



CV180x/CV181x MEDIA PROCESSING SOFTWARE DEVELOPMENT REFERENCE

Version: 1.0.1.1

Release date: 2022-08-16

Copyright © 2020 CVITEK Co., Ltd. All rights reserved.
No part of this document may be reproduced or transmitted in any form or by any means
without prior written consent of CVITEK Co., Ltd.

CONTENTS

1	Disclaimer	2
2	System Overview	3
2.1	Function Overview	3
2.1.1	Objective	3
2.1.2	Definitions and Abbreviations	3
2.2	Design Overview	4
2.2.1	System Architecture	4
3	System Control	6
3.1	Function Overview	6
3.1.1	Objective	6
3.1.2	Definitions and Abbreviations	6
3.2	Design Overview	7
3.2.1	Video Memory Block Pool	7
3.2.2	System Binding	9
3.2.3	The Working Mode of VI and VPSS	9
3.2.4	The Working Mode of VPSS	10
3.2.5	Alignment Requirements For The Video Pipeline	10
3.3	API Reference	11
3.3.1	CVI_SYS_Init	12
3.3.2	CVI_SYS_Exit	13
3.3.3	CVI_SYS_Bind	14
3.3.4	CVI_SYS_UnBind	14
3.3.5	CVI_SYS_GetBindbyDest	15
3.3.6	CVI_SYS_GetBindbySrc	16
3.3.7	CVI_SYS_GetVersion	17
3.3.8	CVI_SYS_GetChipId	17
3.3.9	CVI_SYS_Mmap	18
3.3.10	CVI_SYS_MmapCache	19
3.3.11	CVI_SYS_Munmap	19
3.3.12	CVI_SYS_IonAlloc	20
3.3.13	CVI_SYS_IonAlloc_Cached	21
3.3.14	CVI_SYS_IonFlushCache	22
3.3.15	CVI_SYS_IonInvalidateCache	22
3.3.16	CVI_SYS_IonFree	23
3.3.17	CVI_SYS_SetVIVPSSMode	24
3.3.18	CVI_SYS_GetVIVPSSMode	24
3.3.19	CVI_SYS_SetVPSSMode	25
3.3.20	CVI_SYS_GetVPSSMode	26
3.3.21	CVI_SYS_GetModName	26
3.3.22	CVI_VB_SetConfig	27
3.3.23	CVI_VB_GetConfig	28
3.3.24	CVI_VB_Init	28

3.3.25	CVI_VB_Exit	29
3.3.26	CVI_VB_GetBlock	30
3.3.27	CVI_VB_ReleaseBlock	31
3.3.28	CVI_VB_CreatePool	31
3.3.29	CVI_VB_DestroyPool	32
3.3.30	CVI_VB_PhysAddr2Handle	33
3.3.31	CVI_VB_Handle2PhysAddr	33
3.3.32	CVI_VB_Handle2PoolId	34
3.3.33	CVI_VB_InquireUserCnt	35
3.3.34	CVI_VB_MmapPool	35
3.3.35	CVI_VB_MunmapPool	36
3.3.36	CVI_VB_GetBlockVirAddr	37
3.3.37	CVI_LOG_SetLevelConf	37
3.3.38	CVI_LOG_GetLevelConf	38
3.4	Data Types	39
3.4.1	Base Type	39
3.4.2	MOD_ID_E	39
3.4.3	VB_SOURCE_E	40
3.4.4	ROTATION_E	41
3.4.5	POINT_S	42
3.4.6	SIZE_S	42
3.4.7	RECT_S	43
3.4.8	LDC_ATTR_S	43
3.4.9	MMF_CHN_S	44
3.4.10	MMF_BIND_DEST_S	45
3.4.11	MMF_VERSION_S	45
3.4.12	VB_CONFIG_S	46
3.4.13	VB_POOL_CONFIG_S	46
3.4.14	VI_VPSS_MODE_E	47
3.4.15	VPSS_MODE_E	47
3.4.16	ASPECT_RATIO_E	48
3.4.17	ASPECT_RATIO_S	48
3.4.18	PIXEL_FORMAT_E	49
3.4.19	VIDEO_FRAME_S	50
3.4.20	VIDEO_FRAME_INFO_S	51
3.4.21	BITMAP_S	52
3.5	Error Codes	52
4	Video Input	54
4.1	Function Overview	54
4.1.1	Objective	54
4.1.2	Definitions and Abbreviations	54
4.2	Design Overview	54
4.2.1	System Architecture	54
4.2.2	Video Input PIPE	55
4.2.3	Video Physical Channel	55
4.2.4	Binding Relationship	55
4.3	API Reference	56
4.3.1	CVI_VI_SetDevAttr	57
4.3.2	CVI_VI_GetDevAttr	59
4.3.3	CVI_VI_SetDevAttrEx	60
4.3.4	CVI_VI_GetDevAttrEx	60
4.3.5	CVI_VI_EnableDev	61
4.3.6	CVI_VI_DisableDev	62
4.3.7	CVI_VI_SetDevBindPipe	62
4.3.8	CVI_VI_GetDevBindPipe	63
4.3.9	CVI_VI_SetDevTimingAttr	64
4.3.10	CVI_VI_GetDevTimingAttr	65

4.3.11	CVI_VI_CreatePipe	65
4.3.12	CVI_VI_DestroyPipe	66
4.3.13	CVI_VI_SetPipeAttr	67
4.3.14	CVI_VI_GetPipeAttr	67
4.3.15	CVI_VI_StartPipe	68
4.3.16	CVI_VI_StopPipe	69
4.3.17	CVI_VI_SetPipeCrop	69
4.3.18	CVI_VI_GetPipeCrop	70
4.3.19	CVI_VI_SetPipeDumpAttr	71
4.3.20	CVI_VI_GetPipeDumpAttr	71
4.3.21	CVI_VI_SetPipeFrameSource	72
4.3.22	CVI_VI_GetPipeFrameSource	73
4.3.23	CVI_VI_GetPipeFrame	73
4.3.24	CVI_VI_ReleasePipeFrame	74
4.3.25	CVI_VI_SendPipeRaw	75
4.3.26	CVI_VI_QueryPipeStatus	76
4.3.27	CVI_VI_GetPipeFd	76
4.3.28	CVI_VI_CloseFd	77
4.3.29	CVI_VI_AttachVbPool	77
4.3.30	CVI_VI_DetachVbPool	78
4.3.31	CVI_VI_SetChnAttr	79
4.3.32	CVI_VI_GetChnAttr	80
4.3.33	CVI_VI_EnableChn	80
4.3.34	CVI_VI_DisableChn	81
4.3.35	CVI_VI_SetChnCrop	82
4.3.36	CVI_VI_GetChnCrop	82
4.3.37	CVI_VI_GetChnFrame	83
4.3.38	CVI_VI_ReleaseChnFrame	84
4.3.39	CVI_VI_SetChnRotation	85
4.3.40	CVI_VI_GetChnRotation	86
4.3.41	CVI_VI_SetChnLDCAttr	87
4.3.42	CVI_VI_GetChnLDCAttr	87
4.3.43	CVI_VI_RegChnFlipMirrorCallBack	88
4.3.44	CVI_VI_UnRegChnFlipMirrorCallBack	89
4.3.45	CVI_VI_SetChnFlipMirror	90
4.3.46	CVI_VI_GetChnFlipMirror	90
4.4	Data Types	91
4.4.1	VI_MAX_DEV_NUM	92
4.4.2	VI_MAX_PHY_PIPE_NUM	93
4.4.3	VI_MAX_VIR_PIPE_NUM	93
4.4.4	VI_MAX_PIPE_NUM	93
4.4.5	VI_MAX_PHY_CHN_NUM	94
4.4.6	VI_MAX_CHN_NUM	94
4.4.7	VI_DEV_MIN_WIDTH	94
4.4.8	VI_DEV_MIN_HEIGHT	95
4.4.9	VI_DEV_MAX_WIDTH	95
4.4.10	VI_DEV_MAX_HEIGHT	95
4.4.11	VI_PIPE_OFFLINE_MIN_WIDTH	96
4.4.12	VI_PIPE_OFFLINE_MIN_HEIGHT	96
4.4.13	VI_PIPE_OFFLINE_MAX_WIDTH	96
4.4.14	VI_PIPE_OFFLINE_MAX_HEIGHT	97
4.4.15	VI_PIPE_ONLINE_MIN_WIDTH	97
4.4.16	VI_PIPE_ONLINE_MIN_HEIGHT	97
4.4.17	VI_PIPE_ONLINE_MAX_WIDTH	98
4.4.18	VI_PIPE_ONLINE_MAX_HEIGHT	98
4.4.19	VI_PIPE0_MAX_WIDTH	98
4.4.20	VI_PIPE0_MAX_HEIGHT	99
4.4.21	VI_PIPE1_MAX_WIDTH	99

4.4.22	VI_PIPE1_MAX_HEIGHT	99
4.4.23	VI_PIPE2_MAX_WIDTH	100
4.4.24	VI_PIPE2_MAX_HEIGHT	100
4.4.25	VI_PIPE3_MAX_WIDTH	100
4.4.26	VI_PIPE3_MAX_HEIGHT	101
4.4.27	VI_PIPE4_MAX_WIDTH	101
4.4.28	VI_PIPE4_MAX_HEIGHT	101
4.4.29	VI_DATA_TYPE_E	102
4.4.30	VI_DEV_ATTR_S	102
4.4.31	VI_DEV_BIND_PIPE_S	103
4.4.32	VI_PIPE_ATTR_S	104
4.4.33	VI_DUMP_TYPE_E	105
4.4.34	VI_DUMP_ATTR_S	106
4.4.35	VI_CHN_ATTR_S	107
4.4.36	VI_CROP_INFO_S	108
4.4.37	VI_DEV_TIMING_ATTR_S	108
4.4.38	VI_PIPE_STATUS_S	109
4.4.39	VI_CHN_STATUS_S	109
4.4.40	VI_PIPE_FRAME_SOURCE_E	110
4.4.41	VI_LDC_ATTR_S	110
4.5	Error Codes	111
5	Video Output	112
5.1	Function overview	112
5.1.1	Objective	112
5.1.2	Definitions and Abbreviations	112
5.2	Design Overview	112
5.2.1	System Architecture	112
5.3	API Reference	115
5.3.1	CVI_VO_Enable	116
5.3.2	CVI_VO_Disable	117
5.3.3	CVI_VO_SetPubAttr	118
5.3.4	CVI_VO_GetPubAttr	118
5.3.5	CVI_VO_EnableVideoLayer	119
5.3.6	CVI_VO_DisableVideoLayer	120
5.3.7	CVI_VO_SetVideoLayerAttr	120
5.3.8	CVI_VO_GetVideoLayerAttr	121
5.3.9	CVI_VO_EnableChn	122
5.3.10	CVI_VO_DisableChn	122
5.3.11	CVI_VO_SetChnAttr	123
5.3.12	CVI_VO_GetChnAttr	124
5.3.13	CVI_VO_ShowChn	124
5.3.14	CVI_VO_HideChn	125
5.3.15	CVI_VO_SetChnRotation	126
5.3.16	CVI_VO_GetChnRotation	127
5.3.17	CVI_VO_PauseChn	127
5.3.18	CVI_VO_ResumeChn	128
5.3.19	CVI_VO_CloseFd	129
5.4	Data Types	129
5.4.1	VO_DEV	129
5.4.2	VO_LAYER	130
5.4.3	VO_INTF_TYPE	131
5.4.4	VO_INTF_SYNC_E	131
5.4.5	VO_SYNC_INFO_S	132
5.4.6	VO_PUB_ATTR_S	133
5.4.7	VO_VIDEO_LAYER_ATTR_S	134
5.4.8	VO_CHN_ATTR_S	135
5.5	Error Codes	135

6	Video Processing Subsystem	136
6.1	Function Overview	136
6.1.1	Objective	136
6.1.2	Definitions and Abbreviations	136
6.2	Design Overview	136
6.2.1	System Architecture	136
6.2.2	Note	138
6.3	API Reference	139
6.3.1	CVI_VPSS_CreateGrp	140
6.3.2	CVI_VPSS_DestroyGrp	142
6.3.3	CVI_VPSS_GetGrpAttr	143
6.3.4	CVI_VPSS_SetGrpAttr	143
6.3.5	CVI_VPSS_StartGrp	144
6.3.6	CVI_VPSS_StopGrp	145
6.3.7	CVI_VPSS_ResetGrp	145
6.3.8	CVI_VPSS_GetGrpProcAmpCtrl	146
6.3.9	CVI_VPSS_GetGrpProcAmp	147
6.3.10	CVI_VPSS_SetGrpProcAmp	148
6.3.11	CVI_VPSS_SetGrpParamfromBin	149
6.3.12	CVI_VPSS_GetChnAttr	150
6.3.13	CVI_VPSS_SetChnAttr	151
6.3.14	CVI_VPSS_EnableChn	152
6.3.15	CVI_VPSS_DisableChn	152
6.3.16	CVI_VPSS_SetGrpCrop	153
6.3.17	CVI_VPSS_GetGrpCrop	154
6.3.18	CVI_VPSS_SendFrame	155
6.3.19	CVI_VPSS_GetChnFrame	155
6.3.20	CVI_VPSS_SendChnFrame	157
6.3.21	CVI_VPSS_ReleaseChnFrame	157
6.3.22	CVI_VPSS_GetGrpFrame	158
6.3.23	CVI_VPSS_ReleaseGrpFrame	158
6.3.24	CVI_VPSS_SetChnCrop	158
6.3.25	CVI_VPSS_GetChnCrop	159
6.3.26	CVI_VPSS_SetChnRotation	160
6.3.27	CVI_VPSS_GetChnRotation	161
6.3.28	CVI_VPSS_SetChnLDCAttr	162
6.3.29	CVI_VPSS_GetChnLDCAttr	163
6.3.30	CVI_VPSS_GetChnFd	163
6.3.31	CVI_VPSS_CloseFd	164
6.3.32	CVI_VPSS_AttachVbPool	165
6.3.33	CVI_VPSS_DetachVbPool	166
6.3.34	CVI_VPSS_SetChnBufWrapAttr	166
6.3.35	CVI_VPSS_GetChnBufWrapAttr	167
6.3.36	CVI_VPSS_GetWrapBufferSize	168
6.4	Data Types	169
6.4.1	VPSS_MAX_GRP_NUM	169
6.4.2	VPSS_MAX_CHN_NUM	170
6.4.3	VPSS_MAX_PHY_CHN_NUM	170
6.4.4	VPSS_MIN_IMAGE_WIDTH	170
6.4.5	VPSS_MIN_IMAGE_HEIGHT	171
6.4.6	VPSS_MAX_IMAGE_WIDTH	171
6.4.7	VPSS_MAX_IMAGE_HEIGHT	171
6.4.8	VPSS_MAX_ZOOMIN	172
6.4.9	VPSS_MAX_ZOOMOUT	172
6.4.10	VPSS_GRP	172
6.4.11	VPSS_CHN	173
6.4.12	VPSS_ROUNDING_E	173
6.4.13	VPSS_CROP_COORDINATE_E	174

6.4.14	VPSS_NORMALIZE_S	174
6.4.15	VPSS_CROP_INFO_S	175
6.4.16	VPSS_GRP_ATTR_S	176
6.4.17	VPSS_CHN_ATTR_S	176
6.4.18	VPSS_MOD_PARAM_S	177
6.4.19	PROC_AMP_E	177
6.4.20	PROC_AMP_CTRL_S	178
6.4.21	VPSS_LDC_ATTR_S	179
6.5	Error Codes	179
7	Video Encoding	180
7.1	Function Overview	180
7.1.1	Objective	180
7.1.2	Definitions and Abbreviations	180
7.2	Design Overview	181
7.2.1	Flowchart of Video Encoding Data	181
7.2.2	VENC Channels	181
7.2.3	Rate Control	182
7.2.4	Fixed QP	182
7.2.5	CBR	182
7.2.6	VBR	183
7.2.7	AVBR	183
7.2.8	GOP Structure	183
7.2.9	Advanced Frame Skipping	185
7.2.10	Cropping Encoding	185
7.2.11	ROI	186
7.2.12	Coding Unit (CU) Slice Mode	186
7.2.13	Multi-Encoder Parallel Encoding	187
7.2.14	The Encoding Frame Buffer Calculation	187
7.3	API Reference	187
7.3.1	CVI_VENC_CreateChn	189
7.3.2	CVI_VENC_DestroyChn	190
7.3.3	CVI_VENC_ResetChn	191
7.3.4	CVI_VENC_StartRecvFrame	191
7.3.5	CVI_VENC_StopRecvFrame	193
7.3.6	CVI_VENC_QueryStatus	193
7.3.7	CVI_VENC_SetChnAttr	194
7.3.8	CVI_VENC_GetChnAttr	195
7.3.9	CVI_VENC_GetStream	195
7.3.10	CVI_VENC_ReleaseStream	197
7.3.11	CVI_VENC_SendFrame	198
7.3.12	CVI_VENC_GetFd	200
7.3.13	CVI_VENC_CloseFd	201
7.3.14	CVI_VENC_SetJpegParam	202
7.3.15	CVI_VENC_GetJpegParam	203
7.3.16	CVI_VENC_SetRcParam	204
7.3.17	CVI_VENC_GetRcParam	205
7.3.18	CVI_VENC_SetChnParam	206
7.3.19	CVI_VENC_GetChnParam	207
7.3.20	CVI_VENC_RequestIDR	208
7.3.21	CVI_VENC_SetRoiAttr	208
7.3.22	CVI_VENC_GetRoiAttr	210
7.3.23	CVI_VENC_SetRefParam	210
7.3.24	CVI_VENC_GetRefParam	211
7.3.25	CVI_VENC_SetFrameLostStrategy	212
7.3.26	CVI_VENC_GetFrameLostStrategy	213
7.3.27	CVI_VENC_SetModParam	214
7.3.28	CVI_VENC_GetModParam	214

7.3.29	CVI_VENC_AttachVbPool	215
7.3.30	CVI_VENC_DetachVbPool	216
7.3.31	CVI_VENC_GetH264Entropy	216
7.3.32	CVI_VENC_SetH264Entropy	217
7.3.33	CVI_VENC_InsertUserData	218
7.3.34	CVI_VENC_GetCuPrediction	218
7.3.35	CVI_VENC_SetCuPrediction	219
7.3.36	CVI_VENC_GetH264Trans	220
7.3.37	CVI_VENC_SetH264Trans	220
7.3.38	CVI_VENC_SetH265Trans	221
7.3.39	CVI_VENC_GetH265Trans	222
7.4	Data Types	222
7.4.1	VENC_MAX_CHN_NUM	223
7.4.2	VENC_CHN_PARAM_S	224
7.4.3	VENC_PACK_S	224
7.4.4	VENC_STREAM_S	225
7.4.5	VENC_GOP_ATTR_S	226
7.4.6	VENC_GOP_NORMALP_S	227
7.4.7	VENC_GOP_SMARTP_S	227
7.4.8	VENC_RECV_PIC_PARAM_S	228
7.4.9	VENC_CHN_ATTR_S	228
7.4.10	VENC_ATTR_S	229
7.4.11	VENC_ATTR_H264_S	230
7.4.12	VENC_ATTR_H265_S	230
7.4.13	VENC_STREAM_INFO_S	231
7.4.14	VENC_CHN_STATUS_S	231
7.4.15	VENC_JPEG_PARAM_S	232
7.4.16	VENC_RC_ATTR_S	233
7.4.17	VENC_H264_CBR_S	234
7.4.18	VENC_H264_VBR_S	234
7.4.19	VENC_H264_AVBR_S	235
7.4.20	VENC_H264_FIXQP_S	236
7.4.21	VENC_H264_QPMAP_S	236
7.4.22	VENC_MJPEG_FIXQP_S	237
7.4.23	VENC_MJPEG_CBR_S	238
7.4.24	VENC_H265_CBR_S	238
7.4.25	VENC_H265_VBR_S	239
7.4.26	VENC_H265_AVBR_S	240
7.4.27	VENC_H265_FIXQP_S	240
7.4.28	VENC_H265_QPMAP_S	241
7.4.29	VENC_RC_PARAM_S	242
7.4.30	VENC_PARAM_H264_CBR_S	243
7.4.31	VENC_PARAM_H264_VBR_S	244
7.4.32	VENC_PARAM_H264_AVBR_S	245
7.4.33	VENC_PARAM_H265_CBR_S	246
7.4.34	VENC_PARAM_H265_VBR_S	247
7.4.35	VENC_PARAM_H265_AVBR_S	248
7.4.36	VENC_PARAM_MOD_S	249
7.4.37	VENC_MOD_H264E_S	250
7.4.38	VENC_MOD_H265E_S	251
7.4.39	VENC_MOD_JPEGE_S	251
7.4.40	VENC_CHN_POOL_S	252
7.4.41	VENC_FRAMELOST_S	252
7.4.42	VENC_H264_ENTROPY_S	253
7.4.43	VENC_CU_PREDICTION_S	254
7.4.44	VENC_H264_TRANS_S	254
7.4.45	VENC_H265_TRANS_S	255
7.5	Error Codes	256

8	Video Decoding	258
8.1	Function Overview	258
8.1.1	Objective	258
8.1.2	Definitions and Abbreviations	258
8.2	Design Overview	258
8.2.1	Bitstream Delivery Method	258
8.2.2	Image Output Format	259
8.2.3	Timestamp (PTS) Processing	259
8.2.4	Decoding Frame Buffer Allocation Mode	259
8.3	API Reference	260
8.3.1	CVI_VDEC_CreateChn	260
8.3.2	CVI_VDEC_DestroyChn	262
8.3.3	CVI_VDEC_ResetChn	262
8.3.4	CVI_VDEC_GetChnAttr	263
8.3.5	CVI_VDEC_SetChnAttr	264
8.3.6	CVI_VDEC_StartRecvStream	265
8.3.7	CVI_VDEC_StopRecvStream	266
8.3.8	CVI_VDEC_QueryStatus	267
8.3.9	CVI_VDEC_SetChnParam	268
8.3.10	CVI_VDEC_GetChnParam	269
8.3.11	CVI_VDEC_SendStream	270
8.3.12	CVI_VDEC_GetFrame	271
8.3.13	CVI_VDEC_ReleaseFrame	271
8.3.14	CVI_VDEC_SetModParam	272
8.3.15	CVI_VDEC_GetModParam	273
8.3.16	CVI_VDEC_AttachVbPool	273
8.3.17	CVI_VDEC_DetachVbPool	274
8.4	Data Types	275
8.4.1	VDEC_CHN_ATTR_S	276
8.4.2	VDEC_ATTR_VIDEO_S	276
8.4.3	VIDEO_MODE_E	277
8.4.4	VDEC_CHN_STATUS_S	278
8.4.5	VDEC_DECODE_ERROR_S	279
8.4.6	VDEC_CHN_PARAM_S	279
8.4.7	VDEC_PARAM_VIDEO_S	280
8.4.8	VDEC_PARAM_PICTURE_S	281
8.4.9	VIDEO_DEC_MODE_E	281
8.4.10	VIDEO_OUTPUT_ORDER_E	282
8.4.11	COMPRESS_MODE_E	282
8.4.12	H264_PRTCL_PARAM_S	283
8.4.13	H265_PRTCL_PARAM_S	283
8.4.14	VDEC_PRTCL_PARAM_S	284
8.4.15	VDEC_STREAM_S	284
8.4.16	VDEC_USERDATA_S	285
8.4.17	VIDEO_DISPLAY_MODE_E	285
8.4.18	VDEC_PARAM_MOD_S	286
8.4.19	VDEC_VIDEO_MOD_PARAM_S	287
8.4.20	VDEC_PICTURE_MOD_PARAM_S	288
8.4.21	VDEC_CHN_POOL_S	288
8.5	Error Codes	289
9	Regional Management	290
9.1	Function Overview	290
9.1.1	Objective	290
9.1.2	Definitions and Abbreviations	290
9.2	Design Overview	290
9.2.1	System Architecture	290
9.2.2	Note	291

9.3	API Reference	292
9.3.1	CVI_RGN_Create	292
9.3.2	CVI_RGN_Destroy	293
9.3.3	CVI_RGN_GetAttr	294
9.3.4	CVI_RGN_SetAttr	295
9.3.5	CVI_RGN_SetBitMap	295
9.3.6	CVI_RGN_AttachToChn	296
9.3.7	CVI_RGN_DetachFromChn	297
9.3.8	CVI_RGN_SetDisplayAttr	298
9.3.9	CVI_RGN_GetDisplayAttr	298
9.3.10	CVI_RGN_GetCanvasInfo	299
9.3.11	CVI_RGN_UpdateCanvas	300
9.3.12	CVI_RGN_SetChnPalette	301
9.4	Data Types	302
9.4.1	RGN_TYPE_E	302
9.4.2	RGN_AREA_TYPE_E	302
9.4.3	OSD_COMPRESS_MODE_E	303
9.4.4	OSD_COMPRESS_INFO_S	303
9.4.5	COVER_CHN_ATTR_S	304
9.4.6	COVEREX_CHN_ATTR_S	304
9.4.7	OVERLAY_ATTR_S	305
9.4.8	OVERLAY_CHN_ATTR_S	306
9.4.9	OVERLAYEX_ATTR_S	306
9.4.10	OVERLAYEX_CHN_ATTR_S	307
9.4.11	RGN_ATTR_U	308
9.4.12	RGN_CHN_ATTR_U	308
9.4.13	RGN_ATTR_S	309
9.4.14	RGN_CHN_ATTR_S	309
9.5	Error Codes	310
10	Audio Frequency	311
10.1	Function Overview	311
10.1.1	Objective	311
10.1.2	Definitions and Abbreviations	311
10.2	Design Overview	311
10.2.1	System Architecture	311
10.2.2	Audio Input and Output	314
10.2.2.1	Audio Interface and AI, AO Device	314
10.2.2.2	Principle of Recording and Playing	315
10.2.2.3	Audio Interface Timing	315
10.2.2.4	Resample	316
10.2.2.5	Voice Quality Enhancement (VQE)	316
10.2.3	Audio Encoding and Decoding	322
10.2.3.1	Audio Codec Process	322
10.2.3.2	Audio Codec Protocol	322
10.2.3.3	Speech Frame Structure	322
10.2.4	Built-in Codec	322
10.2.4.1	Overview	322
10.2.4.2	ioctl Function	323
10.3	API Reference	325
10.3.1	Module Properties API:	325
10.3.1.1	CVI_AUDIO_INIT	325
10.3.1.2	CVI_AUDIO_DEINIT	326
10.3.1.3	CVI_AUDIO_SetModParam	326
10.3.1.4	CVI_AUD_SYS_Bind	327
10.3.1.5	CVI_AUDIO_GetModParam	327
10.3.1.6	CVI_AUDIO_RegisterVQEModule	328
10.3.1.7	CVI_AENC_RegisterExternalEncoder	329

10.3.1.8	CVI_AENC_UnRegisterExternalEncoder	329
10.3.1.9	CVI_ADEC_RegisterExternalDecoder	330
10.3.1.10	CVI_ADEC_UnRegisterExternalDecoder	331
10.3.2	Audio Input	331
10.3.2.1	CVI_AI_SetPubAttr	332
10.3.2.2	CVI_AI_GetPubAttr	334
10.3.2.3	CVI_AI_Enable	334
10.3.2.4	CVI_AI_Disable	336
10.3.2.5	CVI_AI_EnableChn	337
10.3.2.6	CVI_AI_DisableChn	338
10.3.2.7	CVI_AI_GetFrame	338
10.3.2.8	CVI_AI_ReleaseFrame	339
10.3.2.9	CVI_AI_SetChnParam	340
10.3.2.10	CVI_AI_GetChnParam	341
10.3.2.11	CVI_AI_EnableReSmp	342
10.3.2.12	CVI_AI_DisableReSmp	343
10.3.2.13	CVI_AI_ClrPubAttr	344
10.3.2.14	CVI_AI_SaveFile	344
10.3.2.15	CVI_AI_QueryFileStatus	345
10.3.2.16	CVI_AI_EnableAecRefFrame	346
10.3.2.17	CVI_AI_DisableAecRefFrame	347
10.3.2.18	CVI_AI_SetVolume	348
10.3.2.19	CVI_AI_GetVolume	348
10.3.3	Voice Quality Enhancement API	349
10.3.3.1	CVI_AI_SetVqeAttr	349
10.3.3.2	CVI_AI_SetTalkVqeAttr	350
10.3.3.3	CVI_AI_GetTalkVqeAttr	351
10.3.3.4	CVI_AI_SetRecordVqeAttr	352
10.3.3.5	CVI_AI_GetRecordVqeAttr	353
10.3.3.6	CVI_AI_EnableVqe	354
10.3.3.7	CVI_AI_DisableVqe	355
10.3.3.8	CVI_AI_SetTrackMode	355
10.3.3.9	CVI_AI_GetTrackMode	356
10.3.3.10	CVI_AO_SetVqeAttr	357
10.3.3.11	CVI_AO_GetVqeAttr	358
10.3.3.12	CVI_AO_EnableVqe	359
10.3.3.13	CVI_VQE_PathSelect	360
10.3.4	Audio Output	360
10.3.4.1	CVI_AO_SetPubAttr	361
10.3.4.2	CVI_AO_GetPubAttr	362
10.3.4.3	CVI_AO_Enable	363
10.3.4.4	CVI_AO_Disable	364
10.3.4.5	CVI_AO_EnableChn	365
10.3.4.6	CVI_AO_DisableChn	365
10.3.4.7	CVI_AO_SendFrame	366
10.3.4.8	CVI_AO_EnableReSmp	367
10.3.4.9	CVI_AO_DisableReSmp	368
10.3.4.10	CVI_AO_PauseChn	369
10.3.4.11	CVI_AO_ResumeChn	369
10.3.4.12	CVI_AO_ClearChnBuf	370
10.3.4.13	CVI_AO_QueryChnStat	371
10.3.4.14	CVI_AO_SetTrackMode	372
10.3.4.15	CVI_AO_GetTrackMode	373
10.3.4.16	CVI_AO_SetVolume	373
10.3.4.17	CVI_AO_GetVolume	374
10.3.4.18	CVI_AO_SetMute	375
10.3.4.19	CVI_AO_GetMute	375
10.3.4.20	CVI_AO_SaveFile	376

10.3.4.21	CVI_AO_ClrPubAttr	376
10.3.5	Audio Encoding	377
10.3.5.1	CVI_AENC_CreateChn	378
10.3.5.2	CVI_AENC_DestroyChn	379
10.3.5.3	CVI_AENC_SendFrame	379
10.3.5.4	CVI_AENC_GetStream	381
10.3.5.5	CVI_AENC_ReleaseStream	382
10.3.5.6	CVI_AENC_SaveFile	383
10.3.5.7	CVI_AENC_QueryFileStatus	383
10.3.5.8	CVI_AENC_GetStreamBuffInfo	384
10.3.5.9	CVI_AENC_SetMute	385
10.3.5.10	CVI_AENC_GetMute	385
10.3.6	Audio Decoding	386
10.3.6.1	CVI_ADEC_CreateChn	386
10.3.6.2	CVI_ADEC_DestroyChn	387
10.3.6.3	CVI_ADEC_SendStream	388
10.3.6.4	CVI_ADEC_ClearChnBuf	388
10.3.6.5	CVI_ADEC_GetFrame	389
10.3.6.6	CVI_ADEC_ReleaseFrame	390
10.3.6.7	CVI_ADEC_SendEndOfStream	391
10.3.7	Built-in Codec	391
10.3.7.1	ACODEC_SOFT_RESET_CTRL	392
10.3.7.2	ACODEC_SET_INPUT_VOL	393
10.3.7.3	ACODEC_GET_INPUT_VOL	394
10.3.7.4	ACODEC_SET_OUTPUT_VOL	394
10.3.7.5	ACODEC_GET_OUTPUT_VOL	395
10.3.7.6	ACODEC_SET_GAIN_MICL	396
10.3.7.7	ACODEC_SET_GAIN_MICR	396
10.3.7.8	ACODEC_SET_DACL_VOL	397
10.3.7.9	ACODEC_SET_DACR_VOL	398
10.3.7.10	ACODEC_SET_ADCL_VOL	399
10.3.7.11	ACODEC_SET_ADCR_VOL	399
10.3.7.12	ACODEC_SET_MICL_MUTE	400
10.3.7.13	ACODEC_SET_MICR_MUTE	401
10.3.7.14	ACODEC_SET_DACL_MUTE	401
10.3.7.15	ACODEC_SET_DACR_MUTE	402
10.3.7.16	ACODEC_GET_GAIN_MICL	403
10.3.7.17	ACODEC_GET_GAIN_MICR	403
10.3.7.18	ACODEC_GET_DACL_VOL	404
10.3.7.19	ACODEC_GET_DACR_VOL	405
10.3.7.20	ACODEC_GET_ADCL_VOL	405
10.3.7.21	ACODEC_GET_ADCR_VOL	406
10.3.7.22	ACODEC_SET_PD_DACL	407
10.3.7.23	ACODEC_SET_PD_DACR	408
10.3.7.24	ACODEC_SET_PD_ADCL	408
10.3.7.25	ACODEC_SET_PD_ADCR	409
10.3.7.26	ACODEC_SET_PD_LINEINL	410
10.3.7.27	ACODEC_SET_PD_LINEINR	411
10.3.8	Resampling	411
10.3.8.1	CVI_Resampler_Create	412
10.3.8.2	CVI_Resampler_Process	412
10.3.8.3	CVI_Resampler_Destroy	413
10.3.8.4	CVI_Resampler_GetMaxOutputNum	414
10.4	Data Types	414
10.4.1	Audio Input / Output	414
10.4.1.1	AIO_MAX_NUM	416
10.4.1.2	AI_DEV_MAX_NUM	416
10.4.1.3	AO_DEV_MAX_NUM	416

10.4.1.4	AI_MAX_CHN_NUM	417
10.4.1.5	AO_MAX_CHN_NUM	417
10.4.1.6	CVI_AUD_MAX_CHANNEL_NUM	417
10.4.1.7	AI_TALKVQE_MASK_AEC	418
10.4.1.8	AI_TALKVQE_MASK_AGC	418
10.4.1.9	AI_TALKVQE_MASK_ANR	418
10.4.1.10	AI_RECORDVQE_MASK_AGC	419
10.4.1.11	MAX_AUDIO_FILE_PATH_LEN	419
10.4.1.12	MAX_AUDIO_FILE_NAME_LEN	419
10.4.1.13	AUDIO_CLKSEL_E	420
10.4.1.14	AUDIO_SAMPLE_RATE_E	420
10.4.1.15	AUDIO_BIT_WIDTH_E	421
10.4.1.16	AIO_MODE_E	422
10.4.1.17	AIO_I2STYPE_E	422
10.4.1.18	AUDIO_SOUND_MODE_E	423
10.4.1.19	AUDIO_MOD_PARAM_S	424
10.4.1.20	AIO_ATTR_S	424
10.4.1.21	AI_CHN_PARAM_S	425
10.4.1.22	AUDIO_FRAME_S	426
10.4.1.23	AEC_FRAME_S	427
10.4.1.24	AUDIO_AGC_CONFIG_S	427
10.4.1.25	AI_AEC_CONFIG_S	428
10.4.1.26	AUDIO_ANR_CONFIG_S	428
10.4.1.27	AUDIO_DELAY_CONFIG_S	429
10.4.1.28	AO_VQE_CONFIG_S	430
10.4.1.29	VQE_WORKSTATE_E	430
10.4.1.30	VQE_RECORD_TYPE	431
10.4.1.31	AI_TALKVQE_CONFIG_S	431
10.4.1.32	AI_RECORDVQE_CONFIG_S	432
10.4.1.33	AUDIO_STREAM_S	433
10.4.1.34	AO_CHN_STATE_S	433
10.4.1.35	AUDIO_TRACK_MODE_E	434
10.4.1.36	AUDIO_FADE_RATE_E	435
10.4.1.37	AUDIO_FADE_S	435
10.4.1.38	G726_BPS_E	436
10.4.1.39	ADPCM_TYPE_E	437
10.4.1.40	ST_CVI_WAV_HEADER	437
10.4.1.41	AUDIO_FILE_STATUS_S	438
10.4.1.42	VQE_MODULE_CONFIG_S	438
10.4.1.43	AUDIO_VQE_REGISTER_S	439
10.4.2	Audio Encoding	440
10.4.2.1	AENC_MAX_CHN_NUM	440
10.4.2.2	AENC_ATTR_G711_S	440
10.4.2.3	AENC_ATTR_G726_S	441
10.4.2.4	AENC_ATTR_ADPCM_S	441
10.4.2.5	AENC_ATTR_LPCM_S	441
10.4.2.6	AENC_CHN_ATTR_S	442
10.4.2.7	AAC_AENC_ENCODER_S	442
10.4.3	Audio Decoding	443
10.4.3.1	MAX_AUDIO_FRAME_NUM	443
10.4.3.2	ADEC_MAX_CHN_NUM	444
10.4.3.3	ADEC_ATTR_G711_S	444
10.4.3.4	ADEC_ATTR_G726_S	444
10.4.3.5	ADEC_ATTR_ADPCM_S	445
10.4.3.6	ADEC_ATTR_LPCM_S	445
10.4.3.7	ADEC_MODE_E	445
10.4.3.8	ADEC_CHN_ATTR_S	446
10.4.3.9	AUDIO_FRAME_INFO_S	447

10.4.3.10	ADEC_CHN_STATE_S	448
10.4.3.11	ADEC_DECODER_S	448
10.4.4	Built-in Codec	449
10.4.4.1	ACODEC_VOL_CTRL	449
10.5	Error Codes	450
10.5.1	Audio Basic Attribute Error Codes	450
10.5.2	Audio Input Error Codes	450
10.5.3	Audio Output Error Codes	451
10.5.4	Audio Encoding Error Codes	452
10.5.5	Audio Decoding Error Codes	453
10.6	Related Tests	454
10.6.1	Unit Test	454
10.6.2	Functional Test	455
10.6.3	Performance Test	456
10.7	Sample Code and Board-side Component Preliminary Testing	456
10.7.1	Description of the Sample Code	456
10.7.2	Preliminary Testing of the Board-side Component	457
11	Geometric Distortion Correction Subsystem	461
11.1	Function Overview	461
11.2	Design Overview	461
11.2.1	System Architecture	461
11.3	API Reference	462
11.3.1	CVI_GDC_BeginJob	462
11.3.2	CVI_GDC_EndJob	463
11.3.3	CVI_GDC_CancelJob	463
11.3.4	CVI_GDC_AddRotationTask	464
11.3.5	CVI_GDC_GenLDCMesh	465
11.3.6	CVI_GDC_AddLDCTask	466
11.4	Data Types	468
11.4.1	GDC_TASK_ATTR_S	468
11.4.2	LDC_ATTR_S	469
11.4.3	FISHEYE_MOUNT_MODE_E	470
12	Proc Debugging Information Explanation	471
12.1	Function Overview	471
12.2	AI	471
12.3	AO	472
12.4	VENC	473
12.5	H265E	475
12.6	H264E	477
12.7	JPEGE	479
12.8	RC	481
12.9	VDEC	483
12.10	LOG	486
12.11	SYS	487
12.12	VB	488
12.13	GDC	492
12.14	REGION	494
12.15	VI	496
12.16	VO	502
12.17	VPSS	505

Revision History

Revision	Date	Description
v0.01	2019/10/12	Draft
v0.02	2020/2/5	Updated Chapter 9 Audio content
v0.03	2020/06/09	Audio sample rate does not support 96k
v0.04	2020/09/09	Updated Chapter 10 GDC content
v0.05	2020/10/22	Update Chapter 9 VQE Algorithm Parameter Definition
V0.2.1.2	2020/11/03	Update Chapter 6 VENC API Update Chapter 7 VDEC API
V0.2.1.3	2020/12/06	Update Chapter 6 VENC-related APIs: RequestIDR, ROIAttr, RefParam, FrameLostStrategy
V0.2.1.4	2021/01/11	<p>Update Chapter 1 LDC_ATTR_S</p> <p>Update Chapter 3 VI_LDC_ATTR_S, CVI_VI_SetChnLDCAttr, CVI_VI_GetChnLDCAttr</p> <p>Update Chapter 5 LDC/ProcAmp related:</p> <p>CVI_VPSS_GetGrpProcAmpCtrl, CVI_VPSS_GetGrpProcAmp,</p> <p>CVI_VPSS_GetGrpProcAmpCtrl, CVI_VPSS_GetGrpProcAmp, CVI_VPSS_SetGrpProcAmp, CVI_VPSS_SetGrpParamfromBin CVI_VPSS_SetGrpProcAmp, CVI_VPSS_SetGrpParamfromBin</p>
V0.2.1.5	2021/01/26	<p>Update related APIs in Chapter 6:</p> <p>CVI_VENC_GetModParam CVI_VENC_ResetChn CVI_VENC_AttachVbPool CVI_VENC_DetachVbPool</p> <p>Chapter 9 Related Audio API Errata</p>
V0.2.1.5	2021/02/02	Chapter 9 Audio API Errata/Additions
V0.2.1.5	2021/03/25	Chapter 9 Audio API Errata

DISCLAIMER



Terms and Conditions

The document and all information contained herein remain the CVITEK Co., Ltd' s ("CVITEK") confidential information, and should not disclose to any third party or use it in any way without CVITEK' s prior written consent.

User shall be liable for any damage and loss caused by unauthority use and disclosure.

CVITEK reserves the right to make changes to information contained in this document at any time and without notice.

All information contained herein is provided in "AS IS" basis, without warranties of any kind, expressed or implied, including without limitation mercantability, non-infringement and fitness for a particular purpose.

In no event shall CVITEK be liable for any third party' s software provided herein, User shall only seek remedy against such third party.

CVITEK especially claims that CVITEK shall have no liable for CVITEK' s work result based on Customer' s specification or published shandard.

Contact Us

Address Building 1, Yard 9, FengHao East Road, Haidian District, Beijing, 100094, China

Building T10, UpperCoast Park, Huizhanwan, Zhancheng Community, Fuhai Street,
Baoan District, Shenzhen, 518100, China

Phone +86-10-57590723 +86-10-57590724

Website <https://www.sophgo.com/>

Forum <https://developer.sophgo.com/forum/index.html>

SYSTEM OVERVIEW

2.1 Function Overview

2.1.1 Objective

The multimedia framework (MMF) provided by CVITEK is used to shorten the time of application development.

This architecture shields the complex underlying design and differences on the chip side, and provides a unified and convenient MMF Programming Interface for applications.

MMF includes the following functions: ISP image preprocessing (including HDR, denoising, edge sharpening, etc.), input image capture and output, image geometric correction, H.265/H.264/JPEG codec, audio capture and output, audio codec, etc.

2.1.2 Definitions and Abbreviations

MMF (Multimedia Framework)
ISP (Image Signal Processor)
VI (Video Input)
VPSS (Video Process Sub-System)
VO (Video Output)
VDEC (Video Decoder)
VENC (Video Encoder)
AI (Audio Input)
AO (Audio Output)
ADEC (Audio Decoder)
AENC (Audio Encoder)
REGION (Regional Management)

2.2 Design Overview

2.2.1 System Architecture

Figure 2-1 shows the system architecture of MMF. From bottom to top, they are:

- Hardware (HW)
Composed of CVITEK SoC and peripheral components, including Flash, DDR, video sensor, etc.
- Driver
Hardware control driver
- Operating System (OS)
OS System Based on Linux
- Input output control (Ioctl)
It is used to control components beyond the scope of SDK, such as MIPI_RX, MIPI_TX.
- System Development Kit (SDK)
Shield the details and differences of the hardware, and provide a unified API for development.
- Application
Applications developed by users based on SDK and ioctl.

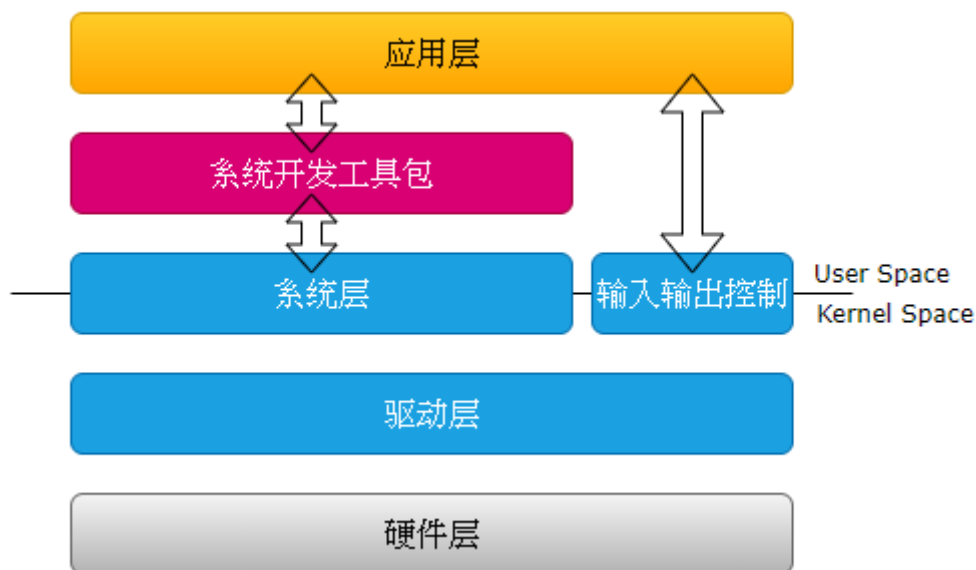


Fig. 2.1: system architecture of MMF

Figure 2-2 shows the main internal processing flow of cvitek media processing platform.

It contains multiple components;

- VI captures the video image, cuts it, optimizes the image, and then transfers the image data to VPSS for processing.
- VDEC decodes the encoded bit stream, and then transfers the image data to VPSS for processing.
- VPSS receives images sent by VI or VDEC, and outputs multiple images with different resolutions for preview, encoding or capturing.
- VO receives the image processed by VPSS and outputs it to the display device according to the set timing.

- REGION can superimpose the user-specified Bitmap as an OSD to image data.

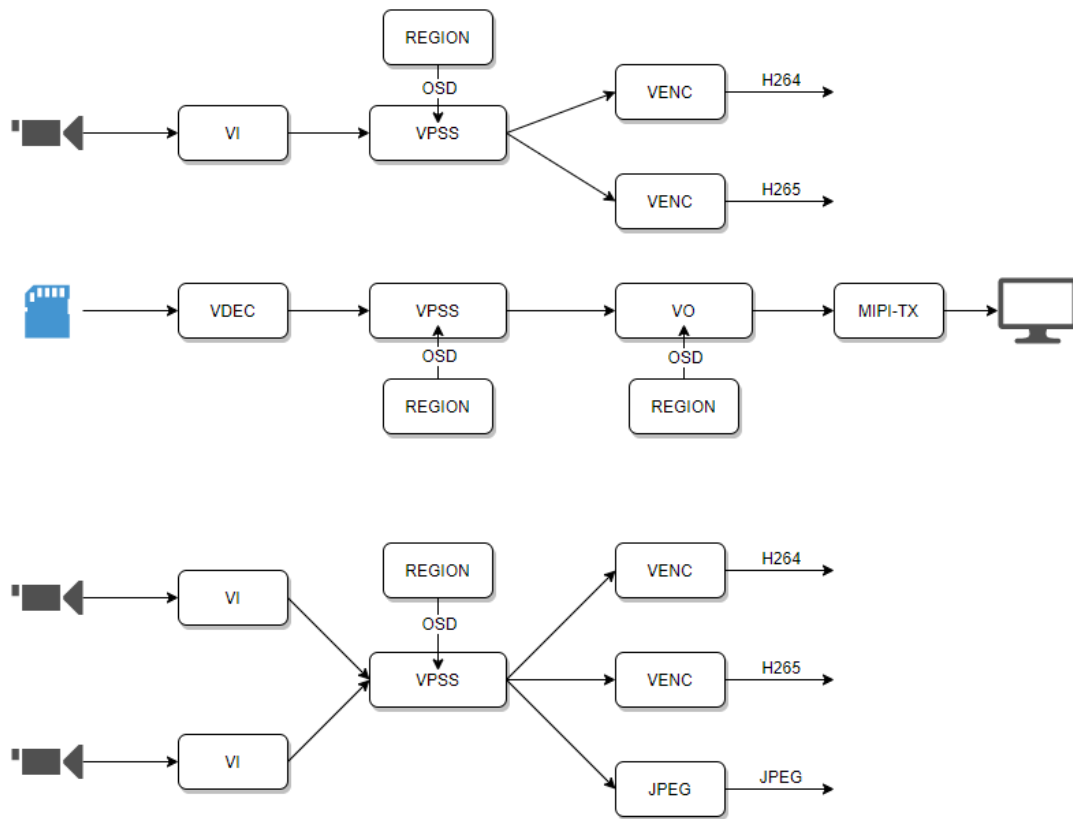


Fig. 2.2: internal processing flow

SYSTEM CONTROL

3.1 Function Overview

3.1.1 Objective

According to the characteristics of each chip, the system control completes the reset and basic initialization of each hardware component, and is responsible for the initialization, control and deinitialization of each function module of MMF system, as well as managing the working state of each function module of MMF system and providing the large physical memory management.

Before the application starts MMF function, it must complete the initialization of MMF system. Similarly, after an application exits the MMF function, it also needs to complete the MMF system deinitialization and release resources.

3.1.2 Definitions and Abbreviations

MMF (Multimedia Framework)
VB (Video Buffer)
VI (Video Input)
VI_CAP (Video Input Capture)
VI_PROC (Video Input Process)
VPSS (Video Process Sub-System)
VO (Video Output)
VDEC (Video Decoder)
VENC (Video Encoder)
AI (Audio Input)
AO (Audio Output)
ADEC (Audio Decoder)
AENC (Audio Encoder)

3.2 Design Overview

3.2.1 Video Memory Block Pool

The video memory buffer pool mainly provides large physical memory management function for each module (VI/VPSS/VO/VDEC/VENC/GDC...), and is responsible for the acquisition, allocation and recovery of memory, so that the physical memory resources can be shared in each media processing module, and unnecessary copying action can be avoided.

Multiple groups of blocks with the same size and continuous physical address form a video buffer pool. The common video chunk pool must be configured before system initialization. Depending on the required functions, the number of public block pools, the size and number of blocks should increase or decrease accordingly.

The number of video buffer pools should be considered as follows:

1. For each additional channel, you need to add two (ping-pong buffer).
2. VO, determined by DisplayBufLen, is an exception to condition 1. The minimum number is 3.
3. If the u32Depth of channel is not 0, the number of u32Depth needs to be increased.
4. Each time an LDC function (lens distortion, rotation, etc.) is added, a memory block pool needs to be added.

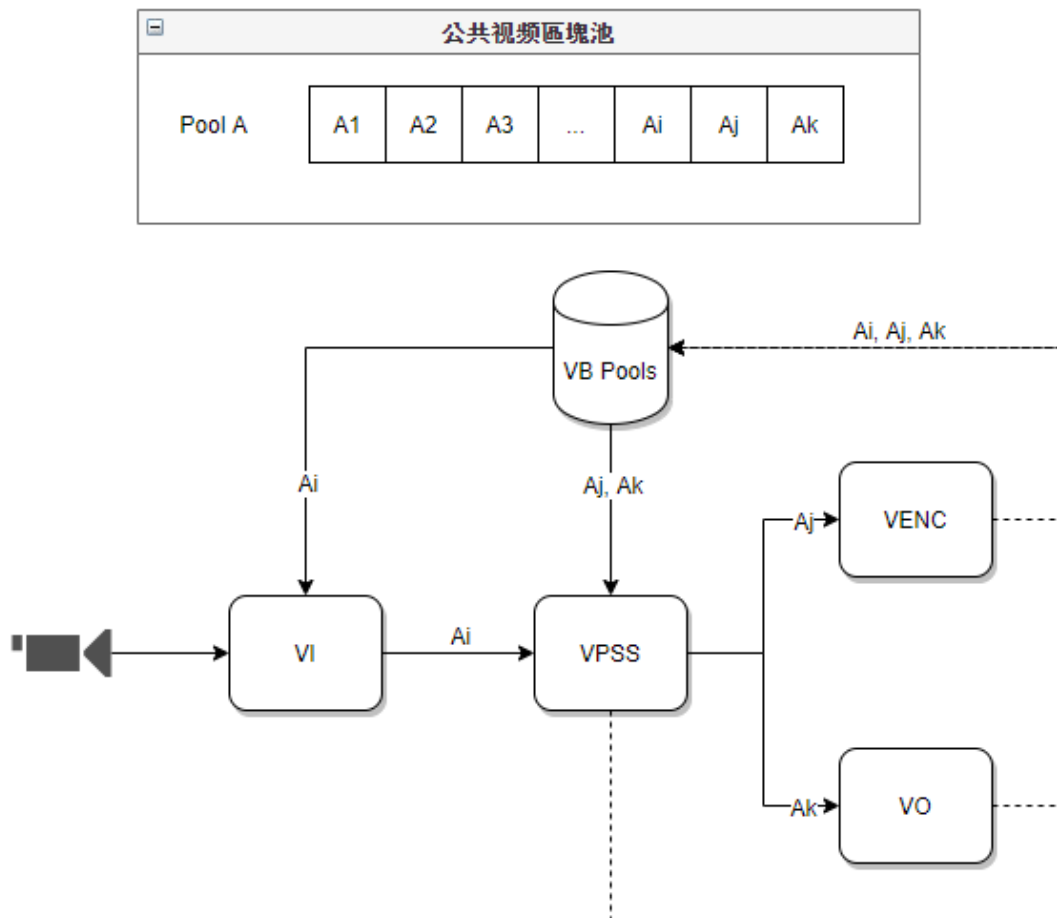
In the case of enough memory space, you can take the maximum space to establish a public video buffer pool; To reduce the memory usage, it is recommended to use multiple public video buffer pools of different sizes.

All video processing channels can obtain video blocks from the common video block pool to save the captured images, as shown in Figure 2-1

1. VI first obtains the video buffer A_i from the public video buffer pool A to store the video data received from the sensor.
2. When VI finishes capturing, the buffer A_i will be sent to the VPSS through the VI, and the VPSS channels 0 and 1 also obtain the video blocks A_j and A_k from the public video block pool A.
3. When VPSS finishes its work, the input buffer A_i will be released back to the public video buffer pool, and A_j will be sent to VENC as the output image buffer, and A_k will be sent to VO as the output image buffer.
4. A_j is released to the public video buffer pool after encoded by VENC, and A_k is released to the public video buffer pool after displayed by VO.

Table 3- 1 Introduction of video buffer pool size calculation interface in `cvi_buffer.h`

Video block pool size calculation interface	Interface introduction
COMMON_GetPicBufferConfig	Data size of each component in the general linear format
COMMON_GetPicBufferSize	The block pool in a general linear format



3.2.2 System Binding

SDK provides system binding interface (CVI_SYS_Bind), that is to establish the relationship between data sources and data receivers by binding them.

After binding, the data generated by the data source will be automatically sent to the receiver.

A data source can be bound to multiple data receivers.

If the data source is not bound, it will eventually automatically return to the video memory buffer pool.

The currently supported binding relationships are shown in table 3-2.

Table 3- 2 Supported binding relationships

Data Source	Data Receiver
VI	VPSS
	VENC
	VO
VPSS	VO
	VENC
	VPSS
VDEC	VPSS
	VENC
	VO
AI	AENC
	AO
ADEC	AO

3.2.3 The Working Mode of VI and VPSS

The working modes of VI and VPSS are divided into online and offline modes. The description of working modes is shown in table 3-3.

Table 3- 3 Description of the working mode of VI and VPSS

Mode	VI_CAP and VI_PROC	VI_PROC and VPSS
Online mode	VI_CAP and VI_PROC transmit data stream online. In this mode, VI_CAP sends data stream to VI_PROC directly, but does not write RAW data to memory.	VI_PROC and VPSS transmit data stream online. In this mode, VI_PROC directly sends the data stream to VPSS instead of writing YUV data to memory.
Offline mode	VI_CAP writes raw data to memory, then VI_PROC reads raw data from memory for post-processing.	VI_PROC writes YUV data to memory, then VPSS reads YUV data from memory for post-processing.

The VI PIPE can be configured to operate in multiple modes, which are described as follows:

- The 0 (long exposure) pipe can have 4 modes
- VI online VPSS offline
- VI online VPSS online
- VI offline VPSS offline
- VI offline VPSS online
- Other PIPE, if marked as “-” in the following table of working modes, it means that YUV cannot be operated independently, only the previous PIPE can operate in HDR.

Table 3- 4 VI PIPE working modes

PIPE ID	0 (long exposure)	1 (short exposure)	2 (long exposure)	3 (short exposure)
Patterndistribution1	Offline	Offline	Offline	Offline
Patterndistribution2	Online	•	Online	•

3.2.4 The Working Mode of VPSS

The VPSS can operate in two modes, namely SINGLE mode and DUAL mode.

Chip	Mode	Device	Number of channels
CV181x	SINGLE	0	4
	DUAL	0	1
		1	3
CV180x	SINGLE	0	3
	DUAL	0	1
		1	2

The default setting is SINGLE.

If it is not SINGLE, it is necessary to specify the operating VPSS device when the VPSS group is created.

3.2.5 Alignment Requirements For The Video Pipeline

When processing data from MEMORY, the modules of each chip have different requirements. Alignment is the reading and writing amount of each line in image processing, which needs to be a multiple of the lines.

For example: YUV420 PLANAR format is 720x480.

Therefore, Y is the data amount of $\text{ALIGN}(720, 32) \times 480 = 736 \times 480$, and U/V is $\text{ALIGN}(360, 32) \times 240 = 384 \times 240$.

It can be modified through APIs such as CVI_VI_SetChnAlign, CVI_VPSS_SetChnAlign, but cannot be less than the hardware limit.

Chip Module	ALIGNMENT
CV181x	VI
CV180x	VPSS
	VO
	64
	64
	64

3.3 API Reference

The MMF system control implements various functions, including system initialization, system binding and unbinding, video block pool initialization, video block pool creation, obtaining the MMF version number, and obtaining the ID of the current chip.

The function module provides the following APIs:

- *CVI_SYS_Init*: Initialize the MMF system.
- *CVI_SYS_Exit* : Deinitialize the MMF system.
- *CVI_SYS_Bind*: Binding the data source to the data receiver.
- *CVI_SYS_UnBind*: Debinding data source to data receiver.
- *CVI_SYS_GetBindbyDest*: Get the bound data source according to the data receiver.
- *CVI_SYS_GetBindbySrc*: Get the bound data receiver according to the data source.
- *CVI_SYS_GetVersion*: Obtain the version number of the current MMF software
- *CVI_SYS_GetChipId*: Obtain the ID of the current chip.
- *CVI_SYS_Mmap*: Memory mapping interface.
- *CVI_SYS_MmapCache*: Memory mapping interface.
- *CVI_SYS_Munmap*: Memory Remap Interface .
- *CVI_SYS_IonAlloc*: User Allocated ION memory.
- *CVI_SYS_IonAlloc_Cached*: The user allocated ION memory will be cacheable.
- *CVI_SYS_IonFlushCache* : Flush the contents of the cache to memory and invalidate the contents of the cache.
- *CVI_SYS_IonInvalidateCache* : Invalidate the contents of the cache.
- *CVI_SYS_IonFree*: User releasing ION memory.
- *CVI_SYS_SetVIVPSSMode*: Set the working mode of VI and VPSS
- *CVI_SYS_GetVIVPSSMode*: Obtain the working mode of VI and VPSS
- *CVI_SYS_SetVPSSMode*: Set the working mode of VPSS
- *CVI_SYS_GetVPSSMode*: Obtain the working mode of VPSS
- *CVI_SYS_GetModName*: Obtain the corresponding string handle for MOD_ID.
- *CVI_VB_SetConfig*: Set MMF video block pool properties.
- *CVI_VB_GetConfig*: Obtaining properties of MMF video block pool.
- *CVI_VB_Init*: Initialize the MMF video block pool.
- *CVI_VB_Exit*: Deinitialize the MMF video block pool.
- *CVI_VB_GetBlock*: Acquire a video block.
- *CVI_VB_ReleaseBlock*: Release a captured video block.
- *CVI_VB_PhysAddr2Handle*: Obtaining the handle of a block by its physical address.
- *CVI_VB_Handle2PhysAddr*: Obtain the physical address of a block.
- *CVI_VB_Handle2PoolId*: Obtain the ID of the block pool where a block resides.
- *CVI_VB_InquireUserCnt*: Query block usage count.
- *CVI_LOG_SetLevelConf*:: Set log level.
- *CVI_LOG_GetLevelConf*: Obtain the log level.

3.3.1 CVI_SYS_Init

【Description】

Initializing the system, including modules such as video Input/Output, video encoding/decoding, video overlay regions, video processing, audio Input/Output, etc.

【Syntax】

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

【Parameter】

None.

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: cvi_sys.h, cvi_comm_sys.h
- Library files: libsys.a

【Note】

- Since the normal operation of MMF system depends on the buffer pool, it is necessary to call CVI_VB_Init to initialize the buffer pool first, and then initialize the MMF system, otherwise the function will run abnormally.
- If the initialization is repeated before it is called for initialization, Success is still returned. In fact, the system has no impact on the running status of the MMF..

【Example】

```
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;
}
```

(continues on next page)

(continued from previous page)

```
s32Ret = CVI_SYS_Init();

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

【Related Topic】

- [CVI_SYS_Exit](#)

3.3.2 CVI_SYS_Exit

【Description】

De-initialize the system. Functions modules including video input and output, video codec, video overlay area, video processing, audio input and output will be destroyed or disabled.

【Syntax】

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

【Parameter】

None.

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: `cvi_sys.h`, `cvi_comm_sys.h`
- Library files: `libsys.a`

【Note】

During de-initialization, if there is a user process blocking MMF, de-initialization will fail. If all calls blocked on MMF return, the de-initialization is successful.

【Example】

Please refer to the example of [CVI_SYS_Init](#).

【Related Topic】

- [CVI_SYS_Init](#)

3.3.3 CVI_SYS_Bind

【Description】

Binding the data source to the data receiver.

【Syntax】

```
CVI_S32 CVI_SYS_Bind(const MMF_CHN_S *pstSrcChn, const MMF_CHN_S *pstDestChn);
```

【Parameter】

Parameter	Description	Input/Output
pstSrcChn	Source channel pointer	Input
pstDestChn	Destination channel pointer	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: cvi_sys.h, cvi_comm_sys.h
- Library files: libsys.a

【Note】

- Please refer to table 3-2 for the binding relationships currently supported by the system.
- Only one data source can be bound to the same data receiver.
- Binding refers to the association between data source and data receiver. After binding, the data generated by the data source will be automatically sent to the receiver.
- VI and VDEC, as data sources, send data to other modules through channels. The user sets the device number to 0, and the SDK does not check the input device number.
- When VPSS serves as the data receiver, the device (GROUP) serves as the receiver to receive data from other modules. The user sets the channel ID to 0
- In other cases, device number and channel number should be specified.

【Example】

None.

【Related Topic】

- [CVI_SYS_UnBind](#)

3.3.4 CVI_SYS_UnBind

【Description】

Data source to data receiver unbinding interface.

【Syntax】

```
CVI_S32 CVI_SYS_UnBind(const MMF_CHN_S *pstSrcChn, const MMF_CHN_S *pstDestChn);
```

【Parameter】

Parameter	Description	Input/Output
pstSrcChn	Source channel pointer	Input
pstDestChn	Destination channel pointer	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: cvi_sys.h, cvi_comm_sys.h
- Library files: libsys.a

【Note】

pstDestChn will directly return Success if the source channel cannot be found. If the bound source channel is found, but the bound source channel does not match the pstSrcChn, a failure is returned.

【Example】

None.

【Related Topic】

- [CVI_SYS_Bind](#)

3.3.5 CVI_SYS_GetBindbyDest

【Description】

Get the bound data source based on the data receiver.

【Syntax】

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

【Parameter】

Parameter	Description	Input/Output
pstDestChn	Destination channel pointer	Input
pstSrcChn	Source channel pointer	Output

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: cvi_sys.h, cvi_comm_sys.h
- Library files: libsys.a

【Note】

None.

【Example】

None.

【Related Topic】

- [CVI_SYS_Bind](#)
- [CVI_SYS_UnBind](#)

3.3.6 CVI_SYS_GetBindbySrc

【Description】

Get the bound data receiver based on the data source.

【Syntax】

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

【Parameter】

Parameter	Description	Input/Output
pstSrcChn	Source channel pointer	Input
pstBindDest	Bound destination channel pointer	Output

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: cvi_sys.h, cvi_comm_sys.h
- Library files: libsys.a

【Note】

None.

【Example】

None.

【Related Topic】

- [CVI_SYS_Bind](#)
- [CVI_SYS_UnBind](#)

3.3.7 CVI_SYS_GetVersion

【Description】

Get the version number of the current MMF software.

【Syntax】

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

【Parameter】

Parameter	Description	Input/Output
pstVersion	Version number of MMF software	Output

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: cvi_sys.h, cvi_comm_sys.h
- Library files: libsys.a

【Note】

None.

【Example】

None.

【Related Topic】

None.

3.3.8 CVI_SYS_GetChipId

【Description】

Get the ID of the current chip.

【Syntax】

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

【Parameter】

Parameter	Description	Input/Output
pu32ChipId	Pointer to the chip ID	Output

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: `cvi_sys.h`, `cvi_comm_sys.h`
- Library files: `libsys.a`

【Note】

None.

【Example】

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

【Related Topic】

None.

3.3.9 CVI_SYS_Mmap

【Description】

Memory mapping interface.

【Syntax】

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

【Parameter】

Parameter	Description	Input/Output
u64PhyAddr	Starting address of the memory unit to be mapped.	Input
u32Size	Number of bytes to be mapped	Input

【Return Value】

Return Value	Description
Non 0	Success.
0	Failure.

【Requirement】

- Header files: `cvi_sys.h`, `cvi_comm_sys.h`
- Library files: `libsys.a`

【Note】

- The entered address must be a valid physical address.
- u32Size cannot be 0

【Example】

None.

【Related Topic】

- [*CVI_SYS_Munmap*](#)

3.3.10 CVI_SYS_MmapCache

【Description】

Memory mapping interface.

【Syntax】

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

【Parameter】

Parameter	Description	Input/Output
u64PhyAddr	Starting address of the memory unit to be mapped.	Input
u32Size	Number of bytes to be mapped	Input

【Return Value】

Return Value	Description
Non 0	Success.
0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: cvi_sys.h, cvi_comm_sys.h
- Library files: libsys.a

【Note】

- The entered address must be a valid physical address.
- u32Size cannot be 0
- If the virtual address obtained through this API is shared with HW DMA, user needs to call invalidate before the CPU accesses it; If you want HW DMA to read after the CPU changes, you need to invalidate.

【Example】

None.

【Related Topic】

- [CVI_SYS_Munmap](#)
- [CVI_SYS_IonFlushCache](#)
- [CVI_SYS_IonInvalidateCache](#)

3.3.11 CVI_SYS_Munmap

【Description】

Memory Remap Interface

【Syntax】

```
CVI_S32 CVI_SYS_Munmap(CVI_VOID *pVirAddr, CVI_U32 u32Size);
```

【Parameter】

Parameter	Description	Input/Output
pVirAddr	The address returned after mmap	Input
u32Size	Number of bytes to be mapped	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: cvi_sys.h, cvi_comm_sys.h
- Library files: libsys.a

【Note】

None.

【Example】

None.

【Related Topic】

- [CVI_SYS_Mmap](#)

3.3.12 CVI_SYS_IonAlloc

【Description】

The user allocates ION memory.

【Syntax】

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

【Parameter】

Parameter	Description	Input/Output
pu64PhyAddr	Physical address pointer.	Output
ppVirAddr	Pointer to the virtual address. If it is NULL, no mapping will be done.	Output
u32Len	Size.	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: cvi_sys.h, cvi_comm_sys.h
- Library files: libsys.a

【Note】

None.

【Example】

None.

【Related Topic】

- [CVI_SYS_IonFree](#)

3.3.13 CVI_SYS_IonAlloc_Cached

【Description】

The user allocated ION memory will be cacheable.

【Syntax】

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

【Parameter】

Parameter	Description	Input/Output
pu64PhyAddr	Physical address pointer.	Output
ppVirAddr	Pointer to the virtual address pointer. If it is NULL, no mapping will be done.	Output
u32Len	Size.	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: [cvi_sys.h](#), [cvi_comm_sys.h](#)
- Library files: [libsys.a](#)

【Note】

- If the virtual address obtained through this API is shared with HW DMA, user needs to call invalidate before the CPU accesses it; If you want HW DMA to read after the CPU changes, you need to invalidate.

【Example】

None.

【Related Topic】

- [CVI_SYS_IonFlushCache](#)
- [CVI_SYS_IonInvalidateCache](#)
- [CVI_SYS_IonFree](#)

3.3.14 CVI_SYS_IonFlushCache

【Description】

Flush the content in the cache to memory and invalidate the content in the cache.

【Syntax】

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

【Parameter】

Parameter	Description	Input/Output
pu64PhyAddr	Physical address pointer.	Input
pVirAddr	Virtual address pointer.	Input
u32Len	Size.	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: cvi_sys.h, cvi_comm_sys.h
- Library files: libsys.a

【Note】

This interface should be used with *CVI_SYS_IonAlloc_Cached* interface.

【Example】

None.

【Related Topic】

- *CVI_SYS_IonAlloc_Cached*
- *CVI_SYS_IonInvalidateCache*

3.3.15 CVI_SYS_IonInvalidateCache

【Description】

To invalidate the contents in the cache.

【Syntax】

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

【Parameter】

Parameter	Description	Input/Output
pu64PhyAddr	Physical address pointer.	Input
pVirAddr	Virtual address pointer.	Input
u32Len	Size.	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: cvi_sys.h, cvi_comm_sys.h
- Library files: libsys.a

【Note】

This interface should be used with *CVI_SYS_IonAlloc_Cached* interface.

【Example】

None.

【Related Topic】

- *CVI_SYS_IonAlloc_Cached*
- *CVI_SYS_IonFlushCache*

3.3.16 CVI_SYS_IonFree

【Description】

The user releases the ION memory.

【Syntax】

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

【Parameter】

Parameter	Description	Input/Output
u64PhyAddr	Physical address pointer.	Input
pVirAddr	Virtual address pointer.	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: cvi_sys.h, cvi_comm_sys.h
- Library files: libsys.a

【Note】

None.

【Example】

None.

【Related Topic】

- *CVI_SYS_IonAlloc*
- *CVI_SYS_IonAlloc_Cached*

3.3.17 CVI_SYS_SetVIVPSSMode

【Description】

Configure VI & VPSS Operation Mode

【Syntax】

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

【Parameter】

Parameter	Description	Input/Output
pstVIVPSSMode	Working mode of VI and VPSS	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: cvi_sys.h, cvi_comm_sys.h
- Library files: libsys.a

【Note】

Must be configured after CVI_SYS_Init and before all VI pipes and VPSS groups are created.

【Example】

None.

【Related Topic】

- [CVI_SYS_GetVIVPSSMode](#)

3.3.18 CVI_SYS_GetVIVPSSMode

【Description】

Get the working mode of VI and VPSS.

【Syntax】

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

【Parameter】

Parameter	Description	Input/Output
pstVIVPSSMode	Working mode of VI and VPSS	Output

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: `cvi_sys.h`, `cvi_comm_sys.h`
- Library files: `libsys.a`

【Note】

None.

【Example】

None.

【Related Topic】

- [*CVI_SYS_SetVIVPSSMode*](#)

3.3.19 CVI_SYS_SetVPSSMode

【Description】

Set the working mode of VPSS.

【Syntax】

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

【Parameter】

Parameter	Description	Input/Output
enVPSSMode	VPSS working mode	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: `cvi_sys.h`, `cvi_comm_sys.h`
- Library files: `libsys.a`

【Note】

Must be set after `CVI_SYS_Init` and before all VPSS groups are created.

The default setting is `VPSS_MODE_SINGLE`.

【Example】

None.

【Related Topic】

- [*CVI_SYS_GetVPSSMode*](#)

3.3.20 CVI_SYS_GetVPSSMode

【Description】

Get the working mode of VPSS.

【Syntax】

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

【Parameter】

None.

【Return Value】

Return Value	Description
VPSS_MODE_E	Current working mode of VPSS

【Requirement】

- Header files: cvi_sys.h, cvi_comm_sys.h
- Library files: libsys.a

【Note】

None.

【Example】

None.

【Related Topic】

- *CVI_SYS_SetVPSSMode*

3.3.21 CVI_SYS_GetModName

【Description】

Get the corresponding string handle for MOD_ID.

【Syntax】

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

【Parameter】

Parameter	Description	Input/Output
id	MOD_ID_E	Input

【Return Value】

Return Value	Description
const CVI_CHAR *	String handle of MOD_ID
NULL	Failure.

【Requirement】

- Header files: cvi_sys.h, cvi_comm_sys.h
- Library files: libsys.a

【Note】

None.

【Example】

None.

【Related Topic】

None.

3.3.22 CVI_VB_SetConfig

【Description】

Set properties of MMF video block pool.

【Syntax】

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

【Parameter】

Parameter	Description	Input/Output
pstVbConfig	Pointer to video block pool attributes.	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: `cvi_sys.h`, `cvi_comm_sys.h`
- Library files: `libsys.a`

【Note】

- You can set block pool properties only when the system is in the uninitialized state. Otherwise a failure will be returned.
- Video buffer configuration varies according to different application scenarios. For details about the configuration rules, please see 2.2.1 Video Block Pool.
- The size of each block in the public block pool varies according to the current image pixel format and whether the image is compressed or not. Please refer to `VB_CONFIG_S` structure for specific allocation size.

【Example】

Refer to example of *CVI_SYS_Init*.

【Related Topic】

- *CVI_VB_GetConfig*

3.3.23 CVI_VB_GetConfig

【Description】

Get properties of MMF video block pool.

【Syntax】

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

【Parameter】

Parameter	Description	Input/Output
pstVbConfig	Pointer to the video block pool attributes.	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: cvi_sys.h, cvi_comm_sys.h
- Libray files: libsys.a

【Note】

You must first call CVI_VB_SetConfig to set the MMF video block pool properties.

【Example】

None.

【Related Topic】

- [CVI_VB_SetConfig](#)

3.3.24 CVI_VB_Init

【Description】

Initialize the MMF video block pool.

【Syntax】

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

【Parameter】

None.

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: cvi_sys.h, cvi_comm_sys.h
- Library files: libsys.a

【Note】

- You must call `CVI_VB_SetConfig` to configure the block pool properties before initialization, otherwise the initialization will fail.
- If the initialization is repeated, Success is still returned, but the system has virtually no effect on the running state of the MMF.

【Example】

Refer to example of [CVI_SYS_Init](#).

【Related Topic】

- [CVI_VB_Exit](#)

3.3.25 CVI_VB_Exit

【Description】

Deinitialize MMF video block pool.

【Syntax】

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

【Parameter】

None.

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: `cvi_sys.h`, `cvi_comm_sys.h`
- Library files: `libsys.a`

【Note】

- Must call `CVI_SYS_Exit` to deinitailize MMF system before de-initializing block pool. Otherwise the system will return failure.
- The initialization can be repeated without returning a failure

【Example】

Refer to the example of [CVI_SYS_Init](#)

【Related Topic】

- [CVI_VB_Init](#)

3.3.26 CVI_VB_GetBlock

【Description】

Get a video block.

【Syntax】

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

【Parameter】

Parameter	Description	Input/Output
Pool	Block pool ID number. Value range: [0, VB_MAX_POOLS)。	Input
u32BlkSize	Block size. Value range: full range of data types, measured in bytes.	Input

【Return Value】

Return Value	Description
Non VB_INVALID_HANDLE	Valid block handle.
VB_INVALID_HANDLE	Obtainment failed.

【Requirement】

- Header files: cvi_sys.h, cvi_comm_sys.h
- Library files: libsys.a

【Note】

- After creating a block pool, users can call this interface to obtain blocks from the block pool; the second parameter u32BlkSize must be less than or equal to the block size specified when creating the block pool.
- When the first parameter Pool is set to invalid ID number VB_INVALID_POOLID, a block of specified size will be obtained from any common block pool;
Otherwise, a block of specified size will be obtained from the specified pool.
If the specified size does not match, no block will be obtained.
- The common block pool is mainly used to store the captured images of VPU(VI/VPSS/VO/GDC).
Because it is shared by multiple modules, improper operation of the common block pool (such as occupying too many blocks) will affect the normal operation of the whole MMF system.

【Example】

None.

【Related Topic】

- [CVI_VB_ReleaseBlock](#)

3.3.27 CVI_VB_ReleaseBlock

【Description】

Release a captured video block.

【Syntax】

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

【Parameter】

Parameter	Description	Input/Output
Block	Block handle	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: cvi_sys.h, cvi_comm_sys.h
- Library files: libsys.a

【Note】

This interface should be called to release blocks after the obtained blocks are used.

【Example】

None.

【Related Topic】

- [CVI_VB_GetBlock](#)

3.3.28 CVI_VB_CreatePool

【Description】

Get a video block.

【Syntax】

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

【Parameter】

Parameter	Description	Input/Output
pstVbPoolCfg	Block pool configuration attribute parameter pointer.	input

【Return Value】

Return Value	Description
Non VB_INVALID_HANDLE	Valid block pool ID.
VB_INVALID_HANDLE	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: `cvi_sys.h`, `cvi_comm_sys.h`
- Library files: `libsys.a`

【Note】

None.

【Example】

None.

【Related Topic】

- [*CVI_VB_DestroyPool*](#)

3.3.29 CVI_VB_DestroyPool

【Description】

Release a captured video block.

【Syntax】

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

【Parameter】

Parameter	Description	Input/Output
Pool	Block pool ID number. Value range: [0, VB_MAX_POOLS)。	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: `cvi_sys.h`, `cvi_comm_sys.h`
- Library files: `libsys.a`

【Note】

- To destroy a non-existent block pool, return `CVI_ERR_VB_ILLEGAL_PARAM`.
- When the MMF block pool is deinitialized, all block pools are destroyed, including user-mode ones.
- Before exiting the VB pool, ensure that no VB in the VB pool is occupied. Otherwise, you cannot exit.
- [0, VB_MAX_POOLS) ID of block pools includes ID of public block pools, module public block pools, and module private block pools. Ensure that Pool is the ID of the block pool created by `CVI_VB_CreatePool`, otherwise a failure will be returned.
- If the virtual address of the current block pool is mapped using the `CVI_VB_MmapPool` interface, the mapping must be removed using the `CVI_VB_MunmapPool` interface before the block pool is destroyed.

【Example】

None.

【Related Topic】

- *CVI_VB_CreatePool*

3.3.30 CVI_VB_PhysAddr2Handle

【Description】

Get a handle from an already acquired block physical address

【Syntax】

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

【Parameter】

Parameter	Description	Input/Output
u64PhyAddr	Block physical address	Input

【Return Value】

Return Value	Description
Non VB_INVALID_HANDLE	Valid block handle.
VB_INVALID_HANDLE	Obtainment failed.

【Requirement】

- Header files: cvi_sys.h, cvi_comm_sys.h
- Library files: libsys.a

【Note】

The physical address should be the address of the valid block obtained from the MMF video block pool.

【Example】

None.

【Related Topic】

None.

3.3.31 CVI_VB_Handle2PhysAddr

【Description】

Get the physical address from an already acquired block handle

【Syntax】

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

【Parameter】

Parameter	Description	Input/Output
Block	Block handle	Input

【Return Value】

Return Value	Description
Non 0	Valid physical address.
0	Invalid return value, illegal block handle.

【Requirement】

- Header files: cvi_sys.h, cvi_comm_sys.h
- Library files: libsys.a

【Note】

Blocks should be valid blocks obtained from the MMF video block pool.

【Example】

None.

【Related Topic】

None.

3.3.32 CVI_VB_Handle2PoolId

【Description】

Get the video block pool ID from an already acquired block handle.

【Syntax】

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

【Parameter】

Parameter	Description	Input/Output
Block	Block handle	Input

【Return Value】

Return Value	Description
Non VB_INVALID_POOLID	Valid video block pool ID.
VB_INVALID_POOLID	Invalid return value, illegal block handle.

【Requirement】

- Header files: cvi_sys.h, cvi_comm_sys.h
- Library files: libsys.a

【Note】

Blocks should be valid blocks obtained from the MMF video block pool.

【Example】

None.

【Related Topic】

None.

3.3.33 CVI_VB_InquireUserCnt

【Description】

Get the block usage count from an already acquired block handle.

【Syntax】

```
CVI_S32 CVI_VB_InquireUserCnt(VB_BLK Block);
```

【Parameter】

Parameter	Description	Input/Output
Block	Block handle	Input

【Return Value】

Return Value	Description
Nonnegative number	Use count.
Negative number	Invalid return value, illegal block handle.

【Requirement】

- Header files: cvi_sys.h, cvi_comm_sys.h
- Library files: libsys.a

【Note】

None.

【Example】

None.

【Related Topic】

None.

3.3.34 CVI_VB_MmapPool

【Description】

Mapping a user-space virtual address for a video block pool.

【Syntax】

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

【Parameter】

Parameter	Description	Input/Output
Pool	Block pool ID number. Value range: [0, VB_MAX_POOLS)。	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: `cvi_sys.h`, `cvi_comm_sys.h`, `cvi_debug.h`
- Library files: `libsys.a`

【Note】

- You must enter a valid video block pool ID.
- Repeated mappings are considered successful.

【Example】

None.

【Related Topic】

None.

3.3.35 CVI_VB_MunmapPool

【Description】

Unmapping a user-space mapping for a video block pool.

【Syntax】

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

【Parameter】

Parameter	Description	Input/Output
Pool	Block pool ID number. Value range: [0, VB_MAX_POOLS)。	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: `cvi_sys.h`, `cvi_comm_sys.h`, `cvi_debug.h`
- Library files: `libsys.a`

【Note】

- You must enter a valid video block pool ID.
- The video block pool must be mapped. If it is not mapped, `CVI_ERR_VB_NOTREADY` is returned
- The virtual address must be released before the block pool is destroyed.

【Example】

None.

【Related Topic】

None.

3.3.36 CVI_VB_GetBlockVirAddr

【Description】

Obtaining the user-space virtual address of a block in a video block pool.

【Syntax】

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

【Parameter】

Parameter	Description	Input/Output
Pool	Block pool ID number. Value range: [0, VB_MAX_POOLS)。	Input
Block	Block handle	Input
ppVirAddr	User-space virtual address	Output

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: cvi_sys.h, cvi_comm_sys.h, cvi_debug.h
- Library files: libsys.a

【Note】

- You must enter a valid video block pool ID and a valid block handle.
- The video block pool must be mapped. If it is not mapped, CVI_ERR_VB_NOTREADY is returned
- If the physical address is not in the current VB pool, CVI_ERR_VB_ILLEGAL_PARAM is returned.

【Example】

None.

【Related Topic】

None.

3.3.37 CVI_LOG_SetLevelConf

【Description】

Set the log level.

【Syntax】

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

【Parameter】

Parameter	Description	Input/Output
pstConf	Log level information structure	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: cvi_sys.h, cvi_comm_sys.h, cvi_debug.h
- Library files: libsys.a

【Note】

None.

【Example】

None.

【Related Topic】

- [CVI_LOG_SetLevelConf](#)

3.3.38 CVI_LOG_GetLevelConf

【Description】

Get the log level.

【Syntax】

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

【Parameter】

Parameter	Description	Input/Output
pstConf->enModId	Need to get the module ID of log level	Input
pstConf->s32Level	Get the log level	Output

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: cvi_sys.h, cvi_comm_sys.h, cvi_debug
- Library files: libsys.a

【Note】

None.

【Example】

None.

【Related Topic】

- [CVI_LOG_SetLevelConf](#)

3.4 Data Types

3.4.1 Base Type

```

/*-----
 * 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;
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_TRUE            1
#define CVI_FALSE           0

```

3.4.2 MOD_ID_E

【Description】

Defining module ID enumeration.

【Syntax】

```

#define FOREACH_MOD(MOD) {\
    MOD(CMPI)    \
    MOD(VB)      \
    MOD(SYS)     \
}

```

(continues on next page)

(continued from previous page)

```

MOD(RGN)      \
MOD(CHNL)     \
MOD(VDEC)     \
MOD(VPSS)     \
MOD(VENC)     \
MOD(H264E)    \
MOD(JPEGE)    \
MOD(MPEG4E)   \
MOD(H265E)    \
MOD(JPEGD)    \
MOD(VO)       \
MOD(VI)       \
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;

```

【Note】

None.

【Related Data Type and Interface】

None.

3.4.3 VB_SOURCE_E

【Description】

Define VB source selection.

【Syntax】

```

typedef enum _VB_SOURCE_E {
    VB_SOURCE_COMMON = 0,
    VB_SOURCE_MODULE = 1,
    VB_SOURCE_PRIVATE = 2,
    VB_SOURCE_USER = 3,
}

```

(continues on next page)

(continued from previous page)

```
VB_SOURCE_BUTT  
} VB_SOURCE_E;
```

【Member】

Member	Description
VB_SOURCE_COMMON	public VB
VB_SOURCE_MODULE	module VB
VB_SOURCE_PRIVATE	private VB
VB_SOURCE_USER	user VB

【Note】

None.

【Related Data Type and Interface】

None.

3.4.4 ROTATION_E

【Description】

Defining rotation angle enumeration.

【Syntax】

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

【Member】

Member	Description
ROTATION_0	No rotation.
ROTATION_90	Rotate 90 degrees.
ROTATION_180	Rotate 180 degrees.
ROTATION_270	Rotate 270 degrees.

【Note】

None.

【Related Data Type and Interface】

None.

3.4.5 POINT_S

【Description】

Define coordinate structure.

【Syntax】

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

【Member】

Member	Description
s32X	Abscissa
s32Y	Ordinate

【Note】

None.

【Related Data Type and Interface】

None.

3.4.6 SIZE_S

【Description】

Define size structure.

【Syntax】

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

【Member】

Member	Description
u32Width	Width
u32Height	Height

【Note】

None.

【Related Data Type and Interface】

- 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

【Description】

Define the width, height, and position of the rectangle.

【Syntax】

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

【Member】

Member	Description
s32X	Abscissa
s32Y	Ordinate
u32Width	Width
u32Height	Height

【Note】

None.

【Related Data Type and Interface】

- [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

【Description】

Define lens distortion correction structure.

【Syntax】

```
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;
```

【Member】

Member	Description
bAspect	Whether the distortion correction is horizontal and vertical in the same proportion.
s32XRatio	The size of horizontal field of view is valid when bAspect=0. [0, 100]
s32YRatio	The size of veritcal field of view is valid when bAspect=0. [0, 100]
s32XYRatio	The overall field of view is valid when bAspect = 1. [0, 100]
s32CenterXOffset	The horizontal offset of the distortion center from the image center. [-511, 511]
s32CenterYOffset	The vertical offset of the distortion center from the image center. [-511, 511]
s32DistortionRatio	The degree of distortion. [-300, 500]

【Note】

None.

【Related Data Type and Interface】

None.

3.4.9 MMF_CHN_S

【Description】

Define module channel structure.

【Syntax】

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

【Member】

Member	Description
enModId	Module ID
s32DevId	Device ID In some modules, it may also be a group ID.
s32ChnId	Channel ID

【Note】

None.

【Related Data Type and Interface】

- CVI_SYS_Bind
- CVI_SYS_UnBind
- CVI_SYS_GetBindByDest
- CVI_SYS_GetBindBySrc

3.4.10 MMF_BIND_DEST_S

【Description】

Define MMF system binding structure.

【Syntax】

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

【Member】

Member	Description
u32Num	Number of bound destinations.
astMmfChn	Channel structure array for binding purposes.

【Note】

None.

【Related Data Type and Interface】

- CVI_SYS_GetBindBySrc

3.4.11 MMF_VERSION_S

【Description】

Define MMF version structure.

【Syntax】

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

【Member】

Member	Description
version	Version description string.

【Note】

None.

【Related Data Type and Interface】

- CVI_SYS_GetVersion

3.4.12 VB_CONFIG_S

【Description】

Define MMF system video block pool structure.

【Syntax】

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

【Member】

Member	Description
u32MaxPoolCnt	Number of common video block pools.
astCommPool	Common video block pool properties.

【Note】

None.

【Related Data Type and Interface】

- CVI_VB_SetConfig
- CVI_VB_GetConfig

3.4.13 VB_POOL_CONFIG_S

【Description】

Define MMF video block pool structure.

【Syntax】

```
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;
```

【Member】

Member	Description
u32BlkSize	Video block size
u32BlkCnt	The number of blocks in the video block pool
enRemapMode	Memory-map mode of block
acName	Block name

【Note】

- u32BlkSize should be calculated according to the required image size, format and other information. If it is too large, it will cause unnecessary waste of memory space; if it is too small, each mod cannot get video blocks to use.
- The video block pool is obtained from the free memory. If the size of the video block pool exceeds the free memory size, the video block pool fails to be created.

【Related Data Type and Interface】

- *VB_CONFIG_S*

3.4.14 VI_VPSS_MODE_E

【Description】

Define the working mode between VI and VPSS.

【Syntax】

```
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;
```

【Member】

Member	Description
VI_OFFLINE_VPSS_OFFLINE	VI_PROC/VI_CAP offline, VI_CAP/VPSS offline
VI_OFFLINE_VPSS_ONLINE	VI_PROC/VI_CAP offline, VI_CAP/VPSS online
VI_ONLINE_VPSS_OFFLINE	VI_PROC/VI_CAP online, VI_CAP/VPSS offline
VI_ONLINE_VPSS_ONLINE	VI_PROC/VI_CAP online, VI_CAP/VPSS online

【Note】

- When VPSS_ONLINE, VPSS can't process by time sharing and can only bind specific VI PIPE.

【Related Data Type and Interface】

None.

3.4.15 VPSS_MODE_E

【Description】

Define the working mode of VPSS enumeration.

【Syntax】

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

【Member】

Member	Description
VPSS_MODE_SINGLE	VPSS works as a single hardware entity.
VPSS_MODE_DUAL	VPSS works in two hardware entities.
VPSS_MODE_RGNEX	VPSS works in two hardware entities. VPSS device 0 is used for RGN overlay, so only VPSS device 1 can operate in online mode.

【Note】

- CV181x/CV180x does not support VPSS MODE RGNEX mode

【Related Data Type and Interface】

None.

3.4.16 ASPECT_RATIO_E**【Description】**

Define rotation angle enumeration.

【Syntax】

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

【Member】

Member	Description
ASPECT_RATIO_NONE	No motion, full screen.
ASPECT_RATIO_AUTO	Keeping video aspect ratio and automatically calculating video area.
ASPECT_RATIO_MANUAL	Manually determine the video area.

【Note】

None.

【Related Data Type and Interface】

None.

3.4.17 ASPECT_RATIO_S**【Description】**

Define screen ratio structure.

【Syntax】

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

【Member】

Member	Description
enMode	The ratio of pictures is listed.
bEnableBgColor	Whether to cover the outside of the picture with a background color.
u32BgColor	The background color of the screen scale is RGB888. Bit [7:0] is B, bit [15:8] is G, and bit [23:16] is R.
stVideoRect	The area of the video. It works only when the enMode is ASPECT_RATIO_MANUAL.

【Note】

None.

【Related Data Type and Interface】

- *ASPECT_RATIO_E*

3.4.18 PIXEL_FORMAT_E**【Description】**

Define pixel format enumeration.

【Syntax】

```
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,
    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,
```

(continues on next page)

(continued from previous page)

```

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_MAX
} PIXEL_FORMAT_E;

```

【Member】

None.

【Note】

None.

【Related Data Type and Interface】

None.

3.4.19 VIDEO_FRAME_S

【Description】

Define video frame information structure.

【Syntax】

```

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 s160offsetTop;
    CVI_S16 s160offsetBottom;
    CVI_S16 s160offsetLeft;
    CVI_S16 s160offsetRight;

    CVI_U32 u32TimeRef;
    CVI_U64 u64PTS;

    void *pPrivateData;
    CVI_U32 u32FrameFlag;
} VIDEO_FRAME_S;

```

【Member】

Member	Description
u32Width	image width
u32Height	image height
enPixelFormat	image pixel format
enBayerFormat	image raw format
enVideoFormat	image format
enCompressMode	image compression format
enDynamicRange	dynamic range
enColorGamut	gamut range
u32Stride	Image row stride.
u64PhyAddr	physical address
pu8VirAddr	virtual address
u32Length	image space size (Bytes)
s16OffsetTop	image top crop width
s16OffsetBottom	image bottom crop width
s16OffsetLeft	image left crop width
s16OffsetRight	image right crop width
u32TimeRef	Image frame sequence number
u64PTS	Image timestamp
pPrivateData	private data
u32FrameFlag	the mark of the current frame

【Note】

None.

【Related Data Type and Interface】

None.

3.4.20 VIDEO_FRAME_INFO_S

【Description】

Define video frame information structure.

【Syntax】

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

【Member】

Member	Description
stVFrame	Video frame
u32PoolId	Cache pool ID

【Note】

None.

【Related Data Type and Interface】

- [VIDEO_FRAME_S](#)

3.4.21 BITMAP_S

【Description】

Define BITMAP information.

【Syntax】

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

【Member】

Member	Description
enPixelFormat	Bitmap pixel format
u32Width	Bitmap width
u32Height	Bitmap height
pData	Bitmap data address

【Note】

None.

【Related Data Type and Interface】

None.

3.5 Error Codes

System control error code

Error Code	Definition	Description
0xC0028003	CVI_ERR_SYS_ILLEGAL_PARAM	Invalid parameter setting
0xC0028006	CVI_ERR_SYS_NULL_PTR	null pointer
0xC0028008	CVI_ERR_SYS_NOT_SUPPORTED	Features not supported
0xC0028009	CVI_ERR_SYS_NOT_PERM	operation not allowed
0xC002800C	CVI_ERR_SYS_NOMEM	memory allocation failure, such as insufficient system memory
0xC002800D	CVI_ERR_SYS_REMAPPING	memory mapping failed
0xC0028010	CVI_ERR_SYS_NOTREADY	The system control property is not configured
0xC0028012	CVI_ERR_SYS_BUSY	The system is busy

Video block pool error code

Error Code	Definition	Description
0xC0018003	CVI_ERR_VB_ILLEGAL_PARAM	Invalid parameter setting
0xC0018005	CVI_ERR_VB_UNEXIST	video block does not exist
0xC0018006	CVI_ERR_VB_NULL_PTR	null pointer
0xC0018009	CVI_ERR_VB_NOT_PERM	operation not allowed
0xC001800C	CVI_ERR_VB_NOMEM	memory allocation failure, such as insufficient system memory
0xC001800D	CVI_ERR_VB_NOBUF	memory buff failed
0xC0018010	CVI_ERR_VB_NOTREADY	The system control property is not configured
0xC0018012	CVI_ERR_VB_BUSY	The system is busy
0xC0018013	CVI_ERR_VB_SIZE_NOT_ENOUGH	video block size is not enough
0xC0018040	CVI_ERR_VB_2MPOOLS	too many cache pools created

VIDEO INPUT

The function of video input (VI) module: receiving video data through MIPI_RX (including MIPI interface, LVDS interface and HISPI interface), BT.1120,BT.656,BT.601 and other interfaces to achieve video data acquisition. VI stores the received data into the designated memory area and processes the received original video image data through ISP.

4.1 Function Overview

4.1.1 Objective

- The video input device supports several kinds of timing input and is responsible for parsing the timing.
- The video input PIPE is bound to the backend of the device and is responsible for further processing the data after it is parsed by the device.
- The video input is responsible for outputting the finally processed data to DDR.

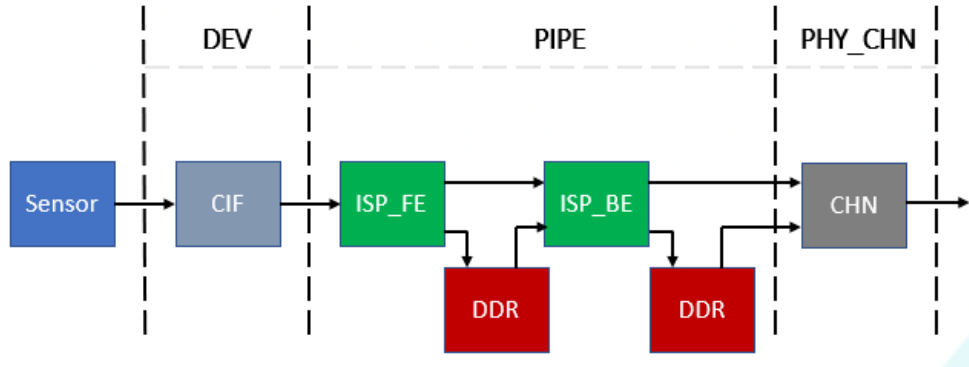
4.1.2 Definitions and Abbreviations

- CIF: Interface of connecting sensor, i.e., MIPI_RX, BT, etc.
- ISP_FE: Used to output AE, AWB, AF and other statistical data to DDR.
- ISP_BE: Adjust the quality and color of image input.
- CHN: Image correction, for image crop, rotation or distortion correction.
- PHY_PIPE: The physical PIPE is bound to the backend of the device and is responsible for further processing the data after it is parsed by the device.
- PHY_CHN: The physical video channel is responsible for outputting the finally processed data to DDR. Before actually outputting the data to DDR, it can perform functions such as cropping.

4.2 Design Overview

4.2.1 System Architecture

- VI is divided into four levels: input device (DEV), image capture device (ISP_FE), image processing (ISP_BE), and image correction (CHN).
- Sensor transmits images through CIF interface and delivers images to ISP_FE, which extracts statistics such as AE, AWB, AF, etc., and stores raw files in DDR. Then, the ISP_BE converts the color space and adjusts the image quality of the raw files in either online or offline mode before



storing the YUV in DDR. Finally, the backend CHN can perform processing such as crop, rotation on the YUV data.

- The parameter configuration of current chip devices is shown in [Table 4.1](#).

Table 4.1: Capability of each chip series

Chip	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

Attention: CV180x does not support HDR.

4.2.2 Video Input PIPE

- The PIPE of VI contains the relevant processing functions of ISP, mainly for pipelining image data processing and outputting YUV image format to channels. Please refer to the “VI and VPSS” work mode description in the “System Control” chapter for the working mode of PIPE.

4.2.3 Video Physical Channel

- Each physical channel has a clipping function and supports typical resolutions, such as 2880x1620@30fps, 2592x1944@30fps, 2560x1440@30fps, 1080p@30fps, etc.

4.2.4 Binding Relationship

- The binding relationship between Dev and CIF is fixed and cannot be modified dynamically.
- The constraint relationship between Dev and timing input interface is shown in [Table 4.2](#).

Table 4.2: Binding Relationship between DEV and MIPI/SLVS interface

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

4.3 API Reference

The Video Input (VI) module enables the activation of the video input device, creation of video input PIPE, configuration of video input channels, binding of the Dev to MIPI devices, and binding of the PIPE to Dev, etc.

The function module provides the following APIs.

- *CVI_VI_SetDevAttr*: Set VI device properties.
- *CVI_VI_GetDevAttr*: Get VI device properties.
- *CVI_VI_SetDevAttrEx*: Set VI device advanced properties.
- *CVI_VI_GetDevAttrEx*: Get VI device advanced properties.
- *CVI_VI_EnableDev*: Enable the VI device.
- *CVI_VI_DisableDev*: Disable VI devices.
- *CVI_VI_SetDevBindPipe*: Set the binding relationship between VI device and physical PIPE.
- *CVI_VI_GetDevBindPipe*: Obtain the physical PIPE bound to the VI device
- *CVI_VI_SetDevTimingAttr*: Set the self-generating timing property.
- *CVI_VI_GetDevTimingAttr*: Get the self-generating timing property.
- *CVI_VI_CreatePipe*: Create a VI PIPE.
- *CVI_VI_DestroyPipe*: Destroy a VI PIPE.
- *CVI_VI_SetPipeAttr*: Set the properties of VI PIPE.
- *CVI_VI_GetPipeAttr*: Get the properties of VI PIPE.
- *CVI_VI_StartPipe*: Enable the VI PIPE.
- *CVI_VI_StopPipe*: Disable the VI PIPE.
- *CVI_VI_SetPipeCrop*: Set the VI physical PIPE clipping function properties.
- *CVI_VI_GetPipeCrop*: Get the VI physical PIPE clipping function properties.
- *CVI_VI_SetPipeDumpAttr*: Set the VI physical PIPE dump property.
- *CVI_VI_GetPipeDumpAttr*: Get the VI physical PIPE dump property.
- *CVI_VI_SetPipeFrameSource*: Set the source of VI PIPE data.
- *CVI_VI_GetPipeFrameSource*: Get the source of VI PIPE data.
- *CVI_VI_GetPipeFrame*: Get the data of VI physical PIPE.
- *CVI_VI_ReleasePipeFrame*: Release the data of VI PIPE .
- *CVI_VI_SendPipeRaw*: Send raw data through VI PIPE.
- *CVI_VI_QueryPipeStatus*: View VI PIPE status.
- *CVI_VI_GetPipeFd*: Get the VI PIPE file descriptor.
- *CVI_VI_CloseFd*: Close the VI file descriptor.
- *CVI_VI_AttachVbPool* : Bind the VI channel to a video cache VB pool.
- *CVI_VI_DetachVbPool* : Unbind the VI channel from a video cache VB pool.
- *CVI_VI_SetChnAttr*: Set VI channel properties.
- *CVI_VI_GetChnAttr*: Get VI channel properties.
- *CVI_VI_EnableChn*: Enable the VI channel.
- *CVI_VI_DisableChn*: Disable the VI channel.

- *CVI_VI_SetChnCrop*: Set VI channel clipping function properties.
- *CVI_VI_GetChnCrop*: Get VI channel clipping function properties.
- *CVI_VI_GetChnFrame*: Capture an image frame from VI channel.
- *CVI_VI_ReleaseChnFrame*: Release an image captured from the VI channel.
- *CVI_VI_SetChnRotation*: Set the properties of VI channel rotation.
- *CVI_VI_GetChnRotation*: Get the properties of VI channel rotation.
- *CVI_VI_RegChnFlipMirrorCallBack* : Register VI channel callback function for flip and mirror.
- *CVI_VI_UnRegChnFlipMirrorCallBack* : Un-Register VI channel callback function for flip and mirror.
- *CVI_VI_SetChnFlipMirror* : Set the properties of VI channel flip and mirror.
- *CVI_VI_GetChnFlipMirror* : Get the properties of VI channel flip and mirror.

4.3.1 CVI_VI_SetDevAttr

【Description】

Set VI device properties. The basic device attributes default to some chip configurations.

【Syntax】

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

【Parameter】

Parameter	Description	Input/Output
ViDev	VI device number. Value range: [0, VI_MAX_DEV_NUM);	Input
pstDevAttr	The property pointer for VI device. Static properties	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: `cv_i_vi.h`, `cv_i_comm_vi.h`
- Library files: `libvpu.a`

【Note】

- Make sure the VI device is disabled before calling this interface. If the VI device is enabled, `CVI_VI_DisableDev` can be used to disable the device.
- The parameter `pstDevAttr` is mainly used to configure the video interface mode for a specified VI device, which is used to interface with peripheral cameras, sensors, or codecs. The supported interface modes include `MIPI_RX` (MIPI/LVDS/HISPI).
- Users need to configure the following information, and the specific attribute meanings are described in the “Data Types” section of Chapter 4.4:
 - Interface mode information: the interface mode can be `MIPI_RX` (MIPI/LVDS/HISPI) and other modes.

- Working mode information: 1 channel multiplexed mode.
- Data layout information: data arrangement under YUV data input.
- Data information: RGB, YUV data input.
- Synchronization timing information: Properties of vertical and horizontal synchronization signals.

【Example】

```

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_SIGNAL, 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)
{
    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);

```

(continues on next page)

(continued from previous page)

```

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;
}
}

```

【Related Topic】

- [CVI_VI_GetDevAttr](#)

4.3.2 CVI_VI_GetDevAttr

【Description】

Get VI device properties.

【Syntax】

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

【Parameter】

Parameter	Description	Input/Output
ViDev	VI device number. Value range: [0, VI_MAX_DEV_NUM).	Input
pstDevAttr	The property pointer for VI device.	Output

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: `cvi_vi.h`, `cvi_comm_vi.h`
- Library files: `libvpu.a`

【Note】

- If the VI device property is not set, the interface will return a failure.

【Example】

None.

【Related Topic】

- [CVI_VI_SetDevAttr](#)

4.3.3 CVI_VI_SetDevAttrEx

【Description】

Set VI device advanced properties.

【Syntax】

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

【Parameter】

Parameter	Description	Input/Output
ViDev	VI device number. Value range: [0, VI_MAX_DEV_NUM).	Input
pstDevAttrEx	The property pointer for VI device advanced. Static properties.	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: cvi_vi.h, cvi_comm_vi.h
- Library files: libvpu.a

【Note】

- This interface is not supported at this time.

【Example】

None.

【Related Topic】

None.

4.3.4 CVI_VI_GetDevAttrEx

【Description】

Get VI device advanced properties.

【Syntax】

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

【Parameter】

Parameter	Description	Input/Output
ViDev	VI device number. Value range: [0, VI_MAX_DEV_NUM).	Input
pstDevAttrEx	The property pointer for VI device advanced. Static properties.	Output

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: cvi_vi.h, cvi_comm_vi.h
- Library files: libvpu.a

【Note】

- This interface is not in use.

【Example】

None.

【Related Topic】

None.

4.3.5 CVI_VI_EnableDev

【Description】

Enable the VI device.

【Syntax】

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

【Parameter】

Parameter	Description	Input/Output
ViDev	VI device number. Value range: [0, VI_MAX_DEV_NUM).	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: cvi_vi.h, cvi_comm_vi.h
- Library files: libvpu.a

【Note】

- If the VI device property is not set, the interface will return failure.
- CV181x supports three VI DEV starting at the same time.
- CV180x supports one VI DEV starting at the same time.

【Example】

None.

【Related Topic】

- [CVI_VI_DisableDev](#)

4.3.6 CVI_VI_DisableDev

【Description】

Disable the VI device.

【Syntax】

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

【Parameter】

Parameter	Description	Input/Output
ViDev	VI device number. Value range: [0, VI_MAX_DEV_NUM).	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: cvi_vi.h, cvi_comm_vi.h
- Library files: libvpu.a

【Note】

- It is recommended to destroy all the physical PIPE bound to the VI device before disabling the VI device.
- After the VI device is disabled, the device will shut down completely. You need to reset the properties to enable the VI device.

【Example】

None.

【Related Topic】

- [CVI_VI_EnableDev](#)

4.3.7 CVI_VI_SetDevBindPipe

【Description】

Set the binding relationship between VI device and physical pipe.

【Syntax】

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

【Parameter】

Parameter	Description	Input/Output
ViDev	VI device number. Value range: [0, VI_MAX_DEV_NUM).	Input
pstDevBindPipe	Pointer to the structure of the physical PIPE information bound to the Dev.	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: cvi_vi.h, cvi_comm_vi.h
- Library files: libvpu.a

【Note】

- The VI device must be enabled before binding the physical pipe.
- This interface is not in use.

【Example】

None.

【Related Topic】

- [CVI_VI_EnableDev](#)

4.3.8 CVI_VI_GetDevBindPipe

【Description】

Get the physical pipe to which the VI device is bound.

【Syntax】

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

【Parameter】

Parameter	Description	Input/Output
ViDev	VI device number. Value range: [0, VI_MAX_DEV_NUM).	Input
pstDevBindPipe	Pointer to the structure of the physical PIPE information bound to the Dev.	Output

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: cvi_vi.h, cvi_comm_vi.h
- Library files: libvpu.a

【Note】

- Before using this interface, you need to enable the device and bind the physical pipe, otherwise it will return failure.
- This interface is not in use.

【Example】

None.

【Related Topic】

- [CVI_VI_SetDevBindPipe](#)

4.3.9 CVI_VI_SetDevTimingAttr**【Description】**

Set the self-generating timing property.

【Syntax】

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

【Parameter】

Parameter	Description	Input/Output
ViDev	VI device number. Value range: [0, VI_MAX_DEV_NUM).	Input
pstTimingAttr	See VI_DEV_TIMING_ATTR_S for details.	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: `cvi_vi.h`, `cvi_comm_vi.h`
- Library files: `libvpu.a`

【Note】

- Before using this interface, you need to configure the DEV attribute and enable the device, otherwise it will return failure.
- When filling RAW with self-generated timing function, the width and height of DEV/ PIPE/ CHN should be consistent with the width and height of RAW file.
- After the time sequence is generated, if RAW is not filled, there will be no image display; for the way of filling RAW, please refer to [CVI_VI_SendPipeRaw](#).
- After enabling self-generated timing, the output frame rate of VI is determined by the effective frame rate generated by configuring self-generated timing.

【Example】

None.

【Related Topic】

- [CVI_VI_EnableDev](#)
- [CVI_VI_SetDevAttr](#)

4.3.10 CVI_VI_GetDevTimingAttr

【Description】

Get the self-generating timing property.

【Syntax】

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

【Parameter】

Parameter	Description	Input/Output
ViDev	VI device number. Value range: [0, VI_MAX_DEV_NUM).	Input
pstTimingAttr	See VI_DEV_TIMING_ATTR_S for details.	Output

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: cvi_vi.h, cvi_comm_vi.h
- Library files: libvpu.a

【Note】

- Before using this interface, you need to configure DevTimingAttr and enable the device, otherwise it will return failure.

【Example】

None.

【Related Topic】

- [CVI_VI_EnableDev](#)
- [CVI_VI_SetDevTimingAttr](#)

4.3.11 CVI_VI_CreatePipe

【Description】

Create a VI PIPE.

【Syntax】

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

【Parameter】

Parameter	Description	Input/Output
ViPipe	VI physical PIPE number.	Input
pstPipeAttr	PIPE attribute structure pointer. See VI_PIPE_ATTR_S for description.	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: cvi_vi.h, cvi_comm_vi.h
- Library files: libvpu.a

【Note】

- Duplicate creation is not supported.

【Example】

None.

【Related Topic】

- [CVI_VI_EnableDev](#)

4.3.12 CVI_VI_DestroyPipe

【Description】

Destroy a VI PIPE.

【Syntax】

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

【Parameter】

Parameter	Description	Input/Output
ViPipe	VI physical PIPE number.	Input
pstPipeAttr	PIPE attribute structure pointer.	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: cvi_vi.h, cvi_comm_vi.h
- Library files: libvpu.a

【Note】

- When the pipe is not created or repeatedly destroyed, calling this interface will prompt that the pipe does not exist.

【Example】

None.

【Related Topic】

- [CVI_VI_CreatePipe](#)

4.3.13 CVI_VI_SetPipeAttr

【Description】

Set the properties of VI PIPE.

【Syntax】

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

【Parameter】

Parameter	Description	Input/Output
ViPipe	VI physical PIPE number.	Input
pstPipeAttr	PIPE attribute structure pointer. See VI_PIPE_ATTR_S for description.	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: cvi_vi.h, cvi_comm_vi.h
- Library files: libvpu.a

【Note】

- Before using this interface, you need to call CVI_VI_CreatePipe(ViPipe) first, otherwise it will fail.
- The PIPE attributes must be legal, and some of the static attributes cannot be dynamically set, please refer to VI_PIPE_ATTR_S for details.

【Example】

None.

【Related Topic】

None.

4.3.14 CVI_VI_GetPipeAttr

【Description】

Get the properties of VI PIPE

【Syntax】

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

【Parameter】

Parameter	Description	Input/Output
ViPipe	VI physical PIPE number.	Input
pstPipeAttr	PIPE attribute structure pointer. See VI_PIPE_ATTR_S for details.	Output

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: cvi_vi.h, cvi_comm_vi.h
- Library files: libvpu.a

【Note】

- Before using this interface, call CVI_VI_SetPipeAttr(ViPipe) first.

【Example】

None.

【Related Topic】

- [CVI_VI_SetPipeAttr](#)

4.3.15 CVI_VI_StartPipe

【Description】

Start VI PIPE.

【Syntax】

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

【Parameter】

Parameter	Description	Input/Output
ViPipe	VI physical PIPE number.	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: cvi_vi.h, cvi_comm_vi.h
- Library files: libvpu.a

【Note】

- Before using this interface, call CVI_VI_CreatePipe first.

【Example】

None.

【Related Topic】

- [CVI_VI_CreatePipe](#)

4.3.16 CVI_VI_StopPipe

【Description】

Disable VI PIPE.

【Syntax】

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

【Parameter】

Parameter	Description	Input/Output
ViPipe	VI physical PIPE number.	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: cvi_vi.h, cvi_comm_vi.h
- Library files: libvpu.a

【Note】

- Before using this interface, the PIPE must have already been created.

【Example】

None.

【Related Topic】

- [CVI_VI_CreatePipe](#)

4.3.17 CVI_VI_SetPipeCrop

【Description】

Set the VI physical PIPE clipping function properties.

【Syntax】

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

【Parameter】

Parameter	Description	Input/Output
ViPipe	VI physical PIPE number.	Input
pstCropInfo	Pointer to structure of clipping function parameter.	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: cvi_vi.h, cvi_comm_vi.h
- Library files: libvpu.a

【Note】

- Before using this interface, the PIPE must have already been created.
- This interface has the same effect as VI_SetChnCrop.

【Example】

None.

【Related Topic】

None.

4.3.18 CVI_VI_GetPipeCrop

【Description】

Get the properties of VI physical PIPE clipping function.

【Syntax】

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

【Parameter】

Parameter	Description	Input/Output
ViPipe	VI physical PIPE number.	Input
pstCropInfo	Pointer to structure of clipping function parameter.	Output

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: cvi_vi.h, cvi_comm_vi.h
- Library files: libvpu.a

【Note】

- Before using this interface, the PIPE must have already been created.

【Example】

None.

【Related Topic】

None.

4.3.19 CVI_VI_SetPipeDumpAttr

【Description】

Set the VI physical PIPE dump property.

【Syntax】

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

【Parameter】

Parameter	Description	Input/Output
ViPipe	Physical pipe number. Value range: [0, VI_MAX_PHY_PIPE_NUM - 1).	Input
pstDumpAttr	Attributes for dumping the VI physical PIPE. See VI_DUMP_ATTR_S for details.	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: cvi_vi.h, cvi_comm_vi.h
- Library files: libvpu.a

【Note】

- Supports dumping 12-bit uncompressed or 6-bit compressed RAW files.
- If the file is in YUV format, it is recommended to use CVI_VI_GetChnFrame.
- Dumping IR data is not supported.

【Example】

None.

【Related Topic】

None.

4.3.20 CVI_VI_GetPipeDumpAttr

【Description】

Get the VI physical PIPE dump property.

【Syntax】

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

【Parameter】

Parameter	Description	Input/Output
ViPipe	Physical pipe number. Value range: [0, VI_MAX_PHY_PIPE_NUM - 1).	Input
pstDumpAttr	Attributes for dumping the VI physical PIPE. See VI_DUMP_ATTR_S for details.	Output

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: `cvi_vi.h`, `cvi_comm_vi.h`
- Library files: `libvpu.a`

【Note】

- PIPE must have already been created.

【Example】

None.

【Related Topic】

- [*CVI_VI_SetPipeDumpAttr*](#)

4.3.21 CVI_VI_SetPipeFrameSource

【Description】

Set the source of VI PIPE data.

【Syntax】

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

【Parameter】

Parameter	Description	Input/Output
ViPipe	VI physical PIPE number.	Input
enSource	The data source of PIPE.	Output

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: `cvi_vi.h`, `cvi_comm_vi.h`
- Library files: `libvpu.a`

【Note】

- PIPE must have been created.

【Example】

None.

【Related Topic】

None.

4.3.22 CVI_VI_GetPipeFrameSource

【Description】

Get the source of VI PIPE data.

【Syntax】

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

【Parameter】

Parameter	Description	Input/Output
ViPipe	VI device number. Value range: [0, VI_MAX_DEV_NUM).	Input
penSource	The data source of PIPE.	Output

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: cvi_vi.h, cvi_comm_vi.h
- Library files: libvpu.a

【Note】

- PIPE must have been created.

【Example】

None.

【Related Topic】

None.

4.3.23 CVI_VI_GetPipeFrame

【Description】

Get the data of VI physical pipe.

【Syntax】

```
CVI_S32 CVI_VI_GetPipeFrame(VI_PIPE ViPipe, VIDEO_FRAME_INFO_S *pstVideoFrame, CVI_
↪S32 s32MilliSec);
```

【Parameter】

Parameter	Description	Input/Output
ViPipe	Physical pipe number. Value range: [0, VI_MAX_PHY_PIPE_NUM - 1).	Input
pstVideoFrame	Pointer to VI PIPE data information.	Output
s32MilliSec	Timeout parameter: the unit of timeout is in milliseconds (ms).	Input/Output

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: cvi_vi.h, cvi_comm_vi.h
- Library files: libvpu.a

【Note】

- PIPE must have been created.
- Call CVI_VI_SetPipeDumpAttr to set the dump property, enable dump, and set depth, otherwise raw data cannot be obtained.
- The timeout parameter s32MilliSec and the valid range for setting is starting from 0.
- If the raw data is obtained earlier than the timeout, the function returns.

【Example】

None.

【Related Topic】

- [CVI_VI_SetPipeDumpAttr](#)

4.3.24 CVI_VI_ReleasePipeFrame

【Description】

Release VI PIPE data

【Syntax】

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

【Parameter】

Parameter	Description	Input/Output
ViPipe	Physical PIPE number. Value range: [0, VI_MAX_PHY_PIPE_NUM - 1);	Input
pstVideoFrame	Pointer to VI PIPE data information.	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: cvi_vi.h, cvi_comm_vi.h
- Library files: libvpu.a

【Note】

- This interface should be used in conjunction with CVI_VI_GetPipeFrame. Failure to release the frame obtained by GetPipeFrame will result in insufficient system memory, which will affect the operation of VI.

- The user must ensure that the information in the pstVideoFrame structure is consistent with that used in GetPipeFrame, otherwise the release may not be successful.

【Example】

None.

【Related Topic】

- [CVI_VI_GetPipeFrame](#)

4.3.25 CVI_VI_SendPipeRaw

【Description】

Send RAW data through VI PIPE.

【Syntax】

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

【Parameter】

Parameter	Description	Input/Output
u32PipeNum	Number of Pipes. Always filled with a value of 1.	Input
PipeId	PIPE number array, which size is u32PipeNum.	Input
pstVideoFrame	RAW data information.	Input
s32MilliSec	The timeout parameter: If it is greater than 0, it indicates timeout mode, and the unit of timeout time is milliseconds (ms).	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: cvi_vi.h, cvi_comm_vi.h
- Library files: libvpu.a

【Note】

- PIPE must be created and started.
- To send RAW, the source of PIPE data needs to be set using CVI_VI_SetPipeFrameSource.
- Support user-defined timing to load RAW data.
- Only supports non-compressed mode.

【Example】

None.

【Related Topic】

- [CVI_VI_SetPipeFrameSource](#)

4.3.26 CVI_VI_QueryPipeStatus

【Description】

Query VI PIPE status.

【Syntax】

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

【Parameter】

Parameter	Description	Input/Output
ViPipe	PIPE number.	Input
pstStatus	PIPE status information.	Output

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: cvi_vi.h, cvi_comm_vi.h
- Library files: libvpu.a

【Note】

- PIPE must be created and started.

【Example】

None.

【Related Topic】

None.

4.3.27 CVI_VI_GetPipeFd

【Description】

Get the VI PIPE file descriptor.

【Syntax】

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

【Parameter】

Parameter	Description	Input/Output
ViPipe	PIPE number.	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: cvi_vi.h, cvi_comm_vi.h
- Library files: libvpu.a

【Note】

- PIPE must be created and started.

【Example】

None.

【Related Topic】

None.

4.3.28 CVI_VI_CloseFd

【Description】

Close the VI file descriptor.

【Syntax】

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

【Parameter】

None.

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: cvi_vi.h, cvi_comm_vi.h
- Library files: libvpu.a

【Note】

- This interface is not supported at this time.

【Example】

None.

【Related Topic】

None.

4.3.29 CVI_VI_AttachVbPool

【Description】

Bind the VI channel to a video cache VB pool.

【Syntax】

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

【Parameter】

Parameter	Description	Input/Output
ViPipe	PIPE number.	Input
ViChn	VI channel number.	Input
VbPool	video cache VB pool info.	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: cvi_vi.h, cvi_comm_vb.h
- Library files: libvpu.a

【Note】

- ViPipe and ViChn must conform to the range of values.
- VbPool must be guaranteed to be a valid pool ID for the VB pool that has been created.

【Example】

None.

【Related Topic】

None.

4.3.30 CVI_VI_DetachVbPool

【Description】

Unbind the VI channel from a video cache VB pool

【Syntax】

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

【Parameter】

Parameter	Description	Input/Output
ViPipe	PIPE number.	Input
ViChn	VI channel number.	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: cvi_vi.h, cvi_comm_vi.h
- Library files: libvpu.a

【Note】

None.

【Example】

None.

【Related Topic】

None.

4.3.31 CVI_VI_SetChnAttr

【Description】

Set VI channel properties.

【Syntax】

```
CVI_S32 CVI_VI_SetChnAttr(VI_PIPE ViPipe, VI_CHN ViChn, const VI_CHN_ATTR_SU
↪*pstChnAttr);
```

【Parameter】

Parameter	Description	Input/Output
ViPipe	PIPE number.	Input
ViChn	VI channel number.	Input
pstChnAttr	VI channel property structure pointer.	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: cvi_vi.h, cvi_comm_vi.h
- Library files: libvpu.a

【Note】

- arget image size stSize: it is required to configure and the size needs to be within the range supported by the VI.
- Pixel format enPixelFormat: Output Pixel format supports NV21.
- Frame Rate Control stFrameRate: Does not support frame rate control.

【Example】

None.

【Related Topic】

None.

4.3.32 CVI_VI_GetChnAttr

【Description】

Gets the VI channel properties.

【Syntax】

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

【Parameter】

Parameter	Description	Input/Output
ViPipe	PIPE number.	Input
ViChn	VI channel number.	Input
pstChnAttr	VI channel property structure pointer.	Output

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: cvi_vi.h, cvi_comm_vi.h
- Library files: libvpu.a

【Note】

- PIPE must have been created or it will return a failure.
- Channel properties need to be set before they can be obtained.

【Example】

None.

【Related Topic】

None.

4.3.33 CVI_VI_EnableChn

【Description】

Enable VI channel.

【Syntax】

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

【Parameter】

Parameter	Description	Input/Output
ViPipe	PIPE number.	Input
ViChn	VI channel number.	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: cvi_vi.h, cvi_comm_vi.h
- Library files: libvpu.a

【Note】

- PIPE must have been created or it will return a failure.
- Channel properties must be set first.
- Before enabling a VI channel, the channel needs to be turned off first.

【Example】

None.

【Related Topic】

None.

4.3.34 CVI_VI_DisableChn

【Description】

Disable the VI channel.

【Syntax】

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

【Parameter】

Parameter	Description	Input/Output
ViPipe	PIPE number.	Input
ViChn	VI channel number.	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: cvi_vi.h, cvi_comm_vi.h
- Library files: libvpu.a

【Note】

- PIPE must have been created or it will return a failure.
- Channel properties must be set first.
- The VI channel needs to be enabled first.

【Example】

None.

【Related Topic】

None.

4.3.35 CVI_VI_SetChnCrop**【Description】**

Set VI channel clipping function properties.

【Syntax】

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

【Parameter】

Parameter	Description	Input/Output
ViPipe	PIPE number.	Input
ViChn	VI channel number.	Input
pstCropInfo	Clipping Function parameter Structure Pointer.	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: cvi_vi.h, cvi_comm_vi.h
- Library files: libvpu.a

【Note】

- PIPE must have been created or it will return a failure.
- Using this API for cropping will result in some parts of the image being lost.
- After using this API for cropping, the input size of VPSS must be the same as the size of pstCropInfo

【Example】

None.

【Related Topic】

None.

4.3.36 CVI_VI_GetChnCrop**【Description】**

Gets the VI channel clipping function properties.

【Syntax】

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


【Parameter】

Parameter	Description	Input/Output
ViPipe	PIPE number.	Input
ViChn	VI channel number.	Input
pstCropInfo	Clipping Function parameter Structure Pointer.	Output

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: cvi_vi.h, cvi_comm_vi.h
- Library files: libvpu.a

【Note】

- PIPE must have been created or it will return a failure.
- Property must be set using SetChnCrop before it can be obtained.

【Example】

None.

【Related Topic】

None.

4.3.37 CVI_VI_GetChnFrame

【Description】

Obtain captured images from the VI channel.

【Syntax】

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

【Parameter】

Parameter	Description	Input/Output
ViPipe	PIPE number.	Input
ViChn	VI channel number.	Input
pstFrameInfo	VI Frame Information Structure Pointer.	Output
s32MilliSec	Timeout parameter: -1 indicates blocking mode; 0 indicates non-blocking mode; A value greater than 0 indicates a time-out mode in milliseconds (ms).	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: cvi_vi.h, cvi_comm_vi.h
- Library files: libvpu.a

【Note】

- PIPE PIPE must have been created or it will return a failure
- Channel properties must be set first.
- The physical address information obtained comes from the VideoBuffer used internally by CVI, so after use, you must call the CVI_VI_ReleaseChnFrame interface to free its memory.
- pstFrameInfo->stVFrame.u64PhyAddr[0] and pstFrameInfo->stVFrame.u64PhyAddr[1]/[2] points to the physical addresses of the brightness and chroma components of the image.
- When u32MilliSec is set to -1, it represents the blocking mode, and the program waits until an image is obtained before returning. If u32MilliSec is greater than 0, it represents the non-blocking mode, and the unit of the parameter is milliseconds, indicating the timeout period. If an image is not obtained within this time, a timeout occurs and the function returns.

【Example】

None.

【Related Topic】

- [CVI_VI_ReleaseChnFrame](#)
- [CVI_VI_SetChnAttr](#)

4.3.38 CVI_VI_ReleaseChnFrame

【Description】

Release a frame of the image obtained from the VI channel.

【Syntax】

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

【Parameter】

Parameter	Description	Input/Output
ViPipe	PIPE number.	Input
ViChn	VI channel number.	Input
pstFrameInfo	VI Frame Information Structure Pointer.	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: cvi_vi.h, cvi_comm_vi.h
- Library files: libvpu.a

【Note】

- PIPE must have been created or it will return a failure.

- Channel properties must be set first.
- This interface must be paired with `CVI_VI_GetChnFrame`.
- Users must ensure that the information in the `pstFrameInfo` structure is consistent with what they get, or the release may be unsuccessful.

【Example】

None.

【Related Topic】

- [CVI_VI_GetChnFrame](#)
- [CVI_VI_SetChnAttr](#)

4.3.39 CVI_VI_SetChnRotation

【Description】

Set the VI channel rotation properties.

【Syntax】

```
CVI_S32 CVI_VI_SetChnRotation(VI_PIPE ViPipe, VI_CHN ViChn, const ROTATION_E_↵
↵enRotation);
```

【Parameter】

Parameter	Description	Input/Output
ViPipe	VI physical PIPE number. Value range: [0, VI_MAX_PHY_PIPE_NUM - 1).	Input
ViChn	VI physical channel number. Value range: [0, VI_MAX_PHY_CHN_NUM - 1).	Input
enRotation	Rotation property. Please refer to ROTATION_E for detail instruction.	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: `cvi_vi.h`, `cvi_comm_vi.h`
- Library files: `libvpu.a`

【Note】

- You need to call `CVI_VI_CreatePipi(ViPipe)` before using this interface, otherwise an error message will be displayed.
- Channel property must be set first.
- Rotation is only supported for three formats: RGB planar, YUV420 planar, and YUV400.
- After rotation, the memory size of the image output from the channel may change, but the size obtained from the channel is still the value set by the user.
- For example, an input image of 1920x1080 rotates 90 degrees and the actual output is 1080x1920.
- Do not support `VI_ONLINE_VPSS_ONLINE/VI_OFFLINE_VPSS_ONLINE` working mode.

【Example】

None.

【Related Topic】

None.

4.3.40 CVI_VI_GetChnRotation

【Description】

Get the properties of the VI channel rotation.

【Syntax】

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

【Parameter】

Parameter	Description	Input/Output
ViPipe	VI physical PIPE number. Value range: [0, VI_MAX_PHY_PIPE_NUM - 1).	Input
ViChn	VI physical channel number. Value range: [0, VI_MAX_PHY_CHN_NUM - 1).	Input
penRotation	The property pointer for rotate. please refer to ROTATION_E for detail instructions.	Output

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: cvi_vi.h, cvi_comm_vi.h
- Library files: libvpu.a

【Note】

- You need to call CVI_VI_CreatePipi(ViPipe) before using this interface, otherwise an error message will be displayed.
- Does not support VI_ONLINE_VPSS_ONLINE/ VI_OFFLINE_VPSS_ONLINE working mode.

【Example】

None.

【Related Topic】

- [CVI_VI_SetChnRotation](#)

4.3.41 CVI_VI_SetChnLDCAttr

【Description】

Sets the properties of the lens distortion correction for the VI channel.

【Syntax】

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

【Parameter】

Parameter	Description	Input/Output
ViPipe	VI physical PIPE number. Value range: [0, VI_MAX_PHY_PIPE_NUM - 1).	Input
ViChn	VI physical channel number. Value range: [0, VI_MAX_PHY_CHN_NUM - 1).	Input
pstLDCAttr	Lens distortion correction properties.	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: cvi_vi.h, cvi_comm_vi.h
- Library files: libvpu.a

【Note】

- You need to call CVI_VI_CreatePipi(ViPipe) before using this interface, otherwise an error message will be displayed.
- Does not support VI_ONLINE_VPSS_ONLINE/ VI_OFFLINE_VPSS_ONLINE working mode.
- Channel property must be set before it can be set.

【Example】

None.

【Related Topic】

- [CVI_VI_GetChnLDCAttr](#)
- VI_LDC_ATTR_S

4.3.42 CVI_VI_GetChnLDCAttr

【Description】

Get the properties of the lens distortion correction for the VI channel.

【Syntax】

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

【Parameter】

Parameter	Description	Input/Output
ViPipe	VI physical PIPE number. Value range: [0, VI_MAX_PHY_PIPE_NUM - 1).	Input
ViChn	VI physical channel number. Value range: [0, VI_MAX_PHY_CHN_NUM - 1).	Input
pstLDCAttr	Lens distortion correction properties.	Output

【Return Value】

Return Value	Description
0	Success.
non0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: cvi_vi.h, cvi_comm_vi.h
- Library files: libvpu.a

【Note】

- You need to call CVI_VI_CreatePipi(ViPipe) before using this interface, otherwise an error message will be displayed.
- Does not support VI_ONLINE_VPSS_ONLINE/ VI_OFFLINE_VPSS_ONLINE working mode.

【Example】

None.

【Related Topic】

- [CVI_VI_SetChnLDCAttr](#)
- VI_LDC_ATTR_S

4.3.43 CVI_VI_RegChnFlipMirrorCallBack

【Description】

Register VI channel callback function for flip and mirror.

【Syntax】

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

【Parameter】

Parameter	Description	Input/Output
ViPipe	VI physical PIPE number.	Input
ViDev	VI device number.	Input
pvData	Callback function for flip and mirror pointer.	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: cvi_vi.h, cvi_comm_vi.h
- Library files: libvpu.a

【Note】

None.

【Example】

None.

【Related Topic】

None.

4.3.44 CVI_VI_UnRegChnFlipMirrorCallBack

【Description】

Un-Register VI channel callback function for flip and mirror.

【Syntax】

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

【Parameter】

Parameter	Description	Input/Output
ViPipe	VI physical PIPE number.	Input
ViDev	VI device number.	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: cvi_vi.h, cvi_comm_vi.h
- Library files: libvpu.a

【Note】

None.

【Example】

None.

【Related Topic】

None.

4.3.45 CVI_VI_SetChnFlipMirror

【Description】

Set the properties of VI channel flip and mirror.

【Syntax】

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

【Parameter】

Parameter	Description	Input/Output
ViPipe	VI physical PIPE number.	Input
ViChn	VI channel number.	Input
bFlip	Flip enable switch.	Input
bMirror	Mirror enable switch.	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: cvi_vi.h, cvi_comm_vi.h
- Library files: libvpu.a

【Note】

- Callback function must be register first.

【Example】

None.

【Related Topic】

None.

4.3.46 CVI_VI_GetChnFlipMirror

【Description】

Get the properties of VI channel flip and mirror.

【Syntax】

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

【Parameter】

Parameter	Description	Input/Output
ViPipe	VI physical PIPE number.	Input
ViChn	VI channel number.	Input
pbFlip	The property pointer for flip.	Output
pbMirror	The property pointer for mirror.	Output

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: `cvi_vi.h`, `cvi_comm_vi.h`
- Library files: `libvpu.a`

【Note】

None.

【Example】

None.

【Related Topic】

None.

4.4 Data Types

The data types related to video input are defined as follows. For the differences between chips, see the supporting capabilities of each chip.

- `VI_MAX_DEV_NUM`: Define the maximum number of VI devices.
- `VI_MAX_PHY_PIPE_NUM`: Defines the maximum number of VI physical PIPE.
- `VI_MAX_VIR_PIPE_NUM`: Define the maximum number of VI virtual PIPE.
- `VI_MAX_PIPE_NUM`: Define the maximum number of VI PIPE
- `VI_MAX_PHY_CHN_NUM`: Defines the maximum number of VI physical channels.
- `VI_MAX_CHN_NUM`: Defines the total number of VI physical channels and extended channels.
- `VI_DEV_MIN_WIDTH`: The minimum width of the image captured by the VI device.
- `VI_DEV_MIN_HEIGHT`: The minimum height of the image captured by the VI device.
- `VI_DEV_MAX_WIDTH`: The maximum width of the image captured by the VI device.
- `VI_DEV_MAX_HEIGHT`: The maximum height of the image captured by the VI device.
- `VI_PIPE_OFFLINE_MIN_WIDTH`: Minimum width of images for offline processing with VI PIPE
- `VI_PIPE_OFFLINE_MIN_HEIGHT`: Minimum height of images for offline processing with VI PIPE
- `VI_PIPE_OFFLINE_MAX_WIDTH`: Maximum width of images for offline processing with VI PIPE
- `VI_PIPE_OFFLINE_MAX_HEIGHT`: Maximum height of images for offline processing with VI PIPE
- `VI_PIPE_ONLINE_MIN_WIDTH`: Minimum width of images for online processing with VI PIPE
- `VI_PIPE_ONLINE_MIN_HEIGHT`: Minimum height of images for online processing with VI PIPE

- *VI_PIPE_ONLINE_MAX_WIDTH*: Maximum width of images for online processing with VI PIPE
- *VI_PIPE_ONLINE_MAX_HEIGHT*: Maximum height of images for online processing with VI PIPE
- *VI_PIPE0_MAX_WIDTH*: Maximum width of images for processing with VI PIPE0
- *VI_PIPE0_MAX_HEIGHT*: Maximum height of images for processing with VI PIPE0
- *VI_PIPE1_MAX_WIDTH*: Maximum width of images for processing with VI PIPE1
- *VI_PIPE1_MAX_HEIGHT*: Maximum height of images for processing with VI PIPE1
- *VI_PIPE2_MAX_WIDTH*: Maximum width of images for processing with VI PIPE2
- *VI_PIPE2_MAX_HEIGHT*: Maximum height of images for processing with VI PIPE2
- *VI_PIPE3_MAX_WIDTH*: Maximum width of images for processing with VI PIPE3
- *VI_PIPE3_MAX_HEIGHT*: Maximum height of images for processing with VI PIPE3
- *VI_PIPE4_MAX_WIDTH*: Maximum width of images for processing with VI PIPE4
- *VI_PIPE4_MAX_HEIGHT*: Maximum height of images for processing with VI PIPE4
- *VI_DATA_TYPE_E*: VI input data type enumeration.
- *VI_DEV_ATTR_S*: Define the properties of the video input device.
- *VI_DEV_BIND_PIPE_S*: Define the binding relationship between VI DEV and PIPE.
- *VI_PIPE_ATTR_S*: Define VI PIPE properties.
- *VI_DUMP_TYPE_E*: Enumerate dump types.
- *VI_DUMP_ATTR_S*: Define the VI PIPE dump property.
- *VI_CHN_ATTR_S*: Define VI channel properties.
- *VI_CROP_INFO_S*: Define the VI CROP information structure.
- *VI_DEV_TIMING_ATTR_S*: Self-generated timing attribute
- *VI_PIPE_STATUS_S*: Define VI PIPE status information.
- *VI_CHN_STATUS_S*: Define VI channel status information.
- *VI_PIPE_FRAME_SOURCE_E*: Define the source type of VI PIPE data.
- *VI_LDC_ATTR_S*: Define the VI lens distortion correction structure

4.4.1 VI_MAX_DEV_NUM

【Description】

Defines the maximum number of VI devices.

【Syntax】

```
#define VI_MAX_DEV_NUM 3
```

【Note】

None.

【Related Data Type and Interface】

None.

4.4.2 VI_MAX_PHY_PIPE_NUM

【Description】

Defines the maximum number of VI physical PIPE.

【Syntax】

```
#define VI_MAX_PHY_PIPE_NUM 5
```

【Note】

None.

【Related Data Type and Interface】

None.

4.4.3 VI_MAX_VIR_PIPE_NUM

【Description】

Define the maximum number of VI virtual PIPE.

【Syntax】

```
#define VI_MAX_VIR_PIPE_NUM 0
```

【Note】

None.

【Related Data Type and Interface】

None.

4.4.4 VI_MAX_PIPE_NUM

【Description】

Define the maximum number of VI PIPE.

【Syntax】

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

【Note】

None.

【Related Data Type and Interface】

None.

4.4.5 VI_MAX_PHY_CHN_NUM

【Description】

Defines the maximum number of VI physical channels.

【Syntax】

```
#define VI_MAX_PHY_CHN_NUM 3
```

【Note】

None.

【Related Data Type and Interface】

None.

4.4.6 VI_MAX_CHN_NUM

【Description】

Defines the total number of VI physical channels and extended channels.

【Syntax】

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

【Note】

None.

【Related Data Type and Interface】

None.

4.4.7 VI_DEV_MIN_WIDTH

【Description】

The minimum width of the image captured by the VI device.

【Syntax】

```
#define VI_DEV_MIN_WIDTH 120
```

【Note】

None.

【Related Data Type and Interface】

None.

4.4.8 VI_DEV_MIN_HEIGHT

【Description】

The minimum height of the image captured by the VI device

【Syntax】

```
#define VI_DEV_MIN_HEIGHT 120
```

【Note】

None.

【Related Data Type and Interface】

None.

4.4.9 VI_DEV_MAX_WIDTH

【Description】

The maximum width of the image captured by the VI device.

【Syntax】

```
#define VI_DEV_MAX_WIDTH 4608
```

【Note】

None.

【Related Data Type and Interface】

None.

4.4.10 VI_DEV_MAX_HEIGHT

【Description】

The maximum height of the image captured by the VI device.

【Syntax】

```
#define VI_DEV_MAX_HEIGHT 4608
```

【Note】

None.

【Related Data Type and Interface】

None.

4.4.11 VI_PIPE_OFFLINE_MIN_WIDTH

【Description】

Minimum width of images for offline processing with VI PIPE

【Syntax】

```
#define VI_PIPE_OFFLINE_MIN_WIDTH 120
```

【Note】

None.

【Related Data Type and Interface】

None.

4.4.12 VI_PIPE_OFFLINE_MIN_HEIGHT

【Description】

Minimum height of images for offline processing with VI PIPE

【Syntax】

```
#define VI_PIPE_OFFLINE_MIN_HEIGHT 120
```

【Note】

None.

【Related Data Type and Interface】

None.

4.4.13 VI_PIPE_OFFLINE_MAX_WIDTH

【Description】

Maximum width of images for offline processing with VI PIPE

【Syntax】

```
#define VI_PIPE_OFFLINE_MAX_WIDTH 4096
```

【Note】

None.

【Related Data Type and Interface】

None.

4.4.14 VI_PIPE_OFFLINE_MAX_HEIGHT

【Description】

Maximum height of images for offline processing with VI PIPE

【Syntax】

```
#define VI_PIPE_OFFLINE_MAX_HEIGHT 2160
```

【Note】

None.

【Related Data Type and Interface】

None.

4.4.15 VI_PIPE_ONLINE_MIN_WIDTH

【Description】

Minimum width of images for online processing with VI PIPE

【Syntax】

```
#define VI_PIPE_ONLINE_MIN_WIDTH 120
```

【Note】

None.

【Related Data Type and Interface】

None.

4.4.16 VI_PIPE_ONLINE_MIN_HEIGHT

【Description】

Minimum height of images for online processing with VI PIPE

【Syntax】

```
#define VI_PIPE_ONLINE_MIN_HEIGHT 120
```

【Note】

None.

【Related Data Type and Interface】

None.

4.4.17 VI_PIPE_ONLINE_MAX_WIDTH

【Description】

Maximum width of images for online processing with VI PIPE

【Syntax】

```
#define VI_PIPE_ONLINE_MAX_WIDTH 2688
```

【Note】

None.

【Related Data Type and Interface】

None.

4.4.18 VI_PIPE_ONLINE_MAX_HEIGHT

【Description】

Maximum height of images for online processing with VI PIPE

【Syntax】

```
#define VI_PIPE_ONLINE_MAX_HEIGHT 1944
```

【Note】

None.

【Related Data Type and Interface】

None.

4.4.19 VI_PIPE0_MAX_WIDTH

【Description】

Maximum width of images for processing with VI PIPE0

【Syntax】

```
#define VI_PIPE0_MAX_WIDTH 4096
```

【Note】

None.

【Related Data Type and Interface】

None.

4.4.20 VI_PIPE0_MAX_HEIGHT

【Description】

Maximum height of images for processing with VI PIPE0

【Syntax】

```
#define VI_PIPE0_MAX_HEIGHT 2160
```

【Note】

None.

【Related Data Type and Interface】

None.

4.4.21 VI_PIPE1_MAX_WIDTH

【Description】

Maximum width of images for processing with VI PIPE1

【Syntax】

```
#define VI_PIPE1_MAX_WIDTH 4096
```

【Note】

None.

【Related Data Type and Interface】

None.

4.4.22 VI_PIPE1_MAX_HEIGHT

【Description】

Maximum height of images for processing with VI PIPE1

【Syntax】

```
#define VI_PIPE2_MAX_WIDTH 2688
```

【Note】

None.

【Related Data Type and Interface】

None.

4.4.23 VI_PIPE2_MAX_WIDTH

【Description】

Maximum width of images for processing with VI PIPE2

【Syntax】

```
#define VI_PIPE2_MAX_WIDTH 2688
```

【Note】

None.

【Related Data Type and Interface】

None.

4.4.24 VI_PIPE2_MAX_HEIGHT

【Description】

Maximum height of images for processing with VI PIPE2

【Syntax】

```
#define VI_PIPE2_MAX_HEIGHT 1944
```

【Note】

None.

【Related Data Type and Interface】

None.

4.4.25 VI_PIPE3_MAX_WIDTH

【Description】

Maximum width of images for processing with VI PIPE3

【Syntax】

```
#define VI_PIPE3_MAX_WIDTH 2688
```

【Note】

None.

【Related Data Type and Interface】

None.

4.4.26 VI_PIPE3_MAX_HEIGHT

【Description】

Maximum height of images for processing with VI PIPE3

【Syntax】

```
#define VI_PIPE3_MAX_HEIGHT 1944
```

【Note】

None.

【Related Data Type and Interface】

None.

4.4.27 VI_PIPE4_MAX_WIDTH

【Description】

Maximum width of images for processing with VI PIPE4

【Syntax】

```
#define VI_PIPE4_MAX_WIDTH 2688
```

【Note】

None.

【Related Data Type and Interface】

None.

4.4.28 VI_PIPE4_MAX_HEIGHT

【Description】

Maximum height of images for processing with VI PIPE4

【Syntax】

```
#define VI_PIPE4_MAX_HEIGHT 1944
```

【Note】

None.

【Related Data Type and Interface】

None.

4.4.29 VI_DATA_TYPE_E

【Description】

VI input data type enumeration.

【Syntax】

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

【Note】

None.

【Related Data Type and Interface】

None.

4.4.30 VI_DEV_ATTR_S

【Description】

Define the properties of the VI device.

【Syntax】

```
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 chn_num;  
    CVI_U32 snrFps;  
} VI_DEV_ATTR_S;
```

【Member】

Member	Description
enIntfMode	Sensor interface mode.
enWorkMode	1, 2, 4-channel composite working mode.
enScanMode	Input scanning mode (progressive, interlaced).
as32AdChnId[VI_MAX_ADCHN_NUM]	The value range is [-1, 3], and it is recommended to set it to the default value of -1 uniformly. This parameter is meaningless.
enDataSeq	Input data order (only applicable to YUV format).
stSynCfg	Synchronous timing configuration, which must be configured in BT.601 mode and is invalid in other modes.
enInputDataType	Input data type, which is generally RGB for Sensor input and YUV for AD input
stSize	The VI device can set the height and width of the image to be captured. Minimum and Maximum widths and heights of captured images: width : [VI_DEV_MIN_WIDTH, VI_DEV_MAX_WIDTH] height: [VI_DEV_MIN_HEIGHT, VI_DEV_MAX_HEIGHT]
stWDRAttr	WDR property
enBayerFormat	The Bayer format of the device, which must be set when the inputDataType is RGB

【Note】

- The u32Width in stSize must be equal to the width of the actual input image, and the u32Height must be equal to the height of the actual input image, otherwise there no image will be output.

【Related Data Type and Interface】

None.

4.4.31 VI_DEV_BIND_PIPE_S

【Description】

Define the binding relationship between VI DEV and PIPE.

【Syntax】

```
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;
```

【Member】

Member	Description
U32Num	Number of PIPEs bound to this VI Dev, Value range: [1, VI_MAX_PIPE_NUM]
PipeId	The PIPE ID bound to the VI Dev.

【Note】

None.

【Related Data Type and Interface】

None.

4.4.32 VI_PIPE_ATTR_S

【Description】

Define VI PIPE properties.

【Syntax】

```
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;
```

【Member】

Member	Description
enPipeBypassMode	Bypass mode for VI PIPE.
bYuvSkip	Whether to turn off downsampling and CSC. CVI_FALSE: yuv skip unenable CVI_TRUE: yuv skip enable
bIspBypass	Whether the ISP is bypassed. CVI_FALSE: ISP working properly. CVI_TRUE: ISP bypass, do not run ISP.
u32MaxW	The input image width. This is a static attribute set when the PIPE is created and cannot be changed. Value range under online mode:: [VI_PIPE_ONLINE_MIN_WIDTH, VI_PIPE_ONLINE_MAX_WIDTH] Value range under offline mode: [VI_PIPE_OFFLINE_MIN_WIDTH, VI_PIPE_OFFLINE_MAX_WIDTH]
u32MaxH	The input image height. Value range: Value range under online mode: [VI_PIPE_ONLINE_MIN_HEIGHT, VI_PIPE_ONLINE_MAX_HEIGHT] Value range under offline mode: [VI_PIPE_OFFLINE_MIN_HEIGHT, VI_PIPE_OFFLINE_MAX_HEIGHT]
enPixFmt	Pixel format.
enCompressMode	Data compression format.
enBitWidth	The bit width of the input image. Set during PIPE creation and cannot be changed, only valid when pixel format enPixFmt is a YUV pixel format.
bNrEn	NR enable switch. CVI_FALSE: unenable; CVI_TRUE: enable
bSharpenEn	Sharpen enable switch.
stFrameRate	Frame rate control.
bDiscardProPic	Whether to discard long exposure frames.
bYuvBypassPath	Whether to switch to YUV pass-through mode

【Note】

None.

【Related Data Type and Interface】

None.

4.4.33 VI_DUMP_TYPE_E

【Description】

Enumerate dump types.

【Syntax】

```
typedef enum _VI_DUMP_TYPE_E {
    VI_DUMP_TYPE_RAW = 0,
    VI_DUMP_TYPE_YUV = 1,
    VI_DUMP_TYPE_IR = 2,
```

(continues on next page)

(continued from previous page)

```

    VI_DUMP_TYPE_BUTT
} VI_DUMP_TYPE_E;

```

【Member】

Member	Description
VI_DUMP_TYPE_RAW	Dump RAW data
VI_DUMP_TYPE_YUV	Dump YUV data
VI_DUMP_TYPE_IR	Dump IR Component data.

【Note】

- Does not support Dump IR data
- To Dump YUV, run CVI_VI_GetChnFrame

【Related Data Type and Interface】

None.

4.4.34 VI_DUMP_ATTR_S

【Description】

Define the PIPE dump property.

【Syntax】

```

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;

```

【Member】

Member	Description
benable	Whether to enable dump.
u32depth	Queue depth for dumping data. Value range: [0, 8]
endumptype	Dumping data type.

【Note】

None.

【Related Data Type and Interface】

- CVI_VI_SetPipeDumpAttr
- CVI_VI_GetPipeDumpAttr

4.4.35 VI_CHN_ATTR_S

【Description】

Define VI channel properties.

【Syntax】

```
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;
```

【Member】

Member	Description
stSize	Target image size. The minimum and maximum width and height of the target image: Height in online mode [VI_PHYCHN_ONLINE_MIN_HEIGHT, VI_PHYCHN_ONLINE_MAX_HEIGHT] Height in offline mode [VI_PHYCHN_OFFLINE_MIN_HEIGHT, VI_PHYCHN_OFFLINE_MAX_HEIGHT] Width in online mode [VI_PHYCHN_ONLINE_MIN_WIDTH, VI_PHYCHN_ONLINE_MAX_WIDTH] Width in offline mode [VI_PHYCHN_OFFLINE_MIN_WIDTH, VI_PHYCHN_OFFLINE_MAX_WIDTH]
enPixelFormat	Target image pixel format.
enDynamicRange	The dynamic range of the target image.
enVideoFormat	Video data format of the target image.
enCompressMode	Compression format of the target image.
bMirror	Mirror enable switch. CVI_FALSE: disable; CVI_TRUE: enable
bFlip	Flip enable switch CVI_FALSE: disable; CVI_TRUE: enable
u32Depth	User polling depth for getting images.
stFrameRate	Frame rate control. Source frame rate range: (0, 240], and -1. When the source frame rate is -1, the target frame rate must be set to -1 (no frame rate control), and in other cases, the target frame rate cannot be greater than the source frame rate.
bLVDSflow	Whether to switch to LVDS mode.
u8TotalChnNum	Total number of channels.

【Note】

- Currently, VI does not support frame rate control

【Related Data Type and Interface】

- CVI_VI_SetChnAttr
- CVI_VI_GetChnAttr

4.4.36 VI_CROP_INFO_S

【Description】

Define the VI CROP information structure.

【Syntax】

```
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;
```

【Member】

Member	Description
bEnable	CROP enable switch
enCropCoordinate	CROP starting point coordinate mode.
stCropRect	Rectangular area of CROP.

【Note】

None.

【Related Data Type and Interface】

None.

4.4.37 VI_DEV_TIMING_ATTR_S

【Description】

User defined timing properties.

【Syntax】

```
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;
```

【Member】

Member	Description
bEnable	User defined timing enable switch.
s32FrmRate	The frame rate of the user-defined timing.

【Note】

- When the user-defined timing is enabled, if the frame rate set by the user exceeds the maximum frame rate of the device, the system will automatically take the maximum frame rate of the device as the effective value.

【Related Data Type and Interface】

- CVI_VI_SetDevTimingAttr

4.4.38 VI_PIPE_STATUS_S

【Description】

Define the status information of VI pipe.

【Syntax】

```
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;
```

【Member】

Member	Description
bEnable	Whether the current pipe is enabled.
u32IntCnt	Interrupt count.
u32FrameRate	Real time frame rate of VI pipe.
u32LostFrame	Lost frame count.
u32VbFail	VB application failure count.
stSize	Current pipe output size

【Note】

None.

【Related Data Type and Interface】

None.

4.4.39 VI_CHN_STATUS_S

【Description】

Define the status information of VI channel.

【Syntax】

```
typedef struct _VI_CHN_STATUS_S {
    CVI_BOOL bEnable; /* RO;Whether this channel is enabled */
    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;chn output size */
} VI_CHN_STATUS_S;
```

【Member】

Member	Description
bEnable	Whether the current PIPE is enabled.
u32FrameRate	Real time frame rate of channel.
u32LostFrame	Lost frame count.
u32VbFail	VB application failure count.
stSize	Current channel output size.

【Note】

None.

【Related Data Type and Interface】

None.

4.4.40 VI_PIPE_FRAME_SOURCE_E

【Description】

Define the source type of VI PIPE data.

【Syntax】

```
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;
```

【Member】

Member	Description
vi_pipe_frame_source_dev	The data comes from the device.
vi_pipe_frame_source_user_fe	The data comes from the user' s data sent in from FE
vi_pipe_frame_source_user_be	The data comes from the user' s data sent in from BE

【Note】

- Do not support data from BE.

【Related Data Type and Interface】

- CVI_VI_SetPipeFrameSource

4.4.41 VI_LDC_ATTR_S

【Description】

Define VI lens distortion correction structure

【Syntax】

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

【Member】

Member	Description
benable	LDC enable
stattr	LDC properties

【Note】

None.

【Related Data Type and Interface】

- LDC_ATTR_S
- CVI_VI_GetChnLDCAttr
- CVI_VI_SetChnLDCAttr

4.5 Error Codes

The error codes of video input API are defined as follows:

Table 4.3: Video input API error code

Error code	Macro definition	Description
0xC00E8001	CVI_ERR_VI_INVALID_DEVID	Invalid device number
0xC00E8002	CVI_ERR_VI_INVALID_CHNID	Invalid channel number
0xC00E8003	CVI_ERR_VI_INVALID_PARA	Invalid parameter setting
0xC00E8004	CVI_ERR_VI_PIPE_EXIST	PIPE already exists
0xC00E8005	CVI_ERR_VI_PIPE_UNEXIST	PIPE does not exist
0xC00E8006	CVI_ERR_VI_INVALID_NULL_PTR	Null pointer error
0xC00E8007	CVI_ERR_VI_FAILED_NOTCONFIG	Device or channel properties are not configured
0xC00E8008	CVI_ERR_VI_NOT_SUPPORT	Operation not supported
0xC00E8009	CVI_ERR_VI_NOT_PERM	Operation forbidden
0xC00E800A	CVI_ERR_VI_INVALID_PIPEID	Invalid PIPE ID
0xC00E800C	CVI_ERR_VI_NOMEM	Memory allocation error
0xC00E800E	CVI_ERR_VI_BUF_EMPTY	The cache is empty
0xC00E800F	CVI_ERR_VI_BUF_FULL	The cache is full
0xC00E8010	CVI_ERR_VI_SYS_NOTREADY	The system is not initialized
0xC00E8012	CVI_ERR_VI_BUSY	The system is busy
0xC00E8040	CVI_ERR_VI_FAILED_NOTENABLE	The device has not been started
0xC00E8041	CVI_ERR_VI_FAILED_NOTDISABLE	The device has not been shut down
0xC00E8042	CVI_ERR_VI_FAILED_CHNOTDISABLE	The channel has not been closed
0xC00E8043	CVI_ERR_VI_CFG_TIMEOUT	Configuration timeout
0xC00E8044	CVI_ERR_VI_NORM_UNMATCH	No matching number found
0xC00E8045	CVI_ERR_VI_INVALID_WAYID	Invalid channel ID
0xC00E8046	CVI_ERR_VI_INVALID_PHYCHNID	Invalid physical channel ID
0xC00E8047	CVI_ERR_VI_FAILED_NOTBIND	Channel unbound
0xC00E8048	CVI_ERR_VI_FAILED_BINDED	Channel binding failed

VIDEO OUTPUT

5.1 Function overview

CV180x does not support this function.

5.1.1 Objective

VO (video output) module reads video and graphics data from memory, and outputs video and graphics through corresponding display device.

5.1.2 Definitions and Abbreviations

DUD (Device Ultra-High Definition)

DHD (Device High Definition)

DSD (Device Standard Definition)

VUD (Video Layer of UHD)

VHD (Video Layer of HD)

VSD (Video Layer of SD)

G0 (Graphics Layer0)

5.2 Design Overview

5.2.1 System Architecture

- Display Device

In SDK, UHD, HD and SD display devices are labeled as DUDx (3840x2160), DHDx (1920x1080) and DSDx (720x576) respectively to indicate the capability of the device.

X is the index number, which starts from 0 to distinguish the display devices.

For example, channel 0 HD device is labeled as DHD0, and channel 1 UHD display device is labeled as DUD1. UHD, HD and SD display devices can be referred to as UD, HD and SD devices respectively.

CV181x has a high definition display device DHD0

- Video Layer

For the corresponding video layer fixed on each display device, the SDK also adopts the way of corresponding display device, marked by VUDx, VHDx and VSDx.

The chip supports display device, video layer and graphics layer. Please refer to table 5.1.

Refer to Table 5.2 for functional comparison of chip devices.

The function comparison of video layer is shown in table 5.3.

The maximum timing supported by the video output interface on the device is shown in table 5.4.

The actual display resolution of video layer and display device depends on the specific output interface.

Table 5.1: supports display device, video layer and graphics layer

Chip	Subitem	HD display device DHD0	
CV183x	Name	Video layer VHD0	Graphic layer G0
	Description	Supports a maximum of 1 screen	
	Output interface	BT.112 0/BT.656, MIPI Tx, LVDS, I80	
CV182x/CV181x	Name	video layer VHD0	Graphic layer G0
	Description	Supports a maximum of 1 screen	
	Output interface	BT.112 0/BT.656, MIPI Tx, LVDS, I80, RGB	

Table 5.2: display device functions

Chip		Maximum output time sequence	Output interface	Overlay display
CV183x	DH D0	1080p@60	BT0112 0/BT.656/MIPI Tx/LCD	Not support
CV182xCV181x	DH D0	1080p@60 720p@60	BT0112 0/BT.656/MIPI Tx/LVDS/RGB	Not support

Table 5.3: video layer functions

Chip	Dynamic binding capability	Zoom capability	Number of channels supported		CSC
			SINGLE mode	MULTI mode	
CV183x	Not support	Not support	1	1	support
CV182x CV181x	Not support	Not support	1	1	support

Table 5.4: the maximum output timing of the device interface

Chip		Maximum output time sequence
	MIPI Tx	1080P@60
	BT.1120	1080P@60
	BT.656	1080P@60
	LCD	720P@60
CV182x CV181x	MIPI Tx	720P@60
	BT.1120	720P@60
	BT.656	M720P@60
	LCD	M720P@60

- Channel

The SDK attributes the channel to the video layer management.

A video layer can display multiple videos.

Each video display area is called a channel, and the video is restricted within the channel and the channel is restricted within the video layer.

For a video layer, the channels above are independent.

At the same time, the channels on different video layers are also independent.

- Resolution

There are two main concepts of resolution:

Device resolution is determined by device timing and determines the number of effective pixels output by the device.

Display resolution refers to the effective display area on the display device, which is determined by the stDispRect member in the video layer properties.

The display resolution must be less than or equal to the device resolution.

- Graphic Layer Binding

Graphics layer binding means that the chip supports binding specific graphics layer to video layer.

CV181x supports one graphics layer (G0 is bound to DHD0, that is, G0 can only overlay VHD0).

- Rotation

VO supports rotation of input images.

You can also do rotation first, and do subsequent processing in the access channel.

Usually, it is applied to some vertical screens, and the rotation is set to 90 or 270 degrees to reach full screen and maintain image scale.

- Input and Output Data Formats

The input data formats supported by the VO modules in chip are shown in table 5.5 below.

The PIXEL FORMAT and VIDEO FORMAT in the table respectively list the input pixel format and video format supported by the chip.

The data format in the table can refer to the chapter “2 system control” .

Table 5.5: input data formats supported by chip

Chip		Input
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 Reference

Video output (VO) realizes video output device, video channel and other functions..

The APIs supported by this function module are introduced from the aspects of device, video layer, channel and so on.

Device related APIs are as follows:

- *CVI_VO_Enable*: Enable video output device.
- *CVI_VO_Disable*: Disable video output device.
- *CVI_VO_SetPubAttr*: Set the properties of the video output device.
- *CVI_VO_GetPubAttr*: Get the properties of the video output device.
- *CVI_VO_CloseFd*: Close the file handle of the video output device.

Video layer related APIs are as follows:

- *CVI_VO_EnableVideoLayer*: Enable video layer
- *CVI_VO_DisableVideoLayer*: Disable video layer
- *CVI_VO_SetVideoLayerAttr*: Set the properties of the video layer device
- *CVI_VO_GetVideoLayerAttr*: Get the properties of the video layer device

Channel related APIs are as follows:

- *CVI_VO_EnableChn*: Enable video output channel.

- *CVI_VO_DisableChn*: Disable video output channel.
- *CVI_VO_SetChnAttr*: Set the properties of the video output channel.
- *CVI_VO_GetChnAttr*: Get the properties of the video output channel.
- *CVI_VO_ShowChn*: Display the specified video output channel.
- *CVI_VO_HideChn*: Hide the specified video output channel.
- *CVI_VO_SetChnRotation*: Set the rotation property of the video output channel.
- *CVI_VO_GetChnRotation*: Get the rotation property of the video output channel.
- *CVI_VO_PauseChn*: Pause the output of the specified VO channel.
- *CVI_VO_ResumeChn*: Resume the output of the specified VO channel.

5.3.1 CVI_VO_Enable

【Description】

Enable video output device.

【Syntax】

```
CVI_S32 CVI_VO_Enable(Vo_DEV VoDev);
```

【Parameter】

Parameter	Description	Input/Output
VoDev	Video output device number. Value range: [0, VO_MAX_DEV_NUM]	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: *cvi_vo.h*, *cvi_comm_vo.h*
- Library files: *libvpu.a*

【Note】

- The device must be enabled before using the video output function.
- Before calling the device enable, you must configure the common properties of the device.
Otherwise, an error message is displayed indicating that the device is not configured.
- If you want to change the VO timing configuration,
you need to call the *CVI_VO_Disable* interface to forcibly disable the VO hardware before enabling it to avoid undesirable transients in the process of changing the timing.

【Example】

```
CVI_S32 s32Ret = CVI_SUCCESS;

s32Ret = CVI_VO_SetPubAttr(VoDev, pstPubAttr);
if (s32Ret != CVI_SUCCESS) {
```

(continues on next page)

(continued from previous page)

```

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;
}

```

【Related Topic】

- [CVI_VO_Disable](#)

5.3.2 CVI_VO_Disable**【Description】**

Disable video output device.

【Syntax】

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

【Parameter】

Parameter	Description	Input/Output
VoDev	Video output device number. Value range: [0, VO_MAX_DEV_NUM]	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: `cvi_vo.h`, `cvi_comm_vo.h`
- library file: `libvpu.a`

【Note】

- The video layer on the device must be forbidden before the device is forbidden.
- You can set device attributes only when the VO device is disabled.

【Example】

None.

【Related Topic】

- [CVI_VO_Enable](#)

5.3.3 CVI_VO_SetPubAttr

【Description】

Set the properties of the video output device.

【Syntax】

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

【Parameter】

Parameter	Description	Input/Output
VoDev	Video output device number. Value range: [0, VO_MAX_DEV_NUM).	Input
pstPubAttr	Pointer to the common attribute structure of the video output device.	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: cvi_vo.h, cvi_comm_vo.h
- Library files: libvpu.a

【Note】

- The property of video output device is static and must be configured before CVI_VO_Enable.
- See VO_DEV for instructions on how to use DEV.
- Please refer to the VO_PUB_ATTR_S chapter for the description of video output device properties.

【Example】

None.

【Related Topic】

- [CVI_VO_GetPubAttr](#)

5.3.4 CVI_VO_GetPubAttr

【Description】

Get the related properties of the video output device.

【Syntax】

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

【Parameter】

Parameter	Description	Input/Output
VoDev	Video output device number. Value range: [0, VO_MAX_DEV_NUM).	Input
pstPubAttr	Pointer to the common attribute structure of the video output device.	Output

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: `cvi_vo.h`, `cvi_comm_vo.h`
- Library files: `libvpu.a`

【Note】

- Obtain the attributes before setting the public attributes of the device so that you can set only the configuration items that you want to change.

【Example】

None.

【Related Topic】

- [*CVI_VO_SetPubAttr*](#)

5.3.5 CVI_VO_EnableVideoLayer

【Description】

Enable the video layer.

【Syntax】

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

【Parameter】

Parameter	Description	Input/Output
VoLayer	Video output video layer ID. Value range: [0, VO_MAX_LAYER_NUM).	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: `cvi_vo.h`, `cvi_comm_vo.h`
- Library files: `libvpu.a`

【Note】

- Before enabling the video layer, the device bound to the video layer must be enabled.

【Example】

None.

【Related Topic】

- [*CVI_VO_DisableVideoLayer*](#)

5.3.6 CVI_VO_DisableVideoLayer

【Description】

Disable the video layer.

【Syntax】

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

【Parameter】

Parameter	Description	Input/Output
VoLayer	Video output video layer ID. Value range: [0, VO_MAX_LAYER_NUM).	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: cvi_vo.h, cvi_comm_vo.h
- Library files: libvpu.a

【Note】

- Before the video layer is disabled, the channel on it must be disabled first.

【Example】

None.

【Related Topic】

- [CVI_VO_EnableVideoLayer](#)

5.3.7 CVI_VO_SetVideoLayerAttr

【Description】

Configure properties of the video layer.

【Syntax】

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

【Parameter】

Parameter	Description	Input/Output
VoLayer	Video output video layer ID. Value range: [0, VO_MAX_LAYER_NUM).	Input
pstLayerAttr	Pointer to the video layer properties structure.	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: cvi_vo.h, cvi_comm_vo.h
- Library files: libvpu.a

【Note】

None.

【Example】

None.

【Related Topic】

- [CVI_VO_GetVideoLayerAttr](#)

5.3.8 CVI_VO_GetVideoLayerAttr

【Description】

Get the related properties of the video layer.

【Syntax】

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

【Parameter】

Parameter	Description	Input/Output
VoLayer	Video output video layer ID. Value range: [0, VO_MAX_LAYER_NUM).	Input
pstLayerAttr	Pointer to the video layer properties structure.	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: cvi_vo.h, cvi_comm_vo.h
- Library files: libvpu.a

【Note】

None.

【Example】

None.

【Related Topic】

- [CVI_VO_SetVideoLayerAttr](#)

5.3.9 CVI_VO_EnableChn

【Description】

Enables the specified video output channel.

【Syntax】

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

【Parameter】

Parameter	Description	Input/Output
VoLayer	Video output video layer ID. Value range: [0, VO_MAX_LAYER_NUM).	Input
VoChn	Video output channel ID. Value range: [0, VO_MAX_CHN_NUM].	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: cvi_vo.h, cvi_comm_vo.h
- Library files: libvpu.a

【Note】

- The video layer on the corresponding device must be enabled before calling.
- The channel must be configured before the channel is enabled. Otherwise an error message indicating that the channel is not configured is displayed.

【Example】

None.

【Related Topic】

- [CVI_VO_DisableChn](#)

5.3.10 CVI_VO_DisableChn

【Description】

Disable the video output channel.

【Syntax】

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

【Parameter】

Parameter	Description	Input/Output
VoLayer	Video output video layer ID. Value range: [0, VO_MAX_LAYER_NUM).	Input
VoChn	Video output channel ID. Value range: [0, VO_MAX_CHN_NUM].	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: `cvi_vo.h`, `cvi_comm_vo.h`
- Library files: `libvpu.a`

【Note】

None.

【Example】

None.

【Related Topic】

- [CVI_VO_EnableChn](#)

5.3.11 CVI_VO_SetChnAttr

【Description】

Configure the properties of the video output channel.

【Syntax】

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

【Parameter】

Parameter	Description	Input/Output
VoLayer	Video output video layer ID. Value range: [0, VO_MAX_LAYER_NUM).	Input
VoChn	Video output channel ID. Value range: [0, VO_MAX_CHN_NUM].	Input
pstChnAttr	Pointer to the properties structure of the video output channel.	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: `cvi_vo.h`, `cvi_comm_vo.h`
- Library files: `libvpu.a`

【Note】

- The channel display area should be smaller than the `stImageSize` set in the video layer properties.

【Example】

None.

【Related Topic】

- [CVI_VO_GetChnAttr](#)

5.3.12 CVI_VO_GetChnAttr**【Description】**

Get the related properties of video output channel.

【Syntax】

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

【Parameter】

Parameter	Description	Input/Output
VoLayer	Video output video layer ID. Value range: [0, VO_MAX_LAYER_NUM).	Input
VoChn	Video output channel ID. Value range: [0, VO_MAX_CHN_NUM].	Input
pstChnAttr	Pointer to the properties structure of the video output channel.	Output

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: cvi_vo.h, cvi_comm_vo.h
- Library files: libvpu.a

【Note】

None.

【Example】

None.

【Related Topic】

- [CVI_VO_SetChnAttr](#)

5.3.13 CVI_VO_ShowChn**【Description】**

Displays the specified video output channel.

【Syntax】

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

【Parameter】

Parameter	Description	Input/Output
VoLayer	Video output video layer ID. Value range: [0, VO_MAX_LAYER_NUM).	Input
VoChn	Video output channel ID. Value range: [0, VO_MAX_CHN_NUM].	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: cvi_vo.h, cvi_comm_vo.h
- Library files: libvpu.a

【Note】

- Before calling, the video channel on the corresponding device must be enabled.
- By default, it is in the display state.

【Example】

None.

【Related Topic】

- [CVI_VO_HideChn](#)

5.3.14 CVI_VO_HideChn

【Description】

Hides the specified video output channel.

【Syntax】

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

【Parameter】

Parameter	Description	Input/Output
VoLayer	Video output video layer ID. Value range: [0, VO_MAX_LAYER_NUM).	Input
VoChn	Video output channel ID. Value range: [0, VO_MAX_CHN_NUM].	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: cvi_vo.h, cvi_comm_vo.h
- Library files: libvpu.a

【Note】

None.

【Example】

None.

【Related Topic】

- [CVI_VO_ShowChn](#)

5.3.15 CVI_VO_SetChnRotation

【Description】

Set the properties of VO channel rotation.

【Syntax】

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

【Parameter】

Parameter	Description	Input/Output
VoLayer	Video output video layer ID. Value range: [0, VO_MAX_LAYER_NUM).	Input
VoChn	Video output channel number. Value range: [0, VO_MAX_CHN_NUM].	Input
enRotation	Rotation properties. See ROTATION_E for details.	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: `cvi_vo.h`, `cvi_comm_vo.h`
- Library files: `libvpu.a`

【Note】

- Before using this interface, you need to call `CVI_VO_SetChnAttr()`; otherwise, the system displays a failure message.
- CV182x/CV181x supports rotation only in NV12, NV21, and YUV400 formats.
- After setting, rotation will apply to the video image entering the channel.

Pay attention to setting the attribute size of the video layer and the video channel.

For example, the size of the channel is set to 1920x1080 and rotated 90 degrees, so the image input to the channel should be 1080x1920.

【Example】

None.

【Related Topic】

None.

5.3.16 CVI_VO_GetChnRotation

【Description】

Get the properties of VO channel rotation.

【Syntax】

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

【Parameter】

Parameter	Description	Input/Output
VoLayer	Video output video layer ID. Value range: [0, VO_MAX_LAYER_NUM).	Input
VoChn	Video output channel number. Value range: [0, VO_MAX_CHN_NUM].	Input
penRotation	Rotation property pointer. See ROTATION_E for details.	Output

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: cvi_vo.h, cvi_comm_vo.h
- Library files: libvpu.a

【Note】

- Before using this interface, you need to call CVI_VO_SetChnAttr(); otherwise, the system displays a failure message.

【Example】

None.

【Related Topic】

- [CVI_VO_SetChnRotation](#)

5.3.17 CVI_VO_PauseChn

【Description】

Pause the output of the specified VO channel.

【Syntax】

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

【Parameter】

Parameter	Description	Input/Output
VoLayer	Video output video layer ID. Value range: [0, VO_MAX_LAYER_NUM).	Input
VoChn	Video output channel number. Value range: [0, VO_MAX_CHN_NUM].	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: cvi_vo.h, cvi_comm_vo.h
- Library files: libvpu.a

【Note】

- The video channel on the corresponding device must be enabled before calling.
- It is allowed to pause the same channel repeatedly without returning failure.

【Example】

None.

【Related Topic】

- [CVI_VO_ResumeChn](#)

5.3.18 CVI_VO_ResumeChn

【Description】

Resume the output of the specified VO channel.

【Syntax】

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

【Parameter】

Parameter	Description	Input/Output
VoLayer	Video output video layer ID. Value range: [0, VO_MAX_LAYER_NUM).	Input
VoChn	Video output channel number. Value range: [0, VO_MAX_CHN_NUM].	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: cvi_vo.h, cvi_comm_vo.h
- Library files: libvpu.a

【Note】

- The video channel on the corresponding device must be enabled before calling.
- It is allowed to resume the same channel repeatedly without returning failure.

【Example】

None.

【Related Topic】

- [CVI_VO_PauseChn](#)

5.3.19 CVI_VO_CloseFd

【Description】

Close the file handle of the video output device.

【Syntax】

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

【Parameter】

None.

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: `cvi_vo.h`, `cvi_comm_vo.h`
- Library files: `libvpu.a`

【Note】

After this interface is invoked, other MMF interfaces on the VO become invalid.

【Example】

None.

【Related Topic】

None.

5.4 Data Types

5.4.1 VO_DEV

【Description】

Define the device number.

【Syntax】

```
typedef CVI_U32 VO_DEV;
```

【Member】

- – Chip
 - escription
- – CV183x
 - There is only one video output device.
 - 0: DHD0
 - Support LCD, MIPI TX, BT.656, BT.1120
- – CV182xCV181x
 - There is only one video output device.
 - 0: DHD0
 - Support LCD, MIPI TX, BT.656, BT.1120, RGB, 180

【Note】

None.

【Related Data Type and Interface】

None.

5.4.2 VO_LAYER

【Description】

Define the video layer number.

【Syntax】

```
typedef CVI_U32 VO_LAYER;
```

【Member】

- – Chip
 - escription
- – CV183x
 - There is only one video output layer.
 - 0: VHD0
- – CV182xCV181x
 - There is only one video output layer.
 - 0: VHD0

【Note】

None.

【Related Data Type and Interface】

None.

5.4.3 VO_INTF_TYPE

【Description】

Define the interface number of the video output device.

【Syntax】

```
#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;
```

【Note】

None.

【Related Data Type and Interface】

None.

5.4.4 VO_INTF_SYNC_E

【Description】

Define the standard timing of video output devices.

【Syntax】

```
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,
    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,
```

(continues on next page)

(continued from previous page)

```

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;

```

【Note】

None.

【Related Data Type and Interface】

None.

5.4.5 VO_SYNC_INFO_S

【Description】

Define the sequential structure of the device.

【Syntax】

```

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;

```

【Member】

Member	Description
bSynm	Synchronous signal mode, 0: embedded sync, 1: separate sync。
bIop	0: interlaced, 1: progressive。
u16FrameRate	Number of updates per second
u16Vact	Vertical image rows
u16Vbb	Vertical back porch rows
u16Vfb	Vertical front porch rows
u16Hact	Phase number of horizontal image
u16Hbb	Horizontal back porch phase number
u16Hfb	Horizontal front porch phase number
u16Hpw	Number of horizontal synchronous phases
u16Vpw	Vertical synchronous rows
bIdv	Is Data valid reversed
bIhs	Is horizontal synchronization reversed
bIvs	Is vertical synchronization reversed

【Note】

None.

【Related Data Type and Interface】

None.

5.4.6 VO_PUB_ATTR_S

【Description】

Define output device structure.

【Syntax】

```
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;
```

【Member】

Member	Description
u32BgColor	Background color. In RGBAAA format, bit[9:0] is B, bit[19:10] is G, and bit[29:20] is R.
enIntfType	The interface of the output device.
enIntfSync	Standard timing of output devices
stSyncInfo	The custom timing of the output device, only works when enIntf-Sync is VO_OUTPUT_USER .
sti80Cfg	Interface properties of the output device when the interface is I80
stLvdsAttr	Interface properties when the output device interface is LCD

【Note】

None.

【Related Data Type and Interface】

None.

5.4.7 VO_VIDEO_LAYER_ATTR_S

【Description】

Define the video layer structure.

【Syntax】

```
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;
```

【Member】

Member	Description
stDispRect	The display range of the video layer should be smaller than that of the device.
stImageSize	Image size. If scaling is not supported, the image size should be equal to stDispRect
u32DispFrmRt	Display the number of updates
enPixFormat	Image format of video layer

【Note】

None.

【Related Data Type and Interface】

None.

5.4.8 VO_CHN_ATTR_S

【Description】

Define output channel structure.

【Syntax】

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

【Member】

Member	Description
u32Priority	In case of multi-channel, the one with high priority (relatively small) will be on the channel.
stRect	The display area of the channel.

【Note】

None.

【Related Data Type and Interface】

None.

5.5 Error Codes

Error Code	Macro Definition	Description
0xc00d8001:	CVI_ERR_VO_INVALID_DEVID	Description
0xc00d8001:	CVI_ERR_VO_INVALID_DEVID	Device ID does not match
0xc00d8002:	CVI_ERR_VO_INVALID_CHNID	Channel ID does not match
0xc00d8003:	CVI_ERR_SYS_ILLEGAL_PARAM	Invalid parameter setting
0xc00d8006:	CVI_ERR_SYS_NULL_PTR	Null pointer
0xc00d8008:	CVI_ERR_SYS_NOT_SUPPORT	Features not supported
0xc00d8009:	CVI_ERR_SYS_NOT_PERM	Operation not allowed
0xc00d800c:	CVI_ERR_SYS_NOMEM	Memory allocation failure, such as insufficient system memory
0xc00d8010:	CVI_ERR_SYS_NOTREADY	The system control property is not configured
0xc00d8012:	CVI_ERR_VO_BUSY	The system is busy
0xc00d8040:	CVI_ERR_VO_DEV_NOT_CONFIG	Device not configured
0xc00d8041:	CVI_ERR_VO_DEV_NOT_ENABLE	Device not enabled
0xc00d8042:	CVI_ERR_VO_DEV_HAS_ENABLE	Device enabled
0xc00d8045:	CVI_ERR_VO_VIDEO_NOT_ENABLE	The video layer is not enabled
0xc00d8046:	CVI_ERR_VO_VIDEO_NOT_DISABLE	Video layer not disabled
0xc00d8047:	CVI_ERR_VO_VIDEO_NOT_CONFIG	Video layer not configured
0xc00d8048:	CVI_ERR_VO_CHN_NOT_DISABLE	Video channel not prohibited
0xc00d8049:	CVI_ERR_VO_CHN_NOT_ENABLE	Video channel not enabled
0xc00d804a:	CVI_ERR_VO_CHN_NOT_CONFIG	Video channel not configured
0xc00d804e:	CVI_ERR_VO_WAIT_TIMEOUT	Waiting timeout
0xc00d804f:	CVI_ERR_VO_INVALID_VFRAME	Invalid video frame
0xc00d8050:	CVI_ERR_VO_INVALID_RECT_PARAM	Invalid matrix parameter
0xc00d8065:	CVI_ERR_VO_CHN_AREA_OVERLAP	Video channel region overlap
0xc00d8066:	CVI_ERR_VO_INVALID_LAYERID	Invalid video layer ID

VIDEO PROCESSING SUBSYSTEM

6.1 Function Overview

6.1.1 Objective

VPSS (video process sub system) is a video processing subsystem, which supports specific image processing functions, including CROP, Scale, pixel format conversion, Mirror/Flip, fixed angle rotation, fisheye correction, Overlay/Overlayex, etc.

6.1.2 Definitions and Abbreviations

VPSS: video process sub system

Grp: Group, which represents a group of VPSS in VPSS

Chn: channel, channels of VPSS group

VB: Video Buffer, the video memory buffer pool

6.2 Design Overview

6.2.1 System Architecture

The following are the basic concepts of VPSS.

- GROUP

VPSS provides the concept of GROUP for users.

VPSS_MAX_GRP_NUM for indicates the maximum number of GROUP.

Each group time-multiplexes the VPSS hardware, which sequentially processes tasks submitted by each group.

- CHANNEL

Channel of VPSS group.

VPSS hardware provides multiple physical channels, and each channel has the functions such as scaling and clipping.

It binds to a physical channel and takes the physical channel output as its input, scaling the image to the user-set target resolution and outputting it.

- CROP

There are two kinds of clipping: Group clipping and physical channel clipping.

- Group clipping. The VPSS crops the input image.

- Physical channel clipping. The output image of each physical channel is cropped by VPSS.
- Pixel format conversion

Support input and output image data format conversion.

- Support mutual conversion of YUV420 planar, YUV422 planar, RGB planar, RGB packed and BGR packed

- Scale

Zoom in and out of an image.

Physical channel supports at least 32 times smaller and at most 32 times larger horizontally and vertically.

- Mirror/Flip

Mirror is the horizontal mirror; Flip means upside down. A 180-degree rotation can be achieved using Mirror + Flip.

- Overlay/Overlayex

Video overlay area.

Call RGN to overlay bitmap on output image of VPSS channel.

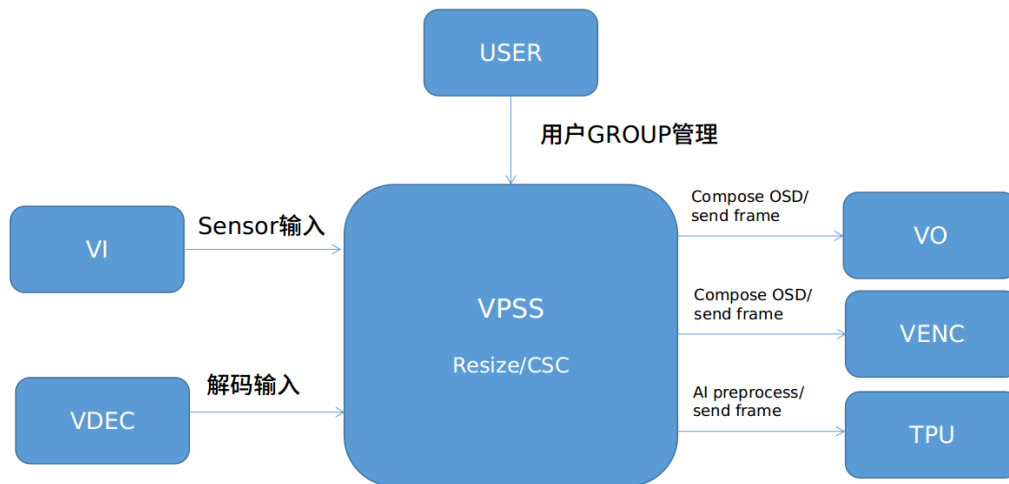
It supports bitmap format such as ARGB4444, ARGB1555, ARGB8888, 256 LUT(ARGB4444), and Font-based.

- Fixed angle rotation

Call GDC to process the output image of VPSS channel.

It supports 0 degrees, 90 degrees, 180 degrees and 270 degrees fixed angle rotation.

The location of Vpss module in the system is shown in the figure below.



By calling the binding interface provided by the SYS module, the VPSS module can be bound to input modules such as VI/VDEC to process decoding and sensor data through VPSS, enabling functions such as image scaling and color space conversion.

At the same time, through the binding interface, the images processed by the VPSS module can be synthesized by OSD and sent to the VO / VENC module.

It also supports some simple AI pre-processing, and then the processed images can be sent to the TPU for AI computation.

Table 6- 1 VPSS Functional limitations

Working Mode	Function		
	Group clipping	Zoom	Channel clipping
VI_OFFLINE_VPSS_OFFLINE	support	support	support
VI_OFFLINE_VPSS_ONLINE	Not support	support	support
VI_ONLINE_VPSS_OFFLINE	support	support	support
VI_ONLINE_VPSS_ONLINE	Not support	support	support

Note: When VPSS is ONLINE, it can only receive data from two front-end sensors through two groups, and cannot serve other different sources.

6.2.2 Note

The data processing flow chart of VPSS is shown in Figure 6-1 below.

There are two groups of input.

When a single input is used, there can be four groups of channel outputs (CV180x is three groups);

When two groups of input are used, the first group of input has only one group of channel for output, and the second group of input has only three groups of channels for output (CV180x is two groups).

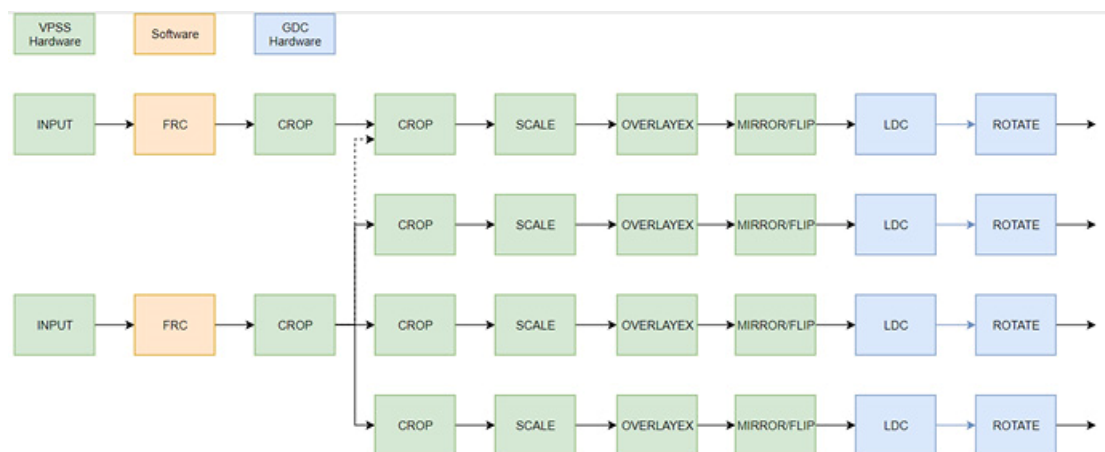


Fig. 6.1: Figure 6- 1 Data flow chart of CV181x

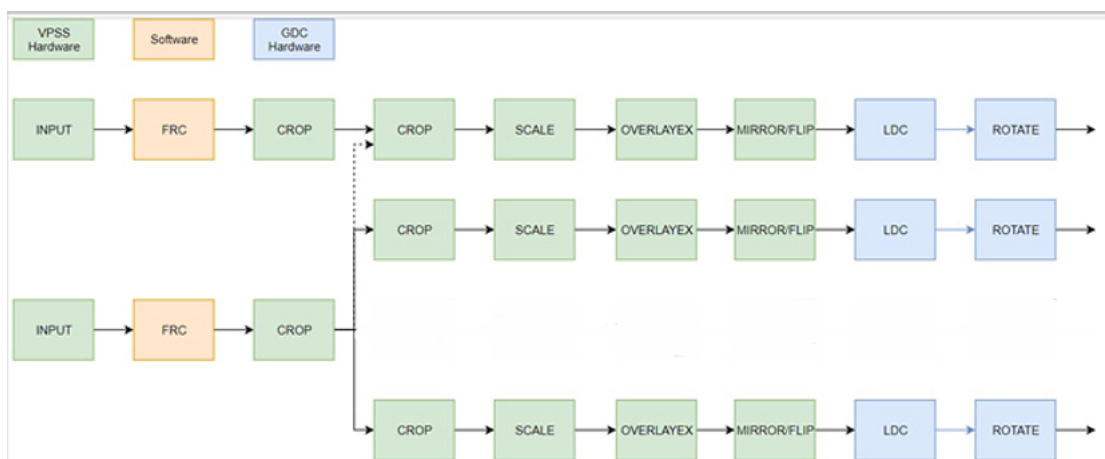


Fig. 6.2: Figure 6- 2 Data flow chart of CV180x

6.3 API Reference

The function module provides the following APIs for users.

- *CVI_VPSS_CreateGrp*: Create a VPSS GROUP.
- *CVI_VPSS_DestroyGrp*: Destroy a VPSS GROUP.
- *CVI_VPSS_GetGrpAttr*: Get properties of VPSS GROUP.
- *CVI_VPSS_SetGrpAttr*: Set properties of VPSS GROUP.
- *CVI_VPSS_StartGrp*: Enable VPSS GROUP.
- *CVI_VPSS_StopGrp*: Disable VPSS GROUP.
- *CVI_VPSS_ResetGrp*: Reset a VPSS GROUP.
- *CVI_VPSS_GetGrpProcAmpCtrl*: Get a description of the color control function of VPSS GROUP.
- *CVI_VPSS_GetGrpProcAmp*: Get the color control property of a VPSS GROUP.
- *CVI_VPSS_SetGrpProcAmp*: Set the color control property of a VPSS GROUP.
- *CVI_VPSS_SetGrpParamfromBin*: Set the property of a VPSS GROUP according to Bin.
- *CVI_VPSS_GetChnAttr*: Get VPSS channel properties.
- *CVI_VPSS_SetChnAttr*: Set VPSS channel properties.
- *CVI_VPSS_EnableChn*: Enable VPSS channel.
- *CVI_VPSS_DisableChn*: Disable VPSS channel.
- *CVI_VPSS_SetGrpCrop*: Set the VPSS GROUP CROP function properties.
- *CVI_VPSS_GetGrpCrop*: Get the VPSS GROUP CROP function properties.
- *CVI_VPSS_SendFrame*: The user sends data to VPSS.
- *CVI_VPSS_GetChnFrame*: The user gets a frame of processed images from the channel.
- *CVI_VPSS_SendChnFrame*: The user send data to the specifies channel of VPSS.
- *CVI_VPSS_ReleaseChnFrame*:The user releases a frame of image processed by channel.
- *CVI_VPSS_GetGrpFrame*: The user gets the group image from VPSS.
- *CVI_VPSS_ReleaseGrpFrame*: The user releases the group image
- *CVI_VPSS_SetChnCrop*: Set the VPSS channel clipping feature properties.
- *CVI_VPSS_GetChnCrop*: Get the VPSS channel clipping feature properties.
- *CVI_VPSS_SetChnRotation*: Set the properties of VPSS channel rotation.
- *CVI_VPSS_GetChnRotation*: Get the VPSS channel image rotation attribute.
- *CVI_VPSS_GetChnFd*: Get the device file handle corresponding to the VPSS channel.
- *CVI_VPSS_CloseFd*: Close the file handle for the VPSS device channel.
- *CVI_VPSS_SetChnBufWrapAttr*: Set the low-latency scrambling property.
- *CVI_VPSS_GetChnBufWrapAttr*: Get the low-latency scrambling property.
- *CVI_VPSS_GetWrapBufferSize*: Get the required low-latency scrambling cache size.

6.3.1 CVI_VPSS_CreateGrp

【Description】

Create a VPSS GROUP.

【Syntax】

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

【Parameter】

Parameter	Description	Input/Output
VpssGrp	VPSS GROUP number. Value Range: [0, VPSS_MAX_GRP_NUM).	Input
pstGrpAttr	VPSS GROUP property pointer.	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: cvi_comm_vpss.h, cvi_vpss.h
- Library files: libvpu.a

【Note】

- Create VPSS_GRP_ATTR_S before using this interface.

【Example】

```
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;
```

(continues on next page)

(continued from previous page)

```

/*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, &stVpssChnAttr[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);
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;
}

```

(continues on next page)

(continued from previous page)

```

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

```

【Related Topic】

- [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**【Description】**

Destroy a VPSS GROUP.

【Syntax】

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

【Parameter】

Parameter	Description	Input/Output
VpssGrp	VPSS GROUP number. Value Range: [0, VPSS_MAX_GRP_NUM).	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: `cvi_comm_vpss.h`, `cvi_vpss.h`
- Library files: `libvpu.a`

【Note】

- Before calling this interface, `CVI_VPSS_StopGrp` must be called to disable this GROUP first.
- The function will release all `vb_jobs` under `grp`.

【Example】

Please refer to [CVI_VPSS_CreateGrp](#).

【Related Topic】

- [CVI_VPSS_CreateGrp](#)

6.3.3 CVI_VPSS_GetGrpAttr

【Description】

Get the properties of VPSS GROUP.

【Syntax】

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

【Parameter】

Parameter	Description	Input/Output
VpssGrp	VPSS GROUP number. Value Range: [0, VPSS_MAX_GRP_NUM).	Input
pstGrpAttr	VPSS GROUP properties pointer.	Output

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: cvi_comm_vpss.h, cvi_vpss.h
- Library files: libvpu.a

【Note】

- Group must have been created.
- The GROUP properties must be valida, and some static properties cannot be set dynamically. Please refer to PSS_GRP_ATTR_S for details.

【Example】

Please refer to [CVI_VPSS_CreateGrp](#).

【Related Topic】

- [CVI_VPSS_CreateGrp](#)

6.3.4 CVI_VPSS_SetGrpAttr

【Description】

Set the properties of VPSS GROUP.

【Syntax】

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

【Parameter】

Parameter	Description	Input/Output
VpssGrp	VPSS GROUP number. Value Range: [0, VPSS_MAX_GRP_NUM).	Input
pstGrpAttr	VPSS GROUP properties pointer.	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: `cvi_comm_vpss.h`, `cvi_vpss.h`
- Library files: `libvpu.a`

【Note】

- Group must have been created.
- The GROUP properties must be valid, and some static properties cannot be set dynamically. Please refer to `PSS_GRP_ATTR_S` for details.

【Example】

Please refer to [CVI_VPSS_CreateGrp](#).

【Related Topic】

- [CVI_VPSS_CreateGrp](#)

6.3.5 CVI_VPSS_StartGrp

【Description】

Enable the VPSS GROUP.

【Syntax】

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

【Parameter】

Parameter	Description	Input/Output
VpssGrp	VPSS GROUP number. Value Range: [0, VPSS_MAX_GRP_NUM).	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: `cvi_comm_vpss.h`, `cvi_vpss.h`
- Library files: `libvpu.a`

【Note】

- Group must have been created.
- Repetitive calls to this function to set the same group will return a Success.

【Example】

Please refer to [CVI_VPSS_CreateGrp](#).

【Related Topic】

- [CVI_VPSS_CreateGrp](#)

6.3.6 CVI_VPSS_StopGrp

【Description】

Disable the VPSS GROUP.

【Syntax】

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

【Parameter】

Parameter	Description	Input/Output
VpssGrp	VPSS GROUP number. Value Range: [0, VPSS_MAX_GRP_NUM).	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: cvi_comm_vpss.h、cvi_vpss.h
- Library files: libvpu.a

【Note】

- Group must have been created, otherwise it will return a failure.
- Repetitive calls to disable the same VPSS group will return a Success

【Example】

Please refer to [CVI_VPSS_CreateGrp](#).

【Related Topic】

- [CVI_VPSS_CreateGrp](#)

6.3.7 CVI_VPSS_ResetGrp

【Description】

Reset the VPSS GROUP.

【Syntax】

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

【Parameter】

Parameter	Description	Input/Output
VpssGrp	VPSS GROUP number. Value Range: [0, VPSS_MAX_GRP_NUM).	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: `cvi_comm_vpss.h`, `cvi_vpss.h`
- Library files: `libvpu.a`

【Note】

Group must have been created .

【Example】

Please refer to [CVI_VPSS_CreateGrp](#).

【Related Topic】

- [CVI_VPSS_CreateGrp](#)

6.3.8 CVI_VPSS_GetGrpProcAmpCtrl

【Description】

Get a VPSS group color control function description.

【Syntax】

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

【Parameter】

Parameter	Description	Input/Output
VpssGrp	VPSS GROUP number. Value Range: [0, VPSS_MAX_GRP_NUM).	Input
type	Type of Proc Amp(color control);	Input
ctrl	Define the specific parameter of ProcAmp, including min, max, step, default, etc	Output

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: `cvi_comm_vpss.h`, `cvi_vpss.h`
- Library files: `libvpu.a`

【Note】

None.

【Example】

```
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;
}
```

(continues on next page)

(continued from previous page)

```

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;
}

```

【Related Topic】

- PROC_AMP_E
- PROC_AMP_CTRL_S
- [CVI_VPSS_GetGrpProcAmp](#)
- [CVI_VPSS_SetGrpProcAmp](#)

6.3.9 CVI_VPSS_GetGrpProcAmp**【Description】**

Get the color control property of a VPSS GROUP.

【Syntax】

```
CVI_S32 CVI\_VPSS\_GetGrpProcAmp(VPSS_GRP VpssGrp, PROC_AMP_E type, CVI_S32 *value);
```

【Parameter】

Parameter	Description	Input/Output
VpssGrp	VPSS GROUP number. Value Range: [0, VPSS_MAX_GRP_NUM).	Input
type	Type of Proc Amp(color control);	Input
value	Configuration of ProcAmp	Output

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: [cvi_comm_vpss.h](#), [cvi_vpss.h](#)
- Library files: [libvpu.a](#)

【Note】

None.

【Example】

Please refer to [CVI_VPSS_GetGrpProcAmpCtrl](#).

【Related Topic】

- PROC_AMP_E
- PROC_AMP_CTRL_S
- *CVI_VPSS_GetGrpProcAmpCtrl*
- *CVI_VPSS_SetGrpProcAmp*

6.3.10 CVI_VPSS_SetGrpProcAmp

【Description】

Set the color control property of a VPSS GROUP.

【Syntax】

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

【Parameter】

Parameter	Description	Input/Output
VpssGrp	VPSS GROUP number. Value Range: [0, VPSS_MAX_GRP_NUM).	Input
type	Type of Proc Amp(color control);	Input
value	Configuration of ProcAmp	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: cvi_comm_vpss.h, cvi_vpss.h
- Library files: libvpu.a

【Note】

None.

【Example】

Please refer to CVI_VPSS_GetGrpProcAmpCtrl.

【Related Topic】

- PROC_AMP_E
- PROC_AMP_CTRL_S
- *CVI_VPSS_GetGrpProcAmpCtrl*
- *CVI_VPSS_GetGrpProcAmp*

6.3.11 CVI_VPSS_SetGrpParamfromBin

【Description】

Set the property of a VPSS GROUP according to Bin.

【Syntax】

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

【Parameter】

Parameter	Description	Input/Output
VpssGrp	VPSS GROUP number. Value Range: [0, VPSS_MAX_GRP_NUM).	Input
scene	VPSS Bin scene settings to be applied	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: cvi_comm_vpss.h, cvi_vpss.h
- Library files: libvpu.a

【Note】

- If a new PQ bin is used, the group must already be created
- If the PQ bin does not exist, default parameters will be used.
- Currently, only the setting of ProcAmp (brightness, contract, hue, saturation) is supported

【Example】

```
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;
```

(continues on next page)

(continued from previous page)

```

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 套用場景 0*/
s32Ret = CVI_VPSS_SetGrpParamfromBin(0, 0);
if (s32Ret != CVI_SUCCESS) {
    SAMPLE_PRT("CVI_VPSS_SetGrpParamfromBin failed with %#x!\n", s32Ret);
    return CVI_FAILURE;
}

```

【Related Topic】

- PROC AMP_E
- PROC AMP_CTRL_S
- *CVI_VPSS_GetGrpProcAmpCtrl*
- *CVI_VPSS_GetGrpProcAmp*

6.3.12 CVI_VPSS_GetChnAttr**【Description】**

Get VPSS channel properties.

【Syntax】

```
CVI_S32 CVI_VPSS_GetChnAttr (VPSS_GRP VpssGrp, VPSS_CHN VpssChn, VPSS_CHN_ATTR_S
↪ *pstChnAttr);
```

【Parameter】

Parameter	Description	Input/Output
VpssGrp	VPSS GROUP number. Value Range: [0, VPSS_MAX_GRP_NUM).	Input
VpssChn	VPSS channel number. Value Range: [0, VPSS_MAX_PHY_CHN_NUM);	Input
pstChnAttr	VPSS channel properties	Output

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: *cvi_comm_vpss.h*, *cvi_vpss.h*

- Library files: libvpu.a

【Note】

Group must have been created.

【Example】

Please refer to [CVI_VPSS_CreateGrp](#).

【Related Topic】

- [CVI_VPSS_CreateGrp](#)

6.3.13 CVI_VPSS_SetChnAttr

【Description】

Set the VPSS channel properties.

【Syntax】

```
CVI_S32 CVI_VPSS_SetChnAttr (VPSS_GRP VpssGrp, VPSS_CHN VpssChn, const VPSS_CHN_ATTR_
↪S *pstChnAttr);
```

【Parameter】

Parameter	Description	Input/Output
VpssGrp	VPSS GROUP number. Value Range: [0, VPSS_MAX_GRP_NUM).	Input
VpssChn	VPSS channel number. Value Range: [0, VPSS_MAX_PHY_CHN_NUM);	Input
pstChnAttr	VPSS channel properties	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: cvi_comm_vpss.h、cvi_vpss.h
- Library files: libvpu.a

【Note】

Group must have been created.

【Example】

Please refer to [CVI_VPSS_CreateGrp](#).

【Related Topic】

- [CVI_VPSS_CreateGrp](#)

6.3.14 CVI_VPSS_EnableChn

【Description】

Enable the VPSS channel.

【Syntax】

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

【Parameter】

Parameter	Description	Input/Output
VpssGrp	VPSS GROUP number. Value Range: [0, VPSS_MAX_GRP_NUM).	Input
VpssChn	VPSS channel number. Value Range: [0, VPSS_MAX_PHY_CHN_NUM);	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: cvi_comm_vpss.h, cvi_vpss.h
- Library files: libvpu.a

【Note】

- Group must have been created.
- Repetitive enable calls will return a Success.

【Example】

Please refer to [CVI_VPSS_CreateGrp](#).

【Related Topic】

- [CVI_VPSS_CreateGrp](#)

6.3.15 CVI_VPSS_DisableChn

【Description】

Disable the VPSS channel.

【Syntax】

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

【Parameter】

Parameter	Description	Input/Output
VpssGrp	VPSS GROUP number. Value Range: [0, VPSS_MAX_GRP_NUM).	Input
VpssChn	VPSS channel number. Value Range: [0, VPSS_MAX_PHY_CHN_NUM);	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: `cvi_comm_vpss.h`, `cvi_vpss.h`
- Library files: `libvpu.a`

【Note】

- Group must have been created.
- Repetitive disable calls will return a Success.

【Example】

Please refer to [CVI_VPSS_CreateGrp](#).

【Related Topic】

- [CVI_VPSS_CreateGrp](#)

6.3.16 CVI_VPSS_SetGrpCrop

【Description】

Set the VPSS GROUP CROP function properties.

【Syntax】

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

【Parameter】

Parameter	Description	Input/Output
VpssGrp	VPSS GROUP number. Value Range: [0, VPSS_MAX_GRP_NUM).	Input
pstCropInfo	CROP function properties	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: `cvi_comm_vpss.h`, `cvi_vpss.h`
- Library files: `libvpu.a`

【Note】

- Group must have been created.
- The size of the crop area cannot be less than the minimum size of the VPSS, and cannot exceed the maximum size.
Otherwise, the failure is returned.

- VPSS Online mode does not support group cropping.
- This setting is invalid when the DIS is enabled

【Example】

Please refer to [CVI_VPSS_CreateGrp](#).

【Related Topic】

- [CVI_VPSS_CreateGrp](#)

6.3.17 CVI_VPSS_GetGrpCrop

【Description】

Get the VPSS GROUP CROP function properties.

【Syntax】

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

【Parameter】

Parameter	Description	Input/Output
VpssGrp	VPSS GROUP number. Value Range: [0, VPSS_MAX_GRP_NUM).	Input
pstCropInfo	CROP function properties	Output

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: `cvi_comm_vpss.h`, `cvi_vpss.h`
- Library files: `libvpu.a`

【Note】

Group must have been created.

【Example】

Please refer to [CVI_VPSS_CreateGrp](#).

【Related Topic】

- [CVI_VPSS_CreateGrp](#)

6.3.18 CVI_VPSS_SendFrame

【Description】

User sends data to VPSS.

【Syntax】

```
CVI_S32 CVI_VPSS_SendFrame (VPSS_GRP VpssGrp, const VIDEO_FRAME_INFO_S *pstVideoFrame,
↪ CVI_S32 s32MilliSec);
```

【Parameter】

Parameter	Description	Input/Output
VpssGrp	VPSS GROUP number. Value Range: [0, VPSS_MAX_GRP_NUM);	Input
pstVideoFrame	Image information to be sent. Please refer to the chapter “system control” for details .	Input
s32MilliSec	Not used in the current version.	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: cvi_comm_vpss.h, cvi_vpss.h
- Library files: libvpu.a

【Note】

- Group must have been created and enabled, otherwise it will return a failure.

【Example】

None.

【Related Topic】

None.

6.3.19 CVI_VPSS_GetChnFrame

【Description】

The user obtains a processed image from the channel.

【Syntax】

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

【Parameter】

Parameter	Description	Input/Output
VpssGrp	VPSS GROUP number. Value Range: [0, VPSS_MAX_GRP_NUM)。	Input
VpssChn	VPSS channel number. Value Range: [0, VPSS_MAX_CHN_NUM);	Input
pstVideoFrame	Image information to be sent. Please refer to the chapter “system control” for details.	Output
s32MilliSec	Time out	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: cvi_comm_vpss.h、cvi_vpss.h
- Library files: libvpu.a

【Note】

- Group must have been created.
- This function saves the acquired image information in pstvideoFrame, including the virtual and physical addresses of the image.

【Example】

```
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;

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);
if (CVI_VPSS_ReleaseChnFrame(0, 0, &pstVideoFrame) != 0)
    SAMPLE_PRT("CVI_VI_ReleaseChnFrame NG\n");
```

【Related Topic】

- [CVI_VPSS_ReleaseChnFrame](#)

6.3.20 CVI_VPSS_SendChnFrame

【Description】

The user specifies channel data to the VPSS, which can be used for image stitching.

【Syntax】

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

【Parameter】

Parameter	Description	Input/Output
VpssGrp	VPSS GROUP number. Value Range: [0, VPSS_MAX_GRP_NUM)。	Input
VpssChn	VPSS channel number. Value Range: [0, VPSS_MAX_CHN_NUM);	Input
pstVideoFrame	Image information to be sent. Please refer to the chapter “system control” for details.	Input
s32MilliSec	Not used in current version.	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: cvi_comm_vpss.h, cvi_vpss.h
- Library files: libvpu.a

【Note】

- Group/Chn must have been created.

【Example】

None.

【Related Topic】

None.

6.3.21 CVI_VPSS_ReleaseChnFrame

【Description】

The user releases a frame of image processed by the channel.

【Syntax】

```
CVI_S32 CVI_VPSS_ReleaseChnFrame (VPSS_GRP VpssGrp, VPSS_CHN VpssChn, VIDEO_FRAME_
↪INFO_S *pstFrameInfo);
```

【Parameter】

Parameter	Description	Input/Output
VpssGrp	VPSS GROUP number. Value Range: [0, VPSS_MAX_GRP_NUM)。	Input
VpssChn	VPSS channel number. Value Range: [0, VPSS_MAX_CHN_NUM);	Input
pstVideoFrame	Image information to be sent. Please refer to the chapter “system control” for details.	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: cvi_comm_vpss.h、cvi_vpss.h
- Library files: libvpu.a

【Note】

This interface should be used in conjunction with *CVI_VPSS_GetChnFrame*.

【Example】

Please refer to *CVI_VPSS_GetChnFrame*.

【Related Topic】

- *CVI_VPSS_GetChnFrame*

6.3.22 CVI_VPSS_GetGrpFrame

【Description】

The function has not been implemented so far.

6.3.23 CVI_VPSS_ReleaseGrpFrame

【Description】

The function has not been implemented so far.

6.3.24 CVI_VPSS_SetChnCrop

【Description】

Set the attributes of VPSS channel clipping function

【Syntax】

```
CVI_S32 CVI_VPSS_SetChnCrop(VPSS_GRP VpssGrp, VPSS_CHN VpssChn, const VPSS_CROP_INFO_
↪ S *pstCropInfo);
```

【Parameter】

Parameter	Description	Input/Output
VpssGrp	VPSS GROUP number. Value Range:[0, VPSS_MAX_GRP_NUM)。	Input
VpssChn	VPSS channel number. Value Range:[0, VPSS_MAX_CHN_NUM);	Input
pstCropInfo	Image information to be sent. Please refer to the chapter “system control” for detailed description.	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: cvi_comm_vpss.h、cvi_vpss.h
- Library files: libvpu.a

【Note】

- Group must have been created.
- The size of the CROP area can neither be less than the minimum size of the VPSS nor exceed the maximum size.

Otherwise, a failure will be returned.

- This function sets the image for the VPSS output channel, while GrpCrop sets the image for the VPSS input.

【Example】

None.

【Related Topic】

- [CVI_VPSS_GetChnCrop](#)

6.3.25 CVI_VPSS_GetChnCrop

【Description】

Get the attributes of VPSS channel clipping function.

【Syntax】

```
CVI_S32 CVI_VPSS_GetChnCrop(VPSS_GRP VpssGrp, VPSS_CHN VpssChn, VPSS_CROP_INFO_S
↪*pstCropInfo);
```

【Parameter】

Parameter	Description	Input/Output
VpssGrp	VPSS GROUP number. Value Range:[0, VPSS_MAX_GRP_NUM)。	Input
VpssChn	VPSS channel number. Value Range:[0, VPSS_MAX_CHN_NUM);	Input
pstCropInfo	CROP function properties.	Output

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: `cvi_comm_vpss.h`, `cvi_vpss.h`
- Library files: `libvpu.a`

【Note】

- Group must have been created.
- This function obtains the image cropping information for the VPSS output channel, while `CVI_VPSS_GetGrpCrop` obtains the image cropping information for the VPSS input.

【Example】

None.

【Related Topic】

- [CVI_VPSS_SetChnCrop](#)

6.3.26 CVI_VPSS_SetChnRotation

【Description】

Set the attributes of VPSS channel rotation.

【Syntax】

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

【Parameter】

Parameter	Description	Input/Output
VpssGrp	VPSS GROUP number. Value Range:[0, VPSS_MAX_GRP_NUM)。	Input
VpssChn	VPSS channel number. Value Range:[0, VPSS_MAX_CHN_NUM);	Input
enRotation	Rotation attributes. See ROTATION_E for details.	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: `cvi_vpss.h`, `cvi_comm_vpss.h`
- Library files: `libvpu.a`

【Note】

- Before using this interface, it is necessary to call [CVI_VPSS_SetChnAttr](#) first, otherwise it will fail.

- The channel attributes must be set before setting this attribute.
- CV181x/CV180x do not support 180-degree rotation, but it can be achieved using mirror+flip.
- Only NV12/NV21/YUV400 rotation formats are supported.
- After rotation, the output image size of the channel may change.

For example, the 1920x1080 input image, rotated 90 degrees, the actual output is 1088x1920.

Since rotation requires 64 pixel alignment, invalid areas will be generated.

If the back end is connected to vo or venc, the back end module will automatically crop the valid part.

In the case of GetChnFrame, the valid region is specified in the VIDEO_FRAME_S structure s16OffsetTop, s16OffsetBottom, s16OffsetLeft, s16OffsetRight.

【Example】

None.

【Related Topic】

None.

6.3.27 CVI_VPSS_GetChnRotation

【Description】

Get the attributes of VPSS channel rotation.

【Syntax】

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

【Parameter】

Parameter	Description	Input/Output
VpssGrp	VPSS GROUP number. Value Range:[0, VPSS_MAX_GRP_NUM)。	Input
VpssChn	VPSS channel number. Value Range:[0, VPSS_MAX_CHN_NUM);	Input
penRotation	Rotation attributes. See ROTATION_E for details.	Output

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: cvi_vpss.h, cvi_comm_vpss.h
- Library files: libvpu.a

【Note】

- Before using this interface, it is necessary to call *CVI_VPSS_SetChnAttr* first, otherwise it will fail.

【Example】

None.

【Related Topic】

- [CVI_VPSS_SetChnRotation](#)

6.3.28 CVI_VPSS_SetChnLDCAttr

【Description】

Set the attributes of VPSS channel lens distortion correction.

【Syntax】

```
CVI_S32 CVI_VPSS_SetChnLDCAttr(VPSS_GRP VpssGrp, VPSS_CHN VpssChn, const VPSS_LDC_ATTR_S *pstLDCAttr);
```

【Parameter】

Parameter	Description	Input/Output
VpssGrp	VPSS GROUP number. Value Range:[0, VPSS_MAX_GRP_NUM)。	Input
VpssChn	VPSS channel number. Value Range:[0, VPSS_MAX_CHN_NUM);	Input
pstLDCAttr	Lens distortion correction attributes.	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: cvi_vpss.h, cvi_comm_vpss.h
- Library files: libvpu.a

【Note】

- Before using this interface, it is necessary to call [CVI_VPSS_SetChnAttr](#) first, otherwise it will fail.
- The channel attributes must be set before setting this attribute
- Only NV21 and YUV400 rotation formats are supported.

【Example】

None.

【Related Topic】

- [CVI_VPSS_GetChnLDCAttr](#)
- VPSS_LDC_ATTR_S

6.3.29 CVI_VPSS_GetChnLDCAttr

【Description】

Get the attributes of VPSS channel lens distortion correction.

【Syntax】

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

【Parameter】

Parameter	Description	Input/Output
VpssGrp	VPSS GROUP number. Value Range:[0, VPSS_MAX_GRP_NUM)。	Input
VpssChn	VPSS channel number. Value Range:[0, VPSS_MAX_CHN_NUM);	Input
pstLDCAttr	Lens distortion correction attributes.	Output

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: cvi_vpss.h, cvi_comm_vpss.h
- Library files: libvpu.a

【Note】

- Before using this interface, it is necessary to call *CVI_VPSS_SetChnAttr* first, otherwise it will fail.

【Example】

None.

【Related Topic】

- *CVI_VPSS_SetChnLDCAttr*
- VPSS_LDC_ATTR_S

6.3.30 CVI_VPSS_GetChnFd

【Description】

Get the device file handle corresponding to the VPSS channel.

【Syntax】

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

【Parameter】

Parameter	Description	Input/Output
VpssGrp	VPSS GROUP number. Value Range:[0, VPSS_MAX_GRP_NUM)。	Input
VpssChn	VPSS channel number. Value Range:[0, VPSS_MAX_CHN_NUM);	Input

【Return Value】

Return Value	Description
Positive value	Success.
Negative value	invalid value

【Requirement】

- Header files: cvi_vpss.h, cvi_comm_vpss.h
- Library files: libvpu.a

【Note】

None.

【Example】

None.

【Related Topic】

None.

6.3.31 CVI_VPSS_CloseFd

【Description】

Close the file handle of the VPSS device channel.

【Syntax】

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

【Parameter】

None.

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: cvi_vpss.h, cvi_comm_vpss.h
- Library files: libvpu.a

【Note】

After calling this interface, other MMF interfaces of VPSS will be invalidated.

【Example】

None.

【Related Topic】

- *CVI_VPSS_GetChnFd*

6.3.32 CVI_VPSS_AttachVbPool

【Description】

Bind the VPSS channel to a video cache VB pool.

【Syntax】

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

【Parameter】

Parameter	Description	Input/Output
VpssGrp	VPSS GROUP number. Value Range:[0, VPSS_MAX_GRP_NUM)。	Input
VpssChn	VPSS channel number. Value Range:[0, VPSS_MAX_CHN_NUM);	Input
hVbPool	Video memory block pool ID	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: `cvi_vpss.h`, `cvi_comm_vpss.h`
- Library files: `libvpu.a`

【Note】

- Group must already be created.
- Repeated calls to this function to set the same group return Success.
- The CVI_VPSS_AttachVbPool interface needs to be adjusted again when switching the VB pool bound to the current group.
- Correct configuration to the VB pool.
- hVbPool must be a valid Pool Id of a created VB pool.
- After calling CVI_VPSS_DetachVbPool, before destroying the created VB, we need to ensure that the VB is not used by the back-end bound module of VPSS.

We can release the VB by sleep or clear the cache of the back-end module channel, and then destroy the cache VB pool.

- It needs to be used with the module parameter of the video cache pool.

If you want to change whether to bind the VB pool, it needs to destroy all VPSS groups, set the module parameter, and create the group again.

- VB size is calculated according to the output image of VPSS channel.

For details, refer to `cvi_buffer.h`.

【Example】

None.

【Related Topic】

None.

6.3.33 CVI_VPSS_DetachVbPool

【Description】

Unbind the VPSS channel from a video cache VB pool

【Syntax】

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

【Parameter】

Parameter	Description	Input/Output
VpssGrp	VPSS GROUP number. Value Range:[0, VPSS_MAX_GRP_NUM)。	Input
VpssChn	VPSS channel number. Value Range:[0, VPSS_MAX_CHN_NUM);	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: cvi_vpss.h, cvi_comm_vpss.h
- Library files: libvpu.a

【Note】

- Group must already be created

【Example】

None.

【Related Topic】

None.

6.3.34 CVI_VPSS_SetChnBufWrapAttr

【Description】

Setting low-latency looping property.

【Syntax】

```
CVI_S32 CVI_VPSS_SetChnBufWrapAttr(VPSS_GRP VpssGrp, VPSS_CHN VpssChn, const VPSS_CHN_
↳BUF_WRAP_S *pstVpssChnBufWrap);
```

【Parameter】

Parameter	Description	Input/Output
VpssGrp	VPSS GROUP number. Value Range:[0, VPSS_MAX_GRP_NUM)。	Input
VpssChn	VPSS channel number. Value Range:[0, VPSS_MAX_CHN_NUM);	Input
pstVpssChnBufWrap	Channel Buffer Loop Attribute structure.	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: cvi_vpss.h, cvi_comm_vpss.h
- Library files: libvpu.a

【Note】

- Group must already be created.
- Channel enablement is not supported.
- The interface can be set only after the channel attributes have been set.
- semi-planar422 graphics and Flip graphics are not supported.
- When channel winding with low delay is enabled, access, brightness, rotation, CoverEx, OverlayEx, rotation at any Angle, and LDC of the channel are invalid.
- In low-delay winding channel binding coding scenarios, the software does not support frame rate control.

【Example】

None.

【Related Topic】

None.

6.3.35 CVI_VPSS_GetChnBufWrapAttr

【Description】

Get the low-latency looping property.

【Syntax】

```
CVI_S32 CVI_VPSS_GetChnBufWrapAttr(VPSS_GRP VpssGrp, VPSS_CHN VpssChn, VPSS_CHN_BUF_
↳ WRAP_S *pstVpssChnBufWrap);
```

【Parameter】

Parameter	Description	Input/Output
VpssGrp	VPSS GROUP number. Value Range:[0, VPSS_MAX_GRP_NUM)。	Input
VpssChn	VPSS channel number. Value Range:[0, VPSS_MAX_CHN_NUM);	Input
pstVpssChnBufWrap	Channel Buffer Loop Attribute structure.	Output

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: cvi_vpss.h, cvi_comm_vpss.h
- Library files: libvpu.a

【Note】

- Group must already be created.

【Example】

None.

【Related Topic】

None.

6.3.36 CVI_VPSS_GetWrapBufferSize

【Description】

Get the required buffer size for low-latency wrapping

【Syntax】

```
CVI_U32 CVI_VPSS_GetWrapBufferSize(CVI_U32 u32Width, CVI_U32 u32Height, PIXEL_FORMAT_
↪E enPixelFormat, CVI_U32 u32BufLine, CVI_U32 u32BufDepth);
```

【Parameter】

Parameter	Description	Input/Output
u32Width	Image width	Input
u32Height	Image height	Input
enPixelFormat	Image pixel format	Input
u32BufLine	Number of lines in a single winding cache	Input
u32BufDepth	Number of winding caches	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: cvi_vpss.h, cvi_comm_vpss.h
- Library files: libvpu.a

【Note】

None.

【Example】

None.

【Related Topic】

None.

6.4 Data Types

The data types of VPSS module are defined as follows:

- *VPSS_MAX_GRP_NUM*: Define the maximum number of VPSS GROUP.
- *VPSS_MAX_CHN_NUM*: Define the maximum number of VPSS channels.
- *VPSS_MAX_PHY_CHN_NUM*: Define the maximum number of VPSS physical channels.
- *VPSS_MIN_IMAGE_WIDTH*: Define the minimum width of the VPSS image.
- *VPSS_MIN_IMAGE_HEIGHT*: Define the minimum height of the VPSS image.
- *VPSS_MAX_IMAGE_WIDTH*: Define the maximum width of the VPSS image.
- *VPSS_MAX_IMAGE_HEIGHT*: Define the maximum height of the VPSS image.
- *VPSS_MAX_ZOOMIN*: Definition of the maximum zoom-in factor of VPSS physical channels.
- *VPSS_MAX_ZOOMOUT*: Definition of the maximum zoom-out factor of VPSS physical channels.
- *VPSS_GRP*: Define the VPSS GROUP number.
- *VPSS_CHN*: Define the VPSS channel number.
- *VPSS_CROP_COORDINATE_E*: Define the start point coordinate mode of CROP.
- *VPSS_CROP_INFO_S*: Define the information required for the CROP function.
- *VPSS_GRP_ATTR_S*: Define the VPSS GROUP attribute.
- *VPSS_CHN_ATTR_S*: Define VPSS physical channel properties.
- *VPSS_MOD_PARAM_S*: Set the module parameters through the interface.
- *VPSS_CHN_BUF_WRAP_S*: Definition of VPSS buffer wrap property.

6.4.1 VPSS_MAX_GRP_NUM

【Description】

Define the maximum number of VPSS GROUP.

【Syntax】

```
#define VPSS_MAX_GRP_NUM 16
```

【Note】

None.

【Related Data Type and Interface】

None.

6.4.2 VPSS_MAX_CHN_NUM

【Description】

Define the maximum number of VPSS channels.

【Syntax】

```
#define VPSS_MAX_CHN_NUM VPSS_MAX_PHY_CHN_NUM
```

【Note】

The maximum number of VPSS channels is the number of physical channels.

【Related Data Type and Interface】

None.

6.4.3 VPSS_MAX_PHY_CHN_NUM

【Description】

Define the maximum number of VPSS physical channels.

【Syntax】

```
// cv181x  
#define VPSS_MAX_PHY_CHN_NUM 4  
// cv180x  
#define VPSS_MAX_PHY_CHN_NUM 3
```

【Note】

None.

【Related Data Type and Interface】

None.

6.4.4 VPSS_MIN_IMAGE_WIDTH

【Description】

Define the minimum width of the VPSS image.

【Syntax】

```
#define VPSS_MIN_IMAGE_WIDTH 32
```

【Note】

None.

【Related Data Type and Interface】

None.

6.4.5 VPSS_MIN_IMAGE_HEIGHT

【Description】

Define the minimum height of the VPSS image.

【Syntax】

```
#define VPSS_MIN_IMAGE_HEIGHT 32
```

【Note】

None.

【Related Data Type and Interface】

None.

6.4.6 VPSS_MAX_IMAGE_WIDTH

【Description】

Define the maximum width of the VPSS image.

【Syntax】

```
#define VPSS_MAX_IMAGE_WIDTH 2880
```

【Note】

None.

【Related Data Type and Interface】

None.

6.4.7 VPSS_MAX_IMAGE_HEIGHT

【Description】

Define the maximum height of the VPSS image.

【Syntax】

```
#define VPSS_MAX_IMAGE_HEIGHT 2880
```

【Note】

None.

【Related Data Type and Interface】

None.

6.4.8 VPSS_MAX_ZOOMIN

【Description】

Definition of the maximum zoom-in factor of VPSS physical channels.

【Syntax】

<code>#define VPSS_MAX_ZOOMIN</code>	<code>32</code>
--------------------------------------	-----------------

【Note】

None.

【Related Data Type and Interface】

None.

6.4.9 VPSS_MAX_ZOOMOUT

【Description】

Definition of the maximum zoom-out factor of VPSS physical channels.

【Syntax】

<code>#define VPSS_MAX_ZOOMOUT</code>	<code>32</code>
---------------------------------------	-----------------

【Note】

None.

【Related Data Type and Interface】

None.

6.4.10 VPSS_GRP

【Description】

Define VPSS group number.

【Syntax】

<code>typedef CVI_U32 VPSS_GRP;</code>
--

【Note】

None.

【Related Data Type and Interface】

None.

6.4.11 VPSS_CHN

【Description】

Define the VPSS channel number.

【Syntax】

```
typedef CVI_U32 VPSS_CHN;
```

【Note】

None.

【Related Data Type and Interface】

None.

6.4.12 VPSS_ROUNDING_E

【Description】

Define the pattern of rounding mode during Normalize.

【Syntax】

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

【Member】

Member	Description
VPSS_ROUNDING_TO_EVEN	Round off, refer to the table below.
VPSS_ROUNDING_AWAY_FROM_ZERO	Round away from zero, refer to the table below.
VPSS_ROUNDING_TRUNCATE	Unconditional rounding, see table below.

【Note】

	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	
−0.2			
−0.5		−1	
−0.8	−1		
−1.2			−1
−1.5	−2	−2	
−1.8			

【Related Data Type and Interface】

6.4.13 VPSS_CROP_COORDINATE_E

【Description】

Define the start point coordinate mode of CROP.

【Syntax】

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

【Member】

Member	Description
VPSS_CROP_RATIO_COOR	Relative coordinates.
VPSS_CROP_ABS_COOR	Absolute coordinates.

【Note】

None.

【Related Data Type and Interface】

- [VPSS_CROP_INFO_S](#)

6.4.14 VPSS_NORMALIZE_S

【Description】

Define the information required for Normalize function.

【Syntax】

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

【Member】

Member	Description
bEnable	Normalize enable switch.
factor	Normalization factor. 1/8192 ~ 8191/8192 or 1/4096 ~ 8191/4096
mean	Normalized root mean square difference. 0 ~ 255
rounding	Method to deal with the decimal part.

【Note】

1. After Normalize is enabled, the Output is int8, -128 ~ 127
2. Normalize is equivalent to convertTo() function in OpenCV

```
cv::convertTo(image, CV_8S, factor, -mean)
```

3. According to the format of VPSS CHN, the results will be different.

If the format is `PIXEL_FORMAT_RGB_888_PLANAR` or `PIXEL_FORMAT_RGB_888`, the operation modes are as follows:

- (Pixel Value R) * factor[0] – mean[0] and then rounding.
- (Pixel Value G) * factor[1] – mean[1] and then rounding.
- (Pixel Value B) * factor[2] – mean[2] and then rounding.

If the format is `PIXEL_FORMAT_BGR_888`, the operation modes are as follows:

- (Pixel Value B) * factor[0] – mean[0] and then rounding.
- (Pixel Value G) * factor[1] – mean[1] and then rounding.
- (Pixel Value R) * factor[2] – mean[2] and then rounding.

【Related Data Type and Interface】

- [*VPSS_ROUNDING_E*](#)

6.4.15 VPSS_CROP_INFO_S

【Description】

Define the information required for the CROP function.

【Syntax】

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

【Member】

Member	Description
bEnable	CROP enable switch.
enCropCoordinate	CROP start point coordinate mode
stCropRect	The rectangular area of CROP.

【Note】

None.

【Related Data Type and Interface】

- [*VPSS_CROP_COORDINATE_E*](#)

6.4.16 VPSS_GRP_ATTR_S

【Description】

Define the VPSS GROUP attribute.

【Syntax】

```
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;
```

【Member】

Member	Description
u32MaxW	The input image width.
u32MaxH	The input image height.
enPixelFormat	The input image pixel format.
stFrameRate	Group frame rate.
u8VpssDev	Specify which hardware this VPSS group will work with.

【Note】

The parameter u8VpssDev should be set to 0 or 1 after setting the VPSS_MODE_DUAL or VPSS_MODE_RGNEX in CVI_SYS_SetVPSSMode to specify different hardware devices.

【Related Data Type and Interface】

- PIXEL_FORMAT_E
- CVI_VPSS_CreateGrp
- CVI_VPSS_SetGrpAttr
- CVI_VPSS_GetGrpAttr

6.4.17 VPSS_CHN_ATTR_S

【Description】

Define VPSS physical channel properties.

【Syntax】

```
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;
```

【Member】

Member	Description
u32Width	Target image width
u32Height	Target image height
enVideoFormat	Target image video format.
enPixelFormat	Target image pixel format
stFrameRate	Frame rate control information
bMirror	Horizontal mirroring enable.
bFlip	Vertical flipping enable.
u32Depth	Get the queue length for obtaining channel images. This is a static property.
stAspectRatio	Amplitude shape ratio parameter.
stNormalize	Performing normalization to accelerate subsequent TPU operation.

【Note】

None.

【Related Data Type and Interface】

- CVI_VPSS_SetChnAttr

6.4.18 VPSS_MOD_PARAM_S

【Description】

Set the module parameters through the interface.

【Syntax】

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

【Member】

Member	Description
u32VpssVbSource	Video memory block pool type.
u32VpssSplitNodeNum	Number of block nodes.

【Note】

This structure attribute is not used in the current version.

【Related Data Type and Interface】

6.4.19 PROC_AMP_E

【Description】

Define the VPSS PROCAMP category.

【Syntax】

```
typedef enum _PROC_AMP_E {
    PROC_AMP_BRIGHTNESS = 0,
    PROC_AMP_CONTRAST,
```

(continues on next page)

(continued from previous page)

```

PROC_AMP_SATURATION,
PROC_AMP_HUE,
PROC_AMP_MAX,
} PROC_AMP_E;

```

【Member】

Member	Description
PROC_AMP_BRIGHTNESS	Brightness value.
PROC_AMP_CONTRAST	Contrast value.
PROC_AMP_SATURATION	Saturation value.
PROC_AMP_HUE	HUE value.

【Note】**【Related Data Type and Interface】**

- *PROC_AMP_CTRL_S*

6.4.20 PROC_AMP_CTRL_S**【Description】**

Define the VPSS PROCAMP property.

【Syntax】

```

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

```

【Member】

Member	Description
minimum	The minimum value of this color control function.
maximum	The maximum value of this color control function.
step	The effective adjustment value of this color control function.
default_value	The default value of this color control function.

【Note】

It is recommended to obtain these attributes before starting the operation on VPSS ProcAmp to avoid the operation failure.

【Related Data Type and Interface】

- PROC_AMP_E
- CVI_VPSS_GetGrpProcAmpCtrl

6.4.21 VPSS_LDC_ATTR_S

【Description】

Define the VPSS lens distortion correction structure.

【Syntax】

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

【Member】

Member	Description
bEnable	LDC enable
stAttr	LDC configuration parameters

【Note】

None.

【Related Data Type and Interface】

- LDC_ATTR_S
- CVI_VI_GetChnLDCAttr
- CVI_VI_SetChnLDCAttr

6.5 Error Codes

API error codes of video processing subsystem are shown in the following table

Error Code	Macro Definition	Description
0xC0068001	CVI_ERR_VPSS_INVALID_GROUP_ID	Invalid VPSS GROUP ID
0xC0068002	CVI_ERR_VPSS_INVALID_CHANNEL_ID	Invalid VPSS channel ID
0xC0068003	CVI_ERR_VPSS_ILLEGAL_PARAMETER	VPSS parameter setting is invalid
0xC0068004	CVI_ERR_VPSS_EXIST	VPSS GROUP already exists
0xC0068005	CVI_ERR_VPSS_UNEXIST	VPSS GROUP not created
0xC0068006	CVI_ERR_VPSS_NULL_PTR	Null pointer error
0xC0068008	CVI_ERR_VPSS_NOT_SUPPORTED	Operation not supported
0xC0068009	CVI_ERR_VPSS_NOT_PERMITTED	operation not permitted
0xC006800c	CVI_ERR_VPSS_NOMEM	Failure to allocate memory
0xC006800d	CVI_ERR_VPSS_NOBUF	Failure to allocate BUF pool
0xC006800e	CVI_ERR_VPSS_BUF_EMPTY	The image queue is empty
0xC0068010	CVI_ERR_VPSS_NOTREADY	VPSS system not initialized
0xC0068012	CVI_ERR_VPSS_BUSY	VPSS system is busy

VIDEO ENCODING

7.1 Function Overview

7.1.1 Objective

Video encoding provides real-time video compression of input video signals using standard video codecs to reduce data size and provide video services for the application layer.

The currently supported input sources are:

- Image data input by user under User Mode
- The video input (VI) module sends the image directly to the encoder
- The video input (VI) module receives the image, which is then processed by the video processing subsystem (VPSS) and sent to the encoder for encoding

Supported video standards currently include:

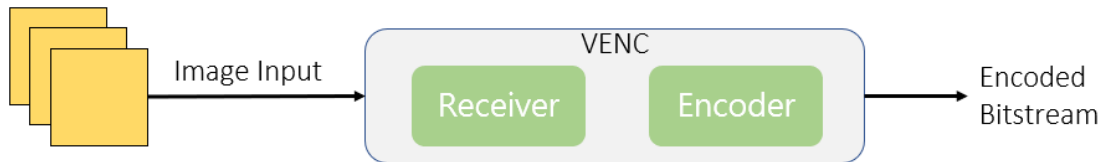
- JPEG
- MJPEG
- H.264
- H.265

7.1.2 Definitions and Abbreviations

- Encoded Bitstream
- Bitrate
- RC Rate Control
- Video Quality
- QP Quantization Parameter
- Fixed QP
- VBR Variable Bitrate
- CBR Constant Bitrate
- AVBR Adaptive Variable Bitrate
- MB Macroblock
- Frame
- Frame Buffer
- Packet

7.2 Design Overview

7.2.1 Flowchart of Video Encoding Data



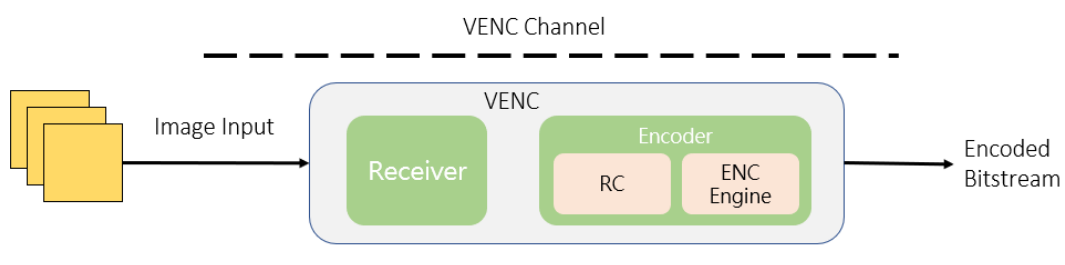
The input of VENC module is the image to be encoded and the output is the encoded Bitstream. VENC contains two sub modules, Receiver and Encoder.

- Receiver
 - Receives image input.
- Encoder
 - Encodes the received images into Encoded Bitstream.

The current video standards supported are:

- JPEG
 - The supported image formats are
 - * YUV422
 - * YUV420
 - * NV12 / NV21
- MJPEG
 - The supported image format is the same as JPEG
- H.264
 - The supported image formats are
 - * YUV420
 - * NV12 / NV21
- H.265
 - The supported image formats are
 - * YUV420
 - * NV12 / NV21

7.2.2 VENC Channels



VENC channel is the basic operation unit of video encoder, which mainly includes VENC related functions and channel related settings.

The system can have multiple video encoding channels, and each channel operates independently.

After each channel is established, users can set the basic parameters of the channel according to their needs, such as image resolution, encoding Bitrate, RC mechanism, etc.

The related parameters are used to initialize VENC.

7.2.3 Rate Control

RC is the main module to control Bitrate and Video Quality in encoder.

It further controls the image quality by the parameter QP in each video standard.

When QP becomes larger, the amount of encoded data will be smaller, but video quality will be worse; When QP becomes smaller, the amount of encoded data will be larger, but Video Quality will be better.

Video encoding will usually run with RC.

If it is picture encoding (such as JPEG), this module will not be needed because there are only single images.

Mode	MJPEG	H.264	H.265
Fixed QP	Supported	Supported	Supported
CBR	Supported	Supported	Supported
VBR	Not supported	Supported	Supported
AVBR	Not supported	Supported	Supported

7.2.4 Fixed QP

Within the statistical time, all MB in the image use the same QP value.

7.2.5 CBR

The Constant Bit Rate (CBR) is a fixed bitrate, and the QP is dynamically controlled.

The Encoded Bitstream remains at the desired bitrate within a specified time interval.

The main parameters are as follows:

- u32StatTime Bitrate (Statistical Time)

The time unit is second.

The encoder will statistically adjust the video quality to meet the desired bitrate within a specified time period.

When this value is smaller, the statistical interval is smaller, and the variation in image quality is relatively larger.

After each frame is compressed, it has a greater impact on the subsequent compression, and the video quality is less stable.

When this value is larger, the statistical interval is larger, and the impact of each compressed frame on the subsequent compression is relatively smaller, and the variation in image quality is smaller, resulting in a more stable video quality.

7.2.6 VBR

VBR (Variable Bit Rate) is a video encoding technique that allows for the dynamic adjustment of the overall bit rate within a set statistical time interval.

This technique allows for the bit rate to change as needed to maintain a consistent level of video quality, resulting in a more stable video stream.

- `u32MaxBitRate` Maximum bitrate
 - Set the maximum bitrate that can be used within the statistical time interval.
- `u32MinQp`
 - Set the minimum MB QP allows for limiting the best quality of the image without using excessive bitrate.

7.2.7 AVBR

AVBR (Adaptive Variable Bit Rate) is an adaptive version of variable bit rate encoding.

It dynamically adjusts the statistical target bit rate based on the level of motion in the scene.

The bitrate control algorithm internally detects the level of motion in the scene, and adjusts the target bitrate higher for higher motion scenes and lower for static scenes.

- `u32MaxBitRate` Maximum bitrate
 - Set the maximum bitrate that can be used within the statistical time interval.
- `u32MinStillPercent`
 - The target bitrate as a percentage of the maximum bitrate when the scene is in a static state.
 - $\text{MinBitrate} = \text{MaxBitrate} * \text{ChangePos} * \text{MinStillPercent}$
 - The target bitrate is adaptively adjusted between `MinBitrate` and `MaxBitrate` based on the scene motion level.
- `U32MaxStillQp`
 - When the scene is completely still, `maxQp` tends to `MaxStillQp` to ensure the picture quality of the still scene.

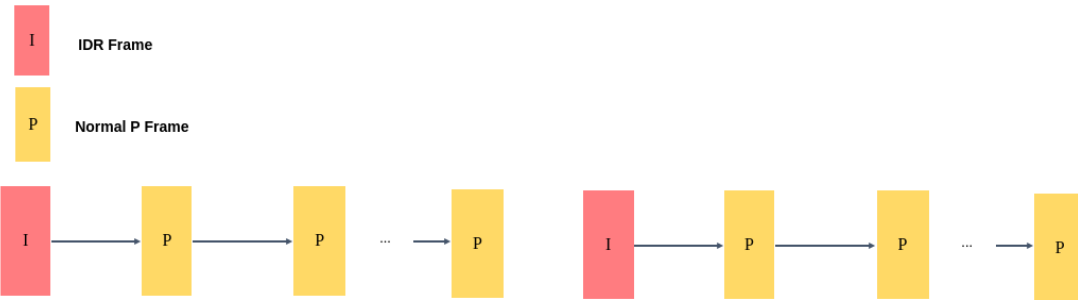
7.2.8 GOP Structure

The GOP (Group of Pictures) structure support for H.264 and H.265 is as follows:

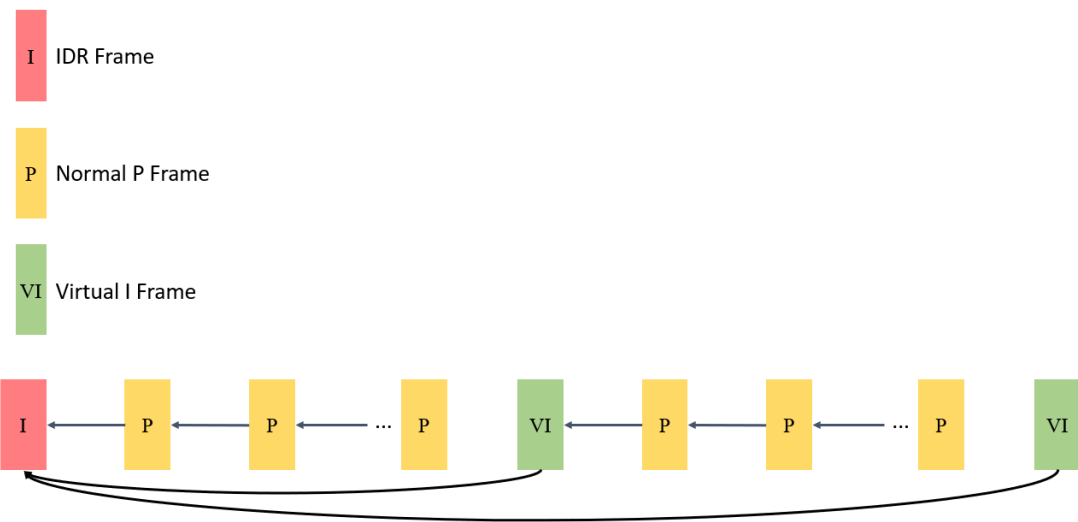
GOP mode	H.264	H.265
NormalP	supported	supported
SmartP	supported	supported

- The GOP structure of NormalP is as follow:
 - The occurrence period of IDR frame is `u32Gop`
- The GOP structure of SmartP is as follow:
 - The occurrence period of VI frame is `u32Gop`, which only refers to the adjacent IDR frame forward.
 - The occurrence period of IDR frame is `u32BgInterval`, which is a long-term reference frame.

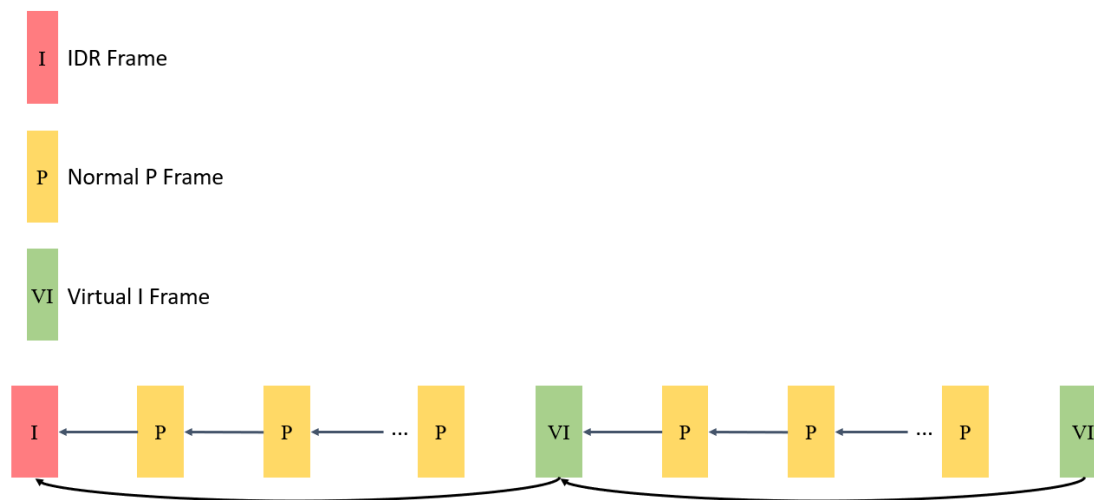
Normal P



H.264 Smart P



H.265 Smart P

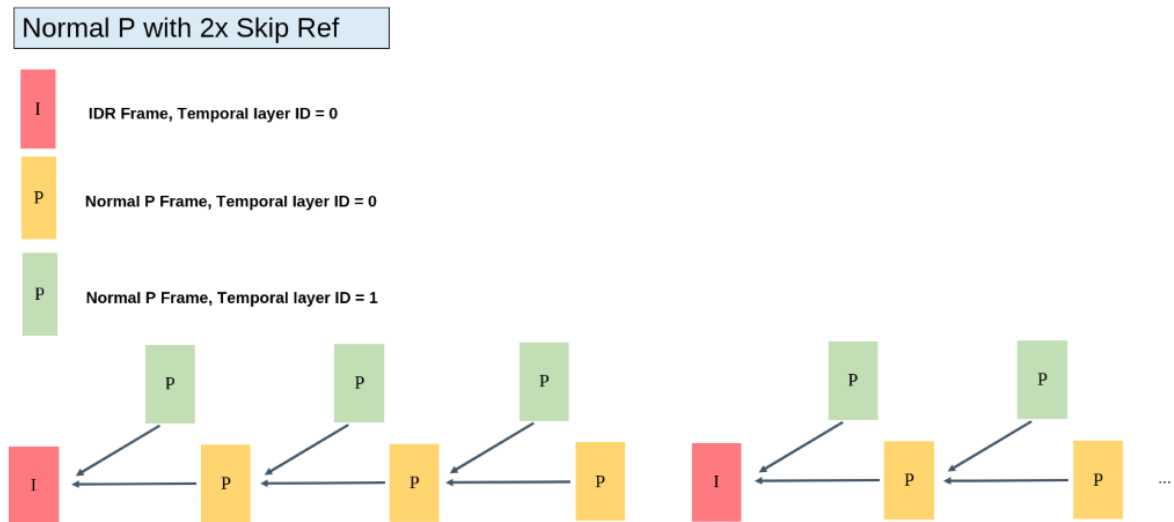


7.2.9 Advanced Frame Skipping

Both H.264 and H.265 support 1, 2 and 4 times frame skipping reference modes.

The default is 1 (no frame skipping).

- 2 times frame skipping reference mode is shown below

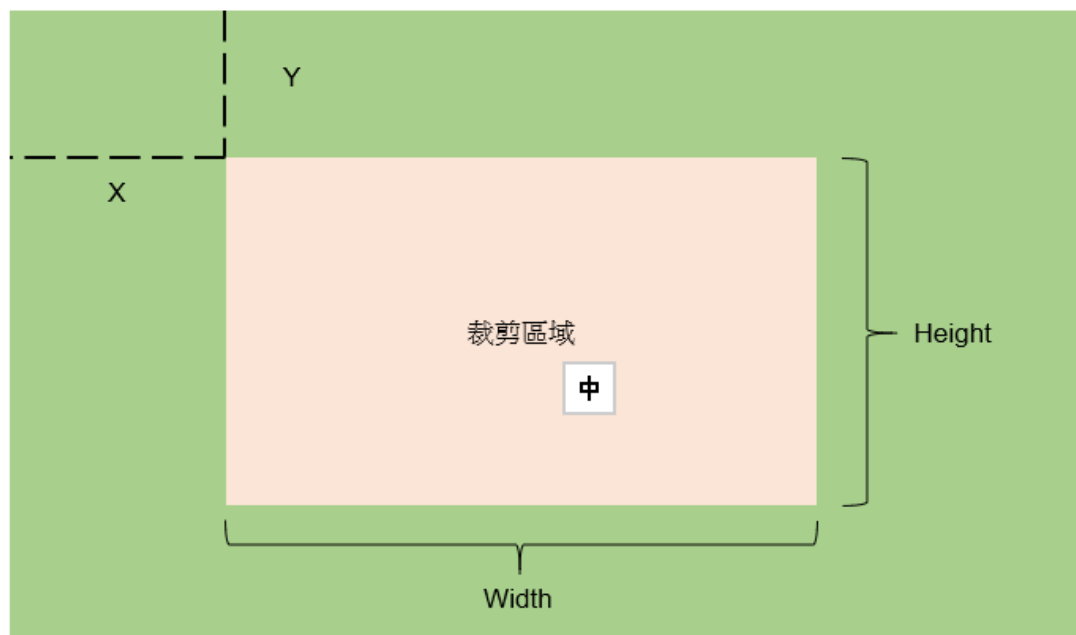


7.2.10 Cropping Encoding

For each source frame, you can only encode a certain area.

The location of the area is specified by X and Y, and the size is still specified by width and height.

Please refer to the usage of CVI_VENC_SetChnParam.



7.2.11 ROI

ROI(Region Of Interest) encoding: Region of Interest (ROI) encoding.

- Users can configure ROI regions to adjust the image Qp in that area, achieving differentiated image quality for local regions in the image.
- Both H.264 and H.265 support 8 ROI settings, and duplicate regions are prioritized based on the ROI index number from 0 to 7.
- Absolute Qp and relative Qp can be configured in ROI area.
 - Absolute Qp mode: the Qp in the ROI area is the Qp value set by the user.
 - Relative Qp mode: the Qp in the ROI area is the Qp of the rate control plus the Qp offset value set by the user.
- Note
 - When the rate control mode is not Fixed Qp mode, the ROI area can be configured.
 - When ROI is enabled in H.264, macroblock-level bitrate control is disabled.
 - In absolute QP mode, the actual encoded QP of macroblocks may differ slightly from the configured QP due to the bitrate control adjustment of macroblock QP.

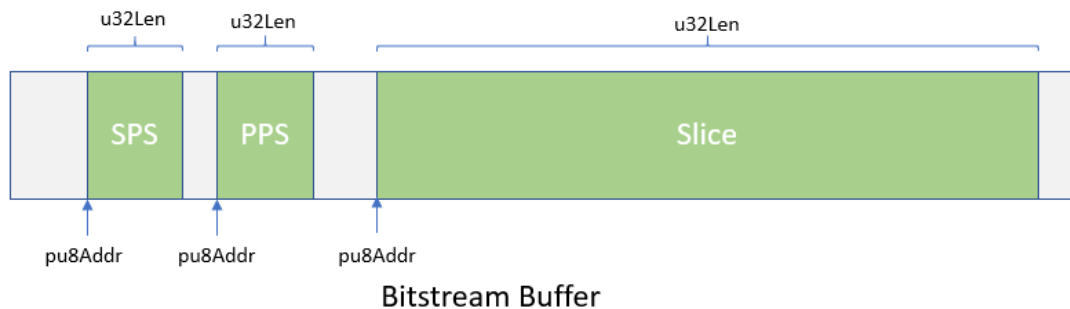
7.2.12 Coding Unit (CU) Slice Mode

After compression, use CVI_VENC_GetStream to obtain Encoded Bitstream. The related information is as follows:

- u32PackCount
 - How many NAL packets does this Frame contain?

For example, the current compressed format is H.264, the first frame is I frame, and the value will be 3 (including SPS, PPS, 1 Slice), and the second frame will be 1(including 1 Slice).
- pstPack
 - According to the number of u32PackCount, pstPack[0] ~ pstPack[u32PackCount - 1] data will be included.

For example, the first I Frame will have pstPack[0], pstPack[1], pstPack[2]
- The following figure shows the Bitstream data of the first I Frame



7.2.13 Multi-Encoder Parallel Encoding

H.264 and H.265 support multi-encoding, with up to 8 parallel encoding as the default maximum.

The following are some points to note:

- The overall performance of the code needs to be below the hardware design limit, beyond which the set framerate cannot be achieved
- The required Frame Buffer will increase in proportion to the number of channels, and the DRAM usage increases accordingly.

7.2.14 The Encoding Frame Buffer Calculation

The allocation of frame buffers for reference frames and reconstructed frames in video encoding can be done using two methods: PrivateVB pool and UserVB pool.

In the PrivateVB pool method, the VENC creates a private VB pool as the buffer for reference and reconstructed frames when creating an encoding channel.

In the UserVB pool method (supported only in H.264/H.265 encoding), the reference and reconstructed frame buffers are not allocated during the creation of an encoding channel.

Instead, the user can use the CVI_VB_CreatePool interface to create the VB pool.

A video buffer VB pool can be created and then a specific encoding channel can be bound to a fixed video buffer VB pool by calling the interface CVI_VENC_AttachVbPool.

The two methods can be selected by setting the corresponding module parameters through the CVI_VENC_SetModParam interface.

Setting the module parameter to VB_SOURCE_PRIVATE indicates the use of the encoding PrivateVB pool mode, while setting it to VB_SOURCE_USER indicates the use of the encoding UserVB pool mode.

In UserVB mode, the maximum memory consumption for encoding frames is as follows

	H.264	H.265
FrameSize	$(\text{align}(\text{width}, 32)) \times (\text{align}(\text{height}, 32)) \times 2 \times 1.5$	$\text{align}((\text{width} \times \text{height} \times 1.5), 4096) \times 2$

7.3 API Reference

The video encoding module mainly provides the creation and destruction of the video coding channel, the reset of the video coding channel, the start and stop of receiving images, the setting and acquisition of the attributes of the coding channel, the acquisition and release of the code stream and other functions.

The following APIs are available:

- *CVI_VENC_CreateChn*: Create an encoding channel.
- *CVI_VENC_DestroyChn*: Destroy the encoding channel.
- *CVI_VENC_StartRecvFrame*: Enable the encoding channel to receive the input image.
- *CVI_VENC_StopRecvFrame*: Stop the encoding channel from receiving the input image.
- *CVI_VENC_QueryStatus*: Query the encoding channel status.
- *CVI_VENC_SetChnAttr*: Set the encoding attributes of the encoding channel.
- *CVI_VENC_GetChnAttr*: Get the encoding attributes of the encoding channel.
- *CVI_VENC_GetStream*: Get encoded Bitstream.

- *CVI_VENC_ReleaseStream*: Releases encoded Bitstream cache.
- *CVI_VENC_SendFrame*: Allow the user to send the RAW image for encoding.
- *CVI_VENC_GetFd*: Get the device file handle corresponding to the encoding channel.
- *CVI_VENC_CloseFd*: Close the device file handle corresponding to the encoding channel.
- *CVI_VENC_SetJpegParam*: Configure the set of JPEG encoding parameters.
- *CVI_VENC_GetJpegParam*: Get the set of JPEG encoding parameters.
- *CVI_VENC_SetRcParam*: Set advanced parameters for channel rate control
- *CVI_VENC_GetRcParam*: Get advanced parameters for channel rate control.
- *CVI_VENC_SetChnParam*: Set Venc channel parameters.
- *CVI_VENC_GetChnParam*: Get Venc channel parameters.
- *CVI_VENC_RequestIDR*: Request IDR frame.
- *CVI_VENC_SetRoiAttr*: Set the ROI (Region of Interest) encoding configuration for an encoding channel.
- *CVI_VENC_GetRoiAttr*: Get the ROI (Region of Interest) encoding configuration for an encoding channel.
- *CVI_VENC_SetRefParam*: Set the advanced frame skipping reference parameters of H.264/H.265 encoding channel.
- *CVI_VENC_GetRefParam*: Get the advanced frame skipping reference parameters of H.264/H.265 encoding channel.
- *CVI_VENC_SetFrameLostStrategy*: Set the frame loss strategy configuration when the instantaneous bit rate exceeds the threshold.
- *CVI_VENC_GetFrameLostStrategy* : Get the frame loss strategy configuration when the instantaneous bit rate exceeds the threshold.
- *CVI_VENC_SetModParam*: Set encoding-related module parameters.
- *CVI_VENC_GetModParam*: Get encoding-related module parameters.
- *CVI_VENC_AttachVbPool*: Bind the encoding channel to a video buffer VB pool.
- *CVI_VENC_DetachVbPool*: Unbind the encoding channel from a video buffer VB pool.
- *CVI_VENC_ResetChn*: Reset the encoding channel.
- *CVI_VENC_GetH264Entropy*: Get H.264 entropy coding information
- *CVI_VENC_SetH264Entropy*: Set H.264 entropy coding information
- *CVI_VENC_InsertUserData*: Insert user data
- *CVI_VENC_GetCuPrediction*: Get CU prediction information
- *CVI_VENC_SetCuPrediction*: Set CU prediction information
- *CVI_VENC_GetH264Trans*: Get H.264 coding channel transform and quantization attributes
- *CVI_VENC_SetH264Trans*: Set H.264 coding channel transform and quantization attributes
- *CVI_VENC_GetH265Trans*: Get H.265 coding channel transform and quantization attributes
- *CVI_VENC_SetH265Trans*: Set H.265 coding channel transform and quantization attributes

7.3.1 CVI_VENC_CreateChn

【Description】

Create an encoding channel

【Syntax】

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

【Parameter】

Parameter	Description	Input/Output
VeChn	VENC Channel ID	Input
pstAttr	VENC_CHN_ATTR_S attribute pointer	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: cvi_comm_venc.h、cvi_venc.h
- Library files: libvenc.a

【Note】

- The channel attributes are divided into three structures:

```
typedef struct _VENC_CHN_ATTR_S {
    VENC_ATTR_S stVencAttr;///The attribute of video encoder
    VENC_RC_ATTR_S stRcAttr;///The attribute of bitrate control
    VENC_GOP_ATTR_S stGopAttr;} VENC_CHN_ATTR_S;
```

- VENC_ATTR_S: Determine the encoding property, which determines the encoding protocol and assigns values to set the corresponding width and height.
- VENC_RC_ATTR_S: Property of the rate control module, which sets the corresponding sub-structure based on the encoding settings and rate control mode (CBR, VBR, AVBR, FIXQP).
- VENC_GOP_ATTR_S :
GOP type attribute, encoding GOP type (encoding single reference frame, P frame, GOP type).
- The recommended coding width and height are 1920x1080 (1080P)、1280x720 (720P) .

【Example】

User can refer to SAMPLE_COMM_VENC_SetChnAttr in sample_common_venc.c for the settings for the corresponding property pointer.

【Related Topic】

- [CVI_VENC_DestroyChn](#)

7.3.2 CVI_VENC_DestroyChn

【Description】

Destroy an encoding channel

【Syntax】

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

【Parameter】

Parameter	Description	Input/Output
VeChn	VENC Channel ID	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: cvi_comm_venc.h、cvi_venc.h
- Library files: libvenc.a

【Note】

Destroying a nonexistent channel will return a failure.

【Example】

See sample_common_venc.c. The frame must be stopped before destroying the channel.

```
CVI_S32 SAMPLE_COMM_VENC_Stop(VENC_CHN VencChn)
{
    CVI_S32 s32Ret;
    //stop transmission of venc frame data reception
    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;
    }

    //stop thread execution
    if (gs_VencTask[VencChn] != 0) {
        pthread_join(gs_VencTask[VencChn], CVI_NULL);
        CVI_VENC_SYNC("GetVencStreamProc done\n");
        gs_VencTask[VencChn] = 0;
    }

    //destroy the venc channel
    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;
}
```

【Related Topic】

- [CVI_VENC_CreateChn](#)

7.3.3 CVI_VENC_ResetChn**【Description】**

Reset the channel.

【Syntax】

```
CVI_S32 CVI_VENC_ResetChn(VENC_CHN VeChn);
```

【Parameter】

Parameter	Description	Input/Output
VeChn	Channel ID	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: cvi_comm_venc.h, cvi_venc.h
- Library files: libvenc.a

【Note】

- Resetting a non-existing channel will return failure CVI_FAILURE.
- If a channel is reset without stopping receiving images, it will return a failure.

【Example】

- None

7.3.4 CVI_VENC_StartRecvFrame**【Description】**

Starts an encoding channel to receive input images and allows specifying the number of frames to receive. Once the specified number of frames is reached, the channel automatically stops receiving images.

【Syntax】

```
CVI_S32 CVI_VENC_StartRecvFrame(VENC_CHN VeChn, const VENC_RECV_PIC_PARAM_S *pstRecvParam);
```

【Parameter】

Parameter	Description	Input/Output
VeChn	Channel ID	Input
pstRecvParam	Pointer to a structure that specifies the number of image frames to be received.	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: cvi_comm_venc.h, cvi_venc.h
- Library files: libvenc.a

【Note】

- If the channel is not created, failure will be returned.
- The image coding can be received only after the encoder is turned on.

【Example】

- Refer to SAMPLE_COMM_VENC_Start function in sample_common_venc.c.

```
CVI_S32 s32Ret;
VENC_RECV_PIC_PARAM_S stRecvParam;

//create venc channel
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;
}

//setting venc bindmode or not
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);
}

// Specify the number of frames to receive and allow frames to be received
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;
```

【Related Topic】

- [CVI_VENC_CreateChn](#)

7.3.5 CVI_VENC_StopRecvFrame

【Description】

Stop the encoding channel from receiving the input image.

【Syntax】

```
CVI_S32 CVI_VENC_StopRecvFrame(VENC_CHN VeChn);
```

【Parameter】

Parameter	Description	Input/Output
VeChn	Channel ID	input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: `cvi_comm_venc.h`, `cvi_venc.h`
- Library files: `libvenc.a`

【Note】

- `CVI_VENC_StopRecvFrame` needs to be called before destroying the channel.
- If the channel doesn't exist or is already destroyed, failure will be returned.
- Calling this interface will only stop receiving the RAW data encoding, the code stream buffer will not be cleared.
- This interface is used to stop the encoding channel from receiving images for encoding.

It must be called before the encoding channel is destroyed or reset.

【Example】

- Refer to `SAMPLE_COMM_VENC_Stop` function in `sample_common_venc.c`.

【Related Topic】

- [*CVI_VENC_DestroyChn*](#)

7.3.6 CVI_VENC_QueryStatus

【Description】

Query the encoding channel status.

【Syntax】

```
CVI_S32 CVI_VENC_QueryStatus(VENC_CHN VeChn, VENC_CHN_STATUS_S *pstStatus);
```

【Parameter】

Parameter	Description	Input/Output
VeChn	VENC Channel ID	Input
pstStatus	Pointer to the status of the encoding channel	Output

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: cvi_comm_venc.h, cvi_venc.h
- Library files: libvenc.a

【Note】

- If the channel is not created, failure will be returned.
- In the encoding channel status structure, u32CurPacks represents the number of packets of the current frame.

Before calling CVI_VENC_GetStream, it should be ensured that u32CurPacks is greater than 0.

7.3.7 CVI_VENC_SetChnAttr

【Description】

Set the encoding channel attributes.

【Syntax】

```
CVI_S32 CVI_VENC_SetChnAttr(VENC_CHN VeChn, const VENC_CHN_ATTR_S *pstChnAttr);
```

【Parameter】

Parameter	Description	Input/Output
VeChn	VENC Channel ID	Input
pstChnAttr	Encoding channel attribute pointer	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: cvi_comm_venc.h, cvi_venc.h
- Library files: libvenc.a

【Note】

None.

【Example】

7.3.8 CVI_VENC_GetChnAttr

【Description】

Get the encoding channel attributes.

【Syntax】

```
CVI_S32 CVI_VENC_GetChnAttr(VENC_CHN VeChn, VENC_CHN_ATTR_S *pstChnAttr);
```

【Parameter】

Parameter	Description	Input/Output
VeChn	VENC Channel number.	Input
pstChnAttr	Encoding channel attribute pointer.	Output

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: cvi_comm_venc.h、cvi_venc.h
- Library files: libvenc.a

【Note】

- Get the attributes of an uncreated channel will return a failure.

【Example】

7.3.9 CVI_VENC_GetStream

【Description】

Get the encoded stream.

【Syntax】

```
CVI_S32 CVI_VENC_GetStream(VENC_CHN VeChn, VENC_STREAM_S *pstStream, CVI_S32_↵
↵S32MilliSec);
```

【Parameter】

Parameter	Description	Input/Output
VeChn	VENC Channel number.	Input
pstFrame	Code stream structure pointer VENC_STREAM_S。	Output
S32MilliSec	Image sending timeout Value range: [-1, + 1) -1: Blocking. 0: Non-blocking. Greater than 0: Timeout.	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: `cvi_comm_venc.h`, `cvi_venc.h`
- Library files: `libvenc.a`

【Note】

- The Venc channel must be created first or this function will return an error.
- In non-bind mode conditions, the `CVI_VENC_SendFrame` must have been called, otherwise the `CVI_VENC_GetStream` will not be able to obtain the bitstream structure.
- `CVI_VENC_GetStream` can be paired with `CVI_VENC_GetChnAttr` before calling, `CVI_VENC_QueryStatus` calls ensure the correctness of the current encoding channel.
- Bit Stream Structure `VENC_STREAM_S` contains four parts:
 - The stream packet information pointer `pstPack` points to a set of `VENC_PACK_S` memory spaces allocated by the caller.
 - Note: `pstPack` space must be given space before calling `CVI_VENC_GetStream`, otherwise an error will be returned.

【Example】

Refer to `sample_venc_lib.c` `SAMPLE_VENC_GetVencStreamProc`:

```
while (pVencChnCtx->chnStat == CHN_STAT_START) {
    VENC_CHN_STATUS_S stStat;
    //bind mode mode check
    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;
        }
    }

    // Start getting venc stream based on venc channel state
    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, VencChn = %d, s32Ret = %d\n",
                        VencChn, s32Ret);
            break;
        }
        if (!stStat.u32CurPacks) {
            CVI_VENC_ERR("u32CurPacks = NULL!\n");
        }
    }
}
```

(continues on next page)

(continued from previous page)

```

    break;
}

// pstPack space needs to be given first
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;
}

//Users can archive or give the removed stStream to the upper
↪ 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

【Description】

Release the encoded stream cache.

【Syntax】

```
CVI_S32 CVI_VENC_ReleaseStream(VENC_CHN VeChn, VENC_STREAM_S *pstStream);
```

【Parameter】

Parameter	Description	Input/Output
VeChn	VENC Channel number.	Input
pstStream	VENC_STREAM_S property pointer.	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: `cvi_comm_venc.h`, `cvi_venc.h`

- Library file: libvenc.a

【Note】

- Verify that the venc channel has been created.
- Please confirm that if CVI_VENC_ReleaseStream is called early after CVI_VENC_GetStream, CVI_VENC_GetStream will not have the right stream guarantee.
- The user must release the obtained bitstream buffer in a timely manner after getting the bitstream, otherwise it may cause the buffer to be full and affect the encoder's encoding.

【Example】

- please refer to the example of CVI_VENC_GetStream

7.3.11 CVI_VENC_SendFrame

【Description】

Send out the image for encoding.

【Syntax】

```
CVI_S32 CVI_VENC_SendFrame(VENC_CHN VeChn, const VIDEO_FRAME_INFO_S *pstFrame, CVI_
↪S32 S32MilliSec);
```

【Parameter】

Parameter	Description	Input/Output
VeChn	VENC Channel number	Input
pstFrame	VIDEO_FRAME_INFO_S property pointer	Input
S32MilliSec	Image sending Timeout Value range: [-1, + 1) • 1: Blocking. 0: Non-blocking. Greater than 0: Timeout.	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: cvi_comm_venc.h, cvi_venc.h
- Library files: libvenc.a

【Note】

- This interface enables users to send images to the encoding channel for encoding.
- Call this interface to send an image.

The user needs to ensure that the encoding channel has been created and is ready to receive input images.

【Example】

Refer to SAMPLE_VENC_GetVencStreamProc in sample_venc_lib.c:

```

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, VencChn = %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) {

```

(continues on next page)

(continued from previous page)

```

    CVI_VENC_ERR("CVI_VENC_ReleaseStream, s32Ret = %d\n", s32Ret);
    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;
    }
}
}
}

```

【Related Topic】

- [CVI_VENC_QueryStatus](#)
- [CVI_VENC_GetStream](#)

7.3.12 CVI_VENC_GetFd**【Description】**

Get the device file handle corresponding to the encoding channel.

【Syntax】

```
CVI_S32 CVI_VENC_GetFd(VENC_CHN VeChn);
```

【Parameter】

Parameter	Description	Input/Output
VeChn	VENC Channel number	Input

【Return Value】

Return Value	Description
Greater than 0	Success.
Less than or equal to 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: `cvi_comm_venc.h`, `cvi_venc.h`
- Library files: `libvenc.a`

【Note】

- none

【Example】

- none

【Related Topic】

- [CVI_VENC_CloseFd](#)

7.3.13 CVI_VENC_CloseFd

【Description】

Close the channel handle.

【Syntax】

```
CVI_S32 CVI_VENC_CloseFd(VENC_CHN VeChn);
```

【Parameter】

Parameter	Description	Input/Output
VeChn	VENC Channel number.	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: `cvi_comm_venc.h`, `cvi_venc.h`
- Library files: `libvenc.a`

【Note】

- This API cannot be used when monitoring Stream using select function.

【Example】

- none

【Related Topic】

- [CVI_VENC_GetFd](#)

7.3.14 CVI_VENC_SetJpegParam

【Description】

Set advanced parameter for JPEG encoding protocol

【Syntax】

```
CVI_S32 CVI_VENC_SetJpegParam(VENC_CHN VeChn, const VENC_JPEG_PARAM_S *pstJpegParam);
```

【Parameter】

Parameter	Description	Input/Output
VeChn	VENC Channel number.	Input
pstJpegParam	Pointer to the collection of advanced parameters for the JPEG protocol encoding channel.	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: cvi_comm_venc.h, cvi_venc.h
- Library files: libvenc.a

【Note】

- This interface is used to set advanced parameters for JPEG protocol encoding channels.
- This interface is used after the Venc channel is created.
- Advanced parameters Composition
 - u32Qfactor: The range of the quantization table factor is [1, 99].
The larger the value of u32Qfactor, the smaller the quantization coefficients in the quantization table, which leads to better image quality and lower compression ratio.
Conversely, the smaller the value of u32Qfactor, the larger the quantization coefficients in the quantization table, which leads to poorer image quality and higher compression ratio.
Note that u32Qfactor=50 is a reserved custom setting item and is not currently supported.
 - u32MCUPerECS: The number of MCU in each ECS.
When the system mode u32MCUPerECS = 0, all MCUs in the current frame are encoded into one ECS.
The value of u32MCUPerECS should be no less than 0 and no greater than $(\text{picwidth}+15)>>4 * (\text{picheight}+15)>>4 * 2$.
- It is recommended that users call this interface before starting the encoding after creating the channel,
reducing the number of calls in encoding process.
It is recommended that the user call the CVI_VENC_GetJpegParam interface before calling this interface to obtain the JpegParam configuration of the current coding channel, and then set it.

【Example】

- Please refer to the SAMPLE_COMM_VENC_SetJpegParam function in the sample_common_venc.


```

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

【Description】

Gets the advanced parameter configuration of the JPEG protocol encoding channel.

【Syntax】

```
CVI_S32 CVI_VENC_GetJpegParam(VENC_CHN VeChn, VENC_JPEG_PARAM_S *pstJpegParam);
```

【Parameter】

Parameter	Description	Input/Output
VeChn	VENC Channel number.	Input
pstJpegParam	Pointer to the collection of advanced parameters for the JPEG protocol encoding channel.	Output

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: cvi_comm_venc.h、cvi_venc.h
- Library files: libvenc.a

【Note】

- This interface is called before the encoding channel is created and the encoding channel is destroyed.
- It is recommended that users call this interface before starting the encoding after creating the channel, reducing the number of calls in encoding process.

【Example】

- please refer to the example of CVI_VENC_SetJpegParam

7.3.16 CVI_VENC_SetRcParam

【Description】

Set advanced parameters for channel rate control.

【Syntax】

```
CVI_S32 CVI_VENC_SetRcParam(VENC_CHN VeChn, const
VENC_RC_PARAM_S *pstRcParam);
```

【Parameter】

Parameter	Description	Input/Output
VeChn	VENC Channel number.	Input
pstRcParam	Advanced parameters for in channel rate control	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: cvi_comm_venc.h、cvi_venc.h
- Library files: libvenc.a

【Note】

- This interface is called before the encoding channel is created and the encoding channel is destroyed.
- This function is called after CVI_VENC_CreateChn and before CVI_VENC_SetChnParam.
- You are advised to invoke the CVI_VENC_GetRcParam interface to obtain the RC advanced parameters, modify them, and then invoke this interface to set the advanced parameters.

【Example】

- refer to the SAMPLE_COMM_VENC_SetRcParam function in sample_common_venc.c :

```
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;
```

(continues on next page)

(continued from previous page)

```

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

【Description】

Get the advanced parameters for channel rate control.

【Syntax】

```
CVI_S32 CVI_VENC_GetRcParam(VENC_CHN VeChn, VENC_RC_PARAM_S *pstRcParam);
```

【Parameter】

Parameter	Description	Input/Output
VeChn	VENC Channel number.	Input
pstRcParam	Advanced parameters for channel rate control	Output

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: `cvi_comm_venc.h`, `cvi_venc.h`
- Library files: `libvenc.a`

【Note】

- This interface is called after the encoding channel is created and before the encoding channel is destroyed.
- Before invoking this interface, you are advised to invoke the `CVI_VENC_GetChnParam` interface to obtain the parameter configuration of the front channel and then set it.

【Example】

None.

7.3.18 CVI_VENC_SetChnParam

【Description】

Set channel parameters.

【Syntax】

```
CVI_S32 CVI_VENC_SetChnParam(VENC_CHN VeChn, const VENC_CHN_PARAM_S *pstChnParam);
```

【Parameter】

Parameter	Description	Input/Output
<code>VeChn</code>	VENC Channel number.	Input
<code>pstChnParam</code>	Channel parameters of Venc.	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: `cvi_comm_venc.h`, `cvi_venc.h`
- Library files: `libvenc.a`

【Note】

- This interface is called after the encoding channel is created and before the encoding channel is destroyed.
- This function is called after `CVI_VENC_CreateChn` and before `CVI_VENC_SetChnParam`.
- The API parameter of this function supports the setting of venc CROP.

The parameter used is `VENC_CHN_PARAM_S`, `stCropCfg`.

The corresponding x, Y axes, width and height can be set.

【Example】

```
CVI_S32 SAMPLE_COMM_VENC_SetChnParam(chnInputCfg *pIc, VENC_CHN VencChn)
{
    VENC_CHN_PARAM_S stChnParam, *pstChnParam = &stChnParam;
    CVI_S32 s32Ret = CVI_SUCCESS;
```

(continues on next page)

(continued from previous page)

```

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

【Description】

Get the channel parameters.

【Syntax】

```
CVI_S32 CVI_VENC_GetChnParam(VENC_CHN VeChn, VENC_CHN_PARAM_S *pstChnParam);
```

【Parameter】

Parameter	Description	Input/Output
VeChn	VENC Channel number	Input
pstChnParam	Pointer to VENC channel parameter	Output

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: cvi_comm_venc.h, cvi_venc.h
- Library files: libvenc.a

【Note】

- None.

【Example】

- None.

7.3.20 CVI_VENC_RequestIDR

【Description】

Request IDR frame.

【Syntax】

```
CVI_S32 CVI_VENC_RequestIDR(VENC_CHN VeChn, CVI_BOOL bInstant);
```

【Parameter】

Parameter	Description	Input/Output
VeChn	VENC Channel number.	Input
bInstant	Enable immediate encoding of IDR frame.	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: cvi_comm_venc.h, cvi_venc.h
- Library files: libvenc.a

【Note】

- If the channel is not created, a failure will be returned.
- After receiving the IDR frame request, regardless of bInstant =0 or not, the IDR frame is immediately encoded, which is not subject to frame rate control.
- Request for IDR frame, only supported by H.264/H.265 encoding protocols.
- This interface is not affected by frame rate control and will produce an IDR frame each time it is called. Frequent calls to this interface can affect the stability of the stream frame rate and bit rate.
- Multiple interface calls before the next frame encoding will only produce one IDR.

It does not work if the original frame is already an IDR frame.

- When advanced frame skipping is enabled, the request IDR frame may be delayed.

【Example】

- None.

7.3.21 CVI_VENC_SetRoiAttr

【Description】

Set the ROI attribute of H.264/H.265 channel.

【Syntax】

```
CVI_S32 CVI_VENC_SetRoiAttr(VENC_CHN VeChn, const VENC_ROI_ATTR_S *pstRoiAttr);
```

【Parameter】

Parameter	Description	Input/Output
VeChn	VENC Channel number.	Input
pstRoiAttr	ROI regional parameters.	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: `cvi_comm_venc.h`, `cvi_venc.h`
- Library files: `libvenc.a`

【Note】

- `u32Index`:

Each channel supports setting up to 8 ROI regions, which are managed according to the index number from 0 to 7.

The `u32Index` parameter represents the index number of the ROI region set by the user. If there are duplicate regions, the ROI regions are prioritized in order of the index number from 0 to 7.

- `bEnable`: Specifies whether the current ROI region is enabled.
- `bAbsQp`: Specifies whether the current ROI region uses absolute QP or relative QP mode.
- `s32Qp`:

When `bAbsQp` is `CVI_True`, `s32Qp` is the QP value set for the ROI area.

When `bAbsQp` is `CVI_False`, `s32Qp` is the QP value set for the internal rate control of the ROI area plus the QP offset value.

- `stRect`:

Specifies the location coordinates and size of the current ROI region.

ROI area must be within the image range.

- By default, the system does not have ROI area enabled.

The user must set and call this interface to start ROI after the coding channel is created and before the coding channel is destroyed.

When this interface is called during encoding, it will take effect at the next frame.

- It is recommended that users call this interface before starting the encoding after creating the channel, reducing the number of calls in encoding process.

It is recommended that users call the `CVI_VENC_GetRoiAttr` interface before calling this interface to obtain the ROI configuration of the current channel before setting.

- After setting the interface, if the current frame is judged to be pskip frame, the pskip frame effect takes precedence.
- When the rate control mode is not Fixed QP mode, the ROI area can be configured.
- When ROI is enabled in H.264, macroblock-level bitrate control is disabled.
- In absolute QP mode, the actual encoded QP of macroblocks may differ slightly from the configured QP due to the bitrate control adjustment of macroblock QP.

In absolute Qp mode, because the rate control adapts the macroblock QP, there may be some differences between the actual coded QP and the set QP.

【Example】

- None.

7.3.22 CVI_VENC_GetRoiAttr

【Description】

Get the ROI attribute of H.264/H.265 channel.

【Syntax】

```
CVI_S32 CVI_VENC_GetRoiAttr(VENC_CHN VeChn, CVI_U32 u32Index, VENC_ROI_ATTR_S_
↪ *pstRoiAttr);
```

【Parameter】

Parameter	Description	Input/Output
VeChn	VENC Channel number.	Input
u32Index	ROI zone index.	Input
pstRoiAttr	ROI zone parameters.	Output

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: cvi_comm_venc.h、cvi_venc.h
- Library files: libvenc.a

【Note】

- According to the u32Index index, the ROI region configuration is obtained.
- The user must set and call this interface after the code channel is created and before the code channel is destroyed
- It is recommended that users call CVI_VENC_SetRoiAttr interface before calling it to obtain the ROI configuration of the current channel before setting.

【Example】

- None.

7.3.23 CVI_VENC_SetRefParam

【Description】

Set the advanced frame skipping reference parameters of H.264/H.265 encoding channel.

【Syntax】

```
CVI_S32 CVI_VENC_SetRefParam(VENC_CHN VeChn,
↪ const VENC_REF_PARAM_S *pstRefParam);
```

【Parameter】

Parameter	Description	Input/Output
VeChn	VENC Channel number.	Input
pstRefParam	H.264/H.265 coding channel advanced frame skipping reference parameters.	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: cvi_comm_venc.h, cvi_venc.h
- Library files: libvenc.a

【Note】

- Both H.264 and H.265 support 1-x, 2-x and 4-x frame skipping reference modes. The default mode is 1-x frame skipping reference mode (no frame skipping).
- This interface needs to be called after creating the channel and before starting encoding to avoid calling during the encoding process.
- The configuration of 1x frame skipping reference mode is : bEnablePred = HI_TRUE, u32Enhance = 0, u32Base = 1
- The configuration of 2x frame skipping reference mode is : bEnablePred = HI_TRUE, u32Enhance = 1, u32Base = 1.
- The configuration of 4x frame skipping reference mode is : bEnablePred = HI_TRUE, u32Enhance = 1, u32Base = 2
- When the Gop Mode of the channel is set to NormalP, the u32Gop of 2x frame skipping reference mode should be a multiple of 2, and the u32Gop of 4x frame skipping reference mode should be a multiple of 4
- When the Gop Mode of the channel is set to SmartP, the u32Gop and u32BgInterval of 2x frame skipping reference mode should be a multiple of 2; the u32Gop and u32BgInterval of 4x frame skipping reference mode should be a multiple of 4

7.3.24 CVI_VENC_GetRefParam**【Description】**

Get the advanced frame skipping reference parameters of H.264/H.265 encoding channel.

【Syntax】

```
CVI_S32 CVI_VENC_GetRefParam(VENC_CHN VeChn, VENC_REF_PARAM_S *pstRefParam);
```

【Parameter】

Parameter	Description	Input/Output
VeChn	VENC Channel number.	Input
pstRefParam	H.264/H.265 coding channel advanced frame skipping reference parameters.	Output

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: `cvi_comm_venc.h`, `cvi_venc.h`
- Library files: `libvenc.a`

【Note】

- None.

【Example】

- None.

7.3.25 CVI_VENC_SetFrameLostStrategy

【Description】

Set the frame dropping strategy when the instantaneous bitrate of the encoding channel exceeds the threshold.

【Syntax】

```
CVI_S32 CVI_VENC_SetFrameLostStrategy(VENC_CHN VeChn, const VENC_FRAMELOST_STRATEGY_S *pstFrmLostParam);
```

【Parameter】

Parameter	Description	Input/Output
VeChn	VENC Channel number.	Input
pstFrmLostParam	The parameters of frame in dropping strategy	Input/Output

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: `cvi_comm_venc.h`, `cvi_venc.h`
- Library files: `libvenc.a`

【Note】

- The frame dropping strategy is only supported by H.264/H.265 encoding protocols.
- The frame dropping strategy configuration is determined by four parameters
 - `bFrmLostOpen`:
When the bitrate exceeds the threshold, dropping frames is enabled to ensure that the peak value of the bit rate in the interval does not exceed a certain limit.
 - `enFrmLostMode`:
The frame dropping method is only supported for encoding as P-skip frames.
 - `u32FrmLostBpsThr`:
According to the system capacity setting, it is recommended to set at least 1.2 times the bit rate
 - `u32EncFrmGaps`:
Limiting the maximum number of consecutive dropped frames can make the pictures during the dropout period smoother, but the peak value of the interval bit rate may be higher.

Setting it to 0 means that there is no limit on the number of consecutive dropped frames, indicating that frames can be dropped continuously as long as the instantaneous bit rate exceeds the threshold, until the instantaneous bit rate is less than or equal to the threshold.

- Users can choose to enable or disable the frame dropping strategy by calling this interface, and it is disabled by default.
- This interface is called after the code channel is created and before the code channel is destroyed .
- pskip frames are only supported by encoding channels with GOP mode VENC_GOPMODE_NORMALP.

If the OSD is enabled for the current frame and the OSD has been updated, it cannot be encoded as a Pskip frame.

- When frame skipping reference mode is enabled, it is not recommended to turn on frame dropping strategy
- Only supports CBR/VBR rate control modes for H.264/H.265.

【Example】

- None.

7.3.26 CVI_VENC_GetFrameLostStrategy

【Description】

Get the frame dropping strategy when the instantaneous bitrate of the encoding channel exceeds the threshold.

【Syntax】

```
CVI_S32 CVI_VENC_GetFrameLostStrategy(VENC_CHN VeChn, VENC_FRAMELOST_STRATEGY
↪ *pstFrmLostParam);
```

【Parameter】

Parameter	Description	Input/Output
VeChn	VENC Channel number.	Input
pstFrmLostParam	The parameters of frame dropping strategy	Output

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: cvi_comm_venc.h、cvi_venc.h
- Library files: libvenc.a

【Note】

- None.

【Example】

- None.

7.3.27 CVI_VENC_SetModParam

【Description】

Set the module parameters related to encoding.

【Syntax】

```
CVI_S32 CVI_VENC_SetModParam(const VENC_PARAM_MOD_S *pstModParam);
```

【Parameter】

Parameter	Description	Input/Output
pstModParam	Encoding module parameter pointer.	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: cvi_comm_venc.h、cvi_venc.h
- Library files: libvenc.a

【Note】

- This interface can be called before and after channel creation.
- This interface is mainly used to set the corresponding encoding VB pool acquisition method.

Users can set the VB mode through the VB_SOURCE_E type variable in the VENC_PARAM_MOD_S structure.

【Example】

- None.

7.3.28 CVI_VENC_GetModParam

【Description】

Get module parameters related to encoding .

【Syntax】

```
CVI_S32 CVI_VENC_GetModParam(VENC_PARAM_MOD_S *pstModParam);
```

【Parameter】

Parameter	Description	Input/Output
pstModParam	Encoding module parameter pointer.	Output

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: `cvi_comm_venc.h`, `cvi_venc.h`
- Library files: `libvenc.a`

【Note】

- Typically used in conjunction with the `CVI_VENC_SetModParam` function, before making the call.

【Example】

- None.

7.3.29 CVI_VENC_AttachVbPool

【Description】

Bind the encoding channel to a video buffer VB pool.

【Syntax】

```
CVI_S32 CVI_VENC_AttachVbPool(VENC_CHN VeChn, const VENC_CHN_POOL_S *pstPool);
```

【Parameter】

Parameter	Description	Input/Output
VeChn	Channel number.	Input
pstPool	The Id number of the video buffer VB pool.	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: `cvi_comm_venc.h`, `cvi_venc.h`, `cvi_comm_vb.h`
- Library files: `libvenc.a`

【Note】

- You must ensure that the channel has been created, otherwise the error code of `CVI_FAILURE` will be returned.
- The user must call the interface `CVI_VB_CreatePool` creates a visual cache VB pool and then calls the interface `CVI_VENC_AttachVbPool` binds the current encoding channel to a fixed pool of VB.

Multiple encoding channels can be bound to the same VB pool, but the same encoding channel cannot be bound to multiple VB pools.

- `pstPool` must be a valid `PoolId` of the created VB pool, including VB pools that store pictures and VB pools that store picture information.
- Only H.264/H.265 encoding supports UserVB pool mode.

If the current encoding frame storage allocation method is not using the encoding UserVB pool, `CVI_FAILURE` will be returned.

- The user must call this interface in `VB_SOURCE_USER` mode set through `CVI_VENC_SetModParam`.

【Example】

- None.

7.3.30 CVI_VENC_DetachVbPool

【Description】

Unbind the encoding channel from a video cache VB pool .

【Syntax】

```
CVI_S32 CVI_VENC_DetachVbPool(VENC_CHN VeChn);
```

【Parameter】

Parameter	Description	Input/Output
VeChn	Channel number.	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: cvi_comm_venc.h, cvi_venc.h, cvi_comm_vb.h
- Library files: libvenc.a

【Note】

- You must ensure that the channel has been created, otherwise the error code of CVI_FAILURE will be returned.

【Example】

- None.

7.3.31 CVI_VENC_GetH264Entropy

【Description】

Get H.264 entropy coding information

【Syntax】

```
CVI_S32 CVI_VENC_GetH264Entropy(VENC_CHN VeChn, VENC_H264_ENTROPY_STRUCT  
↪ *pstH264EntropyEnc);
```

【Parameter】

Parameter	Description	Input/Output
VeChn	Channel number.	Input
pstH264EntropyEnc	Pointer to entropy coding information structure	Output

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: cvi_comm_venc.h、cvi_venc.h
- Library files: libvenc.a

【Note】

- The channel must be created beforehand, otherwise an error code of CVI_FAILURE will be returned.

【Example】

- None.

7.3.32 CVI_VENC_SetH264Entropy

【Description】

Set H.264 entropy coding information

【Syntax】

```
CVI_S32 CVI_VENC_SetH264Entropy(VENC_CHN VeChn,  const VENC_H264_ENTROPY_SU  
↪*pstH264EntropyEnc);
```

【Parameter】

Parameter	Description	Input/Output
VeChn	Channel number.	Input
pstH264EntropyEnc	Pointer to entropy coding information structure	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: cvi_comm_venc.h、cvi_venc.h
- Library files: libvenc.a

【Note】

- The channel must be created beforehand, otherwise an error code of CVI_FAILURE will be returned.

【Example】

- None.

7.3.33 CVI_VENC_InsertUserData

【Description】

Insert user data

【Syntax】

```
CVI_S32 CVI_VENC_InsertUserData(VENC_CHN VeChn, CVI_U8 *pu8Data, CVI_U32 u32Len);
```

【Parameter】

Parameter	Description	Input/Output
VeChn	Channel number	Input
pu8Data	Data head address	Input
u32Len	Data size	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: cvi_comm_venc.h, cvi_venc.h
- Library files: libvenc.a

【Note】

- The channel must be created beforehand, otherwise an error code of CVI_FAILURE will be returned.

【Example】

- None.

7.3.34 CVI_VENC_GetCuPrediction

【Description】

CU prediction information retrieval

【Syntax】

```
CVI_S32 CVI_VENC_GetCuPrediction(VENC_CHN VeChn,  
VENC_CU_PREDICTION_S *pstCuPrediction);
```

【Parameter】

Parameter	Description	Input/Output
VeChn	Channel number.	Input
pstCuPrediction	Pointer to prediction information structure	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: cvi_comm_venc.h, cvi_venc.h
- Library files: libvenc.a

【Note】

- The channel must be created beforehand, otherwise an error code of CVI_FAILURE will be returned.
- This interface is currently only available for H264 encoding

【Example】

- None.

7.3.35 CVI_VENC_SetCuPrediction

【Description】

Set CU prediction information.

【Syntax】

```
CVI_S32 CVI_VENC_SetCuPrediction(VENC_CHN VeChn, VENC_CU_PREDICTION_S_
↪*pstCuPrediction);
```

【Parameter】

Parameter	Description	Input/Output
VeChn	Channel number.	Input
pstCuPrediction	Pointer to prediction information structure	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: cvi_comm_venc.h, cvi_venc.h
- Library files: libvenc.a

【Note】

- The channel must be created beforehand, otherwise an error code of CVI_FAILURE will be returned.
- This interface is currently only available for H264 encoding

【Example】

- None.

7.3.36 CVI_VENC_GetH264Trans

【Description】

Get H.264 coding channel transform and quantization attributes

【Syntax】

```
CVI_S32 CVI_VENC_GetH264Trans(VENC_CHN VeChn, VENC_H264_TRANS_S *pstH264Trans);
```

【Parameter】

Parameter	Description	Input/Output
VeChn	Channel number.	Input
pstH264Trans	H.264 coding channel transform and quantization attributes	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: cvi_comm_venc.h, cvi_venc.h
- Library files: libvenc.a

【Note】

- The channel must be created beforehand, otherwise an error code of CVI_FAILURE will be returned.
- This interface is currently only available for H264 encoding

【Example】

- None.

7.3.37 CVI_VENC_SetH264Trans

【Description】

Set H.264 coding channel transform and quantization attributes

【Syntax】

```
CVI_S32 CVI_VENC_SetH264Trans(VENC_CHN VeChn, const VENC_H264_TRANS_S *pstH264Trans);
```

【Parameter】

Parameter	Description	Input/Output
VeChn	Channel number.	Input
pstH264Trans	H.264 coding channel transform and quantization attributes	Output

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: cvi_comm_venc.h, cvi_venc.h
- Library files: libvenc.a

【Note】

- The channel must be created beforehand, otherwise an error code of CVI_FAILURE will be returned.
- This interface is currently only available for H264 encoding

【Example】

- None.

7.3.38 CVI_VENC_SetH265Trans**【Description】**

Set H.265 coding channel transform and quantization attributes

【Syntax】

```
CVI_S32 CVI_VENC_SetH265Trans(VENC_CHN VeChn,
    const VENC_H265_TRANS_S *pstH265Trans);
```

【Parameter】

Parameter	Description	Input/Output
VeChn	Channel number.	Input
pstH265Trans	H.265 coding channel transform and quantization attributes	Output

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: cvi_comm_venc.h, cvi_venc.h
- Library files: libvenc.a

【Note】

- The channel must be created beforehand, otherwise an error code of CVI_FAILURE will be returned.
- This interface is currently only available for H265 encoding

【Example】

- None.

7.3.39 CVI_VENC_GetH265Trans

【Description】

Get H.265 coding channel transform and quantization attributes

【Syntax】

```
CVI_S32 CVI_VENC_SetH264Trans(VENC_CHN VeChn, const VENC_H264_TRANS_S *pstH264Trans);
```

【Parameter】

Parameter	Description	Input/Output
VeChn	Channel number.	Input
pstH265Trans	H.265 coding channel transform and quantization attributes	Output

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: cvi_comm_venc.h、cvi_venc.h
- Library files: libvenc.a

【Note】

- The channel must be created beforehand, otherwise an error code of CVI_FAILURE will be returned.
- This interface is currently only available for H265 encoding

【Example】

- None.

7.4 Data Types

Relevant data types and data structures are defined as follows:

- *VENC_MAX_CHN_NUM*: Define the maximum number of channels.
- *VENC_CHN_PARAM_S*: Define the Venc channel parameter structure.
- *VENC_PACK_S*: Define frame stream packet structure .
- *VENC_STREAM_S*: Define the frame stream type structure.
- *VENC_ATTR_S*: Define the encoder attribute structure.
- *VENC_GOP_ATTR_S*: Define the structure of GOP mode type.
- *VENC_GOP_NORMALP_S*: Define the normalP structure
- *VENC_GOP_SMARTP_S*: Define the SmartP structure
- *VENC_CHN_ATTR_S*: Define the encoding channel attribute structure.
- *VENC_RECV_PIC_PARAM_S*: Define the frame number structure that the coding channel continuously receives and encodes.

- *VENC_CHN_STATUS_S*: Define the state structure of the encoding channel.
- *VENC_JPEG_PARAM_S*: Define a set of JPEG encoding parameters.
- *VENC_RC_ATTR_S*: Define the rate controller attribute of the coding channel.
- *VENC_H264_CBR_S*: Define the CBR attribute structure of H.264 coding channel.
- *VENC_H264_VBR_S*: Define the VBR attribute structure of H.264 coding channel.
- *VENC_H264_QVBR_S*: Define the QVBR attribute structure of H.264 coding channel.
- *VENC_H264_FIXQP_S*: Define the Fixqp attribute structure of H.264 coding channel.
- *VENC_H264_QPMAP_S*: Define the QPMAP attribute structure of H.264 coding channel.
- *VENC_MJPEG_FIXQP_S*: Define the Fixqp attribute structure of MJPEG encoding channel.
- *VENC_MJPEG_CBR_S*: Define the CBR attribute structure of MJPEG encoding channel.
- *VENC_MJPEG_VBR_S*: Define the VBR attribute structure of MJPEG encoding channel.
- *VENC_H265_CBR_S*: Define the CBR attribute structure of H.265 coding channel.
- *VENC_H265_VBR_S*: Define the VBR attribute structure of H.265 coding channel.
- *VENC_H265_AVBR_S*: Define the AVBR attribute structure of H.265 coding channel.
- *VENC_H265_QVBR_S*: Define the QVBR attribute structure of H.265 coding channel.
- *VENC_H265_FIXQP_S*: Define the Fixqp attribute structure of H.265 coding channel.
- *VENC_H265_QPMAP_S*: Define the QPMAP attribute structure of H.265 coding channel.
- *VENC_RC_PARAM_S*: Define the advanced rate control parameters of the coding channel.
- *VENC_CHN_POOL_S*: Define the VB pool structure which is bound by encoding channel
- *VENC_H264_ENTROPY_S*: Define H.264 entropy coding information structure
- *VENC_CU_PREDICTION_S*: Define CU prediction information structure

7.4.1 VENC_MAX_CHN_NUM

【Description】

Define the maximum number of channels for encoding

【Syntax】

```
#define VENC_MAX_CHN_NUM 16
```

【Member】

【Note】

Because the maximum number of channels is related to memory allocation, it is not open for expansion at present.

【Related Data Type and Interface】

None.

7.4.2 VENC_CHN_PARAM_S

【Description】

Define the Venc channel parameter structure.

【Syntax】

```
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;
```

【Member】

Member	Description
bColor2Grey	Reserved, not used yet
u32Priority	Reserved, not used yet
u32MaxStrmCnt	Reserved, not used yet
u32PollWakeUpFrmCnt	Reserved, not used yet
stCropCfg	Channel intercept (Crop) parameter
stFrameRate	Channel frame rate control parameters

【Note】

None.

【Related Data Type and Interface】

- CVI_VENC_SetChnParam

7.4.3 VENC_PACK_S

【Description】

Define the frame stream packet structure.

【Syntax】

```
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;
    VENC_PACK_INFO_S stPackInfo[8];
} VENC_PACK_S;
```

【Member】

Member	Description
u64PhyAddr	The first physical address of the bitstream.
pu8Addr	The first virtual address of the bitstream.
u32Len	The length of the stream packet.
DataType	Stream type, support H.264/JPEG/H.265 protocol type packet.
bFrameEnd	End of frame identifier. CVI_TRUE: the stream packet is the last packet of the frame. CVI_FALSE: the stream packet is not the last packet of the frame.
u32Offset	The offset between the valid data in the stream packet and the first address pu8Addr of the stream packet.
u64PTS	Time stamp. Unit: us.
u32DataNum	The number of other types of code stream packets in the current code stream packet (the type of current packet is specified by DataType).
stPackInfo[8]	The current stream packet contains other types of stream packet information.

【Note】

None.

【Related Data Type and Interface】

- [VENC_STREAM_S](#)

7.4.4 VENC_STREAM_S

【Description】

Define the frame stream type structure.

【Syntax】

```
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;
    };

    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;
```

【Member】

Member	Description
pstPack	Frame stream packet structure.
u32PackCount	The number of all packets in a frame stream.
u32Seq	Code stream sequence number.
stH264Info/stJpegInfo/stH265Info/stProresInfo	Stream information.
stAdvanceH264Info/stAdvanceJpegInfo/stAdvanceH265Info/stAdvanceProresInfo	Bit stream advanced information.

【Note】

None.

【Related Data Type and Interface】

- [VENC_PACK_S](#)

7.4.5 VENC_GOP_ATTR_S

【Description】

Define the encoder GOP attribute structure.

【Syntax】

```
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;
```

【Member】

Member	Description
enGopMode	Encoding GOP type.
stNormalP	Encoding single reference frame P frame GOP attribute structure.
stDualP	Reserved, not used yet
stSmartP	Encoding intelligent P-frame GOP attribute structure
stAdvSmartP	Reserved, not used yet
stBipredB	Reserved, not used yet

【Note】

None.

【Related Data Type and Interface】

- [CVI_VENC_CreateChn](#)
- [VENC_STREAM_S](#)

7.4.6 VENC_GOP_NORMALP_S

【Description】

Define the encoder NormalP GOP attribute structure

【Syntax】

```
typedef struct _VENC_GOP_NORMALP_S {
    CVI_S32 s32IPQpDelta;
} VENC_GOP_NORMALP_S;
```

【Member】

Member	Description
s32IPQpDelta	QP difference between I frame and P frame

【Note】

None.

【Related Data Type and Interface】

- CVI_VENC_CreateChn

7.4.7 VENC_GOP_SMARTP_S

【Description】

Define the encoder SmartP GOP attribute structure

【Syntax】

```
typedef struct _VENC_GOP_SMARTP_S {
    CVI_U32 u32BgInterval;
    CVI_S32 s32BgQpDelta;
    CVI_S32 s32ViQpDelta;
} VENC_GOP_SMARTP_S;
```

【Member】

Member	Description
u32BgInterval	IDR frame interval
s32BgQpDelta	QP difference between IDR frame and P frame
s32ViQpDelta	QP difference between Vi frame and P frame

【Note】

None.

【Related Data Type and Interface】

- CVI_VENC_CreateChn

7.4.8 VENC_RECV_PIC_PARAM_S

【Description】

Defines the frame number structure that the coding channel continuously receives and encodes.

【Syntax】

```
typedef struct _VENC_RECV_PIC_PARAM_S {
    CVI_S32 s32RecvPicNum;
} VENC_RECV_PIC_PARAM_S;
```

【Member】

Member	Description
s32RecvPicNum	The number of frames continuously received and encoded by the encoding channel. Range: [- 1,0) (0 ∞]

【Note】

None.

【Related Data Type and Interface】

- CVI_VENC_StartRecvFrame

7.4.9 VENC_CHN_ATTR_S

【Description】

Define VENC_CHN_ATTR_S attribute.

【Syntax】

```
typedef struct _VENC_CHN_ATTR_S {
    VENC_ATTR_S stVencAttr;
    VENC_RC_ATTR_S stRcAttr;
    VENC_GOP_ATTR_S stGopAttr;
} VENC_CHN_ATTR_S;
```

【Member】

Member	Description
stVencAttr	Venc attribute
stRcAttr	Rate controller properties.
stGopAttr	GOP Mode type structure. Please refer to the above typedef struct _VENC_GOP_ATTR_S

【Note】

None.

【Related Data Type and Interface】

- CVI_VENC_CreateChn

7.4.10 VENC_ATTR_S

【Description】

Define VENC_ATTR_S attribute.

【Syntax】

```
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;
        VENC_ATTR_JPEG_S stAttrJpege;
        VENC_ATTR_PRORES_S stAttrProres;
    };
} VENC_ATTR_S;
```

【Member】

Member	Description
enType	payload type.
u32MaxPicWidth	Maximum encoded image width
u32MaxPicHeight	Maximum encoded image height
u32BufSize	Encoded bitstream buffer size
u32Profile	Coding level
bByFrame	Encoded bitstream collection method CVI_TRUE: mainly in frame CVI_FALSE: Mainly in packets
u32PicWidth	Encoded image width
u32PicHeight	Encoded image height
stAttrH264e / stAttrH265e /stAttrJpege / stAttrProres	Encoder attribute

【Note】

Open bIsoSendFrmEn at the same time as bEsBufQueueEn, otherwise there will be a burst screen

【Related Data Type and Interface】

7.4.11 VENC_ATTR_H264_S

【Description】

Define H264 encoding attribute.

【Syntax】

```
typedef struct _VENC_ATTR_H264_S {  
    CVI_BOOL bRcnRefShareBuf;  
    CVI_BOOL bSingleLumaBuf;  
} VENC_ATTR_H264_S;
```

【Member】

Member	Description
bRcnRefShareBuf	Reserved
bSingleLumaBuf	<ul style="list-style-type: none">The FrameBuffer of an encoding channel uses a Luma Buffer

【Note】

None.

【Related Data Type and Interface】

7.4.12 VENC_ATTR_H265_S

【Description】

Define H265 encoding attribute.

【Syntax】

```
typedef struct _VENC_ATTR_H265_S {  
    CVI_BOOL bRcnRefShareBuf;  
} VENC_ATTR_H265_S;
```

【Member】

Member	Description
bRcnRefShareBuf	Reserved

【Note】

None.

【Related Data Type and Interface】

7.4.13 VENC_STREAM_INFO_S

【Description】

Define VENC_STREAM_INFO_S attribute.

【Syntax】

```
typedef struct _ VENC_STREAM_INFO_S {
    H265E_REF_TYPE_E enRefType;

    CVI_U32 u32PicBytesNum;
    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;
```

【Member】

Member	Description
u32MeanQp	The average QP of the current Frame

【Note】

None.

【Related Data Type and Interface】

- CVI_VENC_QueryStatus

7.4.14 VENC_CHN_STATUS_S

【Description】

Define VENC_CHN_STATUS_S attribute.

【Syntax】

```
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;
```

【Member】

Member	Description
u32LeftPics	Reserved
u32LeftStreamBytes	Reserved
u32LeftStreamFrames	Number of remaining frames to be received
u32CurPacks	The number of packets currently in the Frame
u32LeftRecvPics	Reserved
u32LeftEncPics	Reserved
bJpegSnapEnd	Reserved
stVencStrmInfo	Reserved

【Note】

None.

【Related Data Type and Interface】

- CVI_VENC_QueryStatus

7.4.15 VENC_JPEG_PARAM_S

【Description】

Define advanced parameters for JPEG encoding protocol.

【Syntax】

```
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;
```

【Member】

Member	Description
u32Qfactor	For details, see RFC2435. The default value is 0.
u8YQt	Y quantization table. (not implemented)
u8CbQt	Cb quantization table. (not implemented)
u8CrQt	Cr quantization table. (not implemented)
u32MCUPerECS	How many MCU are contained in each ECS? The default value is 0, indicating that Ecs are not divided. (not implemented)

【Note】

None.

【Related Data Type and Interface】

- CVI_VENC_SetJpegParam
- CVI_VENC_GetJpegParam

7.4.16 VENC_RC_ATTR_S

【Description】

Defines the rate controller attribute of the coding channel.

【Syntax】

```
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;
```

【Member】

Member	Description
enRcMode	RC mode.
stH264Cbr	H.264 protocol coding channel Cbr mode attribute.
stH264Vbr	H.264 protocol coding channel Vbr mode attribute.
stH264AVbr	H.264 protocol coding channel AVbr mode attribute.
stH264QVbr	H.264 protocol coding channel QVbr mode attribute.
stH264CVbr	H.264 protocol coding channel CVbr mode attribute.
stH264FixQp	H.264 protocol coding channel Fixqp mode attribute.
stH264QpMap	H.264 protocol coding channel QPMAP mode attribute.
stMjpegeFixQp	Mjpeg protocol coding channel Fixqp mode attribute.
stMjpegeCbr	Mjpeg protocol coding channel cbr mode attribute.
stMjpegeVbr	Mjpeg protocol coding channel vbr mode attribute.
stH265Cbr	H.265 protocol coding channel Cbr mode attribute.
stH265Vbr	H.265 protocol coding channel Vbr mode attribute.
stH265AVbr	H.265 protocol coding channel AVbr mode attribute.
stH265QVbr	H.265 protocol coding channel QVbr mode attribute.
stH265CVbr	H.265 protocol coding channel CVbr mode attribute.
stH265FixQp	H.265 protocol coding channel Fixqp mode attribute.
stH265QpMap	H.265 protocol coding channel QPMAP mode attribute.

【Note】

None.

【Related Data Type and Interface】

- CVI_VENC_CreateChn

7.4.17 VENC_H264_CBR_S

【Description】

Structure definition for H.264 video encoding channel' s CBR properties.

【Syntax】

```
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;
```

【Member】

Member	Description
u32Gop	H.264 gop value. Range: [1, 65536]。
u32StatTime	CBR rate statistics time, in seconds. Range: [1, 60].
u32SrcFrameRate	Input frame rate in fps.
fr32DstFrameRate	Encoder output frame rate, in fps.
u32BitRate	Average bitrate in kbps.
bVariFpsEn	Enable Variable FPS. After this function is enabled, the Frame rate conversion function is enabled and disabled.

【Note】

None.

【Related Data Type and Interface】

- CVI_VENC_CreateChn

7.4.18 VENC_H264_VBR_S

【Description】

Structure definition for H.264 video encoding channel' s VBR properties.

【Syntax】

```
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;
```

【Member】

Member	Description
u32Gop	H.264 gop value Range: [1, 65536].
u32StatTime	VBR rate statistics time, in seconds.. Range: [1, 60].
u32SrcFrameRate	Input frame rate in fps,
fr32DstFrameRate	Encoder output frame rate, in fps.
u32MaxBitRate	Maximum output bitrate of the encoder, in kbps.
bVariFpsEn	Enable Variable FPS. After this function is enabled, the Frame rate conversion function is enabled and disabled.

【Note】

None.

【Related Data Type and Interface】

- CVI_VENC_CreateChn

7.4.19 VENC_H264_AVBR_S

【Description】

Structure definition for H.264 video encoding channel' s AVBR properties.

【Syntax】

```
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;
```

【Member】

Member	Description
u32Gop	H.264 gop value. Range: [1, 65536].
u32StatTime	VBR rate statistics time, in seconds.. Range: [1, 60].
u32SrcFrameRate	Input frame rate in fps.
fr32DstFrameRate	Encoder output frame rate, in fps.
u32MaxBitRate	Maximum output bitrate of the encoder, in kbps.
bVariFpsEn	Enable Variable FPS. After this function is enabled, the Frame rate conversion function is enabled and disabled.

【Note】

None.

【Related Data Type and Interface】

- CVI_VENC_CreateChn

7.4.20 VENC_H264_FIXQP_S

【Description】

Structure definition for H.264 video encoding channel' s FIXQP properties.

【Syntax】

```
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;
} VENC_H264_FIXQP_S;
```

【Member】

Member	Description
u32Gop	H.264 gop value Range: [1, 65536]。
u32IQp	QP value of all macroblocks in I frame.
u32SrcFrameRate	Input frame rate in fps.
fr32DstFrameRate	Encoder output frame rate, in fps.
u32PQp	The QP value of all macroblocks in P frame.
u32BQp	QP values of all macroblocks in B frame.
bVariFpsEn	Enable Variable FPS. After this function is enabled, the Frame rate conversion function is enabled and disabled.

【Note】

None.

【Related Data Type and Interface】

- CVI_VENC_CreateChn

7.4.21 VENC_H264_QPMAP_S

【Description】

Structure definition for H.264 video encoding channel' s QPMAP properties.

【Syntax】

```
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;
```

【Member】

Member	Description
u32Gop	H.264 gop value Range: [1, 65536]。
u32SrcFrameRate	Input frame rate in fps.
fr32DstFrameRate	Encoder output frame rate, in fps.
u32StatTime	QPMAP rate statistics time, in seconds..
bVariFpsEn	Enable Variable FPS. After this function is enabled, the Frame rate conversion function is enabled and disabled.

【Note】

None.

【Related Data Type and Interface】

- CVI_VENC_CreateChn

7.4.22 VENC_MJPEG_FIXQP_S

【Description】

Structure definition for MJPEG video encoding channel' s FIXQP properties.

【Syntax】

```
typedef struct _VENC_MJPEG_FIXQP_S {
    CVI_U32 u32SrcFrameRate
    CVI_FR32 fr32DstFrameRate;
    CVI_U32 u32Qfactor;
    CVI_BOOL bVariFpsEn;
} VENC_MJPEG_FIXQP_S;
```

【Member】

Member	Description
u32SrcFrameRate	Input frame rate in fps.
fr32DstFrameRate	Encoder output frame rate, in fps.
bVariFpsEn	Enable Variable FPS. After this function is enabled, the Frame rate conversion function is enabled and disabled.
u32Qfactor	Qfactor of MJPEG encoding

【Note】

None.

【Related Data Type and Interface】

- CVI_VENC_CreateChn

7.4.23 VENC_MJPEG_CBR_S

【Description】

Structure definition for MJPEG video encoding channel' s CBR properties.

【Syntax】

```
typedef struct _VENC_MJPEG_CBR_S {
    CVI_U32 u32StatTime;
    CVI_U32 u32SrcFrameRate;
    CVI_FR32 fr32DstFrameRate;
    CVI_U32 u32BitRate;
    CVI_BOOL bVariFpsEn;
} VENC_MJPEG_CBR_S;
```

【Member】

Member	Description
u32StatTime	CBR rate statistics time, in seconds..
u32BitRate	Average bitrate in kbps.
u32SrcFrameRate	Input frame rate in fps.
fr32DstFrameRate	Encoder output frame rate, in fps.
bVariFpsEn	Enable Variable FPS. After this function is enabled, the Frame rate conversion function is enabled and disabled.

【Note】

None.

【Related Data Type and Interface】

- CVI_VENC_CreateChn

7.4.24 VENC_H265_CBR_S

【Description】

Structure definition for H.265 video encoding channel' s CBR properties.

【Syntax】

```
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;
```

【Member】

Member	Description
u32Gop	H.265 gop value
u32StatTime	CBR rate statistics time, in seconds..
u32SrcFrameRate	Input frame rate in fps.
fr32DstFrameRate	Encoder output frame rate, in fps.
u32BitRate	Average bitrate in kbps.
bVariFpsEn	Enable Variable FPS. After this function is enabled, the Frame rate conversion function is enabled and disabled.

【Note】

None.

【Related Data Type and Interface】

- CVI_VENC_CreateChn

7.4.25 VENC_H265_VBR_S

【Description】

Structure definition for H.265 video encoding channel' s VBR properties.

【Syntax】

```
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;
} VENC_H264_FIXQP_S;
```

【Member】

Member	Description
u32Gop	H.265 gop value
u32StatTime	VBR rate statistics time, in seconds..
u32SrcFrameRate	Input frame rate in fps.
fr32DstFrameRate	Encoder output frame rate, in fps.
u32MaxBitRate	Maximum output bitrate of the encoder, in kbps.
bVariFpsEn	Enable Variable FPS. After this function is enabled, the Frame rate conversion function is enabled and disabled.

【Note】

None.

【Related Data Type and Interface】

- CVI_VENC_CreateChn

7.4.26 VENC_H265_AVBR_S

【Description】

Structure definition for H.265 video encoding channel' s AVBR properties.

【Syntax】

```
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;
```

【Member】

Member	Description
u32Gop	H.265 gop value. range
u32StatTime	AVBR rate statistics time, in seconds.
u32SrcFrameRate	Input frame rate in fps.
fr32DstFrameRate	Encoder output frame rate, in fps.
u32MaxBitRate	Maximum output bitrate of the encoder, in kbps.
bVariFpsEn	Enable Variable FPS. After this function is enabled, the Frame rate conversion function is enabled and disabled.
u32Gop	H.264 gop value. Range: [1, 65536]。

【Note】

None.

【Related Data Type and Interface】

- CVI_VENC_CreateChn

7.4.27 VENC_H265_FIXQP_S

【Description】

Structure definition for H.265 video encoding channel' s FIXQP properties.

【Syntax】

```
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;
} VENC_H264_FIXQP_S;
typedef struct _VENC_H264_FIXQP_S VENC_H265_FIXQP_S;
```

【Member】

Member	Description
u32Gop	H.265 gop value
u32IQp	QP value of all macroblocks in I frame.
u32SrcFrameRate	Input frame rate in fps.
fr32DstFrameRate	Encoder output frame rate, in fps.
u32PQp	QP value of all macroblocks in P frame.
u32BQp	Reserved
bVariFpsEn	Enable Variable FPS. After this function is enabled, the Frame rate conversion function is enabled and disabled.

【Note】

None.

【Related Data Type and Interface】

- CVI_VENC_CreateChn

7.4.28 VENC_H265_QPMAP_S

【Description】

Structure definition for H.265 video encoding channel' s QPMAP properties.

【Syntax】

```
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;
```

【Member】

Member	Description
u32Gop	H.264 gop value Range: [1, 65536]。
u32StatTime	VBR rate statistical time in seconds
u32SrcFrameRate	Input frame rate in fps.
fr32DstFrameRate	Encoder output frame rate, in fps.
enQpMapMode	The method of selecting the QP value as either CU32 or CU64
u32BQp	QP values of all macroblocks in B frame.
bVariFpsEn	Enable Variable FPS. After this function is enabled, the Frame rate conversion function is enabled and disabled.

【Note】

None.

【Related Data Type and Interface】

- CVI_VENC_CreateChn

7.4.29 VENC_RC_PARAM_S

【Description】

Definition of advanced bitrate control parameters for encoding channels. Main structure for bitrate control.

【Syntax】

```
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;
```

【Member】

Member	Description
u32ThrdI	Reserved
u32ThrdP	Reserved
u32ThrdB	Reserved
u32DirectionThrd	Reserved
u32RowQpDelta	The fluctuation amplitude value of the starting Qp for each row of macroblocks in macroblock-level bitrate control relative to the starting Qp of the frame.
s32FirstFrameStartQp	Set the initial Qp value of the first frame, CBR/VBR/AVBR/QVBR/CVBR is valid.
stSceneChangeDetect	Reserved
s32InitialDelay	Influence the frame encoding of the bitrate control.
u32ThrdLv	MAD threshold for macroblock level rate control
bBgEnhanceEn	Reserved
s32BgDeltaQp	The QP difference between IDR frame and P frame in smartP mode.
stParamH264Cbr	Advanced parameters for CBR rate control mode in H.264 channel.
stParamH264Vbr	Advanced parameters for VBR rate control mode in H.264 channel.
stParamH264AVbr	Advanced parameters for AVBR rate control mode in H.264 channel.
stParamH264QVbr	Reserved
stParamH264CVbr	Reserved
stParamMjpegCbr	Advanced parameters for CBR rate control mode in MJPEG channel.
stParamMjpegVbr	Reserved
stParamH265Cbr	Advanced parameters for CBR rate control mode in H.265 channel.
stParamH265Vbr	Advanced parameters for VBR rate control mode in H.265 channel.
stParamH265AVbr	Advanced parameters for AVBR rate control mode in H.265 channel.
stParamH265QVbr	Reserved
stParamH265CVbr	Reserved

【Note】

None.

【Related Data Type and Interface】

- CVI_VENC_SetRcParam
- CVI_VENC_GetRcParam

7.4.30 VENC_PARAM_H264_CBR_S

【Description】

Definition of H.264 CBR advanced parameters

【Syntax】

```
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;
```

(continues on next page)

(continued from previous page)

```

    CVI_BOOL bQpMapEn;
} VENC_PARAM_H264_CBR_S;

```

【Member】

Member	Description
u32MinIprop	minimum IP ratio
u32MaxIprop	maximum IP ratio
u32MaxQp	Minimum QP
u32MinQp	Maximum QP
u32MaxIQp	Maximum QP for I-frames
u32MinIQp	Minimum QP for I-frames
s32MaxReEncodeTimes	Reserved
CVI_BOOL bQpMapEn	Reserved

【Note】

None.

【Related Data Type and Interface】

- CVI_VENC_SetRcParam
- CVI_VENC_GetRcParam

7.4.31 VENC_PARAM_H264_VBR_S

【Description】

Definition of H.264 VBR advanced parameters

【Syntax】

```

typedef struct _VENC_PARAM_H264_VBR_S {
    CVI_S32 s32ChangePos;
    CVI_U32 u32MinIprop;
    CVI_U32 u32MaxIprop;
    CVI_S32 s32MaxReEncodeTimes;
    CVI_BOOL bQpMapEn;

    CVI_U32 u32MaxQp;
    CVI_U32 u32MinQp;
    CVI_U32 u32MaxIQp;
    CVI_U32 u32MinIQp;
} VENC_PARAM_H264_VBR_S;

```

【Member】

Member	Description
s32ChangePos	bitrate control threshold
u32MinIprop	Reserved
u32MaxIprop	maximum IP ratio
s32MaxReEncodeTimes	Reserved
CVI_BOOL bQpMapEn	Reserved
u32MaxQp	Maximum QP
u32MinQp	Minimum QP
u32MaxIQp	maximum QP for I-frames
u32MinIQp	minimum QP for I-frames

【Note】

None.

【Related Data Type and Interface】

- CVI_VENC_SetRcParam
- CVI_VENC_GetRcParam

7.4.32 VENC_PARAM_H264_AVBR_S

【Description】

Definition of H.264 AVBR advanced parameters

【Syntax】

```
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;

    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;
```

【Member】

Member	Description
s32ChangePos	bitrate control threshold
u32MinIprop	Reserved
u32MaxIprop	maximum IP ratio
s32MaxReEncodeTimes	Reserved
CVI_BOOL bQpMapEn	Reserved
s32MinStillPercent	minimum bitrate percentage for static scenes
u32MaxStillQP	Maximum QP for static scenes
u32MinStillPSNR	Reserved
u32MaxQp	Maximum QP
u32MinQp	Minimum QP
u32MaxIQp	Maximum QP for I-frames
u32MinIQp	Minimum QP for I-frames
u32MinQpDelta	The difference between frame-level minimum QP and macroblock-level minimum QP
u32MotionSensitivity	Motion sensitivity
s32AvbrFrmLostOpen	Frame dropping enable
s32AvbrFrmGap	Maximum number of dropped frames
s32AvbrPureStillThr	Still macroblock threshold value

【Note】

None.

【Related Data Type and Interface】

- CVI_VENC_SetRcParam
- CVI_VENC_GetRcParam

7.4.33 VENC_PARAM_H265_CBR_S

【Description】

Definition of H.265 CBR advanced parameters

【Syntax】

```
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;
```

【Member】

Member	Description
u32MinIprop	Reserved
u32MaxIprop	Reserved
u32MaxQp	Maximum QP
u32MinQp	Minimum QP
u32MaxIQp	Maximum QP for I-frames
u32MinIQp	Minimum QP for I-frames
s32MaxReEncodeTimes	Reserved
bQpMapEn	Enable QP map feature or not
enQpMapMode	QpMap mode

【Note】

None.

【Related Data Type and Interface】

- CVI_VENC_SetRcParam
- CVI_VENC_GetRcParam

7.4.34 VENC_PARAM_H265_VBR_S

【Description】

Definition of H.265 VBR advanced parameters

【Syntax】

```
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;
```

【Member】

Member	Description
s32ChangePos	bitrate control threshold
u32MinIprop	Reserved
u32MaxIprop	Reserved
s32MaxReEncodeTimes	Reserved
u32MaxQp	Maximum QP
u32MinQp	Minimum QP
u32MaxIQp	Maximum QP for I-frames
u32MinIQp	Minimum QP for I-frames
CVI_BOOL bQpMapEn	Reserved
enQpMapMode	Reserved

【Note】

None.

【Related Data Type and Interface】

- CVI_VENC_SetRcParam
- CVI_VENC_GetRcParam

7.4.35 VENC_PARAM_H265_AVBR_S

【Description】

Definition of H.265 AVBR advanced parameters

【Syntax】

```
typedef struct _VENC_PARAM_H265_AVBR_S {
    CVI_S32 s32ChangePos;
    CVI_U32 u32MinIprop;
    CVI_U32 u32MaxIprop;
    CVI_S32 s32MaxReEncodeTimes;

    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;
```

【Member】

Member	Description
s32ChangePos	Bitrate control threshold
u32MinIprop	Reserved
u32MaxIprop	Reserved
s32MaxReEncodeTimes	Reserved
s32MinStillPercent	Minimum bitrate percentage for static scenes
u32MaxStillQP	Maximum QP for static scenes
u32MinStillPSNR	Reserved
u32MaxQp	Maximum QP
u32MinQp	Minimum QP
u32MaxIQp	Maximum QP for I-frames
u32MinIQp	Minimum QP for I-frames
u32MinQpDelta	The difference between frame-level minimum QP and macroblock-level minimum QP
u32MotionSensitivity	Motion sensitivity
s32AvbrFrmLostOpen	Frame dropping enable
s32AvbrFrmGap	Maximum number of dropped frames
s32AvbrPureStillThr	Still macroblock threshold value
bQpMapEn	Reserved
enQpMapMode	Reserved

【Note】

None.

【Related Data Type and Interface】

- CVI_VENC_SetRcParam
- CVI_VENC_GetRcParam

7.4.36 VENC_PARAM_MOD_S

【Description】

Encoding-related module parameters

【Syntax】

```
typedef struct _VENC_MODPARAM_S {
    VENC_MODTYPE_E enVencModType; /* RW; VencModType*/
    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;
```

【Member】

Member	Description
enVencModType	Types of module parameters
stVencModParam /stH264eModParam /stH265eModParam /stJpege- ModParam /stRcModParam	Venc / H264e / H265e / Jpege / Rc module parameter structure.

【Note】

None.

【Related Data Type and Interface】

- CVI_VENC_SetModParam
- CVI_VENC_GetModParam

7.4.37 VENC_MOD_H264E_S

【Description】

H.264 encoding-related module parameters

【Syntax】

```
typedef struct _VENC_MOD_H264E_S {
    CVI_U32 u32OneStreamBuffer;
    CVI_U32 u32H264eMiniBufMode;
    CVI_U32 u32H264ePowerSaveEn;
    VB_SOURCE_E enH264eVBSrc;
    CVI_BOOL bQpHstgrmEn;
    CVI_U32 u32UserDataMaxLen;
    CVI_BOOL bSingleEsBuf;
    CVI_U32 u32SingleEsBufSize;
} VENC_MOD_H264E_S;
```

【Member】

Member	Description
u32OneStreamBuffer	Reserved
u32H264eMiniBufMode	Reserved
u32H264ePowerSaveEn	Reserved
enH264eVBSrc	VB mode
bQpHstgrmEn	Reserved
u32UserDataMaxLen	Maximum user data size
bSingleEsBuf	Multiple channels using a shared stream buffer
u32SingleEsBufSize	StreamBuffer size

【Note】

None.

【Related Data Type and Interface】

- CVI_VENC_SetModParam
- CVI_VENC_GetModParam

7.4.38 VENC_MOD_H265E_S

【Description】

H.265 encoding-related module parameters

【Syntax】

```
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;
```

【Member】

Member	Description
u32OneStreamBuffer	Reserved
u32H265eMiniBufMode	Reserved
u32H265ePowerSaveEn	Reserved
enH265eVBSource	VB mode
bQpHstgrmEn	Reserved
u32UserDataMaxLen	Maximum user data size
bSingleEsBuf	Multiple channels using a shared stream buffer
u32SingleEsBufSize	StreamBuffer size
enRefreshType	Refresh type

【Note】

None.

【Related Data Type and Interface】

- CVI_VENC_SetModParam
- CVI_VENC_GetModParam

7.4.39 VENC_MOD_JPEGE_S

【Description】

JPEG encoding-related module parameters

【Syntax】

```
typedef struct _VENC_MOD_JPEGE_S {
    CVI_U32 u32OneStreamBuffer;
    CVI_U32 u32JpegeMiniBufMode;
    CVI_U32 u32JpegClearStreamBuf;
    CVI_BOOL bSingleEsBuf;
    CVI_U32 u32SingleEsBufSize;
    JPEG_FORMAT_E enJpegeFormat;
    JPEG_MARKER_TYPE_E JpegMarkerOrder[JPEG_MARKER_ORDER_CNT];
} VENC_MOD_JPEGE_S;
```

【Member】

Member	Description
u32OneStreamBuffer	Reserved
u32JpegeMiniBufMode	Reserved
u32JpegClearStreamBuf	Reserved
bSingleEsBuf	Multiple channels using a shared stream buffer
u32SingleEsBufSize	StreamBuffer size
enJpegeFormat	JPEG header encoding mode
JpegMarkerOrder	Header encoding mark order

【Note】

None.

【Related Data Type and Interface】

- CVI_VENC_SetModParam
- CVI_VENC_GetModParam

7.4.40 VENC_CHN_POOL_S

【Description】

Define the VB pool structure, which is bound to the encoding channel.

【Syntax】

```
typedef struct _VENC_CHN_POOL_S {
    VB_POOL hPicVbPool; /* RW; vb pool id for pic buffer */
    VB_POOL hPicInfoVbPool; /* RW; vb pool id for pic info buffer */
} VENC_CHN_POOL_S;
```

【Member】

Member	Description
hPicVbPool	VB pool Poold for storing Picture
hPicInfoVbPool	VB pool Poold for storing Picture information

【Note】

None.

【Related Data Type and Interface】

- CVI_VENC_AttachVbPool

7.4.41 VENC_FRAMELOST_S

【Description】

Define the encoding frame dropping structure.

【Syntax】

```
typedef struct _VENC_FRAMELOST_S {
    CVI_BOOL bFrmLostOpen;
    CVI_U32 u32FrmLostBpsThr;
    VENC_FRAMELOST_MODE_E enFrmLostMode;
```

(continues on next page)

(continued from previous page)

```

    CVI_U32 u32EncFrmGaps;
} VENC_FRAMELOST_S;

```

【Member】

Member	Description
bFrmLostOpen	Frame dropping strategy enable
u32FrmLostBpsThr	Frame dropping bitrate threshold value
enFrmLostMode	Frame dropping mode
u32EncFrmGaps	Maximum consecutive dropped frames

【Note】

None.

【Related Data Type and Interface】

- CVI_VENC_SetFrameLostStrategy
- CVI_VENC_GetFrameLostStrategy

7.4.42 VENC_H264_ENTROPY_S

【Description】

Define H.264 entropy coding information structure.

【Syntax】

```

typedef struct _VENC_H264_ENTROPY_S {
    CVI_U32 u32EntropyEncModeI;
    CVI_U32 u32EntropyEncModeP;
    CVI_U32 u32EntropyEncModeB;
    CVI_U32 cabac_init_idc;
} VENC_H264_ENTROPY_S;

```

【Member】

Member	Description
u32EntropyEncModeI	I-frame entropy coding mode: 0 for CAVLC and 1 for CABAC.
u32EntropyEncModeP	P-frame entropy coding mode: 0 for CAVLC and 1 for CABAC.
u32EntropyEncModeB	Reserved, not used yet
cabac_init_idc	Refer to H264 encoding protocol

【Note】

None.

【Related Data Type and Interface】

- CVI_VENC_SetH264Entropy
- CVI_VENC_GetH264Entropy

7.4.43 VENC_CU_PREDICTION_S

【Description】

Define the structure of coding unit prediction properties

【Syntax】

```
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;
```

【Member】

Member	Description
enPredMode	Reserved, not used yet
u32IntraCost	Default to 0, used to reduce the probability of Intra Blocks.
u32Intra32Cost	Reserved, not used yet
u32Intra16Cost	Reserved, not used yet
u32Intra8Cost	Reserved, not used yet
u32Intra4Cost	Reserved, not used yet
u32Inter64Cost	Reserved, not used yet
u32Inter32Cost	Reserved, not used yet
u32Inter16Cost	Reserved, not used yet
u32Inter8Cost	Reserved, not used yet

【Note】

None.

【Related Data Type and Interface】

- CVI_VENC_SetCuPrediction
- CVI_VENC_GetCuPrediction

7.4.44 VENC_H264_TRANS_S

【Description】

Define the structure of the H.264 protocol encoding channel transformation quantization.

【Syntax】

```
typedef struct _VENC_H264_TRANS_S {
    CVI_U32 u32IntraTransMode;
    CVI_U32 u32InterTransMode;
    CVI_BOOL bScalingListValid;
    CVI_U8 InterScalingList8X8[64];
```

(continues on next page)

(continued from previous page)

```

    CVI_U8 IntraScalingList8X8[64];
    CVI_S32 chroma_qp_index_offset;
} VENC_H264_TRANS_S;

```

【Member】

Member	Description
u32IntraTransMode	Reserved, not used yet
u32InterTransMode	Reserved, not used yet
bScalingListValid	Reserved, not used yet
InterScalingList8X8[64]	Reserved, not used yet
IntraScalingList8X8[64]	Reserved, not used yet
chroma_qp_index_offset	For details, see H.264 Protocol. The default value is 0. Value range: [-12, 12]

【Note】

None.

【Related Data Type and Interface】

- CVI_VENC_GetH264Trans
- CVI_VENC_SetH264Trans

7.4.45 VENC_H265_TRANS_S**【Description】**

Define the structure of the H.265 protocol encoding channel transformation quantization.

【Syntax】

```

typedef struct _VENC_H265_TRANS_S {
    CVI_S32 cb_qp_offset;
    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;

```

【Member】

Member	Description
cb_qp_offset	For details, see H.265 Protocol. The default value is 0. Value range: [-12, 12]
cr_qp_offset	For details, see H.265 Protocol. The default value is 0. Value range: [-12, 12]
bScalingListEnabled	Reserved, not used yet
bScalingListTu4Valid	Reserved, not used yet
InterScalingList4X4[2][16]	Reserved, not used yet
IntraScalingList4X4[2][16]	Reserved, not used yet
bScalingListTu8Valid	Reserved, not used yet
InterScalingList8X8[2][64]	Reserved, not used yet
IntraScalingList8X8[2][64]	Reserved, not used yet
bScalingListTu16Valid	Reserved, not used yet
InterScalingList16X16[2][64]	Reserved, not used yet
IntraScalingList16X16[2][64]	Reserved, not used yet
bScalingListTu32Valid	Reserved, not used yet
InterScalingList32X32[64]	Reserved, not used yet
IntraScalingList32X32[64]	Reserved, not used yet

【Note】

None.

【Related Data Type and Interface】

- CVI_VENC_GetH265Trans
- CVI_VENC_SetH265Trans

7.5 Error Codes

The error codes of video encoding API are shown in the following table

Error Code	Macro Definition	Description
0xC0078002	CVI_ERR_VENC_INVALID_CHANID	Invalid channel ID
0xC0078003	CVI_ERR_VENC_ILLEGAL_PARAM	Illegal parameters
0xC0078004	CVI_ERR_VENC_EXIST	Attempts to apply for or create an existing device, channel, or resource
0xC0078005	CVI_ERR_VENC_UNEXIST	Channel does not exist
0xC0078006	CVI_ERR_VENC_NULL_PTR	Null pointer
0xC0078007	CVI_ERR_VENC_NOT_CONFIG	Not configured before use
0xC0078008	CVI_ERR_VENC_NOT_SUPPORT	Operation not supported
0xC0078009	CVI_ERR_VENC_NOT_PERM	The operation is not allowed
0xC007800A	CVI_ERR_VENC_INVALID_PIPEID	Invalid PIPE ID
0xC007800B	CVI_ERR_VENC_INVALID_GROUPID	Invalid GROUP ID
0xC007800C	CVI_ERR_VENC_NOMEM	Memory configuration failure
0xC007800D	CVI_ERR_VENC_NOBUF	Image Buffer configuration failed
0xC007800E	CVI_ERR_VENC_BUF_EMPTY	No data in buffer
0xC007800F	CVI_ERR_VENC_BUF_FULL	buffer full data
0xC0078010	CVI_ERR_VENC_SYS_NOTREADY	System is not ready
0xC0078011	CVI_ERR_VENC_BADADDR	Invalid address
0xC0078012	CVI_ERR_VENC_BUSY	Device in use
0xC0078014	CVI_ERR_VENC_INVALID_VB	Invalid VB
0xC0078040	CVI_ERR_VENC_INIT	Is initializing
0xC0078041	CVI_ERR_VENC_FRC_NO_ENC	FRC actively skips encoding the current frame
0xC0078042	CVI_ERR_VENC_STAT_VFPST_CHANGE	State changed
0xC0078043	CVI_ERR_VENC_EMPTY_STREAM	Empty stream
0xC0078044	CVI_ERR_VENC_EMPTY_PACKET	Packet stream is available
0xC0078045	CVI_ERR_VENC_JPEG_MARKER_ORDER	JPEG marker order
0xC0078047	CVI_ERR_VENC_RC_PARAM	RC parameter setting
0xC007804B	CVI_ERR_VENC_MUTEX_ERROR	Signal error
0xC007804C	CVI_ERR_VENC_INVALID_LD	Undefined underlying illegal return value

VIDEO DECODING

8.1 Function Overview

VDEC module provides video decoding service, The compressed image data is decoded and the original image data is output.

The currently supported input sources are:

- Image data input by user under User Mode

Supported video standards currently include:

CV181x VDEC module only supports PT_JPEG/PT_MJPEG/PT_H264, CV180x only supports PT_JPEG/PT_MJPEG.

8.1.1 Objective

VDEC module provides the corresponding interface to drive the hardware of video decoding, and realizes video decoding function.

8.1.2 Definitions and Abbreviations

Acronym or term	Definition
VDEC	Video Decoder
Output Order	Order of output
Decoding Order	Order of decoding
Display Order	Order of display
Frame	Frame
Stream	Bitstream

8.2 Design Overview

8.2.1 Bitstream Delivery Method

The bitstream delivery method provided by VDEC is as follows:

- Send by frame (VIDEO_MODE_FRAME): Every time a complete frame code stream is sent to the decoder, every time the sending interface is called, the decoder will consider the frame code stream has ended and start decoding the image.

It is necessary to ensure that the code stream sent by the sending interface must be one frame each time.

8.2.2 Image Output Format

According to the H.264 video standard, before the input Stream is decoded, the sequence of output images is not necessarily equal to that of input. Therefore, there are two kinds of playback: Decoding order and Display order.

- Decoding Order: the output order of images is the same as the input order of the stream
 - The decoded Frame can be obtained quickly, but the user needs to ensure the playback order.
For example, if there is a B frame in the general Stream, the display order transformation is required, and the user needs to do related processing.
- Display Order: the output order of images is the same as the playback order
 - The Frame the user obtains is already a Display order and can be played directly in that order.

The current Output order is set as Display order.

8.2.3 Timestamp (PTS) Processing

PTS refers to the time point when the current Frame is playing.

The PTS of the current Frame can be obtained from `CVI_VENC_GetFrame`, and the PTS of the frame will be equal to the PTS attached to `CVI_VENC_SendStream`.

8.2.4 Decoding Frame Buffer Allocation Mode

- Common Mode: ION memory is automatically created for frame storage.
ION size is automatically allocated according to the Width and Height after decoding. Users do not need to manage this memory
- User Mode: Users need to use `CVI_VB_CreatePool` to create a VB Pool.
After creating a channel, they can bind the VB Pool to the channel through `CVI_VDEC_AttachVbPool`
- Private Mode: When creating a channel, a Private VB Pool will be created automatically.
Users do not need to create their own VB pool.
They can set `u32FrameBufSize` and `u32FrameBufCnt` of Private VB Pool through `CVI_VDEC_CreateChn`.

`CVI_VDEC_SetModParam` can be used to set `enVdecVBSource` to select decoding frame buffer allocation mode.

Currently only COMMON Mode and USER Mode are supported.

When using USER Mode, the VB pool cannot be destroyed directly before the channel is detached to VB pool.

You need to make sure that the decoder ends correctly before you can destroy it.

8.3 API Reference

This VDEC function module provides the following APIs for users:

- *CVI_VDEC_CreateChn*: create a video decoding channel.
- *CVI_VDEC_DestroyChn*: destroy the video decoding channel.
- *CVI_VDEC_ResetChn*: reset the video decoding channel.
- *CVI_VDEC_GetChnAttr*: get video decoding channel attributes.
- *CVI_VDEC_SetChnAttr*: set video decoding channel attributes.
- *CVI_VDEC_StartRecvStream*: the decoder starts to receive the bitstream sent by the user.
- *CVI_VDEC_StopRecvStream*: the decoder stops receiving the bitstream sent by the user.
- *CVI_VDEC_QueryStatus*: query the status of decoding channel
- *CVI_VDEC_SetChnParam*: set video decoding channel parameters.
- *CVI_VDEC_GetChnParam*: get video decoding channel parameters.
- *CVI_VDEC_SendStream*: send the bitstream data to the video decoding channel.
- *CVI_VDEC_GetFrame*: get the decoded image from the video decoding channel.
- *CVI_VDEC_ReleaseFrame*: release the decoded image of the video decoding channel.
- *CVI_VDEC_SetModParam*: Set decoding-related module parameters
- *CVI_VDEC_GetModParam*: Get decoding-related module parameters
- *CVI_VDEC_AttachVbPool*: Bind the decoding channel to a video buffer VB pool.
- *CVI_VDEC_DetachVbPool*: Unbind the decoding channel from a video cache VB pool.

8.3.1 CVI_VDEC_CreateChn

【Description】

Create a video decoding channel.

【Syntax】

```
CVI_S32 CVI_VDEC_CreateChn(VDEC_CHN VdChn, const VDEC_CHN_ATTR_S *pstAttr);
```

【Parameter】

Parameter	Description	Input/Output
VdChn	Video decoding channel number.	Input
pstAttr	Decoding channel property pointer.	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: cvi_vdec.h, cvi_comm_video.h, cvi_comm_vdec.h
- Library files: libvdec.so/libvdec.a

【Note】

- CV181x VDEC module only supports PT_JPEG/PT_MJPEG/PT_H264, CV180x only supports PT_JPEG/PT_MJPEG.
- When the system is out of memory, CVII_ERR_VDEC_NOMEM error code will be returned.
- To use JPEG/MJPEG, a VB pool dedicated to VDEC module must be created before creating the decoding channel.

The size of the VB block required for decoding different protocols varies, which can be referred to the vdecInitVBPool function in the sample_vdec_lib.c file.

【Example】

```
CVI_S32 SAMPLE_COMM_VDEC_Start(vdecChnCtx *pvdchnCtx)
{
    //Setting VDEC parameters
    VDEC_CHN_ATTR_S stChnAttr, *pstChnAttr = &stChnAttr;
    VDEC_CHN VdecChn = pvdchnCtx->VdecChn;
    SAMPLE_VDEC_ATTR *psvdatatr = &pvdchnCtx->stSampleVdecAttr;
    VDEC_CHN_PARAM_S stChnParam;

    pstChnAttr->enType = psvdatatr->enType;
    pstChnAttr->enMode = psvdatatr->enMode;
    pstChnAttr->u32PicWidth = psvdatatr->u32Width;
    pstChnAttr->u32PicHeight = psvdatatr->u32Height;
    pstChnAttr->u32StreamBufSize = psvdatatr->u32Width * psvdatatr->u32Height;
    pstChnAttr->u32FrameBufCnt = psvdatatr->u32FrameBufCnt;

    //JPEG, MJPEG need to set VB buffer
    if (psvdatatr->enType == PT_JPEG || psvdatatr->enType == PT_MJPEG) {
        pstChnAttr->enMode = VIDEO_MODE_FRAME;
        pstChnAttr->u32FrameBufSize = VDEC_GetPicBufferSize(
            pstChnAttr->enType, psvdatatr->u32Width, psvdatatr->u32Height,
            psvdatatr->stSampleVdecPicture.enPixelFormat, DATA_BITWIDTH_8, 0);
    }

    //create VDEC channel
    CHECK_CHN_RET(CVI_VDEC_CreateChn(VdecChn, pstChnAttr), VdecChn,
        "CVI_VDEC_CreateChn");

    //confirm the current default parameter
    CHECK_CHN_RET(CVI_VDEC_GetChnParam(VdecChn, &stChnParam), VdecChn,
        "CVI_VDEC_GetChnParam");

    if (psvdatatr->enType == PT_H264 || psvdatatr->enType == PT_H265) {
    } else {
        stChnParam.stVdecPictureParam.enPixelFormat =
            psvdatatr->stSampleVdecPicture.enPixelFormat;
        stChnParam.stVdecPictureParam.u32Alpha =
            psvdatatr->stSampleVdecPicture.u32Alpha;
    }

    //Set display frame.. parameters
    stChnParam.u32DisplayFrameNum = psvdatatr->u32DisplayFrameNum;
    //Set VDEC parameter
    CHECK_CHN_RET(CVI_VDEC_SetChnParam(VdecChn, &stChnParam), VdecChn,
        "CVI_MPI_VDEC_GetChnParam");

    //Enable VDEC frame transmission
    CHECK_CHN_RET(CVI_VDEC_StartRecvStream(VdecChn), VdecChn,
        "CVI_MPI_VDEC_StartRecvStream");

    return CVI_SUCCESS;
}
```

【Related Topic】

- [*CVI_VDEC_DestroyChn*](#)

8.3.2 CVI_VDEC_DestroyChn**【Description】**

Destroy the video decoding channel.

【Syntax】

```
CVI_S32 CVI_VDEC_DestroyChn(VDEC_CHN VdChn);
```

【Parameter】

Parameter	Description	Input/Output
VdChn	Video decoding channel number.	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: `cvi_vdec.h`, `cvi_comm_video.h`, `cvi_comm_vdec.h`
- Library files: `libvdec.so`/`libvdec.a`

【Note】

- Ensure that the channel is created before destruction.
Otherwise, an error message indicating that the channel is not created is returned.
- You must stop receiving the code stream before destroying it (or you have not started receiving the code stream), otherwise an error will be returned.

【Example】

- None

【Related Topic】

- [*CVI_VDEC_CreateChn*](#)

8.3.3 CVI_VDEC_ResetChn**【Description】**

Reset the video decoding channel.

【Syntax】

```
CVI_S32 CVI_VDEC_ResetChn(VDEC_CHN VdChn);
```

【Parameter】

Parameter	Description	Input/Output
VdChn	Channel ID.	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: cvi_vdec.h, cvi_comm_video.h, cvi_comm_vdec.h
- Library files: libvdec.so/libvdec.a

【Note】

- Resetting a channel that does not exist will return CVI_FAILURE.
- Failure is returned if a channel resets without stopping receiving a stream.

【Example】

- None

8.3.4 CVI_VDEC_GetChnAttr**【Description】**

Get video decoding channel attributes.

【Syntax】

```
CVI_S32 CVI_VDEC_GetChnAttr(VDEC_CHN VdChn, VDEC_CHN_ATTR_S *pstAttr);
```

【Parameter】

Parameter	Description	Input/Output
VdChn	Video decoding channel number.	Input
pstAttr	Decode channel property pointer.	Output

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: cvi_vdec.h, cvi_comm_video.h, cvi_comm_vdec.h
- Library files: libvdec.so/libvdec.a

【Note】

- VDEC channel must have been created.
- This function is usually used before a CVI_VDEC_SetChnAttr, or when fetching a decoder frame Call before to confirm the channel is normal

【Example】

Refer to SAMPLE_COMM_VDEC_GetPic function in sample_common_vdec.c.

```

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)
            break;

        s32Ret = CVI_VDEC_GetFrame(
            pstVdecThreadParam->s32ChnId,
            &stVFrame,
            pstVdecThreadParam->s32MilliSec);
        CVI_VDEC_TRACE("leave CVI_VDEC_GetFrame %d\n", s32Ret);
        ...
        ...
    }
}

```

【Related Topic】

- [CVI_VDEC_SetChnAttr](#)

8.3.5 CVI_VDEC_SetChnAttr

【Description】

Set video decoding channel attributes.

【Syntax】

```
CVI_S32 CVI_VDEC_SetChnAttr(VDEC_CHN VdChn, const VDEC_CHN_ATTR_S *pstAttr);
```

【Parameter】

Parameter	Description	Input/Output
VdChn	Video decoding channel number.	Input
pstAttr	Decoding channel property pointer.	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: `cvi_vdec.h`, `cvi_comm_video.h`, `cvi_comm_vdec.h`
- Library files: `libvdec.so/libvdec.a`

【Note】

- VDEC channel must have be created.
- You must stop receiving streams before changing channel properties.
Otherwise, an error is returned.

【Example】

- None.

【Related Topic】

- [CVI_VDEC_GetChnAttr](#)

8.3.6 CVI_VDEC_StartRecvStream

【Description】

The decoder starts to receive the bitstream sent by the user.

【Syntax】

```
CVI_S32 CVI_VDEC_StartRecvStream(VDEC_CHN VdChn);
```

【Parameter】

Parameter	Description	Input/Output
VdChn	Video decoding channel number.	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: `cvi_vdec.h`, `cvi_comm_video.h`, `cvi_comm_vdec.h`
- Library files: `libvdec.so/libvdec.a`

【Note】

- Before starting the receive stream, you must ensure that the channel has been created, otherwise an error will be returned.

【Example】

Refer to the example of `CVI_VDEC_CreateChn`.

【Related Topic】

- *CVI_VDEC_CreateChn*

8.3.7 CVI_VDEC_StopRecvStream

【Description】

The decoder stops receiving the bitstream sent by the user.

【Syntax】

```
CVI_S32 CVI_VDEC_StopRecvStream(VDEC_CHN VdChn);
```

【Parameter】

Parameter	Description	Input/Output
VdChn	Video decoding channel number.	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: `cvi_vdec.h`, `cvi_comm_video.h`, `cvi_comm_vdec.h`
- Library files: `libvdec.so`/`libvdec.a`

【Note】

- This interface must be called before `CVI_VDEC_DestroyChn`.

【Example】

- Refer to `SAMPLE_COMM_VDEC_Stop` in `sample_common_vdec.c`:

```
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;
}
```

【Related Topic】

- None

8.3.8 CVI_VDEC_QueryStatus

【Description】

Query the status of decoding channel.

【Syntax】

```
CVI_S32 CVI_VDEC_QueryStatus(VDEC_CHN VdChn, VDEC_CHN_STATUS_S *pstStatus);
```

【Parameter】

Parameter	Description	Input/Output
VdChn	Video decoding channel number.	Input
pstStatus	Video decoding channel state structure pointer	Output

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: cvi_vdec.h, cvi_comm_video.h, cvi_comm_vdec.h
- Library files: libvdec.so/libvdec.a

【Note】

- Before starting to receive the bitstream, the channel must be created. Otherwise, an error will be returned.

【Example】

- Refer to the usage of SAMPLE_COMM_VDEC_CmdCtrl in sample_common_vdec.c.
- CVI_VDEC_QueryStatus allows users to query the following information:
 - enType: Encoding format.
 - bStartRecvStream: Whether frame to decoder has started to be transmitted.
 - u32DecodeStreamFrames: The number of frames decoded.
 - u32LeftPics: The number of remaining images.
 - stVdecDecErr: Decoder error status (s32FormatErr: format error, s32PicSizeErrSet: Image size error, s32StreamUnsprt: Stream format not supported ...).

【Related Topic】

- None

8.3.9 CVI_VDEC_SetChnParam

【Description】

Set decoding channel parameters.

【Syntax】

```
CVI_S32 CVI_VDEC_SetChnParam(VDEC_CHN VdChn, const VDEC_CHN_PARAM_S *pstParam);
```

【Parameter】

Parameter	Description	Input/Output
VdChn	Video decoding channel number.	Input
pstParam	Channel parameter	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: cvi_vdec.h, cvi_comm_video.h, cvi_comm_vdec.h
- Library files: libvdec.so/libvdec.a

【Note】

- Ensure that the channel is created before starting the receiving stream.
Otherwise, an error is returned.

【Example】

Refer to SAMPLE_COMM_VDEC_Start function in sample_common_vdec.c:

```
CVI_S32 SAMPLE_COMM_VDEC_Start(vdecChnCtx *pvdchnCtx)
{
    VDEC_CHN_ATTR_S stChnAttr, *pstChnAttr = &stChnAttr;
    VDEC_CHN VdecChn = pvdchnCtx->VdecChn;
    SAMPLE_VDEC_ATTR *psvdatatr = &pvdchnCtx->stSampleVdecAttr;
    VDEC_CHN_PARAM_S stChnParam;

    pstChnAttr->enType = psvdatatr->enType;
    pstChnAttr->enMode = psvdatatr->enMode;
    pstChnAttr->u32PicWidth = psvdatatr->u32Width;
    pstChnAttr->u32PicHeight = psvdatatr->u32Height;
    pstChnAttr->u32StreamBufSize = psvdatatr->u32Width * psvdatatr->u32Height;
    pstChnAttr->u32FrameBufCnt = psvdatatr->u32FrameBufCnt;

    if (psvdatatr->enType == PT_JPEG || psvdatatr->enType == PT_MJPEG) {
        pstChnAttr->enMode = VIDEO_MODE_FRAME;
        pstChnAttr->u32FrameBufSize = VDEC_GetPicBufferSize(
            pstChnAttr->enType, psvdatatr->u32Width, psvdatatr->u32Height,
            psvdatatr->stSampleVdecPicture.enPixelFormat, DATA_BITWIDTH_8, 0);
    }

    CHECK_CHN_RET(CVI_VDEC_CreateChn(VdecChn, pstChnAttr), VdecChn, "CVI_VDEC_CreateChn
    ");
```

(continues on next page)

(continued from previous page)

```

CHECK_CHN_RET(CVI_VDEC_GetChnParam(VdecChn, &stChnParam), VdecChn, "CVI_VDEC_
↪GetChnParam");

if (psvdatatr->enType == PT_H264 || psvdatatr->enType == PT_H265) {
} else {
    stChnParam.stVdecPictureParam.enPixelFormat =
        psvdatatr->stSapmleVdecPicture.enPixelFormat;
    stChnParam.stVdecPictureParam.u32Alpha =
        psvdatatr->stSapmleVdecPicture.u32Alpha;
}
stChnParam.u32DisplayFrameNum = psvdatatr->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;
}

```

【Related Topic】

- None.

8.3.10 CVI_VDEC_GetChnParam**【Description】**

Get decoding channel parameters.

【Syntax】

```
CVI_S32 CVI_VDEC_GetChnParam(VDEC_CHN VdChn, VDEC_CHN_PARAM_S *pstParam);
```

【Parameter】

Parameter	Description	Input/Output
VdChn	Video decoding channel number.	Input
pstParam	Channel parameter	Output

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: cvi_vdec.h, cvi_comm_video.h, cvi_comm_vdec.h
- Library files: libvdec.so/libvdec.a

【Note】

- Ensure that the channel is created before starting the receiving stream. Otherwise, an error is returned.

【Example】

Refer to SAMPLE_COMM_VDEC_Start function in sample_common_vdec.c.

【Related Topic】

- [CVI_VDEC_SetChnParam](#)

8.3.11 CVI_VDEC_SendStream**【Description】**

Send the bitstream data to the video decoding channel.

【Syntax】

```
CVI_S32 CVI_VDEC_SendStream(VDEC_CHN VdChn, const VDEC_STREAM_S *pstStream, CVI_S32 ↵
↵s32MilliSec);
```

【Parameter】

Parameter	Description	Input/Output
VdChn	Video decoding channel number.	Input
pstStream	Decoding bitstream data pointer.	Input
s32MilliSec	Sending bitstream flag. Value range: -1: Blocking.	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: cvi_vdec.h, cvi_comm_video.h, cvi_comm_vdec.h
- Library files: libvdec.so/libvdec.a

【Note】

- Before sending data, it is necessary to ensure CVI_VDEC_CreateChn, CVI_VDEC_StartRecvStream has been called.
- This interface in mjpeg, jpeg decoding state, send length 0(pstStream->u32Len) will return CVI_SUCCESS;
- When decoding fails, the error code ERR_CVI_VDEC_SEND_STREAM will be returned.

【Example】

Refer to SAMPLE_COMM_VDEC_SendStream function in sample_common_vdec.c.

【Related Topic】

- None.

8.3.12 CVI_VDEC_GetFrame

【Description】

Get the decoded image of the video decoding channel.

【Syntax】

```
CVI_S32 CVI_VDEC_GetFrame(VDEC_CHN VdChn,
VIDEO_FRAME_INFO_S *pstFrameInfo, CVI_S32 s32MilliSec);
```

【Parameter】

Parameter	Description	Input/Output
VdChn	Video decoding channel number.	Input
s32MilliSec	Sending bitstream flag. Value range: -1: Blocking.	Input
pstFrameInfo	Pointer to the information of the decoded image.	Output

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: cvi_vdec.h, cvi_comm_video.h, cvi_comm_vdec.h
- Library files: libvdec.so/libvdec.a

【Note】

- Ensure that the channel is created before starting the receiving stream.
Otherwise, an error is returned.

【Example】

Refer to SAMPLE_COMM_VDEC_GetPic function in sample_common_vdec.c.

【Related Topic】

None.

8.3.13 CVI_VDEC_ReleaseFrame

【Description】

Release the decoded image of the video decoding channel.

【Syntax】

```
CVI_S32 CVI_VDEC_ReleaseFrame(VDEC_CHN VdChn, const VIDEO_FRAME_INFO_S *pstFrameInfo);
```

【Parameter】

Parameter	Description	Input/Output
VdChn	Video decoding channel number.	Input
pstFrameInfo	Pointer to the information of the decoded image.	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: `cvi_vdec.h`, `cvi_comm_video.h`, `cvi_comm_vdec.h`
- Library files: `libvdec.so`/`libvdec.a`

【Note】

- This interface needs to be paired with `CVI_VDEC_GetFrame`, and the acquired data should be released immediately after use.

If it is not released in time, the decoding process will be blocked and has to wait for resources.

- The released data must be the data obtained by `CVI_VDEC_GetFrame` from the decoding channel, and no modification to the data information structure is allowed.
- The image of the video decoding channel must be released before the channel is destroyed.

【Example】

None.

【Related Topic】

None.

8.3.14 CVI_VDEC_SetModParam

【Description】

Set decoding-related module parameters

【Syntax】

```
CVI_S32 CVI_VDEC_SetModParam(const VDEC_PARAM_MOD_S *pstModParam);
```

【Parameter】

Parameter	Description	Input/Output
<code>pstModParam</code>	Decoding module parameter pointer.	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: `cvi_vdec.h`, `cvi_comm_video.h`, `cvi_comm_vdec.h`
- Library files: `libvdec.so`/`libvdec.a`

【Note】

- This interface can be called before and after channel creation.

- This interface is mainly used to set the corresponding decoding VB pool acquisition method.

Users can set the VB mode through VB_SOURCE_E type variable in VDEC_PARAM_MOD_S structure.

【Example】

- None.

8.3.15 CVI_VDEC_GetModParam

【Description】

Get decoding-related module parameters

【Syntax】

```
CVI_S32 CVI_VDEC_GetModParam(VDEC_PARAM_MOD_S *pstModParam);
```

【Parameter】

Parameter	Description	Input/Output
pstModParam	Decoding module parameter pointer.	Output

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: cvi_vdec.h, cvi_comm_video.h, cvi_comm_vdec.h
- Library files: libvdec.so/libvdec.a

【Note】

- Usually used before CVI_VDEC_SetModParam.

【Example】

- None.

8.3.16 CVI_VDEC_AttachVbPool

【Description】

Bind the decoding channel to a video buffer VB pool.

【Syntax】

```
CVI_S32 CVI_VDEC_AttachVbPool(VDEC_CHN VdChn, const VDEC_CHN_POOL_S *pstPool);
```

【Parameter】

Parameter	Description	Input/Output
VdChn	Channel number.	Input
pstPool	The Id number of the video buffer VB pool.	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: cvi_vdec.h, cvi_comm_video.h, cvi_comm_vdec.h
- Library files: libvdec.so/libvdec.a

【Note】

- Ensure that the channel has been created, otherwise CVI_FAILURE error code will be returned.
- Users must call the interface CVI_VB_CreatePool to create a video buffer VB pool, and then bind the current encoding channel to the fixed PoolId VB pool by calling the interface CVI_VDEC_AttachVbPool.

Multiple encoding channels can be bound to the same VB pool, but the same encoding channel cannot be bound to multiple VB pools.

- pstPool must be a valid PoolId of the created VB pool, including the VB pool for storing Picture and the VB pool for storing Picture information.
- When calling this interface, the user must make sure that it is set in VB_SOURCE_USER mode through CVI_VDEC_SetModParam.

【Example】

- None.

8.3.17 CVI_VDEC_DetachVbPool

【Description】

Unbind the decoding channel from a video cache VB pool.

【Syntax】

```
CVI_S32 CVI_VDEC_DetachVbPool(VDEC_CHN VdChn);
```

【Parameter】

Parameter	Description	Input/Output
VdChn	Channel number.	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: cvi_vdec.h, cvi_comm_video.h, cvi_comm_vdec.h
- Library files: libvdec.so/libvdec.a

【Note】

- Ensure that the channel has been created, otherwise CVI_FAILURE error code will be returned.

【Example】

- None.

8.4 Data Types

The data types and data structures related to video decoding are defined as follows:

- *VDEC_CHN_ATTR_S*:
- *VDEC_ATTR_VIDEO_S*: Video decoding video channel attributes.
- *VIDEO_MODE_E*: Enumeration of code stream transmission methods.
- *VDEC_CHN_STATUS_S*: Channel state structure.
- *VDEC_DECODE_ERROR_S*: Decoding error information structure.
- *VDEC_CHN_PARAM_S*: Decoding channel advanced parameter structure.
- *VDEC_PARAM_VIDEO_S*: Video decoding advanced parameter structure.
- *VDEC_PARAM_PICTURE_S*: Picture decoding advanced parameter structure.
- *VIDEO_DEC_MODE_E*: Decoding mode enumeration.
- *VIDEO_OUTPUT_ORDER_E*: Decoding output order enumeration.
- *COMPRESS_MODE_E*: Decode image compression mode enumeration.
- *H264_PRTCL_PARAM_S*: H.264 protocol related memory allocation parameters.
- *H265_PRTCL_PARAM_S*: H.265 protocol related memory allocation parameters.
- *VDEC_PRTCL_PARAM_S*: Protocol related memory allocation parameter structure.
- *VDEC_STREAM_S*: Decode the bitstream structure.
- *VDEC_USERDATA_S*: User data structure.
- *VIDEO_DISPLAY_MODE_E*: Display mode enumeration.
- *VDEC_CHN_POOL_S*: Define the VB pool structure bound to the decoding channel.
- *VDEC_CHN_ATTR_S*: Decoding channel attributes.
- *VDEC_ATTR_VIDEO_S*: Video decoding channel properties.
- *VIDEO_MODE_E*: Enumeration of bitstream delivery methods.
- *VDEC_CHN_STATUS_S*: Channel state structure.
- *VDEC_DECODE_ERROR_S*: Decoding error information structure.
- *VDEC_CHN_PARAM_S*: Decoding channel advanced parameter structure.
- *VDEC_PARAM_VIDEO_S*: Video decoding advanced parameter structure.
- *VDEC_PARAM_PICTURE_S*: Picture decoding advanced parameter structure.
- *VIDEO_DEC_MODE_E*: Decoding mode enumeration.
- *VIDEO_OUTPUT_ORDER_E*: Decoding output order enumeration.
- *COMPRESS_MODE_E*: Decode image compression mode enumeration.
- *H264_PRTCL_PARAM_S*: Memory allocation parameters related to H.264 protocol.
- *H265_PRTCL_PARAM_S*: Memory allocation parameters related to H.265 protocol.
- *VDEC_PRTCL_PARAM_S*: Memory allocation parameters related to protocol.
- *VDEC_STREAM_S*: Decoding bitstream structure
- *VDEC_USERDATA_S*: User data structure.
- *VIDEO_DISPLAY_MODE_E*: Display mode enumeration.
- *VDEC_CHN_POOL_S*: Define the structure of VB pool bound to the decoding channel.

8.4.1 VDEC_CHN_ATTR_S

【Description】

Define the decoding channel attribute structure.

【Syntax】

```
typedef struct _VDEC_CHN_ATTR_S {
    PAYLOAD_TYPE_E enType; /* RW; video type to be decoded */
    VIDEO_MODE_E enMode; /* RW; send by stream or by frame */
    CVI_U32 u32PicWidth; /* RW; max pic width */
    CVI_U32 u32PicHeight; /* RW; max pic height */
    CVI_U32 u32StreamBufSize; /* RW; stream buffer size(Byte) */
    CVI_U32 u32FrameBufSize; /* RW; frame buffer size(Byte) */
    CVI_U32 u32FrameBufCnt;
    union {
        VDEC_ATTR_VIDEO_S
        stVdecVideoAttr; /* structure with video ( h264/h265) */
    };
} VDEC_CHN_ATTR_S;
```

【Member】

Member	Description
enType	Decode protocol type enumeration. Video Codec common enumeration: PT_JPEG/ PT_H264/ PT_H265/ PT_MJPEG
enMode	Bitstream delivery mode. Currently only support VIDEO_MODE_FRAME
u32PicWidth	Maximum width of decoded image supported by channel (in pixels)
u32PicHeight	Maximum height of decoded image supported by channel (in pixels)
u32StreamBufSize	The size of the bitstream buffer. (u32StreamBufSize = u32Width * u32Height)
u32FrameBufSize	Size of buffer for storing decoded image frames (varies depending on enType).
u32FrameBufCnt	The number of decoded image frames.
stVdecVideoAttr	Video (H.264/H.265) decoding channel properties.

【Note】

CV181x VDEC module only supports PT_JPEG/PT_MJPEG/PT_H264,CV180x only supports PT_JPEG/PT_MJPEG.

【Related Data Type and Interface】

None.

8.4.2 VDEC_ATTR_VIDEO_S

【Description】

Define the video decoding video channel attributes.

【Syntax】

```
typedef struct _VDEC_ATTR_VIDEO_S {
    CVI_U32 u32RefFrameNum;
    CVI_BOOL bTemporalMvpEnable;
    CVI_U32 u32TmvBufSize;
} VDEC_ATTR_VIDEO_S;
```

【Member】

Member	Description
u32RefFrameNum	The number of reference frames. (currently not supported)
bTemporalMvpEnable	Whether time domain motion vector prediction is supported. (currently not supported)
u32TmvBufSize	The size of Tmv Buffer of video decoded image. (currently not supported)

【Note】

None.

【Related Data Type and Interface】

None.

8.4.3 VIDEO_MODE_E

【Description】

Define the bitstream delivery mode.

【Syntax】

```
typedef enum _VIDEO_MODE_E {
    VIDEO_MODE_STREAM = 0, /* send by stream */
    VIDEO_MODE_FRAME, /* send by frame */
    VIDEO_MODE_COMPAT, /* One frame supports multiple packets sending. */
    /* The current frame is considered to end when bEndOfFrame is equal to HI_TRUE */
    VIDEO_MODE_BUTT
} VIDEO_MODE_E;
```

【Member】

Member	Description
VIDEO_MODE_STREAM	Send the bitstream as a stream. This mode is not supported for JPEG/MJPEG decoding (currently not supported)
VIDEO_MODE_FRAME	The bitstream is sent in frame mode.
VIDEO_MODE_COMPAT	Sending bitstream in compatibility mode. (Currently not supported)

【Note】

None.

【Related Data Type and Interface】

None.

8.4.4 VDEC_CHN_STATUS_S

【Description】

Define the channel state structure.

【Syntax】

```
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;
    CVI_U32 u32DecodeStreamFrames;
    VDEC_DECODE_ERROR_S stVdecDecErr;
    CVI_U32 u32Width;
    CVI_U32 u32Height;
} VDEC_CHN_STATUS_S;
```

【Member】

Member	Description
enType	Decoding protocol type enumeration. Video Codec common enumeration: PT_JPEG/ PT_H264/ PT_H265/ PT_MJPEG
u32LeftStreamBytes	The number of bytes to be decoded in the bitstream buffer, including the number of undeciphered bytes in the current frame being decoded.
u32LeftStreamFrames	The number of frames to be decoded in the bitstream buffer, excluding the current frame being decoded. 1 means invalid.
u32LeftPics	The number of pic remaining in the image buffer.
bStartRecvStream	Whether the decoder has started receiving the bitstream.
u32RecvStreamFrames	The number of received frames in the bitstream buffer. - 1 means invalid.
u32DecodeStreamFrames	The number of decoded frames in the bitstream buffer.
stVdecDecErr	Decoding error message.
u32Width	image width
u32Height	image height.

【Note】

CV181x VDEC module only supports PT_JPEG/PT_MJPEG/PT_H264,CV180x only supports PT_JPEG/PT_MJPEG.

【Related Data Type and Interface】

None.

8.4.5 VDEC_DECODE_ERROR_S

【Description】

Define the decoding error information structure.

【Syntax】

```
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;
```

【Member】

Member	Description
s32FormatErr	Unsupported format.
s32PicSizeErrSet	The width (or height) of the image is larger than that of the channel.
s32StreamUnsprt	Unsupported specification (the code stream specification is inconsistent with the specification claimed by the chip).
s32PackErr	There is an error in the bit stream.
s32PrtclNumErrSet	The number of protocol parameters set is insufficient. For example, the number of Slice, Pps and Sps.
s32RefErrSet	The number of reference frames set is insufficient.
s32PicBufSizeErrSet	The memory size of image buffer is insufficient.
s32StreamSizeOver	One frame code stream is too large. When the whole SCDBuffer can not hold the next frame code stream, the SCDBuffer is forced to be cleared.
s32VdecStreamNotRelease	VFMW internal management stream error. The bitstream is held for a long time without being released.

【Note】

None.

【Related Data Type and Interface】

None.

8.4.6 VDEC_CHN_PARAM_S

【Description】

Define advanced parameters of decoding channel.

【Syntax】

```
typedef struct _VDEC_CHN_PARAM_S {
    PAYLOAD_TYPE_E enType;
    CVI_U32 u32DisplayFrameNum;
    union {
```

(continues on next page)

(continued from previous page)

```

VDEC_PARAM_VIDEO_S
stVdecVideoParam;
VDEC_PARAM_PICTURE_S
stVdecPictureParam;
};
} VDEC_CHN_PARAM_S;

```

【Member】

Member	Description
enType	Common enumeration: PT_JPEG/ PT_H264/ PT_H265/ PT_MJPEG
u32DisplayFrameNum	The minimum number of frames to decode the cached image.
stVdecVideoParam	Video (H.264 / H.265) decoding advanced parameters.
stVdecPictureParam	Picture (JPEG/MJPEG) decoding advanced parameters

【Note】

None.

【Related Data Type and Interface】

- CVI_VDEC_GetChnParam
- CVI_VDEC_SetChnParam

8.4.7 VDEC_PARAM_VIDEO_S

【Description】

Define advanced parameters of video decoding.

【Syntax】

```

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;

```

【Member】

Member	Description
s32ErrThreshold	Error threshold. Value range: [0, 100]. 0 stands for losing when there is an error, 100 stands for none.
enDecMode	Decoding mode:
enOutputOrder	Decoding image output order.
enCompressMode	Decoding image compression mode. COMPRESS_MODE_NONE = 0, COMPRESS_MODE_TILE, COMPRESS_MODE_LINE, COMPRESS_MODE_FRAME,
enVideoFormat	Decoding image data format.

【Note】

None.

【Related Data Type and Interface】

This parameter setting is related to VI module.

8.4.8 VDEC_PARAM_PICTURE_S**【Description】**

Define advanced parameters of graphics decoding.

【Syntax】

```
typedef struct _VDEC_PARAM_PICTURE_S {
    PIXEL_FORMAT_E enPixelFormat;
    CVI_U32 u32Alpha;
} VDEC_PARAM_PICTURE_S;
```

【Member】

Member	Description
enPixelFormat	JPEG (MJPEG) decoding output format. Please refer to: typedef enum _PIXEL_FORMAT_E in cvi_comm_video.h
u32Alpha	Global alpha when outputting in ARGB format, only valid when outputting in ARGB (currently not supported).

【Note】

None.

【Related Data Type and Interface】

None.

8.4.9 VIDEO_DEC_MODE_E**【Description】**

Define the video decoding mode enumeration.

【Syntax】

```
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;
```

【Member】

Member	Description
VIDEO_DEC_MODE_IPB	In IPB mode, i.e. I, P and B frames are all decoded.
VIDEO_DEC_MODE_IP	IP mode. Only I frames and P frames are decoded.
VIDEO_DEC_MODE_I	I mode. Only I frames are decoded.

【Note】

None.

【Related Data Type and Interface】

None.

8.4.10 VIDEO_OUTPUT_ORDER_E

【Description】

Define the video decoding output order enumeration.

【Syntax】

```
typedef enum _VIDEO_OUTPUT_ORDER_E {
    VIDEO_OUTPUT_ORDER_DISP = 0,
    VIDEO_OUTPUT_ORDER_DEC,
    VIDEO_OUTPUT_ORDER_BUTT
} VIDEO_OUTPUT_ORDER_E;
```

【Member】

Member	Description
VIDEO_OUTPUT_ORDER_DISP	Display order output.
VIDEO_OUTPUT_ORDER_DEC	Decoding order output.

【Note】

The decoded bitstream with B frame should be set to display order output.

【Related Data Type and Interface】

None.

8.4.11 COMPRESS_MODE_E

【Description】

Define an enumeration of decoding image compression modes.

【Syntax】

```
typedef enum _COMPRESS_MODE_E {
    COMPRESS_MODE_NONE = 0,
    COMPRESS_MODE_TILE,
    COMPRESS_MODE_LINE,
    COMPRESS_MODE_FRAME,
    COMPRESS_MODE_BUTT
} COMPRESS_MODE_E;
```

【Member】

Member	Description
COMPRESS_MODE_NONE	No compression.
COMPRESS_MODE_TILE	Reserved
COMPRESS_MODE_LINE	Reserved
COMPRESS_MODE_FRAME	Reserved

【Note】

None.

【Related Data Type and Interface】

None.

8.4.12 H264_PRTCL_PARAM_S

【Description】

Memory allocation parameters related to H.264 protocol.

【Syntax】

```
typedef struct _H264_PRTCL_PARAM_S {
    CVI_S32 s32MaxSliceNum; /* RW; max slice num support */
    CVI_S32 s32MaxSpsNum; /* RW; max sps num support */
    CVI_S32 s32MaxPpsNum; /* RW; max pps num support */
} H264_PRTCL_PARAM_S;
```

【Member】

Member	Description
s32MaxSliceNum	The maximum number of Slice supported by the channel decoding.
s32MaxSpsNum	The maximum number of SPS supported by the channel decoding.
s32MaxPpsNum	The maximum number of PPS supported by the channel decoding.

【Note】

None.

【Related Data Type and Interface】

- [VDEC_PRTCL_PARAM_S](#)

8.4.13 H265_PRTCL_PARAM_S

【Description】

Memory allocation parameters related to H.265 protocol.

【Syntax】

```
typedef struct _H265_PRTCL_PARAM_S {
    CVI_S32 s32MaxSliceSegmentNum; /* RW; max slice segmnet num support */
    CVI_S32 s32MaxVpsNum; /* RW; max vps num support */
    CVI_S32 s32MaxSpsNum; /* RW; max sps num support */
    CVI_S32 s32MaxPpsNum; /* RW; max pps num support */
} H265_PRTCL_PARAM_S;
```

【Member】

Member	Description
s32MaxSliceSegmentNum	The maximum number of SliceSegment supported by the channel decoding.
s32MaxVpsNum	The maximum number of VPS supported by the channel decoding.
s32MaxSpsNum	The maximum number of SPS supported by the channel decoding.
s32MaxPpsNum	The maximum number of PPS supported by the channel decoding.

【Note】

None.

【Related Data Type and Interface】

- *VDEC_PRTCL_PARAM_S*

8.4.14 VDEC_PRTCL_PARAM_S

【Description】

Memory allocation parameters related to protocol.

【Syntax】

```
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;
```

【Member】

Member	Description
enType	Decoding protocol supported by channel.
stH264PrtclParam	H.264 protocol parameter
stH265PrtclParam	H.265 protocol parameter

【Note】

CV181x VDEC module only supports PT_JPEG/PT_MJPEG/PT_H264,CV180x only supports PT_JPEG/PT_MJPEG.

【Related Data Type and Interface】

None.

8.4.15 VDEC_STREAM_S

【Description】

Define the bitstream structure of video decoding.

【Syntax】

```
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;
```

【Member】

Member	Description
u32Len	The length of the stream packet.
u64PTS	The timestamp of the stream packet.
bEndOfFrame	Whether the current frame ends.
bEndOfStream	Whether all the bitstream has been sent.
pu8Addr	The address of the stream packet.
bDisplay	Whether the current frame is output for display.

【Note】

None.

【Related Data Type and Interface】

None.

8.4.16 VDEC_USERDATA_S

【Description】

Define the user data structure.

【Syntax】

```
typedef struct _VDEC_USERDATA_S {
    CVI_U64 u64PhyAddr;
    CVI_U32 u32Len;
    CVI_BOOL bValid;
    CVI_U8 *pu8Addr;
} VDEC_USERDATA_S;
```

【Member】

Member	Description
u32PhyAddr	The physical address of the user data
u32Len	The length of user data.
bValid	Valid identifier of the current data.
pu8Addr	The virtual address of the user data.

【Note】

None.

【Related Data Type and Interface】

None.

8.4.17 VIDEO_DISPLAY_MODE_E

【Description】

Defines the display mode enumeration.

【Syntax】

```
typedef enum _VIDEO_DISPLAY_MODE_E {
    VIDEO_DISPLAY_MODE_PREVIEW = 0x0,
    VIDEO_DISPLAY_MODE_PLAYBACK = 0x1,
    VIDEO_DISPLAY_MODE_MAX
} VIDEO_DISPLAY_MODE_E;
```

【Member】

Member	Description
VIDEO_DISPLAY_MODE_PREVIEW	Preview mode.
VIDEO_DISPLAY_MODE_PLAYBACK	Playback mode.

【Note】

None.

【Related Data Type and Interface】

None.

8.4.18 VDEC_PARAM_MOD_S

【Description】

Parameters related to decoding modules.

【Syntax】

```
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;
```

【Member】

Member	Description
enVdecVBSource	Source of decoding frame buffer (VB). Value range: only supported VB_SOURCE_COMMON, VB_SOURCE_USER
u32MiniBufMode	Stream buffer configuration mode
u32ParallelMode	VDH decoding mode
stVideoModParam	Parameters of video decoding module. Invalid for JPEG / MJPEG.
stPictureModParam	Picture decoding module parameters. Invalid for H264/H265.

【Note】

None.

【Related Data Type and Interface】

- CVI_VDEC_SetModParam
- CVI_VDEC_GetModParam

8.4.19 VDEC_VIDEO_MOD_PARAM_S

【Description】

Define the parameter structure of video decoding module

【Syntax】

```
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;
```

【Member】

Member	Description
u32MaxPicWidth	Maximum width supported by video decoding. Value range: See Table 7-1, the minimum value is the minimum width of the resolution supported by H.264/H.265 decoding. The maximum value is the maximum width of the resolution supported by H.264/H.265 decoding. The default value is the maximum value.
u32MaxPicHeight	Maximum height supported by video decoding. Value range: See Table 7-1, the minimum value is the minimum width of the resolution supported by H.264/H.265 decoding. The maximum value is the maximum width of the resolution supported by H.264/H.265 decoding. The default value is the maximum value.
u32MaxSliceNum	Maximum number of slice supported by H.264/H.265 decoding. Value range: the minimum value is 1, and the maximum value is the maximum number of slice supported by H.264/H.265 decoding. The default value is the maximum value.
u32VdhMsgNum	The number of VDH decoded message pools.
u32VdhBinSize	Size of the buffer used to cache bin data for VDH decoding.
u32VdhExtMemLevel	Level of external memory allocation for VDH decoding.

【Note】

CV812x only supports H264 decoding.

【Related Data Type and Interface】

- CVI_VDEC_SetModParam
- CVI_VDEC_GetModParam

8.4.20 VDEC_PICTURE_MOD_PARAM_S

【Description】

Define the parameter structure of picture decoding module.

【Syntax】

```
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;
```

【Member】

Member	Description
u32MaxPicWidth	The maximum width supported by image decoding, with a default value of the maximum width supported by the current chip. Value range: the minimum value is the minimum width supported by JPEG/MJPEG decoding, and the maximum value is the maximum width supported by JPEG/MJPEG decoding. The default value is the maximum value.
u32MaxPicHeight	The maximum height supported by the image decoding, with a default value of the maximum height supported by the current chip. The value range is from the minimum width of the supported resolution for JPEG/MJPEG decoding to the maximum width of the supported resolution for JPEG/MJPEG decoding, with a default value of the maximum width.
bSupportProgressive	Whether JPEG / MJPEG decoding supports progressive format.
bDynamicAllocate	When JPEG / MJPEG decoding supports progressive format, the required buf allocation method is 0 by default.
enCapStrategy	The maximum width and height capability set strategy of decoding image.

【Note】

None.

【Related Data Type and Interface】

- CVI_VDEC_SetModParam
- CVI_VDEC_GetModParam

8.4.21 VDEC_CHN_POOL_S

【Description】

Define the VB pool structure bound to the decoding channel.

【Syntax】

```
typedef struct _VDEC_CHN_POOL_S {
    VB_POOL hPicVbPool; /* RW; vb pool id for pic buffer */
    VB_POOL hTmvVbPool; /* RW; vb pool id for tmv buffer */
} VDEC_CHN_POOL_S;
```

【Member】

Member	Description
hPicVbPool	VB pool Poold for storing picture
hTmvVbPool	VB pool Poolld used to store TMV.

【Note】

None.

【Related Data Type and Interface】

- CVI_VDEC_AttachVbPool

8.5 Error Codes

The decoding error code is shown in the table below

Error Code	Macro Definition	Description
0xC0058002	CVI_ERR_VDEC_INVALID_CHANID	Invalid Channel ID.
0xC0058003	CVI_ERR_VDEC_ILLEGAL_PARAM	Illegal parameter
0xC0058004	CVI_ERR_VDEC_EXIST	The channel already exists
0xC0058005	CVI_ERR_VDEC_UNEXIST	Channel not created
0xC0058006	CVI_ERR_VDEC_NULL_PTR	Null pointer
0xC0058007	CVI_ERR_VDEC_NOT_CONFIG	Not configured before use
0xC0058008	CVI_ERR_VDEC_NOT_SUPPORT	Parameter or function not supported
0xC0058009	CVI_ERR_VDEC_NOT_PERM	Operation not permitted
0xC005800C	CVI_ERR_VDEC_NOMEM	Memory allocation failure
0xC005800D	CVI_ERR_VDEC_NOBUF	Buffer allocation failure
0xC005800E	CVI_ERR_VDEC_BUF_EMPTY	No data in buffer
0xC005800F	CVI_ERR_VDEC_BUF_FULL	Buffer full data
0xC0058010	CVI_ERR_VDEC_SYS_NOTREADY	System is not initialized The related module is not loaded
0xC0058011	CVI_ERR_VDEC_BADADDR	Address error
0xC0058012	CVI_ERR_VDEC_BUSY	The system is busy

REGIONAL MANAGEMENT

9.1 Function Overview

9.1.1 Objective

The REGION module is designed to provide users with the ability to overlay OSD (On-Screen Display) on video, allowing them to display information such as time, channel number, location, or overlay specific images, and fill color blocks.

These overlaid layers on the video are collectively referred to as regions.

9.1.2 Definitions and Abbreviations

RGN (REGION region)

9.2 Design Overview

9.2.1 System Architecture

The Region Management feature enables the creation of regions, controls which video they are overlaid on, and allows temporary hiding or moving the overlay to another video.

- Regional Type
 - Overlay:

The Video Overlay Region supports features such as loading Bitmap images and updating background colors.
 - OverlayEx:

The extended Video Overlay Region supports features such as loading Bitmap images and updating background colors, and also provides support for hierarchical layers of regions, distinguishing it from basic overlays.
 - Cover:

The Video Masking Region supports masking with solid color blocks.
 - CoverEx:

The extended video masking region that supports solid color masking and differs from Cover by supporting region hierarchy.
 - Mosaic:

Not supported at the moment
- Regional Hierarchy

Region hierarchy represents the overlay level of a region, where a higher hierarchy value indicates a higher display priority.

In case of overlapping regions, the region with a higher hierarchy value will be displayed on top of the region with a lower hierarchy value.

This feature is only supported by the OverlayEx and CoverEx region types.

- Bitmap Map Loading

This means to fill the Bitmap data into the memory space of the region.

If the format of Bitmap and region is different, for example, Bitmap is ARGB8888 and region is ARGB1555, some data conversion needs to be done.

The Bitmap will be filled from the top left corner of memory, and the Bitmap cannot be larger than regional memory.

There are two ways of Bitmap map loading :

1. Copies the bitmap to the area canvas memory through CVI_RGN_SetBitMap.
2. Get the memory address of the area canvas through CVI_RGN_GetCanvasInfo and update it on the canvas directly. After the update is completed, update the canvas to the displayed canvas by through CVI_RGN_UpdateCanvas.

- Region Properties

When you create an area, you need to set its properties.

Take OerLayEx as an example, properties to be set include pixel format, size and background color.

- Channel Properties

Channel properties define the display properties of a region on a bound channel.

For example, the channel properties of OverLayEx contain the display location.

9.2.2 Note

The modules supported by the Region feature. Note that the VO module is not supported by CV180X.

Type	S upported module	Device number range	Channel number range
OVERLAY	VPSS	[0, VP SS_MAX_GRP_NUM -1]	[0, VP SS_MAX_PHY_CHN_NUM -1]
	VO	[0, VO _MAX_LAYER_NUM -1]	[0, VO_MAX_CHN_NUM -1]
OVERLAYEX	VPSS	[0, VP SS_MAX_GRP_NUM -1]	[0, VP SS_MAX_PHY_CHN_NUM -1]
COVER	VPSS	[0, VP SS_MAX_GRP_NUM -1]	[0, VP SS_MAX_PHY_CHN_NUM -1]
	VO	[0, VO _MAX_LAYER_NUM -1]	[0, VO_MAX_CHN_NUM -1]
COVEREX	VPSS	[0, VP SS_MAX_GRP_NUM -1]	[0, VP SS_MAX_PHY_CHN_NUM -1]

Function	OVER-LAYEX		COVER		COVEREX	
Module	VPSS	VO	VPSS	VO	VPSS	VO
Pixelformat	ARGB1555 ARGB4444 ARGB8888	ARGB1555 ARGB4444 ARGB8888	ARGB1555 ARGB4444 ARGB8888	ARGB1555 ARGB4444 ARGB8888	ARGB1555 ARGB4444 ARGB8888	ARGB1555 ARGB4444 ARGB8888
Overlay hierarchy	N/A	N/A	Supported	N/A	N/A	Supported
Bitmap fill	Supported	Supported	Supported	N/A	N/A	N/A

9.3 API Reference

This function module provides the following APIs for users

- *CVI_RGN_Create*: Create an region.
- *CVI_RGN_Destroy*: Destroy an region.
- *CVI_RGN_GetAttr*: Gets the region properties.
- *CVI_RGN_SetAttr*: Set the region properties.
- *CVI_RGN_SetBitMap*: Set the region bitmap.
- *CVI_RGN_AttachToChn*: Apply a region overlay to a channel.
- *CVI_RGN_DetachFromChn*: Remove a region overlay from a channel.
- *CVI_RGN_SetDisplayAttr*: Set the channel display properties of the region.
- *CVI_RGN_GetDisplayAttr*: Get the channel display properties of the region.
- *CVI_RGN_GetCanvasInfo*: Get the region canvas information.
- *CVI_RGN_UpdateCanvas*: Update region canvas information.
- *CVI_RGN_SetChnPalette*: Setting channel color palette information

9.3.1 CVI_RGN_Create

【Description】

Create an region

【Syntax】

```
CVI_S32 CVI_RGN_Create(RGN_HANDLE Handle, const RGN_ATTR_S *pstRegion);
```

【Parameter】

Parameter	Description	Input/Output
Handle	Region code. Must be an unused handle number Value range::[0, RGN_HANDLE_MAX)。	Input
pstRegion	Region property pointer.	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: `cvi_comm_region.h`, `cvi_region.h`
- Library files: `libvpu.a`

【Note】

- Handle is specified by the user and must be unique and non repeatable.
- Duplicate creation is not supported
- When creating region outside of Overlay/OverLayEx, you only need to specify the region type.
Additional information is specified when `CVI_RGN_AttachToChn` is called.

【Example】

```

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_Destroy (Handle);
if (s32Ret != CVI_SUCCESS) {
    return CVI_FAILURE;
}

```

【Related Topic】

- [*CVI_RGN_Destroy*](#)

9.3.2 CVI_RGN_Destroy**【Description】**

Destroy an area

【Syntax】

```
CVI_S32 CVI_RGN_Destroy(RGN_HANDLE Handle);
```

【Parameter】

Parameter	Description	Input/Output
Handle	Region code. Value range:[0, 0xFFFFFFFF].	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: `cvi_comm_region.h`, `cvi_region.h`
- Library files: `libvpu.a`

【Note】

- Region must have been created

【Example】

Please refer to the example of *CVI_RGN_Create*.

【Related Topic】

- *CVI_RGN_Create*

9.3.3 CVI_RGN_GetAttr

【Description】

Get region attribute

【Syntax】

```
CVI_S32 CVI_RGN_GetAttr(RGN_HANDLE Handle, RGN_ATTR_S *pstRegion);
```

【Parameter】

Parameter	Description	Input/Output
Handle	Region code. Value range:[0, RGN_HANDLE_MAX)。	Input
pstRegion	Region attribute pointer	Output

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: `cvi_comm_region.h`, `cvi_region.h`
- Library files: `libvpu.a`

【Note】

- Region must have been created

【Example】

Please refer to the example of *CVI_RGN_Create*.

【Related Topic】

- *CVI_RGN_SetAttr*

9.3.4 CVI_RGN_SetAttr

【Description】

Set region attributes

【Syntax】

```
CVI_S32 CVI_RGN_SetAttr(RGN_HANDLE Handle, const RGN_ATTR_S *pstRegion);
```

【Parameter】

Parameter	Description	Input/Output
Handle	Region number Value range:[0, RGN_HANDLE_MAX)。	Input
pstRegion	Region attribute pointer	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: *cvi_comm_region.h*, *cvi_region.h*
- Library files: *libvpu.a*

【Note】

- Region must have been created
- This interface is only supported by Overlay/OverlayEx

【Example】

Please refer to the example of *CVI_RGN_Create*.

【Related Topic】

- *CVI_RGN_GetAttr*

9.3.5 CVI_RGN_SetBitMap

【Description】

Set region bitmap and fill the region with a bitmap

【Syntax】

```
CVI_S32 CVI_RGN_SetBitMap(RGN_HANDLE Handle, const BITMAP_S *pstBitmap);
```

【Parameter】

Parameter	Description	Input/Output
Handle	region code. Value range: [0, 0xFFFFFFFF].	Input
pstBitmap	Bitmap attribute pointer Please refer to BITMAP_S of the chapter of system control	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: cvi_comm_region.h、cvi_region.h
- Library files: libvpu.a

【Note】

- Region must have been created
- The Bitmap must be smaller than the size of the region
- The pixel formats of Bitmap and region must be consistent
- Can be repeated called.
- This interface is only supported by Overlay/OverlayEx

【Example】

None.

【Related Topic】

None.

9.3.6 CVI_RGN_AttachToChn

【Description】

Bind the region to the channel

【Syntax】

```
CVI_S32 CVI_RGN_AttachToChn(RGN_HANDLE Handle, const MMF_CHN_S *pstChn, const RGN_CHN_
↳ATTR_S *pstChnAttr);
```

【Parameter】

Parameter	Description	Input/Output
Handle	region code Value range: [0, 0xFFFFFFFF].	Input
pstChn	Channel information pointer	Input
pstChnAttr	Region channel attribute pointer	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: `cvi_comm_region.h`, `cvi_region.h`
- Library files: `libvpu.a`

【Note】

- The region must have been created
- The channel must have been created

【Example】

None.

【Related Topic】

- [*CVI_RGN_DetachFromChn*](#)

9.3.7 CVI_RGN_DetachFromChn

【Description】

Detach the region from the channel

【Syntax】

```
CVI_S32 CVI_RGN_DetachFromChn(RGN_HANDLE Handle, const MMF_CHN_S *pstChn);
```

【Parameter】

Parameter	Description	Input/Output
Handle	Region code Value range: [0, 0xFFFFFFFF].	Input
pstChn	Channel information pointer	Input
pstChnAttr	Region channel attribute pointer	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: `cvi_comm_region.h`, `cvi_region.h`
- Library files: `libvpu.a`

【Note】

- Region must have been created

【Example】

None.

【Related Topic】

- [*CVI_RGN_AttachToChn*](#)

9.3.8 CVI_RGN_SetDisplayAttr

【Description】

Set the channel display attribute of the region

【Syntax】

```
CVI_S32 CVI_RGN_SetDisplayAttr(RGN_HANDLE Handle, const MMF_CHN_S *pstChn, const RGN_
↪CHN_ATTR_S *pstChnAttr);
```

【Parameter】

Parameter	Description	Input/Output
Handle	Region code Value range: [0, 0xFFFFFFFF].	Input
pstChn	channel information pointer	Input
pstChnAttr	region channel attribute pointer	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: cvi_comm_region.h、cvi_region.h
- Library files: libvpu.a

【Note】

- region must have been created
- region must have been bound to channel

【Example】

None.

【Related Topic】

- [CVI_RGN_GetDisplayAttr](#)

9.3.9 CVI_RGN_GetDisplayAttr

【Description】

Get the channel display attribute of the region

【Syntax】

```
CVI_S32 CVI_RGN_GetDisplayAttr(RGN_HANDLE Handle, const MMF_CHN_S *pstChn, RGN_CHN_
↪ATTR_S *pstChnAttr);
```

【Parameter】

Parameter	Description	Input/Output
Handle	Region code Value range: [0, 0xFFFFFFFF].	Input
pstChn	channel information pointer	Input
pstChnAttr	<i>region channel attribute pointer</i>	Output

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: `cvi_comm_region.h`, `cvi_region.h`
- Library files: `libvpu.a`

【Note】

- region must have been created
- region must have been bound to channel

【Example】

None.

【Related Topic】

- [*CVI_RGN_SetDisplayAttr*](#)

9.3.10 CVI_RGN_GetCanvasInfo

【Description】

Get the canvas information of the region

【Syntax】

```
CVI_S32 CVI_RGN_GetCanvasInfo(RGN_HANDLE Handle, RGN_CANVAS_INFO_S *pstCanvasInfo);
```

【Parameter】

Parameter	Description	Input/Output
Handle	Region code Value range: [0, 0xFFFFFFFF].	Input
pstCanvasInfo	region canvas information pointer	Output

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: `cvi_comm_region.h`, `cvi_region.h`
- Library files: `libvpu.a`

【Note】

- region must have been created
- This interface is only supported by Overlay/OverlayEx
-

- The functionality of this interface is similar to `CVI_RGN_SetBitMap`.

The main difference is that during the canvas update process, `CVI_RGN_SetBitMap` directly affects the bound channel and requires an additional memory copy step.

On the other hand, this interface avoids these issues but requires the use of double buffering to implement.

- This interface is used to obtain canvas information, after which the user can directly manipulate the canvas.

Once the updates are complete, the user can then call `CVI_RGN_UpdateCanvas` to update (swap buffer) the canvas.

- After calling this interface, `CVI_RGN_SetBitMap` cannot be called before `CVI_RGN_UpdateCanvas` is invoked.

【Example】

None.

【Related Topic】

- [*CVI_RGN_UpdateCanvas*](#)

9.3.11 CVI_RGN_UpdateCanvas

【Description】

Update region canvas

【Syntax】

```
CVI_S32 CVI_RGN_UpdateCanvas(RGN_HANDLE Handle);
```

【Parameter】

Parameter	Description	Input/Output
Handle	Region code Value range: [0, 0xFFFFFFFF].	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: `cvi_comm_region.h`, `cvi_region.h`
- Library files: `libvpu.a`

【Note】

- region must have been created
- This interface is used in conjunction with `CVI_RGN_GetCanvasInfo`, mainly for canvas switching after canvas content updates.

This can avoid transient effects during the canvas update process.

- The canvas information must be obtained first, and then after the canvas has been updated, this interface should be called to perform the update.
- This interface is only supported by Overlay/OverlayEx

【Example】

None.

【Related Topic】

- [CVI_RGN_GetCanvasInfo](#)

9.3.12 CVI_RGN_SetChnPalette

【Description】

Set channel color palette information

【Syntax】

```
CVI_S32 CVI_RGN_SetChnPalette(RGN_HANDLE Handle, const MMF_CHN_S *pstChn, RGN_PALETTE_
↪ S *pstPalette);
```

【Parameter】

Parameter	Description	Input/Output
Handle	Region code Value range: [0, 0xFFFFFFFF].	Input
pstChn	Channel information pointer	Input
pstPalette	Channel color palette information	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: cvi_comm_region.h、cvi_region.h
- Library files: libvpu.a

【Note】

- Region must have been created
- This interface is only supported by Overlay/OverlayEx

【Example】

None.

【Related Topic】

None.

9.4 Data Types

9.4.1 RGN_TYPE_E

【Description】

Define region type

【Syntax】

```
typedef enum _RGN_TYPE_E {  
    OVERLAY_RGN = 0,  
    COVER_RGN,  
    COVEREX_RGN,  
    OVERLAYEX_RGN,  
    MOSAIC_RGN,  
    RGN_BUTT  
} RGN_TYPE_E;
```

【Member】

Member	Description
OVERLAY_RGN	Video overlay region.
COVER_RGN	The Video Masking Region
COVEREX_RGN	The extended Video Overlay Region
OVERLAYEX_RGN	The extended Video Masking Region
MOSAIC_RGN	Mosaic video region

【Note】

None.

【Related Data Type and Interface】

None.

9.4.2 RGN_AREA_TYPE_E

【Description】

Define the region type COVER and COVEREX.

【Syntax】

```
typedef enum _RGN_AREA_TYPE_E {  
    AREA_RECT = 0,  
    AREA_QUAD_RANGLE,  
    AREA_BUTT  
} RGN_AREA_TYPE_E;
```

【Member】

Member	Description
AREA_RECT	Rectangle area
AREA_QUAD_RANGLE	Any quadrilateral area, not supported yet.

【Note】

None.

【Related Data Type and Interface】

None.

9.4.3 OSD_COMPRESS_MODE_E

【Description】

Define OSD compression mode type

【Syntax】

```
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;
```

【Member】

Member	Description
OSD_COMPRESS_MODE_NONE	Compression mode is not used.
OSD_COMPRESS_MODE_SW	Use software compression mode.
OSD_COMPRESS_MODE_HW	Use hardware compression mode.

【Note】

Only CV181x/CV180x support OSD compression

【Related Data Type and Interface】

None.

9.4.4 OSD_COMPRESS_INFO_S

【Description】

Define OSD compression mode attributes.

【Syntax】

```
typedef struct _OSD_COMPRESS_INFO_S {
    OSD_COMPRESS_MODE_E enOSDCompressMode;
    CVI_U32 u32EstCompressedSize;
    CVI_U32 u32CompressedSize;
} OSD_COMPRESS_INFO_S;
```

【Member】

Member	Description
enOSDCompressMode	the OSD compression mode type.
u32EstCompressedSize	Estimate the memory size that needs to be allocated in software compression mode.
u32CompressedSize	The memory size needs to be allocated in hardware compression mode.

【Note】

Only CV181x/CV180x support OSD compression

【Related Data Type and Interface】

None.

9.4.5 COVER_CHN_ATTR_S**【Description】**

Define the channel attribute of COVER region.

【Syntax】

```
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;
```

【Member】

Member	Description
enCoverType	COVER region type
stRect	Region location and its width and height. The position can be negative, and the part of the rectangle outside the scope of the channel will not be visible. The width and height cannot exceed the size of the channel.
stQuadRangle	Any quadrilateral area, not supported
u32Color	COVER region color. The format is 24bit RGB888.
u32Layer	Region hierarchy. Not supported at present
enCoordinate	Region coordinate type.

【Note】

None.

【Related Data Type and Interface】

None.

9.4.6 COVEREX_CHN_ATTR_S**【Description】**

Define the channel attribute of the COVEREX region.

【Syntax】

```
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;
    RGN_COORDINATE_E enCoordinate;
} COVEREX_CHN_ATTR_S;
```

(continues on next page)

(continued from previous page)

```
};
CVI_U32 u32Color;
CVI_U32 u32Layer;
} COVEREX_CHN_ATTR_S;
```

【Member】

Member	Description
enCoverType	COVEREX region type
stRect	Region location and its width and height. The position can be negative, and the part of the rectangle outside the scope of the channel will not be visible. The width and height cannot exceed the size of the channel.
stQuadRange	<i>Any quadrilateral area, not supported</i>
u32Color	COVER region color. The format is 24bit RGB888.
u32Layer	Region hierarchy. Not supported at present

【Note】

None.

【Related Data Type and Interface】

None.

9.4.7 OVERLAY_ATTR_S

【Description】

Define the attribute of the Overlay region.

【Syntax】

```
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;
```

【Member】

Member	Description
enPixelFormat	Pixel format. <code>PIXEL_FORMAT_ARGB_1555</code> , <code>PIXEL_FORMAT_ARGB_4444</code> , <code>PIXEL_FORMAT_ARGB_8888</code> .
u32BgColor	Region background color. Defined according to enPixelFormat.
stSize	Region width and height.
u32CanvasNum	Amount of region memory.
stCompressInfo	OSD compression mode information.

【Note】

- The stSize parameter will affect the memory allocation size of the region.

It is recommended not to be larger than the width and height of the final linked channel to avoid memory waste.

- The value of u32CanvasNum should be determined based on the usage scenario:

If using CVI_RGN_SetBitMap, set it to 1.

If using CVI_RGN_GetCanvasInfo and want to avoid transient during the canvas update process using double buffer, it is recommended to set it to 2.

【Related Data Type and Interface】

None.

9.4.8 OVERLAY_CHN_ATTR_S

【Description】

Define the channel attribute of the video overlay area.

【Syntax】

```
typedef struct _OVERLAY_CHN_ATTR_S {
    POINT_S stPoint;
    CVI_U32 u32Layer;
    OVERLAY_INVERT_COLOR_S stInvertColor;
} OVERLAY_CHN_ATTR_S;
```

【Member】

Member	Description
stPoint	Region location and its width and height. The position can be negative, and the part of the rectangle outside the scope of the channel will not be visible. The width and height cannot exceed the size of the channel.
u32Layer	Regional level. Not supported at present
stInvertColor	Color Inversion Structure

【Note】

None.

【Related Data Type and Interface】

None.

9.4.9 OVERLAYEX_ATTR_S

【Description】

Define the attribute of the entended Overlay region.

【Syntax】

```
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;
```


【Member】

Member	Description
enPixelFormat	Pixel format. PIXEL_FORMAT_ARGB_1555, PIXEL_FORMAT_ARGB_4444, PIXEL_FORMAT_ARGB_8888。
u32BgColor	Region background color. Defined according to enPixelFormat.
stSize	Region width and height.
u32CanvasNum	Amount of region memory.
stCompressInfo OSDcompression modeinformation.stCompressInfo	OSD compression mode information.

【Note】

- The stSize parameter will affect the memory allocation size of the region.

It is recommended not to be larger than the width and height of the final linked channel to avoid memory waste.

- The value of u32CanvasNum should be determined based on the usage scenario:

If using CVI_RGN_SetBitMap, set it to 1.

If using CVI_RGN_GetCanvasInfo and want to avoid transient during the canvas update process using double buffer, it is recommended to set it to 2.

【Related Data Type and Interface】

None.

9.4.10 OVERLAYEX_CHN_ATTR_S

【Description】

Define the channel attribute of the extended video overlay region.

【Syntax】

```
typedef struct _OVERLAYEX_CHN_ATTR_S {
    POINT_S stPoint;
    CVI_U32 u32Layer;
    OVERLAY_INVERT_COLOR_S stInvertColor;
} OVERLAYEX_CHN_ATTR_S;
```

【Member】

Member	Description
stPoint	Region location and its width and height. The position can be negative, and the part of the rectangle outside the scope of the channel will not be visible. The width and height cannot exceed the size of the channel.
u32Layer	Regional level.
stInvertColor	Color Inversion Structure

【Note】

None.

【Related Data Type and Interface】

None.

9.4.11 RGN_ATTR_U

【Description】

Define union structure of region.

【Syntax】

```
typedef union _RGN_ATTR_U {
    OVERLAY_ATTR_S stOverlay;
    OVERLAYEX_ATTR_S stOverlayEx;
} RGN_ATTR_U;
```

【Member】

Member	Description
stOverlay	Video overlay region properties.
stOverlayEx	Extended video overlay region properties.

【Note】

Only when RGN_TYPE_E is OverlayEx, this attribute need to be set.

【Related Data Type and Interface】

None.

9.4.12 RGN_CHN_ATTR_U

【Description】

Define union structure of region channel.

【Syntax】

```
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;
```

【Member】

Member	iption
stOverlayChn	Video overlay region channel attribute.
stCoverChn	Video masking region channel attribute.
stCoverExChn	Extended masking region channel attribute.
stOverlayExChn	Extended overlay region channel attribute.
stMosaicChn	Mosaic region channel attribute.

【Note】

None.

【Related Data Type and Interface】

None.

9.4.13 RGN_ATTR_S

【Description】

Define region attribute.

【Syntax】

```
typedef struct _RGN_ATTR_S {  
    RGN_TYPE_E enType;  
    RGN_ATTR_U unAttr;  
} RGN_ATTR_S;
```

【Member】

Member	Description
enType	Region type.
unAttr	Union attribute structure of region

【Note】

None.

【Related Data Type and Interface】

None.

9.4.14 RGN_CHN_ATTR_S

【Description】

Define region channel attribute.

【Syntax】

```
typedef struct _RGN_CHN_ATTR_S {  
    CVI_BOOL bShow;  
    RGN_TYPE_E enType;  
    RGN_CHN_ATTR_U unChnAttr;  
} RGN_CHN_ATTR_S;
```

【Member】

Member	Description
bShow	Whether region is shown or not.
enType	Regional type
unChnAttr	Union attribute structure of region

【Note】

None.

【Related Data Type and Interface】

None.

9.5 Error Codes

The error codes of regional system API are shown in the following table

Error Code	Macro Definition	Description
0xC0038001	CVI_ERR_RGN_INVALID_DEVID	Invalid device number
0xC0038002	CVI_ERR_RGN_INVALID_CHNID	Invalid RGN channel number
0xC0038003	CVI_ERR_RGN_ILLEGAL_PARAM	Invalid VPSS parameter setting
0xC0038004	CVI_ERR_RGN_EXIST	RGN has been created
0xC0038005	CVI_ERR_RGN_UNEXIST	RGN not created
0xC0038006	CVI_ERR_RGN_NULL_PTR	Null pointer
0xC0038007	CVI_ERR_RGN_NOT_CONFIG	Module not configured
0xC0038008	CVI_ERR_RGN_NOT_SUPPORTED	Operation not supported
0xC0038009	CVI_ERR_RGN_NOT_PERM	Operation not permitted
0xC003800c	CVI_ERR_RGN_NOMEM	Failure to allocate memory
0xC003800d	CVI_ERR_RGN_NOBUF	Failure to allocate BUF pool
0xC003800e	CVI_ERR_RGN_BUF_EMPTY	BUF pool is empty
0xC003800f	CVI_ERR_RGN_BUF_FULL	BUF pool is full
0xC0038011	CVI_ERR_RGN_BADADDR	RGN ioctl parameter error
0xC0038012	CVI_ERR_RGN_BUSY	RGN system is busy

AUDIO FREQUENCY

10.1 Function Overview

10.1.1 Objective

10.1.2 Definitions and Abbreviations

Abbreviation	Definition
AI	Audio Input:
AO	Audio Output:
AENC	Audio Encode:
ADEC	Audio Decode
VQE	Voice Quality Enhancement
Middleware	middleware
Abbreviation	Definition
Multimedia Framework	Multimedia Framework
Kernel Layer	Linux kernel layer
RES	Resample
AGC	automatic gain control
ANR	Noise Reduction
AEC	Acoustic echo cancellation
Code Driver	driver

10.2 Design Overview

10.2.1 System Architecture

Please refer to 9.1.2.

The audio module mentioned in this article (audio input module, audio output module, audio encoding module, audio decoding module, audio quality enhancement module, resampler) is one of the Middleware multimedia layer interfaces that interconnect with application layer or customer business layer apis.

Users can control corresponding audio components through functions (CVI_AI, CVI_AO, CVI_ADEC, CVI_AENC_related prefix beginning, etc.).

The Application Layer or customer business layer is called Application/Customize Layer.

Refer to 9.3: API Reference in this document for details.

If you need the related API call logic, refer to the example in cvi_sample_audio.c in the SDK to understand the use sequence among basic audio modules (the following figure).

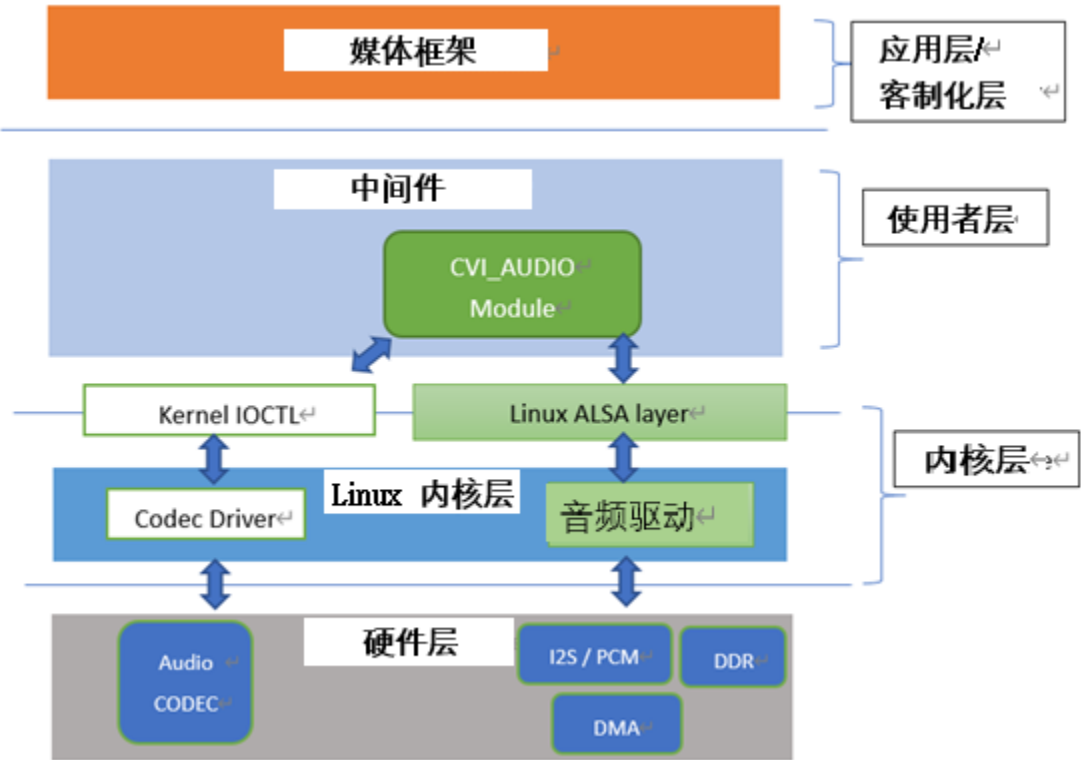


Fig. 10.1: Figure9-1

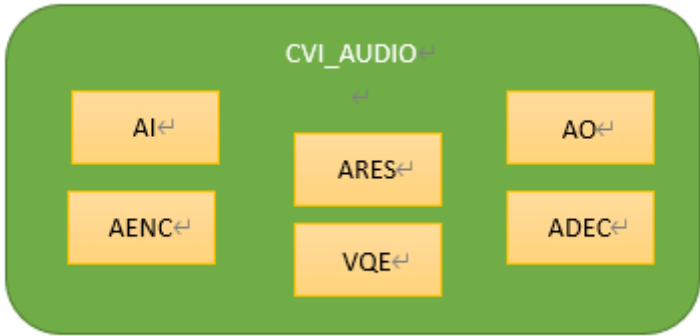


Fig. 10.2: Figure9-2

VI_AUDIO interfaces with Linux kernel layer related driver programs through the Linux standard sound system (ALSA: Advanced Linux Sound Architecture), which enables basic audio input and output functions.

Therefore, it can be understood that when a user calls the `cvi_audio_xxx/ cvi_axx_xx` (e.g. `cvi_adec_xxx`) API, the internal basic and minimum audio unit is a Frame (referred to as an Audio Frame in this document), and the unit of measurement for frames in this document is the number of samples in a frame, rather than bits (bytes).

The size range of a Frame, without using the Voice Quality Enhancement (VQE) module, can be set to 160/320/480 samples as the frame size (not to exceed 512).

When using VQE, the frame size must be set to a multiple of 160 samples, which is designed to be compatible with the internal VQE module.

For users, in addition to the Audio Codec function or related debugging requirements, the basic functions are realized through the `libcvi_audio`, `libcvi_xxx` related so file, and will not directly call the IOCTL function control kernel layer, to ensure the stability and reliability of the system internal resource allocation.

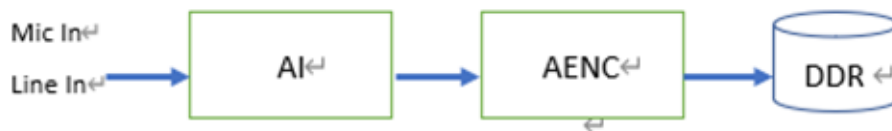


Fig. 10.3: Figure9-3

The figure above depicts the relationship between audio input and encoding.

AENC uses CPU encoding to store the stream in DDR without passing through the core layer, and enables the stream sequentially from the radio end (supporting microphone radio and voice line radio) to the encoding end.

When AI and AENC modules are enabled, the internal API parameters (sampling rate, sound frame size, and number of channels) must be set as consistent as possible;

otherwise, the audio obtained after encoding is abnormal.

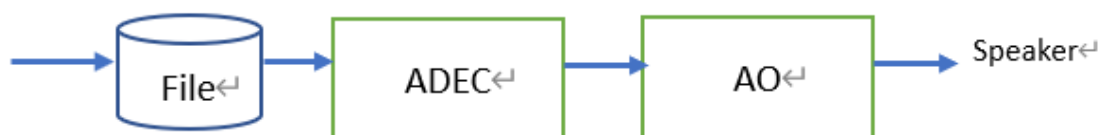


Fig. 10.4: Figure9-4

The figure above describes the relationship between audio decoding and output.

Users can directly put the pcm/raw audio file into the storage device and call ADEC/AO API for broadcasting, and enable it in sequence from the decoding end to the output end.

ADEC and AO parameters should be set as consistent as possible, otherwise the audio will be abnormal.

The figure above illustrates the relationship before and after the VQE.

After the audio is received and before the encoding, the user can enable the AEC/ANR/AGC function through the VQE-related API(see 9-3).

At this time, the unit of the sound frame needs to be set as a multiple of 160 and the operating frequency can be 8KHz/16KHz.

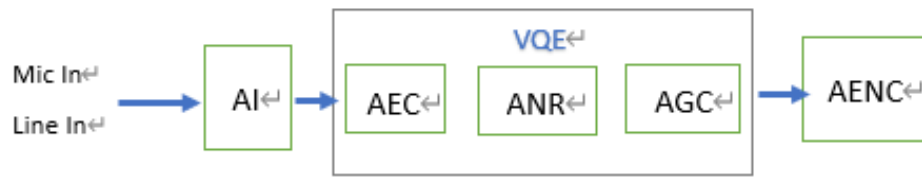


Fig. 10.5: Figure9-5



Fig. 10.6: Figure9-6

VQE includes the front-end VQE (Figure 9-5) and the broadcasting VQE (back-end VQE) described in the figure above.

Currently, back-end VQE is not supported.

10.2.2 Audio Input and Output

10.2.2.1 Audio Interface and AI, AO Device

Audio interface is divided into two types, input (AI) and output (AO) interface, each responsible for recording and playing sound.

The unit that connects with Audio Codec and is responsible for the input function of abstract audio interface is called AI device; AO device is responsible for the output function of abstract audio interface.

According to the functions supported by the interface, it establishes mapping with AI device and AO device respectively.

Audio input / output interface is called AIO (Audio Input / Output) interface for docking with Audio Codec to complete sound recording and playback. AIO interface can be divided into two types: input only or output only.

When it is an input type, it is also called AIP; when it is an output type, it is also called AOP.

If AIP0 only supports audio signal input, AIP0 image is AiDev0; if AOP0 only supports audio signal output, AOP0 image is AoDev0.

The audio input interface (AI) supports PCM and I2S input, and the output is connected with ALSA PCM device according to the Linux kernel Standard Specification.

Cvitek supports a set of preset input and output.

If viewed from the Linux ALSA architecture, the corresponding devices can be regarded as card 0 and card 1, and the relationship is as follows:

IP0 can only support input, AOP0 can only support audio output.

In the case of input and output docking, such as real-time recording and dialing or voice intercom, the sampling rate and bit width of AI / AO equipment must be the same, the number of channels must be the same, and the sampling rate must be the same.

In order to meet the needs of customized products, Cvitek has two more sets of I2S, which can be changed to AiDev (input) pair interface or AoDev (output) pair interface according to the needs of products or customer applications.

10.2.2.2 Principle of Recording and Playing

The audio frames processed by the CVI_AUDIO API are all digital signals. However, in fact, they are analog signals at both the radio and the broadcasting terminals. The digital and analog signals are converted through the Audio Codec. The Audio Codec converts the input analog signal source through I2S or PCM timing and transmits it to the AI module. Similarly, the AO terminal broadcasts through the Audio Codec will do DAC conversion of digital audio with I2S or PCM timing, and then send simulation signal to speaker.

The data transfer is controlled by CPU in DMA mobile memory DDR. When users call CVI_AUDIO API, they only call Audio Codec at the beginning or end, so as to realize the initial and standard end of Audio Codec hardware standard. During the process, it does not involve the capture of simulation signal, nor can it directly change the CPU using DMA mode. Audio Codec is the translator of the working domain, while CPU/DMA is the role of data movers. (see Figure 9-7 below)

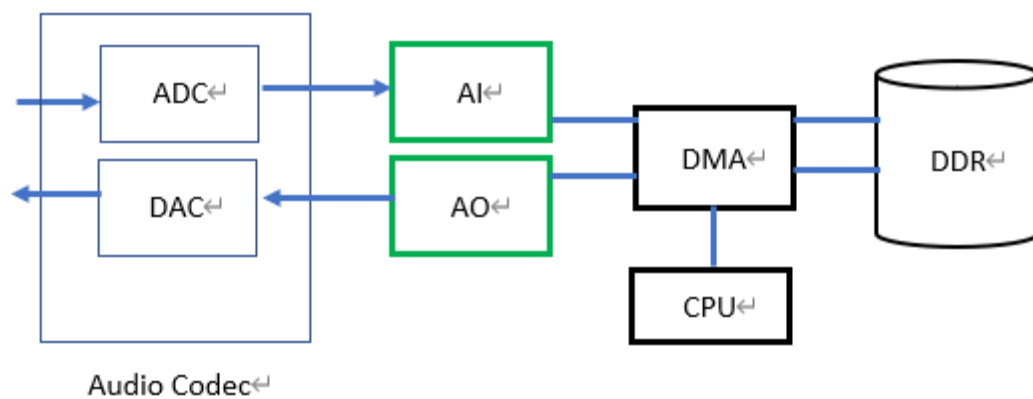


Fig. 10.7: Figure9-7

10.2.2.3 Audio Interface Timing

Cvitek audio timing interface supports I2S, PCM timing mode, and provides a variety of ways to interface with Audio Codec according to customization.

Please refer to hardware related documents for detailed chip hardware specifications.

AI / AO controls the clock to synchronize the timing.

Users can refer to `cvi_sample_comm_audio.c`'s internal `SAMPLE_COMM_AUDIO_Cfg` Acodec API.

The timing setting method of built-in Audio Codec or external Audio Codec will be different.

However, for `cvi_audio` API users, they only need to confirm that the kernel can support and initialize Codec at the time of initial Audio Codec.

For the frame sampling rate, Cvitek uses CPU software to sample, which is not directly related to the master clock.

When AI equipment uses multiplexed I2S receiving mode, the standard I2S protocol only has the concept of left and right channels.

AI equipment can receive 128bit audio data from left and right channels at most.

Please refer to section 9.4.4 for details of Codec.

10.2.2.4 Resample

Voice frame resampling supports conversion of any two different sampling rates, mainly 8kHz frequency doubling.

The input sampling rates supported by resampling are: 8kHz, 11.025kHz, 16kHz, 22.05kHz, 24kHz, 32kHz, 44.1kHz, 48kHz;

the output sampling rates supported are: 8kHz, 11.025kHz, 16kHz, 22.05kHz, 24kHz, 32kHz, 44.1kHz, 48khz.

Users should note that Resampling supports processing mono and stereo audio.

AI resampling, the resampling input sampling rate and AI device property configuration sampling rate are the same, the resampling output sampling rate must be different from AI device property configuration sampling rate;

AO resampling, resampling output sampling rate and AO device property configuration sampling rate are the same, resampling input sampling rate must be different from AO device property configuration

AI-AO data is transmitted by system bind, and resampling of AI or AO is invalid.

Users with AI resampling enabled in user-get mode can use CVI_AI_GetFrame gets the data and gets the corresponding resampled data.

If AO is enabled for resampling, audio data needs to be resampled before being sent to AO, and then sent to AO channel (CVI_AO_SendFrame) to play.

Related API:

- CVI_Resampler_Create: Create and initialize audio resample.
- CVI_Resampler_GetMaxOutputNum: Based on the number of sample points entered, the corresponding number of resampled points is obtained.
- CVI_Resampler_Process: Through this API, the sample number of audio frames is continuously passed in for actual resampling.
- CVI_Resampler_Destroy: Finish the resampling process.

10.2.2.5 Voice Quality Enhancement (VQE)

For the speech signal processing algorithm, when the near end speech signal is interfered by the echo from the far end or the near end stationary noise, the algorithm function in VQE can be used to improve the quality of the speech signal.

VQE provides four solutions, including linear echo cancellation (AEC), nonlinear echo suppression (AES), speech noise reduction (NR) and automatic gain control (AGC).

The algorithm can support 8kHz and 16kHz sampling rate, mono and 16bit sampling length.

The following pages will introduce each function and the parameters used.

For a more detailed understanding, please refer to the Audio Quality Tuning Guide.

Parameter para_fun_config can control the function of AEC、AES、NR and AGC.

The para_spk_fun_config parameter controls the speaker path SSP algorithm function and belongs to UpVQE.

The following table shows the algorithm functions corresponding to each bit.

Table 10.1: para_fun_config parameter description

para_fun_config	Description
bit0:	0: turn offAEC 1: turn onAEC
bit1:	0: turn offAES 1: turn onAES
bit2:	0: turn offNR 1: turn onNR
bit3:	0: turn offAGC 1: turn onAGC
bit4:	0: turn off Notch Filter 1:turn on Notch Filter
bit5:	0: turn offDC Filter 1: turn on DC Filter
bit6:	0: turn off DG 1: turn on DG
bit7:	0: turn off Delay 1: turn on Delay

Table 10.2: para_spk_fun_config parameter description

para_spk_fun_config	Description
bit0:	0: turn off AGC 1: turn on AGC
bit1:	0: turn off EQ 1: turn on EQ

Aiming at the similarities and differences between AI and Ao, VQE processes the data of the two channels through UpVQE and DnVQE scheduling logic respectively.

UpVQE includes AEC, AES, NR and AGC.

DnVQE is currently not supported.

The corresponding parameters can refer to the header file cvi_comm_aio.h.

AGC data structure is as follows:

```
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 data structure is as follows :

```
CVI_U16 para_nr_snr_coeff;
CVI_U16 para_nr_init_sile_time;
```

The AI_TALKVQE_CONFIG_S *pstVqeConfig parameter in the CVI_AI_SetTalkVqeAttr, sets u32OpenMask to decide which function of VQE to turn on.

For a more detailed Description of AGC and NR members, please refer to the corresponding sub-sections below.

Examples are shown in the table below:

Table 10.3: table1: pstVqeConfig parameter description

pstAiVqeAttr.u32OpenMask	Description
pstAiVqeAttr.u32OpenMask = AI_TALKVQE_MASK_AGC;	turn on AGC
pstAiVqeAttr.u32OpenMask = AI_TALKVQE_MASK_ANR;	turn on NR
pstAiVqeAttr.u32OpenMask = (AI_TALKVQE_MASK_ANR AI_TALKVQE_MASK_AGC);	turn on NR, turn on AGC

- AEC/AES (Acoustic Echo Cancellation/Acoustic Echo Suppression)

There is echo interference in the architecture of any duplex communication system.

The echo canceller can eliminate the echo from the speaker output coupled back to the microphone through the near end acoustic path.

The linear adaptive filter module (AEC) combined with the nonlinear echo suppression module (AES) can effectively suppress the echo and improve the quality of voice calls.

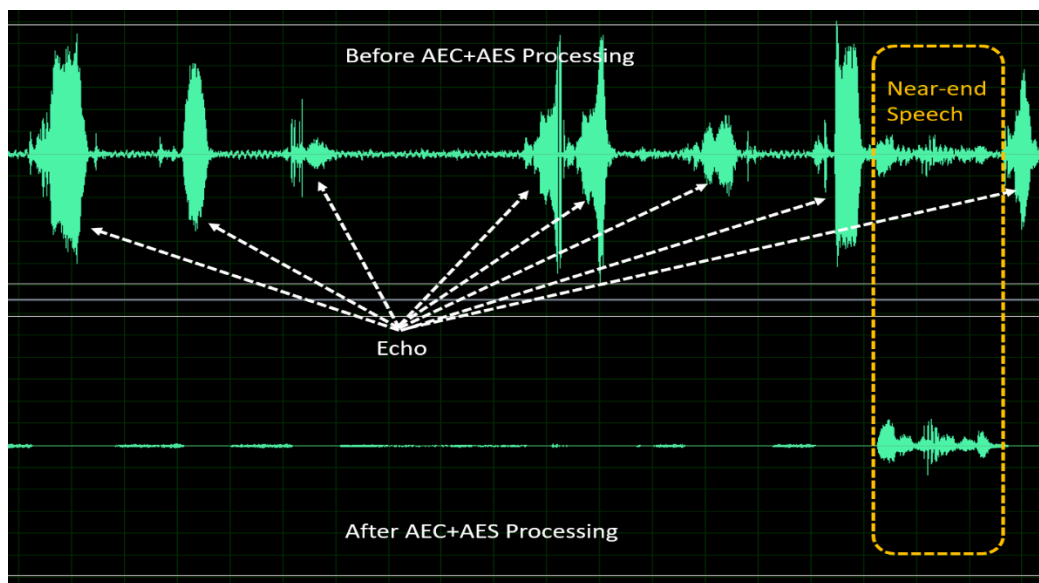


Fig. 10.8: Figure 9-1: Performance before and after AEC + AES treatment

Three adjustable parameters are provided to adjust the performance of AEC / AES:

- para_aec_filter_len:

The length of the adaptive filter.

Adjust the length of filter according to the time of echo tailing.

Choosing a longer length will lead to higher MIPS and power consumption.

- para_aes_std_thrd:

The threshold of residual echo was determined.

When the value is larger, the near end speech quality is better, but the residual echo is more.

On the contrary, when the value is small, the near end voice quality is poor, but the residual echo is less.

- para_aes_supp_coeff:

Residual echo suppression.

The larger the value is, the stronger the residual echo suppression will be, but at the same time, more details will be lost / damaged in the near end speech.

Table 10.4: Table 9-2: AEC/AES parameter description

AEC/AES parameter	Adjustable range	Description
para_aec_filter_len	1 - 13	8kHz sampling rate: [1,13] corresponding to [20ms,260ms] 16kHz sampling rate: [1,13] corresponding to [10ms,130ms]
para_aes_std_thrd	0 - 39	0: minimum threshold of residual echo 39: maximum threshold of residual echo
para_aes_supp_coeff	0 - 100	0: Minimal residual echo suppression 100: Maximum residual echo suppression

- NR (Noise Reduction)

The NR module can suppress the surrounding steady noise, such as fan noise, air conditioning noise, engine noise, white / pink noise.

With the proprietary voice intelligent Voice Activity Detection (VAD) algorithm, NR can maintain the voice signal and effectively suppress the steady noise, so as to improve the quality of voice calls.

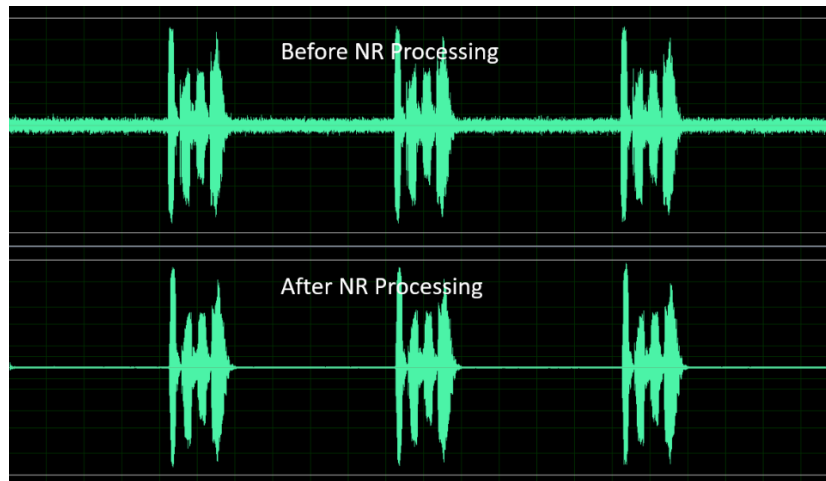


Fig. 10.9: figure 9-2: The performance before and after NR treatment

Three adjustable parameters are provided to adjust the performance of NR:

- para_nr_init_sile_time:

Initial duration of silence.

CODEC will generate random meaningless noise signal when turned on, so it is recommended to set an initial duration of silence to avoid this.

- para_nr_init_sile_time:

You can set this signal to mute.

- para_nr_snr_coeff:

Prior Signal to Noise Ratio (SNR) tracking coefficient.

If the parameter value is larger, NR will have higher noise reduction ability, but the speech signal may be more easily distorted.

On the contrary, if the parameter value is small, NR will suppress less noise signal, but it will have better speech quality performance.

The following table is based on the appropriate adjustment range of this parameter in different SNR environments.

In each SNR case, the larger the parameter value is, the greater the suppression force to stationary noise is.

Table 10.5: Table : NR parameter Description

NR parameter	Adjusting range	Description
para_nr_init_sile_time	0-250	Corresponding to 0s to 5s, with a step of 20ms.

Table 10.6: Table 9-3: para_nr_snr_coeff parameter description

Surrounding SNR environment	Adjusting range	Description
low	0 - 3	0: The least active in noise reduction 3: The most active in noise reduction
medium	4 - 10	4: The least active in noise reduction 10: The most active in noise reduction
high	11 - 20	11: The least active in noise reduction 20: The most active in noise reduction

- AGC (Automatic Gain Control)

AGC module is a signal processing function, which can automatically adjust the output level to a predetermined range to provide a more comfortable hearing experience.

If the input signal is lower than “Target Low” , AGC will adjust the output level to “Target Low”

On the other hand, if the input signal is higher than “Target High” , AGC will adjust the output level to “Target High” .

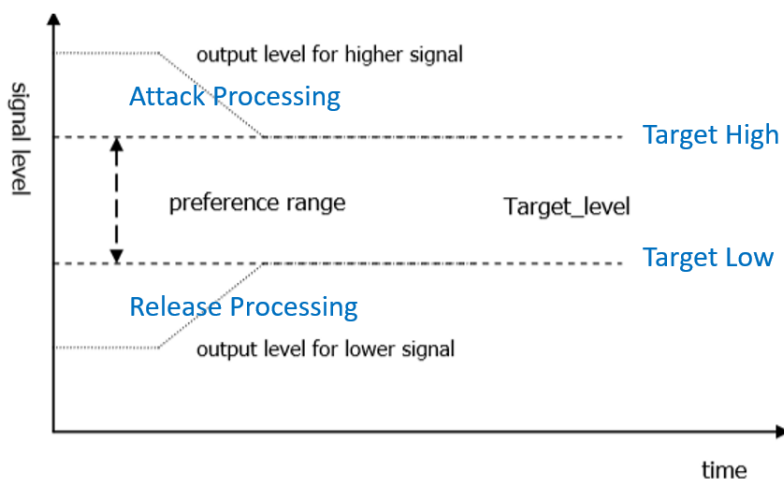


Fig. 10.10: Figure9-3: AGC adjusts signal level

Four adjustable parameters are provided to adjust the performance of AGC

- para_agc_max_gain:
This parameter is the maximum gain that the signal can be amplified.
- para_agc_target_high:



Fig. 10.11: Figure9-4: Performance before and after AGC treatment

This parameter is the “Target High” level AGC will reach.

For the input signal higher than `para_age_target_high`, AGC will converge it to `para_age_target_high`.

- `para_agc_target_low`:

This parameter is the “Target Low” level AGC will reach.

For the input signal lower than `para_age_target_low`, AGC will converge it to `para_age_target_low`.

If `para_age_max_gain` is reached before `para_age_target_low`, AGC will only converge to `para_age_max_gain`.

- `para_agc_vad_ena`:

Speech-activated AGC function.

When this function is turned on and NR and AEC / AES functions are turned on at the same time, AGC can avoid amplifying background stable noise and residual echo, so as to obtain better effect.

Table 10.7: Table9-4: AGC parameter description

AGC parameter	A djustable range	Description
<code>para_agc_max_gain</code>	0 - 6	The maximum lifting gain corresponding to [0,6] is [6dB, 42db], and each order is 6dB
<code>para_agc_target_high</code>	0 - 36	0 to 36 corresponds to 0dB to -36dB
<code>para_agc_target_low</code>	0 - 36	0 to 36 corresponds to 0dB to -36dB
<code>para_agc_vad_ena</code>	0 - 1	0: turn offSpeech-activated AGC function 1: turn onSpeech-activated AGC function

10.2.3 Audio Encoding and Decoding

10.2.3.1 Audio Codec Process

Cvitek audio codec supports G711-A-law, G711-Mu-law, G726, and ADPCM_IMA, the above codec uses CPU software codec.

Users can use bind mode and use CVI_Aud_SYS_Bind, bind AI to AENC to encode the voice frame, or bind AO to ADEC to decode the received encoded voice frame to restore the voice frame to PCM / Raw signal.

If user get mode is used instead of bind mode, users can use CVI_AENC_SendFrame sends the actual single sound frame into the encoding program for encoding, and CVI_ADEC_GetFrame can also be used correspondingly to decode a single audio frame.

Please note that when using user access mode, if the user's call delay is too long, the internal cache may be blocked and the failed audio frame will be returned.

10.2.3.2 Audio Codec Protocol

CodecProtocol (coding)	Bit Rate (Bit RateKbps)	Payload (Suggest original sound frame)	Compress Rate (compress rate)	MOS (Sound quality me asurement)
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 Speech Frame Structure

Cvitek and audio frame structure do not have additional header files.

Internal radio and broadcast are based on audio frame, and each audio frame is RAW / PCM data.

For encoding and decoding in G711, G726 and ADPCM formats, the corresponding audio frame will not add extra header.

Users must pass through CVI_AUDIO_AENC/CVI_AUDIO_ADEC

Related API to get the current codec voice box information.

The advantage of this method is that when the user obtains the sound frame through the get_frame API or the related debug function, he can immediately identify whether the sound frame conforms to the corresponding encoding and decoding rules, without having to parse the header, and can verify whether the encoding and decoding results are normal with the relevant third-party software on the computer after saving.

10.2.4 Built-in Codec

10.2.4.1 Overview

Cvitek AIP0 / AOP0 interface can be connected with built-in audio codec or plug-in audio codec to record and play sound.

Users need to configure the timing and parameters of AIP0 / AOP0 and built-in audio codec to receive or send audio data.

In addition, the built-in audio codec only supports I2S mode.

The ADC codec needs to be configured as the master mode, and the DAC codec needs to be configured as the slave mode.

The built-in audio codec is divided into analog and digital parts.

The analog part can be input by microphone or by LINEIN and support gain adjustment.

The digital part has ADC and DAC, which are mainly responsible for the conversion between analog and digital signals.

ADC can adjust the left and right channel volume respectively, while DAC can't adjust the left and right channel volume separately.

In addition, it supports mute and unmute.

When unmute is cancelled, it will return to the gain value configured before mute.

The MCLK with built-in audio code is provided by CV182X.

The default sampling precision is 16 bit and the sampling frequency is 16Khz.

10.2.4.2 ioctl Function

The user interface with built-in Audio Codec is embodied in the form of IOCTL, which is as follows:

```
CVI_S32 ioctl (CVI_S32 fd, CVI_UL cmd);
```

This function is a standard interface of Linux and has variable parameters.

But in Audio Codec, only three parameters are needed.

Therefore, its grammatical form is equivalent to:

```
CVI_S32 ioctl (CVI_S32 fd, CVI_UL cmd, CMD_DATA_TYPE *cmddata);
```

Among them, CMD_DATA_TYPE changes with the change of parameter cmd.

The detailed description of these three parameters is shown in the table below:

Parameter	Description	Input/Ouput
fd	The built-in audio codec device file descriptor is the return value after calling the open function to open the built-in audio codec device file. ADC and DAC need to be configured separately.	Input
cmd	<p>Main cmd is as follows:</p> <ul style="list-style-type: none"> • ACODEC_SOFT_RESET_CTRL: Restore built-in codec to default settings • ACODEC_SET_INPUT_VOL: Input total volume control • ACODEC_GET_INPUT_VOL: Get the total volume of input • ACODEC_SET_OUTPUT_VOL: Output total volume control • ACODEC_GET_OUTPUT_VOL: Get the total volume of output • ACODEC_SET_GAIN_MICL: Analog gain control of MIC input in left channel • ACODEC_SET_GAIN_MICR: Analog gain control of MIC input in right channel • ACODEC_SET_DACL_VOL: Left channel output volume control • ACODEC_SET_DACR_VOL: right channel output volume control • ACODEC_SET_ADCL_VOL: Left channel input volume control • ACODEC_SET_ADCR_VOL: right channel input volume control • ACODEC_SET_MICL_MUTE: Left channel MIC input mute control • ACODEC_SET_MICR_MUTE: right channel MIC input mute control • ACODEC_SET_DACL_MUTE: Left channel output mute control • ACODEC_SET_DACR_MUTE: right channel output mute control • ACODEC_GET_GAIN_MICL: Gain of analog left channel MIC input • ACODEC_GET_GAIN_MICR: Gain of analog right channel MIC input • ACODEC_GET_DACL_VOL: Get the volume control of the left channel output • ACODEC_GET_DACR_VOL: Get the volume control of the right channel output • ACODEC_GET_ADCL_VOL: Get the volume control of the left channel input • ACODEC_GET_ADCR_VOL: Get the volume control of the right channel input • ACODEC_SET_PD_DACL: Power down control of left channel output • ACODEC_SET_PD_DACR: Power down control of right channel output • ACODEC_SET_PD_ADCL: Power down control of left channel input • ACODEC_SET_PD_ADCR: Power down control of right channel input • ACODEC_SET_PD_LINEINL: Power down control of left channel LINEIN input • ACODEC_SET_PD_LINEINR: Power down control of right channel LINEIN input 	Input
10.2. Design Overview		324
cmddata	Data pointer corresponding to each cmd	Input

10.3 API Reference

This section describes how to use the API of each module of CVI_AUDIO in order.

Note that when using CVI_ADUDIO, please confirm CVI_SYS_Init has been called to ensure that the corresponding system component is initialized.

CVI_AUDIO_Init must also be called to ensure that the software in the audio module has been initialized.

Users should pay attention to that *CVI_AUDIO_Init* only needs to be called once.

Please confirm the relevant modules of CVI_AUDIO has been disabled

when exiting Audio, and call CVI_SYS_Exit subsequently.

10.3.1 Module Properties API:

10.3.1.1 CVI_AUDIO_INIT

【Description】

Initialize the audio module.

【Syntax】

```
CVI_S32 CVI_AUDIO_INIT(void);
```

【Parameter】

Parameter	Description	Input/Output
None	No need to enter a value.	-

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: cvi_comm_aio.h, cvi_audio.h
- Library files: libcvi_audio.a

【Note】

The initialization through this API is required before using the audio module.

Otherwise, the functionality may be abnormal or the memory may be accessed incorrectly.

10.3.1.2 CVI_AUDIO_DEINIT

【Description】

Initialize the audio module.

【Syntax】

```
CVI_S32 CVI_AUDIO_DEINIT(void);
```

【Parameter】

Parameter	Description	Input/Output
None	No need to enter a value.	-

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: cvi_comm_aio.h, cvi_audio.h
- Library files: libcvi_audio.a

【Note】

Before exiting the audio module, it is necessary to release the module using this API.

Otherwise, the functionality may be abnormal or the memory data may be corrupted.

10.3.1.3 CVI_AUDIO_SetModParam

【Description】

Set the audio module properties.

【Syntax】

```
CVI_S32 CVI_AUDIO_SetModParam(const AUDIO_MOD_PARAM_S *pstModParam);
```

【Parameter】

Parameter	Description	Input/Output
pstModParam	Audio module property pointer.	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: cvi_comm_aio.h, cvi_audio.h
- Library files: libcvi_audio.a

【Note】

Before using each audio sub-module, initialization is required through this API.

【Example】

None.

10.3.1.4 CVI_AUD_SYS_Bind**【Description】**

Set the audio module binding properties.

【Syntax】

```
CVI_S32 CVI_AUD_SYS_Bind(const MMF_CHN_S *pstSrcChn, const MMF_CHN_S *pstDestChn);
```

【parameters】

Parameter	Description	Input/Output
pstSrcChn	Audio bind source pointer.	Input
pstDestChn	Audio bind destination pointer.	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: cvi_comm_aio.h, cvi_audio.h
- Library files: libcvi_audio.a

【Note】

1. The binding objects supported are:

(AudIn -> AudEnc),

(AudDec -> AudOut),

2. See SAMPLE_AUDIO_AiAenc and SAMPLE_AUDIO_AdecAo in cvi_sample_audio.c to understand the binding usage.

【Example】

None.

10.3.1.5 CVI_AUDIO_GetModParam**【Description】**

Get the properties of the audio module.

【Syntax】

```
CVI_S32 CVI_AUDIO_GetModParam(AUDIO_MOD_PARAM_S *pstModParam);
```

【Parameter】

Parameter	Description	Input/Output
pstModParam	Audio module property pointer.	Output

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: cvi_comm_aio.h, cvi_audio.h
- Library files: libcvi_audio.a

【Note】

Before using each audio sub-module, initialization is required through this API.

【Example】

None.

10.3.1.6 CVI_AUDIO_RegisterVQEModule**【Description】**

Register VQE module.

【Syntax】

```
CVI_S32 CVI_AUDIO_RegisterVQEModule(const AUDIO_VQE_REGISTER_S *pstVqeRegister);
```

【Parameter】

Parameter	Description	Input/Output
pstVqeRegister	VQE related property pointer	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: cvi_comm_aio.h, cvi_audio.h
- Library files: libcvi_audio.a, libcvi_vqe.so, libaec.so

【Note】

This api is deprecated.

Please use CVI_AI_SetTalkVqeAttr and CVI_AI_EnableVqe.

【Example】

None.

10.3.1.7 CVI_AENC_RegisterExternalEncoder

【Description】

Register audio encoding module.

【Syntax】

```
CVI_S32 CVI_AENC_RegisterExternalEncoder(CVI_S32 *ps32Handle, const AAC_AENC_ENCODER_
↪ S *pstEncoder);
```

【Parameter】

Parameter	Description	Input/Output
ps32Handle	parameter property pointer	Input
pstEncoder	Encoding module function pointer	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: cvi_comm_aio.h, cvi_audio.h, cvi_audio_aac_adp.h
- Library files: libcvi_audio.a, libaacenc2.so

【Note】

- This function is used for AAC audio encoding.
- Before registering the function, make sure that the function pointer in pstEncoder is not null, otherwise there will be errors when calling it internally.
- Related examples can refer to sample/audio/aac_sample/cvi_audio_aac_adp.c
- This function only supports AAC code registration, please use CVI_AENC_CreateChn for related G7xx series coding. See SAMPLE_AUDIO_AiAenc in cvi_sample_audio.c for reference.

【Example】

None.

10.3.1.8 CVI_AENC_UnRegisterExternalEncoder

【Description】

Cancel the encoder.

【Syntax】

```
CVI_S32 CVI_AENC_UnRegisterExternalEncoder(CVI_S32 s32Handle);
```

【Parameter】

Parameter	Description	Input/Output
s32Handle	Register handle (the handle obtained when registering the encoder).	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: cvi_comm_aio.h, cvi_audio.h, cvi_audio_aac_adp.h
- Library files: libcvi_audio.a, libaacenc2.so

【Note】

- This function is used for AAC audio coding.
 - Before unregistering the encoder, all encoding channels created through this encoder must be destroyed. Calling this interface without destroying all channels will result in an error.

【Example】

None.

10.3.1.9 CVI_ADEC_RegisterExternalDecoder**【Description】**

Register audio decoding module.

【Syntax】

```
CVI_S32 CVI_ADEC_RegisterExternalDecoder(CVI_S32 *ps32Handle, const ADEC_DECODER_S_
↪*pstDecoder);
```

【Parameter】

Parameter	Description	Input/Output
ps32Handle	parameter property pointer	Input/Output
pstDecoder	Encoding module function pointer	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: cvi_comm_aio.h, cvi_audio.h , cvi_audio_aac_adp.h
- Library files: libcvi_audio.a, libaadec2.so

【Note】

- This function is used for AAC audio decoding.
- Before registering the function, make sure that the function pointer in pstEncoder is not null, otherwise there will be errors when calling it internally.
- Related examples can refer to sample/audio/aac_sample/cvi_audio_aac_adp.c
- This function only supports AAC code registration, please use CVI_AENC_CreateChn for related G7xx series coding. See SAMPLE_AUDIO_AiAenc in cvi_sample_audio.c for reference.

【Example】

None.

10.3.1.10 CVI_ADEC_UnRegisterExternalDecoder

【Description】

Cancel audio decoding module.

【Syntax】

```
CVI_S32 CVI_ADEC_UnRegisterExternalDecoder(CVI_S32 s32Handle);
```

【Parameter】

Parameter	Description	Input/Output
s32Handle	Register handle	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: cvi_comm_aio.h, cvi_audio.h, cvi_audio_aac_adp.h
- Library files: libcvi_audio.a, libaadec2.so

【Note】

- This function is used for AAC audio decoding.

【Example】

None.

10.3.2 Audio Input

The audio input AI realizes the functions of configuring and enabling the audio input device and acquiring the audio frame data.

The function module provides the following APIs.

- *CVI_AI_SetPubAttr*: Set AI device properties.
- *CVI_AI_GetPubAttr*: Get AI device properties.
- *CVI_AI_Enable*: Enable AI device properties.
- *CVI_AI_Disable*: Disable AI device properties.
- *CVI_AI_EnableChn*: Enable AI channel.
- *CVI_AI_DisableChn*: Disable AI channel.
- *CVI_AI_GetFrame*: Get the audio frame.
- *CVI_AI_ReleaseFrame*: Release the audio frame.
- *CVI_AI_SetChnParam*: Set AI channel parameters.
- *CVI_AI_GetChnParam*: Get AI channel parameters.
- *CVI_AI_EnableReSmp*: Enable AI resampling.
- *CVI_AI_DisableReSmp*: Disable AI resampling.
- *CVI_AI_ClrPubAttr*: Clear AI device properties.

- *CVI_AI_SaveFile*: Enable audio input file recording functionality.
- *CVI_AI_QueryFileStatus*: Query the status of the audio input channel for saving to file.
- *CVI_AI_EnableAecRefFrame*: When the AEC is not turned on, the user can also get the AEC reference frame.
- *CVI_AI_DisableAecRefFrame*: When the AEC is not turned on, it is forbidden to obtain the AEC reference frame.
- *CVI_AI_SetVolume*: Set the AI device volume.
- *CVI_AI_GetVolume*: Get the AI device volume.

10.3.2.1 CVI_AI_SetPubAttr

【Description】

Set AI device properties.

【Syntax】

```
CVI_S32 CVI_AI_SetPubAttr(AUDIO_DEV AiDevId, const AIO_ATTR_S *pstAttr);
```

【Parameter】

Parameter	Description	Input/Output
AiDevId	Audio device number	Input
pstAttr	AI device attribute pointer	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: `cvl_comm_aio.h`, `cvl_audio.h`
- Library files: `libcvl_audio.a`

【Note】

The properties of audio input device determine the format of input data.

The properties of input device include working mode, sampling rate, sampling precision, buffer size, sampling points per frame, extended flag, clock selection and channel number.

These properties should be consistent with the timing of docking codec configuration, that is, they can be successfully docked.

AiDevId is set to 0 by default. If amplification is not required, an error will be returned if it exceeds 2.

enBitwidth :

the bit depth is 8bit, 16bit or 24bit.

In practical application, the sampling accuracy is limited by Audio Codec.

The u32frmnum item in Buffer size AIO_ATTR_S is used to configure the audio of the block used to receive audio data in AI.

UsrFrmDept :

block voice box, it is recommended to be greater than or equal to 5.

enSamplerate: When the audio sampling rate is high, it is recommended to increase the number of sampling points per frame accordingly.

If you want to encode the collected audio data, you should ensure that the duration of each frame is not less than 10ms (For example, the number of sampling points of each frame should be set to at least 160 at the sampling frequency of 16K).

Otherwise there may be abnormal sound after decoding.

enSoundmode :

two channel or single channel voice frame setting.

u32ChnCnt :

channel number.

Channel number refers to the number of AI channels of the current input device, which should be consistent with the configuration of the audio codec to be connected.

It supports 1 channel and 2 channels.

u32EXFlag :

the extended flag is invalid for AI devices.

u32ClkSel :

clock setting.

In this chip CV1835, there is no special setting, just note that AIO attribute should be set consistently in AI / AO.

```
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;
```

【Example】

None.

10.3.2.2 CVI_AI_GetPubAttr

【Description】

Get AI device properties.

【Syntax】

```
CVI_S32 CVI_AI_GetPubAttr(AUDIO_DEV AiDevId, AIO_ATTR_S*pstAttr);
```

【Parameter】

Parameter	Description	Input/Output
AiDevId	Audio device number	Input
pstAttr	AI device attribute pointer	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: cvi_comm_aio.h, cvi_audio.h
- Library files: libcvi_audio.a

【Note】

If not initialized or CVI_AI_SetPubAttr has not been called, the user will get the pointer and value with the content of 0.

AiDevId is set to 0 by default.

If amplification is not required, an error will be returned if it exceeds 2.

【Example】

None.

10.3.2.3 CVI_AI_Enable

【Description】

Enable AI device.

【Syntax】

```
CVI_S32 CVI_AI_Enable(AUDIO_DEV AiDevId);
```

【Parameter】

Parameter	Description	Input/Output
AiDevId	Audio device number. The device number is pre-set to 0, and setting it to a value greater than 2 will return an error, unless customization requires expansion.	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: cvi_comm_aio.h, cvi_audio.h
- Library files: libcvi_audio.a

【Note】

This API is the last function to enable AI.

Please confirm that the relevant properties have been set before calling.

【Example】

```

Audio In Examples:
CVI_S32 i;
CVI_S32 s32Ret;

s32Ret = CVI_AI_SetPubAttr(AiDevId, pstAioAttr);
if (s32Ret) {
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;
}
}
}

```

(continues on next page)

(continued from previous page)

```

        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) {
                printf("%s: CVI_AI_EnableVqe(%d,%d) failed with %#x\n", __func__,
                    AiDevId, i, s32Ret);
            }
            return s32Ret;
        }
    }
}

```

10.3.2.4 CVI_AI_Disable

【Description】

Disable AI device.

【Syntax】

```
CVI_S32 CVI_AI_Disable(AUDIO_DEV AiDevId);
```

【Parameter】

Parameter	Description	Input/Output
AiDevId	Audio device number. The device number is pre-set to 0, and setting it to a value greater than 2 will return an error, unless customization requires expansion.	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: `cvi_comm_aio.h`, `cvi_audio.h`
- Library files: `libcvi_audio.a`

【Note】

This API is the last program to stop an AI device.

Before you disable an AI device, you must disable all enabled AI channels under the device.

If there is an AENC or AO connection, please stop the association with it before calling this API.

【Example】

None.

10.3.2.5 CVI_AI_EnableChn**【Description】**

Enable AI channel.

【Syntax】

```
CVI_S32 CVI_AI_EnableChn(AUDIO_DEV AiDevId, AI_CHN AiChn);
```

【Parameter】

Parameter	Description	Input/Output
AiDevId	Audio device number. The device number is preset to 0, and setting it to a value greater than 2 will return an error, unless customization requires expansion.	Input
AiChn	Audio input channel number. It is independent of the sound mode <code>enSoundmode</code> . Users can obtain the same audio source through different <code>AiChn</code> under the same <code>AiDevId</code> .	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: `cvi_comm_aio.h`, `cvi_audio.h`
- Library files: `libcvi_audio.a`

【Note】

Before enabling an AI channel, you must first enable its AI device.

【Example】

None.

10.3.2.6 CVI_AI_DisableChn

【Description】

Disable AI channel.

【Syntax】

```
CVI_S32 CVI_AI_DisableChn(AUDIO_DEV AiDevId, AI_CHN AiChn);
```

【Parameter】

Parameter	Description	Input/Output
AiDevId	Audio device number. The device number is preset to 0, and setting it to a value greater than 2 will return an error, unless customization requires expansion.	Input
AiChn	Audio input channel number. It is independent of the sound mode enSoundmode. Users can obtain the same audio source through different AiChn under the same AiDevId.	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: cvi_comm_aio.h, cvi_audio.h
- Library files: libcvi_audio.a

【Note】

Please call this API before CVI_AI_Disable to avoid channel parameter residue.

【Example】

None.

10.3.2.7 CVI_AI_GetFrame

【Description】

Get the audio frame.

【Syntax】

```
CVI_S32 CVI_AI_GetFrame(AUDIO_DEV AiDevId, AI_CHN AiChn, AUDIO_FRAME_S *pstFrm, AEC_
↪FRAME_S *pstAecFrm, CVI_S32 s32MilliSec);
```

【Parameter】

Parameter	Description	Input/Output
AiDevId	Audio device number. The device number is preset to 0, and setting it to a value greater than 2 will return an error, unless customization requires expansion.	Input
AiChn	Audio input channel number. It is independent of the sound mode enSoundmode. Users can obtain the same audio source through different AiChn under the same AiDevId.	Input
pstFrm	Audio frame structure pointer. The user obtains the voice frame from the structure pointer.	Output
pstAecFrm	Echo cancellation reference frame structure pointer.	Output
s32MilliSec	The timeout of data acquisition - 1 indicates blocking mode, waiting all the time when there is no data; 0 indicates non-blocking mode, reporting an error and returning when there is no data; > 0 indicates blocking s32MilliSec MS, reporting an error and returning when there is a timeout	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: cvi_comm_aio.h, cvi_audio.h
- Library files: libcvi_audio.a

【Note】

- The value of s32MilliSec must be greater than or equal to -1. The data shall be obtained in blocking mode when it is equal to -1; non-blocking mode shall be used when it is equal to 0; when greater than 0, it will block s32MilliSec for millisecond and will return timeout and report an error if there is no data.
- Before acquiring audio frame data, the corresponding AI channel must be enabled.
- To get the AEC frame, please make sure that the AEC in VQE is turned on.

【Example】

None.

10.3.2.8 CVI_AI_ReleaseFrame**【Description】**

Release the audio frame.

【Syntax】

```
CVI_S32 CVI_AI_ReleaseFrame(AUDIO_DEV AiDevId, AI_CHN AiChn, const AUDIO_FRAME_S
↪ *pstFrm, const AEC_FRAME_S *pstAecFrm);
```

【Parameter】

Parameter	Description	Input/Output
AiDevId	Audio device number. The device number is pre-set to 0, and setting it to a value greater than 2 will return an error, unless customization requires expansion.	Input
AiChn	Audio input channel number. It is independent of the sound mode enSoundmode. Users can obtain the same audio source through different AiChn under the same AiDevId.	Input
pstFrm	Audio frame structure pointer. The user obtains the voice frame from the structure pointer.	Input
pstAecFrm	Echo cancellation reference frame structure pointer.	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: cvi_comm_aio.h, cvi_audio.h
- Library files: libcvi_audio.a

【Note】

If there is no need to release the echo cancellation reference frame, set pstAecFrm to null.

【Example】

None.

10.3.2.9 CVI_AI_SetChnParam**【Description】**

Set AI channel parameters.

【Syntax】

```
CVI_S32 CVI_AI_SetChnParam(AUDIO_DEV AiDevId, AI_CHN AiChn, const AI_CHN_PARAM_S
↪*pstChnParam);
```

【Parameter】

Parameter	Description	Input/Output
AiDevId	Audio device number. The device number is pre-set to 0, and setting it to a value greater than 2 will return an error, unless customization requires expansion.	Input
AiChn	Audio input channel number. It is independent of the sound mode enSoundmode. Users can obtain the same audio source through different AiChn under the same AiDevId.	Input
pstChnParam	Audio channel parameters	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: cvi_comm_aio.h, cvi_audio.h
- Library files: libcvi_audio.a

【Note】

At present, the channel parameter has only one member variable, which is used to set the block depth for the user to obtain the audio frame.

The default depth is 0.

The value of the member variable cannot be greater than 30.

【Example】

None.

10.3.2.10 CVI_AI_GetChnParam**【Description】**

Get AI channel parameters.

【Syntax】

```
CVI_S32 CVI_AI_GetChnParam(AUDIO_DEV AiDevId, AI_CHN AiChn, AI_CHN_PARAM_S
↪*pstChnParam);
```

【Parameter】

Parameter	Description	Input/Output
AiDevId	Audio device number. The device number is pre-set to 0, and setting it to a value greater than 2 will return an error, unless customization requires expansion.	Input
AiChn	Audio input channel number. It is independent of the sound mode enSoundmode. Users can obtain the same audio source through different AiChn under the same AiDevId.	Input
pstChnParam	Audio channel parameters	Output

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: cvi_comm_aio.h, cvi_audio.h

The audio module mentioned in this file is one of the interfaces of Middleware multimedia layer.

- Library files: libcvi_audio.a

【Note】

At present, the channel parameter has only one member variable, which is used to set the block depth of the audio frame.

【Example】

None.

10.3.2.11 CVI_AI_EnableReSmp**【Description】**

Enable AI resampling.

【Syntax】

```
CVI_S32 CVI_AI_EnableReSmp(AUDIO_DEV AiDevId, AI_CHN AiChn, AUDIO_SAMPLE_RATE_E_
↪enOutSampleRate);
```

【Parameter】

Parameter	Description	Input/Output
AiDevId	Audio device number. The device number is pre-set to 0, and setting it to a value greater than 2 will return an error, unless customization requires expansion.	Input
AiChn	Audio input channel number. It is independent of the sound mode enSoundmode. Users can obtain the same audio source through different AiChn under the same AiDevId.	Input
enOutSampleRat	The output sampling rate of audio resampling.	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: cvi_comm_aio.h, cvi_audio.h
- Library files: libcvi_audio.a, libcvi_vqe.so, libcvi_RES1.so

【Note】

This API is not supported when CVI_AUD_SYS_Bind is used.

The sample rate supported is as follows.

```
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*/
}
```

(continues on next page)

(continued from previous page)

```

    AUDIO_SAMPLE_RATE_BUTT,
} AUDIO_SAMPLE_RATE_E;

```

【Example】

None.

10.3.2.12 CVI_AI_DisableReSmp**【Description】**

Disable AI resampling.

【Syntax】

```
CVI_S32 CVI_AI_DisableReSmp(AUDIO_DEV AiDevId, AI_CHN AiChn);
```

【Parameter】

Parameter	Description	Input/Output
AiDevId	Audio device number. The device number is pre-set to 0, and setting it to a value greater than 2 will return an error, unless customization requires expansion.	Input
AiChn	Audio input channel number. It is independent of the sound mode enSoundmode. Users can obtain the same audio source through different AiChn under the same AiDevId.	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: cvi_comm_aio.h, cvi_audio.h
- Library files: libcvi_audio.a, libcvi_RES1.so, libcvi_vqe.so

【Note】

Please disable the AENC channel and AO channel that use the corresponding AI device audio data before calling this interface, otherwise this interface may fail.

【Example】

None.

10.3.2.13 CVI_AI_ClrPubAttr

【Description】

Clear the Pub property.

【Syntax】

```
CVI_S32 CVI_AI_ClrPubAttr(AUDIO_DEV AiDevId);
```

【Parameter】

Parameter	Description	Input/Output
AiDevId	Audio device number. The device number is preset to 0, and setting it to a value greater than 2 will return an error, unless customization requires expansion.	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: cvi_comm_aio.h, cvi_audio.h
- Library files: libcvi_audio.a

【Note】

You need to stop the device before clearing its properties.

Before clearing the device properties, it is recommended to stop all internal connected or user-get mode actions to avoid abnormal behavior caused by the bottom layer still moving audio frames.

【Example】

None.

10.3.2.14 CVI_AI_SaveFile

【Description】

Enable audio input file recording functionality

【Syntax】

```
CVI_S32 CVI_AI_SaveFile(AUDIO_DEV AiDevId, AI_CHN AiChn, const AUDIO_SAVE_FILE_INFO_S_
↪*pstSaveFileInfo);
```

【Parameter】

Parameter	Description	Input/Output
AiDevId	Audio device number. The device number is preset to 0, and setting it to a value greater than 2 will return an error, unless customization requires expansion.	Input
AiChn	Audio input channel number. It is independent of the sound mode enSoundmode. Users can obtain the same audio source through different AiChn under the same AiDevId.	Input
pstSaveFileInfo	Pointer to audio recording file property structure	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: cvi_comm_aio.h, cvi_audio.h
- Library files: libcvi_audio.a

【Note】

This interface is only used for dumping AI in AI-AENC and AI-AO non-system binding mode.

【Example】

None.

10.3.2.15 CVI_AI_QueryFileStatus**【Description】**

Query the status of the audio input channel for saving to file.

【Syntax】

```
CVI_S32 CVI_AI_QueryFileStatus(AUDIO_DEV AiDevId, AI_CHN AiChn, AUDIO_FILE_STATUS_SU
↪*pstFileStatus);
```

【Parameter】

Parameter	Description	Input/Output
AiDevId	Audio device number. The device number is preset to 0, and setting it to a value greater than 2 will return an error, unless customization requires expansion.	Input
AiChn	Audio input channel number. It is independent of the sound mode enSoundmode. Users can obtain the same audio source through different AiChn under the same AiDevId.	Input
pstFileStatus	Pointer to file status property structure	Output

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: cvi_comm_aio.h, cvi_audio.h
- Library files: libcvi_audio.a

【Note】

- This interface is mainly used for debugging, which is not used in general processes.
- This interface is only used for dumping AI in AI-AENC and AI-AO non-system binding mode.
- This interface is used to query whether the audio input channel is in the state of saving files after users call CVI_AI_SaveFile to save files.

If bSaving of pstFileStatus is CVI_TRUE, it indicates that specified size has not been reached; CVI_FALSE indicates that it has reached the specified size.

【Example】

None.

10.3.2.16 CVI_AI_EnableAecRefFrame**【Description】**

When the AEC is not turned on, the user can also get the AEC reference frame.

【Syntax】

```
CVI_S32 CVI_AI_EnableAecRefFrame(AUDIO_DEV AiDevId, AI_CHN AiChn, AUDIO_DEV AoDevId,
↪AO_CHN AoChn);
```

【Parameter】

Parameter	Description	Input/Output
AiDevId	Audio device number. The device number is preset to 0, and setting it to a value greater than 2 will return an error, unless customization requires expansion.	Input
AiChn	Audio input channel number. It is independent of the sound mode enSoundmode. Users can obtain the same audio source through different AiChn under the same AiDevId.	Input
AoDevId	AO device number used to obtain AEC reference frame.	Input
AoChn	AO channel number used to obtain AEC reference frame.	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: cvi_comm_aio.h, cvi_audio.h
- Library files: libcvi_audio.a

【Note】

None.

【Example】

None.

10.3.2.17 CVI_AI_DisableAecRefFrame**【Description】**

When the AEC is not turned on, it is forbidden to obtain the AEC reference frame.

【Syntax】

```
CVI_S32 CVI_AI_DisableAecRefFrame(AUDIO_DEV AiDevId, AI_CHN AiChn);
```

【Parameter】

Parameter	Description	Input/Output
AiDevId	Audio device number. The device number is preset to 0, and setting it to a value greater than 2 will return an error, unless customization requires expansion.	Input
AiChn	Audio input channel number. It is independent of the sound mode enSoundmode. Users can obtain the same audio source through different AiChn under the same AiDevId.	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: cvi_comm_aio.h, cvi_audio.h
- Library files: libcvi_audio.a

【Note】

None.

【Example】

None.

10.3.2.18 CVI_AI_SetVolume

【Description】

Set the input volume.

【Syntax】

```
CVI_S32 CVI_AI_SetVolume(AUDIO_DEV AiDevId, CVI_S32 s32VolumeStep);
```

【Parameter】

Parameter	Description	Input/Output
AiDevId	Audio device number. The device number is pre-set to 0, and setting it to a value greater than 2 will return an error, unless customization requires expansion.	Input
s32VolumeStep	Step size for input volume amplification [24-0, 0:mute].	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: cvi_comm_aio.h, cvi_audio.h
- Library files: libcvi_audio.a/libcvi_audio.so

【Note】

None.

【Example】

None.

10.3.2.19 CVI_AI_GetVolume

【Description】

Get the input volume.

【Syntax】

```
CVI_S32 CVI_AI_GetVolume(AUDIO_DEV AiDevId, CVI_S32 *ps32VolumeStep);
```

【Parameter】

Parameter	Description	Input/Output
AiDevId	Audio device number. The device number is pre-set to 0, and setting it to a value greater than 2 will return an error, unless customization requires expansion.	Input
ps32VolumeStep	Pointer to step size for input volume amplification [24-0, 0:mute].	Output

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: `cvi_comm_aio.h`, `cvi_audio.h`
- Library files: `libcvi_audio.a/libcvi_audio.so`

【Note】

None.

【Example】

None.

10.3.3 Voice Quality Enhancement API

- *CVI_AI_SetVqeAttr*: Set AI' s voice quality enhancement related attributes.
- *CVI_AI_SetTalkVqeAttr*: Set voice-related properties of AI' s speech quality enhancement function.
- *CVI_AI_GetTalkVqeAttr*: Get voice-related properties of AI' s speech quality enhancement function.
- *CVI_AI_SetRecordVqeAttr*: Set voice-related properties of AI' s recording quality enhancement function (currently not supported).
- *CVI_AI_GetRecordVqeAttr*: Get voice-related properties of AI' s recording quality enhancement function (currently not supported).
- *CVI_AI_EnableVqe*: Enable AI' s voice quality enhancement.
- *CVI_AI_DisableVqe*: Disable AI' s voice quality enhancement.
- *CVI_AI_SetTrackMode*: Set channel mode.
- *CVI_AI_GetTrackMode*: Get channel mode.
- *CVI_AO_SetVqeAttr*: Set the voice quality enhancement attributes of AO.
- *CVI_AO_GetVqeAttr*: Get the voice quality enhancement attributes of AO.
- *CVI_AO_EnableVqe*: Enable the voice quality enhancement of AO.
- *CVI_AO_DisableVqe*: Disable the voice quality enhancement of AO
- *CVI_VQE_PathSelect*: Setting VQE algorithm path.

10.3.3.1 CVI_AI_SetVqeAttr

【Description】

Set AI' s voice quality enhancement related attributes.

【Syntax】

```
CVI_S32 CVI_AI_SetVqeAttr(AUDIO_DEV AiDevId, AI_CHN AiChn, AUDIO_DEV AoDevId, AO_CHN_AoChn, AI_VQE_CONFIG_S *pstVqeConfig);
```

【Parameter】

Parameter	Description	Input/Output
AiDevId	Audio device number. The device number is pre-set to 0, and setting it to a value greater than 2 will return an error, unless customization requires expansion.	Input
AiChn	Audio input channel number. It is independent of the sound mode enSoundmode. Users can obtain the same audio source through different AiChn under the same AiDevId.	Input
AoDevId	AO device number for echo cancellation.	Input
AoChn	AO channel number for echo cancellation.	Input
pstVqeConfig	Pointer to Audio Input sound quality enhancement configuration structure.	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: cvi_comm_aio.h, cvi_audio.h
- Library files: libcvi_audio.a, libcvi_RES1.so, libcvi_vqe.so,

【Note】

Before enabling the sound quality enhancement function, you must first set the relevant properties of the sound quality enhancement function of the corresponding AI channel.

Before setting the relevant attributes of AI sound quality enhancement function, the corresponding AI channel must be enabled.

The sound quality enhancement function of the same AI channel does not support dynamic setting of attributes.

To reset the relevant attributes of the sound quality enhancement function of the AI channel, you need to turn off the sound quality function of the AI channel first, and then set the relevant attributes of the sound quality enhancement function of the AI channel.

【Example】

None.

10.3.3.2 CVI_AI_SetTalkVqeAttr**【Description】**

Set voice-related properties of AI' s speech quality enhancement function.

【Syntax】

```
CVI_S32 CVI_AI_SetTalkVqeAttr(AUDIO_DEV AiDevId, AI_CHN AiChn, AUDIO_DEV AoDevId, AO_
↪ CHN AoChn, AI_TALKVQE_CONFIG_S *pstVqeConfig);
```

【Parameter】

Parameter	Description	Input/Output
AiDevId	Audio device number. The device number is preset to 0, and setting it to a value greater than 2 will return an error, unless customization requires expansion.	Input
AiChn	Audio input channel number. It is independent of the sound mode enSoundmode. Users can obtain the same audio source through different AiChn under the same AiDevId.	Input
AoDevId	AO device number for echo cancellation.	Input
AoChn	AO channel number for echo cancellation.	Input
pstVqeConfig	Audio input sound quality enhancement configuration structure pointer.	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: cvi_comm_aio.h, cvi_audio.h
- Library files: libcvi_audio.a, libcvi_RES1.so, libcvi_vqe.so, libaec.so

【Note】

Talk VQE is mainly used in IPC scenarios.

The sound quality enhancement function of the same AI channel does not support dynamic setting of attributes.

To reset the relevant attributes of the sound quality enhancement function of the AI channel, you need to turn off the sound quality function of the AI channel first, and then set the relevant attributes of the sound quality enhancement function of the AI channel.

【Example】

None.

10.3.3.3 CVI_AI_GetTalkVqeAttr**【Description】**

Get voice-related properties of AI' s speech quality enhancement function.

【Syntax】

```
CVI_S32 CVI_AI_GetTalkVqeAttr(AUDIO_DEV AiDevId, AI_CHN AiChn, AI_TALKVQE_CONFIG_S
↪*pstVqeConfig);
```

【Parameter】

Parameter	Description	Input/Output
AiDevId	Audio device number. The device number is preset to 0, and setting it to a value greater than 2 will return an error, unless customization requires expansion.	Input
AiChn	Audio input channel number. It is independent of the sound mode enSoundmode. Users can obtain the same audio source through different AiChn under the same AiDevId.	Input
pstVqeConfig	Audio input sound quality enhancement configuration structure pointer.	Output

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: cvi_comm_aio.h, cvi_audio.h
- Library files: libcvi_audio.a, libcvi_RES1.so, libcvi_vqe.so, libaec.so

【Note】

Before acquiring the sound quality enhancement related attributes, it is necessary to set the sound quality enhancement related attributes of the corresponding AI channel.

【Example】

None.

10.3.3.4 CVI_AI_SetRecordVqeAttr**【Description】**

Set voice-related properties of AI' s recording quality enhancement function

【Syntax】

```
CVI_S32 CVI_AI_SetRecordVqeAttr(AUDIO_DEV AiDevId, AI_CHN AiChn, const AI_RECORDVQE_
↪CONFIG_S *pstVqeConfig);
```

【Parameter】

Parameter	Description	Input/Output
AiDevId	Audio device number. The device number is preset to 0, and setting it to a value greater than 2 will return an error, unless customization requires expansion.	Input
AiChn	Audio input channel number. It is independent of the sound mode enSoundmode. Users can obtain the same audio source through different AiChn under the same AiDevId.	Input
pstVqeConfig	Audio input sound quality enhancement configuration structure pointer.	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: `cvi_comm_aio.h`, `cvi_audio.h`
- Library files: `libcvi_audio.a`, `libcvi_RES1.so`, `libcvi_vqe.so`, `libaec.so`

【Note】

At present, the voice algorithm of CVI182X is mainly set by calling `CVI_AI_SetTalkVqeAttr`.

That Users use this `RecordVqeAttr` (`CVI_AI_SetRecordVqeAttr`/ `GetRecordVqeAttr`) may not support the algorithm or cannot be set, so it is not recommended.

【Example】

None.

10.3.3.5 CVI_AI_GetRecordVqeAttr**【Description】**

Get voice-related properties of AI' s recording quality enhancement function

【Syntax】

```
CVI_AI_GetRecordVqeAttr (AUDIO_DEV AiDevId, AI_CHN AiChn, const AI_RECORDVQE_CONFIG_S_
↪*pstVqeConfig);
```

【Parameter】

Parameter	Description	Input/Output
AiDevId	Audio device number. The device number is pre-set to 0, and setting it to a value greater than 2 will return an error, unless customization requires expansion.	Input
AiChn	Audio input channel number. It is independent of the sound mode <code>enSoundmode</code> . Users can obtain the same audio source through different <code>AiChn</code> under the same <code>AiDevId</code> .	Input
pstVqeConfig	Audio input sound quality enhancement configuration structure pointer.	Output

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: `cvi_comm_aio.h`, `cvi_audio.h`
- Library files: `libcvi_audio.a`, `libcvi_RES1.so`, `libcvi_vqe.so`, `libaec.so`

【Note】

At present, the voice algorithm of CVI182x is mainly set by calling CVI_AI_SetTalkVqeAttr.

That Users use this RecordVqeAttr (CVI_AI_SetRecordVqeAttr/ GetRecordVqeAttr) may not support the algorithm or cannot be set, so it is not recommended.

【Example】

None.

10.3.3.6 CVI_AI_EnableVqe

【Description】

Enable AI' s voice quality enhancement.

【Syntax】

```
CVI_S32 CVI_AI_EnableVqe(AUDIO_DEV AiDevId, AI_CHN AiChn);
```

【Parameter】

Parameter	Description	Input/Output
AiDevId	Audio device number. The device number is pre-set to 0, and setting it to a value greater than 2 will return an error, unless customization requires expansion.	Input
AiChn	Audio input channel number. It is independent of the sound mode enSoundmode. Users can obtain the same audio source through different AiChn under the same AiDevId.	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: cvi_comm_aio.h, cvi_audio.h
- Library files: libcvi_audio.a, libcvi_RES1.so, libcvi_vqe.so, libaec.so

【Note】

- Before enabling the sound quality enhancement function, the corresponding AI channel must be enabled.
- When the sound quality enhancement function of the same AI channel is enabled several times, Success is returned.
- After AI channel is disabled, if AI channel is re enabled and sound quality enhancement is used, this interface should be called to re enable sound quality enhancement.

【Example】

None.

10.3.3.7 CVI_AI_DisableVqe

【Description】

Disable AI' s voice quality enhancement.

【Syntax】

```
CVI_S32 CVI_AI_DisableVqe(AUDIO_DEV AiDevId, AI_CHN AiChn);
```

【Parameter】

Parameter	Description	Input/Output
AiDevId	Audio device number. The device number is preset to 0, and setting it to a value greater than 2 will return an error, unless customization requires expansion.	Input
AiChn	Audio input channel number. It is independent of the sound mode enSoundmode. Users can obtain the same audio source through different AiChn under the same AiDevId.	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: cvi_comm_aio.h, cvi_audio.h
- Library files: libcvi_audio.a, libcvi_RES1.so, libcvi_vqe.so, libaec.so

【Note】

- When AI sound quality enhancement is no longer used, this interface should be called to disable it.

【Example】

None.

10.3.3.8 CVI_AI_SetTrackMode

【Description】

Set AI channel mode.

【Syntax】

```
CVI_S32 CVI_AI_SetTrackMode(AUDIO_DEV AiDevId, AUDIO_TRACK_MODE_E enTrackMode);
```

【Parameter】

Parameter	Description	Input/Output
AiDevId	Audio device number. The device number is preset to 0, and setting it to a value greater than 2 will return an error, unless customization requires expansion.	Input
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, AU- DIO_TRACK_LEFT_MUTE = 5, AU- DIO_TRACK_RIGHT_MUTE = 6, AUDIO_TRACK_BOTH_MUTE = 7, AUDIO_TRACK_BUTT } AU- DIO_TRACK_MODE_E;	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: cvi_comm_aio.h, cvi_audio.h
- Library files: libcvi_audio.a, libcvi_RES1.so, libcvi_vqe.so, libaec.so

【Note】

- It supports acquiring channel mode when AI device works in I2S mode, but not in PCM mode.
- Call this interface after AI device is enabled successfully.
- TrackMode capability is related to audio codec.

If the client uses its own codec, the settings may be different.

【Example】

None.

10.3.3.9 CVI_AI_GetTrackMode**【Description】**

Get AI channel mode.

【Syntax】

```
CVI_S32 CVI_AI_GetTrackMode(AUDIO_DEV AiDevId, AUDIO_TRACK_MODE_E *penTrackMode);
```

【Parameter】

Parameter	Description	Input/Output
AiDevId	Audio device number. The device number is pre-set to 0, and setting it to a value greater than 2 will return an error, unless customization requires expansion.	Input
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, AU- DIO_TRACK_LEFT_MUTE = 5, AU- DIO_TRACK_RIGHT_MUTE = 6, AUDIO_TRACK_BOTH_MUTE = 7, AUDIO_TRACK_BUTT } AU- DIO_TRACK_MODE_E;	Output

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: cvi_comm_aio.h, cvi_audio.h
- Library files: libcvi_audio.a, libcvi_RES1.so, libcvi_vqe.so, libaec.so

【Note】

- It supports acquiring channel mode when AI device works in I2S mode, but not in PCM mode.
- Call this interface after AI device is enabled successfully.
- TrackMode capability is related to audio codec.

If the client uses its own codec, the settings may be different.

【Example】

None.

10.3.3.10 CVI_AO_SetVqeAttr**【Description】**

Set the voice quality enhancement attributes of AO

【Syntax】

```
CVI_S32 CVI_AO_SetVqeAttr(AUDIO_DEV AoDevId, AO_CHN AoChn, AO_VQE_CONFIG_Su
↪ *pstVqeConfig);
```

【Parameter】

Parameter	Description	Input/Output
AoDevId	Audio device number. The device number is preset to 0, and setting it to a value greater than 2 will return an error, unless customization requires expansion.	Input
AoChn	Audio output channel number	Input
pstVqeConfig	Audio output sound quality enhancement configuration structure pointer	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: cvi_comm_aio.h, cvi_audio.h
- Library files: libcvi_audio.a、libaec.so、libcvi_RES1.so

【Note】

- Before enabling the sound quality enhancement, you must first set the sound quality enhancement related properties of the corresponding AP channel.
- The corresponding AO channel must be enabled before setting the related attributes of the sound quality enhancement function of AO.

【Example】

None.

10.3.3.11 CVI_AO_GetVqeAttr**【Description】**

Get the voice quality enhancement attributes of AO

【Syntax】

```
CVI_S32 CVI_AO_GetVqeAttr(AUDIO_DEV AoDevId, AO_CHN AoChn, AO_VQE_CONFIG_S
↪ *pstVqeConfig);
```

【Parameter】

Parameter	Description	Input/Output
AoDevId	Audio device number. The device number is preset to 0, and setting it to a value greater than 2 will return an error, unless customization requires expansion.	Input
AoChn	Audio output channel number	Input
pstVqeConfig	Audio output sound quality enhancement configuration structure pointer	Output

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: `cvi_comm_aio.h`, `cvi_audio.h`
- Library files: `libcvi_audio.a`, `libaec.so`, `libcvi_RES1.so`, `libcvi_vqe.so`

【Note】

- Before obtaining the sound quality enhancement related attributes, you must first set the sound quality enhancement related attributes of the corresponding AO channel.

【Example】

None.

10.3.3.12 CVI_AO_EnableVqe**【Description】**

Enable the voice quality enhancement of AO

【Syntax】

```
CVI_S32 CVI_AO_EnableVqe(AUDIO_DEV AoDevId, AO_CHN AoChn);
```

【Parameter】

Parameter	Description	Input/Output
AoDevId	Audio device number. The device number is preset to 0, and setting it to a value greater than 2 will return an error, unless customization requires expansion.	Input
AoChn	Audio output channel number	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: `cvi_comm_aio.h`, `cvi_audio.h`
- Library files: `libcvi_audio.a`, `libaec.so`, `libcvi_RES1.so`, `libcvi_vqe.so`

【Note】

- The corresponding AO channel must be enabled before enabling sound quality enhancement.
- After the AO channel is disabled, if you re enable the AO channel again and use the sound quality enhancement function, you need to call this interface to re enable the sound quality enhancement function.

【Example】

None.

10.3.3.13 CVI_VQE_PathSelect

【Description】

Setting VQE algorithm path.

【Syntax】

```
CVI_S32 CVI_VQE_PathSelect(E_VQE_ALGO_PATH eVqePath);
```

【Parameter】

Parameter	Description	Input/Output
AoDevId	Audio device number. The device number is pre-set to 0, and setting it to a value greater than 2 will return an error, unless customization requires expansion.	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: `cvl_comm_aio.h`, `cvl_audio.h`
- Library files: `libcvl_audio.a`, `libaec.so`, `libcvl_RES1.so`, `libcvl_vqe.so`

【Note】

This API is only available on ICs that support it and must be enabled before use.

【Example】

None.

10.3.4 Audio Output

Audio output (AO) mainly realizes the functions of enabling audio output device and sending audio frame to output channel.

The API is listed below with additional details.

- `CVI_AO_SetPubAttr`: Set AO device properties.
- `CVI_AO_GetPubAttr`: Get AO device properties.
- `CVI_AO_Enable`: Enable AO devices.
- `CVI_AO_Disable`: Disable AO devices.
- `CVI_AO_EnableChn`: Enable AO channel.
- `CVI_AO_DisableChn`: Disable AO channel.
- `CVI_AO_SendFrame`: Send AO audio frame.
- `CVI_AO_EnableReSmp`: Enable AO resampling.
- `CVI_AO_DisableReSmp`: Disable AO resampling.
- `CVI_AO_PauseChn`: Pause AO channel.
- `CVI_AO_ResumeChn`: Restore AO channel.

- *CVI_AO_ClearChnBuf*: Clear the current audio data buffer in AO channel.
- *CVI_AO_QueryChnStat*: Query the current status of the audio data block in AO channel.
- *CVI_AO_SetTrackMode*: Set the AO device channel mode.
- *CVI_AO_GetTrackMode*: Get the AO device channel mode.
- *CVI_AO_SetVolume*: Set the volume of AO device.
- *CVI_AO_GetVolume*: Get the volume of AO device.
- *CVI_AO_SetMute*: Set the mute status of an AO device
- *CVI_AO_GetMute*: Get the mute status of an AO device
- *CVI_AO_SaveFile*: Enable audio output file saving functionality. (this function is not supported at present. Users can use *CVI_AO_SendFrame* to save files.);
- *CVI_AO_ClrPubAttr*: Clear AO device properties.

10.3.4.1 CVI_AO_SetPubAttr

【Description】

Set AO device properties.

【Syntax】

```
CVI_S32 CVI_AO_SetPubAttr(AUDIO_DEV AoDevId, const AIO_ATTR_S *pstAttr);
```

【Parameter】

Parameter	Description	Input/Output
AoDevId	Audio device number. The device number is pre-set to 0, and setting it to a value greater than 2 will return an error, unless customization requires expansion.	Input
pstAttr	Audio output device properties.	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: *cvl_comm_aio.h*, *cvl_audio.h*
- Library files: *libcvl_audio.a*

【Note】

- Before setting properties, you need to ensure that AO is disabled. If it is enabled, you need to disable AO devices first.
- AO must cooperate with DA in order to work normally. Users must know the relationship between the data format and channel sent by DA in order to send data from the correct channel.
- *u32ClkSel* does not need to be configured under CV182x chip.
- In the main mode of AO device, the key configuration items that determine the output clock of AO device are sampling rate, sampling precision and number of channels.

The sampling precision multiplied by the number of channels is the bit width of one sampling of the timing sequence of AO device.

- CV182x extended flag is invalid for AO device, so it does not need to be set.
- For other items in AO device attribute structure, please refer to the description of related interfaces in AI module.

【Example】

None.

10.3.4.2 CVI_AO_GetPubAttr

【Description】

Get AO device properties.

【Syntax】

```
CVI_S32 CVI_AO_GetPubAttr(AUDIO_DEV AoDevId, AIO_ATTR_S *pstAttr);
```

【Parameter】

Parameter	Description	Input/Output
AoDevId	Audio device number. The device number is pre-set to 0, and setting it to a value greater than 2 will return an error, unless customization requires expansion.	Input
pstAttr	Audio output device properties.	Output

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: -cvi_comm_aio.h, cvi_audio.h
- Library files: ibcvi_audio.a

【Note】

- The property obtained is the property of the previous configuration.
- If the property has never been configured, the error that the property is not configured is returned.

【Example】

```
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

【Description】

Enables AO devices.

【Syntax】

```
CVI_S32 CVI_AO_Enable(AUDIO_DEV AoDevId);
```

【Parameter】

Parameter	Description	Input/Output
AoDevId	Audio device number. The device number is preset to 0, and setting it to a value greater than 2 will return an error, unless customization requires expansion.	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: cvi_comm_aio.h, cvi_audio.h
- Library files: libcvi_audio.a

【Note】

- Require AO device properties to be configured before enabling, otherwise an error with properties not configured will be returned.
- If the AO device is enabled, Success is returned.

【Example】

```
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,
```

(continues on next page)

(continued from previous page)

```

        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);
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

【Description】

Deactivate AO device.

【Syntax】

```
CVI_S32 CVI_AO_Disable(AUDIO_DEV AoDevId);
```

【Parameter】

Parameter	Description	Input/Output
AoDevId	Audio device number. The device number is preset to 0, and setting it to a value greater than 2 will return an error, unless customization requires expansion.	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: cvi_comm_aio.h, cvi_audio.h
- Library files: libcvi_audio.a

【Note】

All AO channels under the device must be disabled before the AO device can be disabled.

【Example】

None.

10.3.4.5 CVI_AO_EnableChn**【Description】**

Enable AO channel.

【Syntax】

```
CVI_S32 CVI_AO_EnableChn(AUDIO_DEV AoDevId, AO_CHN AoChn);
```

【Parameter】

Parameter	Description	Input/Output
AoDevId	Audio device number. The device number is preset to 0, and setting it to a value greater than 2 will return an error, unless customization requires expansion.	Input
AoChn	Audio output channel number. The supported channel range is determined by the maximum number of channels u32ChnCnt in the AO device properties and the channel mode enSoundmode.	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: cvi_comm_aio.h, cvi_audio.h
- Library files: libcvi_audio.a

【Note】

Before enabling the AO channel, the AO device to which it belongs must be enabled, otherwise an error code that the device did not start is returned.

【Example】

None.

10.3.4.6 CVI_AO_DisableChn**【Description】**

Disable AO channel.

【Syntax】

```
CVI_S32 CVI_AO_DisableChn(AUDIO_DEV AoDevId, AO_CHN AoChn);
```

【Parameter】

Parameter	Description	Input/Output
AoDevId	Audio device number. The device number is preset to 0, and setting it to a value greater than 2 will return an error, unless customization requires expansion.	Input
AoChn	Audio output channel number. The supported channel range is determined by the maximum number of channels u32ChnCnt in the AO device properties and the channel mode enSoundmode.	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: cvi_comm_aio.h, cvi_audio.h
- Library files: libcvi_audio.a

【Note】

none

【Example】

None.

10.3.4.7 CVI_AO_SendFrame**【Description】**

Send AO audio frame.

【Syntax】

```
CVI_S32 _AO_SendFrame(AUDIO_DEV AoDevId, AO_CHN AoChn, const AUDIO_FRAME_S *pstData,
↪ CVI_S32 s32MilliSec);
```

【Parameter】

Parameter	Description	Input/Output
AoDevId	Audio device number. The device number is preset to 0, and setting it to a value greater than 2 will return an error, unless customization requires expansion.	Input
AoChn	Audio output channel number. The supported channel range is determined by the maximum number of channels u32ChnCnt in the AO device properties and the channel mode enSoundmode.	Input
pstData	Pointer to Audio frame structure.	Input
s32MilliSec	The timeout for sending data -1 indicates blocking mode; 0 means non-blocking mode; >0 indicates blocking s32MilliSec 毫秒 Returning an error if a timeout occurs.	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: cvi_comm_aio.h, cvi_audio.h
- Library files: libcvi_audio.a

【Note】

- This interface is used to actively send audio frames to AO output.
- The AO channel has been bound to AI or ADEC through the system binding (CVI_SYS_Bind) interface, and calling this interface is not necessary and not recommended.
- When calling this interface to send audio frames to the AO output channel, you must first enable the corresponding AO channel.

【Example】

None.

10.3.4.8 CVI_AO_EnableReSmp**【Description】**

Enable AO resampling.

【Syntax】

```
CVI_S32 CVI_AO_EnableReSmp(AUDIO_DEV AoDevId, AO_CHN AoChn, AUDIO_SAMPLE_RATE_
↪E enInSampleRate);
```

【Parameter】

Parameter	Description	Input/Output
AoDevId	Audio device number. The device number is preset to 0, and setting it to a value greater than 2 will return an error, unless customization requires expansion.	Input
AoChn	Audio output channel number. The supported channel range is determined by the maximum number of channels u32ChnCnt in the AO device properties and the channel mode enSoundmode.	Input
enInSampleRate	Input sampling rate for audio resampling.	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: cvi_comm_aio.h, cvi_audio.h
- Library files: libcvi_audio.a

【Note】

- This API is not supported when CVI_AUD_SYS_Bind is used
- This interface should be called to enable resampling after the AO channel is enabled and before binding the AO channel.
- Resampling is allowed to be enabled repeatedly, but the resampled input sampling rate of the post-configuration must be guaranteed to be the same as the previously configured resampled input sampling rate.
- If the AO channel is disabled and the resampling function is enabled again, you need to invoke this interface to enable resampling again.
- The input sampling rate for AO resampling must be different from the sampling rate for the AO device property configuration.

【Example】

None.

10.3.4.9 CVI_AO_DisableReSmp**【Description】**

Disable AO resampling.

【Syntax】

```
CVI_S32 CVI_AO_DisableReSmp(AUDIO_DEV AoDevId, AO_CHN AoChn);
```

【Parameter】

Parameter	Description	Input/Output
AoDevId	Audio device number. The device number is preset to 0, and setting it to a value greater than 2 will return an error, unless customization requires expansion.	Input
AoChn	Audio output channel number. The supported channel range is determined by the maximum number of channels u32ChnCnt in the AO device properties and the channel mode enSoundmode.	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: cvi_comm_aio.h, cvi_audio.h
- Library files: libcvi_audio.a

【Note】

If AO resampling is no longer used, this interface should be called to disable it.

【Example】

None.

10.3.4.10 CVI_AO_PauseChn

【Description】

Pause AO channel.

【Syntax】

```
CVI_S32 CVI_AO_PauseChn(AUDIO_DEV AoDevId, AO_CHN AoChn);
```

【Parameter】

Parameter	Description	Input/Output
AoDevId	Audio device number. The device number is preset to 0, and setting it to a value greater than 2 will return an error, unless customization requires expansion.	Input
AoChn	Audio output channel number. The supported channel range is determined by the maximum number of channels u32ChnCnt in the AO device properties and the channel mode enSoundmode.	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: cvi_comm_aio.h, cvi_audio.h
- Library files: libcvi_audio.a

【Note】

When the AO channel is paused, if the bound ADEC channel continues to send audio frame data to this channel, the transmitted audio frame data will be blocked.

If the bound AI channel continues to send audio frame data to this channel, the audio frame is put into the buffer when the channel buffer is not full, and the audio frame is discarded when it is full.

Calling this interface to pause the AO channel is not allowed when the AO channel is disabled.

【Example】

None.

10.3.4.11 CVI_AO_ResumeChn

【Description】

Restore AO channel.

【Syntax】

```
CVI_S32 CVI_AO_ResumeChn(AUDIO_DEV AoDevId, AO_CHN AoChn);
```

【Parameter】

Parameter	Description	Input/Output
AoDevId	Audio device number. The device number is preset to 0, and setting it to a value greater than 2 will return an error, unless customization requires expansion.	Input
AoChn	Audio output channel number. The supported channel range is determined by the maximum number of channels u32ChnCnt in the AO device properties and the channel mode enSoundmode.	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: cvi_comm_aio.h, cvi_audio.h
- Library files: libcvi_audio.a

【Note】

- The AO channel can be restored by calling when this interface is paused.
- If the AO channel is in the suspended or enabled state, the interface is invoked successfully. Otherwise the call will return an error.

【Example】

None.

10.3.4.12 CVI_AO_ClearChnBuf**【Description】**

Clear the current audio data buffer in the AO channel.

【Syntax】

```
CVI_S32 CVI_AO_ClearChnBuf(AUDIO_DEV AoDevId, AO_CHN AoChn);
```

【Parameter】

Parameter	Description	Input/Output
AoDevId	Audio device number. The device number is preset to 0, and setting it to a value greater than 2 will return an error, unless customization requires expansion.	Input
AoChn	Audio output channel number. The supported channel range is determined by the maximum number of channels u32ChnCnt in the AO device properties and the channel mode enSoundmode.	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: cvi_comm_aio.h, cvi_audio.h
- Library files: libcvi_audio.a

【Note】

- Call this interface after the AO channel is successfully enabled.
- In order to completely clear all buffer data on decoding playback path, this interface should be used in conjunction with the CVI_ADEC_ClearChnBuf interface.

【Example】

None.

10.3.4.13 CVI_AO_QueryChnStat**【Description】**

Query the current status of audio data block in the AO channel.

【Syntax】

```
CVI_S32 CVI_AO_QueryChnStat(AUDIO_DEV AoDevId, AO_CHN AoChn, AO_CHN_STATE_S_
↪*pstStatus);
```

【Parameter】

Parameter	Description	Input/Output
AoDevId	Audio device number. The device number is pre-set to 0, and setting it to a value greater than 2 will return an error, unless customization requires expansion.	Input
AoChn	Audio output channel number. The supported channel range is determined by the maximum number of channels u32ChnCnt in the AO device properties and the channel mode enSoundmode.	Input
pstStatus	Pointer to block status structure	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: cvi_comm_aio.h, cvi_audio.h
- Library files: libcvi_audio.a

【Note】

Call this interface after the AO channel is successfully enabled.

【Example】

None.

10.3.4.14 CVI_AO_SetTrackMode

【Description】

Set AO device channel mode.

【Syntax】

```
CVI_S32 CVI_AO_SetTrackMode(AUDIO_DEV AoDevId, AUDIO_TRACK_MODE_E enTrackMode);
```

【Parameter】

Parameter	Description	Input/Output
AoDevId	Audio device number. The device number is preset to 0, and setting it to a value greater than 2 will return an error, unless customization requires expansion.	Input
enTrackMode	Audio device channel mode.	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: cvi_comm_aio.h, cvi_audio.h
- Library files: libcvi_audio.a

【Note】

- Call this interface after the AO device is successfully enabled.
- AO supports setting the channel mode when it works in I2S mode, but does not support setting the channel mode when it works in PCM mode.

【Example】

```
CVI_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

【Description】

Gets the channel mode of the AO device.

【Syntax】

```
CVI_S32 CVI_AO_GetTrackMode(AUDIO_DEV AoDevId, AUDIO_TRACK_MODE_E *penTrackMode);
```

【Parameter】

Parameter	Description	Input/Output
AoDevId	Audio device number. The device number is preset to 0, and setting it to a value greater than 2 will return an error, unless customization requires expansion.	Input
enTrackMode	Audio device channel mode.	Output

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: cvi_comm_aio.h, cvi_audio.h
- Library files: libcvi_audio.a

【Note】

- Call this interface after the AO device is successfully enabled.
- AO supports setting the channel mode when it works in I2S mode, but does not support setting the channel mode when it works in PCM mode.

【Example】

None.

10.3.4.16 CVI_AO_SetVolume

【Description】

Set AO device volume.

【Syntax】

```
CVI_S32 CVI_AO_SetVolume(AUDIO_DEV AoDevId, CVI_S32 s32VolumeDb);
```

【Parameter】

Parameter	Description	Input/Output
AoDevId	Audio device number. The device number is preset to 0, and setting it to a value greater than 2 will return an error, unless customization requires expansion.	Input
s32VolumeDb	The volume range is from 32 to 0, which corresponds to a gain range from 0dB to -22.5dB, with a decrease of 1.5dB for each step.	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: cvi_comm_aio.h, cvi_audio.h
- Library files: libcvi_audio.a

【Note】

- Call this interface after the AO device is successfully enabled.

【Example】

None.

10.3.4.17 CVI_AO_GetVolume**【Description】**

Gets the volume size of the AO device.

【Syntax】

```
CVI_S32 CVI_AO_GetVolume(AUDIO_DEV AoDevId, CVI_S32 *ps32VolumeDb);
```

【Parameter】

Parameter	Description	Input/Output
AoDevId	Audio device number. The device number is pre-set to 0, and setting it to a value greater than 2 will return an error, unless customization requires expansion.	Input
ps32VolumeDb	Pointer to the Ao device volume. The volume range is from 32 to 0, which corresponds to a gain range from 0dB to -22.5dB, with a decrease of 1.5dB for each step.	Output

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: cvi_comm_aio.h, cvi_audio.h
- Library files: libcvi_audio.a

【Note】

Call this interface after the AO device is successfully enabled.

【Example】

None.

10.3.4.18 CVI_AO_SetMute

【Description】

Set the mute status of an AO device

【Syntax】

```
CVI_S32 CVI_AO_SetMute(AUDIO_DEV AoDevId, CVI_BOOL bEnable, const AUDIO_FADE_S_
↪*pstFade);
```

【Parameter】

Parameter	Description	Input/Output
AoDevId	Audio device number. The device number is pre-set to 0, and setting it to a value greater than 2 will return an error, unless customization requires expansion.	Input
bEnable	Whether the audio device is muted. CVI_TRUE: Enable mute function; CVI_FALSE: Turn off mute function	Input
pstFade	Fade in and out structure pointers.	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: cvi_comm_aio.h, cvi_audio.h
- Library files: libcvi_audio.a

【Note】

- Call this interface after the AO device is successfully enabled.

【Example】

None.

10.3.4.19 CVI_AO_GetMute

【Description】

Gets the mute status of the AO device.

【Syntax】

```
CVI_S32 CVI_AO_GetMute(AUDIO_DEV AoDevId, CVI_BOOL *pbEnable, AUDIO_FADE_S *pstFade);
```

【Parameter】

Parameter	Description	Input/Output
AoDevId	Audio device number. The device number is pre-set to 0, and setting it to a value greater than 2 will return an error, unless customization requires expansion.	Input
bEnable	Audio device mute state pointer.	Output
pstFade	Fade in and out structure pointers.	Output

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: `cvi_comm_aio.h`, `cvi_audio.h`
- Library files: `libcvi_audio.a`

【Note】

- Call this interface after the AO device is successfully enabled.

【Example】

None.

10.3.4.20 CVI_AO_SaveFile**【Description】**

This function is not supported at present. Users can use `CVI_AO_SendFrame` to save files.

【Syntax】

```
CVI_S32 CVI_AO_SaveFile(AUDIO_DEV AoDevId, AO_CHN AoChn, AUDIO_SAVE_FILE_INFO_S
↪*pstSaveFileInfo);
```

【Parameter】**【Return Value】****【Requirement】**

- Header files:
- Library files:

【Note】**【Example】**

None.

10.3.4.21 CVI_AO_ClrPubAttr**【Description】**

Clear AO device properties.

【Syntax】

```
CVI_S32 CVI_AO_ClrPubAttr(AUDIO_DEV AoDevId);
```

【Parameter】

Parameter	Description	Input/Output
AoDevId	Audio device number. The device number is pre-set to 0, and setting it to a value greater than 2 will return an error, unless customization requires expansion.	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: `cvi_comm_aio.h`, `cvi_audio.h`
- Library files: `libcvi_audio.a`

【Note】

- You need to stop the device before clearing its properties.
- AI device and AO device clocks do not need to be set.

【Example】

None.

10.3.5 Audio Encoding

Audio encoding is mainly dedicated to voice frame data conversion.

Cvitek supports G.711, G.726, LDPCM-related encoding.

The encoded voice frame data is small, but the voice encoding belongs to lossy compression, which results in different sound quality.

The bit rate/sample rate used by different encoding varies.

Users need to know the specifications of the encoding protocol and set the corresponding API.

Otherwise the function will return an error.

Codec	Sampling Rate	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 include a-law/mu-law

The AENC module provides the following APIs:

- *`CVI_AENC_CreateChn`*: Create an audio encoding channel.
- *`CVI_AENC_DestroyChn`*: Destroy the audio encoding channel.
- *`CVI_AENC_SendFrame`*: Send Audio Coded Audio Frames
- *`CVI_AENC_GetStream`*: Gets the audio encoding stream.
- *`CVI_AENC_ReleaseStream`*: Release the audio encoding stream.
- *`CVI_AENC_SaveFile`*: Enabling channel file recording functionality before audio encoding
- *`CVI_AENC_QueryFileStatus`*: Checking the file storage status of the audio encoding channel
- *`CVI_AENC_GetStreamBufInfo`*: Get information about the audio stream buffer.
- *`CVI_AENC_SetMute`*: Set the mute status of an AENC (Audio Encode) channel.
- *`CVI_AENC_GetMute`*: Get the mute status of an AENC (Audio Encode) channel

10.3.5.1 CVI_AENC_CreateChn

【Description】

Create an audio encoding channel.

【Syntax】

```
CVI_S32 CVI_AENC_CreateChn(AENC_CHN AeChn, const AENC_CHN_ATTR_S *pstAttr);
```

【Parameter】

Parameter	Description	Input/Output
AeChn	Encoding device channel number. Range of values: [0, AENC_MAX_CHN_NUM]	Input
pstAttr	Encoded Channel Property Pointer	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: cvi_comm_aio.h, cvi_audio.h, cvi_comm_aenc.h
- Library files: libcvi_audio.a, libcvi_transcode.so, libcvi_RES1.so

【Note】

- Currently supports G711, G726, ADPCM
- The format supports 16-bit S16LE, and others are not supported.

Set AI attribute/AENC attribute.

If you want to use it after VQE, set the length of the sound frame to a multiple of 160.

To comply with the Cvitek VQE specification.

- Some of the attributes of audio encoding need to match the attributes of the input audio data, such as sampling rate, frame length (number of sampling points per frame), etc.
- The buffer size is expressed in frames.

The Value range is [2, MAX_BUFFERING_DEPTH].

You are advised to set the value to more than 10.

Too small buffer configuration may cause exceptions such as frame loss.

【Example】

None.

10.3.5.2 CVI_AENC_DestroyChn

【Description】

Destroy audio codec pass.

【Syntax】

```
CVI_S32 CVI_AENC_DestroyChn(AENC_CHN AeChn);
```

【Parameter】

Parameter	Description	Input/Output
AeChn	Encoding device channel number. Value Range: [0, AENC_MAX_CHN_NUM]	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: cvi_comm_aio.h, cvi_audio.h, cvi_comm_aenc.h
- Library files: libcvi_audio.a, libcvi_transcode.so, libcvi_RES1.so

【Note】

- The call to this function is invalid if the channel is not created.
- If the channel is destroyed while obtaining/releasing the stream or sending the frame, a failure will be returned.
- Users should pay attention when synchronizing the process

【Example】

None.

10.3.5.3 CVI_AENC_SendFrame

【Description】

Send audio encoded audio frames.

【Syntax】

```
CVI_S32 CVI_AENC_SendFrame(AENC_CHN AeChn, const AUDIO_FRAME_S *pstFrm, const AEC_
↪FRAME_S *pstAecFrm);
```

【Parameter】

Parameter	Description	Input/Output
AeChn	Encoding device channel number. Range of values: [0, AENC_MAX_CHN_NUM]	Input
pstFrm	Audio frame structure pointer.	Input
pstAecFrm	Echo cancellation reference frame structure pointer.	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: `cvl_comm_aio.h`, `cvl_audio.h`, `cvl_comm_aenc.h`
- Library files: `libcvl_audio.a`, `libcvl_transcode.so`, `libcvl_RES1.so`

【Note】

- The call to this function is invalid if the channel is not created.
- `pstAecFrm` can be set to NULL if echo cancellation is not required.
- This interface is used to actively send audio frames for encoding. If the AENC channel is already bound to AI through the system binding (`CVI_SYS_Bind`) interface, calling this interface is unnecessary and not advised.

【Example】

```

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 */

```

(continues on next page)

(continued from previous page)

```

/* 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",
↪\n",
        __func__, AoDev, AoChn, s32Ret);
        bStart = CVI_FALSE;
        return NULL;
    }
}

/* finally you must release the stream */
s32Ret = CVI_AI_ReleaseFrame( AiDev,  AiChn, &stFrame,
                             &stAecFrm);

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

【Description】

Gets the encoded stream.

【Syntax】

```

CVI_S32 CVI_AENC_GetStream(AENC_CHN AeChn, AUDIO_STREAM_S *pstStream, CVI_S32↪
↪s32MilliSec);

```

【Parameter】

Parameter	Description	Input/Output
AeChn	Encoding device channel number. Range of values: [0, AENC_MAX_CHN_NUM]	Input
pstStream	The obtained audio stream.	Output
s32MilliSec	Timeout to acquiring data 1 indicates blocking mode and waits until there is no data; 0 means non-blocking mode, error returns when no data is available; >0 indicates blocking s32MilliSec milliseconds. Returning an error if a timeout occurs	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: `cvi_comm_aio.h`, `cvi_audio.h`, `cvi_comm_aenc.h`
- Library files: `libcvi_audio.a`, `libcvi_transcode.so`, `libcvi_RES1.so`

【Note】

- The code stream can be obtained only after the channel is created. Otherwise, a direct failure is returned.

If the channel is destroyed in the process of obtaining the code stream, a failure is returned immediately.

- The value of `s32MilliSec` must be greater than or equal to -1, blocking mode for data when the value is equal to -1, non-blocking mode for data when it is equal to 0, and blocking `s32MilliSec` milliseconds when greater than 0, returning a timeout and reporting if no data.

【Example】

None.

10.3.5.5 CVI_AENC_ReleaseStream**【Description】**

Release the stream obtained from the audio encoding channel.

【Syntax】

```
CVI_S32 CVI_AENC_ReleaseStream(AENC_CHN AeChn, const AUDIO_STREAM_S *pstStream);
```

【Parameter】

Parameter	Description	Input/Output
AeChn	Encoding device channel number. Range of values: [0, AENC_MAX_CHN_NUM]	Input
pstStream	Pointer to the audio stream.	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: `cvi_comm_aio.h`, `cvi_audio.h`, `cvi_comm_aenc.h`
- Library files: `libcvi_audio.a`, `libcvi_transcode.so`, `libcvi_RES1.so`

【Note】

- It is best to release the code stream immediately after use.

If not released in time, the coding process will be blocked.

- The code stream to be released must be the code stream obtained from the channel.

Do not modify the code stream information structure; otherwise, the code stream cannot be released, the buffer of the code stream will be lost, and even program exceptions will be caused.

- When releasing the stream, ensure that the channel has been created.

Otherwise, a direct failure is returned.

If the channel is destroyed during the release process, an immediate failure is returned.

【Example】

None.

10.3.5.6 CVI_AENC_SaveFile**【Description】**

Enabling channel file recording functionality before audio encoding

【Syntax】

```
CVI_S32 CVI_AENC_SaveFile(AENC_CHN AeChn, const AUDIO_SAVE_FILE_INFO_S_
↪*pstSaveFileInfo);
```

【Parameter】

Parameter	Description	Input/Output
AeChn	Audio encoding channel number. Range of values: [0, AENC_MAX_CHN_NUM].	Input
pstSaveFileInfo	Audio save file property structure pointer.	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: cvi_comm_aio.h, cvi_audio.h, cvi_comm_aenc.h
- Library files: libcvi_audio.a, libcvi_transcode.so, libcvi_RES1.so

【Note】

- Cvitek does not support this API.
-Please use CVI_AI_SaveFile.

【Example】

None.

10.3.5.7 CVI_AENC_QueryFileStatus**【Description】**

Checking the file storage status of the audio encoding channel

【Syntax】

```
CVI_S32 CVI_AENC_QueryFileStatus(AENC_CHN AeChn, AUDIO_FILE_STATUS_S *pstFileStatus);
```

【Parameter】

Parameter	Description	Input/Output
AeChn	Audio encoding channel number. Range of values: [0, AENC_MAX_CHN_NUM].	Input
pstFileStatus	Pointer to status attribute structure	Output

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: cvi_comm_aio.h, cvi_audio.h, cvi_comm_aenc.h
- Library files: libcvi_audio.a, libcvi_transcode.so, libcvi_RES1.so

【Note】

- Cvitek does not support this API.
- Please use CVI_AI_SaveFile/CVI_AI_QueryFileStatus

【Example】

None.

10.3.5.8 CVI_AENC_GetStreamBufInfo**【Description】**

Get information about the audio stream buffer.

【Syntax】

```
CVI_S32 CVI_AENC_GetStreamBufInfo(AENC_CHN AeChn, CVI_U32 *pu32PhysAddr, CVI_U32 ↵
↵ *pu32Size);
```

【Parameter】

Parameter	Description	Input/Output
AeChn	Audio encoding channel number. Range of values: [0, AENC_MAX_CHN_NUM].	Input
pu32PhysAddr	The physical address of the audio stream buffer.	Output
pu32Size	Length of audio stream buffer in byte	Output

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: cvi_comm_aio.h, cvi_audio.h, cvi_comm_aenc.h
- Library files: libcvi_audio.a, libcvi_transcode.so, libcvi_RES1.so

【Note】

None.

【Example】

None.

10.3.5.9 CVI_AENC_SetMute

【Description】

Set AENC mute state

【Syntax】

```
CVI_AENC_SetMute(AENC_CHN AeChn,CVI_BOO; bEnable);
```

【Parameter】

Parameter	Description	Input/Output
AeChn	Encoding device channel number. Range of values: [0, AENC_MAX_CHN_NUM]	Input
bEnable	Whether the audio device is muted. CVI_TRUE: Enable mute function; CVI_FALSE: Turn off mute function	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: cvi_audio.h,
- Library files: libcvi_audio.a

【Note】

- Use only after AENC creates a device channel.

【Example】

None.

10.3.5.10 CVI_AENC_GetMute

【Description】

Get the mute status of an AENC (Audio Encode) channel

【Syntax】

```
CVI_AENC_GetMute(AENC_CHN AeChn, CVI_BOOL *pbEnable);
```

【Parameter】

Parameter	Description	Input/Output
AeChn	Encoding device channel number. Range of values: [0, AENC_MAX_CHN_NUM]	Input
pbEnable	Pointer to audio device mute status	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: `cvi_audio.h`
- Library files: `libcvi_audio.a`

【Note】

- Use only after AENC creates a device channel

【Example】

None.

`libcvi_audio.a`, `libcvi_transcode.so`, `libcvi_RES1.so`

10.3.6 Audio Decoding

The main functions of audio decoding include creating decoding channels, decoding audio streams, and obtaining decoded audio frames

The supported APIs and details are listed below:

- *CVI_ADEC_CreateChn*: Create an audio decoding channel.
- *CVI_ADEC_DestroyChn*: Destroy the audio decoding channel.
- *CVI_ADEC_SendStream*: Sends an audio stream to an audio decoding channel.
- *CVI_ADEC_ClearChnBuf*: Clear the current audio data block in the ADEC channel.
- *CVI_ADEC_GetFrame*: Gets the audio decoded frame data.
- *CVI_ADEC_ReleaseFrame*: Release audio decoded frame data.
- *CVI_ADEC_SendEndOfStream*: Sending the end-of-stream (EOS) marker to the decoder and clearing the stream buffer.

10.3.6.1 CVI_ADEC_CreateChn

【Description】

Create an audio decoding channel.

【Syntax】

```
CVI_S32 CVI_ADEC_CreateChn(ADEC_CHN AdChn, const ADEC_CHN_ATTR_S *pstAttr);
```

【Parameter】

Parameter	Description	Input/Output
AdChn	Channel number. Range: [0, ADEC_MAX_CHN_NUM].	Input
pstAttr	Channel property pointer.	Input

【Return Value】

Parameter	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: `cvi_comm_aio.h`, `cvi_audio.h`, `cvi_comm_aenc.h`
- Library files: `libcvi_audio.a`, `libcvi_transcode.so`, `libcvi_RES1.so`

【Note】

- Currently support G711, G726, ADPCM.
- Some properties of audio decoding need to match those of the output device, such as sampling rate, frame length (number of sampling points per frame), etc.

【Example】

None.

10.3.6.2 CVI_ADEC_DestroyChn**【Description】**

Destroy the audio decoding channel.

【Syntax】

```
CVI_S32 CVI_ADEC_DestroyChn(ADEC_CHN AdChn);
```

【Parameter】

Parameter	Description	Input/Output
AdChn	Channel number. Range: [0, ADEC_MAX_CHN_NUM].	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: `cvi_comm_aio.h`, `cvi_audio.h`, `cvi_comm_aenc.h`
- Library files: `libcvi_audio.a`, `libcvi_transcode.so`, `libcvi_RES1.so`

【Note】

- Calling this interface without channel creation returns Success.
- If a stream is being acquired/released or a frame is being sent, destroying the channel will return failure immediately.

【Example】

None.

10.3.6.3 CVI_ADEC_SendStream

【Description】

Sends a stream to an audio decoding channel.

【Syntax】

```
CVI_S32 CVI_ADEC_SendStream(ADEC_CHN AdChn, const AUDIO_STREAM_S *pstStream, CVI_BOOL bBlock);
```

【Parameter】

Parameter	Description	Input/Output
AdChn	Channel number. Range: [0, ADEC_MAX_CHN_NUM].	Input
pstStream	Audio stream	Input
bBlock	CVI_TRUE: Blocked. CVI_FALSE: Non-blocking.	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: cvi_comm_aio.h, cvi_audio.h, cvi_comm_aenc.h
- Library files: libcvi_audio.a, libcvi_transcode.so, libcvi_RES1.so

【Note】

- The stream mode is inefficient and may have delay when the bitstream packet size is not determined to be one frame (greater than or equal to one frame);
- Ensure that a channel has been created before sending data.
Otherwise, a direct failure is returned.
If the channel is destroyed during data sending, a failure is returned immediately.
- Ensure the correctness of the stream data sent to the ADEC channel, otherwise the decoder may exit abnormally.

【Example】

None.

10.3.6.4 CVI_ADEC_ClearChnBuf

【Description】

Clear the current audio data block in the ADEC channel.

【Syntax】

```
CVI_S32 CVI_ADEC_ClearChnBuf(ADEC_CHN AdChn);
```

【Parameter】

Parameter	Description	Input/Output
AdChn	Channel number. Range: [0, ADEC_MAX_CHN_NUM].	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: cvi_comm_aio.h, cvi_audio.h, cvi_comm_aenc.h
- Library files: libcvi_audio.a, libcvi_transcode.so, libcvi_RES1.so

【Note】

- Decoding channel has been created is required.
- Stream decoding is not recommended when using this interface. When using streaming decoding to clear blocks, the user needs to ensure that after clearing blocks, the data sent to the decoder must be a complete frame stream, otherwise the decoder may not operate properly.
- Regardless of whether stream decoding is utilized, it is crucial to ensure synchronization between the data feeding operation and the buffer clearing operation.

【Example】

none

10.3.6.5 CVI_ADEC_GetFrame**【Description】**

Gets the decoded audio frame.

【Syntax】

```
CVI_S32 CVI_ADEC_GetFrame(ADEC_CHN AdChn, AUDIO_FRAME_INFO_S *pstFrmInfo, CVI_BOOL
↪ bBlock);
```

【Parameter】

Parameter	Description	Input/Output
AdChn	Audio decoding channel.	Input
pstFrmInfo	Audio frame data structure.	Output
bBlock	Whether to retrieve data in a blocking mode.	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: -cvi_comm_aio.h, cvi_audio.h, cvi_comm_aenc.h
- Library files: -libcvi_audio.a, libcvi_transcode.so, libcvi_RES1.so

【Note】

- Must be called after the ADEC channel is created.
- When using this interface to obtain decoded frame data, it is recommended to send the stream by frame.
- When using this interface to obtain audio data, unbind ADEC from AO, otherwise the frames are discontinuous.
- When you use this interface to obtain audio frame data, ensure that the decoded frame data is obtained in a timely manner if the sent code stream is sent as stream. Otherwise, an exception may occur.

【Example】

None.

10.3.6.6 CVI_ADEC_ReleaseFrame**【Description】**

Release the acquired audio decoded frame data.

【Syntax】

```
CVI_S32 CVI_ADEC_ReleaseFrame(ADEC_CHN AdChn, AUDIO_FRAME_INFO_S *pstFrmInfo);
```

【Parameter】

Parameter	Description	Input/Output
AdChn	Audio decoding channel.	Input
pstFrmInfo	Audio frame data structure.	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: cvi_comm_aio.h, cvi_audio.h, cvi_comm_aenc.h
- Library files: libcvi_audio.a, libcvi_transcode.so, libcvi_RES1.so

【Note】

- This interface must be used together with the CVI_ADEC_GetFrame interface.
- Must be called after the ADEC channel is created.

【Example】

None.

10.3.6.7 CVI_ADEC_SendEndOfStream

【Description】

Sending the end-of-stream (EOS) marker to the decoder and clearing the stream buffer.

【Syntax】

```
CVI_S32 CVI_ADEC_SendEndOfStream (ADEC_CHN AdChn, CVI_BOOL bInstant);
```

【Parameter】

Parameter	Description	Input/Output
AdChn	Audio decoding channel.	Input
bInstant	The parameter specifies in whether to immediately clear the internal buffer data of the decoder. The Value range is: - CVI_FALSE: Delayed clearing. The decoder' s internal buffer data will not be cleared immediately, and the decoding process will continue until the remaining buffer is not enough for one frame of data. The clearing operation will then be performed. CVI_TRUE: Immediate clearing of the decoder' s internal buffer data."	Input/Output

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: cvi_comm_aio.h, cvi_audio.h, cvi_comm_aenc.h
- Library files: libcvi_audio.a, libcvi_transcode.so, libcvi_RES1.so

【Note】

None.

【Example】

None.

10.3.7 Built-in Codec

The built-in audio codec is mainly operated through ioctl to manipulate the hardware device. The ioctl calls are used to read and write to the registers of the built-in audio codec.

Built-in audio codec standard function cmd

- *ACODEC_SOFT_RESET_CTRL*: Restore Codec to its default setting.
- *ACODEC_SET_INPUT_VOL*: Total input volume control.
- *ACODEC_GET_INPUT_VOL*: Gets the total input volume.
- *ACODEC_SET_OUTPUT_VOL*: Total output volume control.
- *ACODEC_GET_OUTPUT_VOL*: Gets the total output volume.

- *ACODEC_SET_GAIN_MICL*: Analog Gain Control of Left Channel MIC Input
- *ACODEC_SET_GAIN_MICR*: Analog Gain Control of Right Channel MIC Input
- *ACODEC_SET_DACL_VOL*: Left channel output volume control.
- *ACODEC_SET_DACR_VOL*: Right channel output volume control.
- *ACODEC_SET_ADCL_VOL*: Left channel input volume control.
- *ACODEC_SET_ADCR_VOL*: Right channel input volume control.
- *ACODEC_SET_MICL_MUTE*: Left channel MIC input mute control.
- *ACODEC_SET_MICR_MUTE*: Right channel MIC input mute control.
- *ACODEC_SET_DACL_MUTE*: Left channel output mute control.
- *ACODEC_SET_DACR_MUTE*: Right channel output mute control.
- *ACODEC_GET_GAIN_MICL*: Get Analog Gain of Left Channel MIC Input .
- *ACODEC_GET_GAIN_MICR*: Get Analog Gain of Right Channel MIC Input .
- *ACODEC_GET_DACL_VOL*: Gets the volume control of the left channel output.
- *ACODEC_GET_DACR_VOL*: Gets the volume control of the right channel output.
- *ACODEC_GET_ADCL_VOL*: Gets the volume control of the left channel input.
- *ACODEC_GET_ADCR_VOL*: Gets the volume control of the right channel input.
- *ACODEC_SET_PD_DACL*: Power-Off Control of Left Channel Output .
- *ACODEC_SET_PD_DACR*: Power-Off Control of Right Channel Output .
- *ACODEC_SET_PD_ADCL*: Power-Off Control of Left Channel Input .
- *ACODEC_SET_PD_ADCR*: Power-Off Control of Right Channel Input .
- *ACODEC_SET_PD_LINEINL*: Power-Off Control of Left Channel LINEIN Input .
- *ACODEC_SET_PD_LINEINR*: Power-Off Control of Right Channel LINEIN Input .

10.3.7.1 ACODEC_SOFT_RESET_CTRL

【Description】

Restore Codec to its default setting.

【Syntax】

```
CVI_S32 ioctl (CVI_S32 fd, ACODEC_SOFT_RESET_CTRL);
```

【Parameter】

Parameter	Description	Input/Output
fd	File Description of Audio Codec Device	Input
ACODEC_SOFT_RESET_CTRL	ioctl number	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: `acodec.h`

【Note】

None.

【Example】

```
if (ioctl(Acodec_Fd, ACODEC_SOFT_RESET_CTRL))
{
    printf("ioctl reset err!\n");
}
```

10.3.7.2 ACODEC_SET_INPUT_VOL**【Description】**

Total input volume control. Configure the user's desired gain to the digital gain control register.

【Syntax】

```
CVI_S32 ioctl (CVI_S32 fd, ACODEC_SET_INPUT_VOL, CVI_U32 *arg);
```

【Parameter】

Parameter	Description	Input/Output
fd	File Description of Audio Codec Device	Input
ACODEC_SET_INPUT_VOL	IOVOL number	Input
arg	Unsigned integer pointer	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: `acodec.h`

【Note】

- The input volume ranges from [24 to 0]. If the input value is 0, it is mute.
- Both left and right channels are in effect during adjustment.

【Example】

```
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

【Description】

Gets the total input volume.

【Syntax】

```
CVI_S32 ioctl (CVI_S32 fd, ACODEC_GET_INPUT_VOL, CVI_U32 *arg);
```

【Parameter】

Parameter	Description	Input/Output
fd	Audio Codec Device File Description	Input
ACODEC_GET_INPUT_VOL	ioctl number	Input
arg	Unsigned integer pointer	Output

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: `acodec.h`

【Note】

- The input volume ranges from [24 to 0]. If the input value is 0, it is mute.

【Example】

```
CVI_U32 vol;
if (ioctl(Acodec_Fd, ACODEC_SET_INPUT_VOL, &vol))
{
    printf("ioctl err!\n");
}
```

10.3.7.4 ACODEC_SET_OUTPUT_VOL

【Description】

Total output volume control. Configure the user' s desired gain to the digital gain control register.

【Syntax】

```
CVI_S32 ioctl (CVI_S32 fd, ACODEC_SET_OUTPUT_VOL, CVI_U32 *arg);
```

【Parameter】

Parameter	Description	Input/Output
fd	Audio Codec Device File Description	Input
ACODEC_SET_OUTPUT_VOL	ioctl number	Input
arg	Unsigned integer pointer	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: `acodec.h`

【Note】

- The output volume ranges from [32 to 0].
- Both left and right channels are in effect during adjustment.

【Example】

```
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**【Description】**

Get the total output volume

【Syntax】

```
CVI_S32 ioctl (CVI_S32 fd, ACODEC_GET_OUTPUT_VOL, CVI_U32 *arg);
```

【Parameter】

Parameter	Description	Input/Output
fd	Audio Codec Device File Description	Input
ACODEC_GET_OUTPUT_VOL	ioctl number	Input
arg	Unsigned integer pointer	Output

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: `acodec.h`

【Note】

The output volume range is [32~0].

【Example】

```
CVI_U32 vol;
if (ioctl(Acodec_Fd, ACODEC_SET_INPUT_VOL, &vol))
{
    printf("ioctl err!\n");
}
```

10.3.7.6 ACODEC_SET_GAIN_MICL

【Description】

Analog Gain Control of Left Channel MIC Input

【Syntax】

```
CVI_S32 ioctl (CVI_S32 fd, ACODEC_SET_GAIN_MICL, CVI_U32 *arg);
```

【Parameter】

Parameter	Description	Input/Output
fd	Audio Codec Device File Description	Input
ACODEC_SET_GAIN_MICL	MICL number	Input
arg	Unsigned integer pointer	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: `acodec.h`

【Note】

- When using passive audio signal input, it is necessary to appropriately increase the analog gain.
- The input volume range is [24~0]. The input value 0 represent mute.

【Example】

```
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

【Description】

Analog Gain Control of Right Channel MIC Input

【Syntax】

```
CVI_S32 ioctl (CVI_S32 fd, ACODEC_SET_GAIN_MICR, CVI_U32 *arg);
```

【Parameter】

Parameter	Description	Input/Output
fd	Audio Codec Device File Description	Input
ACODEC_SET_GAIN_MICR	MICR number	Input
arg	Unsigned integer pointer	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: acodec.h

【Note】

- When using passive audio signal input, it is necessary to appropriately increase the analog gain.
- The input volume range is [24~0]. The input value 0 represent mute.

【Example】

```
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**【Description】**

Left channel output volume control.

【Syntax】

```
CVI_S32 ioctl (CVI_S32 fd, ACODEC_SET_DACL_VOL, struct cvi_vol_ctrl *arg);
```

【Parameter】

Parameter	Description	Input/Output
fd	Audio Codec Device File Description	Input
ACODEC_SET_DACL_VOL	VOL number	Input
arg	cvi_vol_ctrl structure pointer	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: acodec.h

【Note】

Output volume ranges from [32 to 0].

If vol_ctrl_ctrl_mute is set to 1, the mute will be enabled.

If vol_ctrl.vol_ctrl_mute is set to 0, the mute will be disabled.

【Example】

```

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");
}

```

10.3.7.9 ACODEC_SET_DACR_VOL

【Description】

Right channel output volume control.

【Syntax】

```
CVI_S32 ioctl (CVI_S32 fd, ACODEC_SET_DACR_VOL, struct cvi_vol_ctrl *arg);
```

【Parameter】

Parameter	Description	Input/Output
fd	Audio Codec Device File Description	Input
ACODEC_SET_DACR_VOL	VOLnumber	Input
arg	cvi_vol_ctrl structure pointer	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: `acodec.h`

【Note】

Output volume ranges from [32 to 0]. If `vol_ctrl_ctrl_mute` is set to 1, the mute will be enabled. If `vol_ctrl.vol_ctrl_mute` is set to 0, the mute will be disabled.

【Example】

```

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

【Description】

Left channel input volume control.

【Syntax】

```
CVI_S32 ioctl (CVI_S32 fd, ACODEC_SET_ADCL_VOL, struct cvi_vol_ctrl *arg);
```

【Parameters】

Parameter	Description	Input/Output
fd	Audio Codec device file description	Input
ACODEC_SET_ADCL_VOL	VOL number	Input
arg	cvi_vol_ctrl structure pointer	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files:acodec.h

【Note】

- When the Input is configured as MIC in, the input volume range is [24~0]. If the value of vol_ctrl.vol_ctrl_mute is 1, it will be muted, and if the value is 0, the mute will be cancelled.

【Example】

```
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

【Description】

Right channel input volume control.

【Syntax】

```
CVI_S32 ioctl (CVI_S32 fd, ACODEC_SET_ADCR_VOL, struct cvi_vol_ctrl *arg);
```

【Parameter】

Parameter	Description	Input/Output
fd	Audio codec device file description	Input
ACODEC_SET_ADCR_VOL	VOL number	Input
arg	cvi_vol_ctrl structure pointer	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: `acodec.h`

【Note】

- When the Input is configured as MIC in, the input volume range is [24~0]. If the value of `vol_ctrl.vol_ctrl_mute` is 1, it will be muted, and if the value is 0, the mute will be cancelled.

【Example】

```

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**【Description】**

Left channel MIC input mute control.

【Syntax】

```
CVI_S32 ioctl (CVI_S32 fd, ACODEC_SET_MICL_MUTE, CVI_U32 *arg);
```

【Parameter】

Parameter	Description	Input/Output
fd	Audio codec device file description	Input
ACODEC_SET_MICL_MUTE	Mute Number	Input
arg	Unsigned integer pointer	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: `acodec.h`

【Note】

The arg parameter ranges from 0 to 1, where 0 means unmute and 1 means mute.

【Example】

```

CVI_U32 mic_mute;
mic_mute = 0;
if (ioctl(Acodec_Fd, ACODEC_SET_MICL_MUTE, &mic_mute))

```

(continues on next page)

(continued from previous page)

```
{
printf("ioctl err!\n");
}
```

10.3.7.13 ACODEC_SET_MICR_MUTE

【Description】

Right channel MIC input mute control.

【Syntax】

```
CVI_S32 ioctl (CVI_S32 fd, ACODEC_SET_MICR_MUTE, CVI_U32 *arg);
```

【Parameter】

Parameter	Description	Input/Output
fd	Audio codec device file description	Input
ACODEC_SET_MICR_MUTE	Mute number	Input
arg	Unsigned integer pointer	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: `acodec.h`

【Note】

The `arg` parameter ranges from 0 to 1, where 0 means unmute and 1 means mute.

【Example】

```
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

【Description】

Left channel output mute control.

【Syntax】

```
CVI_S32 ioctl (CVI_S32 fd, ACODEC_SET_DACL_MUTE, CVI_U32 *arg);
```

【Parameter】

Parameter	Description	Input/Output
fd	Audio codec device file description	Input
ACODEC_SET_DACL_MUTE	Mute number	Input
arg	Unsigned integer pointer	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: `acodec.h`

【Note】

The arg parameter ranges from 0 to 1, where 0 means unmute and 1 means mute.

【Example】

```
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**【Description】**

Right channel output mute control.

【Syntax】

```
CVI_S32 ioctl (CVI_S32 fd, ACODEC_SET_DACR_MUTE, CVI_U32 *arg);
```

【Parameter】

Parameter	Description	Input/Output
fd	Audio codec device file description	Input
ACODEC_SET_DACR_MUTE	Mute number	Input
arg	Unsigned integer pointer	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: `acodec.h`

【Note】

The arg parameter ranges from 0 to 1, where 0 means unmute and 1 means mute.

【Example】


```
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

【Description】

Get Analog Gain of Left Channel MIC Input

【Syntax】

```
CVI_S32 ioctl (CVI_S32 fd, ACODEC_GET_GAIN_MICL, CVI_U32 *arg);
```

【Parameter】

Parameter	Description	Input/Output
fd	Audio codec device file description	Input
ACODEC_GET_GAIN_MICL	MICL number	Input
arg	Unsigned integer pointer	Output

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: `acodec.h`

【Note】

None.

【Example】

```
CVI_U32 gain_mic;
mute = 0;
if (ioctl(Acodec_Fd, ACODEC_GET_GAIN_MICL, &gain_mic))
{
printf("ioctl err!\n");
}
```

10.3.7.17 ACODEC_GET_GAIN_MICR

【Description】

Get Analog Gain of Right Channel MIC Input .

【Syntax】

```
CVI_S32 ioctl (CVI_S32 fd, ACODEC_GET_GAIN_MICR, CVI_U32 *arg);
```

【Parameter】

Parameter	Description	Input/Output
fd	Audio codec device file description	Input
ACODEC_GET_GAIN_MICR	MICR number	Input
arg	Unsigned integer pointer	Output

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: `acodec.h`

【Note】

None.

【Example】

```
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**【Description】**

Get the volume control of the left channel output.

【Syntax】

```
CVI_S32 ioctl (CVI_S32 fd, ACODEC_GET_DACL_VOL, struct cvi_vol_ctrl *arg);
```

【Parameter】

Parameter	Description	Input/Output
fd	Audio codec device file description	Input
ACODEC_GET_DACL_VOL	Vol number	Input
arg	cvi_vol_ctrl structure pointer	Output

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: `acodec.h`

【Note】

None.

【Example】

```

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**【Description】**

Gets the volume control of the right channel output.

【Syntax】

```
CVI_S32 ioctl (CVI_S32 fd, ACODEC_GET_DACR_VOL, struct cvi_vol_ctrl *arg);
```

【Parameter】

Parameter	Description	Input/Output
fd	Audio codec device file description	Input
ACODEC_GET_DACR_VOL	ioctl number	Input
arg	cvi_vol_ctrl structure pointer	Output

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: `acodec.h`

【Note】

None.

【Example】

```

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**【Description】**

Gets the volume control of the left channel input.

【Syntax】

```
CVI_S32 ioctl (CVI_S32 fd, ACODEC_GET_ADCL_VOL, struct cvi_vol_ctrl *arg);
```

【Parameter】

Parameter	Description	Input/Output
fd	Audio codec device file description	Input
ACODEC_GET_ADCL_VOL	ioctl number	Input
arg	cvi_vol_ctrl structure pointer	Output

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: `acodec.h`

【Note】

None.

【Example】

```

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**【Description】**

Gets the volume control of the right channel input.

【Syntax】

```
CVI_S32 ioctl (CVI_S32 fd, ACODEC_GET_ADCR_VOL, struct cvi_vol_ctrl *arg);
```

【Parameter】

Parameter	Description	Input/Output
fd	Audio codec device file description	Input
ACODEC_GET_ADCR_VOL	ioctl number	Input
arg	cvi_vol_ctrl structure pointer	Output

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: `acodec.h`

【Note】

None.

【Example】

```

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**【Description】**

Power-Off Control of Left Channel Output .

【Syntax】

```
CVI_S32 ioctl (CVI_S32 fd, ACODEC_SET_PD_DACL, CVI_U32 *arg);
```

【Parameter】

Parameter	Description	Input/Output
fd	Audio codec device file description	Input
ACODEC_SET_PD_DACL	ioctl number	Input
arg	Unsigned integer pointer	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: `acodec.h`

【Note】

This interface can be called when not using DACL to power down DACL.

The arg parameter ranges from 0 to 1.

0 means power up, and 1 means power down.

【Example】

```

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

【Description】

Power-Off Control of Right Channel Output .

【Syntax】

```
CVI_S32 ioctl (CVI_S32 fd, ACODEC_SET_PD_DACR, CVI_U32 *arg);
```

【Parameter】

Parameter	Description	Input/Output
fd	Audio codec device file description	Input
ACODEC_SET_PD_DACR	ioctl number	Input
arg	Unsigned integer pointer	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: `acodec.h`

【Note】

This interface can be called when not using DACL to power down DACL.

The arg parameter ranges from 0 to 1.

0 means power up, and 1 means power down.

【Example】

```
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

【Description】

Power-Off Control of Left Channel Input .

【Syntax】

```
CVI_S32 ioctl (CVI_S32 fd, ACODEC_SET_PD_ADCL, CVI_U32 *arg);
```

【Parameter】

Parameter	Description	Input/Output
fd	Audio codec device file description	Input
ACODEC_SET_PD_ADCL	ioctl number	Input
arg	Unsigned integer pointer	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: `acodec.h`

【Note】

This interface can be called when not using DACL to power down DACL.

The `arg` parameter ranges from 0 to 1.

0 means power up, and 1 means power down.

【Example】

```
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**【Description】**

Power-Off Control of Right Channel Input .

【Syntax】

```
CVI_S32 ioctl (CVI_S32 fd, ACODEC_SET_PD_ADCR, CVI_U32 *arg);
```

【Parameter】

Parameter	Description	Input/Output
<code>fd</code>	Audio codec device file description	Input
<code>ACODEC_SET_PD_ADCR</code>	ioctl number	Input
<code>arg</code>	Unsigned integer pointer	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: `acodec.h`

【Note】

This interface can be called when not using DACL to power down DACL.

The `arg` parameter ranges from 0 to 1.

0 means power up, and 1 means power down.

【Example】

```
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

【Description】

Power-Off Control of Left Channel LINEIN input.

【Syntax】

```
CVI_S32 ioctl (CVI_S32 fd, ACODEC_SET_PD_LINEINL, CVI_U32 *arg);
```

【Parameter】

Parameter	Description	Input/Output
fd	Audio codec device file description	Input
ACODEC_SET_PD_LINEINL	LINEINL number	Input
arg	Unsigned integer pointer	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: `acodec.h`

【Note】

This interface can be called when not using LINEIN to power down LINEIN.

The arg parameter ranges from 0 to 1.

0 means power up, and 1 means power down.

【Example】

```
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

【Description】

Power-Off Control of Right Channel LINEIN input.

【Syntax】

```
CVI_S32 ioctl (CVI_S32 fd, ACODEC_SET_PD_LINEINR, CVI_U32 *arg);
```

【Parameter】

Parameter	Description	Input/Output
fd	Audio codec device file description	Input
ACODEC_SET_PD_LINEINR	LINEINR Number	Input
arg	Unsigned integer pointer	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: `acodec.h`

【Note】

This interface can be called when not using LINEIN to power down LINEIN.

The arg parameter ranges from 0 to 1.

0 means power up, and 1 means power down.

【Example】

```
CVI_U32 pd_ctrl;
pd_ctrl = 0x1;
if (ioctl(Acodec_Fd, ACODEC_SET_PD_LINEINR, &pd_ctrl))
{
    printf("ioctl err!\n");
}
```

10.3.8 Resampling

The Resampler module provides independent resampling processing.

When a client needs to resample data in the upper layer, this module can be used.

The following are related API and detailed description.

- CVI_Resampler_Create*: Create a resampling module.
- CVI_Resampler_Process*: Resampling module data processing.
- CVI_Resampler_Destroy*: Destroy the resampling module.
- CVI_Resampler_GetMaxOutputNum*: Computing maximum output data for resampling.

10.3.8.1 CVI_Resampler_Create

【Description】

Create a resampling module.

【Syntax】

```
CVI_VOID *CVI_Resampler_Create(CVI_S32 s32Inrate, CVI_S32 s32Outrate, CVI_S32
↪s32Chans);
```

【Parameter】

Parameter	Description	Input/Output
s32Inrate	Input sampling rate. Value Range: 8000, 11025, 12000, 16000, 22050, 24000, 32000, 44100, 48000, 64000.	Input
s32Outrate	Output sampling rate. Value Range: 8000, 11025, 12000, 16000, 22050, 24000, 32000, 44100, 48000, 64000.	Input
s32Chans	Number of processing channels (currently Cvitek supports mono channel);	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: cvi_comm_aio.h, cvi_audio.h, cvi_resample_api.h
- Library files: libcvi_audio.a, libcvi_RES1.so

【Note】

The input sampling rate and output sampling rate should be different.

【Example】

None.

10.3.8.2 CVI_Resampler_Process

【Description】

Process one frame of resampled data.

【Syntax】

```
CVI_S32 CVI_Resampler_Process(CVI_VOID *inst, CVI_S16 *s16Inbuf, CVI_S32 s32Insamps,
↪CVI_S16 *s16Outbuf);
```

【Parameter】

Parameter	Description	Input/Output
inst	Resample module handle.	Input
s16Inbuf	Input data buf pointer.	Input
s32Insamps	Number of input sample points. Value Range: [0, 2048].	Input
s16Outbuf	Output data buf pointer	Output

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: `cvi_comm_aio.h`, `cvi_audio.h`, `cvi_resample_api.h`
- Library files: `libcvi_audio.a`, `libcvi_RES1.so`

【Note】

The maximum number of input sample points should be less than 2048.

【Example】

None.

10.3.8.3 CVI_Resampler_Destroy**【Description】**

Destroy a resampling module instance.

【Syntax】

```
CVI_VOID CVI_Resampler_Destroy(CVI_VOID *inst);
```

【Parameter】

Parameter	Description	Input/Output
inst	Resampling module handle.	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: `cvi_comm_aio.h`, `cvi_audio.h`, `cvi_resample_api.h`
- Library files: `libcvi_audio.a`, `libcvi_RES1.so`

【Note】

None.

【Example】

None.

10.3.8.4 CVI_Resampler_GetMaxOutputNum

【Description】

Get the maximum number of output sample points (per channel).

【Syntax】

```
CVI_S32 CVI_Resampler_GetMaxOutputNum(CVI_VOID *inst, CVI_S32 s32Insamps);
```

【Parameter】

Parameter	Description	Input/Output
inst	Resampling module handle.	Input
s32Insamps	Input sampling points for each channel.	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: `cvi_comm_aio.h`, `cvi_audio.h`, `cvi_resample_api.h`
- Library files: `libcvi_audio.a`, `libcvi_RES1.so`

【Note】

None.

【Example】

None.

10.4 Data Types

10.4.1 Audio Input / Output

The definition of data type and data structure related to audio input / output is as follows.

- *AI_DEV_MAX_NUM*: Define the maximum number of audio input devices.
- *AO_DEV_MAX_NUM*: Define the maximum number of audio output devices.
- *AI_MAX_CHN_NUM*: Define the maximum number of channels for an audio input device.
- *AO_MAX_CHN_NUM*: Define the maximum number of channels for an audio output device.
- *CVI_AUD_MAX_CHANNEL_NUM*: Define the maximum number of channels for an audio output device.
- *AI_TALKVQE_MASK_AEC*: Mask of Talk Vqe AEC function.
- *AI_TALKVQE_MASK_AGC*: Mask of Talk Vqe AGC function.
- *AI_TALKVQE_MASK_ANR*: Mask of Talk Vqe ANR function.
- *AI_RECORDVQE_MASK_AGC*: Mask of Record Vqe AGC function.
- *MAX_AUDIO_FILE_PATH_LEN*: Maximum length limitation for the path of the saved audio file.

- *MAX_AUDIO_FILE_NAME_LEN*: Maximum length limitation for the name of the saved audio file.
- *AUDIO_CLKSEL_E*: Define the audio clock source.
- *AUDIO_SAMPLE_RATE_E*: Define the audio sampling rate.
- *AUDIO_BIT_WIDTH_E*: Define the audio sampling accuracy.
- *AIO_MODE_E*: Define the audio input / output working mode.
- *AIO_I2STYPE_E*: Define I2S interface device type.
- *AUDIO_SOUND_MODE_E*: Define the audio channel mode.
- *AUDIO_MOD_PARAM_S*: Define the audio module parameter structure.
- *AIO_ATTR_S*: Define audio input / output device property structure.
- *AI_CHN_PARAM_S*: Define channel parameter structure.
- *AUDIO_FRAME_S*: Define audio frame data structure.
- *AEC_FRAME_S*: Define the information structure of echo cancellation reference frame.
- *AUDIO_AGC_CONFIG_S*: Define the audio AGC configuration information structure.
- *AI_AEC_CONFIG_S*: Define the audio echo cancellation configuration information structure.
- *AUDIO_ANR_CONFIG_S*: Define the information structure of audio voice noise reduction function.
- *VQE_WORKSTATE_E*: Define the working mode of voice quality enhancement.
- *VQE_RECORD_TYPE*: Define the recording type.
- *VQE_EQ_BAND_NUM*: Define the number of adjustable frequency bands for EQ functionality.
- *AI_TALKVQE_CONFIG_S*: Define the structure of audio input sound quality enhancement (Talk) configuration information.
- *AI_RECORDVQE_CONFIG_S*: Define the structure of audio input sound quality enhancement (Record) configuration information
- *AO_VQE_CONFIG_S*: Define the audio output voice quality enhancement configuration information structure.
- *AUDIO_STREAM_S*: Define audio stream structure.
- *AO_CHN_STATE_S*: Define Audio Output Channel Data Block Status Structure.
- *AUDIO_TRACK_MODE_E*: Audio device channel mode type.
- *AUDIO_FADE_RATE_E*: The audio device fade in and fade out rate type.
- *AUDIO_FADE_S*: The audio device fades in and out setting structure.
- *G726_BPS_E*: Defines the G.726 codec rate .
- *ADPCM_TYPE_E*: Define ADPCM codec type.
- *AUDIO_SAVE_FILE_INFO_S*: Definition of the configuration information structure for audio file saving function
- *AUDIO_FILE_STATUS_S*: Define the audio file save status structure.
- *VQE_MODULE_CONFIG_S*: Define the configuration information structure of voice quality enhancement and resampling module.
- *AUDIO_VQE_REGISTER_S*: Define the register structure of sound quality enhancement and resampling module.
- *ST_CVI_WAV_HEADER*: Defines the Wav header reference structure.

The following features are not currently supported.

- `AI_TALKVQE_MASK_HPF` : Mask of Talk Vqe HPF function.
- `AI_TALKVQE_MASK_EQ` : Mask of Talk Vqe EQ function.
- `AI_RECORDVQE_MASK_HPF` : Mask of Record Vqe HPF function.
- `AI_RECORDVQE_MASK_RNR` : Mask of Record Vqe RNR function.
- `AI_RECORDVQE_MASK_HDR` : Mask of Record Vqe HDR function.
- `AI_RECORDVQE_MASK_DRC` : Mask of Record Vqe DRC function.
- `AI_RECORDVQE_MASK_EQ` : Mask of Record Vqe EQ function.
- `AO_VQE_MASK_HPF` : Mask of AO Vqe HPF function.

10.4.1.1 AIO_MAX_NUM

【Description】

Define the maximum number of audio input / output devices.

【Syntax】

```
#define AIO_MAX_NUM      2
```

【Note】

Deprecated.

【Related Data Type and Interface】

None.

10.4.1.2 AI_DEV_MAX_NUM

【Description】

Define the maximum number of audio input devices.

【Syntax】

```
#define AI_DEV_MAX_NUM   1
```

【Note】

Deprecated.

【Related Data Type and Interface】

None.

10.4.1.3 AO_DEV_MAX_NUM

【Description】

Define the maximum number of audio output devices.

【Syntax】

```
#define AO_DEV_MAX_NUM   1
```

【Note】

Deprecated.

【Related Data Type and Interface】

None.

10.4.1.4 AI_MAX_CHN_NUM

【Description】

Define the maximum number of channels for an audio input device.

【Syntax】

```
#define AI_MAX_CHN_NUM    2
```

【Note】

Deprecated.

Please refer to CVI_AUD_MAX_CHANNEL_NUM.

【Related Data Type and Interface】

None.

10.4.1.5 AO_MAX_CHN_NUM

【Description】

Define the maximum number of channels for an audio output devices.

【Syntax】

```
#define AO_MAX_CHN_NUM    1
```

【Note】

Deprecated. Please refer to CVI_AUD_MAX_CHANNEL_NUM.

【Related Data Type and Interface】

None.

10.4.1.6 CVI_AUD_MAX_CHANNEL_NUM

【Description】

Define the maximum number of channels for an audio output device.

【Syntax】

```
#define CVI_AUD_MAX_CHANNEL_NUM    3
```

【Note】

Maximum number of audio channels can be set by AIO_ATTR_S stAttr.u32ChnCnt.

The value cannot exceed CVI_AUD_MAX_CHANNEL_NUM

【Related Data Type and Interface】

None.

10.4.1.7 AI_TALKVQE_MASK_AEC

【Description】

Mask of Talk Vqe AEC function.

【Syntax】

```
#define AI_TALKVQE_MASK_AEC 0x3
```

【Note】

None.

【Related Data Type and Interface】

None.

10.4.1.8 AI_TALKVQE_MASK_AGC

【Description】

Define the Mask of Talk Vqe AGC function.

【Syntax】

```
#define AI_TALKVQE_MASK_AGC 0x8
```

【Note】

None.

【Related Data Type and Interface】

Assign values to the member of structure u32OpenMask of AI_TALKVQE_CONFIG_S to indicate that AGC function is turned on.

For example, u32OpenMask = AI_TALKVQE_MASK_AEC | AI_TALKVQE_MASK_AGC; it indicates that AEC and AGC functions are turned on.

10.4.1.9 AI_TALKVQE_MASK_ANR

【Description】

Define the Mask of talk Vqe ANR function.

【Syntax】

```
#define AI_TALKVQE_MASK_ANR 0x4
```

【Note】

None.

【Related Data Type and Interface】

Assign values to the member of structure u32OpenMask of AI_TALKVQE_CONFIG_S to indicate that ANR function is turned on.

For example, u32OpenMask = AI_TALKVQE_MASK_AEC | AI_TALKVQE_MASK_ANR; it indicates that AEC and ANR functions are turned on.

10.4.1.10 AI_RECORDVQE_MASK_AGC

【Description】

Define the Mask of record Vqe AGC function.

【Syntax】

```
#define AI_RECORDVQE_MASK_AGC 0x20
```

【Note】

None.

【Related Data Type and Interface】

Assign values to the member of structure u32OpenMask of AI_RECORDVQE_CONFIG_S to indicate that AGC function is turned on.

For example, u32OpenMask = AI_RECORDVQE_MASK_HPF AI_RECORDVQE_MASK_AGC; it indicates that HPF and AGC functions are turned on.

10.4.1.11 MAX_AUDIO_FILE_PATH_LEN

【Description】

Define the maximum length limitation for the path of the saved audio file.

【Syntax】

```
#define MAX_AUDIO_FILE_PATH_LEN 256
```

【Note】

None.

【Related Data Type and Interface】

- AUDIO_SAVE_FILE_INFO_S

10.4.1.12 MAX_AUDIO_FILE_NAME_LEN

【Description】

Define the maximum length limitation for the name of the saved audio file.

【Syntax】

```
#define MAX_AUDIO_FILE_NAME_LEN 256
```

【Note】

None.

【Related Data Type and Interface】

- AUDIO_SAVE_FILE_INFO_S

10.4.1.13 AUDIO_CLKSEL_E

【Description】

Define the audio clock source.

【Syntax】

```
typedef enum _AUDIO_CLKSEL_E
{ AUDIO_CLKSEL_BASE      = 0,  /*<Audio base clk. */
  AUDIO_CLKSEL_SPARE,      /*<Audio spare clk. */
  AUDIO_CLKSEL_BUTT,
} AUDIO_CLKSEL_E;
```

【Member】

None.

【Note】

Cvitek users do not need to set the clock at this time.

【Related Data Type and Interface】

- [AUDIO_MOD_PARAM_S](#)

10.4.1.14 AUDIO_SAMPLE_RATE_E

【Description】

Define the audio sampling rate.

【Syntax】

```
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;
```

【Member】

Member	Description
AU-DIO_SAMPLE_RATE_8000	8kHz sample rate
AU-DIO_SAMPLE_RATE_11025	11.025kHz sample rate
AU-DIO_SAMPLE_RATE_16000	16kHz sample rate
AU-DIO_SAMPLE_RATE_22050	22.050kHz sample rate
AU-DIO_SAMPLE_RATE_24000	24kHz sample rate
AU-DIO_SAMPLE_RATE_32000	32kHz sample rate
AU-DIO_SAMPLE_RATE_44100	44.1kHz sample rate
AU-DIO_SAMPLE_RATE_48000	48kHz sample rate
AU-DIO_SAMPLE_RATE_64000	64kHz sample rate

【Note】**【Related Data Type and Interface】**

- [AIO_ATTR_S](#)

10.4.1.15 AUDIO_BIT_WIDTH_E**【Description】**

Define the audio sampling accuracy.

【Syntax】

```
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;
```

【Member】

Member	Description
AUDIO_BIT_WIDTH_8	the sampling accuracy is 8 bits
AUDIO_BIT_WIDTH_16	the sampling accuracy is 16 bits
AUDIO_BIT_WIDTH_24	the sampling accuracy is 24 bits
AUDIO_BIT_WIDTH_32	the sampling accuracy is 32 bits

【Note】

None.

【Related Data Type and Interface】

- [AIO_ATTR_S](#)

10.4.1.16 AIO_MODE_E

【Description】

Define the audio input / output working mode.

【Syntax】

```
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;
```

【Member】

Member	Description
AIO_MODE_I2S_MASTER	I2S master mode
AIO_MODE_I2S_SLAVE	I2S slave mode
AIO_MODE_PCM_SLAVE_STD	PCM slave standard mode
AIO_MODE_PCM_SLAVE_NSTD	PCM slave non-standard mode
AIO_MODE_PCM_MASTER_STD	PCM master standard mode
AIO_MODE_PCM_MASTER_NSTD	PCM master non-standard mode

【Note】

Built-in Cvitek only supports I2S master mode.

【Related Data Type and Interface】

- [AIO_ATTR_S](#)

10.4.1.17 AIO_I2STYPE_E

【Description】

Define I2S interface device type.

【Syntax】

```
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;
```

【Member】

Member	Description
AIO_I2STYPE_INNERCODEC	I2S connect inner audio CODEC
AIO_I2STYPE_INNERHDMI	I2S connect Inner HDMI
AIO_I2STYPE_EXTERN	I2S connect extern hardware

【Note】

Cvitek only supports AIO_I2STYPE_INNERCODEC connecting to inner audio CODEC.

【Related Data Type and Interface】

- [AIO_ATTR_S](#)

10.4.1.18 AUDIO_SOUND_MODE_E**【Description】**

Define the audio channel mode.

【Syntax】

```
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;
```

【Member】

Member	Description
AUDIO_SOUND_MODE_MONO	Mono
AUDIO_SOUND_MODE_STEREO	Stereo

【Note】

The left channel corresponds to channel 0 and the right channel corresponds to channel 1.

For AI, mono input is from the left channel by default.

If it needs to be configured as the right channel input,

- Turn on the right channel only and process.
- Open the left and right channels, process according to the left channel, and use `CVI_AI_SetTrackMode` to configure AI channel mode to `AUDIO_TRACK_EXCHANGE` .

For AO, mono input is from the left channel by default.

If it needs to be configured as the right channel input, you can consider two methods.

- Turn on the right channel only and process.
- Turn on the left and right channels, process according to the left channel, and use `CVI_AO_SetTrackMode` to configure AO channel mode to `AUDIO_TRACK_EXCHANGE` .

For stereo mode, only the left channel (that is, the channel whose number is less than half of `u32ChnCnt` in the device attribute) should be operated, and the SDK will automatically operate the right channel.

【Related Data Type and Interface】

- [AIO_ATTR_S](#)

10.4.1.19 AUDIO_MOD_PARAM_S

【Description】

Define the audio module parameter structure.

【Syntax】

```
typedef struct cviAUDIO_MOD_PARAM_S {
    AUDIO_CLKSEL_E enClkSel;
} AUDIO_MOD_PARAM_S;
```

【Member】

enClkSel audio clock source selection. Please see AUDIO_CLKSEL_E.

【Note】

Cvitek does not need special setting for CLK.

【Related Data Type and Interface】

None.

10.4.1.20 AIO_ATTR_S

【Description】

Define audio input / output device property structure.

【Syntax】

```
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;
```

【Member】

Member	Description
enSamplerate	Audio sample rate (this parameter does not work in slave mode); Static properties.
enBitwidth	Audio sampling accuracy (in slave mode, this parameter must match the sampling accuracy of audio AD/DA); Static properties.
enWorkmode	Audio I / O working mode; Static properties.
enSoundmode	Audio channel mode; Static properties.
u32EXFlag	Value range: {0, 1, 2}. 0:does not extend. 1: It is expanded to 16 bits, and the 8-bit to 16bit extension flag (only valid for AI sampling accuracy of 8bit). 2: The 24 bits are cropped to 16 bits, which may be used in the external codec scenario. Static property, keep parameters, generally set to 1.
u32FrmNum	Number of block frames.
u32PtNumPerFrm	Number of sample points per frame. Value range: G711, G726, ADPCM_DVI4 is 160, 320, 480;
u32ChnCnt	Number of channels supported. Values: 1, 2, 4, 8, 16 (Input and output supports up to 2 channels respectively).
enI2sType	Configure I2S interface device type; Cvitek only supports master mode.
AI Value	AIO_I2STYPE_INNERCODEC, AIO_I2STYPE_EXTERN
AO Value	AIO_I2STYPE_INNERCODEC, AIO_I2STYPE_EXTERN, AIO_I2STYPE_INNERHDMI.

【Note】

The number of sampling points per frame u32PtNumPerFrm and sampling rate enSamplerate determine the frequency of hardware interrupt.

If the frequency is too high, it will affect the performance of the system and interact with other services.

It is suggested that the values of these two parameters satisfy the formula: (u32PtNumPerFrm * 1000) / enSamplerate > = 10.

For example, when the sampling rate is 16000Hz, it is recommended to set the number of sampling points greater than or equal to 160.

【Related Data Type and Interface】

- CVI_AI_SetPubAttr
- CVI_AO_SetPubAttr

10.4.1.21 AI_CHN_PARAM_S**【Description】**

Define channel parameter structure.

【Syntax】

```
typedef struct _AI_CHN_PARAM_S {
    CVI_U32 u32UsrFrmDepth;
} AI_CHN_PARAM_S;
```

【Member】

u32UsrFrmDepth: Audio frame block depth.

【Note】

None.

【Related Data Type and Interface】

None.

10.4.1.22 AUDIO_FRAME_S**【Description】**

Define audio frame data structure.

【Syntax】

```
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;
```

【Member】

Member	Description
enBitwidth	Audio sampling accuracy.
enSoundmode	Audio channel mode.
u64VirAddr [2]	Audio frame data virtual address.
u64PhyAddr[2]	Audio frame data physical address. Not supported at present.
u64TimeStamp	Audio frame timestamp. The unit is s.
u32Seq	Audio frame sequence.
u32Len	Audio frame length: the total sampling amount of a single channel. samples as the unit. 1 sample = 2 bytes. Ex. AIO_ATTR_S parameters setting: u32FrmNum = 320, u32ChnCnt = 2. And <i>u32Len = 320(samples/channel)</i> . u64VirAddr [0] buffer includes the number of bytes should be (u32Len x u32ChnCnt x 2).
u32PoolId[2]	Audio frame block pool ID.

【Note】

u32Len (audio frame length) refers to the data length of a single channel.

u64VirAddr [0], the length is in bytes : (u32Len x bytes_per_sample);

The default channel mode for mono audio is left channel, and the data is arranged as [Left, Left, Left, Left, Left, ...].

Stereo data is arranged as [L, R, L, R, L, R, ...] where L stands for the left channel and R stands for the right channel.

(Note: the left represents a single sample in the left channel, and the right represents a single sample in the left channel.)

u64VirAddr [1]. There is no storage data, which can be customized.

【Related Data Type and Interface】

None.

10.4.1.23 AEC_FRAME_S

【Description】

Define the information structure of echo cancellation reference

【Syntax】

```
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;
```

【Member】

Member	Description
stRefFrame	Echo cancellation reference frame structure.
bValid	Reference frame valid flag. Value range: CVI_TRUE:the reference frame is valid. CVI_FALSE: if the reference frame is invalid, it cannot be used for echo cancellation.
bSysBind	Whether AI and AENC are system bound.

【Note】

None.

【Related Data Type and Interface】

None.

10.4.1.24 AUDIO_AGC_CONFIG_S

【Description】

Define the audio AGC configuration information structure.

【Syntax】

```
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;
```

【Member】

Member	Description
para_agc_max_gain	The maximum gain at which a signal can be amplified.
para_agc_target_high:	AGC will reach the “Target High” level.
para_agc_target_low	AGC will reach the “Target Low” level.
para_agc_vad_enable:	Speech-activated AGC uses speech VAD from NR to avoid amplifying background noise. It is recommended to turn on this function in high SNR environment, and it is better to turn off this function in medium / low SNR environment for better voice quality.

【Note】
【Related Data Type and Interface】

- `AI_VQE_CONFIG_S`
- `AO_VQE_CONFIG_S`

10.4.1.25 AI_AEC_CONFIG_S

【Description】

Define the audio echo cancellation configuration information structure.

【Syntax】

```
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;
```

【Member】

Member	Description
<code>para_aec_filter_len</code>	The length of the adaptive filter
<code>para_aes_std_thrd</code>	Residual Echo Suppression Threshold
<code>para_aes_supp_coeff</code>	Residual Echo Suppression Level

【Note】

When user mode is on, other parameters will take effect;

Otherwise, it is configured according to the default value of the corresponding working mode en-Workstate in according to `AI_VQE_CONFIG_S` / `AI_TALKVQE_CONFIG` The working mode in the `AI_VQE_CONFIG_S` / `AI_TALKVQE_CONFIG_S`.

When configuring parameters, correctness checks for advanced parameters are only performed when the user mode is enabled. Only when the advanced parameters are correct can the configuration be successful.

【Related Data Type and Interface】

- `AI_VQE_CONFIG_S`

10.4.1.26 AUDIO_ANR_CONFIG_S

【Description】

Define the information structure of audio voice noise reduction function.

【Syntax】

```
typedef struct _AUDIO_ANR_CONFIG_S {
    CVI_S8 para_nr_snr_coeff;
    CVI_S8 para_nr_noise_coeff;
} AUDIO_ANR_CONFIG_S;
```

【Member】

Member	Description
para_nr_snr_coeff	Signal-to-Noise Ratio (SNR) tracking coefficient. If it is set to a larger value, NR will have a higher noise reduction ability, but the speech signal may be more easily distorted; If a smaller value is selected, NR will suppress less noise signal, but it will have better speech quality performance.
para_nr_noise_coeff	Noise tracking coefficient. This parameter determines the tracking speed of stationary noise[0 - 14] 0: slowest noise tracking speed 14: fastest noise tracking speed

【Note】

None.

【Related Data Type and Interface】

- `AI_VQE_CONFIG_S`
- `AO_VQE_CONFIG_S`

10.4.1.27 AUDIO_DELAY_CONFIG_S**【Description】**

Definition of Audio Signal Delay Structure.

【Syntax】

```
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;
```

【Member】

Member	Description
para_aec_init_filter_len	The length of the adaptive filter
para_dg_target	Digital Gain. Value range [1-12]. This feature helps reduce residual echo and residual stationary noise.
para_delay_sample	Used to delay the reference signal. Value range: [1-3000] It enables AEC/AES to accelerate convergence at the beginning of the echo.

【Note】

None.

【Related Data Type and Interface】

- `AI_TALKVQE_CONFIG_S`

10.4.1.28 AO_VQE_CONFIG_S

【Description】

Define the audio output voice quality enhancement configuration information structure.

【Syntax】

```
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;
```

【Member】

Member	Description
u32OpenMask	AO Mask value enabled for each Vqe function.
s32WorkSampleRate	Operating sampling frequency. This parameter is the working sampling rate of the internal functional algorithm. Value range: 8KHz/16KHz/48KHz. The default value is 8KHz. (48KHz for Hpf only)
stAgcCfg	Automatic gain control configuration information.
stEqCfg	Voice signal equalization processing configuration.

【Note】

【Related Data Type and Interface】

None.

10.4.1.29 VQE_WORKSTATE_E

【Description】

Define the working mode of voice quality enhancement.

【Syntax】

```
typedef enum _VQE_WORKSTATE_E {
    VQE_WORKSTATE_COMMON    = 0,
    VQE_WORKSTATE_MUSIC     = 1,
    VQE_WORKSTATE_NOISY     = 2
} VQE_WORKSTATE_E;
```

【Member】

Member	Description
VQE_WORKSTATE_COMMON	Common mode.
VQE_WORKSTATE_MUSIC	Music mode.
VQE_WORKSTATE_NOISY	Noise mode.

【Note】

None.

【Related Data Type and Interface】

- AI_VQE_CONFIG_S

- [AO_VQE_CONFIG_S](#)

10.4.1.30 VQE_RECORD_TYPE

【Description】

Define the recording type.

【Syntax】

```
typedef enum _VQE_RECORD_TYPE {
    VQE_RECORD_NORMAL      = 0,
    VQE_RECORD_BUTT, }
VQE_RECORD_TYPE;
```

【Member】

VQE_RECORD_NORMAL: Standard type.

【Note】

Cvitek only supports talk VQE, and record VQE is not used until it is customized.

【Related Data Type and Interface】

- [AI_RECORDVQE_CONFIG_S](#)

10.4.1.31 AI_TALKVQE_CONFIG_S

【Description】

Define the structure of audio input sound quality enhancement (Talk) configuration information.

【Syntax】

```
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;
    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;
```

【Member】

Member	Description
para_client_config	Client parameter configuration.
u32OpenMask	Mask value enabled for each Talk Vqe function.
s32WorkSampleRate	Operating sampling frequency. This parameter is the working sampling rate of the internal functional algorithm. Value range: 8KHz/16KHz/48KHz. The default value is 8KHz. (48KHz for Hpf only)
stAecCfg	Configuration information related to echo cancellation function.
stAnrCfg	Configuration information related to voice noise reduction function.
stAgcCfg	Automatic gain control configuration information.
stAecDelayCfg	Configuration information related to audio signal delay.
s32RevMask	Customized masking threshold setting.
para_notch_freq	Customized frequency elimination.
customize	Customization parameter selection.

【Note】

Cvitek VQE supports only AGC/ANR/AEC.

For example, if RNR/EQ data is set, it will not have an effect

【Related Data Type and Interface】

None.

10.4.1.32 AI_RECORDVQE_CONFIG_S**【Description】**

Define the structure of audio input sound quality enhancement (Record) configuration information.

【Syntax】

```
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;
    AUDIO_AGC_CONFIG_S  stAgcCfg;
} AI_RECORDVQE_CONFIG_S;
```

【Member】

Member	Description
u32OpenMask	Mask value enabled for each Talk Vqe function.
s32WorkSampleRate	Operating sampling frequency. This parameter is the working sampling rate of the internal functional algorithm. Value range: 8KHz/16KHz/48KHz. The default value is 8KHz. (48KHz for Hpf only)
stAgcCfg	Automatic gain control configuration information.
enWorkstate	Working mode
s32InChNum	Number of input channels processed by VQE. Value range: [1, 2].
s32OutChNum	Number of output channels processed by VQE. Value range: [1, 2].
enRecordType	Record type

【Note】

Cvitek VQE supports only AGC/ANR/AEC.

For example, if RNR/EQ data is set, it will not have an effect

【Related Data Type and Interface】

None.

10.4.1.33 AUDIO_STREAM_S**【Description】**

Define audio stream structure.

【Syntax】

```
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;
```

【Member】

Member	Description
pStream	The virtual address of stream
u32PhyAddr	the physics address of stream
u32Len	Audio stream length. AUDIO_STREAM_S structure body, in byte.
u64TimeStamp	Audio stream timestamp
u32Seq	Audio stream sequence

【Note】

None.

【Related Data Type and Interface】

- CVI_AENC_GetStream

10.4.1.34 AO_CHN_STATE_S**【Description】**

Define Audio Output Channel Data Block Status Structure.

【Syntax】

```
typedef struct hiAO_CHN_STATE_S {
    CVI_U32 u32ChnTotalNum;
    CVI_U32 u32ChnFreeNum;
    CVI_U32 u32ChnBusyNum;
} AO_CHN_STATE_S;
```

【Member】

Member	Description
u32ChnTotalNum	Total number of blocks in output channel.
u32ChnFreeNum	Available free blocks
u32ChnBusyNum	Occupied blocks

【Note】

None.

【Related Data Type and Interface】

- CVI_AO_QueryChnStat

10.4.1.35 AUDIO_TRACK_MODE_E**【Description】**

Audio device channel mode type.

【Syntax】

```
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;
```

【Member】

Member	Description
AUDIO_TRACK_NORMAL	Normal mode, no processing
AUDIO_TRACK_BOTH_LEFT	Both channels are left
AUDIO_TRACK_BOTH_RIGHT	Both channels are right channel
AUDIO_TRACK_EXCHANGE	Data exchange between left and right channels, left channel is right channel sound, right channel is left channel sound
AUDIO_TRACK_MIX	The output of left and right channels is the aggregation of left and right channels (mixed)
AUDIO_TRACK_LEFT_MUTE	The left channel is mute, and the right channel plays the original right channel sound
AUDIO_TRACK_RIGHT_MUTE	The right channel is mute, and the left channel plays the original left channel sound
AUDIO_TRACK_BOTH_MUTE	Both left and right channels are mute

【Note】

None.

【Related Data Type and Interface】

- CVI_AI_SetTrackMode

- CVI_AO_SetTrackMode

10.4.1.36 AUDIO_FADE_RATE_E

【Description】

The audio device fade in and fade out rate type.

【Syntax】

```
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;
```

【Member】

Member	Description
AUDIO_FADE_RATE_NONE	No delay between increasing or decreasing the volume
AUDIO_FADE_RATE_10	Volume increments or decrements for every 10ms step.
AUDIO_FADE_RATE_20	Volume increments or decrements for every 20ms step.
AUDIO_FADE_RATE_30	Volume increments or decrements for every 30ms step.
AUDIO_FADE_RATE_50	Volume increments or decrements for every 50ms step.
AUDIO_FADE_RATE_100	Volume increments or decrements for every 100ms step.
AUDIO_FADE_RATE_200	Volume increments or decrements for every 200ms step.

【Note】

When Cvitek uses AUDIO_FADE_RATE_E parameter, please confirm that bFade in AUDIO_FADE_S has been set to CVI_TRUE.

Fade in or fade out will be set gradually according to the set AUDIO_FADE_RATE time delay based on the current volume value until fade in to unmute or fade out to mute.

【Related Data Type and Interface】

None.

10.4.1.37 AUDIO_FADE_S

【Description】

The audio device fades in and out setting structure.

【Syntax】

```
typedef struct hiAUDIO_FADE_S {
    CVI_BOOL      bFade;
    AUDIO_FADE_RATE_E enFadeInRate;
    AUDIO_FADE_RATE_E enFadeOutRate;
} AUDIO_FADE_S;
```

【Member】

Member	Description
bFade	Whether to turn on the fade in and fade out function. CVI_TRUE: Turn on the fading function. CVI_FALSE: Turn off the fading function.
enFadeInRate	Audio output device volume fade-in speed.
enFadeOutRate	Audio output device volume fade-in speed.

【Note】

Cvitek please confirm that the bFade in AUDIO_FADE_S has been set to CVI_TRUE, and the setting of enFadeInRate/enFadeOutRate value will have effect.

【Related Data Type and Interface】

- [CVI_AO_SetMute](#)

10.4.1.38 G726_BPS_E**【Description】**

Defines the G.726 codec rate

【Syntax】

```
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 ... */
    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;
```

【Member】

Member	Description
G726_16K	16kbps G.726。
G726_24K	24kbps G. 726。
G726_32K	32kbps G.726。
G726_40K	40kbps G.726。
MEDIA_G726_16K	16kbps for ASF。
MEDIA_G726_24K	G726 24kbps for ASF。
MEDIA_G726_32K	G726 32kbps for ASF。
MEDIA_G726_40K	G726 40kbps for ASF。

【Note】

None.

【Related Data Type and Interface】

None.

10.4.1.39 ADPCM_TYPE_E

【Description】

Define ADPCM codec type.

【Syntax】

```
typedef enum hiADPCM_TYPE_E {
    ADPCM_TYPE_DVI4 = 0,
    ADPCM_TYPE_IMA,
    ADPCM_TYPE_ORG_DVI4,
    ADPCM_TYPE_BUTT,
} ADPCM_TYPE_E;
```

【Member】

Member	Description
ADPCM_TYPE_DVI4	32kbit/s ADPCM(DVI4)。
ADPCM_TYPE_IMA	32kbit/s ADPCM(IMA)。
ADPCM_TYPE_ORG_DVI4	32kbit/s ADPCM(ORG_DVI4)。

【Note】

None.

【Related Data Type and Interface】

None.

10.4.1.40 ST_CVI_WAV_HEADER

【Description】

Defines the Wav header reference structure.

【Syntax】

```
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
```

(continues on next page)

(continued from previous page)

```

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;

```

【Member】

This structure is for users' reference only, and there is no CVI API as the input variable.

【Note】

None.

【Related Data Type and Interface】

None.

10.4.1.41 AUDIO_FILE_STATUS_S**【Description】**

Define the audio file save status structure.

【Syntax】

```

typedef struct hiAUDIO_FILE_STATUS_S {
    CVI_BOOL    bSaving;
} AUDIO_FILE_STATUS_S;

```

【Member】

Member	Description
bSaving	Checking the file storage status. CVI_TRUE: In the state of saving files;; CVI_FALSE: Not in file storage status.

【Note】

None.

【Related Data Type and Interface】

None.

10.4.1.42 VQE_MODULE_CONFIG_S**【Description】**

Define the configuration information structure of voice quality enhancement and resampling module.

【Syntax】

```

typedef struct _VQE_MODULE_CONFIG_S {
    CVI_VOID *pHandle;
} VQE_MODULE_CONFIG_S;

```

【Member】

Member	Description
pHandle	Register handle.

【Note】

The registration handle of each sound quality enhancement and resampling module can be obtained by calling the handle acquisition interface.

【Related Data Type and Interface】

None.

10.4.1.43 AUDIO_VQE_REGISTER_S**【Description】**

Define the register structure of sound quality enhancement and resampling module.

【Syntax】

```
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;
```

【Member】

Member	Description
pHandle	Register handle.

【Note】

Currently only supports Talk VQE after audio uplink voice recording.

Other VQE are not supported.

【Related Data Type and Interface】

None.

10.4.2 Audio Encoding

The data types and data structures related to audio encoding are defined as follows:

- *AENC_MAX_CHN_NUM*: Define the maximum number of audio coding channels.
- *AENC_ATTR_G711_S*: Define G.711 encoding protocol attribute structure.
- *AENC_ATTR_G726_S*: Define G.726 encoding protocol attribute structure.
- *AENC_ATTR_ADPCM_S*: Define ADPCM encoding protocol attribute structure.
- *AENC_ATTR_LPCM_S*: Define LPCM encoding protocol attribute structure.
- *AENC_CHN_ATTR_S*: Defines the audio encoding channel attribute structure.
- *AAC_AENC_ENCODER_S*: Defines the encoder attribute structure.

10.4.2.1 AENC_MAX_CHN_NUM

【Description】

Define the maximum number of audio coding channels.

【Syntax】

```
#define AENC_MAX_CHN_NUM 1
```

【Note】

ADEC_MAX_CHN_NUM

【Related Data Type and Interface】

None.

10.4.2.2 AENC_ATTR_G711_S

【Description】

Define G.711 encoding protocol attribute structure.

【Syntax】

```
typedef struct hiAENC_ATTR_G711_S
{
    CVI_U32 resv;
}AENC_ATTR_G711_S;
```

【Member】

Member	Description
resv	not used

【Note】

None.

【Related Data Type and Interface】

None.

10.4.2.3 AENC_ATTR_G726_S

【Description】

Define G.726 encoding protocol attribute structure.

【Syntax】

```
typedef struct _AENC_ATTR_G726_S {
    G726_BPS_E enG726bps;
}AENC_ATTR_G726_S;
```

【Member】

Member	Description
enG726bps	G.726 protocol bitrate

【Note】

None.

【Related Data Type and Interface】

G726_BPS_E

10.4.2.4 AENC_ATTR_ADPCM_S

【Description】

Define ADPCM encoding protocol attribute structure.

【Syntax】

```
typedef struct _AENC_ATTR_ADPCM_S {
    ADPCM_TYPE_E enADPCMType;
}AENC_ATTR_ADPCM_S;
```

【Member】

Member	Description
enADPCMType	ADPCM type

【Note】

None.

【Related Data Type and Interface】

ADPCM_TYPE_E

10.4.2.5 AENC_ATTR_LPCM_S

【Description】

Define LPCM encoding protocol attribute structure.

【Syntax】

```
typedef struct _AENC_ATTR_LPCM_S {
    CVI_U32 resv;           /*reserve item*/
}AENC_ATTR_LPCM_S;
```

【Member】

The internal variables of this structure are not used.

【Note】

None.

【Related Data Type and Interface】

None.

10.4.2.6 AENC_CHN_ATTR_S**【Description】**

Defines the audio encoding channel attribute structure. The definition of this structure varies slightly on different chip platforms.

【Syntax】

```
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;
```

【Member】

Member	Description
enType	Audio coding protocol type Static property.
u32PtNumPerFrm	Frame length of audio encoding protocol
u32BufSize	Size of audio encoding block.
pValue	Specific protocol attribute pointer.
bFileDbgMode	Whether in the file storage status

【Note】

None.

【Related Data Type and Interface】

None.

10.4.2.7 AAC_AENC_ENCODER_S**【Description】**

Defines AAC encoder attribute structure.

【Syntax】

```
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 *
```

(continues on next page)

(continued from previous page)

```

    pu8Outbuf,

    CVI_S32 s32InputSizeBytes, CVI_U32 *pu32OutLen);
    CVI_S32 (*pfnCloseEncoder)(CVI_VOID *pEncoder);
} AAC_AENC_ENCODER_S;

```

【Member】

This structure is only used by AAC external LIB link.

This version of SDK is only defined but not supported.

【Note】

This structure is only used by AAC external LIB link.

This version of SDK is only defined but not supported.

If AAC is required, please refer to middleware/sample/audio/aac_sample.

【Related Data Type and Interface】

None.

10.4.3 Audio Decoding

Data types and data structures related to audio decoding are defined as follows:

- *MAX_AUDIO_FRAME_NUM*: Define the maximum number of audio decoding block frames.
- *ADEC_MAX_CHN_NUM*: Define the maximum number of audio decoding channels.
- *ADEC_ATTR_G711_S*: Define G.711 decoding protocol attribute structure.
- *ADEC_ATTR_G726_S*: Define G.726 decoding protocol attribute structure.
- *ADEC_ATTR_ADPCM_S*: Define ADPCM decoding protocol attribute structure.
- *ADEC_ATTR_LPCM_S*: Define LPCM decoding protocol attribute structure.
- *ADEC_MODE_E*: Define the decoding method.
- *ADEC_CHN_ATTR_S*: Define the decoding channel attribute structure.
- *ADEC_CHN_STATE_S*: Define the Audio Decoding Channel Data Buffer Status Structure.
- *ADEC_DECODER_S*: Define the decoder attribute structure.

10.4.3.1 MAX_AUDIO_FRAME_NUM

【Description】

Define the maximum number of audio decoding block frames.

【Syntax】

```
#define MAX_AUDIO_FRAME_NUM    300
```

【Note】

Currently the number of audio internal cache frames is determined by the SDK, so this setting is not open to users and will not have any effect.

【Related Data Type and Interface】

None.

10.4.3.2 ADEC_MAX_CHN_NUM

【Description】

Define the maximum number of audio decoding channels.

【Syntax】

```
#define ADEC_MAX_CHN_NUM      1
```

【Note】

Currently only supports single channel encoding and decoding.

【Related Data Type and Interface】

None.

10.4.3.3 ADEC_ATTR_G711_S

【Description】

Define G.711 decoding protocol attribute structure.

【Syntax】

```
typedef struct _ADEC_ATTR_G711_S {
    CVI_U32 resv;
}ADEC_ATTR_G711_S;
```

【Member】

The variables in this structure are not used in Cvitek chip

【Note】

None.

【Related Data Type and Interface】

None.

10.4.3.4 ADEC_ATTR_G726_S

【Description】

Define G.726 decoding protocol attribute structure.

【Syntax】

```
typedef struct _ADEC_ATTR_G726_S {
    G726_BPS_E enG726bps;
}ADEC_ATTR_G726_S;
```

【Member】

Member	Description
enG726bps	G.726 protocol bitrate

【Note】

None.

【Related Data Type and Interface】

G726_BPS_E

10.4.3.5 ADEC_ATTR_ADPCM_S

【Description】

Define ADPCM decoding protocol attribute structure .

【Syntax】

```
typedef struct _ADEC_ATTR_ADPCM_S {
    ADPCM_TYPE_E enADPCMType;
}ADEC_ATTR_ADPCM_S;
```

【Member】

Member	Description
enADPCMType	ADPCM type

【Note】

None.

【Related Data Type and Interface】

- [ADPCM_TYPE_E](#)

10.4.3.6 ADEC_ATTR_LPCM_S

【Description】

Define LPCM decoding protocol attribute structure.

【Syntax】

```
typedef struct _ADEC_ATTR_LPCM_S {
    CVI_U32 resv;
}ADEC_ATTR_LPCM_S;
```

【Member】

resv is to be extended.

【Note】

None.

【Related Data Type and Interface】

None.

10.4.3.7 ADEC_MODE_E

【Description】

Define the decoding method.

【Syntax】

```
typedef enum _ADEC_MODE_E {
    ADEC_MODE_PACK = 0,
    ADEC_MODE_STREAM ,
    ADEC_MODE_BUTT
}ADEC_MODE_E;
```

【Member】

Member	Description
ADEC_MODE_PACK	Decode in Pack mode.
ADEC_MODE_STREAM	Decode in stream mode.

【Note】

Pack mode is used when the user confirms that the current stream packet is the result of one frame data encoding, the decoder will decode it directly.

If it is not one frame, the decoder will fail.

The efficiency of this mode is relatively high.

If the code stream packet encoded by AENC module is not damaged, this mode can be used for decoding.

Stream mode is used when the user can't confirm whether the current code stream packet is one frame of data, and the decoder needs to determine and block the code stream.

This mode is inefficient, and is generally used in the case of reading file code stream decoding or uncertain code stream packet boundary.

Of course, due to the fixed length of speech coding stream, it is easy to determine the frame boundary in the stream, so it is recommended to use pack decoding.

Cvitek only supports pack mode.

In the case of uncertain stream boundary, Cvitek will make decoding errors due to the misalignment of frame number.

【Related Data Type and Interface】

None.

10.4.3.8 ADEC_CHN_ATTR_S**【Description】**

Define the decoding channel attribute structure.

【Syntax】

```
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;
```

【Member】

Member	Description
enType	Audio decoding protocol type, static property.
u32BufSize	Audio decoding block buffer size. At present, the number of audio internal cache frames is determined by the SDK, so this setting is not open to users and will not have any effect.
enMode	Decoding mode, static property. Mode only supports ADEC_MODE_PACK mode and cannot be detected automatically by setting ADEC_MODE_STREAM.
pValue	Specific protocol attribute pointer.
bFileDbgMode	Whether to turn on save file mode. Please pay attention to the cvi_sample_audio.c sample code in the SDK. This value is true by default for debugging. Users should set it to false in actual use to avoid using performance or memory due to disk saving.
When the user only uses the ADECmodule but not the AO module, the user needs to inform the ADECmodule of the relevant parameter characteristics through the following variable settings.	
s32BytesPerSample	Bytes used for unit sampling. (bit width SL16, 16bits = 2 bytes, at this point, the bytes used for unit sampling should be set to 2) (the examples in SDK are 2).
s32frame_size	Period sample size: the number of samples sent to the ADEC module each time.
s32ChannelNums	The number of channels (mono: 1, dual: 2).
s32Sample_rate	The sampling frequency (HZ) of the code stream to be decoded.

【Note】

None.

【Related Data Type and Interface】

None.

10.4.3.9 AUDIO_FRAME_INFO_S**【Description】**

Define the audio frame information structure after decoding.

【Syntax】

```
typedef struct _AUDIO_FRAME_INFO_S {
    AUDIO_FRAME_S *pstFrame;
    CVI_U32        u32Id;
} AUDIO_FRAME_INFO_S;
```

【Member】

Member	Description
pstFrame	Audio frame pointer.
u32Id	Index of audio frame, range [0, 49].

【Note】

None.

【Related Data Type and Interface】

CVI_ADEC_GetFrame CVI_ADEC_ReleaseFrame

10.4.3.10 ADEC_CHN_STATE_S**【Description】**

Define the Audio Decoding Channel Data Buffer Status Structure.

【Syntax】

```
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;
```

【Member】

Member	Description
bEndOfStream	End state of decoded stream.
u32BufferFrmNum	The total number of cache blocks in the decoding channel.
u32BufferFreeNum	The number of free cache blocks available.
u32BufferBusyNum	Number of occupied cache blocks

【Note】

None.

【Related Data Type and Interface】

None.

10.4.3.11 ADEC_DECODER_S**【Description】**

Define the decoder attribute structure.

【Syntax】

```
typedef struct _ADEC_DECODER_S {
    PAYLOAD_TYPE_E enType;
    CVI_CHAR aszName[17];

    CVI_S32 (*pfnOpenDecoder)(CVI_VOID *pDecoderAttr, CVI_VOID
                             **ppDecoder);
    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;
```

【Member】

Member	Description
enType	Decoding protocol type
aszName	Decoder name.
pfnOpenDecoder	Function pointer to open decoder.
pfnDecodeFrm	Function pointer to decode
pfnGetFrmInfo	Function pointer to get audio frame information.
pfnCloseDecoder	Function pointer to close decoder.
pfnResetDecoder	Clear the buffer and reset the decoder.

【Note】

This structure is specially used to link external AAC decoding lib.

Currently, please refer to middleware/sample/audio/aac_sample for AAC decoding.

【Related Data Type and Interface】

None.

10.4.4 Built-in Codec

Built-in Codec related data types and data structures are defined as follows:

- CVI_vol_ctrl: Define the built-in Audio Codec volume control structure.

10.4.4.1 ACODEC_VOL_CTRL

【Description】

Define the built-in Audio Codec volume control structure.

【Syntax】

```

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;
};

```

【Member】

Member	Description
vol_ctrl	Volume size
vol_ctrl_mute	Mute control

【Note】

None.

【Related Data Type and Interface】

None.

10.5 Error Codes

10.5.1 Audio Basic Attribute Error CodeS

Cvitek audio uses CVI_SUCCESS/CVI_FAILURE to represent the basic returned error code:

```
#define CVI_SUCCESS 0

#define CVI_FAILURE (-1)
```

Users should note that CVI_TRUE/CVI_FALSE is only used as a basic judgment reply, not as a Success or failure judgment:

```
#define CVI_TRUE 1

#define CVI_FALSE 0
```

10.5.2 Audio Input Error Codes

Cvitek audio input error code refers to cvi_comm_aio.h related settings. The corresponding error codes are all beginning with CVI_ERR_AI:

Error Code	Macro Definition	Description
0xAA000001	CVI_ERR_AIO_ILLEGAL_PARAM	Invalid audio input parameter setting
0xAA000002	CVI_ERR_AIO_NULL_PTR	Input parameter null pointer error
0xAA000003	CVI_ERR_AIO_NOT_PERM	Operation not permitted
0xAA000004	CVI_ERR_AIO_REGISTER_ERR	Fail to register
0xA0000005	CVI_ERR_AI_INVALID_DEVID	Illegal device ID
0xA0000006	CVI_ERR_AI_INVALID_CHNID	Illegal channel ID
0xA0000001	CVI_ERR_AI_ILLEGAL_PARAM	Invalid audio input parameter setting
0xA0000002	CVI_ERR_AI_NULL_PTR	Input parameter null pointer error
0xA0000007	CVI_ERR_AI_NOT_CONFIG	Parameter not set
0xA0000008	CVI_ERR_AI_NOT_SUPPORTED	Parameters set in are not supported
0xA0000009	CVI_ERR_AI_NOT_ENABLED	AI cannot be enabled in this state
0xA0000003	CVI_ERR_AI_NOT_PERM	The set parameter is not supported
0xA000000A	CVI_ERR_AI_NOMEM	insufficient memory
0xA000000B	CVI_ERR_AI_NOBUF	The buffer is not set or initialized
0xA000000C	CVI_ERR_AI_BUF_EMPTY	Buffer data is empty
0xA000000D	CVI_ERR_AI_BUF_FULL	Buffer full
0xA000000E	CVI_ERR_AI_SYS_NOTREADY	The system is busy and not ready for use
0xA000000F	CVI_ERR_AI_BUSY	AI module busy
0xA0000010	CVI_ERR_AI_VQE_ERR	VQE module error
0xA0000011	CVI_ERR_AI_VQE_BUF_FULL	VQE Buffer is empty
0xA0000012	CVI_ERR_AI_VQE_FILE_UNEXIST	The VQE configuration file does not exist.

10.5.3 Audio Output Error Codes

Cvitek audio input error code refers to `cvi_comm_aio.h` related settings. The corresponding error codes are all beginning with `CVI_ERR_AO`:

Error Code	Macro Definition	Description
0xA1000001	CVI_ERR_AO_INVALID_DEVID	Illegal device ID
0xA1000002	CVI_ERR_AO_INVALID_CHNID	Illegal channel ID
0xA1000003	CVI_ERR_AO_ILLEGAL_PARAM	Invalid audio output parameter setting
0xA1000004	CVI_ERR_AO_NULL_PTR	Input parameter null pointer error
0xA1000005	CVI_ERR_AO_NOT_CONFIG	Parameter not set
0xA1000006	CVI_ERR_AO_NOT_SUPPORTED	Parameters set in are not supported
0xA1000007	CVI_ERR_AO_NOT_PERM	operation not permitted
0xA1000008	CVI_ERR_AO_NOT_ENABLED	AI cannot be enabled in this state
0xA1000009	CVI_ERR_AO_NOMEM	insufficient memory
0xA100000A	CVI_ERR_AO_NOBUF	The buffer is not set or initialized
0xA100000B	CVI_ERR_AO_BUF_EMPTY	Buffer data is empty
0xA100000C	CVI_ERR_AO_BUF_FULL	Buffer is full
0xA100000D	CVI_ERR_AO_SYS_NOTREADY	The system is busy and not ready for use
0xA100000E	CVI_ERR_AO_BUSY	AO module busy
0xA100000F	CVI_ERR_AO_VQE_ERR	AI_VQE module error

10.5.4 Audio Encoding Error Codes

Cvitek audio input error code refers to `cvi_comm_aenc.h` related settings. The corresponding error codes are all beginning with `CVI_ERR_AENC`:

Error Code	Macro Definition	Description
0xA2000001	CVI_E RR_AENC_INVALID_DEVID	Illegal device ID
0xA2000002	CVI_E RR_AENC_INVALID_CHNID	Illegal channel ID
0xA2000003	CVI_E RR_AENC_ILLEGAL_PARAM	Invalid setting of audio encoding parameters
0xA2000004	CVI_ERR_AENC_EXIST	Audio coding module is on
0xA2000005	CVI_ERR_AENC_UNEXIST	The status of audio coding module is nonexistent
0xA2000006	CVI_ERR_AENC_NULL_PTR	Input parameter null pointer error
0xA2000007	CVI_ERR_AENC_NOT_CONFIG	Parameter not set
0xA2000008	CVI_ERR_AENC_NOT_SUPPORTED	Parameters set in are not supported
0xA2000009	CVI_ERR_AENC_NOT_PERM	Operation not permitted
0xA200000A	CVI_ERR_AENC_NOMEM	insufficient memory
0xA200000B	CVI_ERR_AENC_NOBUF	The buffer is not set or initialized
0xA200000C	CVI_ERR_AENC_BUF_EMPTY	Buffer data is empty
0xA200000D	CVI_ERR_AENC_BUF_FULL	Buffer full
0xA200000E	CVI_ERR_AENC_SYS_NOTREADY	System is busy and not ready for use
0xA200000F	CVI_ERR_AENC_ENCODER_ERR	AENC Coding error
0xA2000010	CVI_ERR_AENC_VQE_ERR	VQE module error

10.5.5 Audio Decoding Error Codes

Cvitek audio input error code refer to cvi_comm_aenc.h related settings. The corresponding error codes are all beginning with CVI_ERR_ADEC:

Error Code	Macro Definition	Description
0xA3000001	CVI_E RR_ADEC_INVALID_DEVID	Illegal device ID
0xA3000002	CVI_E RR_ADEC_INVALID_CHNID	Illegal channel ID
0xA3000003	CVI_E RR_ADEC_ILLEGAL_PARAM	Invalid setting of audio encoding parameters
0xA3000004	CVI_ERR_ADEC_EXIST	Audio coding module is on
0xA3000005	CVI_ERR_ADEC_UNEXIST	The status of audio coding module is nonexistent
0xA3000006	CVI_ERR_ADEC_NULL_PTR	Input parameter null pointer error
0xA3000007	CVI_ERR_ADEC_NOT_CONFIG	Parameter not set
0xA3000008	CVI_ERR_ADEC_NOT_SUPPORTED	Parameters set in are not supported
0xA3000009	CVI_ERR_ADEC_NOT_PERM	Operation not permitted
0xA300000A	CVI_ERR_ADEC_NOMEM	Insufficient memory
0xA300000B	CVI_ERR_ADEC_NOBUF	The buffer is not set or initialized
0xA300000C	CVI_ERR_ADEC_BUF_EMPTY	Buffer data is empty
0xA300000D	CVI_ERR_ADEC_BUF_FULL	Buffer full
0xA300000E	CVI_ERR_ADEC_SYS_NOTREADY	System is busy and not ready for use
0xA300000F	CVI_ERR_ADEC_DECODER_ERR	ADEC coding error
0xA3000010	CVI_ERR_ADEC_BUF_LACK	Insufficient input buffer space for ADEC decoding

10.6 Related Tests

10.6.1 Unit Test

Test purpose: the Audio AEC function test
Test module: Audio VQE - AEC
Test method: sample_audio_aec \$(filename)
Note:Users can use the cvi_aec_test program to convert recorded audiofiles (raw or .wav) with echo to audio files with echo cancelled.Please note that this function only supports sampling rates of 8kHzand 16kHz, and the format must be S16LE.

Test purpose: Audio Codec function test
Test module: Audio Encode/Decode
Test method: sample_audio_transcode \$(filename)
Note:Users can convert audio raw data to g.711/g.726 files throughsample_audio_transcode. Please note that this function only supports8kHz and 16kHz sampling rates, and its format must be S16LE.

10.6.2 Functional Test

Test purpose: Audio Recording function test
Test module: Audio In
Test method: 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

Test purpose: Audio broadcasting function test
Test module: Audio Out
Test method: 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

Test purpose: Audio resampling test
Test module: Audio Resample
Test method: sample_audio_resample (input __raw format file) (inputfile sample rate) (target sample rate)
Note:Examples are as follows:sample_audio_resample record.raw 16000 48000As shown above, the user inputs the raw file in order. The programwill resample the file according to the current sampling rate andtarget sampling rate of the file according to the API inlibcvi_RES1.so. After that, the raw file starting with the outputsampling rate will be output.

Test purpose: Audio output volume setting and test
Test module: Audio Output Volume(DAC codec)
Test method: (set volume): sample_audio 6 (Get the current output volume): sample_audio 8
Note:Users can set the volume through sample audio 6.

10.6.3 Performance Test

Test purpose: test the function of Audio quality enhancement, and measure the corresponding ability of speech algorithm according to parameter adjustment.
Test module:: Audio VQE
Test method: sample_audio_nr \$(filename)
<p>Note: type sample_audio_nr and input. Wav file, VQE related switches and function parameters will be inquired in order, support NR (noise reduction) and AGC (automatic gain control). Usage example: _____ Enter NR off:0 , On:1 : 1 _____ Enter AGC off:0 , On:1 : 1 pstAiVqeAttr.u32OpenMask[0x28] Enter agc_max_gain [0, 3] 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 As shown above, AGC (automatic gain control) has six groups of parameters to set, and NR (noise reduction) has two groups of parameters.</p> <p>Users can refer to chapter 9.2.2 to allocate the parameters suitable for the current environment.</p> <p>After the program is finished, the file name starting with NR_AGC will be output according to the input file name.</p> <p>Users can dial the file or put it on the computer to analyze the audio results.</p>

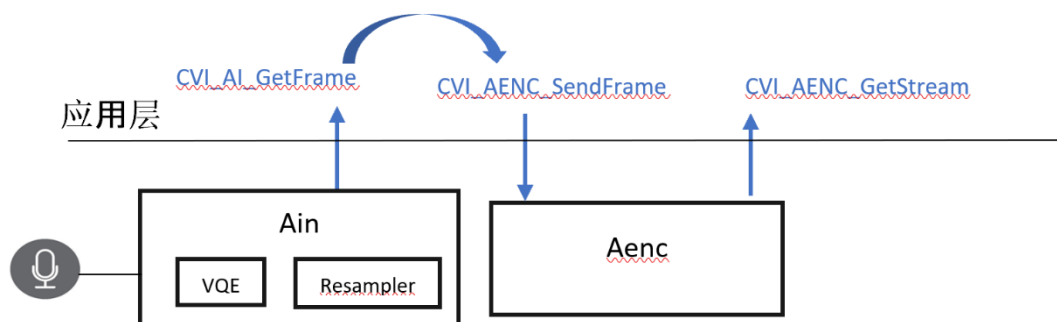
10.7 Sample Code and Board-side Component Preliminary Testing

10.7.1 Description of the Sample Code

User-Get Mode :	<p>Advantage: SDK users can obtain and record the status of each audio frame, and users can perform operations such as drop / delay / copy at the application layer.</p> <p>Disadvantages: SDK users need to open threads to do get/send actions at the application layer.</p>
Bind Mode :	<p>Advantages: SDK users only need to be responsible for fetching or broadcasting data after creating channel parameters.</p> <p>Disadvantage: The application layer cannot deploy and observe the current state.</p>

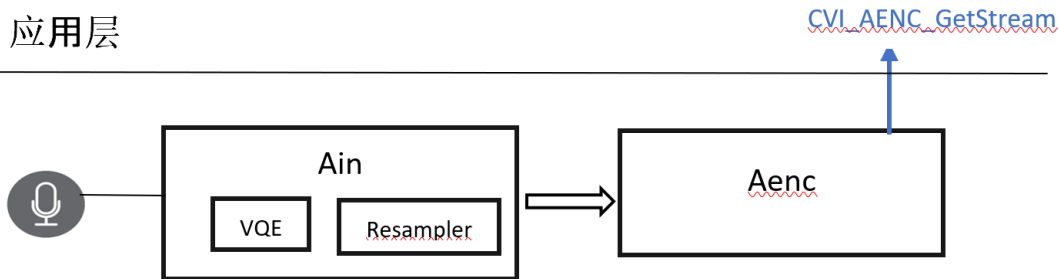
Uplink Audio (from microphone to encoding)

Reference source: cvi_sample_audio.c : SAMPLE_AUDIO_AiAenc



The diagram above illustrates the User-Get Mode.

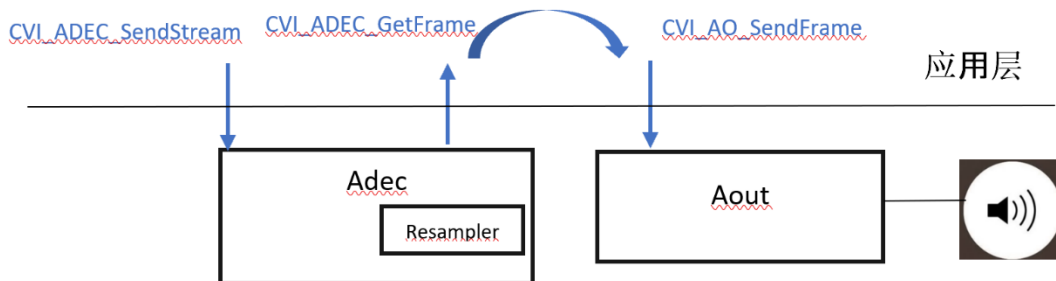
应用层



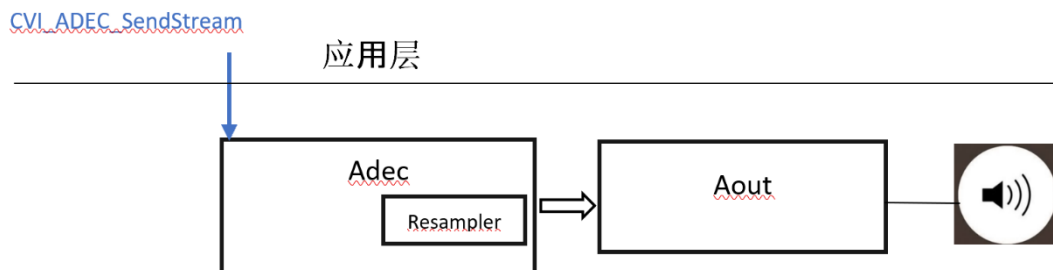
The diagram above illustrates the Bind Mode.

Downlink Audio (from decoding to playback):

Reference source: `cvi_sample_audio.c` : `SAMPLE_AUDIO_AdecAo`



The diagram above illustrates the User-Get Mode.



The diagram above illustrates the Bind Mode.

10.7.2 Preliminary Testing of the Board-side Component

For different versions released with the project, it is important to provide audio version information to synchronize with application engineers.

After receiving the SDK software, users can use audio-related software to obtain version and board audio status information.

The implementation of the `sample_audio` executable program corresponds to the code in `cvi_sample_audio.c`.

During the initial customization of the board, users can perform dual-channel recording using the following method to confirm the audio waveform of the stored file, verify the correct input, and confirm that the speakers are functioning properly by directly playing back the dual-channel recording after recording.

[Checking Your Audio SDK Version]:

When running

sample_audio 7,

the console will display the following:

version [cviaudio_2021_tinyalsa0407A]

The information inside the square brackets [] indicates the audio version.

[Example Code for Recording]:

When running

sample_audio 19,

it corresponds to the code flow in case 19 of cvi_sample_audio.c.

The program prompts the user to enter recording parameters, and the user can simply press Enter to use the default values (as shown in the figure below).

When the console displays: _____>start recording...

Recording will begin for the specified number of seconds according to the user's settings, and the audio will be saved as "sample_record.raw" on disk.

[Example Code for Playback]:

When running

sample_audio 20 檔名.raw,

it corresponds to the code flow in case 20 of cvi_sample_audio.c.

The program prompts the user to enter playback parameters, and the user can simply press Enter to use the default values (as shown in the figure below).

When the console displays: _____>start playing... Playback will begin based on the user's configuration and file name.


```

[root@cvitek]/tmp# sample_audio 19
User console mode
cvi_sample_audio:Enter command id =[19]
start register AAC encoder xxxxxxxxxxxxxxxx
PT_AAC[37]
entype[37]
u32MaxFrmlen[100]
start register AAC encoder end
start register AAC encoder xxxxxxxxxxxxxxxx
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...

```

```

[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 xxxxxxxxxxxxxxxx
PT_AAC[37]
entype[37]
u32MaxFrmlen[100]
start register AAC encoder end
start register AAC encoder xxxxxxxxxxxxxxxx
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]
=====
[cvinaudio] AoDevId[0] in CVI_AO_Enable
[cvinaudio][dbg] 16
[cvinaudio][dbg] create ao output thread ok.
CVI_AO_EnableChn -----> go
[cvinaudio][dbg] CVI_AO_EnableChn DevId:0,chn:0.
----->start playing...
[cvinaudio][dbg] output_thread in sleepus:30000,period:960,rate:16000,ch:2.

```

The Audio Release SO file is a dynamic link library that contains the necessary libraries to ensure the completeness of the Audio SDK API.

Please ensure that the SDK released with your project includes the following libraries:

Name	unctionality
libcvi_audio.so	Audio API integration layer
libcvi_RES1.so	Audio resampling module
libcvi_VoiceEngine.so	Audio encoding/decoding module
libcvi_vqe.so	Audio algorithm integration layer
libssp.so	Audio SSP (including AEC) algorithm
libaec.so	Audio AEC algorithm
libtinyalsa.so	Audio recording and playback ALSA layer
libaacenc2.so	AAC encoding related - please contact FAE personnel for official documentation
libaacdec2.so	AAC decoding related - please contact FAE personnel for official documentation

GEOMETRIC DISTORTION CORRECTION SUBSYSTEM

11.1 Function Overview

The Geometric Distortion Correction Subsystem (GDC) provides functionalities for fisheye correction, rotation, and affine mapping for a frame of image.

This system is designed to correct the distortion and displacement of images caused by the lens characteristics when they are projected onto the sensor.

Users can use the GDC to correct the image accordingly.

The GDC supports rotation and lens distortion correction.

11.2 Design Overview

GDC module uses JOB as the structure of TASK management.

A JOB can contain a multiple TASK.

GDC guarantees that TASK is executed in the order in which JOBS are added.

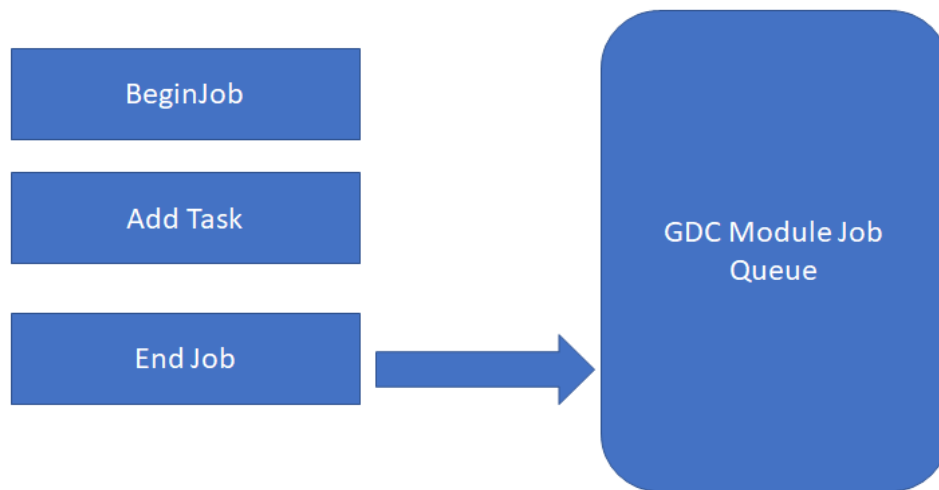
All TASKs under END are submitted and resources released.

If TASK commit fails, you must use Cancel JOB to free resources (GDC driver code has been built-in and users do not need to handle it by themselves)

TASK is used to complete one or more operations on an image, such as fish eye correction, Affine correction, etc.

11.2.1 System Architecture

The working mode of GDC follows the first-come-first-served rule.



11.3 API Reference

11.3.1 CVI_GDC_BeginJob

【Description】

Start a Job.

【Syntax】

```
CVI_S32 CVI_GDC_BeginJob(GDC_HANDLE *phHandle);
```

【Parameter】

Parameter	Description	Input/Output
phHandle	get HANDLE	Output

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: cvi_gdc.h
- Library files: libvpu.so

【Note】

- Multiple jobs can be opened simultaneously in GDC.

For example, while a job for lens distortion correction (LDC) is being performed by VI, a job for rotation can also be executed by VO.

However, it is important to check if the CVI_GDC_BeginJob function returns successfully before using the HANDLE returned by phHandle.

- phHandle cannot be a null or illegal pointer.

【Example】

Please refer to `CVI_GDC_AddRotationTask`

【Related Topic】

- [CVI_GDC_EndJob](#)

11.3.2 CVI_GDC_EndJob

【Description】

End a job, and all Tasks in this job will be submitted to the GDC module

【Syntax】

```
CVI_S32 CVI_GDC_EndJob(GDC_HANDLE hHandle);
```

【Parameter】

Parameter	Description	Input/Output
hHandle	Open Job Handle	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: `cvi_gdc.h`
- Library files: `libvpu.so`

【Note】

- hHandle must be an open job
- hHandle cannot be a null or illegal pointer.

【Example】

Please refer to `CVI_GDC_AddRotationTask`

【Related Topic】

- [CVI_GDC_BeginJob](#)

11.3.3 CVI_GDC_CancelJob

【Description】

Cancel a job, and all Tasks in this job will not be submitted to the GDC module

【Syntax】

```
CVI_S32 CVI_GDC_CancelJob(GDC_HANDLE hHandle);
```

【Parameter】

Parameter	Description	Input/Output
hHandle	opened Job Handle	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: `cvi_gdc.h`
- Library files: `libvpu.so`

【Note】

- `hHandle` must be an open job.
- `phHandle` cannot be a null or illegal pointer.

【Related Topic】

- [CVI_GDC_BeginJob](#)

11.3.4 CVI_GDC_AddRotationTask

【Description】

It is possible to insert a rotation task into an existing job.

However, do not support 180-degree rotation.

【Syntax】

```
CVI_S32 CVI_GDC_AddRotationTask(GDC_HANDLE hHandle, const GDC_TASK_ATTR_S *pstTask,
↪ROTATION_E enRotation);
```

【Parameter】

Parameter	Description	Input/Output
<code>hHandle</code>	Opened Job Handle	Input
<code>pstTask</code>	GDC Task pointer	Input
<code>enRotation</code>	Rotation angle (0, 90, 180, 270);	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: `cvi_gdc.h`, `cvi_comm_gdc.h`
- Library files: `libvpu.so`

【Note】

- `hHandle` must be an open job
- `phHandle` cannot be a null or illegal pointer.

【Example】

```

    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;

```

【Related Topic】

- [CVI_GDC_BeginJob](#)

11.3.5 CVI_GDC_GenLDCMesh

【Description】

Generating a mesh for LDC properties involves creating a mesh that stores the correspondence between the pixels in the original image and the corrected image.

【Syntax】

```

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);

```

【Parameter】

Parameter	Description	Input/Output
U32Width	Image width after correction	Input
u32Height	Image height after correction	Input
pstLDCAttr	Lens Distortion Correction (LDC) properties	Input
name	Mesh name	Input
pu64PhyAddr	Physical address	Input
ppVirAddr	Virtual address	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: cvi_gdc.h, cvi_comm_gdc.h
- Library files: libvpu.so

【Example】

Please refer to CVI_GDC_AddLDCTask

【Related Topic】

- [CVI_GDC_BeginJob](#)

11.3.6 CVI_GDC_AddLDCTask

【Description】

Inserting a lens distortion correction task into an existing job involves two tasks to complete the correction. The first task involves rotating the image by 90 degrees, and the second task involves rotating it by 270 degrees, with correction applied in different directions during each rotation.

【Syntax】

```
CVI_S32 CVI_GDC_AddLDCTask(GDC_HANDLE hHandle, const GDC_TASK_ATTR_S *pstTask, const LDC_ATTR_S *pstLDCAttr, ROTATION_E enRotation);
```

【Parameter】

Parameter	Description	Input/Output
hHandle	Opened Job Handle	Input
pstTask	GDC Task pointer	Input
pstLDCAttr	LDC attributes	Input
enRotation	Rotation angle (90, 270);	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure. For details, please refer to Error Codes .

【Requirement】

- Header files: cvi_gdc.h, cvi_comm_gdc.h

- Library files: libvpu.so

【Note】

- hHandle must be an open job
- phHandle cannot be a null or illegal pointer.

【Example】

```

    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 Failure, 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];
    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));

```

(continues on next page)

(continued from previous page)

```

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 Failure, 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
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 Failure, 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 Failure, s32Ret:0x%x\n", s32Ret);
    CVI_GDC_CancelJob(hHandle);
    goto EXIT1;
}
return 0;

```

【Related Topic】

- *CVI_GDC_BeginJob*

11.4 Data Types

11.4.1 GDC_TASK_ATTR_S

【Description】

Define the attributes of GDC TASK. GDC supports pixel formats such as NV21, NV12, and YUV400.

【Syntax】

```

typedef struct _GDC_TASK_ATTR_S {
    VIDEO_FRAME_INFO_S stImgIn;
    VIDEO_FRAME_INFO_S stImgOut;
    CVI_U64 au64privateData[4];

```

(continues on next page)

(continued from previous page)

```

    CVI_U64 reserved;
} GDC_TASK_ATTR_S;

```

【Member】

Member	Description
stImgIn	Input image properties
stImgOut	Output image properties
au64privateData	Task related private data
reserved	Reserved

【Note】

The image attributes must match the corresponding hardware attributes

【Related Data Type and Interface】

- VIDEO_FRAME_INFO_S

11.4.2 LDC_ATTR_S

【Description】

Define the Lens Distortion Correction (LDC) properties.

【Syntax】

```

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;
} LDC_ATTR_S;

```

【Member】

Member	Description	Value Range
bAspect	Whether aspect ration is keep	bool
s32XRatio	Field angle ration of horizontal. Valid when bAspect=1.	0~100
s32YRatio	Field angle ration of vertical. Valid when bAspect=0.	0~100
s32XYRatio	Field angle ration of all,valid when bAspect=1.	0~100
s32CenterXOffset	Horizontal offset of the image center point relative to the physical center point.	-511~+511
s32CenterYOffset	Vertical offset of the image center point relative to the physical center point.	-511~+511
s32DistortionRatio	Correction strength Negative values for pincushion distortion. Positive values for barrel distortion.	-300~+500

【Note】

The image attributes must match the corresponding hardware attributes

【Related Data Type and Interface】

- VIDEO_FRAME_INFO_S

11.4.3 FISHEYE_MOUNT_MODE_E

【Description】

Define the installation mode in the FISHEYE property.

【Syntax】

```
typedef enum _FISHEYE_MOUNT_MODE_E {  
    FISHEYE_DESKTOP_MOUNT = 0,  
    FISHEYE_CEILING_MOUNT = 1,  
    FISHEYE_WALL_MOUNT = 2,  
    FISHEYE_MOUNT_MODE_BUTT  
} FISHEYE_MOUNT_MODE_E;
```

【Member】

Member	Description
FISH-EYE_DESKTOP_MOUNT	mounted on desktop or floor
FISH-EYE_CEILING_MOUNT	mounted on ceiling
FISHEYE_WALL_MOUNT	mounted on wall

【Note】

None.

【Related Data Type and Interface】

None.

PROC DEBUGGING INFORMATION EXPLANATION

12.1 Function Overview

CV181x/CV180x displays debugging information through the proc file system under Linux, and the debugging information reflects the running status of the current system.

Users can analyze and locate problems through these debugging information.

12.2 AI

【Debug Information】

```
/ # 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
```

【Analysis】

Record the current audio input device Parameters and status information.

【Parameter Description】

Parameter	Decription
AiDev	AI device ID
Workmode	AI working mode master: I2S master mode slave: I2S slave mode
SampleRate	Sample rate. Range: [8k ~ 48K]
BitWidth	Sampling accuracy. Range: [16bit]
I2S0	I2S status of the ADC codec interface. on: open off: closed
SDMA clk	sysDMA clock status on: open off: closed
ADC	ADC codec status 0x0: closed 0x3: open
L-Mute	Left channel mute mode yes: mute mode open no: mute mode closed
R-Mute	Right channel mute mode yes: mute mode open no: mute mode closed
L-Vol	Left channel volume Value range [0~24]
R-Vol	Right channel volume Value range [0~24]

12.3 AO

【Debug Information】

```
/ # 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
```

【Analysis】

Record the current audio output device 参数名称 s and status information.

【Parameter Description】

Parameter	Decription
AiDev	AO device ID
Workmode	AO working mode master: I2S master mode slave: I2S slave mode
SampleRate	Sample rate. Range: [8k ~ 48K]
BitWidth	Sampling accuracy. Range: [16bit]
I2S3	I2S status of the ADC codec interface. on: open off: closed
SDMA clk	sysDMA clock status on: open off: closed
DAC	ADC codec status 0x0: closed 0x3: open
L-Vol	Left channel volume. Range: [0~32]
R-Vol	Right channel volume. Range: [0~32]

12.4 VENC

【Debug Information】

```
# 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:
```

【Analysis】

Record the current video encoding attribute configuration and status information.

【Parameter Description】

Parameter		Decription
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

continues on next page

Table 12.1 – continued from previous page

Parameter		Description
	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
	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)

continues on next page

Table 12.1 – continued from previous page

Parameter		Description
	StartY	Start vertical coordinate of the ROI (unit: pixel)
VENC PTSSTATETimestamp of a frame received by the channel	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
	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

【Debug Information】

```
# 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
↪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
```

【Analysis】

Record the current H.265 video encoding attribute configuration and status information.

【Parameter Description】

Parameter		Description
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

【Debug Information】

```
# cat /proc/cvitek/h264e
Module: [H264E] System Build Time [#1 SMP PREEMPT Sat Jan 23 02:50:26 CST 2021]
-----MODULE PARAM-----
OnePack: 0      H264eVBSrc: 0      PowerSaveEn: 0  MiniBufMode: 0  bQpHstgrmEn: 0
↪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
```

【Analysis】

Record the current H.264 video coding attribute configuration and status information.

【Parameter Description】

Parameter		Decription
MODULEPARAM	OnePack	Mode in which streams are obtained 0: Streams are obtained in multi-packet mode. 1: Streams are obtained in single-packet mode.
	H264eVBSrc	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}
	GopMode	GOP mode
	MaxStrCnt	Maximum number of frames in the stream buffer Default value: 200

12.7 JPEG

【Debug Information】

```
# cat /proc/cvitek/jpeg
Module: [JPEG] 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
```

【Analysis】

Record the encoding properties and status of each channel during JPEG encoding.

【Parameter Description】

Parameter		Description
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 CPU subsystem (ECS)
	ByFrm	Whether to obtain streams by frame
12.7. JPEG		Value: {0, 1} 480
	Qfactor	Channel Qfactor
	C2GEn	Color-to-gray enable Value: {0, 1}

12.8 RC

【Debug Information】

```
# 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␣
↪  BFrNum: 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
```

【Analysis】

Record the current bitrate control information.

【Parameter Description】

Parameter		Description
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-frames relative to the common P-frames Value range: [-10, +30]
	BFrmNum	Number of B-frames Value range: [1, 3] This is currently not supported.
	BQpDelta	QP delta of B-frames relative to P-frames Value range: [-10, +30] The B-frame is currently not supported.
12.8. RC		
	BgInterval	Interval of Bg frames Value range: greater than or equal to the GOP value
	ViQpDelta	QP delta of the virtual I-frames

12.9 VDEC

【Debug Information】

```
# cat /proc/cvitek/vdec
Module: [VDEC] System Build Time [#1 SMP PREEMPT Sat Jan 23 02:50:26 CST 2021]
VdecMaxChnNum: 70      MiniBufMode: 0   enVdecVBSrc: 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:
```

【Analysis】

Record the usage of the current decoding channel and its attribute configuration. It can be used to check the property configuration and the current decoding channel statistical status.

【Parameter Description】

Parameter		Description
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)
	enVdecVBSrc	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
	MaxPicWidth	Maximum width supported for video decoding
	MaxPicHeight	Maximum height supported for video decoding

continues on next page

Table 12.2 – continued from previous page

Parameter		Description
	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-chip 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
	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

continues on next page

Table 12.2 – continued from previous page

Parameter		Description
	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 参数名称 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.
CHNVIDEOATTR &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

continues on next page

Table 12.2 – continued from previous page

Parameter		Description
CHN imageATTR &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

【Debug Information】

# 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)

↪-----	

【Analysis】

- Record the current debugging level of each module.
- cat /proc/cvitek/log is used to obtain the log level information of each module.
- Modify log level:
 - To modify the debug level of a module, use the echo command, forexample:echo “VENC=4” > /proc/cvitek/log
 - Modify the debug level of all modulesecho “ALL=4” > /proc/cvitek/log

【Parameter Description】

Parameter		Decription
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	Module name, followed by the number of log print level.

12.11 SYS

【Debug Information】

# 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				

↪-----								

【Analysis】

Record the current usage of SYS module.

【Parameter Description】

Parameter		Decription
BIND RELATION TABLE	1stMod	The module name of the first level in the binding relationship, and the data is sent from the first level to the second level.
	1stDev	The device number of the first level in the binding relationship, and the data is sent from the first level to the second level.
	1stChn	The channel number of the first level in the binding relationship, and the data is sent from the first level to the second level.
	2ndMod	The module name of the second level in the binding relationship, and the data is sent from the first level to the second level.
	2ndDev	The device number of the second level in the binding relationship, and the data is sent from the first level to the second level.
	2ndChn	The channel number of the second level in the binding relationship, and the data is sent from the first level to the second level.
	3rdMod	The module name of the third level in the binding relationship. If there is a third level binding relationship, the data is sent to the third level by the second level, otherwise it will be null.
	3rdDev	The device number of the third level in the binding relationship.
	3rdChn	The channel number of the third level in the binding relationship.

12.12 VB

【Debug Information】

```
# 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
```

(continues on next page)

(continued from previous page)

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	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	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	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	1	
→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	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	
#5	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	
#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	
→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	
→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	
→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	
→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	

(continues on next page)

(continued from previous page)

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		↵						
Sum	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		↵						

↵-----																				
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									↵	
#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									↵	
#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									↵	
#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									↵	
#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									↵	
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									↵	

↵-----																				

【Analysis】

Record the current VB module buffer usage.

【Parameter Description】

Parameter		Decription
VB PUB CONFIG	MaxPoolCnt	Maximum number of cache pools
	MaxBlkCnt	The maximum number of cache blocks.
COMMON POOL CONFIG	PoolId	Handles to the public cache pool.
	Size	The size of the block in the cache pool.
	Count	The number of blocks in the cache pool.
PER VB POOL INFO	PoolName	Public / private cache pool name, if not set, default to vbpool
	PoolId	Handles to the public / private cache pool
	PhysAddr	The starting physical address of the public / private cache pool.
	VirtAddr	The starting virtual address of the public / private cache pool.
	IsComm	Whether the cache pool is public. Value: {0, 1}.
	Owner	The owner of the cache pool. -2: private pool -1: public pool
	BlkSz	The size of the cache block in the cache pool.
	BlkCnt	The number of cache blocks in the cache pool.
	Free	The number of free cache blocks in the cache pool.
	MinFree	The minimum remaining number of free cache blocks since the program runs. If the count is 0, it indicates that there may be frame loss due to insufficient buffer blocks.
	BLK	Handle to the cache block in the cache pool.
	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	Module name. The corresponding number below indicates how many places of the current module occupy the cache block in the cache pool. 0: not occupied N: N blocks are occupied

12.13 GDC

【Debug Information】

```
# 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)				
# 0	VO	1	SUCCESS	921600	921600	↪
↪ 921600		3952	3539			↪
# 1	VO	1	SUCCESS	921600	921600	↪
↪ 921600		3941	3534			↪
# 2	VO	1	SUCCESS	921600	921600	↪
↪ 921600		3952	3553			↪
# 3	VO	1	SUCCESS	921600	921600	↪
↪ 921600		3938	3540			↪
# 4	VO	1	SUCCESS	921600	921600	↪
↪ 921600		3947	3547			↪
# 5	VO	1	SUCCESS	921600	921600	↪
↪ 921600		3936	3536			↪
# 6	VO	1	SUCCESS	921600	921600	↪
↪ 921600		3948	3551			↪
# 7	VO	1	SUCCESS	921600	921600	↪
↪ 921600		3926	3530			↪
-----MAX WASTE TIME JOB INFO-----						
ModName	TaskNum	State	InSize(pixel)	OutSize(pixel)		
↪ CostTime(us)		HwTime(us)				
VO	1	SUCCESS	921600	921600	↪	
↪ 4007						↪
3530						
-----GDC JOB STATUS-----						
Success	Fail	Cancel	BeginNum	BusyNum	ProcingNum	
754	0	0	0	0	0	↪
↪ 0						
-----GDC TASK STATUS-----						
Success	Fail	Cancel	BusyNum			
754	0	0	0			
-----GDC INT STATUS-----						
IntNum	IntTm(us)	HalProcTm(us)				
754	3536	126				
-----GDC CALL CORRECTION STATUS-----						
TaskSuc	TaskFail	EndSuc	EndFail	CbCnt		
0	0	0	0	0	0	

【Analysis】

Record several tasks recently completed by GDC module, including the most time-consuming tasks recently, and historical cumulative information.

【Parameter Description】

Parameter		Description
RECENT JOB INFO	SeqNo	Print sequence numbers. Value range: [0, 7]
	ModName	Module name that submitted the job
	TaskNum	Task numbers included in the job
	State	The processing state of the job. Value range: {FAIL, SUCCESS, WORKING} FAIL: job execution failed SUCCESS: job execution success WORKING: job execution in progress
	InSize(pixel)	The sum of the input image area of each task in the job Unit: pixels Every time a task is added to the job, the input area of the task is added to this item.
	OutSize(pixel)	The sum of the output image area of each task in the job Unit: pixels Every time a task is added to the job, the output area of the task is added to this item.
	CostTime(us)	The job takes a long time from submission to completion. Unit: us. The time includes the processing time of software, hardware and interrupt service program for the task.
	HwTime(us)	The time that the job takes to process in hardware. Unit: US. This time is hardware processing time, generally shorter than CostTime.
MAX WASTE TIME JOBINFO	All items are the same as members of RECENT JOB INFO	The most time-consuming job in the last 500 tasks. Its items are the same as the members of recentjob info. Please refer to the above for details. When there are more time-consuming tasks or the total number of tasks exceeds 500, the group value is updated. Through this group of values, we can know the recent GDC performance, and whether GDC processing is not timely.
GDC JOB STATUS	Success	Cumulative number of jobs successfully processed. Increase by 1 when hardware processing is successful.
	Fail	Cumulative number of jobs that failed to process.
12.13. GDC		When GDC submits a task to the driver layer and fails, 1 is added. As the value increases,

12.14 REGION

【Debug Information】

```
# 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      Layer      CoordType
X          Y          W          H          Color
# 1      1      VPSS      0      100      100      FFFF      0      Y
30      0      100      100      FFFF      0      Y
↪ABS

-----REGION STATUS OF COVEREX-----
Hdl      Type      Used

-----REGION CHN STATUS OF RECT COVEREX-----
Hdl      Type      Mod      Dev      Chn      bShow      Layer
X          Y          W          H          Color

-----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
```

【Analysis】

Record the resource information of current region.

【Parameter Description】

Parameter		Decription
REGION STATUS OFCOVER	Hdl	Handle number of COVER
	Type	COVER type with a value of 1.
	Used	Whether the resource is occupied. N: not occupied Y: occupied
REGION CHN STATUS OFRECT COVER	Hdl	Handle number of COVER
	Type	COVER type with a value of 1.
	Mod	Module of Attach
	Dev	Device number

continues on next page

Table 12.3 – continued from previous page

Parameter		Description
	Chn	channel number
	bShow	Whether it is displayed in this channel. N: not displayed Y: display
	X	The starting X coordinate displayed in the channel.
	Y	The starting y coordinate displayed in the channel.
	W	COVER width (Unit: pixels)
	H	COVER height (Unit: pixels)
	Color	COVER color
	Layer	The layer displayed in the channel.
	CoorType	ratio coordiante or abs coordinate
REGION STATUS OFCOVEREX	Hdl	Handle number of COVEREX
	Type	COVEREX type, value: 2。
	Used	Whether the resource is occupied. N: not occupied Y: occupied
REGION CHN STATUS OFRECT COVEREX	Hdl	Handle number of COVEREX
	Type	COVEREX type, value: 2。
	Mod	Module of Attach
	Dev	Device number
	Chn	channel number
	bShow	Whether it is displayed in this channel. N: not displayed Y: display
	X	The starting X coordinate displayed in the channel.
	Y	The starting y coordinate displayed in the channel.
	W	COVEREX width (Unit: pixels)
	H	COVEREX height (Unit: pixels)
	Color	COVEREX color
	Layer	The layer displayed in the channel.
REGION STATUS OFOVERLAYEX	Hdl	Handle number of OVERLAYEX
	Type	OVERLAYEX type, value: 0。
	Used	Whether the resource is occupied. N: not occupied Y: occupied
	PiFmt	OVERLAYEX pixel format. Refer to PIXEL_FORMAT_E。

continues on next page

Table 12.3 – continued from previous page

Parameter		Description
	W	OVERLAYEX width (Unit: pixels)
	H	OVERLAYEX height (Unit: pixels)
	BgColor	OVERLAYEX background color
	Phy	The physical address of the memory occupied by OVERLAYEX
	Virt	The virtual address of memory occupied by OVERLAYEX
	Stride	OVERLAYEX memory span in bytes.
	CnvsNum	Number of OVERLAYEX memory
REGION CHN STATUS OFOVERLAYEX	Hdl	Handle number of OVERLAYEX
	Type	OVERLAYEX type, value: 0。
	Mod	Module of Attach
	Dev	Device number
	Chn	channel number
	bShow	Whether it is displayed in this channel. N: not displayed Y: display
	X	The starting X coordinate displayed in the channel.
	Y	The starting y coordinate displayed in the channel.
	Layer	The layer displayed in the channel.

12.15 VI

【Debug Information】

```
# 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

(continues on next page)

(continued from previous page)

-----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							
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			
↪ 1920	1080								

【Analysis】

Record the property configuration and status information of current video input device and channel.

【Parameter Description】

Parameter		Decription
MODULE PARAM	DetectErrFrame	<p>When the signal is unstable, the detected error frame image strategy is discarded in real time.</p> <p>This parameter is only used in debugging, and is not recommended in official products.</p> <p>Set this value to 0 and lose the error frame in real time.</p> <p>>0: when the number of consecutive error frames detected is greater than this value, it is considered as timing mismatch, and the subsequent frames will not be discarded;</p> <p>0: the default value, which means to discard the detected error frame image in real time.</p> <p>< 0: turn off the function of detecting error frame.</p>
	DropErrFrame	<p>When it is detected that the current frame is an error frame, the next few frames may also be error frames and should be discarded.</p> <p>0: the default value, which means that the continuous frame loss function is not enabled, and only the current error frame is discarded;</p> <p>>0: this parameter indicates that when an image error is detected, drop_err_frame frames (including the current frame) will be lost continuously, regardless of whether the subsequent image is correct or not.</p>
VI MODE	DevID	device number Valid range: [0, VI_MAX_DEV_NUM)。
	PrerawFE	PrerawFE (Pre-Raw Front End) working mode.
	PrerawBE	PrerawBE (Pre-Raw Back End) working mode.
	Postraw	Postraw (Post-Raw) working mode
	Scaler	Scaler working mode
VI DEV ATTR1	DevID	device number Valid range: [0, VI_MAX_DEV_NUM)。
	DevEn	Equipment enable. N: Close; Y: Open.

continues on next page

Table 12.4 – continued from previous page

Parameter		Description
	BindPipe	Whether the device is bound to a pipe. N: Unbound; Y: Bound.
	Width	Device width. Unit: pixels
	Height	Device height. Unit: pixels
	IntfM	Input mode. MIPI or BT or LVDS
	WkM	Working mode, now only 1MUX
	ScanM	Interlaced or line by line input. Value : {I, P} VI_SCAN_INTERLACED = I, VI_SCAN_PROGRESSIVE = P
VI DEV ATTR2	DevID	Device number. Valid range: [0, VI_MAX_DEV_NUM)。
	AD0	AD number
	AD1	AD number
	AD2	AD number
	AD3	AD number
	Seq	Data order. Value: {VUVU, UVUV, UYVY, VYUY, YUYV, YVYU}。
	DataType	Input data type, RGB by default.
	WDRMode	WDR mode. WDR_2L1: two in one line mode WDR_2F1: two in one frame mode WDR_3L1: three in one line mode WDR_3F1: three in one frame mode WDR_4L1: four in oneline mode WDR_4F1: four in oneframe mode
VI BIND ATTR	DevID	Device ID. Valid range: [0, VI_MAX_DEV_NUM)。
	PipeNum	Pipe number
	PipeId	PIPE ID. Valid range: [0, VI_MAX_PIPE_NUM)。
VI DEV TIMING ATTR	DevID	Device ID. Valid range: [0, VI_MAX_DEV_NUM)。

continues on next page

Table 12.4 – continued from previous page

Parameter		Description
	DevTimingEn	Whether to enable self-generated timing function: (is_offline_preraw) N: Shut down; ; Y: Open ;
	DevFrmRate	The frame rate of self-generating timing set by the user (When it is greater than the maximum frame rate supported by the device, the maximum frame rate of the device is returned.)
	DevWidth	Device width. Unit: pixels
	DevHeight	Device height. Unit: pixels
VI CHN ATTR1	ChnID	Channel ID.
	Width	Channel output width.
	Height	Channel output height.
	Mirror	mirror enable N: Shut down; Y: Open.
	Flip	flip enable N: Shut down; Y: Open.
	SrcFrmRate	Source frame rate.
	DstFrmRate	Destination frame rate.
	PixFmt	Output pixel format.
	VideoFmt	Output video format.
VI CHN ATT2	CompressMode	Whether to compress N: Shut down; Y: Open.
	DevID	Device ID. Valid range: [0, VI_MAX_DEV_NUM)。
	ChnID	Channel ID.
	Depth	The user obtains the queue depth of the channel frame.
	Align	The channel image line is stride aligned.
VI CHN OUTPUTRESOLUTION	ChnID	Channel ID.
	DevID	Device ID. Valid range: [0, VI_MAX_DEV_NUM)。
	Mirror	mirror enable N: Shut down; Y: Open.
	Flip	flip enable N: Shut down; Y: Open.
	Width	Channel output width.
	Height	Channel output height.
	PixFmt	Output pixel format.
	VideoFmt	Output video format.

continues on next page

Table 12.4 – continued from previous page

Parameter		Description
	CompressMode	Whether to compress N: Shut down; Y: Open.
	FrameRate	Frame rate
VI CHN EARLYINTERRUPT INFO	ChnID	Channel ID.
	DevID	Device ID. Valid range: [0, VI_MAX_DEV_NUM)。
	Enable	Whether to enable the function of reporting interruption in advance. N: Shut down; Y: Open.
	LineCnt	Number of lines that were reported in advance for interruption.
VI CHN CROP INFO	ChnID	Channel ID.
	DevID	Device ID. Valid range: [0, VI_MAX_DEV_NUM)。
	CropEn	Whether the CROP function is enabled. (cvi_isp_s_selection, V4L2_SEL_TGT_CROP) N: Shut down; Y: Open.
	CoorType	Coordinate type. RAT: relative coordinate; ABS: absolute coordinates.
	CoorX	Horizontal direction start coordinate. When the coordinate type is relative, the legal value range is [0,999]; When the coordinate type is absolute, the legal value range is [0, VI_CHN_MAX_WIDTH]。
	CoorY	Starting coordinate in vertical direction. When the coordinate type is relative, the legal value range is [0,999]; When the coordinate type is absolute, the legal value range is [0, VI_CHN_MAX_HEIGHT]。
	Width	CROP RECT width. This cannot exceed the maximum image width.
	Height	CROP RECT height. This cannot exceed the maximum image height.
	TrimX	The coordinates of the starting point of the actual image.

continues on next page

Table 12.4 – continued from previous page

Parameter		Description
	TrimY	The coordinates of the starting point of the actual image.
	TrimWid	Actual image width in pixels.
	TrimHgt	Actual image height in pixels.
VI CHN STATUS	ChnID	Channel ID.
	DevID	Device ID. Valid range: [0, VI_MAX_DEV_NUM)。
	Enable	Channel enable. 0: not enabled; 1: Enable.
	FrameRate	Frame Rate
	IntCnt	Channel interrupt count.
	RecvPic	Number of images received.
	LostFrame	Channel frame loss
	Width	Channel width
	Height	Channel height

12.16 VO

【Debug Information】

```
# 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			

【Analysis】

Record the current VO usage status and attribute configuration, including device status, video layer status and channel status. It can be used to dynamically obtain the current VO usage status for

debugging or testing.

【Parameter Description】

Parameter		Description
DEVICE CONFIG	DevID	Device ID. Value range: [0, VO_MAX_DEV_NUM)
	DevEn	Whether the device is enabled. N: Prohibition; Y: Enable.
	IntfType	Interface type. Value range: CVBS, YPBPR, VGA, BT656, BT1120, LCD, LCD_18BIT, LCD_24BIT, LCD_30BIT, MIPI, MIPI_SLAVE, HDMI, I80
	IntfSync	Interface timing. Value range: [0, VO_OUTPUT_BUTT)。
	BkClr	Background color of the device. Hex RGB888 format.
	DevFrt	Device frame rate, or refresh rate, is related to timing.
VIDEO LAYER STATUS 1	LayerId	Video layer ID. Value range: [0, VO_MAX_LAYER_NUM)。
	VideoEn	Whether the video layer is en- abled. N: Prohibition; Y: Enable.
	PixFmt	Enter the pixel format of the im- age.
	ImgW	The width of the video layer canvas.
	ImgH	The height of the video layer canvas.
	DispX	The starting abscissa of the dis- play area.
	DispY	The starting ordinate of the dis- play area.
	DispW	The width of the display area.
	DispH	The height of the display area.
	DispFrt	The display frame rate of the video layer.
VIDEO LAYER STATUS 2(continue)	LayerId	Video layer ID. Value range: [0, VO_MAX_LAYER_NUM)。
	DevId	The device ID of the video layer binding. Value range: [0, VO_MAX_DEV_NUM)。

continues on next page

Table 12.5 – continued from previous page

Parameter		Description
	EnChNum	Channel enable count. That is, how many channels in the video layer are enabled. Value range: [0, VO_MAX_CHN_NUM)。
	Luma	Brightness. Value range: [0, 100]。
	Cont	Contrast. Value range: [0, 100]。
	Hue	Hue. Value range: [0, 100]。
	Satu	Saturation. Value range: [0, 100]。
	BufLen	Displays the buffer length.
CHN BASIC INFO	LayerId	Video layer ID. Value range: [0, VO_MAX_LAYER_NUM)。
	ChnId	Channel ID. Value range: [0, VO_MAX_CHN_NUM)。
	ChnEn	Whether the channel is enabled. Y: Yes; N: No.
	Prio	Channel priority. Value range: [0, VO_MAX_CHN_NUM)。
	ChnX	The starting abscissa of the channel.
	ChnY	The initial ordinate of the channel.
	ChnW	Channel width.
	ChnH	Channel height.
	RotAngle	Channel rotation angle. Value range: [0, ROTATION_BUTT)
CHN PLAY INFO	LayerId	Video layer ID. Value range: [0, VO_MAX_LAYER_NUM)。
	ChnId	Channel ID. Value range: [0, VO_MAX_CHN_NUM)。
	Show	Whether the channel is displayed. N: Not displayed; Y: Display.
	Pause	Whether the channel is suspended. N: Prohibition; Y: Enable.
	Thrshd	The maximum number of image frames that the channel buffer queue can receive.

continues on next page

Table 12.5 – continued from previous page

Parameter		Description
	ChnFrt	Channel frame rate. Channel playback control can be reflected by this value.
	ChnGap(us)	Channel frame spacing. Inversely proportional to the channel frame rate. Unit: us.
	DispPts	The timestamp of the frame currently being displayed. Unit: us.
	PreDonePts	The timestamp of the previous completed display frame. Unit: us.

12.17 VPSS

【Debug Information】

```
# cat /proc/cvitek/vpss
```

Module: [VPSS], Build Time[#1 SMP PREEMPT Wed Feb 24 15:02:47 CST 2021]

-----MODULE PARAM-----

vpss_vb_source	vpss_split_node_num
0	1

-----VPSS MODE-----

vpss_mode	dev0	dev1	input_mem
single		N	

-----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							

(continues on next page)

(continued from previous page)

-----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					
# 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		

(continues on next page)

(continued from previous page)

# 0	# 2	N	N	0	0
0	0	0	0	0	0
# 0	# 3	N	N	0	0
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	BufLineWrap	BufSize	
# 0	# 0	N	0	0	
# 0	# 1	N	0	0	
# 0	# 2	N	0	0	
# 0	# 3	N	0	0	

【Analysis】

Record the current VPSS attribute configuration and status information.

【Parameter Description】

Parameter		Decription
MODULE PARAM	vpss_vb_source	Video cache pool type. 0: public VB 1: Reserve 2: UserVB
	vpss_split_node_num	The number of block nodes.
VPSS MODE	vpss_mode	VPSS mode Single or Dual mode.
	dev0	The input source of VPSS dev0. ISP or memory.
	dev1	The input source of VPSS dev1. ISP or memory. In single mode, only dev1 is available.
VPSS GRP ATTR	GrpID	GRP ID. Valid Range: [0,VPSS_MAX_GRP_NUM)。
	MaxW	Group input image maximum width.
	MaxH	Group input image maximum height.
	PixFmt	Group input image pixel for- mat.
	SrcFRate	GRP source frame rate.
	DstFRate	GRP target frame rate.
	Dev	The hardware dev number used by Group, which is invalid in single mode. By default, it shows 0.

continues on next page

Table 12.6 – continued from previous page

Parameter		Description
VPSS CHN ATTR	GrpID	GRP ID. Valid range: [0, VPSS_MAX_GRP_NUM)。
	PhyChnID	Physical channel ID number. Valid range: [0, VPSS_MAX_PHY_CHN_NUM)。
	Enable	Whether the channel is enabled. N: Shut down; Y: Open.
	MirrorEn	mirror enable N: Shut down; Y: Open.
	FlipEn	flip enable N: Shut down; Y: Open.
	SrcFRate	Channel frame rate control: source frame rate.
	DstFRate	Channel frame rate control: target frame rate.
	Depth	The length of the queue for the user to obtain the channel image.
	Aspect	Amplitude shape ratio mode. NONE: turn off the amplitude shape ratio AUTO: automatic mode MANUAL: manual mode
	videoX	X coordinate of video position. Manual mode is valid.
	videoY	Y coordinate of video position. Manual mode is valid.
	videoW	Video width. Manual mode is valid.
	videoH	Video height. Manual mode is valid.
	BgColor	Aspect ratio background color. Valid range: [0x0, 0xFFFFFFFF]
VPSS GRP CROP INFO	GrpID	GRP ID number. Valid range: [0, VPSS_MAX_GRP_NUM)。
	CropEn	Whether to enable CROP function. N: Shut down; Y: Open.
	CoorType	Coordinate type. RAT: relative coordinate; ABS: absolute coordinates.
	CoorX	Horizontal initial coordinates.
	CoorY	Vertical initial coordinates.
	Width	CROP RECT width. This cannot exceed the maximum image width.
	Height	CROP RECT height. This cannot exceed the maximum image height.

continues on next page

Table 12.6 – continued from previous page

Parameter		Description
VPSS CHN CROP INFO	GrpID	ID number. Valid range: [0, VPSS_MAX_GRP_NUM)。
	ChnID	CHN ID number. Valid range: [0, VPSS_MAX_PHY_CHN_NUM)。
	CropEn	Whether to enable CROP function. N: Shut down; Y: Open.
	CoorType	Coordinate type. RAT: relative coordinate; ABS: absolute coordinates.
	CoorX	Horizontal initial coordinates.
	CoorY	Vertical initial coordinates.
	Width	CROP RECT width. This cannot exceed the maximum image width.
	Height	CROP RECT height. This cannot exceed the maximum image height.
VPSS GRP WORKSTATUS	GrpID	GRP ID number. Valid range: [0, VPSS_MAX_GRP_NUM)。
	RecvCnt	Number of images received
	LostCnt	The number of images discarded because the queue is full.
	StartFailCnt	The number of Start task failures.
	bStart	Whether to start receiving images.
	CostTime(us)	The time taken to complete the current task.
	MaxCostTime(us)	The execution time of the longest task in history.
	HwCostTime(us)	The current hardware processing time for the completed task.
	HwMaxCostTime(us)	The execution time of the longest historical task in terms of hardware processing time.
VPSS CHN OUTPUTRESOLUTION	GrpID	GRP ID number. Valid range:[0, VPSS_MAX_GRP_NUM)。
	ChnID	Channel ID number. Valid range: [0, VPSS_MAX_PHY_CHN_NUM)。
	Enable	Whether to enable the channel. N: Shut down; Y: Open.
	Width	The width of the target image in pixels.
	Height	The height of the target image in pixels.

continues on next page

Table 12.6 – continued from previous page

Parameter		Description
	Pixfmt	The pixel format of the target image.
	Videofmt	The video format of the target image.
	SendOK	Number of images successfully sent.
	FrameRate	Real time frame rate of channel output.
VPSS CHN ROTATEINFO	GrpID	GRP ID number. Valid range: [0, VPSS_MAX_GRP_NUM)。
	ChnID	channel ID number. Valid range: [0, VPSS_MAX_PHY_CHN_NUM)。
	Rotate	Enumeration of rotation angles.
VPSS CHN LDC INFO	GrpID	GRP ID number. Valid range: [0, VPSS_MAX_GRP_NUM)。
	ChnID	channel ID number. Valid range: [0, VPSS_MAX_PHY_CHN_NUM)。
	Enable	LDC switch. N: Shut down; Y: open
	Aspect	Whether to maintain the aspect ratio. N: not to maintain the aspect ratio; Y: maintain the aspect ratio
	XRatio	Effective when aspect ratio is not maintained. The Crop scale in the horizontal direction.
	YRatio	Effective when aspect ratio is not maintained. The Crop scale in the vertical direction.
	XYRatio	Effective when aspect ratio is maintained. Overall Crop scale in horizontal and vertical directions.
	XOffset	Correct the X-coordinate offset of the center point.
	YOffset	Correct the Y-coordinate offset of the center point.
	DistortionRatio	Coefficient of correction intensity.
DRV_WORK_STATUSDriver work status	dev	Vpss hardware device number
	IspTrigCnt0	Online ISP trigger count for vpss, for sensor0 image.
	IspTrigCnt1	Online ISP trigger count for vpss, for sensor1 image.

continues on next page

Table 12.6 – continued from previous page

Parameter		Description
	IspTrigFailCnt0	Online ISP trigger failure count for vpss, for sensor0 image.
	IspTrigFailCnt1	Online ISP trigger failure count for vpss, for sensor1 image.
	UserTrigCnt	Offline vpss trigger count.
	UserTrigFailCnt	Offline vpss trigger failure count.
	IrqCnt0	Interrupt count for sensor0.
	IrqCnt1	Interrupt count for sensor1.
VPSS CHN BUF WRAPATTR	GrpID	Group ID number.
	ChnID	Channel ID number
	Enable	Enabled or not.
	BufLineWrapBufSize	Size of Slice buffer.