
    VENC_PARAM_MJPEG_CBR_S stParamMjpegCbr;
    VENC_PARAM_MJPEG_VBR_S stParamMjpegVbr;
  };
} VENC_RC_PARAM_S;
```

【成员】

成员名称	描述
u32ThrdI	预留
u32ThrdP	预留
u32ThrdB	预留
u32DirectionThrd	预留
u32RowQpDelta	在宏块级码率控制时，每一行宏块的起始 Qp 相对于帧起始 Qp 的波动幅度值。
s32FirstFrameStartQp	设置第一帧的起始 Qp 值，CBR / VBR / AVBR 有效
stSceneChangeDetect	预留
s32InitialDelay	影响帧编码的码率控制
u32ThrdLv	宏块级码率控制的 mad 门限
bBgEnhanceEn	预留
s32BgDeltaQp	smartP 模式下，IDR 帧与 P 帧的 QP 差值
stParamH264Cbr	H.264 通道 CBR（Constant Bit Rate）码率控制模式高级参数。
stParamH264Vbr	H.264 通道 VBR（Variable Bit Rate）码率控制模式高级参数。
stParamH264AVbr	H.264 通道 AVBR（Adaptive Variable Bit Rate）码率控制模式高级参数。
stParamH264QVbr	预留
stParamH264CVbr	预留
stParamMjpegCbr	MJPEG 信道 CBR（Constant Bit Rate）码率控制模式高级参数。
stParamMjpegVbr	预留
stParamH265Cbr	H.265 通道 CBR（Constant Bit Rate）码率控制模式高级参数。
stParamH265Vbr	H.265 通道 VBR（Variable Bit Rate）码率控制模式高级参数。
stParamH265AVbr	H.265 通道 AVBR（Adaptive Variable Bit Rate）码率控制模式高级参数。
stParamH265QVbr	预留
stParamH265CVbr	预留

【注意事项】

无。

【相关数据类型及接口】

- CVI_VENC_SetRcParam
- CVI_VENC_GetRcParam

7.4.30 VENC_PARAM_H264_CBR_S

【说明】

定义 H264 CBR 高级参数。

【定义】

```
typedef struct _VENC_PARAM_H264_CBR_S {
  CVI_U32 u32MinIprop;
  CVI_U32 u32MaxIprop;
  CVI_U32 u32MaxQp;
  CVI_U32 u32MinQp;
  CVI_U32 u32MaxIQp;
  CVI_U32 u32MinIQp;
  CVI_S32 s32MaxReEncodeTimes;
  CVI_BOOL bQpMapEn;
} VENC_PARAM_H264_CBR_S;
```

【成员】

成员名称	描述
u32MinIprop	最小 IP 比
u32MaxIprop	最大 IP 比
u32MaxQp	最大 QP
u32MinQp	最小 QP
u32MaxIQp	I 帧最大 QP
u32MinIQp	I 帧最小 QP
s32MaxReEncodeTimes	预留
CVI_BOOL bQpMapEn	预留

【注意事项】

无。

【相关数据类型及接口】

- CVI_VENC_SetRcParam
- CVI_VENC_GetRcParam

7.4.31 VENC_PARAM_H264_VBR_S

【说明】

定义 H264 VBR 高级参数。

【定义】

```
typedef struct _VENC_PARAM_H264_VBR_S {
  CVI_S32 s32ChangePos;
  CVI_U32 u32MinIprop;
  CVI_U32 u32MaxIprop;
```

(下页继续)

(续上页)

```
  CVI_S32 s32MaxReEncodeTimes;
  CVI_BOOL bQpMapEn;

  CVI_U32 u32MaxQp;
  CVI_U32 u32MinQp;
  CVI_U32 u32MaxIQp;
  CVI_U32 u32MinIQp;
} VENC_PARAM_H264_VBR_S;
```

【成员】

成员名称	描述
s32ChangePos	码率调节阈值
u32MinIprop	预留
u32MaxIprop	最大 IP 比
s32MaxReEncodeTimes	预留
CVI_BOOL bQpMapEn	预留
u32MaxQp	最大 QP
u32MinQp	最小 QP
u32MaxIQp	I 帧最大 QP
u32MinIQp	I 帧最小 QP

【注意事项】

无。

【相关数据类型及接口】

- CVI_VENC_SetRcParam
- CVI_VENC_GetRcParam

7.4.32 VENC_PARAM_H264_AVBR_S

【说明】

定义 H264 AVBR 高级参数。

【定义】

```
typedef struct _VENC_PARAM_H264_AVBR_S {
  CVI_S32 s32ChangePos;
  CVI_U32 u32MinIprop;
  CVI_U32 u32MaxIprop;
  CVI_S32 s32MaxReEncodeTimes;
  CVI_BOOL bQpMapEn;

  CVI_S32 s32MinStillPercent;
  CVI_U32 u32MaxStillQP;
  CVI_U32 u32MinStillPSNR;
```

(下页继续)

(续上页)

```
  CVI_U32 u32MaxQp;
  CVI_U32 u32MinQp;
  CVI_U32 u32MaxIQp;
  CVI_U32 u32MinIQp;
  CVI_U32 u32MinQpDelta;

  CVI_U32 u32MotionSensitivity;
  CVI_S32 s32AvbrFrmLostOpen;
  CVI_S32 s32AvbrFrmGap;
  CVI_S32 s32AvbrPureStillThr;
} VENC_PARAM_H264_AVBR_S;
```

【成员】

成员名称	描述
s32ChangePos	码率调节阈值
u32MinIprop	预留
u32MaxIprop	最大 IP 比
s32MaxReEncodeTimes	预留
CVI_BOOL bQpMapEn	预留
s32MinStillPercent	静态场景最小码率百分比
u32MaxStillQP	静态场景最大 QP
u32MinStillPSNR	预留
u32MaxQp	最大 QP
u32MinQp	最小 QP
u32MaxIQp	I 帧最大 QP
u32MinIQp	I 帧最小 QP
u32MinQpDelta	帧级最小 QP 和宏块级最小 QP 差值
u32MotionSensitivity	运动灵敏度
s32AvbrFrmLostOpen	丢帧使能
s32AvbrFrmGap	最大丢帧次数
s32AvbrPureStillThr	静止门限值

【注意事项】

无。

【相关数据类型及接口】

- CVI_VENC_SetRcParam
- CVI_VENC_GetRcParam

7.4.33 VENC_PARAM_H265_CBR_S

【说明】

定义 H265 CBR 高级参数。

【定义】

```
typedef struct _VENC_PARAM_H265_CBR_S {
  CVI_U32 u32MinIprop;
  CVI_U32 u32MaxIprop;
  CVI_U32 u32MaxQp;
  CVI_U32 u32MinQp;
  CVI_U32 u32MaxIQp;
  CVI_U32 u32MinIQp;
  CVI_S32 s32MaxReEncodeTimes;
  CVI_BOOL bQpMapEn;
  VENC_RC_QPMAP_MODE_E enQpMapMode;
} VENC_PARAM_H265_CBR_S;
```

【成员】

成员名称	描述
u32MinIprop	预留
u32MaxIprop	预留
u32MaxQp	最大 QP
u32MinQp	最小 QP
u32MaxIQp	I 帧最大 QP
u32MinIQp	I 帧最小 QP
s32MaxReEncodeTimes	预留
bQpMapEn	是否开启 QPMAP 功能
enQpMapMode	QpMap 的模式

【注意事项】

无。

【相关数据类型及接口】

- CVI_VENC_SetRcParam
- CVI_VENC_GetRcParam

7.4.34 VENC_PARAM_H265_VBR_S

【说明】

定义 H265 VBR 高级参数。

【定义】

```
typedef struct _VENC_PARAM_H265_VBR_S {
  CVI_S32 s32ChangePos;
  CVI_U32 u32MinIprop;
  CVI_U32 u32MaxIprop;
  CVI_S32 s32MaxReEncodeTimes;

  CVI_U32 u32MaxQp;
  CVI_U32 u32MinQp;
  CVI_U32 u32MaxIQp;
  CVI_U32 u32MinIQp;

  CVI_BOOL bQpMapEn;
  VENC_RC_QPMAP_MODE_E enQpMapMode;
} VENC_PARAM_H265_VBR_S;
```

【成员】

成员名称	描述
s32ChangePos	码率调节阈值
u32MinIprop	预留
u32MaxIprop	预留
s32MaxReEncodeTimes	预留
u32MaxQp	最大 QP
u32MinQp	最小 QP
u32MaxIQp	I 帧最大 QP
u32MinIQp	I 帧最小 QP
CVI_BOOL bQpMapEn	预留
enQpMapMode	预留

【注意事项】

无。

【相关数据类型及接口】

- CVI_VENC_SetRcParam
- CVI_VENC_GetRcParam

7.4.35 VENC_PARAM_H265_AVBR_S

【说明】

定义 H265 AVBR 高级参数。

【定义】

```
typedef struct _VENC_PARAM_H265_AVBR_S {
  CVI_S32 s32ChangePos;
  CVI_U32 u32MinIprop;
  CVI_U32 u32MaxIprop;
  CVI_S32 s32MaxReEncodeTimes;
```

(下页继续)

(续上页)

```
  CVI_S32 s32MinStillPercent;
  CVI_U32 u32MaxStillQP;
  CVI_U32 u32MinStillPSNR;

  CVI_U32 u32MaxQp;
  CVI_U32 u32MinQp;
  CVI_U32 u32MaxIQp;
  CVI_U32 u32MinIQp;
  CVI_U32 u32MinQpDelta;

  CVI_U32 u32MotionSensitivity;
  CVI_S32 s32AvbrFrmLostOpen;
  CVI_S32 s32AvbrFrmGap;
  CVI_S32 s32AvbrPureStillThr;
  CVI_BOOL bQpMapEn;
  VENC_RC_QPMAP_MODE_E enQpMapMode;
} VENC_PARAM_H265_AVBR_S;
```

【成员】

成员名称	描述
s32ChangePos	码率调节阈值
u32MinIprop	预留
u32MaxIprop	预留
s32MaxReEncodeTimes	预留
s32MinStillPercent	静态场景最小码率百分比
u32MaxStillQP	静态场景最大 QP
u32MinStillPSNR	预留
u32MaxQp	最大 QP
u32MinQp	最小 QP
u32MaxIQp	I 帧最大 QP
u32MinIQp	I 帧最小 QP
u32MinQpDelta	帧级最小 QP 和宏块级最小 QP 差值
u32MotionSensitivity	运动灵敏度
s32AvbrFrmLostOpen	丢帧使能
s32AvbrFrmGap	最大丢帧次数
s32AvbrPureStillThr	静止门限值
bQpMapEn	预留
enQpMapMode	预留

【注意事项】

无。

【相关数据类型及接口】

- CVI_VENC_SetRcParam
- CVI_VENC_GetRcParam

7.4.36 VENC_PARAM_MOD_S

【说明】

编码相关模块参数。

【定义】

```
typedef struct _VENC_MODPARAM_S {
 VENC_MODTYPE_E enVencModType;
 union {
 VENC_MOD_VENC_S stVencModParam;
 VENC_MOD_H264E_S stH264eModParam;
 VENC_MOD_H265E_S stH265eModParam;
 VENC_MOD_JPEGE_S stJpegeModParam;
 VENC_MOD_RC_S stRcModParam;
 };
} VENC_PARAM_MOD_S;
```

【成员】

成员名称	描述
enVencModType	模块参数的类型
stVencModParam /stH264eModParam /stH265eModParam /stJpegeModParam /stRcModParam	Venc / H264e / H265e / Jpege / Rc 模块参数结构。

【注意事项】

无。

【相关数据类型及接口】

- CVI_VENC_SetModParam
- CVI_VENC_GetModParam

7.4.37 VENC_MOD_H264E_S

【说明】

H264 编码相关模块参数。

【定义】

```
typedef struct _VENC_MOD_H264E_S {
 CVI_U32 u32OneStreamBuffer;
 CVI_U32 u32H264eMiniBufMode;
 CVI_U32 u32H264ePowerSaveEn;
 VB_SOURCE_E enH264eVBSource;
 CVI_BOOL bQpHstgrmEn;
 CVI_U32 u32UserDataMaxLen;
 CVI_BOOL bSingleEsBuf;
```

(下页继续)

(续上页)

```
  CVI_U32 u32SingleEsBufSize;
} VENC_MOD_H264E_S;
```

【成员】

成员名称	描述
u32OneStreamBuffer	预留
u32H264eMiniBufMode	预留
u32H264ePowerSaveEn	预留
enH264eVBSource	VB 模式
bQpHstgrmEn	预留
u32UserDataMaxLen	用户数据最大长度
bSingleEsBuf	多个通道使用一个 streamBuffer
u32SingleEsBufSize	streamBuffer 的大小

【注意事项】

无。

【相关数据类型及接口】

- CVI_VENC_SetModParam
- CVI_VENC_GetModParam

7.4.38 VENC_MOD_H265E_S

【说明】

H265 编码相关模块参数。

【定义】

```
typedef struct _VENC_MOD_H265E_S {
  CVI_U32 u32OneStreamBuffer;
  CVI_U32 u32H265eMiniBufMode;
  CVI_U32 u32H265ePowerSaveEn;
  VB_SOURCE_E enH265eVBSource;
  CVI_BOOL bQpHstgrmEn;
  CVI_U32 u32UserDataMaxLen;
  CVI_BOOL bSingleEsBuf;
  CVI_U32 u32SingleEsBufSize;
  H265E_REFRESH_TYPE_E enRefreshType;
} VENC_MOD_H265E_S;
```

【成员】

成员名称	描述
u32OneStreamBuffer	预留
u32H265eMiniBufMode	预留
u32H265ePowerSaveEn	预留
enH265eVBSource	VB 模式
bQpHstgrmEn	预留
u32UserDataMaxLen	用户数据最大长度
bSingleEsBuf	多个通道使用一个 stream Buffer
u32SingleEsBufSize	streamBuffer 的大小
enRefreshType	刷新类型

【注意事项】

无。

【相关数据类型及接口】

- CVI_VENC_SetModParam
- CVI_VENC_GetModParam

7.4.39 VENC_MOD_JPEGE_S

【说明】

JPEG 编码相关模块参数。

【定义】

```
typedef struct _VENC_MOD_JPEGE_S {
  CVI_U32 u32OneStreamBuffer;
  CVI_U32 u32JpegeMiniBufMode;
  CVI_U32 u32JpegClearStreamBuf;
  CVI_BOOL bSingleEsBuf;
  CVI_U32 u32SingleEsBufSize;
  JPEGE_FORMAT_E enJpegeFormat;
  JPEGE_MARKER_TYPE_E JpegMarkerOrder[JPEG_MARKER_ORDER_CNT];
} VENC_MOD_JPEGE_S;
```

【成员】

成员名称	描述
u32OneStreamBuffer	预留
u32JpegeMiniBufMode	预留
u32JpegClearStreamBuf	预留
bSingleEsBuf	多个通道使用一个 stream Buffer
u32SingleEsBufSize	streamBuffer 的大小
enJpegeFormat	JPEG Header 编码方式
JpegMarkerOrder	Header 编码 mark 顺序

【注意事项】

无。

【相关数据类型及接口】

- CVI_VENC_SetModParam
- CVI_VENC_GetModParam

7.4.40 VENC_CHN_POOL_S

【说明】

定义编码通道绑定的 VB 池结构体。

【定义】

```
typedef struct _VENC_CHN_POOL_S {
  VB_POOL hPicVbPool;
  VB_POOL hPicInfoVbPool;
} VENC_CHN_POOL_S;
```

【成员】

成员名称	描述
hPicVbPool	用于存储 Picture 的 VB 池 Poold
hPicInfoVbPool	用于存储 Picture 信息的 VB 池 PoolId。

【注意事项】

无。

【相关数据类型及接口】

- CVI_VENC_AttachVbPool

7.4.41 VENC_FRAMELOST_S

【说明】

定义编码丢帧结构体。

【定义】

```
typedef struct _VENC_FRAMELOST_S {
  CVI_BOOL bFrmLostOpen;
  CVI_U32 u32FrmLostBpsThr;
  VENC_FRAMELOST_MODE_E enFrmLostMode;
  CVI_U32 u32EncFrmGaps;
} VENC_FRAMELOST_S;
```

【成员】

成员名称	描述
bFrmLostOpen	丢帧策略使能
u32FrmLostBpsThr	丢帧 bitrate 门限值
enFrmLostMode	丢帧模式
u32EncFrmGaps	允许最大连续丢帧数量

【注意事项】

无。

【相关数据类型及接口】

- CVI_VENC_SetFrameLostStrategy
- CVI_VENC_GetFrameLostStrategy

7.4.42 VENC_H264_ENTROPY_S

【说明】

定义 H264 熵编码结构体

【定义】

```
typedef struct _VENC_H264_ENTROPY_S {
  CVI_U32 u32EntropyEncModeI;
  CVI_U32 u32EntropyEncModeP;
  CVI_U32 u32EntropyEncModeB;
  CVI_U32 cabac_init_idc;
} VENC_H264_ENTROPY_S;
```

【成员】

成员名称	描述
u32EntropyEncModeI	I 帧熵编码模式 0:cavlc, 1:cabac
u32EntropyEncModeP	P 帧熵编码模式 0:cavlc, 1:cabac
u32EntropyEncModeB	预留，暂未使用
cabac_init_idc	参看 H264 协议

【注意事项】

无。

【相关数据类型及接口】

- CVI_VENC_SetH264Entropy
- CVI_VENC_GetH264Entropy

7.4.43 VENC_CU_PREDICTION_S

【说明】

定义 CU 预测属性结构体

【定义】

```
typedef struct _VENC_CU_PREDICTION_S {
  OPERATION_MODE_E enPredMode;

  CVI_U32 u32IntraCost;
  CVI_U32 u32Intra32Cost;
  CVI_U32 u32Intra16Cost;
  CVI_U32 u32Intra8Cost;
  CVI_U32 u32Intra4Cost;

  CVI_U32 u32Inter64Cost;
  CVI_U32 u32Inter32Cost;
  CVI_U32 u32Inter16Cost;
  CVI_U32 u32Inter8Cost;
} VENC_CU_PREDICTION_S;
```

【成员】

成员名称	描述
enPredMode	预留，暂未使用
u32IntraCost	默认为 0，用于降低 Intra Block 出现的概率
u32Intra32Cost	预留，暂未使用
u32Intra16Cost	预留，暂未使用
u32Intra8Cost	预留，暂未使用
u32Intra4Cost	预留，暂未使用
u32Inter64Cost	预留，暂未使用
u32Inter32Cost	预留，暂未使用
u32Inter16Cost	预留，暂未使用
u32Inter8Cost	预留，暂未使用

【注意事项】

无。

【相关数据类型及接口】

- CVI_VENC_SetCuPrediction
- CVI_VENC_GetCuPrediction

7.4.44 VENC_H264_TRANS_S

【说明】

定义 H.264 协议编码通道变换量化的结构体

【定义】

```
typedef struct _VENC_H264_TRANS_S {
  CVI_U32 u32IntraTransMode;
  CVI_U32 u32InterTransMode;
  CVI_BOOL bScalingListValid;
  CVI_U8 InterScalingList8X8[64];
  CVI_U8 IntraScalingList8X8[64];
  CVI_S32 chroma_qp_index_offset;
} VENC_H264_TRANS_S;
```

【成员】

成员名称	描述
u32IntraTransMode	预留，暂未使用
u32InterTransMode	预留，暂未使用
bScalingListValid	预留，暂未使用
InterScalingList8X8[64]	预留，暂未使用
IntraScalingList8X8[64]	预留，暂未使用
chroma_qp_index_offset	具体含义请参见 H.264 协议。 系统默认值为 0。 取值范围：[-12, 12]

【注意事项】

无。

【相关数据类型及接口】

- CVI_VENC_GetH264Trans
- CVI_VENC_SetH264Trans

7.4.45 VENC_H265_TRANS_S

【说明】

定义 H.265 协议编码通道变换量化的结构体

【定义】

```
typedef struct _VENC_H265_TRANS_S {
  CVI_S32 cb_qp_offset;
```

（下页继续）

(续上页)

```
  CVI_S32 cr_qp_offset;

  CVI_BOOL bScalingListEnabled;

  CVI_BOOL bScalingListTu4Valid;
  CVI_U8 InterScalingList4X4[2][16];
  CVI_U8 IntraScalingList4X4[2][16];

  CVI_BOOL bScalingListTu8Valid;
  CVI_U8 InterScalingList8X8[2][64];
  CVI_U8 IntraScalingList8X8[2][64];

  CVI_BOOL bScalingListTu16Valid;
  CVI_U8 InterScalingList16X16[2][64];
  CVI_U8 IntraScalingList16X16[2][64];

  CVI_BOOL bScalingListTU32Valid;
  CVI_U8 InterScalingList32X32[64];
  CVI_U8 IntraScalingList32X32[64];
} VENC_H265_TRANS_S;
```

【成员】

成员名称	描述
cb_qp_offset	具体含义请参见 H.265 协议。 系统默认值为 0。 取值范围：[-12, 12]
cr_qp_offset	具体含义请参见 H.265 协议。 系统默认值为 0。 取值范围：[-12, 12]
bScalingListEnabled	预留，暂未使用
bScalingListTu4Valid	预留，暂未使用
InterScalingList4X4[2][16]	预留，暂未使用
IntraScalingList4X4[2][16]	预留，暂未使用
bScalingListTu8Valid	预留，暂未使用
InterScalingList8X8[2][64]	预留，暂未使用
IntraScalingList8X8[2][64]	预留，暂未使用
bScalingListTu16Valid	预留，暂未使用
InterScalingList16X16[2][64]	预留，暂未使用
IntraScalingList16X16[2][64]	预留，暂未使用
bScalingListTu32Valid	预留，暂未使用
InterScalingList32X32[64]	预留，暂未使用
IntraScalingList32X32[64]	预留，暂未使用

【注意事项】

无。

【相关数据类型及接口】

- CVI_VENC_GetH265Trans
- CVI_VENC_SetH265Trans

7.5 错误码

视频编码 API 错误码如下表所示：

错误代码	宏定义	描述
0xC0078002	CVI_ERR_VENC_INVALID_CHNID	无效的通道 ID
0xC0078003	CVI_ERR_VENC_ILLEGAL_PARAM	非法参数
0xC0078004	CVI_ERR_VENC_EXIST	试图申请或者创建已经存在的设备、通道或者资源
0xC0078005	CVI_ERR_VENC_UNEXIST	通道不存在
0xC0078006	CVI_ERR_VENC_NULL_PTR	有空指针
0xC0078007	CVI_ERR_VENC_NOT_CONFIG	使用前未配置
0xC0078008	CVI_ERR_VENC_NOT_SUPPORT	操作不支持
0xC0078009	CVI_ERR_VENC_NOT_PERM	操作不被允许
0xC007800A	CVI_ERR_VENC_INVALID_PIPEID	非法 PipeID
0xC007800B	CVI_ERR_VENC_INVALID_GRPID	非法 group id
0xC007800C	CVI_ERR_VENC_NOMEM	内存配置失败
0xC007800D	CVI_ERR_VENC_NOBUF	影像 Buffer 配置失败
0xC007800E	CVI_ERR_VENC_BUF_EMPTY	缓冲区无数据
0xC007800F	CVI_ERR_VENC_BUF_FULL	缓冲区满数据
0xC0078010	CVI_ERR_VENC_SYS_NOTREADY	系统未准备好
0xC0078011	CVI_ERR_VENC_BADADDR	非法地址
0xC0078012	CVI_ERR_VENC_BUSY	装置使用中
0xC0078014	CVI_ERR_VENC_INVALID_VB	非法 vb
0xC0078040	CVI_ERR_VENC_INIT	初始化中
0xC0078041	CVI_ERR_VENC_FRC_NO_ENC	Frc 主动不编码当前帧
0xC0078042	CVI_ERR_VENC_STAT_VFPS_CHANGE	帧率发生改变
0xC0078043	CVI_ERR_VENC_EMPTY_STREAM_FRAME	无编码出码流
0xC0078044	CVI_ERR_VENC_EMPTY_PACK	无码流可获取
0xC0078045	CVI_ERR_VENC_JPEG_MARKER_ORDER	不支持的格式
0xC0078047	CVI_ERR_VENC_RC_PARAM	RC 参数设置
0xC007804B	CVI_ERR_VENC_MUTEX_ERROR	信号错误
0xC007804C	CVI_ERR_VENC_INVALILD_RET	底层非法返回值

8 视频解码

8.1 功能概述

VDEC 模块提供視頻解碼服務，将标准影音编码器压缩后的影像数据進行解碼，输出图像原始数据。

目前有支持的输入源为：

- 用户在 User Mode 自行输入的影像数据

目前支持的视讯标准为：

CV181x VDEC 模块支持 PT_JPEG/PT_MJPEG/PT_H264,CV180X 支持 PT_JPEG/PT_MJPEG。

8.1.1 目的

VDEC 模块提供驱动视频解码硬件工作的对应接口，实现视频解码功能。

8.1.2 定义及缩写

缩写名称 Acronym or term	定义 Definition
VDEC	Video Decoder 视频解码器
Output Order	输出顺序
Decoding Order	译码顺序
Display Order	播放顺序
Frame	帧
Stream	码流

8.2 设计概述

8.2.1 码流发送方式

VDEC 提供的码流发送方式:

- 按帧发送（VIDEO_MODE_FRAME）: 每次发送完整一帧码流到解码器，每调用一次发送接口，解码器就认为该帧码流已经结束，开始解码图像，需保证每次调用发送接口发送的码流必须为一帧。

8.2.2 图像输出方式

按照 H.264 视讯标准，输入的 Stream 经译码之前，输出的影像顺序未必等于输入的顺序，因此会有在播放时会分为 Decoding order 和 Display order 两种。

- Decoding Order: 输出影像的顺序等于输入 Stream 的顺序
  - 可以快速地取得译码后的 Frame, 但使用者需要自行保证播放顺序, 例如: 一般 Stream 中如果有 B frame, 会需要做 display order 的转换, 使用者需要作相关的处理。
- Display Order: 输出影像的顺序等于最后播放的顺序
  - 使用者取得的 Frame 已经是 Display order, 可以直接照该顺序播放。

目前的 Output Order 为 Display Order。

8.2.3 时间戳（PTS）处理

PTS 指的是当前 Frame 在播放时的时间点, 当前 Frame 的 PTS 可以由 CVI_VENC_GetFrame 中获得，而该 Frame 的 PTS 会等于 CVI_VENC_SendStream 时，所附加的 PTS。

8.2.4 解码帧存分配方式

- Common Mode: 自動建立 ION 内存進行幀存,ION 大小根據解碼后 Width 和 Height 自動分配, 不需要使用者管理
- User Mode: 使用者需要使用 CVI_VB_CreatePool, 建立 VB Pool, 在创建通道后, 透过 CVI_VDEC_AttachVbPool 绑定该 VB Pool 至通道
- Private Mode: 在创建通道时, 会自动建立 Private VB Pool, 使用者不需自行建立 VB Pool, 可以透过 CVI_VDEC_CreateChn, 设定 Private VB Pool 的 u32FrameBufSize 和 u32FrameBufCnt

可使用CVI_VDEC_SetModParam 设定 enVdecVBSource 来选择解码帧存分配方式, 目前只支持 COMMON 和 USER Mode。

使用 User Mode 时，在通道还没有 detach VB Pool 前, 不能直接 destroy VB Pool，需要确定 Decoder 正确结束才可以 destroy。

8.3 API 参考

该功能模块为用户提供以下 API:

- CVI_VDEC_CreateChn：创建视频解码通道。
- CVI_VDEC_DestroyChn：销毁视频解码通道。
- CVI_VDEC_ResetChn：复位视频解码通道。
- CVI_VDEC_GetChnAttr：获取视频解码通道属性。
- CVI_VDEC_SetChnAttr：设置视频解码通道属性。
- CVI_VDEC_StartRecvStream：解码器开始接收用户发送的码流。
- CVI_VDEC_StopRecvStream：解码器停止接收用户发送的码流。
- CVI_VDEC_QueryStatus：查询解码通道状态。
- CVI_VDEC_SetChnParam：设置解码通道参数。
- CVI_VDEC_GetChnParam：获取解码通道参数。
- CVI_VDEC_SendStream：向视频解码通道发送码流数据。
- CVI_VDEC_GetFrame：获取视频解码通道的解码图像。
- CVI_VDEC_ReleaseFrame：释放视频解码通道的解码图像。
- CVI_VDEC_SetModParam：设置解码相关的模块参数。
- CVI_VDEC_GetModParam：获取解码相关的模块参数。
- CVI_VDEC_AttachVbPool：将解码通道绑定到某个视频缓存 VB 池中。
- CVI_VDEC_DetachVbPool：将解码通道从某个视频缓存 VB 池中解绑定。

8.3.1 CVI_VDEC_CreateChn

【描述】

创建视频解码通道。

【语法】

```
CVI_S32 CVI_VDEC_CreateChn(VDEC_CHN VdChn, const VDEC_CHN_ATTR_S *pstAttr);
```

【参数】

参数名称	描述	输入/输出
VdChn	视频解码通道号。	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，其值为错误码 。

【需求】

- 头文件：cvi_vdec.h，cvi_comm_video.h，cvi_comm_vdec.h
- 库文件：ibvdec.so/libvdec.a

【注意】

- CV181x VDEC 模块支持 T_JPEG/PT_MJPEG/PT_H264,CV180X 支持 PT_JPEG/PT_MJPEG。
- 系统内存不足时会返回 CVII_ERR_VDEC_NOMEM 的错误码。
- 使用 JPEG/MJPEG 时要在创建解码信道之前要先创建专属于 VDEC 的模块 VB 池，不同协议解码所需的图像 VB 块大小不同，可参考 sample_vdec_lib.c 内的 vdecInitVBPool 函式。

【举例】

```
CVI_S32 SAMPLE_COMM_VDEC_Start(vdecChnCtx *pvdchnCtx)
{
  //设定VDEC 相关参数
  VDEC_CHN_ATTR_S stChnAttr, *pstChnAttr = &stChnAttr;
  VDEC_CHN VdecChn = pvdchnCtx->VdecChn;
  SAMPLE_VDEC_ATTR *psvdattr = &pvdchnCtx->stSampleVdecAttr;
  VDEC_CHN_PARAM_S stChnParam;

  pstChnAttr->enType = psvdattr->enType;
  pstChnAttr->enMode = psvdattr->enMode;
  pstChnAttr->u32PicWidth = psvdattr->u32Width;
  pstChnAttr->u32PicHeight = psvdattr->u32Height;
  pstChnAttr->u32StreamBufSize = psvdattr->u32Width * psvdattr->u32Height;
  pstChnAttr->u32FrameBufCnt = psvdattr->u32FrameBufCnt;

  //JPEG, MJPEG 需要设定VB buffer
  if (psvdattr->enType == PT_JPEG || psvdattr->enType == PT_MJPEG) {
    pstChnAttr->enMode = VIDEO_MODE_FRAME;
    pstChnAttr->u32FrameBufSize = VDEC_GetPicBufferSize(
        pstChnAttr->enType, psvdattr->u32Width, psvdattr->u32Height,
        psvdattr->stSapmleVdecPicture.enPixelFormat, DATA_BITWIDTH_8, 0);
  }
  //创建VDEC通道
  CHECK_CHN_RET(CVI_VDEC_CreateChn(VdecChn, pstChnAttr), VdecChn,
                                "CVI_VDEC_CreateChn");
  //确认目前default 参数
  CHECK_CHN_RET(CVI_VDEC_GetChnParam(VdecChn, &stChnParam), VdecChn,
                              "CVI_VDEC_GetChnParam");

  if (psvdattr->enType == PT_H264 || psvdattr->enType == PT_H265) {
  } else {
    stChnParam.stVdecPictureParam.enPixelFormat =
      psvdattr->stSapmleVdecPicture.enPixelFormat;
```

(下页继续)

(续上页)

```
    stChnParam.stVdecPictureParam.u32Alpha =
      psvdattr->stSapmleVdecPicture.u32Alpha;
  }
  //设定display frame..等参数
  stChnParam.u32DisplayFrameNum = psvdattr->u32DisplayFrameNum;
  //设定VDEC 参数
  CHECK_CHN_RET(CVI_VDEC_SetChnParam(VdecChn, &stChnParam), VdecChn,
                          "CVI_MPI_VDEC_GetChnParam");
  //Enable VDEC frame 传输
  CHECK_CHN_RET(CVI_VDEC_StartRecvStream(VdecChn), VdecChn,
                        "CVI_MPI_VDEC_StartRecvStream");
  return CVI_SUCCESS;
}
```

【相关主题】

CVI_VDEC_DestroyChn

8.3.2 CVI_VDEC_DestroyChn

【描述】

销毁视频解码通道。

【语法】

```
CVI_S32 CVI_VDEC_DestroyChn(VDEC_CHN VdChn);
```

【参数】

参数名称	描述	输入/输出
VdChn	视频解码通道号。	输入

【返回值】

返回值	描述
0	成功
非 0	失败，其值为错误码 。

【需求】

- 头文件：cvi_vdec.h，cvi_comm_video.h，cvi_comm_vdec.h
- 库文件：ibvdec.so/libvdec.a

【注意】

- 销毁前必须保证通道已创建，否则会返回通道未创建的错误。
- 销毁前必须停止接收码流（或者尚未开始接收码流），否则返回错误。

【举例】

- 无。

【相关主题】

CVI_VDEC_CreateChn

8.3.3 CVI_VDEC_ResetChn

【描述】

复位通道。

【语法】

```
CVI_S32 CVI_VDEC_ResetChn(VDEC_CHN VdChn);
```

【参数】

参数名称	描述	输入/输出
VdChn	通道号。	输入

【返回值】

返回值	描述
0	成功
非 0	失败，其值为错误码 。

【需求】

- 头文件：cvi_vdec.h，cvi_comm_video.h，cvi_comm_vdec.h
- 库文件：ibvdec.so/libvdec.a

【注意】

- Reset 并不存在的通道，返回失败 CVI_FAILURE。
- 如果一个通道没有停止接收码流而 reset 通道，则返回失败。

【举例】

- 无。

8.3.4 CVI_VDEC_GetChnAttr

【描述】

获取视频解码通道属性。

【语法】

```
CVI_S32 CVI_VDEC_GetChnAttr(VDEC_CHN VdChn, VDEC_CHN_ATTR_S *pstAttr);
```

【参数】

参数名称	描述	输入/输出
VdChn	视频解码通道号。	输入
pstAttr	解码通道属性指针。	输出

【返回值】

返回值	描述
0	成功
非 0	失败，其值为错误码 。

【需求】

- 头文件：cvi_vdec.h，cvi_comm_video.h，cvi_comm_vdec.h
- 库文件：ibvdec.so/libvdec.a

【注意】

- VDEC 通道必须已经创建。
- 此函式通常在CVI_VDEC_SetChnAttr 前使用，或是在获取 decoder frame 之前呼叫，以确认通道正常。

【举例】

可参阅 sample_common_vdec.c 内的 SAMPLE_COMM_VDEC_GetPic 函式。

```
CVI_VOID *SAMPLE_COMM_VDEC_GetPic(CVI_VOID *pArgs)
{
  VDEC_THREAD_PARAM_S *pstVdecThreadParam = (VDEC_THREAD_PARAM_S *)pArgs;
  FILE *fp = CVI_NULL;
  CVI_S32 s32Ret, s32Cnt = 0;
  VDEC_CHN_ATTR_S stAttr;
  VIDEO_FRAME_INFO_S stVFrame;
  CVI_CHAR cSaveFile[256];
  prctl(PR_SET_NAME, "VdecGetPic", 0, 0, 0);
  s32Ret = CVI_VDEC_GetChnAttr(pstVdecThreadParam->s32ChnId, &stAttr);
  if (s32Ret != CVI_SUCCESS) {
    CVI_VDEC_ERR("chn %d get chn attr fail for %#x!\n",
        pstVdecThreadParam->s32ChnId,
        s32Ret);
    return (CVI_VOID *)(CVI_FAILURE);
  }

  if (stAttr.enType != PT_JPEG && stAttr.enType != PT_H264 && stAttr.enType != PT_H265) {
    CVI_VDEC_ERR("chn %d enType %d do not support save file!\n",
        pstVdecThreadParam->s32ChnId,
        stAttr.enType);
    return (CVI_VOID *)(CVI_FAILURE);
  }

  while (1) {
    if (pstVdecThreadParam->eThreadCtrl == THREAD_CTRL_STOP)
```

（下页继续）

(续上页)

```
    break;

  s32Ret = CVI_VDEC_GetFrame(
      pstVdecThreadParam->s32ChnId,
      &stVFrame,
      pstVdecThreadParam->s32MilliSec);
  CVI_VDEC_TRACE("leave CVI_VDEC_GetFrame %d\n", s32Ret);
  ...
  ...
  }
}
```

【相关主题】

CVI_VDEC_SetChnAttr

8.3.5 CVI_VDEC_SetChnAttr

【描述】

设置视频解码通道属性。

【语法】

```
CVI_S32 CVI_VDEC_SetChnAttr(VDEC_CHN VdChn, const VDEC_CHN_ATTR_S *pstAttr);
```

【参数】

参数名称	描述	输入/输出
VdChn	视频解码通道号。	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，其值为错误码 。

【需求】

- 头文件：cvi_vdec.h，cvi_comm_video.h，cvi_comm_vdec.h
- 库文件：ibvdec.so/libvdec.a

【注意】

- VDEC 通道必须已经创建。
- 切换通道属性之前必须先停止接收码流，否则返回错误。

【举例】

- 无。

【相关主题】

CVI_VDEC_GetChnAttr

8.3.6 CVI_VDEC_StartRecvStream

【描述】

解码器开始接收用户发送的码流。

【语法】

```
CVI_S32 CVI_VDEC_StartRecvStream(VDEC_CHN VdChn);
```

【参数】

参数名称	描述	输入/输出
VdChn	视频解码通道号。	输入

【返回值】

返回值	描述
0	成功
非 0	失败，其值为错误码 。

【需求】

- 头文件：cvi_vdec.h，cvi_comm_video.h，cvi_comm_vdec.h
- 库文件：ibvdec.so/libvdec.a

【注意】

- 启动接收码流前必须保证通道已创建，否则会返回错误。

【举例】

请参见CVI_VDEC_CreateChn 举例

【相关主题】

CVI_VDEC_CreateChn

8.3.7 CVI_VDEC_StopRecvStream

【描述】

解码器停止接收用户发送的码流。

【语法】

```
CVI_S32 CVI_VDEC_StopRecvStream(VDEC_CHN VdChn);
```

【参数】

参数名称	描述	输入/输出
VdChn	视频解码通道号。	输入

【返回值】

返回值	描述
0	成功
非 0	失败，其值为错误码 。

【需求】

- 头文件：cvi_vdec.h，cvi_comm_video.h，cvi_comm_vdec.h
- 库文件：ibvdec.so/libvdec.a

【注意】

- 调用此接口，必须在CVI_VDEC_DestroyChn 之前。

【举例】

请参见 sample_common_vdec.c 内 SAMPLE_COMM_VDEC_Stop:

```
CVI_S32 SAMPLE_COMM_VDEC_Stop(CVI_S32 s32ChnNum)
{
  CVI_S32 i;
  for (i = 0; i < s32ChnNum; i++) {
    CHECK_CHN_RET(CVI_VDEC_DestroyChn(i), i, "CVI_MPI_VDEC_DestroyChn");
    CHECK_CHN_RET(CVI_VDEC_StopRecvStream(i), i, "CVI_MPI_VDEC_StopRecvStream");
  }

  return CVI_SUCCESS;
}
```

【相关主题】

- 无。

8.3.8 CVI_VDEC_QueryStatus

【描述】

查询解码通道状态。

【语法】

```
CVI_S32 CVI_VDEC_QueryStatus(VDEC_CHN VdChn, VDEC_CHN_STATUS_S *pstStatus);
```

【参数】

参数名称	描述	输入/输出
VdChn	视频解码通道号。	输入
pstStatus	视频解码通道状态结构体指针	输出

【返回值】

返回值	描述
0	成功
非 0	失败，其值为错误码 。

【需求】

- 头文件：cvi_vdec.h，cvi_comm_video.h，cvi_comm_vdec.h
- 库文件：ibvdec.so/libvdec.a

【注意】

- 启动接收码流前必须保证通道已创建，否则会返回错误。

【举例】

- 可参阅 sample_common_vdec.c 内之 SAMPLE_COMM_VDEC_CmdCtrl 用法。
- CVI_VDEC_QueryStatus 让用户可以查询以下信息：
  - enType：解碼格式。
  - bStartRecvStream：是否已经开始传送 frame 给予译码器。
  - u32DecodeStreamFrames：已解碼 frame 数。
  - u32LeftPics：剩余影像张数。
  - stVdecDecErr：译码器错误状态（s32FormatErr：格式错误码，s32PicSizeErrSet：影像大小错误，s32StreamUnsprt：不支援码流格式…）。

【相关主题】

- 无。

8.3.9 CVI_VDEC_SetChnParam

【描述】

设置解码通道参数。

【语法】

```
CVI_S32 CVI_VDEC_SetChnParam(VDEC_CHN VdChn, const VDEC_CHN_PARAM_S
↪*pstParam);
```

【参数】

参数名称	描述	输入/输出
VdChn	视频解码通道号。	输入
pstParam	通道参数。	输入

【返回值】

返回值	描述
0	成功
非 0	失败，其值为错误码 。

【需求】

- 头文件：cvi_vdec.h，cvi_comm_video.h，cvi_comm_vdec.h
- 库文件：ibvdec.so/libvdec.a

【注意】

- 启动接收码流前必须保证通道已创建，否则会返回错误。

【举例】

参阅 sample_common_vdec.c 内 SAMPLE_COMM_VDEC_Start 函式：

```
CVI_S32 SAMPLE_COMM_VDEC_Start(vdecChnCtx *pvdchnCtx)
{
  VDEC_CHN_ATTR_S stChnAttr, *pstChnAttr = &stChnAttr;
  VDEC_CHN VdecChn = pvdchnCtx->VdecChn;
  SAMPLE_VDEC_ATTR *psvdattr = &pvdchnCtx->stSampleVdecAttr;
  VDEC_CHN_PARAM_S stChnParam;

  pstChnAttr->enType = psvdattr->enType;
  pstChnAttr->enMode = psvdattr->enMode;
  pstChnAttr->u32PicWidth = psvdattr->u32Width;
  pstChnAttr->u32PicHeight = psvdattr->u32Height;
  pstChnAttr->u32StreamBufSize = psvdattr->u32Width * psvdattr->u32Height;
  pstChnAttr->u32FrameBufCnt = psvdattr->u32FrameBufCnt;

  if (psvdattr->enType == PT_JPEG || psvdattr->enType == PT_MJPEG) {
    pstChnAttr->enMode = VIDEO_MODE_FRAME;
    pstChnAttr->u32FrameBufSize = VDEC_GetPicBufferSize(
        pstChnAttr->enType, psvdattr->u32Width, psvdattr->u32Height,
        psvdattr->stSapmleVdecPicture.enPixelFormat, DATA_BITWIDTH_8, 0);
  }

  CHECK_CHN_RET(CVI_VDEC_CreateChn(VdecChn, pstChnAttr), VdecChn, "CVI_VDEC_
↪CreateChn");

  CHECK_CHN_RET(CVI_VDEC_GetChnParam(VdecChn, &stChnParam), VdecChn, "CVI_
↪VDEC_GetChnParam");

  if (psvdattr->enType == PT_H264 || psvdattr->enType == PT_H265) {
  } else {
    stChnParam.stVdecPictureParam.enPixelFormat =
      psvdattr->stSapmleVdecPicture.enPixelFormat;
```

(下页继续)

(续上页)

```
    stChnParam.stVdecPictureParam.u32Alpha =
      psvdattr->stSapmleVdecPicture.u32Alpha;
  }
  stChnParam.u32DisplayFrameNum = psvdattr->u32DisplayFrameNum;

  CHECK_CHN_RET(CVI_VDEC_SetChnParam(VdecChn, &stChnParam), VdecChn, "CVI_MPI_
↪VDEC_GetChnParam");

  CHECK_CHN_RET(CVI_VDEC_StartRecvStream(VdecChn), VdecChn, "CVI_MPI_VDEC_
↪StartRecvStream");
  return CVI_SUCCESS;
}
```

【相关主题】

- 无。

8.3.10 CVI_VDEC_GetChnParam

【描述】

获取解码通道参数。

【语法】

```
CVI_S32 CVI_VDEC_GetChnParam(VDEC_CHN VdChn, VDEC_CHN_PARAM_S *pstParam);
```

【参数】

参数名称	描述	输入/输出
VdChn	视频解码通道号。	输入
pstParam	通道参数。	输出

【返回值】

返回值	描述
0	成功
非 0	失败，其值为错误码 。

【需求】

- 头文件：cvi_vdec.h，cvi_comm_video.h，cvi_comm_vdec.h
- 库文件：ibvdec.so/libvdec.a

【注意】

- 启动接收码流前必须保证通道已创建，否则会返回错误。

【举例】

可参阅 sample_common_vdec.c 内的 SAMPLE_COMM_VDEC_Start 函式。

【相关主题】

CVI_VDEC_SetChnParam

8.3.11 CVI_VDEC_SendStream

【描述】

发送码流数据至视频解码通道。

【语法】

```
CVI_S32 CVI_VDEC_SendStream(VDEC_CHN VdChn, const VDEC_STREAM_S *pstStream,
↪CVI_S32 s32MilliSec);
```

【参数】

参数名称	描述	输入/输出
VdChn	视频解码通道号。	输入
pstStream	解码码流数据指针。	输入
s32MilliSec	送码流方式标志。 取值范围： -1：阻塞。	输入

【返回值】

返回值	描述
0	成功
非 0	失败，其值为错误码 。

【需求】

- 头文件：cvi_vdec.h，cvi_comm_video.h，cvi_comm_vdec.h
- 库文件：ibvdec.so/libvdec.a

【注意】

- 发送数据前必须保证已经调用CVI_VDEC_CreateChn ，CVI_VDEC_StartRecvStream 。
- 此接口在 mjpeg，jpeg 译码状态下，送入长度 0(pstStream->u32Len) 会返回 CVI_SUCCESS;
- 译码失败会回传错误码：ERR_CVI_VDEC_SEND_STREAM。

【举例】

请参阅 sample_common_vdec.c 内 SAMPLE_COMM_VDEC_SendStream 函式。

【相关主题】

- 无。

8.3.12 CVI_VDEC_GetFrame

【描述】

获取视频解码通道的解码图像。

【语法】

```
CVI_S32 CVI_VDEC_GetFrame(VDEC_CHN VdChn, VIDEO_FRAME_INFO_S *pstFrameInfo,
↪CVI_S32 s32MilliSec);
```

【参数】

参数名称	描述	输入/输出
VdChn	视频解码通道号。	输入
s32MilliSec	获取图像方式标志。 取值范围： -1：阻塞。	输入
pstFrameInfo	获取的解码图像信息。	输出

【返回值】

返回值	描述
0	成功
非 0	失败，其值为错误码 。

【需求】

- 头文件：cvi_vdec.h，cvi_comm_video.h，cvi_comm_vdec.h
- 库文件：ibvdec.so/libvdec.a

【注意】

- 获取解码图像时必须保证通道已经被创建，否则返回错误。

【举例】

参阅 sample_common_vdec.c 内 SAMPLE_COMM_VDEC_GetPic 函式。

【相关主题】

无。

8.3.13 CVI_VDEC_ReleaseFrame

【描述】

释放视频解码通道的图像。

【语法】

```
CVI_S32 CVI_VDEC_ReleaseFrame(VDEC_CHN VdChn, const VIDEO_FRAME_INFO_S
↪*pstFrameInfo);
```

【参数】

参数名称	描述	输入/输出
VdChn	视频解码通道号。	输入
pstFrameInfo	解码后的图像信息指针	输入

【返回值】

返回值	描述
0	成功
非 0	失败，其值为错误码 。

【需求】

- 头文件：cvi_vdec.h，cvi_comm_video.h，cvi_comm_vdec.h
- 库文件：ibvdec.so/libvdec.a

【注意】

- 此接口需与CVI_VDEC_GetFrame 配对使用，获取的数据应当在使用完之后立即释放，如果不及时释放，会导致解码过程阻塞等待资源。
- 释放的数据必须是CVI_VDEC_GetFrame 从该通道获取的数据，不得对数据信息结构体进行任何修改。
- 释放视频解码信道的图像必须在信道销毁前。

【举例】

无。

【相关主题】

无。

8.3.14 CVI_VDEC_SetModParam

【描述】

设置解码相关的模块参数。

【语法】

```
CVI_S32 CVI_VDEC_SetModParam(const VDEC_PARAM_MOD_S *pstModParam);
```

【参数】

参数名称	描述	输入/输出
pstModParam	解码模块参数指针。	输入

【返回值】

返回值	描述
0	成功
非 0	失败，其值为错误码 。

【需求】

- 头文件：cvi_vdec.h，cvi_comm_video.h，cvi_comm_vdec.h
- 库文件：ibvdec.so/libvdec.a

【注意】

- 此接口可以在通道创建前后呼叫。
- 此接口主要用于设置对应解码 VB pool 获取方式，用户可透过结构体VDEC_PARAM_MOD_S 内的 VB_SOURCE_E 类型变量设定 VB mode。

【举例】

- 无。

8.3.15 CVI_VDEC_GetModParam

【描述】

获取解码相关的模块参数。

【语法】

```
CVI_S32 CVI_VDEC_GetModParam(VDEC_PARAM_MOD_S *pstModParam);
```

【参数】

参数名称	描述	输入/输出
pstModParam	解码模块参数指针。	输出

【返回值】

返回值	描述
0	成功
非 0	失败，其值为错误码 。

【需求】

- 头文件：cvi_vdec.h，cvi_comm_video.h，cvi_comm_vdec.h
- 库文件：ibvdec.so/libvdec.a

【注意】

- 通常在CVI_VDEC_SetModParam 呼叫前搭配使用。

【举例】

- 无。

8.3.16 CVI_VDEC_AttachVbPool

【描述】

将解码通道绑定到某个视频缓存 VB 池中。

【语法】

```
CVI_S32 CVI_VDEC_AttachVbPool(VDEC_CHN VdChn, const VDEC_CHN_POOL_S *pstPool);
```

【参数】

参数名称	描述	输入/输出
VdChn	通道号。	输入
pstPool	视频缓存 VB 池的 Id 号	输入

【返回值】

返回值	描述
0	成功
非 0	失败，其值为错误码 。

【需求】

- 头文件：cvi_vdec.h，cvi_comm_video.h，cvi_comm_vdec.h
- 库文件：ibvdec.so/libvdec.a

【注意】

- 必须保证通道已创建，否则会返回 CVI_FAILURE 的错误码。

- 用户必须调用接口 CVI_VB_CreatePool 创建一个视频缓存 VB 池，再通过调用接口CVI_VDEC_AttachVbPool 把当前编码通道绑定到固定 PoolId 的 VB 池中。可以把多个编码通道绑定到同一个 VB 池中，但是不能把同一个编码通道绑定到多个 VB 池中。
- pstPool 必须是已创建 VB 池的有效 PoolId，包含存储 Picture 的 VB 池和存储存储 Picture 信息的 VB 池。
- 用户调用此接口，必须确定透过CVI_VDEC_SetModParam 设定在 VB_SOURCE_USER 模式。

【举例】

- 无。

8.3.17 CVI_VDEC_DetachVbPool

【描述】

将解码通道从某个视频缓存 VB 池中解绑定。

【语法】

```
CVI_S32 CVI_VDEC_DetachVbPool(VDEC_CHN VdChn);
```

【参数】

参数名称	描述	输入/输出
VdChn	通道号。	输入

【返回值】

返回值	描述
0	成功
非 0	失败，其值为错误码 。

【需求】

- 头文件：cvi_vdec.h，cvi_comm_video.h，cvi_comm_vdec.h
- 库文件：ibvdec.so/libvdec.a

【注意】

- 必须保证通道已创建，否则会返回 CVI_FAILURE 的错误码。

【举例】

- 无。

8.4 数据类型

视频解码相关数据类型、数据结构定义如下：

- VDEC_CHN_ATTR_S：解码通道属性。
- VDEC_ATTR_VIDEO_S：视频解码视频通道属性。
- VIDEO_MODE_E：码流发送方式枚举。
- VDEC_CHN_STATUS_S：通道状态结构体。
- VDEC_DECODE_ERROR_S：解码错误信息结构体。
- VDEC_CHN_PARAM_S：解码通道高级参数结构体。
- VDEC_PARAM_VIDEO_S：视频解码高级参数结构体。
- VDEC_PARAM_PICTURE_S：图片解码高级参数结构体。
- VIDEO_DEC_MODE_E：解码模式枚举。
- VIDEO_OUTPUT_ORDER_E：解码输出顺序枚举。
- COMPRESS_MODE_E：解码图像压缩模式枚举。
- H264_PRTCL_PARAM_S: H.264 协议相关的内存分配参数。
- H265_PRTCL_PARAM_S: H.265 协议相关的内存分配参数。
- VDEC_PRTCL_PARAM_S：协议相关的内存分配参数结构体。
- VDEC_STREAM_S：解码码流结构体。
- VDEC_USERDATA_S：用户数据结构体。
- VIDEO_DISPLAY_MODE_E：显示模式枚举。
- VDEC_CHN_POOL_S：定义解码通道绑定的 VB 池结构体。

8.4.1 VDEC_CHN_ATTR_S

【说明】

定义解码通道属性结构体。

【定义】

```
typedef struct _VDEC_CHN_ATTR_S {
  PAYLOAD_TYPE_E enType;
  VIDEO_MODE_E enMode;
  CVI_U32 u32PicWidth;
  CVI_U32 u32PicHeight;
  CVI_U32 u32StreamBufSize;
  CVI_U32 u32FrameBufSize;
  CVI_U32 u32FrameBufCnt;
  union {
    VDEC_ATTR_VIDEO_S
```

（下页继续）

(续上页)

```
    stVdecVideoAttr;
  };
} VDEC_CHN_ATTR_S;
```

【成员】

成员名称	描述
enType	解码协议类型枚举值。Video Codec 常使用枚举：PT_JPEG/PT_H264/PT_MJPEG
enMode	码流发送方式。目前仅支持 VIDEO_MODE_FRAME
u32PicWidth	通道支持的解码图像最大宽（以像素为单位）
u32PicHeight	通道支持的解码图像最大高（以像素为单位）
u32StreamBufSize	码流缓存的大小。(u32StreamBufSize = u32Width * u32Height)
u32FrameBufSize	解码图像帧存 buffer 大小。(随 enType 而异)
u32FrameBufCnt	解码图像帧存个数。
stVdecVideoAttr	视频 (H.264) 解码通道属性。

【注意事项】

CV181x VDEC 模块支持 PT_JPEG/PT_MJPEG/PT_H264,CV180X 支持 PT_JPEG/PT_MJPEG。

【相关数据类型及接口】

无。

8.4.2 VDEC_ATTR_VIDEO_S

【说明】

定义视频解码视频通道属性。

【定义】

```
typedef struct _VDEC_ATTR_VIDEO_S {
  CVI_U32 u32RefFrameNum;
  CVI_BOOL bTemporalMvpEnable;
  CVI_U32 u32TmvBufSize;
} VDEC_ATTR_VIDEO_S;
```

【成员】

成员名称描述	描述
u32RefFrameNum	参考帧的数目。(目前不支持)
bTemporalMvpEnable	是否支持时域运动矢量预测。(目前不支持)
u32TmvBufSize	视频解码图像 Tmv Buffer 大小。(目前不支持)

【注意事项】

无。

【相关数据类型及接口】

无。

8.4.3 VIDEO_MODE_E

【说明】

定义码流发送方式。

【定义】

```
typedef enum _VIDEO_MODE_E {
  VIDEO_MODE_STREAM = 0,
  VIDEO_MODE_FRAME,
  VIDEO_MODE_COMPAT,
  VIDEO_MODE_BUTT
} VIDEO_MODE_E;
```

【成员】

成员名称描述	描述
VIDEO_MODE_STREAM	按流方式发送码流。JPEG/MJPEG 解码不支持此模式。(目前不支持)
VIDEO_MODE_FRAME	按帧方式发送码流。
VIDEO_MODE_COMPAT	兼容模式发送码流。(目前不支持)

【注意事项】

无。

【相关数据类型及接口】

无。

8.4.4 VDEC_CHN_STATUS_S

【说明】

定义通道状态结构体。

【定义】

```
typedef struct _VDEC_CHN_STATUS_S {
  PAYLOAD_TYPE_E enType;
  CVI_S32 u32LeftStreamBytes;
  CVI_S32 u32LeftStreamFrames;
  CVI_S32 u32LeftPics;
  CVI_BOOL bStartRecvStream;
  CVI_U32 u32RecvStreamFrames;
```

(下页继续)

(续上页)

```
  CVI_U32 u32DecodeStreamFrames;
  VDEC_DECODE_ERROR_S stVdecDecErr;
  CVI_U32 u32Width;
  CVI_U32 u32Height;
} VDEC_CHN_STATUS_S;
```

【成员】

成员名称描述	描述
enType	解码协议。常使用枚举: PT_JPEG/ PT_H264/ PT_MJPEG
u32LeftStreamBytes	码流 buffer 中待解码的 byte 数，包括正在解码的当前帧中未解码的 byte 数。
u32LeftStreamFrames	码流 buffer 中待解码的帧数，不包括正在解码的当前帧。-1 表示无效。
u32LeftPics	图像 buffer 中剩余的 pic 数目。
bStartRecvStream	解码器是否已经启动接收码流。
u32RecvStreamFrames	码流 buffer 中已接收码流帧数。-1 表示无效。
u32DecodeStreamFrames	码流 buffer 中已解码帧数。
stVdecDecErr	解码错误信息。
u32Width	图像宽度
u32Height	图像高度

【注意事项】

CV181x VDEC 模块支持 PT_JPEG/PT_MJPEG/PT_H264,CV180X 支持 PT_JPEG/PT_MJPEG。

【相关数据类型及接口】

无。

8.4.5 VDEC_DECODE_ERROR_S

【说明】

定义解码错误信息结构体。

【定义】

```
typedef struct _VDEC_DECODE_ERROR_S {
  CVI_S32 s32FormatErr;
  CVI_S32 s32PicSizeErrSet;
  CVI_S32 s32StreamUnsprt;
  CVI_S32 s32PackErr;
  CVI_S32 s32PrtclNumErrSet;
  CVI_S32 s32RefErrSet;
  CVI_S32 s32PicBufSizeErrSet;
  CVI_S32 s32StreamSizeOver;
  CVI_S32 s32VdecStreamNotRelease;
} VDEC_DECODE_ERROR_S;
```

【成员】

成员名称	描述
s32FormatErr	不支持的格式。
s32PicSizeErrSet	图像的宽（或高）比通道的宽（或高）大。
s32StreamUnsprt	不支持的规格（码流规格与处理器宣称支持的规格不一致）。
s32PackErr	码流有错。
s32PrtclNumErrSet	设置的协议参数个数不够。比如：Slice、Pps、Sps 个数。
s32RefErrSet	设置的参考帧个数不够。
s32PicBufSizeErrSet	图像 buffer 内存大小不够。
s32StreamSizeOver	一帧码流太大了，当整个 SCDbuffer 都装不下一帧码流时，强制清空 SCDbuffer。
s32VdecStreamNotRelease	VFMW 内部管理码流错误，出现长时间不释放码流的情况。

【注意事项】

无。

【相关数据类型及接口】

无。

8.4.6 VDEC_CHN_PARAM_S

【说明】

定义解码通道高级参数。

【定义】

```
typedef struct _VDEC_CHN_PARAM_S {
  PAYLOAD_TYPE_E enType;
  CVI_U32 u32DisplayFrameNum;
  union {
    VDEC_PARAM_VIDEO_S
    stVdecVideoParam;
    VDEC_PARAM_PICTURE_S
    stVdecPictureParam;
  };
} VDEC_CHN_PARAM_S;
```

【成员】

成员名称	描述
enType	常使用枚举：PT_JPEG/ PT_H264/PT_MJPEG
u32DisplayFrameNum	解码缓存图像的最小帧数。
stVdecVideoParam	视频 (H.264/H.265) 解码高级参数。
stVdecPictureParam	图片 (JPEG/MJPEG) 解码高级参数

【注意事项】

CV181x VDEC 模块支持 PT_JPEG/PT_MJPEG/PT_H264,CV180X 支持 PT_JPEG/PT_MJPEG。

【相关数据类型及接口】

- CVI_VDEC_GetChnParam
- CVI_VDEC_SetChnParam

8.4.7 VDEC_PARAM_VIDEO_S

【说明】

定义视频解码高级参数。

【定义】

```
typedef struct _VDEC_PARAM_VIDEO_S {
  CVI_S32 s32ErrThreshold;
  VIDEO_DEC_MODE_E enDecMode;
  VIDEO_OUTPUT_ORDER_E enOutputOrder;
  COMPRESS_MODE_E enCompressMode;
  VIDEO_FORMAT_E enVideoFormat;
} VDEC_PARAM_VIDEO_S;
```

【成员】

成员名称	描述
s32ErrThreshold	错误阈值。取值范围：[0, 100]。 0 代表有错即丢 100 代表无。
enDecMode	解码模式
enOutputOrder	解码图像输出顺序。
enCompressMode	解码图像压缩模式。 COMPRESS_MODE_NONE = 0, COMPRESS_MODE_TILE, COMPRESS_MODE_LINE, COMPRESS_MODE_FRAME,
enVideoFormat	解码图像数据格式。

【注意事项】

无。

【相关数据类型及接口】

此参数设定与 VI 模块相关。

8.4.8 VDEC_PARAM_PICTURE_S

【说明】

定义图形解码高级参数。

【定义】

```
typedef struct _VDEC_PARAM_PICTURE_S {
  PIXEL_FORMAT_E enPixelFormat;
  CVI_U32 u32Alpha;
} VDEC_PARAM_PICTURE_S;
```

【成员】

成员名称	描述
enPixelFormat	JPEG(MJPEG) 解码输出格式。请参阅: cvi_comm_video.h 内 typedef enum _PIXEL_FORMAT_E
u32Alpha	ARGB 格式输出时的全局 alpha，仅 ARGB 输出时有效。(目前不支持)

【注意事项】

无。

【相关数据类型及接口】

无。

8.4.9 VIDEO_DEC_MODE_E

【说明】

定义视频解码模式枚举。

【定义】

```
typedef enum _VIDEO_DEC_MODE_E {
  VIDEO_DEC_MODE_IPB = 0,
  VIDEO_DEC_MODE_IP,
  VIDEO_DEC_MODE_I,
  VIDEO_DEC_MODE_BUTT
} VIDEO_DEC_MODE_E;
```

【成员】

成员名称	描述
VIDEO_DEC_MODE_IPB	IPB 模式，即 I、P、B 帧都解码。
VIDEO_DEC_MODE_IP	IP 模式，即只解码 I 帧和 P 帧。
VIDEO_DEC_MODE_I	I 模式，即只解码 I 帧。

【注意事项】

无。

【相关数据类型及接口】

无。

8.4.10 VIDEO_OUTPUT_ORDER_E

【说明】

定义视频解码输出顺序枚举。

【定义】

```
typedef enum _VIDEO_OUTPUT_ORDER_E {
  VIDEO_OUTPUT_ORDER_DISP = 0,
  VIDEO_OUTPUT_ORDER_DEC,
  VIDEO_OUTPUT_ORDER_BUTT
} VIDEO_OUTPUT_ORDER_E;
```

【成员】

成员名称	描述
VIDEO_OUTPUT_ORDER_DISP	显示序输出。
VIDEO_OUTPUT_ORDER_DEC	解码序输出。

【注意事项】

解码有 B 帧的码流应设置为显示序输出。

【相关数据类型及接口】

无。

8.4.11 COMPRESS_MODE_E

【说明】

定义解码图像压缩模式枚举。

【定义】

```
typedef enum _COMPRESS_MODE_E {
  COMPRESS_MODE_NONE = 0,
  COMPRESS_MODE_TILE,
  COMPRESS_MODE_LINE,
  COMPRESS_MODE_FRAME,
  COMPRESS_MODE_BUTT
} COMPRESS_MODE_E;
```

【成员】

成员名称	描述
COMPRESS_MODE_NONE	不压缩。
COMPRESS_MODE_TILE	预留
COMPRESS_MODE_LINE	预留
COMPRESS_MODE_FRAME	预留

【注意事项】

无。

【相关数据类型及接口】

无。

8.4.12 H264_PRTCL_PARAM_S

【说明】

与 H.264 协议相关的内存分配参数。

【定义】

```
typedef struct _H264_PRTCL_PARAM_S {
  CVI_S32 s32MaxSliceNum;
  CVI_S32 s32MaxSpsNum;
  CVI_S32 s32MaxPpsNum;
} H264_PRTCL_PARAM_S;
```

【成员】

成员名称	描述
s32MaxSliceNum	该通道解码支持的最大 Slice 个数。
s32MaxSpsNum	该通道解码支持的最大 SPS 个数。
s32MaxSpsNum	该通道解码支持的最大 PPS 个数。

【注意事项】

无。

【相关数据类型及接口】

VDEC_PRTCL_PARAM_S

8.4.13 H265_PRTCL_PARAM_S

【说明】

定义与 H.265 协议相关的内存分配参数。

【定义】

```
typedef struct _H265_PRTCL_PARAM_S {
  CVI_S32 s32MaxSliceSegmentNum;
  CVI_S32 s32MaxVpsNum;
  CVI_S32 s32MaxSpsNum;
  CVI_S32 s32MaxPpsNum;
} H265_PRTCL_PARAM_S;
```

【成员】

成员名称	描述
s32MaxSliceSegmentNum	该通道解码支持的最大 SliceSegment 个数。
s32MaxVpsNum	该通道解码支持的最大 VPS 个数。
s32MaxSpsNum	该通道解码支持的最大 SPS 个数。
s32MaxPpsNum	该通道解码支持的最大 PPS 个数。

【注意事项】

CV181x VDEC 模块支持 PT_JPEG/PT_MJPEG/PT_H264,CV180X 支持 PT_JPEG/PT_MJPEG。

【相关数据类型及接口】

VDEC_PRTCL_PARAM_S

8.4.14 VDEC_PRTCL_PARAM_S

【说明】

定义与协议相关的内存分配参数。

【定义】

```
typedef struct _VDEC_PRTCL_PARAM_S {
  PAYLOAD_TYPE_E
  enType;
  union {
    H264_PRTCL_PARAM_S
    stH264PrtclParam;
    H265_PRTCL_PARAM_S
    stH265PrtclParam;
  };
} VDEC_PRTCL_PARAM_S;
```

【成员】

成员名称	描述
enType	解码通道支持的协议。
stH264PrtclParam	H.264 协议参数。
stH265PrtclParam	H.265 协议参数。

【注意事项】

CV181x VDEC 模块支持 PT_JPEG/PT_MJPEG/PT_H264,CV180X 支持 PT_JPEG/PT_MJPEG。

【相关数据类型及接口】

无。

8.4.15 VDEC_STREAM_S

【说明】

定义视频解码的码流结构体。

【定义】

```
typedef struct _VDEC_STREAM_S {
  CVI_U32 u32Len;
  CVI_U64 u64PTS;
  CVI_BOOL bEndOfFrame;
  CVI_BOOL bEndOfStream;
  CVI_BOOL bDisplay;
  CVI_U8 *pu8Addr;
} VDEC_STREAM_S;
```

【成员】

成员名称	描述
u32Len	码流包的长度。
u64PTS	码流包的时间戳。
bEndOfFrame	当前帧是否结束。
bEndOfStream	是否发完所有码流。
pu8Addr	码流包的地址。
bDisplay	当前帧是否输出显示。

【注意事项】

无。

【相关数据类型及接口】

无。

8.4.16 VDEC_USERDATA_S

【说明】

定义用户数据结构体。

【定义】

```
typedef struct _VDEC_USERDATA_S {
  CVI_U64 u64PhyAddr;
  CVI_U32 u32Len;
  CVI_BOOL bValid;
  CVI_U8 *pu8Addr;
} VDEC_USERDATA_S;
```

【成员】

成员名称	描述
u32PhyAddr	用户数据的物理地址
u32Len	用户数据的长度。
bValid	当前数据的有效标识。
pu8Addr	用户数据的虚拟地址。

【注意事项】

无。

【相关数据类型及接口】

无。

8.4.17 VIDEO_DISPLAY_MODE_E

【说明】

定义显示模式枚举。

【定义】

```
typedef enum _VIDEO_DISPLAY_MODE_E {
  VIDEO_DISPLAY_MODE_PREVIEW = 0x0,
  VIDEO_DISPLAY_MODE_PLAYBACK = 0x1,
  VIDEO_DISPLAY_MODE_MAX
} VIDEO_DISPLAY_MODE_E;
```

【成员】

成员名称	描述
VIDEO_DISPLAY_MODE_PREVIEW	预览模式。
VIDEO_DISPLAY_MODE_PLAYBACK	回放模式。

【注意事项】

无。

【相关数据类型及接口】

无。

8.4.18 VDEC_PARAM_MOD_S

【说明】

解码相关模块参数。

【定义】

```
typedef struct _VDEC_MOD_PARAM_S {
  VB_SOURCE_E enVdecVBSource;
  CVI_U32 u32MiniBufMode;
  CVI_U32 u32ParallelMode;
  VDEC_VIDEO_MOD_PARAM_S stVideoModParam;
  VDEC_PICTURE_MOD_PARAM_S stPictureModParam;
} VDEC_MOD_PARAM_S;
```

【成员】

成员名称	描述
enVdecVBSource	解码帧存 VB 来源。 取值范围：仅支持 VB_SOURCE_COMMON, VB_SOURCE_USER
u32MiniBufMode	码流 buffer 配置模式
u32ParallelMode	VDH 解码模式
stVideoModParam	视频解码模块参数。对 JPEG/MJPEG 无效。
stPictureModParam	图片解码模块参数。对 H264/H265 无效。

【注意事项】

无。

【相关数据类型及接口】

- CVI_VDEC_SetModParam
- CVI_VDEC_GetModParam

8.4.19 VDEC_VIDEO_MOD_PARAM_S

【说明】

定义视频解码模块参数结构体

【定义】

```
typedef struct _VDEC_VIDEO_MOD_PARAM_S {
  CVI_U32 u32MaxPicWidth;
  CVI_U32 u32MaxPicHeight;
  CVI_U32 u32MaxSliceNum;
  CVI_U32 u32VdhMsgNum;
  CVI_U32 u32VdhBinSize;
  CVI_U32 u32VdhExtMemLevel;
} VDEC_VIDEO_MOD_PARAM_S;
```

【成员】

成员名称	描述
u32MaxPicWidth	视频解码支持的最大宽度。 取值范围：参见表 7-1，最小值为 H.264/H.265 解码支持分辨率的最小宽度。 最大值为 H.264/H.265 解码支持分辨率的最大宽度。默认为最大值。
u32MaxPicHeight	视频解码支持的最大高度。 取值范围：参见表 7-1，最小值为 H.264/H.265 解码支持分辨率的最小宽度。 最大值为 H.264/H.265 解码支持分辨率的最大宽度。默认为最大值。
u32MaxSliceNum	H.264/H.265 解码支持的最大 slice 个数。 取值范围：最小值为 1， 最大值为 H264/H265 解码支持 slice 的最大个数。 默认为最大值。
u32VdhMsgNum	VDH 解码消息池个数。
u32VdhBinSize	VDH 解码用来缓存 bin 数据 buffer 的大小。
u32VdhExtMemLevel	VDH 解码片外内存分配级别。

【注意事项】

CV181x VDEC 模块支持 PT_JPEG/PT_MJPEG/PT_H264,CV180X 支持 PT_JPEG/PT_MJPEG。

【相关数据类型及接口】

- CVI_VDEC_SetModParam
- CVI_VDEC_GetModParam

8.4.20 VDEC_PICTURE_MOD_PARAM_S

【说明】

定义图片解码模块参数结构体。

【定义】

```
typedef struct _VDEC_PICTURE_MOD_PARAM_S {
  CVI_U32 u32MaxPicWidth;
  CVI_U32 u32MaxPicHeight;
  CVI_BOOL bSupportProgressive;
  CVI_BOOL bDynamicAllocate;
  VDEC_CAPACITY_STRATEGY_E enCapStrategy;
} VDEC_PICTURE_MOD_PARAM_S;
```

【成员】

成员名称	描述
u32MaxPicWidth	图片解码支持的 最大宽度，默认值为当前处理器支持最大宽度。 取值范围：最小值为 JPEG/MJPEG 解码支持分辨率的最小宽度。 最大值为 JPEG/MJPEG 解码支持分辨率的最大宽度。默认为最大值。
u32MaxPicHeight	图片解码支持的最大高度，默认值为当前处理器支持最大高度。 取值范围：最小值为 JPEG/MJPEG 解码支持分辨率的最小宽度。 最大值为 JPEG/MJPEG 解码支持分辨率的最大宽度。默认为最大值。
bSupportProgressive	JPEG/MJPEG 解码是否支持 progressive 格式。
bDynamicAllocate	JPEG/MJPEG 解码支持 progressive 格式时，所需 buf 分配方式，默认值为 0。
enCapStrategy	解码图像的最大宽高能力集策略。

【注意事项】

无。

【相关数据类型及接口】

- CVI_VDEC_SetModParam
- CVI_VDEC_GetModParam

8.4.21 VDEC_CHN_POOL_S

【说明】

定义解码通道绑定的 VB 池结构体。

【定义】

```
typedef struct _VDEC_CHN_POOL_S {
  VB_POOL hPicVbPool;
  VB_POOL hTmvVbPool;
} VDEC_CHN_POOL_S;
```

【成员】

成员名称	描述
hPicVbPool	用于存储 Picture 的 VB 池 Poold
hTmvVbPool	用于存储 Tmv 的 VB 池 PoolId。

【注意事项】

无。

【相关数据类型及接口】

- CVI_VDEC_AttachVbPool

8.5 错误码

解碼错误码如下表所示：

错误代码	宏定义	描述
0xC0058002	CVI_ERR_VDEC_INVALID_CHNID	通道 ID 超出合法范围
0xC0058003	CVI_ERR_VDEC_ILLEGAL_PARAM	参数超出合法范围
0xC0058004	CVI_ERR_VDEC_EXIST	通道已经存在的
0xC0058005	CVI_ERR_VDEC_UNEXIST	通道未创建
0xC0058006	CVI_ERR_VDEC_NULL_PTR	空指针
0xC0058007	CVI_ERR_VDEC_NOT_CONFIG	使用前未配置
0xC0058008	CVI_ERR_VDEC_NOT_SUPPORT	不支持的参数或功能
0xC0058009	CVI_ERR_VDEC_NOT_PERM	操作不允许
0xC005800C	CVI_ERR_VDEC_NOMEM	分配内存失败
0xC005800D	CVI_ERR_VDEC_NOBUF	分配缓存失败
0xC005800E	CVI_ERR_VDEC_BUF_EMPTY	缓冲区中无数据
0xC005800F	CVI_ERR_VDEC_BUF_FULL	缓冲区中数据满
0xC0058010	CVI_ERR_VDEC_SYS_NOTREADY	系统没有初始化相关模块没有加载
0xC0058011	CVI_ERR_VDEC_BADADDR	地址错误
0xC0058012	CVI_ERR_VDEC_BUSY	系统忙

9 区域管理

9.1 功能概述

9.1.1 目的

REGION 模块，用于提供用户在视频中迭加 OSD 的功能，用以显示讯息（如：时间、通道号、地点等），或是迭加特定图片，也可以填充色块。这些迭加在视频上的涂层在此统称为区域。

9.1.2 定义及缩写

RGN（REGION 区域）

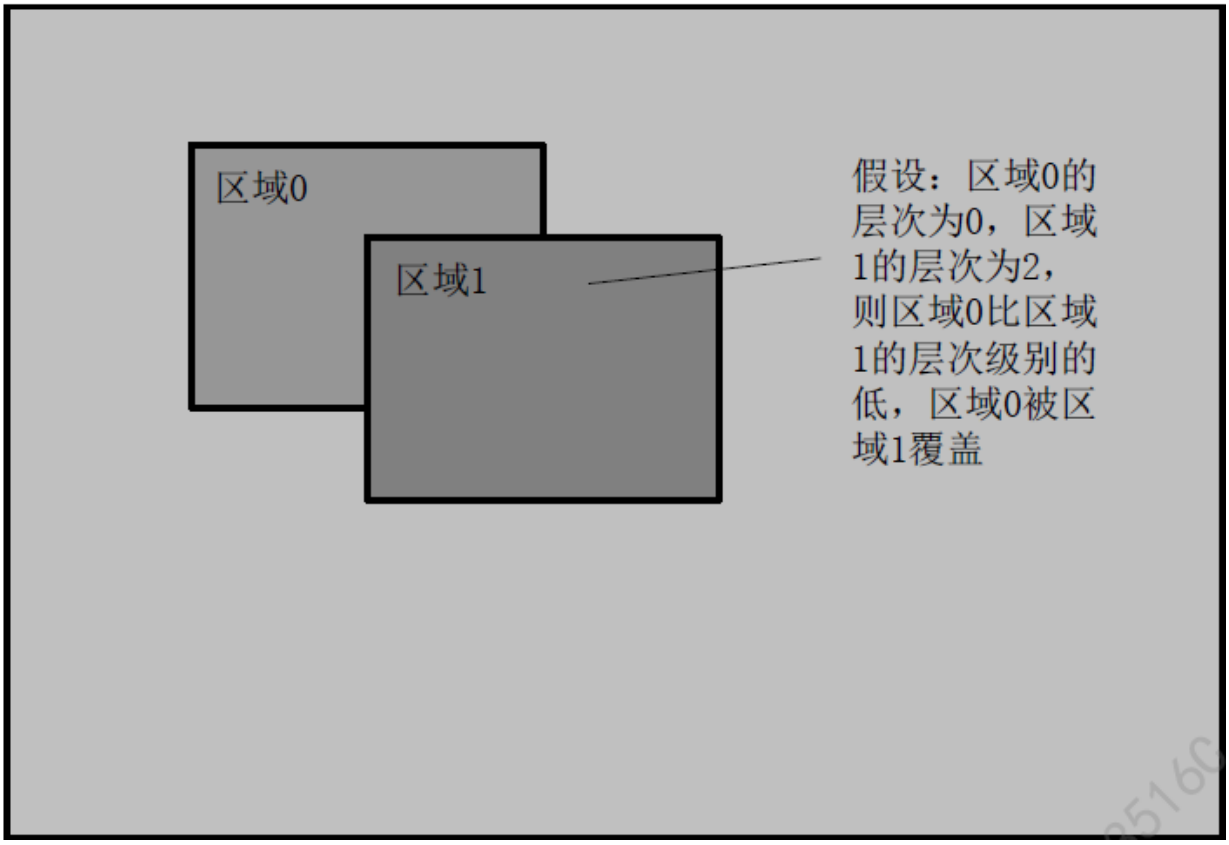
9.2 设计概述

9.2.1 系统架构

区域管理可以实现区域的创建，控制迭加到哪个视频，并可以控制暂时隐藏，或是改迭加到其他的视频中。

- 区域类型
  - Overlay: 视频迭加区域，支持 Bitmap 图片加载、背景色更新等功能
  - OverlayEx: 扩展视频迭加区域，支持 Bitmap 图片加载、背景色更新等功能，与 Overlay 不同处为支持区域层次
  - Cover: 视频遮挡区域，支持纯色块遮挡
  - CoverEx: 扩展视频遮挡区域，支持纯色块遮挡，与 Cover 不同处为支持区域层次
  - Mosaic: 暂不支持
- 区域层次

  区域层次表示区域的迭加级别，层次值越大，表示区域的显示级别越高。当发生重迭时，层次大的将会覆盖层次小的。仅 OverlayEx, CoverEx 类型的区域支持。



使用支持重迭的区域类型时 (OverlayEx, CoverEx)，VPSS mode 需设定为 VPSS_MODE_RGNEX，使用 CVI_SYS_SetVPSSMode 来设定。

- Bitmap 图加载

  这是指将 Bitmap 的数据填到区域的内存空间中，若是 Bitmap 和区域的格式有差异，例如 Bitmap 为 ARGB8888，区域为 ARGB1555，需要做些数据的转换。Bitmap 会从内存的左上角开始填充，Bitmap 不可大于区域内存。

  Bitmap 图加载支持两种方式：

1. 透过CVI_RGN_SetBitMap 将 Bitmap 复制到区域画布内存。
2. 透过CVI_RGN_GetCanvasInfo 取得区域画布内存的地址，直接更新在画布上。更新完成后，在由CVI_RGN_UpdateCanvas 让此画布更新为显示的画布。

- 区域属性

  创建一个区域时，需要设定该区域的属性。以 OverLayEx 为例，包含像素格式，大小和背景色等。

- 通道属性

  通道属性定义区域在某个绑定通道上的显示特性。例如，OverlayEx 的通道属性包含了显示位置。

9.2.2 注意事项

region 支援的模块信息 (CV180X 不支援 VO 模块)

类型	支援模块	设备号范围	通道号范围
OVERLAY	VPSS	[0, VPSS_MAX_GRP_NUM -1]	[0, VPSS_MAX_PHY_CHN_NUM -1]
	VO	[0, VO_MAX_LAYER_NUM -1]	[0, VO_MAX_CHN_NUM - 1]
OVER-LAYEX	VPSS	[0, VPSS_MAX_GRP_NUM -1]	[0, VPSS_MAX_PHY_CHN_NUM -1]
COVER	VPSS	[0, VPSS_MAX_GRP_NUM -1]	[0, VPSS_MAX_PHY_CHN_NUM -1]
	VO	[0, VO_MAX_LAYER_NUM -1]	[0, VO_MAX_CHN_NUM - 1]
COVEREX	VPSS	[0, VPSS_MAX_GRP_NUM -1]	[0, VPSS_MAX_PHY_CHN_NUM -1]

功能	OVERLAY		OVER-LAYEX	COVER		COVEREX
模块	VPSS	VO	VPSS	VPSS	VO	VPSS
像素格式	ARGB1555 ARGB4444 ARGB8888	ARGB1555 ARGB4444 ARGB8888	ARGB1555 ARGB4444 ARGB8888	ARGB1555 ARGB4444 ARGB8888	ARGB1555 ARGB4444 ARGB8888	ARGB1555 ARGB4444 ARGB8888
迭加层次	N/A	N/A	支持	N/A	N/A	支持
位图填充	支持	支持	支持	N/A	N/A	N/A

9.3 API 参考

该功能模块为用户提供以下 API:

- CVI_RGN_Create：创建一个区域。
- CVI_RGN_Destroy：销毁一个区域。
- CVI_RGN_GetAttr：获取区域属性。

- CVI_RGN_SetAttr：设置区域属性。
- CVI_RGN_SetBitMap：设置区域位图。
- CVI_RGN_AttachToChn：将区域叠加到通道上。
- CVI_RGN_DetachFromChn：将区域从通道中撤出。
- CVI_RGN_SetDisplayAttr：设置区域的通道显示属性。
- CVI_RGN_GetDisplayAttr：获取区域的通道显示属性。
- CVI_RGN_GetCanvasInfo：获取区域画布信息。
- CVI_RGN_UpdateCanvas：更新区域画布信息。
- CVI_RGN_SetChnPalette：设置通道色盘信息。

9.3.1 CVI_RGN_Create

【描述】

创建一个区域

【语法】

```
CVI_S32 CVI_RGN_Create(RGN_HANDLE Handle, const RGN_ATTR_S *pstRegion);
```

【参数】

参数名称	描述	输入/输出
Handle	区域号。 必须是未使用过的 Handle 号 取值范围:[0, 0XFFFFFFFF]。	输入
pstRegion	区域属性指针。	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，请参见错误码 。

【需求】

- 头文件：cvi_comm_region.h、cvi_region.h
- 库文件：libvpu.a

【注意】

- Handle 由用户指定，必须为单一不可重复。
- 不支持重复创建

· 创建 Overlay/OverLayEx 以外的区域时，只需指定区域类型即可。其他信息，是在调用CVI_RGN_AttachToChn 时指定。

【举例】

```
 CVI_S32 s32Ret;
RGN_HANDLE Handle = 0;
RGN_ATTR_S stRegion;
RGN_ATTR_S stRgnAttr;

stRegion.enType = OVERLAYEX_RGN;
stRegion.unAttr.stOverlayEx.enPixelFormat = PIXEL_FORMAT_ARGB_1555;
stRegion.unAttr.stOverlayEx.stSize.u32Height = 200;
stRegion.unAttr.stOverlayEx.stSize.u32Width = 300;
stRegion.unAttr.stOverlayEx.u32BgColor = 0x00000000; // ARGB1555 transparent
stRegion.unAttr.stOverlayEx.u32CanvasNum = 2;
s32Ret = CVI_RGN_Create(Handle, &stRegion);
if (s32Ret != CVI_SUCCESS) {
  return CVI_FAILURE;
}
s32Ret = CVI_RGN_GetAttr (Handle, &stRgnAttr);
if (s32Ret != CVI_SUCCESS) {
  return CVI_FAILURE;
}
stRgnAttr.unAttr.stOverlay.u32BgColor = 0x0000801f; // ARGB1555 blue
s32Ret = CVI_RGN_SetAttr (Handle, &stRgnAttr);
if (s32Ret != CVI_SUCCESS) {
  return CVI_FAILURE;
}
s32Ret = CVI_RGN_Destory (Handle);
if (s32Ret != CVI_SUCCESS) {
  return CVI_FAILURE;
}
```

【相关主题】

CVI_RGN_Destroy

9.3.2 CVI_RGN_Destroy

【描述】

销毁一个区域

【语法】

```
CVI_S32 CVI_RGN_Destroy(RGN_HANDLE Handle);
```

【参数】

参数名称	描述	输入/输出
Handle	区域号。 取值范围: [0, 0XFFFFFFFF]。	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，请参见错误码 。

【需求】

- 头文件：cvi_comm_region.h、cvi_region.h
- 库文件：libvpu.a

【注意】

- 区域必须已经创建

【举例】

参见CVI_RGN_Create 举例

【相关主题】

CVI_RGN_Create

9.3.3 CVI_RGN_GetAttr

【描述】

获取区域属性

【语法】

```
CVI_S32 CVI_RGN_GetAttr(RGN_HANDLE Handle, RGN_ATTR_S *pstRegion);
```

【参数】

参数名称	描述	输入/输出
Handle	区域号。 取值范围：[0, 0XFFFFFFFF]。	输入
pstRegion	区域属性指针	输出

【返回值】

返回值	描述
0	成功。
非 0	失败，请参见错误码 。

【需求】

- 头文件：cvi_comm_region.h、cvi_region.h
- 库文件：libvpu.a

【注意】

- 区域必须已经创建

【举例】

参见CVI_RGN_Create 举例

【相关主题】

CVI_RGN_SetAttr

9.3.4 CVI_RGN_SetAttr

【描述】

设置区域属性

【语法】

```
CVI_S32 CVI_RGN_SetAttr(RGN_HANDLE Handle, const RGN_ATTR_S *pstRegion);
```

【参数】

参数名称	描述	输入/输出
Handle	区域号。 取值范围: [0, 0XFFFFFFFF]。	输入
pstRegion	区域属性指针	输入

【返回值】

返回值	描述
0	成功。
非 0	失败, 请参见错误码 。

【需求】

- 头文件：cvi_comm_region.h、cvi_region.h
- 库文件：libvpu.a

【注意】

- 区域必须已经创建
- 仅 Overlay/OverlayEx 支援此接口

【举例】

参见CVI_RGN_Create 举例

【相关主题】

CVI_RGN_GetAttr

9.3.5 CVI_RGN_SetBitMap

【描述】

设置区域位图，将 Bitmap 填充到区域中。

【语法】

```
CVI_S32 CVI_RGN_SetBitMap(RGN_HANDLE Handle, const BITMAP_S *pstBitmap);
```

【参数】

参数名称	描述	输入/输出
Handle	区域号。 取值范围: [0, 0XFFFFFFFF]。	输入
pstBitmap	Bitmap 属性指针可参考”系统控制”章节的 BITMAP_S	输入

【返回值】

返回值	描述
0	成功。
非 0	失败, 请参见错误码 。

【需求】

- 头文件：cvi_comm_region.h、cvi_region.h
- 库文件：libvpu.a

【注意】

- 区域必须已经创建
- Bitmap 必须小于等于区域的大小
- Bitmap 和区域的像素格式必须一致
- 可多次调用
- 只支援 Overlay/OverlayEx

【举例】

无。

【相关主题】

无。

9.3.6 CVI_RGN_AttachToChn

【描述】

将区域叠加到通道上

【语法】

```
CVI_S32 CVI_RGN_AttachToChn(RGN_HANDLE Handle, const MMF_CHN_S *pstChn, const
↪RGN_CHN_ATTR_S *pstChnAttr);
```

【参数】

参数名称	描述	输入/输出
Handle	区域号。 取值范围: [0, 0XFFFFFFFF]。	输入
pstChn	通道信息指针	输入
pstChnAttr	区域通道属性指针	输入

【返回值】

返回值	描述
0	成功。
非 0	失败, 请参见错误码 。

【需求】

- 头文件：cvi_comm_region.h、cvi_region.h
- 库文件：libvpu.a

【注意】

- 区域必须已经创建
- 通道必须已经创建

【举例】

无。

【相关主题】

CVI_RGN_DetachFromChn

9.3.7 CVI_RGN_DetachFromChn

【描述】

将区域从通道中撤出

【语法】

```
CVI_S32 CVI_RGN_DetachFromChn(RGN_HANDLE Handle, const MMF_CHN_S *pstChn);
```

【参数】

参数名称	描述	输入/输出
Handle	区域号。 取值范围: [0, 0XFFFFFFFF]。	输入
pstChn	通道信息指针	输入

【返回值】

返回值	描述
0	成功。
非 0	失败, 请参见错误码 。

【需求】

- 头文件：cvi_comm_region.h、cvi_region.h
- 库文件：libvpu.a

【注意】

- 区域必须已经创建

【举例】

无。

【相关主题】

CVI_RGN_AttachToChn

9.3.8 CVI_RGN_SetDisplayAttr

【描述】

设置区域的通道显示属性

【语法】

```
CVI_S32 CVI_RGN_SetDisplayAttr(RGN_HANDLE Handle, const MMF_CHN_S *pstChn, const␣
↪RGN_CHN_ATTR_S *pstChnAttr);
```

【参数】

参数名称	描述	输入/输出
Handle	区域号。 取值范围: [0, 0XFFFFFFFF]。	输入
pstChn	通道信息指针	输入
pstChnAttr	区域通道属性指针	输入

【返回值】

返回值	描述
0	成功。
非 0	失败, 请参见错误码 。

【需求】

- 头文件：cvi_comm_region.h、cvi_region.h
- 库文件：libvpu.a

【注意】

- 区域必须已经创建
- 区域必须已经跟通道绑定

【举例】

无。

【相关主题】

CVI_RGN_GetDisplayAttr

9.3.9 CVI_RGN_GetDisplayAttr

【描述】

获取区域的通道显示属性

【语法】

```
CVI_S32 CVI_RGN_GetDisplayAttr(RGN_HANDLE Handle, const MMF_CHN_S *pstChn, RGN_
↪CHN_ATTR_S *pstChnAttr);
```

【参数】

参数名称	描述	输入/输出
Handle	区域号。 取值范围: [0, 0XFFFFFFFF]。	输入
pstChn	通道信息指针	输入
pstChnAttr	区域通道属性指针	输出

【返回值】

返回值	描述
0	成功。
非 0	失败，请参见错误码 。

【需求】

- 头文件：cvi_comm_region.h、cvi_region.h
- 库文件：libvpu.a

【注意】

- 区域必须已经创建
- 区域必须已经跟通道绑定

【举例】

无。

【相关主题】

CVI_RGN_SetDisplayAttr

9.3.10 CVI_RGN_GetCanvasInfo

【描述】

获取区域的画布信息

【语法】

```
CVI_S32 CVI_RGN_GetCanvasInfo(RGN_HANDLE Handle, RGN_CANVAS_INFO_S␣
↪*pstCanvasInfo);
```

【参数】

参数名称描述	描述	输入/输出
Handle	区域号。 取值范围：[0, 0XFFFFFFFF]。	输入
pstCanvasInfo	区域画布信息指针	输出

【返回值】

返回值	描述
0	成功。
非 0	失败，请参见错误码 。

【需求】

- 头文件：cvi_comm_region.h、cvi_region.h

- 库文件：libvpu.a

【注意】

- 区域必须已经创建
- 仅支援 Overlay/OverlayEx
- 此接口功能类似CVI_RGN_SetBitMap ，主要差异在于CVI_RGN_SetBitMap 在更新画布的过程中，是直接作用在所绑定的通道上，并且会多一次内存拷贝。而此接口不会有以上问题，但需要两块画布 (double buffer) 来实现。
- 此接口用于获取画布信息，之后用户可以直接对画布做操作，在完成更新之后，在调用进行CVI_RGN_UpdateCanvas 更新 (swap buffer)。
- 在调用了此接口之后，未调用CVI_RGN_UpdateCanvas 之前，不可调用CVI_RGN_SetBitMap 。

【举例】

无。

【相关主题】

CVI_RGN_UpdateCanvas

9.3.11 CVI_RGN_UpdateCanvas

【描述】

更新区域的画布

【语法】

```
CVI_S32 CVI_RGN_UpdateCanvas(RGN_HANDLE Handle);
```

【参数】

参数名称	描述	输入/输出
Handle	区域号。 取值范围：[0, 0XFFFFFFFF]。	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，请参见错误码 。

【需求】

- 头文件：cvi_comm_region.h、cvi_region.h
- 库文件：libvpu.a

【注意】

- 区域必须已经创建
- 此接口是用以配合CVI_RGN_GetCanvasInfo 使用。
  主要适用于画布内容更新之后，进行画布切换。
  如此可以避免画布更新过程中的瞬时。
- 必须先获取画布信息，在对画布更新完成之后，在调用此接口进行更新。
- 仅支援 Overlay/OverlayEx

【举例】

无。

【相关主题】

CVI_RGN_GetCanvasInfo

9.3.12 CVI_RGN_SetChnPalette

【描述】

设置通道色盘信息

【语法】

```
CVI_S32 CVI_RGN_SetChnPalette(RGN_HANDLE Handle, const MMF_CHN_S *pstChn, RGN_
↪PALETTE_S *pstPalette);
```

【参数】

参数名称	描述	输入/输出
Handle	区域号。 取值范围: [0, 0XFFFFFFFF]。	输入
pstChn	绑定通道号的信息	输入
pstPalette	色盘信息	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，请参见错误码 。

【需求】

- 头文件：cvi_comm_region.h、cvi_region.h
- 库文件：libvpu.a

【注意】

- 区域必须已经创建
- 仅支援 Overlay/OverlayEx

【举例】

无。

【相关主题】

无。

9.4 数据类型

9.4.1 RGN_TYPE_E

【说明】

定义区域类型。

【定义】

```
typedef enum _RGN_TYPE_E {
  OVERLAY_RGN = 0,
  COVER_RGN,
  COVEREX_RGN,
  OVERLAYEX_RGN,
  MOSAIC_RGN,
  RGN_BUTT
} RGN_TYPE_E;
```

【成员】

成员名称	描述
OVERLAY_RGN	视频迭加区域。
COVER_RGN	视频遮挡区域。
COVEREX_RGN	扩充视频遮挡区域。
OVERLAYEX_RGN	扩充视频迭加区域。
MOSAIC_RGN	Mosaic 视频区域

【注意事项】

无。

【相关数据类型及接口】

无。

9.4.2 RGN_AREA_TYPE_E

【说明】

定义 COVER、COVEREX 区域类型。

【定义】

```
typedef enum _RGN_AREA_TYPE_E {
  AREA_RECT = 0,
  AREA_QUAD_RANGLE,
  AREA_BUTT
} RGN_AREA_TYPE_E;
```

【成员】

成员名称	描述
AREA_RECT	矩形区域。
AREA_QUAD_RANGLE	任意四边形区域，尚未支持。

【注意事项】

无。

【相关数据类型及接口】

无。

9.4.3 OSD_COMPRESS_MODE_E

【说明】

定义 OSD 压缩模式类型。

【定义】

```
typedef enum _OSD_COMPRESS_MODE_E {
  OSD_COMPRESS_MODE_NONE = 0,
  OSD_COMPRESS_MODE_SW,
  OSD_COMPRESS_MODE_HW,
  OSD_COMPRESS_MODE_BUTT
} OSD_COMPRESS_MODE_E;
```

【成员】

成员名称	描述
OSD_COMPRESS_MODE_NONE	不使用压缩模式。
OSD_COMPRESS_MODE_SW	使用软件压缩模式。
OSD_COMPRESS_MODE_HW	使用硬件压缩模式。

【注意事项】

仅 CV181x/CV180x 支持 OSD 压缩。

【相关数据类型及接口】

无。

9.4.4 OSD_COMPRESS_INFO_S

【说明】

定义 OSD 压缩模式属性。

【定义】

```
typedef struct _OSD_COMPRESS_INFO_S {
  OSD_COMPRESS_MODE_E enOSDCompressMode;
  CVI_U32 u32EstCompressedSize;
  CVI_U32 u32CompressedSize;
} OSD_COMPRESS_INFO_S;
```

【成员】

成员名称	描述
enOSDCompressMode	定义 OSD 压缩模式类型。
u32EstCompressedSize	预估软件压缩模式下需要分配在内存大小。
u32CompressedSize	硬件压缩模式下需要分配在内存大小。

【注意事项】

仅 CV181x/CV180x 支持 OSD 压缩。

【相关数据类型及接口】

无。

9.4.5 COVER_CHN_ATTR_S

【说明】

定义 COVER 区域的通道属性。

【定义】

```
typedef struct _COVER_CHN_ATTR_S {
  RGN_AREA_TYPE_E enCoverType;
  union {
    RECT_S stRect;
    RGN_QUADRANGLE_S stQuadRangle;
  };
  CVI_U32 u32Color;
  CVI_U32 u32Layer;
  RGN_COORDINATE_E enCoordinate;
} COVER_CHN_ATTR_S;
```

【成员】

成员名称	描述
enCoverType	COVER 区域类型。
stRect	区域位置和宽高。 位置可为负数，矩形在通道的范围以外的部分会看不到。 宽高不可超过通道的大小。
stQuadRangle	任意四边形区域，尚未支持。
u32Color	COVER 区域颜色。 格式为 24bit RGB888。
u32Layer	区域层次。 目前不支持
enCoordinate	区域坐标类型。

【注意事项】

无。

【相关数据类型及接口】

无。

9.4.6 COVEREX_CHN_ATTR_S

【说明】

定义 COVEREX 区域的通道属性。

【定义】

```
typedef struct _COVEREX_CHN_ATTR_S {
  RGN_AREA_TYPE_E enCoverType;
  union {
    RECT_S stRect;
    RGN_QUADRANGLE_S stQuadRangle;
  };
  CVI_U32 u32Color;
  CVI_U32 u32Layer;
} COVEREX_CHN_ATTR_S;
```

【成员】

成员名称	描述
enCoverType	COVEREX 区域类型。
stRect	区域位置和宽高。 位置可为负数，矩形在通道的范围以外的部分会看不到。 宽高不可超过通道的大小。
stQuadRangle	任意四边形区域，尚未支持。
u32Color	COVEREX 区域颜色。 格式为 24bit RGB888。
u32Layer	区域层次。

【注意事项】

无。

【相关数据类型及接口】

无。

9.4.7 OVERLAY_ATTR_S

【说明】

定义视频迭加区域的属性。

【定义】

```
typedef struct _OVERLAY_ATTR_S {
  PIXEL_FORMAT_E enPixelFormat;
  CVI_U32 u32BgColor;
  SIZE_S stSize;
  CVI_U32 u32CanvasNum;
  OSD_COMPRESS_INFO_S stCompressInfo;
} OVERLAY_ATTR_S;
```

【成员】

成员名称	描述
enPixelFormat	像素格式。 PIXEL_FORMAT_ARGB_1555, PIXEL_FORMAT_ARGB_4444, PIXEL_FORMAT_ARGB_8888。
u32BgColor	区域背景色。 根据 enPixelFormat 定义。
stSize	区域宽高。
u32CanvasNum	区域内存数量。
stCompressInfo	OSD 压缩模式结构体。

【注意事项】

- stSize 会影响区域分配的内存大小，建议不要大于最终链接通道的宽高，以免浪费内存。
- u32CanvasNum 根据使用情况决定。
  若使用CVI_RGN_SetBitMap，为 1 即可。
  若使用CVI_RGN_GetCanvasInfo，希望 double buffer 避免更新过程中的瞬时，则建议为 2。

【相关数据类型及接口】

无。

9.4.8 OVERLAY_CHN_ATTR_S

【说明】

定义视频迭加区域的通道属性。

【定义】

```
typedef struct _OVERLAY_CHN_ATTR_S {
  POINT_S stPoint;
  CVI_U32 u32Layer;
  OVERLAY_INVERT_COLOR_S stInvertColor;
} OVERLAY_CHN_ATTR_S;
```

【成员】

成员名称	描述
stPoint	区域位置和宽高。 位置可为负数，矩形在通道的范围以外的部分会看不到。 宽高不可超过通道的大小。
u32Layer	区域层次。 目前不支持。
stInvertColor	颜色反转结构体。

【注意事项】

无。

【相关数据类型及接口】

无。

9.4.9 OVERLAYEX_ATTR_S

【说明】

定义扩展视频迭加区域的属性。

【定义】

```
typedef struct _OVERLAYEX_ATTR_S {
 PIXEL_FORMAT_E enPixelFormat;
 CVI_U32 u32BgColor;
 SIZE_S stSize;
 CVI_U32 u32CanvasNum;
 OSD_COMPRESS_INFO_S stCompressInfo;
} OVERLAYEX_ATTR_S;
```

【成员】

成员名称	描述
enPixelFormat	像素格式。 PIXEL_FORMAT_ARGB_1555, PIXEL_FORMAT_ARGB_4444, PIXEL_FORMAT_ARGB_8888。
u32BgColor	区域背景色。 根据 enPixelFormat 定义。
stSize	区域宽高。
u32CanvasNum	区域内存数量。
stCompressInfo	OSD 压缩模式结构体。

【注意事项】

- stSize 会影响区域分配的内存大小，建议不要大于最终链接通道的宽高，以免浪费内存。
- u32CanvasNum 根据使用情况决定。
  若使用CVI_RGN_SetBitMap，为 1 即可。
  若使用CVI_RGN_GetCanvasInfo，希望 double buffer 避免更新过程中的瞬时，则建议为 2。

【相关数据类型及接口】

无。

9.4.10 OVERLAYEX_CHN_ATTR_S

【说明】

定义扩展视频迭加区域的通道属性。

【定义】

```
typedef struct _OVERLAYEX_CHN_ATTR_S {
 POINT_S stPoint;
 CVI_U32 u32Layer;
 OVERLAY_INVERT_COLOR_S stInvertColor;
} OVERLAYEX_CHN_ATTR_S;
```

【成员】

成员名称	描述
stPoint	区域位置和宽高。 位置可为负数，矩形在通道的范围以外的部分会看不到。 宽高不可超过通道的大小。
u32Layer	区域层次。
stInvertColor	颜色反转结构体。

【注意事项】

无。

【相关数据类型及接口】

无。

9.4.11 RGN_ATTR_U

【说明】

定义区域联合体。

【定义】

```
typedef union _RGN_ATTR_U {
 OVERLAY_ATTR_S stOverlay;
 OVERLAYEX_ATTR_S stOverlayEx;
} RGN_ATTR_U;
```

【成员】

成员名称	描述
stOverlay	视频迭加区域属性。
stOverlayEx	扩展视频迭加区域属性。

【注意事项】

仅有 RGN_TYPE_E 为 Overlay/OverlayEx 时，需设置此属性。

【相关数据类型及接口】

无。

9.4.12 RGN_CHN_ATTR_U

【说明】

定义区域通道联合体。

【定义】

```
typedef union _RGN_CHN_ATTR_U {
  OVERLAY_CHN_ATTR_S stOverlayChn;
  COVER_CHN_ATTR_S stCoverChn;
  COVEREX_CHN_ATTR_S stCoverExChn;
  OVERLAYEX_CHN_ATTR_S stOverlayExChn;
  MOSAIC_CHN_ATTR_S stMosaicChn;
} RGN_CHN_ATTR_U;
```

【成员】

成员名称	描述
stOverlayChn	视频迭加区域通道属性。
stCoverChn	遮挡区域通道属性。
stCoverExChn	扩充遮挡区域通道属性。
stOverlayExChn	扩充视频迭加区域通道属性。
stMosaicChn	Mosaic 区域通道属性。

【注意事项】

无。

【相关数据类型及接口】

无。

9.4.13 RGN_ATTR_S

【说明】

定义区域属性。

【定义】

```
typedef struct _RGN_ATTR_S {
  RGN_TYPE_E enType;
  RGN_ATTR_U unAttr;
} RGN_ATTR_S;
```

【成员】

成员名称	描述
enType	区域类型。
unAttr	区域属性联合体。

【注意事项】

无。

【相关数据类型及接口】

无。

9.4.14 RGN_CHN_ATTR_S

【说明】

定义区域通道属性。

【定义】

```
typedef struct _RGN_CHN_ATTR_S {
  CVI_BOOL bShow;
  RGN_TYPE_E enType;
  RGN_CHN_ATTR_U unChnAttr;
} RGN_CHN_ATTR_S;
```

【成员】

成员名称	描述
bShow	区域是否显示。
enType	区域类型。
unChnAttr	区域通道联合体

【注意事项】

无。

【相关数据类型及接口】

无。

9.5 错误码

区域系统 API 错误码如下表所示：

错误代码	宏定义	描述
0xC0038001	CVI_ERR_RGN_INVALID_DEVID	设备号无效
0xC0038002	CVI_ERR_RGN_INVALID_CHNID	RGN 通道号无效
0xC0038003	CVI_ERR_RGN_ILLEGAL_PARAM	RGN 参数设置无效
0xC0038004	CVI_ERR_RGN_EXIST	RGN 已创建
0xC0038005	CVI_ERR_RGN_UNEXIST	RGN 未创建
0xC0038006	CVI_ERR_RGN_NULL_PTR	输入空指针错误
0xC0038007	CVI_ERR_RGN_NOT_CONFIG	模块没有设置
0xC0038008	CVI_ERR_RGN_NOT_SUPPORT	操作不支持
0xC0038009	CVI_ERR_RGN_NOT_PERM	操作不允许
0xC003800c	CVI_ERR_RGN_NOMEM	分配内存失败
0xC003800d	CVI_ERR_RGN_NOBUF	分配 BUF 池失败
0xC003800e	CVI_ERR_RGN_BUF_EMPTY	BUF 池为空
0xC003800f	CVI_ERR_RGN_BUF_FULL	BUF 池已满
0xC0038011	CVI_ERR_RGN_BADADDR	RGN ioclt 参数错误
0xC0038012	CVI_ERR_RGN_BUSY	RGN 系统忙

10 音频

10.1 功能概述

10.1.1 目的

10.1.2 定义及缩写

缩写名称 Acronym or term	定义 Definition
AI	Audio Input: 音频输入模块
AO	Audio Output: 音频输出模块
AENC	Audio Encode: 音频编码模块
ADEC	Audio Decode 音频解碼模块
VQE	Voice Quality Enhancement 语音音质增强模块
RES	Resample 重采样
AGC	自动增益控制
ANR	语音降躁
AEC	听觉回声消除
Kernel Layer	Linux 内核层
Multimedia Framework	媒体框架
Middleware	中间件
Code Driver	驱动

10.2 设计概述

10.2.1 系统架构

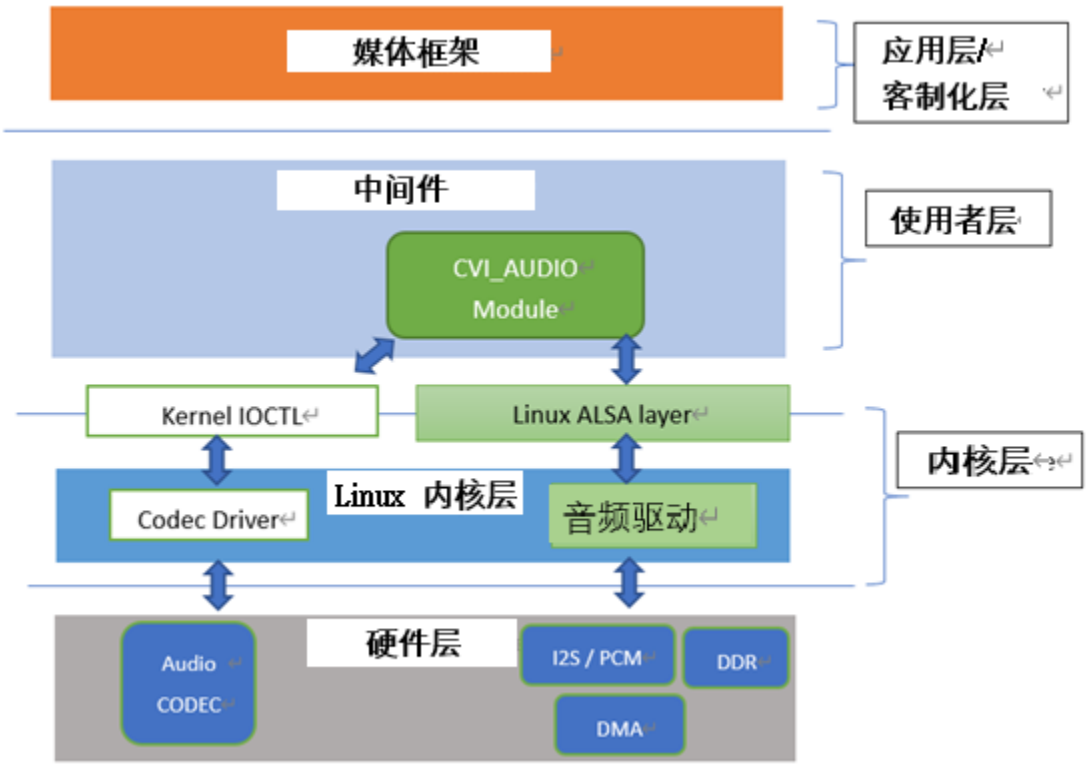


图 10.1: 图 9-1

如上图所示，对应缩写名称及中文请参考 9.1.2, 本文提及的音频模块 (音频输入模块、音频输出模块、音频编码模块、音频解碼模块、语音音质增强模块、重采样) 属于 Middleware 多媒体层接口之一，往上负责与应用层或客户业务层 API 对接，使用者可以透过（CVI_AI_，CVI_AO_，CVI_ADE C_，CVI_AENC_ 相关前缀开头等）函数往下控制对应音频组件，应用层或称客户业务层 (Application/Customize Layer) 如需了解音频 API 接口，可参照本文档 9.3: API 参考做详细了解，如需相关 API 调用逻辑，可参照 SDK 内 cvi_sample_audio.c 内范例了解基本音频模块间 (下图) 的使用顺序。

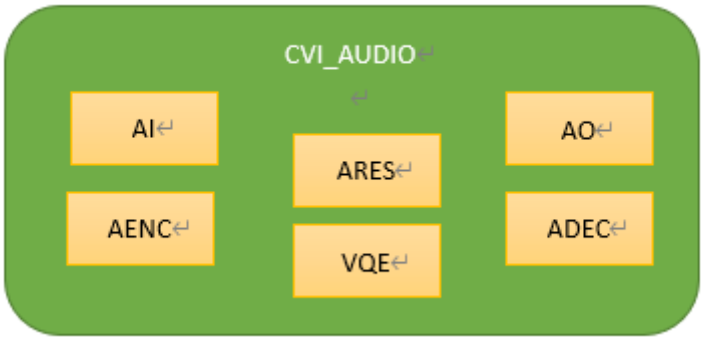


图 10.2: 图 9-2

CVI_AUDIO 往下对接 Linux 内核层 (Kernel Layer) 相关驱动程序，透过 Linux 标准声音体系 (ALSA: Advanced Linux Sound Architecture)，实现基本音频输入及输出功能，因此可知，使用者调用 cvi_audio_xxx/ cvi_axx_xx(ex. cvi_adec_xxx) API 时内部的基本及最小的音频单位为 Frame(本文内所称音框、音帧皆指 Audio Frame)，文档内音框的计量单位为取样点数 (1 Frame = numbers of samples)，而不采用位 (bytes) 计算。Frame 的大小范围，在不使用语音音质增进 (VQE) 模块下，可以设定 160/320/480 取样点为音框大小 (请勿超过 512)。在使用 VQE 的状态下，必须设定音框大小最小单位为 160 取样点的倍数，此方式是为了配合内部 VQE 模块设计。

对于用户而言，除了 Audio Codec 功能或是有相关调试需求，基本功能皆透过 libcvi_audio、libcvi_xxx 相关 so 档实现，而不会直接呼叫 IOCTL 函示控制内核层，用以确保系统内部资源分配的稳定性及可靠性。

图 10.3: 图 9-3

上图描述音频输入及编码的关系，AENC 使用 RISC-V 编码不透过内核层将码流储存于 DDR 内，并由收音端 (支持麦克风收音及语音线路收音) 到编码端依序使能。在使能 Audio Input、AENC 模块时，内部 API 参数设定需尽量一致 (采样率、音框大小，通道数)，否则编码后获取的音频异常。

图 10.4: 图 9-4

上图描述音频解碼及输出的关系，用户可以直接放入 pcm/raw 原始音档到存储设备内并呼叫 ADEC/Audio Output API 播音，并依序由解码端到输出端使能，ADEC 及 Audio Output 参数

设定请尽量一致，否则播放音频异常。

图 10.5: 图 9-5

上图描述了 VQE 前后的关系，音频收音后及编码前，用户可以透过 VQE 相关 API(参阅 9-3)开启 AEC/ANR/AGC 功能，此时音框单位需设定为 160 的倍数，工作频率支持 8Khz/16Khz。

图 10.6: 图 9-6

VQE 包含了前端 VQE(图 9-5)，及上图描述播音端 VQE(后端 VQE)，目前不支持后端 VQE。

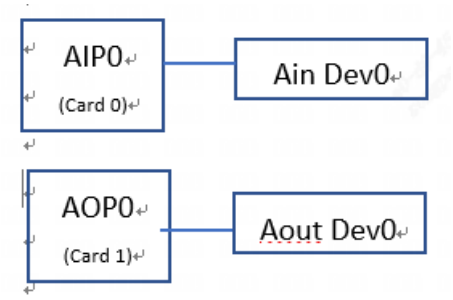
10.2.2 音频输入与输出

10.2.2.1 音频接口及 Audio Input、Audio Output 设备

音频接口分两类，输入 (Audio Input) 及输出 (Audio Output) 接口，各负责声音的录制和播放。与 Audio Codec 对接并负责抽象音频接口输入功能的单元，称之为 Audio Input 设备；Audio Output 设备则是负责抽象音频接口输出的功能，依据该接口支持的功能，分别与 Audio Input 设备和 Audio Output 设备建立映射。

音频输入输出接口简称为 AIO（Audio Input/Output）接口，用于和 Audio Codec 对接，完成声音的录制和播放。AIO 接口分为两种类型：只支持输入或只支持输出，当为输入类型时，又称为 AIP，当为输出类型时，又称为 AOP。AIP0 只支持音频信号输入，则 AIP0 映像为 AiDev0；AOP0 只支持音频信号输出，AOP0 映像为 AoDev0。

音频输入接口 (Audio Input) 支持 PCM 及 I2S 输入，输出依照 Linux 内核标准规范对接 ALSA PCM device，Cvitek 支持一组预设输入及输出，若以 Linux ALSA 架构来看，对应设备可视为 card 0、card 1，其关系如下图:



AIP0 仅可支持输入，AOP0 仅可支持音频输出，在输入及输出对接的状况下，例如: 实时录音及拨音或语音对讲，此时 Audio Input/Audio Output 设备的取样率及位宽度需相同，信道数目必须一致，采样率也必须一致。为因应客制化产品需求，Cvitek 有多两组 I2S，可随产品或客户应用需要将多出的两组 I2S 更动设定为 AiDev(输入) 对接口，或是改为 AoDev(输出) 对接口。

10.2.2.2 录音与播放原理

CVI_AUDIO API 所处理的音框皆为数字化后的信号，然而实际在收音端及播音端皆为模拟信号，数字及仿真信号是透过 Audio Codec 做转换，Audio Codec 透过 I2S 或 PCM 时序将输入的仿真信号源转换并传给 Audio Input 模块，同理，Audio Output 端播音时也透过 Audio Codec 将数字音频以 I2S 或 PCM 时序做 DAC 转换后向 Speaker 发送仿真信号。

数据的搬移由 RISC-V 控制 DMA 移动内存 DDR 内的数据，用户呼叫 CVI_AUDIO API 时仅在初始或结束对 Audio Codec 做呼叫，用以实现 Codec 硬件的标准初始与标准结束，过程中并不会涉及仿真讯号的撷取，亦无法直接对 RISC-V 使用 DMA 方式做更动，Audio Codec 为工作域的转换者，而 RISC-V/DMA 则是数据搬移者的角色。(参阅下图 9-7)

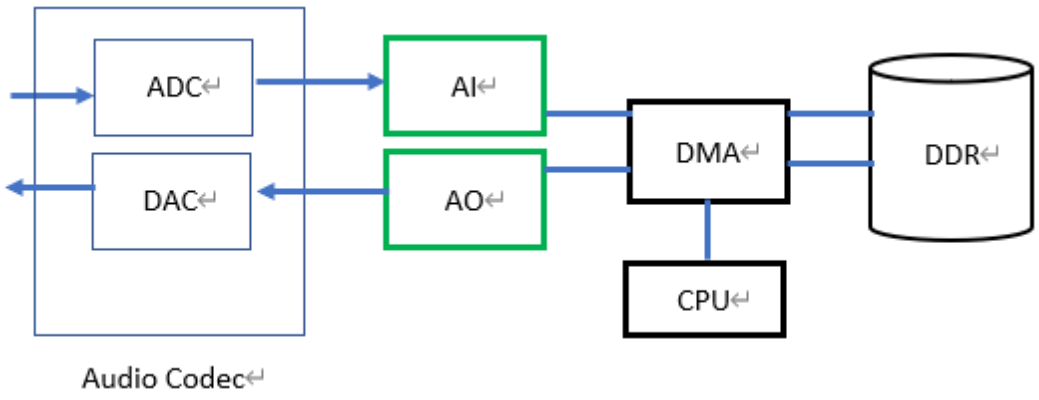


图 10.7: 图 9-7

10.2.2.3 音频接口时序

cvitek 音频时序接口支持 I2S、PCM 时序模式，并依据客制化提供多种方式与 Audio Codec 对接，详细处理器硬件规格可参阅硬件相关文件。Audio Input/Audio Output 控制时钟以同步时序，使用者可参阅 cvi_sample_audio.c 内部 SAMPLE_COMM_AUDIO_Cfg Acodec API，内置 Audio Codec 或是外部 Audio Codec 时序设置方式会有所不同，但对于 cvi_audio API 使用者而言，仅需在初始 Audio Codec 时确认 kernel 可支持并初始化 Codec。对于音框采样率，Cvitek 使用 RISC-V 软件采样，并不会与主时钟有直接关联。Audio Input 设备使用多路复用的 I2S 接收模式时，标准的 I2S 协议只有左右声道这个概念，Audio Input 设备最大支持左右声道各接收 128bit 音频数据，Codec 细部内容请参阅 9.4.4 章节。

10.2.2.4 重采样

音框重采样支持任意两种不同采样率的转换，主要是以 8kHz 倍频为主。重采样支持的输入采样率为：8kHz、11.025kHz、16kHz、22.05kHz、24kHz、32kHz、44.1kHz、48kHz; 支持的输出采样率为：8kHz、11.025kHz，16kHz、22.05kHz、24kHz、32kHz、44.1kHz、48kHz。用者须注意，重采样可支持处理单声道、双声道。Audio Input 的重采样，则重采样的输入采样率与 Audio Input 设备属性配置的采样率相同，重采样的输出采样率必须与 Audio Input 设备属性配置的采样率不相同；Audio Output 的重采样，则重采样的输出采样率与 Audio Output 设备属性配置的采样率相同，重采样的输入采样率必须与 Audio Output 设备属性配置的采样率不相同。

Audio Input-Audio Output 的数据传输方式为系统绑定方式 (system bind)，Audio Input 或 Audio Output 的重采样无效。使用者在 user-get mode 状态下，启用 Audio Input 重采样功能，则可在 CVI_AI_GetFrame 获取数据会取得对应的重采样数据。Audio Output 如果启用重采样功能，则音频数据在发送给 Audio Output 之前，需先执行重采样处理，处理完成后再发送给 Audio Output 通道 (CVI_AO_SendFrame) 进行播放。

对应 API:

- CVI_Resampler_Create: 创建并初始 audio resample。
- CVI_Resampler_GetMaxOutputNum: 依据输入的样本点数，得到重采样后的对应点数。
- CVI_Resampler_Process: 透过此 API，持续传入音框样本数，进行实际的重采样。
- CVI_Resampler_Destroy: 结束重采样流程。

10.2.2.5 语音音质增强 (VQE)

针对语音讯号处理算法，当近端语音讯号遭受到来自远端的回声干扰或是近端平稳噪声的干扰时，可采用 VQE 内的算法功能，进而提高语音讯号的品质。VQE 提供四种解决方案，包括线性回声消除（AEC），非线性回声抑制（AES），语音降噪（NR）和自动增益控制（AGC），算法可支持 8kHz 及 16kHz 采样率，单声道，16 位采样长度。在接下来的页面会介绍常用功能及所使用的参数，若想更详细的了解请参考音频质量调试指南。

参数 **para_fun_config** 可以控制 AEC、AES、NR 和 AGC 功能。参数 **para_spk_fun_config** 可控制扬声器路径 SSP 算法功能，属于 UpVQE。各个位对应的算法功能如下表所示。

表 10.1: 表 9-1: para_fun_config 参数说明

para_fun_config	描述
位 0	0: 关闭 AEC 1: 开启 AEC
位 1	0: 关闭 AES 1: 开启 AES
位 2	0: 关闭 NR 1: 开启 NR
位 3	0: 关闭 AGC 1: 开启 AGC
位 4	0: 关闭 Notch Filter 1: 开启 Notch Filter
位 5	0: 关闭 DC Filter 1: 开启 DC Filter
位 6	0: 关闭 DG 1: 开启 DG
位 7	0: 关闭 Delay 1: 开启 Delay

表 10.2: 表 9-1: para_spk_fun_config 参数说明

para_spk_fun_config	描述 (扬声器路径)
位 0	0: 关闭 AGC 1: 开启 AGC
位 1	0: 关闭 EQ 1: 开启 EQ

VQE 针对 Audio Input 和 Audio Output 两条通路的异同点，分别通过 UpVQE 和 DnVQE 两个调度逻辑来处理两个通路的数据，UpVQE 包含 AEC、AES、NR、AGC。DnVQE 目前不支持。

对应参数可参考表头檔 cvi_comm_aio.h 。

AGC 数据结构如下:

```
CVI_S8 para_agc_max_gain;
CVI_S8 para_agc_target_high;
CVI_S8 para_agc_target_low;
CVI_BOOL para_agc_vad_ena;;
```

NR 数据结构如下:

```
CVI_U16 para_nr_snr_coeff;
CVI_U16 para_nr_init_sile_time;
```

CVI_AI_SetTalkVqeAttr 中的 AI_TALKVQE_CONFIG_S *pstVqeConfig 参数，设定 u32OpenMask 来决定要开启 VQE 何种功能，详细 AGC，NR 成员描述请参阅下方对应子章节。

范例如下表格:

表 10.3: 表 9-2: pstVqeConfig 参数说明

pstAiVqeAttr.u32OpenMask	描述
pstAiVqeAttr.u32OpenMask = AI_TALKVQE_MASK_AGC;	开启 AGC
pstAiVqeAttr.u32OpenMask = AI_TALKVQE_MASK_ANR;	开启 NR
pstAiVqeAttr.u32OpenMask = (AI_TALKVQE_MASK_ANR |AI_TALKVQE_MASK_AGC);	开启 NR, 开启 AGC

- **AEC/AES (Acoustic Echo Cancellation/Acoustic Echo Suppression)**

任何双工通话系统的架构都存在着回声的干扰。回声消除器可以消除通过近端声学路径耦合回麦克风的扬声器输出的回声。采用所提供的解决方案，线性自适应滤波器模块（AEC）搭配非线性回声抑制模块（AES）可以有效地抑制回声，从而提高语音通话品质。

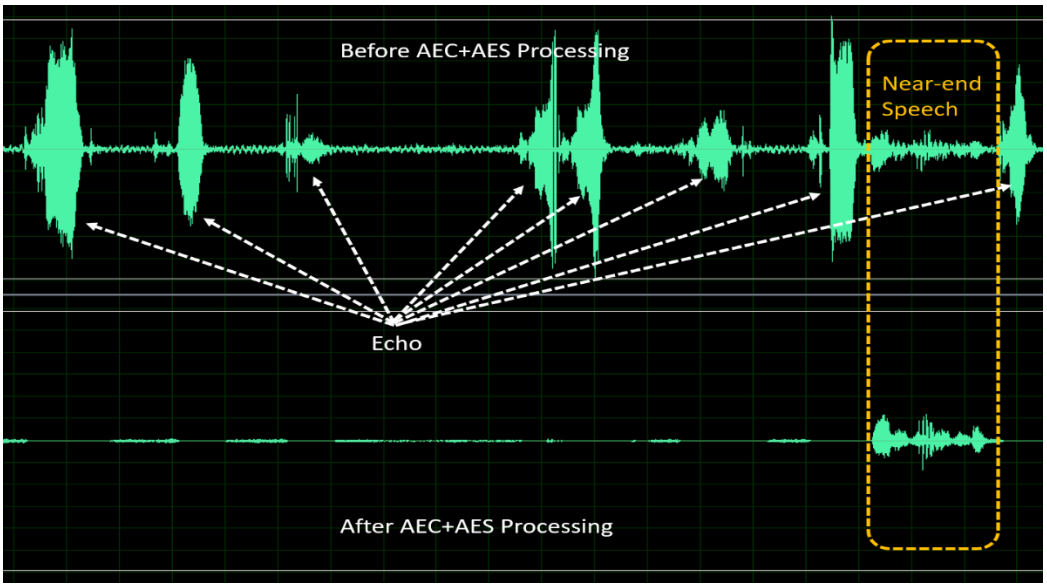


图 10.8: 图 9-1: AEC+AES 处理前后的性能

提供三个可调参数，用于调整 AEC/AES 的性能，它们分别是：

- **para_aec_filter_len**: 自适应滤波器的长度。根据不同样机回声拖尾的时间调整适当的滤波器长度。若选择较长的长度，会导致较高的 MIPS 和功耗。
- **para_aes_std_thrd**: 残留回声判断阈值。值设较大时，近端语音品质较佳但残留回声较多。反之，值设较小时，近端语音品质较差但残留回声较少。
- **para_aes_supp_coeff**: 残留回声抑制力道。值设越大，对残留回声抑制力道越大，但同时也会对近端语音带来越多细节音的丢失/损伤。

表 10.4: 表 9-3: AEC/AES 参数说明

AEC/AES 参数	可调范围	描述
para_aec_filter_len	1 - 13	8kHz 采样率: [1,13] 对应 [20ms,260ms] 16kHz 采样率: [1,13] 对应 [10ms,130ms]
para_aes_std_thrd	0 - 39	0: 残留回声判断阈值最小 39: 残留回声判断阈值最大
para_aes_supp_coeff	0 - 100	0: 残留回声抑制力道最小 100: 残留回声抑制力道最大

- **NR (Noise Reduction)**

NR 模块可以抑制周围的平稳噪音，例如风扇噪音，空调噪音，引擎噪音，白/粉红杂讯，…等等。凭靠着专有的语音智能 Voice Activity Detection（VAD）算法，NR 可以保持住语音信号，同时又可以有效地抑制平稳噪声，从而提高语音通话的品质。

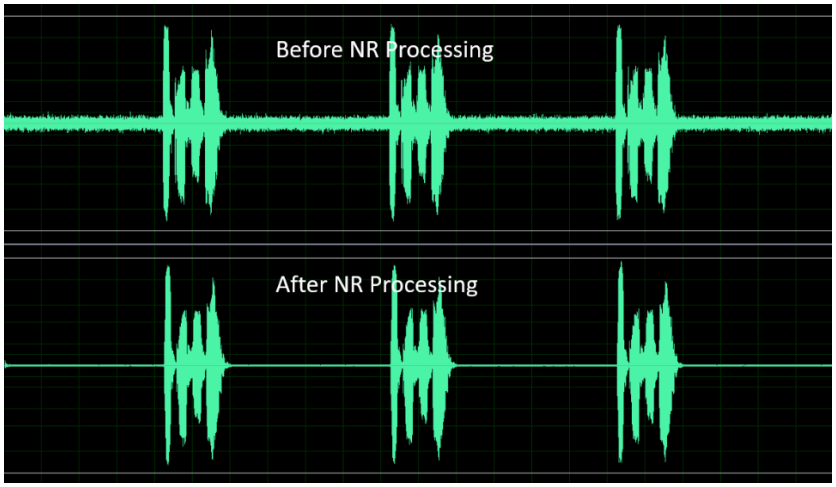


图 10.9: 图 9-2: NR 处理前后的性能

提供三个可调参数，用于调适 NR 的性能，它们分别是：

- para_nr_init_sile_time: 初始静音的时间长度。CODEC 开电瞬间会产生随机无意义的噪音讯号
- para_nr_init_sile_time: 可将这段讯号设置成静音。
- para_nr_snr_coeff: signal-to-Noise。Ratio（SNR）跟踪系数。

  若参数值较大，则 NR 会具有较高的降噪能力，但语音信号可能会较容易失真。

  相反地 ，参数值较小，则 NR 将抑制较少的噪声信号，但会具有较好的语音品质性能。

  下表是基于不同 SNR 环境下，此参数合适的调整范围，在每种 SNR 情况下，参数值越大，对 stationary noise 的抑制力道就越大。

表 10.5: 表: NR 参数说明

NR 参数	可调范围	描述
para_nr_init_sile_time	0 - 250	对应 0s 至 5s， 每阶 20ms

表 10.6: 表 9-4: para_nr_snr_coeff 参数说明

周遭的 SNR 环境	调整范围	描述
低	0 - 3	0: 在降噪方面最不积极 3: 在降噪方面最积极
中	4 - 10	4: 在降噪方面最不积极 10: 在降噪方面最积极
高	11 - 20	11: 在降噪方面最不积极 20: 在降噪方面最积极

- **AGC (Automatic Gain Control)**

AGC 模块是一种信号处理功能，可以自动将输出电平调整到预定范围，以提供更舒适的听觉体验。如果输入信号低于 “Target Low”，则 AGC 会将输出电平往 “Target Low” 调整。另一方面，如果输入信号高于 “Target High”，则 AGC 会将输出电平往 “Target High” 调整。

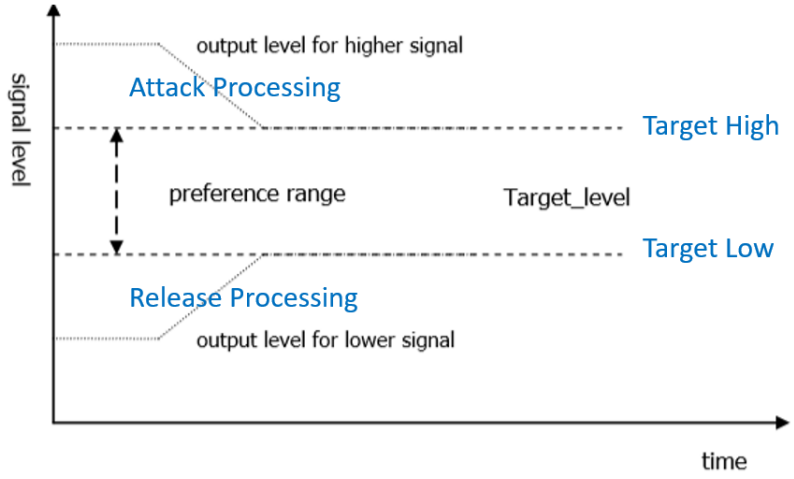


图 10.10: 图 9-3: AGC 调整信号电平

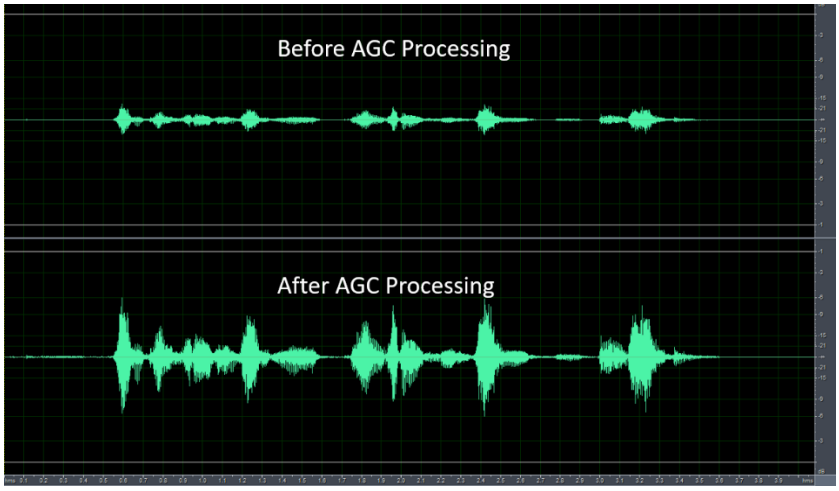


图 10.11: 图 9-4: AGC 处理前后的性能

提供四个可调参数，用于调整 AGC 的性能，它们分别是：

- **para_agc_max_gain:** 此参数是信号可以被放大的最大增益。
- **para_agc_target_high:** 此参数是 AGC 将会去达到的 “Target High” 水平。对于高于 **para_agc_target_high** 的输入信号，AGC 会将其收敛到 **para_agc_target_high** 。
- **para_agc_target_low:** 此参数是 AGC 将会去达到的 “Target Low” 水平。对于低于 **para_agc_target_low** 的输入信号，AGC 会将其收敛到 **para_agc_target_low** 。若在达到 **para_agc_target_low** 之前就已达到 **para_agc_max_gain** ，则 AGC 仅会收敛到 **para_agc_max_gain** 。
- **para_agc_vad_ena:** Speech-activated AGC 功能。开启此功能并同时开启 NR 及 AEC/AES 功能时，能使 AGC 避免放大背景平稳噪声及残留回声，以获得较佳的效果。

表 10.7: 表 9-5: AGC 参数说明

AGC 参数	可调范围	描述
para_agc_max_gain	0 - 6	[0,6] 对应的最大提升增益为 [6dB,42dB]，每阶为 6dB
para_agc_target_high	0 - 36	0 至 36 对应 0dB 至-36dB
para_agc_target_low	0 - 72	0 至 36 对应 0dB 至-72dB
para_agc_vad_ena	0 - 1	0: 关闭 Speech-activated AGC 功能 1: 开启 Speech-activated AGC 功能

10.2.3 音频编码与解码

10.2.3.1 音频编解码流程

Cvitek 音频编解码支持 G711-A-law、G711-Mu-law、G726、ADPCM_IMA，以上编解码使用 RISC-V 软件编解码。用户可以使用 bind mode(绑定模式) 并通过 CVI_Aud_SYS_Bind，将 Audio Input 与 AENC 绑定，进行音框后的编码，亦可将 Audio Output 与 ADEC 绑定后，进行接收编码音框后的解码程序，将音框还原为 PCM/Raw 信号。若不使用绑定模式 (bind mode)，而使用 user get mode(用户获取模式)，用户可通过 CVI_AENC_SendFrame 将实际单一音框送入编码程序进行编码，相对应的也可使用 CVI_ADEC_GetFrame，进行单一音框的解码，请注意在使用用户获取模式时，用户如果调用延迟过久，则有可能造成内部缓存阻塞，致使回传失败的音框。

10.2.3.2 音频编解码协议

CodecProtocol(编码)	Bit Rate (比特率 Kbps)	Playload (原始音框)	Compress Rate (压缩率)	MOS (音质测量)
G711	64	160/320	1:2	4.1
G726	16 、24、32、40	160/320/480	1:4	3.85
ADPCM-IMA	32	160/320/480	1:4	3.7

10.2.3.3 语音帧结构

Cvitek 与音帧结构并未附加额外头文件，内部收音、播音是以音框为单位，而每一个音框皆为 RAW/PCM 数据，G711、G726、ADPCM 格式的编解码，对应的音框也不会增加多余标头，使用者必须透过 CVI_AUDIO_AENC_/CVI_AUDIO_ADEC_ 相关 API 去获得目前编译码音框的信息。此作法优点为，当使用者透过 get_frame API 或是相关 debug 函式去获取音框时，可以立即的辨识该音框是否合乎对应编译码的规则，而不需再去解析标头，并可在做存盘后于计算机端用相关第三方软件验证编译码的结果是否正常。

10.2.4 内置 Codec

10.2.4.1 概述

Cvitek AIP0/AOP0 接口可与内置的 audio codec 或外挂的 audio codec 对接来实现进行声音的录制与播放。用户需要正确配置 AIP0/AOP0 和内置 audio codec 对接的时序与参数才可接收或发送音频数据。另内置 audio codec 仅支持 I2S 模式, ADC codec 需配置为主模式，DAC codec 需配置为从模式。

内置 audio codec 分为模拟和数字部分。模拟部分可以选择由麦克风输入或由 LINEIN 输入，并支持增益调节。数字部分有 ADC 和 DAC，主要负责完成模拟与数字信号之间的转换，ADC 可分别调节左右声道音量，DAC 左右声道音量无法分别调节。另支持静音与撤销静音，撤销静音时会恢复为静音前配置的增益值。

内置 audio code 的工作时钟 (MCLK) 是由 CV182x 所提供，默认采样精度为 16 bit，采样频率为 16Khz。

10.2.4.2 ioctl 函数

内置 Audio Codec 的用户态接口以 ioctl 形式体现，其形式如下：

```
CVI_S32 ioctl (CVI_S32 fd, CVI_UL cmd);
```

该函数是 Linux 标准接口，具备可变参数特性。但在 Audio Codec 中，实际只需要 3 个参数。因此，其语法形式等同于：

```
CVI_S32 ioctl (CVI_S32 fd, CVI_UL cmd, CMD_DATA_TYPE *cmddata);
```

其中，CMD_DATA_TYPE 随参数 cmd 的变化而变化。这 3 个参数的详细描述如下表所示

参数描述	描述	输入/输出
fd	内置 audio codec 设备文件描述符，是调用 open 函数打开内置 audio codec 设备文件之后的返回值。 ADC 与 DAC 需分别配置。	输入
cmd	主要的 cmd 如下：- ACODEC_SOFT_RESET_CTRL: 将内置 Codec 恢复为默认设置 - ACODEC_SET_INPUT_VOL: 输入总音量控制 - ACODEC_GET_INPUT_VOL: 获取输入总音量 - ACODEC_SET_OUTPUT_VOL: 输出总音量控制 - ACODEC_GET_OUTPUT_VOL: 获取输出总音量 - ACODEC_SET_GAIN_MICL: 左声道 MIC 输入的模拟增益控制 - ACODEC_SET_GAIN_MICR: 右声道 MIC 输入的模拟增益控制 - ACODEC_SET_DACL_VOL: 左声道输出音量控制 - ACODEC_SET_DACR_VOL: 右声道输出音量控制 - ACODEC_SET_ADCL_VOL: 左声道输入音量控制 - ACODEC_SET_ADCR_VOL: 右声道输入音量控制 - ACODEC_SET_MICL_MUTE: 左声道 MIC 输入静音控制 - ACODEC_SET_MICR_MUTE: 右声道 MIC 输入静音控制 - ACODEC_SET_DACL_MUTE: 左声道输出静音控制 - ACODEC_SET_DACR_MUTE: 右声道输出静音控制 - ACODEC_GET_GAIN_MICL: 获取模拟左声道 MIC 输入的增益 - ACODEC_GET_GAIN_MICR: 获取模拟右声道 MIC 输入的增益 - ACODEC_GET_DACL_VOL: 获取左声道输出的音量控制 - ACODEC_GET_DACR_VOL: 获取右声道输出的音量控制 - ACODEC_GET_ADCL_VOL: 获取左声道输入的音量控制 - ACODEC_GET_ADCR_VOL: 获取右声道输入的音量控制 - ACODEC_SET_PD_DACL: 左声道输出的下电控制 - ACODEC_SET_PD_DACR: 右声道输出的下电控制 - ACODEC_SET_PD_ADCL: 左声道输入的下电控制 - ACODEC_SET_PD_ADCR: 右声道输入的下电控制 - ACODEC_SET_PD_LINEINL: 左声道 LINEIN 输入的下电控制 - ACODEC_SET_PD_LINEINR: 右声道 LINEIN 输入的下电控制	输入

10.3 API 参考

本章节将依序描述 CVI_AUDIO 各模块 API 使用方式，请注意在使用 CVI_ADUDIO 时请先确认 CVI_SYS_Init 已被调用，才能确保对应的系统组件初始化，CVI_AUDIO_INIT 也必须被调用，才能确保音频模块中的软件已被初始化，使用者注意 CVI_AUDIO_INIT 仅需呼叫一次。在退出 Audio 的使用时，请确认 CVI_AUDIO 相关模块已做了 Disbale，并接着调用 CVI_SYS_Exit。

10.3.1 模块属性 API

10.3.1.1 CVI_AUDIO_INIT

【描述】

初始化 audio 模块。

【语法】

```
CVI_S32 CVI_AUDIO_INIT(void);
```

【参数】

参数名称	描述	输入/输出
None	无需输入值	

【返回值】

参数名称	描述
0	成功。
非 0	失败，其值为错误码 。

【需求】

- 头文件：cvi_comm_aio.h，cvi_audio.h
- 库文件：libcvi_audio.a

【注意事项】

使用 audio 模块前，必需透过此 API 做初始化。否则功能会异常或内存数据错误。

10.3.1.2 CVI_AUDIO_DEINIT

【描述】

初始化 audio 模块。

【语法】

```
CVI_S32 CVI_AUDIO_DEINIT(void);
```

【参数】

参数名称	描述	输入/输出
None	无需输入值	

【返回值】

参数名称	描述
0	成功。
非 0	失败，其值为错误码 。

【需求】

- 头文件：cvi_comm_aio.h，cvi_audio.h
- 库文件：libcvi_audio.a

【注意事项】

离开 audio 模块前，必需透过此 API 做释放模块。否则功能会异常或记忆体数据错误。

10.3.1.3 CVI_AUDIO_SetModParam

【描述】

设置 Audio 模块参数属性。

【语法】

```
CVI_S32 CVI_AUDIO_SetModParam(const AUDIO_MOD_PARAM_S *pstModParam);
```

【参数】

参数名称	描述	输入/输出
pstModParam	AUDIO 模块参数属性指针。	输入

【返回值】

参数名称	描述
0	成功。
非 0	失败，其值为错误码 。

【需求】

- 头文件：cvi_comm_aio.h，cvi_audio.h
- 库文件：libcvi_audio.a

【注意事项】

在各 audio 子模块使用前，需通过此 API 做初始化。

【举例】

无。

10.3.1.4 CVI_AUD_SYS_Bind

【描述】

设置 Audio 模块绑定属性。

【语法】

```
CVI_S32 CVI_AUD_SYS_Bind(const MMF_CHN_S *pstSrcChn, const MMF_CHN_S
↪*pstDestChn);
```

【参数】

参数名称	描述	输入/输出
pstSrcChn	Audio 绑定来源指针。	输入
pstDestChn	Audio 绑定目的指针。	输入

【返回值】

参数名称	描述
0	成功。
非 0	失败，其值为错误码 。

【需求】

- 头文件：cvi_comm_aio.h，cvi_audio.h
- 库文件：libcvi_audio.a

【注意事项】

1. 支持绑定对象为：

   (AudIn -> AudEnc),

   (AudDec ->AudOut),

2. 可参阅 cvi_sample_audio.c 内的 SAMPLE_AUDIO_AiAenc 及 SAMPLE_AUDIO_AdecAo 了解绑定用法。

【举例】

无。

10.3.1.5 CVI_AUDIO_GetModParam

【描述】

获取 AUDIO 模块参数属性。

【语法】

```
CVI_S32 CVI_AUDIO_GetModParam(AUDIO_MOD_PARAM_S *pstModParam);
```

【参数】

参数名称	描述	输入/输出
pstModParam	AUDIO 模块参数属性指针。	输出

【返回值】

参数名称	描述
0	成功。
非 0	失败，其值为错误码 。

【需求】

- 头文件：cvi_comm_aio.h，cvi_audio.h
- 库文件：libcvi_audio.a

【注意事项】

在各 audio 子模块使用前，需透过此 API 做初始化。

【举例】

无。

10.3.1.6 CVI_AUDIO_RegisterVQEModule

【描述】

注册 VQE 模块

【语法】

```
CVI_S32 CVI_AUDIO_RegisterVQEModule(const AUDIO_VQE_REGISTER_S, *pstVqeRegister);
```

【参数】

参数名称	描述	输入/输出
pstVqeRegister	VQE 相关参数属性指针	输入

【返回值】

参数名称	描述
0	成功。
非 0	失败，其值为错误码 。

【需求】

- 头文件：cvi_comm_aio.h，cvi_audio.h
- 库文件：libcvi_audio.a，libcvi_vqe.so，libaec.so

【注意事项】

此 api 已弃用。请使用 CVI_AI_SetTalkVqeAttr，CVI_AI_EnableVqe

【举例】

无。

10.3.1.7 CVI_AENC_RegisterExternalEncoder

【描述】

注册音频编码模块

【语法】

```
CVI_S32 CVI_AENC_RegisterExternalEncoder(CVI_S32 *ps32Handle, const AAC_AENC_
↪ENCODER_S *pstEncoder);
```

【参数】

参数名称	描述	输入/输出
ps32Handle	参数属性指针	输入
pstEncoder	编码模块函数指针	输入

【返回值】

参数名称	描述
0	成功。
非 0	失败，其值为错误码 。

【需求】

- 头文件：cvi_comm_aio.h，cvi_audio.h，cvi_audio_aac_adp.h
- 库文件：libcvi_audio.a，libaacenc2.so

【注意事项】

- 此函数针对 AAC 音频编码使用。
- 函数注册前，需确认 pstEncoder 内的函数指针不为空，否则内部调用时会有错误。
- 相关范例可搭配 SDK 内：sample/audio/aac_sample/cvi_audio_aac_adp.c 调用参考。

- 此函数仅支持 AAC 编码注册，相关 G7xx 系列编码请使用CVI_AENC_CreateChn，可参照范例 cvi_sample_audio.c 内 SAMPLE_AUDIO_AiAenc 。

【举例】

无。

10.3.1.8 CVI_AENC_UnRegisterExternalEncoder

【描述】

注销编码器。

【语法】

```
CVI_S32 CVI_AENC_UnRegisterExternalEncoder(CVI_S32 s32Handle);
```

【参数】

参数名称	描述	输入/输出
s32Handle	注册句柄（注册编码器时获得的句柄）	输入

【返回值】

参数名称	描述
0	成功。
非 0	失败，其值为错误码 。

【需求】

- 头文件：cvi_comm_aio.h，cvi_audio.h，cvi_audio_aac_adp.h
- 库文件：libcvi_audio.a，libaacenc2.so

【注意事项】

- 此函数针对 AAC 音频编码使用。
- 注销编码器前，需要先销毁通过该编码器创建的所有编码通道，未销毁或者销毁过程中调用此接口将返回报错。

【举例】

无。

10.3.1.9 CVI_ADEC_RegisterExternalDecoder

【描述】

注册音频解码模块

【语法】

```
CVI_S32 CVI_ADEC_RegisterExternalDecoder(CVI_S32 *ps32Handle, const ADEC_DECODER_S
↪*pstDecoder);
```

【参数】

参数名称	描述	输入/输出
ps32Handle	参数属性指针	输入
pstDecoder	编码模块函数指针	输入

【返回值】

参数名称	描述
0	成功。
非 0	失败，其值为错误码 。

【需求】

- 头文件：cvi_comm_aio.h，cvi_audio.h ，cvi_audio_aac_adp.h
- 库文件：libcvi_audio.a，libaadec2.so

【注意事项】

- 此函数针对 AAC 音频解码使用。
- 函数注册前，需确认 pstDecoder 内的函数指真不为空，否则内部调用时会有错误。
- 相关范例可搭配 SDK 内：sample/audio/aac_sample/cvi_audio_aac_adp.c 调用参考。
- 此函数仅支持 AAC 译码注册，相关 G7xx 系列请使用 CVI_ADEC_CreateChn，可参照范例 cvi_sample_audio.c 内 SAMPLE_AUDIO_AdecAo 。

【举例】

无。

10.3.1.10 CVI_ADEC_UnRegisterExternalDecoder

【描述】

注销音频解码模块

【语法】

```
CVI_S32 CVI_ADEC_UnRegisterExternalDecoder(CVI_S32 s32Handle);
```

【参数】

参数名称	描述	输入/输出
s32Handle	注册句柄	输入

【返回值】

参数名称	描述
0	成功。
非 0	失败，其值为错误码 。

【需求】

- 头文件：cvi_comm_aio.h，cvi_audio.h，cvi_audio_aac_adp.h
- 库文件：libcvi_audio.a，libaadec2.so

【注意事项】

- 此函数针对 AAC 音频解码使用。

【举例】

无。

10.3.2 音频输入

音频输入实现配置及启用设备、获取音频帧数据等功能。

该功能模块提供以下 API:

- CVI_AI_SetPubAttr：设置 Audio Input 设备属性。
- CVI_AI_GetPubAttr：获取 Audio Input 设备属性。
- CVI_AI_Enable：启用 Audio Input 设备。
- CVI_AI_Disable：禁用 Audio Input 设备。
- CVI_AI_EnableChn：启用 Audio Input 通道。
- CVI_AI_DisableChn：禁用 Audio Input 通道。
- CVI_AI_GetFrame：获取音讯帧。
- CVI_AI_ReleaseFrame：释放音讯帧。
- CVI_AI_SetChnParam：设置 Audio Input 通道参数。
- CVI_AI_GetChnParam：获取 Audio Input 通道参数。
- CVI_AI_EnableReSmp：启用 Audio Input 重采样。
- CVI_AI_DisableReSmp：禁用 Audio Input 重采样。
- CVI_AI_ClrPubAttr：清除 Audio Input 设备属性。
- CVI_AI_SaveFile：开启音频输入保存文件功能。

- CVI_AI_QueryFileStatus： 查询音频输入信道是否处于存盘的状态。
- CVI_AI_EnableAecRefFrame： 在 AEC 不打开的情况下也使使用者能获取到 AEC 参考帧。
- CVI_AI_DisableAecRefFrame： 在 AEC 不打开的情况下禁止获取 AEC 参考帧。
- CVI_AI_SetVolume： 设置 Audio Input 设备音量。
- CVI_AI_GetVolume：获取 Audio Input 设备音量。

10.3.2.1 CVI_AI_SetPubAttr

【描述】

设置 Audio Input 设备属性

【语法】

```
CVI_S32 CVI_AI_SetPubAttr(AUDIO_DEV AiDevId, const AIO_ATTR_S *pstAttr);
```

【参数】

参数名称	描述	输入/输出
AiDevId	音频设备号	输入
pstAttr	Audio Input 设备属性指针	输入

【返回值】

参数名称	描述
0	成功。
非 0	失败，其值为错误码 。

【需求】

- 头文件：cvi_comm_aio.h，cvi_audio.h
- 库文件：libcvi_audio.a

【注意事项】

- 音频输入设备的属性决定了输入数据的格式，输入设备属性包括工作模式、采样率、采样精度、buffer 大小、每帧的采样点数、扩展标志、时钟选择和通道数目。这些属性应与对接 Codec 配置的时序一致，即能成功对接。
- AiDevId 预设为 0，如未要求扩增，超过 2 会返回错误。
- enBitwidth：位深度 8bit、16bit 或 24bit，实际应用中采样精度还受 Audio Codec 限制。
- buffer 大小 AIO_ATTR_S 中的 u32FrmNum 项用于配置 Audio Input 中用于接收音频数据的区块的音频。
- UsrFrmDept：区块音框，建议需大于等于 5。
- enSamplerate 当音频采样率较高时，建议相应地增加每帧的采样点数目。

- 如要将这些采集到的音频数据送编码，则应保证每帧的持续时长不少于 10ms（例如 16K 的采样频率下每帧的采样点数至少应设置为 160），否则解码后声音可能有异常。
- enSoundmode：双声道或单声道语音音框设置
- u32ChnCnt：信道数目信道数目指当前输入设备的 Audio Input 功能的信道数目，需与对接的 Audio Codec 的配置保持一致；支持 1 路、2 路。
- u32EXFlag：扩展标志对 Audio Input 设备无效
- u32ClkSel：时钟设置，在此 processor CV182x 无需特别设置，仅需注意 aio attribute 在 Audio Input/Audio Output 需设定一致。

```
typedef struct cviAIO_ATTR_S {
    AUDIO_SAMPLE_RATE_E enSamplerate;   /* sample rate */
    AUDIO_BIT_WIDTH_E   enBitwidth;     /* bitwidth */
    AIO_MODE_E          enWorkmode; /* master or slave mode */
    AUDIO_SOUND_MODE_E  enSoundmode;    /* momo or steror */
    CVI_U32  u32EXFlag; /* not used in this chip */
    CVI_U32 u32FrmNum;
    /* frame num in buf[2,MAX_AUDIO_FRAME_NUM] */
    CVI_U32 u32PtNumPerFrm;
    /* point num per frame (160/240/320/480/1024/2048) */
    CVI_U32 u32ChnCnt;  /* channel number on FS, valid value:1/2/4/8 */
    CVI_U32 u32ClkSel;  /* 0: AI and AO clock is separate*/
    /* 1: AI and AO clock is inseparate, AI use AO's clock*/
    AIO_I2STYPE_E enI2sType;    /* i2s type */
} AIO_ATTR_S;
```

【举例】

无。

10.3.2.2 CVI_AI_GetPubAttr

【描述】

获取 Audio Input 设备属性

【语法】

```
CVI_S32 CVI_AI_GetPubAttr(AUDIO_DEV AiDevId, AIO_ATTR_S*pstAttr);
```

【参数】

参数名称	描述	输入/输出
AiDevId	音频设备号	输入
pstAttr	Audio Input 设备属性指针	输出

【返回值】

参数名称	描述
0	成功。
非 0	失败，其值为错误码 。

【需求】

- 头文件：cvi_comm_aio.h，cvi_audio.h
- 库文件：libcvi_audio.a

【注意事项】

如未曾初始化或是 CVI_AI_SetPubAttr 未曾呼叫过，则使用者会拿到内容为 0 的指针及数值。AiDevId 预设为 0，如未要求扩增，超过 2 会返回错误。

【举例】

无。

10.3.2.3 CVI_AI_Enable

【描述】

启用 Audio Input 设备

【语法】

```
CVI_S32 CVI_AI_Enable(AUDIO_DEV AiDevId);
```

【参数】

参数名称	描述	输入/输出
AiDevId	音频设备号。设备号预设为 0，在客制化未要求扩充下，设置超过 2，返回错误	输入

【返回值】

参数名称	描述
0	成功。
非 0	失败，其值为错误码 。

【需求】

- 头文件：cvi_comm_aio.h，cvi_audio.h
- 库文件：libcvi_audio.a

【注意事项】

此 API 为最后使能 Audio Input 的函数，在调用前请确认相关属性已设定完成。

【举例】

```
Audio In使用范例:
CVI_S32 i;
CVI_S32 s32Ret;

s32Ret = CVI_AI_SetPubAttr(AiDevId, pstAioAttr);
if (s32Ret) {
```

(下页继续)

(续上页)

```
  printf("%s: CVI_AI_SetPubAttr(%d) failed with %#x\n"
    , __func__,
    AiDevId,
    s32Ret);
  return s32Ret;
}

s32Ret = CVI_AI_Enable(AiDevId);
if (s32Ret) {
                printf("%s: CVI_AI_Enable(%d) failed with %#x\n", __func__, AiDevId,
                       s32Ret);
                return s32Ret;
}
for (i = 0; i < s32AiChnCnt >> pstAioAttr->enSoundmode; i++) {
                s32Ret = CVI_AI_EnableChn(AiDevId, i / (pstAioAttr->enSoundmode + 1));
                if (s32Ret) {
                        printf("%s: CVI_AI_EnableChn(%d,%d) failed with %#x\n", __func__,
                             AiDevId, i, s32Ret);
                        return s32Ret;
                }

                if (bResampleEn == CVI_TRUE) {
                s32Ret = CVI_AI_EnableReSmp(AiDevId, i, enOutSampleRate);
                if (s32Ret) {
                printf("%s: CVI_AI_EnableReSmp(%d,%d) failed with %#x\n",
__func__,
                        AiDevId, i,
s32Ret);
                        return s32Ret;
                }
                }

                if (pstAiVqeAttr != NULL) {
                        CVI_BOOL bAiVqe = CVI_TRUE;
                        s32Ret = CVI_AI_SetTalkVqeAttr(
                                0,
                                0,
                                0,
                                0,
                                (AI_TALKVQE_CONFIG_S *)pstAiVqeAttr);
                                break;
                }

                if (s32Ret) {
                printf("%s: SetAiVqe%d(%d,%d) failed with %#x\n",
                       __func__,
u32AiVqeType,
                       AiDevId, i, s32Ret);
                       return s32Ret;
                }

                if (bAiVqe) {
                s32Ret = CVI_AI_EnableVqe(AiDevId, i);
                if (s32Ret) {
```

(下页继续)

(续上页)

```
                printf("%s: CVI_AI_EnableVqe(%d,%d) failed with %#x\n", __func__,
                                        AiDevId, i, s32Ret);
                return s32Ret;
                }
                }
        }
```

10.3.2.4 CVI_AI_Disable

【描述】

停用 Audio Input 设备

【语法】

```
CVI_S32 CVI_AI_Disable(AUDIO_DEV AiDevId);
```

【参数】

参数名称	描述	输入/输出
AiDevId	音频设备号。设备号预设为 0，在客制化未要求扩充下，设置超过 2，返回错误。	输入

【返回值】

参数名称	描述
0	成功。
非 0	失败，其值为错误码 。

【需求】

- 头文件：cvi_comm_aio.h，cvi_audio.h
- 库文件：libcvi_audio.a

【注意事项】

此 API 为停止 Audio Input 设备的最后一道程序，禁用 Audio Input 设备前必须先禁用该设备下已启用的所有 Audio Input 通道，如有 AENC 或 Audio Output 相接，请先停止与之关联再调用此。

【举例】

无。

10.3.2.5 CVI_AI_EnableChn

【描述】

启用 Audio Input 通道

【语法】

```
CVI_S32 CVI_AI_EnableChn(AUDIO_DEV AiDevId, AI_CHN AiChn);
```

【参数】

参数名称	描述	输入/输出
AiDevId	音频设备号。设备号预设为 0，在客制化未要求扩充下，设置超过 2，返回错误	输入
AiChn	音频输入通道号。与声道模式描述入 enSoundmode 无关，用户可透过不同 AiChn 在同 AiDevId 下获取同设备声源。	输入

【返回值】

参数名称	描述
0	成功。
非 0	失败，其值为错误码 。

【需求】

- 头文件：cvi_comm_aio.h，cvi_audio.h
- 库文件：libcvi_audio.a

【注意事项】

启用 Audio Input 通道前，必须先启用其所属 Audio Input 设备。

【举例】

无。

10.3.2.6 CVI_AI_DisableChn

【描述】

禁用 Audio Input 通道。

【语法】

```
CVI_S32 CVI_AI_DisableChn(AUDIO_DEV AiDevId, AI_CHN AiChn);
```

【参数】

参数名称	描述	输入/输出
AiDevId	音频设备号。设备号预设为 0，在客制化未要求扩充下，设置超过 2，返回错误	输入
AiChn	音频输入通道号。与声道模式描述入 enSoundmode 无关，用户可透过不同 AiChn 在同 AiDevId 下获取同设备声源。	输入

【返回值】

参数名称	描述
0	成功。
非 0	失败，其值为错误码 。

【需求】

- 头文件：cvi_comm_aio.h，cvi_audio.h
- 库文件：libcvi_audio.a

【注意事项】

请在CVI_AI_Disable 前呼叫此 API，避免通道参数残留。

【举例】

无。

10.3.2.7 CVI_AI_GetFrame

【描述】

获取音框

【语法】

```
CVI_S32 CVI_AI_GetFrame(AUDIO_DEV AiDevId, AI_CHN AiChn, AUDIO_FRAME_S *pstFrm,
↪ AEC_FRAME_S *pstAecFrm , CVI_S32 s32MilliSec);
```

【参数】

参数名称	描述	输入/输出
AiDevId	音频设备号。设备号预设为 0，在客制化未要求扩充下，设置超过 2，返回错误	输入
AiChn	音频输入通道号。与声道模式描述入 enSoundmode 无关，用户可透过不同 AiChn 在同 AiDevId 下获取同设备声源。	输入
pstFrm	音频帧结构体指针。用户由此结构体指针获得音框。	输出
pstAecFrm	回声抵消参考帧结构体指针。	输出
s32MilliSec	获取数据的超时时间 -1 表示阻塞模式，无数据时一直等待； 0 表示非阻塞模式，无数据时则报错返回； >0 表示阻塞 s32MilliSec 毫秒，超时则报错返回	输入

【返回值】

参数名称	描述
0	成功。
非 0	失败，其值为错误码 。

【需求】

- 头文件：cvi_comm_aio.h，cvi_audio.h
- 库文件：libcvi_audio.a

【注意事项】

- s32MilliSec 的值必须大于等于-1，等于-1 时采用阻塞模式获取数据，等于 0 时采用非阻塞模式获取数据，大于 0 时，阻塞 s32MilliSec 毫秒后，没有数据则返回超时并报错。
- 获取音频帧数据前，必须先使能对应的 Audio Input 通道。
- 欲获取 AEC 音框，请先确定 VQE 内 AEC 已开启。

【举例】

无。

10.3.2.8 CVI_AI_ReleaseFrame

【描述】

释放音框。

【语法】

```
CVI_S32 CVI_AI_ReleaseFrame(AUDIO_DEV AiDevId, AI_CHN AiChn, const AUDIO_FRAME_
↪S *pstFrm, const AEC_FRAME_S *pstAecFrm);
```

【参数】

参数名称	描述	输入/输出
AiDevId	音频设备号。设备号预设为 0，在客制化未要求扩充下，设置超过 2，返回错误	输入
AiChn	音频输入通道号。与声道模式描述入 enSoundmode 无关，用户可透过不同 AiChn 在同 AiDevId 下获取同设备声源。	输入
pstFrm	音频帧结构体指针。用户由此结构体指针获得音框。	输入
pstAecFrm	回声抵消参考帧结构体指针。	输入

【返回值】

参数名称	描述
0	成功。
非 0	失败，其值为错误码 。

【需求】

- 头文件：cvi_comm_aio.h，cvi_audio.h
- 库文件：libcvi_audio.a

【注意事项】

如果不需要释放回声抵消参考帧，pstAecFrm 置为 NULL 即可。

【举例】

无。

10.3.2.9 CVI_AI_SetChnParam

【描述】

设置 Audio Input 通道参数。

【语法】

```
CVI_S32 CVI_AI_SetChnParam(AUDIO_DEV AiDevId, AI_CHN AiChn, const AI_CHN_
↪PARAM_S *pstChnParam);
```

【参数】

参数名称	描述	输入/输出
AiDevId	音频设备号。设备号预设为 0，在客制化未要求扩充下，设置超过 2，返回错误	输入
AiChn	音频输入通道号。与声道模式描述入 enSoundmode 无关，用户可透过不同 AiChn 在同 AiDevId 下获取同设备声源。	输入
pstChnParam	音频通道参数	输入

【返回值】

参数名称	描述
0	成功。
非 0	失败，其值为错误码 。

【需求】

- 头文件：cvi_comm_aio.h，cvi_audio.h
- 库文件：libcvi_audio.a

【注意事项】

通道参数目前只有一个成员变量，用于设置用户获取音频帧的区块深度，

默认深度为 0。该成员变量的值不能大于 30。

【举例】

无。

10.3.2.10 CVI_AI_GetChnParam

【描述】

获取 Audio Input 通道参数。

【语法】

```
CVI_S32 CVI_AI_GetChnParam(AUDIO_DEV AiDevId, AI_CHN AiChn, AI_CHN_PARAM_S
↪*pstChnParam);
```

【参数】

参数名称	描述	输入/输出
AiDevId	音频设备号。设备号预设为 0，在客制化未要求扩充下，设置超过 2，返回错误	输入
AiChn	音频输入通道号。与声道模式描述入 enSoundmode 无关，用户可透过不同 AiChn 在同 AiDevId 下获取同设备声源。	输入
pstChnParam	音频通道参数	输出

【返回值】

参数名称	描述
0	成功。
非 0	失败，其值为错误码 。

【需求】

- 头文件：cvi_comm_aio.h，cvi_audio.h 此文件提及的音频模块属于 Middleware 多媒体层接口之一
- 库文件：libcvi_audio.a

【注意事项】

通道参数目前只有一个成员变量，用于设置用户获取音频帧的区块深度。

【举例】

无。

10.3.2.11 CVI_AI_EnableReSmp

【描述】

启用 Audio Input 重采样。

【语法】

```
CVI_S32 CVI_AI_EnableReSmp(AUDIO_DEV AiDevId, AI_CHN AiChn, AUDIO_SAMPLE_
↪RATE_E enOutSampleRate);
```

【参数】

参数名称	描述	输入/输出
AiDevId	音频设备号。设备号预设为 0，在客制化未要求扩充下，设置超过 2，返回错误	输入
AiChn	音频输入通道号。与声道模式描述入 enSoundmode 无关，用户可透过不同 AiChn 在同 AiDevId 下获取同设备声源。	输入
enOutSampleRat	音频重采样的输出采样率。	输入

【返回值】

参数名称	描述
0	成功。
非 0	失败，其值为错误码 。

【需求】

- 头文件：cvi_comm_aio.h，cvi_audio.h
- 库文件：libcvi_audio.a，libcvi_vqe.so，libcvi_RES1.so

【注意事项】

在使用CVI_AUD_SYS_Bind 的状态下，不支持此 API

采样率支持程度如下:

```
typedef enum _AUDIO_SAMPLE_RATE_E {
  AUDIO_SAMPLE_RATE_8000  = 8000,   /* 8K samplerate*/
  AUDIO_SAMPLE_RATE_11025 = 11025,  /* 11.025K samplerate*/
  AUDIO_SAMPLE_RATE_16000 = 16000,  /* 16K samplerate*/
  AUDIO_SAMPLE_RATE_22050 = 22050,  /* 22.050K samplerate*/
  AUDIO_SAMPLE_RATE_24000 = 24000,  /* 24K samplerate*/
  AUDIO_SAMPLE_RATE_32000 = 32000,  /* 32K samplerate*/
  AUDIO_SAMPLE_RATE_44100 = 44100,  /* 44.1K samplerate*/
  AUDIO_SAMPLE_RATE_48000 = 48000,  /* 48K samplerate*/
  AUDIO_SAMPLE_RATE_64000 = 64000,  /* 64K samplerate*/
  AUDIO_SAMPLE_RATE_BUTT,
} AUDIO_SAMPLE_RATE_E;
```

【举例】

无。

10.3.2.12 CVI_AI_DisableReSmp

【描述】

关闭 Audio Input 重采样功能。

【语法】

```
CVI_S32 CVI_AI_DisableReSmp(AUDIO_DEV AiDevId, AI_CHN AiChn);
```

【参数】

参数名称	描述	输入/输出
AiDevId	音频设备号。设备号预设为 0，在客制化未要求扩充下，设置超过 2，返回错误	输入
AiChn	音频输入通道号。与声道模式描述入 enSoundmode 无关，用户可透过不同 AiChn 在同 AiDevId 下获取同设备声源。	输入

【返回值】

参数名称	描述
0	成功。
非 0	失败，其值为错误码 。

【需求】

- 头文件：cvi_comm_aio.h，cvi_audio.h
- 库文件：libcvi_audio.a，libcvi_RES1.so，libcvi_vqe.so

【注意事项】

要求在调用此接口之前，先禁用使用该 Audio Input 设备相应通道音频数据的 AENC 信道和 Audio Output 信道，否则可能导致该接口调用失败。

【举例】

无。

10.3.2.13 CVI_AI_ClrPubAttr

【描述】

清空 Pub 属性。

【语法】

```
CVI_S32 CVI_AI_ClrPubAttr(AUDIO_DEV AiDevId);
```

【参数】

参数名称	描述	输入/输出
AiDevId	音频设备号。设备号预设为 0，在客制化未要求扩充下，设置超过 2，返回错误	输入

【返回值】

参数名称	描述
0	成功。
非 0	失败，其值为错误码 。

【需求】

- 头文件：cvi_comm_aio.h，cvi_audio.h
- 库文件：libcvi_audio.a

【注意事项】

清除设备属性前，需要先停止设备。

清除设备属性前，建议停止所有内部相连或是 user-get mode 之动作，避免底层仍在搬移音框而造成行为异常。

【举例】

无。

10.3.2.14 CVI_AI_SaveFile

【描述】

开启音频输入保存文件功能。

【语法】

```
CVI_S32 CVI_AI_SaveFile(AUDIO_DEV AiDevId, AI_CHN AiChn, const AUDIO_SAVE_FILE_
↪INFO_S *pstSaveFileInfo);
```

【参数】

参数名称	描述	输入/输出
AiDevId	音频设备号。设备号预设为 0，在客制化未要求扩充下，设置超过 2，返回错误	输入
AiChn	音频输入通道号。与声道模式描述入 enSoundmode 无关，用户可透过不同 AiChn 在同 AiDevId 下获取同设备声源。	输入
pstSaveFileInfo	音频保存文件属性结构体指针。	输入

【返回值】

参数名称	描述
0	成功。
非 0	失败，其值为错误码 。

【需求】

- 头文件：cvi_comm_aio.h，cvi_audio.h
- 库文件：libcvi_audio.a

【注意事项】

此接口仅用于 Audio Input-AENC、Audio Input-Audio Output 非系统绑定模式下 dump Audio Input 使用。

【举例】

无。

10.3.2.15 CVI_AI_QueryFileStatus

【描述】

查询音频输入通道是否处于存文件的状态。

【语法】

```
CVI_S32 CVI_AI_QueryFileStatus(AUDIO_DEV AiDevId, AI_CHN AiChn, AUDIO_FILE_
↪STATUS_S* pstFileStatus);
```

【参数】

参数名称	描述	输入/输出
AiDevId	音频设备号。设备号预设为 0，在客制化未要求扩充下，设置超过 2，返回错误	输入
AiChn	音频输入通道号。与声道模式描述入 enSoundmode 无关，用户可透过不同 AiChn 在同 AiDevId 下获取同设备声源。	输入
pstFileStatus	档案状态属性结构体指针。	输出

【返回值】

参数名称	描述
0	成功。
非 0	失败，其值为错误码 。

【需求】

- 头文件：cvi_comm_aio.h，cvi_audio.h
- 库文件：libcvi_audio.a

【注意事项】

- 此接口主要用于除错，一般流程不会用到。
- 此接口仅用于 Audio Input-AENC、Audio Input-Audio Output 非系统绑定模式下 dump Audio Input 使用。
- 此接口用于查询音频输入通道是否处于存文件的状态，当用户调用CVI_AI_SaveFile 存储文件后，可调用此接口查询存储的文件是否达到了指定的大小，如果 pstFileStatus 的 bSaving 为 CVI_TRUE，说明还没有达到指定大小，为 CVI_FALSE 则已经达到指定大小。

【举例】

无。

10.3.2.16 CVI_AI_EnableAecRefFrame

【描述】

在 AEC 不打开的情况下也使用户能获取到 AEC 参考帧。

【语法】

```
CVI_S32 CVI_AI_EnableAecRefFrame(AUDIO_DEV AiDevId, AI_CHN AiChn, AUDIO_DEV␣
↪AoDevId, AO_CHN AoChn);
```

【参数】

参数名称	描述	输入/输出
AiDevId	音频设备号。设备号预设为 0，在客制化未要求扩充下，设置超过 2，返回错误	输入
AiChn	音频输入通道号。与声道模式描述入 enSoundmode 无关，用户可透过不同 AiChn 在同 AiDevId 下获取同设备声源。	输入
AoDevId	用于获取 AEC 参考帧的 Audio Output 设备号。	输入
AoChn	用于获取 AEC 参考帧的 Audio Output 通道号描述	输入

【返回值】

参数名称	描述
0	成功。
非 0	失败，其值为错误码 。

【需求】

- 头文件：cvi_comm_aio.h，cvi_audio.h
- 库文件：libcvi_audio.a

【注意事项】

无。

【举例】

无。

10.3.2.17 CVI_AI_DisableAecRefFrame

【描述】

在 AEC 不打开的情况下禁止获取 AEC 参考帧。

【语法】

```
CVI_S32 CVI_AI_DisableAecRefFrame(AUDIO_DEV AiDevId, AI_CHN AiChn);
```

【参数】

参数名称	描述	输入/输出
AiDevId	音频设备号。设备号预设为 0，在客制化未要求扩充下，设置超过 2，返回错误	输入
AiChn	音频输入通道号。与声道模式描述入 enSoundmode 无关，用户可透过不同 AiChn 在同 AiDevId 下获取同设备声源。	输入

【返回值】

参数名称	描述
0	成功。
非 0	失败，其值为错误码 。

【需求】

- 头文件：cvi_comm_aio.h，cvi_audio.h
- 库文件：libcvi_audio.a

【注意事项】

无。

【举例】

无。

10.3.2.18 CVI_AI_SetVolume

【描述】

设置输入音量。

【语法】

```
CVI_S32 CVI_AI_SetVolume(AUDIO_DEV AiDevId, CVI_S32 s32VolumeStep);
```

【参数】

参数名称	描述	输入/输出
AiDevId	音频设备号。设备号预设为 0，在客制化未要求扩充下，设置超过 2，返回错误	输入
s32VolumeStep	输入音量放大步阶 [24-0, 0:mute]。	输入

【返回值】

参数名称	描述
0	成功。
非 0	失败，其值为错误码 。

【需求】

- 头文件：cvi_comm_aio.h，cvi_audio.h
- 库文件：libcvi_audio.a/libcvi_audio.so

【注意事项】

无。

【举例】

无。

10.3.2.19 CVI_AI_GetVolume

【描述】

获取输入音量。

【语法】

```
CVI_S32 CVI_AI_GetVolume(AUDIO_DEV AiDevId, CVI_S32 *ps32VolumeStep);
```

【参数】

参数名称	描述	输入/输出
AiDevId	音频设备号。设备号预设为 0，在客制化未要求扩充下，设置超过 2，返回错误	输入
ps32VolumeStep	音量步阶指针 [24-0, 0:mute]。	输出

【返回值】

参数名称	描述
0	成功。
非 0	失败，其值为错误码 。

【需求】

- 头文件：cvi_comm_aio.h，cvi_audio.h
- 库文件：libcvi_audio.a/libcvi_audio.so

【注意事项】

无。

【举例】

无。

10.3.3 语音音质增强 API

- CVI_AI_SetVqeAttr：设置 Audio Input 的声音质量增强功能相关属性。
- CVI_AI_SetTalkVqeAttr：设置 Audio Input 的语音质量增强功能语音相关属性。
- CVI_AI_GetTalkVqeAttr：获取 Audio Input 的语音质量增强功能语音相关属性。
- CVI_AI_SetRecordVqeAttr：设置 Audio Input 的录音质量增强功能相关属性。(目前不支持);
- CVI_AI_GetRecordVqeAttr：获取 Audio Input 的录音质量增强功能相关属性。(目前不支持);
- CVI_AI_EnableVqe：使能 Audio Input 的声音质量增强功能。
- CVI_AI_DisableVqe：禁用 Audio Input 的声音质量增强功能。
- CVI_AI_SetTrackMode：设置声道模式。

- CVI_AI_GetTrackMode：获取声道模式。
- CVI_AO_SetVqeAttr：设置 Audio Output 的声音质量增强功能相关属性。
- CVI_AO_GetVqeAttr：获取 Audio Output 的声音质量增强功能相关属性。
- CVI_AO_EnableVqe：使能 Audio Output 的声音质量增强功能。
- CVI_AO_DisableVqe：禁用 Audio Output 的声音质量增强功能。
- CVI_VQE_PathSelect：VQE 算法路径设置。

10.3.3.1 CVI_AI_SetVqeAttr

【描述】

设置 Audio Input 的声音质量增强功能相关属性。

【语法】

```
CVI_S32 CVI_AI_SetVqeAttr(AUDIO_DEV AiDevId, AI_CHN AiChn, AUDIO_DEV AoDevId,␣
↪AO_CHN AoChn, AI_VQE_CONFIG_S *pstVqeConfig);
```

【参数】

参数名称	描述	输入/输出
AiDevId	音频设备号。设备号预设为 0，在客制化未要求扩充下，设置超过 2，返回错误	输入
AiChn	音频输入通道号。与声道模式描述入 enSoundmode 无关，用户可透过不同 AiChn 在同 AiDevId 下获取同设备声源。	输入
AoDevId	Audio Output 设备号，用于回声抵消。	输入
AoChn	用于回声抵消的 Audio Output 通道号。	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】

参数名称	描述
0	成功。
非 0	失败，其值为错误码 。

【需求】

- 头文件：cvi_comm_aio.h，cvi_audio.h
- 库文件：libcvi_audio.a，libcvi_RES1.so，libcvi_vqe.so，

【注意事项】

启用声音质量增强功能前必须先设置相对应 Audio Input 通道的声音质量增强功能相关属性。

设置 Audio Input 的声音质量增强功能相关属性前，必须先使能对应的 Audio Input 通道。

相同 Audio Input 信道的声音质量增强功能不支持动态设置属性，重新设置 Audio Input 通道的声音质量增强功能相关属性时，需要先关闭 Audio Input 通道的声音质量功能，再设置 Audio Input 通道的声音质量增强功能相关属性。

【举例】

无。

10.3.3.2 CVI_AI_SetTalkVqeAttr

【描述】

设置 Audio Input 的声音质量增强功能（Talk）相关属性。

【语法】

```
CVI_S32 CVI_AI_SetTalkVqeAttr(AUDIO_DEV AiDevId, AI_CHN AiChn, AUDIO_DEV AoDevId,
↪ AO_CHN AoChn, AI_TALKVQE_CONFIG_S *pstVqeConfig);
```

【参数】

参数名称	描述	输入/输出
AiDevId	音频设备号。设备号预设为 0，在客制化未要求扩充下，设置超过 2，返回错误	输入
AiChn	音频输入通道号。与声道模式描述入 enSoundmode 无关，用户可透过不同 AiChn 在同 AiDevId 下获取同设备声源。	输入
AoDevId	Audio Output 设备号，用于回声抵消。	输入
AoChn	用于回声抵消的 Audio Output 通道号。	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】

参数名称	描述
0	成功。
非 0	失败，其值为错误码 。

【需求】

- 头文件：cvi_comm_aio.h，cvi_audio.h
- 库文件：libcvi_audio.a，libcvi_RES1.so，libcvi_vqe.so，libssp.so

【注意事项】

Talk Vqe 主要在 IPC 场景下使用。

相同 Audio Input 信道的声音质量增强功能不支持动态设置属性，重新设置 Audio Input 通道的声音质量增强功能相关属性时，需要先关闭 Audio Input 通道的声音质量功能，再设置 Audio Input 通道的声音质量增强功能相关属性。

【举例】

无。

10.3.3.3 CVI_AI_GetTalkVqeAttr

【描述】

获取 Audio Input 的声音质量增强功能（Talk）相关属性

【语法】

```
CVI_S32 CVI_AI_GetTalkVqeAttr(AUDIO_DEV AiDevId, AI_CHN AiChn, AI_TALKVQE_
↪CONFIG_S *pstVqeConfig);
```

【参数】

参数名称	描述	输入/输出
AiDevId	音频设备号。设备号预设为 0，在客制化未要求扩充下，设置超过 2，返回错误	输入
AiChn	音频输入通道号。与声道模式描述入 enSoundmode 无关，用户可透过不同 AiChn 在同 AiDevId 下获取同设备声源。	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】

参数名称	描述
0	成功。
非 0	失败，其值为错误码 。

【需求】

- 头文件：cvi_comm_aio.h，cvi_audio.h
- 库文件：libcvi_audio.a，libcvi_RES1.so，libcvi_vqe.so，libssp.so

【注意事项】

获取声音质量增强功能相关属性前必须先设置相对应 Audio Input 通道的声音质量增强功能相关属性。

【举例】

无。

10.3.3.4 CVI_AI_SetRecordVqeAttr

【描述】

设置 Audio Input 的声音质量增强功能（Record）相关属性。

【语法】

```
CVI_S32 CVI_AI_SetRecordVqeAttr(AUDIO_DEV AiDevId, AI_CHN AiChn, const AI_
↪RECORDVQE_CONFIG_S *pstVqeConfig);
```

【参数】

参数名称	描述	输入/输出
AiDevId	音频设备号。设备号预设为 0，在客制化未要求扩充下，设置超过 2，返回错误	输入
AiChn	音频输入通道号。与声道模式描述入 enSoundmode 无关，用户可透过不同 AiChn 在同 AiDevId 下获取同设备声源。	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】

参数名称	描述
0	成功。
非 0	失败，其值为错误码 。

【需求】

- 头文件：cvi_comm_aio.h，cvi_audio.h
- 库文件：libcvi_audio.a，libcvi_RES1.so，libcvi_vqe.so，libssp.so

【注意事项】

CVI182x 目前语音算法主要设定是由CVI_AI_SetTalkVqeAttr 调用，使用者用此 RecordVqeAttr (CVI_AI_SetRecordVqeAttr/ GetRecordVqeAttr) 可能对于算法会不支持或无法设入，因此不建议使用。

【举例】

无。

10.3.3.5 CVI_AI_GetRecordVqeAttr

【描述】

设置 Audio Input 的声音质量增强功能（Record）相关属性。

【语法】

```
CVI_AI_GetRecordVqeAttr (AUDIO_DEV AiDevId, AI_CHN AiChn, const AI_RECORDVQE_
↪CONFIG_S *pstVqeConfig);
```

【参数】

参数名称	描述	输入/输出
AiDevId	音频设备号。设备号预设为 0，在客制化未要求扩充下，设置超过 2，返回错误	输入
AiChn	音频输入通道号。与声道模式描述入 enSoundmode 无关，用户可透过不同 AiChn 在同 AiDevId 下获取同设备声源。	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】

参数名称	描述
0	成功。
非 0	失败，其值为错误码 。

【需求】

- 头文件：cvi_comm_aio.h，cvi_audio.h
- 库文件：libcvi_audio.a，libcvi_RES1.so，libcvi_vqe.so，libssp.so

【注意事项】

CVI182x 目前语音算法主要设定是由CVI_AI_GetTalkVqeAttr 调用，使用者用此 RecordVqeAttr (CVI_AI_SetRecordVqeAttr/ GetRecordVqeAttr) 可能对于算法会不支持或无法设入，因此不建议使用。

【举例】

无。

10.3.3.6 CVI_AI_EnableVqe

【描述】

启用 Audio Input 的声音质量增强功能。

【语法】

```
CVI_S32 CVI_AI_EnableVqe(AUDIO_DEV AiDevId, AI_CHN AiChn);
```

【参数】

参数名称	描述	输入/输出
AiDevId	音频设备号。设备号预设为 0，在客制化未要求扩充下，设置超过 2，返回错误	输入
AiChn	音频输入通道号。与声道模式描述入 en Soundmode 无关，用户可透过不同 AiChn 在同 AiDevId 下获取同设备声源。	输入

【返回值】

参数名称	描述
0	成功。
非 0	失败，其值为错误码 。

【需求】

- 头文件：cvi_comm_aio.h，cvi_audio.h
- 库文件：libcvi_audio.a，libcvi_RES1.so，libcvi_vqe.so，libssp.so

【注意事项】

- 启用声音质量增强功能前必须先启用相对应的 Audio Input 通道。
- 多次使能相同 Audio Input 通道的声音质量增强功能时，返回成功。
- 禁用 Audio Input 通道后，如果重新启用 Audio Input 通道，并使用声音质量增强功能，需调用此接口重新启用声音质量增强功能。

【举例】

无。

10.3.3.7 CVI_AI_DisableVqe

【描述】

禁用 Audio Input 的声音质量增强功能

【语法】

```
CVI_S32 CVI_AI_DisableVqe(AUDIO_DEV AiDevId, AI_CHN AiChn);
```

【参数】

参数名称	描述	输入/输出
AiDevId	音频设备号。设备号预设为 0，在客制化未要求扩充下，设置超过 2，返回错误	输入
AiChn	音频输入通道号。与声道模式描述入 enSoundmode 无关，用户可透过不同 AiChn 在同 AiDevId 下获取同设备声源。	输入

【返回值】

参数名称	描述
0	成功。
非 0	失败，其值为错误码 。

【需求】

- 头文件：cvi_comm_aio.h，cvi_audio.h
- 库文件：libcvi_audio.a，libcvi_RES1.so，libcvi_vqe.so，libssp.so

【注意事项】

- 不再使用 Audio Input 声音质量增强功能时，应该调用此接口将其禁用。

【举例】

无。

10.3.3.8 CVI_AI_SetTrackMode

【描述】

设置 Audio Input 声道模式

【语法】

```
CVI_S32 CVI_AI_SetTrackMode(AUDIO_DEV AiDevId, AUDIO_TRACK_MODE_E
↪enTrackMode);
```

【参数】

参数名称	描述	输入/输出
AiDevId	音频设备号。设备号预设为 0，在客制化未要求扩充下，设置超过 2，返回错误	输入
enTrackMode	typedef enum _AUDIO_TRACK_MODE_E { AUDIO_TRACK_NORMAL = 0, AUDIO_TRACK_BOTH_LEFT = 1, AUDIO_TRACK_BOTH_RIGHT = 2, AUDIO_TRACK_EXCHANGE = 3, AUDIO_TRACK_MIX = 4, AUDIO_TRACK_LEFT_MUTE = 5, AUDIO_TRACK_RIGHT_MUTE = 6, AUDIO_TRACK_BOTH_MUTE = 7, AUDIO_TRACK_BUTT } AUDIO_TRACK_MODE_E;	输入

【返回值】

参数名称	描述
0	成功。
非 0	失败，其值为错误码 。

【需求】

- 头文件：cvi_comm_aio.h，cvi_audio.h
- 库文件：libcvi_audio.a，libcvi_RES1.so，libcvi_vqe.so，libssp.so

【注意事项】

- Audio Input 设备工作在 I2S 模式时，支持获取声道模式，PCM 模式下不支持。
- 在 Audio Input 设备成功启用后再调用此接口。
- TrackMode 能力与 audio codec 有关，如客户端使用自己的 codec，设置可能有所不同。

【举例】

无。

10.3.3.9 CVI_AI_GetTrackMode

【描述】

获取 Audio Input 声道模式

【语法】

```
CVI_S32 CVI_AI_GetTrackMode(AUDIO_DEV AiDevId, AUDIO_TRACK_MODE_E
↪*penTrackMode);
```

【参数】

参数名称	描述	输入/输出
AiDevId	音频设备号。设备号预设为 0，在客制化未要求扩充下，设置超过 2，返回错误	输入
enTrackMode	typedef enum _AUDIO_TRACK_MODE_E { AUDIO_TRACK_NORMAL = 0, AUDIO_TRACK_BOTH_LEFT = 1, AUDIO_TRACK_BOTH_RIGHT = 2, AUDIO_TRACK_EXCHANGE = 3, AUDIO_TRACK_MIX = 4, AUDIO_TRACK_LEFT_MUTE = 5, AUDIO_TRACK_RIGHT_MUTE = 6, AUDIO_TRACK_BOTH_MUTE = 7, AUDIO_TRACK_BUTT } AUDIO_TRACK_MODE_E;	输出

【返回值】

参数名称	描述
0	成功。
非 0	失败，其值为错误码 。

【需求】

- 头文件：cvi_comm_aio.h，cvi_audio.h
- 库文件：libcvi_audio.a，libcvi_RES1.so，libcvi_vqe.so，libssp.so

【注意事项】

- Audio Input 设备工作在 I2S 模式时，支持获取声道模式，PCM 模式下不支持。
- 在 Audio Input 设备成功启用后再调用此接口。
- TrackMode 能力与 audio codec 有关，如客户端使用自己的 codec，设置可能有所不同。

【举例】

无。

10.3.3.10 CVI_AO_SetVqeAttr

【描述】

设置 Audio Output 的声音质量增强功能相关属性。

【语法】

```
CVI_S32 CVI_AO_SetVqeAttr(AUDIO_DEV AoDevId, AO_CHN AoChn, AO_VQE_CONFIG_S
↪*pstVqeConfig);
```

【参数】

参数名称	描述	输入/输出
AoDevId	音频设备号。设备号预设为 0，在客制化未要求扩充下，设置超过 2，返回错误。	输入
AoChn	音频输出通道号	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】

参数名称	描述
0	成功。
非 0	失败，其值为错误码 。

【需求】

- 头文件：cvi_comm_aio.h，cvi_audio.h
- 库文件：libcvi_audio.a、libssp.so、libcvi_RES1.so

【注意事项】

- 启用声音质量增强功能前必须先设置相对应 Audio Output 通道的声音质量增强功能相关属性。
- 设置 Audio Output 的声音质量增强功能相关属性前，必须先使能对应的 Audio Output 通道。

【举例】

无。

10.3.3.11 CVI_AO_GetVqeAttr

【描述】

获取 Audio Output 的声音质量增强功能相关属性。

【语法】

```
CVI_S32 CVI_AO_GetVqeAttr(AUDIO_DEV AoDevId, AO_CHN AoChn, AO_VQE_CONFIG_S
↪*pstVqeConfig);
```

【参数】

参数名称	描述	输入/输出
AoDevId	音频设备号。设备号预设为 0，在客制化未要求扩充下，设置超过 2，返回错误。	输入
AoChn	音频输出通道号	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】

参数名称	描述
0	成功。
非 0	失败，其值为错误码 。

【需求】

- 头文件：cvi_comm_aio.h，cvi_audio.h
- 库文件：libcvi_audio.a、libssp.so、libcvi_RES1.so、libcvi_vqe.so

【注意事项】

- 获取声音质量增强功能相关属性前必须先设置相对应 Audio Output 通道的声音质量增强功能相关属性

【举例】

无。

10.3.3.12 CVI_AO_EnableVqe

【描述】

使能 Audio Output 的声音质量增强功能。

【语法】

```
CVI_S32 CVI_AO_EnableVqe(AUDIO_DEV AoDevId, AO_CHN AoChn);
```

【参数】

参数名称	描述	输入/输出
AoDevId	音频设备号。设备号预设为 0，在客制化未要求扩充下，设置超过 2，返回错误。	输入
AoChn	音频输出通道号。	输入

【返回值】

参数名称	描述
0	成功。
非 0	失败，其值为错误码 。

【需求】

- 头文件：cvi_comm_aio.h，cvi_audio.h
- 库文件：libcvi_audio.a、libssp.so、libcvi_RES1.so、libcvi_vqe.so

【注意事项】

- 启用声音质量增强功能前必须先启用相对应 Audio Output 通道。
- 禁用 Audio Output 通道后，如果重新启用 Audio Output 通道，并使用声音质量增强功能，需调用此接口重新启用声音质量增强功能。

【举例】

无。

10.3.3.13 CVI_VQE_PathSelect

【描述】

VQE 算法路径设置。

【语法】

```
CVI_S32 CVI_VQE_PathSelect(E_VQE_ALGO_PATH eVqePath);
```

【参数】

参数名称	描述	输入/输出
eVqePath	音频设备号。设备号预设为 0，在客制化未要求扩充下，设置超过 2，返回错误。	输入

【返回值】

参数名称	描述
0	成功。
非 0	失败，其值为错误码 。

【需求】

- 头文件：cvi_comm_aio.h，cvi_audio.h
- 库文件：libcvi_audio.so、libssp.so、libcvi_vqe.so

【注意事项】

此 API 仅在支持 IC 下可使用，使用时需在 ENABLE 前使能。

【举例】

无。

10.3.4 音频输出

音频输出（Audio Output）主要实现启用音频输出设备、发送音频帧到输出信道等功能。

下面列举 API 并附加细部内容:

- CVI_AO_SetPubAttr： 设置 Audio Output 设备属性。
- CVI_AO_GetPubAttr： 获取 Audio Output 设备属性。
- CVI_AO_Enable： 启用 Audio Output 设备。
- CVI_AO_Disable： 禁用 Audio Output 设备。
- CVI_AO_EnableChn： 启用 Audio Output 通道。
- CVI_AO_DisableChn： 禁用 Audio Output 通道。
- CVI_AO_SendFrame： 发送 Audio Output 音频帧。
- CVI_AO_EnableReSmp： 启用 Audio Output 重采样。
- CVI_AO_DisableReSmp： 禁用 Audio Output 重采样。
- CVI_AO_PauseChn： 暂停 Audio Output 通道。
- CVI_AO_ResumeChn： 恢复 Audio Output 通道。
- CVI_AO_ClearChnBuf： 清除 Audio Output 通道中当前的音频数据区块。
- CVI_AO_QueryChnStat： 查询 Audio Output 通道中当前的音频数据区块状态。
- CVI_AO_SetTrackMode： 设置 Audio Output 设备声道模式。
- CVI_AO_GetTrackMode： 获取 Audio Output 设备声道模式。
- CVI_AO_SetVolume： 设置 Audio Output 设备音量大小。
- CVI_AO_GetVolume： 获取 Audio Output 设备音量大小。
- CVI_AO_SetMute： 设置 Audio Output 设备静音状态。
- CVI_AO_GetMute： 获取 Audio Output 设备静音状态。
- CVI_AO_SaveFile： 开启音频输出保存文件功能。(目前不支持此功能，用户可透过 CVI_AO_SendFrame 前存盘);
- CVI_AO_ClrPubAttr： 清除 Audio Output 设备属性。

10.3.4.1 CVI_AO_SetPubAttr

【描述】

设置 Audio Output 设备属性

【语法】

```
CVI_S32 CVI_AO_SetPubAttr(AUDIO_DEV AoDevId,const AIO_ATTR_S *pstAttr);
```

【参数】

参数名称	描述	输入/输出
AoDevId	音频设备号。设备号预设为 0，在客制化未要求扩充下，设置超过 2，返回错误。	输入
pstAttr	音频输出设备属性。	输入

【返回值】

参数名称	描述
0	成功。
非 0	失败，其值为错误码 。

【需求】

- 头文件：cvi_comm_aio.h，cvi_audio.h
- 库文件：libcvi_audio.a

【注意事项】

- 在设置属性之前需要保证 Audio Output 处于禁用状态，如果处于启用状态则需要首先禁用 Audio Output 设备。
- Audio Output 必须和 DA 配合起来才能正常工作，用户必须清楚 DA 发送的数据格式和通道的关系才能从正确的通道发送数据。
- u32ClkSel 在 CV182x processor 下无需配置。
- Audio Output 设备主模式时，决定 Audio Output 设备输出时钟的关键配置项是采样率、采样精度以及通道数目，采样精度乘以通道数目即为 Audio Output 设备时序一次采样的位宽。
- CV182x 扩展标志对 Audio Output 设备无效，无需设定。
- Audio Output 设备属性结构体中其他项请参见 Audio Input 模块中相关接口的描述。

【举例】

无。

10.3.4.2 CVI_AO_GetPubAttr

【描述】

获取 Audio Output 设备属性。

【语法】

```
CVI_S32 CVI_AO_GetPubAttr(AUDIO_DEV AoDevId, AIO_ATTR_S *pstAttr);
```

【参数】

参数名称	描述	输入/输出
AoDevId	音频设备号。设备号预设为 0，在客制化未要求扩充下，设置超过 2，返回错误。	输入
pstAttr	音频输出设备属性指针。	输出

【返回值】

参数名称	描述
0	成功。
非 0	失败，其值为错误码 。

【需求】

- 头文件：cvi_comm_aio.h，cvi_audio.h
- 库文件：ibcvi_audio.a

【注意事项】

- 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。

【举例】

```
CVI_S32 s32ret;
AUDIO_DEV AoDevId = 0;
AIO_ATTR_S stAttr;

s32ret = CVI_AO_GetPubAttr(AoDevId, &stAttr);
if(s32ret != CVI_SUCCESS) {
printf("get ao %d attr err:0x%x\n", AoDevId,s32ret);
return s32ret; }
```

10.3.4.3 CVI_AO_Enable

【描述】

使能 Audio Output 设备。

【语法】

```
CVI_S32 CVI_AO_Enable(AUDIO_DEV AoDevId);
```

【参数】

参数名称	描述	输入/输出
AoDevId	音频设备号。设备号预设为 0，在客制化未要求扩充下，设置超过 2，返回错误。	输入

【返回值】

参数名称	描述
0	成功。
非 0	失败，其值为错误码 。

【需求】

- 头文件：cvi_comm_aio.h，cvi_audio.h
- 库文件：libcvi_audio.a

【注意事项】

- 要求在启用前配置 Audio Output 设备属性，否则会返回属性未配置的错误。
- 如果 Audio Output 设备已经启用，则直接返回成功。

【举例】

```
CVI_S32 i;
CVI_S32 s32Ret;

s32Ret = CVI_AO_SetPubAttr(AoDevId, pstAioAttr);
if (s32Ret != CVI_SUCCESS) {
        printf("%s: CVI_AO_SetPubAttr(%d) failed with %#x!\n", __func__,
               AoDevId, s32Ret);
        return CVI_FAILURE;
}

s32Ret = CVI_AO_Enable(AoDevId);
if (s32Ret != CVI_SUCCESS) {
        printf("%s: CVI_AO_Enable(%d) failed with %#x!\n", __func__, AoDevId,
               s32Ret);
        return CVI_FAILURE;
}

for (i = 0; i < s32AoChnCnt; i++) {
        s32Ret = CVI_AO_EnableChn(AoDevId, i / (pstAioAttr->enSoundmode + 1));
        if (s32Ret != CVI_SUCCESS) {
                printf("%s: CVI_AO_EnableChn(%d) failed with %#x!\n", __func__, i,
                       s32Ret);
                return CVI_FAILURE;
        }

        if (bResampleEn == CVI_TRUE) {
                s32Ret = CVI_AO_DisableReSmp(AoDevId, i);
                s32Ret |= CVI_AO_EnableReSmp(AoDevId, i, enInSampleRate);
                if (s32Ret != CVI_SUCCESS) {
                        printf("%s: CVI_AO_EnableReSmp(%d,%d) failed with %#x!\n", __func__,
                               AoDevId, i, s32Ret);
                        return CVI_FAILURE;
                }
        }

}

s32Ret = CVI_AO_EnableChn(AoDevId, AO_SYSCHN_CHNID);
```

（下页继续）

(续上页)

```
if (s32Ret != CVI_SUCCESS) {
        printf("%s: CVI_AO_EnableChn(%d) failed with %#x!\n", __func__, i,
               s32Ret);
        return CVI_FAILURE;
}
```

10.3.4.4 CVI_AO_Disable

【描述】

停用 Audio Output 设备。

【语法】

```
CVI_S32 CVI_AO_Disable(AUDIO_DEV AoDevId);
```

【参数】

参数名称	描述	输入/输出
AoDevId	音频设备号。设备号预设为 0，在客制化未要求扩充下，设置超过 2，返回错误。	输入

【返回值】

参数名称	描述
0	成功。
非 0	失败，其值为错误码 。

【需求】

- 头文件：cvi_comm_aio.h，cvi_audio.h
- 库文件：libcvi_audio.a

【注意事项】

禁用 Audio Output 设备前必须先禁用设备下所有 Audio Output 通道。

【举例】

无。

10.3.4.5 CVI_AO_EnableChn

【描述】

启用 Audio Output 通道。

【语法】

```
CVI_S32 CVI_AO_EnableChn(AUDIO_DEV AoDevId, AO_CHN AoChn);
```

【参数】

参数名称	描述	输入/输出
AoDevId	音频设备号。设备号预设为 0，在客制化未要求扩充下，设置超过 2，返回错误。	输入
AoChn	音频输出通道号。支持的通道范围由 Audio Output 描述入设备属性中的最大通道个数 u32ChnCnt 决定与声道模式 enSoundmode 决定。	输入

【返回值】

参数名称	描述
0	成功。
非 0	失败，其值为错误码 。

【需求】

- 头文件：cvi_comm_aio.h，cvi_audio.h
- 库文件：libcvi_audio.a

【注意事项】

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

【举例】

无。

10.3.4.6 CVI_AO_DisableChn

【描述】

禁用 Audio Output 通道。

【语法】

```
CVI_S32 CVI_AO_DisableChn(AUDIO_DEV AoDevId, AO_CHN AoChn);
```

【参数】

参数名称	描述	输入/输出
AoDevId	音频设备号。设备号预设为 0，在客制化未要求扩充下，设置超过 2，返回错误。	输入
AoChn	音频输出通道号。支持的通道范围由 Audio Output 描述入设备属性中的最大通道个数 u32ChnCnt 决定与声道模式 enSoundmode 决定。	输入

【返回值】

参数名称	描述
0	成功。
非 0	失败，其值为错误码 。

【需求】

- 头文件：cvi_comm_aio.h，cvi_audio.h
- 库文件：libcvi_audio.a

【注意事项】

无。

【举例】

无。

10.3.4.7 CVI_AO_SendFrame

【描述】

发送 Audio Output 音框。

【语法】

```
CVI_S32 _AO_SendFrame(AUDIO_DEV AoDevId, AO_CHN AoChn, const AUDIO_FRAME_S␣
↪*pstData, CVI_S32 s32MilliSec);
```

【参数】

参数名称	描述	输入/输出
AoDevId	音频设备号。设备号预设为 0，在客制化未要求扩充下，设置超过 2，返回错误。	输入
AoChn	音频输出通道号。支持的通道范围由 Audio Output 描述入设备属性中的最大通道个数 u32ChnCnt 决定与声道模式 enSoundmode 决定。	输入
pstData	音频帧结构体指针。	输入
s32MilliSec	发送数据的超时时间 -1 表示阻塞模式； 0 描述入表示非阻塞模式； >0 表示阻塞 s32MilliSec 毫秒，超时则报错返回。	输入

【返回值】

参数名称	描述
0	成功。
非 0	失败，其值为错误码 。

【需求】

- 头文件：cvi_comm_aio.h，cvi_audio.h
- 库文件：libcvi_audio.a

【注意事项】

- 该接口用于用户主动发送音频帧至 Audio Output 输出。
- Audio Output 通道已经通过系统绑定 CVI_SYS_Bind）接口与 Audio Input 或 ADEC 绑定，不需要也不建议调此接口。
- 调用该接口发送音频帧到 Audio Output 输出时，必须先使能对应的 Audio Output 通道。

【举例】

无。

10.3.4.8 CVI_AO_EnableReSmp

【描述】

启用 Audio Output 重采样。

【语法】

```
CVI_S32 CVI_AO_EnableReSmp(AUDIO_DEV AoDevId, AO_CHN AoChn, AUDIO_SAMPLE_
↪RATE_E enInSampleRate);
```

【参数】

参数名称	描述	输入/输出
AoDevId	音频设备号。设备号预设为 0，在客制化未要求扩充下，设置超过 2，返回错误。	输入
AoChn	音频输出通道号。支持的通道范围由 Audio Output 描述入设备属性中的最大通道个数 u32ChnCnt 决定与声道模式 enSoundmode 决定。	输入
enInSampleRate	音频重采样的输入采样率。	输入

【返回值】

参数名称	描述
0	成功。
非 0	失败，其值为错误码 。

【需求】

- 头文件：cvi_comm_aio.h，cvi_audio.h

- 库文件：libcvi_audio.a

【注意事项】

- 在使用 CVI_AUD_SYS_Bind 的状态下，不支持此 API
- 应该在启用 Audio Output 通道之后，绑定 Audio Output 通道之前，调用此接口启用重采样功能。
- 允许重复启用重采样功能，但必须保证后配置的重采样输入采样率与之前配置的重采样输入采样率一样。
- 在禁用 Audio Output 通道后，如果重新启用 Audio Output 通道，并使用重采样功能，需调用此接口重新启用重采样。
- Audio Output 重采样的输入采样率必须与 Audio Output 设备属性配置的采样率不相同。

【举例】

无。

10.3.4.9 CVI_AO_DisableReSmp

【描述】

禁用 Audio Output 重采样。

【语法】

```
CVI_S32 CVI_AO_DisableReSmp(AUDIO_DEV AoDevId, AO_CHN AoChn);
```

【参数】

参数名称	描述	输入/输出
AoDevId	音频设备号。设备号预设为 0，在客制化未要求扩充下，设置超过 2，返回错误。	输入
AoChn	音频输出通道号。支持的通道范围由 Audio Output 描述入设备属性中的最大通道个数 u32ChnCnt 决定与声道模式 enSoundmode 决定。	输入

【返回值】

参数名称	描述
0	成功。
非 0	失败，其值为错误码 。

【需求】

- 头文件：cvi_comm_aio.h，cvi_audio.h
- 库文件：libcvi_audio.a

【注意事项】

不再使用 Audio Output 重采样功能的话，应该调用此接口将其禁用

【举例】

无。

10.3.4.10 CVI_AO_PauseChn

【描述】

暂停 Audio Output 通道。

【语法】

```
CVI_S32 CVI_AO_PauseChn(AUDIO_DEV AoDevId, AO_CHN AoChn);
```

【参数】

参数名称	描述	输入/输出
AoDevId	音频设备号。设备号预设为 0，在客制化未要求扩充下，设置超过 2，返回错误。	输入
AoChn	音频输出通道号。支持的通道范围由 Audio Output 设备属性中的最大通道个数 u32ChnCnt 决定与声道模式 enSoundmode 决定。	输入

【返回值】

参数名称	描述
0	成功。
非 0	失败，其值为错误码 。

【需求】

- 头文件：cvi_comm_aio.h，cvi_audio.h
- 库文件：libcvi_audio.a

【注意事项】

Audio Output 通道暂停后，如果绑定的 ADEC 信道继续向此信道发送音频帧数据，发送的音频帧数据将会被阻塞；而如果绑定的 Audio Input 信道继续向此信道发送音频帧数据，在通道缓冲未满的情况下则将音频帧放入缓冲区，在满的情况下则将音频帧丢弃。

Audio Output 通道为禁用状态时，不允许调用此接口暂停 Audio Output 通道。

【举例】

无。

10.3.4.11 CVI_AO_ResumeChn

【描述】

恢复 Audio Output 通道。

【语法】

```
CVI_S32 CVI_AO_ResumeChn(AUDIO_DEV AoDevId, AO_CHN AoChn);
```

【参数】

参数名称	描述	输入/输出
AoDevId	音频设备号。设备号预设为 0，在客制化未要求扩充下，设置超过 2，返回错误。	输入
AoChn	音频输出通道号。支持的通道范围由 Audio Output 设备属性中的最大通道个数 u32ChnCnt 决定与声道模式 enSoundmode 决定。	输入

【返回值】

参数名称	描述
0	成功。
非 0	失败，其值为错误码 。

【需求】

- 头文件：cvi_comm_aio.h，cvi_audio.h
- 库文件：libcvi_audio.a

【注意事项】

- Audio Output 通道暂停后可以通过调用此接口重新恢复
- Audio Output 通道为暂停状态或使能状态下，调用此接口返回成功；否则调用将返回错误

【举例】

无。

10.3.4.12 CVI_AO_ClearChnBuf

【描述】

清除 Audio Output 通道中当前的音频数据区块。

【语法】

```
CVI_S32 CVI_AO_ClearChnBuf(AUDIO_DEV AoDevId, AO_CHN AoChn);
```

【参数】

参数名称	描述	输入/输出
AoDevId	音频设备号。设备号预设为 0，在客制化未要求扩充下，设置超过 2，返回错误。	输入
AoChn	音频输出通道号。支持的通道范围由 Audio Output 设备属性中的最大通道个数 u32ChnCnt 决定与声道模式 enSoundmode 决定。	输入

【返回值】

参数名称	描述
0	成功。
非 0	失败，其值为错误码 。

【需求】

- 头文件：cvi_comm_aio.h，cvi_audio.h
- 库文件：libcvi_audio.a

【注意事项】

- 在 Audio Output 通道成功启用后再调用此接口。
- 为完全清除解码回放通路上所有区块数据，此接口还应该与 CVI_ADEC_ClearChnBuf 接口配合使用。
- 不推荐使用

【举例】

无。

10.3.4.13 CVI_AO_QueryChnStat

【描述】

查询 Audio Output 通道中当前的音频数据区块状态。

【语法】

```
CVI_S32 CVI_AO_QueryChnStat(AUDIO_DEV AoDevId, AO_CHN AoChn, AO_CHN_STATE_
↪S *pstStatus);
```

【参数】

参数名称	描述	输入/输出
AoDevId	音频设备号。设备号预设为 0，在客制化未要求扩充下，设置超过 2，返回错误。	输入
AoChn	音频输出通道号。支持的通道范围由 Audio Output 设备属性中的最大通道个数 u32ChnCnt 决定与声道模式 enSoundmode 决定。	输入
pstStatus	区块状态结构体指针。	输出

【返回值】

参数名称	描述
0	成功。
非 0	失败，其值为错误码 。

【需求】

- 头文件：cvi_comm_aio.h，cvi_audio.h
- 库文件：libcvi_audio.a

【注意事项】

在 Audio Output 通道成功启用后再调用此接口。

【举例】

无。

10.3.4.14 CVI_AO_SetTrackMode

【描述】

设置 Audio Output 设备声道模式。

【语法】

```
CVI_S32 CVI_AO_SetTrackMode(AUDIO_DEV AoDevId, AUDIO_TRACK_MODE_E
↪enTrackMode);
```

【参数】

参数名称	描述	输入/输出
AoDevId	音频设备号。设备号预设为 0，在客制化未要求扩充下，设置超过 2，返回错误。	输入
enTrackMode	音频设备声道模式。	输入

【返回值】

参数名称	描述
0	成功。
非 0	失败，其值为错误码 。

【需求】

- 头文件：cvi_comm_aio.h，cvi_audio.h
- 库文件：libcvi_audio.a

【注意事项】

- 在 Audio Output 设备成功启用后再调用此接口。
- Audio Output 设备工作在 I2S 模式时，支持设置声道模式，PCM 模式下不支持。

【举例】

```
CVO_S32 s32Ret;
AUDIO_DEV AoDev = 0;
AUDIO_TRACK_MODE_E enTrackMode = AUDIO_TRACK_NORMAL;
AUDIO_TRACK_MODE_E temp;

s32Ret = CVI_AO_SetTrackMode(AoDev, enTrackMode);

if (CVI_SUCCESS != s32Ret) {
printf("Ao set track mode failure! AoDev: %d, enTrackMode: %d, s32Ret: 0x%x.\n", AoDev,
↪enTrackMode, s32Ret);
return s32Ret;
}

s32Ret = CVI_AO_GetTrackMode(AoDev, &temp);
if (s32Ret!=CVI_SUCCESS) {
printf("Ao get track mode failure! AoDev: %d, s32Ret: 0x%x.\n", AoDev, s32Ret);
return s32Ret;
}
```

10.3.4.15 CVI_AO_GetTrackMode

【描述】

获取 Audio Output 设备声道模式。

【语法】

```
CVI_S32 CVI_AO_GetTrackMode(AUDIO_DEV AoDevId, AUDIO_TRACK_MODE_E
↪*penTrackMode);
```

【参数】

参数名称	描述	输入/输出
AoDevId	音频设备号。设备号预设为 0，在客制化未要求扩充下，设置超过 2，返回错误。	输入
enTrackMode	音频设备声道模式。	输出

【返回值】

参数名称	描述
0	成功。
非 0	失败，其值为错误码 。

【需求】

- 头文件：cvi_comm_aio.h，cvi_audio.h
- 库文件：libcvi_audio.a

【注意事项】

- 在 Audio Output 设备成功启用后再调用此接口。

· Audio Output 设备工作在 I2S 模式时，支持获取声道模式，PCM 模式下不支持。

【举例】

无。

10.3.4.16 CVI_AO_SetVolume

【描述】

设置 Audio Output 设备音量大小。

【语法】

```
CVI_S32 CVI_AO_SetVolume(AUDIO_DEV AoDevId, CVI_S32 s32VolumeDb);
```

【参数】

参数名称	描述	输入/输出
AoDevId	音频设备号。设备号预设为 0，在客制化未要求扩充下，设置超过 2，返回错误。	输入
s32VolumeDb	音量范围为 [32~0]，分别对应增益 [0d -22.5dB]，每一阶降 1.5dB。	输入

【返回值】

参数名称	描述
0	成功。
非 0	失败，其值为错误码 。

【需求】

· 头文件：cvi_comm_aio.h，cvi_audio.h

· 库文件：libcvi_audio.a

【注意事项】

· 在 Audio Output 设备成功启用后再调用此接口。

【举例】

无。

10.3.4.17 CVI_AO_GetVolume

【描述】

获取 Audio Output 设备音量大小。

【语法】

```
CVI_S32 CVI_AO_GetVolume(AUDIO_DEV AoDevId, CVI_S32 *ps32VolumeDb);
```

【参数】

参数名称	描述	输入/输出
AoDevId	音频设备号。设备号预设为 0，在客制化未要求扩充下，设置超过 2，返回错误。	输入
ps32VolumeDb	音频设备音量大小指针。音量范围为 [32~0]，分别对应增益 [0dB ~ -22.5dB]，每一阶降 1.5dB。	输出

【返回值】

参数名称	描述
0	成功。
非 0	失败，其值为错误码 。

【需求】

- 头文件：cvi_comm_aio.h，cvi_audio.h
- 库文件：libcvi_audio.a

【注意事项】

在 Audio Output 设备成功启用后再调用此接口。

【举例】

无。

10.3.4.18 CVI_AO_SetMute

【描述】

设置 Audio Output 设备静音状态。

【语法】

```
CVI_S32 CVI_AO_SetMute(AUDIO_DEV AoDevId, CVI_BOOL bEnable, const AUDIO_FADE_S
↪*pstFade);
```

【参数】

参数名称	描述	输入/输出
AoDevId	音频设备号。设备号预设为 0，在客制化未要求扩充下，设置超过 2，返回错误。	输入
bEnable	音频设备是否启用静音。CVI_TRUE：启用静音功能；CVI_FALSE：关闭静音功能	输入
pstFade	淡入淡出结构体指针。	输入

【返回值】

参数名称	描述
0	成功。
非 0	失败，其值为错误码 。

【需求】

- 头文件：cvi_comm_aio.h，cvi_audio.h
- 库文件：libcvi_audio.a

【注意事项】

- 在 Audio Output 设备成功启用后再调用此接口。

【举例】

无。

10.3.4.19 CVI_AO_GetMute

【描述】

获取 Audio Output 设备静音状态。

【语法】

```
CVI_S32 CVI_AO_GetMute(AUDIO_DEV AoDevId, CVI_BOOL *pbEnable, AUDIO_FADE_S
↪*pstFade);
```

【参数】

参数名称	描述	输入/输出
AoDevId	音频设备号。设备号预设为 0，在客制化未要求扩充下，设置超过 2，返回错误。	输入
bEnable	音频设备静音状态指针。	输出
pstFade	淡入淡出结构体指针。	输出

【返回值】

参数名称	描述
0	成功。
非 0	失败，其值为错误码 。

【需求】

- 头文件：cvi_comm_aio.h，cvi_audio.h
- 库文件：libcvi_audio.a

【注意事项】

- 在 Audio Output 设备成功启用后再调用此接口。

【举例】

无。

10.3.4.20 CVI_AO_SaveFile

【描述】

目前不支持此功能，用户可透过 CVI_AO_SendFrame 前存盘。

【语法】

```
CVI_S32 CVI_AO_SaveFile(AUDIO_DEV AoDevId, AO_CHN AoChn, AUDIO_SAVE_FILE_
↪INFO_S* pstSaveFileInfo);
```

【参数】

【返回值】

参数名称	描述
0	成功。
非 0	失败，其值为错误码 。

【需求】

【注意事项】

【举例】

无。

10.3.4.21 CVI_AO_ClrPubAttr

【描述】

清除 Audio Output 设备属性。

【语法】

```
CVI_S32 CVI_AO_ClrPubAttr(AUDIO_DEV AoDevId);
```

【参数】

参数名称	描述	输入/输出
AoDevId	音频设备号。设备号预设为 0，在客制化未要求扩充下，设置超过 2，返回错误。	输入

【返回值】

参数名称	描述
0	成功。
非 0	失败，其值为错误码 。

【需求】

- 头文件：cvi_comm_aio.h，cvi_audio.h
- 库文件：libcvi_audio.a

【注意事项】

- 清除设备属性前，需要先停止设备。
- Audio Input 设备及 Audio Output 设备时钟无需设定。

【举例】

无。

10.3.5 音频编码

音频编码主要专责音框数据转换，Cvitek 支持 G.711、G.726、LDPCM 相关编码，编码后的音框数据较小，但语音编码属于有损压缩 (lossy compression)，音质会有所差异，不同编码使用的 bit rate/sample rate 亦有所不同，用户需要知道该编码协议的规范并设定对应 API，否则函式将回报错误。

Codec	Sampling Rate(KHZ)	Bandwidth(kbps)	Nominal Bandwidth(kbps)	License
G.711*	8	64	87.2	Open Source
G.726	8	16/24/32/40	47.2/55.2	Open Source

*G.711 包含 a-law/mu-law

AENC 模块提供以下 API:

- CVI_AENC_CreateChn：创建音频编码通道。
- CVI_AENC_DestroyChn：销毁音频编码通道。
- CVI_AENC_SendFrame：发送音频编码音频帧
- CVI_AENC_GetStream：获取音频编码码流。
- CVI_AENC_ReleaseStream：释放音频编码码流。
- CVI_AENC_SaveFile：开启音频编码之前信道存文件功能。
- CVI_AENC_QueryFileStatus：查询音频编码通道是否处于存文件的状态。
- CVI_AENC_GetStreamBufInfo：获取音频码流 buffer 相关信息。
- CVI_AENC_SetMute：设置 AENC 静音状态。
- CVI_ADEC_GetMute：获取 AENC 静音状态。

10.3.5.1 CVI_AENC_CreateChn

【描述】

创建音频编码通道。

【语法】

```
CVI_S32 CVI_AENC_CreateChn(AENC_CHN AeChn, const AENC_CHN_ATTR_S *pstAttr);
```

【参数】

参数名称	描述	输入/输出
AeChn	编码设备信道号。取值范围：[0, AENC_MAX_CHN_NUM]	输入
pstAttr	编码通道属性指针	输入

【返回值】

参数名称	描述
0	成功。
非 0	失败，其值为错误码 。

【需求】

- 头文件：cvi_comm_aio.h，cvi_audio.h，cvi_comm_aenc.h
- 库文件：libcvi_audio.a，libcvi_VoiceEngine.so，libcvi_RES1.so

【注意事项】

- 目前支持 G711、G726、ADPCM
- 支持输入 pcm 音框，音讯格式 (format) 支持 16 位小端序 (S16LE) 其余不支持，使用时请同步将 Audio Input attribute/AENC attribute 做对应设置，如要在 VQE 之后使用，请将音框长度设为 160 的倍数，以符合 Cvitek VQE 规范。
- 音频编码的部分属性需要与输入的音频数据属性相匹配，例如采样率、帧长（每帧采样点数目）等
- buffer 大小以帧为单位，取值范围是 [2, MAX_BUFFERING_DEPTH]，建议配置为 10 以上，过小的 buffer 配置可能导致丢帧等异常。

【举例】

无。

10.3.5.2 CVI_AENC_DestroyChn

【描述】

销毁音频编码通。

【语法】

```
CVI_S32 CVI_AENC_DestroyChn(AENC_CHN AeChn);
```

【参数】

参数名称	描述	输入/输出
AeChn	编码设备信道号。取值范围：[0, AENC_MAX_CHN_NUM]	输入

【返回值】

参数名称	描述
0	成功。
非 0	失败，其值为错误码 。

【需求】

- 头文件：cvi_comm_aio.h，cvi_audio.h、cvi_comm_aenc.h
- 库文件：libcvi_audio.a，libcvi_VoiceEngine.so，libcvi_RES1.so

【注意事项】

- 通道如未创建，则调用本函式无效。
- 如果正在获取/释放码流或者发送帧时销毁该通道，则会返回失败，用户同步处理时需要注意。

【举例】

无。

10.3.5.3 CVI_AENC_SendFrame

【描述】

发送音频编码音频帧。

【语法】

```
CVI_S32 CVI_AENC_SendFrame(AENC_CHN AeChn, const AUDIO_FRAME_S *pstFrm, const␣
↪AEC_FRAME_S *pstAecFrm);
```

【参数】

参数名称	描述	输入/输出
AeChn	编码设备信道号。取值范围：[0, AENC_MAX_CHN_NUM]	输入
pstFrm	音频帧结构体指针。	输入
pstAecFrm	回声抵消参考帧结构体指针。	输入

【返回值】

参数名称	描述
0	成功。
非 0	失败，其值为错误码 。

【需求】

- 头文件：cvi_comm_aio.h，cvi_audio.h、cvi_comm_aenc.h
- 库文件：libcvi_audio.a，libcvi_VoiceEngine.so，libcvi_RES1.so

【注意事项】

- 通道如未创建，呼叫本函式无效。

- 如果不需要回声抵消，pstAecFrm 可置为 NULL。
- 该接口用于用户主动发送音频帧进行编码，如果 AENC 通道已经通过系统绑定接口与 Audio Input 绑定，不需要也不建议调此接口。

【举例】

```
s32Ret = CVI_AI_GetChnParam(AiDev, AiChn, &stAiChnPara);
if (s32Ret != CVI_SUCCESS) {
        printf("%s: Get ai chn param failed\n", __func__);
        return NULL;
}

stAiChnPara.u32UsrFrmDepth = 10;
s32Ret = CVI_AI_SetChnParam( AiDev,  AiChn, &stAiChnPara);
if (s32Ret != CVI_SUCCESS) {
        printf("%s: set ai chn param failed\n", __func__);
        return NULL;
}

while ( 1) {
        /* get frame from ai chn */
        memset(&stAecFrm, 0, sizeof(AEC_FRAME_S));
        s32Ret = CVI_AI_GetFrame( AiDev,  AiChn, &stFrame,
                                  &stAecFrm, CVI_FALSE);
        if (s32Ret != CVI_SUCCESS) {
                printf("CVI_AI_GetFrame none!!\n");
                continue;
        }
        /* send frame to encoder */
        if ( bSendAenc == CVI_TRUE) {
                s32Ret = CVI_AENC_SendFrame( AencChn, &stFrame, &stAecFrm);
                if (s32Ret != CVI_SUCCESS) {
                        printf("%s: CVI_AENC_SendFrame(%d), failed with %#x!\n",
                               __func__,  AencChn, s32Ret);
                        bStart = CVI_FALSE;
                        return NULL;
                }
        }
        /* send frame to ao */
        /* If owner toggle bSendAenc, do not toggle bSendAo */
        /* You cannot send encode frame to CVI_AO_SendFrame */
        /* It cannot play out encode frame by only AO_SendFrame*/
        if ( bSendAo == CVI_TRUE) {
                s32Ret = CVI_AO_SendFrame( AoDev,  AoChn, &stFrame, 1000);
                if (s32Ret != CVI_SUCCESS) {
                        printf("%s: CVI_AO_SendFrame(%d, %d), failed with %#x!\n",
                               __func__,  AoDev,  AoChn, s32Ret);
                        bStart = CVI_FALSE;
                        return NULL;
                }
        }

        /* finally you must release the stream */
        s32Ret = CVI_AI_ReleaseFrame( AiDev,  AiChn, &stFrame,
                                      &stAecFrm);
```

(下页继续)

(续上页)

```
        if (s32Ret != CVI_SUCCESS) {
                printf("%s: CVI_AI_ReleaseFrame(%d, %d), failed with %#x!\n",
                        __func__, AiDev, AiChn, s32Ret);
                bStart = CVI_FALSE;
                return NULL;
        }
}
```

10.3.5.4 CVI_AENC_GetStream

【描述】

获取编码后码流。

【语法】

```
CVI_S32 CVI_AENC_GetStream(AENC_CHN AeChn, AUDIO_STREAM_S *pstStream, CVI_S32
↪s32MilliSec);
```

【参数】

参数名称	描述	输入/输出
AeChn	编码设备信道号。取值范围：[0, AENC_MAX_CHN_NUM]	输入
pstStream	获取的音频码流。	输出
s32MilliSec	获取数据的超时时间 -1 表示阻塞模式，无数据时一直等待； 0 表示非阻塞模式，无数据时则报错返回； >0 表示阻塞 s32MilliSec 毫秒，超时则报错返回。	输入

【返回值】

参数名称	描述
0	成功。
非 0	失败，其值为错误码 。

【需求】

- 头文件：cvi_comm_aio.h，cvi_audio.h、cvi_comm_aenc.h
- 库文件：libcvi_audio.a，libcvi_VoiceEngine.so，libcvi_RES1.so

【注意事项】

- 必须创建通道后才可能获取码流，否则直接返回失败，如果在获取码流过程中销毁通道则会立刻返回失败。

- s32MilliSec 的值必须大于等于-1，等于-1 时采用阻塞模式获取数据，等于 0 时采用非阻塞模式获取数据，大于 0 时，阻塞 s32MilliSec 毫秒后，没有数据则返回超时并报。

【举例】

无。

10.3.5.5 CVI_AENC_ReleaseStream

【描述】

释放从音频编码通道获取的码流。

【语法】

```
CVI_S32 CVI_AENC_ReleaseStream(AENC_CHN AeChn, const AUDIO_STREAM_S *pstStream);
```

【参数】

参数名称	描述	输入/输出
AeChn	编码设备信道号。取值范围：[0, AENC_MAX_CHN_NUM]	输入
pstStream	获取的音频码流之指针。	输入

【返回值】

参数名称	描述
0	成功。
非 0	失败，其值为错误码 。

【需求】

- 头文件：cvi_comm_aio.h，cvi_audio.h、cvi_comm_aenc.h
- 库文件：libcvi_audio.a，libcvi_VoiceEngine.so，libcvi_RES1.so

【注意事项】

- 码流最好能够在使用完之后立即释放，如果不及时释放，会导致编码过程阻塞。
- 释放的码流必须是从该通道获取的码流，不得对码流信息结构体进行任何修改，否则会导致码流不能释放，使此码流 buffer 丢失，甚至导致程序异常。
- 释放码流时必须保证通道已经被创建，否则直接返回失败，如果在释放码流过程中销毁通道则会立刻返回失败。

【举例】

无。

10.3.5.6 CVI_AENC_SaveFile

【描述】

开启音频编码之前信道存文件功能。

【语法】

```
CVI_S32 CVI_AENC_SaveFile(AENC_CHN AeChn, const AUDIO_SAVE_FILE_INFO_S
↪*pstSaveFileInfo);
```

【参数】

参数名称	描述	输入/输出
AeChn	音频编码通道号。取值范围：[0, AENC_MAX_CHN_NUM]。	输入
pstSaveFileInfo	音频保存文件属性结构体指针。	输入

【返回值】

参数名称	描述
0	成功。
非 0	失败，其值为错误码 。

【需求】

- 头文件：cvi_comm_aio.h，cvi_audio.h、cvi_comm_aenc.h
- 库文件：libcvi_audio.a，libcvi_VoiceEngine.so，libcvi_RES1.so

【注意事项】

- Cvitek 不支援此 API。
- 请使用CVI_AI_SaveFile 。

【举例】

无。

10.3.5.7 CVI_AENC_QueryFileStatus

【描述】

查询频编码通道是否处于存文件的状。

【语法】

```
CVI_S32 CVI_AENC_QueryFileStatus(AENC_CHN AeChn, AUDIO_FILE_STATUS_S*
↪pstFileStatus);
```

【参数】

参数名称	描述	输入/输出
AeChn	音频编码通道号。取值范围：[0, AENC_MAX_CHN_NUM]。	输入
pstFileStatus	状态属性结构体指针。	输出

【返回值】

参数名称	描述
0	成功。
非 0	失败，其值为错误码 。

【需求】

- 头文件：cvi_comm_aio.h，cvi_audio.h、cvi_comm_aenc.h
- 库文件：libcvi_audio.a，libcvi_VoiceEngine.so，libcvi_RES1.so

【注意事项】

- Cvitek 不支援此 API。
- 请使用CVI_AI_SaveFile / CVI_AI_QueryFileStatus

【举例】

无。

10.3.5.8 CVI_AENC_GetStreamBufInfo

【描述】

获取音频码流 buffer 相关信息。

【语法】

```
CVI_S32 CVI_AENC_GetStreamBufInfo(AENC_CHN AeChn, CVI_U32 *pu32PhysAddr, CVI_U32
↪*pu32Size);
```

【参数】

参数名称	描述	输入/输出
AeChn	音频编码通道号。取值范围：[0, AENC_MAX_CHN_NUM]。	输入
pu32PhysAddr	音频码流 buffer 的物理地址。	输出
pu32Size	音频码流 buffer 的长度，以 byte 为单位	输出

【返回值】

参数名称	描述
0	成功。
非 0	失败，其值为错误码 。

【需求】

- 头文件：cvi_comm_aio.h，cvi_audio.h、cvi_comm_aenc.h
- 库文件：libcvi_audio.a，libcvi_VoiceEngine.so，libcvi_RES1.so

【注意事项】

无。

【举例】

无。

10.3.5.9 CVI_AENC_SetMute

【描述】

设置 AENC 静音状态

【语法】

```
CVI_AENC_SetMute (AENC_CHN AeChn, CVI_BOOL bEnable);
```

【参数】

参数名称	描述	输入/输出
AeChn	编码设备信道号。取值范围：[0, AENC_MAX_CHN_NUM]	输入
bEnable	音频设备是否启用静音。CVI_TRUE：启用静音功能；CVI_FALSE：关闭静音功能	输入

【返回值】

参数名称	描述
0	成功。
非 0	失败，其值为错误码 。

【需求】

- 头文件：cvi_audio.h
- 库文件：libcvi_audio.a

【注意事项】

- 在 AENC 创建设备信号通道后才能使用

【举例】

无。

10.3.5.10 CVI_AENC_GetMute

【描述】

得到 AENC 设备静音状态

【语法】

```
CVI_AENC_GetMute(AENC_CHN AeChn, CVI_BOOL *pbEnable);
```

【参数】

参数名称	描述	输入/输出
AeChn	编码设备信道号。取值范围：[0, AENC_MAX_CHN_NUM]	输入
pbEnable	音频设备禁音状态指针	输入

【返回值】

参数名称	描述
0	成功。
非 0	失败，其值为错误码 。

【需求】

- 头文件：cvi_audio.h
- 库文件：libcvi_audio.a

【注意事项】

- 在 AENC 创建设备信号通道后才能使用

【举例】

无。

10.3.6 音频解码

音频解码主要实现创建解码通道、发送音频码流解码及获取解码后音频帧等功能。

下面列出支持的 API 及细部内容:

- CVI_ADEC_CreateChn： 创建音频解码通道。
- CVI_ADEC_DestroyChn： 销毁音频解码通道。
- CVI_ADEC_SendStream： 发送音频码流到音频解码通道。
- CVI_ADEC_ClearChnBuf： 清除 ADEC 通道中当前的音频数据区块。
- CVI_ADEC_GetFrame： 获取音频解码帧数据。
- CVI_ADEC_ReleaseFrame： 释放音频解码帧数据。
- CVI_ADEC_SendEndOfStream： 向解码器发送码流结束标识符，并清除码流 buffer。

10.3.6.1 CVI_ADEC_CreateChn

【描述】

创建音频解码通道。

【语法】

```
CVI_S32 CVI_ADEC_CreateChn(ADEC_CHN AdChn, const ADEC_CHN_ATTR_S *pstAttr);
```

【参数】

参数名称	描述	输入/输出
AdChn	通道号。取值范围：[0, ADEC_MAX_CHN_NUM]。	输入
pstAttr	通道属性指针。	输入

【返回值】

参数名称	描述
0	成功。
非 0	失败，其值为错误码 。

【需求】

- 头文件：cvi_comm_aio.h，cvi_audio.h、cvi_comm_adec.h
- 库文件：libcvi_audio.a，libcvi_VoiceEngine.so，libcvi_RES1.so

【注意事项】

- 目前支持 G711、G726、ADPCM。
- 音频解码的部分属性需要与输出设备属性相匹配，例如采样率、帧长（每帧采样点数目）等

【举例】

无。

10.3.6.2 CVI_ADEC_DestroyChn

【描述】

销毁音频解码通道。

【语法】

```
CVI_S32 CVI_ADEC_DestroyChn(ADEC_CHN AdChn);
```

【参数】

参数名称	描述	输入/输出
AdChn	通道号。取值范围：[0, ADEC_MAX_CHN_NUM]。	输入

【返回值】

参数名称	描述
0	成功。
非 0	失败，其值为错误码 。

【需求】

- 头文件：cvi_comm_aio.h，cvi_audio.h、cvi_comm_adec.h
- 库文件：libcvi_audio.a，libcvi_VoiceEngine.so，libcvi_RES1.so

【注意事项】

- 通道未创建的情况下调用此接口会返回成功。
- 如果正在获取/释放码流或者发送帧，销毁该通道则这些操作都会立即返回失败。

【举例】

无。

10.3.6.3 CVI_ADEC_SendStream

【描述】

向音频解码通道发送码流。

【语法】

```
CVI_S32 CVI_ADEC_SendStream(ADEC_CHN AdChn, const AUDIO_STREAM_S *pstStream,␣
↪CVI_BOOL bBlock);
```

【参数】

参数名称	描述	输入/输出
AdChn	通道号。取值范围：[0, ADEC_MAX_CHN_NUM]。	输入
pstStream	音频码流	输入
bBlock	CVI_TRUE：阻塞。CVI_FALSE：非阻塞。	输入

【返回值】

参数名称	描述
0	成功。
非 0	失败，其值为错误码 。

【需求】

- 头文件：cvi_comm_aio.h，cvi_audio.h、cvi_comm_adec.h
- 库文件：libcvi_audio.a，libcvi_VoiceEngine.so，libcvi_RES1.so

【注意事项】

- stream 方式用于不确定码流包为一帧 (大于等于一帧) 的情况下，效率较低，且可能会有延迟
- 发送数据时必须保证通道已经被创建，否则直接返回失败，如果在送数据过程中销毁通道则会立刻返回失败。
- 确保发送给 ADEC 通道的码流数据的正确性，否则可能引起解码器异常退出。

【举例】

无。

10.3.6.4 CVI_ADEC_ClearChnBuf

【描述】

清除 ADEC 通道中当前的音频数据区块。

【语法】

```
CVI_S32 CVI_ADEC_ClearChnBuf(ADEC_CHN AdChn);
```

【参数】

参数名称	描述	输入/输出
AdChn	通道号。取值范围：[0, ADEC_MAX_CHN_NUM]。	输入

【返回值】

参数名称	描述
0	成功。
非 0	失败，其值为错误码 。

【需求】

- 头文件：cvi_comm_aio.h，cvi_audio.h、cvi_comm_adec.h
- 库文件：libcvi_audio.a，libcvi_VoiceEngine.so，libcvi_RES1.so

【注意事项】

- 要求解码通道已经被创建。
- 使用本接口时，不建议使用 stream 方式解码。使用 stream 方式解码进行清除区块作时，用户需要确保清除完区块后，发送给解码器的数据必须是完整的一帧码流，否则可能导致解码器不能正常工作。
- 无论是否使用 stream 方式解码，都要确保送数据解码的操作和清除区块的操作之间的同步。

【举例】

无。

10.3.6.5 CVI_ADEC_GetFrame

【描述】

获取解碼后音框。

【语法】

```
CVI_S32 CVI_ADEC_GetFrame(ADEC_CHN AdChn, AUDIO_FRAME_INFO_S *pstFrmInfo,
↪CVI_BOOL bBlock);
```

【参数】

参数名称	描述	输入/输出
AdChn	音频解码通道。	输入
pstFrmInfo	音频帧数据结构体。	输出
bBlock	是否以阻塞方式获取。	输入

【返回值】

参数名称	描述
0	成功。
非 0	失败，其值为错误码 。

【需求】

- 头文件：cvi_comm_aio.h，cvi_audio.h、cvi_comm_adec.h
- 库文件：libcvi_audio.a，libcvi_VoiceEngine.so，libcvi_RES1.so

【注意事项】

- 必须在 ADEC 通道创建之后调用
- 使用本接口获取解码帧数据时，建议发送码流时按帧发送
- 使用本接口获取音频数据时，请解除 ADEC 与 Audio Output 的绑定关系，否则获取到的帧是不连续的。
- 使用本接口获取音频帧数据时，如果发送码流按流发送，请务必保证获取解码帧数据的及时性，否则会有异常。

【举例】

无。

10.3.6.6 CVI_ADEC_ReleaseFrame

【描述】

释放获取到的音频解码帧数据。

【语法】

```
CVI_S32 CVI_ADEC_ReleaseFrame(ADEC_CHN AdChn, AUDIO_FRAME_INFO_S *pstFrmInfo);
```

【参数】

参数名称	描述	输入/输出
AdChn	音频解码通道。	输入
pstFrmInfo	音频帧数据结构体。	输入

【返回值】

参数名称	描述
0	成功。
非 0	失败，其值为错误码 。

【需求】

- 头文件：cvi_comm_aio.h，cvi_audio.h、cvi_comm_adec.h
- 库文件：libcvi_audio.a，libcvi_VoiceEngine.so，libcvi_RES1.so

【注意事项】

- 本接口必须与接口 CVI_ADEC_GetFrame 配合使用。
- 必须在 ADEC 通道创建之后调用。

【举例】

无。

10.3.6.7 CVI_ADEC_SendEndOfStream

【描述】

向解码器发送码流结束标识符，并清除码流 buffer。

【语法】

```
CVI_S32 CVI_ADEC_SendEndOfStream (ADEC_CHN AdChn, CVI_BOOL bInstant);
```

【参数】

参数名称	描述	输入/输出
AdChn	音频解码通道。	输入
bInstant	是否立即清除解码器内部的区块数据。 取值范围： CVI_FALSE：延时清除。不会立即清除解码器内部的缓存数据，解码会继续进行，直到剩余 buffer 不足一帧数据时进行清除操作。 CVI_TRUE：立即清除解码器内部区块数据。	输入

【返回值】

参数名称	描述
0	成功。
非 0	失败，其值为错误码 。

【需求】

- 头文件：cvi_comm_aio.h，cvi_audio.h、cvi_comm_adec.h
- 库文件：libcvi_audio.a，libcvi_VoiceEngine.so，libcvi_RES1.so

【注意事项】

无。

【举例】

无。

10.3.7 内置 Codec

内置 audio codec 主要透过 ioctl 来对硬体设备做操作。Ioctl 调用实现的是对内置 audio codec 寄存器的读写操作。

内置 audio codec 标准功能 cmd

- ACODEC_SOFT_RESET_CTRL： 将 Codec 恢复为默认设置。
- ACODEC_SET_INPUT_VOL： 输入总音量控制。
- ACODEC_GET_INPUT_VOL： 获取输入总音量。
- ACODEC_SET_OUTPUT_VOL： 输出总音量控制。
- ACODEC_GET_OUTPUT_VOL： 获取输出总音量。
- ACODEC_SET_GAIN_MICL： 左声道 MIC 输入的模拟增益控制。
- ACODEC_SET_GAIN_MICR： 右声道 MIC 输入的模拟增益控制。
- ACODEC_SET_DACL_VOL： 左声道输出音量控制。

- ACODEC_SET_DACR_VOL：右声道输出音量控制。
- ACODEC_SET_ADCL_VOL：左声道输入音量控制。
- ACODEC_SET_ADCR_VOL：右声道输入音量控制。
- ACODEC_SET_MICL_MUTE：左声道 MIC 输入静音控制。
- ACODEC_SET_MICR_MUTE：右声道 MIC 输入静音控制。
- ACODEC_SET_DACL_MUTE：左声道输出静音控制。
- ACODEC_SET_DACR_MUTE：右声道输出静音控制。
- ACODEC_GET_GAIN_MICL：获取模拟左声道 MIC 输入的增益。
- ACODEC_GET_GAIN_MICR：获取模拟右声道 MIC 输入的增益。
- ACODEC_GET_DACL_VOL：获取左声道输出的音量控制。
- ACODEC_GET_DACR_VOL：获取右声道输出的音量控制。
- ACODEC_GET_ADCL_VOL：获取左声道输入的音量控制。
- ACODEC_GET_ADCR_VOL：获取右声道输入的音量控制。
- ACODEC_SET_PD_DACL：左声道输出的下电控制。
- ACODEC_SET_PD_DACR：右声道输出的下电控制。
- ACODEC_SET_PD_ADCL：左声道输入的下电控制。
- ACODEC_SET_PD_ADCR：右声道输入的下电控制。
- ACODEC_SET_PD_LINEINL：左声道 LINEIN 输入的下电控制。
- ACODEC_SET_PD_LINEINR：右声道 LINEIN 输入的下电控制。

10.3.7.1 ACODEC_SOFT_RESET_CTRL

【描述】

将 Codec 恢复为默认设置。

【语法】

```
CVI_S32 ioctl (CVI_S32 fd, ACODEC_SOFT_RESET_CTRL);
```

【参数】

参数名称	描述	输入/输出
fd	Audio Codec 设备文件描述	输入
ACODEC_SOFT_RESET_CTRL	ioctl 命令	输入

【返回值】

参数名称	描述
0	成功。
非 0	失败，其值为错误码 。

【需求】

- 头文件：- 头文件：acodec.h

【注意事项】

无。

【举例】

```
if (ioctl(Acodec_Fd, ACODEC_SOFT_RESET_CTRL))
{
    printf("ioctl reset err!\n");
}
```

10.3.7.2 ACODEC_SET_INPUT_VOL

【描述】

输入总音量控制。把用户期望设置的增益配置到数字增益控制寄存器。

【语法】

```
CVI_S32 ioctl (CVI_S32 fd, ACODEC_SET_INPUT_VOL, CVI_U32 *arg);
```

【参数】

参数名称	描述	输入/输出
fd	Audio Codec 设备文件描述	输入
ACODEC_SET_INPUT_VOL	ioctl 命令	输入
arg	无符号整型指针	输入

【返回值】

参数名称	描述
0	成功。
非 0	失败，其值为错误码 。

【需求】

- 头文件：- 头文件：acodec.h

【注意事项】

- 输入音量范围为 [24~0] 。若输入值为 0 则为静音。
- 调节时左右声道同时生效。

【举例】

```
CVI_U32 vol = 20;
if (ioctl(Acodec_Fd, ACODEC_SET_INPUT_VOL, &vol))
{
printf("ioctl err!\n");
}
```

10.3.7.3 ACODEC_GET_INPUT_VOL

【描述】

获取输入总音量。

【语法】

```
CVI_S32 ioctl (CVI_S32 fd, ACODEC_GET_INPUT_VOL, CVI_U32 *arg);
```

【参数】

参数名称	描述	输入/输出
fd	Audio Codec 设备文件描述	输入
ACODEC_GET_INPUT_VOL	ioctl 命令	输入
arg	无符号整型指针	输出

【返回值】

参数名称	描述
0	成功。
非 0	失败，其值为错误码 。

【需求】

- 头文件：- 头文件：acodec.h

【注意事项】

- 输入音量范围为 [24~0] ，若输出值 0 则为静音。

【举例】

```
CVI_U32 vol;
if (ioctl(Acodec_Fd, ACODEC_SET_INPUT_VOL, &vol))
{
printf("ioctl err!\n");
}
```

10.3.7.4 ACODEC_SET_OUTPUT_VOL

【描述】

输出总音量控制。把用户期望设置的增益配置到数字增益控制寄存器。

【语法】

```
CVI_S32 ioctl (CVI_S32 fd, ACODEC_SET_OUTPUT_VOL, CVI_U32 *arg);
```

【参数】

参数名称	描述	输入/输出
fd	Audio Codec 设备文件描述	输入
ACODEC_SET_OUTPUT_VOL	ioctl 号	输入
arg	无符号整型指针	输入

【返回值】

参数名称	描述
0	成功。
非 0	失败，其值为错误码 。

【需求】

- 头文件：- 头文件：acodec.h

【注意事项】

输出音量范围为 [32~0], 调节时左右声道同时生效。

【举例】

```
CVI_U32 vol = 7;
if (ioctl(Acodec_Fd, ACODEC_SET_OUTPUT_VOL, &vol))
{
printf("ioctl err!\n");
}
```

10.3.7.5 ACODEC_GET_OUTPUT_VOL

【描述】

获取输出总音量。

【语法】

```
CVI_S32 ioctl (CVI_S32 fd, ACODEC_GET_OUTPUT_VOL, CVI_U32 *arg);
```

【参数】

参数名称	描述	输入/输出
fd	Audio Codec 设备文件描述	输入
ACODEC_GET_OUTPUT_VOL	ioctl 号	输入
arg	无符号整型指针	输出

【返回值】

参数名称	描述
0	成功。
非 0	失败，其值为错误码 。

【需求】

- 头文件：acodec.h

【注意事项】

输出音量范围为 [32~0]。

【举例】

```
CVI_U32 vol;
if (ioctl(Acodec_Fd, ACODEC_SET_INPUT_VOL, &vol))
{
printf("ioctl err!\n");
}
```

10.3.7.6 ACODEC_SET_GAIN_MICL

【描述】

左声道 MIC 输入的模拟增益控制。

【语法】

```
CVI_S32 ioctl (CVI_S32 fd, ACODEC_SET_GAIN_MICL, CVI_U32 *arg);
```

【参数】

参数名称	描述	输入/输出
fd	Audio Codec 设备文件描述	输入
ACODEC_SET_GAIN_MICL	MICL	输入
arg	无符号整型指针描述	输入

【返回值】

参数名称	描述
0	成功。
非 0	失败，其值为错误码 。

参数名称	输入/输出
CVI_SUCESS	成功
CVI_FAILURE	失败

【需求】

- 头文件：acodec.h

【注意事项】

- 在使用无源音频信号输入时，需适当的增大模拟增益。
- 输入音量范围为 [24~0], 若输入值为 0 则表示静音。

【举例】

```
CVI_U32 gain_mic;
gain_mic = 5;
if (ioctl(Acodec_Fd, ACODEC_SET_GAIN_MICL, &gain_mic))
{
printf("ioctl err!\n");
}
```

10.3.7.7 ACODEC_SET_GAIN_MICR

【描述】

右声道 MIC 输入的模拟增益控制。

【语法】

```
CVI_S32 ioctl (CVI_S32 fd, ACODEC_SET_GAIN_MICR, CVI_U32 *arg);
```

【参数】

参数名称	描述	输入/输出
fd	Audio Codec 设备文件描述	输入
ACODEC_SET_GAIN_MICR	MICR	输入
arg	无符号整型指针	输入

【返回值】

参数名称	描述
0	成功。
非 0	失败，其值为错误码 。

【需求】

- 头文件：acodec.h

【注意事项】

- 在使用无源音频信号输入时，需适当的增大模拟增益。
- 输入音量范围为 [24~0], 若输入值为 0 则表示静音。

【举例】

```
CVI_U32 gain_mic;
gain_mic = 5;
if (ioctl(Acodec_Fd, ACODEC_SET_GAIN_MICR, &gain_mic))
{
printf("ioctl err!\n");
}
```

10.3.7.8 ACODEC_SET_DACL_VOL

【描述】

左声道输出音量控制。

【语法】

```
CVI_S32 ioctl (CVI_S32 fd, ACODEC_SET_DACL_VOL, struct cvi_vol_ctrl *arg);
```

【参数】

参数名称	描述	输入/输出
fd	Audio Codec 设备文件描述	输入
ACODEC_SET_DACL_VOL	ioctl 命令	输入
arg	cvi_vol_ctrl 结构体指针描述	输入

【返回值】

参数名称	描述
0	成功。
非 0	失败，其值为错误码 。

【需求】

- 头文件：acodec.h

【注意事项】

输出音量范围为 [32~0]。若输入值 vol_ctrl.vol_ctrl_mute 为 1 则变成静音，vol_ctrl.vol_ctrl_mute 为 0 则为撤销静音。

【举例】

```
struct cvi_vol_ctrl vol_ctrl;
vol_ctrl.vol_ctrl_mute = 0x0;
vol_ctrl.vol_ctrl = 0x05;
if (ioctl(Acodec_Fd, ACODEC_SET_DACL_VOL, &vol_ctrl))
{
printf("ioctl err!\n");
}
```

10.3.7.9 ACODEC_SET_DACR_VOL

【描述】

右声道输出音量控制。

【语法】

```
CVI_S32 ioctl (CVI_S32 fd, ACODEC_SET_DACR_VOL, struct cvi_vol_ctrl *arg);
```

【参数】

参数名称	描述	输入/输出
fd	Audio Codec 设备文件描述	输入
ACODEC_SET_DACR_VOL	IoctlVOL号	输入
arg	cvi_vol_ctrl 结构体指针描述	输入

【返回值】

参数名称	描述
0	成功。
非 0	失败，其值为错误码 。

【需求】

- 头文件：acodec.h

【注意事项】

输出音量范围为 [32~0]。若输入值 vol_ctrl.vol_ctrl_mute 为 1 则变成静音，vol_ctrl.vol_ctrl_mute 为 0 则为撤销静音。

【举例】

```
struct cvi_vol_ctrl vol_ctrl;
vol_ctrl.vol_ctrl_mute = 0x0;
vol_ctrl.vol_ctrl = 0x05;
if (ioctl(Acodec_Fd, ACODEC_SET_DACR_VOL, &vol_ctrl))
{
printf("ioctl err!\n");
}
```

10.3.7.10 ACODEC_SET_ADCL_VOL

【描述】

左声道输入音量控制。

【语法】

```
CVI_S32 ioctl (CVI_S32 fd, ACODEC_SET_ADCL_VOL, struct cvi_vol_ctrl *arg);
```

【参数】

参数名称	描述	输入/输出
fd	Audio Codec 设备文件描述	输入
ACODEC_SET_ADCL_VOL	IoctlVOL号	输入
arg	cvi_vol_ctrl 结构体指针描述	输入

【返回值】

参数名称	描述
0	成功。
非 0	失败，其值为错误码 。

【需求】

- 头文件：acodec.h

【注意事项】

- Input 配置为 MIC in 时，输入音量范围为 [24~0]。若输入值 vol_ctrl.vol_ctrl_mute 为 1 则变成静音，vol_ctrl.vol_ctrl_mute 为 0 则为撤销静音

【举例】

```
struct cvi_vol_ctrl vol_ctrl;
vol_ctrl.vol_ctrl_mute = 0x0;
vol_ctrl.vol_ctrl = 0x05;
if (ioctl(Acodec_Fd, ACODEC_SET_ADCL_VOL, &vol_ctrl))
{
printf("ioctl err!\n");
}
```

10.3.7.11 ACODEC_SET_ADCR_VOL

【描述】

右声道输入音量控制。

【语法】

```
CVI_S32 ioctl (CVI_S32 fd, ACODEC_SET_ADCR_VOL, struct cvi_vol_ctrl *arg);
```

【参数】

参数名称	描述	输入/输出
fd	Audio Codec 设备文件描述	输入
ACODEC_SET_ADCR_VOL	IoctlVOL号	输入
arg	cvi_vol_ctrl 结构体指针描述	输入

【返回值】

参数名称	描述
0	成功。
非 0	失败，其值为错误码 。

【需求】

- 头文件：acodec.h

【注意事项】

- Input 配置为 MIC in 时，输入音量范围为 [24~0]。若输入值 vol_ctrl.vol_ctrl_mute 为 1 则变成静音，vol_ctrl.vol_ctrl_mute 为 0 则为撤销静音

【举例】

```
struct cvi_vol_ctrl vol_ctrl;
vol_ctrl.vol_ctrl_mute = 0x0;
vol_ctrl.vol_ctrl = 0x05;
if (ioctl(Acodec_Fd, ACODEC_SET_ADCR_VOL, &vol_ctrl))
{
printf("ioctl err!\n");
}
```

10.3.7.12 ACODEC_SET_MICL_MUTE

【描述】

左声道 MIC 输入静音控制。

【语法】

```
CVI_S32 ioctl (CVI_S32 fd, ACODEC_SET_MICL_MUTE, CVI_U32 *arg);
```

【参数】

参数名称	描述	输入/输出
fd	Audio Codec 设备文件描述	输入
ACODEC_SET_MICL_MUTE	IoctL 号	输入
arg	无符号整型指针	输入

【返回值】

参数名称	描述
0	成功。
非 0	失败，其值为错误码 。

【需求】

- 头文件：acodec.h

【注意事项】

arg 参数范围 [0, 1]，0 表示撤销静音，1 表示静音。

【举例】

```
CVI_U32 mic_mute;
mic_mute = 0;
if (ioctl(Acodec_Fd, ACODEC_SET_MICL_MUTE, &mic_mute))
{
printf("ioctl err!\n");
}
```

10.3.7.13 ACODEC_SET_MICR_MUTE

【描述】

右声道 MIC 输入静音控制。

【语法】

```
CVI_S32 ioctl (CVI_S32 fd, ACODEC_SET_MICR_MUTE, CVI_U32 *arg);
```

【参数】

参数名称	描述	输入/输出
fd	Audio Codec 设备文件描述	输入
ACODEC_SET_MICR_MUTE	Ioctl 码	输入
arg	无符号整型指针	输入

【返回值】

参数名称	描述
0	成功。
非 0	失败，其值为错误码 。

【需求】

- 头文件：acodec.h

【注意事项】

arg 参数范围 [0, 1]，0 表示撤销静音，1 表示静音。

【举例】

```
CVI_U32 mic_mute;
mic_mute = 0;
if (ioctl(Acodec_Fd, ACODEC_SET_MICR_MUTE, &mic_mute))
{
printf("ioctl err!\n");
}
```

10.3.7.14 ACODEC_SET_DACL_MUTE

【描述】

左声道输出静音控制。

【语法】

```
CVI_S32 ioctl (CVI_S32 fd, ACODEC_SET_DACL_MUTE, CVI_U32 *arg);
```

【参数】

参数名称	描述	输入/输出
fd	Audio Codec 设备文件描述	输入
ACODEC_SET_DACL_MUTE	设置左声道输出静音控制命令字	输入
arg	无符号整型指针	输入

【返回值】

参数名称	描述
0	成功。
非 0	失败，其值为错误码 。

【需求】

- 头文件：acodec.h

【注意事项】

arg 参数范围 [0, 1]，0 表示撤销静音，1 表示静音。

【举例】

```
CVI_U32 mute;
mute = 0;
if (ioctl(Acodec_Fd, ACODEC_SET_DACL_MUTE, &mute))
{
printf("ioctl err!\n");
}
```

10.3.7.15 ACODEC_SET_DACR_MUTE

【描述】

右声道输出静音控制。

【语法】

```
CVI_S32 ioctl (CVI_S32 fd, ACODEC_SET_DACR_MUTE, CVI_U32 *arg);
```

【参数】

参数名称	描述	输入/输出
fd	Audio Codec 设备文件描述	输入
ACODEC_SET_DACR_MUTE	设置右声道输出静音控制命令字	输入
arg	无符号整型指针	输入

【返回值】

参数名称	描述
0	成功。
非 0	失败，其值为错误码 。

【需求】

· 头文件：acodec.h

【注意事项】

arg 参数范围 [0, 1]，0 表示撤销静音，1 表示静音。

【举例】

```
CVI_U32 mute;
mute = 0;
if (ioctl(Acodec_Fd, ACODEC_SET_DACR_MUTE, &mute))
{
printf("ioctl err!\n");
}
```

10.3.7.16 ACODEC_GET_GAIN_MICL

【描述】

获取模拟左声道 MIC 输入的增益。

【语法】

```
CVI_S32 ioctl (CVI_S32 fd, ACODEC_GET_GAIN_MICL, CVI_U32 *arg);
```

【参数】

参数名称	描述	输入/输出
fd	Audio Codec 设备文件描述	输入
ACODEC_GET_GAIN_MICL	模式号	输入
arg	无符号整型指针	输出

【返回值】

参数名称	描述
0	成功。
非 0	失败，其值为错误码 。

【需求】

· 头文件：acodec.h

【注意事项】

无。

【举例】

```
CVI_U32 gain_mic;
mute = 0;
if (ioctl(Acodec_Fd, ACODEC_GET_GAIN_MICL, &gain_mic))
{
```

（下页继续）

(续上页)

```
  printf("ioctl err!\n");
}
```

10.3.7.17 ACODEC_GET_GAIN_MICR

【描述】

获取模拟右声道 MIC 输入的增益。

【语法】

```
CVI_S32 ioctl (CVI_S32 fd, ACODEC_GET_GAIN_MICR, CVI_U32 *arg);
```

【参数】

参数名称	描述	输入/输出
fd	Audio Codec 设备文件描述	输入
ACODEC_GET_GAIN_MICR	MIC 增益	输入
arg	无符号整型指针	输出

【返回值】

参数名称	描述
0	成功。
非 0	失败，其值为错误码 。

【需求】

- 头文件：acodec.h

【注意事项】

无。

【举例】

```
CVI_U32 gain_mic;
mute = 0;
if (ioctl(Acodec_Fd, ACODEC_GET_GAIN_MICR, &gain_mic))
{
printf("ioctl err!\n");
}
```

10.3.7.18 ACODEC_GET_DACL_VOL

【描述】

获取左声道输出的音量控制。

【语法】

```
CVI_S32 ioctl (CVI_S32 fd, ACODEC_GET_DACL_VOL, struct cvi_vol_ctrl *arg);
```

【参数】

参数名称	描述	输入/输出
fd	Audio Codec 设备文件描述	输入
ACODEC_GET_DACL_VOL	获取左声道输出的音量控制	输入
arg	cvi_vol_ctrl 结构体指针描述	输出

【返回值】

参数名称	描述
0	成功。
非 0	失败，其值为错误码 。

【需求】

- 头文件：acodec.h

【注意事项】

无。

【举例】

```
struct cvi_vol_ctrl vol_ctrl;
if (ioctl(Acodec_Fd, ACODEC_GET_DACL_VOL, & vol_ctrl))
{
printf("ioctl err!\n");
}
```

10.3.7.19 ACODEC_GET_DACR_VOL

【描述】

获取右声道输出的音量控制。

【语法】

```
CVI_S32 ioctl (CVI_S32 fd, ACODEC_GET_DACR_VOL, struct cvi_vol_ctrl *arg);
```

【参数】

参数名称	描述	输入/输出
fd	Audio Codec 设备文件描述	输入
ACODEC_GET_DACR_VOL	ioctl 命令	输入
arg	cvi_vol_ctrl 结构体指针描述	输出

【返回值】

参数名称	描述
0	成功。
非 0	失败，其值为错误码 。

【需求】

- 头文件：acodec.h

【注意事项】

无。

【举例】

```
struct cvi_vol_ctrl vol_ctrl;
if (ioctl(Acodec_Fd, ACODEC_GET_DACR_VOL, & vol_ctrl))
{
printf("ioctl err!\n");
}
```

10.3.7.20 ACODEC_GET_ADCL_VOL

【描述】

获取左声道输入的音量控制。

【语法】

```
CVI_S32 ioctl (CVI_S32 fd, ACODEC_GET_ADCL_VOL, struct cvi_vol_ctrl *arg);
```

【参数】

参数名称	描述	输入/输出
fd	Audio Codec 设备文件描述	输入
ACODEC_GET_ADCL_VOL	ioctl 命令	输入
arg	cvi_vol_ctrl 结构体指针描述	输出

【返回值】

参数名称	描述
0	成功。
非 0	失败，其值为错误码 。

【需求】

- 头文件：acodec.h

【注意事项】

无。

【举例】

```
struct cvi_vol_ctrl vol_ctrl;
if (ioctl(Acodec_Fd, ACODEC_GET_ADCL_VOL, & vol_ctrl))
{
printf("ioctl err!\n");
}
```

10.3.7.21 ACODEC_GET_ADCR_VOL

【描述】

获取右声道输入的音量控制。

【语法】

```
CVI_S32 ioctl (CVI_S32 fd, ACODEC_GET_ADCR_VOL, struct cvi_vol_ctrl *arg);
```

【参数】

参数名称	描述	输入/输出
fd	Audio Codec 设备文件描述	输入
ACODEC_GET_ADCR_VOL	ioctl 命令	输入
arg	cvi_vol_ctrl 结构体指针描述	输出

【返回值】

参数名称	描述
0	成功。
非 0	失败，其值为错误码 。

【需求】

- 头文件：acodec.h

【注意事项】

无。

【举例】

```
struct cvi_vol_ctrl vol_ctrl;
if (ioctl(Acodec_Fd, ACODEC_GET_ADCR_VOL, & vol_ctrl))
{
printf("ioctl err!\n");
}
```

10.3.7.22 ACODEC_SET_PD_DACL

【描述】

左声道输出的下电控制。

【语法】

```
CVI_S32 ioctl (CVI_S32 fd, ACODEC_SET_PD_DACL, CVI_U32 *arg);
```

【参数】

参数名称	描述	输入/输出
fd	Audio Codec 设备文件描述	输入
ACODEC_SET_PD_DACL	DACL号	输入
arg	无符号整型指针	输入

【返回值】

参数名称	描述
0	成功。
非 0	失败，其值为错误码 。

【需求】

- 头文件：acodec.h

【注意事项】

不使用 DACL 时，可以调用此接口，将 DACL 下电。arg 参数范围 [0, 1]。0 表示上电，1 表示下电。

【举例】

```
CVI_U32 pd_ctrl;
pd_ctrl = 0x1;
if (ioctl(Acodec_Fd, ACODEC_SET_PD_DACL, &pd_ctrl))
{
printf("ioctl err!\n");
}
```

10.3.7.23 ACODEC_SET_PD_DACR

【描述】

右声道输出的下电控制。

【语法】

```
CVI_S32 ioctl (CVI_S32 fd, ACODEC_SET_PD_DACR, CVI_U32 *arg);
```

【参数】

参数名称	描述	输入/输出
fd	Audio Codec 设备文件描述	输入
ACODEC_SET_PD_DACR	DACR 号	输入
arg	无符号整型指针描述	输入

【返回值】

参数名称	描述
0	成功。
非 0	失败，其值为错误码 。

【需求】

- 头文件：acodec.h

【注意事项】

不使用 DACR 时，可以调用此接口，将 DACR 下电。arg 参数范围 [0, 1]。0 表示上电，1 表示下电。

【举例】

```
CVI_U32 pd_ctrl;
pd_ctrl = 0x1;
if (ioctl(Acodec_Fd, ACODEC_SET_PD_DACR, &pd_ctrl))
{
printf("ioctl err!\n");
}
```

10.3.7.24 ACODEC_SET_PD_ADCL

【描述】

左声道输入的下电控制。

【语法】

```
CVI_S32 ioctl (CVI_S32 fd, ACODEC_SET_PD_ADCL, CVI_U32 *arg);
```

【参数】

参数名称	描述	输入/输出
fd	Audio Codec 设备文件描述	输入
ACODEC_SET_PD_ADCL	ADCL 号	输入
arg	无符号整型指针	输入

【返回值】

参数名称	描述
0	成功。
非 0	失败，其值为错误码 。

【需求】

· 头文件：acodec.h

【注意事项】

不使用 ADCL 时，可以调用此接口，将 ADCL 下电。arg 参数范围 [0, 1]。0 表示上电，1 表示下电。

【举例】

```
CVI_U32 pd_ctrl;
pd_ctrl = 0x1;
if (ioctl(Acodec_Fd, ACODEC_SET_PD_ADCL, &pd_ctrl))
{
printf("ioctl err!\n");
}
```

10.3.7.25 ACODEC_SET_PD_ADCR

【描述】

右声道输入的下电控制。

【语法】

```
CVI_S32 ioctl (CVI_S32 fd, ACODEC_SET_PD_ADCR, CVI_U32 *arg);
```

【参数】

参数名称	描述	输入/输出
fd	Audio Codec 设备文件描述	输入
ACODEC_SET_PD_ADCR	ioctl 号	输入
arg	无符号整型指针描述	输入

【返回值】

参数名称	描述
0	成功。
非 0	失败，其值为错误码 。

【需求】

· 头文件：acodec.h

【注意事项】

不使用 ADCR 时，可以调用此接口，将 ADCR 下电。arg 参数范围 [0, 1]。0 表示上电，1 表示下电。

【举例】

```
CVI_U32 pd_ctrl;
pd_ctrl = 0x1;
if (ioctl(Acodec_Fd, ACODEC_SET_PD_ADCR, &pd_ctrl))
{
printf("ioctl err!\n");
}
```

10.3.7.26 ACODEC_SET_PD_LINEINL

【描述】

左声道 LINEIN 输入的下电控制。

【语法】

```
CVI_S32 ioctl (CVI_S32 fd, ACODEC_SET_PD_LINEINL, CVI_U32 *arg);
```

【参数】

参数名称	描述	输入/输出
fd	Audio Codec 设备文件描述	输入
ACODEC_SET_PD_LINEINL	0x1E	输入
arg	无符号整型指针	输入

【返回值】

参数名称	描述
0	成功。
非 0	失败，其值为错误码 。

【需求】

- 头文件：acodec.h

【注意事项】

不使用 LINEINL 时，可以调用此接口，将 LINEINL 下电。arg 参数范围 [0, 1]。0 表示上电，1 表示下电。

【举例】

```
CVI_U32 pd_ctrl;
pd_ctrl = 0x1;
if (ioctl(Acodec_Fd, ACODEC_SET_PD_LINEINL, &pd_ctrl))
{
printf("ioctl err!\n");
}
```

10.3.7.27 ACODEC_SET_PD_LINEINR

【描述】

右声道 LINEIN 输入的下电控制。

【语法】

```
CVI_S32 ioctl (CVI_S32 fd, ACODEC_SET_PD_LINEINR, CVI_U32 *arg);
```

【参数】

参数名称	描述	输入/输出
fd	Audio Codec 设备文件描述	输入
ACODEC_SET_PD_LINEINR	LINEINR	输入
arg	无符号整型指针	输入

【返回值】

参数名称	描述
0	成功。
非 0	失败，其值为错误码 。

【需求】

- 头文件：acodec.h

【注意事项】

不使用 LINEINR 时，可以调用此接口，将 LINEINR 下电。arg 参数范围 [0, 1]。0 表示上电，1 表示下电。

【举例】

```
CVI_U32 pd_ctrl;
pd_ctrl = 0x1;
if (ioctl(Acodec_Fd, ACODEC_SET_PD_LINEINR, &pd_ctrl))
{
printf("ioctl err!\n");
}
```

10.3.8 重采样

Resampler 模块提供单独的重采样处理。当客户需要在上层对数据重采样时，可以使用该模块。

以下为相关 API 及细部说明:

- CVI_Resampler_Create： 创建一个重采样模块。
- CVI_Resampler_Process： 重采样模块数据处理。
- CVI_Resampler_Destroy： 销毁重采样模块。
- CVI_Resampler_GetMaxOutputNum： 计算重采样最大输出数据。

10.3.8.1 CVI_Resampler_Create

【描述】

创建一个重采样模块。

【语法】

```
CVI_VOID* CVI_Resampler_Create(CVI_S32 s32Inrate, CVI_S32 s32Outrate, CVI_S32 s32Chans);
```

【参数】

参数名称	描述	输入/输出
s32Inrate	输入采样率。取值范围：8000, 11025, 12000, 16000, 22050, 24000, 32000, 44100, 48000, 64000。	输入
s32Outrate	输出采样率。取值范围：8000, 11025, 12000, 16000, 22050, 24000, 32000, 44100, 48000, 64000。	输入
s32Chans	处理声道数 (目前 Cvitek 支持单声道)	输入

【返回值】

参数名称	描述
0	成功。
非 0	失败，其值为错误码 。

【需求】

- 头文件：cvi_comm_aio.h，cvi_audio.h、cvi_resample_api.h
- 库文件：libcvi_audio.a，libcvi_RES1.so

【注意事项】

输入采样率和输出采样率需保证不一致。

【举例】

无。

10.3.8.2 CVI_Resampler_Process

【描述】

处理一帧重采样数据。

【语法】

```
CVI_S32 CVI_Resampler_Process(CVI_VOID* inst, CVI_S16* s16Inbuf, CVI_S32 s32Insamps,
↪CVI_S16* s16Outbuf);
```

【参数】

参数名称	描述	输入/输出
inst	重采样模块句柄。	输入
s16Inbuf	输入数据 buf 指针。	输入
s32Insamps	输入数据采样点数。取值范围：[0, 2048]	输入
s16Outbuf	输出数据 buf 指针	输出

【返回值】

参数名称	描述
0	成功。
非 0	失败，其值为错误码 。

【需求】

- 头文件：cvi_comm_aio.h，cvi_audio.h、cvi_resample_api.h
- 库文件：libcvi_audio.a，libcvi_RES1.so

【注意事项】

最大输入采样点数需小于 2048。

【举例】

无。

10.3.8.3 CVI_Resampler_Destroy

【描述】

销毁一个重采样模块实例。

【语法】

```
CVI_VOID CVI_Resampler_Destroy(CVI_VOID* inst);
```

【参数】

参数名称	描述	输入/输出
inst	重采样模块句柄。	输入

【返回值】

参数名称	描述
0	成功。
非 0	失败，其值为错误码 。

【需求】

- 头文件：cvi_comm_aio.h，cvi_audio.h、cvi_resample_api.h
- 库文件：libcvi_audio.a，libcvi_RES1.so

【注意事项】

无。

【举例】

无。

10.3.8.4 CVI_Resampler_GetMaxOutputNum

【描述】

获取最大输出采样点数（每一声道）。

【语法】

```
CVI_S32 CVI_Resampler_GetMaxOutputNum(CVI_VOID* inst, CVI_S32 s32Insamps);
```

【参数】

参数名称	描述	输入/输出
inst	重采样模块句柄。	输入
s32Insamps	每一声道输入采样点数。	输入

【返回值】

参数名称	描述
0	成功。
非 0	失败，其值为错误码 。

【需求】

- 头文件：cvi_comm_aio.h，cvi_audio.h、cvi_resample_api.h
- 库文件：libcvi_audio.a，libcvi_RES1.so

【注意事项】

无。

【举例】

无。

10.4 数据类型

10.4.1 音频输入/输出

音频输入/输出相关数据类型、数据结构定义如下：

- AI_DEV_MAX_NUM：定义音频输入设备的最大数
- AO_DEV_MAX_NUM：定义音频输出设备的最大数

- AI_MAX_CHN_NUM：定义音频输入设备的最大通道数。
- AO_MAX_CHN_NUM：定义音频输出设备的最大通道数。
- CVI_AUD_MAX_CHANNEL_NUM：定义音频输出设备的最大通道数。
- AI_TALKVQE_MASK_AEC：Talk Vqe AEC 功能的 Mask。
- AI_TALKVQE_MASK_AGC：Talk Vqe AGC 功能的 Mask。
- AI_TALKVQE_MASK_ANR：Talk Vqe ANR 功能的 Mask。
- AI_RECORDVQE_MASK_AGC：Record Vqe AGC 功能的 Mask。
- MAX_AUDIO_FILE_PATH_LEN：音频保存文件的路径的最大长度限制。
- MAX_AUDIO_FILE_NAME_LEN：音频保存文件的名称的最大长度限制。
- AUDIO_CLKSEL_E：定义音频时钟源。
- AUDIO_SAMPLE_RATE_E：定义音频采样率。
- AUDIO_BIT_WIDTH_E：定义音频采样精度。
- AIO_MODE_E：定义音频输入/输出工作模式。
- AIO_I2STYPE_E：定义设备 I2S 对接设备类型。
- AUDIO_SOUND_MODE_E：定义音频声道模式。
- AUDIO_MOD_PARAM_S：定义音频模块参数结构体。
- AIO_ATTR_S：定义音频输入/输出设备属性结构体。
- AI_CHN_PARAM_S：定义通道参数结构体。
- AUDIO_FRAME_S：定义音频帧数据结构体。
- AEC_FRAME_S：定义回声抵消参考帧信息结构体。
- AUDIO_AGC_CONFIG_S：定义音频自动增益控制配置信息结构体。
- AI_AEC_CONFIG_S：定义音频回声抵消配置信息结构体。
- AUDIO_ANR_CONFIG_S：定义音频语音降噪功能配置信息结构体。
- AUDIO_DELAY_CONFIG_S: 定义音频信号延迟结构体。
- VQE_WORKSTATE_E：定义声音质量增强的工作模式。
- VQE_RECORD_TYPE：定义录音类型。
- VQE_EQ_BAND_NUM：定义 EQ 功能可调节的频段数。
- AI_TALKVQE_CONFIG_S：定义音频输入声音质量增强（Talk）配置信息结构体。
- AI_RECORDVQE_CONFIG_S：定义音频输入声音质量增强（Record）配置信息结构体。
- AO_VQE_CONFIG_S：定义音频输出声音质量增强配置信息结构体。
- AUDIO_STREAM_S：定义音频码流结构体。
- AO_CHN_STATE_S：音频输出通道的数据区块状态结构体。
- AUDIO_TRACK_MODE_E：音频设备声道模式类型。

- AUDIO_FADE_RATE_E：音频设备淡入淡出速率类型。
- AUDIO_FADE_S：音频设备淡入淡出设置结构体。
- G726_BPS_E：定义 G.726 编解码协议速率。
- ADPCM_TYPE_E：定义 ADPCM 编解码协议类型。
- AUDIO_SAVE_FILE_INFO_S：定义音频保存文件功能配置信息结构体。
- AUDIO_FILE_STATUS_S：定义音频文件保存状态结构体。
- VQE_MODULE_CONFIG_S：定义声音质量增强及重采样模块配置信息结构体。
- AUDIO_VQE_REGISTER_S：定义声音质量增强及重采样模块注册结构体。
- ST_CVI_WAV_HEADER：定义 Wav 标头参考结构体。

以下功能目前不支持：

- AI_TALKVQE_MASK_HPF：Talk Vqe HPF 功能的 Mask。
- AI_TALKVQE_MASK_EQ：Talk Vqe EQ 功能的 Mask。
- AI_RECORDVQE_MASK_HPF：Record Vqe HPF 功能的 Mask。
- AI_RECORDVQE_MASK_RNR：Record Vqe RNR 功能的 Mask。
- AI_RECORDVQE_MASK_HDR：Record Vqe HDR 功能的 Mask。
- AI_RECORDVQE_MASK_DRC：Record Vqe DRC 功能的 Mask。
- AI_RECORDVQE_MASK_EQ：Record Vqe EQ 功能的 Mask。
- AO_VQE_MASK_HPF：AO Vqe HPF 功能的 Mask。

10.4.1.1 AIO_MAX_NUM

【说明】

定义音频输入/输出设备的最大个数。

【定义】

```
#define AIO_MAX_NUM        2
```

【注意事项】

已弃用。

【相关数据类型及接口】

无。

10.4.1.2 AI_DEV_MAX_NUM

【说明】

定义音频输入设备的最大个数。

【定义】

```
#define AI_DEV_MAX_NUM    1
```

【注意事项】

已弃用。

【相关数据类型及接口】

无。

10.4.1.3 AO_DEV_MAX_NUM

【说明】

定义音频输出设备的最大个数。

【定义】

```
#define AO_DEV_MAX_NUM    1
```

【注意事项】

已弃用。

【相关数据类型及接口】

无。

10.4.1.4 AI_MAX_CHN_NUM

【说明】

定义音频输入设备的最大通道个数。

【定义】

```
#define AI_MAX_CHN_NUM    2
```

【注意事项】

已经弃用参考CVI_AUD_MAX_CHANNEL_NUM

【相关数据类型及接口】

无。

10.4.1.5 AO_MAX_CHN_NUM

【说明】

定义音频输出设备的最大通道个数。

【定义】

```
#define AO_MAX_CHN_NUM      1
```

【注意事项】

已经弃用参考CVI_AUD_MAX_CHANNEL_NUM

【相关数据类型及接口】

无。

10.4.1.6 CVI_AUD_MAX_CHANNEL_NUM

【说明】

定义音频输入/输出设备的最大通道个数。

【定义】

```
#define CVI_AUD_MAX_CHANNEL_NUM      3
```

【注意事项】

音频设备最大通道数用户可通过参数设置 AIO_ATTR_S stAttr;stAttr.u32ChnCnt 设置，但最大不超过 CVI_AUD_MAX_CHANNEL_NUM

【相关数据类型及接口】

无。

10.4.1.7 AI_TALKVQE_MASK_AEC

【说明】

定义 Talk Vqe AEC 功能的 Mask。

【定义】

```
#define AI_TALKVQE_MASK_AEC  0x3
```

【注意事项】

无。

【相关数据类型及接口】

无。

10.4.1.8 AI_TALKVQE_MASK_AGC

【说明】

定义 Talk Vqe AGC 功能的 Mask。

【定义】

```
#define AI_TALKVQE_MASK_AGC 0x8
```

【注意事项】

无。

【相关数据类型及接口】

赋值给 AI_TALKVQE_CONFIG_S 结构体成员 u32OpenMask 表示开启 AGC 功能。

如 u32OpenMask = AI_TALKVQE_MASK_AEC | AI_TALKVQE_MASK_AGC; 表示开启 AEC 和 AGC 功能。

10.4.1.9 AI_TALKVQE_MASK_ANR

【说明】

定义 Talk Vqe ANR 功能的 Mask。

【定义】

```
#define AI_TALKVQE_MASK_ANR 0x4
```

【注意事项】

无。

【相关数据类型及接口】

赋值给 AI_TALKVQE_CONFIG_S 结构体成员 u32OpenMask 表示开启 ANR 功能。

如 u32OpenMask = AI_TALKVQE_MASK_AEC | AI_TALKVQE_MASK_ANR; 表示开启 AEC 和 ANR 功能。

10.4.1.10 AI_RECORDVQE_MASK_AGC

【说明】

定义 Record Vqe AGC 功能的 Mask。

【定义】

```
#define AI_RECORDVQE_MASK_AGC 0x20
```

【注意事项】

无。

【相关数据类型及接口】

赋值给 AI_RECORDVQE_CONFIG_S 结构体成员 u32OpenMask 表示开启 AGC 功能。

如 u32OpenMask =AI_RECORDVQE_MASK_HPF | AI_RECORDVQE_MASK_AGC；表示开启 HPF 和 AGC 功能。

10.4.1.11 MAX_AUDIO_FILE_PATH_LEN

【说明】

定义音频保存文件的路径的最大长度限制。

【定义】

```
#define MAX_AUDIO_FILE_PATH_LEN 256
```

【注意事项】

无。

【相关数据类型及接口】

AUDIO_SAVE_FILE_INFO_S

10.4.1.12 MAX_AUDIO_FILE_NAME_LEN

【说明】

定义音频保存文件的名称的最大长度限制。

【定义】

```
#define MAX_AUDIO_FILE_NAME_LEN 256
```

【注意事项】

无。

【相关数据类型及接口】

AUDIO_SAVE_FILE_INFO_S

10.4.1.13 AUDIO_CLKSEL_E

【说明】

定义音频时钟源。

【定义】

```
typedef enum _AUDIO_CLKSEL_E
{ AUDIO_CLKSEL_BASE      = 0,  /*<Audio base clk. */
  AUDIO_CLKSEL_SPARE,          /*<Audio spare clk. */
  AUDIO_CLKSEL_BUTT,
} AUDIO_CLKSEL_E;
```

【成员】

无。

【注意事项】

Cvitek 使用者无需对此时钟下设定。

【相关数据类型及接口】

AUDIO_MOD_PARAM_S

10.4.1.14 AUDIO_SAMPLE_RATE_E

【说明】

定义音频采样率。

【定义】

```
typedef enum _AUDIO_SAMPLE_RATE_E
{ AUDIO_SAMPLE_RATE_8000 =8000,  /* 8K samplerate */
AUDIO_SAMPLE_RATE_11025 =11025,  /* 11.025K samplerate */
AUDIO_SAMPLE_RATE_16000 =16000,  /* 16K samplerate */
AUDIO_SAMPLE_RATE_22050 =22050,  /* 22.050K samplerate */
AUDIO_SAMPLE_RATE_24000 =24000,  /* 24K samplerate */
AUDIO_SAMPLE_RATE_32000 =32000,  /* 32K samplerate */
AUDIO_SAMPLE_RATE_44100 =44100,  /* 44.1K samplerate */
AUDIO_SAMPLE_RATE_48000 =48000,  /* 48K samplerate */
AUDIO_SAMPLE_RATE_64000=64000,  /* 64K samplerate*/
AUDIO_SAMPLE_RATE_BUTT, }AUDIO_SAMPLE_RATE_E;
```

【成员】

成员名称	描述
AUDIO_SAMPLE_RATE_8000	8kHz 采样率
AUDIO_SAMPLE_RATE_11025	11.025kHz 采样率
AUDIO_SAMPLE_RATE_16000	16kHz 采样率
AUDIO_SAMPLE_RATE_22050	22.050kHz 采样率
AUDIO_SAMPLE_RATE_24000	24kHz 采样率
AUDIO_SAMPLE_RATE_32000	32kHz 采样率
AUDIO_SAMPLE_RATE_44100	44.1kHz 采样率
AUDIO_SAMPLE_RATE_48000	48kHz 采样率
AUDIO_SAMPLE_RATE_64000	64kHz 采样率

【注意事项】

无。

【相关数据类型及接口】

AIO_ATTR_S

10.4.1.15 AUDIO_BIT_WIDTH_E

【说明】

定义音频采样精度。

【定义】

```
typedef enum _AUDIO_BIT_WIDTH_E {
  AUDIO_BIT_WIDTH_8 =0,  /* 8bit width */
  AUDIO_BIT_WIDTH_16 =1,  /* 16bit width */
  AUDIO_BIT_WIDTH_24 =2,  /* 24bit width */
  AUDIO_BIT_WIDTH_32 =3,  /* 32bit width */
  AUDIO_BIT_WIDTH_BUTT, }AUDIO_BIT_WIDTH_E;
```

【成员】

成员名称	描述
AUDIO_BIT_WIDTH_8	采样精度为 8bit 位宽
AUDIO_BIT_WIDTH_16	采样精度为 16bit 位宽
AUDIO_BIT_WIDTH_24	采样精度为 24bit 位宽
AUDIO_BIT_WIDTH_32	采样精度为 32bit 位宽

【注意事项】

无。

【相关数据类型及接口】

AIO_ATTR_S

10.4.1.16 AIO_MODE_E

【说明】

定义音频输入/输出设备工作模式。

【定义】

```
typedef enum hiAIO_MODE_E {
  AIO_MODE_I2S_MASTER = 0,  /* AIO I2S master mode */
  AIO_MODE_I2S_SLAVE,     /* AIO I2S slave mode */
  AIO_MODE_PCM_SLAVE_STD, /* AIO PCM slave standard mode */
  AIO_MODE_PCM_SLAVE_NSTD, /* AIO PCM slave non-standard mode */
  AIO_MODE_PCM_MASTER_STD,  /* AIO PCM master standard mode */
  AIO_MODE_PCM_MASTER_NSTD, /* AIO PCM master non-standard mode */
  AIO_MODE_BUTT
}AIO_MODE_E;
```

【成员】

成员名称	描述
AIO_MODE_I2S_MASTER	I2S 主模式
AIO_MODE_I2S_SLAVE	I2S 从模式
AIO_MODE_PCM_SLAVE_STD	PCM 从模式（标准协议）
AIO_MODE_PCM_SLAVE_NSTD	PCM 从模式（自定义协议）
AIO_MODE_PCM_MASTER_STD	PCM 主模式（标准协议）
AIO_MODE_PCM_MASTER_NSTD	PCM 主模式（自定义协议）

【注意事项】

Cvitek 内建，只支持 I2S 主模式。

【相关数据类型及接口】

AIO_ATTR_S

10.4.1.17 AIO_I2STYPE_E

【说明】

定义设备 I2S 对接设备类型。

【定义】

```
typedef enum {
  AIO_I2STYPE_INNERCODEC = 0, /* AIO I2S connect inner audio CODEC */
  AIO_I2STYPE_INNERHDMI,      /* AIO I2S connect Inner HDMI */
  AIO_I2STYPE_EXTERN,         /* AIO I2S connect extern hardware */
} AIO_I2STYPE_E;
```

【成员】

成员名称	描述
AIO_I2STYPE_INNERCODEC	对接内置 CODEC
AIO_I2STYPE_INNERHDMI	对接内置 HDMI
AIO_I2STYPE_EXTERN	对接外接设备

【注意事项】

Cvitek 仅支援 AIO_I2STYPE_INNERCODEC 对接内置 CODEC。

【相关数据类型及接口】

AIO_ATTR_S

10.4.1.18 AUDIO_SOUND_MODE_E

【说明】

定义音频声道模式。

【定义】

```
typedef enum _AIO_SOUND_MODE_E {
  AUDIO_SOUND_MODE_MONO =0, /*mono*/
  AUDIO_SOUND_MODE_STEREO =1, /*stereo*/
  AUDIO_SOUND_MODE_BUTT
}AUDIO_SOUND_MODE_E;
```

【成员】

成员名称	描述
AUDIO_SOUND_MODE_MONO	单声道
AUDIO_SOUND_MODE_STEREO	双声道

【注意事项】

左声道对应通道 0，右声道对应通道 1。

对 Audio Input 来说，单声道默认从左声道输入，如果需要配置为右声道输入，

- 仅打开右声道并处理。
- 打开左右声道，按左声道处理，使用 CVI_AI_SetTrackMode 配置 Audio Input 声道模式为 “AUDIO_TRACK_EXCHANGE”。

对 AO 来说，单声道默认从左声道输出，如果需要配置为右声道输出，可以考虑两种方法：

- 仅打开右声道并处理。
- 打开左右声道，按左声道处理，使用 CVI_AO_SetTrackMode 配置 AO 声道模式为 “AUDIO_TRACK_EXCHANGE”。

对于双声道模式，只应对左声道（即编号小于设备属性中通道数 u32ChnCnt 一半的通道）进行操作，SDK 内部会自动对右声道也进行相应的操作。

【相关数据类型及接口】

AIO_ATTR_S

10.4.1.19 AUDIO_MOD_PARAM_S

【说明】

定义音频模块参数结构体。

【定义】

```
typedef struct cviAUDIO_MOD_PARAM_S {
  AUDIO_CLKSEL_E enClkSel;
} AUDIO_MOD_PARAM_S;
```

【成员】

enClkSel 音频时钟源选择。请参见AUDIO_CLKSEL_E。

【注意事项】

Cvitek 无须针对 CLK 做特殊设定

【相关数据类型及接口】

无。

10.4.1.20 AIO_ATTR_S

【说明】

定义音频输入/输出设备属性结构体。

【定义】

```
typedef struct _AIO_ATTR_S {
  AUDIO_SAMPLE_RATE_E enSamplerate; /*sample rate*/
  AUDIO_BIT_WIDTH_E enBitwidth;   /*bitwidth*/
  AIO_MODE_E   enWorkmode;   /*master or slave mode*/
  AUDIO_SOUND_MODE_E enSoundmode; /*momo or steror*/
  CVI_U32     u32EXFlag;   /*expand 8bit to 16bit */
  CVI_U32     u32FrmNum;   /*frame num in buffer*/
  CVI_U32     u32PtNumPerFrm; /*number of samples*/
  CVI_U32     u32ChnCnt;
  CVI_U32     u32ClkSel;
  AIO_I2STYPE_E       enI2sType;
}AIO_ATTR_S;
```

【成员】

成员名称	描述
enSamplerate	音频采样率（从模式下，此参数不起作用）。静态属性。
enBitwidth	音频采样精度（从模式下，此参数必须和音频 AD/DA 的采样精度匹配）。静态属性。
enWorkmode	音频输入/输出工作模式。静态属性。
enSoundmode	音频声道模式。静态属性。
u32EXFlag	取值范围：{0, 1, 2}。 0：不扩展。 1：扩展成 16 位，8bit 到 16bit 扩展标志（只对 Audio Input 采样精度为 8bit 时有效）。 2：24 位裁剪成 16 位，在外置 Codec 的场景下可能用到。 静态属性，保留参数，一般设置成 1 即可。
u32FrmNum	区块帧数目。
u32PtNumPerFrm	每帧的采样点个数。 取值范围：G711、G726、ADPCM_DVI4 编码时取值为 160、320、480；
u32ChnCnt	支持的通道数目。
enI2sType	配置设备 I2S 类型，Cvitek 仅支持主模式。
Audio Input 取值	AIO_I2STYPE_INNERCODEC AIO_I2STYPE_EXTERN
AO 取值	AIO_I2STYPE_INNERCODEC AIO_I2STYPE_EXT ERN， AIO_I2STYPE_INNERHDMI。

【注意事项】

每帧的采样点个数 u32PtNumPerFrm 和采样率 enSamplerate 的取值决定了硬件产生中断的频率，频率过高会影响系统的性能，跟其他业务也会相互影响，

建议这两个参数的取值满足算式：(u32PtNumPerFrm*1000)/ enSamplerate>=10，

比如在采样率为 16000Hz 时，建议设置采样点个数大于或者等于 160。

【相关数据类型及接口】

CVI_AI_SetPubAttr 和 CVI_AO_SetPubAttr

10.4.1.21 AI_CHN_PARAM_S

【说明】

定义通道参数结构体。

【定义】

```
typedef struct _AI_CHN_PARAM_S {
  CVI_U32 u32UsrFrmDepth;
} AI_CHN_PARAM_S;
```

【成员】

u32UsrFrmDepth: 音频帧区块深度。

【注意事项】

无。

【相关数据类型及接口】

无。

10.4.1.22 AUDIO_FRAME_S

【说明】

定义音频帧结构体。

【定义】

```
typedef struct _AUDIO_FRAME_S {
  AUDIO_BIT_WIDTH_E  enBitwidth;
  AUDIO_SOUND_MODE_E  enSoundmode;
  CVI_U64  u64VirAddr [2];
  CVI_U64  u64TimeStamp;
  CVI_U32  u32Seq;
  CVI_U32  u32Len;
  CVI_U32  u32PoolId[2];
} AUDIO_FRAME_S;
```

【成员】

成员名称	描述
enBitwidth	音频采样精度。
enSoundmode	音频声道模式。
u64VirAddr [2]	音频帧数据虚拟地址。
u64PhyAddr[2]	音频帧数据物理地址。目前不支持
u64TimeStamp	音频帧时间戳。以 μs 为单位。
u32Seq	音频帧序号。
u32Len	音频帧长度：单一通道总取样量。 samples(单位取样量) 为单位。 1 sample = 2 bytes。 Ex. AIO_ATTR_S 参数设置 u32FrmNum = 320, u32ChnCnt = 2. 则 *u32Len = 320(samples/channel). u64VirAddr [0] buffer 内含有 bytes 数目应为 (u32Len x u32ChnCnt x 2) |
u32PoolId[2]	音频帧区块池 ID。

【注意事项】

u32Len（音频帧长度）指单个声道的数据长度。

u64VirAddr [0]，长度为 bytes：(u32Len x bytes_per_sample)；

单声道默认为左声道, 资料排列为 [左, 左, 左, 左, 左,…] 。

立体声数据按左右声道, 资料排列为 [左, 右, 左, 右, 左, 右…] 。

(注：左代表左声道单一 sample, 右左声道单一 sample)

u64VirAddr [1]，无存放数据，可供客制化使用。

【相关数据类型及接口】

无。

10.4.1.23 AEC_FRAME_S

【说明】

定义音频回声抵消参考帧信息结构体。

【定义】

```
typedef struct _AEC_FRAME_S {
  AUDIO_FRAME_S stRefFrame;  /* aec reference audio frame */
  CVI_BOOL  bValid;   /* whether frame is valid */
  CVI_BOOL  bSysBind; /* whether is sysbind */
}AEC_FRAME_S;
```

【成员】

成员名称	描述
stRefFrame	回声抵消参考帧结构体。
bValid	参考帧有效的标志。 取值范围： CVI_TRUE：参考帧有效。 CVI_FALSE：参考帧无效， 无效时不能使用此参考帧进行回声抵消。
bSysBind	Audio Input 和 AENC 是否系统绑定。

【注意事项】

无。

【相关数据类型及接口】

无。

10.4.1.24 AUDIO_AGC_CONFIG_S

【说明】

定义音频自动增益控制配置信息结构体。

【定义】

```
typedef struct _AUDIO_AGC_CONFIG_S {
  CVI_S8 para_agc_max_gain;
  CVI_S8 para_agc_target_high;
  CVI_S8 para_agc_target_low;
  CVI_BOOL para_agc_vad_ena;
} AUDIO_AGC_CONFIG_S;
```

【成员】

成员名称	描述
para_agc_max_gain	信号可以被放大的最大增益。
para_agc_target_high:	AGC 将会去达到的 “Target High” 水平。
para_agc_target_low	AGC 将会去达到的 “Target Low” 水平。
para_agc_vad_enable	Speech-activated AGC 功能应用了来自 NR 的 Speech VAD 以避免放大背景噪声。 建议在处于高 SNR 的环境中开启此功能，而当处于中/低 SNR 环境中最好关闭此功能以获得较佳的语音品质

【注意事项】

无。

【相关数据类型及接口】

- AI_VQE_CONFIG_S
- AO_VQE_CONFIG_S

10.4.1.25 AI_AEC_CONFIG_S

【说明】

定义音频回声抵消配置信息结构体。

【定义】

```
typedef struct _AI_AEC_CONFIG_S {
  CVI_U16 para_aec_filter_len;
  CVI_U16 para_aes_std_thrd;
  CVI_U16 para_aes_supp_coeff;
} AI_AEC_CONFIG_S;
```

【成员】

成员名称	描述
para_aex_filter_len	自适应滤波器的长度
para_aes_std_thrd	残留回声判断阈值
para_aes_supp_coeff	残留回声抑制力道

【注意事项】

当用户模式开启时，其他参数才生效，

否则按照 AI_VQE_CONFIG_S/ AI_TALKVQE_CONFIG_S 中的工作模式 enWorkstate 对应的默认值来配置。

配置参数时，只有在用户模式开启时，才会对高级参数做正确性检查，只有正确的高级参数才能配置成功。

【相关数据类型及接口】

- AI_VQE_CONFIG_S

10.4.1.26 AUDIO_ANR_CONFIG_S

【说明】

定义音频语音降噪功能配置信息结构体。

【定义】

```
typedef struct _AUDIO_ANR_CONFIG_S {
  CVI_S8 para_nr_snr_coeff;
  CVI_S8 para_nr_noise_coeff;
} AUDIO_ANR_CONFIG_S;
```

【成员】

成员名称	描述
para_nr_snr_coeff	Signal-to-Noise Ratio（SNR）跟踪系数。 如果将其设置为较大的值，则 NR 会具有较高的降噪能力，但语音信号可能会较容易失真; 如果选择较小的值，则 NR 将抑制较少的噪声信号，但会具有较好的语音品质性能。
para_nr_noise_coeff	噪声跟踪系数。 此参数决定平稳噪声的跟踪速度。 [0 ~ 14] 0: 最慢的噪音追踪速度 14: 最快的噪音追踪速度

【注意事项】

无。

【相关数据类型及接口】

- AI_VQE_CONFIG_S
- AO_VQE_CONFIG_S

10.4.1.27 AUDIO_DELAY_CONFIG_S

【说明】

定义音频信号延迟结构体。

【定义】

```
typedef struct _AUDIO_DELAY_CONFIG_S {
  /* the initial filter length of linear AEC to support up for echo tail, [1, 13] */
  CVI_U16 para_aec_init_filter_len;
  /* the digital gain target, [1, 12] */
  CVI_U16 para_dg_target;
  /* the delay sample for ref signal, [1, 3000] */
  CVI_U16 para_delay_sample;
} AUDIO_DELAY_CONFIG_S;
```

【成员】

成员名称	描述
para_aec_init_filter_len	自适应滤波器的长度
para_dg_target	Digital Gain。范围 [1 –12] 此功能有助于降低 residual echo 及 residual stationary noise.
para_delay_sample	用于延迟参考信号 范围 [1-3000] 可使 AEC/AES 加速收敛一开始出现的回声。

【注意事项】

无。

【相关数据类型及接口】

- AI_TALKVQE_CONFIG_S

10.4.1.28 VQE_WORKSTATE_E

【说明】

定义声音质量增强的工作模式。

【定义】

```
typedef enum _VQE_WORKSTATE_E {
  VQE_WORKSTATE_COMMON = 0,
  VQE_WORKSTATE_MUSIC  = 1,
  VQE_WORKSTATE_NOISY  = 2
} VQE_WORKSTATE_E;
```

【成员】

成员名称	描述
VQE_WORKSTATE_COMMON	一般模式。
VQE_WORKSTATE_MUSIC	音乐模式。
VQE_WORKSTATE_NOISY	噪声模式。

【注意事项】

无。

【相关数据类型及接口】

- AI_VQE_CONFIG_S
- AO_VQE_CONFIG_S

10.4.1.29 VQE_RECORD_TYPE

【说明】

定义录音类型。

【定义】

```
typedef enum _VQE_RECORD_TYPE {
  VQE_RECORD_NORMAL      = 0,
  VQE_RECORD_BUTT, }
  VQE_RECORD_TYPE;
```

【成员】

VQE_RECORD_NORMAL: 标准类型。

【注意事项】

Cvitek 仅支持 talk VQE，record VQE 除非客制化，目前无使用。

【相关数据类型及接口】

- AI_RECORDVQE_CONFIG_S

10.4.1.30 AI_TALKVQE_CONFIG_S

【说明】

定义音频输入声音质量增强（Talk）配置信息结构体。

【定义】

```
typedef struct _AI_TALKVQE_CONFIG_S {
  CVI_U16   para_client_config;
  CVI_U32   u32OpenMask;
  CVI_S32   s32WorkSampleRate;
  /* Sample Rate: 8KHz/16KHz. default: 8KHz*/
  //MIC IN VQE setting
  AI_AEC_CONFIG_S     stAecCfg;
```

（下页继续）

(续上页)

```
  AUDIO_ANR_CONFIG_S  stAnrCfg;
  AUDIO_AGC_CONFIG_S  stAgcCfg;
  AUDIO_DELAY_CONFIG_S stAecDelayCfg;
  CVI_S32 s32RevMask;//turn this flag to default 0x11
  CVI_S32 para_notch_freq;//user can ignore this flag
  CVI_CHAR customize[MAX_AUDIO_VQE_CUSTOMIZE_NAME];
} AI_TALKVQE_CONFIG_S;
```

【成员】

成员名称	描述
para_client_config	客户端参数配置
u32OpenMask	Talk Vqe 的各功能使能的 Mask 值。
s32WorkSampleRate	工作采样频率。 该参数为内部功能算法工作采样率。 取值范围：8KHz/16KHz。 默认值为 8KHz。
stAecCfg	回声抵消功能相关配置信息。
stAnrCfg	语音降噪功能相关配置信息。
stAgcCfg	自动增益控制相关配置信息。
stAecDelayCfg	信号延迟相关配置信息
s32RevMask	客制化遮蔽值设定。
para_notch_freq	客制化频率消除。
customize	客制化参数选择

【注意事项】

Cvitek VQE 仅支持 AGC/ANR/AEC 其余不支持

例如:RNR/EQ 相关数据设入后，并不会有效果

【相关数据类型及接口】

无。

10.4.1.31 AI_RECORDVQE_CONFIG_S

【说明】

定义音频输入声音质量增强（Record）配置信息结构体。

【定义】

```
typedef struct _AI_RECORDVQE_CONFIG_S {
  CVI_U32   u32OpenMask;
  CVI_S32           s32WorkSampleRate;
  CVI_S32           s32FrameSample;
  VQE_WORKSTATE_E    enWorkstate;
  CVI_S32      s32InChNum;
  CVI_S32     s32OutChNum;
  VQE_RECORD_TYPE      enRecordType;
```

(下页继续)

(续上页)

```
  AUDIO_AGC_CONFIG_S  stAgcCfg;
} AI_RECORDVQE_CONFIG_S;
```

【成员】

成员名称	描述
u32OpenMask	Record Vqe 的各功能使能的 Mask 值。
s32WorkSampleRate	工作采样频率。 该参数为内部功能算法工作采样率。 取值范围：16KHz 或者 48KHz
s32FrameSample	VQE 的帧长，即采样点数目。支持范围 [80, 4096]。
enWorkstate	工作模式。
s32InChNum	VQE 处理的输入通道数目。取值范围：[1, 2]。
s32OutChNum	VQE 处理的输出通道数目。取值范围：[1, 2]。
enRecordType	录音类型。
stAgcCfg	自动增益控制相关配置信息。

【注意事项】

Cvitek VQE 仅支持 AGC/ANR/AEC 其余不支持

例如:RNR/EQ 相关数据设置后，并不会有效果

【相关数据类型及接口】

无。

10.4.1.32 AO_VQE_CONFIG_S

【说明】

定义音频输出声音质量增强配置信息结构体。

【定义】

```
typedef struct _AO_VQE_CONFIG_S {
  CVI_U32    u32OpenMask;
  CVI_S32 s32WorkSampleRate;
  /* Sample Rate: 8KHz/16KHz default: 8KHz*/
  AUDIO_SPK_AGC_CONFIG_S stAgcCfg;
  AUDIO_SPK_EQ_CONFIG_S stEqCfg;
} AO_VQE_CONFIG_S;
```

【成员】

成员名称	描述
u32OpenMask	AO Vqe 的各功能使能的 Mask 值。
s32WorkSampleRate	工作采样频率。 该参数为内部功能算法工作采样率。 取值范围：8KHz/16KHz/48KHz。 默认值为 8KHz。(仅 Hpf 支持 48KHz)
stAgcCfg	自动增益控制相关配置信息。
stEqCfg	语音讯号均衡处理配置

【注意事项】

不支持

【相关数据类型及接口】

无。

10.4.1.33 AUDIO_STREAM_S

【说明】

定义音频码流结构体。

【定义】

```
typedef struct _AUDIO_STREAM_S {
  CVI_U8 *pStream;        /* the virtual address of stream */
  CVI_U32 u32PhyAddr;     /* the physics address of stream */
  CVI_U32 u32Len;         /* stream lenth, by bytes */
  CVI_U64 u64TimeStamp;   /* frame time stamp*/
  CVI_U32 u32Seq;     /* frame seq,if stream is not a valid frame, u32Seq is 0*/
} AUDIO_STREAM_S;
```

【成员】

成员名称	描述
pStream	音频码流数据指针。
u32PhyAddr	音频码流的物理地址。
u32Len	音频码流长度。AUDIO_STREAM_S 结构体内，以 byte 为单位。
u64TimeStamp	音频码流时间戳。
u32Seq	音频码流序号。

【注意事项】

无。

【相关数据类型及接口】

CVI_AENC_GetStream

10.4.1.34 AO_CHN_STATE_S

【说明】

音频输出通道的数据区块状态结构体。

【定义】

```
typedef struct hiAO_CHN_STATE_S {
  CVI_U32              u32ChnTotalNum;
  CVI_U32              u32ChnFreeNum;
  CVI_U32              u32ChnBusyNum;
} AO_CHN_STATE_S;
```

【成员】

成员名称	描述
u32ChnTotalNum	输出通道总的区块数。
u32ChnFreeNum	可用的空闲区块数。
u32ChnBusyNum	被占用区块数。

【注意事项】

无。

【相关数据类型及接口】

CVI_AO_QueryChnStat

10.4.1.35 AUDIO_TRACK_MODE_E

【说明】

定义音频设备声道模式类型。

【定义】

```
typedef enum hiAUDIO_TRACK_MODE_E {
  AUDIO_TRACK_NORMAL     = 0,
  AUDIO_TRACK_BOTH_LEFT  = 1,
  AUDIO_TRACK_BOTH_RIGHT = 2,
  AUDIO_TRACK_EXCHANGE   = 3,
  AUDIO_TRACK_MIX        = 4,
  AUDIO_TRACK_LEFT_MUTE  = 5,
  AUDIO_TRACK_RIGHT_MUTE = 6,
  AUDIO_TRACK_BOTH_MUTE  = 7,
  AUDIO_TRACK_BUTT } AUDIO_TRACK_MODE_E;
```

【成员】

成员名称	描述
AUDIO_TRACK_NORMAL	正常模式，不做处理
AUDIO_TRACK_BOTH_LEFT	两个声道全部为左声道声音
AUDIO_TRACK_BOTH_RIGHT	两个声道全部为右声道声音
AUDIO_TRACK_EXCHANGE	左右声道数据互换，左声道为右声道声音，右声道为左声道声音
AUDIO_TRACK_MIX	左右两个声道输出为左右声道相加（混音）
AUDIO_TRACK_LEFT_MUTE	左声道静音，右声道播放原右声道声音
AUDIO_TRACK_RIGHT_MUTE	右声道静音，左声道播放原左声道声音
AUDIO_TRACK_BOTH_MUTE	左右声道均静音

【注意事项】

无。

【相关数据类型及接口】

- CVI_AI_SetTrackMode
- CVI_AO_SetTrackMode

10.4.1.36 AUDIO_FADE_RATE_E

【说明】

定义音频输出设备淡入淡出速度类型。

【定义】

```
typedef enum _AUDIO_FADE_RATE_E {
  AUDIO_FADE_RATE_NONE  = 0,
  AUDIO_FADE_RATE_10 = 10,
  AUDIO_FADE_RATE_20 = 20,
  AUDIO_FADE_RATE_30 = 30,
  AUDIO_FADE_RATE_50 = 50,
  AUDIO_FADE_RATE_100 = 100,
  AUDIO_FADE_RATE_200 = 200,
  AUDIO_FADE_RATE_BUTT = -1
} AUDIO_FADE_RATE_E;
```

【成员】

成员名称	描述
AUDIO_FADE_RATE_NONE	音量递增或递减中间无延迟
AUDIO_FADE_RATE_10	音量递增或递减每 10ms 更动步阶
AUDIO_FADE_RATE_20	音量递增或递减每 20ms 更动步阶
AUDIO_FADE_RATE_30	音量递增或递减每 30ms 更动步阶
AUDIO_FADE_RATE_50	音量递增或递减每 50ms 更动步阶
AUDIO_FADE_RATE_100	音量递增或递减每 100ms 更动步阶
AUDIO_FADE_RATE_200	音量递增或递减每 200ms 更动步阶

【注意事项】

Cvitek 使用 AUDIO_FADE_RATE_E 参数时，请确认 AUDIO_FADE_S 内的 bFade 已经设为 CVI_TRUE，淡入或淡出会以目前音量值逐步依设定的 AUDIO_FADE_RATE 做时间延迟设置，直到淡入至 unmute 或是淡出至 mute。

【相关数据类型及接口】

无。

10.4.1.37 AUDIO_FADE_S

【说明】

音频输出设备淡入淡出配置结构体。

【定义】

```
typedef struct hiAUDIO_FADE_S {
  CVI_BOOL            bFade;
  AUDIO_FADE_RATE_E enFadeInRate;
  AUDIO_FADE_RATE_E enFadeOutRate;
} AUDIO_FADE_S;
```

【成员】

成员名称	描述
bFade	是否开启淡入淡出功能。 CVI_TRUE：开启淡入淡出功能。 CVI_FALSE：关闭淡入淡出功能。
enFadeInRate	音频输出设备音量淡入速度。
enFadeOutRate	音频输出设备音量淡出速度。

【注意事项】

Cvitek 请确认 AUDIO_FADE_S 内的 bFade 已经设为 CVI_TRUE，enFadeInRate/enFadeOutRate 值的设定才有所作用。

【相关数据类型及接口】

CVI_AO_SetMute

10.4.1.38 G726_BPS_E

【说明】

定义 G.726 编解码协议速率。

【定义】

```
typedef enum hiG726_BPS_E {
  G726_16K = 0,       /* G726 16kbps, see RFC3551.txt  4.5.4 G726-16 */
  G726_24K,           /* G726 24kbps, see RFC3551.txt  4.5.4 G726-24 */
  G726_32K,           /* G726 32kbps, see RFC3551.txt  4.5.4 G726-32 */
  G726_40K,           /* G726 40kbps, see RFC3551.txt  4.5.4 G726-40 */
  MEDIA_G726_16K,  /* G726 16kbps for ASF ... */
```

(下页继续)

(续上页)

```
  MEDIA_G726_24K,  /* G726 24kbps for ASF ... */
  MEDIA_G726_32K,  /* G726 32kbps for ASF ... */
  MEDIA_G726_40K,  /* G726 40kbps for ASF ... */
  G726_BUTT,
} G726_BPS_E;
```

【成员】

成员名称	描述
G726_16K	16kbps G.726。
G726_24K	24kbps G. 726。
G726_32K	32kbps G.726。
G726_40K	40kbps G.726。
MEDIA_G726_16K G726	16kbps for ASF。
MEDIA_G726_24K	G726 24kbps for ASF。
MEDIA_G726_32K	G726 32kbps for ASF。
MEDIA_G726_40K	G726 40kbps for ASF。

【注意事项】

无。

【相关数据类型及接口】

无。

10.4.1.39 ADPCM_TYPE_E

【说明】

定义 ADPCM 编解码协议类型。

【定义】

```
typedef enum hiADPCM_TYPE_E {
  ADPCM_TYPE_DVI4 = 0,
  ADPCM_TYPE_IMA,
  ADPCM_TYPE_ORG_DVI4,
  ADPCM_TYPE_BUTT,
} ADPCM_TYPE_E;
```

【成员】

成员名称	描述
ADPCM_TYPE_DVI4	32kbit/s ADPCM(DVI4)。
ADPCM_TYPE_IMA	32kbit/s ADPCM(IMA)。
ADPCM_TYPE_ORG_DVI4	32kbit/s ADPCM(ORG_DVI4)。

【注意事项】

无。

【相关数据类型及接口】

无。

10.4.1.40 ST_CVI_WAV_HEADER

【说明】

定义 WAV 标头参考结构体。

【定义】

```
typedef struct _cvi_wavHEADER {
  /* RIFF string */
  CVI_U8 riff[4];
  // overall size of file in bytes
  CVI_U32 overall_size;
  // WAVE string
  CVI_U8 wave[4];
  // fmt string with trailing null char
  CVI_U8 fmt_chunk_marker[4];
  // length of the format data
  CVI_U32 length_of_fmt;
  // format type. 1-PCM, 3- IEEE float, 6 - 8bit A law, 7 - 8bit mu law
  CVI_U16 format_type;
  // no.of channels
  CVI_U16 channels;
  // sampling rate (blocks per second)
  CVI_U32 sample_rate;
  // SampleRate * NumChannels * BitsPerSample/8
  CVI_U32 byterate;
  // NumChannels * BitsPerSample/8
  CVI_U16 block_align;
  // bits per sample, 8- 8bits, 16- 16 bits etc
  CVI_U16 bits_per_sample;
  // DATA string or FLLR string
  CVI_U8 data_chunk_header[4];
  // NumSamples * NumChannels * BitsPerSample/8 - size of the next chunk that will be read
  CVI_U32 data_size;
} ST_CVI_WAV_HEADER;
```

【成员】

此结构体仅供用户参考，并无 CVI API 作为输入变量。

【注意事项】

无。

【相关数据类型及接口】

无。

10.4.1.41 AUDIO_FILE_STATUS_S

【说明】

定义音频文件保存状态结构体。

【定义】

```
typedef struct hiAUDIO_FILE_STATUS_S {
  CVI_BOOL    bSaving;
} AUDIO_FILE_STATUS_S;
```

【成员】

成员名称	描述
bSaving	是否处于存文件状态。 CVI_TRUE：处于存文件状态； CVI_FALSE：不处于存文件状态。

【注意事项】

无。

【相关数据类型及接口】

无。

10.4.1.42 VQE_MODULE_CONFIG_S

【说明】

定义声音质量增强及重采样模块配置信息结构体。

【定义】

```
typedef struct _VQE_MODULE_CONFIG_S {
  CVI_VOID *pHandle;
} VQE_MODULE_CONFIG_S;
```

【成员】

成员名称	描述
pHandle	注册句柄。

【注意事项】

各声音质量增强及重采样模块的注册句柄，用户可通过调用句柄获取接口进行获取。

【相关数据类型及接口】

无。

10.4.1.43 AUDIO_VQE_REGISTER_S

【说明】

定义声音质量增强及重采样模块注册结构体。

【定义】

```
typedef struct _AUDIO_VQE_REGISTER_S {
  VQE_MODULE_CONFIG_S stResModCfg;
  VQE_MODULE_CONFIG_S stHpfModCfg;
  VQE_MODULE_CONFIG_S stHdrModCfg;
  VQE_MODULE_CONFIG_S stGainModCfg;

  // Record VQE
  VQE_MODULE_CONFIG_S stRecordModCfg;

  // Talk VQE
  VQE_MODULE_CONFIG_S stAecModCfg;
  VQE_MODULE_CONFIG_S stAnrModCfg;
  VQE_MODULE_CONFIG_S stAgcModCfg;
  VQE_MODULE_CONFIG_S stEqModCfg;

  // CviFi VQE
  VQE_MODULE_CONFIG_S stRnrModCfg;
  VQE_MODULE_CONFIG_S stDrcModCfg;
  VQE_MODULE_CONFIG_S stPeqModCfg;
} AUDIO_VQE_REGISTER_S;
```

【成员】

成员名称	描述
pHandle	注册句柄。

【注意事项】

目前仅支持 audio uplink 语音录音后的 Talk VQE。其余 VQE 不支持。

【相关数据类型及接口】

无。

10.4.2 音频编码

音频编码相关数据类型、数据结构定义如下：

- AENC_MAX_CHN_NUM：定义音频编码通道的最大个数。
- AENC_ATTR_G711_S：定义 G.711 编码协议属性结构体。
- AENC_ATTR_G726_S：定义 G.726 编码协议属性结构体。
- AENC_ATTR_ADPCM_S：定义 ADPCM 编码协议属性结构体。
- AENC_ATTR_LPCM_S：定义 LPCM 编码协议属性结构体。

- AENC_CHN_ATTR_S：定义音频编码通道属性结构体。
- AAC_AENC_ENCODER_S：定义编码器属性结构体。

10.4.2.1 AENC_MAX_CHN_NUM

【说明】

定义音频编码通道的最大个数。

【定义】

```
#define AENC_MAX_CHN_NUM      3
```

【注意事项】

- ADEC_MAX_CHN_NUM

【相关数据类型及接口】

无。

10.4.2.2 AENC_ATTR_G711_S

【说明】

定义 G.711 编码协议属性结构体。

【定义】

```
typedef struct hiAENC_ATTR_G711_S
{
  CVI_U32 resv;
}AENC_ATTR_G711_S;
```

【成员】

成员名称	描述
resv	无使用

【注意事项】

无。

【相关数据类型及接口】

无。

10.4.2.3 AENC_ATTR_G726_S

【说明】

定义 G.726 编码协议属性结构体。

【定义】

```
typedef struct _AENC_ATTR_G726_S {
```

【成员】

成员名称	描述
enG726bps	G.726 协议码率。

【注意事项】

无。

【相关数据类型及接口】

G726_BPS_E

10.4.2.4 AENC_ATTR_ADPCM_S

【说明】

定义 ADPCM 编码协议属性结构体。

【定义】

```
typedef struct _AENC_ATTR_ADPCM_S {
  ADPCM_TYPE_E enADPCMType;
}AENC_ATTR_ADPCM_S;
```

【成员】

成员名称	描述
enADPCMType	ADPCM 类型。

【注意事项】

无。

【相关数据类型及接口】

ADPCM_TYPE_E

10.4.2.5 AENC_ATTR_LPCM_S

【说明】

定义 LPCM 编码协议属性结构体。

【定义】

```
typedef struct _AENC_ATTR_LPCM_S {
  CVI_U32 resv;          /*reserve item*/
}AENC_ATTR_LPCM_S;
```

【成员】

此结构内部变量无使用。

【注意事项】

无。

【相关数据类型及接口】

无。

10.4.2.6 AENC_CHN_ATTR_S

【说明】

定义音频编码通道属性结构体。该结构体的定义在不同处理器平台上略有不同。

【定义】

```
typedef struct _AENC_CHN_ATTR_S {
  PAYLOAD_TYPE_E enType;
  CVI_U32  u32PtNumPerFrm;
  CVI_U32 u32BufSize;
  CVI_VOID *pValue;
  CVI_BOOL bFileDbgMode;
}AENC_CHN_ATTR_S;
```

【成员】

成员名称	描述
enType	音频编码协议类型，静态属性。
u32PtNumPerFrm	音频编码协议对应的帧长
u32BufSize	音频编码区块大小。
pValue	具体协议属性指针。
bFileDbgMode	是否存文件状态

【注意事项】

无。

【相关数据类型及接口】

无。

10.4.2.7 AAC_AENC_ENCODER_S

【说明】

定义 AAC 编码器属性结构体。

【定义】

```
typedef struct _AAC_AENC_ENCODER_S {
  PAYLOAD_TYPE_E  enType;
  CVI_U32     u32MaxFrmLen;
  CVI_CHAR    aszName[17];
  /* encoder type,be used to print proc information */
  CVI_S32 (*pfnOpenEncoder)(CVI_VOID *pEncoderAttr, CVI_VOID **ppEncoder);
  /* pEncoder is the handle to control the encoder */
      CVI_S32  (*pfnEncodeFrm)(CVI_VOID *pEncoder, CVI_S16 * inputdata, CVI_U8 *
  pu8Outbuf,

          CVI_S32 s32InputSizeBytes, CVI_U32 *pu32OutLen);
  CVI_S32 (*pfnCloseEncoder)(CVI_VOID *pEncoder);
} AAC_AENC_ENCODER_S;
```

【成员】

此结构体仅供 AAC 外部 LIB 连结使用，此版 SDK 仅定义，尚未支持。

【注意事项】

此结构体仅供 AAC 外部 LIB 连结使用，此版 SDK 仅定义，尚未支持。如须使用 AAC 请参照 middleware/sample/audio/aac_sample 内容。

【相关数据类型及接口】

无。

10.4.3 音频解码

音频解码相关数据类型、数据结构定义如下：

- MAX_AUDIO_FRAME_NUM：定义最大音频解码区块帧数。
- ADEC_MAX_CHN_NUM：定义音频解码通道的最大个数。
- ADEC_ATTR_G711_S：定义 G.711 解码协议属性结构体。
- ADEC_ATTR_G726_S：定义 G.726 解码协议属性结构体。
- ADEC_ATTR_ADPCM_S：定义 ADPCM 解码协议属性结构体。
- ADEC_ATTR_LPCM_S：定义 LPCM 解码协议属性结构体。
- ADEC_MODE_E：定义解码方式。
- ADEC_CHN_ATTR_S：定义解码通道属性结构体。
- ADEC_CHN_STATE_S：定义音频解码通道的数据缓存状态结构体。
- ADEC_DECODER_S：定义解码器属性结构体。

10.4.3.1 MAX_AUDIO_FRAME_NUM

【说明】

定义最大音频解码区块帧数。

【定义】

```
#define MAX_AUDIO_FRAME_NUM      300
```

【注意事项】

目前音频内部缓存帧数, 由 SDK 内部决定, 因此此设置对于使用者而言并未开放亦不会有任何效力。

【相关数据类型及接口】

无。

10.4.3.2 ADEC_MAX_CHN_NUM

【说明】

定义音频解码通道的最大个数。

【定义】

```
#define ADEC_MAX_CHN_NUM      3
```

【注意事项】

目前仅支持单信道编译码。

【相关数据类型及接口】

无。

10.4.3.3 ADEC_ATTR_G711_S

【说明】

定义 G.711 解码协议属性结构体。

【定义】

```
typedef struct _ADEC_ATTR_G711_S {
  CVI_U32 resv;
}ADEC_ATTR_G711_S;
```

【成员】

此结构内变量在 cvitek 处理器下无使用

【注意事项】

无。

【相关数据类型及接口】

无。

10.4.3.4 ADEC_ATTR_G726_S

【说明】

定义 G.726 解码协议属性结构体。

【定义】

```
typedef struct _ADEC_ATTR_G726_S {
  G726_BPS_E enG726bps;
}ADEC_ATTR_G726_S;
```

【成员】

成员名称	描述
enG726bps	G.726 协议码率。

【注意事项】

无。

【相关数据类型及接口】

G726_BPS_E

10.4.3.5 ADEC_ATTR_ADPCM_S

【说明】

定义 ADPCM 解码协议属性结构体。

【定义】

```
typedef struct _ADEC_ATTR_ADPCM_S {
  ADPCM_TYPE_E enADPCMType;
}ADEC_ATTR_ADPCM_S;
```

【成员】

成员名称	描述
enADPCMType	ADPCM 类型。

【注意事项】

无。

【相关数据类型及接口】

ADPCM_TYPE_E

10.4.3.6 ADEC_ATTR_LPCM_S

【说明】

定义 LPCM 解码协议属性结构体。

【定义】

```
typedef struct _ADEC_ATTR_LPCM_S {
  CVI_U32 resv;
}ADEC_ATTR_LPCM_S;
```

【成员】

resv 待扩展用。

【注意事项】

无。

【相关数据类型及接口】

无。

10.4.3.7 ADEC_MODE_E

【说明】

定义解码方式。

【定义】

```
typedef enum _ADEC_MODE_E {
  ADEC_MODE_PACK = 0,
  ADEC_MODE_STREAM ,
  ADEC_MODE_BUTT
}ADEC_MODE_E;
```

【成员】

成员名称	描述
ADEC_MODE_PACK	Pack 方式解码。
ADEC_MODE_STREAM	stream 方式解码。

【注意事项】

pack 方式用于用户确认当前码流包为一帧数据编码结果的情况下，

解码器会直接对其进行解码，如果不是一帧，解码器会出错。

这种模式的效率比较高，在使用 AENC 模块编码的码流包如果没有破坏，均可以使用此方式解码。

stream 方式用于用户不能确认当前码流包是不是一帧数据的情况下，解码器需要对码流进行判断并区块，此工作方式的效率低下，一般用于读文件码流送解码或者不确定码流包边界的情况。

当然由于语音编码码流长度固定，很容易确定在码流中的帧边界，推荐使用 pack 方式解码。

Cvitek 仅支持 pack 模式，在不确定 stream 边界的状况下，Cvitek 会因为 frame 数不对齐而译码错误。

【相关数据类型及接口】

无。

10.4.3.8 ADEC_CHN_ATTR_S

【说明】

定义解码通道属性结构体。

【定义】

```
typedef struct _ADEC_CH_ATTR_S {
  PAYLOAD_TYPE_E enType;
  CVI_U32        u32BufSize; /*buf size[2~CVI_MAX_AUDIO_FRAME_NUM]*/
  ADEC_MODE_E    enMode;/*decode mode*/
  /* CVI_VOID ATTRIBUTE      *pValue;*/
  CVI_VOID *pValue;
  CVI_BOOL bFileDbgMode;
  //if ao not enable
  CVI_S32 s32BytesPerSample;
  CVI_S32 s32frame_size; //in samples
  CVI_S32 s32ChannelNums; // 1 or 2
  CVI_S32 s32Sample_rate;;
}ADEC_CHN_ATTR_S;
```

【成员】

成员名称	描述
enType	音频解码协议类型，静态属性。
u32BufSize	音频解码区块缓存大小。目前音频内部缓存帧数，由 SDK 内部决定，因此此设置对于使用者而言并未开放亦不会有任何效力。
enMode	解码方式，静态属性。Mode 仅支持，ADEC_MODE_PACK 模式，无法透过设定 ADEC_MODE_STREAM 自动侦测。
pValue	具体协议属性指针。
bFileDbgMode	是否开启保存文件模式。请注意在 SDK 内的 cvi_sample_audio.c 范例码，此值默认是为 True，以便做 debug 使用，使用者实际运用上应设为 False 避免因存盘占用效能或内存。
使用者仅单独使用 ADEC 模块，而未使用 AO 模块时，需透过以下变量设置告知 ADEC 模块相关参数特性。	
s32BytesPerSample	单位采样使用字节。 (位宽度 SL16, 16bits = 2 bytes, 此时单位采样使用字节应设为 2) (SDK 内的范例即使用皆为 2) 。
s32frame_size	Period sample s ize: 每次送入给 ADEC 模块的采样数。
s32ChannelNums	通道数（单声道：1\双声道：2)。
s32Sample_rate	欲译码码流的采样频率 (HZ)。

【注意事项】

无。

【相关数据类型及接口】

无。

10.4.3.9 AUDIO_FRAME_INFO_S

【说明】

定义解码后的音频帧信息结构体。

【定义】

```
typedef struct _AUDIO_FRAME_INFO_S {
  AUDIO_FRAME_S *pstFrame;
  CVI_U32        u32Id;
} AUDIO_FRAME_INFO_S;
```

【成员】

成员名称	描述
pstFrame	音频帧指针。
u32Id	音频帧的索引，范围 [0, 49]

【注意事项】

无。

【相关数据类型及接口】

- CVI_ADEC_GetFrame
- CVI_ADEC_ReleaseFrame

10.4.3.10 ADEC_CHN_STATE_S

【说明】

定义音频解码通道的数据缓存状态结构体。

【定义】

```
typedef struct _ADEC_CHN_STATE_S {
  CVI_BOOL bEndOfStream;          /* EOS flag */
  CVI_U32 u32BufferFrmNum;        /* total number of channel buffer */
  CVI_U32 u32BufferFreeNum;       /* free number of channel buffer */
  CVI_U32 u32BufferBusyNum;       /* busy number of channel buffer */
} ADEC_CHN_STATE_S;
```

【成员】

成员名称	描述
bEndOfStream	解码码流结束状态。
u32BufferFrmNum	解码通道总的缓存块数。
u32BufferFreeNum	可用的空闲缓存块数。
u32BufferBusyNum	被占用缓存块数

【注意事项】

无。

【相关数据类型及接口】

无。

10.4.3.11 ADEC_DECODER_S

【说明】

定义解码器属性结构体。

【定义】

```
typedef struct _ADEC_DECODER_S {
  PAYLOAD_TYPE_E  enType;
  CVI_CHAR   aszName[17];

  CVI_S32 (*pfnOpenDecoder)(CVI_VOID *pDecoderAttr, CVI_VOID
                            **ppDecoder);
```

(下页继续)

(续上页)

```
  CVI_S32 (*pfnDecodeFrm)(CVI_VOID *pDecoder, CVI_U8 **pu8Inbuf,
          CVI_S32 *ps32LeftByte,
          CVI_U16 *pu16Outbuf, CVI_U32 *pu32OutLen, CVI_U32
                          *pu32Chns);
  CVI_S32 (*pfnGetFrmInfo)(CVI_VOID *pDecoder, CVI_VOID *pInfo);
  CVI_S32 (*pfnCloseDecoder)(CVI_VOID *pDecoder);
  CVI_S32 (*pfnResetDecoder)(CVI_VOID *pDecoder);
} ADEC_DECODER_S;
```

【成员】

成员名称	描述
enType	解码协议类型。
aszName	解码器名称。
pfnOpenDecoder	打开解码器的函数指针。
pfnDecodeFrm	进行解码的函数指针
pfnGetFrmInfo	获取音频帧信息的函数指针。
pfnCloseDecoder	关闭解码器的函数指针。
pfnResetDecoder	清空缓存 buffer，复位解码器。

【注意事项】

此结构专用于链接外部 AAC 译码 lib，目前 AAC 译码请参阅 middleware/sample/audio/aac_sample 使用。

【相关数据类型及接口】

无。

10.4.4 内置 Codec

内置 Codec 相关数据类型、数据结构定义如下：

- cvi_vol_ctrl：定义内置 Audio Codec 音量控制结构体。

10.4.4.1 ACODEC_VOL_CTRL

【说明】

定义内置 Audio Codec 音量控制结构体。

【定义】

```
struct cvi_vol_ctrl {
/* volume control, adc range: 0x00~0x1f, 0x17F:mute. dac range: 0x00~0x0f, 0x0f:mute */
  CVI_U32 vol_ctrl;
  /* adc/dac mute control, 1:mute, 0:unmute */
  CVI_U32 vol_ctrl_mute;
};
```

【成员】

成员名称	描述
vol_ctrl	音量大小
vol_ctrl_mute	静音控制

【注意事项】

无。

【相关数据类型及接口】

无。

10.5 错误码

10.5.1 音频基础属性错误码

Cvitek 音频使用 CVI_SUCCESS/CVI_FAILURE 来代表基本的错误回传码:

```
#define CVI_SUCCESS 0

#define CVI_FAILURE (-1)

// 使用者需注意，CVI_TRUE/CVI_
↪FALSE仅拿来做基本的判断回复，并未被使用作为成功或失败的判断:

#define CVI_TRUE 1

#define CVI_FALSE 0
```

10.5.2 音频输入错误码

Cvitek 音频输入错误码，可以参照 cvi_comm_aio.h 相关定，对应错误码皆为 CVI_ERR_AI 开头:

错误代码	宏定义	描述
0xAA000001	CVI_ERR_AIO_ILLEGAL_PARAM	音频输入参数设置无效
0xAA000002	CVI_ERR_AIO_NULL_PTR	输入参数空指针错误
0xAA000003	CVI_ERR_AIO_NOT_PERM	操作不允许
0xAA000004	CVI_ERR_AIO_REGISTER_ERR	注册失败
0xA0000005	CVI_ERR_AI_INVALID_DEVID	装置 ID 不合法
0xA0000006	CVI_ERR_AI_INVALID_CHNID	频道 ID 不合法
0xA0000001	CVI_ERR_AI_ILLEGAL_PARAM	音频输入参数设置无效
0xA0000002	CVI_ERR_AI_NULL_PTR	输入参数空指针错误
0xA0000007	CVI_ERR_AI_NOT_CONFIG	参数未设置
0xA0000008	CVI_ERR_AI_NOT_SUPPORT	设定入内的参数不支持
0xA0000009	CVI_ERR_AI_NOT_ENABLED	Audio Input 在此状态下无法使能
0xA0000003	CVI_ERR_AI_NOT_PERM	设定的参数不支持
0xA000000A	CVI_ERR_AI_NOMEM	内存不足
0xA000000B	CVI_ERR_AI_NOBUF	Buffer 未设定或是 Buffer 未初始
0xA000000C	CVI_ERR_AI_BUF_EMPTY	Buffer 资料为空
0xA000000D	CVI_ERR_AI_BUF_FULL	Buffer 已满
0xA000000E	CVI_ERR_AI_SYS_NOTREADY	系统忙碌，系统尚未至可使用状态
0xA000000F	CVI_ERR_AI_BUSY	Audio Input 模块忙碌
0xA0000010	CVI_ERR_AI_VQE_ERR	VQE 模块错误
0xA0000011	CVI_ERR_AI_VQE_BUF_FULL	VQE Buffer 为空
0xA0000012	CVI_ERR_AI_VQE_FILE_UNEXIST	VQE 配置文件不存在

10.5.3 音频输出错误码

Cvitek 音频输出错误码，可以参照 cvi_comm_aio.h 相关定，对应错误码皆为 CVI_ERR_AO 开头:

错误代码	宏定义	描述
0xA1000001	CVI_ERR_AO_INVALID_DEVID	装置 ID 不合法
0xA1000002	CVI_ERR_AO_INVALID_CHNID	频道 ID 不合法
0xA1000003	CVI_ERR_AO_ILLEGAL_PARAM	音频输出参数设置无效
0xA1000004	CVI_ERR_AO_NULL_PTR	输入参数空指针错误
0xA1000005	CVI_ERR_AO_NOT_CONFIG	参数未设置
0xA1000006	CVI_ERR_AO_NOT_SUPPORT	设定入内的参数不支持
0xA1000007	CVI_ERR_AO_NOT_PERM	操作不允许
0xA1000008	CVI_ERR_AO_NOT_ENABLED	Audio Input 在此状态下无法使能
0xA1000009	CVI_ERR_AO_NOMEM	内存不足
0xA100000A	CVI_ERR_AO_NOBUF	Buffer 未设定或是 Buffer 未初始
0xA100000B	CVI_ERR_AO_BUF_EMPTY	Buffer 资料为空
0xA100000C	CVI_ERR_AO_BUF_FULL	Buffer 已满
0xA100000D	CVI_ERR_AO_SYS_NOTREADY	系统忙碌，系统尚未至可使用状态
0xA100000E	CVI_ERR_AO_BUSY	Audio Output 模块忙碌
0xA100000F	CVI_ERR_AO_VQE_ERR	AI_VQE 模块错误

10.5.4 音频编码错误码

Cvitek 音频编码错误码，可以参照 cvi_comm_aenc.h 相关定，对应错误码皆为 CVI_ERR_AENC 开头:

错误代码	宏定义	描述
0xA2000001	CVI_ERR_AENC_INVALID_DEVID	装置 ID 不合法
0xA2000002	CVI_ERR_AENC_INVALID_CHNID	频道 ID 不合法
0xA2000003	CVI_ERR_AENC_ILLEGAL_PARAM	音频编码参数设置无效
0xA2000004	CVI_ERR_AENC_EXIST	音频编码模块已开启使用
0xA2000005	CVI_ERR_AENC_UNEXIST	音频编码模块状态为不存在
0xA2000006	CVI_ERR_AENC_NULL_PTR	输入参数空指针错误
0xA2000007	CVI_ERR_AENC_NOT_CONFIG	参数未设置
0xA2000008	CVI_ERR_AENC_NOT_SUPPORT	设定入内的参数不支持
0xA2000009	CVI_ERR_AENC_NOT_PERM	操作不允许
0xA200000A	CVI_ERR_AENC_NOMEM	内存不足
0xA200000B	CVI_ERR_AENC_NOBUF	Buffer 未设定或是 Buffer 未初始
0xA200000C	CVI_ERR_AENC_BUF_EMPTY	Buffer 资料为空
0xA200000D	CVI_ERR_AENC_BUF_FULL	Buffer 已满
0xA200000E	CVI_ERR_AENC_SYS_NOTREADY	系统忙碌, 系统尚未至可使用状态
0xA200000F	CVI_ERR_AENC_ENCODER_ERR	AENC 编码错误
0xA2000010	CVI_ERR_AENC_VQE_ERR	VQE 模块错误

10.5.5 音频解码错误码

Cvitek 音频解碼错误码，可以参照 cvi_comm_adec.h 相关定，对应错误码皆为 CVI_ERR_ADEC 开头:

错误代码	宏定义	描述
0xA3000001	CVI_ERR_ADEC_INVALID_DEVID	设置 ID 不合法
0xA3000002	CVI_ERR_ADEC_INVALID_CHNID	频道 ID 不合法
0xA3000003	CVI_ERR_ADEC_ILLEGAL_PARAM	音频编码参数设置无效
0xA3000004	CVI_ERR_ADEC_EXIST	音频解码模块已开启使用
0xA3000005	CVI_ERR_ADEC_UNEXIST	音频解码模块状态为不存在
0xA3000006	CVI_ERR_ADEC_NULL_PTR	输入参数空指针错误
0xA3000007	CVI_ERR_ADEC_NOT_CONFIG	参数未设置
0xA3000008	CVI_ERR_ADEC_NOT_SUPPORT	设定入内的参数不支持
0xA3000009	CVI_ERR_ADEC_NOT_PERM	操作不允许
0xA300000A	CVI_ERR_ADEC_NOMEM	内存不足
0xA300000B	CVI_ERR_ADEC_NOBUF	Buffer 未设定或是 Buffer 未初始
0xA300000C	CVI_ERR_ADEC_BUF_EMPTY	Buffer 资料为空
0xA300000D	CVI_ERR_ADEC_BUF_FULL	Buffer 已满
0xA300000E	CVI_ERR_ADEC_SYS_NOTREADY	系统忙碌, 系统尚未至可使用状态
0xA300000F	CVI_ERR_ADEC_DECODER_ERR	ADEC 解码错误
0xA3000010	CVI_ERR_ADEC_BUF_LACK	ADE C 解码输入缓存空间不足

10.6 相关测试

10.6.1 单元测试

测试目的: 测试 Audio AEC 功能
测试模块: Audio VQE - AEC
测试方式: sample_audio_aec $(filename)
说明: 使用者可以将经过回声状态下采集之音档 (raw or.wav)， 透过 cvi_aec_test 程序，将原始文件转换成经过回声消除后的音档。 请注意，该功能仅支持 8kHz、16kHz 采样率，其格式必须为 S16LE。

测试目的: 测试 Audio 编解码功能
测试模块: Audio Encode/Decode
测试方式: sample_audio_transcode $(filename)
说明: 使用者可以透过 sample_audio_transcode，将 audio raw data 与 g.711/g.726 做档案间的格式转换。 请注意，该功能仅支持 8kHz、16kHz 采样率，其格式必须为 S16LE。

10.6.2 功能测试

测试目的: 测试 Audio 录音功能
测试模块: Audio In
测试方式: sample_audio 4
./sample_audio 4 --list -r 8000 -R 8000 -c 2 -p 320 -C 0 -V 0 -FCvi_8k_2chn.raw -T 10

测试目的: 测试 Audio 播音功能
测试模块: Audio Out
测试方式: sample_audio 5
./sample_audio 5 --list -r 8000 -R 8000 -c 2 -p 320 -C 0 -V 0 -FCvi_8k_2chn.raw -T 10

测试目的: 测试 Audio 重采样
测试模块: Audio Resample
测试方式: sample_audio_resample (输入 _raw 格式檔) (输入檔采样率)(目标采样率)
说明: 使用范例如下: sample_audio_resample record.raw 16000 48000 如上所示，使用者依序输入 raw 格试档案，目前档案的采样率及目标采样率， 程序会依据 libcvi_RES1.so 内的 API 进行档案的重采样动作， 结束后会产出以输出采样率为开头的 raw 檔。

测试目的: Audio 输出音量设置及测试
测试模块: Audio Output Volume(DAC codec)
测试方式:(设置音量): sample_audio 6 (获取目前输出音量): sample_audio 8
说明: 用户可透过 sample_audio 6 设置音量.

10.6.3 性能测试

测试目的: 测试 Audio 音质增进功能测试，依据参数调适，衡量语音算法对应能力。
测试模块: Audio VQE
测试方式: sample_audio_nr $(filename)
说明: 键入 sample_audio_nr 并输入.wav 档案，会依序询问 VQE 相关开关及功能参数， 支持 NR(语音降噪功能) 及 AGC(自动增益控制)。 使用范例: ---------- Enter NR off:0 , On:1 : 1 ---------- Enter AGC off:0 , On:1 : 1 pstAiVqeAttr.u32OpenMask[0x0C] Enter agc_max_gain [0, 6] 1 Enter agc_target_high[0, 36] 2 Enter agc_target_low [0, 36] 6 Enter agc_vad_enable [0, 1] 1 Enter agc_vad_cnt [1, 25] 13 Enter agc_cut6_enable [0, 1] 1 AGC param: [1, 2, 6, 1, 13, 1] Enter nr_snr_coeff [0, 20] 15 Enter nr_noise_coeff [0, 14] 2 如上所示，AGC(自动增益控制) 有六组参数可供设定， NR(语音降噪功能) 有两组参数， 使用者可参考章节 9.2.2 调配适合当下环境的参数， 程序结束后会依据输入档名，产出 NR_AGC_ 等字样开头的檔名。 使用者可对该档进行拨音或放至计算机上分析音频结果。

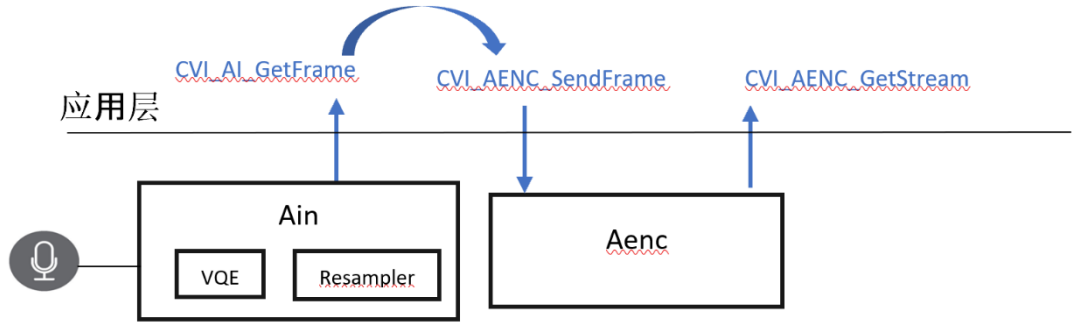
10.7 范例码及板端初步测试

10.7.1 范例码说明

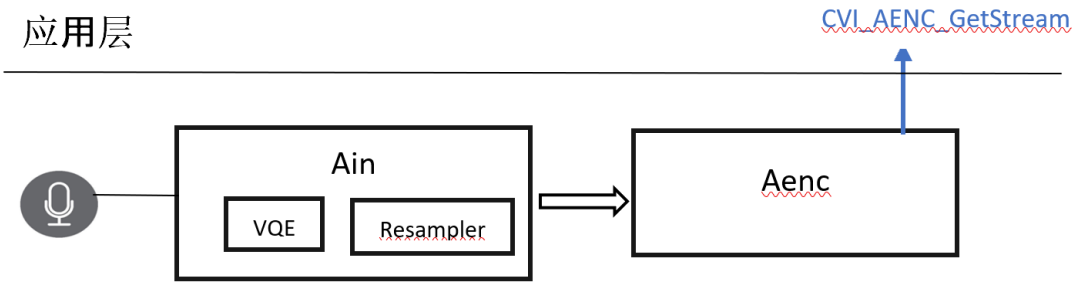
User-Get Mode :	优点: SDK 使用者可以拿到并侧录每个 audio frame 状态。 使用者在应用层可以 drop / delay / copy 等操作。 缺点: SDK 使用者在应用层需自行开线程做 get / send 动作。
Bind Mode :	优点: SDK 用户只要创建完通道参数, 只须负责拿取或播送资料。缺点: 应用层无法调配及观测目前状态。

Uplink Audio (收音-> 编码):

参照码源: cvi_sample_audio.c : SAMPLE_AUDIO_AiAenc



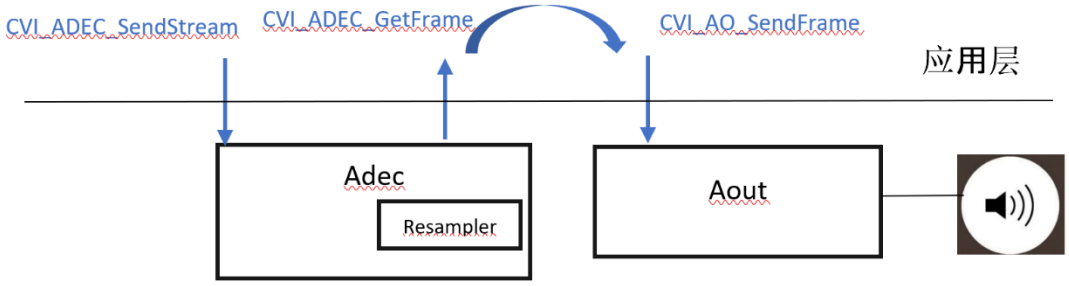
User-Get Mode 示意图如上



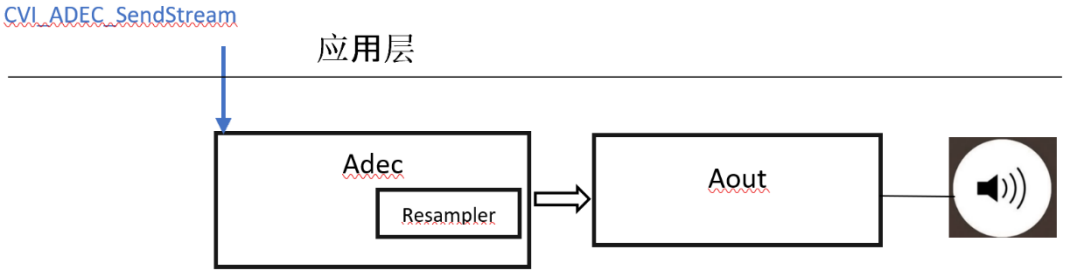
Bind Mode 示意图如上

Downlink Audio (解碼-> 播音):

参照码源: cvi_sample_audio.c : SAMPLE_AUDIO_AdecAo



User-Get Mode 示意图如上



Bind Mode 示意图如上

10.7.2 板端初步测试

随项目释出的版本不同，如须与应用工程人员同步时，音频版本信息显得相当重要，用户拿到 SDK 软件后，可以透过音频相关软件得知版本以及板子音频状态。sample_audio 执行程序所对应的内容即为 cvi_sample_audio.c 内的实做。

客制化板子初期，使用者可以透过下列方式先做双声道录音，确认储存的档案有音频波形，确认输入端正确，并可在录音后直接双声道播音确认喇叭运作正常。

[了解你的 Audio SDK 版本]:

sample_audio 7

console 会显示如下:

version [cviaudio_2021_tinyalsa0407A]

其中 [] 内所显示的即为音频版本信息。

[录音范例码]:

sample_audio 19

对应范例码流程: cvi_sample_audio.c ,case 19。
依序提示使用者录音时的参数设定，使用者可以直接键入 enter 使用默认值 (如下图)。

当 console 画面出现:

————————————>start recording…

开始依据使用者设定秒数录音，并存盘为 sample_record.raw

```
[root@cvitek]/tmp# sample_audio 19
User console mode
cvi_sample_audio:Enter command id =[19]
start register AAC encoder xxxxxxxxxxxxxx
PT_AAC[37]
entype[37]
u32MaxFrmLen[100]
start register AAC encoder end
start register AAC encoder xxxxxxxxxxxxxx
PT_AAC[37]
start register AAC decoder end
[sample code]recording frame by frame
====================================
Enter sample rate(default:16000)

input default val[16000]
====================================
====================================
Enter channel numbers(1 or 2)(default:2)

input default val[2]
====================================
====================================
Enter period size(samples per frame)(default:320)

input default val[320]
====================================
====================================
VQE on? 0:No  1:Yes (default:0)

input default val[0]
====================================
[cviaudio][dbg] cyclebuffer init dev[0] chn[0] sizebytes[12800]
[cviaudio]  CVI_AI_Enable AiDevId[0]
[cviaudio][dbg] setting PCM_IN
[cviaudio]  PCM_FORMAT_S16_LE
[cviaudio][dbg] pcm in open success card[0]
[cviaudio][dbg] ain period bytes[1280]
[cviaudio] [_tinyalsa_prepare_pcm]arecord success..parsin VQE status
stThreadCfg.DevId [0]
stThreadCfg.pAlsaConfig card id [0]
[cviaudio][dbg] create ain output thread ok.dev[0]
====================================
How many seconds you want to record(default:10s)
[cviaudio][dbg] [AudioInputThread]card id[0]devid[0]

input default val[10]
====================================
---------------------------------->start recording...
```

[播音范例码]:

sample_audio 20 檔名.raw

对应范例码流程: cvi_sample_audio.c ,case 20。

依序提示使用者播音时的参数设定，使用者可以直接键入 enter 使用默认值 (如下图)。

当 console 画面出现:

————————————>start playing…

开始依据用户配置文件名播音。

```
[root@cvitek]/tmp# sample_audio 20 sample_record.raw
User option mode...parsing user option....
parsing filename
cvg->filename[sample_record.raw]
cvi_sample_audio:Enter command id =[20]
start register AAC encoder xxxxxxxxxxxxxx
PT_AAC[37]
entype[37]
u32MaxFrmLen[100]
start register AAC encoder end
start register AAC encoder xxxxxxxxxxxxxx
PT_AAC[37]
start register AAC decoder end
[sample code]playing audio frame by frame
[SAMPLE_AUDIO_SEND_AUDIO_FRAME_BY_FRAME]filename:sample_record.raw
Play raw format file(Not a wav file)
====================================
Enter channel numbers(1 or 2)(default:2)

input default val[2]
====================================
====================================
Enter sample rate(default:16000)

input default val[16000]
====================================
====================================
Enter period size(samples per frame)(default:960)

input default val[960]
====================================
[cviaudio] AoDevId[0] in CVI_AO_Enable
[cviaudio][dbg] 16
[cviaudio][dbg] create ao output thread ok.
CVI_AO_EnableChn ------------> go
[cviaudio][dbg] CVI_AO_EnableChn DevId:0,chn:0.
--------------------------------->start playing...
[cviaudio][dbg] output_thread in sleepus:30000,period:960,rate:16000,ch:2.
```

音频释出动态链接文件 (Audio Release SO file)，请确保项目释出的 SDK 内有以下 lib 以保证音频 SDK API 完整性：

名称	功能
libcvi_audio.so	Audio API 对接层
libcvi_RES1.so	Audio 重采样模块
libcvi_VoiceEngine.so	Audio 编解碼模块
libcvi_vqe.so	Audio 算法对接层
libssp.so	Audio SSP(含 AEC) 算法
libaec.so	Audio AEC 算法
libtinyalsa.so	Audio 音频录音播音 ALSA 层
libaacenc2.so	AAC 编码相关-正式文件请洽 FAE 人员
libaacdec2.so	AAC 译码相关-正式文件请洽 FAE 人员

11 几何畸变矫正子系统

11.1 功能概述

几何畸变矫正子系统 (Geometric Distortion Correction Subsystem, 简称 GDC) 提供了对一帧图做鱼眼校正，旋转以及仿射映像等功能。

这个系统是因应镜头因其特性，进入感光组件的影像会有扭曲位移等现象，让用户可以对影像做校正，该子系统包含了图像旋转和镜头畸变矫正功能。

11.2 设计概述

GDC 模块使用 JOB 作为管理 TASK 的结构。一个 JOB 可以包含复数 TASK。

GDC 保证 TASK 按照添加到 JOB 的顺序执行。

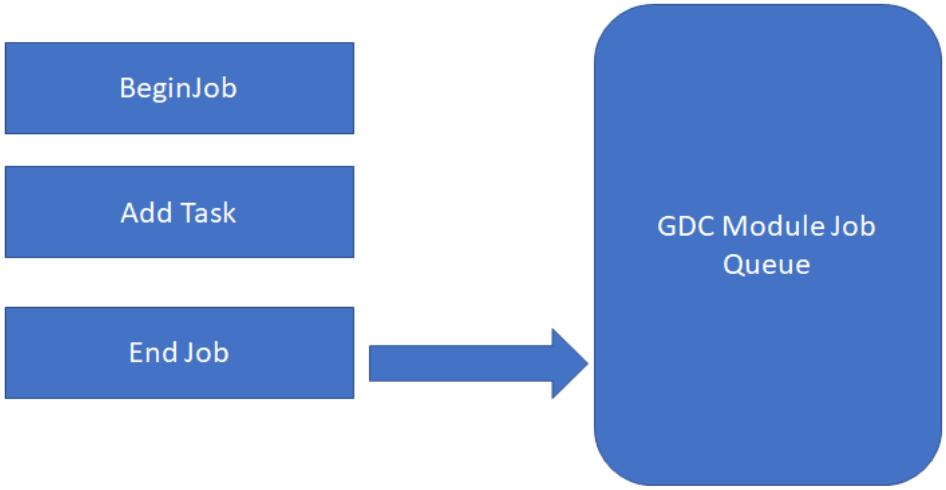
在 END 时会提交其下所有的 TASK，并把资源释放出来。

若是 TASK 提交失败，必须用 Cancel JOB 以释放资源。(GDC driver code 已经内建处理，使用者无需自行处理)

而 TASK 则用于对一幅图像完成具体的一个或多个操作，比如旋转，镜头矫正等。

11.2.1 系统架构

GDC 的工作模式采用先至先服务规则。



11.3 API 参考

11.3.1 CVI_GDC_BeginJob

【描述】

开始一个 Job。

【语法】

```
CVI_S32 CVI_GDC_BeginJob(GDC_HANDLE *phHandle);
```

【参数】

参数名称	描述	输入/输出
phHandle	取得 HANDLE	输出

【返回值】

返回值	描述
0	成功。
非 0	失败, 请参见错误码 。

【需求】

- 头文件：cvi_gdc.h
- 库文件：libvpu.so

【注意】

- 可同时开启多个 job，比如 VI 进行做 LDC job 的时候，VO 也可同时做旋转 job，但必须判断CVI_GDC_BeginJob 函数返回成功后才能使用 phHandle 返回的 HANLDE。
- phHandle 不能为空指针或非法指针。

【举例】

Please refer to CVI_GDC_AddRotationTask

【相关主题】

CVI_GDC_EndJob

11.3.2 CVI_GDC_EndJob

【描述】

结束一个 job, 所有在此 job 中的 Task 会被提交到 GDC 模块

【语法】

```
CVI_S32 CVI_GDC_EndJob(GDC_HANDLE hHandle);
```

【参数】

参数名称	描述	输入/输出
hHandle	已开启的 Job Handle	输入

【返回值】

返回值	描述
0	成功。
非 0	失败, 请参见错误码 。

【需求】

- 头文件：cvi_gdc.h
- 库文件：libvpu.so

【注意】

- hHandle 必须为已经开启的 Job
- hHandle 不能为空指针或非法指针。

【举例】

Please refer to CVI_GDC_AddRotationTask

【相关主题】

CVI_GDC_BeginJob

11.3.3 CVI_GDC_CancelJob

【描述】

取消一个 job, 所有在此 job 中的 Task 不会被提交到 GDC 模块

【语法】

```
CVI_S32 CVI_GDC_CancelJob(GDC_HANDLE hHandle);
```

【参数】

参数名称	描述	输入/输出
hHandle	已开启的 Job Handle	输入

【返回值】

返回值	描述
0	成功。
非 0	失败, 请参见错误码 。

【需求】

- 头文件：cvi_gdc.h
- 库文件：libvpu.so

【注意】

- hHandle 必须为已经开启的 Job
- phHandle 不能为空指针或非法指针。

【相关主题】

CVI_GDC_BeginJob

11.3.4 CVI_GDC_AddRotationTask

【描述】

在既有 Job 中安插一个旋转的 Task，不支持旋转 180 度。

【语法】

```
CVI_S32 CVI_GDC_AddRotationTask(GDC_HANDLE hHandle, const GDC_TASK_ATTR_S␣
↪*pstTask, ROTATION_E enRotation);
```

【参数】

参数名称	描述	输入/输出
hHandle	已开启的 Job Handle	输入
pstTask	GDC Task pointer	输入
enRotation	旋转角度 (90, 270);	输入

【返回值】

返回值	描述
0	成功。
非 0	失败, 请参见错误码 。

【需求】

- 头文件：cvi_gdc.h, cvi_comm_gdc.h
- 库文件：libvpu.so

【注意】

- hHandle 必须为已经开启的 Job
- phHandle 不能为空指针或非法指针。

【举例】

```
GDC_HANDLE hHandle;
struct gdc_job *job;
GDC_TASK_ATTR_S stTask;
VB_BLK blk;
VB_S *vb_out;
CVI_U32 buf_size;
CVI_S32 ret;
MMF_CHN_S *_chn;
SIZE_S size_out;

if (rot == ROTATION_90) {
  size_out = vb_in->buf.size;
} else {
  size_out.u32Width = vb_in->buf.size.u32Height;
  size_out.u32Height = vb_in->buf.size.u32Width;
}

// get buf for gdc output.
buf_size = vb_in->buf.length[0] + vb_in->buf.length[1] + vb_in->buf.length[2];
blk = CVI_VB_GetBlock(VB_INVALID_POOLID, buf_size);

vb_in->mod_ids |= BIT(CVI_ID_GDC);
vb_out = (VB_S *)blk;
vb_out->mod_ids = BIT(CVI_ID_GDC);
v4l2_get_frame_info(enPixFormat, size_out, &vb_out->buf, vb_out->phy_addr);

CVI_GDC_BeginJob(&hHandle);

stTask.reserved = CVI_GDC_MAGIC;
CVI_GDC_AddRotationTask(hHandle, &stTask, rot);

job = (struct gdc_job *)hHandle;
job->sync_io = sync_io;
ret = CVI_GDC_EndJob(hHandle);
return ret;
```

【相关主题】

CVI_GDC_BeginJob

11.3.5 CVI_GDC_GenLDCMesh

【描述】

为 LDC 属性生成 mesh，mesh 用于存储原始图像和矫正后的图像像素点的对应性。

【语法】

```
CVI_S32 CVI_GDC_GenLDCMesh(CVI_U32 u32Width, CVI_U32 u32Height, const LDC_ATTR_
↪S *pstLDCAttr, const char *name, CVI_U64 *pu64PhyAddr, CVI_VOID
**ppVirAddr);
```

【参数】

参数名称	描述	输入/输出
U32Width	矫正后图像宽	输入
u32Height	矫正后图像高	输入
pstLDCAttr	镜头畸变矫正属性	输入
name	mesh 名字	输入
pu64PhyAddr	物理地址	输入
ppVirAddr	虚拟地址	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，请参见错误码 。

【需求】

- 头文件：cvi_gdc.h, cvi_comm_gdc.h
- 库文件：libvpu.so

【举例】

```
Please refer to CVI_GDC_AddLDCTask
```

【相关主题】

CVI_GDC_BeginJob

11.3.6 CVI_GDC_AddLDCTask

【描述】

在既有 Job 中安插一个镜头矫正的 Task。一次镜头矫正会分 2 个 task 完成，第一次会先旋转 90 度，第二次会再旋转 270 度，两次旋转过程中会伴随不同方向的矫正过程。

【语法】

```
CVI_S32 CVI_GDC_AddLDCTask(GDC_HANDLE hHandle, const GDC_TASK_ATTR_S␣
↪*pstTask, const LDC_ATTR_S *pstLDCAttr, ROTATION_E enRotation);
```

【参数】

参数名称	描述	输入/输出
hHandle	已开启的 Job Handle	输入
pstTask	GDC Task pointer	输入
pstLDCAttr	LDC 属性	输入
enRotation	旋转角度 (90, 270);	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，请参见错误码 。

【需求】

- 头文件：cvi_gdc.h, cvi_comm_gdc.h
- 库文件：libvpu.so

【注意】

- hHandle 必须为已经开启的 Job
- phHandle 不能为空指针或非法指针。

【举例】

```
stTask.stImgOut.stVFrame.u32Width = ALIGN(stSize.u32Width, DEFAULT_ALIGN);
stTask.stImgOut.stVFrame.u32Height = ALIGN(stSize.u32Height, DEFAULT_ALIGN);
s32Ret = CVI_GDC_GenLDCMesh(stTask.stImgOut.stVFrame.u32Width,
stTask.stImgOut.stVFrame.u32Height
  , &stLDCAttr, "ldc_user", &u64PhyAddr, &pVirAddr);
if (s32Ret != CVI_SUCCESS) {
  printf("CVI_GDC_GenLDCMesh failed, s32Ret:0x%x", s32Ret);
  CVI_GDC_CancelJob(hHandle);
  return -1;
}

GDC_TASK_ATTR_S stTaskArr[2];
SIZE_S size_out[2];
ROTATION_E enRotationOut[2];
```

（下页继续）

(续上页)

```
CVI_U32 mesh_1st_size;

// Rotate 90/270 for 1st job
size_out[0].u32Width = ALIGN(stSize.u32Height, DEFAULT_ALIGN);
size_out[0].u32Height = ALIGN(stSize.u32Width, DEFAULT_ALIGN);

if (enRotation == ROTATION_0 || enRotation == ROTATION_180) {
  size_out[1].u32Width = ALIGN(stSize.u32Width, DEFAULT_ALIGN);
  size_out[1].u32Height = ALIGN(stSize.u32Height, DEFAULT_ALIGN);
} else {
  size_out[1].u32Width = ALIGN(stSize.u32Height, DEFAULT_ALIGN);
  size_out[1].u32Height = ALIGN(stSize.u32Width, DEFAULT_ALIGN);
}
stTask.stImgOut.stVFrame.u32Width = size_out[0].u32Width;
stTask.stImgOut.stVFrame.u32Height = size_out[0].u32Height;
switch (enRotation) {
default:
case ROTATION_0:
  enRotationOut[0] = ROTATION_90;
  enRotationOut[1] = ROTATION_270;
  break;
case ROTATION_90:
  enRotationOut[0] = ROTATION_90;
  enRotationOut[1] = ROTATION_0;
  break;
case ROTATION_180:
  enRotationOut[0] = ROTATION_90;
  enRotationOut[1] = ROTATION_90;
  break;
case ROTATION_270:
  enRotationOut[0] = ROTATION_270;
  enRotationOut[1] = ROTATION_0;
  break;
}
memcpy(&stTaskArr[0], &stTask, sizeof(stTask));
stTaskArr[0].reserved = 0;//magic id
stTaskArr[0].au64privateData[0] = u64PhyAddr;
stTaskArr[0].au64privateData[1] = (uintptr_t)pVirAddr;
stTaskArr[0].au64privateData[2] = CVI_FALSE;
s32Ret = CVI_GDC_AddLDCTask(hHandle, &stTaskArr[0], &stLDCAttr, enRotationOut[0]);
if (s32Ret != CVI_SUCCESS) {
  printf("CVI_GDC_AddLDCTask 1st failed, s32Ret:0x%x", s32Ret);
  CVI_GDC_CancelJob(hHandle);
  return -1;
}

mesh_gen_get_1st_size(size_out[0], &mesh_1st_size);
memcpy(&stTaskArr[1].stImgIn, &stTask.stImgOut, sizeof(stTask.stImgOut));
memcpy(&stTaskArr[1].stImgOut, &stTask.stImgIn, sizeof(stTask.stImgIn));
stTaskArr[1].stImgOut.stVFrame.u32Width = size_out[1].u32Width;
stTaskArr[1].stImgOut.stVFrame.u32Height = size_out[1].u32Height;
stTaskArr[1].reserved = 0;//magic id
stTaskArr[1].au64privateData[0] = u64PhyAddr + mesh_1st_size;
stTaskArr[1].au64privateData[1] = (uintptr_t)pVirAddr;//save orig mesh addr
```

(下页继续)

(续上页)

```
stTaskArr[1].au64privateData[2] = CVI_TRUE;
s32Ret = CVI_GDC_AddLDCTask(hHandle, &stTaskArr[1], &stLDCAttr, enRotationOut[1]);
if (s32Ret != CVI_SUCCESS) {
  printf("CVI_GDC_AddLDCTask 2nd failed, s32Ret:0x%x", s32Ret);
  CVI_GDC_CancelJob(hHandle);
  return -1;
}
memcpy(&stTask, &stTaskArr[1], sizeof(GDC_TASK_ATTR_S));

s32Ret = CVI_GDC_EndJob(hHandle);
if (s32Ret != CVI_SUCCESS) {
  printf("CVI_GDC_EndJob failed, s32Ret:0x%x\n", s32Ret);
  CVI_GDC_CancelJob(hHandle);
  goto EXIT1;
}
return 0;
```

【相关主题】

CVI_GDC_BeginJob

11.4 数据类型

11.4.1 GDC_TASK_ATTR_S

【说明】

定义 GDC TASK 的属性。GDC 支持的像素格式有 NV21/NV12/YUV400。

【定义】

```
typedef struct _GDC_TASK_ATTR_S {
  VIDEO_FRAME_INFO_S stImgIn;
  VIDEO_FRAME_INFO_S stImgOut;
  CVI_U64 au64privateData[4];
  CVI_U64 reserved;
} GDC_TASK_ATTR_S;
```

【成员】

成员名称	描述
StImgIn	输入影像属性
stImgOut	输出影像属性
au64privateData	Task 相关的私有数据
reserved	预留

【注意事项】

影像属性必须满足相对应硬件属性

【相关数据类型及接口】

- VIDEO_FRAME_INFO_S

11.4.2 LDC_ATTR_S

【说明】

定义镜头矫正的属性。

【定义】

```
typedef struct _LDC_ATTR_S {
  CVI_BOOL bAspect; /* RW;Whether aspect ration  is keep */
  CVI_S32 s32XRatio; /* RW; Range: [0, 100], field angle ration of  horizontal,valid when bAspect=0.
↪*/
  CVI_S32 s32YRatio; /* RW; Range: [0, 100], field angle ration of  vertical,valid when bAspect=0.*/
  CVI_S32 s32XYRatio; /* RW; Range: [0, 100], field angle ration of  all,valid when bAspect=1.*/
  CVI_S32 s32CenterXOffset;
  CVI_S32 s32CenterYOffset;
  CVI_S32 s32DistortionRatio;
  GRID_INFO_ATTR_S stGridInfoAttr;
} LDC_ATTR_S;
```

【成员】

成员名称	描述	范围
bAspect	视野调整过程中是否保持幅型比	bool
s32XRatio	视野大小参数，bAspect=1 时有效	0 ~ 100
s32YRatio	X 方向视野大小参数，bAspect=0 时有效	0 ~ 100
s32XYRatio	Y 方向视野大小参数，bAspect=0 时有效	0 ~ 100
s32CenterXOffset	图像中心点相对于物理中心点的水平偏移	-51 ~ +511
s32CenterYOffset	图像中心点相对于物理中心点的垂直偏移	-51 ~ +511
s32DistortionRatio	矫正强度，负数为枕型，正数为桶型	-300 ~ +500
stGridInfoAttr	gridinfo 参数	/

【注意事项】

影像属性必须满足相对应硬件属性

【相关数据类型及接口】

- VIDEO_FRAME_INFO_S
- GRID_INFO_ATTR_S

11.4.3 GRID_INFO_ATTR_S

【说明】

定义 GridInfo 的属性。

【定义】

```
typedef struct _GRID_INFO_ATTR_S {
        CVI_BOOL Enable;
        char gridFileName[128];
        char gridBindName[128];
        CVI_BOOL isBlending;
        CVI_BOOL bEISEnable; /* enable EIS */
        uint8_t homoRgnNum;
} GRID_INFO_ATTR_S;
```

【成员】

成员名称	描述	范围
bEnable	是否开启 gridinfo	bool
gridFileName	gridinfo 文件名称	/
gridBindName	gridinfo 绑定名称	/
isBlending	暂未使用	bool
bEISEnable	暂未使用	bool
homoRgnNum	暂未使用	/

【注意事项】

影像属性必须满足相对应硬件属性

【相关数据类型及接口】

- LDC_ATTR_S

11.5 错误码

错误代码	宏定义	描述
0XC01E800D	CVI_ERR_GDC_NOBUF	分配内存失败
0XC01E800E	CVI_ERR_GDC_BUF_EMPTY	内存空闲
0XC01E8006	CVI_ERR_GDC_NULL_PTR	空指针
0XC01E8003	CVI_ERR_GDC_ILLEGAL_PARAM	参数错误
0XC01E800F	CVI_ERR_GDC_BUF_FULL	内存已满
0XC01E8010	CVI_ERR_GDC_SYS_NOTREADY	要求未完成
0XC01E8008	CVI_ERR_GDC_NOT_SUPPORT	不支持
0XC01E8009	CVI_ERR_GDC_NOT_PERMITTED	没有 TASK
0XC01E8012	CVI_ERR_GDC_BUSY	系统忙碌

12 Proc 调试信息说明

12.1 功能概述

CV181x/CV180x 透过 linux 下的 proc 文件系统来显示调试信息，调试信息反应当前系统的运行状态。用户可透过这些调试信息来作问题分析与定位。

12.2 AI

【调试信息】

```
/ # cat /proc/audio_debug/cviteka_adc

------------- CVI AI ATTRIBUTE -------------
AiDev    Workmode    SampleRate    BitWidth
  0       slave        128000        16

-------------  CVI AI STATUS   -------------
I2S0 is off

SDMA clk is off

ADC is off (0)

L-Mute   R-Mute
  no       no

L-Vol          R-Vol
  0             0
```

【调试信息分析】

记录当前音频输入设备参数以及状态信息。

【参数说明】

参数	描述
AiDev	AI 设备号
Workmode	AI 工作模式: master: I2S 主模式 slave: I2S 从模式
SampleRate	采样率。范围 [8k ~ 48K]
BitWidth	采样精度。范围 [16bit]
I2S0	ADC codec 对接的 I2S 状态 on: 开启 off: 关闭
SDMA clk	sysDMA clock 状态 on: 开启 off: 关闭
ADC	ADC codec 状态 0x0: 关闭 0x3: 开启
L-Mute	左声道静音模式: yes: 静音模式开启 no: 静音模式关闭
R-Mute	右声道静音模式: yes: 静音模式开启 no: 静音模式关闭
L-Vol	左声道声量。范围 [0~24]
R-Vol	右声道声量。范围 [0~24]

12.3 AO

【调试信息】

```
/ # cat /proc/audio_debug/cviteka_dac

------------ CVI AO ATTRIBUTE ------------
AiDev    Workmode    SampleRate    BitWidth
  1       master        64000        16

------------  CVI AO STATUS   ------------
I2S3 is off

SDMA clk is off
DAC is off (0)
L-Mute   R-Mute
  no       no

L-Vol          R-Vol
  32           32
```

【调试信息分析】

记录当前音频输出设备参数以及状态信息。

【参数说明】

参数	描述
AiDev	AO 设备号
Workmode	AO 工作模式: master: I2S 主模式 slave: I2S 从模式
SampleRate	采样率。范围 [8k ~ 48K]
BitWidth	采样精度。范围 [16bit]
I2S3	ADC codec 对接的 I2S 状态 on: 开启 off: 关闭
SDMA clk	sysDMA clock 状态 on: 开启 off: 关闭
DAC	DAC codec 状态 0x0: 关闭 0x3: 开启
L-Vol	左声道声量。范围 [0~32]
R-Vol	右声道声量。范围 [0~32]

12.4 VENC

【调试信息】

```
# cat /proc/cvitek/venc
Module: [VENC] System Build Time [#1 SMP PREEMPT Sat Jan 23 02:50:26 CST 2021]
-----MODULE PARAM----------------------------------------------
VencBufferCache: 0      FrameBufRecycle: 0      VencMaxChnNum: 70
-----VENC CHN ATTR 1-------------------------------------------
ID: 0   Width: 1920     Height: 1080    Type: H264      RcMode: CBR     ByFrame: Y      Sequence: 0
↪  LeftBytes: 0    LeftFrm: 0      CurPacks: 0     GopMode: NORMALP        Prio: 0
-----VENC CHN ATTR 2-------------------------------------------
VeStr: Y        SrcFr: 0        TarFr: 0        Timeref: 0      PixFmt: YUV420  PicAddr: 0x130000000
↪WakeUpFrmCnt: 0
-----VENC CROP INFO--------------------------------------------
ID: 0   CropEn: N       StartX: 0       StartY: 0       Width: 1920     Height: 1080
-----VENC PTS STATE--------------------------------------------
ID: 0   RcvFirstFrmPts: 0       RcvFrmPts: 254
-----VENC CHN PERFORMANCE--------------------------------------
ID: 0   InFPS: 25       OutFPS: 25      HwEncTime: 12 ms

----- CVITEK Debug Level STATE ---------------------------------
VencDebugMask: 0x0      VencStartFrmIdx: 0      VencEndFrmIdx: 0        VencDumpPath:
```

【调试信息分析】

记录当前视频编码属性配置以及状态信息。

【参数说明】

参数		描述
MODULE PARAM	VencBufferCache	Whether the encoding stream buffer uses the cache mode 0: no 1: yes
	FrameBufRecycle	Whether the idle buffers used for store reference frames and advanced smartP-frames are recycled during encoding 0: not recycled 1: recycled
	VencMaxChnNum	Maximum number of encoding channels
VENC CHN ATTR1	ID	VENC channel ID
	Width	VENC channel width
	Height	VENC channel height
	Type	VENC channel type
	ByFrame	Mode of obtaining streams 0: by packet 1: by frame
	Sequence	Sequence number When the streams are obtained by frame, it represents the frame sequence number. When the streams are obtained by packet, it represents the packet sequence number.
	LeftBytes	Remaining bytes in a stream buffer
	LeftFrm	Number of remaining stream frames in a stream buffer
	CurPacks	Number of stream packets for the current frame (invalid currently)
	GopMode	GOP mode
	prio	Channel priority
VENC CHN ATTR 2	VeStr	Whether to start encoding
	SrcFr	Source frame rate (input frame rate) used by the VENC for controlling the frame rate
	TarFr	Target frame rate used by the VENC for controlling the frame rate
	Timeref	Timeref of the latest frame in the busy queue

下页继续

表 12.1 – 续上页

参数		描述
	PixFmt	Format of the frame that is being encoded
	PicAddr	Address for the frame that is being encoded
	WakeUpFrmCnt	Number of frames of the specified wakeup blocking interface when the channel usage times out or the streams are obtained in block mode
VENCCROP INFO	ID	VENC channel ID
	CropEn	Whether to enable the cropping function of the VENC channel
	StartX	Start horizontal coordinate of the image to be cropped
	StartY	Start vertical coordinate of the image to be cropped
	Width	Width of the cropped image
	Height	Height of the cropped image
ROI INFO	ID	Channel ID
	Index	Region of interest (ROI) index
	bRoiEn	ROI enable
	bAbsQp	Whether the ROI uses the absolute QP mode
	Qp	QP value configured by the ROI
	Width	ROI width (unit: pixel)
	Height	ROI height (unit: pixel)
	StartX	Start horizontal coordinate of the ROI (unit: pixel)
	StartY	Start vertical coordinate of the ROI (unit: pixel)
VENC PTS STATE Timestamp of a framereceived by thechannel	ID	Channel ID
	RcvFirstFrmPts	Timestamp of the first frame received by the channel
	RcvFrmPts	Timestamp of the current frame received by the channel
VENC CHN PERFORMANCE	ID	Channel ID

下页继续

表 12.1 – 续上页

参数		描述
	InFPS	Input FPS of venc channel (from VI or VPSS or app)
	OutFPS	Output FPS of venc channel
	HwEncTime	Hw encode time cost
CVITEK DEBUG STATE	VencDebugMask	The debug mask of middleware
	VencStartFrmIdx	The start frame of middleware debugging
	VencEndFrmIdx	The end frame of middleware debugging
	VencDumpPath	The YUV src frame dump path

12.5 H265E

【调试信息】

```
# cat /proc/cvitek/h265e
Module: [H265E] System Build Time [#1 SMP PREEMPT Sat Jan 23 02:50:26 CST 2021]
-----MODULE PARAM----------------------------------------------
OnePack: 0      H265eVBSource: 0      PowerSaveEn: 0  MiniBufMode: 0  bQpHstgrmEn: 0
-----CHN ATTR--------------------------------------------------
ID: 0   MaxWidth: 1920  MaxHeight: 1080       Width: 1920     Height: 1080    Profile: 0      C2GEn:␣
↪0      BufSize: 0      ByFrame: 1      GopMode: NORMALP        MaxStrCnt: 0
```

【调试信息分析】

记录当前 H.265 视频编码属性配置以及状态信息。

【参数说明】

参数		描述
MODULEPARAM	OnePack	Mode in which streams are obtained 0: Streams are obtained in multi-packet mode. 1: Streams are obtained in single-packet mode.
	H265eVBSource	Mode in which the VB for the reference frames and reconstruction frames are obtained 2: Private VB mode 3: User VB mode
	PowerSaveEn	Low-power parameter control switch 0: The low-power parameters are disabled. 1: The low-power parameters are enabled.
	MiniBufMode	Mode of allocating the stream buffers 0: The stream buffers are allocated based on the resolution. 1: The lower limit for the stream buffer size is 32 KB. Users need to ensure that the size of the allocated stream buffer is appropriate.
	bQpHstgrmEn	Whether to display the QP histogram in the advanced stream information 0: no 1: yes
CHN ATTR	ID	Channel ID
	MaxWidth	Maximum width of the encoding channel (unit: pixel)
	MaxHeight	Maximum height of the encoding channel (unit: pixel)
	Width	Width (unit: pixel)
	Height	Height (unit: pixel)
	profile	Encoding channel profile MP: main profile
	C2GEn	Color-to-gray enable Value range: {0, 1}
	BufSize	Stream buffer size (unit: byte)
	ByFrame	Whether to obtain streams by frame Value range: {0, 1}
	GopMode	GOP mode
	MaxStrCnt	Maximum number of frames in the stream buffer Default value: 200

12.6 H264E

【调试信息】

```
# cat /proc/cvitek/h264e
Module: [H264E] System Build Time [#1 SMP PREEMPT Sat Jan 23 02:50:26 CST 2021]
-----MODULE PARAM----------------------------------------------------
OnePack: 0      H264eVBSource: 0      PowerSaveEn: 0  MiniBufMode: 0  bQpHstgrmEn: 0
-----CHN ATTR--------------------------------------------------------
ID: 0   MaxWidth: 1920  MaxHeight: 1080      Width: 1920     Height: 1080    Profile: 0      C2GEn:␣
↪0      BufSize: 0      ByFrame: 1      GopMode: NORMALP      MaxStrCnt: 0
```

【调试信息分析】

记录当前 H.264 视频编码属性配置以及状态信息。

【参数说明】

参数		描述
MODULEPARAM	OnePack	Mode in which streams are obtained 0: Streams are obtained in multi-packet mode. 1: Streams are obtained in single-packet mode.
	H264eVBSource	Mode in which the VB is obtained for the reference frame and reconstruction frame 2: The private VBs are used. 3: The VBs are allocated by the user.
	PowerSaveEn	Low-power parameter control switch 0: The low-power parameters are disabled. 1: The low-power parameters are enabled.
	MiniBufMode	Mode of allocating the stream buffers 0: The stream buffers are allocated based on the resolution. 1: The lower limit for the stream buffer size is 32 KB. Users need to ensure that the size of the allocated stream buffer is appropriate.
	bQpHstgrmEn	Whether to display the QP histogram in the advanced stream information 0: no 1: yes
CHN ATTR	ID	Channel ID
	MaxWidth	Maximum width of the encoding channel (unit: pixel)
	MaxHeight	Maximum height of the encoding channel (unit: pixel)
	Width	Width (unit: pixel)
	Height	Height (unit: pixel)
	profile	Encoding channel profile Base: baseline MP: main profile HP: high profile
	C2GEn	Color-to-gray enable Value range: {0, 1}
	BufSize	Stream buffer size (unit: byte)
	ByFrame	Whether to obtain streams by frame Value range: {0, 1}

12.7 JPEGE

【调试信息】

```
# cat /proc/cvitek/jpege
Module: [JPEGE] System Build Time [#1 SMP PREEMPT Sat Jan 23 02:50:26 CST 2021]
-----MODULE PARAM-----------------------------------------------
OnePack: 0      JpegeMiniBufMode: 0     JpegClearStreamBuf: 0   JpegeDeringMode: 0
-----CHN ATTR---------------------------------------------------
ID: 0   bMjpeg: Y       PicType: YUV422         MaxWidth: 3840  MaxHeight: 2160         Width: 3840␣
↪   Height: 2160    BufSize: 0      ByFrm: 1        MCU: 1  Qfactor: 0      C2GEn: 0        DcfEn: 0
```

【调试信息分析】

记录 JPEG 编码过程中，各通道的编码属性、状态。

【参数说明】

参数		描述
MODULEPARAM	OnePack	Mode in which streams are obtained 0: Streams are obtained in multi-packet mode. 1: Streams are obtained in single-packet mode.
	JpegeMiniBufMode	Mode of allocating the stream buffers 0: The stream buffers are allocated based on the resolution. 1: The lower limit for the stream buffer size is 32 KB. Users need to ensure that the size of the allocated stream buffer is appropriate.
	JpegClearStreamBuf	Whether to clear the stream buffers when the attribute of a JPEG encoding channel is set. 0: The stream buffers and context count are reserved. 1: The stream buffers are cleared.
	JpegeDeringMode	De-ring effect mode enable for the JPEG encoding channel 0: disabled 1: enabled. Under the same quantization table and Qfactor, the ring phenomenon can be reduced and the image file size can be decreased, but some image clarity and details are also lost.
CHN ATTR	ID	Channel ID
	bMjpeg	MJPEG encoding No: JPEG snapshot Yes: MJPEG encoding
	PicType	Picture type: YVU422 or YVU420
	MaxWidth	Maximum width of the encoding channel (unit: pixel)
	MaxHeight	Maximum height of the encoding channel (unit: pixel)
	Width	Picture width
	Height	Picture height
	BufSize	Stream buffer size (unit: byte)
	MCU	Number of minimum coded units (MCUs) in each embedded RISC-V subsystem

12.8 RC

【调试信息】

```
# cat /proc/cvitek/rc
Module: [RC] System Build Time [#1 SMP PREEMPT Sat Jan 23 02:50:26 CST 2021]
------BASE PARAMS 1----------------------------------------------------
ChnId: 0        Gop: 5  StatTm: 4294967295      ViFr: 0        TrgFr: 0        ProType: H264   RcMode:␣
↪CBR    Br(kbps): 2048  FluLev: 0       IQp: 0  PQp: 0  BQp: 0
------BASE PARAMS 2----------------------------------------------------
ChnId: 0        MinQp: 26       MaxQp: 42       MinIQp: 36      MaxIQp: 42      EnableIdr: 0    ␣
↪bQpMapEn: 0    QpMapMode: N/A
-----GOP MODE ATTR-------------------------------------------------------
ChnId: 0        GopMode: NORMALP        IpQpDelta: 0    SPInterval: 0   SPQpDelta: 0    ␣
↪BFrmNum: 0     BQpDelta: 0     BgInterval: 0   ViQpDelta: 0
-----RUN CBR PARAM ------------------------------------------
ChnId: 0        MinIprop: 0     MaxIprop: 0     MaxQp: 42       MinQp: 26       MaxIQp: 42      ␣
↪MinIQp: 36     MaxReEncTimes: 0
```

【调试信息分析】

记录当前编码码率控制的信息。

【参数说明】

参数		描述
BASEPARAMS1	ChnId	VENC channel ID
	Gop	Encoding group of images (GOP)
	StatTm	Bit rate statistics (unit: second)
	ViFr	Frame rate for transmitting images by the VI
	TrgFr	Target frame rate for encoding
	ProType	Encoding type
	RcMode	Bit rate control mode (CBR, VBR, QPMAP, or FixQp)
	Br(kbps)	The unit of the bit rate is kbit/s.
	FluLev	Fluctuation level, valid only for the CBR mode
	IQp	I-frame QP, valid only for the FixQp mode
	PQp	P-frame QP, valid only for the FixQp mode
	BQp	B-frame QP, valid only for the FixQp mode This is currently not supported.
BASEPARAMS 2	ChnId	VENC channel ID
	EnableIdr	IDR enable switch Y: enabled N: disabled
	bQpMapEn	QpMap enable switch Y: enabled N: disabled
	QpMapMode	Mode of the QP value used for CU32 or CU64, valid only for H.265. **MEANQP**, **MINQP**, and **MAXQP** indicate the average, minimum, and maximum QP value, respectively.
GOP MODEATTR	ChnId	VENC channel ID
	GopMode	GOP mode
	IpQpDelta	QP delta of I-frames relative to P-frames. QP deltas of the background frame and P-frame are displayed in SmartP mode. Value range: [–10, +30]
	SPInterval	Interval of the special P-frames
		Value range: less than or equal to the GOP value.
	SPQpDelta	QP delta of the special P-

12.9 VDEC

【调试信息】

```
# cat /proc/cvitek/vdec
Module: [VDEC] System Build Time [#1 SMP PREEMPT Sat Jan 23 02:50:26 CST 2021]
VdecMaxChnNum: 70       MiniBufMode: 0  enVdecVBSource: 0       ParallelMode: 0
MaxPicWidth: 4096       MaxPicHeight: 2160      MaxSliceNum: 200        VdhMsgNum: 0
↪VdhBinSize: 0  VdhExtMemLevel: 0
SupportProgressive: 0   DynamicAllocate: 0      CapStrategy: 0
----- CHN COMM ATTR & PARAMS -----------------------------------
ID: 0   TYPE: H265      MaxW: 4096      MaxH: 2304      Width: 1920     Height: 1080    Stride: 1920
↪  PixelFormat: 13         PTS: 0  PA: 0x126568000
StrInputMode: FRAME/NOBLOCK     StrBufSize: 9437184     FrmBufSize: 0   FrmBufCnt: 3
↪TmvBufSize: 0
ID: 0   DispNum: 2      DispMode: PLAYBACK      SetUserPic: N   EnUserPic: N    Rotation: 0
↪PicPoolId: -1  TmvPoolId: -1   STATE: START
----- CHN VIDEO ATTR & PARAMS ----------------------------------
ID: 0   VfmwID: 0       RefNum: 0       TemporalMvp: N  ErrThr: 0       DecMode: IPB
↪OutPutOrder: DISP      Compress: NONE  VideoFormat: 0  MaxVPS: 0       MaxSPS: 0
↪MaxPPS: 0      MaxSlice: 200
----- CHN PICTURE ATTR & PARAMS--------------------------------
ID: 0   PixelFormat: 0  Alpha: 0

----- CVITEK Debug Level STATE ---------------------------------------
VdecDebugMask: 0x0      VdecStartFrmIdx: 0      VdecEndFrmIdx: 0        VdecDumpPath:
```

【调试信息分析】

记录当前解码通道的使用状况及其属性配置。可用于检查属性配置以及当前解码通道统计状态。

【参数说明】

参数		描述
MODULEPARAM	VdecMaxChnNum	Maximum number of decoding channels supported by the VDEC
	MiniBufMode	Whether the stream buffer reduction mode is used 0: unused 1: used (the reduction mode is valid only when streams are decoded by frame)
	enVdecVBSource	Mode of allocating the VDEC video buffer (VB) 1: module VB 2: private VB 3: user VB
	ParallelMode	VDH decoding mode 0: non-parallel mode 1: parallel mode

下页继续

表 12.2 – 续上页

参数		描述
	MaxPicWidth	Maximum width supported for video decoding
	MaxPicHeight	Maximum height supported for video decoding
	MaxSliceNum	Maximum number of slices supported for video decoding
	VdhMsgNum	Number of VDH message pools
	VdhBinSize	Size of the buffer for storing binary data of VDH decoding
	VdhExtMemLevel	Off-processor memory allocation level for VDH decoding
	MaxJpegeWidth	Maximum width of an image to be decoded
	MaxJpegeHeight	Maximum height of an image to be decoded
	SupportProgressive	Whether the progressive format is supported 0: no 1: yes
	DynamicAllocate	Buffer allocation mode when the progressive format is supported 0: static allocation 1: dynamic allocation
	CapStrategy	Capability strategy for the maximum width and height of a decoded image 0: capability strategy based on the module 1: capability strategy based on the channel
CHNCOMMATTR &PARAMS	ID	VDEC channel ID
	TYPE	VDEC channel type PT_H264 PT_H265 PT_MJPEG PT_JPEG
	MaxW	Configured maximum width of a decoded image
	MaxH	Configured maximum height of a decoded image
	Width	Width of a decoded image
	Height	Height of a decoded image

下页继续

表 12.2 – 续上页

参数		描述
	StrmInputMode	Stream transmission mode of the VDEC channel The modes can be classified into two types: FRAME, STREAM, and COMPAT: transmit by frame, stream, and, in compatible mode, respectively BLOCK, NOBLOCK and TIMEOUT: streams in block, non-block and timeout mode
	StrBufSize	Stream buffer size
	FrmBufSize	Size of frame buffers, valid only in private VB mode
	FrmBufCnt	Number of frame buffers, valid only in private VB mode
	TmvBufSize	TMV buffer size. This parameter is valid only in private VB mode.
	DispNum	Number of displayed frames Value range: [0, 16]
	DispMode	Display mode Value range: PLAYBACK and PREVIEW
	SetUserPic	Whether to set user images
	EnUserPic	Whether to enable user images
	Rotation	Rotated angle of the VDEC image
	PicPoolId	VB pool ID of the frame buffer, valid only in private VB and user VB modes
	TmvPoolId	VB pool ID of the Tmv, valid only in private VB and user VB modes
	STATE	Whether the VDEC channel starts to receive streams START: The channel starts to receive streams. STOP: The channel stops receiving streams.

下页继续

表 12.2 – 续上页

参数		描述
CHN VIDEO ATTR & PARAMS	ID	VDEC channel ID
	VfmwID	Video firmware (VFMW) channel ID
	RefNum	Number of reference frames Value range: [0, 16]
	TemporalMvp	Whether to support time-domain motion vector prediction
	ErrThr	Stream error rate threshold
	DecMode	Decoding mode
	OutPutOrder	Output sequence of a decoded image
	Compress	Whether the decoded output image can be compressed
	VideoFormat	Data format of images to be decoded
	MaxVPS	Maximum number of supported VPSs, only H.265 decoding is valid
	MaxSPS	Maximum number of supported SPSs
	MaxPPS	Maximum number of supported PPSs
	MaxSlice	Maximum number of supported slices
CHN image ATTR & PARAMS	ID	VDEC channel ID
	PixelFormat	Output format of JPEG images
	Alpha	Global alpha value of JPEG images in ARGB format
CVITEK DEBUG STATE	VdecDebugMask	The debug mask of middleware
	VdecStartFrmIdx	The start frame of middleware debugging
	VdecEndFrmIdx	The end frame of middleware debugging
	VdecDumpPath	The Bitstream src dump path

12.10 LOG

【调试信息】

```
# cat /proc/cvitek/log
-----CURRENT LOG LEVEL-----------------------------------------------------------------
BASE          ( 4)
VB            ( 4)
SYS           ( 4)
RGN           ( 4)
CHNL          ( 4)
VDEC          ( 4)
VPSS          ( 4)
VENC          ( 4)
H264E         ( 4)
JPEGE         ( 4)
MPEG4E        ( 4)
H265E         ( 4)
JPEGD         ( 4)
VO            ( 4)
VI            ( 4)
DIS           ( 4)
RC            ( 4)
AIO           ( 4)
AI            ( 4)
AO            ( 4)
AENC          ( 4)
ADEC          ( 4)
AUD           ( 4)
VPU           ( 4)
ISP           ( 4)
IVE           ( 4)
USER          ( 4)
PROC          ( 4)
LOG           ( 6)
H264D         ( 4)
GDC           ( 4)
PHOTO         ( 4)
FB            ( 4)
----------------------------------------------------------------------------------------------------
```

【调试信息分析】

- 记录当前各个模块的调试级别。
- cat /proc/cvitek/log 用于获取各个模块 log 级别信息。
- 修改 log 级别:

修改某个模块的调试等级使用 echo 命令，比如: echo “VENC=4” > /proc/cvitek/log
修改所有模块的调试等级使用: echo “ALL=4” > /proc/cvitek/log

【参数说明】

参数		描述
CURRENT LOG LEVEL	BASE/VB/SYS/RGN/ CHNL/VDEC/VPSS/VENC/ H264E/JPEGE/MPEG4E/ H265E/JPEGD/VO/VI/ DIS/RC/AIO/AI/AO/AENC/ ADEC/AUD/VPU/ISP/ IVE/USER/PROC/LOG/ H264D/GDC/PHOTO/FB	模块名，后面的数字为 log 打印等级。

12.11 SYS

【调试信息】

```
# cat /proc/cvitek/sys

Module: [SYS], Version[], Build Time[#1 SMP PREEMPT Wed Feb 24 15:02:47 CST 2021]
-----BIND RELATION TABLE----------------------------------------------------------------------------------------
1stMod    1stDev    1stChn    2ndMod    2ndDev    2ndChn    3rdMod    3rdDev    3rdChn
VI          0         0         VPSS        0         0         VENC        0         0
VPSS        0         1         VENC        0         1         null        0         0

--------------------------------------------------------------------------------------------------------------
```

【调试信息分析】

记录当前 SYS 模块的使用情况。

【参数说明】

参数		描述
BIND RELATION TABLE	1stMod	绑定关系中第一级的模块名，数据由第一级发送给第二级。
	1stDev	绑定关系中第一级的设备号，数据由第一级发送给第二级。
	1stChn	绑定关系中第一级的通道号，数据由第一级发送给第二级。
	2ndMod	绑定关系中第二级的模块名，数据由第一级发送给第二级。
	2ndDev	绑定关系中第二级的设备号，数据由第一级发送给第二级。
	2ndChn	绑定关系中第二级的通道号，数据由第一级发送给第二级。
	3rdMod	绑定关系中第三级的模块名，如果有第三级绑定关系，则数据由第二级发送给第三级，否则显示为 null。
	3rdDev	绑定关系中第三级的设备号。
	3rdChn	绑定关系中第三级的通道号。

12.12 VB

【调试信息】

```
# cat /proc/cvitek/vb

-----VB PUB CONFIG------------------------------------------------------------------------------
MaxPoolCnt(512) ,  MaxBlkCnt(128)

-----COMMON POOL CONFIG-------------------------------------------------------------------
    PoolId(  0)        Size(     3145728)        Count( 12)
    PoolId(  1)        Size(    13283328)        Count(  5)

-------------------------------------------------------------------------------------------------------------
PoolName  : vbpool
PoolId    : 0
PhysAddr  : 0x130000000
VirtAddr  : 0x0
IsComm    : 1
Owner     : -1
BlkSz     : 3145728
BlkCnt    : 12
Free      : 9
MinFree   : 9

BLK  BASE  VB  SYS  RGN  CHNL  VDEC  VPSS  VENC  H264E  JPEGE  MPEG4E ␣
↪H265E  JPEGD  VO  VI  DIS
RC  AIO  AI  AO  AENC  ADEC  AUD  VPU  ISP  IVE  USER  PROC  LOG  H264D ␣
↪GDC  PHOTO  FB
#0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
#1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
#2  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
#3  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
0  0  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0
#4  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0
0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
#5  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0
0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
#6  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
#7  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
#8  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
#9  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
#10  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
#11  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
Sum  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  2  0
```

(下页继续)

(续上页)

```
0  0  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0

----------------------------------------------------------------------------------------------------
PoolName  : vbpool
PoolId    : 1
PhysAddr  : 0x132400000
VirtAddr  : 0x0
IsComm    : 1
Owner     : -1
BlkSz     : 13283328
BlkCnt    : 5
Free      : 5
MinFree   : 5

BLK  BASE  VB  SYS  RGN  CHNL  VDEC  VPSS  VENC  H264E  JPEGE  MPEG4E
↪H265E  JPEGD  VO  VI  DIS
RC  AIO  AI  AO  AENC  ADEC  AUD  VPU  ISP  IVE  USER  PROC  LOG  H264D
↪GDC  PHOTO  FB
#0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
#1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
#2  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
#3  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
#4  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
Sum  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0

----------------------------------------------------------------------------------------------------
```

【调试信息分析】

记录当前 VB 模块的 buffer 使用情况。

【参数说明】

参数		描述
VB PUB CONFIG	MaxPoolCnt	最大的缓存池的个数
	MaxBlkCnt	最大的缓存块的个数
COMMON POOL CONFIG	PoolId	公共缓存池的句柄。
	Size	缓存池内块的大小。
	Count	缓存池内块的个数。
PER VB POOL INFO	PoolName	公共/私有缓存池名称，若未设定预设为 vbpool
	PoolId	公共/私有缓存池的句柄
	PhysAddr	公共/私有缓存池的开始物理地址。
	VirtAddr	公共/私有缓存池的开始虚拟地址。
	IsComm	是否公共缓存池。 取值：{0, 1}。
	Owner	缓存池的拥有者。 · 2：私有池。 · 1：公共池。
	BlkSz	缓存池内缓存块的大小。
	BlkCnt	缓存池内缓存块的个数。
	Free	缓存池空闲缓存块的个数。
	MinFree	程序运行以来，空闲缓存块的最小剩余个数。若该计数为 0，则说明可能存在因缓存块不够而丢帧的情况。
	BLK	缓存池内缓存块的句柄。
	BASE/VB/SYS/RGN/ CHNL/VDEC/VPSS/VENC/ H264E/JPEGE/MPEG4E/ H265E/JPEGD/VO/VI/ DIS/RC/AIO/AI/AO/AENC/ ADEC/AUD/VPU/ISP/ IVE/USER/PROC/LOG/ H264D/GDC/PHOTO/FB	模块名 下面对应的数字表示当前模块有多少个地方占用缓存池内的该缓存块。 · 0：没占用。 · N：占用 N 个缓存块。

12.13 GDC

【调试信息】

```
# cat /proc/cvitek/gdc

Module: [GDC], Build Time[#1 SMP PREEMPT Wed Feb 24 15:02:47 CST 2021]

-----------------------------RECENT JOB INFO---------------------------
   SeqNo   ModName   TaskNum     State       InSize(pixel)      OutSize(pixel)
↪CostTime(us)HwTime(us)
```

(下页继续)

(续上页)

```
    # 0       VO    1    SUCCESS 921600   921600    3952    3539
    # 1       VO    1    SUCCESS 921600   921600    3941    3534
    # 2       VO    1    SUCCESS 921600   921600    3952    3553
    # 3       VO    1    SUCCESS 921600   921600    3938    3540
    # 4       VO    1    SUCCESS 921600   921600    3947    3547
    # 5       VO    1    SUCCESS 921600   921600    3936    3536
    # 6       VO    1    SUCCESS 921600   921600    3948    3551
    # 7       VO    1    SUCCESS 921600   921600    3926    3530

-------------------------------MAX WASTE TIME JOB INFO--------------------
  ModName   TaskNum     State       InSize(pixel)      OutSize(pixel)       CostTime(us)HwTime(us)
VO   1    SUCCESS921600   921600     4007
3530

-------------------------------GDC JOB STATUS-----------------------------
  Success      Fail    Cancel  BeginNum   BusyNumProcingNum
     754         0         0 000

-------------------------------GDC TASK STATUS----------------------------
  Success      Fail    Cancel   BusyNum
     754         0         0 0

-------------------------------GDC INT STATUS-----------------------------
    IntNumIntTm(us)       HalProcTm(us)
     754     3536      126

-------------------------------GDC CALL CORRECTION STATUS-----------------
  TaskSuc  TaskFail    EndSuc   EndFail     CbCnt
0         0      0  0     0
```

【调试信息分析】

记录 GDC 模块最近完成的若干任务、最近耗时最大的任务、历史累计信息。

【参数说明】

参数		描述
RECENT JOB INFO 最近完成的 job 的信息	SeqNo	打印序号。取值范围：[0，7]
	ModName	提交该 job 的模块名。
	TaskNum	该 job 包含的 task 数目。
	State	该 job 的处理状态。 取值范围：{FA IL、SUCCESS、WORKING} FAIL：表示 job 执行失败 SUC CESS：表示 job 执行成功 WOR KING：表示 job 正在处理
	InSize(pixel)	该 job 下各 task 的输入图像面积之和 单位：像素 每向该 job 添加一个 task，此项就加上该 task 的输入面积。
	OutSize(pixel)	该 job 下各 task 的输出图像面积之和 单位：像素 每向该 job 添加一个 task，此项就加上该 task 的输出面积。
	CostTime(us)	该 job 从提交到完成的耗时时长。 单位：us。 该时间包括针对该任务的软件、硬件及中断服务程序处理时间。
	HwTime(us)	该 job 在硬件中处理耗时时长。 单位：us。 该时间是硬件处理的时间，一般比 CostTime 要短。
MAX WASTE TIME JOBINFO 最近耗时最大的 job 信息	各项同 RECENT 最 JOB INFO 的成员 INFO 的成	500 个任务中耗时最长的 job 的信息。其各项同 RECENT JOB，具体意义请参见前述。 当出现耗时更长的任务或任务总数已超过 500 时，就更新该组值。 通过该组值可知最近的 GDC 运行性能，以及是否出现过 GDC 处理不及时的情况。
GDC JOB STATUSGDC 任务状态	Success 当硬	累计成功处理的 job 数。 理成功时加 1。
	Fail	累计处理失败的 job 数。 当 GDC 提交任务给驱动层并失败时加 1。该值增加时可通过查看日志了解失败原因。
	Cancel	累计的取消的 job 数。

12.14 REGION

【调试信息】

```
# cat /proc/cvitek/rgn

Module: [RGN], Build Time[#1 SMP PREEMPT Wed Feb 24 15:02:47 CST 2021]

------REGION STATUS OF COVER--------------------------------------------
    Hdl     Type     Used
    # 1         1         Y

------REGION CHN STATUS OF RECT COVER-----------------------------------
    Hdl     Type     Mod     Dev     Chn     bShow
       X         Y         W         H    Color     Layer    CoorType
    # 1         1      VPSS        0         1         Y
     30         0       100       100    FFFF         0         ABS

------REGION STATUS OF COVEREX------------------------------------------
    Hdl     Type     Used

------REGION CHN STATUS OF RECT COVEREX---------------------------------
    Hdl     Type     Mod     Dev     Chn     bShow
       X         Y         W         H    Color     Layer

------REGION STATUS OF OVERLAYEX----------------------------------------
    Hdl     Type         Used              PiFmt       W       H     BgColor                 Phy
    Virt    Stride   CnvsNum
    # 2         0            Y        ARGB_1555    300     200           0      136375000
↪    7f78785000      608       2

------REGION CHN STATUS OF OVERLAYEX------------------------------------
    Hdl     Type     Mod     Dev     Chn    bShow       X       Y    Layer
    # 2         0      VPSS        0         0         Y      10      10         0
```

【调试信息分析】

记录当前区域资源信息。

【参数说明】

参数		描述
REGION STATUS OFCOVERCOVER 状态	Hdl	COVER 的 Handle 号。
	Type	COVER 类型，值为 1。
	Used	该资源是否被占用。 · N：未占用。 · Y：占用。

下页继续

表 12.3 – 续上页

参数		描述
REGION CHN STATUS OFRECT COVERRECT-COVER 在通道中的显示状态	Hdl	COVER 的 Handle 号。
	Type	COVER 类型，值为 1。
	Mod	Attach 的模块。
	Dev	设备号。
	Chn	通道号。
	bShow	是否在该通道显示。 N：隐藏。 Y：显示。
	X	在该通道显示的起始 X 坐标。
	Y	在该通道显示的起始 Y 坐标。
	W	COVER 宽度（单位：像素）。
	H	COVER 高度（单位：像素）。
	Color	COVER 颜色。
	Layer	在该通道显示的层次。
	CoorType	ratio coordiante or abs coordinate
REGION STATUS OF COVEREX COVEREX 状态	Hdl	COVEREX 的 Handle 号。
	Type	COVEREX 类型，值为 2。
	Used	该资源是否被占用。 N：未占用。 Y：占用。
REGION CHN STATUS OF RECT COVEREX RECT COVEREX 在通道中的显示状态	Hdl	COVEREX 的 Handle 号。
	Type	COVEREX 类型，值为 2。
	Mod	Attach 的模块。
	Dev	设备号。
	Chn	通道号。
	bShow	是否在该通道显示。 N：隐藏。 Y：显示。
	X	在该通道显示的起始 X 坐标。
	Y	在该通道显示的起始 Y 坐标。
	W	COVERE X 宽度（单位：像素）。

下页继续

表 12.3 – 续上页

参数		描述
	H	COVERE X 高度（单位：像素）。
	Color	COVEREX 颜色。
	Layer	在该通道显示的层次。
REGION STATUS OF OVERLAYEX OVERLAYEX 状态	Hdl	OVERLAYEX 的 Handle 号。
	Type	O VERLAYEX 类型，值为 0。
	Used	该资源是否被占用。 N：未占用。 Y：占用。
	PiFmt	OVERLAYEX 像素格式，参看 PIXEL_FORMAT_E。
	W	OVERLAYE X 宽度（单位：像素）。
	H	OVERLAYE X 高度（单位：像素）。
	BgColor	OVERLAYEX 背景色。
	Phy	OVERLAYE X 占用内存的物理地址。
	Virt	OVERLAYE X 占用内存的虚拟地址。
	Stride	OVERLAYEX 内存跨度（单位：byte）。
	CnvsNum	OVERLAYEX 内存数目
REGION CHN STATUS OF OVERLAYEX OVERLAYEX 在通道中的显示状态	Hdl	OVERLAYEX 的 Handle 号。
	Type	O VERLAYEX 类型，值为 0。
	Mod	Attach 的模块。
	Dev	设备号。
	Chn	通道号。
	bShow	是否在该通道显示。 N：隐藏。 Y：显示。
	X	在该通道显示的起始 X 坐标。
	Y	在该通道显示的起始 Y 坐标。
	Layer	在该通道显示的层次。

12.15 VI

【调试信息】

```
# cat /proc/cvitek/vi

----------MODULE PARAM----------
      DetectErrFrame  DropErrFrame
            0             0

----------VI MODE----------
      DevID   PrerawFE       PrerawBE       Postraw        Scaler
       0      online         online         offline        offline

----------VI DEV ATTR1----------
      DevID   DevEn   BindPipe       Width   Height  IntfM   WkM     ScanM
       0      Y       Y             1920    1080    MIPI    1MUX    P

----------VI DEV ATTR2----------
      DevID   AD0     AD1     AD2     AD3     Seq     DataType        WDRMode
       0    -1      -1      -1      -1      N/A     RGB             None

----------VI BIND ATTR----------
      DevID   PipeNum        PipeId
       0      0              0

----------VI DEV TIMING ATTR----------
      DevID   DevTimingEn     DevFrmRate      DevWidth        DevHeight
       0        N              0              1920            1080

----------VI CHN ATTR1----------
      DevID   ChnID   Width   Height  Mirror  Flip    SrcFRate        DstFRate        PixFmt
↪VideoFmt
       0      0     1920    1080      N       N       -1              -1            NV21      SDR8

----------VI CHN ATT2----------
      DevID   ChnID   CompressMode    Depth   Align
       0      0     None             0      32

----------VI CHN OUTPUT RESOLUTION----------
      DevID   ChnID   Mirror  Flip    Width   Height  PixFmt  VideoFmt        CompressMode
↪FrameRate
       0      0       N       N       1920    1080    NV21      SDR8          None            -1

----------VI CHN ROTATE INFO----------
      DevID   ChnID   Rotate
       0      0       0

----------VI CHN EARLY INTERRUPT INFO----------
      DevID   ChnID   Enable  LineCnt
       0      0       N       0

----------VI CHN CROP INFO----------
      DevID   ChnID   CropEn  CoorType        CoorX   CoorY   Width   Height  TrimX   TrimY
↪TrimWid TrimHgt
```

(下页继续)

(续上页)

```
      0     0     N     RAT          0     0     0     0     0     0     0     0

-------------------------------VI CHN STATUS-------------------------------------
     DevID   ChnID   Enable  FrameRate      IntCnt  RecvPic LostFrame      VbFail  Width
↪Height
      0     0     N       0         1550    1550      0          0   1920    1080
```

【调试信息分析】

记录当前视频输入设备及信道的属性配置以及状态信息。

【参数说明】

参数		描述
MODULE PARAMVI 模块参数	DetectErrFrame 误帧	信号不稳定时，实时丢弃侦测到的错策略。该参数仅限于调试时使用，正式产品中不建议使用，把该值配成 0，实时丢掉错误帧。 >0：当检测到连续错误帧数大于该值时，则认为是时序配错，后面的帧不再丢弃； 0：默认值，表示实时丢弃检测到的错误帧图像。 <0：关闭检测错误帧功能。
	DropErrFrame	当检测到当前帧是错误帧时，认为接下来的几帧也可能是错误帧，应该丢弃。 0：默认值，表示不开启连续丢帧功能，只丢弃当前的错误帧； > 0：该参数表示当检测到图像错误时，连续丢 drop_err_frame 帧 (含当前帧)，不管后面的图像是否正确。
VI MODEVI PIPE 的工作模式	DevID	设备号。有效范围：[0, VI_MAX_DEV_NUM)。
	PrerawFE	PrerawFE 工作模式
	PrerawBE	PrerawBE 工作模式
	Postraw	Postraw 工作模式
	Scaler	Scaler 工作模式
VI DEV ATTR1 视频输入设备属性 1	DevID	设备号。有效范围：[0, VI_MAX_DEV_NUM)。
	DevEn	设备使能。 N：关闭； Y：打开。
	BindPipe	设备是否绑定 pipe。 N：未绑定； Y：已绑定。

下页继续

表 12.4 – 续上页

参数		描述
	Width	设备宽度，单位：像素。
	Height	设备高度，单位：像素。
	IntfM	输入模式。MIPI or BT or LVDS
	WkM	工作模式，现在只有 1MUX
	ScanM	隔行或逐行输入。 取值：{I, P}。 VI_SCAN_INTERLACED= I, VI_SCAN_PROGRESSIVE = P
VI DEV ATTR2 视频输入设备属性 2	DevID	设备号。有效范围：[0, VI_MAX_DEV_NUM)。
	AD0	AD 号。
	AD1	AD 号。
	AD2	AD 号。
	AD3	AD 号。
	Seq	数据顺序。 取值：{VUVU, UVUV, UYVY, VYUY, YUYV, YVYU}。
	DataType	输入数据类型，默认为 RGB。
	WDRMode	WDR 模式。 WDR_2L1：二合一行模式 WDR_2F1：二合一帧模式 WDR_3L1：三合一行模式 WDR_3F1：三合一帧模式 WDR_4L1：四合一行模式 WDR_4F1：四合一帧模式
VI BIND ATTR 绑定关系	DevID	设备号。有效范围：[0, VI_MAX_DEV_NUM)。
	PipeNum	Pipe 数目
	PipeId	PIPE 号。 有效范围：[0, VI_MAX_PIPE_NUM)。
VI DEV TIMING ATTR 自产生时序属性	DevID	设备号。有效范围：[0, VI_MAX_DEV_NUM)。
	DevTimingEn	是否使能自产生时序功能：(is_offline_preraw) N：关闭； Y：打开；
	DevFrmRate	用户设置的自产生时序帧率；（当其大于设备支持的最大帧率时，返回设备最大帧率）
	DevWidth	设备宽度，单位：像素；
	DevHeight	设备高度，单位：像素；

下页继续

表 12.4 – 续上页

参数		描述
VI CHN ATTR1 物理通道属性 1	DevID	设备号。有效范围：[0, VI_MAX_DEV_NUM)。
	ChnID	通道号。
	Width	通道输出宽度。
	Height	通道输出高度。
	Mirror	mirror 使能 N：关闭；Y：打开。
	Flip	flip 使能 N：关闭；Y：打开。
	SrcFRate	源帧率。
	DstFRate	目的帧率。
	PixFmt	输出像素格式。
	VideoFmt	输出视频格式。
VI CHN ATT2 物理通道属性 2	DevID	设备号。有效范围：[0, VI_MAX_DEV_NUM)。
	ChnID	通道号。
	CompressMode	是否压缩 N：关闭；Y：打开。
	Depth	用户获取通道帧的队列深度。
	Align	通道图像行 stride 对齐。
VI CHN OUTPUTRESOLUTIONVI 通道输出的图像属性	DevID	设备号。有效范围：[0, VI_MAX_DEV_NUM)。
	ChnID	通道号。
	Mirror	mirror 使能 N：关闭；Y：打开。
	Flip	flip 使能 N：关闭；Y：打开。
	Width	通道输出宽度。
	Height	通道输出高度。
	PixFmt	输出像素格式。
	VideoFmt	输出视频格式。
	CompressMode	是否压缩 N：关闭；Y：打开。
	FrameRate	帧率。
VI CHN ROTATE INFOVI 通道旋转属性	DevID	设备号。有效范围：[0, VI_MAX_DEV_NUM)。
	ChnID	通道号。
	Rotate	通道旋转角度。
VI CHN EARLYINTERRUPT INFO 提前上报中断信息	DevID VI_MAX	设备号。有效范围：[0, VI_MAX_DEV_NUM)。
	ChnID	通道号。
	Enable	是否使能提前上报中断的功能。N：关闭；Y：打开。
	LineCnt	提前上报中断的行数。
VI CHN CROP INFOVI 通道裁剪属性	DevID	设备号。有效范围：[0, VI_MAX_DEV_NUM)。
	ChnID	通道号。

下页继续

表 12.4 – 续上页

参数		描述
	CropEn	是否使能 CROP 功能。(cvi_isp_s_selection, V4L2_SEL_TGT_CROP) N：关闭；Y：打开。
	CoorType	坐标类型。RAT：相对坐标；ABS：绝对坐标。
	CoorX	水平方向起始坐标。坐标类型为相对坐标时，合法取值范围为 [0, 999]；坐标类型为绝对坐标时，合法取值范围为 [0, VI_CHN_MAX_WIDTH]。
	CoorY	垂直方向起始坐标。坐标类型为相对坐标时，合法取值范围为 [0, 999]；坐标类型为绝对坐标时，合法取值范围为 [0, VI_CHN_MAX_HEIGHT]。
	Width	CROP RECT 的宽，不能超出最大图像宽度。
	Height	CROP RECT 的高，不能超出最大图像高度。
	TrimX	实际图像起始点坐标。
	TrimY	实际图像起始点坐标。
	TrimWid	实际图像宽，单位：像素。
	TrimHgt	实际图像高，单位：像素。
VI CHN STATUS 通道状态信息	DevID	设备号。有效范围：[0, VI_MAX_DEV_NUM)。
	ChnID	通道号。
	Enable	通道使能。0：不使能；1：使能。
	FrameRate	帧率。
	IntCnt	通道中断计数。
	RecvPic	接收到的图像数目。
	LostFrame	通道丢帧数。
	VbFail	通道获取 VB 失败计数。
	Width	通道宽度。
	Height	通道高度。

12.16 VO

【调试信息】

```
# cat /proc/cvitek/vo

------------------------------DEVICE CONFIG------------------------------
   DevID     DevEn          IntfType          IntfSync          BkClr    DevFrt
    # 0         Y              MIPI          720x1280@60         3FF       106

------------------------------VIDEO LAYER STATUS 1-----------------------
 LayerId   VideoEn           PixFmt      ImgW      ImgH     DispX     DispY     DispW     DispH
↪DispFrt
    # 0        Y     YUV_PLANAR_420       720      1280         0         0       720      1280
↪     25

------------------------------VIDEO LAYER STATUS 2 (continue)------------
 LayerId     DevId   EnChNum      Luma      Cont       Hue      Satu    BufLen
    # 0         0         1               128       128         0       128         3

------------------------------CHN BASIC INFO-----------------------------
 LayerId     ChnId     ChnEn      Prio      ChnX      ChnY      ChnW      ChnH  RotAngle
    # 0      # 0         Y            0         0         0       720      1280        90

------------------------------CHN PLAY INFO------------------------------
 LayerId     ChnId      Show     Pause    Thrshd    ChnFrt        ChnGap(us)   DispPts
↪PreDonePts
    # 0      # 0         N         N         3           24           41666      1580552617
↪  1580472618
```

【调试信息分析】

记录当前 VO 的使用状况及其属性配置，包含设备状态、视频层状态和通道状态。可用于动态获取当前 VO 的使用状态以便于调试或测试。

【参数说明】

参数		描述
DEVICE CONFIG	DevID	设备 ID。 取值范围：[0,VO_MAX_DEV_NUM)
	DevEn	设备是否使能。 N：禁止； Y：使能。
	IntfType	接口类型。 取值范围： CVBS, YPBPR, VGA, BT656, BT1120, LCD, LCD_18BIT, LCD_24BIT, LCD_30BIT, MIPI, MIPI_SLAVE, HDMI, I80

下页继续

表 12.5 – 续上页

参数		描述
	IntfSync	接口时序。 取值范围：[0,VO_OUTPUT_BUTT)。
	BkClr	设备背景色。十六进制 RGB888 格式。
	DevFrt	设备帧率，即刷新率，与时序相关。
VIDEO LAYER STATUS 1	LayerId	视频层 ID。 取值范围：[0,VO_MAX_LAYER_NUM)。
	VideoEn	视频层是否使能。 N：禁止； Y：使能。
	PixFmt	输入图像像素格式。
	ImgW	视频层画布宽度。
	ImgH	视频层画布高度。
	DispX	显示区域起始横坐标。
	DispY	显示区域起始纵坐标。
	DispW	显示区域宽度。
	DispH	显示区域高度。
	DispFrt	视频层显示帧率。
VIDEO LAYER STATUS 2(continue)	LayerId	视频层 ID。 取值范围：[0,VO_MAX_LAYER_NUM)。
	DevId	视频层绑定的设备 ID。 取值范围：[0,VO_MAX_DEV_NUM)。
	EnChNum	通道使能计数。即该视频层有多少个通道处于使能状态。 取值范围：[0,VO_MAX_CHN_NUM)。
	Luma	亮度。 取值范围：[0, 100]。
	Cont	对比度。 取值范围：[0, 100]。
	Hue	色调。 取值范围：[0, 100]。
	Satu	饱和度。 取值范围：[0, 100]。
	BufLen	显示 buffer 长度。
CHN BASIC INFO	LayerId	视频层 ID。 取值范围：[0,VO_MAX_LAYER_NUM)。
	ChnId	通道 ID。 取值范围：[0,VO_MAX_CHN_NUM)。

下页继续

表 12.5 – 续上页

参数		描述
	ChnEn	通道是否使能。 Y：是； N：否。
	Prio	通道优先级。 取值范围：[0,VO_MAX_CHN_NUM)。
	ChnX	通道起始横坐标。
	ChnY	通道起始纵坐标。
	ChnW	通道宽度。
	ChnH	通道高度。
	RotAngle	通道旋转角度。 取值范围：[0,ROTATION_BUTT)
CHN PLAY INFO	LayerId	视频层 ID。 取值范围：[0,VO_MAX_LAYER_NUM)。
	ChnId	通道 ID。 取值范围：[0,VO_MAX_CHN_NUM)。
	Show	通道是否显示。 N：隐藏； Y：显示。
	Pause	通道是否暂停。 N：禁止； Y：使能。
	Thrshd	通道缓冲队列最大可接收的图像帧数。
	ChnFrt	通道帧率。通道播放控制可以通过该数值反映。
	ChnGap(us)	通道帧间隔。与通道帧率成反比。单位：us。
	DispPts	当前正在显示帧的时间戳。单位：us。
	PreDonePts	前一张完成显示帧的时间戳。单位：us。

12.17 VPSS

【调试信息】

```
# cat /proc/cvitek/vpss

Module: [VPSS], Build Time[#1 SMP PREEMPT Wed Feb 24 15:02:47 CST 2021]

----------------------------MODULE PARAM----------------------------
```

(下页继续)

(续上页)

```
        vpss_vb_source      vpss_split_node_num
             0                    1
-------------------------------VPSS MODE---------------------------------
             vpss_mode          dev0          dev1
               single                N      input_mem

-------------------------------VPSS GRP ATTR-----------------------------------------
   GrpID      MaxW      MaxH       PixFmt              SrcFRate  DstFRate  dev
    # 0       1920      1080     YUV_PLANAR_420         -1        -1          0

-------------------------------VPSS CHN ATTR-----------------------------
   GrpID  PhyChnID    Enable  MirrorEn    FlipEn  SrcFRate  DstFRate
   Depth    Aspect    videoX    videoY    videoW    videoH     BgColor
    # 0      # 0         Y         N         N        30         30
      0     AUTO         0         0         0         0         0x0
    # 0      # 1         Y         N         N        30         30
      0     AUTO         0         0         0         0         0x0
    # 0      # 2         N         N         N         0          0
      0     NONE         0         0         0         0         0x0
    # 0      # 3         N         N         N         0          0
      0     NONE         0         0         0         0         0x0

-------------------------------VPSS GRP CROP INFO-------------------------
   GrpID    CropEn  CoorType     CoorX     CoorY     Width     Height
    # 0        N       RAT          0         0         0         0

-------------------------------VPSS CHN CROP INFO-------------------------
   GrpID     ChnID    CropEn  CoorType     CoorX     CoorY     Width     Height
    # 0      # 0        N       RAT          0         0         0         0
    # 0      # 1        N       RAT          0         0         0         0
    # 0      # 2        N       RAT          0         0         0         0
    # 0      # 3        N       RAT          0         0         0         0

-------------------------------VPSS GRP WORK STATUS-----------------------------------------------------
   GrpID   RecvCnt   LostCnt       StartFailCnt    bStart       CostTime(us)     MaxCostTime(us)
HwCostTime(us)   HwMaxCostTime(us)
    # 0      905        0              0              Y             7241              9007
   7029                 7038

-------------------------------VPSS CHN OUTPUT RESOLUTION-----------------
   GrpID     ChnID    Enable     Width     Height          Pixfmt          Videofmt    SendOK␣
↪FrameRate
    # 0      # 0        Y         1920      1080     YUV_PLANAR_420    LINEAR      905
24
    # 0      # 1        Y         1280      720      YUV_PLANAR_420    LINEAR      905
24
    # 0      # 2        N         0         0           RGB_888         LINEAR       0
0
    # 0      # 3        N         0         0           RGB_888         LINEAR       0
0

-------------------------------VPSS CHN ROTATE INFO----------------------
   GrpID     ChnID    Rotate
    # 0      # 0        0
```

(下页继续)

(续上页)

```
    # 0       # 1       0
    # 0       # 2       0
    # 0       # 3       0

------------------------------VPSS CHN LDC INFO-----------------------
   GrpID     ChnID    Enable    Aspect    XRatio    YRatio
 XYRatio   XOffset   YOffset    DistortionRatio
    # 0      # 0        N          N          0         0
       0        0        0              0
    # 0      # 1        N          N          0         0
       0        0        0              0
    # 0      # 2        N          N          0         0
       0        0        0              0
    # 0      # 3        N          N          0         0
       0        0        0              0
------------------------------DRV WORK STATUS------------------------------
        dev        IspTrigCnt0        IspTrigCnt1     IspTrigFailCnt0     IspTrigFailCnt1
 UserTrigCnt     UserTrigFailCnt           IrqCnt0           IrqCnt1
        # 0              0              0              0              0
           0              0              0              0
        # 1              0              0              0              0
           905            0            905              0

------------------------------VPSS CHN BUF WRAP ATTR---------------------
   GrpID     ChnID    Enable   BufLineWrapBufSize
    # 0      # 0       N       0       0
    # 0      # 1       N       0       0
    # 0      # 2       N       0       0
# 0      # 3       N       0       0
```

【调试信息分析】

记录当前 VPSS 属性配置以及状态信息。

【参数说明】

参数		描述
MODULE PARAM	vpss_vb_source	视频缓存池类型。 0：公共 VB 1：Reserve 2：UserVB
	vpss_split_node_num	分块节点数量。
VPSS MODE	vpss_mode	v pss 模式，single 或者 dualmode。
	dev0	vpss dev0 的输入来源，isp 或者 memory。
	dev1	vpss dev1 的输入来源，isp 或者 memory。singlemode 情况下只有 dev1。

下页继续

表 12.6 – 续上页

参数		描述
VPSS GRP ATTRVPSS GRP 属性。	GrpID	GRP ID 号。 有效范围: [0, VPSS_MAX_GRP_NUM)。
	MaxW	组输入图像最大宽度。
	MaxH	组输入图像最大高度。
	PixFmt	组输入图像像素格式。
	SrcFRate	GRP 源帧率。
	DstFRate	GRP 目标帧率。
	dev	Gro up 使用硬件 dev 号，singlemode 下无效，默认显示 0。
VPSS CHN ATTR VPSS 各物理通道属性	GrpID	GRP ID 号。 有效范围: [0, X_GRP_NUM)。
	PhyChnID	物理通道 ID 号。有效范围: [0, VPSS_MAX_PHY_CHN_NUM)。
	Enable	是否使能该通道。 N: 关闭; Y: 打开。
	MirrorEn	是否使能 mirror 功能。 N: 关闭; Y: 打开。
	FlipEn	是否使能 flip 功能。 N: 关闭; Y: 打开。
	SrcFRate	通道帧率控制: 源帧率。
	DstFRate	通道帧率控制: 目标帧率。
	Depth	用户获取通道图像的队列长度。
	Aspect	幅形比模式。NONE: 关闭幅形比 AUTO: 自动模式 MANUAL: 手动模式
	videoX	视频位置 X 坐标。手动模式有效。
	videoY	视频位置 Y 坐标。手动模式有效。
	videoW	视频宽度。手动模式有效。
	videoH	视频高度。手动模式有效。
	BgColor	幅形比背景色。有效范围: [0x0, 0xFFFFFF]
VPSS GRP CROP INFOVPSS 组 CROP 信息	GrpID V	GRP ID 号。有效范围: [0, S_MAX_GRP_NUM)。
	CropEn	是否使能 CROP 功能。N: 关闭; Y: 打开。

下页继续

表 12.6 – 续上页

参数		描述
	CoorType	坐标类型。RAT：相对坐标；ABS：绝对坐标。
	CoorX	水平方向起始坐标。
	CoorY	垂直方向起始坐标。
	Width	CROP RECT 的宽，不能超出最大图像宽度。
	Height	CROP RECT 的高，不能超出最大图像高度。
VPSS CHN CROP INFO 通道 CROP 信息	GrpID VP	ID 号。有效范围：[0, _MAX_GRP_NUM)。
	ChnID	CHN ID 号。有效范围：[0, VPSS_MAX_PHY_CHN_NUM)。
	CropEn	是否使能 CROP 功能。N：关闭；Y：打开。
	CoorType	坐标类型。RAT：相对坐标；ABS：绝对坐标
	CoorX	水平方向起始坐标。
	CoorY	垂直方向起始坐标。
	Width	CROP RECT 的宽，不能超出最大图像宽度。
	Height	CROP RECT 的高，不能超出最大图像高度。
VPSS GRP WORKSTATUSVPSSGRP 工作状态信息	GrpID	GRP ID 号。有效范围：[0, VPSS_MAX_GRP_NUM)。
	RecvCnt	接收到的图像数
	LostCnt	因队列满而丢弃的图像数。
	StartFailCnt	Start 任务失败次数。
	bStart	是否开始接收图像。
	CostTime(us)	当前完成任务的耗时。
	MaxCostTime(us)	历史上耗时最长任务的执行时间。
	HwCostTime(us)	当前完成任务的硬件耗时
	HwMaxCostTime(us)	历史上硬件耗时最长任务的执行时间。
VPSS CHN OUTPUTRESOLUTIONVPSS 通道输出分辨率	GrpID	GRP ID 号。有效范围：[0, VPSS_MAX_GRP_NUM)。
	ChnID	通道 ID 号。有效范围：[0, VPSS_MAX_PHY_CHN_NUM)。
	Enable	通道使能。N：关闭 Y：打开。
	Width	目标图像的宽度，以像素为单位。
	Height	目标图像的高度，以像素为单位。
	Pixfmt	目标图像的像素格式。

下页继续

表 12.6 – 续上页

参数		描述
	Videofmt	目标图像的视频格式。
	SendOK	成功发送的图像数量。
	FrameRate	通道输出的实时帧率。
VPSS CHN ROTATEINFO 通道旋转的信息	GrpID	GRP ID 号。有效范围：[0, VPSS_MAX_GRP_NUM)。
	ChnID	通道 ID 号。有效范围：[0, VPSS_MAX_PHY_CHN_NUM)。
	Rotate	旋转的角度枚举。
VPSS CHN LDC INFOVPSS 物理通道 LDC 属性	GrpID VPSS	GRP ID 号。有效范围：[0, AX_GRP_NUM)。
	ChnID	通道 ID 号。有效范围：[0, VPSS_MAX_PHY_CHN_NUM)。
	Enable	LDC 开关。N：关闭；Y：打开。
	Aspect	是否保持幅形比。N：不保持幅形比；Y：保持幅形比。
	XRatio	不保持幅形比时候生效。水平方向上的 Crop 比例。
	YRatio	不保持幅形比时候生效。垂直方向上的 Crop 比例。
	XYRatio	保持幅形比时候生效。水平和垂直方向上的整体 Crop 比例。
	XOffset	校正中心点的 X 坐标偏移。
	YOffset	校正中心点的 Y 坐标偏移。
	DistortionRatio	校正强度系数。
DRV WORK STATUSDriver 工作状态	dev	vpss 硬件 dev 编号
	IspTrigCnt0	Online 下 isp trigger vpss 次数, 针对 sensor0 图像。
	IspTrigCnt1	Online 下 isp trigger vpss 次数, 针对 sensor1 图像。
	IspTrigFailCnt0	Online 下 isp trigger vpss 失败次数, 针对 sensor0 图像。
	IspTrigFailCnt1	Online 下 isp trigger vpss 失败次数, 针对 sensor1 图像。
	UserTrigCnt	Offline 下 vpss trigger 次数。
	UserTrigFailCnt	Offline 下 vpss trigger 失败次数。
	IrqCnt0	中断次数，针对 sensor0。
	IrqCnt1	中断次数，针对 sensor1。
VPSS CHN BUF WRAPATTRvpss chn 的 slice buf 属性	GrpID	GRP ID 号。
	ChnID	CHN ID 号。
	Enable	是否开启。
	BufLineWrapBufSize	Slice buf 的大小。