



CV180X & CV181X Cardv 车载 Turnkey 方案使用手册

Version: 1.0.0

Release date: 2024-05-11

©2022 北京晶视智能科技有限公司
本文件所含信息归北京晶视智能科技有限公司所有。
未经授权，严禁全部或部分复制或披露该等信息。

目录

1	声明	2
2	概述	3
2.1	阅读说明	3
2.2	目录结构	4
3	通用组件	5
3.1	Cmdmng	5
3.1.1	API 参考	5
3.1.1.1	CVI_CMDMNG_SendMqCmd	5
3.1.1.2	CVI_CMDMNG_SendMqCmd_Str	6
3.2	Cvinet	7
3.2.1	API 参考	7
3.2.1.1	CVI_NET_Init	7
3.2.1.2	CVI_NET_DeInit	8
3.2.1.3	CVI_NET_AddCgiResponse	9
3.2.1.4	CVI_NET_SetVideoPath	9
3.2.1.5	CVI_NET_RegisterCgiCmd	10
3.2.1.6	CVI_NET_RegisterCgiCmd_CGI	11
3.2.1.7	CVI_NET_AddResponse	12
3.2.1.8	CVI_NET_SetConnetcFlage	12
3.2.2	数据类型	13
3.2.2.1	CGI_CMD_MAX_LEN	13
3.2.2.2	CVI_NET_SYSCALL_CMD_TO_CALLBACK	14
3.2.2.3	CVI_NET_WIFIAPPMAPTO_S	14
3.2.2.4	CVI_NET_WIFIAPPMAPTO_CGI_S	15
3.3	Devnmng	15
3.3.1	API 参考	15
3.3.1.1	CVI_GAUGEMNG_GetPercentage	16
3.3.1.2	CVI_GAUGEMNG_RegisterEvent	17
3.3.1.3	CVI_GAUGEMNG_DeInit	17
3.3.1.4	CVI_GAUGEMNG_init	18
3.3.1.5	CVI_GAUGEMNG_GetBatteryLevel	19
3.3.1.6	CVI_GAUGEMNG_GetChargeState	19
3.3.1.7	CVI_GSENSORMNG_RegisterEvent	20
3.3.1.8	CVI_GSENSORMNG_Init	21
3.3.1.9	CVI_GSENSORMNG_SetSensitivity	22
3.3.1.10	CVI_GSENSORMNG_MenuSetSensitivity	22
3.3.1.11	CVI_GSENSORMNG_GetAttr	23
3.3.1.12	CVI_GSENSORMNG_SetAttr	24
3.3.1.13	CVI_GSENSORMNG_OpenInterrupt	24

3.3.1.14	CVI_GSENSORMNG_DeInit	25
3.3.1.15	CVI_KEYMNG_RegisterEvent	26
3.3.1.16	CVI_KEYMNG_Init	26
3.3.1.17	CVI_KEYMNG_DeInit	27
3.3.1.18	CVI_LEDMNG_Init	28
3.3.1.19	CVI_LEDMNG_DeInit	28
3.3.1.20	CVI_LEDMNG_Control	29
3.3.1.21	CVI_WATCHDOGMNG_Init	30
3.3.1.22	CVI_WATCHDOGMNG_DeInit	30
3.3.1.23	CVI_WIFIMNG_Start	31
3.3.1.24	CVI_WIFIMNG_Stop	32
3.3.2	数据类型	32
3.3.2.1	CVI_GAUGEMNG_CFG_S	33
3.3.2.2	CVI_GSENSORMNG_ATTR_S	33
3.3.2.3	CVI_GSENSORMNG_CFG_S	34
3.3.2.4	CVI_KEYMNG_KEY_NUM_EACH_GRP	34
3.3.2.5	CVI_KEYMNG_KEY_TYPE_E	35
3.3.2.6	CVI_KEYMNG_KEY_CLICK_ATTR_S	35
3.3.2.7	CVI_KEYMNG_KEY_HOLD_ATTR_S	36
3.3.2.8	CVI_KEYMNG_KEY_ATTR_S	36
3.3.2.9	CVI_KEYMNG_KEY_CFG_S	37
3.3.2.10	CVI_KEYMNG_GRP_KEY_CFG_S	37
3.3.2.11	CVI_KEYMNG_CFG_S	38
3.4	Filemng	38
3.4.1	API 参考	38
3.4.1.1	CVI_FILEMNG_Init	39
3.4.1.2	CVI_FILEMNG_RegisterEvent	40
3.4.1.3	CVI_FILEMNG_DeInit	41
3.4.1.4	CVI_FILEMNG_SetDiskState	41
3.4.1.5	CVI_FILEMNG_CheckDiskSpace	42
3.4.1.6	CVI_FILEMNG_AddFile	43
3.4.1.7	CVI_FILEMNG_RemoveFile	43
3.4.1.8	CVI_FILEMNG_SetSearchScope	44
3.4.1.9	CVI_FILEMNG_GetFileObjCnt	45
3.4.1.10	CVI_FILEMNG_GetFileByIndex	46
3.4.1.11	CVI_FILEMNG_GetFileInfoByName	46
3.4.1.12	CVI_FILEMNG_SpacemonitorCheckSpace	47
3.4.1.13	CVI_FILEMNG_GeneratePhotoName	48
3.4.1.14	CVI_FILEMNG_GenerateRecordName	49
3.4.1.15	CVI_FILEMNG_RenameMovToEmr	49
3.4.1.16	CVI_FILEMNG_FileCoverStatus	50
3.4.1.17	CVI_FILEMNG_AlignRecordFileSize	51
3.4.1.18	CVI_FILEMNG_CreateSDConfigFile	52
3.4.1.19	CVI_FILEMNG_RecoverAddFileName	52
3.4.1.20	CVI_FILEMNG_RecoverRemoveFileName	53
3.4.1.21	CVI_FILEMNG_SDConfigFileIsExist	54
3.4.1.22	CVI_FILEMNG_PreallocateState	54
3.4.1.23	CVI_FILEMNG_GetDirType	55
3.4.1.24	CVI_FILEMNG_SetRemoveLoop	56
3.4.2	数据类型	56

3.4.2.1	CVI_APPCOMM_MAX_PATH_LEN	57
3.4.2.2	CVI_FILEMNG_DTCF_MAX_PHOTO_DIR	57
3.4.2.3	CVI_FILEMNG_DTCF_CFG_S	57
3.4.2.4	CVI_FILEMNG_COMM_CFG_S	59
3.4.2.5	CVI_FILEMNG_FILE_TYPE_E	59
3.4.2.6	CVI_FILEMNG_FILE_INFO_S	60
3.5	Ispexposure	60
3.5.1	API 参考	60
3.5.1.1	CVI_ISPEXP_Init	61
3.5.1.2	CVI_ISPEXP_DeInit	61
3.6	Liveviewmng	62
3.6.1	API 参考	62
3.6.1.1	CVI_LIVEVIEWMNG_Switch	62
3.6.1.2	CVI_LIVEVIEWMNG_MoveUp	63
3.6.1.3	CVI_LIVEVIEWMNG_MoveDown	64
3.6.1.4	CVI_LIVEVIEWMNG_Mirror	64
3.6.1.5	CVI_LIVEVIEWMNG_Filp	65
3.7	Photomng	66
3.7.1	API 参考	66
3.7.1.1	CVI_PHOTOMNG_ContCallBack	66
3.7.1.2	CVI_POHTOMNG_RegisterEvent	67
3.8	Playbackmng	67
3.8.1	API 参考	68
3.8.1.1	CVI_PLAYBACKMNG_EventCallBack	68
3.8.1.2	CVI_PLAYBACKMNG_RegisterEvent	69
3.8.2	数据类型	69
3.8.2.1	CVI_PLAYER_SERVICE_EVENT_TYPE_E	70
3.8.2.2	CVI_PLAYER_SERVICE_EVENT_S	71
3.9	Powercontrol	71
3.9.1	API 参考	71
3.9.1.1	CVI_POWERCTRL_Init	72
3.9.1.2	CVI_POWERCTRL_GetTaskAttr	72
3.9.1.3	CVI_POWERCTRL_SetTaskAttr	73
3.9.1.4	CVI_POWERCTRL_EventPreProc	74
3.9.1.5	CVI_POWERCTRL_DeInit	75
3.9.2	数据类型	75
3.9.2.1	CVI_PWRCTRL_TASK_PROC_CALLBACK	76
3.9.2.2	CVI_PWRCTRL_TASK_CFG_S	76
3.9.2.3	CVI_PWRCTRL_CFG_S	77
3.9.2.4	CVI_PWRCTRL_TASK_E	77
3.9.2.5	CVI_PWRCTRL_EVENT_TYPE_E	78
3.9.2.6	CVI_PWRCTRL_WAKEUP_TACTICS_E	78
3.9.2.7	CVI_PWRCTRL_EVENT_SCOPE_E	79
3.9.2.8	CVI_PWRCTRL_EVENT_COMMON_ATTR_S	80
3.9.2.9	CVI_PWRCTRL_EVENT_WAKEUP_ATTR_S	80
3.9.2.10	CVI_PWRCTRL_EVENT_CONTROL_E	81
3.9.2.11	CVI_PWRCTRL_EVENT_CONTROL_ATTR_S	81
3.9.2.12	CVI_PWRCTRL_EVENT_ATTR_S	82
3.10	Recordmng	82
3.10.1	API 参考	82

3.10.1.1	CVI_RECORDMNG_EventCallBack	83
3.10.1.2	CVI_RECORDMNG_ContCallBack	83
3.10.1.3	CVI_RECORDMNG_RegisterEvent	84
3.11	Storagemng	85
3.11.1	API 参考	85
3.11.1.1	CVI_STORAGEMNG_RegisterEvent	85
3.11.1.2	CVI_STORAGEMNG_Create	86
3.11.1.3	CVI_STORAGEMNG_Destroy	87
3.11.1.4	CVI_STORAGEMNG_GetFSInfo	87
3.11.1.5	CVI_STORAGEMNG_GetInfo	88
3.11.1.6	CVI_STORAGEMNG_Format	89
3.11.1.7	CVI_STORAGEMNG_Mount	89
3.11.1.8	CVI_STORAGEMNG_Umount	90
3.12	System	91
3.12.1	API 参考	91
3.12.1.1	CVI_SYSTEM_SetDateTime	91
3.12.1.2	CVI_SYSTEM_GetRTCDateTime	92
3.12.1.3	CVI_SYSTEM_SetDefaultDateTime	93
3.12.1.4	CVI_SYSTEM_Reboot	93
3.12.1.5	CVI_SYSTEM_BootSound	94
3.12.1.6	CVI_SYSTEM_GetStartupWakeupSource	95
3.12.1.7	CVI_SYSTEM_SetGpioWakeup	95
3.12.2	数据类型	96
3.12.2.1	CVI_SYSTEM_STARTUP_SRC_E	96
3.12.2.2	CVI_SYSTEM_TM_S	97
3.13	Utils	97
3.13.1	API 参考	98
3.13.1.1	CVI_MD5_Init	98
3.13.1.2	CVI_MD5_Update	99
3.13.1.3	CVI_MD5_Final	99
3.13.1.4	CVI_MD5_Transform	100
3.13.1.5	CVI_MD5_Encode	101
3.13.1.6	CVI_MD5_Decode	101
3.13.1.7	CVI_TIMEDTASK_DeInit	102
3.13.1.8	CVI_TIMEDTASK_init	103
3.13.1.9	CVI_TIMEDTASK_Create	104
3.13.1.10	CVI_TIMEDTASK_Destroy	104
3.13.1.11	CVI_TIMEDTASK_GetAttr	105
3.13.1.12	CVI_TIMEDTASK_SetAttr	106
3.13.1.13	CVI_TIMEDTASK_ResetTime	107
3.13.2	数据类型	107
3.13.2.1	CVI_MD5_CTX_S	108
3.13.2.2	CVI_TIMEDTASK_ATTR_S	108
3.13.2.3	CVI_TIMEDTASK_CFG_S	109
3.14	Videomd	109
3.14.1	API 参考	109
3.14.1.1	CVI_MOTION_DETECT_SetState	110
3.14.1.2	CVI_MOTION_DETECT_Init	110
3.14.1.3	CVI_MOTION_DETECT_DeInit	111
3.14.2	数据类型	112

3.14.2.1	CVI_MOTION_DETECT_ATTR_S	112
3.15	Volmng	113
3.15.1	API 参考	113
3.15.1.1	CVI_VOICEPLAY_Init	113
3.15.1.2	CVI_VOICEPLAY_DeInit	114
3.15.1.3	CVI_VOICEPLAY_Push	114
3.15.1.4	CVI_VOICEPLAY_SetAmplifier	115
3.15.1.5	CVI_VOICEPLAY_SetAmplifierFlage	116
3.15.1.6	CVI_VOICEPLAY_SetVolume	117
3.15.1.7	CVI_VOICEPLAY_GetVolume	117
3.15.2	数据类型	118
3.15.2.1	CVI_VOICE_MAX_SEGMENT_CNT	118
3.15.2.2	CVI_VOICEPLAY_VOICE_S	119
3.15.2.3	CVI_VOICEPLAY_VOICETABLE_S	119
3.15.2.4	CVI_VOICEPLAY_AOUT_OPT_S	120
3.15.2.5	CVI_VOICEPLAY_CFG_S	120
3.16	ADASmng	121
3.16.1	API 参考	121
3.16.1.1	CVI_ADASMNG_VoiceCallback	121
3.16.1.2	CVI_ADASMNG_LabelCallback	122
3.16.1.3	CVI_ADASMNG_RegisterEvent	123
3.16.2	数据类型	123
3.16.2.1	CVI_EVENT_ADASMNG_E	124
3.16.2.2	CVI_ADASMNG_CMD_E	124
3.17	Speechmng	125
3.17.1	API 参考	125
3.17.1.1	CVI_SPEECHMNG_Init	125
3.17.1.2	CVI_SPEECHMNG_DeInit	126
3.17.1.3	CVI_SPEECHMNG_StartSpeech	127
3.17.1.4	CVI_SPEECHMNG_StopSpeech	127
3.17.2	数据类型	128
3.17.2.1	CVI_EVENT_SPEECHMNG_E	128
3.17.2.2	CVI_SPEECHMNG_CMD_E	129
4	Dashcam	131
4.1	参数管理	131
4.1.1	目录结构	131
4.1.2	双系统参数流程设计	132
4.1.2.1	Flash 分区	132
4.1.2.2	参数加载流程	134
4.1.2.3	参数读写方案	135
4.2	资源管理	135
4.2.1	目录结构	135
4.2.2	定制化资源	136
4.2.2.1	定制化声音	136
4.2.2.2	定制化 LogoOSD	136
4.3	媒体管理	136
4.3.1	目录结构	137
4.3.2	开发参考	137
4.3.2.1	媒体通用 API	137

4.3.2.1.1	CVI_MEDIA_StartAudioInTask	139
4.3.2.1.2	CVI_MEDIA_StopAudioInTask	139
4.3.2.1.3	CVI_MEDIA_VideoInit	140
4.3.2.1.4	CVI_MEDIA_VideoDeInit	141
4.3.2.1.5	CVI_MEDIA_VcapInit	141
4.3.2.1.6	CVI_MEDIA_VcapDeInit	142
4.3.2.1.7	CVI_MEDIA_VbInit	143
4.3.2.1.8	CVI_MEDIA_VbInitPlayBack	143
4.3.2.1.9	CVI_MEDIA_VbDeInit	144
4.3.2.1.10	CVI_MEDIA_DispInit	145
4.3.2.1.11	CVI_MEDIA_DispDeInit	145
4.3.2.1.12	CVI_MEDIA_LiveViewSerInit	146
4.3.2.1.13	CVI_MEDIA_LiveViewSerDeInit	147
4.3.2.1.14	CVI_MEDIA_AiInit	147
4.3.2.1.15	CVI_MEDIA_VencInit	148
4.3.2.1.16	CVI_MEDIA_VencDeInit	149
4.3.2.1.17	CVI_MEDIA_AiDeInit	149
4.3.2.1.18	CVI_MEDIA_AencInit	150
4.3.2.1.19	CVI_MEDIA_AencDeInit	151
4.3.2.1.20	CVI_MEDIA_RecordSerInit	151
4.3.2.1.21	CVI_MEDIA_RecordSerDeInit	152
4.3.2.1.22	CVI_MEDIA_RtspSerInit	153
4.3.2.1.23	CVI_MEDIA_RtspSerDeInit	153
4.3.2.1.24	CVI_MEDIA_AoInit	154
4.3.2.1.25	CVI_MEDIA_AoDeInit	155
4.3.2.1.26	CVI_MEDIA_PlayBackSerInit	155
4.3.2.1.27	CVI_MEDIA_PlayBackSerDeInit	156
4.3.2.1.28	CVI_MEDIA_Res2RecordMediaMode	157
4.3.2.1.29	CVI_MEDIA_Res2PhotoMediaMode	157
4.3.2.1.30	CVI_MEDIA_SetAntiFlicker	158
4.3.2.1.31	CVI_MEDIA_Is_CameraEnabled	159
4.3.2.1.32	CVI_MEDIA_GetCtx	160
4.3.2.1.33	CVI_MEDIA_VIDEOMD_Init	160
4.3.2.1.34	CVI_MEDIA_VIDEOMD_DeInit	161
4.3.2.1.35	CVI_MEDIA_APP_RTSP_Init	162
4.3.2.1.36	CVI_MEDIA_APP_RTSP_DeInit	162
4.3.2.1.37	CVI_MEDIA_SwitchRTSPChanel	163
4.3.2.1.38	CVI_MEDIA_PhotoVprocInit	164
4.3.2.1.39	CVI_MEDIA_PhotoVprocDeInit	164
4.3.2.1.40	CVI_MEDIA_PhotoSerInit	165
4.3.2.1.41	CVI_MEDIA_PhotoSerDeInit	166
4.3.2.1.42	CVI_MEDIA_SensorDet	166
4.3.2.1.43	CVI_MEDIA_ADASInit	167
4.3.2.1.44	CVI_MEDIA_ADASDeInit	168
4.3.2.1.45	CVI_MEDIA_QRCodeInit	168
4.3.2.1.46	CVI_MEDIA_QRCodeDeInit	169
4.3.2.2	数据结构	170
4.3.2.2.1	MAX_CAMERA_INSTANCES	170
4.3.2.2.2	MAX_DEV_INSTANCES	171
4.3.2.2.3	MAX_RTSP_CNT	171

4.3.2.2.4	MAX_VPROC_CNT	172
4.3.2.2.5	MAX_VENC_CNT	172
4.3.2.2.6	CVI_MEDIA_SYSHANDLE_S	172
4.3.2.2.7	CVI_MEDIA_CAMERA_SERVICE_S	173
4.3.2.2.8	CVI_MEDIA_PARAM_INIT_S	174
4.3.3	模块分析	175
4.3.3.1	典型媒体通路初始化	175
4.3.3.2	典型媒体通路重置处理	177
4.4	状态管理	178
4.4.1	模块上下文	178
4.4.2	模块依赖	178
4.4.2.1	StateMachine 模块	178
4.4.2.2	EventHub 模块	179
4.4.3	模块分析	179
4.4.3.1	子模块划分	179
4.4.3.2	典型业务状态	180
4.4.3.3	典型场景分析	181
4.4.4	模式管理	181
4.4.4.1	目录结构	181
4.4.4.2	API 参考	182
4.4.4.2.1	CVI_MODEMNG_Init	183
4.4.4.2.2	CVI_MODEMNG_Deinit	184
4.4.4.2.3	CVI_MODEMNG_SendMessage	184
4.4.4.2.4	CVI_MODEMNG_GetCurMode	185
4.4.4.2.5	CVI_MODEMNG_GetCurWorkMode	186
4.4.4.2.6	CVI_MODEMNG_GetModeState	186
4.4.4.2.7	CVI_MODEMNG_SetModeState	187
4.4.4.2.8	CVI_MODEMNG_SetEmrState	188
4.4.4.2.9	CVI_MODEMNG_LiveViewUp	188
4.4.4.2.10	CVI_MODEMNG_LiveViewDown	189
4.4.4.2.11	CVI_MODEMNG_GetCardState	190
4.4.4.2.12	CVI_MODEMNG_GetCamStatus	191
4.4.4.2.13	CVI_MODEMNG_GetCamIspInfoStatus	191
4.4.4.2.14	CVI_MODEMNG_SetCardState	192
4.4.4.2.15	CVI_MODEMNG_GetInProgress	193
4.4.4.2.16	CVI_MODEMNG_GetEmrState	193
4.4.4.2.17	CVI_MODEMNG_TEST_MAIN_Create	194
4.4.4.2.18	CVI_MODEMNG_TEST_MAIN_Destroy	195
4.4.4.2.19	CVI_MODEMNG_SetMediaLoopTime	195
4.4.4.2.20	CVI_MODEMNG_SetMediaAudio	196
4.4.4.2.21	CVI_MODEMNG_SetMenuWifiStatus	197
4.4.4.2.22	CVI_MODEMNG_SetMediaOsd	197
4.4.4.2.23	CVI_MODEMNG_SetMediaLapseTime	198
4.4.4.2.24	CVI_MODEMNG_SetMediaVencFormat	199
4.4.4.2.25	CVI_MODEMNG_LiveViewSwitch	199
4.4.4.2.26	CVI_MODEMNG_RegisterEvent	200
4.4.4.2.27	CVI_MODEMNG_Format	201
4.4.4.2.28	CVI_MODEMNG_OpenUvcMode	201
4.4.4.2.29	CVI_MODEMNG_CloseUvcMode	202
4.4.4.2.30	CVI_MODEMNG_ContextInit	203

4.4.4.2.31	CVI_MODEMNG_SetDefaultParam	203
4.4.4.2.32	CVI_MODEMNG_OpenPhotoMode	204
4.4.4.2.33	CVI_MODEMNG_ClosePhotoMode	205
4.4.4.2.34	CVI_MODEMNG_SetParkingRec	205
4.4.4.3	数据结构	206
4.4.4.3.1	CVI_MODEMNG_EXIT_MODE_E	206
4.4.4.3.2	CVI_MODEMNG_EXIT_MODE_CALLBACK	207
4.4.4.3.3	CVI_MODEMNG_CONFIG_S	207
4.5	网络管理	208
4.5.1	工作流程	208
4.5.2	http 请求构成	209
4.5.3	开发参考	210
4.5.3.1	网络管理通用 API	210
4.5.3.1.1	CVI_NETCTRL_Init	212
4.5.3.1.2	CVI_NETCTRL_DeInit	213
4.5.3.1.3	CVI_NETCTRL_NetToUiConnectState	214
4.5.3.1.4	CVI_XML_GetPicture	214
4.5.3.1.5	CVI_XML_SetCapture	215
4.5.3.1.6	CVI_XML_SetMovieRecordBitrate	216
4.5.3.1.7	CVI_XML_SetMovieLiveViewBitrate	217
4.5.3.1.8	CVI_XML_ChangeLiveviewSize	218
4.5.3.1.9	CVI_XML_EdisableMoviehdr	219
4.5.3.1.10	CVI_XML_EdisableMotionDetection	220
4.5.3.1.11	CVI_XML_GetFreePicture	221
4.5.3.1.12	CVI_XML_StartandStopMovieRecord	222
4.5.3.1.13	CVI_XML_SetMovieRecord	223
4.5.3.1.14	CVI_XML_SetCyclicRecordValue	224
4.5.3.1.15	CVI_XML_SetMovieEvValue	225
4.5.3.1.16	CVI_XML_EdisableMovieAudio	226
4.5.3.1.17	CVI_XML_EdisableDateInprint	227
4.5.3.1.18	CVI_XML_GetMovieMaxRecordTime	228
4.5.3.1.19	CVI_XML_EDisableMovieGSensorSensitivity	229
4.5.3.1.20	CVI_XML_SetAutoRecording	230
4.5.3.1.21	CVI_XML_StartandStopMovieLiveview	231
4.5.3.1.22	CVI_XML_GetMovieRecordingTime	232
4.5.3.1.23	CVI_XML_TriggerRAWencode	233
4.5.3.1.24	CVI_XML_GetRAWencodeJPEG	234
4.5.3.1.25	CVI_XML_GetStreamingAddr	235
4.5.3.1.26	CVI_XML_SetMovieFliphorMirror	236
4.5.3.1.27	CVI_XML_ChangeSystemMode	237
4.5.3.1.28	CVI_XML_QueryCurrentSupportCommand	238
4.5.3.1.29	CVI_XML_SetWiFiSSID	239
4.5.3.1.30	CVI_XML_SetWiFiPassPhrase	240
4.5.3.1.31	CVI_XML_SetSystemDate	241
4.5.3.1.32	CVI_XML_SetSystemTime	242
4.5.3.1.33	CVI_XML_SelectLanguage	243
4.5.3.1.34	CVI_XML_CommandWouldFormatStorage	244
4.5.3.1.35	CVI_XML_ResetallSettingTodeFault	245
4.5.3.1.36	CVI_XML_GetProjectVersion	246
4.5.3.1.37	CVI_XML_StartTouupdateFirmware	247

4.5.3.1.38	CVI_XML_QueryAllSettingCommandStatus	248
4.5.3.1.39	CVI_XML_ListAllFileCreatTime	249
4.5.3.1.40	CVI_XML_CommandSmartPhoneCheck	250
4.5.3.1.41	CVI_XML_GetDiskFreeSpace	251
4.5.3.1.42	CVI_XML_ReconnectWiFi	252
4.5.3.1.43	CVI_XML_SaveMenuInformation	253
4.5.3.1.44	CVI_XML_GetCardsSatus	254
4.5.3.1.45	CVI_XML_GetUpdateFWpath	255
4.5.3.1.46	CVI_XML_SetUpPIPStyle	256
4.5.3.1.47	CVI_XML_GetSSIDandPassPhrase	257
4.5.3.1.48	CVI_XML_GetMovieSizeCapacity	258
4.5.3.1.49	CVI_XML_QueryMenuItem	259
4.5.3.1.50	CVI_XML_QueryCurrentDVRmode	260
4.5.3.1.51	CVI_XML_GetThumbnail	261
4.5.3.1.52	CVI_XML_DeleteOneFile	262
4.5.3.1.53	CVI_XML_DeleteAll	263
4.5.3.1.54	CVI_XML_GetMovieFileInformation	264
4.5.3.1.55	CVI_XML_RoadCamStart	265
4.5.3.1.56	CVI_XML_PhoneApp	266
4.5.3.1.57	CVI_XML_SensorNum	267
4.5.3.1.58	CVI_CGI_SetMovieRecord	268
4.5.3.1.59	CVI_CGI_SetCyclicRecordValue	269
4.5.3.1.60	CVI_CGI_SetGSensorSensitivity	270
4.5.3.1.61	CVI_CGI_SetVideoTimeLapse	271
4.5.3.1.62	CVI_CGI_SetMovieEvValue	272
4.5.3.1.63	CVI_CGI_CommandWouldFormatStorage	273
4.5.3.1.64	CVI_CGI_DeleteOneFile	274
4.5.3.1.65	CVI_CGI_SetGetSSIDandPassPhrase	275
4.5.3.1.66	CVI_CGI_SetSSIDandPassPhrase	276
4.5.3.1.67	CVI_CGI_ResetAllSettingTodeFault	277
4.5.3.1.68	CVI_CGI_GetProjectVersion	277
4.5.3.1.69	CVI_CGI_GetMenuUUID	278
4.5.3.1.70	CVI_CGI_GetMJPEGStatus	279
4.5.3.1.71	CVI_CGI_GetPreviewStatus	280
4.5.3.1.72	CVI_CGI_GetCamMenu	281
4.5.3.1.73	CVI_CGI_SetEdisableMovieAudio	282
4.5.3.1.74	CVI_CGI_GetEdisableMovieAudio	283
4.5.3.1.75	CVI_CGI_Heartbeat	284
4.5.3.1.76	CVI_CGI_MovieRecord	285
4.5.3.1.77	CVI_CGI_CardInfo	286
4.5.3.1.78	CVI_CGI_VideoresCapability	287
4.5.3.1.79	CVI_CGI_ListAllFileCreatTime	288
4.5.3.1.80	CVI_CGI_GetScreenrail	289
4.5.3.1.81	CVI_CGI_SetSystemTime	290
4.6	usb 控制	291
4.6.1	模块 API	291
4.6.1.1	CVI_USBCTRL_Init	291
4.6.1.2	CVI_USBCTRL_Deinit	292
4.6.1.3	CVI_USBCTRL_RegisterEvent	293
4.7	UI	293

4.7.1	模块上下文	293
4.7.2	目录结构	294
4.8	INIT	295
4.8.1	目录结构	295
4.8.2	模块分析	296
4.8.2.1	Alios 端	296
4.8.2.2	Linux 端	297
5	流媒体后视镜	298
5.1	参数管理	298
5.1.1	ini 文件分类	298
5.1.2	ini 文件解析	301
5.1.2.1	config_devmng.ini	301
5.1.2.2	config_filemng.ini	302
5.1.2.3	config_menu.in	302
5.1.2.4	config_media_comm.ini	304
5.1.2.5	config_mediamode_cam0_comm.ini	308
5.1.2.6	config_mediamode_cam1_comm.ini	308
5.1.2.7	config_workmode_photo.ini	308
5.1.2.8	config_media_cam0_photo_1440p.ini	309
5.1.3	Video pipeline	314
5.2	状态管理	314
5.2.1	业务状态	315
5.2.2	Base 状态处理消息	316
5.3	UI	316
5.3.1	目录结构	316
5.3.2	API 参考	317
5.3.2.1	CVI_UIAPP_Start	317
5.3.2.2	CVI_UIAPP_Stop	318
5.3.3	UI 功能菜单	318
5.3.4	UI 事件划分	321
5.3.5	典型事件	322
5.3.5.1	休眠管理	322

修订记录

Revision	Date	Description
1.0.0	2024/05/11	初版

1 声明



法律声明

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

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

联系我们

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

深圳市宝安区福海街道展城社区会展湾云岸广场 T10 栋

电话 +86-10-57590723 +86-10-57590724

邮编 100094（北京）518100（深圳）

官方网站 <https://www.sophgo.com/>

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

2 概述

2.1 阅读说明

该文档为 CVITEK 车载产品 Turnkey 方案开发指南，该开发指南旨在讲述 APP 层，如图 2-1 所示。

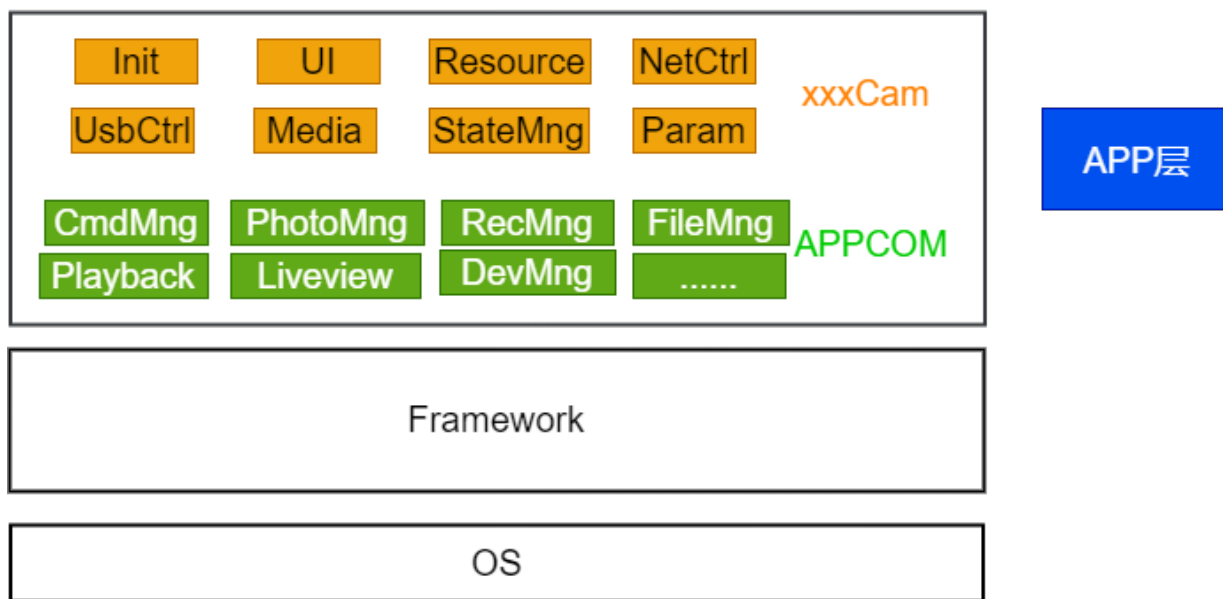


图 2.1: APP 层模块框架

模块说明：

- OS：提供基本的操作系统及基础的外围器件驱动。
- Framework：提供媒体采集、处理、编码功能以及基于业务的接口抽象层，详情参考文档 Framework 通用软件开发指南。
- APP 层：具体的业务实现，又分为两大部分：
 - APPCOM：通用组件，可用于不同的产品，如行车产品，运动相机。
 - xxxCam：产品层模块。

该文档包含下述主要内容：

- 介绍 APPCOM 的通用组件及 API。
- 介绍产品层模块 Dashcam（行车产品）。
- 介绍行车产品中的流媒体后视镜。

2.2 目录结构

APP 层目录包含如下组成部分：

```

application
├── /build          //Makefile, 指定所需的头文件和库
├── /common         //通用组件，可用于不同的产品
│   ├── /cmdmng     // 管理消息至消息队列
│   ├── /cvinet     //网络管理模块
│   ├── /devmng     //设备管理
│   ├── /filemng    //文件系统管理
│   ├── /ispexposure //isp 曝光
│   ├── /liveviewmng //预览管理
│   ├── /phomng     //拍照模式管理
│   ├── /playbackmng //回放模式管理
│   ├── /powercontrol //电源控制模块
│   ├── /recordmng  //录像管理模块
│   ├── /storagemng //存储设备管理
│   ├── /system     //用于系统设置
│   ├── /utils      //实用工具，包含MD5算法和定时器任务
│   ├── /videomng   //运动检测
│   └── /volmng     //音频
├── /dashcam       //行车产品
│   ├── /modules   // 产品层模块
│   │   ├── /main  // INIT
│   │   ├── /media //媒体管理
│   │   ├── /mode  //状态管理
│   │   ├── /netctrl //网络控制
│   │   ├── /param //参数管理
│   │   ├── /resource //资源管理
│   │   ├── /ui    //ui界面
│   │   └── /usbctrl //usb控制
├── /tool          //工具
│   ├── /awtkres   //awtk工具库，比如生成字库
│   ├── /bmp2bitmap // 将bmp位图转为bitmap
│   ├── /cc_tools  // 终端命令cc_tools,可用于压测和一些无屏操作
│   └── /test_scripts //压测脚本

```

3 通用组件

APP 层的通用组件供产品层模块（xxxCam）使用。

3.1 Cmdmng

该通用组件是用于管理进程间通信的消息队列。

3.1.1 API 参考

该功能模块为用户提供以下 API：

- `CVI_CMDMNG_SendMqCmd`：向客户端的消息队列发送消息。
- `CVI_CMDMNG_SendMqCmd_Str`：向客户端的消息队列发送消息，可额外附带字符串信息。

3.1.1.1 CVI_CMDMNG_SendMqCmd

【描述】

向客户端的消息队列发送消息。

【语法】

```
int32_t CVI_CMDMNG_SendMqCmd(int32_t client_id, int32_t chn_id, int32_t cmd_id, int32_t arg_val);
```

【参数】

参数名称	描述	输入/输出
client_id	客户端 id	输入
chn_id	管道 id	输入
cmd_id	命令 id	输入
arg_val	选项	输入

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_cmdmng.h
- 库文件: libcvi_cmdmng.a/libcvi_cmdmng.so

【注意】

无

【举例】

无

【相关主题】

无

3.1.1.2 CVI_CMDMNG_SendMqCmd_Str

【描述】

向客户端的消息队列发送消息，可额外附带字符串信息。

【语法】

```
int32_t CVI_CMDMNG_SendMqCmd_Str(int32_t client_id, int32_t chn_id, int32_t cmd_id,
    ↪ int32_t arg_val, const char *str);
```

【参数】

参数名称	描述	输入/输出
client_id	客户端 id	输入
chn_id	管道 id	输入
cmd_id	命令 id	输入
arg_val	选项	输入
str	选项	输入

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_cmdmng.h
- 库文件: libcvi_cmdmng.a/libcvi_cmdmng.so

【注意】

无

【举例】

无

【相关主题】

无

3.2 Cvinet

该组件用于网络控制。其作用范围可参考[网络管理](#)。

3.2.1 API 参考

该功能模块为用户提供以下 API:

- `CVI_NET_Init` : 初始化 net, 启动 tthttpd 线程。
- `CVI_NET_DeInit` : 销毁 net, 停止 tthttpd 线程。
- `CVI_NET_AddCgiResponse` : xml 文件生成。
- `CVI_NET_SetVideoPath` : 设置视频读取路径。
- `CVI_NET_RegisterCgiCmd` : 命令回调函数注册。
- `CVI_NET_RegisterCgiCmd_CGI` : 命令回调函数注册, 解析 CGI 指令。
- `CVI_NET_AddResponse` : xml 返回数据包。
- `CVI_NET_SetConnetcFlage` : 网络连接标志。

3.2.1.1 CVI_NET_Init

【描述】

初始化 net, 启动 tthttpd 线程。

【语法】

```
int32_t CVI_NET_Init(void);
```

【参数】

无

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: `cvi_net.h`
- 库文件: `libcvi_net.a/libcvi_net.so`

【注意】

无

【举例】

无

【相关主题】

无

3.2.1.2 CVI_NET_DeInit**【描述】**

销毁 net, 停止 tthttpd 线程

【语法】

```
int32_t CVI_NET_DeInit(void);
```

【参数】

无

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: `cvi_net.h`
- 库文件: `libcvi_net.a/libcvi_net.so`

【注意】

无

【举例】

无

【相关主题】

无

3.2.1.3 CVI_NET_AddCgiResponse

【描述】

xml 文件生成

【语法】

```
int32_t CVI_NET_AddCgiResponse(int32_t len, void *param, char *pszfmt, ...);
```

【参数】

参数名称	描述	输入/输出
param	Thttpd 句柄	输入
len	字符长度	输入
pszfmt	命令字符内容	输入
...	可变参数	输入

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_net.h
- 库文件: libcvi_net.a/libcvi_net.so

【注意】

无

【举例】

无

【相关主题】

无

3.2.1.4 CVI_NET_SetVideoPath

【描述】

设置视频读取路径

【语法】

```
int32_t CVI_NET_SetVideoPath(const char *path);
```

【参数】

参数名称	描述	输入/输出
path	文件路径	输入

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_net.h
- 库文件: libcvi_net.a/libcvi_net.so

【注意】

无

【举例】

无

【相关主题】

无

3.2.1.5 CVI_NET_RegisterCgiCmd**【描述】**

命令回调函数注册

【语法】

```
int32_t CVI_NET_RegisterCgiCmd(CVI_NET_WIFIAPPMAPTO_S *cgi_cmd);
```

【参数】

参数名称	描述	输入/输出
cgi_cmd	Cmd 命令注册	输入

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_net.h
- 库文件: libcvi_net.a/libcvi_net.so

【注意】

无

【举例】

无

【相关主题】

无

3.2.1.6 CVI_NET_RegisterCgiCmd_CGI

【描述】

命令函数回调注册，解析 CGI 指令

【语法】

```
int32_t CVI_NET_RegisterCgiCmd_CGI(CVI_NET_WIFIAPPMAPTO_CGI_S *cgi_cmd);
```

【参数】

参数名称	描述	输入/输出
cgi_cmd	CGI 命令	输入

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件：cvi_net.h
- 库文件：libcvi_net.a/libcvi_net.so

【注意】

无

【举例】

无

【相关主题】

无

3.2.1.7 CVI_NET_AddResponse

【描述】

xml 返回数据包

【语法】

```
int32_t CVI_NET_AddResponse(void *param, char *pszfmt, int32_t len);
```

【参数】

参数名称	描述	输入/输出
param	Thttpd 句柄	输入
pszfmt	数据包内容	输入
len	数据包大小	输入

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_net.h
- 库文件: libcvi_net.a/libcvi_net.so

【注意】

无

【举例】

无

【相关主题】

无

3.2.1.8 CVI_NET_SetConnetcFlage

【描述】

网络连接标志

【语法】

```
void CVI_NET_SetConnetcFlage(int32_t flage);
```

【参数】

参数名称	描述	输入/输出
flage	连接标志	输入

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: `cvi_net.h`
- 库文件: `libcvi_net.a/libcvi_net.so`

【注意】

无

【举例】

无

【相关主题】

无

3.2.2 数据类型

视频输入相关数据类型定义如下:

- `CVI_NET_SYSCALL_CMD_TO_CALLBACK`: 命令回调函数。
- `CVI_NET_WIFIAPPMAPTO_S`: wifi app 命令, 用于整数命令。
- `CVI_NET_WIFIAPPMAPTO_CGI_S`: wifi app 命令, 用于字符串命令。
- `CGI_CMD_MAX_LEN`: 最大字符串长度。

3.2.2.1 CGI_CMD_MAX_LEN

【说明】

最大字符串长度

【定义】

```
#define CGI_CMD_MAX_LEN    (32)
```

【注意事项】

无。

【相关数据类型及接口】

无。

3.2.2.2 CVI_NET_SYSCALL_CMD_TO_CALLBACK

【说明】

命令回调函数

【定义】

```
typedef int32_t (*CVI_NET_SYSCALL_CMD_TO_CALLBACK)(httpd_conn *pram, char *sptr);
```

【成员】

成员名称	描述
pram	网络连接参数
sptr	命令字符串

【注意事项】

无。

【相关数据类型及接口】

无。

3.2.2.3 CVI_NET_WIFIAPPMAPTO_S

【说明】

wifi app 命令，用于整数命令

【定义】

```
typedef struct _WIFIAPPMAPTO {  
    int32_t cmd;  
    CVI_NET_SYSCALL_CMD_TO_CALLBACK callback;  
} CVI_NET_WIFIAPPMAPTO_S;
```

【成员】

成员名称	描述
cmd	整数命令
callback	回调函数

【注意事项】

无。

【相关数据类型及接口】

无。

3.2.2.4 CVI_NET_WIFIAPPMAPTO_CGI_S

【说明】

wifi app 命令，用于字符串命令

【定义】

```
typedef struct _WIFIAPPMAPTO_CGI {
    char cmd[CGI_CMD_MAX_LEN];
    CVI_NET_SYSCALL_CMD_TO_CALLBACK callback;
} CVI_NET_WIFIAPPMAPTO_CGI_S;
```

【成员】

成员名称	描述
cmd	字符串命令
callback	命令回调函数串

【注意事项】

无。

【相关数据类型及接口】

无。

3.3 Devmng

该通用组件用于设备管理，包括电池，加速度传感器，按键，LED，WiFi 和看门狗。

3.3.1 API 参考

该功能模块为用户提供以下 API:

- CVI_GAUGEMNG_GetPercentage：获得电池电量百分比。
- CVI_GAUGEMNG_Init：电池电量管理初始化。
- CVI_GAUGEMNG_GetBatteryLevel：获得电池电量水平。
- CVI_GAUGEMNG_GetChargeState：获取电池是否正在充电。
- CVI_GAUGEMNG_DeInit：电池电量管理反初始化。
- CVI_GAUGEMNG_RegisterEvent：电池电量状态注册。
- CVI_GSENSORMNG_RegisterEvent：加速度传感器状态注册。
- CVI_GSENSORMNG_Init：加速度传感器管理初始化。
- CVI_GSENSORMNG_SetSensitivity：设置传感器灵敏度。
- CVI_GSENSORMNG_MenuSetSensitivity：菜单中设置传感器灵敏度。

- CVI_GSENSORMNG_GetAttr：获取加速度传感器属性。
- CVI_GSENSORMNG_SetAttr：设置加速度传感器属性。
- CVI_GSENSORMNG_OpenInterrupt：打开传感器中断。
- CVI_GSENSORMNG_DeInit：加速度传感器管理反初始化。
- CVI_KEYMNG_RegisterEvent：按键状态注册。
- CVI_KEYMNG_Init：按键管理初始化。
- CVI_KEYMNG_DeInit：按键管理反初始化。
- CVI_LEDMNG_Control：设置 GPIO 状态。
- CVI_LEDMNG_Init：led 灯初始化。
- CVI_LEDMNG_DeInit：led 灯反初始化。
- CVI_WATCHDOGMNG_Init：看门狗管理初始化。
- CVI_WATCHDOGMNG_DeInit：看门狗管理反初始化。
- CVI_WIFIMNG_Start：启动 wifi 化。
- CVI_WIFIMNG_Stop：停止 wifi。

3.3.1.1 CVI_GAUGEMNG_GetPercentage

【描述】

获得电池电量百分比

【语法】

```
int32_t CVI_GAUGEMNG_GetPercentage(void);
```

【参数】

无

【返回值】

返回值	描述
vbat (大于等于 0)	电池电量百分比
-1	失败

【需求】

- 头文件：cvi_gaugemng.h
- 库文件：libcvi_devmng.a/libcvi_devmng.so

【注意】

无

【举例】

无

【相关主题】

无

3.3.1.2 CVI_GAUGEMNG_RegisterEvent**【描述】**

电池电量状态注册

【语法】

```
int32_t CVI_GAUGEMNG_RegisterEvent(void);
```

【参数】

无

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_gaugemng.h
- 库文件: libcvi_devmng.a/libcvi_devmng.so

【注意】

无

【举例】

无

【相关主题】

无

3.3.1.3 CVI_GAUGEMNG_DeInit**【描述】**

电池电量管理反初始化

【语法】

```
int32_t CVI_GAUGEMNG_DeInit(void);
```

【参数】

无

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_gaugemng.h
- 库文件: libcvi_devmng.a/libcvi_devmng.so

【注意】

无

【举例】

无

【相关主题】

无

3.3.1.4 CVI_GAUGEMNG_init

【描述】

电池电量管理初始化

【语法】

```
int32_t CVI_GAUGEMNG_Init(const CVI_GAUGEMNG_CFG_S* pstCfg);
```

【参数】

参数名称	描述	输入/输出
pstCfg	电池电量配置	输入

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_gaugemng.h
- 库文件: libcvi_devmng.a/libcvi_devmng.so

【注意】

无

【举例】

无

【相关主题】

无

3.3.1.5 CVI_GAUGEMNG_GetBatteryLevel

【描述】

获得电池电量水平

【语法】

```
int32_t CVI_GAUGEMNG_GetBatteryLevel(uint8_t* ps32Level);
```

【参数】

参数名称	描述	输入/输出
ps32Level	电池电量水平	输出

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_gaugemng.h
- 库文件: libcvi_devmng.a/libcvi_devmng.so

【注意】

无

【举例】

无

【相关主题】

无

3.3.1.6 CVI_GAUGEMNG_GetChargeState

【描述】

电池是否正在充电

【语法】

```
int32_t CVI_GAUGEMNG_GetChargeState(bool* pbCharge);
```

【参数】

参数名称	描述	输入/输出
pbCharge	电池充电状态	输出

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_gaugemng.h
- 库文件: libcvi_devmng.a/libcvi_devmng.so

【注意】

无

【举例】

无

【相关主题】

无

3.3.1.7 CVI_GSENSORMNG_RegisterEvent

【描述】

加速度传感器状态注册

【语法】

```
int32_t CVI_GSENSORMNG_RegisterEvent(void);
```

【参数】

无

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_gsensormng.h
- 库文件: libcvi_devmng.a/libcvi_devmng.so

【注意】

无

【举例】

无

【相关主题】

无

3.3.1.8 CVI_GSENSORMNG_Init

【描述】

加速度传感器管理初始化

【语法】

```
int32_t CVI_GSENSORMNG_Init(const CVI_GSENSORMNG_CFG_S* pstCfg);
```

【参数】

参数名称	描述	输入/输出
pstCfg	加速度传感器配置	输入

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_gsensormng.h
- 库文件: libcvi_devmng.a/libcvi_devmng.so

【注意】

无

【举例】

无

【相关主题】

无

3.3.1.9 CVI_GSENSORMNG_SetSensitivity

【描述】

设置传感器灵敏度

【语法】

```
int32_t CVI_GSENSORMNG_SetSensitivity(CVI_HAL_GSENSOR_SENSITIVITY_E enSensitivity);
```

【参数】

参数名称	描述	输入/输出
enSensitivity	灵敏度	输入

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_gsensormng.h
- 库文件: libcvi_devmng.a/libcvi_devmng.so

【注意】

无

【举例】

无

【相关主题】

无

3.3.1.10 CVI_GSENSORMNG_MenuSetSensitivity

【描述】

菜单中设置传感器灵敏度

【语法】

```
void CVI_GSENSORMNG_MenuSetSensitivity(CVI_HAL_GSENSOR_SENSITIVITY_E enSensitivity);
```

【参数】

参数名称	描述	输入/输出
enSensitivity	灵敏度	输入

【返回值】

无

【需求】

- 头文件: cvi_gsensormng.h
- 库文件: libcvi_devmng.a/libcvi_devmng.so

【注意】

无

【举例】

无

【相关主题】

无

3.3.1.11 CVI_GSENSORMNG_GetAttr

【描述】

获取加速度传感器属性

【语法】

```
int32_t CVI_GSENSORMNG_GetAttr(CVI_GSENSORMNG_ATTR_S* pstAttr);
```

【参数】

参数名称	描述	输入/输出
pstAttr	传感器属性	输出

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_gsensormng.h
- 库文件: libcvi_devmng.a/libcvi_devmng.so

【注意】

无

【举例】

无

【相关主题】

无

3.3.1.12 CVI_GSENSORMNG_SetAttr

【描述】

设置加速度传感器属性

【语法】

```
int32_t CVI_GSENSORMNG_SetAttr(const CVI_GSENSORMNG_ATTR_S* pstAttr);
```

【参数】

参数名称	描述	输入/输出
pstAttr	传感器属性	输入

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_gsensormng.h
- 库文件: libcvi_devnmng.a/libcvi_devnmng.so

【注意】

无

【举例】

无

【相关主题】

无

3.3.1.13 CVI_GSENSORMNG_OpenInterrupt

【描述】

打开传感器中断

【语法】

```
int32_t CVI_GSENSORMNG_OpenInterrupt(int32_t IntNum);
```

【参数】

参数名称	描述	输入/输出
IntNum	中断号	输入

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_gsensormng.h
- 库文件: libcvi_devmng.a/libcvi_devmng.so

【注意】

无

【举例】

无

【相关主题】

无

3.3.1.14 CVI_GSENSORMNG_DeInit

【描述】

加速度传感器管理反初始化

【语法】

```
int32_t CVI_GSENSORMNG_DeInit(void);
```

【参数】

无

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_gsensormng.h
- 库文件: libcvi_devmng.a/libcvi_devmng.so

【注意】

无

【举例】

无

【相关主题】

无

3.3.1.15 CVI_KEYMNG_RegisterEvent

【描述】

按键状态注册

【语法】

```
int32_t CVI_KEYMNG_RegisterEvent(void);
```

【参数】

无

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_keymng.h
- 库文件: libcvi_devmng.a/libcvi_devmng.so

【注意】

无

【举例】

无

【相关主题】

无

3.3.1.16 CVI_KEYMNG_Init

【描述】

按键管理初始化

【语法】

```
int32_t CVI_KEYMNG_Init(CVI_KEYMNG_CFG_S KeyCfg);
```

【参数】

参数名称	描述	输入/输出
KeyCfg	按键配置	输入

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_keymng.h
- 库文件: libcvi_devmng.a/libcvi_devmng.so

【注意】

无

【举例】

无

【相关主题】

无

3.3.1.17 CVI_KEYMNG_DeInit

【描述】

按键管理反初始化

【语法】

```
int32_t CVI_KEYMNG_DeInit(void);
```

【参数】

无

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_keymng.h
- 库文件: libcvi_devmng.a/libcvi_devmng.so

【注意】

无

【举例】

无

【相关主题】

无

3.3.1.18 CVI_LED_MNG_Init

【描述】

led 灯初始化

【语法】

```
int32_t CVI_LED_MNG_Init();
```

【参数】

无

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_ledmng.h
- 库文件: libcvi_devmng.a/libcvi_devmng.so

【注意】

无

【举例】

无

【相关主题】

无

3.3.1.19 CVI_LED_MNG_DeInit

【描述】

led 灯反初始化

【语法】

```
int32_t CVI_LED_MNG_DeInit(void);
```

【参数】

无

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_ledmng.h
- 库文件: libcvi_devmng.a/libcvi_devmng.so

【注意】

无

【举例】

无

【相关主题】

无

3.3.1.20 CVI_LED_MNG_Control

【描述】

设置 GPIO 状态

【语法】

```
void CVI_LED_MNG_Control(int32_t control);
```

【参数】

参数名称	描述	输入/输出
control	控制使能	输入

【返回值】

无

【需求】

- 头文件: cvi_ledmng.h
- 库文件: libcvi_devmng.a/libcvi_devmng.so

【注意】

无

【举例】

无

【相关主题】

无

3.3.1.21 CVI_WATCHDOGMNG_Init

【描述】

看门狗管理初始化

【语法】

```
CVI_S32 CVI_WATCHDOGMNG_Init(CVI_S32 s32Time_s);
```

【参数】

参数名称	描述	输入/输出
s32Time_s	超时时间	输入

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_watchdogmng.h
- 库文件: libcvi_devmng.a/libcvi_devmng.so

【注意】

无

【举例】

无

【相关主题】

无

3.3.1.22 CVI_WATCHDOGMNG_DeInit

【描述】

看门狗管理反初始化

【语法】

```
CVI_S32 CVI_WATCHDOGMNG_DeInit(CVI_VOID);
```

【参数】

无

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_watchdogmng.h
- 库文件: libcvi_devmng.a/libcvi_devmng.so

【注意】

无

【举例】

无

【相关主题】

无

3.3.1.23 CVI_WIFIMNG_Start

【描述】

启动 wifi

【语法】

```
int32_t CVI_WIFIMNG_Start(CVI_HAL_WIFI_CFG_S WifiCfg, char *pstDefaultssid);
```

【参数】

参数名称	描述	输入/输出
WifiCfg	wifi 配置	输入
pstDefaultssid	默认 ssid	输入

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_wifimng.h
- 库文件: libcvi_devmng.a/libcvi_devmng.so

【注意】

无

【举例】

无

【相关主题】

无

3.3.1.24 CVI_WIFIMNG_Stop

【描述】

停止 wifi

【语法】

```
int32_t CVI_WIFIMNG_Stop(void);
```

【参数】

无

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件：cvi_wifimng.h
- 库文件：libcvi_devmng.a/libcvi_devmng.so

【注意】

无

【举例】

无

【相关主题】

无

3.3.2 数据类型

视频输入相关数据类型定义如下：

- CVI_GAUGEMNG_CFG_S：电池电量配置。
- CVI_GSENSORMNG_ATTR_S：加速度传感器属性。
- CVI_GSENSORMNG_CFG_S：加速度传感器配置。
- CVI_KEYMNG_KEY_NUM_EACH_GRP：每组键数量。
- CVI_KEYMNG_KEY_TYPE_E：按键类型。

- CVI_KEYMNG_KEY_CLICK_ATTR_S：点击类型属性。
- CVI_KEYMNG_KEY_HOLD_ATTR_S：hold 类型属性。
- CVI_KEYMNG_KEY_ATTR_S：按键管理属性。
- CVI_KEYMNG_KEY_CFG_S：按键管理配置。
- CVI_KEYMNG_GRP_KEY_CFG_S：按键组配置。
- CVI_KEYMNG_CFG_S：按键管理上下文。

3.3.2.1 CVI_GAUGEMNG_CFG_S

【说明】

电池电量配置

【定义】

```
typedef struct cviGAUGEMNG_CFG_S{  
    int32_t s32LowLevel; /**< in percent */  
    int32_t s32UltraLowLevel; /**< in percent */  
} CVI_GAUGEMNG_CFG_S;
```

【成员】

成员名称	描述
s32LowLevel	低电量
s32UltraLowLevel	极低电量

【注意事项】

无。

【相关数据类型及接口】

无。

3.3.2.2 CVI_GSENSORMNG_ATTR_S

【说明】

加速度传感器属性

【定义】

```
typedef struct cviGSENSORMNG_ATTR_S{  
    uint32_t u32SampleRate; /**<sample rate,0 mean Adopt default,not config,unit kps*/  
} CVI_GSENSORMNG_ATTR_S;
```

【成员】

成员名称	描述
u32SampleRate	采样率

【注意事项】

无。

【相关数据类型及接口】

无。

3.3.2.3 CVI_GSENSORMNG_CFG_S

【说明】

加速度传感器配置

【定义】

```
typedef struct cviGSENSORMNG_CFG_S{  
    int32_t gsensor_level;  
    int32_t gsensor_enable;  
    CVI_HAL_GSENSOR_SENSITIVITY_E enSensitivity;  
    CVI_GSENSORMNG_ATTR_S stAttr;  
} CVI_GSENSORMNG_CFG_S;
```

【成员】

成员名称	描述
gsensor_level	等级
gsensor_enable	使能
enSensitivity	灵敏度
stAttr	传感器属性

【注意事项】

无。

【相关数据类型及接口】

无。

3.3.2.4 CVI_KEYMNG_KEY_NUM_EACH_GRP

【说明】

每组键数量

【定义】

```
#define CVI_KEYMNG_KEY_NUM_EACH_GRP (4) /**<key number in group-key*/
```

【注意事项】

无。

【相关数据类型及接口】

无。

3.3.2.5 CVI_KEYMNG_KEY_TYPE_E

【说明】

按键类型

【定义】

```
typedef enum cviKEYMNG_KEY_TYPE_E{  
    CVI_KEYMNG_KEY_TYPE_CLICK = 0, /**<support click and longclick event */  
    CVI_KEYMNG_KEY_TYPE_HOLD,      /**<support keydown and keyup event */  
} CVI_KEYMNG_KEY_TYPE_E;
```

【成员】

成员名称	描述
CVI_KEYMNG_KEY_TYPE_CLICK	点击类型
CVI_KEYMNG_KEY_TYPE_HOLD	hold 类型

【注意事项】

无。

【相关数据类型及接口】

无。

3.3.2.6 CVI_KEYMNG_KEY_CLICK_ATTR_S

【说明】

点击类型属性

【定义】

```
typedef struct cviKEYMNG_KEY_CLICK_ATTR_S{  
    bool bLongClickEnable; /**<tture: support click and longclick event; false: only support click event */  
    uint32_t u32LongClickTime_msec; /**<long click check time, valid when longclick enabled */  
} CVI_KEYMNG_KEY_CLICK_ATTR_S;
```

【成员】

成员名称	描述
bLongClickEnable	使能长按
u32LongClickTime_msec	长按时间

【注意事项】

无。

【相关数据类型及接口】

无。

3.3.2.7 CVI_KEYMNG_KEY_HOLD_ATTR_S

【说明】

hold 类型属性

【定义】

```
typedef struct cviKEYMNG_KEY_HOLD_ATTR_S{  
} CVI_KEYMNG_KEY_HOLD_ATTR_S;
```

【成员】

无

【注意事项】

无。

【相关数据类型及接口】

无。

3.3.2.8 CVI_KEYMNG_KEY_ATTR_S

【说明】

按键管理属性

【定义】

```
typedef struct cviKEYMNG_KEY_ATTR_S{  
    CVI_KEYMNG_KEY_TYPE_E enType; /**<click type or hold type*/  
    int32_t s32Id; /**<key id */  
    union tagKEYMNG_KEY_ATTR_U  
    {  
        CVI_KEYMNG_KEY_CLICK_ATTR_S stClickKeyAttr; /**<click attr type */  
        CVI_KEYMNG_KEY_HOLD_ATTR_S stHoldKeyAttr; /**<hold attr type */  
    } unAttr;  
} CVI_KEYMNG_KEY_ATTR_S;
```

【成员】

成员名称	描述
enType	按键类型
s32Id	键值
unAttr	键属性枚举量

【注意事项】

无。

【相关数据类型及接口】

无。

3.3.2.9 CVI_KEYMNG_KEY_CFG_S

【说明】

按键管理配置

【定义】

```
typedef struct cviKEYMNG_KEY_CFG_S{  
    uint32_t u32KeyCnt;    /**<key count*/  
    CVI_KEYMNG_KEY_ATTR_S astKeyAttr[CVI_KEYMNG_KEY_IDX_BUTT];  
} CVI_KEYMNG_KEY_CFG_S;
```

【成员】

成员名称	描述
u32KeyCnt	按键数量
astKeyAttr	按键管理属性数组

【注意事项】

无。

【相关数据类型及接口】

无。

3.3.2.10 CVI_KEYMNG_GRP_KEY_CFG_S

【说明】

按键组配置

【定义】

```
typedef struct cviKEYMNG_GRP_KEY_CFG_S  
  
    bool bEnable;    /**<ture: support group key event; false: not support */  
    /**<only support two keys group at present*/  
    uint32_t au32GrpKeyIdx[CVI_KEYMNG_KEY_NUM_EACH_GRP];  
} CVI_KEYMNG_GRP_KEY_CFG_S;
```

【成员】

成员名称	描述
bEnable	使能
au32GrpKeyIdx	组键值

【注意事项】

无。

【相关数据类型及接口】

无。

3.3.2.11 CVI_KEYMNG_CFG_S

【说明】

按键管理上下文

【定义】

```
typedef struct cviKEYMNG_CFG_S{  
    CVI_KEYMNG_KEY_CFG_S stKeyCfg;  
    CVI_KEYMNG_GRP_KEY_CFG_S stGrpKeyCfg;  
} CVI_KEYMNG_CFG_S;
```

【成员】

成员名称	描述
stKeyCfg	按键管理配置
stGrpKeyCfg	按键组配置

【注意事项】

无。

【相关数据类型及接口】

无。

3.4 Filemng

该通用组件用于文件系统管理，可参考文档 Framework 通用软件开发指南的 DTCTF 组件。

3.4.1 API 参考

该功能模块为用户提供以下 API:

- CVI_FILEMNG_RegisterEvent : 文件状态注册
- CVI_FILEMNG_Init : 文件管理初始化
- CVI_FILEMNG_DeInit : 文件管理反初始化
- CVI_FILEMNG_SetDiskState : 设置磁盘状态
- CVI_FILEMNG_CheckDiskSpace : 检查磁盘空间
- CVI_FILEMNG_AddFile : 添加文件
- CVI_FILEMNG_RemoveFile : 文件删除
- CVI_FILEMNG_SetSearchScope : 设置文件搜索范围
- CVI_FILEMNG_GetFileObjCnt : 获取文件数量
- CVI_FILEMNG_GetFileByIndex : 根据索引获取文件

- `CVI_FILEMNG_GetFileInfoByName` : 根据名字获取文件信息
- `CVI_FILEMNG_SpaceMonitorCheckSpace` : 触发空间监测器检查 SD 卡空间
- `CVI_FILEMNG_GeneratePhotoName` 生成照片名
- `CVI_FILEMNG_GenerateRecordName` : 生成录像文件名
- `CVI_FILEMNG_RenameMovToEmr` : 普通录像文件名重命名为紧急录像文件名
- `CVI_FILEMNG_FileCoverStatus` : 停止空间监测器检查 SD 卡空间
- `CVI_FILEMNG_AlignRecordFileSize` : 修复文件并指定文件大小
- `CVI_FILEMNG_CreateSDConfigFile` : 创建 `cvi_sdconfig` 文件
- `CVI_FILEMNG_RecoverAddFileName` : 给录像文件命名
- `CVI_FILEMNG_RecoverRemoveFileName` : 更新 `cvi_sdconfig` 文件中文件名
- `CVI_FILEMNG_SDConfigFileIsExist` : 检查 `cvi_sdconfig` 文件是否存在
- `CVI_FILEMNG_PreallocateState` : 是否预分配
- `CVI_FILEMNG_GetDirType` : 获取文件夹类型
- `CVI_FILEMNG_SetRemoveLoop` : 停止循环录像

3.4.1.1 CVI_FILEMNG_Init

【描述】

文件管理初始化

【语法】

```
int32_t CVI_FILEMNG_Init(const CVI_FILEMNG_COMM_CFG_S *pstCfg, const CVI_
→FILEMNG_DTCF_CFG_S *pstDTCF_Cfg);
```

【参数】

参数名称	描述	输入/输出
<code>pstCfg</code>	文件公共配置	输入
<code>pstDTCF_Cfg</code>	dtcf 配置	输入

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: `cvi_filemng.h/cvi_filemng_dtcf.h`
- 库文件: `libcvi_filemng.a/libcvi_filemng.so`

【注意】

无

【举例】

无

【相关主题】

无

3.4.1.2 CVI_FILEMNG_RegisterEvent

【描述】

文件状态注册

【语法】

```
int32_t CVI_FILEMNG_RegisterEvent(void);
```

【参数】

无

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_filemng.h/cvi_filemng_dtcf.h
- 库文件: libcvi_filemng.a/libcvi_filemng.so

【注意】

无

【举例】

无

【相关主题】

无

3.4.1.3 CVI_FILEMNG_DeInit

【描述】

文件管理反初始化

【语法】

```
int32_t CVI_FILEMNG_DeInit(void);
```

【参数】

无

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_filemng.h/cvi_filemng_dtcf.h
- 库文件: libcvi_filemng.a/libcvi_filemng.so

【注意】

无

【举例】

无

【相关主题】

无

3.4.1.4 CVI_FILEMNG_SetDiskState

【描述】

设置磁盘状态

【语法】

```
int32_t CVI_FILEMNG_SetDiskState(bool bAvailable);
```

【参数】

参数名称	描述	输入/输出
bAvailable	dtcf 配置	输入

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvt_filemng.h/cvt_filemng_dtcf.h
- 库文件: libcvt_filemng.a/libcvt_filemng.so

【注意】

无

【举例】

无

【相关主题】

无

3.4.1.5 CVI_FILEMNG_CheckDiskSpace

【描述】

检查磁盘空间

【语法】

```
int32_t CVI_FILEMNG_CheckDiskSpace(void);
```

【参数】

无

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvt_filemng.h/cvt_filemng_dtcf.h
- 库文件: libcvt_filemng.a/libcvt_filemng.so

【注意】

无

【举例】

无

【相关主题】

无

3.4.1.6 CVI_FILEMNG_AddFile

【描述】

添加文件

【语法】

```
int32_t CVI_FILEMNG_AddFile(const char *pszFilePath);
```

【参数】

参数名称	描述	输入/输出
pstDTCF_Cfg	文件路径	输入

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_filemng.h/cvi_filemng_dtcf.h
- 库文件: libcvi_filemng.a/libcvi_filemng.so

【注意】

无

【举例】

无

【相关主题】

无

3.4.1.7 CVI_FILEMNG_RemoveFile

【描述】

文件删除

【语法】

```
int32_t CVI_FILEMNG_RemoveFile(const char *pszFilePath);
```

【参数】

参数名称	描述	输入/输出
pstpszFilePath	文件路径	输入

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_filemng.h/cvi_filemng_dtcf.h
- 库文件: libcvi_filemng.a/libcvi_filemng.so

【注意】

无

【举例】

无

【相关主题】

无

3.4.1.8 CVI_FILEMNG_SetSearchScope

【描述】

设置文件搜索范围

【语法】

```
int32_t CVI_FILEMNG_SetSearchScope(CVI_DTCF_DIR_E aenDirs[DTCF_DIR_BUTT], uint32_t u32DirCount, uint32_t *pu32FileObjCnt);
```

【参数】

参数名称	描述	输入/输出
aenDirs	文件夹类型数组	输入
u32DirCount	文件夹数量	输入
pu32FileObjCnt	文件数量	输出

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_filemng.h/cvi_filemng_dtcf.h
- 库文件: libcvi_filemng.a/libcvi_filemng.so

【注意】

无

【举例】

无

【相关主题】

无

3.4.1.9 CVI_FILEMNG_GetFileObjCnt**【描述】**

获取文件数量

【语法】

```
int32_t CVI_FILEMNG_GetFileObjCnt(CVI_FILEMNG_FILE_TYPE_E enType, uint32_t  
↔ *pu32FileObjCnt);
```

【参数】

参数名称	描述	输入/输出
enType	文件类型	输入
pu32FileObjCnt	文件数量	输入

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_filemng.h/cvi_filemng_dtcf.h
- 库文件: libcvi_filemng.a/libcvi_filemng.so

【注意】

无

【举例】

无

【相关主题】

无

3.4.1.10 CVI_FILEMNG_GetFileByIndex

【描述】

根据索引获取文件

【语法】

```
int32_t CVI_FILEMNG_GetFileByIndex(uint32_t u32FileIdx, char *pazFileName, uint32_t u32Length);
```

【参数】

参数名称	描述	输入/输出
u32FileIdx	索引	输入
pazFileName	文件名	输出
u32Length	文件名长度	输入

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_filemng.h/cvi_filemng_dtcf.h
- 库文件: libcvi_filemng.a/libcvi_filemng.so

【注意】

无

【举例】

无

【相关主题】

无

3.4.1.11 CVI_FILEMNG_GetFileInfoByName

【描述】

根据名字获取文件信息

【语法】

```
int32_t CVI_FILEMNG_GetFileInfoByName(const char *pszFilePath, CVI_FILEMNG_FILE_INFO_S *pstFileInfo);
```

【参数】

参数名称	描述	输入/输出
pszFilePath	文件路径	输入
pstFileInfo	文件信息	输出

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvt_filemng.h/cvt_filemng_dtcf.h
- 库文件: libcvt_filemng.a/libcvt_filemng.so

【注意】

无

【举例】

无

【相关主题】

无

3.4.1.12 CVI_FILEMNG_SpacemonitorCheckSpace

【描述】

触发空间监测器检查 SD 卡空间

【语法】

```
int32_t CVI_FILEMNG_SpacemonitorCheckSpace(void);
```

【参数】

无

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvt_filemng.h/cvt_filemng_dtcf.h
- 库文件: libcvt_filemng.a/libcvt_filemng.so

【注意】

无

【举例】

无

【相关主题】

无

3.4.1.13 CVI_FILEMNG_GeneratePhotoName**【描述】**

生成照片文件名

【语法】

```
int32_t CVI_FILEMNG_GeneratePhotoName(CVI_DTCF_FILE_TYPE_E enType, CVI_DTCF_
↳DIR_E enDir, bool bPreAlloc, char *FileName);
```

【参数】

参数名称	描述	输入/输出
enType	文件类型	输入
enDir	dtcf 文件夹类型	输入
bPreAlloc	是否预分配	输入
FileName	文件名	输出

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_filemng.h/cvi_filemng_dtcf.h
- 库文件: libcvi_filemng.a/libcvi_filemng.so

【注意】

无

【举例】

无

【相关主题】

无

3.4.1.14 CVI_FILEMNG_GenerateRecordName

【描述】

生成录像文件名

【语法】

```
int32_t CVI_FILEMNG_GenerateRecordName(CVI_DTCF_FILE_TYPE_E enType, CVI_DTCF_
↳DIR_E enDir, char *pstFileName);
```

【参数】

参数名称	描述	输入/输出
enType	文件类型	输入
enDir	dtcf 文件夹类型	输入
pstFileName	文件名	输出

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_filemng.h/cvi_filemng_dtcf.h
- 库文件: libcvi_filemng.a/libcvi_filemng.so

【注意】

无

【举例】

无

【相关主题】

无

3.4.1.15 CVI_FILEMNG_RenameMovToEmr

【描述】

录像文件名重命名为紧急录像文件名

【语法】

```
int32_t CVI_FILEMNG_RenameMovToEmr(const char *pazFilePath);
```

【参数】

参数名称	描述	输入/输出
pstpaz-FilePathCfg	文件路径	输入

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_filemng.h/cvi_filemng_dtcf.h
- 库文件: libcvi_filemng.a/libcvi_filemng.so

【注意】

无

【举例】

无

【相关主题】

无

3.4.1.16 CVI_FILEMNG_FileCoverStatus

【描述】

停止空间监测器检查 SD 卡空间

【语法】

```
int32_t CVI_FILEMNG_FileCoverStatus(bool en);
```

【参数】

参数名称	描述	输入/输出
en	使能	输入

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_filemng.h/cvi_filemng_dtcf.h

- 库文件: libcvi_filemng.a/libcvi_filemng.so

【注意】

无

【举例】

无

【相关主题】

无

3.4.1.17 CVI_FILEMNG_AlignRecordFileSize

【描述】

修复文件并指定文件大小

【语法】

```
int32_t CVI_FILEMNG_AlignRecordFileSize(bool enftruncate, uint32_t size);
```

【参数】

参数名称	描述	输入/输出
enftruncate	截断使能	输入
size	文件大小	输入

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_filemng.h/cvi_filemng_dtcf.h
- 库文件: libcvi_filemng.a/libcvi_filemng.so

【注意】

无

【举例】

无

【相关主题】

无

3.4.1.18 CVI_FILEMNG_CreateSDConfigFile

【描述】

创建 cvi_sdconfig 文件

【语法】

```
int32_t CVI_FILEMNG_CreateSDConfigFile(void);
```

【参数】

无

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_filemng.h/cvi_filemng_dtcf.h
- 库文件: libcvi_filemng.a/libcvi_filemng.so

【注意】

无

【举例】

无

【相关主题】

无

3.4.1.19 CVI_FILEMNG_RecoverAddFileName

【描述】

给录像文件命名

【语法】

```
int32_t CVI_FILEMNG_RecoverAddFileName(const char *filePath);
```

【参数】

参数名称	描述	输入/输出
filePath	文件路径	输入

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_filemng.h/cvi_filemng_dtcf.h
- 库文件: libcvi_filemng.a/libcvi_filemng.so

【注意】

无

【举例】

无

【相关主题】

无

3.4.1.20 CVI_FILEMNG_RecoverRemoveFileName

【描述】

更新 cvi_sdconfig 文件中文件名

【语法】

```
int32_t CVI_FILEMNG_RecoverRemoveFileName(const char *filePath);
```

【参数】

参数名称	描述	输入/输出
filePath	文件路径	输入

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_filemng.h/cvi_filemng_dtcf.h
- 库文件: libcvi_filemng.a/libcvi_filemng.so

【注意】

无

【举例】

无

【相关主题】

无

3.4.1.21 CVI_FILEMNG_SDConfigFileIsExist**【描述】**

检查 cvi_sdconfig 文件是否存在

【语法】

```
bool CVI_FILEMNG_SDConfigFileIsExist(void);
```

【参数】

无

【返回值】

返回值	描述
true	成功
false	失败

【需求】

- 头文件: cvi_filemng.h/cvi_filemng_dtcf.h
- 库文件: libcvi_filemng.a/libcvi_filemng.so

【注意】

无

【举例】

无

【相关主题】

无

3.4.1.22 CVI_FILEMNG_PreallocateState**【描述】**

是否预分配

【语法】

```
void CVI_FILEMNG_PreallocateState(CVI_FILEMNG_DTCF_CFG_S *pstDTCF_Cfg);
```

【参数】

参数名称	描述	输入/输出
pstDTCF_Cfg	dtcf 配置	输入

【返回值】

无

【需求】

- 头文件: cvi_filemng.h/cvi_filemng_dtcf.h
- 库文件: libcvi_filemng.a/libcvi_filemng.so

【注意】

无

【举例】

无

【相关主题】

无

3.4.1.23 CVI_FILEMNG_GetDirType

【描述】

获取文件夹类型

【语法】

```
int32_t CVI_FILEMNG_GetDirType(int32_t id, CVI_DTCF_DIR_E base);
```

【参数】

参数名称	描述	输入/输出
id	sensor id	输入
base	dtcf 文件夹类型	输入

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_filemng.h/cvi_filemng_dtcf.h
- 库文件: libcvi_filemng.a/libcvi_filemng.so

【注意】

无

【举例】

无

【相关主题】

无

3.4.1.24 CVI_FILEMNG_SetRemoveLoop**【描述】**

停止循环录像

【语法】

```
int32_t CVI_FILEMNG_SetRemoveLoop(int32_t en);
```

【参数】

参数名称	描述	输入/输出
en	使能	输入

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_filemng.h/cvi_filemng_dtcf.h
- 库文件: libcvi_filemng.a/libcvi_filemng.so

【注意】

无

【举例】

无

【相关主题】

无

3.4.2 数据类型

视频输入相关数据类型定义如下:

- CVI_FILEMNG_DTCF_CFG_S : dtcf 文件系统配置
- CVI_FILEMNG_COMM_CFG_S : 文件管理通用配置
- CVI_FILEMNG_FILE_TYPE_E : 文件类型枚举
- CVI_FILEMNG_FILE_INFO_S : 文件信息

- CVI_FILEMNG_DTCF_MAX_PHOTO_DIR：最大拍照文件夹数量。
- CVI_APPCOMM_MAX_PATH_LEN：app 通用最大路径长度。

3.4.2.1 CVI_APPCOMM_MAX_PATH_LEN

【说明】

app 通用最大路径长度

【定义】

```
#define CVI_APPCOMM_MAX_PATH_LEN (128)
```

【注意事项】

无

【相关数据类型及接口】

无。

3.4.2.2 CVI_FILEMNG_DTCF_MAX_PHOTO_DIR

【说明】

最大拍照文件夹数量

【定义】

```
#define CVI_FILEMNG_DTCF_MAX_PHOTO_DIR (MAX_CAMERA_INSTANCES)
```

【注意事项】

MAX_CAMERA_INSTANCES 来自参数管理的 config_devmng.ini 文件，编译时赋值，详情可参考流媒体后视镜 Param。

【相关数据类型及接口】

无。

3.4.2.3 CVI_FILEMNG_DTCF_CFG_S

【说明】

dtcf 文件系统配置

【定义】

```
typedef struct cviFILEMNG_DTCF_CFG_S {  
    /**<pre allocate file count */  
    uint32_t u32PreAllocCnt[CVI_FILEMNG_DTCF_MAX_PHOTO_DIR];  
    /**<pre allocate file size ,suggest 2M, unit:byte */  
    uint32_t u32PreAllocUnit[CVI_FILEMNG_DTCF_MAX_PHOTO_DIR];  
    bool preAllocFilesEnable;  
    uint32_t u32PreallocReservedMemory;
```

(下页继续)

(续上页)

```

uint32_t u32PreAllocPercentage[DTCF_DIR_BUTT];/**<pre allocate file count */
/**<pre allocate file size ,suggest 2M, unit:byte */
uint32_t u32PreAllocFileUnit[DTCF_DIR_BUTT];
uint8_t u8SharePercent;/**<0:share 1~100:movie space ratio;eg.80 */
/**<unit:MB;filemanager will publish a warning when the remaining space is less than
→u32WarningStage */
uint32_t u32WarningStage;
/**<unit:MB;u32GuaranteedStage should NOT be less than u32WarningStage;the loop coverage is
→disable when u32GuaranteedStage is 0 */
uint32_t u32GuaranteedStage;
char szRootDir[CVI_APPCOMM_MAX_PATH_LEN];/**<file manager top directory name in
→mount path. eg."CVICAM" */
char aszDirNames[DTCF_DIR_BUTT][CVI_DIR_LEN_MAX]; /**<eg.{"EMR", "EMR_s",
→"Movie", "Movie_s", "", "", "", "", "Photo"} */
uint32_t u32RemoveLoopEn;
} CVI_FILEMNG_DTCF_CFG_S;

```

【成员】

成员名称	描述
u32PreAllocCnt	预分配拍照文件数
u32PreAllocUnit	预分配拍照文件大小
preAllocFilesEnable	预分配使能
u32PreallocReservedMemory	预分配保留内存
u32PreAllocPercentage	预分配百分比
u32PreAllocFileUnit	预分配文件数
u32RemoveLoopEn	循环覆盖使能
u8SharePercent	共享百分比
u32WarningStag	警告百分比
u32GuaranteedStage	开启循环覆盖百分比
szRootDir	文件系统根路径
aszDirNames	文件夹名

【注意事项】

DTCF_DIR_BUTT, CVI_DIR_LEN_MAX 来自 DTCF 组件，详情参考 Framework 通用软件开发指南。

【相关数据类型及接口】

无。

3.4.2.4 CVI_FILEMNG_COMM_CFG_S

【说明】

文件管理通用配置

【定义】

```
typedef struct cviFILEMNG_COMM_CFG_S {  
    char szMntPath[CVI_APPCOMM_MAX_PATH_LEN]; /**<disk mount path. eg. "/app/sd/" */  
} CVI_FILEMNG_COMM_CFG_S;
```

【成员】

成员名称	描述
szMntPath	挂载路径

【注意事项】

无。

【相关数据类型及接口】

无。

3.4.2.5 CVI_FILEMNG_FILE_TYPE_E

【说明】

文件类型枚举

【定义】

```
typedef enum cviFILEMNG_FILE_TYPE_E {  
    CVI_FILEMNG_FILE_TYPE_RECORD = 0, /**<record file. eg. *.MP4,*.LRV,*.MOV,etc */  
    CVI_FILEMNG_FILE_TYPE_PHOTO,    /**<photo file. eg. *.JPG,*.DNG,etc */  
    CVI_FILEMNG_FILE_TYPE_BUTT  
} CVI_FILEMNG_FILE_TYPE_E;
```

【成员】

成员名称	描述
CVI_FILEMNG_FILE_TYPE_RECORD	录像文件
CVI_FILEMNG_FILE_TYPE_PHOTO	拍照文件

【注意事项】

无。

【相关数据类型及接口】

无

3.4.2.6 CVI_FILEMNG_FILE_INFO_S

【说明】

文件信息

【定义】

```
typedef struct cviFILEMNG_FILE_INFO_S {  
    char szAbsPath[CVI_APPCOMM_MAX_PATH_LEN];    /**<file name ,eg. "/app/sd/CAM/  
    ↪Photo/2017_05_27_11281500.JPG" */  
    char szCreateTime[CVI_FILEMNG_MAX_DATETIME_LEN]; /**<file create time ,eg."2017/05/  
    ↪27 11:28:15" */  
    uint64_t u64FileSize_byte;    /**<file size in bytes. eg. 120,100,100 */  
    uint32_t u32Duration_sec;    /**<record file duration in seconds. eg. 300 */  
} CVI_FILEMNG_FILE_INFO_S;
```

【成员】

成员名称	描述
szAbsPath	绝对路径
szCreateTime	创建时间
u64FileSize_byte	文件大小
u32Duration_sec	文件时长

【注意事项】

无。

【相关数据类型及接口】

无

3.5 Ispexposure

该通用组件与 isp 曝光有关。

3.5.1 API 参考

该功能模块为用户提供以下 API:

- CVI_ISPEXP_Init : 传感器 isp 初始化
- CVI_ISPEXP_DeInit : 传感器 isp 曝光反初始化

3.5.1.1 CVI_ISPEXP_Init

【描述】

传感器 isp 初始化

【语法】

```
int32_t CVI_ISPEXP_Init(bool *cam_enable);
```

【参数】

参数名称	描述	输入/输出
cam_enable	sensor 使能	输入

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_ispexposure.h

【注意】

无

【举例】

无

【相关主题】

无

3.5.1.2 CVI_ISPEXP_DeInit

【描述】

传感器 isp 曝光反初始化

【语法】

```
void CVI_ISPEXP_DeInit(void);
```

【参数】

无

【返回值】

无

【需求】

- 头文件: `cvi_ispexposure.h`

【注意】

无

【举例】

无

【相关主题】

无

3.6 Liveviewmng

该通用组件与 Vo 预览有关。

3.6.1 API 参考

该功能模块为用户提供以下 API:

- `CVI_LIVEVIEWMNG_Switch` : 切换预览镜头
- `CVI_LIVEVIEWMNG_MoveUp` : 预览界面上移, 目前仅支持单镜头
- `CVI_LIVEVIEWMNG_MoveDown` : 预览界面下移, 目前仅支持单镜头
- `CVI_LIVEVIEWMNG_Mirror` : 预览界面镜面翻转
- `CVI_LIVEVIEWMNG_Filp` 预览界面翻转

3.6.1.1 CVI_LIVEVIEWMNG_Switch

【描述】

切换预览镜头

【语法】

```
int32_t CVI_LIVEVIEWMNG_Switch(uint32_t val);
```

【参数】

参数名称	描述	输入/输出
val	预览值	输入

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_liveviewmng.h
- 库文件: libcvi_liveviewmng.a/libcvi_liveviewmng.so

【注意】

无

【举例】

无

【相关主题】

无

3.6.1.2 CVI_LIVEVIEWMNG_MoveUp

【描述】

预览界面上移, 目前仅支持单镜头

【语法】

```
int32_t CVI_LIVEVIEWMNG_MoveUp(int32_t wndid);
```

【参数】

参数名称	描述	输入/输出
wndid	窗口值	输入

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_liveviewmng.h
- 库文件: libcvi_liveviewmng.a/libcvi_liveviewmng.so

【注意】

无

【举例】

无

【相关主题】

无

3.6.1.3 CVI_LIVEVIEWMNG_MoveDown

【描述】

预览界面下移，目前仅支持单镜头

【语法】

```
int32_t CVI_LIVEVIEWMNG_MoveDown(int32_t wndid);
```

【参数】

参数名称	描述	输入/输出
wndid	窗口值	输入

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件：cvi_liveviewmng.h
- 库文件：libcvi_liveviewmng.a/libcvi_liveviewmng.so

【注意】

无

【举例】

无

【相关主题】

无

3.6.1.4 CVI_LIVEVIEWMNG_Mirror

【描述】

预览界面镜面翻转

【语法】

```
int32_t CVI_LIVEVIEWMNG_Mirror(uint32_t val);
```

【参数】

参数名称	描述	输入/输出
val	预览值	输入

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_liveviewmng.h
- 库文件: libcvi_liveviewmng.a/libcvi_liveviewmng.so

【注意】

无

【举例】

无

【相关主题】

无

3.6.1.5 CVI_LIVEVIEWMNG_Filp

【描述】

预览界面翻转

【语法】

```
int32_t CVI_LIVEVIEWMNG_Filp(uint32_t val);
```

【参数】

参数名称	描述	输入/输出
val	预览值	输入

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_liveviewmng.h
- 库文件: libcvi_liveviewmng.a/libcvi_liveviewmng.so

【注意】

无

【举例】

无

【相关主题】

无

3.7 Photomng

该通用组件用于拍照模式。

3.7.1 API 参考

该功能模块为用户提供以下 API:

- `CVI_PHOTOMNG_ContCallBack` : 注册拍照状态的回调函数
- `CVI_POHTOMNG_RegisterEvent` : 拍照状态注册

3.7.1.1 CVI_PHOTOMNG_ContCallBack

【描述】

注册拍照状态的回调函数

【语法】

```
int32_t CVI_PHOTOMNG_ContCallBack(CVI_PHOTO_EVENT_E event_type, const char_
↳ *filename, void *param);
```

【参数】

参数名称	描述	输入/输出
event_type	拍照状态类型	输入
filename	文件名	输入
param	状态参数	输入

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: `cvi_photomng.h`
- 库文件: `libcvi_photomng.a/libcvi_photomng.so`

【注意】

无

【举例】

无

【相关主题】

无

3.7.1.2 CVI_POHTOMNG_RegisterEvent

【描述】

拍照状态注册

【语法】

```
int32_t CVI_POHTOMNG_RegisterEvent(void);
```

【参数】

无

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_photomng.h
- 库文件: libcvi_photomng.a/libcvi_photomng.so

【注意】

无

【举例】

无

【相关主题】

无

3.8 Playbackmng

该通用组件用于回放模式。

3.8.1 API 参考

该功能模块为用户提供以下 API:

- `CVI_PLAYBACKMNG_EventCallBack`: 注册回放状态的回调函数
- `CVI_PLAYBACKMNG_RegisterEvent`: 回放状态注册

3.8.1.1 `CVI_PLAYBACKMNG_EventCallBack`

【描述】

注册回放状态的回调函数

【语法】

```
void CVI_PLAYBACKMNG_EventCallBack(CVI_PLAYER_SERVICE_HANDLE_T hdl, CVI_
↳PLAYER_SERVICE_EVENT_S *event_t);
```

【参数】

参数名称	描述	输入/输出
hdl	回放服务句柄	输入
event_t	回放事件	输入

【返回值】

无

【需求】

- 头文件: `cvi_playbackmng.h`
- 库文件: `libcvi_playbackmng.a/libcvi_playbackmng.so`

【注意】

无

【举例】

无

【相关主题】

无

3.8.1.2 CVI_PLAYBACKMNG_RegisterEvent

【描述】

回放状态注册

【语法】

```
int32_t CVI_PLAYBACKMNG_RegisterEvent(void);
```

【参数】

无

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_playbackmng.h
- 库文件: libcvi_playbackmng.a/libcvi_playbackmng.so

【注意】

无

【举例】

无

【相关主题】

无

3.8.2 数据类型

视频输入相关数据类型定义如下:

- CVI_PLAYER_SERVICE_EVENT_TYPE_E: 回放服务事件类型枚举
- CVI_PLAYER_SERVICE_EVENT_S: 回放服务事件

3.8.2.1 CVI_PLAYER_SERVICE_EVENT_TYPE_E

【说明】

回放服务事件类型枚举

【定义】

```
typedef enum {  
    CVI_PLAYER_SERVICE_EVENT_UNKNOWN,  
    CVI_PLAYER_SERVICE_EVENT_OPEN_FAILED,  
    CVI_PLAYER_SERVICE_EVENT_PLAY,  
    CVI_PLAYER_SERVICE_EVENT_PLAY_FINISHED,  
    CVI_PLAYER_SERVICE_EVENT_PLAY_PROGRESS,  
    CVI_PLAYER_SERVICE_EVENT_PAUSE,  
    CVI_PLAYER_SERVICE_EVENT_RESUME,  
    CVI_PLAYER_SERVICE_EVENT_RECOVER_START,  
    CVI_PLAYER_SERVICE_EVENT_RECOVER_PROGRESS,  
    CVI_PLAYER_SERVICE_EVENT_RECOVER_FAILED,  
    CVI_PLAYER_SERVICE_EVENT_RECOVER_FINISHED,  
} CVI_PLAYER_SERVICE_EVENT_TYPE_E;
```

【成员】

成员名称	描述
CVI_PLAYER_SERVICE_EVENT_UNKNOWN	未知事件
CVI_PLAYER_SERVICE_EVENT_OPEN_FAILED	打开文件失败
CVI_PLAYER_SERVICE_EVENT_PLAY	文件播放
CVI_PLAYER_SERVICE_EVENT_PLAY_PROGRESS	文件播放进度
CVI_PLAYER_SERVICE_EVENT_PLAY_FINISHED	文件播放结束
CVI_PLAYER_SERVICE_EVENT_PAUSE	文件播放暂停
CVI_PLAYER_SERVICE_EVENT_RESUME	文件播放暂停恢复
CVI_PLAYER_SERVICE_EVENT_RECOVER_START	开始文件修复
CVI_PLAYER_SERVICE_EVENT_RECOVER_PROGRESS	文件修复进度
CVI_PLAYER_SERVICE_EVENT_RECOVER_FAILED	文件修复失败
CVI_PLAYER_SERVICE_EVENT_RECOVER_FINISHED	文件修复完成

【注意事项】

无。

【相关数据类型及接口】

无。

3.8.2.2 CVI_PLAYER_SERVICE_EVENT_S

【说明】

回放服务事件

【定义】

```
typedef struct{
    CVI_PLAYER_SERVICE_EVENT_TYPE_E type;
    double value;
} CVI_PLAYER_SERVICE_EVENT_S;
```

【成员】

成员名称	描述
type	回放服务事件类型
value	保留

【注意事项】

无。

【相关数据类型及接口】

无。

3.9 Powercontrol

该通用组件用于电源管理。

3.9.1 API 参考

该功能模块为用户提供以下 API:

- CVI_POWERCTRL_Init : 电源控制初始化
- CVI_POWERCTRL_GetTaskAttr : 获取电源控制任务属性
- CVI_POWERCTRL_SetTaskAttr : 设置电源控制任务属性
- CVI_POWERCTRL_EventPreProc : 电源控制状态处理
- CVI_POWERCTRL_DeInit : 电源控制反初始化

3.9.1.1 CVI_POWERCTRL_Init

【描述】

电源控制初始化

【语法】

```
int32_t CVI_POWERCTRL_Init(const CVI_PWRCTRL_CFG_S* pstCfg);
```

【参数】

参数名称	描述	输入/输出
pstCfg	电源控制配置	输入

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_powercontrol.h
- 库文件: libcvi_powercontrol.a/libcvi_powercontrol.so

【注意】

无

【举例】

无

【相关主题】

无

3.9.1.2 CVI_POWERCTRL_GetTaskAttr

【描述】

获取电源控制任务属性

【语法】

```
int32_t CVI_POWERCTRL_GetTaskAttr(CVI_PWRCTRL_TASK_E enType, CVI_TIMEDTASK_↪ATTR_S* pstTaskAttr);
```

【参数】

参数名称	描述	输入/输出
enType	电源控制任务类型	输入
pstTaskAttr	任务属性	输出

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_powercontrol.h
- 库文件: libcvi_powercontrol.a/libcvi_powercontrol.so

【注意】

无

【举例】

无

【相关主题】

无

3.9.1.3 CVI_POWERCTRL_SetTaskAttr**【描述】**

设置电源控制任务属性

【语法】

```
int32_t CVI_POWERCTRL_SetTaskAttr(CVI_PWRCTRL_TASK_E enType, const CVI_
→TIMEDTASK_ATTR_S* pstTaskAttr);
```

【参数】

参数名称	描述	输入/输出
enType	电源控制任务类型	输入
pstTaskAttr	任务属性	输入

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_powercontrol.h
- 库文件: libcvi_powercontrol.a/libcvi_powercontrol.so

【注意】

无

【举例】

无

【相关主题】

无

3.9.1.4 CVI_POWERCTRL_EventPreProc

【描述】

电源控制状态处理

【语法】

```
int32_t CVI_POWERCTRL_EventPreProc(const CVI_PWRCTRL_EVENT_ATTR_S*  
    ↪ pstEventAttr, bool* pbEventContinueHandle);
```

【参数】

参数名称	描述	输入/输出
pstEventAttr	事件属性	输入
pbEventContinueHandle	事件继续句柄	输入

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_powercontrol.h
- 库文件: libcvi_powercontrol.a/libcvi_powercontrol.so

【注意】

无

【举例】

无

【相关主题】

无

3.9.1.5 CVI_POWERCTRL_DeInit

【描述】

电源控制反初始化

【语法】

```
int32_t CVI_POWERCTRL_DeInit(void);
```

【参数】

无

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件：cvi_powercontrol.h
- 库文件：libcvi_powercontrol.a/libcvi_powercontrol.so

【注意】

无

【举例】

无

【相关主题】

无

3.9.2 数据类型

视频输入相关数据类型定义如下：

- CVI_PWRCTRL_TASK_PROC_CALLBACK：休眠回调处理函数
- CVI_PWRCTRL_TASK_CFG_S：电源管理任务配置
- CVI_PWRCTRL_CFG_S：电源管理配置
- CVI_PWRCTRL_TASK_E：电源管理任务
- CVI_PWRCTRL_EVENT_TYPE_E：电源管理事件类型枚举
- CVI_PWRCTRL_WAKEUP_TACTICS_E：唤醒策略枚举
- CVI_PWRCTRL_EVENT_SCOPE_E：电源管理事件作用域枚举
- CVI_PWRCTRL_EVENT_COMMON_ATTR_S：电源管理普通类型事件属性
- CVI_PWRCTRL_EVENT_WAKEUP_ATTR_S：电源管理事件唤醒属性

- CVI_PWRCTRL_EVENT_CONTROL_E：电源管理事件控制枚举
- CVI_PWRCTRL_EVENT_CONTROL_ATTR_S：电源管理控制事件属性
- CVI_PWRCTRL_EVENT_ATTR_S：电源管理事件属性

3.9.2.1 CVI_PWRCTRL_TASK_PROC_CALLBACK

【说明】

休眠回调处理函数

【定义】

```
typedef int32_t (*CVI_PWRCTRL_TASK_PROC_CALLBACK)(void* pvPrivData);/**< dormant_  
↪callback process function */
```

【成员】

成员名称	描述
pvPrivData	参数

【注意事项】

无。

【相关数据类型及接口】

无。

3.9.2.2 CVI_PWRCTRL_TASK_CFG_S

【说明】

电源管理任务配置

【定义】

```
typedef struct cviPWRCTRL_TASK_CFG__S /**< task static attribute */{  
    CVI_TIMEDTASK_ATTR_S stAttr;  
    CVI_PWRCTRL_TASK_PROC_CALLBACK pfnDormantProc;  
    void* pvDormantPrivData;  
    CVI_PWRCTRL_TASK_PROC_CALLBACK pfnWakeupProc;  
    void* pvWakeupPrivData;  
} CVI_PWRCTRL_TASK_CFG_S;
```

【成员】

成员名称	描述
stAttr	定时器任务属性
pfnDormantProc	休眠回调处理函数
pvDormantPrivData	休眠回调处理函数参数
pfnWakeupProc	唤醒回调处理函数
pvWakeupPrivData	唤醒回调处理函数参数

【注意事项】

无。

【相关数据类型及接口】

无。

3.9.2.3 CVI_PWRCTRL_CFG_S

【说明】

电源管理配置

【定义】

```
typedef struct cviPWRCTRL_CFG_S{  
    CVI_PWRCTRL_TASK_CFG_S astTaskCfg[CVI_PWRCTRL_TASK_BUIT];  
} CVI_PWRCTRL_CFG_S;
```

【成员】

成员名称	描述
astTaskCfg	电源控制任务配置数组

【注意事项】

无。

【相关数据类型及接口】

无。

3.9.2.4 CVI_PWRCTRL_TASK_E

【说明】

电源管理任务

【定义】

```
typedef enum cviPWRCTRL_TASK_E /**<pwrctrl task type */{  
    CVI_PWRCTRL_TASK_SCREENORMANT = 0,  
    CVI_PWRCTRL_TASK_SYSTEMDORMANT,  
    CVI_PWRCTRL_TASK_BUIT  
}CVI_PWRCTRL_TASK_E;
```

【成员】

成员名称	描述
CVI_PWRCTRL_TASK_SCREENORMANT	屏幕休眠
CVI_PWRCTRL_TASK_SYSTEMDORMANT	系统休眠

【注意事项】

无。

【相关数据类型及接口】

无。

3.9.2.5 CVI_PWRCTRL_EVENT_TYPE_E

【说明】

电源管理事件类型枚举

【定义】

```
typedef enum cviPWRCTRL_EVENT_TYPE_E /**<event type */{
    CVI_PWRCTRL_EVENT_TYPE_WAKEUP = 0,    /**<wakeup dormant*/
    CVI_PWRCTRL_EVENT_TYPE_CONTROL,        /**<pause or resumme dormant check */
    CVI_PWRCTRL_EVENT_TYPE_COMMON,         /**<general event,no both of wakeup and...
→control function */
    CVI_PWRCTRL_EVENT_TYPE_BUIT
}CVI_PWRCTRL_EVENT_TYPE_E;
```

【成员】

成员名称	描述
CVI_PWRCTRL_EVENT_TYPE_WAKEUP	唤醒类型
CVI_PWRCTRL_EVENT_TYPE_CONTROL	控制类型
CVI_PWRCTRL_EVENT_TYPE_COMMON	普通类型，除唤醒类型和控制类型

【注意事项】

无。

【相关数据类型及接口】

无。

3.9.2.6 CVI_PWRCTRL_WAKEUP_TACTICS_E

【说明】

唤醒策略枚举

【定义】

```
typedef enum cviPWRCTRL_WAKEUP_TACTICS_E /**< wakeup event process tactics type,
→whether event need continue proc or not*/{
    /**<after dong wakeup,event need not continue to proc*/
    CVI_PWRCTRL_WAKEUP_TACTICS_DISCARD = 0,
    /**<after dong wakeup,event need continue to proc*/
    CVI_PWRCTRL_WAKEUP_TACTICS_CONTINUE,
    CVI_PWRCTRL_WAKEUP_TACTICS_BUIT
}CVI_PWRCTRL_WAKEUP_TACTICS_E;
```

【成员】

成员名称	描述
CVI_PWRCTRL_WAKEUP_TACTICS_DISARM	唤醒ID事件不处理
CVI_PWRCTRL_WAKEUP_TACTICS_CONARM	唤醒ID事件处理

【注意事项】

无。

【相关数据类型及接口】

无。

3.9.2.7 CVI_PWRCTRL_EVENT_SCOPE_E

【说明】

电源管理事件作用域枚举

【定义】

```
typedef enum cviPWRCTRL_EVENT_SCOPE_E /**< event action scope */{
    /**control event: control(pause or resume) system dormant check at normal state(no system
    dormant) common event: do not continue handle the event at sys dormant
    wakeup event: wakeup sstem at sys dormant */
    CVI_PWRCTRL_EVENT_SCOPE_SYSTEM = 0,
    /**control event: control(pause or resume) system dormant and screen dormant check at normal state
    common event: do not continue handle the event at sys dormant or screen dormant
    wakeup event: wakeup sstem at sys dormant or screen dormant */
    CVI_PWRCTRL_EVENT_SCOPE_SYSTEM_SCREEN,
    CVI_PWRCTRL_EVENT_SCOPE_BUIT
}CVI_PWRCTRL_EVENT_SCOPE_E;
```

【成员】

成员名称	描述
CVI_PWRCTRL_EVENT_SCOPE_SYSTEM	系统作用域
CVI_PWRCTRL_EVENT_SCOPE_SYSTEM_SCREEN	系统屏幕作用域

【注意事项】

作用域具体内容见英文解释

【相关数据类型及接口】

无。

3.9.2.8 CVI_PWRCTRL_EVENT_COMMON_ATTR_S

【说明】

电源管理普通类型事件属性

【定义】

```
typedef struct cviPWRCTRL_EVENT_COMMON_ATTR_S/**< common event attribute */{  
    CVI_PWRCTRL_EVENT_SCOPE_E enType;  
    bool bResetTimer;  
} CVI_PWRCTRL_EVENT_COMMON_ATTR_S
```

【成员】

成员名称	描述
enType	电源控制事件作用域
bResetTimer	重置定时器使能

【注意事项】

无。

【相关数据类型及接口】

无。

3.9.2.9 CVI_PWRCTRL_EVENT_WAKEUP_ATTR_S

【说明】

电源管理事件唤醒属性

【定义】

```
typedef struct cviPWRCTRL_EVENT_WAKEUP_ATTR_S/**< wakeup event attribute */{  
    CVI_PWRCTRL_WAKEUP_TACTICS_E enType;  
    CVI_PWRCTRL_EVENT_COMMON_ATTR_S stCommonCfg;  
} CVI_PWRCTRL_EVENT_WAKEUP_ATTR_S;
```

【成员】

成员名称	描述
enType	唤醒策略
stCommonCfg	电源控制普通类型事件属性

【注意事项】

无。

【相关数据类型及接口】

无。

3.9.2.10 CVI_PWRCTRL_EVENT_CONTROL_E

【说明】

电源管理事件控制枚举

【定义】

```
typedef enum cviPWRCTRL_EVENT_CONTROL_E/**< control event type of action*/{  
    CVI_PWRCTRL_EVENT_CONTROL_PAUSE = 0,**<after wakeup ,ui need not to proc*/  
    CVI_PWRCTRL_EVENT_CONTROL_RESUME,**<after wakeup ,ui continue to proc*/  
    CVI_PWRCTRL_EVENT_CONTROL_BUIT  
}CVI_PWRCTRL_EVENT_CONTROL_E;
```

【成员】

成员名称	描述
CVI_PWRCTRL_EVENT_CONTROL_PAUSE	唤醒后 ui 不处理事件
CVI_PWRCTRL_EVENT_CONTROL_RESUME	唤醒后 ui 继续处理事件

【注意事项】

无。

【相关数据类型及接口】

无。

3.9.2.11 CVI_PWRCTRL_EVENT_CONTROL_ATTR_S

【说明】

电源管理控制事件属性

【定义】

```
typedef struct cviPWRCTRL_EVENT_CONTROL_ATTR_S/**< control event attribute*/{  
    CVI_PWRCTRL_EVENT_CONTROL_E enType;  
    CVI_PWRCTRL_EVENT_COMMON_ATTR_S stCommonCfg;  
} CVI_PWRCTRL_EVENT_CONTROL_ATTR_S;
```

【成员】

成员名称	描述
enType	电源管理事件控制枚举类型
CVI_PWRCTRL_EVENT_COMMON_ATTR_S	电源管理普通类型事件属性

【注意事项】

无。

【相关数据类型及接口】

无。

3.9.2.12 CVI_PWRCTRL_EVENT_ATTR_S

【说明】

电源管理事件属性

【定义】

```
typedef struct cviPWRCTRL_EVENT_ATTR_S /**< event configure */
{
    CVI_PWRCTRL_EVENT_TYPE_E enType;
    union tagPWRCTRL_EVENT_ATTR_U
    {
        CVI_PWRCTRL_EVENT_WAKEUP_ATTR_S stWakeupCfg; /**< wakeup event cfg */
        CVI_PWRCTRL_EVENT_CONTROL_ATTR_S stCtrlCfg; /**< control event cfg */
        CVI_PWRCTRL_EVENT_COMMON_ATTR_S stCommonCfg;
    } unCfg;
} CVI_PWRCTRL_EVENT_ATTR_S;
```

【成员】

成员名称	描述
enType	电源管理事件类型
unCfg	电源管理事件属性枚举

【注意事项】

无。

【相关数据类型及接口】

无。

3.10 Recordmng

该通用组件用于录像管理。

3.10.1 API 参考

该功能模块为用户提供以下 API:

- CVI_RECORDERMNG_EventCallBack : 录像事件回调函数, 目前用于紧急录像
- CVI_RECORDERMNG_ContCallBack : 录像事件回调函数
- CVI_RECORDERMNG_RegisterEvent : 录像状态注册

3.10.1.1 CVI_RECORDMNG_EventCallBack

【描述】

录像事件回调函数，目前用于紧急录像

【语法】

```
int32_t CVI_RECORDMNG_EventCallBack(CVI_REC_EVENT_E event_type, const char_
↪ *filename, void *param);
```

【参数】

参数名称	描述	输入/输出
event_type	录像类型	输入
filename	文件名	输入
param	事件参数	输入

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件：cvi_recordmng.h
- 库文件：libcvi_recordmng.a/libcvi_recordmng.so

【注意】

无

【举例】

无

【相关主题】

无

3.10.1.2 CVI_RECORDMNG_ContCallBack

【描述】

录像事件回调函数

【语法】

```
int32_t CVI_RECORDMNG_ContCallBack(CVI_REC_EVENT_E event_type, const char_
↪ *filename, void *param);
```

【参数】

参数名称	描述	输入/输出
event_type	录像类型	输入
filename	文件名	输入
param	事件参数	输入

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_recordmng.h
- 库文件: libcvi_recordmng.a/libcvi_recordmng.so

【注意】

无

【举例】

无

【相关主题】

无

3.10.1.3 CVI_RECORDMNG_RegisterEvent

【描述】

录像状态注册

【语法】

```
int32_t CVI_RECORDMNG_RegisterEvent(void);
```

【参数】

无

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_recordmng.h
- 库文件: libcvi_recordmng.a/libcvi_recordmng.so

【注意】

无

【举例】

无

【相关主题】

无

3.11 Storagemng

该通用组件用于内存管理。

3.11.1 API 参考

该功能模块为用户提供以下 API:

- `CVI_STORAGEMNG_RegisterEvent` : 存储状态注册
- `CVI_STORAGEMNG_Create` : 创建存储管理
- `CVI_STORAGEMNG_Destroy` : 销毁存储管理
- `CVI_STORAGEMNG_GetFSInfo` : 获取文件系统信息
- `CVI_STORAGEMNG_GetInfo` : 获取存储设备信息
- `CVI_STORAGEMNG_Format` : 格式化
- `CVI_STORAGEMNG_Mount` : 挂载设备
- `CVI_STORAGEMNG_Umount` : 卸载设备

3.11.1.1 `CVI_STORAGEMNG_RegisterEvent`

【描述】

存储状态注册

【语法】

```
int32_t CVI_STORAGEMNG_RegisterEvent(void);
```

【参数】

无

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_storagemng.h
- 库文件: libcvi_storagemng.a/libcvi_storagemng.so

【注意】

无

【举例】

无

【相关主题】

无

3.11.1.2 CVI_STORAGEMNG_Create

【描述】

创建存储管理

【语法】

```
int32_t CVI_STORAGEMNG_Create(STG_DEVINFO_S *devinfo);
```

【参数】

参数名称	描述	输入/输出
devinfo	存储设备信息	输入

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_storagemng.h
- 库文件: libcvi_storagemng.a/libcvi_storagemng.so

【注意】

无

【举例】

无

【相关主题】

无

3.11.1.3 CVI_STORAGEMNG_Destroy

【描述】

销毁存储管理

【语法】

```
int32_t CVI_STORAGEMNG_Destroy(void);
```

【参数】

无

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_storagemng.h
- 库文件: libcvi_storagemng.a/libcvi_storagemng.so

【注意】

无

【举例】

无

【相关主题】

无

3.11.1.4 CVI_STORAGEMNG_GetFSInfo

【描述】

获取文件系统信息

【语法】

```
int32_t CVI_STORAGEMNG_GetFSInfo(STG_FS_INFO_S *pstFSInfo);
```

【参数】

参数名称	描述	输入/输出
pstFSInfo	文件系统信息	输出

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_storagemng.h
- 库文件: libcvi_storagemng.a/libcvi_storagemng.so

【注意】

无

【举例】

无

【相关主题】

无

3.11.1.5 CVI_STORAGEMNG_GetInfo

【描述】

获取存储设备信息

【语法】

```
int32_t CVI_STORAGEMNG_GetInfo(CVI_STG_DEV_INFO_S *pstInfo);
```

【参数】

参数名称	描述	输入/输出
pstInfo	存储设备信息	输出

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_storagemng.h
- 库文件: libcvi_storagemng.a/libcvi_storagemng.so

【注意】

无

【举例】

无

【相关主题】

无

3.11.1.6 CVI_STORAGEMNG_Format**【描述】**

格式化

【语法】

```
int32_t CVI_STORAGEMNG_Format(char *labelname);
```

【参数】

参数名称	描述	输入/输出
labelname	标签名	输入

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_storagemng.h
- 库文件: libcvi_storagemng.a/libcvi_storagemng.so

【注意】

无

【举例】

无

【相关主题】

无

3.11.1.7 CVI_STORAGEMNG_Mount**【描述】**

挂载设备

【语法】

```
int32_t CVI_STORAGEMNG_Mount(void);
```

【参数】

无

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_storagemng.h
- 库文件: libcvi_storagemng.a/libcvi_storagemng.so

【注意】

无

【举例】

无

【相关主题】

无

3.11.1.8 CVI_STORAGEMNG_Umount

【描述】

卸载设备

【语法】

```
int32_t CVI_STORAGEMNG_Umount(void);
```

【参数】

无

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_storagemng.h
- 库文件: libcvi_storagemng.a/libcvi_storagemng.so

【注意】

无

【举例】

无

【相关主题】

无

3.12 System

该通用组件用于系统设置。

3.12.1 API 参考

该功能模块为用户提供以下 API:

- CVI_SYSTEM_SetDateTime : 设置系统时间
- CVI_SYSTEM_GetRTCDateTime : 获取 RTC 时间 (硬件时间)
- CVI_SYSTEM_SetDefaultDateTime : 默认时间设置
- CVI_SYSTEM_Reboot : 系统重启
- CVI_SYSTEM_GetStartupWakeupSource : 获取启动唤醒源
- CVI_SYSTEM_BootSound : 系统启动声播放
- CVI_SYSTEM_SetGpioWakeup : 设置唤醒源

3.12.1.1 CVI_SYSTEM_SetDateTime

【描述】

设置系统时间

【语法】

```
int32_t CVI_SYSTEM_SetDateTime(const CVI_SYSTEM_TM_S* pstDateTime);
```

【参数】

参数名称	描述	输入/输出
pstDateTime	系统时间	输入

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_system.h
- 库文件: libcvi_system.a/libcvi_system.so

【注意】

无

【举例】

无

【相关主题】

无

3.12.1.2 CVI_SYSTEM_GetRTCDateTime

【描述】

获取 RTC 时间 (硬件时间)

【语法】

```
int32_t CVI_SYSTEM_GetRTCDateTime(CVI_SYSTEM_TM_S* pstDateTime);
```

【参数】

参数名称	描述	输入/输出
pstDateTime	系统时间	输出

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_system.h
- 库文件: libcvi_system.a/libcvi_system.so

【注意】

无

【举例】

无

【相关主题】

无

3.12.1.3 CVI_SYSTEM_SetDefaultDateTime

【描述】

默认时间设置

【语法】

```
int32_t CVI_SYSTEM_SetDefaultDateTime(void);
```

【参数】

无

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_system.h
- 库文件: libcvi_system.a/libcvi_system.so

【注意】

无

【举例】

无

【相关主题】

无

3.12.1.4 CVI_SYSTEM_Reboot

【描述】

系统重启

【语法】

```
void CVI_SYSTEM_Reboot(int32_t need_sync);
```

【参数】

参数名称	描述	输入/输出
need_sync	数据同步	输入

【返回值】

无

【需求】

- 头文件: cvi_system.h
- 库文件: libcvi_system.a/libcvi_system.so

【注意】

无

【举例】

无

【相关主题】

无

3.12.1.5 CVI_SYSTEM_BootSound

【描述】

启动系统引导声

【语法】

```
int32_t CVI_SYSTEM_BootSound(CVI_MAPI_AO_HANDLE_T AoHdl);
```

【参数】

参数名称	描述	输入/输出
AoHdl	音频输出句柄	输入

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_system.h
- 库文件: libcvi_system.a/libcvi_system.so

【注意】

无

【举例】

无

【相关主题】

无

3.12.1.6 CVI_SYSTEM_GetStartupWakeupSource

【描述】

获取启动唤醒源

【语法】

```
int32_t CVI_SYSTEM_GetStartupWakeupSource(CVI_SYSTEM_STARTUP_SRC_E*  
↪ penStartupSrc);
```

【参数】

参数名称	描述	输入/输出
penStartupSrc	唤醒源	输出

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_system.h
- 库文件: libcvi_system.a/libcvi_system.so

【注意】

无

【举例】

无

【相关主题】

无

3.12.1.7 CVI_SYSTEM_SetGpioWakeup

【描述】

设置唤醒源

【语法】

```
int32_t CVI_SYSTEM_SetGpioWakeup(CVI_GPIO_NUM_E gpio, uint32_t value);
```

【参数】

参数名称	描述	输入/输出
gpio	GPIO 端口	输入
value	GPIO 值	输入

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_system.h
- 库文件: libcvi_system.a/libcvi_system.so

【注意】

无

【举例】

无

【相关主题】

无

3.12.2 数据类型

视频输入相关数据类型定义如下:

- CVI_SYSTEM_STARTUP_SRC_E : 启动源枚举
- CVI_SYSTEM_TM_S : 定义系统时间结构体

3.12.2.1 CVI_SYSTEM_STARTUP_SRC_E

【说明】

启动源枚举

【定义】

```
typedef enum cviSYSTEM_STARTUP_SRC_E{
    CVI_SYSTEM_STARTUP_SRC_GSENSORWAKEUP = 0, /* gsensor wake up*/
    CVI_SYSTEM_STARTUP_SRC_USBWAKEUP,          /* usb wake up*/
    CVI_SYSTEM_STARTUP_SRC_STARTUP,             /* power key start up*/
    CVI_SYSTEM_STARTUP_SRC_BUTTON
} CVI_SYSTEM_STARTUP_SRC_E;
```

【成员】

成员名称	描述
CVI_SYSTEM_STARTUP_SRC_GSENSORWAKEUP	加速度计唤醒源
CVI_SYSTEM_STARTUP_SRC_USBWAKEUP	USB 唤醒源
CVI_SYSTEM_STARTUP_SRC_STARTUP	电源键启动

【注意事项】

无。

【相关数据类型及接口】

无。

3.12.2.2 CVI_SYSTEM_TM_S

【说明】

定义系统时间结构体

【定义】

```
typedef struct tagCVI_SYSTEM_TM_S {  
    int32_t s32year;  
    int32_t s32mon;  
    int32_t s32mday;  
    int32_t s32hour;  
    int32_t s32min;  
    int32_t s32sec;  
} CVI_SYSTEM_TM_S;
```

【成员】

成员名称	描述
s32year	年份
s32mon	月
s32mday	日
s32hour	时
s32min	分
s32sec	秒

【注意事项】

无。

【相关数据类型及接口】

无。

3.13 Utils

实用工具，包含 MD5 算法和定时器任务。

3.13.1 API 参考

该功能模块为用户提供以下 API:

- CVI_MD5_Init : MD5 初始化
- CVI_MD5_Update : MD5 更新
- CVI_MD5_Final : MD5 最终结果
- CVI_MD5_Transform : MD5 转换
- CVI_MD5_Encode : MD5 编码
- CVI_MD5_Decode : MD5 解码
- CVI_TIMEDTASK_Init : 定时器任务初始化
- CVI_TIMEDTASK_DeInit : 定时器任务反初始化
- CVI_TIMEDTASK_Create : 定时器任务创建
- CVI_TIMEDTASK_Destroy : 定时器任务销毁
- CVI_TIMEDTASK_GetAttr 获取定时器任务属性
- CVI_TIMEDTASK_SetAttr : 设置定时器任务属性
- CVI_TIMEDTASK_ResetTime : 重置时间

3.13.1.1 CVI_MD5_Init

【描述】

MD5 初始化

【语法】

```
void CVI_MD5_Init(CVI_MD5_CTX_S *context);
```

【参数】

参数名称	描述	输入/输出
context	MD5 上下文	输入

【返回值】

无

【需求】

- 头文件: md5.h
- 库文件: libcvi_md5.a/libcvi_md5.so

【注意】

无

【举例】

无

【相关主题】

无

3.13.1.2 CVI_MD5_Update

【描述】

MD5 更新

【语法】

```
void CVI_MD5_Update(CVI_MD5_CTX_S *context, unsigned char *input, uint32_t inputlen);
```

【参数】

参数名称	描述	输入/输出
context	MD5 上下文	输入
input	输入	输入
inputlen	输入长度	输入

【返回值】

无

【需求】

- 头文件: md5.h
- 库文件: libcvi_md5.a/libcvi_md5.so

【注意】

无

【举例】

无

【相关主题】

无

3.13.1.3 CVI_MD5_Final

【描述】

MD5 最终结果

【语法】

```
void CVI_MD5_Final(CVI_MD5_CTX_S *context, unsigned char digest[16]);
```

【参数】

参数名称	描述	输入/输出
context	MD5 上下文	输入
digest	摘要	输出

【返回值】

无

【需求】

- 头文件: md5.h
- 库文件: libcvi_md5.a/libcvi_md5.so

【注意】

无

【举例】

无

【相关主题】

无

3.13.1.4 CVI_MD5_Transform

【描述】

MD5 转换

【语法】

```
void CVI_MD5_Transform(uint32_t state[4], unsigned char block[64]);
```

【参数】

参数名称	描述	输入/输出
state	状态	输出
block	缓存块	输入

【返回值】

无

【需求】

- 头文件: md5.h
- 库文件: libcvi_md5.a/libcvi_md5.so

【注意】

无

【举例】

无

【相关主题】

无

3.13.1.5 CVI_MD5_Encode

【描述】

MD5 编码

【语法】

```
void CVI_MD5_Encode(unsigned char *output, uint32_t *input, uint32_t len);
```

【参数】

参数名称	描述	输入/输出
output	输出	输出
input	输入	输入
len	长度	输入

【返回值】

无

【需求】

- 头文件: md5.h
- 库文件: libcvi_md5.a/libcvi_md5.so

【注意】

无

【举例】

无

【相关主题】

无

3.13.1.6 CVI_MD5_Decode

【描述】

MD5 解码

【语法】

```
void CVI_MD5_Decode(uint32_t *output, unsigned char *input, uint32_t len);
```


【参数】

参数名称	描述	输入/输出
output	输出	输出
input	输入	输入
len	长度	输入

【返回值】

无

【需求】

- 头文件: md5.h
- 库文件: libcvi_md5.a/libcvi_md5.so

【注意】

无

【举例】

无

【相关主题】

无

3.13.1.7 CVI_TIMEDTASK_DeInit

【描述】

定时器任务反初始化

【语法】

```
int32_t CVI_TIMEDTASK_DeInit(void);
```

【参数】

无

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_timedtask.h
- 库文件: libcvi_timedtask.a/libcvi_timedtask.so

【注意】

无

【举例】

无

【相关主题】

无

3.13.1.8 CVI_TIMEDTASK_init

【描述】

定时器任务初始化

【语法】

```
int32_t CVI_TIMEDTASK_init(void);
```

【参数】

无

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_timedtask.h
- 库文件: libcvi_timedtask.a/libcvi_timedtask.so

【注意】

无

【举例】

无

【相关主题】

无

3.13.1.9 CVI_TIMEDTASK_Create

【描述】

定时器任务创建

【语法】

```
int32_t CVI_TIMEDTASK_Create(const CVI_TIMEDTASK_CFG_S *pstTimeTskCfg, uint32_t  
→ *pTimeTskid);
```

【参数】

参数名称	描述	输入/输出
pstTimeTskCfg	定时器任务配置	输入
pTimeTskid	定时器任务 id	输入

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_timedtask.h
- 库文件: libcvi_timedtask.a/libcvi_timedtask.so

【注意】

无

【举例】

无

【相关主题】

无

3.13.1.10 CVI_TIMEDTASK_Destroy

【描述】

定时器任务销毁

【语法】

```
int32_t CVI_TIMEDTASK_Destroy(uint32_t TimeTskid);
```

【参数】

参数名称	描述	输入/输出
TimeTskid	定时器任务 id	输入

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_timedtask.h
- 库文件: libcvi_timedtask.a/libcvi_timedtask.so

【注意】

无

【举例】

无

【相关主题】

无

3.13.1.11 CVI_TIMEDTASK_GetAttr

【描述】

获取定时器任务属性

【语法】

```
int32_t CVI_TIMEDTASK_GetAttr(uint32_t TimeTskid, CVI_TIMEDTASK_ATTR_S_
↪ *pstTimeTskAttr);
```

【参数】

参数名称	描述	输入/输出
TimeTskid	定时器任务 id	输入
pstTimeTskAttr	定时器任务属性	输出

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_timedtask.h
- 库文件: libcvi_timedtask.a/libcvi_timedtask.so

【注意】

无

【举例】

无

【相关主题】

无

3.13.1.12 CVI_TIMEDTASK_SetAttr

【描述】

设置定时器任务属性

【语法】

```
int32_t CVI_TIMEDTASK_SetAttr(uint32_t TimeTskid, const CVI_TIMEDTASK_ATTR_S_
→*pstTimeTskAttr);
```

【参数】

参数名称	描述	输入/输出
TimeTskid	定时器任务 id	输入
pstTimeTskAttr	定时器任务属性	输入

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_timedtask.h
- 库文件: libcvi_timedtask.a/libcvi_timedtask.so

【注意】

无

【举例】

无

【相关主题】

无

3.13.1.13 CVI_TIMEDTASK_ResetTime

【描述】

重置时间

【语法】

```
int32_t CVI_TIMEDTASK_ResetTime(uint32_t TimeTskid);
```

【参数】

参数名称	描述	输入/输出
TimeTskid	定时器任务 id	输入

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_timedtask.h
- 库文件: libcvi_timedtask.a/libcvi_timedtask.so

【注意】

无

【举例】

无

【相关主题】

无

3.13.2 数据类型

视频输入相关数据类型定义如下:

- CVI_MD5_CTX_S : 定义 MD5 上下文
- CVI_TIMEDTASK_ATTR_S : 定时器任务属性
- CVI_TIMEDTASK_CFG_S : 定时器任务配置

3.13.2.1 CVI_MD5_CTX_S

【说明】

定义 MD5 上下文

【定义】

```
typedef struct
{
    uint32_t count[2];
    uint32_t state[4];
    unsigned char buffer[64];
} CVI_MD5_CTX_S;
```

【成员】

成员名称	描述
count	数量
state	状态
buffer	缓存块

【注意事项】

无。

【相关数据类型及接口】

无。

3.13.2.2 CVI_TIMEDTASK_ATTR_S

【说明】

定时器任务属性

【定义】

```
typedef struct cviTIMEDTASK_ATTR_S {
    bool bEnable;
    uint32_t u32Time_sec; /**<timed-task trigger time, canbe reset */
    bool periodic; /**< periodic or ont-shot */
} CVI_TIMEDTASK_ATTR_S;
```

【成员】

成员名称	描述
bEnable	使能
u32Time_sec	时间间隔
periodic	周期使能

【注意事项】

无。

【相关数据类型及接口】

无。

3.13.2.3 CVI_TIMEDTASK_CFG_S**【说明】**

定时器任务配置

【定义】

```
typedef struct cviTIMEDTASK_CFG_S {  
    CVI_TIMEDTASK_ATTR_S stAttr;  
    CVI_TIMER_PROC *timerProc;  
    void *pvPrivData;  
} CVI_TIMEDTASK_CFG_S;
```

【成员】

成员名称	描述
stAttr	定时器任务属性
stattimerProce	定时器处理
pvPrivData	定时器处理参数

【注意事项】

无。

【相关数据类型及接口】

无。

3.14 Videomd

该通用组件用于运动检测。

3.14.1 API 参考

该功能模块为用户提供以下 API:

- CVI_MOTION_DETECT_SetState : 设置运动状态
- CVI_MOTION_DETECT_Init : 运动检测初始化
- CVI_MOTION_DETECT_DeInit : 运动检测反初始化

3.14.1.1 CVI_MOTION_DETECT_SetState

【描述】

设置运动状态

【语法】

```
void CVI_MOTION_DETECT_SetState(int32_t id, int32_t en);
```

【参数】

参数名称	描述	输入/输出
id	ID	输入
en	使能	输入

【返回值】

无

【需求】

- 头文件: cvi_videomd.h

【注意】

无

【举例】

无

【相关主题】

无

3.14.1.2 CVI_MOTION_DETECT_Init

【描述】

运动检测初始化

【语法】

```
int32_t CVI_MOTION_DETECT_Init(CVI_MOTION_DETECT_ATTR_S *attr);
```

【参数】

参数名称	描述	输入/输出
attr	运动探测属性	输入

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_videomd.h

【注意】

无

【举例】

无

【相关主题】

无

3.14.1.3 CVI_MOTION_DETECT_DeInit

【描述】

运动检测反初始化

【语法】

```
void CVI_MOTION_DETECT_DeInit(int32_t id);
```

【参数】

参数名称	描述	输入/输出
id	ID	输入

【返回值】

无

【需求】

- 头文件: cvi_videomd.h

【注意】

无

【举例】

无

【相关主题】

无

3.14.2 数据类型

视频输入相关数据类型定义如下：

- CVI_MOTION_DETECT_ATTR_S：运动检测属性

3.14.2.1 CVI_MOTION_DETECT_ATTR_S

【说明】

运动检测属性

【定义】

```
typedef struct _CVI_MOTION_DETECT_ATTR_S{  
    int32_t      camid;  
    int32_t      state;  
    int32_t      threshold;  
    CVI_MAPI_VPROC_HANDLE_T  vprocHandle;  
    uint32_t      vprocChnId;  
    uint32_t      isExtVproc;  
    uint32_t      w;  
    uint32_t      h;  
} CVI_MOTION_DETECT_ATTR_S;
```

【成员】

成员名称	描述
camid	cam id
state	状态
threshold	阈值
vprocHandle	vpss 句柄
vprocChnId	vpss 通道 id
isExtVproc	是否是扩展 vpss
w	宽度
h	高度

【注意事项】

无。

【相关数据类型及接口】

无。

3.15 Volmng

该通用组件用于音量管理。

3.15.1 API 参考

该功能模块为用户提供以下 API:

- `CVI_VOICEPLAY_Init` : 声音播放初始化
- `CVI_VOICEPLAY_DeInit` : 声音播放反初始化
- `CVI_VOICEPLAY_Push` : 将声音放入播放队列
- `CVI_VOICEPLAY_SetAmplifier` : 声音增益使能
- `CVI_VOICEPLAY_SetAmplifierFlage` : 声音增益标志使能
- `CVI_VOICEPLAY_SetVolume` : 设置音量
- `CVI_VOICEPLAY_GetVolume` : 获取音量

3.15.1.1 `CVI_VOICEPLAY_Init`

【描述】

声音播放初始化

【语法】

```
int32_t CVI_VOICEPLAY_Init(const CVI_VOICEPLAY_CFG_S* pstCfg);
```

【参数】

参数名称	描述	输入/输出
pstCfg	声音播放配置	输入

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: `cvi_volmd.h`
- 库文件: `libcvi_volmng.a/libcvi_volmng.so`

【注意】

无

【举例】

无

【相关主题】

无

3.15.1.2 CVI_VOICEPLAY_DeInit**【描述】**

声音播放反初始化

【语法】

```
int32_t CVI_VOICEPLAY_DeInit(void);
```

【参数】

无

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_volmd.h
- 库文件: libcvi_voymng.a/libcvi_voymng.so

【注意】

无

【举例】

无

【相关主题】

无

3.15.1.3 CVI_VOICEPLAY_Push**【描述】**

放声音进播放队列

【语法】

```
int32_t CVI_VOICEPLAY_Push(const CVI_VOICEPLAY_VOICE_S* pstVoice, int32_t_  
↪ u32Timeout_ms);
```

【参数】

参数名称	描述	输入/输出
pstVoice	声音	输入
u32Timeout_ms	超时时间	输入

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_volmd.h
- 库文件: libcvi_voimng.a/libcvi_voimng.so

【注意】

无

【举例】

无

【相关主题】

无

3.15.1.4 CVI_VOICEPLAY_SetAmplifier

【描述】

声音增益使能

【语法】

```
int32_t CVI_VOICEPLAY_SetAmplifier(bool en);
```

【参数】

参数名称	描述	输入/输出
en	使能	输入

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: `cvi_volmd.h`
- 库文件: `libcvi_voimng.a/libcvi_voimng.so`

【注意】

无

【举例】

无

【相关主题】

无

3.15.1.5 CVI_VOICEPLAY_SetAmplifierFlage

【描述】

声音增益标志使能

【语法】

```
int32_t CVI_VOICEPLAY_SetAmplifierFlage(bool en);
```

【参数】

参数名称	描述	输入/输出
en	使能	输入

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: `cvi_volmd.h`
- 库文件: `libcvi_voimng.a/libcvi_voimng.so`

【注意】

无

【举例】

无

【相关主题】

无

3.15.1.6 CVI_VOICEPLAY_SetVolume

【描述】

设置音量

【语法】

```
int32_t CVI_VOICEPLAY_SetVolume(int32_t volume);
```

【参数】

参数名称	描述	输入/输出
volume	音量	输入

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_volmd.h
- 库文件: libcvi_volumng.a/libcvi_volumng.so

【注意】

无

【举例】

无

【相关主题】

无

3.15.1.7 CVI_VOICEPLAY_GetVolume

【描述】

获取音量

【语法】

```
int32_t CVI_VOICEPLAY_GetVolume(int32_t *volume);
```

【参数】

参数名称	描述	输入/输出
volume	音量	输出

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_volmd.h
- 库文件: libcvi_voimng.a/libcvi_voimng.so

【注意】

无

【举例】

无

【相关主题】

无

3.15.2 数据类型

视频输入相关数据类型定义如下:

- CVI_VOICEPLAY_VOICE_S : 定义播放声音结构体
- CVI_VOICEPLAY_VOICETABLE_S : 定义播放声音列表
- CVI_VOICEPLAY_CFG_S : 定义播放声音配置
- CVI_VOICEPLAY_AOUT_OPT_S : 定义声音播放选择
- CVI_VOICE_MAX_SEGMENT_CNT : 定义最大声音种类数

3.15.2.1 CVI_VOICE_MAX_SEGMENT_CNT

【说明】

定义最大声音种类数

【定义】

```
#define CVI_VOICE_MAX_SEGMENT_CNT (5)
```

【成员】

无

【注意事项】

无。

【相关数据类型及接口】

无。

3.15.2.2 CVI_VOICEPLAY_VOICE_S

【说明】

定义播放声音结构体

【定义】

```
typedef struct _CVI_VOICEPLAY_VOICE_S{  
    uint32_t volume;  
    uint32_t u32VoiceCnt;  
    uint32_t au32VoiceIdx[CVI_VOICE_MAX_SEGMENT_CNT];  
    bool bDroppable;  
} CVI_VOICEPLAY_VOICE_S;
```

【成员】

成员名称	描述
volume	音量
u32VoiceCnt	声音数
au32VoiceIdx	声音 id 数组
bDroppable	丢声使能

【注意事项】

无。

【相关数据类型及接口】

无。

3.15.2.3 CVI_VOICEPLAY_VOICETABLE_S

【说明】

定义播放声音结构体

【定义】

```
typedef struct _CVI_VOICEPLAY_VOICETABLE_S  
{  
    uint32_t u32VoiceIdx;  
    char aszFilePath[CVI_APPCOMM_MAX_PATH_LEN];  
} CVI_VOICEPLAY_VOICETABLE_S
```

【成员】

成员名称	描述
u32VoiceIdx	声音 id
aszFilePath	声音存放路径

【注意事项】

无。

【相关数据类型及接口】

无。

3.15.2.4 CVI_VOICEPLAY_AOUT_OPT_S**【说明】**

定义声音播放选择

【定义】

```
typedef struct _CVI_LITEPLAYER_AOUT_OPT_S {  
    void * hAudDevHdl;    // device id  
    void * hAudTrackHdl;  // chn id  
} CVI_VOICEPLAY_AOUT_OPT_S;
```

【成员】

成员名称	描述
hAudDevHdl	设备句柄
hAudTrackHdl	声道句柄

【注意事项】

无。

【相关数据类型及接口】

无。

3.15.2.5 CVI_VOICEPLAY_CFG_S**【说明】**

定义播放声音配置

【定义】

```
typedef struct __CVI_VOICEPLAY_CFG_S  
{  
    uint32_t u32MaxVoiceCnt;  
    CVI_VOICEPLAY_VOICETABLE_S* pstVoiceTab;  
    CVI_VOICEPLAY_AOUT_OPT_S stAoutOpt;  
} CVI_VOICEPLAY_CFG_S;
```

【成员】

成员名称	描述
u32MaxVoiceCnt	最大声音数
pstVoiceTab	播放声音列表
stAoutOpt	声音播放选择

【注意事项】

无。

【相关数据类型及接口】

无。

3.16 ADASmng

该通用组件用于 ADAS 车辆检测。

3.16.1 API 参考

该功能模块为用户提供以下 API:

- `CVI_ADASMNG_VoiceCallback` : 声音提醒回调函数
- `CVI_ADASMNG_LabelCallback` : 车辆及车道线回调函数
- `CVI_ADASMNG_RegisterEvent` : ADAS 组件主题注册

3.16.1.1 CVI_ADASMNG_VoiceCallback

【描述】

声音提醒回调函数

【语法】

```
int32_t CVI_ADASMNG_VoiceCallback(int32_t index);
```

【参数】

参数名称	描述	输入/输出
index	命令值	输入

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: `cvi_adasmng.h`
- 库文件: `libcvi_adasmng.a/libcvi_adasmng.so`

【注意】

无

【举例】

无

【相关主题】

无

3.16.1.2 CVI_ADASMNG_LabelCallback**【描述】**

车辆及车道线回调函数

【语法】

```
iint32_t CVI_ADASMNG_LabelCallback(int32_t camid, int32_t index, uint32_t count, char*_  
↪coordinates);
```

【参数】

参数名称	描述	输入/输出
camid	sensor id	输入
index	命令值	输入
count	数量	输入
coordinates	坐标数组	输入

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_adasmng.h
- 库文件: libcvi_adasmng.a/libcvi_adasmng.so

【注意】

无

【举例】

无

【相关主题】

无

3.16.1.3 CVI_ADASMNG_RegisterEvent

【描述】

ADAS 主题注册

【语法】

```
int32_t CVI_ADASMNG_RegisterEvent(void);
```

【参数】

无

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_adasmng.h
- 库文件: libcvi_adasmng.a/libcvi_adasmng.so

【注意】

无

【举例】

无

【相关主题】

无

3.16.2 数据类型

视频输入相关数据类型定义如下:

- CVI_EVENT_ADASMNG_E : ADAS 组件主题。
- CVI_ADASMNG_CMD_E : ADAS 命令枚举量

3.16.2.1 CVI_EVENT_ADASMNG_E

【说明】

ADAS 组件主题。

【定义】

```
typedef enum cviEVENT_ADASMNG_E {
    CVI_EVENT_ADASMNG_CAR_MOVING,
    CVI_EVENT_ADASMNG_CAR_CLOSING,
    CVI_EVENT_ADASMNG_CAR_COLLISION,
    CVI_EVENT_ADASMNG_CAR_LANE,
    CVI_EVENT_ADASMNG_LABEL_CAR,
    CVI_EVENT_ADASMNG_LABEL_LANE,
    CVI_EVENT_ADASMNG_BUTT
} CVI_EVENT_ADASMNG_E;
```

【成员】

成员名称	描述
CVI_EVENT_ADASMNG_CAR_MOVING	前车启动主题
CVI_EVENT_ADASMNG_CAR_CLOSING	保持车距主题
CVI_EVENT_ADASMNG_CAR_COLLISION	碰撞警告主题
CVI_EVENT_ADASMNG_CAR_LANE	车道偏离主题
CVI_EVENT_ADASMNG_LABEL_CAR	车辆标注主题
CVI_EVENT_ADASMNG_LABEL_LANE	车道线标注主题

【注意事项】

无。

【相关数据类型及接口】

无。

3.16.2.2 CVI_ADASMNG_CMD_E

【说明】

ADAS 命令枚举量

【定义】

```
typedef enum _CVI_ADASMNG_CMD_E
{
    CVI_ADASMNG_NORMAL = 0,
    CVI_ADASMNG_CAR_MOVING,
    CVI_ADASMNG_CAR_CLOSING,
    CVI_ADASMNG_CAR_COLLISION,
    CVI_ADASMNG_CAR_LANE,
    CVI_ADASMNG_LABEL_CAR,
    CVI_ADASMNG_LABEL_LANE,
    CVI_ADASMNG_BUTT
} CVI_ADASMNG_CMD_E;
```

【成员】

成员名称	描述
CVI_ADASMNG_NORMAL	行驶正常
CVI_ADASMNG_CAR_MOVING	前车启动
CVI_ADASMNG_CAR_CLOSING	保持车距
CVI_ADASMNG_CAR_COLLISION	碰撞警告
CVI_ADASMNG_CAR_LANE	车辆标注
CVI_ADASMNG_LABEL_LANE	车道线标注

【注意事项】

无。

【相关数据类型及接口】

无。

3.17 Speechmng

该通用组件用于语音识别。可实现语音控制功能，包括切换前后路，拍照，录像及开关 wifi。

3.17.1 API 参考

该功能模块为用户提供以下 API：

- CVI_SPEECHMNG_Init：语音识别初始化
- CVI_SPEECHMNG_DeInit：语音识别反初始化
- CVI_SPEECHMNG_StartSpeech：开始语音识别
- CVI_SPEECHMNG_StopSpeech：停止语音识别

3.17.1.1 CVI_SPEECHMNG_Init

【描述】

语音识别初始化

【语法】

```
int32_t CVI_SPEECHMNG_Init(CVI_SPEECHMNG_PARAM_S* SpeechCfg);
```

【参数】

参数名称	描述	输入/输出
SpeechCfg	语音识别参数	输入

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: `cvi_speechmng.h`
- 库文件: `libcvi_speechmng.a/libcvi_speechmng.so`

【注意】

无

【举例】

无

【相关主题】

无

3.17.1.2 CVI_SPEECHMNG_DeInit

【描述】

语音识别反初始化

【语法】

```
int32_t CVI_SPEECHMNG_DeInit(void);
```

【参数】

无

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: `cvi_speechmng.h`
- 库文件: `libcvi_speechmng.a/libcvi_speechmng.so`

【注意】

无

【举例】

无

【相关主题】

无

3.17.1.3 CVI_SPEECHMNG_StartSpeech

【描述】

开始语音识别

【语法】

```
int32_t CVI_SPEECHMNG_DeInit(void);
```

【参数】

无

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_speechmng.h
- 库文件: libcvi_speechmng.a/libcvi_speechmng.so

【注意】

无

【举例】

无

【相关主题】

无

3.17.1.4 CVI_SPEECHMNG_StopSpeech

【描述】

暂停语音识别

【语法】

```
int32_t CVI_SPEECHMNG_StopSpeech(void);
```

【参数】

无

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: `cvi_speechmng.h`
- 库文件: `libcvi_speechmng.a/libcvi_speechmng.so`

【注意】

无

【举例】

无

【相关主题】

无

3.17.2 数据类型

视频输入相关数据类型定义如下:

- `CVI_EVENT_SPEECHMNG_E`: 语音识别主题。
- `CVI_SPEECHMNG_CMD_E`: 语音控制命令枚举量

3.17.2.1 `CVI_EVENT_SPEECHMNG_E`

【说明】

语音识别主题

【定义】

```
typedef enum cviEVENT_SPEECHMNG_E {  
    CVI_EVENT_SPEECHMNG_STARTREC,  
    CVI_EVENT_SPEECHMNG_STOPREC,  
    CVI_EVENT_SPEECHMNG_OPENFRONT,  
    CVI_EVENT_SPEECHMNG_OPENREAR,  
    CVI_EVENT_SPEECHMNG_CLOSESCREEN,  
    CVI_EVENT_SPEECHMNG_OPENSSCREEN,  
    CVI_EVENT_SPEECHMNG_EMRREC,  
    CVI_EVENT_SPEECHMNG_PIV,  
    CVI_EVENT_SPEECHMNG_CLOSEWIFI,  
    CVI_EVENT_SPEECHMNG_OPENWIFI,  
    CVI_EVENT_SPEECHMNG_BUTT  
} CVI_EVENT_SPEECHMNG_E;
```

【成员】

成员名称	描述
CVI_EVENT_SPEECHMNG_STARTREC	开始录像主题
CVI_EVENT_SPEECHMNG_STOPREC	停止录像主题
CVI_EVENT_SPEECHMNG_OPENFRONT	打开前路主题
CVI_EVENT_SPEECHMNG_OPENREAR	打开后路主题
CVI_EVENT_SPEECHMNG_CLOSESCREEN	关闭屏幕主题
CVI_EVENT_SPEECHMNG_OPENSSCREEN	关闭屏幕主题
CVI_EVENT_SPEECHMNG_EMRREC	开始紧急录像主题
CVI_EVENT_SPEECHMNG_PIV	开始拍照主题
CVI_EVENT_SPEECHMNG_OPENWIFI	打开 WIFI 主题
CVI_EVENT_SPEECHMNG_CLOSEWIFI	关闭 WIFI 主题

【注意事项】

无。

【相关数据类型及接口】

无。

3.17.2.2 CVI_SPEECHMNG_CMD_E**【说明】**

语音控制命令枚举量

【定义】

```
typedef enum _CVI_SPEECHMNG_CMD_E
{
    CVI_SPEECHMNG_NONE = 0,
    CVI_SPEECHMNG_SHOW_FRONT,
    CVI_SPEECHMNG_SHOW_REAR,
    CVI_SPEECHMNG_TURN_OFF_SCREEN,
    CVI_SPEECHMNG_OPEN_SCREEN,
    CVI_SPEECHMNG_LOCK_VEDIO,
    CVI_SPEECHMNG_TAKE_PICTURE,
    CVI_SPEECHMNG_TURN_OFF_RECORDING,
    CVI_SPEECHMNG_OPEN_RECORDING,
    CVI_SPEECHMNG_OPEN_WIFI,
    CVI_SPEECHMNG_TURN_OFF_WIFI,
    CVI_SPEECHMNG_BUIT
} CVI_SPEECHMNG_CMD_E;
```

【成员】

成员名称	描述
CVI_SPEECHMNG_TURN_OFF_RECORDING	开始录像
CVI_SPEECHMNG_OPEN_RECORDING	停止录像
CVI_SPEECHMNG_SHOW_FRONT	打开前路
CVI_SPEECHMNG_SHOW_REAR	打开后路
CVI_SPEECHMNG_TURN_OFF_SCREEN	关闭屏幕
CVI_SPEECHMNG_OPEN_SCREEN	关闭屏幕
CVI_SPEECHMNG_LOCK_VEDIO	开始紧急录像
CVI_SPEECHMNG_TAKE_PICTURE	开始拍照
CVI_SPEECHMNG_OPEN_WIFI	打开 WIFI
CVI_SPEECHMNG_TURN_OFF_WIFI	关闭 WIFI

【注意事项】

无。

【相关数据类型及接口】

无。

4 Dashcam

该章节介绍行车产品的产品层模块，可用于不同形态的行车产品，如行车记录仪，流媒体后视镜。

4.1 参数管理

参数管理模块(param)如图 2-1 所示。该模块与产品形态强相关,通过组合通用组件和 Framework 层组件定义出适合本产品形态的数据结构，主要包含以下文件：

- 文件管理
- 设备管理
- 工作模式
- 媒体配置
- UI 菜单

模块支持以下功能：

- 获取指定参数
- 保存指定参数
- 参数恢复默认

4.1.1 目录结构

参数管理目录包含如下组成部分：

```
/param
├── /inicfgs                //参数配置文件目录，以ini文件形式存放
│   ├── /cv180x            //车载板子版本号
│   └── /cv181x            //车载板子版本号
│       ├── /carrecorder    //行车记录仪
│       ├── /nonescreen     //无屏产品
│       └── /reaview         //流媒体后视镜
└── /ini2bin                //工具源码目录
    ├── cvi_param_ini_devmng.c //设备参数管理
    ├── cvi_param_ini_filemng.c //文件参数管理
    └── cvi_param_ini_workmode.c //工作模式管理
```

(下页继续)

(续上页)

```
|   ├── cvi_param_ini_media.c    //媒体配置管理
|   ├── cvi_param_ini_menu.c    //UI菜单管理
|   ├── cvi_param_ini2bin.c     //ini文件打包成bin
|   ├── /bin2flash              //工具源码目录
|   ├── /core                   //参数管理源码目录
|   └── /Makefile
```

注意:

- 不同形态产品的 ini 参数配置文件不同，例如流媒体后视镜，详情参考[流媒体后视镜 Param](#)。
- ini 参数以模块化方式，分为设备，文件，工作模式，媒体配置和 UI 菜单。统一管理，支持项目定制，同时与对外数据结构分离，低耦合，有利于后续项目开发。
- 参数数据聚合存储，尽可能减少重复参数，通过数据组装对外提供完整配置，从而小型化内存。

4.1.2 双系统参数流程设计

对于双系统 Linux+ Alios 架构，Linux 有读写 Flash 的能力，而 Alios 没有读写 Flash 的能力。在快速启动场景中，Alios 启动时需要根据系统参数启动媒体业务，因此系统参数采用裸读写 Flash 方式，在 uboot 启动时加载系统参数到指定 DDR，从而满足 Alios、Linux 在初始化时直接使用。

4.1.2.1 Flash 分区

图 4-1 为 Flash 典型分区，为了满足系统参数恢复默认、保存参数时异常恢复，Flash 上保存 2 处参数：

- 主参数分区 (APPCFG)：设备启动时使用的参数。
- 备份参数分区 (APPCFGDEF)：若主参数分区不完整，设备启动时使用的参数。



图 4.1: Flash 典型分区

Flash 典型分区说明：

- fip：引导程序分区，支持从参数分区读取参数到 DDR，支持 SD 升级等业务。
- 2nd：Alios 系统分区。
- BOOT：Linux 内核分区。
- APPCFG：主参数分区。

- APPCFGDEF: 备份参数分区。
- ENV:
- ENV_BAK
- MISC: 开机 logo 分区。
- ROOTFS: Linux 根文件系统分区。
- SYSTEM: Linux 应用文件系统分区。

为了制作参数相关镜像，提供如下工具：

- ini2bin 工具：将 ini 文件形式的参数转为二进制形式的参数 app_cfg.bin/app_cfg_def.bin，提供服务器端版本 ini2bin_pc 和板端版本 ini2bin_board。
- bin2flash 工具：在板端将 app_cfg.bin/app_cfg_def.bin 更新到 APPCFG/APPCFGDEF 分区，用于板端参数修改调试。

参数镜像的制作流程，如图 4-2 所示。

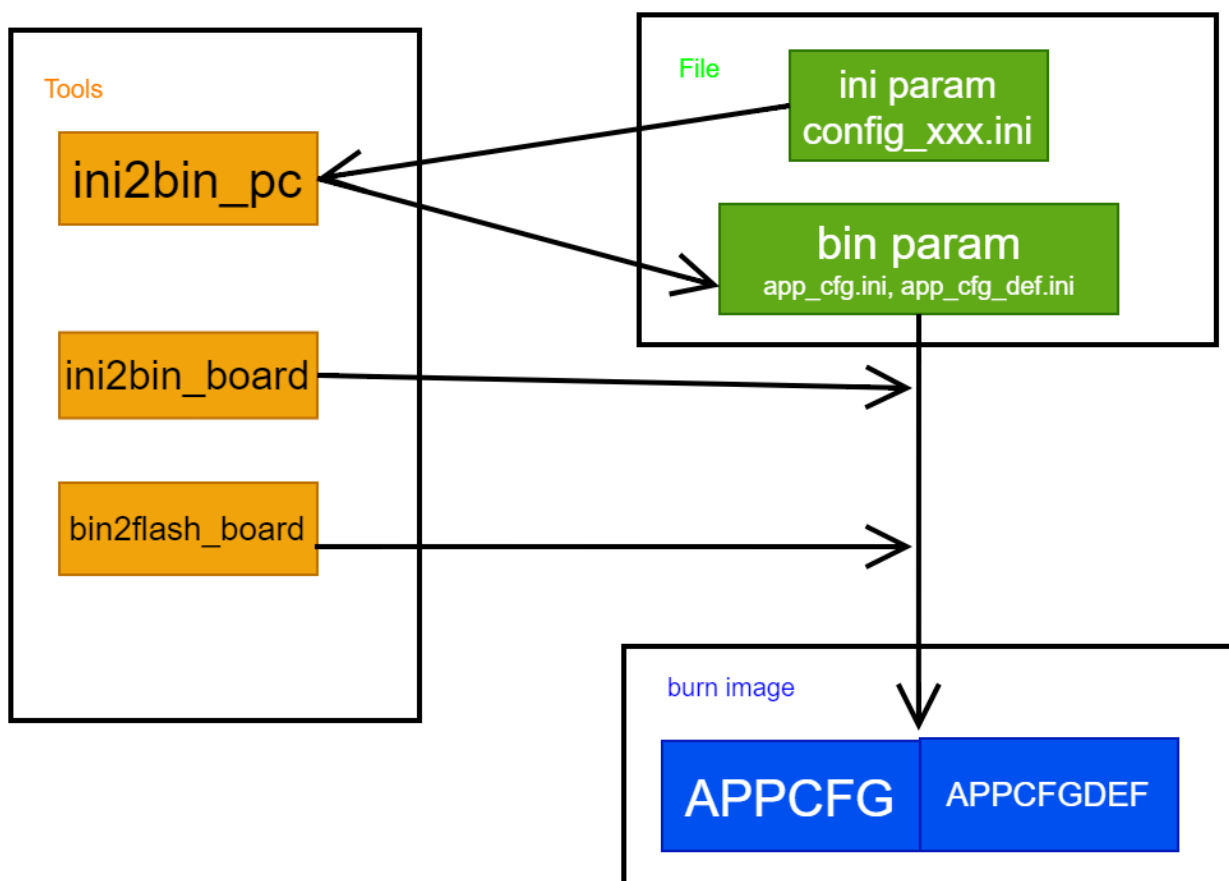


图 4.2: 参数制作流程

参数配置在板端路径默认是/mnt/system/bin/param，修改参数调测步骤如下：

- 进入参数目录：cd /mnt/system/bin/param
- 修改对应 ini 文件中的参数配置值

- 在板端执行工具 ini2bin_board: ./ini2bin_board
- 在板端执行工具 bin2flash: ./bin2flash
- 重启设备后参数生效

4.1.2.2 参数加载流程

设备启动时参数加载流程如图 4-3 所示：

- fip 从参数分区加载参数到约定 DDR。
- Alios APP 初始化参数模块、启动业务。
- Linux APP 初始化参数模块、启动业务。

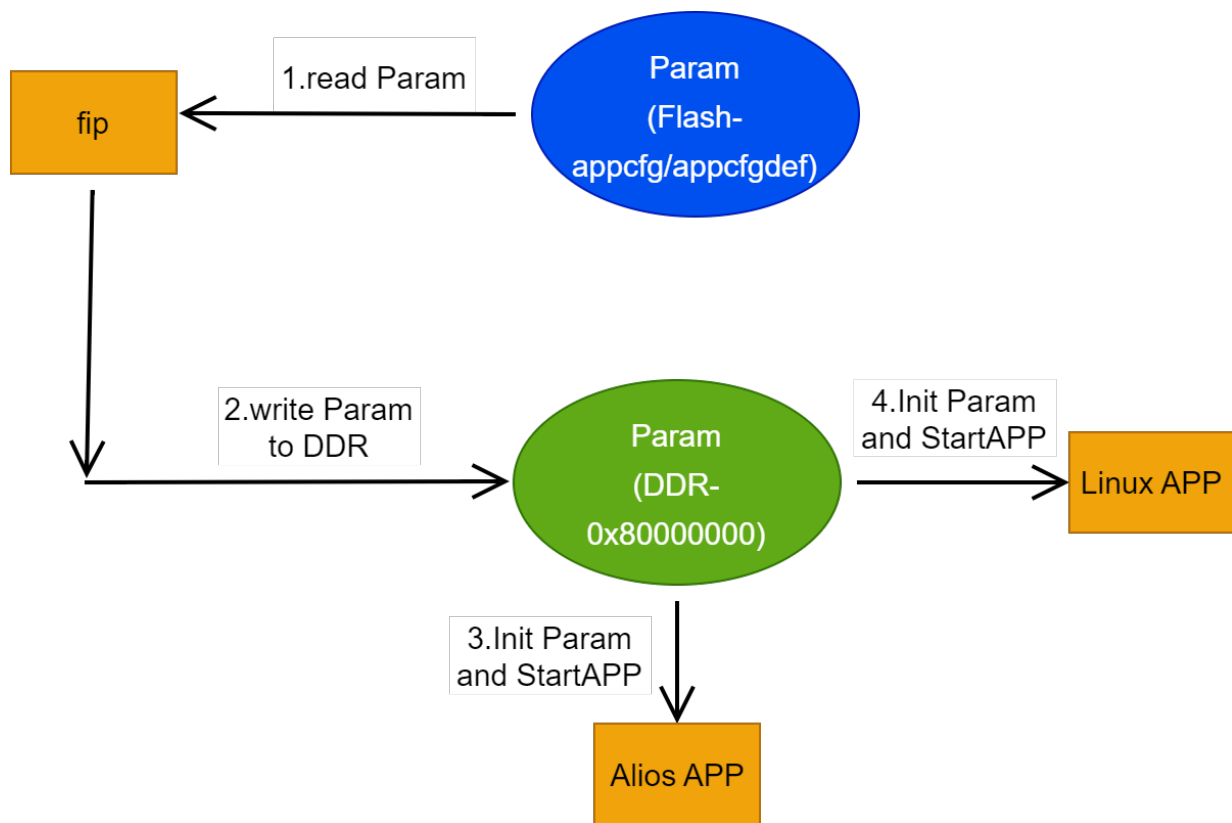


图 4.3: 参数加载流程

4.1.2.3 参数读写方案

双系统参数读写方案采用了 Flash Raw 方式，而 Flash 块的擦写次数有限，所以采用方案是 Alios APP 仅使用参数，Linux APP 负责擦写、同步、保存系统参数。当用户界面修改并保存配置时，比如开关 OSD，Linux APP 首先将配置更新到 DDR，然后在同步到 Flash 参数分区 appcfg/appcfgdef，具体流程如图 4-4 所示：

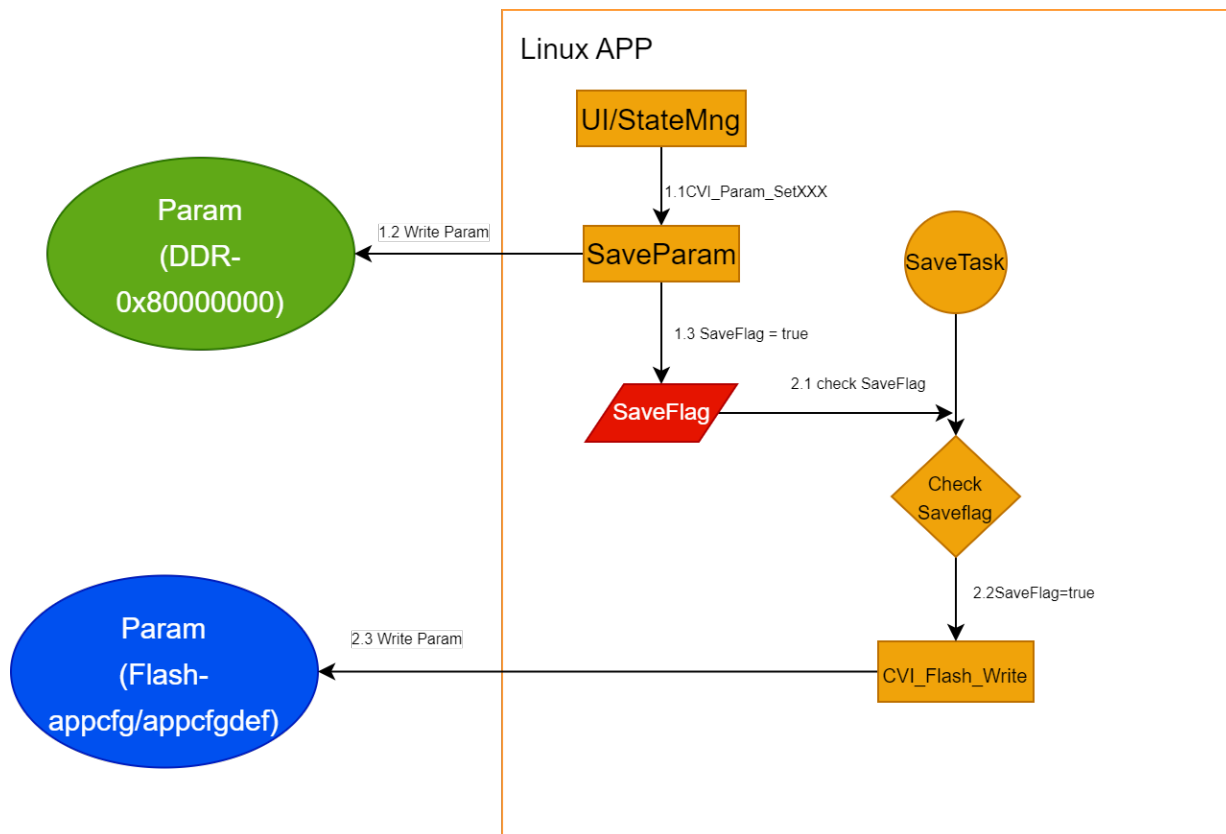


图 4.4: 参数读写方案

4.2 资源管理

资源管理模块的目的是将程序运行所依赖的资源数据（如声音源、osd 图片）与程序分离，方便开发人员在不修改程序的情况下进行资源替换。

4.2.1 目录结构

资源管理目录包含如下组成部分：

```

/resource
├── /bmp                //位图
│   └── cvitek.bmp
├── /voice              //音频文件
│   └── click.wav      //屏幕点击
  
```

(下页继续)

(续上页)

```
| |—— keytone.wav      //按键音distance
| |—— photo.wav        //拍照
| |—— rec.wav          //录像提示
| |—— collision.wav     //adas:碰撞警告
| |—— distance.wav     //adas:保持车距
| |—— lane.wav         //adas:车道偏移
| |—— moving.wav       //adas:前车启动
|—— Makefile
```

4.2.2 定制化资源

这里举例如何定制化。

4.2.2.1 定制化声音

1. 将需要替换的声音 WAV 文件替换到原始资源目录 voice 下。
2. 重新编译，即可将声音替换。

4.2.2.2 定制化 LogoOSD

1. 在 PC 上通过画图工具打开一张需要使用的 Logo 图片，另存为位深度为 24 位的 BMP 图片 osdlogo.bmp。
2. 将 osdlogo.bmp 文件替换到原始资源目录 bmp 下，如果需要调整 LogoOSD 大小和位置，可通过 inicfgs 参数配置文件中修改。
3. 在 Makefile 文件中将 osdlogo.bmp 替换原来的 logo 名字。
4. 重新编译即可将默认的 Logo 替换成 osdlogo.bmp 图片。

4.3 媒体管理

Media 模块上下文关系，如图 2-1 示，Media 模块通过对 framwork 中间件和 APPCOM 组件接口进行组合加工提供媒体接口，方便产品其它模块进行媒体流程设计。

Media 模块提供以下 3 类功能接口：

- 媒体通用配置接口：主要包括媒体初始化、媒体去初始化、媒体重置等接口，接口说明详见[媒体通用 API](#)。
- 媒体属性配置接口：主要包括 OSD 开关接口
- 媒体调测接口：主要用于 ISP

4.3.1 目录结构

媒体管理目录包含如下组成部分：

```
/media
├── /include
│   ├── cvi_media_dump.h
│   ├── cvi_media_init.h
│   └── cvi_media_osd.h
├── /src
│   ├── cvi_media_dump.c //媒体调测
│   ├── cvi_media_init.c //媒体通用配置
│   └── cvi_media_osd.c //媒体属性配置
└── Makefile
```

4.3.2 开发参考

4.3.2.1 媒体通用 API

该模块为用户提供以下 API：

- CVI_MEDIA_StartAudioInTask：开启录音服务
- CVI_MEDIA_StopAudioInTask：停止录音服务
- CVI_MEDIA_VideoInit：视频初始化，包括传感器初始化和相应的 vpss 初始化
- CVI_MEDIA_VideoDeInit：视频反初始化
- CVI_MEDIA_VcapInit：停止录音服务
- CVI_MEDIA_VcapDeInit：视频反初始化
- CVI_MEDIA_VbInit：视频池初始化
- CVI_MEDIA_VbInitPlayBack：用于回放的视频池初始化
- CVI_MEDIA_VbDeInit：视频池反初始化
- CVI_MEDIA_DispInit：视频输出初始化
- CVI_MEDIA_DispDeInit：视频输出反初始化
- CVI_MEDIA_LiveViewSerInit：预览初始化
- CVI_MEDIA_LiveViewSerDeInit：预览反初始化
- CVI_MEDIA_AiInit：音频输入初始化
- CVI_MEDIA_VencInit：视频编码初始化
- CVI_MEDIA_VencDeInit：视频编码反初始化
- CVI_MEDIA_AiDeInit：音频输入反初始化
- CVI_MEDIA_AencInit：音频编码初始化
- CVI_MEDIA_AencDeInit：音频编码反初始化

- CVI_MEDIA_RecordSerInit : 录像初始化
- CVI_MEDIA_RecordSerDeInit : 录像反初始化
- CVI_MEDIA_RtspSerInit : rtsp 初始化
- CVI_MEDIA_RtspSerDeInit : rtsp 反初始化
- CVI_MEDIA_AoInit : 音频输出初始化
- CVI_MEDIA_AoDeInit : 音频输出反初始化
- CVI_MEDIA_PlayBackSerInit : 回放初始化
- CVI_MEDIA_PlayBackSerDeInit : 回放反初始化
- CVI_MEDIA_Res2RecordMediaMode : 录像模式下 ahd 传感器分辨率设置
- CVI_MEDIA_Res2PhotoMediaMode : 拍照模式下 ahd 传感器分辨率设置
- CVI_MEDIA_SetAntiFlicker : 初始化 AE anti flicker 功能
- CVI_MEDIA_Is_CameraEnabled : 摄像头是否已经使能
- CVI_MEDIA_GetCtx : 获取媒体参数
- CVI_MEDIA_VIDEOMD_Init : 视频运动检测初始化
- CVI_MEDIA_VIDEOMD_DeInit : 视频运动检测反初始化
- CVI_MEDIA_APP_RTSP_Init : APP 连入时 rtsp 初始化
- CVI_MEDIA_APP_RTSP_DeInit : APP 断开时 rtsp 反初始化
- CVI_MEDIA_SwitchRTSPChanel : 切换 rtsp 播放窗口
- CVI_MEDIA_PhotoVprocInit : 拍照模式 vpss 初始化
- CVI_MEDIA_PhotoVprocDeInit : 拍照模式 vpss 反初始化
- CVI_MEDIA_PhotoSerInit : 拍照初始化
- CVI_MEDIA_PhotoSerDeInit : 拍照反初始化
- CVI_MEDIA_SensorDet : 传感器检测初始化
- CVI_MEDIA_ADASInit : ADAS 媒体初始化
- CVI_MEDIA_ADASDeInit : ADAS 媒体反初始化
- CVI_MEDIA_QRCodeInit : 二维码媒体初始化
- CVI_MEDIA_QRCodeDeInit : 二维码媒体反初始化

4.3.2.1.1 CVI_MEDIA_StartAudioInTask

【描述】

开启录音服务

【语法】

```
int32_t CVI_MEDIA_StartAudioInTask(void);
```

【参数】

无

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_media_init.h
- 库文件: libcvi_media.a/libcvi_media.so

【注意】

无

【举例】

无

【相关主题】

无

4.3.2.1.2 CVI_MEDIA_StopAudioInTask

【描述】

停止录音服务

【语法】

```
int32_t CVI_MEDIA_StopAudioInTask(void);
```

【参数】

无

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvt_media_init.h
- 库文件: libcvt_media.a/libcvt_media.so

【注意】

无

【举例】

无

【相关主题】

无

4.3.2.1.3 CVI_MEDIA_VideoInit**【描述】**

视频初始化, 包括传感器初始化和相应的 vps 初始化

【语法】

```
int32_t CVI_MEDIA_VideoInit(void);
```

【参数】

无

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvt_media_init.h
- 库文件: libcvt_media.a/libcvt_media.so

【注意】

无

【举例】

无

【相关主题】

无

4.3.2.1.4 CVI_MEDIA_VideoDeInit

【描述】

视频反初始化

【语法】

```
int32_t CVI_MEDIA_VideoDeInit(void);
```

【参数】

无

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_media_init.h
- 库文件: libcvi_media.a/libcvi_media.so

【注意】

无

【举例】

无

【相关主题】

无

4.3.2.1.5 CVI_MEDIA_VcapInit

【描述】

视频初始化, 包括传感器初始化和相应的 vpss 初始化

【语法】

```
int32_t CVI_MEDIA_VcapInit(void);
```

【参数】

无

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvt_media_init.h
- 库文件: libcvt_media.a/libcvt_media.so

【注意】

无

【举例】

无

【相关主题】

无

4.3.2.1.6 CVI_MEDIA_VcapDeInit**【描述】**

视频反初始化

【语法】

```
int32_t CVI_MEDIA_VcapDeInit(void);
```

【参数】

无

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvt_media_init.h
- 库文件: libcvt_media.a/libcvt_media.so

【注意】

无

【举例】

无

【相关主题】

无

4.3.2.1.7 CVI_MEDIA_VbInit

【描述】

视频池初始化

【语法】

```
int32_t CVI_MEDIA_VbInit(void);
```

【参数】

无

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_media_init.h
- 库文件: libcvi_media.a/libcvi_media.so

【注意】

无

【举例】

无

【相关主题】

无

4.3.2.1.8 CVI_MEDIA_VbInitPlayBack

【描述】

用于回放的视频池初始化

【语法】

```
int32_t CVI_MEDIA_VbInitPlayBack(void);
```

【参数】

无

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvt_media_init.h
- 库文件: libcvt_media.a/libcvt_media.so

【注意】

无

【举例】

无

【相关主题】

无

4.3.2.1.9 CVI_MEDIA_VbDeInit**【描述】**

视频池反初始化

【语法】

```
int32_t CVI_MEDIA_VbDeInit(void);
```

【参数】

无

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvt_media_init.h
- 库文件: libcvt_media.a/libcvt_media.so

【注意】

无

【举例】

无

【相关主题】

无

4.3.2.1.10 CVI_MEDIA_DispInit

【描述】

视频输出初始化

【语法】

```
int32_t CVI_MEDIA_DispInit(bool windowMode);
```

【参数】

参数名称	描述	输入/输出
windowMode	窗口使能	输入

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_media_init.h
- 库文件: libcvi_media.a/libcvi_media.so

【注意】

无

【举例】

无

【相关主题】

无

4.3.2.1.11 CVI_MEDIA_DispDeInit

【描述】

视频输出反初始化

【语法】

```
int32_t CVI_MEDIA_DispDeInit(void);
```

【参数】

无

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_media_init.h
- 库文件: libcvi_media.a/libcvi_media.so

【注意】

无

【举例】

无

【相关主题】

无

4.3.2.1.12 CVI_MEDIA_LiveViewSerInit

【描述】

预览初始化

【语法】

```
int32_t CVI_MEDIA_LiveViewSerInit(void);
```

【参数】

无

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_media_init.h
- 库文件: libcvi_media.a/libcvi_media.so

【注意】

无

【举例】

无

【相关主题】

无

4.3.2.1.13 CVI_MEDIA_LiveViewSerDeInit

【描述】

预览反初始化

【语法】

```
int32_t CVI_MEDIA_LiveViewSerDeInit(void);
```

【参数】

无

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_media_init.h
- 库文件: libcvi_media.a/libcvi_media.so

【注意】

无

【举例】

无

【相关主题】

无

4.3.2.1.14 CVI_MEDIA_AiInit

【描述】

音频输入初始化

【语法】

```
int32_t CVI_MEDIA_AiInit(void);
```

【参数】

无

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_media_init.h
- 库文件: libcvi_media.a/libcvi_media.so

【注意】

无

【举例】

无

【相关主题】

无

4.3.2.1.15 CVI_MEDIA_VencInit

【描述】

视频编码初始化

【语法】

```
int32_t CVI_MEDIA_VencInit(void);
```

【参数】

无

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_media_init.h
- 库文件: libcvi_media.a/libcvi_media.so

【注意】

无

【举例】

无

【相关主题】

无

4.3.2.1.16 CVI_MEDIA_VencDeInit

【描述】

视频编码反初始化

【语法】

```
int32_t CVI_MEDIA_VencDeInit(void);
```

【参数】

无

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_media_init.h
- 库文件: libcvi_media.a/libcvi_media.so

【注意】

无

【举例】

无

【相关主题】

无

4.3.2.1.17 CVI_MEDIA_AiDeInit

【描述】

音频输入反初始化

【语法】

```
int32_t CVI_MEDIA_AiDeInit(void);
```

【参数】

无

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_media_init.h
- 库文件: libcvi_media.a/libcvi_media.so

【注意】

无

【举例】

无

【相关主题】

无

4.3.2.1.18 CVI_MEDIA_AencInit

【描述】

音频编码初始化

【语法】

```
int32_t CVI_MEDIA_AencInit(void);
```

【参数】

无

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_media_init.h
- 库文件: libcvi_media.a/libcvi_media.so

【注意】

无

【举例】

无

【相关主题】

无

4.3.2.1.19 CVI_MEDIA_AencDeInit

【描述】

音频编码反初始化

【语法】

```
int32_t CVI_MEDIA_AencDeInit(void);
```

【参数】

无

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_media_init.h
- 库文件: libcvi_media.a/libcvi_media.so

【注意】

无

【举例】

无

【相关主题】

无

4.3.2.1.20 CVI_MEDIA_RecordSerInit

【描述】

录像初始化

【语法】

```
int32_t CVI_MEDIA_RecordSerInit(void);
```

【参数】

无

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_media_init.h
- 库文件: libcvi_media.a/libcvi_media.so

【注意】

无

【举例】

无

【相关主题】

无

4.3.2.1.21 CVI_MEDIA_RecordSerDeInit

【描述】

录像反初始化

【语法】

```
int32_t CVI_MEDIA_RecordSerDeInit(void);
```

【参数】

无

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_media_init.h
- 库文件: libcvi_media.a/libcvi_media.so

【注意】

无

【举例】

无

【相关主题】

无

4.3.2.1.22 CVI_MEDIA_RtspSerInit

【描述】

rtsp 初始化

【语法】

```
int32_t CVI_MEDIA_RtspSerInit(void);
```

【参数】

无

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_media_init.h
- 库文件: libcvi_media.a/libcvi_media.so

【注意】

无

【举例】

无

【相关主题】

无

4.3.2.1.23 CVI_MEDIA_RtspSerDeInit

【描述】

rtsp 反初始化

【语法】

```
int32_t CVI_MEDIA_RtspSerDeInit(void);
```

【参数】

无

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_media_init.h
- 库文件: libcvi_media.a/libcvi_media.so

【注意】

无

【举例】

无

【相关主题】

无

4.3.2.1.24 CVI_MEDIA_AoInit

【描述】

音频输出初始化

【语法】

```
int32_t CVI_MEDIA_AoInit(void);
```

【参数】

无

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_media_init.h
- 库文件: libcvi_media.a/libcvi_media.so

【注意】

无

【举例】

无

【相关主题】

无

4.3.2.1.25 CVI_MEDIA_AoDeInit

【描述】

音频输出反初始化

【语法】

```
int32_t CVI_MEDIA_AoDeInit(void);
```

【参数】

无

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_media_init.h
- 库文件: libcvi_media.a/libcvi_media.so

【注意】

无

【举例】

无

【相关主题】

无

4.3.2.1.26 CVI_MEDIA_PlayBackSerInit

【描述】

回放初始化

【语法】

```
int32_t CVI_MEDIA_PlayBackSerInit(void);
```

【参数】

无

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_media_init.h
- 库文件: libcvi_media.a/libcvi_media.so

【注意】

无

【举例】

无

【相关主题】

无

4.3.2.1.27 CVI_MEDIA_PlayBackSerDeInit

【描述】

回放反初始化

【语法】

```
int32_t CVI_MEDIA_PlayBackSerDeInit(void);
```

【参数】

无

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_media_init.h
- 库文件: libcvi_media.a/libcvi_media.so

【注意】

无

【举例】

无

【相关主题】

无

4.3.2.1.28 CVI_MEDIA_Res2RecordMediaMode

【描述】

录像模式下 ahd 传感器分辨率设置

【语法】

```
uint32_t CVI_MEDIA_Res2RecordMediaMode(int32_t res);
```

【参数】

参数名称	描述	输入/输出
res	模式	输入

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_media_init.h
- 库文件: libcvi_media.a/libcvi_media.so

【注意】

无

【举例】

无

【相关主题】

无

4.3.2.1.29 CVI_MEDIA_Res2PhotoMediaMode

【描述】

拍照模式下 ahd 传感器分辨率设置

【语法】

```
uint32_t CVI_MEDIA_Res2PhotoMediaMode(int32_t res);
```

【参数】

参数名称	描述	输入/输出
res	模式	输入

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_media_init.h
- 库文件: libcvi_media.a/libcvi_media.so

【注意】

无

【举例】

无

【相关主题】

无

4.3.2.1.30 CVI_MEDIA_SetAntiFlicker

【描述】

初始化 AE anti flicker 功能

【语法】

```
int32_t CVI_MEDIA_SetAntiFlicker(void);
```

【参数】

无

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_media_init.h
- 库文件: libcvi_media.a/libcvi_media.so

【注意】

无

【举例】

无

【相关主题】

无

4.3.2.1.31 CVI_MEDIA_Is_CameraEnabled**【描述】**

摄像头是否已经使能

【语法】

```
bool CVI_MEDIA_Is_CameraEnabled(int32_t cam_index);
```

【参数】

参数名称	描述	输入/输出
cam_index	cam id	输入

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_media_init.h
- 库文件: libcvi_media.a/libcvi_media.so

【注意】

无

【举例】

无

【相关主题】

无

4.3.2.1.32 CVI_MEDIA_GetCtx

【描述】

获取媒体参数

【语法】

```
CVI_MEDIA_PARAM_INIT_S* CVI_MEDIA_GetCtx(void);
```

【参数】

无

【返回值】

返回值	描述
非 0	媒体参数

【需求】

- 头文件: cvi_media_init.h
- 库文件: libcvi_media.a/libcvi_media.so

【注意】

无

【举例】

无

【相关主题】

无

4.3.2.1.33 CVI_MEDIA_VIDEOMD_Init

【描述】

视频运动检测初始化

【语法】

```
int32_t CVI_MEDIA_VIDEOMD_Init(void);
```

【参数】

无

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvt_media_init.h
- 库文件: libcvt_media.a/libcvt_media.so

【注意】

无

【举例】

无

【相关主题】

无

4.3.2.1.34 CVI_MEDIA_VIDEOMD_DeInit**【描述】**

视频运动检测反初始化

【语法】

```
int32_t CVI_MEDIA_VIDEOMD_DeInit(void);
```

【参数】

无

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvt_media_init.h
- 库文件: libcvt_media.a/libcvt_media.so

【注意】

无

【举例】

无

【相关主题】

无

4.3.2.1.35 CVI_MEDIA_APP_RTSP_Init

【描述】

APP 连入时 rtsp 初始化

【语法】

```
int32_t CVI_MEDIA_APP_RTSP_Init(uint32_t id, char *name);
```

【参数】

参数名称	描述	输入/输出
id	cam id	输入
name	rtsp 名	输入

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_media_init.h
- 库文件: libcvi_media.a/libcvi_media.so

【注意】

无

【举例】

无

【相关主题】

无

4.3.2.1.36 CVI_MEDIA_APP_RTSP_DeInit

【描述】

APP 断开时 rtsp 反初始化

【语法】

```
int32_t CVI_MEDIA_APP_RTSP_DeInit(void);
```

【参数】

无

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_media_init.h
- 库文件: libcvi_media.a/libcvi_media.so

【注意】

无

【举例】

无

【相关主题】

无

4.3.2.1.37 CVI_MEDIA_SwitchRTSPChanel

【描述】

切换 rtsp 播放窗口

【语法】

```
int32_t CVI_MEDIA_SwitchRTSPChanel(uint32_t value, uint32_t id, char *name);
```

【参数】

参数名称	描述	输入/输出
value	窗口值	输入
id	cam id	输入
name	rtsp 名	输入

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_media_init.h
- 库文件: libcvi_media.a/libcvi_media.so

【注意】

无

【举例】

无

【相关主题】

无

4.3.2.1.38 CVI_MEDIA_PhotoVprocInit**【描述】**

拍照模式下 vpss 初始化

【语法】

```
int32_t CVI_MEDIA_PhotoVprocInit(void);
```

【参数】

无

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_media_init.h
- 库文件: libcvi_media.a/libcvi_media.so

【注意】

无

【举例】

无

【相关主题】

无

4.3.2.1.39 CVI_MEDIA_PhotoVprocDeInit**【描述】**

拍照模式 vpss 反初始化

【语法】

```
int32_t CVI_MEDIA_PhotoVprocDeInit(void);
```

【参数】

无

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_media_init.h
- 库文件: libcvi_media.a/libcvi_media.so

【注意】

无

【举例】

无

【相关主题】

无

4.3.2.1.40 CVI_MEDIA_PhotoSerInit

【描述】

拍照初始化

【语法】

```
int32_t CVI_MEDIA_PhotoSerInit(void);
```

【参数】

无

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_media_init.h
- 库文件: libcvi_media.a/libcvi_media.so

【注意】

无

【举例】

无

【相关主题】

无

4.3.2.1.41 CVI_MEDIA_PhotoSerDeInit**【描述】**

拍照反初始化

【语法】

```
int32_t CVI_MEDIA_PhotoSerDeInit(void);
```

【参数】

无

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_media_init.h
- 库文件: libcvi_media.a/libcvi_media.so

【注意】

无

【举例】

无

【相关主题】

无

4.3.2.1.42 CVI_MEDIA_SensorDet**【描述】**

传感器检测初始化

【语法】

```
int32_t CVI_MEDIA_SensorDet(void);
```

【参数】

无

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_media_init.h
- 库文件: libcvi_media.a/libcvi_media.so

【注意】

无

【举例】

无

【相关主题】

无

4.3.2.1.43 CVI_MEDIA_ADASInit

【描述】

ADAS 媒体初始化

【语法】

```
int32_t CVI_MEDIA_ADASInit(void);
```

【参数】

无

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_media_init.h
- 库文件: libcvi_media.a/libcvi_media.so

【注意】

无

【举例】

无

【相关主题】

无

4.3.2.1.44 CVI_MEDIA_ADASDeInit**【描述】**

ADAS 媒体反初始化

【语法】

```
int32_t CVI_MEDIA_ADASDeInit(void);
```

【参数】

无

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_media_init.h
- 库文件: libcvi_media.a/libcvi_media.so

【注意】

无

【举例】

无

【相关主题】

无

4.3.2.1.45 CVI_MEDIA_QRCodeInit**【描述】**

二维码媒体初始化

【语法】

```
int32_t CVI_MEDIA_QRCodeInit(void);
```

【参数】

无

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_media_init.h
- 库文件: libcvi_media.a/libcvi_media.so

【注意】

无

【举例】

无

【相关主题】

无

4.3.2.1.46 CVI_MEDIA_QRCodeDeInit

【描述】

二维码媒体反初始化

【语法】

```
int32_t CVI_MEDIA_QRCodeDeInit(void);
```

【参数】

无

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_media_init.h
- 库文件: libcvi_media.a/libcvi_media.so

【注意】

无

【举例】

无

【相关主题】

无

4.3.2.2 数据结构

媒体通用相关数据结构定义如下:

- CVI_MEDIA_SYSHANDLE_S : 媒体系统句柄
- CVI_MEDIA_CAMERA_SERVICE_S : App 服务结构体
- CVI_MEDIA_PARAM_INIT_S : 媒体参数
- MAX_CAMERA_INSTANCES : 最大传感器数目
- MAX_DEV_INSTANCES : 最大设备数目
- MAX_RTSP_CNT : 最大连接 rtsp 数目
- MAX_VPROC_CNT : 最大 VPSS 数目
- MAX_VENC_CNT : 最大编码通道数目

4.3.2.2.1 MAX_CAMERA_INSTANCES**【说明】**

最大传感器数目

【定义】

该参数来自参数管理 的 config_devmng.ini 文件, 编译时赋值, 详情可参考流媒体后视镜 Param。

【成员】

无

【注意事项】

一个传感器可以切多个镜头。

【相关数据类型及接口】

无。

4.3.2.2.2 MAX_DEV_INSTANCES

【说明】

最大设备数目

【定义】

该参数来自参数管理的 config_devmng.ini 文件，编译时赋值，详情可参考流媒体后视镜 Param。

【成员】

无

【注意事项】

传感器和设备通常是同一个意思，但在“一托多”项目中，一个设备通过拓展接口可以连接多个传感器。

【相关数据类型及接口】

无。

4.3.2.2.3 MAX_RTSP_CNT

【说明】

最大连接 rtsp 数目，包含 PC 端和手机端。

【定义】

```
#define MAX_APP_RTSP_CNT          (1) //最大手机app rtsp连接数目，  
#define MAX_RTSP_CNT              (MAX_CAMERA_INSTANCES + MAX_APP_RTSP_  
→CNT)
```

【成员】

无

【注意事项】

无。

【相关数据类型及接口】

无。

4.3.2.2.4 MAX_VPROC_CNT

【说明】

最大 vps 数目

【定义】

```
#define MAX_VPROC_CNT (16)
```

【成员】

无

【注意事项】

无。

【相关数据类型及接口】

无。

4.3.2.2.5 MAX_VENC_CNT

【说明】

最大编码通道数目。

【定义】

```
#define MAX_VENC_CNT (MAX_CAMERA_INSTANCES * 4)
```

【成员】

无

【注意事项】

无。

【相关数据类型及接口】

无。

4.3.2.2.6 CVI_MEDIA_SYSHANDLE_S

【说明】

媒体系统句柄

【定义】

```
typedef struct cviMEDIA_SYSHANDLE_S {  
    CVI_MAPI_VCAP_SENSOR_HANDLE_T sns[MAX_DEV_INSTANCES];  
    CVI_MAPI_VPROC_HANDLE_T vproc[MAX_VPROC_CNT];  
    CVI_MAPI_VENC_HANDLE_T venchdl[MAX_CAMERA_INSTANCES][MAX_VENC_  
→CNT];  
};
```

(下页继续)

(续上页)

```

CVI_MAPI_ACAP_HANDLE_T    aihdl;
CVI_MAPI_AENC_HANDLE_T    aenchdl;
CVI_MAPI_DISP_HANDLE_T    dispHdl;
CVI_MAPI_AO_HANDLE_T      aohdl;
} CVI_MEDIA_SYSHANDLE_S;

```

【成员】

成员名称	描述
sns	传感器输入句柄数组
vproc	vpss 句柄数组
venchdl	编码句柄数组
aihdl	音频输入句柄
aenchdl	音频编码句柄
dispHdl	视频输出句柄
aohdl	音频输出句柄

【注意事项】

无。

【相关数据类型及接口】

无。

4.3.2.2.7 CVI_MEDIA_CAMERA_SERVICE_S

【说明】

App 服务结构体

【定义】

```

typedef struct cviCAMERA_SERVICE_S {
    CVI_RECORD_SERVICE_HANDLE_T    RecordHdl[MAX_CAMERA_INSTANCES];
    CVI_RECORD_SERVICE_PARAM_S     RecordParams[MAX_CAMERA_INSTANCES];

    CVI_PHOTO_SERVICE_HANDLE_T     PhotoHdl[MAX_CAMERA_INSTANCES];
    CVI_PHOTO_SERVICE_PARAM_S      PhotoParams[MAX_CAMERA_INSTANCES];
#ifdef SERVICES_RTSP_ON
    CVI_RTSP_SERVICE_HANDLE_T      RtspHdl[MAX_RTSP_CNT];
    CVI_RTSP_SERVICE_PARAM_S       RtspParams[MAX_RTSP_CNT];
#endif
#ifdef SERVICES_ADAS_ON
    CVI_ADAS_SERVICE_HANDLE_T      ADASHdl[MAX_CAMERA_INSTANCES];
    CVI_ADAS_SERVICE_PARAM_S       ADASParams[MAX_CAMERA_INSTANCES];
#endif
#ifdef SERVICES_PLAYER_ON
    CVI_PLAYER_SERVICE_HANDLE_T    PsHdl;
    CVI_PLAYER_SERVICE_PARAM_S     PsParam;
#endif
#ifdef SERVICES_LIVEVIEW_ON

```

(下页继续)

(续上页)

```

    CVI_LIVEVIEW_SERVICE_HANDLE_T  LvHdl;
    CVI_LIVEVIEW_SERVICE_PARAM_S   LvParams;
#endif
#ifdef SERVICES_QRCODE_ON
    CVI_QRCODE_SERVICE_HANDLE_T    QRCodeHdl[MAX_CAMERA_INSTANCES];
    CVI_QRCODE_SERVICE_PARAM_S     QRCodeParams[MAX_CAMERA_INSTANCES];
#endif
    CVI_MT_HANDLE_T                MtHdl;
    CVI_MT_SERVICE_PARAM_T          MtParam;
} CVI_MEDIA_CAMERA_SERVICE_S;

```

【成员】

成员名称	描述
LvHdl	预览服务句柄
LvParams	预览服务参数
RecordHdl	录像服务句柄
RecordParams	录像服务参数
PhotoHdl	拍照服务句柄
PhotoParams	拍照服务参数
RtspHdl	rtsp 服务句柄数组
RtspParams	rtsp 服务参数数组
PsHdl	回放服务句柄
PsParam	回放服务参数
MtHdl	模式测试句柄
MtParam	模式测试服务参数
ADASHdl	ADAS 服务句柄
ADASParams	ADAS 服务参数
QRCodeHdl	二维码服务句柄
QRCodeParams	二维码服务参数

【注意事项】

无。

【相关数据类型及接口】

无。

4.3.2.2.8 CVI_MEDIA_PARAM_INIT_S**【说明】**

媒体初始化参数

【定义】

```

typedef struct cviMEDIA_PARAM_INIT_S {
    CVI_MEDIA_SYSHANDLE_S      SysHandle;
    CVI_MEDIA_CAMERA_SERVICE_S SysServices;
} CVI_MEDIA_PARAM_INIT_S;

```

【成员】

成员名称	描述
SysHandle	媒体系统句柄
SysServices	App 服务结构体

【注意事项】

无。

【相关数据类型及接口】

无。

4.3.3 模块分析

这里列举媒体通用流程

4.3.3.1 典型媒体通路初始化

以录像模式初始化为例，如图 4-5 示

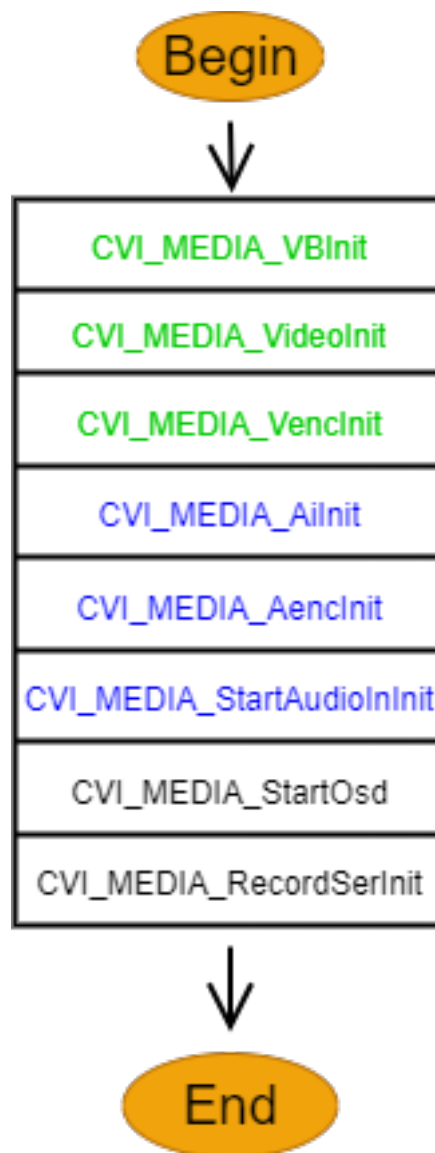


图 4.5: 录像模式媒体通路初始化流程

注意:

- 模块依赖条件：提前完成芯片管脚配置、时钟配置、媒体驱动加载等，详情可参考[INIT 模块](#) 所示。
- 支持 UI 时，视频输出初始化需要提前初始化，保证图像层可用。
- 支持 OSD 时，视频初始化前需要保证 OSD 已经初始化。
- 媒体通路反初始化按相反的顺序调用相对应的 API 即可。

4.3.3.2 典型媒体通路重置处理

以录像模式初始化为例，如图 4-6 示

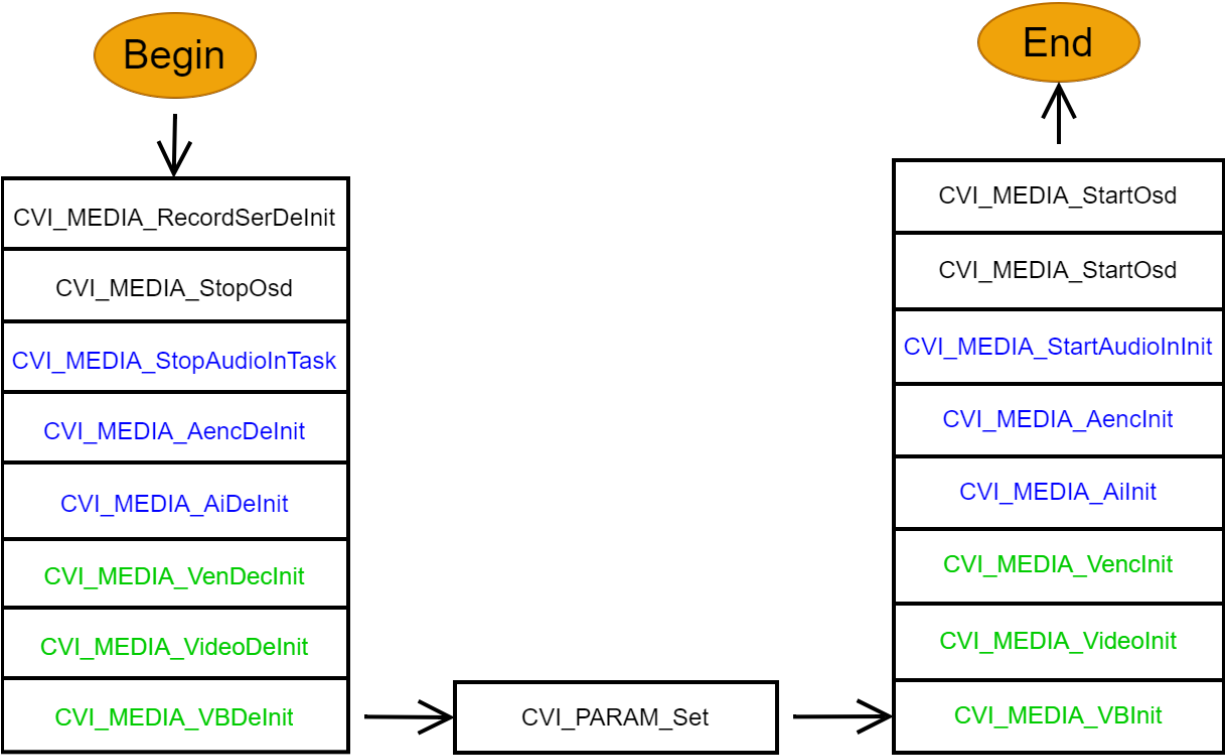


图 4.6: 录像模式媒体通路重置处理流程

注意:

- 该流程展示的是媒体通路进行全部重新初始化。有些媒体通路模块可以根据参数来进行单独或者部分重新初始化，例如只执行 CVI_MEDIA_RecordSerDeInit 和 CVI_MEDIA_RecordSerInit，这样切换时间较短。

4.4 状态管理

StateMng 模块是 APP 层中实现具体业务模式、状态管理以及相应事件处理的功能模块，目的为降低模块间耦合，提高代码的可复用性和可扩展性。不同的产品形态根据不同的业务需求定义出业务模式 (Mode)，每种业务模式对应于一种业务状态 (State)。

4.4.1 模块上下文

StateMng 模块上下文关系如图 4-7 所示：

- NetCtrl/UI 模块通过 SendMessage 函数向 StateMng 模块发送相应调用，StateMng 模块在具体业务状态中组织产品层各功能模块 (Param/Media/Mode/APP COM) 及 Framework 组件，完成相应业务逻辑。
- StateMng 模块通过 EVENTHUB_Register 函数向 EventHub 模块注册相应事件 (start/stop/setting 等)，StateMng 模块通过 EVENTHUB_Subscribe 函数向 EventHub 模块订阅系统事件并注册事件处理回调函数 (EventProc 函数)。
- 事件发生时 EventHub 模块调用 EVENTHUB_Publish 函数发布事件，从而调用 EventProc 函数进行相应事件处理。

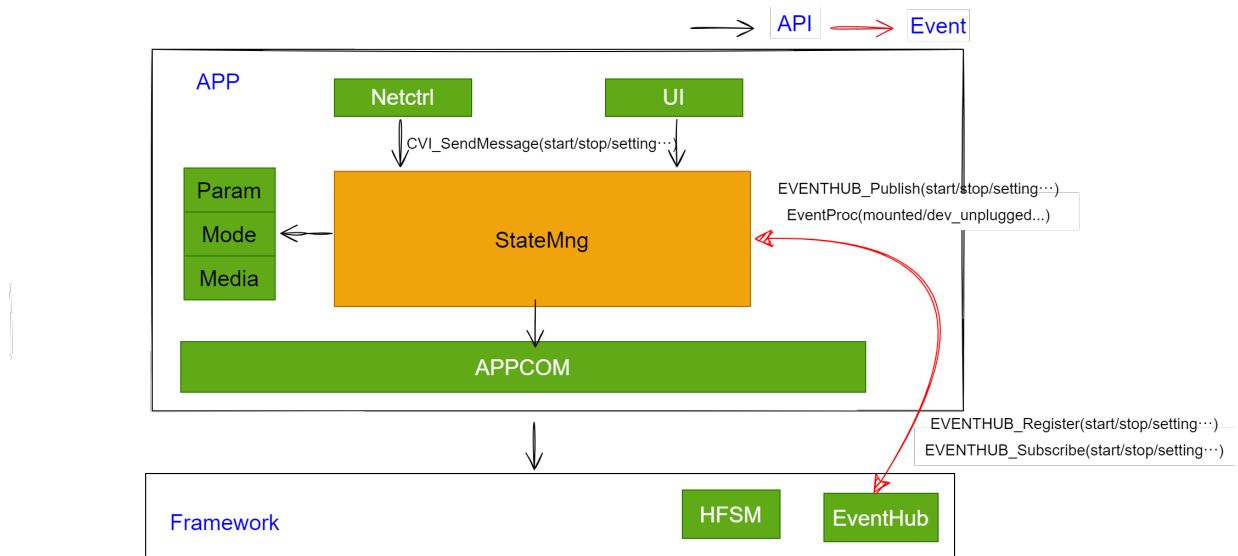


图 4.7: StateMng 模块上下文

4.4.2 模块依赖

4.4.2.1 StateMachine 模块

StateMachine 模块实现了 HFSM 框架，详情参考 Framework 通用软件开发指南。StateMng 模块基于此模块实现业务状态管理，如下表所示。

表 4.1: StateMng 模块应用 StateMachine 模块

编号	StateMng 模块在业务状态相关的工作
1	明确定义具体的状态，并添加到 StateMachine 模块
2	明确定义不同状态下需要处理的事件
3	编写事件处理函数，并注册到 StateMachine 模块
4	按 StateMachine 模块提供的状态转换机制实现相应的逻辑

4.4.2.2 EventHub 模块

EventHub 模块是一个基于发布/订阅模式的，统一管理系统中事件订阅和发布的模块，详细介绍请参考 Framework 通用软件开发指南。StateMng 模块基于此模块实现事件处理，如下表所示。

表 4.2: StateMng 模块应用 EventHub 模块

编号	StateMng 模块在事件处理相关的工作
1	明确需要 StateMng 模块处理的事件，包含需要发布和需要订阅的两部分
2	向 EventHub 模块注册需要发布的事件
3	向 EventHub 模块订阅需要处理的事件，订阅时注册相应的事件处理函数

4.4.3 模块分析

模块分析主要涉及子模块划分，二级层次状态机，典型业务状态和典型场景分析。

4.4.3.1 子模块划分

StateMng 模块子模块划分如图 4-8 所示：

- 交互：StateMng 模块 SendMessage 函数内部通过 HFSM_SendAsyncMessage 函数将消息发送到状态机消息队列中待处理；注册给 EventHub 模块的 EventProc 函数内部根据消息进行相应的处理。
- HFSM States：StateMng 模块创建的状态机中包含的各业务状态（对应于需求中的各业务模式）。状态机初始化后，发送到状态机消息队列中的消息被转发到当前状态，通过调用系统中相应模块实现业务功能。HFSM States 分为 Base 和子状态，称为二级层次状态机，其目的是为了减少一些重复的逻辑，子状态仅关注业务强相关的事件处理，通用事件处理由 Base 状态处理

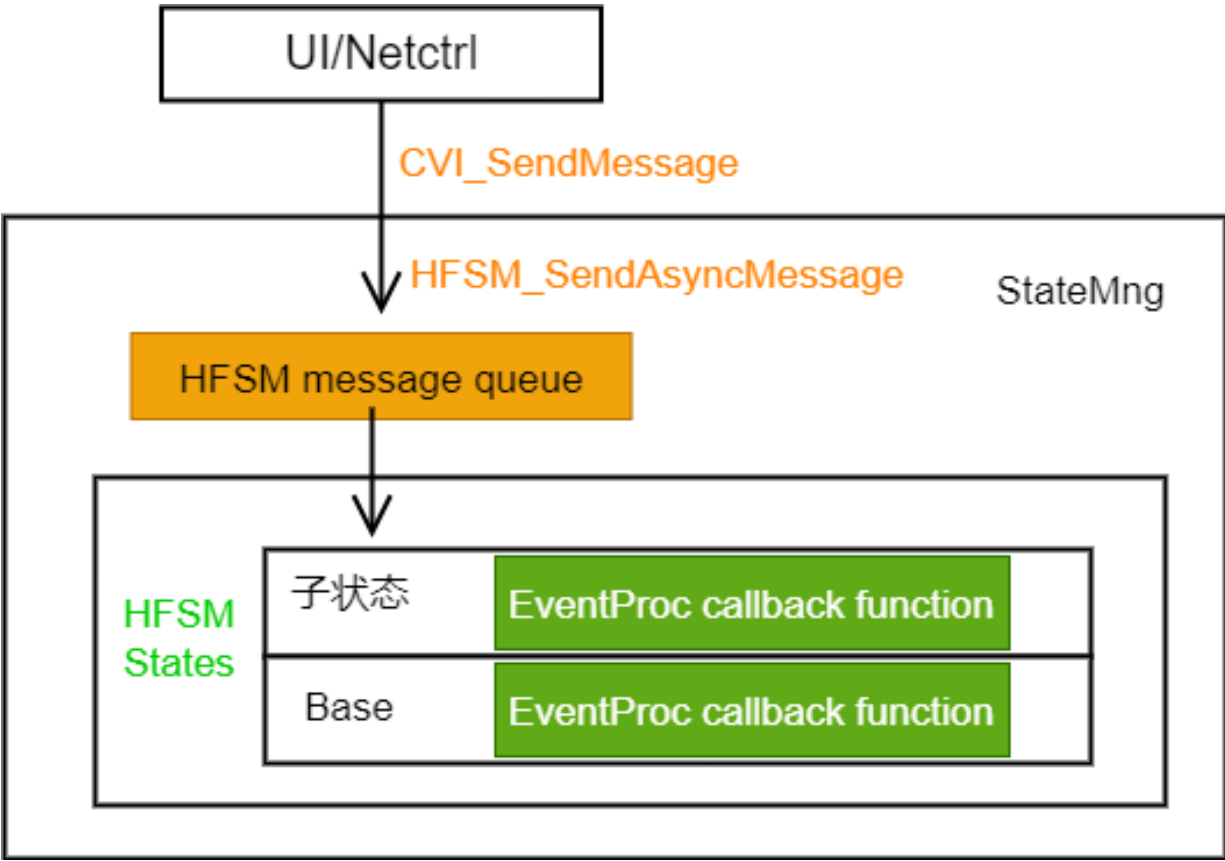


图 4.8: StateMng 模块子模块划分

4.4.3.2 典型业务状态

本小节以普通录像为例分析 HFSM 实例中典型业务状态的实现, 详情参考流媒体后视镜 statemng。

业务状态的基本实现主要包含实现表格中的五个函数, 具体函数的逻辑同业务状态要实现的功能紧密相关, 如下表所示。

表 4.3: 普通录像状态的实现

函数	函数解释
init()	普通录像状态的注册和录像模块的初始化
deinit()	录像模块的去初始化
open()	普通录像状态的初始化, 包含普通录像媒体通路的初始化, 录像任务属性的获取和创建, 状态上下文变量的更新, 系统参数的保存等
close()	普通录像状态的去初始化, 包含录像任务的检查和停止 (非必须), 录像任务的销毁, 状态上下文变量的重置
Msg-Proc()	普通录像状态下事件的处理, 普通录像状态事件处理包含业务相关事件、系统事件及系统运行异常事件, 详见下表

表 4.4: 普通录像状态事件处理

状态	事件	事件处理
base	Mouted	SD 卡插入, 更新 FileMng (文件管理) 模块状态
base	Device unplugged	SD 卡拔出, 按当前状态 (是否正在录像) 停止录像任务, 更新 FileMng (文件管理) 模块状态
base	Format	SD 卡格式化, 格式化处理逻辑的调用
base	Switch WorkMode	切换模式, 调用 StateMachine 模块模式切换机制
子状态	Start	启动录像任务
子状态	Stop	停止录像任务
子状态	Setting	按设置参数重新创建录像任务
子状态	RecordMng error	录像错误, 停止录像任务

4.4.3.3 典型场景分析

1.StateMng 模块初始化: 主要包含二级 HFSM 的创建, 事件的订阅, HFSM 初始化及激活, 以及开机动作的处理。以开机模式为普通录像为例, HFSM 激活的过程从构建的二级层次状态机的父节点开始, 依次调用 Base 状态的 open 函数, 录像状态的 open 函数, 最终系统当前状态为录像状态。

2. 业务状态间切换: UI/NetCtrl 发送的 Swith WorkMode 消息放入 HFSM 消息队列后, 首先被分发到当前状态进行处理, 该消息的处理由“行为继承”继承自 Base 状态, 当前状态先将消息转发给 Base 状态, Base 状态 MsgProc 函数对 Swith WorkMode 消息的处理依赖于 StateMachine 模块的机制, 将目标状态设置给 HFSM, 从而触发切换逻辑

3. 模块去初始化: 录像模块去初始化流程中主要包含各业务状态的去初始化, 媒体通路去初始化, HFSM 实例的关闭和销毁。HFSM 的关闭依次调用当前状态的 close 函数、Base 状态的 close 函数

4.4.4 模式管理

该模块介绍目前已有的模式及相关 API。

4.4.4.1 目录结构

模式管理目录包含如下组成部分:

```
/mode
├── /include
│   ├── cvi_mode.h           //用户API
│   └── cvi_modetest.h       //用于终端命令切换模式
├── /src
│   ├── cvi_bootmode.c       //开机模式
│   ├── cvi_modebase.c       //涉及Base状态
│   ├── cvi_modectrl.c       //模式控制
│   └── cvi_modeinner.c      //内部函数API
```

(下页继续)

(续上页)

```

|   |—— cvi_moviemode.c      //录像模式
|   |—— cvi_photomode.c     //拍照模式
|   |—— cvi_playbackmode.c  //回放模式
|   |—— cvi_storagemode.c   //存储模式
|   |—— cvi_testmode.c      //测试模式
|   |—— cvi_updatemode.c    //升级模式
|   |—— cvi_usbmenumode.c   //usb模式
|   |—— cvi_uvcmode.c       //uvc模式
|   |—— cvi_modecommon.c    //模式通用的函数
|—— Makefile

```

4.4.4.2 API 参考

该功能模块为用户提供以下 API:

- CVI_MODEMNG_Init : 模式管理初始化
- CVI_MODEMNG_Deinit : 模式管理反初始化
- CVI_MODEMNG_SendMessage : 发送消息
- CVI_MODEMNG_GetCurMode : 获取当前工作模式
- CVI_MODEMNG_GetCurWorkMode : 获取当前工作模式
- CVI_MODEMNG_SetParkingRec : 使能泊车录像
- CVI_MODEMNG_GetModeState : 获取当前工作模式状态
- CVI_MODEMNG_SetModeState : 设置当前工作模式状态
- CVI_MODEMNG_SetEmrState : 紧急录像使能
- CVI_MODEMNG_LiveViewUp : 预览视角上移
- CVI_MODEMNG_LiveViewDown : 预览视角上移
- CVI_MODEMNG_GetCardState : 获取 sd 卡状态
- CVI_MODEMNG_GetCamStatus : 获取传感器是否运行
- CVI_MODEMNG_GetCamIspInfoStatus : 获取传感器 isp 是否使能
- CVI_MODEMNG_SetCardState : 设置 sd 卡状态
- CVI_MODEMNG_GetInProgress : 获取是否进行中
- CVI_MODEMNG_GetEmrState : 是否紧急录像中
- CVI_MODEMNG_ContextInit : 模式管理配置初始化
- CVI_MODEMNG_SetDefaultParam : 设置默认参数
- CVI_MODEMNG_TEST_MAIN_Create : 测试模式创建
- CVI_MODEMNG_TEST_MAIN_Destroy : 测试模式销毁
- CVI_MODEMNG_SetMediaLoopTime : 设置录像时长
- CVI_MODEMNG_SetMediaAudio : 设置音频

- CVI_MODEMNG_SetMenuWifiStatus : 设置 WiFi 状态
- CVI_MODEMNG_SetMediaOsd : 设置 osd 状态
- CVI_MODEMNG_SetMediaLapseTime : 设置缩时间隔
- CVI_MODEMNG_SetMediaVencFormat : 设置编码格式
- CVI_MODEMNG_LiveViewSwitch : 切换预览镜头
- CVI_MODEMNG_RegisterEvent : 模式状态注册
- CVI_MODEMNG_Format : sd 卡格式化
- CVI_MODEMNG_OpenUvcMode : 打开 uvc 模式
- CVI_MODEMNG_CloseUvcMode : 关闭 uvc 模式
- CVI_MODEMNG_OpenPhotoMode : 打开照相模式
- CVI_MODEMNG_ClosePhotoMode : 关闭照相模式

4.4.4.2.1 CVI_MODEMNG_Init

【描述】

模式管理初始化

【语法】

```
int32_t CVI_MODEMNG_Init(CVI_MODEMNG_CONFIG_S *stModemngCfg);
```

【参数】

参数名称	描述	输入/输出
stModemngCfg	模式管理配置	输入

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_mode.h
- 库文件: libcvi_mode.a/libcvi_mode.so

【注意】

无

【举例】

无

【相关主题】

无

4.4.4.2.2 CVI_MODEMNG_Deinit

【描述】

模式管理反初始化

【语法】

```
int32_t CVI_MODEMNG_Deinit();
```

【参数】

无

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_mode.h
- 库文件: libcvi_mode.a/libcvi_mode.so

【注意】

无

【举例】

无

【相关主题】

无

4.4.4.2.3 CVI_MODEMNG_SendMessage

【描述】

发送消息

【语法】

```
int32_t CVI_MODEMNG_SendMessage(const CVI_MESSAGE_S *pstMsg);
```

【参数】

参数名称	描述	输入/输出
pstMsg	消息	输入

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_mode.h
- 库文件: libcvi_mode.a/libcvi_mode.so

【注意】

无

【举例】

无

【相关主题】

无

4.4.4.2.4 CVI_MODEMNG_GetCurMode

【描述】

获取当前工作模式

【语法】

```
int32_t CVI_MODEMNG_GetCurMode(uint32_t *ModeId);
```

【参数】

参数名称	描述	输入/输出
ModeId	工作模式	输出

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_mode.h
- 库文件: libcvi_mode.a/libcvi_mode.so

【注意】

无

【举例】

无

【相关主题】

无

4.4.4.2.5 CVI_MODEMNG_GetCurWorkMode

【描述】

获取当前工作模式

【语法】

```
int32_t CVI_MODEMNG_GetCurWorkMode(void);
```

【参数】

无

【返回值】

返回值	描述
非 0	当前工作模式

【需求】

- 头文件: cvi_mode.h
- 库文件: libcvi_mode.a/libcvi_mode.so

【注意】

无

【举例】

无

【相关主题】

无

4.4.4.2.6 CVI_MODEMNG_GetModeState

【描述】

获取当前工作模式状态

【语法】

```
int32_t CVI_MODEMNG_GetModeState(uint32_t *state);
```

【参数】

参数名称	描述	输入/输出
state	当前工作模式状态	输出

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_mode.h
- 库文件: libcvi_mode.a/libcvi_mode.so

【注意】

无

【举例】

无

【相关主题】

无

4.4.4.2.7 CVI_MODEMNG_SetModeState

【描述】

设置当前工作模式状态

【语法】

```
int32_t CVI_MODEMNG_SetModeState(uint32_t state);
```

【参数】

参数名称	描述	输入/输出
state	工作模式状态	输入

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_mode.h
- 库文件: libcvi_mode.a/libcvi_mode.so

【注意】

无

【举例】

无

【相关主题】

无

4.4.4.2.8 CVI_MODEMNG_SetEmrState

【描述】

紧急录像使能

【语法】

```
int32_t CVI_MODEMNG_SetEmrState(bool en);
```

【参数】

参数名称	描述	输入/输出
en	使能	输入

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_mode.h
- 库文件: libcvi_mode.a/libcvi_mode.so

【注意】

无

【举例】

无

【相关主题】

无

4.4.4.2.9 CVI_MODEMNG_LiveViewUp

【描述】

预览视角上移

【语法】

```
int32_t CVI_MODEMNG_LiveViewUp(uint32_t viewwin);
```

【参数】

参数名称	描述	输入/输出
viewwin	视窗	输入

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_mode.h
- 库文件: libcvi_mode.a/libcvi_mode.so

【注意】

无

【举例】

无

【相关主题】

无

4.4.4.2.10 CVI_MODEMNG_LiveViewDown

【描述】

预览视角上移

【语法】

```
int32_t CVI_MODEMNG_LiveViewDown(uint32_t viewwin);
```

【参数】

参数名称	描述	输入/输出
viewwin	视窗	输入

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_mode.h

- 库文件: libcvi_mode.a/libcvi_mode.so

【注意】

无

【举例】

无

【相关主题】

无

4.4.4.2.11 CVI_MODEMNG_GetCardState

【描述】

获取 sd 卡状态

【语法】

```
uint32_t CVI_MODEMNG_GetCardState(void);
```

【参数】

无

【返回值】

返回值	描述
非 0	sd 卡状态

【需求】

- 头文件: cvi_mode.h
- 库文件: libcvi_mode.a/libcvi_mode.so

【注意】

无

【举例】

无

【相关主题】

无

4.4.4.2.12 CVI_MODEMNG_GetCamStatus

【描述】

获取传感器是否运行

【语法】

```
void CVI_MODEMNG_GetCamStatus(bool *cam_enable);
```

【参数】

参数名称	描述	输入/输出
cam_enable	传感器状态数组	输入

【返回值】

无

【需求】

- 头文件: cvi_mode.h
- 库文件: libcvi_mode.a/libcvi_mode.so

【注意】

无

【举例】

无

【相关主题】

无

4.4.4.2.13 CVI_MODEMNG_GetCamIspInfoStatus

【描述】

获取传感器 isp 是否使能

【语法】

```
void CVI_MODEMNG_GetCamIspInfoStatus(bool *cam_isp_enable);
```

【参数】

参数名称	描述	输入/输出
cam_isp_enable	传感器 isp 状态数组	输入

【返回值】

无

【需求】

- 头文件: cvi_mode.h
- 库文件: libcvi_mode.a/libcvi_mode.so

【注意】

无

【举例】

无

【相关主题】

无

4.4.4.2.14 CVI_MODEMNG_SetCardState

【描述】

设置 sd 卡状态

【语法】

```
bool CVI_MODEMNG_SetCardState(uint32_t u32CardState);
```

【参数】

参数名称	描述	输入/输出
u32CardState	sd 卡状态	输入

【返回值】

返回值	描述
true	成功
false	失败

【需求】

- 头文件: cvi_mode.h
- 库文件: libcvi_mode.a/libcvi_mode.so

【注意】

无

【举例】

无

【相关主题】

无

4.4.4.2.15 CVI_MODEMNG_GetInProgress

【描述】

获取是否进行中

【语法】

```
bool CVI_MODEMNG_GetInProgress(void);
```

【参数】

无

【返回值】

返回值	描述
true	成功
false	失败

【需求】

- 头文件: cvi_mode.h
- 库文件: libcvi_mode.a/libcvi_mode.so

【注意】

无

【举例】

无

【相关主题】

无

4.4.4.2.16 CVI_MODEMNG_GetEmrState

【描述】

是否紧急录像中

【语法】

```
bool CVI_MODEMNG_GetEmrState(void);
```

【参数】

无

【返回值】

返回值	描述
true	成功
false	失败

【需求】

- 头文件: cvi_mode.h
- 库文件: libcvi_mode.a/libcvi_mode.so

【注意】

无

【举例】

无

【相关主题】

无

4.4.4.2.17 CVI_MODEMNG_TEST_MAIN_Create

【描述】

测试模式创建

【语法】

```
int32_t CVI_MODEMNG_TEST_MAIN_Create(void);
```

【参数】

无

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_mode.h
- 库文件: libcvi_mode.a/libcvi_mode.so

【注意】

无

【举例】

无

【相关主题】

无

4.4.4.2.18 CVI_MODEMNG_TEST_MAIN_Destroy

【描述】

测试模式销毁

【语法】

```
int32_t CVI_MODEMNG_TEST_MAIN_Destroy(void);
```

【参数】

无

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_mode.h
- 库文件: libcvi_mode.a/libcvi_mode.so

【注意】

无

【举例】

无

【相关主题】

无

4.4.4.2.19 CVI_MODEMNG_SetMediaLoopTime

【描述】

设置录像时长

【语法】

```
void CVI_MODEMNG_SetMediaLoopTime(int32_t value);
```

【参数】

参数名称	描述	输入/输出
value	时长	输入

【返回值】

无

【需求】

- 头文件: cvi_mode.h
- 库文件: libcvi_mode.a/libcvi_mode.so

【注意】

无

【举例】

无

【相关主题】

无

4.4.4.2.20 CVI_MODEMNG_SetMediaAudio**【描述】**

设置音频

【语法】

```
void CVI_MODEMNG_SetMediaAudio(int32_t value);
```

【参数】

参数名称	描述	输入/输出
value	使能	输入

【返回值】

无

【需求】

- 头文件: cvi_mode.h
- 库文件: libcvi_mode.a/libcvi_mode.so

【注意】

无

【举例】

无

【相关主题】

无

4.4.4.2.21 CVI_MODEMNG_SetMenuWifiStatus

【描述】

设置 WiFi 状态

【语法】

```
void CVI_MODEMNG_SetMenuWifiStatus(int32_t value);
```

【参数】

参数名称	描述	输入/输出
value	使能	输入

【返回值】

无

【需求】

- 头文件: cvi_mode.h
- 库文件: libcvi_mode.a/libcvi_mode.so

【注意】

无

【举例】

无

【相关主题】

无

4.4.4.2.22 CVI_MODEMNG_SetMediaOsd

【描述】

设置 osd 状态

【语法】

```
void CVI_MODEMNG_SetMediaOsd(int32_t value);
```

【参数】

参数名称	描述	输入/输出
value	使能	输入

【返回值】

无

【需求】

- 头文件: cvi_mode.h
- 库文件: libcvi_mode.a/libcvi_mode.so

【注意】

无

【举例】

无

【相关主题】

无

4.4.4.2.23 CVI_MODEMNG_SetMediaLapseTime

【描述】

设置缩时间隔

【语法】

```
void CVI_MODEMNG_SetMediaLapseTime(int32_t value);
```

【参数】

参数名称	描述	输入/输出
value	使能	输入

【返回值】

无

【需求】

- 头文件: cvi_mode.h
- 库文件: libcvi_mode.a/libcvi_mode.so

【注意】

无

【举例】

无

【相关主题】

无

4.4.4.2.24 CVI_MODEMNG_SetMediaVencFormat

【描述】

设置编码格式

【语法】

```
void CVI_MODEMNG_SetMediaVencFormat(int32_t value);
```

【参数】

参数名称	描述	输入/输出
value	使能	输入

【返回值】

无

【需求】

- 头文件: cvi_mode.h
- 库文件: libcvi_mode.a/libcvi_mode.so

【注意】

无

【举例】

无

【相关主题】

无

4.4.4.2.25 CVI_MODEMNG_LiveViewSwitch

【描述】

切换预览镜头

【语法】

```
int32_t CVI_MODEMNG_LiveViewSwitch(uint32_t viewwin);
```

【参数】

参数名称	描述	输入/输出
viewwin	视窗	输入

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_mode.h
- 库文件: libcvi_mode.a/libcvi_mode.so

【注意】

无

【举例】

无

【相关主题】

无

4.4.4.2.26 CVI_MODEMNG_RegisterEvent

【描述】

模式状态注册

【语法】

```
int32_t CVI_MODEMNG_RegisterEvent(void);
```

【参数】

无

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_mode.h
- 库文件: libcvi_mode.a/libcvi_mode.so

【注意】

无

【举例】

无

【相关主题】

无

4.4.4.2.27 CVI_MODEMNG_Format

【描述】

sd 卡格式化

【语法】

```
int32_t CVI_MODEMNG_Format(char *labelname);
```

【参数】

参数名称	描述	输入/输出
labelname	标签	输入

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_mode.h
- 库文件: libcvi_mode.a/libcvi_mode.so

【注意】

无

【举例】

无

【相关主题】

无

4.4.4.2.28 CVI_MODEMNG_OpenUvcMode

【描述】

打开 uvc 模式

【语法】

```
int32_t CVI_MODEMNG_OpenUvcMode(void);
```

【参数】

无

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_mode.h
- 库文件: libcvi_mode.a/libcvi_mode.so

【注意】

无

【举例】

无

【相关主题】

无

4.4.4.2.29 CVI_MODEMNG_CloseUvcMode**【描述】**

关闭 uvc 模式

【语法】

```
int32_t CVI_MODEMNG_CloseUvcMode(void);
```

【参数】

无

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_mode.h
- 库文件: libcvi_mode.a/libcvi_mode.so

【注意】

无

【举例】

无

【相关主题】

无

4.4.4.2.30 CVI_MODEMNG_ContextInit**【描述】**

模式管理配置初始化

【语法】

```
int32_t CVI_MODEMNG_ContextInit(const CVI_MODEMNG_CONFIG_S* pstModemngCfg);
```

【参数】

参数名称	描述	输入/输出
stModemngCfg	模式管理配置	输入

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_mode.h
- 库文件: libcvi_mode.a/libcvi_mode.so

【注意】

无

【举例】

无

【相关主题】

无

4.4.4.2.31 CVI_MODEMNG_SetDefaultParam**【描述】**

设置默认参数

【语法】

```
int32_t CVI_MODEMNG_SetDefaultParam(void);
```

【参数】

无

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_mode.h
- 库文件: libcvi_mode.a/libcvi_mode.so

【注意】

无

【举例】

无

【相关主题】

无

4.4.4.2.32 CVI_MODEMNG_OpenPhotoMode

【描述】

打开照相模式

【语法】

```
int32_t CVI_MODEMNG_OpenPhotoMode(void);
```

【参数】

无

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_mode.h
- 库文件: libcvi_mode.a/libcvi_mode.so

【注意】

无

【举例】

无

【相关主题】

无

4.4.4.2.33 CVI_MODEMNG_ClosePhotoMode

【描述】

关闭照相模式

【语法】

```
int32_t CVI_MODEMNG_ClosePhotoMode(void);
```

【参数】

无

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_mode.h
- 库文件: libcvi_mode.a/libcvi_mode.so

【注意】

无

【举例】

无

【相关主题】

无

4.4.4.2.34 CVI_MODEMNG_SetParkingRec

【描述】

使能泊车录像

【语法】

```
int32_t CVI_MODEMNG_SetParkingRec(bool en);
```


【参数】

参数名称	描述	输入/输出
en	使能	输入

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件：cvi_mode.h
- 库文件：libcvi_mode.a/libcvi_mode.so

【注意】

无

【举例】

无

【相关主题】

无

4.4.4.3 数据结构

相关数据类型定义如下：

- `CVI_MODEMNG_EXIT_MODE_E`：触发退出模式管理事件枚举
- `CVI_MODEMNG_EXIT_MODE_CALLBACK`：定义退出模式管理时回调函数
- `CVI_MODEMNG_CONFIG_S`：模式管理配置

4.4.4.3.1 CVI_MODEMNG_EXIT_MODE_E**【说明】**

触发退出模式管理事件枚举

【定义】

```
typedef enum cviEXIT_MODE_E
{
    CVI_MODEMNG_EXIT_MODE_POWEROFF,
    CVI_MODEMNG_EXIT_MODE_REBOOT,
    CVI_MODEMNG_EXIT_MODE_SERVICE_QUIT,
    CVI_MODEMNG_EXIT_MODE_BUTT
} CVI_MODEMNG_EXIT_MODE_E;
```

【成员】

成员名称	描述
CVI_MODEMNG_EXIT_MODE_POWEROFF	电源关闭
CVI_MODEMNG_EXIT_MODE_REBOOT	重启
CVI_MODEMNG_EXIT_MODE_SERVICE_EXIT	服务退出

【注意事项】

无。

【相关数据类型及接口】

无。

4.4.4.3.2 CVI_MODEMNG_EXIT_MODE_CALLBACK**【说明】**

定义退出模式管理时回调函数

【定义】

```
typedef int32_t (*CVI_MODEMNG_EXIT_MODE_CALLBACK)(CVI_MODEMNG_EXIT_MODE_
↪MODE_E enExitMode);
```

【成员】

成员名称	描述
enExitMode	触发退出模式管理事件

【注意事项】

无。

【相关数据类型及接口】

无。

4.4.4.3.3 CVI_MODEMNG_CONFIG_S**【说明】**

模式管理配置

【定义】

```
typedef struct cviPDT_MODEMNG_CONFIG_S
{
    CVI_MODEMNG_EXIT_MODE_CALLBACK pfnExitCB;
} CVI_MODEMNG_CONFIG_S;
```

【成员】

成员名称	描述
pfnExitCB	退出模式管理时回调函数

【注意事项】

无。

【相关数据类型及接口】

无。

4.5 网络管理

网络管理 (NetCtrl)，是采用 custom= 或 cgi-bin 格式操作设备的功能模块，实际应用中，可用于 web 浏览器、手机 app 对接相关的应用程序，比如修改 WIFI 用户名与密码、通过手机拍照、开关录像音频。网络控制模块在系统中的位置，如图 2-1 示。

4.5.1 工作流程

网络控制依赖 Web 服务器 thttpd、通用组件 CviNet，其工作过程如图 4-9 所示：

- 客户端通过 http 请求的方式，将 CGI 命令与参数发送出去，图 4-9 步骤 1
- Web 服务器 thttpd 接收到 http 请求后，将合法请求的内容移交给 Cvinet 模块进行解析，图 4-9 步骤 2
- Cvinet 模块对请求内容进行解析后，得出 CGI 命令与对应的参数列表；根据 CGI 命令找到对应的 CGI 命令回调函数，执行回调函数，图 4-9 步骤 3
- NetCtrl 模块实现了 CGI 命令回调函数，它调用具体功能模块接口，例如 Param、Statemng 等模块的接口，实现具体的功能；然后将执行结果通过特定格式，以字符串的形式返回客户端，图 4-9 步骤 4

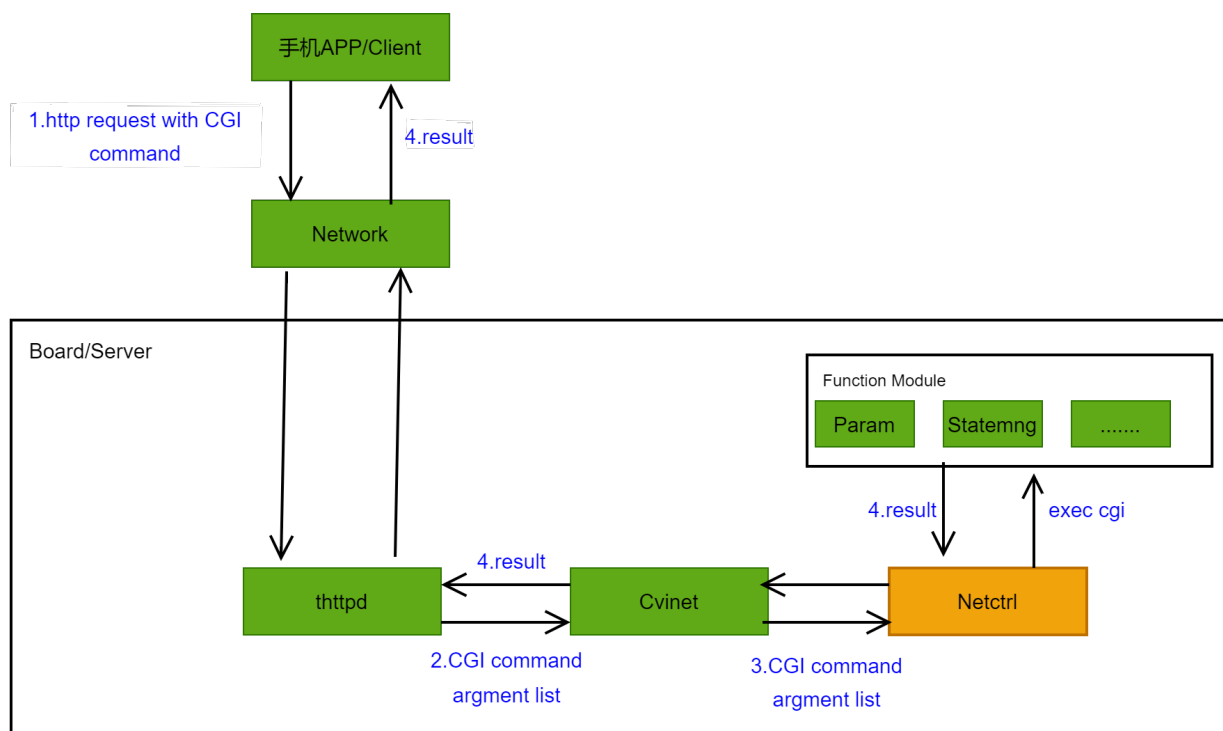


图 4.9: Netctrl 工作流程

4.5.2 http 请求构成

http 请求是一个 url，由以下字段组成，如图 4-10 所示：

- http 协议头
- 设备端 IP 地址
- cgi 样式 /cgi-bin/ 或者?custom= 样式
- CGI 命令
- ? (连接符)
- 参数列表 (可选段，多个参数之间用 & 符号连接)

http://	IP	/cgi-bin/ or ?custom=	命令	?	paramlist
---------	----	--------------------------	----	---	-----------

图 4.10: http request url 构成

当前通用指令格式：http://IP/?custom=1&cmd=指令&par=参数格式，例如：http://192.168.1.254/?
 ↳ custom=1&cmd=1002&par=0

另外也可支持CGI格式：http://IP/cgi-bin/Config.cgi?action=set(get)&property=指令&

↳ value=参数格式，例如：http://192.168.1.1/cgi-bin/Config.cgi?action=set&property=LoopingVideo&
 ↳ value=1MIN，这是设置录像时长为1min (下页继续)

(续上页)

4.5.3 开发参考

4.5.3.1 网络管理通用 API

该功能模块为用户提供以下 API:

- CVI_NETCTRL_Init : 网络管理模块的初始化
- CVI_NETCTRL_DeInit : 网络管理模块的去初始化
- CVI_NETCTRL_NetToUiConnectState : 网络是否连接 UI
- CVI_XML_GetPicture : 获取照片
- CVI_XML_SetCapture : 设置抓拍分辨率
- CVI_XML_GetFreePicture : 查询可拍照片数量
- CVI_XML_StartandStopMovieRecord : 开始/停止录像
- CVI_XML_SetMovieRecord : 设置录像分辨率
- CVI_XML_SetCyclicRecordValue : 设置循环录像分辨率
- CVI_XML_EdisableMoviehdr : 设置 WDR
- CVI_XML_SetMovieEvValue : 设置曝光补偿
- CVI_XML_EdisableMotionDetection : 设置运动检测
- CVI_XML_EdisableMovieAudio : 录音开关
- CVI_XML_EdisableDateInprint : 时间水印开关
- CVI_XML_GetMovieMaxRecordTime : 查询可录像时间
- CVI_XML_ChangeLiveviewSize : 设置预览大小
- CVI_XML_EdisableMovieGSensorSensitivity : 设置 gSensor 灵敏度
- CVI_XML_SetAutoRecording : 自动录像开关
- CVI_XML_SetMovieRecordBitrate : 设置录像码率
- CVI_XML_SetMovieLiveViewBitrate : 设置预览码率
- CVI_XML_StartandStopMovieLiveview : 重新开启/停止直播流
- CVI_XML_GetMovieRecordingTime : 当前循环时间段录像时间
- CVI_XML_TriggerRAWencode : 触发截帧拍照动作
- CVI_XML_GetRAWencodeJPEG : 获取截帧拍照的图片
- CVI_XML_GetStreamingAddr : 获取直播流地址
- CVI_XML_SetMovieFliphorMirror : 设置镜像方式
- CVI_XML_ChangeSystemMode : 切换模式

- CVI_XML_QueryCurrentSupportCommand : 查询可以支持的 cmd
- CVI_XML_SetWiFiSSID : 设置 SSID
- CVI_XML_SetWiFiPassPhrase : 设置 wifi 密码
- CVI_XML_SetSystemDate : 设置日期
- CVI_XML_SetSystemTime : 设置时间
- CVI_XML_SelectLanguage : 语言设置
- CVI_XML_CommandWouldFormatStorage : 格式化 nand 或 SD
- CVI_XML_ResetallSettingTodeFault : 恢复出厂设置
- CVI_XML_GetProjectVersion : 查看固件版本
- CVI_XML_StartToupdateFirmware : 开始烧录固件
- CVI_XML_QueryaAllSettingCommandStatus : 查询各个设置状态
- CVI_XML_ListAllFileCreatTime : 查询文件清单
- CVI_XML_CommandSmartPhoneCheck : 发送心跳包
- CVI_XML_GetDiskFreeSpace : 查询 SD 卡剩余空间
- CVI_XML_ReconnectWiFi : 重启 WIFI
- CVI_XML_SaveMenuInformation : 将保存设置写进 flash
- CVI_XML_GetCardsSatus : 获取 SD 卡的状态
- CVI_XML_GetUpdateFWpath : 上传 FW 的 bin 文件到 DVR
- CVI_XML_SetUpPIPStyle : 设置 pip 方式
- CVI_XML_GetSSIDandPassPhrase : 获取 wifi 密码
- CVI_XML_GetMovieSizeCapacity : 录像可以支持的录像分辨率
- CVI_XML_QueryMenuItem : 查询支持的菜单设置项
- CVI_XML_QueryCurrentDVRmode : 查询当前模式
- CVI_XML_GetThumbnail : 获取缩略图 (列表小图)
- CVI_XML_DeleteOneFile : 删除指定路径的文件
- CVI_XML_DeleteAll : 删除所有文件
- CVI_XML_GetMovieFileInformation : 获取指定文件的宽高信息
- CVI_XML_RoadCamStart : RoadCam 开机报文
- CVI_XML_PhoneApp : 6 帧探开机报文
- CVI_XML_SensorNum : 摄像头个数
- CVI_CGI_SetMovieRecord : 视频分辨率的设置
- CVI_CGI_SetCyclicRecordValue : 录影分段时间设置
- CVI_CGI_SetGSensorSensitivity : 碰撞锁定的灵敏度的设置
- CVI_CGI_SetVideoTimeLapse : 缩时录影的设置

- CVI_CGI_SetMovieEvValue : 曝光设定的设置
- CVI_CGI_CommandWouldFormatStorage : 格式化 SD 卡
- CVI_CGI_DeleteOneFile : 删除文件
- CVI_CGI_SetGetSSIDandPassPhrase : 设置 wifi 账号和密码
- CVI_CGI_SetSSIDandPassPhrase : 获取 wifi 账号和密码
- CVI_CGI_ResetAllSettingTodeFault : 恢复出厂设置
- CVI_CGI_GetProjectVersion : 版本值获取
- CVI_CGI_GetMenuUUID : 设备的唯一标志
- CVI_CGI_GetMJPEGStatus : 获取当前设备的录像状态
- CVI_CGI_GetPreviewStatus : 获取当前设备的视频流状态
- CVI_CGI_GetCamMenu : 获取相机目录
- CVI_CGI_GetEdisableMovieAudio : 获取录音开关
- CVI_CGI_SetEdisableMovieAudio : 设置录音开关
- CVI_CGI_Heartbeat : 心跳包
- CVI_CGI_MovieRecord : 更改录像, 录像, 停止录像, 拍照。
- CVI_CGI_CardInfo : 获取 SD 卡信息
- CVI_CGI_VideoresCapability : 是否可以录像
- CVI_CGI_ListAllFileCreatTime : 获取文件清单
- CVI_CGI_GetScrennail : 获取文件缩略图
- CVI_CGI_SetSystemTime : 设置当前时间

4.5.3.1.1 CVI_NETCTRL_Init

【描述】

网络管理模块的初始化

【语法】

```
int32_t CVI_NETCTRL_Init(void);
```

【参数】

无

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_netcmdregistration.h
- 库文件: libcvi_netctrl.a/libcvi_netctrl.so

【注意】

无

【举例】

无

【相关主题】

无

4.5.3.1.2 CVI_NETCTRL_DeInit

【描述】

网络管理模块的去初始化

【语法】

```
int32_t CVI_NETCTRL_DeInit(void)
```

【参数】

无

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_netcmdregistration.h
- 库文件: libcvi_netctrl.a/libcvi_netctrl.so

【注意】

无

【举例】

无

【相关主题】

无

4.5.3.1.3 CVI_NETCTRL_NetToUiConnectState

【描述】

网络是否连接 UI

【语法】

```
int32_t CVI_NETCTRL_NetToUiConnectState(void);
```

【参数】

无

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_netcmdregistration.h
- 库文件: libcvi_netctrl.a/libcvi_netctrl.so

【注意】

无

【举例】

无

【相关主题】

无

4.5.3.1.4 CVI_XML_GetPicture

【描述】

获取照片

【语法】

```
int32_t CVI_XML_GetPicture(httpd_conn *hc, char *str);
```

【参数】

参数名称	描述	输入/输出
hc	http 协议结构体	输入
str	执行参数	输入

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_netctrl.h
- 库文件: libcvi_netctrl.a/libcvi_netctrl.so

【网络 URL】

http://192.168.1.254/?custom=1&cmd=1001

【网络 URL 参数】

无

【http 返回报文信息】

```
<?xml version="1.0" encoding="UTF-8" ?>
<Function>
<Cmd>1001</Cmd>
<Status>0</Status>
<File>
<NAME>2000_0101_002223_001.JPG</NAME>
<FPATH>A:\Novatek\Photo\2000_0101_002223_001.JPG</FPATH>
</File>
<FREEPICNUM>317</FREEPICNUM>
</Function>
```

【举例】

无

4.5.3.1.5 CVI_XML_SetCapture**【描述】**

设置抓拍分辨率

【语法】

```
int32_t CVI_XML_SetCapture(httpd_conn *hc, char *str)
```

【参数】

参数名称	描述	输入/输出
hc	http 协议结构体	输入
str	执行参数	输入

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: `cvi_netctrl.h`
- 库文件: `libcvi_netctrl.a/libcvi_netctrl.so`

【网络 URL】

<http://192.168.1.254/?custom=1&cmd=1002&par=0>

【网络 URL 参数】

参数名称	描述
0	1440P
1	1080P

【http 返回报文信息】**【举例】**

无

4.5.3.1.6 CVI_XML_SetMovieRecordBitrate**【描述】**

设置录像码率

【语法】

```
int32_t CVI_XML_SetMovieRecordBitrate(httpd_conn *hc, char *str)
```

【参数】

参数名称	描述	输入/输出
hc	http 协议结构体	输入
str	执行参数	输入

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: `cvi_netctrl.h`
- 库文件: `libcvi_netctrl.a/libcvi_netctrl.so`

【网络 URL】

<http://192.168.1.254/?custom=1&cmd=2013&str=400>

【网络 URL 参数】

参数名称	描述
400	400kbp/s

【http 返回报文信息】**【举例】**

无

4.5.3.1.7 CVI_XML_SetMovieLiveViewBitrate**【描述】**

设置预览码率

【语法】

```
int32_t CVI_XML_SetMovieLiveViewBitrate(httpd_conn *hc, char *str)
```

【参数】

参数名称	描述	输入/输出
hc	http 协议结构体	输入
str	执行参数	输入

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_netctrl.h
- 库文件: libcvi_netctrl.a/libcvi_netctrl.so

【网络 URL】

<http://192.168.1.254/?custom=1&cmd=2014&str=300>

【网络 URL 参数】

参数名称	描述
300	码率 300kbp/s

【http 返回报文信息】

【举例】

无

4.5.3.1.8 CVI_XML_ChangeLiveviewSize**【描述】**

设置预览大小

【语法】

```
int32_t CVI_XML_ChangeLiveviewSize(httpd_conn *hc, char *str)
```

【参数】

参数名称	描述	输入/输出
hc	http 协议结构体	输入
str	执行参数	输入

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_netctrl.h
- 库文件: libcvi_netctrl.a/libcvi_netctrl.so

【网络 URL】

<http://192.168.1.254/?custom=1&cmd=2010&par=2>

【网络 URL 参数】

参数名称	描述
2	

【http 返回报文信息】**【举例】**

无

4.5.3.1.9 CVI_XML_EdisableMoviehdr

【描述】

设置 WDR

【语法】

```
int32_t CVI_XML_EdisableMoviehdr(httpd_conn *hc, char *str)
```

【参数】

参数名称	描述	输入/输出
hc	http 协议结构体	输入
str	执行参数	输入

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_netctrl.h
- 库文件: libcvi_netctrl.a/libcvi_netctrl.so

【网络 URL】

<http://192.168.1.254/?custom=1&cmd=2004&par=0>

【网络 URL 参数】

参数名称	描述
0	关闭
1	开启

【http 返回报文信息】

【举例】

无

4.5.3.1.10 CVI_XML_EdisableMotionDetection

【描述】

设置运动检测

【语法】

```
int32_t CVI_XML_EdisableMotionDetection(httpd_conn *hc, char *str)
```

【参数】

参数名称	描述	输入/输出
hc	http 协议结构体	输入
str	执行参数	输入

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_netctrl.h
- 库文件: libcvi_netctrl.a/libcvi_netctrl.so

【网络 URL】

<http://192.168.1.254/?custom=1&cmd=2006&par=1>

【网络 URL 参数】

参数名称	描述
0	关闭
1	开启

【http 返回报文信息】

【举例】

无

4.5.3.1.11 CVI_XML_GetFreePicture

【描述】

查询可拍照片数量

【语法】

```
int32_t CVI_XML_GetFreePicture (httpd_conn *hc, char *str);
```

【参数】

参数名称	描述	输入/输出
hc	http 协议结构体	输入
str	执行参数	输入

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_netctrl.h
- 库文件: libcvi_netctrl.a/libcvi_netctrl.so

【网络 URL】

<http://192.168.1.254/?custom=1&cmd=1003>

【网络 URL 参数】

无

【http 返回报文信息】

```
<?xml version="1.0" encoding="UTF-8" ?>
<Function>
<Cmd>1001</Cmd>
<Status>0</Status>
<Value>100</Value>
</Function>
```

【举例】

无

4.5.3.1.12 CVI_XML_StartandStopMovieRecord

【描述】

开始/停止录像

【语法】

```
int32_t CVI_XML_StartandStopMovieRecord (httpd_conn *hc, char *str);
```

【参数】

参数名称	描述	输入/输出
hc	http 协议结构体	输入
str	执行参数	输入

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_netctrl.h
- 库文件: libcvi_netctrl.a/libcvi_netctrl.so

【网络 URL】

<http://192.168.1.254/?custom=1&cmd=2001&par=1>

【网络 URL 参数】

参数名称	描述
0	停止录像
1	开始录像

【http 返回报文信息】

```
<?xml version="1.0" encoding="UTF-8" ?>
<Function>
<Cmd>2001</Cmd>
<Status>0</Status>
</Function>
```

【举例】

无

4.5.3.1.13 CVI_XML_SetMovieRecord

【描述】

设置录像分辨率

【语法】

```
int32_t CVI_XML_SetMovieRecord (httpd_conn *hc, char *str);
```

【参数】

参数名称	描述	输入/输出
hc	http 协议结构体	输入
str	执行参数	输入

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_netctrl.h
- 库文件: libcvi_netctrl.a/libcvi_netctrl.so

【网络 URL】

<http://192.168.1.254/?custom=1&cmd=2002&par=1>

【网络 URL 参数】

参数名称	描述
0	720p
3	1080p
7	1440p

【http 返回报文信息】

```
<?xml version="1.0" encoding="UTF-8" ?>
<Function>
<Cmd>2002</Cmd>
<Status>0</Status>
</Function>
```

【举例】

无

4.5.3.1.14 CVI_XML_SetCyclicRecordValue

【描述】

设置录像循环录像时间

【语法】

```
int32_t CVI_XML_SetCyclicRecordValue (httpd_conn *hc, char *str);
```

【参数】

参数名称	描述	输入/输出
hc	http 协议结构体	输入
str	执行参数	输入

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_netctrl.h
- 库文件: libcvi_netctrl.a/libcvi_netctrl.so

【网络 URL】

<http://192.168.1.254/?custom=1&cmd=2003&par=1>

【网络 URL 参数】

参数名称	描述
0	1 MIN
1	3 MIN
2	5 MIN

【http 返回报文信息】

```
<?xml version="1.0" encoding="UTF-8" ?>
<Function>
<Cmd>2003</Cmd>
<Status>0</Status>
</Function>
```

【举例】

无

4.5.3.1.15 CVI_XML_SetMovieEvValue

【描述】

设置曝光补偿

【语法】

```
int32_t CVI_XML_SetMovieEvValue (httpd_conn *hc, char *str);
```

【参数】

参数名称	描述	输入/输出
hc	http 协议结构体	输入
str	执行参数	输入

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_netctrl.h
- 库文件: libcvi_netctrl.a/libcvi_netctrl.so

【网络 URL】

<http://192.168.1.254/?custom=1&cmd=2005&par=1>

【网络 URL 参数】

参数名称	描述
0	EVP200
1	EVP167
2	EVP133
3	EVP100
4	EVP67
5	EVP33
6	EV0
7	EVN33
8	EVN67
9	EVN100
10	EVN133
11	EVN167
12	EVN200

【http 返回报文信息】

```
<?xml version="1.0" encoding="UTF-8" ?>
<Function>
<Cmd>2005</Cmd>
<Status>0</Status>
</Function>
```

【举例】

无

4.5.3.1.16 CVI_XML_EdisableMovieAudio**【描述】**

开始/停止录音

【语法】

```
int32_t CVI_XML_EdisableMovieAudio (httpd_conn *hc, char *str);
```

【参数】

参数名称	描述	输入/输出
hc	http 协议结构体	输入
str	执行参数	输入

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_netctrl.h
- 库文件: libcvi_netctrl.a/libcvi_netctrl.so

【网络 URL】

<http://192.168.1.254/?custom=1&cmd=2007&par=1>

【网络 URL 参数】

参数名称	描述
0	停止录音
1	开始录音

【http 返回报文信息】

```
<?xml version="1.0" encoding="UTF-8" ?>
<Function>
<Cmd>2007</Cmd>
<Status>0</Status>
</Function>
```

【举例】

无

4.5.3.1.17 CVI_XML_EdisableDateInprint**【描述】**

时间水印开关

【语法】

```
int32_t CVI_XML_EdisableDateInprint (httpd_conn *hc, char *str);
```

【参数】

参数名称	描述	输入/输出
hc	http 协议结构体	输入
str	执行参数	输入

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_netctrl.h
- 库文件: libcvi_netctrl.a/libcvi_netctrl.so

【网络 URL】

<http://192.168.1.254/?custom=1&cmd=1001>

【网络 URL 参数】

参数名称	描述
0	关闭时间水印
1	开启时间水印

【http 返回报文信息】

```
<?xml version="1.0" encoding="UTF-8" ?>
<Function>
<Cmd>2008</Cmd>
<Status>0</Status>
</Function>
```

【举例】

无

4.5.3.1.18 CVI_XML_GetMovieMaxRecordTime**【描述】**

查询最大可录像时间

【语法】

```
int32_t CVI_XML_GetMovieMaxRecordTime (httpd_conn *hc, char *str);
```

【参数】

参数名称	描述	输入/输出
hc	http 协议结构体	输入
str	执行参数	输入

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_netctrl.h
- 库文件: libcvi_netctrl.a/libcvi_netctrl.so

【网络 URL】

<http://192.168.1.254/?custom=1&cmd=2009>

【网络 URL 参数】

无

【http 返回报文信息】

```
<?xml version="1.0" encoding="UTF-8" ?>
<Function>
<Cmd>2009</Cmd>
<Status>0</Status>
<Value>471</Value>
</Function>
```

【Value 值】

Value	描述
471	最大可录像时间数值

【举例】

无

4.5.3.1.19 CVI_XML_EDisableMovieGSensorSensitivity

【描述】

设置 GSensor 灵敏度

【语法】

```
int32_t CVI_XML_EDisableMovieGSensorSensitivity (httpd_conn *hc, char *str);
```

【参数】

参数名称	描述	输入/输出
hc	http 协议结构体	输入
str	执行参数	输入

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_netctrl.h
- 库文件: libcvi_netctrl.a/libcvi_netctrl.so

【网络 URL】

<http://192.168.1.254/?custom=1&cmd=2011&par=1>

【网络 URL 参数】

参数名称	描述
0	关闭 Gsensor
1	低
2	中
3	高

【http 返回报文信息】


```
<?xml version="1.0" encoding="UTF-8" ?>
<Function>
<Cmd>2011</Cmd>
<Status>0</Status>
</Function>
```

【举例】

无

4.5.3.1.20 CVI_XML_SetAutoRecording**【描述】**

自动录像开关

【语法】

```
int32_t CVI_XML_SetAutoRecording (httpd_conn *hc, char *str);
```

【参数】

参数名称	描述	输入/输出
hc	http 协议结构体	输入
str	执行参数	输入

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_netctrl.h
- 库文件: libcvi_netctrl.a/libcvi_netctrl.so

【网络 URL】

<http://192.168.1.254/?custom=1&cmd=2012&par=1>

【网络 URL 参数】

参数名称	描述
0	关闭自动录像
1	开启自动录像

【http 返回报文信息】

```
<?xml version="1.0" encoding="UTF-8" ?>
<Function>
<Cmd>2012</Cmd>
<Status>0</Status>
</Function>
```

【举例】

无

4.5.3.1.21 CVI_XML_StartandStopMovieLiveview**【描述】**

开启/关闭直播流

【语法】

```
int32_t CVI_XML_StartandStopMovieLiveview (httpd_conn *hc, char *str);
```

【参数】

参数名称	描述	输入/输出
hc	http 协议结构体	输入
str	执行参数	输入

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_netctrl.h
- 库文件: libcvi_netctrl.a/libcvi_netctrl.so

【网络 URL】

<http://192.168.1.254/?custom=1&cmd=2015&par=1>

【网络 URL 参数】

参数名称	描述
0	关闭直播流
1	开启直播流

【http 返回报文信息】

```
<?xml version="1.0" encoding="UTF-8" ?>
<Function>
<Cmd>2015</Cmd>
<Status>0</Status>
</Function>
```

【举例】

无

4.5.3.1.22 CVI_XML_GetMovieRecordingTime**【描述】**

查询录像状态

【语法】

```
int32_t CVI_XML_GetMovieRecordingTime (httpd_conn *hc, char *str);
```

【参数】

参数名称	描述	输入/输出
hc	http 协议结构体	输入
str	执行参数	输入

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_netctrl.h
- 库文件: libcvi_netctrl.a/libcvi_netctrl.so

【网络 URL】

<http://192.168.1.254/?custom=1&cmd=2016>

【网络 URL 参数】

无

【http 返回报文信息】

```
<?xml version="1.0" encoding="UTF-8" ?>
<Function>
<Cmd>2016</Cmd>
<Status>0</Status>
<Value>1</Value>
</Function>
```

【举例】

无

4.5.3.1.23 CVI_XML_TriggerRAWencode**【描述】**

触发截帧拍照动作

【语法】

```
int32_t CVI_XML_TriggerRAWencode (httpd_conn *hc, char *str);
```

【参数】

参数名称	描述	输入/输出
hc	http 协议结构体	输入
str	执行参数	输入

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_netctrl.h
- 库文件: libcvi_netctrl.a/libcvi_netctrl.so

【网络 URL】

<http://192.168.1.254/?custom=1&cmd=2017>

【网络 URL 参数】

无

【http 返回报文信息】

```
<?xml version="1.0" encoding="UTF-8" ?>
<Function>
<Cmd>2017</Cmd>
<Status>0</Status>
</Function>
```

【举例】

无

4.5.3.1.24 CVI_XML_GetRAWencodeJPEG

【描述】

获取截帧拍照图像

【语法】

```
int32_t CVI_XML_GetRAWencodeJPEG (httpd_conn *hc, char *str);
```

【参数】

参数名称	描述	输入/输出
hc	http 协议结构体	输入
str	执行参数	输入

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_netctrl.h
- 库文件: libcvi_netctrl.a/libcvi_netctrl.so

【网络 URL】

<http://192.168.1.254/?custom=1&cmd=2018>

【网络 URL 参数】

无

【http 返回报文信息】

```
<?xml version="1.0" encoding="UTF-8" ?>
<Function>
<Cmd>2018</Cmd>
<Status>0</Status>
</Function>
```

【举例】

无

4.5.3.1.25 CVI_XML_GetStreamingAddr

【描述】

获取直播流地址

【语法】

```
int32_t CVI_XML_GetStreamingAddr (httpd_conn *hc, char *str);
```

【参数】

参数名称	描述	输入/输出
hc	http 协议结构体	输入
str	执行参数	输入

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_netctrl.h
- 库文件: libcvi_netctrl.a/libcvi_netctrl.so

【网络 URL】

<http://192.168.1.254/?custom=1&cmd=2019>

【网络 URL 参数】

无

【http 返回报文信息】

```
<?xml version="1.0" encoding="UTF-8" ?>
<List>
<MovieLiveViewLink>rtsp://192.168.1.254/cvi_cam_phone_app</MovieLiveViewLink>
</List>
```

【举例】

无

4.5.3.1.26 CVI_XML_SetMovieFliphorMirror

【描述】

设置镜像方式

【语法】

```
int32_t CVI_XML_SetMovieFliphorMirror (httpd_conn *hc, char *str);
```

【参数】

参数名称	描述	输入/输出
hc	http 协议结构体	输入
str	执行参数	输入

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_netctrl.h
- 库文件: libcvi_netctrl.a/libcvi_netctrl.so

【网络 URL】

<http://192.168.1.254/?custom=1&cmd=2023&par=1>

【网络 URL 参数】

参数名称	描述
0	正
1	反

【http 返回报文信息】

```
<?xml version="1.0" encoding="UTF-8" ?>
<Function>
<Cmd>2023</Cmd>
<Status>0</Status>
</Function>
```

【举例】

无

4.5.3.1.27 CVI_XML_ChangeSystemMode

【描述】

切换模式

【语法】

```
int32_t CVI_XML_ChangeSystemMode (httpd_conn *hc, char *str);
```

【参数】

参数名称	描述	输入/输出
hc	http 协议结构体	输入
str	执行参数	输入

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_netctrl.h
- 库文件: libcvi_netctrl.a/libcvi_netctrl.so

【网络 URL】

<http://192.168.1.254/?custom=1&cmd=3001&par=1>

【网络 URL 参数】

参数名称	描述
0	拍照模式
1	录像模式（默认）
2	回放模式

【http 返回报文信息】

```
<?xml version="1.0" encoding="UTF-8" ?>
<Function>
<Cmd>3001</Cmd>
<Status>0</Status>
</Function>
```

【举例】

无

4.5.3.1.28 CVI_XML_QueryCurrentSupportCommand

【描述】

查询可以支持的 cmd

【语法】

```
int32_t CVI_XML_QueryCurrentSupportCommand (httpd_conn *hc, char *str);
```

【参数】

参数名称	描述	输入/输出
hc	http 协议结构体	输入
str	执行参数	输入

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_netctrl.h
- 库文件: libcvi_netctrl.a/libcvi_netctrl.so

【网络 URL】

<http://192.168.1.254/?custom=1&cmd=3002&par=1>

【网络 URL 参数】

无

【http 返回报文信息】

```
<?xml version="1.0" encoding="UTF-8" ?>
<Function>
<Cmd>1001</Cmd>
<Cmd>1002</Cmd>
<Cmd>1003</Cmd>
<Cmd>2016</Cmd>
</Function>
```

【举例】

无

4.5.3.1.29 CVI_XML_SetWiFiSSID

【描述】

设置 WIFI 名称

【语法】

```
int32_t CVI_XML_SetWiFiSSID (httpd_conn *hc, char *str);
```

【参数】

参数名称	描述	输入/输出
hc	http 协议结构体	输入
str	执行参数	输入

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_netctrl.h
- 库文件: libcvi_netctrl.a/libcvi_netctrl.so

【网络 URL】

<http://192.168.1.254/?custom=1&cmd=3003&str=ABCDE>

【网络 URL 参数】

参数名称	描述
ABCDE	WIFI 名称

【http 返回报文信息】

```
<?xml version="1.0" encoding="UTF-8" ?>
<Function>
<Cmd>3003</Cmd>
<Status>0</Status>
</Function>
```

【举例】

无

4.5.3.1.30 CVI_XML_SetWiFiPassPhrase

【描述】

设置 WIFI 密码

【语法】

```
int32_t CVI_XML_SetWiFiPassPhrase (httpd_conn *hc, char *str);
```

【参数】

参数名称	描述	输入/输出
hc	http 协议结构体	输入
str	执行参数	输入

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_netctrl.h
- 库文件: libcvi_netctrl.a/libcvi_netctrl.so

【网络 URL】

<http://192.168.1.254/?custom=1&cmd=3004&str=12345678>

【网络 URL 参数】

参数名称	描述
12345678	WIFI 密码

【http 返回报文信息】

```
<?xml version="1.0" encoding="UTF-8" ?>
<Function>
<Cmd>3004</Cmd>
<Status>0</Status>
</Function>
```

【举例】

无

4.5.3.1.31 CVI_XML_SetSystemDate

【描述】

设置日期

【语法】

```
int32_t CVI_XML_SetSystemDate (httpd_conn *hc, char *str);
```

【参数】

参数名称	描述	输入/输出
hc	http 协议结构体	输入
str	执行参数	输入

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_netctrl.h
- 库文件: libcvi_netctrl.a/libcvi_netctrl.so

【网络 URL】

<http://192.168.1.254/?custom=1&cmd=3005&str=2023-12-08>

【网络 URL 参数】

参数名称	描述
2023-12-08	日期

【http 返回报文信息】

```
<?xml version="1.0" encoding="UTF-8" ?>
<Function>
<Cmd>3005</Cmd>
<Status>0</Status>
</Function>
```

【举例】

无

4.5.3.1.32 CVI_XML_SetSystemTime

【描述】

设置时间

【语法】

```
int32_t CVI_XML_SetSystemTime (httpd_conn *hc, char *str);
```

【参数】

参数名称	描述	输入/输出
hc	http 协议结构体	输入
str	执行参数	输入

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_netctrl.h
- 库文件: libcvi_netctrl.a/libcvi_netctrl.so

【网络 URL】

<http://192.168.1.254/?custom=1&cmd=3006&str=17:10:30>

【网络 URL 参数】

参数名称	描述
17:10:30	时间

【http 返回报文信息】

```
<?xml version="1.0" encoding="UTF-8" ?>
<Function>
<Cmd>3006</Cmd>
<Status>0</Status>
</Function>
```

【举例】

无

4.5.3.1.33 CVI_XML_SelectLanguage

【描述】

设置语言

【语法】

```
int32_t CVI_XML_SelectLanguage (httpd_conn *hc, char *str);
```

【参数】

参数名称	描述	输入/输出
hc	http 协议结构体	输入
str	执行参数	输入

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_netctrl.h
- 库文件: libcvi_netctrl.a/libcvi_netctrl.so

【网络 URL】

<http://192.168.1.254/?custom=1&cmd=3007&str=0>

【网络 URL 参数】

参数名称	描述
0	中文
1	英文

【http 返回报文信息】

```
<?xml version="1.0" encoding="UTF-8" ?>
<Function>
<Cmd>3007</Cmd>
<Status>0</Status>
</Function>
```

【举例】

无

4.5.3.1.34 CVI_XML_CommandWouldFormatStorage

【描述】

格式化 SD 卡

【语法】

```
int32_t CVI_XML_CommandWouldFormatStorage (httpd_conn *hc, char *str);
```

【参数】

参数名称	描述	输入/输出
hc	http 协议结构体	输入
str	执行参数	输入

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_netctrl.h
- 库文件: libcvi_netctrl.a/libcvi_netctrl.so

【网络 URL】

<http://192.168.1.254/?custom=1&cmd=3010&str=0>

【网络 URL 参数】

参数名称	描述
0	格式化闪存
1	格式化 SD 卡

【http 返回报文信息】

```
<?xml version="1.0" encoding="UTF-8" ?>
<Function>
<Cmd>3010</Cmd>
<Status>0</Status>
</Function>
```

【举例】

无

4.5.3.1.35 CVI_XML_ResetallSettingTodeFault

【描述】

恢复出厂设置

【语法】

```
int32_t CVI_XML_ResetallSettingTodeFault (httpd_conn *hc, char *str);
```

【参数】

参数名称	描述	输入/输出
hc	http 协议结构体	输入
str	执行参数	输入

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_netctrl.h
- 库文件: libcvi_netctrl.a/libcvi_netctrl.so

【网络 URL】

<http://192.168.1.254/?custom=1&cmd=3011>

【网络 URL 参数】

无

【http 返回报文信息】

```
<?xml version="1.0" encoding="UTF-8" ?>
<Function>
<Cmd>3011</Cmd>
<Status>0</Status>
</Function>
```

【举例】

无

4.5.3.1.36 CVI_XML_GetProjectVersion

【描述】

查看固件版本

【语法】

```
int32_t CVI_XML_GetProjectVersion (httpd_conn *hc, char *str);
```

【参数】

参数名称	描述	输入/输出
hc	http 协议结构体	输入
str	执行参数	输入

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_netctrl.h
- 库文件: libcvi_netctrl.a/libcvi_netctrl.so

【网络 URL】

<http://192.168.1.254/?custom=1&cmd=3012>

【网络 URL 参数】

无

【http 返回报文信息】

```
<?xml version="1.0" encoding="UTF-8" ?>
<Function>
<Cmd>3012</Cmd>
<Status>0</Status>
<String>ECOSDEMO00010027</String>
</Function>
```

【举例】

无

4.5.3.1.37 CVI_XML_StartTouupdateFirmware

【描述】

烧录固件版本

【语法】

```
int32_t CVI_XML_StartTouupdateFirmware (httpd_conn *hc, char *str);
```

【参数】

参数名称	描述	输入/输出
hc	http 协议结构体	输入
str	执行参数	输入

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_netctrl.h
- 库文件: libcvi_netctrl.a/libcvi_netctrl.so

【网络 URL】

http://192.168.1.254/?custom=1&cmd=3013&par=generic_0.1.8&1

【网络 URL 参数】

参数名称	描述
generic_0.1.8	版本号
更新方式: 0	直接更新 flash
更新方式: 1	更新 sd 卡内的固件

【http 返回报文信息】

```
<?xml version="1.0" encoding="UTF-8" ?>
<Function>
<Cmd>3013</Cmd>
<Status>0</Status>
</Function>
```

【举例】

无

4.5.3.1.38 CVI_XML_QueryAllSettingCommandStatus

【描述】

查询各个设置状态

【语法】

```
int32_t CVI_XML_QueryAllSettingCommandStatus (httpd_conn *hc, char *str);
```

【参数】

参数名称	描述	输入/输出
hc	http 协议结构体	输入
str	执行参数	输入

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_netctrl.h
- 库文件: libcvi_netctrl.a/libcvi_netctrl.so

【网络 URL】

<http://192.168.1.254/?custom=1&cmd=3014>

【网络 URL 参数】

无

【http 返回报文信息】

```
<?xml version="1.0" encoding="UTF-8" ?>
<Function>
<Cmd>1002</Cmd>
<Status>1</Status>
<Cmd>2016</Cmd>
<Status>0</Status>
<Cmd>2002</Cmd>
<Status>1</Status>
<Cmd>2003</Cmd>
<Status>1</Status>
</Function>
```

【举例】

无

4.5.3.1.39 CVI_XML_ListAllFileCreatTime

【描述】

查询文件清单

【语法】

```
int32_t CVI_XML_ListAllFileCreatTime (httpd_conn *hc, char *str);
```

【参数】

参数名称	描述	输入/输出
hc	http 协议结构体	输入
str	执行参数	输入

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_netctrl.h
- 库文件: libcvi_netctrl.a/libcvi_netctrl.so

【网络 URL】

<http://192.168.1.254/?custom=1&cmd=3015>

【网络 URL 参数】

无

【http 返回报文信息】

```
<xml version="1.0" encoding="UTF-8" ?>
<LIST>
<ALLFile>
<File>
<NAME>2000_0101_002223_001.JPG</NAME>
<FPATH>A:\NOVATEK\PHOTO\2000_0101_002223_001.JPG</FPATH>
<SIZE>2794693</SIZE>
<TIMECODE>673252044</TIMECODE>
<TIME>2000/01/01 00:22:24</TIME>
<ATTR>32</ATTR>
</File>
</ALLFile>
</LIST>
```

【举例】

无

4.5.3.1.40 CVI_XML_CommandSmartPhoneCheck

【描述】

发送心跳包

【语法】

```
int32_t CVI_XML_CommandSmartPhoneCheck (httpd_conn *hc, char *str);
```

【参数】

参数名称	描述	输入/输出
hc	http 协议结构体	输入
str	执行参数	输入

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_netctrl.h
- 库文件: libcvi_netctrl.a/libcvi_netctrl.so

【网络 URL】

<http://192.168.1.254/?custom=1&cmd=3016>

【网络 URL 参数】

无

【http 返回报文信息】

```
<?xml version="1.0" encoding="UTF-8" ?>

<Function>
<Cmd>3016</Cmd>
<Status>0</Status>
</Function>
```

【举例】

无

4.5.3.1.41 CVI_XML_GetDiskFreeSpace

【描述】

查询 SD 卡剩余空间

【语法】

```
int32_t CVI_XML_GetDiskFreeSpace (httpd_conn *hc, char *str);
```

【参数】

参数名称	描述	输入/输出
hc	http 协议结构体	输入
str	执行参数	输入

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_netctrl.h
- 库文件: libcvi_netctrl.a/libcvi_netctrl.so

【网络 URL】

<http://192.168.1.254/?custom=1&cmd=3017>

【网络 URL 参数】

无

【http 返回报文信息】

```
<?xml version="1.0" encoding="UTF-8" ?>
<Function>
<Cmd>3017</Cmd>
<Status>0</Status>
<Value>5159288832</Value>
</Function>
```

【Value】

Value	描述
5159288832	SD 卡剩余空间字节数

【举例】

无

4.5.3.1.42 CVI_XML_ReconnectWiFi

【描述】

重启 WiFi

【语法】

```
int32_t CVI_XML_ReconnectWiFi (httpd_conn *hc, char *str);
```

【参数】

参数名称	描述	输入/输出
hc	http 协议结构体	输入
str	执行参数	输入

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_netctrl.h
- 库文件: libcvi_netctrl.a/libcvi_netctrl.so

【网络 URL】

<http://192.168.1.254/?custom=1&cmd=3018>

【网络 URL 参数】

无

【http 返回报文信息】

```
<?xml version="1.0" encoding="UTF-8" ?>

<Function>
<Cmd>3018</Cmd>
<Status>0</Status>
</Function>
```

【举例】

无

4.5.3.1.43 CVI_XML_SaveMenuInformation

【描述】

将设置参数写入 Flash

【语法】

```
int32_t CVI_XML_SaveMenuInformation (httpd_conn *hc, char *str);
```

【参数】

参数名称	描述	输入/输出
hc	http 协议结构体	输入
str	执行参数	输入

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_netctrl.h
- 库文件: libcvi_netctrl.a/libcvi_netctrl.so

【网络 URL】

<http://192.168.1.254/?custom=1&cmd=3021>

【网络 URL 参数】

无

【http 返回报文信息】

```
<?xml version="1.0" encoding="UTF-8" ?>

<Function>
<Cmd>3021</Cmd>
<Status>0</Status>
</Function>
```

【举例】

无

4.5.3.1.44 CVI_XML_GetCardsSatus

【描述】

获取 SD 卡状态

【语法】

```
int32_t CVI_XML_GetCardsSatus (httpd_conn *hc, char *str);
```

【参数】

参数名称	描述	输入/输出
hc	http 协议结构体	输入
str	执行参数	输入

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_netctrl.h
- 库文件: libcvi_netctrl.a/libcvi_netctrl.so

【网络 URL】

<http://192.168.1.254/?custom=1&cmd=3024>

【网络 URL 参数】

无

【http 返回报文信息】

```
<?xml version="1.0" encoding="UTF-8" ?>
<Function>
<Cmd>3017</Cmd>
<Status>0</Status>
<Value>1</Value>
</Function>
```

【Value】

Value	描述
0	SD 卡未插入
1	SD 卡已插入

【举例】

无

4.5.3.1.45 CVI_XML_GetUpdateFWpath

【描述】

上传 FW 的 bin 文件到 DVR

【语法】

```
int32_t CVI_XML_GetUpdateFWpath (httpd_conn *hc, char *str);
```

【参数】

参数名称	描述	输入/输出
hc	http 协议结构体	输入
str	执行参数	输入

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_netctrl.h
- 库文件: libcvi_netctrl.a/libcvi_netctrl.so

【网络 URL】

```
<html> <body> <form name=frm2 action=" http://192.168.1.254/?custom=1&cmd=5001&par=e99e89853231ed639d9985ade8dc8a95" method=post enctype="multipart/form-data"> <input name=fileupload2 size=70 type=file><br/> <input name=upbtn type=submit value=" Upload Custom files" > </form> </div> </body> </html>
```

【网络 URL 参数】

无

【http 返回报文信息】

无

【举例】

无

4.5.3.1.46 CVI_XML_SetUpPIPStyle

【描述】

设置前/后路摄像头

【语法】

```
int32_t CVI_XML_SetUpPIPStyle (httpd_conn *hc, char *str);
```

【参数】

参数名称	描述	输入/输出
hc	http 协议结构体	输入
str	执行参数	输入

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_netctrl.h
- 库文件: libcvi_netctrl.a/libcvi_netctrl.so

【网络 URL】

<http://192.168.1.254/?custom=1&cmd=3028&str=0>

【网络 URL 参数】

参数名称	描述
0	前路
1	后路

【http 返回报文信息】

```
<?xml version="1.0" encoding="UTF-8" ?>
<Function>
<Cmd>3028</Cmd>
<Status>0</Status>
</Function>
```

【举例】

无

4.5.3.1.47 CVI_XML_GetSSIDandPassPhrase

【描述】

获取 WiFi 账号和密码

【语法】

```
int32_t CVI_XML_GetSSIDandPassPhrase (httpd_conn *hc, char *str);
```

【参数】

参数名称	描述	输入/输出
hc	http 协议结构体	输入
str	执行参数	输入

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_netctrl.h
- 库文件: libcvi_netctrl.a/libcvi_netctrl.so

【网络 URL】

<http://192.168.1.254/?custom=1&cmd=3029>

【网络 URL 参数】

无

【http 返回报文信息】

```
<?xml version="1.0" encoding="UTF-8" ?>
<List>
<SSID> NVT_CARDV_XXXXXX</SSID>
< PASSPHRASE>12345678</PASSPHRASE>
</List>
```

【Value】

Value	描述
SSID	WIFI 名称
PASSPHRASE	WIFI 密码

【举例】

无

4.5.3.1.48 CVI_XML_GetMovieSizeCapacity

【描述】

查询录像可以支持的分辨率

【语法】

```
int32_t CVI_XML_GetMovieSizeCapacity (httpd_conn *hc, char *str);
```

【参数】

参数名称	描述	输入/输出
hc	http 协议结构体	输入
str	执行参数	输入

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_netctrl.h
- 库文件: libcvi_netctrl.a/libcvi_netctrl.so

【网络 URL】

<http://192.168.1.254/?custom=1&cmd=3030>

【网络 URL 参数】

无

【http 返回报文信息】

```
<?xml version="1.0" encoding="UTF-8" ?>
<LIST>
<Item>
<Name>1920X1080P</Name>
<Index>2</Index>
<Size>1920*1080</Size>
<FrameRate>25</FrameRate>
<Type>4</Type>
</Item>
<Item>
<Name>2560X1440P</Name>
<Index>6</Index>
<Size>2560*1440</Size>
<FrameRate>25</FrameRate>
<Type>4</Type>
</Item>
</LIST>
```

【举例】

无

4.5.3.1.49 CVI_XML_QueryMenuItem**【描述】**

查询支持的菜单设置项

【语法】

```
int32_t CVI_XML_QueryMenuItem (httpd_conn *hc, char *str);
```

【参数】

参数名称	描述	输入/输出
hc	http 协议结构体	输入
str	执行参数	输入

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_netctrl.h
- 库文件: libcvi_netctrl.a/libcvi_netctrl.so

【网络 URL】

<http://192.168.1.254/?custom=1&cmd=3031>

【网络 URL 参数】

无

【http 返回报文信息】

```
<?xml version="1.0" encoding="UTF-8" ?>
<LIST>
<Item>
<Cmd>1002</Cmd>
<Name>PHOTO_SIZE</Name>
<MenuList>
<Option>
<Index>0</Index>
<Id>12M</Id>
</Option>
<Option>
<Index>1</Index>
```

(下页继续)

(续上页)

```
<Id>10M</Id>
</Option>
</MenuList>
</Item>
<CHK>A6B7</CHK>
</LIST>
```

【举例】

无

4.5.3.1.50 CVI_XML_QueryCurrentDVRmode

【描述】

获取当前模式

【语法】

```
int32_t CVI_XML_QueryCurrentDVRmode (httpd_conn *hc, char *str);
```

【参数】

参数名称	描述	输入/输出
hc	http 协议结构体	输入
str	执行参数	输入

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_netctrl.h
- 库文件: libcvi_netctrl.a/libcvi_netctrl.so

【网络 URL】

http://192.168.1.254/?custom=1&cmd=3037

【网络 URL 参数】

无

【http 返回报文信息】

```
<?xml version="1.0" encoding="UTF-8" ?>
<Function>
<Cmd>3036</Cmd>
<Status>0</Status>
</Function>
```

【Status】

Status	描述
0	录像模式
1	拍照模式
2	回放模式

【举例】

无

4.5.3.1.51 CVI_XML_GetThumbnail

【描述】

获取缩略图

【语法】

```
int32_t CVI_XML_GetThumbnail (httpd_conn *hc, char *str);
```

【参数】

参数名称	描述	输入/输出
hc	http 协议结构体	输入
str	执行参数	输入

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_netctrl.h
- 库文件: libcvi_netctrl.a/libcvi_netctrl.so

【网络 URL】

http://192.168.1.254/mnt/sd/CARDV/MOVIE/1970_01_01_080921_00.MOV?custom=1&cmd=4001

【网络 URL 参数】

无

【http 返回报文信息】

缩略图

【举例】

无

4.5.3.1.52 CVI_XML_DeleteOneFile

【描述】

删除指定路径文件

【语法】

```
int32_t CVI_XML_DeleteOneFile (httpd_conn *hc, char *str);
```

【参数】

参数名称	描述	输入/输出
hc	http 协议结构体	输入
str	执行参数	输入

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_netctrl.h
- 库文件: libcvi_netctrl.a/libcvi_netctrl.so

【网络 URL】

http://192.168.1.254/?custom=1&cmd=4003&str=/mnt/sd/CARDV/PHOTO/2014_03_21_171606_00.JPG

【网络 URL 参数】

无

【http 返回报文信息】

```
<?xml version="1.0" encoding="UTF-8" ?>
<Function>
<Cmd>4003</Cmd>
<Status>0</Status>
</Function>
```

【举例】

无

4.5.3.1.53 CVI_XML_DeleteAll

【描述】

删除所有文件

【语法】

```
int32_t cvi_cvi_xml_Deleteall (httpd_conn *hc, char *str);
```

【参数】

参数名称	描述	输入/输出
hc	http 协议结构体	输入
str	执行参数	输入

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_netctrl.h
- 库文件: libcvi_netctrl.a/libcvi_netctrl.so

【网络 URL】

<http://192.168.1.254/?custom=1&cmd=4004>

【网络 URL 参数】

无

【http 返回报文信息】

```
<?xml version="1.0" encoding="UTF-8" ?>
<Function>
<Cmd>4004</Cmd>
<Status>0</Status>
</Function>
```

【举例】

无

4.5.3.154 CVI_XML_GetMovieFileInformation

【描述】

获取指定文件的宽、高和视场

【语法】

```
int32_t CVI_XML_GetMovieFileInformation (httpd_conn *hc, char *str);
```

【参数】

参数名称	描述	输入/输出
hc	http 协议结构体	输入
str	执行参数	输入

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_netctrl.h
- 库文件: libcvi_netctrl.a/libcvi_netctrl.so

【网络 URL】

<http://192.168.1.254/?custom=1&cmd=4005>

【网络 URL 参数】

无

【http 返回报文信息】

```
<?xml version="1.0" encoding="UTF-8" ?>
<Function>
<Cmd>4005</Cmd>
<Status>0</Status>

<String>Width:1920, Height:1080, Length:51</String>
</Function>
```

【举例】

无

4.5.3.1.55 CVI_XML_RoadCamStart

【描述】

RoadCam APP 握手成功响应

【语法】

```
int32_t CVI_XML_RoadCamStart (httpd_conn *hc, char *str);
```

【参数】

参数名称	描述	输入/输出
hc	http 协议结构体	输入
str	执行参数	输入

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_netctrl.h
- 库文件: libcvi_netctrl.a/libcvi_netctrl.so

【网络 URL】

<http://192.168.1.254/?custom=1&cmd=8567>

【网络 URL 参数】

无

【http 返回报文信息】

```
<?xml version="1.0" encoding="UTF-8" ?>
<Function>
<Cmd>8567</Cmd>
<Status>0</Status>

</Function>
```

【举例】

无

4.5.3.1.56 CVI_XML_PhoneApp

【描述】

6 帧探 APP 握手成功响应

【语法】

```
int32_t CVI_XML_PhoneApp (httpd_conn *hc, char *str);
```

【参数】

参数名称	描述	输入/输出
hc	http 协议结构体	输入
str	执行参数	输入

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_netctrl.h
- 库文件: libcvi_netctrl.a/libcvi_netctrl.so

【网络 URL】

<http://192.168.1.254/?custom=1&cmd=9090>

【网络 URL 参数】

无

【http 返回报文信息】

```
<?xml version="1.0" encoding="UTF-8" ?>
<Function>
<Cmd>9090</Cmd>
<Status>0</Status>

</Function>
```

【举例】

无

4.5.3.1.57 CVI_XML_SensorNum

【描述】

获取摄像头的个数

【语法】

```
int32_t CVI_XML_SensorNum (httpd_conn *hc, char *str);
```

【参数】

参数名称	描述	输入/输出
hc	http 协议结构体	输入
str	执行参数	输入

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_netctrl.h
- 库文件: libcvi_netctrl.a/libcvi_netctrl.so

【网络 URL】

<http://192.168.1.254/?custom=1&cmd=9095>

【网络 URL 参数】

无

【http 返回报文信息】

```
<?xml version="1.0" encoding="UTF-8" ?>
<Function>
<Cmd>9095</Cmd>
<Status>0</Status>

<Value>2</Value>
</Function>
```

【举例】

无

4.5.3.1.58 CVI_CGI_SetMovieRecord

【描述】

设置录像分辨率

【语法】

```
int32_t CVI_CGI_SetMovieRecord (httpd_conn *hc, char *str);
```

【参数】

参数名称	描述	输入/输出
hc	http 协议结构体	输入
str	执行参数	输入

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_netctrl.h
- 库文件: libcvi_netctrl.a/libcvi_netctrl.so

【网络 URL】

<http://192.168.1.1/cgi-bin/Config.cgi?action=set&property=Videores&value=1440p25>

【网络 URL 参数】

Value 值
1440p25
1080p25
720p25

【http 返回报文信息】

0
OK

【举例】

无

4.5.3.1.59 CVI_CGI_SetCyclicRecordValue

【描述】

设置录像时长

【语法】

```
int32_t CVI_CGI_SetCyclicRecordValue (httpd_conn *hc, char *str);
```

【参数】

参数名称	描述	输入/输出
hc	http 协议结构体	输入
str	执行参数	输入

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_netctrl.h
- 库文件: libcvi_netctrl.a/libcvi_netctrl.so

【网络 URL】

<http://192.168.1.1/cgi-bin/Config.cgi?action=set&property=LoopingVideo&value=1MIN>

【网络 URL 参数】

Value 值
1MIN
3MIN
5MIN

【http 返回报文信息】

0
OK

【举例】

无

4.5.3.1.60 CVI_CGI_SetGSensorSensitivity

【描述】

设置 GSensor 灵敏度

【语法】

```
int32_t CVI_CGI_SetGSensorSensitivity (httpd_conn *hc, char *str);
```

【参数】

参数名称	描述	输入/输出
hc	http 协议结构体	输入
str	执行参数	输入

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_netctrl.h
- 库文件: libcvi_netctrl.a/libcvi_netctrl.so

【网络 URL】

<http://192.168.1.1/cgi-bin/Config.cgi?action=set&property=GSensor&value=OFF>

【网络 URL 参数】

Value 值
OFF
Low
Middle
High

【http 返回报文信息】

0
OK

【举例】

无

4.5.3.1.61 CVI_CGI_SetVideoTimeLapse

【描述】

设置缩时录影时长

【语法】

```
int32_t CVI_CGI_SetVideoTimeLapse (httpd_conn *hc, char *str);
```

【参数】

参数名称	描述	输入/输出
hc	http 协议结构体	输入
str	执行参数	输入

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_netctrl.h
- 库文件: libcvi_netctrl.a/libcvi_netctrl.so

【网络 URL】

<http://192.168.1.1/cgi-bin/Config.cgi?action=set&property=VideoTimeLapse&value=1fps/s>

【网络 URL 参数】

Value 值
OFF
1fps/s
3fps/s、5fps/s

【http 返回报文信息】

0
OK

【举例】

无

4.5.3.1.62 CVI_CGI_SetMovieEvValue

【描述】

设置曝光补偿数值

【语法】

```
int32_t CVI_CGI_SetMovieEvValue (httpd_conn *hc, char *str);
```

【参数】

参数名称	描述	输入/输出
hc	http 协议结构体	输入
str	执行参数	输入

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_netctrl.h
- 库文件: libcvi_netctrl.a/libcvi_netctrl.so

【网络 URL】

<http://192.168.1.1/cgi-bin/Config.cgi?action=set&property=Exposure&value=EVN200>

【网络 URL 参数】

Value 值
EVP200
EVP167
EVP133
EVP100
EVP67
EVP33
EV0
EVN33
EVN67
EVN100
EVN133
EVN167
EVN200

【http 返回报文信息】

```
0
OK
```

【举例】

无

4.5.3.1.63 CVI_CGI_CommandWouldFormatStorage**【描述】**

格式化 SD 卡

【语法】

```
int32_t CVI_CGI_CommandWouldFormatStorage (httpd_conn *hc, char *str);
```

【参数】

参数名称	描述	输入/输出
hc	http 协议结构体	输入
str	执行参数	输入

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_netctrl.h
- 库文件: libcvi_netctrl.a/libcvi_netctrl.so

【网络 URL】

<http://192.168.1.1/cgi-bin/Config.cgi?action=set&property=SD0value=format>

【网络 URL 参数】

无

【http 返回报文信息】

```
0
OK
```

【举例】

无

4.5.3.1.64 CVI_CGI_DeleteOneFile

【描述】

删除文件

【语法】

```
int32_t CVI_CGI_DeleteOneFile (httpd_conn *hc, char *str);
```

【参数】

参数名称	描述	输入/输出
hc	http 协议结构体	输入
str	执行参数	输入

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_netctrl.h
- 库文件: libcvi_netctrl.a/libcvi_netctrl.so

【网络 URL】

```
http://192.168.1.1/cgi-bin/Config.cgi?action=del&property=\protect\__xunadd_text_character:nN{\textdollar}{\textdollar}SD\protect\__xunadd_text_character:nN{\textdollar}{\textdollar}Normal\protect\__xunadd_text_character:nN{\textdollar}{\textdollar}F\protect\__xunadd_text_character:nN{\textdollar}{\textdollar}DT180718-101134F.MOV
```

【网络 URL 参数】

文件路径

【http 返回报文信息】

```
0
OK
```

【举例】

无

4.5.3.1.65 CVI_CGI_SetGetSSIDandPassPhrase

【描述】

设置 WIFI 账号和密码

【语法】

```
int32_t CVI_CGI_SetGetSSIDandPassPhrase (httpd_conn *hc, char *str);
```

【参数】

参数名称	描述	输入/输出
hc	http 协议结构体	输入
str	执行参数	输入

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_netctrl.h
- 库文件: libcvi_netctrl.a/libcvi_netctrl.so

【网络 URL】

http://192.168.1.1/cgi-bin/Config.cgi?action=set&property=Net.WIFI_AP.SSID&value=(ssid)&property=N

【网络 URL 参数】

参数名称	描述
Net.WIFI_AP.SSID	WIFI 名称
Net.WIFI_AP.CryptoKey	WIFI 密码

【http 返回报文信息】

```
0
OK
```

【举例】

无

4.5.3.1.66 CVI_CGI_SetSSIDandPassPhrase

【描述】

获取 WIFI 名称和密码

【语法】

```
int32_t CVI_CGI_SetSSIDandPassPhrase (httpd_conn *hc, char *str);
```

【参数】

参数名称	描述	输入/输出
hc	http 协议结构体	输入
str	执行参数	输入

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_netctrl.h
- 库文件: libcvi_netctrl.a/libcvi_netctrl.so

【网络 URL】

http://192.168.1.1/cgi-bin/Config.cgi?action=get&property=Net.WIFI_AP.SSID&property=Net.WIFI_AP.CryptoKey

【网络 URL 参数】

无

【http 返回报文信息】

```
0
OK
Net.WIFI_AP.SSID = ssid
Net.WIFI_AP.CryptoKey = pwd
```

【举例】

无

4.5.3.1.67 CVI_CGI_ResetAllSettingTodeFault

【描述】

恢复出厂设置

【语法】

```
int32_t CVI_CGI_ResetAllSettingTodeFault (httpd_conn *hc, char *str);
```

【参数】

参数名称	描述	输入/输出
hc	http 协议结构体	输入
str	执行参数	输入

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_netctrl.h
- 库文件: libcvi_netctrl.a/libcvi_netctrl.so

【网络 URL】

<http://192.168.1.1/cgi-bin/Config.cgi?action=set&property=FactoryReset>

【网络 URL 参数】

无

【http 返回报文信息】

```
0  
OK
```

【举例】

无

4.5.3.1.68 CVI_CGI_GetProjectVersion

【描述】

版本值的获取

【语法】

```
int32_t CVI_CGI_GetProjectVersion (httpd_conn *hc, char *str);
```


【参数】

参数名称	描述	输入/输出
hc	http 协议结构体	输入
str	执行参数	输入

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_netctrl.h
- 库文件: libcvi_netctrl.a/libcvi_netctrl.so

【网络 URL】

<http://192.168.1.1/cgi-bin/Config.cgi?action=get&property=Camera.Menu.FWversion>

【网络 URL 参数】

无

【http 返回报文信息】

```
0
OK
Camera.Menu.Fwversion = CVITEK20210901
```

【举例】

无

4.5.3.1.69 CVI_CGI_GetMenuUUID**【描述】**

获取设备唯一标志

【语法】

```
int32_t CVI_CGI_GetMenuUUID (httpd_conn *hc, char *str);
```

【参数】

参数名称	描述	输入/输出
hc	http 协议结构体	输入
str	执行参数	输入

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_netctrl.h
- 库文件: libcvi_netctrl.a/libcvi_netctrl.so

【网络 URL】

<http://192.168.1.1/cgi-bin/Config.cgi?action=get&property=Camera.Menu.UUID>

【网络 URL 参数】

无

【http 返回报文信息】

```
0
OK
Camera.Menu.UUID = 2C:C3:E6:A7:32:4F
```

【举例】

无

4.5.3.1.70 CVI_CGI_GetMJPEGStatus

【描述】

获取录像状态

【语法】

```
int32_t CVI_CGI_GetMJPEGStatus (httpd_conn *hc, char *str);
```

【参数】

参数名称	描述	输入/输出
hc	http 协议结构体	输入
str	执行参数	输入

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_netctrl.h

- 库文件: libcvi_netctrl.a/libcvi_netctrl.so

【网络 URL】

`http://192.168.1.1/cgi-bin/Config.cgi?action=get&property=Camera.Preview.MJPEG.status.*`

【网络 URL 参数】

无

【http 返回报文信息】

```
0
OK
Camera.Preview.MJPEG.status.record=Standy
```

【Value 值】

Value 值	描述
Camera.Preview.RTSP.av	视频编号
Camera.Preview.Source.1.Camid	前路或后路

【举例】

无

4.5.3.1.71 CVI_CGI_GetPreviewStatus

【描述】

获取设备视频流状态

【语法】

```
int32_t CVI_CGI_GetPreviewStatus (httpd_conn *hc, char *str);
```

【参数】

参数名称	描述	输入/输出
hc	http 协议结构体	输入
str	执行参数	输入

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_netctrl.h
- 库文件: libcvi_netctrl.a/libcvi_netctrl.so

【网络 URL】

`http://192.168.1.1/cgi-bin/Config.cgi?action=get&property=Camera.Preview.*`

【网络 URL 参数】

无

【http 返回报文信息】

```
0
OK

Camera.Preview.RTSP.av=4/5

Camera.Preview.Source.1.Camid=front
```

【Value 值】

Value 值	描述
Standby	停止录像
Recording	正在录像

【举例】

无

4.5.3.1.72 CVI_CGI_GetCamMenu**【描述】**

获取相机目录

【语法】

```
int32_t CVI_CGI_GetCamMenu (httpd_conn *hc, char *str);
```

【参数】

参数名称	描述	输入/输出
hc	http 协议结构体	输入
str	执行参数	输入

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: `cvi_netctrl.h`
- 库文件: `libcvi_netctrl.a/libcvi_netctrl.so`

【网络 URL】

http://192.168.1.1/cgi-bin/Config.cgi?action=get&property=Camera.Menu.*

【网络 URL 参数】

无

【http 返回报文信息】

```
0
OK
Camera.Menu.VideoRes=1480p
Camera.Menu.LoopingVideo=1MIN
Camera.Menu.EV=EVP067
Camera.Menu.GSensor=OFF
Camera.Menu.FWversion=CVITEK20210901
Camera.Menu.SoundRecord=ON
Camera.Menu.VideoTimeLapse=OFF
```

【Value 值】

Value 值	描述
Camera.Menu.VideoRes	录像分辨
Camera.Menu.LoopingVideo	录像时长
Camera.Menu.EV	曝光补偿值
Camera.Menu.GSensor	GSensor 灵敏度
Camera.Menu.FWversion	固件版本
Camera.Menu.SoundRecord	录音开关
Camera.Menu.VideoTimeLapse	缩时录影时长

【举例】

无

4.5.3.1.73 CVI_CGI_SetEdisableMovieAudio

【描述】

设置录音开关

【语法】

```
int32_t CVI_CGI_SetEdisableMovieAudio (httpd_conn *hc, char *str);
```

【参数】

参数名称	描述	输入/输出
hc	http 协议结构体	输入
str	执行参数	输入

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: `cvi_netctrl.h`
- 库文件: `libcvi_netctrl.a/libcvi_netctrl.so`

【网络 URL】

<http://192.168.1.1/cgi-bin/Config.cgi?action=set&property=SoundRecord&value=ON>

【网络 URL 参数】

Value 值	描述
ON	打开
OFF	关闭

【http 返回报文信息】

```
0
OK
```

【举例】

无

4.5.3.1.74 CVI_CGI_GetEdisableMovieAudio**【描述】**

获取录音开关

【语法】

```
int32_t CVI_CGI_GetEdisableMovieAudio (httpd_conn *hc, char *str);
```

【参数】

参数名称	描述	输入/输出
hc	http 协议结构体	输入
str	执行参数	输入

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: `cvi_netctrl.h`
- 库文件: `libcvi_netctrl.a/libcvi_netctrl.so`

【网络 URL】

`http://192.168.1.1/cgi-bin/Config.cgi?action=Get&property=SoundRecord`

【网络 URL 参数】

无

【http 返回报文信息】

```
0
OK

Camera.Menu.SoundRecord = ON
```

【Value 值】

Value 值	描述
ON	打开
OFF	关闭

【举例】

无

4.5.3.1.75 CVI_CGI_Heartbeat**【描述】**

心跳包

【语法】

```
int32_t CVI_CGI_Heartbeat (httpd_conn *hc, char *str);
```

【参数】

参数名称	描述	输入/输出
hc	http 协议结构体	输入
str	执行参数	输入

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: `cvi_netctrl.h`
- 库文件: `libcvi_netctrl.a/libcvi_netctrl.so`

【网络 URL】

`http://192.168.1.1/cgi-bin/Config.cgi?action=Get&property=Heartbeat`

【网络 URL 参数】

无

【http 返回报文信息】

```
0
OK
```

【举例】

无

4.5.3.1.76 CVI_CGI_MovieRecord**【描述】**

更改录像, 包含录像, 停止录像, 拍照

【语法】

```
int32_t CVI_CGI_MovieRecord(httpd_conn *hc, char *str)
```

【参数】

参数名称	描述	输入/输出
hc	http 协议结构体	输入
str	执行参数	输入

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: `cvi_netctrl.h`
- 库文件: `libcvi_netctrl.a/libcvi_netctrl.so`

【网络 URL】

`/cgi-bin/Config.cgi?action=set&property=Video&value=*/`

【网络 URL 参数】

无

【http 返回报文信息】

【举例】

无

4.5.3.1.77 CVI_CGI_CardInfo

【描述】

获取 SD 卡信息

【语法】

```
int32_t CVI_CGI_CardInfo (httpd_conn *hc, char *str);
```

【参数】

参数名称	描述	输入/输出
hc	http 协议结构体	输入
str	执行参数	输入

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_netctrl.h
- 库文件: libcvi_netctrl.a/libcvi_netctrl.so

【网络 URL】

http://192.168.1.1/cgi-bin/Config.cgi?action=get&property=Camera.Menu.CardInfo.*

【网络 URL 参数】

无

【http 返回报文信息】

```
0
OK

Camera.Menu.CardInfo.CardStatus=NORMAL
Camera.Menu.CardInfo.total= 32G
Camera.Menu.CardInfo.remain= 29G
```

【Value 值】

Value 值	描述
Camera.Menu.CardInfo.CardStatus	SD 卡状态
Camera.Menu.CardInfo.total	SD 卡总容量
Camera.Menu.CardInfo.free	SD 卡剩余容量

【举例】

无

4.5.3.1.78 CVI_CGI_VideoesCapability**【描述】**

是否可以录像

【语法】

```
int32_t CVI_CGI_VideoesCapability (httpd_conn *hc, char *str);
```

【参数】

参数名称	描述	输入/输出
hc	http 协议结构体	输入
str	执行参数	输入

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_netctrl.h
- 库文件: libcvi_netctrl.a/libcvi_netctrl.so

【网络 URL】

<http://192.168.1.1/cgi-bin/Config.cgi?action=get&property=Camera.Menu.VideoesCapability>

【网络 URL 参数】

无

【http 返回报文信息】

```
0
OK

Camera.Menu.VideoesCapability=ON
```

【Value 值】

Value 值	描述
ON	可以录像
OFF	停止录像

【举例】

无

4.5.3.1.79 CVI_CGI_ListAllFileCreatTime

【描述】

列出所有文件创建时间

【语法】

```
int32_t CVI_CGI_ListAllFileCreatTime (httpd_conn *hc, char *str);
```

【参数】

参数名称	描述	输入/输出
hc	http 协议结构体	输入
str	执行参数	输入

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_netctrl.h
- 库文件: libcvi_netctrl.a/libcvi_netctrl.so

【网络 URL】

<http://192.168.1.1/cgi-bin/Config.cgi?action=dir&property=Normal&format=all&count=20&from=0>

【网络 URL 参数】

无

【http 返回报文信息】

```
<?xml version="1.0" encoding="UTF-8" ?>
<Normal>
<file>
<name>/mnt/sd/CARDV/MOVIE/2022_01_01_102131_00.MOV</name>
```

(下页继续)

(续上页)

```

<format>MOV</format>
<size>20971520</size>
<attr>RW</attr>
<time>2022-01-01 10:21:34</time>
</file>
<file>
<name>/mnt/sd/CARDV/MOVIE/2022_01_01_102031_00.MOV</name>
<format>MOV</format>
<size>41943040</size>
<attr>RW</attr>
<time>2022-01-01 10:21:31</time>
</file>
<file>
<name>/mnt/sd/CARDV/MOVIE/2022_01_01_091508_00.MOV</name>
<format>MOV</format>
<size>20971520</size>
<attr>RW</attr>
<time>2022-01-01 09:15:09</time>
</file>
<amount>3</amount>
</Normal>

```

【举例】

无

4.5.3.1.80 CVI_CGI_GetScrennail**【描述】**

获取缩略图

【语法】

```
int32_t CVI_CGI_GetScrennail (httpd_conn *hc, char *str);
```

【参数】

参数名称	描述	输入/输出
hc	http 协议结构体	输入
str	执行参数	输入

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_netctrl.h

- 库文件: libcvi_netctrl.a/libcvi_netctrl.so

【网络 URL】

<http://192.168.1.1/thumb/Normal/F/FILE211229-163306F.MOV>

【网络 URL 参数】

无

【http 返回报文信息】

缩略图

【举例】

无

4.5.3.1.81 CVI_CGI_SetSystemTime

【描述】

设置当前时间

【语法】

```
int32_t CVI_CGI_SetSystemTime (httpd_conn *hc, char *str);
```

【参数】

参数名称	描述	输入/输出
hc	http 协议结构体	输入
str	执行参数	输入

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_netctrl.h
- 库文件: libcvi_netctrl.a/libcvi_netctrl.so

【网络 URL】

[http://192.168.1.1/cgi-bin/Config.cgi? action=set&property=TimeSettings&value=2023\\$10\\$17\\$19\\$24\\$01](http://192.168.1.1/cgi-bin/Config.cgi? action=set&property=TimeSettings&value=2023$10$17$19$24$01)

【网络 URL 参数】

无

【http 返回报文信息】

```
0
```

```
OK
```

【举例】

无

4.6 usb 控制

USBCTRL 模块为产品层集成 USB 功能的控制模块，其主要职能如下：

- 从参数模块获取相关参数，完成 USB 初始化；
- 实现 USB 事件回调处理逻辑，并发布 USB 事件，用于与 UI、状态管理交互。

4.6.1 模块 API

该功能模块为用户提供以下 API：

- `CVI_USBCTRL_Init`：USB 控制初始化
- `CVI_USBCTRL_Deinit`：USB 控制反初始化
- `CVI_USBCTRL_RegisterEvent`：USB 状态注册

4.6.1.1 `CVI_USBCTRL_Init`

【描述】

USB 控制初始化

【语法】

```
int32_t CVI_USBCTRL_Init(void);
```

【参数】

无

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件：cvi_usbctrl.h
- 库文件：libcvi_usbctrl.a/libcvi_usbctrl.so

【注意】

无

【举例】

无

【相关主题】

无

4.6.1.2 CVI_USBCTRL_Deinit

【描述】

USB 控制反初始化

【语法】

```
int32_t CVI_USBCTRL_Deinit(void);
```

【参数】

无

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_usbctrl.h
- 库文件: libcvi_usbctrl.a/libcvi_usbctrl.so

【注意】

无

【举例】

无

【相关主题】

无

4.6.1.3 CVI_USBCTRL_RegisterEvent

【描述】

USB 状态注册

【语法】

```
int32_t CVI_USBCTRL_RegisterEvent(void);
```

【参数】

无

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件: cvi_usbctrl.h
- 库文件: libcvi_usbctrl.a/libcvi_usbctrl.so

【注意】

无

【举例】

无

【相关主题】

无

4.7 UI

UI 模块是实现用户与产品进行人机交互的模块，负责响应用户触摸操作、用户按键操作、StateMng 事件、USB 插拔事件、FileMng 事件、SD 卡插拔事件等；UI 模块位置如图 2-1 所示。

4.7.1 模块上下文

UI 模块与各模块的上下关系如图 4-11 所示：

- UI 模块通过 EventHub 模块的 CVI_EventHUB_Subscribe 函数订阅系统事件并注册事件处理回调函数 ui_eventcb。当 UI 订阅的事件发生时，EventHub 模块在回调 ui_eventcb 函数处理事件
- UI 模块向 awtk 模块注册回调函数（OnShow、OnHide），事件发生时 awtk 模块回调相应的函数进行处理。如当界面显示时，awtk 会回调 OnShow 函数并执行 OnShow 函数中的代码逻辑

- UI 模块通过函数接口向 StateMng 模块发送相应调用。

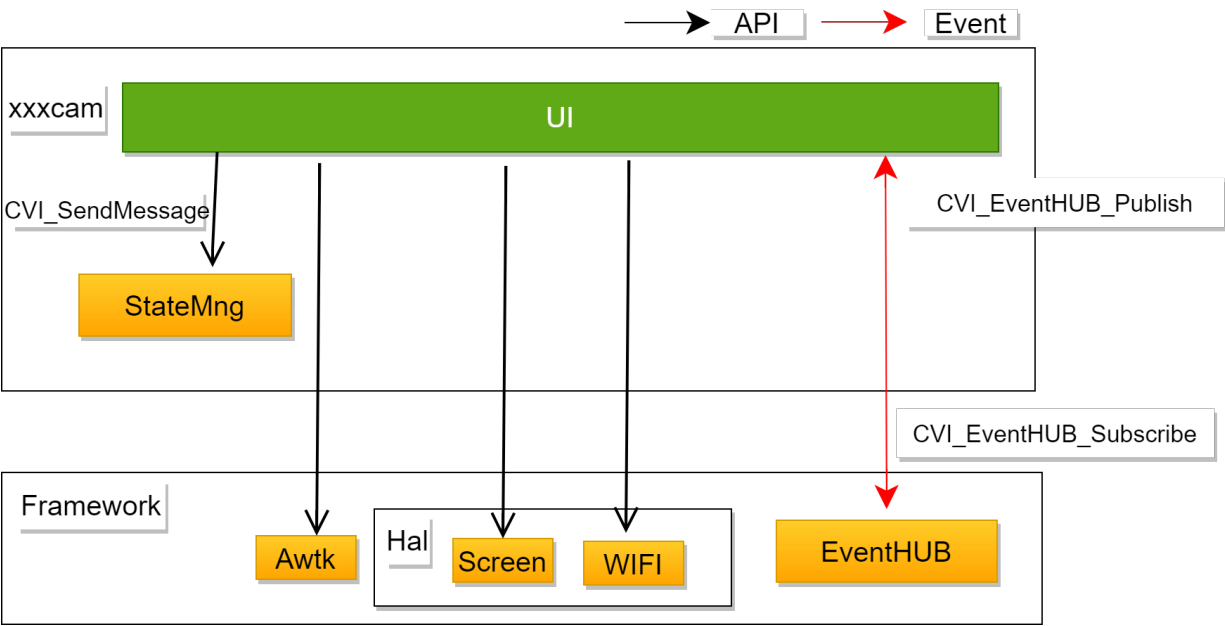


图 4.11: UI 模块上下文

注意:

- awtk 属于第三方库，用于 gui 绘制，请先了解 awtk 的重要概念，如控件，控件管理。

4.7.2 目录结构

```
/ui
├── /carrecorder: 行车记录仪UI
├── /nonscreen: 无屏UI
├── /rearview: 流媒体后视镜UI
├── /lib: 最终使用的库
├── /lib_arm: arm架构生成的库
├── /lib_musl_riscv64: riscv架构生成的库
├── /res
├── /turnkey: turnkey方案
│   ├── /design/default: 源文件
│   │   ├── /fonts: 字库
│   │   ├── /images: 图片
│   │   ├── /styles
│   │   ├── /ui: xml文件
│   │   └── /strings: 中英文
│   ├── /res: 源文件经过awtk工具生成的bin文件工具
│   ├── /include: 头文件
│   └── /src: .c文件
└── Makefile
```

注意:

- src 文件夹存放 ui 各个界面的.c 文件，与不同形态的产品强相关，例如流媒体后视镜，详情参考流媒体后视镜 UI。

4.8 INIT

INIT 模块为产品业务的集成模块，根据产品规格和组件依赖关系构建初始化流程，完成产品业务及服务的启动。INIT 模块在系统中对外的依赖关系如图 4-12 所示：

初始化流程主要包含以下方面：

- 操作系统配置，如双核驱动、共享文件系统初始化；
- 芯片配置，如管脚配置、系统时钟、媒体驱动加载；
- 外设配置，如屏幕配置、外设驱动加载；
- 业务服务启动，如媒体初始化、开机业务。

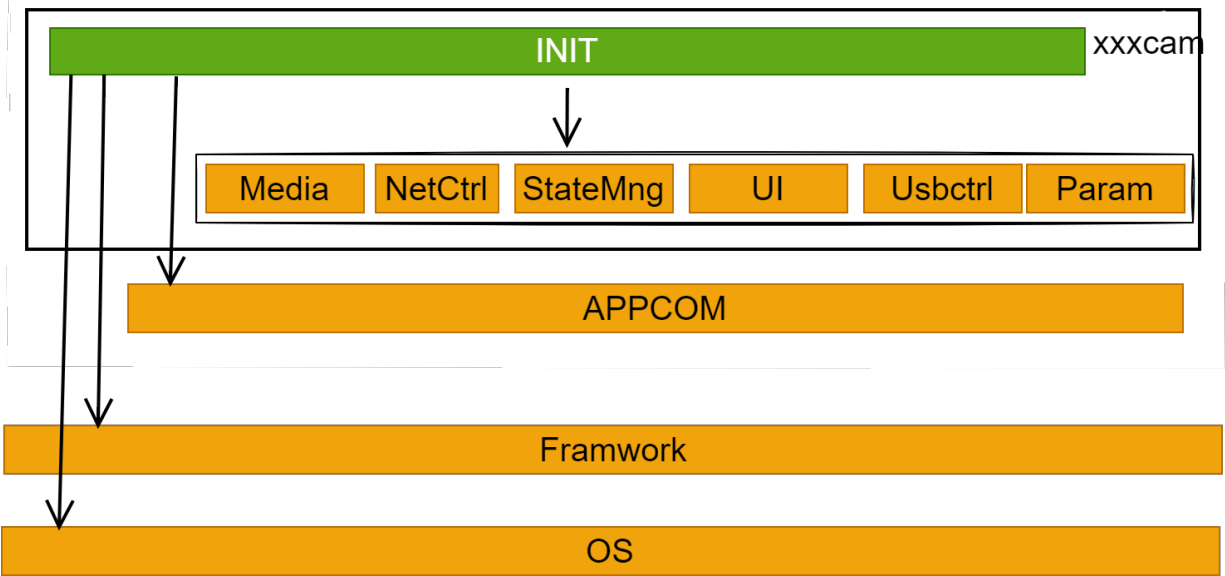


图 4.12: INIT 模块依赖关系框图

4.8.1 目录结构

```
/platform/alios/solutions/cardv/application //Alios 系统
├── app_main.c //Alios系统初始化main函数
├── /media //
└── /common //OS相关初始化
    ↳ 包含芯片相关初始化，外围设备初始化，媒体服务业务初始化
```

(下页继续)

(续上页)

/applications/dashcam/modules/main/src	//Linux 系统
└── cv_i_dashcam_main.c	//Linux端初始化main函数

4.8.2 模块分析

在双系统架构中，初始化业务分为 Alios 端和 Linux 端，其中 Alios 端侧重初始化媒体相关业务，Linux 端侧重初始化服务类业务，比如图 4-12 中 UI、NetCtrl, Media, StateMng, Usbctrl, Param。

4.8.2.1 Alios 端

OS 初始化主要功能如下表所示：

表 4.5: OS 初始化主要功能列表

函数	功能描述
_UartPinmux	uart pinmux
_MipiRxPinmux	mipi rx pinmux
_MipiTxPinmux	mipi tx pinmux
_SensorPinmux	传感器 pinmux
_AudioPinmux	音频 pinmux
_VBatPinmux	电池 pinmux
YOC_SYSTEM_Init	YOC 系统初始化，alios 端使用 yoc 平台
YOC_SYSTEM_ToolInit	cli and ulog init

Media 相关初始化如下表所示：

表 4.6: OS 初始化主要功能列表

函数	功能描述
sys_core_init	sys/base init
cvi_cif_init	cif
vi_core_init	vi
vpss_core_init	vpss
vo_core_init	vo
rgn_core_init	rgn
cvi_ldc_probe	ldc
cvi_ahd_init	ahd
CVI_MSG_Init	双核通信
CVI_Media_PanelInit	屏幕初始化
CVI_Media_Vdec_Logo	开机 Logo

4.8.2.2 Linux 端

Linux 端初始化主要处理与业务功能强相关的初始化操作，如状态管理业务、按键业务、储存管理业务、文件管理业务、UI 等，Linux 端模块初始化如下表所示：

表 4.7: Linux 端模块初始化列表

函数	功能描述
CVI_PARAM_Init	从 flash 获取 ini 参数
CVI_SYSTEM_SetDefaultDateTime	设置系统时间
CVI_MSG_Init	双核通信初始化
CVI_ModuleAoStartThread	音频输出线程
CVI_EVENTHUB_Init	EVENTHUB 初始化
CVI_STORAGEMNG_RegisterEvent	存储管理注册
CVI_FILEMNG_RegisterEvent	文件管理注册
CVI_RECMNG_RegisterEvent	录像管理注册
CVI_POHTOMNG_RegisterEvent	拍照管理注册
CVI_PLAYBACKMNG_RegisterEvent	回放管理注册
CVI_ADASMNG_RegisterEvent	ADAS 管理注册
CVI_MODEMNG_RegisterEvent	模式管理注册
CVI_SYSTEM_GetStartupWakeupSource	设置唤醒源
CVI_MODEMNG_Init	模式管理初始化
CVI_MODEMNG_ContextInit	模式管理配置
CVI_MODEMNG_CreateHFSMInstance	创建 HFSM 实例
CVI_MODEMNG_InitStates	初始并添加所有模式状态，如 base, Movie, Photo, Playback 等
CVI_MODEMNG_SubscribeEvents	状态机向 EVENTHUB 模块订阅事件
CVI_MODEMNG_ActivateHFSM	激活状态机
CVI_MODEMNG_InitStorage	存储服务初始化
CVI_ModuleDelayedStartThread	设备管理初始化，如按键，ADC，LED 等
CVI_NETCTRL_Init	网络控制初始化

5 流媒体后视镜

这里介绍行车产品中的流媒体后视镜：

- 流媒体后视镜（rearview）包含前后双路摄像头，可提供车辆后方实时高清画面，采用触摸屏；
- 流媒体后视镜支持循环录像、紧急录像、拍照、停车监控、VO 预览切换、录像文件与拍照文件回放、参数设置等功能；
- VO 预览默认后置视频，支持单镜头 VO 预览视频范围上下可调。

5.1 参数管理

参数管理提供产品的参数管理与配置功能，参数管理原理与流程请参考[参数管理](#)。

下面介绍 gc4653_bt656_1440p25_128M 文件夹下的 ini 配置，gc4653 代表第一路传感器类型，bt656 代表第二路传感器类型，1440p25 代表前路摄像头分辨率 1440p 和帧率 25，128M 代表 128M DDR。

注意：

- 应用层选择 ini 配置文件夹：
 - 1./cpsl/mmf/Kbuild: 这个文件会根据选择的传感器类型设置一些路径变量
 - 2./build/base.mk: 将上述路径变量组合为文件夹路径 PARAM_FULL_PATH

5.1.1 ini 文件分类

参数主要分成设备管理参数、文件管理参数、UI 菜单参数、工作模式参数、媒体参数，然后通过 config_access_entry.ini 将各类参数的 ini 文件统筹起来管理，如图 5-1 所示：



图 5.1: ini 文件分类

各 ini 文件的功能如下表所示：

表 5.1: ini 文件功能

ini 文件名	功能
config_access_entry.ini	是 ini 解析的入口文件，指定了各 ini 文件的路径与名字
config_devmng.ini	设备管理参数配置文件，包括存储管理、按键管理，wifi 和 pwm
config_filemng.ini	文件管理参数配置文件，包括 DTCF 配置，文件夹名和预分配大小
config_menu.ini	ui 设置菜单，包括录制时长，编码，缩时间隔，音频开关，osd 开关，循环录像开关
config_workmode_usb.ini	usb 参数配置文件，包括 usb storage、usb uvc 的配置参数等
config_media_comm.ini	媒体通用参数，包括 vo，窗口管理，音频参数，通用录像模式参数，rtsp 参数，抓拍，缩略图，通用拍照模式参数，运动检测
config_mediamode_cam0_comm.ini	cam0 媒体通用参数，设置 cam0 开机后默认的工作模式
config_mediamode_cam1_comm.ini	cam1 媒体通用参数，设置 cam1 开机后默认的工作模式
config_workmode_photo.ini	拍照模式参数配置文件，包括 vi_vpss_mode, vpss_mode, cam0 和 1 默认的媒体参数
config_media_cam0_photo_1440p.ini	媒体参数：拍照模式下分辨率 1440P 的 cam0 参数配置文件，包括 vb 池，sensor_cfg.ini, vi, vpss, venc, osd
config_media_cam1_photo_1080p.ini	媒体参数：拍照模式下分辨率 1080P 的 cam1 参数配置文件，包括 vb 池，sensor_cfg.ini, vi, vpss, venc, osd
config_workmode_record.ini	录像模式参数配置文件，包括 vi_vpss_mode, vpss_mode, cam0 和 1 默认的媒体参数
config_media_cam0_record_1080p25.ini	媒体参数：录像模式下分辨率 1080P，帧率 25 的 cam0 参数配置文件，包括 vb 池，sensor_cfg.ini, vi, vpss, venc, osd
config_media_cam0_record_1440p25.ini	媒体参数：录像模式下分辨率 1440P，帧率 25 的 cam0 参数配置文件，包括 vb 池，sensor_cfg.ini, vi, vpss, venc, osd
config_media_cam1_record_720p30.ini	媒体参数：录像模式下分辨率 720P，帧率 30 的 cam1 参数配置文件，包括 vb 池，sensor_cfg.ini, vi, vpss, venc, osd
config_media_cam1_record_1080p25.ini	媒体参数：录像模式下分辨率 1080P，帧率 25 的 cam1 参数配置文件，包括 vb 池，sensor_cfg.ini, vi, vpss, venc, osd
config_media_cam1_record_1080p30.ini	媒体参数：录像模式下分辨率 1080P，帧率 30 的 cam1 参数配置文件，包括 vb 池，sensor_cfg.ini, vi, vpss, venc, osd

5.1.2 ini 文件解析

下面介绍各类 ini 文件的部分内容。

5.1.2.1 config_devmng.ini

```
; storage config
[storage]
dev_state = ; 0:unplugged 1:connecting 2:connected 3:idle
stg_state = ; 0:unplugged 1:connecting 2:error 3:checking ...
fs_type =
work_mode =
dev_port = "cvi.0" //存储设备端口号，即存储设备对应 MCI 卡槽号
speed_class =
speed_grade =
dev_type =
dev_path = "/dev/mmcblk0p1" //设备路径
mnt_path = "/mnt/sd/" //挂载路径

[keymng.key]
key_cnt = 1 //按键数量
key_type0 = 0 //类型
key_id0 = 1 //
longkey_enable0 = 1 //使能
longkey_time0 = 2000 //长按时间

[keymng.grpkey] //按键组管理
enable = 0
key_idx0 = 0
key_idx1 = 1

; wifi
[wifi_config]
enable = 1 ; 0:disabled 1:enable
mode = 1 ; 0:STA mode 1: AP mode
ssid = "cvi_cardv_ap" ; 32 char max
password = "12345678" ; 32 char max
ssid_hide = 0 ; 0:hide 1:show
channel = 6 //wifi通道数

[pwm_config]
enable = 1 ; 0:disabled 1:enable
group = 2 ; pwmchip0/4/8/12
channel = 0 ; pwmchip0:pwm0~3 pwmchip1:pwm4~7
period = 10000 ; unit:ns 100MHz
duty_cycle = 8000 ; 0 ~ 10000 //占空比

[sensor_config]
dev_num = 2 //设备数目，在程序中与 MAX_DEV_INSTANCES 相对应，这里指的是dev0 和 dev1,
cam_num = 2 //这个指的是传感器数量，在程序中与 MAX_CAMERA_INSTANCES 相对应，
//一般情况下dev 和 cam 一一对应，但若存在扩展接口，则 dev 可以一对多 cam。
```


5.1.2.2 config_filemng.ini

```

; common config
[common]
mnt_path = "/mnt/sd/" //挂载路径

; dtcf config
[dtcf]
root_path      = "CARDV"
pre_alloc_cnt  = "100"; // 每个文件夹的最大文件数量
pre_alloc_unit = "2097152";unit: byte //
→预分配文件大小, 这里是2M, 这个是专门为拍照文件, 拍照数据覆盖预分配文件内容
sharepercent   = 95 //空间大小共享百分比, 假如sd卡容量为100G, 那么文件夹总容量不超过95G
warning        = 4 //
→警告空间百分比, 指的是录制文件所占容量最少百分比, 用于判断当卡里存在许多非录制文件
guaranteed     = 5 //开启循环覆盖时的剩余容量百分比

//这里的例子是给出两路sensor; 如果是一路sensor, 就只能设置一路的文件夹, 否则会引起系统崩溃!!!
[dir_name]
dir_name0 = "EMR" //前路紧急录像
dir_name2 = "MOVIE" //前路正常录像
dir_name6 = "EMR_b" //后路紧急录像
dir_name8 = "MOVIE_b" //后路正常录像
dir_name12 = "PHOTO" //前路拍照
dir_name13 = "PHOTO_b" //后路拍照

[prealloc]
enable = 0 //是否预分配
reserve_memory = 5 ;percentage //保留空间百分比
pre_alloc_cnt_dir_0 = "10"; percentage //前路紧急录像最大所占空间百分比
pre_alloc_unit_dir_0 = "40";unit: MB //每个前路紧急录像文件预分配大小
pre_alloc_cnt_dir_2 = "40"; percentage //前路正常录像最大所占空间百分比
pre_alloc_unit_dir_2 = "80";unit: MB //每个前路正常录像文件预分配大小
pre_alloc_cnt_dir_6 = "10";percentage //后路紧急录像最大所占空间百分比
pre_alloc_unit_dir_6 = "40";unit: MB //每个后路紧急录像文件预分配大小
pre_alloc_cnt_dir_8 = "30";percentage //后路正常录像最大所占空间百分比
pre_alloc_unit_dir_8 = "60";unit: MB //每个后路正常录像文件预分配大小
pre_alloc_cnt_dir_12 = "5";percentage //前路拍照最大所占空间百分比
pre_alloc_unit_dir_12 = "2";unit: MB //每个前路拍照文件预分配大小
pre_alloc_cnt_dir_13 = "5";percentage
pre_alloc_unit_dir_13 = "2";unit: MB

```

5.1.2.3 config_menu.in

```

[video_size] //前路录像分辨率选项
num = 2 //选项个数
current = 0 //默认值, 即前路录像分辨率为1440p, 帧率25
description0 = "CVI_MEDIA_VIDEO_SIZE_2560X1440P25" //分辨率1440p, 帧率25
value0 = 0 //代表值
description1 = "CVI_MEDIA_VIDEO_SIZE_1920X1080P25" //分辨率1080p, 帧率25
value1 = 1 //代表值

[video_loop] //录像时长

```

(下页继续)

(续上页)

```
num    = 3 //选项个数
current = 0 //默认值
description0 = "1MIN" //时长一分钟
value0      = 0
description1 = "3MIN"
value1      = 1
description2 = "5MIN"
value2      = 2

[video_codec] //视频编码
num    = 2
current = 0
description0 = "H264"
value0      = 0
description1 = "H265"
value1      = 1

[lapse_time] //缩时时长
num    = 4
current = 0
description0 = "OFF" //关
value0      = 0
description1 = "1S" //1S, 即1S只取一帧
value1      = 1
description2 = "2S"
value2      = 2
description3 = "3S"
value3      = 3

[audio_enable] //音频使能
num    = 2
current = 1
description0 = "OFF"
value0      = 0
description1 = "ON"
value1      = 1

[osd_enable] //osd 使能
num    = 2
current = 1
description0 = "OFF"
value0      = 0
description1 = "ON"
value1      = 1

[UserData] //
bBootFirst = 0

[rec_loop] //循环录像使能
num    = 2
current = 1
description0 = "OFF"
value0      = 0
description1 = "ON"
value1      = 1
```

5.1.2.4 config_media_comm.ini

```

[work_mode]
poweron_mode = "record" //cardv_app起来后默认时录像模式

; vo config
[vo_config]
vo_cnt = 1 //输出屏幕个数

[vo0]
width = 1920 // 输出分辨率宽
height = 440 // 输出分辨率高
fps = -1 //采取默认帧率
rotate = 1 ; 0:0 1:90 2:180 3:270 //旋转90度

; window config
[window_config]
window_cnt = 2 //输出屏幕分为两个窗口，分别展示前后路录像

[window0] //窗口0
enable = 1 //使能
used_crop = 1 //裁剪使能
small_win_enable = 1 //小窗口使能
bind_vproc_id = 0 //绑定的vpss group
bind_vproc_chn_id = 1 //绑定的vpss 通道
x = 0 //窗口起始x坐标，这个用于屏幕只显示一个窗口
y = 0 //窗口起始y坐标
width = 1920 //窗口宽度
height = 440 //窗口高度
s_x = 0 //小窗口起始x坐标,, 这个用于屏幕显示两个窗口，这个位于左侧
s_y = 0 //小窗口起始y坐标
s_width = 960 //小窗口宽度
s_height = 440 //小窗口高度
onestep = 20 //每步长度
ystep = 0
mirror = 0 //镜面翻转使能
filp = 0 //翻转使能

[window1]
enable = 1
used_crop = 1
small_win_enable = 1
bind_vproc_id = 1 //绑定的vpss group
bind_vproc_chn_id = 1 //绑定的vpss 通道
x = 0
y = 0
width = 1920
height = 440
s_x = 960 //小窗口起始x坐标，这个用于屏幕显示两个窗口，这个位于右侧
s_y = 0
s_width = 960
s_height = 440
onestep = 20
ystep = 0
mirror = 0

```

(下页继续)

(续上页)

```

filp      = 0

; Ai config
[ai] //音频输入
sample_rate = 16000 //采样率
channel     = 2 //通道数目
num_per_frm = 640 //每帧采样数
audio_channel = 1 //单声道
isVquOn = 0 //vqe使能
volume = 20 //音量

; Aenc config
[aenc] //音频编码
format     = 3 ; 0: aac, 3: disable //3 默认是pcm
sample_rate = 16000
num_per_frm = 1024
channel     = 2

; Ao config
[ao] //音频输出
sample_rate = 16000
channel     = 2
num_per_frm = 320
audio_channel = 1
volume = 32

; record
[record_config] //录像配置
rec_cnt = 2 //两路录像，即前后路录像

[record0] //前路录像配置
enable      = 1 //使能
//绑定的venc id，与文件config_media_cam0_record_xxx.ini 中的 venc 通道 id = 0 相对应。
bind_venc_id = 0
audio_status = 1 //录像时可同时录制音频使能
file_type    = 1 ; // 0:MP4, 1:MOV, 2:TS, 3:ES, 4:NONE //文件格式
split_time   = 60000 ; msec //录制文件时长
pre_time     = 0 ; sec //前录时间，指的是开始录像时缓冲已存的视频帧时长
post_time    = 10 ; sec //后录时间
timelapse_recorder_fps = 25 //缩时录像帧率
timelapse_recorder_gop = 0 //缩时录像间隔，例如1s代表1s原录像视频取一帧，这里代表正常录像
memory_buffer_sec = 0 //预录缓存视频时长
pre_alloc_unit = 20 ; //pre allocate unit MB
short_file_ms = 500 //短文件时长
devmodel = "sophgo"

[record1] //后路录像配置
enable      = 1
//绑定的venc id，与文件config_media_cam1_record_xxx.ini 中的 venc 通道 id = 4 相对应。
bind_venc_id = 4
audio_status = 1
file_type    = 1 ; // 0:MP4, 1:MOV, 2:TS, 3:ES, 4:NONE
split_time   = 60000 ; msec
pre_time     = 0 ; sec

```

(下页继续)

(续上页)

```
post_time      = 10 ; sec
timelapse_recorder_fps = 25
timelapse_recorder_gop = 0
memory_buffer_sec = 0
pre_alloc_unit = 20 ;//pre allocate unit  MB
short_file_ms = 500
devmodel = "sophgo"

; rtsp config
[rtsp_config] //rtsp配置
rtsp_cnt      = 2 //最多两路rtsp连接
port=554      //端口号
timeout=120   //连接超时时长，单位为秒

[rtsp0] //前路
enable        = 1
//绑定的venc id, 与文件config_media_cam0_record_xxx.ini 中的 venc 通道 id = 1 相对应。
bind_venc_id = 1
max_conn      = 2 //最大连接数
name = "cvi_cam_0" //名字

[rtsp1] //后路
enable        = 1
//绑定的venc id, 与文件config_media_cam1_record_xxx.ini 中的 venc 通道 id = 5 相对应。
bind_venc_id = 5
max_conn      = 2
name = "cvi_cam_1"

; PIV config
[piv_config] //抓拍配置
piv_cnt       = 2 //抓拍通道数

[piv0] //前路
//绑定的venc id, 与文件config_media_cam0_record_xxx.ini中的 venc 通道 id = 2 相对应。
bind_venc_id  = 2

[piv1]
//绑定的venc id, 与文件config_media_cam1_record_xxx.ini 中的 venc 通道 id = 6 相对应。
bind_venc_id  = 6

; thm config
[thm_config] //缩略图
thm_cnt       = 2

[thm0]
bind_venc_id  = 3 //绑定的venc id, 与文件config_media_cam0_record_xxx.ini和 config_media_
↪cam0_photo_1440p.ini中的 venc 通道 id = 3 相对应。

[thm1]
bind_venc_id  = 7 //绑定的venc id, 与文件config_media_cam1_record_xxx.ini 和 config_media_
↪cam1_photo_1080p.ini中的 venc 通道 id = 7 相对应。

[photo_config] //拍照模式拍照
```

(下页继续)

(续上页)

```
photo_cnt    = 2
vprocdev_id  = 1 //拍照的设备号为dev1

[photo0]
enable       = 1
//绑定的venc id, 与config_media_cam0_photo_1440p.ini中的 venc 通道 id = 2 相对应。
bind_venc_id = 2
[photo1]
enable       = 1
//绑定的venc id, 与文件 config_media_cam1_photo_1080p.ini中的 venc 通道 id = 6 相对应。
bind_venc_id = 6

[motionDet_config] //运动检测
md_cnt = 2
motionSensitivity = 50 //灵敏度

[md0]
enable = 1
bind_vproc_id = 0 //vpss group 0
bind_vproc_chn_id = 4
//vpss 通道 4, 与config_media_cam0_photo_1440p.ini中的 vpss chn_id = 4 相对应。
[md1]
enable = 1
bind_vproc_id = 1 //vpss group 1
bind_vproc_chn_id = 4 //vpss 通道 4

[adas_config]//adas参数配置
adas_cnt = 1 //数量, 目前仅支持前路
model_fps = 12.5 //模型输入帧率
model_vb_width = 640 //模型内部创建的vb池宽度
model_vb_height = 384 //模型内部创建的vb池高度
car_model_path = "/mnt/system/vehicle_det.cvimodel" //车辆检测模型路径
lane_model_path = "/mnt/system/lane_det.cvimodel" //车道线检测模型路径

[adas0] //第一个adas
enable = 1 //使能
bind_vproc_id = 0 //连接的vpss id
bind_vproc_chn_id = 2
//连接的vpss chn id, 与config_media_cam0_photo_1440p.ini中的 vpss chn_id = 2相对应。

[qrcode_config] //二维码参数配置
qrcode_cnt = 0 //个数

[qrcode0]
enable = 1 //使能
bind_vproc_id = 0 //连接的vpss id
bind_vproc_chn_id = 5
//连接的vpss chn id, 与config_media_cam0_photo_1440p.ini中的 vpss chn_id = 5 相对应。
```

5.1.2.5 config_mediamode_cam0_comm.ini

```
[camera]
enable      = 1 ; 0:disable, 1:enable
osdshow     = 1 ; 0:disable, 1:enable
cam_id      = 0
cur_mode    = "record_1440p25" //这个对应config_media_cam0_record_1440p25.ini

[mediamode]
count       = 3 //这个对应的是config_media_cam0_xxx_xxx.ini的文件数
```

5.1.2.6 config_mediamode_cam1_comm.ini

```
[camera]
enable      = 1 ; 0:disable, 1:enable
osdshow     = 1 ; 0:disable, 1:enable
cam_id      = 1
cur_mode    = "record_1080p25" //这个对应config_media_cam1_record_1080p25.ini

[mediamode]
count       = 4 //这个对应的是config_media_cam1_xxx_xxx.ini的文件数
```

5.1.2.7 config_workmode_photo.ini

```
[common]
work_mode = "photo" //指定为拍照模式
cam_num = 2 //两路摄像头

[vi_vpss_mode]
pipe0     = 0 ; 0: vi offline vpss offline, 1: vi offline vpss online
pipe1     = 0 ; 0: vi offline vpss offline, 1: vi offline vpss online

[vpss_mode]
mode      = 1 ; 0: single mode, 1: dual Mode
//vpss硬件有两个输入硬件img0和img1（对应下面描述的dev0和1，以及4个输出硬件通道；
//single mode下vpss硬件仅使用img1，并可以使用4个输出硬件通道；dual
→mode下img1对应3个硬件输出通道，img0对应1个硬件输出通道

[vpss_mode.dev0]
input     = 0 ; 0: input mem, 1: input isp
//当为dual mode，dev0的input必须来自memory，也就是vpss offline，这是硬件决定的
pipe      = 0 ;
[vpss_mode.dev1]
input     = 0 ; 0: input mem, 1: input isp
//当为dual mode，dev1的input必须来自isp，也就是vpss online，这是硬件决定的
pipe      = 0 ;

[config0]
cam_id = 0 //第一路
media_mode = "photo_1440p" //指定媒体工作配置，也就是config_media_cam0_photo_1440p.ini
```

(下页继续)

(续上页)

```
[config1]
cam_id = 1 // 第二路
media_mode = "photo_1080p" ////指定媒体工作配置，也就是config_media_cam1_photo_1080p.ini
```

5.1.2.8 config_media_cam0_photo_1440p.ini

```
[common]
media_mode = "photo_1440p" //与文件名相对应

; vb pool config
[vb_config] // 公有视频池，即公有池，共享，各个视频帧都可以使用
vbpool_cnt = 4 //视频池个数

[vbpool0]
enable = 1
frame_width = 2560
frame_height = 1440
frame_fmt = 19 ; 12: yuv422 13: yuv420 19:NV21
blk_cnt = 2 //内存块个数
//内存块的大小是根据视频帧大小分配，一般为 (frame_width 64位对齐) x frame_height x 1.5

[vbpool1]
enable = 1
frame_width = 440
frame_height = 1920
frame_fmt = 19 ; 12: yuv422 13: yuv420 19:NV21
blk_cnt = 3

[vbpool2]
enable = 1
frame_width = 320
frame_height = 180
frame_fmt = 19 ; 12: yuv422 13: yuv420 19:NV21
blk_cnt = 3

[vbpool3]
enable = 1
frame_width = 1920
frame_height = 1080
frame_fmt = 19 ; 12: yuv422 13: yuv420 19:NV21
blk_cnt = 2

; sensor
[sensor_config] //传感器配置，原厂配置
enable = 1
id = 0
wdrmode = 0 ; 0:linear, 3:wdr 2to1
i2cid = 3 //i2c id
i2caddr = 41 ; // i2c 地址
cam_clk_id = 0; 0:clk0, 1:clk1
rst_gpio_inx = 0; 0: group A, 1: group B ...
rst_gpio_pin = 2; 0: PIN 0, 1: PIN 1, 2: PIN 2....
```

(下页继续)

(续上页)

```

rst_gpio_pol = 1; 1: OF_GPIO_ACTIVE_LOW, 2: OF_GPIO_SINGLE_ENDED, 4: OF_GPIO_
↪OPEN_DRAIN
           ; 8: OF_GPIO_TRANSITORY, 16: OF_GPIO_PULL_UP, 32: OF_GPIO_PULL_
↪DOWN
hwsync = 0
mipidev = 0
laneid0 = 2
laneid1 = 4
laneid2 = 1
laneid3 = -1
laneid4 = -1 //lane 序
swap0 = 0
swap1 = 0
swap2 = 0
swap3 = 0
swap4 = 0 //pn_swap

; vcap
[vcap_config] //视频输入配置
enable = 1
id = 0
width = 2560 //输入宽
height = 1440 //高
fmt = 19 ; //12 yuv422 //13 yuv420; //19 nv21;
cpress = 1 ;NONE : 0 TILE : 1,LINE:2, FRAME:3
mirror = 0 //镜面翻转
filp = 0
fps = 25 ; // vi fps setting, support float
enfbm = 1 //fbm (frame buffer mode) 使能

; vproc
[vproc_config] //vpss配置
id = 0 ;//online mode this id must 0
width = 2560 //vpss 输入宽
height = 1440 //vpss 输入高
fmt = 19 ; //12 yuv422 //13 yuv420;
srcfps = -1 //与输入源的帧率保持一致
dstfps = -1 //与srcfps保持一致
chn_num = 3 //输出通道3个

[vproc_chn0]
chn_id = 0
chn_enable = 1
chn_width = 1920 //输出宽
chn_height = 1080
chn_vfmt = 0 ;// 0 linear
chn_fmt = 19; //12 yuv422 //13 yuv420 //19 nv21;
chn_depth = 0
chn_srcfps = -1
chn_dstfps = -1
chn_mirror = 0
chn_filp = 0
; lowdelay_cnt= 288

```

(下页继续)

(续上页)

```
[vproc_chn1]
chn_id      = 1
chn_enable  = 1
chn_width   = 1920
chn_height  = 440
chn_vfmt    = 0 ;// 0 linear
chn_fmt     = 19; //12 yuv422 //13 yuv420;
chn_depth   = 1
chn_srcfps  = -1
chn_dstfps  = -1
chn_mirror  = 0
chn_filp    = 0
chn_vbcnt   = 3 //
```

→这个指的是私有池的个数，会创建独属于这个通道的视频池，，这个通道只能使用这个视频池

```
[vproc_chn2]
chn_id      = 2
chn_enable  = 1
chn_width   = 320
chn_height  = 180
chn_vfmt    = 0 ;// 0 linear
chn_fmt     = 19; //12 yuv422 //13 yuv420;
chn_depth   = 0
chn_srcfps  = -1
chn_dstfps  = -1
chn_mirror  = 0
chn_filp    = 0
```

//这个通道用于vpss single mode,此时vpss硬件实现“一出四”(一个输入，4个输出通道)，
//config_workmode_photo.ini文件设置vpss为dual mode,所以这个通道不使能

```
[vproc_chn3]
chn_id      = 3
chn_enable  = 0
chn_width   = 320
chn_height  = 180
chn_vfmt    = 0 ;// 0 linear
chn_fmt     = 19; //12 yuv422 //13 yuv420;
chn_depth   = 0
chn_srcfps  = -1
chn_dstfps  = -1
chn_mirror  = 0
chn_filp    = 0
```

//ext_chn only in dual mode, 专门用于dev0, 由于此时是“一出一”(一个输入，1个输出通道)，
//所以每使能一个ext_chn, 创建一个vpss grp, 然后分时复用dev0;而其他grp分时复用dev1

```
[ext_chn0];
chn_enable  = 0
chn_id      = 3
bind_vproc_chn_id = 2
chn_width   = 320
chn_height  = 180
chn_vfmt    = 0 ;// 0 linear
chn_fmt     = 19; //12 yuv422 //13 yuv420;
chn_depth   = 0
```

(下页继续)

(续上页)

```

chn_srcfps = -1
chn_dstfps = -1
chn_mirror = 0
chn_filp   = 0

[ext_chn1] ;//ext chn only in dual mode
chn_enable = 1
chn_id     = 4
bind_vproc_chn_id = 3
chn_width  = 320
chn_height = 180
chn_vfmt   = 0 ;// 0 linear
chn_fmt    = 15; //12 yuv422 //13 yuv420;
chn_depth  = 0
chn_srcfps = -1
chn_dstfps = -1
chn_mirror = 0
chn_filp   = 0

[ext_chn2] ;//ext chn only in dual mode
chn_enable = 0
chn_id     = 5
bind_vproc_chn_id = 0
chn_width  = 640
chn_height = 360
chn_vfmt   = 0 ;// 0 linear
chn_fmt    = 13; //12 yuv422 //13 yuv420;
chn_depth  = 0
chn_srcfps = -1
chn_dstfps = -1
chn_mirror = 0
chn_filp   = 0

; venc
[venc_config] //编码配置
chn_num = 2

[venc_chn0]
enable    = 1
id        = 2
bind_vproc_id    = 0 //绑定的vpss
bind_vproc_chn_id = 0 //绑定的vpss 通道
width     = 640 //编码输出宽
height    = 480
codec     = 2 ; 0:h264 1:h265 2:jpeg 3:mjpeg
pixel_fmt = 19; //12 yuv422 //13 yuv420;
single_EsBuf= 1 //使能，这个使多个编码通道使用一个缓存，从而节省内存
jpeg_quality = 80 //MJPEG 编码的 Qfactor
bufSize     = 4194304; 4x1024x1024 //缓存大小，1024对齐,至少为width X height X 1/4

[venc_chn1]
enable    = 1
id        = 3
bind_vproc_id    = 0

```

(下页继续)

(续上页)

```

bind_vproc_chn_id = 2
width      = 320
height     = 180
codec      = 2 ; 0:h264 1:h265 2:jpeg 3:mjpeg
pixel_fmt  = 13; //12 yuv422 //13 yuv420;
single_EsBuf= 1
jpeg_quality = 80
bufSize    = 65536

[venc_chn2]
enable     = 0
id         = 1
bind_vproc_id      = 0
bind_vproc_chn_id = 1
; width      = 640
; height     = 360
width       = 1280
height      = 720
codec       = 0 ; 0:h264 1:h265 2:jpeg 3:mjpeg
pixel_fmt   = 19; //12 yuv422 //13 yuv420;
gop         = 30 //Group of Pictures。顾名思义，就是一组图片，30帧
fps         = 30; // support float
rate_ctrl_mode = 2 ; 0: CBR 1:VBR 2:AVBR 3:QVBR 4:FIXQP 5:QPMAP //码率控制方式
bit_rate    = 1500 ;kbps //码率
iqp         = 32 //I 帧所有宏块 Qp 值
ppq         = 32 //P 帧所有宏块 Qp 值
min_iqp     = 18 ;//I帧最小QP
max_iqp     = 44 ;//I帧最大QP
min_qp      = 20 ;//最小QP
max_qp      = 50 ;//最大QP
; bufSize    = 262144
bufSize     = 1024000
initialdelay = 1000; //影响帧编码的码率控制
thrdlv      = 2 ;//宏块级码率控制的mad门限
stattime    = 2 ;//CBR 码率统计时间，以秒为单位
firstFrameStartQp = 36 ;//设置第一帧的起始 Qp 值， CBR / VBR / AVBR 有效
maxBitRate  = 1500 ;//编码器输出最大码率，以 kbps 为单位
gop_mode    = 0
maxIprop    = 100 ;//最大IP比
minIprop    = 1 ;//最小IP比
minStillPercent = 60; //静态场景最小码率百分比
maxStillQP  = 46 ;//静态场景最大QP
avbrPureStillThr = 50 ;//静止门限值
motionSensitivity = 24; //运动灵敏度
bgDeltaQp   = 0 ;//smartP模式下，IDR帧与P帧的QP差值
rowQpDelta  = 0 ;//在宏块级码率控制时，每一行宏块的起始 Qp 相对于帧起始 Qp 的波动幅度值
change_pos  = 90; //码率调节阈值
ipqpDelta   = 3 ;//I帧和P帧的QP差值
videoSignalType_Flag = 1 ;///* RW; Range:[0,1]; If 1, video singnal info will be encoded into vui. */
video_Format = 5 ;///* RW; H.264e Range:[0,7], H.265e Range:[0,5]; see the protocol for the meaning.
↪*/
videoFull_Flag = 1 ;///* RW; Range: {0,1}; see the protocol for the meaning.*/

; osd config

```

(下页继续)

(续上页)

```
[osd_config] //osd配置
osd_cnt    = 1

[osd_content0]
type       = 0 ;//0: time 1:customeer string 2:bmp 3:circle //0代表时间戳
color      = 65535 ;//argb1555 0xffff
time_fmt   = 0 ;//yyyy-mm-dd
width      = 24 //宽
height     = 30
bg_color   = 32767 ; argb1555 0x7fff
display_num = 1 //展示窗口数
```

5.1.3 Video pipeline

本小章节是为了梳理不同模式下媒体参数配置的 video pipeline。

拍照模式 video pipeline 如图 5-2 所示：

- 使用的参数主要来自 config_media_cam0_photo_1440p.ini 和 config_media_cam1_photo_1080p.ini。

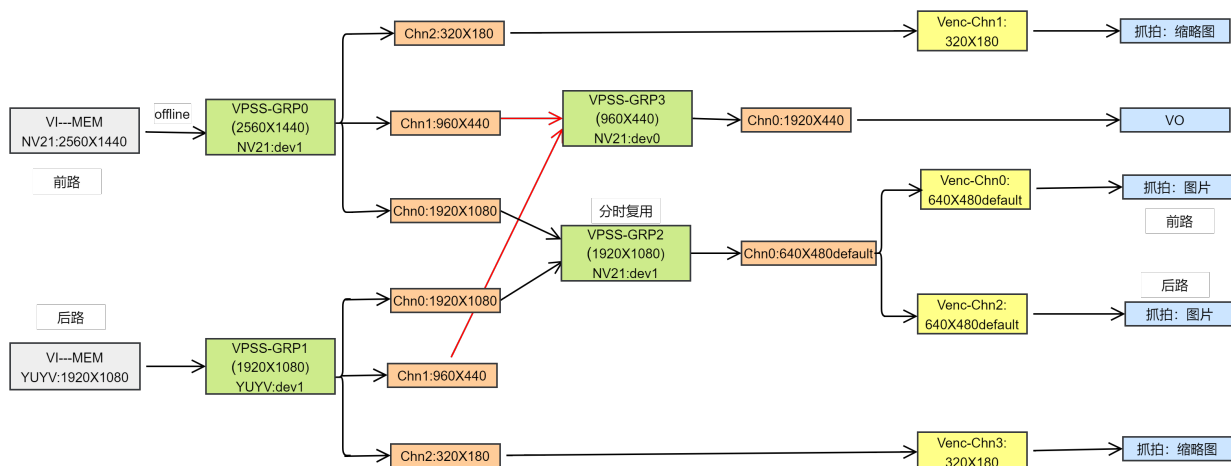


图 5.2: 拍照模式 video pipeline

5.2 状态管理

状态管理 (State Manager, 简写 StateMng) 承载与管理各个业务状态，原理说明请参考状态管理。

5.2.1 业务状态

流媒体后视镜包含 5 个业务状态，如下表所示：

表 5.2: 流媒体后视镜业务状态

业务状态	业务模式枚举
普通录像	CVI_WORK_MODE_MOVIE
缩时录像	CVI_WORK_MODE_LAPSE
拍照模式	CVI_WORK_MODE_PHOTO
回放模式	CVI_WORK_MODE_PLAYBACK
升级模式	CVI_WORK_MODE_UPDATE

普通录像 (Movie)：进行普通的循环录像，是最常用的模式；正常开机即会进入循环录像模式，此模式下可以进行 VO 预览、拍照、启动紧急录像、打开 setting。

缩时录像 (Lapse)：打开 setting，选择缩时间隔即可开启，功能选项与普通录像大致相同，但不能开启紧急录像。

拍照模式 (Photo)：通过触摸操作，可以进入拍照模式，打开 setting 可选择不同的拍照分辨率，最高可达 12M。

回放模式 (Playback)：通过触摸操作，可以进入回放模式，退出回放，则进入循环录像模式。

升级模式 (Update)：OTA 网络升级，该模式目前能通过板端命令 `cc_tools` 或者 pc 端连接产品 WiFi 进行（参考 API: `CVI_XML_StartToupdateFirmware`）。

层次状态机表述如图 5-3 所示。

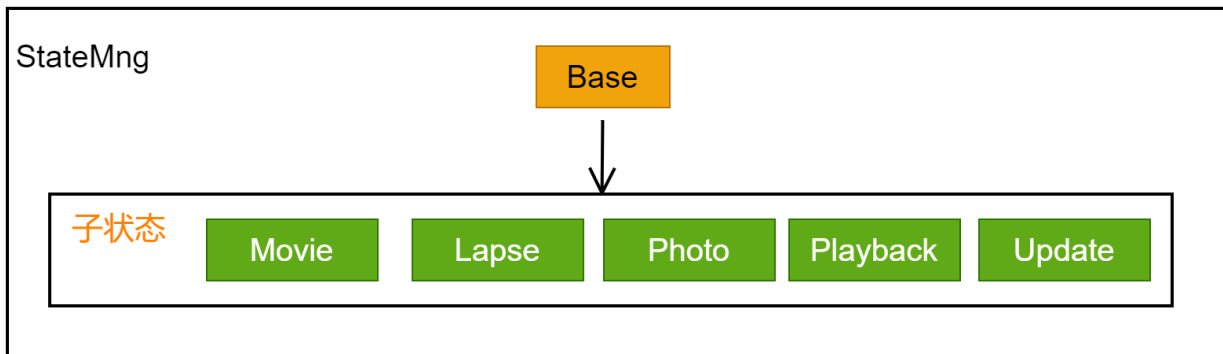


图 5.3: Rearview 层次状态机

注意：

- 这 5 个状态不是同时激活，而是通过 Base 处理消息 `Switch_Workmode` 进行模式切换。

5.2.2 Base 状态处理消息

base 状态处理的消息，如下表所示：

表 5.3: 流媒体后视镜 base 状态处理消息

消息名	描述
CVI_EVENT_STORAGEMNG_DEV_UNPLUGED	SD 卡拔出
CVI_EVENT_STORAGEMNG_DEV_CONNECTING	SD 卡挂载中
CVI_EVENT_STORAGEMNG_DEV_ERROR	SD 卡错误
CVI_EVENT_STORAGEMNG_MOUNT_FAILED	SD 卡挂载失败
CVI_EVENT_STORAGEMNG_MOUNT_READ_ONLY	SD 卡挂载只读
CVI_EVENT_STORAGEMNG_FS_CHECKING	FS 检测中
CVI_EVENT_STORAGEMNG_MOUNTED	SD 卡挂载完成
CVI_EVENT_STORAGEMNG_FS_EXCEPTION	FS 修复异常
CVI_EVENT_MODEMNG_MODESWITCH	工作模式切换
CVI_EVENT_MODEMNG_POWEROFF	抓拍结束
CVI_EVENT_RECMNG_SPLITSTART	录像文件切分
CVI_EVENT_MODEMNG_SCREEN_DORMANT	屏幕休眠
CVI_EVENT_RECMNG_WRITE_ERROR	录像文件写入失败

5.3 UI

这里介绍流媒体后视镜的 UI 模块，原理参考[UI 模块](#)。

5.3.1 目录结构

```
/rearview
├── /include
│   ├── ui_common.h           //各界面函数
│   └── ui_windowmng.h       //窗口管理
├── /src
│   ├── awtk_app_conf.h      //宏
│   ├── ui_busing_page.c     //拍照提示界面
│   ├── ui_common.c         //通用
│   ├── ui_dir_page.c        //文件夹界面
│   ├── ui_filelist_page.c   //文件界面
│   ├── ui_format_page.c     //格式化界面
│   ├── ui_home_page.c       //录像界面
│   ├── ui_home_photo_page.c //拍照界面
│   ├── ui_main.c           //主函数
│   ├── ui_playback_page.c   //回放界面
│   ├── ui_powercontrol.c    //电源控制
│   └── ui_reset_page.c      //重启
```

(下页继续)

(续上页)

```
| |—— ui_settime_page.c      //设置时间
| |—— ui_setting_page.c     //设置界面
| |—— ui_windowmng.c        //窗口管理，即界面管理
| |—— ui_wrnmsg_page        //警告界面
|—— Makefile
```

5.3.2 API 参考

- CVI_UIAPP_Start：启动 UI
- CVI_UIAPP_Stop：停止 UI

5.3.2.1 CVI_UIAPP_Start

【描述】

启动 UI

【语法】

```
int32_t CVI_UIAPP_Start(void);
```

【参数】

无

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件：ui_common.h/ui_windowmng.h
- 库文件：libcar_ui.a/libcar_ui.so

【注意】

无

【举例】

无

【相关主题】

无

5.3.2.2 CVI_UIAPP_Stop

【描述】

停止 UI

【语法】

```
int32_t CVI_UIAPP_Stop(void);
```

【参数】

无

【返回值】

返回值	描述
0	成功
非 0	失败

【需求】

- 头文件：ui_common.h/ui_windowmng.h
- 库文件：libcar_ui.a/libcar_ui.so

【注意】

无

【举例】

无

【相关主题】

无

5.3.3 UI 功能菜单

UI 功能菜单如图 5-4 所示：

- 录像界面意味着进入了普通录像模式，可通过录像设置界面的缩时间隔开启缩时录像模式。
- 拍照界面意味着进入了拍照模式。
- 回放界面意味着进入了回放模式。

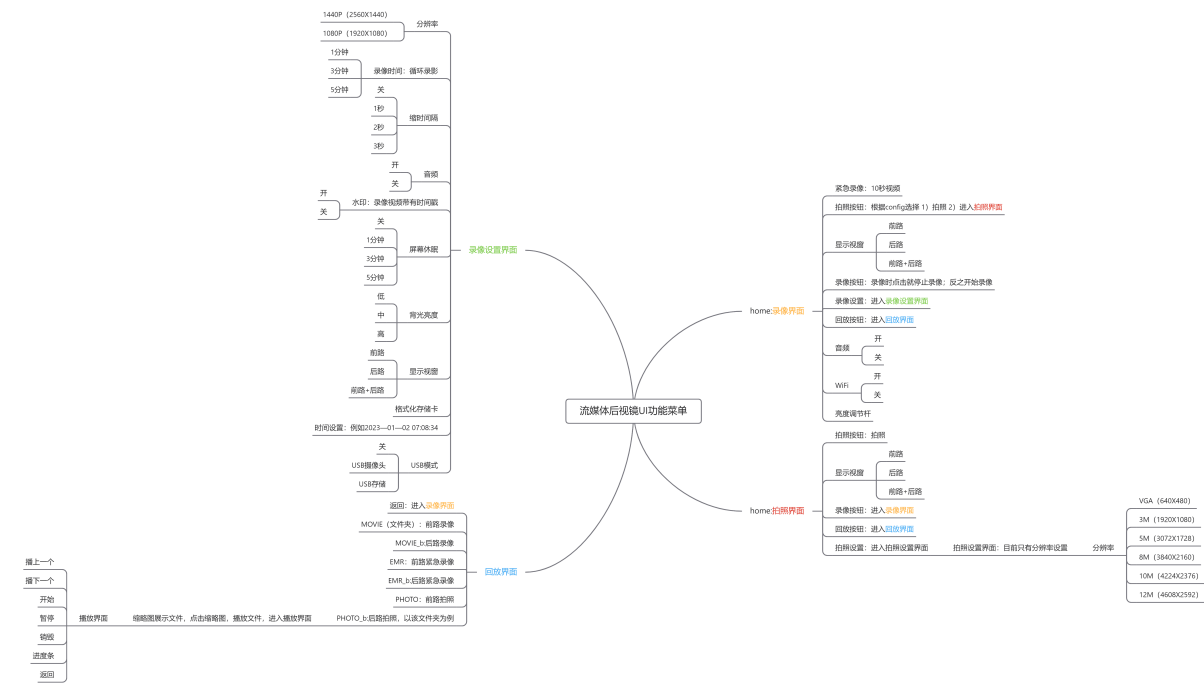


图 5.4: UI 功能菜单

录像界面如图 5-5 所示：

- 1. 紧急录像按钮；2. 拍照按钮；3. 显示视窗；4. 录像按钮；5. 录像设置；6. 回放按钮；7. 音频开关；8.wifi 开关；9. 前路镜头；10. 后路镜头
- 11. 录像提示；12. 亮度调节杆；13. 录像时长 1 分钟提示；14. 前路镜头录像分辨率提示；15.WiFi 提示，此时已开；16. 音频提示，此时已关；17. 电量显示；18. 时间



图 5.5: 录像界面

拍照界面如图 5-6 所示：



图 5.6: 拍照界面

播放界面如图 5-7 所示：

- 1. 播上一个视频；2. 播放；3. 暂停；4. 播下一个视频；5. 删除；6. 返回；7. 进度显示条；8. 视频文件名



图 5.7: 播放界面

录像设置界面如图 5-8 所示：

- 1. 当前设置；2. 进入子界面，修改当前设置；3. 返回

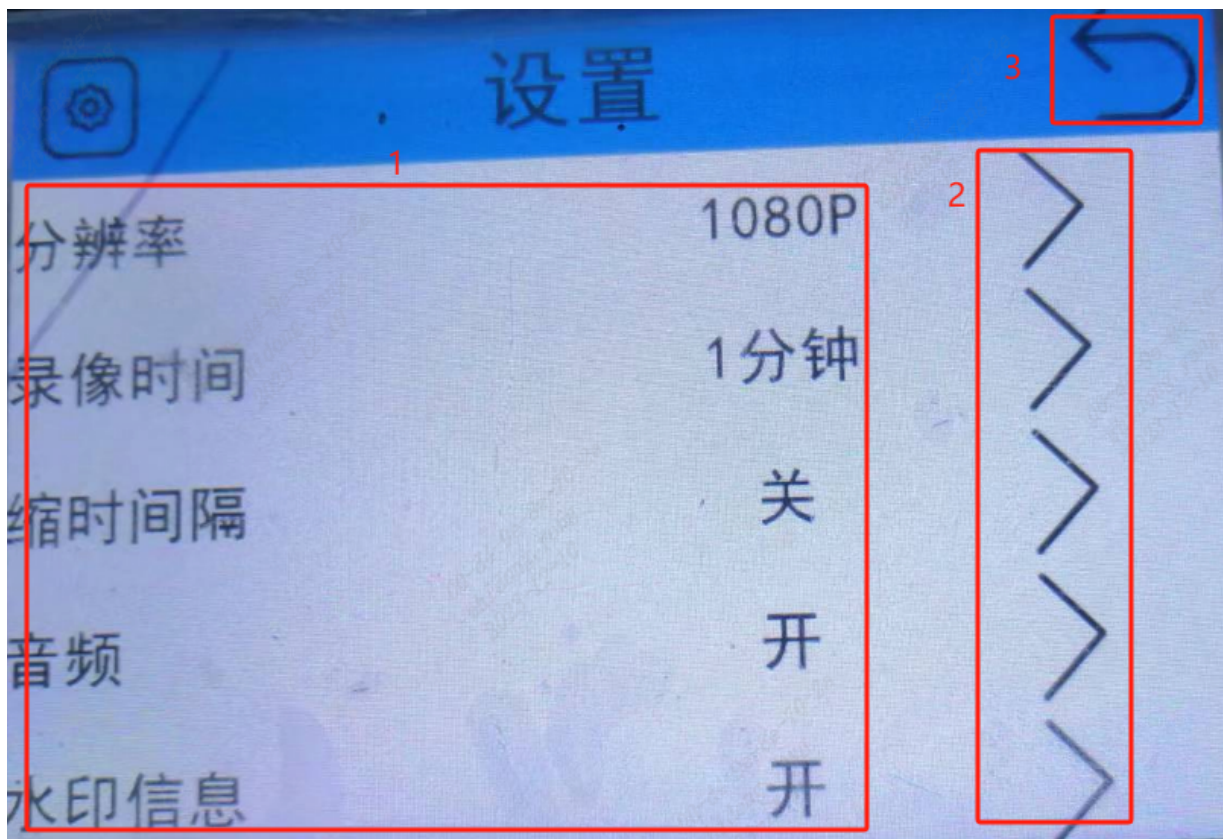


图 5.8: 录像设置界面

5.3.4 UI 事件划分

ui 处理事件的划分参考了状态管理的 base 和子状态，同样分为通用事件和子状态事件，如图 5-9 所示。

- xxx 指的是工作模式
- 事件先经过 ui_common_eventcb 函数处理，再经过 ui_xxx_page_eventcb 处理。

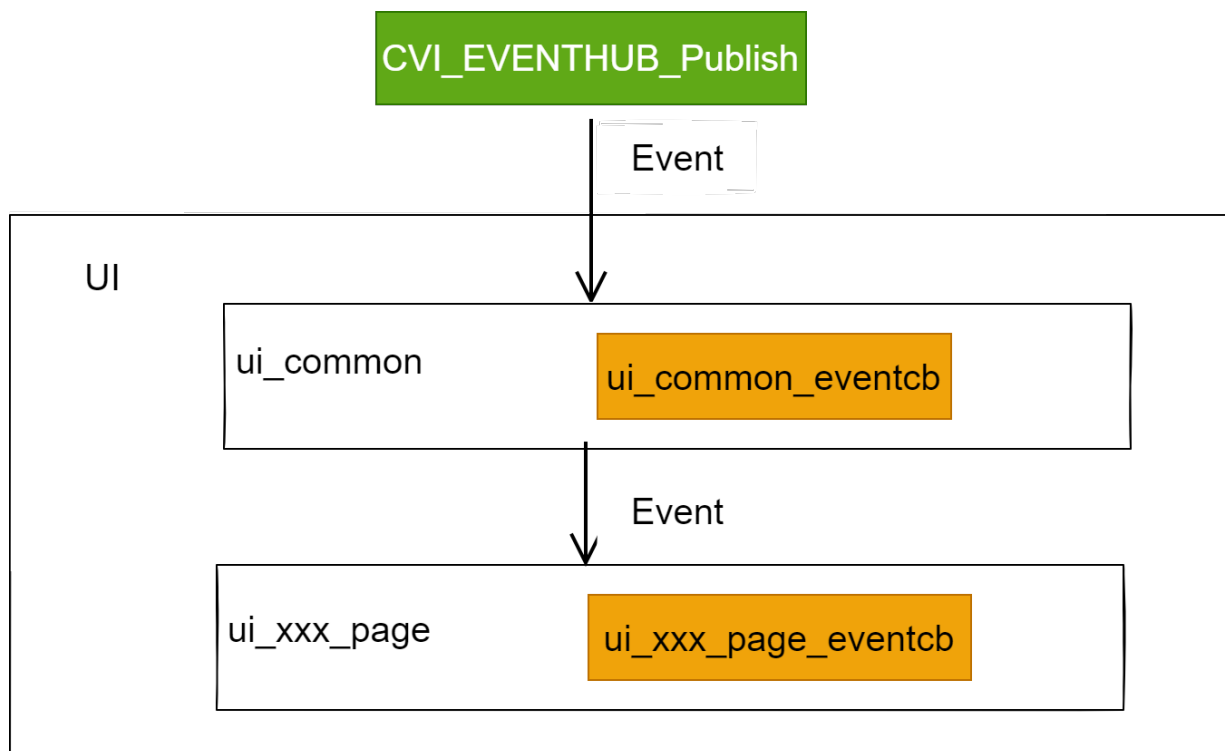


图 5.9: UI 事件划分

5.3.5 典型事件

5.3.5.1 休眠管理

1. 订阅事件: UI 在初始化函数中调用 `UI_COMMON_SubscribeEvents` 向 EventHub 订阅 UI 关心的事件, 如按键事件、sd 卡插拔事件等, 并设定事件发生时回调函数为 `ui_common_eventcb`。
2. 回调: 事件发生时, EventHub 回调 `ui_common_eventcb` 函数, 回调函数中过滤启停失败事件, 然后将过滤后的事件交给 `UI_POWERCTRL_PreProcessEvent` 函数处理。
3. 定义事件行为: `UI_POWERCTRL_PreProcessEvent` 根据不同的事件类型定义不同的行为, 如唤醒休眠、暂停休眠、恢复休眠、重置定时器等。例如按键事件发生时, 如当前处于休眠状态则唤醒休眠, 如处于非休眠状态则重置定时器, 最后将事件交给 `EventPreProc` 处理。
4. `EventPreProc` 根据行为进行休眠策略调整。