

CANN
8.0.0alpha001

应用开发接口参考

文档版本	01
发布日期	2024-12-19



版权所有 © 华为技术有限公司 2024。保留一切权利。

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

商标声明



HUAWEI和其他华为商标均为华为技术有限公司的商标。

本文档提及的其他所有商标或注册商标，由各自的所有人拥有。

注意

您购买的产品、服务或特性等应受华为公司商业合同和条款的约束，本文档中描述的全部或部分产品、服务或特性可能不在您的购买或使用范围之内。除非合同另有约定，华为公司对本文档内容不做任何明示或暗示的声明或保证。

由于产品版本升级或其他原因，本文档内容会不定期进行更新。除非另有约定，本文档仅作为使用指导，本文档中的所有陈述、信息和建议不构成任何明示或暗示的担保。

安全声明

产品生命周期政策

华为公司对产品生命周期的规定以“产品生命周期终止政策”为准，该政策的详细内容请参见如下网址：
<https://support.huawei.com/ecolumnsweb/zh/warranty-policy>

漏洞处理流程

华为公司对产品漏洞管理的规定以“漏洞处理流程”为准，该流程的详细内容请参见如下网址：
<https://www.huawei.com/cn/psirt/vul-response-process>
如企业客户须获取漏洞信息，请参见如下网址：
<https://securitybulletin.huawei.com/enterprise/cn/security-advisory>

华为初始证书权责说明

华为公司对随设备出厂的初始数字证书，发布了“华为设备初始数字证书权责说明”，该说明的详细内容请参见如下网址：
<https://support.huawei.com/enterprise/zh/bulletins-service/ENEWS2000015766>

华为企业业务最终用户许可协议(EULA)

本最终用户许可协议是最终用户（个人、公司或其他任何实体）与华为公司就华为软件的使用所缔结的协议。最终用户对华为软件的使用受本协议约束，该协议的详细内容请参见如下网址：
<https://e.huawei.com/cn/about/eula>

产品资料生命周期策略

华为公司针对随产品版本发布的售后客户资料（产品资料），发布了“产品资料生命周期策略”，该策略的详细内容请参见如下网址：
<https://support.huawei.com/enterprise/zh/bulletins-website/ENEWS2000017760>

目录

1 AscendCL API (C&C++) 1

1.1 接口列表..... 1

1.1.1 接口版本说明..... 1

1.1.2 V1.1.0 及以上版本..... 2

1.1.3 V1.2.0 及以上版本..... 17

1.1.4 V1.3.0 及以上版本..... 18

1.1.5 V1.4.0 及以上版本..... 20

1.1.6 V1.5.0 及以上版本..... 21

1.1.6.1 接口列表 (Atlas 200/300/500 推理产品&Atlas 训练系列产品&Atlas 推理系列产品) 21

1.1.6.2 接口列表 (Atlas 200/500 A2 推理产品) 22

1.1.7 V1.6.0 及以上版本..... 41

1.1.7.1 接口列表 (Atlas 200/300/500 推理产品&Atlas 训练系列产品&Atlas 推理系列产品&Atlas 200/500 A2 推理产品) 41

1.1.7.2 接口列表 (Atlas A2 训练系列产品/Atlas 800I A2 推理产品) 49

1.1.8 V1.7.0 及以上版本..... 64

1.1.9 V1.8.0 及以上版本..... 66

1.1.10 V1.9.0 及以上版本..... 69

1.1.11 V1.10.0 及以上版本..... 71

1.1.12 V1.11.0 及以上版本..... 72

1.1.12.1 接口列表 (Atlas 200/300/500 推理产品&Atlas 训练系列产品&Atlas 推理系列产品&Atlas 200/500 A2 推理产品&Atlas A2 训练系列产品/Atlas 800I A2 推理产品) 72

1.2 废弃接口/返回码列表..... 74

1.3 同步&异步 API 说明..... 75

1.4 系统配置..... 75

1.4.1 acclInit..... 75

1.4.2 acclFinalize..... 84

1.4.3 acclrtGetVersion..... 85

1.4.4 acclSetCompileopt..... 85

1.4.5 acclGetCompileopt..... 86

1.4.6 acclGetCompileoptSize..... 87

1.4.7 acclGetRecentErrMsg..... 87

1.4.8 acclAppLog..... 88

1.4.9 acclDataTypeSize..... 89

1.4.10 acclFloat16ToFloat..... 89

1.4.11 aclFloatToFloat16.....	90
1.4.12 aclGenGraphAndDumpForOp.....	90
1.4.13 aclGetCannAttributeList.....	92
1.4.14 aclGetCannAttribute.....	92
1.4.15 aclGetDeviceCapability.....	93
1.5 运行时管理.....	94
1.5.1 Device 管理.....	94
1.5.1.1 aclrtSetDevice.....	94
1.5.1.2 aclrtResetDevice.....	95
1.5.1.3 aclrtResetDeviceForce.....	96
1.5.1.4 aclrtGetDevice.....	97
1.5.1.5 aclrtGetRunMode.....	98
1.5.1.6 aclrtSetTsDevice.....	98
1.5.1.7 aclrtGetDeviceCount.....	99
1.5.1.8 aclrtGetDeviceUtilizationRate.....	99
1.5.1.9 aclrtQueryDeviceStatus.....	100
1.5.1.10 aclrtGetSocName.....	101
1.5.1.11 aclrtSetDeviceSatMode.....	101
1.5.1.12 aclrtGetDeviceSatMode.....	102
1.5.1.13 aclrtDeviceCanAccessPeer.....	102
1.5.1.14 aclrtDeviceEnablePeerAccess.....	103
1.5.1.15 aclrtDeviceDisablePeerAccess.....	104
1.5.1.16 aclrtGetOverflowStatus.....	105
1.5.1.17 aclrtResetOverflowStatus.....	106
1.5.1.18 aclrtSynchronizeDevice.....	106
1.5.1.19 aclrtSynchronizeDeviceWithTimeout.....	107
1.5.2 Context 管理.....	108
1.5.2.1 aclrtCreateContext.....	108
1.5.2.2 aclrtDestroyContext.....	109
1.5.2.3 aclrtSetCurrentContext.....	109
1.5.2.4 aclrtGetCurrentContext.....	110
1.5.2.5 aclrtCtxSetSysParamOpt.....	111
1.5.2.6 aclrtCtxGetSysParamOpt.....	111
1.5.3 Stream 管理.....	112
1.5.3.1 aclrtCreateStream.....	112
1.5.3.2 aclrtCreateStreamV2.....	113
1.5.3.3 aclrtSetStreamConfigOpt.....	114
1.5.3.4 aclrtCreateStreamWithConfig.....	115
1.5.3.5 aclrtDestroyStream.....	117
1.5.3.6 aclrtDestroyStreamForce.....	117
1.5.3.7 aclrtSetStreamOverflowSwitch.....	118
1.5.3.8 aclrtGetStreamOverflowSwitch.....	119

1.5.3.9 aclrtSetStreamFailureMode.....	119
1.5.3.10 aclrtStreamQuery.....	120
1.5.3.11 aclrtSynchronizeStream.....	121
1.5.3.12 aclrtSynchronizeStreamWithTimeout.....	121
1.5.4 Event 管理.....	122
1.5.4.1 aclrtCreateEvent.....	122
1.5.4.2 aclrtCreateEventWithFlag.....	123
1.5.4.3 aclrtCreateEventExWithFlag.....	124
1.5.4.4 aclrtDestroyEvent.....	125
1.5.4.5 aclrtRecordEvent.....	126
1.5.4.6 aclrtResetEvent.....	127
1.5.4.7 aclrtQueryEvent (废弃)	128
1.5.4.8 aclrtQueryEventStatus.....	128
1.5.4.9 aclrtQueryEventWaitStatus.....	129
1.5.4.10 aclrtSynchronizeEvent.....	130
1.5.4.11 aclrtSynchronizeEventWithTimeout.....	130
1.5.4.12 aclrtEventElapsedTime.....	131
1.5.4.13 aclrtStreamWaitEvent.....	132
1.5.4.14 aclrtSetOpWaitTimeout.....	133
1.5.5 内存管理.....	133
1.5.5.1 总体说明.....	133
1.5.5.2 aclrtMalloc.....	134
1.5.5.3 aclrtMallocAlign32.....	136
1.5.5.4 aclrtMallocCached.....	137
1.5.5.5 aclrtMemFlush.....	138
1.5.5.6 aclrtMemInvalidate.....	138
1.5.5.7 aclrtFree.....	139
1.5.5.8 aclrtMallocHost.....	139
1.5.5.9 aclrtFreeHost.....	140
1.5.5.10 aclrtMemset.....	141
1.5.5.11 aclrtMemsetAsync.....	142
1.5.5.12 aclrtMemcpy.....	143
1.5.5.13 aclrtMemcpyAsync.....	144
1.5.5.14 aclrtGetMemInfo.....	145
1.5.5.15 aclrtMemcpy2d.....	146
1.5.5.16 aclrtMemcpy2dAsync.....	148
1.5.5.17 aclrtAllocatorRegister.....	149
1.5.5.18 aclrtAllocatorUnregister.....	150
1.5.5.19 aclrtMallocPhysical.....	151
1.5.5.20 aclrtFreePhysical.....	152
1.5.5.21 aclrtReserveMemAddress.....	153
1.5.5.22 aclrtReleaseMemAddress.....	154

1.5.5.23 aclrtMapMem.....	154
1.5.5.24 aclrtUnmapMem.....	155
1.5.5.25 aclrtMemExportToShareableHandle.....	156
1.5.5.26 aclrtDeviceGetBareTgid.....	158
1.5.5.27 aclrtMemSetPidToShareableHandle.....	158
1.5.5.28 aclrtMemImportFromShareableHandle.....	159
1.5.5.29 aclrtMemGetAllocationGranularity.....	160
1.5.5.30 aclrtCmoAsync.....	161
1.5.6 执行控制.....	162
1.5.6.1 aclrtSubscribeReport.....	162
1.5.6.2 aclrtLaunchCallback.....	163
1.5.6.3 aclrtProcessReport.....	164
1.5.6.4 aclrtUnSubscribeReport.....	165
1.5.6.5 aclrtSubscribeHostFunc.....	166
1.5.6.6 aclrtProcessHostFunc.....	167
1.5.6.7 aclrtUnSubscribeHostFunc.....	168
1.5.6.8 aclrtSetOpExecuteTimeOut.....	168
1.5.7 异常处理.....	169
1.5.7.1 aclrtSetExceptionInfoCallback.....	170
1.5.7.2 aclrtGetTaskIdFromExceptionInfo.....	171
1.5.7.3 aclrtGetStreamIdFromExceptionInfo.....	171
1.5.7.4 aclrtGetThreadIdFromExceptionInfo.....	172
1.5.7.5 aclrtGetDevicIdFromExceptionInfo.....	172
1.5.7.6 aclrtGetErrorCodeFromExceptionInfo.....	173
1.5.7.7 aclrtGetMemUceInfo.....	174
1.5.7.8 aclrtDeviceTaskAbort.....	174
1.5.7.9 aclrtMemUceRepair.....	175
1.5.7.10 aclrtPeekAtLastError.....	176
1.5.7.11 aclrtGetLastError.....	176
1.5.8 Kernel 加载与执行.....	177
1.5.8.1 aclrtCreateBinary.....	177
1.5.8.2 aclrtDestroyBinary.....	177
1.5.8.3 aclrtBinaryLoad.....	178
1.5.8.4 aclrtBinaryUnLoad.....	179
1.5.8.5 aclrtBinaryGetFunction.....	179
1.5.8.6 aclrtLaunchKernel.....	180
1.5.9 算力 Group 查询与设置.....	181
1.5.9.1 aclrtSetGroup.....	181
1.5.9.2 aclrtGetGroupCount.....	182
1.5.9.3 aclrtGetAllGroupInfo.....	182
1.5.9.4 aclrtGetGroupInfoDetail.....	183
1.5.9.5 aclrtCreateGroupInfo.....	184

1.5.9.6 aclrtDestroyGroupInfo.....	185
1.6 单算子调用.....	186
1.6.1 单算子 API 执行.....	186
1.6.2 单算子模型执行.....	187
1.6.2.1 aclopSetModelDir.....	187
1.6.2.2 aclopLoad.....	188
1.6.2.3 aclopSetCompileFlag（废弃）.....	189
1.6.2.4 aclopCompile.....	190
1.6.2.5 aclopExecute（废弃）.....	191
1.6.2.6 aclopExecuteV2.....	193
1.6.2.7 aclopCompileAndExecute.....	196
1.6.2.8 aclopCompileAndExecuteV2.....	198
1.6.2.9 aclopExecWithHandle.....	201
1.6.2.10 aclopInferShape.....	203
1.6.2.11 aclopStartDumpArgs.....	204
1.6.2.12 aclopStopDumpArgs.....	205
1.6.2.13 aclopRegisterCompileFunc.....	206
1.6.2.14 aclopUnregisterCompileFunc.....	207
1.6.2.15 aclopCreateKernel.....	207
1.6.2.16 aclopSetKernelArgs.....	208
1.6.2.17 aclopSetKernelWorkspaceSizes.....	209
1.6.2.18 aclopUpdateParams.....	210
1.6.2.19 aclopSetMaxOpQueueNum.....	211
1.6.3 CBLAS 接口.....	212
1.6.3.1 总体说明.....	212
1.6.3.2 关于 A、B、C 矩阵的 Shape 及内存大小计算公式.....	212
1.6.3.3 aclblasGemvEx.....	213
1.6.3.4 aclblasCreateHandleForGemvEx.....	215
1.6.3.5 aclblasHgemv.....	216
1.6.3.6 aclblasCreateHandleForHgemv.....	218
1.6.3.7 aclblasS8gemv.....	218
1.6.3.8 aclblasCreateHandleForS8gemv.....	220
1.6.3.9 aclblasGemmEx.....	221
1.6.3.10 aclblasCreateHandleForGemmEx.....	223
1.6.3.11 aclblasHgemm.....	224
1.6.3.12 aclblasCreateHandleForHgemm.....	226
1.6.3.13 aclblasS8gemm.....	227
1.6.3.14 aclblasCreateHandleForS8gemm.....	228
1.6.3.15 aclopCast.....	229
1.6.3.16 aclopCreateHandleForCast.....	230
1.7 模型推理.....	231
1.7.1 模型加载和卸载.....	231

1.7.1.1 aclmdlLoadFromFile.....	231
1.7.1.2 aclmdlLoadFromMem.....	232
1.7.1.3 aclmdlLoadFromFileWithMem.....	233
1.7.1.4 aclmdlLoadFromMemWithMem.....	235
1.7.1.5 aclmdlLoadFromFileWithQ.....	236
1.7.1.6 aclmdlLoadFromMemWithQ.....	238
1.7.1.7 aclmdlSetConfigOpt.....	239
1.7.1.8 aclmdlLoadWithConfig.....	240
1.7.1.9 aclmdlUnload.....	241
1.7.1.10 aclmdlBundleLoadFromFile.....	242
1.7.1.11 aclmdlBundleLoadFromMem.....	243
1.7.1.12 aclmdlBundleGetModelId.....	244
1.7.1.13 aclmdlBundleGetModelNum.....	244
1.7.1.14 aclmdlBundleUnload.....	245
1.7.2 模型执行.....	245
1.7.2.1 aclmdlExecute.....	246
1.7.2.2 aclmdlExecuteAsync.....	247
1.7.2.3 aclmdlSetExecConfigOpt.....	249
1.7.2.4 aclmdlExecuteV2.....	249
1.7.2.5 aclmdlExecuteAsyncV2.....	251
1.7.2.6 aclmdlQuerySize.....	252
1.7.2.7 aclmdlQuerySizeFromMem.....	253
1.7.2.8 aclmdlQueryExeOMDesc.....	254
1.7.2.9 aclmdlSetDynamicBatchSize.....	254
1.7.2.10 aclmdlSetDynamicHWSIZE.....	255
1.7.2.11 aclmdlSetInputAIPP.....	256
1.7.2.12 aclmdlGetFirstAippInfo.....	259
1.7.2.13 aclmdlGetAippType.....	259
1.7.2.14 aclmdlSetAIPPByInputIndex.....	260
1.7.2.15 aclmdlSetInputDynamicDims.....	262
1.7.2.16 aclmdlCreateAndGetOpDesc.....	264
1.7.2.17 aclmdlInitDump.....	265
1.7.2.18 aclmdlSetDump.....	267
1.7.2.19 acldumpRegCallback.....	273
1.7.2.20 acldumpUnregCallback.....	274
1.7.2.21 aclmdlFinalizeDump.....	275
1.8 媒体数据处理 V1.....	276
1.8.1 总体说明.....	276
1.8.2 基本概念.....	281
1.8.3 内存申请与释放.....	282
1.8.3.1 aclvppMalloc.....	282
1.8.3.2 aclvppFree.....	283

1.8.4 通道创建与释放.....	283
1.8.4.1 VPC/JPEGD/JPEGE/PNGD 图片处理通道.....	283
1.8.4.1.1 acldvppCreateChannel.....	284
1.8.4.1.2 acldvppDestroyChannel.....	284
1.8.4.2 VDEC 视频解码通道.....	285
1.8.4.2.1 aclvdecCreateChannel.....	285
1.8.4.2.2 aclvdecDestroyChannel.....	286
1.8.4.3 VENC 视频编码通道.....	287
1.8.4.3.1 aclvencCreateChannel.....	287
1.8.4.3.2 aclvencDestroyChannel.....	288
1.8.5 VPC 功能.....	289
1.8.5.1 功能说明.....	289
1.8.5.2 约束说明.....	294
1.8.5.3 性能指标说明.....	308
1.8.5.4 acldvppVpcResizeAsync.....	315
1.8.5.5 acldvppVpcCropAsync.....	317
1.8.5.6 acldvppVpcCropResizeAsync.....	320
1.8.5.7 acldvppVpcBatchCropAsync.....	323
1.8.5.8 acldvppVpcBatchCropResizeAsync.....	326
1.8.5.9 acldvppVpcCropAndPasteAsync.....	329
1.8.5.10 acldvppVpcCropResizePasteAsync.....	331
1.8.5.11 acldvppVpcBatchCropAndPasteAsync.....	334
1.8.5.12 acldvppVpcBatchCropResizePasteAsync.....	338
1.8.5.13 acldvppVpcBatchCropResizeMakeBorderAsync.....	341
1.8.5.14 acldvppVpcConvertColorAsync.....	345
1.8.5.15 acldvppVpcPyrDownAsync.....	346
1.8.5.16 acldvppVpcEqualizeHistAsync.....	348
1.8.5.17 acldvppVpcMakeBorderAsync.....	349
1.8.5.18 acldvppVpcCalcHistAsync.....	350
1.8.6 JPEGD 功能.....	352
1.8.6.1 功能及约束说明.....	352
1.8.6.2 性能指标说明.....	357
1.8.6.3 acldvppJpegDecodeAsync.....	359
1.8.6.4 acldvppJpegGetImageInfo.....	361
1.8.6.5 acldvppJpegGetImageInfoV2.....	362
1.8.6.6 acldvppJpegPredictDecSize.....	363
1.8.7 JPEGE 功能.....	364
1.8.7.1 功能及约束说明.....	364
1.8.7.2 性能指标说明.....	365
1.8.7.3 acldvppJpegEncodeAsync.....	367
1.8.7.4 acldvppJpegPredictEncSize.....	369
1.8.8 PNGD 功能.....	369

1.8.8.1 功能及约束说明.....	370
1.8.8.2 性能指标说明.....	371
1.8.8.3 aclvppPngDecodeAsync.....	373
1.8.8.4 aclvppPngGetImageInfo.....	375
1.8.8.5 aclvppPngPredictDecSize.....	375
1.8.9 VDEC 功能.....	376
1.8.9.1 功能及约束说明.....	376
1.8.9.2 性能指标说明.....	379
1.8.9.3 aclvdecSendFrame.....	385
1.8.9.4 aclvdecSendSkippedFrame.....	388
1.8.9.5 aclvdecCallback.....	389
1.8.10 VENC 功能.....	390
1.8.10.1 功能及约束说明.....	390
1.8.10.2 性能指标说明.....	391
1.8.10.3 aclvencSendFrame.....	393
1.8.10.4 aclvencCallback.....	394
1.9 媒体数据处理 V2.....	395
1.9.1 总体功能及约束说明.....	395
1.9.2 基本概念.....	400
1.9.3 公共接口.....	401
1.9.3.1 hi_mpi_sys_init.....	401
1.9.3.2 hi_mpi_sys_full_init.....	402
1.9.3.3 hi_mpi_sys_exit.....	402
1.9.3.4 hi_mpi_dvpp_malloc.....	403
1.9.3.5 hi_mpi_dvpp_free.....	404
1.9.3.6 hi_mpi_dvpp_get_image_info.....	405
1.9.3.7 hi_mpi_dvpp_get_version.....	406
1.9.3.8 hi_mpi_sys_create_epoll.....	406
1.9.3.9 hi_mpi_sys_ctl_epoll.....	407
1.9.3.10 hi_mpi_sys_wait_epoll.....	408
1.9.3.11 hi_mpi_sys_close_epoll.....	409
1.9.3.12 hi_mpi_sys_set_chn_csc_matrix.....	410
1.9.3.13 hi_mpi_sys_get_chn_csc_matrix.....	411
1.9.3.14 hi_mpi_sys_get_image_align_info.....	412
1.9.3.15 hi_mpi_sys_bind.....	413
1.9.3.16 hi_mpi_sys_unbind.....	414
1.9.3.17 hi_mpi_vb_create_pool.....	415
1.9.3.18 hi_mpi_vb_destroy_pool.....	416
1.9.3.19 hi_mpi_vb_get_block.....	416
1.9.3.20 hi_mpi_vb_release_block.....	417
1.9.3.21 hi_mpi_vb_handle_to_addr.....	418
1.9.3.22 hi_mpi_vb_block_copyref.....	418

1.9.4 AI 音频输入/AO 音频输出.....	419
1.9.4.1 功能及约束说明.....	419
1.9.4.2 hi_mpi_ai_set_pub_attr.....	420
1.9.4.3 hi_mpi_ai_enable.....	421
1.9.4.4 hi_mpi_ai_disable.....	421
1.9.4.5 hi_mpi_ai_set_chn_attr.....	422
1.9.4.6 hi_mpi_ai_enable_chn.....	423
1.9.4.7 hi_mpi_ai_disable_chn.....	424
1.9.4.8 hi_mpi_ai_get_frame.....	425
1.9.4.9 hi_mpi_ai_release_frame.....	426
1.9.4.10 hi_mpi_ai_enable_resample.....	427
1.9.4.11 hi_mpi_ai_disable_resample.....	428
1.9.4.12 hi_mpi_ao_set_pub_attr.....	429
1.9.4.13 hi_mpi_ao_enable.....	430
1.9.4.14 hi_mpi_ao_disable.....	431
1.9.4.15 hi_mpi_ao_enable_chn.....	431
1.9.4.16 hi_mpi_ao_disable_chn.....	432
1.9.4.17 hi_mpi_ao_send_frame.....	433
1.9.4.18 hi_mpi_ao_get_chn_delay.....	435
1.9.4.19 hi_mpi_ao_enable_resample.....	435
1.9.4.20 hi_mpi_ao_disable_resample.....	436
1.9.5 AENC 音频编码/ADEC 音频解码.....	437
1.9.5.1 功能及约束说明.....	437
1.9.5.2 hi_mpi_aenc_create_chn.....	439
1.9.5.3 hi_mpi_aenc_destroy_chn.....	440
1.9.5.4 hi_mpi_aenc_get_stream.....	440
1.9.5.5 hi_mpi_aenc_release_stream.....	441
1.9.5.6 hi_mpi_adec_create_chn.....	442
1.9.5.7 hi_mpi_adec_destroy_chn.....	443
1.9.5.8 hi_mpi_adec_send_stream.....	443
1.9.6 音量调节.....	444
1.9.6.1 功能描述.....	444
1.9.6.2 HI_ACODEC_SET_DACL_VOLUME.....	445
1.9.6.3 HI_ACODEC_SET_DACR_VOLUME.....	445
1.9.6.4 HI_ACODEC_GET_DACL_VOLUME.....	446
1.9.6.5 HI_ACODEC_GET_DACR_VOLUME.....	447
1.9.6.6 HI_ACODEC_SET_ADCL_VOLUME.....	448
1.9.6.7 HI_ACODEC_SET_ADCR_VOLUME.....	449
1.9.6.8 HI_ACODEC_GET_ADCL_VOLUME.....	449
1.9.6.9 HI_ACODEC_GET_ADCR_VOLUME.....	450
1.9.7 ISP 系统控制及 3A 算法注册.....	451
1.9.7.1 功能描述.....	451

1.9.7.2 基本概念.....	452
1.9.7.3 hi_mpi_isp_sensor_reg_callback.....	452
1.9.7.4 hi_mpi_isp_sensor_unreg_callback.....	453
1.9.7.5 hi_mpi_isp_mem_init.....	454
1.9.7.6 hi_mpi_isp_set_pub_attr.....	455
1.9.7.7 hi_mpi_isp_get_pub_attr.....	456
1.9.7.8 hi_mpi_isp_init.....	457
1.9.7.9 hi_mpi_isp_run.....	458
1.9.7.10 hi_mpi_isp_exit.....	459
1.9.7.11 hi_mpi_isp_set_sns_slave_attr.....	460
1.9.7.12 hi_mpi_isp_get_sns_slave_attr.....	461
1.9.7.13 hi_mpi_ae_register.....	461
1.9.7.14 hi_mpi_ae_unregister.....	462
1.9.7.15 hi_mpi_ae_sensor_reg_callback.....	463
1.9.7.16 hi_mpi_ae_sensor_unreg_callback.....	464
1.9.7.17 hi_mpi_awb_register.....	465
1.9.7.18 hi_mpi_awb_unregister.....	466
1.9.7.19 hi_mpi_awb_sensor_reg_callback.....	467
1.9.7.20 hi_mpi_awb_sensor_unreg_callback.....	468
1.9.8 MIPI Rx ioctl 命令字.....	469
1.9.8.1 功能描述.....	469
1.9.8.2 基本概念.....	470
1.9.8.3 HI_MIPI_SET_DEV_ATTR.....	472
1.9.8.4 HI_MIPI_SET_HS_MODE.....	473
1.9.8.5 HI_MIPI_ENABLE_MIPI_CLOCK.....	473
1.9.8.6 HI_MIPI_DISABLE_MIPI_CLOCK.....	474
1.9.8.7 HI_MIPI_RESET_MIPI.....	475
1.9.8.8 HI_MIPI_UNRESET_MIPI.....	475
1.9.8.9 HI_MIPI_ENABLE_SLVS_CLOCK.....	476
1.9.8.10 HI_MIPI_DISABLE_SLVS_CLOCK.....	476
1.9.8.11 HI_MIPI_RESET_SLVS.....	477
1.9.8.12 HI_MIPI_UNRESET_SLVS.....	477
1.9.8.13 HI_MIPI_CONFIG_SENSOR_CLOCK.....	478
1.9.8.14 HI_MIPI_ENABLE_SENSOR_CLOCK.....	479
1.9.8.15 HI_MIPI_DISABLE_SENSOR_CLOCK.....	479
1.9.8.16 HI_MIPI_RESET_SENSOR.....	480
1.9.8.17 HI_MIPI_UNRESET_SENSOR.....	480
1.9.9 VI 视频输入功能.....	481
1.9.9.1 功能说明.....	481
1.9.9.2 约束说明.....	481
1.9.9.3 基本概念.....	482
1.9.9.4 hi_mpi_vi_set_dev_bind_pipe.....	485

1.9.9.5 hi_mpi_vi_get_dev_bind_pipe.....	486
1.9.9.6 hi_mpi_vi_set_dev_attr.....	487
1.9.9.7 hi_mpi_vi_get_dev_attr.....	487
1.9.9.8 hi_mpi_vi_enable_dev.....	488
1.9.9.9 hi_mpi_vi_disable_dev.....	489
1.9.9.10 hi_mpi_vi_create_pipe.....	490
1.9.9.11 hi_mpi_vi_destroy_pipe.....	491
1.9.9.12 hi_mpi_vi_set_vi_vpss_mode.....	492
1.9.9.13 hi_mpi_vi_get_vi_vpss_mode.....	493
1.9.9.14 hi_mpi_vi_set_pipe_attr.....	494
1.9.9.15 hi_mpi_vi_get_pipe_attr.....	494
1.9.9.16 hi_mpi_vi_start_pipe.....	495
1.9.9.17 hi_mpi_vi_stop_pipe.....	496
1.9.9.18 hi_mpi_vi_set_pipe_pre_crop.....	497
1.9.9.19 hi_mpi_vi_get_pipe_pre_crop.....	498
1.9.9.20 hi_mpi_vi_set_pipe_pre_be_crop.....	498
1.9.9.21 hi_mpi_vi_set_pipe_bind_strobe.....	499
1.9.9.22 hi_mpi_vi_set_pipe_frame_dump_attr.....	500
1.9.9.23 hi_mpi_vi_get_pipe_frame_dump_attr.....	501
1.9.9.24 hi_mpi_vi_get_pipe_frame.....	502
1.9.9.25 hi_mpi_vi_release_pipe_frame.....	503
1.9.9.26 hi_mpi_vi_set_pipe_frame_source.....	504
1.9.9.27 hi_mpi_vi_get_pipe_frame_source.....	505
1.9.9.28 hi_mpi_vi_send_pipe_yuv.....	505
1.9.9.29 hi_mpi_vi_send_pipe_raw.....	507
1.9.9.30 hi_mpi_vi_pipe_get_buffer.....	508
1.9.9.31 hi_mpi_vi_pipe_release_buffer.....	509
1.9.9.32 hi_mpi_vi_set_pipe_vc_number.....	510
1.9.9.33 hi_mpi_vi_get_pipe_vc_number.....	511
1.9.9.34 hi_mpi_vi_get_pipe_fd.....	512
1.9.9.35 hi_mpi_vi_set_chn_attr.....	512
1.9.9.36 hi_mpi_vi_get_chn_attr.....	513
1.9.9.37 hi_mpi_vi_enable_chn.....	514
1.9.9.38 hi_mpi_vi_disable_chn.....	515
1.9.9.39 hi_mpi_vi_set_chn_dis_config.....	516
1.9.9.40 hi_mpi_vi_get_chn_dis_config.....	517
1.9.9.41 hi_mpi_vi_set_chn_dis_attr.....	518
1.9.9.42 hi_mpi_vi_get_chn_dis_attr.....	519
1.9.9.43 hi_mpi_vi_set_chn_dis_param.....	519
1.9.9.44 hi_mpi_vi_get_chn_dis_param.....	520
1.9.9.45 hi_mpi_vi_set_chn_ldc_attr.....	521
1.9.9.46 hi_mpi_vi_get_chn_ldc_attr.....	522

1.9.9.47 hi_mpi_vi_get_chn_frame.....	523
1.9.9.48 hi_mpi_vi_release_chn_frame.....	524
1.9.9.49 hi_mpi_vi_get_chn_fd.....	525
1.9.9.50 hi_mpi_vi_set_chn_low_delay_attr.....	526
1.9.9.51 hi_mpi_vi_get_chn_low_delay_attr.....	527
1.9.10 Region 区域管理.....	528
1.9.10.1 功能及约束说明.....	528
1.9.10.2 hi_mpi_rgn_create.....	530
1.9.10.3 hi_mpi_rgn_destroy.....	531
1.9.10.4 hi_mpi_rgn_attach_to_chn.....	532
1.9.10.5 hi_mpi_rgn_detach_from_chn.....	533
1.9.10.6 hi_mpi_rgn_set_display_attr.....	534
1.9.10.7 hi_mpi_rgn_get_display_attr.....	535
1.9.10.8 hi_mpi_rgn_get_canvas_info.....	536
1.9.10.9 hi_mpi_rgn_update_canvas.....	537
1.9.11 VPSS 视频处理功能.....	537
1.9.11.1 功能说明.....	537
1.9.11.2 约束说明.....	539
1.9.11.3 hi_mpi_vpss_create_grp.....	540
1.9.11.4 hi_mpi_vpss_destroy_grp.....	541
1.9.11.5 hi_mpi_vpss_start_grp.....	542
1.9.11.6 hi_mpi_vpss_stop_grp.....	543
1.9.11.7 hi_mpi_vpss_set_grp_crop.....	544
1.9.11.8 hi_mpi_vpss_get_grp_crop.....	545
1.9.11.9 hi_mpi_vpss_set_grp_param.....	546
1.9.11.10 hi_mpi_vpss_get_grp_param.....	547
1.9.11.11 hi_mpi_vpss_set_grp_fisheye_cfg.....	547
1.9.11.12 hi_mpi_vpss_get_grp_fisheye_cfg.....	548
1.9.11.13 hi_mpi_vpss_set_chn_attr.....	549
1.9.11.14 hi_mpi_vpss_get_chn_attr.....	550
1.9.11.15 hi_mpi_vpss_enable_chn.....	551
1.9.11.16 hi_mpi_vpss_disable_chn.....	552
1.9.11.17 hi_mpi_vpss_get_chn_frame.....	553
1.9.11.18 hi_mpi_vpss_release_chn_frame.....	554
1.9.11.19 hi_mpi_vpss_set_chn_fisheye.....	555
1.9.11.20 hi_mpi_vpss_get_chn_fisheye.....	556
1.9.11.21 hi_mpi_vpss_get_chn_fd.....	557
1.9.12 VO 视频输出功能.....	558
1.9.12.1 hi_mpi_vo_set_pub_attr.....	558
1.9.12.2 hi_mpi_vo_enable.....	559
1.9.12.3 hi_mpi_vo_disable.....	560
1.9.12.4 hi_mpi_vo_set_video_layer_attr.....	561

1.9.12.5 hi_mpi_vo_get_video_layer_attr.....	562
1.9.12.6 hi_mpi_vo_enable_video_layer.....	562
1.9.12.7 hi_mpi_vo_disable_video_layer.....	563
1.9.12.8 hi_mpi_vo_set_chn_attr.....	564
1.9.12.9 hi_mpi_vo_enable_chn.....	565
1.9.12.10 hi_mpi_vo_disable_chn.....	566
1.9.12.11 hi_mpi_vo_set_chn_param.....	567
1.9.12.12 hi_mpi_vo_set_chn_frame_rate.....	568
1.9.12.13 hi_mpi_vo_pause_chn.....	569
1.9.12.14 hi_mpi_vo_resume_chn.....	570
1.9.12.15 hi_mpi_vo_hide_chn.....	571
1.9.12.16 hi_mpi_vo_show_chn.....	572
1.9.12.17 hi_mpi_vo_send_frame.....	573
1.9.12.18 hi_mpi_vo_create_pool.....	574
1.9.12.19 hi_mpi_vo_handle_to_phys_addr.....	575
1.9.12.20 hi_mpi_vo_destroy_pool.....	575
1.9.12.21 hi_mpi_vo_bind_layer.....	575
1.9.12.22 hi_mpi_vo_unbind_layer.....	576
1.9.13 TDE 图形绘制功能.....	577
1.9.13.1 hi_tde_open.....	577
1.9.13.2 hi_tde_close.....	578
1.9.13.3 hi_tde_begin_job.....	578
1.9.13.4 hi_tde_end_job.....	580
1.9.13.5 hi_tde_cancel_job.....	581
1.9.13.6 hi_tde_wait_for_done.....	581
1.9.13.7 hi_tde_wait_all_done.....	582
1.9.13.8 hi_tde_quick_copy.....	583
1.9.13.9 hi_tde_quick_fill.....	584
1.9.13.10 hi_tde_pattern_fill.....	585
1.9.14 HIFB 叠加图形层管理功能.....	587
1.9.14.1 功能及约束说明.....	587
1.9.14.2 FBIOPUT_VSCREENINFO.....	588
1.9.14.3 FBIOPUT_VSCREENINFO.....	589
1.9.14.4 FBIOPUT_FSCREENINFO.....	590
1.9.14.5 FBIOPAN_DISPLAY.....	591
1.9.14.6 FBIOPUT_CMAP.....	593
1.9.14.7 FBIOPUT_CMAP.....	593
1.9.14.8 FBIOPUT_SCREEN_ORIGIN_HIFB.....	594
1.9.14.9 FBIOPUT_SCREEN_ORIGIN_HIFB.....	595
1.9.14.10 FBIOPUT_SHOW_HIFB.....	596
1.9.14.11 FBIOPUT_SHOW_HIFB.....	597
1.9.14.12 FBIOPUT_ALPHA_HIFB.....	598

1.9.14.13 FBIOPUT_ALPHA_HIFB.....	599
1.9.15 HDMI 外设.....	599
1.9.15.1 hi_mpi_hdmi_init.....	599
1.9.15.2 hi_mpi_hdmi_deinit.....	600
1.9.15.3 hi_mpi_hdmi_open.....	601
1.9.15.4 hi_mpi_hdmi_close.....	602
1.9.15.5 hi_mpi_hdmi_set_attr.....	603
1.9.15.6 hi_mpi_hdmi_get_attr.....	604
1.9.15.7 hi_mpi_hdmi_start.....	604
1.9.15.8 hi_mpi_hdmi_stop.....	605
1.9.15.9 hi_mpi_hdmi_set_infoframe.....	606
1.9.15.10 hi_mpi_hdmi_get_sink_capability.....	607
1.9.15.11 hi_mpi_hdmi_register_callback.....	608
1.9.16 VPC 图像处理功能.....	609
1.9.16.1 功能说明.....	609
1.9.16.2 约束说明.....	616
1.9.16.3 性能指标说明.....	625
1.9.16.4 hi_mpi_vpc_create_chn.....	628
1.9.16.5 hi_mpi_vpc_destroy_chn.....	630
1.9.16.6 hi_mpi_vpc_resize.....	630
1.9.16.7 hi_mpi_vpc_crop.....	633
1.9.16.8 hi_mpi_vpc_crop_resize.....	636
1.9.16.9 hi_mpi_vpc_crop_resize_paste.....	638
1.9.16.10 hi_mpi_vpc_convert_color.....	642
1.9.16.11 hi_mpi_vpc_convert_color_v2.....	643
1.9.16.12 hi_mpi_vpc_convert_color_to_yuv420.....	645
1.9.16.13 hi_mpi_vpc_copy_make_border.....	647
1.9.16.14 hi_mpi_vpc_pyrdown.....	650
1.9.16.15 hi_mpi_vpc_calc_hist.....	652
1.9.16.16 hi_mpi_vpc_equalize_hist.....	654
1.9.16.17 hi_mpi_vpc_crop_resize_make_border.....	655
1.9.16.18 hi_mpi_vpc_batch_crop_resize_paste.....	658
1.9.16.19 hi_mpi_vpc_batch_crop_resize_make_border.....	661
1.9.16.20 hi_mpi_vpc_get_process_result.....	664
1.9.16.21 hi_mpi_vpc_sys_create_chn.....	666
1.9.16.22 hi_mpi_vpc_set_roundview_stitching_param.....	666
1.9.16.23 hi_mpi_vpc_get_roundview_stitching_param.....	667
1.9.16.24 hi_mpi_vpc_roundview_stitching.....	668
1.9.16.25 hi_mpi_vpc_crop_resize_resize_paste.....	669
1.9.16.26 hi_mpi_vpc_set_chn_workspace.....	671
1.9.16.27 hi_mpi_vpc_get_lut_mem_size.....	672
1.9.16.28 hi_mpi_vpc_get_affine_lut.....	672

1.9.16.29 hi_mpi_vpc_get_perspective_lut.....	673
1.9.16.30 hi_mpi_vpc_get_remap_lut.....	674
1.9.16.31 hi_mpi_vpc_lut_remap.....	675
1.9.16.32 hi_mpi_vpc_median_blur.....	679
1.9.16.33 hi_mpi_vpc_erode.....	680
1.9.16.34 hi_mpi_vpc_dilate.....	681
1.9.16.35 hi_mpi_vpc_blur.....	683
1.9.16.36 hi_mpi_vpc_gaussian_blur.....	686
1.9.16.37 hi_mpi_vpc_filter2d.....	689
1.9.16.38 hi_mpi_vpc_rotate.....	692
1.9.16.39 hi_mpi_vpc_draw_mosaic.....	694
1.9.16.40 hi_mpi_vpc_draw_cover.....	695
1.9.16.41 hi_mpi_vpc_draw_line.....	697
1.9.16.42 hi_mpi_vpc_draw_osd.....	698
1.9.16.43 hi_mpi_vpc_get_affine_transform.....	699
1.9.16.44 hi_mpi_vpc_get_rotation_matrix.....	700
1.9.16.45 hi_mpi_vpc_warp_affine.....	701
1.9.16.46 hi_mpi_vpc_flip.....	704
1.9.16.47 hi_mpi_vpc_set_chn_opt_attr.....	706
1.9.17 VDEC 视频解码功能/JPEGD 图片解码功能.....	707
1.9.17.1 JPEGD 功能及约束说明.....	707
1.9.17.2 JPEGD 性能指标数据.....	711
1.9.17.3 VDEC 功能及约束说明.....	713
1.9.17.4 VDEC 性能指标数据.....	716
1.9.17.5 hi_mpi_vdec_create_chn.....	718
1.9.17.6 hi_mpi_vdec_destroy_chn.....	721
1.9.17.7 hi_mpi_vdec_get_chn_attr.....	721
1.9.17.8 hi_mpi_vdec_set_chn_attr.....	722
1.9.17.9 hi_mpi_vdec_start_rcv_stream.....	723
1.9.17.10 hi_mpi_vdec_stop_rcv_stream.....	724
1.9.17.11 hi_mpi_vdec_query_status.....	725
1.9.17.12 hi_mpi_vdec_reset_chn.....	726
1.9.17.13 hi_mpi_vdec_set_chn_param.....	727
1.9.17.14 hi_mpi_vdec_get_chn_param.....	728
1.9.17.15 hi_mpi_vdec_set_protocol_param.....	729
1.9.17.16 hi_mpi_vdec_get_protocol_param.....	730
1.9.17.17 hi_mpi_vdec_send_stream.....	731
1.9.17.18 hi_mpi_vdec_get_frame.....	733
1.9.17.19 hi_mpi_vdec_release_frame.....	735
1.9.17.20 hi_mpi_vdec_get_fd.....	736
1.9.17.21 hi_mpi_vdec_close_fd.....	737
1.9.17.22 hi_vdec_get_pic_buf_size.....	737

1.9.17.23 hi_vdec_get_tmv_buf_size.....	738
1.9.17.24 hi_mpi_vdec_set_jpegd_precision_mode.....	739
1.9.17.25 hi_mpi_vdec_get_jpegd_output_info.....	740
1.9.17.26 hi_mpi_vdec_set_display_mode.....	741
1.9.17.27 hi_mpi_vdec_get_display_mode.....	742
1.9.18 VENC 视频编码功能/JPEGE 图片编码功能.....	743
1.9.18.1 JPEGE 功能及约束说明.....	743
1.9.18.2 JPEGE 性能指标说明.....	747
1.9.18.3 VENC 功能及约束说明.....	748
1.9.18.4 VENC 性能指标说明.....	750
1.9.18.5 hi_mpi_venc_create_chn.....	751
1.9.18.6 hi_mpi_venc_destroy_chn.....	753
1.9.18.7 hi_mpi_venc_start_chn.....	754
1.9.18.8 hi_mpi_venc_stop_chn.....	755
1.9.18.9 hi_mpi_venc_query_status.....	756
1.9.18.10 hi_mpi_venc_get_stream.....	758
1.9.18.11 hi_mpi_venc_release_stream.....	761
1.9.18.12 hi_mpi_venc_send_frame.....	762
1.9.18.13 hi_mpi_venc_set_mod_param.....	763
1.9.18.14 hi_mpi_venc_get_mod_param.....	764
1.9.18.15 hi_mpi_venc_request_idr.....	765
1.9.18.16 hi_mpi_venc_get_fd.....	766
1.9.18.17 hi_mpi_venc_close_fd.....	767
1.9.18.18 hi_mpi_venc_set_jpeg_param.....	768
1.9.18.19 hi_mpi_venc_get_jpeg_param.....	769
1.9.18.20 hi_mpi_venc_set_chn_param.....	770
1.9.18.21 hi_mpi_venc_get_chn_param.....	770
1.9.18.22 hi_mpi_venc_set_scene_mode.....	771
1.9.18.23 hi_mpi_venc_get_scene_mode.....	772
1.9.18.24 hi_mpi_venc_set_rc_param.....	773
1.9.18.25 hi_mpi_venc_get_rc_param.....	774
1.9.18.26 hi_mpi_venc_set_jpeg_huffman_param.....	775
1.9.18.27 hi_mpi_venc_get_jpeg_huffman_param.....	776
1.9.18.28 hi_mpi_venc_compact_jpeg_tables.....	777
1.9.18.29 hi_mpi_venc_send_jpege_frame.....	778
1.9.18.30 hi_mpi_venc_get_jpege_predicted_size.....	780
1.9.18.31 hi_mpi_venc_set_roi_attr.....	780
1.9.18.32 hi_mpi_venc_get_roi_attr.....	782
1.9.18.33 hi_mpi_venc_set_ref_param.....	783
1.9.18.34 hi_mpi_venc_get_ref_param.....	786
1.9.18.35 hi_mpi_venc_set_h264_vui.....	787
1.9.18.36 hi_mpi_venc_get_h264_vui.....	788

1.9.18.37 hi_mpi_venc_set_h265_vui.....	789
1.9.18.38 hi_mpi_venc_get_h265_vui.....	791
1.9.18.39 hi_mpi_venc_set_cu_pred.....	792
1.9.18.40 hi_mpi_venc_get_cu_pred.....	793
1.9.18.41 hi_mpi_venc_set_intra_refresh.....	794
1.9.18.42 hi_mpi_venc_get_intra_refresh.....	795
1.9.18.43 hi_mpi_venc_set_chn_attr.....	796
1.9.18.44 hi_mpi_venc_get_chn_attr.....	798
1.9.18.45 hi_mpi_venc_set_slice_split.....	798
1.9.18.46 hi_mpi_venc_get_slice_split.....	799
1.9.18.47 hi_mpi_venc_send_frame_ex.....	800
1.9.18.48 hi_mpi_venc_get_qpmap_stride.....	801
1.9.18.49 hi_mpi_venc_get_qpmap_size.....	802
1.9.18.50 hi_mpi_venc_set_streamcopy_param.....	803
1.9.18.51 hi_mpi_venc_get_streamcopy_param.....	803
1.9.19 PNGD 图片解码功能.....	804
1.9.19.1 功能及约束说明.....	804
1.9.19.2 性能指标说明.....	807
1.9.19.3 hi_mpi_pngd_create_chn.....	807
1.9.19.4 hi_mpi_pngd_destroy_chn.....	809
1.9.19.5 hi_mpi_pngd_send_stream.....	809
1.9.19.6 hi_mpi_pngd_get_image_data.....	810
1.9.19.7 hi_mpi_png_get_image_info.....	812
1.9.20 枚举值及数据结构.....	812
1.9.20.1 公共.....	813
1.9.20.1.1 基本数据类型说明.....	813
1.9.20.1.2 hi_pixel_format.....	813
1.9.20.1.3 hi_payload_type.....	817
1.9.20.1.4 hi_video_field.....	819
1.9.20.1.5 hi_dynamic_range.....	819
1.9.20.1.6 hi_color_gamut.....	821
1.9.20.1.7 hi_video_supplement.....	821
1.9.20.1.8 hi_video_frame.....	822
1.9.20.1.9 hi_video_frame_info.....	826
1.9.20.1.10 hi_vb_src.....	826
1.9.20.1.11 hi_mod_id.....	827
1.9.20.1.12 hi_data_bit_width.....	828
1.9.20.1.13 hi_video_format.....	828
1.9.20.1.14 hi_img_info.....	829
1.9.20.1.15 hi_pic_buf_attr.....	830
1.9.20.1.16 hi_jpeg_raw_format.....	830
1.9.20.1.17 hi_dvpp_epoll_ctl_op.....	831

1.9.20.1.18 hi_dvpp_epoll_event_type.....	831
1.9.20.1.19 hi_dvpp_epoll_event.....	832
1.9.20.1.20 hi_csc_matrix.....	832
1.9.20.1.21 hi_coefficient.....	834
1.9.20.1.22 hi_csc_coefficient.....	835
1.9.20.1.23 hi_video_size.....	835
1.9.20.1.24 hi_img_stream.....	836
1.9.20.1.25 hi_pic_info.....	836
1.9.20.1.26 hi_png_color_format.....	837
1.9.20.1.27 hi_rect.....	837
1.9.20.1.28 hi_op_mode.....	838
1.9.20.1.29 hi_point.....	838
1.9.20.1.30 hi_crop_info.....	838
1.9.20.1.31 hi_frame_rate_ctrl.....	839
1.9.20.1.32 hi_compress_mode.....	839
1.9.20.1.33 hi_mpp_chn.....	840
1.9.20.1.34 hi_size.....	841
1.9.20.1.35 hi_module_type.....	841
1.9.20.1.36 hi_img_base_info.....	841
1.9.20.1.37 hi_img_align_info.....	842
1.9.20.1.38 hi_coord.....	842
1.9.20.1.39 hi_aspect_ratio_type.....	843
1.9.20.1.40 hi_aspect_ratio.....	843
1.9.20.1.41 hi_video_display_mode.....	844
1.9.20.1.42 hi_wdr_mode.....	844
1.9.20.1.43 hi_vb_pool.....	845
1.9.20.1.44 hi_vb_blk.....	846
1.9.20.1.45 hi_vb_remap_mode.....	846
1.9.20.1.46 hi_vb_pool_config.....	846
1.9.20.2 音频相关.....	847
1.9.20.2.1 基本数据类型.....	847
1.9.20.2.2 hi_audio_sample_rate.....	847
1.9.20.2.3 hi_aio_mode.....	848
1.9.20.2.4 hi_audio_bit_width.....	849
1.9.20.2.5 hi_audio_snd_mode.....	850
1.9.20.2.6 hi_aio_i2s_type.....	850
1.9.20.2.7 hi_aio_attr.....	851
1.9.20.2.8 hi_audio_frame.....	852
1.9.20.2.9 hi_audio_stream.....	852
1.9.20.2.10 hi_aec_frame.....	853
1.9.20.2.11 hi_audio_frame_info.....	854
1.9.20.2.12 hi_acodec_volume_ctrl.....	854

1.9.20.2.13 音量调节宏.....	855
1.9.20.2.14 hi_aenc_chn_attr.....	855
1.9.20.2.15 hi_adec_mode.....	856
1.9.20.2.16 hi_adec_chn_attr.....	857
1.9.20.2.17 hi_adec_attr_g711.....	857
1.9.20.2.18 hi_ai_chn_mode.....	858
1.9.20.2.19 hi_ai_chn_attr.....	858
1.9.20.2.20 hi_aac_type.....	858
1.9.20.2.21 hi_aac_bps.....	859
1.9.20.2.22 hi_aac_transport_type.....	860
1.9.20.2.23 hi_aenc_attr_aac.....	860
1.9.20.2.24 hi_adec_attr_aac.....	861
1.9.20.3 ISP 系统控制及 3A 算法注册.....	861
1.9.20.3.1 hi_sensor_id.....	861
1.9.20.3.2 hi_isp_sns_attr_info.....	862
1.9.20.3.3 hi_isp_cmos_sensor_image_mode.....	862
1.9.20.3.4 hi_isp_cmos_alg_key.....	863
1.9.20.3.5 hi_isp_cmos_noise_calibration.....	864
1.9.20.3.6 hi_isp_cmos_sensor_max_resolution.....	865
1.9.20.3.7 hi_isp_cmos_sensor_mode.....	865
1.9.20.3.8 hi_isp_cmos_wdr_switch_attr.....	865
1.9.20.3.9 hi_isp_cmos_default.....	866
1.9.20.3.10 hi_isp_cmos_black_level.....	868
1.9.20.3.11 hi_isp_sns_type.....	869
1.9.20.3.12 hi_isp_sns_commbus.....	869
1.9.20.3.13 hi_isp_i2c_data.....	870
1.9.20.3.14 hi_isp_ssp_data.....	871
1.9.20.3.15 hi_isp_sns_regs_info.....	872
1.9.20.3.16 hi_isp_sensor_exp_func.....	873
1.9.20.3.17 hi_isp_sensor_register.....	875
1.9.20.3.18 hi_isp_bayer_format.....	875
1.9.20.3.19 hi_isp_pub_attr.....	876
1.9.20.3.20 hi_isp_frame_info.....	876
1.9.20.3.21 hi_slave_dev.....	878
1.9.20.3.22 hi_isp_slave_sns_sync.....	879
1.9.20.3.23 hi_isp_3a_alg_lib.....	881
1.9.20.3.24 hi_isp_ae_accuracy_type.....	881
1.9.20.3.25 hi_isp_ae_accuracy.....	882
1.9.20.3.26 hi_isp_ae_sensor_default.....	883
1.9.20.3.27 hi_isp_ae_sensor_exp_func.....	889
1.9.20.3.28 hi_isp_ae_sensor_register.....	890
1.9.20.3.29 hi_isp_awb_agc_table.....	891

1.9.20.3.30 hi_isp_awb_ccm_tab.....	891
1.9.20.3.31 hi_isp_awb_ccm.....	892
1.9.20.3.32 hi_isp_awb_sensor_default.....	892
1.9.20.3.33 hi_isp_awb_sensor_exp_func.....	894
1.9.20.3.34 hi_isp_awb_sensor_register.....	894
1.9.20.4 MIPI Rx ioctl 命令字参数.....	895
1.9.20.4.1 combo_dev_t.....	895
1.9.20.4.2 sns_rst_source_t.....	895
1.9.20.4.3 sns_clk_source_t.....	895
1.9.20.4.4 lane_divide_mode_t.....	896
1.9.20.4.5 input_mode_t.....	896
1.9.20.4.6 img_rect_t.....	896
1.9.20.4.7 slvs_lane_rate_t.....	897
1.9.20.4.8 slvs_err_check_mode_t.....	897
1.9.20.4.9 data_type_t.....	898
1.9.20.4.10 mipi_data_rate_t.....	899
1.9.20.4.11 mipi_wdr_mode_t.....	899
1.9.20.4.12 slvs_wdr_mode_t.....	900
1.9.20.4.13 lvds_wdr_mode_t.....	900
1.9.20.4.14 mipi_dev_attr_t.....	901
1.9.20.4.15 lvds_sync_mode_t.....	902
1.9.20.4.16 lvds_bit_endian_t.....	902
1.9.20.4.17 lvds_vsync_type_t.....	902
1.9.20.4.18 lvds_vsync_attr_t.....	903
1.9.20.4.19 lvds_fid_type_t.....	904
1.9.20.4.20 lvds_fid_attr_t.....	904
1.9.20.4.21 lvds_dev_attr_t.....	905
1.9.20.4.22 slvs_dev_attr_t.....	906
1.9.20.4.23 combo_dev_attr_t.....	907
1.9.20.4.24 sns_clk_freq_t.....	908
1.9.20.4.25 sns_clk_cfg_t.....	909
1.9.20.5 VI 视频输入.....	909
1.9.20.5.1 hi_vi_dev.....	909
1.9.20.5.2 hi_vi_pipe.....	909
1.9.20.5.3 hi_vi_chn.....	910
1.9.20.5.4 hi_vi_intf_mode.....	910
1.9.20.5.5 hi_data_rate.....	911
1.9.20.5.6 hi_vi_scan_mode.....	911
1.9.20.5.7 hi_vi_data_seq.....	912
1.9.20.5.8 hi_vi_data_type.....	912
1.9.20.5.9 hi_vi_dev_attr.....	913
1.9.20.5.10 hi_vi_dev_bind_pipe.....	914

1.9.20.5.11 hi_vi_wdr_attr.....	914
1.9.20.5.12 hi_vi_pipe_attr.....	915
1.9.20.5.13 hi_vi_vpss_mode.....	916
1.9.20.5.14 hi_vi_vpss_mode_type.....	917
1.9.20.5.15 hi_vi_pipe_bypass_mode.....	918
1.9.20.5.16 hi_vi_frame_dump_attr.....	918
1.9.20.5.17 hi_vi_pipe_frame_source.....	919
1.9.20.5.18 hi_vi_chn_attr.....	919
1.9.20.5.19 hi_dis_config.....	921
1.9.20.5.20 hi_dis_attr.....	922
1.9.20.5.21 hi_dis_mode.....	923
1.9.20.5.22 hi_dis_motion_level.....	924
1.9.20.5.23 hi_dis_pdt_type.....	924
1.9.20.5.24 hi_dis_param.....	925
1.9.20.5.25 hi_vi_ldc_attr.....	925
1.9.20.5.26 hi_ldc_v1_attr.....	926
1.9.20.5.27 hi_ldc_v2_attr.....	927
1.9.20.5.28 hi_ldc_version.....	928
1.9.20.5.29 hi_vi_low_delay_info.....	929
1.9.20.6 Region 区域管理.....	929
1.9.20.6.1 HI_RGN_CLUT_NUM.....	929
1.9.20.6.2 HI_RGN_HANDLE_MAX.....	929
1.9.20.6.3 hi_rgn_handle.....	929
1.9.20.6.4 hi_phys_addr_t.....	930
1.9.20.6.5 hi_rgn_type.....	930
1.9.20.6.6 hi_rgn_overlayex_attr.....	931
1.9.20.6.7 hi_rgn_overlayex_chn_attr.....	932
1.9.20.6.8 hi_rgn_cover_chn_attr.....	933
1.9.20.6.9 hi_rgn_mosaic_chn_attr.....	933
1.9.20.6.10 hi_rgn_type_attr.....	934
1.9.20.6.11 hi_rgn_type_chn_attr.....	934
1.9.20.6.12 hi_rgn_attr.....	935
1.9.20.6.13 hi_rgn_chn_attr.....	935
1.9.20.6.14 hi_rgn_canvas_info.....	936
1.9.20.7 VPSS 视频处理.....	936
1.9.20.7.1 基本数据类型说明.....	936
1.9.20.7.2 hi_vpss_crop_info.....	937
1.9.20.7.3 hi_vpss_nr_type.....	938
1.9.20.7.4 hi_vpss_nr_motion_mode.....	938
1.9.20.7.5 hi_vpss_nr_attr.....	939
1.9.20.7.6 hi_vpss_grp_attr.....	939
1.9.20.7.7 hi_vpss_nr_version.....	940

1.9.20.7.8 hi_vpss_nrx_v3.....	941
1.9.20.7.9 hi_v200_vpss_ley.....	941
1.9.20.7.10 hi_v200_vpss_sfy.....	942
1.9.20.7.11 hi_v200_vpss_mdy.....	944
1.9.20.7.12 hi_v200_vpss_tfy.....	945
1.9.20.7.13 hi_v200_vpss_nr_c.....	947
1.9.20.7.14 hi_vpss_nrx_param_manual_v3.....	947
1.9.20.7.15 hi_vpss_nrx_param_auto_v3.....	948
1.9.20.7.16 hi_vpss_nrx_param_v3.....	948
1.9.20.7.17 hi_vpss_grp_nrx_param.....	949
1.9.20.7.18 hi_vpss_grp_param.....	949
1.9.20.7.19 hi_fisheye_cfg.....	950
1.9.20.7.20 hi_vpss_chn_mode.....	950
1.9.20.7.21 hi_vpss_chn_attr.....	951
1.9.20.7.22 hi_fisheye_correction_attr.....	953
1.9.20.7.23 hi_fisheye_mount_mode.....	954
1.9.20.7.24 hi_fisheye_view_mode.....	954
1.9.20.7.25 hi_fisheye_rgn_attr.....	955
1.9.20.7.26 hi_fisheye_attr.....	956
1.9.20.8 VO 视频输出.....	957
1.9.20.8.1 hi_vo_dev.....	957
1.9.20.8.2 hi_vo_layer.....	958
1.9.20.8.3 hi_vo_chn.....	958
1.9.20.8.4 hi_vo_intf_type.....	959
1.9.20.8.5 hi_vo_intf_sync.....	959
1.9.20.8.6 hi_vo_sync_info.....	963
1.9.20.8.7 hi_vo_partition_mode.....	964
1.9.20.8.8 hi_vo_pub_attr.....	965
1.9.20.8.9 hi_vo_video_layer_attr.....	965
1.9.20.8.10 hi_vo_chn_attr.....	966
1.9.20.8.11 hi_vo_chn_param.....	967
1.9.20.9 TDE 图形绘制.....	967
1.9.20.9.1 hi_tde_color_format.....	967
1.9.20.9.2 hi_tde_surface.....	971
1.9.20.9.3 hi_tde_rect.....	972
1.9.20.9.4 hi_tde_none_src.....	973
1.9.20.9.5 hi_tde_single_src.....	973
1.9.20.9.6 hi_tde_double_src.....	974
1.9.20.9.7 hi_tde_fill_color.....	974
1.9.20.9.8 hi_tde_blend_cmd.....	975
1.9.20.9.9 hi_tde_blend_mode.....	976
1.9.20.9.10 hi_tde_rop_mode.....	978

1.9.20.9.11 hi_tde_clip_mode.....	979
1.9.20.9.12 hi_tde_colorkey_mode.....	980
1.9.20.9.13 hi_tde_colorkey_component.....	980
1.9.20.9.14 hi_tde_colorkey.....	981
1.9.20.9.15 hi_tde_blend_opt.....	982
1.9.20.9.16 hi_tde_alpha_blending.....	983
1.9.20.9.17 hi_tde_out_alpha_from.....	983
1.9.20.9.18 hi_tde_csc_opt.....	984
1.9.20.9.19 hi_tde_pattern_fill_opt.....	985
1.9.20.10 HIFB 叠加图形层管理.....	985
1.9.20.10.1 fb_var_screeninfo.....	986
1.9.20.10.2 fb_fix_screeninfo.....	988
1.9.20.10.3 fb_cmap.....	989
1.9.20.10.4 hi_fb_point.....	990
1.9.20.10.5 hi_fb_alpha.....	990
1.9.20.10.6 fb_bitfield.....	991
1.9.20.11 HDMI 外设.....	991
1.9.20.11.1 hi_hdmi_id.....	991
1.9.20.11.2 hi_hdmi_video_format.....	992
1.9.20.11.3 hi_hdmi_deep_color.....	995
1.9.20.11.4 hi_hdmi_sample_rate.....	996
1.9.20.11.5 hi_hdmi_bit_depth.....	997
1.9.20.11.6 hi_hdmi_attr.....	998
1.9.20.11.7 hi_hdmi_infoframe_type.....	999
1.9.20.11.8 hi_hdmi_color_space.....	999
1.9.20.11.9 hi_hdmi_bar_info.....	1000
1.9.20.11.10 hi_hdmi_scan_info.....	1001
1.9.20.11.11 hi_hdmi_colorimetry.....	1001
1.9.20.11.12 hi_hdmi_ex_colorimetry.....	1002
1.9.20.11.13 hi_pic_aspect_ratio.....	1003
1.9.20.11.14 hi_hdmi_active_aspect_ratio.....	1004
1.9.20.11.15 hi_hdmi_pic_scaline.....	1005
1.9.20.11.16 hi_hdmi_rgb_quant_range.....	1005
1.9.20.11.17 hi_hdmi_pixel_repetition.....	1006
1.9.20.11.18 hi_hdmi_content_type.....	1007
1.9.20.11.19 hi_hdmi_ycc_quant_range.....	1008
1.9.20.11.20 hi_hdmi_avi_infoframe.....	1008
1.9.20.11.21 hi_hdmi_audio_chn_cnt.....	1010
1.9.20.11.22 hi_hdmi_coding_type.....	1010
1.9.20.11.23 hi_hdmi_audio_sample_size.....	1012
1.9.20.11.24 hi_hdmi_audio_sample_freq.....	1013
1.9.20.11.25 hi_hdmi_level_shift_val.....	1013

1.9.20.11.26 hi_hdmi_lfe_playback_level.....	1015
1.9.20.11.27 hi_hdmi_audio_infoframe.....	1015
1.9.20.11.28 hi_hdmi_vendorspec_infoframe.....	1016
1.9.20.11.29 hi_hdmi_infoframe_unit.....	1017
1.9.20.11.30 hi_hdmi_infoframe.....	1017
1.9.20.11.31 hi_hdmi_sink_capability.....	1018
1.9.20.11.32 hi_hdmi_event_type.....	1018
1.9.20.11.33 hi_hdmi_callback.....	1019
1.9.20.11.34 hi_hdmi_callback_func.....	1019
1.9.20.12 VPC 图像处理.....	1020
1.9.20.12.1 hi_vpc_chn.....	1020
1.9.20.12.2 hi_vpc_pic_info.....	1020
1.9.20.12.3 hi_vpc_crop_region.....	1021
1.9.20.12.4 hi_vpc_resize_info.....	1021
1.9.20.12.5 hi_vpc_crop_region_info.....	1022
1.9.20.12.6 hi_vpc_crop_resize_region.....	1023
1.9.20.12.7 hi_vpc_crop_resize_paste_region.....	1023
1.9.20.12.8 hi_vpc_bord_type.....	1024
1.9.20.12.9 hi_vpc_make_border_info.....	1025
1.9.20.12.10 hi_vpc_histogram_config.....	1025
1.9.20.12.11 hi_vpc_lut_remap.....	1026
1.9.20.12.12 hi_vpc_scalar.....	1026
1.9.20.12.13 hi_vpc_chn_attr.....	1027
1.9.20.12.14 hi_vpc_crop_resize_border_region.....	1028
1.9.20.12.15 hi_csc_conf.....	1028
1.9.20.12.16 hi_roundview_stitching_param.....	1029
1.9.20.12.17 hi_roundview_stitching_pic_param.....	1029
1.9.20.12.18 hi_stiching_ipm_param.....	1030
1.9.20.12.19 hi_stiching_ipm_table.....	1031
1.9.20.12.20 hi_stitching_gain_param.....	1032
1.9.20.12.21 hi_stitching_histogram_type.....	1033
1.9.20.12.22 hi_stitching_gain_type.....	1033
1.9.20.12.23 hi_stitching_histogram_param.....	1034
1.9.20.12.24 hi_vpc_crop_resize_resize_paste_region.....	1034
1.9.20.12.25 hi_vpc_workspace_param.....	1035
1.9.20.12.26 hi_workspace_func.....	1036
1.9.20.12.27 hi_remap_lut.....	1036
1.9.20.12.28 hi_point_pair_info.....	1037
1.9.20.12.29 hi_float_point.....	1037
1.9.20.12.30 hi_map_param.....	1038
1.9.20.12.31 hi_warp_transform_param.....	1039
1.9.20.12.32 hi_transform_config.....	1039

1.9.20.12.33 hi_morph_shapes.....	1040
1.9.20.12.34 hi_rotation.....	1040
1.9.20.12.35 hi_blk_size.....	1041
1.9.20.12.36 hi_median_blur_param.....	1042
1.9.20.12.37 hi_median_blur_config.....	1042
1.9.20.12.38 hi_blur_param.....	1043
1.9.20.12.39 hi_blur_config.....	1043
1.9.20.12.40 hi_gaussian_blur_param.....	1044
1.9.20.12.41 hi_gaussian_blur_config.....	1044
1.9.20.12.42 hi_filter_2d_param.....	1045
1.9.20.12.43 hi_filter_2d_config.....	1045
1.9.20.12.44 hi_mosaic_param.....	1046
1.9.20.12.45 hi_rotate_param.....	1047
1.9.20.12.46 hi_mosaic.....	1047
1.9.20.12.47 hi_cover_type.....	1048
1.9.20.12.48 hi_cover_param.....	1048
1.9.20.12.49 hi_cover.....	1049
1.9.20.12.50 hi_quad_cover.....	1049
1.9.20.12.51 hi_line_param.....	1050
1.9.20.12.52 hi_line.....	1050
1.9.20.12.53 hi_osd_param.....	1051
1.9.20.12.54 hi_osd.....	1051
1.9.20.12.55 hi_osd_inverted_color.....	1052
1.9.20.12.56 hi_transform_matrix.....	1053
1.9.20.12.57 hi_matrix_type.....	1053
1.9.20.12.58 hi_flip_param.....	1054
1.9.20.12.59 hi_flip_mode.....	1054
1.9.20.12.60 hi_opt_attr.....	1055
1.9.20.12.61 hi_opt_attr_type.....	1055
1.9.20.13 VDEC 视频/JPEGD 图像解码.....	1056
1.9.20.13.1 hi_vdec_chn.....	1056
1.9.20.13.2 hi_vdec_send_mode.....	1056
1.9.20.13.3 hi_vdec_frame_type.....	1057
1.9.20.13.4 hi_quick_mark_mode.....	1057
1.9.20.13.5 hi_vdec_pic_info.....	1058
1.9.20.13.6 hi_vdec_video_attr.....	1061
1.9.20.13.7 hi_vdec_chn_attr.....	1062
1.9.20.13.8 hi_h264_protocol_param.....	1064
1.9.20.13.9 hi_h265_protocol_param.....	1065
1.9.20.13.10 hi_vdec_protocol_param.....	1065
1.9.20.13.11 hi_video_dec_mode.....	1066
1.9.20.13.12 hi_video_out_order.....	1066

1.9.20.13.13 hi_vdec_video_param.....	1067
1.9.20.13.14 hi_vdec_pic_param.....	1068
1.9.20.13.15 hi_vdec_chn_param.....	1068
1.9.20.13.16 hi_vdec_dec_err.....	1069
1.9.20.13.17 hi_vdec_chn_status.....	1070
1.9.20.13.18 hi_vdec_stream.....	1071
1.9.20.13.19 hi_vdec_supplement_info.....	1072
1.9.20.13.20 hi_vdec_video_supplement_info.....	1072
1.9.20.13.21 hi_jpegd_precision_mode.....	1072
1.9.20.14 VENC 视频/JPEGE 图像编码.....	1073
1.9.20.14.1 hi_venc_chn.....	1073
1.9.20.14.2 hi_venc_stream.....	1073
1.9.20.14.3 hi_venc_pack.....	1074
1.9.20.14.4 hi_venc_chn_attr.....	1075
1.9.20.14.5 hi_venc_attr.....	1076
1.9.20.14.6 hi_venc_rc_attr.....	1079
1.9.20.14.7 hi_venc_rc_mode.....	1081
1.9.20.14.8 hi_venc_h264_cbr.....	1083
1.9.20.14.9 hi_venc_h264_vbr.....	1084
1.9.20.14.10 hi_venc_h264_avbr.....	1085
1.9.20.14.11 hi_venc_h264_qvbr.....	1086
1.9.20.14.12 hi_venc_h264_cvbr.....	1087
1.9.20.14.13 hi_venc_h264_fixqp.....	1089
1.9.20.14.14 hi_venc_h264_qpmap.....	1090
1.9.20.14.15 hi_venc_mjpeg_cbr.....	1090
1.9.20.14.16 hi_venc_mjpeg_vbr.....	1090
1.9.20.14.17 hi_venc_mjpeg_fixqp.....	1091
1.9.20.14.18 hi_venc_h265_cbr.....	1091
1.9.20.14.19 hi_venc_h265_vbr.....	1092
1.9.20.14.20 hi_venc_h265_avbr.....	1093
1.9.20.14.21 hi_venc_h265_qvbr.....	1094
1.9.20.14.22 hi_venc_h265_cvbr.....	1095
1.9.20.14.23 hi_venc_h265_fixqp.....	1097
1.9.20.14.24 hi_venc_h265_qpmap.....	1098
1.9.20.14.25 hi_venc_gop_attr.....	1098
1.9.20.14.26 hi_venc_gop_mode.....	1099
1.9.20.14.27 hi_venc_gop_normal_p.....	1100
1.9.20.14.28 hi_venc_gop_dual_p.....	1100
1.9.20.14.29 hi_venc_gop_smart_p.....	1101
1.9.20.14.30 hi_venc_gop_adv_smart_p.....	1101
1.9.20.14.31 hi_venc_gop_bipred_b.....	1102
1.9.20.14.32 hi_venc_start_param.....	1102

1.9.20.14.33 hi_venc_chn_status.....	1102
1.9.20.14.34 hi_venc_h264_stream_info.....	1103
1.9.20.14.35 hi_venc_ref_type.....	1104
1.9.20.14.36 hi_venc_jpeg_stream_info.....	1105
1.9.20.14.37 hi_venc_h265_stream_info.....	1105
1.9.20.14.38 hi_venc_prores_stream_info.....	1107
1.9.20.14.39 hi_venc_mod_param.....	1107
1.9.20.14.40 hi_venc_mod_type.....	1108
1.9.20.14.41 hi_venc_venc_mod_param.....	1108
1.9.20.14.42 hi_venc_h264_mod_param.....	1109
1.9.20.14.43 hi_venc_h265_mod_param.....	1110
1.9.20.14.44 hi_venc_jpeg_mod_param.....	1111
1.9.20.14.45 hi_venc_jpeg_param.....	1112
1.9.20.14.46 hi_venc_h264_adv_stream_info.....	1113
1.9.20.14.47 hi_venc_h265_adv_stream_info.....	1114
1.9.20.14.48 hi_venc_data_type.....	1115
1.9.20.14.49 hi_venc_h264_nalu_type.....	1116
1.9.20.14.50 hi_venc_jpege_pack_type.....	1116
1.9.20.14.51 hi_venc_h265_nalu_type.....	1117
1.9.20.14.52 hi_venc_prores_pack_type.....	1117
1.9.20.14.53 hi_venc_chn_param.....	1117
1.9.20.14.54 hi_venc_stream_info.....	1118
1.9.20.14.55 hi_venc_pack_info.....	1119
1.9.20.14.56 hi_frame_rate_ctrl.....	1120
1.9.20.14.57 hi_venc_rc_qpmap_mode.....	1120
1.9.20.14.58 hi_venc_sse_info.....	1121
1.9.20.14.59 hi_venc_h264_attr.....	1121
1.9.20.14.60 hi_venc_h265_attr.....	1122
1.9.20.14.61 hi_venc_jpeg_attr.....	1123
1.9.20.14.62 hi_venc_prores_attr.....	1123
1.9.20.14.63 hi_venc_scene_mode.....	1124
1.9.20.14.64 hi_venc_rc_param.....	1124
1.9.20.14.65 hi_venc_h264_cbr_param.....	1127
1.9.20.14.66 hi_venc_h264_vbr_param.....	1128
1.9.20.14.67 hi_venc_h264_avbr_param.....	1129
1.9.20.14.68 hi_venc_h264_qvbr_param.....	1131
1.9.20.14.69 hi_venc_h264_cvbr_param.....	1133
1.9.20.14.70 hi_venc_h265_vbr_param.....	1135
1.9.20.14.71 hi_venc_h265_avbr_param.....	1136
1.9.20.14.72 hi_venc_h265_qvbr_param.....	1138
1.9.20.14.73 hi_venc_h265_cvbr_param.....	1140
1.9.20.14.74 hi_venc_h265_cbr_param.....	1141

1.9.20.14.75 hi_venc_scene_chg_detect.....	1142
1.9.20.14.76 hi_venc_huffman_dc_table.....	1143
1.9.20.14.77 hi_venc_huffman_ac_table.....	1143
1.9.20.14.78 hi_venc_jpeg_huffman_param.....	1144
1.9.20.14.79 hi_venc_mpf_cfg.....	1144
1.9.20.14.80 hi_venc_mjpeg_cbr_param.....	1145
1.9.20.14.81 hi_venc_mjpeg_vbr_param.....	1145
1.9.20.14.82 hi_venc_h264_ref_slice_type.....	1145
1.9.20.14.83 hi_venc_roi_attr.....	1145
1.9.20.14.84 hi_venc_ref_param.....	1146
1.9.20.14.85 hi_venc_h264_vui.....	1147
1.9.20.14.86 hi_venc_h265_vui.....	1147
1.9.20.14.87 hi_venc_vui_aspect_ratio.....	1148
1.9.20.14.88 hi_venc_vui_h264_time_info.....	1149
1.9.20.14.89 hi_venc_vui_h265_time_info.....	1149
1.9.20.14.90 hi_venc_vui_video_signal.....	1150
1.9.20.14.91 hi_venc_vui_bitstream_restric.....	1151
1.9.20.14.92 hi_venc_cu_prediction.....	1152
1.9.20.14.93 hi_venc_intra_refresh.....	1153
1.9.20.14.94 hi_venc_intra_refresh_mode.....	1155
1.9.20.14.95 hi_venc_slice_split.....	1155
1.9.20.14.96 hi_venc_frame_type.....	1156
1.9.20.14.97 hi_user_rc_info.....	1156
1.9.20.14.98 hi_user_roi_map.....	1157
1.9.20.14.99 hi_user_frame_info.....	1157
1.9.20.14.100 hi_venc_streamcopy_param.....	1158
1.9.20.15 PNGD 图像解码.....	1158
1.9.20.15.1 hi_pngd_chn.....	1158
1.9.20.15.2 hi_pngd_chn_attr.....	1159
1.9.21 返回码列表.....	1159
1.9.21.1 公共返回码.....	1159
1.9.21.2 音频相关返回码.....	1160
1.9.21.3 ISP 系统控制返回码.....	1164
1.9.21.4 VI 视频输入返回码.....	1164
1.9.21.5 区域管理返回码.....	1165
1.9.21.6 VPSS 视频处理子系统返回码.....	1166
1.9.21.7 VO 视频输出返回码.....	1166
1.9.21.8 TDE 图形绘制返回码.....	1167
1.9.21.9 HDMI 外设返回码.....	1168
1.9.21.10 VPC 图像处理返回码.....	1169
1.9.21.11 VDEC 视频解码/JPEGD 图片解码返回码.....	1170
1.9.21.12 VENC 视频编码/JPEGE 图片编码返回码.....	1171

1.9.21.13 PNGD 图像解码返回码.....	1173
1.10 Profiling 数据采集.....	1174
1.10.1 Profiling AscendCL API (通过 Profiling AscendCL API 采集并落盘性能数据)	1174
1.10.1.1 功能介绍.....	1174
1.10.1.2 aclprofInit.....	1175
1.10.1.3 aclprofSetConfig.....	1176
1.10.1.4 aclprofStart.....	1178
1.10.1.5 aclprofStop.....	1179
1.10.1.6 aclprofFinalize.....	1180
1.10.2 Profiling AscendCL API for Extension (Profiling AscendCL API 扩展接口)	1180
1.10.2.1 功能介绍.....	1180
1.10.2.2 aclprofCreateStamp.....	1181
1.10.2.3 aclprofSetStampTraceMessage.....	1181
1.10.2.4 aclprofMark.....	1182
1.10.2.5 aclprofMarkEx.....	1183
1.10.2.6 aclprofPush.....	1184
1.10.2.7 aclprofPop.....	1184
1.10.2.8 aclprofRangeStart.....	1185
1.10.2.9 aclprofRangeStop.....	1186
1.10.2.10 aclprofDestroyStamp.....	1186
1.10.3 Profiling AscendCL API for Subscription (订阅算子信息的 Profiling AscendCL API)	1187
1.10.3.1 功能介绍.....	1187
1.10.3.2 aclprofModelSubscribe.....	1188
1.10.3.3 aclprofModelUnSubscribe.....	1189
1.10.3.4 aclprofGetOpDescSize.....	1189
1.10.3.5 aclprofGetOpNum.....	1190
1.10.3.6 aclprofGetOpTypeLen.....	1191
1.10.3.7 aclprofGetOpType.....	1191
1.10.3.8 aclprofGetOpNameLen.....	1192
1.10.3.9 aclprofGetOpName.....	1193
1.10.3.10 aclprofGetOpStart.....	1194
1.10.3.11 aclprofGetOpEnd.....	1194
1.10.3.12 aclprofGetOpDuration.....	1195
1.10.3.13 aclprofGetModelId.....	1196
1.10.4 Profiling AscendCL API (PyTorch 场景标记迭代时间)	1197
1.10.4.1 aclprofGetStepTimestamp.....	1197
1.11 数据传输.....	1197
1.11.1 Tensor 数据传输.....	1197
1.11.1.1 acltdtCreateChannel.....	1197
1.11.1.2 acltdtSendTensor.....	1198
1.11.1.3 acltdtReceiveTensor.....	1199
1.11.1.4 acltdtStopChannel.....	1200

1.11.1.5 acltdtDestroyChannel.....	1201
1.11.1.6 acltdtCreateChannelWithCapacity.....	1201
1.11.1.7 acltdtQueryChannelSize.....	1202
1.11.1.8 acltdtGetSliceInfoFromItem.....	1203
1.11.1.9 acltdtCleanChannel.....	1204
1.11.2 共享队列管理.....	1204
1.11.2.1 功能及约束说明.....	1204
1.11.2.2 acltdtCreateQueue.....	1204
1.11.2.3 acltdtDestroyQueue.....	1205
1.11.2.4 acltdtEnqueueData.....	1206
1.11.2.5 acltdtDequeueData.....	1207
1.11.2.6 acltdtEnqueue.....	1208
1.11.2.7 acltdtDequeue.....	1208
1.11.2.8 acltdtBindQueueRoutes.....	1209
1.11.2.9 acltdtUnbindQueueRoutes.....	1210
1.11.2.10 acltdtQueryQueueRoutes.....	1211
1.11.2.11 acltdtGrantQueue.....	1212
1.11.2.12 acltdtAttachQueue.....	1213
1.11.3 共享 Buffer 管理.....	1214
1.11.3.1 acltdtAllocBuf.....	1214
1.11.3.2 acltdtFreeBuf.....	1215
1.11.3.3 acltdtGetBufData.....	1216
1.11.3.4 acltdtSetBufUserData.....	1217
1.11.3.5 acltdtGetBufUserData.....	1218
1.11.3.6 acltdtSetBufDataLen.....	1218
1.11.3.7 acltdtGetBufDataLen.....	1219
1.11.3.8 acltdtCopyBufRef.....	1220
1.11.3.9 acltdtAppendBufChain.....	1221
1.11.3.10 acltdtGetBufChainNum.....	1222
1.11.3.11 acltdtGetBufFromChain.....	1222
1.12 特征向量检索.....	1223
1.12.1 总体说明.....	1223
1.12.2 aclfvInit.....	1224
1.12.3 aclfvRelease.....	1225
1.12.4 aclfvRepoAdd.....	1226
1.12.5 aclfvRepoDel.....	1227
1.12.6 aclfvDel.....	1228
1.12.7 aclfvModify.....	1229
1.12.8 aclfvSearch.....	1230
1.13 数据类型及其操作接口.....	1231
1.13.1 aclError.....	1231
1.13.2 aclDataType.....	1252

1.13.3 aclFloat16.....	1252
1.13.4 aclFormat.....	1252
1.13.5 aclrtMemMallocPolicy.....	1253
1.13.6 aclvppPixelFormat.....	1255
1.13.7 aclvppStreamFormat.....	1256
1.13.8 aclvppJpegFormat.....	1256
1.13.9 aclvppCscMatrix.....	1256
1.13.10 aclrtContext.....	1258
1.13.11 aclrtStream.....	1258
1.13.12 aclrtEvent.....	1258
1.13.13 aclrtEventStatus.....	1258
1.13.14 aclrtRunMode.....	1258
1.13.15 aclTransType.....	1258
1.13.16 aclComputeType.....	1258
1.13.17 aclfvSearchType.....	1259
1.13.18 aclAippInputFormat.....	1259
1.13.19 aclAippInfo.....	1259
1.13.20 aclAippDims.....	1260
1.13.21 aclmdlIODims.....	1260
1.13.22 aclrtGroupAttr.....	1260
1.13.23 aclprofAicoreMetrics.....	1261
1.13.24 aclvppBorderType.....	1261
1.13.25 aclmdlInputAippType.....	1261
1.13.26 acltdtTensorType.....	1261
1.13.27 aclCompileOpt.....	1262
1.13.28 aclvppChannelDescParamType.....	1276
1.13.29 aclvdecChannelDescParamType.....	1277
1.13.30 aclvencChannelDescParamType.....	1280
1.13.31 aclmdlConfigAttr.....	1282
1.13.32 aclMemType.....	1286
1.13.33 aclOpCompileFlag.....	1286
1.13.34 aclrtEventWaitStatus.....	1286
1.13.35 aclprofStepTag.....	1286
1.13.36 acltdtBuf.....	1286
1.13.37 acltdtQueueAttrType.....	1286
1.13.38 acltdtQueueRouteParamType.....	1287
1.13.39 acltdtQueueRouteQueryMode.....	1287
1.13.40 acltdtQueueRouteQueryInfoParamType.....	1287
1.13.41 aclvppChannelMode.....	1287
1.13.42 aclopEngineType.....	1287
1.13.43 aclopCompileType.....	1287
1.13.44 aclrtEventRecordedStatus.....	1287

1.13.45	aclrtFloatOverflowMode.....	1287
1.13.46	acldvppJpegdPrecisionMode.....	1288
1.13.47	aclmdlExecConfigAttr.....	1288
1.13.48	aclrtStreamConfigAttr.....	1289
1.13.49	aclSysParamOpt.....	1290
1.13.50	aclrtUtilizationInfo.....	1291
1.13.51	aclrtStreamStatus.....	1291
1.13.52	aclrtMemHandleType.....	1291
1.13.53	aclrtMemAllocationType.....	1291
1.13.54	aclrtMemAttr.....	1291
1.13.55	aclrtMemLocationType.....	1291
1.13.56	aclrtMemLocation.....	1291
1.13.57	aclrtPhysicalMemProp.....	1291
1.13.58	aclrtDrvMemHandle.....	1292
1.13.59	aclrtDeviceStatus.....	1292
1.13.60	aclrtMemGranularityOptions.....	1292
1.13.61	aclCannAttr.....	1292
1.13.62	aclDeviceInfo.....	1292
1.13.63	aclmdlExeOMDesc.....	1292
1.13.64	aclrtMemUcelInfo.....	1293
1.13.65	aclmdlIODimsRange.....	1293
1.13.66	aclrtCmoType.....	1294
1.13.67	aclrtLastErrLevel.....	1294
1.13.68	aclmdlAIPP.....	1294
1.13.68.1	aclmdlCreateAIPP.....	1294
1.13.68.2	aclmdlSetAIPPCscParams.....	1294
1.13.68.3	aclmdlSetAIPPInputFormat.....	1297
1.13.68.4	aclmdlSetAIPPRbuvsSwapSwitch.....	1297
1.13.68.5	aclmdlSetAIPPAxSwapSwitch.....	1298
1.13.68.6	aclmdlSetAIPPSrcImageSize.....	1298
1.13.68.7	aclmdlSetAIPPScfParams.....	1299
1.13.68.8	aclmdlSetAIPPCropParams.....	1301
1.13.68.9	aclmdlSetAIPPPaddingParams.....	1302
1.13.68.10	aclmdlSetAIPPDtcPixelMean.....	1303
1.13.68.11	aclmdlSetAIPPDtcPixelMin.....	1304
1.13.68.12	aclmdlSetAIPPPixelVarReci.....	1305
1.13.68.13	aclmdlGetAippDataSize.....	1306
1.13.68.14	aclmdlDestroyAIPP.....	1307
1.13.69	aclopHandle.....	1307
1.13.69.1	aclopCreateHandle.....	1307
1.13.69.2	aclopDestroyHandle.....	1308
1.13.70	aclDataBuffer.....	1309

1.13.70.1	aclCreateDataBuffer.....	1309
1.13.70.2	aclDestroyDataBuffer.....	1309
1.13.70.3	aclGetDataBufferAddr.....	1310
1.13.70.4	aclGetDataBufferSize (废弃)	1310
1.13.70.5	aclGetDataBufferSizeV2.....	1311
1.13.70.6	aclUpdateDataBuffer.....	1311
1.13.71	aclmdlDataset.....	1312
1.13.71.1	aclmdlCreateDataset.....	1312
1.13.71.2	aclmdlDestroyDataset.....	1313
1.13.71.3	aclmdlAddDatasetBuffer.....	1313
1.13.71.4	aclmdlGetDatasetNumBuffers.....	1314
1.13.71.5	aclmdlGetDatasetBuffer.....	1314
1.13.71.6	aclmdlSetDatasetTensorDesc.....	1315
1.13.71.7	aclmdlGetDatasetTensorDesc.....	1316
1.13.72	aclmdlDesc.....	1317
1.13.72.1	aclmdlCreateDesc.....	1317
1.13.72.2	aclmdlDestroyDesc.....	1317
1.13.72.3	aclmdlGetDesc.....	1317
1.13.72.4	aclmdlGetNumInputs.....	1318
1.13.72.5	aclmdlGetNumOutputs.....	1319
1.13.72.6	aclmdlGetInputSizeByIndex.....	1319
1.13.72.7	aclmdlGetOutputSizeByIndex.....	1320
1.13.72.8	aclmdlGetInputDims.....	1321
1.13.72.9	aclmdlGetInputDimsV2.....	1323
1.13.72.10	aclmdlGetOutputDims.....	1325
1.13.72.11	aclmdlGetInputNameByIndex.....	1325
1.13.72.12	aclmdlGetOutputNameByIndex.....	1326
1.13.72.13	aclmdlGetInputFormat.....	1327
1.13.72.14	aclmdlGetOutputFormat.....	1328
1.13.72.15	aclmdlGetInputDataType.....	1328
1.13.72.16	aclmdlGetOutputDataType.....	1329
1.13.72.17	aclmdlGetInputIndexByName.....	1329
1.13.72.18	aclmdlGetOutputIndexByName.....	1330
1.13.72.19	aclmdlGetDynamicBatch.....	1330
1.13.72.20	aclmdlGetDynamicHW.....	1331
1.13.72.21	aclmdlGetCurOutputDims.....	1332
1.13.72.22	aclmdlGetInputDynamicGearCount.....	1333
1.13.72.23	aclmdlGetInputDynamicDims.....	1334
1.13.72.24	aclmdlGetTensorRealName.....	1335
1.13.72.25	aclmdlGetOpAttr.....	1336
1.13.72.26	aclmdlGetDescFromFile.....	1337
1.13.72.27	aclmdlGetDescFromMem.....	1337

1.13.72.28 aclmdlGetInputDimsRange.....	1338
1.13.73 aclTensorDesc.....	1339
1.13.73.1 aclCreateTensorDesc.....	1339
1.13.73.2 aclDestroyTensorDesc.....	1340
1.13.73.3 aclGetTensorDescType.....	1341
1.13.73.4 aclGetTensorDescFormat.....	1341
1.13.73.5 aclGetTensorDescSize.....	1342
1.13.73.6 aclGetTensorDescElementCount.....	1342
1.13.73.7 aclGetTensorDescNumDims.....	1343
1.13.73.8 aclGetTensorDescDim (废弃)	1343
1.13.73.9 aclGetTensorDescDimV2.....	1344
1.13.73.10 aclGetTensorDescDimRange.....	1345
1.13.73.11 aclSetTensorDescName.....	1346
1.13.73.12 aclGetTensorDescName.....	1346
1.13.73.13 aclTransTensorDescFormat.....	1347
1.13.73.14 aclSetTensorStorageShape (废弃)	1347
1.13.73.15 aclSetTensorStorageFormat (废弃)	1348
1.13.73.16 aclSetTensorOriginShape.....	1349
1.13.73.17 aclSetTensorOriginFormat.....	1349
1.13.73.18 aclSetTensorShape.....	1350
1.13.73.19 aclSetTensorFormat.....	1351
1.13.73.20 aclSetTensorShapeRange.....	1352
1.13.73.21 aclSetTensorDynamicInput.....	1352
1.13.73.22 aclGetTensorDescByIndex.....	1353
1.13.73.23 aclGetTensorDescAddress.....	1354
1.13.73.24 aclSetTensorConst.....	1354
1.13.73.25 aclSetTensorPlaceMent.....	1355
1.13.73.26 aclSetTensorValueRange.....	1355
1.13.74 aclopAttr.....	1356
1.13.74.1 aclopCreateAttr.....	1356
1.13.74.2 aclopDestroyAttr.....	1356
1.13.74.3 aclopSetAttrBool.....	1357
1.13.74.4 aclopSetAttrInt.....	1357
1.13.74.5 aclopSetAttrFloat.....	1358
1.13.74.6 aclopSetAttrString.....	1358
1.13.74.7 aclopSetAttrListBool.....	1359
1.13.74.8 aclopSetAttrListInt.....	1360
1.13.74.9 aclopSetAttrListFloat.....	1360
1.13.74.10 aclopSetAttrListString.....	1361
1.13.74.11 aclopSetAttrListListInt.....	1361
1.13.74.12 aclopSetAttrDataType.....	1362
1.13.74.13 aclopSetAttrListDataType.....	1363

1.13.75 acldvppChannelDesc.....	1363
1.13.75.1 acldvppCreateChannelDesc.....	1363
1.13.75.2 acldvppDestroyChannelDesc.....	1364
1.13.75.3 acldvppGetChannelDescChannelId.....	1364
1.13.75.4 acldvppSetChannelDescMode.....	1365
1.13.75.5 acldvppSetChannelDescParam.....	1366
1.13.75.6 acldvppGetChannelDescParam.....	1366
1.13.76 acldvppPicDesc.....	1367
1.13.76.1 acldvppCreatePicDesc.....	1367
1.13.76.2 acldvppSetPicDesc 系列接口.....	1368
1.13.76.3 acldvppGetPicDesc 系列接口.....	1369
1.13.76.4 acldvppDestroyPicDesc.....	1370
1.13.77 acldvppRoiConfig.....	1370
1.13.77.1 acldvppCreateRoiConfig.....	1370
1.13.77.2 acldvppSetRoiConfig 系列接口.....	1372
1.13.77.3 acldvppDestroyRoiConfig.....	1373
1.13.78 acldvppResizeConfig.....	1374
1.13.78.1 acldvppCreateResizeConfig.....	1374
1.13.78.2 acldvppSetResizeConfigInterpolation.....	1374
1.13.78.3 acldvppGetResizeConfigInterpolation.....	1377
1.13.78.4 acldvppDestroyResizeConfig.....	1378
1.13.79 acldvppJpegeConfig.....	1378
1.13.79.1 acldvppCreateJpegeConfig.....	1378
1.13.79.2 acldvppSetJpegeConfigLevel.....	1379
1.13.79.3 acldvppGetJpegeConfigLevel.....	1380
1.13.79.4 acldvppDestroyJpegeConfig.....	1380
1.13.80 aclvdecChannelDesc.....	1381
1.13.80.1 aclvdecCreateChannelDesc.....	1381
1.13.80.2 aclvdecSetChannelDesc 系列接口.....	1381
1.13.80.3 aclvdecGetChannelDesc 系列接口.....	1384
1.13.80.4 aclvdecSetChannelDescParam.....	1386
1.13.80.5 aclvdecGetChannelDescParam.....	1387
1.13.80.6 aclvdecDestroyChannelDesc.....	1388
1.13.81 acldvppStreamDesc.....	1388
1.13.81.1 acldvppCreateStreamDesc.....	1388
1.13.81.2 acldvppSetStreamDesc 系列接口.....	1389
1.13.81.3 acldvppGetStreamDesc 系列接口.....	1390
1.13.81.4 acldvppDestroyStreamDesc.....	1391
1.13.82 aclvdecFrameConfig.....	1391
1.13.82.1 aclvdecCreateFrameConfig.....	1391
1.13.82.2 aclvdecDestroyFrameConfig.....	1392
1.13.83 acldvppBatchPicDesc.....	1392

1.13.83.1 acldvppCreateBatchPicDesc.....	1392
1.13.83.2 acldvppGetPicDesc.....	1393
1.13.83.3 acldvppDestroyBatchPicDesc.....	1393
1.13.84 aclvencChannelDesc.....	1394
1.13.84.1 aclvencCreateChannelDesc.....	1394
1.13.84.2 aclvencSetChannelDesc 系列接口.....	1394
1.13.84.3 aclvencGetChannelDesc 系列接口.....	1397
1.13.84.4 aclvencSetChannelDescParam.....	1399
1.13.84.5 aclvencGetChannelDescParam.....	1399
1.13.84.6 aclvencDestroyChannelDesc.....	1400
1.13.85 aclvencFrameConfig.....	1401
1.13.85.1 aclvencCreateFrameConfig.....	1401
1.13.85.2 aclvencSetFrameConfig 系列接口.....	1401
1.13.85.3 aclvencGetFrameConfig 系列接口.....	1402
1.13.85.4 aclvencDestroyFrameConfig.....	1403
1.13.86 acldvppLutMap.....	1404
1.13.86.1 acldvppCreateLutMap.....	1404
1.13.86.2 acldvppDestroyLutMap.....	1404
1.13.86.3 acldvppGetLutMapDims.....	1405
1.13.86.4 acldvppGetLutMapData.....	1405
1.13.87 acldvppBorderConfig.....	1406
1.13.87.1 acldvppCreateBorderConfig.....	1406
1.13.87.2 acldvppSetBorderConfig 系列接口.....	1407
1.13.87.3 acldvppGetBorderConfig 系列接口.....	1408
1.13.87.4 acldvppDestroyBorderConfig.....	1409
1.13.88 acldvppHist.....	1410
1.13.88.1 acldvppCreateHist.....	1410
1.13.88.2 acldvppDestroyHist.....	1410
1.13.88.3 acldvppGetHistDims.....	1411
1.13.88.4 acldvppGetHistData.....	1411
1.13.88.5 acldvppGetHistRetCode.....	1412
1.13.88.6 acldvppClearHist.....	1413
1.13.89 aclfvRepoRange.....	1413
1.13.89.1 aclfvCreateRepoRange.....	1413
1.13.89.2 aclfvDestroyRepoRange.....	1414
1.13.90 aclfvFeatureInfo.....	1415
1.13.90.1 aclfvCreateFeatureInfo.....	1415
1.13.90.2 aclfvDestroyFeatureInfo.....	1416
1.13.91 aclfvQueryTable.....	1416
1.13.91.1 aclfvCreateQueryTable.....	1416
1.13.91.2 aclfvDestroyQueryTable.....	1417
1.13.92 aclfvSearchInput.....	1418

1.13.92.1 acldvCreateSearchInput.....	1418
1.13.92.2 acldvDestroySearchInput.....	1419
1.13.93 acldvSearchResult.....	1419
1.13.93.1 acldvCreateSearchResult.....	1419
1.13.93.2 acldvDestroySearchResult.....	1420
1.13.94 acldvInitPara.....	1421
1.13.94.1 acldvCreateInitPara.....	1421
1.13.94.2 acldvDestroyInitPara.....	1422
1.13.94.3 acldvSet1NTopNum.....	1422
1.13.94.4 acldvSetNMTopNum.....	1423
1.13.95 acldprofConfig.....	1424
1.13.95.1 acldprofCreateConfig.....	1424
1.13.95.2 acldprofDestroyConfig.....	1426
1.13.96 acldprofSubscribeConfig.....	1426
1.13.96.1 acldprofCreateSubscribeConfig.....	1426
1.13.96.2 acldprofDestroySubscribeConfig.....	1427
1.13.97 acldprofStepInfo.....	1428
1.13.97.1 acldprofCreateStepInfo.....	1428
1.13.97.2 acldprofDestroyStepInfo.....	1428
1.13.98 acldtdDataItem.....	1429
1.13.98.1 acldtdCreateDataItem.....	1429
1.13.98.2 acldtdDestroyDataItem.....	1430
1.13.98.3 acldtdGetTensorTypeFromItem.....	1431
1.13.98.4 acldtdGetDataFromItem.....	1431
1.13.98.5 acldtdGetDataAddrFromItem.....	1432
1.13.98.6 acldtdGetDataSizeFromItem.....	1432
1.13.98.7 acldtdGetDimNumFromItem.....	1433
1.13.98.8 acldtdGetDimsFromItem.....	1434
1.13.99 acldtdDataset.....	1434
1.13.99.1 acldtdCreateDataset.....	1434
1.13.99.2 acldtdDestroyDataset.....	1435
1.13.99.3 acldtdGetDataItem.....	1435
1.13.99.4 acldtdAddDataItem.....	1436
1.13.99.5 acldtdGetDatasetSize.....	1437
1.13.99.6 acldtdGetDatasetName.....	1437
1.13.100 acldmdlConfigHandle.....	1438
1.13.100.1 acldmdlCreateConfigHandle.....	1438
1.13.100.2 acldmdlDestroyConfigHandle.....	1438
1.13.101 acldtdQueueAttr.....	1439
1.13.101.1 acldtdCreateQueueAttr.....	1439
1.13.101.2 acldtdDestroyQueueAttr.....	1439
1.13.101.3 acldtdSetQueueAttr.....	1440

1.13.101.4 acltdtGetQueueAttr.....	1441
1.13.102 acltdtQueueRoute.....	1441
1.13.102.1 acltdtCreateQueueRoute.....	1441
1.13.102.2 acltdtDestroyQueueRoute.....	1442
1.13.102.3 acltdtGetQueueRouteParam.....	1442
1.13.103 acltdtQueueRouteList.....	1443
1.13.103.1 acltdtCreateQueueRouteList.....	1443
1.13.103.2 acltdtDestroyQueueRouteList.....	1444
1.13.103.3 acltdtAddQueueRoute.....	1444
1.13.103.4 acltdtGetQueueRoute.....	1445
1.13.103.5 acltdtGetQueueRouteNum.....	1445
1.13.104 acltdtQueueRouteQueryInfo.....	1446
1.13.104.1 acltdtCreateQueueRouteQueryInfo.....	1446
1.13.104.2 acltdtDestroyQueueRouteQueryInfo.....	1446
1.13.104.3 acltdtSetQueueRouteQueryInfo.....	1447
1.13.105 aclGraphDumpOption.....	1447
1.13.105.1 aclCreateGraphDumpOpt.....	1447
1.13.105.2 aclDestroyGraphDumpOpt.....	1448
1.13.106 aclmdlExecConfigHandle.....	1448
1.13.106.1 aclmdlCreateExecConfigHandle.....	1448
1.13.106.2 aclmdlDestroyExecConfigHandle.....	1449
1.13.107 aclrtStreamConfigHandle.....	1449
1.13.107.1 aclrtCreateStreamConfigHandle.....	1449
1.13.107.2 aclrtDestroyStreamConfigHandle.....	1450
1.13.108 aclrtAllocatorDesc.....	1450
1.13.108.1 aclrtAllocatorCreateDesc.....	1450
1.13.108.2 aclrtAllocatorDestroyDesc.....	1451
1.13.108.3 aclrtAllocatorSetObjToDesc.....	1451
1.13.108.4 aclrtAllocatorSetAllocFuncToDesc.....	1452
1.13.108.5 aclrtAllocatorSetAllocAdviseFuncToDesc.....	1453
1.13.108.6 aclrtAllocatorSetFreeFuncToDesc.....	1453
1.13.108.7 aclrtAllocatorSetGetAddrFromBlockFuncToDesc.....	1454
2 AscendCL API (Python)	1456
2.1 废弃接口/返回码列表.....	1456
2.2 同步&异步 API 说明.....	1458
2.3 系统配置.....	1458
2.3.1 函数: init.....	1458
2.3.2 函数: finalize.....	1465
2.3.3 函数: get_version.....	1465
2.3.4 函数: get_recent_err_msg.....	1466
2.3.5 函数: get_cann_attribute_list.....	1467
2.3.6 函数: get_cann_attribute.....	1467

2.3.7 函数: get_device_capability.....	1468
2.4 公共接口.....	1468
2.4.1 函数: float16_to_float.....	1468
2.4.2 函数: float_to_float16.....	1468
2.4.3 函数: create_data_buffer.....	1469
2.4.4 函数: destroy_data_buffer.....	1469
2.4.5 函数: get_data_buffer_addr.....	1470
2.4.6 函数: get_data_buffer_size.....	1470
2.4.7 函数: get_data_buffer_size_v2.....	1470
2.4.8 函数: update_data_buffer.....	1471
2.4.9 函数: data_type_size.....	1471
2.4.10 函数: create_tensor_desc.....	1472
2.4.11 函数: destroy_tensor_desc.....	1472
2.4.12 函数: get_tensor_desc_type.....	1472
2.4.13 函数: get_tensor_desc_format.....	1473
2.4.14 函数: get_tensor_desc_size.....	1473
2.4.15 函数: get_tensor_desc_element_count.....	1473
2.4.16 函数: get_tensor_desc_num_dims.....	1474
2.4.17 函数: get_tensor_desc_dim.....	1474
2.4.18 函数: get_tensor_desc_dim_v2.....	1475
2.4.19 函数: get_tensor_desc_dim_range.....	1475
2.4.20 函数: set_tensor_desc_name.....	1476
2.4.21 函数: get_tensor_desc_name.....	1476
2.4.22 函数: trans_tensor_desc_format.....	1477
2.4.23 函数: set_tensor_storage_shape.....	1477
2.4.24 函数: set_tensor_storage_format.....	1478
2.4.25 函数: set_tensor_origin_shape.....	1478
2.4.26 函数: set_tensor_origin_format.....	1479
2.4.27 函数: set_tensor_shape.....	1480
2.4.28 函数: set_tensor_format.....	1480
2.4.29 函数: set_tensor_shape_range.....	1481
2.4.30 函数: set_tensor_dynamic_input.....	1481
2.4.31 函数: get_tensor_desc_by_index.....	1482
2.4.32 函数: get_tensor_desc_address.....	1482
2.4.33 函数: set_tensor_const.....	1483
2.4.34 函数: set_tensor_place_ment.....	1483
2.4.35 函数: set_tensor_value_range.....	1484
2.5 日志管理.....	1484
2.5.1 函数: app_log.....	1484
2.6 Util 模块.....	1485
2.6.1 函数: numpy_to_ptr.....	1485
2.6.2 函数: numpy_contiguous_to_ptr.....	1486

2.6.3 函数: ptr_to_numpy..... 1487

2.6.4 函数: bytes_to_ptr..... 1488

2.6.5 函数: ptr_to_bytes..... 1489

2.6.6 thread..... 1489

2.6.6.1 函数: start_thread..... 1489

2.6.6.2 函数: stop_thread..... 1490

2.7 Runtime (rt) 1490

2.7.1 Device 管理..... 1490

2.7.1.1 函数: set_device..... 1490

2.7.1.2 函数: reset_device..... 1491

2.7.1.3 函数: get_device..... 1492

2.7.1.4 函数: get_run_mode..... 1492

2.7.1.5 函数: set_ts_device..... 1492

2.7.1.6 函数: get_device_count..... 1493

2.7.1.7 函数: get_device_utilization_rate..... 1493

2.7.1.8 函数: query_device_status..... 1494

2.7.1.9 函数: synchronize_device..... 1494

2.7.1.10 函数: get_soc_name..... 1495

2.7.1.11 函数: set_device_sat_mode..... 1495

2.7.1.12 函数: get_device_sat_mode..... 1496

2.7.2 Context 管理..... 1496

2.7.2.1 函数: create_context..... 1496

2.7.2.2 函数: destroy_context..... 1497

2.7.2.3 函数: set_context..... 1498

2.7.2.4 函数: get_context..... 1499

2.7.2.5 函数: get_overflow_status..... 1499

2.7.2.6 函数: reset_overflow_status..... 1500

2.7.3 Stream 管理..... 1500

2.7.3.1 函数: create_stream..... 1500

2.7.3.2 函数: create_stream_with_config..... 1502

2.7.3.3 函数: destroy_stream..... 1503

2.7.3.4 函数: destroy_stream_force..... 1504

2.7.3.5 函数: set_stream_overflow_switch..... 1504

2.7.3.6 函数: get_stream_overflow_switch..... 1505

2.7.3.7 函数: set_stream_failure_mode..... 1506

2.7.3.8 函数: stream_query..... 1506

2.7.3.9 函数: synchronize_stream..... 1507

2.7.3.10 函数: synchronize_stream_with_timeout..... 1507

2.7.4 Event 管理..... 1508

2.7.4.1 函数: create_event..... 1508

2.7.4.2 函数: create_event_with_flag..... 1508

2.7.4.3 函数: create_event_ex_with_flag..... 1509

2.7.4.4 函数: destroy_event.....	1510
2.7.4.5 函数: record_event.....	1511
2.7.4.6 函数: reset_event.....	1512
2.7.4.7 函数: query_event.....	1512
2.7.4.8 函数: query_event_status.....	1513
2.7.4.9 函数: query_event_wait_status.....	1514
2.7.4.10 函数: synchronize_event.....	1514
2.7.4.11 函数: synchronize_event_with_timeout.....	1514
2.7.4.12 函数: event_elapsed_time.....	1515
2.7.4.13 函数: stream_wait_event.....	1516
2.7.4.14 函数: set_op_wait_timeout.....	1516
2.7.5 执行控制.....	1517
2.7.5.1 函数: subscribe_report.....	1517
2.7.5.2 函数: launch_callback.....	1518
2.7.5.3 函数: process_report.....	1519
2.7.5.4 函数: unsubscribe_report.....	1520
2.7.6 内存管理.....	1521
2.7.6.1 总体说明.....	1521
2.7.6.2 函数: malloc.....	1522
2.7.6.3 函数: malloc_cached.....	1523
2.7.6.4 函数: mem_flush.....	1524
2.7.6.5 函数: mem_invalidate.....	1524
2.7.6.6 函数: get_mem_info.....	1525
2.7.6.7 函数: free.....	1526
2.7.6.8 函数: malloc_host.....	1526
2.7.6.9 函数: free_host.....	1527
2.7.6.10 函数: memset.....	1528
2.7.6.11 函数: memset_async.....	1528
2.7.6.12 函数: memcpy.....	1529
2.7.6.13 函数: memcpy_async.....	1529
2.7.6.14 函数: memcpy2d.....	1530
2.7.6.15 函数: memcpy2d_async.....	1531
2.7.6.16 函数: malloc_physical.....	1532
2.7.6.17 函数: free_physical.....	1533
2.7.6.18 函数: reserve_mem_address.....	1534
2.7.6.19 函数: release_mem_address.....	1535
2.7.6.20 函数: map_mem.....	1535
2.7.6.21 函数: unmap_mem.....	1536
2.7.6.22 函数: mem_export_to_shareable_handle.....	1537
2.7.6.23 函数: device_get_bare_tgid.....	1539
2.7.6.24 函数: mem_set_pid_to_shareable_handle.....	1539
2.7.6.25 函数: mem_import_from_shareable_handle.....	1540

2.7.6.26 函数: mem_get_allocation_granularity.....	1541
2.7.7 异常处理.....	1542
2.7.7.1 函数: set_exception_info_callback.....	1542
2.7.7.2 函数: get_task_id_from_exception_info.....	1542
2.7.7.3 函数: get_stream_id_from_exception_info.....	1543
2.7.7.4 函数: get_thread_id_from_exception_info.....	1543
2.7.7.5 函数: get_device_id_from_exception_info.....	1544
2.7.7.6 函数: get_error_code_from_exception_info.....	1544
2.8 模型加载与执行 (mdl)	1545
2.8.1 模型加载与执行.....	1545
2.8.1.1 函数: load_from_file.....	1545
2.8.1.2 函数: load_from_mem.....	1546
2.8.1.3 函数: load_from_file_with_mem.....	1546
2.8.1.4 函数: load_from_mem_with_mem.....	1548
2.8.1.5 函数: execute.....	1549
2.8.1.6 函数: execute_v2.....	1550
2.8.1.7 函数: execute_async.....	1552
2.8.1.8 函数: unload.....	1553
2.8.1.9 函数: query_size.....	1554
2.8.1.10 函数: query_size_from_mem.....	1554
2.8.1.11 函数: set_dynamic_batch_size.....	1555
2.8.1.12 函数: set_dynamic_hw_size.....	1556
2.8.1.13 函数: create_and_get_op_desc.....	1557
2.8.1.14 函数: init_dump.....	1558
2.8.1.15 函数: set_dump.....	1559
2.8.1.16 函数: finalize_dump.....	1566
2.8.1.17 函数: get_aipp_type.....	1567
2.8.1.18 函数: set_aipp_by_input_index.....	1567
2.8.1.19 函数: set_input_dynamic_dims.....	1570
2.8.1.20 函数: create_config_handle.....	1571
2.8.1.21 函数: destroy_config_handle.....	1572
2.8.1.22 函数: set_config_opt.....	1572
2.8.1.23 函数: load_with_config.....	1573
2.8.1.24 函数: get_tensor_real_name.....	1574
2.8.1.25 函数: create_exec_config_handle.....	1575
2.8.1.26 函数: destroy_exec_config_handle.....	1575
2.8.1.27 函数: set_exec_config_opt.....	1575
2.8.2 aclmdlAIPP.....	1576
2.8.2.1 函数: create_aipp.....	1576
2.8.2.2 函数: destroy_aipp.....	1577
2.8.2.3 设置动态 AIPP 参数.....	1577
2.8.2.3.1 函数: set_aipp_input_format.....	1577

2.8.2.3.2 函数: set_aipp_csc_params.....	1578
2.8.2.3.3 函数: set_aipp_rbuw_swap_switch.....	1580
2.8.2.3.4 函数: set_aipp_ax_swap_switch.....	1580
2.8.2.3.5 函数: set_aipp_src_image_size.....	1581
2.8.2.3.6 函数: set_aipp_crop_params.....	1581
2.8.2.3.7 函数: set_aipp_padding_params.....	1582
2.8.2.3.8 函数: set_aipp_dtc_pixel_mean.....	1583
2.8.2.3.9 函数: set_aipp_dtc_pixel_min.....	1584
2.8.2.3.10 函数: set_aipp_pixel_var_reci.....	1585
2.8.2.3.11 函数: set_input_aipp.....	1585
2.8.3 aclmdlDataset.....	1588
2.8.3.1 总体说明.....	1588
2.8.3.2 函数: create_dataset.....	1588
2.8.3.3 函数: destroy_dataset.....	1588
2.8.3.4 函数: add_dataset_buffer.....	1589
2.8.3.5 函数: get_dataset_num_buffers.....	1589
2.8.3.6 函数: get_dataset_buffer.....	1590
2.8.3.7 函数: set_dataset_tensor_desc.....	1590
2.8.3.8 函数: get_dataset_tensor_desc.....	1591
2.8.4 aclmdlDesc.....	1592
2.8.4.1 函数: create_desc.....	1592
2.8.4.2 函数: destroy_desc.....	1592
2.8.4.3 函数: get_desc.....	1593
2.8.4.4 函数: get_num_inputs.....	1593
2.8.4.5 函数: get_num_outputs.....	1593
2.8.4.6 函数: get_input_size_by_index.....	1594
2.8.4.7 函数: get_output_size_by_index.....	1594
2.8.4.8 函数: get_input_dims.....	1595
2.8.4.9 函数: get_input_dims_v2.....	1597
2.8.4.10 函数: get_output_dims.....	1599
2.8.4.11 函数: get_input_name_by_index.....	1600
2.8.4.12 函数: get_output_name_by_index.....	1600
2.8.4.13 函数: get_input_format.....	1601
2.8.4.14 函数: get_output_format.....	1602
2.8.4.15 函数: get_input_data_type.....	1602
2.8.4.16 函数: get_output_data_type.....	1602
2.8.4.17 函数: get_input_index_by_name.....	1603
2.8.4.18 函数: get_output_index_by_name.....	1603
2.8.4.19 函数: get_dynamic_batch.....	1604
2.8.4.20 函数: get_dynamic_hw.....	1604
2.8.4.21 函数: get_cur_output_dims.....	1605
2.8.4.22 函数: get_first_aipp_info.....	1606

2.8.4.23 函数: get_input_dynamic_gear_count.....	1607
2.8.4.24 函数: get_input_dynamic_dims.....	1607
2.8.4.25 函数: get_op_attr.....	1609
2.9 媒体数据处理 V1.....	1609
2.9.1 总体说明.....	1609
2.9.2 基本概念.....	1613
2.9.3 内存申请与释放.....	1614
2.9.3.1 函数: dvpp_malloc.....	1614
2.9.3.2 函数: dvpp_free.....	1615
2.9.4 通道创建与释放.....	1615
2.9.4.1 VPC/JPEGD/JPEGE/PNGD 图片处理通道.....	1616
2.9.4.1.1 函数: dvpp_create_channel.....	1616
2.9.4.1.2 函数: dvpp_destroy_channel.....	1616
2.9.4.2 VDEC 视频解码通道.....	1617
2.9.4.2.1 函数: vdec_create_channel.....	1617
2.9.4.2.2 函数: vdec_destroy_channel.....	1618
2.9.4.3 VENC 视频编码通道.....	1619
2.9.4.3.1 函数: venc_create_channel.....	1619
2.9.4.3.2 函数: venc_destroy_channel.....	1620
2.9.5 common.....	1620
2.9.5.1 函数: dvpp_create_channel_desc.....	1620
2.9.5.2 函数: dvpp_destroy_channel_desc.....	1621
2.9.5.3 函数: dvpp_get_channel_desc_channel_id.....	1621
2.9.5.4 函数: dvpp_set_channel_desc_mode.....	1622
2.9.5.5 函数: dvpp_set_channel_desc_param.....	1622
2.9.5.6 函数: dvpp_get_channel_desc_param.....	1623
2.9.5.7 函数: vdec_set_channel_desc_param.....	1624
2.9.5.8 函数: vdec_get_channel_desc_param.....	1624
2.9.5.9 函数: dvpp_create_pic_desc.....	1625
2.9.5.10 函数: dvpp_destroy_pic_desc.....	1625
2.9.5.11 函数: dvpp_set_pic_desc_data.....	1626
2.9.5.12 函数: dvpp_set_pic_desc_size.....	1626
2.9.5.13 函数: dvpp_set_pic_desc_format.....	1626
2.9.5.14 函数: dvpp_set_pic_desc_width.....	1627
2.9.5.15 函数: dvpp_set_pic_desc_height.....	1627
2.9.5.16 函数: dvpp_set_pic_desc_width_stride.....	1628
2.9.5.17 函数: dvpp_set_pic_desc_height_stride.....	1628
2.9.5.18 函数: dvpp_set_pic_desc_ret_code.....	1629
2.9.5.19 函数: dvpp_get_pic_desc_data.....	1629
2.9.5.20 函数: dvpp_get_pic_desc_size.....	1630
2.9.5.21 函数: dvpp_get_pic_desc_format.....	1630
2.9.5.22 函数: dvpp_get_pic_desc_width.....	1630

2.9.5.23 函数: dvpp_get_pic_desc_height.....	1631
2.9.5.24 函数: dvpp_get_pic_desc_width_stride.....	1631
2.9.5.25 函数: dvpp_get_pic_desc_height_stride.....	1631
2.9.5.26 函数: dvpp_get_pic_desc_ret_code.....	1632
2.9.5.27 函数: dvpp_create_stream_desc.....	1632
2.9.5.28 函数: dvpp_destroy_stream_desc.....	1632
2.9.5.29 函数: dvpp_set_stream_desc_data.....	1633
2.9.5.30 函数: dvpp_set_stream_desc_size.....	1633
2.9.5.31 函数: dvpp_set_stream_desc_format.....	1634
2.9.5.32 函数: dvpp_set_stream_desc_timestamp.....	1634
2.9.5.33 函数: dvpp_set_stream_desc_ret_code.....	1635
2.9.5.34 函数: dvpp_set_stream_desc_eos.....	1635
2.9.5.35 函数: dvpp_get_stream_desc_data.....	1636
2.9.5.36 函数: dvpp_get_stream_desc_size.....	1636
2.9.5.37 函数: dvpp_get_stream_desc_format.....	1637
2.9.5.38 函数: dvpp_get_stream_desc_timestamp.....	1637
2.9.5.39 函数: dvpp_get_stream_desc_ret_code.....	1637
2.9.5.40 函数: dvpp_get_stream_desc_eos.....	1638
2.9.5.41 函数: dvpp_create_batch_pic_desc.....	1638
2.9.5.42 函数: dvpp_destroy_batch_pic_desc.....	1638
2.9.5.43 函数: dvpp_get_pic_desc.....	1639
2.9.5.44 函数: dvpp_create_border_config.....	1639
2.9.5.45 函数: dvpp_destroy_border_config.....	1640
2.9.5.46 函数: dvpp_set_border_config_value.....	1640
2.9.5.47 函数: dvpp_set_border_config_border_type.....	1641
2.9.5.48 函数: dvpp_set_border_config_top.....	1641
2.9.5.49 函数: dvpp_set_border_config_bottom.....	1642
2.9.5.50 函数: dvpp_set_border_config_left.....	1642
2.9.5.51 函数: dvpp_set_border_config_right.....	1643
2.9.5.52 函数: dvpp_get_border_config_value.....	1643
2.9.5.53 函数: dvpp_get_border_config_border_type.....	1643
2.9.5.54 函数: dvpp_get_border_config_top.....	1644
2.9.5.55 函数: dvpp_get_border_config_bottom.....	1644
2.9.5.56 函数: dvpp_get_border_config_left.....	1645
2.9.5.57 函数: dvpp_get_border_config_right.....	1645
2.9.6 VPC 功能.....	1645
2.9.6.1 功能说明.....	1645
2.9.6.2 约束说明.....	1650
2.9.6.3 性能指标说明.....	1664
2.9.6.4 函数: dvpp_vpc_resize_async.....	1673
2.9.6.5 函数: dvpp_vpc_crop_async.....	1676
2.9.6.6 函数: dvpp_vpc_crop_resize_async.....	1679

2.9.6.7 函数: dvpp_vpc_batch_crop_async.....	1681
2.9.6.8 函数: dvpp_vpc_batch_crop_resize_async.....	1684
2.9.6.9 函数: dvpp_vpc_crop_and_paste_async.....	1687
2.9.6.10 函数: dvpp_vpc_crop_resize_paste_async.....	1690
2.9.6.11 函数: dvpp_vpc_batch_crop_and_paste_async.....	1693
2.9.6.12 函数: dvpp_vpc_batch_crop_resize_paste_async.....	1696
2.9.6.13 函数: dvpp_vpc_batch_crop_resize_make_border_async.....	1699
2.9.6.14 函数: dvpp_vpc_convert_color_async.....	1702
2.9.6.15 函数: dvpp_vpc_pyr_down_async.....	1703
2.9.6.16 函数: dvpp_create_roi_config.....	1705
2.9.6.17 函数: dvpp_destroy_roi_config.....	1706
2.9.6.18 函数: dvpp_set_roi_config.....	1706
2.9.6.19 函数: dvpp_set_roi_config_left.....	1707
2.9.6.20 函数: dvpp_set_roi_config_right.....	1708
2.9.6.21 函数: dvpp_set_roi_config_top.....	1708
2.9.6.22 函数: dvpp_set_roi_config_bottom.....	1709
2.9.6.23 函数: dvpp_create_resize_config.....	1709
2.9.6.24 函数: dvpp_destroy_resize_config.....	1710
2.9.6.25 函数: dvpp_set_resize_config_interpolation.....	1710
2.9.6.26 函数: dvpp_get_resize_config_interpolation.....	1713
2.9.7 JPEGD 功能.....	1714
2.9.7.1 功能及约束说明.....	1714
2.9.7.2 性能指标说明.....	1719
2.9.7.3 函数: dvpp_jpeg_decode_async.....	1721
2.9.7.4 函数: dvpp_jpeg_get_image_info.....	1723
2.9.7.5 函数: dvpp_jpeg_get_image_info_v2.....	1723
2.9.7.6 函数: dvpp_jpeg_predict_dec_size.....	1724
2.9.8 JPEGE 功能.....	1724
2.9.8.1 功能及约束说明.....	1724
2.9.8.2 性能指标说明.....	1726
2.9.8.3 函数: dvpp_jpeg_encode_async.....	1728
2.9.8.4 函数: dvpp_jpeg_predict_enc_size.....	1729
2.9.8.5 函数: dvpp_create_jpege_config.....	1730
2.9.8.6 函数: dvpp_destroy_jpege_config.....	1730
2.9.8.7 函数: dvpp_set_jpege_config_level.....	1731
2.9.8.8 函数: dvpp_get_jpege_config_level.....	1731
2.9.9 VDEC 功能.....	1732
2.9.9.1 功能及约束说明.....	1732
2.9.9.2 性能指标说明.....	1734
2.9.9.3 函数: vdec_send_frame.....	1742
2.9.9.4 函数: vdec_send_skipped_frame.....	1744
2.9.9.5 函数: vdec_create_frame_config.....	1746

2.9.9.6 函数: vdec_destroy_frame_config.....	1746
2.9.9.7 函数: vdec_create_channel_desc.....	1747
2.9.9.8 函数: vdec_destroy_channel_desc.....	1747
2.9.9.9 函数: vdec_set_channel_desc_channel_id.....	1747
2.9.9.10 函数: vdec_set_channel_desc_thread_id.....	1748
2.9.9.11 函数: vdec_set_channel_desc_callback.....	1749
2.9.9.12 函数: vdec_set_channel_desc_entype.....	1749
2.9.9.13 函数: vdec_set_channel_desc_out_pic_format.....	1750
2.9.9.14 函数: vdec_set_channel_desc_out_pic_width.....	1750
2.9.9.15 函数: vdec_set_channel_desc_out_pic_height.....	1751
2.9.9.16 函数: vdec_set_channel_desc_ref_frame_num.....	1752
2.9.9.17 函数: vdec_set_channel_desc_out_mode.....	1752
2.9.9.18 函数: vdec_set_channel_desc_bit_depth.....	1753
2.9.9.19 函数: vdec_get_channel_desc_channel_id.....	1754
2.9.9.20 函数: vdec_get_channel_desc_thread_id.....	1754
2.9.9.21 函数: vdec_get_channel_desc_callback.....	1754
2.9.9.22 函数: vdec_get_channel_desc_entype.....	1755
2.9.9.23 函数: vdec_get_channel_desc_out_pic_format.....	1755
2.9.9.24 函数: vdec_get_channel_desc_out_pic_width.....	1755
2.9.9.25 函数: vdec_get_channel_desc_out_pic_height.....	1756
2.9.9.26 函数: vdec_get_channel_desc_ref_frame_num.....	1756
2.9.9.27 函数: vdec_get_channel_desc_out_mode.....	1756
2.9.9.28 函数: vdec_get_channel_desc_bit_depth.....	1757
2.9.10 VENC 功能.....	1757
2.9.10.1 功能及约束说明.....	1757
2.9.10.2 性能指标说明.....	1758
2.9.10.3 函数: venc_send_frame.....	1760
2.9.10.4 函数: venc_create_channel_desc.....	1761
2.9.10.5 函数: venc_destroy_channel_desc.....	1761
2.9.10.6 函数: venc_set_channel_desc_thread_id.....	1762
2.9.10.7 函数: venc_set_channel_desc_callback.....	1762
2.9.10.8 函数: venc_set_channel_desc_entype.....	1763
2.9.10.9 函数: venc_set_channel_desc_pic_format.....	1763
2.9.10.10 函数: venc_set_channel_desc_pic_width.....	1764
2.9.10.11 函数: venc_set_channel_desc_pic_height.....	1764
2.9.10.12 函数: venc_set_channel_desc_key_frame_interval.....	1765
2.9.10.13 函数: venc_set_channel_desc_buf_addr.....	1766
2.9.10.14 函数: venc_set_channel_desc_buf_size.....	1766
2.9.10.15 函数: venc_set_channel_desc_rc_mode.....	1767
2.9.10.16 函数: venc_set_channel_desc_src_rate.....	1768
2.9.10.17 函数: venc_set_channel_desc_max_bit_rate.....	1768
2.9.10.18 函数: venc_get_channel_desc_channel_id.....	1769

2.9.10.19 函数: venc_get_channel_desc_thread_id.....	1769
2.9.10.20 函数: venc_get_channel_desc_callback.....	1770
2.9.10.21 函数: venc_get_channel_desc_entype.....	1770
2.9.10.22 函数: venc_get_channel_desc_pic_format.....	1771
2.9.10.23 函数: venc_get_channel_desc_pic_width.....	1771
2.9.10.24 函数: venc_get_channel_desc_pic_height.....	1771
2.9.10.25 函数: venc_get_channel_desc_key_frame_interval.....	1772
2.9.10.26 函数: venc_get_channel_desc_buf_addr.....	1772
2.9.10.27 函数: venc_get_channel_desc_buf_size.....	1773
2.9.10.28 函数: venc_get_channel_desc_rc_mode.....	1773
2.9.10.29 函数: venc_get_channel_desc_src_rate.....	1773
2.9.10.30 函数: venc_get_channel_desc_max_bit_rate.....	1774
2.9.10.31 函数: venc_set_channel_desc_param.....	1774
2.9.10.32 函数: venc_get_channel_desc_param.....	1775
2.9.10.33 函数: venc_create_frame_config.....	1776
2.9.10.34 函数: venc_destroy_frame_config.....	1776
2.9.10.35 函数: venc_set_frame_config_force_i_frame.....	1777
2.9.10.36 函数: venc_set_frame_config_eos.....	1777
2.9.10.37 函数: venc_get_frame_config_force_i_frame.....	1778
2.9.10.38 函数: venc_get_frame_config_eos.....	1778
2.9.11 PNGD 功能.....	1779
2.9.11.1 功能及约束说明.....	1779
2.9.11.2 性能指标说明.....	1780
2.9.11.3 函数: dvpp_png_decode_async.....	1782
2.9.11.4 函数: dvpp_png_get_image_info.....	1784
2.9.11.5 函数: dvpp_png_predict_dec_size.....	1784
2.10 媒体数据处理 V2.....	1785
2.10.1 总体说明.....	1785
2.10.2 基本概念.....	1788
2.10.3 公共接口.....	1788
2.10.3.1 acl.himpi.sys_init.....	1788
2.10.3.2 acl.himpi.sys_exit.....	1789
2.10.3.3 acl.himpi.dvpp_malloc.....	1790
2.10.3.4 acl.himpi.dvpp_free.....	1790
2.10.3.5 acl.himpi.dvpp_get_image_info.....	1791
2.10.3.6 acl.himpi.dvpp_get_version.....	1792
2.10.3.7 acl.himpi.sys_create_epoll.....	1792
2.10.3.8 acl.himpi.sys_ctl_epoll.....	1793
2.10.3.9 acl.himpi.sys_wait_epoll.....	1794
2.10.3.10 acl.himpi.sys_close_epoll.....	1795
2.10.3.11 acl.himpi.sys_get_image_align_info.....	1795
2.10.3.12 acl.himpi.sys_set_chn_csc_matrix.....	1796

2.10.3.13 acl.himpi.sys_get_chn_csc_matrix.....	1798
2.10.4 VPC 图像处理接口.....	1798
2.10.4.1 VPC 功能.....	1798
2.10.4.2 约束说明.....	1804
2.10.4.3 性能指标说明.....	1815
2.10.4.4 acl.himpi.vpc_create_chn.....	1819
2.10.4.5 acl.himpi.vpc_destroy_chn.....	1820
2.10.4.6 acl.himpi.vpc_resize.....	1821
2.10.4.7 acl.himpi.vpc_crop.....	1824
2.10.4.8 acl.himpi.vpc_crop_resize.....	1827
2.10.4.9 acl.himpi.vpc_crop_resize_paste.....	1829
2.10.4.10 acl.himpi.vpc_convert_color.....	1831
2.10.4.11 acl.himpi.vpc_convert_color_v2.....	1833
2.10.4.12 acl.himpi.vpc_convert_color_to_yuv420.....	1835
2.10.4.13 acl.himpi.vpc_copy_make_border.....	1837
2.10.4.14 acl.himpi.vpc_pyrdown.....	1839
2.10.4.15 acl.himpi.vpc_calc_hist.....	1842
2.10.4.16 acl.himpi.vpc_calc_hist_v2.....	1844
2.10.4.17 acl.himpi.vpc_equalize_hist.....	1846
2.10.4.18 acl.himpi.vpc_get_process_result.....	1848
2.10.4.19 acl.himpi.vpc_sys_create_chn.....	1848
2.10.4.20 acl.himpi.vpc_crop_resize_make_border.....	1849
2.10.4.21 acl.himpi.vpc_batch_crop_resize_paste.....	1852
2.10.4.22 acl.himpi.vpc_batch_crop_resize_make_border.....	1855
2.10.4.23 acl.himpi.vpc_blur.....	1858
2.10.4.24 acl.himpi.vpc_crop_resize_resize_paste.....	1861
2.10.4.25 acl.himpi.vpc_dilate.....	1863
2.10.4.26 acl.himpi.vpc_draw_cover.....	1864
2.10.4.27 acl.himpi.vpc_draw_line.....	1865
2.10.4.28 acl.himpi.vpc_draw_mosaic.....	1866
2.10.4.29 acl.himpi.vpc_draw_osd.....	1867
2.10.4.30 acl.himpi.vpc_erode.....	1868
2.10.4.31 acl.himpi.vpc_filter2d.....	1869
2.10.4.32 acl.himpi.vpc_flip.....	1871
2.10.4.33 acl.himpi.vpc_gaussian_blur.....	1873
2.10.4.34 acl.himpi.vpc_get_affine_lut.....	1875
2.10.4.35 acl.himpi.vpc_get_affine_transform.....	1876
2.10.4.36 acl.himpi.vpc_get_lut_mem_size.....	1877
2.10.4.37 acl.himpi.vpc_get_perspective_lut.....	1877
2.10.4.38 acl.himpi.vpc_get_remap_lut.....	1878
2.10.4.39 acl.himpi.vpc_get_rotation_matrix.....	1879
2.10.4.40 acl.himpi.vpc_lut_remap.....	1879

2.10.4.41 acl.himpi.vpc_median_blur.....	1883
2.10.4.42 acl.himpi.vpc_rotate.....	1884
2.10.4.43 acl.himpi.vpc_set_chn_workspace.....	1885
2.10.4.44 acl.himpi.vpc_warp_affine.....	1886
2.10.5 VDEC 视频解码/JPEGD 图像解码接口.....	1889
2.10.5.1 JPEGD 功能及约束说明.....	1889
2.10.5.2 JPEGD 性能指标数据.....	1893
2.10.5.3 VDEC 功能及约束说明.....	1895
2.10.5.4 VDEC 性能指标数据.....	1898
2.10.5.5 acl.himpi.vdec_create_chn.....	1900
2.10.5.6 acl.himpi.vdec_destroy_chn.....	1902
2.10.5.7 acl.himpi.vdec_get_chn_attr.....	1903
2.10.5.8 acl.himpi.vdec_set_chn_attr.....	1904
2.10.5.9 acl.himpi.vdec_start_rcv_stream.....	1904
2.10.5.10 acl.himpi.vdec_stop_rcv_stream.....	1905
2.10.5.11 acl.himpi.vdec_query_status.....	1906
2.10.5.12 acl.himpi.vdec_reset_chn.....	1907
2.10.5.13 acl.himpi.vdec_set_chn_param.....	1907
2.10.5.14 acl.himpi.vdec_get_chn_param.....	1908
2.10.5.15 acl.himpi.vdec_set_protocol_param.....	1909
2.10.5.16 acl.himpi.vdec_get_protocol_param.....	1910
2.10.5.17 acl.himpi.vdec_send_stream.....	1911
2.10.5.18 acl.himpi.vdec_get_frame.....	1913
2.10.5.19 acl.himpi.vdec_release_frame.....	1915
2.10.5.20 acl.himpi.vdec_get_fd.....	1916
2.10.5.21 acl.himpi.vdec_close_fd.....	1917
2.10.5.22 acl.himpi.vdec_get_pic_buf_size.....	1917
2.10.5.23 acl.himpi.vdec_get_tmv_buf_size.....	1918
2.10.5.24 acl.himpi.vdec_set_jpegd_precision_mode.....	1919
2.10.5.25 acl.himpi.vdec_get_jpegd_output_info.....	1919
2.10.6 VENC 视频/JPEGE 图像编码接口.....	1920
2.10.6.1 JPEGE 功能及约束说明.....	1920
2.10.6.2 JPEGE 性能指标说明.....	1926
2.10.6.3 VENC 功能及约束说明.....	1927
2.10.6.4 VENC 性能指标说明.....	1928
2.10.6.5 acl.himpi.venc_create_chn.....	1929
2.10.6.6 acl.himpi.venc_destroy_chn.....	1932
2.10.6.7 acl.himpi.venc_start_chn.....	1933
2.10.6.8 acl.himpi.venc_stop_chn.....	1933
2.10.6.9 acl.himpi.venc_query_status.....	1934
2.10.6.10 acl.himpi.venc_get_stream.....	1935
2.10.6.11 acl.himpi.venc_release_stream.....	1939

2.10.6.12 acl.himpi.venc_send_frame.....	1940
2.10.6.13 acl.himpi.venc_set_mod_param.....	1941
2.10.6.14 acl.himpi.venc_get_mod_param.....	1942
2.10.6.15 acl.himpi.venc_request_idr.....	1943
2.10.6.16 acl.himpi.venc_get_fd.....	1943
2.10.6.17 acl.himpi.venc_close_fd.....	1944
2.10.6.18 acl.himpi.venc_set_jpeg_param.....	1944
2.10.6.19 acl.himpi.venc_get_jpeg_param.....	1945
2.10.6.20 acl.himpi.venc_set_chn_param.....	1946
2.10.6.21 acl.himpi.venc_get_chn_param.....	1947
2.10.6.22 acl.himpi.venc_set_scene_mode.....	1947
2.10.6.23 acl.himpi.venc_get_scene_mode.....	1948
2.10.6.24 acl.himpi.venc_set_rc_param.....	1949
2.10.6.25 acl.himpi.venc_get_rc_param.....	1949
2.10.6.26 acl.himpi.venc_set_jpeg_huffman_param.....	1950
2.10.6.27 acl.himpi.venc_get_jpeg_huffman_param.....	1951
2.10.6.28 acl.himpi.venc_compact_jpeg_tables.....	1951
2.10.6.29 acl.himpi.venc_send_jpege_frame.....	1952
2.10.6.30 acl.himpi.venc_get_jpege_predicted_size.....	1954
2.10.6.31 acl.himpi.venc_get_cu_pred.....	1954
2.10.6.32 acl.himpi.venc_set_cu_pred.....	1955
2.10.6.33 acl.himpi.venc_get_h264_vui.....	1956
2.10.6.34 acl.himpi.venc_set_h264_vui.....	1957
2.10.6.35 acl.himpi.venc_get_h265_vui.....	1958
2.10.6.36 acl.himpi.venc_set_h265_vui.....	1959
2.10.6.37 acl.himpi.venc_get_intra_refresh.....	1960
2.10.6.38 acl.himpi.venc_set_intra_refresh.....	1961
2.10.6.39 acl.himpi.venc_get_ref_param.....	1962
2.10.6.40 acl.himpi.venc_set_ref_param.....	1963
2.10.6.41 acl.himpi.venc_get_roi_attr.....	1965
2.10.6.42 acl.himpi.venc_set_roi_attr.....	1966
2.10.6.43 acl.himpi.venc_set_chn_attr.....	1968
2.10.6.44 acl.himpi.venc_get_chn_attr.....	1969
2.10.6.45 acl.himpi.venc_set_slice_split.....	1970
2.10.6.46 acl.himpi.venc_get_slice_split.....	1971
2.10.7 PNGD 图像处理接口.....	1971
2.10.7.1 功能及约束说明.....	1971
2.10.7.2 性能指标说明.....	1974
2.10.7.3 acl.himpi.pngd_create_chn.....	1975
2.10.7.4 acl.himpi.pngd_destroy_chn.....	1976
2.10.7.5 acl.himpi.pngd_send_stream.....	1976
2.10.7.6 acl.himpi.pngd_get_image_data.....	1978

2.10.7.7 acl.himpi.png_get_image_info.....	1979
2.10.8 数据类型.....	1979
2.10.8.1 总体说明.....	1979
2.10.8.2 公共.....	1979
2.10.8.2.1 基本数据类型说明.....	1980
2.10.8.2.2 hi_pixel_format.....	1980
2.10.8.2.3 hi_payload_type.....	1984
2.10.8.2.4 hi_video_field.....	1986
2.10.8.2.5 hi_dynamic_range.....	1986
2.10.8.2.6 hi_color_gamut.....	1988
2.10.8.2.7 hi_video_supplement.....	1988
2.10.8.2.8 hi_video_frame.....	1989
2.10.8.2.9 hi_video_frame_info.....	1993
2.10.8.2.10 hi_vb_src.....	1993
2.10.8.2.11 hi_mod_id.....	1994
2.10.8.2.12 hi_data_bit_width.....	1995
2.10.8.2.13 hi_video_format.....	1995
2.10.8.2.14 hi_img_info.....	1995
2.10.8.2.15 hi_pic_buf_attr.....	1996
2.10.8.2.16 hi_jpeg_raw_format.....	1997
2.10.8.2.17 hi_dvpp_epoll_ctl_op.....	1997
2.10.8.2.18 hi_dvpp_epoll_event_type.....	1998
2.10.8.2.19 hi_dvpp_epoll_event.....	1998
2.10.8.2.20 hi_csc_matrix.....	1998
2.10.8.2.21 hi_coefficient.....	2000
2.10.8.2.22 hi_csc_coefficient.....	2001
2.10.8.2.23 hi_video_size.....	2001
2.10.8.2.24 hi_rect.....	2002
2.10.8.2.25 hi_op_mode.....	2002
2.10.8.2.26 hi_point.....	2002
2.10.8.2.27 hi_size.....	2003
2.10.8.2.28 hi_compress_mode.....	2003
2.10.8.2.29 hi_module_type.....	2004
2.10.8.2.30 hi_img_base_info.....	2004
2.10.8.2.31 hi_img_align_info.....	2005
2.10.8.3 视频/图像解码.....	2005
2.10.8.3.1 hi_vdec_send_mode.....	2005
2.10.8.3.2 hi_vdec_frame_type.....	2006
2.10.8.3.3 hi_quick_mark_mode.....	2007
2.10.8.3.4 hi_vdec_pic_info.....	2007
2.10.8.3.5 hi_vdec_video_attr.....	2011
2.10.8.3.6 hi_vdec_chn_attr.....	2011

2.10.8.3.7 hi_h264_protocol_param.....	2013
2.10.8.3.8 hi_h265_protocol_param.....	2014
2.10.8.3.9 hi_vdec_protocol_param.....	2015
2.10.8.3.10 hi_video_dec_mode.....	2015
2.10.8.3.11 hi_video_out_order.....	2016
2.10.8.3.12 hi_vdec_video_param.....	2016
2.10.8.3.13 hi_vdec_param_picture.....	2017
2.10.8.3.14 hi_vdec_chn_param.....	2017
2.10.8.3.15 hi_vdec_dec_err.....	2018
2.10.8.3.16 hi_vdec_chn_status.....	2019
2.10.8.3.17 hi_vdec_stream.....	2020
2.10.8.3.18 hi_vdec_supplement_info.....	2021
2.10.8.3.19 hi_vdec_video_supplement_info.....	2021
2.10.8.3.20 hi_jpegd_precision_mode.....	2021
2.10.8.4 视频/图像编码.....	2022
2.10.8.4.1 hi_venc_stream.....	2022
2.10.8.4.2 hi_venc_pack.....	2023
2.10.8.4.3 hi_venc_chn_attr.....	2024
2.10.8.4.4 hi_venc_attr.....	2025
2.10.8.4.5 hi_venc_rc_attr.....	2029
2.10.8.4.6 hi_venc_rc_mode.....	2033
2.10.8.4.7 hi_venc_h264_cbr.....	2034
2.10.8.4.8 hi_venc_h264_vbr.....	2036
2.10.8.4.9 hi_venc_h264_avbr.....	2037
2.10.8.4.10 hi_venc_h264_qvbr.....	2038
2.10.8.4.11 hi_venc_h264_cvbr.....	2039
2.10.8.4.12 hi_venc_h264_fixqp.....	2041
2.10.8.4.13 hi_venc_h264_qpmap.....	2042
2.10.8.4.14 hi_venc_mjpeg_cbr.....	2042
2.10.8.4.15 hi_venc_mjpeg_vbr.....	2042
2.10.8.4.16 hi_venc_mjpeg_fixqp.....	2042
2.10.8.4.17 hi_venc_h265_cbr.....	2043
2.10.8.4.18 hi_venc_h265_vbr.....	2044
2.10.8.4.19 hi_venc_h265_avbr.....	2045
2.10.8.4.20 hi_venc_h265_qvbr.....	2046
2.10.8.4.21 hi_venc_h265_cvbr.....	2047
2.10.8.4.22 hi_venc_h265_fixqp.....	2049
2.10.8.4.23 hi_venc_h265_qpmap.....	2050
2.10.8.4.24 hi_venc_gop_attr.....	2050
2.10.8.4.25 hi_venc_gop_mode.....	2051
2.10.8.4.26 hi_venc_gop_normal_p.....	2052
2.10.8.4.27 hi_venc_gop_dual_p.....	2052

2.10.8.4.28 hi_venc_gop_smart_p.....	2053
2.10.8.4.29 hi_venc_gop_adv_smart_p.....	2053
2.10.8.4.30 hi_venc_gop_bipred_b.....	2054
2.10.8.4.31 hi_venc_start_param.....	2054
2.10.8.4.32 hi_venc_chn_status.....	2054
2.10.8.4.33 hi_venc_h264_stream_info.....	2055
2.10.8.4.34 hi_venc_ref_type.....	2056
2.10.8.4.35 hi_venc_jpeg_stream_info.....	2057
2.10.8.4.36 hi_venc_h265_stream_info.....	2057
2.10.8.4.37 hi_venc_prores_stream_info.....	2059
2.10.8.4.38 hi_venc_mod_param.....	2059
2.10.8.4.39 hi_venc_mod_type.....	2060
2.10.8.4.40 hi_venc_venc_mod_param.....	2060
2.10.8.4.41 hi_venc_h264_mod_param.....	2061
2.10.8.4.42 hi_venc_h265_mod_param.....	2062
2.10.8.4.43 hi_venc_jpeg_mod_param.....	2063
2.10.8.4.44 hi_venc_jpeg_param.....	2064
2.10.8.4.45 hi_venc_h264_adv_stream_info.....	2065
2.10.8.4.46 hi_venc_h265_adv_stream_info.....	2066
2.10.8.4.47 hi_venc_data_type.....	2067
2.10.8.4.48 hi_venc_h264_nalu_type.....	2068
2.10.8.4.49 hi_venc_jpege_pack_type.....	2068
2.10.8.4.50 hi_venc_h265_nalu_type.....	2068
2.10.8.4.51 hi_venc_prores_pack_type.....	2069
2.10.8.4.52 hi_venc_chn_param.....	2069
2.10.8.4.53 hi_venc_stream_info.....	2070
2.10.8.4.54 hi_venc_pack_info.....	2071
2.10.8.4.55 hi_frame_rate_ctrl.....	2072
2.10.8.4.56 hi_venc_rc_qpmap_mode.....	2072
2.10.8.4.57 hi_venc_sse_info.....	2073
2.10.8.4.58 hi_venc_h264_attr.....	2073
2.10.8.4.59 hi_venc_h265_attr.....	2074
2.10.8.4.60 hi_venc_pic_rcv_mode.....	2075
2.10.8.4.61 hi_venc_jpeg_attr.....	2075
2.10.8.4.62 hi_venc_prores_frame_rate.....	2075
2.10.8.4.63 hi_venc_prores_aspect_ratio.....	2075
2.10.8.4.64 hi_venc_prores_attr.....	2076
2.10.8.4.65 hi_crop_info.....	2076
2.10.8.4.66 hi_venc_scene_mode.....	2076
2.10.8.4.67 hi_venc_rc_param.....	2076
2.10.8.4.68 hi_venc_h264_cbr_param.....	2080
2.10.8.4.69 hi_venc_h264_vbr_param.....	2081

2.10.8.4.70 hi_venc_h264_avbr_param.....	2083
2.10.8.4.71 hi_venc_h264_qvbr_param.....	2084
2.10.8.4.72 hi_venc_h264_cvbr_param.....	2086
2.10.8.4.73 hi_venc_h265_vbr_param.....	2088
2.10.8.4.74 hi_venc_h265_avbr_param.....	2089
2.10.8.4.75 hi_venc_h265_qvbr_param.....	2091
2.10.8.4.76 hi_venc_h265_cvbr_param.....	2093
2.10.8.4.77 hi_venc_h265_cbr_param.....	2095
2.10.8.4.78 hi_venc_scene_chg_detect.....	2096
2.10.8.4.79 hi_venc_huffman_dc_table.....	2096
2.10.8.4.80 hi_venc_huffman_ac_table.....	2097
2.10.8.4.81 hi_venc_jpeg_huffman_param.....	2097
2.10.8.4.82 hi_venc_mpf_cfg.....	2098
2.10.8.4.83 hi_venc_mjpeg_cbr_param.....	2098
2.10.8.4.84 hi_venc_mjpeg_vbr_param.....	2098
2.10.8.4.85 hi_venc_roi_attr.....	2098
2.10.8.4.86 hi_venc_ref_param.....	2099
2.10.8.4.87 hi_venc_h264_vui.....	2100
2.10.8.4.88 hi_venc_h265_vui.....	2100
2.10.8.4.89 hi_venc_vui_aspect_ratio.....	2101
2.10.8.4.90 hi_venc_vui_h264_time_info.....	2102
2.10.8.4.91 hi_venc_vui_h265_time_info.....	2102
2.10.8.4.92 hi_venc_vui_video_signal.....	2103
2.10.8.4.93 hi_venc_vui_bitstream_restric.....	2104
2.10.8.4.94 hi_venc_cu_prediction.....	2105
2.10.8.4.95 hi_venc_intra_refresh.....	2106
2.10.8.4.96 hi_venc_intra_refresh_mode.....	2108
2.10.8.4.97 hi_venc_slice_split.....	2109
2.10.8.5 图像处理.....	2109
2.10.8.5.1 hi_vpc_pic_info.....	2110
2.10.8.5.2 hi_vpc_crop_region.....	2110
2.10.8.5.3 hi_vpc_resize_info.....	2111
2.10.8.5.4 hi_vpc_crop_region_info.....	2112
2.10.8.5.5 hi_vpc_crop_resize_region.....	2112
2.10.8.5.6 hi_vpc_crop_resize_paste_region.....	2112
2.10.8.5.7 hi_vpc_bord_type.....	2113
2.10.8.5.8 hi_vpc_make_border_info.....	2114
2.10.8.5.9 hi_vpc_histogram_config.....	2114
2.10.8.5.10 hi_vpc_lut_remap.....	2115
2.10.8.5.11 hi_csc_conf.....	2115
2.10.8.5.12 hi_vpc_scalar.....	2116
2.10.8.5.13 hi_vpc_chn_attr.....	2116

2.10.8.5.14 hi_vpc_crop_resize_border_region.....	2117
2.10.8.5.15 hi_vpc_crop_resize_resize_paste_region.....	2118
2.10.8.5.16 hi_vpc_workspace_param.....	2119
2.10.8.5.17 hi_workspace_func.....	2120
2.10.8.5.18 hi_remap_lut.....	2120
2.10.8.5.19 hi_point_pair_info.....	2121
2.10.8.5.20 hi_float_point.....	2121
2.10.8.5.21 hi_warp_transform_param.....	2122
2.10.8.5.22 hi_transform_config.....	2122
2.10.8.5.23 hi_morph_shapes.....	2123
2.10.8.5.24 hi_rotation.....	2123
2.10.8.5.25 hi_blk_size.....	2124
2.10.8.5.26 hi_median_blur_param.....	2124
2.10.8.5.27 hi_median_blur_config.....	2125
2.10.8.5.28 hi_blur_param.....	2125
2.10.8.5.29 hi_blur_config.....	2126
2.10.8.5.30 hi_gaussian_blur_param.....	2126
2.10.8.5.31 hi_gaussian_blur_config.....	2127
2.10.8.5.32 hi_filter_2d_param.....	2127
2.10.8.5.33 hi_filter_2d_config.....	2128
2.10.8.5.34 hi_mosaic_param.....	2129
2.10.8.5.35 hi_rotate_param.....	2129
2.10.8.5.36 hi_mosaic.....	2130
2.10.8.5.37 hi_cover_type.....	2130
2.10.8.5.38 hi_cover_param.....	2130
2.10.8.5.39 hi_cover.....	2131
2.10.8.5.40 hi_quad_cover.....	2131
2.10.8.5.41 hi_line_param.....	2132
2.10.8.5.42 hi_line.....	2132
2.10.8.5.43 hi_osd_param.....	2133
2.10.8.5.44 hi_osd.....	2133
2.10.8.5.45 hi_osd_inverted_color.....	2135
2.10.8.5.46 hi_transform_matrix.....	2135
2.10.8.5.47 hi_matrix_type.....	2136
2.10.8.5.48 hi_flip_param.....	2136
2.10.8.5.49 hi_flip_mode.....	2136
2.10.8.5.50 hi_map_param.....	2137
2.10.8.6 PNGD 图像解码.....	2138
2.10.8.6.1 hi_pngd_chn_attr.....	2138
2.10.8.6.2 hi_img_stream.....	2138
2.10.8.6.3 hi_pic_info.....	2139
2.10.8.6.4 hi_png_color_format.....	2139

2.10.9 返回码列表.....	2139
2.10.9.1 公共返回码.....	2140
2.10.9.2 VPC 图像处理返回码.....	2140
2.10.9.3 VDEC 视频/JPEGD 图像解码返回码.....	2141
2.10.9.4 VENC 视频/JPEGE 图像编码返回码.....	2142
2.10.9.5 PNGD 图像解码返回码.....	2143
2.11 算子加载与执行 (op)	2144
2.11.1 函数: set_model_dir.....	2145
2.11.2 函数: load.....	2145
2.11.3 函数: set_op_execute_time_out.....	2146
2.11.4 函数: execute.....	2147
2.11.5 函数: execute_v2.....	2148
2.11.6 函数: execute_with_handle.....	2151
2.11.7 函数: cast.....	2152
2.11.8 函数: create_handle_for_cast.....	2152
2.11.9 函数: create_attr.....	2153
2.11.10 函数: destroy_attr.....	2153
2.11.11 函数: set_attr_bool.....	2154
2.11.12 函数: set_attr_int.....	2154
2.11.13 函数: set_attr_float.....	2154
2.11.14 函数: set_attr_string.....	2155
2.11.15 函数: set_attr_list_bool.....	2155
2.11.16 函数: set_attr_list_int.....	2156
2.11.17 函数: set_attr_list_float.....	2157
2.11.18 函数: set_attr_list_string.....	2157
2.11.19 函数: set_attr_list_list_int.....	2158
2.11.20 函数: create_handle.....	2158
2.11.21 函数: destroy_handle.....	2159
2.11.22 函数: infer_shape.....	2159
2.11.23 函数: set_attr_data_type.....	2160
2.11.24 函数: set_attr_list_data_type.....	2161
2.12 CBLAS 接口 (blas)	2161
2.12.1 总体说明.....	2161
2.12.2 关于 A、B、C 矩阵的 Shape 及内存大小计算公式.....	2162
2.12.3 函数: gemv_ex.....	2163
2.12.4 函数: create_handle_for_gemv_ex.....	2165
2.12.5 函数: hgemv.....	2166
2.12.6 函数: create_handle_for_hgemv.....	2167
2.12.7 函数: s8gemv.....	2168
2.12.8 函数: create_handle_for_s8gemv.....	2169
2.12.9 函数: gemm_ex.....	2170
2.12.10 函数: create_handle_for_gemm_ex.....	2172

2.12.11 函数: hgemm.....	2173
2.12.12 函数: create_handle_for_hgemm.....	2174
2.12.13 函数: s8gemm.....	2175
2.12.14 函数: create_handle_for_s8gemm.....	2177
2.13 算子编译.....	2177
2.13.1 函数: register_compile_func.....	2177
2.13.2 函数: unregister_compile_func.....	2178
2.13.3 函数: create_kernel.....	2178
2.13.4 函数: set_kernel_args.....	2179
2.13.5 函数: set_kernel_workspace_sizes.....	2180
2.13.6 函数: update_params.....	2181
2.14 特征向量检索.....	2181
2.14.1 总体说明.....	2181
2.14.2 函数: init.....	2182
2.14.3 函数: release.....	2183
2.14.4 函数: repo_add.....	2184
2.14.5 函数: repo_del.....	2184
2.14.6 函数: delete.....	2185
2.14.7 函数: modify.....	2186
2.14.8 函数: search.....	2186
2.14.9 函数: create_init_para.....	2187
2.14.10 函数: destroy_init_para.....	2188
2.14.11 函数: set_1n_top_num.....	2189
2.14.12 函数: set_nm_top_num.....	2189
2.14.13 函数: create_feature_info.....	2190
2.14.14 函数: destroy_feature_info.....	2191
2.14.15 函数: create_repo_range.....	2191
2.14.16 函数: destroy_repo_range.....	2192
2.14.17 函数: create_query_table.....	2192
2.14.18 函数: destroy_query_table.....	2193
2.14.19 函数: create_search_input.....	2194
2.14.20 函数: destroy_search_input.....	2194
2.14.21 函数: create_search_result.....	2195
2.14.22 函数: destroy_search_result.....	2196
2.15 Profiling 数据采集.....	2197
2.15.1 Profiling pyACL API (通过 Profiling pyACL API 采集并落盘性能数据)	2197
2.15.1.1 功能介绍.....	2197
2.15.1.2 函数: create_config.....	2198
2.15.1.3 函数: set_config.....	2199
2.15.1.4 函数: destroy_config.....	2202
2.15.1.5 函数: init.....	2202
2.15.1.6 函数: start.....	2203

2.15.1.7 函数: stop.....	2203
2.15.1.8 函数: finalize.....	2204
2.15.2 Profiling pyACL API for Extension (Profiling pyACL API 扩展接口)	2204
2.15.2.1 功能介绍.....	2204
2.15.2.2 函数: create_stamp.....	2205
2.15.2.3 函数: set_stamp_trace_message.....	2205
2.15.2.4 函数: mark.....	2206
2.15.2.5 函数: push.....	2206
2.15.2.6 函数: pop.....	2207
2.15.2.7 函数: range_start.....	2207
2.15.2.8 函数: range_stop.....	2208
2.15.2.9 函数: destroy_stamp.....	2208
2.15.3 Profiling pyACL API for Subscription (订阅算子信息的 Profiling pyACL API)	2209
2.15.3.1 功能介绍.....	2209
2.15.3.2 函数: create_subscribe_config.....	2210
2.15.3.3 函数: destroy_subscribe_config.....	2211
2.15.3.4 函数: model_subscribe.....	2211
2.15.3.5 函数: model_un_subscribe.....	2212
2.15.3.6 函数: model_unsubscribe.....	2212
2.15.3.7 函数: get_op_desc_size.....	2213
2.15.3.8 函数: get_op_num.....	2213
2.15.3.9 函数: get_op_type_len.....	2214
2.15.3.10 函数: get_op_name_len.....	2214
2.15.3.11 函数: get_op_type.....	2215
2.15.3.12 函数: get_op_type_v2.....	2215
2.15.3.13 函数: get_op_name.....	2216
2.15.3.14 函数: get_op_name_v2.....	2217
2.15.3.15 函数: get_op_start.....	2217
2.15.3.16 函数: get_op_end.....	2218
2.15.3.17 函数: get_op_duration.....	2218
2.15.3.18 函数: get_model_id.....	2219
2.15.4 Profiling pyACL API (PyTorch 场景标记迭代时间)	2219
2.15.4.1 函数: get_step_timestamp.....	2219
2.15.4.2 函数: create_step_info.....	2220
2.15.4.3 函数: destroy_step_info.....	2220
2.16 数据类型及其操作接口.....	2221
2.16.1 aclError.....	2221
2.16.2 aclDataType.....	2237
2.16.3 aclFloat16.....	2238
2.16.4 aclFormat.....	2238
2.16.5 aclrtMemMallocPolicy.....	2239
2.16.6 aclvppPixelFormat.....	2241

2.16.7 acldvppStreamFormat.....	2244
2.16.8 acldvppJpegFormat.....	2244
2.16.9 acldvppCscMatrix.....	2244
2.16.10 aclrtEventStatus.....	2246
2.16.11 aclrtRunMode.....	2247
2.16.12 aclTransType.....	2247
2.16.13 aclComputeType.....	2247
2.16.14 aclfvSearchType.....	2247
2.16.15 aclAippInputFormat.....	2248
2.16.16 aclAippInfo.....	2248
2.16.17 aclAippDims.....	2251
2.16.18 aclmdlIODims.....	2251
2.16.19 aclrtGroupAttr.....	2251
2.16.20 aclprofAicoreMetrics.....	2252
2.16.21 acldvppBorderType.....	2253
2.16.22 acldvppJpegdPrecisionMode.....	2254
2.16.23 aclmdlInputAippType.....	2254
2.16.24 acldvppChannelDescParamType.....	2254
2.16.25 aclvdecChannelDescParamType.....	2256
2.16.26 aclvencChannelDescParamType.....	2258
2.16.27 aclmdlConfigAttr.....	2259
2.16.28 aclmdlExecConfigAttr.....	2262
2.16.29 aclMemType.....	2262
2.16.30 aclrtEventWaitStatus.....	2263
2.16.31 aclrtEventRecordedStatus.....	2263
2.16.32 aclprofStepTag.....	2263
2.16.33 acldvppChannelMode.....	2264
2.16.34 aclrtFloatOverflowMode.....	2264
2.16.35 aclrtUtilizationInfo.....	2264
2.16.36 aclrtPhysicalMemProp.....	2265
2.16.37 aclrtStreamStatus.....	2265
2.16.38 aclrtDeviceStatus.....	2265
2.16.39 aclrtMemAttr.....	2266
2.16.40 aclrtMemHandleType.....	2266
2.16.41 aclDeviceInfo.....	2266
2.16.42 aclCannAttr.....	2266
2.16.43 aclrtMemGranularityOptions.....	2267
3 ISP API.....	2268
3.1 接口简介.....	2269
3.2 ISP 系统控制及 3A 算法注册.....	2271
3.2.1 hi_mpi_isp_sensor_reg_callback.....	2271
3.2.2 hi_mpi_isp_sensor_unreg_callback.....	2272

3.2.3 hi_mpi_isp_mem_init.....	2272
3.2.4 hi_mpi_isp_set_pub_attr.....	2273
3.2.5 hi_mpi_isp_get_pub_attr.....	2274
3.2.6 hi_mpi_isp_init.....	2275
3.2.7 hi_mpi_isp_run.....	2275
3.2.8 hi_mpi_isp_exit.....	2276
3.2.9 hi_mpi_isp_set_sns_slave_attr.....	2277
3.2.10 hi_mpi_isp_get_sns_slave_attr.....	2278
3.2.11 hi_mpi_ae_register.....	2278
3.2.12 hi_mpi_ae_unregister.....	2279
3.2.13 hi_mpi_ae_sensor_reg_callback.....	2279
3.2.14 hi_mpi_ae_sensor_unreg_callback.....	2280
3.2.15 hi_mpi_awb_register.....	2281
3.2.16 hi_mpi_awb_unregister.....	2281
3.2.17 hi_mpi_awb_sensor_reg_callback.....	2282
3.2.18 hi_mpi_awb_sensor_unreg_callback.....	2283
3.2.19 公共数据类型.....	2284
3.2.19.1 基本数据类型说明.....	2284
3.2.19.2 hi_video_field.....	2284
3.2.19.3 hi_pixel_format.....	2285
3.2.19.4 hi_video_format.....	2289
3.2.19.5 hi_compress_mode.....	2290
3.2.19.6 hi_dynamic_range.....	2290
3.2.19.7 hi_color_gamut.....	2292
3.2.19.8 hi_video_supplement.....	2292
3.2.19.9 hi_video_frame.....	2293
3.2.19.10 hi_mod_id.....	2295
3.2.19.11 hi_video_frame_info.....	2296
3.2.19.12 hi_rect.....	2297
3.2.19.13 hi_op_mode.....	2297
3.2.19.14 hi_point.....	2297
3.2.19.15 hi_size.....	2298
3.2.19.16 hi_wdr_mode.....	2298
3.2.19.17 hi_vi_pipe.....	2299
3.2.20 ISP 系统控制及 3A 算法注册数据类型.....	2299
3.2.20.1 hi_sensor_id.....	2299
3.2.20.2 hi_isp_sns_attr_info.....	2300
3.2.20.3 hi_isp_cmos_sensor_image_mode.....	2300
3.2.20.4 hi_isp_cmos_alg_key.....	2300
3.2.20.5 hi_isp_cmos_noise_calibration.....	2302
3.2.20.6 hi_isp_cmos_sensor_max_resolution.....	2303
3.2.20.7 hi_isp_dng_blc_repeat_dim.....	2303

3.2.20.8 hi_isp_dng_repeat_pattern_dim.....	2303
3.2.20.9 hi_isp_dng_srational.....	2304
3.2.20.10 hi_isp_dng_default_scale.....	2304
3.2.20.11 hi_isp_dng_cfa_layout_type.....	2305
3.2.20.12 hi_isp_dng_raw_format.....	2305
3.2.20.13 hi_isp_cmos_sensor_mode.....	2306
3.2.20.14 hi_isp_cmos_dng_color_param.....	2307
3.2.20.15 hi_isp_cmos_wdr_switch_attr.....	2307
3.2.20.16 hi_isp_cmos_default.....	2308
3.2.20.17 hi_isp_cmos_black_level.....	2310
3.2.20.18 hi_isp_sns_type.....	2310
3.2.20.19 hi_isp_sns_commbus.....	2311
3.2.20.20 hi_isp_i2c_data.....	2311
3.2.20.21 hi_isp_ssp_data.....	2313
3.2.20.22 hi_isp_sns_regs_info.....	2314
3.2.20.23 hi_isp_sensor_exp_func.....	2315
3.2.20.24 hi_isp_sensor_register.....	2317
3.2.20.25 hi_isp_bayer_format.....	2317
3.2.20.26 hi_isp_pub_attr.....	2317
3.2.20.27 hi_isp_frame_info.....	2318
3.2.20.28 hi_slave_dev.....	2320
3.2.20.29 hi_isp_slave_sns_sync.....	2320
3.2.20.30 hi_isp_3a_alg_lib.....	2323
3.2.20.31 hi_isp_ae_accuracy_type.....	2323
3.2.20.32 hi_isp_ae_accuracy.....	2324
3.2.20.33 hi_isp_ae_sensor_default.....	2325
3.2.20.34 hi_isp_ae_sensor_exp_func.....	2331
3.2.20.35 hi_isp_ae_sensor_register.....	2333
3.2.20.36 hi_isp_awb_agc_table.....	2333
3.2.20.37 hi_isp_awb_ccm_tab.....	2334
3.2.20.38 hi_isp_awb_ccm.....	2334
3.2.20.39 hi_isp_awb_sensor_default.....	2335
3.2.20.40 hi_isp_awb_sensor_exp_func.....	2336
3.2.20.41 hi_isp_awb_sensor_register.....	2337
3.3 ISP 系统控制高级参数配置.....	2337
3.3.1 hi_mpi_isp_run_once.....	2337
3.3.2 hi_mpi_isp_set_module_ctrl.....	2338
3.3.3 hi_mpi_isp_get_module_ctrl.....	2339
3.3.4 hi_mpi_isp_set_ctrl_param.....	2339
3.3.5 hi_mpi_isp_get_ctrl_param.....	2340
3.3.6 hi_mpi_isp_ae_lib_reg_callback.....	2340
3.3.7 hi_mpi_isp_ae_lib_unreg_callback.....	2341

3.3.8 hi_mpi_isp_awb_lib_reg_callback.....	2342
3.3.9 hi_mpi_isp_awb_lib_unreg_callback.....	2343
3.3.10 hi_mpi_isp_set_bind_attr.....	2343
3.3.11 hi_mpi_isp_get_bind_attr.....	2344
3.3.12 hi_mpi_isp_get_vd_time_out.....	2345
3.3.13 hi_mpi_isp_set_fmw_state.....	2345
3.3.14 hi_mpi_isp_get_fmw_state.....	2346
3.3.15 数据类型.....	2347
3.3.15.1 hi_isp_bayer_raw_bit.....	2347
3.3.15.2 hi_isp_fmw_state.....	2347
3.3.15.3 hi_isp_wdr_mode.....	2348
3.3.15.4 hi_isp_module_ctrl.....	2348
3.3.15.5 hi_isp_vd_type.....	2350
3.3.15.6 hi_isp_cmos_lsc.....	2351
3.3.15.7 hi_isp_acs_y_shading_lut.....	2352
3.3.15.8 hi_isp_acs_color_shading_lut.....	2352
3.3.15.9 hi_isp_acs_calib_param.....	2353
3.3.15.10 hi_isp_cmos_acs.....	2354
3.3.15.11 hi_isp_lsc_cabli_table.....	2354
3.3.15.12 hi_isp_cmos_expander_point.....	2355
3.3.15.13 hi_isp_expander_attr.....	2355
3.3.15.14 hi_isp_cmos_clut.....	2356
3.3.15.15 hi_isp_cmos_drc.....	2357
3.3.15.16 hi_isp_cmos_demosaic.....	2360
3.3.15.17 hi_isp_cmos_pregamma.....	2361
3.3.15.18 hi_isp_cmos_gamma.....	2362
3.3.15.19 hi_isp_cmos_sharpen_manual.....	2362
3.3.15.20 hi_isp_cmos_sharpen_auto.....	2365
3.3.15.21 hi_isp_cmos_sharpen.....	2369
3.3.15.22 hi_isp_cmos_edgemark.....	2370
3.3.15.23 hi_isp_cmos_hlc.....	2370
3.3.15.24 hi_isp_cmos_ldci.....	2371
3.3.15.25 hi_isp_cmos_dpc.....	2372
3.3.15.26 hi_isp_cmos_ge.....	2373
3.3.15.27 hi_isp_cmos_afc.....	2374
3.3.15.28 hi_isp_cmos_bayernr.....	2374
3.3.15.29 hi_isp_cmos_ca.....	2376
3.3.15.30 hi_isp_cmos_wdr.....	2377
3.3.15.31 hi_isp_cmos_dehaze.....	2379
3.3.15.32 hi_isp_cmos_lcac_manual.....	2379
3.3.15.33 hi_isp_cmos_lcac_auto.....	2380
3.3.15.34 hi_isp_cmos_lcac.....	2380

3.3.15.35 hi_isp_bnr_lsc_calib_table.....	2382
3.3.15.36 hi_isp_3a_alg_lib.....	2383
3.3.15.37 hi_isp_bind_attr.....	2383
3.3.15.38 hi_isp_ctrl_proc_write.....	2384
3.3.15.39 hi_isp_ctrl_cmd.....	2384
3.3.15.40 hi_isp_stitch_attr.....	2386
3.3.15.41 hi_isp_ae_register.....	2387
3.3.15.42 hi_isp_ae_exp_func.....	2387
3.3.15.43 hi_isp_ae_param.....	2388
3.3.15.44 hi_isp_fe_ae_stat_1.....	2389
3.3.15.45 hi_isp_fe_ae_stat_2.....	2390
3.3.15.46 hi_isp_fe_ae_stat_3.....	2390
3.3.15.47 hi_isp_fe_ae_stitch_stat_3.....	2391
3.3.15.48 hi_isp_ae_info.....	2391
3.3.15.49 hi_isp_ae_stat_attr.....	2394
3.3.15.50 hi_isp_ae_result.....	2396
3.3.15.51 hi_isp_awb_register.....	2399
3.3.15.52 hi_isp_awb_exp_func.....	2399
3.3.15.53 hi_isp_awb_param.....	2400
3.3.15.54 hi_isp_awb_stat_result.....	2401
3.3.15.55 hi_isp_awb_stat_1.....	2402
3.3.15.56 hi_isp_awb_info.....	2402
3.3.15.57 hi_isp_awb_raw_stat_attr.....	2403
3.3.15.58 hi_isp_awb_result.....	2405
3.3.15.59 hi_isp_dcf_update_info.....	2406
3.3.15.60 hi_isp_ctrl_param.....	2407
3.3.15.61 hi_isp_quick_start_param.....	2409
3.3.15.62 hi_isp_awb_alg.....	2410
3.3.15.63 hi_isp_init_attr.....	2410
3.4 AE.....	2412
3.4.1 AE 控制模块.....	2412
3.4.1.1 hi_mpi_isp_set_exposure_attr.....	2412
3.4.1.2 hi_mpi_isp_get_exposure_attr.....	2414
3.4.1.3 hi_mpi_isp_set_wdr_exposure_attr.....	2414
3.4.1.4 hi_mpi_isp_get_wdr_exposure_attr.....	2415
3.4.1.5 hi_mpi_isp_set_ae_route_attr.....	2415
3.4.1.6 hi_mpi_isp_get_ae_route_attr.....	2417
3.4.1.7 hi_mpi_isp_set_ae_route_attr_ex.....	2418
3.4.1.8 hi_mpi_isp_get_ae_route_attr_ex.....	2420
3.4.1.9 hi_mpi_isp_set_ae_route_sf_attr.....	2420
3.4.1.10 hi_mpi_isp_get_ae_route_sf_attr.....	2422
3.4.1.11 hi_mpi_isp_set_ae_route_sf_attr_ex.....	2422

3.4.1.12 hi_mpi_isp_get_ae_route_sf_attr_ex.....	2424
3.4.1.13 hi_mpi_isp_query_exposure_info.....	2425
3.4.1.14 hi_mpi_isp_set_exp_convert.....	2426
3.4.1.15 hi_mpi_isp_get_exp_convert.....	2426
3.4.2 AI 控制模块.....	2427
3.4.2.1 hi_mpi_isp_set_iris_attr.....	2427
3.4.2.2 hi_mpi_isp_get_iris_attr.....	2428
3.4.2.3 hi_mpi_isp_set_dciris_attr.....	2428
3.4.2.4 hi_mpi_isp_get_dciris_attr.....	2429
3.4.2.5 hi_mpi_isp_set_piris_attr.....	2430
3.4.2.6 hi_mpi_isp_get_piris_attr.....	2431
3.4.3 数据类型.....	2431
3.4.3.1 hi_isp_ae_mode.....	2431
3.4.3.2 hi_isp_ae_strategy.....	2432
3.4.3.3 hi_isp_ae_delay.....	2432
3.4.3.4 hi_isp_ae_range.....	2433
3.4.3.5 hi_isp_antiflicker_mode.....	2433
3.4.3.6 hi_isp_antiflicker.....	2434
3.4.3.7 hi_isp_subflicker.....	2435
3.4.3.8 hi_isp_fswdr_mode.....	2436
3.4.3.9 hi_isp_ae_fswdr_attr.....	2437
3.4.3.10 hi_isp_ae_attr.....	2437
3.4.3.11 hi_isp_me_attr.....	2442
3.4.3.12 hi_isp_exposure_attr.....	2443
3.4.3.13 hi_isp_wdr_exposure_attr.....	2444
3.4.3.14 hi_isp_ae_route_node.....	2446
3.4.3.15 hi_isp_ae_route.....	2448
3.4.3.16 hi_isp_ae_route_ex_node.....	2448
3.4.3.17 hi_isp_ae_route_ex.....	2450
3.4.3.18 hi_isp_exp_info.....	2451
3.4.3.19 hi_isp_prior_frame.....	2454
3.4.3.20 hi_isp_iris_status.....	2455
3.4.3.21 hi_isp_iris_type.....	2456
3.4.3.22 hi_isp_iris_f_no.....	2456
3.4.3.23 hi_isp_mi_attr.....	2457
3.4.3.24 hi_isp_dciris_attr.....	2458
3.4.3.25 hi_isp_piris_attr.....	2459
3.4.3.26 hi_isp_iris_attr.....	2464
3.4.3.27 hi_isp_exp_conv_param.....	2465
3.4.3.28 hi_isp_ae_convert_param.....	2466
3.4.3.29 hi_isp_ae_param_reg.....	2466
3.5 AWB.....	2467

3.5.1 概述.....	2467
3.5.2 AWB 控制模块.....	2467
3.5.2.1 hi_mpi_isp_set_wb_attr.....	2467
3.5.2.2 hi_mpi_isp_get_wb_attr.....	2468
3.5.2.3 hi_mpi_isp_set_awb_attr_ex.....	2468
3.5.2.4 hi_mpi_isp_get_awb_attr_ex.....	2469
3.5.2.5 hi_mpi_isp_query_wb_info.....	2469
3.5.2.6 hi_mpi_isp_cal_gain_by_temp.....	2470
3.5.3 数据类型.....	2471
3.5.3.1 hi_isp_awb_attr.....	2471
3.5.3.2 hi_isp_awb_cbcr_track_attr.....	2474
3.5.3.3 hi_isp_awb_lum_histogram_attr.....	2478
3.5.3.4 hi_isp_awb_alg_type.....	2479
3.5.3.5 hi_isp_awb_ct_limit_attr.....	2480
3.5.3.6 hi_isp_mwb_attr.....	2481
3.5.3.7 hi_isp_wb_attr.....	2482
3.5.3.8 hi_isp_awb_multi_ls_type.....	2483
3.5.3.9 hi_isp_awb_indoor_outdoor_status.....	2483
3.5.3.10 hi_isp_awb_attr_ex.....	2484
3.5.3.11 hi_isp_awb_extra_light_source_info.....	2487
3.5.3.12 hi_isp_awb_in_out_attr.....	2488
3.5.3.13 hi_isp_wb_info.....	2491
3.6 CCM.....	2492
3.6.1 概述.....	2492
3.6.2 hi_mpi_isp_set_saturation_attr.....	2493
3.6.3 hi_mpi_isp_get_saturation_attr.....	2493
3.6.4 hi_mpi_isp_set_ccm_attr.....	2494
3.6.5 hi_mpi_isp_get_ccm_attr.....	2495
3.6.6 数据类型.....	2495
3.6.6.1 hi_isp_saturation_attr.....	2495
3.6.6.2 hi_isp_saturation_manual.....	2496
3.6.6.3 hi_isp_saturation_auto.....	2497
3.6.6.4 hi_isp_color_matrix_attr.....	2498
3.6.6.5 hi_isp_color_matrix_auto.....	2498
3.6.6.6 hi_isp_color_matrix_param.....	2499
3.6.6.7 hi_isp_color_matrix_manual.....	2500
3.7 Auto Color Shading.....	2501
3.7.1 hi_mpi_isp_get_auto_color_shading_attr.....	2501
3.7.2 hi_mpi_isp_set_auto_color_shading_attr.....	2501
3.7.3 数据类型.....	2502
3.7.3.1 hi_isp_acs_attr.....	2502
3.8 查询内部状态信息（当前版本不支持）.....	2503

3.8.1 功能描述.....	2503
3.8.2 hi_mpi_isp_query_inner_state_info.....	2503
3.8.3 数据类型.....	2504
3.8.3.1 hi_isp_inner_state_info.....	2504
3.9 PreGamma.....	2506
3.9.1 hi_mpi_isp_get_pregamma_attr.....	2506
3.9.2 hi_mpi_isp_set_pregamma_attr.....	2506
3.9.3 数据类型.....	2507
3.9.3.1 hi_isp_pregamma_attr.....	2507
3.10 CLUT.....	2508
3.10.1 hi_mpi_isp_get_clut_coeff.....	2508
3.10.2 hi_mpi_isp_set_clut_coeff.....	2508
3.10.3 hi_mpi_isp_get_clut_attr.....	2509
3.10.4 hi_mpi_isp_set_clut_attr.....	2509
3.10.5 数据类型.....	2510
3.10.5.1 hi_isp_clut_attr.....	2510
3.10.5.2 hi_isp_clut_lut.....	2510
3.11 CSC.....	2513
3.11.1 hi_mpi_isp_set_csc_attr.....	2513
3.11.2 hi_mpi_isp_get_csc_attr.....	2513
3.11.3 数据类型.....	2514
3.11.3.1 hi_isp_csc_attr.....	2514
3.11.3.2 hi_isp_csc_matrix.....	2515
3.12 LDCI.....	2516
3.12.1 hi_mpi_isp_get_ldci_attr.....	2516
3.12.2 hi_mpi_isp_set_ldci_attr.....	2517
3.12.3 数据类型.....	2517
3.12.3.1 hi_isp_ldci_gauss_coef_attr.....	2517
3.12.3.2 hi_isp_ldci_he_wgt_attr.....	2518
3.12.3.3 hi_isp_ldci_manual_attr.....	2518
3.12.3.4 hi_isp_ldci_auto_attr.....	2519
3.12.3.5 hi_isp_ldci_attr.....	2519
3.13 ColorTone.....	2520
3.13.1 hi_mpi_isp_set_color_tone_attr.....	2520
3.13.2 hi_mpi_isp_get_color_tone_attr.....	2521
3.13.3 数据类型.....	2521
3.13.3.1 hi_isp_color_tone_attr.....	2521
3.14 WDR.....	2522
3.14.1 hi_mpi_isp_get_fswdr_attr.....	2523
3.14.2 hi_mpi_isp_set_fswdr_attr.....	2524
3.14.3 数据类型.....	2524
3.14.3.1 hi_isp_wdr_merge_mode.....	2525

3.14.3.2 hi_isp_fswdr_mdt_attr.....	2525
3.14.3.3 hi_isp_wdr_combine_attr.....	2526
3.14.3.4 hi_isp_fusion_attr.....	2527
3.14.3.5 hi_isp_fswdr_manual_attr.....	2528
3.14.3.6 hi_isp_fswdr_auto_attr.....	2528
3.14.3.7 hi_isp_wdr_fs_attr.....	2529
3.15 CAC.....	2529
3.15.1 hi_mpi_isp_get_global_cac_attr.....	2530
3.15.2 hi_mpi_isp_set_global_cac_attr.....	2530
3.15.3 hi_mpi_isp_get_local_cac_attr.....	2531
3.15.4 hi_mpi_isp_set_local_cac_attr.....	2531
3.15.5 数据类型.....	2532
3.15.5.1 hi_isp_depurplestr_manual_attr.....	2532
3.15.5.2 hi_isp_depurplestr_auto_attr.....	2532
3.15.5.3 hi_isp_local_cac_attr.....	2534
3.15.5.4 hi_isp_global_cac_attr.....	2536
3.16 CA.....	2537
3.16.1 hi_mpi_isp_get_ca_attr.....	2537
3.16.2 hi_mpi_isp_set_ca_attr.....	2538
3.16.3 数据类型.....	2538
3.16.3.1 hi_isp_ca_lut.....	2538
3.16.3.2 hi_isp_ca_attr.....	2539
3.17 去 FPN.....	2540
3.17.1 hi_mpi_isp_set_fpn_attr.....	2540
3.17.2 hi_mpi_isp_get_fpn_attr.....	2540
3.17.3 hi_mpi_isp_fpn_calibrate.....	2541
3.17.4 数据类型.....	2542
3.17.4.1 hi_isp_fpn_frame_info.....	2542
3.17.4.2 hi_isp_fpn_calibrate_attr.....	2543
3.17.4.3 hi_isp_fpn_attr.....	2543
3.17.4.4 hi_isp_fpn_manual_attr.....	2544
3.17.4.5 hi_isp_fpn_auto_attr.....	2544
3.17.4.6 hi_isp_fpn_type.....	2545
3.18 黑电平.....	2545
3.18.1 hi_mpi_isp_get_black_level_attr.....	2545
3.18.2 hi_mpi_isp_set_black_level_attr.....	2546
3.18.3 数据类型.....	2547
3.18.3.1 hi_isp_black_level_attr.....	2547
3.19 Demosaic.....	2547
3.19.1 hi_mpi_isp_get_demosaic_attr.....	2547
3.19.2 hi_mpi_isp_set_demosaic_attr.....	2548
3.19.3 数据类型.....	2548

3.19.3.1 hi_isp_demosaic_manual_attr.....	2548
3.19.3.2 hi_isp_demosaic_auto_attr.....	2550
3.19.3.3 hi_isp_demosaic_attr.....	2551
3.20 Anti FalseColor.....	2551
3.20.1 hi_mpi_isp_get_anti_false_color_attr.....	2551
3.20.2 hi_mpi_isp_set_anti_false_color_attr.....	2552
3.20.3 数据类型.....	2552
3.20.3.1 hi_isp_anti_false_color_manual_attr.....	2553
3.20.3.2 hi_isp_anti_false_color_auto_attr.....	2553
3.20.3.3 hi_isp_anti_false_color_attr.....	2554
3.21 NR.....	2554
3.21.1 hi_mpi_isp_get_nr_attr.....	2554
3.21.2 hi_mpi_isp_set_nr_attr.....	2555
3.21.3 数据类型.....	2555
3.21.3.1 hi_isp_nr_manual_attr.....	2555
3.21.3.2 hi_isp_nr_auto_attr.....	2556
3.21.3.3 hi_isp_nr_wdr_attr.....	2557
3.21.3.4 hi_isp_nr_attr.....	2558
3.22 Crosstalk Removal.....	2559
3.22.1 hi_mpi_isp_get_crosstalk_attr.....	2559
3.22.2 hi_mpi_isp_set_crosstalk_attr.....	2559
3.22.3 数据类型.....	2560
3.22.3.1 hi_isp_cr_attr.....	2560
3.23 Defect Pixel.....	2562
3.23.1 hi_mpi_isp_get_dp_dynamic_attr.....	2562
3.23.2 hi_mpi_isp_set_dp_dynamic_attr.....	2562
3.23.3 hi_mpi_isp_get_dp_static_attr.....	2563
3.23.4 hi_mpi_isp_set_dp_static_attr.....	2564
3.23.5 hi_mpi_isp_get_dp_calibrate.....	2564
3.23.6 hi_mpi_isp_set_dp_calibrate.....	2565
3.23.7 数据类型.....	2566
3.23.7.1 hi_isp_static_dp_type.....	2566
3.23.7.2 hi_isp_status.....	2566
3.23.7.3 hi_isp_dp_static_calibrate.....	2567
3.23.7.4 hi_isp_dp_static_attr.....	2569
3.23.7.5 hi_isp_dp_dynamic_manual_attr.....	2571
3.23.7.6 hi_isp_dp_dynamic_auto_attr.....	2572
3.23.7.7 hi_isp_dp_dynamic_attr.....	2573
3.24 Mesh Shading.....	2574
3.24.1 hi_mpi_isp_set_mesh_shading_attr.....	2574
3.24.2 hi_mpi_isp_get_mesh_shading_attr.....	2575
3.24.3 hi_mpi_isp_set_mesh_shading_gain_lut_attr.....	2575

3.24.4 hi_mpi_isp_get_mesh_shading_gain_lut_attr.....	2576
3.24.5 数据类型.....	2576
3.24.5.1 hi_isp_shading_attr.....	2577
3.24.5.2 hi_isp_shading_gain_lut.....	2577
3.24.5.3 hi_isp_bnr_lsc_gain_lut.....	2578
3.24.5.4 hi_isp_shading_lut_attr.....	2579
3.25 DRC.....	2581
3.25.1 hi_mpi_isp_set_drc_attr.....	2581
3.25.2 hi_mpi_isp_get_drc_attr.....	2582
3.25.3 数据类型.....	2582
3.25.3.1 hi_isp_drc_manual_attr.....	2582
3.25.3.2 hi_isp_drc_auto_attr.....	2583
3.25.3.3 hi_isp_drc_asymmetry_curve_attr.....	2583
3.25.3.4 hi_isp_drc_curve_select.....	2584
3.25.3.5 hi_isp_drc_attr.....	2585
3.26 Gamma.....	2588
3.26.1 hi_mpi_isp_get_gamma_attr.....	2588
3.26.2 hi_mpi_isp_set_gamma_attr.....	2588
3.26.3 数据类型.....	2589
3.26.3.1 hi_isp_gamma_attr.....	2589
3.26.3.2 hi_isp_gamma_curve_type.....	2590
3.27 HLC.....	2591
3.27.1 hi_mpi_isp_get_hlc_attr.....	2591
3.27.2 hi_mpi_isp_set_hlc_attr.....	2591
3.27.3 数据类型.....	2592
3.27.3.1 hi_isp_hlc_attr.....	2592
3.28 EdgeMark.....	2593
3.28.1 hi_mpi_isp_get_edge_mark_attr.....	2593
3.28.2 hi_mpi_isp_set_edge_mark_attr.....	2593
3.28.3 数据类型.....	2594
3.28.3.1 hi_isp_edge_mark_attr.....	2594
3.29 Sharpen.....	2595
3.29.1 hi_mpi_isp_set_sharpen_attr.....	2595
3.29.2 hi_mpi_isp_get_sharpen_attr.....	2595
3.29.3 数据类型.....	2596
3.29.3.1 hi_isp_sharpen_manual_attr.....	2596
3.29.3.2 hi_isp_sharpen_auto_attr.....	2600
3.29.3.3 hi_isp_sharpen_attr.....	2604
3.30 Dehaze.....	2605
3.30.1 hi_mpi_isp_get_dehaze_attr.....	2605
3.30.2 hi_mpi_isp_set_dehaze_attr.....	2606
3.30.3 数据类型.....	2606

3.30.3.1 hi_isp_dehaze_manual_attr.....	2606
3.30.3.2 hi_isp_dehaze_auto_attr.....	2607
3.30.3.3 hi_isp_dehaze_attr.....	2607
3.31 Auto Flicker Type Detection.....	2608
3.31.1 功能描述.....	2608
3.31.2 hi_mpi_isp_calc_flicker_type.....	2608
3.31.3 数据类型.....	2609
3.31.3.1 hi_isp_calc_flicker_input.....	2610
3.31.3.2 hi_isp_calc_flicker_output.....	2610
3.31.3.3 hi_isp_calc_flicker_type.....	2611
3.32 MotionFusion.....	2611
3.32.1 hi_mpi_mfusion_set_attr.....	2611
3.32.2 数据类型.....	2612
3.32.2.1 hi_mfusion_steady_detect_attr.....	2612
3.32.2.2 hi_mfusion_attr.....	2614
3.33 统计信息.....	2614
3.33.1 概述.....	2614
3.33.2 AE 统计信息.....	2614
3.33.2.1 AE 区域统计信息.....	2614
3.33.2.2 带权重的 AE 全局统计信息.....	2615
3.33.2.3 带权重的 AE 直方图信息.....	2615
3.33.3 MG 统计信息.....	2616
3.33.3.1 MG 区域统计信息.....	2616
3.33.4 AWB 统计信息.....	2617
3.33.4.1 AWB 区域统计信息.....	2618
3.33.4.2 AWB 全局统计信息.....	2618
3.33.5 API 参考.....	2618
3.33.5.1 hi_mpi_isp_set_stats_cfg.....	2618
3.33.5.2 hi_mpi_isp_get_stats_cfg.....	2619
3.33.5.3 hi_mpi_isp_get_ae_stats.....	2620
3.33.5.4 hi_mpi_isp_get_mg_stats.....	2620
3.33.5.5 hi_mpi_isp_get_wb_stats.....	2621
3.33.6 数据类型.....	2621
3.33.6.1 hi_isp_stats_cfg.....	2621
3.33.6.2 hi_isp_stats_ctrl.....	2622
3.33.6.3 hi_isp_ae_stats_cfg.....	2623
3.33.6.4 hi_isp_ae_switch.....	2625
3.33.6.5 hi_isp_ae_four_plane_mode.....	2625
3.33.6.6 hi_isp_ae_hist_skip.....	2626
3.33.6.7 hi_isp_ae_hist_offset_x.....	2626
3.33.6.8 hi_isp_ae_hist_offset_y.....	2627
3.33.6.9 hi_isp_ae_hist_config.....	2627

3.33.6.10 hi_isp_ae_stat_mode.....	2628
3.33.6.11 hi_isp_ae_crop.....	2628
3.33.6.12 hi_isp_awb_switch.....	2629
3.33.6.13 hi_isp_awb_gain_switch.....	2630
3.33.6.14 hi_isp_awb_crop.....	2630
3.33.6.15 hi_isp_wb_stats_cfg.....	2631
3.33.6.16 hi_isp_ae_grid_info.....	2632
3.33.6.17 hi_isp_ae_stats.....	2633
3.33.6.18 hi_isp_ae_stitch_stats.....	2634
3.33.6.19 hi_isp_mg_grid_info.....	2636
3.33.6.20 hi_isp_mg_stats.....	2636
3.33.6.21 hi_isp_awb_grid_info.....	2637
3.33.6.22 hi_isp_wb_stats.....	2638
3.33.6.23 hi_isp_wb_stitch_stats.....	2639
3.33.6.24 hi_isp_focus_stats_cfg.....	2640
3.33.6.25 hi_isp_af_cfg.....	2641
3.33.6.26 hi_isp_af_peak_mode.....	2642
3.33.6.27 hi_isp_af_squ_mode.....	2643
3.33.6.28 hi_isp_af_crop.....	2644
3.33.6.29 hi_isp_af_stats_pos.....	2645
3.33.6.30 hi_isp_af_raw_cfg.....	2646
3.33.6.31 hi_isp_af_pre_filter_cfg.....	2646
3.33.6.32 hi_isp_af_level_depend.....	2646
3.33.6.33 hi_isp_af_coring.....	2647
3.33.6.34 hi_isp_af_h_param.....	2648
3.33.6.35 hi_isp_af_v_param.....	2648
3.33.6.36 hi_isp_af_fv_param.....	2649
3.33.6.37 hi_isp_focus_zone.....	2650
3.33.6.38 hi_isp_fe_focus_stats.....	2651
3.33.6.39 hi_isp_be_focus_stats.....	2651
3.33.6.40 hi_isp_focus_grid_info.....	2651
3.33.6.41 hi_isp_af_stats.....	2652
3.34 错误码.....	2653
4 OpenHiva API.....	2654
4.1 接口简介.....	2654
4.2 返回码列表.....	2659
4.3 类型定义.....	2662
4.3.1 OpenHiva::Node 类型定义.....	2663
4.3.2 OpenHiva::TopicOptions 类型定义.....	2663
4.3.3 OpenHiva::Publisher 类型定义.....	2664
4.3.4 OpenHiva::Subscriber 类型定义.....	2664
4.3.5 OpenHiva::ServiceOptions 类型定义.....	2664

4.3.6 OpenHiva::ServiceServer 类型定义.....	2665
4.3.7 OpenHiva::ServiceClient 类型定义.....	2665
4.3.8 OpenHiva::ScheduleGroup 类型定义.....	2665
4.3.9 OpenHiva::BufferMessage 类型定义.....	2665
4.3.10 OpenHiva::HivaBuffer 类型定义.....	2666
4.3.11 OpenHiva::HivaBufferPool 类型定义.....	2667
4.3.12 OpenHiva::HivaMessage 类型定义.....	2668
4.3.13 OpenHiva::HivaMessageTraits 类型定义.....	2668
4.3.14 Hiva::Time 类型定义.....	2668
4.3.15 Hiva::Duration 类型定义.....	2670
4.3.16 Hiva::Rate 类型定义.....	2671
4.4 初始化、退出和调度接口.....	2672
4.4.1 OpenHiva::Init.....	2672
4.4.2 OpenHiva::Ready.....	2673
4.4.3 OpenHiva::Shutdown.....	2673
4.4.4 OpenHiva::WaitForShutdown.....	2674
4.4.5 OpenHiva::SpinOnce.....	2674
4.5 OpenHiva::Node 接口.....	2675
4.5.1 OpenHiva::Node.....	2675
4.5.2 OpenHiva::Node::CreatePublisher.....	2676
4.5.3 OpenHiva::Node::CreateSubscriber.....	2677
4.5.4 OpenHiva::Node::CreateServer.....	2678
4.5.5 OpenHiva::Node::CreateClient.....	2679
4.5.6 OpenHiva::Node::Ready.....	2680
4.5.7 OpenHiva::Node::GetName.....	2680
4.5.8 OpenHiva::Node::GetNameSpace.....	2681
4.6 OpenHiva::TopicOptions 接口.....	2681
4.6.1 OpenHiva::TopicOptions::BuildShmOptions.....	2681
4.6.2 OpenHiva::TopicOptions::BuildQueueOptions.....	2682
4.6.3 OpenHiva::TopicOptions::BuildGroupName.....	2683
4.6.4 OpenHiva::TopicOptions::BuildTransportOptions.....	2683
4.6.5 OpenHiva::TopicOptions::BuildDDSQos.....	2684
4.7 OpenHiva::Publisher 接口.....	2686
4.7.1 OpenHiva::Publisher::Publisher.....	2686
4.7.2 OpenHiva::Publisher::Publish.....	2686
4.7.3 OpenHiva::Publisher::Ready.....	2687
4.8 OpenHiva::Subscriber 接口.....	2688
4.8.1 OpenHiva::Subscriber::Subscriber.....	2688
4.8.2 OpenHiva::Subscriber::Ready.....	2688
4.8.3 OpenHiva::Subscriber::StartReadMessage.....	2689
4.8.4 OpenHiva::Subscriber::StopReadMessage.....	2689
4.9 OpenHiva::ServiceServer 接口.....	2690

4.9.1 OpenHiva::ServiceServer::Enable.....	2690
4.9.2 OpenHiva::ServiceServer::Disable.....	2691
4.9.3 OpenHiva::ServiceServer::Destroy.....	2691
4.10 OpenHiva::ServiceClient 接口.....	2692
4.10.1 OpenHiva::ServiceClient::InvokeService.....	2692
4.10.2 OpenHiva::ServiceClient::Destroy.....	2693
4.11 NnEngine 接口.....	2693
4.11.1 Hiva::CreateNnEngine.....	2693
4.11.2 Hiva::DestroyNnEngine.....	2695
4.12 定时器接口.....	2696
4.12.1 Hiva::CreateTimer.....	2696
4.12.2 Hiva::Timer::Start.....	2697
4.12.3 Hiva::Timer::IsValid.....	2698
4.12.4 Hiva::Timer::SetPeriod.....	2698
4.12.5 Hiva::Timer::Stop.....	2699
4.12.6 Hiva::CreateWallTimer.....	2700
4.12.7 Hiva::WallTimer::Start.....	2701
4.12.8 Hiva::WallTimer::IsValid.....	2702
4.12.9 Hiva::WallTimer::SetPeriod.....	2702
4.12.10 Hiva::WallTimer::Stop.....	2703
4.12.11 Hiva::CreateSteadyTimer.....	2704
4.12.12 Hiva::SteadyTimer::Start.....	2705
4.12.13 Hiva::SteadyTimer::IsValid.....	2705
4.12.14 Hiva::SteadyTimer::SetPeriod.....	2706
4.12.15 Hiva::SteadyTimer::Stop.....	2707
4.13 配置管理相关接口.....	2707
4.13.1 约束与限制.....	2707
4.13.2 CfgMgr::BatchReset.....	2708
4.13.3 CfgMgr::CancelMonitor.....	2709
4.13.4 CfgMgr::CleanStartConfig.....	2710
4.13.5 CfgMgr::CreateSnapshot.....	2710
4.13.6 CfgMgr::CreateSnapshotToFile.....	2711
4.13.7 CfgMgr::DeleteSnapshot.....	2712
4.13.8 CfgMgr::Get.....	2713
4.13.9 CfgMgr::GetAllStartConfig.....	2714
4.13.10 CfgMgr::GetAllSnapshot.....	2715
4.13.11 CfgMgr::GetStartConfig.....	2715
4.13.12 CfgMgr::Has.....	2716
4.13.13 CfgMgr::Load.....	2717
4.13.14 CfgMgr::Monitor.....	2718
4.13.15 CfgMgr::Param.....	2719
4.13.16 CfgMgr::Reload.....	2720

4.13.17 CfgMgr::Reset.....	2721
4.13.18 CfgMgr::RevertSnapshot.....	2722
4.13.19 CfgMgr::RevertSnapshotFromFile.....	2723
4.13.20 CfgMgr::Set.....	2723
4.13.21 CfgMgr::SetNextStartConfig.....	2724
4.13.22 CfgMgr::SetUpgrade.....	2725

1 AscendCL API (C&C++)

- 1.1 接口列表
- 1.2 废弃接口/返回码列表
- 1.3 同步&异步API说明
- 1.4 系统配置
- 1.5 运行时管理
- 1.6 单算子调用
- 1.7 模型推理
- 1.8 媒体数据处理V1
- 1.9 媒体数据处理V2
- 1.10 Profiling数据采集
- 1.11 数据传输
- 1.12 特征向量检索
- 1.13 数据类型及其操作接口

1.1 接口列表

1.1.1 接口版本说明

V1.1.0为第一个AscendCL接口版本，根据AscendCL接口的变更，该版本号会随之改变，您可以调用[aclrtGetVersion](#)接口获取AscendCL接口版本号、再通过本章的接口列表查阅变更接口。

AscendCL接口版本号命名采用：A.B.C模式，其中：

- A表示有不兼容修改，从1开始，如果出现接口的不兼容变更时，加1；
- B表示新增接口，按照迭代周期，有新增接口时加1；
- C表示bug修复，从0开始，表示本版本仅解决了问题。在A、B不变的情况下，如果本版本修复了问题，则C加1；但A、B增加的时候，C一般为0。

 说明

昇腾AI处理器对枚举值、数据类型及其操作接口的支持情况请参见[1.9.20 枚举值及数据结构](#)、[1.13 数据类型及其操作接口](#)。

接口的返回码说明请参见[1.13.1 aclError](#)，如果接口调用失败，则接口的出参无效，不可继续使用。

接口命名规则的说明请参见表达约定。

1.1.2 V1.1.0 及以上版本

各版本对AscendCL接口的支持情况如下所示，各标识的含义如下：

- √：支持。
- x：不支持。
- 废弃：后续版本待废弃，具体废弃接口说明请参见[1.2 废弃接口/返回码列表](#)。

目录	接口	Atlas 200/300/500 推理产品	Atlas 训 练系列产 品	Atlas 推 理系列产 品
系统配置	acclnit	√	√	√
	aclFinalize	√	√	√
	aclrtGetVersion	√	√	√
	aclSetCompileopt	√	√	√
	aclGetRecentErrMsg	√	√	√
	aclAppLog	√	√	√
	aclDataTypeSize	√	√	√
	aclFloat16ToFloat	√	√	√
	aclFloatToFloat16	√	√	√
	aclGenGraphAndDumpForOp	√	√	√
运行时管理 >Device管 理	aclrtSetDevice	√	√	√
	aclrtResetDevice	√	√	√
	aclrtGetDevice	√	√	√
	aclrtGetRunMode	√	√	√
	aclrtSetTsDevice	√	√	√
	aclrtGetDeviceCount	√	√	√
	aclrtSynchronizeDevice	√	√	√

目录	接口	Atlas 200/300/500 推理产品	Atlas 训 练系列产 品	Atlas 推 理系列产 品
	aclrtGetSocName	√	√	√
运行时管理 >Context管 理	aclrtCreateContext	√	√	√
	aclrtDestroyContext	√	√	√
	aclrtSetCurrentCont ext	√	√	√
	aclrtGetCurrentCon text	√	√	√
运行时管理 >Stream管 理	aclrtCreateStream	√	√	√
	aclrtDestroyStream	√	√	√
	aclrtSynchronizeStr eam	√	√	√
运行时管理 >Event管理	aclrtCreateEvent	√	√	√
	aclrtCreateEventWit hFlag	√	√	√
	aclrtDestroyEvent	√	√	√
	aclrtRecordEvent	√	√	√
	aclrtResetEvent	√	√	√
	aclrtQueryEvent	废弃	废弃	废弃
	aclrtQueryEventSta tus	√	√	√
	aclrtQueryEventWai tStatus	√	√	√
	aclrtSynchronizeEve nt	√	√	√
	aclrtEventElapsedTi me	√	√	√
	aclrtStreamWaitEve nt	√	√	√
	aclrtSetOpWaitTim eout	√	√	x
运行时管理 >内存管理	aclrtMalloc	√	√	√
	aclrtMallocCached	√	√	√
	aclrtMemFlush	√	√	√

目录	接口	Atlas 200/300/500 推理产品	Atlas 训 练系列产 品	Atlas 推 理系列产 品
	aclrtMemInvalidate	√	√	√
	aclrtFree	√	√	√
	aclrtMallocHost	√	√	√
	aclrtFreeHost	√	√	√
	aclrtMemset	√	√	√
	aclrtMemsetAsync	√	√	√
	aclrtMemcpy	√	√	√
	aclrtMemcpyAsync	√	√	√
	aclrtGetMemInfo	√	√	√
	aclrtDeviceCanAccessPeer	x	√	√
	aclrtDeviceEnablePeerAccess	x	√	√
	aclrtDeviceDisablePeerAccess	x	√	√
	aclrtMemcpy2d	√	√	仅Atlas 推理系列加速卡产品支持该接口，Atlas 推理系列加速模块产品不支持该接口。
	aclrtMemcpy2dAsync	√	√	仅Atlas 推理系列加速卡产品支持该接口，Atlas 推理系列加速模块产品不支持该接口。
运行时管理 >执行控制	aclrtSubscribeReport	√	√	√

目录	接口	Atlas 200/300/500 推理产品	Atlas 训 练系列产 品	Atlas 推 理系列产 品
	aclrtLaunchCallbac k	√	√	√
	aclrtProcessReport	√	√	√
	aclrtUnSubscribeRe port	√	√	√
运行时管理 >异常处理	aclrtSetExceptionInf oCallback	√	√	√
	aclrtGetTaskIdFrom ExceptionInfo	√	√	√
	aclrtGetStreamIdFr omExceptionInfo	√	√	√
	aclrtGetThreadIdFro mExceptionInfo	√	√	√
	aclrtGetDevicIdFro mExceptionInfo	√	√	√
运行时管理 >算力Group 查询与设置	aclrtSetGroup	x	x	x
	aclrtGetGroupCoun t	x	x	x
	aclrtCreateGroupInf o	x	x	x
	aclrtDestroyGroupI nfo	x	x	x
	aclrtGetAllGroupInf o	x	x	x
	aclrtGetGroupInfoD etail	x	x	x
模型推理> 模型加载	aclmdlLoadFromFil e	√	√	√
	aclmdlLoadFromMe m	√	√	√
	aclmdlLoadFromFil eWithMem	√	√	√
	aclmdlLoadFromMe mWithMem	√	√	√
	aclmdlLoadFromFil eWithQ	x	x	x

目录	接口	Atlas 200/300/500 推理产品	Atlas 训 练系列产 品	Atlas 推 理系列产 品
	acldmLoadFromMemWithQ	x	x	x
	acldmSetConfigOpt	√	√	√
	acldmLoadWithConfig	√	√	√
模型推理> 模型执行	acldmExecute	√	√	√
	acldmExecuteAsync	√	√	√
	acldmUnload	√	√	√
	acldmQuerySize	√	√	√
	acldmQuerySizeFromMem	√	√	√
	acldmSetDynamicBatchSize	√	√	√
	acldmSetDynamicHWSIZE	√	√	√
	acldmSetInputAIPP	√	x	√
	acldmGetFirstAippInfo	√	x	√
	acldmGetAippType	√	x	√
	acldmSetAIPPByInputIndex	√	x	√
	acldmSetInputDynamicDims	√	√	√
	acldmCreateAndGetOpDesc	√	√	√
	acldmInitDump	√	√	√
	acldmSetDump	√	√	√
	acldmFinalizeDump	√	√	√
单算子执行> 单算子模型执行	aclopSetModelDir	√	√	√
	aclopLoad	√	√	√

目录	接口	Atlas 200/300/500 推理产品	Atlas 训 练系列产 品	Atlas 推 理系列产 品
	aclopSetCompileFlag	x	√	√
	aclopCompile	x	√	√
	aclopExecute	废弃	废弃	废弃
	aclopExecuteV2	√	√	√
	aclopCompileAndExecute	√	√	√
	aclopExecWithHandle	√	√	√
	aclopInferShape	√	√	√
	aclopRegisterCompileFunc	√	√	√
	aclopUnregisterCompileFunc	√	√	√
	aclopCreateKernel	√	√	√
	aclopSetKernelArgs	√	√	√
	aclopSetKernelWorkspaceSizes	√	√	√
	aclopUpdateParams	√	√	√
单算子执行 >CBLAS接口	aclblasGemvEx	√	√	√
	aclblasCreateHandleForGemvEx	√	√	√
	aclblasHgemv	√	√	√
	aclblasCreateHandleForHgemv	√	√	√
	aclblasS8gemv	√	√	√
	aclblasCreateHandleForS8gemv	√	√	√
	aclblasGemmEx	√	√	√
	aclblasCreateHandleForGemmEx	√	√	√
	aclblasHgemm	√	√	√

目录	接口	Atlas 200/300/500 推理产品	Atlas 训 练系列产 品	Atlas 推 理系列产 品
	aclblasCreateHandleForHgemm	√	√	√
	aclblasS8gemm	√	√	√
	aclblasCreateHandleForS8gemm	√	√	√
	aclopCast	√	√	√
	aclopCreateHandleForCast	√	√	√
媒体数据处理V1>内存 申请与释放	acldvppMalloc	√	√	√
	acldvppFree	√	√	√
媒体数据处理V1>通道 创建与释放	acldvppCreateChannel	√	√	√
	acldvppDestroyChannel	√	√	√
	acldvdecCreateChannel	√	√	√
	acldvdecDestroyChannel	√	√	√
	acldvencCreateChannel	√	√	√
	acldvencDestroyChannel	√	√	√
媒体数据处理V1>VPC 功能	acldvppVpcResizeAsync	√	√	√
	acldvppVpcCropAsync	√	√	√
	acldvppVpcCropResizeAsync	√	√	√
	acldvppVpcBatchCropAsync	√	√	√
	acldvppVpcBatchCropResizeAsync	√	√	√
	acldvppVpcCropAndPasteAsync	√	√	√

目录	接口	Atlas 200/300/500 推理产品	Atlas 训 练系列产 品	Atlas 推 理系列产 品
	acldvppVpcCropResi zePasteAsync	√	√	√
	acldvppVpcBatchCr opAndPasteAsync	√	√	√
	acldvppVpcBatchCr opResizePasteAsync	√	√	√
	acldvppVpcBatchCr opResizeMakeBord erAsync	x	x	√
	acldvppVpcConvert ColorAsync	x	x	√
	acldvppVpcPyrDow nAsync	x	x	√
	acldvppVpcEqualize HistAsync	x	x	√
	acldvppVpcMakeBo rderAsync	x	x	√
	acldvppVpcCalcHist Async	x	x	√
媒体数据处 理V1>JPEGD 功能	acldvppJpegDecode Async	√	√	√
	acldvppJpegGetIma gelInfo	√	√	√
	acldvppJpegGetIma gelInfoV2	√	√	√
	acldvppJpegPredict DecSize	√	√	√
媒体数据处 理V1>JPEG 功能	acldvppJpegEncode Async	√	√	√
	acldvppJpegPredict EncSize	√	√	√
媒体数据处 理V1>PNGD 功能	acldvppPngDecode Async	√	√	x
	acldvppPngGetIma gelInfo	√	√	x
	acldvppPngPredictD ecSize	√	√	x

目录	接口	Atlas 200/300/500 推理产品	Atlas 训 练系列产 品	Atlas 推 理系列产 品
媒体数据处 理V1>VDEC 功能	aclvdecSendFrame	√	√	√
	aclvdecSendSkippe dFrame	√	√	√
	aclvdecCallback	√	√	√
媒体数据处 理V1>VENC 功能	aclvencSendFrame	√	x	√
	aclvencCallback	√	x	√
媒体数据处 理V2>公共 接口	hi_mpi_sys_init	x	x	√
	hi_mpi_sys_exit	x	x	√
	hi_mpi_dvpp_mallo c	x	x	√
	hi_mpi_dvpp_free	x	x	√
	hi_mpi_dvpp_get_i mage_info	x	x	√
	hi_mpi_dvpp_get_ve rsion	x	x	√
	hi_mpi_sys_create_e poll	x	x	√
	hi_mpi_sys_ctl_epoll	x	x	√
	hi_mpi_sys_wait_ep oll	x	x	√
	hi_mpi_sys_close_e poll	x	x	√
	hi_mpi_sys_set_chn _csc_matrix	x	x	√
	hi_mpi_sys_get_chn _csc_matrix	x	x	√
媒体数据处 理V2>VPC 功能	hi_mpi_vpc_create_ chn	x	x	√
	hi_mpi_vpc_destroy _chn	x	x	√
	hi_mpi_vpc_resize	x	x	√
	hi_mpi_vpc_crop	x	x	√
	hi_mpi_vpc_crop_re size	x	x	√

目录	接口	Atlas 200/300/500 推理产品	Atlas 训 练系列产 品	Atlas 推 理系列产 品
	hi_mpi_vpc_crop_re size_paste	x	x	√
	hi_mpi_vpc_convert _color	x	x	√
	hi_mpi_vpc_convert _color_to_yuv420	x	x	√
	hi_mpi_vpc_copy_m ake_border	x	x	√
	hi_mpi_vpc_pyrdow n	x	x	√
	hi_mpi_vpc_calc_his t	x	x	√
	hi_mpi_vpc_equaliz e_hist	x	x	√
	hi_mpi_vpc_crop_re size_make_border	x	x	√
	hi_mpi_vpc_batch_c rop_resize_paste	x	x	√
	hi_mpi_vpc_batch_c rop_resize_make_bo rder	x	x	√
	hi_mpi_vpc_get_pro cess_result	x	x	√
	hi_mpi_vpc_sys_cre ate_chn	x	x	√
媒体数据处 理V2>VDEC 功能/JPEGD 功能	hi_mpi_vdec_create _chn	x	x	√
	hi_mpi_vdec_destro y_chn	x	x	√
	hi_mpi_vdec_get_ch n_attr	x	x	√
	hi_mpi_vdec_set_ch n_attr	x	x	x
	hi_mpi_vdec_start_r ecv_stream	x	x	√
	hi_mpi_vdec_stop_r ecv_stream	x	x	√

目录	接口	Atlas 200/300/500 推理产品	Atlas 训 练系列产 品	Atlas 推 理系列产 品
	hi_mpi_vdec_query_status	x	x	√
	hi_mpi_vdec_reset_chn	x	x	√
	hi_mpi_vdec_set_chn_param	x	x	√
	hi_mpi_vdec_get_chn_param	x	x	√
	hi_mpi_vdec_set_protocol_param	x	x	x
	hi_mpi_vdec_get_protocol_param	x	x	√
	hi_mpi_vdec_send_stream	x	x	√
	hi_mpi_vdec_get_frame	x	x	√
	hi_mpi_vdec_release_frame	x	x	√
	hi_mpi_vdec_get_fd	x	x	√
	hi_mpi_vdec_close_fd	x	x	√
	hi_vdec_get_pic_buf_size	x	x	√
	hi_vdec_get_tmv_buf_size	x	x	√
媒体数据处理V2>VENC 功能/JPEGE 功能	hi_mpi_venc_create_chn	x	x	√
	hi_mpi_venc_destroy_chn	x	x	√
	hi_mpi_venc_start_chn	x	x	√
	hi_mpi_venc_stop_chn	x	x	√
	hi_mpi_venc_query_status	x	x	√

目录	接口	Atlas 200/300/500 推理产品	Atlas 训 练系列产 品	Atlas 推 理系列产 品
	hi_mpi_venc_get_stream	x	x	√
	hi_mpi_venc_release_stream	x	x	√
	hi_mpi_venc_send_frame	x	x	√
	hi_mpi_venc_set_mode_param	x	x	√
	hi_mpi_venc_get_mode_param	x	x	√
	hi_mpi_venc_request_idr	x	x	√
	hi_mpi_venc_get_fd	x	x	√
	hi_mpi_venc_close_fd	x	x	x
	hi_mpi_venc_set_jpeg_param	x	x	√
	hi_mpi_venc_get_jpeg_param	x	x	√
	hi_mpi_venc_set_chn_param	x	x	√
	hi_mpi_venc_set_scene_mode	x	x	√
	hi_mpi_venc_set_rc_param	x	x	√
	hi_mpi_venc_get_rc_param	x	x	√
	hi_mpi_venc_set_jpeg_huffman_param	x	x	√
	hi_mpi_venc_get_jpeg_huffman_param	x	x	√
	hi_mpi_venc_compact_jpeg_tables	x	x	√
媒体数据处理V2>PNGD 功能	hi_mpi_pngd_create_chn	x	x	√

目录	接口	Atlas 200/300/500 推理产品	Atlas 训 练系列产 品	Atlas 推 理系列产 品
	hi_mpi_pngd_destro y_chn	x	x	√
	hi_mpi_pngd_send_ stream	x	x	√
	hi_mpi_pngd_get_i mage_data	x	x	√
	hi_mpi_png_get_im age_info	x	x	√
Profiling数 据采集 >Profiling AscendCL API (通过 Profiling AscendCL API采集并落 盘性能数 据)	aclprofInit	√	√	√
	aclprofStart	√	√	√
	aclprofStop	√	√	√
	aclprofFinalize	√	√	√
Profiling AscendCL API for Extension (Profiling AscendCL API扩展接 口)	aclprofCreateStamp	√	√	√
	aclprofSetStampTra ceMessage	√	√	√
	aclprofMark	√	√	√
	aclprofPush	√	√	√
	aclprofPop	√	√	√
	aclprofRangeStart	√	√	√
	aclprofRangeStop	√	√	√
	aclprofDestroyStam p	√	√	√
Profiling AscendCL API for Subscriptio n (订阅算 子信息的 Profiling AscendCL API)	aclprofModelSubscr ibe	√	√	√
	aclprofModelUnSub scribe	√	√	√
	aclprofGetOpDescSi ze	√	√	√
	aclprofGetOpNum	√	√	√
	aclprofGetOpTypeL en	√	√	√

目录	接口	Atlas 200/300/500 推理产品	Atlas 训 练系列产 品	Atlas 推 理系列产 品
	aclprofGetOpType	√	√	√
	aclprofGetOpName Len	√	√	√
	aclprofGetOpName	√	√	√
	aclprofGetOpStart	√	√	√
	aclprofGetOpEnd	√	√	√
	aclprofGetOpDurati on	√	√	√
	aclprofGetModelId	√	√	√
Profiling AscendCL API (PyTorch场 景标记迭代 时间)	aclprOFGETSTEPTI MESTAMp	√	√	√
数据传输 >Tensor数 据传输	acldtdCreateChanne l	x	√	x
	acldtdSendTensor	x	√	x
	acldtdReceiveTensor	x	√	x
	acldtdStopChannel	x	√	x
	acldtdDestroyChann el	x	√	x
	acldtdCreateChanne lWithCapacity	x	√	x
	acldtdQueryChannel Size	x	√	x
数据传输> 共享队列管 理	acldtdCreateQueue	√	√	√
	acldtdDestroyQueue	√	√	√
	acldtdEnqueueData	√	√	√
	acldtdDequeueData	√	√	√
	acldtdEnqueue	x	x	x
	acldtdDequeue	x	x	x
	acldtdBindQueueRo utes	√	√	√

目录	接口	Atlas 200/300/500 推理产品	Atlas 训 练系列产 品	Atlas 推 理系列产 品
	acldtdUnbindQueueRoutes	√	√	√
	acldtdQueryQueueRoutes	√	√	√
	acldtdGrantQueue	x	x	x
	acldtdAttachQueue	x	x	x
数据传输> 共享Buffer 管理	acldtdAllocBuf	√	√	√
	acldtdFreeBuf	√	√	√
	acldtdGetBufData	√	√	√
特征向量检索	aclfvInit	x	x	仅Atlas 推理系列加速卡产品支持该接口，Atlas 推理系列加速模块产品不支持该接口。
	aclfvRelease	x	x	仅Atlas 推理系列加速卡产品支持该接口，Atlas 推理系列加速模块产品不支持该接口。
	aclfvRepoAdd	x	x	仅Atlas 推理系列加速卡产品支持该接口，Atlas 推理系列加速模块产品不支持该接口。

目录	接口	Atlas 200/300/500 推理产品	Atlas 训 练系列产 品	Atlas 推 理系列产 品
	aclfvRepoDel	x	x	仅Atlas 推 理系列加 速卡产品 支持该接 口，Atlas 推理系列 加速模块 产品不支持该接 口。
	aclfvDel	x	x	仅Atlas 推 理系列加 速卡产品 支持该接 口，Atlas 推理系列 加速模块 产品不支持该接 口。
	aclfvModify	x	x	仅Atlas 推 理系列加 速卡产品 支持该接 口，Atlas 推理系列 加速模块 产品不支持该接 口。
	aclfvSearch	x	x	仅Atlas 推 理系列加 速卡产品 支持该接 口，Atlas 推理系列 加速模块 产品不支持该接 口。

1.1.3 V1.2.0 及以上版本

在V1.1.0基础上，新增了如下接口。

各版本对AscendCL接口的支持情况如下所示，各标识的含义如下：

- √: 支持。
- x: 不支持。
- 废弃: 后续版本待废弃，具体废弃接口说明请参见[1.2 废弃接口/返回码列表](#)。

目录	接口	Atlas 200/300/500 推理产品	Atlas 训练系列产品	Atlas 推理系列产品
单算子执行>单算子模型执行	aclopCompileAndExecuteV2	√	√	√
媒体数据处理V2>VDEC功能/JPEGD功能	hi_mpi_vdec_set_jpegd_precision_mode	x	x	√
媒体数据处理V2>VPC功能	hi_mpi_vpc_convert_color_v2	x	x	√
数据类型及其操作接口>acltdtDataset	acltdtGetDatasetName	x	√	x
模型推理>模型加载	aclmdlLoadFromFileWithQ	√	√	√
模型推理>模型加载	aclmdlLoadFromMemWithQ	√	√	√

1.1.4 V1.3.0 及以上版本

在V1.2.0基础上，新增了如下接口。

各版本对AscendCL接口的支持情况如下所示，各标识的含义如下：

- √: 支持。
- x: 不支持。
- 废弃: 后续版本待废弃，具体废弃接口说明请参见[1.2 废弃接口/返回码列表](#)。

目录	接口	Atlas 200/300/500 推理产品	Atlas 训练系列产品	Atlas 推理系列产品
媒体数据处理V2>VPC功能	hi_mpi_vpc_set_roundview_stitching_param	x	x	x
	hi_mpi_vpc_get_roundview_stitching_param	x	x	x

目录	接口	Atlas 200/300/500 推理产品	Atlas 训练系列产品	Atlas 推理系列产品
	hi_mpi_vpc_roundview_stitching	x	x	x
	hi_vpc_crop_resize_resize_paste	x	x	x
	hi_mpi_vpc_set_chn_workspace	x	x	x
	hi_mpi_vpc_get_lut_mem_size	x	x	x
	hi_mpi_vpc_get_affine_lut	x	x	x
	hi_mpi_vpc_get_perspective_lut	x	x	x
	hi_mpi_vpc_get_remap_lut	x	x	x
	hi_mpi_vpc_lut_remap	x	x	x
	hi_mpi_vpc_median_blur	x	x	x
	hi_mpi_vpc_erode	x	x	x
	hi_mpi_vpc_dilate	x	x	x
	hi_mpi_vpc_blur	x	x	x
	hi_mpi_vpc_gaussian_blur	x	x	x
	hi_mpi_vpc_filter2d	x	x	x
	hi_mpi_vpc_rotate	x	x	x
	hi_mpi_vpc_draw_mosaic	x	x	x
	hi_mpi_vpc_draw_cover	x	x	x
	hi_mpi_vpc_draw_line	x	x	x
	hi_mpi_vpc_draw_osd	x	x	x
媒体数据处理 V2>VENC功能/JPEGE功能	hi_mpi_venc_send_jpeg_frame	x	x	√
	hi_mpi_venc_get_jpeg_predicted_size	x	x	√

目录	接口	Atlas 200/300/500 推理产品	Atlas 训练系列产品	Atlas 推理系列产品
数据传输>共享Buffer管理	acltdtSetBufUserData	√	x	√
	acltdtGetBufUserData	√	x	√
	acltdtSetBufDataLen	√	x	√
	acltdtGetBufDataLen	√	x	√
	acltdtCopyBufRef	√	x	√
	acltdtAppendBufChain	√	x	√
	acltdtGetBufChainNum	√	x	√
	acltdtGetBufFromChain	√	x	√

1.1.5 V1.4.0 及以上版本

在V1.3.0基础上，新增了如下接口。

各版本对AscendCL接口的支持情况如下所示，各标识的含义如下：

- √：支持。
- x：不支持。
- 废弃：后续版本待废弃，具体废弃接口说明请参见[1.2 废弃接口/返回码列表](#)。

目录	接口	Atlas 200/300/500 推理产品	Atlas 训练系列产品	Atlas 推理系列产品
Profiling数据采集 >Profiling AscendCL API（通过Profiling AscendCL API采集并落盘性能数据）	aclprofSetConfig	√	√	√
媒体数据处理V2>VPCC功能	hi_mpi_vpc_get_affine_transform	x	x	x
	hi_mpi_vpc_get_rotation_matrix	x	x	x

目录	接口	Atlas 200/300/500 推理产品	Atlas 训练系列产品	Atlas 推理系列产品
	hi_mpi_vpc_warp_affine	x	x	x
	hi_mpi_vpc_flip	x	x	x
媒体数据处理V2>VENC功能/JPEGE功能	hi_mpi_venc_set_h264_vui	x	x	√
	hi_mpi_venc_get_h264_vui	x	x	√
	hi_mpi_venc_set_h265_vui	x	x	√
	hi_mpi_venc_get_h265_vui	x	x	√
	hi_mpi_venc_get_scene_mode	x	x	x
	hi_mpi_venc_set_roi_attr	x	x	x
	hi_mpi_venc_get_roi_attr	x	x	x
	hi_mpi_venc_set_ref_param	x	x	x
	hi_mpi_venc_get_ref_param	x	x	x
系统配置	aclrtSetDeviceSatMode	x	x	x
	aclrtGetDeviceSatMode	x	x	x
运行时管理>Stream管理	aclrtSetStreamOverflowSwitch	x	x	x
	aclrtGetStreamOverflowSwitch	x	x	x
	aclrtSynchronizeStreamWithTimeout	√	√	√

1.1.6 V1.5.0 及以上版本

1.1.6.1 接口列表 (Atlas 200/300/500 推理产品&Atlas 训练系列产品&Atlas 推理系列产品)

针对以下产品，在V1.4.0基础上，新增了接口：

- Atlas 200/300/500 推理产品
- Atlas 训练系列产品
- Atlas 推理系列产品

各版本对AscendCL接口的支持情况如下所示，各标识的含义如下：

- √: 支持。

- x: 不支持。
- 废弃: 后续版本待废弃, 具体废弃接口说明请参见[1.2 废弃接口/返回码列表](#)。

目录	接口	Atlas 200/300/500 推理产品	Atlas 训练系列产品	Atlas 推理系列产品
媒体数据处理V2>VENC功能/JPEGE功能	hi_mpi_venc_set_cu_pred	x	x	√
	hi_mpi_venc_get_cu_pred	x	x	√
	hi_mpi_venc_set_intra_refresh	x	x	√
	hi_mpi_venc_get_intra_refresh	x	x	√
媒体数据处理V2>公共接口	hi_mpi_sys_bind	x	x	x
	hi_mpi_sys_unbind	x	x	x
媒体数据处理V2>ISP系统控制及3A算法注册	该章节下的所有接口	x	x	x
媒体数据处理V2>MIPI Rx ioctl命令字	该章节下的所有接口	x	x	x
媒体数据处理V2>VI功能	该章节下的所有接口	x	x	x
媒体数据处理V2>VPSS功能	该章节下的所有接口	x	x	x
运行时管理>Stream管理	aclrtCreateStreamWithConfig	√	√	√
	aclrtDestroyStreamForce	√	√	√

1.1.6.2 接口列表 (Atlas 200/500 A2 推理产品)

- 针对Atlas 200/500 A2推理产品, 在V1.5.0版本第一次发布。
- Atlas 200/500 A2推理产品对AscendCL接口的支持情况如下所示, 各标识的含义如下:
- √: 支持。
 - x: 不支持。
 - 废弃: 后续版本待废弃, 具体废弃接口说明请参见[1.2 废弃接口/返回码列表](#)。

目录	接口	Atlas 200/500 A2推理产品
系统配置	aclInit	√
	aclFinalize	√
	aclrtGetVersion	√
	aclSetCompileopt	√
	aclGetRecentErrMsg	√
	aclAppLog	√
	aclDataTypeSize	√
	aclFloat16ToFloat	√
	aclFloatToFloat16	√
	aclGenGraphAndDumpForOp	√
运行时管理>Device管理	aclrtSetDevice	√
	aclrtResetDevice	√
	aclrtGetDevice	√
	aclrtGetRunMode	√
	aclrtSetTsDevice	√
	aclrtGetDeviceCount	√
	aclrtSynchronizeDevice	√
	aclrtGetSocName	√
	aclrtSetDeviceSatMode	x
	aclrtGetDeviceSatMode	x
运行时管理>Context管理	aclrtCreateContext	√
	aclrtDestroyContext	√
	aclrtSetCurrentContext	√
	aclrtGetCurrentContext	√
运行时管理>Stream管理	aclrtCreateStream	√
	aclrtCreateStreamWithConfig	√
	aclrtDestroyStream	√
	aclrtDestroyStreamForce	√

目录	接口	Atlas 200/500 A2推理产品
	aclrtSetStreamOverflowSwitch	x
	aclrtGetStreamOverflowSwitch	x
	aclrtSetStreamFailureMode	√
	aclrtSynchronizeStream	√
运行时管理>Event管理	aclrtCreateEvent	√
	aclrtCreateEventWithFlag	√
	aclrtDestroyEvent	√
	aclrtRecordEvent	√
	aclrtResetEvent	√
	aclrtQueryEvent	废弃
	aclrtQueryEventStatus	√
	aclrtQueryEventWaitStatus	√
	aclrtSynchronizeEvent	√
	aclrtSynchronizeEventWithTimeout	√
	aclrtEventElapsedTime	√
	aclrtStreamWaitEvent	√
	aclrtSetOpWaitTimeout	√
运行时管理>内存管理	aclrtMalloc	√
	aclrtMallocCached	√
	aclrtMemFlush	√
	aclrtMemInvalidate	√
	aclrtFree	√
	aclrtMallocHost	√
	aclrtFreeHost	√
	aclrtMemset	√
	aclrtMemsetAsync	√

目录	接口	Atlas 200/500 A2推理产品
	aclrtMemcpy	√
	aclrtMemcpyAsync	√
	aclrtGetMemInfo	√
	aclrtDeviceCanAccessPeer	x
	aclrtDeviceEnablePeerAccess	x
	aclrtDeviceDisablePeerAccess	x
	aclrtMemcpy2d	√ RC形态不支持调用本接口。
	aclrtMemcpy2dAsync	√ RC形态不支持调用本接口。
运行时管理>执行控制	aclrtSubscribeReport	√
	aclrtLaunchCallback	√
	aclrtProcessReport	√
	aclrtUnSubscribeReport	√
运行时管理>异常处理	aclrtSetExceptionInfoCallback	√
	aclrtGetTaskIdFromExceptionInfo	√
	aclrtGetStreamIdFromExceptionInfo	√
	aclrtGetThreadIdFromExceptionInfo	√
	aclrtGetDeviceIdFromExceptionInfo	√
	aclrtGetErrorCodeFromExceptionInfo	√
运行时管理>算力Group查询与设置	aclrtSetGroup	x
	aclrtGetGroupCount	x
	aclrtCreateGroupInfo	x
	aclrtDestroyGroupInfo	x
	aclrtGetAllGroupInfo	x

目录	接口	Atlas 200/500 A2推理产品
	aclrtGetGroupInfoDetail	x
模型推理>模型加载	aclmdlLoadFromFile	√
	aclmdlLoadFromMem	√
	aclmdlLoadFromFileWithMem	√
	aclmdlLoadFromMemWithMem	√
	aclmdlLoadFromFileWithQ	√
	aclmdlLoadFromMemWithQ	√
	aclmdlSetConfigOpt	√
	aclmdlLoadWithConfig	√
模型推理>模型执行	aclmdlExecute	√
	aclmdlExecuteAsync	√
	aclmdlUnload	√
	aclmdlQuerySize	√
	aclmdlQuerySizeFromMem	√
	aclmdlSetDynamicBatchSize	√
	aclmdlSetDynamicHWSi ze	√
	aclmdlSetInputAIPP	√
	aclmdlGetFirstAippInfo	√
	aclmdlGetAippType	√
	aclmdlSetAIPPByInputIn dex	√
	aclmdlSetInputDynamic Dims	√
	aclmdlCreateAndGetOp Desc	√
	aclmdlInitDump	√
	aclmdlSetDump	√

目录	接口	Atlas 200/500 A2推理产品
	aclmdlFinalizeDump	√
单算子执行>单算子模型执行	aclopSetModelDir	√
	aclopLoad	√
	aclopSetCompileFlag	废弃
	aclopCompile	√
	aclopExecute	废弃
	aclopExecuteV2	√
	aclopCompileAndExecute	√
	aclopCompileAndExecuteV2	√
	aclopExecWithHandle	√
	aclopInferShape	√
	aclrtSetOpExecuteTimeOut	√
	aclopRegisterCompileFunc	√
	aclopUnregisterCompileFunc	√
	aclopCreateKernel	√
	aclopSetKernelArgs	√
	aclopSetKernelWorkspaceSizes	√
	aclopUpdateParams	√
单算子执行>CBLAS接口	aclblasGemvEx	√
	aclblasCreateHandleForGemvEx	√
	aclblasHgemv	√
	aclblasCreateHandleForHgemv	√
	aclblasS8gemv	√
	aclblasCreateHandleForS8gemv	√
	aclblasGemmEx	√

目录	接口	Atlas 200/500 A2推理产品
	aclblasCreateHandleForGemmEx	√
	aclblasHgemm	√
	aclblasCreateHandleForHgemm	√
	aclblasS8gemm	√
	aclblasCreateHandleForS8gemm	√
	aclopCast	√
	aclopCreateHandleForCast	√
媒体数据处理V1>内存申请与释放	acldvppMalloc	√
	acldvppFree	√
媒体数据处理V1>通道创建与释放	acldvppCreateChannel	√
	acldvppDestroyChannel	√
	acldvdecCreateChannel	√
	acldvdecDestroyChannel	√
	acldvencCreateChannel	√
	acldvencDestroyChannel	√
媒体数据处理V1>VPC功能	acldvppVpcResizeAsync	√
	acldvppVpcCropAsync	√
	acldvppVpcCropResizeAsync	√
	acldvppVpcBatchCropAsync	√
	acldvppVpcBatchCropResizeAsync	√
	acldvppVpcCropAndPasteAsync	√
	acldvppVpcCropResizePasteAsync	√
	acldvppVpcBatchCropAndPasteAsync	√

目录	接口	Atlas 200/500 A2推理产品
	acldvppVpcBatchCropResizePasteAsync	√
	acldvppVpcBatchCropResizeMakeBorderAsync	√
	acldvppVpcConvertColorAsync	√
	acldvppVpcPyrDownAsync	x
	acldvppVpcEqualizeHistAsync	√
	acldvppVpcMakeBorderAsync	√
	acldvppVpcCalcHistAsync	√
媒体数据处理 V1>JPEGD功能	acldvppJpegDecodeAsync	√
	acldvppJpegGetImageInfo	√
	acldvppJpegGetImageInfoV2	√
	acldvppJpegPredictDecSize	√
媒体数据处理 V1>JPEGE功能	acldvppJpegEncodeAsync	√
	acldvppJpegPredictEncSize	√
媒体数据处理 V1>PNGD功能	acldvppPngDecodeAsync	√
	acldvppPngGetImageInfo	√
	acldvppPngPredictDecSize	√
媒体数据处理 V1>VDEC功能	aclvdecSendFrame	√
	aclvdecSendSkippedFrame	√
	aclvdecCallback	√
媒体数据处理 V1>VENC功能	aclvencSendFrame	√
	aclvencCallback	√

目录	接口	Atlas 200/500 A2推理产品
媒体数据处理V2>公共接口	hi_mpi_sys_init	√
	hi_mpi_sys_exit	√
	hi_mpi_dvpp_malloc	√
	hi_mpi_dvpp_free	√
	hi_mpi_dvpp_get_image_info	√
	hi_mpi_dvpp_get_version	√
	hi_mpi_sys_create_epoll	√
	hi_mpi_sys_ctl_epoll	√
	hi_mpi_sys_wait_epoll	√
	hi_mpi_sys_close_epoll	√
	hi_mpi_sys_set_chn_csc_matrix	√
	hi_mpi_sys_get_chn_csc_matrix	√
	hi_mpi_sys_get_image_align_info	√
	hi_mpi_sys_bind	√
	hi_mpi_sys_unbind	√
媒体数据处理V2>ISP系统控制及3A算法注册	hi_mpi_isp_sensor_reg_callback	√
	hi_mpi_isp_sensor_unreg_callback	√
	hi_mpi_isp_mem_init	√
	hi_mpi_isp_set_pub_attr	√
	hi_mpi_isp_get_pub_attr	√
	hi_mpi_isp_init	√
	hi_mpi_isp_run	√
	hi_mpi_isp_exit	√
	hi_mpi_isp_set_sns_slave_attr	√
	hi_mpi_isp_get_sns_slave_attr	√

目录	接口	Atlas 200/500 A2推理产品
	hi_mpi_ae_register	√
	hi_mpi_ae_unregister	√
	hi_mpi_ae_sensor_reg_callback	√
	hi_mpi_ae_sensor_unreg_callback	√
	hi_mpi_awb_register	√
	hi_mpi_awb_unregister	√
	hi_mpi_awb_sensor_reg_callback	√
	hi_mpi_awb_sensor_unregister_callback	√
媒体数据处理V2>MIPI Rx ioctl命令字	HI_MIPI_SET_DEV_ATTR	√
	HI_MIPI_SET_HS_MODE	√
	HI_MIPI_ENABLE_MIPI_CLOCK	√
	HI_MIPI_DISABLE_MIPI_CLOCK	√
	HI_MIPI_RESET_MIPI	√
	HI_MIPI_UNRESET_MIPI	√
	HI_MIPI_ENABLE_SLVS_CLOCK	√
	HI_MIPI_DISABLE_SLVS_CLOCK	√
	HI_MIPI_RESET_SLVS	√
	HI_MIPI_UNRESET_SLVS	√
	HI_MIPI_CONFIG_SENSOR_CLOCK	√
	HI_MIPI_ENABLE_SENSOR_CLOCK	√
	HI_MIPI_DISABLE_SENSOR_CLOCK	√
	HI_MIPI_RESET_SENSOR	√
	HI_MIPI_UNRESET_SENSOR	√

目录	接口	Atlas 200/500 A2推理产品
媒体数据处理V2>VI视频输入功能	hi_mpi_vi_set_dev_bind_pipe	√
	hi_mpi_vi_get_dev_bind_pipe	√
	hi_mpi_vi_set_dev_attr	√
	hi_mpi_vi_get_dev_attr	√
	hi_mpi_vi_enable_dev	√
	hi_mpi_vi_disable_dev	√
	hi_mpi_vi_create_pipe	√
	hi_mpi_vi_destroy_pipe	√
	hi_mpi_vi_start_pipe	√
	hi_mpi_vi_stop_pipe	√
	hi_mpi_vi_set_pipe_pre_crop	√
	hi_mpi_vi_get_pipe_pre_crop	√
	hi_mpi_vi_set_pipe_frame_dump_attr	√
	hi_mpi_vi_get_pipe_frame_dump_attr	√
	hi_mpi_vi_get_pipe_frame	√
	hi_mpi_vi_release_pipe_frame	√
	hi_mpi_vi_set_pipe_frame_source	√
	hi_mpi_vi_get_pipe_frame_source	√
	hi_mpi_vi_send_pipe_yuv	√
	hi_mpi_vi_send_pipe_raw	√
	hi_mpi_vi_pipe_get_buffer	√
	hi_mpi_vi_pipe_release_buffer	√

目录	接口	Atlas 200/500 A2推理产品
	hi_mpi_vi_get_pipe_fd	√
	hi_mpi_vi_set_chn_attr	√
	hi_mpi_vi_get_chn_attr	√
	hi_mpi_vi_enable_chn	√
	hi_mpi_vi_disable_chn	√
	hi_mpi_vi_set_chn_dis_c onfig	√
	hi_mpi_vi_get_chn_dis_c onfig	√
	hi_mpi_vi_set_chn_dis_at tr	√
	hi_mpi_vi_get_chn_dis_a ttr	√
	hi_mpi_vi_set_chn_dis_p aram	√
	hi_mpi_vi_get_chn_dis_p aram	√
	hi_mpi_vi_set_chn_ldc_at tr	√
	hi_mpi_vi_get_chn_ldc_a ttr	√
	hi_mpi_vi_get_chn_fram e	√
	hi_mpi_vi_release_chn_fr ame	√
	hi_mpi_vi_get_chn_fd	√
媒体数据处理 V2>VPSS视频处理功 能	hi_mpi_vpss_create_grp	√
	hi_mpi_vpss_destroy_grp	√
	hi_mpi_vpss_start_grp	√
	hi_mpi_vpss_stop_grp	√
	hi_mpi_vpss_set_grp_cro p	√
	hi_mpi_vpss_get_grp_cro p	√

目录	接口	Atlas 200/500 A2推理产品
	hi_mpi_vpss_set_grp_param	√
	hi_mpi_vpss_get_grp_param	√
	hi_mpi_vpss_set_grp_fisheye_cfg	√
	hi_mpi_vpss_get_grp_fisheye_cfg	√
	hi_mpi_vpss_set_chn_attr	√
	hi_mpi_vpss_get_chn_attr	√
	hi_mpi_vpss_enable_chn	√
	hi_mpi_vpss_disable_chn	√
	hi_mpi_vpss_get_chn_frame	√
	hi_mpi_vpss_release_chn_frame	√
	hi_mpi_vpss_set_chn_fisheye	√
	hi_mpi_vpss_get_chn_fisheye	√
	hi_mpi_vpss_get_chn_fd	√
媒体数据处理V2>VPC功能	hi_mpi_vpc_create_chn	√
	hi_mpi_vpc_destroy_chn	√
	hi_mpi_vpc_resize	√
	hi_mpi_vpc_crop	√
	hi_mpi_vpc_crop_resize	√
	hi_mpi_vpc_crop_resize_paste	√
	hi_mpi_vpc_convert_color	√
	hi_mpi_vpc_convert_color_to_yuv420	√
	hi_mpi_vpc_copy_make_border	√

目录	接口	Atlas 200/500 A2推理产品
	hi_mpi_vpc_pyrdown	√
	hi_mpi_vpc_calc_hist	√
	hi_mpi_vpc_equalize_hist	√
	hi_mpi_vpc_crop_resize_make_border	√
	hi_mpi_vpc_batch_crop_resize_paste	√
	hi_mpi_vpc_batch_crop_resize_make_border	√
	hi_mpi_vpc_get_process_result	√
	hi_mpi_vpc_sys_create_chn	√
	hi_mpi_vpc_set_roundview_stitching_param	x
	hi_mpi_vpc_get_roundview_stitching_param	x
	hi_mpi_vpc_roundview_stitching	x
	hi_vpc_crop_resize_resize_paste	x
	hi_mpi_vpc_set_chn_workspace	√
	hi_mpi_vpc_get_lut_memory_size	√
	hi_mpi_vpc_get_affine_lut	√
	hi_mpi_vpc_get_perspective_lut	√
	hi_mpi_vpc_get_remap_lut	√
	hi_mpi_vpc_lut_remap	√
	hi_mpi_vpc_median_blur	√
	hi_mpi_vpc_erode	√
	hi_mpi_vpc_dilate	√

目录	接口	Atlas 200/500 A2推理产品
	hi_mpi_vpc_blur	√
	hi_mpi_vpc_gaussian_blur	√
	hi_mpi_vpc_filter2d	√
	hi_mpi_vpc_rotate	√
	hi_mpi_vpc_draw_mosaic	√
	hi_mpi_vpc_draw_cover	√
	hi_mpi_vpc_draw_line	√
	hi_mpi_vpc_draw_osd	√
	hi_mpi_vpc_get_affine_transform	√
	hi_mpi_vpc_get_rotation_matrix	√
	hi_mpi_vpc_warp_affine	√
	hi_mpi_vpc_flip	√
媒体数据处理 V2>VDEC功能/JPEGD功能	hi_mpi_vdec_create_chn	√
	hi_mpi_vdec_destroy_chn	√
	hi_mpi_vdec_get_chn_attr	√
	hi_mpi_vdec_set_chn_attr	√
	hi_mpi_vdec_start_recv_stream	√
	hi_mpi_vdec_stop_recv_stream	√
	hi_mpi_vdec_query_status	√
	hi_mpi_vdec_reset_chn	√
	hi_mpi_vdec_set_chn_param	√
	hi_mpi_vdec_get_chn_param	√

目录	接口	Atlas 200/500 A2推理产品
	hi_mpi_vdec_set_protocol_param	x
	hi_mpi_vdec_get_protocol_param	√
	hi_mpi_vdec_send_stream	√
	hi_mpi_vdec_get_frame	√
	hi_mpi_vdec_release_frame	√
	hi_mpi_vdec_get_fd	√
	hi_mpi_vdec_close_fd	x
	hi_vdec_get_pic_buf_size	√
	hi_vdec_get_tmvp_buf_size	√
	hi_mpi_vdec_set_jpegd_precision_mode	√
	hi_mpi_vdec_get_jpegd_output_info	√
媒体数据处理 V2>VENC功能/JPEGE功能	hi_mpi_venc_create_chn	√
	hi_mpi_venc_destroy_chn	√
	hi_mpi_venc_start_chn	√
	hi_mpi_venc_stop_chn	√
	hi_mpi_venc_query_status	√
	hi_mpi_venc_get_stream	√
	hi_mpi_venc_release_stream	√
	hi_mpi_venc_send_frame	√
	hi_mpi_venc_set_mod_param	√
	hi_mpi_venc_get_mod_param	√
	hi_mpi_venc_request_idr	√
	hi_mpi_venc_get_fd	√

目录	接口	Atlas 200/500 A2推理产品
	hi_mpi_venc_close_fd	x
	hi_mpi_venc_set_jpeg_param	√
	hi_mpi_venc_get_jpeg_param	√
	hi_mpi_venc_set_chn_param	√
	hi_mpi_venc_set_scene_mode	√
	hi_mpi_venc_get_scene_mode	√
	hi_mpi_venc_set_rc_param	√
	hi_mpi_venc_get_rc_param	√
	hi_mpi_venc_set_jpeg_huffman_param	√
	hi_mpi_venc_get_jpeg_huffman_param	√
	hi_mpi_venc_compact_jpeg_tables	√
	hi_mpi_venc_send_jpege_frame	√
	hi_mpi_venc_get_jpege_predicted_size	√
	hi_mpi_venc_set_roi_attr	√
	hi_mpi_venc_get_roi_attr	√
	hi_mpi_venc_set_ref_param	√
	hi_mpi_venc_get_ref_param	√
	hi_mpi_venc_set_h264_vui	√
	hi_mpi_venc_get_h264_vui	√
	hi_mpi_venc_set_h265_vui	√

目录	接口	Atlas 200/500 A2推理产品
	hi_mpi_venc_get_h265_vui	√
	hi_mpi_venc_set_cu_pred	√
	hi_mpi_venc_get_cu_pred	√
	hi_mpi_venc_set_intra_refresh	√
	hi_mpi_venc_get_intra_refresh	√
媒体数据处理 V2>PNGD功能	hi_mpi_pngd_create_chn	√
	hi_mpi_pngd_destroy_chn	√
	hi_mpi_pngd_send_stream	√
	hi_mpi_pngd_get_image_data	√
	hi_mpi_png_get_image_info	√
Profiling AscendCL API for Extension (Profiling AscendCL API扩展接口)	aclprofInit	√
	aclprofSetConfig	√
	aclprofStart	√
	aclprofStop	√
	aclprofFinalize	√
Profiling AscendCL API for Extension (Profiling AscendCL API扩展接口)	aclprofCreateStamp	√
	aclprofSetStampTraceMessage	√
	aclprofMark	√
	aclprofPush	√
	aclprofPop	√
	aclprofRangeStart	√
	aclprofRangeStop	√
	aclprofDestroyStamp	√

目录	接口	Atlas 200/500 A2推理产品
Profiling AscendCL API for Subscription (订阅算子信息的 Profiling AscendCL API)	aclprofModelSubscribe	√
	aclprofModelUnSubscribe	√
	aclprofGetOpDescSize	√
	aclprofGetOpNum	√
	aclprofGetOpTypeLen	√
	aclprofGetOpType	√
	aclprofGetOpNameLen	√
	aclprofGetOpName	√
	aclprofGetOpStart	√
	aclprofGetOpEnd	√
	aclprofGetOpDuration	√
	aclprofGetModelId	√
Profiling AscendCL API (PyTorch场景标记迭代时间)	aclprofGetStepTimestamp	√
数据传输>Tensor数据传输	acltdtCreateChannel	x
	acltdtSendTensor	x
	acltdtReceiveTensor	x
	acltdtStopChannel	x
	acltdtDestroyChannel	x
	acltdtCreateChannelWithCapacity	x
	acltdtQueryChannelSize	x
数据传输>共享队列管理	acltdtCreateQueue	√
	acltdtDestroyQueue	√
	acltdtEnqueueData	√
	acltdtDequeueData	√
	acltdtEnqueue	x
	acltdtDequeue	x
	acltdtBindQueueRoutes	√

目录	接口	Atlas 200/500 A2推理产品
	acldtUnbindQueueRoutes	√
	acldtQueryQueueRoutes	√
	acldtGrantQueue	x
	acldtAttachQueue	x
数据传输>共享Buffer管理	acldtAllocBuf	√
	acldtFreeBuf	√
	acldtGetBufData	√
	acldtSetBufUserData	√
	acldtGetBufUserData	√
	acldtSetBufDataLen	√
	acldtGetBufDataLen	√
	acldtCopyBufRef	√
	acldtAppendBufChain	√
	acldtGetBufChainNum	√
	acldtGetBufFromChain	√

1.1.7 V1.6.0 及以上版本

1.1.7.1 接口列表 (Atlas 200/300/500 推理产品&Atlas 训练系列产品&Atlas 推理系列产品&Atlas 200/500 A2 推理产品)

针对以下产品，在V1.5.0基础上，新增了接口：

- Atlas 200/300/500 推理产品
- Atlas 训练系列产品
- Atlas 推理系列产品
- Atlas 200/500 A2推理产品

各版本对AscendCL接口的支持情况如下所示，各标识的含义如下：

- √：支持。
- x：不支持。
- 废弃：后续版本待废弃，具体废弃接口说明请参见[1.2 废弃接口/返回码列表](#)。

目录	接口	Atlas 200/300/500 推理产品	Atlas 训练系列产品	Atlas 推理系列产品	Atlas 200/500 A2推理产品
媒体数据处理 V2>VDEC 视频解码功能/ JPEG图片解码功能	hi_mpi_vdec_set_display_mode	x	x	x	√
	hi_mpi_vdec_get_display_mode	x	x	x	√
媒体数据处理 V2>AI音频输入/AO音频输出	hi_mpi_ai_set_publisher	x	x	x	√
	hi_mpi_ai_enable	x	x	x	√
	hi_mpi_ai_disable	x	x	x	√
	hi_mpi_ai_set_channel_attr	x	x	x	√
	hi_mpi_ai_enable_channel	x	x	x	√
	hi_mpi_ai_disable_channel	x	x	x	√
	hi_mpi_ai_get_frame	x	x	x	√
	hi_mpi_ai_release_frame	x	x	x	√
	hi_mpi_ai_enable_resample	x	x	x	√
	hi_mpi_ai_disable_resample	x	x	x	√
	hi_mpi_ao_set_publisher	x	x	x	√
	hi_mpi_ao_enable	x	x	x	√
	hi_mpi_ao_disable	x	x	x	√
	hi_mpi_ao_enable_channel	x	x	x	√
	hi_mpi_ao_disable_channel	x	x	x	√
	hi_mpi_ao_send_frame	x	x	x	√
	hi_mpi_ao_get_channel_delay	x	x	x	√

目录	接口	Atlas 200/300/500 推理产品	Atlas 训练系列产品	Atlas 推理系列产品	Atlas 200/500 A2推理产品
	hi_mpi_ao_enable_resample	x	x	x	√
	hi_mpi_ao_disable_resample	x	x	x	√
媒体数据处理V2>音量调节	HI_ACODEC_SET_DACL_VOLUME	x	x	x	√
	HI_ACODEC_SET_DACR_VOLUME	x	x	x	√
	HI_ACODEC_GET_DACL_VOLUME	x	x	x	√
	HI_ACODEC_GET_DACR_VOLUME	x	x	x	√
	HI_ACODEC_SET_ADCL_VOLUME	x	x	x	√
	HI_ACODEC_SET_ADCR_VOLUME	x	x	x	√
	HI_ACODEC_GET_ADCL	x	x	x	√
	HI_ACODEC_GET_ADCR	x	x	x	√
媒体数据处理V2>AENC音频编码/ADEC音频解码	hi_mpi_aenc_create_chn	x	x	x	√
	hi_mpi_aenc_destroy_chn	x	x	x	√
	hi_mpi_aenc_get_stream	x	x	x	√
	hi_mpi_aenc_release_stream	x	x	x	√
	hi_mpi_adec_create_chn	x	x	x	√
	hi_mpi_adec_destroy_chn	x	x	x	√
	hi_mpi_adec_send_stream	x	x	x	√

目录	接口	Atlas 200/300 /500 推 理产品	Atlas 训 练系列产 品	Atlas 推 理系列产 品	Atlas 200/50 0 A2推 理产品
媒体数据 处理 V2>Regio n区域管 理	hi_mpi_rgn_create	x	x	x	√
	hi_mpi_rgn_destroy	x	x	x	√
	hi_mpi_rgn_attach_to_chn	x	x	x	√
	hi_mpi_rgn_detach_from_chn	x	x	x	√
	hi_mpi_rgn_set_display_attr	x	x	x	√
	hi_mpi_rgn_get_display_attr	x	x	x	√
	hi_mpi_rgn_get_canvas_info	x	x	x	√
	hi_mpi_rgn_update_canvas	x	x	x	√
媒体数据 处理 V2>VO视 频输出功 能	hi_mpi_vo_set_pub_attr	x	x	x	√
	hi_mpi_vo_enable	x	x	x	√
	hi_mpi_vo_disable	x	x	x	√
	hi_mpi_vo_set_video_layer_attr	x	x	x	√
	hi_mpi_vo_get_video_layer_attr	x	x	x	√
	hi_mpi_vo_enable_video_layer	x	x	x	√
	hi_mpi_vo_disable_video_layer	x	x	x	√
	hi_mpi_vo_set_chn_attr	x	x	x	√
	hi_mpi_vo_enable_chn	x	x	x	√
	hi_mpi_vo_disable_chn	x	x	x	√
	hi_mpi_vo_set_chn_param	x	x	x	√

目录	接口	Atlas 200/300/500 推理产品	Atlas 训练系列产品	Atlas 推理系列产品	Atlas 200/500 A2推理产品
	hi_mpi_vo_set_chn_frame_rate	x	x	x	√
	hi_mpi_vo_pause_chn	x	x	x	√
	hi_mpi_vo_resume_chn	x	x	x	√
	hi_mpi_vo_hide_chn	x	x	x	√
	hi_mpi_vo_show_chn	x	x	x	√
	hi_mpi_vo_send_frame	x	x	x	√
	hi_mpi_vo_create_pool	x	x	x	√
	hi_mpi_vo_handle_to_phys_addr	x	x	x	√
	hi_mpi_vo_destroy_pool	x	x	x	√
	hi_mpi_vo_bind_layer	x	x	x	√
	hi_mpi_vo_unbind_layer	x	x	x	√
媒体数据处理 V2>TDE 图形绘制功能	hi_tde_open	x	x	x	√
	hi_tde_close	x	x	x	√
	hi_tde_begin_job	x	x	x	√
	hi_tde_end_job	x	x	x	√
	hi_tde_cancel_job	x	x	x	√
	hi_tde_wait_for_done	x	x	x	√
	hi_tde_wait_all_done	x	x	x	√
	hi_tde_quick_copy	x	x	x	√
	hi_tde_quick_fill	x	x	x	√
	hi_tde_pattern_fill	x	x	x	√

目录	接口	Atlas 200/300/500 推理产品	Atlas 训练系列产品	Atlas 推理系列产品	Atlas 200/500 A2推理产品
媒体数据处理 V2>HIFB 叠加图形层管理功能	FBIOGET_VSCREENINFO	x	x	x	√
	FBIOPUT_VSCREENINFO	x	x	x	√
	FBIOGET_FSCREENINFO	x	x	x	√
	FBIOPAN_DISPLAY	x	x	x	√
	FBIOGET_CMAP	x	x	x	√
	FBIOPUT_CMAP	x	x	x	√
	FBIOGET_SCREEN_ORIGIN_HIFB	x	x	x	√
	FBIOPUT_SCREEN_ORIGIN_HIFB	x	x	x	√
	FBIOGET_SHOW_HIFB	x	x	x	√
	FBIOPUT_SHOW_HIFB	x	x	x	√
	FBIOGET_ALPHA_HIFB	x	x	x	√
	FBIOPUT_ALPHA_HIFB	x	x	x	√
媒体数据处理 V2>HDMI I/外设	hi_mpi_hdmi_init	x	x	x	√
	hi_mpi_hdmi_deinit	x	x	x	√
	hi_mpi_hdmi_open	x	x	x	√
	hi_mpi_hdmi_close	x	x	x	√
	hi_mpi_hdmi_set_attr	x	x	x	√
	hi_mpi_hdmi_get_attr	x	x	x	√
	hi_mpi_hdmi_start	x	x	x	√
	hi_mpi_hdmi_stop	x	x	x	√
	hi_mpi_hdmi_set_infoframe	x	x	x	√

目录	接口	Atlas 200/300/500 推理产品	Atlas 训练系列产品	Atlas 推理系列产品	Atlas 200/500 A2推理产品
媒体数据处理 V2>VENC 视频编码功能/ JPEG图片编码功能	hi_mpi_venc_set_chn_attr	x	x	√	√
	hi_mpi_venc_get_chn_attr	x	x	√	√
媒体数据处理 V2>VI 视频输入功能	hi_mpi_vi_set_pipe_attr	x	x	x	√
	hi_mpi_vi_get_pipe_attr	x	x	x	√
	hi_mpi_vi_set_pipe_vc_number	x	x	x	√
	hi_mpi_vi_get_pipe_vc_number	x	x	x	√
	hi_mpi_vi_set_chn_low_delay_attr	x	x	x	√
	hi_mpi_vi_get_chn_low_delay_attr	x	x	x	√
	hi_mpi_vi_set_vi_vpss_mode	x	x	x	√
	hi_mpi_vi_get_vi_vpss_mode	x	x	x	√
	hi_mpi_vi_set_pipe_pre_crop	x	x	x	√
	hi_mpi_vi_set_pipe_bind_strobe	x	x	x	√
单算子执行>单算子模型执行	aclrtSetOpExecuteTimeOut	x	√	√	√
系统配置	aclGetCompileopt	√	√	√	√
	aclGetCompileoptSize	√	√	√	√
运行时管理 >Stream管理	aclrtSetStreamFailureMode	x	√	√	√

目录	接口	Atlas 200/300/500 推理产品	Atlas 训练系列产品	Atlas 推理系列产品	Atlas 200/500 A2推理产品
	aclrtSetStreamConfigOpt	x	x	x	x
	aclrtCreateStreamV2	x	x	x	x
运行时管理>Event管理	aclrtSynchronizeEventWithTimeout	√	√	√	√
模型推理>模型执行	aclmdlSetExecConfigOpt	√	√	√	√
	aclmdlExecuteV2	√	√	√	√
	aclmdlExecuteAsyncV2	x	x	x	x
运行时管理>内存管理	aclrtAllocatorRegister	x	√	√	x
	aclrtAllocatorUnregister	x	√	√	x
	aclrtMallocAlign32	√	√	√	√
运行时管理>Context管理	aclrtCtxSetSysParamOpt	√	√	√	√
	aclrtCtxGetSysParamOpt	√	√	√	√
	aclrtGetOverflowStatus	√	√	√	√
	aclrtResetOverflowStatus	√	√	√	√
运行时管理>Device管理	aclrtGetDeviceUtilizationRate	√	√	√	√
单算子执行>单算子API执行	公共接口 (AscendCL meta) 说明 定义算子时依赖的一些 AscendCL元接口，通过这些元接口可以构建所需的数据结构，如 aclTensor、aclScalar、aclIntArray等。	√	√	√	√

目录	接口	Atlas 200/300/500 推理产品	Atlas 训练系列产品	Atlas 推理系列产品	Atlas 200/500 A2推理产品
	NN类算子接口 (aclnn) 说明 aclnn提供的核心功能接口，包括张量基础计算（加/减/乘/除等）以及NN相关的如卷积、池化、激活函数等操作。	x	√	x	x

1.1.7.2 接口列表 (Atlas A2 训练系列产品/Atlas 800I A2 推理产品)

针对Atlas A2训练系列产品，在V1.6.0版本第一次发布。

Atlas A2训练系列产品对AscendCL接口的支持情况如下所示，各标识的含义如下：

- √：支持。
- x：不支持。
- 废弃：后续版本待废弃，具体废弃接口说明请参见[1.2 废弃接口/返回码列表](#)。

目录	接口	Atlas A2训练系列产品/Atlas 800I A2推理产品
系统配置	aclInit	√
	aclFinalize	√
	aclrtGetVersion	√
	aclSetCompileopt	√
	aclGetRecentErrMsg	√
	aclAppLog	√
	aclDataTypeSize	√
	aclFloat16ToFloat	√
	aclFloatToFloat16	√
	aclGenGraphAndDumpForOp	√
运行时管理>Device管理	aclrtSetDevice	√
	aclrtResetDevice	√
	aclrtGetDevice	√
	aclrtGetRunMode	√

目录	接口	Atlas A2训练系列产品/Atlas 800I A2推理产品
	aclrtSetTsDevice	√
	aclrtGetDeviceCount	√
	aclrtGetDeviceUtilizationRate	√
	aclrtSynchronizeDevice	√
	aclrtGetSocName	√
	aclrtSetDeviceSatMode	√
	aclrtGetDeviceSatMode	√
运行时管理>Context管理	aclrtCreateContext	√
	aclrtDestroyContext	√
	aclrtSetCurrentContext	√
	aclrtGetCurrentContext	√
	aclrtCtxSetSysParamOpt	√
	aclrtCtxGetSysParamOpt	√
	aclrtGetOverflowStatus	√
	aclrtResetOverflowStatus	√
运行时管理>Stream管理	aclrtCreateStream	√
	aclrtCreateStreamV2	x
	aclrtSetStreamConfigOpt	x
	aclrtCreateStreamWithConfig	√
	aclrtDestroyStream	√
	aclrtDestroyStreamForce	√
	aclrtSetStreamOverflowSwitch	√
	aclrtGetStreamOverflowSwitch	√
	aclrtSetStreamFailureMode	√
	aclrtSynchronizeStream	√
	aclrtSynchronizeStreamWithTimeout	√
运行时管理>Event管理	aclrtCreateEvent	√

目录	接口	Atlas A2训练系列产品/Atlas 800I A2推理产品
	aclrtCreateEventWithFlag	√
	aclrtDestroyEvent	√
	aclrtRecordEvent	√
	aclrtResetEvent	√
	aclrtQueryEvent	废弃
	aclrtQueryEventStatus	√
	aclrtQueryEventWaitStatus	√
	aclrtSynchronizeEvent	√
	aclrtSynchronizeEventWithTimeout	√
	aclrtEventElapsedTime	√
	aclrtStreamWaitEvent	√
	aclrtSetOpWaitTimeout	√
运行时管理>内存管理	aclrtMalloc	√
	aclrtMallocAlign32	√
	aclrtMallocCached	√
	aclrtMemFlush	√
	aclrtMemInvalidate	√
	aclrtFree	√
	aclrtMallocHost	√
	aclrtFreeHost	√
	aclrtMemset	√
	aclrtMemsetAsync	√
	aclrtMemcpy	√
	aclrtMemcpyAsync	√
	aclrtGetMemInfo	√
	aclrtDeviceCanAccessPeer	√
	aclrtDeviceEnablePeerAccess	√
	aclrtDeviceDisablePeerAccess	√

目录	接口	Atlas A2训练系列产品/Atlas 800I A2推理产品
	aclrtMemcpy2d	√
	aclrtMemcpy2dAsync	√
	aclrtAllocatorRegister	√
	aclrtAllocatorUnregister	√
运行时管理>执行控制	aclrtSubscribeReport	√
	aclrtLaunchCallback	√
	aclrtProcessReport	√
	aclrtUnSubscribeReport	√
运行时管理>异常处理	aclrtSetExceptionInfoCallback	√
	aclrtGetTaskIdFromExceptionInfo	√
	aclrtGetStreamIdFromExceptionInfo	√
	aclrtGetThreadIdFromExceptionInfo	√
	aclrtGetDeviceIdFromExceptionInfo	√
	aclrtGetErrorCodeFromExceptionInfo	√
运行时管理>算力Group查询与设置	aclrtSetGroup	x
	aclrtGetGroupCount	x
	aclrtCreateGroupInfo	x
	aclrtDestroyGroupInfo	x
	aclrtGetAllGroupInfo	x
	aclrtGetGroupInfoDetail	x
模型推理>模型加载	aclmdlLoadFromFile	√
	aclmdlLoadFromMem	√
	aclmdlLoadFromFileWithMem	√
	aclmdlLoadFromMemWithMem	√

目录	接口	Atlas A2训练系列产品/Atlas 800I A2推理产品
	aclmdlLoadFromFileWithQ	√
	aclmdlLoadFromMemWithQ	√
	aclmdlSetConfigOpt	√
	aclmdlLoadWithConfig	√
模型推理>模型执行	aclmdlExecute	√
	aclmdlExecuteAsync	√
	aclmdlSetExecConfigOpt	√
	aclmdlExecuteV2	√
	aclmdlExecuteAsyncV2	x
	aclmdlUnload	√
	aclmdlQuerySize	√
	aclmdlQuerySizeFromMem	√
	aclmdlSetDynamicBatchSize	√
	aclmdlSetDynamicHWSIZE	√
	aclmdlSetInputAIPP	x
	aclmdlGetFirstAippInfo	x
	aclmdlGetAippType	x
	aclmdlSetAIPPByInputIndex	x
	aclmdlSetInputDynamicDims	√
	aclmdlCreateAndGetOpDesc	√
	aclmdlInitDump	√
	aclmdlSetDump	√
	aclmdlFinalizeDump	√
单算子执行>单算子API执行	公共接口 (AscendCL meta) 说明 定义算子时依赖的一些AscendCL元接口，通过这些元接口可以构建所需的数据结构，如aclTensor、aclScalar、aclIntArray等。	√

目录	接口	Atlas A2训练系列产品/Atlas 800I A2推理产品
	NN类算子接口 (aclnn) 说明 aclnn提供的核心功能接口，包括张量基础计算（加/减/乘/除等）以及NN相关的如卷积、池化、激活函数等操作。	√
单算子执行>单算子模型执行	aclopSetModelDir	√
	aclopLoad	√
	aclopSetCompileFlag	废弃
	aclopCompile	√
	aclopExecute	废弃
	aclopExecuteV2	√
	aclopCompileAndExecute	√
	aclopCompileAndExecuteV2	√
	aclopExecWithHandle	√
	aclopInferShape	√
	aclrtSetOpExecuteTimeOut	√
	aclopRegisterCompileFunc	√
	aclopUnregisterCompileFunc	√
	aclopCreateKernel	√
	aclopSetKernelArgs	√
	aclopSetKernelWorkspaceSizes	√
	aclopUpdateParams	√
单算子执行>CBLAS接口	aclblasGemvEx	√
	aclblasCreateHandleForGemvEx	√
	aclblasHgemv	√
	aclblasCreateHandleForHgemv	√
	aclblasS8gemv	√
	aclblasCreateHandleForS8gemv	√

目录	接口	Atlas A2训练系列产品/Atlas 800I A2推理产品
	aclblasGemmEx	√
	aclblasCreateHandleForGemmEx	√
	aclblasHgemm	√
	aclblasCreateHandleForHgemm	√
	aclblasS8gemm	√
	aclblasCreateHandleForS8gemm	√
	aclopCast	√
	aclopCreateHandleForCast	√
媒体数据处理V1>内存申请与释放	acldvppMalloc	√
	acldvppFree	√
媒体数据处理V1>通道创建与释放	acldvppCreateChannel	√
	acldvppDestroyChannel	√
	acldvdecCreateChannel	√
	acldvdecDestroyChannel	√
	acldvencCreateChannel	x
	acldvencDestroyChannel	x
媒体数据处理V1>VPC功能	acldvppVpcResizeAsync	√
	acldvppVpcCropAsync	√
	acldvppVpcCropResizeAsync	√
	acldvppVpcBatchCropAsync	√
	acldvppVpcBatchCropResizeAsync	√
	acldvppVpcCropAndPasteAsync	√
	acldvppVpcCropResizePasteAsync	√
	acldvppVpcBatchCropAndPasteAsync	√

目录	接口	Atlas A2训练系列产品/Atlas 800I A2推理产品
	acldvppVpcBatchCropResizePasteAsync	√
	acldvppVpcBatchCropResizeMakeBorderAsync	√
	acldvppVpcConvertColorAsync	√
	acldvppVpcPyrDownAsync	x
	acldvppVpcEqualizeHistAsync	√
	acldvppVpcMakeBorderAsync	√
	acldvppVpcCalcHistAsync	√
媒体数据处理V1>JPEGD功能	acldvppJpegDecodeAsync	√
	acldvppJpegGetImageInfo	√
	acldvppJpegGetImageInfoV2	√
	acldvppJpegPredictDecSize	√
媒体数据处理V1>JPEGE功能	acldvppJpegEncodeAsync	√
	acldvppJpegPredictEncSize	√
媒体数据处理V1>PNGD功能	acldvppPngDecodeAsync	√
	acldvppPngGetImageInfo	√
	acldvppPngPredictDecSize	√
媒体数据处理V1>VDEC功能	acldvdecSendFrame	√
	acldvdecSendSkippedFrame	√
	acldvdecCallback	√
媒体数据处理V1>VENC功能	acldvencSendFrame	x
	acldvencCallback	x
媒体数据处理V2>公共接口	hi_mpi_sys_init	√
	hi_mpi_sys_exit	√
	hi_mpi_dvpp_malloc	√
	hi_mpi_dvpp_free	√
	hi_mpi_dvpp_get_image_info	√
	hi_mpi_dvpp_get_version	√

目录	接口	Atlas A2训练系列产品/Atlas 800I A2推理产品
	hi_mpi_sys_create_epoll	√
	hi_mpi_sys_ctl_epoll	√
	hi_mpi_sys_wait_epoll	√
	hi_mpi_sys_close_epoll	√
	hi_mpi_sys_set_chn_csc_matrix	√
	hi_mpi_sys_get_chn_csc_matrix	√
	hi_mpi_sys_get_image_align_info	√
	hi_mpi_sys_bind	x
	hi_mpi_sys_unbind	x
媒体数据处理V2>AI音频输入/AO音频输出	该章节下的所有接口	x
媒体数据处理V2>AENC音频编码/ADEC音频解码	该章节下的所有接口	x
媒体数据处理V2>音量调节	该章节下的命令字	x
媒体数据处理V2>ISP系统控制及3A算法注册	该章节下的所有接口	x
媒体数据处理V2>MIPI Rx ioctl命令字	该章节下的命令字	x
媒体数据处理V2>VI视频输入功能	该章节下的所有接口	x
媒体数据处理V2>Region区域管理	该章节下的所有接口	x
媒体数据处理V2>VPSS视频处理功能	该章节下的所有接口	x
媒体数据处理V2>VO视频输出功能	该章节下的所有接口	x
媒体数据处理V2>TDE图形绘制功能	该章节下的所有接口	x
媒体数据处理V2>HIFB叠加图形层管理功能	该章节下的所有接口	x

目录	接口	Atlas A2训练系列产品/Atlas 800I A2推理产品
媒体数据处理V2>HDMI外设	该章节下的所有接口	x
媒体数据处理V2>VPC功能	hi_mpi_vpc_create_chn	√
	hi_mpi_vpc_destroy_chn	√
	hi_mpi_vpc_resize	√
	hi_mpi_vpc_crop	√
	hi_mpi_vpc_crop_resize	√
	hi_mpi_vpc_crop_resize_paste	√
	hi_mpi_vpc_convert_color	√
	hi_mpi_vpc_convert_color_to_yuv420	√
	hi_mpi_vpc_copy_make_border	√
	hi_mpi_vpc_pyrdown	√
	hi_mpi_vpc_calc_hist	√
	hi_mpi_vpc_equalize_hist	√
	hi_mpi_vpc_crop_resize_make_border	√
	hi_mpi_vpc_batch_crop_resize_paste	√
	hi_mpi_vpc_batch_crop_resize_make_border	√
	hi_mpi_vpc_get_process_result	√
	hi_mpi_vpc_sys_create_chn	√
	hi_mpi_vpc_set_roundview_stitching_param	x
	hi_mpi_vpc_get_roundview_stitching_param	x
	hi_mpi_vpc_roundview_stitching	x
	hi_vpc_crop_resize_resize_paste	x

目录	接口	Atlas A2训练系列产品/Atlas 800I A2推理产品
	hi_mpi_vpc_set_chn_workspa ce	√
	hi_mpi_vpc_get_lut_mem_siz e	√
	hi_mpi_vpc_get_affine_lut	√
	hi_mpi_vpc_get_perspective_l ut	√
	hi_mpi_vpc_get_remap_lut	√
	hi_mpi_vpc_lut_remap	√
	hi_mpi_vpc_median_blur	√
	hi_mpi_vpc_erode	√
	hi_mpi_vpc_dilate	√
	hi_mpi_vpc_blur	√
	hi_mpi_vpc_gaussian_blur	√
	hi_mpi_vpc_filter2d	√
	hi_mpi_vpc_rotate	√
	hi_mpi_vpc_draw_mosaic	√
	hi_mpi_vpc_draw_cover	√
	hi_mpi_vpc_draw_line	√
	hi_mpi_vpc_draw_osd	√
	hi_mpi_vpc_get_affine_transf orm	√
	hi_mpi_vpc_get_rotation_mat rix	√
	hi_mpi_vpc_warp_affine	√
	hi_mpi_vpc_flip	√
媒体数据处理V2>VDEC功 能/JPEGD功能	hi_mpi_vdec_create_chn	√
	hi_mpi_vdec_destroy_chn	√
	hi_mpi_vdec_get_chn_attr	√
	hi_mpi_vdec_set_chn_attr	√

目录	接口	Atlas A2训练系列产品/Atlas 800I A2推理产品
	hi_mpi_vdec_start_rcv_stream	√
	hi_mpi_vdec_stop_rcv_stream	√
	hi_mpi_vdec_query_status	√
	hi_mpi_vdec_reset_chn	√
	hi_mpi_vdec_set_chn_param	√
	hi_mpi_vdec_get_chn_param	√
	hi_mpi_vdec_set_protocol_param	x
	hi_mpi_vdec_get_protocol_param	√
	hi_mpi_vdec_send_stream	√
	hi_mpi_vdec_get_frame	√
	hi_mpi_vdec_release_frame	√
	hi_mpi_vdec_get_fd	√
	hi_mpi_vdec_close_fd	x
	hi_vdec_get_pic_buf_size	√
	hi_vdec_get_tmv_buf_size	√
	hi_mpi_vdec_set_jpegd_precision_mode	√
	hi_mpi_vdec_get_jpegd_output_info	√
	hi_mpi_vdec_set_display_mode	√
	hi_mpi_vdec_get_display_mode	√
媒体数据处理V2>VENC功能/JPEGE功能	hi_mpi_venc_create_chn	√
	hi_mpi_venc_destroy_chn	√
	hi_mpi_venc_start_chn	√
	hi_mpi_venc_stop_chn	√
	hi_mpi_venc_query_status	√

目录	接口	Atlas A2训练系列产品/Atlas 800I A2推理产品
	hi_mpi_venc_get_stream	√
	hi_mpi_venc_release_stream	√
	hi_mpi_venc_send_frame	√
	hi_mpi_venc_set_mod_param	√
	hi_mpi_venc_get_mod_param	√
	hi_mpi_venc_request_idr	√
	hi_mpi_venc_get_fd	√
	hi_mpi_venc_close_fd	x
	hi_mpi_venc_set_jpeg_param	√
	hi_mpi_venc_get_jpeg_param	√
	hi_mpi_venc_set_chn_param	√
	hi_mpi_venc_set_scene_mode	x
	hi_mpi_venc_get_scene_mode	√
	hi_mpi_venc_set_rc_param	√
	hi_mpi_venc_get_rc_param	√
	hi_mpi_venc_set_jpeg_huffman_param	√
	hi_mpi_venc_get_jpeg_huffman_param	√
	hi_mpi_venc_compact_jpeg_tables	√
	hi_mpi_venc_send_jpege_frame	√
	hi_mpi_venc_get_jpege_predicted_size	√
	hi_mpi_venc_set_roi_attr	x
	hi_mpi_venc_get_roi_attr	x
	hi_mpi_venc_set_ref_param	x
	hi_mpi_venc_get_ref_param	x
	hi_mpi_venc_set_h264_vui	√

目录	接口	Atlas A2训练系列产品/Atlas 800I A2推理产品
	hi_mpi_venc_get_h264_vui	√
	hi_mpi_venc_set_h265_vui	√
	hi_mpi_venc_get_h265_vui	√
	hi_mpi_venc_set_cu_pred	x
	hi_mpi_venc_get_cu_pred	x
	hi_mpi_venc_set_intra_refresh	x
	hi_mpi_venc_get_intra_refresh	x
	hi_mpi_venc_set_chn_attr	x
	hi_mpi_venc_get_chn_attr	x
媒体数据处理V2>PNGD功能	hi_mpi_pngd_create_chn	√
	hi_mpi_pngd_destroy_chn	√
	hi_mpi_pngd_send_stream	√
	hi_mpi_pngd_get_image_data	√
	hi_mpi_png_get_image_info	√
Profiling AscendCL API for Extension (Profiling AscendCL API扩展接口)	aclprofInit	√
	aclprofSetConfig	√
	aclprofStart	√
	aclprofStop	√
	aclprofFinalize	√
Profiling AscendCL API for Extension (Profiling AscendCL API扩展接口)	aclprofCreateStamp	√
	aclprofSetStampTraceMessage	√
	aclprofMark	√
	aclprofPush	√
	aclprofPop	√
	aclprofRangeStart	√
	aclprofRangeStop	√

目录	接口	Atlas A2训练系列产品/Atlas 800I A2推理产品
	aclprofDestroyStamp	√
Profiling AscendCL API for Subscription (订阅算子信息的Profiling AscendCL API)	aclprofModelSubscribe	√
	aclprofModelUnSubscribe	√
	aclprofGetOpDescSize	√
	aclprofGetOpNum	√
	aclprofGetOpTypeLen	√
	aclprofGetOpType	√
	aclprofGetOpNameLen	√
	aclprofGetOpName	√
	aclprofGetOpStart	√
	aclprofGetOpEnd	√
	aclprofGetOpDuration	√
	aclprofGetModelId	√
Profiling AscendCL API (PyTorch场景标记迭代时间)	aclprofGetStepTimestamp	√
数据传输>Tensor数据传输	acltdtCreateChannel	x
	acltdtSendTensor	√
	acltdtReceiveTensor	√
	acltdtStopChannel	√
	acltdtDestroyChannel	√
	acltdtCreateChannelWithCapacity	√
	acltdtQueryChannelSize	√
数据传输>共享队列管理	acltdtCreateQueue	√
	acltdtDestroyQueue	√
	acltdtEnqueueData	√
	acltdtDequeueData	√
	acltdtEnqueue	x
	acltdtDequeue	x

目录	接口	Atlas A2训练系列产品/Atlas 800I A2推理产品
	acldtdBindQueueRoutes	√
	acldtdUnbindQueueRoutes	√
	acldtdQueryQueueRoutes	√
	acldtdGrantQueue	x
	acldtdAttachQueue	x
数据传输>共享Buffer管理	acldtdAllocBuf	√
	acldtdFreeBuf	√
	acldtdGetBufData	√
	acldtdSetBufUserData	x
	acldtdGetBufUserData	x
	acldtdSetBufDataLen	x
	acldtdGetBufDataLen	x
	acldtdCopyBufRef	x
	acldtdAppendBufChain	x
	acldtdGetBufChainNum	x
	acldtdGetBufFromChain	x
特征向量检索	该章节下的所有接口	x

1.1.8 V1.7.0 及以上版本

针对以下产品，在V1.6.0基础上，新增了接口：

- Atlas 200/300/500 推理产品
- Atlas 训练系列产品
- Atlas 推理系列产品
- Atlas 200/500 A2推理产品
- Atlas A2训练系列产品/Atlas 800I A2推理产品

各版本对AscendCL接口的支持情况如下所示，各标识的含义如下：

- √：支持。
- x：不支持。
- 废弃：后续版本待废弃，具体废弃接口说明请参见[1.2 废弃接口/返回码列表](#)。

目录	接口	Atlas 200/300/500 推理产品	Atlas 训练系列产品	Atlas 推理系列产品	Atlas 200/500 A2 推理产品	Atlas A2训练系列产品/ Atlas 800I A2推理产品
单算子执行>单算子模型执行	aclopStartDump Args	x	√	x	x	√
	aclopStopDump Args	x	√	x	x	√
运行时管理>执行控制	aclrtSubscribeHostFunc	x	x	x	x	x
	aclrtProcessHostFunc	x	x	x	x	x
	aclrtUnSubscribeHostFunc	x	x	x	x	x
单算子执行>单算子API执行>DVPP类算子接口	acldvppNormalize	x	x	x	x	√
	acldvppDecodeJpeg	x	x	x	x	√
	acldvppResize	x	x	x	x	√
	acldvppImgToTensor	x	x	x	x	√
	acldvppAdjustBrightness	x	x	x	x	√
	acldvppAdjustHue	x	x	x	x	√
	acldvppAdjustSaturation	x	x	x	x	√
	acldvppAdjustContrast	x	x	x	x	√
单算子执行>单算子API执行>融合类算子接口	aclnnPromptFlashAttention	x	x	x	x	√
	aclnnIncrFlashAttention	x	x	x	x	√
	aclnnFlashAttentionScore	x	x	x	x	√

目录	接口	Atlas 200/300/500 推理产品	Atlas 训练系列产品	Atlas 推理系列产品	Atlas 200/500 A2 推理产品	Atlas A2训练系列产品/ Atlas 800I A2推理产品
	aclnnFlashAttentionScoreGrad	x	x	x	x	√
	aclnnMoeFFN	x	x	x	x	√
媒体数据处理 V2>VPC 图像处理 功能	hi_mpi_vpc_set_chn_opt_attr	x	x	x	x	x
运行时管理 >Stream 管理	aclrtStreamQuery	√	√	√	√	√
运行时管理 >内存 管理	aclrtReserveMemAddress	√	√	√	√	√
	aclrtReleaseMemAddress	√	√	√	√	√
	aclrtMallocPhysical	√	√	√	√	√
	aclrtFreePhysical	√	√	√	√	√
	aclrtMapMem	√	√	√	√	√
	aclrtUnmapMem	√	√	√	√	√
运行时管理 >Device 管理	aclrtQueryDeviceStatus	x	√	√	√	√

1.1.9 V1.8.0 及以上版本

- 针
- Atlas 200/300/500 推理产品
 - Atlas 训练系列产品
 - Atlas 推理系列产品

- Atlas 200/500 A2推理产品
- Atlas A2训练系列产品/Atlas 800I A2推理产品

各版本对AscendCL接口的支持情况如下所示，各标识的含义如下：

- √：支持。
- x：不支持。
- 废弃：后续版本待废弃，具体废弃接口说明请参见[1.2 废弃接口/返回码列表](#)。

目录	接口	Atlas 200/300/500 推理产品	Atlas 训练系列产品	Atlas 推理系列产品	Atlas 200/500 A2 推理产品	Atlas A2训练系列产品/Atlas 800I A2推理产品
单算子执行>单算子API执行>DVPP算子接口	acldvppHorizontalFlip	x	x	x	x	√
	acldvppVerticalFlip	x	x	x	x	√
	acldvppGaussianBlur	x	x	x	x	√
	acldvppWarpAffine	x	x	x	x	√
	acldvppWarpPerspective	x	x	x	x	√
	acldvppCropAndResize	x	x	x	x	√
	acldvppPad	x	x	x	x	√
	acldvppInit	x	x	x	x	√
	acldvppFinalize	x	x	x	x	√
	acldvppRotate	x	x	x	x	√
	aclRgbToGrayscale	x	x	x	x	√
	acldvppCrop	x	x	x	x	√
	acldvppInvert	x	x	x	x	√
	acldvppPosterize	x	x	x	x	√
	acldvppSolarize	x	x	x	x	√

目录	接口	Atlas 200/300/ 500 推理 产品	Atlas 训 练系列 产品	Atlas 推理系 列产品	Atlas 200/5 00 A2 推理产 品	Atlas A2训练 系列产 品/ Atlas 800I A2推理 产品
媒体数据 处理 V2>VEN C视频编 码功能/ JPEG图 片编码功 能	hi_mpi_venc_set _slice_split	x	x	√	√	x
	hi_mpi_venc_get _slice_split	x	x	√	√	x
运行时管 理 >Kernel 加载与执 行	aclrtCreateBinar y	√	√	√	√	√
	aclrtDestroyBina ry	√	√	√	√	√
	aclrtBinaryLoad	√	√	√	√	√
	aclrtBinaryUnLo ad	√	√	√	√	√
	aclrtBinaryGetFu nction	√	√	√	√	√
	aclrtLaunchKern el	√	√	√	√	√
媒体数据 处理 V2>HD MI外设	hi_mpi_hdmi_ge t_sink_capability	x	x	x	√	x
	hi_mpi_hdmi_re gister_callback	x	x	x	√	x
运行时管 理>内存 管理	aclrtMemExport ToShareableHan dle	√	√	√	√	√
	aclrtDeviceGetB areTgid	√	√	√	√	√
	aclrtMemSetPid ToShareableHan dle	√	√	√	√	√
	aclrtMemImport FromShareableH andle	√	√	√	√	√

目录	接口	Atlas 200/300/500 推理产品	Atlas 训练系列产品	Atlas 推理系列产品	Atlas 200/500 A2 推理产品	Atlas A2训练系列产品/ Atlas 800I A2推理产品
	aclrtMemGetAllLocationGranularity	√	√	√	√	√

1.1.10 V1.9.0 及以上版本

针对以下产品，在V1.8.0基础上，新增了接口：

- Atlas 200/300/500 推理产品
- Atlas 训练系列产品
- Atlas 推理系列产品
- Atlas 200/500 A2推理产品
- Atlas A2训练系列产品/Atlas 800I A2推理产品

各版本对AscendCL接口的支持情况如下所示，各标识的含义如下：

- √：支持。
- x：不支持。
- 废弃：后续版本待废弃，具体废弃接口说明请参见[1.2 废弃接口/返回码列表](#)。

目录	接口	Atlas 200/300/500 推理产品	Atlas 训练系列产品	Atlas 推理系列产品	Atlas 200/500 A2 推理产品	Atlas A2训练系列产品/ Atlas 800I A2推理产品
运行时管理>Event 管理	aclrtCreateEventExWithFlag	√	√	√	√	√
单算子执行>单算子API执行>DVPP 类算子接口	acldvppEqualize	x	x	x	x	√
	acldvppAdjustSharpness	x	x	x	x	√
	acldvppAutoContrast	x	x	x	x	√
	acldvppErase	x	x	x	x	√

目录	接口	Atlas 200/300/ 500 推理 产品	Atlas 训 练系列 产品	Atlas 推理系 列产品	Atlas 200/5 00 A2 推理产 品	Atlas A2训练 系列产 品/ Atlas 800I A2推理 产品
	acldvppConvert Color	x	x	x	x	√
系统配置	aclGetCannAttri buteList	√	√	√	√	√
	aclGetCannAttri bute	√	√	√	√	√
	aclGetDeviceCap ability	√	√	√	√	√
数据传输 >Tensor 数据传输	acltdtGetSliceInf oFromItem	x	√	x	x	√
模型推理 >模型执 行	acldumpRegCall back	√	√	√	√	√
	acldumpUnregC allback	√	√	√	√	√
	aclmdlQueryExe OMDesc	x	x	x	x	x
媒体数据 处理V2> 公共接口	hi_mpi_vb_creat e_pool	x	x	x	x	x
	hi_mpi_vb_destr oy_pool	x	x	x	x	x
	hi_mpi_vb_get_b lock	x	x	x	x	x
	hi_mpi_vb_relea se_block	x	x	x	x	x
	hi_mpi_vb_handl e_to_addr	x	x	x	x	x
	hi_mpi_vb_block _copyref	x	x	x	x	x
	hi_mpi_sys_full_i nit	x	x	x	x	x

1.1.11 V1.10.0 及以上版本

针对以下产品，在V1.9.0基础上，新增了接口：

- Atlas 200/300/500 推理产品
- Atlas 训练系列产品
- Atlas 推理系列产品
- Atlas 200/500 A2推理产品
- Atlas A2训练系列产品/Atlas 800I A2推理产品

各版本对AscendCL接口的支持情况如下所示，各标识的含义如下：

- √：支持。
- x：不支持。
- 废弃：后续版本待废弃，具体废弃接口说明请参见[1.2 废弃接口/返回码列表](#)。

目录	接口	Atlas 200/300/500 推理产品	Atlas 训练系列产品	Atlas 推理系列产品	Atlas 200/500 A2推理产品	Atlas A2训练系列产品/Atlas 800I A2推理产品
数据类型及其操作接口 >aclmdl AIPP	aclmdlGetAippDataSize	√	√	√	√	√
单算子执行>单算子API执行>DVPP类算子接口	acldvppEncodeJpeg	x	x	x	x	√
Profiling 数据采集>Profiling AscendCL API for Extension (Profiling AscendCL API扩展接口)	aclprofMarkEx	√	√	√	√	√

目录	接口	Atlas 200/300/500 推理产品	Atlas 训练系列产品	Atlas 推理系列产品	Atlas 200/500 A2 推理产品	Atlas A2训练系列产品/Atlas 800I A2推理产品
媒体数据处理 V2>VENC C视频编码功能/ JPEG图片编码功能	hi_mpi_venc_send_frame_ex	x	x	√	x	x
	hi_mpi_venc_get_qpmap_stride	x	x	√	x	x
	hi_mpi_venc_get_qpmap_size	x	x	√	x	x
媒体数据处理 V2>VENC C视频编码功能/ JPEG图片编码功能	hi_mpi_venc_set_streamcopy_param	x	x	√	x	x
	hi_mpi_venc_get_streamcopy_param	x	x	√	x	x

1.1.12 V1.11.0 及以上版本

1.1.12.1 接口列表 (Atlas 200/300/500 推理产品&Atlas 训练系列产品&Atlas 推理系列产品&Atlas 200/500 A2 推理产品&Atlas A2 训练系列产品/Atlas 800I A2 推理产品)

针对以下产品，在V1.10.0基础上，新增了接口：

- Atlas 200/300/500 推理产品
- Atlas 训练系列产品
- Atlas 推理系列产品
- Atlas 200/500 A2推理产品
- Atlas A2训练系列产品/Atlas 800I A2推理产品

各版本对AscendCL接口的支持情况如下所示，各标识的含义如下：

- √：支持。
- x：不支持。
- 废弃：后续版本待废弃，具体废弃接口说明请参见[1.2 废弃接口/返回码列表](#)。

目录	接口	Atlas 200/300/500 推理产品	Atlas 训练系列产品	Atlas 推理系列产品	Atlas 200/500 A2 推理产品	Atlas A2训练系列产品/ Atlas 800I A2推理产品
运行时管理>异常处理	aclrtGetMemUceInfo	x	x	x	x	√
	aclrtDeviceTaskAbort	x	x	x	x	√
	aclrtMemUceRepair	x	x	x	x	√
数据传输>Tensor数据传输	acltdtCleanChannel	x	√	x	x	√
系统配置	aclopSetMaxOpQueueNum	√	√	√	√	√
模型推理>模型加载和卸载	aclmdlBundleLoadFromFile	√	√	√	√	√
	aclmdlBundleLoadFromMem	√	√	√	√	√
	aclmdlBundleGetModelId	√	√	√	√	√
	aclmdlBundleGetModelNum	√	√	√	√	√
	aclmdlBundleUnload	√	√	√	√	√
运行时管理>Device管理	aclrtResetDeviceForce	√	√	√	√	√
数据类型及其操作接口>aclmdlDesc	aclmdlGetInputDimsRange	√	√	√	√	√
运行时管理>内存管理	aclrtCmoAsync	x	x	x	x	√

目录	接口	Atlas 200/300/500 推理产品	Atlas 训练系列产品	Atlas 推理系列产品	Atlas 200/500 A2 推理产品	Atlas A2训练系列产品/ Atlas 800I A2推理产品
运行时管理>Device管理	aclrtSynchronizeDeviceWithTimeout	√	√	√	√	√
运行时管理>异常处理	aclrtPeekAtLastError	√	√	√	√	√
	aclrtGetLastError	√	√	√	√	√

1.2 废弃接口/返回码列表

接口

- [aclopExecute](#)接口
此接口后续版本会废弃，请使用[aclopExecuteV2](#)接口。
- [aclGetTensorDescDim](#)接口
此接口后续版本会废弃，请使用[aclGetTensorDescDimV2](#)接口。
- [aclGetDataBufferSize](#)接口
此接口后续版本会废弃，请使用[aclGetDataBufferSizeV2](#)接口。
- [aclSetTensorStorageShape](#)接口
此接口后续版本会废弃，请使用[aclSetTensorShape](#)接口。
- [aclSetTensorStorageFormat](#)接口
此接口后续版本会废弃，请使用[aclSetTensorFormat](#)接口。
- [aclrtQueryEvent](#)接口
此接口后续版本会废弃，请使用[aclrtQueryEventStatus](#)接口。
- [aclopSetCompileFlag](#)接口
此接口后续版本会废弃，请使用[aclSetCompileopt](#)接口设置编译选项。

返回码

- [ACL_ERROR_NONE](#)返回码
此返回码后续版本会废弃，请使用[ACL_SUCCESS](#)返回码。
- [ACL_ERROR_NOT_STATIC_AIPP](#)
此返回码后续版本会废弃，请使用[ACL_ERROR_GE_AIPP_NOT_EXIST](#)返回码。

- **ACL_ERROR_STREAM_NOT_SUBSCRIBE**
此返回码后续版本会废弃，请使用**ACL_ERROR_RT_STREAM_NO_CB_REG**返回码。
- **ACL_ERROR_THREAD_NOT_SUBSCRIBE**
此返回码后续版本会废弃，请使用**ACL_ERROR_RT_THREAD_SUBSCRIBE**返回码。
- **ACL_ERROR_WAIT_CALLBACK_TIMEOUT**
此返回码后续版本会废弃，请使用**ACL_ERROR_RT_REPORT_TIMEOUT**返回码。
- **ACL_ERROR_INVALID_DEVICE**
此返回码后续版本会废弃，请使用**ACL_ERROR_RT_INVALID_DEVICEID**返回码。
- **ACL_ERROR_GROUP_NOT_SET**
此返回码后续版本会废弃，请使用**ACL_ERROR_RT_GROUP_NOT_SET**返回码。
- **ACL_ERROR_GROUP_NOT_CREATE**
此返回码后续版本会废弃，请使用**ACL_ERROR_RT_GROUP_NOT_CREATE**返回码。

1.3 同步&异步 API 说明

CANN支持以下几类显式同步，调用此类接口后，主机线程会阻塞直到相关的任务执行完成。

- **设备同步：aclrtSynchronizeDevice**
阻塞当前主机线程直到device上所有显式或隐式创建的stream都完成所有先前下发的任务。应该尽量少使用该函数，以免拖延主机运行。
- **流同步：aclrtSynchronizeStream**
阻塞当前主机线程直到指定的stream中完成所有下发的任务。
- **事件同步：aclrtSynchronizeEvent**
阻塞当前主机线程直到指定的Event事件完成。属于更细粒度的同步。

对于**异步接口**，主机线程调用异步接口后仅代表下发任务，在任务未完成前，异步接口已向主机线程返回成功。用户需要调用上面的显式同步接口阻塞主机线程，等待任务完成，否则可能会导致训练或推理等业务异常、Device断链掉卡等未知情况。

1.4 系统配置

1.4.1 aclInit

函数功能

AscendCL初始化函数。

约束说明

- 使用AscendCL接口开发应用时，必须先调用[aclInit](#)接口，否则可能会导致后续系统内部资源初始化出错，进而导致其它业务异常。
- 一个进程内只能调用一次[aclInit](#)接口，且与[aclFinalize](#)去初始化接口配对使用。

函数原型

[aclError](#) [aclInit](#)(const char *configPath)

参数说明

参数名	输入/输出	说明
configPath	输入	配置文件所在路径（包含文件名）的指针。配置文件内容为json格式（json文件内的“{”的层级最多为10，“[”的层级最多为10）。 初始化AscendCL时，可通过该配置文件开启或设置以下功能，如果以下的默认配置已满足需求，无需修改，可向aclInit接口中传入NULL，或者可将配置文件配置为空json串（即配置文件中只有{}）。 ● Dump信息配置，包括以下配置： - 模型Dump配置（用于导出模型中每一层算子输入和输出数据）、单算子Dump配置（用于导出单个算子的输入和输出数据），导出的数据用于与指定模型或算子进行比对，定位精度问题，配置示例、说明及约束请参见配置文件示例（模型Dump配置、单算子Dump配置）。默认不启用该dump配置。 通过本接口启用Dump配置，需通过dump_path参数配置Dump数据的落盘路径，如果无需落盘，则可以通过回调函数获取Dump数据，请参见1.7.2.19 acldumpRegCallback。 - 异常算子Dump配置（用于导出异常算子的输入输出数据、workspace信息、Tiling信息），导出的数据用于分析AI Core Error问题，配置示例请参见配置文件示例（异常算子Dump配置）。默认不启用该dump配置。 - 溢出算子Dump配置（用于导出模型中溢出算子的输入和输出数据），导出的数据用于分析溢出原因，定位模型精度的问题，配置示例、说明及约束请参见配置文件示例（溢出算子Dump配置）。默认不启用该dump配置。 - 算子Dump Watch模式配置（用于开启指定算子输出数据的观察模式），在定位部分算子精度问题且已排除算子本身的计算问题后，若怀疑被其它算子踩踏内存导致精度问题，可开启Dump Watch模式，配置示例及约束请参见配置文件示例（算子Dump Watch模式配置）。默认不开启Dump Watch模式。 ● Profiling采集信息配置，配置示例、说明及约束请参见《性能调优工具使用指南》。默认不启用Profiling采集信息配置。 ● 算子缓存信息老化配置，通过单算子模型方式执行单个算子时（aclopUpdateParams接口执行单算子除外），为节约内存和平衡调用性能，可通过max_opqueue_num参数配置“算子类型-单算子模型”映射队列的最大长度，如果长度达到最大，则会先删除长期未使用的映射信息以及缓存中的单算子模型，再加载最新的映射信息以及对应的单算子模型。如果不配置映射队列的最大长度，则默认最大长度为20000。示例

参数名	输入/输出	说明
		及约束说明请参见配置文件示例（算子缓存信息老化配置）。 关于单算子模型的介绍请参见单算子调用流程。 ● 错误信息上报模式配置，用于控制aclGetRecentErrMsg接口按进程或线程级别获取错误信息，默认按线程级别。示例请参见配置文件示例（错误信息上报模式配置）。 ● 默认Device配置（用于配置默认的计算设备），配置示例、说明请参见默认Device配置示例。 若同时通过aclrtSetDevice接口指定Device，则aclrtSetDevice接口优先级高。 如果用户开启默认Device功能后，若需要显式创建Context，则需要调用aclrtSetDevice，否则可能会导致业务异常。 说明 建议不要同时配置dump信息和Profiling采集信息，否则dump操作会影响系统性能，导致Profiling采集的性能数据指标不准确。

返回值说明

返回0表示成功，返回其他值表示失败。

配置文件示例（模型 Dump 配置、单算子 Dump 配置）

模型Dump配置示例如下：

```
{
  "dump":{
    "dump_list":[
      { "model_name":"ResNet-101"
      },
      {
        "model_name":"ResNet-50",
        "layer":[
          "conv1 conv1_relu",
          "res2a_branch2ares2a_branch2a_relu",
          "res2a_branch1",
          "pool1"
        ]
      }
    ],
    "dump_path":"$HOME/output",
    "dump_mode":"output",
    "dump_op_switch":"off",
    "dump_data":"tensor"
  }
}
```

单算子Dump配置示例如下：

```
{
  "dump":{
    "dump_path":"output",
    "dump_list":[],
  }
}
```

```
"dump_op_switch":"on",  
  "dump_data":"tensor"  
}
```

表 1-1 acl.json 文件格式说明

配置项	参数说明
dump_list	(必选) 待dump数据的整网模型列表。 创建模型dump配置信息，当存在多个模型需要dump时，需要每个模型之间用英文逗号隔开。 在单算子执行场景下，dump_list为空。
model_name	模型名称，各个模型的model_name值须唯一。 - 模型加载方式为文件加载时，填入模型文件的名称，不需要带后缀名；也可以配置为ATC模型文件转换后的json文件里的最外层"name"字段对应值。 - 模型加载方式为内存加载时，配置为ATC模型文件转换后的json文件里的最外层"name"字段对应值。
layer	IO性能相对较差时，可能会出现由于数据量过大导致执行超时，所以不建议全量dump，请指定算子进行dump。通过该字段可以指定需要dump的算子名，支持指定为ATC模型转换后的算子名，也支持指定为转换前的原始算子名，配置时需注意： - 需按格式配置，每行配置模型中的一个算子名，且每个算子之间用英文逗号隔开。 - 用户可以无需设置model_name，此时会默认dump所有model下的相应算子。如果配置了model_name，则dump对应model下的相应算子。 - 若指定的算子其输入涉及data算子，会同时将data算子信息dump出来；若需dump data算子，需要一并填写data节点算子的后继节点，才能dump出data节点算子数据。 - 当需要dump模型中所有算子时，不需要包含layer字段。
dump_path	(必选) dump数据文件存储到运行环境的目录，该目录需要提前创建且确保安装时配置的运行用户具有读写权限。 支持配置绝对路径或相对路径： - 绝对路径配置以 “/” 开头，例如：\$HOME/output。 - 相对路径配置直接以目录名开始，例如：output。

配置项	参数说明
dump_mode	dump数据模式。 • input: dump算子的输入数据。 • output: dump算子的输出数据，默认取值output。 • all: dump算子的输入、输出数据。 注意，配置为all时，由于部分算子在执行过程中会修改输入数据，例如集合通信类算子HcomAllGather、HcomAllReduce等，因此系统在进行dump时，会在算子执行前dump算子输入，在算子执行后dump算子输出，这样，针对同一个算子，算子输入、输出的dump数据是分开落盘，会出现多个dump文件，在解析dump文件后，用户可通过文件内容判断是输入还是输出。
dump_level	dump数据级别，取值： • op: 按算子级别dump数据。 • kernel: 按kernel级别dump数据。 • all: 默认值，op和kernel级别的数据都dump。 默认配置下，dump数据文件会比较多，例如有一些aclnn开头的dump文件，若用户对dump性能有要求或内存资源有限时，则可以将该参数设置为op级别，以便提升dump性能、精简dump数据文件数量。 说明 算子是一个运算逻辑的表示（如加减乘除运算），kernel是运算逻辑真正进行计算处理的实现，需要分配具体的计算设备完成计算。
dump_op_switch	单算子模型dump数据开关。 • on: 开启单算子模型dump。 • off: 关闭单算子模型dump，默认取值off。

配置项	参数说明
dump_step	指定采集哪些迭代的Dump数据。推理场景无需配置。不配置该参数，默认所有迭代都会产生dump数据，数据量比较大，建议按需指定迭代。 多个迭代用“ ”分割，例如：0 5 10；也可以用“-”指定迭代范围，例如：0 3-5 10。 配置示例： <pre>{ "dump":{ "dump_list":[.....], "dump_path": "\$HOME/output", "dump_mode": "output", "dump_op_switch": "off" "dump_step": "0 3-5 10" } }</pre> 说明 训练场景下，若通过acl.json中的dump_step参数指定采集哪些迭代的Dump数据，又同时在GEInitialize接口中配置了ge.exec.dumpStep参数（该参数也用于指定采集哪些迭代的Dump数据），则以最后配置的参数为准。GEInitialize接口的详细介绍请参见《Ascend Graph开发指南》。
dump_data	算子dump内容类型，取值： • tensor: dump算子数据，默认为tensor。 • stats: dump算子统计数据，结果文件为csv格式，文件中包含算子名称、输入/输出的数据类型、最大值、最小值等。 通常dump数据量太大并且耗时长，可以先dump算子统计数据，根据统计数据识别可能异常的算子，然后再dump算子数据。 模型Dump场景下，会根据dump_mode处的配置统计算子输入或算子输出或算子输入&输出的信息。

配置文件示例（异常算子 Dump 配置）

将dump_scene参数设置为lite_exception，启用异常算子Dump功能，同时可配合使用ASCEND_WORK_PATH环境变量配置落盘路径，否则落盘在应用程序的当前执行目录下。**注意**，异常算子Dump配置，不能与模型Dump配置或单算子Dump配置同时开启，否则模型Dump或单算子Dump不生效。

配置文件中的示例内容如下：

```
{
  "dump":{
    "dump_scene": "lite_exception"
  }
}
```

此处收集的dump文件无法通过文本工具直接查看其内容，若需查看dump文件内容，先将dump文件转换为numpy格式文件后，再通过Python查看numpy格式文件，详细转换步骤请参见《[精度调试工具使用指南](#)》中的“查看dump数据文件”章节。

配置文件示例（溢出算子 Dump 配置）

溢出算子Dump配置的相关约束说明如下：

- 将dump_debug配置为on表示开启溢出算子配置，不配置dump_debug或将dump_debug配置为off表示不开启溢出算子配置。
- 若开启溢出算子配置，则dump_path必须配置，表示导出dump文件的存储路径。

支持配置绝对路径或相对路径：

- 绝对路径配置以“/”开头，例如：/home。
- 相对路径配置直接以目录名开始，例如：output。

获取导出的数据文件后，如何解析请参见溢出算子数据采集及分析。

- 溢出算子Dump配置，不能与模型Dump配置或单算子Dump配置同时开启，否则会返回报错。
- 仅支持采集AI Core算子的溢出数据。

配置文件中的示例内容如下：

```
{
  "dump":{
    "dump_path":"output",
    "dump_debug":"on"
  }
}
```

配置文件示例（算子 Dump Watch 模式配置）

将dump_scene参数设置为watcher，开启算子Dump Watch模式，详细配置说明及约束如下：

- 若开启算子Dump Watch模式，则不支持同时开启溢出算子Dump（配置dump_debug参数）或开启单算子模型Dump（配置dump_op_switch参数），否则报错。
- 在dump_list中，通过layer参数配置可能踩踏其它算子内存的算子名称，通过watcher_nodes参数配置可能被其它算子踩踏输出内存导致精度有问题的算子名称。
 - 若不指定layer，则模型内所有支持Dump的算子在执行后，都会将watcher_nodes中配置的算子的输出Dump出来。
 - layer和watcher_node处配置的算子都必须是静态图、静态子图中的算子，否则不生效。
 - 若layer和watcher_node处配置的算子名称相同，或者layer处配置的是集合通信类算子（算子类型以Hcom开头，例如HcomAllReduce），则不导出dump文件。
 - 对于融合算子，watcher_node处配置的算子名称必须是融合后的算子名称，若配置融合前的算子名称，则不导出dump文件。
 - dump_list内暂不支持配置model_name。
- 开启算子Dump Watch模式，则dump_path必须配置，表示导出dump文件的存储路径。

支持配置绝对路径或相对路径：

- 绝对路径配置以“/”开头，例如：/home。
- 相对路径配置直接以目录名开始，例如：output。

- 通过dump_mode参数控制导出watcher_nodes中所配置算子的哪部分数据，当前仅支持配置为output。

配置文件中的示例内容如下，配置效果为：当执行完A算子和B算子时，会把C算子和D算子的输出Dump出来，导出C算子opType.A_To_C.*、D算子opType.A_To_D.*、C算子opType.B_To_C.*、D算子opType.B_To_D.*四个dump文件，用于排查A算子、B算子是否会踩踏C算子、D算子的输出内存。

```
{
  "dump":{
    "dump_list":[
      {
        "layer":["A", "B"],
        "watcher_nodes":["C", "D"]
      }
    ],
    "dump_path":"/home/",
    "dump_mode":"output",
    "dump_scene":"watcher"
  }
}
```

此处收集的dump文件无法通过文本工具直接查看其内容，若需查看dump文件内容，先将dump文件转换为numpy格式文件后，再通过Python查看numpy格式文件，详细转换步骤请参见《[精度调试工具使用指南](#)》中的“查看dump数据文件”章节。

配置文件示例（算子缓存信息老化配置）

算子缓存信息老化配置的相关约束说明如下：

- 对于静态加载的算子，调用[aclopSetModelDir](#)接口加载指定目录下的单算子模型或调用[aclopLoad](#)接口加载指定单算子模型时，老化配置无效，不会对该部分的算子信息做老化。
- 在线编译算子的场景下，调用[aclopCompile](#)接口编译算子或调用[aclopCompileAndExecute](#)接口编译执行算子时，接口内部会按照入参加载单算子模型，老化配置有效。

如果用户调用[aclopCompile](#)接口编译算子、调用[aclopExecuteV2](#)接口执行算子，则在编译算子后需及时执行算子，否则可能导致执行算子时，算子信息已被老化，需要重新编译。建议调用[aclopCompileAndExecuteV2](#)接口编译执行算子。

- AscendCL内部分开维护固定Shape和动态Shape算子的映射队列，最大长度都为max_opqueue_num参数值。
- max_opqueue_num参数值为静态加载算子的单算子模型个数和在线编译算子的单算子模型个数的总和，因此max_opqueue_num参数值应大于当前进程中可用的、静态加载算子的单算子模型个数，否则会导致在线编译算子的信息无法老化。

配置文件中的示例内容如下：

```
{
  "max_opqueue_num": "10000"
}
```

配置文件示例（错误信息上报模式配置）

err_msg_mode参数取值范围：0为默认值，表示按线程级别获取错误信息；1表示按进程级别获取错误信息。

配置文件中的示例内容如下：

```
{  
    "err_msg_mode": "1"  
}
```

默认 Device 配置示例

default_device参数处设置Device ID，Device ID可设置为0或十进制正整数，用户可用[aclrtGetDeviceCount](#)接口获取可用的Device数量后，这个Device ID的取值范围：
[0, (可用的Device数量-1)]。

配置文件中的示例内容如下：

```
{  
    "defaultDevice":{  
        "default_device":"0"  
    }  
}
```

相关接口

AscendCL还提供了其它使能Dump或Profiling的接口，如下，与aclInit不同的是，以下这些接口相对灵活，可以在一个进程内调用多次接口，每次调用接口时可以基于不同的Dump配置或Profiling配置。

- 获取Dump数据，参见[1.7.2.17 aclmdlInitDump](#)、[1.7.2.18 aclmdlSetDump](#)、[1.7.2.21 aclmdlFinalizeDump](#)。
- 获取Profiling数据，参见[1.10 Profiling数据采集](#)。

参考资源

接口调用示例，参见AscendCL初始化。

1.4.2 aclFinalize

函数功能

AscendCL去初始化函数，用于释放进程内的AscendCL相关资源。

在该接口内，默认增加2000ms延时（实际最大延时可达2000ms），用于Device业务日志回传，保证ERROR级别和EVENT级别日志不丢失，您可以将ASCEND_LOG_DEVICE_FLUSH_TIMEOUT环境变量设置为0（命令示例：export ASCEND_LOG_DEVICE_FLUSH_TIMEOUT=0），去除该默认延时。关于ASCEND_LOG_DEVICE_FLUSH_TIMEOUT环境变量的详细描述请参见《[日志参考](#)》。

约束说明

应用进程退出前，应显式调用该接口实现AscendCL去初始化，否则可能会导致异常，例如应用进程退出时有异常报错。

不建议在析构函数中调用aclFinalize接口，否则在进程退出时可能由于单例析构顺序未知而导致进程异常退出的问题。

函数原型

aclError aclFinalize()

参数说明

无

返回值说明

返回0表示成功，返回其他值表示失败。

参考资源

接口调用示例，参见AscendCL初始化。

1.4.3 aclrtGetVersion

函数功能

查询接口版本号，AscendCL接口版本号命名采用：A.B.C模式，其中，A表示有不兼容修改，B表示新增接口，C表示bug修复。

函数原型

aclError aclrtGetVersion(int32_t *majorVersion, int32_t *minorVersion, int32_t *patchVersion)

参数说明

参数名	输入/输出	说明
majorVersion	输出	主版本号的指针，从1开始，如果出现接口的不兼容变更时，加1。
minorVersion	输出	次版本号的指针，从0开始，按照迭代周期，有新增接口时加1。
patchVersion	输出	补丁版本号的指针，从0开始，表示本版本仅解决了问题，在majorVersion、minorVersion不变的情况下加1；但majorVersion、minorVersion增加的时候，patchVersion一般为0。

返回值说明

返回0表示成功，返回其他值表示失败。

1.4.4 aclSetCompileopt

函数功能

用户可调用本接口设置对应的编译选项，该选项为进程级全局共享，算子或模型开始编译时，以当前的编译选项为准，一次编译过程中不会变更。

用户可以在调用[aclopCompile](#)接口编译算子或[aclopCompileAndExecute](#)接口编译执行算子前，调用本接口设置编译选项。

函数原型

[aclError](#) [aclSetCompileopt](#)([aclCompileOpt](#) opt, const char *value)

参数说明

参数名	输入/输出	说明
opt	输入	编译选项。
value	输入	编译选项具体值的指针。

返回值说明

返回0表示成功，返回其他值表示失败。

1.4.5 [aclGetCompileopt](#)

函数功能

用户可调用本接口获取编译选项的值。

用户可以调用本接口获取当前编译选项的值，在调用[aclSetCompileopt](#)前，本接口返回的是各编译选项的默认值；在调用[aclSetCompileopt](#)后，本接口返回的是用户设置的值。

函数原型

[aclError](#) [aclGetCompileopt](#)([aclCompileOpt](#) opt, char *value, size_t length)

参数说明

参数名	输入/输出	说明
opt	输入	编译选项。
value	输出	存放编译选项值的内存指针。 用户在调用本接口前需要申请一块内存来保存编译选项的值，内存大小可以通过 aclGetCompileoptSize 获得。
length	输入	value指向内存的大小，单位：Byte。

返回值说明

返回0表示成功，返回其他值表示失败。

1.4.6 aclGetCompileoptSize

函数功能

用户可调用本接口获取存放编译选项值的内存大小。

用户在调用[aclGetCompileopt](#)前需要申请一块内存来保存编译选项的值，内存大小可以通过调用本接口获得。

函数原型

`size_t aclGetCompileoptSize(aclCompileOpt opt)`

参数说明

参数名	输入/输出	说明
opt	输入	编译选项。

返回值说明

返回存放该编译选项值的内存大小，单位：Byte。

1.4.7 aclGetRecentErrMsg

函数功能

获取并清空与本接口在同一个进程或线程中的其它AscendCL接口调用失败时的错误描述信息。

获取进程级别、还是线程级别的错误描述信息由[aclInit](#)接口中的err_msg_mode配置控制，默认线程级别。

约束说明

建议在每次调用AscendCL接口失败时都调用[aclGetRecentErrMsg](#)接口，以便获取调用AscendCL接口异常时的错误描述信息，用于定位问题。同一个进程或线程中多次调用[aclGetRecentErrMsg](#)接口后，只有最后一次调用[aclGetRecentErrMsg](#)接口返回的错误描述字符串的指针有效，之前[aclGetRecentErrMsg](#)接口返回的错误描述字符串指针不能使用，否则可能导致内存非法访问。

如果没有在每次调用AscendCL接口失败时调用[aclGetRecentErrMsg](#)接口，则可能导致错误信息堆积、丢失。

函数原型

`const char *aclGetRecentErrMsg()`

参数说明

无

返回值说明

返回错误描述字符串的指针。如果通过本接口获取到多条错误描述信息，最上面的错误描述信息为最新的。

获取错误描述信息失败时，返回nullptr。

1.4.8 aclAppLog

函数功能

将日志记录到日志文件中。

AscendCL提供ACL_APP_LOG宏，封装了aclAppLog接口，推荐用户调用ACL_APP_LOG宏，传入日志级别、日志描述、fmt中的可变参数。日志文件的详细说明，请参见《[日志参考](#)》。

```
#define ACL_APP_LOG(level, fmt, ...) \
    aclAppLog(level, __FUNCTION__, __FILE__, __LINE__, fmt, ##__VA_ARGS__)
```

函数原型

void aclAppLog(aclLogLevel logLevel, const char *func, const char *file, uint32_t line, const char *fmt, ...)

参数说明

参数名	输入/输出	说明
logLevel	输入	日志级别。 typedef enum { ACL_DEBUG = 0, ACL_INFO = 1, ACL_WARNING = 2, ACL_ERROR = 3, } aclLogLevel;
func	输入	表示用户在哪个接口中调用aclAppLog接口，固定配置为__FUNCTION__
file	输入	表示用户在哪个文件中调用aclAppLog接口，固定配置为__FILE__
line	输入	表示用户在哪一行中调用aclAppLog接口，固定配置为__LINE__
fmt	输入	日志描述。 在调用格式化函数时，fmt中参数的类型、个数必须与实际参数类型、个数保持一致。
...	输入	fmt中的可变参数，根据日志内容添加。

返回值说明

无

调用示例

```
//若fmt中存在可变参数，需提前定义
uint32_t modelId = 1;
ACL_APP_LOG(ACL_INFO, "load model success, modelId is %u", modelId);
```

1.4.9 aclDataTypeSize

函数功能

获取aclDataType数据的大小，单位Byte。

函数原型

size_t aclDataTypeSize(aclDataType dataType)

参数说明

参数名	输入/输出	说明
dataType	输入	指定要获取大小的aclDataType数据。

返回值说明

返回aclDataType数据的大小，单位Byte。

1.4.10 aclFloat16ToFloat

函数功能

将aclFloat16类型的数据转换为float（指float32）类型的数据。

函数原型

float aclFloat16ToFloat(aclFloat16 value)

参数说明

参数名	输入/输出	说明
value	输入	待转换的数据。

返回值说明

转换后的数据。

1.4.11 aclFloatToFloat16

函数功能

将float（指float32）类型的数据转换为[aclFloat16](#)类型的数据。

函数原型

[aclFloat16](#) aclFloatToFloat16(float value)

参数说明

参数名	输入/输出	说明
value	输入	待转换的数据。

返回值说明

转换后的数据。

注意，由于C语言无float16类型，此处返回值使用uint16_t对数据进行存储。若用户需要打印，需自行将uint16_t解释成float16进行打印，或者转成float进行打印。

1.4.12 aclGenGraphAndDumpForOp

函数功能

构建单个算子的图并Dump，生成.txt文件。此处的图是指表达算子IR（Intermediate Representation）的结构，图编译过程中由于融合、拆分等原因会导致图不断变换、进而生成新的图，这些变换的目的是为了将用户表达的图适配到昇腾AI处理器上，获得更高性能。

使用场景：算子调优场景下，调用本接口构建单个算子的图并Dump，生成.txt文件，作为算子调优的输入数据之一。

函数原型

```
aclError aclGenGraphAndDumpForOp(const char *opType,  
int numInputs,  
const aclTensorDesc *const inputDesc[],  
const aclDataBuffer *const inputs[],  
int numOutputs,  
const aclTensorDesc *const outputDesc[],  
aclDataBuffer *const outputs[],  
const aclopAttr *attr,
```



```
aclopEngineType engineType,  
const char *graphDumpPath,  
const aclGraphDumpOption *graphDumpOpt)
```

参数说明

参数名	输入/输出	说明
opType	输入	指定算子类型名称的指针。
numInputs	输入	算子输入tensor的数量。
inputDesc	输入	算子输入tensor描述的指针数组。 需提前调用 aclCreateTensorDesc 接口创建aclTensorDesc类型。 inputDesc数组中的元素个数必须与numInputs参数值保持一致，且inputs数组与inputDesc数组中的元素必须一一对应。
inputs	输入	算子输入tensor的指针数组。 需提前调用 aclCreateDataBuffer 接口创建aclDataBuffer类型的数据。 inputs数组中的元素个数必须与numInputs参数值保持一致，且inputs数组与inputDesc数组中的元素必须一一对应。
numOutputs	输入	算子输出tensor的数量。
outputDesc	输入	算子输出tensor描述的指针数组。 需提前调用 aclCreateTensorDesc 接口创建aclTensorDesc类型。 outputDesc数组中的元素个数必须与numOutputs参数值保持一致，且outputs数组与outputDesc数组中的元素必须一一对应。
outputs	输入	算子输出tensor的指针数组。 需提前调用 aclCreateDataBuffer 接口创建aclDataBuffer类型的数据。 outputs数组中的元素个数必须与numOutputs参数值保持一致，且outputs数组与outputDesc数组中的元素必须一一对应。
attr	输入	算子属性的指针。 需提前调用 aclopCreateAttr 接口创建aclopAttr类型。
engineType	输入	算子执行引擎。

参数名	输入/输出	说明
graphDumpPath	输入	Dump单算子图所生成的文件（文件名后缀为.txt）的保存路径的指针。 路径需要由用户提前创建，路径中如果不指定文件名，则由AscendCL内部指定文件名。 如果多次Dump时，指定同一个路径，且文件名相同，则最新一次的文件会覆盖之前的文件。
graphDumpOpt	输入	单算子图的dump选项的指针。 当前该参数预留，仅支持传入nullptr。

返回值说明

返回0表示成功，返回其他值表示失败。

1.4.13 aclGetCannAttributeList

函数功能

查询运行环境中CANN软件 and 对应芯片支持的特性列表。

函数原型

```
aclError aclGetCannAttributeList(const aclCannAttr **cannAttrList, size_t *num)
```

参数说明

参数名	输入/输出	说明
cannAttrList	输出	用于保存运行环境支持的特性枚举数组。 用户无需提前申请内存，应用进程退出时，内存自动释放。
num	输出	用于保存支持的特性数量，与cannAttrList数组长度保持一致。

返回值说明

返回0表示成功，返回其他值表示失败。

1.4.14 aclGetCannAttribute

函数功能

查询运行环境是否支持指定的CANN特性。

函数原型

aclError **aclGetCannAttribute**(**aclCannAttr** cannAttr, int32_t *value)

参数说明

参数名	输入/输出	说明
cannAttr	输入	特性列表枚举值，一次查询可指定其中一项。
value	输出	是否支持： 1：支持 0：不支持

返回值说明

返回0表示成功，返回其他值表示失败。

1.4.15 aclGetDeviceCapability

函数功能

查询运行环境中对应Device的硬件规格大小。

函数原型

aclError **aclGetDeviceCapability**(uint32_t deviceId, **aclDeviceInfo** deviceInfo, int64_t *value)

参数说明

参数名	输入/输出	说明
deviceId	输入	Device ID。 用户调用 aclrtGetDeviceCount 接口获取可用的Device数量后，这个Device ID的取值范围：[0, (可用的Device数量-1)]
deviceInfo	输入	指定Device上的硬件规格类型。
value	输出	硬件规格值。

返回值说明

返回0表示成功，返回其他值表示失败。

1.5 运行时管理

1.5.1 Device 管理

1.5.1.1 aclrtSetDevice

函数功能

指定当前线程中用于运算的Device。

对于Atlas 200/300/500 推理产品，调用本接口会隐式创建默认Context，默认Context中包含2个Stream，1个默认Stream和1个执行内部同步的Stream。

对于Atlas 200/500 A2推理产品，调用本接口会隐式创建默认Context，该默认Context中包含1个默认Stream。

对于Atlas 训练系列产品，调用本接口会隐式创建默认Context，该默认Context中包含1个默认Stream。

对于Atlas A2训练系列产品/Atlas 800I A2推理产品，调用本接口会隐式创建默认Context，该默认Context中包含1个默认Stream。

对于Atlas 推理系列产品，调用本接口会隐式创建默认Context，在标准形态下，该默认Context中包含2个Stream，1个默认Stream和1个执行内部同步的Stream。

约束说明

- 调用[aclrtSetDevice](#)接口指定运算的Device后，若不使用Device上的资源时，可调用[aclrtResetDevice](#)或[aclrtResetDeviceForce](#)接口及时释放本进程使用的Device资源（若不调用这两个接口，功能上不会有问题，因为在进程退出时也会释放本进程使用的Device资源）：
 - 若调用[aclrtResetDevice](#)接口释放Device资源：
[aclrtResetDevice](#)接口内部涉及引用计数的实现，建议[aclrtResetDevice](#)接口与[aclrtSetDevice](#)接口配对使用，[aclrtSetDevice](#)接口每被调用一次，则引用计数加一。[aclrtResetDevice](#)接口每被调用一次，则该引用计数减一，当引用计数减到0时，才会真正释放Device上的资源。
 - 若调用[aclrtResetDeviceForce](#)接口释放Device资源：
[aclrtResetDeviceForce](#)接口可与[aclrtSetDevice](#)接口配对使用，也可不与[aclrtSetDevice](#)接口配对使用，若不配对使用，一个进程中，针对同一个Device，调用一次或多次[aclrtSetDevice](#)接口后，仅需调用一次本接口可释放Device上的资源。
- 在不同进程或线程中支持调用[aclrtSetDevice](#)接口指定同一个Device用于运算。在同一个进程中的多个线程中，如果调用[aclrtSetDevice](#)接口指定同一个Device用于运算，这时隐式创建的默认Context是同一个。
- 多Device场景下，可在进程中通过[aclrtSetDevice](#)接口切换到其它Device，也可以调用[aclrtSetCurrentContext](#)接口通过切换Context来切换Device。

函数原型

```
aclError aclrtSetDevice(int32_t deviceId)
```

参数说明

参数名	输入/输出	说明
deviceId	输入	Device ID。 用户调用 aclrtGetDeviceCount 接口获取可用的Device数量后，这个Device ID的取值范围：[0, (可用的Device数量-1)]

返回值说明

返回0表示成功，返回其他值表示失败。

参考资源

接口调用流程及示例，参见运行管理资源申请与释放。

1.5.1.2 aclrtResetDevice

函数功能

复位当前运算的Device，释放Device上的资源。释放的资源包括默认Context、默认Stream以及默认Context下创建的所有Stream。若默认Context或默认Stream下的任务还未完成，系统会等待任务完成后再释放。

aclrtResetDevice接口内部涉及引用计数的实现，建议aclrtResetDevice接口与aclrtSetDevice接口配对使用，aclrtSetDevice接口每被调用一次，则引用计数加一。aclrtResetDevice接口每被调用一次，则该引用计数减一，当引用计数减到0时，才会真正释放Device上的资源。

如果多次调用aclrtSetDevice接口而不调用aclrtResetDevice接口释放本线程使用的Device资源，功能上不会有问题，因为在进程退出时也会释放本进程使用的Device资源。

约束说明

若要复位的Device上存在显式创建的Context、Stream、Event，在复位前，建议遵循如下接口调用顺序，否则可能会导致业务异常。

接口调用顺序：调用[aclrtDestroyEvent](#)接口释放Event/调用[aclrtDestroyStream](#)接口释放显式创建的Stream-->调用[aclrtDestroyContext](#)释放显式创建的Context-->调用[aclrtResetDevice](#)接口

函数原型

```
aclError aclrtResetDevice(int32_t deviceId)
```

参数说明

参数名	输入/输出	说明
deviceId	输入	Device ID。

返回值说明

返回0表示成功，返回其他值表示失败。

参考资源

接口调用流程及示例，参见运行管理资源申请与释放。

1.5.1.3 aclrtResetDeviceForce

函数功能

复位当前运算的Device，释放Device上的资源。释放的资源包括默认Context、默认Stream以及默认Context下创建的所有Stream。若默认Context或默认Stream下的任务还未完成，系统会等待任务完成后再释放。

aclrtResetDeviceForce接口可与aclrtSetDevice接口配对使用，也可不与aclrtSetDevice接口配对使用，若不配对使用，一个进程中，针对同一个Device，调用一次或多次aclrtSetDevice接口后，仅需调用一次本接口可释放Device上的资源。

```
# 与aclrtSetDevice接口配对使用：
aclrtSetDevice(1) -> aclrtResetDeviceForce(1) -> aclrtSetDevice(1) -> aclrtResetDeviceForce(1)

# 与aclrtSetDevice接口不配对使用：
aclrtSetDevice(1) -> aclrtSetDevice(1) -> aclrtResetDeviceForce(1)
```

约束说明

- 多线程场景下，针对同一个Device，如果每个线程中都调用[aclrtSetDevice](#)接口、[aclrtResetDeviceForce](#)接口，如下所示，线程2中的[aclrtResetDeviceForce](#)接口会返回报错，因为线程1中[aclrtResetDeviceForce](#)接口已经释放了Device 1的资源：

```
时间线 ----->
线程1: aclrtSetDevice(1)      aclrtResetDeviceForce(1)
线程2: aclrtSetDevice(1)      aclrtResetDeviceForce(1)
```

多线程场景下，正确方式是应在线程执行的最后，调用一次[aclrtResetDeviceForce](#)释放Device资源，如下所示：

```
时间线 ----->
线程1: aclrtSetDevice(1)
线程2: aclrtSetDevice(1)      aclrtResetDeviceForce(1)
```

- [aclrtResetDevice](#)接口与[aclrtResetDeviceForce](#)接口可以混用，但混用时，若两个Reset接口的调用次数、调用顺序不对，接口会返回报错。
混用时的正确方式：
两个Reset接口都分别与Set接口配对使用，且[aclrtResetDeviceForce](#)接口在[aclrtResetDevice](#)接口之后
aclrtSetDevice(1) -> aclrtResetDevice(1) -> aclrtSetDevice(1) -> aclrtResetDeviceForce(1)
aclrtSetDevice(1) -> aclrtSetDevice(1) -> aclrtResetDevice(1) -> aclrtResetDeviceForce(1)
混用时的错误方式：

```
# aclrtResetDevice接口内部涉及引用计数的实现，当aclrtResetDevice接口每被调用一次，则该引用计数减1，当引用计数减到0时，会真正释放Device上的资源，此时再调用aclrtResetDevice或aclrtResetDeviceForce接口都会报错
aclrtSetDevice(1) -> aclrtSetDevice(1) -> aclrtResetDevice(1)-->aclrtResetDevice(1)-->aclrtResetDeviceForce(1)
aclrtSetDevice(1) -> aclrtSetDevice(1) -> aclrtResetDevice(1)-->aclrtResetDeviceForce(1)-->aclrtResetDeviceForce(1)
# aclrtResetDeviceForce接口在aclrtResetDevice接口之后，否则接口返回报错
aclrtSetDevice(1) -> aclrtSetDevice(1) -> aclrtResetDeviceForce(1)-->aclrtResetDevice(1)
```

函数原型

aclError **aclrtResetDeviceForce(int32_t deviceId)**

参数说明

参数名	输入/输出	说明
deviceId	输入	Device ID。

返回值说明

返回0表示成功，返回其他值表示失败。

1.5.1.4 aclrtGetDevice

函数功能

获取当前正在使用的Device的ID。

约束说明

如果没有提前指定Device，则调用aclrtGetDevice接口时，返回错误。指定Device的方式包括：在[aclInit](#)接口中开启默认Device、调用[aclrtSetDevice](#)接口显式指定Device、调用[aclrtCreateContext](#)接口隐式指定Device。

函数原型

aclError **aclrtGetDevice(int32_t *deviceId)**

参数说明

参数名	输入/输出	说明
deviceId	输出	Device ID的指针。

返回值说明

返回0表示成功，返回其他值表示失败。

1.5.1.5 aclrtGetRunMode

函数功能

获取当前AI软件栈的运行模式。

函数原型

```
aclError aclrtGetRunMode(aclrtRunMode *runMode)
```

参数说明

参数名	输入/输出	说明
runMode	输出	运行模式的指针。 - ACL_DEVICE: AI软件栈运行在Device的Control CPU或板端环境上。 Atlas 训练系列产品，不支持该选项。 Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该选项。 - ACL_HOST: AI软件栈运行在Host CPU上。

返回值说明

返回0表示成功，返回其他值表示失败。

参考资源

接口调用流程及示例代码，参见运行管理资源申请与释放。

1.5.1.6 aclrtSetTsDevice

函数功能

设置本次计算需要使用的Task Schedule。

函数原型

```
aclError aclrtSetTsDevice(aclrtTsId tsId)
```


参数说明

参数名	输入/输出	说明
tsId	输入	指定本次计算需要使用的Task Schedule。如果昇腾AI处理器中只有AI CORE Task Schedule，没有AI VECTOR Task Schedule，则设置该参数无效，默认使用AI CORE Task Schedule。 <pre>typedef enum aclrtTsId { ACL_TS_ID_AICORE = 0, //使用AICORE Task Schedule ACL_TS_ID_AIVECTOR = 1, //使用AIVECTOR Task Schedule ACL_TS_ID_RESERVED = 2, } aclrtTsId;</pre>

返回值说明

返回0表示成功，返回其他值表示失败。

1.5.1.7 aclrtGetDeviceCount

函数功能

获取可用Device的数量。

函数原型

aclError aclrtGetDeviceCount(uint32_t *count)

参数说明

参数名	输入/输出	说明
count	输出	Device数量的指针。

返回值说明

返回0表示成功，返回其他值表示失败。

1.5.1.8 aclrtGetDeviceUtilizationRate

函数功能

查询Device上Cube、Vector、AI CPU等的利用率。

约束说明

- 查询Device内存利用率为预留功能，当前版本不支持，若调用本接口查询内存利用率，查询到的利用率为-1。

- 昇腾虚拟化实例场景下，不支持调用本接口查询利用率，接口返回值无实际意义。
- 开启Profiling功能时，不支持调用本接口查询利用率，接口返回值无实际意义。
- Atlas 200/300/500 推理产品上没有Vector，调用本接口查询Vector利用率时，查询到的利用率为-1。
- Atlas 训练系列产品上没有Vector，调用本接口查询Vector利用率时，查询到的利用率为-1。

函数原型

aclError aclrtGetDeviceUtilizationRate(int32_t deviceId, **aclrtUtilizationInfo** *utilizationInfo)

参数说明

参数名	输入/输出	说明
deviceId	输入	Device ID。 用户调用 aclrtGetDeviceCount 接口获取可用的Device数量后，这个Device ID的取值范围：[0, (可用的Device数量-1)]
utilizationInfo	输出	利用率信息结构体指针。

返回值说明

返回0表示成功，返回其他值表示失败。

1.5.1.9 aclrtQueryDeviceStatus

函数功能

查询Device状态是正常可用、还是异常不可用。

函数原型

aclError aclrtQueryDeviceStatus(int32_t deviceId, **aclrtDeviceStatus** *deviceStatus)

参数说明

参数名	输入/输出	说明
deviceId	输入	Device ID。 用户调用 aclrtGetDeviceCount 接口获取可用的Device数量后，这个Device ID的取值范围：[0, (可用的Device数量-1)]

参数名	输入/输出	说明
deviceStatus	输出	Device状态。

返回值说明

返回0表示成功，返回其他值表示失败。

1.5.1.10 aclrtGetSocName

函数功能

查询当前运行环境的芯片版本。

函数原型

`const char *aclrtGetSocName()`

参数说明

无

返回值说明

返回芯片版本字符串的指针。如果通过该接口获取芯片版本失败，则返回空指针。

1.5.1.11 aclrtSetDeviceSatMode

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 200/500 A2推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

函数功能

设置当前Device的浮点计算结果输出模式。

约束说明

调用该接口成功后，后续在该Device上新创建的Stream按设置的模式生效，对之前已创建的Stream不生效。

函数原型

`aclError aclrtSetDeviceSatMode(aclrtFloatOverflowMode mode)`

参数说明

参数名	输入/输出	说明
mode	输入	设置浮点计算结果输出模式。

返回值说明

返回0表示成功，返回其他值表示失败。

1.5.1.12 aclrtGetDeviceSatMode

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 200/500 A2推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

函数功能

查询当前Device的浮点计算结果输出模式。

函数原型

aclError aclrtGetDeviceSatMode(**aclrtFloatOverflowMode** *mode)

参数说明

参数名	输入/输出	说明
mode	输出	获取浮点计算结果输出模式。

返回值说明

返回0表示成功，返回其他值表示失败。

1.5.1.13 aclrtDeviceCanAccessPeer

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 200/500 A2推理产品，不支持该接口。

函数功能

查询Device之间是否支持内存复制。

约束说明

- 仅支持物理机和容器场景；
- 仅支持同一个PCIe Switch内Device之间的内存复制。AI Server场景下，虽然是跨PCIe Switch，但也支持Device之间的内存复制。
- 仅支持同一个物理机或容器内的Device之间的内存复制操作；
- 仅支持同一个进程内、线程间的Device之间的内存复制，不支持不同进程间Device之间的内存复制。
- Host的相关端口要已使能（例如Host BIOS等），同一个PCIe Switch内Device之间的拷贝才能端到端可用。

函数原型

```
aclError aclrtDeviceCanAccessPeer(int32_t *canAccessPeer, int32_t deviceId,
int32_t peerDeviceId)
```

参数说明

参数名	输入/输出	说明
canAccessPeer	输出	通过deviceId参数指定的Device和通过peerDeviceId参数指定的Device之间是否支持内存复制，1表示支持，0表示不支持。
deviceId	输入	指定Device的ID，不能与peerDeviceId参数值相同。 用户调用 aclrtGetDeviceCount 接口获取可用的Device数量后，这个Device ID的取值范围：[0, (可用的Device数量-1)]
peerDeviceId	输入	指定Device的ID，不能与deviceId参数值相同。 用户调用 aclrtGetDeviceCount 接口获取可用的Device数量后，这个Device ID的取值范围：[0, (可用的Device数量-1)]

返回值说明

返回0表示成功，返回其他值表示失败。

参考资源

接口调用示例，参见两个Device间的数据传输。

1.5.1.14 aclrtDeviceEnablePeerAccess

Atlas 200/300/500 推理产品，不支持该接口。
Atlas 200/500 A2推理产品，不支持该接口。

函数功能

使能当前Device与指定Device之间的内存复制。使能内存复制是Device级的。

约束说明

可提前调用[aclrtDeviceCanAccessPeer](#)接口显查询当前Device与指定Device之间能否进行内存复制。使能Device间的内存复制功能后，若想关闭该功能，可调用[aclrtDeviceDisablePeerAccess](#)接口。

函数原型

aclError [aclrtDeviceEnablePeerAccess](#)(int32_t peerDeviceId, uint32_t flags)

参数说明

参数名	输入/输出	说明
peerDeviceId	输入	Device ID，该ID不能与当前Device的ID相同。 用户调用 aclrtGetDeviceCount 接口获取可用的Device数量后，这个Device ID的取值范围：[0, (可用的Device数量-1)]
flags	输入	保留参数，当前必须设置为0。

返回值说明

返回0表示成功，返回其他值表示失败。

参考资源

接口调用示例，参见两个Device间的数据传输。

1.5.1.15 [aclrtDeviceDisablePeerAccess](#)

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 200/500 A2推理产品，不支持该接口。

函数功能

关闭当前Device与指定Device之间的内存复制功能。关闭内存复制功能是Device级的。

约束说明

调用[aclrtDeviceEnablePeerAccess](#)接口使能当前Device与指定Device之间的内存复制后，可调用[aclrtDeviceDisablePeerAccess](#)接口关闭内存复制功能。

Atlas 推理系列产品，Control CPU开放形态下，应用程序运行在Device的Control CPU上时，该接口不支持Device之间的内存复制。

函数原型

aclError [aclrtDeviceDisablePeerAccess](#)(int32_t peerDeviceId)

参数说明

参数名	输入/输出	说明
peerDeviceId	输入	Device ID，该ID不能与当前Device的ID相同。 用户调用 aclrtGetDeviceCount 接口获取可用的Device数量后，这个Device ID的取值范围：[0, (可用的Device数量-1)]

返回值说明

返回0表示成功，返回其他值表示失败。

参考资源

接口调用示例，参见两个Device间的数据传输。

1.5.1.16 aclrtGetOverflowStatus

Atlas 200/300/500 推理产品，不支持该接口。

函数功能

获取当前Device下所有Stream上任务的溢出状态，并将状态值拷贝到用户申请的Device内存中。异步接口。

约束说明

- 该接口是异步接口，调用接口成功仅表示任务下发成功，不表示任务执行成功。调用该接口后，需调用同步等待接口（例如，[aclrtSynchronizeStream](#)）确保任务已执行完成，否则可能会导致训练或推理等业务异常、Device断链掉卡等未知情况。
- Atlas A2训练系列产品/Atlas 800I A2推理产品上，调用本接口查询出来的溢出状态是进程级别的。

函数原型

[aclError](#) [aclrtGetOverflowStatus](#)(void *outputAddr, size_t outputSize, [aclrtStream](#) stream)

参数说明

参数名	输入/输出	说明
outputAddr	输入&输出	用户申请的Device内存，需通过 aclrtMalloc 接口申请。 如果需要在Host侧查看数据，可调用 aclrtMemcpy 或 aclrtMemcpyAsync 接口，将Device侧的数据传输到Host侧。

参数名	输入/输出	说明
outputSize	输入	需申请的Device内存大小，单位Byte，固定大小为64Byte。
stream	输入	指定Stream，用于下发溢出状态查询任务。

返回值说明

返回0表示成功，返回其他值表示失败。

1.5.1.17 aclrtResetOverflowStatus

Atlas 200/300/500 推理产品，不支持该接口。

函数功能

清除当前Device下所有Stream上任务的溢出状态。异步接口。

约束说明

- 该接口是异步接口，调用接口成功仅表示任务下发成功，不表示任务执行成功。调用该接口后，需调用同步等待接口（例如，[aclrtSynchronizeStream](#)）确保任务已执行完成，否则可能会导致训练或推理等业务异常、Device断链掉卡等未知情况。
- Atlas A2训练系列产品/Atlas 800I A2推理产品上，调用本接口清除的溢出状态是进程级别的。

函数原型

aclError [aclrtResetOverflowStatus](#)([aclrtStream](#) stream)

参数说明

参数名	输入/输出	说明
stream	输入	指定Stream，用于下发溢出状态复位任务。

返回值说明

返回0表示成功，返回其他值表示失败。

1.5.1.18 aclrtSynchronizeDevice

函数功能

阻塞应用程序运行，直到正在运算中的Device完成运算。

多Device场景下，调用该接口等待的是当前Context对应的Device。

函数原型

```
aclError aclrtSynchronizeDevice(void)
```

参数说明

无

返回值说明

返回0表示成功，返回其他值表示失败。

参考资源

- 接口调用流程，参见多Device切换。
- 接口调用示例，参见Device的同步等待示例代码。

1.5.1.19 aclrtSynchronizeDeviceWithTimeout

函数功能

阻塞应用程序运行，直到正在运算中的Device完成运算。该接口是在[aclrtSynchronizeDevice](#)接口基础上进行了增强，支持用户设置超时时间，当应用程序异常时可根据所设置的超时时间自行退出，超时退出时本接口返回ACL_ERROR_RT_STREAM_SYNC_TIMEOUT。

多Device场景下，调用该接口等待的是当前Context对应的Device。

函数原型

```
aclError aclrtSynchronizeDeviceWithTimeout(int32_t timeout)
```

参数说明

参数名	输入/输出	说明
timeout	输入	接口的超时时间。 取值说明如下： • -1：表示永久等待，和接口aclrtSynchronizeDevice功能一样； • >0：配置具体的超时时间，单位是毫秒。

返回值说明

返回0表示成功，返回其他值表示失败。

1.5.2 Context 管理

1.5.2.1 aclrtCreateContext

函数功能

在当前进程或线程中显式创建一个Context。

对于Atlas 200/300/500 推理产品，该Context中包含2个Stream，1个默认Stream和1个执行内部同步的Stream。

对于Atlas 推理系列产品，在标准形态下，该Context中包含2个Stream，1个默认Stream和1个执行内部同步的Stream。

对于Atlas 200/500 A2推理产品，该Context中包含2个Stream，1个默认Stream和1个执行内部同步的Stream。

对于Atlas 训练系列产品，该Context中包含1个默认Stream。

对于Atlas A2训练系列产品/Atlas 800I A2推理产品，该Context中包含1个默认Stream。

说明

如果在应用程序中没有调用[aclrtSetDevice](#)接口，那么在首次调用[aclrtCreateContext](#)接口时，系统内部会根据该接口传入的Device ID，为该Device绑定一个默认Stream（一个Device仅绑定一个默认Stream），因此在首次调用[aclrtCreateContext](#)接口时，占用的Stream数量 = Device上绑定的默认Stream + Context中包含的Stream。

约束说明

支持以下使用场景：

- 若不调用[aclrtCreateContext](#)接口显式创建Context，那系统会使用默认Context，该默认Context是在调用[aclrtSetDevice](#)接口时隐式创建的。
 - 隐式创建Context：适合简单、无复杂交互逻辑的应用，但缺点在于，在多线程编程中，执行结果取决于线程调度的顺序。
 - 显式创建Context：适合大型、复杂交互逻辑的应用，且便于提高程序的可读性、可维护性。
- 在某一进程中指定Device，该进程内的多个线程可共用在此Device上显式创建的Context（调用[aclrtCreateContext](#)接口显式创建Context）。
- 若在某一进程内创建多个Context（Context的数量与Stream相关，Stream数量有限制，请参见[aclrtCreateStream](#)），当前线程在同一时刻内只能使用其中一个Context，建议通过[aclrtSetCurrentContext](#)接口明确指定当前线程的Context，增加程序的可维护性。

函数原型

aclError [aclrtCreateContext](#)([aclrtContext](#) *context, int32_t deviceId)

参数说明

参数名	输入/输出	说明
context	输出	Context的指针。
deviceId	输入	在指定的Device下创建Context。 用户调用 aclrtGetDeviceCount 接口获取可用的Device数量后，这个Device ID的取值范围：[0, (可用的Device数量-1)]

返回值说明

返回0表示成功，返回其他值表示失败。

1.5.2.2 aclrtDestroyContext

函数功能

销毁一个Context，释放Context的资源。只能销毁通过[aclrtCreateContext](#)接口创建的Context。

函数原型

[aclError](#) [aclrtDestroyContext](#)([aclrtContext](#) context)

参数说明

参数名	输入/输出	说明
context	输入	需销毁的Context。

返回值说明

返回0表示成功，返回其他值表示失败。

1.5.2.3 aclrtSetCurrentContext

函数功能

设置线程的Context。

约束说明

- 支持以下场景：
 - 如果在某线程（例如：thread1）中调用[aclrtCreateContext](#)接口显式创建一个Context（例如：ctx1），则可以不调用[aclrtSetCurrentContext](#)接口指定该线程的Context，系统默认将ctx1作为thread1的Context。

- 如果没有调用[aclrtCreateContext](#)接口显式创建Context，则系统将默认Context作为线程的Context，此时，不能通过[aclrtDestroyContext](#)接口来释放默认Context。
- 如果多次调用[aclrtSetCurrentContext](#)接口设置线程的Context，以最后一次为准。
- 若给线程设置的Context所对应的Device已经被复位，则不能将该Context设置为线程的Context，否则会导致业务异常。
- 推荐在某一线程中创建的Context，在该线程中使用。若在线程A中调用[aclrtCreateContext](#)接口创建Context，在线程B中使用该Context，则需由用户自行保证两个线程中同一个Context下同一个Stream中任务执行的顺序。

函数原型

aclError [aclrtSetCurrentContext](#)([aclrtContext](#) context)

参数说明

参数名	输入/输出	说明
context	输入	指定线程当前的Context。

返回值说明

返回0表示成功，返回其他值表示失败。

1.5.2.4 [aclrtGetCurrentContext](#)

函数功能

获取线程的Context。

约束说明

如果用户多次调用[aclrtSetCurrentContext](#)接口设置当前线程的Context，则获取的是最后一次设置的Context。

函数原型

aclError [aclrtGetCurrentContext](#)([aclrtContext](#) *context)

参数说明

参数名	输入/输出	说明
context	输出	线程当前Context的指针。

返回值说明

返回0表示成功，返回其他值表示失败。

1.5.2.5 aclrtCtxSetSysParamOpt

函数功能

设置当前Context中的系统参数值。

约束说明

多次调用本接口，以最后一次设置的值为准。

函数原型

```
aclError aclrtCtxSetSysParamOpt(aclSysParamOpt opt, int64_t value)
```

参数说明

参数名	输入/输出	说明
opt	输入	系统参数。
value	输入	系统参数值。

返回值说明

返回0表示成功，返回其他值表示失败。

1.5.2.6 aclrtCtxGetSysParamOpt

函数功能

获取当前Context中的系统参数值。

约束说明

系统参数无默认值，如果不调用[aclrtCtxSetSysParamOpt](#)接口设置系统参数的值，直接调用本接口获取系统参数的值，接口会返回失败。

函数原型

```
aclError aclrtCtxGetSysParamOpt(aclSysParamOpt opt, int64_t *value)
```

参数说明

参数名	输入/输出	说明
opt	输入	系统参数。
value	输出	存放系统参数值的内存的指针。

返回值说明

返回0表示成功，返回其他值表示失败。

1.5.3 Stream 管理

1.5.3.1 aclrtCreateStream

函数功能

创建一个Stream。

约束说明

- 每个Context对应一个默认Stream，该默认Stream是调用[aclrtSetDevice](#)接口或[aclrtCreateContext](#)接口隐式创建的。推荐调用[aclrtCreateStream](#)接口显式创建Stream。
 - 隐式创建Stream：适合简单、无复杂交互逻辑的应用，但缺点在于，在多线程编程中，执行结果取决于线程调度的顺序。
 - 显式创建Stream：**推荐显式**，适合大型、复杂交互逻辑的应用，且便于提高程序的可读性、可维护性。
- 对于Atlas 200/300/500 推理产品，硬件资源最多支持1024个Stream，如果已存在多个默认Stream，只能显式创建N个Stream（N=1024-默认Stream个数-执行内部同步的Stream个数），例如，若已存在一个默认Stream和一个执行内部同步的Stream，则只能显式创建1022个Stream。
- 对于Atlas 200/500 A2推理产品，硬件资源最多支持512个Stream，如果已存在多个默认Stream，只能显式创建N个Stream（N=512-默认Stream个数-执行内部同步的Stream个数），例如，若已存在一个默认Stream和一个执行内部同步的Stream，则只能显式创建510个Stream。
- 对于Atlas 推理系列产品，硬件资源最多支持1024个Stream，
 - 标准形态下，如果已存在多个默认Stream，只能显式创建N个Stream（N=1024-默认Stream个数-执行内部同步的Stream个数），例如，若已存在一个默认Stream和一个执行内部同步的Stream，则只能显式创建1022个Stream。
- 对于Atlas 训练系列产品，硬件资源最多支持2048个Stream，如果已存在多个默认Stream，只能显式创建N个Stream（N=2048-默认Stream个数-执行内部同步的Stream个数），例如，若已存在一个默认Stream和一个执行内部同步的Stream，则只能显式创建2046个Stream。
多进程场景下，若一次性创建的Stream数量总和接近2048，可能会出现创建Stream失败的情况，此时，建议：（1）清理冗余Stream，减少不必要的

Stream；（2）调整代码逻辑，分批创建Stream，例如第一批创建部分Stream，然后第二批再创建部分Stream，以此类推，直到Stream总数接近2048。

- 对于Atlas A2训练系列产品/Atlas 800I A2推理产品，硬件资源最多支持2048个Stream，如果已存在多个默认Stream，只能显式创建N个Stream（N=2048-默认Stream个数-执行内部同步的Stream个数），例如，若已存在一个默认Stream和一个执行内部同步的Stream，则只能显式创建2046个Stream。

函数原型

```
aclError aclrtCreateStream(aclrtStream *stream)
```

参数说明

参数名	输入/输出	说明
stream	输出	Stream的指针。

返回值说明

返回0表示成功，返回其他值表示失败。

参考资源

接口调用流程及示例，参见运行管理资源申请与释放。

1.5.3.2 aclrtCreateStreamV2

- Atlas 200/300/500 推理产品不支持该接口。
- Atlas 200/500 A2推理产品不支持该接口。
- Atlas 训练系列产品不支持该接口。
- Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。
- Atlas 推理系列产品，不支持该接口。

函数功能

创建一个Stream，支持创建Stream时增加Stream配置。

约束说明

- 本接口需要配合其它接口一起使用，创建Stream，接口调用顺序如下：
 - a. 调用[aclrtCreateStreamConfigHandle](#)接口创建Stream配置对象。
 - b. 多次调用[aclrtSetStreamConfigOpt](#)接口设置配置对象中每个属性的值。
 - c. 调用[aclrtCreateStreamV2](#)接口创建Stream。
 - d. Stream使用完成后，调用[aclrtDestroyStreamConfigHandle](#)接口销毁Stream配置对象，调用[aclrtDestroyStream](#)接口销毁Stream。

函数原型

aclError **aclrtCreateStreamV2**(**aclrtStream** *stream, const **aclrtStreamConfigHandle** *handle)

参数说明

参数名	输入/输出	说明
stream	输出	Stream的指针。
handle	输入	Stream配置对象的指针。与 aclrtSetStreamConfigOpt 中的handle保持一致。

返回值说明

返回0表示成功，返回其他值表示失败。

1.5.3.3 aclrtSetStreamConfigOpt

Atlas 200/300/500 推理产品不支持该接口。

Atlas 200/500 A2推理产品不支持该接口。

Atlas 训练系列产品不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

函数功能

设置Stream配置对象中的各属性的取值。

约束说明

本接口需要配合其它接口一起使用，创建Stream，接口调用顺序如下：

1. 调用**aclrtCreateStreamConfigHandle**接口创建Stream配置对象。
2. 多次调用**aclrtSetStreamConfigOpt**接口设置配置对象中每个属性的值。
3. 调用**aclrtCreateStreamV2**接口创建Stream。
4. Stream使用完成后，调用**aclrtDestroyStreamConfigHandle**接口销毁Stream配置对象，调用**aclrtDestroyStream**接口销毁Stream。

函数原型

aclError **aclrtSetStreamConfigOpt**(**aclrtStreamConfigHandle** *handle, **aclrtStreamConfigAttr** attr, const void *attrValue, size_t valueSize)

参数说明

参数名	输入/输出	说明
handle	输出	Stream配置对象的指针。需提前调用 aclrtCreateStreamConfigHandle 接口创建该对象。
attr	输入	指定需设置的属性。
attrValue	输入	指向属性值的指针，attr对应的属性取值。 如果属性值本身是指针，则传入该指针的地址。
valueSize	输入	attrValue部分的数据长度。 用户可使用C/C++标准库的函数sizeof(*attrValue)查询数据长度。

返回值说明

返回0表示成功，返回其他值表示失败。

1.5.3.4 aclrtCreateStreamWithConfig

函数功能

在当前进程或线程中创建一个Stream。

在Atlas 200/300/500 推理产品上，使用本接口与[aclrtCreateStream](#)接口是等价的。

在Atlas 推理系列产品上，相比[aclrtCreateStream](#)接口，使用本接口可以创建一个快速下发任务的Stream，但会增加内存消耗或CPU的性能消耗。

在Atlas 训练系列产品上，相比[aclrtCreateStream](#)接口，使用本接口可以创建一个快速下发任务的Stream，但会增加内存消耗或CPU的性能消耗。

Atlas A2训练系列产品/Atlas 800I A2推理产品，相比[aclrtCreateStream](#)接口，使用本接口可以创建一个快速下发任务的Stream，但会增加内存消耗或CPU的性能消耗。

Atlas 200/500 A2推理产品，使用本接口与[aclrtCreateStream](#)接口是等价的。

函数原型

aclError [aclrtCreateStreamWithConfig](#)([aclrtStream](#) *stream, uint32_t priority, uint32_t flag)

参数说明

参数名	输入/输出	说明
stream	输出	Stream的指针。

参数名	输入/输出	说明
priority	输入	优先级。 Atlas 200/300/500 推理产品上，当前固定设置为0，预留参数，暂不使用。 Atlas 训练系列产品上，当前固定设置为0，预留参数，暂不使用。 Atlas A2训练系列产品/Atlas 800I A2推理产品上，当前固定设置为0，预留参数，暂不使用。 Atlas 200/500 A2推理产品上，当前固定设置为0，预留参数，暂不使用。 Atlas 推理系列产品上，该参数取值范围：[0, 7]，总共最多支持8个优先级，数字越小代表优先级越高，其中，0的优先级最高，7的优先级最低。配置取值范围以外的值，本接口返回报错。
flag	输入	Stream指针的flag。 取值范围： ● ACL_STREAM_FAST_LAUNCH：使用该模式创建出来的Stream，在使用Stream时，下发任务的速度更快。相比aclrtCreateStream接口创建出来的Stream，在使用Stream时才会申请系统内部资源，导致下发任务的时长增加，使用本接口的ACL_STREAM_FAST_LAUNCH模式创建Stream时，会在创建Stream时预申请系统内部资源，因此创建Stream的时长增加，下发任务的时长缩短，总体来说，创建一次Stream，使用多次的场景下，总时长缩短，但创建Stream时预申请内部资源会增加内存消耗。 ● ACL_STREAM_FAST_SYNC：使用该模式创建出来的Stream，在调用aclrtSynchronizeStream接口时，会阻塞当前线程，主动查询任务的执行状态，一旦任务完成，立即返回。相比aclrtCreateStream接口创建出来的Stream，在调用aclrtSynchronizeStream接口时，会一直被动等待Device上任务执行完成的通知，等待时间长，使用本接口的ACL_STREAM_FAST_SYNC模式创建的Stream，没有被动等待，总时长缩短，但主动查询的操作会增加CPU的性能消耗。 说明 配置取值范围之外的值，本接口创建出来的Stream等同于aclrtCreateStream接口。

返回值说明

返回0表示成功，返回其他值表示失败。

1.5.3.5 aclrtDestroyStream

函数功能

销毁指定Stream，销毁通过[aclrtCreateStream](#)或[aclrtCreateStreamWithConfig](#)或[aclrtCreateStreamV2](#)接口创建的Stream，若Stream上有未完成的任务，会等待任务完成后再销毁Stream。

约束说明

- 在调用[aclrtDestroyStream](#)接口销毁指定Stream前，需要先调用[aclrtSynchronizeStream](#)接口确保Stream中的任务都已完成。
- 调用[aclrtDestroyStream](#)接口销毁指定Stream时，需确保该Stream在当前Context下。
- 在调用[aclrtDestroyStream](#)接口销毁指定Stream时，需确保其它接口没有正在使用该Stream。

函数原型

[aclError](#) [aclrtDestroyStream](#)([aclrtStream](#) stream)

参数说明

参数名	输入/输出	说明
stream	输入	待销毁的Stream。

返回值说明

返回0表示成功，返回其他值表示失败。

参考资源

接口调用流程及示例，参见运行管理资源申请与释放。

1.5.3.6 aclrtDestroyStreamForce

函数功能

销毁指定Stream，销毁通过[aclrtCreateStream](#)或[aclrtCreateStreamWithConfig](#)接口创建的Stream，若Stream上有未完成的任务，不会等待任务完成，直接强制销毁Stream。

在Atlas 200/300/500 推理产品上，使用本接口与[aclrtDestroyStream](#)接口是等价的。

约束说明

调用本接口销毁指定Stream时，需确保该Stream在当前Context下。

函数原型

aclError aclrtDestroyStreamForce(**aclrtStream** stream)

参数说明

参数名	输入/输出	说明
stream	输入	待销毁的Stream。

返回值说明

返回0表示成功，返回其他值表示失败。

1.5.3.7 aclrtSetStreamOverflowSwitch

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 200/500 A2推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

函数功能

饱和模式下，对接上层训练框架时（例如PyTorch），针对指定Stream，打开或关闭溢出检测开关，关闭后无法通过溢出检测算子获取任务是否溢出。

约束说明

- 在调用本接口前，可调用**aclrtSetDeviceSatMode**接口设置饱和模式。
- 调用该接口打开或关闭溢出检测开关后，仅对后续新下的任务生效，已下发的任务仍维持原样。

函数原型

aclError aclrtSetStreamOverflowSwitch(**aclrtStream** stream, uint32_t flag)

参数说明

参数名	输入/输出	说明
stream	输入	待操作Stream。 若传入NULL，则操作默认Stream。
flag	输入	溢出检测开关，取值范围如下： 0：关闭 1：打开

返回值说明

返回0表示成功，返回其他值表示失败。

1.5.3.8 aclrtGetStreamOverflowSwitch

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 200/500 A2推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

函数功能

针对指定Stream，获取其当前溢出检测开关是否打开。

函数原型

```
aclError aclrtGetStreamOverflowSwitch(aclrtStream stream, uint32_t *flag)
```

参数说明

参数名	输入/输出	说明
stream	输入	待操作Stream，若传入NULL，则操作默认Stream。
flag	输出	溢出检测开关，取值范围如下： 0：关闭 1：打开

返回值说明

返回0表示成功，返回其他值表示失败。

1.5.3.9 aclrtSetStreamFailureMode

Atlas 200/300/500 推理产品，不支持该接口。

函数功能

当一个Stream上下发了多个任务时，可通过本接口指定任务调度模式，以便控制某个任务失败后是否继续执行下一个任务。

约束说明

针对指定Stream只能调用一次本接口设置任务调度模式。

函数原型

```
aclError aclrtSetStreamFailureMode(aclrtStream stream, uint64_t mode)
```

参数说明

参数名	输入/输出	说明
stream	输入	待操作Stream。 不支持指定默认Stream（即该参数传入NULL）的任务调度模式。
mode	输入	当一个Stream上下发了多个任务时，可通过本参数指定任务调度模式，以便控制某个任务失败后是否继续执行下一个任务。 取值范围如下： - ACL_CONTINUE_ON_FAILURE：默认值，某个任务失败后，继续执行下一个任务； - ACL_STOP_ON_FAILURE：某个任务失败后，停止执行后续的任务。

返回值说明

返回0表示成功，返回其他值表示失败。

1.5.3.10 aclrtStreamQuery

函数功能

查询指定Stream上的所有任务的执行状态。

约束说明

- 本接口支持在以下场景使用：
- Atlas 200/300/500 推理产品上支持使用该接口查询任务的执行状态
 - 昇腾虚拟化实例（Ascend Virtual Instance）场景下支持使用该接口查询任务的执行状态

函数原型

```
aclError aclrtStreamQuery(aclrtStream stream, aclrtStreamStatus *status)
```

参数说明

参数名	输入/输出	说明
stream	输入	Stream的指针。
status	输出	Stream上的任务状态。

返回值说明

返回0表示成功，返回其他值表示失败。

1.5.3.11 aclrtSynchronizeStream

函数功能

阻塞应用程序运行，直到指定Stream中的所有任务都完成。

函数原型

aclError aclrtSynchronizeStream(**aclrtStream** stream)

参数说明

参数名	输入/输出	说明
stream	输入	指定需要完成所有任务的Stream。

返回值说明

返回0表示成功，返回其他值表示失败。

参考资源

接口调用示例，参见Stream内任务的同步等待示例代码。

1.5.3.12 aclrtSynchronizeStreamWithTimeout

函数功能

阻塞应用程序运行，直到指定Stream中的所有任务都完成，该接口是在[aclrtSynchronizeStream](#)接口基础上进行了增强，支持用户设置超时时间，当应用程序异常时可根据所设置的超时时间自行退出。

函数原型

```
aclError aclrtSynchronizeStreamWithTimeout(aclrtStream stream, int32_t timeout)
```

参数说明

参数名	输入/输出	说明
stream	输入	指定需要完成所有任务的Stream。
timeout	输入	接口的超时时间。 取值说明如下： • -1：表示永久等待，和接口aclrtSynchronizeStream功能一样； • >0：配置具体的超时时间，单位是毫秒。

返回值说明

返回0表示成功，返回其他值表示失败。

1.5.4 Event 管理

1.5.4.1 aclrtCreateEvent

函数功能

创建一个Event，创建出来的Event可用于统计两个Event之间的耗时、多Stream场景下的同步等待等场景。

约束说明

调用本接口创建Event后，后续调用[aclrtRecordEvent](#)接口时，系统内部才会申请Event资源，因此会受Event数量的限制，Event达到上限后，系统内部会等待资源释放。

Event数量限制如下：

Atlas 200/300/500 推理产品，单个Device上最多支持1023个Event。

Atlas 推理系列产品，单个Device上最多支持1023个Event。

Atlas 200/500 A2推理产品，单个Device上最多支持65535个Event。

Atlas 训练系列产品，单个Device上最多支持65535个Event。

Atlas A2训练系列产品/Atlas 800I A2推理产品，单个Device上最多支持65535个Event。

函数原型

```
aclError aclrtCreateEvent(aclrtEvent *event)
```

参数说明

参数名	输入/输出	说明
event	输出	Event的指针。

返回值说明

返回0表示成功，返回其他值表示失败。

相关接口

您也可以使用[aclrtCreateEventWithFlag](#)或[aclrtCreateEventExWithFlag](#)接口创建一个带标识的Event，不同标识的Event用于不同的功能。

参考资料

接口调用流程及调用示例，参见Event的同步等待示例代码、Stream间任务的同步等待接口调用流程及示例代码。

1.5.4.2 aclrtCreateEventWithFlag

函数功能

创建一个带flag的Event，不同flag的Event用于不同的功能。支持创建Event时携带多个flag（按位进行或操作），从而同时使能对应flag的功能。

约束说明

调用本接口创建Event时，flag为bitmap，支持将flag设置为单个宏、或者对多个宏进行或操作：

- 若flag参数值**包含**ACL_EVENT_SYNC宏，则创建出来的Event数量受限，具体如下：
Atlas 200/300/500 推理产品，单个Device上最多支持1023个Event。
Atlas 推理系列产品，单个Device上最多支持1023个Event。
Atlas 200/500 A2推理产品，单个Device上最多支持65535个Event。
Atlas 训练系列产品，单个Device上最多支持65535个Event。
Atlas A2训练系列产品/Atlas 800I A2推理产品，单个Device上最多支持65535个Event。
- 若flag参数值**不包含**ACL_EVENT_SYNC宏，则不支持在以下API中使用本接口创建的Event：[aclrtResetEvent](#)、[aclrtStreamWaitEvent](#)、[aclrtQueryEventWaitStatus](#)。

函数原型

```
aclError aclrtCreateEventWithFlag(aclrtEvent *event, uint32_t flag)
```

参数说明

参数名	输入/输出	说明
event	输出	Event的指针。
flag	输入	Event指针的flag。 当前支持将flag设置为如下宏： - ACL_EVENT_TIME_LINE 使能该bit表示创建的Event需要记录时间戳信息。注意：使能时间戳功能会影响Event相关接口的性能。 #define ACL_EVENT_TIME_LINE 0x00000008u - ACL_EVENT_SYNC 使能该bit表示创建的Event支持多Stream间的同步。 #define ACL_EVENT_SYNC 0x00000001u - ACL_EVENT_CAPTURE_STREAM_PROGRESS 使能该bit表示创建的Event用于跟踪stream的任务执行进度。 #define ACL_EVENT_CAPTURE_STREAM_PROGRESS 0x00000002u

返回值说明

返回0表示成功，返回其他值表示失败。

1.5.4.3 aclrtCreateEventExWithFlag

函数功能

创建一个带flag的Event，不同flag的Event用于不同的功能。支持创建Event时携带多个flag（按位进行或操作），从而同时使能对应flag的功能。创建Event时，Event资源不受硬件限制。

约束说明

- 采用本API创建的Event，不支持在以下接口中使用：[aclrtResetEvent](#)、[aclrtQueryEvent](#)、[aclrtQueryEventWaitStatus](#)。
- 调用本接口创建Event时，flag为bitmap，支持将flag设置为单个宏、或者对多个宏进行或操作。
 - 若flag参数值包含ACL_EVENT_SYNC宏，后续调用[aclrtRecordEvent](#)接口时，系统内部才会申请Event资源，因此会受Event数量的限制，Event达到上限后，系统内部会等待资源释放。
Event数量限制如下：
Atlas 200/300/500 推理产品，单个Device上最多支持1023个Event。
Atlas 推理系列产品，单个Device上最多支持1023个Event。

Atlas 200/500 A2推理产品，单个Device上最多支持65535个Event。

Atlas 训练系列产品，单个Device上最多支持65535个Event。

Atlas A2训练系列产品/Atlas 800I A2推理产品，单个Device上最多支持65535个Event。

- 若flag参数值**不包含**ACL_EVENT_SYNC宏，则不支持在[aclrtStreamWaitEvent](#)接口中使用本接口创建的Event。

函数原型

```
aclError aclrtCreateEventExWithFlag(aclrtEvent *event, uint32_t flag)
```

参数说明

参数名	输入/输出	说明
event	输出	Event的指针。
flag	输入	Event指针的flag。 当前支持将flag设置为如下宏： • ACL_EVENT_TIME_LINE 使能该bit表示创建的Event需要记录时间戳信息。注意：使能时间戳功能会影响Event相关接口的性能。 #define ACL_EVENT_TIME_LINE 0x00000008u • ACL_EVENT_SYNC 使能该bit表示创建的Event支持多Stream间的同步。 #define ACL_EVENT_SYNC 0x00000001u • ACL_EVENT_CAPTURE_STREAM_PROGRESS 使能该bit表示创建的Event用于跟踪stream的任务执行进度。 #define ACL_EVENT_CAPTURE_STREAM_PROGRESS 0x00000002u

返回值说明

返回0表示成功，返回其他值表示失败。

1.5.4.4 aclrtDestroyEvent

函数功能

销毁一个Event，支持在Event未完成前调用本接口销毁Event。此时，本接口不会阻塞线程等Event完成，Event相关资源会在Event完成时被自动释放。

约束说明

在调用aclrtDestroyEvent接口销毁指定Event时，需确保其它接口没有正在使用该Event。

函数原型

aclError **aclrtDestroyEvent(aclrtEvent event)**

参数说明

参数名	输入/输出	说明
event	输入	待销毁的Event。

返回值说明

返回0表示成功，返回其他值表示失败。

参考资源

接口调用流程及调用示例，参见Event的同步等待示例代码、Stream间任务的同步等待接口调用流程及示例代码。

1.5.4.5 aclrtRecordEvent

函数功能

在Stream中记录一个Event。本接口被调用时，AscendCL会捕获当前Stream上已下发的任务，并记录到Event事件中，因此后续若调用[aclrtQueryEventStatus](#)或[aclrtStreamWaitEvent](#)接口时，会检查或等待该Event事件中所捕获的任务都已经完成。

对于使用[aclrtCreateEventExWithFlag](#)创建的Event：

- 本接口支持对同一个Event多次record实现Event复用，每次Record会重新捕获当前Stream上已下发的任务，并覆盖保存到Event中。在调用[aclrtStreamWaitEvent](#)接口时，会使用最近一次Event中所保存的任务，且不会被后续的[aclrtRecordEvent](#)调用影响。
- 在首次调用本接口前，由于Event中没有任务，因此调用[aclrtQueryEventStatus](#)接口时会返回ACL_EVENT_RECORDED_STATUS_COMPLETE。

约束说明

- 该接口是异步接口，调用接口成功仅表示任务下发成功，不表示任务执行成功。调用该接口后，需调用同步等待接口（例如，[aclrtSynchronizeStream](#)）确保任务已执行完成，否则可能会导致训练或推理等业务异常、Device断链掉卡等未知情况。
- [aclrtRecordEvent](#)接口与[aclrtStreamWaitEvent](#)接口配合使用时，主要用于多Stream之间同步的场景，在调用[aclrtRecordEvent](#)接口时，系统内部会申请Event资源，在调用[aclrtStreamWaitEvent](#)接口之后，请及时调用[aclrtResetEvent](#)接口释放Event资源。

接口调用顺序：[aclrtCreateEvent](#)-->[aclrtRecordEvent](#)-->[aclrtStreamWaitEvent](#)-->[aclrtResetEvent](#)

函数原型

aclError aclrtRecordEvent(**aclrtEvent** event, **aclrtStream** stream)

参数说明

参数名	输入/输出	说明
event	输入	待记录的Event。
stream	输入	将指定Event记录在指定的Stream中。 如果使用默认Stream，此处设置为NULL。

返回值说明

返回0表示成功，返回其他值表示失败。

参考资源

接口调用流程及调用示例，参见Event的同步等待示例代码、Stream间任务的同步等待接口调用流程及示例代码。

1.5.4.6 aclrtResetEvent

函数功能

复位一个Event。用户需确保等待Stream中的任务都完成后，再复位Event，异步接口。

约束说明

- 通过aclrtCreateEventExWithFlag接口创建的Event，不支持调用本接口。
- 该接口是异步接口，调用接口成功仅表示任务下发成功，不表示任务执行成功。调用该接口后，需调用同步等待接口（例如，**aclrtSynchronizeStream**）确保任务已执行完成，否则可能会导致训练或推理等业务异常、Device断链掉卡等未知情况。

函数原型

aclError aclrtResetEvent(**aclrtEvent** event, **aclrtStream** stream)

参数说明

参数名	输入/输出	说明
event	输入	待复位的Event。
stream	输入	指定Event所在的Stream。

返回值说明

返回0表示成功，返回其他值表示失败。

1.5.4.7 aclrtQueryEvent (废弃)

须知

此接口后续版本会废弃，请使用[aclrtQueryEventStatus](#)接口。

函数功能

查询与本接口在同一线程中的[aclrtRecordEvent](#)接口所记录的Event是否执行完成。

函数原型

aclError [aclrtQueryEvent](#)([aclrtEvent](#) event, [aclrtEventStatus](#) *status)

参数说明

参数名	输入/输出	说明
event	输入	指定待查询的Event。
status	输出	Event状态的指针。

返回值说明

返回0表示成功，返回其他值表示失败。

1.5.4.8 aclrtQueryEventStatus

函数功能

查询该Event捕获的所有任务的执行状态。具体见[aclrtRecordEvent](#)接口参考Event捕获的细节。

约束说明

如果用户在不同线程上分别调用[aclrtRecordEvent](#)和[aclrtQueryEventStatus](#)，可能由于多线程导致这两个API的执行时间乱序，进而导致查询到的Event对象的完成状态不符合预期。

函数原型

aclError [aclrtQueryEventStatus](#)([aclrtEvent](#) event, [aclrtEventRecordedStatus](#) *status)

参数说明

参数名	输入/输出	说明
event	输入	指定待查询的Event。
status	输出	Event状态的指针。 如果该Event捕获的所有任务都已经执行完成则返回 ACL_EVENT_RECORDED_STATUS_COMPLETE，如果有任 何一个任务未执行完成则返回 ACL_EVENT_RECORDED_STATUS_NOT_READY。

返回值说明

返回0表示成功，返回其他值表示失败。

1.5.4.9 aclrtQueryEventWaitStatus

函数功能

调用[aclrtStreamWaitEvent](#)接口后查询该Event对应的等待任务是否都执行完成。

约束说明

通过[aclrtCreateEventExWithFlag](#)接口创建的Event，不支持调用本接口。

函数原型

aclError aclrtQueryEventWaitStatus(**aclrtEvent** event, **aclrtEventWaitStatus** *status)

参数说明

参数名	输入/输出	说明
event	输入	指定待查询的Event。
status	输出	Event状态的指针。

返回值说明

返回0表示成功，返回其他值表示失败。

1.5.4.10 aclrtSynchronizeEvent

函数功能

阻塞当前线程运行直到Event捕获的所有任务都执行完成。具体见[aclrtRecordEvent](#)接口参考Event捕获的细节。

函数原型

aclError aclrtSynchronizeEvent(**aclrtEvent** event)

参数说明

参数名	输入/输出	说明
event	输入	需等待的Event。

返回值说明

返回0表示成功，返回其他值表示失败。

参考资源

接口调用示例，参见Event的同步等待示例代码。

1.5.4.11 aclrtSynchronizeEventWithTimeout

函数功能

阻塞当前线程运行直到Event捕获的所有任务都执行完成（具体见[aclrtRecordEvent](#)接口参考Event捕获的细节），该接口是在接口[aclrtSynchronizeEvent](#)基础上进行了增强，支持用户设置永久等待、或配置具体的超时时间，若配置具体的超时时间，则当应用程序异常时可根据所设置的超时时间自行退出。

约束说明

Atlas 200/300/500 推理产品，不支持调用接口配置具体的超时时间，仅支持设置永久等待。

函数原型

aclError aclrtSynchronizeEventWithTimeout(**aclrtEvent** event, int32_t timeout)

参数说明

参数名	输入/输出	说明
event	输入	需等待的Event。
timeout	输入	接口的超时时间。 取值说明如下： • -1：表示永久等待，和接口aclrtSynchronizeEvent功能一样； • >0：配置具体的超时时间，单位是毫秒。

返回值说明

返回0表示成功，返回其他值表示失败。

1.5.4.12 aclrtEventElapsedTime

函数功能

统计两个Event之间的耗时。

约束说明

接口调用顺序：调用[aclrtCreateEvent](#)/[aclrtCreateEventWithFlag](#)接口创建Event-->调用[aclrtRecordEvent](#)接口在同一个Stream中记录起始Event、结尾Event-->调用[aclrtSynchronizeStream](#)接口阻塞应用程序运行，直到指定Stream中的所有任务都完成-->调用[aclrtEventElapsedTime](#)接口统计两个Event之间的耗时

函数原型

aclError [aclrtEventElapsedTime](#)(float *ms, [aclrtEvent](#) startEvent, [aclrtEvent](#) endEvent)

参数说明

参数名	输入/输出	说明
ms	输出	表示两个Event之间耗时的指针，单位为毫秒。
start	输入	起始Event。
end	输入	结尾Event。

返回值说明

返回0表示成功，返回其他值表示失败。

1.5.4.13 aclrtStreamWaitEvent

函数功能

阻塞指定Stream的运行，直到指定的Event完成，支持多个Stream等待同一个Event的场景。异步接口。

提交到Stream上的所有后续任务都需要等待Event捕获的任务都完成后才能开始执行。具体见[aclrtRecordEvent](#)接口了解Event捕获的细节。

约束说明

- 该接口是异步接口，调用接口成功仅表示任务下发成功，不表示任务执行成功。调用该接口后，需调用同步等待接口（例如，[aclrtSynchronizeStream](#)）确保任务已执行完成，否则可能会导致训练或推理等业务异常、Device断链掉卡等未知情况。
- [aclrtRecordEvent](#)接口与[aclrtStreamWaitEvent](#)接口配合使用时，主要用于多Stream之间同步的场景，在调用[aclrtRecordEvent](#)接口时，系统内部会申请Event资源，在调用[aclrtStreamWaitEvent](#)接口之后，请及时调用[aclrtResetEvent](#)接口释放Event资源。

接口调用顺序：[aclrtCreateEvent](#)-->[aclrtRecordEvent](#)-->[aclrtStreamWaitEvent](#)-->[aclrtResetEvent](#)

- 一个进程内，调用[aclInit](#)接口初始化AscendCL后，调用[aclrtSetOpWaitTimeout](#)接口设置超时时间，本进程内后续调用[aclrtStreamWaitEvent](#)接口下发的任务支持在所设置的超时时间内等待，若等待的时间超过所设置的超时时间，则AscendCL会返回报错。

由于[aclrtStreamWaitEvent](#)接口是异步接口，调用接口成功仅表示任务下发成功，不表示任务执行成功，因此若等待的时间超过所设置的超时时间，则在调用[aclrtSynchronizeStream](#)接口后，会返回报错。

函数原型

aclError [aclrtStreamWaitEvent](#)([aclrtStream](#) stream, [aclrtEvent](#) event)

参数说明

参数名	输入/输出	说明
stream	输入	指定需要等待Event完成的Stream。 如果使用默认Stream，此处设置为NULL。
event	输入	需等待的Event。

返回值说明

返回0表示成功，返回其他值表示失败。

参考资源

接口调用流程及调用示例，参见Event的同步等待示例代码、Stream间任务的同步等待接口调用流程及示例代码。

1.5.4.14 aclrtSetOpWaitTimeout

Atlas 推理系列产品，不支持该接口。

函数功能

本接口用于设置等待Event完成的超时时间。

约束说明

- 不调用本接口，则默认不超时；一个进程内多次调用本接口，则以最后一次设置的时间为准。
- 一个进程内，调用[aclInit](#)接口初始化AscendCL后，调用本接口设置超时时间，本进程内后续调用[aclrtStreamWaitEvent](#)接口下发的任务支持在所设置的超时时间内等待。
由于[aclrtStreamWaitEvent](#)接口是异步接口，调用接口成功仅表示任务下发成功，不表示任务执行成功，因此若等待的时间超过所设置的超时时间，则在调用[aclrtSynchronizeStream](#)接口后，会返回报错。

函数原型

aclError [aclrtSetOpWaitTimeout](#)(uint32_t timeout)

参数说明

参数名	输入/输出	说明
timeout	输入	设置超时时间，单位为秒。 将该参数设置为0时，表示不超时。

返回值说明

返回0表示成功，返回其他值表示失败。

1.5.5 内存管理

1.5.5.1 总体说明

- 对于媒体数据处理功能，关于内存使用，有以下注意事项：
 - a. 若涉及媒体数据处理功能，由于媒体数据处理功能对存放输入、输出数据的内存有更高的要求（例如，内存首地址128字节对齐），因此需调用专用的内存申请接口，如下：

- 调用**媒体数据处理V1**版本的接口对图片进行抠图、缩放等操作时，调用[acldvppMalloc](#)接口申请内存。
- 调用**媒体数据处理V2**版本的接口对图片进行抠图、缩放等操作时，调用[hi_mpi_dvpp_malloc](#)接口申请内存。
- b. 调用[a](#)申请出来的内存可以满足媒体数据处理的要求，也可以在其它任务中使用，例如，从性能角度，为了减少拷贝，媒体数据处理的输出作为模型推理的输入，实现内存复用。
- c. 但由于媒体数据处理访问的地址空间有限，为确保媒体数据处理时内存足够，除媒体数据处理功能外的其它功能（例如，模型加载），建议调用[内存管理](#)章节下的[aclrtMalloc](#)接口、或[aclrtMallocHost](#)接口、或[aclrtMallocCached](#)接口申请内存。

Atlas 推理系列产品上，媒体数据处理功能在每个进程内，可访问的地址空间最大为16GB。

Atlas A2训练系列产品/Atlas 800I A2推理产品上，媒体数据处理功能在每个进程内，可访问的地址空间最大为16GB。
- 对于Atlas 200/300/500 推理产品，若涉及申请Device上的大页内存，需注意，当前版本为防止大页内存不够用，系统内部已提前预留（Device操作系统上的可用内存*25%≈1940M）大页内存，用户在申请大页内存时优先使用这部分预留内存。Device操作系统上的可用内存指使用free命令查询得到的total内存。
 - 在标准形态下，使用完预留的大页内存后，系统内部会根据Device上的内存动态扩展或缩减大页内存。
- 对于Atlas A2训练系列产品/Atlas 800I A2推理产品，若涉及申请Device上的大页内存，需注意，当前版本为防止大页内存不够用，系统内部已提前预留大页内存，用户在使用大页内存前，可调用[aclrtGetMemInfo](#)接口，查询空闲的HBM大页内存（对应ACL_HBM_MEM_HUGE类型）、HBM普通内存（对应ACL_HBM_MEM_NORMAL类型）。
- Ascend EP形态下，Device启动后首次执行推理或训练任务时，会将AI CPU算子从Host搬迁到Device上并缓存在Device上，用于提升性能，因此会占用部分Device内存（100M~200M，不同芯片有所不同）。若Device重启，这部分AI CPU算子缓存才会释放。

以下产品涉及该约束：

 - Atlas 推理系列产品
 - Atlas 200/500 A2推理产品
 - Atlas 训练系列产品
 - Atlas A2 训练系列产品

1.5.5.2 aclrtMalloc

函数功能

在Device上分配size大小的线性内存，并通过*devPtr返回已分配内存的指针。本接口分配的内存会进行字节对齐，会对用户申请的size向上对齐成32字节整数倍后再多加32字节。

约束说明

- 若涉及媒体数据处理功能，由于媒体数据处理功能对存放输入、输出数据的内存有更高的要求（例如，内存首地址128字节对齐），因此需调用专用的内存申请接口，如下：
 - 调用**媒体数据处理V1**版本的接口对图片进行抠图、缩放等操作时，调用**aclDvppMalloc**接口申请内存。
 - 调用**媒体数据处理V2**版本的接口对图片进行抠图、缩放等操作时，调用**hi_mpi_dvpp_malloc**接口申请内存。
- 本接口分配的内存不会进行对内容进行初始化，建议在使用内存前先调用**aclrtMemset**接口先初始化内存，清除内存中的随机数。
- 本接口内部不会进行隐式的Device同步或流同步。如果申请内存成功或申请内存失败会立刻返回结果。
- 使用**aclrtMalloc**接口申请的内存，需要通过**aclrtFree**接口释放内存。
- 频繁调用**aclrtMalloc**接口申请内存、调用**aclrtFree**接口释放内存，会损耗性能，建议用户提前做内存预先分配或二次管理，避免频繁申请/释放内存。
- 若用户需申请大块内存并自行划分、管理内存时，建议使用**aclrtMallocAlign32**接口，该接口相比**aclrtMalloc**接口，只会对用户申请的size向上对齐成32字节整数倍，不会再多加32字节。
不管是**aclrtMalloc**接口，还是**aclrtMallocAlign32**接口，若用户使用本接口申请大块内存并自行划分、管理内存时，每段内存需同时满足以下需求：
 - 内存大小向上对齐成32整数倍+32字节（ $m = \text{ALIGN\_UP}[len, 32] + 32$ 字节）；
 - 内存起始地址需满足64字节对齐（ $\text{ALIGN\_UP}[m, 64]$ ）。

说明

len表示某段内存的大小， $\text{ALIGN\_UP}[len, k]$ 表示向上按k字节对齐： $((len-1)/k+1)*k$ 。

函数原型

aclError **aclrtMalloc**(void **devPtr, size_t size, **aclrtMemMallocPolicy** policy)

参数说明

参数名	输入/输出	说明
devPtr	输出	“Device上已分配内存的指针”的指针。
size	输入	申请内存的大小，单位Byte。 size不能为0。
policy	输入	内存分配规则。

返回值说明

返回0表示成功，返回其他值表示失败。

参考资源

接口调用示例，参见数据传输。

1.5.5.3 aclrtMallocAlign32

函数功能

在Device上分配size大小的线性内存，并通过*devPtr返回已分配内存的指针。本接口分配的内存会进行字节对齐，会对用户申请的size向上对齐成32字节整数倍。

与aclrtMalloc接口相比，本接口只会对用户申请的size向上对齐成32字节整数倍，不会再多加32字节。

约束说明

- 若涉及媒体数据处理功能，由于媒体数据处理功能对存放输入、输出数据的内存有更高的要求（例如，内存首地址128字节对齐），因此需调用专用的内存申请接口，如下：
 - 调用[媒体数据处理V1](#)版本的接口对图片进行抠图、缩放等操作时，调用[acldvppMalloc](#)接口申请内存。
 - 调用[媒体数据处理V2](#)版本的接口对图片进行抠图、缩放等操作时，调用[hi_mpi_dvpp_malloc](#)接口申请内存。
- 本接口分配的内存不会进行对内容进行初始化。
- 本接口内部不会进行隐式的Device同步或流同步。如果申请内存成功或申请内存失败会立刻返回结果。
- 使用aclrtMallocAlign32接口申请的内存，需要通过[aclrtFree](#)接口释放内存。
- 频繁调用aclrtMallocAlign32接口申请内存、调用[aclrtFree](#)接口释放内存，会损耗性能，建议用户提前做内存预先分配或二次管理，避免频繁申请/释放内存。
- 若用户使用本接口申请大块内存并自行划分、管理内存时，每段内存需同时满足以下需求：
 - 内存大小向上对齐成32整数倍+32字节（ $m = \text{ALIGN\_UP}[\text{len}, 32] + 32$ 字节）；
 - 内存起始地址需满足64字节对齐（ $\text{ALIGN\_UP}[m, 64]$ ）。

说明

len表示某段内存的大小， $\text{ALIGN\_UP}[\text{len}, k]$ 表示向上按k字节对齐： $((\text{len}-1)/k+1)*k$ 。

函数原型

aclError [aclrtMallocAlign32](#)(void **devPtr, size_t size, [aclrtMemMallocPolicy](#) policy)

参数说明

参数名	输入/输出	说明
devPtr	输出	“Device上已分配内存的指针”的指针。

参数名	输入/输出	说明
size	输入	申请内存的大小，单位Byte。 size不能为0。
policy	输入	内存分配规则。

返回值说明

返回0表示成功，返回其他值表示失败。

1.5.5.4 aclrtMallocCached

函数功能

在Device上申请size大小的线性内存，通过*devPtr返回已分配内存的指针，该接口在任何场景下申请的内存都是支持cache缓存。

使用aclrtMallocCached接口申请的内存与使用[aclrtMalloc](#)接口申请的内存是等价的，都支持cache缓存，不需要用户处理cpu与npu之间的cache一致性。

调用[媒体数据处理](#)的接口前，若需要申请Device上的内存存放输入或输出数据，需调用[aclDvppMalloc](#)申请内存。

约束说明

其它约束与[aclrtMalloc](#)接口相同。

函数原型

aclError aclrtMallocCached(void **devPtr, size_t size, **aclrtMemMallocPolicy** policy)

参数说明

参数名	输入/输出	说明
devPtr	输出	“Device上已分配内存的指针”的指针。
size	输入	申请内存的大小，单位Byte。 size不能为0。
policy	输入	内存分配规则。

返回值说明

返回0表示成功，返回其他值表示失败。

1.5.5.5 aclrtMemFlush

函数功能

将cache中的数据刷新到ddr中，并将cache中的内容设置成无效。

该版本不需要用户处理cpu与npu之间的cache一致性，无需调用该接口。

函数原型

aclError aclrtMemFlush(void *devPtr, size_t size)

参数说明

参数名	输入/输出	说明
devPtr	输入	要flush的ddr内存起始地址指针。
size	输入	要flush的ddr内存大小，单位Byte。 size不能为0。

返回值说明

返回0表示成功，返回其他值表示失败。

1.5.5.6 aclrtMemInvalidate

函数功能

将cache中的数据设置成无效。

该版本不需要用户处理cpu与npu之间的cache一致性，无需调用该接口。

函数原型

aclError aclrtMemInvalidate(void *devPtr, size_t size)

参数说明

参数名	输入/输出	说明
devPtr	输入	需要将其中cache数据置为无效的ddr内存起始地址指针。
size	输入	ddr内存大小，单位Byte。 size不能为0。

返回值说明

返回0表示成功，返回其他值表示失败。

1.5.5.7 aclrtFree

函数功能

释放Device上的内存。

约束说明

- **aclrtFree**接口只能释放通过**aclrtMalloc**接口或**aclrtMallocCached**接口或**aclrtMallocAlign32**接口申请的内存。
- 本接口会立刻释放传入的内存，函数内部不会进行隐式的Device同步或流同步。用户需要确保调用完本接口后不再对该内存指针进行访问。

函数原型

aclError **aclrtFree**(void *devPtr)

参数说明

参数名	输入/输出	说明
devPtr	输入	待释放内存的指针。 如果传入的devPtr为空指针，本接口会返回报错。

返回值说明

返回0表示成功，返回其他值表示失败。

参考资源

接口调用示例，参见数据传输。

1.5.5.8 aclrtMallocHost

函数功能

申请Host内存（该内存是锁页内存），由系统保证内存首地址64字节对齐。

需注意：Ascend RC形态下，调用该接口申请的是Device内存，且Device上的内存按普通页申请，如需首地址64字节对齐，需要用户自行处理对齐。

需注意：Device Control CPU开放形态下，调用该接口申请的是Device内存，且Device上的内存按普通页申请，如需首地址64字节对齐，需要用户自行处理对齐。

约束说明

- 本接口分配的内存不会进行对内容进行初始化，建议在使用内存前先调用[aclrtMemset](#)接口先初始化内存，清除内存中的随机数。
- 本接口内部不会进行隐式的device同步或流同步。如果申请内存成功或申请内存失败会立刻返回结果。
- 使用[aclrtMallocHost](#)接口申请的内存，需要通过[aclrtFreeHost](#)接口释放内存。
- 使用[aclrtMallocHost](#)接口分配过多的锁页内存，将导致操作系统用于分页的物理内存变少，从而降低系统整体的性能。
- 频繁调用[aclrtMallocHost](#)接口申请内存、调用[aclrtFreeHost](#)接口释放内存，会损耗性能，建议用户提前做内存预先分配或二次管理，避免频繁申请/释放内存。
- 若用户使用本接口申请大块内存并自行划分、管理内存时，每段内存需同时满足以下需求：
 - 内存大小向上对齐成32整数倍+32字节（ $m = \text{ALIGN\_UP}[\text{len}, 32] + 32$ 字节）；
 - 内存起始地址需满足64字节对齐（ $\text{ALIGN\_UP}[m, 64]$ ）。

说明

len表示某段内存的大小，ALIGN_UP[len,k]表示向上按k字节对齐： $((\text{len}-1)/k+1)*k$ 。

函数原型

```
aclError aclrtMallocHost(void **hostPtr, size_t size)
```

参数说明

参数名	输入/输出	说明
size	输入	申请内存的大小，单位Byte。 size不能为0。
hostPtr	输出	“已分配内存的指针”的指针。

返回值说明

返回0表示成功，返回其他值表示失败。

参考资源

接口调用示例，参见数据传输。

1.5.5.9 aclrtFreeHost

函数功能

释放内存。

约束说明

- 本接口会立刻释放传入的内存，函数内部不会进行隐式的Device同步或流同步。用户需要确保调用完本接口后不再对该内存指针进行访问。
- [aclrtFreeHost](#)接口只能释放通过[aclrtMallocHost](#)接口申请的内存。

函数原型

aclError [aclrtFreeHost](#)(void *hostPtr)

参数说明

参数名	输入/输出	说明
hostPtr	输入	待释放内存的指针。

返回值说明

返回0表示成功，返回其他值表示失败。

参考资源

接口调用示例，参见数据传输。

1.5.5.10 aclrtMemset

函数功能

初始化内存，将内存中的内容设置为指定的值。
要初始化的内存支持在Host侧或Device侧，系统根据地址判定是Host还是Device。

约束说明

本接口会立刻进行内存初始化，函数内部不会进行隐式的device同步或流同步。

函数原型

aclError [aclrtMemset](#)(void *devPtr, size_t maxCount, int32_t value, size_t count)

参数说明

参数名	输入/输出	说明
devPtr	输入	内存起始地址的指针。
maxCount	输入	内存的最大长度，单位Byte。

参数名	输入/输出	说明
value	输入	设置的值。
count	输入	需要设置为指定值的内存长度，单位Byte。

返回值说明

返回0表示成功，返回其他值表示失败。

1.5.5.11 aclrtMemsetAsync

函数功能

初始化内存，将内存中的内容设置为指定的值，异步接口。
要初始化的内存支持在Host侧或Device侧，系统根据地址判定是Host还是Device。

约束说明

该接口是异步接口，调用接口成功仅表示任务下发成功，不表示任务执行成功。调用该接口后，需调用同步等待接口（例如，[aclrtSynchronizeStream](#)）确保内存初始化的任务已执行完成，否则可能会导致训练或推理等业务异常、Device断链掉卡等未知情况。

函数原型

aclError [aclrtMemsetAsync](#)(void *devPtr, size_t maxCount, int32_t value, size_t count, [aclrtStream](#) stream)

参数说明

参数名	输入/输出	说明
devPtr	输入	内存起始地址的指针。
maxCount	输入	内存的最大长度，单位Byte。
value	输入	设置的值。
count	输入	需要设置为指定值的内存长度，单位Byte。
stream	输入	指定stream。

返回值说明

返回0表示成功，返回其他值表示失败。

1.5.5.12 aclrtMemcpy

函数功能

实现内存复制。

约束说明

- 本接口会立刻进行内存复制，函数内部不会进行隐式的device同步或流同步。
- Atlas 训练系列产品，如果执行Device间的内存复制，需先调用[aclrtDeviceCanAccessPeer](#)接口查询两个Device间是否支持内存复制、调用[aclrtDeviceEnablePeerAccess](#)接口使用两个Device间的内存复制，再调用本接口进行内存复制。
- Atlas A2训练系列产品/Atlas 800I A2推理产品，如果执行Device间的内存复制，需先调用[aclrtDeviceCanAccessPeer](#)接口查询两个Device间是否支持内存复制、调用[aclrtDeviceEnablePeerAccess](#)接口使用两个Device间的内存复制，再调用本接口进行内存复制。
- Atlas 推理系列产品，如果执行Device间的内存复制，需先调用[aclrtDeviceCanAccessPeer](#)接口查询两个Device间是否支持内存复制、调用[aclrtDeviceEnablePeerAccess](#)接口使用两个Device间的内存复制，再调用本接口进行内存复制。

函数原型

aclError [aclrtMemcpy](#)(void *dst, size_t destMax, const void *src, size_t count, aclrtMemcpyKind kind)

参数说明

参数名	输入/输出	说明
dst	输入	目的内存地址指针。
destMax	输入	目的内存地址的最大内存长度，单位Byte。
src	输入	源内存地址指针。
count	输入	内存复制的长度，单位Byte。
kind	输入	内存复制的类型，预留参数，配置枚举值中的值无效，系统内部会根据源内存地址指针、目的内存地址指针判断是否可以将源地址的数据复制到目的地址，如果不可以，则系统会返回报错。 <pre>typedef enum aclrtMemcpyKind { ACL_MEMCPY_HOST_TO_HOST, // Host内的内存复制 ACL_MEMCPY_HOST_TO_DEVICE, // Host到Device的内存复制 ACL_MEMCPY_DEVICE_TO_HOST, // Device到Host的内存复制 ACL_MEMCPY_DEVICE_TO_DEVICE, // Device内或Device间的内存复制 ACL_MEMCPY_DEFAULT, // 由系统根据源、目的内存地址自行判断拷贝方向 } aclrtMemcpyKind;</pre>

返回值说明

返回0表示成功，返回其他值表示失败。

参考资源

接口调用示例，参见数据传输。

1.5.5.13 aclrtMemcpyAsync

函数功能

实现内存复制，异步接口。

约束说明

- 调用本接口进行内存复制时，源地址和目的地址都必须64字节对齐。
- 该接口是异步接口，调用接口成功仅表示任务下发成功，不表示任务执行成功。调用该接口后，需调用同步等待接口（例如，[aclrtSynchronizeStream](#)）确保内存复制的任务已执行完成，否则可能会导致训练或推理等业务异常、Device断链掉卡等未知情况。
- 若调用本接口传入的kind为ACL_MEMCPY_HOST_TO_DEVICE或ACL_MEMCPY_DEVICE_TO_HOST，并且Host内存不是通过[aclrtMallocHost](#)申请得到的，本接口不会报错，但是在实际执行到该任务时会发生不可预知的错误。
- 本接口不支持异步Host内的内存复制功能，若传入的kind为ACL_MEMCPY_HOST_TO_HOST时，接口返回报错ACL_ERROR_RT_FEATURE_NOT_SUPPORT。
- Ascend RC形态下，在板端运行应用时，选择涉及Host的内存复制类型后，系统内部会默认使用Device内的内存复制。
当前以下产品版本支持Ascend RC形态：
 - Atlas 200/300/500 推理产品
 - Atlas 200/500 A2推理产品
- 如果执行Device间的内存复制，需先调用[aclrtDeviceCanAccessPeer](#)接口查询两个Device间是否支持内存复制、调用[aclrtDeviceEnablePeerAccess](#)接口使用两个Device间的内存复制，再调用本接口进行内存复制。
目前以下产品版本支持[aclrtDeviceCanAccessPeer](#)、[aclrtDeviceEnablePeerAccess](#)接口：
 - Atlas 推理系列产品
 - Atlas 训练系列产品
 - Atlas A2训练系列产品/Atlas 800I A2推理产品

函数原型

```
aclError aclrtMemcpyAsync(void *dst, size_t destMax, const void *src, size_t count, aclrtMemcpyKind kind, aclrtStream stream)
```

参数说明

参数名	输入/输出	说明
dst	输入	目的内存地址指针。
destMax	输入	目的内存地址的最大内存长度，单位Byte。
src	输入	源内存地址指针。
count	输入	内存复制的长度，单位Byte。
kind	输入	内存复制的类型。 typedef enum aclrtMemcpyKind { ACL_MEMCPY_HOST_TO_HOST, // Host内的内存复制 ACL_MEMCPY_HOST_TO_DEVICE, // Host到Device的内存复制 ACL_MEMCPY_DEVICE_TO_HOST, // Device到Host的内存复制 ACL_MEMCPY_DEVICE_TO_DEVICE, // Device内或Device间的内存复制 ACL_MEMCPY_DEFAULT, // 由系统根据源、目的内存地址自行判断拷贝方向 } aclrtMemcpyKind;
stream	输入	指定stream。

返回值说明

返回0表示成功，返回其他值表示失败。

参考资源

接口调用示例，参见数据传输。

1.5.5.14 aclrtGetMemInfo

函数功能

根据指定属性，获取Device上AscendCL应用可用内存的空闲大小和总大小，不包括系统预留内存大小。

约束说明

- 调用本接口前是必须先指定用于计算的Device（例如调用aclrtSetDevice接口指定用于计算的Device），因此本接口中不体现Device ID。
- 请根据实际硬件支持的情况，选择对应属性的内存，否则调用本接口获取到的空闲大小和总大小都为0。
该约束适用以下项：
Atlas 200/300/500 推理产品
Atlas 训练系列产品
Atlas A2训练系列产品/Atlas 800I A2推理产品
- 由于Device上没有HBM相关的内存，若指定查询HBM相关内存，则AscendCL系统内部自动转换为查询DDR相关内存，例如，若查询ACL_HBM_MEM，则

AscendCL系统内部实际查询ACL_DDR_MEM；若查询ACL_HBM_MEM_HUGE，则AscendCL系统内部实际查询ACL_DDR_MEM_HUGE。

该约束适用于以下项：

- Atlas 200/500 A2推理产品
- Atlas 推理系列产品

函数原型

```
aclError aclrtGetMemInfo(aclrtMemAttr attr, size_t *free, size_t *total)
```

参数说明

参数名	输入/输出	说明
attr	输入	需要查询的内存的属性值。
free	输出	对应属性内存空闲大小的指针，单位Byte。
total	输出	对应属性内存总大小的指针，单位Byte。

返回值说明

返回0表示成功，返回其他值表示失败。

1.5.5.15 aclrtMemcpy2d

Atlas 推理系列加速模块产品不支持该接口。

函数功能

实现同步内存复制，主要用于矩阵数据的复制。

函数原型

```
aclError aclrtMemcpy2d(void *dst, size_t dpitch, const void *src, size_t spitch, size_t width, size_t height, aclrtMemcpyKind kind)
```

约束说明

- 当前仅支持ACL_MEMCPY_HOST_TO_DEVICE类型和ACL_MEMCPY_DEVICE_TO_HOST类型的内存复制。
- Atlas 200/300/500 推理产品，Ascend RC形态下，不支持调用本接口。
- Atlas 200/500 A2推理产品，Ascend RC形态下，不支持调用本接口。

参数说明

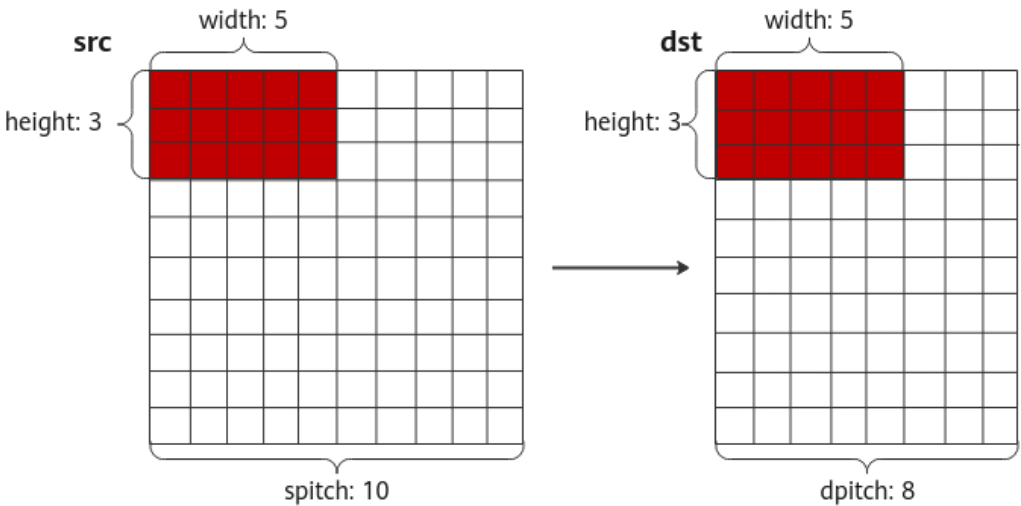
参数名	输入/输出	说明
dst	输入	目的内存地址指针。
dpitch	输入	目的内存中相邻两列向量的地址距离。
src	输入	源内存地址指针。
spitch	输入	源内存中相邻两列向量的地址距离。
width	输入	待复制的数据宽度。
height	输入	待复制的数据高度。
kind	输入	内存复制的类型。 typedef enum aclrtMemcpyKind { ACL_MEMCPY_HOST_TO_HOST, // Host内的内存复制 ACL_MEMCPY_HOST_TO_DEVICE, // Host到Device的内存复制 ACL_MEMCPY_DEVICE_TO_HOST, // Device到Host的内存复制 ACL_MEMCPY_DEVICE_TO_DEVICE, // Device内或Device间的内存复制 ACL_MEMCPY_DEFAULT, // 由系统根据源、目的内存地址自行判断拷贝方向 } aclrtMemcpyKind;

返回值说明

返回0表示成功，返回其他值表示失败。

参考信息

本接口的内存复制示意图：



复制数据时，顺序为：每一行是从左到右，两行之间从上到下

1.5.5.16 aclrtMemcpy2dAsync

Atlas 推理系列加速模块产品不支持该接口。

函数功能

实现异步内存复制，主要用于矩阵数据的复制。异步接口。

约束说明

- 该接口是异步接口，调用接口成功仅表示任务下发成功，不表示任务执行成功。调用该接口后，需调用同步等待接口（例如，[aclrtSynchronizeStream](#)）确保内存复制的任务已执行完成，否则可能会导致训练或推理等业务异常、Device断链掉卡等未知情况。
- 当前仅支持ACL_MEMCPY_HOST_TO_DEVICE类型和ACL_MEMCPY_DEVICE_TO_HOST类型的内存复制。
- Atlas 200/300/500 推理产品，Ascend RC形态下，不支持调用本接口。
- Atlas 200/500 A2推理产品，Ascend RC形态下，不支持调用本接口。

函数原型

aclError [aclrtMemcpy2dAsync](#)(void *dst, size_t dpitch, const void *src, size_t spitch, size_t width, size_t height, [aclrtMemcpyKind](#) kind, [aclrtStream](#) stream)

参数说明

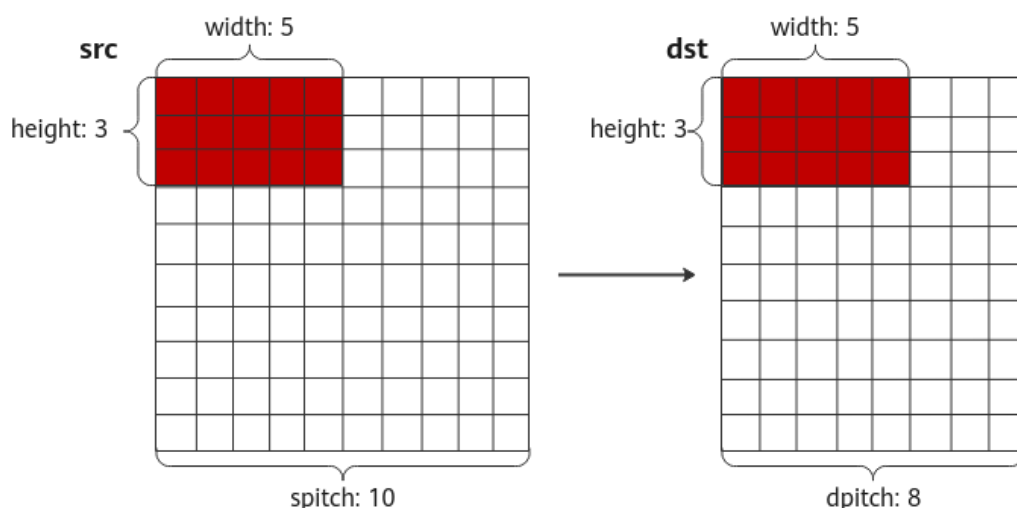
参数名	输入/输出	说明
dst	输入	目的内存地址指针。
dpitch	输入	目的内存中相邻两列向量的地址距离。
src	输入	源内存地址指针。
spitch	输入	源内存中相邻两列向量的地址距离。
width	输入	待复制的数据宽度。
height	输入	待复制的数据高度。 height最大设置为5*1024*1024=5242880，否则接口返回失败。
kind	输入	内存复制的类型。 typedef enum aclrtMemcpyKind { ACL_MEMCPY_HOST_TO_HOST, // Host内的内存复制 ACL_MEMCPY_HOST_TO_DEVICE, // Host到Device的内存复制 ACL_MEMCPY_DEVICE_TO_HOST, // Device到Host的内存复制 ACL_MEMCPY_DEVICE_TO_DEVICE, // Device内或Device间的内存复制 ACL_MEMCPY_DEFAULT, // 由系统根据源、目的内存地址自行判断拷贝方向 } aclrtMemcpyKind;
stream	输入	指定stream。

返回值说明

返回0表示成功，返回其他值表示失败。

参考信息

本接口的内存复制示意图：



复制数据时，顺序为：每一行是从左到右，两行之间从上到下

1.5.5.17 aclrtAllocatorRegister

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 200/500 A2推理产品，不支持该接口。

函数功能

调用该接口注册用户提供的Allocator以及Allocator对应的回调函数，用于使用用户提供的Allocator。

约束说明

- 当前仅支持在单算子模型执行、动态shape模型推理场景下使用本接口。
单算子模型场景下，需在算子执行接口（例如：[aclopExecuteV2](#)、[aclopCompileAndExecuteV2](#)等）之前调用本接口。
动态shape模型推理场景，本接口需配合[aclmdlExecuteAsync](#)接口一起使用，且需在[aclmdlExecuteAsync](#)接口之前调用本接口。
- 调用本接口前，需要先调用[aclrtAllocatorCreateDesc](#)创建Allocator描述符，再分别调用[aclrtAllocatorSetObjToDesc](#)、[aclrtAllocatorSetAllocFuncToDesc](#)、[aclrtAllocatorSetGetAddrFromBlockFuncToDesc](#)、[aclrtAllocatorSetFreeFuncToDesc](#)设置Allocator对象及回调函数。Allocator描述符使用完成后，可调用[aclrtAllocatorDestroyDesc](#)接口销毁Allocator描述符。
- 对于同一条流，多次调用本接口，以最后一次注册为准。

- 对于不同流，如果用户使用同一个Allocator，不可以多条流并发执行，在执行下一条Stream前，需要对上一Stream做流同步。
- 将Allocator中的内存释放给操作系统前，需要先调用[aclrtSynchronizeStream](#)接口执行流同步，确保Stream中的任务已执行完成。

函数原型

aclError [aclrtAllocatorRegister](#)([aclrtStream](#) stream, [aclrtAllocatorDesc](#) allocatorDesc)

参数说明

参数名	输入/输出	说明
stream	输入	该Allocator需要注册的Stream。 传入的stream参数值不能为NULL，否则返回报错。
allocatorDesc	输入	Allocator描述符指针。

返回值说明

返回0表示成功，返回其他值表示失败。

参考资源

接口调用示例，参见动态Shape输入（设置Shape范围）。

1.5.5.18 aclrtAllocatorUnregister

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 200/500 A2推理产品，不支持该接口。

函数功能

取消注册用户提供的Allocator以及Allocator对应的回调函数，用于取消使用用户提供的Allocator。

约束说明

- 用户销毁Allocator前，调用本接口取消注册。
- 待取消注册的Stream不存在，或多次调用本接口取消注册，本接口内部不做任何操作，返回成功。

函数原型

aclError [aclrtAllocatorUnregister](#)([aclrtStream](#) stream)

参数说明

参数名	输入/输出	说明
stream	输入	该Allocator对应的stream。

返回值说明

返回0表示成功，返回其他值表示失败。

参考资源

接口调用示例，参见动态Shape输入（设置Shape范围）。

1.5.5.19 aclrtMallocPhysical

Atlas 200/300/500 推理产品，不支持该接口。

函数功能

申请Device物理内存，并返回一个物理内存handle。

本接口可配合[aclrtReserveMemAddress](#)接口（申请虚拟内存）、[aclrtMapMem](#)接口（建立虚拟内存与物理内存之间的映射）使用，以便申请地址连续的虚拟内存、最大化利用物理内存。

本接口可配合[aclrtMemExportToShareableHandle](#)接口（导出物理内存handle）、[aclrtMemImportFromShareableHandle](#)（导入共享handle）使用，用于实现多进程之间的物理内存共享。同时，也支持在共享物理内存时，使用虚拟内存，请参见[aclrtMemExportToShareableHandle](#)接口处的说明。

约束说明

- Ascend RC形态不支持调用本接口。
- 当前版本仅支持ACL_HBM_MEM_HUGE（表示HBM大页内存）、ACL_HBM_MEM_NORMAL（表示HBM普通页内存）两种内存类型，若传入ACL_HBM_MEM_NORMAL类型，系统内部当前也按HBM大页内存申请。
在Atlas 推理系列产品上，由于Device上没有HBM内存，使用本接口申请ACL_HBM_MEM_HUGE类型的内存时，AscendCL系统内部自动转换为申请ACL_DDR_MEM_HUGE类型的DDR内存。
在 Atlas 200/500 A2推理产品上，由于Device上没有HBM内存，使用本接口申请ACL_HBM_MEM_HUGE类型的内存时，AscendCL系统内部自动转换为申请ACL_DDR_MEM_HUGE类型的DDR内存。

函数原型

aclError **aclrtMallocPhysical**(**aclrtDrvMemHandle** *handle, size_t size, const **aclrtPhysicalMemProp** *prop, uint64_t flags)

参数说明

参数名	输入/输出	说明
handle	输出	存放物理内存信息的handle。
size	输入	物理内存大小，单位Byte。 先调用 aclrtMemGetAllocationGranularity 接口获取内存申请粒度，然后再调用本接口申请物理内存时size按获取到的内存申请粒度对齐，以便节约内存。 当前内存申请粒度固定2M，因此size当前需按照2M对齐。
prop	输入	物理内存属性信息。
flags	输入	预留，当前只能设置为0。

返回值说明

返回0表示成功，返回其他值表示失败。

1.5.5.20 aclrtFreePhysical

Atlas 200/300/500 推理产品，不支持该接口。

函数功能

释放通过[aclrtMallocPhysical](#)接口申请的物理内存。

约束说明

- Ascend RC形态不支持调用本接口。
- 若该物理内存与虚拟内存存在映射关系，则释放物理内存前，需调用[aclrtUnmapMem](#)接口取消该物理内存与虚拟内存的映射。

函数原型

aclError [aclrtFreePhysical](#)([aclrtDrvMemHandle](#) handle)

参数说明

参数名	输入/输出	说明
handle	输入	待释放的物理内存信息handle。

返回值说明

返回0表示成功，返回其他值表示失败。

1.5.5.21 aclrtReserveMemAddress

函数功能

预留虚拟内存。

本接口需与以下其它接口配合使用，以便申请地址连续的虚拟内存、最大化利用物理内存：

1. 申请虚拟内存（[aclrtReserveMemAddress](#)接口）；
2. 申请物理内存（[aclrtMallocPhysical](#)接口）；
3. 将虚拟内存映射到物理内存（[aclrtMapMem](#)接口）；
4. 执行任务（调用具体的任务接口）；
5. 取消虚拟内存与物理内存的映射（[aclrtUnmapMem](#)接口）；
6. 释放物理内存（[aclrtFreePhysical](#)接口）；
7. 释放虚拟内存（[aclrtReleaseMemAddress](#)接口）。

约束说明

Ascend RC形态不支持调用本接口。

使用本接口预留的虚拟内存，单进程场景下只支持调用[aclrtMemcpyAsync](#)接口实现两个Device之间的数据拷贝。

函数原型

```
aclError aclrtReserveMemAddress(void **virPtr, size_t size, size_t alignment,  
void *expectPtr, uint64_t flags)
```

参数说明

参数名	输入/输出	说明
virPtr	输出	“已分配的虚拟内存地址的指针”的指针。
size	输入	虚拟内存大小，单位Byte。 size不能为0。
alignment	输入	虚拟地址对齐值，预留，当前只能设置为0。
expectPtr	输入	指定期望返回的虚拟内存起始地址，预留，当前只能传nullptr。
flags	输入	大页/普通页标志，此处的标志需与 aclrtMallocPhysical 接口的内存类型保持一致。 参数取值如下： ● 0：普通页 ● 1：大页

返回值说明

返回0表示成功，返回其他值表示失败。

1.5.5.22 aclrtReleaseMemAddress

函数功能

释放通过[aclrtReserveMemAddress](#)接口申请的虚拟内存。

本接口需与以下其它接口配合使用，以便申请地址连续的虚拟内存、最大化利用物理内存：

1. 申请虚拟内存（[aclrtReserveMemAddress](#)接口）；
2. 申请物理内存（[aclrtMallocPhysical](#)接口）；
3. 将虚拟内存映射到物理内存（[aclrtMapMem](#)接口）；
4. 执行任务（调用具体的任务接口）；
5. 取消虚拟内存与物理内存的映射（[aclrtUnmapMem](#)接口）；
6. 释放物理内存（[aclrtFreePhysical](#)接口）；
7. 释放虚拟内存（[aclrtReleaseMemAddress](#)接口）。

约束说明

- Ascend RC形态不支持调用本接口。
- 若该虚拟内存与物理内存存在映射关系，则释放虚拟内存前，需调用[aclrtUnmapMem](#)接口取消该虚拟内存与物理内存的映射。

函数原型

aclError [aclrtReleaseMemAddress](#)(void *virPtr)

参数说明

参数名	输入/输出	说明
virPtr	输入	待释放的虚拟内存地址指针。

返回值说明

返回0表示成功，返回其他值表示失败。

1.5.5.23 aclrtMapMem

函数功能

将虚拟内存映射到物理内存。

本接口需与以下其它接口配合使用，以便申请地址连续的虚拟内存、最大化利用物理内存：

- 1. 申请虚拟内存（[aclrtReserveMemAddress](#)接口）；
- 2. 申请物理内存（[aclrtMallocPhysical](#)接口）；
- 3. 将虚拟内存映射到物理内存（[aclrtMapMem](#)接口）；
- 4. 执行任务（调用具体的任务接口）；
- 5. 取消虚拟内存与物理内存的映射（[aclrtUnmapMem](#)接口）；
- 6. 释放物理内存（[aclrtFreePhysical](#)接口）；
- 7. 释放虚拟内存（[aclrtReleaseMemAddress](#)接口）。

约束说明

Ascend RC形态不支持调用本接口。

函数原型

[aclError](#) [aclrtMapMem](#)(void *virPtr, size_t size, size_t offset, [aclrtDrvMemHandle](#) handle, uint64_t flags)

参数说明

参数名	输入/输出	说明
virPtr	输入	待映射的虚拟内存地址指针。 这个地址不一定是起始地址，用户也可以根据起始地址自行偏移后，再映射。 虚拟内存地址当前需按照2M对齐。
size	输入	待映射的内存大小，单位Byte。 size当前需按照2M对齐。
offset	输入	物理内存偏移值，当前只能设置为0。
handle	输入	物理内存信息handle。
flags	输入	预留，当前只能设置为0。

返回值说明

返回0表示成功，返回其他值表示失败。

1.5.5.24 [aclrtUnmapMem](#)

函数功能

取消虚拟内存与物理内存之间的映射关系。

本接口需与以下其它接口配合使用，以便申请地址连续的虚拟内存、最大化利用物理内存：

1. 申请虚拟内存（[aclrtReserveMemAddress](#)接口）；
2. 申请物理内存（[aclrtMallocPhysical](#)接口）；
3. 将虚拟内存映射到物理内存（[aclrtMapMem](#)接口）；
4. 执行任务（调用具体的任务接口）；
5. 取消虚拟内存与物理内存的映射（[aclrtUnmapMem](#)接口）；
6. 释放物理内存（[aclrtFreePhysical](#)接口）；
7. 释放虚拟内存（[aclrtReleaseMemAddress](#)接口）。

约束说明

Ascend RC形态不支持调用本接口。

函数原型

aclError [aclrtUnmapMem](#)(void *virPtr)

参数说明

参数名	输入/输出	说明
virPtr	输入	待取消映射的虚拟内存地址指针。

返回值说明

返回0表示成功，返回其他值表示失败。

1.5.5.25 aclrtMemExportToShareableHandle

函数功能

将本进程通过[aclrtMallocPhysical](#)接口获取到的物理内存handle导出，以便后续将物理内存共享给其它进程。

本接口需与以下其它关键接口配合使用，以便实现内存共享的目的（此处以A、B进程为例，说明两个进程间的物理内存共享接口调用流程）：

1. 在A进程中：
 - a. 调用[aclrtMallocPhysical](#)接口，申请物理内存。
先调用[aclrtMemGetAllocationGranularity](#)接口获取内存申请粒度，然后在调用[aclrtMallocPhysical](#)接口申请物理内存时size按获取到的内存申请粒度对齐，以便节约内存。
若需申请地址连续的虚拟内存、最大化利用物理内存，此处可配合[aclrtReserveMemAddress](#)、[aclrtMapMem](#)等接口申请虚拟内存、建立虚拟内存与物理内存之间的映射，请参见对应接口的说明。
 - b. 调用[aclrtMemExportToShareableHandle](#)接口，导出物理内存handle，输出shareableHandle。

- c. 获取B进程的进程ID，并调用[aclrtMemSetPidToShareableHandle](#)接口，将B进程的进程ID设置为白名单。
 - d. 调用[aclrtFreePhysical](#)接口，释放物理内存。
涉及共享内存的进程都需要释放物理内存，所有涉及内存共享的进程都完成释放操作，物理内存才真正释放。释放物理内存后，引用的分配内存将释放回操作系统，之后使用handle将导致未定义的行为。
2. 在B进程中：
- a. 调用[aclrtDeviceGetBareTgid](#)接口，获取B进程的进程ID。
本接口内部在获取进程ID时已适配物理机、虚拟机场景，用户只需调用本接口获取进程ID，再配置其它接口使用，达到物理内存共享的目的。若用户不调用本接口、自行获取进程ID，可能会导致后续使用进程ID异常。
 - b. 调用[aclrtMemImportFromShareableHandle](#)，获取shareableHandle里的信息，并返回本进程中的handle。
在调用[aclrtMemImportFromShareableHandle](#)接口前，需确保待共享的物理内存存在，不能提前释放。
若需申请地址连续的虚拟内存、最大化利用物理内存地址的目的，此处可配合[aclrtReserveMemAddress](#)、[aclrtMapMem](#)等接口申请虚拟内存、建立虚拟内存与物理内存之间的映射，请参见对应接口的说明。
 - c. 调用[aclrtFreePhysical](#)接口，释放物理内存。

约束说明

- Ascend RC形态不支持调用本接口。
- 只支持Device侧的物理内存共享。
- 支持跨Device共享物理内存。
跨Device共享物理内存仅支持如下项，且需配合[aclrtDeviceEnablePeerAccess](#)接口使用：
Atlas 推理系列产品
Atlas 训练系列产品
Atlas A2训练系列产品/Atlas 800I A2推理产品
- 通过[aclrtFreePhysical](#)接口释放物理内存，实现销毁shareableHandle。若有进程还在使用shareableHandle，则等待shareableHandle使用完成后再执行销毁任务。
- handle与shareableHandle是一一对应的关系，在同一个进程中，不允许一对多、或多对一，否则报错，例如重复调用本接口导出时则会返回报错。
- 不支持算力切分和算力分组场景。

函数原型

aclError [aclrtMemExportToShareableHandle](#)([aclrtDrvMemHandle](#) handle, [aclrtMemHandleType](#) handleType, uint64_t flags, uint64_t *shareableHandle)

参数说明

参数名	输入/输出	说明
handle	输入	存放物理内存信息的handle。 需先在本进程调用aclrtMallocPhysical接口申请物理内存，该接口调用成功，会返回一个handle。
handleType	输入	预留参数，当前固定填ACL_MEM_HANDLE_TYPE_NONE。
flags	输入	预留参数，当前固定填0。
shareableHandle	输出	标识共享给其它进程的shareableHandle。

返回值说明

返回0表示成功，返回其他值表示失败。

1.5.5.26 aclrtDeviceGetBareTgid

函数功能

获取当前进程的进程ID。

本接口内部在获取进程ID时已适配物理机、虚拟机场景，用户只需调用本接口获取进程ID，再配置其它接口使用（配合流程请参见[aclrtMemExportToShareableHandle](#)），达到物理内存共享的目的。若用户不调用本接口、自行获取进程ID，可能会导致后续使用进程ID异常。

函数原型

aclError aclrtDeviceGetBareTgid(int32_t *pid)

参数说明

参数名	输入/输出	说明
pid	输出	进程ID。

返回值说明

返回0表示成功，返回其他值表示失败。

1.5.5.27 aclrtMemSetPidToShareableHandle

函数功能

设置可共享内存的进程白名单，只有配置了白名单的进程才能用[aclrtMemExportToShareableHandle](#)接口导出的shareableHandle。

本接口需与其它接口配合使用，以便实现内存共享的目的，请参见[aclrtMemExportToShareableHandle](#)接口处的说明。

约束说明

在调用[aclrtMemExportToShareableHandle](#)接口的进程中，调用本接口设置进程白名单。

Ascend RC形态不支持调用本接口。

函数原型

aclError [aclrtMemSetPidToShareableHandle](#)(uint64_t shareableHandle, int32_t *pid, size_t pidNum)

参数说明

参数名	输入/输出	说明
shareableHandle	输入	通过 aclrtMemExportToShareableHandle 接口导出的shareableHandle。
pid	输入	用于存放白名单进程ID的数组。 进程ID可调用 aclrtDeviceGetBareTgid 接口获取， Docker场景下获取到的是物理机上的进程ID，非Docker场景下获取到的是进程ID。
pidNum	输入	白名单进程数量，与pid参数数组的大小保持一致。

返回值说明

返回0表示成功，返回其他值表示失败。

1.5.5.28 aclrtMemImportFromShareableHandle

函数功能

在本进程中获取shareableHandle里的信息，并返回本进程中的handle，用于在本进程中建立虚拟地址与物理地址之间的映射关系。本接口还支持生成指定Device上的handle。

本接口需与其它接口配合使用，以便实现内存共享的目的，配合使用流程请参见[aclrtMemExportToShareableHandle](#)接口处的说明。

约束说明

- Ascend RC形态不支持调用本接口。
- 在调用本接口前，需确保待共享的物理内存存在，不能提前释放。
- handle与shareableHandle是一一对应的关系，在同一个进程中，不允许一对多、或多对一，否则报错，例如重复调用本接口导入时则会返回报错。
- 不支持同一个进程中调用[aclrtMemImportFromShareableHandle](#)、[aclrtMemExportToShareableHandle](#)这两个接口，只支持跨进程调用。
- 支持在一个Device上调用[aclrtMemExportToShareableHandle](#)接口导出handle，然后调用本接口生成另一个Device上的handle。
- 若调用[aclrtFreePhysical](#)销毁handle，则需所有调用本接口的进程都销毁shareableHandle的情况下，handle才会正在销毁。
- 支持跨Device共享物理内存。
跨Device共享物理内存仅支持如下项，且需配合[aclrtDeviceEnablePeerAccess](#)接口使用：
Atlas 推理系列产品
Atlas 训练系列产品
Atlas A2训练系列产品/Atlas 800I A2推理产品

函数原型

```
aclError aclrtMemImportFromShareableHandle(uint64_t shareableHandle,
int32_t deviceId, aclrtDrvMemHandle *handle)
```

参数说明

参数名	输入/输出	说明
shareableHandle	输入	待共享的shareableHandle。 与 aclrtMemExportToShareableHandle 接口中导出的shareableHandle保持一致。
deviceId	输入	用于生成指定Device ID上的handle。 用户调用 aclrtGetDeviceCount 接口获取可用的Device数量后，这个Device ID的取值范围：[0, (可用的Device数量-1)]
handle	输出	本进程的物理内存handle。

返回值说明

返回0表示成功，返回其他值表示失败。

1.5.5.29 aclrtMemGetAllocationGranularity

函数功能

查询内存申请粒度。系统内部计算给定分配规范的最小粒度或建议粒度，并以granularity参数返回粒度。此粒度可用作对齐、地址大小或地址映射的倍数。

约束说明

Ascend RC形态不支持调用本接口。

函数原型

aclError aclrtMemGetAllocationGranularity(**aclrtPhysicalMemProp** *prop, **aclrtMemGranularityOptions** option, size_t *granularity)

参数说明

参数名	输入/输出	说明
prop	输入	物理内存属性信息。
option	输入	最小粒度或推荐粒度。
granularity	输出	内存申请粒度，单位为Byte。 当前只支持2M。

返回值说明

返回0表示成功，返回其他值表示失败。

1.5.5.30 aclrtCmoAsync

Atlas 200/300/500 推理产品不支持该接口。

Atlas 训练系列产品不支持该接口。

Atlas 推理系列产品不支持该接口。

Atlas 200/500 A2推理产品不支持该接口。

函数功能

实现Device上的Cache内存操作。

函数原型

aclError aclrtCmoAsync(void *src, size_t size, **aclrtCmoType** cmoType, **aclrtStream** stream)

参数说明

参数名	输入/输出	说明
src	输入	待操作的Device内存的地址。
size	输入	待操作的Device内存大小，单位Byte。
cmoType	输入	Cache内存操作类型。 当前仅支持ACL_RT_CMO_TYPE_PREFETCH（内存预取）。
stream	输入	指定stream。

返回值说明

返回0表示成功，返回其他值表示失败。

1.5.6 执行控制

1.5.6.1 aclrtSubscribeReport

函数功能

异步任务场景下，注册处理Stream上回调函数的线程。

本接口需与以下其它接口配合使用，以便实现异步场景下的callback功能：

1. 定义并实现回调函数，函数原型为：typedef void (*aclrtCallback)(void *userData)；
2. 新建线程，在线程函数内，调用[aclrtProcessReport](#)接口设置超时时间（需循环调用），等待回调任务执行；
3. 调用[aclrtSubscribeReport](#)接口建立第2步中的线程和Stream的绑定关系，该Stream下发的回调函数将在绑定的线程中执行；
4. 在指定Stream上执行异步任务（例如异步推理任务）；
5. 调用[aclrtLaunchCallback](#)接口在Stream的任务队列中下发回调任务，触发第2步中注册的线程处理回调函数，每调用一次[aclrtLaunchCallback](#)接口，就会触发一次回调函数的执行；
6. 异步任务全部执行完成后，取消线程注册（[aclrtUnSubscribeReport](#)接口）。

约束说明

- 支持多次调用[aclrtSubscribeReport](#)接口给多个Stream（仅支持同一Device内的多个Stream）注册同一个处理回调函数的线程；
- 为确保Stream内的任务按调用顺序执行，不支持调用[aclrtSubscribeReport](#)接口给同一个Stream注册多个处理回调函数的线程；
- 在Atlas 200/300/500 推理产品上，单进程内调用[aclrtSubscribeReport](#)接口注册的线程数量如果超过128个，则接口返回失败；
- 在Atlas 训练系列产品上，单进程内调用[aclrtSubscribeReport](#)接口注册的线程数量如果超过1024个，则接口返回失败；

- 在Atlas A2训练系列产品/Atlas 800I A2推理产品上，单进程内调用[aclrtSubscribeReport](#)接口注册的线程数量如果超过1024个，则接口返回失败；
- 在Atlas 推理系列产品上，单进程内调用[aclrtSubscribeReport](#)接口注册的线程数量如果超过1024个，则接口返回失败；
- Atlas 200/500 A2推理产品，单进程内调用[aclrtSubscribeReport](#)接口注册的线程数量如果超过1024个，则接口返回失败；
- 考虑操作系统的线程切换性能开销，建议调用[aclrtSubscribeReport](#)接口注册的线程数量控制在32个以下（包括32）；
- 同一个进程内，在不同的Device上注册回调函数的线程时，不能指定同一个线程ID。

函数原型

[aclError](#) [aclrtSubscribeReport](#)(uint64_t threadId, [aclrtStream](#) stream)

参数说明

参数名	输入/输出	说明
threadId	输入	指定线程的ID。
stream	输入	指定Stream。

返回值说明

返回0表示成功，返回其他值表示失败。

参考资源

接口调用流程及示例代码，参见[异步模型推理](#)。

1.5.6.2 [aclrtLaunchCallback](#)

函数功能

异步任务场景下，在Stream的任务队列中下发一个回调任务，系统内部在执行到该回调任务时，会在Stream上注册的线程（通过[aclrtSubscribeReport](#)接口注册的线程）中执行回调函数。异步接口。

本接口需与以下其它接口配合使用，以便实现异步场景下的callback功能：

1. 定义并实现回调函数，函数原型为：typedef void (*[aclrtCallback](#))(void *userData)；
2. 新建线程，在线程函数内，调用[aclrtProcessReport](#)接口设置超时时间（需循环调用），等待回调任务执行；
3. 调用[aclrtSubscribeReport](#)接口建立第2步中的线程和Stream的绑定关系，该Stream下发的回调函数将在绑定的线程中执行；
4. 在指定Stream上执行异步任务（例如异步推理任务）；

5. 调用[aclrtLaunchCallback](#)接口在Stream的任务队列中下发回调任务，触发第2步中注册的线程处理回调函数，每调用一次[aclrtLaunchCallback](#)接口，就会触发一次回调函数的执行；
6. 异步任务全部执行完成后，取消线程注册（[aclrtUnSubscribeReport](#)接口）。

约束说明

该接口是异步接口，调用接口成功仅表示任务下发成功，不表示任务执行成功。调用该接口后，需调用同步等待接口（例如，[aclrtSynchronizeStream](#)）确保任务已执行完成，否则可能会导致训练或推理等业务异常、Device断链掉卡等未知情况。

函数原型

```
aclError aclrtLaunchCallback(aclrtCallback fn, void *userData,
aclrtCallbackBlockType blockType, aclrtStream stream)
```

参数说明

参数名	输入/输出	说明
fn	输入	指定要增加的回调函数。 回调函数的函数原型为： typedef void (*aclrtCallback)(void *userData)
userData	输入	待传递给回调函数的用户数据的指针。
blockType	输入	指定回调函数调用是否阻塞Device运行。 typedef enum aclrtCallbackBlockType { ACL_CALLBACK_NO_BLOCK, //非阻塞，Atlas 200/300/500 推理产品不支持该选项，该选项预留 ACL_CALLBACK_BLOCK, //阻塞 } aclrtCallbackBlockType;
stream	输入	指定Stream。

返回值说明

返回0表示成功，返回其他值表示失败。

参考资源

接口调用流程及示例代码，参见异步模型推理。

1.5.6.3 aclrtProcessReport

函数功能

异步任务场景下，调用本接口设置超时时间，等待[aclrtLaunchCallback](#)接口下发的回调任务执行。

本接口需与以下其它接口配合使用，以便实现异步场景下的callback功能：

1. 定义并实现回调函数，函数原型为：typedef void (*aclrtCallback)(void *userData);
2. 新建线程，在线程函数内，调用[aclrtProcessReport](#)接口设置超时时间（需循环调用），等待回调任务执行；
3. 调用[aclrtSubscribeReport](#)接口建立第2步中的线程和Stream的绑定关系，该Stream下发的回调函数将在绑定的线程中执行；
4. 在指定Stream上执行异步任务（例如异步推理任务）；
5. 调用[aclrtLaunchCallback](#)接口在Stream的任务队列中下发回调任务，触发第2步中注册的线程处理回调函数，每调用一次[aclrtLaunchCallback](#)接口，就会触发一次回调函数的执行；
6. 异步任务全部执行完成后，取消线程注册（[aclrtUnSubscribeReport](#)接口）。

函数原型

```
aclError aclrtProcessReport(int32_t timeout)
```

参数说明

参数名	输入/输出	说明
timeout	输入	超时时间，单位为ms。 取值范围： • -1：表示无限等待 • 大于0（不包含0）：表示等待的时间

返回值说明

返回0表示成功，返回其他值表示失败。

参考资源

接口调用流程及示例代码，参见异步模型推理。

1.5.6.4 aclrtUnSubscribeReport

函数功能

异步任务场景下，取消线程注册，Stream上的回调函数不再由指定线程处理。

本接口需与以下其它接口配合使用，以便实现异步场景下的callback功能：

1. 定义并实现回调函数，函数原型为：typedef void (*aclrtCallback)(void *userData);
2. 新建线程，在线程函数内，调用[aclrtProcessReport](#)接口设置超时时间（需循环调用），等待回调任务执行；
3. 调用[aclrtSubscribeReport](#)接口建立第2步中的线程和Stream的绑定关系，该Stream下发的回调函数将在绑定的线程中执行；

- 4. 在指定Stream上执行异步任务（例如异步推理任务）；
- 5. 调用[aclrtLaunchCallback](#)接口在Stream的任务队列中下发回调任务，触发第2步中注册的线程处理回调函数，每调用一次[aclrtLaunchCallback](#)接口，就会触发一次回调函数的执行；
- 6. 异步任务全部执行完成后，取消线程注册（[aclrtUnSubscribeReport](#)接口）。

函数原型

aclError [aclrtUnSubscribeReport](#)(uint64_t threadId, [aclrtStream](#) stream)

参数说明

参数名	输入/输出	说明
threadId	输入	指定线程的ID。
stream	输入	指定Stream。

返回值说明

返回0表示成功，返回其他值表示失败。

参考资源

接口调用流程及示例代码，参见异步模型推理。

1.5.6.5 [aclrtSubscribeHostFunc](#)

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

Atlas 200/500 A2推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

模型中有CPU算子、且调用[aclmdlExecuteV2](#)或[aclmdlExecuteAsyncV2](#)接口执行模型推理时，调用本接口注册处理Stream上回调函数的线程（线程需由用户自行创建），再配合调用[aclrtProcessHostFunc](#)接口触发回调函数、在模型执行之后调用[aclrtUnSubscribeHostFunc](#)接口取消注册。

约束说明

- 支持多次调用[aclrtSubscribeHostFunc](#)接口给多个Stream（仅支持同一Device内的多个Stream）注册同一个处理回调函数的线程。
- 为确保Stream内的任务按调用顺序执行，不支持调用[aclrtSubscribeHostFunc](#)接口给同一个Stream注册多个处理回调函数的线程。

函数原型

aclError aclrtSubscribeHostFunc(uint64_t hostFuncThreadId, **aclrtStream** exeStream)

参数说明

参数名	输入/输出	说明
hostFuncThreadId	输入	指定线程的ID。
exeStream	输入	指定Stream。

返回值说明

返回0表示成功，返回其他值表示失败。

1.5.6.6 aclrtProcessHostFunc

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

Atlas 200/500 A2推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

等待指定时间后，触发回调处理，由**aclrtSubscribeHostFunc**接口指定的线程处理回调。

线程需由用户提前自行创建，并自定义线程函数，在线程函数内调用本接口，等待指定时间后通过系统内部进行算子计算。

函数原型

aclError aclrtProcessHostFunc(int32_t timeout)

参数说明

参数名	输入/输出	说明
timeout	输入	超时时间，单位为ms。 取值范围： -1：表示无限等待 - 大于0（不包含0）：表示等待的时间

返回值说明

返回0表示成功，返回其他值表示失败。

1.5.6.7 aclrtUnSubscribeHostFunc

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

Atlas 200/500 A2推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

与[aclrtSubscribeHostFunc](#)接口配合使用，调用模型执行接口后，调用本接口取消线程注册，Stream上的回调函数不再由指定线程处理。

函数原型

aclError aclrtUnSubscribeHostFunc(uint64_t hostFuncThreadId, **aclrtStream** exeStream)

参数说明

参数名	输入/输出	说明
hostFuncThreadId	输入	指定线程的ID。
exeStream	输入	指定Stream。

返回值说明

返回0表示成功，返回其他值表示失败。

1.5.6.8 aclrtSetOpExecuteTimeOut

Atlas 200/300/500 推理产品，不支持该接口。

函数功能

设置算子执行的超时时间。

约束说明

- 不调用本接口，算子执行的默认超时时间如下：

版本	AI Core算子的默认超时时间	AI CPU算子的默认超时时间
Atlas 推理系列产品	547秒	28秒
Atlas 训练系列产品	68秒	28秒
Atlas A2训练系列产品/ Atlas 800I A2推理产品	1091秒	28秒
Atlas 200/500 A2推理产品	1091秒	28秒

- 一个进程内多次调用本接口，则以最后一次设置的时间为准，允许设置的最大时间如下：

版本	AI Core算子的最大超时时间	AI CPU算子的最大超时时间
Atlas 推理系列产品	547秒	不支持设置
Atlas 训练系列产品	2176秒	2176秒
Atlas A2训练系列产品/ Atlas 800I A2推理产品	1091秒	1091秒
Atlas 200/500 A2推理产品	1091秒	1091秒

函数原型

aclError aclrtSetOpExecuteTimeOut(uint32_t timeout)

参数说明

参数名	输入/输出	说明
timeout	输入	设置超时时间，单位为秒。 将该参数设置为0时，表示使用默认超时时间。

返回值说明

返回0表示成功，返回其他值表示失败。

1.5.7 异常处理

1.5.7.1 aclrtSetExceptionInfoCallback

函数功能

设置异常回调函数。

约束说明

- 您需要在执行异步任务之前，设置异常回调函数，当Device上的任务执行异常时，系统会向用户设置的异常回调函数中传入一个包含任务ID、Stream ID、线程ID、Device ID以及错误码的[aclrtExceptionInfo](#)结构体指针，并执行回调函数，用户可以再分别调用[aclrtGetTaskIdFromExceptionInfo](#)、[aclrtGetStreamIdFromExceptionInfo](#)、[aclrtGetThreadIdFromExceptionInfo](#)、[aclrtGetDeviceIdFromExceptionInfo](#)、[aclrtGetErrorCodeFromExceptionInfo](#)接口获取产生异常的任务ID、Stream ID、线程ID、Device ID以及错误码，便于定位问题。
使用场景举例：例如，在调用[aclrtExecuteV2](#)接口前，调用[aclrtSetExceptionInfoCallback](#)接口设置异常回调函数，当算子在Device执行异常时，系统会向用户设置的异常回调函数中传入一个包含任务ID、Stream ID、线程ID、Device ID以及错误码的[aclrtExceptionInfo](#)结构体指针，并执行回调函数。
- 如果多次设置异常回调函数，以最后一次设置为准。
- 如果想清空回调函数，可调用[aclrtSetExceptionInfoCallback](#)接口，将入参设置为空指针。

函数原型

aclError [aclrtSetExceptionInfoCallback](#)([aclrtExceptionInfoCallback](#) callback)

参数说明

参数名	输入/输出	说明
callback	输入	指定要注册的回调函数。 回调函数的函数原型为： typedef void (*aclrtExceptionInfoCallback) (aclrtExceptionInfo *exceptionInfo);

返回值说明

返回0表示成功，返回其他值表示失败。

参考资源

接口调用示例，参见AI Core异常信息获取。

1.5.7.2 aclrtGetTaskIdFromExceptionInfo

函数功能

获取异常信息中的任务ID。该接口与[aclrtSetExceptionInfoCallback](#)接口配合使用。

函数原型

`uint32_t aclrtGetTaskIdFromExceptionInfo(const aclrtExceptionInfo *info)`

参数说明

参数名	输入/输出	说明
info	输入	异常信息的指针。 在执行任务之前调用 aclrtSetExceptionInfoCallback 接口，系统会将产生异常的任务ID、Stream ID、线程ID、Device ID存放在aclExceptionInfo结构体中。

返回值说明

返回异常信息中的任务ID，返回值为0xFFFFFFFF（以十六进制为例）时表示Device异常。

参考资源

接口调用示例，参见AI Core异常信息获取。

1.5.7.3 aclrtGetStreamIdFromExceptionInfo

函数功能

获取异常信息中的Stream ID。该接口与[aclrtSetExceptionInfoCallback](#)接口配合使用。

函数原型

`uint32_t aclrtGetStreamIdFromExceptionInfo(const aclrtExceptionInfo *info)`

参数说明

参数名	输入/输出	说明
info	输入	异常信息的指针。 在执行任务之前调用 aclrtSetExceptionInfoCallback 接口，系统会将产生异常的任务ID、Stream ID、线程ID、Device ID存放在aclExceptionInfo结构体中。

返回值说明

返回异常信息中的Stream ID，返回值为0xFFFFFFFF（以十六进制为例）时表示Device异常。

参考资源

接口调用示例，参见AI Core异常信息获取。

1.5.7.4 aclrtGetThreadIdFromExceptionInfo

函数功能

获取异常信息中的线程ID。该接口与[aclrtSetExceptionInfoCallback](#)接口配合使用。

函数原型

uint32_t aclrtGetThreadIdFromExceptionInfo(const aclrtExceptionInfo *info)

参数说明

参数名	输入/输出	说明
info	输入	异常信息的指针。 在执行任务之前调用 aclrtSetExceptionInfoCallback 接口，系统会将产生异常的任务ID、Stream ID、线程ID、Device ID存放在aclExceptionInfo结构体中。

返回值说明

返回异常信息中的线程ID，返回值为0xFFFFFFFF（以十六进制为例）时表示Device异常。

1.5.7.5 aclrtGetDeviceIdFromExceptionInfo

函数功能

获取异常信息中的Device ID。该接口与[aclrtSetExceptionInfoCallback](#)接口配合使用。

函数原型

uint32_t aclrtGetDeviceIdFromExceptionInfo(const aclrtExceptionInfo *info)

参数说明

参数名	输入/输出	说明
info	输入	异常信息的指针。 在执行任务之前调用 aclrtSetExceptionInfoCallback 接口，系统会将产生异常的任务ID、Stream ID、线程ID、Device ID存放在aclExceptionInfo结构体中。

返回值说明

返回异常信息中的Device ID，返回值为0xFFFFFFFF（以十六进制为例）时表示Device异常。

参考资源

接口调用示例，参见AI Core异常信息获取。

1.5.7.6 aclrtGetErrorCodeFromExceptionInfo

函数功能

获取异常信息中的错误码。该接口与[aclrtSetExceptionInfoCallback](#)接口配合使用。

函数原型

uint32_t aclrtGetErrorCodeFromExceptionInfo(const aclrtExceptionInfo *info)

参数说明

参数名	输入/输出	说明
info	输入	异常信息的指针。 在执行任务之前调用 aclrtSetExceptionInfoCallback 接口，系统会将产生异常的任务ID、Stream ID、线程ID、Device ID、错误码存放在aclExceptionInfo结构体中。

返回值说明

返回异常信息中的错误码，返回值为0xFFFFFFFF（以十六进制为例）时表示Device异常。

1.5.7.7 aclrtGetMemUcelInfo

须知

本接口为预留接口，暂不支持。

函数功能

获取内存UCE（uncorrect error，指系统硬件不能直接处理恢复内存错误）的错误虚拟地址。

函数原型

aclError **aclrtGetMemUcelInfo**(int32_t deviceId, **aclrtMemUcelInfo** *memUcelInfoArray, size_t arraySize, size_t *retSize)

参数说明

参数名	输入/输出	说明
deviceId	输入	Device ID。 与 aclrtSetDevice 接口中Device ID保持一致。
memUcelInfoArray	输入&输出	aclrtMemUcelInfo数组的指针。
arraySize	输入	传入aclrtMemUcelInfo数组的长度。
retSize	输出	实际返回的aclrtMemUcelInfo数组的有效长度。

返回值说明

返回0表示成功，返回非0表示失败。

1.5.7.8 aclrtDeviceTaskAbort

须知

本接口为预留接口，暂不支持。

函数功能

停止指定Device上的正在执行的任务，同时丢弃指定Device上已下发的任务。该接口支持用户设置永久等待、或配置具体的超时时间，若配置具体的超时时间，则调用本接口超出超时时间，则接口返回报错。

函数原型

```
aclError aclrtDeviceTaskAbort(int32_t deviceId, uint32_t timeout);
```

参数说明

参数名	输入/输出	说明
deviceId	输入	Device ID。 与 aclrtSetDevice 接口中Device ID保持一致。
timeout	输入	超时时间。 取值说明如下： 0：表示永久等待； >0：配置具体的超时时间，单位是毫秒。最大超时时间36分钟。

返回值说明

返回0表示成功，返回非0表示失败。

1.5.7.9 aclrtMemUceRepair

须知

本接口为预留接口，暂不支持。

函数功能

修复内存UCE的错误虚拟地址。

函数原型

```
aclError aclrtMemUceRepair(int32_t deviceId, aclrtMemUceInfo *memUceInfoArray, size_t arraySize)
```

参数说明

参数名	输入/输出	说明
deviceId	输入	Device ID。 与 aclrtSetDevice 接口中Device ID保持一致。
memUceInfoArray	输入	aclrtMemUceInfo数组的指针。

参数名	输入/输出	说明
arraySize	输入	传入aclrtMemUceInfo数组的长度。

返回值说明

返回0表示成功，返回非0表示失败。

1.5.7.10 aclrtPeekAtLastError

函数功能

获取当前线程的Runtime（运行时管理模块）错误码，仅获取但不清空错误码。

函数原型

aclError aclrtPeekAtLastError(**aclrtLastErrLevel** level)

参数说明

参数名	输入/输出	说明
level	输入	指定获取错误码的级别，当前仅支持线程级别。

返回值说明

返回0表示成功，返回非0表示失败。

1.5.7.11 aclrtGetLastError

函数功能

获取当前线程的Runtime（运行时管理模块）错误码，获取后清空当前线程的错误码，这时在线程中无新增错误码之前，调用本接口获取到的是表示成功的返回码ACL_SUCCESS。

函数原型

aclError aclrtGetLastError(**aclrtLastErrLevel** level)

参数说明

参数名	输入/输出	说明
level	输入	指定获取错误码的级别，当前仅支持线程级别。

返回值说明

返回0表示成功，返回非0表示失败。

1.5.8 Kernel 加载与执行

1.5.8.1 aclrtCreateBinary

函数功能

创建aclrtBinary类型的数据，该数据类型用于描述算子二进制信息。仅支持Ascend C自定义算子。

如需销毁aclrtBinary类型的数据，请参见[aclrtDestroyBinary](#)。

函数原型

aclrtBinary **aclrtCreateBinary**(const void *data, size_t dataLen)

参数说明

参数名	输入/输出	说明
data	输入	存放算子二进制文件（*.o文件）数据的内存地址指针。应用运行在Host时，此处需申请Host上的内存；应用运行在Device时，此处需申请Device上的内存。内存申请接口请参见 1.5.5 内存管理 。
dataLen	输入	内存大小，单位Byte。

返回值说明

返回aclrtBinary类型的指针。

1.5.8.2 aclrtDestroyBinary

函数功能

销毁通过[aclrtCreateBinary](#)接口创建的aclrtBinary类型的数据。仅支持Ascend C自定义算子。

注意，此处仅销毁aclrtBinary的数据，调用aclrtBinaryCreate接口时传入的data内存需由用户自行、及时释放，否则可能会导致内存异常。

函数原型

aclError aclrtDestroyBinary(aclrtBinary binary)

参数说明

参数名	输入/输出	说明
binary	输入	待销毁的aclrtBinary类型的指针。

返回值说明

返回0表示成功，返回非0表示失败。

1.5.8.3 aclrtBinaryLoad

函数功能

解析、加载算子二进制文件，输出指向算子二进制的binHandle，同时将算子二进制文件数据拷贝至当前Context对应的Device上。仅支持Ascend C自定义算子。

函数原型

aclError aclrtBinaryLoad(const aclrtBinary binary, aclrtBinHandle *binHandle)

参数说明

参数名	输入/输出	说明
binary	输入	算子二进制信息。 此处需先调用aclrtBinaryCreate接口，获取aclrtBinary类型数据的指针。
binHandle	输出	指向二进制的handle。 typedef void* aclrtBinHandle

返回值说明

返回0表示成功，返回其他值表示失败。

1.5.8.4 aclrtBinaryUnload

函数功能

删除binHandle指向的算子二进制数据，同时也删除通过aclrtBinaryLoad接口拷贝到Device上的算子二进制数据。仅支持Ascend C自定义算子。

约束说明

调用本接口删除算子二进制数据时，需跟aclrtBinaryLoad接口在同一个Context下，这样才能一并删除通过aclrtBinaryLoad接口拷贝到Device上的算子二进制数据，否则可能会导致Device上的算子二进制数据删除异常。

函数原型

aclError aclrtBinaryUnload(aclrtBinHandle binHandle)

参数说明

参数名	输入/输出	说明
binHandle	输入	指向算子二进制的handle。 该handle在调用aclrtBinaryLoad接口时生成。

返回值说明

返回0表示成功，返回其他值表示失败。

1.5.8.5 aclrtBinaryGetFunction

函数功能

根据kernelName，查找对应的kernel对象，使用funcHandle表达。仅支持Ascend C自定义算子。

函数原型

aclError aclrtBinaryGetFunction(const aclrtBinHandle binHandle, const char *kernelName, aclrtFuncHandle *funcHandle)

参数说明

参数名	输入/输出	说明
binHandle	输入	指向算子二进制的handle。 调用 aclrtBinaryLoad 接口获取binHandle。

参数名	输入/输出	说明
kernelName	输入	Kernel名称。
funcHandle	输出	标识指定Kernel的funcHandle表达。 typedef void* aclrtFuncHandle;

返回值说明

返回0表示成功，返回其他值表示失败。

1.5.8.6 aclrtLaunchKernel

函数功能

启动对应Kernel的计算任务。仅支持Ascend C自定义算子。

约束说明

该接口是异步接口，调用接口成功仅表示任务下发成功，不表示任务执行成功。调用该接口后，需调用同步等待接口（例如，[aclrtSynchronizeStream](#)）确保任务已执行完成，否则可能会导致训练或推理等业务异常、Device断链掉卡等未知情况。

函数原型

aclError [aclrtLaunchKernel](#)(aclrtFuncHandle funcHandle, uint32_t blockDim, const void *argsData, size_t argsSize, aclrtStream stream)

参数说明

参数名	输入/输出	说明
funcHandle	输入	调用 aclrtBinaryGetFunction 接口根据kernelName获取funcHandle。
blockDim	输入	指定核函数将会在几个核上执行。
argsData	输入	存放Kernel所有入参数据的Device内存地址指针。 内存申请接口请参见 1.5.5 内存管理 。
argsSize	输入	argsData参数值的大小，单位为Byte。
stream	输入	指定执行任务的Stream，可复用已创建的Stream节省资源或调用 aclrtCreateStream 接口创建Stream，再作为入参在此处传入。

返回值说明

返回0表示成功，返回其他值表示失败。

1.5.9 算力 Group 查询与设置

1.5.9.1 aclrtSetGroup

- Atlas 200/300/500 推理产品，不支持该接口。
- Atlas 200/500 A2推理产品，不支持该接口。
- Atlas 训练系列产品，不支持该接口。
- Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

指定当前运算使用哪个Group的算力，该接口必须在指定Context后调用。

约束说明

- Atlas 推理系列产品，Ascend EP标准形态下，不支持调用本接口。
- 调用DSMI提供的dsmi_create_capability_group接口创建分组后，再调用AscendCL接口进行算力Group查询与设置。

AscendCL接口调用顺序：调用[aclrtSetDevice](#)接口指定计算设备-->调用[aclrtGetAllGroupInfo](#)接口获取所有Group信息-->调用[aclrtGetGroupCount](#)接口获取Group数量-->调用[aclrtGetGroupInfoDetail](#)接口获取指定Group信息-->调用[aclrtSetGroup](#)接口设置分组-->执行其它任务-->调用[aclrtResetDevice](#)接口释放计算设备。

说明

若刷新分组（例如调用dcmi_create_capability_group接口新增分组、调用dcmi_delete_capability_group接口删除分组等），需重启业务AscendCL应用进程。

函数原型

[aclError](#) [aclrtSetGroup](#)(int32_t groupId)

参数说明

参数名	输入/输出	说明
groupId	输入	表示Group的ID，用于指定当前计算要使用的Group。您需要提前调用 aclrtGetGroupInfoDetail 接口获取Group的ID。

返回值说明

返回0表示成功，返回其他值表示失败。

1.5.9.2 aclrtGetGroupCount

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 200/500 A2推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

查询当前Context下可以使用的Group个数。

约束说明

- Atlas 推理系列产品，Ascend EP标准形态下，不支持调用本接口。
- 调用DSMI提供的dsmi_create_capability_group接口创建分组后，再调用AscendCL接口进行算力Group查询与设置。
AscendCL接口调用顺序：调用[aclrtSetDevice](#)接口指定计算设备-->调用[aclrtGetAllGroupInfo](#)接口获取所有Group信息-->调用[aclrtGetGroupCount](#)接口获取Group数量-->调用[aclrtGetGroupInfoDetail](#)接口获取指定Group信息-->调用[aclrtSetGroup](#)接口设置分组-->执行其它任务-->调用[aclrtResetDevice](#)接口释放计算设备。

说明

若刷新分组（例如调用dcmi_create_capability_group接口新增分组、调用dcmi_delete_capability_group接口删除分组等），需重启业务AscendCL应用进程。

函数原型

aclError [aclrtGetGroupCount](#)(uint32_t *count)

参数说明

参数名	输入/输出	说明
count	输出	当前Context下可用Group个数的指针。

返回值说明

返回0表示成功，返回其他值表示失败。

1.5.9.3 aclrtGetAllGroupInfo

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 200/500 A2推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

查询当前Context下可以使用的所有Group的详细算力信息。

约束说明

- Atlas 推理系列产品，Ascend EP标准形态下，不支持调用本接口。
- 调用DSMI提供的dsmi_create_capability_group接口创建分组后，再调用AscendCL接口进行算力Group查询与设置。
AscendCL接口调用顺序：调用[aclrtSetDevice](#)接口指定计算设备-->调用[aclrtGetAllGroupInfo](#)接口获取所有Group信息-->调用[aclrtGetGroupCount](#)接口获取Group数量-->调用[aclrtGetGroupInfoDetail](#)接口获取指定Group信息-->调用[aclrtSetGroup](#)接口设置分组-->执行其它任务-->调用[aclrtResetDevice](#)接口释放计算设备。

说明

若刷新分组（例如调用dcmi_create_capability_group接口新增分组、调用dcmi_delete_capability_group接口删除分组等），需重启业务AscendCL应用进程。

函数原型

aclError [aclrtGetAllGroupInfo](#)([aclrtGroupInfo](#) *groupInfo)

参数说明

参数名	输入/输出	说明
groupInfo	输出	获取所有Group对应的详细算力信息的指针。 需提前调用 aclrtCreateGroupInfo 接口创建 aclrtGroupInfo 类型的数据。

返回值说明

返回0表示成功，返回其他值表示失败。

1.5.9.4 [aclrtGetGroupInfoDetail](#)

- Atlas 200/300/500 推理产品，不支持该接口。
- Atlas 200/500 A2推理产品，不支持该接口。
- Atlas 训练系列产品，不支持该接口。
- Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

查询当前Context下指定Group的算力信息。

约束说明

- Atlas 推理系列产品，Ascend EP标准形态下，不支持调用本接口。
- 调用DSMI提供的dsmi_create_capability_group接口创建分组后，再调用AscendCL接口进行算力Group查询与设置。
AscendCL接口调用顺序：调用[aclrtSetDevice](#)接口指定计算设备-->调用[aclrtGetAllGroupInfo](#)接口获取所有Group信息-->调用[aclrtGetGroupCount](#)接口获取Group数量-->调用[aclrtGetGroupInfoDetail](#)接口获取指定Group信息-->调用[aclrtSetGroup](#)接口设置分组-->执行其它任务-->调用[aclrtResetDevice](#)接口释放计算设备。

说明

若刷新分组（例如调用dcmi_create_capability_group接口新增分组、调用dcmi_delete_capability_group接口删除分组等），需重启业务AscendCL应用进程。

函数原型

```
aclError aclrtGetGroupInfoDetail(const aclrtGroupInfo *groupInfo, int32_t groupIndex, aclrtGroupAttr attr, void *attrValue, size_t valueLen, size_t *paramRetSize)
```

参数说明

参数名	输入/输出	说明
groupInfo	输入	指定算力详细信息的首地址的指针。 需提前调用 aclrtGetAllGroupInfo 接口创获取所有Group的算力信息。
groupIndex	输入	访问groupInfo连续内存块的Group索引。 Group索引的取值范围：[0, (Group数量-1)]，用户可调用 aclrtGetGroupCount 接口获取Group数量。
attr	输入	指定要获取其算力值的算力属性。
attrValue	输出	获取指定算力属性所对应的算力值的指针。 用户需根据每个属性的属性值数据类型申请对应大小的内存，用于存放属性值。
valueLen	输入	表示attrValue的最大长度，单位为Byte。
paramRetSize	输出	实际返回的attrValue大小的指针，单位为Byte。

返回值说明

返回0表示成功，返回其他值表示失败。

1.5.9.5 aclrtCreateGroupInfo

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 200/500 A2推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

根据实际支持的Group数量创建[aclrtGroupInfo](#)类型的连续内存块，并返回对应指针。

如需销毁[aclrtGroupInfo](#)类型的数据，请参见[aclrtDestroyGroupInfo](#)。

约束说明

Atlas 推理系列产品，Ascend EP标准形态下，不支持调用本接口。

函数原型

aclrtGroupInfo *aclrtCreateGroupInfo()

参数说明

无

返回值说明

返回[aclrtGroupInfo](#)类型的指针，如果无Group或不支持Group则返回nullptr。

1.5.9.6 aclrtDestroyGroupInfo

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 200/500 A2推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

销毁[aclrtGroupInfo](#)类型的数据，释放相关的内存。只能销毁通过[aclrtCreateGroupInfo](#)接口创建的[aclrtGroupInfo](#)类型。

约束说明

Atlas 推理系列产品，Ascend EP标准形态下，不支持调用本接口。

函数原型

aclError aclrtDestroyGroupInfo(aclrtGroupInfo *groupInfo)

参数说明

参数名	输入/输出	说明
groupInfo	输入	待销毁的aclrtGroupInfo类型数据的指针。

返回值说明

返回0表示成功，非零表示失败。

1.6 单算子调用

1.6.1 单算子 API 执行

须知

- 除本章中**单算子API执行**方式调用单算子，AscendCL还提供了基于**单算子模型执行**方式的单算子调用方式，详细请参见单算子调用方式：单算子模型执行、单算子API执行。
- 对于接口文档中未申明支持的场景（如产品型号、数据类型、数据格式等），不推荐用户使用，当前版本暂不保证其调用效果。

本章主要介绍如[表1-2](#)所示的算子API，提供了详细的接口介绍，包括接口原型、功能描述、参数说明、调用示例等，指导用户快速上手调用算子API。

算子接口的**调用流程**请参见单算子API执行接口调用流程，其**调用示例**（以NN算子调用为例）请参见调用NN算子接口示例代码。

表 1-2 单算子 API 执行接口列表

接口分类	说明	接口获取	产品支持度
NN算子	Neural Network算子，覆盖TensorFlow、Pytorch、MindSpore、ONNX等常见框架的深度学习算法计算类型，包括Softmax、MatMul、Convolution等典型计算，接口前缀为aclnnXxx。	参考《 算子加速库接口参考 》中“NN算子接口”章节。	Atlas 200/300/500 推理产品不支持该接口。

接口分类	说明	接口获取	产品支持度
融合算子	提供了Flash Attention、MC2等高性能融合算子，通过将多个独立“小算子”（如向量Vector、矩阵Cube）融合成一个“大算子”，达到提升算子性能的目的，接口前缀为aclnnXxx。	参考《 算子加速库接口参考 》中“融合算子接口”章节。	Atlas 200/300/500 推理产品不支持该接口。 Atlas 200/500 A2 推理产品不支持该接口。 Atlas 训练系列产品不支持该接口。
DVPP算子	Digital Vision Pre-Processing算子，主要实现图像缩放、JPEG图像解码、图像归一化等功能，接口前缀为acldvppXxx。	参考《 算子加速库接口参考 》中“DVPP算子接口”章节。	Atlas 200/300/500 推理产品不支持该接口。 Atlas 200/500 A2 推理产品不支持该接口。 Atlas 训练系列产品不支持该接口。 Atlas 推理系列产品不支持该接口。

1.6.2 单算子模型执行

1.6.2.1 aclopSetModelDir

函数功能

设置加载模型文件的目录，该模型文件是由单算子编译成的*.om文件。

约束说明

- 动态Shape算子场景下，模型文件加载环境中的算子库包安装版本需与模型文件编译环境的版本一致，否则在加载算子时会报错。
- 昇腾虚拟化实例场景下，模型文件加载环境中的算子库包安装版本需与模型文件编译环境的版本一致，否则在加载算子时会报错。
- 在加载前，请先根据单算子om文件的大小评估内存空间是否足够，内存空间不足，会导致应用程序异常。

Atlas 200/300/500 推理产品，在Host上运行应用时，一个进程内正在执行的算子的最大个数上限是40000000，在Device上运行应用时，一个进程内正在执行的算子的最大个数上限是2000000。

Atlas 200/500 A2推理产品，在Host上运行应用时，一个进程内正在执行的算子的最大个数上限是40000000。

Atlas 训练系列产品，在Host上运行应用时，一个进程内正在执行的算子的最大个数上限是400000000。

Atlas A2训练系列产品/Atlas 800I A2推理产品，在Host上运行应用时，一个进程内正在执行的算子的最大个数上限是400000000。

Atlas 推理系列产品，在Host上运行应用时，一个进程内正在执行的算子的最大个数上限是400000000，在Device上运行应用时，一个进程内正在执行的算子的最大个数上限是20000000。

函数原型

aclError **aclopSetModelDir**(const char *modelDir)

参数说明

参数名	输入/输出	说明
modelDir	输入	模型文件路径的指针。 此处可设置多级目录，但系统最多从多级目录的最后一级开始，读取三级目录下的模型文件。 例如，将modelDir设置为"dir0/dir1"，dir1下的目录层级为“dir2/dir3/dir4”，这时系统只支持读取“dir1”、“dir1/dir2”、“dir1/dir2/dir3”目录下的模型文件。

返回值说明

返回0表示成功，返回其他值表示失败。

参考资源

- 接口调用流程参见单算子调用流程。
- 接口调用示例请参见执行固定Shape算子示例代码。

1.6.2.2 aclopLoad

函数功能

从内存中加载单算子模型数据（单算子模型数据是指“单算子编译成*.om文件后，再将om文件读取到内存中”的数据），由用户管理内存。

约束说明

- 动态Shape算子场景下或者昇腾虚拟化实例场景下，单算子模型数据加载环境中的算子库包安装版本需与om模型文件编译环境的版本一致，否则在加载算子时会报错。
- 在加载前，请先根据单算子om文件的大小评估内存空间是否足够，内存空间不足，会导致应用程序异常。

Atlas 200/300/500 推理产品，在Host上运行应用时，一个进程内正在执行的算子的最大个数上限是40000000，在Device上运行应用时，一个进程内正在执行的算子的最大个数上限是2000000。

Atlas 200/500 A2推理产品，在Host上运行应用时，一个进程内正在执行的算子的最大个数上限是40000000。

Atlas 训练系列产品，在Host上运行应用时，一个进程内正在执行的算子的最大个数上限是40000000。

Atlas A2训练系列产品/Atlas 800I A2推理产品，在Host上运行应用时，一个进程内正在执行的算子的最大个数上限是2000000。

Atlas 推理系列产品，在Host上运行应用时，一个进程内正在执行的算子的最大个数上限是40000000，在Device上运行应用时，一个进程内正在执行的算子的最大个数上限是2000000。

函数原型

```
aclError aclopLoad(const void *model, size_t modelSize)
```

参数说明

参数名	输入/输出	说明
model	输入	单算子模型数据的内存地址指针。
modelSize	输入	内存中的模型数据长度，单位Byte。

返回值说明

返回0表示成功，返回其他值表示失败。

参考资源

接口调用流程参见单算子调用流程。

1.6.2.3 aclopSetCompileFlag (废弃)

须知

此接口后续版本会废弃，请使用[aclSetCompileopt](#)接口设置编译选项。

函数功能

在编译算子前调用本接口设置算子编译标识，该设置进程级全局共享。

函数原型

```
aclError aclopSetCompileFlag(aclOpCompileFlag flag)
```

参数说明

参数名	输入/输出	说明
flag	输入	用于设置算子编译标识。 如果没有设置flag，则默认按照精确编译。

返回值说明

返回0表示成功，返回其他值表示失败。

1.6.2.4 aclopCompile

函数功能

编译指定算子。在编译算子前，可以调用[aclSetCompileopt](#)接口设置编译选项。

约束说明

- 每个算子的输入、输出组织不同，需要应用在调用时严格按照算子输入、输出参数来组织算子，用户在调用aclopCompile时，AscendCL根据optype、输入tensor的描述、输出tensor的描述、attr等信息查找对应的任务，再编译算子。
- 编译动态Shape的算子时，如果Shape具体值不明确，但Shape范围明确，则在调用[aclCreateTensorDesc](#)接口创建[aclTensorDesc](#)类型时，需要将dims数组的表示动态维度的元素值设置为-1，再调用[aclSetTensorShapeRange](#)接口设置tensor的各个维度的取值范围；如果Shape具体值以及Shape范围都不明确（该场景预留），则在调用[aclCreateTensorDesc](#)接口创建[aclTensorDesc](#)类型时，需要将dims数组的值设置为-2，例如：int64_t dims[] = {-2}。
- 编译有可选输入的算子时，如果可选输入不使用，则需按此种方式创建[aclTensorDesc](#)类型的数据：[aclCreateTensorDesc](#)(ACL_DT_UNDEFINED, 0, nullptr, ACL_FORMAT_UNDEFINED)，表示数据类型设置为ACL_DT_UNDEFINED，Format设置为ACL_FORMAT_UNDEFINED，Shape信息为nullptr。
- 编译有constant输入的算子时，需要先调用[aclSetTensorConst](#)接口设置constant输入。调用aclopCompile接口、[aclopExecuteV2](#)接口前，设置的constant输入数据必须保持一致。

函数原型

```
aclError aclopCompile(const char *opType,  
int numInputs,  
const aclTensorDesc *const inputDesc[],  
int numOutputs,  
const aclTensorDesc *const outputDesc[],  
const aclopAttr *attr,
```

```
aclopEngineType engineType,  
aclopCompileType compileFlag,  
const char* opPath)
```

参数说明

参数名	输入/输出	说明
opType	输入	算子类型名称的指针。
numInputs	输入	算子输入tensor的数量。
inputDesc	输入	算子输入tensor的描述的指针数组。 需提前调用 aclCreateTensorDesc 接口创建aclTensorDesc类型。 inputDesc数组中的元素个数必须与numInputs参数值保持一致。
numOutputs	输入	算子输出tensor的数量。
outputDesc	输入	算子输出tensor的描述的指针数组。 需提前调用 aclCreateTensorDesc 接口创建aclTensorDesc类型。 outputDesc数组中的元素个数必须与numOutputs参数值保持一致。
attr	输入	算子属性的指针。 需提前调用 aclopCreateAttr 接口创建aclopAttr类型。
engineType	输入	算子执行引擎。
compileFlag	输入	算子编译标识。
opPath	输入	算子实现文件（*.py）的路径的指针，不包含文件名。预留参数，当前仅支持设置为nullptr。

返回值说明

返回0表示成功，返回其他值表示失败。

1.6.2.5 aclopExecute（废弃）

须知

此接口后续版本会废弃，请使用[aclopExecuteV2](#)接口。

函数功能

执行指定的算子，异步接口。

约束说明

- 该接口是异步接口，调用接口成功仅表示任务下发成功，不表示任务执行成功。调用该接口后，需调用同步等待接口（例如，[aclrtSynchronizeStream](#)）确保任务已执行完成，否则可能会导致训练或推理等业务异常、Device断链掉卡等未知情况。
- 多线程场景下，不支持调用本接口时指定同一个Stream或使用默认Stream，否则可能任务执行异常。
- 每个算子的输入、输出、属性不同，需要应用在调用时严格按照算子输入、输出、属性来组织算子。用户在调用[aclopExecute](#)接口时，AscendCL根据optype、输入tensor的描述、输出tensor的描述、attr等信息查找对应的任务，并下发执行。

函数原型

```
aclError aclopExecute(const char *opType,  
int numInputs,  
const aclTensorDesc *const inputDesc[],  
const aclDataBuffer *const inputs[],  
int numOutputs,  
const aclTensorDesc *const outputDesc[],  
aclDataBuffer *const outputs[],  
const aclopAttr *attr,  
aclrtStream stream)
```

参数说明

参数名	输入/输出	说明
opType	输入	算子类型名称的指针。
numInputs	输入	算子输入tensor的数量。
inputDesc	输入	算子输入tensor描述的指针数组。 需提前调用 aclCreateTensorDesc 接口创建aclTensorDesc类型。 inputDesc数组中的元素个数必须与numInputs参数值保持一致，且inputs数组与inputDesc数组中的元素必须一一对应。

参数名	输入/输出	说明
inputs	输入	算子输入tensor的指针数组。 需提前调用 aclCreateDataBuffer 接口创建aclDataBuffer类型的数据。 inputs数组中的元素个数必须与numInputs参数值保持一致，且inputs数组与inputDesc数组中的元素必须一一对应。
numOutputs	输入	算子输出tensor的数量。 outputDesc数组中的元素个数必须与numOutputs参数值保持一致。
outputDesc	输入	算子输出tensor描述的指针数组。 需提前调用 aclCreateTensorDesc 接口创建aclTensorDesc类型。 outputDesc数组中的元素个数必须与numOutputs参数值保持一致，且outputs数组与outputDesc数组中的元素必须一一对应。
outputs	输出	算子输出tensor的指针数组。 需提前调用 aclCreateDataBuffer 接口创建aclDataBuffer类型的数据。 outputs数组中的元素个数必须与numOutputs参数值保持一致，且outputs数组与outputDesc数组中的元素必须一一对应。
attr	输入	算子属性的指针。 需提前调用 aclopCreateAttr 接口创建aclopAttr类型。
stream	输入	该算子需要加载的stream。

返回值说明

返回0表示成功，返回其他值表示失败。

1.6.2.6 aclopExecuteV2

函数功能

执行指定的算子。

约束说明

- 该接口是异步接口，调用接口成功仅表示任务下发成功，不表示任务执行成功。调用该接口后，需调用同步等待接口（例如，[aclrtSynchronizeStream](#)）确保任务已执行完成，否则可能会导致训练或推理等业务异常、Device断链掉卡等未知情况。

- 多线程场景下，不支持调用本接口时指定同一个Stream或使用默认Stream，否则可能任务执行异常。
- 每个算子的输入、输出、属性不同，需要应用在调用时严格按照算子输入、输出、属性来组织算子。用户在调用aclopExecuteV2接口时，AscendCL根据optype、输入tensor的描述、输出tensor的描述、attr等信息查找对应的任务，并下发执行。
- 对于支持动态Shape的算子，无法明确算子输出Shape时，可调用[aclopInferShape](#)接口获取算子的输出Shape：
 - 若可获取到准确的输出Shape，则使用准确的输出Shape来构造outputDesc参数，作为aclopExecuteV2的输入。该场景下，aclopExecuteV2接口是**异步接口**，对于异步接口，调用接口成功仅表示任务下发成功，不表示任务执行成功。调用该接口后，需调用同步等待接口（例如，[aclrtSynchronizeStream](#)）确保任务已执行完成。
 - 若无法获取准确的输出Shape，仅能获取输出Shape范围，则使用Shape最大值来构造outputDesc参数，作为aclopExecuteV2的输入。该场景下，在调用aclopExecuteV2接口执行算子后，系统内部会计算出准确的输出Shape，通过aclopExecuteV2接口的outputDesc参数输出，此时aclopExecuteV2接口是**同步接口**。
 - （该场景预留）若无法获取准确的输出Shape以及Shape范围，则需由用户预估一个最大的Shape来构造outputDesc参数，作为aclopExecuteV2的输入。该场景下，在调用aclopExecuteV2接口执行算子后，系统内部会计算出准确的输出Shape，通过aclopExecuteV2接口的outputDesc参数输出，此时aclopExecuteV2接口是**同步接口**。
- 执行有可选输入的算子时，如果可选输入不使用：
 - 则需按此种方式创建[aclTensorDesc](#)类型的数据：[aclCreateTensorDesc](#)(ACL_DT_UNDEFINED, 0, nullptr, ACL_FORMAT_UNDEFINED)，表示数据类型设置为ACL_DT_UNDEFINED，Format设置为ACL_FORMAT_UNDEFINED，Shape信息为nullptr。
 - 则需按此种方式创建[aclDataBuffer](#)类型的数据：[aclCreateDataBuffer](#)(nullptr, 0)，同时aclDataBuffer中的数据不需要释放，因为是空指针。
- 执行有constant输入的算子时，需要先调用[aclSetTensorConst](#)接口设置constant输入。

调用[aclopCompile](#)接口、aclopExecuteV2接口时，设置的constant输入数据必须保持一致。

对于算子有constant输入的场景，如果未调用[aclSetTensorConst](#)接口设置constant输入，则需调用[aclSetTensorPlaceMent](#)设置TensorDesc的placement属性，将memType设置为Host内存。
- 在执行单算子时，一般要求用户传入的输入/输出tensor数据是存放在Device内存的，比如两个tensor相加的场景。但是，也存在部分算子，除了将featureMap、weight等Device内存中的tensor数据作为输入外，还需tensor shape信息、learningRate、tensor的dims信息等通常在Host内存中的tensor数据也作为输入，此时，用户不需要额外将这些Host内存中的tensor数据拷贝到Device上作为输入，只需要调用[aclSetTensorPlaceMent](#)接口设置对应TensorDesc的placement属性为Host内存，在执行单算子时，设置了placement属性为Host内存的TensorDesc，其对应的tensor数据必须存放在Host内存中，AscendCL内部会自动将Host上的tensor数据拷贝到Device上。

函数原型

```
aclError aclopExecuteV2(const char *opType,  
int numInputs,  
aclTensorDesc *inputDesc[],  
aclDataBuffer *inputs[],  
int numOutputs,  
aclTensorDesc *outputDesc[],  
aclDataBuffer *outputs[],  
aclOpAttr *attr,  
aclrtStream stream)
```

参数说明

参数名	输入/输出	说明
opType	输入	算子类型名称的指针。
numInputs	输入	算子输入tensor的数量。
inputDesc	输入	算子输入tensor描述的指针数组。 需提前调用 aclCreateTensorDesc 接口创建aclTensorDesc类型。 inputDesc数组中的元素个数必须与numInputs参数值保持一致，且inputs数组与inputDesc数组中的元素必须一一对应。
inputs	输入	算子输入tensor的指针数组。 需提前调用 aclCreateDataBuffer 接口创建aclDataBuffer类型的数据。 inputs数组中的元素个数必须与numInputs参数值保持一致，且inputs数组与inputDesc数组中的元素必须一一对应。
numOutputs	输入	算子输出tensor的数量。
outputDesc	输入&输出	算子输出tensor描述的指针数组。 需提前调用 aclCreateTensorDesc 接口创建aclTensorDesc类型。 outputDesc数组中的元素个数必须与numOutputs参数值保持一致，且outputs数组与outputDesc数组中的元素必须一一对应。

参数名	输入/输出	说明
outputs	输出	算子输出tensor的指针数组。 需提前调用 aclCreateDataBuffer 接口创建aclDataBuffer类型的数据。 outputs数组中的元素个数必须与numOutputs参数值保持一致，且outputs数组与outputDesc数组中的元素必须一一对应。
attr	输入	算子属性的指针。 需提前调用 aclopCreateAttr 接口创建aclopAttr类型。
stream	输入	该算子需要加载的stream。

返回值说明

返回0表示成功，返回其他值表示失败。

参考资源

- 接口调用流程参见单算子调用流程。
- 接口调用示例请参见执行固定Shape算子示例代码。

1.6.2.7 aclopCompileAndExecute

函数功能

编译并执行指定的算子，当前只支持固定Shape的算子，异步接口。在编译执行算子前，可以调用[aclSetCompileopt](#)接口设置编译选项。

约束说明

- 该接口是异步接口，调用接口成功仅表示任务下发成功，不表示任务执行成功。调用该接口后，需调用同步等待接口（例如，[aclrtSynchronizeStream](#)）确保任务已执行完成，否则可能会导致训练或推理等业务异常、Device断链掉卡等未知情况。
- 多线程场景下，不支持调用本接口时指定同一个Stream或使用默认Stream，否则可能任务执行异常。
- 每个算子的输入、输出、属性不同，需要应用在调用时严格按照算子输入、输出、属性来组织算子，用户在调用aclopCompileAndExecute时，AscendCL根据optype、输入tensor的描述、输出tensor的描述、attr等信息查找对应的任务，再编译、执行算子。
- 编译、执行有可选输入的算子时，如果可选输入不使用，则：
 - 需按此种方式创建aclTensorDesc类型的数据：
[aclCreateTensorDesc](#)(ACL_DT_UNDEFINED, 0, nullptr, ACL_FORMAT_UNDEFINED)，表示数据类型设置为ACL_DT_UNDEFINED，Format设置为ACL_FORMAT_UNDEFINED，Shape信息为nullptr；

- 需按此种方式创建[aclDataBuffer](#)类型的数据：[aclCreateDataBuffer](#)(nullptr, 0)，同时[aclDataBuffer](#)中的数据不需要释放，因为是空指针。
- 编译、执行有constant输入的算子时，需要先调用[aclSetTensorConst](#)接口设置constant输入。
对于算子有constant输入的场景，如果未调用[aclSetTensorConst](#)接口设置constant输入，则需调用[aclSetTensorPlaceMent](#)设置TensorDesc的placement属性，将memType设置为Host内存。
- 在执行单算子时，一般要求用户传入的输入/输出tensor数据是存放在Device内存的，比如两个tensor相加的场景。但是，也存在部分算子，除了将featureMap、weight等Device内存中的tensor数据作为输入外，还需tensor shape信息、learningRate、tensor的dims信息等通常在Host内存中的tensor数据也作为输入，此时，用户不需要额外将这些Host内存中的tensor数据拷贝到Device上作为输入，只需要调用[aclSetTensorPlaceMent](#)接口设置对应TensorDesc的placement属性为Host内存，在执行单算子时，设置了placement属性为Host内存的TensorDesc，其对应的tensor数据必须存放在Host内存中，AscendCL内部会自动将Host上的tensor数据拷贝到Device上。

函数原型

```
aclError aclopCompileAndExecute(const char *opType,  
  
int numInputs,  
  
const aclTensorDesc *const inputDesc[],  
  
const aclDataBuffer *const inputs[],  
  
int numOutputs,  
  
const aclTensorDesc *const outputDesc[],  
aclDataBuffer *const outputs[],  
  
const aclopAttr *attr,  
aclopEngineType engineType,  
aclopCompileType compileFlag,  
  
const char *opPath,  
aclrtStream stream)
```

参数说明

参数名	输入/输出	说明
opType	输入	算子类型名称的指针。
numInputs	输入	算子输入tensor的数量。

参数名	输入/输出	说明
inputDesc	输入	算子输入tensor描述的指针数组。 需提前调用 aclCreateTensorDesc 接口创建aclTensorDesc类型。 inputDesc数组中的元素个数必须与numInputs参数值保持一致，且inputs数组与inputDesc数组中的元素必须一一对应。
inputs	输入	算子输入tensor的指针数组。 需提前调用 aclCreateDataBuffer 接口创建aclDataBuffer类型的数据。 inputs数组中的元素个数必须与numInputs参数值保持一致，且inputs数组与inputDesc数组中的元素必须一一对应。
numOutputs	输入	算子输出tensor的数量。
outputDesc	输入	算子输出tensor描述的指针数组。 需提前调用 aclCreateTensorDesc 接口创建aclTensorDesc类型。 outputDesc数组中的元素个数必须与numOutputs参数值保持一致，且outputs数组与outputDesc数组中的元素必须一一对应。
outputs	输入&输出	算子输出tensor的指针数组。 需提前调用 aclCreateDataBuffer 接口创建aclDataBuffer类型的数据。 outputs数组中的元素个数必须与numOutputs参数值保持一致，且outputs数组与outputDesc数组中的元素必须一一对应。
attr	输入	算子属性的指针。 需提前调用 aclopCreateAttr 接口创建aclopAttr类型。
engineType	输入	算子执行引擎。
compileFlag	输入	算子编译标识。
opPath	输入	算子实现文件（*.py）路径的指针，不包含文件名。预留参数，当前仅支持设置为nullptr。
stream	输入	该算子需要加载的stream。

返回值说明

返回0表示成功，返回其他值表示失败。

1.6.2.8 aclopCompileAndExecuteV2

函数功能

编译并执行指定的算子，当前只支持固定Shape的算子。在编译执行算子前，可以调用[aclSetCompileopt](#)接口设置编译选项。

约束说明

- 多线程场景下，不支持调用本接口时指定同一个Stream或使用默认Stream，否则可能任务执行异常。
- 每个算子的输入、输出、属性不同，需要应用在调用时严格按照算子输入、输出、属性来组织算子，用户在调用[aclopCompileAndExecuteV2](#)时，AscendCL根据optype、输入tensor的描述、输出tensor的描述、attr等信息查找对应的任务，再编译、执行算子。
- 编译、执行有可选输入的算子时，如果可选输入不使用，则：
 - 需按此种方式创建[aclTensorDesc](#)类型的数据：[aclCreateTensorDesc](#)(ACL_DT_UNDEFINED, 0, nullptr, ACL_FORMAT_UNDEFINED)，表示数据类型设置为ACL_DT_UNDEFINED，Format设置为ACL_FORMAT_UNDEFINED，Shape信息为nullptr；
 - 需按此种方式创建[aclDataBuffer](#)类型的数据：[aclCreateDataBuffer](#)(nullptr, 0)，同时[aclDataBuffer](#)中的数据不需要释放，因为是空指针。
- 编译、执行有constant输入的算子时，需要先调用[aclSetTensorConst](#)接口设置constant输入。

对于算子有constant输入的场景，如果未调用[aclSetTensorConst](#)接口设置constant输入，则需调用[aclSetTensorPlaceMent](#)设置TensorDesc的placement属性，将memType设置为Host内存。
- 在执行单算子时，一般要求用户传入的输入/输出tensor数据是存放在Device内存中的，比如两个tensor相加的场景。但是，也存在部分算子，除了将featureMap、weight等Device内存中的tensor数据作为输入外，还需tensor shape信息、learningRate、tensor的dims信息等通常在Host内存中的tensor数据也作为输入，此时，用户不需要额外将这些Host内存中的tensor数据拷贝到Device上作为输入，只需要调用[aclSetTensorPlaceMent](#)接口设置对应TensorDesc的placement属性为Host内存，在执行单算子时，设置了placement属性为Host内存的TensorDesc，其对应的tensor数据必须存放在Host内存中，AscendCL内部会自动将Host上的tensor数据拷贝到Device上。
- 对于动态Shape的算子，无法明确算子输出Shape时，可调用[aclopInferShape](#)接口获取算子的输出Shape：
 - 若可获取到准确的输出Shape，则使用准确的输出Shape来构造outputDesc参数，作为[aclopCompileAndExecuteV2](#)的输入。该场景下，[aclopCompileAndExecuteV2](#)接口是**异步接口**，对于异步接口，调用接口成功仅表示任务下发成功，不表示任务执行成功。调用该接口后，需调用同步等待接口（例如，[aclrtSynchronizeStream](#)）确保任务已执行完成。
 - 若无法获取准确的输出Shape，仅能获取输出Shape范围，则使用Shape最大值来构造outputDesc参数，作为[aclopCompileAndExecuteV2](#)的输入。该场景下，在调用[aclopCompileAndExecuteV2](#)接口执行算子后，系统内部会计算出准确的输出Shape，通过[aclopCompileAndExecuteV2](#)接口的outputDesc参数输出，此时[aclopCompileAndExecuteV2](#)接口是**同步接口**。
 - （该场景预留）若无法获取准确的输出Shape以及Shape范围，则需由用户预估一个最大的Shape来构造outputDesc参数，作为[aclopCompileAndExecuteV2](#)的输入。该场景下，在调用[aclopCompileAndExecuteV2](#)接口执行算子后，系统内部会计算出准确的输出

Shape，通过aclopCompileAndExecuteV2接口的outputDesc参数输出，此时aclopCompileAndExecuteV2接口是**同步接口**。

函数原型

```
aclError aclopCompileAndExecuteV2(const char *opType,  
int numInputs,  
aclTensorDesc *inputDesc[],  
aclDataBuffer *inputs[],  
int numOutputs,  
aclTensorDesc *outputDesc[],  
aclDataBuffer *outputs[],  
aclopAttr *attr,  
aclopEngineType engineType,  
aclopCompileType compileFlag,  
const char *opPath,  
aclrtStream stream)
```

参数说明

参数名	输入/输出	说明
opType	输入	算子类型名称的指针。
numInputs	输入	算子输入tensor的数量。
inputDesc	输入	算子输入tensor描述的指针数组。 需提前调用 aclCreateTensorDesc 接口创建aclTensorDesc类型。 inputDesc数组中的元素个数必须与numInputs参数值保持一致，且inputs数组与inputDesc数组中的元素必须一一对应。
inputs	输入	算子输入tensor的指针数组。 需提前调用 aclCreateDataBuffer 接口创建aclDataBuffer类型的数据。 inputs数组中的元素个数必须与numInputs参数值保持一致，且inputs数组与inputDesc数组中的元素必须一一对应。
numOutputs	输入	算子输出tensor的数量。

参数名	输入/输出	说明
outputDesc	输入&输出	算子输出tensor描述的指针数组。 需提前调用 aclCreateTensorDesc 接口创建aclTensorDesc类型。 outputDesc数组中的元素个数必须与numOutputs参数值保持一致，且outputs数组与outputDesc数组中的元素必须一一对应。
outputs	输入&输出	算子输出tensor的指针数组。 需提前调用 aclCreateDataBuffer 接口创建aclDataBuffer类型的数据。 outputs数组中的元素个数必须与numOutputs参数值保持一致，且outputs数组与outputDesc数组中的元素必须一一对应。
attr	输入	算子属性的指针。 需提前调用 aclopCreateAttr 接口创建aclopAttr类型。
engineType	输入	算子执行引擎。
compileFlag	输入	算子编译标识。
opPath	输入	算子实现文件（*.py）路径的指针，不包含文件名。预留参数，当前仅支持设置为nullptr。
stream	输入	该算子需要加载的stream。

返回值说明

返回0表示成功，返回其他值表示失败。

1.6.2.9 aclopExecWithHandle

函数功能

以Handle方式调用一个算子，不支持动态Shape算子，动态Shape算子请使用[aclopExecuteV2](#)。异步接口。

约束说明

- 该接口是异步接口，调用接口成功仅表示任务下发成功，不表示任务执行成功。调用该接口后，需调用同步等待接口（例如，[aclrtSynchronizeStream](#)）确保任务已执行完成，否则可能会导致训练或推理等业务异常、Device断链掉卡等未知情况。
- 多线程场景下，不支持调用本接口时指定同一个Stream或使用默认Stream，否则可能任务执行异常。

- 执行有可选输入的算子时，如果可选输入不使用，则需按此种方式创建 **aclDataBuffer** 类型的数据：`aclCreateDataBuffer(nullptr, 0)`，同时 `aclDataBuffer` 中的数据不需要释放，因为是空指针。

函数原型

```
aclError aclopExecWithHandle(aclHandle *handle,  
int numInputs,  
const aclDataBuffer *const inputs[],  
int numOutputs,  
aclDataBuffer *const outputs[],  
aclrtStream stream);
```

参数说明

参数名	输入/输出	说明
handle	输入	算子执行算子handle的指针。 需提前调用 aclCreateHandle 接口创建 aclHandle 类型的数据。
numInputs	输入	算子输入tensor的数量。
inputs	输入	算子输入tensor的指针数组。 需提前调用 aclCreateDataBuffer 接口创建 aclDataBuffer 类型的数据。 inputs数组中的元素个数必须与numInputs参数值保持一致。
numOutputs	输入	算子输出tensor的数量。
outputs	输出	算子输出tensor的指针数组。 需提前调用 aclCreateDataBuffer 接口创建 aclDataBuffer 类型的数据。 outputs数组中的元素个数必须与numOutputs参数值保持一致
stream	输入	该算子需要加载的stream。

返回值说明

返回0表示成功，返回其他值表示失败。

1.6.2.10 acloplnferShape

函数功能

根据算子的输入Shape、输入值推导出算子的输出Shape。

约束说明

- 推导算子的输出Shape包含三种场景：
 - 根据Shape推导，可以得到算子的准确输出Shape，则返回准确输出Shape；
 - 根据Shape推导，无法得到算子的准确输出Shape，但可以得到输出Shape的范围，则在输出参数outputDesc中将算子输出tensor描述中的动态维度的维度值记为-1。该场景下，用户可调用[aclGetTensorDescDimRange](#)接口获取tensor描述中指定维度的范围值。
 - （该场景预留）根据Shape推导，无法得到算子的准确输出Shape以及Shape范围，则在输出参数outputDesc中将算子输出tensor描述中的动态维度的维度值记为-2。
- 如果算子有动态输入DYNAMIC_INPUT或动态输出DYNAMIC_OUTPUT，在调用acloplnferShape接口推导算子的输出Shape前，需先调用[aclSetTensorDescName](#)接口设置所有输入和输出的tensor描述的名称，且名称必须按照如下要求（算子IR原型中定义的输入/输出名称请参见《[算子加速库接口参考](#)》。）
 - 对于必选输入、可选输入、必选输出，名称必须与算子IR原型中定义的输入/输出名称保持一致。
 - 对于动态输入、动态输出，名称必须是：算子IR原型中定义的输入/输出名称+编号。编号根据动态输入/输出的个数确定，从0开始，0对应第一个动态输入/输出，1对应第二个动态输入/输出，以此类推。

例如某个算子有2个输入（第1个是必选输入x，第二个是动态输入y且输入个数为2）、1个必选输出z，则调用[aclSetTensorDescName](#)接口设置名称的代码示例如下：

```
aclSetTensorDescName(inputTensorDesc[0], "x");  
aclSetTensorDescName(inputTensorDesc[1], "y0");  
aclSetTensorDescName(inputTensorDesc[2], "y1");  
aclSetTensorDescName(outputTensorDesc[0], "z");
```

函数原型

```
aclError acloplnferShape(const char *opType,  
  
int numInputs,  
  
aclTensorDesc *inputDesc[],  
  
aclDataBuffer *inputs[],  
  
int numOutputs,  
  
aclTensorDesc *outputDesc[],  
  
aclopAttr *attr)
```

参数说明

参数名	输入/输出	说明
opType	输入	算子类型名称的指针。
numInputs	输入	算子输入tensor的数量。
inputDesc	输入	算子输入tensor描述的指针数组。 需提前调用 aclCreateTensorDesc 接口创建aclTensorDesc类型。 inputDesc数组中的元素个数必须与numInputs参数值保持一致。
inputs	输入	算子输入tensor的指针数组。 需提前调用 aclCreateDataBuffer 接口创建aclDataBuffer类型的数据。 此处算子输入tensor数据的内存必须根据应用运行模式来确定，应用运行在Host时，此处需申请Host上的内存；应用运行在Device时，此处需申请Device上的内存。内存申请接口请参见 1.5.5 内存管理 。
numOutputs	输入	算子输出tensor的数量。
outputDesc	输出	算子输出tensor描述的指针数组。 需提前调用 aclCreateTensorDesc 接口创建aclTensorDesc类型。 outputDesc数组中的元素个数必须与numOutputs参数值保持一致
attr	输入	算子属性。

返回值说明

返回0表示成功，返回其他值表示失败。

参考资源

接口调用示例，参见执行动态Shape算子示例代码（不注册算子选择器）。

1.6.2.11 aclopStartDumpArgs

- Atlas 200/300/500 推理产品，不支持该接口。
- Atlas 推理系列产品，不支持该接口。
- Atlas 200/500 A2推理产品，不支持该接口。

函数功能

调用本接口开启算子信息统计功能，并需与[aclopStopDumpArgs](#)接口配合使用，将算子信息文件输出到path参数指定的目录，一个shape对应一个算子信息文件，文件中包含算子类型、算子属性、算子输入&输出的format/数据类型/shape等信息。

使用场景：例如要统计某个模型执行涉及哪些算子，可在模型执行之前调用aclopStartDumpArgs接口，在模型执行之后调用aclopStopDumpArgs接口，接口调用成功后，在path参数指定的目录下生成每个算子shape的算子信息文件。

约束说明

仅支持在单算子API执行场景下使用本接口，否则无法生成dump文件。

函数原型

[aclError](#) aclopStartDumpArgs(uint32_t dumpType, const char *path)

参数说明

参数名	输入/输出	说明
dumpType	输入	指定dump信息的类型。 当前仅支持ACL_OP_DUMP_OP_AICORE_ARGS，表示统计所有算子信息。 <code>#define ACL_OP_DUMP_OP_AICORE_ARGS 0x00000001U</code>
path	输入	指定dump文件的保存路径，支持绝对路径或相对路径（指相对应用可执行文件所在的目录），但用户需确保路径存在或者该路径可以被创建。

返回值说明

返回0表示成功，返回其他值表示失败。

1.6.2.12 aclopStopDumpArgs

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

Atlas 200/500 A2推理产品，不支持该接口。

函数功能

调用本接口关闭算子信息统计功能，并需与[aclopStartDumpArgs](#)接口配合使用，将算子信息文件输出到path参数指定的目录，一个shape对应一个算子信息文件，文件中包含算子类型、算子属性、算子输入&输出的format/数据类型/shape等信息。

使用场景：例如要统计某个模型执行涉及哪些算子，可在模型执行之前调用aclopStartDumpArgs接口，在模型执行之后调用aclopStopDumpArgs接口，接口调用成功后，在path参数指定的目录下生成每个算子shape的算子信息文件。

约束说明

仅支持在单算子API执行场景下使用本接口，否则无法生成dump文件。

函数原型

aclError **aclopStopDumpArgs**(uint32_t dumpType)

参数说明

参数名	输入/输出	说明
dumpType	输入	指定dump信息的类型。 当前仅支持ACL_OP_DUMP_OP_AICORE_ARGS，表示统计所有算子信息。 #define ACL_OP_DUMP_OP_AICORE_ARGS 0x00000001U

返回值说明

返回0表示成功，返回其他值表示失败。

1.6.2.13 aclopRegisterCompileFunc

函数功能

动态Shape场景下，注册算子选择器，用于在算子执行时，能针对不同shape，选择相应的Tiling策略。

约束说明

如果某算子已注册算子选择器，则不允许重新注册，如果需要变更算子选择器，必须先调用**aclopUnregisterCompileFunc**接口取消注册，然后再调用**aclopRegisterCompileFunc**接口重新注册。

函数原型

aclError **aclopRegisterCompileFunc**(const char *opType, aclopCompileFunc func)

参数说明

参数名	输入/输出	说明
opType	输入	算子类型的指针。

参数名	输入/输出	说明
func	输入	算子选择器回调函数，函数定义： typedef aclError (*aclopCompileFunc)(int numInputs, const aclTensorDesc *const inputDesc[], int numOutputs, const aclTensorDesc *const outputDesc[], const aclopAttr *opAttr, aclopKernelDesc *aclopKernelDesc);

返回值说明

返回0表示成功，返回其他值表示失败。

参考资源

接口调用流程及示例，参见执行动态Shape算子示例代码（注册算子选择器）。

1.6.2.14 aclopUnregisterCompileFunc

函数功能

动态Shape场景下，取消注册算子选择器。

函数原型

aclError aclopUnregisterCompileFunc(const char *opType)

参数说明

参数名	输入/输出	说明
opType	输入	算子类型的指针。

返回值说明

返回0表示成功，返回其他值表示失败。

1.6.2.15 aclopCreateKernel

函数功能

动态Shape场景下，将算子注册到系统内部，运行算子时使用。

函数原型

aclError aclopCreateKernel(const char *opType,
const char *kernelId,

```
const char *kernelName,  
  
void *binData,  
  
int binSize,  
  
aclopEngineType enginetype,  
  
aclDataDeallocator deallocator)
```

参数说明

参数名	输入/输出	说明
opType	输入	算子类型的指针。
kernelId	输入	算子执行时要指定的Kernel ID的指针。
kernelName	输入	算子Kernel名称的指针，和算子二进制文件中的kernelName保持一致。
binData	输入	算子Kernel文件的内存地址指针。
binSize	输入	算子Kernel文件的内存大小，单位为Byte。
enginetype	输入	表示算子执行引擎。 typedef enum aclEngineType { ACL_ENGINE_SYS, //不关心具体执行引擎时填写 ACL_ENGINE_AICORE, //将算子编译成AI Core算子 ACL_ENGINE_VECTOR, //将算子编译成Vector Core算子 } aclopEngineType;
deallocator	输入	释放binData内存的回调函数，调用者根据binData的构造方法设置对应的释放函数。如果数据由调用者释放则设置为空。 函数签名为： typedef void (*aclDataDeallocator)(void *data, size_t length);

返回值说明

返回0表示成功，返回其他值表示失败。

参考资源

接口调用流程及示例，参见执行动态Shape算子示例代码（注册算子选择器）。

1.6.2.16 aclopSetKernelArgs

函数功能

动态Shape场景下，设置算子Tiling参数、执行并发数。

函数原型

```
aclError aclopSetKernelArgs(aclopKernelDesc *kernelDesc,  
const char *kernelId,  
uint32_t blockDim,  
const void *args,  
uint32_t argSize)
```

参数说明

参数名	输入/输出	说明
kernelDesc	输入	Kernel描述缓存，aclopKernelDesc类型的指针。 typedef struct aclopKernelDesc aclopKernelDesc;
kernelId	输入	算子执行时要指定的Kernel ID的指针，与调用 aclopCreateKernel 时传递的kernelId一致。
blockDim	输入	Kernel执行的并发AI Core核数。 建议此处设置的blockDim和TIK算子实现时的使用的AI Core核数保持一致。
args	输入	Tiling参数的指针。
argSize	输入	Tiling参数内存大小，单位为Byte。

返回值说明

返回0表示成功，返回其他值表示失败。

参考资源

接口调用流程及示例，参见执行动态Shape算子示例代码（注册算子选择器）。

1.6.2.17 aclopSetKernelWorkspaceSizes

函数功能

动态Shape场景下，设置算子Workspace参数。为非必须接口，根据算子情况可选。

函数原型

```
aclError aclopSetKernelWorkspaceSizes(aclopKernelDesc *kernelDesc, int  
numWorkspaces, size_t *workspaceSizes)
```

参数说明

参数名	输入/输出	说明
kernelDesc	输入	Kernel描述缓存，aclopKernelDesc类型的指针。 typedef struct aclopKernelDesc aclopKernelDesc
numWorkspaces	输入	Workspaces个数。
workspaceSizes	输入	Workspaces大小的数组地址指针。

返回值说明

返回0表示成功，返回其他值表示失败。

1.6.2.18 aclopUpdateParams

函数功能

在算子动态Shape场景下，根据指定算子，触发调用算子选择器。

函数原型

```
aclError aclopUpdateParams(const char *opType,  
int numInputs,  
const aclTensorDesc *const inputDesc[],  
int numOutputs,  
const aclTensorDesc *const outputDesc[],  
const aclopAttr *attr)
```

参数说明

参数名	输入/输出	说明
opType	输入	算子类型名称的指针。
numInputs	输入	算子输入tensor的数量。
inputDesc	输入	算子输入tensor的描述的指针数组。 需提前调用 aclCreateTensorDesc 接口创建aclTensorDesc类型。 inputDesc数组中的元素个数必须与numInputs参数值保持一致。

参数名	输入/输出	说明
numOutputs	输入	算子输出tensor的数量。
outputDesc	输入	算子输出tensor的描述的指针数组。 需提前调用 aclCreateTensorDesc 接口创建aclTensorDesc类型。 outputDesc数组中的元素个数必须与numOutputs参数值保持一致。
attr	输入	算子属性的指针。 需提前调用 aclopCreateAttr 接口创建aclopAttr类型。

返回值说明

返回0表示成功，返回其他值表示失败。

参考资源

接口调用流程及示例，参见执行动态Shape算子示例代码（注册算子选择器）。

1.6.2.19 aclopSetMaxOpQueueNum

函数功能

通过单算子模型方式执行单个算子时，配置算子缓存信息老化信息，以达到节约内存和平衡调用性能的目的。

关于单算子模型的介绍请参见单算子调用流程。

函数原型

aclError aclopSetMaxOpQueueNum(uint64_t maxOpNum)

参数说明

参数名	输入/输出	说明
maxOpNum	输入	“算子类型-单算子模型”映射队列的最大长度。 如果长度达到最大，则会先删除长期未使用的映射信息以及缓存中的单算子模型，再加载最新的映射信息以及对应的单算子模型。 通过单算子模型方式执行单个算子时（aclopUpdateParams接口执行单算子除外），如果不调用本接口配置映射队列的最大长度，则默认最大长度为20000。 aclopUpdateParams接口执行单算子时，如果不调用本接口配置映射队列的最大长度，则表示无需老化算子缓存信息。

返回值说明

返回0表示成功，返回其他值表示失败。

1.6.3 CBLAS 接口

1.6.3.1 总体说明

- 调用本章节接口实现“矩阵-矩阵”或“矩阵-向量”相乘的运算，本章中的接口内部封装了系统内置的矩阵乘算子GEMM，接口调用流程参见单算子调用流程。
按照单算子调用流程，在使用本章节的接口前，需要先使用ATC（Ascend Tensor Compiler）工具将内置的矩阵乘算子GEMM的算子描述信息（包括输入输出Tensor描述、算子属性等）编译成适配昇腾AI处理器的离线模型（*.om文件），用于验证矩阵乘算子GEMM的运行结果。
- 本章的CBLAS接口都是异步接口，调用接口成功仅表示任务下发成功，不表示任务执行成功。调用该接口后，需调用同步等待接口（例如，[aclrtSynchronizeStream](#)）确保任务已执行完成。

1.6.3.2 关于 A、B、C 矩阵的 Shape 及内存大小计算公式

- A、B、C矩阵是否转置的标记如果设置为ACL_TRANS_N或ACL_TRANS_T，则用户在申请内存存放A、B、C矩阵数据时，实际申请的内存要和实际数据大小匹配，Shape及内存大小的计算公式如下：
 - A矩阵：shape = (m, k)；内存大小 = $m * k * \text{sizeof}(\text{dataTypeA})$
 - B矩阵：shape = (k, n)；内存大小 = $k * n * \text{sizeof}(\text{dataTypeB})$
 - C矩阵：shape = (m, n)；内存大小 = $m * n * \text{sizeof}(\text{dataTypeC})$
- A、B、C矩阵是否转置的标记如果设置为ACL_TRANS_NZ，表示采用内部数据格式，矩阵Shape为4维，Shape及内存大小的计算公式如下（假设m,k,n分别为原始轴）：
 - 当矩阵A和矩阵B中数据的类型为[aclFloat16](#)，在计算实际内存大小时，m、k、n均按16对齐向上取整计算：

- A矩阵: shape = ($\lceil k/16 \rceil$, $\lceil m/16 \rceil$, 16, 16); 内存大小 = $\lceil m/16 \rceil * 16 * \lceil k/16 \rceil * 16 * \text{sizeof}(\text{dataTypeA})$
- B矩阵: shape = ($\lceil n/16 \rceil$, $\lceil k/16 \rceil$, 16, 16); 内存大小 = $\lceil k/16 \rceil * 16 * \lceil n/16 \rceil * 16 * \text{sizeof}(\text{dataTypeB})$
- C矩阵: shape = ($\lceil n/16 \rceil$, $\lceil m/16 \rceil$, 16, 16); 内存大小 = $\lceil m/16 \rceil * 16 * \lceil n/16 \rceil * 16 * \text{sizeof}(\text{dataTypeC})$
- 当矩阵A和矩阵B中数据的类型为int8_t, 在计算实际内存大小时, reduce轴按32对齐向上取整计算, 非reduce轴按16对齐向上取整计算:
 - A矩阵: shape = ($\lceil k/32 \rceil$, $\lceil m/16 \rceil$, 16, 32); 内存大小 = $\lceil m/16 \rceil * 16 * \lceil k/32 \rceil * 32 * \text{sizeof}(\text{dataTypeA})$
 - B矩阵: shape = ($\lceil k/32 \rceil$, $\lceil n/16 \rceil$, 32, 16); 内存大小 = $\lceil k/32 \rceil * 32 * \lceil n/16 \rceil * 16 * \text{sizeof}(\text{dataTypeB})$
 - C矩阵: shape = ($\lceil n/16 \rceil$, $\lceil m/16 \rceil$, 16, 16); 内存大小 = $\lceil m/16 \rceil * 16 * \lceil n/16 \rceil * 16 * \text{sizeof}(\text{dataTypeC})$

 说明

$\lceil \rceil$ 表示向上对齐。

1.6.3.3 aclblasGemvEx

函数功能

执行矩阵-向量的乘法, $y = \alpha Ax + \beta y$, 输入数据、输出数据的数据类型通过入参设置, 异步接口。

约束说明

A、x、y的数据类型支持仅支持以下组合, α 和 β 的数据类型与y一致。

A的数据类型	x的数据类型	y的数据类型
aclFloat16	aclFloat16	aclFloat16
aclFloat16	aclFloat16	float(float32)
int8_t	int8_t	float(float32)
int8_t	int8_t	int32_t

函数原型

```
aclError aclblasGemvEx(aclTransType transA,  
int m,  
int n,  
const void *alpha,
```

```
const void *a,  
int lda,  
aclDataType dataTypeA,  
const void *x,  
int incx,  
aclDataType dataTypeX,  
const void *beta,  
void *y,  
int incy,  
aclDataType dataTypeY,  
aclComputeType type,  
aclrtStream stream);
```

参数说明

参数名	输入/输出	说明
transA	输入	A矩阵是否转置的标记。
m	输入	矩阵A的行数，存储矩阵乘数据时，行优先。
n	输入	矩阵A的列数。
alpha	输入	用于执行乘操作的标量α的指针。
a	输入	矩阵A的指针。
lda	输入	A矩阵的主维，此时选择转置，按行优先，则lda为A的列数。预留参数，当前只能设置为-1。
dataTypeA	输入	矩阵A的数据类型。
x	输入	向量x的指针。
incx	输入	x连续元素之间的步长。 预留参数，当前只能设置为-1。
dataTypeX	输入	向量x的数据类型。
beta	输入	用于执行乘操作的标量β的指针。
y	输入&输出	向量y的指针。
incy	输入	y连续元素之间的步长。 预留参数，当前只能设置为-1。
dataTypeY	输入	向量y的数据类型。

参数名	输入/输出	说明
type	输入	计算精度，默认高精度。
stream	输入	执行算子所在的Stream。

返回值说明

返回0表示成功，返回其他值表示失败。

参考资源

接口调用流程，参见单算子调用流程。

1.6.3.4 aclblasCreateHandleForGemvEx

函数功能

创建矩阵-向量乘的handle，输入数据、输出数据的数据类型通过入参设置，同步接口。

创建handle成功后，需调用[aclOpExecWithHandle](#)接口执行算子。

约束说明

A、x、y的数据类型支持仅支持以下组合：

A的数据类型	x的数据类型	y的数据类型
aclFloat16	aclFloat16	aclFloat16
aclFloat16	aclFloat16	float(float32)
int8_t	int8_t	float(float32)
int8_t	int8_t	int32_t

函数原型

```
aclError aclblasCreateHandleForGemvEx(aclTransType transA,  
int m,  
int n,  
aclDataType dataTypeA,  
aclDataType dataTypeX,  
aclDataType dataTypeY,
```

aclComputeType type,
aclopHandle **handle)

参数说明

参数名	输入/输出	说明
transA	输入	A矩阵是否转置的标记。
m	输入	矩阵A的行数，存储矩阵乘数据时，行优先。
n	输入	矩阵A的列数。
dataTypeA	输入	矩阵A的数据类型。
dataTypeX	输入	向量x的数据类型。
dataTypeY	输入	向量y的数据类型。
type	输入	计算精度，默认高精度。
handle	输出	“算子执行handle的指针”的指针。

返回值说明

返回0表示成功，返回其他值表示失败。

1.6.3.5 aclblasHgemv

函数功能

执行矩阵-向量的乘法， $y = \alpha Ax + \beta y$ ，输入数据和输出数据的数据类型为 **aclFloat16**，异步接口：

函数原型

aclError aclblasHgemv(**aclTransType** transA,
int m,
int n,
const **aclFloat16** *alpha,
const **aclFloat16** *a,
int lda,
const **aclFloat16** *x,
int incx,
const **aclFloat16** *beta,

```
aclFloat16 *y,  
int incy,  
aclComputeType type,  
aclrtStream stream)
```

参数说明

参数名	输入/输出	说明
transA	输入	A矩阵是否转置的标记。
m	输入	矩阵A的行数，存储矩阵乘数据时，行优先。
n	输入	矩阵A的列数。
alpha	输入	用于执行乘操作的标量α的指针。
a	输入	矩阵A的指针。
lda	输入	A矩阵的主维，此时选择转置，按行优先，则lda为A的列数。预留参数，当前只能设置为-1。
x	输入	向量x的指针。
incx	输入	x连续元素之间的步长。 预留参数，当前只能设置为-1。
beta	输入	用于执行乘操作的标量β的指针。
y	输入&输出	向量y的指针。
incy	输入	y连续元素之间的步长。 预留参数，当前只能设置为-1。
type	输入	计算精度，默认高精度。
stream	输入	执行算子所在的Stream。

返回值说明

返回0表示成功，返回其他值表示失败。

参考资源

接口调用流程，参见单算子调用流程。

1.6.3.6 aclblasCreateHandleForHgemv

函数功能

创建矩阵-向量乘的handle，输入数据和输出数据的数据类型为[aclFloat16](#)，同步接口。

创建handle成功后，需调用[aclopExecWithHandle](#)接口执行算子。

函数原型

```
aclError aclblasCreateHandleForHgemv(aclTransType transA,  
int m,  
int n,  
aclComputeType type,  
aclopHandle **handle)
```

参数说明

参数名	输入/输出	说明
transA	输入	A矩阵是否转置的标记。
m	输入	矩阵A的行数，存储矩阵乘数据时，行优先。
n	输入	矩阵A的列数。
type	输入	计算精度，默认高精度。
handle	输出	“算子执行handle的指针”的指针。

返回值说明

返回0表示成功，返回其他值表示失败。

1.6.3.7 aclblasS8gemv

函数功能

执行矩阵-向量的乘法， $y = \alpha Ax + \beta y$ ，输入数据的数据类型为[int8_t](#)，输出数据的数据类型为[int32_t](#)，异步接口：

函数原型

```
aclError aclblasS8gemv(aclTransType transA,  
int m,
```



```
int n,  
const int32_t *alpha,  
const int8_t *a,  
int lda,  
const int8_t *x,  
int incx,  
const int32_t *beta,  
int32_t *y,  
int incy,  
aclComputeType type,  
aclrtStream stream)
```

参数说明

参数名	输入/输出	说明
transA	输入	A矩阵是否转置的标记。
m	输入	矩阵A的行数，存储矩阵乘数据时，行优先。
n	输入	矩阵A的列数。
alpha	输入	用于执行乘操作的标量α的指针。
a	输入	矩阵A的指针。
lda	输入	A矩阵的主维，此时选择转置，按行优先，则lda为A的列数。预留参数，当前只能设置为-1。
x	输入	向量x的指针。
incx	输入	x连续元素之间的步长。 预留参数，当前只能设置为-1。
beta	输入	用于执行乘操作的标量β的指针。
y	输入&输出	向量y的指针。
incy	输入	y连续元素之间的步长。 预留参数，当前只能设置为-1。
type	输入	计算精度，默认高精度。
stream	输入	执行算子所在的Stream。

返回值说明

返回0表示成功，返回其他值表示失败。

参考资源

接口调用流程，参见单算子调用流程。

1.6.3.8 aclblasCreateHandleForS8gemv

函数功能

创建矩阵-向量乘的handle，输入数据的数据类型为int8_t，输出数据的数据类型为int32_t，同步接口。

创建handle成功后，需调用[aclopExecWithHandle](#)接口执行算子。

函数原型

```
aclError aclblasCreateHandleForS8gemv(aclTransType transA,  
int m,  
int n,  
aclComputeType type,  
aclHandle **handle)
```

参数说明

参数名	输入/输出	说明
transA	输入	A矩阵是否转置的标记。
m	输入	矩阵A的行数，存储矩阵乘数据时，行优先。
n	输入	矩阵A的列数。
type	输入	计算精度，默认高精度。
handle	输出	“算子执行handle的指针”的指针。

返回值说明

返回0表示成功，返回其他值表示失败。

1.6.3.9 aclblasGemmEx

函数功能

执行矩阵-矩阵的乘法， $C = \alpha AB + \beta C$ ，输入数据、输出数据的数据类型通过入参设置，异步接口：

约束说明

A、B、C的数据类型支持仅支持以下组合， α 和 β 的数据类型与C一致。

A的数据类型	B的数据类型	C的数据类型
<code>aclFloat16</code>	<code>aclFloat16</code>	<code>aclFloat16</code>
<code>aclFloat16</code>	<code>aclFloat16</code>	<code>float(float32)</code>
<code>int8_t</code>	<code>int8_t</code>	<code>float(float32)</code>
<code>int8_t</code>	<code>int8_t</code>	<code>int32_t</code>

函数原型

```
aclError aclblasGemmEx(aclTransType transA,  
aclTransType transB,  
aclTransType transC,  
int m,  
int n,  
int k,  
const void *alpha,  
const void *matrixA,  
int lda,  
aclDataType dataTypeA,  
const void *matrixB,  
int ldb,  
aclDataType dataTypeB,  
const void *beta,  
void *matrixC,  
int ldc,  
aclDataType dataTypeC,  
aclComputeType type,
```

aclrtStream stream)

参数说明

参数名	输入/输出	说明
transA	输入	A矩阵是否转置的标记。
transB	输入	B矩阵是否转置的标记。
transC	输入	C矩阵的标记，当前仅支持ACL_TRANS_N。
m	输入	矩阵A的行数与矩阵C的行数。
n	输入	矩阵B的列数与矩阵C的列数。
k	输入	矩阵A的列数与矩阵B的行数。
alpha	输入	用于执行乘操作的标量α的指针。
matrixA	输入	矩阵A的指针。
lda	输入	A矩阵的主维，此时选择转置，按行优先，则lda为A的列数。预留参数，当前只能设置为-1。
dataTypeA	输入	矩阵A的数据类型。
matrixB	输入	矩阵B的指针。
ldb	输入	B矩阵的主维，此时选择转置，按行优先，则ldb为B的列数。预留参数，当前只能设置为-1。
dataTypeB	输入	矩阵B的数据类型。
beta	输入	用于执行乘操作的标量β的指针。
matrixC	输入&输出	矩阵C的指针。
ldc	输入	C矩阵的主维，预留参数，当前只能设置为-1。
dataTypeC	输入	矩阵C的数据类型。
type	输入	计算精度，默认高精度。
stream	输入	执行算子所在的Stream。

返回值说明

返回0表示成功，返回其他值表示失败。

参考资源

接口调用流程，参见单算子调用流程。

1.6.3.10 aclblasCreateHandleForGemmEx

函数功能

创建矩阵-矩阵乘的handle，输入数据、输出数据的数据类型通过入参设置，同步接口。

创建handle成功后，需调用[aclopExecWithHandle](#)接口执行算子。

约束说明

A、B、C的数据类型支持仅支持以下组合：

A的数据类型	B的数据类型	C的数据类型
aclFloat16	aclFloat16	aclFloat16
aclFloat16	aclFloat16	float(float32)
int8_t	int8_t	float(float32)
int8_t	int8_t	int32_t

函数原型

```
aclError aclblasCreateHandleForGemmEx(aclTransType transA,  
aclTransType transB,  
aclTransType transC,  
int m,  
int n,  
int k,  
aclDataType dataTypeA,  
aclDataType dataTypeB,  
aclDataType dataTypeC,  
aclComputeType type,  
aclopHandle **handle)
```

参数说明

参数名	输入/输出	说明
transA	输入	A矩阵是否转置的标记。

参数名	输入/输出	说明
transB	输入	B矩阵是否转置的标记。
transC	输入	C矩阵的标记，当前仅支持ACL_TRANS_N。
m	输入	矩阵A的行数与矩阵C的行数。
n	输入	矩阵B的列数与矩阵C的列数。
k	输入	矩阵A的列数与矩阵B的行数。
dataTypeA	输入	矩阵A的数据类型。
dataTypeB	输入	矩阵B的数据类型。
dataTypeC	输入	矩阵C的数据类型。
type	输入	计算精度，默认高精度。
handle	输出	“算子执行handle的指针”的指针。

返回值说明

返回0表示成功，返回其他值表示失败。

1.6.3.11 aclblasHgemm

函数功能

执行矩阵-矩阵的乘法， $C = \alpha AB + \beta C$ ，输入数据和输出数据的数据类型为[aclFloat16](#)，异步接口：

函数原型

```
aclError aclblasHgemm(aclTransType transA,
aclTransType transB,
aclTransType transC,
int m,
int n,
int k,
const aclFloat16 *alpha,
const aclFloat16 *matrixA,
int lda,
const aclFloat16 *matrixB,
int ldb,
```

```
const aclFloat16 *beta,  
aclFloat16 *matrixC,  
int ldc,  
aclComputeType type,  
aclrtStream stream)
```

参数说明

参数名	输入/输出	说明
transA	输入	A矩阵是否转置的标记。
transB	输入	B矩阵是否转置的标记。
transC	输入	C矩阵的标记，当前仅支持ACL_TRANS_N。
m	输入	矩阵A的行数与矩阵C的行数。
n	输入	矩阵B的列数与矩阵C的列数。
k	输入	矩阵A的列数与矩阵B的行数。
alpha	输入	用于执行乘操作的标量α的指针。
matrixA	输入	矩阵A的指针。
lda	输入	A矩阵的主维，此时选择转置，按行优先，则lda为A的列数。预留参数，当前只能设置为-1。
matrixB	输入	矩阵B的指针。
ldb	输入	B矩阵的主维，此时选择转置，按行优先，则ldb为B的列数。预留参数，当前只能设置为-1。
beta	输入	用于执行乘操作的标量β的指针。
matrixC	输入&输出	矩阵C的指针。
ldc	输入	C矩阵的主维，预留参数，当前只能设置为-1。
type	输入	计算精度，默认高精度。
stream	输入	执行算子所在的Stream。

返回值说明

返回0表示成功，返回其他值表示失败。

参考资源

接口调用流程，参见单算子调用流程。

1.6.3.12 aclblasCreateHandleForHgemm

函数功能

创建矩阵-矩阵乘的handle，输入数据和输出数据的数据类型为[aclFloat16](#)，同步接口。

创建handle成功后，需调用[aclopExecWithHandle](#)接口执行算子。

函数原型

```
aclError aclblasCreateHandleForHgemm(aclTransType transA,  
aclTransType transB,  
aclTransType transC,  
int m,  
int n,  
int k,  
aclComputeType type,  
aclopHandle **handle)
```

参数说明

参数名	输入/输出	说明
transA	输入	A矩阵是否转置的标记。
transB	输入	B矩阵是否转置的标记。
transC	输入	C矩阵的标记，当前仅支持ACL_TRANS_N。
m	输入	矩阵A的行数与矩阵C的行数。
n	输入	矩阵B的列数与矩阵C的列数。
k	输入	矩阵A的列数与矩阵B的行数。
type	输入	计算精度，默认高精度。
handle	输出	“算子执行handle的指针”的指针。

返回值说明

返回0表示成功，返回其他值表示失败。

1.6.3.13 aclblasS8gemm

函数功能

执行矩阵-矩阵的乘法， $C = \alpha AB + \beta C$ ，输入数据的数据类型为int8_t，输出数据的数据类型为int32_t，异步接口：

函数原型

```
aclError aclblasS8gemm(aclTransType transA,  
aclTransType transB,  
aclTransType transC,  
int m,  
int n,  
int k,  
const int32_t *alpha,  
const int8_t *matrixA,  
int lda,  
const int8_t *matrixB,  
int ldb,  
const int32_t *beta,  
int32_t *matrixC,  
int ldc,  
aclComputeType type,  
aclrtStream stream)
```

参数说明

参数名	输入/输出	说明
transA	输入	A矩阵是否转置的标记。
transB	输入	B矩阵是否转置的标记。
transC	输入	C矩阵的标记，当前仅支持ACL_TRANS_N。
m	输入	矩阵A的行数与矩阵C的行数。
n	输入	矩阵B的列数与矩阵C的列数。
k	输入	矩阵A的列数与矩阵B的行数。
alpha	输入	用于执行乘操作的标量α的指针。

参数名	输入/输出	说明
matrixA	输入	矩阵A的指针。
lda	输入	A矩阵的主维，此时选择转置，按行优先，则lda为A的列数。预留参数，当前只能设置为-1。
matrixB	输入	矩阵B的指针。
ldb	输入	B矩阵的主维，此时选择转置，按行优先，则ldb为B的列数。预留参数，当前只能设置为-1。
beta	输入	用于执行乘操作的标量β的指针。
matrixC	输入&输出	矩阵C的指针。
ldc	输入	C矩阵的主维，预留参数，当前只能设置为-1。
type	输入	计算精度，默认高精度。
stream	输入	执行算子所在的Stream。

返回值说明

返回0表示成功，返回其他值表示失败。

参考资源

接口调用流程，参见单算子调用流程。

1.6.3.14 aclblasCreateHandleForS8gemm

函数功能

创建矩阵-矩阵乘的handle，输入数据的数据类型为int8_t，输出数据的数据类型为int32_t，同步接口。

创建handle成功后，需调用[aclopExecWithHandle](#)接口执行算子。

函数原型

```
aclError aclblasCreateHandleForS8gemm(aclTransType transA,  
aclTransType transB,  
aclTransType transC,  
int m,  
int n,  
int k,
```

```
aclComputeType type,  
aclHandle **handle)
```

参数说明

参数名	输入/输出	说明
transA	输入	A矩阵是否转置的标记。
transB	输入	B矩阵是否转置的标记。
transC	输入	C矩阵的标记，当前仅支持ACL_TRANS_N。
m	输入	矩阵A的行数与矩阵C的行数。
n	输入	矩阵B的列数与矩阵C的列数。
k	输入	矩阵A的列数与矩阵B的行数。
type	输入	计算精度，默认高精度。
handle	输出	“算子执行handle的指针”的指针。

返回值说明

返回0表示成功，返回其他值表示失败。

1.6.3.15 aclopCast

函数功能

转换输入数据的数据类型，异步接口。

函数原型

```
aclError aclopCast(const aclTensorDesc *srcDesc,  
const aclDataBuffer *srcBuffer,  
const aclTensorDesc *dstDesc,  
aclDataBuffer *dstBuffer,  
uint8_t truncate,  
aclrtStream stream)
```

参数说明

参数名	输入/输出	说明
srcDesc	输入	输入tensor描述的指针。
srcBuffer	输入	输入tensor的指针。
dstDesc	输入	输出tensor描述的指针。
dstBuffer	输出	输出tensor的指针。
truncate	输入	预留。
stream	输入	执行算子所在的Stream。

返回值说明

返回0表示成功，返回其他值表示失败。

参考资源

接口调用流程，参见单算子调用流程。

1.6.3.16 aclopCreateHandleForCast

函数功能

创建数据类型转换的handle，同步接口。

创建handle成功后，需调用[aclopExecWithHandle](#)接口执行算子。

函数原型

```
aclError aclopCreateHandleForCast(aclTensorDesc *srcDesc,  
aclTensorDesc *dstDesc,  
uint8_t truncate,  
aclHandle **handle)
```

参数说明

参数名	输入/输出	说明
srcDesc	输入	输入tensor描述的指针。
dstDesc	输入	输出tensor描述的指针。
truncate	输入	预留。

参数名	输入/输出	说明
handle	输出	“算子执行handle的指针”的指针。

返回值说明

返回0表示成功，返回其他值表示失败。

1.7 模型推理

1.7.1 模型加载和卸载

1.7.1.1 aclmdlLoadFromFile

函数功能

从文件加载离线模型数据（适配昇腾AI处理器的离线模型），由系统内部管理内存。

约束说明

模型加载、模型执行、模型卸载的操作必须在同一个Context下（关于Context的创建请参见[1.5.1.1 aclrtSetDevice](#)或[1.5.2.1 aclrtCreateContext](#)）。在加载前，请先根据模型文件的大小评估内存空间是否足够，内存空间不足，会导致应用程序异常。

函数原型

aclError aclmdlLoadFromFile(const char *modelPath, uint32_t *modelId)

参数说明

参数名	输入/输出	说明
modelPath	输入	模型文件路径的指针，路径中包含文件名。运行程序（APP）的用户需要对该存储路径有访问权限。 说明 - 此处的模型文件是适配昇腾AI处理器的离线模型，即*.om文件。关于如何获取om文件，请参见《ATC工具使用指南》。 - 对om模型文件大小有限制的场景下，本接口支持权重文件外置功能。具体使用方法如下：使用ATC工具生成om文件时，将--external_weight参数设置为1（1表示将原始网络中的Const/Constant节点的权重保存在单独的文件中，且该文件保存在与om文件同级的weight目录下），那么在使用本接口加载om文件时，需将weight目录与om文件放在同级目录下，这时AscendCL会自行到weight目录下查找权重文件，否则可能会导致单独的权重文件加载不成功。

参数名	输入/输出	说明
modelId	输出	模型ID的指针。 系统成功加载模型后，返回模型ID作为后续操作时识别模型的标志。

返回值说明

返回0表示成功，返回其他值表示失败。

相关接口

当前AscendCL还提供了[aclmdlSetConfigOpt](#)接口、[aclmdlLoadWithConfig](#)接口来实现模型加载，通过配置对象中的属性来区分，在加载模型时是从文件加载，还是从内存加载，以及内存是由系统内部管理，还是由用户管理。

参考资源

接口调用流程，参见模型加载。

1.7.1.2 aclmdlLoadFromMem

函数功能

从内存加载离线模型数据，由系统内部管理模型运行的内存。

约束说明

模型加载、模型执行、模型卸载的操作必须在同一个Context下（关于Context的创建请参见[1.5.1.1 aclrtSetDevice](#)或[1.5.2.1 aclrtCreateContext](#)）。在加载前，请先根据模型文件的大小评估内存空间是否足够，内存空间不足，会导致应用程序异常。

函数原型

aclError [aclmdlLoadFromMem](#)(const void* model, size_t modelSize, uint32_t* modelId)

参数说明

参数名	输入/输出	说明
model	输入	存放模型数据的内存地址指针。 应用运行在Host时，此处需申请Host上的内存；应用运行在Device时，此处需申请Device上的内存。内存申请接口请参见 1.5.5 内存管理 。
modelSize	输入	内存中的模型数据长度，单位Byte。

参数名	输入/输出	说明
modelId	输出	模型ID的指针。 系统成功加载模型后，返回模型ID作为后续操作时识别模型的标志。

返回值说明

返回0表示成功，返回其他值表示失败。

相关接口

当前AscendCL还提供了[aclmdlSetConfigOpt](#)接口、[aclmdlLoadWithConfig](#)接口来实现模型加载，通过配置对象中的属性来区分，在加载模型时是从文件加载，还是从内存加载，以及内存是由系统内部管理，还是由用户管理。

参考资源

接口调用流程，参见模型加载。

1.7.1.3 aclmdlLoadFromFileWithMem

函数功能

从文件加载离线模型数据（适配昇腾AI处理器的离线模型），由用户自行管理模型运行的内存。

约束说明

模型加载、模型执行、模型卸载的操作必须在同一个Context下（关于Context的创建请参见[1.5.1.1 aclrtSetDevice](#)或[1.5.2.1 aclrtCreateContext](#)）。

函数原型

[aclError](#) [aclmdlLoadFromFileWithMem](#)(const char *modelPath, uint32_t *modelId, void *workPtr, size_t workSize, void *weightPtr, size_t weightSize)

参数说明

参数名	输入/输出	说明
modelPath	输入	模型文件路径的指针，路径中包含文件名。运行程序（APP）的用户需要对该存储路径有访问权限。 此处的模型文件是适配昇腾AI处理器的离线模型，即*.om文件。 说明 此处的模型文件是适配昇腾AI处理器的离线模型，即*.om文件。关于如何获取om文件，请参见《ATC工具使用指南》。
modelId	输出	模型ID的指针。 系统成功加载模型后，返回模型ID作为后续操作时识别模型的标志。
workPtr	输入	Device上模型所需工作内存（存放模型执行过程中的临时数据）的地址指针，由用户自行管理，模型执行过程中不能释放该内存。 如果在workPtr参数处传入空指针，表示由系统管理内存。 说明 由用户自行管理工作内存时，如果多个模型串行执行，支持共用同一个工作内存，但用户需确保模型的串行执行顺序、且工作内存的大小需按多个模型中最大工作内存的大小来申请，例如通过以下方式保证串行： • 同步模型执行时，加锁，保证执行任务串行。 • 异步模型执行时，使用同一个stream，保证执行任务串行。
workSize	输入	模型所需工作内存的大小，单位Byte。workPtr为空指针时无效。
weightPtr	输入	Device上模型权值内存（存放权值数据）的地址指针，由用户自行管理，模型执行过程中不能释放该内存。 如果在weightPtr参数处传入空指针，表示由系统管理内存。 说明 由用户自行管理权值内存时，在多线程场景下，对于同一个模型，如果在每个线程中都加载了一次，支持共用weightPtr，因为weightPtr内存存在推理过程中是只读的。此处需注意，在共用weightPtr期间，不能释放weightPtr。
weightSize	输入	模型所需权值内存的大小，单位Byte。weightPtr为空指针时无效。

返回值说明

返回0表示成功，返回其他值表示失败。

相关接口

当前AscendCL还提供了[aclmdlSetConfigOpt](#)接口、[aclmdlLoadWithConfig](#)接口来实现模型加载，通过配置对象中的属性来区分，在加载模型时是从文件加载，还是从内存加载，以及内存是由系统内部管理，还是由用户管理。

参考资源

接口调用流程及示例代码，参见模型加载。

1.7.1.4 aclmdlLoadFromMemWithMem

函数功能

从内存加载离线模型数据，由用户自行管理模型运行的内存。

约束说明

模型加载、模型执行、模型卸载的操作必须在同一个Context下（关于Context的创建请参见[1.5.1.1 aclrtSetDevice](#)或[1.5.2.1 aclrtCreateContext](#)）。在加载前，请先根据模型文件的大小评估内存空间是否足够，内存空间不足，会导致应用程序异常。

函数原型

aclError [aclmdlLoadFromMemWithMem](#)(const void *model, size_t modelSize, uint32_t *modelId, void *workPtr, size_t workSize, void *weightPtr, size_t weightSize)

参数说明

参数名	输入/输出	说明
model	输入	存放模型数据的内存地址指针。 应用运行在Host时，此处需申请Host上的内存；应用运行在Device时，此处需申请Device上的内存。内存申请接口请参见 1.5.5 内存管理 。
modelSize	输入	模型数据长度，单位Byte。
modelId	输出	模型ID的指针。 系统成功加载模型后，返回模型ID作为后续操作时识别模型的标志。

参数名	输入/输出	说明
workPtr	输入	Device上模型所需工作内存（存放模型执行过程中的临时数据）的地址指针，由用户自行管理，模型执行过程中不能释放该内存。 如果在workPtr参数处传入空指针，表示由系统管理内存。 说明 由用户自行管理工作内存时，如果多个模型串行执行，支持共用同一个工作内存，但用户需确保模型的串行执行顺序、且工作内存的大小需按多个模型中最大工作内存的大小来申请，例如通过以下方式保证串行： • 同步模型执行时，加锁，保证执行任务串行。 • 异步模型执行时，使用同一个stream，保证执行任务串行。
workSize	输入	模型所需工作内存的大小，单位Byte。workPtr为空指针时无效。
weightPtr	输入	Device上模型权值内存（存放权值数据）的地址指针，由用户自行管理，模型执行过程中不能释放该内存。 如果在weightPtr参数处传入空指针，表示由系统管理内存。 说明 由用户自行管理权值内存时，在多线程场景下，对于同一个模型，如果在每个线程中都加载了一次，支持共用weightPtr，因为weightPtr内存存在推理过程中是只读的。此处需注意，在共用weightPtr期间，不能释放weightPtr。
weightSize	输入	模型所需权值内存的大小，单位Byte。weightPtr为空指针时无效。

返回值说明

返回0表示成功，返回其他值表示失败。

相关接口

当前AscendCL还提供了[aclmdlSetConfigOpt](#)接口、[aclmdlLoadWithConfig](#)接口来实现模型加载，通过配置对象中的属性来区分，在加载模型时是从文件加载，还是从内存加载，以及内存是由系统内部管理，还是由用户管理。

参考资源

接口调用流程，参见模型加载。

1.7.1.5 aclmdlLoadFromFileWithQ

函数功能

从文件加载离线模型数据（适配昇腾AI处理器的离线模型），模型的输入、输出数据都存放在队列中。

约束说明

- 本接口只支持加载固定Shape输入的模型。
- 模型加载、模型执行、模型卸载的操作必须在同一个Context下（关于Context的创建请参见[1.5.1.1 aclrtSetDevice](#)或[1.5.2.1 aclrtCreateContext](#)）。

函数原型

aclError aclmdlLoadFromFileWithQ(const char *modelPath, uint32_t *modelId, const uint32_t *inputQ, size_t inputQNum, const uint32_t *outputQ, size_t outputQNum)

参数说明

参数名	输入/输出	说明
modelPath	输入	模型文件路径的指针，路径中包含文件名。运行程序（APP）的用户需要对该存储路径有访问权限。 此处的模型文件是适配昇腾AI处理器的离线模型，即*.om文件。 说明 • 此处的模型文件是适配昇腾AI处理器的离线模型，即*.om文件。关于如何获取om文件，请参见《ATC工具使用指南》。 • 对om模型文件大小有限制的场景下，本接口支持权重文件外置功能。具体使用方法如下：使用ATC工具生成om文件时，将--external_weight参数设置为1（1表示将原始网络中的Const/Constant节点的权重保存在单独的文件中，且该文件保存在与om文件同级的weight目录下），那么在使用本接口加载om文件时，需将weight目录与om文件放在同级目录下，这时AscendCL会自行到weight目录下查找权重文件，否则可能会导致单独的权重文件加载不成功。
modelId	输出	模型ID的指针。 系统成功加载模型后，返回模型ID作为后续操作时识别模型的标志。
inputQ	输入	队列ID的指针，一个模型的输入对应一个队列ID。
inputQNum	输入	输入队列大小。
outputQ	输入	队列ID的指针，一个模型的输出对应一个队列ID。
outputQNum	输入	输出队列大小。

返回值说明

返回0表示成功，返回其他值表示失败。

相关接口

当前AscendCL还提供了[aclmdlSetConfigOpt](#)接口、[aclmdlLoadWithConfig](#)接口来实现模型加载，通过配置对象中的属性来区分，在加载模型时是从文件加载，还是从内存加载，以及内存是由系统内部管理，还是由用户管理。

1.7.1.6 aclmdlLoadFromMemWithQ

函数功能

从内存加载离线模型数据（适配昇腾AI处理器的离线模型），模型的输入、输出数据都存放在队列中。

约束说明

- 本接口只支持加载固定Shape输入的模型。
- 模型加载、模型执行、模型卸载的操作必须在同一个Context下（关于Context的创建请参见[1.5.1.1 aclrtSetDevice](#)或[1.5.2.1 aclrtCreateContext](#)）。

函数原型

aclError [aclmdlLoadFromMemWithQ](#)(const void *model, size_t modelSize, uint32_t *modelId, const uint32_t *inputQ, size_t inputQNum, const uint32_t *outputQ, size_t outputQNum)

参数说明

参数名	输入/输出	说明
model	输入	存放模型数据的内存地址指针。
modelSize	输入	内存中的模型数据长度，单位Byte。
modelId	输出	模型ID的指针。 系统成功加载模型后，返回模型ID作为后续操作时识别模型的标志。
inputQ	输入	队列ID的指针，一个模型的输入对应一个队列ID。
inputQNum	输入	输入队列大小。
outputQ	输入	队列ID的指针，一个模型的输出对应一个队列ID。
outputQNum	输入	输出队列大小。

返回值说明

返回0表示成功，返回其他值表示失败。

相关接口

当前AscendCL还提供了[aclmdlSetConfigOpt](#)接口、[aclmdlLoadWithConfig](#)接口来实现模型加载，通过配置对象中的属性来区分，在加载模型时是从文件加载，还是从内存加载，以及内存是由系统内部管理，还是由用户管理。

1.7.1.7 aclmdlSetConfigOpt

函数功能

设置模型加载的配置对象中的各属性的取值，包括模型执行的优先级、模型的文件路径或内存地址、内存大小等。

本接口需要与以下其它接口配合，实现模型加载功能：

1. 调用[aclmdlCreateConfigHandle](#)接口创建模型加载的配置对象。
2. 多次调用[aclmdlSetConfigOpt](#)接口设置配置对象中每个属性的值。
3. 调用[aclmdlLoadWithConfig](#)接口指定模型加载时需要的配置信息，并进行模型加载。
4. 模型加载成功后，调用[aclmdlDestroyConfigHandle](#)接口销毁。

函数原型

aclError [aclmdlSetConfigOpt](#)([aclmdlConfigHandle](#) *handle, [aclmdlConfigAttr](#) attr, const void *attrValue, size_t valueSize)

参数说明

参数名	输入/输出	说明
handle	输出	模型加载的配置对象的指针。需提前调用 aclmdlCreateConfigHandle 接口创建该对象。
attr	输入	指定需设置的属性。
attrValue	输入	指向属性值的指针，attr对应的属性取值。 如果属性值本身是指针，则传入该指针的地址。
valueSize	输入	attrValue部分的数据长度。 用户可使用C/C++标准库的函数sizeof(*attrValue)查询数据长度。

返回值说明

返回0表示成功，返回其他值表示失败。

相关接口

使用[aclmdlSetConfigOpt](#)接口、[aclmdlLoadWithConfig](#)接口时，是通过配置对象中的属性来区分，在加载模型时是从文件加载，还是从内存加载，以及内存是由系统内部管理，还是由用户管理。

当前AscendCL还提供了以下接口实现模型加载的功能，从使用的接口上区分从文件加载，还是从内存加载，以及内存是由系统内部管理，还是由用户管理。

- [aclmdlLoadFromFile](#)接口
- [aclmdlLoadFromMem](#)接口
- [aclmdlLoadFromFileWithMem](#)接口
- [aclmdlLoadFromMemWithMem](#)接口
- [aclmdlLoadFromFileWithQ](#)接口
- [aclmdlLoadFromMemWithQ](#)接口

1.7.1.8 aclmdlLoadWithConfig

函数功能

指定模型加载时需要的配置信息，并进行模型加载。

本接口需要与以下其它接口配合，实现模型加载功能：

1. 调用[aclmdlCreateConfigHandle](#)接口创建模型加载的配置对象。
2. 多次调用[aclmdlSetConfigOpt](#)接口设置配置对象中每个属性的值。
3. 调用[aclmdlLoadWithConfig](#)接口指定模型加载时需要的配置信息，并进行模型加载。
4. 模型加载成功后，调用[aclmdlDestroyConfigHandle](#)接口销毁。

约束说明

在加载前，请先根据模型文件的大小评估内存空间是否足够，内存空间不足，会导致应用程序异常。

函数原型

aclError [aclmdlLoadWithConfig](#)(const [aclmdlConfigHandle](#) *handle, uint32_t *modelId)

参数说明

参数名	输入/输出	说明
handle	输入	模型加载的配置对象的指针。需提前调用 aclmdlCreateConfigHandle 接口创建该对象，与 aclmdlSetConfigOpt 中的handle保持一致。
modelId	输出	模型ID的指针。 系统成功加载模型后会返回的模型ID。

返回值说明

返回0表示成功，返回其他值表示失败。

相关接口

使用[aclmdlSetConfigOpt](#)接口、[aclmdlLoadWithConfig](#)接口时，是通过配置对象中的属性来区分，在加载模型时是从文件加载，还是从内存加载，以及内存是由系统内部管理，还是由用户管理。

当前AscendCL还提供了以下接口实现模型加载的功能，从使用的接口上区分从文件加载，还是从内存加载，以及内存是由系统内部管理，还是由用户管理。

- [aclmdlLoadFromFile](#)接口
- [aclmdlLoadFromMem](#)接口
- [aclmdlLoadFromFileWithMem](#)接口
- [aclmdlLoadFromMemWithMem](#)接口
- [aclmdlLoadFromFileWithQ](#)接口
- [aclmdlLoadFromMemWithQ](#)接口

1.7.1.9 aclmdlUnload

函数功能

系统完成模型推理后，可调用该接口卸载模型，释放资源。

约束说明

- 在调用[aclmdlUnload](#)接口卸载指定模型时，需确保其它接口没有正在使用该模型。
- 模型加载、模型执行、模型卸载的操作必须在同一个Context下（关于Context的创建请参见[1.5.1.1 aclrtSetDevice](#)或[1.5.2.1 aclrtCreateContext](#)）。

函数原型

[aclError](#) [aclmdlUnload](#)(uint32_t modelId)

参数说明

参数名	输入/输出	说明
modelId	输入	需卸载的模型的ID。

返回值说明

返回0表示成功，返回其他值表示失败。

参考资源

接口调用流程，参见AscendCL接口调用流程。

1.7.1.10 aclmdlBundleLoadFromFile

函数功能

模型执行阶段若涉及动态更新变量的场景，可调用本接口从文件加载离线模型数据（适配昇腾AI处理器的离线模型），由系统内部管理内存。

本接口需与以下其它接口配合使用，以便实现动态更新变量的目的，关键接口的调用流程如下：

1.

基于Ascend Graph方式编译并保存模型，模型中包含多个图，例如推理图、变量初始化图、变量更新图等。
此处是调用aclgrphBundleBuildModel接口编译模型、调用aclgrphBundleSaveModel接口保存模型，接口详细描述参见《[Ascend Graph开发指南](#)》。
2.

调用[aclmdlBundleLoadFromFile](#)或[aclmdlBundleLoadFromMem](#)接口加载模型。
3.

调用[aclmdlBundleGetModelId](#)接口获取多个图的ID。
4.

根据多个图的ID，分别调用模型执行接口（例如[aclmdlExecute](#)）执行各个图。
若涉及变量更新，则在执行变量更新图之前，先调用[aclmdlSetDatasetTensorDesc](#)接口设置图的tensor描述信息，再执行变量更新图，然后再执行一次推理图。
5.

推理结束后，调用[aclmdlBundleUnload](#)接口卸载模型。

函数原型

aclError [aclmdlBundleLoadFromFile](#)(const char *modelPath, uint32_t *bundleId)

参数说明

参数名	输入/输出	说明
modelPath	输入	模型文件路径的指针，路径中包含文件名。运行程序（APP）的用户需要对该存储路径有访问权限。 此处的模型文件是基于Ascend Graph方式构建出来的，调用aclgrphBundleBuildModel接口编译模型、调用aclgrphBundleSaveModel接口保存模型，接口详细描述参见《 Ascend Graph开发指南 》。
bundleId	输出	系统成功加载模型后，返回bundleId作为后续操作时识别模型的标志。

返回值说明

返回0表示成功，返回其他值表示失败。

参考资源

接口调用流程及示例代码请参见权重更新。

1.7.1.11 aclmdlBundleLoadFromMem

函数功能

模型执行阶段若涉及动态更新变量的场景，从内存加载离线模型数据，由系统内部管理模型运行的内存。

本接口需与以下其它接口配合使用，以便实现动态更新变量的目的，请参见[aclmdlBundleLoadFromFile](#)处的说明。

函数原型

aclError aclmdlBundleLoadFromMem(const void *model, size_t modelSize, uint32_t *bundleId)

参数说明

参数名	输入/输出	说明
model	输入	存放模型数据的内存地址指针。 此处的模型文件是基于Ascend Graph方式构建出来的，调用aclgrphBundleBuildModel接口编译模型、调用aclgrphBundleSaveModel接口保存模型，再由用户自行将保存出来的om模型文件读入内存，Ascend Graph接口详细描述参见《 Ascend Graph开发指南 》。 应用运行在Host时，此处需申请Host上的内存；应用运行在Device时，此处需申请Device上的内存。内存申请接口请参见 1.5.5 内存管理 。
modelSize	输入	内存中的模型数据长度，单位Byte。
bundleId	输出	系统成功加载模型后，返回bundleId作为后续操作时识别模型的标志。

返回值说明

返回0表示成功，返回其他值表示失败。

1.7.1.12 aclmdlBundleGetModelId

函数功能

获取实际可执行的模型ID。

本接口需与以下其它接口配合使用，以便实现动态更新变量的目的，请参见[aclmdlBundleLoadFromFile](#)处的说明。

函数原型

aclError aclmdlBundleGetModelId(uint32_t bundleId, size_t index, uint32_t *modelId)

参数说明

参数名	输入/输出	说明
bundleId	输入	通过 aclmdlBundleLoadFromFile 接口或 aclmdlBundleLoadFromMem 接口加载模型成功后返回的bundleId。
index	输入	索引。 用户调用 aclmdlBundleGetModelNum 接口获取模型数量后，这个index的取值范围：[0, (模型数量-1)]。
modelId	输出	实际可执行的模型ID。

返回值说明

返回0表示成功，返回其他值表示失败。

参考资源

接口调用流程及示例代码请参见[权重更新](#)。

1.7.1.13 aclmdlBundleGetModelNum

函数功能

根据bundleId获取实际可执行的模型数。

函数原型

aclError aclmdlBundleGetModelNum(uint32_t bundleId, size_t *modelNum)

参数说明

参数名	输入/输出	说明
bundleId	输入	通过 aclmdlBundleLoadFromFile 接口或 aclmdlBundleLoadFromMem 接口加载模型成功后返回的bundleId。
modelNum	输出	实际可执行的模型数。

返回值说明

返回0表示成功，返回其他值表示失败。

1.7.1.14 aclmdlBundleUnload

函数功能

系统完成模型推理后，可调用该接口卸载通过[aclmdlBundleLoadFromFile](#)接口或[aclmdlBundleLoadFromMem](#)接口加载的模型，释放资源。

本接口需与以下其它接口配合使用，以便实现动态更新变量的目的，请参见[aclmdlBundleLoadFromFile](#)处的说明。

函数原型

aclError [aclmdlBundleUnload](#)(uint32_t bundleId)

参数说明

参数名	输入/输出	说明
bundleId	输入	通过 aclmdlBundleLoadFromFile 接口或 aclmdlBundleLoadFromMem 接口加载模型成功后返回的bundleId。

返回值说明

返回0表示成功，返回其他值表示失败。

参考资源

接口调用流程及示例代码请参见[权重更新](#)。

1.7.2 模型执行

1.7.2.1 aclmdlExecute

函数功能

执行模型推理，直到返回推理结果。

约束说明

- 若由于业务需求，必须在多线程中使用同一个modelId，则用户线程间需加锁，保证刷新输入输出内存、保证执行是连续操作，例如：

```
// 线程A的接口调用顺序：
lock(handle1) -> aclrtMemcpy刷新输入输出内存 -> aclmdlExecute执行推理 -> unlock(handle1)

// 线程B的接口调用顺序：
lock(handle1) -> aclrtMemcpy刷新输入输出内存 -> aclmdlExecute执行推理 -> unlock(handle1)
```

- 模型加载、模型执行、模型卸载的操作必须在同一个Context下（关于Context的创建请参见[1.5.1.1 aclrtSetDevice](#)或[1.5.2.1 aclrtCreateContext](#)）。
- 存放模型输入/输出数据的Device内存，可以使用以下接口申请：[aclrtMalloc](#)、或[aclrtMallocHost](#)、或[aclrtMallocCached](#)接口、或[acldvppMalloc](#)接口、或[hi_mpi_dvpp_malloc](#)接口。

其中：

- 各内存申请接口的使用场景、使用约束请参见各内存申请接口的说明。
- [aclrtMallocHost](#)接口仅在如下场景才能申请Device内存：仅当AscendCL应用程序在Device上运行时，申请的才是Device内存。
- [acldvppMalloc](#)接口和[hi_mpi_dvpp_malloc](#)接口是媒体数据处理功能专用的内存申请接口，一般从性能角度，为了减少拷贝，媒体数据处理的输出作为模型推理的输入，实现内存复用。
- 由于硬件对内存有对齐和补齐要求，若用户使用这些接口申请大块内存并自行划分、管理内存时，需满足对应接口的对齐和补齐约束，请参见内存二次分配管理。

函数原型

```
aclError aclmdlExecute(uint32_t modelId, const aclmdlDataset *input,
aclmdlDataset *output)
```

参数说明

参数名	输入/输出	说明
modelId	输入	指定推理模型的ID。 调用模型加载接口（例如 aclmdlLoadFromFile 接口、 aclmdlLoadFromMem 等）成功后，会返回模型ID，该ID作为本接口的输入。
input	输入	模型推理的输入数据的指针。

参数名	输入/输出	说明
output	输出	模型推理输出数据的指针。 调用aclCreateDataBuffer接口创建存放对应index输出数据的aclDataBuffer类型时，支持在data参数处传入nullptr，同时size需设置为0，表示创建一个空的aclDataBuffer类型，然后在模型执行过程中，系统内部自行计算并申请该index输出的内存。使用该方式可节省内存，但内存数据使用结束后，需由用户释放内存并重置aclDataBuffer，同时，系统内部申请内存时涉及内存拷贝，可能涉及性能损耗。 释放内存并重置aclDataBuffer的示例代码如下： <pre>aclDataBuffer *dataBuffer = aclmdlGetDatasetBuffer(output, 0); // 根据index获取对应的dataBuffer void *data = aclGetDataBufferAddr(dataBuffer); // 获取data的Device指针 aclrtFree(data); // 释放Device内存 aclUpdateDataBuffer(dataBuffer, nullptr, 0); // 重置dataBuffer里面内容，以便下次推理</pre>

返回值说明

返回0表示成功，返回其他值表示失败。

参考资源

接口调用流程及示例代码，参见模型执行。

1.7.2.2 aclmdlExecuteAsync

函数功能

执行模型推理，异步接口。

约束说明

- 该接口是异步接口，调用接口成功仅表示任务下发成功，不表示任务执行成功。调用该接口后，需调用同步等待接口（例如，[aclrtSynchronizeStream](#)）确保任务已执行完成，否则可能会导致训练或推理等业务异常、Device断链掉卡等未知情况。
- 对同一个modelId的模型，不能调用[aclmdlExecuteAsync](#)接口执行多Stream并发场景下的模型推理。错误示例如下，该示例中，两次[aclmdlExecuteAsync](#)接口多Stream并发执行，导致报错：

```
//.....
aclmdlExecuteAsync(modelId1, input, output, stream1);
aclmdlExecuteAsync(modelId1, input, output, stream2);
aclrtSynchronizeStream(stream1);
aclrtSynchronizeStream(stream2);
//.....
```
- 若由于业务需求，必须在多线程中使用同一个modelId，则用户线程间需加锁，保证刷新输入输出内存、保证执行是连续操作，例如：

```
// 线程A的接口调用顺序：
lock(handle1) -> aclrtMemcpyAsync(stream1)刷新输入输出内存 ->
```

```
aclmdlExecuteAsync(modelId1,stream1)执行推理 -> unlock(handle1)

// 线程B的接口调用顺序:
lock(handle1) -> aclrtMemcpyAsync(stream1)刷新输入输出内存 ->
aclmdlExecuteAsync(modelId1,stream1)执行推理 -> unlock(handle1)
```

- 模型加载、模型执行、模型卸载的操作必须在同一个Context下（关于Context的创建请参见[1.5.1.1 aclrtSetDevice](#)或[1.5.2.1 aclrtCreateContext](#)）。
- 存放模型输入/输出数据的内存，可以使用以下接口申请：[aclrtMalloc](#)、或[aclrtMallocHost](#)、或[aclrtMallocCached](#)接口、或[acldvppMalloc](#)接口、或[hi_mpi_dvpp_malloc](#)接口。

其中，[acldvppMalloc](#)接口和[hi_mpi_dvpp_malloc](#)接口是媒体数据处理功能专用的内存申请接口，一般从性能角度，为了减少拷贝，媒体数据处理的输出作为模型推理的输入，实现内存复用。

由于硬件对内存有对齐和补齐要求，若用户使用这些接口申请大块内存并自行划分、管理内存时，需满足对应接口的对齐和补齐约束，请参见内存二次分配管理。

- 若需要使用外置Allocator，则注册Allocator时的stream需与模型执行时的stream保持一致。

函数原型

```
aclError aclmdlExecuteAsync(uint32_t modelId, const aclmdlDataset *input,
aclmdlDataset *output, aclrtStream stream)
```

参数说明

参数名	输入/输出	说明
modelId	输入	指定需要执行推理的模型的ID。 调用 aclmdlLoadFromFile 接口/ aclmdlLoadFromMem 接口/ aclmdlLoadFromFileWithMem 接口/ aclmdlLoadFromMemWithMem 接口加载模型成功后，会返回模型ID。
input	输入	模型推理的输入数据的指针。
output	输出	模型推理的输出数据的指针。
stream	输入	指定Stream。

返回值说明

返回0表示成功，返回其他值表示失败。

参考资源

接口调用流程及示例代码，参见异步模型推理。

1.7.2.3 aclmdlSetExecConfigOpt

函数功能

设置模型执行的配置对象中的各属性的取值。

约束说明

- 本接口需要配合其它接口一起使用，实现模型执行，接口调用顺序如下：
- 1. 调用[aclmdlCreateExecConfigHandle](#)接口创建模型执行的配置对象。
 - 2. 多次调用[aclmdlSetExecConfigOpt](#)接口设置配置对象中每个属性的值。
 - 3. 调用[aclmdlExecuteV2](#)或[aclmdlExecuteAsyncV2](#)接口指定模型执行时需要的配置信息，并进行模型执行。
 - 4. 模型执行成功后，调用[aclmdlDestroyExecConfigHandle](#)接口销毁。

函数原型

aclError [aclmdlSetExecConfigOpt](#)([aclmdlExecConfigHandle](#) *handle,
[aclmdlExecConfigAttr](#) attr, const void *attrValue, size_t valueSize)

参数说明

参数名	输入/输出	说明
handle	输出	模型执行的配置对象的指针。需提前调用 aclmdlCreateExecConfigHandle 接口创建该对象。
attr	输入	指定需设置的属性。
attrValue	输入	指向属性值的指针，attr对应的属性取值。 如果属性值本身是指针，则传入该指针的地址。
valueSize	输入	attrValue部分的数据长度。 用户可使用C/C++标准库的函数sizeof(*attrValue)查询数据长度。

返回值说明

返回0表示成功，返回其他值表示失败。

1.7.2.4 aclmdlExecuteV2

函数功能

根据[aclmdlSetExecConfigOpt](#)接口所配置的属性，执行模型推理，直到返回推理结果。该接口是在[aclmdlExecute](#)接口基础上进行了增强，支持在执行模型推理时控制Stream任务的超时时间、Event任务的超时时间。

约束说明

- 本接口需要配合其它接口一起使用，实现模型执行，接口调用顺序如下：
 - a. 调用[aclmdlCreateExecConfigHandle](#)接口创建模型执行的配置对象。
 - b. 多次调用[aclmdlSetExecConfigOpt](#)接口设置配置对象中每个属性的值。
 - c. 调用[aclmdlExecuteV2](#)接口指定模型执行时需要的配置信息，并进行模型执行。
 - d. 模型加载成功后，调用[aclmdlDestroyExecConfigHandle](#)接口销毁。
- 其他约束与[aclmdlExecute](#)一致。

函数原型

aclError [aclmdlExecuteV2](#)(uint32_t modelId, const [aclmdlDataset](#) *input, [aclmdlDataset](#) *output, [aclrtStream](#) stream, const [aclmdlExecConfigHandle](#) *handle)

参数说明

参数名	输入/输出	说明
modelId	输入	指定需要执行推理的模型的ID。 调用 aclmdlLoadFromFile 接口/ aclmdlLoadFromMem 接口/ aclmdlLoadFromFileWithMem 接口/ aclmdlLoadFromMemWithMem 接口加载模型成功后，会返回模型ID。
input	输入	模型推理的输入数据的指针。
output	输出	模型推理的输出数据的指针。 调用 aclCreateDataBuffer 接口创建存放对应index输出数据的 aclDataBuffer 类型时，支持在data参数处传入nullptr，同时size需设置为0，表示创建一个空的 aclDataBuffer 类型，然后在模型执行过程中，系统内部自行计算并申请该index输出的内存。使用该方式可节省内存，但内存数据使用结束后，需由用户释放内存并重置 aclDataBuffer ，同时，系统内部申请内存时涉及内存拷贝，可能涉及性能损耗。 释放内存并重置 aclDataBuffer 的示例代码如下： aclDataBuffer *dataBuffer = aclmdlGetDatasetBuffer (output, 0); // 根据index获取对应的dataBuffer void *data = aclGetDataBufferAddr (dataBuffer); // 获取data的Device指针 aclrtFree (data); // 释放Device内存 aclUpdateDataBuffer (dataBuffer, nullptr, 0); // 重置dataBuffer里面内容，以便下次推理
stream	输入	指定Stream。
handle	输入	模型执行的配置对象的指针。与 aclmdlSetExecConfigOpt 中的handle保持一致。

返回值说明

返回0表示成功，返回其他值表示失败。

1.7.2.5 aclmdlExecuteAsyncV2

- Atlas 200/300/500 推理产品不支持该接口。
- Atlas 200/500 A2推理产品不支持该接口。
- Atlas 训练系列产品不支持该接口。
- Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。
- Atlas 推理系列产品，不支持该接口。

函数功能

根据[aclmdlSetExecConfigOpt](#)所配置的属性，执行模型推理，直到返回推理结果。该接口是在[aclmdlExecuteAsync](#)接口基础上进行了增强，支持在执行模型推理时控制Stream任务的超时时间、Event任务的超时时间。异步接口。

约束说明

- 本接口需要配合其它接口一起使用，实现模型执行，接口调用顺序如下：
 - a. 调用[aclmdlCreateExecConfigHandle](#)接口创建模型执行的配置对象。
 - b. 多次调用[aclmdlSetExecConfigOpt](#)接口设置配置对象中每个属性的值。
 - c. 调用[aclmdlExecuteAsyncV2](#)接口指定模型执行时需要的配置信息，并进行模型执行。
 - d. 模型加载成功后，调用[aclmdlDestroyExecConfigHandle](#)接口销毁。
- 该接口是异步接口，调用接口成功仅表示任务下发成功，不表示任务执行成功。调用该接口后，需调用同步等待接口（例如，[aclrtSynchronizeStream](#)）确保任务已执行完成，否则可能会导致训练或推理等业务异常、Device断链掉卡等未知情况。
- 其他约束与[aclmdlExecuteAsync](#)一致。

函数原型

aclError [aclmdlExecuteAsyncV2](#)(uint32_t modelId, const [aclmdlDataset](#) *input, [aclmdlDataset](#) *output, [aclrtStream](#) stream, const [aclmdlExecConfigHandle](#) *handle)

参数说明

参数名	输入/输出	说明
modelId	输入	指定需要执行推理的模型的ID。 调用 aclmdlLoadFromFile 接口/ aclmdlLoadFromMem 接口/ aclmdlLoadFromFileWithMem 接口/ aclmdlLoadFromMemWithMem 接口加载模型成功后，会返回模型ID。

参数名	输入/输出	说明
input	输入	模型推理的输入数据的指针。
output	输出	模型推理的输出数据的指针。
stream	输入	指定Stream。
handle	输入	模型执行的配置对象的指针。与 aclmdlSetExecConfigOpt 中的handle保持一致。

返回值说明

返回0表示成功，返回其他值表示失败。

1.7.2.6 aclmdlQuerySize

函数功能

根据模型文件获取模型执行时所需的权值内存大小、工作内存大小。

约束说明

工作和权重内存为Device内存，而且需要用户申请和释放。

当由用户管理内存时，为确保内存不浪费，在申请工作内存、权值内存前，需要调用本接口查询模型运行时所需工作内存、权值内存的大小。

如果模型输入数据的Shape不确定，则不能调用[aclmdlQuerySize](#)接口查询内存大小，在加载模型时，就无法由用户管理内存，因此需选择由系统管理内存的模型加载接口（例如，[aclmdlLoadFromFile](#)、[aclmdlLoadFromMem](#)）。

函数原型

[aclError](#) [aclmdlQuerySize](#)(const char *fileName, size_t *workSize, size_t *weightSize)

参数说明

参数名	输入/输出	说明
fileName	输入	模型文件路径的指针，路径中包含文件名。运行程序（APP）的用户需要对该路径有访问权限。 此处的模型文件是适配昇腾AI处理器的离线模型，即*.om文件。 说明 关于如何获取om文件，请参见《 ATC工具使用指南 》。
workSize	输出	模型执行时所需的工作内存大小的指针，单位Byte。

参数名	输入/输出	说明
weightSize	输出	模型执行时所需权值内存大小的指针，单位Byte。

返回值说明

返回0表示成功，返回其他值表示失败。

参考资源

接口调用流程及示例代码，参见模型加载。

1.7.2.7 aclmdlQuerySizeFromMem

函数功能

根据内存中的模型数据获取模型执行时所需的权值内存大小、内存大小。

约束说明

工作和权重内存为Device内存，而且需要用户申请和释放。

当由用户管理内存时，为确保内存不浪费，在申请工作内存、权值内存前，需要调用本接口查询模型运行时所需工作内存、权值内存的大小。

如果模型输入数据的Shape不确定，则不能调用本接口查询内存大小，在加载模型时，就无法由用户管理内存，因此需选择由系统管理内存的模型加载接口（例如，[aclmdlLoadFromFile](#)、[aclmdlLoadFromMem](#)）。

函数原型

aclError aclmdlQuerySizeFromMem(const void *model, size_t modelSize, size_t *workSize, size_t *weightSize)

参数说明

参数名	输入/输出	说明
model	输入	模型数据的指针。
modelSize	输入	模型数据长度，单位Byte。
workSize	输出	模型执行时所需的工作内存大小的指针，单位Byte。
weightSize	输出	模型执行时所需权值内存大小的指针，单位Byte。

返回值说明

返回0表示成功，返回其他值表示失败。

1.7.2.8 aclmdlQueryExeOMDesc

Atlas 200/300/500 推理产品不支持该接口。

Atlas 200/500 A2推理产品不支持该接口。

Atlas 训练系列产品不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

函数功能

根据模型文件获取模型执行时所需的工作内存、权值内存、模型描述信息、静态和动态shape任务等的内存大小。

函数原型

aclError aclmdlQueryExeOMDesc(const char *fileName, **aclmdlExeOMDesc** *mdlPartitionSize)

参数说明

参数名	输入/输出	说明
fileName	输入	模型文件路径的指针，路径中包含文件名。运行程序（APP）的用户需要对该路径有访问权限。 此处的模型文件是适配昇腾AI处理器的离线模型，即或*.exeom文件。 说明 关于如何获取exeom文件，请参见《 ATC工具使用指南 》。
mdlPartitionSize	输出	模型执行时所需的各分区大小的结构体指针，分区大小单位为Byte。

返回值说明

返回0表示成功，返回其他值表示失败。

1.7.2.9 aclmdlSetDynamicBatchSize

函数功能

在动态Batch场景下，在模型执行前调用本接口设置每次模型推理时处理图片的数量。

函数原型

aclError aclmdlSetDynamicBatchSize(uint32_t modelId, **aclmdlDataset** *dataset, size_t index, uint64_t batchSize)

参数说明

参数名	输入/输出	说明
modelId	输入	模型ID。 调用 aclmdlLoadFromFile 接口/ aclmdlLoadFromMem 接口/ aclmdlLoadFromFileWithMem 接口/ aclmdlLoadFromMemWithMem 接口加载模型成功后，会返回模型ID。
dataset	输入&输出	模型推理的输入数据的指针。 使用 aclmdlDataset 类型的数据描述模型推理时的输入数据，输入的内存地址、内存大小用 aclDataBuffer 类型的数据来描述。
index	输入	标识动态Batch输入的输入index，需调用 aclmdlGetInputIndexByName 接口获取，输入名称固定为ACL_DYNAMIC_TENSOR_NAME。
batchSize	输入	每次模型推理时处理图片的数量。 此处设置的batch size只能是模型构建时设置的BatchSize档位中的某一档。 如果不清楚模型构建时的Batch档位设置，也可以调用 aclmdlGetDynamicBatch 接口获取指定模型支持的Batch档位数以及每一档中的batch size。

返回值说明

返回0表示成功，返回其他值表示失败。

参考资源

接口调用流程及示例，参见模型动态Shape输入推理。

1.7.2.10 aclmdlSetDynamicHWSIZE

函数功能

动态分辨率场景下，在模型执行前调用本接口设置模型推理时输入图片的高和宽。

函数原型

aclError [aclmdlSetDynamicHWSIZE](#)([uint32_t](#) modelId, [aclmdlDataset](#) *dataset, [size_t](#) index, [uint64_t](#) height, [uint64_t](#) width)

参数说明

参数名	输入/输出	说明
modelId	输入	模型ID。 调用 aclmdlLoadFromFile 接口/ aclmdlLoadFromMem 接口/ aclmdlLoadFromFileWithMem 接口/ aclmdlLoadFromMemWithMem 接口加载模型成功后，会返回模型ID。
dataset	输入&输出	模型推理的输入数据的指针。 使用 aclmdlDataset 类型的数据描述模型推理时的输入数据，输入的内存地址、内存大小用 aclDataBuffer 类型的数据来描述。
index	输入	标识动态分辨率输入的输入index，需调用 aclmdlGetInputIndexByName 接口获取，输入名称固定为ACL_DYNAMIC_TENSOR_NAME。
height	输入	需设置的H值。 此处设置的分辨率只能是模型构建时设置的分辨率档位中的某一档。 如果不清楚模型构建时的分辨率档位，也可以调用 aclmdlGetDynamicHW 接口获取指定模型支持的分辨率档位数以及每一档中的宽、高。
width	输入	需设置的W值。 此处设置的分辨率只能是模型构建时设置的分辨率档位中的某一档。 如果不清楚模型构建时的分辨率档位，也可以调用 aclmdlGetDynamicHW 接口获取指定模型支持的分辨率档位数以及每一档中的宽、高。

返回值说明

返回0表示成功，返回其他值表示失败。

参考资源

接口调用流程及示例，参见模型动态Shape输入推理。

1.7.2.11 aclmdlSetInputAIPP

函数功能

动态AIPP场景下，根据指定的动态AIPP输入的输入index，设置模型推理时的AIPP参数值。

动态AIPP支持的几种操作的计算方式及其计算顺序如下：抠图->色域转换->缩放（当前版本不支持缩放）->减均值/归一化->padding。

约束说明

- 经过动态AIPP处理后的图像的宽、高必须与原始模型中输入Shape中的宽、高保持一致。
- 多Batch场景下，根据每个Batch的配置计算出动态AIPP后输出图片的宽、高，经过动态AIPP后每个Batch的输出图片宽、高必须是一致的。计算输出图片宽、高的计算公式如表1-3所示。
- 抠图或者缩放或者padding之后，对图片宽、高的校验规则如下，其中，aippOutputW、aippOutputH分别表示AIPP输出图片的宽、高，其它参数是[aclmdlSetAIPPSrcImageSize](#)、[aclmdlSetAIPPScfParams](#)、[aclmdlSetAIPPCropParams](#)、[aclmdlSetAIPPPaddingParams](#)接口的入参：

表 1-3 输出图片宽、高计算公式

抠图	缩放	补边 (padding)	动态AIPP输出图片的宽、高
否	否	否	aippOutputW=srcImageSizeW , aippOutputH=srcImageSizeH
是	否	否	aippOutputW=cropSizeW , aippOutputH=cropSizeH
是	是	否	aippOutputW=scfOutputSize W , aippOutputH=scfOutputSizeH
是	否	是	aippOutputW=cropSizeW + paddingSizeLeft + paddingSizeRight , aippOutputH=cropSizeH + paddingSizeTop + paddingSizeBottom
否	否	是	aippOutputW=srcImageSizeW + paddingSizeLeft + paddingSizeRight , aippOutputH=srcImageSizeH + paddingSizeTop + paddingSizeBottom
否	是	是	aippOutputW=scfOutputSize W + paddingSizeLeft + paddingSizeRight , aippOutputH=scfOutputSizeH + paddingSizeTop + paddingSizeBottom
否	是	否	aippOutputW=scfOutputSize W , aippOutputH=scfOutputSizeH

抠图	缩放	补边 (padding)	动态AIPP输出图片的宽、高
是	是	是	aippOutputW=scfOutputSizeW + paddingSizeLeft + paddingSizeRight, aippOutputH=scfOutputSizeH + paddingSizeTop + paddingSizeBottom

函数原型

```
aclError aclmdlSetInputAIPP(uint32_t modelId, aclmdlDataset *dataset, size_t index, const aclmdlAIPP *aippParmsSet)
```

参数说明

参数名	输入/输出	说明
modelId	输入	模型的ID。 调用 aclmdlLoadFromFile 接口/ aclmdlLoadFromMem 接口/ aclmdlLoadFromFileWithMem 接口/ aclmdlLoadFromMemWithMem 接口加载模型成功后，会返回模型ID。
dataset	输入	模型推理的输入数据的指针。 使用 aclmdlDataset 类型的数据描述模型推理时的输入数据，输入的内存地址、内存大小用 aclDataBuffer 类型的数据来描述。
index	输入	标识动态AIPP输入的输入index。 多个动态AIPP输入的场景下，用户可调用 aclmdlGetAippType 接口获取指定模型输入所关联的动态AIPP输入的输入index。 为保证向前兼容，如果明确只有一个动态AIPP输入，可调用 aclmdlGetInputIndexByName 接口获取，输入名称固定为ACL_DYNAMIC_AIPP_NAME。
aippParmsSet	输入	动态AIPP参数对象的指针。 提前调用 aclmdlCreateAIPP 接口创建 aclmdlAIPP 类型的数据。

返回值说明

返回0表示成功，返回其他值表示失败。

参考资源

接口调用流程及示例，参见模型动态AIPP推理。

1.7.2.12 aclmdlGetFirstAippInfo

函数功能

获取模型AIPP（包括静态AIPP和动态AIPP）的配置信息。

AIPP支持的几种操作的计算方式及其计算顺序如下：抠图->色域转换->缩放（当前版本不支持缩放）->减均值/归一化->padding。

约束说明

如果使用本接口获取模型中动态AIPP的信息，只能获取[aclAippInfo](#)结构体中如下参数的值：srcFormat、srcDatatype、srcDimNum、shapeCount、outDims，其它参数值无效。

函数原型

```
aclError aclmdlGetFirstAippInfo(uint32_t modelId, size_t index, aclAippInfo *aippInfo)
```

参数说明

参数名	输入/输出	说明
modelId	输入	模型ID。 调用 aclmdlLoadFromFile 接口/ aclmdlLoadFromMem 接口/ aclmdlLoadFromFileWithMem 接口/ aclmdlLoadFromMemWithMem 接口加载模型成功后，会返回模型ID。
index	输入	模型中输入的index。
aippInfo	输出	获取指定输入上AIPP配置信息的指针。 详细说明及参数解释，请参考《 ATC工具使用指南 》。

返回值说明

返回0表示成功，返回非0表示失败。

1.7.2.13 aclmdlGetAippType

函数功能

获取指定模型的指定输入所支持的AIPP类型（动态AIPP或静态AIPP）及动态AIPP输入对应的index值。

函数原型

```
aclError aclmdlGetAippType(uint32_t modelId, size_t index,
aclmdlInputAippType *type, size_t *dynamicAttachedDataIndex)
```

参数说明

参数名	输入/输出	说明
modelId	输入	指定模型的ID。 调用 aclmdlLoadFromFile 接口/ aclmdlLoadFromMem 接口/ aclmdlLoadFromFileWithMem 接口/ aclmdlLoadFromMemWithMem 接口加载模型 成功后，会返回模型ID。
index	输入	模型中输入的index。
type	输出	指定模型输入的AIPP类型的指针。
dynamicAttachedDataIndex	输出	当type不为ACL_DATA_WITH_DYNAMIC_AIPP 时，该值返回0xFFFFFFFF，表示无效。 当type为ACL_DATA_WITH_DYNAMIC_AIPP时， 该值返回动态AIPP输入（用于配置动态AIPP参 数）的index。

返回值说明

返回0表示成功，返回其他值表示失败。

参考资源

接口调用示例，参见[动态AIPP（多个动态AIPP输入）](#)。

1.7.2.14 aclmdlSetAIPPByInputIndex

函数功能

动态AIPP场景下，根据指定的模型输入的输入index，设置模型推理时的AIPP参数值。
动态AIPP支持的几种操作的计算方式及其计算顺序如下：抠图->色域转换->缩放（当前版本不支持缩放）->减均值/归一化->padding。

约束说明

- 经过动态AIPP处理后的图像的宽、高必须与原始模型中输入Shape中的宽、高保持一致。
- 多Batch场景下，根据每个Batch的配置计算出动态AIPP后输出图片的宽、高，经过动态AIPP后每个Batch的输出图片宽、高必须是一致的。计算输出图片宽、高的计算公式如[表1-4](#)所示。

- 抠图或者缩放或者padding之后，对图片宽、高的校验规则如下，其中，aippOutputW、aippOutputH分别表示AIPP输出图片的宽、高，其它参数是[aclmdlSetAIPPSrcImageSize](#)、[aclmdlSetAIPPScfParams](#)、[aclmdlSetAIPPCropParams](#)、[aclmdlSetAIPPPaddingParams](#)接口的入参：

表 1-4 输出图片宽、高计算公式

抠图	缩放	补边 (padding)	动态AIPP输出图片的宽、高
否	否	否	aippOutputW=srcImageSizeW , aippOutputH=srcImageSizeH
是	否	否	aippOutputW=cropSizeW , aippOutputH=cropSizeH
是	是	否	aippOutputW=scfOutputSize W , aippOutputH=scfOutputSizeH
是	否	是	aippOutputW=cropSizeW + paddingSizeLeft + paddingSizeRight , aippOutputH=cropSizeH + paddingSizeTop + paddingSizeBottom
否	否	是	aippOutputW=srcImageSizeW + paddingSizeLeft + paddingSizeRight , aippOutputH=srcImageSizeH + paddingSizeTop + paddingSizeBottom
否	是	是	aippOutputW=scfOutputSize W + paddingSizeLeft + paddingSizeRight , aippOutputH=scfOutputSizeH + paddingSizeTop + paddingSizeBottom
否	是	否	aippOutputW=scfOutputSize W , aippOutputH=scfOutputSizeH
是	是	是	aippOutputW=scfOutputSize W + paddingSizeLeft + paddingSizeRight , aippOutputH=scfOutputSizeH + paddingSizeTop + paddingSizeBottom

函数原型

```
aclError aclmdlSetAIPPByInputIndex(uint32_t modelId, aclmdlDataset
*dataset, size_t index, const aclmdlAIPP *aippParmsSet)
```

参数说明

参数名	输入/输出	说明
modelId	输入	模型的ID。 调用 aclmdlLoadFromFile 接口/ aclmdlLoadFromMem 接口/ aclmdlLoadFromFileWithMem 接口/ aclmdlLoadFromMemWithMem 接口加载模型成功后，会返回模型ID。
dataset	输入	模型推理的输入数据的指针。 使用 aclmdlDataset 类型的数据描述模型推理时的输入数据，输入的内存地址、内存大小用 aclDataBuffer 类型的数据来描述。
index	输入	表示在第几个输入上设置动态AIPP参数。如果该输入没有关联动态AIPP输入，则返回报错。 可调用 aclmdlGetAippType 查询指定模型的指定输入是否有关联的动态AIPP输入，若有，则本接口的index参数值与 aclmdlGetAippType 接口的index参数值保持一致；若无，则无需设置动态AIPP参数。
aippParmsSet	输入	动态AIPP参数对象的指针。 提前调用 aclmdlCreateAIPP 接口创建 aclmdlAIPP 类型的数据。

返回值说明

返回0表示成功，返回其他值表示失败。

参考资源

接口调用示例，参见[动态AIPP（多个动态AIPP输入）](#)。

1.7.2.15 aclmdlSetInputDynamicDims

函数功能

如果模型输入的Shape是动态的、输入数据Format为ND格式（ND表示支持任意格式），在模型执行前调用本接口设置模型推理时具体维度的值。

函数原型

aclError **aclmdlSetInputDynamicDims**(uint32_t modelId, **aclmdlDataset** *dataset, size_t index, const **aclmdlIODims** *dims)

参数说明

参数名	输入/输出	说明
modelId	输入	模型ID。 调用 aclmdlLoadFromFile 接口/ aclmdlLoadFromMem 接口/ aclmdlLoadFromFileWithMem 接口/ aclmdlLoadFromMemWithMem 接口加载模型成功后，会返回模型ID。
dataset	输入&输出	模型推理的输入数据的指针。 使用 aclmdlDataset 类型的数据描述模型推理时的输入数据，输入的内存地址、内存大小用 aclDataBuffer 类型的数据来描述。
index	输入	标识动态维度的输入index。 需调用 aclmdlGetInputIndexByName 接口获取，输入名称固定为ACL_DYNAMIC_TENSOR_NAME。
dims	输入	具体某一档上的所有维度信息的指针。 此处设置的动态维度的值只能是模型构建时设置的档位中的某一档。 如果不清楚模型构建时的动态维度档位，也可以调用 aclmdlGetInputDynamicDims 接口获取指定模型支持的动态维度档位数以及每一档中的值。 例如： 使用ATC工具进行模型转换时， input_shape="data:1,1,40,-1;label:1,-1;mask:-1,-1"， dynamic_dims="20,20,1,1; 40,40,2,2; 80,60,4,4"，若输入数据的真实维度为（1,1,40,20,1,20,1,1），则dims结构体信息的填充示例如下（name暂不使用）： dims.dimCount = 8 dims.dims[0] = 1 dims.dims[1] = 1 dims.dims[2] = 40 dims.dims[3] = 20 dims.dims[4] = 1 dims.dims[5] = 20 dims.dims[6] = 1 dims.dims[7] = 1

返回值说明

返回0表示成功，返回其他值表示失败。

参考资源

接口调用流程及示例，参见模型动态Shape输入推理。

1.7.2.16 aclmdlCreateAndGetOpDesc

函数功能

获取指定算子的描述信息，包括算子名、输入tensor描述、输出tensor描述，如果查询不到指定算子，则返回报错。该接口不支持动态Shape场景。

约束说明

使用场景举例：执行整网模型推理时（**不支持动态Shape场景**），如果产生AI Core报错，可以调用本接口获取报错算子的描述信息，再做进一步错误排查。

推荐的接口调用顺序如下：

1. 定义并实现异常回调函数fn(aclrtExceptionInfoCallback类型)，回调函数原型请参见[1.5.7.1 aclrtSetExceptionInfoCallback](#)。

实现回调函数的关键步骤如下：

- a. 在异常回调函数fn内调用[aclrtGetDeviceldFromExceptionInfo](#)、[aclrtGetStreamIdFromExceptionInfo](#)、[aclrtGetTaskIdFromExceptionInfo](#)接口分别获取Device ID、Stream ID、Task ID。
- b. 在异常回调函数fn内调用[aclmdlCreateAndGetOpDesc](#)接口获取算子的描述信息。
- c. 在异常回调函数fn内调用[aclGetTensorDescByIndex](#)接口获取指定算子输入/输出的tensor描述。
- d. 在异常回调函数fn内调用如下接口获取tensor描述中的数据，进行进一步分析。

例如，调用[aclGetTensorDescAddress](#)接口获取tensor数据的内存地址（用户可从该内存地址中获取tensor数据）、调用[aclGetTensorDescType](#)接口获取tensor描述中的数据类型、调用[aclGetTensorDescFormat](#)接口获取tensor描述中的Format、调用[aclGetTensorDescNumDims](#)接口获取tensor描述中的Shape维度个数、调用[aclGetTensorDescDimV2](#)接口获取Shape中指定维度的大小。

2. 调用[aclrtSetExceptionInfoCallback](#)接口设置异常回调函数。
3. 执行模型推理。

如果存在AI Core报错，则触发回调函数fn，获取算子的信息，进行进一步分析。

函数原型

```
aclError aclmdlCreateAndGetOpDesc(uint32_t deviceId, uint32_t streamId,
uint32_t taskId, char *opName, size_t opNameLen, aclTensorDesc **inputDesc,
size_t *numInputs, aclTensorDesc **outputDesc, size_t *numOutputs)
```

参数说明

参数名	输入/输出	说明
deviceId	输入	Device ID。 调用 aclrtGetDeviceIdFromExceptionInfo 接口获取异常信息中的Device ID，作为本接口的输入。
streamId	输入	Stream ID。 调用 aclrtGetStreamIdFromExceptionInfo 接口获取异常信息中的Stream ID，作为本接口的输入。
taskId	输入	Task ID。 调用 aclrtGetTaskIdFromExceptionInfo 接口获取异常信息中的Task ID，作为本接口的输入。
opName	输出	算子名称字符串的指针。
opNameLen	输入	算子名称字符串长度。 若用户指定的长度比实际算子名称的长度短，则返回报错。
inputDesc	输出	算子所有输入的tensor描述的指针，指向一块连续内存的首地址。
numInputs	输出	输入个数的指针。
outputDesc	输出	算子所有输出的tensor描述的指针，指向一块连续内存的首地址。
numOutputs	输出	输出个数的指针。

返回值说明

返回0表示成功，返回其他值表示失败。

参考资源

接口调用示例，参见AI Core异常信息获取。

1.7.2.17 aclmdlInitDump

函数功能

Dump初始化。

本接口需与其它接口配合使用实现以下功能：

- **Dump数据落盘到文件**
[aclmdlInitDump](#)接口、[aclmdlSetDump](#)接口、[aclmdlFinalizeDump](#)接口配合使用，用于将Dump数据记录到文件中。一个进程内，可以根据需求多次调用这些接口，基于不同的Dump配置信息，获取Dump数据。场景举例如下：

- 执行两个不同的模型，需要设置不同的Dump配置信息，接口调用顺序：
aclInit接口-->**aclmdlInitDump**接口-->**aclmdlSetDump**接口-->模型1加载-->模型1执行-->**aclmdlFinalizeDump**接口-->模型1卸载-->**aclmdlInitDump**接口-->**aclmdlSetDump**接口-->模型2加载-->模型2执行-->**aclmdlFinalizeDump**接口-->模型2卸载-->执行其它任务-->**aclFinalize**接口
- 同一个模型执行两次，第一次需要Dump，第二次无需Dump，接口调用顺序：**aclInit**接口-->**aclmdlInitDump**接口-->**aclmdlSetDump**接口-->模型加载-->模型执行-->**aclmdlFinalizeDump**接口-->模型卸载-->模型加载-->模型执行-->执行其它任务-->**aclFinalize**接口
- **Dump数据不落盘到文件，直接通过回调函数获取**
aclmdlInitDump接口、**acldumpRegCallback**接口（通过该接口注册的回调函数需由用户自行实现，回调函数实现逻辑中包括获取Dump数据及数据长度）、**acldumpUnregCallback**接口、**aclmdlFinalizeDump**接口配合使用，用于通过回调函数获取Dump数据。场景举例如下：
 - 执行一个模型，通过回调获取Dump数据：
aclInit接口-->**acldumpRegCallback**接口-->**aclmdlInitDump**接口-->模型加载-->模型执行-->**aclmdlFinalizeDump**接口-->**acldumpUnregCallback**接口-->模型卸载-->**aclFinalize**接口
 - 执行两个不同的模型，通过回调获取Dump数据，该场景下，只要不调用**acldumpUnregCallback**接口取消注册回调函数，则可通过回调函数获取两个模型的dump数据：
aclInit接口-->**acldumpRegCallback**接口-->**aclmdlInitDump**接口-->模型1加载-->模型1执行-->-->模型2加载-->模型2执行-->**aclmdlFinalizeDump**接口-->模型卸载-->**acldumpUnregCallback**接口-->**aclFinalize**接口

约束说明

- 对于模型Dump配置、单算子Dump配置、溢出算子Dump配置，如果已经通过**aclInit**接口配置了dump信息，则调用**aclmdlInitDump**接口时会返回失败。
- 必须在调用**aclInit**接口之后、模型加载接口之前调用**aclmdlInitDump**接口。

函数原型

aclError **aclmdlInitDump()**

参数说明

无

返回值说明

返回0表示成功，返回其他值表示失败。

相关接口

AscendCL还提供了**aclInit**接口，在AscendCL初始化阶段，通过*.json文件传入Dump配置信息，运行应用后获取Dump数据的方式。该种方式，一个进程内，只能调用一次**aclInit**接口，如果要修改Dump配置信息，需修改*.json文件中的配置。

1.7.2.18 aclmdlSetDump

函数功能

设置Dump参数。

aclmdlInitDump接口、**aclmdlSetDump**接口、**aclmdlFinalizeDump**接口配合使用，用于将Dump数据记录到文件中。一个进程内，可以根据需求多次调用这些接口，基于不同的Dump配置信息，获取Dump数据。场景举例如下：

- 执行两个不同的模型，需要设置不同的Dump配置信息，接口调用顺序：**aclInit**接口-->**aclmdlInitDump**接口-->**aclmdlSetDump**接口-->模型1加载-->模型1执行-->**aclmdlFinalizeDump**接口-->模型1卸载-->**aclmdlInitDump**接口-->**aclmdlSetDump**接口-->模型2加载-->模型2执行-->**aclmdlFinalizeDump**接口-->模型2卸载-->执行其它任务-->**aclFinalize**接口
- 同一个模型执行两次，第一次需要Dump，第二次无需Dump，接口调用顺序：**aclInit**接口-->**aclmdlInitDump**接口-->**aclmdlSetDump**接口-->模型加载-->模型执行-->**aclmdlFinalizeDump**接口-->模型卸载-->模型加载-->模型执行-->执行其它任务-->**aclFinalize**接口

约束说明

- 只有在调用本接口之后加载模型，配置的Dump信息有效。在调用本接口之前已经加载的模型不受影响，除非用户在调用本接口后重新加载该模型。
例如以下接口调用顺序中，加载的模型1不受影响，配置的Dump信息仅对加载的模型2有效：
aclmdlInitDump接口-->模型1加载-->**aclmdlSetDump**接口-->模型2加载-->**aclmdlFinalizeDump**接口
- 多次调用本接口对同一个模型配置了Dump信息，系统内处理时会采用覆盖策略。
例如以下接口调用顺序中，第二次调用本接口配置的Dump信息会覆盖第一次配置的Dump信息：
aclmdlInitDump接口-->**aclmdlSetDump**接口-->**aclmdlSetDump**接口-->模型1加载-->**aclmdlFinalizeDump**接口

函数原型

aclError **aclmdlSetDump**(const char *dumpCfgPath)

参数说明

参数名	输入/输出	说明
dumpCfgPath	输入	配置文件路径的指针，包含文件名。配置文件格式为json格式。 当前支持以下Dump信息配置： ● 模型Dump配置（用于导出模型中每一层算子输入和输出数据）、单算子Dump配置（用于导出单个算子的输入和输出数据），导出的数据用于与指定模型或算子进行比对，定位精度问题，配置示例、说明及约束请参见配置文件示例（模型Dump配置、单算子Dump配置）。默认不启用该dump配置。 ● 异常算子Dump配置（用于导出异常算子的输入输出数据、workspace信息、Tiling信息），导出的数据用于分析AI Core Error问题，配置示例请参见配置文件示例（异常算子Dump配置）。默认不启用该dump配置。 ● 溢出算子Dump配置（用于导出模型中溢出算子的输入和输出数据），导出的数据用于分析溢出原因，定位模型精度的问题，配置示例、说明及约束请参见配置文件示例（溢出算子Dump配置）。默认不启用该dump配置。 ● 算子Dump Watch模式配置（用于开启指定算子输出数据的观察模式），在定位部分算子精度问题且已排除算子本身的计算问题后，若怀疑被其它算子踩踏内存导致精度问题，可开启Dump Watch模式，配置示例及约束请参见配置文件示例（算子Dump Watch模式配置）。默认不开启Dump Watch模式。

配置文件示例（模型 Dump 配置、单算子 Dump 配置）

模型Dump配置示例如下：

```
{
  "dump":{
    "dump_list":[
      { "model_name":"ResNet-101"
      },
      {
        "model_name":"ResNet-50",
        "layer":[
          "conv1conv1_relu",
          "res2a_branch2ares2a_branch2a_relu",
          "res2a_branch1",
          "pool1"
        ]
      }
    ],
    "dump_path":"$HOME/output",
    "dump_mode":"output",
    "dump_op_switch":"off",
    "dump_data":"tensor"
  }
}
```

```
}  
}
```

单算子Dump配置示例如下：

```
{  
  "dump":{  
    "dump_path":"output",  
    "dump_list":[],  
    "dump_op_switch":"on",  
    "dump_data":"tensor"  
  }  
}
```

表 1-5 acl.json 文件格式说明

配置项	参数说明
dump_list	（必选）待dump数据的整网模型列表。 创建模型dump配置信息，当存在多个模型需要dump时，需要每个模型之间用英文逗号隔开。 在单算子执行场景下，dump_list为空。
model_name	模型名称，各个模型的model_name值须唯一。 - 模型加载方式为文件加载时，填入模型文件的名称，不需要带后缀名；也可以配置为ATC模型文件转换后的json文件里的最外层"name"字段对应值。 - 模型加载方式为内存加载时，配置为ATC模型文件转换后的json文件里的最外层"name"字段对应值。
layer	IO性能相对较差时，可能会出现由于数据量过大导致执行超时，所以不建议全量dump，请指定算子进行dump。通过该字段可以指定需要dump的算子名，支持指定为ATC模型转换后的算子名，也支持指定为转换前的原始算子名，配置时需注意： - 需按格式配置，每行配置模型中的一个算子名，且每个算子之间用英文逗号隔开。 - 用户可以无需设置model_name，此时会默认dump所有model下的相应算子。如果配置了model_name，则dump对应model下的相应算子。 - 若指定的算子其输入涉及data算子，会同时将data算子信息dump出来；若需dump data算子，需要一并填写data节点算子的后继节点，才能dump出data节点算子数据。 - 当需要dump模型中所有算子时，不需要包含layer字段。

配置项	参数说明
dump_path	(必选) dump数据文件存储到运行环境的目录，该目录需要提前创建且确保安装时配置的运行用户具有读写权限。 支持配置绝对路径或相对路径： - 绝对路径配置以 “/” 开头，例如：\$HOME/output。 - 相对路径配置直接以目录名开始，例如：output。
dump_mode	dump数据模式。 - input: dump算子的输入数据。 - output: dump算子的输出数据，默认取值output。 - all: dump算子的输入、输出数据。 注意，配置为all时，由于部分算子在执行过程中会修改输入数据，例如集合通信类算子HcomAllGather、HcomAllReduce等，因此系统在进行dump时，会在算子执行前dump算子输入，在算子执行后dump算子输出，这样，针对同一个算子，算子输入、输出的dump数据是分开落盘，会出现多个dump文件，在解析dump文件后，用户可通过文件内容判断是输入还是输出。
dump_level	dump数据级别，取值： - op: 按算子级别dump数据。 - kernel: 按kernel级别dump数据。 - all: 默认值，op和kernel级别的数据都dump。 默认配置下，dump数据文件会比较多，例如有一些aclnn开头的dump文件，若用户对dump性能有要求或内存资源有限时，则可以将该参数设置为op级别，以便提升dump性能、精简dump数据文件数量。 说明 算子是一个运算逻辑的表示（如加减乘除运算），kernel是运算逻辑真正进行计算处理的实现，需要分配具体的计算设备完成计算。
dump_op_switch	单算子模型dump数据开关。 - on: 开启单算子模型dump。 - off: 关闭单算子模型dump，默认取值off。

配置项	参数说明
dump_step	指定采集哪些迭代的Dump数据。推理场景无需配置。不配置该参数，默认所有迭代都会产生dump数据，数据量比较大，建议按需指定迭代。 多个迭代用“ ”分割，例如：0 5 10；也可以用“-”指定迭代范围，例如：0 3-5 10。 配置示例： <pre>{ "dump":{ "dump_list":[.....], "dump_path": "\$HOME/output", "dump_mode": "output", "dump_op_switch": "off" "dump_step": "0 3-5 10" } }</pre> 说明 训练场景下，若通过acl.json中的dump_step参数指定采集哪些迭代的Dump数据，又同时在GEInitialize接口中配置了ge.exec.dumpStep参数（该参数也用于指定采集哪些迭代的Dump数据），则以最后配置的参数为准。GEInitialize接口的详细介绍请参见《Ascend Graph开发指南》。
dump_data	算子dump内容类型，取值： • tensor: dump算子数据，默认为tensor。 • stats: dump算子统计数据，结果文件为csv格式，文件中包含算子名称、输入/输出的数据类型、最大值、最小值等。 通常dump数据量太大并且耗时长，可以先dump算子统计数据，根据统计数据识别可能异常的算子，然后再dump算子数据。 模型Dump场景下，会根据dump_mode处的配置统计算子输入或算子输出或算子输入&输出的信息。

配置文件示例（异常算子 Dump 配置）

将dump_scene参数设置为lite_exception，启用异常算子Dump功能，同时可配合使用ASCEND_WORK_PATH环境变量配置落盘路径，否则落盘在应用程序的当前执行目录下。**注意**，异常算子Dump配置，不能与模型Dump配置或单算子Dump配置同时开启，否则模型Dump或单算子Dump不生效。

配置文件中的示例内容如下：

```
{
  "dump":{
    "dump_scene": "lite_exception"
  }
}
```

此处收集的dump文件无法通过文本工具直接查看其内容，若需查看dump文件内容，先将dump文件转换为numpy格式文件后，再通过Python查看numpy格式文件，详细转换步骤请参见《[精度调试工具使用指南](#)》中的“查看dump数据文件”章节。

配置文件示例（溢出算子 Dump 配置）

溢出算子Dump配置的相关约束说明如下：

- 将dump_debug配置为on表示开启溢出算子配置，不配置dump_debug或将dump_debug配置为off表示不开启溢出算子配置。
- 若开启溢出算子配置，则dump_path必须配置，表示导出dump文件的存储路径。

支持配置绝对路径或相对路径：

- 绝对路径配置以“/”开头，例如：/home。
- 相对路径配置直接以目录名开始，例如：output。

获取导出的数据文件后，如何解析请参见溢出算子数据采集及分析。

- 溢出算子Dump配置，不能与模型Dump配置或单算子Dump配置同时开启，否则会返回报错。
- 仅支持采集AI Core算子的溢出数据。

配置文件中的示例内容如下：

```
{
  "dump":{
    "dump_path":"output",
    "dump_debug":"on"
  }
}
```

配置文件示例（算子 Dump Watch 模式配置）

将dump_scene参数设置为watcher，开启算子Dump Watch模式，详细配置说明及约束如下：

- 若开启算子Dump Watch模式，则不支持同时开启溢出算子Dump（配置dump_debug参数）或开启单算子模型Dump（配置dump_op_switch参数），否则报错。
- 在dump_list中，通过layer参数配置可能踩踏其它算子内存的算子名称，通过watcher_nodes参数配置可能被其它算子踩踏输出内存导致精度有问题的算子名称。
 - 若不指定layer，则模型内所有支持Dump的算子在执行后，都会将watcher_nodes中配置的算子的输出Dump出来。
 - layer和watcher_node处配置的算子都必须是静态图、静态子图中的算子，否则不生效。
 - 若layer和watcher_node处配置的算子名称相同，或者layer处配置的是集合通信类算子（算子类型以Hcom开头，例如HcomAllReduce），则不导出dump文件。
 - 对于融合算子，watcher_node处配置的算子名称必须是融合后的算子名称，若配置融合前的算子名称，则不导出dump文件。
 - dump_list内暂不支持配置model_name。
- 开启算子Dump Watch模式，则dump_path必须配置，表示导出dump文件的存储路径。

支持配置绝对路径或相对路径：

- 绝对路径配置以“/”开头，例如：/home。
- 相对路径配置直接以目录名开始，例如：output。

- 通过dump_mode参数控制导出watcher_nodes中所配置算子的哪部分数据，当前仅支持配置为output。

配置文件中的示例内容如下，配置效果为：当执行完A算子和B算子时，会把C算子和D算子的输出Dump出来，导出C算子opType.A_To_C.*、D算子opType.A_To_D.*、C算子opType.B_To_C.*、D算子opType.B_To_D.*四个dump文件，用于排查A算子、B算子是否会踩踏C算子、D算子的输出内存。

```
{
  "dump":{
    "dump_list":[
      {
        "layer":["A", "B"],
        "watcher_nodes":["C", "D"]
      }
    ],
    "dump_path":"/home/",
    "dump_mode":"output",
    "dump_scene":"watcher"
  }
}
```

此处收集的dump文件无法通过文本工具直接查看其内容，若需查看dump文件内容，先将dump文件转换为numpy格式文件后，再通过Python查看numpy格式文件，详细转换步骤请参见《[精度调试工具使用指南](#)》中的“查看dump数据文件”章节。

返回值说明

返回0表示成功，返回其他值表示失败。

相关接口

AscendCL还提供了[aclInit](#)接口，在AscendCL初始化阶段，通过*.json文件传入Dump配置信息，运行应用后获取Dump数据的方式。该种方式，一个进程内，只能调用一次[aclInit](#)接口，如果要修改Dump配置信息，需修改*.json文件中的配置。

1.7.2.19 acldumpRegCallback

函数功能

Dump数据回调函数注册接口。

[aclmdlInitDump](#)接口、[acldumpRegCallback](#)接口（通过该接口注册的回调函数需由用户自行实现，回调函数实现逻辑中包括获取Dump数据及数据长度）、[acldumpUnregCallback](#)接口、[aclmdlFinalizeDump](#)接口配合使用，用于通过回调函数获取Dump数据。场景举例如下：

- 执行一个模型，通过回调获取Dump数据：
支持以下两种方式：
 - 在[aclInit](#)接口处不启用模型Dump配置、单算子Dump配置
[aclInit](#)接口-->[aclmdlInitDump](#)接口-->[acldumpRegCallback](#)接口-->模型加载-->模型执行-->[acldumpUnregCallback](#)接口-->[aclmdlFinalizeDump](#)接口-->模型卸载-->[aclFinalize](#)接口
 - 在[aclInit](#)接口处启用模型Dump配置、单算子Dump配置，在[aclInit](#)接口处启用Dump配置时需配置落盘路径，但如果调用了[acldumpRegCallback](#)接口，则落盘不生效，以回调函数获取的Dump数据为准

- aclInit接口-->acldumpRegCallback接口-->模型加载-->模型执行-->acldumpUnregCallback接口-->模型卸载-->aclFinalize接口
- 执行两个不同的模型，通过回调获取Dump数据，该场景下，只要不调用acldumpUnregCallback接口取消注册回调函数，则可通过回调函数获取两个模型的dump数据：
aclInit接口-->aclmdlInitDump接口-->acldumpRegCallback接口-->模型1加载-->模型1执行-->-->模型2加载-->模型2执行-->acldumpUnregCallback接口-->aclmdlFinalizeDump接口-->模型卸载-->aclFinalize接口

函数原型

aclError acldumpRegCallback(int32_t (* const messageCallback)(const acldumpChunk *, int32_t len), int32_t flag)

参数说明

参数名	输入/输出	说明
messageCallback	输入	回调函数指针，用于接收回调数据的回调。 • acldumpChunk结构体的定义如下，在实现messageCallback回调函数时可以获得acldumpChunk结构体中的dataBuf、bufLen等参数值，用于获取Dump数据及其数据长度：<pre>typedef struct acldumpChunk { char fileName[ACL_DUMP_MAX_FILE_PATH_LENGTH]; // 待落盘的Dump数据文件名，ACL_DUMP_MAX_FILE_PATH_LENGTH表示文件名最大长度，当前为4096 uint32_t bufLen; // dataBuf数据长度，单位Byte uint32_t isLastChunk; // 标识Dump数据是否为最后一个分片，0表示不是最后一个分片，1表示最后一个分片 int64_t offset; // Dump数据文件内容的偏移，其中-1表示文件追加内容 int32_t flag; // 预留Dump数据标识，当前数据无标识 uint8_t dataBuf[0]; // Dump数据的内存地址 } acldumpChunk;</pre> • len：表示acldumpChunk结构体的长度，单位Byte。
flag	输入	在调用回调接口后是否还落盘dump数据： • 0：不落盘，当前仅支持0

返回值说明

返回0表示成功，返回其他值表示失败。

1.7.2.20 acldumpUnregCallback

函数功能

Dump数据回调函数取消注册接口。

aclmdlInitDump接口、**acldumpRegCallback**接口（通过该接口注册的回调函数需由用户自行实现，回调函数实现逻辑中包括获取Dump数据及数据长度）、**acldumpUnregCallback**接口、**aclmdlFinalizeDump**接口配合使用，用于通过回调函数获取Dump数据。场景举例如下：

- 执行一个模型，通过回调获取Dump数据：
aclInit接口-->**acldumpRegCallback**接口-->**aclmdlInitDump**接口-->模型加载-->模型执行-->**aclmdlFinalizeDump**接口-->**acldumpUnregCallback**接口-->模型卸载-->**aclFinalize**接口
- 执行两个不同的模型，通过回调获取Dump数据，该场景下，只要不调用**acldumpUnregCallback**接口取消注册回调函数，则可通过回调函数获取两个模型的dump数据：
aclInit接口-->**acldumpRegCallback**接口-->**aclmdlInitDump**接口-->模型1加载-->模型1执行-->-->模型2加载-->模型2执行-->**aclmdlFinalizeDump**接口-->模型卸载-->**acldumpUnregCallback**接口-->**aclFinalize**接口

约束说明

acldumpUnregCallback需要和**acldumpRegCallback**配合使用，且必须在**acldumpRegCallback**调用后才有效。

函数原型

void acldumpUnregCallback()

参数说明

无

返回值说明

无

1.7.2.21 aclmdlFinalizeDump

函数功能

Dump去初始化。

本接口需与其它接口配合使用实现以下功能：

- **Dump数据落盘到文件**
aclmdlInitDump接口、**aclmdlSetDump**接口、**aclmdlFinalizeDump**接口配合使用，用于将Dump数据记录到文件中。一个进程内，可以根据需求多次调用这些接口，基于不同的Dump配置信息，获取Dump数据。场景举例如下：
 - 执行两个不同的模型，需要设置不同的Dump配置信息，接口调用顺序：
aclInit接口-->**aclmdlInitDump**接口-->**aclmdlSetDump**接口-->模型1加载-->模型1执行-->**aclmdlFinalizeDump**接口-->模型1卸载-->**aclmdlInitDump**接口-->**aclmdlSetDump**接口-->模型2加载-->模型2执行-->**aclmdlFinalizeDump**接口-->模型2卸载-->执行其它任务-->**aclFinalize**接口
 - 同一个模型执行两次，第一次需要Dump，第二次无需Dump，接口调用顺序：
aclInit接口-->**aclmdlInitDump**接口-->**aclmdlSetDump**接口-->模型加

载-->模型执行-->[aclmdlFinalizeDump](#)接口-->模型卸载-->模型加载-->模型执行-->执行其它任务-->[aclFinalize](#)接口

- **Dump数据不落盘到文件，直接通过回调函数获取**

[aclmdlInitDump](#)接口、[aclDumpRegCallback](#)接口（通过该接口注册的回调函数需由用户自行实现，回调函数实现逻辑中包括获取Dump数据及数据长度）、[aclDumpUnregCallback](#)接口、[aclmdlFinalizeDump](#)接口配合使用，用于通过回调函数获取Dump数据。场景举例如下：

- 执行一个模型，通过回调获取Dump数据：

[aclInit](#)接口-->[aclDumpRegCallback](#)接口-->[aclmdlInitDump](#)接口-->模型加载-->模型执行-->[aclmdlFinalizeDump](#)接口-->[aclDumpUnregCallback](#)接口-->模型卸载-->[aclFinalize](#)接口

- 执行两个不同的模型，通过回调获取Dump数据，该场景下，只要不调用[aclDumpUnregCallback](#)接口取消注册回调函数，则可通过回调函数获取两个模型的dump数据：

[aclInit](#)接口-->[aclDumpRegCallback](#)接口-->[aclmdlInitDump](#)接口-->模型1加载-->模型1执行-->-->模型2加载-->模型2执行-->[aclmdlFinalizeDump](#)接口-->模型卸载-->[aclDumpUnregCallback](#)接口-->[aclFinalize](#)接口

函数原型

[aclError](#) [aclmdlFinalizeDump](#)()

参数说明

无

返回值说明

返回0表示成功，返回其他值表示失败。

相关接口

AscendCL还提供了[aclInit](#)接口，在AscendCL初始化阶段，通过*.json文件传入Dump配置信息，运行应用后获取Dump数据的方式。该种方式，一个进程内，只能调用一次[aclInit](#)接口，如果要修改Dump配置信息，需修改*.json文件中的配置。

1.8 媒体数据处理 V1

1.8.1 总体说明

多版本接口差异

- 本手册中媒体数据处理V1版本与媒体数据处理V2版本的接口都是描述处理媒体数据的接口，用于实现视频编解码、图像编解码、图像处理等功能，但两套接口不能混用。
 - V2版本的功能比V1版本更多，如下：
 - JPEG: V2版本接口支持高级的参数配置，如huffman表配置。

- VENC: V2版本接口支持更加细化的码控参数配置和效果调优，如I/P帧QP、宏块码控等。
- VDEC: V2版本接口支持更细化的内存控制，如设置输入码流缓存。
- 视频数据获取功能（ISP系统控制&MIPI命令字&VI功能）：仅V2版本接口支持。
- VPSS视频处理：仅V2版本接口支持。
- 音频相关功能，包括录音、播音、音量调节：仅V2版本接口支持。
- 视频数据展示功能（VO功能&HDMI外设）：仅V2版本接口支持。
- 建议使用V2版本中的接口，保证后续版本接口功能以及业务的连续演进。
- V1版本中的接口是为了兼容旧版本，保证使用该部分接口的用户能继续使用，后续版本不再演进。

功能说明

本章节介绍媒体数据处理V1版本（DVPP，Digital Vision Pre-Processing）的功能，如表1-6所示。

表 1-6 功能说明

功能	功能
VPC（Vision Preprocessing Core）	负责图像处理功能，支持对图片做抠图、缩放、格式转换等操作，详细描述请参见1.8.5.2 约束说明。
JPEGD（JPEG Decoder）	负责完成图像解码功能，将.jpg、.jpeg、.JPG、.JPEG图片解码成YUV格式图片，详细描述请参见1.8.6.1 功能及约束说明。
JPEGE（JPEG Encoder）	负责完成图像编码功能，将YUV格式图片编码成.jpg图片，详细描述请参见1.8.7.1 功能及约束说明。
VDEC（Video Decoder）	负责视频解码，详细描述请参见1.8.9.1 功能及约束说明。
VENC（Video Encoder）	负责视频编码，详细描述请参见1.8.10.1 功能及约束说明。
PNGD（PNG Decoder）	负责PNG格式图片的解码，详细描述请参见1.8.8.1 功能及约束说明。

功能支持度说明

昇腾AI处理器对媒体数据处理V1版本各功能的支持度如表1-7所示。

各标识的含义如下：

- √：支持。
- x：不支持。

表 1-7 功能支持度说明

昇腾AI处理器	VPC	JPEG D	JPEG E	PNG D	VDEC	VENC
Atlas 200/300/500 推理产品	√	√	√	√	√	√
Atlas 200/500 A2推理产品	√	√	√	√	√	√
Atlas 训练系列产品	√	√	√	√	√	x
Atlas A2训练系列产品/Atlas 800I A2推理产品	√	√	√	√	√	x
Atlas 推理系列产品	√	√	√	√	√	√

整体约束说明

使用本章中介绍的接口，有以下注意点：

- 关于异步接口：

对于本章介绍的异步接口，调用接口成功仅表示任务下发成功，不表示任务执行成功，对于有依赖的接口，为确保能按序执行任务，建议用户在多个接口中指定同一个Stream，因为同一个Stream中的任务按接口调用顺序执行。

在调用异步口对图片进行解码、抠图、缩放等操作时，如果任务之间有依赖，一定要调用[aclrtSynchronizeStream](#)接口确保在同一个Stream中的任务按序执行。

从性能角度考虑，建议一个Stream上下发多个异步媒体数据处理任务后，执行一次[aclrtSynchronizeStream](#)接口。

调用异步接口后，不能马上释放资源，需调用同步等待接口（例如，[aclrtSynchronizeStream](#)）确保Device侧任务执行完成后才能释放。
- 关于内存申请/释放：
 - a. 实现媒体数据处理的VPC、JPEGD、JPEG E等功能前，若需要申请Device上的内存存放输入或输出数据，需调用[acldvppMalloc](#)申请内存、调用[acldvppFree](#)接口释放内存。如果多个功能串联使用的场景，需要复用同一段内存，则按最大内存要求来申请内存。
 - b. 调用[a](#)申请出来的内存可以满足媒体数据处理的要求，也可以在其它任务中使用，例如，从性能角度，为了减少拷贝，媒体数据处理的输出作为模型推理的输入，实现内存复用。
 - c. 但由于媒体数据处理访问的地址空间有限，为确保媒体数据处理时内存足够，除媒体数据处理功能外的其它功能（例如，模型加载），建议调用[内存管理](#)章节下的[aclrtMalloc](#)接口、或[aclrtMallocHost](#)接口、或[aclrtMallocCached](#)接口申请内存。

Atlas 推理系列产品上，媒体数据处理功能在每个进程内，可访问的地址空间最大为16GB。

Atlas A2训练系列产品/Atlas 800I A2推理产品上，媒体数据处理功能在每个进程内，可访问的地址空间最大为16GB。
- 关于通道的要求：

实现媒体数据处理的各功能前，必须调用接口创建对应功能的通道，创建通道的接口请参见[1.8.4 通道创建与释放](#)。通道的创建与销毁会涉及资源的申请与释放，反复创建与销毁通道会影响业务性能，因此建议根据实际场景管理通道，例如，

如果有持续VPC图片处理，则创建VPC的通道后，等到所有VPC功能调用完成后，再销毁该VPC通道。

通道数量多，会影响Device的CPU占用率和内存占用，通道数量建议参考各功能章节下的性能指标的路数。

在Atlas 推理系列产品上，各功能的通道数有一定的限制，VPC的通道数最多256，JPEGD与VDEC共用通道且通道数最多256，JPEGE与VENC共用通道且通道数最多256，PNGD的通道数最多128。创建通道的接口请参见[1.8.4 通道创建与释放](#)。

Atlas A2训练系列产品/Atlas 800I A2推理产品上，各功能的通道数有一定的限制，VPC的通道数最多256，JPEGD功能和VDEC功能共用通道且通道总数最多256（其中，JPEGD解码通道总数最多256，VDEC解码通道总数最多32），JPEGE的通道数最多256，PNGD的通道数最多128。创建通道的接口请参见[1.8.4 通道创建与释放](#)。

Atlas 200/500 A2推理产品上，各功能的通道数有一定的限制，VPC的通道数最多128，JPEGD的通道数最多128，VDEC的通道数最多128，JPEGE的通道数最多128，VENC的通道数最多128，PNGD的通道数最多128。创建通道的接口请参见[1.8.4 通道创建与释放](#)。

 说明

对于Atlas 推理系列产品，昇腾虚拟化实例（Ascend Virtual Instance）场景下，如果通道总数不为整数，则向下取整：

- VPC通道总数 = (被分配的VPC硬件单元数量 / VPC硬件单元总数) * 256
- VDEC通道和JPEGD通道总数 = (被分配的VDEC硬件单元+JPEGD硬件单元数量之和 / VDEC硬件单元+JPEGD硬件单元总数之和) * 256
- VENC通道和JPEGE通道总数 = (被分配的VENC硬件单元+JPEGE硬件单元数量之和 / VENC硬件单元+JPEGE硬件单元总数之和) * 256
- PNGD通道总数 = 切分规格 * 128

对于PNGD功能，若使用以下昇腾虚拟化实例（Ascend Virtual Instance）模板，则对通道数的限制不同：

- 使用vir04_4c_dvpp模板时，通道总数固定为128。
- 使用vir04_3c_ndvpp模板时，表示不使用DVPP功能，因此通道总数为0。

对于Atlas A2训练系列产品/Atlas 800I A2推理产品，昇腾虚拟化实例（Ascend Virtual Instance）场景下，如果通道总数不为整数，则向下取整：

- VPC通道总数最多256。
- VDEC通道 = (被分配的VDEC硬件单元 / VDEC硬件单元) * 32，如果通道总数不为整数，则向下取整。JPEGD的通道数不受算力影响，但JPEGD+VDEC的总通道数最大256。
- JPEGE通道总数最多256。
- PNGD通道总数 = 切分规格 * 128

对于PNGD功能，若使用以下昇腾虚拟化实例（Ascend Virtual Instance）模板，则对通道数的限制不同：

- 使用vir12_3c_32g_m、vir10_3c_16g_m、vir10_3c_32g_m模板时，通道总数固定为128。
- 使用vir12_3c_32g_nm、vir10_3c_16g_nm、vir10_3c_32g_nm模板时，表示不使用DVPP功能，因此通道总数为0。

对于Atlas 200/500 A2推理产品，昇腾虚拟化实例（Ascend Virtual Instance）场景下，如果通道总数不为整数，则向下取整：

- VPC通道总数最多128。
- VDEC通道 = (被分配的VDEC硬件单元 / VDEC硬件单元) * 128，如果通道总数不为整数，则向下取整。JPEGD的通道数不随算力影响，但JPEGD+VDEC的总通道数最大128。
- VENC通道 = (被分配的VENC硬件单元 / VENC硬件单元) * 128，如果通道总数不为整数，则向下取整。JPEGE的通道数不随算力影响，但JPEGE+VENC的总通道数最大128。

昇腾虚拟化实例（Ascend Virtual Instance）的详细描述，以及各场景下的算力资源规格请参见《昇腾虚拟化实例（AVI）用户指南》。

1.8.2 基本概念

表 1-8 关键概念列表

概念	描述
宽stride (widthStride)	指一行图像跨距，表示输入/输出图片对齐后的宽，RGB格式和YUV格式的宽stride计算方式不一样。 各功能对宽stride的要求不同，请参见对应功能的约束说明： • VPC功能，请参见图片格式、宽高对齐、内存约束。 • JPEGD功能，请参见1.8.6.1 功能及约束说明。 • JPEGE功能，请参见1.8.7.1 功能及约束说明。 • PNGD功能，请参见1.8.8.1 功能及约束说明。 • VDEC功能，请参见1.8.9.1 功能及约束说明。 • VENC功能，请参见1.8.10.1 功能及约束说明。
高stride (heightStride)	指图像在内存中的行数，表示输入/输出图片对齐后的高。 各功能对高stride的要求不同，请参见对应功能的约束说明： • VPC功能，请参见图片格式、宽高对齐、内存约束。 • JPEGD功能，请参见1.8.6.1 功能及约束说明。 • JPEGE功能，请参见1.8.7.1 功能及约束说明。 • PNGD功能，请参见1.8.8.1 功能及约束说明。 • VDEC功能，请参见1.8.9.1 功能及约束说明。 • VENC功能，请参见1.8.10.1 功能及约束说明。
上/下/左/右偏移	通过配置上偏移、下偏移、左偏移、右偏移可以实现两个功能：指定抠图区域或贴图区域的位置，控制抠图或贴图区域的宽、高，右偏移-左偏移+1=宽，下偏移-上偏移+1=高。参见图1-1图1-6。
抠图区域	指用户指定的需抠出的图片区域。 抠图区域最小分辨率为10*6，最大分辨率为4096*4096。 抠图区域的约束，请参见抠图、贴图约束。
贴图区域	指在输出图片中用户指定的区域，贴图区域最小分辨率为10*6，最大分辨率为4096*4096。 贴图区域的约束，请参见抠图、贴图约束。
非8K	非8K表示10*6~4096*4096（包括4096）分辨率范围的输入图片。 非8K场景下，输入图片宽stride、高stride的取值范围如下：（宽stride/系数）在32~4096（包括4096）范围内，高stride在6~4096。 系数与输入图片的格式有关： • YUV400、YUV420SP、YUV422SP、YUV444SP、YUV440SP：系数为1。 • YUV422packed：系数为2。 • YUV444packed、RGB888：系数为3。 • ARGB8888：系数为4。

概念	描述
8K	8K表示宽或高在4096~8192（不包括4096）范围内的输入图片。 8K场景下，输入图片宽stride、高stride的取值范围如下： Atlas 200/300/500 推理产品上，宽stride在4096~8192范围内或高stride在4096~8192范围内（不包括4096）。 Atlas 训练系列产品上，宽stride在4096~8192范围内或高stride在4096~8192范围内（不包括4096）。 Atlas 推理系列产品上，宽stride在4096~16384范围内（不包括4096）。

1.8.3 内存申请与释放

1.8.3.1 acldvppMalloc

函数功能

该接口主要用于分配内存给Device侧媒体数据处理时使用，申请的大页内存满足数据处理的要求（例如，内存首地址128字节对齐），同步接口。

约束说明

- 调用该接口申请内存后，如果内存不使用，需及时调用[acldvppFree](#)接口释放内存。
- 频繁调用[acldvppMalloc](#)接口申请内存、调用[acldvppFree](#)接口释放内存，会损耗性能，建议用户提前做内存预先分配或二次管理，避免频繁申请/释放内存。
- 调用[acldvppMalloc](#)接口申请内存时，会对用户输入的size按向上对齐成32整数倍后，再多加32字节。
- 媒体数据处理的输出作为模型推理的输入时，若用户使用本接口申请大块内存并自行划分、管理内存时，每段内存需同时满足以下要求：
 - 内存大小向上对齐成32整数倍+32字节（ $m = \text{ALIGN\_UP}[\text{len}, 32] + 32$ 字节）；
 - 内存起始地址需满足128字节对齐（ $\text{ALIGN\_UP}[m, 128]$ ）。

说明

len表示某段内存的大小，ALIGN_UP[len,k]表示向上按k字节对齐： $((\text{len}-1)/k+1)*k$ 。

- 调用该接口申请大页内存失败，仅表示系统内的大页内存不够。关于系统内预留大页的说明请参见[1.5.5.1 总体说明](#)。

函数原型

aclError [acldvppMalloc](#)(void **devPtr, size_t size)

参数说明

参数名	输入/输出	说明
devPtr	输出	“Device上已分配内存的指针”的指针。
size	输入	申请内存的大小，单位Byte。

返回值说明

返回0表示成功，返回其他值表示失败。

参考资源

接口调用流程及示例，参见DVPP图像/视频处理（媒体数据处理V1）。

1.8.3.2 acldevppFree

函数功能

释放通过[acldevppMalloc](#)接口申请的内存，同步接口。

函数原型

aclError [acldevppFree](#)(void *devPtr)

参数说明

参数名	输入/输出	说明
devPtr	输入	待释放内存的指针。

返回值说明

返回0表示成功，返回其他值表示失败。

参考资源

接口调用流程及示例，参见DVPP图像/视频处理（媒体数据处理V1）。

1.8.4 通道创建与释放

1.8.4.1 VPC/JPEGD/JPEGE/PNGD 图片处理通道

1.8.4.1.1 acldvppCreateChannel

函数功能

创建图片数据处理的通道，同一个通道可以重复使用，销毁后不再可用，同步接口。

约束说明

- 通道为非线程安全，即不同线程要求创建不同的通道。
- 同一个通道不能同时在多个Stream中并发使用。
- acldvppCreateChannel接口内部调用1次[aclrtCreateStream](#)接口创建了1个Stream，该Stream用于创建通道的任务，创建通道的任务结束后，该Stream资源自动释放。
用户在创建通道时，[aclrtCreateStream](#)接口的约束需要遵循，例如Stream的数量等。

函数原型

```
aclError acldvppCreateChannel(acldvppChannelDesc *channelDesc)
```

参数说明

参数名	输入/输出	说明
channelDesc	输入&输出	通道描述信息的指针。 需提前调用 acldvppCreateChannelDesc 接口创建acldvppChannelDesc类型的数据。 同时，对于Atlas 推理系列产品，您还需要调用 acldvppSetChannelDescMode 接口指定通道描述信息中的通道模式，明确图片数据处理通道用于实现哪种功能，目前支持VPC、JPEGD、JPEG、PNGD功能。若不指定通道模式，则系统默认4种模式的通道都创建，可能会占用通道资源（通道数的限制，请参见1.8.1 总体说明）。

返回值说明

返回0表示成功，返回其他值表示失败。

参考资源

接口调用流程及示例，参见DVPP图像/视频处理（媒体数据处理V1）。

1.8.4.1.2 acldvppDestroyChannel

函数功能

销毁图片数据处理的通道，只能销毁通过[acldvppCreateChannel](#)接口创建的通道，同步接口。

约束说明

如果在调用[aclvppDestroyChannel](#)接口销毁通道前，已调用[aclvppDestroyChannelDesc](#)接口销毁了通道描述信息，那么在调用[aclvppDestroyChannel](#)接口销毁通道时会报错。

函数原型

```
aclError aclvppDestroyChannel(aclvppChannelDesc *channelDesc)
```

参数说明

参数名	输入/输出	说明
channelDesc	输入	通道描述信息的指针。 与调用 aclvppCreateChannel 接口创建通道时指定的channelDesc保持一致。

返回值说明

返回0表示成功，返回其他值表示失败。

参考资源

接口调用流程及示例，参见DVPP图像/视频处理（媒体数据处理V1）。

1.8.4.2 VDEC 视频解码通道

1.8.4.2.1 aclvdecCreateChannel

函数功能

创建视频解码处理的通道，同一个通道可以重复使用，销毁后不再可用，同步接口。

约束说明

- 通道为非线程安全，即不同线程要求创建不同的通道。
 - 通道数的限制，请参见[1.8.1 总体说明](#)。
 - [aclvdecCreateChannel](#)接口内部封装了[aclrtCreateStream](#)接口和[aclrtSubscribeReport](#)：
 - 调用2次[aclrtCreateStream](#)接口创建了2个Stream，一个Stream用于视频码流解码任务，另一个用于回调函数的处理。这些Stream资源只有在调用[aclvdecDestroyChannel](#)接口销毁通道后才能自动释放。
 - 调用[aclrtSubscribeReport](#)接口指定处理Stream上回调函数的线程，回调函数和线程是由用户调用[aclvdecSetChannelDesc系列接口](#)时指定的。
- 用户在实现VDEC功能时，无需再单独调用[aclrtCreateStream](#)接口、[aclrtSubscribeReport](#)接口，但[aclrtCreateStream](#)接口、

[aclrtSubscribeReport](#)接口的约束需要遵循，例如Stream的数量、处理Stream上回调函数的线程数量等。

函数原型

```
aclError aclvdecCreateChannel(aclvdecChannelDesc *channelDesc)
```

参数说明

参数名	输入/输出	说明
channelDesc	输入&输出	通道描述信息的指针。 需提前调用 aclvdecCreateChannelDesc 接口创建aclvdecChannelDesc类型的数据，再调用 aclvdecSetChannelDesc系列接口 设置通道描述信息的属性。

返回值说明

返回0表示成功，返回其他值表示失败。

参考资源

接口调用流程及示例，参见DVPP图像/视频处理（媒体数据处理V1）。

1.8.4.2.2 aclvdecDestroyChannel

函数功能

销毁视频解码处理的通道，只能销毁通过[aclvdecCreateChannel](#)接口创建的通道，同步接口。

约束说明

- 销毁通道接口会等待已发送帧解码完成且用户的回调函数处理完成后再销毁通道。
- [aclvdecDestroyChannel](#)接口内部封装了[aclrtUnSubscribeReport](#)接口取消线程注册（Stream上的回调函数不再由指定线程处理）、[aclrtDestroyStream](#)接口销毁Stream。用户在实现VDEC功能时，无需再单独调用[aclrtUnSubscribeReport](#)接口、[aclrtDestroyStream](#)接口。
- 如果在调用[aclvdecDestroyChannel](#)接口销毁通道前，已调用[aclvdecDestroyChannelDesc](#)接口销毁了通道描述信息，那么在调用[aclvdecDestroyChannel](#)接口销毁通道时会报错。

函数原型

```
aclError aclvdecDestroyChannel(aclvdecChannelDesc *channelDesc)
```

参数说明

参数名	输入/输出	说明
channelDesc	输入	通道描述信息的指针。 与调用 aclvdecCreateChannel 接口创建通道时指定的channelDesc保持一致。

返回值说明

返回0表示成功，返回其他值表示失败。

参考资源

接口调用流程及示例，参见DVPP图像/视频处理（媒体数据处理V1）。

1.8.4.3 VENC 视频编码通道

1.8.4.3.1 aclvencCreateChannel

Atlas 训练系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

创建视频编码处理的通道，同一个通道可以重复使用，销毁后不再可用，同步接口。

约束说明

- 通道为非线程安全，即不同线程要求创建不同的通道。
- 通道数的限制，请参见[1.8.1 总体说明](#)。
- [aclvencCreateChannel](#)接口内部调用2次[aclrtCreateStream](#)接口创建了2个Stream，一个Stream用于视频码流编码任务，另一个Stream用于回调函数的处理。这些Stream资源只有在调用[aclvencDestroyChannel](#)接口销毁通道后才能自动释放。
用户在实现VENC功能时，[aclrtCreateStream](#)接口的约束需要遵循，例如Stream的数量等。

函数原型

[aclError](#) [aclvencCreateChannel](#)([aclvencChannelDesc](#) *channelDesc)

参数说明

参数名	输入/输出	说明
channelDesc	输入&输出	通道描述信息的指针。 需提前调用 aclvencCreateChannelDesc 接口创建aclvencChannelDesc类型的数据，再调用 aclvencSetChannelDesc系列接口 设置通道描述信息的属性。

返回值说明

返回0表示成功，返回其他值表示失败。

参考资源

接口调用流程及示例，参见DVPP图像/视频处理（媒体数据处理V1）。

1.8.4.3.2 aclvencDestroyChannel

Atlas 训练系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

销毁视频编码处理的通道，只能销毁通过[aclvencCreateChannel](#)接口创建的通道，同步接口。

约束说明

- 销毁通道接口会等待用户的回调函数处理完成后再销毁。
- 如果在调用aclvencDestroyChannel接口销毁通道前，已调用[aclvencDestroyChannelDesc](#)接口销毁了通道描述信息，那么在调用aclvencDestroyChannel接口销毁通道时会报错。

函数原型

aclError aclvencDestroyChannel([aclvencChannelDesc](#) *channelDesc)

参数说明

参数名	输入/输出	说明
channelDesc	输入	通道描述信息的指针 与调用 aclvencCreateChannel 接口创建通道时指定的channelDesc保持一致。

返回值说明

返回0表示成功，返回其他值表示失败。

参考资源

接口调用流程及示例，参见DVPP图像/视频处理（媒体数据处理V1）。

1.8.5 VPC 功能

1.8.5.1 功能说明

功能说明

VPC（Vision Preprocessing Core）各功能的介绍、各版本的功能支持度如下表所示。本章的接口都是异步接口，调用接口成功仅表示任务下发成功，不表示任务执行成功。调用该接口后，需调用同步等待接口（例如，[aclrtSynchronizeStream](#)）确保任务已执行完成。

功能	描述	Atlas 200/300/500 推理产品	Atlas 训练系列产品	Atlas 推理系列产品	Atlas 200/500 A2推理产品	Atlas A2 训练系列产品/ Atlas 800I A2 推理产品
抠图	从输入图片中抠出需要用的图片区域，支持一图多框和多图多框。	√	√	√	√	√
缩放	- 针对不同分辨率的图像，支持8K缩放、非8K缩放。 - 支持单图裁剪缩放（支持非压缩格式）、一图多框裁剪缩放（支持非压缩格式）。 - 其它缩放方式，如：原图缩放、等比例缩放（缩放前后图片的宽高比例相同）。	√	√	√	√	√

功能	描述	Atlas 200/300/500 推理产品	Atlas 训练系列产品	Atlas 推理系列产品	Atlas 200/500 A2推理产品	Atlas A2 训练系列产品/ Atlas 800I A2 推理产品
叠加	从输入图片中抠出来的图，对抠出的图进行缩放后，放在用户输出图片的指定区域，输出图片可以是空白图片（由用户申请的空输出内存产生的），也可以是已有图片（由用户申请输出内存后将已有图片读入输出内存），只有当输出图片是已有图片时，才表示叠加。	√	√	√	√	√
拼接	从输入图片中抠多张图片，对抠出的图进行缩放后，放到输出图片的指定区域。	√	√	√	√	√
图像金字塔	对原图进行多次高斯滤波之后向下采样产生不同尺寸的图像。	x	x	√	x	x
直方图统计	统计图像每个通道（RGB/YUV）的像素值分布。	x	x	√	√	√
色彩重映射	根据配置信息将图片从原图映射为另一张图。	x	x	√	√	√
边界填充	对图像进行边界填充。	x	x	√	√	√
格式转换	支持RGB格式、YUV格式之间的格式转换。	√	√	√	√	√

功能	描述	Atlas 200/300/500 推理产品	Atlas 训练系列产品	Atlas 推理系列产品	Atlas 200/500 A2推理产品	Atlas A2 训练系列产品/ Atlas 800I A2 推理产品
图像灰度化	将彩色图像转化为灰度图像。需注意，输入为灰度图像、输出只能为灰度图像。	√ 从 YUV420SP 格式的输入图像数据中只取 Y 分量的数据	√ 从 YUV420SP 格式的输入图像数据中只取 Y 分量的数据	√ 实现图像灰度化的操作是输出 YUV400 格式的输入图片	√ 实现图像灰度化的操作是输出 YUV400 格式的输入图片	√ 实现图像灰度化的操作是输出 YUV400 格式的输入图片

功能示意图(Atlas 200/300/500 推理产品)(Atlas 训练系列产品)(Atlas 推理系列产品)

图 1-1 VPC 功能示意图（抠图+缩放+叠加）

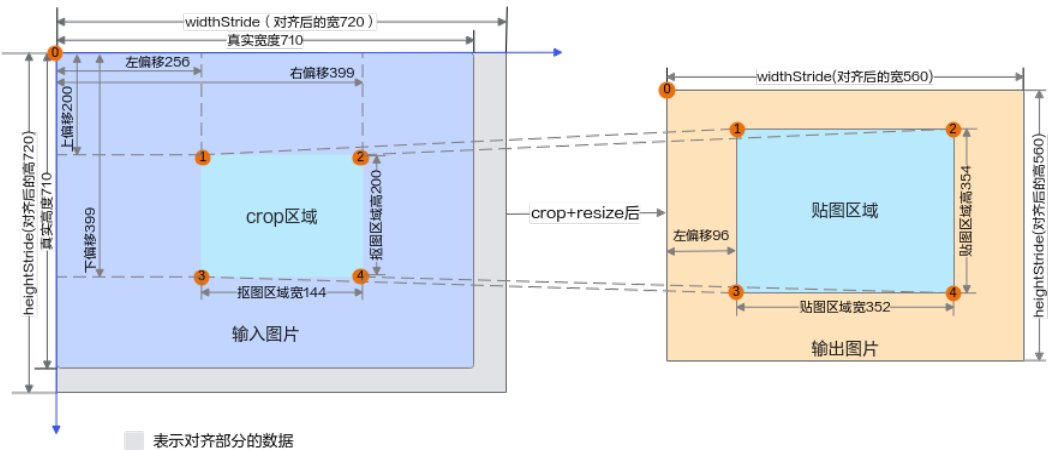


图 1-2 VPC 功能示意图（拼接）

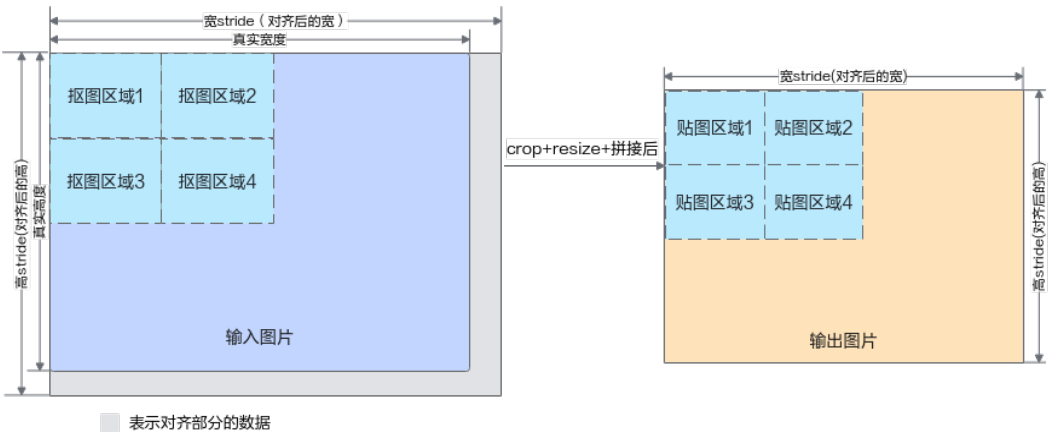


图 1-3 等比例缩放（贴图区域在输出图片的中心位置），即缩放前后图片的宽高比例相同

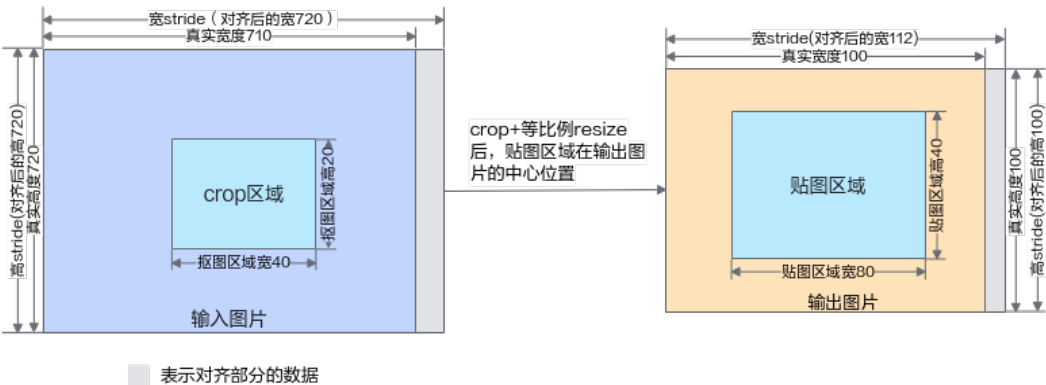


图 1-4 等比例缩放（贴图区域在输出图片的上下居中位置）

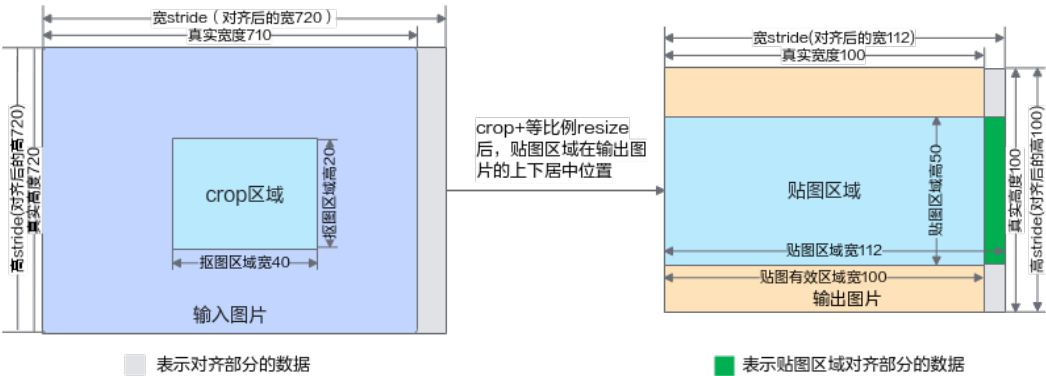
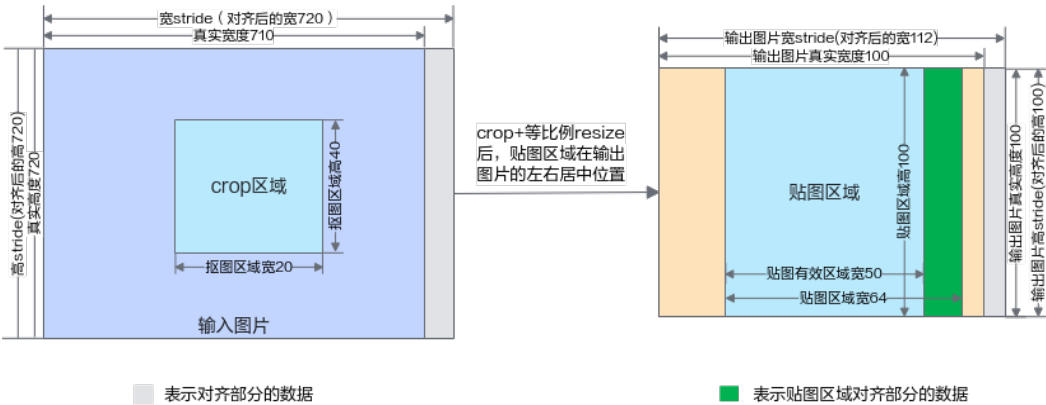


图 1-5 等比例缩放（贴图区域在输出图片的左右居中位置）



功能示意图(Atlas A2 训练系列产品/Atlas 800I A2 推理产品)(Atlas 200/500 A2 推理产品)

图 1-6 VPC 功能示意图（抠图+缩放+叠加）

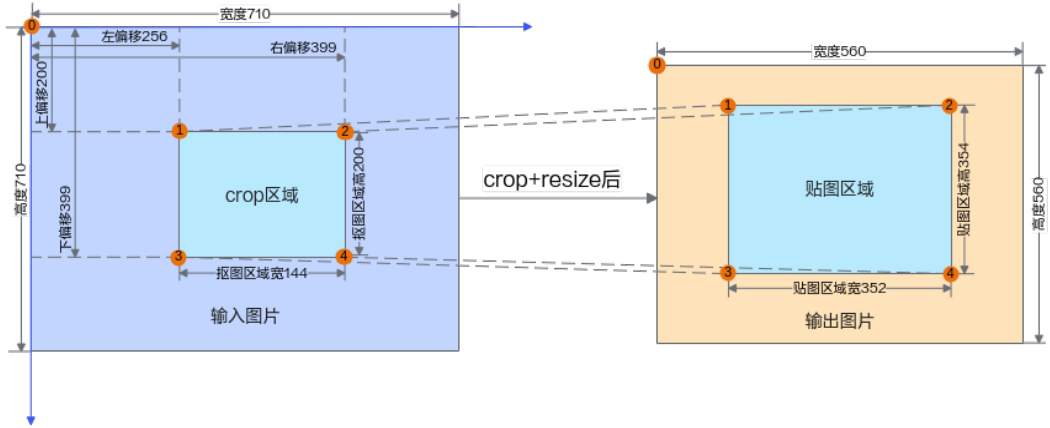


图 1-7 VPC 功能示意图（拼接）

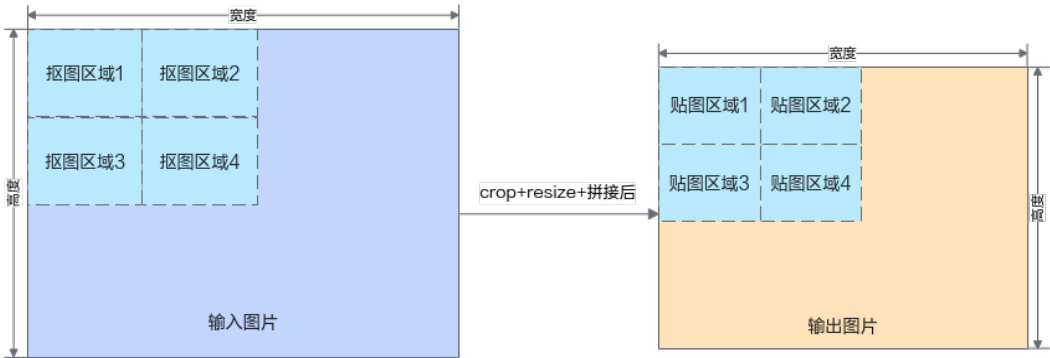
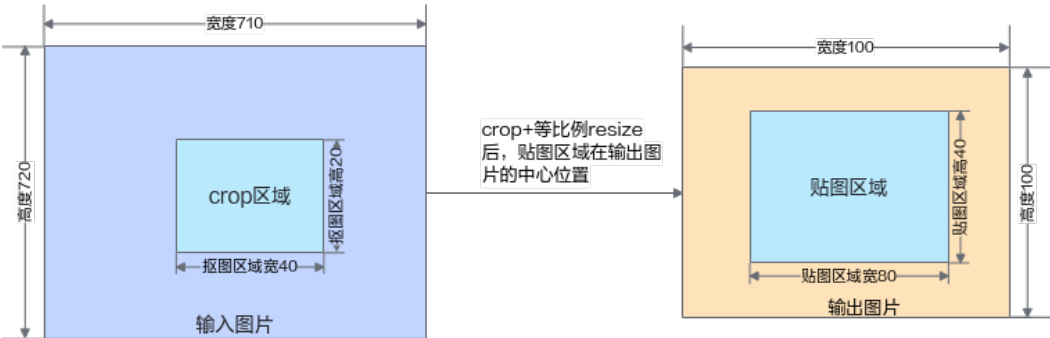


图 1-8 等比例缩放（贴图区域在输出图片的中心位置），即缩放前后图片的宽高比例相同



参考资源

RGB、YUV格式图像的各分量排布示意图。示例：SP图像以YUV420SP为例，Packed和RGB图像以ARGB图像为例。

格式	分辨率	宽stride	高stride	buffer大小					
yuv420sp	4*4	4	4	24					
内存排列									
y11	y12	y13	y14						
y21	y22	y23	y24						
y31	y32	y33	y34						
y41	y42	y43	y44						
u11	v11	u13	v13						
u31	v31	u33	v33						
格式	分辨率	宽stride	高stride	buffer大小					
yuv420sp	4*4	6	6	54					
内存排列 x为无效数据									
y11	y12	y13	y14	x	x				
y21	y22	y23	y24	x	x				
y31	y32	y33	y34	x	x				
y41	y42	y43	y44	x	x				
x	x	x	x	x	x				
x	x	x	x	x	x				
u11	v11	u13	v13	x	x				
u31	v31	u33	v33	x	x				
x	x	x	x	x	x				
格式	分辨率	宽stride	高stride	buffer大小					
argb	2*2	10	4	40					
内存排列 x为无效数据									
a11	r11	g11	b11	a12	r12	g12	b12	x	x
a21	r21	g21	b21	a22	r22	g22	b22	x	x
x	x	x	x	x	x	x	x	x	x
x	x	x	x	x	x	x	x	x	x

1.8.5.2 约束说明

本节中8K、非8K的概念如下：

- **非8K**：宽*高在10*6~4096*4096范围内，包括4096。
- **8K**：宽或高在4096~8192，不包括4096。

图片分辨率约束

- 输入图片分辨率
10*6~8192*8192
在调用接口实现VPC功能时，各接口对分辨率的要求可能不同，请参见**1.8.5 VPC功能**下各接口的说明。其中，当输入图片格式为YUV440SP、YUV440P时，输入图片的宽最大值为4096。

- 输出图片分辨率

版本	分辨率范围
Atlas 200/300/500 推理产品 Atlas 训练系列产品 Atlas 推理系列产品	10*6~4096*4096

版本	分辨率范围
Atlas A2训练系列产品/Atlas 800I A2推理产品 Atlas 200/500 A2推理产品	10*6~4096*8192

图片格式、宽高对齐、内存约束

VPC在处理图片时，需调用[acldvppMalloc](#)接口申请Device上的输入、输出内存，调用[acldvppFree](#)接口释放输入、输出内存，这部分内存的生命周期由用户自行管理。

图片格式、宽高对齐约束请参见[表1-9](#)、[表1-10](#)。

- Atlas 200/300/500 推理产品和Atlas 训练系列产品，在8K场景下，输入图片格式仅支持YUV420SP NV12 8bit和YUV420SP NV21 8bit。
- Atlas 推理系列产品，在8K场景下，输入图片格式支持如下格式：
- YUV400 8bit
 - YUV420SP NV12 8bit、YUV420SP NV21 8bit
 - YUV422SP 8bit、YVU422SP 8bit
 - YUV444SP 8bit、YVU444SP 8bit
 - YUV422P YUYV 8bit、YUV422P UYVY 8bit、YUV422P YVYU 8bit、YUV422P VYUY 8bit
- Atlas A2训练系列产品/Atlas 800I A2推理产品，在8K场景下，输入图片格式支持如下格式：
- YUV400 8bit
 - YUV420SP NV12 8bit、YUV420SP NV21 8bit
 - YUV422SP 8bit、YVU422SP 8bit
 - YUV444SP 8bit、YVU444SP 8bit
 - YUV422P YUYV 8bit、YUV422P UYVY 8bit、YUV422P YVYU 8bit、YUV422P VYUY 8bit
- Atlas 200/500 A2推理产品，在8K场景下，输入图片格式支持如下格式：
- YUV400 8bit
 - YUV420SP NV12 8bit、YUV420SP NV21 8bit
 - YUV422SP 8bit、YVU422SP 8bit
 - YUV444SP 8bit、YVU444SP 8bit
 - YUV422P YUYV 8bit、YUV422P UYVY 8bit、YUV422P YVYU 8bit、YUV422P VYUY 8bit

 说明

在调用接口实现VPC功能时：

- 图片格式的定义请参见[aclDvppPixelFormat](#)，宽stride、高stride、8K等概念请参见[1.8.2 基本概念](#)。
- 不同版本中，宽stride、高stride的最小值、最大值不同，如下：
 - Atlas 200/300/500 推理产品上，宽stride最小32、最大16384（ $16384=4096*4$ ，宽是4096的argb格式的图像，1个像素占用4个字节，一行像素就占用 $4096*4$ ，即宽stride）；高stride最小6、最大16384。
 - Atlas 训练系列产品上，宽stride最小32、最大16384（ $16384=4096*4$ ，宽是4096的argb格式的图像，1个像素占用4个字节，一行像素就占用 $4096*4$ ，即宽stride）；高stride最小6、最大16384。
 - Atlas 推理系列产品上，宽stride最小32、最大16384（ $16384=4096*4$ ，宽是4096的argb格式的图像，1个像素占用4个字节，一行像素就占用 $4096*4$ ，即宽stride）；高stride最小6、最大16384。
 - Atlas 200/500 A2推理产品上，宽stride最小10、最大16384（ $16384=4096*4$ ，宽是4096的argb格式的图像，1个像素占用4个字节，一行像素就占用 $4096*4$ ，即宽stride）；高stride最小6、最大16384。
 - Atlas A2训练系列产品/Atlas 800I A2推理产品上，宽stride最小10、最大16384（ $16384=4096*4$ ，宽是4096的argb格式的图像，1个像素占用4个字节，一行像素就占用 $4096*4$ ，即宽stride）；高stride最小6、最大16384。

表 1-9 输入图片格式、宽高对齐、内存大小约束

版本	输入图片格式	输入图片宽、高对齐要求	输入图片宽stride、高stride、内存大小要求 Atlas 200/300/500 推理产品 Atlas 训练系列产品	输入图片宽stride、高stride、内存大小要求 Atlas 推理系列产品	输入图片宽stride、高stride、内存大小要求 Atlas 200/500 A2推理产品 Atlas A2训练系列产品/Atlas 800I A2推理产品
通用格式，各版本都支持	YUV400 8bit	无对齐要求	宽stride为宽16对齐后的值。 高stride为高2对齐后的值。 内存大小（单位Byte）= 宽stride * 高stride * 3/2	宽stride为宽16对齐后的值。 高stride无对齐要求，与高相同即可。 内存大小（单位Byte）= 宽stride * 高stride	宽stride无对齐要求，与宽相同即可。 高stride无对齐要求，与高相同即可。 内存大小（单位Byte）= 宽stride * 高stride

版本	输入图片格式	输入图片宽、高对齐要求	输入图片宽stride、高stride、内存大小要求 Atlas 200/300/500推理产品 Atlas 训练系列产品	输入图片宽stride、高stride、内存大小要求 Atlas 推理系列产品	输入图片宽stride、高stride、内存大小要求 Atlas 200/500 A2推理产品 Atlas A2训练系列产品/Atlas 800I A2推理产品
	YUV420SP NV12 8bit	宽2对齐 高2对齐	宽stride为宽16对齐后的值。 高stride为高2对齐后的值。 内存大小 (单位Byte) = 宽stride * 高stride * 3/2	宽stride为宽16对齐后的值。 高stride为高2对齐后的值。 内存大小 (单位Byte) = 宽stride * 高stride * 3/2	宽stride无对齐要求，与宽相同即可。 高stride为高2对齐后的值。 内存大小 (单位Byte) = 宽stride * 高stride * 3/2
	YUV420SP NV21 8bit				
	YUV422SP 8bit	宽2对齐 高无对齐要求	宽stride为宽16对齐后的值。 高stride为高2对齐后的值。 内存大小 (单位Byte) = 宽stride * 高stride * 2	宽stride为宽16对齐后的值。 高stride无对齐要求，与高相同即可。 内存大小 (单位Byte) = 宽stride * 高stride * 2	宽stride无对齐要求，与宽相同即可。 高stride无对齐要求，与高相同即可。 内存大小 (单位Byte) = 宽stride * 高stride * 2
	YVU422SP 8bit				
	YUV444SP 8bit	无对齐要求	宽stride为宽16对齐后的值。 高stride为高2对齐后的值。 内存大小 (单位Byte) = 宽stride * 高stride * 3	宽stride为宽16对齐后的值。 高stride无对齐要求，与高相同即可。 内存大小 (单位Byte) = 宽stride * 高stride * 3	宽stride无对齐要求，与宽相同即可。 高stride无对齐要求，与高相同即可。 内存大小 (单位Byte) = 宽stride * 高stride * 3
	YVU444SP 8bit				
	YUV422Packed YUYV 8bit	宽2对齐 高无对齐要求	宽stride为宽16对齐后、再乘以2的值。 高stride为高2对齐后的值。 内存大小 (单位Byte) = 宽stride * 高stride	宽stride为宽16对齐后、再乘以2的值。 高stride无对齐要求，与高相同即可。 内存大小 (单位Byte) = 宽stride * 高stride	宽stride为宽乘以2的值。 高stride无对齐要求，与高相同即可。 内存大小 (单位Byte) = 宽stride * 高stride

版本	输入图片格式	输入图片宽、高对齐要求	输入图片宽stride、高stride、内存大小要求 Atlas 200/300/500推理产品 Atlas 训练系列产品	输入图片宽stride、高stride、内存大小要求 Atlas 推理系列产品	输入图片宽stride、高stride、内存大小要求 Atlas 200/500 A2推理产品 Atlas A2训练系列产品/Atlas 800I A2推理产品
	YUV422PackedUYVY8bit				
	YUV422PackedYVYU8bit				
	YUV422PackedVYUY8bit				
	YUV444Packed8bit	无对齐要求	宽stride为宽16对齐后、再乘以3的值。 高stride为高2对齐后的值。 内存大小（单位Byte）= 宽stride * 高stride	宽stride为宽16对齐后、再乘以3的值。 高stride无对齐要求，与高相同即可。 内存大小（单位Byte）= 宽stride * 高stride	宽stride为宽乘以3的值。 高stride无对齐要求，与高相同即可。 内存大小（单位Byte）= 宽stride * 高stride
	RGB888				
	BGR888				
	ARGB888	无对齐要求	宽stride为宽16对齐后、再乘以4的值。 高stride为高2对齐后的值。 内存大小（单位Byte）= 宽stride * 高stride	宽stride为宽16对齐后、再乘以4的值。 高stride无对齐要求，与高相同即可。 内存大小（单位Byte）= 宽stride * 高stride	宽stride为宽乘以4的。 高stride无对齐要求，与高相同即可。 内存大小（单位Byte）= 宽stride * 高stride
	ABGR888				
	RGBA888				
	BGRA888				

版本	输入图片格式	输入图片宽、高对齐要求	输入图片宽stride、高stride、内存大小要求 Atlas 200/300/500推理产品 Atlas 训练系列产品	输入图片宽stride、高stride、内存大小要求 Atlas 推理系列产品	输入图片宽stride、高stride、内存大小要求 Atlas 200/500 A2推理产品 Atlas A2训练系列产品/Atlas 800I A2推理产品
该部分格式仅以下版本支持： Atlas 推理系列产品 Atlas 200/500 A2推理产品 Atlas A2训练系列产品/Atlas 800I A2推理产品	YUV440SP 8bit	宽无对齐要求 高2对齐	-	宽stride为宽16对齐后的值。 高stride为高2对齐后的值。 内存大小（单位Byte）= 宽stride * 高stride * 2	宽stride无对齐要求，与宽相同即可。 高stride为高2对齐后的值。 内存大小（单位Byte）= 宽stride * 高stride * 2
	YVU440SP 8bit				
该部分格式仅以下版本支持： Atlas 200/500 A2推理产品 Atlas A2训练系列产品/Atlas 800I A2推理产品	YVU420Planar	宽2对齐，高2对齐	-	-	宽stride无对齐要求，与宽相同即可。 高stride无对齐要求，与高相同即可。 内存大小（单位Byte）= 宽stride * 高stride * 3/2
	YUV420Planar				
	YVU422Planar	无对齐要求			宽stride无对齐要求，与宽相同即可。 高stride无对齐要求，与高相同即可。 内存大小（单位Byte）= 宽stride * 高stride * 2
	YUV422Planar				

版本	输入图片格式	输入图片宽、高对齐要求	输入图片宽stride、高stride、内存大小要求 Atlas 200/300/500推理产品 Atlas 训练系列产品	输入图片宽stride、高stride、内存大小要求 Atlas 推理系列产品	输入图片宽stride、高stride、内存大小要求 Atlas 200/500 A2推理产品 Atlas A2训练系列产品/Atlas 800I A2推理产品
	YVU444Planar	无对齐要求			宽stride无对齐要求，与宽相同即可。
	YUV444Planar				高stride无对齐要求，与高相同即可。
		内存大小（单位Byte）= 宽stride * 高stride * 3			
	YVU444Packed 8bit	无对齐要求			宽stride无对齐要求，宽stride为宽乘以3的值。
		高stride无对齐要求，与高相同即可。			
		内存大小（单位Byte）= 宽stride * 高stride			
	YUV440Planar	宽无对齐要求，但宽≤4096 高2对齐			宽stride无对齐要求，与宽相同即可。
YVU440Planar	高stride无对齐要求，与高相同即可。				
		内存大小（单位Byte）= 宽stride * 高stride * 2			
	RGB888Planar	无对齐要求			宽stride无对齐要求，与宽相同即可。
	BGR888Planar				高stride无对齐要求，与高相同即可。
					内存大小（单位Byte）= 宽stride * 高stride * 3

版本	输入图 片格式	输入图 片宽、 高对齐 要求	输入图片宽 stride、高 stride、内存 大小要求 Atlas 200/300/500 推理产品 Atlas 训练系 列产品	输入图片宽 stride、高 stride、内存大 小要求 Atlas 推理系列 产品	输入图片宽 stride、高 stride、内存大 小要求 Atlas 200/500 A2推理产品 Atlas A2训练系 列产品/Atlas 800I A2推理产 品
Atlas A2训练 系列产 品/Atlas 800I A2 推理产 品	YUV40 0 FP32	无对齐 要求	-	-	宽stride无对齐要 求，宽stride为宽 乘以3的值；高 stride无对齐要 求，与高相同即 可。 内存大小（单位 Byte）= 宽stride * 高stride
	RGB88 8 FP32				
	BGR88 8 FP32				
	RGB88 8Plana r FP32 BGR88 8Plana r FP32	无对齐 要求			宽stride无对齐要 求，与宽相同即 可；高stride无对 齐要求，与高相 同即可。 内存大小（单位 Byte）= 宽stride * 高stride * 3

表 1-10 输出图片格式、宽高对齐、内存大小约束

版本	输出图片格式	输出图片宽、高对齐要求	输出图片宽stride、高stride、内存大小要求 Atlas 200/300/500 推理产品 Atlas 训练系列产品	输出图片宽stride、高stride、内存大小要求 Atlas 推理系列产品	输出图片宽stride、高stride、内存大小要求 Atlas 200/500 A2推理产品 Atlas A2训练系列产品/Atlas 800I A2推理产品
通用格式，各版本都支持	YUV420 SP NV12 8bit	宽2对齐 高2对齐	宽stride为宽16对齐后的值。 高stride为高2对齐后的值。 内存大小（单位Byte）= 宽stride * 高stride * 3/2	宽stride为宽16对齐后的值。 高stride为高2对齐后的值。 内存大小（单位Byte）= 宽stride * 高stride * 3/2	宽stride无对齐要求，与宽相同即可。 高stride为高2对齐后的值。 内存大小（单位Byte）= 宽stride * 高stride * 3/2
	YUV420 SP NV21 8bit				
该部分格式仅以下版本支持： Atlas 推理系列产品 Atlas 200/500 A2推理产品 Atlas A2训练系列产品/Atlas 800I A2推理产品	YUV400 8bit	无对齐要求	-	宽stride为宽16对齐后的值。 高stride无对齐要求，与高相同即可。 内存大小（单位Byte）= 宽stride * 高stride	宽stride无对齐要求，与宽相同即可；高stride无对齐要求，与高相同即可。 内存大小（单位Byte）= 宽stride * 高stride
	YUV422 SP 8bit	宽2对齐		宽stride为宽16对齐后的值。 高stride无对齐要求，与高相同即可。 内存大小（单位Byte）= 宽stride * 高stride * 2	宽stride无对齐要求，与宽相同即可；高stride无对齐要求，与高相同即可。 内存大小（单位Byte）= 宽stride * 高stride * 2
	YVU422 SP 8bit	高无对齐要求			
	YUV444 Packed 8bit	无对齐要求		宽stride为宽16对齐后、再乘以3的值。 高stride无对齐要求，与高相同即可。 内存大小（单位Byte）= 宽stride * 高stride	宽stride为宽乘以3的值；高stride无对齐要求，与高相同即可。 内存大小（单位Byte）= 宽stride * 高stride
	RGB888				
	BGR888				

版本	输出图片格式	输出图片宽、高对齐要求	输出图片宽stride、高stride、内存大小要求 Atlas 200/300/500推理产品 Atlas 训练系列产品	输出图片宽stride、高stride、内存大小要求 Atlas 推理系列产品	输出图片宽stride、高stride、内存大小要求 Atlas 200/500 A2推理产品 Atlas A2训练系列产品/Atlas 800I A2推理产品
	ARGB8888	无对齐要求		宽stride为宽16对齐后、再乘以4的值。 高stride无对齐要求，与高相同即可。 内存大小（单位Byte）= 宽stride * 高stride	宽stride为宽乘以4的值；高stride无对齐要求，与高相同即可。 内存大小（单位Byte）= 宽stride * 高stride
	ABGR8888				
	RGBA8888				
	BGRA8888				
该部分格式仅以下版本支持： Atlas 200/500 A2推理产品 Atlas A2训练系列产品/ Atlas 800I A2推理产品	YVU444 Packed	无对齐要求	-	-	宽stride为宽乘以3的值；高stride无对齐要求，与高相同即可。 内存大小（单位Byte）= 宽stride * 高stride
	RGB888 Planar BGR888 Planar	无对齐要求	-	-	宽stride无对齐要求，与宽相同即可；高stride无对齐要求，与高相同即可。 内存大小（单位Byte）= 宽stride * 高stride * 3

版本	输出图片格式	输出图片宽、高对齐要求	输出图片宽stride、高stride、内存大小要求 Atlas 200/300/500推理产品 Atlas 训练系列产品	输出图片宽stride、高stride、内存大小要求 Atlas 推理系列产品	输出图片宽stride、高stride、内存大小要求 Atlas 200/500 A2推理产品 Atlas A2训练系列产品/Atlas 800I A2推理产品
该部分格式仅以下版本支持： Atlas A2训练系列产品/ Atlas 800I A2推理产品	YUV400 FP32 RGB888 FP32 BGR888 FP32	无对齐要求	-	-	宽stride无对齐要求，宽stride为宽乘以3的值；高stride无对齐要求，与高相同即可。 内存大小（单位Byte）= 宽stride * 高stride
	RGB888 Planar FP32 BGR888 Planar FP32	无对齐要求			宽stride无对齐要求，与宽相同即可；高stride无对齐要求，与高相同即可。 内存大小（单位Byte）= 宽stride * 高stride * 3

抠图、贴图约束

产品版本	抠图、贴图约束说明
Atlas 200/300/500 推理产品	• 抠图区域不超出输入图片区域。 • 贴图区域不超出输出图片区域，最大贴图个数256个。 • 贴图区域相对输出图片的左偏移16对齐。 - 贴图时可直接放置在输出图片的最左侧，即相对输出图片的左偏移为0，0也满足16对齐的要求。 - 输出图片的贴图宽度建议16对齐，如果不是16对齐，会多写一段无效数据使其16对齐。见图1-4、图1-5，贴图区域旁边的绿色框就表示无效数据。 - 在等比例缩放场景下，由于贴图左偏移16对齐的限制，如果直接调用aclvppVpcCropResizePasteAsync接口实现缩放功能，贴图的位置可能不在输出图片的正中心位置，此时可通过抠图、缩放、填充几个功能配合使用（调用aclvppVpcBatchCropResizeMakeBorderAsync接口），实现等比例缩放。 • 抠图、贴图区域的奇数、偶数限制： 仅输出图片格式为YUV420SP，贴图区域奇数、偶数限制为：左偏移和上偏移为偶数、右偏移和下偏移为奇数。 • 缩放场景下，贴图/抠图的宽或高缩放比例范围：[1/32, 16]。 • 推理场景下，VPC的输出作为模型推理的输入时，由于贴图区域的宽*高有16*2对齐的约束，若贴图有效区域的宽*高不满足16*2对齐时，则贴图区域中会存在一些补边的无效数据，为保证推理精度，贴图有效区域的宽*高不满足16*2对齐时，用户需借助AIPP抠出有效区域，去除无效数据。

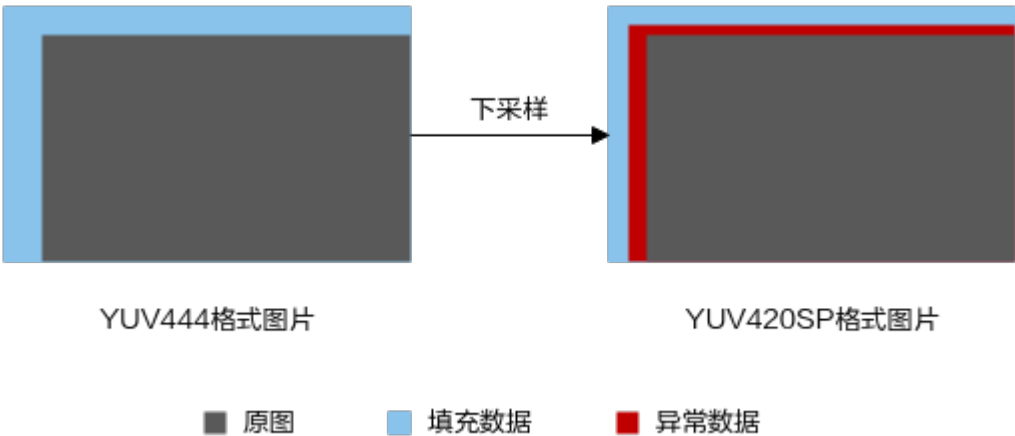
产品版本	抠图、贴图约束说明
Atlas 训练系列产品	● 抠图区域不超出输入图片区域。 ● 贴图区域不超出输出图片区域，最大贴图个数256个。 ● 贴图区域相对输出图片的左偏移16对齐。 - 贴图时可直接放置在输出图片的最左侧，即相对输出图片的左偏移为0，0也满足16对齐的要求。 - 输出图片的贴图宽度建议16对齐，如果不是16对齐，会多写一段无效数据使其16对齐。见图1-4、图1-5，贴图区域旁边的绿色框就表示无效数据。 - 在等比例缩放场景下，由于贴图左偏移16对齐的限制，如果直接调用acldvppVpcCropResizePasteAsync接口实现缩放功能，贴图的位置可能不在输出图片的正中心位置，此时可通过抠图、缩放、填充几个功能配合使用（调用acldvppVpcBatchCropResizeMakeBorderAsync接口），实现等比例缩放。 ● 抠图、贴图区域的奇数、偶数限制： 仅输出图片格式为YUV420SP，贴图区域奇数、偶数限制为：左偏移和上偏移为偶数、右偏移和下偏移为奇数。 ● 缩放场景下，贴图/抠图的宽或高缩放比例范围：[1/32, 16]。 ● 推理场景下，VPC的输出作为模型推理的输入时，由于贴图区域的宽*高有16*2对齐的约束，若贴图有效区域的宽*高不满足16*2对齐时，则贴图区域中会存在一些补边的无效数据，为保证推理精度，贴图有效区域的宽*高不满足16*2对齐时，用户需借助AIPP抠出有效区域，去除无效数据。
Atlas 推理系列产品	● 抠图区域不超出输入图片区域。 ● 贴图区域不超出输出图片区域，最大贴图个数256个。 ● 贴图区域相对输出图片的左偏移16对齐。 - 贴图时可直接放置在输出图片的最左侧，即相对输出图片的左偏移为0，0也满足16对齐的要求。 - 输出图片的贴图宽度建议16对齐，如果不是16对齐，会多写一段无效数据使其16对齐。见图1-4、图1-5，贴图区域旁边的绿色框就表示无效数据。 - 在等比例缩放场景下，由于贴图左偏移16对齐的限制，如果直接调用acldvppVpcCropResizePasteAsync接口实现缩放功能，贴图的位置可能不在输出图片的正中心位置，此时可通过抠图、缩放、填充几个功能配合使用（调用acldvppVpcBatchCropResizeMakeBorderAsync接口），实现等比例缩放。 ● 抠图、贴图区域的奇数、偶数限制： 输出图片格式为YUV420SP，贴图区域奇数、偶数限制为：左偏移和上偏移为偶数、右偏移和下偏移为奇数。 输出图片格式为YUV422SP，贴图区域奇数、偶数限制为：左偏移为偶数、右偏移为奇数。 其它格式，贴图区域没有偏移奇偶数的限制。

产品版本	抠图、贴图约束说明
Atlas 200/500 A2推理产品	• 抠图区域不超出输入图片区域。 • 贴图区域不超出输出图片区域，最大贴图个数256个。 • 抠图、贴图区域的奇数、偶数限制： 抠图起始坐标无奇数、偶数限制； 当输出图片格式为YUV420时，贴图区奇数、偶数限制为：左偏移和上偏移为偶数、右偏移和下偏移为奇数； 当输出图片格式为YUV422时，贴图区奇数、偶数限制为：左偏移为偶数、右偏移为奇数。
Atlas A2训练系列产品/ Atlas 800I A2推理产品	• 抠图区域不超出输入图片区域。 • 贴图区域不超出输出图片区域，最大贴图个数256个。 • 抠图、贴图区域的奇数、偶数限制： 抠图起始坐标无奇数、偶数限制； 输出图片格式为YUV420时，贴图区奇数、偶数限制为：左偏移和上偏移为偶数、右偏移和下偏移为奇数； 输出图片格式为YUV422时，贴图区奇数、偶数限制为：左偏移为偶数、右偏移为奇数。

YUV 格式图像下采样约束

VPC在处理图片时，会根据输入或输出图片格式，将输入图片格式转换为YUV444或RGB用于内部处理，YUV444或RGB没有宽高奇偶数的限制，但当输出图片格式为YUV420SP或YUV422SP格式时，会进行下采样处理，由于YUV420SP或YUV422SP格式本身的数据排布导致宽高存在奇偶数限制，因此输出图片的边缘可能存在异常数据。

图 1-9 异常效果图片举例



出现异常数据的根因在于，在计算过程中，出现输出图片位置处于奇数起始点时，此时YUV444格式的图片是正确的，但是下采样到YUV420SP格式时，由于奇数行和偶数行是共用同一个uv的，导致图片起始行的Y与上一行的UV组成新的像素，产生异常数据。

1.8.5.3 性能指标说明

本节仅描述各产品对应的昇腾AI处理器的性能指标参考数据，各产品由于其内存、CPU等配置不同，在性能规格上会存在差异，详细产品的性能规格请查阅[昇腾社区](#)的官方说明。

本节以单个昇腾AI处理器为例，描述基本场景性能指标参考，如下表所示，其它分辨率可以等量估算。

性能指标说明（Atlas 200/300/500 推理产品）

以下性能数据，是基于一个stream上下发10个异步媒体数据处理任务后，执行一次[aclrtSynchronizeStream](#)接口。

- 对于非8K图像，单个Device的基本场景性能指标参考如下（1路对应一个通道，一个通道对应一个线程）：
对于1080p的图像，若存在Host->Device的图片数据拷贝，由于拷贝带宽限制，最大总帧率约为1000fps。
对于4K的图像，若存在Host->Device的图片数据拷贝，由于拷贝带宽限制，最大总帧率约为250fps。

表 1-11 Atlas 200/300/500 推理产品的昇腾 AI 处理器性能指标参考数据

场景举例	总帧率
- 输入图像分辨率：1080p（1920*1080） - 输出图像分辨率：1080p（1920*1080） - 输入/输出图片格式：YUV420SP - n路（n<4）	n*360fps
- 输入图像分辨率：1080p（1920*1080） - 输出图像分辨率：1080p（1920*1080） - 输入/输出图片格式：YUV420SP - n路（n≥4）	1440fps
- 输入图像分辨率：4K图像（3840*2160） - 输出图像分辨率：4K图像（3840*2160） - 输入/输出图片格式：YUV420SP - n路（n<4）	n*90fps

场景举例	总帧率
• 输入图像分辨率：4K图像（ 3840*2160 ） • 输出图像分辨率：4K图像（ 3840*2160 ） • 输入/输出图片格式：YUV420SP • n路（ n≥4 ）	360fps

- 对于8K图像，单个Device的典型场景性能指标参考如下（ 1路对应一个通道，一个通道对应一个线程 ）：

表 1-12 Atlas 200/300/500 推理产品的昇腾 AI 处理器性能指标参考数据

场景举例	总帧率
• 输入图像分辨率：8K图像（ 7680*4320 ） • 输出图像分辨率：1080p（ 1920*1080 ） • 输入/输出图片格式：YUV420SP • n路（ n<4 ）	n*25fps
• 输入图像分辨率：8K图像（ 7680*4320 ） • 输出图像分辨率：1080p（ 1920*1080 ） • 输入/输出图片格式：YUV420SP • n路（ n≥4 ）	100fps
• 输入图像分辨率：8K图像（ 7680*4320 ） • 输出图像分辨率：4K图像（ 3840*2160 ） • 输入/输出图片格式：YUV420SP • n路（ n<4 ）	n*25fps
• 输入图像分辨率：8K图像（ 7680*4320 ） • 输出图像分辨率：4K图像（ 3840*2160 ） • 输入/输出图片格式：YUV420SP • n路（ n≥4 ）	100fps

 说明

- VPC处理性能与处理过程中的图像分辨率强相关，以输入图像和输出图像中的最大分辨率作为基准分辨率，基准分辨率越大，处理耗时越久，性能越低。
- 调用VPC批处理接口（接口命名中包含Batch，例如aclvppVpcBatchCropAsync接口）时，由于图像处理单元DVPP（Digital Video Pre-Processing）内部多个VPC硬件单元会并行处理图片任务，因此单路就可以达到最大总帧率。

性能指标说明（Atlas 推理系列产品）

以下性能数据，是基于一个stream上下发64个异步媒体数据处理任务后，执行一次[aclrtSynchronizeStream](#)接口，且Host服务器配置为：CPU主频为Intel(R) Xeon(R) CPU E5-2680 v4 @ 2.40GHz，CPU核心数为14。

单个Device的基本场景性能指标参考如下（1路对应一个通道，一个通道对应一个线程）：

表 1-13 Atlas 推理系列产品的昇腾 AI 处理器性能指标参考数据

场景举例	总帧率
• 输入图片分辨率：1080p（1920*1080） • 输出图片分辨率：1080p（1920*1080） • 输入/输出图片格式：YUV420SP • n路（1≤n≤12）	n*360fps
• 输入图片分辨率：1080p（1920*1080） • 输出图片分辨率：1080p（1920*1080） • 输入/输出图片格式：YUV420SP • n路（n>12）	4320fps
• 输入图片分辨率：4K图像（3840*2160） • 输出图片分辨率：4K图像（3840*2160） • 输入/输出图片格式：YUV420SP • n路（1≤n≤12）	n*90fps
• 输入图片分辨率：4K图像（3840*2160） • 输出图片分辨率：4K图像（3840*2160） • 输入/输出图片格式：YUV420SP • n路（n>12）	1080fps

场景举例	总帧率
● 输入图片分辨率：8K图像（7680*4320） ● 输出图片分辨率：4K图像（3840*2160） ● 输入/输出图片格式：YUV420SP ● n路（1≤n≤12）	n*22.5fps
● 输入图片分辨率：8K图像（7680*4320） ● 输出图片分辨率：4K图像（3840*2160） ● 输入/输出图片格式：YUV420SP ● n路（n>12）	270fps

 说明

调用VPC批处理接口（接口命名中包含Batch，例如acldvppVpcBatchCropAsync接口）时，由于图像处理单元DVPP（Digital Video Pre-Processing）内部多个VPC硬件单元会并行处理图片任务，因此单路就可以达到最大总帧率。

性能指标说明（Atlas 训练系列产品）

以下性能数据，是基于一个stream上下发10个异步媒体数据处理任务后，执行一次[aclrtSynchronizeStream](#)接口。

- 对于非8K图像，单个Device的基本场景性能指标参考如下（1路对应一个通道，一个通道对应一个线程）：

表 1-14 Atlas 训练系列产品的昇腾 AI 处理器性能指标参考数据

场景举例	总帧率
● 输入图像分辨率：1080p（1920*1080） ● 输出图像分辨率：1080p（1920*1080） ● 输入/输出图片格式：YUV420SP ● n路（n<16）	n*360fps
● 输入图像分辨率：1080p（1920*1080） ● 输出图像分辨率：1080p（1920*1080） ● 输入/输出图片格式：YUV420SP ● n路（n≥16）	5760fps

场景举例	总帧率
- 输入图像分辨率：4K图像（3840*2160） - 输出图像分辨率：4K图像（3840*2160） - 输入/输出图片格式：YUV420SP - n路（n<16）	n*90fps
- 输入图像分辨率：4K图像（3840*2160） - 输出图像分辨率：4K图像（3840*2160） - 输入/输出图片格式：YUV420SP - n路（n≥16）	1440fps

- 对于8K图像，单个Device的典型场景性能指标参考如下（1路对应一个通道，一个通道对应一个线程）：

表 1-15 Atlas 训练系列产品的昇腾 AI 处理器性能指标参考数据

场景举例	总帧率
- 输入图像分辨率：8K图像（7680*4320） - 输出图像分辨率：1080p（1920*1080） - 输入/输出图片格式：YUV420SP - n路（n<16）	n*25fps
- 输入图像分辨率：8K图像（7680*4320） - 输出图像分辨率：1080p（1920*1080） - 输入/输出图片格式：YUV420SP - n路（n≥16）	400fps
- 输入图像分辨率：8K图像（7680*4320） - 输出图像分辨率：4K图像（3840*2160） - 输入/输出图片格式：YUV420SP - n路（n<16）	n*25fps

场景举例	总帧率
• 输入图像分辨率：8K图像（ 7680*4320 ） • 输出图像分辨率：4K图像（ 3840*2160 ） • 输入/输出图片格式：YUV420SP • n路（ n≥16 ）	400fps

 说明

调用VPC批处理接口（接口命名中包含Batch，例如acldvppVpcBatchCropAsync接口）时，由于图像处理单元DVPP（Digital Video Pre-Processing）内部多个VPC硬件单元会并行处理图片任务，因此单路就可以达到最大总帧率。

性能指标说明（Atlas 200/500 A2 推理产品）

单个Device场景下的性能指标参考如下（1路对应一个通道，一个通道对应一个线程，或者n路对应一个通道，一个通道对应n个线程）：

表 1-16 Atlas 200/500 A2 推理产品的昇腾 AI 处理器性能指标参考数据

场景举例	总帧率
• 输入图片分辨率：1080p（ 1920*1080 ） • 输出图片分辨率：1080p（ 1920*1080 ） • 输入/输出图片格式：YUV420SP • n路（ n<2 ）	n*800fps
• 输入图片分辨率：1080p（ 1920*1080 ） • 输出图片分辨率：1080p（ 1920*1080 ） • 输入/输出图片格式：YUV420SP • n路（ n≥2 ）	1600fps
• 输入图片分辨率：4K图像（ 3840*2160 ） • 输出图片分辨率：4K图像（ 3840*2160 ） • 输入/输出图片格式：YUV420SP • n路（ n<2 ）	n*200fps
• 输入图片分辨率：4K图像（ 3840*2160 ） • 输出图片分辨率：4K图像（ 3840*2160 ） • 输入/输出图片格式：YUV420SP • n路（ n≥2 ）	400fps

场景举例	总帧率
- 输入图片分辨率：8K图像（7680*4320） - 输出图片分辨率：4K图像（3840*2160） - 输入/输出图片格式：YUV420SP - n路（$n < 2$）	$n * 100\text{fps}$
- 输入图片分辨率：8K图像（7680*4320） - 输出图片分辨率：4K图像（3840*2160） - 输入/输出图片格式：YUV420SP - n路（$n \geq 2$）	200fps

 说明

调用VPC批处理接口（接口命名中包含batch，例如acldvppVpcBatchCropAsync接口）时，由于图像处理单元DVPP（Digital Video Pre-Processing）内部多个VPC硬件单元会并行处理图片任务，因此单路就可以达到最大总帧率。

性能指标说明（Atlas A2 训练系列产品/Atlas 800I A2 推理产品）

表 1-17 Atlas A2 训练系列产品/Atlas 800I A2 推理产品的昇腾 AI 处理器性能指标参考数据

场景举例	总帧率
- 输入图片分辨率：1080p（1920*1080） - 输出图片分辨率：1080p（1920*1080） - 输入/输出图片格式：YUV420SP - n路（$1 \leq n \leq 10$）	$n * 1440\text{fps}$
- 输入图片分辨率：1080p（1920*1080） - 输出图片分辨率：1080p（1920*1080） - 输入/输出图片格式：YUV420SP - n路（$n > 10$）	14400fps
- 输入图片分辨率：4K图像（3840*2160） - 输出图片分辨率：4K图像（3840*2160） - 输入/输出图片格式：YUV420SP - n路（$1 \leq n \leq 10$）	$n * 360\text{fps}$
- 输入图片分辨率：4K图像（3840*2160） - 输出图片分辨率：4K图像（3840*2160） - 输入/输出图片格式：YUV420SP - n路（$n > 10$）	3600fps

场景举例	总帧率
● 输入图片分辨率：8K图像（7680*4320） ● 输出图片分辨率：4K图像（3840*2160） ● 输入/输出图片格式：YUV420SP ● n路（1≤n≤10）	n*90fps
● 输入图片分辨率：8K图像（7680*4320） ● 输出图片分辨率：4K图像（3840*2160） ● 输入/输出图片格式：YUV420SP ● n路（n>10）	900fps

1.8.5.4 acldvppVpcResizeAsync

函数功能

将输入图片缩放到输出图片大小，异步接口。

约束说明

版本	约束
Atlas 200/300/500 推理产品 Atlas 训练系列产品	● 关于输入图片分辨率、图片格式： - 输入图片分辨率在[10*6, 4096*4096]范围内时，支持图片格式、宽高对齐、内存约束处说明的输入图片格式。 - 输入图片的宽在(4096, 8192]或高在(4096, 8192]范围内时，支持格式如下： 支持acldvppPixelFormat枚举值中的如下枚举项： PIXEL_FORMAT_YUV_SEMIPLANAR_420 = 1, // YUV420SP NV12 8bit PIXEL_FORMAT_YVU_SEMIPLANAR_420 = 2, // YUV420SP NV21 8bit - 如果输入图片的宽高不是2对齐，VPC会自动处理成向下2对齐，例如输入图片的宽高为15*15，VPC会先处理成14*14，再进行缩放。 ● 关于输出图片分辨率、图片格式请参见图片格式、宽高对齐、内存约束。 ● 关于各图片格式的宽高对齐、内存大小约束，请参见图片格式、宽高对齐、内存约束。

版本	约束
Atlas 推理系列产品	- 关于输入图片分辨率、图片格式： - 输入图片分辨率在[10*6, 4096*4096]范围内时，支持图片格式、宽高对齐、内存约束处说明的输入图片格式。 - 输入图片的宽在(4096, 8192]或高在(4096, 8192]范围内时，支持格式如下： 支持aclDvppPixelFormat枚举值中的如下枚举项： PIXEL_FORMAT_YUV_400 = 0, // YUV400 8bit PIXEL_FORMAT_YUV_SEMIPLANAR_420 = 1, // YUV420SP NV12 8bit PIXEL_FORMAT_YVU_SEMIPLANAR_420 = 2, // YUV420SP NV21 8bit PIXEL_FORMAT_YUV_SEMIPLANAR_422 = 3, // YUV422SP 8bit PIXEL_FORMAT_YVU_SEMIPLANAR_422 = 4, // YVU422SP 8bit PIXEL_FORMAT_YUV_SEMIPLANAR_444 = 5, // YUV444SP 8bit PIXEL_FORMAT_YVU_SEMIPLANAR_444 = 6, // YVU444SP 8bit PIXEL_FORMAT_YUYV_PACKED_422 = 7, // YUV422Packed YUYV 8bit PIXEL_FORMAT_UYVY_PACKED_422 = 8, // YUV422Packed UYVY 8bit PIXEL_FORMAT_VYVU_PACKED_422 = 9, // YUV422Packed VYVU 8bit PIXEL_FORMAT_VYUY_PACKED_422 = 10, // YUV422Packed VYUY 8bit - 如果输入图片的宽高不是2对齐，VPC会自动处理成向下2对齐，例如输入图片的宽高为15*15，VPC会先处理成14*14，再进行缩放。 - 关于输出图片分辨率、图片格式请参见图片格式、宽高对齐、内存约束。 - 关于各图片格式的宽高对齐、内存大小约束，请参见图片格式、宽高对齐、内存约束。 - 当输入、输出图片格式都为ARGB8888、ABGR8888、RGBA8888、BGRA8888格式时，在经过本接口缩放处理后，A分量值会变为0。
Atlas A2训练系列产品/Atlas 800I A2推理产品 Atlas 200/500 A2推理产品	- 关于输入、输出图片分辨率： - 输入图片分辨率：[10*6, 8192*8192] - 输出图片分辨率：[10*6, 4096*8192] - 关于图片格式、宽高对齐、内存等总体约束，请参见图片格式、宽高对齐、内存约束。 - 当输入、输出图片格式都为ARGB8888、ABGR8888、RGBA8888、BGRA8888格式时，在经过本接口缩放处理后，A分量值会变为0。

函数原型

```
aclError aclDvppVpcResizeAsync(aclDvppChannelDesc *channelDesc,  
aclDvppPicDesc *inputDesc,  
aclDvppPicDesc *outputDesc,
```

```
acldvppResizeConfig *resizeConfig,  
aclrtStream stream)
```

参数说明

参数名	输入/输出	说明
channelDesc	输入	通道描述信息的指针。 与调用 acldvppCreateChannel 接口创建通道时指定的channelDesc保持一致。
inputDesc	输入	输入图片信息的指针。 - 调用acldvppCreatePicDesc接口创建图片描述信息。 - 调用acldvppSetPicDesc系列接口设置图片描述参数（例如，内存地址、内存大小、图片格式、宽、高等）。
outputDesc	输入&输出	输出图片信息的指针。 outputDesc参数作为输入时，需要用户调用如下接口： - 调用acldvppCreatePicDesc接口创建图片描述信息。 - 调用acldvppSetPicDesc系列接口设置图片描述参数（例如，内存地址、内存大小、图片格式、宽、高等）。
resizeConfig	输入	图片缩放配置数据的指针。 调用 acldvppCreateResizeConfig 接口创建图片缩放配置数据。
stream	输入	指定Stream。

返回值说明

返回0表示任务下发成功，返回非0表示任务下发失败。

参考资源

接口调用流程及示例，参见VPC图像处理典型功能

1.8.5.5 acldvppVpcCropAsync

函数功能

按指定区域从一张输入图片中抠出一张子图，再将抠的图片存放到输出内存中，作为输出图片，异步接口。

约束说明

版本	约束
Atlas 200/300/500 推理产品 Atlas 训练系列产品	- 关于输入图片分辨率、图片格式： - 输入图片分辨率在[10*6, 4096*4096]范围内时，支持图片格式、宽高对齐、内存约束处说明的输入图片格式。 - 输入图片的宽在(4096, 8192]或高在(4096, 8192]范围内时，支持格式如下： 支持acldvppPixelFormat枚举值中的如下枚举项： PIXEL_FORMAT_YUV_SEMIPLANAR_420 = 1, // YUV420SP NV12 8bit PIXEL_FORMAT_YVU_SEMIPLANAR_420 = 2, // YUV420SP NV21 8bit - 关于输出图片分辨率、图片格式请参见图片格式、宽高对齐、内存约束。 - 关于各图片格式的宽高对齐、内存大小约束，请参见图片格式、宽高对齐、内存约束。 - 输出图片区域与cropArea不一致时会对图片再做一次缩放操作，默认缩放算法为“自研的高阶滤波算法”。 - 由于通过本接口抠的图直接作为输出图片，所以需满足输出图片最大分辨率4096*4096的限制，因此本接口中抠图区域最大分辨率为4096*4096。

版本	约束
Atlas 推理系列产品	- 关于输入图片分辨率、图片格式： - 输入图片分辨率在[10*6, 4096*4096]范围内时，支持图片格式、宽高对齐、内存约束处说明的输入图片格式。 - 输入图片的宽在(4096, 8192]或高在(4096, 8192]范围内时，支持格式如下： 支持acldvppPixelFormat枚举值中的如下枚举项： PIXEL_FORMAT_YUV_400 = 0, // YUV400 8bit PIXEL_FORMAT_YUV_SEMIPLANAR_420 = 1, // YUV420SP NV12 8bit PIXEL_FORMAT_YVU_SEMIPLANAR_420 = 2, // YUV420SP NV21 8bit PIXEL_FORMAT_YUV_SEMIPLANAR_422 = 3, // YUV422SP 8bit PIXEL_FORMAT_YVU_SEMIPLANAR_422 = 4, // YVU422SP 8bit PIXEL_FORMAT_YUV_SEMIPLANAR_444 = 5, // YUV444SP 8bit PIXEL_FORMAT_YVU_SEMIPLANAR_444 = 6, // YVU444SP 8bit PIXEL_FORMAT_YUYV_PACKED_422 = 7, // YUV422Packed YUYV 8bit PIXEL_FORMAT_UYVY_PACKED_422 = 8, // YUV422Packed UYVY 8bit PIXEL_FORMAT_VYVU_PACKED_422 = 9, // YUV422Packed VYVU 8bit PIXEL_FORMAT_VYUY_PACKED_422 = 10, // YUV422Packed VYUY 8bit - 关于输出图片分辨率、图片格式请参见图片格式、宽高对齐、内存约束。 - 关于各图片格式的宽高对齐、内存大小约束，请参见图片格式、宽高对齐、内存约束。 - 输出图片区域与cropArea不一致时会对图片再做一次缩放操作，默认缩放算法为“业界通用的Bilinear算法”。 - 由于通过本接口抠的图直接作为输出图片，所以需满足输出图片最大分辨率4096*4096的限制，因此本接口中抠图区域最大分辨率为4096*4096。
Atlas A2训练系列产品/Atlas 800I A2推理产品 Atlas 200/500 A2推理产品	- 关于输入、输出图片分辨率： - 输入图片分辨率：[10*6, 8192*8192] - 输出图片分辨率：[10*6, 4096*8192] - 关于图片格式、宽高对齐、内存等总体约束，请参见图片格式、宽高对齐、内存约束。 - 输出图片区域与cropArea不一致时会对图片再做一次缩放操作，默认缩放算法为“业界通用的Bilinear算法”。 - 由于通过本接口抠的图直接作为输出图片，所以需满足输出图片最大分辨率4096*8192的限制，因此本接口中抠图区域最大分辨率为4096*8192。

函数原型

```
aclError acldvppVpcCropAsync(acldvppChannelDesc *channelDesc,  
                             acldvppPicDesc *inputDesc,
```

```
acldvppPicDesc *outputDesc,  
acldvppRoiConfig *cropArea,  
aclrtStream stream)
```

参数说明

参数名	输入/输出	说明
channelDesc	输入	通道描述信息的指针。 与调用 acldvppCreateChannel 接口创建通道时指定的channelDesc保持一致。
inputDesc	输入	输入图片信息的指针。 - 调用acldvppCreatePicDesc接口创建图片描述信息。 - 调用acldvppSetPicDesc系列接口设置图片描述参数（例如，内存地址、内存大小、图片格式、宽、高等）。
outputDesc	输入&输出	输出图片信息的指针。 outputDesc参数作为输入时，需要用户调用如下接口： - 调用acldvppCreatePicDesc接口创建图片描述信息。 - 调用acldvppSetPicDesc系列接口设置图片描述参数（例如，内存地址、内存大小、图片格式、宽、高等）。
cropArea	输入	抠图区域位置的指针。 调用 acldvppCreateRoiConfig 接口创建区域位置数据。
stream	输入	指定Stream。

返回值说明

返回0表示任务下发成功，返回非0表示任务下发失败。

参考资源

接口调用流程及示例，参见VPC图像处理典型功能

1.8.5.6 acldvppVpcCropResizeAsync

函数功能

按指定区域从一张输入图片中抠出一张子图，再将抠的图片存放到输出内存中，作为输出图片，异步接口。

输出图片区域与cropArea不一致时会对图片再做一次缩放操作，按照resizeConfig处配置的缩放算法。

约束说明

版本	约束
Atlas 200/300/500 推理产品 Atlas 训练系列产品	- 关于输入图片分辨率、图片格式： - 输入图片分辨率在[10*6, 4096*4096]范围内时，支持图片格式、宽高对齐、内存约束处说明的输入图片格式。 - 输入图片的宽在(4096, 8192]或高在(4096, 8192]范围内时，支持格式如下： 支持acldvppPixelFormat枚举值中的如下枚举项： PIXEL_FORMAT_YUV_SEMIPLANAR_420 = 1, // YUV420SP NV12 8bit PIXEL_FORMAT_YVU_SEMIPLANAR_420 = 2, // YUV420SP NV21 8bit - 关于输出图片分辨率、图片格式请参见图片格式、宽高对齐、内存约束。 - 关于各图片格式的宽高对齐、内存大小约束，请参见图片格式、宽高对齐、内存约束。
Atlas 推理系列产品	- 关于输入图片分辨率、图片格式： - 输入图片分辨率在[10*6, 4096*4096]范围内时，支持图片格式、宽高对齐、内存约束处说明的输入图片格式。 - 输入图片的宽在(4096, 8192]或高在(4096, 8192]范围内时，支持格式如下： 支持acldvppPixelFormat枚举值中的如下枚举项： PIXEL_FORMAT_YUV_400 = 0, // YUV400 8bit PIXEL_FORMAT_YUV_SEMIPLANAR_420 = 1, // YUV420SP NV12 8bit PIXEL_FORMAT_YVU_SEMIPLANAR_420 = 2, // YUV420SP NV21 8bit PIXEL_FORMAT_YUV_SEMIPLANAR_422 = 3, // YUV422SP 8bit PIXEL_FORMAT_YVU_SEMIPLANAR_422 = 4, // YVU422SP 8bit PIXEL_FORMAT_YUV_SEMIPLANAR_444 = 5, // YUV444SP 8bit PIXEL_FORMAT_YVU_SEMIPLANAR_444 = 6, // YVU444SP 8bit PIXEL_FORMAT_YUYV_PACKED_422 = 7, // YUV422Packed YUYV 8bit PIXEL_FORMAT_UYVY_PACKED_422 = 8, // YUV422Packed UYVY 8bit PIXEL_FORMAT_YVYU_PACKED_422 = 9, // YUV422Packed YVYU 8bit PIXEL_FORMAT_VYUY_PACKED_422 = 10, // YUV422Packed VYUY 8bit - 关于输出图片分辨率、图片格式请参见图片格式、宽高对齐、内存约束。 - 关于各图片格式的宽高对齐、内存大小约束，请参见图片格式、宽高对齐、内存约束。 - 当输入、输出图片格式都为ARGB8888、ABGR8888、RGBA8888、BGRA8888格式时，在经过本接口缩放处理后，A分量值会变为0。

版本	约束
Atlas A2训练系列产品/Atlas 800I A2推理产品 Atlas 200/500 A2推理产品	- 关于输入、输出图片分辨率： - 输入图片分辨率：[10*6, 8192*8192] - 输出图片分辨率：[10*6, 4096*8192] - 关于图片格式、宽高对齐、内存等总体约束，请参见图片格式、宽高对齐、内存约束。 - 当输入、输出图片格式都为ARGB8888、ABGR8888、RGBA8888、BGRA8888格式时，在经过本接口缩放处理后，A分量值会变为0。

函数原型

```
aclError acldvppVpcCropResizeAsync(acldvppChannelDesc *channelDesc,  
acldvppPicDesc *inputDesc,  
acldvppPicDesc *outputDesc,  
acldvppRoiConfig *cropArea,  
acldvppResizeConfig *resizeConfig,  
aclrtStream stream)
```

参数说明

参数名	输入/输出	说明
channelDesc	输入	通道描述信息的指针。 与调用 acldvppCreateChannel 接口创建通道时指定的channelDesc保持一致。
inputDesc	输入	输入图片信息的指针。 - 调用acldvppCreatePicDesc接口创建图片描述信息。 - 调用acldvppSetPicDesc系列接口设置图片描述参数（例如，内存地址、内存大小、图片格式、宽、高等）。
outputDesc	输入&输出	输出图片信息的指针。 outputDesc参数作为输入时，需要用户调用如下接口： - 调用acldvppCreatePicDesc接口创建图片描述信息。 - 调用acldvppSetPicDesc系列接口设置图片描述参数（例如，内存地址、内存大小、图片格式、宽、高等）。

参数名	输入/输出	说明
cropArea	输入	抠图区域位置的指针。 调用 aclDvppCreateRoiConfig 接口创建区域位置数据。
resizeConfig	输入	图片缩放配置数据的指针。 需提前调用 aclDvppCreateResizeConfig 接口创建图片缩放配置数据。
stream	输入	指定Stream。

返回值说明

返回0表示任务下发成功，返回非0表示任务下发失败。

参考资源

接口调用示例，参见抠图缩放（一图一框）示例代码。

1.8.5.7 aclDvppVpcBatchCropAsync

函数功能

对于一张或多张输入图片，支持从每个输入图片中抠出一张或多张子图并一一对应存放到多个输出区域。异步接口。

约束说明

版本	约束
Atlas 200/300/500 推理产品 Atlas 训练系列产品	- 关于输入图片分辨率、图片格式： - 输入图片分辨率在[10*6, 4096*4096]范围内时，支持图片格式、宽高对齐、内存约束处说明的输入图片格式。 - 输入图片的宽在(4096, 8192]或高在(4096, 8192]范围内时，支持格式如下： 支持aclDvppPixelFormat枚举值中的如下枚举项： PIXEL_FORMAT_YUV_SEMIPLANAR_420 = 1, // YUV420SP NV12 8bit PIXEL_FORMAT_YVU_SEMIPLANAR_420 = 2, // YUV420SP NV21 8bit - 关于输出图片分辨率、图片格式请参见图片格式、宽高对齐、内存约束。 - 关于各图片格式的宽高对齐、内存大小约束，请参见图片格式、宽高对齐、内存约束。 - 输出图片区域与cropArea不一致时会对图片再做一次缩放操作，默认缩放算法为“自研的高阶滤波算法”。

版本	约束
Atlas 推理系列产品	- 关于输入图片分辨率、图片格式： - 输入图片分辨率在[10*6, 4096*4096]范围内时，支持图片格式、宽高对齐、内存约束处说明的输入图片格式。 - 输入图片的宽在(4096, 8192]或高在(4096, 8192]范围内时，支持格式如下： 支持aclDvppPixelFormat枚举值中的如下枚举项： PIXEL_FORMAT_YUV_400 = 0, // YUV400 8bit PIXEL_FORMAT_YUV_SEMIPLANAR_420 = 1, // YUV420SP NV12 8bit PIXEL_FORMAT_YVU_SEMIPLANAR_420 = 2, // YUV420SP NV21 8bit PIXEL_FORMAT_YUV_SEMIPLANAR_422 = 3, // YUV422SP 8bit PIXEL_FORMAT_YVU_SEMIPLANAR_422 = 4, // YVU422SP 8bit PIXEL_FORMAT_YUV_SEMIPLANAR_444 = 5, // YUV444SP 8bit PIXEL_FORMAT_YVU_SEMIPLANAR_444 = 6, // YVU444SP 8bit PIXEL_FORMAT_YUYV_PACKED_422 = 7, // YUV422Packed YUYV 8bit PIXEL_FORMAT_UYVY_PACKED_422 = 8, // YUV422Packed UYVY 8bit PIXEL_FORMAT_VYVU_PACKED_422 = 9, // YUV422Packed VYVU 8bit PIXEL_FORMAT_VYUY_PACKED_422 = 10, // YUV422Packed VYUY 8bit - 关于输出图片分辨率、图片格式请参见图片格式、宽高对齐、内存约束。 - 关于各图片格式的宽高对齐、内存大小约束，请参见图片格式、宽高对齐、内存约束。 - 输出图片区域与cropArea不一致时会对图片再做一次缩放操作，默认缩放算法为“业界通用的Bilinear算法”。
Atlas A2训练系列产品/Atlas 800I A2推理产品 Atlas 200/500 A2推理产品	- 关于输入、输出图片分辨率： - 输入图片分辨率：[10*6, 8192*8192] - 输出图片分辨率：[10*6, 4096*8192] - 关于图片格式、宽高对齐、内存等总体约束，请参见图片格式、宽高对齐、内存约束。 - 输出图片区域与cropArea不一致时会对图片再做一次缩放操作，默认缩放算法为“业界通用的Bilinear算法”。

函数原型

```
aclError aclDvppVpcBatchCropAsync(aclDvppChannelDesc *channelDesc,  
aclDvppBatchPicDesc *srcBatchPicDescs,  
uint32_t *roiNums,  
uint32_t size,  
aclDvppBatchPicDesc *dstBatchPicDescs,  
aclDvppRoiConfig *cropAreas[],
```

aclrtStream stream)

参数说明

参数名	输入/输出	说明
channelDesc	输入	通道描述信息的指针。 与调用 aclvppCreateChannel 接口创建通道时指定的channelDesc保持一致。
srcBatchPicDescs	输入	批量输入图片描述信息的指针。 - 调用aclvppCreateBatchPicDesc接口创建批量图片描述信息。 - 调用aclvppGetPicDesc接口获取指定图片的图片描述信息。 - 调用aclvppSetPicDesc系列接口设置图片描述参数（例如，内存地址、内存大小、图片格式、宽、高等）。
roiNums	输入	抠图数量的指针，当前最大抠图数量为256。 roiNums为数组，数组总和小于等于256，与dstBatchPicDescs结构体中的batchSize值保持一致。 roiNums[0]+...+roiNums[size-1] <= 256
size	输入	表示roiNums数组中的元素个数，个数小于等于256。
dstBatchPicDescs	输入&输出	批量输出图片描述信息的指针。 dstBatchPicDescs参数作为输入时，需要用户调用如下接口： - 调用aclvppCreateBatchPicDesc接口创建批量图片描述信息。 - 调用aclvppGetPicDesc接口获取指定图片的图片描述信息。 - 调用aclvppSetPicDesc系列接口设置图片描述参数（例如，内存地址、内存大小、图片格式、宽、高等）。
cropAreas	输入	抠图区域位置的指针数组。 调用 aclvppCreateRoiConfig 接口创建区域位置数据。
stream	输入	指定Stream。

返回值说明

返回0表示任务下发成功，返回非0表示任务下发失败。

1.8.5.8 aclvppVpcBatchCropResizeAsync

函数功能

对于一张或多张输入图片，支持从每个输入图片中抠出一张或多张子图并——对应存放到多个输出区域，输出图片区域与抠图区域不一致时会对图片再做一次缩放操作，按照resizeConfig处配置的缩放算法。异步接口。

约束说明

版本	约束
Atlas 200/300/500 推理产品 Atlas 训练系列产品	- 关于输入图片分辨率、图片格式： - 输入图片分辨率在[10*6, 4096*4096]范围内时，支持图片格式、宽高对齐、内存约束处说明的输入图片格式。 - 输入图片的宽在(4096, 8192]或高在(4096, 8192]范围内时，支持格式如下： 支持aclvppPixelFormat枚举值中的如下枚举项： PIXEL_FORMAT_YUV_SEMIPLANAR_420 = 1, // YUV420SP NV12 8bit PIXEL_FORMAT_YVU_SEMIPLANAR_420 = 2, // YUV420SP NV21 8bit - 关于输出图片分辨率、图片格式请参见图片格式、宽高对齐、内存约束。 - 关于各图片格式的宽高对齐、内存大小约束，请参见图片格式、宽高对齐、内存约束。

版本	约束
Atlas 推理系列产品	- 关于输入图片分辨率、图片格式： - 输入图片分辨率在[10*6, 4096*4096]范围内时，支持图片格式、宽高对齐、内存约束处说明的输入图片格式。 - 输入图片的宽在(4096, 8192]或高在(4096, 8192]范围内时，支持格式如下： 支持aclDvppPixelFormat枚举值中的如下枚举项： PIXEL_FORMAT_YUV_400 = 0, // YUV400 8bit PIXEL_FORMAT_YUV_SEMIPLANAR_420 = 1, // YUV420SP NV12 8bit PIXEL_FORMAT_YVU_SEMIPLANAR_420 = 2, // YUV420SP NV21 8bit PIXEL_FORMAT_YUV_SEMIPLANAR_422 = 3, // YUV422SP 8bit PIXEL_FORMAT_YVU_SEMIPLANAR_422 = 4, // YVU422SP 8bit PIXEL_FORMAT_YUV_SEMIPLANAR_444 = 5, // YUV444SP 8bit PIXEL_FORMAT_YVU_SEMIPLANAR_444 = 6, // YVU444SP 8bit PIXEL_FORMAT_YUYV_PACKED_422 = 7, // YUV422Packed YUYV 8bit PIXEL_FORMAT_UYVY_PACKED_422 = 8, // YUV422Packed UYVY 8bit PIXEL_FORMAT_VYVU_PACKED_422 = 9, // YUV422Packed VYVU 8bit PIXEL_FORMAT_VYUY_PACKED_422 = 10, // YUV422Packed VYUY 8bit - 关于输出图片分辨率、图片格式请参见图片格式、宽高对齐、内存约束。 - 关于各图片格式的宽高对齐、内存大小约束，请参见图片格式、宽高对齐、内存约束。 - 当输入、输出图片格式都为ARGB8888、ABGR8888、RGBA8888、BGRA8888格式时，在经过本接口缩放处理后，A分量值会变为0。
Atlas A2训练系列产品/Atlas 800I A2推理产品 Atlas 200/500 A2推理产品	- 关于输入、输出图片分辨率： - 输入图片分辨率：[10*6, 8192*8192] - 输出图片分辨率：[10*6, 4096*8192] - 关于图片格式、宽高对齐、内存等总体约束，请参见图片格式、宽高对齐、内存约束。 - 当输入、输出图片格式都为ARGB8888、ABGR8888、RGBA8888、BGRA8888格式时，在经过本接口缩放处理后，A分量值会变为0。

函数原型

```
aclError aclDvppVpcBatchCropResizeAsync(aclDvppChannelDesc *channelDesc,
aclDvppBatchPicDesc *srcBatchPicDescs,
uint32_t *roiNums,
uint32_t size,
aclDvppBatchPicDesc *dstBatchPicDescs,
```

```
acldvppRoiConfig *cropAreas[],  
acldvppResizeConfig *resizeConfig,  
aclrtStream stream)
```

参数说明

参数名	输入/输出	说明
channelDesc	输入	通道描述信息的指针。 与调用 acldvppCreateChannel 接口创建通道时指定的channelDesc保持一致。
srcBatchPicDescs	输入	批量输入图片描述信息的指针。 - 调用acldvppCreateBatchPicDesc接口创建批量图片描述信息。 - 调用acldvppGetPicDesc接口获取指定图片的图片描述信息。 - 调用acldvppSetPicDesc系列接口设置图片描述参数（例如，内存地址、内存大小、图片格式、宽、高等）。
roiNums	输入	抠图数量的指针，当前最大抠图数量为256。 roiNums为数组，数组总和小于等于256，与dstBatchPicDescs结构体中的batchSize值保持一致。 roiNums[0]+...+roiNums[size-1] <= 256
size	输入	表示roiNums数组中的元素个数，个数小于等于256。
dstBatchPicDescs	输入&输出	批量输出图片描述信息的指针。 dstBatchPicDescs参数作为输入时，需要用户调用如下接口： - 调用acldvppCreateBatchPicDesc接口创建批量图片描述信息。 - 调用acldvppGetPicDesc接口获取指定图片的图片描述信息。 - 调用acldvppSetPicDesc系列接口设置图片描述参数（例如，内存地址、内存大小、图片格式、宽、高等）。
cropAreas	输入	抠图区域位置的指针数组。 调用 acldvppCreateRoiConfig 接口创建区域位置数据。
resizeConfig	输入	图片缩放配置数据的指针。 需提前调用 acldvppCreateResizeConfig 接口创建图片缩放配置数据。
stream	输入	指定Stream。

返回值说明

返回0表示任务下发成功，返回非0表示任务下发失败。

1.8.5.9 acldvppVpcCropAndPasteAsync

函数功能

按指定区域从一个输入图片中抠出一张子图，再将抠的子图贴到目标图片的指定位置，作为输出图片。异步接口。

约束说明

版本	约束
Atlas 200/300/500 推理产品 Atlas 训练系列产品	- 关于输入图片分辨率、图片格式： - 输入图片分辨率在[10*6, 4096*4096]范围内时，支持图片格式、宽高对齐、内存约束处说明的输入图片格式。 - 输入图片的宽在(4096, 8192]或高在(4096, 8192]范围内时，支持格式如下： 支持acldvppPixelFormat枚举值中的如下枚举项： PIXEL_FORMAT_YUV_SEMIPLANAR_420 = 1, // YUV420SP NV12 8bit PIXEL_FORMAT_YVU_SEMIPLANAR_420 = 2, // YUV420SP NV21 8bit - 关于输出图片分辨率、图片格式请参见图片格式、宽高对齐、内存约束。 - 关于各图片格式的宽高对齐、内存大小约束，请参见图片格式、宽高对齐、内存约束。 - 当pasteArea的宽高与cropArea的宽高不一致时会对图片做一次缩放操作，默认缩放算法为“自研的高阶滤波算法”。

版本	约束
Atlas 推理系列产品	- 关于输入图片分辨率、图片格式： - 输入图片分辨率在[10*6, 4096*4096]范围内时，支持图片格式、宽高对齐、内存约束处说明的输入图片格式。 - 输入图片的宽在(4096, 8192]或高在(4096, 8192]范围内时，支持格式如下： 支持acldvppPixelFormat枚举值中的如下枚举项： PIXEL_FORMAT_YUV_400 = 0, // YUV400 8bit PIXEL_FORMAT_YUV_SEMIPLANAR_420 = 1, // YUV420SP NV12 8bit PIXEL_FORMAT_YVU_SEMIPLANAR_420 = 2, // YUV420SP NV21 8bit PIXEL_FORMAT_YUV_SEMIPLANAR_422 = 3, // YUV422SP 8bit PIXEL_FORMAT_YVU_SEMIPLANAR_422 = 4, // YVU422SP 8bit PIXEL_FORMAT_YUV_SEMIPLANAR_444 = 5, // YUV444SP 8bit PIXEL_FORMAT_YVU_SEMIPLANAR_444 = 6, // YVU444SP 8bit PIXEL_FORMAT_YUYV_PACKED_422 = 7, // YUV422Packed YUYV 8bit PIXEL_FORMAT_UYVY_PACKED_422 = 8, // YUV422Packed UYVY 8bit PIXEL_FORMAT_VYVU_PACKED_422 = 9, // YUV422Packed VYVU 8bit PIXEL_FORMAT_VYUY_PACKED_422 = 10, // YUV422Packed VYUY 8bit - 关于输出图片分辨率、图片格式请参见图片格式、宽高对齐、内存约束。 - 关于各图片格式的宽高对齐、内存大小约束，请参见图片格式、宽高对齐、内存约束。 - 当pasteArea的宽高与cropArea的宽高不一致时会对图片做一次缩放操作，默认缩放算法为“业界通用的Bilinear算法”。
Atlas A2训练系列产品/Atlas 800I A2推理产品 Atlas 200/500 A2推理产品	- 关于图片分辨率、图片格式、宽高对齐、内存等总体约束，请参见图片格式、宽高对齐、内存约束。 - 当pasteArea的宽高与cropArea的宽高不一致时会对图片做一次缩放操作，默认缩放算法为“业界通用的Bilinear算法”。 - 由于YUV格式图像下采样约束，当输出图片格式为YUV420SP或YUV422SP格式时，贴图的顶部偏移、左侧偏移建议为偶数。

函数原型

```
aclError acldvppVpcCropAndPasteAsync(acldvppChannelDesc *channelDesc,  
acldvppPicDesc *inputDesc,  
acldvppPicDesc *outputDesc,  
acldvppRoiConfig *cropArea,  
acldvppRoiConfig *pasteArea,
```


aclrtStream stream)

参数说明

参数名	输入/输出	说明
channelDesc	输入	通道描述信息的指针。 与调用 aclvppCreateChannel 接口创建通道时指定的channelDesc保持一致。
inputDesc	输入	输入图片信息的指针。 - 调用aclvppCreatePicDesc接口创建图片描述信息。 - 调用aclvppSetPicDesc系列接口设置图片描述参数（例如，内存地址、内存大小、图片格式、宽、高等）。
outputDesc	输入&输出	输出图片信息的指针。 outputDesc参数作为输入时，需要用户调用如下接口： - 调用aclvppCreatePicDesc接口创建图片描述信息。 - 调用aclvppSetPicDesc系列接口设置图片描述参数（例如，内存地址、内存大小、图片格式、宽、高等）。
cropArea	输入	抠图区域位置的指针。 调用 aclvppCreateRoiConfig 接口创建区域位置数据。
pasteArea	输入	贴图区域位置的指针。 - 调用aclvppCreateRoiConfig接口创建区域位置数据。 - 贴图区域左偏移需要16对齐。
stream	输入	指定Stream。

返回值说明

返回0表示任务下发成功，返回非0表示任务下发失败。

1.8.5.10 aclvppVpcCropResizePasteAsync

函数功能

按指定区域从一个输入图片中抠出一张子图，再将抠的子图贴到目标图片的指定位置，作为输出图片。异步接口。

当pasteArea的宽高与cropArea的宽高不一致时会对图片做一次缩放操作，按照resizeConfig处配置的缩放算法。

约束说明

版本	约束
Atlas 200/300/500 推理产品 Atlas 训练系列产品	- 关于输入图片分辨率、图片格式： - 输入图片分辨率在[10*6, 4096*4096]范围内时，支持图片格式、宽高对齐、内存约束处说明的输入图片格式。 - 输入图片的宽在(4096, 8192]或高在(4096, 8192]范围内时，支持格式如下： 支持acldvppPixelFormat枚举值中的如下枚举项： PIXEL_FORMAT_YUV_SEMIPLANAR_420 = 1, // YUV420SP NV12 8bit PIXEL_FORMAT_YVU_SEMIPLANAR_420 = 2, // YUV420SP NV21 8bit - 关于输出图片分辨率、图片格式请参见图片格式、宽高对齐、内存约束。 - 关于各图片格式的宽高对齐、内存大小约束，请参见图片格式、宽高对齐、内存约束。
Atlas 推理系列产品	- 关于输入图片分辨率、图片格式： - 输入图片分辨率在[10*6, 4096*4096]范围内时，支持图片格式、宽高对齐、内存约束处说明的输入图片格式。 - 输入图片的宽在(4096, 8192]或高在(4096, 8192]范围内时，支持格式如下： 支持acldvppPixelFormat枚举值中的如下枚举项： PIXEL_FORMAT_YUV_400 = 0, // YUV400 8bit PIXEL_FORMAT_YUV_SEMIPLANAR_420 = 1, // YUV420SP NV12 8bit PIXEL_FORMAT_YVU_SEMIPLANAR_420 = 2, // YUV420SP NV21 8bit PIXEL_FORMAT_YUV_SEMIPLANAR_422 = 3, // YUV422SP 8bit PIXEL_FORMAT_YVU_SEMIPLANAR_422 = 4, // YVU422SP 8bit PIXEL_FORMAT_YUV_SEMIPLANAR_444 = 5, // YUV444SP 8bit PIXEL_FORMAT_YVU_SEMIPLANAR_444 = 6, // YVU444SP 8bit PIXEL_FORMAT_YUYV_PACKED_422 = 7, // YUV422Packed YUYV 8bit PIXEL_FORMAT_UYVY_PACKED_422 = 8, // YUV422Packed UYVY 8bit PIXEL_FORMAT_YVYU_PACKED_422 = 9, // YUV422Packed YVYU 8bit PIXEL_FORMAT_VYUY_PACKED_422 = 10, // YUV422Packed VYUY 8bit - 关于输出图片分辨率、图片格式请参见图片格式、宽高对齐、内存约束。 - 关于各图片格式的宽高对齐、内存大小约束，请参见图片格式、宽高对齐、内存约束。 - 当输入、输出图片格式都为ARGB8888、ABGR8888、RGBA8888、BGRA8888格式时，在经过本接口缩放处理后，A分量值会变为0。

版本	约束
Atlas A2训练系列产品/Atlas 800I A2推理产品 Atlas 200/500 A2推理产品	- 关于图片分辨率、图片格式、宽高对齐、内存等总体约束，请参见图片格式、宽高对齐、内存约束。 - 由于YUV格式图像下采样约束，当输出图片格式为YUV420SP或YUV422SP格式时，贴图的顶部偏移、左侧偏移建议为偶数。 - 当输入、输出图片格式都为ARGB8888、ABGR8888、RGBA8888、BGRA8888格式时，在经过本接口缩放处理后，A分量值会变为0。

函数原型

```
aclError acldvppVpcCropResizePasteAsync(aclvppChannelDesc *channelDesc,  
aclvppPicDesc *inputDesc,  
aclvppPicDesc *outputDesc,  
aclvppRoiConfig *cropArea,  
aclvppRoiConfig *pasteArea,  
aclvppResizeConfig *resizeConfig,  
aclrtStream stream)
```

参数说明

参数名	输入/输出	说明
channelDesc	输入	通道描述信息的指针。 与调用 acldvppCreateChannel 接口创建通道时指定的channelDesc保持一致。
inputDesc	输入	输入图片信息的指针。 - 调用acldvppCreatePicDesc接口创建图片描述信息。 - 调用acldvppSetPicDesc系列接口设置图片描述参数（例如，内存地址、内存大小、图片格式、宽、高等）。
outputDesc	输入&输出	输出图片信息的指针。 outputDesc参数作为输入时，需要用户调用如下接口： - 调用acldvppCreatePicDesc接口创建图片描述信息。 - 调用acldvppSetPicDesc系列接口设置图片描述参数（例如，内存地址、内存大小、图片格式、宽、高等）。

参数名	输入/输出	说明
cropArea	输入	抠图区域位置的指针。 调用 aclvppCreateRoiConfig 接口创建区域位置数据。
pasteArea	输入	贴图区域位置的指针。 - 调用aclvppCreateRoiConfig接口创建区域位置数据。 - 贴图区域左偏移需要16对齐。
resizeConfig	输入	图片缩放配置数据的指针。 需提前调用 aclvppCreateResizeConfig 接口创建图片缩放配置数据。
stream	输入	指定Stream。

返回值说明

返回0表示任务下发成功，返回非0表示任务下发失败。

参考资源

接口调用示例，参见抠图缩放（一图一框）示例代码。

1.8.5.11 aclvppVpcBatchCropAndPasteAsync

函数功能

按指定区域从一张或多张输入图片中抠出一个或多个子图，再将抠的子图贴到目标图片的指定位置，作为输出图片。异步接口。

约束说明

版本	约束
Atlas 200/300/500 推理产品 Atlas 训练系列产品	- 关于输入图片分辨率、图片格式： - 输入图片分辨率在[10*6, 4096*4096]范围内时，支持图片格式、宽高对齐、内存约束处说明的输入图片格式。 - 输入图片的宽在(4096, 8192]或高在(4096, 8192]范围内时，支持格式如下： 支持acldvppPixelFormat枚举值中的如下枚举项： PIXEL_FORMAT_YUV_SEMIPLANAR_420 = 1, // YUV420SP NV12 8bit PIXEL_FORMAT_YVU_SEMIPLANAR_420 = 2, // YUV420SP NV21 8bit - 关于输出图片分辨率、图片格式请参见图片格式、宽高对齐、内存约束。 - 关于各图片格式的宽高对齐、内存大小约束，请参见图片格式、宽高对齐、内存约束。 - 当pasteArea的宽高与cropArea的宽高不一致时会对图片做一次缩放操作，默认缩放算法为“自研的高阶滤波算法”。
Atlas 推理系列产品	- 关于输入图片分辨率、图片格式： - 输入图片分辨率在[10*6, 4096*4096]范围内时，支持图片格式、宽高对齐、内存约束处说明的输入图片格式。 - 输入图片的宽在(4096, 8192]或高在(4096, 8192]范围内时，支持格式如下： 支持acldvppPixelFormat枚举值中的如下枚举项： PIXEL_FORMAT_YUV_400 = 0, // YUV400 8bit PIXEL_FORMAT_YUV_SEMIPLANAR_420 = 1, // YUV420SP NV12 8bit PIXEL_FORMAT_YVU_SEMIPLANAR_420 = 2, // YUV420SP NV21 8bit PIXEL_FORMAT_YUV_SEMIPLANAR_422 = 3, // YUV422SP 8bit PIXEL_FORMAT_YVU_SEMIPLANAR_422 = 4, // YVU422SP 8bit PIXEL_FORMAT_YUV_SEMIPLANAR_444 = 5, // YUV444SP 8bit PIXEL_FORMAT_YVU_SEMIPLANAR_444 = 6, // YVU444SP 8bit PIXEL_FORMAT_YUYV_PACKED_422 = 7, // YUV422Packed YUYV 8bit PIXEL_FORMAT_UYVY_PACKED_422 = 8, // YUV422Packed UYVY 8bit PIXEL_FORMAT_VYU_PACKED_422 = 9, // YUV422Packed VYU 8bit PIXEL_FORMAT_VYUY_PACKED_422 = 10, // YUV422Packed VYUY 8bit - 关于输出图片分辨率、图片格式请参见图片格式、宽高对齐、内存约束。 - 关于各图片格式的宽高对齐、内存大小约束，请参见图片格式、宽高对齐、内存约束。 - 当pasteArea的宽高与cropArea的宽高不一致时会对图片做一次缩放操作，默认缩放算法为“业界通用的Bilinear算法”。

版本	约束
Atlas A2训练系列产品/Atlas 800I A2推理产品 Atlas 200/500 A2推理产品	- 关于图片分辨率、图片格式、宽高对齐、内存等总体约束，请参见图片格式、宽高对齐、内存约束。 - 当pasteArea的宽高与cropArea的宽高不一致时会对图片做一次缩放操作，默认缩放算法为“业界通用的Bilinear算法”。 - 由于YUV格式图像下采样约束，当输出图片格式为YUV420SP或YUV422SP格式时，贴图的顶部偏移、左侧偏移建议为偶数。

函数原型

```
aclError acldvppVpcBatchCropAndPasteAsync(aclvppChannelDesc
*channelDesc,
aclvppBatchPicDesc *srcBatchPicDescs,
uint32_t *roiNums,
uint32_t size,
aclvppBatchPicDesc *dstBatchPicDescs,
aclvppRoiConfig *cropAreas[],
aclvppRoiConfig *pasteAreas[],
aclrtStream stream)
```

参数说明

参数名	输入/输出	说明
channelDesc	输入	通道描述信息的指针。 与调用 acldvppCreateChannel 接口创建通道时指定的channelDesc保持一致。
srcBatchPicDescs	输入	指定批量输入图片描述信息的指针。 - 调用acldvppCreateBatchPicDesc接口创建批量图片描述信息。 - 调用acldvppGetPicDesc接口获取指定图片的图片描述信息。 - 调用acldvppSetPicDesc系列接口设置图片描述参数（例如，内存地址、内存大小、图片格式、宽、高等）。

参数名	输入/输出	说明
roiNums	输入	每个输入图片抠图数量的指针，当前最大抠图数量为256。 roiNums为数组，数组总和小于等于256，与dstBatchPicDescs结构体中的batchSize值保持一致。 roiNums[0]+...+roiNums[size-1] <= 256
size	输入	表示roiNums数组中的元素个数，个数小于等于256。
dstBatchPicDescs	输入&输出	批量输出图片描述信息的指针。 dstBatchPicDescs参数作为输入时，需要用户调用如下接口： - 调用aclDvppCreateBatchPicDesc接口创建批量图片描述信息。 - 调用aclDvppGetPicDesc接口获取指定图片的图片描述信息。 - 调用aclDvppSetPicDesc系列接口设置图片描述参数（例如，内存地址、内存大小、图片格式、宽、高等）。
cropAreas	输入	抠图区域位置的指针数组。 - 调用aclDvppCreateRoiConfig接口创建区域位置数据。 - cropAreas、pasteAreas数组中的元素个数与dstBatchPicDescs结构体中的batchSize值相等。
pasteAreas	输入	贴图区域位置的指针数组。 - 调用aclDvppCreateRoiConfig接口创建区域位置数据。 - 贴图区域左偏移需要16对齐。 - cropAreas、pasteAreas数组中的元素个数与dstBatchPicDescs参数中的batchSize值相等。
stream	输入	指定Stream。

返回值说明

返回0表示任务下发成功，返回非0表示任务下发失败。

参考资源

接口调用示例，参见抠图贴图（一图多框）示例代码。

1.8.5.12 acldvppVpcBatchCropResizePasteAsync

函数功能

按指定区域从一张或多张输入图片中抠出一个或多个子图，再将抠的子图贴到目标图片的指定位置，作为输出图片。异步接口。

当pasteArea的宽高与cropArea的宽高不一致时会对图片做一次缩放操作，按照resizeConfig处配置的缩放算法。

约束说明

版本	约束
Atlas 200/300/500 推理产品 Atlas 训练系列产品	- 关于输入图片分辨率、图片格式： - 输入图片分辨率在[10*6, 4096*4096]范围内时，支持图片格式、宽高对齐、内存约束处说明的输入图片格式。 - 输入图片的宽在(4096, 8192]或高在(4096, 8192]范围内时，支持格式如下： 支持acldvppPixelFormat枚举值中的如下枚举项： PIXEL_FORMAT_YUV_SEMIPLANAR_420 = 1, // YUV420SP NV12 8bit PIXEL_FORMAT_YVU_SEMIPLANAR_420 = 2, // YUV420SP NV21 8bit - 关于输出图片分辨率、图片格式请参见图片格式、宽高对齐、内存约束。 - 关于各图片格式的宽高对齐、内存大小约束，请参见图片格式、宽高对齐、内存约束。

版本	约束
Atlas 推理系列产品	- 关于输入图片分辨率、图片格式： - 输入图片分辨率在[10*6, 4096*4096]范围内时，支持图片格式、宽高对齐、内存约束处说明的输入图片格式。 - 输入图片的宽在(4096, 8192]或高在(4096, 8192]范围内时，支持格式如下： 支持acldvppPixelFormat枚举值中的如下枚举项： PIXEL_FORMAT_YUV_400 = 0, // YUV400 8bit PIXEL_FORMAT_YUV_SEMIPLANAR_420 = 1, // YUV420SP NV12 8bit PIXEL_FORMAT_YVU_SEMIPLANAR_420 = 2, // YUV420SP NV21 8bit PIXEL_FORMAT_YUV_SEMIPLANAR_422 = 3, // YUV422SP 8bit PIXEL_FORMAT_YVU_SEMIPLANAR_422 = 4, // YVU422SP 8bit PIXEL_FORMAT_YUV_SEMIPLANAR_444 = 5, // YUV444SP 8bit PIXEL_FORMAT_YVU_SEMIPLANAR_444 = 6, // YVU444SP 8bit PIXEL_FORMAT_YUYV_PACKED_422 = 7, // YUV422Packed YUYV 8bit PIXEL_FORMAT_UYVY_PACKED_422 = 8, // YUV422Packed UYVY 8bit PIXEL_FORMAT_VYVU_PACKED_422 = 9, // YUV422Packed VYVU 8bit PIXEL_FORMAT_VYUY_PACKED_422 = 10, // YUV422Packed VYUY 8bit - 关于输出图片分辨率、图片格式请参见图片格式、宽高对齐、内存约束。 - 关于各图片格式的宽高对齐、内存大小约束，请参见图片格式、宽高对齐、内存约束。 - 当输入、输出图片格式都为ARGB8888、ABGR8888、RGBA8888、BGRA8888格式时，在经过本接口缩放处理后，A分量值会变为0。
Atlas A2训练系列产品/Atlas 800I A2推理产品 Atlas 200/500 A2推理产品	- 关于图片分辨率、图片格式、宽高对齐、内存等总体约束，请参见图片格式、宽高对齐、内存约束。 - 由于YUV格式图像下采样约束，当输出图片格式为YUV420SP或YUV422SP格式时，贴图的顶部偏移、左侧偏移建议为偶数。 - 当输入、输出图片格式都为ARGB8888、ABGR8888、RGBA8888、BGRA8888格式时，在经过本接口缩放处理后，A分量值会变为0。

函数原型

```
aclError acldvppVpcBatchCropResizePasteAsync(acldvppChannelDesc
*channelDesc,
acldvppBatchPicDesc *srcBatchPicDescs,
uint32_t *roiNums,
uint32_t size,
acldvppBatchPicDesc *dstBatchPicDescs,
```

```
acldvppRoiConfig *cropAreas[],  
acldvppRoiConfig *pasteAreas[],  
acldvppResizeConfig *resizeConfig,  
aclrtStream stream)
```

参数说明

参数名	输入/输出	说明
channelDesc	输入	通道描述信息的指针。 与调用 acldvppCreateChannel 接口创建通道时指定的channelDesc保持一致。
srcBatchPicDescs	输入	批量输入图片描述信息的指针。 - 调用acldvppCreateBatchPicDesc接口创建批量图片描述信息。 - 调用acldvppGetPicDesc接口获取指定图片的图片描述信息。 - 调用acldvppSetPicDesc系列接口设置图片描述参数（例如，内存地址、内存大小、图片格式、宽、高等）。
roiNums	输入	每个输入图片抠图数量的指针，当前最大抠图数量为256。 roiNums为数组，数组总和小于等于256，与dstBatchPicDescs结构体中的batchSize值保持一致。 roiNums[0]+...+roiNums[size-1] <= 256
size	输入	表示roiNums数组中的元素个数，个数小于等于256。
dstBatchPicDescs	输入&输出	批量输出图片描述信息的指针。 dstBatchPicDescs参数作为输入时，需要用户调用如下接口： - 调用acldvppCreateBatchPicDesc接口创建批量图片描述信息。 - 调用acldvppGetPicDesc接口获取指定图片的图片描述信息。 - 调用acldvppSetPicDesc系列接口设置图片描述参数（例如，内存地址、内存大小、图片格式、宽、高等）。
cropAreas	输入	抠图区域位置的指针数组。 - 调用acldvppCreateRoiConfig接口创建区域位置数据。 - cropAreas、pasteAreas数组中的元素个数与dstBatchPicDescs结构体中的batchSize值相等。

参数名	输入/输出	说明
pasteAreas	输入	贴图区域位置的指针数组。 - 调用acldvppCreateRoiConfig接口创建区域位置数据。 - 贴图区域左偏移需要16对齐。 - cropAreas、pasteAreas数组中的元素个数与dstBatchPicDescs参数中的batchSize值相等。
resizeConfig	输入	图片缩放配置数据的指针。 需提前调用 acldvppCreateResizeConfig 接口创建图片缩放配置数据。
stream	输入	指定Stream。

返回值说明

返回0表示任务下发成功，返回非0表示任务下发失败。

1.8.5.13 acldvppVpcBatchCropResizeMakeBorderAsync

Atlas 200/300/500 推理产品，不支持该接口。
Atlas 训练系列产品，不支持该接口。

函数功能

支持从一张或多张输入图片中抠出一张或多张子图，对每个子图进行缩放，扩边填充后输出至目标图像内存地址。异步接口。

约束说明

版本	约束
Atlas 推理系列产品	- 关于输入图片分辨率、图片格式： - 输入图片分辨率在[10*6, 4096*4096]范围内时，支持图片格式、宽高对齐、内存约束处说明的输入图片格式。 - 输入图片的宽在(4096, 8192]或高在(4096, 8192]范围内时，支持格式如下： 支持aclDvppPixelFormat枚举值中的如下枚举项：<pre>PIXEL_FORMAT_YUV_400 = 0, // YUV400 8bit PIXEL_FORMAT_YUV_SEMIPLANAR_420 = 1, // YUV420SP NV12 8bit PIXEL_FORMAT_YVU_SEMIPLANAR_420 = 2, // YUV420SP NV21 8bit PIXEL_FORMAT_YUV_SEMIPLANAR_422 = 3, // YUV422SP 8bit PIXEL_FORMAT_YVU_SEMIPLANAR_422 = 4, // YVU422SP 8bit PIXEL_FORMAT_YUV_SEMIPLANAR_444 = 5, // YUV444SP 8bit PIXEL_FORMAT_YVU_SEMIPLANAR_444 = 6, // YVU444SP 8bit PIXEL_FORMAT_YUYV_PACKED_422 = 7, // YUV422Packed YUYV 8bit PIXEL_FORMAT_UYVY_PACKED_422 = 8, // YUV422Packed UYVY 8bit PIXEL_FORMAT_YVYU_PACKED_422 = 9, // YUV422Packed YVYU 8bit PIXEL_FORMAT_VYUY_PACKED_422 = 10, // YUV422Packed VYUY 8bit</pre> - 关于输出图片分辨率、图片格式请参见图片格式、宽高对齐、内存约束。 - 关于各图片格式的宽高对齐、内存大小约束，请参见图片格式、宽高对齐、内存约束。 - 当输入、输出图片格式都为ARGB8888、ABGR8888、RGBA8888、BGRA8888格式时，在经过本接口缩放处理后，A分量值会变为0。 - 使用本接口实现等比例缩放功能时，相关的描述及约束请参见抠图、贴图约束。 - 由于YUV格式图像下采样约束，当输出图片格式为YUV420SP或YUV422SP格式时，需注意： - 对于YUV420SP输出格式，上下左右填充的尺寸建议为偶数； - 对于YUV422SP输出格式，左右填充的尺寸建议为偶数。

版本	约束
Atlas A2训练系列产品/Atlas 800I A2推理产品 Atlas 200/500 A2推理产品	- 关于图片分辨率、图片格式、宽高对齐、内存等总体约束，请参见图片格式、宽高对齐、内存约束。 - 当输入、输出图片格式都为ARGB8888、ABGR8888、RGBA8888、BGRA8888格式时，在经过本接口缩放处理后，A分量值会变为0。 - 使用本接口实现等比例缩放功能时，相关的描述及约束请参见抠图、贴图约束。 - 由于YUV格式图像下采样约束，当输出图片格式为YUV420SP或YUV422SP格式时，需注意： - 对于YUV420SP输出格式，上下左右填充的尺寸建议为偶数； - 对于YUV422SP输出格式，左右填充的尺寸建议为偶数。

函数原型

```
aclError acldvppVpcBatchCropResizeMakeBorderAsync(acldvppChannelDesc
*channelDesc,
acldvppBatchPicDesc *srcBatchPicDescs,
uint32_t *roiNums,
uint32_t size,
acldvppBatchPicDesc *dstBatchPicDescs,
acldvppRoiConfig *cropAreas[],
acldvppBorderConfig *borderCfgs[],
acldvppResizeConfig *resizeConfig,
aclrtStream stream)
```

参数说明

参数名	输入/输出	说明
channelDesc	输入	通道描述信息的指针。 与调用 acldvppCreateChannel 接口创建通道时指定的channelDesc保持一致。

参数名	输入/输出	说明
srcBatchPicDescs	输入	批量输入图片描述信息的指针。 - 调用aclvppCreateBatchPicDesc接口创建批量图片描述信息。 - 调用aclvppGetPicDesc接口获取指定图片的图片描述信息。 - 调用aclvppSetPicDesc系列接口设置图片描述参数（例如，内存地址、内存大小、图片格式、宽、高等）。
roiNums	输入	每个输入图片抠图数量的指针，当前最大抠图数量为256。 roiNums为数组，数组总和小于等于256，与dstBatchPicDescs结构体中的batchSize值保持一致。 $\text{roiNums}[0] + \dots + \text{roiNums}[\text{size}-1] \leq 256$
size	输入	表示roiNums数组中的元素个数，个数小于等于256。
dstBatchPicDescs	输入&输出	批量输出图片描述信息的指针。 dstBatchPicDescs参数作为输入时，需要用户调用如下接口： - 调用aclvppCreateBatchPicDesc接口创建批量图片描述信息。 - 调用aclvppGetPicDesc接口获取指定图片的图片描述信息。 - 调用aclvppSetPicDesc系列接口设置图片描述参数（例如，内存地址、内存大小、图片格式、宽、高等）。
cropAreas	输入	抠图区域位置的指针数组。 需提前调用 aclvppCreateRoiConfig 接口创建区域位置数据。 cropAreas、borderCfgs数组中的元素个数与dstBatchPicDescs结构体中的batchSize值相等。
borderCfgs	输入	扩边填充配置的指针数组。 borderCfgs数组中的元素与cropAreas数组中的元素一一对应。 扩边填充配置参数中的borderType参数值仅支持BORDER_CONSTANT和BORDER_REPLICATE。
resizeConfig	输入	图片缩放配置数据的指针。 需提前调用 aclvppCreateResizeConfig 接口创建图片缩放配置数据。 VPC进行缩放处理时，缩放后的宽/高是输出图片的宽/高减去扩边的值。

参数名	输入/输出	说明
stream	输入	指定Stream。

返回值说明

返回0表示任务下发成功，返回非0表示任务下发失败。

1.8.5.14 aclvppVpcConvertColorAsync

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

函数功能

转换图片的格式，输出图片的宽高须与输入图片的宽高一致。异步接口。

约束说明

- 输入、输出图片分辨率为[10*6, 4096*4096]。
- 输入、输出图片的格式请参见[图片格式](#)、[宽高对齐](#)、[内存约束](#)。
- Atlas 推理系列产品上，当输入格式是YUV400时，输出格式只能是YUV400。

函数原型

```
aclError aclvppVpcConvertColorAsync(aclvppChannelDesc *channelDesc,  
aclvppPicDesc *inputDesc,  
aclvppPicDesc *outputDesc,  
aclrtStream stream)
```

参数说明

参数名	输入/输出	说明
channelDesc	输入	通道描述信息的指针。 与调用 aclvppCreateChannel 接口创建通道时指定的channelDesc保持一致。
inputDesc	输入	输入图片信息的指针。 • 调用aclvppCreatePicDesc接口创建图片描述信息。 • 调用aclvppSetPicDesc系列接口设置图片描述参数（例如，内存地址、内存大小、图片格式、宽、高等）。

参数名	输入/输出	说明
outputDesc	输入&输出	输出图片信息的指针。 outputDesc参数作为输入时，需要用户调用如下接口： - 调用aclDvppCreatePicDesc接口创建图片描述信息。 - 调用aclDvppSetPicDesc系列接口设置图片描述参数（例如，内存地址、内存大小、图片格式、宽、高等）。 需指定输出图片的宽、高、宽stride、高stride、内存地址、内存大小、格式等。 - 配置输出图片的宽、高时，需与输入图片的宽、高保持一致，否则会返回报错。 如果将输出图片的宽或高配置为0，则VPC内部会将输入图片的宽、高分别作为输出图片的宽、高；同时，VPC内部会按对齐要求计算宽stride、高stride，不同图片格式的对齐要求不同，请参见图片格式、宽高对齐、内存约束。 - 配置输出图片的宽stride、高stride时，满足对齐要求即可，不同图片格式的对齐要求不同，请参见图片格式、宽高对齐、内存约束。
stream	输入	指定Stream。

返回值说明

返回0表示任务下发成功，返回非0表示任务下发失败。

参考资源

接口调用示例，参见格式转换示例代码。

1.8.5.15 aclDvppVpcPyrDownAsync

- Atlas 200/300/500 推理产品，不支持该接口。
- Atlas 训练系列产品，不支持该接口。
- Atlas 200/500 A2推理产品，不支持该接口。
- Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

对图像进行金字塔缩放，将输出图片缩放为输入图片的一半。异步接口。

约束说明

- 关于输入图片

- 输入图片分辨率: [20*12, 2048*2048]
- 输入图片格式: PIXEL_FORMAT_YUV_400
- 关于输出图片
 - 输出图像分辨率: 输入图像分辨率/2
 - 图像格式: PIXEL_FORMAT_YUV_400
- PIXEL_FORMAT_YUV_400图片格式的对齐要求请参见[图片格式、宽高对齐、内存约束](#)

函数原型

```
aclError aclvppVpcPyrDownAsync(aclvppChannelDesc *channelDesc,  
aclvppPicDesc *inputDesc,  
aclvppPicDesc *outputDesc,  
void *reserve,  
aclrtStream stream)
```

参数说明

参数名	输入/输出	说明
channelDesc	输入	通道描述信息的指针。 与调用 aclvppCreateChannel 接口创建通道时指定的channelDesc保持一致。
inputDesc	输入	输入图片信息的指针。 • 调用aclvppCreatePicDesc接口创建图片描述信息。 • 调用aclvppSetPicDesc系列接口设置图片描述参数（例如，内存地址、内存大小、图片格式、宽、高等）。
outputDesc	输入&输出	输出图片信息的指针。 outputDesc参数作为输入时，需要用户调用如下接口： • 调用aclvppCreatePicDesc接口创建图片描述信息。 • 调用aclvppSetPicDesc系列接口设置图片描述参数（例如，内存地址、内存大小、图片格式、宽、高等）。
reserve	输入	金字塔缩放配置参数的指针，预留参数。
stream	输入	指定Stream。

返回值说明

返回0表示任务下发成功，返回非0表示任务下发失败。

1.8.5.16 aclDvppVpcEqualizeHistAsync

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

函数功能

实现图片的色彩重映射。异步接口。

约束说明

- 输入图片分辨率为[10*6, 4096*4096]。
- 输入图片格式支持[aclDvppPixelFormat](#)枚举值中的如下枚举项：

```
PIXEL_FORMAT_YUV_400 = 0,           // YUV400 8bit
PIXEL_FORMAT_YUV_SEMIPLANAR_420 = 1, // YUV420SP NV12 8bit
PIXEL_FORMAT_YVU_SEMIPLANAR_420 = 2, // YUV420SP NV21 8bit
PIXEL_FORMAT_YUV_SEMIPLANAR_422 = 3, // YUV422SP 8bit
PIXEL_FORMAT_YVU_SEMIPLANAR_422 = 4, // YVU422SP 8bit
PIXEL_FORMAT_YUV_PACKED_444 = 11,    // YUV444P 8bit
PIXEL_FORMAT_RGB_888 = 12,           // RGB888
PIXEL_FORMAT_BGR_888 = 13,           // BGR888
```
- 输出图片分辨率、输出图片格式、widthStride对齐要求、heightStride对齐要求等，必须与输入图片格的要求保持一致。
- 不同图片格式的对齐要求请参见[图片格式](#)、[宽高对齐](#)、[内存约束](#)。

函数原型

```
aclError aclDvppVpcEqualizeHistAsync(const aclDvppChannelDesc
*channelDesc,

const aclDvppPicDesc *inputDesc,

aclDvppPicDesc *outputDesc,

const aclDvppLutMap *lutMap,

aclrtStream stream)
```

参数说明

参数名	输入/输出	说明
channelDesc	输入	通道描述信息的指针。 与调用 aclDvppCreateChannel 接口创建通道时指定的channelDesc保持一致。
inputDesc	输入	输入图片信息的指针。 - 调用aclDvppCreatePicDesc接口创建图片描述信息。 - 调用aclDvppSetPicDesc系列接口设置图片描述参数（例如，内存地址、内存大小、图片格式、宽、高等）。

参数名	输入/输出	说明
outputDesc	输入&输出	输出图片信息的指针。 outputDesc参数作为输入时，需要用户调用如下接口： - 调用acldvppCreatePicDesc接口创建图片描述信息。 - 调用acldvppSetPicDesc系列接口设置图片描述参数（例如，内存地址、内存大小、图片格式、宽、高等）。
lutMap	输入	色彩重映射表的指针。 调用 acldvppCreateLutMap 接口创建acldvppLutMap类型的数据（表示色彩重映射表），再调用 acldvppGetLutMapData 获取内存指针，由用户自行向该内存中写入色彩重映射表数据，写完数据后，再将acldvppLutMap类型的数据作为入参传入acldvppVpcEqualizeHistAsync。
stream	输入	指定Stream。

返回值说明

返回0表示任务下发成功，返回非0表示任务下发失败。

1.8.5.17 acldvppVpcMakeBorderAsync

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

函数功能

实现图片边界填充功能。异步接口。

约束说明

- 输入图片分辨率为[10*6, 4096*4096]，输入图片格式、对齐、内存等约束，请参见[图片格式、宽高对齐、内存约束](#)。
- 输出图片分辨率、图片格式、对齐、内存等约束，请参见[1.8.5.2 约束说明](#)。
- 由于[YUV格式图像下采样约束](#)，当输出图片格式为YUV420SP或YUV422SP格式时，需注意：
 - 对于YUV420SP输出格式，上下左右填充的尺寸建议为偶数；
 - 对于YUV422SP输出格式，左右填充的尺寸建议为偶数。

函数原型

aclError acldvppVpcMakeBorderAsync(const **acldvppChannelDesc** *channelDesc,

```
const acldvppPicDesc *inputDesc,  
acldvppPicDesc *outputDesc,  
const acldvppBorderConfig *borderConfig,  
aclrtStream stream)
```

参数说明

参数名	输入/输出	说明
channelDesc	输入	通道描述信息的指针。 与调用 acldvppCreateChannel 接口创建通道时指定的channelDesc保持一致。
inputDesc	输入	输入图片信息的指针。 - 调用acldvppCreatePicDesc接口创建图片描述信息。 - 调用acldvppSetPicDesc系列接口设置图片描述参数（例如，内存地址、内存大小、图片格式、宽、高等）。
outputDesc	输入&输出	输出图片信息的指针。 outputDesc参数作为输入时，需要用户调用如下接口： - 调用acldvppCreatePicDesc接口创建图片描述信息。 - 调用acldvppSetPicDesc系列接口设置图片描述参数（例如，内存地址、内存大小、图片格式、宽、高等）。
borderConfig	输入	填充边界配置的指针。 调用 acldvppCreateBorderConfig 接口创建填充边界配置。
stream	输入	指定Stream。

返回值说明

返回0表示任务下发成功，返回非0表示任务下发失败。

1.8.5.18 acldvppVpcCalcHistAsync

Atlas 200/300/500 推理产品，不支持该接口。
Atlas 训练系列产品，不支持该接口。

函数功能

统计图像中每个像素的各颜色分量的分布。异步接口。

约束说明

- 关于输入图片分辨率的要求：

版本	约束
Atlas 推理系列产品	输入图片分辨率：[10*6, 4096*4096]。
Atlas A2训练系列产品/Atlas 800I A2推理产品 Atlas 200/500 A2推理产品	输入图片分辨率：[10*6, 4096*8192]。

- 输入图片格式支持[aclDvppPixelFormat](#)枚举值中的如下枚举项：

```
PIXEL_FORMAT_YUV_400 = 0,           // YUV400 8bit
PIXEL_FORMAT_YUV_SEMIPLANAR_420 = 1, // YUV420SP NV12 8bit
PIXEL_FORMAT_YVU_SEMIPLANAR_420 = 2, // YUV420SP NV21 8bit
PIXEL_FORMAT_YUV_SEMIPLANAR_422 = 3, // YUV422SP 8bit
PIXEL_FORMAT_YVU_SEMIPLANAR_422 = 4, // YVU422SP 8bit
PIXEL_FORMAT_YUV_SEMIPLANAR_444 = 5, // YUV444SP 8bit
PIXEL_FORMAT_YVU_SEMIPLANAR_444 = 6, // YVU444SP 8bit
PIXEL_FORMAT_YUYV_PACKED_422 = 7,    // YUV422Packed YUYV 8bit
PIXEL_FORMAT_UYVY_PACKED_422 = 8,    // YUV422Packed UYVY 8bit
PIXEL_FORMAT_YVYU_PACKED_422 = 9,    // YUV422Packed YVYU 8bit
PIXEL_FORMAT_VYUY_PACKED_422 = 10,   // YUV422Packed VYUY 8bit
PIXEL_FORMAT_YUV_PACKED_444 = 11,    // YUV444P 8bit
PIXEL_FORMAT_RGB_888 = 12,           // RGB888
PIXEL_FORMAT_BGR_888 = 13,           // BGR888
PIXEL_FORMAT_YUV_SEMIPLANAR_440 = 1000, // YUV440SP 8bit
PIXEL_FORMAT_YVU_SEMIPLANAR_440 = 1001, // YVU440SP 8bit
```

- 关于图片格式、宽高对齐、内存等总体约束，请参见[图片格式、宽高对齐、内存约束](#)。

函数原型

```
aclError aclDvppVpcCalcHistAsync(aclDvppChannelDesc *channelDesc,
aclDvppPicDesc *srcPicDesc,
aclDvppHist *hist,
void *reserve,
aclrtStream stream)
```

参数说明

参数名	输入/输出	说明
channelDesc	输入	通道描述信息的指针。 与调用 aclDvppCreateChannel 接口创建通道时指定的channelDesc保持一致。

参数名	输入/输出	说明
srcPicDesc	输入	输入图片信息的指针。 - 调用aclDvppCreatePicDesc接口创建图片描述信息。 - 调用aclDvppSetPicDesc系列接口设置图片描述参数（例如，内存地址、内存大小、图片格式、宽、高等）。 说明 当图片格式为YUV440SP 8bit、YVU440SP 8bit时，Y分量统计准确，UV分量统计上存在误差。
hist	输入&输出	直方图描述信息的指针。 调用 aclDvppCreateHist 接口创建直方图描述信息作为输入。 调用本接口后，输出每张图像各颜色分量的像素值（0-255）分布情况。
reserve	输入	直方图统计配置的指针，预留，当前可填NULL。
stream	输入	指定Stream。

返回值说明

返回0表示任务下发成功，返回非0表示任务下发失败。

1.8.6 JPEGD 功能

1.8.6.1 功能及约束说明

功能说明

JPEGD（JPEG Decoder）实现.jpg、.jpeg、.JPG、.JPEG图片文件的解码。

- JPEGD在解码图片时，支持对图片进行旋转。**
如果输入图片的码流中包含Orientation信息（代表捕获图像时摄像机相对于场景的方向），则JPEGD在解码时会解析Orientation信息，将图片进行90度、180度、270度或镜像旋转。旋转后输出图片的宽stride、高stride、输出内存仍需满足[图片格式、宽高对齐、内存约束](#)中的要求。
如果输入图片的码流异常，导致JPEGD解码时无法读取Orientation信息，则不能实现图片旋转的功能。
针对以下产品版本，旋转图片时存在约束：

产品版本	约束
Atlas 200/300/500 推理产品	多路并发解码JPEG图片场景下，如果解码的JPEG图片中，包含带旋转信息的大分辨率图片，例如3840*2160分辨率及以上的图片，则可能导致图片解码时间过长，从而导致用户进程超时退出。
Atlas 200/500 A2推理产品 Atlas A2训练系列产品/Atlas 800I A2推理产品	● JPEGD 422格式旋转图片，若原图高为奇数，并且旋转方式为宽高对换，则旋转后宽存在黑边，建议用户将宽向下对齐到偶数使用（即去除黑边），比如原图为200*101，若旋转后为101*200，则建议用户实际使用区域为100*200。 ● JPEGD 440格式旋转图片，若原图宽为奇数，并且旋转方式为宽高对换，则旋转后高存在黑边，建议用户将高向下对齐到偶数使用（即去除黑边），比如原图为201*101，若旋转后为100*201，则建议用户实际使用区域为100*200。

● **JPEGD在解码图片时，支持按源图片格式解码。**

源图片格式解码是指解码前后图片的编码格式保持一致，例如解码前输入图片为jpeg(440)，解码后输出图片为YUV440SP V在前U在后或YUV440SP U在前V在后。

使用源图片格式解码，有以下方式：

- 调用[aclDvppJpegGetImageInfoV2](#)接口，获取JPEGD解码前输入图片的编码格式，在调用JPEGD解码接口时，将输出图片格式设置的与输入图片编码格式一致。
- 在调用JPEGD解码接口时，将输出图片格式配置为PIXEL_FORMAT_UNKNOWN，输出格式默认按源图片格式输出、且是V在前U在后的Semi-Planar格式。例如，JPEG输入图片编码格式为jpeg(440)，输出图片格式配置为PIXEL_FORMAT_UNKNOWN，JPEGD解码后，实际输出图片格式为YUV440SP V在前U在后。

 说明

JPEGD解码后的输出图片，如果要直接作为模型推理的输入，建议将输出图片格式配置为PIXEL_FORMAT_UNKNOWN，这时JPEGD使用源图片格式解码（但这里要确保解码后的图片格式模型是支持的），保证模型推理的精度。

JPEGD解码后的输出图片，如果直接作为VPC的输入，该场景下若使用源图片格式解码时，则需要关注解码后的输出图片格式VPC是否支持（VPC输入图片的格式请参见[1.8.5.2 约束说明](#)），如果VPC不支持，则用户需按VPC支持的情况指定JPEGD的输出图片格式。

图片分辨率约束

● **输入图片分辨率**

版本	分辨率范围
Atlas 200/300/500 推理产品 Atlas 训练系列产品 Atlas 推理系列产品	最大分辨率：8192*8192，最小分辨率：32*32。

版本	分辨率范围
Atlas A2训练系列产品/Atlas 800I A2推理产品 Atlas 200/500 A2推理产品	最大分辨率：16384*16384，最小分辨率：32*32。

- 输出图片分辨率
JPEGD只对图片解码，不会改变图片分辨率，因此输出与输入的图片分辨率保持一致。

图片格式、宽高对齐、内存约束

实现JPEGD图片解码功能时，需调用[aclDvppMalloc](#)接口申请Device上的输入、输出内存，调用[aclDvppFree](#)接口释放输入、输出内存，这部分内存的生命周期由用户自行管理。

- 输入内存的大小就是指实际的输入图片所占用的大小。
- 输出内存的大小，可调用[aclDvppJpegPredictDecSize](#)接口预估，计算公式参见下表。

实现JPEGD图片解码功能时，仅支持Huffman编码，压缩前的原图像色彩空间为YUV，像素的各分量比例为4:4:4或4:2:2或4:2:0或4:0:0或4:4:0，不支持算术编码、不支持渐进JPEG格式、不支持JPEG2000格式。

说明

输出图片格式的定义请参见[aclDvppPixelFormat](#)，宽stride、高stride等概念请参见[1.8.2 基本概念](#)。

表 1-18 图片格式、宽高对齐、内存大小约束

输入图片格式 (YUV分量比例)	输出图片格式	输出图片宽、高对齐要求	输出图片宽stride、高stride、内存大小要求 Atlas 200/300/500推理产品 Atlas 训练系列产品	输出图片宽stride、高stride、内存大小要求 Atlas 推理系列产品 Atlas 200/500 A2推理产品 Atlas A2训练系列产品/Atlas 800I A2推理产品
jpeg(444)	YVU444SP 8bit	无对齐要求	宽stride为宽128对齐后的值。 高stride为高16对齐后的值。 内存大小 (单位 Byte) $\geq$ 宽stride * 高stride * 3	宽stride为宽64对齐后的值。 高stride为高16对齐后的值。 内存大小 (单位 Byte) $\geq$ 宽stride * 高stride * 3
	YUV444SP 8bit			

输入图 片格式 (YUV 分量比 例)	输出图片格式	输出图 片宽、 高对齐 要求	输出图片宽stride、 高stride、内存大小 要求 Atlas 200/300/500 推理产品 Atlas 训练系列产品	输出图片宽stride、 高stride、内存大小 要求 Atlas 推理系列产品 Atlas 200/500 A2推 理产品 Atlas A2训练系列产 品/Atlas 800I A2推 理产品
	YUV420SP NV12 8bit	宽2对齐 高2对齐	宽stride为宽128对齐 后的值。 高stride为高16对齐 后的值。 内存大小 (单位 Byte) $\geq$ 宽stride * 高 stride * 3/2	宽stride为宽64对齐后 的值。 高stride为高16对齐后 的值。 内存大小 (单位 Byte) $\geq$ 宽stride * 高 stride * 3/2
	YUV420SP NV21 8bit			
jpeg(42 2)	YUV422SP 8bit	宽2对齐 高2对齐	宽stride为宽128对齐 后的值。 高stride为高16对齐 后的值。 内存大小 (单位 Byte) $\geq$ 宽stride * 高 stride * 2	宽stride为宽64对齐后 的值。 高stride为高16对齐后 的值。 内存大小 (单位 Byte) $\geq$ 宽stride * 高 stride * 2
	YUV422SP 8bit			
	YUV420SP NV12 8bit	宽2对齐 高2对齐	宽stride为宽128对齐 后的值。 高stride为高16对齐 后的值。 内存大小 (单位 Byte) $\geq$ 宽stride * 高 stride * 3/2	宽stride为宽64对齐后 的值。 高stride为高16对齐后 的值。 内存大小 (单位 Byte) $\geq$ 宽stride * 高 stride * 3/2
	YUV420SP NV21 8bit			
jpeg(42 0)	YUV420SP NV12 8bit	宽2对齐 高2对齐	宽stride为宽128对齐 后的值。 高stride为高16对齐 后的值。 内存大小 (单位 Byte) $\geq$ 宽stride * 高 stride * 3/2	宽stride为宽64对齐后 的值。 高stride为高16对齐后 的值。 内存大小 (单位 Byte) $\geq$ 宽stride * 高 stride * 3/2
	YUV420SP NV21 8bit			
jpeg(40 0)	YUV420SP NV12 8bit	宽2对齐 高2对齐	宽stride为宽128对齐 后的值。 高stride为高16对齐 后的值。 内存大小 (单位 Byte) $\geq$ 宽stride * 高 stride * 3/2	宽stride为宽64对齐后 的值。 高stride为高16对齐后 的值。 内存大小 (单位 Byte) $\geq$ 宽stride * 高 stride * 3/2
	YUV420SP NV21 8bit			

输入图片格式 (YUV 分量比例)	输出图片格式	输出图片宽、高对齐要求	输出图片宽stride、高stride、内存大小要求 Atlas 200/300/500 推理产品 Atlas 训练系列产品	输出图片宽stride、高stride、内存大小要求 Atlas 推理系列产品 Atlas 200/500 A2推理产品 Atlas A2训练系列产品/Atlas 800I A2推理产品
	YUV400 8bit Atlas 200/300/500 推理产品和 Atlas 训练系列产品不支持该格式	无对齐要求	宽stride为宽128对齐后的值。 高stride为高16对齐后的值。 内存大小 (单位 Byte) $\geq$ 宽stride * 高stride * 2	宽stride为宽64对齐后的值。 高stride为高16对齐后的值。 内存大小 (单位 Byte) $\geq$ 宽stride * 高stride
jpeg(440)	YVU440SP 8bit	宽2对齐 高2对齐	宽stride为宽128对齐后的值。 高stride为高16对齐后的值。 内存大小 (单位 Byte) $\geq$ 宽stride * 高stride * 2	宽stride为宽64对齐后的值。 高stride为高16对齐后的值。 内存大小 (单位 Byte) $\geq$ 宽stride * 高stride * 2
	YUV440SP 8bit		宽stride为宽128对齐后的值。 高stride为高16对齐后的值。 内存大小 (单位 Byte) $\geq$ 宽stride * 高stride * 2	宽stride为宽64对齐后的值。 高stride为高16对齐后的值。 内存大小 (单位 Byte) $\geq$ 宽stride * 高stride * 2
	YUV420SP NV12 8bit	宽2对齐 高2对齐	宽stride为宽128对齐后的值。 高stride为高16对齐后的值。 内存大小 (单位 Byte) $\geq$ 宽stride * 高stride * 3/2	宽stride为宽64对齐后的值。 高stride为高16对齐后的值。 内存大小 (单位 Byte) $\geq$ 宽stride * 高stride * 3/2
	YUV420SP NV21 8bit		宽stride为宽128对齐后的值。 高stride为高16对齐后的值。 内存大小 (单位 Byte) $\geq$ 宽stride * 高stride * 3/2	宽stride为宽64对齐后的值。 高stride为高16对齐后的值。 内存大小 (单位 Byte) $\geq$ 宽stride * 高stride * 3/2

软、硬件约束

- 硬件约束：
 - 最多支持4张Huffman表，其中包括2 张DC（直流）表和2 张AC（交流）表。
 - 最多支持3张量化表。
 - 只支持8bit采样精度。
 - 只支持对顺序式编码的图片进行解码。
 - 只支持基于DCT（Discrete Cosine Transform）变换的JPEG 格式解码。
 - 只支持一个SOS（Start of Scan）标志的图片解码。
- 软件约束：
 - 支持3个SOS标志的图片解码。

- 支持mcu (Minimum Coded Unit) 数据不足的异常图片解码。

精度相关约束

在Atlas 200/300/500 推理产品和Atlas 训练系列产品上，JPEG+VPC串联使用时，由于JPEG解码后的输出图片的宽stride*高stride有128*16对齐的约束，因此解码后的输出图片的宽、高有一些补边的无效数据，所以在执行VPC功能时（例如缩放时），需调用[aclDvppSetPicDescWidth](#)和[aclDvppSetPicDescHeight](#)接口正确设置输入图片的原图宽高，这样VPC在缩放图片前会先根据原图宽高自行抠图，目的是去除无效数据对图像精度的影响。

在Atlas 推理系列产品上，JPEG+VPC串联使用时，由于JPEG解码后的输出图片的宽stride*高stride有64*16对齐的约束，因此解码后的输出图片的宽、高有一些补边的无效数据，所以在执行VPC功能时（例如缩放时），需调用[aclDvppSetPicDescWidth](#)和[aclDvppSetPicDescHeight](#)接口正确设置输入图片的原图宽高，这样VPC在缩放图片前会先根据原图宽高自行抠图，目的是去除无效数据对图像精度的影响。

在Atlas A2训练系列产品/Atlas 800I A2推理产品上，JPEG+VPC串联使用时，由于JPEG解码后的输出图片的宽stride*高stride有64*16对齐的约束，因此解码后的输出图片的宽、高有一些补边的无效数据，所以在执行VPC功能时（例如缩放时），需调用[aclDvppSetPicDescWidth](#)和[aclDvppSetPicDescHeight](#)接口正确设置输入图片的原图宽高，这样VPC在缩放图片前会先根据原图宽高自行抠图，目的是去除无效数据对图像精度的影响。

在Atlas 200/500 A2推理产品上，JPEG+VPC串联使用时，由于JPEG解码后的输出图片的宽stride*高stride有64*16对齐的约束，因此解码后的输出图片的宽、高有一些补边的无效数据，所以在执行VPC功能时（例如缩放时），需调用[aclDvppSetPicDescWidth](#)和[aclDvppSetPicDescHeight](#)接口正确设置输入图片的原图宽高，这样VPC在缩放图片前会先根据原图宽高自行抠图，目的是去除无效数据对图像精度的影响。

1.8.6.2 性能指标说明

本节仅描述各产品对应的昇腾AI处理器的性能指标参考数据，各产品由于其内存、CPU等配置不同，在性能规格上会存在差异，详细产品的性能规格请查阅[昇腾社区](#)的官方说明。

本节以单个昇腾AI处理器为例，描述基本场景性能指标参考，如下表所示，其它分辨率可以等量估算。

下表中的1路对应一个通道，一个通道对应一个线程；各分辨率的含义为：1080p指分辨率为1920*1080的图片，4K指分辨率为3840*2160的图片。

JPEG性能指标是基于硬件解码的性能，JPEG硬件解码不支持3个SOS的图片解码，对于硬件不支持的格式，会使用软件解码，软件解码性能参考为1080P 15fps。JPEG解码的输出图片如果涉及旋转，则性能指标低于软件解码的参考值，例如对于1080P的图片，性能指标低于15fps。

性能指标说明（Atlas 200/300/500 推理产品）

以下性能数据，是基于一个stream上下发1个异步媒体数据处理任务后，执行一次[aclrtSynchronizeStream](#)接口。

表 1-19 Atlas 200/300/500 推理产品的昇腾 AI 处理性能指标参考数据

场景举例	总帧率
1080p*1路	128fps
1080p*n路 (n≥2)	256fps
4k*1路	32fps
4k*n路 (n≥2)	64fps

性能指标说明（Atlas 推理系列产品）

以下性能数据，是基于一个stream上下发10个异步媒体数据处理任务后，执行一次 [aclrtSynchronizeStream](#)接口，且Host服务器配置为：CPU主频为Intel(R) Xeon(R) CPU E5-2680 v4 @ 2.40GHz，CPU核心数为14。

表 1-20 Atlas 推理系列产品的昇腾 AI 处理性能指标参考数据

场景举例	总帧率
1080p*n路 (1≤n≤16)	n*128fps
1080p*n路 (n>16)	2048fps
4k*n路 (1≤n≤16)	n*32fps
4k*n路 (n>16)	512fps

性能指标说明（Atlas 训练系列产品）

以下性能数据，是基于一个stream上下发1个异步媒体数据处理任务后，执行一次 [aclrtSynchronizeStream](#)接口。

表 1-21 Atlas 训练系列产品的昇腾 AI 处理性能指标参考数据

场景举例	总帧率
1080p*n路 (1≤n≤16)	n*128fps
1080p*n路 (n>16)	2048fps
4k*n路 (1≤n≤16)	n*32fps
4k*n路 (n>16)	512fps

性能指标说明（Atlas 200/500 A2 推理产品）

以下性能数据，是基于一个stream上下发10个异步媒体数据处理任务后，执行一次 [aclrtSynchronizeStream](#)接口。

表 1-22 Atlas 200/500 A2 推理产品的昇腾 AI 处理性能指标参考数据

场景举例	总帧率
1080p*n路 (1≤n≤2)	n*256fps
1080p*n路 (n>2)	512fps
4k*n路 (1≤n≤2)	n*64fps
4k*n路 (n>2)	128fps

性能指标数据 (Atlas A2 训练系列产品/Atlas 800I A2 推理产品)

以下性能数据，是基于一个stream上下发10个异步媒体数据处理任务后，执行一次[aclrtSynchronizeStream](#)接口。

表 1-23 Atlas A2 训练系列产品/Atlas 800I A2 推理产品的昇腾 AI 处理性能指标参考数据

输入图片格式	输入图片文件大小	场景举例	总帧率
YUV400/420	≤500KB	1080p*n路 (1≤n≤28)	n*512fps
		1080p*n路 (n>28)	14336fps
YUV440/422/444	≤500KB	1080p*n路 (1≤n≤28)	n*256fps
		1080p*n路 (n>28)	7168fps
YUV400/420	≤2MB	4k*n路 (1≤n≤28)	n*128fps
		4k*n路 (n>28)	3584fps
YUV440/422/444	≤2MB	4k*n路 (1≤n≤28)	n*64fps
		4k*n路 (n>28)	1792fps

1.8.6.3 acldvppJpegDecodeAsync

函数功能

解码.jpg、.jpeg、.JPG、.JPEG图片，异步接口。

约束说明

该接口是异步接口，调用接口成功仅表示任务下发成功，不表示任务执行成功。调用该接口后，需调用同步等待接口（例如，[aclrtSynchronizeStream](#)）确保任务已执行完成，否则可能会导致训练或推理等业务异常、Device断链掉卡等未知情况。

函数原型

```
aclError acldvppJpegDecodeAsync(aclvppChannelDesc *channelDesc,  
const void *data,  
uint32_t size,  
aclvppPicDesc *outputDesc,  
aclrtStream stream)
```

参数说明

参数名	输入/输出	说明
channelDesc	输入	通道描述信息的指针。 与调用 aclvppCreateChannel 接口创建通道时指定的channelDesc保持一致。
data	输入	输入图片内存地址的指针。
size	输入	输入图片的实际数据大小，单位Byte。

参数名	输入/输出	说明
outputDesc	输入&输出	输出图片信息的指针。 outputDesc参数作为输入时，需要用户调用如下接口： - 调用acldvppCreatePicDesc接口创建图片描述信息。 - 调用acldvppSetPicDesc系列接口设置输出图片的内存地址、内存大小、格式、widthStride、heightStride。 - 输出图片的内存大小可提前调用acldvppJpegPredictDecSize接口预估。 - JPEG原图的宽、高可通过acldvppJpegGetImageInfo/acldvppJpegGetImageInfoV2接口获取。 - 输出图片格式、输出图片的宽高对齐要求，请参见1.8.6.1 功能及约束说明。 outputDesc参数作为输出时，用户可以从内存地址中获取解码后的输出图片数据、可以调用acldvppGetPicDesc系列接口获取输出图片的宽/高。 说明 Atlas 推理系列产品、Atlas 200/500 A2推理产品、Atlas A2训练系列产品/Atlas 800I A2推理产品上，如果解码后的输出图片数据需要在后续操作（例如，使用VPC实现抠图、缩放等功能）中使用，建议在解码后调用acldvppGetPicDesc系列接口获取输出图片的宽/高，因为解码过程中可能会对输出图片的宽高执行对齐操作，例如，jpeg(444) 源码的图片解码成YUV420SP格式的输出图片时，当jpeg(444) 源码图片的宽/高为奇数时，解码出来的YUV420SP格式的输出图片的宽/高理论上应该为奇数，但是YUV420SP格式本身要求图片的宽/高都为偶数，这时JPEGD会对奇数宽/高做向下2对齐的操作。
stream	输入	指定Stream。

返回值说明

返回0表示任务下发成功，返回非0表示任务下发失败。

参考资源

接口调用流程及示例，参见JPEGD图像解码。

1.8.6.4 acldvppJpegGetImageInfo

函数功能

从存放JPEG图片数据的内存中读取JPEG图片的宽、高、颜色通道个数。

该接口会检查图片格式是否支持解码，如果遇到不支持的码流则该接口会返回ACL_ERROR_UNSUPPORTED_JPEG错误码，该错误码的详细描述请参见[aclError](#)。

函数原型

```
aclError acldvppJpegGetImageInfo(const void *data,  
uint32_t size,  
uint32_t *width,  
uint32_t *height,  
int32_t *components)
```

参数说明

参数名	输入/输出	说明
data	输入	存放JPEG图片数据的内存地址指针。
size	输入	内存大小，单位为Byte。
width	输出	图片宽的指针。
height	输出	图片高的指针。
components	输出	颜色通道个数的指针。

返回值说明

返回0表示获取信息成功，返回非0表示获取信息失败。

1.8.6.5 acldvppJpegGetImageInfoV2

函数功能

从存放JPEG图片数据的内存中读取JPEG图片的宽、高、颜色通道个数、JPEG原图编码格式。

该接口会检查图片格式是否支持解码，如果遇到不支持的码流则该接口会返回ACL_ERROR_UNSUPPORTED_JPEG错误码，该错误码的详细描述请参见[aclError](#)。

函数原型

```
aclError acldvppJpegGetImageInfoV2(const void *data,  
uint32_t size,  
uint32_t *width,  
uint32_t *height,  
int32_t *components,  
aclDvppJpegFormat *format)
```


参数说明

参数名	输入/输出	说明
data	输入	存放JPEG图片数据的内存地址指针。
size	输入	内存大小，单位为Byte。
width	输出	图片宽的指针。
height	输出	图片高的指针。
components	输出	颜色通道个数的指针。
format	输出	JPEG原图编码格式的指针。

返回值说明

返回0表示获取信息成功，返回非0表示获取信息失败。

1.8.6.6 aclvppJpegPredictDecSize

函数功能

根据存放JPEG图片数据的内存预估JPEG图片解码后所需的输出内存的大小。

函数原型

```
aclError aclvppJpegPredictDecSize(const void *data,  
uint32_t dataSize,  
aclvppPixelFormat outputPixelFormat,  
uint32_t *decSize)
```

参数说明

参数名	输入/输出	说明
data	输入	存放JPEG图片数据的内存地址指针。
size	输入	内存大小，单位为Byte。
outputPixelFormat	输入	解码后的输出图片的格式。
decSize	输出	预估JPEG图片解码后所需输出内存大小的指针，单位为Byte。

返回值说明

返回0表示获取信息成功，返回非0表示获取信息失败。

参考资源

接口调用流程及示例，参见JPEGD图像解码。

1.8.7 JPEGG 功能

1.8.7.1 功能及约束说明

功能说明

JPEGG (JPEG Encoder) 将YUV格式图片编码成JPEG压缩格式的图片文件，例如*.jpg。

图片分辨率约束

- 输入图片分辨率：

版本	分辨率范围
Atlas 200/300/500 推理产品 Atlas 训练系列产品 Atlas 推理系列产品 Atlas A2训练系列产品/Atlas 800I A2推理产品	最大分辨率：8192*8192，最小分辨率：32*32。
Atlas 200/500 A2推理产品	最大分辨率：16384*16384，最小分辨率：32*32。

- 输出图片分辨率
JPEGG只对图片编码，不会改变图片分辨率，因此输出与输入的图片分辨率保持一致。

图片格式、宽高对齐、内存约束

实现JPEGG图片编码功能时，需调用[aclDvppMalloc](#)接口申请Device上的输入、输出内存，调用[aclDvppFree](#)接口释放输入、输出内存，这部分内存的生命周期由用户自行管理。

- 输入内存的大小请参见下表中的计算公式。
- 输出内存的大小就是指实际的编码后图片所占用的大小，可调用[aclDvppJpegPredictEncSize](#)接口预估输出内存大小。

📖 说明

输入图片格式的定义请参见[aclDvppPixelFormat](#)，宽stride、高stride等概念请参见[1.8.2 基本概念](#)。

表 1-24 图片格式、宽高对齐、内存大小约束

输入图片格式	输入图片宽、高对齐要求	输入图片宽stride、高stride、内存大小要求	输出图片格式
YUV420SP NV12 8bit	宽2对齐 高2对齐	宽stride为宽16对齐后的值，兼容对齐到16的倍数如128（按128对齐时，性能更优）。 高stride支持以下两种配置： - 配置为与输入图片的高度相同的数值； - 或配置为输入图片的高度向上对齐到16的数值（最小为32）。该方式的使用场景举例：JPEGD的输出图片直接作为JPEGE的输入（JPEGD输出图片高度是向上对齐到16的）。 内存大小（单位Byte） = 宽stride * 高stride * 3/2	JPEG压缩格式的图片文件，例如*.jpg。 只支持huffman编码，不支持算术编码，不支持渐进编码。
YUV420SP NV21 8bit			
YUV422Packed UYVY 8bit	宽2对齐 高无对齐要求	宽stride为宽的两倍再16对齐后的值。 高stride支持以下两种配置： - 配置为与输入图片的高度相同的数值； - 或配置为输入图片的高度向上对齐到16的数值（最小为32）。该方式的使用场景举例：JPEGD的输出图片直接作为JPEGE的输入（JPEGD输出图片高度是向上对齐到16的）。 内存大小（单位Byte） = 宽stride * 高stride	
YUV422Packed UYVY 8bit			
YUV422Packed YVYU 8bit			
YUV422Packed VYUY 8bit			

1.8.7.2 性能指标说明

本节仅描述各产品对应的昇腾AI处理器的性能指标参考数据，各产品由于其内存、CPU等配置不同，在性能规格上会存在差异，详细产品的性能规格请查阅[昇腾社区](#)的官方说明。

本节以单个昇腾AI处理器为例，描述基本场景性能指标参考，如下表所示，其它分辨率可以等量估算。

下表中的1路对应一个通道，一个通道对应一个线程；各分辨率的含义为：1080p指分辨率为1920*1080的图片，4K指分辨率为3840*2160的图片。

性能指标说明（Atlas 200/300/500 推理产品）

以下性能数据，是基于一个stream上下发10个异步媒体数据处理任务后，执行一次[aclrtSynchronizeStream](#)接口。

表 1-25 Atlas 200/300/500 推理产品的昇腾 AI 处理器性能指标参考数据

场景举例	总帧率
1080p*n路 (n≥1)	64fps
4k*n路 (n≥1)	16fps

性能指标说明（Atlas 推理系列产品）

以下性能数据，是基于一个stream上下发10个异步媒体数据处理任务后，执行一次[aclrtSynchronizeStream](#)接口，且Host服务器配置为：CPU主频为Intel(R) Xeon(R) CPU E5-2680 v4 @ 2.40GHz，CPU核心数为14。

表 1-26 Atlas 推理系列产品的昇腾 AI 处理器性能指标参考数据

场景举例（输入图片格式：YUV420 8bit）	总帧率
1080p*n路 (1≤n≤8)	n*128fps
1080p*n路 (n>8)	1024fps
4k*n路 (1≤n≤8)	n*32fps
4k*n路 (n>8)	256fps

性能指标说明（Atlas 训练系列产品）

以下性能数据，是基于一个stream上下发10个异步媒体数据处理任务后，执行一次[aclrtSynchronizeStream](#)接口。

表 1-27 Atlas 训练系列产品的昇腾 AI 处理器性能指标参考数据

场景举例（输入图片格式：YUV420 8bit）	总帧率
1080p*n路 (1≤n≤8)	n*128fps

场景举例（输入图片格式：YUV420 8bit）	总帧率
1080p*n路（n>8）	1024fps
4k*n路（1≤n≤8）	n*32fps
4k*n路（n>8）	256fps

性能指标说明（Atlas 200/500 A2 推理产品）

以下性能数据，是基于一个stream上下发10个异步媒体数据处理任务后，执行一次[aclrtSynchronizeStream](#)接口。

表 1-28 Atlas 200/500 A2 推理产品的昇腾 AI 处理器性能指标参考数据

场景举例（输入图片格式：YUV420 8bit）	总帧率
1080p*n路	256fps
4k*n路	64fps

性能指标说明（Atlas A2 训练系列产品/Atlas 800I A2 推理产品）

以下性能数据，是基于一个stream上下发10个异步媒体数据处理任务后，执行一次[aclrtSynchronizeStream](#)接口。

表 1-29 Atlas A2 训练系列产品/Atlas 800I A2 推理产品的昇腾 AI 处理器性能指标参考数据

场景举例（输入图片格式：YUV420 8bit）	总帧率
1080p*n路（1≤n≤4）	n*256fps
1080p*n路（n>4）	1024fps
4k*n路（1≤n≤4）	n*64fps
4k*n路（n>4）	256fps

1.8.7.3 aclDvppJpegEncodeAsync

函数功能

将YUV格式图片编码成.jpg图片，异步接口。

约束说明

该接口是异步接口，调用接口成功仅表示任务下发成功，不表示任务执行成功。调用该接口后，需调用同步等待接口（例如，[aclrtSynchronizeStream](#)）确保任务已执行完成，否则可能会导致训练或推理等业务异常、Device断链掉卡等未知情况。

函数原型

```
aclError acldvppJpegEncodeAsync(acldvppChannelDesc *channelDesc,  
acldvppPicDesc *inputDesc,  
const void *data,  
uint32_t *size,  
acldvppJpegeConfig *config,  
aclrtStream stream);
```

参数说明

参数名	输入/输出	说明
channelDesc	输入	通道描述信息的指针。 与调用 acldvppCreateChannel 接口创建通道时指定的channelDesc保持一致。
inputDesc	输入	输入图片描述信息的指针。 - 调用acldvppCreatePicDesc接口创建图片描述信息。 - 调用acldvppSetPicDesc系列接口设置图片描述参数（例如，图片格式、宽、高等）。 输入图片分辨率（≤通道的最大宽度*高度）、图像格式的要求，请参见1.8.7.1 功能及约束说明。
data	输出	输出内存地址的指针，存放编码后的数据。
size	输入&输出	输出内存大小的指针，单位Byte。 size作为输入参数时，可提前调用 acldvppJpegPredictEncSize 接口预估输出内存大小。 size作为输出参数时，表示实际输出内存大小，可能与调用 acldvppJpegPredictEncSize 接口预估的内存大小存在差异，如果用户需要取用编码后的数据，请使用实际输出内存大小。
config	输入	图片编码配置数据的指针。 调用 acldvppCreateJpegeConfig 接口创建图片编码配置数据，调用 acldvppSetJpegeConfigLevel 接口设置编码配置数据。
stream	输入	指定Stream。

返回值说明

返回0表示任务下发成功，返回非0表示任务下发失败。

参考资源

接口调用流程及示例，参见JPEG图片编码。

1.8.7.4 aclvppJpegPredictEncSize

函数功能

根据输入图片描述信息、图片编码配置数据预估图片编码后所需的输出内存的大小。

函数原型

```
aclError aclvppJpegPredictEncSize(const aclvppPicDesc *inputDesc, const
aclvppJpegeConfig *config, uint32_t *size)
```

参数说明

参数名	输入/输出	说明
inputDesc	输入	输入图片描述信息的指针。 调用 aclvppCreatePicDesc 接口创建图片描述信息，调用 aclvppSetPicDesc系列接口 设置图片描述参数（例如，图片格式、宽、高等）。
config	输入	图片编码配置数据的指针。 调用 aclvppCreateJpegeConfig 接口创建图片编码配置数据。
size	输出	预估JPEG图片编码后所需输出内存大小的指针，单位为Byte。 预估的输出内存会大于实际输出内存，实际输出内存需从 aclvppJpegEncodeAsync 接口的输出参数size获取。

返回值说明

返回0表示成功，返回其他值表示失败。

参考资源

接口调用流程及示例，参见JPEG图片编码。

1.8.8 PNGD 功能

1.8.8.1 功能及约束说明

功能说明

PNGD (PNG Decoder) 功能：实现PNG格式图片的解码。

PNGD在解码图片时，支持按源图片格式解码。在调用PNGD解码接口时，直接将输出图片格式配置为PIXEL_FORMAT_UNKNOWN，输出格式默认按源图片格式输出。

源图片格式解码是指解码前后图片的编码格式保持一致，例如解码前输入图片格式为RGB，解码后输出图片格式为RGB888。

图片分辨率约束

- 输入图片分辨率：
最大分辨率4096*4096，最小分辨率32*32。
- 输出图片分辨率
PNGD只对图片解码，不会改变图片分辨率，因此输出与输入的图片分辨率保持一致。

图片格式、宽高对齐、内存约束

实现图片解码功能时，需调用[aclvppMalloc](#)接口申请Device上的输入、输出内存，调用[aclvppFree](#)接口释放输入、输出内存，这部分内存的生命周期由用户自行管理。

- 输入内存的大小就是指实际的输入图片所占用的大小。
- 输出内存的大小可调用[aclvppPngPredictDecSize](#)接口预估。

 说明

输出图片格式的定义请参见[aclvppPixelFormat](#)，宽stride、高stride等概念请参见[1.8.2 基本概念](#)。

表 1-30 图片格式、宽高对齐、内存大小约束

输入图片格式	输出图片格式	输出图片宽、高对齐要求	输出图片宽stride、高stride、内存大小要求
RGB	RGB888	无对齐要求	宽stride为宽128对齐后再乘以3的值。 高stride为高16对齐后的值。 输出内存的大小可调用 aclvppPngPredictDecSize 接口预估。
GRAY	RGB888	无对齐要求	宽stride为宽128对齐后再乘以3的值。 高stride为高16对齐后的值。 输出内存的大小可调用 aclvppPngPredictDecSize 接口预估。

输入图片格式	输出图片格式	输出图片宽、高 对齐要求	输出图片宽stride、高stride、内存 大小要求
RGBA	RGB888	无对齐要求	宽stride为宽128对齐后再乘以3的 值。 高stride为高16对齐后的值。 输出内存的大小可调用 acldvppPngPredictDecSize 接口预 估。
	RGBA8888 8bit	无对齐要求	宽stride为宽128对齐后再乘以4的 值。 高stride为高16对齐后的值。 输出内存的大小可调用 acldvppPngPredictDecSize 接口预 估。
AGRAY	RGB888	无对齐要求	宽stride为宽128对齐后再乘以3的 值。 高stride为高16对齐后的值。 输出内存的大小可调用 acldvppPngPredictDecSize 接口预 估。
	RGBA8888 8bit	无对齐要求	宽stride为宽128对齐后再乘以4的 值。 高stride为高16对齐后的值。 输出内存的大小可调用 acldvppPngPredictDecSize 接口预 估。

1.8.8.2 性能指标说明

本节仅描述各产品对应的昇腾AI处理器的性能指标参考数据，各产品由于其内存、CPU等配置不同，在性能规格上会存在差异，详细产品的性能规格请查阅[昇腾社区](#)的官方说明。

本节以单个昇腾AI处理器为例，描述基本场景性能指标参考，如下表所示，其它分辨率可以等量估算。

下表中的1路对应一个通道，一个通道对应一个线程；各分辨率的含义为：1080p指分辨率为1920*1080的图片，4K指分辨率为3840*2160的图片。

性能指标说明（Atlas 200/300/500 推理产品）

以下性能数据，是基于一个stream上下发10个异步媒体数据处理任务后，执行一次[aclrtSynchronizeStream](#)接口。

表 1-31 Atlas 200/300/500 推理产品的昇腾 AI 处理器性能指标参考数据

场景举例	总帧率
1080p*n路 (1≤n≤5)	n*4fps
1080p*n路 (n≥ 6)	24fps
4k*n路 (1≤n≤5)	n*1fps
4k*n路 (n≥ 6)	6fps

性能指标说明（Atlas 推理系列产品）

以下性能数据，是基于一个stream上下发10个异步媒体数据处理任务后，执行一次[aclrtSynchronizeStream](#)接口，且Host服务器配置为：CPU主频为Intel(R) Xeon(R) CPU E5-2680 v4 @ 2.40GHz，CPU核心数为14。

表 1-32 Atlas 推理系列产品的昇腾 AI 处理器性能指标参考数据

场景举例	总帧率
1080p*n路 (1≤n≤5)	n*4fps
1080p*n路 (n≥ 6)	24fps
4k*n路 (1≤n≤5)	n*1fps
4k*n路 (n≥ 6)	6fps

性能指标说明（Atlas 训练系列产品）

以下性能数据，是基于一个stream上下发10个异步媒体数据处理任务后，执行一次[aclrtSynchronizeStream](#)接口。

表 1-33 Atlas 训练系列产品的昇腾 AI 处理器性能指标参考数据

场景举例	总帧率
1080p*n路 (1≤n≤24)	n*4fps
1080p*n路 (n>24)	96fps
4k*n路 (1≤n≤24)	n*1fps
4k*n路 (n>24)	24fps

性能指标说明（Atlas 200/500 A2 推理产品）

以下性能数据，是基于一个stream上下发10个异步媒体数据处理任务后，执行一次[aclrtSynchronizeStream](#)接口。

表 1-34 Atlas 200/500 A2 推理产品的昇腾 AI 处理器性能指标参考数据

场景举例	总帧率
1080p*n路 (1≤n≤5)	n*4fps
1080p*n路 (n≥ 6)	24fps
4k*n路 (1≤n≤5)	n*1fps
4k*n路 (n≥ 6)	6fps

性能指标说明 (Atlas A2 训练系列产品/Atlas 800I A2 推理产品)

以下性能数据，是基于一个stream上下发10个异步媒体数据处理任务后，执行一次[aclrtSynchronizeStream](#)接口。

表 1-35 Atlas A2 训练系列产品/Atlas 800I A2 推理产品的昇腾 AI 处理器性能指标参考数据

场景举例	总帧率
1080p*n路 (1≤n≤5)	n*4fps
1080p*n路 (n≥ 6)	24fps
4k*n路 (1≤n≤5)	n*1fps
4k*n路 (n≥ 6)	6fps

1.8.8.3 acldvppPngDecodeAsync

函数功能

解码png图片，异步接口。

功能	输入图片格式	输出图片格式
调用本接口时将输出图片格式设置为 PIXEL_FORMAT_UNKNOWN 表示不指定输出图片格式，输出图片格式与输入 图片格式一致	RGB/GRAY	RGB
调用本接口时将输出图片格式设置为 PIXEL_FORMAT_UNKNOWN 表示不指定输出图片格式，输出图片格式与输入 图片格式一致	RGBA/AGRAY	RGBA
调用本接口时将输出图片格式设置为 PIXEL_FORMAT_RGB_888，表示指定输出图片 格式，将输入图片格式转换为输出图片格式	RGB/RGBA/ GRAY/AGRAY	RGB

约束说明

该接口是异步接口，调用接口成功仅表示任务下发成功，不表示任务执行成功。调用该接口后，需调用同步等待接口（例如，[aclrtSynchronizeStream](#)）确保任务已执行完成，否则可能会导致训练或推理等业务异常、Device断链掉卡等未知情况。

函数原型

```
aclError acldvppPngDecodeAsync(aclvppChannelDesc *channelDesc,  
const void *data,  
uint32_t size,  
aclvppPicDesc *outputDesc,  
aclrtStream stream)
```

参数说明

参数名	输入/输出	说明
channelDesc	输入	通道描述信息的指针。 与调用 acldvppCreateChannel 接口创建通道时指定的channelDesc保持一致。
data	输入	输入图片内存地址的指针。
size	输入	输入图片的实际数据大小，单位Byte。
outputDesc	输入&输出	输出图片信息的指针。 outputDesc参数作为输入时，需要用户调用如下接口： - 调用acldvppCreatePicDesc接口创建图片描述信息。 - 调用acldvppSetPicDesc系列接口设置输出图片的内存地址、内存大小、格式。 - 输出图片的内存大小可提前调用acldvppPngPredictDecSize接口获取。 - png原图信息（例如宽、高）可通过acldvppPngGetImageInfo接口获取。 - 输出图片格式、内存等要求，请参见1.8.8.1 功能及约束说明。 outputDesc参数作为输出时，用户可以从内存地址中获取解码后的输出图片数据。
stream	输入	指定Stream。

返回值说明

返回0表示任务下发成功，返回非0表示任务下发失败。

参考资源

接口调用流程及示例，参见PNGD图片解码。

1.8.8.4 aclDvppPngGetImageInfo

函数功能

从存放png图片数据的内存中读取png图片的宽、高。

函数原型

```
aclError aclDvppPngGetImageInfo(const void *data,  
uint32_t dataSize,  
uint32_t *width,  
uint32_t *height,  
int32_t *components)
```

参数说明

参数名	输入/输出	说明
data	输入	存放png图片数据内存地址的指针。
dataSize	输入	内存大小，单位为Byte。
width	输出	图片宽的指针。
height	输出	图片高的指针。
components	输出	颜色通道个数的指针。

返回值说明

返回0表示成功，返回其他值表示失败。

1.8.8.5 aclDvppPngPredictDecSize

函数功能

根据存放png图片数据的内存计算出png图片解码后所需的输出内存的大小。

函数原型

```
aclError aclDvppPngPredictDecSize(const void *data,  
uint32_t dataSize,  
aclDvppPixelFormat outputPixelFormat,
```

uint32_t *decSize)

参数说明

参数名	输入/输出	说明
data	输入	存放png图片数据内存地址的指针。
size	输入	内存大小，单位为Byte。
outputPixelFormat	输入	解码后的输出图片的格式，支持设置如下格式： - PIXEL_FORMAT_UNKNOWN：按占用内存大小最大的PNG解码输出格式计算内存大小。 - PIXEL_FORMAT_RGB_888 - PIXEL_FORMAT_RGBA_8888
decSize	输出	png图片解码后所需输出内存大小的指针，单位为Byte。

返回值说明

返回0表示成功，返回其他值表示失败。

参考资源

接口调用流程及示例，参见PNGD图片解码。

1.8.9 VDEC 功能

1.8.9.1 功能及约束说明

功能说明

VDEC (Video Decoder) 实现视频的解码，VDEC内部经过VPC处理后，输出指定格式的图片，参见[码流/图片格式](#)、[宽高对齐](#)、[内存约束](#)。

分辨率约束

- 输入码流分辨率：

版本	分辨率范围
Atlas 200/300/500 推理产品 Atlas 训练系列产品 Atlas 推理系列产品 Atlas A2训练系列产品/Atlas 800I A2推理产品	最大分辨率4096*4096，最小分辨率128*128。
Atlas 200/500 A2推理产品	对于H264输入码流，最大分辨率8192*8192，最小分辨率128*128。 对于H265输入码流，最大分辨率5400*8192，最小分辨率128*128。

• 输出图片分辨率：

版本	分辨率范围
Atlas 200/300/500 推理产品 Atlas 训练系列产品 Atlas 推理系列产品 Atlas A2训练系列产品/Atlas 800I A2推理产品	最大分辨率4096*4096，最小分辨率10*6。
Atlas 200/500 A2推理产品	输出图片最大分辨率8192*8192；当输入码流宽度小于或等于4096时，输出图片最小分辨率为10*6，当输入码流宽度大于4096时，输出图片最小分辨率为128*128。

码流/图片格式、宽高对齐、内存约束

实现VDEC视频解码功能时，输入或输出内存的生命周期由用户自行管理：

- 输入内存
调用[acldvppMalloc](#)/[acldvppFree](#)接口申请/释放内存、或调用[aclrtMalloc](#)/[aclrtFree](#)接口申请/释放内存，内存大小就是指实际的输入码流所占用的大小。
调用[acldvppMalloc](#)接口申请的内存为媒体数据处理的专用内存，但专用内存的地址空间有限，若关注内存规划或内存资源有限时，使用VDEC视频解码功能时，建议调用[aclrtMalloc](#)接口申请内存。
- 输出内存
需调用[acldvppMalloc](#)接口申请Device上的输出内存，调用[acldvppFree](#)接口释放输出内存，内存大小请参见下表中的计算公式。

 说明

输出图片格式的定义请参见[acldvppPixelFormat](#)，宽stride、高stride等概念请参见[1.8.2 基本概念](#)。

VDEC视频解码时，输入码流支持如下格式：

- H264 bp/mp/hp level5.1 YUV420编码的码流，当前只支持annex-B的裸码流。
- H265 8/10bit level5.1 YUV420编码的码流，当前只支持annex-B的裸码流。

VDEC视频解码时，输出图片格式、宽高对齐要求等如下表所示：

表 1-36 码流/图片格式、宽高对齐、内存大小约束

版本	输出图片格式	输出图片宽、高对齐要求	输出图片宽stride、高stride、内存大小要求 Atlas 200/300/500推理产品 Atlas 训练系列产品 Atlas 推理系列产品	输出图片宽stride、高stride、内存大小要求 Atlas 200/500 A2推理产品 Atlas A2训练系列产品/Atlas 800I A2推理产品
通用格式，各版本都支持	YUV420SP NV12 8bit	宽2对齐 高2对齐	宽stride为宽16对齐后的值。如果用户设置的输出图片宽度小于16，则宽stride最小为32。 高stride为高2对齐后的值。 内存大小（单位Byte）≥ 宽stride * 高stride * 3/2	宽stride无对齐要求，与宽相同即可，最小10，最大24576。 高stride无对齐要求，与高相同即可，最小6，最大24576。 内存大小（单位Byte）≥ 宽stride * 高stride * 3/2
	YUV420SP NV21 8bit			
该部分格式仅以下版本支持： Atlas 推理系列产品 Atlas 200/500 A2推理产品 Atlas A2训练系列产品/ Atlas 800I A2推理产品	RGB888	无对齐要求	宽stride为宽16对齐后再乘以3的值。 高stride无对齐要求。 内存大小（单位Byte）≥ 宽stride * 高stride	宽stride为宽乘以3的值，最小30，最大24576。 高stride无对齐要求，最小6，最大24576。 内存大小（单位Byte）≥ 宽stride * 高stride
	BGR888			

其它约束

- VDEC只支持按帧输入码流进行解码。
- 若码流中有坏帧、缺帧等情况，解码器VDEC可能会丢帧。
- 通过隔行扫描方式编码出来的码流，VDEC仅支持解码H264 8bit编码的码流。

1.8.9.2 性能指标说明

本节仅描述各产品对应的昇腾AI处理器的性能指标参考数据，各产品由于其内存、CPU等配置不同，在性能规格上会存在差异，详细产品的性能规格请查阅[昇腾社区](#)的官方说明。

本节以单个昇腾AI处理器为例，描述基本场景性能指标参考，如下表所示，其它分辨率可以等量估算。

下表中的1路对应一个通道，一个通道对应一个线程；各分辨率的含义为：720p指分辨率为1280*720的图片，1080p指分辨率为1920*1080的图片，4K指分辨率为3840*2160的图片。

性能指标说明（Atlas 200/300/500 推理产品）

- 针对如下典型场景下各分辨率的码流，单个Device的VDEC解码的路数推荐如下值：

表 1-37 Atlas 200/300/500 推理产品的昇腾 AI 处理器性能指标参考数据

分辨率	总性能	单路性能（n路）	每路VDEC解码的最大内存消耗（此处以H265格式的输入码流、参考帧数量9个为例，作为参考值）
4k	120fps	120fps/n（推荐n=4，每路30fps）	约182MB
1080p	480fps	480fps/n（推荐n=16，每路30fps）	约56MB
≤720p	960fps	960fps/n（推荐n=32，每路30fps）	约35MB

下表以1080P分辨率的输入码流为例，说明每路VDEC解码的最大内存消耗的计算公式，在计算公式中：

- 输入码流缓存大小：1080P分辨率及以下的输入码流，输入码流缓存大小默认为6M；1080P分辨率以上的输入码流，输入码流缓存大小默认为9M。
- 解码图像帧存大小：1080P分辨率的输入码流，该参数值为3MB。其它分辨率时参数值可等量折算。
- 视频解码图像Tmv缓存大小：H264格式、1080P分辨率的输入码流，该参数值为0.5MB；H265格式、1080P分辨率的输入码流，该参数值为1MB。其它分辨率时参数值可等量折算。

- 参考帧数量：以最大参考帧个数为例，系统内部会根据实际码流的参考帧个数自适应。

每路VDEC解码的内存消耗计算公式	场景举例	内存消耗（单位为MB）
$4\text{MB} + \text{输入码流缓存大小} * 2 + (\text{解码图像帧存大小} + \text{视频解码图像Tmv缓存大小}) * (\text{参考帧数量} + 1)$	• 输入码流格式H264 • 输入码流分辨率1080P • 输入码流缓存大小为6M • 解码图像帧存大小为3MB • 视频解码图像Tmv缓存大小为0.5MB	51MB（参考帧数量9个） 26.5MB（参考帧数量2个）
	• 输入码流格式H265 • 输入码流分辨率1080P • 输入码流缓存大小为6M • 解码图像帧存大小为3MB • 视频解码图像Tmv缓存大小为1MB	56MB（参考帧数量9个） 28MB（参考帧数量2个）

- 关于解码路数与帧率的建议
注意：下表中给出的规格建议供参考，如果单进程内启动的路数超过下表中的建议，则可能出现内存不足或性能不够的问题，进而导致创建解码通道失败或执行解码缓慢。

720p指分辨率为1280*720的图片；1080p指分辨率为1920*1080的图片；4K指分辨率为3840*2160的图片。

典型分辨率	单进程内各启动路数时的规格建议（依据输入帧率得出）				
-	输入帧率 ≥25fps	20fps<输入帧率 <25fps	15fps<输入帧率 ≤20fps	10fps<输入帧率 ≤15fps	输入帧率 ≤10fps
≤720p	32路	32路	32路	32路	32路
1080p	16路	19路	24路	32路	32路
4K	4路	4路	6路	8路	12路

性能指标说明（Atlas 推理系列产品）

以下性能数据，基于以下Host服务器配置：CPU主频为Intel(R) Xeon(R) CPU E5-2680 v4 @ 2.40GHz，CPU核心数为14。

表 1-38 Atlas 推理系列产品的昇腾 AI 处理器性能指标参考数据

场景举例	总帧率
720p*n路(1≤n≤12)	n*480fps 说明 针对 Atlas 300V 视频解析卡，当1<n≤12时，总帧率都为n*400fps。
720p*n路(n>12)	5760fps 说明 针对 Atlas 300V 视频解析卡，当n>12时，总帧率都为4800fps。
1080p*n路 (1≤n≤12)	n*240fps 说明 针对 Atlas 300V 视频解析卡，当1<n≤12时，总帧率都为n*200fps。
1080p*n路(n>12)	2880fps 说明 针对 Atlas 300V 视频解析卡，当n>12时，总帧率都为2400fps。
4k*n路(1≤n≤12)	n*60fps
4k*n路(n>12)	720fps

下表说明了典型场景下每路VDEC解码的内存消耗，在内存消耗的计算公式中：

- 输入码流缓存大小：等于输入码流的宽*高*2。
- 解码图像帧存大小：1080P分辨率的输入码流，该参数值为3MB。其它分辨率时参数值可等量折算。
- 视频解码图像Tmv缓存大小：H264格式、1080P分辨率的输入码流，该参数值为0.5MB；H265格式、1080P分辨率的输入码流，该参数值为1MB。其它分辨率时参数值可等量折算。
- 参考帧数量可由用户调用[aclvdecSetChannelDescRefFrameNum](#)接口设置。
- 解码后缓存图像帧数固定为2个。

每路VDEC解码的内存消耗计算公式	场景举例	内存消耗（单位为MB）
6MB + 输入码流缓存大小 * 2 + (解码图像帧存大小 + 视频解码图像Tmv缓存大小) * (参考帧数量 + 解码后缓存图像帧数 + 1)	• 输入码流格式H264 • 输入码流分辨率1080P • 输入码流缓存大小为4MB • 解码图像帧存大小为3MB • 视频解码图像Tmv缓存大小为0.5MB • 解码后缓存图像帧数2个	52.5MB（参考帧数量8个） 31.5MB（参考帧数量2个）

每路VDEC解码的内存消耗计算公式	场景举例	内存消耗（单位为MB）
	• 输入码流格式H265 • 输入码流分辨率1080P • 输入码流缓存大小为4MB • 解码图像帧存大小为3MB • 视频解码图像Tmv缓存大小为1MB • 解码后缓存图像帧数2个	58MB（参考帧数量8个） 34MB（参考帧数量2个）

性能指标说明（Atlas 训练系列产品）

表 1-39 Atlas 训练系列产品的昇腾 AI 处理器性能指标参考数据

场景举例	总帧率
1080p*n路*4进程 (n≤32)	n*30*4fps
4k*n路*4进程 (n≤8)	n*30*4fps

下表以1080P分辨率的输入码流为例，说明每路VDEC解码的最大内存消耗的计算公式，在计算公式中：

- 输入码流缓存大小：1080P分辨率及以下的输入码流，输入码流缓存大小默认为6M；1080P分辨率以上的输入码流，输入码流缓存大小默认为9M。
- 解码图像帧存大小：1080P分辨率的输入码流，该参数值为3MB。其它分辨率时参数值可等量折算。
- 视频解码图像Tmv缓存大小：H264格式、1080P分辨率的输入码流，该参数值为0.5MB；H265格式、1080P分辨率的输入码流，该参数值为1MB。其它分辨率时参数值可等量折算。
- 参考帧数量：以最大参考帧个数为例，系统内部会根据实际码流的参考帧个数自适应。

每路VDEC解码的内存消耗计算公式	场景举例	内存消耗（单位为MB）
4MB + 输入码流缓存大小 * 2 + (解码图像帧存大小 + 视频解码图像Tmv缓存大小) * (参考帧数量 + 1)	• 输入码流格式H264 • 输入码流分辨率1080P • 输入码流缓存大小为6M • 解码图像帧存大小为3MB • 视频解码图像Tmv缓存大小为0.5MB	51MB（参考帧数量9个） 26.5MB（参考帧数量2个）
	• 输入码流格式H265 • 输入码流分辨率1080P • 输入码流缓存大小为6M • 解码图像帧存大小为3MB • 视频解码图像Tmv缓存大小为1MB	56MB（参考帧数量9个） 28MB（参考帧数量2个）

性能指标说明（Atlas 200/500 A2 推理产品）

表 1-40 Atlas 200/500 A2 推理产品的昇腾 AI 处理器性能指标参考数据

场景举例	总帧率
720p*n路(1≤n≤4)	n*600fps
720p*n路(n>4)	2400fps
1080p*n路(1≤n≤4)	n*300fps
1080p*n路(n>4)	1200fps
4k*n路(1≤n≤4)	n*75fps
4k*n路(n>4)	300fps

下表说明了典型场景下每路VDEC解码的内存消耗，在内存消耗的计算公式中：

- 输入码流缓存大小：等于输入码流的宽*高*2。
- 解码图像帧存大小：1080P分辨率的输入码流，该参数值为3MB。其它分辨率时参数值可等量折算。
- 视频解码图像Tmv缓存大小：H264格式、1080P分辨率的输入码流，该参数值为0.5MB；H265格式、1080P分辨率的输入码流，该参数值为1MB。其它分辨率时参数值可等量折算。
- 参考帧数量可由用户调用[aclvdecSetChannelDescRefFrameNum](#)接口设置。

- 解码后缓存图像帧数固定为2个。

每路VDEC解码的内存消耗计算公式	场景举例	内存消耗（单位为MB）
6MB + 输入码流缓存大小 * 2 + (解码图像帧存大小 + 视频解码图像Tmv缓存大小) * (参考帧数量 + 解码后缓存图像帧数 + 1)	● 输入码流格式H264 ● 输入码流分辨率1080P ● 输入码流缓存大小为4MB ● 解码图像帧存大小为3MB ● 视频解码图像Tmv缓存大小为0.5MB ● 解码后缓存图像帧数2个	52.5MB（参考帧数量8个） 31.5MB（参考帧数量2个）
	● 输入码流格式H265 ● 输入码流分辨率1080P ● 输入码流缓存大小为4MB ● 解码图像帧存大小为3MB ● 视频解码图像Tmv缓存大小为1MB ● 解码后缓存图像帧数2个	58MB（参考帧数量8个） 34MB（参考帧数量2个）

性能指标说明（Atlas A2 训练系列产品/Atlas 800I A2 推理产品）

表 1-41 Atlas A2 训练系列产品/Atlas 800I A2 推理产品的昇腾 AI 处理器性能指标参考数据

场景举例	总帧率
720p*n路(1≤n≤2)	n*480fps
720p*n路(n>2)	960fps
1080p*n路(1≤n≤2)	n*240fps
1080p*n路(n>2)	480fps
4k*n路(1≤n≤2)	n*60fps
4k*n路(n>2)	120fps

下表说明了典型场景下每路VDEC解码的内存消耗，在内存消耗的计算公式中：

- 输入码流缓存大小：等于输入码流的宽*高*2。
- 解码图像帧存大小：1080P分辨率的输入码流，该参数值为3MB。其它分辨率时参数值可等量折算。
- 视频解码图像Tmv缓存大小：H264格式、1080P分辨率的输入码流，该参数值为0.5MB；H265格式、1080P分辨率的输入码流，该参数值为1MB。其它分辨率时参数值可等量折算。
- 参考帧数量可由用户调用[aclvdecSetChannelDescRefFrameNum](#)接口设置。
- 解码后缓存图像帧数固定为2个。

每路VDEC解码的内存消耗计算公式	场景举例	内存消耗（单位为MB）
6MB + 输入码流缓存大小 * 2 + (解码图像帧存大小 + 视频解码图像Tmv缓存大小) * (参考帧数量 + 解码后缓存图像帧数 + 1)	• 输入码流格式H264 • 输入码流分辨率1080P • 输入码流缓存大小为4MB • 解码图像帧存大小为3MB • 视频解码图像Tmv缓存大小为0.5MB • 解码后缓存图像帧数2个	52.5MB（参考帧数量8个） 31.5MB（参考帧数量2个）
	• 输入码流格式H265 • 输入码流分辨率1080P • 输入码流缓存大小为4MB • 解码图像帧存大小为3MB • 视频解码图像Tmv缓存大小为1MB • 解码后缓存图像帧数2个	58MB（参考帧数量8个） 34MB（参考帧数量2个）

1.8.9.3 aclvdecSendFrame

函数功能

将待解码的输入内存和解码输出内存一起传到解码器进行解码，异步接口。

约束说明

- 该接口是异步接口，调用接口成功仅表示任务下发成功，不表示任务执行成功。调用该接口后，需调用同步等待接口（例如，[aclrtSynchronizeStream](#)）确保任务已执行完成，否则可能会导致训练或推理等业务异常、Device断链掉卡等未知情况。
- 发送数据前必须保证通道已经被创建，否则返回错误。
- 发送码流时须按帧发送，一次只发送完整的一帧码流。
- 不能发送eos为0的空码流包（码流长度为0或码流地址为空）。
- 发送完所有码流后，可以发送eos为1的帧，表示当前码流文件结束。发送eos后，本接口会等待已发送帧全部解码并且用户的回调函数处理完成后才返回。
- 由于码流异常、解码超时等原因，可能导致aclvdecSendFrame接口发送帧或发送eos失败，建议用户在编写代码时，获取该接口的返回码，当该接口调用失败时，进行异常处理。
- aclvdecSendFrame接口内部封装了[aclrtLaunchCallback](#)接口，用于在Stream的任务队列中增加一个需要执行的回调函数。用户在实现VDEC功能时，无需再单独调用[aclrtLaunchCallback](#)接口。

函数原型

```
aclError aclvdecSendFrame(aclvdecChannelDesc *channelDesc,  
aclvppStreamDesc *input,  
aclvppPicDesc *output,  
aclvdecFrameConfig *config,  
void* userData)
```

参数说明

参数名	输入/输出	说明
channelDesc	输入	通道描述信息的指针。 与调用aclvdecCreateChannel接口创建通道时指定的channelDesc保持一致。 调用aclvdecSetChannelDesc系列接口设置通道描述信息的属性，包括解码通道号、线程、回调函数、视频编码协议等。 说明 仅在Atlas 200/300/500 推理产品和Atlas 训练系列产品上，由于软件约束，不支持调用aclvdecSetChannelDescRefFrameNum接口设置通道描述信息的对应属性，若调用，则会返回报错。

参数名	输入/输出	说明
input	输入	输入码流描述信息的指针，输入内存用户需提前申请。 - 调用acldvppCreateStreamDesc接口创建码流描述信息，调用acldvppSetStreamDesc系列接口设置视频码流信息的属性（例如，码流数据的内存地址）。输入码流格式以通道描述信息中设置的格式为准，调用acldvppSetStreamDescFormat接口设置输入码流格式无效。 - 输入视频码流的分辨率、视频格式的要求，请参见1.8.9.1 功能及约束说明。
output	输入&输出	输出图片描述信息的指针，输出内存用户需提前申请。 output参数作为输入时，需要用户调用如下接口： - 调用acldvppCreatePicDesc接口创建图片描述信息。 - 调用acldvppSetPicDesc系列接口设置输出图片的数据内存、内存大小、宽、高、格式。 - 如果此处设置输出图片的宽高，则需要同时设置width、height、widthStride、heightStride这4个参数，且当width<16时，widthStride必须对齐到32，不能对齐到16；如果这4个参数都不设置，则以输入码流的宽高为准；如果设置部分参数，则接口返回错误。 - 如果此处设置输出图片的格式，则以此处为准；如果此处不设置，则以通道描述信息中设置的为准。 - 输出图像格式的要求，请参见1.8.9.1 功能及约束说明。 output参数作为输出时，用户需在回调函数中调用acldvppGetPicDesc系列接口获取解码后的输出图片数据。隔行码流场景下，隔行码流每帧发送两场，解码时其中一场无图像输出，属于正常现象，会返回ERR_DECODE_NOPIC = 0x20000错误码；隔行码流的解码输出数据都在奇数场对应的输出buffer中。
config	输入	解码配置的指针，预留，当前可填NULL。
userData	输入	用户自定义数据的指针。 如果用户需要获取解码的帧序号，则可以在userData参数处定义，然后解码的帧序号可以通过userData参数传递给VDEC的回调函数，用于确定回调函数中处理的是第几帧数据。

返回值说明

返回0表示成功，返回非0表示失败。

参考资源

接口调用流程及示例，参见VDEC视频解码。

1.8.9.4 aclvdecSendSkippedFrame

函数功能

如果不想获取某一帧的解码结果，可以调用本接口，将待解码的码流（输入内存）传到解码器进行解码，此时，解码结果最终不会输出，解码完成的回调函数中返回的output为nullptr。异步接口。

约束说明

- 该接口是异步接口，调用接口成功仅表示任务下发成功，不表示任务执行成功。调用该接口后，需调用同步等待接口（例如，[aclrtSynchronizeStream](#)）确保任务已执行完成，否则可能会导致训练或推理等业务异常、Device断链掉卡等未知情况。
- 发送数据前必须保证通道已经被创建，否则返回错误。
- 发送码流时须按帧发送，一次只发送完整的一帧码流。
- 不能发送eos为0的空码流包（码流长度为0或码流地址为空）。
- 不可以发送eos为1的帧。
- 由于码流异常、解码超时等原因，可能导致aclvdecSendFrame接口发送帧或发送eos失败，建议用户在编写代码时，获取该接口的返回码，当该接口调用失败时，进行异常处理。
- aclvdecSendSkippedFrame接口内部封装了[aclrtLaunchCallback](#)接口，用于在Stream的任务队列中增加一个需要执行的回调函数。用户在实现VDEC功能时，无需再单独调用[aclrtLaunchCallback](#)接口。

函数原型

```
aclError aclvdecSendSkippedFrame(aclvdecChannelDesc *channelDesc,  
acldvppStreamDesc *input,  
aclvdecFrameConfig *config,  
void *userData)
```

参数说明

参数名	输入/输出	说明
channelDesc	输入	通道描述信息的指针。 与调用 aclvdecCreateChannel 接口创建通道时指定的channelDesc保持一致。 调用 aclvdecSetChannelDesc系列接口 设置通道描述信息的属性，包括解码通道号、线程、回调函数、视频编码协议等。 说明 仅在Atlas 200/300/500 推理产品和Atlas 训练系列产品上，由于软件约束，不支持调用 aclvdecSetChannelDescRefFrameNum 接口设置通道描述信息的对应属性，若调用，则会返回报错。
input	输入	输入码流描述信息的指针，输入内存用户需提前申请。 - 调用acldvppCreateStreamDesc接口创建码流描述信息，调用acldvppSetStreamDesc系列接口设置视频码流信息的属性（例如，码流格式）。输入码流格式以通道描述信息中设置的格式为准，调用acldvppSetStreamDescFormat接口设置输入码流格式无效。 - 输入视频码流的分辨率、视频格式的要求，请参见1.8.9.1 功能及约束说明。
config	输入	解码配置的指针，预留，当前可填NULL。
userData	输入	用户自定义数据的指针。 如果用户需要获取解码的帧序号，则可以在userData参数处定义，然后解码的帧序号可以通过userData参数传递给VDEC的回调函数，用于确定回调函数中处理的是第几帧数据。

返回值说明

返回0表示成功，返回非0表示失败。

1.8.9.5 aclvdecCallback

函数功能

视频解码回调函数，同步接口。用户需自定义该回调函数。

约束说明

- 在回调函数中不能执行销毁通道操作，否则会导致程序执行死锁。
- 回调函数处理的时延应满足发帧帧率要求，否则会影响[aclvdecSendFrame](#)接口进行视频帧处理的实时性。

- 注销回调线程要在所有Callback执行完成后（即注销送码流线程之后）。
- 当使用了抽帧功能后（调用[aclvdecSendSkippedFrame](#)接口），回调相应帧的output为nullptr。
对于抽帧的情况，输出Buffer无解码结果，且**抽掉的帧**按照顺序输出。
对于不抽帧的情况，输出Buffer中才会有解码结果，且**不抽的帧**按照顺序输出。
如果解码码流中同时存在抽帧、不抽帧的情况，则系统不保证抽掉的帧与不抽的帧之间按照顺序输出。

函数原型

void (* aclvdecCallback) (aclvppStreamDesc * input, aclvppPicDesc * output, void* userData)

参数说明

参数名	输入/输出	说明
input	输入	输入码流描述信息的指针。与 aclvdecSendFrame 接口中的input一致。
output	输入	VDEC解码后的输出图片描述信息的指针，输出内存用户需提前申请。
userData	输入	用户自定义数据的指针。

返回值说明

无

参考资源

接口调用流程及示例，参见VDEC视频解码。

1.8.10 VENC 功能

1.8.10.1 功能及约束说明

Atlas 训练系列产品上，不支持该功能。
Atlas A2训练系列产品/Atlas 800I A2推理产品上，不支持该功能。

功能说明

VENC (Video Encoder) 将YUV420SP NV12/NV21-8bit图片数据编码成H264/H265格式的视频码流。

在Atlas 200/300/500 推理产品上，不支持单进程多线程场景。

分辨率约束

- 输入图片分辨率：

版本	分辨率范围
Atlas 200/300/500 推理产品	最大分辨率1920*1920，最小分辨率128*128。
Atlas 推理系列产品	宽和高的范围都是128~4096，对于H264格式的码流，最大分辨率的乘积不超过4096*2304。
Atlas 200/500 A2推理产品	最大分辨率8192*8192，最小分辨率114*114。
- 输出码流分辨率
VENC只对图片编码，不会改变图片分辨率，因此输出与输入的图片分辨率保持一致。

码流/图片格式、宽高对齐、内存约束

实现VENC视频编码功能时：

- 输入内存
需调用[acldvppMalloc](#)接口申请Device上的输入内存，调用[acldvppFree](#)接口释放输入内存，这部分内存的生命周期由用户自行管理。内存大小参见下表中的计算公式。
- 输出内存
不需要用户管理输出内存，由系统管理内存。

说明

输入图片格式的定义请参见[acldvppPixelFormat](#)，宽stride、高stride等概念请参见[1.8.2 基本概念](#)。

表 1-42 图片格式、宽高对齐、内存大小约束

输入图片格式	输入图片宽、高对齐要求	输入图片宽stride、高stride、内存大小要求	输出码流格式
YUV420SP NV12 8bit	宽2对齐 高2对齐	宽stride为宽16对齐后的值。 高stride无需设置。 内存大小（单位Byte）= 宽stride * 高 * 3/2	- H264 BP/MP/HP - H265 MP（仅支持Slice码流）
YUV420SP NV21 8bit			

1.8.10.2 性能指标说明

- Atlas 训练系列产品上，不支持该功能。
- Atlas A2训练系列产品/Atlas 800I A2推理产品上，不支持该功能。

本节仅描述各产品对应的昇腾AI处理器的性能指标参考数据，各产品由于其内存、CPU等配置不同，在性能规格上会存在差异，详细产品的性能规格请查阅[昇腾社区](#)的官方说明。

本节以单个昇腾AI处理器为例，描述基本场景性能指标参考，如下表所示，其它分辨率可以等量估算。

下表中的1路对应一个通道，一个通道对应一个线程；各分辨率的含义为：720p指分辨率为1280*720的图片，1080p指分辨率为1920*1080的图片，4K指分辨率为3840*2160的图片。

性能指标说明（Atlas 200/300/500 推理产品）

表 1-43 Atlas 200/300/500 推理产品的昇腾 AI 处理器性能指标参考数据

场景举例	总帧率
1080p*n路（一个进程对应一路）	30fps

性能指标说明（Atlas 推理系列产品）

以下性能数据，基于以下Host服务器配置：CPU主频为Intel(R) Xeon(R) CPU E5-2680 v4 @ 2.40GHz，CPU核心数为14。

表 1-44 Atlas 推理系列产品的昇腾 AI 处理器性能指标参考数据

场景举例	总帧率
720p*n路(1≤n≤3)	n*480fps
720p*n路(n>3)	1440fps
1080p*n路(1≤n≤3)	n*240fps
1080p*n路(n>3)	720fps
4k*n路(1≤n≤3)	n*60fps
4k*n路(n>3)	180fps

性能指标说明（Atlas 200/500 A2 推理产品）

表 1-45 Atlas 200/500 A2 推理产品的昇腾 AI 处理器性能指标参考数据

场景举例	总帧率
720p*n路(1≤n≤3)	n*400fps
720p*n路(n>3)	1200fps
1080p*n路(1≤n≤3)	n*200fps

场景举例	总帧率
1080p*n路(n>3)	600fps
4k*n路(1≤n≤3)	n*50fps
4k*n路(n>3)	150fps

1.8.10.3 aclvencSendFrame

Atlas 训练系列产品，不支持该接口。
Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

将待编码的图片传到编码器进行编码。异步接口。

约束说明

- 该接口是异步接口，调用接口成功仅表示任务下发成功，不表示任务执行成功。调用该接口后，需调用同步等待接口（例如，[aclrtSynchronizeStream](#)）确保任务已执行完成，否则可能会导致训练或推理等业务异常、Device断链掉卡等未知情况。
- 发送数据前必须保证通道已经被创建，否则返回错误。
- 不能发送eos为0的空码流包（码流长度为0或码流地址为空）。
- 结束视频编码时可以发送eos为1的空图片，表示当前编码结束。
- 编码时，输入内存需要在[aclvencCallback](#)回调函数内获取结果之后才能进行释放。

函数原型

```
aclError aclvencSendFrame(aclvencChannelDesc *channelDesc,  
aclvppPicDesc *input,  
void *reserve,  
aclvencFrameConfig *config,  
void *userdata)
```

参数说明

参数名	输入/输出	说明
channelDesc	输入	通道描述信息的指针。 与调用 aclvencCreateChannel 接口创建通道时指定的channelDesc保持一致。 在通道描述信息中指定视频编码的回调函数。

参数名	输入/输出	说明
input	输入	输入图片描述信息的指针，输入内存用户需提前申请。 - 调用acldvppCreatePicDesc接口创建图片描述信息，调用acldvppSetPicDesc系列接口设置图片描述参数（例如，宽、高等）。输入图片格式以通道描述信息中设置的格式为准，调用acldvppSetPicDescFormat接口设置输入图片格式无效。 - 输入图片的分辨率、格式的要求，请参见1.8.10.1 功能及约束说明。
reserve	输入	预留参数，暂不使用。
config	输入	单帧配置数据的指针。
userdata	输入	用户自定义数据的指针。

返回值说明

返回0表示成功，返回非0表示失败。

参考资源

接口调用流程及示例，参见VENC视频编码。

1.8.10.4 aclvencCallback

Atlas 训练系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

视频编码回调函数，同步接口。用户需自定义该回调函数。

约束说明

- 在回调函数中不能执行销毁通道操作，否则会导致程序执行死锁。
- 回调函数处理的时延应满足发帧帧率要求，否则会影响[aclvencSendFrame](#)接口进行视频帧处理的实时性。
- 注销回调线程要在所有Callback执行完成后。

函数原型

```
void (*aclvencCallback)(acldvppPicDesc *input, acldvppStreamDesc *output, void *userdata)
```


参数说明

参数名	输入/输出	说明
input	输入	输入图片描述信息的指针。
output	输入	VENC编码后的输出帧码流描述信息的指针。
userdata	输入	用户自定义数据的指针。

返回值说明

无

参考资源

接口调用流程及示例，参见VENC视频编码。

1.9 媒体数据处理 V2

1.9.1 总体功能及约束说明

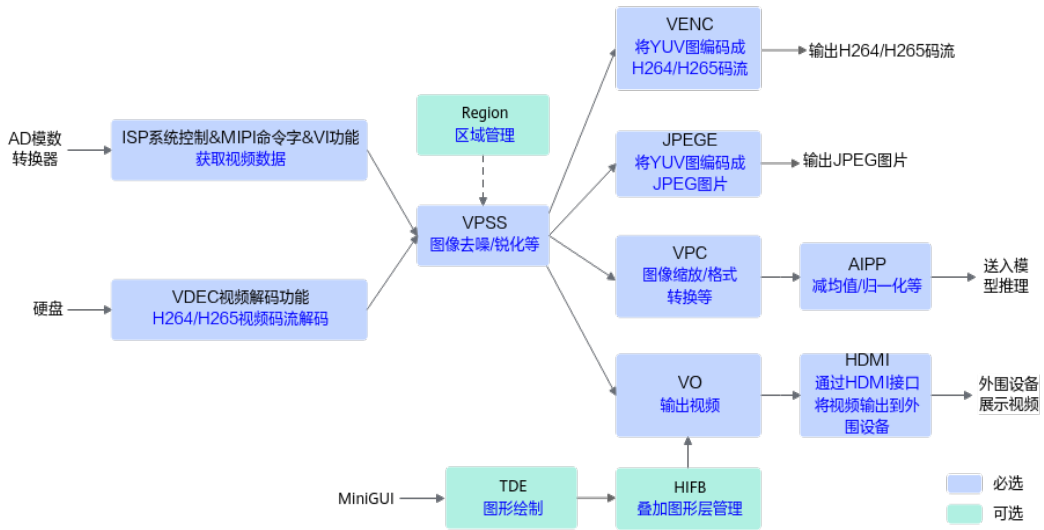
多版本接口差异

本手册中媒体数据处理V1版本与媒体数据处理V2版本的接口都是描述处理媒体数据的接口，用于实现视频编解码、图像编解码、图像处理等功能，但两套接口不能混用。

- V2版本的功能比V1版本更多，如下：
 - JPEGE：V2版本接口支持高级的参数配置，如huffman表配置。
 - VENC：V2版本接口支持更加细化的码控参数配置和效果调优，如I/P帧QP、宏块码控等。
 - VDEC：V2版本接口支持更细化的内存控制，如设置输入码流缓存。
 - 视频数据获取功能（ISP系统控制&MIPI命令字&VI功能）：仅V2版本接口支持。
 - VPSS视频处理：仅V2版本接口支持。
 - 音频相关功能，包括录音、播音、音量调节：仅V2版本接口支持。
 - 视频数据展示功能（VO功能&HDMI外设）：仅V2版本接口支持。
- 建议使用V2版本中的接口，保证后续版本接口功能以及业务的连续演进。
- V1版本中的接口是为了兼容旧版本，保证使用该部分接口的用户能继续使用，后续版本不再演进。

图像/视频/音频数据处理的典型功能介绍

图 1-10 图像/视频数据处理



各功能的介绍如下表所示，昇腾AI处理器对这些功能的支持度请参见各版本功能支持度说明（AIPP当前各版本均支持）。

功能	子功能模块	描述
获取 视频 数据	ISP（Image Signal Processing）系统控制	系统控制部分用于注册3A算法、注册Sensor驱动、初始化ISP firmware、运行ISP firmware、退出ISP firmware、配置ISP属性等功能。
	MIPI Rx ioctl命令字	MIPI Rx是一个支持多种差分视频输入接口的采集单元，通过combo-PHY接收MIPI/LVDS/sub-LVDS/HiSpi接口的数据，通过不同的功能模式配置，MIPI Rx可以支持多种速度和分辨率的数据传输需求，支持多种外部输入设备。
	VI（Video Input）	VI模块捕获视频图像，可对其做裁剪、防抖、颜色优化、亮度优化、噪声去除等处理，并输出YUV或RAW格式的图像数据。
展示 视频 数据	VO（Video Output）	VO模块接收VPSS处理后的输出图像，可进行播放控制等处理，最后按用户配置的输出协议（当前仅支持HDMI）输出给外围视频设备。 VO可配合TDE（Two Dimensional Engine）模块、HIFB（Hisilicon Framebuffer）模块，利用硬件分别进行图形绘制、叠加图形层管理。
	HDMI（High Definition Multimedia Interfac）	HDMI是全数字化影像和声音发送接口，可以发送未压缩的音频及视频信号。

功能	子功能模块	描述
	TDE (Two Dimensional Engine)	TDE是图形二维加速引擎，它利用硬件为 OSD (On Screen Display) 和 GUI (Graphics User Interface) 提供快速的图形绘制功能，主要有快速拷贝、快速色彩填充、模式填充 (当前仅支持Alpha Blending操作) 。
	HIFB (Hisilicon Framebuffer)	HIFB用于管理叠加图形层，它不仅提供Linux Framebuffer的基本功能，还在Linux Framebuffer的基础上增加图层显示起始位置修改、层间Alpha等扩展功能。
区域管理	Region	叠加在视频上的OSD (On Screen Display)和遮挡在视频上的色块统称为区域。 区域管理模块 ，用于统一管理这些区域资源，用于在视频上显示一些特定信息 (如通道号、时间戳等)、或在视频中填充色块用于遮挡，当前该功能需配合VPSS一起使用。
图像/视频数据处理	VPSS (Video Process Sub-System)	VPSS模块支持对输入图像进行统一预处理，如去噪、去隔行、裁剪等，然后再对各通道分别进行处理，如缩放、加边框等。
	AIPP (Artificial Intelligence Pre-Processing)	AIPP人工智能预处理，在AI Core上完成数据预处理，主要功能包括改变图像尺寸 (抠图、填充等)、色域转换 (转换图像格式)、减均值/乘系数 (改变图像像素) 等。 AIPP区分为静态AIPP和动态AIPP。您只能选择静态AIPP或动态AIPP中的一种来处理图片，不能同时配置静态AIPP和动态AIPP两种方式。 ● 静态AIPP：模型转换时设置AIPP模式为静态，同时设置AIPP参数，模型生成后，AIPP参数值被保存在离线模型 (*.om) 中，每次模型推理过程采用固定的AIPP预处理参数 (无法修改) 。 如果使用静态AIPP方式，多Batch情况下共用同一份AIPP参数，AIPP参数值在使用ATC工具进行模型转换时设置，ATC工具的详细说明请《 ATC工具使用指南 》。 ● 动态AIPP：模型转换时仅设置AIPP模式为动态，每次模型推理前，根据需求，在执行模型前设置动态AIPP参数值，然后在模型执行时可使用不同的AIPP参数。如果使用动态AIPP方式，多Batch可使用不同的AIPP参数，各Batch所使用的AIPP参数值通过AscendCL提供的接口来设置，请参见模型动态AIPP推理中的介绍。

功能	子功能模块	描述
	DVPP (Digital Vision Pre-Processing)	DVPP是昇腾AI处理器内置的图像处理单元，通过AscendCL媒体数据处理接口提供强大的媒体处理硬加速能力，主要功能包括以下功能： • VPC (Vision Preprocessing Core)：处理YUV、RGB等格式的图片，包括缩放、抠图、图像金字塔、色域转换等。 • JPEGD (JPEG Decoder)：JPEG压缩格式-->YUV格式的图片解码。 • JPEGE (JPEG Encoder)：YUV格式-->JPEG压缩格式的图片编码。 • VDEC (Video Decoder)：H264/H265格式-->YUV/RGB格式的视频码流解码。 • VENC (Video Encoder)：YUV420SP格式-->H264/H265格式的视频码流编码。 • PNGD (PNG Decoder)：PNG格式-->RGB格式的图片解码。 说明 AIPP、DVPP可以分开独立使用，也可以组合使用。组合使用场景下，一般先使用DVPP对图片/视频进行解码、抠图、缩放等基本处理，但由于DVPP硬件上的约束，DVPP处理后的图片格式、分辨率有可能不满足模型的要求，因此还需要再经过AIPP进一步做色域转换、抠图、填充等处理。 例如，在Atlas 200/300/500 推理产品和Atlas 训练系列产品上，由于DVPP视频解码仅支持输出YUV格式的图片，如果模型需要RGB格式的图片，则需要再经过AIPP做色域转换的处理。
音频数据获取和输出	AI (Audio Input)	AI模块捕获音频数据 。
	AO (Audio Output)	通过ADEC模块解码后的音频数据，AO模块支持播放音频。
音频数据编解码	AENC (Audio Encoder)	通过AI模块获取的音频数据，AENC模块支持对其进行编码，输出音频码流。
	ADEC (Audio Decoder)	ADEC支持解码G.711a、G.711u等协议的音频码流，再通过AO模块播放音频。

功能支持度说明

昇腾AI处理器对媒体数据处理V2版本各功能的支持度如下表所示。

各标识的含义如下：

- √: 支持。
- x: 不支持。

昇腾AI 处理器	V P C	JPE GD	JP E GE	P N G D	V D EC	VE NC	视频 数据 获取	VPSS 视频 处理	音频功 能（录 音/播 音/ 音量 调 节）	视频 数据 展 示	区 域 管 理
Atlas 200/300 /500 推 理产品	x	x	x	x	x	x	x	x	x	x	x
Atlas 训 练系列 产品	x	x	x	x	x	x	x	x	x	x	x
Atlas 推 理系列 产品	√	√	√	√	√	√	x	x	x	x	x
Atlas 200/500 A2推理 产品	√	√	√	√	√	√	√	√	√	√	√
Atlas A2训练 系列产 品/Atlas 800I A2 推理产 品	√	√	√	√	√	x	x	x	x	x	x

整体约束说明

使用本章中介绍的接口，有以下注意点：

- 关于内存申请/释放：
 - 实现媒体数据处理的VPC、JPEGD、JPEGE等功能前，若需要申请内存存放输入或输出数据，需调用[hi_mpi_dvpp_malloc](#)申请内存、调用[hi_mpi_dvpp_free](#)接口释放内存。如果多个功能串联使用的场景，需要复用同一段内存，则按最大内存要求来申请内存。
 - 调用[a](#)申请出来的内存可以满足媒体数据处理的要求，也可以在其它任务中使用，例如，从性能角度，为了减少拷贝，媒体数据处理的输出作为模型推理的输入，实现内存复用。
 - 但由于媒体数据处理访问的地址空间有限，为确保媒体数据处理时内存足够，除媒体数据处理功能外的其它功能（例如，模型加载），建议调用[内存管理](#)章节下的[aclrtMalloc](#)接口、或[aclrtMallocHost](#)接口、或[aclrtMallocCached](#)接口申请内存。
针对Atlas 推理系列产品：媒体数据处理功能在每个进程内，可访问的地址空间最大为16GB。

针对Atlas A2训练系列产品/Atlas 800I A2推理产品：媒体数据处理功能在每个进程内，可访问的地址空间最大为16GB。

- 关于通道的要求：
实现媒体数据处理的各功能前，必须调用接口创建对应功能的通道，请分别参见[1.9.16 VPC图像处理功能](#)、[1.9.17 VDEC视频解码功能/JPEGD图片解码功能](#)、[1.9.18 VENC视频编码功能/JPEGE图片编码功能](#)、[1.9.19 PNGD图片解码功能](#)章节下的通道创建与销毁接口，查看接口说明以及通道数的最大限制。
通道的创建与销毁会涉及资源的申请与释放，反复创建与销毁通道会影响业务性能，因此建议根据实际场景管理通道，例如，如果有持续VPC图片处理，则创建VPC的通道后，等到所有VPC功能调用完成后，再销毁该VPC通道。
通道数量多，会影响Device的CPU占用率和内存占用，通道数量建议参考各功能章节下的性能指标的路数。
- 本章节描述结构体、枚举值等，预留的字段需要手动设置为0，避免后续版本的兼容性问题。
结构体或枚举值中的预留字段，其名称中包含*_BUTT字符串的，例如HI_COMPRESS_MODE_BUTT。

1.9.2 基本概念

表 1-46 概念解释

概念	描述
宽stride (widthStride)	指一行图像跨距，表示输入图片对齐后的宽，RGB格式和YUV格式的宽stride计算方式不一样。 各功能对宽stride的要求不同，请参见对应功能的约束说明： ● VPC功能，请参见图片格式、宽高对齐、内存约束。 ● JPEGD功能，请参见1.9.17.1 JPEGD功能及约束说明。 ● JPEGE功能，请参见1.9.18.1 JPEGE功能及约束说明。 ● PNGD功能，请参见1.9.19.1 功能及约束说明。 ● VDEC功能，请参见1.9.17.3 VDEC功能及约束说明。 ● VENC功能，请参见1.9.18.3 VENC功能及约束说明。
高stride (heightStride)	指图像在内存中的行数，表示输入图片对齐后的高。 各功能对高stride的要求不同，请参见对应功能的约束说明： ● VPC功能，请参见图片格式、宽高对齐、内存约束。 ● JPEGD功能，请参见1.9.17.1 JPEGD功能及约束说明。 ● JPEGE功能，请参见1.9.18.1 JPEGE功能及约束说明。 ● PNGD功能，请参见1.9.19.1 功能及约束说明。 ● VDEC功能，请参见1.9.17.3 VDEC功能及约束说明。 ● VENC功能，请参见1.9.18.3 VENC功能及约束说明。
抠图区域	指用户指定的需抠出的图片区域。抠图起始坐标无奇数、偶数限制。 抠图区域最小分辨率为10*6，最大分辨率为8192*8192。 抠图区域的约束请参见 抠图、贴图约束 。

概念	描述
贴图区域	指在输出图片中用户指定的区域。 贴图区域最小分辨率为10*6，最大分辨率为4096*4096。 贴图区域的约束请参见 抠图 、 贴图约束 。
上/左偏移	通过配置上偏移、左偏移、抠图/贴图区域的宽和高可以指定抠图区域或贴图区域的位置。

1.9.3 公共接口

1.9.3.1 hi_mpi_sys_init

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

函数功能

用户处理图像、视频等媒体数据之前，需先调用本接口对媒体数据处理系统底层进行初始化。

约束说明

- 需在申请AscendCL运行管理资源后调用本接口。AscendCL运行管理资源申请流程请参见运行管理资源申请与释放。
- 多个进程同时运行情况下，每个进程都必须调用本接口初始化。
- 一个进程内，可以调用本接口反复初始化，不返回失败。
- 多Device场景下，针对每一个Device，都需要调用hi_mpi_sys_init接口进行初始化。

函数原型

hi_s32 hi_mpi_sys_init(**hi_void**)

参数说明

无

返回值说明

- 0：成功
- 非0：失败，参见[1.9.21.1 公共返回码](#)

参考资源

接口调用流程及示例，参见DVPP图像/视频处理（媒体数据处理V2）。

1.9.3.2 hi_mpi_sys_full_init

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

Atlas 200/500 A2推理产品，不支持使用该接口。

函数功能

环视拼接3秒快速出图显示场景下，需调用该接口全量初始化VPC、VENC硬件资源。

约束说明

- 需在hi_mpi_sys_init接口之后调用本接口，否则会返回报错；
- 本接口只需调用一次。

函数原型

hi_s32 hi_mpi_sys_full_init(**hi_void**)

参数说明

无

返回值说明

- 0：成功
- 非0：失败，参见[1.9.21.1 公共返回码](#)

1.9.3.3 hi_mpi_sys_exit

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

函数功能

用户处理完图像、视频等媒体数据之后，需先调用本接口对媒体数据处理系统底层进行去初始化。

约束说明

- 可以调用本接口反复去初始化，不返回失败。
- 调用本接口去初始化之后，会释放系统内部媒体数据处理的资源，因此不能再调用媒体数据处理V2章节下的其它接口（除[hi_mpi_sys_init](#)接口外），否则可能出现接口调用异常、返回码错误等情况。
- 需在释放AscendCL运行管理资源前调用本接口。AscendCL运行管理资源申请流程请参见运行管理资源申请与释放。

函数原型

hi_s32 hi_mpi_sys_exit(**hi_void**)

参数说明

无

返回值说明

- 0: 成功
- 非0: 失败, 参见[1.9.21.1 公共返回码](#)

参考资源

接口调用流程及示例, 参见DVPP图像/视频处理 (媒体数据处理V2)。

1.9.3.4 hi_mpi_dvpp_malloc

Atlas 200/300/500 推理产品, 不支持该接口。

Atlas 训练系列产品, 不支持该接口。

函数功能

申请Device上的内存, 申请的内存满足媒体数据处理的要求 (例如, 内存首地址128字节对齐)。

约束说明

- 调用该接口申请内存后, 这部分内存的生命周期由用户自行管理, 如果不使用, 必须调用[hi_mpi_dvpp_free](#)接口及时释放内存。
- 通过该接口申请的内存仅支持在单个进程内使用, 不能在多个进程之间共享。
- 调用hi_mpi_dvpp_malloc接口申请内存时, 会对用户输入的size按向上对齐成32整数倍后, 再多加32字节。
- 媒体数据处理的输出作为模型推理的输入时, 若用户使用本接口申请大块内存并自行划分、管理内存时, 每段内存需同时满足以下要求:
 - 内存大小向上对齐成32整数倍+32字节 ($m = \text{ALIGN\_UP}[\text{len}, 32] + 32$ 字节);
 - 内存起始地址需满足128字节对齐 ($\text{ALIGN\_UP}[m, 128]$)。

说明

len表示某段内存的大小, $\text{ALIGN\_UP}[\text{len}, k]$ 表示向上按k字节对齐: $((\text{len}-1)/k+1)*k$ 。

函数原型

hi_s32 hi_mpi_dvpp_malloc(**hi_u32** dev_id, **hi_void** **dev_ptr, **hi_u64** size)

参数说明

参数名	输入/输出	说明
dev_id	输入	Device ID，预留参数，该参数配置无效。 标准形态下，在Host上调用本接口时，DVPP内部根据AscendCL接口 aclrtSetDevice 的device申请内存。
dev_ptr	输出	“Device上已分配内存的指针”的指针。
size	输入	申请内存的大小，单位Byte。

返回值说明

- 0：成功
- 非0：失败

参考资源

接口调用流程及示例，参见DVPP图像/视频处理（媒体数据处理V2）。

1.9.3.5 hi_mpi_dvpp_free

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

函数功能

释放Device上的内存。

hi_mpi_dvpp_free接口只能释放通过[hi_mpi_dvpp_malloc](#)接口申请的内存。

函数原型

[hi_s32](#) hi_mpi_dvpp_free([hi_void](#) *dev_ptr)

参数说明

参数名	输入/输出	说明
dev_ptr	输入	待释放内存的指针。

返回值说明

- 0：成功
- 非0：失败

参考资源

接口调用流程及示例，参见DVPP图像/视频处理（媒体数据处理V2）。

1.9.3.6 hi_mpi_dvpp_get_image_info

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

函数功能

根据输入码流，获取按源图格式输出时的图片宽、高、宽stride、高stride、解码后占用的内存大小、源图片格式等信息。该接口会检查图片格式是否支持解码，如果遇到不支持的码流则该接口会返回HI_ERR_VDEC_NOT_SUPPORT错误码，该错误码的详细描述请参见1.9.21.11 VDEC视频解码/JPEGD图片解码返回码。

函数原型

```
hi_s32 hi_mpi_dvpp_get_image_info(hi_payload_type img_type, const
hi_vdec_stream *stream, hi_img_info *img_info);
```

参数说明

参数名	输入/输出	说明
img_type	输入	要解析的码流类型。 当前仅支持码流类型为HI_PT_JPEG。
stream	输入	输入码流信息的指针。 Atlas 推理系列产品，标准形态下，在Host上调用本接口时，该结构体内的addr参数配置的地址必须是Host上的内存地址。 Atlas A2训练系列产品/Atlas 800I A2推理产品，在Host上调用本接口时，该结构体内的addr参数配置的地址必须是Host上的内存地址。 Atlas 200/500 A2推理产品，EP模式下，在Host上调用本接口时，该结构体内的addr参数配置的地址必须是Host上的内存地址。 Atlas 200/500 A2推理产品，RC模式下，在Device上调用本接口时，该结构体内的addr参数配置的地址必须是Device上的内存地址。
img_info	输出	图片信息的指针，包含图片宽、高、宽stride、高stride、解码后占用的内存大小、源图片格式等信息。 其中，源图格式通过pixel_format参数返回。

返回值说明

- 0：成功
- 非0：失败，参见[1.9.21.11 VDEC视频解码/JPEGD图片解码返回码](#)。

1.9.3.7 hi_mpi_dvpp_get_version

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

函数功能

查询接口版本号，版本号命名采用：A.B.C模式，其中，A表示有不兼容修改，B表示新增接口，C表示bug修复。

函数原型

```
hi_s32 hi_mpi_dvpp_get_version(hi_s32 *major_version, hi_s32 *minor_version, hi_s32 *patch_version)
```

参数说明

参数名	输入/输出	说明
major_version	输出	主版本号指针，从1开始，如果出现接口的不兼容变更时，加1。
minor_version	输出	次版本号指针，从0开始，按照迭代周期，有新增接口时加1。
patch_version	输出	补丁版本号指针，从0开始，表示本版本仅解决了问题，在major_version、minor_version不变的情况下加1；但major_version、minor_version增加的时候，patch_version一般为0。

返回值说明

- 0：成功
- 非0：失败，参见[1.9.21.1 公共返回码](#)

1.9.3.8 hi_mpi_sys_create_epoll

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

函数功能

创建新的媒体数据处理的Epoll实例，使用Epoll编程模型操作媒体数据处理通道的输入/输出数据。

约束说明

- [hi_mpi_sys_create_epoll](#)、[hi_mpi_sys_ctl_epoll](#)、[hi_mpi_sys_wait_epoll](#)、[hi_mpi_sys_close_epoll](#)这几个接口要配合使用，实现使用Epoll编程模型操作媒体数据处理通道的输入/输出数据。
- Atlas 推理系列产品，标准形态下，仅能使用AscendCL提供的Epoll接口（例如，[hi_mpi_sys_create_epoll](#)），不能使用Linux操作系统的Epoll函数。
- Atlas A2训练系列产品/Atlas 800I A2推理产品，仅能使用AscendCL提供的Epoll接口（例如，[hi_mpi_sys_create_epoll](#)），不能使用Linux操作系统的Epoll函数。
- Atlas 200/500 A2推理产品，仅能使用AscendCL提供的Epoll接口（例如，[hi_mpi_sys_create_epoll](#)），不能使用Linux操作系统的Epoll函数。
- VPC功能不支持使用该接口。

函数原型

[hi_s32](#) hi_mpi_sys_create_epoll([hi_s32](#) size, [hi_s32](#) *epoll_fd)

参数说明

参数名	输入/输出	说明
size	输入	Epoll实例最大可处理的DVPP通道文件句柄个数，当前该字段为预留，传入一个正数即可。
epoll_fd	输出	Epoll实例句柄的指针。句柄个数受限Linux操作系统的open files（一个任务最多可以同时打开的文件个数）资源限制，您可以使用ulimit -al命令查看open files。

返回值说明

- 0：成功
- 非0：失败，参见[1.9.21.1 公共返回码](#)

参考资源

接口调用流程及示例，参见JPEG图片编码、VENC视频编码。

1.9.3.9 hi_mpi_sys_ctl_epoll

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

函数功能

在媒体数据处理Epoll实例中，对对应的媒体数据处理通道进行控制操作（增、删、改）。

约束说明

- [hi_mpi_sys_create_epoll](#)、[hi_mpi_sys_ctl_epoll](#)、[hi_mpi_sys_wait_epoll](#)、[hi_mpi_sys_close_epoll](#)这几个接口要配合使用，实现使用epoll编程模型操作媒体数据处理通道的输入/输出数据。
- VPC功能不支持使用该接口。

函数原型

[hi_s32](#) [hi_mpi_sys_ctl_epoll](#)([hi_s32](#) [epoll_fd](#), [hi_s32](#) [operation](#), [hi_s32](#) [fd](#), [hi_dvpp_epoll_event](#) *[event](#))

参数说明

参数名	输入/输出	说明
epoll_fd	输入	Epoll实例句柄，句柄需提前通过 hi_mpi_sys_create_epoll 接口创建。
operation	输入	操作类型（增、删、改），参见 hi_dvpp_epoll_ctl_op 。
fd	输入	媒体数据处理通道的文件句柄。VENC场景下，您可以调用 hi_mpi_venc_get_fd 接口提前获取该文件句柄。VDEC场景下，您可以调用 hi_mpi_vdec_get_fd 接口提前获取该文件句柄。
event	输入	Epoll事件信息的指针。 该参数用于描述对应媒体数据处理通道事件类型以及关联的用户数据。进行删除操作时，该字段不用，可直接传入NULL。

返回值说明

- 0：成功
- 非0：失败，参见[1.9.21.1 公共返回码](#)

参考资源

接口调用流程及示例，参见JPEGE图片编码、VENC视频编码。

1.9.3.10 hi_mpi_sys_wait_epoll

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

函数功能

使用Epoll模型进行媒体数据处理通道输入输出操作，等待媒体数据处理Epoll实例产生事件。

约束说明

- [hi_mpi_sys_create_epoll](#)、[hi_mpi_sys_ctl_epoll](#)、[hi_mpi_sys_wait_epoll](#)、[hi_mpi_sys_close_epoll](#)这几个接口要配合使用，实现使用Epoll编程模型操作媒体数据处理通道的输入/输出数据。
- VPC功能不支持使用该接口。

函数原型

[hi_s32](#) [hi_mpi_sys_wait_epoll](#)([hi_s32](#) [epoll_fd](#), [hi_dvpp_epoll_event](#) *[events](#), [hi_s32](#) [max_events](#), [hi_s32](#) [timeout](#), [hi_s32](#) *[event_num](#))

参数说明

参数名	输入/输出	说明
epoll_fd	输入	媒体数据处理Epoll实例句柄的指针。
events	输出	hi_dvpp_epoll_event 数组，保存产生的事件信息。
max_events	输入	最大可返回的事件数，不应超过 events 数组的元素个数。
timeout	输入	超时时间，单位是毫秒。 • -1：永久阻塞 • 0：立即返回 • >0：配置具体的超时时间
event_num	输出	Epoll实际产生的事件数的指针。

返回值说明

- 0：成功
- 非0：失败，参见[1.9.21.1 公共返回码](#)

参考资源

接口调用流程及示例，参见JPEGE图片编码、VENC视频编码。

1.9.3.11 [hi_mpi_sys_close_epoll](#)

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

函数功能

使用Epoll模型进行媒体数据处理通道输入输出操作，销毁媒体数据处理Epoll实例。

约束说明

- [hi_mpi_sys_create_epoll](#)、[hi_mpi_sys_ctl_epoll](#)、[hi_mpi_sys_wait_epoll](#)、[hi_mpi_sys_close_epoll](#)这几个接口要配合使用，实现使用epoll编程模型操作媒体数据处理通道的输入/输出数据。
- VPC功能不支持使用该接口。

函数原型

[hi_s32](#) hi_mpi_sys_close_epoll([hi_s32](#) epoll_fd)

参数说明

参数名	输入/输出	说明
epoll_fd	输入	待销毁媒体数据处理Epoll实例句柄，句柄通过 hi_mpi_sys_create_epoll 接口创建。

返回值说明

- 0：成功
- 非0：失败，参见[1.9.21.1 公共返回码](#)

参考资源

接口调用流程及示例，参见JPEGE图片编码、VENC视频编码。

1.9.3.12 hi_mpi_sys_set_chn_csc_matrix

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

函数功能

在处理图像或视频前，YUV与RGB之间格式转换场景下，如果需要调整色域转换参数时，在创建对应图像或视频处理通道后，可调用该接口设置色域转换参数。

约束说明

如果不调用本接口设置色域转换参数，默认使用[BT601 wide标准的色域转换矩阵参数值](#)。

如果默认的色域转换参数不适用，仅需修改一次，该通道内的其它任务都使用修改后的色域转换参数，则在创建对应图像或视频处理通道后，可调用本接口设置色域转换参数，再调用具体的图像或视频处理接口，然后再调用对应的图像或视频处理结果获取接口。

如果默认的色域转换参数不适用，不同任务使用不同的色域转换参数，需多次修改，则在创建对应图像或视频处理通道后，在上一次任务执行结束后、下一次任务执行前调用本接口设置色域转换参数（如果上一次任务没有执行结束，就再次调用本接口设

置色域转换参数，可能会导致任务处理的结果不正确），再调用具体的图像或视频处理接口，然后再调用对应的图像或视频处理结果获取接口。

调用顺序示例如下：

1. [hi_mpi_sys_set_chn_csc_matrix](#)（第一次修改色域转换参数）
2. [hi_mpi_vpc_crop](#)（抠图）
3. [hi_mpi_vpc_get_process_result](#)（等待抠图任务结束）
4. [hi_mpi_sys_set_chn_csc_matrix](#)（第二次修改色域转换参数）
5. [hi_mpi_vpc_resize](#)（缩放）
6. [hi_mpi_vpc_get_process_result](#)（等待缩放任务结束）

函数原型

[hi_s32](#) [hi_mpi_sys_set_chn_csc_matrix](#)([hi_mod_id](#) mode, [hi_s32](#) chn, [hi_csc_matrix](#) csc_matrix, [hi_csc_coefficient](#) *csc_coefficient)

参数说明

参数名	输入/输出	说明
mode	输入	模块ID，当前只支持HI_ID_VDEC和HI_ID_VPC。
chn	输入	通道号。当mode设置为HI_ID_VDEC时，通道号范围请参见 1.9.17.5 hi_mpi_vdec_create_chn 处chn参数的取值范围；当mode设置为HI_ID_VPC时，通道号范围请参见 1.9.16.4 hi_mpi_vpc_create_chn 处chn参数的取值范围。
csc_matrix	输入	色域转换矩阵的数据标准。
csc_coefficient	输入	色域转换矩阵参数值的指针。 如果将csc_matrix参数设置为HI_CSC_MATRIX_USER，则需要使用csc_coefficient参数设置色域转换矩阵参数；否则，此处传任意值。

返回值说明

- 0：成功
- 非0：失败，参见[1.9.21.1 公共返回码](#)

1.9.3.13 hi_mpi_sys_get_chn_csc_matrix

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

函数功能

用户使用该接口获取通道的色域转换矩阵参数值。

约束说明

- 在创建通道成功后，才可以调用本接口获取通道的色域转换矩阵参数值。
- 不调用[hi_mpi_sys_set_chn_csc_matrix](#)接口设置色域转换矩阵参数时，返回默认的[BT601 wide](#)标准的色域转换矩阵参数值。

函数原型

```
hi_s32 hi_mpi_sys_get_chn_csc_matrix(hi_mod_id mode, hi_s32 chn,
hi_csc_matrix *csc_matrix, hi_csc_coefficient *csc_coefficient)
```

参数说明

参数名	输入/输出	说明
mode	输入	模块ID，当前只支持HI_ID_VDEC和HI_ID_VPC。
chn	输入	通道号。 当mode设置为HI_ID_VDEC时，通道号范围请参见 1.9.17.5 hi_mpi_vdec_create_chn 处chn参数的取值范围； 当mode设置为HI_ID_VPC时，通道号范围请参见 1.9.16.4 hi_mpi_vpc_create_chn 处chn参数的取值范围。
csc_matrix	输出	色域转换矩阵的数据标准的指针。
csc_coefficient	输出	色域转换矩阵参数值的指针。

返回值说明

- 0：成功
- 非0：失败，参见[1.9.21.1 公共返回码](#)

1.9.3.14 hi_mpi_sys_get_image_align_info

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

函数功能

根据模块、图片宽高计算出宽高对齐后的值以及存放图片数据的内存大小。

约束说明

- 图片宽、高不能小于该模块最小宽高、不能大于最大宽高，图片格式必须是该模块所支持的。

- 各模块支持的图片格式、对齐要求请参见[1.9.16 VPC图像处理功能](#)、[1.9.17 VDEC视频解码功能/JPEGD图片解码功能](#)、[1.9.18 VENC视频编码功能/JPEGE图片编码功能](#)、[1.9.19 PNGD图片解码功能](#)下的约束说明。
图片格式不支持HI_PIXEL_FORMAT_UNKNOWN。

函数原型

```
hi_s32 hi_mpi_sys_get_image_align_info(const hi_module_type mod_type[],
const hi_u32 mod_num, const hi_img_base_info *img_base_info,
hi_img_align_info *img_align_info)
```

参数说明

参数名	输入/输出	说明
mod_type	输入	模块类型数组。 如果数组中只有一个模块，则系统内部按该模块的对齐值来计算；如果数组中有多个模块，则系统内部按多个模块中的最大对齐值来计算；如果将模块类型设置为HI_MOD_ALL，则系统内部会从所有模块中选取最大对齐值来计算，便于简化管理各模块的对齐差异，操作简便，但可能占用内存较大。
mod_num	输入	模块类型数组长度。
img_base_info	输入	图片基本信息结构体。
img_align_info	输出	图片对齐信息、内存大小的结构体。

返回值说明

- 0：成功
- 非0：失败，参见[1.9.21.1 公共返回码](#)

1.9.3.15 hi_mpi_sys_bind

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

调用本接口建立数据接收者与数据源之间的关联关系，同一个数据接收者只能绑定一个数据源，绑定后，数据源生成的数据将自动发送给接收者。

约束说明

- Ascend EP标准形态下，不支持使用该接口。
- 当前支持如下数据源、数据接收者之间的绑定关系：

数据源	数据接收者	说明
视频输入VI	视频处理VPSS	VPSS Group ID取值在 [256, 268)范围内。
视频解码VDEC	视频处理VPSS	VPSS Group ID取值在[0, 256)范围内。
视频处理VPSS	视频输出VO	VPSS Group ID取值在[0, 256)范围内。
音频解码ADEC	音频输出AO	-
音频输入AI	音频编码AENC	-

- 除以下场景可将设备号或通道号设置为0，其它场景均需根据实际使用的设备号、通道号来设置。
 - VI和VDEC作为数据源，是以通道为发送者，向其他模块发送数据，用户将设备号src_chn.dev_id参数设置为0，SDK不检查输入的设备号。
 - VPSS作为数据接收者时，是以设备（GROUP）为接收者，接收其他模块发来的数据，用户将通道号dst_chn.chn_id参数设置为0。

函数原型

hi_s32 hi_mpi_sys_bind(const **hi_mpp_chn** *src_chn, const **hi_mpp_chn** *dest_chn)

参数说明

参数名	输入/输出	说明
src_chn	输入	源通道指针。
dest_chn	输入	目的通道指针。

返回值说明

- 0：成功
- 非0：失败，参见[1.9.21.1 公共返回码](#)

1.9.3.16 hi_mpi_sys_unbind

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

调用本接口解除数据源与数据接收者之间的关联关系。

约束说明

- Ascend EP标准形态下，不支持使用该接口。
- dst_chn如果找不到绑定的源通道，则直接返回成功。如果找到了绑定的源通道，但是绑定的源通道和src_chn不匹配，则返回失败。

函数原型

```
hi_s32 hi_mpi_sys_unbind(const hi_mpp_chn *src_chn, const hi_mpp_chn *dest_chn)
```

参数说明

参数名	输入/输出	说明
src_chn	输入	源通道指针。
dest_chn	输入	目的通道指针。

返回值说明

- 0：成功
- 非0：失败，参见[1.9.21.1 公共返回码](#)

1.9.3.17 hi_mpi_vb_create_pool

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

Atlas 200/500 A2推理产品，不支持使用该接口。

函数功能

调用本接口创建内存池。

函数原型

```
hi_vb_pool hi_mpi_vb_create_pool(const hi_vb_pool_config *vb_pool_cfg)
```

参数说明

参数名	输入/输出	说明
vb_pool_cfg	输入	内存池配置参数。

返回值说明

- 成功，返回内存池ID
- 失败，返回HI_VB_INVALID_POOLID，无效的内存池ID

1.9.3.18 hi_mpi_vb_destroy_pool

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

Atlas 200/500 A2推理产品，不支持使用该接口。

函数功能

调用本接口销毁内存池。

函数原型

[hi_s32](#) hi_mpi_vb_destroy_pool([hi_vb_pool](#) pool)

参数说明

参数名	输入/输出	说明
pool	输入	内存池ID，由 hi_mpi_vb_create_pool 接口返回。

返回值说明

- 0：成功
- 非0：失败，参见[1.9.21.1 公共返回码](#)

1.9.3.19 hi_mpi_vb_get_block

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

Atlas 200/500 A2推理产品，不支持使用该接口。

函数功能

调用本接口获取内存块。

函数原型

```
hi_vb_blk hi_mpi_vb_get_block(hi_vb_pool pool, hi_u64 blk_size, const hi_char *mmz_name)
```

参数说明

参数名	输入/输出	说明
pool	输入	内存池ID，由hi_mpi_vb_create_pool接口返回。
blk_size	输入	内存块大小。
mmz_name	输入	内存池名字，当pool=-1时，可以基于内存名字获取内存块。

返回值说明

- 成功，返回内存块handle
- 失败，返回HI_VB_INVALID_HANDLE，无效的内存块handle

1.9.3.20 hi_mpi_vb_release_block

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

Atlas 200/500 A2推理产品，不支持使用该接口。

函数功能

调用本接口释放内存块。

函数原型

```
hi_s32 hi_mpi_vb_release_block(hi_vb_blk vb_blk)
```

参数说明

参数名	输入/输出	说明
vb_blk	输入	内存块handle，由 hi_mpi_vb_get_block 接口返回。

返回值说明

- 0：成功
- 非0：失败，参见[1.9.21.1 公共返回码](#)

1.9.3.21 hi_mpi_vb_handle_to_addr

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

Atlas 200/500 A2推理产品，不支持使用该接口。

函数功能

调用本接口将内存块handle转换成地址。

函数原型

[hi_void*](#) [hi_mpi_vb_handle_to_addr](#)([hi_vb_blk](#) vb_blk)

参数说明

参数名	输入/输出	说明
vb_blk	输入	内存块handle，由 hi_mpi_vb_get_block 接口返回。

返回值说明

- 成功，返回地址
- 失败，返回空

1.9.3.22 hi_mpi_vb_block_copyref

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。
Atlas 200/500 A2推理产品，不支持使用该接口。

函数功能

调用本接口增加内存块引用计数。

函数原型

hi_s32 hi_mpi_vb_block_copyref(hi_vb_blk vb_blk)

参数说明

参数名	输入/输出	说明
vb_blk	输入	内存块handle，由 hi_mpi_vb_get_block 接口返回。

返回值说明

- 0：成功
- 非0：失败，参见[1.9.21.1 公共返回码](#)

1.9.4 AI 音频输入/AO 音频输出

1.9.4.1 功能及约束说明

- AI（Audio Input）音频输入、AO（Audio Output）音频输出API主要实现配置及启用音频输入设备、获取音频帧数据、播放等功能，本章中的接口均不支持多进程，且同一个设备ID不支持在多线程中使用。
- 对于内置Audio Codec，主要通过ioctl命令字实现对硬件设备的操作。
- 在录音起始阶段，由于硬件存在资源准备的过程，因此可能在起始阶段瞬间存在音频数据波动现象，但并不影响后续录音功能。
- 关于AI、AO设备的说明如下：

表 1-47 AO 设备

AO dev	输出对接器件
0	对接外置codec0
1	对接外置codec1
2	内置codec
3	HDMI0
4	HDMI1

表 1-48 AI 设备

AI dev	输入对接器件
0	对接外置codec0
1	对接外置codec1
2	内置codec

1.9.4.2 hi_mpi_ai_set_pub_attr

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

设置AI设备属性。

约束说明

- 仅支持I2S主模式。
- 在双声道和右声道模式下，chn_cnt需为2；在左声道模式下，chn_cnt需为1。

函数原型

hi_s32 hi_mpi_ai_set_pub_attr(**hi_audio_dev** ai_dev, **hi_aio_attr** *attr);

参数说明

参数名	输入/输出	说明。
ai_dev	输入	音频设备号，取值范围：[0, 2]。
attr	输入	AI设备属性。

返回值说明

- 0：成功
- 非0：失败，参见[1.9.21.2 音频相关返回码](#)。

参考资源

接口调用流程，参见[音频获取&音频播放](#)。

1.9.4.3 hi_mpi_ai_enable

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

启用AI设备。

约束说明

- 启用AI设备前，必须先调用[hi_mpi_ai_set_pub_attr](#)接口配置AI设备属性，否则返回属性未配置错误。
- 如果AI设备已经处于enable状态，则直接返回成功。

函数原型

[hi_s32](#) hi_mpi_ai_enable([hi_audio_dev](#) ai_dev);

参数说明

参数名	输入/输出	说明。
ai_dev	输入	音频设备号，取值范围：[0, 2]。

返回值说明

- 0：成功
- 非0：失败，参见[1.9.21.2 音频相关返回码](#)。

参考资源

接口调用流程，参见[音频获取&音频播放](#)。

1.9.4.4 hi_mpi_ai_disable

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

禁用AI设备。

约束说明

- 如果AI设备已经处于禁用状态，则直接返回成功。
- 禁用AI设备前，必须先调用[hi_mpi_ai_disable_chn](#)接口禁用该设备下已启用的所有AI通道。

函数原型

[hi_s32](#) [hi_mpi_ai_disable](#)([hi_audio_dev](#) ai_dev);

参数说明

参数名	输入/输出	说明。
ai_dev	输入	音频设备号，取值范围：[0, 2]。

返回值说明

- 0：成功
- 非0：失败，参见[1.9.21.2 音频相关返回码](#)。

参考资源

接口调用流程，参见[音频获取&音频播放](#)。

1.9.4.5 hi_mpi_ai_set_chn_attr

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

设置AI通道属性。

函数原型

[hi_s32](#) [hi_mpi_ai_set_chn_attr](#)([hi_audio_dev](#) ai_dev, [hi_ai_chn](#) ai_chn, const [hi_ai_chn_attr](#) *chn_attr);

参数说明

参数名	输入/输出	说明。
ai_dev	输入	音频设备号，取值范围：[0, 2]。 NVR（ Network Video Recorder ）场景中仅涉及dev2。
ai_chn	输入	音频输入通道号。 目前只支持单声道，所以ai_chn取值固定为0。
chn_attr	输入	音频通道属性结构体指针。 - 通道属性结构体中目前只有一个成员变量，用于设置通道的工作模式，默认工作模式为HI_AI_CHN_MODE_FAST。 - NVR（ Network Video Recorder ）场景中需要是使用HI_AI_CHN_MODE_NORMAL标准模式，支持调用hi_mpi_ai_enable_resample接口开启重采样功能。 - HI_AI_CHN_MODE_FAST模式下不支持重采样功能，不支持绑定AI与AENC，通过AI获取的音频数据，用户需调用hi_mpi_ai_get_frame接口获取音频数据。

返回值说明

- 0：成功
- 非0：失败，参考[1.9.21.2 音频相关返回码](#)。

参考资源

接口调用流程，参见NVR场景语音对讲。

1.9.4.6 hi_mpi_ai_enable_chn

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

启用AI通道。

约束说明

- 启用AI通道前，必须先调用[hi_mpi_ai_set_pub_attr](#)接口配置AI设备属性，否则返回属性未配置错误。
- 启用AI通道前，必须先调用[hi_mpi_ai_enable](#)接口启用其所属的AI设备，否则返回设备未启动的错误码。

- 录音为左声道和双声道时，ai_chn固定为0；右声道时，ai_chn固定为1。
- 如果AI设备下该通道已经处于enable状态，则直接返回成功。

函数原型

```
hi_s32 hi_mpi_ai_enable_chn(hi_audio_dev ai_dev, hi_ai_chn ai_chn);
```

参数说明

参数名	输入/输出	说明
ai_dev	输入	音频设备号，取值范围：[0, 2]。
ai_chn	输入	录音通道号。

返回值说明

- 0：成功
- 非0：失败，参见[1.9.21.2 音频相关返回码](#)。

参考资源

接口调用流程，参见[音频获取&音频播放](#)。

1.9.4.7 hi_mpi_ai_disable_chn

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

禁用AI通道。

约束说明

- 录音为左声道和双声道时，ai_chn固定为0；右声道时，ai_chn固定为1。
- 如果已经设置属性，但AI chn已经处于禁用状态，则直接返回成功；如果未设置属性，此时调用该接口会返回属性未配置的错误。

函数原型

```
hi_s32 hi_mpi_ai_disable_chn(hi_audio_dev ai_dev, hi_ai_chn ai_chn);
```

参数说明

参数名	输入/输出	说明
ai_dev	输入	音频设备号，取值范围：[0, 2]。
ai_chn	输入	录音通道号。

返回值说明

- 0：成功
- 非0：失败，参见[1.9.21.2 音频相关返回码](#)。

参考资源

接口调用流程，参见[音频获取&音频播放](#)。

1.9.4.8 hi_mpi_ai_get_frame

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

获取音频帧。

约束说明

- 获取音频帧数据前，必须先调用[hi_mpi_ai_enable](#)接口使能对应的AI设备。
- 录音为左声道和双声道时，ai_chn需固定为0；右声道时，ai_chn为1。
- 本接口与hi_mpi_ai_release_frame接口需成对出现，调用本接口获取音频帧，音频帧使用完毕之后应该及时调用hi_mpi_ai_release_frame接口释放音频帧。

函数原型

```
hi_s32 hi_mpi_ai_get_frame(hi_audio_dev ai_dev, hi_ai_chn ai_chn,  
hi_audio_frame *frame, hi_aec_frame *aec_frame, hi_s32 milli_sec);
```

参数说明

参数名	输入/输出	说明
ai_dev	输入	音频设备号，取值范围：[0, 2]。
ai_chn	输入	音频输入通道号。

参数名	输入/输出	说明
frame	输出	音频帧结构体指针。
aec_frame	输出	回声抵消参考帧结构体指针。 预留参数，当前设置为NULL。
milli_sec	输入	配置获取数据的超时时间，单位是毫秒，取值范围如下： • -1：表示阻塞模式，无数据时一直等待； • 0：表示非阻塞模式，无数据时则报错返回； • >0：需要配置具体的超时时间，表示超过指定时间后无数据，则报错返回。

返回值说明

- 0：成功
- 非0：失败，参见[1.9.21.2 音频相关返回码](#)。

参考资源

接口调用流程，参见[音频获取&音频播放](#)。

1.9.4.9 hi_mpi_ai_release_frame

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

释放音频帧。

约束说明

- 录音为左声道和双声道时，ai_chn需固定为0；右声道时，ai_chn为1。
- 本接口与hi_mpi_ai_get_frame接口需成对出现，调用hi_mpi_ai_get_frame接口获取音频帧，音频帧使用完毕之后应该及时调用本接口释放音频帧。
- 不支持重复调用本接口。

函数原型

```
hi_s32 hi_mpi_ai_release_frame(hi_audio_dev ai_dev, hi_ai_chn ai_chn, const hi_audio_frame *frame, const hi_aec_frame *aec_frame);
```


参数说明

参数名	输入/输出	描述
ai_dev	输入	音频设备号，取值范围：[0, 2]。
ai_chn	输入	音频输入通道号。
frame	输入	音频帧结构体指针。 预留参数，当前设置为NULL。
aec_frame	输入	设置为NULL。

返回值说明

- 0：成功
- 非0：失败，参见[1.9.21.2 音频相关返回码](#)。

参考资源

接口调用流程，参见[音频获取&音频播放](#)。

1.9.4.10 hi_mpi_ai_enable_resample

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

启用AI重采样。

约束说明

- 调用[hi_mpi_ai_enable_chn](#)接口启用AI通道后，调用[hi_mpi_sys_bind](#)接口绑定VENC与AI前，调用本接口启用重采样功能。
- 调用[hi_mpi_ai_disable_chn](#)接口禁用AI通道后，如果重新启用AI通道，并使用重采样功能，需调用本接口再次启用重采样。
- 由于奈奎斯特采样定理的限制，在音频采样率为8kHz时，则所支持的音频频率小于4kHz。
- 由于AI的采样率固定为48kHz，G.711a、G.711u协议的采样率仅支持8kHz，因此需启用重采样功能；而48kHz在AAC协议采样率支持的范围内，因此使用AAC协议时，在AI时无需重采样。

函数原型

```
hi_s32 hi_mpi_ai_enable_resample(hi_audio_dev ai_dev, hi_ai_chn ai_chn,
hi_audio_sample_rate out_sample_rate);
```

参数说明

参数名	输入/输出	说明。
ai_dev	输入	音频设备号，取值范围： [0, 2]。 NVR (Network Video Recorder) 场景中仅涉及 dev2。
ai_chn	输入	音频输入通道号。 目前只支持单声道（左声道），所以ai_chn取值固定为 0。
hi_audio_sample_rate	输入	音频重采样的输出采样率，当前只支持8k。

返回值说明

- 0：成功
- 非0：失败，参考[1.9.21.2 音频相关返回码](#)。

参考资源

接口调用流程，参见NVR场景语音对讲。

1.9.4.11 hi_mpi_ai_disable_resample

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

禁用AI重采样。

函数原型

```
hi_s32 hi_mpi_ai_disable_resample(hi_audio_dev ai_dev, hi_ai_chn ai_chn);
```

参数说明

参数名	输入/输出	说明。
ai_dev	输入	音频设备号，取值范围：[0, 2]。 NVR（ Network Video Recorder ）场景中仅涉及dev2。
ai_chn	输入	音频输入通道号。 目前只支持单声道（左声道），所以ai_chn取值固定为0。

返回值说明

- 0：成功
- 非0：失败，参考[1.9.21.2 音频相关返回码](#)。

参考资源

接口调用流程，参见NVR场景语音对讲。

1.9.4.12 hi_mpi_ao_set_pub_attr

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

设置AO设备属性。

约束说明

- 在设置属性之前，需要先调用[hi_mpi_ao_disable](#)接口禁用AO设备。
- 在双声道和右声道模式下，chn_cnt需为2；在左声道模式下，chn_cnt需为1。

函数原型

```
hi_s32 hi_mpi_ao_set_pub_attr(hi_audio_dev ao_dev, hi_ao_attr *attr);
```

参数说明

参数名	输入/输出	描述
ao_dev	输入	音频设备号，取值范围：[0, 4]。

参数名	输入/输出	描述
attr	输入	AO设备属性。

返回值说明

- 0：成功。
- 非0：失败，参见[1.9.21.2 音频相关返回码](#)。

参考资源

接口调用流程，参见[音频获取&音频播放](#)。

1.9.4.13 hi_mpi_ao_enable

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

启用AO设备。

约束说明

- 在启用AO设备前，必须先调用[hi_mpi_ao_set_pub_attr](#)接口配置AO设备属性，否则会返回属性未配置的错误。
- 如果AO设备已经处于enable状态，则直接返回成功。

函数原型

```
hi_s32 hi_mpi_ao_enable(hi_audio_dev ao_dev);
```

参数说明

参数名	输入/输出	描述
ao_dev	输入	音频设备号，取值范围：[0, 4]。

返回值说明

- 0：成功。
- 非0：失败，参见[1.9.21.2 音频相关返回码](#)。

参考资源

接口调用流程，参见音频获取&音频播放。

1.9.4.14 hi_mpi_ao_disable

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

禁用AO设备。

约束说明

- 如果AO设备已经处于disable状态，则直接返回成功。
- 禁用AO设备前，必须先调用[hi_mpi_ao_disable_chn](#)接口禁用该设备下已启用的所有AO通道。

函数原型

```
hi_s32 hi_mpi_ao_disable(hi_audio_dev ao_dev);
```

参数说明

参数名	输入/输出	描述
ao_dev	输入	音频设备号，取值范围：[0, 4]。

返回值说明

- 0：成功。
- 非0：失败，参见[1.9.21.2 音频相关返回码](#)。

参考资源

接口调用流程，参见音频获取&音频播放。

1.9.4.15 hi_mpi_ao_enable_chn

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

启用AO通道。

约束说明

- 启用AO通道前，必须先调用[hi_mpi_ao_set_pub_attr](#)接口配置AO设备属性，否则返回属性未配置错误。
- 启用AO通道前，必须先调用[hi_mpi_ao_enable](#)接口启用其所属的AO设备，否则返回设备未启动的错误码。
- 录音为左声道和双声道时，ao_chn固定为0；右声道时，ao_chn固定为1。
- 如果AO设备已经处于enable状态，则直接返回成功。

函数原型

```
hi_s32 hi_mpi_ao_enable_chn(hi_audio_dev ao_dev, hi_ao_chn ao_chn);
```

参数说明

参数名	输入/输出	描述
ao_dev	输入	音频设备号，取值范围：[0, 4]。
ao_chn	输入	音频输出通道号。

返回值说明

- 0：成功。
- 非0：失败，参见[1.9.21.2 音频相关返回码](#)。

参考资源

接口调用流程，参见[音频获取&音频播放](#)。

1.9.4.16 hi_mpi_ao_disable_chn

- Atlas 200/300/500 推理产品，不支持该接口。
- Atlas 训练系列产品，不支持该接口。
- Atlas 推理系列产品，不支持该接口。
- Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

禁用AO通道。

约束说明

- 录音为左声道和双声道时，ao_chn固定为0；右声道时，ao_chn固定为1。
- 如果已经设置属性，但AO_chn已经处于禁用状态，则直接返回成功；如果未设置属性，此时调用该接口会报属性未配置的错误。

函数原型

```
hi_s32 hi_mpi_ao_disable_chn(hi_audio_dev ao_dev, hi_ao_chn ao_chn);
```

参数说明

参数名	输入/输出	描述
ao_dev	输入	音频设备号，取值范围：[0, 4]。
ao_chn	输入	音频输出通道号。

返回值说明

- 0：成功。
- 非0：失败，参见1.9.21.2 音频相关返回码。

参考资源

接口调用流程，参见音频获取&音频播放。

1.9.4.17 hi_mpi_ao_send_frame

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

发送AO音频帧。

约束说明

- 该接口用于用户主动发送音频帧至AO输出。
- 调用该接口发送音频帧到AO输出时，必须先调用hi_mpi_ao_enable_chn接口使能对应的AO通道。
- 播音为左声道和双声道时，ai_chn需固定为0；右声道时，ai_chn为1。
- 音频帧结构体的成员变量len和virt_addr需要匹配，len代表帧长（单位：字节），virt_addr为传入的音频数据的地址。**建议**每次传入的数据为一帧数据，即长度len为point_num_per_frame与位宽（16bit为2字节，24bit为3字节）的乘积。

- 对长度len限制如下：首先根据len与位宽计算出数据点数（数据点数=len/位宽对应的字节数），数据点数的范围为[1, **frame_num * point_num_per_frame**]，最大上限为4096。
 - 举例1：如果frame_num=2，point_num_per_frame=480（frame_num和point_num_per_frame的范围限制可以参见结构体**hi_aio_attr**），此时frame_num * point_num_per_frame=960，且960<4096，那么数据点数的范围为[1, 960]。
 - 举例2：如果frame_num=30，point_num_per_frame=4096，此时frame_num * point_num_per_frame=122880，且122880>4096，那么数据点数的范围为[1, 4096]。
 - 举例3：如果根据frame_num和point_num_per_frame确定的数据点数范围为[1, 960]。若此时len=960，位宽为16位（即2字节），计算得到数据点数为960/2=480，在[1, 960]范围之内，即符合要求；若此时len=3840，位宽为16位（即2字节），计算得到数据点数为3840/2=1920，在[1, 960]范围之外，则会返回报错。

函数原型

```
hi_s32 hi_mpi_ao_send_frame(hi_audio_dev ao_dev, hi_ao_chn ao_chn, const hi_audio_frame *data, hi_s32 milli_sec);
```

参数说明

参数名	输入/输出	说明
ao_dev	输入	音频设备号，取值范围：[0, 4]。
ao_chn	输入	音频输出通道号。
data	输入	音频帧结构体。
milli_sec	输入	配置发送数据的超时时间，单位是毫秒，取值范围如下： -1：表示阻塞模式，无数据时一直等待； 0：表示非阻塞模式，无数据时则报错返回； >0：需要配置具体的超时时间，表示超过指定时间后无数据，则报错返回。

返回值说明

- 0：成功。
- 非0：失败，参见**1.9.21.2 音频相关返回码**。

参考资源

接口调用流程，参见音频获取&音频播放。

1.9.4.18 hi_mpi_ao_get_chn_delay

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

获取AO通道中当前的音频延时大小。

约束说明

- 该接口要在AO通道成功启用之后调用。
- 获取AO通道中当前的音频延时大小，当延时查询到的延时为0时停止等待，进入播音停止流程。

函数原型

```
hi_s32 hi_mpi_ao_get_chn_delay(hi_audio_dev ao_dev, hi_ao_chn ao_chn,
hi_u32 *milli_sec);
```

参数说明

参数名	输入/输出	说明
ao_dev	输入	音频设备号，取值范围：[0, 4]。
ao_chn	输入	音频输出通道号。
milli_sec	输出	音频延时的指针。音频延时大小以毫秒为单位。

返回值说明

- 0：成功。
- 非0：失败，参见[1.9.21.2 音频相关返回码](#)。

参考资源

接口调用流程，参见[音频获取&音频播放](#)。

1.9.4.19 hi_mpi_ao_enable_resample

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

启用AO重采样。

约束说明

- 调用hi_mpi_ao_enable_chn接口启用AO通道之后，调用[hi_mpi_sys_bind](#)接口绑定ADEC与AO之前，调用本接口启用重采样功能。
- 调用hi_mpi_ao_disable_chn接口禁用AO通道后，如果重新启用AO通道，并使用重采样功能，需调用本接口再次启用重采样。
- AO重采样的输入采样率必须与AO设备属性配置的采样率不相同。
- 由于奈奎斯特采样定理的限制，在音频采样率为8kHz时，则所支持的音频频率小于4kHz。
- 由于AO的采样率固定为48kHz，G.711a、G.711u协议的采样率仅支持8kHz，因此需启用重采样功能；而48kHz在AAC协议采样率支持的范围内，因此使用AAC协议时，在AO时无需重采样。

函数原型

[hi_s32](#) [hi_mpi_ao_enable_resample](#)([hi_audio_dev](#) ao_dev, [hi_ao_chn](#) ao_chn, [hi_audio_sample_rate](#) in_sample_rate);

参数说明

参数名	输入/输出	说明。
ao_dev	输入	音频设备号，取值范围：[0, 4]。 NVR（Network Video Recorder）场景中仅涉及dev2、dev3。
ao_chn	输入	音频输出通道号。目前只支持单声道（左声道），所以ao_chn取值固定为0。
hi_audio_sample_rate	输入	音频重采样的输入采样率,目前只支持8k。

返回值说明

- 0：成功
- 非0：失败，参考[1.9.21.2 音频相关返回码](#)。

参考资源

接口调用流程，参见NVR场景语音对讲。

1.9.4.20 hi_mpi_ao_disable_resample

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

禁用 AO 重采样。

函数原型

```
hi_s32 hi_mpi_ao_disable_resample(hi_audio_dev ao_dev, hi_ao_chn ao_chn);
```

参数说明

参数名	输入/输出	说明。
ao_dev	输入	音频设备号，取值范围：[0, 4]。 NVR（ Network Video Recorder ）场景中仅涉及dev2、dev3。
ao_chn	输入	音频输出通道号。 目前只支持单声道（左声道），所以ao_chn取值固定为0。

返回值说明

- 0：成功
- 非0：失败，参考[1.9.21.2 音频相关返回码](#)。

参考资源

接口调用流程，参见NVR场景语音对讲。

1.9.5 AENC 音频编码/ADEC 音频解码

1.9.5.1 功能及约束说明

当前仅支持G.711a、G.711u、AAC（ Advanced Audio Coding ）协议、单声道数据进行编码、解码，且为软件编码、解码。本章中的接口均不支持多进程，且同一个设备ID不支持在多线程中使用。

音频编码时，G.711a、G.711u码流遵循以下表格中描述的帧结构，即在每帧码流数据的净荷数据之前填充有4个字节的帧头；音频解码时，需要读取相应的帧头信息。在G.711a、G.711u码流中，目前帧头中的数据净荷长度需为40，80，120，160，240（单位：hi_s16），所以对应的每帧采样点数即为80，160，240，320，480。但在VQE（ Voice Quality Enhancement ）框架中，编解码操作是与AI、AO绑定在一起，所以支持的每帧采样点数范围缩小。

参数位置(单位: hi_s16)	参数比特位说明	参数含义
0	[15:8]	数据帧类型标志位。 01: 语音帧; 其他: 保留。
	[7:0]	保留。
1	[15:8]	帧循环计数器: 0 ~ 255。
	[7:0]	数据净荷长度(单位: hi_s16)。
2	[15:0]	净荷数据。
3	[15:0]	净荷数据。
.....	[15:0]	净荷数据。
n+1	[15:0]	净荷数据。
n+2	[15:0]	净荷数据。

G.711a、G.711u、AAC协议的相关规格及优缺点如下:

协议	采样率	每帧采样点数	码率 (kbps)	压缩率	CPU 消耗	描述
G.711a、G.711u	8kHz	80/160/240/320/480	64	2	1 MHz	• 优点: 语音质量最好; CPU 消耗小; 支持广泛, 协议免费。 • 缺点: 压缩效率低。

协议	采样率	每帧采样 点数	码率 (kbps)	压缩率	CPU 消耗	描述
AAC 仅支持 MPEG-4 版本、无 CRC (Cyclic Redunda ncy Check) 校验的音 频码流	48kHz	1024	[48, 256]	与码率相 关，码率 越小，压 缩率越大	-	• 优点：持 更高采样 率的编解 码，能够 保留原始 音频的更 多高频信 息，同时 拥有更多 的码率范 围选择， 压缩率更 高。 • 缺点：算 法复杂度 高。

1.9.5.2 hi_mpi_aenc_create_chn

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

创建音频编码通道。

约束说明

通道如果已经被创建，则会返回通道已创建的错误。

函数原型

hi_s32 hi_mpi_aenc_create_chn(**hi_aenc_chn** aenc_chn, const **hi_aenc_chn_attr** *attr);

参数说明

参数名	输入/输出	说明。
aenc_chn	输入	通道号，取值范围[0, 6)
attr	输入	音频编码通道属性指针。

返回值说明

- 0：成功
- 非0：失败，参考[1.9.21.2 音频相关返回码](#)。

1.9.5.3 hi_mpi_aenc_destroy_chn

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

销毁音频编码通道。

函数原型

```
hi_s32 hi_mpi_aenc_destroy_chn(hi_aenc_chn aenc_chn);
```

参数说明

参数名	输入/输出	说明。
aenc_chn	输入	通道号，取值范围[0, 6)

返回值说明

- 0：成功
- 非0：失败，参考[1.9.21.2 音频相关返回码](#)。

1.9.5.4 hi_mpi_aenc_get_stream

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

获取音频编码后的码流。

约束说明

- 必须调用[hi_mpi_aenc_create_chn](#)接口创建通道后才可能获取码流，否则直接返回失败。

- 本接口与hi_mpi_aenc_release_stream接口需成对出现，调用本接口获取码流，码流使用完毕之后应该及时调用hi_mpi_aenc_release_stream接口释放码流。

函数原型

```
hi_s32 hi_mpi_aenc_get_stream(hi_aenc_chn aenc_chn, hi_audio_stream
*stream, hi_s32 milli_sec);
```

参数说明

参数名	输入/输出	说明。
aenc_chn	输入	编码通道号，取值范围[0，6)
stream	输出	获取的音频码流。
milli_sec	输入	配置获取数据的超时时间，单位是毫秒，取值范围如下： ● -1：表示阻塞模式，无数据时一直等待； ● 0：表示非阻塞模式，无数据时则报错返回； ● >0：需要配置具体的超时时间，表示超过指定时间后无数据，则报错返回。

返回值说明

- 0：成功
- 非0：失败，参考1.9.21.2 音频相关返回码。

1.9.5.5 hi_mpi_aenc_release_stream

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

释放音频编码后的码流。

约束说明

- 必须调用hi_mpi_aenc_create_chn接口创建通道后才可能获取码流，否则直接返回失败。
- 本接口与hi_mpi_aenc_get_stream接口需成对出现，调用hi_mpi_aenc_get_stream接口获取码流，码流使用完毕之后应该及时调用本接口释放码流。
- 不支持重复调用本接口。

函数原型

```
hi_s32 hi_mpi_aenc_release_stream(hi_aenc_chn aenc_chn, const  
hi_audio_stream *stream);
```

参数说明

参数名	输入/输出	说明。
aenc_chn	输入	编码通道号，取值范围[0, 6)
stream	输出	待释放的音频码流。

返回值说明

- 0：成功
- 非0：失败，参考[1.9.21.2 音频相关返回码](#)。

1.9.5.6 hi_mpi_adec_create_chn

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

创建音频解码通道。

约束说明

通道如果已经被创建，则会返回通道已创建的错误。

函数原型

```
hi_s32 hi_mpi_adec_create_chn(hi_adec_chn adec_chn, const hi_adec_chn_attr  
*attr);
```

参数说明

参数名	输入/输出	说明。
adec_chn	输入	通道号，取值范围[0, 6)
attr	输入	音频解码通道属性指针。

返回值说明

- 0：成功
- 非0：失败，参考[1.9.21.2 音频相关返回码](#)。

1.9.5.7 hi_mpi_adec_destroy_chn

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

销毁音频解码通道。

函数原型

```
hi_s32 hi_mpi_adec_destroy_chn(hi_adec_chn adec_chn);
```

参数说明

参数名	输入/输出	说明
adec_chn	输入	通道号，取值范围[0, 6)

返回值说明

- 0：成功
- 非0：失败，参考[1.9.21.2 音频相关返回码](#)。

1.9.5.8 hi_mpi_adec_send_stream

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

向音频解码通道发送码流。

约束说明

- 创建音频解码通道时（[hi_mpi_adec_create_chn](#)）可以通过hi_adec_mode变量指定解码方式为pack方式或stream方式，**推荐使用pack方式**。

- **pack方式**: 用于用户确认当前码流包为一帧数据编码结果的情况下, 解码器会直接进行对其解码, 如果不是一帧, 解码器会出错。
例如, 对于G.711a文件来说, 帧头中有数据长度信息, 假设此时的数据长度为240 (单位: s16), 即为 $240 \times 2 = 480$ 字节, 由于帧头长度为4字节, 所以完整的一帧长度为 $480 + 4 = 484$ 字节, 即每次数据应该读484字节。
- **stream方式**: 用于不确定码流包为是否为一帧的情况, 该方式效率较低, 且可能会有延迟或者缓存满的情况。使用场景适合程序长时间运行不退出、发送端和接收端无匹配要求、效率不高的场景。
- 发送数据时必须保证音频解码通道已经被创建, 否则直接返回失败。
- 支持阻塞或非阻塞方式发送码流, 当阻塞方式发送码流时, 如果用于缓存解码后的音频帧的buffer满, 则此接口调用会被阻塞, 直至解码后的音频帧数据被取走。
- 该接口只支持单声道、且带有符合要求的语音帧头的G.711a、G.711u音源进行解码。

函数原型

```
hi_s32 hi_mpi_adec_send_stream(hi_adec_chn adec_chn, const  
hi_audio_stream *stream, hi_bool block);
```

参数说明

参数名	输入/输出	说明。
adec_chn	输入	通道号, 取值范围[0, 6)
stream	输入	音频码流。
block	输入	阻塞标识: • HI_TRUE: 阻塞。 • HI_FALSE: 非阻塞。

返回值说明

- 0: 成功
- 非0: 失败, 参考[1.9.21.2 音频相关返回码](#)。

1.9.6 音量调节

1.9.6.1 功能描述

- 音量调节可以实现对在内置codec (即 device 2) 播音、录音时的音量调节。
- 调节音量共涉及DACL (播音左声道)、DACR (播音右声道)、ADCL (录音左声道)、ADCR (录音右声道) 四种情况。
- 需在播音录音开始之前调节音量, 播音录音结束之后音量不会回到默认大小, 而是保持在调节之后的状态。
- 多进程调节音量时, 音量的调节效果以最后一次调节的大小为准。

- 播音时如果出现削波现象、声音失真，可能是音量过大，需要调小音量，或者调整音源。
- 录音时如果出现削波现象、声音失真，可能外部输入的信号过大，用户需自行调整外部输入的信号。

1.9.6.2 HI_ACODEC_SET_DACL_VOLUME

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

左声道输出音量控制。

约束说明

- 左声道输出音量范围：[0, 127]，赋值越大，音量越小。赋值为 0 时，音量最大，为6db，赋值为 127 时，音量最小，为静音。
- 例如hi_acodec_volume_ctrl结构体中volume_ctrl变量被配置为x，音量会被调节为 (6-x) db。

函数原型

int ioctl(int fd, HI_ACODEC_SET_DACL_VOLUME, hi_acodec_volume_ctrl *arg);

参数说明

参数名	输入/输出	说明
fd	输入	Audio Codec设备文件描述符。
HI_ACODEC_SET_DACL_VOLUME	输入	ioctl号。
arg	输入	内置Audio Codec音量控制参数。

返回值说明

- 0：成功。
- 非0：失败。

1.9.6.3 HI_ACODEC_SET_DACR_VOLUME

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

右声道输出音量控制。

约束说明

- 右声道输出音量范围：[0, 127]，赋值越大，音量越小。赋值为 0 时，音量最大，为6db，赋值为 127 时，音量最小，为静音。
- 例如hi_acodec_volume_ctrl结构体中volume_ctrl变量被配置为x，音量会被调节为 (6-x) db。

函数原型

```
int ioctl(int fd, HI_ACODEC_SET_DACR_VOLUME, hi_acodec_volume_ctrl*arg);
```

参数说明

参数名	输入/输出	说明
fd	输入	Audio Codec设备文件描述符。
HI_ACODEC_SET_DACR_VOLUME	输入	ioctl号。
arg	输入	内置Audio Codec音量控制参数。

返回值说明

- 0：成功。
- 非0：失败。

1.9.6.4 HI_ACODEC_GET_DACL_VOLUME

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

获取左声道输出的音量。

约束说明

- 左声道输出音量范围：[0, 127]，赋值越大，音量越小。赋值为 0 时，音量最大，为6db，赋值为 127 时，音量最小，为静音。

- 例如获取hi_acodec_volume_ctrl结构体中volume_ctrl变量的值为x，则代表音量被调节为 (6-x) db。

函数原型

int ioctl(int fd, HI_ACODEC_GET_DACL_VOLUME, hi_acodec_volume_ctrl*arg);

参数说明

参数名	输入/输出	说明
fd	输入	Audio Codec设备文件描述符。
HI_ACODEC_GET_DACL_VOLUME	输入	ioctl号。
arg	输出	内置Audio Codec音量控制参数。

返回值说明

- 0：成功。
- 非0：失败。

1.9.6.5 HI_ACODEC_GET_DACR_VOLUME

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

获取右声道输出的音量控制。

约束说明

- 右声道输出音量范围：[0, 127]，赋值越大，音量越小。赋值为 0 时，音量最大，为6db，赋值为 127 时，音量最小，为静音。
- 例如获取hi_acodec_volume_ctrl结构体中volume_ctrl变量的值为x，则代表音量被调节为 (6-x) db。

函数原型

int ioctl(int fd, HI_ACODEC_GET_DACR_VOLUME, hi_acodec_volume_ctrl*arg);

参数说明

参数名	输入/输出	说明
fd	输入	Audio Codec设备文件描述符。
HI_ACODEC_GET_DACR_VOLUME	输入	ioctl号。
arg	输出	内置Audio Codec音量控制参数。

返回值说明

- 0：成功。
- 非0：失败。

1.9.6.6 HI_ACODEC_SET_ADCL_VOLUME

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

左声道输入音量控制。

约束说明

- 左声道输入音量范围：[0, 127]，赋值越大，音量越小。赋值为 0 时，音量最大，为30db，赋值为 127 时，音量最小，为-97db。
- 例如hi_acodec_volume_ctrl结构体中volume_ctrl变量被配置为y，音量会被调节为 (30-y) db。

函数原型

int ioctl(int fd, HI_ACODEC_SET_ADCL_VOLUME, hi_acodec_volume_ctrl*arg);

参数说明

参数名	输入/输出	说明
fd	输入	Audio Codec设备文件描述符。
HI_ACODEC_SET_ADCL_VOLUME	输入	ioctl号。
arg	输入	内置Audio Codec音量控制参数。

返回值说明

- 0：成功。
- 非0：失败。

1.9.6.7 HI_ACODEC_SET_ADCR_VOLUME

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

右声道输入音量控制。

约束说明

- 右声道输入音量范围：[0, 127]，赋值越大，音量越小。赋值为 0 时，音量最大，为30db，赋值为 127 时，音量最小，为-97db。
- 例如hi_acodec_volume_ctrl结构体中volume_ctrl变量被配置为y，音量会被调节为 (30-y) db。

函数原型

```
int ioctl(int fd, HI_ACODEC_SET_ADCR_VOLUME, hi_acodec_volume_ctrl*arg);
```

参数说明

参数名	输入/输出	说明
fd	输入	Audio Codec设备文件描述符。
HI_ACODEC_SET_ADCR_VOLUME	输入	ioctl号。
arg	输入	内置Audio Codec音量控制参数。

返回值说明

- 0：成功。
- 非0：失败。

1.9.6.8 HI_ACODEC_GET_ADCL_VOLUME

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

获取左声道输入的音量控制。

约束说明

- 左声道输入音量范围：[0, 127]，赋值越大，音量越小。赋值为 0 时，音量最大，为30db，赋值为 127 时，音量最小，为-97db。
- 例如获取hi_acodec_volume_ctrl结构体中volume_ctrl变量的值为y，则代表音量被调节为（ 30-y ）db。

函数原型

```
int ioctl(int fd, HI_ACODEC_GET_ADCL_VOLUME, hi_acodec_volume_ctrl*arg);
```

参数说明

参数名	输入/输出	说明
fd	输入	Audio Codec设备文件描述符。
HI_ACODEC_GET_ADCL_VOLUME	输入	ioctl号。
arg	输出	内置Audio Codec音量控制参数。

返回值说明

- 0：成功。
- 非0：失败。

1.9.6.9 HI_ACODEC_GET_ADCR_VOLUME

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

获取右声道输入的音量控制。

约束说明

- 右声道输入音量范围：[0, 127]，赋值越大，音量越小。赋值为 0 时，音量最大，为30db，赋值为 127 时，音量最小，为-97db。
- 例如获取hi_acodec_volume_ctrl结构体中volume_ctrl变量的值为y，则代表音量被调节为（ 30-y ）db。

函数原型

```
int ioctl(int fd, HI_ACODEC_GET_ADCR _VOLUME, hi_acodec_volume_ctrl*arg);
```

参数说明

参数名	输入/输出	说明
fd	输入	Audio Codec设备文件描述符。
HI_ACODEC_GET_ADCR _VOLUME	输入	ioctl号。
arg	输出	内置Audio Codec音量控制参数。

返回值说明

- 0：成功。
- 非0：失败。

1.9.7 ISP 系统控制及 3A 算法注册

本文中接口，如无特殊说明，不支持多进程。

1.9.7.1 功能描述

ISP (Image Signal Processing) 系统控制

系统控制部分用于注册3A算法、注册Sensor驱动、初始化ISP firmware、运行ISP firmware、退出ISP firmware、配置ISP属性等功能。ISP的Firmware包含三部分，一部分是ISP控制单元和基础算法库，一部分是AE/AWB/AF算法库，一部分是Sensor库。

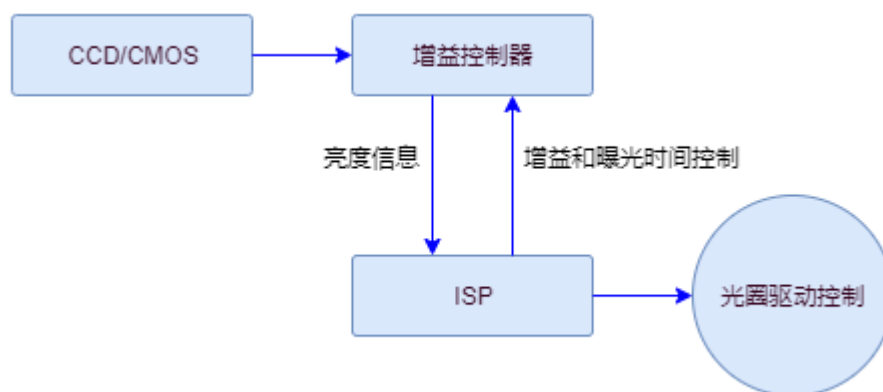
AE (Auto Exposure) 自动曝光

AE模块实现的功能是：根据自动测光系统获得当前图像的曝光量，再自动配置镜头光圈、Sensor快门及增益来获得最佳的图像质量。

自动曝光的算法主要分：

- 光圈优先，优先调整光圈到合适的位置，再分配曝光时间和增益，只适合p-iris 镜头，这样能均衡噪声和景深。
- 快门优先，优先分配曝光时间，再分配Sensor增益和ISP增益，这样拍摄的图像噪声会比较小。
- 增益优先，优先分配Sensor增益和ISP增益，再分配曝光时间，适合拍摄运动物体的场景。当前AE算法也支持客户设定更灵活的曝光分配策略，AE模块的工作流程下图所示。

图 1-11 AE 模块工作流程图



AWB (Automatic White Balance) 自动白平衡

色温随可见光的光谱成分变化而变化，在低色温光源下，白色物体偏红，在高色温光源下，白色物体偏蓝，人眼可根据大脑的记忆判断，识别物体的真实颜色。AWB算法的功能是降低外界光源对物体真实颜色的影响，使得我们采集的颜色信息转变为在理想日光光源下的无偏色信息。

1.9.7.2 基本概念

- 曝光时间：Sensor 积累电荷的时间，是 Sensor pixel 从开始曝光到电量被读出的这段时间。
- 曝光增益：对 Sensor 的输出电荷的总的放大系数，一般有数字增益和模拟增益，模拟增益引入的噪声会稍小，所以一般优先用模拟增益。
- 光圈：光圈是镜头中可以改变中间孔大小的机械装置。
- 抗闪烁：由于电灯电源工频与 Sensor 的帧率不匹配而导致的画面闪烁，一般通过限定曝光时间和修改 Sensor 的帧率来达到抗闪烁的效果。
- 色温：色温是按绝对黑体定义的，光源辐射在可见区和绝对黑体的辐射完全相同时，此时黑体的温度称此光源的色温。
- 白平衡：在不同色温的光源下，白色在传感器中的响应会偏蓝或偏红。白平衡算法通过调整 R, G, B 三个颜色通道的强度，使白色真实呈现。

1.9.7.3 hi_mpi_isp_sensor_reg_callback

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

Atlas A2 训练系列产品/Atlas 800I A2 推理产品，不支持该接口。

函数功能

ISP 提供的 Sensor 注册的回调接口，用于获取差异化的初始化参数，并控制 Sensor。

函数原型

```
hi_s32 hi_mpi_isp_sensor_reg_callback(hi_vi_pipe vi_pipe, const  
hi_isp_sns_attr_info *sns_attr_info, const hi_isp_sensor_register *sns_register)
```

约束说明

此接口不支持多进程操作。

参数说明

参数名	输入/输出	说明
vi_pipe	输入	VI PIPE号。 取值范围：[0, 12)。
sns_attr_info	输入	向ISP注册的Sensor的属性。 该结构体内的sensor_id是Sensor库中自定义的值，需确保唯一，主要用于校对向ISP注册的Sensor和向3A注册的Sensor是否为同一个Sensor。
sns_register	输入	Sensor注册结构体指针。

返回值说明

- 0：成功
- 非0：失败，参见ISP返回码。

参考资源

接口调用流程，参见视频数据获取功能。

1.9.7.4 hi_mpi_isp_sensor_unreg_callback

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

ISP提供的取消Sensor注册的回调接口。

函数原型

```
hi_s32 hi_mpi_isp_sensor_unreg_callback(hi_vi_pipe vi_pipe, hi_sensor_id  
sensor_id)
```

约束说明

此接口不支持多进程操作。

参数说明

参数名	输入/输出	说明
vi_pipe	输入	VI PIPE号。 取值范围：[0, 12)。
sensor_id	输入	向ISP注册的Sensor的Id。 sensor_id是Sensor库中自定义的值，需确保唯一，主要用于校对向ISP反注册的Sensor和向3A反注册的Sensor是否为同一个Sensor。

返回值说明

- 0：成功
- 非0：失败，参见[ISP返回码](#)。

参考资源

接口调用流程，参见[视频数据获取功能](#)。

1.9.7.5 hi_mpi_isp_mem_init

- Atlas 200/300/500 推理产品，不支持该接口。
- Atlas 训练系列产品，不支持该接口。
- Atlas 推理系列产品，不支持该接口。
- Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

初始化ISP内部资源。

函数原型

```
hi_s32 hi_mpi_isp_mem_init(hi_vi_pipe vi_pipe)
```

约束说明

- 调用该接口前，必须先调用[hi_mpi_isp_sensor_reg_callback](#)接口向ISP注册Sensor。
- 不支持多进程，必须要与hi_mpi_isp_sensor_reg_callback、hi_mpi_isp_ae_lib_reg_callback、hi_mpi_isp_awb_lib_reg_callback、hi_mpi_isp_init、hi_mpi_isp_run、hi_mpi_isp_exit、hi_mpi_ae_register、hi_mpi_awb_register接口在同一个进程调用。

- 当前业务正在运行hi_mpi_isp_run接口时，不能调用本接口。
- 推荐调用hi_mpi_isp_exit接口后，再调用本接口重新初始化。
- 不支持相同vi_pipe时，多线程执行ISP创建和销毁（即多线程同时调用hi_mpi_isp_sensor_reg_callback、hi_mpi_isp_ae_lib_reg_callback、hi_mpi_isp_awb_lib_reg_callback、hi_mpi_isp_init、hi_mpi_isp_run、hi_mpi_isp_exit、hi_mpi_ae_register、hi_mpi_awb_register接口）

参数说明

参数名	输入/输出	说明
vi_pipe	输入	VI PIPE号。 取值范围：[0, 12)。

返回值说明

- 0：成功
- 非0：失败，参见[ISP返回码](#)。

参考资源

接口调用流程，参见[视频数据获取功能](#)。

1.9.7.6 hi_mpi_isp_set_pub_attr

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

设置ISP公共属性，例如裁剪分辨率、帧率等。

函数原型

[hi_s32](#) hi_mpi_isp_set_pub_attr([hi_vi_pipe](#) vi_pipe, const [hi_isp_pub_attr](#) *pub_attr)

约束说明

- 调用本接口前，必须先调用[hi_mpi_isp_mem_init](#)接口初始化ISP内部资源。
- ISP支持运行过程中动态裁剪图像的起始位置。
- 调用本接口后ISP内的处理流程：
ISP firmware判断图像WDR模式、分辨率、帧率是否变化，若都不变则直接返回；否则，ISP firmware会调用通过[hi_mpi_isp_sensor_reg_callback](#)接口注册

的回调函数[pfn_cmos_set_wdr_mode](#)、[pfn_cmos_set_image_mode](#)改变Sensor模式。

- 若Sensor模式不改变（回调函数返回值-2），判断ISP的裁剪宽高是否变化，若有变化，ISP firmware切换分辨率，并调用通过[hi_mpi_isp_sensor_reg_callback](#)接口注册的回调函数[pfn_cmos_sensor_init](#)重新配置Sensor；若无变化，就不变。
- 若Sensor模式改变（回调函数返回值为0），则ISP firmware会调用通过[hi_mpi_isp_sensor_reg_callback](#)接口注册的回调函数[pfn_cmos_sensor_init](#)重新配置Sensor。
- 用户可以修改回调函数[pfn_cmos_set_image_mode](#)的实现来调整Sensor模式切换的顺序。如只提供了5M30fps和1080P60fps初始化序列的Sensor，若要运行1080P30fps，可以从5M30fps裁剪得到，也可以从1080P60fps降帧得到，修改[pfn_cmos_set_image_mode](#)函数实现即可。
- 切换线性模式和帧WDR模式时，同样会判断[pfn_cmos_set_image_mode](#)的返回值，因此线性模式和帧WDR模式应该采用不同的 image_mode，才能保证切换成功。

参数说明

参数名	输入/输出	说明
vi_pipe	输入	VI PIPE号。 取值范围：[0, 12)。
pub_attr	输入	ISP公共属性。

返回值说明

- 0：成功
- 非0：失败，参见[ISP返回码](#)。

参考资源

接口调用流程，参见[视频数据获取功能](#)。

1.9.7.7 hi_mpi_isp_get_pub_attr

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

获取ISP公共属性。

函数原型

```
hi_s32 hi_mpi_isp_get_pub_attr(hi_vi_pipe vi_pipe, hi_isp_pub_attr *pub_attr)
```

参数说明

参数名	输入/输出	说明
vi_pipe	输入	VI PIPE号。 取值范围：[0, 12)。
pub_attr	输出	ISP公共属性。

返回值说明

- 0：成功
- 非0：失败，参见ISP返回码。

1.9.7.8 hi_mpi_isp_init

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

初始化ISP firmware。

函数原型

```
hi_s32 hi_mpi_isp_init(hi_vi_pipe vi_pipe)
```

约束说明

- 调用本接口前，必须先调用hi_mpi_isp_set_pub_attr接口设置图像公共属性。
- 为确保前一次进程退出前资源已完全清理，推荐先调用hi_mpi_isp_exit接口，再调用本接口初始化ISP firmware。
- 不支持多进程，必须要与hi_mpi_isp_sensor_reg_callback、hi_mpi_isp_run、hi_mpi_isp_exit、hi_mpi_ae_register、hi_mpi_awb_register接口在同一个进程调用。
- 不支持重复调用本接口。
- 不支持相同vi_pipe时，多线程执行ISP创建和销毁（即多线程同时调用hi_mpi_isp_sensor_reg_callback、hi_mpi_isp_run、hi_mpi_isp_exit、hi_mpi_ae_register、hi_mpi_awb_register接口）。

- ISP初始化后，需要一帧时间给硬件读取算法系数表。所以调用本接口后一帧时间内，不能调用hi_mpi_vi_stop_pipe接口停止VI PIPE。

参数说明

参数名	输入/输出	说明
vi_pipe	输入	VI PIPE号。 取值范围：[0, 12)。

返回值说明

- 0：成功
- 非0：失败，参见[ISP返回码](#)。

参考资源

接口调用流程，参见[视频数据获取功能](#)。

1.9.7.9 hi_mpi_isp_run

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

运行ISP firmware。

函数原型

[hi_s32](#) hi_mpi_isp_run([hi_vi_pipe](#) vi_pipe)

约束说明

- 调用本接口前，必须先调用[hi_mpi_isp_init](#)接口初始化ISP firmware。
- 不支持多进程，必须要与[hi_mpi_isp_sensor_reg_callback](#)、[hi_mpi_isp_init](#)、[hi_mpi_isp_exit](#)、[hi_mpi_ae_register](#)、[hi_mpi_awb_register](#)接口在同一个进程调用。
- 该接口是阻塞接口，建议用户采用单独的线程调用本接口。

参数说明

参数名	输入/输出	说明
vi_pipe	输入	PIPE号。 取值范围：[0, 12)。

返回值说明

- 0：成功
- 非0：失败，参见[ISP返回码](#)。

参考资源

接口调用流程，参见[视频数据获取功能](#)。

1.9.7.10 hi_mpi_isp_exit

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

退出ISP firmware。

函数原型

[hi_s32](#) hi_mpi_isp_exit([hi_vi_pipe](#) vi_pipe)

约束说明

- 调用[hi_mpi_isp_run](#)之后，再调用本接口退出ISP firmware。
- 不支持多进程，必须要与[hi_mpi_isp_sensor_reg_callback](#)、[hi_mpi_isp_init](#)、[hi_mpi_isp_run](#)、[hi_mpi_ae_register](#)、[hi_mpi_awb_register](#)接口在同一个进程调用。
- 支持重复调用本接口。
- 不支持相同vi_pipe时，多线程执行ISP创建和销毁（即多线程同时调用[hi_mpi_isp_sensor_reg_callback](#)、[hi_mpi_isp_run](#)、[hi_mpi_isp_exit](#)、[hi_mpi_ae_register](#)、[hi_mpi_awb_register](#)接口）。

参数说明

参数名	输入/输出	说明
vi_pipe	输入	PIPE号。 取值范围：[0, 12)。

返回值说明

- 0：成功
- 非0：失败，参见[ISP返回码](#)。

参考资源

接口调用流程，参见[视频数据获取功能](#)。

1.9.7.11 hi_mpi_isp_set_sns_slave_attr

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

设置从模式Sensor行场同步信号。

从模式Sensor需要其提供行同步信号XHS和场同步信号XVS，在这两个信号的控制下进行曝光与数据读出，这个接口主要配置同步信号发生模块，使其输出Sensor要求的行场时序。此接口一般在Sensor库里调用。

函数原型

[hi_s32](#) hi_mpi_isp_set_sns_slave_attr([hi_slave_dev](#) slave_dev, const [hi_isp_slave_sns_sync*](#) sns_sync)

约束说明

从信号有两组绑定关系，首先pipe 和vsync之间有绑定关系，其次vsync内部还可选择不同的信号源slave，即不同pipe可以选择不同的从信号设备vsync，不同的从信号设备vsync又可以选择不同的信号源slave。

参数说明

参数名	输入/输出	说明
slave_dev	输入	从信号设备号。
sns_sync	输入	同步信号配置。

返回值说明

- 0：成功
- 非0：失败，参见[ISP返回码](#)。

1.9.7.12 hi_mpi_isp_get_sns_slave_attr

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

获取从模式Sensor行场同步信号。

函数原型

[hi_s32](#) hi_mpi_isp_get_sns_slave_attr([hi_slave_dev](#) slave_dev,
[hi_isp_slave_sns_sync](#) *sns_sync)

参数说明

参数名	输入/输出	说明
slave_dev	输入	从信号设备号。
sns_sync	输出	同步信号配置。

返回值说明

- 0：成功
- 非0：失败，参见[ISP返回码](#)。

1.9.7.13 hi_mpi_ae_register

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

向ISP注册昇腾AE库。

函数原型

`hi_s32 hi_mpi_ae_register(hi_vi_pipe vi_pipe, hi_isp_3a_alg_lib *ae_lib)`

约束说明

- AE库可以注册多个实例。
- 此接口不支持多进程操作。

参数说明

参数名	输入/输出	说明
vi_pipe	输入	VI PIPE号。 取值范围：[0, 12)。
ae_lib	输入	AE算法库结构体指针。

返回值说明

- 0：成功
- 非0：失败，参见[ISP返回码](#)。

参考资源

接口调用流程，参见[视频数据获取功能](#)。

1.9.7.14 hi_mpi_ae_unregister

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

向ISP取消注册昇腾AE库。

函数原型

hi_s32 hi_mpi_ae_unregister(**hi_vi_pipe** vi_pipe, **hi_isp_3a_alg_lib** *ae_lib)

约束说明

此接口不支持多进程操作。

参数说明

参数名	输入/输出	说明
vi_pipe	输入	VI PIPE号。 取值范围：[0, 12)。
ae_lib	输入	AE 算法库结构体指针。

返回值说明

- 0：成功
- 非0：失败，参见[ISP返回码](#)。

参考资源

接口调用流程，参见[视频数据获取功能](#)。

1.9.7.15 hi_mpi_ae_sensor_reg_callback

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

AE库提供的Sensor注册的回调接口。AE通过Sensor注册的一系列回调接口，获取差异化的初始化参数，并控制Sensor。

函数原型

hi_s32 hi_mpi_ae_sensor_reg_callback(**hi_vi_pipe** vi_pipe, **hi_isp_3a_alg_lib** *ae_lib, const **hi_isp_sns_attr_info** *sns_attr_info, const **hi_isp_ae_sensor_register** *pregister)

约束说明

此接口不支持多进程操作。

参数说明

参数名	输入/输出	说明
vi_pipe	输入	VI PIPE号。 取值范围：[0, 12)。
ae_lib	输入	AE算法库结构体指针。
sns_attr_info	输入	向AE注册的Sensor的属性。 该结构体内的sensor_id是sensor 库中自定义的值，需确保唯一，主要用于校对向ISP注册的Sensor和向3A注册的Sensor是否为同一个Sensor。
pregister	输入	Sensor注册结构体指针。

返回值说明

- 0：成功
- 非0：失败，参见[ISP返回码](#)。

参考资源

接口调用流程，参见[视频数据获取功能](#)。

1.9.7.16 hi_mpi_ae_sensor_unreg_callback

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

AE库提供的Sensor注销的回调接口。

函数原型

hi_s32 hi_mpi_ae_sensor_unreg_callback(**hi_vi_pipe** vi_pipe, **hi_isp_3a_alg_lib** *ae_lib, **hi_sensor_id** sensor_id)

约束说明

此接口不支持多进程操作。

参数说明

参数名	输入/输出	说明
vi_pipe	输入	VI PIPE号。 取值范围：[0, 12)。
ae_lib	输入	AE算法库结构体指针。
sensor_id	输入	向AE反注册的Sensor的Id。 sensor_id是Sensor库中自定义的值，主要用于校对向ISP反注册的Sensor和向3A反注册的Sensor是否为同一个Sensor。

返回值说明

- 0：成功
- 非0：失败，参见[ISP返回码](#)

参考资源

接口调用流程，参见[视频数据获取功能](#)。

1.9.7.17 hi_mpi_awb_register

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

向ISP注册昇腾AWB库。

函数原型

[hi_s32](#) hi_mpi_awb_register([hi_vi_pipe](#) vi_pipe, [hi_isp_3a_alg_lib](#) *awb_lib)

约束说明

- AWB库可以注册多个实例。
- 此接口不支持多进程操作。

参数说明

参数名	输入/输出	说明
vi_pipe	输入	VI PIPE号。 取值范围：[0, 12)。
awb_lib	输入	AWB算法库结构体指针。

返回值说明

- 0：成功
- 非0：失败，参见[ISP返回码](#)。

参考资源

接口调用流程，参见[视频数据获取功能](#)。

1.9.7.18 hi_mpi_awb_unregister

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

向ISP取消注册昇腾AWB库。

函数原型

[hi_s32](#) hi_mpi_awb_unregister([hi_vi_pipe](#) vi_pipe, [hi_isp_3a_alg_lib](#) *awb_lib)

约束说明

此接口不支持多进程操作。

参数说明

参数名	输入/输出	说明
vi_pipe	输入	VI PIPE号。 取值范围：[0, 12)。
awb_lib	输入	AWB算法库结构体指针。

返回值说明

- 0：成功
- 非0：失败，参见ISP返回码。

参考资源

接口调用流程，参见视频数据获取功能。

1.9.7.19 hi_mpi_awb_sensor_reg_callback

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

AWB库提供的Sensor注册的回调接口。AWB通过Sensor注册的一系列回调接口，获取差异化的初始化参数，并控制Sensor。

函数原型

hi_s32 hi_mpi_awb_sensor_reg_callback(**hi_vi_pipe** vi_pipe, **hi_isp_3a_alg_lib** *awb_lib, const **hi_isp_sns_attr_info** *sns_attr_info, const **hi_isp_awb_sensor_register** * awb_sns_register)

约束说明

此接口不支持多进程操作。

参数说明

参数名	输入/输出	说明
vi_pipe	输入	VI PIPE号。 取值范围：[0, 12)。
awb_lib	输入	AWB算法库结构体指针。
sns_attr_info	输入	向AWB注册的Sensor的属性。 该结构体内的sensor_id是Sensor库中自定义的值，需确保唯一，主要用于校对向ISP注册的Sensor和向3A注册的Sensor是否为同一个Sensor。
awb_sns_register	输入	Sensor注册结构体指针。

返回值说明

- 0：成功
- 非0：失败，参见[ISP返回码](#)。

参考资源

接口调用流程，参见[视频数据获取功能](#)。

参考样例

```
hi_isp_3a_alg_lib awb_lib;
hi_isp_awb_sensor_register awb_register;
hi_isp_sns_attr_info sns_attr_info;
hi_isp_awb_sensor_exp_func *exp_funcs = &awb_register.sns_exp;
memset(exp_funcs, 0, sizeof(hi_isp_awb_sensor_exp_func));
exp_funcs->pfn_cmos_get_awb_default = cmos_get_awb_default;
hi_vi_pipe vi_pipe = 0;
awb_lib.id = 0;
sns_attr_info.sensor_id = IMX178_ID;
strcpy(awb_lib.lib_name, HI_AWB_LIB_NAME);
ret = hi_mpi_awb_sensor_reg_callback(vi_pipe, &awb_lib, &sns_attr_info, &awb_register);
if (ret) {
    printf("sensor register callback function to awb lib failed!\n");
    return ret;
}
```

1.9.7.20 hi_mpi_awb_sensor_unreg_callback

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

AWB库提供的Sensor注销的回调接口。

函数原型

hi_s32 hi_mpi_awb_sensor_unreg_callback(**hi_vi_pipe** vi_pipe, **hi_isp_3a_alg_lib** *awb_lib, **hi_sensor_id** sensor_id)

约束说明

- 此接口不支持多进程操作。

参数说明

参数名	输入/输出	说明
vi_pipe	输入	VI PIPE号。 取值范围：[0, 12)。

参数名	输入/输出	说明
awb_lib	输入	AWB算法库结构体指针。
sensor_id	输入	向AWB注册的Sensor的Id。

返回值说明

- 0：成功
- 非0：失败，参见ISP返回码。

参考资源

接口调用流程，参见视频数据获取功能。

1.9.8 MIPI Rx ioctl 命令字

1.9.8.1 功能描述

MIPI Rx通过低电压差分信号接收原始视频数据，将接收到的串行差分信号（serial differential signal）转化为DC（Digital Camera）时序后传递给下一级模块VICAP（Video Capture）。MIPI Rx支持MIPI D-PHY、sub-LVDS、HiSPi（High-Speed Serial Pixel Interface）等串行视频信号输入。

SLVS-EC接口由SONY公司定义，用于高帧率和高分辨率图像采集，它可以将高速串行的数据转化为DC（Digital Camera）时序后传递给下一级模块 VICAP（Video Capture）。SLVS-EC串行视频接口可以提供更高的传输带宽，更低的功耗，在组包方式上，数据的冗余度也更低。在应用中SLVS-EC接口提供了更加可靠和稳定的传输。

说明

当前版本仅配套支持MIPI CSI接口以及SLVS-EC接口。

MIPI Rx

MIPI Rx是一个支持多种差分视频输入接口的采集单元，通过combo-PHY接收MIPI/LVDS/sub-LVDS/HiSPi接口的数据，通过不同的功能模式配置，MIPI Rx可以支持多种速度和分辨率的数据传输需求，支持多种外部输入设备。

- MIPI Rx最大支持Lane个数：
MIPI Rx最大支持8个Lane MIPI输入或8个Lane sub-LVDS输入或8个Lane HiSPi输入。
- MIPI Rx能同时对接多个Sensor，最多对接4个Sensor。
- MIPI Rx最大能同时对接不同数量的Sensor，每个Sensor需要的Lane也不尽相同。因此用户需要确定MIPI Rx 的LANE分布模式。具体的Lane 分布模式请参见下表：

表 1-49 MIPI Rx Lane 分布模式

Mode	DEV0	DEV1	DEV2	DEV3
0	L0~L7	N	N	N
1	L0~L3	N	L4~L7	N
2	L0~L3	N	L4、L6	L5、L7
3	L0、L2	L1、L3	L4、L6	L5、L7

SLVS-EC

SLVS-EC接口支持更高帧率更大分辨率图像的采集，通过SLVS-EC的PHY接收高速串行的数据转化为DC（Digital Camera）时序，通过不同的功能模式配置，SLVS-EC可以支持多种速度和分辨率的数据传输需求，支持多种外部输入设备。

SLVS-EC 最大支持 8Lane SLVS 输入。

Lane 管脚及复用说明

MIPI Rx与SLVS-EC的Lane复用管脚，同一时刻同一个Lane只能被MIPI Rx和SLVS-EC中的一个使用。

具体的Lane 管脚连接请参见下表。

表 1-50 MIPI Rx 与 SLVS-EC 的 Lane 复用关系图

LANE	MIPI0	MIPI1	MIPI2	MIPI3	SLVS0	SLVS1
Lane0	√	-	-	-	√	√
Lane1	√	√	-	-	√	√
Lane2	√	-	-	-	√	√
Lane3	√	√	-	-	√	√
Lane4	√	-	√	-	√	√
Lane5	√	-	√	√	√	√
Lane6	√	-	√	-	√	√
Lane7	√	-	√	√	√	√

- MIPI0对应VICAP的DEV 0，MIPI1对应VICAP的DEV1，依次类推。
- SLVS0对应VICAP的DEV0，SLVS1对应VICAP的DEV1。

1.9.8.2 基本概念

- MIPI

MIPI的全称是Mobile Industry Processor Interface(移动行业处理器接口)，本文描述的MIPI接口特指物理层使用D-PHY传输规范，协议层使用CSI-2的通信接口。

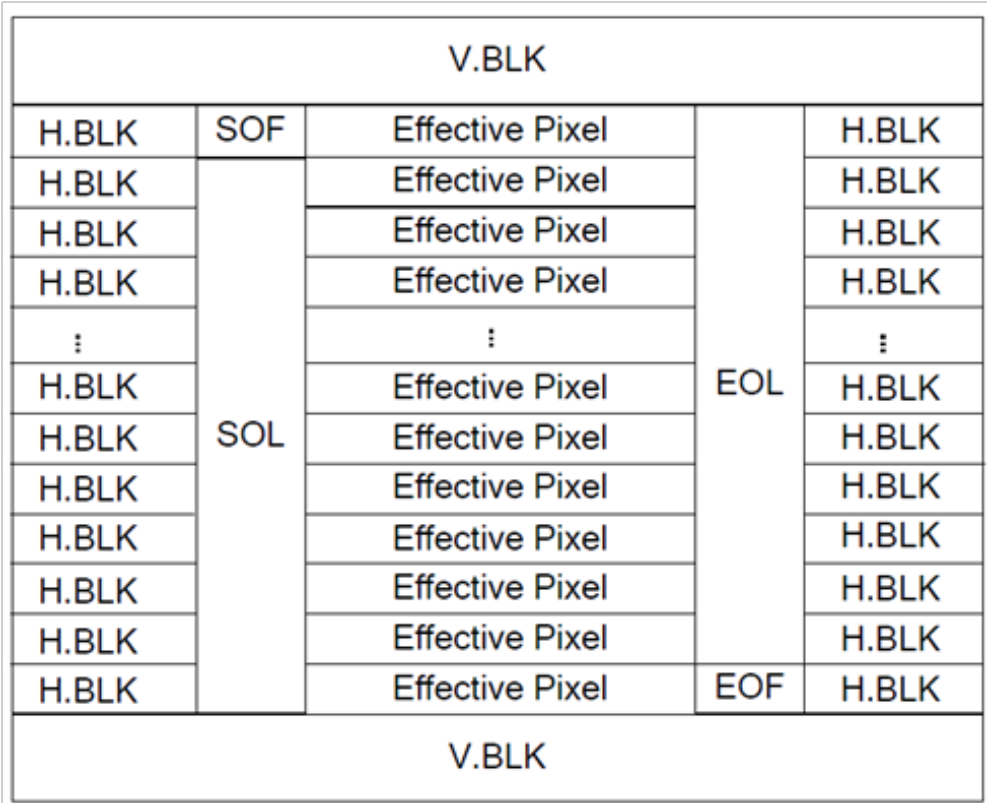
- sub-LVDS
LVDS的全称是Low Voltage differential Signaling(低压差分信号)，通过同步码区分消隐区和有效数据。sub-LVDS是LVDS的低压变种，常见于sony Sensor。
- SLVS-EC
SLVS-EC的全称是Scalable Low Voltage Signaling Embedded Clock，是与MIPI并列的接口，用于高帧率和高分辨率图像采集。
- Lane
用于连接发送端和接收端的一对高速差分线，既可以是时钟Lane，也可以是数据Lane。
- Link
发送端和接收端之间的时钟Lane和至少一个数据Lane组成一个 Link，本文中的link 是一个软件概念，每一个 link 包括两个数据 lane。
- 同步码
MIPI接口使用CSI-2 里面的短包进行同步，sub-LVDS 使用同步码区分有效数据和消隐区。sub-LVDS 有两种同步方式：
 - 使用 SOF/EOF 表示帧起始和结束，使用SOL/EOL 表示行的起始和结束。同步方式如下图所示。

图 1-12 SOF/EOF/SOL/EOL 同步方式

V.BLK				
H.BLK	SOF	Effective Pixel	EOL	H.BLK
H.BLK	SOL	Effective Pixel		H.BLK
H.BLK		Effective Pixel		H.BLK
H.BLK		Effective Pixel		H.BLK
:		:		:
H.BLK		Effective Pixel		H.BLK
H.BLK		Effective Pixel		H.BLK
H.BLK		Effective Pixel		H.BLK
H.BLK		Effective Pixel		H.BLK
H.BLK		Effective Pixel		H.BLK
H.BLK		Effective Pixel		H.BLK
H.BLK		Effective Pixel	EOF	H.BLK
V.BLK				

- 使用SAV(invalid) EAV(invalid)表示消隐区的无效数据开始和结束，使用 SAV(valid) EAV(valid)表示有效像素数据的开始和结束。
每个同步码由4个字段组成，每个字段的位宽与像素数据位宽保持一致。前3个字段为固定基准码字，第4个字段由Sensor厂家确定。
由于不同的Sensor可能会有不同的同步码，所以需要根据Sensor配置同步码。同步方式如下图所示。

图 1-13 SAV/EAV 同步方式



- DOL
DOL的全称是 Digital Overlap，指SONY的WDR 功能。

1.9.8.3 HI_MIPI_SET_DEV_ATTR

Atlas 200/300/500 推理产品，不支持该命令字。

Atlas 训练系列产品，不支持该命令字。

Atlas 推理系列产品，不支持该命令字。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该命令字。

功能描述

设置MIPI Rx、SLVS和并口设备属性。

定义

```
#define HI_MIPI_SET_DEV_ATTR_IOW(HI_MIPI_IOC_MAGIC, 0x01, combo_dev_attr_t)
```

参数

- HI_MIPI_IOC_MAGIC: MIPI Rx ioctl命令的幻数。
#define HI_MIPI_IOC_MAGIC 'm'
- 0x01: 标识命令字功能，用于设置MIPI Rx、SLVS和并口设备属性。

- combo_dev_attr_t: combo设备属性。

返回值

- 0: 成功
- -1: 失败

注意事项

操作Sensor复位信号线和时钟信号线会对所连接到该信号线的所有Sensor都产生效果。

1.9.8.4 HI_MIPI_SET_HS_MODE

Atlas 200/300/500 推理产品，不支持该命令字。

Atlas 训练系列产品，不支持该命令字。

Atlas 推理系列产品，不支持该命令字。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该命令字。

功能描述

设置MIPI Rx的Lane分布模式，对SLVS无作用。

定义

```
#define HI_MIPI_SET_HS_MODE _IOW(HI_MIPI_IOC_MAGIC, 0x0b, lane_divide_mode_t)
```

参数

- HI_MIPI_IOC_MAGIC: MIPI Rx ioctl命令的幻数。
#define HI_MIPI_IOC_MAGIC 'm'
- 0x0b: 标识命令字功能，用于设置MIPI Rx的Lane分布模式。
- lane_divide_mode_t: MIPI Rx的Lane分布模式。

返回值

- 0: 成功
- -1: 失败

注意事项

- 执行本命令字后，系统内部会对MIPI Rx子系统进行复位和解复位动作，因此在调用其它命令字前需先调用本命令字，否则会导致其它命令字的配置被复位。
- 在接多路Sensor输入时，建议在初始时根据硬件连接设置Lane的分布模式，在之后的多路Sensor采集过程中不能再使用本命令字，否则会导致本命令字之前的其它配置被复位。

1.9.8.5 HI_MIPI_ENABLE_MIPI_CLOCK

Atlas 200/300/500 推理产品，不支持该命令字。

Atlas 训练系列产品，不支持该命令字。

Atlas 推理系列产品，不支持该命令字。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该命令字。

功能描述

打开MIPI设备的时钟。

定义

```
#define HI_MIPI_ENABLE_MIPI_CLOCK_IOW(HI_MIPI_IOC_MAGIC, 0x0c, combo_dev_t)
```

参数

- HI_MIPI_IOC_MAGIC: MIPI Rx ioctl命令的幻数。

```
#define HI_MIPI_IOC_MAGIC 'm'
```
- 0x0c: 标识命令字功能，用于打开MIPI设备的时钟。
- combo_dev_t: 设备号。

返回值

- 0: 成功
- 1: 失败

1.9.8.6 HI_MIPI_DISABLE_MIPI_CLOCK

Atlas 200/300/500 推理产品，不支持该命令字。

Atlas 训练系列产品，不支持该命令字。

Atlas 推理系列产品，不支持该命令字。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该命令字。

功能描述

关闭MIPI设备的时钟。

定义

```
#define HI_MIPI_DISABLE_MIPI_CLOCK_IOW(HI_MIPI_IOC_MAGIC, 0x0d, combo_dev_t)
```

参数

- HI_MIPI_IOC_MAGIC: MIPI Rx ioctl命令的幻数。

```
#define HI_MIPI_IOC_MAGIC 'm'
```
- 0x0d: 标识命令字功能，用于关闭MIPI设备的时钟。
- combo_dev_t: 设备号。

返回值

- 0: 成功

- -1: 失败

1.9.8.7 HI_MIPI_RESET_MIPI

Atlas 200/300/500 推理产品，不支持该命令字。

Atlas 训练系列产品，不支持该命令字。

Atlas 推理系列产品，不支持该命令字。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该命令字。

功能描述

复位MIPI Rx。

定义

```
#define HI_MIPI_RESET_MIPI_IOW(HI_MIPI_IOC_MAGIC, 0x07, combo_dev_t)
```

参数

- HI_MIPI_IOC_MAGIC: MIPI Rx ioctl命令的幻数。
#define HI_MIPI_IOC_MAGIC 'm'
- 0x07: 标识命令字功能，用于复位MIPI Rx。
- combo_dev_t: 设备号。

返回值

- 0: 成功
- -1: 失败

1.9.8.8 HI_MIPI_UNRESET_MIPI

Atlas 200/300/500 推理产品，不支持该命令字。

Atlas 训练系列产品，不支持该命令字。

Atlas 推理系列产品，不支持该命令字。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该命令字。

功能描述

撤销复位MIPI Rx。

功能定义

```
#define HI_MIPI_UNRESET_MIPI_IOW(HI_MIPI_IOC_MAGIC, 0x08, combo_dev_t)
```

参数

- HI_MIPI_IOC_MAGIC: MIPI Rx ioctl命令的幻数。
#define HI_MIPI_IOC_MAGIC 'm'
- 0x08: 标识命令字功能，用于撤销复位MIPI Rx。

- combo_dev_t: 设备号。

返回值

- 0: 成功
- -1: 失败

1.9.8.9 HI_MIPI_ENABLE_SLVS_CLOCK

Atlas 200/300/500 推理产品，不支持该命令字。

Atlas 训练系列产品，不支持该命令字。

Atlas 推理系列产品，不支持该命令字。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该命令字。

功能描述

打开SLVS设备的时钟。

定义

```
#define HI_MIPI_ENABLE_SLVS_CLOCK_IOW(HI_MIPI_IOC_MAGIC, 0x0e, combo_dev_t)
```

参数

- HI_MIPI_IOC_MAGIC: MIPI Rx ioctl命令的幻数。
#define HI_MIPI_IOC_MAGIC 'm'
- 0x0e: 标识命令字功能，用于打开SLVS设备的时钟。
- combo_dev_t: 设备号。

返回值

- 0: 成功
- -1: 失败

1.9.8.10 HI_MIPI_DISABLE_SLVS_CLOCK

Atlas 200/300/500 推理产品，不支持该命令字。

Atlas 训练系列产品，不支持该命令字。

Atlas 推理系列产品，不支持该命令字。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该命令字。

功能描述

关闭SLVS设备的时钟。

定义

```
#define HI_MIPI_DISABLE_SLVS_CLOCK_IOW(HI_MIPI_IOC_MAGIC, 0x0f, combo_dev_t)
```

参数

- HI_MIPI_IOC_MAGIC: MIPI Rx ioctl命令的幻数。
`#define HI_MIPI_IOC_MAGIC 'm'`
- 0x0f: 标识命令字功能, 用于关闭SLVS设备的时钟。
- combo_dev_t: 设备号。

返回值

- 0: 成功
- -1: 失败

1.9.8.11 HI_MIPI_RESET_SLVS

Atlas 200/300/500 推理产品, 不支持该命令字。

Atlas 训练系列产品, 不支持该命令字。

Atlas 推理系列产品, 不支持该命令字。

Atlas A2训练系列产品/Atlas 800I A2推理产品, 不支持该命令字。

功能描述

复位SLVS。

定义

```
#define HI_MIPI_RESET_SLVS_IOW(HI_MIPI_IOC_MAGIC, 0x09, combo_dev_t)
```

参数

- HI_MIPI_IOC_MAGIC: MIPI Rx ioctl命令的幻数。
`#define HI_MIPI_IOC_MAGIC 'm'`
- 0x09: 标识命令字功能, 用于复位SLVS。
- combo_dev_t: 设备号。

返回值

- 0: 成功
- -1: 失败

1.9.8.12 HI_MIPI_UNRESET_SLVS

Atlas 200/300/500 推理产品, 不支持该命令字。

Atlas 训练系列产品, 不支持该命令字。

Atlas 推理系列产品, 不支持该命令字。

Atlas A2训练系列产品/Atlas 800I A2推理产品, 不支持该命令字。

功能描述

撤销复位SLVS。

定义

```
#define HI_MIPI_UNRESET_SLVS_IOW(HI_MIPI_IOC_MAGIC, 0x0a, combo\_dev\_t)
```

参数

- HI_MIPI_IOC_MAGIC: MIPI Rx ioctl命令的幻数。

```
#define HI_MIPI_IOC_MAGIC 'm'
```
- 0x0a: 标识命令字功能，用于撤销复位SLVS。
- [combo_dev_t](#): 设备号。

返回值

- 0: 成功
- -1: 失败

1.9.8.13 HI_MIPI_CONFIG_SENSOR_CLOCK

Atlas 200/300/500 推理产品，不支持该命令字。

Atlas 训练系列产品，不支持该命令字。

Atlas 推理系列产品，不支持该命令字。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该命令字。

功能描述

配置Sensor的输入时钟频率。

定义

```
#define HI_MIPI_CONFIG_SENSOR_CLOCK_IOW(HI_MIPI_IOC_MAGIC, 0x17, sns\_clk\_cfg\_t)
```

参数

- HI_MIPI_IOC_MAGIC: MIPI Rx ioctl命令的幻数。

```
#define HI_MIPI_IOC_MAGIC 'm'
```
- 0x17: 标识命令字功能，用于配置输入给Sensor的时钟频率。
- [sns_clk_cfg_t](#): Sensor从模式下输入给Sensor的时钟配置信息。
在Sensor工作在从模式或者需要芯片给Sensor提供时钟时，需要在流程中调用此命令字配置输入给Sensor的时钟配置信息。

返回值

- 0: 成功
- -1: 失败

注意事项

在配置Sensor时钟时，必须先使用[HI_MIPI_DISABLE_SENSOR_CLOCK](#)命令字禁用Sensor时钟。

1.9.8.14 HI_MIPI_ENABLE_SENSOR_CLOCK

Atlas 200/300/500 推理产品，不支持该命令字。

Atlas 训练系列产品，不支持该命令字。

Atlas 推理系列产品，不支持该命令字。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该命令字。

功能描述

打开Sensor的时钟。

定义

```
#define HI_MIPI_ENABLE_SENSOR_CLOCK _IOW(HI_MIPI_IOC_MAGIC, 0x10, sns_clk_source_t)
```

参数

- HI_MIPI_IOC_MAGIC: MIPI Rx ioctl命令的幻数。
#define HI_MIPI_IOC_MAGIC 'm'
- 0x10: 标识命令字功能，用于打开Sensor的时钟。
- sns_clk_source_t: Sensor的时钟源。

返回值

- 0: 成功
- 1: 失败

1.9.8.15 HI_MIPI_DISABLE_SENSOR_CLOCK

Atlas 200/300/500 推理产品，不支持该命令字。

Atlas 训练系列产品，不支持该命令字。

Atlas 推理系列产品，不支持该命令字。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该命令字。

功能描述

关闭Sensor的时钟。

定义

```
#define HI_MIPI_DISABLE_SENSOR_CLOCK _IOW(HI_MIPI_IOC_MAGIC, 0x11, sns_clk_source_t)
```

参数

- HI_MIPI_IOC_MAGIC: MIPI Rx ioctl命令的幻数。
#define HI_MIPI_IOC_MAGIC 'm'
- 0x11: 标识命令字功能，用于关闭Sensor的时钟。
- sns_clk_source_t: Sensor的时钟源。

返回值

- 0: 成功
- -1: 失败

1.9.8.16 HI_MIPI_RESET_SENSOR

Atlas 200/300/500 推理产品，不支持该命令字。

Atlas 训练系列产品，不支持该命令字。

Atlas 推理系列产品，不支持该命令字。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该命令字。

功能描述

复位Sensor。

定义

```
#define HI_MIPI_RESET_SENSOR_IOW(HI_MIPI_IOC_MAGIC, 0x05, sns_rst_source_t)
```

参数

- HI_MIPI_IOC_MAGIC: MIPI Rx ioctl命令的幻数。

```
#define HI_MIPI_IOC_MAGIC 'm'
```
- 0x05: 标识命令字功能，用于复位Sensor。
- sns_rst_source_t: Sensor复位信号线编号。

返回值

- 0: 成功
- -1: 失败

注意事项

部分Sensor对复位时间有要求，可以在撤销复位前增加延时操作。

1.9.8.17 HI_MIPI_UNRESET_SENSOR

Atlas 200/300/500 推理产品，不支持该命令字。

Atlas 训练系列产品，不支持该命令字。

Atlas 推理系列产品，不支持该命令字。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该命令字。

功能描述

撤销复位Sensor。

定义

```
#define HI_MIPI_UNRESET_SENSOR_IOW(HI_MIPI_IOC_MAGIC, 0x06, sns_rst_source_t)
```

参数

- HI_MIPI_IOC_MAGIC: MIPI Rx ioctl命令的幻数。
`#define HI_MIPI_IOC_MAGIC 'm'`
- 0x06: 标识命令字功能，用于复位Sensor。
- sns_rst_source_t: Sensor复位信号线编号。

返回值

- 0: 成功
- -1: 失败

1.9.9 VI 视频输入功能

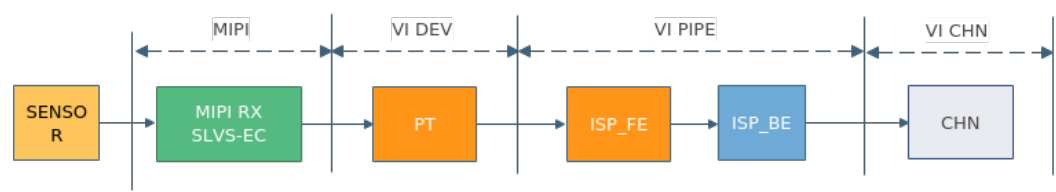
1.9.9.1 功能说明

视频输入（VI）模块实现的功能：通过MIPI Rx(含MIPI CSI接口、LVDS接口和HISPI接口)，SLVS-EC等接口接收视频数据。VI将接收到的数据存入到指定的内存区域，在此过程中，VI可以对接收到的原始视频图像数据进行裁剪、防抖、颜色优化、亮度优化、噪声去除等处理，并输出YUV或RAW格式的图像数据，实现视频数据的获取。

说明

当前版本仅配套支持MIPI CSI接口以及SLVS-EC接口。

图 1-14 VI 功能框图



VI从功能上划分了VI DEV（输入设备）、VI PIPE（图示为物理PIPE，虚拟PIPE只包含ISP_BE）、VI CHN（图示为物理通道）三个层级。各层级所支持的设备、PIPE、通道数量说明如下：

-	DEV	PHY_PIPE	VIR_PIPE	PHY_CHN	EXT_CHN
Atlas 200/500 A2 推理产品	4	4	8	4	0

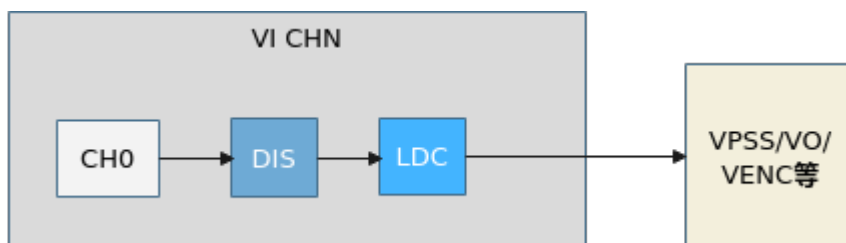
1.9.9.2 约束说明

- 最大并发4路接入，总体性能规格为4096W*2160H@45fps。
- 单路视频输入最大分辨率为16384*16384，最小分辨率为120*120。在开启不同功能后，分辨率规格会有额外约束，具体如下：

- 开启视频防抖后，最小分辨率为1280*720，最大分辨率为4096*2160。
- 开启畸变矫正后，最小分辨率为640*480，最大分辨率为8192*8192。
- 开启RAW图压缩功能后，最大分辨率规格为4096*4096。
- 开启RAW图帧压缩功能后，最小分辨率为640*480。
- 当Sensor模组直出YUV图像时，仅pipe2和pipe3支持YUV422 8bit位宽图像接入，宽度取值范围[560, 3840]，高度取值范围[300, 2160]。
- 所有输入和输出分辨率，宽度必须以4像素对齐，高度必须4像素对齐。
- 支持WDR，最多支持2帧WDR，支持多种WDR时序。
- 支持处理的数据位宽为：8bit、10bit、12bit、14bit。
- VI的接口，除pipe、chn取流相关的接口（hi_mpi_vi_get_pipe_frame、hi_mpi_vi_release_pipe_frame、hi_mpi_vi_pipe_get_buffer、hi_mpi_vi_pipe_release_buffer、hi_mpi_vi_get_chn_frame、hi_mpi_vi_release_chn_frame）外，其他接口都不支持多线程的并发调用；所有接口，都不支持多进程调用。

1.9.9.3 基本概念

- VI设备/PIPE/通道
 - 视频输入设备VI DEV
视频输入设备支持若干种时序输入，负责对时序进行解析。
 - 视频输入处理VI PIPE
 - 视频输入处理物理PIPE
视频输入处理PIPE绑定在视频输入设备后端，负责设备解析后的图像算法处理的ISP FE和ISP BE部分，对图像数据进行流水线处理，输出YUV图像格式给通道。
 - 视频输入处理虚拟PIPE
视频输入处理虚拟PIPE不绑定设备，负责其他模块或用户发送过来的数据再处理，不支持PIPE中的ISP_FE，直接从PIPE中的ISP_BE开始。
 - 视频通道VI CHN
 - 视频物理通道PHY_CHN
物理通道负责通过不同的硬件加速单元，实现DIS、LDC等算法处理，并将最终处理后的数据输出到内存中。通道功能框图如下：



说明

开启DIS、LDC等功能后，需要在调用接口[hi_mpi_vi_set_chn_attr](#)时，每个功能额外分配1块VB资源，即chn属性的depth要额外+1。

- 视频扩展通道EXT_CHN

当前版本不支持。

● 绑定关系

为了将前端的Camera模组视频流通过MIPI接口接入到VI中处理，在芯片板级连线上，首先要满足DEV设备和MIPI 设备之间的绑定约束关系，其次为了软件侧的灵活调度编排，用户可自行配置调整DEV设备和PIPE之间的绑定关系。

- DEV和MIPI设备的绑定关系

VI DEV与MIPI RX/SLVS-EC设备的绑定关系固定，不可配置。对应关系如下：

表 1-51 VI DEV 与 MIPI RX/SLVS-EC 接口的绑定关系

VI DEV	MIPI RX	SLVS-EC
0	0	0
1	1	1
2	2	X
3	3	X

- DEV和PIPE的绑定关系

- 每个PIPE都可以与任意Dev通过接口[hi_mpi_vi_set_dev_bind_pipe](#)设置绑定关系，且必须在调用接口[hi_mpi_vi_create_pipe](#)创建PIPE前绑定，不能动态修改绑定关系。
- 线性模式（非WDR模式）下只需要将一个物理PIPE绑定一个Dev；WDR模式下需要多个物理PIPE绑定同一个Dev，且只有绑定到Dev的第1条物理PIPE的通道有数据输出，其他绑定的物理PIPE的通道没有图像数据输出。例如2合1WDR模式，我们需要将PIPE0、PIPE1绑定到同一个Dev0设备上，只有PIPE0上的通道有数据输出，PIPE1上的通道无数据输出。

● 镜头畸变校正（LDC）

镜头畸变校正，一些低端镜头容易产生图像畸变，需要根据畸变程度对其图像进行校正。

● 视频防抖（DIS）

DIS模块通过比较当前图像与前两帧图像采用不同自由度的防抖算法计算出当前图像在各个轴方向上的抖动偏移向量，然后根据抖动偏移向量对当前图像进行校正，从而起到防抖的效果。

● 低延时

低延时指图像写出指定的行数到DDR后，VI上报一个中断，把图像发给后端模块处理，可以减少延时，且硬件会有机制保证图像是先写后读，不会出现读图像错误。ISP_BE后接3DNR、DIS、LDC支持低延时功能。

● 从模式

部分Sensor需要工作在从模式，此时需要由外部提供行场曝光同步信息，需要使用VI的从模式模块。用户需要根据Sensor管脚的连线和下表确定使用哪个从模式模块，然后选择对应的物理PIPE号创建物理PIPE，否则会没有数据。

表 1-52 从模式与 PIPE 的对应关系表

从模式编号	可连接Sensor数目	管脚名称	建议使用PIPE的编号
0	2	SENSOR_VS0 SENSOR_HS0	0、1
1	2	SENSOR_VS1 SENSOR_HS1	2、3

- VI功能涉及的缩略语如下：

表 1-53 缩略语列表

缩略语	全称	含义
VI	Video Input	视频输入
WDR	Wide Dynamic Range	宽动态
BE	Backend	ISP pipeline中FPN及之后的部分
FE	Frontend	ISP pipeline中FPN算法之前的部分
DIS	Digital Image Stabilization	数字防抖
GME	Global Motion Estimation	全局运动估计
GYRO	Gyroscope	陀螺仪
LDC	Lens Distortion Correction	镜头畸变矫正
MIPI	Mobile Industry Processor Interface	移动行业处理器界面
LVDS	Low-Voltage Differential Signal	低压差分信号
HISPI	High-Speed Serial Pixel Interface	高速串行像素接口
SLVS-EC	Scalable Low-Voltage Signaling interface with an Embedded Clock	可伸缩的、含嵌入式时钟的低电压信号接口
IPC	IP camera	网络摄像机
DV	Digital Visual	数字摄像

缩略语	全称	含义
PMF	Perspective Mapping Function	投影变换映射函数
MPP	Media Processing Platform	媒体处理平台
ISP	Image Signal Processor	图像信号处理器
Gyro DIS	Gyroscope Digital Image Stabilization	陀螺仪数字防抖

1.9.9.4 hi_mpi_vi_set_dev_bind_pipe

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

一对多设置VI设备与物理PIPE的绑定关系。

约束说明

- 必须在调用[hi_mpi_vi_enable_dev](#)接口使用VI设备后、调用[hi_mpi_vi_create_pipe](#)接口创建PIPE之前才能调用本接口绑定物理PIPE。
- 不支持绑定已经创建的PIPE。
- 建议将带宽高的VI设备绑定到pipe0或pipe1上。

函数原型

[hi_s32](#) hi_mpi_vi_set_dev_bind_pipe([hi_vi_dev](#) vi_dev, const [hi_vi_dev_bind_pipe](#) *dev_bind_pipe)

参数说明

参数名	输入/输出	说明
vi_dev	输入	VI设备号。 取值范围：[0, 4)。

参数名	输入/输出	说明
dev_bind_pipe	输入	VI物理PIPE号。 取值范围：[0, 4)。 WDR模式下，设置绑定关系时，必须pipe0和pipe1一组，pipe2和pipe3一组，即将hi_vi_dev_bind_pipe结构体内的num参数设置为2，将hi_vi_dev_bind_pipe结构体内的pipe_id数组内的元素设置为0、1两个值，或者2、3两个值。

返回值说明

- 0：成功
- 非0：失败，参见[1.9.21.4 VI视频输入返回码](#)。

参考资源

接口调用流程，参见[视频数据获取功能](#)。

1.9.9.5 hi_mpi_vi_get_dev_bind_pipe

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

获取VI设备与物理PIPE的绑定关系。

函数原型

[hi_s32](#) hi_mpi_vi_get_dev_bind_pipe([hi_vi_dev](#) vi_dev, [hi_vi_dev_bind_pipe](#) *dev_bind_pipe)

参数说明

参数名	输入/输出	说明
vi_dev	输入	VI设备号。 取值范围：[0, 4)。
dev_bind_pipe	输出	VI 物理PIPE号。 取值范围：[0, 4)。

返回值说明

- 0：成功
- 非0：失败，参见1.9.21.4 VI视频输入返回码。

1.9.9.6 hi_mpi_vi_set_dev_attr

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

设置VI设备属性。

约束说明

在调用前要保证VI设备处于禁用状态。如果VI设备已处于开启状态，可以使用 [hi_mpi_vi_disable_dev](#)来禁用设备。

函数原型

[hi_s32](#) hi_mpi_vi_set_dev_attr([hi_vi_dev](#) vi_dev, const [hi_vi_dev_attr](#) *dev_attr)

参数说明

参数名	输入/输出	说明
vi_dev	输入	VI设备号。 取值范围：[0, 4)。
dev_attr	输入	VI设备属性指针。静态属性。 参数dev_attr主要用来配置指定VI设备的视频接口模式、数据类型、图像宽高等属性，用于与外围Sensor对接。

返回值说明

- 0：成功
- 非0：失败，参见1.9.21.4 VI视频输入返回码。

参考资源

接口调用流程，参见视频数据获取功能。

1.9.9.7 hi_mpi_vi_get_dev_attr

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

获取VI设备属性。

约束说明

如果未设置VI设备属性，该接口将返回失败。

函数原型

```
hi_s32 hi_mpi_vi_get_dev_attr(hi_vi_dev vi_dev, hi_vi_dev_attr *dev_attr)
```

参数说明

参数名	输入/输出	说明
vi_dev	输入	VI设备号。 取值范围：[0, 4)。
dev_attr	输出	VI设备属性指针。

返回值说明

- 0：成功
- 非0：失败，参见[1.9.21.4 VI视频输入返回码](#)。

1.9.9.8 hi_mpi_vi_enable_dev

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

启用VI设备。

约束说明

- 启用VI设备前，必须先调用[hi_mpi_vi_set_dev_attr](#)接口设置设备属性，否则返回失败。

- 可重复启用VI设备，不返回失败。

函数原型

hi_s32 hi_mpi_vi_enable_dev(**hi_vi_dev** vi_dev)

参数说明

参数名	输入/输出	说明
vi_dev	输入	VI设备号。 取值范围：[0, 4)。

返回值说明

- 0：成功
- 非0：失败，参见[1.9.21.4 VI视频输入返回码](#)。

参考资源

接口调用流程，参见[视频数据获取功能](#)。

1.9.9.9 hi_mpi_vi_disable_dev

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

禁用VI设备。

约束说明

- 需先调用[hi_mpi_vi_destroy_pipe](#)接口销毁所有与该VI设备绑定的物理PIPE后，再禁用VI设备。
- 可重复禁用，不返回失败。
- 禁用VI设备后，VI设备处理完全关闭的状态，在下次调用[hi_mpi_vi_enable_dev](#)接口启用VI设备前，需要重新调用[hi_mpi_vi_set_dev_attr](#)接口设置该设备的属性。

函数原型

hi_s32 hi_mpi_vi_disable_dev(**hi_vi_dev** vi_dev)

参数说明

参数名	输入/输出	说明
vi_dev	输入	VI设备号。 取值范围：[0, 4)。

返回值说明

- 0：成功
- 非0：失败，参见[1.9.21.4 VI视频输入返回码](#)。

参考资源

接口调用流程，参见[视频数据获取功能](#)。

1.9.9.10 hi_mpi_vi_create_pipe

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

创建一个VI PIPE。

约束说明

- 物理PIPE属性中的size、pixel_format、bit_width等必须与VI设备的属性保持一致，否则会出现错误。
- 同一个PIPE号，不支持重复创建。
- 不管输入图像宽度多大，只要在PIPE所支持宽度规格内，都可支持数据压缩；仅pipe0和pipe1支持压缩；当输入图像大于4096时，不支持压缩。
- WDR模式下，需要创建多个物理PIPE绑定到同一个开了WDR的设备上，当进行切换时，需要把所有绑定到该设备的物理PIPE销毁再重建。不能使用上次用过但未销毁的物理PIPE，否则可能造成错误。

函数原型

[hi_s32](#) hi_mpi_vi_create_pipe([hi_vi_pipe](#) vi_pipe, [hi_vi_pipe_attr](#) *pipe_attr)

参数说明

参数名	输入/输出	说明
vi_pipe	输入	PIPE号。 取值范围：[0, 12)。
pipe_attr	输入	PIPE的属性结构体指针。 静态属性。

返回值说明

- 0：成功
- 非0：失败，参见[1.9.21.4 VI视频输入返回码](#)。

参考资源

接口调用流程，参见[视频数据获取功能](#)。

1.9.9.11 hi_mpi_vi_destroy_pipe

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

销毁一个VI PIPE。

约束说明

- 使用本接口前，需先调用[hi_mpi_vi_stop_pipe](#)停止PIPE。
- 在未创建PIPE或重复销毁PIPE时，调用本接口，将提示PIPE不存在。

函数原型

[hi_s32](#) hi_mpi_vi_destroy_pipe([hi_vi_pipe](#) vi_pipe)

参数说明

参数名	输入/输出	说明
vi_pipe	输入	PIPE号。 取值范围：[0, 12)。

返回值说明

- 0：成功
- 非0：失败，参见1.9.21.4 VI视频输入返回码。

参考资源

接口调用流程，参见视频数据获取功能。

1.9.9.12 hi_mpi_vi_set_vi_vpss_mode

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

Atlas 200/500 A2推理产品，不支持该接口。

函数功能

设置VI模块、VPSS模块的工作模式，包括在线模式、离线模式。

约束说明

必须在调用hi_mpi_sys_init接口后、创建VI PIPE和VPSS组前调用本接口设置工作模式。各模式的说明如下表所示。

表 1-54 VI 和 VPSS 的工作模式说明

模式	VI工作模式（VI内部FE和BE之间）	VPSS工作模式（VI与VPSS之间）
在线模式	VI内部FE处理完成后，视频流不写出到DDR，直接送给BE继续处理。 VI在线模式下，并发规格降低为1路。 当前版本不支持。	VI处理完成后，视频流不写出到DDR，直接送给VPSS继续处理。该模式下，VI CHN不会产生图像数据，最终图像数据需从VPSS CHN中获取。
离线模式	VI内部FE处理完成后，视频流输出写出到DDR，BE从FE写出的DDR中读取图像继续处理。	VI完成后，视频流输出写出到DDR，并可从VI CHN获取图像数据。 离线模式下，用户也可通过hi_mpi_sys_bind接口设置VI和VPSS的绑定关系，此时被绑定的VPSS会自动从VI写出的DDR中读取图像继续处理。

函数原型

hi_s32 hi_mpi_vi_set_vi_vpss_mode(const **hi_vi_vpss_mode** *vi_vpss_mode)

参数说明

参数名	输入/输出	说明
vi_vpss_mode	输入	VI、VPSS工作模式。

返回值说明

- 0：成功
- 非0：失败，参见[1.9.21.4 VI视频输入返回码](#)。

1.9.9.13 hi_mpi_vi_get_vi_vpss_mode

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

Atlas 200/500 A2推理产品，不支持该接口。

函数功能

获取VI、VPSS当前的工作模式。

函数原型

hi_s32 hi_mpi_vi_get_vi_vpss_mode(**hi_vi_vpss_mode** *vi_vpss_mode)

参数说明

参数名	输入/输出	说明
vi_vpss_mode	输出	VI、VPSS工作模式。

返回值说明

- 0：成功
- 非0：失败，参见[1.9.21.4 VI视频输入返回码](#)。

1.9.9.14 hi_mpi_vi_set_pipe_attr

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

Atlas 200/500 A2推理产品，不支持该接口。

函数功能

设置VI PIPE属性。

约束说明

- 必须先调用[hi_mpi_vi_create_pipe](#)接口，创建VI PIPE后，才能调用该接口。

函数原型

hi_s32 hi_mpi_vi_set_pipe_attr([hi_vi_pipe](#) vi_pipe, const [hi_vi_pipe_attr](#) *pipe_attr)

参数说明

参数名	输入/输出	说明
vi_pipe	输入	PIPE号。 取值范围：[0, 12)。
pipe_attr	输入	PIPE的属性结构体指针。 静态属性。

返回值说明

- 0：成功
- 非0：失败，参见[1.9.21.4 VI视频输入返回码](#)。

参考资源

接口调用流程，参见[视频数据获取功能](#)。

1.9.9.15 hi_mpi_vi_get_pipe_attr

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

获取VI PIPE的属性。

约束说明

- 必须先调用[hi_mpi_vi_create_pipe](#)接口，创建VI PIPE后，才能调用该接口。
- PIPE属性必须合法，其中部分静态属性不可动态设置，具体请参见[hi_vi_pipe_attr](#)。

函数原型

```
hi_s32 hi_mpi_vi_get_pipe_attr(hi_vi_pipe vi_pipe, hi_vi_pipe_attr *pipe_attr)
```

参数说明

参数名	输入/输出	说明
vi_pipe	输入	PIPE号。 取值范围：[0, 12)。
pipe_attr	输入	PIPE的属性结构体指针。 静态属性。

返回值说明

- 0：成功
- 非0：失败，参见[1.9.21.4 VI视频输入返回码](#)。

参考资源

接口调用流程，参见[视频数据获取功能](#)。

1.9.9.16 hi_mpi_vi_start_pipe

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

启用VI PIPE。

约束说明

- 调用本接口前，必须先调用[hi_mpi_vi_create_pipe](#)接口创建PIPE。
- 支持重复调用本接口操作同一个PIPE。

函数原型

[hi_s32](#) hi_mpi_vi_start_pipe([hi_vi_pipe](#) vi_pipe)

参数说明

参数名	输入/输出	说明
vi_pipe	输入	PIPE号。 取值范围：[0, 12)。

返回值说明

- 0：成功
- 非0：失败，参见[1.9.21.4 VI视频输入返回码](#)。

参考资源

接口调用流程，参见[视频数据获取功能](#)。

1.9.9.17 hi_mpi_vi_stop_pipe

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

禁用VI PIPE。

约束说明

- 调用本接口前，必须先调用[hi_mpi_vi_create_pipe](#)接口创建PIPE。
- 支持重复调用该接口操作同一个PIPE。

函数原型

[hi_s32](#) hi_mpi_vi_stop_pipe([hi_vi_pipe](#) vi_pipe)

参数说明

参数名	输入/输出	说明
vi_pipe	输入	PIPE号。 取值范围：[0, 12)。

返回值说明

- 0：成功
- 非0：失败，参见[1.9.21.4 VI视频输入返回码](#)。

参考资源

接口调用流程，参见[视频数据获取功能](#)。

1.9.9.18 hi_mpi_vi_set_pipe_pre_crop

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

设置VI物理PIPE输入端的裁剪功能属性，在VI DEV输出给VI PIPE处理前进行裁剪。

约束说明

- 调用本接口前，必须先调用[hi_mpi_vi_create_pipe](#)接口创建PIPE。
- 需在调用[hi_mpi_vi_start_pipe](#)接口启动PIPE前调用本接口，启动PIPE后，不可调用本接口。

函数原型

hi_s32 hi_mpi_vi_set_pipe_pre_crop([hi_vi_pipe](#) vi_pipe, const [hi_crop_info](#) *crop_info)

参数说明

参数名	输入/输出	说明
vi_pipe	输入	PIPE号。 取值范围：[0, 12)。
crop_info	输入	裁剪区域属性。

返回值说明

- 0：成功
- 非0：失败，参见[1.9.21.4 VI视频输入返回码](#)。

1.9.9.19 hi_mpi_vi_get_pipe_pre_crop

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

获取VI 物理PIPE输入端的裁剪功能属性。

约束说明

- 调用本接口前，必须先调用[hi_mpi_vi_create_pipe](#)接口创建PIPE。
- 不支持虚拟PIPE。

函数原型

[hi_s32](#) hi_mpi_vi_get_pipe_pre_crop([hi_vi_pipe](#) vi_pipe, [hi_crop_info](#) *crop_info)

参数说明

参数名	输入/输出	说明
vi_pipe	输入	PIPE号。 取值范围：[0, 12)。
crop_info	输出	裁剪区域属性。

返回值说明

- 0：成功
- 非0：失败，参见[1.9.21.4 VI视频输入返回码](#)。

1.9.9.20 hi_mpi_vi_set_pipe_pre_be_crop

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

Atlas 200/500 A2推理产品，不支持该接口。

函数功能

设置VI PIPE内部BE输入端的裁剪功能属性。

约束说明

- 调用本接口前，必须先调用[hi_mpi_vi_create_pipe](#)接口创建PIPE。
- 需在调用[hi_mpi_vi_start_pipe](#)接口启动PIPE前调用本接口，启动PIPE后，不可调用本接口。
- 仅支持物理pipe，不支持虚拟pipe。

函数原型

[hi_s32](#) hi_mpi_vi_set_pipe_pre_be_crop([hi_vi_pipe](#) vi_pipe, const [hi_crop_info](#) *crop_info)

参数说明

参数名	输入/输出	说明
vi_pipe	输入	PIPE号。 取值范围：[0, 4)。
crop_info	输入	裁剪区域属性。

返回值说明

- 0：成功
- 非0：失败，参见[1.9.21.4 VI视频输入返回码](#)。

1.9.9.21 hi_mpi_vi_set_pipe_bind_strobe

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

Atlas 200/500 A2推理产品，不支持该接口。

函数功能

设置VI PIPE与对外strobe输入信号的绑定关系。

约束说明

- 调用本接口前，必须先调用[hi_mpi_vi_create_pipe](#)接口创建PIPE。
- 必须在调用[hi_mpi_vi_start_pipe](#)启动pipe前调用本接口。

函数原型

```
hi_s32 hi_mpi_vi_set_pipe_bind_strobe(hi_vi_pipe vi_pipe, hi_u32 strobe_id)
```

参数说明

参数名	输入/输出	说明
vi_pipe	输入	物理PIPE号。 取值范围：[0, 4)。
strobe_id	输入	strobe管脚编号。 取值范围：[0, 3)。

返回值说明

- 0：成功
- 非0：失败，参见[1.9.21.4 VI视频输入返回码](#)。

1.9.9.22 hi_mpi_vi_set_pipe_frame_dump_attr

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

设置VI 物理PIPE dump属性。

约束说明

- 调用本接口前，必须先调用[hi_mpi_vi_create_pipe](#)接口创建PIPE。
- 需要dump不同的像素格式数据，可以在dump前调用[hi_mpi_vi_create_pipe](#)接口创建pipe时指定像素格式，像素格式需要与Sensor模组实际输出的格式对齐。
- 需要dump裁剪后的图像，可以在dump前调用[hi_mpi_vi_set_pipe_pre_crop](#)接口设置pipe的裁剪。
- 需要dump压缩或者非压图像，可以在dump前调用[hi_mpi_vi_create_pipe](#)接口创建pipe时指定压缩类型。
- 不支持虚拟PIPE。

- 当pipe已经通过hi_mpi_vi_start_pipe接口启动后再调用当前接口，会清除内部缓存的用户未通过hi_mpi_vi_get_pipe_frame接口取走的图像。

函数原型

hi_s32 hi_mpi_vi_set_pipe_frame_dump_attr(**hi_vi_pipe** vi_pipe, const **hi_vi_frame_dump_attr** *dump_attr)

参数说明

参数名	输入/输出	说明
vi_pipe	输入	物理PIPE号。 取值范围：[0, 4)。
dump_attr	输入	VI 物理PIPE dump的属性。

返回值说明

- 0：成功
- 非0：失败，参见[1.9.21.4 VI视频输入返回码](#)。

参考资源

接口调用流程，参见[视频数据获取功能](#)。

1.9.9.23 hi_mpi_vi_get_pipe_frame_dump_attr

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

获取VI物理PIPE dump属性。

约束说明

- 调用本接口前，必须先调用[hi_mpi_vi_create_pipe](#)接口创建PIPE。
- 不支持虚拟PIPE不支持。

函数原型

hi_s32 hi_mpi_vi_get_pipe_frame_dump_attr(**hi_vi_pipe** vi_pipe, **hi_vi_frame_dump_attr** *dump_attr)

参数说明

参数名	输入/输出	说明
vi_pipe	输入	物理PIPE号。 取值范围：[0, 4)。
dump_attr	输出	VI 物理PIPE dump的属性。

返回值说明

- 0：成功
- 非0：失败，参见[1.9.21.4 VI视频输入返回码](#)。

参考资源

接口调用流程，参见[视频数据获取功能](#)。

1.9.9.24 hi_mpi_vi_get_pipe_frame

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

获取VI物理PIPE的数据。

约束说明

- 调用本接口前，必须先调用[hi_mpi_vi_create_pipe](#)接口创建PIPE。
- 调用[hi_mpi_vi_set_pipe_frame_dump_attr](#)接口开启dump、设置depth，否则内部不会预留临时的内存空间，最终导致获取不到图像数据。
- 不支持虚拟PIPE。

函数原型

[hi_s32](#) hi_mpi_vi_get_pipe_frame([hi_vi_pipe](#) vi_pipe, [hi_video_frame_info](#) *frame_info, [hi_s32](#) milli_sec)

参数说明

参数名	输入/输出	说明
vi_pipe	输入	物理PIPE号。 取值范围：[0, 4)
frame_info	输出	VI PIPE 数据信息的指针。
milli_sec	输入	超时参数，取值范围： • -1：表示阻塞模式； • 0：表示非阻塞模式； • >0：配置具体的超时时间，单位为毫秒（ms）。

返回值说明

- 0：成功
- 非0：失败，参见[1.9.21.4 VI视频输入返回码](#)。

参考资源

接口调用流程，参见[视频数据获取功能](#)。

1.9.9.25 hi_mpi_vi_release_pipe_frame

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

释放VI PIPE的数据。

约束说明

- 本接口需与 [hi_mpi_vi_get_pipe_frame](#)接口配对使用，接口调用次数保持一致，否则返回报错。
- 用户需先调用[hi_mpi_vi_get_pipe_frame](#)接口获取frame_info，再将获取到的frame_info传入本接口，否则可能造成释放不成功。

函数原型

hi_s32 hi_mpi_vi_release_pipe_frame([hi_vi_pipe](#) vi_pipe, const [hi_video_frame_info](#) *frame_info)

参数说明

参数名	输入/输出	说明
vi_pipe	输入	物理PIPE号。 取值范围：[0, 4)
frame_info	输入	VI PIPE数据信息的指针。

返回值说明

- 0：成功
- 非0：失败，参见[1.9.21.4 VI视频输入返回码](#)。

参考资源

接口调用流程，参见[视频数据获取功能](#)。

1.9.9.26 hi_mpi_vi_set_pipe_frame_source

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

设置VI PIPE数据的来源。

约束说明

- 调用本接口前，必须先调用[hi_mpi_vi_create_pipe](#)接口创建PIPE。

函数原型

```
hi_s32 hi_mpi_vi_set_pipe_frame_source(hi_vi_pipe vi_pipe, const  
hi_vi_pipe_frame_source source)
```

参数说明

参数名	输入/输出	说明
vi_pipe	输入	PIPE 号。 取值范围：[0, 12)。
source	输入	PIPE 的数据来源。

返回值说明

- 0：成功
- 非0：失败，参见[1.9.21.4 VI视频输入返回码](#)。

参考资源

接口调用流程，参见[视频数据获取功能](#)。

1.9.9.27 hi_mpi_vi_get_pipe_frame_source

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

获取VI PIPE数据的来源。

约束说明

- 调用本接口前，必须先调用[hi_mpi_vi_create_pipe](#)接口创建PIPE。

函数原型

hi_s32 hi_mpi_vi_get_pipe_frame_source(**hi_vi_pipe** vi_pipe,
hi_vi_pipe_frame_source *source)

参数说明

参数名	输入/输出	说明
vi_pipe	输入	PIPE 号。 取值范围：[0, 12)。
source	输出	PIPE 的数据来源。

返回值说明

- 0：成功
- 非0：失败，参见[1.9.21.4 VI视频输入返回码](#)。

1.9.9.28 hi_mpi_vi_send_pipe_yuv

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

通过VI PIPE发送YUV数据。

约束说明

- 调用本接口前，必须先调用[hi_mpi_vi_start_pipe](#)接口启动PIPE。
- 发送YUV数据前，需要先调用[hi_mpi_vi_set_pipe_frame_source](#)接口将PIPE的数据来源设置为HI_VI_PIPE_FRAME_SOURCE_USER，此时来自Sensor的数据不再会送给ISP BE处理，ISP BE只会处理用户送下来的帧数据。
- 通过本接口发送YUV数据时，对YUV数据有如下要求：
 - YUV数据的宽高必须与[hi_mpi_vi_create_pipe](#)接口创建PIPE时设置的size宽高保持一致。
 - 送下来的YUV帧信息必须为真实有效的通过[hi_mpi_vi_pipe_get_buffer](#)接口获取的帧信息。
 - YUV数据像素格式必须与[hi_mpi_vi_create_pipe](#)接口创建PIPE时设置的像素格式保持一致。
 - YUV数据field必须为HI_VIDEO_FIELD_FRAME。
 - YUV数据视频格式video_format必须为HI_VIDEO_FORMAT_LINEAR。
 - YUV数据压缩模式compress_mode必须为不压缩HI_COMPRESS_MODE_NONE。
 - YUV数据动态范围dynamic_range必须为HI_DYNAMIC_RANGE_SDR8、HI_DYNAMIC_RANGE_SDR10、HI_DYNAMIC_RANGE_XDR。
- WDR模式下，不支持调用本接口。
- 只能从BE送，不支持从FE送。

函数原型

[hi_s32](#) [hi_mpi_vi_send_pipe_yuv](#)([hi_vi_pipe](#) vi_pipe, const [hi_video_frame_info](#) *frame_info, [hi_s32](#) milli_sec)

参数说明

参数名	输入/输出	说明
vi_pipe	输入	PIPE 号。 取值范围：[0, 12)。
frame_info	输入	VI 帧信息结构指针。

参数名	输入/输出	说明
milli_sec	输入	超时参数，取值范围： • -1：表示阻塞模式； • 0：表示非阻塞模式； • >0：配置具体的超时时间，单位为毫秒（ms）。

返回值说明

- 0：成功
- 非0：失败，参见[1.9.21.4 VI视频输入返回码](#)。

参考资源

接口调用流程，参见[视频数据获取功能](#)。

1.9.9.29 hi_mpi_vi_send_pipe_raw

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

通过VI PIPE发送RAW数据。

约束说明

- 调用本接口前，必须先依次调用[hi_mpi_vi_create_pipe](#)接口创建PIPE、调用[hi_mpi_vi_start_pipe](#)接口启动PIPE，在调用[hi_mpi_vi_create_pipe](#)接口创建PIPE时，PIPE属性的isp_bypass必须为HI_FALSE。
- 发送RAW数据前，需要先调用[hi_mpi_vi_set_pipe_frame_source](#)接口将PIPE的数据来源设置为HI_VI_PIPE_FRAME_SOURCE_USER，此时来自Sensor的数据不再会送给ISP BE处理，ISP BE只会处理用户送下来的帧数据。
- WDR模式下，每个PIPE的RAW在frame_info存放的顺序必须与[hi_mpi_vi_set_dev_bind_pipe](#)接口绑定PIPE时的顺序保持一致，目的是保证长短曝光帧的顺序正确，每个PIPE的RAW的属性(宽高，像素格式，压缩等等)要与[hi_mpi_vi_create_pipe](#)接口创建时保持一致。
- 通过本接口发送RAW数据时，对RAW数据有如下要求：
 - 送下来的RAW帧信息必须为真实有效的通过[hi_mpi_vi_pipe_get_buffer](#)接口获取的帧信息。
 - RAW数据像素格式及图像大小必须与[hi_mpi_vi_create_pipe](#)接口创建PIPE时设置的像素格式及图像大小保持一致。

- RAW数据field必须为HI_VIDEO_FIELD_FRAME。
- RAW数据video_format必须为HI_VIDEO_FORMAT_LINEAR。
- RAW数据compress_mode仅支持HI_COMPRESS_MODE_NONE、HI_COMPRESS_MODE_LINE、HI_COMPRESS_MODE_FRAME。
- RAW数据动态范围dynamic_range必须为HI_DYNAMIC_RANGE_SDR8。

函数原型

```
hi_s32 hi_mpi_vi_send_pipe_raw(hi_u32 pipe_num, hi_vi_pipe pipe_id[], const hi_video_frame_info *frame_info[], hi_s32 milli_sec)
```

参数说明

参数名	输入/输出	说明
pipe_num	输入	PIPE数量。 当工作模式为线性模式时，该参数取值为 1；当工作模式为WDR模式时，该参数取值需要与WDR的PIPE数相同。具体模式可通过hi_mpi_vi_set_dev_attr接口设置WDR属性配置。
pipe_id	输入	PIPE号数组，数组内的元素个数与pipe_num参数值保证一致。
frame_info	输入	RAW数据信息。
milli_sec	输入	超时参数，取值范围： • -1：表示阻塞模式； • 0：表示非阻塞模式； • >0： 配置具体的超时时间，单位为毫秒（ms）。

返回值说明

- 0：成功
- 非0：失败，参见1.9.21.4 VI视频输入返回码。

参考资源

接口调用流程，参见视频数据获取功能。

1.9.9.30 hi_mpi_vi_pipe_get_buffer

- Atlas 200/300/500 推理产品，不支持该接口。
- Atlas 训练系列产品，不支持该接口。
- Atlas 推理系列产品，不支持该接口。
- Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

获取内部空闲的内存块，用于用户指定输入数据。

约束说明

- 调用本接口前，必须先调用[hi_mpi_vi_create_pipe](#)接口创建PIPE。

函数原型

hi_s32 hi_mpi_vi_pipe_get_buffer(**hi_vi_pipe** vi_pipe, **hi_video_frame_info** *frame_info)

参数说明

参数名	输入/输出	说明
vi_pipe	输入	PIPE 号。 取值范围：[0, 12)。
frame_info	输出	视频信息结构体指针。

返回值说明

- 0：成功
- 非0：失败，参见[1.9.21.4 VI视频输入返回码](#)。

参考资源

接口调用流程，参见[视频数据获取功能](#)。

1.9.9.31 hi_mpi_vi_pipe_release_buffer

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

释放通过[hi_mpi_vi_pipe_get_buffer](#)接口获取的视频帧相关资源。

函数原型

hi_s32 hi_mpi_vi_pipe_release_buffer(**hi_vi_pipe** vi_pipe, const **hi_video_frame_info** *frame_info)

参数说明

参数名	输入/输出	说明
vi_pipe	输入	PIPE 号。 取值范围：[0, 12)。
frame_info	输入	视频帧信息。

返回值说明

- 0：成功
- 非0：失败，参见[1.9.21.4 VI视频输入返回码](#)。

参考资源

接口调用流程，参见[视频数据获取功能](#)。

1.9.9.32 hi_mpi_vi_set_pipe_vc_number

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

Atlas 200/500 A2推理产品，不支持该接口。

函数功能

设置VI物理PIPE对接前端Sensor的VC号。

约束说明

- 必须调用[hi_mpi_vi_create_pipe](#)接口创建PIPE后、调用[hi_mpi_vi_start_pipe](#)接口前调用本接口。
- 有部分Sensor在2合1的WDR场景下输出的VC号为1、2等，如果不单独通过本接口指定VC号，则会与系统内部默认的VC号0、1不匹配，导致获取数据失败。
- 不支持虚拟PIPE。

函数原型

[hi_s32](#) hi_mpi_vi_set_pipe_vc_number([hi_vi_pipe](#) vi_pipe, [hi_u32](#) vc_number)

参数说明

参数名	输入/输出	说明
vi_pipe	输入	物理PIPE号。 取值范围：[0, 4)
vc_number	输入	VC号。 取值范围：[0, 4)

返回值说明

- 0：成功
- 非0：失败，参见[1.9.21.4 VI视频输入返回码](#)。

1.9.9.33 hi_mpi_vi_get_pipe_vc_number

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

Atlas 200/500 A2推理产品，不支持该接口。

函数功能

获取VI物理PIPE对接前端Sensor的VC号。

约束说明

- 必须在PIPE创建后，开启PIPE之前调用。
- 虚拟PIPE不支持。

函数原型

hi_s32 hi_mpi_vi_get_pipe_vc_number(**hi_vi_pipe** vi_pipe, **hi_u32** *vc_number)

参数说明

参数名	输入/输出	说明
vi_pipe	输入	物理PIPE号。 取值范围：[0, 4)
vc_number	输出	VC号。

返回值说明

- 0：成功
- 非0：失败，参见[1.9.21.4 VI视频输入返回码](#)。

1.9.9.34 hi_mpi_vi_get_pipe_fd

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

获取VI PIPE对应的文件句柄，可用于通过select或epoll方式等待可用的PIPE输出结果。

函数原型

[hi_s32](#) hi_mpi_vi_get_pipe_fd([hi_vi_pipe](#) vi_pipe)

参数说明

参数名	输入/输出	说明
vi_pipe	输入	PIPE号。 取值范围：[0, 12)。

返回值说明

当大于或等于0时，接口调用成功，返回对应句柄，否则表示接口调用失败。

参考资源

接口调用流程，参见[视频数据获取功能](#)。

1.9.9.35 hi_mpi_vi_set_chn_attr

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

设置VI通道属性。

约束说明

- 调用本接口前，必须先调用[hi_mpi_vi_create_pipe](#)接口创建PIPE。

函数原型

```
hi_s32 hi_mpi_vi_set_chn_attr(hi_vi_pipe vi_pipe, hi_vi_chn vi_chn, const hi_vi_chn_attr *chn_attr)
```

参数说明

参数名	输入/输出	说明
vi_pipe	输入	PIPE号。 取值范围：[0, 12)。
vi_chn	输入	VI通道号。 取值范围：[0, 1)
chn_attr	输入	VI通道属性结构体指针。

返回值说明

- 0：成功
- 非0：失败，参见[1.9.21.4 VI视频输入返回码](#)。

参考资源

接口调用流程，参见[视频数据获取功能](#)。

1.9.9.36 hi_mpi_vi_get_chn_attr

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

获取VI通道属性。

约束说明

- 调用本接口前，必须先调用[hi_mpi_vi_create_pipe](#)接口创建PIPE。

- 必须先调用[hi_mpi_vi_set_chn_attr](#)接口设置通道属性。

函数原型

[hi_s32](#) hi_mpi_vi_get_chn_attr([hi_vi_pipe](#) vi_pipe, [hi_vi_chn](#) vi_chn, [hi_vi_chn_attr](#) *chn_attr)

参数说明

参数名	输入/输出	说明
vi_pipe	输入	PIPE号。 取值范围：[0, 12)。
vi_chn	输入	VI通道号。 取值范围：[0, 1)
chn_attr	输出	VI通道属性结构体指针。

返回值说明

- 0：成功
- 非0：失败，参见[1.9.21.4 VI视频输入返回码](#)。

1.9.9.37 hi_mpi_vi_enable_chn

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

启用VI通道。

约束说明

- 调用本接口前，必须先调用[hi_mpi_vi_create_pipe](#)接口创建PIPE。
- 必须先调用[hi_mpi_vi_set_chn_attr](#)接口设置通道属性。
- 支持重复启用VI通道，不返回失败。

函数原型

[hi_s32](#) hi_mpi_vi_enable_chn([hi_vi_pipe](#) vi_pipe, [hi_vi_chn](#) vi_chn)

参数说明

参数名	输入/输出	说明
vi_pipe	输入	PIPE号。 取值范围：[0, 12)。
vi_chn	输入	VI通道号。 取值范围：[0, 1)

返回值说明

- 0：成功
- 非0：失败，参见[1.9.21.4 VI视频输入返回码](#)。

参考资源

接口调用流程，参见[视频数据获取功能](#)。

1.9.9.38 hi_mpi_vi_disable_chn

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

禁用VI通道。

约束说明

- 调用本接口前，必须先调用[hi_mpi_vi_create_pipe](#)接口创建PIPE。
- 支持重复禁用VI通道，不返回失败。

函数原型

hi_s32 hi_mpi_vi_disable_chn([hi_vi_pipe](#) vi_pipe, [hi_vi_chn](#) vi_chn)

参数说明

参数名	输入/输出	说明
vi_pipe	输入	PIPE号。 取值范围：[0, 12)。

参数名	输入/输出	说明
vi_chn	输入	VI通道号。 取值范围：[0, 1)

返回值说明

- 0：成功
- 非0：失败，参见[1.9.21.4 VI视频输入返回码](#)。

参考资源

接口调用流程，参见[视频数据获取功能](#)。

1.9.9.39 hi_mpi_vi_set_chn_dis_config

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

设置VI通道的DIS视频防抖功能配置信息。

约束说明

- 调用本接口前，必须先调用[hi_mpi_vi_create_pipe](#)接口创建PIPE。
- 调用本接口将hi_dis_config.scale设置为HI_TRUE且调用[hi_mpi_vi_set_chn_dis_attr](#)接口开启DIS功能时，DIS配置信息中的crop_ratio参数可以在DIS功能保持开启的情况下动态修改；其它情况要修改DIS配置信息参数时，必须先调用[hi_mpi_vi_set_chn_dis_attr](#)接口关闭DIS功能再修改参数，不可动态修改。

函数原型

hi_s32 hi_mpi_vi_set_chn_dis_config(**hi_vi_pipe** vi_pipe, **hi_vi_chn** vi_chn, const **hi_dis_config** *dis_config)

参数说明

参数名	输入/输出	说明
vi_pipe	输入	PIPE号。 取值范围：[0, 12)。

参数名	输入/输出	说明
vi_chn	输入	VI通道号。 取值范围：[0, 1)
dis_config	输入	DIS配置信息结构体指针。

返回值说明

- 0：成功
- 非0：失败，参见[1.9.21.4 VI视频输入返回码](#)。

1.9.9.40 hi_mpi_vi_get_chn_dis_config

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

获取VI通道的DIS配置信息。

约束说明

- 调用本接口前，必须先调用[hi_mpi_vi_create_pipe](#)接口创建PIPE。

函数原型

[hi_s32](#) hi_mpi_vi_get_chn_dis_config([hi_vi_pipe](#) vi_pipe, [hi_vi_chn](#) vi_chn, [hi_dis_config](#) *dis_config)

参数说明

参数名	输入/输出	说明
vi_pipe	输入	PIPE号。 取值范围：[0, 12)。
vi_chn	输入	VI通道号。 取值范围：[0, 1)
dis_config	输出	DIS配置信息结构体指针。

返回值说明

- 0：成功
- 非0：失败，参见[1.9.21.4 VI视频输入返回码](#)。

1.9.9.41 hi_mpi_vi_set_chn_dis_attr

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

设置VI通道的DIS属性。

约束说明

- 调用本接口前，必须先调用[hi_mpi_vi_create_pipe](#)接口创建PIPE。
- 必须先调用[hi_mpi_vi_enable_chn](#)接口开启VI通道、调用[hi_mpi_vi_set_chn_dis_config](#)接口设置DIS配置后，才能设置DIS属性。
- 调用本接口开启DIS功能，则不能调用[hi_mpi_vi_set_pipe_pre_crop](#)接口开启通道裁剪功能，如果要裁剪后的图像做DIS，建议在前端MIPI或者PIPE前端做裁剪。
- 修改DIS配置或修改通道属性，必须先关闭DIS。

函数原型

[hi_s32](#) hi_mpi_vi_set_chn_dis_attr([hi_vi_pipe](#) vi_pipe, [hi_vi_chn](#) vi_chn, const [hi_dis_attr](#) *dis_attr)

参数说明

参数名	输入/输出	说明
vi_pipe	输入	PIPE号。 取值范围：[0, 12)。
vi_chn	输入	VI通道号。 取值范围：[0, 1)
dis_attr	输入	DIS属性结构体指针。

返回值说明

- 0：成功
- 非0：失败，参见[1.9.21.4 VI视频输入返回码](#)。

1.9.9.42 hi_mpi_vi_get_chn_dis_attr

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

获取VI通道的DIS属性。

约束说明

- 调用本接口前，必须先调用[hi_mpi_vi_create_pipe](#)接口创建PIPE。
- 建议先调用[hi_mpi_vi_set_chn_dis_attr](#)接口设置DIS属性后，再使用该接口。

函数原型

hi_s32 hi_mpi_vi_get_chn_dis_attr(**hi_vi_pipe** vi_pipe, **hi_vi_chn** vi_chn, **hi_dis_attr** *dis_attr)

参数说明

参数名	输入/输出	说明
vi_pipe	输入	PIPE号。 取值范围：[0, 12)。
vi_chn	输入	VI通道号。 取值范围：[0, 1)
dis_attr	输出	DIS属性结构体指针。

返回值说明

- 0：成功
- 非0：失败，参见[1.9.21.4 VI视频输入返回码](#)。

1.9.9.43 hi_mpi_vi_set_chn_dis_param

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

设置VI通道的DIS可选参数，仅可调整GME模式下的防抖效果，GYRO模式下无效。

约束说明

- 调用本接口前，必须先调用[hi_mpi_vi_create_pipe](#)接口创建PIPE。

函数原型

```
hi_s32 hi_mpi_vi_set_chn_dis_param(hi_vi_pipe vi_pipe, hi_vi_chn vi_chn, const hi_dis_param *dis_param)
```

参数说明

参数名	输入/输出	说明
vi_pipe	输入	PIPE号。 取值范围：[0, 12)。
vi_chn	输入	VI通道号。 取值范围：[0, 1)
dis_param	输入	DIS可选参数结构体指针。

返回值说明

- 0：成功
- 非0：失败，参见[1.9.21.4 VI视频输入返回码](#)。

1.9.9.44 hi_mpi_vi_get_chn_dis_param

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

获取VI通道的DIS可选参数，仅在GME模式下有效，GYRO模式下无效。

约束说明

- 调用本接口前，必须先调用[hi_mpi_vi_create_pipe](#)接口创建PIPE。

函数原型

```
hi_s32 hi_mpi_vi_get_chn_dis_param(hi_vi_pipe vi_pipe, hi_vi_chn vi_chn,
hi_dis_param *dis_param)
```

参数说明

参数名	输入/输出	说明
vi_pipe	输入	PIPE号。 取值范围：[0, 12)。
vi_chn	输入	VI通道号。 取值范围：[0, 1)
dis_param	输出	DIS可选参数属性结构体指针。

返回值说明

- 0：成功
- 非0：失败，参见1.9.21.4 VI视频输入返回码。

1.9.9.45 hi_mpi_vi_set_chn_ldc_attr

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

设置VI镜头畸变校正（ LDC ）属性。

约束说明

- 调用本接口前，必须先调用hi_mpi_vi_create_pipe接口创建PIPE。
- 必须调用hi_mpi_vi_set_chn_attr接口设置通道属性后，再调用本接口。

函数原型

```
hi_s32 hi_mpi_vi_set_chn_ldc_attr(hi_vi_pipe vi_pipe, hi_vi_chn vi_chn, const
hi_vi_ldc_attr *ldc_attr)
```

参数说明

参数名	输入/输出	说明
vi_pipe	输入	PIPE号。 取值范围：[0, 12)。
vi_chn	输入	VI通道号。 取值范围：[0, 1)
ldc_attr	输入	LDC属性结构体指针。

返回值说明

- 0：成功
- 非0：失败，参见[1.9.21.4 VI视频输入返回码](#)。

1.9.9.46 hi_mpi_vi_get_chn_ldc_attr

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

获取VI镜头畸变校正（ LDC ）属性。

约束说明

- 调用本接口前，必须先调用[hi_mpi_vi_create_pipe](#)接口创建PIPE。

函数原型

[hi_s32](#) hi_mpi_vi_get_chn_ldc_attr([hi_vi_pipe](#) vi_pipe, [hi_vi_chn](#) vi_chn, [hi_vi_ldc_attr](#) *ldc_attr)

参数说明

参数名	输入/输出	说明
vi_pipe	输入	PIPE号。 取值范围：[0, 12)。
vi_chn	输入	VI通道号。 取值范围：[0, 1)

参数名	输入/输出	说明
ldc_attr	输入&输出	LDC属性结构体指针。 ldc_attr参数作为输入时，需要指定该参数结构体内的ldc_version，否则会返回报错。

返回值说明

- 0：成功
- 非0：失败，参见[1.9.21.4 VI视频输入返回码](#)。

1.9.9.47 hi_mpi_vi_get_chn_frame

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

从指定VI通道获取的视频图像信息，图像信息主要包括：图像的宽度、高度、像素格式、时间戳以及YUV各分量的地址等等。

约束说明

- 调用本接口前，必须先调用[hi_mpi_vi_create_pipe](#)接口创建PIPE。
- 通道已启用且通道队列深度不为0时，才能调用本接口获取到图像。
- 支持多次获取后再释放，但建议获取和释放接口配对使用。
- 获取的地址信息来自模块内部使用的临时内存，因此使用完之后，必须要调用[hi_mpi_vi_release_chn_frame](#)接口释放其内存。
- 用户可结合[hi_mpi_vi_get_chn_fd](#)接口，采用select/epoll方式等待可读图像信息，等到有可用的视频输入数据后再通过非阻塞方式使用hi_mpi_vi_get_chn_frame获取图像。

函数原型

```
hi_s32 hi_mpi_vi_get_chn_frame(hi_vi_pipe vi_pipe, hi_vi_chn vi_chn,
hi_video_frame_info *frame_info, hi_s32 milli_sec)
```

参数说明

参数名	输入/输出	说明
vi_pipe	输入	PIPE号。 取值范围：[0, 12)。
vi_chn	输入	VI通道号。 取值范围：[0, 1)
frame_info	输入	VI帧信息结构体。
milli_sec	输入	超时参数 。取值范围： -1：表示阻塞模式，程序一直等待，直到获取到图像才返回； 0：表示非阻塞模式； >0： 配置具体的超时时间，单位为毫秒（ms），在配置的超时时间内如果没有获取到图像，则超时返回。

返回值说明

- 0：成功
- 非0：失败，参见[1.9.21.4 VI视频输入返回码](#)。

参考资源

接口调用流程，参见[视频数据获取功能](#)。

1.9.9.48 hi_mpi_vi_release_chn_frame

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

释放一帧从VI通道获取的图像。

约束说明

- 本接口需与[hi_mpi_vi_get_chn_frame](#)接口配对使用，接口调用次数保持一致，否则返回报错。
- 用户需先调用[hi_mpi_vi_get_chn_frame](#)接口获取frame_info，再将获取到的frame_info传入本接口，否则可能造成释放不成功。

函数原型

```
hi_s32 hi_mpi_vi_release_chn_frame(hi_vi_pipe vi_pipe, hi_vi_chn vi_chn, const hi_video_frame_info *frame_info)
```

参数说明

参数名	输入/输出	说明
vi_pipe	输入	PIPE号。 取值范围：[0, 12)。
vi_chn	输入	VI通道号。 取值范围：[0, 1)
frame_info	输入	VI帧信息结构体。

返回值说明

- 0：成功
- 非0：失败，参见1.9.21.4 VI视频输入返回码。

参考资源

接口调用流程，参见视频数据获取功能。

1.9.9.49 hi_mpi_vi_get_chn_fd

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

获取VI通道文件描述符 。

函数原型

```
hi_s32 hi_mpi_vi_get_chn_fd(hi_vi_pipe vi_pipe, hi_vi_chn vi_chn)
```

参数说明

参数名	输入/输出	说明
vi_pipe	输入	PIPE号。 取值范围：[0, 12)。
vi_chn	输入	VI通道号。 取值范围：[0, 1)

返回值说明

当大于或等于0时，接口调用成功，返回对应句柄，否则表示接口调用失败。

参考资源

接口调用流程，参见视频数据获取功能。

1.9.9.50 hi_mpi_vi_set_chn_low_delay_attr

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

设置VI PIPE低延时属性。

约束说明

- 调用本接口前，必须先调用[hi_mpi_vi_create_pipe](#)接口创建PIPE。
- 为确保设置的低延时生效，需：
 - 调用[hi_mpi_sys_bind](#)接口绑定VI与VPSS。
 - 已开启视频防抖（DIS）、镜头畸变校正（LDC）中的任意一个功能或组合功能。

函数原型

hi_s32 hi_mpi_vi_set_chn_low_delay_attr([hi_vi_pipe](#) vi_pipe, [hi_vi_chn](#) vi_chn, const [hi_vi_low_delay_info](#) *delay_info)

参数说明

参数名	输入/输出	说明
vi_pipe	输入	VI PIPE号。 取值范围：[0, 12)。
vi_chn	输入	VI通道号。 取值范围：[0, 1)
delay_info	输入	低延时功能参数结构体指针。

返回值说明

- 0：成功
- 非0：失败，参见[1.9.21.4 VI视频输入返回码](#)。

1.9.9.51 hi_mpi_vi_get_chn_low_delay_attr

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

获取VI通道低延时属性。

约束说明

- 调用本接口前，必须先调用[hi_mpi_vi_create_pipe](#)接口创建PIPE。

函数原型

[hi_s32](#) hi_mpi_vi_get_chn_low_delay_attr([hi_vi_pipe](#) vi_pipe, [hi_vi_chn](#) vi_chn, [hi_vi_low_delay_info](#) *delay_info)

参数说明

参数名	输入/输出	说明
vi_pipe	输入	VI PIPE号。 取值范围：[0, 12)。
vi_chn	输入	VI通道号。 取值范围：[0, 1)

参数名	输入/输出	说明
delay_info	输出	低延时功能参数结构体指针。

返回值说明

- 0：成功
- 非0：失败，参见[1.9.21.4 VI视频输入返回码](#)。

1.9.10 Region 区域管理

1.9.10.1 功能及约束说明

功能说明

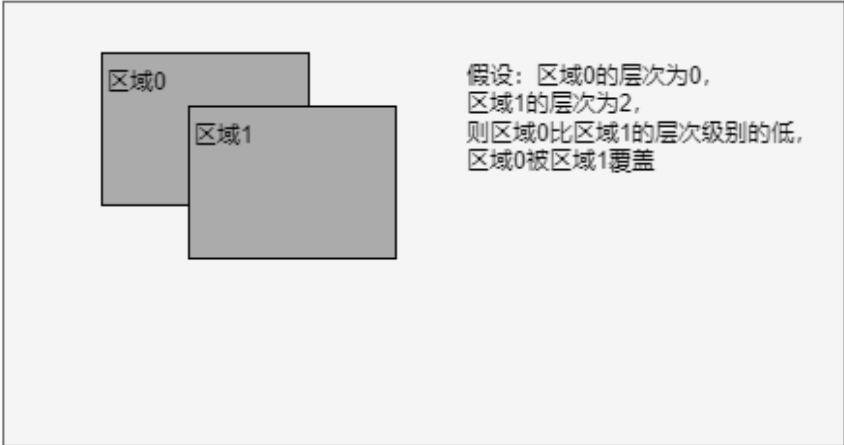
叠加在视频上的OSD (On Screen Display)和遮挡在视频上的色块统称为区域。**区域管理模块**，用于统一管理这些区域资源，用于在视频上显示一些特定信息（如通道号、时间戳等）、或在视频中填充色块用于遮挡。

例如，实际应用中，用户通过创建一个区域，通过调用[hi_mpi_rgn_attach_to_chn](#)接口，将该区域叠加到指定的一个或多个VPSS通道中，同时支持在每个通道中指定区域的显示属性，如位置、层次、透明度等。

基本概念

表 1-55 概念解释

概念	描述
区域类型	区域类型包括以下： • OVERLAY：视频叠加区域，其中区域支持位图的加载、背景色更新等功能。 • OVERLAYEX：扩展视频叠加区域，功能与 OVERLAY 类似，支持位图加载、背景色更新等。 • COVER：视频遮挡区域，其中区域支持纯色块遮挡。 • COVEREX：扩展视频遮挡区域，功能与 COVER 类似，支持纯色块遮挡。 • LINE：扩展的线条视频叠加区域，支持颜色、粗细等调节。 • MOSAIC：马赛克遮挡区域，支持精度调节。 • MOSAICEX：扩展的马赛克遮挡区域，支持精度调节。 • CORNER_RECTEX：角框区域，支持大小，颜色，粗细等调节。 其中，OVERLAYEX 类型的区域会绑定到VPSS的Channel上；MOSAIC、COVER 类型的区域会绑定到VPSS的Group上。VPSS的Group和Channel区别详见1.9.11.1 功能说明

概念	描述
区域层次	指区域的叠加级别，层次值越大，表示区域的显示级别越高。当发生重叠时，层次值大的将会覆盖层次值小的。 图 1-15 层次叠加示意图  假设：区域0的层次为0， 区域1的层次为2， 则区域0比区域1的层次级别的低， 区域0被区域1覆盖
位图填充	指将位图的内存值填充到区域内存空间中，位图将会从区域的左上角开始填充。当位图小于区域时，只能填充一部分内存，剩余部分保持原有值；位图大小等于区域时，将刚好全部填充；当位图大于区域时，位图只能将自身和区域一样大小的内存信息填充到区域中。位图填充仅针对OVERLAYEX区域类型有效。
区域属性	用户创建一个区域时，需要设置该区域的属性信息，包含OVERLAYEX包含像素格式、大小、背景色等。
通道显示属性	指区域在某通道的显示特征。 例如，MOSAIC的通道显示属性包含显示位置、区域大小、层次、mosaic块大小。当区域的通道显示属性中的is_show（区域是否显示）为TRUE时，表示区域会显示在该通道中；反之，表示区域在该通道中存在，但处于隐藏状态。

约束说明

表 1-56 region 支持的模块信息

类型	支持模块 (mod_id)	设备号 (dev_id) 取值范围	通道号 (chn_id) 取值范围
OVERLAY	当前不支持	-	-
COVER	VPSS (HI_ID_VPSS)	VPSS Group ID取 值范围：[0,255]	0

类型	支持模块 (mod_id)	设备号 (dev_id) 取值范围	通道号 (chn_id) 取值范围
OVERLAYEX	VPSS (HI_ID_VPSS)	VPSS Group ID取 值范围: [0,255]	[0,1]
COVEREX	当前不支持	-	-
LINE	当前不支持	-	-
MOSAIC	VPSS (HI_ID_VPSS)	VPSS Group ID取 值范围: [0,255]	0
MOSAICEX	当前不支持	-	-
CORNER_RECTEX	当前不支持	-	-

1.9.10.2 hi_mpi_rgn_create

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

Atlas 200/500 A2推理产品，Ascend EP标准形态下，不支持使用该接口。

函数功能

创建不同类型的区域。

约束说明

- 不支持重复创建。
- 创建区域时，本接口只进行基本的参数的检查，譬如：最小宽高，最大宽高等；当调用[hi_mpi_rgn_attach_to_chn](#)接口将区域叠加到通道上时，根据各通道模块支持类型的约束条件进行更加有针对性的参数检查，譬如支持的像素格式等。

函数原型

[hi_s32](#) hi_mpi_rgn_create([hi_rgn_handle](#) handle, const [hi_rgn_attr](#) *rgn_attr)

参数说明

参数名	输入/输出	说明
handle	输入	区域句柄号。 必须是未使用的handle号，由用户指定，意义等同于ID号。 取值范围: [0, 1024)。

参数名	输入/输出	说明
rgn_attr	输入	区域属性指针，不能为空。 当前仅支持创建COVER、MOSAIC、OVERLAYEX类型的区域。其它区域属性，如区域位置、层次等，需调用 hi_mpi_rgn_attach_to_chn 接口时指定。

返回值说明

- 0：获取数据成功
- 非0：失败，参见[19.18.5-区域管理返回码](#)

参考资源

接口调用流程，参见NVR场景视频解码、处理和显示。

1.9.10.3 hi_mpi_rgn_destroy

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

Atlas 200/500 A2推理产品，Ascend EP标准形态下，不支持使用该接口。

函数功能

销毁区域。

约束说明

区域必须已创建。

函数原型

[hi_s32](#) hi_mpi_rgn_destroy([hi_rgn_handle](#) handle)

参数说明

参数名	输入/输出	说明
handle	输入	区域句柄号。 取值范围：[0, 1024)。

返回值说明

- 0：获取数据成功
- 非0：失败，参见[19.18.5-区域管理返回码](#)

参考资源

接口调用流程，参见NVR场景视频解码、处理和显示。

1.9.10.4 hi_mpi_rgn_attach_to_chn

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

Atlas 200/500 A2推理产品，Ascend EP标准形态下，不支持使用该接口。

函数功能

将区域叠加到通道上。

约束说明

- 区域必须已创建。
- 支持多次调用本接口，但此接口不用于改变hi_mpi_rgn_create接口创建的区域属性。
- 如果将某个区域重复叠加到指定通道上，接口返回成功，但不会改变之前设置的区域通道显示属性。
- 最多支持将4个不同的MOSAIC类型的区域绑定到同一个通道上。
- 最多支持将8个不同的COVER类型的区域绑定到同一个通道上。
- 最多支持将8个不同的OVERLAYEX类型的区域绑定到同一个通道上。

函数原型

hi_s32 hi_mpi_rgn_attach_to_chn(**hi_rgn_handle** handle, const **hi_mpp_chn** *chn, const **hi_rgn_chn_attr** *chn_attr)

参数说明

参数名	输入/输出	说明
handle	输入	区域句柄号。 取值范围：[0, 1024)。
chn	输入	通道结构体指针，不能为空。 该参数的详细配置请参见 表2 region支持的模块信息 。

参数名	输入/输出	说明
chn_attr	输入	区域通道显示属性指针，不能为空。

返回值说明

- 0：获取数据成功
- 非0：失败，参见[19.18.5-区域管理返回码](#)

参考资源

接口调用流程，参见NVR场景视频解码、处理和显示。

1.9.10.5 hi_mpi_rgn_detach_from_chn

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

Atlas 200/500 A2推理产品，Ascend EP标准形态下，不支持使用该接口。

函数功能

从通道中删除叠加区域。

约束说明

- 区域必须已创建。
- 支持多次调用本接口。

函数原型

hi_s32 hi_mpi_rgn_detach_from_chn(**hi_rgn_handle** handle, const **hi_mpp_chn** *chn)

参数说明

参数名	输入/输出	说明
handle	输入	区域句柄号。 取值范围：[0, 1024)。
chn	输入	通道结构体指针，不能为空。 该参数的详细配置请参见 表2 region支持的模块信息 。

返回值说明

- 0: 获取数据成功
- 非0: 失败, 参见[19.18.5-区域管理返回码](#)

参考资源

接口调用流程, 参见NVR场景视频解码、处理和显示。

1.9.10.6 hi_mpi_rgn_set_display_attr

Atlas 200/300/500 推理产品, 不支持该接口。

Atlas 训练系列产品, 不支持该接口。

Atlas 推理系列产品, 不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品, 不支持该接口。

Atlas 200/500 A2推理产品, Ascend EP标准形态下, 不支持使用该接口。

函数功能

设置区域的通道显示属性。

约束说明

- 区域必须已创建。
- 建议先获取属性, 再设置。
- 必须先调用[hi_mpi_rgn_attach_to_chn](#)接口将区域叠加到通道上。

函数原型

[hi_s32](#) hi_mpi_rgn_set_display_attr([hi_rgn_handle](#) handle, const [hi_mpp_chn](#) *chn, const [hi_rgn_chn_attr](#) *chn_attr)

参数说明

参数名	输入/输出	说明
handle	输入	区域句柄号。 取值范围: [0, 1024)。
chn	输入	通道结构体指针, 不能为空。 该参数的详细配置请参见 表2 region支持的模块信息 。
chn_attr	输入	区域通道显示属性指针, 不能为空。 其中, 静态属性不能修改, 动态属性可以被修改。

返回值说明

- 0：获取数据成功
- 非0：失败，参见[19.18.5-区域管理返回码](#)

参考资源

接口调用流程，参见NVR场景视频解码、处理和显示。

1.9.10.7 hi_mpi_rgn_get_display_attr

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

Atlas 200/500 A2推理产品，Ascend EP标准形态下，不支持使用该接口。

函数功能

获取区域的通道显示属性。

约束说明

区域必须已创建。

函数原型

hi_s32 hi_mpi_rgn_get_display_attr(**hi_rgn_handle** handle, const **hi_mpp_chn** *chn, **hi_rgn_chn_attr** *chn_attr)

参数说明

参数名	输入/输出	说明
handle	输入	区域句柄号。 取值范围：[0, 1024)。
chn	输入	通道结构体指针，不能为空。 该参数的详细配置请参见 表2 region支持的模块信息 。
chn_attr	输出	区域通道显示属性指针，不能为空。

返回值说明

- 0：获取数据成功
- 非0：失败，参见[19.18.5-区域管理返回码](#)

参考资源

接口调用流程，参见NVR场景视频解码、处理和显示。

1.9.10.8 hi_mpi_rgn_get_canvas_info

- Atlas 200/300/500 推理产品，不支持该接口。
- Atlas 训练系列产品，不支持该接口。
- Atlas 推理系列产品，不支持该接口。
- Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。
- Atlas 200/500 A2推理产品，Ascend EP标准形态下，不支持使用该接口。

函数功能

获取区域的显示画布信息。

约束说明

- 区域必须已创建。
- 主要用于OVERLAYEX类型导入位图数据。
- 本接口用于获取区域对应的画布信息，在得到画布地址之后，用户可直接对画布进行操作，譬如：将bmp数据直接填写到该画布中。然后通过调用 [hi_mpi_rgn_update_canvas](#)接口更新显示画布数据。

函数原型

```
hi_s32 hi_mpi_rgn_get_canvas_info(hi_rgn_handle handle, hi_rgn_canvas_info *canvas_info)
```

参数说明

参数名	输入/输出	说明
handle	输入	区域句柄号。 取值范围：[0, 1024)。
canvas_info	输出	区域显示画布信息。

返回值说明

- 0：获取数据成功
- 非0：失败，参见[19.18.5-区域管理返回码](#)

参考资源

接口调用流程，参见NVR场景视频解码、处理和显示。

1.9.10.9 hi_mpi_rgn_update_canvas

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

Atlas 200/500 A2推理产品，Ascend EP标准形态下，不支持使用该接口。

函数功能

更新显示画布。

约束说明

- 区域必须已创建。
- 本接口配合[hi_mpi_rgn_get_canvas_info](#)使用。主要用于画布内存数据更新之后，进行画布切换显示。
- 每次调用本接口前，都必须先获取画布信息，然后在调用本接口进行更新。
- 本接口仅支持OVERLAYEX类型的区域。

函数原型

[hi_s32](#) hi_mpi_rgn_update_canvas([hi_rgn_handle](#) handle)

参数说明

参数名	输入/输出	说明
handle	输入	区域句柄号。 取值范围：[0, 1024)。

返回值说明

- 0：获取数据成功
- 非0：失败，参见[19.18.5-区域管理返回码](#)

参考资源

接口调用流程，参见NVR场景视频解码、处理和显示。

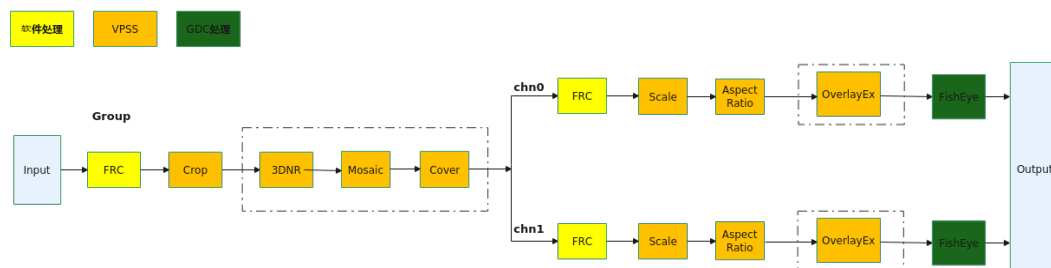
1.9.11 VPSS 视频处理功能

1.9.11.1 功能说明

VPSS模块接收VI/VDEC模块发送过来的图像，可对图像进行抠图、缩放、鱼眼矫正等处理，并实现同源输出两路不同分辨率的图像数据用于编码、预览或抓拍。

用户可调用[hi_mpi_sys_bind](#)接口，将VI/VDEC的视频流绑定到VPSS上，送给VPSS做进一步的处理，VPSS通过组Group以及组上的通道Channel来提供对应的处理能力，最终将图像输出给用户或下级模块(例如VO)。

图 1-16 功能框图



VPSS相关概念说明如下：

- Group
VPSS对用户提供的组（ Group ）的概念，各个组通过分时复用方式共享VPSS硬件资源。
- Channel
VPSS组的通道，一个VPSS组提供多个通道，每个通道具有独立的缩放、鱼眼矫正等功能。
- FRC（ Frame Rate Control ）
帧率控制，按控制点分为组帧率控制和通道帧率控制。
 - 组帧率控制，在[hi_mpi_vpss_create_grp](#)创建组时，可通过帧率控制属性控制各个Group对输入图像的处理。
 - 通道帧率控制，在[hi_mpi_vpss_set_chn_attr](#)设置组通道属性时，可通过帧率控制属性控制各个通道图像的处理。
- CROP
从图像中裁剪指定区域。
- 3DNR（ Three-dimensional Noise Reduction ）
3D去噪。通过参数配置，把图像中的高斯噪声去除，使得图像变得平滑，有助于降低编码码率。
- Mosaic
马赛克，对 VPSS 输出图像在指定区域填充马赛克块。该功能需要配合区域管理功能一起使用。
- Cover
视频遮挡区域，对 VPSS的输出图像填充纯色块。该功能需要配合区域管理功能一起使用。
遮挡区域坐标类型分为绝对坐标遮挡和相对坐标比例遮挡。
- Aspect Ratio
幅形比，指定输出画面相对于输入画面的宽高纵横比。
- OverlayEx
视频叠加区域，对 VPSS物理通道的输出图像叠加位图。该功能需要配合区域管理功能一起使用。

- FISHEYE
鱼眼矫正功能。
- Scale
缩放，对图像进行缩小放大 。 缩放倍数指水平、垂直各缩放多少倍。

1.9.11.2 约束说明

- 所有输入和输出分辨率的宽度必须以4像素对齐，高度必须4像素对齐。
- 视频防抖、畸变矫正下，鱼眼单矫正使用的是内部同一个硬件加速器，该加速器整体规格为4096*2160@45帧，所以仅开启鱼眼单区域矫正时，才能按4096*2160@45满规格运行，否则如果同时开启VI的视频防抖、畸变矫正功能，鱼眼规格降低，预计降一半。
- VPSS的接口，除chn取流相关的接口([hi_mpi_vpss_get_chn_frame](#)、[hi_mpi_vpss_release_chn_frame](#))外，其他接口都不支持多线程并发调用。
- 在不同Group ID范围内的VPSS Group支持能力存在差别，相关的能力如下表所示：

VPSS Group ID 范围	能力说明
Group ID取值在 [0, 256) 范围内	● 不支持3DNR功能。 ● 与区域管理模块配合使用，支持马赛克、Cover、OSD等功能。 ● 支持与VDEC、VO模块绑定，不支持与VI模块绑定(如何绑定请参考hi_mpi_sys_bind接口)。 ● 输入、输出图像所支持处理的最小分辨率为64*64，最大分辨率为4096*8192。 ● 若开启鱼眼矫正，输入图像分辨率约束为：宽在 [1920,4096] 范围内，高在 [1080,8192] 范围内。 ● 不开启鱼眼矫正下，VPSS相关Group的整体处理性能规格为1200FPS@1080P。若开启OverlayEx视频叠加区域功能，则对整体性能规格有影响，影响程度取决于该功能的输出图片分辨率。 ● 适用NVR场景，该场景支持多进程调用，具体使用流程请参见NVR场景视频解码、处理和显示。

VPSS Group ID 范围	能力说明
Group ID取值在 [256, 268) 范围内	- 支持3DNR功能。 - 不支持与区域管理模块配合使用。 - 支持与VI模块绑定，不支持与VDEC、VO模块绑定(如何绑定请参考hi_mpi_sys_bind接口)。 - 图像所支持处理的宽高有效范围为[64, 16384]，包括通过hi_mpi_vpss_set_grp_crop设置的裁剪后的宽高或者通过hi_mpi_vpss_set_chn_attr接口设置的图像输出宽高。当用户调用hi_mpi_vpss_create_grp接口，开启3DNR后，图像宽高需满足如下约束： - 不开启3DNR压缩：宽度[64, 16384]；高度[64, 8192]。 - 开启3DNR压缩：宽度[640, 16384]；高度[480, 4096]。 - 若开启鱼眼矫正，输入图像分辨率约束为：宽在[1920,8192]范围内，高在[1080,8192]范围内。 - VPSS整体处理性能规格为4096*2160@45帧。 - 适用Camera场景，该场景下VPSS需配套VI接口一起使用，VPSS相关业务必须和VI业务归属在同一个进程中实现，即只允许有一个进程实例在运行，具体使用流程请参见Camera场景视频数据获取和处理。

1.9.11.3 hi_mpi_vpss_create_grp

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

创建一个VPSS组。

约束说明

- 不支持重复创建。
- 如需使能3DNR，则高度不能超过8192，否则将自动bypass 3dnr算法。

函数原型

```
hi_s32 hi_mpi_vpss_create_grp(hi_vpss_grp grp, const hi_vpss_grp_attr *grp_attr);
```

参数说明

参数名	输入/输出	说明
grp	输入	VPSS组ID。 取值范围：[0, 264)。如果通过 hi_mpi_vpss_create_grp 接口的hi_vpss_grp_attr.nr_en参数开启3DNR功能，则VPSS组ID的取值范围为：[256, 268)。
grp_attr	输入	VPSS组属性指针。

返回值说明

- 0：成功
- 非0：失败，参见[6.10.7-VPSS视频处理子系统返回码](#)

参考资源

接口调用流程，参见Camera场景视频数据获取和处理、NVR场景视频解码、处理和显示。

1.9.11.4 hi_mpi_vpss_destroy_grp

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

销毁一个VPSS组。

约束说明

- VPSS组必须已创建，请参见[hi_mpi_vpss_create_grp](#)接口。
- 调用此接口之前，必须先调用[hi_mpi_vpss_stop_grp](#)接口禁用此组。
- 调用此接口时，会一直等待此VPSS组中的当前任务处理结束，才会真正销毁该VPSS组。

函数原型

[hi_s32](#) hi_mpi_vpss_destroy_grp([hi_vpss_grp](#) grp);

参数说明

参数名	输入/输出	说明
grp	输入	VPSS组ID。 取值范围：[0, 264)。如果通过 hi_mpi_vpss_create_grp 接口的hi_vpss_grp_attr.nr_en参数开启3DNR功能，则VPSS组ID的取值范围为：[256, 268)。

返回值说明

- 0：成功
- 非0：失败，参见[6.10.7-VPSS视频处理子系统返回码](#)

参考资源

接口调用流程，参见Camera场景视频数据获取和处理、NVR场景视频解码、处理和显示。

1.9.11.5 hi_mpi_vpss_start_grp

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

启用VPSS组。

约束说明

- VPSS组必须已创建，请参见[hi_mpi_vpss_create_grp](#)接口。
- 重复调用该接口启用同一个组返回成功。

函数原型

[hi_s32](#) hi_mpi_vpss_start_grp([hi_vpss_grp](#) grp);

参数说明

参数名	输入/输出	说明
grp	输入	VPSS组ID。 取值范围：[0, 264)。如果通过 hi_mpi_vpss_create_grp 接口的hi_vpss_grp_attr.nr_en参数开启3DNR功能，则VPSS组ID的取值范围为：[256, 268)。

返回值说明

- 0：成功
- 非0：失败，参见[6.10.7-VPSS视频处理子系统返回码](#)

参考资源

接口调用流程，参见Camera场景视频数据获取和处理、NVR场景视频解码、处理和显示。

1.9.11.6 hi_mpi_vpss_stop_grp

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

禁用VPSS组。

约束说明

- VPSS组必须已创建，请参见[hi_mpi_vpss_create_grp](#)接口。
- 重复调用本接口禁用同一个组返回成功。

函数原型

[hi_s32](#) hi_mpi_vpss_stop_grp([hi_vpss_grp](#) grp);

参数说明

参数名	输入/输出	说明
grp	输入	VPSS组ID。 取值范围：[0, 264)。如果通过 hi_mpi_vpss_create_grp 接口的hi_vpss_grp_attr.nr_en参数开启3DNR功能，则VPSS组ID的取值范围为：[256, 268)。

返回值说明

- 0：成功
- 非0：失败，参见[6.10.7-VPSS视频处理子系统返回码](#)

参考资源

接口调用流程，参见Camera场景视频数据获取和处理、NVR场景视频解码、处理和显示。

1.9.11.7 hi_mpi_vpss_set_grp_crop

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

设置VPSS组的裁剪参数。

约束说明

VPSS组必须已创建，请参见[hi_mpi_vpss_create_grp](#)接口。

函数原型

```
hi_s32 hi_mpi_vpss_set_grp_crop(hi_vpss_grp grp, const hi_vpss_crop_info *crop_info);
```

参数说明

参数名	输入/输出	说明
grp	输入	VPSS组ID。 取值范围：[0, 264)。如果通过 hi_mpi_vpss_create_grp 接口的hi_vpss_grp_attr.nr_en参数开启3DNR功能，则VPSS组ID的取值范围为：[256, 268)。
crop_info	输入	裁剪区域信息指针。

返回值说明

- 0：成功
- 非0：失败，参见[6.10.7-VPSS视频处理子系统返回码](#)

参考资源

接口调用流程，参见Camera场景视频数据获取和处理。

1.9.11.8 hi_mpi_vpss_get_grp_crop

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

获取VPSS组的裁剪参数。

函数原型

[hi_s32](#) hi_mpi_vpss_get_grp_crop([hi_vpss_grp](#) grp, [hi_vpss_crop_info](#) *crop_info);

参数说明

参数名	输入/输出	说明
grp	输入	VPSS组ID。 取值范围：[0, 264)。如果通过 hi_mpi_vpss_create_grp 接口的hi_vpss_grp_attr.nr_en参数开启3DNR功能，则VPSS组ID的取值范围为：[256, 268)。

参数名	输入/输出	说明
crop_info	输出	裁剪区域信息指针。

返回值说明

- 0：成功
- 非0：失败，参见[6.10.7-VPSS视频处理子系统返回码](#)

1.9.11.9 hi_mpi_vpss_set_grp_param

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

设置VPSS组参数。

约束说明

- VPSS组必须已创建，请参见[hi_mpi_vpss_create_grp](#)接口。
- 请务必先调用[hi_mpi_vpss_get_grp_param](#)获取当前参数，再进行设置操作。

函数原型

```
hi_s32 hi_mpi_vpss_set_grp_param(hi_vpss_grp grp, const hi_vpss_grp_param *grp_param);
```

参数说明

参数名	输入/输出	说明
grp	输入	VPSS组ID。 取值范围：[256, 268)
grp_param	输入	组参数设置。

返回值说明

- 0：成功
- 非0：失败，参见[6.10.7-VPSS视频处理子系统返回码](#)

参考资源

接口调用流程，参见Camera场景视频数据获取和处理。

1.9.11.10 hi_mpi_vpss_get_grp_param

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

获取VPSS组参数。

约束说明

VPSS组必须已创建，请参见[hi_mpi_vpss_create_grp](#)接口。

函数原型

hi_s32 hi_mpi_vpss_get_grp_param([hi_vpss_grp](#) grp, [hi_vpss_grp_param](#) *grp_param);

参数说明

参数名	输入/输出	说明
grp	输入	VPSS组ID。 取值范围：[256, 268)
grp_param	输入&输出	组参数设置。 grp_param参数作为输入时，需要指定该参数结构体内的nr_param.nr_version，否则会返回报错。

返回值说明

- 0：成功
- 非0：失败，参见[6.10.7-VPSS视频处理子系统返回码](#)

1.9.11.11 hi_mpi_vpss_set_grp_fisheye_cfg

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

设置VPSS组鱼眼镜头LMF (Lens Map Function) 参数。

约束说明

- VPSS组必须已创建，请参见[hi_mpi_vpss_create_grp](#)接口。
- LMF参数需要按照镜头厂商的推荐参数进行转换后再配置，如果镜头厂商未提供，则需要关闭LMF功能。
- 正确的LMF参数符合 $lmf\_coef[i + 1] \geq lmf\_coef[i] + 5$ && $lmf\_coef[i + 1] \leq lmf\_coef[i] + 31$ && $lmf\_coef[57] < 1024 < lmf\_coef[85]$ && $lmf\_coef[0] = 0$ 的规律，如果配置的参数不满足此规律则会报错，如果配置的参数有误则可能导致不出图等异常现象。

函数原型

```
hi_s32 hi_mpi_vpss_set_grp_fisheye_cfg(hi_vpss_grp grp, const hi_fisheye_cfg *fisheye_cfg);
```

参数说明

参数名	输入/输出	说明
grp	输入	VPSS组ID。 取值范围：[256, 268)。
fisheye_cfg	输入	鱼眼镜头LMF参数结构体指针。

返回值说明

- 0：成功
- 非0：失败，参见[6.10.7-VPSS视频处理子系统返回码](#)

参考资源

接口调用流程，参见Camera场景视频数据获取和处理。

1.9.11.12 hi_mpi_vpss_get_grp_fisheye_cfg

- Atlas 200/300/500 推理产品，不支持该接口。
- Atlas 训练系列产品，不支持该接口。
- Atlas 推理系列产品，不支持该接口。
- Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

获取VPSS组鱼眼镜头LMF参数。

约束说明

VPSS组必须已创建，请参见[hi_mpi_vpss_create_grp](#)接口。

函数原型

```
hi_s32 hi_mpi_vpss_get_grp_fisheye_cfg(hi_vpss_grp grp, hi_fisheye_cfg
*fisheye_cfg);
```

参数说明

参数名	输入/输出	说明
grp	输入	VPSS组ID。 取值范围：[256, 268)。
fisheye_cfg	输出	鱼眼镜头LMF参数结构体指针。

返回值说明

- 0：成功
- 非0：失败，参见[6.10.7-VPSS视频处理子系统返回码](#)

1.9.11.13 hi_mpi_vpss_set_chn_attr

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

设置VPSS物理通道属性。

约束说明

VPSS组必须已创建，请参见[hi_mpi_vpss_create_grp](#)接口。

函数原型

```
hi_s32 hi_mpi_vpss_set_chn_attr(hi_vpss_grp grp, hi_vpss_chn chn, const
hi_vpss_chn_attr *chn_attr);
```

参数说明

参数名	输入/输出	说明
grp	输入	VPSS组ID。 取值范围：[0, 264)。如果通过 hi_mpi_vpss_create_grp 接口的hi_vpss_grp_attr.nr_en参数开启3DNR功能，则VPSS组ID的取值范围为：[256, 268)。
chn	输入	VPSS 通道号。 取值范围：[0, 2)
chn_attr	输入	VPSS通道属性。

返回值说明

- 0：成功
- 非0：失败，参见[6.10.7-VPSS视频处理子系统返回码](#)

参考资源

接口调用流程，参见Camera场景视频数据获取和处理、NVR场景视频解码、处理和显示。

1.9.11.14 hi_mpi_vpss_get_chn_attr

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

获取VPSS物理通道属性。

约束说明

VPSS组必须已创建，请参见[hi_mpi_vpss_create_grp](#)接口。

函数原型

[hi_s32](#) hi_mpi_vpss_get_chn_attr([hi_vpss_grp](#) grp, [hi_vpss_chn](#) chn, [hi_vpss_chn_attr](#) *chn_attr);

参数说明

参数名	输入/输出	说明
grp	输入	VPSS组ID。 取值范围：[0, 264)。如果通过 hi_mpi_vpss_create_grp 接口的hi_vpss_grp_attr.nr_en参数开启3DNR功能，则VPSS组ID的取值范围为：[256, 268)。
chn	输入	VPSS 通道号。 取值范围：[0, 2)
chn_attr	输出	VPSS通道属性。

返回值说明

- 0：成功
- 非0：失败，参见[6.10.7-VPSS视频处理子系统返回码](#)

1.9.11.15 hi_mpi_vpss_enable_chn

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

启用VPSS通道。

约束说明

- 需先调用[hi_mpi_vpss_set_chn_attr](#)接口设置通道属性后，再调用本接口启用通道。
- 调用本接口重复启用同一个通道返回成功
- VPSS组必须已创建，请参见[hi_mpi_vpss_create_grp](#)接口。

函数原型

[hi_s32](#) hi_mpi_vpss_enable_chn([hi_vpss_grp](#) grp, [hi_vpss_chn](#) chn);

参数说明

参数名	输入/输出	说明
grp	输入	VPSS组ID。 取值范围：[0, 264)。如果通过 hi_mpi_vpss_create_grp 接口的hi_vpss_grp_attr.nr_en参数开启3DNR功能，则VPSS组ID的取值范围为：[256, 268)。
chn	输入	VPSS 通道号。 取值范围：[0, 2)

返回值说明

- 0：成功
- 非0：失败，参见[6.10.7-VPSS视频处理子系统返回码](#)

参考资源

接口调用流程，参见Camera场景视频数据获取和处理、NVR场景视频解码、处理和显示。

1.9.11.16 hi_mpi_vpss_disable_chn

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

禁用VPSS通道。

约束说明

- 调用本接口重复禁用同一个通道返回成功
- VPSS组必须已创建，请参见[hi_mpi_vpss_create_grp](#)接口。

函数原型

[hi_s32](#) hi_mpi_vpss_disable_chn([hi_vpss_grp](#) grp, [hi_vpss_chn](#) chn);

参数说明

参数名	输入/输出	说明
grp	输入	VPSS组ID。 取值范围：[0, 264)。如果通过 hi_mpi_vpss_create_grp 接口的hi_vpss_grp_attr.nr_en参数开启3DNR功能，则VPSS组ID的取值范围为：[256, 268)。
chn	输入	VPSS 通道号。 取值范围：[0, 2)

返回值说明

- 0：成功
- 非0：失败，参见[6.10.7-VPSS视频处理子系统返回码](#)

参考资源

接口调用流程，参见Camera场景视频数据获取和处理、NVR场景视频解码、处理和显示。

1.9.11.17 hi_mpi_vpss_get_chn_frame

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

用户从通道获取一帧处理完成的图像。

约束说明

- VPSS组必须已创建，请参见[hi_mpi_vpss_create_grp](#)接口。
- 只有在通过[hi_mpi_vpss_get_chn_attr](#)接口设置为USER模式下，并且队列深度不为0，才能获取到图像。
- 调用该接口获取图像，不会对后端绑定的模块有影响。如后端绑定VO显示，可以在显示过程中获取图像，VO仍正常显示，不会受到影响。
- 解码回放场景，由于不允许出现丢帧，VPSS只要有一个通道不处理新图像（通道已开启），则整个VPSS不处理新图像。例如说开启了通道0和通道1，两者都不绑定后端，通道图像队列长度都设为2，此时从通道0中最多获取出2帧已缓存的图像，因为通道1缓存2帧后未处理新图像，所以VPSS不会再处理新图像。

函数原型

```
hi_s32 hi_mpi_vpss_get_chn_frame(hi_vpss_grp grp, hi_vpss_chn chn,
hi_video_frame_info *frame_info, hi_s32 milli_sec);
```

参数说明

参数名	输入/输出	说明
grp	输入	VPSS组ID。 取值范围：[0, 264)。如果通过 hi_mpi_vpss_create_grp 接口的hi_vpss_grp_attr.nr_en 参数开启3DNR功能，则VPSS组ID的取值范围为：[256, 268)。
chn	输入	VPSS通道号。 取值范围：[0, 2)
frame_info	输出	图像信息。
milli_sec	输入	超时时间。 取值范围： • -1：表示阻塞模式，程序一直等待，直到获取到图像才返回； • 0：表示非阻塞模式； • >0： 配置具体的超时时间，单位为毫秒（ms）。在此指定时间内如果没有获取到图像，则超时返回。

返回值说明

- 0：成功
- 非0：失败，参见[6.10.7-VPSS视频处理子系统返回码](#)

参考资源

接口调用流程，参见Camera场景视频数据获取和处理、NVR场景视频解码、处理和显示。

1.9.11.18 hi_mpi_vpss_release_chn_frame

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

用户释放一帧通道图像。

约束说明

调用hi_mpi_vpss_get_chn_frame接口获取到图像数据后，需及时调用本接口释放资源。

函数原型

```
hi_s32 hi_mpi_vpss_release_chn_frame(hi_vpss_grp grp, hi_vpss_chn chn, const hi_video_frame_info *frame_info);
```

参数说明

参数名	输入/输出	说明
grp	输入	VPSS组ID。 取值范围：[0, 264)。如果通过 hi_mpi_vpss_create_grp接口的hi_vpss_grp_attr.nr_en 参数开启3DNR功能，则VPSS组ID的取值范围为：[256, 268)。
chn	输入	VPSS 通道号。 取值范围：[0, 2)
frame_info	输入	图像信息。

返回值说明

- 0：成功
- 非0：失败，参见6.10.7-VPSS视频处理子系统返回码

参考资源

接口调用流程，参见Camera场景视频数据获取和处理、NVR场景视频解码、处理和显示。

1.9.11.19 hi_mpi_vpss_set_chn_fisheye

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

设置VPSS物理通道鱼眼属性。

约束说明

- VPSS组必须已创建，请参见[hi_mpi_vpss_create_grp](#)接口。
- 调用本接口配置鱼眼属性前必须先调用[hi_mpi_vpss_set_chn_attr](#)接口配置对应物理通道的属性。
- 物理通道如果需要开启鱼眼矫正，建议在启用通道前配置鱼眼属性。
- 动态改变鱼眼属性时，鱼眼矫正后的输出图像大小应小于启用通道前配置的大小，如果未进行配置则应小于对应通道的宽高属性
- 物理通道开启了鱼眼矫正后，不支持动态改变通道Crop属性以及通道宽高属性。
- 输出图像的非校正区域是随机数据，可能会出现异常图像。
- 通道输出VIDEO_FORMAT_TILE_16x8格式时，不支持鱼眼。
- 如果在本接口中，hi_fisheye_attr属性开启了LMF参数功能，则必须调用VPSS的[hi_mpi_vpss_set_grp_fisheye_cfg](#)设置LMF（ Lens Map Function ）参数，否则进行鱼眼处理时会因为获取不到用户配置的LMF参数而报错。

函数原型

```
hi_s32 hi_mpi_vpss_set_chn_fisheye(hi_vpss_grp grp, hi_vpss_chn chn, const hi_fisheye_correction_attr *correction_attr);
```

参数说明

参数名	输入/输出	说明
grp	输入	VPSS组ID。 取值范围：[0, 264)。
chn	输入	VPSS 通道号。 取值范围：[0, 2)。
correction_attr	输入	VPSS物理通道的鱼眼属性结构体指针。

返回值说明

- 0：成功
- 非0：失败，参见[6.10.7-VPSS视频处理子系统返回码](#)

参考资源

接口调用流程，参见Camera场景视频数据获取和处理。

1.9.11.20 hi_mpi_vpss_get_chn_fisheye

- Atlas 200/300/500 推理产品，不支持该接口。
- Atlas 训练系列产品，不支持该接口。
- Atlas 推理系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

获取用户前一次调用hi_mpi_vpss_set_chn_fisheye接口设置的VPSS物理通道鱼眼属性。

约束说明

- VPSS组必须已创建，请参见[hi_mpi_vpss_create_grp](#)接口。
- 必须先设置才能获取，否则会获取失败。

函数原型

```
hi_s32 hi_mpi_vpss_get_chn_fisheye(hi_vpss_grp grp, hi_vpss_chn chn,
hi_fisheye_correction_attr *correction_attr);
```

参数说明

参数名	输入/输出	说明
grp	输入	VPSS组ID。 取值范围：[0, 264)。
chn	输入	VPSS 通道号。 取值范围：[0, 2)。
correction_attr	输出	VPSS物理通道的鱼眼属性结构体指针。

返回值说明

- 0：成功
- 非0：失败，参见[6.10.7-VPSS视频处理子系统返回码](#)

1.9.11.21 hi_mpi_vpss_get_chn_fd

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

获取VPSS通道对应的设备文件句柄。

函数原型

```
hi_s32 hi_mpi_vpss_get_chn_fd(hi_vpss_grp grp, hi_vpss_chn chn);
```

参数说明

参数名	输入/输出	说明
grp	输入	VPSS组ID。 取值范围：[0, 264)。如果通过 hi_mpi_vpss_create_grp 接口的hi_vpss_grp_attr.nr_en 参数开启3DNR功能，则VPSS组ID的取值范围为：[256, 268)。
chn	输入	VPSS 通道号。 取值范围：[0, 2)。

返回值说明

当大于或等于0时，接口调用成功，返回对应句柄，否则表示接口调用失败。

参考资源

接口调用流程，参见Camera场景视频数据获取和处理、NVR场景视频解码、处理和显示。

1.9.12 VO 视频输出功能

1.9.12.1 hi_mpi_vo_set_pub_attr

- Atlas 200/300/500 推理产品，不支持该接口。
- Atlas 训练系列产品，不支持该接口。
- Atlas 推理系列产品，不支持该接口。
- Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

配置视频输出设备的公共属性。

约束说明

- 视频输出设备属性，必须在执行[hi_mpi_vo_enable](#)前配置。
- 该接口暂不支持多进程。

函数原型

```
hi_s32 hi_mpi_vo_set_pub_attr(hi_vo_dev dev, const hi_vo_pub_attr*pub_attr)
```

参数说明

参数名	输入/输出	说明
dev	输入	视频输出设备号。 取值范围：[0, 2)。
pub_attr	输入	视频输出设备公共属性结构体指针。 静态属性。

返回值说明

- 0：成功
- 非0：失败，参见[1.9.21.7 VO视频输出返回码](#)

参考资源

接口调用流程，参见NVR场景视频解码、处理和显示。

1.9.12.2 hi_mpi_vo_enable

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

使用视频输出功能前必须先调用本接口使能视频输出设备。

约束说明

- 在调用设备使能前，必须对设备公共属性进行配置，否则返回设备未配置错误。
- 如果设备已经使能，调用此接口则返回未禁用错误，不支持重复使能。
- 为适应开机画面与正常操作界面间顺畅切换，此处需要检查VO（Video Output）硬件是否已经使能，如果已使能则返回成功，且沿用已有接口和时序配置。
- 如果希望更改VO的接口或时序配置，则需要先调用[hi_mpi_vo_disable](#)接口，强制禁用VO硬件后再使能。
- 该接口暂不支持多进程。

函数原型

[hi_s32](#) hi_mpi_vo_enable([hi_vo_dev](#) dev)

参数说明

参数名	输入/输出	说明
dev	输入	视频输出设备号。 取值范围：[0, 2)。

返回值说明

- 0：成功
- 非0：失败，参见[1.9.21.7 VO视频输出返回码](#)

参考资源

接口调用流程，参见NVR场景视频解码、处理和显示。

1.9.12.3 hi_mpi_vo_disable

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

禁用视频输出设备。

约束说明

- 设备禁用前必须先禁用该设备上的视频层。
- 设备禁用前，如果有使能回写功能，则必须先禁用回写功能。
- 设备禁用后需要重新调用[hi_mpi_vo_set_pub_attr](#)设置设备公共属性，才可使能设备。
- 该接口暂不支持多进程。

函数原型

[hi_s32](#) hi_mpi_vo_disable([hi_vo_dev](#) dev)

参数说明

参数名	输入/输出	说明
dev	输入	视频输出设备号。 取值范围：[0, 2)。

返回值说明

- 0：成功
- 非0：失败，参见[1.9.21.7 VO视频输出返回码](#)

参考资源

接口调用流程，参见NVR场景视频解码、处理和显示。

1.9.12.4 hi_mpi_vo_set_video_layer_attr

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

设置视频层属性。

约束说明

- 需要在视频层所绑定的设备处于使能状态时才能设置视频层属性。
- 设置视频层属性（SINGLE模式下除了layer_attr中display_rect的x,y）必须在视频层禁用的情况下进行。
- 该接口暂不支持多进程。

函数原型

hi_s32 hi_mpi_vo_set_video_layer_attr(**hi_vo_layer** layer, const **hi_vo_video_layer_attr** *layer_attr)

参数说明

参数名	输入/输出	说明
layer	输入	视频输出视频层号。 取值范围：[0, 9)。 不支持图形层和级联视频层。
layer_attr	输入	视频层属性结构体指针。

返回值说明

- 0：成功
- 非0：失败，参见[1.9.21.7 VO视频输出返回码](#)

参考资源

接口调用流程，参见NVR场景视频解码、处理和显示。

1.9.12.5 hi_mpi_vo_get_video_layer_attr

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

获取视频层属性。

约束说明

- 建议在设置视频层属性前先调用此接口获取视频层属性。
- 该接口暂不支持多进程。

函数原型

hi_s32 hi_mpi_vo_get_video_layer_attr(**hi_vo_layer** layer,
hi_vo_video_layer_attr *layer_attr)

参数说明

参数名	输入/输出	说明
layer	输入	视频输出视频层号。 取值范围：[0, 9)。 不支持图形层和级联视频层。
layer_attr	输出	视频层属性结构体指针。

返回值说明

- 0：成功
- 非0：失败，参见[1.9.21.7 VO视频输出返回码](#)

1.9.12.6 hi_mpi_vo_enable_video_layer

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

使能视频层。

约束说明

- 视频层使能前必须保证该视频层所绑定的设备处于使能状态。
- 视频层使能前必须保证该视频层已经配置。
- 该接口暂不支持多进程。

函数原型

[hi_s32](#) [hi_mpi_vo_enable_video_layer](#)([hi_vo_layer](#) layer)

参数说明

参数名	输入/输出	说明
layer	输入	视频输出视频层号。 取值范围：[0, 9)。 不支持图形层和级联视频层。

返回值说明

- 0：成功
- 非0：失败，参见[1.9.21.7 VO视频输出返回码](#)

参考资源

接口调用流程，参见NVR场景视频解码、处理和显示。

1.9.12.7 hi_mpi_vo_disable_video_layer

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

禁用视频层。

约束说明

- 视频层禁用前必须保证其上的通道全部禁用。

- 在禁用视频层时，非直通情况下，如果用户没有释放从VO获取的图像buffer资源，该接口会返回HI_ERR_VO_BUSY的错误码，表示VO创建的VB资源没有释放。多见于用户调用获取屏幕图像未释放的情况下。
- 在禁用视频层时，必须保证将该设备的回写功能禁用。
- 该接口暂不支持多进程。

函数原型

```
hi_s32 hi_mpi_vo_disable_video_layer(hi_vo_layer layer)
```

参数说明

参数名	输入/输出	说明
layer	输入	视频输出视频层号。 取值范围：[0, 9)。 不支持图形层和级联视频层。

返回值说明

- 0：成功
- 非0：失败，参见[1.9.21.7 VO视频输出返回码](#)

参考资源

接口调用流程，参见NVR场景视频解码、处理和显示。

1.9.12.8 hi_mpi_vo_set_chn_attr

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

配置指定视频输出通道的属性。

约束说明

- 通道显示区域不能超过视频层属性中设定的画布大小(img_size大小)。
- 该接口为动态设置接口，可在VO设备使能且相应视频层已配置的情况下调用。
- 设备的通道属性rect需2对齐，且宽高必须不小于32。起始点的纵坐标建议按照4对齐配置，否则可能会出现不可预料的图像质量问题。
- 如果设备有视频层放大的情况，rect是放大前视频层上的通道原始的起始位置和宽高，放大后的通道的显示起始位置和宽高会按视频层放大的比例偏移或放大。

- 该接口暂不支持多进程。

函数原型

```
hi_s32 hi_mpi_vo_set_chn_attr(hi_vo_layer layer, hi_vo_chn chn, const hi_vo_chn_attr *chn_attr)
```

参数说明

参数名	输入/输出	说明
layer	输入	视频输出视频层号。 取值范围：[0, 9)。 不支持图形层和级联视频层。
chn	输入	视频输出通道的通道号。 取值范围：[0, 64)。
chn_attr	输入	视频通道属性指针。

返回值说明

- 0：成功
- 非0：失败，参见[1.9.21.7 VO视频输出返回码](#)

参考资源

接口调用流程，参见NVR场景视频解码、处理和显示。

1.9.12.9 hi_mpi_vo_enable_chn

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

使能指定的视频输出通道。

约束说明

- 调用前必须保证视频层绑定关系存在，否则将返回失败。
- 调用前必须使能相应设备上的视频层。
- 通道使能前必须进行通道配置，否则返回通道未配置的错误。
- 允许重复使能同一视频输出通道，不返回失败。

- 该接口暂不支持多进程。

函数原型

```
hi_s32 hi_mpi_vo_enable_chn (hi_vo_layer layer, hi_vo_chn chn)
```

参数说明

参数名	输入/输出	说明
layer	输入	视频输出视频层号。 取值范围：[0, 9)。 不支持图形层和级联视频层。
chn	输入	视频输出通道的通道号。 取值范围：[0, 64)。

返回值说明

- 0：成功
- 非0：失败，参见[1.9.21.7 VO视频输出返回码](#)

参考资源

接口调用流程，参见NVR场景视频解码、处理和显示。

1.9.12.10 hi_mpi_vo_disable_chn

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

禁用指定的视频输出通道。

约束说明

- 允许重复禁用同一视频输出通道，不返回失败。
- 当高清设备的通道绑定VPSS（Video Processing Subsystem）时，建议先调用本接口停止通道后，再解绑定VO通道与VPSS通道的绑定关系，否则可能出现HI_ERR_VO_BUSY的超时返回错误。
- 该接口暂不支持多进程。

函数原型

```
hi_s32 hi_mpi_vo_disable_chn(hi_vo_layer layer, hi_vo_chn chn)
```

参数说明

参数名	输入/输出	说明
layer	输入	视频输出视频层号。 取值范围：[0, 9)。 不支持图形层和级联视频层。
chn	输入	视频输出通道的通道号。 取值范围：[0, 64)。

返回值说明

- 0：成功
- 非0：失败，参见1.9.21.7 VO视频输出返回码

参考资源

接口调用流程，参见NVR场景视频解码、处理和显示。

1.9.12.11 hi_mpi_vo_set_chn_param

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

本接口需与VPSS功能配合使用（调用hi_mpi_sys_bind接口绑定VO与VPSS），通过本接口配置幅型比参数，VPSS根据本接口配置的参数实现幅型比功能。

约束说明

- 本接口为动态设置接口，需在VO设备使能且相应视频层和通道属性已配置的情况下调用。
- 调用hi_mpi_vo_set_video_layer_attr接口将视频层的分割模式设置为HI_VO_PARTITION_MODE_MULTI，且调用VPSS的hi_mpi_vpss_set_chn_attr接口将VPSS通道的工作模式设置为HI_VPSS_CHN_MODE_AUTO模式、输入图像宽高比与通道宽高比不相等时，可通过本接口设置参数，否则参数设置无效。
- 该接口暂不支持多进程。

函数原型

```
hi_s32 hi_mpi_vo_set_chn_param(hi_vo_layer layer, hi_vo_chn chn, const hi_vo_chn_param *chn_param)
```

参数说明

参数名	输入/输出	说明
layer	输入	视频输出视频层号。 取值范围：[0, 9)。 不支持图形层和级联视频层。
chn	输入	视频输出通道的通道号。 取值范围：[0, 64)。
chn_param	输入	视频通道参数指针

返回值说明

- 0：成功
- 非0：失败，参见1.9.21.7 VO视频输出返回码

1.9.12.12 hi_mpi_vo_set_chn_frame_rate

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

设置指定视频输出通道的显示帧率。

约束说明

- 需先调用hi_mpi_vo_set_chn_attr接口设置VO相应视频输出通道属性，才能调用本接口，否则返回失败。
- 调用hi_mpi_vo_enable_chn接口会使通道重置为正常速度播放，因此，必须在通道使能后再调用本接口进行播放控制。
- 在通道禁用的情况下，调用hi_mpi_vo_set_chn_attr接口会将通道帧率重置为显示帧率。
通道帧率为通道处理图像数据的帧率，显示帧率为显示设备的帧率，显示帧率可以在调用hi_mpi_vo_set_video_layer_attr接口设置视频层属性时配置。
- VO与VPSS绑定场景下，该接口设置的帧率生效。
- 该接口暂不支持多进程。

函数原型

```
hi_s32 hi_mpi_vo_set_chn_frame_rate(hi_vo_layer layer, hi_vo_layer chn, hi_s32 frame_rate)
```

参数说明

参数名	输入/输出	说明
layer	输入	视频输出视频层号。 取值范围：[0, 9)。 layer参数设置为0、1时有效，设置其它值无效。
chn	输入	视频输出通道的通道号。 取值范围：[0, 64)。
frame_rate	输入	通道帧率，帧率可设置为Nx，其中N为[1,64]的任意整数，x为设备帧率。

返回值说明

- 0：成功
- 非0：失败，参见[1.9.21.7 VO视频输出返回码](#)

1.9.12.13 hi_mpi_vo_pause_chn

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

暂停指定的视频输出通道。

约束说明

- 调用[hi_mpi_vo_enable_chn](#)接口启用通道后，才可以调用本接口。
- 若调用[hi_mpi_vo_set_video_layer_attr](#)接口将视频层的分割模式设置为HI_VO_PARTITION_MODE_SINGLE时，则不支持暂停指定通道，调用本接口设置参数无效，但不影响视频显示结果。
- 调用[hi_mpi_vo_send_frame](#)接口将视频图像送入指定输出通道显示时，需至少2个图像帧轮转（即定义2个hi_video_frame_info结构体变量），否则暂停功能失效。
- 允许重复暂停同一通道，不返回失败。
- 该接口暂不支持多进程。

函数原型

hi_s32 hi_mpi_vo_pause_chn(int layer, int chn)

参数说明

参数名	输入/输出	说明
layer	输入	视频输出视频层号。 取值范围：[0, 9)。 layer参数设置为0、1时有效，设置其它值无效。
chn	输入	视频输出通道的通道号。 取值范围：[0, 64)。

返回值说明

- 0：成功
- 非0：失败，参见[1.9.21.7 VO视频输出返回码](#)

1.9.12.14 hi_mpi_vo_resume_chn

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

恢复指定的视频输出通道。

约束说明

- 调用[hi_mpi_vo_enable_chn](#)接口启用通道后，才可以调用本接口。
- 若调用[hi_mpi_vo_set_video_layer_attr](#)接口将视频层的分割模式设置为HI_VO_PARTITION_MODE_SINGLE时，则不支持恢复指定通道，调用本接口设置参数无效，但不影响视频显示结果。
- 调用[hi_mpi_vo_send_frame](#)接口将视频图像送入指定输出通道显示时，需至少2个图像帧轮转（即定义2个hi_video_frame_info结构体变量），否则恢复功能失效。
- 允许重复恢复同一通道，不返回失败。
- 该接口暂不支持多进程。

函数原型

hi_s32 hi_mpi_vo_resume_chn(int layer, int chn)

参数说明

参数名	输入/输出	说明
layer	输入	视频输出视频层号。 取值范围：[0, 9)。 layer参数设置为0、1时有效，设置其它值无效。
chn	输入	视频输出通道的通道号。 取值范围：[0, 64)。

返回值说明

- 0：成功
- 非0：失败，参见[1.9.21.7 VO视频输出返回码](#)

1.9.12.15 hi_mpi_vo_hide_chn

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

隐藏指定通道。

约束说明

- 调用该接口前需要由用户保证通道已经使能。
- 当VO与VPSS配合使用场景下，在资源释放时，建议先调用本接口隐藏通道，再调用[hi_mpi_sys_unbind](#)接口解除VO与VPSS的绑定关系，否则可能出现HI_ERR_VO_BUSY的错误。
- 隐藏通道后对该通道的送图/暂停/恢复等通道操作无效。必须在通道恢复后重新对通道进行送图操作，通道才可重新显示画面
- 该接口暂不支持多进程。

函数原型

[hi_s32](#) hi_mpi_vo_hide_chn([hi_vo_layer](#) layer, [hi_vo_chn](#) chn)

参数说明

参数名	输入/输出	说明
layer	输入	视频输出视频层号。 取值范围：[0, 9)。 layer参数设置为0、1时有效，设置其它值无效。
chn	输入	视频输出通道的通道号。 取值范围：[0, 64)。

返回值说明

- 0：成功
- 非0：失败，参见[1.9.21.7 VO视频输出返回码](#)

1.9.12.16 hi_mpi_vo_show_chn

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

显示指定通道。

约束说明

- 调用前需保证视频输出通道已使能，且视频输出通道所在的视频层已经使能。
- 默认情况下通道处于显示状态。
- VO在SINGLE模式下不支持显示指定通道，接口参数配置无效，不影响显示结果。
- 使能通道后，程序退出时必须由用户保证正确销毁全部申请的通道资源、vb相关内存资源，否则属于未定义行为，不保证后续送显。
- 该接口暂不支持多进程。

函数原型

```
hi_s32 hi_mpi_vo_show_chn(hi_vo_layer layer, hi_vo_chn chn)
```

参数说明

参数名	输入/输出	说明
layer	输入	视频输出视频层号。 取值范围：[0, 9)。 layer参数设置为0、1时有效，设置其它值无效。
chn	输入	视频输出通道的通道号。 取值范围：[0, 64)。

返回值说明

- 0：成功
- 非0：失败，参见[1.9.21.7 VO视频输出返回码](#)

1.9.12.17 hi_mpi_vo_send_frame

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

将视频图像送入指定输出通道显示。

约束说明

- 调用该接口前必须保证通道已经使能。
- 输入视频数据信息要符合VO数据的要求。宽和高需要与实际图像宽高相符，且均不能小于32，宽高要求以2对齐。
- single模式下只有一个通道可被显示，故调用[hi_mpi_vo_set_chn_attr](#)接口设置通道优先级后，只显示优先级最高的通道。
- 该接口暂不支持多进程。
- 当前只支持single模式，multi模式为预留，暂不拦截。

函数原型

[hi_s32](#) hi_mpi_vo_send_frame([hi_vo_layer](#) layer, [hi_vo_chn](#) chn, const [hi_video_frame_info](#) *frame_info, [hi_s32](#) milli_sec)

参数说明

参数名	输入/输出	说明
layer	输入	视频输出视频层号。 取值范围：[0, 9)。 不支持图形层和级联视频层。
chn	输入	视频输出通道的通道号。 取值范围：[0, 64)。
frame_info	输入	视频数据信息指针。 single模式下用户读取的图片对应宽高不可大于目标通道的宽高。
milli_sec	输入	超时参数，取值范围如下（当前仅支持0）： • -1：表示阻塞模式，一直等到有图像发送为止； • 0：表示非阻塞模式； • >0：配置具体的超时时间，超过该时间则不再等待，单位为毫秒（ms）。

返回值说明

- 0：成功
- 非0：失败，参见[1.9.21.7 VO视频输出返回码](#)

1.9.12.18 hi_mpi_vo_create_pool

函数功能

创建VO显示内存池。预留接口，暂不支持。

函数原型

hi_s32 hi_mpi_vo_create_pool(const **hi_u64** size)

参数说明

参数名	输入/输出	说明
size	输入	申请内存池大小

返回值说明

- 0：成功

- 非0：失败，参见[1.9.21.7 VO视频输出返回码](#)

1.9.12.19 hi_mpi_vo_handle_to_phys_addr

函数功能

根据内存池句柄。预留接口，暂不支持。

函数原型

[hi_u64](#) hi_mpi_vo_handle_to_phys_addr([hi_s32](#) handle)

参数说明

参数名	输入/输出	说明
handle	输入	内存池句柄

返回值说明

- 0：失败，地址为NULL
- 非0：成功，申请的内存地址。

1.9.12.20 hi_mpi_vo_destroy_pool

函数功能

销毁VO显示内存池。预留接口，暂不支持。

函数原型

[hi_s32](#) hi_mpi_vo_destroy_pool([hi_s32](#) handle)

参数说明

参数名	输入/输出	说明
handle	输入	内存池句柄

返回值说明

- 0：成功
- 非0：失败，参见[1.9.21.7 VO视频输出返回码](#)

1.9.12.21 hi_mpi_vo_bind_layer

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

绑定视频层或图形层到某个设备，当前版本不支持绑定视频层。

约束说明

- 对图形层已经存在的绑定关系，不支持重复绑定。需要先解除绑定，再重新绑定。
- 不能在图形层使能的情况下，调用该接口设置绑定关系。
- 图形层仅支持G2、G3、G4调用本接口，G0、G1为固定绑定关系，不可修改绑定。
- 图形层G2、G3支持绑定到DHD0或DHD1，G4支持绑定到DHD1或不绑定，G3与G4不支持同时绑定到DHD1。
- 该接口暂不支持多进程。

函数原型

[hi_s32](#) [hi_mpi_vo_bind_layer](#)([hi_vo_layer](#) layer, [hi_vo_dev](#) dev)

参数说明

参数名	输入/输出	说明
layer	输入	视频/图形层号。 取值范围：[0,9)。
dev	输入	视频输出设备号。 取值范围：[0, 2)。

返回值说明

- 0：成功
- 非0：失败，参见[1.9.21.7 VO视频输出返回码](#)

1.9.12.22 hi_mpi_vo_unbind_layer

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

从某个设备解绑定视频层或图形层。

约束说明

- 将视频层或图形层从设备解除绑定前，必须禁用视频层或图形层。
- 重复解绑定返回成功。
- 不强制要求传入的形参dev必须是之前layer层绑定的设备号用户调用此接口就视为解除layer层的绑定关系。
- 该接口暂不支持多进程。

函数原型

[hi_s32](#) hi_mpi_vo_unbind_layer([hi_vo_layer](#) layer, [hi_vo_dev](#) dev)

参数说明

参数名	输入/输出	说明
layer	输入	视频/图形层号。 取值范围：[0,9)。
dev	输入	视频输出设备号。 取值范围：[0, 2)。

返回值说明

- 0：成功
- 非0：失败，参见[1.9.21.7 VO视频输出返回码](#)

1.9.13 TDE 图形绘制功能

1.9.13.1 hi_tde_open

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

打开TDE (Two Dimension Engine) 设备。

约束说明

- 在进行TDE的相关接口操作前必须进行打开操作，保证TDE设备处于打开状态。
- 该接口不支持多进程。

函数原型

[hi_s32](#) [hi_tde_open](#)([hi_void](#))

参数说明

无

返回值说明

- 0：成功
- 非0：失败，参见[1.9.21.8 TDE图形绘制返回码](#)

1.9.13.2 hi_tde_close

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

关闭TDE设备。

约束说明

- 调用[hi_tde_open](#)接口打开TDE设备后才可调用本接口。
- 调用[hi_tde_open](#)与[hi_tde_close](#)的次数需要对应。
- 该接口不支持多进程。

函数原型

[hi_s32](#) [hi_tde_close](#)([hi_void](#))

参数说明

无

返回值说明

- 0：成功
- 非0：失败，参见[1.9.21.8 TDE图形绘制返回码](#)

1.9.13.3 hi_tde_begin_job

Atlas 200/300/500 推理产品，不支持该接口。

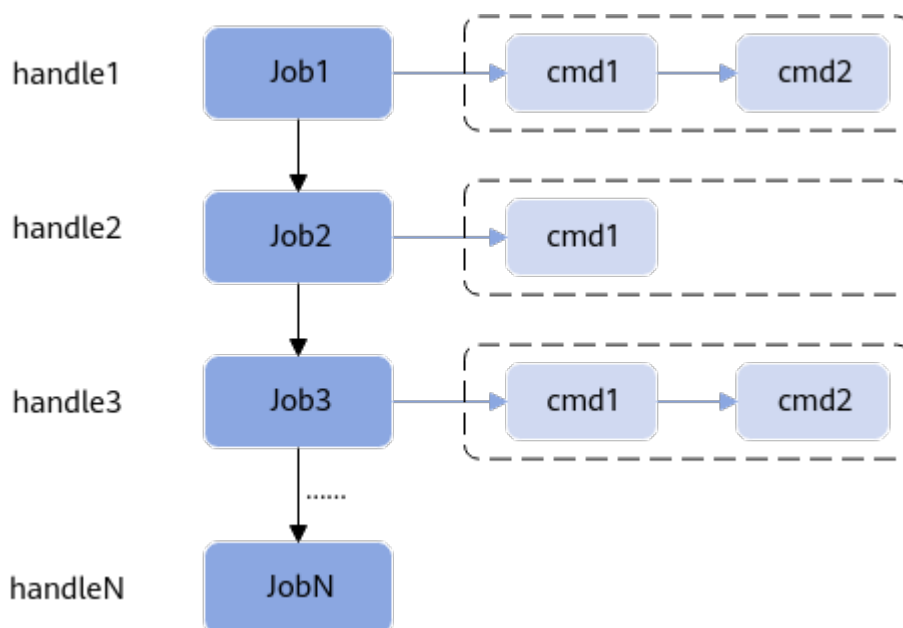
Atlas 训练系列产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

创建TDE任务，获取任务句柄。TDE以任务形式管理TDE命令，一个TDE任务对应一个job，是一个或多个TDE命令的集合；一个TDE任务内的命令是按顺序执行的，如下图所示。成功创建TDE任务、添加TDE命令后，通过[hi_tde_end_job](#)提交任务。



约束说明

- 调用[hi_tde_open](#)接口打开TDE设备后才可调用本接口。
- TDE能够同时执行的任务数、命令数最大均不超过200。
- [hi_tde_begin_job](#)需要和[hi_tde_end_job](#)配对使用。
- 该接口不支持多进程。

函数原型

[hi_s32](#) hi_tde_begin_job([hi_void](#))

参数说明

无

返回值说明

- 正数：获取的任务句柄
- 负数：失败，参见[1.9.21.8 TDE图形绘制返回码](#)

1.9.13.4 hi_tde_end_job

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

提交已创建的TDE任务。

约束说明

- 在调用此接口前应保证调用[hi_tde_open](#)打开TDE设备，并且调用[hi_tde_begin_job](#)获得了有效的任务句柄。
- 提交任务后，此任务对应的handle会变为无效，再次提交会出现错误码HI_ERR_TDE_INVALID_HANDLE。
- 该接口不支持多进程。

函数原型

hi_s32 hi_tde_end_job(**hi_s32** handle, **hi_bool** is_sync, **hi_bool** is_block, **hi_u32** time_out);

参数说明

参数名	输入/输出	说明
handle	输入	tde任务句柄。
is_sync	输入	同步标志，预留参数，当前固定设置为HI_FALSE。
is_block	输入	阻塞标志。 - HI_TRUE：阻塞，提交的TDE任务完成或等待指定超时时间后返回。 - HI_FALSE：非阻塞，直接返回，但需要配合hi_tde_wait_for_done或hi_tde_wait_all_done接口使用。
time_out	输入	超时时间，单位:ms。 当is_block参数设置为阻塞时，可通过本参数设置超时时间；当is_block参数设置为非阻塞时，设置本参数无效。

返回值说明

- 0：成功
- 非0：失败，参见[1.9.21.8 TDE图形绘制返回码](#)

1.9.13.5 hi_tde_cancel_job

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

在调用[hi_tde_begin_job](#)创建任务之后、调用hi_tde_end_job提交任务之前，可调用本接口取消对应的TDE任务及任务中的命令。

约束说明

- 在调用此接口前应保证调用[hi_tde_open](#)打开TDE设备，并且调用[hi_tde_begin_job](#)获得了有效的任务句柄。
- 调用hi_tde_end_job接口提交的任务，无法调用本接口取消。
- 取消后的任务不再有效，不能再向该任务添加命令操作，也不能提交该任务。
- 该接口不支持多进程。

函数原型

[hi_s32](#) hi_tde_cancel_job([hi_s32](#) handle);

参数说明

参数名	输入/输出	说明
handle	输入	tde任务句柄。

返回值说明

- 0：成功
- 非0：失败，参见[1.9.21.8 TDE图形绘制返回码](#)

1.9.13.6 hi_tde_wait_for_done

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

调用hi_tde_end_job接口、且使用非阻塞方式提交TDE任务后，调用本接口等待指定TDE任务完成。本接口为阻塞接口，会阻塞到指定TDE任务完成后退出本接口。

约束说明

- 在调用此接口前应保证调用[hi_tde_open](#)打开TDE设备，并且调用[hi_tde_begin_job](#)获得了有效的任务句柄。
- 该接口不支持多进程。

函数原型

```
hi_s32 hi_tde_wait_for_done(hi_s32 handle);
```

参数说明

参数名	输入/输出	说明
handle	输入	tde任务句柄。

返回值说明

- 0：成功
- 非0：失败，参见[1.9.21.8 TDE图形绘制返回码](#)

1.9.13.7 hi_tde_wait_all_done

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

调用hi_tde_end_job接口、且使用非阻塞方式提交TDE任务后，等待当前TDE设备上所有任务完成。本接口为阻塞接口，会阻塞到所有TDE任务完成后退出本接口。

约束说明

- 在调用此接口前应保证调用[hi_tde_open](#)打开TDE设备，并且调用[hi_tde_begin_job](#)获得了有效的任务句柄。
- 该接口不支持多进程。

函数原型

```
hi_s32 hi_tde_wait_all_done(hi_void)
```

参数说明

无

返回值说明

- 0：成功
- 非0：失败，参见[1.9.21.8 TDE图形绘制返回码](#)

1.9.13.8 hi_tde_quick_copy

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

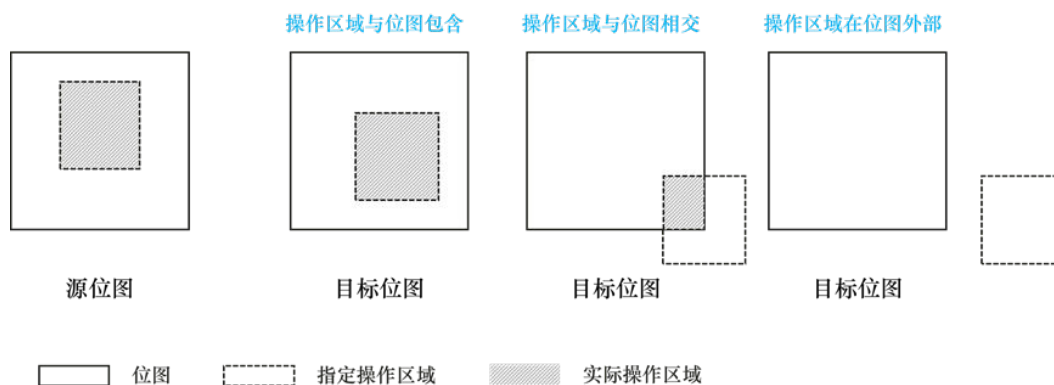
Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

向指定TDE任务中添加快速拷贝命令，实现将源位图的操作区域拷贝到目标位图的操作区域。

约束说明

- 在调用此接口前应保证调用[hi_tde_open](#)打开TDE设备，并且调用[hi_tde_begin_job](#)获得了有效的任务句柄。
- 区域大小长度及宽度支持的区间为[1,4096]。
- 如果目标位图包含目标位图的操作区域，则无需其他操作。如果操作区域与目标位图相交，则裁剪操作区域，有效的操作区域为灰色相交部分。若指定的操作区域与位图不相交，位图及位图操作区域的关系如下图所示。



- 快速拷贝不支持格式转换，源位图和目标位图格式必须一致，当前仅支持ARGB8888格式。
- 快速拷贝不支持缩放功能，如果源和目的操作区域尺寸不一致，则按照两者最小公共区域进行拷贝搬运。
- 指定操作区域要和指定的位图有公共区域，否则会返回错误。
- 该接口不支持多进程。

函数原型

[hi_s32](#) hi_tde_quick_copy([hi_s32](#) handle, const [hi_tde_single_src](#)*single_src)

参数说明

参数名	输入/输出	说明
handle	输入	tde任务句柄。
single_src	输入	单源位图区域信息和目标位图区域信息结构体。

返回值说明

- 0：成功
- 非0：失败，参见[1.9.21.8 TDE图形绘制返回码](#)

1.9.13.9 hi_tde_quick_fill

Atlas 200/300/500 推理产品，不支持该接口。

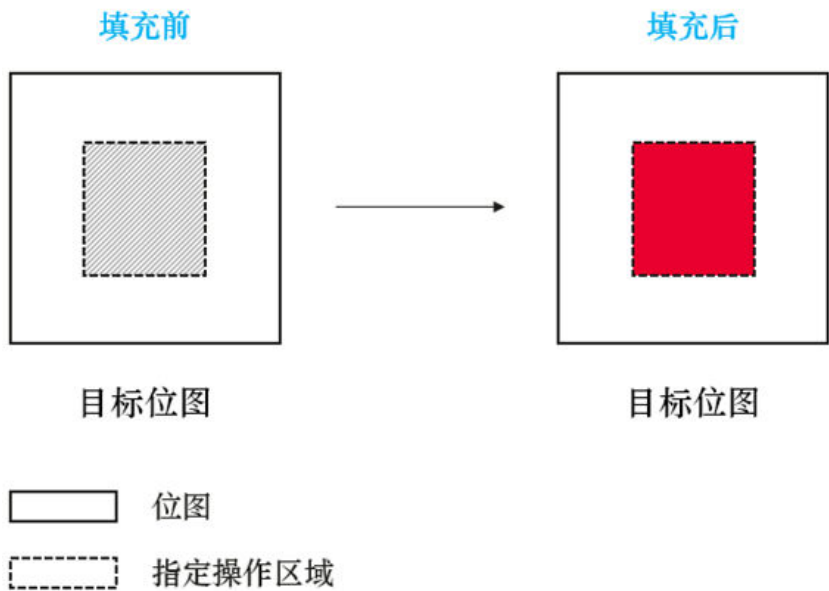
Atlas 训练系列产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

向指定TDE任务中添加快速填充命令，实现将指定的颜色数值纯色填充到目标位图的操作区域中，具体如下图所示。



约束说明

- 在调用此接口前应保证调用[hi_tde_open](#)打开TDE设备，并且调用[hi_tde_begin_job](#)获得了有效的任务句柄。

- 区域大小长度及宽度支持的区间为[1,4096]。
- 指定操作区域要和指定的位图有公共区域，否则会返回错误。
- 当前仅支持ARGB8888格式。
- 该接口不支持多进程。

函数原型

hi_s32 hi_tde_quick_fill(**hi_s32** handle, const **hi_tde_none_src** *none_src, **hi_u32** fill_data)

参数说明

参数名	输入/输出	说明
handle	输入	tde任务句柄。
none_src	输入	无源位图操作区域位图信息结构体。
fill_data	输入	填充颜色。

返回值说明

- 0：成功
- 非0：失败，参见[1.9.21.8 TDE图形绘制返回码](#)

1.9.13.10 hi_tde_pattern_fill

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

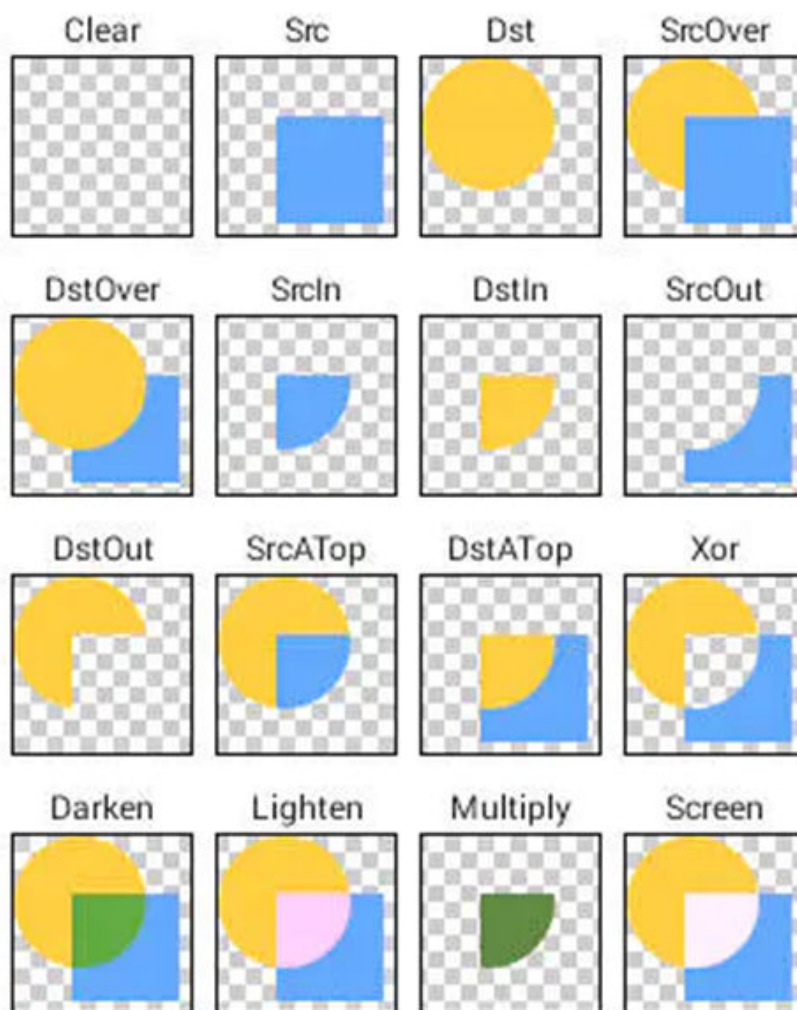
Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

向指定TDE任务中添加模式填充命令，当前仅支持Alpha Blending操作。

模式填充：输入前景位图和背景位图，将前景位图和背景位图的操作区域做附加运算后，平铺在背景位图的操作区域，若前景位图的操作区域大于背景位图的操作区域，则自动进行裁减。

Alpha Blending：通过计算前景和背景重叠后的混合颜色，将前景色和背景色结合获得混合后的新颜色。前景色的透明度不限，如果前景色完全透明，混合后的颜色就是背景色，如果前景色完全不透明，混合后的颜色就是前景色。介于(0, 1)中间的透明度，混合后的颜色，需要通过前景色和背景色的加权公式来计算，混合计算必须使用平直Alpha颜色值，以下是blending的前景、背景组合示例：



约束说明

- 在调用此接口前应保证调用[hi_tde_open](#)打开TDE设备，并且调用[hi_tde_begin_job](#)获得了有效的任务句柄。
- 区域大小长度及宽度支持的区间为[1, 4096]；前景位图宽度支持区间为[1, 256]，高度为[1, 4096]。
- 若前景位图操作区域大于目标位图操作区域，则会自动进行裁剪；若前景位图操作区域大于背景位图操作区域，则自动裁减。
- 模式填充背景位图操作区域长度和宽度要和目标位图操作区域长度和宽度保持一致。
- 该操作不能做缩放。
- 操作区域要和指定的位图有公共区域，否则会返回错误。
- 当前仅支持ARGB8888格式。
- 该接口不支持多进程。

函数原型

```
hi_s32 hi_tde_pattern_fill(hi_s32 handle, const hi_tde_double_src *double_src,
const hi_tde_pattern_fill_opt *fill_opt)
```

参数说明

参数名	输入/输出	说明
handle	输入	tde任务句柄。
double_src	输入	双源位图区域信息和目标位图区域信息结构体。
fill_opt	输入	附加操作，当前只支持Alpha Bending。

返回值说明

- 0：成功
- 非0：失败，参见1.9.21.8 TDE图形绘制返回码

1.9.14 HIFB 叠加图形层管理功能

1.9.14.1 功能及约束说明

HIFB（Hisilicon FrameBuffer）支持的图形层如下表所示，其中G0、G1为超高清图形层，G2图形层为鼠标层，G3、G4图形层为标清图形层。

表 1-57 各图层基本信息说明

图形层	类型	HIFB名称 (fb设备)	默认绑定VO设备	支持颜色格式	分辨率	显存
G0	超高清图形层	fb0	0	• ARGB8888 • ARGB1555 • ARGB4444	• 最小：32*32 • 最大（默认）：1920*1080	8100KB
G1	超高清图形层	fb1	1	• ARGB8888 • ARGB1555 • ARGB4444	• 最小：32*32 • 最大（默认）：1920*1080	8100KB

图形层	类型	HIFB名称 (fb设备)	默认绑定VO设备	支持颜色格式	分辨率	显存
G2	鼠标层	fb2	0	• ARGB8888 • ARGB1555 • ARGB4444	• 最小：8*8 • 最大（默认）：256*256	256KB
G3	标清图形层	fb3	0	• ARGB8888 • ARGB1555 • ARGB4444 • CLUT4 • CLUT2	• 最小：8*8 • 最大（默认）：720*576	1620KB
G4	标清图形层	fb4	1	• ARGB8888 • ARGB1555 • ARGB4444 • CLUT4 • CLUT2	• 最小：8*8 • 最大（默认）：720*576	1620KB

1.9.14.2 FBIOGET_VSCREENINFO

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

获取Framebuffer屏幕的可变信息，包括分辨率、像素格式等信息。

约束说明

- 高清设备的图形层默认分辨率为1920*1080，鼠标层的默认分辨率为256*256，标清设备的图形层的默认分辨率为720*576，像素格式默认为ARGB8888。

- 使用HIFB接口需先调用[hi_mpi_vo_enable](#)启用VO设备。
- 该接口不支持多进程，多线程。

函数原型

```
int ioctl (int fd, FBIOGET_VSCREENINFO, fb_var_screeninfo *var)
```

参数说明

参数名	输入/输出	说明
fd	输入	Framebuffer设备文件描述符。 应用程序中调用open("/dev/fbx"), 会返回当前VO设备对应的Framebuffer设备文件描述符。
FBIOGET_VSCREENINFO	输入	ioctl号。
var	输出	可变信息结构体指针。

返回值说明

- 0：成功
- 非0：失败，返回Linux内核原生错误码，请单击[Link](#)查看详细说明

1.9.14.3 FBIOPUT_VSCREENINFO

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

功能描述

设置Framebuffer屏幕的可变信息，包括分辨率、像素格式等信息。

约束说明

- 可见屏幕宽度（xres）与宽度方向的偏移（xoffset）的和不能超过虚拟分辨率宽度（xres_virtual），yres同理。
- 虚拟分辨率的设置需要在当前图层的显存约束范围内（各个图层的最大/最小分辨率及显存限制如[表1-57](#)所示）。虚拟分辨率所需显存的计算公式如下：

$$M_G = xres\_virtual \times yres\_virtual \times (bits\_per\_pixel / 8)$$

- 必须保证实际分辨率与偏移的和在虚拟分辨率范围内，否则系统会自动调整实际分辨率的大小让其在虚拟分辨率范围内。

- 使用HIFB接口需先调用[hi_mpi_vo_enable](#)启用VO设备。
- 该接口不支持多进程，多线程。

函数原型

`int ioctl (int fd, FBIOPUT_VSCREENINFO, fb\_var\_screeninfo *var)`

参数说明

参数名	输入/输出	说明
fd	输入	Framebuffer设备文件描述符。 应用程序中调用open("/dev/fbx")，会返回当前VO设备对应的Framebuffer设备文件描述符。
FBIOPUT_VSCREENINFO	输入	ioctl号。
var	输入	可变信息结构体指针。

返回值说明

- 0：成功
- 非0：失败

1.9.14.4 FBIOPUT_VSCREENINFO

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

功能描述

获取Framebuffer的固定信息，包括显存用户态地址、显存大小等信息。

约束说明

- 使用HIFB接口需先调用[hi_mpi_vo_enable](#)启用VO设备。
- 显存通过内核mmap函数获取用户态虚拟地址；注意该用户态虚拟地址无法被加速器访问，若需配置给加速器的用户态地址，则需使用[fb_fix_screeninfo](#)中mmio_start参数。
- 无需显示图形层时，用户需调用内核munmap接口释放用户态虚拟地址，否则会导致该图形层不可用，且驱动无法退出。
- 该接口不支持多进程、多线程。

函数原型

```
int ioctl (int fd, FBIOGET_FSCREENINFO, fb_fix_screeninfo *fix)
```

参数说明

参数名	输入/输出	说明
fd	输入	Framebuffer设备文件描述符。 应用程序中调用open("/dev/fbx"), 会返回当前VO设备对应的Framebuffer设备文件描述符。
FBIOGET_FSCREENINFO	输入	ioctl号。
fix	输出	固定信息结构体指针。

返回值说明

- 0: 成功
- 非0: 失败

1.9.14.5 FBIOPAN_DISPLAY

Atlas 200/300/500 推理产品，不支持该接口。

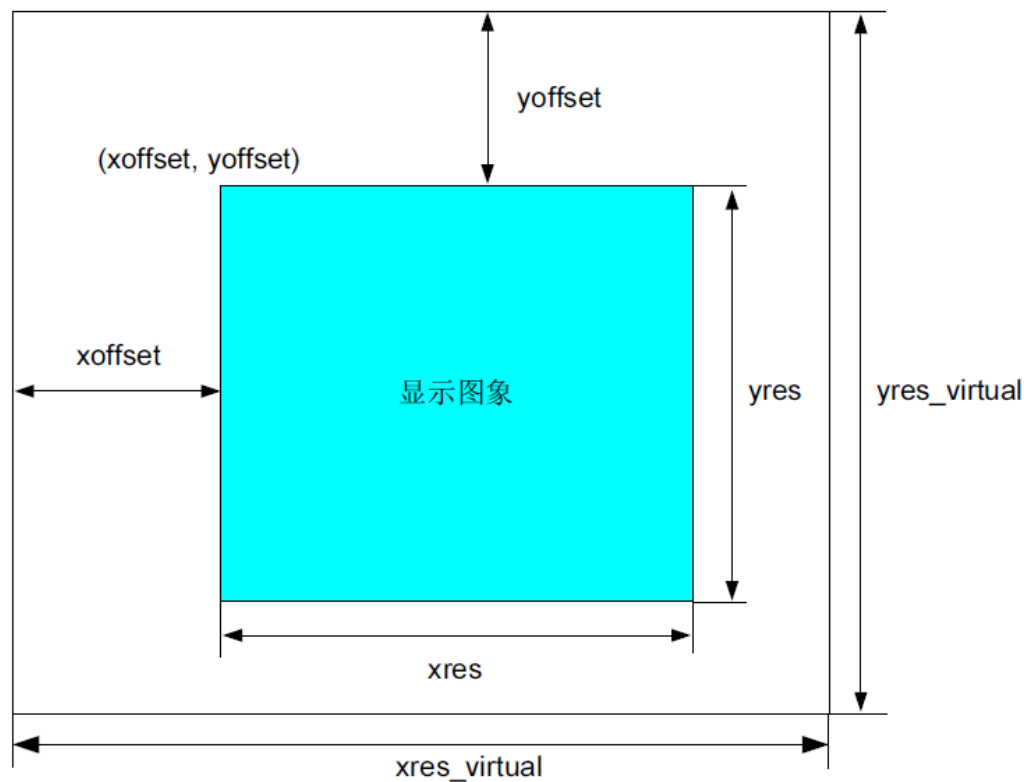
Atlas 训练系列产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

功能描述

设置从虚拟分辨率中的不同偏移处开始显示，实际分辨率不变，如下图所示，（ xres_virtual, yres_virtual ）是虚拟分辨率，（ xres, yres ）是实际显示的分辨率，(xoffset, yoffset)是显示的偏移。



约束说明

- 使用HIFB接口需先调用`hi_mpi_vo_enable`启用VO设备。
- 必须保证实际分辨率与偏移的和在虚拟分辨率范围内，否则设置不成功。
- 该接口不支持多进程、多线程。

函数原型

```
int ioctl (int fd, FBIOPAN_DISPLAY, fb_var_screeninfo *var)
```

参数说明

参数名	输入/输出	说明
fd	输入	Framebuffer设备文件描述符。 应用程序中调用 <code>open("/dev/fbx")</code> ，会返回当前VO设备对应的Framebuffer设备文件描述符。
FBIOPAN_DISPLAY	输入	ioctl号。
var	输入	可变信息结构体指针。

返回值说明

- 0：成功
- 非0：失败

1.9.14.6 FBIOGET_CMAP

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

功能描述

获取颜色表信息。

约束说明

- 使用HIFB接口需先调用[hi_mpi_vo_enable](#)启用VO设备。
- 只有G3和G4图形层才支持颜色表格式，支持格式为CLUT4/CLUT2。
- 该接口不支持多进程、多线程。

函数原型

int ioctl (int fd, FBIOGET_CMAP, [fb_cmap](#) *cmap)

参数说明

参数名	输入/输出	说明
fd	输入	Framebuffer设备文件描述符。 应用程序中调用open("/dev/fbx")，会返回当前VO设备对应的Framebuffer设备文件描述符。
FBIOGET_CMAP	输入	ioctl号。
cmap	输出	颜色表结构体指针。

返回值说明

- 0：成功
- 非0：失败

1.9.14.7 FBIOPUT_CMAP

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

功能描述

设置颜色表信息。

约束说明

- 使用HIFB接口需先调用`hi_mpi_vo_enable`启用VO设备。
- 只有G3和G4图形层才支持颜色表格式，支持格式为CLUT4/CLUT2。
- 该接口不支持多进程、多线程。

函数原型

```
int ioctl (int fd, FBIOPUT_CMAP, fb_cmap *cmap)
```

参数说明

参数名	输入/输出	说明
fd	输入	Framebuffer设备文件描述符。 应用程序中调用 <code>open("/dev/fbx")</code> ，会返回当前VO设备对应的Framebuffer设备文件描述符。
FBIOPUT_CMAP	输入	ioctl号。
cmap	输入	颜色表结构体指针。

返回值说明

- 0：成功
- 非0：失败

1.9.14.8 FBIOGET_SCREEN_ORIGIN_HIFB

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

功能描述

获取叠加层在屏幕上显示的起始点坐标。

约束说明

- 使用HIFB接口需先调用[hi_mpi_vo_enable](#)启用VO设备。
- 该接口不支持多进程、多线程。

函数原型

int ioctl (int fd, FBIOGET_SCREEN_ORIGIN_HIFB, [hi_fb_point](#) *point)

参数说明

参数名	输入/输出	说明
fd	输入	Framebuffer设备文件描述符。 应用程序中调用open("/dev/fbx"), 会返回当前VO设备对应的Framebuffer设备文件描述符。
FBIOGET_SCREEN_ORIGIN_HIFB	输入	ioctl号。
point	输出	起始点坐标结构体指针。

返回值说明

- 0: 成功
- 非0: 失败

1.9.14.9 FBIOPUT_SCREEN_ORIGIN_HIFB

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

功能描述

设置叠加层在屏幕上显示的起始点坐标。

约束说明

- 使用HIFB接口需先调用[hi_mpi_vo_enable](#)启用VO设备。
- 起始点坐标必须非负。
- 该接口不支持多进程、多线程。

函数原型

int ioctl (int fd, FBIOPUT_SCREEN_ORIGIN_HIFB, [hi_fb_point](#) *point)

参数说明

参数名	输入/输出	说明
fd	输入	Framebuffer设备文件描述符。 应用程序中调用open("/dev/fbx"), 会返回当前VO设备对应的Framebuffer设备文件描述符。
FBIOPUT_SCREEN_ORIGIN_HIFB	输入	ioctl号。
point	输入	坐标结构体指针。

返回值说明

- 0: 成功
- 非0: 失败

1.9.14.10 FBIOPUT_SCREEN_ORIGIN_HIFB

Atlas 200/300/500 推理产品, 不支持该接口。

Atlas 训练系列产品, 不支持该接口。

Atlas 推理系列产品, 不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品, 不支持该接口。

功能描述

获取当前叠加层的显示状态。

约束说明

- 使用HIFB接口需先调用[hi_mpi_vo_enable](#)启用VO设备。
- 该接口不支持多进程、多线程。

函数原型

int ioctl (int fd, FBIOPUT_SCREEN_ORIGIN_HIFB, [hi_bool](#) *bshow)

参数说明

参数名	输入/输出	说明
fd	输入	Framebuffer设备文件描述符。 应用程序中调用open("/dev/fbx"), 会返回当前VO设备对应的Framebuffer设备文件描述符。

参数名	输入/输出	说明
FBIOGET_SHOW_HIFB	输入	ioctl号。
bshow	输出	显示状态指针。 - HI_TRUE：代表叠加层当前状态为显示。 - HI_FALSE：代表叠加层当前状态为隐藏。

返回值说明

- 0：成功
- 非0：失败

1.9.14.11 FBIOPUT_SHOW_HIFB

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

功能描述

显示或隐藏该叠加层。

约束说明

- 使用HIFB接口需先调用[hi_mpi_vo_enable](#)启用VO设备。
- 该接口不支持多进程、多线程。

函数原型

int ioctl (**int** fd, FBIOPUT_SHOW_HIFB, **hi_bool** *bshow)

参数说明

参数名	输入/输出	说明
fd	输入	Framebuffer设备文件描述符。 应用程序中调用open("/dev/fbx")，会返回当前VO设备对应的Framebuffer设备文件描述符。
FBIOPUT_SHOW_HIFB	输入	ioctl号。

参数名	输入/输出	说明
bshow	输入	显示状态指针。 - HI_TRUE: 代表设置叠加层状态为显示。 - HI_FALSE: 代表设置叠加层状态为隐藏。

返回值说明

- 0: 成功
- 非0: 失败

1.9.14.12 FBIOGET_ALPHA_HIFB

Atlas 200/300/500 推理产品, 不支持该接口。

Atlas 训练系列产品, 不支持该接口。

Atlas 推理系列产品, 不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品, 不支持该接口。

功能描述

获取叠加层的Alpha信息。

约束说明

- 使用HIFB接口需先调用[hi_mpi_vo_enable](#)启用VO设备。
- 该接口不支持多进程、多线程。

函数原型

```
int ioctl (int fd, FBIOGET_ALPHA_HIFB, hi_fb_alpha *alpha)
```

参数说明

参数名	输入/输出	说明
fd	输入	Framebuffer设备文件描述符。 应用程序中调用open("/dev/fbx"), 会返回当前VO设备对应的Framebuffer设备文件描述符。
FBIOGET_ALPHA_HIFB	输入	ioctl号。
alpha	输出	Alpha属性指针。

返回值说明

- 0：成功
- 非0：失败

1.9.14.13 FBIOPUT_ALPHA_HIFB

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

功能描述

设置叠加层的Alpha信息。

约束说明

- 使用HIFB接口需先调用[hi_mpi_vo_enable](#)启用VO设备。
- 该接口不支持多进程、多线程。

函数原型

`int ioctl (int fd, FBIOPUT_ALPHA_HIFB, hi\_fb\_alpha *alpha)`

参数说明

参数名	输入/输出	说明
fd	输入	Framebuffer设备文件描述符。 应用程序中调用open("/dev/fbx")，会返回当前VO设备对应的Framebuffer设备文件描述符。
FBIOPUT_ALPHA_HIFB	输入	ioctl号。
alpha	输出	Alpha属性指针。

返回值说明

- 0：成功
- 非0：失败

1.9.15 HDMI 外设

1.9.15.1 hi_mpi_hdmi_init

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

初始化HDMI。

约束说明

- 本接口需要与[hi_mpi_hdmi_deinit](#)接口成对调用，即本接口被调用后，需要调用[hi_mpi_hdmi_deinit](#)接口，才允许再次调用[hi_mpi_hdmi_init](#)接口。
- 本接口暂不支持多进程。

函数原型

[hi_s32](#) hi_mpi_hdmi_init(void)

参数说明

无

返回值说明

- 0：成功
- 非0：失败，参见[1.9.21.9 HDMI外设返回码](#)

参考资源

接口调用流程，参见NVR场景视频解码、处理和显示。

1.9.15.2 hi_mpi_hdmi_deinit

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

去初始化HDMI。

约束说明

- 本接口需要与[hi_mpi_hdmi_init](#)接口成对调用，即[hi_mpi_hdmi_init](#)函数被调用后，才允许调用[hi_mpi_hdmi_deinit](#)接口，且保证二者调用次数相同。
- 本接口暂不支持多进程。

函数原型

[hi_s32](#) hi_mpi_hdmi_deinit(void)

参数说明

无

返回值说明

- 0：成功
- 非0：失败，参见[1.9.21.9 HDMI外设返回码](#)

参考资源

接口调用流程，参见NVR场景视频解码、处理和显示。

1.9.15.3 hi_mpi_hdmi_open

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

打开指定的HDMI接口。

约束说明

- [hi_mpi_hdmi_open](#)之前必须先调用[hi_mpi_hdmi_init](#)。
- 重复打开HDMI返回成功。
- 本接口暂时不支持多进程场景。
- 本接口需要与[hi_mpi_hdmi_close](#)接口成对调用，即[hi_mpi_hdmi_open](#)函数被调用后，退出时需要调用[hi_mpi_hdmi_close](#)接口，且需要保持两个接口入参中的[hi_hdmi_id](#)参数值一致。

函数原型

[hi_s32](#) hi_mpi_hdmi_open([hi_hdmi_id](#) hdmi)

参数说明

参数名	输入/输出	说明
hdmi	输入	HDMI接口号。 取值范围：[0, 2)。

返回值说明

- 0：成功
- 非0：失败，参见[1.9.21.9 HDMI外设返回码](#)

参考资源

接口调用流程，参见NVR场景视频解码、处理和显示。

1.9.15.4 hi_mpi_hdmi_close

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

关闭指定的HDMI接口。

约束说明

- 调用[hi_mpi_hdmi_close](#)接口之前必须先调用[hi_mpi_hdmi_open](#)接口，且需要保持两个接口入参中的[hi_hdmi_id](#)参数值一致。
- 本接口暂时不支持多进程场景。
- 本接口需要与[hi_mpi_hdmi_open](#)接口成对调用，即[hi_mpi_hdmi_open](#)函数被调用后，才允许调用[hi_mpi_hdmi_close](#)接口，且保证二者调用次数相同。

函数原型

[hi_s32](#) hi_mpi_hdmi_close([hi_hdmi_id](#) hdmi);

参数说明

参数名	输入/输出	说明
hdmi	输入	HDMI接口号。 取值范围：[0, 2)。

返回值说明

- 0：成功
- 非0：失败，参见[1.9.21.9 HDMI外设返回码](#)

参考资源

接口调用流程，参见NVR场景视频解码、处理和显示。

1.9.15.5 hi_mpi_hdmi_set_attr

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

设置HDMI属性。

约束说明

- 在调用[hi_mpi_hdmi_open](#)接口打开HDMI之后、调用[hi_mpi_hdmi_start](#)接口启用HDMI之前，调用本接口设置HDMI属性，但需保证[hi_mpi_hdmi_open](#)接口与本接口中的[hi_hdmi_id](#)参数值保持一致。
若已调用[hi_mpi_hdmi_start](#)启动HDMI，则应先调用[hi_mpi_hdmi_stop](#)停止HDMI，设置属性后再重新启动。不遵循该流程使用的行为是未定义的，暂不支持。
- 若只设置部分属性，设置前应先调用[hi_mpi_hdmi_get_attr](#)接口获取属性，赋值该部分属性后再设置。
- 本接口暂时不支持多进程场景。

函数原型

[hi_s32](#) hi_mpi_hdmi_set_attr([hi_hdmi_id](#) hdmi, const [hi_hdmi_attr](#) *attr)

参数说明

参数名	输入/输出	说明
hdmi	输入	HDMI接口号。 取值范围：[0, 2)。
attr	输入	HDMI属性结构体指针。部分属性暂时不支持，见数据类型 hi_hdmi_attr 说明。

返回值说明

- 0：成功
- 非0：失败，参见[1.9.21.9 HDMI外设返回码](#)

参考资源

接口调用流程，参见NVR场景视频解码、处理和显示。

1.9.15.6 hi_mpi_hdmi_get_attr

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

获取HDMI属性。

约束说明

- 在调用[hi_mpi_hdmi_open](#)接口打开HDMI之后，调用本接口获取HDMI属性，但需保证[hi_mpi_hdmi_open](#)接口与本接口中的[hi_hdmi_id](#)参数值保持一致。
- 本接口暂时不支持多进程场景。

函数原型

[hi_s32](#) hi_mpi_hdmi_get_attr([hi_hdmi_id](#) hdmi, [hi_hdmi_attr](#) *attr)

参数说明

参数名	输入/输出	说明
hdmi	输入	HDMI接口号。 取值范围：[0, 2)。
attr	输入	HDMI属性结构体指针。部分属性暂时不支持，见数据类型 hi_hdmi_attr 说明。

返回值说明

- 0：成功
- 非0：失败，参见[1.9.21.9 HDMI外设返回码](#)

1.9.15.7 hi_mpi_hdmi_start

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

启动指定的HDMI，使能其送显能力。

约束说明

- [hi_mpi_hdmi_start](#)接口被调用前必须先调用[hi_mpi_hdmi_open](#)接口来打开对应的HDMI，且需要保持两个接口入参中的[hi_hdmi_id](#)参数值一致
- 本接口暂时不支持多进程场景。

函数原型

[hi_s32](#) hi_mpi_hdmi_start([hi_hdmi_id](#) hdmi)

参数说明

参数名	输入/输出	说明
hdmi	输入	HDMI接口号。 取值范围：[0, 2)。

返回值说明

- 0：成功
- 非0：失败，参见[1.9.21.9 HDMI外设返回码](#)

参考资源

接口调用流程，参见NVR场景视频解码、处理和显示。

1.9.15.8 hi_mpi_hdmi_stop

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

停止指定的HDMI接口，不使其送显能力。

约束说明

- [hi_mpi_hdmi_stop](#)接口被调用前必须先调用[hi_mpi_hdmi_open](#)接口来打开对应的HDMI，且需要保持两个接口入参中的[hi_hdmi_id](#)参数值一致
- 本接口暂时不支持多进程场景。

函数原型

[hi_s32](#) hi_mpi_hdmi_stop([hi_hdmi_id](#) hdmi)

参数说明

参数名	输入/输出	说明
hdmi	输入	HDMI接口号。 取值范围：[0, 2)。

返回值说明

- 0：成功
- 非0：失败，参见[1.9.21.9 HDMI外设返回码](#)

参考资源

接口调用流程，参见NVR场景视频解码、处理和显示。

1.9.15.9 hi_mpi_hdmi_set_infotrame

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

设置信息帧。

约束说明

- 调用本接口前，必须先打开HDMI（调用[hi_mpi_hdmi_open](#)），再调用[hi_mpi_hdmi_set_attr](#)接口设置基本属性，最后调用本接口补充设置输出颜色空间、宽高比、时序属性（若无需指定上述参数，使用默认值，可不调用本接口），且保持三个接口入参中的[hi_hdmi_id](#)值一致。
- 本接口目前只支持HI_INFOFRAME_TYPE_AUDIO和HI_INFOFRAME_TYPE_AVI信息帧。
- HI_INFOFRAME_TYPE_AVI信息帧中只支持修改[hi_hdmi_infotrame](#)结构体的timing_mode参数（必须与VO输入的具体时序保持一致）、color_space参数（用户送图时需要设置为YUV444）、aspect_ratio参数（仅支持设置为16比9、4比3）为非默认值，
- HI_INFOFRAME_TYPE_AUDIO信息帧中均需要设置为默认参数，暂不支持修改，详细赋值要求请见[hi_hdmi_audio_infotrame](#)数据类型描述。
- 本接口的部分属性不支持设置，详情请见[hi_hdmi_infotrame](#)结构体及其成员说明。
- 调用该接口设置信息帧相关属性后，会生效成attr属性。
- 本接口属于高级接口，一般不需要调用。若用户使用，则应根据已设置的音视频相关属性（如enVideoFmt），及遵从《High-Definition Multimedia Interface

Specification Version 1.4b 》、《 High-Definition Multimedia Interface Specification Version 2.0 》、《 CEA-861-D 》与《 CEA-861-F 》标准基础上设置信息帧，不依据音视频属性遵从标准发送信息帧的行为是未定义的，暂不支持。

- 本接口暂时不支持多进程场景。

函数原型

hi_s32 hi_mpi_hdmi_set_infoframe(**hi_hdmi_id** hdmi, const **hi_hdmi_infoframe** *infoframe)

参数说明

参数名	输入/输出	说明
hdmi	输入	HDMI接口号。 取值范围：[0, 2)。
infoframe	输入	HDMI信息帧结构体指针。

返回值说明

- 0：成功
- 非0：失败，参见[1.9.21.9 HDMI外设返回码](#)

1.9.15.10 hi_mpi_hdmi_get_sink_capability

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

获取指定端口的能力集。

约束说明

若在打开HDMI设备（通过[hi_mpi_hdmi_open](#)接口打开）之后调用本接口，可获取设备是否连接的信息；若在打开HDMI设备（通过[hi_mpi_hdmi_open](#)接口打开）且插入线缆之后调用本接口才可以获取Sink端能力集的所有信息。

函数原型

hi_s32 hi_mpi_hdmi_get_sink_capability(**hi_hdmi_id** hdmi, **hi_hdmi_sink_capability** *capability)

参数说明

参数名	输入/输出	说明
hdmi	输入	HDMI接口号。 取值范围：[0, 2)。
capability	输入	HDMI Sink端能力集结构体指针。

返回值说明

- 0：成功
- 非0：失败，参见[1.9.21.9 HDMI外设返回码](#)

1.9.15.11 hi_mpi_hdmi_register_callback

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

用户态注册HDMI事件回调函数。

约束说明

- 调用前必须先调用[hi_mpi_hdmi_open](#)；
- 建议用户注册HDMI事件回调函数；
例如，当产生热插拔事件时，可以通过注册的回调函数读取热插拔后产生的能力集信息为依据更改HDMI的属性，然后重新启动HDMI，使HDMI的属性适配新插入的对端显示器/电视。
若用户不注册事件回调函数，则当事件产生时，HDMI内部会采取默认的处理方式，默认支持原屏幕热插拔；若用户注册事件回调函数，则直接按用户注册的函数处理，系统内部的默认热插拔函数会失效，需由用户自行保证对热插拔的支持。
- 回调函数注册只支持插入和拔出两个事件；
一般在插入事件处理函数中使用[hi_mpi_hdmi_get_sink_capability](#)获取sink端能力集。
- 回调函数注册需要由用户保证入参中的函数指针不存在内存泄漏、函数本身运行挂死等问题；
- 若本接口被调用时，HDMI显示设备已连接，则第一次拔出事件不响应，第二次才开始响应。

函数原型

```
hi_s32 hi_mpi_hdmi_register_callback(hi_hdmi_id hdmi, const
hi_hdmi_callback_func *callback_func)
```

参数说明

参数名	输入/输出	说明
hdmi	输入	HDMI接口号。 取值范围：[0, 2)。
callback_func	输入	HDMI回调函数结构体。

返回值说明

- 0：成功
- 非0：失败，参见[1.9.21.9 HDMI外设返回码](#)

1.9.16 VPC 图像处理功能

1.9.16.1 功能说明

Atlas 200/300/500 推理产品上，不支持本章中的VPC功能。

Atlas 训练系列产品上，不支持本章中的VPC功能。

功能说明

VPC (Vision Preprocessing Core) 各功能的介绍、各版本的功能支持度如下表所示。

功能	描述	Atlas 推理系列产品	Atlas 200/500 A2推理产品	Atlas A2训练系列产品/Atlas 800I A2推理产品
抠图	从输入图片中抠出需要用的图片区域，支持一图多框和多图多框。	√	√	√

功能	描述	Atlas 推理系列产品	Atlas 200/500 A2推理产品	Atlas A2训练系列产品/Atlas 800I A2推理产品
缩放	- 针对不同分辨率的图像，支持8K缩放、非8K缩放。 - 支持单图裁剪缩放（支持非压缩格式）、一图多框裁剪缩放（支持非压缩格式）。 - 其它缩放方式，如：原图缩放、等比例缩放（缩放前后图片的宽高比例相同）。	√	√	√
叠加	从输入图片中抠出来的图，对抠出的图进行缩放后，放在用户输出图片的指定区域，输出图片可以是空白图片（由用户申请的空输出内存产生的），也可以是已有图片（由用户申请输出内存后将已有图片读入输出内存），只有当输出图片是已有图片时，才表示叠加。	√	√	√
拼接	从输入图片中抠多张图片，对抠出的图进行缩放后，放到输出图片的指定区域。	√	√	√
图像金字塔	对原图进行多次高斯滤波之后向下采样产生不同尺寸的图像。	√	x	x
直方图统计	统计图像每个通道（RGB/YUV）的像素值分布。	√	√	√
色彩重映射	根据配置信息将图片从原图映射为另一张图。	√	√	√
边界填充	对图像进行边界填充。	√	√	√
格式转换	支持RGB格式、YUV格式之间的格式转换。	√	√	√
图像灰度化	将彩色图像转化为灰度图像。需注意，输入为灰度图像、输出只能为灰度图像。 实现图像灰度化的操作是输出YUV400格式的输出图片。	√	√	√

功能	描述	Atlas 推理系列产品	Atlas 200/500 A2推理产品	Atlas A2训练系列产品/Atlas 800I A2推理产品
环视拼接	主要用于车载场景生成全景图。 使用该功能时，输入图片为4张，一般为前、左、后、右四个方向的图像，经过畸变矫正、增益补偿、图像融合，最终将4张图片融合成一张全景图片。其中，增益补偿是可选步骤，可在调用参数设置接口时选择是否进行增益补偿，通过增益补偿可平衡不同摄像头光照，达到更好的效果；图像融合当前采用的是加权融合方式，即像素加权平均法。	x	x	x
Remap变换	根据像素位置LUT对输入图像进行几何形变，典型的用途包括：镜头畸变校正、仿射变换、透视变换，功能示意图请参见图1-25。 形变方式可以用下述公式描述： $dst(x,y)=src(LUT(x,y))$ 。 其中， $dst(x,y)$ 为输出图像在坐标 (x,y) 处的像素值，像素位置 $LUT(x,y)$ 是输出图像在 (x,y) 处的像素对应输入图像中的横、纵坐标值， $src(LUT(x,y))$ 为输入图像在坐标 $LUT(x,y)$ 处的像素值。通用的像素位置LUT可通过用户提供的map1（横坐标映射矩阵）和map2（纵坐标映射矩阵）生成，仿射变换或透视变换的像素位置LUT也可通过用户提供的3个点或4个点对信息计算生成。	x	√	√
滤波	对输入图片做滤波处理，当前可支持中值滤波/腐蚀/膨胀/高斯滤波/均值滤波/卷积滤波。	x	√	√
旋转	对输入图片做固定角度的旋转，支持90度/180度/270度。	x	√	√
马赛克	对输入图片做马赛克处理。	x	√	x
覆盖	对输入图片做局部覆盖操作。	x	√	x

功能	描述	Atlas 推理系列产品	Atlas 200/500 A2推理产品	Atlas A2训练系列产品/Atlas 800I A2推理产品
画线	对输入图片做画线处理。	x	√	x
添加水印	对输入图片做添加水印的处理。	x	√	x

功能示意图(Atlas 推理系列产品)

图 1-17 VPC 功能示意图（抠图+缩放+叠加）

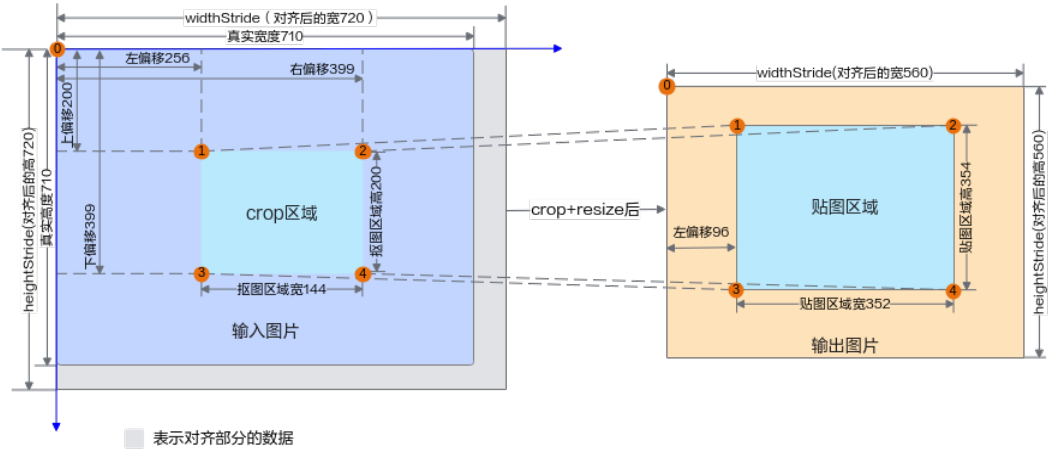


图 1-18 VPC 功能示意图（拼接）

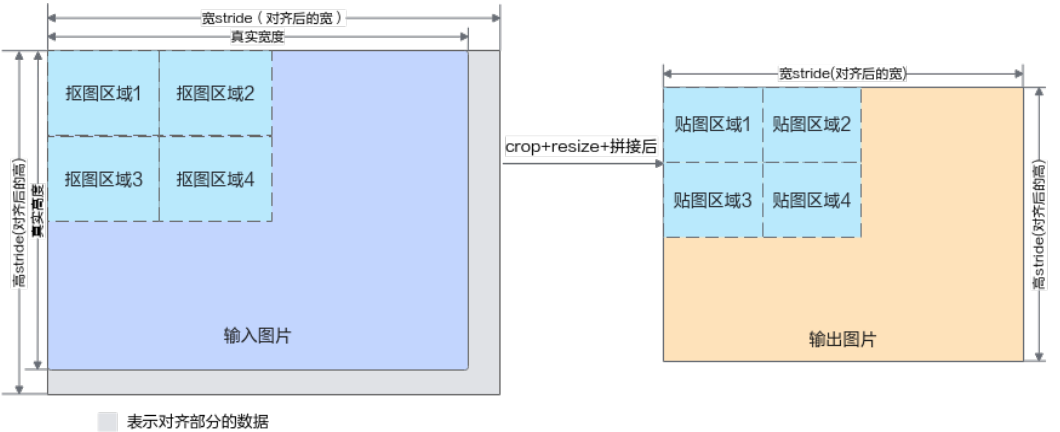


图 1-19 等比例缩放（贴图区域在输出图片的中心位置），即缩放前后图片的宽高比例相同

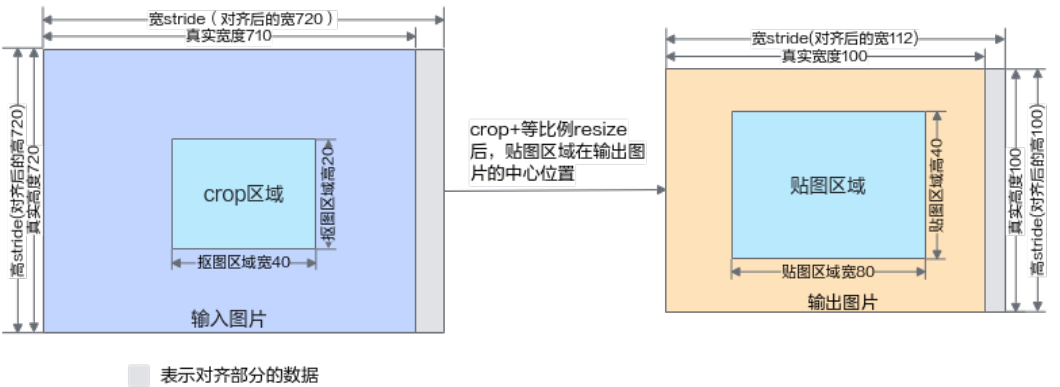


图 1-20 等比例缩放（贴图区域在输出图片的上下居中位置）

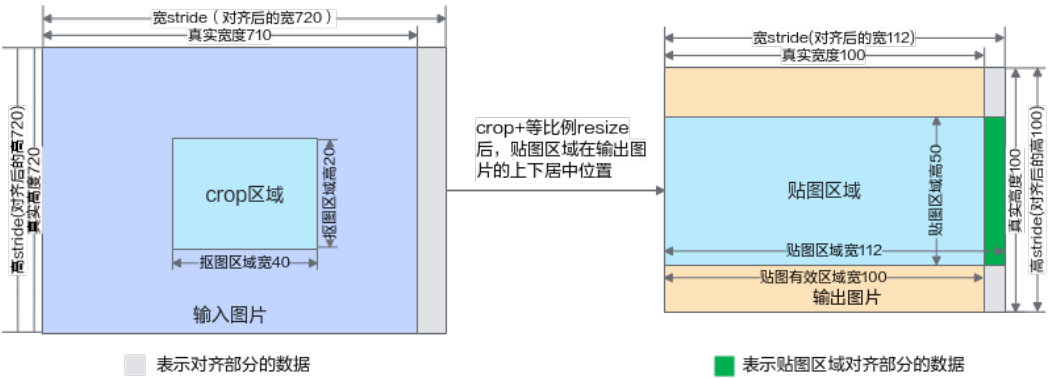
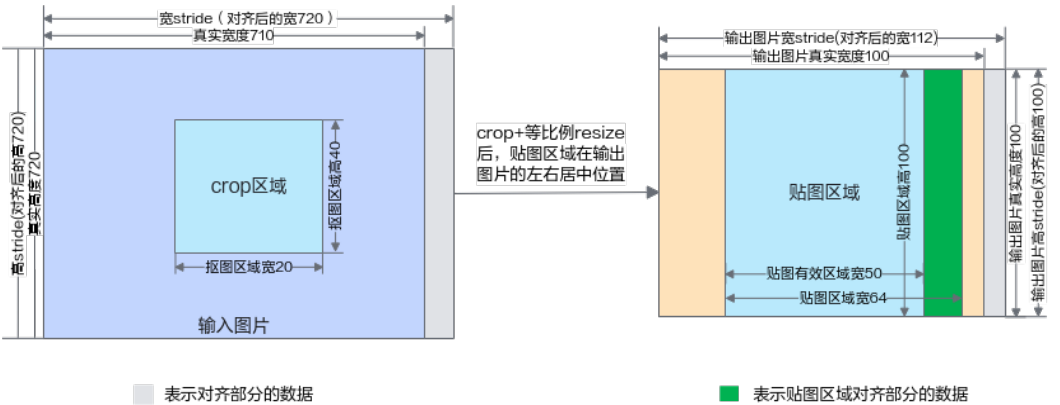


图 1-21 等比例缩放（贴图区域在输出图片的左右居中位置）



功能示意图(Atlas A2 训练系列产品/Atlas 800I A2 推理产品)(Atlas 200/500 A2 推理产品)

图 1-22 VPC 功能示意图（抠图+缩放+叠加）

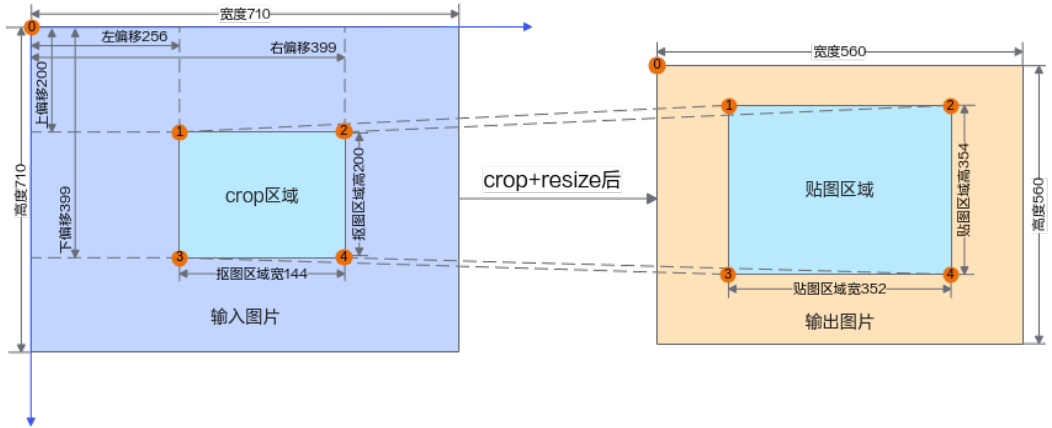


图 1-23 VPC 功能示意图（拼接）

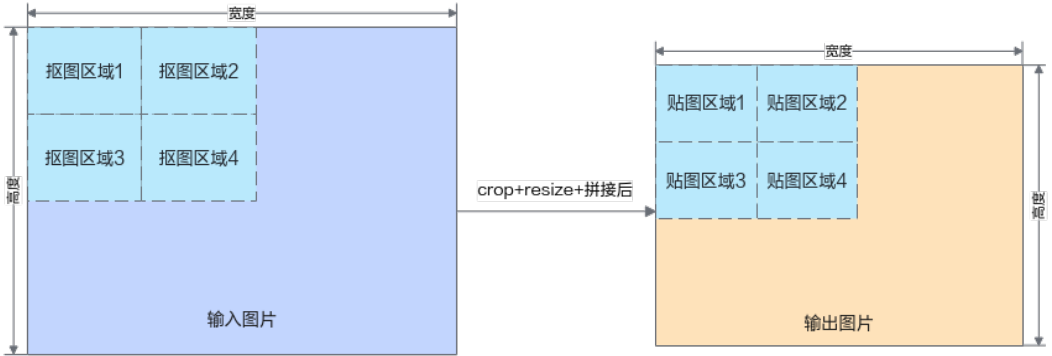


图 1-24 等比例缩放（贴图区域在输出图片的中心位置），即缩放前后图片的宽高比例相同

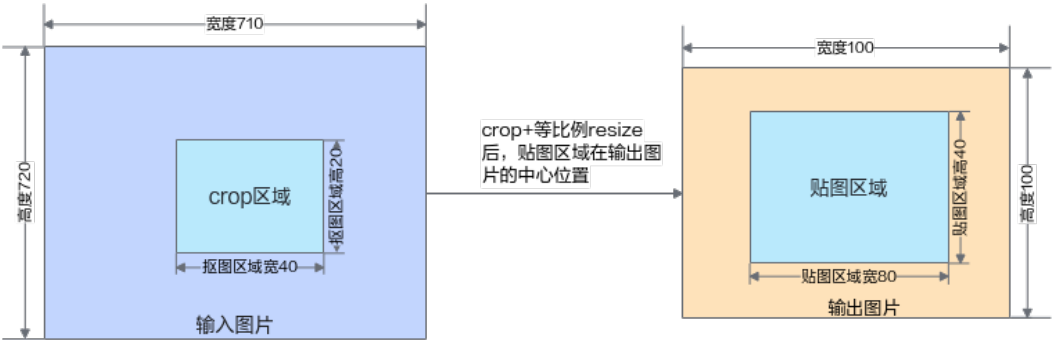
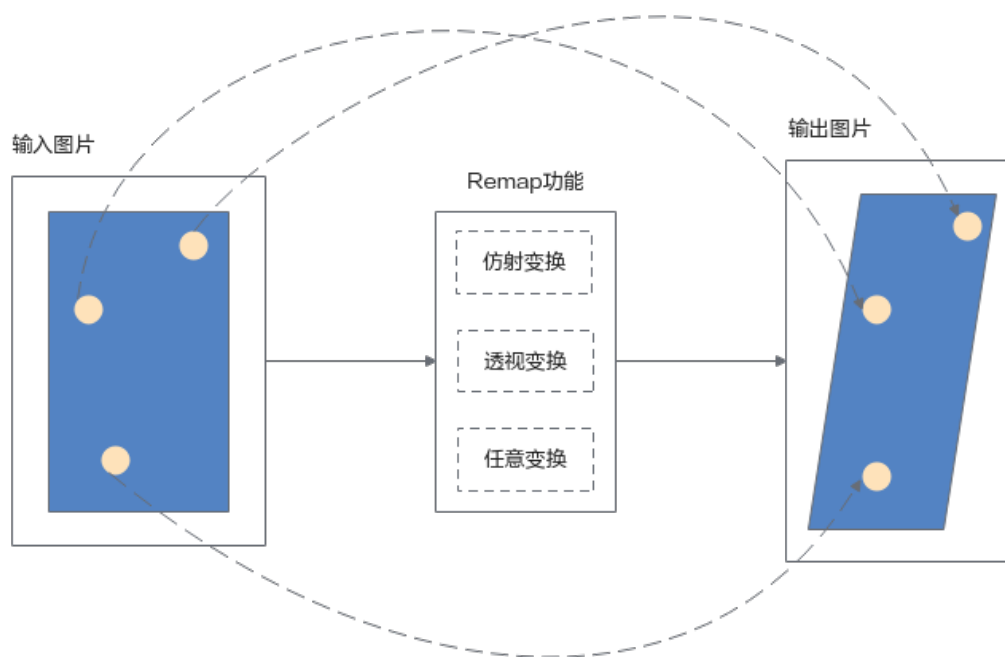


图 1-25 Remap 功能



参考说明

RGB、YUV格式图像的各分量排布示意图。示例：SP图像以YUV420SP为例，Packed和RGB图像以ARGB图像为例。

格式	分辨率	宽stride	高stride	buffer大小					
yuv420sp	4*4	4	4	24					
内存排列									
y11	y12	y13	y14						
y21	y22	y23	y24						
y31	y32	y33	y34						
y41	y42	y43	y44						
u11	v11	u13	v13						
u31	v31	u33	v33						
格式	分辨率	宽stride	高stride	buffer大小					
yuv420sp	4*4	6	6	54					
内存排列 x为无效数据									
y11	y12	y13	y14	x	x				
y21	y22	y23	y24	x	x				
y31	y32	y33	y34	x	x				
y41	y42	y43	y44	x	x				
x	x	x	x	x	x				
x	x	x	x	x	x				
u11	v11	u13	v13	x	x				
u31	v31	u33	v33	x	x				
x	x	x	x	x	x				
格式	分辨率	宽stride	高stride	buffer大小					
argb	2*2	10	4	40					
内存排列 x为无效数据									
a11	r11	g11	b11	a12	r12	g12	b12	x	x
a21	r21	g21	b21	a22	r22	g22	b22	x	x
x	x	x	x	x	x	x	x	x	x
x	x	x	x	x	x	x	x	x	x

1.9.16.2 约束说明

Atlas 200/300/500 推理产品上，该功能为预留功能，暂不支持。

Atlas 训练系列产品上，该功能为预留功能，暂不支持。

图片分辨率约束

- 输入图片分辨率
VPC处理图片时，通用的输入分辨率范围为10*6~8192*8192，其中部分接口的分辨率限制可能不同，请以具体接口内的说明为准，请参见[1.9.16 VPC图像处理功能](#)下各接口的说明。其中，当输入图片格式为YUV440SP、YUV440P时，输入图片的宽最大值为4096。
- 输出图片分辨率

版本	分辨率范围
Atlas 推理系列产品	10*6~4096*4096

版本	分辨率范围
Atlas A2训练系列产品/Atlas 800I A2推理产品 Atlas 200/500 A2推理产品	10*6~4096*8192

内存申请/释放接口说明

输入、输出内存大小要求请参见[图片格式](#)、[宽高对齐](#)、[内存约束](#)中的计算公式。

版本	内存申请/释放接口
Atlas 推理系列产品	VPC在处理图片时，需调用 hi_mpi_dvpp_malloc 接口申请Device上的输入、输出内存，调用 hi_mpi_dvpp_free 接口释放输入、输出内存，这部分内存的生命周期由用户自行管理。
Atlas A2训练系列产品/Atlas 800I A2推理产品 Atlas 200/500 A2推理产品	VPC在处理图片时，支持以下两种内存申请接口： - 调用aclrtMalloc\aclrtFree接口申请\释放Device上的输入、输出内存，这部分内存的生命周期由用户自行管理。 - 调用hi_mpi_dvpp_malloc\hi_mpi_dvpp_free接口申请\释放Device上的输入、输出内存，这部分内存的生命周期由用户自行管理。 注意：调用hi_mpi_dvpp_malloc接口申请的内存为媒体数据处理的专用内存，但专用内存的地址空间有限，若关注内存规划或内存资源有限时，建议调用aclrtMalloc接口申请内存。

图片格式、宽高对齐、内存约束

说明

在调用接口实现VPC功能时：

- 图片格式的定义请参见[hi_pixel_format](#)，宽stride、高stride等概念请参见[1.9.2 基本概念](#)。
- 不同版本中，宽stride、高stride的最小值、最大值不同，如下：
 - Atlas 推理系列产品上，宽stride最小32、最大16384（16384=4096*4，宽是4096的argb格式的图像，1个像素占用4个字节，一行像素就占用4096*4，即宽stride）；高stride最小6、最大16384。
 - Atlas 200/500 A2推理产品上，宽stride最小10、高stride最小6。
 - Atlas A2训练系列产品/Atlas 800I A2推理产品上，宽stride最小10、高stride最小6。
- 各接口对图片格式的要求可能不同，请参见[1.9.16 VPC图像处理功能](#)下各接口的说明。

表 1-58 输入图片格式、宽高对齐、内存大小约束

版本	图片格式	图片宽、高 对齐要求	图片宽 stride、高 stride对齐要求 Atlas 推理系 列产品	图片宽 stride、高 stride对齐要求 Atlas 200/500 A2推理产品 Atlas A2训练 系列产品/ Atlas 800I A2 推理产品	内存大小要 求 (单位 Byte)
通用格 式，各版 本都支持	YUV400 8bit	无对齐 要求	宽stride为宽 16对齐后的 值； 高stride无对齐 要求，与高相 同即可。	宽stride无对齐 要求，与宽相 同即可； 高stride无对齐 要求，与高相 同即可。	宽stride * 高stride
	YUV420SP NV12 8bit YUV420SP NV21 8bit	宽2对 齐 高2对 齐	宽stride为宽 16对齐后的 值； 高stride为高2 对齐后的值。	宽stride无对齐 要求，与宽相 同即可； 高stride无对齐 要求，与高相 同即可。	宽stride * 高stride * 3/2
	YUV422SP 8bit YVU422SP 8bit	宽2对 齐 高无对 齐要求	宽stride为宽 16对齐后的 值； 高stride无对齐 要求，与高相 同即可。	宽stride无对齐 要求，与宽相 同即可； 高stride无对齐 要求，与高相 同即可。	宽stride * 高stride * 2
	YUV444SP 8bit YVU444SP 8bit	无对齐 要求	宽stride为宽 16对齐后的 值； 高stride无对齐 要求，与高相 同即可。	宽stride无对齐 要求，与宽相 同即可； 高stride无对齐 要求，与高相 同即可。	宽stride * 高stride * 3

版本	图片格式	图片宽、高对齐要求	图片宽stride、高stride对齐要求 Atlas 推理系列产品	图片宽stride、高stride对齐要求 Atlas 200/500 A2推理产品 Atlas A2训练系列产品/ Atlas 800I A2推理产品	内存大小要求 (单位 Byte)
	YUV422Packed YUYV 8bit YUV422Packed UYVY 8bit YUV422Packed YVYU 8bit YUV422Packed VYUY 8bit	宽2对齐 高无对齐要求	宽stride为宽16对齐后、再乘以2的值； 高stride无对齐要求，与高相同即可。	宽stride为宽*2的值； 高stride无对齐要求，与高相同即可。	宽stride * 高stride
	YUV444Packed 8bit RGB888 BGR888	无对齐要求	宽stride为宽16对齐后、再乘以3的值； 高stride无对齐要求，与高相同即可。	宽stride为宽*3的值； 高stride无对齐要求，与高相同即可。	宽stride * 高stride
	ARGB8888 ABGR8888 RGBA8888 BGRA8888	无对齐要求	宽stride为宽16对齐后、再乘以4的值； 高stride无对齐要求，与高相同即可。	宽stride为宽*4的值； 高stride无对齐要求，与高相同即可。	宽stride * 高stride
	YUV440SP 8bit YVU440SP 8bit	宽无对齐要求，但宽≤4096 高2对齐	宽stride为宽16对齐后的值； 高stride为高2对齐后的值。	宽stride无对齐要求，与宽相同即可； 高stride为高2对齐后的值。	宽stride * 高stride * 2

版本	图片格式	图片宽、高对齐要求	图片宽stride、高stride对齐要求 Atlas 推理系列产品	图片宽stride、高stride对齐要求 Atlas 200/500 A2推理产品 Atlas A2训练系列产品/ Atlas 800I A2推理产品	内存大小要求 (单位 Byte)
该部分格式仅以下版本支持： Atlas 200/500 A2推理产品 Atlas A2训练系列产品/Atlas 800I A2推理产品	YVU420Planar YUV420Planar	宽2对齐，高2对齐	-	宽stride无对齐要求，与宽相同即可；高stride无对齐要求，与高相同即可。	宽stride * 高stride * 3/2
	YVU422Planar YUV422Planar	无对齐要求	-	宽stride无对齐要求，与宽相同即可；高stride无对齐要求，与高相同即可。	宽stride * 高stride * 2
	YVU444Planar YUV444Planar	无对齐要求	-	宽stride无对齐要求，与宽相同即可；高stride无对齐要求，与高相同即可。	宽stride * 高stride * 3
	YVU444Packed 8bit	无对齐要求	-	宽stride无对齐要求，宽stride为宽*3的值；高stride无对齐要求，与高相同即可。	宽stride * 高stride
	YUV440Planar YVU440Planar	宽无对齐要求，但宽≤4096 高2对齐	-	宽stride无对齐要求，与宽相同即可；高stride无对齐要求，与高相同即可。	宽stride * 高stride * 2
	RGB888Planar BGR888Planar	无对齐要求	-	宽stride无对齐要求，与宽相同即可；高stride无对齐要求，与高相同即可。	宽stride * 高stride * 3

版本	图片格式	图片宽、高 对齐要求	图片宽 stride、高 stride对齐要求 Atlas 推理系 列产品	图片宽 stride、高 stride对齐要求 Atlas 200/500 A2推理产品 Atlas A2训练 系列产品/ Atlas 800I A2 推理产品	内存大小要求 (单位 Byte)
该部分格式仅以下版本支持： Atlas A2训练系列产品/Atlas 800I A2推理产品	YUV400 FP32	无对齐要求	-	宽stride无对齐要求，宽stride为宽*4（4表示FP32占4个字节）的值；高stride无对齐要求，与高相同即可。	宽stride * 高stride
	RGB888 FP32 BGR888 FP32	无对齐要求	-	宽stride无对齐要求，宽stride为宽*3*4（4表示FP32占4个字节）的值；高stride无对齐要求，与高相同即可。	宽stride * 高stride
	RGB888Planar FP32 BGR888Planar FP32	无对齐要求	-	宽stride无对齐要求，宽stride为宽*4（4表示FP32占4个字节）的值；高stride无对齐要求，与高相同即可。	宽stride * 高stride * 3

表 1-59 输出图片格式、宽高对齐、内存大小约束

-	图片格式	图片宽、高对齐要求	图片宽stride、高stride对齐要求 Atlas 推理系列产品	图片宽stride、高stride对齐要求 Atlas 200/500 A2推理产品 Atlas A2训练系列产品/ Atlas 800I A2推理产品	内存大小要求（单位Byte）
通用格式，各版本都支持	YUV400 8bit	无对齐要求	宽stride为宽16对齐后的值； 高stride无对齐要求，与高相同即可。	宽stride无对齐要求，与宽相同即可； 高stride无对齐要求，与高相同即可。	宽stride * 高stride
	YUV420SP NV12 8bit YUV420SP NV21 8bit	宽2对齐 高2对齐	宽stride为宽16对齐后的值； 高stride为高2对齐后的值。	宽stride无对齐要求，与宽相同即可； 高stride无对齐要求，与高相同即可。	宽stride * 高stride * 3/2
	YUV422SP 8bit YVU422SP 8bit	宽2对齐 高无对齐要求	宽stride为宽16对齐后的值； 高stride无对齐要求，与高相同即可。	宽stride无对齐要求，与宽相同即可； 高stride无对齐要求，与高相同即可。	宽stride * 高stride * 2
	YUV444Packed 8bit RGB888 BGR888	无对齐要求	宽stride为宽16对齐后、再乘以3的值； 高stride无对齐要求，与高相同即可。	宽stride为宽*3的值； 高stride无对齐要求，与高相同即可。	宽stride * 高stride
	ARGB8888 ABGR8888 RGBA8888 BGRA8888	无对齐要求	宽stride为宽16对齐后、再乘以4的值； 高stride无对齐要求，与高相同即可。	宽stride为宽*4的值； 高stride无对齐要求，与高相同即可。	宽stride * 高stride

-	图片格式	图片宽、高对齐要求	图片宽stride、高stride对齐要求 Atlas 推理系列产品	图片宽stride、高stride对齐要求 Atlas 200/500 A2推理产品 Atlas A2训练系列产品/ Atlas 800I A2推理产品	内存大小要求 (单位 Byte)
该部分格式仅以下版本支持： Atlas 200/500 A2推理产品 Atlas A2训练系列产品/Atlas 800I A2推理产品	YUV422Packed YUYV 8bit YUV422Packed UYVY 8bit YUV422Packed YVYU 8bit YUV422Packed VYUY 8bit	宽无对齐要求 高2对齐	-	宽stride为宽*2的值； 高stride无对齐要求，与高相同即可。	宽stride * 高stride
该部分格式仅以下版本支持： Atlas 200/500 A2推理产品 Atlas A2训练系列产品/Atlas 800I A2推理产品	YVU444Packed	无对齐要求	-	宽stride为宽*3的值； 高stride无对齐要求，与高相同即可。	宽stride * 高stride
	RGB888Planar BGR888Planar	无对齐要求	-	宽stride无对齐要求，与宽相同即可；高stride无对齐要求，与高相同即可。	宽stride * 高stride * 3
该部分格式仅以下版本支持： Atlas A2训练系列产品/Atlas 800I A2推理产品	YUV400 FP32	无对齐要求	-	宽stride无对齐要求，宽stride为宽*4（4表示FP32占4个字节）的值；高stride无对齐要求，与高相同即可。	宽stride * 高stride

-	图片格式	图片宽、高 对齐要求	图片宽 stride、高 stride对齐要求 Atlas 推理系 列产品	图片宽 stride、高 stride对齐要求 Atlas 200/500 A2 推理产品 Atlas A2训练 系列产品/ Atlas 800I A2 推理产品	内存大小要求 (单位 Byte)
	RGB888 FP32 BGR888 FP32	无对齐 要求	-	宽stride无对 齐要求, 宽 stride为宽*3*4 (4表示FP32 占4个字节) 的值; 高 stride无对 齐要求, 与高相 同即可。	宽stride * 高 stride
	RGB888Pla nar FP32 BGR888Pla nar FP32	无对齐 要求	-	宽stride无对 齐要求, 宽 stride为宽*4 (4表示FP32 占4个字节) 的值; 高 stride无对 齐要求, 与高相 同即可。	宽stride * 高 stride * 3

抠图、贴图约束

- 抠图区域不超出输入图片区域。
- 抠图、贴图区域的奇数、偶数限制为：
输出图片格式为YUV420SP，贴图区域奇数、偶数限制为：左偏移和上偏移为偶数、右偏移和下偏移为奇数。
输出图片格式为YUV422SP，贴图区域奇数、偶数限制为：左偏移为偶数、右偏移为奇数。
其它格式，贴图区域没有偏移奇偶数的限制。
- 贴图区域不超出输出图片区域，最大贴图个数256个。
在Atlas 推理系列产品上，由于16对齐的要求，还存在以下约束：
 - 贴图区域相对输出图片的左偏移16对齐。贴图时可直接放置在输出图片的最左侧，即相对输出图片的左偏移为0，0也满足16对齐的要求。
 - 输出图片的贴图宽度建议16对齐，如果不是16对齐，会多写一段无效数据使其16对齐，如图1-20、图1-21所示，贴图区域旁边的绿色框就表示无效数据。

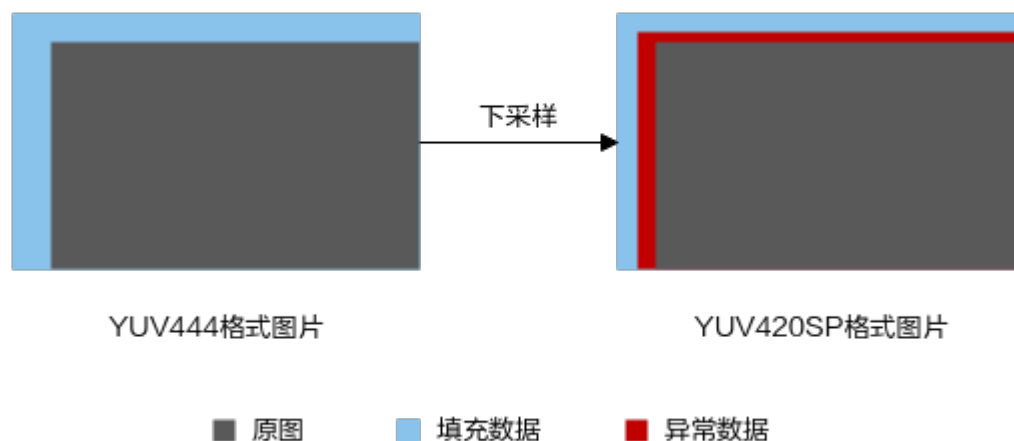
推理场景下，VPC的输出作为推理的输入时，为保证推理精度，贴图有效区域的宽不满足16对齐时，在贴图前用户可先将贴图有效区域的宽缩放成16对齐，或在推理前借助AIPP抠出有效区域，去除无效数据。

- 等比例缩放场景下，由于贴图左偏移16对齐的限制，如果直接调用 [hi_mpi_vpc_crop_resize_paste](#) 接口实现缩放功能，贴图的位置可能不在输出图片的正中心位置，此时可通过抠图、缩放、填充几个功能配合使用（调用 [hi_mpi_vpc_crop_resize_make_border](#) 或 [hi_mpi_vpc_batch_crop_resize_make_border](#) 接口），实现等比例缩放。

YUV 格式图像下采样约束

VPC在处理图片时，会根据输入或输出图片格式，将输入图片格式转换为YUV444或RGB用于内部处理，YUV444或RGB没有宽高奇偶数的限制，但当输出图片格式为YUV420SP或YUV422SP格式时，会进行下采样处理，由于YUV420SP或YUV422SP格式本身的数据排布导致宽高存在奇偶数限制，因此输出图片的边缘可能存在异常数据。

图 1-26 异常效果图片举例



出现异常数据的根因在于，在计算过程中，出现输出图片位置处于奇数起始点时，此时YUV444格式的图片是正确的，但是下采样到YUV420SP格式时，由于奇数行和偶数行是共用同一个uv的，导致图片起始行的Y与上一行的UV组成新的像素，产生异常数据。

1.9.16.3 性能指标说明

Atlas 200/300/500 推理产品上，该功能为预留功能，暂不支持。

Atlas 训练系列产品上，该功能为预留功能，暂不支持。

性能指标说明

本节仅描述各产品对应的昇腾AI处理器的性能指标参考数据，各产品由于其内存、CPU等配置不同，在性能规格上会存在差异，详细产品的性能规格请查阅[昇腾社区](#)的官方说明。

本节以单个昇腾AI处理器为例，描述基本场景性能指标参考，如下表所示，其它分辨率可以等量估算。

下表中的1路对应一个通道，一个通道对应一个线程；各分辨率的含义为：720p指分辨率为1280*720的图片，1080p指分辨率为1920*1080的图片，4K指分辨率为3840*2160的图片。

表 1-60 Atlas 推理系列产品的昇腾 AI 处理器性能指标参考数据

场景举例	总帧率
• 输入图片分辨率：1080p (1920*1080) • 输出图片分辨率：1080p (1920*1080) • 输入/输出图片格式：YUV420SP • n路 (1≤n≤12)	n*360fps
• 输入图片分辨率：1080p (1920*1080) • 输出图片分辨率：1080p (1920*1080) • 输入/输出图片格式：YUV420SP • n路 (n>12)	4320fps
• 输入图片分辨率：4K图像 (3840*2160) • 输出图片分辨率：4K图像 (3840*2160) • 输入/输出图片格式：YUV420SP • n路 (1≤n≤12)	n*90fps
• 输入图片分辨率：4K图像 (3840*2160) • 输出图片分辨率：4K图像 (3840*2160) • 输入/输出图片格式：YUV420SP • n路 (n>12)	1080fps
• 输入图片分辨率：8K图像 (7680*4320) • 输出图片分辨率：4K图像 (3840*2160) • 输入/输出图片格式：YUV420SP • n路 (1≤n≤12)	n*22.5fps
• 输入图片分辨率：8K图像 (7680*4320) • 输出图片分辨率：4K图像 (3840*2160) • 输入/输出图片格式：YUV420SP • n路 (n>12)	270fps

表 1-61 Atlas 200/500 A2 推理产品的昇腾 AI 处理器性能指标参考数据

场景举例	总帧率
- 输入图片分辨率：1080p (1920*1080) - 输出图片分辨率：1080p (1920*1080) - 输入/输出图片格式：YUV420SP - n路 (n<2)	n*800fps
- 输入图片分辨率：1080p (1920*1080) - 输出图片分辨率：1080p (1920*1080) - 输入/输出图片格式：YUV420SP - n路 (n≥2)	1600fps
- 输入图片分辨率：4K图像 (3840*2160) - 输出图片分辨率：4K图像 (3840*2160) - 输入/输出图片格式：YUV420SP - n路 (n<2)	n*200fps
- 输入图片分辨率：4K图像 (3840*2160) - 输出图片分辨率：4K图像 (3840*2160) - 输入/输出图片格式：YUV420SP - n路 (n≥2)	400fps
- 输入图片分辨率：8K图像 (7680*4320) - 输出图片分辨率：4K图像 (3840*2160) - 输入/输出图片格式：YUV420SP - n路 (n<2)	n*100fps
- 输入图片分辨率：8K图像 (7680*4320) - 输出图片分辨率：4K图像 (3840*2160) - 输入/输出图片格式：YUV420SP - n路 (n≥2)	200fps

表 1-62 Atlas A2 训练系列产品/Atlas 800I A2 推理产品的昇腾 AI 处理器性能指标参考数据

场景举例	总帧率
- 输入图片分辨率：1080p (1920*1080) - 输出图片分辨率：1080p (1920*1080) - 输入/输出图片格式：YUV420SP - n路 (1≤n≤10)	n*1440fps

场景举例	总帧率
- 输入图片分辨率：1080p (1920*1080) - 输出图片分辨率：1080p (1920*1080) - 输入/输出图片格式：YUV420SP - n路 (n>10)	14400fps
- 输入图片分辨率：4K图像 (3840*2160) - 输出图片分辨率：4K图像 (3840*2160) - 输入/输出图片格式：YUV420SP - n路 (1≤n≤10)	n*360fps
- 输入图片分辨率：4K图像 (3840*2160) - 输出图片分辨率：4K图像 (3840*2160) - 输入/输出图片格式：YUV420SP - n路 (n>10)	3600fps
- 输入图片分辨率：8K图像 (7680*4320) - 输出图片分辨率：4K图像 (3840*2160) - 输入/输出图片格式：YUV420SP - n路 (1≤n≤10)	n*90fps
- 输入图片分辨率：8K图像 (7680*4320) - 输出图片分辨率：4K图像 (3840*2160) - 输入/输出图片格式：YUV420SP - n路 (n>10)	900fps

 说明

- VPC处理性能与处理过程中的图像分辨率强相关，以输入图像和输出图像中的最大分辨率作为基准分辨率，基准分辨率越大，处理耗时越久，性能越低。
- 调用VPC批处理接口（接口命名中包含batch，例如hi_mpi_vpc_batch_crop_resize_paste接口）时，由于图像处理单元DVPP（Digital Video Pre-Processing）内部多个VPC硬件单元会并行处理图片任务，因此单路就可以达到最大总帧率。

1.9.16.4 hi_mpi_vpc_create_chn

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

函数功能

根据设置的通道属性创建图像处理通道，由用户指定通道号。

约束说明

- Atlas 推理系列产品上，单个Device上的通道号不能超出最大的通道号范围。用户在使用VPC功能或VDEC功能（内部会使用VPC功能）时，都会占用VPC通道号，为防止VPC功能与VDEC功能的通道号冲突，用户在使用VPC功能时，通道号的取值范围为[0,255]；在使用VDEC功能时，VDEC内部使用的VPC通道号范围为[256,511]。
- 如果参数attr为空，会返回错误码HI_ERR_VPC_NULL_PTR。
- 在创建图像处理通道之前必须保证通道未创建（或者已经销毁），否则会直接返回失败。

函数原型

```
hi_s32 hi_mpi_vpc_create_chn(hi_vpc_chn chn, const hi_vpc_chn_attr *attr)
```

参数说明

参数名	输入/输出	说明
chn	输入	图片处理通道号。 Atlas 推理系列产品，该参数的取值范围：[0, 256)，通道总数最多256。 Atlas A2训练系列产品/Atlas 800I A2推理产品，该参数的取值范围：[0, 256)，通道总数最多256。 Atlas 200/500 A2推理产品，该参数的取值范围：[0, 128)，通道总数最多128。 说明 对于Atlas 推理系列产品，昇腾虚拟化实例（Ascend Virtual Instance）场景下，通道总数 = （被分配的VPC硬件单元 / VPC硬件单元总数）* 256，如果通道总数不为整数，则向下取整。 对于Atlas A2训练系列产品/Atlas 800I A2推理产品，昇腾虚拟化实例（Ascend Virtual Instance）场景下，通道总数最多256。 对于Atlas 200/500 A2推理产品，昇腾虚拟化实例（Ascend Virtual Instance）场景下，通道总数最多128。 昇腾虚拟化实例（Ascend Virtual Instance）的详细描述，以及各场景下的算力资源规格请参见《昇腾虚拟化实例（AVI）用户指南》。
attr	输入	图片处理通道属性指针。

返回值说明

- 0：成功
- 非0：失败，参见[1.9.21.10 VPC图像处理返回码](#)

参考资源

接口调用流程及示例，参见VPC图片处理典型功能。

1.9.16.5 hi_mpi_vpc_destroy_chn

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

函数功能

销毁创建的图像处理通道。

约束说明

通道号不能超出最大的通道号范围。

函数原型

[hi_s32](#) hi_mpi_vpc_destroy_chn([hi_vpc_chn](#) chn)

参数说明

参数名	输入/输出	说明
chn	输入	图片处理通道号。 Atlas 推理系列产品，该参数的取值范围：[0, 256)，通道总数最多256。 Atlas A2训练系列产品/Atlas 800I A2推理产品，该参数的取值范围：[0, 256)，通道总数最多256。 Atlas 200/500 A2推理产品，该参数的取值范围：[0, 128)，通道总数最多128。

返回值说明

- 0：成功
- 非0：失败，参见[1.9.21.10 VPC图像处理返回码](#)

参考资源

接口调用流程及示例，参见VPC图片处理典型功能。

1.9.16.6 hi_mpi_vpc_resize

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

函数功能

对图像进行按照一定比例进行缩放或者缩放到固定尺寸。异步接口。

约束说明

版本	约束
Atlas 推理系列产品	- 关于输入图片分辨率、图片格式： - 输入图片分辨率在[10*6, 4096*4096]范围内时，支持图片格式、宽高对齐、内存约束处说明的输入图片格式。 - 输入图片的宽在(4096, 8192]或高在(4096, 8192]范围内时，支持格式如下： 支持hi_pixel_format枚举值中的如下枚举项：<pre>HI_PIXEL_FORMAT_YUV_400 = 0, // YUV400 8bit HI_PIXEL_FORMAT_YUV_SEMIPLANAR_420 = 1, // YUV420SP NV12 8bit HI_PIXEL_FORMAT_YVU_SEMIPLANAR_420 = 2, // YUV420SP NV21 8bit HI_PIXEL_FORMAT_YUV_SEMIPLANAR_422 = 3, // YUV422SP 8bit HI_PIXEL_FORMAT_YVU_SEMIPLANAR_422 = 4, // YVU422SP 8bit HI_PIXEL_FORMAT_YUV_SEMIPLANAR_444 = 5, // YUV444SP 8bit HI_PIXEL_FORMAT_YVU_SEMIPLANAR_444 = 6, // YVU444SP 8bit HI_PIXEL_FORMAT_YUYV_PACKED_422 = 7, // YUV422Packed YUYV 8bit HI_PIXEL_FORMAT_UYVY_PACKED_422 = 8, // YUV422Packed UYVY 8bit HI_PIXEL_FORMAT_YVYU_PACKED_422 = 9, // YUV422Packed YVYU 8bit HI_PIXEL_FORMAT_VYUY_PACKED_422 = 10, // YUV422Packed VYUY 8bit</pre> - 关于各图片格式的宽高对齐、内存大小约束，请参见图片格式、宽高对齐、内存约束。 - 关于输出图片分辨率、图片格式、宽高对齐、内存大小约束请参见1.9.16.2 约束说明。 - 当输入、输出图片格式都为ARGB8888、ABGR8888、RGBA8888、BGRA8888格式时，在经过本接口缩放处理后，A分量值会变为0。 - 设置fx/fy之后，缩放后的图片的分辨率必须在[10*6~4096*4096]范围内，否则返回错误。 - 缩放可以通过fx/fy或者目标图片宽高两种方式来指定，但是两者均配置以目标尺寸缩放优先。两者不可同时为0。

版本	约束
Atlas 200/500 A2推理产品 Atlas A2训练系列产品/Atlas 800I A2推理产品	- 使用本接口对输入、输出图片分辨率的要求如下： - 输入图片分辨率：[10*6, 16384*16384] 当输出图片宽度大于4096时，输入图片最小分辨率为128*16。 当输入图片宽度大于8192时，输入图片高度最小为16。 - 输出图片分辨率：[10*6, 16384*16384] - 关于图片格式、宽高对齐、内存等约束，请参见图片格式、宽高对齐、内存约束。 - 当输入、输出图片格式都为ARGB8888、ABGR8888、RGBA8888、BGRA8888格式时，在经过本接口缩放处理后，A分量值会变为0。 - 设置fx/fy之后，缩放后的图片的分辨率必须在[10*6~16384*16384]范围内，否则返回错误。 - 缩放可以通过fx/fy或者目标图片宽高两种方式来指定，但是两者均配置以目标尺寸缩放优先。两者不可同时为0。

函数原型

hi_s32 hi_mpi_vpc_resize(**hi_vpc_chn** chn, const **hi_vpc_pic_info** *source_pic, **hi_vpc_pic_info** *dest_pic, **hi_double** fx, **hi_double** fy, **hi_u32** interpolation, **hi_u32** *task_id, **hi_s32** milli_sec)

参数说明

参数名	输入/输出	说明
chn	输入	图片处理通道号。 Atlas 推理系列产品，该参数的取值范围：[0, 256)，通道总数最多256。 Atlas A2训练系列产品/Atlas 800I A2推理产品，该参数的取值范围：[0, 256)，通道总数最多256。 Atlas 200/500 A2推理产品，该参数的取值范围：[0, 128)，通道总数最多128。
source_pic	输入	原始图片信息的指针。
dest_pic	输入	目标图片信息的指针。
fx	输入	宽的缩放比例。
fy	输入	高的缩放比例。

参数名	输入/输出	说明
interpolation	输入	缩放算法。此处配置的缩放算法建议与训练模型时的缩放算法保持一致。 Atlas 推理系列产品，支持如下缩放算法： 0：业界通用的Bilinear算法（与OpenCV算法的计算过程类似，当输入和输出图片格式都为RGB时，在[1/32, 512]的缩放范围内，与OpenCV的单个像素值最大差异为正负1） 1：业界通用的Nearest neighbor 算法（与OpenCV算法的计算过程类似） Atlas A2训练系列产品/Atlas 800I A2推理产品，支持如下缩放算法： 0：业界通用的Bilinear算法（与OpenCV-3.4.2版本算法的计算结果相同） 1：业界通用的Nearest neighbor 算法（与OpenCV-3.4.2版本算法的计算结果相同） Atlas 200/500 A2推理产品，支持如下缩放算法： 0：业界通用的Bilinear算法（与OpenCV-3.4.2版本算法的计算结果相同） 1：业界通用的Nearest neighbor 算法（与OpenCV-3.4.2版本算法的计算结果相同）
task_id	输出	任务ID的指针，用来区分任务。
milli_sec	输入	超时时间配置，单位是毫秒，取值范围如下： -1：阻塞方式 0：非阻塞方式 >0：超时方式，配置具体的超时时间。超时时间受操作系统影响，一般偏差在操作系统的—个时间片内，例如，操作系统的—个时间片为4ms，用户设置的milli_sec参数值为1，则实际的超时时间在1ms到5ms范围内。在CPU负载高场景下，超时时间仍可能存在波动。

返回值说明

- 0：成功
- 非0：失败，参见[1.9.21.10 VPC图像处理返回码](#)

参考资源

接口调用流程及示例，参见VPC图片处理典型功能。

1.9.16.7 hi_mpi_vpc_crop

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

函数功能

按照指定区域从一张输入图片中抠出一张或多张子图。异步接口。

约束说明

版本	约束
Atlas 推理系列产品	- 关于输入图片分辨率、图片格式： - 输入图片分辨率在[10*6, 4096*4096]范围内时，支持图片格式、宽高对齐、内存约束处说明的输入图片格式。 - 输入图片的宽在(4096, 8192]或高在(4096, 8192]范围内时，支持格式如下： 支持hi_pixel_format枚举值中的如下枚举项：<pre>HI_PIXEL_FORMAT_YUV_400 = 0, // YUV400 8bit HI_PIXEL_FORMAT_YUV_SEMIPLANAR_420 = 1, // YUV420SP NV12 8bit HI_PIXEL_FORMAT_YVU_SEMIPLANAR_420 = 2, // YUV420SP NV21 8bit HI_PIXEL_FORMAT_YUV_SEMIPLANAR_422 = 3, // YUV422SP 8bit HI_PIXEL_FORMAT_YVU_SEMIPLANAR_422 = 4, // YVU422SP 8bit HI_PIXEL_FORMAT_YUV_SEMIPLANAR_444 = 5, // YUV444SP 8bit HI_PIXEL_FORMAT_YVU_SEMIPLANAR_444 = 6, // YVU444SP 8bit HI_PIXEL_FORMAT_YUYV_PACKED_422 = 7, // YUV422Packed YUYV 8bit HI_PIXEL_FORMAT_UYVY_PACKED_422 = 8, // YUV422Packed UYVY 8bit HI_PIXEL_FORMAT_YVYU_PACKED_422 = 9, // YUV422Packed YVYU 8bit HI_PIXEL_FORMAT_VYUY_PACKED_422 = 10, // YUV422Packed VYUY 8bit</pre> - 关于各图片格式的宽高对齐、内存大小约束，请参见图片格式、宽高对齐、内存约束。 - 关于输出图片分辨率、图片格式、宽高对齐、内存大小约束请参见1.9.16.2 约束说明。 - 单张1080P的图像处理超时时间建议设置3ms，以此类推，如果是2张1080P的图像，建议设置超时时间为6ms；如果是3张1080P图片，建议设置超时时间为9ms。其他分辨率，按照图像尺寸进行近似的等价折算，例如，3840*2160分辨率的图像建议超时时间设置为12ms。

版本	约束
Atlas 200/500 A2推理产品 Atlas A2训练系列产品/Atlas 800I A2推理产品	- 关于图片分辨率、图片格式、宽高对齐、内存等约束，请参见1.9.16.2 约束说明。 - 单张1080P的图像处理超时时间建议设置3ms，以此类推，如果是2张1080P的图像，建议设置超时时间为6ms；如果是3张1080P图片，建议设置超时时间为9ms。其他分辨率，按照图像尺寸进行近似的等价折算，例如，3840*2160分辨率的图像建议超时时间设置为12ms。

函数原型

```
hi_s32 hi_mpi_vpc_crop(hi_vpc_chn chn, const hi_vpc_pic_info *source_pic,
hi_vpc_crop_region_info crop_info[], hi_u32 count, hi_u32 *task_id, hi_s32
milli_sec)
```

参数说明

参数名	输入/输出	说明
chn	输入	图片处理通道号。 Atlas 推理系列产品，该参数的取值范围：[0, 256)，通道总数最多256。 Atlas A2训练系列产品/Atlas 800I A2推理产品，该参数的取值范围：[0, 256)，通道总数最多256。 Atlas 200/500 A2推理产品，该参数的取值范围：[0, 128)，通道总数最多128。
source_pic	输入	原始图片信息的指针。
crop_info	输入	抠图图片信息数组，该数组长度与count参数值保持一致。
count	输入	抠图区域的数量，取值范围[1,256]。
task_id	输出	任务ID的指针，用来区分任务。
milli_sec	输入	超时时间配置，单位是毫秒，取值范围如下： -1：阻塞方式 0：非阻塞方式 >0：超时方式，配置具体的超时时间。超时时间受操作系统影响，一般偏差在操作系统的片内，例如，操作系统的片为4ms，用户设置的milli_sec参数值为1，则实际的超时时间在1ms到5ms范围内。在CPU负载高场景下，超时时间仍可能存在波动。

返回值说明

- 0：成功
- 非0：失败，参见[1.9.21.10 VPC图像处理返回码](#)

参考资源

接口调用流程及示例，参见VPC图片处理典型功能。

1.9.16.8 hi_mpi_vpc_crop_resize

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

函数功能

从一张大图中扣出一张或多张子图，并缩放到指定尺寸。异步接口。

约束说明

版本	约束
Atlas 推理系列产品	- 关于输入图片分辨率、图片格式： - 输入图片分辨率在[10*6, 4096*4096]范围内时，支持图片格式、宽高对齐、内存约束处说明的输入图片格式。 - 输入图片的宽在(4096, 8192]或高在(4096, 8192]范围内时，支持格式如下： 支持hi_pixel_format枚举值中的如下枚举项： HI_PIXEL_FORMAT_YUV_400 = 0, // YUV400 8bit HI_PIXEL_FORMAT_YUV_SEMIPLANAR_420 = 1, // YUV420SP NV12 8bit HI_PIXEL_FORMAT_YVU_SEMIPLANAR_420 = 2, // YUV420SP NV21 8bit HI_PIXEL_FORMAT_YUV_SEMIPLANAR_422 = 3, // YUV422SP 8bit HI_PIXEL_FORMAT_YVU_SEMIPLANAR_422 = 4, // YVU422SP 8bit HI_PIXEL_FORMAT_YUV_SEMIPLANAR_444 = 5, // YUV444SP 8bit HI_PIXEL_FORMAT_YVU_SEMIPLANAR_444 = 6, // YVU444SP 8bit HI_PIXEL_FORMAT_YUYV_PACKED_422 = 7, // YUV422Packed YUYV 8bit HI_PIXEL_FORMAT_UYVY_PACKED_422 = 8, // YUV422Packed UYVY 8bit HI_PIXEL_FORMAT_YVYU_PACKED_422 = 9, // YUV422Packed YVYU 8bit HI_PIXEL_FORMAT_VYUY_PACKED_422 = 10, // YUV422Packed VYUY 8bit - 关于各图片格式的宽高对齐、内存大小约束，请参见图片格式、宽高对齐、内存约束。 - 关于输出图片分辨率、图片格式、宽高对齐、内存大小约束请参见1.9.16.2 约束说明。 - 当输入、输出图片格式都为ARGB8888、ABGR8888、RGBA8888、BGRA8888格式时，在经过本接口缩放处理后，A分量值会变为0。 - 若是crop的宽高与resize之后的宽高一致，则不进行缩放；resize宽高必须与输出宽高一致。
Atlas 200/500 A2推理产品 Atlas A2训练系列产品/ Atlas 800I A2推理产品	- 关于图片分辨率、图片格式、宽高对齐、内存等约束，请参见1.9.16.2 约束说明。 - 当输入、输出图片格式都为ARGB8888、ABGR8888、RGBA8888、BGRA8888格式时，在经过本接口缩放处理后，A分量值会变为0。 - 若是crop的宽高与resize之后的宽高一致，则不进行缩放；resize宽高必须与输出宽高一致。

函数原型

hi_s32 hi_mpi_vpc_crop_resize(**hi_vpc_chn** chn, const **hi_vpc_pic_info** *source_pic, **hi_vpc_crop_resize_region** crop_resize_info[], **hi_u32** count, **hi_u32** *task_id, **hi_s32** milli_sec)

参数说明

参数名	输入/输出	说明
chn	输入	图片处理通道号。 Atlas 推理系列产品，该参数的取值范围：[0, 256)，通道总数最多256。 Atlas A2训练系列产品/Atlas 800I A2推理产品，该参数的取值范围：[0, 256)，通道总数最多256。 Atlas 200/500 A2推理产品，该参数的取值范围：[0, 128)，通道总数最多128。
source_pic	输入	原始图片信息的指针。
crop_resize_info	输入	抠图并缩放的图片信息数组，该数组长度与count参数值保持一致。
count	输入	抠图并缩放的图片数量，取值范围[1,256]。
task_id	输出	任务ID的指针，用来区分任务。
milli_sec	输入	超时时间配置，单位是毫秒，取值范围如下： • -1：阻塞方式 • 0：非阻塞方式 • >0：超时方式，配置具体的超时时间。超时时间受操作系统影响，一般偏差在操作系统的片内，例如，操作系统的片为4ms，用户设置的milli_sec参数值为1，则实际的超时时间在1ms到5ms范围内。在CPU负载高场景下，超时时间仍可能存在波动。

返回值说明

- 0：成功
- 非0：失败，参见[1.9.21.10 VPC图像处理返回码](#)

1.9.16.9 hi_mpi_vpc_crop_resize_paste

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

函数功能

从输入图片中抠图，对抠出的子图进行缩放后，贴在用户输出图片的指定区域。异步接口。

约束说明

版本	约束
Atlas 推理系列产品	- 关于输入图片分辨率、图片格式： - 输入图片分辨率在[10*6, 4096*4096]范围内时，支持图片格式、宽高对齐、内存约束处说明的输入图片格式。 - 输入图片的宽在(4096, 8192]或高在(4096, 8192]范围内时，支持格式如下： 支持hi_pixel_format枚举值中的如下枚举项： HI_PIXEL_FORMAT_YUV_400 = 0, // YUV400 8bit HI_PIXEL_FORMAT_YUV_SEMIPLANAR_420 = 1, // YUV420SP NV12 8bit HI_PIXEL_FORMAT_YVU_SEMIPLANAR_420 = 2, // YUV420SP NV21 8bit HI_PIXEL_FORMAT_YUV_SEMIPLANAR_422 = 3, // YUV422SP 8bit HI_PIXEL_FORMAT_YVU_SEMIPLANAR_422 = 4, // YVU422SP 8bit HI_PIXEL_FORMAT_YUV_SEMIPLANAR_444 = 5, // YUV444SP 8bit HI_PIXEL_FORMAT_YVU_SEMIPLANAR_444 = 6, // YVU444SP 8bit HI_PIXEL_FORMAT_YUYV_PACKED_422 = 7, // YUV422Packed YUYV 8bit HI_PIXEL_FORMAT_UYVY_PACKED_422 = 8, // YUV422Packed UYVY 8bit HI_PIXEL_FORMAT_YVYU_PACKED_422 = 9, // YUV422Packed YVYU 8bit HI_PIXEL_FORMAT_VYUY_PACKED_422 = 10, // YUV422Packed VYUY 8bit - 关于各图片格式的宽高对齐、内存大小约束，请参见图片格式、宽高对齐、内存约束。 - 关于输出图片分辨率、图片格式、宽高对齐、内存大小约束请参见1.9.16.2 约束说明。 - 当输入、输出图片格式都为ARGB8888、ABGR8888、RGBA8888、BGRA8888格式时，在经过本接口缩放处理后，A分量值会变为0。 - 输出图片的数量最多等于抠图区域的数量，最少是一张图片。也就是说，如果从输入图片中扣n张子图，那么贴图的目标图片可以是同一张大图，也可以是多张大图。取决于hi_vpc_crop_resize_paste_region结构体中指定的地址。 - 由于hi_vpc_crop_resize_paste_region结构体内容比较丰富，使用者可以自由组合功能。例如：若是crop的宽高与resize之后的宽高一致，则不进行缩放。 - 使用贴图功能时，如果resize之后的宽度非16对齐，则会存在随机数据影响输出图片（贴图之外的区域）。

版本	约束
Atlas 200/500 A2推理产品 Atlas A2训练系列产品/ Atlas 800I A2推理产品	- 关于图片分辨率、图片格式、宽高对齐、内存等约束，请参见1.9.16.2 约束说明。 - 由于YUV格式图像下采样约束，当输出图片格式为YUV420SP或YUV422SP格式时，贴图的顶部偏移、左侧偏移需为偶数。 - 当输入、输出图片格式都为ARGB8888、ABGR8888、RGBA8888、BGRA8888格式时，在经过本接口缩放处理后，A分量值会变为0。 - 输出图片的数量最多等于抠图区域的数量，最少是一张图片。也就是说，如果从输入图片中扣n张子图，那么贴图的目标图片可以是同一张大图，也可以是多张大图。取决于hi_vpc_crop_resize_paste_region结构体中指定的地址。 - 由于hi_vpc_crop_resize_paste_region结构体内容比较丰富，使用者可以自由组合功能。例如：若是crop的宽高与resize之后的宽高一致，则不进行缩放。

函数原型

[hi_s32](#) [hi_mpi_vpc_crop_resize_paste](#)([hi_vpc_chn](#) chn, const [hi_vpc_pic_info](#) *source_pic, [hi_vpc_crop_resize_paste_region](#) crop_resize_paste_info[], [hi_u32](#) count, [hi_u32](#) *task_id, [hi_s32](#) milli_sec)

参数说明

参数名	输入/输出	说明
chn	输入	图片处理通道号。 Atlas 推理系列产品，该参数的取值范围：[0, 256)，通道总数最多256。 Atlas A2训练系列产品/Atlas 800I A2推理产品，该参数的取值范围：[0, 256)，通道总数最多256。 Atlas 200/500 A2推理产品，该参数的取值范围：[0, 128)，通道总数最多128。
source_pic	输入	原始图片信息的指针。
crop_resize_paste_info	输入	抠图缩放并贴图的图片信息数组，该数组长度与count参数值保持一致。
count	输入	抠图缩放贴图的图片数量，取值范围[1,256]。
task_id	输出	任务ID的指针，用来区分任务。

参数名	输入/输出	说明
milli_sec	输入	超时时间配置，单位是毫秒，取值范围如下： • -1：阻塞方式 • 0：非阻塞方式 • >0：超时方式，配置具体的超时时间。超时时间受操作系统影响，一般偏差在操作系统的-一个时间片内，例如，操作系统的-一个时间片为4ms，用户设置的milli_sec参数值为1，则实际的超时时间在1ms到5ms范围内。在CPU负载高场景下，超时时间仍可能存在波动。

返回值说明

- 0：成功
- 非0：失败，参见[1.9.21.10 VPC图像处理返回码](#)

1.9.16.10 hi_mpi_vpc_convert_color

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

函数功能

转换图片的格式。异步接口。

约束说明

版本	约束
Atlas 推理系列产品	• 输入、输出图片分辨率：[10*6, 4096*4096]。 • 输入图片格式、输出图片格式、宽高对齐约束、内存约束请参见图片格式、宽高对齐、内存约束。当输入格式是YUV400时，输出格式只能是YUV400。
Atlas 200/500 A2推理产品 Atlas A2训练系列产品/ Atlas 800I A2推理产品	• 输入、输出图片分辨率为[10*6, 4096*8192]。 • 输入图片格式、输出图片格式、宽高对齐约束、内存约束请参见图片格式、宽高对齐、内存约束。

函数原型

```
hi_s32 hi_mpi_vpc_convert_color(hi_vpc_chn chn, const hi_vpc_pic_info
*source_pic, hi_vpc_pic_info *dest_pic, hi_u32 *task_id, hi_s32 milli_sec)
```


参数说明

参数名	输入/输出	说明
chn	输入	图片处理通道号。 Atlas 推理系列产品, 该参数的取值范围: [0, 256), 通道总数最多256。 Atlas A2训练系列产品/Atlas 800I A2推理产品, 该参数的取值范围: [0, 256), 通道总数最多256。 Atlas 200/500 A2推理产品, 该参数的取值范围: [0, 128), 通道总数最多128。
source_pic	输入	原始图片信息的指针。
dest_pic	输入	目标图片信息的指针。 需指定输出图片的宽、高、宽stride、高stride、内存地址、内存大小、格式等。 - 配置输出图片的宽、高时, 需与输入图片的宽、高保持一致, 否则会返回报错。 如果将输出图片的宽或高配置为0, 则VPC内部会将输入图片的宽、高分别作为输出图片的宽、高; 同时, VPC内部会按对齐要求计算宽stride、高stride, 不同图片格式的对齐要求不同, 请参见表1-59。 - 配置输出图片的宽stride、高stride时, 满足对齐要求即可, 不同图片格式的对齐要求不同, 请参见表1-59。
task_id	输出	任务ID的指针, 用来区分任务。
milli_sec	输入	超时时间配置, 单位是毫秒, 取值范围如下: -1: 阻塞方式 0: 非阻塞方式 >0: 超时方式, 配置具体的超时时间。超时时间受操作系统影响, 一般偏差在操作系统的—个时间片内, 例如, 操作系统的—个时间片为4ms, 用户设置的milli_sec参数值为1, 则实际的超时时间在1ms到5ms范围内。在CPU负载高场景下, 超时时间仍可能存在波动。

返回值说明

- 0: 成功
- 非0: 失败, 参见[1.9.21.10 VPC图像处理返回码](#)

1.9.16.11 hi_mpi_vpc_convert_color_v2

Atlas 200/300/500 推理产品, 不支持该接口。

Atlas 训练系列产品, 不支持该接口。

函数功能

转换图片的格式，在[hi_mpi_vpc_convert_color](#)接口的基础上扩展功能，支持设置RGBA格式图片的透明度。异步接口。

约束说明

版本	约束
Atlas 推理系列产品	- 输入、输出图片分辨率：[10*6, 4096*4096]。 - 输入图片格式、输出图片格式、宽高对齐约束、内存约束请参见图片格式、宽高对齐、内存约束。当输入格式是YUV400时，输出格式只能是YUV400。
Atlas 200/500 A2推理产品 Atlas A2训练系列产品/Atlas 800I A2推理产品	- 输入、输出图片分辨率为[10*6, 4096*8192]。 - 输入图片格式、输出图片格式、宽高对齐约束、内存约束请参见图片格式、宽高对齐、内存约束。

函数原型

[hi_s32](#) [hi_mpi_vpc_convert_color_v2](#)([hi_vpc_chn](#) chn, const [hi_vpc_pic_info](#) *source_pic, [hi_vpc_pic_info](#) *dest_pic, [hi_csc_conf](#) *conf, [hi_u32](#) *task_id, [hi_s32](#) milli_sec)

参数说明

参数名	输入/输出	说明
chn	输入	图片处理通道号。 Atlas 推理系列产品，该参数的取值范围：[0, 256)，通道总数最多256。 Atlas A2训练系列产品/Atlas 800I A2推理产品，该参数的取值范围：[0, 256)，通道总数最多256。 Atlas 200/500 A2推理产品，该参数的取值范围：[0, 128)，通道总数最多128。
source_pic	输入	原始图片信息的指针。

参数名	输入/输出	说明
dest_pic	输入	目标图片信息的指针。 需指定输出图片的宽、高、宽stride、高stride、内存地址、内存大小、格式等。 - 配置输出图片的宽、高时，需与输入图片的宽、高保持一致，否则会返回报错。 如果将输出图片的宽或高配置为0，则VPC内部会将输入图片的宽、高分别作为输出图片的宽、高；同时，VPC内部会按对齐要求计算宽stride、高stride，不同图片格式的对齐要求不同，请参见表1-59。 - 配置输出图片的宽stride、高stride时，满足对齐要求即可，不同图片格式的对齐要求不同，请参见表1-59。
conf	输入	输出图片透明度值的指针。 透明度只针对以下图像格式有效： PIXEL_FORMAT_ARGB_8888 = 14, PIXEL_FORMAT_ABGR_8888 = 15, PIXEL_FORMAT_RGBA_8888 = 16, PIXEL_FORMAT_BGRA_8888 = 17,
task_id	输出	任务ID的指针，用来区分任务。
milli_sec	输入	超时时间配置，单位是毫秒，取值范围如下： -1：阻塞方式 0：非阻塞方式 >0：超时方式，配置具体的超时时间。超时时间受操作系统影响，一般偏差在操作系统的片一个时间片内，例如，操作系统的片一个时间片为4ms，用户设置的milli_sec参数值为1，则实际的超时时间在1ms到5ms范围内。在CPU负载高场景下，超时时间仍可能存在波动。

返回值说明

- 0：成功
- 非0：失败，参见1.9.21.10 VPC图像处理返回码

1.9.16.12 hi_mpi_vpc_convert_color_to_yuv420

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

函数功能

转换图片的格式，且转换后的格式仅支持YUV420 semi-planar/YVU420 semi-planar。异步接口。

约束说明

- 本接口兼容旧昇腾AI处理器版本上的格式转换接口，兼容场景下的输入、输出图片格式如下，兼容场景下输出的二进制相同：
 - 输入图片格式：YUV420 semi-planar/YVU420 semi-planar/YUV422 semi-planar/YVU422 semi-planar/YUV444 packed/YUV444 semi-planar/YVU444 semi-planar
 - 输出图片格式：YUV420 semi-planar/YVU420 semi-planar
 - 在以上输入图片格式、输出图片格式场景下，本接口与 [hi_mpi_vpc_convert_color](#) 接口输出的二进制不一致。
- 关于输入分辨率的要求：

版本	约束
Atlas 推理系列产品	输入图片分辨率：[10*6, 4096*4096]。
Atlas 200/500 A2推理产品 Atlas A2训练系列产品/ Atlas 800I A2推理产品	输入图片分辨率：[10*6,4096*8192]。

- 输入图片格式支持[hi_pixel_format](#)枚举值中的如下枚举项：

```
HI_PIXEL_FORMAT_YUV_SEMIPLANAR_420 = 1,    // YUV420SP NV12 8bit
HI_PIXEL_FORMAT_YVU_SEMIPLANAR_420 = 2,    // YUV420SP NV21 8bit
HI_PIXEL_FORMAT_YUV_SEMIPLANAR_422 = 3,    // YUV422SP 8bit
HI_PIXEL_FORMAT_YVU_SEMIPLANAR_422 = 4,    // YVU422SP 8bit
HI_PIXEL_FORMAT_YUV_SEMIPLANAR_444 = 5,    // YUV444SP 8bit
HI_PIXEL_FORMAT_YVU_SEMIPLANAR_444 = 6,    // YVU444SP 8bit
HI_PIXEL_FORMAT_YUVV_PACKED_422 = 7,       // YUV422Packed YUYV 8bit
HI_PIXEL_FORMAT_UYVY_PACKED_422 = 8,       // YUV422Packed UYVY 8bit
HI_PIXEL_FORMAT_VYVU_PACKED_422 = 9,       // YUV422Packed VYVU 8bit
HI_PIXEL_FORMAT_VYUY_PACKED_422 = 10,      // YUV422Packed VYUY 8bit
HI_PIXEL_FORMAT_YUV_PACKED_444 = 11,       // YUV444Packed 8bit
HI_PIXEL_FORMAT_RGB_888 = 12,              // RGB888
HI_PIXEL_FORMAT_BGR_888 = 13,              // BGR888
HI_PIXEL_FORMAT_ARGB_8888 = 14,            // ARGB8888
HI_PIXEL_FORMAT_ABGR_8888 = 15,            // ABGR8888
HI_PIXEL_FORMAT_RGBA_8888 = 16,            // RGBA8888
HI_PIXEL_FORMAT_BGRA_8888 = 17,            // BGRA8888
HI_PIXEL_FORMAT_YUV_SEMIPLANAR_440 = 1000, // YUV440SP 8bit
HI_PIXEL_FORMAT_YVU_SEMIPLANAR_440 = 1001, // YVU440SP 8bit
```
- 输出图片格式支持[hi_pixel_format](#)枚举值中的如下枚举项，设置其它图片格式无效，接口会返回失败：

```
HI_PIXEL_FORMAT_YUV_SEMIPLANAR_420 = 1,    // YUV420SP NV12 8bit
HI_PIXEL_FORMAT_YVU_SEMIPLANAR_420 = 2,    // YUV420SP NV21 8bit
```
- 关于各图片格式的宽高对齐、内存大小约束，请参见[图片格式、宽高对齐、内存约束](#)。

函数原型

```
hi_s32 hi_mpi_vpc_convert_color_to_yuv420(hi_vpc_chn chn, const
hi_vpc_pic_info *source_pic, hi_vpc_pic_info *dest_pic, hi_u32 *task_id, hi_s32
milli_sec)
```

参数说明

参数名	输入/输出	说明
chn	输入	图片处理通道号。 Atlas 推理系列产品, 该参数的取值范围: [0, 256), 通道总数最多256。 Atlas A2训练系列产品/Atlas 800I A2推理产品, 该参数的取值范围: [0, 256), 通道总数最多256。 Atlas 200/500 A2推理产品, 该参数的取值范围: [0, 128), 通道总数最多128。
source_pic	输入	原始图片信息的指针。
dest_pic	输入	目标图片信息的指针。 需指定输出图片的宽、高、宽stride、高stride、内存地址、内存大小、格式等。 - 配置输出图片的宽、高时, 需与输入图片的宽、高保持一致, 否则会返回报错。 如果将输出图片的宽或高配置为0, 则VPC内部会将输入图片的宽、高分别作为输出图片的宽、高; 同时, VPC内部会按对齐要求计算宽stride、高stride, 不同图片格式的对齐要求不同, 请参见表1-59。 - 配置输出图片的宽stride、高stride时, 满足对齐要求即可, 不同图片格式的对齐要求不同, 请参见表1-59。
task_id	输出	任务ID的指针, 用来区分任务。
milli_sec	输入	超时时间配置, 单位是毫秒, 取值范围如下: -1: 阻塞方式 0: 非阻塞方式 >0: 超时方式, 配置具体的超时时间。超时时间受操作系统影响, 一般偏差在操作系统的—个时间片内, 例如, 操作系统的—个时间片为4ms, 用户设置的milli_sec参数值为1, 则实际的超时时间在1ms到5ms范围内。在CPU负载高场景下, 超时时间仍可能存在波动。

返回值说明

- 0: 成功
- 非0: 失败, 参见[1.9.21.10 VPC图像处理返回码](#)

1.9.16.13 hi_mpi_vpc_copy_make_border

Atlas 200/300/500 推理产品, 不支持该接口。

Atlas 训练系列产品, 不支持该接口。

函数功能

对输入图像进行边界填充。异步接口。

约束说明

版本	约束
Atlas 推理系列产品	• 输入、输出图片分辨率为[10*6, 4096*4096]。 • 输入、输出图片的格式、各图片格式的宽高对齐、内存大小约束，请参见图片格式、宽高对齐、内存约束。 扩充图像的宽没有16对齐的要求，但是内存申请仍然要按照16的对齐要求进行申请，即Stride要按照16对齐。 • 由于YUV格式图像下采样约束，当输出图片格式为YUV420SP或YUV422SP格式时，需注意： - 对于YUV420SP输出格式，上下左右填充的尺寸建议为偶数； - 对于YUV422SP输出格式，左右填充的尺寸建议为偶数。

版本	约束
Atlas 200/500 A2推理产品 Atlas A2训练系列产品/ Atlas 800I A2推理产品	- 输入、输出图片分辨率为[10*6, 4096*8192]。 - 输入、输出图片的格式、各图片格式的宽高对齐、内存大小约束，请参见图片格式、宽高对齐、内存约束。 - 根据make_border_info.border_type参数设置的扩充边缘像素的类型，本接口支持的输入、输出图片格式有所不同： - 当参数值设置为HI_BORDER_CONSTANT或HI_BORDER_REPLICATE时，支持的输入、输出图片格式请参见图片格式、宽高对齐、内存约束； - 当参数值设置为HI_BORDER_REFLECT或HI_BORDER_REFLECT_101时，支持的输入、输出图片格式如下所示：<pre>HI_PIXEL_FORMAT_YUV_400 = 0, // YUV400 8bit HI_PIXEL_FORMAT_YUYV_PACKED_422 = 7, // YUV422Packed YUYV 8bit HI_PIXEL_FORMAT_UYVY_PACKED_422 = 8, // YUV422Packed UYVY 8bit HI_PIXEL_FORMAT_YVYU_PACKED_422 = 9, // YUV422Packed YVYU 8bit HI_PIXEL_FORMAT_VYUY_PACKED_422 = 10, // YUV422Packed VYUY 8bit HI_PIXEL_FORMAT_YUV_PACKED_444 = 11, // YUV444 Package 8bit HI_PIXEL_FORMAT_RGB_888 = 12, // RGB888 HI_PIXEL_FORMAT_BGR_888 = 13, // BGR888 HI_PIXEL_FORMAT_ARGB_8888 = 14, // ARGB8888 HI_PIXEL_FORMAT_ABGR_8888 = 15, // ABGR8888 HI_PIXEL_FORMAT_RGBA_8888 = 16, // RGBA8888 HI_PIXEL_FORMAT_BGRA_8888 = 17, // BGRA8888 HI_PIXEL_FORMAT_YVU_PACKED_444 = 58, // YVU444 Package 8bit</pre> - 各图片格式的宽高对齐、内存大小约束，请参见图片格式、宽高对齐、内存约束。 - 由于YUV格式图像下采样约束，当输出图片格式为YUV420SP或YUV422SP格式时，需注意： - 对于YUV420SP输出格式，上下左右填充的尺寸建议为偶数； - 对于YUV422SP输出格式，左右填充的尺寸建议为偶数。

函数原型

[hi_s32](#) hi_mpi_vpc_copy_make_border([hi_vpc_chn](#) chn, const [hi_vpc_pic_info](#) *source_pic, [hi_vpc_pic_info](#) *dest_pic, [hi_vpc_make_border_info](#) make_border_info, [hi_u32](#) *task_id, [hi_s32](#) milli_sec)

参数说明

参数名	输入/输出	说明
chn	输入	图片处理通道号。 Atlas 推理系列产品，该参数的取值范围：[0, 256)，通道总数最多256。 Atlas A2训练系列产品/Atlas 800I A2推理产品，该参数的取值范围：[0, 256)，通道总数最多256。 Atlas 200/500 A2推理产品，该参数的取值范围：[0, 128)，通道总数最多128。
source_pic	输入	原始图片信息的指针。
dest_pic	输入	目标图片信息的指针。
make_border_info	输入	边界填充信息。
task_id	输出	任务ID的指针，用来区分任务。
milli_sec	输入	超时时间配置，单位是毫秒，取值范围如下： -1：阻塞方式 0：非阻塞方式 >0：超时方式，配置具体的超时时间。超时时间受操作系统影响，一般偏差在操作系统的 一个时间片内，例如，操作系统的 一个时间片为4ms，用户设置的 milli_sec 参数值为1，则实际的超时时间在1ms到5ms 范围内。在CPU负载高场景下，超时时间仍可能存在波动。

返回值说明

- 0：成功
- 非0：失败，参见[1.9.21.10 VPC图像处理返回码](#)

1.9.16.14 hi_mpi_vpc_pyrdown

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas 200/500 A2推理产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

对图像进行金字塔缩放，当前仅支持YUV400的图片格式。异步接口。

函数原型

hi_s32 hi_mpi_vpc_pyrdown(**hi_vpc_chn** chn, const **hi_vpc_pic_info** *source_pic, **hi_vpc_pic_info** dest_pic[], **hi_u32** filter_level, **hi_s8** gaussian_filter[][5], **hi_u16** divisor, **hi_vpc_make_border_info** make_border_info, **hi_u32** *task_id, **hi_s32** milli_sec)

参数说明

参数名	输入/输出	说明
chn	输入	图片处理通道号。 Atlas 推理系列产品，该参数的取值范围：[0, 256)，通道总数最多256。
source_pic	输入	输入图片信息的指针。输入图片分辨率最大支持2048*2048，仅支持YUV 400格式输入。根据filter_level, 最小输入分辨率限制如下： filter_level = 1，最小输入分辨率为20*12 filter_level = 2，最小输入分辨率为40*24 filter_level = 3，最小输入分辨率为80*48 filter_level = 4，最小输入分辨率为160*96
dest_pic	输入	目标图片信息，该数组长度与filter_level参数值保持一致。 dest_pic[0]保存的是原图宽高都缩小为1/2的图， dest_pic[1]保存的是原图宽高都缩小为1/4的图， dest_pic[2]保存的是原图宽高都缩小为1/8的图， dest_pic[3]保存的是原图宽高都缩小为1/16的图。
filter_level	输入	参数有效范围是1-4，指定金字塔图像层数以及图像数量。
gaussian_filter	输入	高斯滤波参数。 将该参数设置为空指针，表示使用如下默认值： {1, 4, 6, 4, 1}, {4, 16, 24, 16, 4}, {6, 24, 36, 24, 6}, {4, 16, 24, 16, 4}, {1, 4, 6, 4, 1}}
divisor	输入	滤波器除数，必须是2的幂次方。
make_border_info	输入	边界填充信息。 固定填充2个像素，支持的填充类型为： HI_BORDER_CONSTANT、HI_BOARD_REPLICATE和 HI_BOARD_REFLECT。
task_id	输出	任务ID的指针，用来区分任务。

参数名	输入/输出	说明
milli_sec	输入	超时时间配置，单位是毫秒，取值范围如下： -1：阻塞方式 0：非阻塞方式 >0：超时方式，配置具体的超时时间。超时时间受操作系统影响，一般偏差在操作系统的片内，例如，操作系统的片为4ms，用户设置的milli_sec参数值为1，则实际的超时时间在1ms到5ms范围内。在CPU负载高场景下，超时时间仍可能存在波动。

返回值说明

- 0：成功
- 非0：失败，参见[1.9.21.10 VPC图像处理返回码](#)

1.9.16.15 hi_mpi_vpc_calc_hist

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

函数功能

统计图像每个通道的像素值分布。异步接口。

函数原型

[hi_s32](#) hi_mpi_vpc_calc_hist([hi_vpc_chn](#) chn, const [hi_vpc_pic_info](#) *source_pic, [hi_vpc_histogram_config](#) *hist_config, [hi_u32](#) *task_id, [hi_s32](#) milli_sec)

约束说明

- 关于输入分辨率的要求：

版本	约束
Atlas 推理系列产品	输入图片分辨率：[10*6, 4096*4096]。
Atlas 200/500 A2推理产品 Atlas A2训练系列产品/ Atlas 800I A2推理产品	输入图片分辨率：[10*6, 4096*8192]。

- 输入图片格式支持[hi_pixel_format](#)枚举值中的如下枚举项：

HI_PIXEL_FORMAT_YUV_400 = 0, // YUV400 8bit
HI_PIXEL_FORMAT_YUV_SEMIPLANAR_420 = 1, // YUV420SP NV12 8bit
HI_PIXEL_FORMAT_YVU_SEMIPLANAR_420 = 2, // YUV420SP NV21 8bit
HI_PIXEL_FORMAT_YUV_SEMIPLANAR_422 = 3, // YUV422SP 8bit

```
HI_PIXEL_FORMAT_YVU_SEMIPLANAR_422 = 4, // YVU422SP 8bit
HI_PIXEL_FORMAT_YUV_SEMIPLANAR_444 = 5, // YUV444SP 8bit
HI_PIXEL_FORMAT_YVU_SEMIPLANAR_444 = 6, // YVU444SP 8bit
HI_PIXEL_FORMAT_YUYV_PACKED_422 = 7, // YUV422Packed YUYV 8bit
HI_PIXEL_FORMAT_UYVY_PACKED_422 = 8, // YUV422Packed UYVY 8bit
HI_PIXEL_FORMAT_YVYU_PACKED_422 = 9, // YUV422Packed YVYU 8bit
HI_PIXEL_FORMAT_VYUY_PACKED_422 = 10, // YUV422Packed VYUY 8bit
HI_PIXEL_FORMAT_YUV_PACKED_444 = 11, // YUV444P 8bit
HI_PIXEL_FORMAT_RGB_888 = 12, // RGB888
HI_PIXEL_FORMAT_BGR_888 = 13, // BGR888
HI_PIXEL_FORMAT_YUV_SEMIPLANAR_440 = 1000, // YUV440SP 8bit
HI_PIXEL_FORMAT_YVU_SEMIPLANAR_440 = 1001, // YVU440SP 8bit
```

- 关于各图片格式的宽高对齐、内存大小约束，请参见[图片格式、宽高对齐、内存约束](#)。

参数说明

参数名	输入/输出	说明
chn	输入	图片处理通道号。 Atlas 推理系列产品，该参数的取值范围：[0, 256)，通道总数最多256。 Atlas A2训练系列产品/Atlas 800I A2推理产品，该参数的取值范围：[0, 256)，通道总数最多256。 Atlas 200/500 A2推理产品，该参数的取值范围：[0, 128)，通道总数最多128。
source_pic	输入	输入图片信息的指针。 说明 当图片格式为YUV440SP 8bit、YVU440SP 8bit时，Y分量统计准确，UV分量统计上存在误差。
hist_config	输出	统计结果的指针。 hist_config是一个包含3个数组的结构体，用来存放每张图像3个分量的0-255的分布情况。
task_id	输出	任务ID的指针，用来区分任务。
milli_sec	输入	超时时间配置，单位是毫秒，取值范围如下： -1：阻塞方式 0：非阻塞方式 >0：超时方式，配置具体的超时时间。超时时间受操作系统影响，一般偏差在操作系统的片内，例如，操作系统的片为4ms，用户设置的milli_sec参数值为1，则实际的超时时间在1ms到5ms范围内。在CPU负载高场景下，超时时间仍可能存在波动。

返回值说明

- hist_config：统计结果。

- task_id: 此次任务分配的ID，用来区分任务。

1.9.16.16 hi_mpi_vpc_equalize_hist

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

函数功能

将输入图片中指定分量的指定像素值重映射成其它像素值，作为输出图片，主要用于图像增强的场景。通过图片的色彩重映射以达到图片对比度更好，清晰度更好的效果。异步接口。

约束说明

- 输入图片分辨率为[10*6, 4096*4096]。
- 输入图片格式支持[hi_pixel_format](#)枚举值中的如下枚举项：

```
HI_PIXEL_FORMAT_YUV_400 = 0,           // YUV400 8bit
HI_PIXEL_FORMAT_YUV_SEMIPLANAR_420 = 1, // YUV420SP NV12 8bit
HI_PIXEL_FORMAT_YVU_SEMIPLANAR_420 = 2, // YUV420SP NV21 8bit
HI_PIXEL_FORMAT_YUV_SEMIPLANAR_422 = 3, // YUV422SP 8bit
HI_PIXEL_FORMAT_YVU_SEMIPLANAR_422 = 4, // YVU422SP 8bit
HI_PIXEL_FORMAT_YUV_PACKED_444 = 11,    // YUV444P 8bit
HI_PIXEL_FORMAT_RGB_888 = 12,           // RGB888
HI_PIXEL_FORMAT_BGR_888 = 13,           // BGR888
```
- 输出图片格式与输入图片格式保持一致。
- 关于各图片格式的宽高对齐、内存大小约束，请参见[图片格式、宽高对齐、内存约束](#)。

函数原型

hi_s32 hi_mpi_vpc_equalize_hist(**hi_vpc_chn** chn, const **hi_vpc_pic_info*** source_pic, **hi_vpc_pic_info** *dest_pic, const **hi_vpc_lut_remap** *lut_remap, **hi_u32** *task_id, **hi_s32** milli_sec)

参数说明

参数名	输入/输出	说明
chn	输入	图片处理通道号。 Atlas 推理系列产品，该参数的取值范围：[0, 256)，通道总数最多256。 Atlas A2训练系列产品/Atlas 800I A2推理产品，该参数的取值范围：[0, 256)，通道总数最多256。 Atlas 200/500 A2推理产品，该参数的取值范围：[0, 128)，通道总数最多128。
source_pic	输入	输入图片信息的指针。
dest_pic	输入	目标图片信息的指针。

参数名	输入/输出	说明
lut_remap	输入	图像lut_remap的配置的指针。 lut_remap是一个768 Byte的数组，配置给硬件进行图片像素重映射。
task_id	输出	任务ID的指针，用来区分任务。
milli_sec	输入	超时时间配置，单位是毫秒，取值范围如下： • -1：阻塞方式 • 0：非阻塞方式 • >0：超时方式，配置具体的超时时间。超时时间受操作系统影响，一般偏差在操作系统的<code>一个时间片</code>内，例如，操作系统的<code>一个时间片</code>为4ms，用户设置的<code>milli_sec</code>参数值为1，则实际的超时时间在1ms到5ms范围内。在CPU负载高场景下，超时时间仍可能存在波动。

返回值说明

- 0：成功
- 非0：失败，参见[1.9.21.10 VPC图像处理返回码](#)

1.9.16.17 hi_mpi_vpc_crop_resize_make_border

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

函数功能

按指定区域从一张输入图片中抠出一个或多个子图，对子图缩放后，再将每个子图按指定类型填充，作为一张或多张目标图片输出，主要用于等比例缩放场景。异步接口。

约束说明

版本	约束
Atlas 推理系列产品	- 关于输入图片分辨率、图片格式： - 输入图片分辨率在[10*6, 4096*4096]范围内时，支持图片格式、宽高对齐、内存约束处说明的输入图片格式。 - 输入图片的宽在(4096, 8192]或高在(4096, 8192]范围内时，支持格式如下： 支持hi_pixel_format枚举值中的如下枚举项： HI_PIXEL_FORMAT_YUV_400 = 0, // YUV400 8bit HI_PIXEL_FORMAT_YUV_SEMIPLANAR_420 = 1, // YUV420SP NV12 8bit HI_PIXEL_FORMAT_YVU_SEMIPLANAR_420 = 2, // YUV420SP NV21 8bit HI_PIXEL_FORMAT_YUV_SEMIPLANAR_422 = 3, // YUV422SP 8bit HI_PIXEL_FORMAT_YVU_SEMIPLANAR_422 = 4, // YVU422SP 8bit HI_PIXEL_FORMAT_YUV_SEMIPLANAR_444 = 5, // YUV444SP 8bit HI_PIXEL_FORMAT_YVU_SEMIPLANAR_444 = 6, // YVU444SP 8bit HI_PIXEL_FORMAT_YUVV_PACKED_422 = 7, // YUV422Packed YUYV 8bit HI_PIXEL_FORMAT_UYVY_PACKED_422 = 8, // YUV422Packed UYVY 8bit HI_PIXEL_FORMAT_YVYU_PACKED_422 = 9, // YUV422Packed YVYU 8bit HI_PIXEL_FORMAT_VYUY_PACKED_422 = 10, // YUV422Packed VYUY 8bit - 关于各图片格式的宽高对齐、内存大小约束，请参见图片格式、宽高对齐、内存约束。 - 关于输出图片分辨率、图片格式、宽高对齐、内存大小约束请参见1.9.16.2 约束说明。 - 当输入、输出图片格式都为ARGB8888、ABGR8888、RGBA8888、BGRA8888格式时，在经过本接口缩放处理后，A分量值会变为0。 - 由于hi_vpc_crop_resize_border_region结构体内容比较丰富，使用者可以自由组合功能。但存在一些约束，例如：若是crop的宽高与resize之后的宽高一致，则不进行缩放；若偏移值为0，则不进行填充。 - 使用本接口实现等比例缩放功能时，相关的描述及约束请参见抠图、贴图约束。 - 由于YUV格式图像下采样约束，当输出图片格式为YUV420SP或YUV422SP格式时，需注意： - 对于YUV420SP输出格式，上下左右填充的尺寸建议为偶数； - 对于YUV422SP输出格式，左右填充的尺寸建议为偶数。

版本	约束
Atlas 200/500 A2推理产品 Atlas A2训练系列产品/ Atlas 800I A2推理产品	- 关于图片分辨率、图片格式、宽高对齐、内存等约束，请参见1.9.16.2 约束说明。 - 设置fx/fy之后，缩放后的图片的分辨率必须在[10*6, 16384*16384]范围内，否则返回错误。 - 当输入、输出图片格式都为ARGB8888、ABGR8888、RGBA8888、BGRA8888格式时，在经过本接口缩放处理后，A分量值会变为0。 - 由于hi_vpc_crop_resize_border_region结构体内容比较丰富，使用者可以自由组合功能。但存在一些约束，例如：若是crop的宽高与resize之后的宽高一致，则不进行缩放；若偏移值为0，则不进行填充。 - 使用本接口实现等比例缩放功能时，相关的描述及约束请参见抠图、贴图约束。 - 由于YUV格式图像下采样约束，当输出图片格式为YUV420SP或YUV422SP格式时，需注意： - 对于YUV420SP输出格式，上下左右填充的尺寸建议为偶数； - 对于YUV422SP输出格式，左右填充的尺寸建议为偶数。

函数原型

[hi_s32](#) [hi_mpi_vpc_crop_resize_make_border](#)([hi_vpc_chn](#) chn, const [hi_vpc_pic_info](#) *source_pic, [hi_vpc_crop_resize_border_region](#) crop_resize_make_border_info[], [hi_u32](#) count, [hi_u32](#) *task_id, [hi_s32](#) milli_sec)

参数说明

参数名	输入/输出	说明
chn	输入	图片处理通道号。 Atlas 推理系列产品，该参数的取值范围：[0, 256)，通道总数最多256。 Atlas A2训练系列产品/Atlas 800I A2推理产品，该参数的取值范围：[0, 256)，通道总数最多256。 Atlas 200/500 A2推理产品，该参数的取值范围：[0, 128)，通道总数最多128。
source_pic	输入	原始图片信息的指针。

参数名	输入/输出	说明
crop_resize_make_b_order_info	输入	抠图缩放填充的图片信息数组，需要用户设置抠图、缩放、填充信息以及目标图片的内存地址，VPC将抠图缩放填充后的结果数据存放在用户指定的内存地址中。如果多个填充结果的数据都指向同一个内存地址，后一次的结果数据会覆盖前一次的结果数据，导致输出图片跟预期不一致。 当前填充类型仅支持HI_BORDER_CONSTANT和HI_BORDER_REPLICATE，可以Padding到4096*4096。crop_resize_make_border_info数组的长度与count参数值保持一致。
count	输入	抠图缩放填充的图片数量，取值范围[1,256]。
task_id	输出	任务ID的指针，用来区分任务。
milli_sec	输入	超时时间配置，单位是毫秒，取值范围如下： • -1：阻塞方式 • 0：非阻塞方式 • >0：超时方式，配置具体的超时时间。超时时间受操作系统影响，一般偏差在操作系统的 一个时间片内，例如，操作系统的 一个时间片为4ms，用户设置的milli_sec参数值为1，则实际的超时时间在1ms到5ms范围内。在CPU负载高场景下，超时时间仍可能存在波动。

返回值说明

- 0：成功
- 非0：失败，参见[1.9.21.10 VPC图像处理返回码](#)

1.9.16.18 hi_mpi_vpc_batch_crop_resize_paste

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

函数功能

[hi_mpi_vpc_crop_resize_paste](#)的扩展接口，支持一次处理多张输入图片。可输入多张原图，从每张原图中抠一张或多张子图缩放并贴图到每张目标图上，目标图片的数量与子图数量保持一致。异步接口。

约束说明

版本	约束
Atlas 推理系列产品	- 关于输入图片分辨率、图片格式： - 输入图片分辨率在[10*6, 4096*4096]范围内时，支持图片格式、宽高对齐、内存约束处说明的输入图片格式。 - 输入图片的宽在(4096, 8192]或高在(4096, 8192]范围内时，支持格式如下： 支持hi_pixel_format枚举值中的如下枚举项：<pre>HI_PIXEL_FORMAT_YUV_400 = 0, // YUV400 8bit HI_PIXEL_FORMAT_YUV_SEMIPLANAR_420 = 1, // YUV420SP NV12 8bit HI_PIXEL_FORMAT_YVU_SEMIPLANAR_420 = 2, // YUV420SP NV21 8bit HI_PIXEL_FORMAT_YUV_SEMIPLANAR_422 = 3, // YUV422SP 8bit HI_PIXEL_FORMAT_YVU_SEMIPLANAR_422 = 4, // YVU422SP 8bit HI_PIXEL_FORMAT_YUV_SEMIPLANAR_444 = 5, // YUV444SP 8bit HI_PIXEL_FORMAT_YVU_SEMIPLANAR_444 = 6, // YVU444SP 8bit HI_PIXEL_FORMAT_YUYV_PACKED_422 = 7, // YUV422Packed YUYV 8bit HI_PIXEL_FORMAT_UYVY_PACKED_422 = 8, // YUV422Packed UYVY 8bit HI_PIXEL_FORMAT_YVYU_PACKED_422 = 9, // YUV422Packed YVYU 8bit HI_PIXEL_FORMAT_VYUY_PACKED_422 = 10, // YUV422Packed VYUY 8bit</pre> - 关于各图片格式的宽高对齐、内存大小约束，请参见图片格式、宽高对齐、内存约束。 - 关于输出图片分辨率、图片格式、宽高对齐、内存大小约束请参见1.9.16.2 约束说明。 - 当输入、输出图片格式都为ARGB8888、ABGR8888、RGBA8888、BGRA8888格式时，在经过本接口缩放处理后，A分量值会变为0。 - 由于hi_vpc_crop_resize_paste_region结构体内容比较丰富，使用者可以自由组合功能。但存在一些通用约束，例如： - 若是crop的宽高和原图宽高一致，则不进行crop。 - 若是crop的宽高与resize之后的宽高一致，则不进行缩放。 - 若是crop、resize之后的宽高和目标图片一致，并且贴图左上角坐标等于(0,0)，则不进行贴图。

版本	约束
Atlas 200/500 A2推理产品 Atlas A2训练系列产品/Atlas 800I A2推理产品	- 关于图片分辨率、各图片格式的宽高对齐、内存大小约束，请参见1.9.16.2 约束说明。 - 当输入、输出图片格式都为ARGB8888、ABGR8888、RGBA8888、BGRA8888格式时，在经过本接口缩放处理后，A分量值会变为0。 - 由于YUV格式图像下采样约束，当输出图片格式为YUV420SP或YUV422SP格式时，贴图的顶部偏移、左侧偏移建议为偶数。 - 由于hi_vpc_crop_resize_paste_region结构体内容比较丰富，使用者可以自由组合功能。但存在一些通用约束，例如： - 若是crop的宽高和原图宽高一致，则不进行crop。 - 若是crop的宽高与resize之后的宽高一致，则不进行缩放。 - 若是crop、resize之后的宽高和目标图片一致，并且贴图左上角坐标等于(0,0)，则不进行贴图。

函数原型

[hi_s32](#) hi_mpi_vpc_batch_crop_resize_paste([hi_vpc_chn](#) chn, const [hi_vpc_pic_info](#) *source_pic[], [hi_u32](#) pic_num, [hi_vpc_crop_resize_paste_region](#) crop_resize_paste_info[], [hi_u32](#) count[], [hi_u32](#) *task_id, [hi_s32](#) milli_sec)

参数说明

参数名	输入/输出	说明
chn	输入	图片处理通道号。 Atlas 推理系列产品，该参数的取值范围：[0, 256)，通道总数最多256。 Atlas A2训练系列产品/Atlas 800I A2推理产品，该参数的取值范围：[0, 256)，通道总数最多256。 Atlas 200/500 A2推理产品，该参数的取值范围：[0, 128)，通道总数最多128。
source_pic	输入	原始图片信息的指针数组。
pic_num	输入	原始图片数量，与source_pic数组长度、count数组长度保持一致。

参数名	输入/输出	说明
crop_resize_paste_info	输入	抠图缩放并贴图信息结构体数组，需要用户设置抠图、缩放、贴图信息以及目标图片的内存地址，VPC将抠图缩放贴图后的结果数据存放在用户指定的内存地址中。
count	输入	每张原图抠图缩放贴图的图片数量，数组内第一个元素的值表示第一张原图的抠图缩放贴图的图片数量，第二个元素的值表示第二张原图的抠图缩放贴图的图片数量，以此类推。count数组内元素值之和等于crop_resize_paste_info数组的长度。count数组内元素值之和的取值范围[1,256]。
task_id	输出	任务ID的指针，用来区分任务。
milli_sec	输入	超时时间配置，单位是毫秒，取值范围如下： • -1：阻塞方式 • 0：非阻塞方式 • >0：超时方式，配置具体的超时时间。超时时间受操作系统影响，一般偏差在操作系统的—个时间片内，例如，操作系统的—个时间片为4ms，用户设置的milli_sec参数值为1，则实际的超时时间在1ms到5ms范围内。在CPU负载高场景下，超时时间仍可能存在波动。

返回值说明

- 0：成功
- 非0：失败，参见[1.9.21.10 VPC图像处理返回码](#)

1.9.16.19 hi_mpi_vpc_batch_crop_resize_make_border

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

函数功能

[hi_mpi_vpc_batch_crop_resize_make_border](#)的扩展接口，支持批处理模式。可输入多张原图，按指定区域从每张输入图片中抠出一个或多个子图，对子图缩放后，再将子图贴到每张目标图片（目标图片的数量与子图数量保持一致）的指定位置，区域之外按指定类型填充。异步接口。

约束说明

版本	约束
Atlas 推理系列产品	- 关于输入图片分辨率、图片格式： - 输入图片分辨率在[10*6, 4096*4096]范围内时，支持图片格式、宽高对齐、内存约束处说明的输入图片格式。 - 输入图片的宽在(4096, 8192]或高在(4096, 8192]范围内时，支持格式如下： 支持hi_pixel_format枚举值中的如下枚举项： HI_PIXEL_FORMAT_YUV_400 = 0, // YUV400 8bit HI_PIXEL_FORMAT_YUV_SEMIPLANAR_420 = 1, // YUV420SP NV12 8bit HI_PIXEL_FORMAT_YVU_SEMIPLANAR_420 = 2, // YUV420SP NV21 8bit HI_PIXEL_FORMAT_YUV_SEMIPLANAR_422 = 3, // YUV422SP 8bit HI_PIXEL_FORMAT_YVU_SEMIPLANAR_422 = 4, // YVU422SP 8bit HI_PIXEL_FORMAT_YUV_SEMIPLANAR_444 = 5, // YUV444SP 8bit HI_PIXEL_FORMAT_YVU_SEMIPLANAR_444 = 6, // YVU444SP 8bit HI_PIXEL_FORMAT_YUYV_PACKED_422 = 7, // YUV422Packed YUYV 8bit HI_PIXEL_FORMAT_UYVY_PACKED_422 = 8, // YUV422Packed UYVY 8bit HI_PIXEL_FORMAT_YVYU_PACKED_422 = 9, // YUV422Packed YVYU 8bit HI_PIXEL_FORMAT_VYUY_PACKED_422 = 10, // YUV422Packed VYUY 8bit - 关于各图片格式的宽高对齐、内存大小约束，请参见图片格式、宽高对齐、内存约束。 - 关于输出图片分辨率、图片格式、宽高对齐、内存大小约束请参见1.9.16.2 约束说明。 - 当输入、输出图片格式都为ARGB8888、ABGR8888、RGBA8888、BGRA8888格式时，在经过本接口缩放处理后，A分量值会变为0。 - 由于hi_vpc_crop_resize_border_region结构体内容比较丰富，使用者可以自由组合功能。但存在一些通用约束，例如： - 若是crop的宽高和原图宽高一致，则不进行crop。 - 若是crop的宽高与resize之后的宽高一致，则不进行缩放。 - 若是crop、resize之后的宽高和目标图片的宽高一致，并且左上角坐标等于(0,0)，则不进行填充。 - 使用本接口实现等比例缩放功能时，相关的描述及约束请参见抠图、贴图约束。 - 由于YUV格式图像下采样约束，当输出图片格式为YUV420SP或YUV422SP格式时，需注意：

版本	约束
	- 对于YUV420SP输出格式，上下左右填充的尺寸建议为偶数； - 对于YUV422SP输出格式，左右填充的尺寸建议为偶数。
Atlas 200/500 A2推理产品 Atlas A2训练系列产品/Atlas 800I A2推理产品	• 关于图片分辨率、各图片格式的宽高对齐、内存大小约束，请参见1.9.16.2 约束说明。 • 当输入、输出图片格式都为ARGB8888、ABGR8888、RGBA8888、BGRA8888格式时，在经过本接口缩放处理后，A分量值会变为0。 • 由于hi_vpc_crop_resize_border_region结构体内容比较丰富，使用者可以自由组合功能。但存在一些通用约束，例如： - 若是crop的宽高和原图宽高一致，则不进行crop。 - 若是crop的宽高与resize之后的宽高一致，则不进行缩放。 - 若是crop、resize之后的宽高和目标图片的宽高一致，并且左上角坐标等于(0,0)，则不进行填充。 • 使用本接口实现等比例缩放功能时，相关的描述及约束请参见抠图、贴图约束。 • 由于YUV格式图像下采样约束，当输出图片格式为YUV420SP或YUV422SP格式时，需注意： - 对于YUV420SP输出格式，上下左右填充的尺寸建议为偶数； - 对于YUV422SP输出格式，左右填充的尺寸建议为偶数。

函数原型

```
hi_s32 hi_mpi_vpc_batch_crop_resize_make_border(hi_vpc_chn chn, const  
hi_vpc_pic_info *source_pic[], hi_u32 pic_num,  
hi_vpc_crop_resize_border_region crop_resize_make_border_info[], hi_u32  
count[], hi_u32 *task_id, hi_s32 milli_sec)
```

参数说明

参数名	输入/输出	说明
chn	输入	图片处理通道号。 Atlas 推理系列产品，该参数的取值范围：[0, 256)，通道总数最多256。 Atlas A2训练系列产品/Atlas 800I A2推理产品，该参数的取值范围：[0, 256)，通道总数最多256。 Atlas 200/500 A2推理产品，该参数的取值范围：[0, 128)，通道总数最多128。
source_pic	输入	原始图片信息的指针数组。
pic_num	输入	原始图片数量，与source_pic数组长度、count数组长度保持一致。
crop_resize_make_border_info	输入	抠图缩放填充的信息结构体，需要用户设置抠图、缩放、填充信息以及目标图片的内存地址，VPC将抠图缩放填充后的结果数据存放在用户指定的内存地址中。当前填充类型仅支持HI_BORDER_CONSTANT和HI_BORDER_REPLICATE，可以填充到4096*4096。
count	输入	每张原图抠图缩放填充的图片数量，count数组内元素值之和等于crop_resize_make_border_info数组的长度。count数组内元素值之和的取值范围[1,256]。
task_id	输出	任务ID的指针，用来区分任务。
milli_sec	输入	超时时间配置，单位是毫秒，取值范围如下： • -1：阻塞方式 • 0：非阻塞方式 • >0：超时方式，配置具体的超时时间。超时时间受操作系统影响，一般偏差在操作系统的—个时间片内，例如，操作系统的—个时间片为4ms，用户设置的milli_sec参数值为1，则实际的超时时间在1ms到5ms范围内。在CPU负载高场景下，超时时间仍可能存在波动。

返回值说明

- 0：成功
- 非0：失败，参见[1.9.21.10 VPC图像处理返回码](#)

1.9.16.20 hi_mpi_vpc_get_process_result

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

函数功能

等待任务处理完成，再获取图片处理结果。

由于VPC功能接口是异步接口（例如：[hi_mpi_vpc_crop](#)），因此在调用VPC功能接口之后，还需要调用本接口等待任务处理完成，然后再获取处理结果。

函数原型

```
hi_s32 hi_mpi_vpc_get_process_result(hi_vpc_chn chn, hi_u32 task_id, hi_s32 milli_sec)
```

参数说明

参数名	输入/输出	说明
chn	输入	图片处理通道号。 Atlas 推理系列产品，该参数的取值范围：[0, 256)，通道总数最多256。 Atlas A2训练系列产品/Atlas 800I A2推理产品，该参数的取值范围：[0, 256)，通道总数最多256。 Atlas 200/500 A2推理产品，该参数的取值范围：[0, 128)，通道总数最多128。
task_id	输入	任务ID。 此处通过任务ID来明确需要获取哪次任务执行后的结果数据。 任务ID可以在调用抠图、缩放等接口后获取，请参见 1.9.16.6 hi_mpi_vpc_resize~1.9.16.19 hi_mpi_vpc_batch_crop_resize_make_border 。
milli_sec	输入	超时时间配置，单位是毫秒，取值范围如下： • -1：阻塞方式 • 0：非阻塞方式 • >0：超时方式，配置具体的超时时间。超时时间受操作系统影响，一般偏差在操作系统的<code>一个时间片</code>内，例如，操作系统的<code>一个时间片</code>为4ms，用户设置的<code>milli_sec</code>参数值为1，则实际的超时时间在1ms到5ms范围内。在CPU负载高场景下，超时时间仍可能存在波动。

返回值说明

- 0：成功
- 非0：失败，参见[1.9.21.10 VPC图像处理返回码](#)

参考资源

接口调用流程及示例，参见VPC图片处理典型功能。

1.9.16.21 hi_mpi_vpc_sys_create_chn

Atlas 200/300/500 推理产品，不支持该接口。
Atlas 训练系列产品，不支持该接口。

函数功能

根据设置的通道属性创建图像处理通道，由系统返回可用通道号给用户。

函数原型

```
hi_s32 hi_mpi_vpc_sys_create_chn(hi_vpc_chn *chnl, const hi_vpc_chn_attr *attr)
```

参数说明

参数名	输入/输出	说明
chnl	输出	图片处理通道号的指针，调用该接口后，由系统返回可用通道号给用户。 Atlas 推理系列产品，该参数的取值范围：[0, 256)，通道总数最多256。 Atlas A2训练系列产品/Atlas 800I A2推理产品，该参数的取值范围：[0, 256)，通道总数最多256。 Atlas 200/500 A2推理产品，该参数的取值范围：[0, 128)，通道总数最多128。
attr	输入	图片处理通道属性指针。 如果参数attr为空，会返回错误码HI_ERR_VPC_NULL_PTR。

返回值说明

- 0：成功
- 非0：失败，参见[1.9.21.10 VPC图像处理返回码](#)

1.9.16.22 hi_mpi_vpc_set_roundview_stitching_param

Atlas 200/300/500 推理产品，不支持该接口。
Atlas 200/500 A2推理产品，不支持该接口。
Atlas 训练系列产品，不支持该接口。
Atlas 推理系列产品，不支持该接口。
Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

设置环视拼接参数，包括畸变矫正、增益补偿参数。

约束说明

- 输入、输出图片宽高必须准确配置，分辨率为[10, 6]到[4096,4096]。
- 针对一个通道，只能调用本接口设置一次环视拼接参数。
- 设置拼接参数时，拼接表中的图片编号需设置成功，否则可能出现拼接后的输出图片异常，具体图片编号的设置请参见[1.9.20.12.19 hi_stiching_ipm_table](#)。

函数原型

[hi_s32](#) [hi_mpi_vpc_set_roundview_stitching_param](#)([hi_vpc_chn](#) chn, const [hi_roundview_stitching_param](#) *stitch_param)

参数说明

参数名	输入/输出	说明
chn	输入	图片处理通道号。
stitch_param	输入	环视拼接参数信息，包括畸变矫正、增益补偿参数。

返回值说明

- 0：成功
- 非0：失败，参见[1.9.21.10 VPC图像处理返回码](#)

1.9.16.23 hi_mpi_vpc_get_roundview_stitching_param

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 200/500 A2推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

获取环视拼接参数，包括畸变矫正、增益补偿参数。

约束说明

调用本接口前，必须正确设置ipm_table_address、dest_pic_width、dest_pic_height和ipm_table_size参数。

函数原型

```
hi_s32 hi_mpi_vpc_get_roundview_stitching_param(hi_vpc_chn chn,  
hi_roundview_stitching_param *stitch_param)
```

参数说明

参数名	输入/输出	说明
chn	输入	图片处理通道号。
stitch_param	输出	环视拼接参数。 输出结果中的imp_param.ipm_table_address中的float类型参数值，与用户设置输入时的参数存在浮点精度差异（不超过0.01）。

返回值说明

- 0：成功
- 非0：失败，参见[1.9.21.10 VPC图像处理返回码](#)

1.9.16.24 hi_mpi_vpc_roundview_stitching

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 200/500 A2推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

下发环视拼接任务。

约束说明

不能在一个通道内并发下发多个环视拼接任务。需要[hi_mpi_vpc_get_process_result](#)后，再通过hi_mpi_vpc_roundview_stitching接口下发环视拼接任务。

函数原型

```
hi_s32 hi_mpi_vpc_roundview_stitching(hi_vpc_chn chn,  
hi_roundview_stitching_pic_param *roundview_stitching_pic_param, hi_u32  
*task_id, hi_s32 milli_sec)
```

参数说明

参数名	输入/输出	说明
chn	输入	图片处理通道号。
roundview_stitching_pic_param	输入	输入和输出图片信息。 - 输入： - 输入图片数量：当前环视拼接功能仅支持对4张输入图片； - 输入图片格式：支持YUV420SP、YVU420SP； - 输入图片分辨率：取值范围在10*6~4096*4096之间，包含10*6和4096*4096。 - 输出： - 输出图片数量：1张图片； - 输出图片格式：支持YUV420SP、YVU420SP、RGB888、BGR888； - 输出图片分辨率：取值范围在10*6~4096*4096之间，包含10*6和4096*4096。
milli_sec	输入	超时时间，单位是毫秒。 -1：阻塞方式 0：非阻塞方式 >0：超时方式，配置具体的超时时间。
task_id	输出	此次任务分配的ID，用来区分任务。

返回值说明

- 0：成功
- 非0：失败，参见[1.9.21.10 VPC图像处理返回码](#)

1.9.16.25 hi_mpi_vpc_crop_resize_resize_paste

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 200/500 A2推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

从输入图片中抠图，对抠出的子图进行两次缩放后（两次缩放目的是对敏感图像信息模糊脱敏），贴在用户输出图片的指定区域。异步接口。

约束说明

- 输出图片的数量最多等于抠图区域的数量，最少是一张图片。也就是说，如果从输入图片中扣n张子图，那么贴图的目标图片可以是同一张大图，也可以是多张大图。取决于hi_vpc_crop_resize_resize_paste_region结构体中指定的地址。
- 当输入、输出图片格式都为ARGB8888、ABGR8888、RGBA8888、BGRA8888格式时，在经过本接口缩放处理后，A分量值会变为0。
- VPC功能的约束请参见[1.9.16.2 约束说明](#)。

函数原型

```
hi_s32 hi_mpi_vpc_crop_resize_resize_paste(hi_vpc_chn chn, const  
hi_vpc_pic_info *source_pic, hi_vpc_crop_resize_resize_paste_region  
crop_resize_resize_paste_info[], hi_u32 count, hi_u32 *task_id, hi_s32 milli_sec)
```

参数说明

参数名	输入/输出	说明
chn	输入	图片处理通道号。
source_pic	输入	原始图片信息的指针。
crop_resize_resize_paste_info	输入	抠图缩放并贴图的图片信息数组，该数组长度与count参数值保持一致。
count	输入	抠图缩放贴图的图片数量，取值范围[1,256]。
task_id	输出	任务ID的指针，用来区分任务。
milli_sec	输入	超时时间配置，单位是毫秒，取值范围如下： -1：阻塞方式 0：非阻塞方式 >0：超时方式，配置具体的超时时间。超时时间受操作系统影响，一般偏差在操作系统的—个时间片内，例如，操作系统的—个时间片为4ms，用户设置的milli_sec参数值为1，则实际的超时时间在1ms到5ms范围内。在CPU负载高场景下，超时时间仍可能存在波动。

返回值说明

- 0：成功
- 非0：失败，参见[1.9.21.10 VPC图像处理返回码](#)

1.9.16.26 hi_mpi_vpc_set_chn_workspace

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

函数功能

由于硬件限制，在执行Remap、仿射变换等任务时，需要提前申请临时缓存，因此可调用本接口申请临时缓存调用本接口后，系统内部会根据输入图片、输出图片的最大宽高信息申请需使用的内部临时缓存，通过该接口申请的临时缓存，在销毁通道时会自动释放。同步接口。

函数原型

```
hi_s32 hi_mpi_vpc_set_chn_workspace(hi_vpc_chn chn,  
hi_vpc_workspace_param *workspace_param, hi_u32 param_cnt)
```

约束说明

对于每个通道，该接口只能设置一次，后续不能修改。

参数说明

参数名	输入/输出	说明
chn	输入	图片处理通道号。 Atlas A2训练系列产品/Atlas 800I A2推理产品，该参数的取值范围：[0, 256)，通道总数最多256。 Atlas 200/500 A2推理产品，该参数的取值范围：[0, 128)，通道总数最多128。
workspace_param	输入	指向workspace_param类型数组的指针。 此处需根据所使用的Remap功能或仿射变换功能来设置hi_vpc_workspace_param.func参数值。 若在一个应用程序中，涉及Remap、仿射变换两种功能，则可以在workspace_param数组中包含2个元素，一个元素的hi_vpc_workspace_param.func设置为REMAP，一个元素的hi_vpc_workspace_param.func设置为AFFINE。
param_cnt	输入	数组长度，需与workspace_param数组长度保持一致。

返回值说明

- 0：成功
- 非0：失败，[1.9.21.10 VPC图像处理返回码](#)

1.9.16.27 hi_mpi_vpc_get_lut_mem_size

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

函数功能

根据输出图片宽高信息获取像素位置重映射表所需的内存大小。同步接口。

函数原型

```
hi_s32 hi_mpi_vpc_get_lut_mem_size(hi_u32 width, hi_u32 height, hi_u32 *lut_size)
```

约束说明

调用本接口时，width * height的取值范围为：[10*6, 4096*8192]。

参数说明

参数名	输入/输出	说明
width	输入	输出图像宽度。
height	输入	输出图像高度。
lut_size	输出	像素位置重映射表的内存大小，单位Byte。

返回值说明

- 0：成功
- 非0：失败，[1.9.21.10 VPC图像处理返回码](#)

1.9.16.28 hi_mpi_vpc_get_affine_lut

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

函数功能

根据输入、输出图片中的像素位置坐标点，获取仿射变换像素位置重映射表。同步接口。

函数原型

```
hi_s32 hi_mpi_vpc_get_affine_lut(hi_point_pair_info *point_pair_info, hi_u32 interpolation, hi_remap_lut *remap_lut)
```

参数说明

参数名	输入/输出	说明
point_pair_info	输入	三对输入、输出图片中的像素点坐标信息，每对中分别包含一个输入图片像素点坐标、一个输出图片像素点坐标。
interpolation	输入	缩放算法。 支持如下缩放算法： 0：业界通用的Bilinear算法（与OpenCV算法的计算过程类似） 1：业界通用的Nearest neighbor 算法（与OpenCV算法的计算过程类似）
remap_lut	输入&输出	仿射变换像素位置重映射表信息。 在hi_remap_lut结构体内： - lut参数作为输入时，需由用户提前申请该内存，内存大小lut_size可调用hi_mpi_vpc_get_lut_mem_size接口获取。 - lut参数作为输出时，在成功调用本接口后，用户可从该内存中获取仿射变换LUT表信息。

返回值说明

- 0：成功
- 非0：失败，[1.9.21.10 VPC图像处理返回码](#)

1.9.16.29 hi_mpi_vpc_get_perspective_lut

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

函数功能

根据输入、输出图片中的像素位置坐标点，获取透视变换像素位置重映射表。同步接口。

函数原型

[hi_s32](#) hi_mpi_vpc_get_perspective_lut([hi_point_pair_info](#) *point_pair_info, [hi_u32](#) interpolation, [hi_remap_lut](#) *remap_lut)

参数说明

参数名	输入/输出	说明
point_pair_info	输入	四对输入、输出图片中的像素点坐标信息，每对中分别包含一个输入图片像素点坐标、一个输出图片像素点坐标。
interpolation	输入	支持如下缩放算法： 0：业界通用的Bilinear算法（与OpenCV算法的计算过程类似） 1：业界通用的Nearest neighbor 算法（与OpenCV算法的计算过程类似）
remap_lut	输入输出	透视变换像素位置重映射表。 在 hi_remap_lut 结构体内： - lut参数作为输入时，需由用户提前申请该内存，内存大小lut_size可调用hi_mpi_vpc_get_lut_mem_size接口获取。 - lut参数作为输出时，在成功调用本接口后，用户可从该内存中获取仿射变换LUT表信息。

返回值说明

- 0：成功
- 非0：失败，[1.9.21.10 VPC图像处理返回码](#)

1.9.16.30 hi_mpi_vpc_get_remap_lut

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

函数功能

根据用户输入的Remap表（例如仿射、透视、鱼眼等），采用业界通用的Bilinear算法（与OpenCV算法的计算过程类似），获取系统使用的像素位置重映射表。同步接口。

函数原型

```
hi_s32 hi_mpi_vpc_get_remap_lut(hi_map_param *map_param, hi_remap_lut *remap_lut)
```


参数说明

参数名	输入/输出	说明
map_param	输入	用户原始map表信息。
remap_lut	输入&输出	像素位置重映射表。 在 hi_remap_lut 结构体内： - lut参数作为输入时，需由用户提前申请该内存，内存大小lut_size可调用hi_mpi_vpc_get_lut_mem_size接口获取。 - lut参数作为输出时，在成功调用本接口后，用户可从该内存中获取仿射变换LUT表信息。

返回值说明

- 0：成功
- 非0：失败，[1.9.21.10 VPC图像处理返回码](#)

1.9.16.31 hi_mpi_vpc_lut_remap

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

函数功能

基于输入像素位置重映射表的方式对图片进行Remap处理。异步接口。

约束说明

版本	约束
Atlas 200/500 A2推理产品	- 输入、输出图片分辨率的取值范围为：[10*6, 4096*8192]。 - 支持如下输入图片格式： HI_PIXEL_FORMAT_YUV_400 = 0, // YUV400 8bit HI_PIXEL_FORMAT_YUV_SEMIPLANAR_420 = 1, // YUV420SP NV12 8bit HI_PIXEL_FORMAT_YVU_SEMIPLANAR_420 = 2, // YUV420SP NV21 8bit HI_PIXEL_FORMAT_RGB_888 = 12, // RGB888 HI_PIXEL_FORMAT_BGR_888 = 13, // BGR888 HI_PIXEL_FORMAT_YUV_PACKED_444 = 11, // YUV444P 8bit HI_PIXEL_FORMAT_YVU_PACKED_444 = 58, // YVU444P 8bit - 支持如下输出图片格式： HI_PIXEL_FORMAT_YUV_400 = 0, // YUV400 8bit HI_PIXEL_FORMAT_YUV_SEMIPLANAR_420 = 1, // YUV420SP NV12 8bit HI_PIXEL_FORMAT_YVU_SEMIPLANAR_420 = 2, // YUV420SP NV21 8bit HI_PIXEL_FORMAT_RGB_888 = 12, // RGB888 HI_PIXEL_FORMAT_BGR_888 = 13, // BGR888 HI_PIXEL_FORMAT_YUV_PACKED_444 = 11, // YUV444P 8bit HI_PIXEL_FORMAT_YVU_PACKED_444 = 58, // YVU444P 8bit HI_PIXEL_FORMAT_YUYV_PACKED_422 = 7, // YUV422P YUYV 8bit HI_PIXEL_FORMAT_UYVY_PACKED_422 = 8, // YUV422P UYVY 8bit HI_PIXEL_FORMAT_YVYU_PACKED_422 = 9, // YUV422P YVYU 8bit HI_PIXEL_FORMAT_VYUY_PACKED_422 = 10, // YUV422P VYUY 8bit - 关于图片宽高对齐、内存等约束，请参见1.9.16.2 约束说明。 - 在任务执行过程中，需要临时缓存，因此每个通道第一次执行本接口前需先调用 hi_mpi_vpc_set_chn_workspace接口申请内部临时缓存。 由于硬件约束，除YUV420SP NV12、YUV420SP NV21图片格式外，其它格式在输入与输出保持一致的情况下，无需调用 hi_mpi_vpc_set_chn_workspace接口申请内部临时缓存，达到节省内存的目的。 - 单张1080P（1920*1080）的图像处理超时时间建议设置10ms，其他分辨率，按照图像尺寸进行近似的等价折算，例如，4096*4096分辨率的图像建议超时时间设置为40ms。 - 由于YUV格式图像下采样约束，当输出图片格式为YUV420SP或YUV422SP格式时，在配置边界填充类型时，建议设置为“边界复制模式”，避免输出图片边缘异常数据。

版本	约束
Atlas A2训练系列产品/ Atlas 800I A2推理产品	- 输入、输出图片分辨率的取值范围为：[10*6, 4096*8192]。 - 支持如下输入图片格式： HI_PIXEL_FORMAT_YUV_400 = 0, // YUV400 8bit HI_PIXEL_FORMAT_YUV_SEMIPLANAR_420 = 1, // YUV420SP NV12 8bit HI_PIXEL_FORMAT_YVU_SEMIPLANAR_420 = 2, // YUV420SP NV21 8bit HI_PIXEL_FORMAT_RGB_888 = 12, // RGB888 HI_PIXEL_FORMAT_BGR_888 = 13, // BGR888 HI_PIXEL_FORMAT_YUV_PACKED_444 = 11, // YUV444P 8bit HI_PIXEL_FORMAT_YVU_PACKED_444 = 58, // YVU444P 8bit HI_PIXEL_FORMAT_YUV400_FP32 = 138, // YUV400 FP32, 选择该输入图片格式，则输出图片格式必须与输入保持一致 HI_PIXEL_FORMAT_RGB_888_FP32 = 1012, // RGB888 FP32, 选择该输入图片格式，则输出图片格式必须与输入保持一致 HI_PIXEL_FORMAT_BGR_888_FP32 = 1013, // BGR888 FP32, 选择该输入图片格式，则输出图片格式必须与输入保持一致 - 支持如下输出图片格式： HI_PIXEL_FORMAT_YUV_400 = 0, // YUV400 8bit HI_PIXEL_FORMAT_YUV_SEMIPLANAR_420 = 1, // YUV420SP NV12 8bit HI_PIXEL_FORMAT_YVU_SEMIPLANAR_420 = 2, // YUV420SP NV21 8bit HI_PIXEL_FORMAT_RGB_888 = 12, // RGB888 HI_PIXEL_FORMAT_BGR_888 = 13, // BGR888 HI_PIXEL_FORMAT_YUV_PACKED_444 = 11, // YUV444P 8bit HI_PIXEL_FORMAT_YVU_PACKED_444 = 58, // YVU444P 8bit HI_PIXEL_FORMAT_YUYV_PACKED_422 = 7, // YUV422P YUYV 8bit HI_PIXEL_FORMAT_UYVY_PACKED_422 = 8, // YUV422P UYVY 8bit HI_PIXEL_FORMAT_VYU_PACKED_422 = 9, // YUV422P VYU 8bit HI_PIXEL_FORMAT_VYUY_PACKED_422 = 10, // YUV422P VYUY 8bit HI_PIXEL_FORMAT_YUV400_FP32 = 138, // YUV400 FP32, 选择该输出图片格式，则输入图片格式必须与输出保持一致 HI_PIXEL_FORMAT_RGB_888_FP32 = 1012, // RGB888 FP32, 选择该输出图片格式，则输入图片格式必须与输出保持一致 HI_PIXEL_FORMAT_BGR_888_FP32 = 1013, // BGR888 FP32, 选择该输出图片格式，则输入图片格式必须与输出保持一致 - 关于图片宽高对齐、内存等约束，请参见1.9.16.2 约束说明。 - 在任务执行过程中，需要临时缓存，因此每个通道第一次执行本接口前需先调用hi_mpi_vpc_set_chn_workspace接口申请内部临时缓存。 由于硬件约束，除YUV420SP NV12、YUV420SP NV21图片格式外，其它格式在输入与输出保持一致的情况下，无需调用hi_mpi_vpc_set_chn_workspace接口申请内部临时缓存，达到节省内存的目的。 - 单张1080P (1920*1080) 的图像处理超时时间建议设置10ms，其他分辨率，按照图像尺寸进行近似的等价折算，例如，4096*4096分辨率的图像建议超时时间设置为40ms。

版本	约束
	由于YUV格式图像下采样约束，当输出图片格式为YUV420SP或YUV422SP格式时，在配置边界填充类型时，建议设置为“边界复制模式”，避免输出图片边缘异常数据。

函数原型

```
hi_s32 hi_mpi_vpc_lut_remap(hi_vpc_chn chn, hi_warp_transform_param
*transform_param, hi_remap_lut *remap_lut, hi_u32 *task_id, hi_s32 milli_sec)
```

参数说明

参数名	输入/输出	说明
chn	输入	图片处理通道号。 Atlas A2训练系列产品/Atlas 800I A2推理产品，该参数的取值范围：[0, 256)，通道总数最多256。 Atlas 200/500 A2推理产品，该参数的取值范围：[0, 128)，通道总数最多128。
transform_param	输入	Remap变换参数。
remap_lut	输入	像素位置重映射表信息。 需提前调用以下get接口获取像素位置重映射表信息： hi_mpi_vpc_get_affine_lut、或 hi_mpi_vpc_get_perspective_lut、或 hi_mpi_vpc_get_remap_lut，但本接口的缩放算法需与get接口的缩放算法保持一致，否则可能导致remap结果不符合预期。
milli_sec	输入	超时时间配置，单位是毫秒，取值范围如下： -1：阻塞方式 0：非阻塞方式 >0：超时方式，配置具体的超时时间。超时时间受操作系统影响，一般偏差在操作系统的 一个时间片内，例如，操作系统的 一个时间片为4ms，用户设置的 milli_sec 参数值为1，则实际的超时时间在1ms到5ms 范围内。在CPU负载高场景下，超时时间仍可能存在波动。
task_id	输出	任务ID的指针，用来区分任务。

返回值说明

- 0: 成功
- 非0: 失败, [1.9.21.10 VPC图像处理返回码](#)

1.9.16.32 hi_mpi_vpc_median_blur

Atlas 200/300/500 推理产品, 不支持该接口。

Atlas 训练系列产品, 不支持该接口。

Atlas 推理系列产品, 不支持该接口。

函数功能

对输入图片做中值滤波处理。

约束说明

- 输入、输出图片分辨率的取值范围为: [10*6, 4096*8192], 且输入、输出图片的分辨率需保持一致。
- 支持如下输入图片格式 (输出图片格式与输入保持一致) :

```
HI_PIXEL_FORMAT_YUV_400 = 0,           // YUV400 8bit
HI_PIXEL_FORMAT_YUV_SEMIPLANAR_420 = 1, // YUV420SP NV12 8bit
HI_PIXEL_FORMAT_YVU_SEMIPLANAR_420 = 2, // YUV420SP NV21 8bit
HI_PIXEL_FORMAT_YUV_SEMIPLANAR_422 = 3, // YUV422SP 8bit
HI_PIXEL_FORMAT_YVU_SEMIPLANAR_422 = 4, // YVU422SP 8bit
HI_PIXEL_FORMAT_YUYV_PACKED_422 = 7,    // YUV422Packed YUYV 8bit
HI_PIXEL_FORMAT_YVYU_PACKED_422 = 9,    // YUV422Packed YVYU 8bit
HI_PIXEL_FORMAT_YUV_PACKED_444 = 11,    // YUV444 Package 8bit
HI_PIXEL_FORMAT_RGB_888 = 12,           // RGB888
HI_PIXEL_FORMAT_BGR_888 = 13,           // BGR888
HI_PIXEL_FORMAT_YVU_PACKED_444 = 58,    // YVU444 Package 8bit
HI_PIXEL_FORMAT_RGB_888_PLANAR = 69,    // RGB888 Planar
HI_PIXEL_FORMAT_BGR_888_PLANAR = 70,    // BGR888 Planar
```

函数原型

[hi_s32](#) hi_mpi_vpc_median_blur([hi_vpc_chn](#) chn, const [hi_median_blur_param](#)* median_blur_param, [hi_u32](#)* task_id, [hi_s32](#) milli_sec)

参数说明

参数名	输入/输出	说明
chn	输入	图片处理通道号。 Atlas A2训练系列产品/Atlas 800I A2推理产品, 该参数的取值范围: [0, 256), 通道总数最多256。 Atlas 200/500 A2推理产品, 该参数的取值范围: [0, 128), 通道总数最多128。

参数名	输入/输出	说明
median_blur_param	输入	中值滤波信息 在 hi_median_blur_param 结构体内配置输入、输出图片信息，以及中值滤波参数。其中，中值滤波参数在 hi_median_blur_config 结构体内配置。
milli_sec	输入	超时时间配置，单位是毫秒，取值范围如下： • -1：阻塞方式 • 0：非阻塞方式 • >0：超时方式，配置具体的超时时间。超时时间受操作系统影响，一般偏差在操作系统的片内，例如，操作系统的片为4ms，用户设置的milli_sec参数值为1，则实际的超时时间在1ms到5ms范围内。在CPU负载高场景下，超时时间仍可能存在波动。
task_id	输出	任务ID的指针，用来区分任务。

返回值说明

- 0：成功
- 非0：失败，[1.9.21.10 VPC图像处理返回码](#)

1.9.16.33 hi_mpi_vpc_erode

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

函数功能

对输入图片做腐蚀处理。

约束说明

- 输入、输出图片分辨率的取值范围为：[10*6, 4096*8192]，且输入、输出图片的分辨率需保持一致。
- 支持如下输入图片格式（输出图片格式与输入保持一致）：

```
HI_PIXEL_FORMAT_YUV_400 = 0,           // YUV400 8bit
HI_PIXEL_FORMAT_YUV_SEMIPLANAR_420 = 1, // YUV420SP NV12 8bit
HI_PIXEL_FORMAT_YVU_SEMIPLANAR_420 = 2, // YUV420SP NV21 8bit
HI_PIXEL_FORMAT_YUV_SEMIPLANAR_422 = 3, // YUV422SP 8bit
HI_PIXEL_FORMAT_YVU_SEMIPLANAR_422 = 4, // YVU422SP 8bit
HI_PIXEL_FORMAT_YUYV_PACKED_422 = 7,    // YUV422Packed YUYV 8bit
HI_PIXEL_FORMAT_YVYU_PACKED_422 = 9,    // YUV422Packed YVYU 8bit
HI_PIXEL_FORMAT_YUV_PACKED_444 = 11,    // YUV444 Package 8bit
HI_PIXEL_FORMAT_RGB_888 = 12,           // RGB888
HI_PIXEL_FORMAT_BGR_888 = 13,           // BGR888
HI_PIXEL_FORMAT_YVU_PACKED_444 = 58,    // YVU444 Package 8bit
```

```
HI_PIXEL_FORMAT_RGB_888_PLANAR = 69,    // RGB888 Planar
HI_PIXEL_FORMAT_BGR_888_PLANAR = 70,    // BGR888 Planar
```

- 由于YUV格式图像下采样约束，当输出图片格式为YUV420SP或YUV422SP格式时，在配置边界填充类型时，建议设置为“边界复制模式”，避免输出图片边缘异常数据。

函数原型

hi_s32 hi_mpi_vpc_erode(**hi_vpc_chn** chn, const **hi_blur_param*** erode_param, **hi_u32*** task_id, **hi_s32** milli_sec)

参数说明

参数名	输入/输出	说明
chn	输入	图片处理通道号。 Atlas A2训练系列产品/Atlas 800I A2推理产品，该参数的取值范围：[0, 256)，通道总数最多256。 Atlas 200/500 A2推理产品，该参数的取值范围：[0, 128)，通道总数最多128。
erode_param	输入	腐蚀信息。 在 hi_blur_param 结构体内配置输入、输出图片信息，以及腐蚀参数。
milli_sec	输入	超时时间配置，单位是毫秒，取值范围如下： • -1：阻塞方式 • 0：非阻塞方式 • >0：超时方式，配置具体的超时时间。超时时间受操作系统影响，一般偏差在操作系统的片一个时间片内，例如，操作系统的片一个时间片为4ms，用户设置的milli_sec参数值为1，则实际的超时时间在1ms到5ms范围内。在CPU负载高场景下，超时时间仍可能存在波动。
task_id	输出	任务ID的指针，用来区分任务。

返回值说明

- 0：成功
- 非0：失败，[1.9.21.10 VPC图像处理返回码](#)

1.9.16.34 hi_mpi_vpc_dilate

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

函数功能

对输入图片做膨胀处理。

约束说明

- 输入、输出图片分辨率的取值范围为：[10*6, 4096*8192]，且输入、输出图片的分辨率需保持一致。
- 支持如下输入图片格式（输出图片格式与输入保持一致）：

```
HI_PIXEL_FORMAT_YUV_400 = 0,           // YUV400 8bit
HI_PIXEL_FORMAT_YUV_SEMIPLANAR_420 = 1, // YUV420SP NV12 8bit
HI_PIXEL_FORMAT_YVU_SEMIPLANAR_420 = 2, // YUV420SP NV21 8bit
HI_PIXEL_FORMAT_YUV_SEMIPLANAR_422 = 3, // YUV422SP 8bit
HI_PIXEL_FORMAT_YVU_SEMIPLANAR_422 = 4, // YVU422SP 8bit
HI_PIXEL_FORMAT_YUYV_PACKED_422 = 7,    // YUV422Packed YUYV 8bit
HI_PIXEL_FORMAT_YVYU_PACKED_422 = 9,    // YUV422Packed YVYU 8bit
HI_PIXEL_FORMAT_YUV_PACKED_444 = 11,    // YUV444 Package 8bit
HI_PIXEL_FORMAT_RGB_888 = 12,           // RGB888
HI_PIXEL_FORMAT_BGR_888 = 13,           // BGR888
HI_PIXEL_FORMAT_YVU_PACKED_444 = 58,    // YVU444 Package 8bit
HI_PIXEL_FORMAT_RGB_888_PLANAR = 69,    // RGB888 Planar
HI_PIXEL_FORMAT_BGR_888_PLANAR = 70,    // BGR888 Planar
```
- 由于YUV格式图像下采样约束，当输出图片格式为YUV420SP或YUV422SP格式时，在配置边界填充类型时，建议设置为“边界复制模式”，避免输出图片边缘异常数据。

函数原型

`hi_s32 hi_mpi_vpc_dilate(hi_vpc_chn chn, const hi_blur_param* dilate_param, hi_u32* task_id, hi_s32 milli_sec)`

参数说明

参数名	输入/输出	说明
chn	输入	图片处理通道号。 Atlas A2训练系列产品/Atlas 800I A2推理产品，该参数的取值范围：[0, 256)，通道总数最多256。 Atlas 200/500 A2推理产品，该参数的取值范围：[0, 128)，通道总数最多128。
dilate_param	输入	膨胀信息。 在hi_blur_param结构体内配置输入、输出图片信息，以及膨胀参数。

参数名	输入/输出	说明
milli_sec	输入	超时时间配置，单位是毫秒，取值范围如下： • -1：阻塞方式 • 0：非阻塞方式 • >0：超时方式，配置具体的超时时间。超时时间受操作系统影响，一般偏差在操作系统的<code>一个时间片</code>内，例如，操作系统的<code>一个时间片</code>为4ms，用户设置的<code>milli_sec</code>参数值为1，则实际的超时时间在1ms到5ms范围内。在CPU负载高场景下，超时时间仍可能存在波动。
task_id	输出	任务ID的指针，用来区分任务。

返回值说明

- 0：成功
- 非0：失败，[1.9.21.10 VPC图像处理返回码](#)

1.9.16.35 hi_mpi_vpc_blur

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

函数功能

对输入图片做均值滤波处理。

约束说明

版本	约束
Atlas 200/500 A2推理产品	- 输入、输出图片分辨率的取值范围为：[10*6, 4096*8192]，且输入、输出图片的分辨率需保持一致。 - 支持如下输入图片格式（输出图片格式与输入保持一致）： HI_PIXEL_FORMAT_YUV_400 = 0, // YUV400 8bit HI_PIXEL_FORMAT_YUV_SEMIPLANAR_420 = 1, // YUV420SP NV12 8bit HI_PIXEL_FORMAT_YVU_SEMIPLANAR_420 = 2, // YUV420SP NV21 8bit HI_PIXEL_FORMAT_YUV_SEMIPLANAR_422 = 3, // YUV422SP 8bit HI_PIXEL_FORMAT_YVU_SEMIPLANAR_422 = 4, // YVU422SP 8bit HI_PIXEL_FORMAT_YUYV_PACKED_422 = 7, // YUV422Packed YUYV 8bit HI_PIXEL_FORMAT_YVYU_PACKED_422 = 9, // YUV422Packed YVYU 8bit HI_PIXEL_FORMAT_YUV_PACKED_444 = 11, // YUV444 Package 8bit HI_PIXEL_FORMAT_RGB_888 = 12, // RGB888 HI_PIXEL_FORMAT_BGR_888 = 13, // BGR888 HI_PIXEL_FORMAT_YVU_PACKED_444 = 58, // YVU444 Package 8bit HI_PIXEL_FORMAT_RGB_888_PLANAR = 69, // RGB888 Planar HI_PIXEL_FORMAT_BGR_888_PLANAR = 70, // BGR888 Planar - 实现原理和opencv一致，浮点转定点运算部分有差异，精度不能完全对标。 - 由于YUV格式图像下采样约束，当输出图片格式为YUV420SP或YUV422SP格式时，在配置边界填充类型时，建议设置为“边界复制模式”，避免输出图片边缘异常数据。

版本	约束
Atlas A2训练系列产品/ Atlas 800I A2推理产品	- 输入、输出图片分辨率的取值范围为：[10*6, 4096*8192]，且输入、输出图片的分辨率需保持一致。 - 支持如下输入图片格式（输出图片格式与输入保持一致）： <pre>HI_PIXEL_FORMAT_YUV_400 = 0, // YUV400 8bit HI_PIXEL_FORMAT_YUV_SEMIPLANAR_420 = 1, // YUV420SP NV12 8bit HI_PIXEL_FORMAT_YVU_SEMIPLANAR_420 = 2, // YUV420SP NV21 8bit HI_PIXEL_FORMAT_YUV_SEMIPLANAR_422 = 3, // YUV422SP 8bit HI_PIXEL_FORMAT_YVU_SEMIPLANAR_422 = 4, // YVU422SP 8bit HI_PIXEL_FORMAT_YUYV_PACKED_422 = 7, // YUV422Packed YUYV 8bit HI_PIXEL_FORMAT_YVYU_PACKED_422 = 9, // YUV422Packed YVYU 8bit HI_PIXEL_FORMAT_YUV_PACKED_444 = 11, // YUV444 Package 8bit HI_PIXEL_FORMAT_RGB_888 = 12, // RGB888 HI_PIXEL_FORMAT_BGR_888 = 13, // BGR888 HI_PIXEL_FORMAT_YVU_PACKED_444 = 58, // YVU444 Package 8bit HI_PIXEL_FORMAT_RGB_888_PLANAR = 69, // RGB888 Planar HI_PIXEL_FORMAT_BGR_888_PLANAR = 70, // BGR888 Planar HI_PIXEL_FORMAT_YUV400_FP32 = 138, // YUV400 Package 每个像素用1个float32数据表示 HI_PIXEL_FORMAT_RGB_888_PLANAR_FP32 = 1006, // RGB888 Planar 每个像素用1个float32数据表示 HI_PIXEL_FORMAT_BGR_888_PLANAR_FP32 = 1007, // BGR888 Planar 每个像素用1个float32数据表示 HI_PIXEL_FORMAT_RGB_888_FP32 = 1012, // RGB888 Package, 每个像素用1个float32数据表示 HI_PIXEL_FORMAT_BGR_888_FP32 = 1013, // BGR888 Package, 每个像素用1个float32数据表示</pre> - 由于YUV格式图像下采样约束，当输出图片格式为YUV420SP或YUV422SP格式时，在配置边界填充类型时，建议设置为“边界复制模式”，避免输出图片边缘异常数据。

函数原型

hi_s32 hi_mpi_vpc_blur(**hi_vpc_chn** chn, const **hi_blur_param*** blur_param, **hi_u32*** task_id, **hi_s32** milli_sec)

参数说明

参数名	输入/输出	说明
chn	输入	图片处理通道号。 Atlas A2训练系列产品/Atlas 800I A2推理产品，该参数的取值范围：[0, 256)，通道总数最多256。 Atlas 200/500 A2推理产品，该参数的取值范围：[0, 128)，通道总数最多128。
blur_param	输入	滤波信息。 在 hi_blur_param 结构体内配置输入、输出图片信息，以及均值滤波参数。
milli_sec	输入	超时时间配置，单位是毫秒，取值范围如下： • -1：阻塞方式 • 0：非阻塞方式 • >0：超时方式，配置具体的超时时间。超时时间受操作系统影响，一般偏差在操作系统的片一个时间片内，例如，操作系统的片一个时间片为4ms，用户设置的milli_sec参数值为1，则实际的超时时间在1ms到5ms范围内。在CPU负载高场景下，超时时间仍可能存在波动。
task_id	输出	任务ID的指针，用来区分任务。

返回值说明

- 0：成功
- 非0：失败，[1.9.21.10 VPC图像处理返回码](#)

1.9.16.36 hi_mpi_vpc_gaussian_blur

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

函数功能

对输入图片做高斯滤波处理。

约束说明

版本	约束
Atlas 200/500 A2推理产品	- 输入、输出图片分辨率的取值范围为：[10*6, 4096*8192]，且输入、输出图片的分辨率需保持一致。 - 支持如下输入图片格式（输出图片格式与输入保持一致）： HI_PIXEL_FORMAT_YUV_400 = 0, // YUV400 8bit HI_PIXEL_FORMAT_YUV_SEMIPLANAR_420 = 1, // YUV420SP NV12 8bit HI_PIXEL_FORMAT_YVU_SEMIPLANAR_420 = 2, // YUV420SP NV21 8bit HI_PIXEL_FORMAT_YUV_SEMIPLANAR_422 = 3, // YUV422SP 8bit HI_PIXEL_FORMAT_YVU_SEMIPLANAR_422 = 4, // YVU422SP 8bit HI_PIXEL_FORMAT_YUYV_PACKED_422 = 7, // YUV422Packed YUYV 8bit HI_PIXEL_FORMAT_YVYU_PACKED_422 = 9, // YUV422Packed YVYU 8bit HI_PIXEL_FORMAT_YUV_PACKED_444 = 11, // YUV444 Package 8bit HI_PIXEL_FORMAT_RGB_888 = 12, // RGB888 HI_PIXEL_FORMAT_BGR_888 = 13, // BGR888 HI_PIXEL_FORMAT_YVU_PACKED_444 = 58, // YVU444 Package 8bit HI_PIXEL_FORMAT_RGB_888_PLANAR = 69, // RGB888 Planar HI_PIXEL_FORMAT_BGR_888_PLANAR = 70, // BGR888 Planar - 实现原理和opencv一致，浮点转定点运算部分有差异，精度不能完全对标。 - 由于YUV格式图像下采样约束，当输出图片格式为YUV420SP或YUV422SP格式时，在配置边界填充类型时，建议设置为“边界复制模式”，避免输出图片边缘异常数据。

版本	约束
Atlas A2训练系列产品/ Atlas 800I A2推理产品	- 输入、输出图片分辨率的取值范围为：[10*6, 4096*8192]，且输入、输出图片的分辨率需保持一致。 - 支持如下输入图片格式（输出图片格式与输入保持一致）： HI_PIXEL_FORMAT_YUV_400 = 0, // YUV400 8bit HI_PIXEL_FORMAT_YUV_SEMIPLANAR_420 = 1, // YUV420SP NV12 8bit HI_PIXEL_FORMAT_YVU_SEMIPLANAR_420 = 2, // YUV420SP NV21 8bit HI_PIXEL_FORMAT_YUV_SEMIPLANAR_422 = 3, // YUV422SP 8bit HI_PIXEL_FORMAT_YVU_SEMIPLANAR_422 = 4, // YVU422SP 8bit HI_PIXEL_FORMAT_YUYV_PACKED_422 = 7, // YUV422Packed YUYV 8bit HI_PIXEL_FORMAT_YVYU_PACKED_422 = 9, // YUV422Packed YVYU 8bit HI_PIXEL_FORMAT_YUV_PACKED_444 = 11, // YUV444 Package 8bit HI_PIXEL_FORMAT_RGB_888 = 12, // RGB888 HI_PIXEL_FORMAT_BGR_888 = 13, // BGR888 HI_PIXEL_FORMAT_YVU_PACKED_444 = 58, // YVU444 Package 8bit HI_PIXEL_FORMAT_RGB_888_PLANAR = 69, // RGB888 Planar HI_PIXEL_FORMAT_BGR_888_PLANAR = 70, // BGR888 Planar HI_PIXEL_FORMAT_YUV400_FP32 = 138, // YUV400 Package 每个像素用1个float32数据表示 HI_PIXEL_FORMAT_RGB_888_PLANAR_FP32 = 1006, // RGB888 Planar 每个像素用1个float32数据表示 HI_PIXEL_FORMAT_BGR_888_PLANAR_FP32 = 1007, // BGR888 Planar 每个像素用1个float32数据表示 HI_PIXEL_FORMAT_RGB_888_FP32 = 1012, // RGB888 Package, 每个像素用1个float32数据表示 HI_PIXEL_FORMAT_BGR_888_FP32 = 1013, // BGR888 Package, 每个像素用1个float32数据表示 - 由于YUV格式图像下采样约束，当输出图片格式为YUV420SP或YUV422SP格式时，在配置边界填充类型时，建议设置为“边界复制模式”，避免输出图片边缘异常数据。

函数原型

hi_s32 hi_mpi_vpc_gaussian_blur(hi_vpc_chn chn, const hi_gaussian_blur_param* gaussian_blur_param, hi_u32* task_id, hi_s32 milli_sec)

参数说明

参数名	输入/输出	说明
chn	输入	图片处理通道号。 Atlas A2训练系列产品/Atlas 800I A2推理产品，该参数的取值范围：[0, 256)，通道总数最多256。 Atlas 200/500 A2推理产品，该参数的取值范围：[0, 128)，通道总数最多128。
gaussian_blur_param	输入	高斯滤波信息。 在 hi_gaussian_blur_param 结构体内配置输入、输出图片信息，以及高斯滤波参数。
milli_sec	输入	超时时间配置，单位是毫秒，取值范围如下： • -1：阻塞方式 • 0：非阻塞方式 • >0：超时方式，配置具体的超时时间。超时时间受操作系统影响，一般偏差在操作系统的<code>一个时间片</code>内，例如，操作系统的<code>一个时间片</code>为4ms，用户设置的<code>milli_sec</code>参数值为1，则实际的超时时间在1ms到5ms范围内。在CPU负载高场景下，超时时间仍可能存在波动。
task_id	输出	任务ID的指针，用来区分任务。

返回值说明

- 0：成功
- 非0：失败，[1.9.21.10 VPC图像处理返回码](#)

1.9.16.37 hi_mpi_vpc_filter2d

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

函数功能

对输入图片做2D卷积滤波处理。

约束说明

版本	约束
Atlas 200/500 A2推理产品	- 输入、输出图片分辨率的取值范围为：[10*6, 4096*8192]，且输入、输出图片的分辨率需保持一致。 - 支持如下输入图片格式（输出图片格式与输入保持一致）： HI_PIXEL_FORMAT_YUV_400 = 0, // YUV400 8bit HI_PIXEL_FORMAT_YUV_SEMIPLANAR_420 = 1, // YUV420SP NV12 8bit HI_PIXEL_FORMAT_YVU_SEMIPLANAR_420 = 2, // YUV420SP NV21 8bit HI_PIXEL_FORMAT_YUV_SEMIPLANAR_422 = 3, // YUV422SP 8bit HI_PIXEL_FORMAT_YVU_SEMIPLANAR_422 = 4, // YVU422SP 8bit HI_PIXEL_FORMAT_YUYV_PACKED_422 = 7, // YUV422Packed YUYV 8bit HI_PIXEL_FORMAT_YVYU_PACKED_422 = 9, // YUV422Packed YVYU 8bit HI_PIXEL_FORMAT_YUV_PACKED_444 = 11, // YUV444 Package 8bit HI_PIXEL_FORMAT_RGB_888 = 12, // RGB888 HI_PIXEL_FORMAT_BGR_888 = 13, // BGR888 HI_PIXEL_FORMAT_YVU_PACKED_444 = 58, // YVU444 Package 8bit HI_PIXEL_FORMAT_RGB_888_PLANAR = 69, // RGB888 Planar HI_PIXEL_FORMAT_BGR_888_PLANAR = 70, // BGR888 Planar - 实现原理和opencv一致，浮点转定点运算部分有差异，精度不能完全对标。 - 由于YUV格式图像下采样约束，当输出图片格式为YUV420SP或YUV422SP格式时，在配置边界填充类型时，建议设置为“边界复制模式”，避免输出图片边缘异常数据。

版本	约束
Atlas A2训练系列产品/ Atlas 800I A2推理产品	- 输入、输出图片分辨率的取值范围为：[10*6, 4096*8192]，且输入、输出图片的分辨率需保持一致。 - 支持如下输入图片格式（输出图片格式与输入保持一致）： <pre>HI_PIXEL_FORMAT_YUV_400 = 0, // YUV400 8bit HI_PIXEL_FORMAT_YUV_SEMIPLANAR_420 = 1, // YUV420SP NV12 8bit HI_PIXEL_FORMAT_YVU_SEMIPLANAR_420 = 2, // YUV420SP NV21 8bit HI_PIXEL_FORMAT_YUV_SEMIPLANAR_422 = 3, // YUV422SP 8bit HI_PIXEL_FORMAT_YVU_SEMIPLANAR_422 = 4, // YVU422SP 8bit HI_PIXEL_FORMAT_YUYV_PACKED_422 = 7, // YUV422Packed YUYV 8bit HI_PIXEL_FORMAT_YVYU_PACKED_422 = 9, // YUV422Packed YVYU 8bit HI_PIXEL_FORMAT_YUV_PACKED_444 = 11, // YUV444 Package 8bit HI_PIXEL_FORMAT_RGB_888 = 12, // RGB888 HI_PIXEL_FORMAT_BGR_888 = 13, // BGR888 HI_PIXEL_FORMAT_YVU_PACKED_444 = 58, // YVU444 Package 8bit HI_PIXEL_FORMAT_RGB_888_PLANAR = 69, // RGB888 Planar HI_PIXEL_FORMAT_BGR_888_PLANAR = 70, // BGR888 Planar HI_PIXEL_FORMAT_YUV400_FP32 = 138, // YUV400 Package 每个像素用1个float32数据表示 HI_PIXEL_FORMAT_RGB_888_PLANAR_FP32 = 1006, // RGB888 Planar 每个像素用1个float32数据表示 HI_PIXEL_FORMAT_BGR_888_PLANAR_FP32 = 1007, // BGR888 Planar 每个像素用1个float32数据表示 HI_PIXEL_FORMAT_RGB_888_FP32 = 1012, // RGB888 Package, 每个像素用1个float32数据表示 HI_PIXEL_FORMAT_BGR_888_FP32 = 1013, // BGR888 Package, 每个像素用1个float32数据表示</pre> - 由于YUV格式图像下采样约束，当输出图片格式为YUV420SP或YUV422SP格式时，在配置边界填充类型时，建议设置为“边界复制模式”，避免输出图片边缘异常数据。

函数原型

hi_s32 hi_mpi_vpc_filter2d(hi_vpc_chn chn, const hi_filter_2d_param*
filter_2d_param, hi_u32* task_id, hi_s32 milli_sec);

参数说明

参数名	输入/输出	说明
chn	输入	图片处理通道号。 Atlas A2训练系列产品/Atlas 800I A2推理产品，该参数的取值范围：[0, 256)，通道总数最多256。 Atlas 200/500 A2推理产品，该参数的取值范围：[0, 128)，通道总数最多128。
filter_2d_param	输入	2D卷积滤波信息。 在 hi_filter_2d_param 结构体内配置输入、输出图片信息，以及卷积滤波参数。
milli_sec	输入	超时时间配置，单位是毫秒，取值范围如下： • -1：阻塞方式 • 0：非阻塞方式 • >0：超时方式，配置具体的超时时间。超时时间受操作系统影响，一般偏差在操作系统的<code>一个时间片</code>内，例如，操作系统的<code>一个时间片</code>为4ms，用户设置的<code>milli_sec</code>参数值为1，则实际的超时时间在1ms到5ms范围内。在CPU负载高场景下，超时时间仍可能存在波动。
task_id	输出	任务ID的指针，用来区分任务。

返回值说明

- 0：成功
- 非0：失败，[1.9.21.10 VPC图像处理返回码](#)

1.9.16.38 hi_mpi_vpc_rotate

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

函数功能

对输入图片做90度/180度/270度操作，输出图片宽高也要随旋转角度置换。

约束说明

版本	约束
Atlas 200/500 A2推理产品	- 输入、输出图片分辨率的取值范围为：[10*6, 4096*8192]。 - 支持如下输入图片格式（输出图片格式与输入保持一致）： HI_PIXEL_FORMAT_YUV_400 = 0, // YUV400 8bit HI_PIXEL_FORMAT_YUV_SEMIPLANAR_420 = 1, // YUV420SP NV12 8bit HI_PIXEL_FORMAT_YVU_SEMIPLANAR_420 = 2, // YUV420SP NV21 8bit HI_PIXEL_FORMAT_YUV_PACKED_444 = 11, // YUV444 Package 8bit HI_PIXEL_FORMAT_RGB_888 = 12, // RGB888 HI_PIXEL_FORMAT_BGR_888 = 13, // BGR888 HI_PIXEL_FORMAT_YVU_PACKED_444 = 58, // YVU444 Package 8bit HI_PIXEL_FORMAT_RGB_888_PLANAR = 69, // RGB888 Planar HI_PIXEL_FORMAT_BGR_888_PLANAR = 70, // BGR888 Planar
Atlas A2训练系列产品/ Atlas 800I A2推理产品	- 输入、输出图片分辨率的取值范围为：[10*6, 4096*8192]。 - 支持如下输入图片格式（输出图片格式与输入保持一致）： HI_PIXEL_FORMAT_YUV_400 = 0, // YUV400 8bit HI_PIXEL_FORMAT_YUV_PACKED_444 = 11, // YUV444 Package 8bit HI_PIXEL_FORMAT_RGB_888 = 12, // RGB888 HI_PIXEL_FORMAT_BGR_888 = 13, // BGR888 HI_PIXEL_FORMAT_YVU_PACKED_444 = 58, // YVU444 Package 8bit HI_PIXEL_FORMAT_YUV400_FP32 = 138, // YUV400 Package 每个像素用1个float32数据表示 HI_PIXEL_FORMAT_RGB_888_FP32 = 1012, // RGB888 Package, 每个像素用1个float32数据表示 HI_PIXEL_FORMAT_BGR_888_FP32 = 1013, // BGR888 Package, 每个像素用1个float32数据表示

函数原型

hi_s32 hi_mpi_vpc_rotate(**hi_vpc_chn** chn, const **hi_rotate_param*** rotate_param, **hi_u32*** task_id, **hi_s32** milli_sec)

参数说明

参数名	输入/输出	说明
chn	输入	图片处理通道号。 Atlas A2训练系列产品/Atlas 800I A2推理产品，该参数的取值范围：[0, 256)，通道总数最多256。 Atlas 200/500 A2推理产品，该参数的取值范围：[0, 128)，通道总数最多128。
rotate_param	输入	旋转信息。 在 hi_rotate_param 结构体内配置输入、输出图片信息，以及旋转参数。
milli_sec	输入	超时时间配置，单位是毫秒，取值范围如下： • -1：阻塞方式 • 0：非阻塞方式 • >0：超时方式，配置具体的超时时间。超时时间受操作系统影响，一般偏差在操作系统的片一个时间片内，例如，操作系统的片一个时间片为4ms，用户设置的milli_sec参数值为1，则实际的超时时间在1ms到5ms范围内。在CPU负载高场景下，超时时间仍可能存在波动。
task_id	输出	任务ID的指针，用来区分任务。

返回值说明

- 0：成功
- 非0：失败，[1.9.21.10 VPC图像处理返回码](#)

1.9.16.39 hi_mpi_vpc_draw_mosaic

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

函数功能

对输入图片的部分区域做马赛克处理。

约束说明

- 输入、输出图片分辨率的取值范围为：[10*6, 4096*8192]，且输入、输出图片的分辨率需保持一致。
- 支持如下输入图片格式（输出图片格式与输入保持一致）：

```
HI_PIXEL_FORMAT_YUV_400 = 0,           // YUV400 8bit
HI_PIXEL_FORMAT_YUV_SEMIPLANAR_420 = 1, // YUV420SP NV12 8bit
HI_PIXEL_FORMAT_YVU_SEMIPLANAR_420 = 2, // YUV420SP NV21 8bit
HI_PIXEL_FORMAT_YUV_SEMIPLANAR_422 = 3, // YUV422SP 8bit
HI_PIXEL_FORMAT_YVU_SEMIPLANAR_422 = 4, // YVU422SP 8bit
HI_PIXEL_FORMAT_YUYV_PACKED_422 = 7,    // YUV422Packed YUYV 8bit
HI_PIXEL_FORMAT_YVYU_PACKED_422 = 9,    // YUV422Packed YVYU 8bit
HI_PIXEL_FORMAT_YUV_PACKED_444 = 11,    // YUV444 Package 8bit
HI_PIXEL_FORMAT_RGB_888 = 12,           // RGB888
HI_PIXEL_FORMAT_BGR_888 = 13,           // BGR888
HI_PIXEL_FORMAT_YVU_PACKED_444 = 58,    // YVU444 Package 8bit
HI_PIXEL_FORMAT_RGB_888_PLANAR = 69,    // RGB888 Planar
HI_PIXEL_FORMAT_BGR_888_PLANAR = 70,    // BGR888 Planar
```

函数原型

```
hi_s32 hi_mpi_vpc_draw_mosaic(hi_vpc_chn chn, const hi_mosaic_param*
mosaic_param, hi_u32* task_id, hi_s32 milli_sec)
```

参数说明

参数名	输入/输出	说明
chn	输入	图片处理通道号。 Atlas 200/500 A2推理产品，该参数的取值范围：[0, 128)，通道总数最多128。
mosaic_param	输入	马赛克信息。 在hi_mosaic_param结构体内配置输入、输出图片信息，以及马赛克参数。
milli_sec	输入	超时时间配置，单位是毫秒，取值范围如下： -1：阻塞方式 0：非阻塞方式 >0：超时方式，配置具体的超时时间。超时时间受操作系统影响，一般偏差在操作系统的—个时间片内，例如，操作系统的—个时间片为4ms，用户设置的milli_sec参数值为1，则实际的超时时间在1ms到5ms范围内。在CPU负载高场景下，超时时间仍可能存在波动。
task_id	输出	任务ID的指针，用来区分任务。

返回值说明

- 0：成功
- 非0：失败，[1.9.21.10 VPC图像处理返回码](#)

1.9.16.40 hi_mpi_vpc_draw_cover

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

函数功能

对输入图片的部分区域做覆盖处理。

约束说明

- 输入、输出图片分辨率的取值范围为：[10*6, 4096*8192]，且输入、输出图片的分辨率需保持一致。
- 支持如下输入图片格式（输出图片格式与输入保持一致）：

```
HI_PIXEL_FORMAT_YUV_400 = 0,           // YUV400 8bit
HI_PIXEL_FORMAT_YUV_SEMIPLANAR_420 = 1, // YUV420SP NV12 8bit
HI_PIXEL_FORMAT_YVU_SEMIPLANAR_420 = 2, // YUV420SP NV21 8bit
HI_PIXEL_FORMAT_YUV_SEMIPLANAR_422 = 3, // YUV422SP 8bit
HI_PIXEL_FORMAT_YVU_SEMIPLANAR_422 = 4, // YVU422SP 8bit
HI_PIXEL_FORMAT_YUYV_PACKED_422 = 7,    // YUV422Packed YUYV 8bit
HI_PIXEL_FORMAT_YVYU_PACKED_422 = 9,    // YUV422Packed YVYU 8bit
HI_PIXEL_FORMAT_YUV_PACKED_444 = 11,    // YUV444 Package 8bit
HI_PIXEL_FORMAT_RGB_888 = 12,           // RGB888
HI_PIXEL_FORMAT_BGR_888 = 13,           // BGR888
HI_PIXEL_FORMAT_YVU_PACKED_444 = 58,    // YVU444 Package 8bit
HI_PIXEL_FORMAT_RGB_888_PLANAR = 69,    // RGB888 Planar
HI_PIXEL_FORMAT_BGR_888_PLANAR = 70,    // BGR888 Planar
```

函数原型

hi_s32 hi_mpi_vpc_draw_cover(**hi_vpc_chn** chn, const **hi_cover_param*** cover_param, **hi_u32*** task_id, **hi_s32** milli_sec)

参数说明

参数名	输入/输出	说明
chn	输入	图片处理通道号。 Atlas 200/500 A2推理产品，该参数的取值范围：[0, 128)，通道总数最多128。
cover_param	输入	覆盖信息。 在 hi_cover_param 结构体内配置输入、输出图片信息，以及覆盖参数。
milli_sec	输入	超时时间配置，单位是毫秒，取值范围如下： -1：阻塞方式 0：非阻塞方式 >0：超时方式，配置具体的超时时间。超时时间受操作系统影响，一般偏差在操作系统的<code>一个时间片</code>内，例如，操作系统的<code>一个时间片</code>为4ms，用户设置的<code>milli_sec</code>参数值为1，则实际的超时时间在1ms到5ms范围内。在CPU负载高场景下，超时时间仍可能存在波动。

参数名	输入/输出	说明
task_id	输出	任务ID的指针，用来区分任务。

返回值说明

- 0：成功
- 非0：失败，[1.9.21.10 VPC图像处理返回码](#)

1.9.16.41 hi_mpi_vpc_draw_line

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

函数功能

对输入图片做画线处理。

约束说明

- 输入、输出图片分辨率的取值范围为：[10*6, 4096*8192]，且输入、输出图片的分辨率需保持一致。
- 支持如下输入图片格式（输出图片格式与输入保持一致）：

```
HI_PIXEL_FORMAT_YUV_400 = 0,           // YUV400 8bit
HI_PIXEL_FORMAT_YUV_SEMIPLANAR_420 = 1, // YUV420SP NV12 8bit
HI_PIXEL_FORMAT_YVU_SEMIPLANAR_420 = 2, // YUV420SP NV21 8bit
HI_PIXEL_FORMAT_YUV_SEMIPLANAR_422 = 3, // YUV422SP 8bit
HI_PIXEL_FORMAT_YVU_SEMIPLANAR_422 = 4, // YVU422SP 8bit
HI_PIXEL_FORMAT_YUYV_PACKED_422 = 7,    // YUV422Packed YUYV 8bit
HI_PIXEL_FORMAT_YVYU_PACKED_422 = 9,    // YUV422Packed YVYU 8bit
HI_PIXEL_FORMAT_YUV_PACKED_444 = 11,    // YUV444 Package 8bit
HI_PIXEL_FORMAT_RGB_888 = 12,           // RGB888
HI_PIXEL_FORMAT_BGR_888 = 13,           // BGR888
HI_PIXEL_FORMAT_YVU_PACKED_444 = 58,    // YVU444 Package 8bit
HI_PIXEL_FORMAT_RGB_888_PLANAR = 69,    // RGB888 Planar
HI_PIXEL_FORMAT_BGR_888_PLANAR = 70,    // BGR888 Planar
```

函数原型

[hi_s32](#) hi_mpi_vpc_draw_line([hi_vpc_chn](#) chn, const [hi_line_param](#)*
line_param, [hi_u32](#)* task_id, [hi_s32](#) milli_sec)

参数说明

参数名	输入/输出	说明
chn	输入	图片处理通道号。 Atlas 200/500 A2推理产品，该参数的取值范围：[0, 128)，通道总数最多128。
line_param	输入	画线信息。 在 <code>hi_line_param</code> 结构体内配置输入、输出图片信息，以及画线参数。
milli_sec	输入	超时时间配置，单位是毫秒，取值范围如下： -1：阻塞方式 0：非阻塞方式 >0：超时方式，配置具体的超时时间。超时时间受操作系统影响，一般偏差在操作系统的<code>一个时间片</code>内，例如，操作系统的<code>一个时间片</code>为4ms，用户设置的<code>milli_sec</code>参数值为1，则实际的超时时间在1ms到5ms范围内。在CPU负载高场景下，超时时间仍可能存在波动。
task_id	输出	任务ID的指针，用来区分任务。

返回值说明

- 0：成功
- 非0：失败，[1.9.21.10 VPC图像处理返回码](#)

1.9.16.42 hi_mpi_vpc_draw_osd

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

函数功能

对输入图片做叠加处理，可用于添加水印的场景。

约束说明

- 输入、输出图片分辨率的取值范围为：[10*6, 4096*8192]，且输入、输出图片的分辨率需保持一致。
- 支持如下输入图片格式（输出图片格式与输入保持一致）：

HI_PIXEL_FORMAT_YUV_400 = 0, // YUV400 8bit
HI_PIXEL_FORMAT_YUV_SEMIPLANAR_420 = 1, // YUV420SP NV12 8bit
HI_PIXEL_FORMAT_YVU_SEMIPLANAR_420 = 2, // YUV420SP NV21 8bit


```
HI_PIXEL_FORMAT_YUV_SEMIPLANAR_422 = 3, // YUV422SP 8bit
HI_PIXEL_FORMAT_YVU_SEMIPLANAR_422 = 4, // YVU422SP 8bit
HI_PIXEL_FORMAT_YUYV_PACKED_422 = 7,    // YUV422Packed YUYV 8bit
HI_PIXEL_FORMAT_YVYU_PACKED_422 = 9,    // YUV422Packed YVYU 8bit
HI_PIXEL_FORMAT_YUV_PACKED_444 = 11,    // YUV444 Package 8bit
HI_PIXEL_FORMAT_RGB_888 = 12,           // RGB888
HI_PIXEL_FORMAT_BGR_888 = 13,           // BGR888
HI_PIXEL_FORMAT_YVU_PACKED_444 = 58,    // YVU444 Package 8bit
HI_PIXEL_FORMAT_RGB_888_PLANAR = 69,    // RGB888 Planar
HI_PIXEL_FORMAT_BGR_888_PLANAR = 70,    // BGR888 Planar
```

函数原型

```
hi_s32 hi_mpi_vpc_draw_osd(hi_vpc_chn chn, hi_osd_param* osd_param,
hi_u32* task_id, hi_s32 milli_sec)
```

参数说明

参数名	输入/输出	说明
chn	输入	图片处理通道号。 Atlas 200/500 A2推理产品，该参数的取值范围：[0, 128)，通道总数最多128。
osd_param	输入	叠加信息。 在hi_osd_param结构体内配置输入、输出图片信息，以及叠加参数。
milli_sec	输入	超时时间配置，单位是毫秒，取值范围如下： -1：阻塞方式 0：非阻塞方式 >0：超时方式，配置具体的超时时间。超时时间受操作系统影响，一般偏差在操作系统的 一个时间片内，例如，操作系统的 一个时间片为4ms，用户设置的 milli_sec参数值为1，则实际的超时时间在1ms到5ms 范围内。在CPU负载高场景下，超时时间仍可能存在波动。
task_id	输出	任务ID的指针，用来区分任务。

返回值说明

- 0：成功
- 非0：失败，1.9.21.10 VPC图像处理返回码

1.9.16.43 hi_mpi_vpc_get_affine_transform

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

函数功能

获取仿射变换需要的变换矩阵。

函数原型

```
hi_s32 hi_mpi_vpc_get_affine_transform(hi_point_pair_info *point_pair_info,  
hi_transform_matrix *matrix)
```

参数说明

参数名	输入/输出	说明
point_pair_info	输入	三对输入、输出图片中的像素点坐标信息，每对中分别包含一个输入图片像素点坐标、一个输出图片像素点坐标。
matrix	输出	仿射变换需要的变换矩阵。

返回值说明

- 0：成功
- 非0：失败，[1.9.21.10 VPC图像处理返回码](#)

1.9.16.44 hi_mpi_vpc_get_rotation_matrix

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

函数功能

获取仿射变换需要的旋转矩阵。

函数原型

```
hi_s32 hi_mpi_vpc_get_rotation_matrix(hi_float_point center_point, hi_double  
angle, hi_double scale, hi_transform_matrix *matrix)
```

参数说明

参数名	输入/输出	说明
center_point	输入	输入图像的旋转中心点。
angle	输入	旋转角度，正值表示逆时针旋转。

参数名	输入/输出	说明
scale	输入	缩放参数。 scale参数值大于1表示放大图片，参数值小于1表示缩小图片。
matrix	输出	仿射变换的旋转矩阵。

返回值说明

- 0：成功
- 非0：失败，[1.9.21.10 VPC图像处理返回码](#)

1.9.16.45 hi_mpi_vpc_warp_affine

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

函数功能

对输入图片做仿射变换。

约束说明

- 输入、输出图片分辨率的取值范围为：[10*6, 4096*8192]。
- 关于输入、输出图片格式的要求如下（关于图片宽高对齐、内存等约束，请参见[1.9.16.2 约束说明](#)）：

版本	约束
Atlas 200/500 A2推理产品	- 支持如下输入图片格式： HI_PIXEL_FORMAT_YUV_400 = 0, // YUV400 8bit HI_PIXEL_FORMAT_YUV_SEMIPLANAR_420 = 1, // YUV420SP NV12 8bit HI_PIXEL_FORMAT_YVU_SEMIPLANAR_420 = 2, // YUV420SP NV21 8bit HI_PIXEL_FORMAT_RGB_888 = 12, // RGB888 HI_PIXEL_FORMAT_BGR_888 = 13, // BGR888 HI_PIXEL_FORMAT_YUV_PACKED_444 = 11, // YUV444P 8bit HI_PIXEL_FORMAT_YVU_PACKED_444 = 58, // YVU444P 8bit - 支持如下输出图片格式： HI_PIXEL_FORMAT_YUV_400 = 0, // YUV400 8bit HI_PIXEL_FORMAT_YUV_SEMIPLANAR_420 = 1, // YUV420SP NV12 8bit HI_PIXEL_FORMAT_YVU_SEMIPLANAR_420 = 2, // YUV420SP NV21 8bit HI_PIXEL_FORMAT_RGB_888 = 12, // RGB888 HI_PIXEL_FORMAT_BGR_888 = 13, // BGR888 HI_PIXEL_FORMAT_YUV_PACKED_444 = 11, // YUV444P 8bit HI_PIXEL_FORMAT_YVU_PACKED_444 = 58, // YVU444P 8bit HI_PIXEL_FORMAT_YUYV_PACKED_422 = 7, // YUV422P YUYV 8bit HI_PIXEL_FORMAT_UYVY_PACKED_422 = 8, // YUV422P UYVY 8bit HI_PIXEL_FORMAT_YVYU_PACKED_422 = 9, // YUV422P YVYU 8bit HI_PIXEL_FORMAT_VYUY_PACKED_422 = 10, // YUV422P VYUY 8bit

版本	约束
Atlas A2训练系列产品/ Atlas 800I A2推理产品	- 支持如下输入图片格式： HI_PIXEL_FORMAT_YUV_400 = 0, // YUV400 8bit HI_PIXEL_FORMAT_YUV_SEMIPLANAR_420 = 1, // YUV420SP NV12 8bit HI_PIXEL_FORMAT_YVU_SEMIPLANAR_420 = 2, // YUV420SP NV21 8bit HI_PIXEL_FORMAT_RGB_888 = 12, // RGB888 HI_PIXEL_FORMAT_BGR_888 = 13, // BGR888 HI_PIXEL_FORMAT_YUV_PACKED_444 = 11, // YUV444P 8bit HI_PIXEL_FORMAT_YVU_PACKED_444 = 58, // YVU444P 8bit HI_PIXEL_FORMAT_YUV400_FP32 = 138, // YUV400 FP32, 选择该输入图片格式，则输出图片格式必须与输入保持一致 HI_PIXEL_FORMAT_RGB_888_FP32 = 1012, // RGB888 FP32, 选择该输入图片格式，则输出图片格式必须与输入保持一致 HI_PIXEL_FORMAT_BGR_888_FP32 = 1013, // BGR888 FP32, 选择该输入图片格式，则输出图片格式必须与输入保持一致 - 支持如下输出图片格式： HI_PIXEL_FORMAT_YUV_400 = 0, // YUV400 8bit HI_PIXEL_FORMAT_YUV_SEMIPLANAR_420 = 1, // YUV420SP NV12 8bit HI_PIXEL_FORMAT_YVU_SEMIPLANAR_420 = 2, // YUV420SP NV21 8bit HI_PIXEL_FORMAT_RGB_888 = 12, // RGB888 HI_PIXEL_FORMAT_BGR_888 = 13, // BGR888 HI_PIXEL_FORMAT_YUV_PACKED_444 = 11, // YUV444P 8bit HI_PIXEL_FORMAT_YVU_PACKED_444 = 58, // YVU444P 8bit HI_PIXEL_FORMAT_YUYV_PACKED_422 = 7, // YUV422P YUYV 8bit HI_PIXEL_FORMAT_UYVY_PACKED_422 = 8, // YUV422P UYVY 8bit HI_PIXEL_FORMAT_YVYU_PACKED_422 = 9, // YUV422P YVYU 8bit HI_PIXEL_FORMAT_VYUY_PACKED_422 = 10, // YUV422P VYUY 8bit HI_PIXEL_FORMAT_YUV400_FP32 = 138, // YUV400 FP32, 选择该输出图片格式，则输入图片格式必须与输出保持一致 HI_PIXEL_FORMAT_RGB_888_FP32 = 1012, // RGB888 FP32, 选择该输出图片格式，则输入图片格式必须与输出保持一致 HI_PIXEL_FORMAT_BGR_888_FP32 = 1013, // BGR888 FP32, 选择该输出图片格式，则输入图片格式必须与输出保持一致

- 在任务执行过程中，需要临时缓存，因此每个通道第一次执行本接口前需先调用 [hi_mpi_vpc_set_chn_workspace](#)接口申请内部临时缓存。
- 由于YUV格式图像下采样约束，当输出图片格式为YUV420SP或YUV422SP格式时，在配置边界填充类型时，建议设置为“边界复制模式”，避免输出图片边缘异常数据。

函数原型

```
hi_s32 hi_mpi_vpc_warp_affine(hi_vpc_chn chn, hi_transform_matrix *matrix,
hi_warp_transform_param *transform_param, hi_u32 *task_id, hi_s32
milli_sec)
```

参数说明

参数名	输入/输出	说明
chn	输入	图片处理通道号。 Atlas A2训练系列产品/Atlas 800I A2推理产品，该参数的取值范围：[0, 256)，通道总数最多256。 Atlas 200/500 A2推理产品，该参数的取值范围：[0, 128)，通道总数最多128。
matrix	输入	使用 hi_mpi_vpc_get_affine_transform 或者 hi_mpi_vpc_get_rotation_matrix 接口获取的仿射变换矩阵。
transform_param	输入	仿射变换参数，在 hi_warp_transform_param 结构体内配置输入、输出图片信息。
milli_sec	输入	超时时间配置，单位是毫秒，取值范围如下： • -1：阻塞方式 • 0：非阻塞方式 • >0：超时方式，配置具体的超时时间。超时时间受操作系统影响，一般偏差在操作系统的 一个时间片内，例如，操作系统的 一个时间片为4ms，用户设置的 milli_sec参数值为1，则实际的超时时间在1ms到5ms 范围内。在CPU负载高场景下，超时时间仍可能存在波动。
task_id	输出	任务ID的指针，用来区分任务。

返回值说明

- 0：成功
- 非0：失败，[1.9.21.10 VPC图像处理返回码](#)

1.9.16.46 hi_mpi_vpc_flip

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

函数功能

对输入图片做翻转处理，支持水平方向翻转、竖直方向翻转和水平竖直方向同时翻转。

约束说明

- 输入、输出图片分辨率的取值范围为：[10*6, 4096*8192]，且输入、输出图片的分辨率需保持一致。

- 支持如下输入图片格式：(输出格式需与输入保持一致)

```
HI_PIXEL_FORMAT_YUV_400 = 0,           // YUV400 8bit
HI_PIXEL_FORMAT_YUV_SEMIPLANAR_420 = 1, // YUV420SP NV12 8bit
HI_PIXEL_FORMAT_YVU_SEMIPLANAR_420 = 2, // YUV420SP NV21 8bit
HI_PIXEL_FORMAT_YUV_SEMIPLANAR_422 = 3, // YUV422SP 8bit
HI_PIXEL_FORMAT_YVU_SEMIPLANAR_422 = 4, // YVU422SP 8bit
HI_PIXEL_FORMAT_YUYV_PACKED_422 = 7,    // YUV422Packed YUYV 8bit
HI_PIXEL_FORMAT_UYVY_PACKED_422 = 8,    // YUV422Packed UYVY 8bit
HI_PIXEL_FORMAT_YVYU_PACKED_422 = 9,    // YUV422Packed YVYU 8bit
HI_PIXEL_FORMAT_VYUY_PACKED_422 = 10,   // YUV422Packed VYUY 8bit
HI_PIXEL_FORMAT_YUV_PACKED_444 = 11,    // YUV444 Package 8bit
HI_PIXEL_FORMAT_RGB_888 = 12,           // RGB888
HI_PIXEL_FORMAT_BGR_888 = 13,           // BGR888
HI_PIXEL_FORMAT_ARGB_8888 = 14,        // ARGB_8888
HI_PIXEL_FORMAT_ABGR_8888 = 15,        // ABGR_8888
HI_PIXEL_FORMAT_RGBA_8888 = 16,         // RGBA_8888
HI_PIXEL_FORMAT_BGRA_8888 = 17,        // BGRA_8888
HI_PIXEL_FORMAT_YVU_PLANAR_444 = 22,   // YVU444 Planar 8bit
HI_PIXEL_FORMAT_YVU_PACKED_444 = 58,   // YVU444 Package 8bit
HI_PIXEL_FORMAT_RGB_888_PLANAR = 69,   // RGB888 Planar
HI_PIXEL_FORMAT_BGR_888_PLANAR = 70,   // BGR888 Planar
```

函数原型

hi_s32 hi_mpi_vpc_flip(**hi_vpc_chn** chn, **hi_flip_param*** flip_param, **hi_u32*** task_id, **hi_s32** milli_sec)

参数说明

参数名	输入/输出	说明
chn	输入	图片处理通道号。 Atlas A2训练系列产品/Atlas 800I A2推理产品，该参数的取值范围：[0, 256)，通道总数最多256。 Atlas 200/500 A2推理产品，该参数的取值范围：[0, 128)，通道总数最多128。
flip_param	输入	翻转信息。 在 hi_flip_param 结构体内配置输入、输出图片信息以及翻转模式。

参数名	输入/输出	说明
milli_sec	输入	超时时间配置，单位是毫秒，取值范围如下： • -1：阻塞方式 • 0：非阻塞方式 • >0：超时方式，配置具体的超时时间。超时时间受操作系统影响，一般偏差在操作系统的—个时间片内，例如，操作系统的—个时间片为4ms，用户设置的milli_sec参数值为1，则实际的超时时间在1ms到5ms范围内。在CPU负载高场景下，超时时间仍可能存在波动。
task_id	输出	任务ID的指针，用来区分任务。

返回值说明

- 0：成功
- 非0：失败，[1.9.21.10 VPC图像处理返回码](#)

1.9.16.47 hi_mpi_vpc_set_chn_opt_attr

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

Atlas 200/500 A2推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

设置通道可选属性，目前仅支持设置通道优先级。

约束说明

- 针对同一通道，本接口只允许在创建通道后调用一次，对同一通道重复调用将会返回失败；
- 通道优先级只能将通道设置为高优先级，不调用本接口则默认为低优先级；针对不同通道，如果都设置为高优先级，则这些通道的任务按下发顺序调用。
- 高优先级通道中的任务会被优先调度，但不会影响已下发的低优先级通道的任务；
建议在高优先级通道中只下发需要处理时延敏感的关键任务，避免高优先级通道中的任务长时间占用VPC资源而导致低优先级通道中任务一直排队等待。
- VDEC任务中也会调用VPC资源，其默认使用低优先级，设置VPC的通道优先级也会对VDEC任务中的VPC资源调度产生影响；

函数原型

```
hi_s32 hi_mpi_vpc_set_chn_opt_attr(hi_vpc_chn chn, const hi_opt_attr*  
opt_attr)
```

参数说明

参数名	输入/输出	说明
chn	输入	图片处理通道号。
opt_attr	输入	通道可选属性信息。

返回值说明

- 0：成功
- 非0：失败，[1.9.21.10 VPC图像处理返回码](#)

1.9.17 VDEC 视频解码功能/JPEGD 图片解码功能

1.9.17.1 JPEGD 功能及约束说明

Atlas 200/300/500 推理产品上，不支持本章中的JPEGD功能。

Atlas 训练系列产品上，不支持本章中的JPEGD功能。

功能说明

JPEGD (JPEG Decoder) 实现.jpg、.jpeg、.JPG、.JPEG图片文件的解码。

- JPEGD在解码图片时，支持对图片进行旋转。**
如果输入图片的码流中包含Orientation信息（代表捕获图像时摄像机相对于场景的方向），则JPEGD在解码时会解析Orientation信息，将图片进行90度、180度、270度或镜像旋转。旋转后输出图片的宽stride、高stride、输出内存仍需满足[图片格式、宽高对齐、内存约束](#)中的要求。如果输入图片的码流异常，导致JPEGD解码时无法读取Orientation信息，则不能实现图片旋转的功能。

针对以下产品版本，旋转图片时存在约束：

产品版本	约束
Atlas 200/500 A2推理产品 Atlas A2训练系列产品/Atlas 800I A2推理产品	JPEGD 422格式 旋转图片，若原图高为奇数，并且旋转方式为宽高对换，则旋转后宽存在黑边，建议用户将宽向下对齐到偶数使用（即去除黑边），比如原图为200*101，若旋转后为101*200，则建议用户实际使用区域为100*200。 JPEGD 440格式 旋转图片，若原图宽为奇数，并且旋转方式为宽高对换，则旋转后高存在黑边，建议用户将高向下对齐到偶数使用（即去除黑边），比如原图为201*101，若旋转后为100*201，则建议用户实际使用区域为100*200。

- **JPEGD在解码图片时，支持按源图片格式解码。**

源图片格式解码是指解码前后图片的编码格式保持一致，例如解码前输入图片为jpeg(440)，解码后输出图片为YUV440SP V在前U在后或YUV440SP U在前V在后。

使用源图片格式解码，有以下方式：

- 在调用JPEGD解码接口时，直接将输出图片格式配置为HI_PIXEL_FORMAT_UNKNOWN，输出格式默认按源图片格式输出、且是V在前U在后的Semi-Planar格式。例如，JPEG源图片格式为jpeg(440)，输出图片格式配置为HI_PIXEL_FORMAT_UNKNOWN，JPEGD解码后，实际输出图片格式为YUV440SP V在前U在后。

此种方式，由于不知道输出图片格式，因此需要用户申请尽量大的内存或调用[hi_mpi_dvpp_get_image_info](#)接口获取解码输出内存大小，防止内存不够，无法存放输出图片。

- 先调用[hi_mpi_dvpp_get_image_info](#)接口根据传入的jpeg源图片，获取按源图解码时的输出图片的宽、高、宽stride、高stride、解码输出内存大小、图片格式等信息后，再调用JPEGD解码接口，使用通过[hi_mpi_dvpp_get_image_info](#)接口获取的图片格式来设置输出图片格式。

说明

JPEGD解码后的输出图片，如果要直接作为模型推理的输入，建议将输出图片格式配置为HI_PIXEL_FORMAT_UNKNOWN，这时JPEGD使用源图片格式解码（但这里要确保解码后的图片格式模型是支持的），保证模型推理的精度。

JPEGD解码后的输出图片，如果直接作为VPC的输入，该场景下若使用源图片格式解码时，则需要关注解码后的输出图片格式VPC是否支持（VPC输入图片的格式请参见[1.9.16.2 约束说明](#)），如果VPC不支持，则用户需按VPC支持的情况指定JPEGD的输出图片格式。

图片分辨率约束

- 输入图片分辨率

版本	分辨率范围
Atlas 推理系列产品	最大分辨率：8192*8192，最小分辨率：32*32。
Atlas A2训练系列产品/Atlas 800I A2推理产品 Atlas 200/500 A2推理产品	最大分辨率：16384*16384，最小分辨率：32*32。

- 输出图片分辨率

JPEGD只对图片解码，不会改变图片分辨率，因此输出与输入的图片分辨率保持一致。

内存申请/释放接口说明

输入内存的大小就是指实际的输入图片所占用的大小，输出内存的大小参见[表1-63](#)中的计算公式。

版本	内存申请/释放接口
Atlas 推理系列产品	JPEGD图片解码时，调用 hi_mpi_dvpp_malloc \ hi_mpi_dvpp_free 接口申请\释放Device上的输入、输出内存，这部分内存的生命周期由用户自行管理。
Atlas A2训练系列产品/Atlas 800I A2推理产品 Atlas 200/500 A2推理产品	JPEGD图片解码时，支持以下 两种 内存申请接口： - 调用aclrtMalloc\aclrtFree接口申请\释放Device上的输入、输出内存，这部分内存的生命周期由用户自行管理。 - 调用hi_mpi_dvpp_malloc\hi_mpi_dvpp_free接口申请\释放Device上的输入、输出内存，这部分内存的生命周期由用户自行管理。 注意： 调用 hi_mpi_dvpp_malloc 接口申请的内存为媒体数据处理的专用内存，但专用内存的地址空间有限，若关注内存规划或内存资源有限时，建议调用 aclrtMalloc 接口申请内存。

图片格式、宽高对齐、内存约束

实现JPEGD图片解码功能时，仅支持Huffman编码，压缩前的原图像色彩空间为YUV，像素的各分量比例为4:4:4或4:2:2或4:2:0或4:0:0或4:4:0，不支持算术编码、不支持渐进JPEG格式、不支持JPEG2000格式。

说明

输出图片格式的定义请参见[hi_pixel_format](#)，宽stride、高stride等概念请参见[1.9.2 基本概念](#)。

表 1-63 图片格式、宽高对齐、内存大小约束

输入图片格式 (YUV 分量比例)	输出图片格式	输出图片宽、高对齐要求	输出图片宽stride、高stride、内存大小要求
jpeg(444)	YVU444SP 8bit	无对齐要求	宽stride为宽64对齐后的值。 高stride为高16对齐后的值。 内存大小 (单位Byte) ≥ 宽stride * 高stride * 3
	YUV444SP 8bit		
	YUV420SP NV12 8bit	宽2对齐 高2对齐	宽stride为宽64对齐后的值。 高stride为高16对齐后的值。 内存大小 (单位Byte) ≥ 宽stride * 高stride * 3/2
	YUV420SP NV21 8bit		
jpeg(422)	YVU422SP 8bit	宽2对齐 高无对齐要求	宽stride为宽64对齐后的值。 高stride为高16对齐后的值。 内存大小 (单位Byte) ≥ 宽stride * 高stride * 2
	YUV422SP 8bit		

输入图片格式 (YUV 分量比例)	输出图片格式	输出图片宽、高对齐要求	输出图片宽stride、高stride、内存大小要求
	YUV420SP NV12 8bit	宽2对齐 高2对齐	宽stride为宽64对齐后的值。 高stride为高16对齐后的值。 内存大小 (单位Byte) ≥ 宽stride * 高stride * 3/2
	YUV420SP NV21 8bit		
jpeg(420)	YUV420SP NV12 8bit	宽2对齐 高2对齐	宽stride为宽64对齐后的值。 高stride为高16对齐后的值。 内存大小 (单位Byte) ≥ 宽stride * 高stride * 3/2
	YUV420SP NV21 8bit		
jpeg(400)	YUV420SP NV12 8bit	宽2对齐 高2对齐	宽stride为宽64对齐后的值。 高stride为高16对齐后的值。 内存大小 (单位Byte) ≥ 宽stride * 高stride * 3/2
	YUV420SP NV21 8bit		
	YUV400 8bit	无对齐要求	宽stride为宽64对齐后的值。 高stride为高16对齐后的值。 内存大小 (单位Byte) ≥ 宽stride * 高stride
jpeg(440)	YVU440SP 8bit	宽无对齐要求 高2对齐	宽stride为宽64对齐后的值。 高stride为高16对齐后的值。 内存大小 (单位Byte) ≥ 宽stride * 高stride * 2
	YUV440SP 8bit		
	YUV420SP NV12 8bit	宽2对齐 高2对齐	宽stride为宽64对齐后的值。 高stride为高16对齐后的值。 内存大小 (单位Byte) ≥ 宽stride * 高stride * 3/2
	YUV420SP NV21 8bit		

软、硬件约束

- 硬件约束：
 - 最多支持4张Huffman表，其中包括2 张DC (Direct Current) (直流) 表和 2 张AC (Alternating Current) (交流) 表；
 - 最多支持3张量化表；
 - 只支持8bit采样精度；
 - 只支持对顺序式编码的图片进行解码；
 - 只支持基于DCT (Discrete Cosine Transform) 变换的JPEG 格式解码；
 - 只支持一个SOS (Start of Scan) 标志的图片解码。
- 软件约束：
 - 支持3个SOS标志的图片解码；

- 支持mcu (Minimum Coded Unit) 数据不足的异常图片解码。

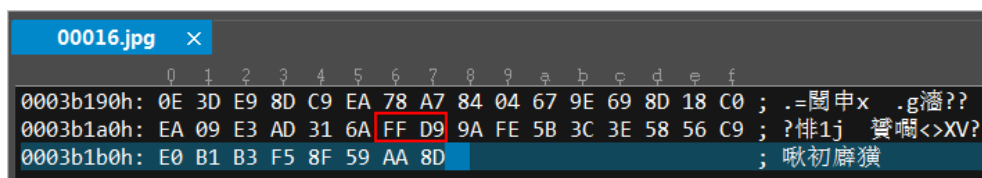
精度相关约束

JPEG+VPC串联使用时，由于JPEG解码后的输出图片的宽stride*高stride有64*16对齐的约束，因此解码后的输出图片的宽、高有一些补边的无效数据，所以在执行VPC功能时（例如缩放时），需在输入图片的hi_vpc_pic_info.picture_width、hi_vpc_pic_info.picture_height参数处正确设置输入图片的原图宽高，这样VPC在缩放图片前会先根据原图宽高自行抠图，目的是去除无效数据对图像精度的影响。

其它注意事项

若图片内EOI (End Of Image，标记代码为0xFFD9) 之后，还有用户自定义的数据，则JPEG在对图片进行解码时，会直接清零EOI之后的8字节数据，若用户需要保留这些自定义的数据，则将图片数据读入内存之后，需要提前备份这部分数据，再传给JPEG处理。

若需要查看图片内EOI之后是否存在自定义数据，可以使用二进制查看工具打开图片查看，例如下图中的FFD9标记符之后就存在自定义数据。



1.9.17.2 JPEG 性能指标数据

Atlas 200/300/500 推理产品上，不支持本章中的JPEG功能。

Atlas 训练系列产品上，不支持本章中的JPEG功能。

性能指标说明

本节仅描述各产品对应的昇腾AI处理器的性能指标参考数据，各产品由于其内存、CPU等配置不同，在性能规格上会存在差异，详细产品的性能规格请查阅[昇腾社区](#)的官方说明。

本节以单个昇腾AI处理器为例，描述基本场景性能指标参考，如下表所示，其它分辨率可以等量估算。

下表中的1路对应一个通道，一个通道对应一个线程；各分辨率的含义为：720p指分辨率为1280*720的图片，1080p指分辨率为1920*1080的图片，4K指分辨率为3840*2160的图片。

JPEG性能指标是基于硬件解码的性能，JPEG硬件解码不支持3个SOS的图片解码，对于硬件不支持的格式，会使用软件解码，软件解码性能参考为1080P 15fps。JPEG解码的输出图片如果涉及旋转，则性能指标低于软件解码的参考值，例如对于1080P的图片，性能指标低于15fps。

表 1-64 Atlas 推理系列产品的昇腾 AI 处理器性能指标参考数据

场景举例	总帧率
1080p*n路 (1≤n≤16)	n*128fps
1080p*n路 (n>16)	2048fps
4k*n路 (1≤n≤16)	n*32fps
4k*n路 (n>16)	512fps

表 1-65 Atlas 200/500 A2 推理产品的昇腾 AI 处理器性能指标参考数据

场景举例	总帧率
1080p*n路 (1≤n≤2)	n*256fps
1080p*n路 (n>2)	512fps
4k*n路 (1≤n≤2)	n*64fps
4k*n路 (n>2)	128fps

表 1-66 Atlas A2 训练系列产品/Atlas 800I A2 推理产品的昇腾 AI 处理器性能指标参考数据

输入图片格式	输入图片文件大小	场景举例	总帧率
YUV400/420	≤500KB	1080p*n路 (1≤n≤28)	n*512fps
		1080p*n路 (n>28)	14336fps
YUV440/422/444	≤500KB	1080p*n路 (1≤n≤28)	n*256fps
		1080p*n路 (n>28)	7168fps
YUV400/420	≤2MB	4k*n路 (1≤n≤28)	n*128fps
		4k*n路 (n>28)	3584fps
YUV440/422/444	≤2MB	4k*n路 (1≤n≤28)	n*64fps
		4k*n路 (n>28)	1792fps

1.9.17.3 VDEC 功能及约束说明

Atlas 200/300/500 推理产品上，不支持本章中的VDEC功能。

Atlas 训练系列产品上，不支持本章中的VDEC功能。

功能说明

VDEC (Video Decoder) 实现视频的解码。

分辨率约束

- 输入码流分辨率：

版本	分辨率范围
Atlas 推理系列产品 Atlas A2训练系列产品/Atlas 800I A2推理产品	最大分辨率4096*4096，最小分辨率128*128。
Atlas 200/500 A2推理产品	对于H264输入码流，最大分辨率8192*8192，最小分辨率128*128。 对于H265输入码流，最大分辨率5400*8192，最小分辨率128*128。

- 输出图片分辨率：

版本	分辨率范围
Atlas 推理系列产品 Atlas A2训练系列产品/Atlas 800I A2推理产品	最大分辨率4096*4096，最小分辨率10*6。
Atlas 200/500 A2推理产品	输出图片最大分辨率8192*8192；当输入码流宽度小于或等于4096时，输出图片最小分辨率为10*6，当输入码流宽度大于4096时，输出图片最小分辨率为128*128。

内存申请/释放接口说明

输入内存的大小就是指实际的输入码流所占用的大小，输出内存的大小参见表1-67中的计算公式。

版本	内存申请/释放接口
Atlas 推理系列产品	VDEC视频解码时，调用 hi_mpi_dvpp_malloc \ hi_mpi_dvpp_free 接口申请\释放Device上的输入、输出内存，这部分内存的生命周期由用户自行管理。

版本	内存申请/释放接口
Atlas A2训练系列产品/Atlas 800I A2推理产品 Atlas 200/500 A2推理产品	VDEC视频解码时，支持以下两种内存申请接口： - 调用aclrtMalloc\aclrtFree接口申请\释放Device上的输入、输出内存，这部分内存的生命周期由用户自行管理。 - 调用hi_mpi_dvpp_malloc\hi_mpi_dvpp_free接口申请\释放Device上的输入、输出内存，这部分内存的生命周期由用户自行管理。 注意：调用hi_mpi_dvpp_malloc接口申请的内存为媒体数据处理的专用内存，但专用内存的地址空间有限，若关注内存规划或内存资源有限时，建议调用aclrtMalloc接口申请内存。

码流/图片格式、宽高对齐、内存约束

- 输入码流格式
 - H264 bp/mp/hp level5.1 YUV420编码的码流，当前只支持annex-B的裸码流。
 - H265 8/10bit level5.1 YUV420编码的码流，当前只支持annex-B的裸码流。
- 输出图片要求：

表 1-67 码流/图片格式、宽高对齐、内存大小约束

输出图片格式	输出图片宽、高对齐要求	输出图片宽stride、高stride、内存大小要求 Atlas 推理系列产品	输出图片宽stride、高stride、内存大小要求 Atlas 200/500 A2推理产品 Atlas A2训练系列产品/Atlas 800I A2推理产品
YUV420SP NV12 8bit	宽2对齐 高2对齐	宽stride为宽16对齐后的值，最小32，最大16384。	宽stride无对齐要求，与宽相同即可，最小10，最大24576。
YUV420SP NV21 8bit	宽2对齐 高2对齐	高stride为高2对齐后的值，最小6，最大16384。 内存大小（单位Byte） $\geq$ 宽stride * 高stride * 3/2	高stride无对齐要求，与高相同即可，最小6，最大24576。 内存大小（单位Byte） $\geq$ 宽stride * 高stride * 3/2

输出图片格式	输出图片宽、高对齐要求	输出图片宽stride、高stride、内存大小要求 Atlas 推理系列产品	输出图片宽stride、高stride、内存大小要求 Atlas 200/500 A2推理产品 Atlas A2训练系列产品/Atlas 800I A2推理产品
RGB888	无对齐要求	宽stride为宽16对齐后再乘以3的值，最小48，最大16384。 高stride无对齐要求，最小6，最大16384。 内存大小（单位Byte）≥ 宽stride * 高stride	宽stride为宽乘以3的值，最小30，最大24576。 高stride无对齐要求，与高相同即可，最小6，最大24576。 内存大小（单位Byte）≥ 宽stride * 高stride
BGR888	无对齐要求		
RGB888 Planar Atlas A2训练系列产品/Atlas 800I A2推理产品支持该格式。	无对齐要求	-	宽stride无对齐要求，与宽相同即可，最小10，最大24576。 高stride无对齐要求，与高相同即可，最小6，最大24576。 内存大小（单位Byte）≥ 宽stride * 高stride
BGR888 Planar Atlas A2训练系列产品/Atlas 800I A2推理产品支持该格式。	无对齐要求	-	

 说明

输出图片格式的定义请参见[hi_pixel_format](#)，宽stride、高stride等概念请参见[1.9.2 基本概念](#)。

其它约束

- Atlas A2训练系列产品/Atlas 800I A2推理产品，若输出图片格式为YUV420SP NV12 10bit或YUV420SP NV21 10bit，使用VDEC功能时，不支持缩放图片、色域转换。
- VDEC只支持对按帧输入码流进行解码。
- 若码流中有坏帧、缺帧等情况，解码器VDEC解码时会将该帧标记为解码失败，并上报异常。
- 通过隔行扫描方式编码出来的码流，VDEC仅支持解码H264 8bit编码的码流。
- 同时配置按帧发送码流与按解码序输出，可以达到快速解码和快速输出的目的，但这种场景不支持解码含有B帧的码流。

1.9.17.4 VDEC 性能指标数据

Atlas 200/300/500 推理产品上，不支持本章中的VDEC功能。

Atlas 训练系列产品上，不支持本章中的VDEC功能。

性能指标说明

本节仅描述各产品对应的昇腾AI处理器的性能指标参考数据，各产品由于其内存、CPU等配置不同，在性能规格上会存在差异，详细产品的性能规格请查阅[昇腾社区](#)的官方说明。

本节以单个昇腾AI处理器为例，描述基本场景性能指标参考，如下表所示，其它分辨率可以等量估算。

下表中的1路对应一个通道，一个通道对应一个线程；各分辨率的含义为：720p指分辨率为1280*720的图片，1080p指分辨率为1920*1080的图片，4K指分辨率为3840*2160的图片。

表 1-68 Atlas 推理系列产品的昇腾 AI 处理器性能指标参考数据

场景举例	总帧率
720p*n路(1≤n≤12)	n*480fps 说明 针对 Atlas 300V 视频解析卡，当1<n≤12时，总帧率都为n*400fps。
720p*n路(n>12)	5760fps 说明 针对 Atlas 300V 视频解析卡，当n>12时，总帧率都为4800fps。
1080p*n路(1≤n≤12)	n*240fps 说明 针对 Atlas 300V 视频解析卡，当1<n≤12时，总帧率都为n*200fps。
1080p*n路(n>12)	2880fps 说明 针对 Atlas 300V 视频解析卡，当n>12时，总帧率都为2400fps。
4k*n路(1≤n≤12)	n*60fps
4k*n路(n>12)	720fps

表 1-69 Atlas 200/500 A2 推理产品的昇腾 AI 处理器性能指标参考数据

场景举例	总帧率
720p*n路(1≤n≤4)	n*600fps
720p*n路(n>4)	2400fps

场景举例	总帧率
1080p*n路(1≤n≤4)	n*300fps
1080p*n路(n>4)	1200fps
4k*n路(1≤n≤4)	n*75fps
4k*n路(n>4)	300fps

表 1-70 Atlas A2 训练系列产品/Atlas 800I A2 推理产品的昇腾 AI 处理器性能指标参考数据

场景举例	总帧率
720p*n路(1≤n≤2)	n*480fps
720p*n路(n>2)	960fps
1080p*n路(1≤n≤2)	n*240fps
1080p*n路(n>2)	480fps
4k*n路(1≤n≤2)	n*60fps
4k*n路(n>2)	120fps

下表以1080P分辨率的输入码流为例，说明每路VDEC解码的最大内存消耗的计算公式，在计算公式中：

- 输入码流缓存大小：大于或等于解码通道大小（宽*高）的 3/4倍。
- 解码图像帧存大小：可调用[hi_vdec_get_pic_buf_size](#)接口获取解码图像帧存大小，该参数值跟输入码流分辨率相关。
- 视频解码图像Tmv缓存大小：可调用[hi_vdec_get_tmv_buf_size](#)接口获取视频解码图像Tmv缓存大小，该参数值跟输入码流分辨率相关。
- 输入码流缓存大小、解码图像帧存大小、视频解码图像Tmv缓存大小、参考帧数量均由用户调用[hi_mpi_vdec_create_chn](#)接口创建通道时设置。
- 解码后缓存图像帧数由用户在调用[hi_mpi_vdec_set_chn_param](#)接口时设置。

每路VDEC解码的内存消耗计算公式	场景举例	内存消耗 (单位为MB)
6MB + 输入码流缓存大小 * 2 + (解码图像帧存大小 + 视频解码图像Tmv缓存大小) * (参考帧数量 + 解码后缓存图像帧数 + 1)	• 输入码流格式H264 • 输入码流分辨率1080P • 输入码流缓存大小为4MB • 解码图像帧存大小为3MB • 视频解码图像Tmv缓存大小为0.5MB • 解码后缓存图像帧数2个	52.5MB (参考帧数量为8) 31.5MB (参考帧数量为2)
	• 输入码流格式H265 • 输入码流分辨率1080P • 输入码流缓存大小为4MB • 解码图像帧存大小为3MB • 视频解码图像Tmv缓存大小为1MB • 解码后缓存图像帧数2个	58MB (参考帧数量为8) 34MB (参考帧数量为2)

1.9.17.5 hi_mpi_vdec_create_chn

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

函数功能

根据设置的通道属性创建解码通道。

约束说明

- 单个Device上的通道号不能超出最大的通道号范围。
- 在创建解码通道之前必须保证通道未创建（或者已经销毁），否则会直接返回错误。
- 系统内存不足时会返回HI_ERR_VDEC_NO_MEM的错误码，可考虑扩展操作系统内存。
- 如果需要解码的H.264码流有B帧，或者需要解码的H.265码流支持时域运动矢量预测（sps_temporal_mvp_enabled_flag = 1），则创建通道时需要设置此通道支持时域运动矢量预测（temporal_mvp_en 设置为 1），还需要为其分配输出每一

帧Tmv (Temporal Motion Vector) 信息的VB (Video Buffer) 块, 该VB块的大小比图像VB块小很多, 所需个数为RefFrameNum+1, 具体大小通过调用函数 [hi_vdec_get_tmv_buf_size](#) 获取, 否则会导致解码出现花屏等错误。

- H264、H265解码每个解码通道所需VB个数至少为参考帧+显示帧+1, JPEG解码每个解码通道所需VB个数至少为显示帧+1。不同协议解码所需的图像VB块大小不同, 具体大小通过调用函数[hi_vdec_get_pic_buf_size](#)获取。
- 如果 H.264解码不需要解码B帧, 或者 H.265解码不需要解码支持时域运动矢量预测 (sps_temporal_mvp_enabled_flag = 1) 的码流, 则创建通道时可设置此通道不支持时域运动矢量预测 (temporal_mvp_en 设置为 0), 此种情况不输出Tmv信息, 可以不用创建Tmv VB池, 节省内存。
- 用户需要根据解码码流配置解码所需的帧存VB大小 frame_buf_size 和个数 frame_buf_cnt, 以及存放Tmv信息的VB大小 tmv_buf_size, 解码器内部按此配置创建相应的私有VB池。
- 只要帧存buffer和Tmvbuffer足够大, 解码通道能解码在最大最小分辨率范围内的任意分辨率码流。通道宽高当前只与码流buffer、SCDbuffer (start code detector buffer) 大小有关。

函数原型

[hi_s32](#) hi_mpi_vdec_create_chn([hi_vdec_chn](#) chn, const [hi_vdec_chn_attr](#) *attr)

参数说明

参数名	输入/输出	说明
chn	输入	解码通道号。 Atlas 推理系列产品，该参数的取值范围：[0, 256)，JPEGD功能和VDEC功能共用通道，且通道总数最多256。 Atlas A2训练系列产品/Atlas 800I A2推理产品，该参数的取值范围：[0, 256)，JPEGD功能和VDEC功能共用通道，且通道总数最多256，其中，JPEGD解码通道总数最多256，VDEC解码通道总数最多32。 Atlas 200/500 A2推理产品，该参数的取值范围：[0, 128)，JPEGD功能和VDEC功能共用通道，且通道总数最多128。 说明 对于Atlas 推理系列产品，昇腾虚拟化实例（Ascend Virtual Instance）场景下，VDEC通道和JPEGD通道总数 = (被分配的VDEC硬件单元+JPEGD硬件单元数量之和 / VDEC硬件单元 +JPEGD硬件单元总数之和) * 256，如果通道总数不为整数，则向下取整。 对于Atlas A2训练系列产品/Atlas 800I A2推理产品，昇腾虚拟化实例（Ascend Virtual Instance）场景下，VDEC通道 = (被分配的VDEC硬件单元 / VDEC硬件单元) * 32，如果通道总数不为整数，则向下取整。JPEGD的通道数不受算力影响，但JPEGD+VDEC的总通道数最大256。 对于Atlas 200/500 A2推理产品，昇腾虚拟化实例（Ascend Virtual Instance）场景下，VDEC通道 = (被分配的VDEC硬件单元 / VDEC硬件单元) * 128，如果通道总数不为整数，则向下取整。JPEGD的通道数不随算力影响，但JPEGD+VDEC的总通道数最大128。 昇腾虚拟化实例（Ascend Virtual Instance）的详细描述，以及各场景下的算力资源规格请参见《昇腾虚拟化实例（AVI）用户指南》。
attr	输入	解码通道属性指针。 如果参数attr为空，会返回错误码HI_ERR_VDEC_NULL_PTR。 当通道属性attr中的值超过解码能力集时，会返回HI_ERR_VDEC_ILLEGAL_PARAM的错误码。

返回值说明

- 0：成功
- 非0：失败，参见[1.9.21.11 VDEC视频解码/JPEGD图片解码返回码](#)

参考资源

接口调用流程及示例，参见JPEGD图片解码、VDEC视频解码。

1.9.17.6 hi_mpi_vdec_destroy_chn

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

函数功能

销毁解码通道，以释放相关资源。

约束说明

- 销毁前必须保证通道已创建，否则会返回通道未创建的错误。
- 销毁前必须停止接收码流（或者尚未开始接收码流），否则返回错误码 HI_ERR_VDEC_NOT_PERM。

函数原型

[hi_s32](#) hi_mpi_vdec_destroy_chn([hi_vdec_chn](#) chn)

参数说明

参数名	输入/输出	说明
chn	输入	解码通道号。 Atlas 推理系列产品，该参数的取值范围：[0, 256)，JPEGD功能和VDEC功能共用通道，且通道总数最多256。 Atlas A2训练系列产品/Atlas 800I A2推理产品，该参数的取值范围：[0, 256)，JPEGD功能和VDEC功能共用通道，且通道总数最多256，其中，JPEGD解码通道总数最多256，VDEC解码通道总数最多32。 Atlas 200/500 A2推理产品，该参数的取值范围：[0, 128)，JPEGD功能和VDEC功能共用通道，且通道总数最多128。

返回值说明

- 0：成功
- 非0：失败，参见[1.9.21.11 VDEC视频解码/JPEGD图片解码返回码](#)

参考资源

接口调用流程及示例，参见JPEGD图片解码、VDEC视频解码。

1.9.17.7 hi_mpi_vdec_get_chn_attr

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

函数功能

获取视频解码通道属性。

约束说明

获取属性前必须保证通道已创建，否则会返回通道未创建的错误码 HI_ERR_VDEC_UNEXIST。

函数原型

```
hi_s32 hi_mpi_vdec_get_chn_attr(hi_vdec_chn chn, hi_vdec_chn_attr *attr)
```

参数说明

参数名	输入/输出	说明
chn	输入	解码通道号。 Atlas 推理系列产品，该参数的取值范围：[0, 256)，JPEGD功能和VDEC功能共用通道，且通道总数最多256。 Atlas A2训练系列产品/Atlas 800I A2推理产品，该参数的取值范围：[0, 256)，JPEGD功能和VDEC功能共用通道，且通道总数最多256，其中，JPEGD解码通道总数最多256，VDEC解码通道总数最多32。 Atlas 200/500 A2推理产品，该参数的取值范围：[0, 128)，JPEGD功能和VDEC功能共用通道，且通道总数最多128。
attr	输出	解码通道属性指针。

返回值说明

- 0：成功
- 非0：失败，参见[1.9.21.11 VDEC视频解码/JPEGD图片解码返回码](#)

1.9.17.8 hi_mpi_vdec_set_chn_attr

函数功能

设置解码通道属性。
当前版本不支持该接口。

约束说明

- 获取属性前必须保证通道已创建，否则会返回通道未创建的错误码 HI_ERR_VDEC_UNEXIST。
- 可以改变的属性有type、mode、ref_frame_num，其它属性不能改变。其中，type只支持HI_PT_H264、HI_PT_H265。

- 解码器内部会在改变属性时会自动复位解码通道。

函数原型

hi_s32 hi_mpi_vdec_set_chn_attr(**hi_vdec_chn** chn, const **hi_vdec_chn_attr** *attr)

参数说明

参数名	输入/输出	说明
chn	输入	解码通道号。 Atlas 推理系列产品，该参数的取值范围：[0, 256)，JPEGD功能和VDEC功能共用通道，且通道总数最多256。 Atlas A2训练系列产品/Atlas 800I A2推理产品，该参数的取值范围：[0, 256)，JPEGD功能和VDEC功能共用通道，且通道总数最多256，其中，JPEGD解码通道总数最多256，VDEC解码通道总数最多32。 Atlas 200/500 A2推理产品，该参数的取值范围：[0, 128)，JPEGD功能和VDEC功能共用通道，且通道总数最多128。
attr	输入	解码通道属性指针。

返回值说明

- 0：成功
- 非0：失败，参见[1.9.21.11 VDEC视频解码/JPEGD图片解码返回码](#)

1.9.17.9 hi_mpi_vdec_start_recv_stream

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

函数功能

创建通道之后，启动解码之前，解码器开始接收用户发送的码流。

约束说明

- 启动接收码流前必须保证通道已创建，否则会返回通道未创建的错误码 HI_ERR_VDEC_UNEXIST。
- 启动接收码流之后，才能调用[hi_mpi_vdec_send_stream](#)发送码流。
- 重复调用启动接收码流接口时，返回成功。

函数原型

hi_s32 hi_mpi_vdec_start_recv_stream(**hi_vdec_chn** chn)

参数说明

参数名	输入/输出	说明
chn	输入	解码通道号。 Atlas 推理系列产品，该参数的取值范围：[0, 256)，JPEGD功能和VDEC功能共用通道，且通道总数最多256。 Atlas A2训练系列产品/Atlas 800I A2推理产品，该参数的取值范围：[0, 256)，JPEGD功能和VDEC功能共用通道，且通道总数最多256，其中，JPEGD解码通道总数最多256，VDEC解码通道总数最多32。 Atlas 200/500 A2推理产品，该参数的取值范围：[0, 128)，JPEGD功能和VDEC功能共用通道，且通道总数最多128。

返回值说明

- 0：成功
- 非0：失败，参见[1.9.21.11 VDEC视频解码/JPEGD图片解码返回码](#)

参考资源

接口调用流程及示例，参见JPEGD图片解码、VDEC视频解码。

1.9.17.10 hi_mpi_vdec_stop_rcv_stream

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

函数功能

码流发送结束之后，解码器停止接收用户发送的码流，在销毁解码通道之前必须调用此接口。

约束说明

- 停止接收码流前必须保证通道已创建，否则会返回通道未创建的错误码HI_ERR_VDEC_UNEXIST。
- 调用此接口后，再调用发送码流接口[hi_mpi_vdec_send_stream](#)会返回失败。
- 重复调用停止接收码流接口时，返回成功。

函数原型

[hi_s32](#) hi_mpi_vdec_stop_rcv_stream([hi_vdec_chn](#) chn)

参数说明

参数名	输入/输出	说明
chn	输入	解码通道号。 Atlas 推理系列产品，该参数的取值范围：[0, 256)，JPEGD功能和VDEC功能共用通道，且通道总数最多256。 Atlas A2训练系列产品/Atlas 800I A2推理产品，该参数的取值范围：[0, 256)，JPEGD功能和VDEC功能共用通道，且通道总数最多256，其中，JPEGD解码通道总数最多256，VDEC解码通道总数最多32。 Atlas 200/500 A2推理产品，该参数的取值范围：[0, 128)，JPEGD功能和VDEC功能共用通道，且通道总数最多128。

返回值说明

- 0：成功
- 非0：失败，参见[1.9.21.11 VDEC视频解码/JPEGD图片解码返回码](#)

参考资源

接口调用流程及示例，参见JPEGD图片解码、VDEC视频解码。

1.9.17.11 hi_mpi_vdec_query_status

Atlas 200/300/500 推理产品，不支持该接口。
Atlas 训练系列产品，不支持该接口。

函数功能

解码过程中通过此接口来查询解码通道状态。

约束说明

查询通道状态前必须保证通道已创建，否则会返回通道未创建的错误码HI_ERR_VDEC_UNEXIST。

函数原型

```
hi_s32 hi_mpi_vdec_query_status(hi_vdec_chn chn, hi_vdec_chn_status *status)
```

参数说明

参数名	输入/输出	说明
chn	输入	解码通道号。 Atlas 推理系列产品，该参数的取值范围：[0, 256)，JPEGD功能和VDEC功能共用通道，且通道总数最多256。 Atlas A2训练系列产品/Atlas 800I A2推理产品，该参数的取值范围：[0, 256)，JPEGD功能和VDEC功能共用通道，且通道总数最多256，其中，JPEGD解码通道总数最多256，VDEC解码通道总数最多32。 Atlas 200/500 A2推理产品，该参数的取值范围：[0, 128)，JPEGD功能和VDEC功能共用通道，且通道总数最多128。
status	输出	通道状态的指针。

返回值说明

- 0：成功
- 非0：失败，参见[1.9.21.11 VDEC视频解码/JPEGD图片解码返回码](#)

1.9.17.12 hi_mpi_vdec_reset_chn

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

函数功能

解码通道长时间没有反应的情况下或者解码失败等情况下，可以调用此接口复位通道，尝试恢复。

约束说明

- 复位前必须保证通道已创建，否则会返回错误码HI_ERR_VDEC_UNEXIST。
- 复位前必须停止接收码流，否则返回错误码HI_ERR_VDEC_NOT_PERM。

函数原型

```
hi_s32 hi_mpi_vdec_reset_chn(hi_vdec_chn chn)
```

参数说明

参数名	输入/输出	说明
chn	输入	解码通道号。 Atlas 推理系列产品，该参数的取值范围：[0, 256)，JPEGD功能和VDEC功能共用通道，且通道总数最多256。 Atlas A2训练系列产品/Atlas 800I A2推理产品，该参数的取值范围：[0, 256)，JPEGD功能和VDEC功能共用通道，且通道总数最多256，其中，JPEGD解码通道总数最多256，VDEC解码通道总数最多32。 Atlas 200/500 A2推理产品，该参数的取值范围：[0, 128)，JPEGD功能和VDEC功能共用通道，且通道总数最多128。

返回值说明

- 0：成功
- 非0：失败，参见[1.9.21.11 VDEC视频解码/JPEGD图片解码返回码](#)

1.9.17.13 hi_mpi_vdec_set_chn_param

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

函数功能

改变通道参数的时候调用此接口，设置解码通道参数。

约束说明

- 必须保证通道已创建，否则会返回错误码HI_ERR_VDEC_UNEXIST。
- 该接口可设置通道的一些高级属性，创建通道后有默认值，默认值见数据类型hi_vdec_chn_param的说明。
- 各项参数如超过合法范围，会返回 HI_ERR_VDEC_ILLEGAL_PARAM 的错误码，各参数范围见数据类型hi_vdec_chn_param的说明。
- 调用[hi_mpi_vdec_create_chn](#)接口创建通道时设置按帧发送码流（目前仅支持按帧发送），同时，调用本接口设置解码序输出码流，即将本接口的chn_param->video_param.out_order参数设置为HI_VIDEO_OUT_ORDER_DEC，可以达到快速解码和快速输出的目的，但这种场景不支持解码含有B帧的码流。

函数原型

[hi_s32](#) hi_mpi_vdec_set_chn_param([hi_vdec_chn](#) chn, const [hi_vdec_chn_param](#) *chn_param)

参数说明

参数名	输入/输出	说明
chn	输入	解码通道号。 Atlas 推理系列产品，该参数的取值范围：[0, 256)，JPEGD功能和VDEC功能共用通道，且通道总数最多256。 Atlas A2训练系列产品/Atlas 800I A2推理产品，该参数的取值范围：[0, 256)，JPEGD功能和VDEC功能共用通道，且通道总数最多256，其中，JPEGD解码通道总数最多256，VDEC解码通道总数最多32。 Atlas 200/500 A2推理产品，该参数的取值范围：[0, 128)，JPEGD功能和VDEC功能共用通道，且通道总数最多128。
chn_param	输入	解码通道参数指针。

返回值说明

- 0：成功
- 非0：失败，参见[1.9.21.11 VDEC视频解码/JPEGD图片解码返回码](#)

参考资源

接口调用流程及示例，参见JPEGD图片解码、VDEC视频解码。

1.9.17.14 hi_mpi_vdec_get_chn_param

Atlas 200/300/500 推理产品，不支持该接口。
Atlas 训练系列产品，不支持该接口。

函数功能

修改通道参数前可先调用此接口获取通道参数。

约束说明

必须保证通道已创建，否则会返回错误码HI_ERR_VDEC_UNEXIST。

函数原型

[hi_s32](#) hi_mpi_vdec_get_chn_param([hi_vdec_chn](#) chn, [hi_vdec_chn_param](#) *chn_param)

参数说明

参数名	输入/输出	说明
chn	输入	解码通道号。 Atlas 推理系列产品，该参数的取值范围：[0, 256)，JPEGD功能和VDEC功能共用通道，且通道总数最多256。 Atlas A2训练系列产品/Atlas 800I A2推理产品，该参数的取值范围：[0, 256)，JPEGD功能和VDEC功能共用通道，且通道总数最多256，其中，JPEGD解码通道总数最多256，VDEC解码通道总数最多32。 Atlas 200/500 A2推理产品，该参数的取值范围：[0, 128)，JPEGD功能和VDEC功能共用通道，且通道总数最多128。
chn_param	输出	解码通道参数指针。

返回值说明

- 0：成功
- 非0：失败，参见[1.9.21.11 VDEC视频解码/JPEGD图片解码返回码](#)

参考资源

接口调用流程及示例，参见JPEGD图片解码、VDEC视频解码。

1.9.17.15 hi_mpi_vdec_set_protocol_param

函数功能

设置协议相关的内存分配通道参数，如slice、pps（Picture Parameter Set）、sps（Sequence Parameter Set）等。

当前版本不支持该接口。

约束说明

- 必须保证通道已创建，否则会返回错误码HI_ERR_VDEC_UNEXIST。
- 设置协议参数前必须先停止接收码流，否则返回错误码HI_ERR_VDEC_NOT_PERM。
- 切换协议参数时，解码器内部会复位解码通道。
- 协议参数所需的内存是解码器内部根据当前码流实际vps（Video Parameter Set）/sps/pps/slice个数使用vmalloc动态分配的，最大上限不超过此接口设置的最大vps/sps/pps/slice个数。

函数原型

```
hi_s32 hi_mpi_vdec_set_protocol_param(hi_vdec_chn chn, const
hi_vdec_protocol_param *protocol_param)
```

参数说明

参数名	输入/输出	说明
chn	输入	解码通道号。 Atlas 推理系列产品，该参数的取值范围：[0, 256)，JPEGD功能和VDEC功能共用通道，且通道总数最多256。 Atlas A2训练系列产品/Atlas 800I A2推理产品，该参数的取值范围：[0, 256)，JPEGD功能和VDEC功能共用通道，且通道总数最多256，其中，JPEGD解码通道总数最多256，VDEC解码通道总数最多32。 Atlas 200/500 A2推理产品，该参数的取值范围：[0, 128)，JPEGD功能和VDEC功能共用通道，且通道总数最多128。
protocol_param	输入	解码通道参数指针。

返回值说明

- 0：成功
- 非0：失败，参见[1.9.21.11 VDEC视频解码/JPEGD图片解码返回码](#)

1.9.17.16 hi_mpi_vdec_get_protocol_param

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

函数功能

获取协议相关的内存分配通道参数，如slice、pps（Picture Parameter Set）、sps（Sequence Parameter Set）等。

约束说明

必须保证通道已创建，否则会返回错误码HI_ERR_VDEC_UNEXIST。

函数原型

```
hi_s32 hi_mpi_vdec_get_protocol_param(hi_vdec_chn chn,
hi_vdec_protocol_param *protocol_param)
```


参数说明

参数名	输入/输出	说明
chn	输入	解码通道号。 Atlas 推理系列产品，该参数的取值范围：[0, 256)，JPEGD功能和VDEC功能共用通道，且通道总数最多256。 Atlas A2训练系列产品/Atlas 800I A2推理产品，该参数的取值范围：[0, 256)，JPEGD功能和VDEC功能共用通道，且通道总数最多256，其中，JPEGD解码通道总数最多256，VDEC解码通道总数最多32。 Atlas 200/500 A2推理产品，该参数的取值范围：[0, 128)，JPEGD功能和VDEC功能共用通道，且通道总数最多128。
protocol_param	输出	解码通道参数指针。

返回值说明

- 0：成功
- 非0：失败，参见[1.9.21.11 VDEC视频解码/JPEGD图片解码返回码](#)

1.9.17.17 hi_mpi_vdec_send_stream

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

函数功能

解码前，向解码通道发送码流数据及存放解码结果的buffer。

约束说明

- 发送数据前必须保证已经调用[hi_mpi_vdec_start_rcv_stream](#)接口启动接收码流，否则直接返回该操作不允许的错误码HI_ERR_VDEC_NOT_PERM。如果在发送数据过程中停止接收码流，就会立刻返回HI_ERR_VDEC_NOT_PERM。
- 发送数据前必须保证通道已经被创建，否则直接返回通道未创建的错误码HI_ERR_VDEC_UNEXIST。如果在发送码流过程中复位通道或者销毁通道，就会立刻返回错误码HI_ERR_VDEC_UNEXIST。
- 发送码流时需要按照创建解码通道时设置的发送方式（仅支持按帧发送）进行发送。按帧发送时，调用此接口一次，必须发送完整的一帧码流，否则，解码会出现错误。
- 不能发送end_of_stream为0的空码流包（码流长度为0或码流地址为空），否则返回错误码。发送end_of_stream为1的空码流包时，解码器会把所有码流全部解完并输出全部图像。除此之外，其它情况应该把end_of_stream置为0。

- 当码流buffer为空且装不下当前包码流时，会返回参数超出范围的错误码HI_ERR_VDEC_ILLEGAL_PARAM。
- 以非阻塞方式发送码流，如果码流缓冲区已满，会立刻返回错误码HI_ERR_VDEC_BUF_FULL。
- 以超时方式发送码流，到达设定的超时时间还不能成功发送码流会返回错误码HI_ERR_VDEC_BUF_FULL。
- 视频解码时支持设置是否将解码结果写入输出内存中，hi_vdec_stream结构体中将need_display设置为0时，此帧码流的解码结果将不会写入到hi_vdec_pic_info结构体中配置的输出内存中，此时允许将输出内存地址配置为NULL。
[hi_mpi_vdec_get_frame](#)接口仍然能获取到结构hi_video_frame_info和hi_vdec_stream的信息，但是此时输出内存中无解码结果数据，需要用户释放输出内存（如果用户传入的输出内存地址是NULL，则无需释放）。
- 解码时，输入输出内存均需要在调用[hi_mpi_vdec_get_frame](#)接口获取结果之后才能进行释放。
- 视频解码，输入输出内存要求、输入码流格式、输出图片格式请参见[1.9.17.3 VDEC功能及约束说明](#)。
- 图像解码，输入输出内存要求、输入图片格式、输出图片格式请参见[1.9.17.1 JPEGD功能及约束说明](#)。

函数原型

[hi_s32](#) hi_mpi_vdec_send_stream([hi_vdec_chn](#) chn, const [hi_vdec_stream](#) *stream, [hi_vdec_pic_info](#) *vdec_pic_info, [hi_s32](#) milli_sec)

参数说明

参数名	输入/输出	说明
chn	输入	解码通道号。 Atlas 推理系列产品，该参数的取值范围：[0, 256)，JPEGD功能和VDEC功能共用通道，且通道总数最多256。 Atlas A2训练系列产品/Atlas 800I A2推理产品，该参数的取值范围：[0, 256)，JPEGD功能和VDEC功能共用通道，且通道总数最多256，其中，JPEGD解码通道总数最多256，VDEC解码通道总数最多32。 Atlas 200/500 A2推理产品，该参数的取值范围：[0, 128)，JPEGD功能和VDEC功能共用通道，且通道总数最多128。
stream	输入	输入码流信息的指针。 该结构体内的addr参数配置的地址为Device上的内存地址。
vdec_pic_info	输入	输出图片信息的指针。

参数名	输入/输出	说明
milli_sec	输入	超时时间，单位是毫秒。 • -1：阻塞方式 • 0：非阻塞方式 • >0：超时方式，配置具体的超时时间。超时时间受操作系统影响，一般偏差在操作系统的-一个时间片内，例如，操作系统的-一个时间片为4ms，用户设置的milli_sec参数值为1，则实际的超时时间在1ms到5ms范围内。在CPU负载高场景下，超时时间仍可能存在波动。

返回值说明

- 0：成功
- 非0：失败，参见[1.9.21.11 VDEC视频解码/JPEGD图片解码返回码](#)

参考资源

接口调用流程及示例，参见JPEGD图片解码、VDEC视频解码。

1.9.17.18 hi_mpi_vdec_get_frame

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

函数功能

解码后，获取解码通道的解码图像及输入Stream。

JPEGD图像解码场景下，在调用本接口时，如果硬件解码失败，则会在本接口内部触发软件解码。

约束说明

- 向VDEC获取解码结果，输入buffer和输出buffer一起获取。
- JPEGD图片解码时，默认对于jpeg(420)、jpeg(422)、jpeg(440)格式的输入图片，如果输入图片的宽高为奇数，JPEGD解码输出图片的宽高为向下2对齐；默认对于jpeg(444)、jpeg(400)格式的输入图片，JPEGD解码输出图片的宽高与输入图片保持一致。若需调整输出图片的宽、高对齐模式，可调用[hi_mpi_vdec_set_jpegd_precision_mode](#)接口。

函数原型

```
hi_s32 hi_mpi_vdec_get_frame(hi_vdec_chn chn, hi_video_frame_info  
*frame_info, hi_vdec_supplement_info *supplement, hi_vdec_stream *stream,  
hi_s32 milli_sec)
```

参数说明

参数名	输入/输出	说明
chn	输入	解码通道号。 Atlas 推理系列产品, 该参数的取值范围: [0, 256), JPEGD功能和VDEC功能共用通道, 且通道总数最多256。 Atlas A2训练系列产品/Atlas 800I A2推理产品, 该参数的取值范围: [0, 256), JPEGD功能和VDEC功能共用通道, 且通道总数最多256, 其中, JPEGD解码通道总数最多256, VDEC解码通道总数最多32。 Atlas 200/500 A2推理产品, 该参数的取值范围: [0, 128), JPEGD功能和VDEC功能共用通道, 且通道总数最多128。
frame_info	输出	已解码的图像信息的指针。解码后的数据存放在Device内存中。
supplement	输出	解码图像补充信息的指针。预留参数, 暂未使用, 建议用户设置为NULL。
stream	输出	已解码的输入码流信息指针。
milli_sec	输入	超时时间, 单位是毫秒。 • -1: 阻塞方式 • 0: 非阻塞方式 • >0: 超时方式, 配置具体的超时时间。超时时间受操作系统影响, 一般偏差在操作系统的片内, 例如, 操作系统的片为4ms, 用户设置的milli_sec参数值为1, 则实际的超时时间在1ms到5ms范围内。在CPU负载高场景下, 超时时间仍可能存在波动。 说明 JPEGD图像解码场景下, 此处配置的超时时间仅指获取硬件解码结果的等待时间, 不包含软件解码的时间。

返回值说明

- 0: 成功, 获取解码数据成功后, 需调用[hi_mpi_vdec_release_frame](#)接口释放解码相关资源。
本接口虽返回成功, 但有可能获取到的是解码失败的数据, 解码是否成功, 需要查看hi_video_frame_info结构体的v_frame成员下的[frame_flag](#)参数值说明。
- 非0: 失败, 参见[1.9.21.11 VDEC视频解码/JPEGD图片解码返回码](#)

注意:

 - 如果在解码图像时未创建通道, 则会返回错误码HI_ERR_VDEC_UNEXIST。
 - 如果在获取图像的过程中销毁通道, 则会返回错误码HI_ERR_VDEC_UNEXIST。

- 如果在获取解码图像的过程中复位通道，则会返回错误码HI_ERR_VDEC_UNEXIST。
- 以非阻塞方式获取解码图像，如果缓冲区内无图像，则会返回错误码HI_ERR_VDEC_BUF_EMPTY。
- 以超时方式获取解码图像，到达设定的超时时间还不能获取到硬件解码结果则会返回错误码HI_ERR_VDEC_BUF_EMPTY，此时则可增大超时时间，或循环调用本接口等待解码结果。

参考资源

接口调用流程及示例，参见JPEG图片解码、VDEC视频解码。

1.9.17.19 hi_mpi_vdec_release_frame

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

函数功能

解码之后，释放资源。

约束说明

- 此接口需与hi_mpi_vdec_get_frame配对使用，获取的数据应当在使用完之后立即释放，如果不及时释放，会导致解码过程阻塞等待资源。
- 释放的数据必须是调用hi_mpi_vdec_get_frame接口从该通道获取的数据，不得对数据信息结构体进行任何修改。

函数原型

```
hi_s32 hi_mpi_vdec_release_frame(hi_vdec_chn chn, const hi_video_frame_info *frame_info)
```

参数说明

参数名	输入/输出	说明
chn	输入	解码通道号。 Atlas 推理系列产品，该参数的取值范围：[0, 256)，JPEG功能 and VDEC功能共用通道，且通道总数最多256。 Atlas A2训练系列产品/Atlas 800I A2推理产品，该参数的取值范围：[0, 256)，JPEG功能 and VDEC功能共用通道，且通道总数最多256，其中，JPEG解码通道总数最多256，VDEC解码通道总数最多32。 Atlas 200/500 A2推理产品，该参数的取值范围：[0, 128)，JPEG功能 and VDEC功能共用通道，且通道总数最多128。

参数名	输入/输出	说明
frame_info	输入	解码后的图像信息指针，调用 hi_mpi_vdec_get_frame 接口获取。

返回值说明

- 0：成功
- 非0：失败，参见[1.9.21.11 VDEC视频解码/JPEGD图片解码返回码](#)

参考资源

接口调用流程及示例，参见JPEGD图片解码、VDEC视频解码。

1.9.17.20 hi_mpi_vdec_get_fd

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

函数功能

获取视频解码通道的设备文件句柄。

约束说明

在进程退出时会自动关闭文件句柄，回收资源。

函数原型

[hi_s32](#) hi_mpi_vdec_get_fd([hi_vdec_chn](#) chn)

参数说明

参数名	输入/输出	说明
chn	输入	解码通道号。 Atlas 推理系列产品，该参数的取值范围：[0, 256)，JPEGD功能和VDEC功能共用通道，且通道总数最多256。 Atlas A2训练系列产品/Atlas 800I A2推理产品，该参数的取值范围：[0, 256)，JPEGD功能和VDEC功能共用通道，且通道总数最多256，其中，JPEGD解码通道总数最多256，VDEC解码通道总数最多32。 Atlas 200/500 A2推理产品，该参数的取值范围：[0, 128)，JPEGD功能和VDEC功能共用通道，且通道总数最多128。

返回值说明

- 0或正数值：有效返回值
- 非正数值：无效返回值

1.9.17.21 hi_mpi_vdec_close_fd

函数功能

关闭视频解码的设备文件句柄。

当前版本不支持该接口。

函数原型

```
hi_s32 hi_mpi_vdec_close_fd(hi_vdec_chn chn)
```

参数说明

参数名	输入/输出	说明
chn	输入	解码通道号。 Atlas 推理系列产品，该参数的取值范围：[0, 256)，JPEGD功能和VDEC功能共用通道，且通道总数最多256。 Atlas A2训练系列产品/Atlas 800I A2推理产品，该参数的取值范围：[0, 256)，JPEGD功能和VDEC功能共用通道，且通道总数最多256，其中，JPEGD解码通道总数最多256，VDEC解码通道总数最多32。 Atlas 200/500 A2推理产品，该参数的取值范围：[0, 128)，JPEGD功能和VDEC功能共用通道，且通道总数最多128。

返回值说明

- 0：成功
- 非0：失败，参见[1.9.21.11 VDEC视频解码/JPEGD图片解码返回码](#)

1.9.17.22 hi_vdec_get_pic_buf_size

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

函数功能

获取解码图像需要的Buffer大小。用户在两个地方有使用这个函数：计算视频解码需要的内部缓冲区的大小，在初始化的时候配置对应的通道参数；计算图像解码的输出内存。

约束说明

提供给用户来计算输出图片缓冲区大小。

函数原型

```
hi_u32 hi_vdec_get_pic_buf_size(hi_payload_type type, hi_pic_buf_attr
*buf_attr)
```

参数说明

参数名	输入/输出	说明
type	输入	解码视频类型，当前支持HI_PT_H264、HI_PT_H265、HI_PT_JPEG。
buf_attr	输入	图片缓冲区信息的指针。

返回值说明

图像存储需要的图像大小。

解码视频类型为HI_PT_JPEG（表示JPEGD解码）时，pixel_format传入不支持的格式或者HI_PIXEL_FORMAT_UNKNOWN，返回值为0。

参考资源

接口调用流程及示例，参见VDEC视频解码。

1.9.17.23 hi_vdec_get_tmv_buf_size

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

函数功能

获取解码图像需要的矢量预测缓冲区的大小。

约束说明

提供给用户来计算矢量预测缓冲区大小，在创建通道的时候初始化对应的参数。

函数原型

```
hi_u32 hi_vdec_get_tmv_buf_size(hi_payload_type type, hi_u32 width, hi_u32
height)
```


参数说明

参数名	输入/输出	说明
type	输入	解码视频类型。
width	输入	解码图像宽。
height	输入	解码图像高。

返回值说明

图像存储需要矢量预测缓冲区大小。

参考资源

接口调用流程及示例，参见VDEC视频解码。

1.9.17.24 hi_mpi_vdec_set_jpegd_precision_mode

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

函数功能

设置JPEGD解码输出图片的宽、高对齐模式。

约束说明

- 调用本接口设置对齐模式前，必须保证通道已创建，否则会返回通道未创建的错误码HI_ERR_VDEC_UNEXIST。
- 调用本接口设置对齐模式，仅对JPEGD解码通道有效，其它类型通道返回操作不支持。
- 在调用hi_mpi_vdec_start_recv_stream接口之前调用本接口设置对齐模式，在调用hi_mpi_vdec_start_recv_stream接口之后调用本接口，系统会返回操作不允许的错误提示。

函数原型

hi_s32 hi_mpi_vdec_set_jpegd_precision_mode(**hi_vdec_chn** chn, **const hi_jpegd_precision_mode** mode)

参数说明

参数名	输入/输出	说明
chn	输入	解码通道号。 Atlas 推理系列产品，该参数的取值范围：[0, 256)，JPEGD功能和VDEC功能共用通道，且通道总数最多256。 Atlas A2训练系列产品/Atlas 800I A2推理产品，该参数的取值范围：[0, 256)，JPEGD功能和VDEC功能共用通道，且通道总数最多256，其中，JPEGD解码通道总数最多256，VDEC解码通道总数最多32。 Atlas 200/500 A2推理产品，该参数的取值范围：[0, 128)，JPEGD功能和VDEC功能共用通道，且通道总数最多128。
mode	输入	设置JPEGD解码输出图片的宽、高对齐模式。

返回值说明

- 0：成功
- 非0：失败，参见[1.9.21.11 VDEC视频解码/JPEGD图片解码返回码](#)

1.9.17.25 hi_mpi_vdec_get_jpegd_output_info

Atlas 200/300/500 推理产品，不支持该接口。
Atlas 训练系列产品，不支持该接口。

函数功能

图片解码场景下，根据输入码流，获取按源图格式或指定图片格式输出时的图片宽、高、宽stride、高stride、解码后占用的内存大小、图片格式等信息。
该接口会检查图片格式是否支持解码，如果遇到不支持的码流则该接口会返回HI_ERR_VDEC_NOT_SUPPORT错误码，该错误码的详细描述请参见[1.9.21.1 公共返回码](#)。

函数原型

```
hi_s32 hi_mpi_vdec_get_jpegd_output_info(const hi_vdec_stream *stream,
const hi_pixel_format output_format, hi_img_info *img_info)
```

参数说明

参数名	输入/输出	说明
stream	输入	输入码流信息的指针。 Atlas 推理系列产品，标准形态下，在Host上调用本接口时，该结构体内的addr参数配置的地址必须是Host上的内存地址。 Atlas A2训练系列产品/Atlas 800I A2推理产品，在Host上调用本接口时，该结构体内的addr参数配置的地址必须是Host上的内存地址。 Atlas 200/500 A2推理产品，EP模式下，在Host上调用本接口时，该结构体内的addr参数配置的地址必须是Host上的内存地址。 Atlas 200/500 A2推理产品，RC模式下，在Device上调用本接口时，该结构体内的addr参数配置的地址必须是Device上的内存地址。
output_format	输入	输出图片格式。 若按源图格式获取相关信息，则将该参数值设置为HI_PIXEL_FORMAT_UNKNOWN；若按指定格式获取相关信息，则将该参数值设置为对应的图片格式，支持的图片格式请参见 图片格式、宽高对齐、内存约束 。
img_info	输出	输出图片信息结构体，包含图片宽、高、宽stride、高stride、解码后占用的内存大小、图片格式等信息。 其中，输出图片格式，通过img_pixel_format参数返回。

返回值说明

- 0：成功
- 非0：失败，参见[1.9.21.1 公共返回码](#)、[1.9.21.11 VDEC视频解码/JPEGD图片解码返回码](#)。

1.9.17.26 hi_mpi_vdec_set_display_mode

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

函数功能

若调用[hi_mpi_sys_bind](#)接口绑定数据接收者VPSS与数据源VDEC后，可调用本接口设置显示模式。不设置显示模式时，默认为回放模式。

约束说明

- 设置显示模式前必须保证通道已创建，否则会返回通道未创建的错误码 HI_ERR_VDEC_UNEXIT。
- 预览模式（ HI_VIDEO_DISPLAY_MODE_PREVIEW ）：预览模式下VDEC绑定的直接后级模块（比如 VPSS）以非阻塞方式接收解码图像，即当VPSS的图像Buffer满时（解码帧存个数比 VPSS 缓存队列个数多），VPSS丢弃VDEC发送过来的图像，以达到不反压VDEC 解码的目的，实现实时预览。需要注意的是，当解码帧存个数比VPSS缓存队列个数少时，即使开启预览模式，VPSS还是会反压解码。
- 回放模式（ HI_VIDEO_DISPLAY_MODE_PLAYBACK ）：回放模式下VDEC绑定的直接后级模块（比如 VPSS）以阻塞方式接收解码图像，即当VPSS的图像Buffer满时，拒绝接收VDEC发送过来的图像，VDEC发现当前图像发送失败后启动图像重新发送机制，直到图像发送成功为止。回放模式下VDEC绑定的直接后级模块能够反压VDE 解码，以达到不丢弃任何一帧解码图像的回放效果。

函数原型

```
hi_s32 hi_mpi_vdec_set_display_mode(hi_vdec_chn chn, hi_video_display_mode display_mode)
```

参数说明

参数名	输入/输出	说明
chn	输入	解码通道号。 Atlas A2训练系列产品/Atlas 800I A2推理产品，该参数的取值范围：[0, 256)，JPEGD功能和VDEC功能共用通道，且通道总数最多256，其中，JPEGD解码通道总数最多256，VDEC解码通道总数最多32。 Atlas 200/500 A2推理产品，该参数的取值范围：[0, 128)，JPEGD功能和VDEC功能共用通道，且通道总数最多128。
display_mode	输入	显示模式枚举。

返回值说明

- 0：成功
- 非0：失败，参见[1.9.21.1 公共返回码](#)、[1.9.21.11 VDEC视频解码/JPEGD图片解码返回码](#)。

1.9.17.27 hi_mpi_vdec_get_display_mode

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

函数功能

获取显示模式。

约束说明

获取显示模式前必须保证通道已创建，否则会返回通道未创建的错误码 HI_ERR_VDEC_UNEXIT。

函数原型

```
hi_s32 hi_mpi_vdec_get_display_mode(hi_vdec_chn chn,  
hi_video_display_mode *display_mode)
```

参数说明

参数名	输入/输出	说明
chn	输入	解码通道号。 Atlas A2训练系列产品/Atlas 800I A2推理产品，该参数的取值范围：[0, 256)，JPEGD功能和VDEC功能共用通道，且通道总数最多256，其中，JPEGD解码通道总数最多256，VDEC解码通道总数最多32。 Atlas 200/500 A2推理产品，该参数的取值范围：[0, 128)，JPEGD功能和VDEC功能共用通道，且通道总数最多128。
display_mode	输出	显示模式枚举指针。

返回值说明

- 0：成功
- 非0：失败，参见[1.9.21.1 公共返回码](#)、[1.9.21.11 VDEC视频解码/JPEGD图片解码返回码](#)。

1.9.18 VENC 视频编码功能/JPEGE 图片编码功能

1.9.18.1 JPEGE 功能及约束说明

Atlas 200/300/500 推理产品上，不支持本章中的JPEGE功能。
Atlas 训练系列产品上，不支持本章中的JPEGE功能。

功能说明

JPEGE (JPEG Encoder) 将YUV格式图片编码成JPEG压缩格式的图片文件，例如 *.jpg。

图片分辨率约束

- 输入图片分辨率：

版本	分辨率范围
Atlas 推理系列产品 Atlas A2训练系列产品/ Atlas 800I A2推理产品	最大分辨率：8192*8192，最小分辨率：32*32。
Atlas 200/500 A2推理产品	最大分辨率：16384*16384，最小分辨率：32*32。
- 输出图片分辨率
JPEGE只对图片编码，不会改变图片分辨率，因此输出与输入的图片分辨率保持一致。

内存申请/释放接口说明

输入内存的大小就是指实际的输入码流所占用的大小，输出内存的大小参见[表1-71](#)中的计算公式。

版本	输入内存申请/释放接口	输出内存申请/释放接口
Atlas 推理系列产品	JPEGE图片编码功能时，调用 hi_mpi_dvpp_malloc \hi_mpi_dvpp_free 接口申请\释放Device上的输入内存，这部分内存的生命周期由用户自行管理。	支持以下两种方式： ● 不需要用户管理、由DVPP内部管理输出内存时，调用 hi_mpi_venc_send_frame接口发送原始图像进行图像编码。 在调用hi_mpi_venc_create_chn接口创建通道时，必须正确设置 hi_venc_chn_attr.venc_attr.buf_size参数值（参数描述请参见1.9.20.14.5 hi_venc_attr）。 该方式下，相比由用户管理内存，输出结果数据的JPEG头中不存在COM注释字段，数据长度会短一点，但需要用户从DVPP返回的内存中拷贝输出结果数据到指定内存。 ● 由用户自行管理输出内存、管理内存的生命周期，调用 hi_mpi_venc_send_jpege_frame接口发送原始图像进行图像编码。 在调用hi_mpi_venc_create_chn接口创建通道时，需将 hi_venc_chn_attr.venc_attr.buf_size参数值设置为0（参数描述请参见1.9.20.14.5 hi_venc_attr），然后调用 hi_mpi_venc_get_jpege_predicted_size接口预估输出内存大小，调用 hi_mpi_dvpp_malloc \hi_mpi_dvpp_free接口申请\释放输出内存。 该方式下，直接在调用 hi_mpi_venc_send_jpege_frame接口时，设置输出内存地址，输出结果数据直接存放到用户设置的内存中，相比由系统管理内存的方式，用户可减少一次“从DVPP返回的内存中拷贝输出结果数据到指定内存”的操作，但输出结果数据的JPEG头中可能会存在COM注释字段（字段长度范围4~19Byte），数据长度会长一点。

版本	输入内存申请/释放接口	输出内存申请/释放接口
Atlas A2训练系列产品/Atlas 800I A2推理产品 Atlas 200/500 A2推理产品	JPEGE图片编码功能时，支持以下两种内存申请接口： - 调用aclrtMalloc \ aclrtFree接口申请\释放Device上的输入内存，这部分内存的生命周期由用户自行管理。 - 调用hi_mpi_dvpp_malloc \ hi_mpi_dvpp_free接口申请\释放Device上的输入内存，这部分内存的生命周期由用户自行管理。 注意：调用hi_mpi_dvpp_malloc接口申请的内存为媒体数据处理的专用内存，但专用内存的地址空间有限，若关注内存规划或内存资源有限时，建议调用aclrtMalloc接口申请内存。	支持以下两种方式： - 不需要用户管理、由DVPP内部管理输出内存时，调用hi_mpi_venc_send_frame接口发送原始图像进行图像编码。 在调用hi_mpi_venc_create_chn接口创建通道时，必须正确设置hi_venc_chn_attr.venc_attr.buf_size参数值（参数描述请参见1.9.20.14.5 hi_venc_attr）。 - 该方式下，相比由用户管理内存，输出结果数据的JPEG头中不存在COM注释字段，数据长度会短一点，但需要用户从DVPP返回的内存中拷贝输出结果数据到指定内存。 - 由用户自行管理输出内存、管理内存的生命周期，调用hi_mpi_venc_send_jpege_frame接口发送原始图像进行图像编码。 在调用hi_mpi_venc_create_chn接口创建通道时，需将hi_venc_chn_attr.venc_attr.buf_size参数值设置为0（参数描述请参见1.9.20.14.5 hi_venc_attr），然后调用hi_mpi_venc_get_jpege_predicted_size接口预估输出内存大小，调用hi_mpi_dvpp_malloc \ hi_mpi_dvpp_free接口申请\释放输出内存（也支持调用aclrtMalloc \ aclrtFree接口申请\释放Device上的输出内存）。 - 该方式下，直接在调用hi_mpi_venc_send_jpege_frame接口时，设置输出内存地址，输出结果数据直接存放到用户设置的内存中，相比由系统管理内存的方式，用户可减少一次“从DVPP返回的内存中拷贝输出结果数据到指定内存”的操作，但输出结果数据的JPEG头中可能会存在COM注释字段（字段长度范围4~19Byte），数据长度会长一点。

图片格式、宽高对齐、内存约束

表 1-71 图片格式、宽高对齐、内存大小约束

输入图片格式	输入图片宽、高对齐要求	输入图片宽stride、高stride、内存大小要求	输出图片格式
YUV420SP NV12 8bit	宽2对齐 高2对齐	宽stride为宽16对齐后的值，兼容对齐到16的倍数如128（按128对齐时，性能更优）。 高stride无需设置。 内存大小（单位Byte）= 宽stride * 高 * 3/2	JPEG压缩格式的图片文件，例如*.jpg。 只支持huffman编码，不支持算术编码，不支持渐进编码。
YUV420SP NV21 8bit			
YUV422Packed YUYV 8bit	宽2对齐 高无对齐要求	宽stride为宽的两倍再16对齐后的值。 高stride无需设置。 内存大小（单位Byte）= 宽stride * 高	
YUV422Packed UYVY 8bit			
YUV422Packed YVYU 8bit			
YUV422Packed VYUY 8bit			

说明

输入图片格式的定义请参见[hi_pixel_format](#)，宽stride、高stride等概念请参见[1.9.2 基本概念](#)。

1.9.18.2 JPEG 性能指标说明

Atlas 200/300/500 推理产品上，不支持本章中的JPEG功能。

Atlas 训练系列产品上，不支持本章中的JPEG功能。

性能指标说明

本节仅描述各产品对应的昇腾AI处理器的性能指标参考数据，各产品由于其内存、CPU等配置不同，在性能规格上会存在差异，详细产品的性能规格请查阅[昇腾社区](#)的官方说明。

本节以单个昇腾AI处理器为例，描述基本场景性能指标参考，如下表所示，其它分辨率可以等量估算。

下表中的1路对应一个通道，一个通道对应一个线程；各分辨率的含义为：1080p指分辨率为1920*1080的图片，4K指分辨率为3840*2160的图片。

表 1-72 Atlas 推理系列产品的昇腾 AI 处理器性能指标参考数据

场景举例（输入图片格式：YUV420 8bit）	总帧率
1080p*n路（1≤n≤8）	n*128fps
1080p*n路（n>8）	1024fps
4k*n路（1≤n≤8）	n*32fps
4k*n路（n>8）	256fps

表 1-73 Atlas 200/500 A2 推理产品的昇腾 AI 处理器性能指标参考数据

场景举例（输入图片格式：YUV420 8bit）	总帧率
1080p*n路	256fps
4k*n路	64fps

表 1-74 Atlas A2 训练系列产品/Atlas 800I A2 推理产品的昇腾 AI 处理器性能指标参考数据

场景举例（输入图片格式：YUV420 8bit）	总帧率
1080p*n路（1≤n≤4）	n*256fps
1080p*n路（n>4）	1024fps
4k*n路（1≤n≤4）	n*64fps
4k*n路（n>4）	256fps

1.9.18.3 VENC 功能及约束说明

Atlas 200/300/500 推理产品上，不支持本章中的VENC功能。

Atlas 训练系列产品上，不支持本章中的VENC功能。

Atlas A2训练系列产品/Atlas 800I A2推理产品上，不支持该功能。

功能说明

VENC（Video Encoder）将YUV420SP NV12/NV21-8bit图片数据编码成H264/H265格式的视频码流。

分辨率约束

- 输入图片分辨率:

版本	分辨率范围
Atlas 推理系列产品	宽和高的范围都是128~4096，对于H264格式的码流，最大分辨率的乘积不超过4096*2304。
Atlas 200/500 A2推理产品	最大分辨率8192*8192，最小分辨率114*114

- 输出码流分辨率
VENC只对图片编码，不会改变图片分辨率，因此输出与输入的图片分辨率保持一致。

内存申请/释放接口说明

输入内存的大小参见表1-75中的计算公式，输出内存无需用户管理、由系统管理。

版本	内存申请/释放接口
Atlas 推理系列产品	VENC视频编码时，调用 hi_mpi_dvpp_malloc \ hi_mpi_dvpp_free 接口申请\释放Device上的输入内存，这部分内存的生命周期由用户自行管理。
Atlas 200/500 A2推理产品	VENC视频编码时，支持以下 两种 内存申请接口： - 调用aclrtMalloc\aclrtFree接口申请\释放Device上的输入内存，这部分内存的生命周期由用户自行管理。 - 调用hi_mpi_dvpp_malloc\hi_mpi_dvpp_free接口申请\释放Device上的输入内存，这部分内存的生命周期由用户自行管理。 注意： 调用 hi_mpi_dvpp_malloc 接口申请的内存为媒体数据处理的专用内存，但专用内存的地址空间有限，若关注内存规划或内存资源有限时，建议调用 aclrtMalloc 接口申请内存。

码流/图片格式、宽高对齐、内存约束

表 1-75 图片格式、宽高对齐、内存大小约束

输入图片格式	输入图片宽、高对齐要求	输入图片宽stride、高stride、内存大小要求	输出码流格式
YUV420SP NV12 8bit	宽2对齐 高2对齐	宽stride为宽16对齐后的值。 高stride无需设置。 内存大小（单位Byte）= 宽stride * 高 * 3/2	- H264 BP/MP/HP - H265 MP（仅支持Slice码流）
YUV420SP NV21 8bit			

 说明

输入图片格式的定义请参见[hi_pixel_format](#)，宽stride、高stride等概念请参见[1.9.2 基本概念](#)。

1.9.18.4 VENC 性能指标说明

Atlas 200/300/500 推理产品上，不支持本章中的VENC功能。

Atlas 训练系列产品上，不支持本章中的VENC功能。

Atlas A2训练系列产品/Atlas 800I A2推理产品上，不支持该功能。

性能指标说明

本节仅描述各产品对应的昇腾AI处理器的性能指标参考数据，各产品由于其内存、CPU等配置不同，在性能规格上会存在差异，详细产品的性能规格请查阅[昇腾社区](#)的官方说明。

本节以单个昇腾AI处理器为例，描述基本场景性能指标参考，如下表所示，其它分辨率可以等量估算。

下表中的1路对应一个通道，一个通道对应一个线程；各分辨率的含义为：720p指分辨率为1280*720的图片，1080p指分辨率为1920*1080的图片，4K指分辨率为3840*2160的图片。

表 1-76 Atlas 推理系列产品的昇腾 AI 处理器性能指标参考数据

场景举例	总帧率
720p*n路(1≤n≤3)	n*480fps
720p*n路(n>3)	1440fps
1080p*n路(1≤n≤3)	n*240fps
1080p*n路(n>3)	720fps
4k*n路(1≤n≤3)	n*60fps
4k*n路(n>3)	180fps

表 1-77 Atlas 200/500 A2 推理产品的昇腾 AI 处理器性能指标参考数据

场景举例	总帧率
720p*n路(1≤n≤3)	n*400fps
720p*n路(n>3)	1200fps
1080p*n路(1≤n≤3)	n*200fps
1080p*n路(n>3)	600fps
4k*n路(1≤n≤3)	n*50fps
4k*n路(n>3)	150fps

1.9.18.5 hi_mpi_venc_create_chn

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

函数功能

创建编码通道。

约束说明

- 单个Device上的通道号不能超出最大的通道号范围。
- 编码通道属性由三部分组成，编码器属性、码率控制器属性和帧结构类型属性，帧结构类型属性简称GOP（ Group of Pictures ）类型属性。
- 编码器属性必须设置编码码流buffer深度、获取码流方式、编码profile等，[表 1-78](#)详细描述了各种协议的各项属性的特性。
- 推荐的编码宽高为：3840*2160(4k)、1920*1080(1080P)、1280*720(720P)。
- H.264/H.265编码帧存由YHeaderSize、CHheaderSize、YSize、CSize、PmeSize、PmeInfoSize和TmvSize共同构成，编码器默认根据最大宽高计算帧存进行内存分配，在设置通道宽高时需保证根据通道宽高计算的每一部分帧存大小都不能大于最大宽高计算的帧存大小。
- 码率控制器属性中的输入/输出帧率（ src_frame_rate/dst_frame_rate ）是用于配合编码过程中的码率（ bit_rate ）分配，不是最终的播放器显示帧率，若想要设置显示帧率请参考[hi_mpi_venc_set_h264_vui](#)或[hi_mpi_venc_set_h265_vui](#)接口。

建议用户将码控帧率与显示帧率设置一致，否则播放时的码率可能与用户设置的编码码率不同，播放时的码率会随着所设置的显示帧率而变化。

函数原型

[hi_s32](#) hi_mpi_venc_create_chn([hi_venc_chn](#) chn, const [hi_venc_chn_attr](#) *attr)

参数说明

参数名	输入/输出	说明
chn	输入	编码通道号。 Atlas 推理系列产品，编码通道号的取值范围：[0, 256)，JPEGE功能和VENC功能共用通道，且通道总数最多256。 Atlas A2训练系列产品/Atlas 800I A2推理产品，编码通道号的取值范围：[0, 256)，JPEGE功能通道总数最多256。 Atlas 200/500 A2推理产品，编码通道号的取值范围：[0, 128)，JPEGE功能和VENC功能共用通道，且通道总数最多128。 说明 对于Atlas 推理系列产品，昇腾虚拟化实例（Ascend Virtual Instance）场景下，VENC通道和JPEGE通道总数 = (被分配的VENC硬件单元+JPEGE硬件单元数量之和 / VENC硬件单元+JPEGE硬件单元总数之和) * 256，如果通道总数不为整数，则向下取整。 对于Atlas A2训练系列产品/Atlas 800I A2推理产品，昇腾虚拟化实例（Ascend Virtual Instance）场景下，JPEGE通道总数最多256。 对于Atlas 200/500 A2推理产品，昇腾虚拟化实例（Ascend Virtual Instance）场景下，VENC通道 = (被分配的VENC硬件单元 / VENC硬件单元) * 128，如果通道总数不为整数，则向下取整。JPEGE的通道数不随算力影响，但JPEGE+VENC的总通道数最大128。 昇腾虚拟化实例（Ascend Virtual Instance）的详细描述，以及各场景下的算力资源规格请参见《昇腾虚拟化实例（AVI）用户指南》。
attr	输入	编码通道属性的指针。

返回值说明

- 0：成功
- 非0：失败，参见[1.9.21.12 VENC视频编码/JPEGE图片编码返回码](#)

参考资源

接口调用流程及示例，参见JPEGE图片编码、VENC视频编码。

参考信息

表 1-78 编码器属性的约束

编码协议	编码方式	码流 buffer 深度	获取码流模式	编码 profile
H.264	Frame	- 当 mini_buf_mode=0时, Buffer≥ max_pic_width*max_pic_height*3/4; - 当 mini_buf_mode=1时, Buffer≥32*1024 Byte;	Frame/ Slice	Baseline Mainprofile Highprofile
JPEG	Frame	- mini_buf_mode=0 时, 如果原图格式为YUV420, Buffer最小值为原图宽16对齐*原图高16对齐*3/2 ; 如果原图格式为 YUV422 Packed, Buffer最小值为原图宽16对齐*原图高16对齐*2 - mini_buf_mode=1 时, Buffer≥32*1024 Byte;	Frame/Ecs	Baseline
H.265	Frame	- mini_buf_mode=0时, Buffer≥ max_pic_width*max_pic_height*3/4; - mini_buf_mode=1时, Buffer≥32*1024 Byte	Frame/ Slice	Main profile Main 10 profile (暂不支持)
		- mini_buf_mode=0时, Buffer≥ max_pic_width*max_pic_height*(3/4)*(5/4) ; - mini_buf_mode=1时, Buffer≥32*1024 Byte	Frame/ Slice	Main 10 profile (暂不支持)

1.9.18.6 hi_mpi_venc_destroy_chn

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

函数功能

用户退出编码流程时，需要关闭之前打开的编码通道，以释放相关资源。

约束说明

- 销毁并不存在的通道，返回失败。

- 销毁前必须停止接收图像，否则返回失败。

函数原型

```
hi_s32 hi_mpi_venc_destroy_chn(hi_venc_chn chn)
```

参数说明

参数名	输入/输出	说明
chn	输入	编码通道号。 Atlas 推理系列产品，编码通道号的取值范围：[0, 256)，JPEG功能 and VENC功能共用通道，且通道总数最多256。 Atlas A2训练系列产品/Atlas 800I A2推理产品，编码通道号的取值范围：[0, 256)，JPEG功能通道总数最多256。 Atlas 200/500 A2推理产品，编码通道号的取值范围：[0, 128)，JPEG功能和VENC功能共用通道，且通道总数最多128。

返回值说明

- 0：成功
- 非0：失败，参见[1.9.21.12 VENC视频编码/JPEG图片编码返回码](#)

参考资源

接口调用流程及示例，参见JPEG图片编码、VENC视频编码。

1.9.18.7 hi_mpi_venc_start_chn

Atlas 200/300/500 推理产品，不支持该接口。
Atlas 训练系列产品，不支持该接口。

函数功能

开启编码通道接收输入图片，允许指定接收帧数，超出指定的帧数后自动停止接收图像。

约束说明

- 如果通道未创建，则返回错误码HI_ERR_VENC_UNEXIST。
- 需要接收的图像帧数设置为-1时表示不指定帧数。
- 如果通道已经开始接收图像，没有停止接收图像前再一次调用此接口指定接收帧数，返回操作不允许。
- 如果通道已经开始接收图像，没有停止接收图像前允许再一次调用此接口不指定接收帧数。

- 只有开启接收之后编码器才开始接收图像编码。

函数原型

```
hi_s32 hi_mpi_venc_start_chn(hi_venc_chn chn, const hi_venc_start_param *recv_param)
```

参数说明

参数名	输入/输出	说明
chn	输入	编码通道号。 Atlas 推理系列产品，编码通道号的取值范围：[0, 256)，JPEGE功能和VENC功能共用通道，且通道总数最多256。 Atlas A2训练系列产品/Atlas 800I A2推理产品，编码通道号的取值范围：[0, 256)，JPEGE功能通道总数最多256。 Atlas 200/500 A2推理产品，编码通道号的取值范围：[0, 128)，JPEGE功能和VENC功能共用通道，且通道总数最多128。
recv_param	输入	接收图像参数的指针，用于指定需要接收的图像帧数。

返回值说明

- 0：成功
- 非0：失败，参见[1.9.21.12 VENC视频编码/JPEGE图片编码返回码](#)

参考资源

接口调用流程及示例，参见JPEGE图片编码、VENC视频编码。

1.9.18.8 hi_mpi_venc_stop_chn

Atlas 200/300/500 推理产品，不支持该接口。
Atlas 训练系列产品，不支持该接口。

函数功能

停止编码通道接收输入图片。

约束说明

- 如果通道未创建，则返回失败。
- 此接口并不判断当前是否停止接收，即允许重复停止接收不返回错误。
- 此接口用于编码通道停止接收图像来编码，在编码通道销毁或复位前必须停止接收图像。

- 调用此接口仅停止接收原始数据编码，码流buffer并不会被清除。

函数原型

```
hi_s32 hi_mpi_venc_stop_chn(hi_venc_chn chn)
```

参数说明

参数名	输入/输出	说明
chn	输入	编码通道号。 Atlas 推理系列产品，编码通道号的取值范围：[0, 256)，JPEGE功能和VENC功能共用通道，且通道总数最多256。 Atlas A2训练系列产品/Atlas 800I A2推理产品，编码通道号的取值范围：[0, 256)，JPEGE功能通道总数最多256。 Atlas 200/500 A2推理产品，编码通道号的取值范围：[0, 128)，JPEGE功能和VENC功能共用通道，且通道总数最多128。

返回值说明

- 0：成功
- 非0：失败，参见[1.9.21.12 VENC视频编码/JPEGE图片编码返回码](#)

参考资源

接口调用流程及示例，参见JPEGE图片编码、VENC视频编码。

1.9.18.9 hi_mpi_venc_query_status

Atlas 200/300/500 推理产品，不支持该接口。
Atlas 训练系列产品，不支持该接口。

函数功能

查询编码通道状态。

约束说明

- 如果通道未创建，则返回失败。
- 此接口用于查询此函数调用时刻的编码器状态，status包含以下信息：
 - 在编码通道状态结构体中，left_pics表示待编码的帧个数。在复位通道前，可以通过查询是否还有图像待编码来决定复位时机，防止复位时将可能需要编码的帧清理出去。
 - 在编码通道状态结构体中，left_stream_bytes表示码流buffer中用户暂未取走的byte数目。

在复位通道前，可以通过查询是否还有码流没有被处理来决定复位时机，防止复位时将可能需要的码流清理出去。

- 在编码通道状态结构体中，left_stream_frames表示码流buffer中用户暂未取走的帧数目。
在复位通道前，可以通过查询是否还有图像的码流没有被取走来决定复位时机，防止复位时将可能需要的码流清理出去。
- 在编码通道状态结构体中，cur_packs表示当前帧的码流包个数。在调用 [hi_mpi_venc_get_stream](#)之前应确保cur_packs大于0。
在按包获取时当前帧可能不是一个完整帧（被取走一部分），按帧获取时表示当前一个完整帧的包个数（如果没有一帧数据则为0）。用户在需要按帧获取码流时，需要查询一个完整帧的包个数，在这种情况下，通常可以在select成功后执行query操作，此时，cur_packs是当前完整帧中包的个数。
- 在编码通道状态结构体中，left_recv_pics表示调用[hi_mpi_venc_start_chn](#)接口后剩余等待接收的帧数目。
- 在编码通道状态结构体中，left_enc_pics表示调用[hi_mpi_venc_start_chn](#)接口后剩余等待编码的帧数目。
- 如果调用[hi_mpi_venc_start_chn](#)但没有指定接收帧数，left_recv_pics和left_enc_pics数目始终为0。

函数原型

[hi_s32](#) [hi_mpi_venc_query_status](#)([hi_venc_chn](#) chn, [hi_venc_chn_status](#) *status)

参数说明

参数名	输入/输出	说明
chn	输入	编码通道号。 Atlas 推理系列产品，编码通道号的取值范围：[0, 256)，JPEG功能 and VENC功能共用通道，且通道总数最多256。 Atlas A2训练系列产品/Atlas 800I A2推理产品，编码通道号的取值范围：[0, 256)，JPEG功能通道总数最多256。 Atlas 200/500 A2推理产品，编码通道号的取值范围：[0, 128)，JPEG功能和VENC功能共用通道，且通道总数最多128。
status	输出	编码通道状态的指针。

返回值说明

- 0：成功
- 非0：失败，参见[1.9.21.12 VENC视频编码/JPEG图片编码返回码](#)

参考资源

接口调用流程及示例，参见JPEG图片编码、VENC视频编码。

1.9.18.10 hi_mpi_venc_get_stream

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

函数功能

获取编码的码流。

约束说明

- 如果通道未创建，返回失败。
- 如果stream为空，返回错误码HI_ERR_VENC_NULL_PTR。
- 如果milli_sec小于-1，返回错误码HI_ERR_VENC_ILLEGAL_PARAM。
- 支持超时方式获取。支持select/poll系统调用。
 - milli_sec = 0 时，则为非阻塞获取，即如果缓冲无数据，则返回错误码HI_ERR_VENC_BUF_EMPTY。
 - milli_sec = -1 时，则为阻塞，即如果缓冲无数据，则会等待有数据时才返回获取成功。
 - milli_sec > 0 时，则为超时，即如果缓冲无数据，则会等待用户设定的超时时间，若在设定的时间内有数据则返回获取成功，否则返回超时失败。
- 支持按包或按帧方式获取码流。如果按包获取，则：
 - 对于JPEG编码协议，每次获取的是一个参数包或数据包。
 - 对于H.265编码协议，每次获取的是一个 NAL 单元。
- 码流结构体hi_venc_stream包含以下部分：
 - 码流包信息指针pack
指向一组hi_venc_pack结构体的连续的内存空间，该空间由调用者分配。如果是按包获取，则此空间不小于sizeof (hi_venc_pack) 的大小；如果按帧获取，则此空间不小于N*sizeof (hi_venc_pack) 的大小，其中，N代表当前帧之中的包的个数，可以在 select之后通过[hi_mpi_venc_query_status](#)接口获得。
 - 码流包个数pack_cnt
在输入时，此值指定pack中hi_venc_pack结构体内存的个数。按包获取时，pack_cnt必须不小于 1；按帧获取时，pack_cnt必须不小于当前帧的包个数。在函数调用成功后，pack_cnt返回实际填充pack的包的个数。
 - 序号seq按帧获取时是帧序号；按包获取时为包序号。
 - 码流特征信息，数据类型为联合体，包含了不同编码协议对应的码流特征信息h264_info/jpeg_info/h265_info，码流特征信息的输出用于支持用户的上层应用。
 - 码流高级特征信息，数据类型为联合体，包含了不同编码协议对应的码流高级特征信息h264_adv_info/h265_adv_info，码流高级特征信息的输出用于支持用户的上层应用。
- 如果用户长时间不获取码流，码流缓冲区就会满。一个编码通道如果发生码流缓冲区满，就会不再启动编码，直到用户获取码流并释放码流，从而有足够的码流缓冲可以用于编码时，才开始继续编码。
- 建议用户获取码流接口调用与释放码流的接口调用成对出现，且尽快释放码流，防止出现由于用户获取码流，释放不及时而导致的码流buffer满，停止编码。

- 建议用户使用select方式获取码流，且按照如下的流程：
 - 调用hi_mpi_venc_query_status 函数查询编码通道状态；
 - 确保cur_packs和left_stream_frames同时大于0；
 - 调用malloc分配cur_packs个包信息结构体；
 - 调用hi_mpi_venc_get_stream获取编码码流；
 - 调用hi_mpi_venc_release_stream释放码流缓存。
- 当one_stream_buf为1时，用户获取一帧码流时只会得到一个码流包（不包含用户数据）的地址，即pack_cnt为1，地址为pack[0].addr。而在hi_venc_pack结构体中增加offset，用来指出一帧码流中有效数据的地址和pack[0].addr的偏移。如图1-27所示（以 H.264 单包为例），pack[0].data_num = 3,即3种NAL包，分别为SPS、PPS、SEI：
 - pack[0].pack_info[0]. pack_type. h264_type = HI_VENC_H264_NALU_SPS
 - pack[0].pack_info[1]. pack_type. h264_type = HI_VENC_H264_NALU_PPS
 - pack[0].pack_info[2]. pack_type. h264_type = HI_VENC_H264_NALU_SEI其他协议以此类推，data_num和 pack_info只在单包模式才有意义。
- 两种模式可通过调用接口hi_mpi_venc_set_mod_param设置h.265e/h264e/jpeg模块参数one_stream_buf来选择。
 - one_stream_buf=1 表示单包模式，系统默认one_stream_buf=1；注意，JPEGE目前只支持单帧单包模式。
 - one_stream_buf=0 表示非单包模式。

函数原型

```
hi_s32 hi_mpi_venc_get_stream(hi_venc_chn chn, hi_venc_stream *stream,
hi_s32 milli_sec)
```

参数说明

参数名	输入/输出	说明
chn	输入	编码通道号。 Atlas 推理系列产品，编码通道号的取值范围：[0, 256)，JPEGE功能和VENC功能共用通道，且通道总数最多256。 Atlas A2训练系列产品/Atlas 800I A2推理产品，编码通道号的取值范围：[0, 256)，JPEGE功能通道总数最多256。 Atlas 200/500 A2推理产品，编码通道号的取值范围：[0, 128)，JPEGE功能和VENC功能共用通道，且通道总数最多128。

参数名	输入/输出	说明
stream	输出	输出码流数据的指针。 该结构体内的addr参数配置的地址为Device上的内存地址，表示输出的码流存放在Device的内存中。 Atlas 推理系列产品，标准形态下，用户在Host上调用该接口后，需要访问输出码流数据，则需要将Device中的数据传输回Host。 Atlas A2训练系列产品/Atlas 800I A2推理产品，用户在Host上调用该接口后，需要访问输出码流数据，则需要将Device中的数据传输回Host。 Atlas 200/500 A2推理产品，EP模式下，用户在Host上调用该接口后，需要访问输出码流数据，则需要将Device中的数据传输回Host。
milli_sec	输入	超时时间，单位是毫秒。 • -1：阻塞方式 • 0：非阻塞方式 • >0：超时方式，配置具体的超时时间。超时时间受操作系统影响，一般偏差在操作系统的—个时间片内，例如，操作系统的—个时间片为4ms，用户设置的milli_sec参数值为1，则实际的超时时间在1ms到5ms范围内。在CPU负载高场景下，超时时间仍可能存在波动。

返回值说明

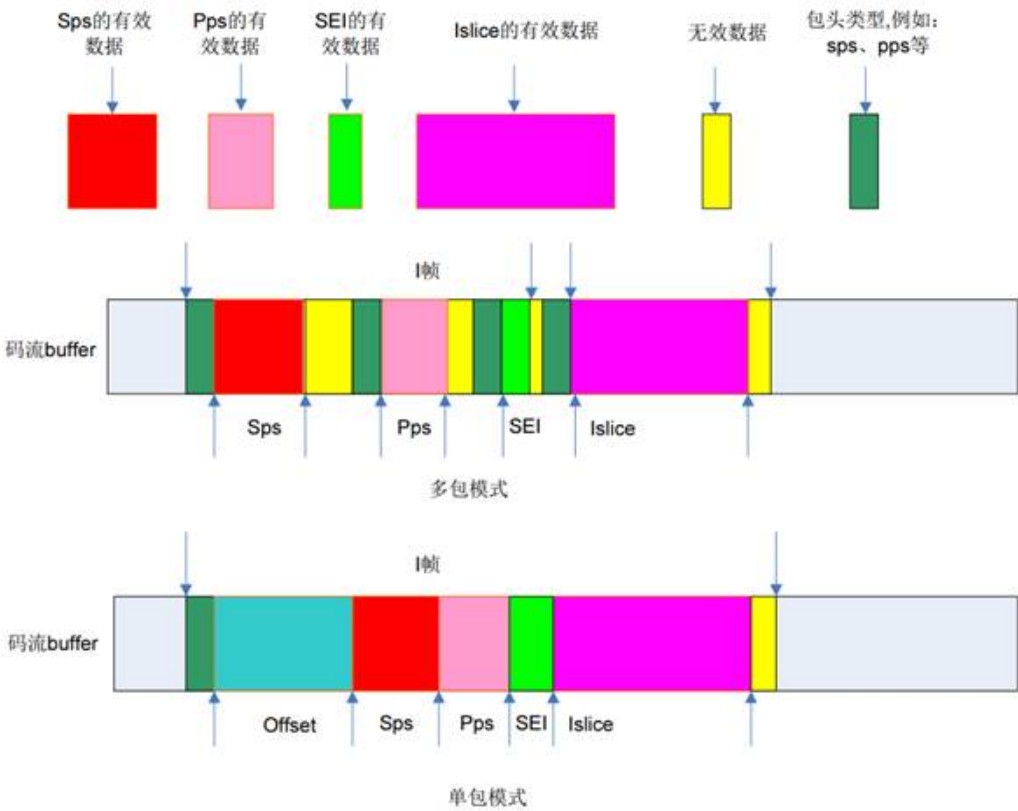
- 0：成功，获取编码结果数据成功后，需调用[hi_mpi_venc_release_stream](#)接口释放码流缓存资源。
本接口虽返回成功，但有可能获取到的是编码失败的数据，编码是否成功，需要查看hi_venc_stream结构体的pack成员下的len参数值，len=0表示编码失败。
- 非0：失败，参见[1.9.21.12 VENC视频编码/JPEG图片编码返回码](#)

参考资源

接口调用流程及示例，参见JPEG图片编码、VENC视频编码。

其它参考信息

图 1-27 码流包结构图



1.9.18.11 hi_mpi_venc_release_stream

Atlas 200/300/500 推理产品，不支持该接口。
Atlas 训练系列产品，不支持该接口。

函数功能

释放码流缓存。

约束说明

- 如果通道未创建，则返回错误码HI_ERR_VENC_UNEXIST。
- 此接口应当和 [hi_mpi_venc_get_stream](#) 配对起来使用，用户获取码流后必须及时释放已经获取的码流缓存，否则可能会导致码流buffer满，影响编码器编码，并且用户必须按先获取先释放的顺序释放已经获取的码流缓存。
- 在编码通道复位以后，所有未释放的码流包均无效，不能再使用或者释放这部分无效的码流缓存。
- 释放无效的码流会返回失败HI_ERR_VENC_ILLEGAL_PARAM。

函数原型

```
hi_s32 hi_mpi_venc_release_stream(hi_venc_chn chn, hi_venc_stream *stream)
```

参数说明

参数名	输入/输出	说明
chn	输入	编码通道号。 Atlas 推理系列产品，编码通道号的取值范围：[0, 256)，JPEG功能 and VENC功能共用通道，且通道总数最多256。 Atlas A2训练系列产品/Atlas 800I A2推理产品，编码通道号的取值范围：[0, 256)，JPEG功能通道总数最多256。 Atlas 200/500 A2推理产品，编码通道号的取值范围：[0, 128)，JPEG功能和VENC功能共用通道，且通道总数最多128。
stream	输入	编码后的输出码流数据的指针。 如果stream为空，则返回错误码HI_ERR_VENC_NULL_PTR。

返回值说明

- 0：成功
- 非0：失败，参见1.9.21.12 VENC视频编码/JPEG图片编码返回码

参考资源

接口调用流程及示例，参见JPEG图片编码、VENC视频编码。

1.9.18.12 hi_mpi_venc_send_frame

Atlas 200/300/500 推理产品，不支持该接口。
Atlas 训练系列产品，不支持该接口。

函数功能

发送原始图像进行视频或图像编码。

约束说明

- 视频输入的原始图像大小必须与编码通道的大小保持一致；JPEG输入的原始图像大小小于或等于编码通道的大小。
- 调用该接口发送图像，用户需要保证编码通道已创建且开启接收输入图片。
- 编码时，输入内存需要在调用hi_mpi_venc_get_stream接口获取结果之后才能进行释放。

- 视频编码，对输入、输出的约束请参见[1.9.18.3 VENC功能及约束说明](#)。
- 图像编码，对输入、输出的约束请参见[1.9.18.1 JPEG功能及约束说明](#)。

函数原型

hi_s32 hi_mpi_venc_send_frame(hi_venc_chn chn, const hi_video_frame_info *frame, hi_s32 milli_sec)

参数说明

参数名	输入/输出	说明
chn	输入	编码通道号。 Atlas 推理系列产品，编码通道号的取值范围：[0, 256)，JPEG功能和VENC功能共用通道，且通道总数最多256。 Atlas A2训练系列产品/Atlas 800I A2推理产品，编码通道号的取值范围：[0, 256)，JPEG功能通道总数最多256。 Atlas 200/500 A2推理产品，编码通道号的取值范围：[0, 128)，JPEG功能和VENC功能共用通道，且通道总数最多128。
frame	输入	原始图像信息的指针。
milli_sec	输入	超时时间，单位是毫秒。 • -1：阻塞方式 • 0：非阻塞方式 • >0：超时方式，配置具体的超时时间。超时时间受操作系统影响，一般偏差在操作系统的片内，例如，操作系统的片为4ms，用户设置的milli_sec参数值为1，则实际的超时时间在1ms到5ms范围内。在CPU负载高场景下，超时时间仍可能存在波动。

返回值说明

- 0：成功
- 非0：失败，参见[1.9.21.12 VENC视频编码/JPEG图片编码返回码](#)

参考资源

接口调用流程及示例，参见JPEG图片编码、VENC视频编码。

1.9.18.13 hi_mpi_venc_set_mod_param

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

设置编码相关的模块参数。

约束说明

此接口在通道创建前调用，如果通道已经创建，则返回错误码HI_ERR_VENC_NOT_PERM。建议用户在调用此接口之前先调用hi_mpi_venc_get_mod_param接口，获取当前通道的编码模块参数配置，然后再根据需求设置相应参数。

函数原型

```
hi_s32 hi_mpi_venc_set_mod_param(const hi_venc_mod_param *mod_param)
```

参数说明

参数名	输入/输出	说明
mod_param	输入	编码模块参数指针。

返回值说明

- 0：成功
- 非0：失败，参见1.9.21.12 VENC视频编码/JPEGE图片编码返回码

参考资源

接口调用流程及示例，参见JPEGE图片编码、VENC视频编码。

1.9.18.14 hi_mpi_venc_get_mod_param

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

获取编码相关的模块参数。

约束说明

建议用户在调用此接口之前先设置hi_venc_mod_param结构体中的mod_type参数，接口会根据不同的模块类型去获取相应模块参数。

函数原型

hi_s32 hi_mpi_venc_get_mod_param(**hi_venc_mod_param** *mod_param)

参数说明

参数名	输入/输出	说明
mod_param	输出	编码模块参数指针。

返回值说明

- 0：成功
- 非0：失败，参见[1.9.21.12 VENC视频编码/JPEG图片编码返回码](#)

参考资源

接口调用流程及示例，参见[JPEG图片编码](#)、[VENC视频编码](#)。

1.9.18.15 hi_mpi_venc_request_idr

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

请求IDR帧。

约束说明

- 如果通道未创建，则返回失败。
- 接受IDR（Instantaneous Decoder Refresh）帧请求后，当instant=0时，则在帧率控制的下一帧编出IDR帧，当instant=1时，则立即编出IDR帧，不受帧率控制约束。
- IDR帧请求，只支持H.264/H.265编码协议。
- 由于目前的使用场景目标帧率与源帧率的值一致，所以此接口不受帧率控制影响。当instant设置为0或1时，都是每调用一次接口即编出一个IDR，调用频繁会影响码流帧率和码率的稳定，使用时需要注意。
- 当GOP模式为Smartp或B帧模式下，请求IDR帧会延时生效。
Atlas 推理系列产品，当前不支持Smartp或B帧模式。
Atlas 200/500 A2推理产品，当前不支持B帧模式。

函数原型

hi_s32 hi_mpi_venc_request_idr(**hi_venc_chn** chn, **hi_bool** instant)

参数说明

参数名	输入/输出	说明
chn	输入	编码通道号。 Atlas 推理系列产品，编码通道号的取值范围：[0, 256)，JPEGE功能和VENC功能共用通道，且通道总数最多256。 Atlas 200/500 A2推理产品，编码通道号的取值范围：[0, 128)，JPEGE功能和VENC功能共用通道，且通道总数最多128。
instant	输入	是否使能立即编码IDR帧。

返回值说明

- 0：成功
- 非0：失败，参见[1.9.21.12 VENC视频编码/JPEGE图片编码返回码](#)

1.9.18.16 hi_mpi_venc_get_fd

Atlas 200/300/500 推理产品，不支持该接口。
Atlas 训练系列产品，不支持该接口。

函数功能

获取编码通道对应的文件句柄。

约束说明

在进程退出时会自动关闭文件句柄，回收资源。

函数原型

[hi_s32](#) hi_mpi_venc_get_fd([hi_venc_chn](#) chn)

参数说明

参数名	输入/输出	说明
chn	输入	编码通道号。 Atlas 推理系列产品，编码通道号的取值范围：[0, 256)，JPEG功能 and VENC功能共用通道，且通道总数最多256。 Atlas A2训练系列产品/Atlas 800I A2推理产品，编码通道号的取值范围：[0, 256)，JPEG功能通道总数最多256。 Atlas 200/500 A2推理产品，编码通道号的取值范围：[0, 128)，JPEG功能和VENC功能共用通道，且通道总数最多128。

返回值说明

- 0或正数值：有效返回值
- 非正数值：无效返回值

参考资源

接口调用流程及示例，参见JPEG图片编码、VENC视频编码。

1.9.18.17 hi_mpi_venc_close_fd

函数功能

关闭编码通道对应的设备文件句柄。
当前版本不支持该接口。

约束说明

用户可以不调用本接口，在应用程序执行完成，对应的进程退出后，系统会自动清理句柄；如果用户主动调用本接口，必须确保在调用[hi_mpi_venc_destroy_chn](#)接口之后再调用。此接口不能与其它 MPI 接口同时调用，用户必须保证此接口与其它接口在时间上是串行调用的。

函数原型

[hi_s32](#) hi_mpi_venc_close_fd([hi_venc_chn](#) chn)

参数说明

参数名	输入/输出	说明
chn	输入	编码通道号。

返回值说明

- 0：关闭成功
- 非0：失败，参见[1.9.21.12 VENC视频编码/JPEGE图片编码返回码](#)

1.9.18.18 hi_mpi_venc_set_jpeg_param

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

函数功能

设置JPEG协议编码通道的量化表高级配置参数，用户需要调整JPEGE量化参数表时才需要使用该接口。

约束说明

本接口在编码通道创建之后、编码通道销毁之前调用。

用户可以先调用[hi_mpi_venc_get_jpeg_param](#)获取当前参数，然后对该参数进行修改后再调用本接口设置。当hi_venc_jpeg_param结构体内的qfactor参数值配置为0xFFFFFFFF，QT（Quantisation Table）表的Cb分量（Cb表示蓝色分量）和Cr分量（Cr表示红色分量）需保持一致。

函数原型

[hi_s32](#) hi_mpi_venc_set_jpeg_param([hi_venc_chn](#) chn, const [hi_venc_jpeg_param](#) *jpeg_param)

参数说明

参数名	输入/输出	说明
chn	输入	编码通道号。 Atlas 推理系列产品，编码通道号的取值范围：[0, 256)，JPEGE功能和VENC功能共用通道，且通道总数最多256。 Atlas A2训练系列产品/Atlas 800I A2推理产品，编码通道号的取值范围：[0, 256)，JPEGE功能通道总数最多256。 Atlas 200/500 A2推理产品，编码通道号的取值范围：[0, 128)，JPEGE功能和VENC功能共用通道，且通道总数最多128。
jpeg_param	输入	JPEG协议编码通道的高级参数集合的指针。

返回值说明

- 0：成功
- 非0：失败，参见[1.9.21.12 VENC视频编码/JPEGE图片编码返回码](#)

1.9.18.19 hi_mpi_venc_get_jpeg_param

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

函数功能

获取JPEG协议编码通道的量化表高级配置参数。

约束说明

本接口在编码通道创建之后、编码通道销毁之前调用。

调用[hi_mpi_venc_set_jpeg_param](#)接口设置参数前，可以调用该接口获取当前JPEGE的量化表参数。

函数原型

hi_s32 hi_mpi_venc_get_jpeg_param(**hi_venc_chn** chn, **hi_venc_jpeg_param** *jpeg_param)

参数说明

参数名	输入/输出	说明
chn	输入	编码通道号。 Atlas 推理系列产品，编码通道号的取值范围：[0, 256)，JPEGE功能和VENC功能共用通道，且通道总数最多256。 Atlas A2训练系列产品/Atlas 800I A2推理产品，编码通道号的取值范围：[0, 256)，JPEGE功能通道总数最多256。 Atlas 200/500 A2推理产品，编码通道号的取值范围：[0, 128)，JPEGE功能和VENC功能共用通道，且通道总数最多128。
jpeg_param	输出	JPEG协议编码通道的高级参数集合的指针。

返回值说明

- 0：成功
- 非0：失败，参见[1.9.21.12 VENC视频编码/JPEGE图片编码返回码](#)

1.9.18.20 hi_mpi_venc_set_chn_param

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

设置通道参数。

约束说明

- 本接口在编码通道创建之后、编码通道销毁之前调用。如果针对未创建的通道，调用本接口设置参数，则返回失败。
- Atlas 200/500 A2推理产品，JPEG图片编码场景下，不支持调用本接口。

函数原型

```
hi_s32 hi_mpi_venc_set_chn_param(hi_venc_chn chn, const hi_venc_chn_param *chn_param)
```

参数说明

参数名	输入/输出	说明
chn	输入	编码通道号。 Atlas 推理系列产品，编码通道号的取值范围：[0, 256)，JPEG功能和VENC功能共用通道，且通道总数最多256。 Atlas 200/500 A2推理产品，编码通道号的取值范围：[0, 128)，JPEG功能和VENC功能共用通道，且通道总数最多128。
chn_param	输入	VENC通道参数的指针。

返回值说明

- 0：成功
- 非0：失败，参见[1.9.21.12 VENC视频编码/JPEG图片编码返回码](#)

1.9.18.21 hi_mpi_venc_get_chn_param

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

获取通道参数。

约束说明

- 本接口在编码通道创建之后、编码通道销毁之前调用。如果针对未创建的通道，调用本接口获取参数，则返回失败。
- Atlas 200/500 A2推理产品，JPEG图片编码场景下，不支持调用本接口。

函数原型

hi_s32 hi_mpi_venc_get_chn_param(hi_venc_chn chn, hi_venc_chn_param *chn_param)

参数说明

参数名	输入/输出	说明
chn	输入	编码通道号。 Atlas 推理系列产品，编码通道号的取值范围：[0, 256)，JPEG功能和VENC功能共用通道，且通道总数最多256。 Atlas 200/500 A2推理产品，编码通道号的取值范围：[0, 128)，JPEG功能和VENC功能共用通道，且通道总数最多128。
chn_param	输出	VENC通道参数的指针。

返回值说明

- 0：成功
- 非0：失败，参见[1.9.21.12 VENC视频编码/JPEG图片编码返回码](#)

1.9.18.22 hi_mpi_venc_set_scene_mode

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

设置场景模式。

约束说明

- 本接口在编码通道创建之后、编码通道销毁之前调用。
- 不同的编码模式场景对图像质量有影响，默认场景模式为HI_VENC_SCENE_0。

函数原型

hi_s32 hi_mpi_venc_set_scene_mode(**hi_venc_chn** chn, const **hi_venc_scene_mode** scene_mode)

参数说明

参数名	输入/输出	说明
chn	输入	编码通道号。 Atlas 推理系列产品，编码通道号的取值范围：[0, 256)，JPEG功能 and VENC功能共用通道，且通道总数最多256。 Atlas 200/500 A2推理产品，编码通道号的取值范围：[0, 128)，JPEG功能和VENC功能共用通道，且通道总数最多128。
scene_mode	输入	场景模式。

返回值说明

- 0：成功
- 非0：失败，参见[1.9.21.12 VENC视频编码/JPEG图片编码返回码](#)

1.9.18.23 hi_mpi_venc_get_scene_mode

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

获取场景模式。

约束说明

本接口在编码通道创建之后、编码通道销毁之前调用。

函数原型

hi_s32 hi_mpi_venc_get_scene_mode(**hi_venc_chn** chn, **hi_venc_scene_mode** *scene_mode)

参数说明

参数名	输入/输出	说明
chn	输入	编码通道号。 Atlas 推理系列产品，编码通道号的取值范围：[0, 256)，JPEG功能 and VENC功能共用通道，且通道总数最多256。 Atlas 200/500 A2推理产品，编码通道号的取值范围：[0, 128)，JPEG功能 and VENC功能共用通道，且通道总数最多128。
scene_mode	输出	场景模式指针。

返回值说明

- 0：成功
- 非0：失败，参见[1.9.21.12 VENC视频编码/JPEG图片编码返回码](#)

1.9.18.24 hi_mpi_venc_set_rc_param

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

设置视频编码通道码率控制器的高级参数的值。

约束说明

- 本接口在编码通道创建之后、编码通道销毁之前调用。
- 如果不调用本接口设置编码通道码率控制器的高级参数的值，就会默认采用这些高级参数的默认值。
- 建议用户先调用[hi_mpi_venc_get_rc_param](#)接口获取编码通道码率控制器的高级参数的值，再调用本接口对高级参数进行设置。

函数原型

```
hi_s32 hi_mpi_venc_set_rc_param(hi_venc_chn chn, const hi_venc_rc_param *rc_param)
```

参数说明

参数名	输入/输出	说明
chn	输入	编码通道号。 Atlas 推理系列产品，编码通道号的取值范围：[0, 256)，JPEG功能 and VENC功能共用通道，且通道总数最多256。 Atlas 200/500 A2推理产品，编码通道号的取值范围：[0, 128)，JPEG功能 and VENC功能共用通道，且通道总数最多128。
rc_param	输入	编码通道码率控制器的高级参数的指针。

返回值说明

- 0：成功
- 非0：失败，参见[1.9.21.12 VENC视频编码/JPEG图片编码返回码](#)

1.9.18.25 hi_mpi_venc_get_rc_param

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

获取视频编码通道码率控制器的高级参数的值。

约束说明

本接口在编码通道创建之后、编码通道销毁之前调用。

函数原型

```
hi_s32 hi_mpi_venc_get_rc_param(hi_venc_chn chn, hi_venc_rc_param *rc_param)
```

参数说明

参数名	输入/输出	说明
chn	输入	编码通道号。 Atlas 推理系列产品，编码通道号的取值范围：[0, 256)，JPEGE功能和VENC功能共用通道，且通道总数最多256。 Atlas 200/500 A2推理产品，编码通道号的取值范围：[0, 128)，JPEGE功能和VENC功能共用通道，且通道总数最多128。
rc_param	输出	编码通道码率控制器的高级参数的指针。

返回值说明

- 0：成功
- 非0：失败，参见[1.9.21.12 VENC视频编码/JPEGE图片编码返回码](#)

1.9.18.26 hi_mpi_venc_set_jpeg_huffman_param

Atlas 200/300/500 推理产品，不支持该接口。
Atlas 训练系列产品，不支持该接口。

函数功能

设置编码通道的JPEGE huffman编码高级参数。

约束说明

本接口在编码通道创建之后、编码通道销毁之前调用。

函数原型

[hi_s32](#) hi_mpi_venc_set_jpeg_huffman_param([hi_venc_chn](#) chn, const [hi_venc_jpeg_huffman_param](#) *jpeg_huffman_param)

参数说明

参数名	输入/输出	说明
chn	输入	编码通道号。 Atlas 推理系列产品，编码通道号的取值范围：[0, 256)，JPEG功能 and VENC功能共用通道，且通道总数最多256。 Atlas A2训练系列产品/Atlas 800I A2推理产品，编码通道号的取值范围：[0, 256)，JPEG功能通道总数最多256。 Atlas 200/500 A2推理产品，编码通道号的取值范围：[0, 128)，JPEG功能和VENC功能共用通道，且通道总数最多128。
jpeg_huffman_param	输入	huffman编码参数的指针。

返回值说明

- 0：成功
- 非0：失败，参见[1.9.21.12 VENC视频编码/JPEG图片编码返回码](#)

1.9.18.27 hi_mpi_venc_get_jpeg_huffman_param

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

函数功能

获取编码通道的JPEG huffman编码高级参数。

约束说明

- 本接口在编码通道创建之后、编码通道销毁之前调用。
- 不调用[hi_mpi_venc_set_jpeg_huffman_param](#)接口设置huffman编码高级参数时，返回默认的标准huffman编码参数。

函数原型

[hi_s32](#) hi_mpi_venc_get_jpeg_huffman_param([hi_venc_chn](#) chn,
[hi_venc_jpeg_huffman_param](#) *jpeg_huffman_data)

参数说明

参数名	输入/输出	说明
chn	输入	编码通道号。 Atlas 推理系列产品，编码通道号的取值范围：[0, 256)，JPEG功能 and VENC功能共用通道，且通道总数最多256。 Atlas A2训练系列产品/Atlas 800I A2推理产品，编码通道号的取值范围：[0, 256)，JPEG功能通道总数最多256。 Atlas 200/500 A2推理产品，编码通道号的取值范围：[0, 128)，JPEG功能和VENC功能共用通道，且通道总数最多128。
jpeg_huffman_data	输出	huffman高级参数的指针。

返回值说明

- 0：成功
- 非0：失败，参见[1.9.21.12 VENC视频编码/JPEG图片编码返回码](#)

1.9.18.28 hi_mpi_venc_compact_jpeg_tables

Atlas 200/300/500 推理产品，不支持该接口。
Atlas 训练系列产品，不支持该接口。

函数功能

配置是否对高级参数huffman或QT表进行压缩。

约束说明

本接口在编码通道创建之后、编码通道销毁之前调用。
调用此接口后，如果huffman表或QT表的Y、Cb、Cr分量内容一致时，支持压缩。

函数原型

```
hi_s32 hi_mpi_venc_compact_jpeg_tables(hi_venc_chn chn, hi_u32 table_type, hi_bool enable)
```

参数说明

参数名	输入/输出	说明
chn	输入	编码通道号。 Atlas 推理系列产品，编码通道号的取值范围：[0, 256)，JPEGE功能和VENC功能共用通道，且通道总数最多256。 Atlas A2训练系列产品/Atlas 800I A2推理产品，编码通道号的取值范围：[0, 256)，JPEGE功能通道总数最多256。 Atlas 200/500 A2推理产品，编码通道号的取值范围：[0, 128)，JPEGE功能和VENC功能共用通道，且通道总数最多128。
table_type	输入	压缩类型，取值范围：仅支持0，表示huffman表数据和QT量化表数据都压缩。
enable	输入	是否压缩，0表示不压缩，1表示压缩。

返回值说明

- 0：成功
- 非0：失败，参见[1.9.21.12 VENC视频编码/JPEGE图片编码返回码](#)

1.9.18.29 hi_mpi_venc_send_jpege_frame

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

函数功能

发送原始图像进行图像编码，并支持配置编码结果的输出内存地址。

约束说明

- 在调用[hi_mpi_venc_create_chn](#)接口创建通道时，将参数hi_venc_chn_attr.venc_attr.buf_size参数值设置为0，本接口才有效，否则返回错误码HI_ERR_VENC_NOT_SUPPORT。
- JPEGE输入图像分辨率必须小于或等于编码通道的宽、高。
- 调用该接口发送图像，用户需要保证编码通道已创建且开启接收输入图片。
- 编码时，输入、输出内存均需要在调用[hi_mpi_venc_get_stream](#)接口获取结果之后才能进行释放。
- 图像编码，对输入、输出的约束请参见[1.9.18.1 JPEGE功能及约束说明](#)。

函数原型

```
hi_s32 hi_mpi_venc_send_jpege_frame(hi_venc_chn chn, const  
hi_video_frame_info *frame, const hi_img_stream* jpege_stream, hi_s32  
milli_sec)
```

参数说明

参数名	输入/输出	说明
chn	输入	编码通道号。 Atlas 推理系列产品，编码通道号的取值范围：[0, 256)，JPEG功能 and VENC功能共用通道，且通道总数最多256。 Atlas A2训练系列产品/Atlas 800I A2推理产品，编码通道号的取值范围：[0, 256)，JPEG功能通道总数最多256。 Atlas 200/500 A2推理产品，编码通道号的取值范围：[0, 128)，JPEG功能和VENC功能共用通道，且通道总数最多128。
frame	输入	原始图像信息的指针。
jpege_stream	输入	用于指定编码结果的输出内存地址和长度。 输出内存首地址需满足16字节对齐，内存大小可以调用hi_mpi_venc_get_jpege_predicted_size获取，再可调用hi_mpi_dvpp_malloc接口申请输出内存。
milli_sec	输入	超时时间，单位是毫秒。 -1：阻塞方式 0：非阻塞方式 >0：超时方式，配置具体的超时时间。超时时间受操作系统影响，一般偏差在操作系统的片内，例如，操作系统的片为4ms，用户设置的milli_sec参数值为1，则实际的超时时间在1ms到5ms范围内。在CPU负载高场景下，超时时间仍可能存在波动。

返回值说明

- 0：成功
- 非0：失败，参见[1.9.21.12 VENC视频编码/JPEG图片编码返回码](#)

参考资源

接口调用流程及示例，参见JPEG图片编码、VENC视频编码。

1.9.18.30 hi_mpi_venc_get_jpege_predicted_size

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

函数功能

预估JPEG编码一帧图片所需的输出缓冲区大小。

函数原型

```
hi_s32 hi_mpi_venc_get_jpege_predicted_size(const hi_video_frame_info*
frame, const hi_venc_jpeg_param* jpeg_param, hi_u32 *size)
```

参数说明

参数名	输入/输出	说明
frame	输入	原始图像信息的指针。
jpeg_param	输入	JPEG协议编码通道的高级参数集合的指针，预留参数。
size	输出	输出缓冲区大小的指针，单位为Byte。

返回值说明

- 0：成功
- 非0：失败，参见[1.9.21.12 VENC视频编码/JPEG图片编码返回码](#)

1.9.18.31 hi_mpi_venc_set_roi_attr

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

设置H.264/H.265通道的ROI属性，用户可以通过配置ROI区域，对该区域的图像Qp进行限制，从而实现图像中该区域的Qp与其他图像区域的差异化。关于ROI编码功能的详细说明请参见[ROI编码功能说明](#)。

约束说明

- 本接口用于设置H.264/H.265通道的ROI属性。
- 本接口在编码通道创建之后，编码通道销毁之前设置。此接口在编码过程中被调用时，等到下一个帧时生效。
- 建议用户在创建通道之后，启动编码之前调用此接口，减少在编码过程中调用的次数。

- 建议用户在调用此接口之前先调用[hi_mpi_venc_get_roi_attr](#)接口，获取当前通道的ROI配置，然后再进行设置。
- 设置该接口后，如果当前帧判断编码为pskip 帧，以pskip帧效果优先。
- 设置该接口后，可能会出现感兴趣区域QP值不全为设置值的现象；这是因为在编码过程中，有些编码块的残差会为0，没有数据需要被量化，QP值不生效所致，属于正常现象，不影响编码效果。

函数原型

hi_s32 [hi_mpi_venc_set_roi_attr](#)([hi_venc_chn](#) chn, const [hi_venc_roi_attr](#) *roi_attr)

参数说明

参数名	输入/输出	说明
chn	输入	编码通道号。 Atlas 推理系列产品，编码通道号的取值范围：[0, 256)，JPEGE功能和VENC功能共用通道，且通道总数最多256。 Atlas 200/500 A2推理产品，编码通道号的取值范围：[0, 128)，JPEGE功能和VENC功能共用通道，且通道总数最多128。
roi_attr	输入	ROI 区域参数指针。

返回值说明

- 0：成功
- 非0：失败，参见[1.9.21.12 VENC视频编码/JPEGE图片编码返回码](#)

ROI 编码功能说明

ROI (Region Of Interest) 编码：对感兴趣区域编码。

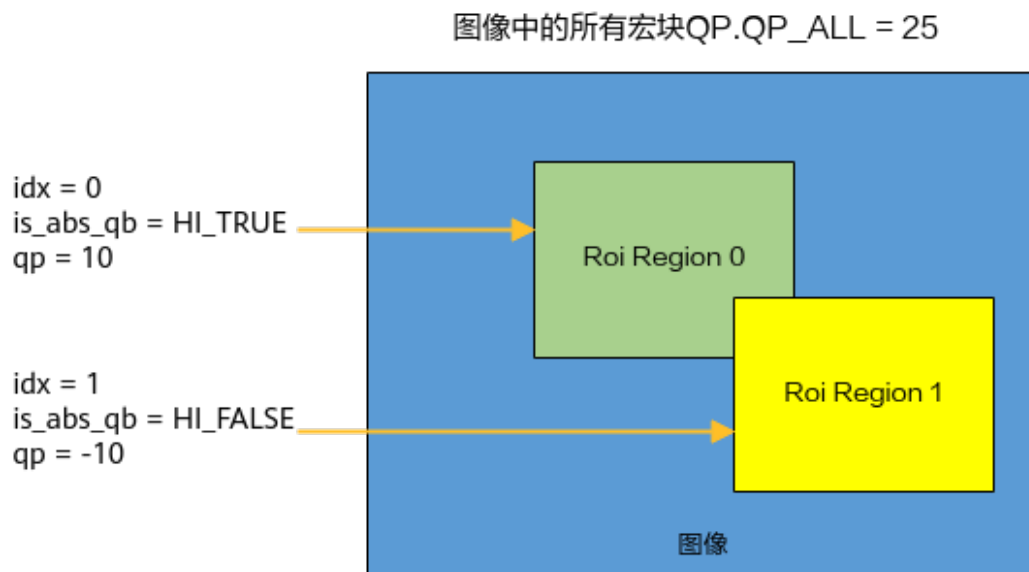
用户可以通过配置ROI区域，对该区域的图像Qp进行限制，从而实现图像中该区域的Qp与其它图像区域的差异化。系统现仅支持对H.264/H.265通道进行ROI设置。系统提供了8个感兴趣区域，可供用户同时使用。

8个区域可以互相叠加，且叠加时的优先级按照0～7的索引号依次提高，这里，叠加优先级是指发生叠加时，图像区域的最终Qp值的判定，最终的区域Qp值按照优先级最高的区域设定。ROI区域可配置绝对Qp和相对Qp两种模式。

- 绝对Qp：ROI区域的Qp为用户设定的Qp 值。
- 相对Qp：ROI区域的Qp为码率控制产生的Qp与用户设定的Qp偏移值的和。

以下图为例，假如图像Qp为25，即图像中所有宏块Qp值为25。Roi区域0设置为绝对Qp模式，Qp值为10，索引为0；Roi区域1设置为相对Qp模式，Qp为-10，索引为1。区域0的index小于区域1的index，所以在发生互相重叠的图像区域按高优先级的区域

(区域1) Qp设置。区域0除了发生重叠的部分的Qp值等于10。区域1 的Qp值为25-10=15。



1.9.18.32 hi_mpi_venc_get_roi_attr

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

获取H.264/H.265通道的ROI配置属性。关于ROI编码功能的详细说明请参见[ROI编码功能说明](#)。

约束说明

- 本接口用于获取H.264/H.265协议编码通道的指定索引号的ROI配置。
- 本接口在编码通道创建之后，编码通道销毁之前调用。
- 建议用户在创建通道之后，启动编码之前调用此接口，减少在编码过程中调用的次数。

函数原型

```
hi_s32 hi_mpi_venc_get_roi_attr(hi_venc_chn chn, hi_u32 idx, hi_venc_roi_attr *roi_attr)
```

参数说明

参数名	输入/输出	说明
chn	输入	编码通道号。 Atlas 推理系列产品，编码通道号的取值范围：[0, 256)，JPEGE功能和VENC功能共用通道，且通道总数最多256。 Atlas 200/500 A2推理产品，编码通道号的取值范围：[0, 128)，JPEGE功能和VENC功能共用通道，且通道总数最多128。
idx	输入	H.264/H.265协议编码通道ROI区域索引。
roi_attr	输出	ROI区域参数指针。

返回值说明

- 0：成功
- 非0：失败，参见[1.9.21.12 VENC视频编码/JPEGE图片编码返回码](#)

1.9.18.33 hi_mpi_venc_set_ref_param

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

设置H.264/H.265编码通道高级跳帧参考参数，用于改变帧间参考关系。关于高级跳帧模式的详细说明请参见[高级跳帧模式说明](#)。

约束说明

- 创建H.264/H.265协议编码通道时，默认跳帧参考模式是1倍跳帧参考模式。如果用户需要修改编码通道的跳帧参考，建议在创建通道之后，启动编码之前调用此接口，减少在编码过程中调用的次数。
- 本接口在编码过程中被调用时，等到下一个I帧时生效。
- 如果用户设置1倍跳帧参考模式，对应的配置为：pred_en=HI_TRUE，enhance=0，base =1；如果设置2倍跳帧参考模式，对应的配置为：pred_en=HI_TRUE，enhance=1，base = 1；如果设置 4 倍跳帧参考模式，对应的配置为：pred_en =HI_TRUE，enhance = 1，base = 2。
- 如果用户设置通道的Gop Mode为HI_VENC_GOP_MODE_DUAL_P时，sp_interval只支持设置为 $sp\_interval = (enhance+1) * base$ ，其中，当sp_interval 不为0时，pred_en只支持设置为 HI_TRUE。
- 如果用户设置通道的Gop Mode为HI_VENC_GOP_MODE_SMART_P时，pred_en只支持设置为HI_TRUE。

- 如果用户设置通道的Gop Mode为HI_VENC_GOP_MODE_BIPREDB时，enhance必须等于b_frame_num。

函数原型

```
hi_s32 hi_mpi_venc_set_ref_param(hi_venc_chn chn, const hi_venc_ref_param *ref_param)
```

参数说明

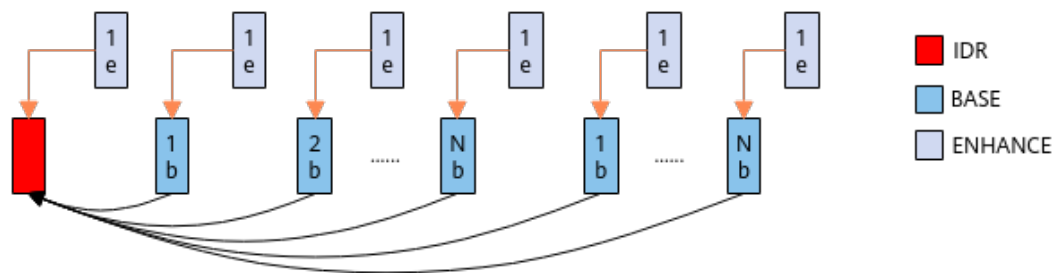
参数名	输入/输出	说明
chn	输入	编码通道号。 Atlas 推理系列产品，编码通道号的取值范围：[0, 256)，JPEGE功能和VENC功能共用通道，且通道总数最多256。 Atlas 200/500 A2推理产品，编码通道号的取值范围：[0, 128)，JPEGE功能和VENC功能共用通道，且通道总数最多128。
ref_param	输入	H.264/H.265编码通道高级跳帧参考参数指针。

返回值说明

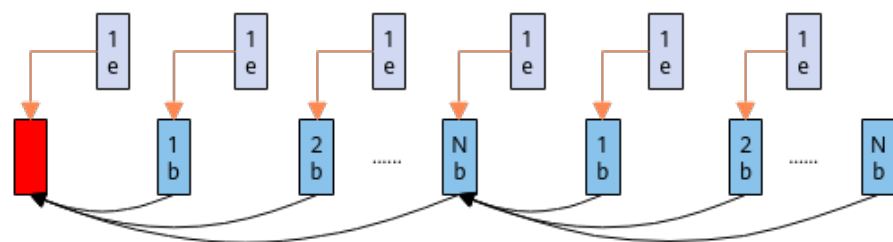
- 0：成功
- 非0：失败，参见1.9.21.12 VENC视频编码/JPEGE图片编码返回码

高级跳帧模式说明

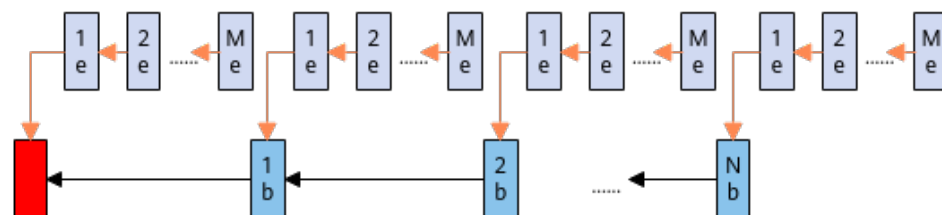
- 高级跳帧参考模式涉及3 个参数：base、enhance 和pred_en，其含义请参考hi_venc_ref_param。高级跳帧参考模式示意图如下图所示。
- 注意：dual_p 模式且sp_interval不等于0下配置高级跳帧参考需要满足如下条件： $base \times (enhance + 1) = sp\_interval$ ，例如：4 倍高级跳帧参考（base=2，enhance=1），那么dual_p 的sp_interval = $2 \times (1 + 1) = 4$ 。
 - normal_p高级跳帧参考示意图



(base = N, enhance = 1, pred_en = HI_FALSE)

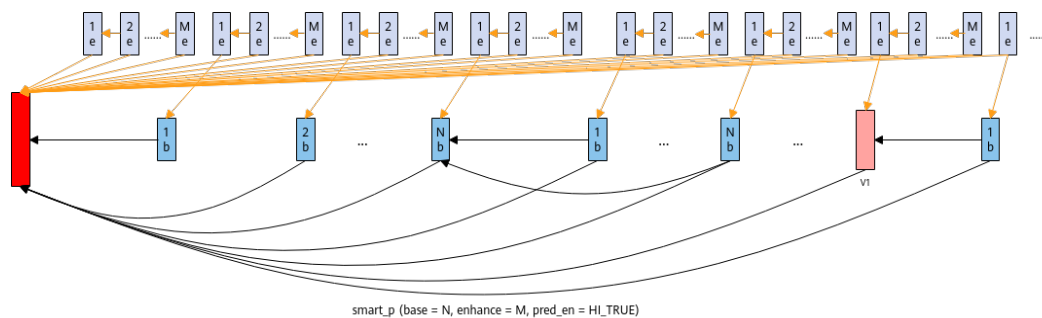


(base = N, enhance = 1, pred_en = HI_TRUE)

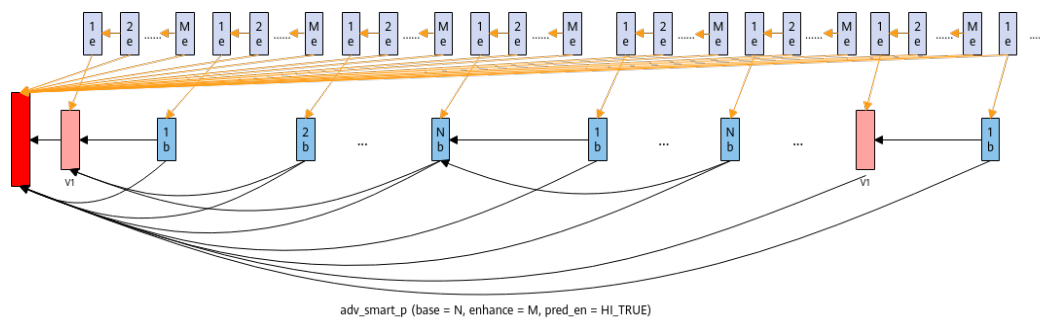


(base = 1, enhance = M, pred_en = HI_TRUE)

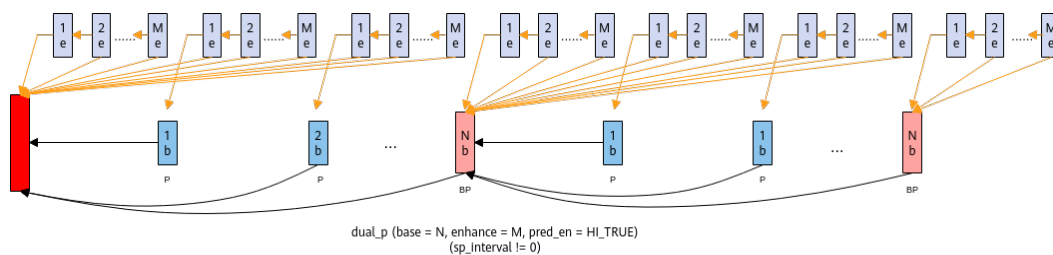
- smart_p高级跳帧参考示意图



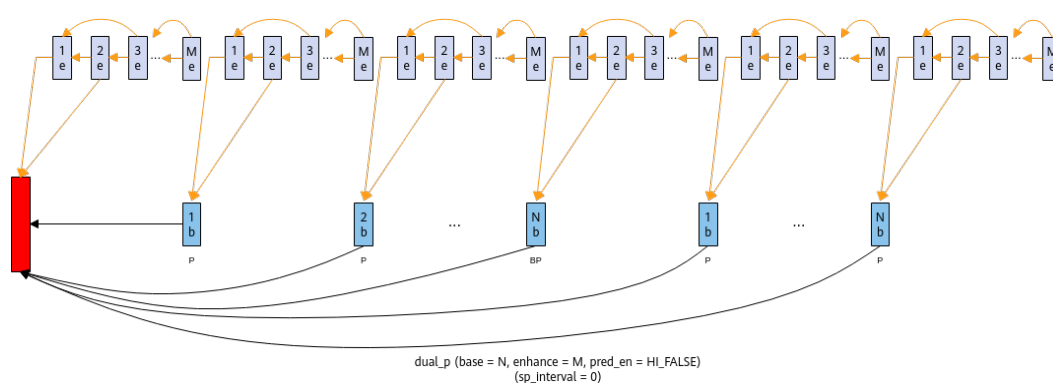
- adv_smart_p高级跳帧参考示意图



• dual_p (sp_interval!=0)高级跳帧参考示意图



• dual_p (sp_interval=0)高级跳帧参考示意图



1.9.18.34 hi_mpi_venc_get_ref_param

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

获取H.264/H.265编码通道高级跳帧参考参数。关于高级跳帧模式的详细说明请参见[高级跳帧模式说明](#)。

约束说明

本接口在编码通道创建之后、编码通道销毁之前调用。

函数原型

```
hi_s32 hi_mpi_venc_get_ref_param(hi_venc_chn chn, hi_venc_ref_param
*ref_param)
```

参数说明

参数名	输入/输出	说明
chn	输入	编码通道号。 Atlas 推理系列产品，编码通道号的取值范围：[0, 256)，JPEGE功能和VENC功能共用通道，且通道总数最多256。 Atlas 200/500 A2推理产品，编码通道号的取值范围：[0, 128)，JPEGE功能和VENC功能共用通道，且通道总数最多128。
ref_param	输出	H.264/H.265编码通道高级跳帧参考参数指针。

返回值说明

- 0：成功
- 非0：失败，参见[1.9.21.12 VENC视频编码/JPEGE图片编码返回码](#)

1.9.18.35 hi_mpi_venc_set_h264_vui

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

设置H.264协议编码通道的Vui参数。

Vui（ Video usability information ）参数主要包含解码图像的格式信息，比如采样横纵比、光电转换特性、颜色空间等。Vui参数在协议中属于可选参数，不影响视频编码过程，但会建议解码器/播放器做一些矫正处理。

约束说明

- 本接口用于设置H.264协议编码通道的Vui配置。
- 本接口在编码通道创建之后，编码通道销毁之前设置。此接口在编码过程中被调用时，等到下一个I帧时生效。
- 建议用户在创建通道之后，启动编码之前调用此接口，减少在编码过程中调用的次数。
- 用户应当仅在H.264协议编码通道中调用本接口。
- 目前仅开放hi_venc_h264_vui中的hi_venc_vui_video_signal参数和hi_venc_vui_h264_time_info参数供用户配置，其它参数预留，均需配置为默认

值，否则会导致编码失败。建议用户在调用此接口之前先调用 [hi_mpi_venc_get_h264_vui](#)接口，获取当前编码通道的Vui默认配置，然后再进行设置，避免预留参数出错。

- [hi_venc_vui_h264_time_info](#)参数包含时钟计数相关参数，默认情况下为不生效状态，用户想要控制显示帧率时需要主动配置相关参数。注意，显示帧率只影响播放器的播放速度，若用户想设置编码过程中的码控帧率需配置 [hi_mpi_venc_create_chn](#)接口中的输入/输出帧率参数。
建议用户将码控帧率与显示帧率设置一致，否则播放时的码率可能与用户设置的编码码率不同，播放时的码率会随着所设置的显示帧率而变化。
如果用户需要调整播放速度，则建议在合理范围内调整显示帧率，一般正常视频播放速度为30fps，表示显示帧率为30fps，若将显示帧率调大，则播放速度加快，例如3倍速播放，则显示帧率设置为90fps；若将显示帧率调小，则播放速度减慢，例如0.5倍速播放，则显示帧率设置为15fps。
- 由于Vui只是协议可选参数而非强制参数，调用本接口设置参数（如显示帧率、像素值域范围等）的最终效果取决于解码器/播放器行为，或者用户自定义的解码/播放行为。

函数原型

```
hi_s32 hi_mpi_venc_set_h264_vui(hi_venc_chn chn, const hi_venc_h264_vui *h264_vui)
```

参数说明

参数名	输入/输出	说明
chn	输入	编码通道号。 Atlas 推理系列产品，编码通道号的取值范围：[0, 256)，JPEGE功能和VENC功能共用通道，且通道总数最多256。 Atlas 200/500 A2推理产品，编码通道号的取值范围：[0, 128)，JPEGE功能和VENC功能共用通道，且通道总数最多128。
h264_vui	输入	H.264协议编码通道的Vui参数。

返回值说明

- 0：成功
- 非0：失败，参见[1.9.21.12 VENC视频编码/JPEGE图片编码返回码](#)

1.9.18.36 hi_mpi_venc_get_h264_vui

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

获取H.264协议编码通道的Vui配置。

约束说明

- 本接口用于获取H.264协议编码通道的Vui配置。
- 本接口在编码通道创建之后，编码通道销毁之前调用。
- 建议用户在创建通道之后，启动编码之前调用此接口，减少在编码过程中调用的次数。
- 用户应当仅在H.264协议编码通道中调用本接口。

函数原型

```
hi_s32 hi_mpi_venc_get_h264_vui(hi_venc_chn chn, hi_venc_h264_vui
*h264_vui)
```

参数说明

参数名	输入/输出	说明
chn	输入	编码通道号。 Atlas 推理系列产品，编码通道号的取值范围：[0, 256)，JPEG功能和VENC功能共用通道，且通道总数最多256。 Atlas 200/500 A2推理产品，编码通道号的取值范围：[0, 128)，JPEG功能和VENC功能共用通道，且通道总数最多128。
h264_vui	输出	H.264协议编码通道的Vui属性。

返回值说明

- 0：成功
- 非0：失败，参见[1.9.21.12 VENC视频编码/JPEG图片编码返回码](#)

1.9.18.37 hi_mpi_venc_set_h265_vui

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

设置H.265协议编码通道的Vui参数。

Vui (Video usability information) 参数主要包含解码图像的格式信息，比如采样横纵比、光电转换特性、颜色空间等。Vui参数在协议中属于可选参数，不影响视频编码过程，但会建议解码器/播放器做一些矫正处理。

约束说明

- 本接口用于设置H.265协议编码通道的Vui配置。
- 本接口在编码通道创建之后，编码通道销毁之前设置。此接口在编码过程中被调用时，等到下一个帧时生效。
- 建议用户在创建通道之后，启动编码之前调用此接口，减少在编码过程中调用的次数。
- 用户应当仅在H.265协议编码通道中调用本接口。
- 目前仅开放hi_venc_h265_vui中的hi_venc_vui_video_signal参数和hi_venc_vui_h265_time_info参数供用户配置，其它参数预留，均需配置为默认值，否则会导致编码失败。建议用户在调用此接口之前先调用[hi_mpi_venc_get_h265_vui](#)接口，获取当前编码通道的Vui默认配置，然后再进行设置，避免预留参数出错。
- hi_venc_vui_h265_time_info参数包含时钟计数相关参数，默认情况下为不生效状态，用户想要控制显示帧率时需要主动配置相关参数。注意，此处的显示帧率只影响播放器的播放速度，用户想控制编码过程中的码控帧率需配置[hi_mpi_venc_create_chn](#)接口中的hi_venc_rc_attr参数。
建议用户将码控帧率与显示帧率设置一致，否则播放时的码率可能与用户设置的编码码率不同，播放时的码率会随着所设置的显示帧率而变化。
如果用户需要调整播放速度，则建议在合理范围内调整显示帧率，一般正常视频播放速度为30fps，表示显示帧率为30fps，若将显示帧率调大，则播放速度加快，例如3倍速播放，则显示帧率设置为90fps；若将显示帧率调小，则播放速度减慢，例如0.5倍速播放，则显示帧率设置为15fps。
- 由于Vui只是协议可选参数而非强制参数，本接口设置参数（如显示帧率、像素值域范围等）的最终效果取决于解码器/播放器行为，或者用户自定义的解码/播放行为。

函数原型

```
hi_s32 hi_mpi_venc_set_h265_vui(hi_venc_chn chn, const hi_venc_h265_vui *h265_vui)
```

参数说明

参数名	输入/输出	说明
chn	输入	编码通道号。 Atlas 推理系列产品，编码通道号的取值范围：[0, 256)，JPEGE功能和VENC功能共用通道，且通道总数最多256。 Atlas 200/500 A2推理产品，编码通道号的取值范围：[0, 128)，JPEGE功能和VENC功能共用通道，且通道总数最多128。
h265_vui	输入	H.265协议编码通道的Vui参数。

返回值说明

- 0：成功
- 非0：失败，参见[1.9.21.12 VENC视频编码/JPEGE图片编码返回码](#)

1.9.18.38 hi_mpi_venc_get_h265_vui

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

获取H.265协议编码通道的Vui配置。

约束说明

- 本接口用于获取H.265协议编码通道的Vui配置。
- 本接口在编码通道创建之后，编码通道销毁之前调用。
- 建议用户在创建通道之后，启动编码之前调用此接口，减少在编码过程中调用的次数。
- 用户应当仅在H.265协议编码通道中调用本接口。

函数原型

[hi_s32](#) hi_mpi_venc_get_h265_vui([hi_venc_chn](#) chn, [hi_venc_h265_vui](#) *h265_vui)

参数说明

参数名	输入/输出	说明
chn	输入	编码通道号。 Atlas 推理系列产品，编码通道号的取值范围：[0, 256)，JPEGE功能和VENC功能共用通道，且通道总数最多256。 Atlas 200/500 A2推理产品，编码通道号的取值范围：[0, 128)，JPEGE功能和VENC功能共用通道，且通道总数最多128。
h265_vui	输出	H.265协议编码通道的Vui属性。

返回值说明

- 0：成功

- 非0：失败，参见[1.9.21.12 VENC视频编码/JPEGE图片编码返回码](#)

1.9.18.39 hi_mpi_venc_set_cu_pred

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

本接口用于调节通道CU/MB的模式选择的倾向性，支持调节帧间预测（Inter）和帧内预测（Intra）的倾向性配置以及编码块大小的倾向性配置。

编码协议以像素块作为基本处理单元，H.265/H.264中分别称为CU(Coding Unit)/MB(MacroBlock)。本接口中的CU模式指Inter/Intra预测模式中对不同尺寸像素块的选择。

约束说明

- 仅H.265/H.264编码通道支持。
- 建议用户在创建通道之后，启动编码之前调用本接口，减少编码过程中调用本接口的次数。若在编码过程中调用本接口，则下一帧编码时配置生效。
- 建议用户在调用此接口之前先调用[hi_mpi_venc_get_cu_pred](#)接口，获取CU模式选择的倾向性配置，然后再进行设置。
- 注意某些场景下特定模式不使能（例如，H.264中最大MB尺寸为16*16，且baseline和main profile下inter16_cost不使能，high profile下inter8_cost不使能），即使用户设置了这些特定模式的倾向性参数也不生效。

函数原型

hi_s32 hi_mpi_venc_set_cu_pred([hi_venc_chn](#) chn, const [hi_venc_cu_prediction](#) *cu_pred)

参数说明

参数名	输入/输出	说明
chn	输入	编码通道号。 Atlas 推理系列产品，编码通道号的取值范围：[0, 256)，JPEGE功能和VENC功能共用通道，且通道总数最多256。 Atlas 200/500 A2推理产品，编码通道号的取值范围：[0, 128)，JPEGE功能和VENC功能共用通道，且通道总数最多128。

参数名	输入/输出	说明
cu_pred	输入	CU模式选择的倾向性参数。 该结构体内包含以下参数： - pred_mode：倾向性选择模式，支持auto和manual两种模式。auto模式下由驱动内部完成倾向性配置，即用户的配置参数不生效；manual模式允许用户根据实际场景完成自定义的倾向性配置。 - intra32_cost/intra16_cost/intra8_cost/intra4_cost/inter64_cost/inter32_cost/inter16_cost/inter8_cost分别设置各自模式的倾向性大小。默认每个cost值为8，即不做倾向性配置。每个模式的cost设置越大意味着选择该模式的倾向性越小。

返回值说明

- 0：成功
- 非0：失败，参见[1.9.21.12 VENC视频编码/JPEG图片编码返回码](#)

1.9.18.40 hi_mpi_venc_get_cu_pred

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

获取通道CU/MB的模式选择的倾向性设置。

约束说明

- 本接口用于获取CU模式选择的倾向性配置。
- 本接口在编码通道创建之后，编码通道销毁之前调用。
- 仅H.265/H.264编码通道支持。

函数原型

```
hi_s32 hi_mpi_venc_get_cu_pred(hi_venc_chn chn, hi_venc_cu_prediction *cu_pred)
```


参数说明

参数名	输入/输出	说明
chn	输入	编码通道号。 Atlas 推理系列产品，编码通道号的取值范围：[0, 256)，JPEGE功能和VENC功能共用通道，且通道总数最多256。 Atlas 200/500 A2推理产品，编码通道号的取值范围：[0, 128)，JPEGE功能和VENC功能共用通道，且通道总数最多128。
cu_pred	输出	CU模式选择的倾向性参数。

返回值说明

- 0：成功
- 非0：失败，参见[1.9.21.12 VENC视频编码/JPEGE图片编码返回码](#)

1.9.18.41 hi_mpi_venc_set_intra_refresh

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

设置P帧刷新Islice的参数。

约束说明

- 仅支持H.265/H.264、GOP模式为HI_VENC_GOP_MODE_NORMAL_P的编码通道。
- 调用[hi_mpi_venc_set_ref_param](#)接口设置高级跳帧参考后需要重新进行设置，不支持高级跳帧参考pred_en为0。
- 建议在创建通道后、启动编码前先调用[hi_mpi_venc_get_intra_refresh](#)接口获取P帧刷Islice的设置参数后，再调用本接口设置P帧刷Islice的参数。
- 调用本接口使能P帧刷Islice时，会生成一个I帧，且需等下一个GOP的IDR帧才能生效；关闭P帧刷Islice参数时会立即生效。

函数原型

```
hi_s32 hi_mpi_venc_set_intra_refresh(hi_venc_chn chn, const hi_venc_intra_refresh *intra_refresh);
```


参数说明

参数名	输入/输出	说明
chn	输入	编码通道号。 Atlas 推理系列产品，编码通道号的取值范围：[0, 256)，JPEGE功能和VENC功能共用通道，且通道总数最多256。 Atlas 200/500 A2推理产品，编码通道号的取值范围：[0, 128)，JPEGE功能和VENC功能共用通道，且通道总数最多128。
intra_refresh	输入	刷Islice的参数。

返回值说明

- 0：成功
- 非0：失败，参见[1.9.21.12 VENC视频编码/JPEGE图片编码返回码](#)

1.9.18.42 hi_mpi_venc_get_intra_refresh

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

获取P帧刷新Islice的设置参数。

约束说明

- 本接口必须在编码通道创建之后，编码通道销毁之前调用。
- 切换GOP模式时，获取的属性是已设置的最新的属性。

函数原型

```
hi_s32 hi_mpi_venc_get_intra_refresh(hi_venc_chn chn, hi_venc_intra_refresh *intra_refresh)
```

参数说明

参数名	输入/输出	说明
chn	输入	编码通道号。 Atlas 推理系列产品，编码通道号的取值范围：[0, 256)，JPEGE功能和VENC功能共用通道，且通道总数最多256。 Atlas 200/500 A2推理产品，编码通道号的取值范围：[0, 128)，JPEGE功能和VENC功能共用通道，且通道总数最多128。
intra_refresh	输出	刷Islice的参数。

返回值说明

- 0：成功
- 非0：失败，参见[1.9.21.12 VENC视频编码/JPEGE图片编码返回码](#)

1.9.18.43 hi_mpi_venc_set_chn_attr

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

本接口用于设置编码通道的属性。

约束说明

- 仅H.265/H.264编码通道支持。
- 建议用户在调用此接口之前先调用[hi_mpi_venc_get_chn_attr](#)接口，获取当前通道属性，然后再进行设置。
- 编码通道属性包括了编码器属性、帧结构GOP属性和码率控制器RC属性三部分。编码通道属性分为动态属性和静态属性两种。其中，静态属性的属性值在通道创建时配置，在通道创建之后不能被修改；动态属性的属性值在通道创建时配置，有部分参数不支持修改，支持修改的参数在通道销毁之前可以被修改。如果设置静态属性或者不支持修改的参数，则返回失败。静态属性、属性是否支持修改、属性取值范围具体请参见[hi_venc_chn_attr](#)。
- 设置编码通道属性，未修改gop_mode 时，则通道属性立即生效；修改gop_mode时，则通道属性延时生效，最大延时生效时间如下表所示（**当前不支持HI_VENC_GOP_MODE_LOW_DELAY_B模式**）：

gop_mode修改前	gop_mode修改后	最大延时（以编码帧的个数为单位）
HI_VENC_GOP_MODE_LOW_DELAY_B	非 HI_VENC_GOP_MODE_LOW_DELAY_B	1+b_frame_num
非 HI_VENC_GOP_MODE_LOW_DELAY_B	任意不同于改变前的 gop_mode	1

- 修改gop_mode时，通道的P帧刷Islice、跳帧参考的属性（ H.264/H.265 ）、pic_order_cnt_type（ H.264 ）会恢复默认值，如需继续使用需要重新配置。
- 如果修改GOP、RC属性中的一种，RC模块会重新初始化，所有RC高级参数都恢复到默认值。如果客户程序中有对RC高级参数的调整，需要重新设置。RC高级参数请参考[hi_mpi_venc_set_rc_param](#)等。
- 如果同时修改gop_mode和rc_mode时，由于存在延迟生效（延时参见上表），为了防止调用[hi_mpi_venc_get_rc_param](#)获取的RC参数值不准确，建议等待延迟参数生效后再执行相应操作。
- 调用本接口可修改VENC视频编码时的通道宽度pic_width和高度pic_height，但需注意：
 - 调用本接口修改宽、高时，若系统内部还缓存着未编码完的视频数据，这时系统会丢弃缓存的视频数据，导致该部分数据丢失。
 - 调用本接口修改宽、高后，通过[hi_mpi_venc_send_frame](#)下发的width、height参数值需要和修改后的pic_width、pic_height参数值一致。
 - 如果在调用[hi_mpi_venc_get_stream](#)与[hi_mpi_venc_release_stream](#)之间修改宽、高，那么hi_mpi_venc_release_stream接口释放刚接收的码流时会返回HI_ERR_VENC_ILLEGAL_PARAM错误码。

函数原型

hi_s32 hi_mpi_venc_set_chn_attr([hi_venc_chn](#) chn, const [hi_venc_chn_attr](#) *chn_attr)

参数说明

参数名	输入/输出	说明
chn	输入	编码通道号。 Atlas 推理系列产品，编码通道号的取值范围：[0, 256)，JPEGE功能和VENC功能共用通道，且通道总数最多256。 Atlas 200/500 A2推理产品，编码通道号的取值范围：[0, 128)，JPEGE功能和VENC功能共用通道，且通道总数最多128。
chn_attr	输入	编码通道属性的指针。

返回值说明

- 0：成功
- 非0：失败，参见[1.9.21.12 VENC视频编码/JPEGE图片编码返回码](#)

1.9.18.44 hi_mpi_venc_get_chn_attr

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

获取编码通道当前的属性。

约束说明

- 本接口在编码通道创建之后、编码通道销毁之前调用。
- 仅H.265/H.264编码通道支持。

函数原型

[hi_s32](#) hi_mpi_venc_get_chn_attr([hi_venc_chn](#) chn, [hi_venc_chn_attr](#) *chn_attr)

参数说明

参数名	输入/输出	说明
chn	输入	编码通道号。 Atlas 推理系列产品，编码通道号的取值范围：[0, 256)，JPEGE功能和VENC功能共用通道，且通道总数最多256。 Atlas 200/500 A2推理产品，编码通道号的取值范围：[0, 128)，JPEGE功能和VENC功能共用通道，且通道总数最多128。
chn_attr	输出	编码通道属性的指针。

返回值说明

- 0：成功
- 非0：失败，参见[1.9.21.12 VENC视频编码/JPEGE图片编码返回码](#)

1.9.18.45 hi_mpi_venc_set_slice_split

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

本接口用于设置通道的slice分割配置。

约束说明

- 仅H.265/H.264编码通道支持。
- 本接口在编码通道创建之后，编码通道销毁之前设置。若在编码过程中调用本接口，则下一帧编码时配置生效。
- 建议用户在调用此接口之前先调用hi_mpi_venc_get_slice_split接口，获取当前通道的slice_split配置，然后再进行设置。

函数原型

```
hi_s32 hi_mpi_venc_set_slice_split(hi_venc_chn chn, const hi_venc_slice_split
*slice_split)
```

参数说明

参数名	输入/输出	说明
chn	输入	编码通道号。 Atlas 推理系列产品，编码通道号的取值范围：[0, 256)，JPEG功能 and VENC功能共用通道，且通道总数最多256。 Atlas 200/500 A2推理产品，编码通道号的取值范围：[0, 128)，JPEG功能 and VENC功能共用通道，且通道总数最多128。
slice_split	输入	H.264/H.265协议编码的码流分割参数。

返回值说明

- 0：成功
- 非0：失败，参见[1.9.21.12 VENC视频编码/JPEG图片编码返回码](#)

1.9.18.46 hi_mpi_venc_get_slice_split

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

本接口用于获取通道的slice分割配置。

约束说明

- 仅H.265/H.264编码通道支持。
- 本接口在编码通道创建之后、编码通道销毁之前调用。

函数原型

```
hi_s32 hi_mpi_venc_get_slice_split(hi_venc_chn chn, hi_venc_slice_split
*slice_split)
```

参数说明

参数名	输入/输出	说明
chn	输入	编码通道号。 Atlas 推理系列产品，编码通道号的取值范围：[0, 256)，JPEG功能 and VENC功能共用通道，且通道总数最多256。 Atlas 200/500 A2推理产品，编码通道号的取值范围：[0, 128)，JPEG功能和VENC功能共用通道，且通道总数最多128。
slice_split	输出	H.264/H.265协议编码的码流分割参数。

返回值说明

- 0：成功
- 非0：失败，参见[1.9.21.12 VENC视频编码/JPEG图片编码返回码](#)

1.9.18.47 hi_mpi_venc_send_frame_ex

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas 200/500 A2推理产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

本接口支持用户对于H .264和H.265编码通路发送原始图像及该图的QpMap表信息进行编码。

约束说明

- 仅H.265/H.264编码通道支持。
- 视频输入的原始图像大小必须与编码通道的大小保持一致。
- 调用该接口发送图像，用户需要保证编码通道已创建且开启接收输入图片。

- 编码时，输入内存需要在调用[hi_mpi_venc_get_stream](#)接口获取结果之后才能进行释放。
- 视频编码，对输入、输出的约束请参见[1.9.18.3 VENC功能及约束说明](#)。

函数原型

hi_s32 [hi_mpi_venc_send_frame_ex](#)([hi_venc_chn](#) chn, const [hi_user_frame_info](#) *frame, [hi_s32](#) milli_sec)

参数说明

参数名	输入/输出	说明
chn	输入	编码通道号。 Atlas 推理系列产品，编码通道号的取值范围：[0, 256)，JPEG功能 and VENC功能共用通道，且通道总数最多256。
frame	输入	原始图像信息的指针。
milli_sec	输入	超时时间，单位是毫秒。 • -1：阻塞方式 • 0：非阻塞方式 • >0：超时方式，配置具体的超时时间。超时时间受操作系统影响，一般偏差在操作系统的片一个时间片内，例如，操作系统的片一个时间片为4ms，用户设置的milli_sec参数值为1，则实际的超时时间在1ms到5ms范围内。在CPU负载高场景下，超时时间仍可能存在波动。

返回值说明

- 0：成功
- 非0：失败，参见[1.9.21.12 VENC视频编码/JPEG图片编码返回码](#)

1.9.18.48 hi_mpi_venc_get_qpmap_stride

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas 200/500 A2推理产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

本接口支持用户获取qpmap映射表stride。

函数原型

hi_s32 hi_mpi_venc_get_qpmap_stride(**hi_u32** width)

参数说明

参数名	输入/输出	说明
width	输入	编码图像的宽。

返回值说明

- 0：失败
- 非0：成功

1.9.18.49 hi_mpi_venc_get_qpmap_size

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas 200/500 A2推理产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

本接口支持用户获取qpmap映射表大小。

约束说明

仅H.265/H.264编码通道支持。

函数原型

hi_s32 hi_mpi_venc_get_qpmap_size(**hi_payload_type** type, **hi_u32** width, **hi_u32** height)

参数说明

参数名	输入/输出	说明
type	输入	编码协议类型。
width	输入	编码图像的宽。
height	输入	编码图像的高。

返回值说明

- 0：失败
- 非0：成功

1.9.18.50 hi_mpi_venc_set_streamcopy_param

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas 200/500 A2推理产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

本接口支持用户设置帧拷贝参数。

约束说明

- 仅H.265/H.264编码通道支持。
- 在使能拷贝帧、但拷贝间隔设置为0时，H265 SPS/Slice按照新协议处理(H264协议未变)，后续可以调整拷贝间隔，此时不需要重新请求I帧就可以拷贝生效；在未使能拷贝帧，后续动态修改为使能拷贝帧，此时需要新的IDR后才生效。
- 打开帧拷贝后，时域tmv会自动失效

函数原型

hi_s32 hi_mpi_venc_set_streamcopy_param(**hi_venc_chn** chn, const **hi_venc_streamcopy_param** *streamcopy_param)

参数说明

参数名	输入/输出	说明
chn	输入	编码通道号。
streamcopy_param	输入	帧拷贝参数。

返回值说明

- 0：成功
- 非0：失败，参见[1.9.21.12 VENC视频编码/JPEGE图片编码返回码](#)

1.9.18.51 hi_mpi_venc_get_streamcopy_param

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas 200/500 A2推理产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

本接口支持用户获取帧拷贝参数。

约束说明

仅H.265/H.264编码通道支持。

函数原型

```
hi_s32 hi_mpi_venc_get_streamcopy_param(hi_venc_chn chn,
hi_venc_streamcopy_param *streamcopy_param)
```

参数说明

参数名	输入/输出	说明
chn	输入	编码通道号。
streamcopy_param	输出	帧拷贝参数。

返回值说明

- 0：成功
- 非0：失败，参见1.9.21.12 VENC视频编码/JPEGE图片编码返回码

1.9.19 PNGD 图片解码功能

1.9.19.1 功能及约束说明

Atlas 200/300/500 推理产品上，不支持本章中的PNGD功能。

Atlas 训练系列产品上，不支持本章中的PNGD功能。

功能说明

PNGD (PNG Decoder) 功能：实现PNG格式图片的解码。

PNGD在解码图片时，支持按源图片格式解码。源图片格式解码是指解码前后图片的编码格式保持一致，例如解码前输入图片格式为RGB，解码后输出图片格式为RGB888。

如果不清楚PNG源图片格式、但想使用源格式解码时，有以下两种方式：

- 在调用PNGD解码接口时，直接将输出图片格式配置为HI_PIXEL_FORMAT_UNKNOWN，输出格式默认按源图片格式输出。此种方式，

由于不知道输出图片格式，因此需要用户申请尽量大的内存，防止内存不够，无法存放输出图片。

输入图片格式	操作	实际输出图片格式
RGB/GRAY	将输出图片格式设置为 HI_PIXEL_FORMAT_UNK OWN	RGB888
RGBA/AGRAY	将输出图片格式设置为 HI_PIXEL_FORMAT_UNK OWN	RGBA8888

- 先调用[hi_mpi_png_get_image_info](#)接口根据传入的PNG源图片，获取按源图解码时的输出图片的宽、高、宽stride、高stride、解码输出内存大小、图片格式等信息后，再调用PNGD解码接口，使用通过hi_mpi_png_get_image_info接口获取的图片格式来设置输出图片格式。

图片分辨率约束

- 输入图片分辨率：
最大分辨率4096*4096，最小分辨率32*32。
- 输出图片分辨率
PNGD只对图片解码，不会改变图片分辨率，因此输出与输入的图片分辨率保持一致。

内存申请/释放接口说明

输入内存的大小就是指实际的输入图片所占用的大小，输出内存的大小参见[表1-79](#)中的计算公式。

版本	内存申请/释放接口
Atlas 推理系列产品	PNG图片解码时，调用 hi_mpi_dvpp_malloc \ hi_mpi_dvpp_free 接口申请\释放Device上的输入、输出内存，这部分内存的生命周期由用户自行管理。
Atlas A2训练系列产品/Atlas 800I A2推理产品 Atlas 200/500 A2推理产品	PNG图片解码时，支持以下两种内存申请接口： - 调用aclrtMalloc\aclrtFree接口申请\释放Device上的输入、输出内存，这部分内存的生命周期由用户自行管理。 - 调用hi_mpi_dvpp_malloc\hi_mpi_dvpp_free接口申请\释放Device上的输入、输出内存，这部分内存的生命周期由用户自行管理。 注意：调用hi_mpi_dvpp_malloc接口申请的内存为媒体数据处理专用内存，但专用内存的地址空间有限，若关注内存规划或内存资源有限时，建议调用aclrtMalloc接口申请内存。

图片格式、宽高对齐、内存约束

若不确定输出格式、将输出图片格式设置为HI_PIXEL_FORMAT_UNKNOWN时:

- 宽stride:
如果输入图片是RGB或GRAY格式，则宽stride为输出图片的宽先向上128对齐后再乘以3的值；如果输入图片是RGBA或AGRAY格式，则宽stride为输出图片的宽先向上128对齐后再乘以4的值。
- 高stride: 为输出图片的高向上16对齐后的值。

表 1-79 图片格式、宽高对齐、内存大小约束

输入图片格式	输出图片格式	输出图片宽、高对齐要求	输出图片宽stride、高stride、内存大小要求
RGB	RGB888	无对齐要求	宽stride为宽向上1对齐、或16对齐、或128对齐后再乘以3的值。 高stride的取值范围: [输出图片的高, 输出图片的高向上128对齐]。 内存大小 (单位Byte) = 宽stride * 高stride
GRAY	RGB888		
RGBA	RGB888		宽stride为宽向上1对齐、或16对齐、或128对齐后再乘以3的值。 高stride的取值范围: [输出图片的高, 输出图片的高向上128对齐]。 内存大小 (单位Byte) = 宽stride * 高stride
	RGBA8888 8bit		宽stride为宽向上1对齐、或16对齐、或128对齐后再乘以4的值。 高stride的取值范围: [输出图片的高, 输出图片的高向上128对齐]。 内存大小 (单位Byte) = 宽stride * 高stride
AGRAY	RGB888		宽stride为宽向上1对齐、或16对齐、或128对齐后再乘以3的值。 高stride的取值范围: [输出图片的高, 输出图片的高向上128对齐]。 内存大小 (单位Byte) = 宽stride * 高stride
	RGBA8888 8bit		宽stride为宽向上1对齐、或16对齐、或128对齐后再乘以4的值。 高stride的取值范围: [输出图片的高, 输出图片的高向上128对齐]。 内存大小 (单位Byte) = 宽stride * 高stride

 说明

- 输出图片格式的定义请参见[hi_pixel_format](#)，宽stride、高stride等概念请参见[1.9.2 基本概念](#)。
- 如果用户将宽stride、高stride设置为0，传入对应的接口，PNGD内部在处理时，会默认根据输出图片格式将宽stride向上128对齐、高stride向上16对齐。

其它约束

PNGD只支持对完整PNG图片进行解码，不支持将一张PNG图片分割成多个数据包后，由PNGD解码。

1.9.19.2 性能指标说明

Atlas 200/300/500 推理产品上，当前版本不支持该功能。
Atlas 训练系列产品上，当前版本不支持该功能。

性能指标说明

本节以单个昇腾AI处理器为例，描述基本场景性能指标参考，如下表所示，其它分辨率可以等量估算。

下表中的1路对应一个通道，一个通道对应一个线程；各分辨率的含义为：1080p指分辨率为1920*1080的图片，4K指分辨率为3840*2160的图片。

场景举例	总帧率
1080p*n路 (1≤n≤5)	n*4fps
1080p*n路 (n≥ 6)	24fps
4k*n路 (1≤n≤5)	n*1fps
4k*n路 (n≥ 6)	6fps

1.9.19.3 hi_mpi_pngd_create_chn

Atlas 200/300/500 推理产品，不支持该接口。
Atlas 训练系列产品，不支持该接口。

函数功能

根据设置的通道属性创建解码通道。

约束说明

- 单个Device上的通道号不能超出最大的通道号范围。
- 如果参数attr为空，会返回错误码HI_ERR_PNGD_NULL_PTR。
- 在创建PNG解码通道之前必须保证通道未创建（或者已经销毁），否则会直接返回错误。

- 系统内存不足时会返回HI_ERR_PNGD_NO_MEM的错误码，可考虑扩展OS内存。
- 当通道属性attr中的值超过解码能力集时，会返回HI_ERR_PNGD_ILLEGAL_PARAM的错误码。

函数原型

```
hi_s32 hi_mpi_pngd_create_chn(hi_pngd_chn chn, const hi_pngd_chn_attr *attr)
```

参数说明

参数名	输入/输出	说明
chn	输入	解码通道号。 该参数的取值范围：[0, 128)，通道总数最多128。 说明 对于Atlas 推理系列产品，昇腾虚拟化实例（Ascend Virtual Instance）场景下，使用不同的昇腾虚拟化实例（Ascend Virtual Instance）模板，对通道数的限制不同： • 使用vir04_4c_dvpp模板时，通道总数固定为128。 • 使用vir04_3c_ndvpp模板时，表示不使用DVPP功能，因此通道总数为0。 • 使用其它模板时，通道总数 = 切分规格 * 128，如果通道总数不为整数，则向下取整。 对于Atlas A2训练系列产品/Atlas 800I A2推理产品，昇腾虚拟化实例（Ascend Virtual Instance）场景下，使用不同的昇腾虚拟化实例（Ascend Virtual Instance）模板，对通道数的限制不同： • 使用vir12_3c_32g_m、vir10_3c_16g_m、vir10_3c_32g_m模板时，通道总数固定为128。 • 使用vir12_3c_32g_nm、vir10_3c_16g_nm、vir10_3c_32g_nm模板时，表示不使用DVPP功能，因此通道总数为0。 • 使用其它模板时，通道总数 = 切分规格 * 128，如果通道总数不为整数，则向下取整。 昇腾虚拟化实例（Ascend Virtual Instance）的详细描述，以及各场景下的算力资源规格请参见《昇腾虚拟化实例（AVI）用户指南》。
attr	输入	解码通道属性指针。

返回值说明

- 0：成功
- 非0：失败，参见[1.9.21.13 PNGD图像解码返回码](#)

参考资源

接口调用流程及示例，参见PNGD图片解码。

1.9.19.4 hi_mpi_pngd_destroy_chn

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

函数功能

用户退出流程时，需要关闭之前创建的解码通道，以释放相关资源。

约束说明

销毁前必须保证通道已创建，否则会返回通道未创建的错误。

函数原型

hi_s32 hi_mpi_pngd_destroy_chn(**hi_pngd_chn** chn)

参数说明

参数名	输入/输出	说明
chn	输入	解码通道号。 该参数的取值范围：[0, 128)，通道总数最多128。

返回值说明

- 0：成功
- 非0：失败，参见[1.9.21.13 PNGD图像解码返回码](#)

参考资源

接口调用流程及示例，参见PNGD图片解码。

1.9.19.5 hi_mpi_pngd_send_stream

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

函数功能

解码前，向解码通道发送码流数据及存放解码结果的buffer。

约束说明

- 发送数据前必须保证通道已经被创建，否则直接返回通道未创建的错误码HI_ERR_PNGD_UNEXIST。如果在发送码流过程中销毁通道，就会立刻返回错误码HI_ERR_PNGD_UNEXIST。
- 发送码流时需要送入整张png图片。否则，解码会出现错误。

- 以非阻塞方式发送码流，如果码流缓冲区已满，会立刻返回错误码 HI_ERR_PNGD_BUF_FULL。
- 以超时方式发送码流，到达设定的超时时间还不能成功发送码流会返回错误码 HI_ERR_PNGD_BUF_FULL。
- 图像解码时，对输入、输出图片的要求请参见1.9.19.1 功能及约束说明。
- 解码时，输入输出内存必须使用hi_mpi_dvpp_malloc接口和hi_mpi_dvpp_free接口进行申请和释放，输入输出内存均需要在调用hi_mpi_pngd_get_image_data接口获取结果之后才能进行释放。

函数原型

```
hi_s32 hi_mpi_pngd_send_stream(hi_pngd_chn chn, const hi_img_stream
*stream, hi_pic_info *png_pic_info, hi_s32 milli_sec)
```

参数说明

参数名	输入/输出	说明
chn	输入	解码通道号。 该参数的取值范围：[0, 128)，通道总数最多128。
stream	输入	输入码流信息的指针。 该结构体内的addr参数配置的地址为Device上的内存地址。
png_pic_info	输入	输出图片信息的指针。
milli_sec	输入	超时时间，单位是毫秒。 • -1：阻塞方式 • 0：非阻塞方式 • >0：超时方式，配置具体的超时时间。超时时间受操作系统影响，一般偏差在操作系统的 一个时间片内，例如，操作系统的 一个时间片为4ms，用户设置的 milli_sec参数值为1，则实际的超时时间在1ms到5ms 范围内。在CPU负载高场景下，超时时间仍可能存在波动。

返回值说明

- 0：成功
- 非0：失败，参见1.9.21.13 PNGD图像解码返回码

参考资源

接口调用流程及示例，参见PNGD图片解码。

1.9.19.6 hi_mpi_pngd_get_image_data

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

函数功能

解码后，获取解码通道的解码图像及输入Stream。

约束说明

- 此接口通过改变milli_sec值支持阻塞方式、非阻塞方式、超时方式获取解码图像。
- 获取解码图像时必须保证通道已经被创建，否则直接返回通道未创建的错误码 HI_ERR_PNGD_UNEXIST。如果在获取图像的过程中销毁通道，就会立刻返回错误码 HI_ERR_PNGD_UNEXIST。
- 以非阻塞方式获取解码图像，如果缓冲区内无图像，会立刻返回错误码 HI_ERR_PNGD_BUF_EMPTY。
- 以超时方式获取解码图像，到达设定的超时时间还不能获取到图像则会返回错误码 HI_ERR_PNGD_BUF_EMPTY。
- 向PNGD获取解码结果，输入buffer和输出buffer一起获取。
- PNGD图片解码时，关于输入、输出图片的要求请参见[1.9.19.1 功能及约束说明](#)。

函数原型

```
hi_s32 hi_mpi_pngd_get_image_data(hi_pngd_chn chn, hi_pic_info
*png_pic_info, hi_img_stream *stream, hi_s32 milli_sec)
```

参数说明

参数名	输入/输出	说明
chn	输入	解码通道号。 该参数的取值范围：[0, 128)，通道总数最多128。
png_pic_info	输出	已解码的图像信息的指针。解码后的数据存放在Device内存中。
stream	输出	已解码的输入码流信息的指针。
milli_sec	输入	超时时间，单位是毫秒。 • -1：阻塞方式 • 0：非阻塞方式 • >0：超时方式，配置具体的超时时间。超时时间受操作系统影响，一般偏差在操作系统的 一个时间片内，例如，操作系统的 一个时间片为4ms，用户设置的 milli_sec参数值为1，则实际的超时时间在1ms到5ms 范围内。在CPU负载高场景下，超时时间仍可能存在波动。

返回值说明

- 0：获取数据成功
- 非0：失败，参见[1.9.21.13 PNGD图像解码返回码](#)。HI_ERR_PNGD_BUF_EMPTY表示无数据，其他错误码表示解码异常

参考资源

接口调用流程及示例，参见PNGD图片解码。

1.9.19.7 hi_mpi_png_get_image_info

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

函数功能

根据输入码流，获取png源图片的宽、高、源格式及源格式解码时输出图片的宽stride、高stride、以及所需的内存大小等信息。

约束说明

提供给用户来计算解码后输出图片占用的内存大小。当前仅支持解析png图片。通过本接口获取的widthStride向上128对齐、heightStride向上16对齐。

函数原型

hi_s32 hi_mpi_png_get_image_info(const **hi_img_stream** *png_stream,
hi_img_info *img_info)

参数说明

参数名	输入/输出	说明
png_stream	输入	输入码流信息的指针。 标准形态下，在Host上调用本接口时，该结构体内的addr参数配置的地址必须是Host上的内存地址。
img_info	输出	图片信息的指针。

返回值说明

- 0：成功
- 非0：失败，参见[1.9.21.13 PNGD图像解码返回码](#)

1.9.20 枚举值及数据结构

1.9.20.1 公共

此部分作为公共部分，描述了本模块所支持以及未来可能支持的图片格式，以及支持的Payload（数据内容）类型等等。

1.9.20.1.1 基本数据类型说明

```
typedef unsigned char hi_u8;
typedef signed char hi_s8;
typedef unsigned short hi_u16;
typedef short hi_s16;
typedef unsigned int hi_u32;
typedef int hi_s32;
typedef unsigned long long hi_u64;
typedef long long hi_s64;
typedef char hi_char;
typedef double hi_double;
typedef hi_u32 hi_fr32;
typedef float hi_float;

typedef enum {
    HI_FALSE = 0,
    HI_TRUE = 1,
} hi_bool;

#define hi_void void
#define HI_NULL 0L
#define HI_SUCCESS 0
#define HI_FAILURE (-1)
```

1.9.20.1.2 hi_pixel_format

说明

DVPP所支持的各种图片格式的枚举。

定义

```
typedef enum {
    HI_PIXEL_FORMAT_YUV_400 = 0,           // YUV400 8bit
    HI_PIXEL_FORMAT_YUV_SEMIPLANAR_420 = 1, // YUV420SP NV12 8bit
    HI_PIXEL_FORMAT_YVU_SEMIPLANAR_420 = 2, // YUV420SP NV21 8bit
    HI_PIXEL_FORMAT_YUV_SEMIPLANAR_422 = 3, // YUV422SP 8bit
    HI_PIXEL_FORMAT_YVU_SEMIPLANAR_422 = 4, // YVU422SP 8bit
    HI_PIXEL_FORMAT_YUV_SEMIPLANAR_444 = 5, // YUV444SP 8bit
    HI_PIXEL_FORMAT_YVU_SEMIPLANAR_444 = 6, // YVU444SP 8bit
    HI_PIXEL_FORMAT_YUYV_PACKED_422 = 7,    // YUV422 Package YUYV 8bit
    HI_PIXEL_FORMAT_UYVY_PACKED_422 = 8,    // YUV422 Package UYVY 8bit
    HI_PIXEL_FORMAT_VYUY_PACKED_422 = 9,    // YUV422 Package VYUY 8bit
    HI_PIXEL_FORMAT_YUYV_PACKED_444 = 10,   // YUV422 Package VYUY 8bit
    HI_PIXEL_FORMAT_YUV_PACKED_444 = 11,    // YUV444 Package 8bit
    HI_PIXEL_FORMAT_RGB_888 = 12,           // RGB888
    HI_PIXEL_FORMAT_BGR_888 = 13,           // BGR888
    HI_PIXEL_FORMAT_ARGB_8888 = 14,         // ARGB8888
    HI_PIXEL_FORMAT_ABGR_8888 = 15,         // ABGR8888
    HI_PIXEL_FORMAT_RGBA_8888 = 16,         // RGBA8888
    HI_PIXEL_FORMAT_BGRA_8888 = 17,         // BGRA8888
    HI_PIXEL_FORMAT_YUV_SEMI_PLANNER_420_10BIT = 18, // YUV420SP 10bit
    HI_PIXEL_FORMAT_YVU_SEMI_PLANNER_420_10BIT = 19, // YVU420sp 10bit
    HI_PIXEL_FORMAT_YVU_PLANAR_420 = 20,    // YVU420P 8bit
    HI_PIXEL_FORMAT_YVU_PLANAR_422 = 21,    // YVU422P 8bit
    HI_PIXEL_FORMAT_YVU_PLANAR_444 = 22,    // YVU444P 8bit
    HI_PIXEL_FORMAT_RGB_444 = 23,           // RGB444 R:4bit G:4bit B:4bit, 当前不支持该格式
    HI_PIXEL_FORMAT_BGR_444 = 24,           // BGR444 R:4bit G:4bit B:4bit, 当前不支持该格式
    HI_PIXEL_FORMAT_ARGB_4444 = 25,        // ARGB4444 A:4bit R:4bit G:4bit B:4bit
    HI_PIXEL_FORMAT_ABGR_4444 = 26,        // ABGR4444 A:4bit R:4bit G:4bit B:4bit, 当前不支持该格式
```

```
HI_PIXEL_FORMAT_RGBA_4444 = 27, // RGBA4444 R:4bit G:4bit B:4bit A:4bit, 当前不支持该格式
HI_PIXEL_FORMAT_BGRA_4444 = 28, // BGRA4444 B:4bit G:4bit R:4bit A:4bit, 当前不支持该格式
HI_PIXEL_FORMAT_RGB_555 = 29, // RGB555 R:5bit G:5bit B:5bit, 当前不支持该格式
HI_PIXEL_FORMAT_BGR_555 = 30, // BGR555 B:5bit G:5bit R:5bit, 当前不支持该格式
HI_PIXEL_FORMAT_RGB_565 = 31, // RGB565 R:5bit G:6bit B:5bit, 当前不支持该格式
HI_PIXEL_FORMAT_BGR_565 = 32, // BGR565 B:5bit G:6bit R:5bit, 当前不支持该格式
HI_PIXEL_FORMAT_ARGB_1555 = 33, // ARGB1555 A:1bit R:5bit G:6bit B:5bit
HI_PIXEL_FORMAT_ABGR_1555 = 34, // ABGR1555 A:1bit B:5bit G:6bit R:5bit, 当前不支持该格式
HI_PIXEL_FORMAT_RGBA_1555 = 35, // RGBA1555 A:1bit B:5bit G:6bit R:5bit, 当前不支持该格式
HI_PIXEL_FORMAT_BGRA_1555 = 36, // BGRA1555 A:1bit B:5bit G:6bit R:5bit, 当前不支持该格式
HI_PIXEL_FORMAT_ARGB_8565 = 37, // ARGB8565 A:8bit R:5bit G:6bit B:5bit, 当前不支持该格式
HI_PIXEL_FORMAT_ABGR_8565 = 38, // ABGR8565 A:8bit B:5bit G:6bit R:5bit, 当前不支持该格式
HI_PIXEL_FORMAT_RGBA_8565 = 39, // RGBA8565 A:8bit R:5bit G:6bit B:5bit, 当前不支持该格式
HI_PIXEL_FORMAT_BGRA_8565 = 40, // BGRA8565 A:8bit B:5bit G:6bit R:5bit, 当前不支持该格式
HI_PIXEL_FORMAT_ARGB_CLUT2 = 41, // ARGB Color Lookup Table 2bit
HI_PIXEL_FORMAT_ARGB_CLUT4 = 42, // ARGB Color Lookup Table 4bit

HI_PIXEL_FORMAT_RGB_BAYER_8BPP = 50,
HI_PIXEL_FORMAT_RGB_BAYER_10BPP = 51,
HI_PIXEL_FORMAT_RGB_BAYER_12BPP = 52,
HI_PIXEL_FORMAT_RGB_BAYER_14BPP = 53,
HI_PIXEL_FORMAT_RGB_BAYER_16BPP = 54, // RGB Bayer 16bit, Bayer图像, 当前不支持该格式
HI_PIXEL_FORMAT_YUV_PLANAR_420 = 55, // YUV420P 8bit
HI_PIXEL_FORMAT_YUV_PLANAR_422 = 56, // YUV422P 8bit
HI_PIXEL_FORMAT_YUV_PLANAR_444 = 57, // YUV444P 8bit
HI_PIXEL_FORMAT_YVU_PACKED_444 = 58, // YVU444 Package 8bit
HI_PIXEL_FORMAT_XYUV_PACKED_444 = 59, // AYUV444 Package 8bit
HI_PIXEL_FORMAT_XYVU_PACKED_444 = 60, // AYVU444 Package 8bit
HI_PIXEL_FORMAT_YUV_SEMIPLANAR_411 = 61, // YUV411SP 8bit
HI_PIXEL_FORMAT_YVU_SEMIPLANAR_411 = 62, // YVU411SP 8bit
HI_PIXEL_FORMAT_YUV_PLANAR_411 = 63, // YUV411P 8bit
HI_PIXEL_FORMAT_YVU_PLANAR_411 = 64, // YVU411P 8bit
HI_PIXEL_FORMAT_YUV_PLANAR_440 = 65, // YUV440P 8bit
HI_PIXEL_FORMAT_YVU_PLANAR_440 = 66, // YVU440P 8bit

HI_PIXEL_FORMAT_RGB_888_PLANAR = 69, // RGB888 Planar
HI_PIXEL_FORMAT_BGR_888_PLANAR = 70, // BGR888 Planar
HI_PIXEL_FORMAT_HSV_888_PACKAGE = 71, // HSV Package, HSV图像package格式, 当前不支持该格式
HI_PIXEL_FORMAT_HSV_888_PLANAR = 72, // HSV Planar, HSV图像Planar格式, 当前不支持该格式
HI_PIXEL_FORMAT_LAB_888_PACKAGE = 73, // LAB Package, LAB图像package格式, 当前不支持该格式
HI_PIXEL_FORMAT_LAB_888_PLANAR = 74, // LAB Planar, LAB图像Planar格式, 当前不支持该格式
HI_PIXEL_FORMAT_S8C1 = 75, // Signed 8bit for 1pixel 1Channel, 每个像素用1个8bit有符号数
数据表示的单通道图像, 当前不支持该格式
HI_PIXEL_FORMAT_S8C2_PACKAGE = 76, // Signed 8bit for 1pixel 2Channel Package, 每个像素用2个
8bit有符号数表示的双通道图像Package格式, 当前不支持该格式
HI_PIXEL_FORMAT_S8C2_PLANAR = 77, // Signed 8bit for 1pixel 2Channel Planar, 每个像素用2个
8bit有符号数表示的双通道图像Planar格式, 当前不支持该格式
HI_PIXEL_FORMAT_S16C1 = 78, // Signed 16bit 1pixel 1Channel, 每个像素用1个16bit有符号数
数据表示的单通道图像, 当前不支持该格式
HI_PIXEL_FORMAT_U8C1 = 79, // Unsigned 8bit 1pixel 1Channel, 每个像素用1个8bit无符号数
数据表示的单通道图像, 当前不支持该格式
HI_PIXEL_FORMAT_U16C1 = 80, // Unsigned 16bit 1pixel 1Channel, 每个像素用1个16bit无符号
数据表示的单通道图像, 当前不支持该格式
HI_PIXEL_FORMAT_S32C1 = 81, // Signed 32bit 1pixel 1Channel, 每个像素用1个32bit有符号数
数据表示的单通道图像, 当前不支持该格式
HI_PIXEL_FORMAT_U32C1 = 82, // Unsigned 32bit 1pixel 1Channel, 每个像素用1个32bit无符号
数据表示的单通道图像, 当前不支持该格式
HI_PIXEL_FORMAT_U64C1 = 83, // Unsigned 64bit 1pixel 1Channel, 每个像素用1个64bit无符号
数据表示的单通道图像, 当前不支持该格式
HI_PIXEL_FORMAT_S64C1 = 84, // Signed 64bit 1pixel 1Channel, 每个像素用1个64bit有符号数
数据表示的单通道图像, 当前不支持该格式

HI_PIXEL_FORMAT_RGB_888_INT8 = 110, // RGB888 Package 每个像素的单分量占用1个8bit有符号数
HI_PIXEL_FORMAT_BGR_888_INT8 = 111, // BGR888 Package 每个像素的单分量占用1个8bit有符号数
HI_PIXEL_FORMAT_RGB_888_INT16 = 112, // RGB888 Package 每个像素的单分量占用1个16bit有符号数
```

```
HI_PIXEL_FORMAT_BGR_888_INT16 = 113, // BGR888 Package 每个像素的单分量占用1个16bit有符号数
HI_PIXEL_FORMAT_RGB_888_INT32 = 114, // RGB888 Package 每个像素的单分量占用1个32bit有符号数
HI_PIXEL_FORMAT_BGR_888_INT32 = 115, // BGR888 Package 每个像素的单分量占用1个32bit有符号数
HI_PIXEL_FORMAT_RGB_888_UINT16 = 116, // RGB888 Package 每个像素的单分量占用1个16bit无符号数
HI_PIXEL_FORMAT_BGR_888_UINT16 = 117, // BGR888 Package 每个像素的单分量占用1个16bit无符号数
HI_PIXEL_FORMAT_RGB_888_UINT32 = 118, // RGB888 Package 每个像素的单分量占用1个32bit无符号数
HI_PIXEL_FORMAT_BGR_888_UINT32 = 119, // BGR888 Package 每个像素的单分量占用1个32bit无符号数
HI_PIXEL_FORMAT_RGB_888_PLANAR_INT8 = 120, // RGB888 Planar 每个像素的单分量占用1个8bit有符号数
HI_PIXEL_FORMAT_BGR_888_PLANAR_INT8 = 121, // BGR888 Planar 每个像素的单分量占用1个8bit有符号数
HI_PIXEL_FORMAT_RGB_888_PLANAR_INT16 = 122, // RGB888 Planar 每个像素的单分量占用1个16bit有符号数
HI_PIXEL_FORMAT_BGR_888_PLANAR_INT16 = 123, // BGR888 Planar 每个像素的单分量占用1个16bit有符号数
HI_PIXEL_FORMAT_RGB_888_PLANAR_INT32 = 124, // RGB888 Planar 每个像素的单分量占用1个32bit有符号数
HI_PIXEL_FORMAT_BGR_888_PLANAR_INT32 = 125, // BGR888 Planar 每个像素的单分量占用1个32bit有符号数
HI_PIXEL_FORMAT_RGB_888_PLANAR_UINT16 = 126, // RGB888 Planar 每个像素的单分量占用1个16bit无符号数
HI_PIXEL_FORMAT_BGR_888_PLANAR_UINT16 = 127, // BGR888 Planar 每个像素的单分量占用1个16bit无符号数
HI_PIXEL_FORMAT_RGB_888_PLANAR_UINT32 = 128, // RGB888 Planar 每个像素的单分量占用1个32bit无符号数
HI_PIXEL_FORMAT_BGR_888_PLANAR_UINT32 = 129, // BGR888 Planar 每个像素的单分量占用1个32bit无符号数
HI_PIXEL_FORMAT_YUV400_UINT16 = 130, // YUV400 Package 每个像素的单分量占用1个16bit无符号数
HI_PIXEL_FORMAT_YUV400_UINT32 = 131, // YUV400 Package 每个像素的单分量占用1个32bit无符号数
HI_PIXEL_FORMAT_YUV400_UINT64 = 132, // YUV400 Package 每个像素的单分量占用1个64bit无符号数
HI_PIXEL_FORMAT_YUV400_INT8 = 133, // YUV400 Package 每个像素的单分量占用1个8bit有符号数
HI_PIXEL_FORMAT_YUV400_INT16 = 134, // YUV400 Package 每个像素的单分量占用1个16bit有符号数
HI_PIXEL_FORMAT_YUV400_INT32 = 135, // YUV400 Package 每个像素的单分量占用1个32bit有符号数
HI_PIXEL_FORMAT_YUV400_INT64 = 136, // YUV400 Package 每个像素的单分量占用1个64bit有符号数
HI_PIXEL_FORMAT_YUV400_FP16 = 137, // YUV400 Package 每个像素用1个float16数据表示
HI_PIXEL_FORMAT_YUV400_FP32 = 138, // YUV400 Package 每个像素用1个float32数据表示
HI_PIXEL_FORMAT_YUV400_FP64 = 139, // YUV400 Package 每个像素用1个float64数据表示
HI_PIXEL_FORMAT_YUV400_BF16 = 140, // YUV400 Package 每个像素用1个BFloat16数据表示, 当前不支持该格式

HI_PIXEL_FORMAT_YUV_SEMIPLANAR_440 = 1000, // YUV440SP 8bit
HI_PIXEL_FORMAT_YVU_SEMIPLANAR_440 = 1001, // YVU440SP 8bit
HI_PIXEL_FORMAT_FLOAT32 = 1002, // Float 32bit for 1pixel, 每个像素用1个float32数据表示, 当前不支持该格式
HI_PIXEL_FORMAT_BUTT = 1003,

HI_PIXEL_FORMAT_RGB_888_PLANAR_FP16 = 1004, // RGB888 Planar 每个像素用1个float16数据表示
HI_PIXEL_FORMAT_BGR_888_PLANAR_FP16 = 1005, // BGR888 Planar 每个像素用1个float16数据表示
HI_PIXEL_FORMAT_RGB_888_PLANAR_FP32 = 1006, // RGB888 Planar 每个像素用1个float32数据表示
HI_PIXEL_FORMAT_BGR_888_PLANAR_FP32 = 1007, // BGR888 Planar 每个像素用1个float32数据表示
HI_PIXEL_FORMAT_RGB_888_PLANAR_BF16 = 1008, // RGB888 Planar 每个像素用1个BFloat16数据表示, 当前不支持该格式
HI_PIXEL_FORMAT_BGR_888_PLANAR_BF16 = 1009, // BGR888 Planar 每个像素用1个BFloat16数据表示, 当前不支持该格式
HI_PIXEL_FORMAT_RGB_888_FP16 = 1010, // RGB888 Package, 每个像素用1个float16数据表示
HI_PIXEL_FORMAT_BGR_888_FP16 = 1011, // BGR888 Package, 每个像素用1个float16数据表示
HI_PIXEL_FORMAT_RGB_888_FP32 = 1012, // RGB888 Package, 每个像素用1个float32数据表示
HI_PIXEL_FORMAT_BGR_888_FP32 = 1013, // BGR888 Package, 每个像素用1个float32数据表示
HI_PIXEL_FORMAT_RGB_888_BF16 = 1014, // RGB888 Package 每个像素用1个BFloat16数据表示, 当前
```

不支持该格式

HI_PIXEL_FORMAT_BGR_888_BF16 = 1015, // BGR888 Package 每个像素用1个BFloat16数据表示, 当前不支持该格式

HI_PIXEL_FORMAT_UNKNOWN = 10000

} hi_pixel_format;

参考信息

下文以HI_PIXEL_FORMAT_S8C1、HI_PIXEL_FORMAT_S8C2_PACKAGE、HI_PIXEL_FORMAT_S8C2_PLANAR为例说明排布格式，供参考。

图 1-28 HI_PIXEL_FORMAT_S8C1

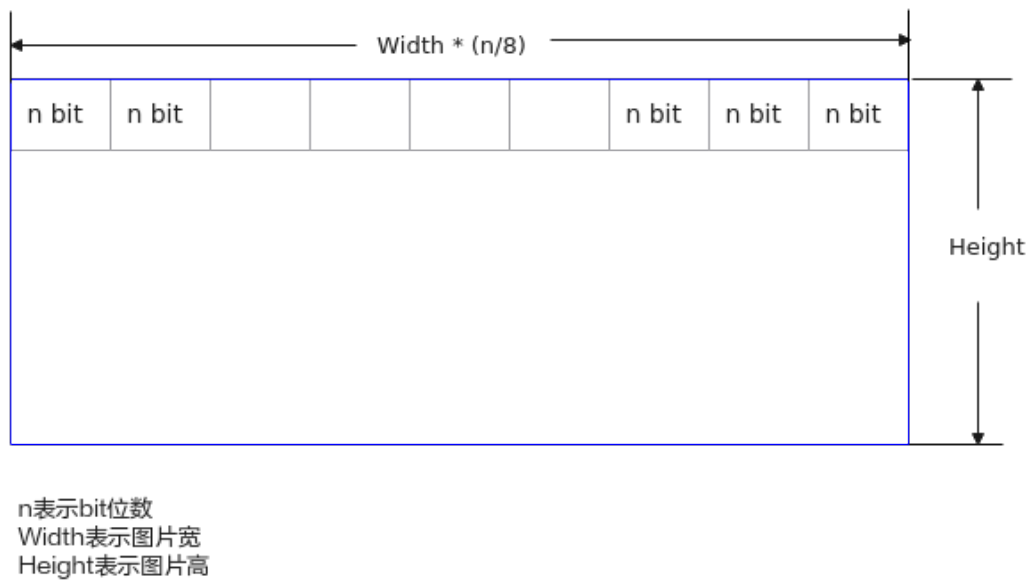


图 1-29 HI_PIXEL_FORMAT_S8C2_PACKAGE

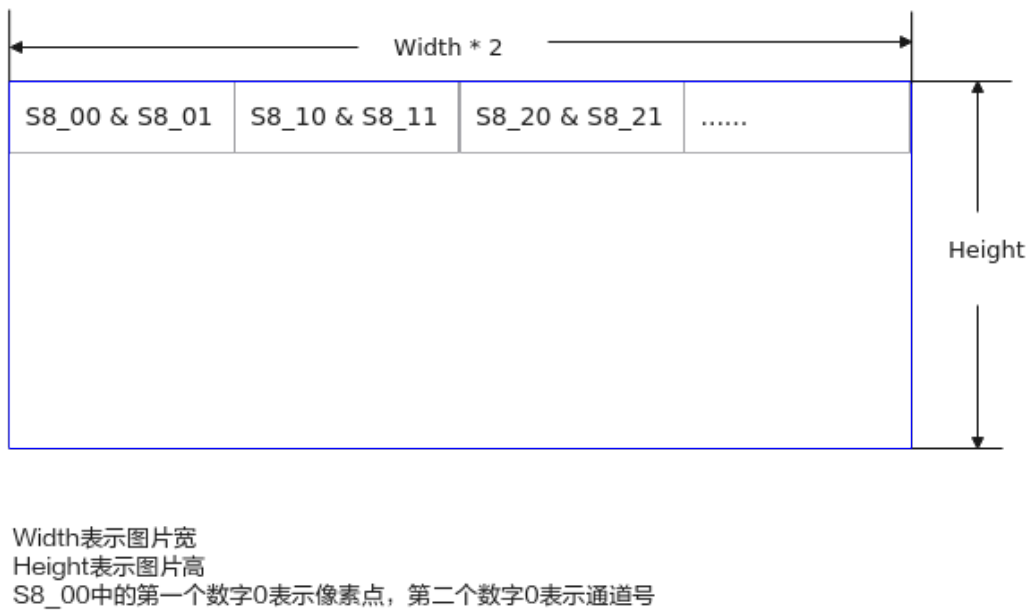
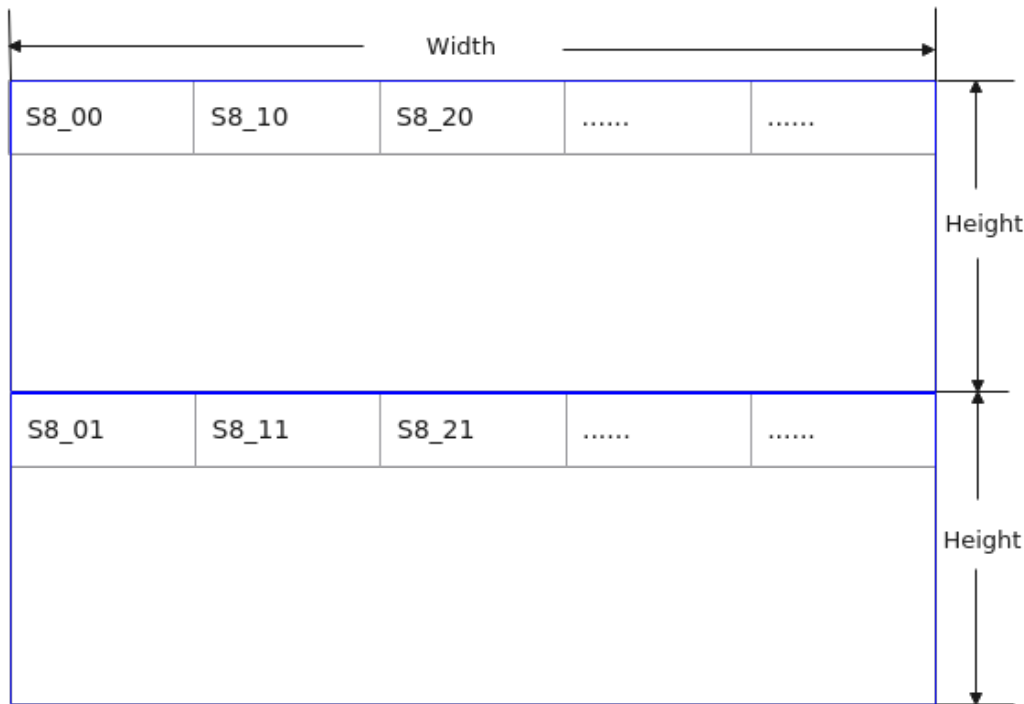


图 1-30 HI_PIXEL_FORMAT_S8C2_PLANAR



Width表示图片宽
Height表示图片高
S8_00中的第一个数字0表示像素点，第二个数字0表示通道号

1.9.20.1.3 hi_payload_type

说明

定义图像或视频类型枚举。
目前DVPP模块仅支持HI_PT_H264、HI_PT_H265、HI_PT_JPEG、HI_PT_VPC，HI_PT_PNG，其中各功能支持的图像或视频类型如下：

- VPC: HI_PT_VPC
- JPEGD: HI_PT_JPEG
- JPEGG: HI_PT_JPEG
- VDEC: HI_PT_H264、HI_PT_H265
- VENC: HI_PT_H264、HI_PT_H265
- PNGD: HI_PT_PNG

定义

```
typedef enum {  
    HI_PT_PCMU      = 0,  
    HI_PT_1016      = 1,  
    HI_PT_G721      = 2,  
    HI_PT_GSM       = 3,  
    HI_PT_G723      = 4,  
}
```

```
HI_PT_DVI4_8K      = 5,
HI_PT_DVI4_16K     = 6,
HI_PT_LPC          = 7,
HI_PT_PCMA         = 8,
HI_PT_G722         = 9,
HI_PT_S16BE_STEREO = 10,
HI_PT_S16BE_MONO   = 11,
HI_PT_QCELP        = 12,
HI_PT_CN           = 13,
HI_PT_MPEGAUDIO    = 14,
HI_PT_G728         = 15,
HI_PT_DVI4_3       = 16,
HI_PT_DVI4_4       = 17,
HI_PT_G729         = 18,
HI_PT_G711A        = 19,
HI_PT_G711U        = 20,
HI_PT_G726         = 21,
HI_PT_G729A        = 22,
HI_PT_LPCM         = 23,
HI_PT_CelB         = 25,
HI_PT_JPEG         = 26,
HI_PT_CUSM         = 27,
HI_PT_NV           = 28,
HI_PT_PICW         = 29,
HI_PT_CPV          = 30,
HI_PT_H261         = 31,
HI_PT_MPEGVIDEO    = 32,
HI_PT_MPEG2TS      = 33,
HI_PT_H263         = 34,
HI_PT_SPEG         = 35,
HI_PT_MPEG2VIDEO   = 36,
HI_PT_AAC          = 37,
HI_PT_WMA9STD      = 38,
HI_PT_HEAAC        = 39,
HI_PT_PCM_VOICE    = 40,
HI_PT_PCM_AUDIO    = 41,
HI_PT_MP3          = 43,
HI_PT_ADPCMA       = 49,
HI_PT_AEC          = 50,
HI_PT_X_LD         = 95,
HI_PT_H264         = 96,
HI_PT_D_GSM_HR     = 200,
HI_PT_D_GSM_EFR    = 201,
HI_PT_D_L8         = 202,
HI_PT_D_RED        = 203,
HI_PT_D_VDVI       = 204,
HI_PT_D_BT656      = 220,
HI_PT_D_H263_1998  = 221,
HI_PT_D_MP1S       = 222,
HI_PT_D_MP2P       = 223,
HI_PT_D_BMPEG      = 224,
HI_PT_MP4VIDEO     = 230,
HI_PT_MP4AUDIO     = 237,
HI_PT_VC1          = 238,
HI_PT_JVC_ASF      = 255,
HI_PT_D_AVI        = 256,
HI_PT_DIVX3        = 257,
HI_PT_AVS          = 258,
HI_PT_REAL8        = 259,
HI_PT_REAL9        = 260,
HI_PT_VP6          = 261,
HI_PT_VP6F         = 262,
HI_PT_VP6A         = 263,
HI_PT_SOIRENSEN    = 264,
HI_PT_H265         = 265,
HI_PT_VP8          = 266,
HI_PT_MVC          = 267,
HI_PT_PNG          = 268,
HI_PT_AMR          = 1001,
```



```
HI_PT_MJPEG      = 1002,  
HI_PT_AMRWB      = 1003,  
HI_PT_PRORES     = 1006,  
HI_PT_OPUS       = 1007,  
HI_PT_VPC        = 2000,  
HI_PT_BUTT  
} hi_payload_type;
```

1.9.20.1.4 hi_video_field

说明

定义视频图像帧场类型。

定义

```
typedef enum {  
    HI_VIDEO_FIELD_TOP = 0x1,  
    HI_VIDEO_FIELD_BOTTOM = 0x2,  
    HI_VIDEO_FIELD_INTERLACED = 0x3,  
    HI_VIDEO_FIELD_FRAME = 0x4,  
    HI_VIDEO_FIELD_BUTT  
} hi_video_field;
```

成员

成员名称	描述
HI_VIDEO_FIELD_TOP	顶场类型。
HI_VIDEO_FIELD_BOTTOM	底场类型。
HI_VIDEO_FIELD_INTERLACED	两场间插类型。
HI_VIDEO_FIELD_FRAME	帧类型。
HI_VIDEO_FIELD_BUTT	保留值。

1.9.20.1.5 hi_dynamic_range

说明

定义动态范围枚举。

定义

```
typedef enum {  
    HI_DYNAMIC_RANGE_SDR8 = 0,  
    HI_DYNAMIC_RANGE_SDR10,  
    HI_DYNAMIC_RANGE_HDR10,  
    HI_DYNAMIC_RANGE_HLG,  
    HI_DYNAMIC_RANGE_SLF,  
    HI_DYNAMIC_RANGE_XDR,
```

```
HI_DYNAMIC_RANGE_BUTT
} hi_dynamic_range;
```

成员

成员名称	描述
HI_DYNAMIC_RANGE_SDR8	8bit 数据的标准动态范围。
HI_DYNAMIC_RANGE_SDR10	10bit 数据的标准动态范围。
HI_DYNAMIC_RANGE_HDR10	10bit 数据的高动态范围。
HI_DYNAMIC_RANGE_HLG	带有混合对数的Gamma。
HI_DYNAMIC_RANGE_SLF	暂时无效。
HI_DYNAMIC_RANGE_XDR	10bit 数据，算法处理的一个中间类型数据，用户无需关心。
HI_DYNAMIC_RANGE_BUTT	保留值。

注意事项

各个动态范围对应的曲线如下：

Dynamic Range	Transfer Characteristic
SDR8/ SDR10	$V = \alpha * L_c^{0.45} - (\alpha - 1)$ for $1 \geq L_c \geq \beta$ $V = 4.500 * L_c$ for $\beta > L_c \geq 0$
HDR10	对于 L_c 的所有值， $V = ((c_1 + c_2 * L_c^n) \div (1 + c_3 * L_c^n))$ $c_1 = c_3 - c_2 + 1 = 3424 \div 4096 = 0.8359375$ $c_2 = 32 * 2413 \div 4096 = 18.8515625$ $c_3 = 32 * 2392 \div 4096 = 18.6875$ $m = 128 * 2523 \div 4096 = 78.84375$ $n = 0.25 * 2610 \div 4096 = 0.1593017578125$ 其中，峰白区 L_c 等于1通常是为了表示参考输出发光亮度，即10000坎德拉每平方米。
HLG	$V = a * \ln(12 * L_c - b) + c$ for $1 \geq L_c > 1 \div 12$ $V = \text{Sqrt}(3) * L_c^{0.5}$ for $1 \div 12 \geq L_c \geq 0$ $a = 0.17883277, b = 0.28466892, c = 0.55991073$

1.9.20.1.6 hi_color_gamut

说明

定义色域范围枚举。

定义

```
typedef enum {  
    HI_COLOR_GAMUT_BT601 = 0,  
    HI_COLOR_GAMUT_BT709,  
    HI_COLOR_GAMUT_BT2020,  
    HI_COLOR_GAMUT_USER,  
    HI_COLOR_GAMUT_BUTT  
} hi_color_gamut;
```

成员

成员名称	描述
HI_COLOR_GAMUT_BT601	BT601色域
HI_COLOR_GAMUT_BT709	BT709色域。
HI_COLOR_GAMUT_BT2020	BT2020色域。
HI_COLOR_GAMUT_USER	用户自定义的色域，非标准色域。
HI_COLOR_GAMUT_BUTT	保留值。

1.9.20.1.7 hi_video_supplement

说明

定义视频图像帧补充信息。

定义

```
typedef struct {  
    hi_u64 misc_info_phys_addr;  
    hi_u64 jpeg_dcf_phys_addr;  
    hi_u64 isp_info_phys_addr;  
    hi_u64 low_delay_phys_addr;  
    hi_u64 bnr_rnt_phys_addr;  
    hi_u64 motion_data_phys_addr;  
    union {  
        hi_u64 frame_dng_phys_addr;  
        hi_u64 offset;  
    }  
    hi_void* ATTRIBUTE misc_info_virt_addr;  
    hi_void* ATTRIBUTE jpeg_dcf_virt_addr;  
    hi_void* ATTRIBUTE isp_info_virt_addr;  
    hi_void* ATTRIBUTE low_delay_virt_addr;
```

```
hi_void* ATTRIBUTE bnr_mot_virt_addr;  
hi_void* ATTRIBUTE motion_data_virt_addr;  
hi_void* ATTRIBUTE frame_dng_virt_addr;  
} hi_video_supplement;
```

成员

成员名称	描述
misc_info_phys_addr	视频扩展信息的物理地址。暂不支持，用户不能配置修改。
jpeg_dcf_phys_addr	Jpeg DCF (Design rule for Camera File system) 信息的物理地址。暂不支持，用户不能配置，系统使用默认值0。
isp_info_phys_addr	ISP (Image Signal Processor) 辅助信息的物理地址，用户不能配置修改。
low_delay_phys_addr	低延时信息物理地址。暂不支持。
bnr_rnt_phys_addr	预留参数，暂不支持，用户不能配置修改。
motion_data_phys_addr	运动数据物理地址。暂不支持，用户不能配置修改。
frame_dng_phys_addr	预留参数，暂不支持，用户不能配置修改。
offset	图像偏移信息，仅配套hi_mpi_vi_send_pipe_raw接口使用，其它场景用户不能配置修改。
misc_info_virt_addr	视频扩展信息的虚拟地址。暂不支持，用户不能配置修改。
jpeg_dcf_virt_addr	Jpeg DCF信息的虚拟地址。暂不支持，用户不能配置修改。
isp_info_virt_addr	ISP辅助信息的内核态虚拟地址。
low_delay_virt_addr	低延时信息虚拟地址。暂不支持，用户不能配置修改。
bnr_mot_virt_addr	预留参数，暂不支持，用户不能配置修改。
motion_data_virt_addr	运动信息虚拟地址。暂不支持，用户不能配置修改。
frame_dng_virt_addr	预留参数，暂不支持，用户不能配置修改。

1.9.20.1.8 hi_video_frame

说明

定义视频原始图像帧结构。

定义

```
typedef struct {  
    hi_u32 width;
```

```
hi_u32      height;  
hi_video_field      field;  
hi_pixel_format      pixel_format;  
hi_video_format      video_format;  
hi_compress_mode      compress_mode;  
hi_dynamic_range      dynamic_range;  
hi_color_gamut      color_gamut;  
  
hi_u32      header_stride[HI_MAX_COLOR_COMPONENT];  
hi_u32      width_stride[HI_MAX_COLOR_COMPONENT];  
hi_u32      height_stride[HI_MAX_COLOR_COMPONENT];  
  
hi_u64      header_phys_addr[HI_MAX_COLOR_COMPONENT];  
hi_u64      phys_addr[HI_MAX_COLOR_COMPONENT];  
hi_void* ATTRIBUTE header_virt_addr[HI_MAX_COLOR_COMPONENT];  
hi_void* ATTRIBUTE virt_addr[HI_MAX_COLOR_COMPONENT];  
  
hi_u32      time_ref;  
hi_u64      pts;  
  
hi_u64      user_data[HI_MAX_USER_DATA_NUM];  
hi_u32      frame_flag;  
hi_video_supplement supplement;  
} hi_video_frame;
```

成员

成员名称	描述
width	图像宽度。
height	图像高度。
field	帧场模式。预留参数。
pixel_format	视频图像像素格式。
video_format	视频图像格式。
compress_mode	视频压缩模式。
dynamic_range	动态范围。预留参数。
color_gamut	色域范围。预留参数。
header_stride	图像压缩头跨距。预留参数。
width_stride	输出图像分量的宽度数据跨距。YUV图像则为Y、U、V分量的数据跨距。RGB图像则为R、G、B分量的数据跨距。
height_stride	输出图像分量的高度数据跨距。YUV图像则为Y、U、V分量的数据跨距。RGB图像则为R、G、B分量的数据跨距。 若涉及VI视频采集、ISP系统控制、VPSS视频处理、VENC视频编码、JPEG图片编码功能时，该参数作为预留参数，暂不支持。
header_phys_addr	压缩头物理地址。预留参数。
phys_addr	物理地址。预留参数。
header_virt_addr	压缩头虚拟地址。预留参数。

成员名称	描述
virt_addr	图像在Device内存中的起始虚拟地址。 作为输入图像时，对于YUV图像，virt_addr[0]为图像起始地址，也是Y分量起始地址，virt_addr[1]为U分量起始地址，virt_addr[2]为V分量起始地址； 作为输出图像时，对于YUV和RGB图像，virt_addr[0]为图像在device上的起始地址，virt_addr[1]和virt_addr[2]为保留字段。
time_ref	图像帧序列号。 解码场景下，该参数预留。 JPEGE编码场景下，该参数预留。 VENC编码场景下，针对每一帧设置该参数时，按照帧的顺序，该参数值需递增且为偶数，例如，第一帧时该参数值设置为2，第二帧时该参数值设置为4，以此类推。
pts	图像时间戳。
user_data	私有数据。预留参数。

成员名称	描述
frame_flag	● 在VDEC解码场景下，该参数表示该帧解码是否成功，不管成功或失败，用户都需及时释放输入、输出内存，对于解码失败的帧，系统不保证按照显示序或解码序按顺序输出，本参数取值范围： - 0：解码成功。 - 1：解码失败，可能由于输入码流异常、设置的码流格式与实际码流不一致等问题导致。 - 2：隔行码流场景下使用，隔行码流每帧发送两场，带两块输出内存；解码时其中一块内存无图像输出，属于正常现象，用户需及时释放内存；隔行码流的解码输出数据都在奇数场对应的输出buffer中。 - 3：参考帧个数设置错误导致解码失败。 - 4：VDEC解码帧存大小设置错误导致解码失败。 ● 在JPEGD解码场景下，该参数表示该帧解码是否成功，不管成功或失败，用户都需及时释放输入、输出内存，本参数取值范围： - 0：解码成功。 - 1：解码失败，可能由于输入码流异常、设置的码流格式与实际码流不一致等问题导致。 - 4：JPEGD解码输出内存大小过小导致解码失败，可提前调用hi_mpi_vdec_get_jpegd_output_info接口获取输出内存大小。 - 5：仅JPEGD解码场景使用，表示JPEGD解码配置参数设置错误，例如，设置的宽高超出限制、设置的目标解码格式不支持等。 ● 在VENC编码场景下，不同bit位用于使能不同的功能： - 该参数的bit0表示是否使能 slice_temporal_mvp_enabled_flag语法（表示帧间预测是否使用时域MV（MotionVector）预测），取值范围如下，但仅H265协议下该参数生效： - 0：slice_temporal_mvp_enabled_flag为True，表示使用时域MV预测。 - 1：slice_temporal_mvp_enabled_flag为False，表示不使用时域MV预测。 - 该参数的bit1表示是否使能图像扩边，取值范围如下，H264/H265协议下该参数生效： - 0：表示使能图像扩边。 - 1：表示关闭图像扩边。 ● 在JPEGE编码场景下，取值0或1，但当前版本暂未使用该参数，参数值无效。

成员名称	描述
supplement	图像的补充信息。 Atlas 200/500 A2推理产品上，VI模块接口会使用该参数，并由VI模块内部自动完成初始化，其它功能该参数为透传参数。 Atlas 推理系列产品上，该参数为透传参数。 Atlas A2训练系列产品/Atlas 800I A2推理产品上，该参数为透传参数。

注意事项

10bit数据不压缩时在内存中的存储方式是紧凑排列的。

10bit YUV段压缩数据存储是前8bit和后2bit分开存储。

10bit tile 64*16 数据（即VDH（video decoder hardware）解码之后的数据）不管是压缩还是非压缩，在内存中的存储方式都是前8bit和后2bit分开存储。

1.9.20.1.9 hi_video_frame_info

说明

定义视频图像帧信息结构体。

定义

```
typedef struct {  
    hi_video_frame v_frame;  
    hi_u32 pool_id;  
    hi_mod_id mod_id;  
} hi_video_frame_info;
```

成员

成员名称	描述
v_frame	视频图像帧属性结构体。
pool_id	视频缓存池ID，预留参数。
mod_id	DVPP内的子模块ID，预留参数。

1.9.20.1.10 hi_vb_src

说明

定义VB来源选择。

定义

```
typedef enum {  
    HI_VB_SRC_COMMON = 0, //公共VB  
    HI_VB_SRC_MOD    = 1, //模块VB  
    HI_VB_SRC_PRIVATE = 2, //私有VB  
    HI_VB_SRC_USER    = 3, //用户VB  
    HI_VB_SRC_BUTT  
} hi_vb_src;
```

1.9.20.1.11 hi_mod_id

说明

定义模块ID枚举类型。

定义

```
typedef enum {  
    HI_ID_CMPI = 0,  
    HI_ID_VB = 1,  
    HI_ID_SYS = 2,  
    HI_ID_RGN = 3,  
    HI_ID_CHNL = 4,  
    HI_ID_VDEC = 5,  
    HI_ID_AVS = 6,  
    HI_ID_VPC = 7,  
    HI_ID_VENC = 8,  
    HI_ID_SVP = 9,  
    HI_ID_H264E = 10,  
    HI_ID_JPEGE = 11,  
    HI_ID_MPEG4E = 12,  
    HI_ID_H265E = 13,  
    HI_ID_JPEGD = 14,  
    HI_ID_VO = 15,  
    HI_ID_VI = 16,  
    HI_ID_DIS = 17,  
    HI_ID_VALG = 18,  
    HI_ID_RC = 19,  
    HI_ID_AIO = 20,  
    HI_ID_AI = 21,  
    HI_ID_AO = 22,  
    HI_ID_AENC = 23,  
    HI_ID_ADEC = 24,  
    HI_ID_VPU = 25,  
    HI_ID_PCIV = 26,  
    HI_ID_PCIVFMW = 27,  
    HI_ID_ISP = 28,  
    HI_ID_IVE = 29,  
    HI_ID_USER = 30,  
    HI_ID_DCCM = 31,  
    HI_ID_DCCS = 32,  
    HI_ID_PROC = 33,  
    HI_ID_LOG = 34,  
    HI_ID_VFMW = 35,  
    HI_ID_H264D = 36,  
    HI_ID_GDC = 37,  
    HI_ID_PHOTO = 38,  
    HI_ID_FB = 39,  
    HI_ID_HDMI = 40,  
    HI_ID_VOIE = 41,  
    HI_ID_TDE = 42,  
    HI_ID_HDR = 43,  
    HI_ID_PRORES = 44,  
    HI_ID_VGS = 45,  
  
    HI_ID_FD = 47,
```

```
HI_ID_ODT = 48,  
HI_ID_VQA = 49,  
HI_ID_LPR = 50,  
HI_ID_SVP_NNIE = 51,  
HI_ID_SVP_DSP = 52,  
HI_ID_DPU_RECT = 53,  
HI_ID_DPU_MATCH = 54,  
  
HI_ID_MOTIONSENSOR = 55,  
HI_ID_MOTIONFUSION = 56,  
  
HI_ID_GYRODIS = 57,  
HI_ID_PM = 58,  
HI_ID_SVP_ALG = 59,  
HI_ID_IVP = 60,  
HI_ID_MCF = 61,  
HI_ID_VPSS = 62,  
HI_ID_DRV_VPC = 63,  
HI_ID_PNGD = 64,  
HI_ID_BUTT = 0x100,  
} hi_mod_id;
```

1.9.20.1.12 hi_data_bit_width

说明

定义数据比特位宽类型。

目前DVPP各功能仅支持HI_DATA_BIT_WIDTH_8和HI_DATA_BIT_WIDTH_10两个数据位宽格式。

目前VI功能支持HI_DATA_BIT_WIDTH_8、HI_DATA_BIT_WIDTH_10、HI_DATA_BIT_WIDTH_12、HI_DATA_BIT_WIDTH_14几种数据位宽格式。

定义

```
typedef enum {  
    HI_DATA_BIT_WIDTH_8 = 0,  
    HI_DATA_BIT_WIDTH_10,  
    HI_DATA_BIT_WIDTH_12,  
    HI_DATA_BIT_WIDTH_14,  
    HI_DATA_BIT_WIDTH_16,  
    HI_DATA_BIT_WIDTH_BUTT  
} hi_data_bit_width;
```

1.9.20.1.13 hi_video_format

说明

定义视频格式枚举。

定义

```
typedef enum {  
    HI_VIDEO_FORMAT_LINEAR = 0,  
    HI_VIDEO_FORMAT_TILE_64x16,  
    HI_VIDEO_FORMAT_BUTT  
} hi_video_format;
```

成员

成员名称	描述
HI_VIDEO_FORMAT_LINEAR	线性存储的视频格式。
HI_VIDEO_FORMAT_TILE_64x16	TILE 格式存储的视频格式，其中，tile的块大小为64像素宽，16行像素高，视频/图像编码暂不支持。
HI_VIDEO_FORMAT_BUTT	保留值。

1.9.20.1.14 hi_img_info

说明

定义图片信息的结构体。

定义

```
typedef struct {  
    hi_u32 width;  
    hi_u32 height;  
    hi_u32 width_stride;  
    hi_u32 height_stride;  
    hi_u32 img_buf_size;  
    union {  
        hi_jpeg_raw_format pixel_format;  
        hi_png_color_format png_pixel_format;  
        hi_pixel_format img_pixel_format;  
    };  
    hi_u32 reserved[4];  
} hi_img_info;
```

成员

成员名称	描述
width	图片宽。
height	图片高。
width_stride	图片宽stride。
height_stride	图片高stride。
img_buf_size	存放图片的内存大小，单位是Byte。
pixel_format	JPEG图片格式。
png_pixel_format	PNG图片格式。
img_pixel_format	通用图片格式。

成员名称	描述
reserved	预留参数，为保证后续版本兼容性，请务必使用memset结构体方式进行清零初始化，在代码中必须避免显式对reserved字段进行访问。

1.9.20.1.15 hi_pic_buf_attr

说明

定义图片缓冲区信息的结构体。

定义

```
typedef struct {  
    hi_u32 width;  
    hi_u32 height;  
    hi_u32 align;  
    hi_data_bit_width bit_width;  
    hi_pixel_format pixel_format;  
    hi_compress_mode compress_mode;  
} hi_pic_buf_attr;
```

成员

成员名称	描述
width	图片宽。
height	图片高。
align	对齐参数，预留参数，暂不支持。
bit_width	图片位宽。
pixel_format	图片格式。 VDEC视频解码时，暂不使用该参数；JPEGD图像解码时，表示解码输出图像格式，当前支持的输出图像格式请参见 1.9.17.1 JPEGD功能及约束说明 。
compress_mode	压缩模式。

1.9.20.1.16 hi_jpeg_raw_format

说明

定义JPEG源图片格式的枚举。

定义

```
typedef enum {  
    HI_JPEG_RAW_FORMAT_YUV444 = 0,  
    HI_JPEG_RAW_FORMAT_YUV422 = 1,  
};
```

```
HI_JPEG_RAW_FORMAT_YUV420 = 2,  
HI_JPEG_RAW_FORMAT_YUV440 = 3,  
HI_JPEG_RAW_FORMAT_YUV400 = 4,  
HI_JPEG_RAW_FORMAT_YUV411 = 5,  
HI_JPEG_RAW_FORMAT_MAX = 100  
} hi_jpeg_raw_format;
```

1.9.20.1.17 hi_dvpp_epoll_ctl_op

说明

定义DVPP epoll操作类型。

定义

```
typedef enum {  
    HI_DVPP_EPOLL_CTL_ADD = 1,  
    HI_DVPP_EPOLL_CTL_MOD = 2,  
    HI_DVPP_EPOLL_CTL_DEL = 3,  
    HI_DVPP_EPOLL_CTL_BUTT  
} hi_dvpp_epoll_ctl_op;
```

成员

成员名称	描述
HI_DVPP_EPOLL_CTL_ADD	向epoll实例中添加新的DVPP通道的文件句柄。
HI_DVPP_EPOLL_CTL_MOD	修改epoll实例中已有DVPP通道的事件类型或对应的用户数据。
HI_DVPP_EPOLL_CTL_DEL	从epoll实例中删除DVPP通道的文件句柄。
HI_DVPP_EPOLL_CTL_BUTT	保留值。

1.9.20.1.18 hi_dvpp_epoll_event_type

说明

定义DVPP epoll事件类型。

定义

```
typedef enum {  
    HI_DVPP_EPOLL_IN = 1u,  
    HI_DVPP_EPOLL_OUT = 1u << 1u,  
    HI_DVPP_EPOLL_ET = (hi_u32)1u << 31u  
} hi_dvpp_epoll_event_type;
```

成员

成员名称	描述
HI_DVPP_EPOLL_IN	可读事件，标识DVPP通道中存在输出数据。
HI_DVPP_EPOLL_OUT	可写事件，标识DVPP通道可以进行新的数据输入。预留功能，暂不支持。
HI_DVPP_EPOLL_ET	边缘触发方式，不设定则DVPP epoll默认为水平触发。

1.9.20.1.19 hi_dvpp_epoll_event

说明

定义DVPP epoll事件信息。

定义

```
typedef struct {  
    hi_u32 events;  
    hi_void *data;  
} hi_dvpp_epoll_event;
```

成员

成员名称	描述
events	DVPP epoll事件类型，由hi_dvpp_epoll_event_type中定义的类型组成的位掩码（bitmask），例如处理可读和可写两种事件时，此处配置为：HI_DVPP_EPOLL_IN HI_DVPP_EPOLL_OUT。
data	关联到DVPP通道的用户自定义数据。

1.9.20.1.20 hi_csc_matrix

说明

定义色域转换矩阵的数据标准。
基于各标准的色域转换矩阵，其各分量的转换公式如下：

```
# YUV转RGB:  
# | R | | matrix_r0c0 matrix_r0c1 matrix_r0c2 | | Y - input_bias_0 |  
# | G | = | matrix_r1c0 matrix_r1c1 matrix_r1c2 | * | U - input_bias_1 |  
# | B | | matrix_r2c0 matrix_r2c1 matrix_r2c2 | | V - input_bias_2 |  
# BGR转YUV:  
# | Y | | output_bias_0 | | matrix_r0c0 matrix_r0c1 matrix_r0c2 | | R |  
# | U | = | output_bias_1 | + | matrix_r1c0 matrix_r1c1 matrix_r1c2 | * | G |  
# | V | | output_bias_2 | | matrix_r2c0 matrix_r2c1 matrix_r2c2 | | B |
```

定义

```
typedef enum {  
    HI_CSC_MATRIX_BT601_WIDE = 0,  
    HI_CSC_MATRIX_BT601_NARROW,  
    HI_CSC_MATRIX_BT709_WIDE,  
    HI_CSC_MATRIX_BT709_NARROW,  
    HI_CSC_MATRIX_BT2020_WIDE,  
    HI_CSC_MATRIX_BT2020_NARROW,  
    HI_CSC_MATRIX_USER = 100,  
    HI_CSC_MATRIX_BUTT  
} hi_csc_matrix;
```

成员

表 1-80 色域转换矩阵的数据标准

成员名称	描述
HI_CSC_MATRIX_BT601_WIDE	基于BT601 wide标准的色域转换矩阵。默认为HI_CSC_MATRIX_BT601_WIDE。 基于该标准的色域转换矩阵，各参数值如下： # YUV转RGB： # R 1.000 0.000 1.402 Y - 0 # G = 1.000 -0.344 -0.714 * U - 128 # B 1.000 1.772 0.000 V - 128 # RGB转YUV： # Y -0.5 0.299 0.587 0.114 R # U = 127.5 + -0.168 -0.331 0.500 * G # V 127.5 0.500 -0.419 -0.081 B
HI_CSC_MATRIX_BT601_NARROW	基于BT601 narrow标准的色域转换矩阵。 基于该标准的色域转换矩阵，各参数值如下： # YUV转RGB： # R 1.16438 0.00000 1.59602 Y - 16 # G = 1.16438 -0.39176 -0.81297 * U - 128 # B 1.16438 2.01723 0.00000 V - 128 # RGB转YUV： # Y 16 0.25679 0.51564 0.10014 R # U = 128 + -0.14491 -0.29099 0.43922 * G # V 128 0.42941 -0.36779 -0.07143 B
HI_CSC_MATRIX_BT709_WIDE	基于BT709 wide标准的色域转换矩阵。 基于该标准的色域转换矩阵，各参数值如下： # YUV转RGB： # R 1.00000 0.00000 1.57480 Y - 0 # G = 1.00000 -0.18732 -0.46812 * U - 128 # B 1.00000 1.85560 0.00000 V - 128 # RGB转YUV： # Y 0 0.21260 0.71520 0.07220 R # U = 128 + -0.11457 -0.38543 0.50000 * G # V 128 0.50000 -0.45415 -0.04585 B
HI_CSC_MATRIX_BT709_NARROW	基于BT709 narrow标准的色域转换矩阵。 基于该标准的色域转换矩阵，各参数值如下： # YUV转RGB： # R 1.16438 0.00000 1.79274 Y - 16 # G = 1.16438 -0.21325 -0.53291 * U - 128 # B 1.16438 2.11240 0.00000 V - 128 # RGB转YUV： # Y 16 0.18259 0.62825 0.06342 R # U = 128 + -0.09840 -0.33857 0.43922 * G # V 128 0.42941 -0.39894 -0.04027 B

成员名称	描述
HI_CSC_MATRIX_BT2020_WIDE	基于BT2020 wide标准的色域转换矩阵。 基于该标准的色域转换矩阵，各参数值如下： # YUV转RGB： # R 1.00000 0.00000 1.47460 Y - 0 # G = 1.00000 -0.16455 -0.57135 * U - 128 # B 1.00000 1.88140 0.00000 V - 128 # RGB转YUV： # Y 0 0.26270 0.67800 0.05930 R # U = 128 + -0.13963 -0.36037 0.50000 * G # V 128 0.50000 -0.45979 -0.04021 B
HI_CSC_MATRIX_BT2020_NARROW	基于BT2020 narrow标准的色域转换矩阵。 基于该标准的色域转换矩阵，各参数值如下： # YUV转RGB： # R 1.16438 0.00000 1.67868 Y - 16 # G = 1.16438 -0.18733 -0.65042 * U - 128 # B 1.16438 2.14177 0.00000 V - 128 # RGB转YUV： # Y 16 0.22564 0.59558 0.05209 R # U = 128 + -0.11992 -0.31656 0.43922 * G # V 128 0.42941 -0.40389 -0.03533 B
HI_CSC_MATRIX_USER	用户自定义色域转换矩阵。
HI_CSC_MATRIX_BUTT	保留值。

1.9.20.1.21 hi_coefficient

说明

定义色域转换矩阵参数。

定义

```
typedef struct {  
    hi_double csc_matrix_r0_c0;  
    hi_double csc_matrix_r0_c1;  
    hi_double csc_matrix_r0_c2;  
    hi_double csc_matrix_r1_c0;  
    hi_double csc_matrix_r1_c1;  
    hi_double csc_matrix_r1_c2;  
    hi_double csc_matrix_r2_c0;  
    hi_double csc_matrix_r2_c1;  
    hi_double csc_matrix_r2_c2;  
    hi_double csc_bias_r0;  
    hi_double csc_bias_r1;  
    hi_double csc_bias_r2;  
} hi_coefficient;
```

成员

成员名称	描述
csc_matrix_r0_c0	设置色域转换矩阵参数R0C0。
csc_matrix_r0_c1	设置色域转换矩阵参数R0C1。

成员名称	描述
csc_matrix_r0_c2	设置色域转换矩阵参数R0C2。
csc_matrix_r1_c0	设置色域转换矩阵参数R1C0。
csc_matrix_r1_c1	设置色域转换矩阵参数R1C1。
csc_matrix_r1_c2	设置色域转换矩阵参数R1C2。
csc_matrix_r2_c0	设置色域转换矩阵参数R2C0。
csc_matrix_r2_c1	设置色域转换矩阵参数R2C1。
csc_matrix_r2_c2	设置色域转换矩阵参数R2C2。
csc_bias_r0	设置偏移R0。
csc_bias_r1	设置偏移R1。
csc_bias_r2	设置偏移R2。

1.9.20.1.22 hi_csc_coefficient

说明

定义色域转换矩阵参数的结构体。

定义

```
typedef struct {  
    hi_coefficient yuv_to_rgb_coefficient;  
    hi_coefficient rgb_to_yuv_coefficient;  
} hi_csc_coefficient;
```

成员

成员名称	描述
yuv_to_rgb_coefficient	yuv转rgb的色域转换参数。
rgb_to_yuv_coefficient	rgb转yuv的色域转换参数。

1.9.20.1.23 hi_video_size

说明

预留结构体，暂不支持。

定义

```
typedef struct {  
    hi_u32 width;
```

```
    hi_u32 height;  
} hi_video_size;
```

1.9.20.1.24 hi_img_stream

说明

定义图片码流信息。

定义

```
typedef struct {  
    hi_payload_type type;  
    hi_u8 *ATTRIBUTE addr;  
    hi_u32 len;  
    hi_u64 pts;  
    hi_s32 reserved[2];  
} hi_img_stream;
```

成员

成员名称	描述
type	图片类型。
addr	图片在内存中的起始虚拟地址。
len	存放图片的内存大小，单位Byte。
pts	图片时间戳，预留参数。
reserved	预留参数，为保证后续版本兼容性，请务必使用memset结构体方式进行清零初始化，在代码中必须避免显式对reserved字段进行访问。

1.9.20.1.25 hi_pic_info

说明

定义图片信息。

定义

```
typedef struct {  
    hi_void* picture_address;  
    hi_u32 picture_buffer_size;  
    hi_u32 picture_width;  
    hi_u32 picture_height;  
    hi_u32 picture_width_stride;  
    hi_u32 picture_height_stride;  
    hi_pixel_format picture_format;  
} hi_pic_info;
```

成员

成员名称	描述
picture_address	图片在Device内存中的起始虚拟地址。
picture_buffer_size	存放图片的内存大小，单位Byte。
picture_width	图片宽。
picture_height	图片高。
picture_width_stride	图片宽stride。
picture_height_stride	图片高stride。
picture_format	图片格式。

1.9.20.1.26 hi_png_color_format

说明

定义PNG源图片格式的枚举。

定义

```
typedef enum {  
    HI_PNG_COLOR_FORMAT_GRAY = 0x0, // gray bitmap  
    HI_PNG_COLOR_FORMAT_RGB = 0x2, // RGB bitmap  
    HI_PNG_COLOR_FORMAT_CLUT = 0x3, // PLTE调色板数据  
    HI_PNG_COLOR_FORMAT_AGRAY = 0x4, // gray bitmap with alpha  
    HI_PNG_COLOR_FORMAT_ARGB = 0x6, // RGB bitmap with alpha  
    HI_PNG_COLOR_FORMAT_BUTT = 0x100  
} hi_png_color_format;
```

1.9.20.1.27 hi_rect

说明

定义矩形宽高和左上角坐标。

定义

```
typedef struct {  
    hi_s32 x;  
    hi_s32 y;  
    hi_s32 width;  
    hi_s32 height;  
} hi_rect;
```

成员

成员名称	描述
x	左上角x轴

成员名称	描述
y	左上角y轴
width	矩形宽度
height	矩形高度

1.9.20.1.28 hi_op_mode

说明

定义操作模式枚举。

定义

```
typedef enum {  
    HI_OP_MODE_AUTO = 0, // 自动模式，一般此模式使用程序内部的默认参数  
    HI_OP_MODE_MANUAL = 1 // 手动模式，一般此模式使用用户配置的参数  
} hi_op_mode;
```

1.9.20.1.29 hi_point

说明

定义坐标点结构体。

定义

```
typedef struct {  
    hi_s32 x;  
    hi_s32 y;  
} hi_point;
```

成员

成员名称	描述
x	x轴坐标值。
y	y轴坐标值。

1.9.20.1.30 hi_crop_info

说明

裁剪CROP属性结构体。

定义

```
typedef struct {  
    hi_bool enable;
```

```
    hi_rect rect;  
} hi_crop_info;
```

成员

成员名称	描述
enable	是否开启裁剪功能。 取值范围： - HI_FALSE：不开启。 - HI_TRUE：开启。
rect	裁剪起始位置与宽高信息。

1.9.20.1.31 hi_frame_rate_ctrl

说明

定义帧率控制结构体 。

定义

```
typedef struct {  
    hi_s32 src_frame_rate;  
    hi_s32 dst_frame_rate;  
} hi_frame_rate_ctrl;
```

成员

成员名称	描述
src_frame_rate	源帧率，即输入帧率。 取值范围-1或[1, 240]。
dst_frame_rate	目的帧率，即输出帧率。当源帧率为-1时，目标帧率必须为-1，表示不进行帧率控制，其它情况下，目标帧率不能大于源帧率。 取值范围[-1, src_frame_rate]。

1.9.20.1.32 hi_compress_mode

说明

定义视频压缩数据格式枚举。

定义

```
typedef enum {  
    HI_COMPRESS_MODE_NONE = 0,  
    HI_COMPRESS_MODE_SEG,  
    HI_COMPRESS_MODE_TILE,
```

```
HI_COMPRESS_MODE_HFBC,  
HI_COMPRESS_MODE_LINE,  
HI_COMPRESS_MODE_FRAME,  
HI_COMPRESS_MODE_BUTT  
} hi_compress_mode;
```

成员

成员名称	描述
HI_COMPRESS_MODE_NONE	不压缩。
HI_COMPRESS_MODE_SEG	段压缩，以256字节为一个压缩片段。
HI_COMPRESS_MODE_TILE	Tile压缩，以一个tile为一个压缩片段。
HI_COMPRESS_MODE_HFBC	本昇腾AI处理器特有的HFBC压缩模式，压缩单位是64*8。
HI_COMPRESS_MODE_LINE	行压缩，以一整行为一个压缩片段进行压缩。如ISP WDR模式下内部的RAW图像采用行压缩。
HI_COMPRESS_MODE_FRAME	帧压缩，以一整帧为一个压缩片段进行压缩。如ISP线性模式下内部的RAW图像采用帧压缩，或者3DNR内部参考帧和重构帧采用帧压缩。
HI_COMPRESS_MODE_BUTT	预留值。

1.9.20.1.33 hi_mpp_chn

说明

定义模块设备通道结构体。

定义

```
typedef struct {  
    hi_mod_id mod_id;  
    hi_s32 dev_id;  
    hi_s32 chn_id;  
} hi_mpp_chn;
```

成员

成员名称	描述
mod_id	模块号。
dev_id	设备号。
chn_id	通道号。

1.9.20.1.34 hi_size

说明

定于矩形尺寸的结构体。

定义

```
typedef struct {  
    hi_u32 width;  
    hi_u32 height;  
} hi_size;
```

成员

成员名称	描述
width	宽度
height	高度

1.9.20.1.35 hi_module_type

说明

定义模块类型。

定义

```
typedef enum {  
    HI_MOD_VPC = 0,  
    HI_MOD_VENC = 1,  
    HI_MOD_VDEC = 2,  
    HI_MOD_JPEGE = 3,  
    HI_MOD_JPEGD = 4,  
    HI_MOD_PNGD = 5,  
    HI_MOD_ALL = 100,  
    HI_MOD_BUTT = 255 // 预留给  
} hi_module_type;
```

1.9.20.1.36 hi_img_base_info

说明

定义图片基本信息。

定义

```
typedef struct {  
    hi_u32 width;  
    hi_u32 height;  
    hi_pixel_format pixel_format;  
    hi_u64 reserved[2];  
} hi_img_base_info;
```

成员

成员名称	描述
width	图片宽度。
height	图片高度。
pixel_format	图片格式。
reserved	预留参数，为保证后续版本兼容性，请务必使用memset结构体方式进行清零初始化，在代码中必须避免显式对reserved字段进行访问。

1.9.20.1.37 hi_img_align_info

说明

定义图片对齐信息。

定义

```
typedef struct {  
    hi_u32 width_stride;  
    hi_u32 height_stride;  
    hi_u64 img_buf_size;  
    hi_u64 reserved[2];  
} hi_img_align_info;
```

成员

成员名称	描述
width_stride	图片宽度对齐后的值。
height_stride	图片高度对齐后的值。
img_buf_size	存放图片数据所需的内存大小，单位Byte。
reserved	预留参数，为保证后续版本兼容性，请务必使用memset结构体方式进行清零初始化，在代码中必须避免显式对reserved字段进行访问。

1.9.20.1.38 hi_coord

说明

定义矩形区域信息结构体。

定义

```
typedef enum {  
    HI_COORD_ABS = 0,
```



```
HI_COORD_RATIO
} hi_coord;
```

成员

成员名称	描述
HI_COORD_ABS	绝对坐标。
HI_COORD_RATIO	相对坐标，即坐标值是以与当前图像宽高的比率来表示，具体比率转换与实际使用接口为准。

1.9.20.1.39 hi_aspect_ratio_type

说明

定义幅型比类型。

定义

```
typedef enum {
    HI_ASPECT_RATIO_NONE = 0,      /* full screen */
    HI_ASPECT_RATIO_AUTO = 1,      /* ratio no change, 1:1*/
    HI_ASPECT_RATIO_MANUAL = 2
} hi_aspect_ratio_type;
```

成员

成员名称	描述
HI_ASPECT_RATIO_NONE	无幅型比。
HI_ASPECT_RATIO_AUTO	自动模式。
HI_ASPECT_RATIO_MANUAL	手动模式。

1.9.20.1.40 hi_aspect_ratio

说明

定义坐标点信息结构体。

定义

```
typedef struct {
    hi_aspect_ratio_type mode; /* aspect ratio mode: none/auto/manual */
    hi_u32 bg_color; /* background color, RGB 888 */
    hi_rect video_rect; /* valid in ASPECT_RATIO_MANUAL mode */
} hi_aspect_ratio;
```

成员

成员名称	描述
mode	幅型比类型。
bg_color	幅型比背景颜色。
video_rect	幅型比视频区域。

1.9.20.1.41 hi_video_display_mode

说明

显示模式枚举。

定义

```
typedef enum {  
    HI_VIDEO_DISPLAY_MODE_PREVIEW = 0,  
    HI_VIDEO_DISPLAY_MODE_PLAYBACK = 1  
}hi_video_display_mode;
```

成员

成员名称	描述
HI_VIDEO_DISPLAY_MODE_PREVIEW	预览模式。
HI_VIDEO_DISPLAY_MODE_PLAYBACK	回放模式。

1.9.20.1.42 hi_wdr_mode

说明

定义宽动态模式。

定义

```
typedef enum {  
    HI_WDR_MODE_NONE = 0,  
    HI_WDR_MODE_BUILT_IN,  
    HI_WDR_MODE_QUDRA,  
    HI_WDR_MODE_2To1_LINE,  
    HI_WDR_MODE_2To1_FRAME,  
    HI_WDR_MODE_3To1_LINE,  
    HI_WDR_MODE_3To1_FRAME,  
    HI_WDR_MODE_4To1_LINE,  
    HI_WDR_MODE_4To1_FRAME,  
    HI_WDR_MODE_BUTT,  
} hi_wdr_mode;
```

成员

成员名称	描述
HI_WDR_MODE_NONE	线性模式。
HI_WDR_MODE_BUILT_IN	Sensor合成WDR模式。
HI_WDR_MODE_QUDRA	QUDRA WDR模式。当前版本不支持。
HI_WDR_MODE_2To1_LINE	2 帧合成行WDR模式。
HI_WDR_MODE_2To1_FRAME	2 帧合成帧WDR模式。
HI_WDR_MODE_3To1_LINE	3 帧合成行WDR模式。当前版本不支持。
HI_WDR_MODE_3To1_FRAME	3 帧合成帧WDR模式。当前版本不支持。
HI_WDR_MODE_4To1_LINE	4 帧合成行WDR模式。当前版本不支持。
HI_WDR_MODE_4To1_FRAME	4 帧合成帧WDR模式。当前版本不支持。
HI_WDR_MODE_BUTT	预留值。

 说明

此处的WDR模式配置，需要与VI、Sensor、MIPI RX配置保持一致，否则会导致出图异常。

1.9.20.1.43 hi_vb_pool

说明

定义视频缓存池ID。

定义

```
typedef hi_u32 hi_vb_pool;
```

成员

成员名称	描述
hi_vb_pool	视频缓存池ID。

1.9.20.1.44 hi_vb_blk

说明

定义内存块handle。

定义

```
typedef hi_u32 hi_vb_blk;
```

成员

成员名称	描述
hi_vb_blk	内存块handle。

1.9.20.1.45 hi_vb_remap_mode

说明

定义物理地址与虚拟地址的映射关系。

定义

```
typedef enum hiVB_REMAP_MODE_E {  
    VB_REMAP_MODE_NONE = 0,  
    VB_REMAP_MODE_NOCACHE = 1,  
    VB_REMAP_MODE_CACHED = 2,  
} hi_vb_remap_mode;
```

成员

成员名称	描述
VB_REMAP_MODE_NONE	不做映射。
VB_REMAP_MODE_NOCACHE	预留值。
VB_REMAP_MODE_CACHED	预留值。

1.9.20.1.46 hi_vb_pool_config

说明

定义视频缓存池配置信息。

定义

```
typedef struct {  
    hi_u64 blk_size;
```

```
hi_u32 blk_cnt;  
hi_vb_remap_mode remap_mode;  
hi_char mmz_name[HI_MAX_MMZ_NAME_LEN];  
} hi_vb_pool_config;
```

成员

成员名称	描述
blk_size	内存块大小，单位Byte。
blk_cnt	内存块数量。
remap_mode	当前固定配置成VB_REMAP_MODE_NONE。
mmz_name	内存池名称。 HI_MAX_MMZ_NAME_LEN表示内存名称的最大长度，为32。

1.9.20.2 音频相关

1.9.20.2.1 基本数据类型

```
typedef hi_s32 hi_audio_dev;  
typedef hi_s32 hi_ai_chn;  
typedef hi_s32 hi_ao_chn;  
typedef hi_s32 hi_aenc_chn;  
typedef hi_s32 hi_adec_chn;
```

1.9.20.2.2 hi_audio_sample_rate

说明

定义音频采样率。

定义

```
typedef enum {  
    HI_AUDIO_SAMPLE_RATE_8000 = 8000, /* 8kHz sample rate */  
    HI_AUDIO_SAMPLE_RATE_12000 = 12000, /* 12kHz sample rate */  
    HI_AUDIO_SAMPLE_RATE_11025 = 11025, /* 11.025kHz sample rate */  
    HI_AUDIO_SAMPLE_RATE_16000 = 16000, /* 16kHz sample rate */  
    HI_AUDIO_SAMPLE_RATE_22050 = 22050, /* 22.05kHz sample rate */  
    HI_AUDIO_SAMPLE_RATE_24000 = 24000, /* 24kHz sample rate */  
    HI_AUDIO_SAMPLE_RATE_32000 = 32000, /* 32kHz sample rate */  
    HI_AUDIO_SAMPLE_RATE_44100 = 44100, /* 44.1kHz sample rate */  
    HI_AUDIO_SAMPLE_RATE_48000 = 48000, /* 48kHz sample rate */  
    HI_AUDIO_SAMPLE_RATE_64000 = 64000, /* 64kHz sample rate */  
    HI_AUDIO_SAMPLE_RATE_96000 = 96000, /* 96kHz sample rate */  
    HI_AUDIO_SAMPLE_RATE_BUTT  
} hi_audio_sample_rate;
```

成员

成员名称	描述
HI_AUDIO_SAMPLE_RATE_8000	8kHz采样率。当前版本不支持。
HI_AUDIO_SAMPLE_RATE_12000	12kHz采样率。当前版本不支持。
HI_AUDIO_SAMPLE_RATE_11025	11.025kHz采样率。当前版本不支持。
HI_AUDIO_SAMPLE_RATE_16000	16kHz采样率。当前版本不支持。
HI_AUDIO_SAMPLE_RATE_22050	22.050kHz采样率。当前版本不支持。
HI_AUDIO_SAMPLE_RATE_24000	24kHz采样率。当前版本不支持。
HI_AUDIO_SAMPLE_RATE_32000	32kHz采样率。当前版本不支持。
HI_AUDIO_SAMPLE_RATE_44100	44.1kHz采样率。当前版本不支持。
HI_AUDIO_SAMPLE_RATE_48000	48kHz采样率。
HI_AUDIO_SAMPLE_RATE_64000	64kHz采样率。当前版本不支持。
HI_AUDIO_SAMPLE_RATE_96000	96kHz采样率。当前版本不支持。
HI_AUDIO_SAMPLE_RATE_BUTT	预留值。

1.9.20.2.3 hi_aio_mode

说明

定义音频输入输出设备工作模式。

定义

```
typedef enum {  
    HI_AIO_MODE_I2S_MASTER = 0, /* AIO I2S master mode */  
    HI_AIO_MODE_I2S_SLAVE, /* AIO I2S slave mode */  
    HI_AIO_MODE_PCM_SLAVE_STD, /* AIO PCM slave standard mode */  
    HI_AIO_MODE_PCM_SLAVE_NON_STD, /* AIO PCM slave non-standard mode */  
    HI_AIO_MODE_PCM_MASTER_STD, /* AIO PCM master standard mode */  
    HI_AIO_MODE_PCM_MASTER_NON_STD, /* AIO PCM master non-standard mode */  
    HI_AIO_MODE_BUTT  
} hi_aio_mode;
```

成员

成员名称	描述
HI_AIO_MODE_I2S_MASTER	I2S(Integrated Interchip Sound)主模式。
HI_AIO_MODE_I2S_SLAVE	I2S从模式。当前版本不支持。
HI_AIO_MODE_PCM_SLAVE_STD	PCM(Pulse Code Modulation)从模式（标准协议）。当前版本不支持。
HI_AIO_MODE_PCM_SLAVE_NON_STD	PCM从模式（自定义协议）。当前版本不支持。
HI_AIO_MODE_PCM_MASTER_STD	PCM主模式（标准协议）。当前版本不支持。
HI_AIO_MODE_PCM_MASTER_NON_STD	PCM主模式（自定义协议）。当前版本不支持。
HI_AIO_MODE_BUTT	预留值。

1.9.20.2.4 hi_audio_bit_width

说明

定义音频采样位宽。

定义

```
typedef enum {  
    HI_AUDIO_BIT_WIDTH_8 = 0, /* 8bit width */  
    HI_AUDIO_BIT_WIDTH_16 = 1, /* 16bit width */  
    HI_AUDIO_BIT_WIDTH_24 = 2, /* 24bit width */  
    HI_AUDIO_BIT_WIDTH_BUTT  
} hi_audio_bit_width;
```

成员

成员名称	描述
HI_AUDIO_BIT_WIDTH_8	采样位宽为8bit。当前版本不支持。
HI_AUDIO_BIT_WIDTH_16	采样位宽为16bit。
HI_AUDIO_BIT_WIDTH_24	采样位宽为24bit。

1.9.20.2.5 hi_audio_snd_mode

说明

定义音频声道模式。

定义

```
typedef enum {  
    HI_AUDIO_SOUND_MODE_MONO = 0, /* mono */  
    HI_AUDIO_SOUND_MODE_STEREO = 1, /* stereo */  
    HI_AUDIO_SOUND_MODE_BUTT  
} hi_audio_snd_mode;
```

成员

成员名称	描述
HI_AUDIO_SOUND_MODE_MONO	单声道。
HI_AUDIO_SOUND_MODE_STEREO	双声道。

1.9.20.2.6 hi_aio_i2s_type

说明

定义设备 I2S 对接设备类型。

定义

```
typedef enum {  
    HI_AIO_I2STYPE_INNERCODEC = 0,  
    HI_AIO_I2STYPE_INNERHDMI,  
    HI_AIO_I2STYPE_EXTERN  
} hi_aio_i2s_type;
```

成员

成员名称	描述
HI_AIO_I2STYPE_INNERCODEC	对接内置编解码器CODEC。
HI_AIO_I2STYPE_INNERHDMI	对接内置HDMI。当前版本不支持。
HI_AIO_I2STYPE_EXTERN	对接外接设备（例如编解码器）。当前版本不支持。

1.9.20.2.7 hi_aio_attr

说明

定义音频输入输出设备属性结构体。

定义

```
typedef struct {
    hi_audio_sample_rate sample_rate;
    hi_audio_bit_width bit_width;
    hi_aio_mode work_mode;
    hi_audio_snd_mode snd_mode;
    hi_u32 expand_flag;

    hi_u32 frame_num;
    hi_u32 point_num_per_frame;
    hi_u32 chn_cnt;
    hi_u32 clk_share;
    hi_aio_i2s_type i2s_type;
} hi_aio_attr;
```

成员

成员名称	描述
sample_rate	音频采样率（从模式下，此参数不起作用），仅支持48k。
bit_width	音频采样精度（从模式下，此参数必须和音频AD/DA的采样精度匹配），只支持16bit和24bit。
work_mode	音频输入输出工作模式。 仅支持0（master模式），如果配置其它值，系统自动设置为0。
snd_mode	音频声道模式，0表示单声道，1表示立体声。
expand_flag	音频位宽扩展标识。 预留参数，设置无效，配置其它值，系统自动设置为0。
frame_num	缓存帧数目。取值范围：[2, 300]。
point_num_per_frame	每帧的采样点个数。AI取值范围为：[480, 2048]，AO取值范围为：[480, 4096]。
chn_cnt	每路I2S支持的通道数目，最大支持2个通道。双声道时固定为2，单声道时固定为1。
clk_share	配置AI设备0是否复用AO设备0的帧同步时钟及位流时钟。 预留参数，设置无效，配置其它值，系统自动设置为1。
i2s_type	配置设备I2S类型。 仅支持0（对接内置编解码器CODEC），如果配置其它值，系统自动设置为0。

1.9.20.2.8 hi_audio_frame

说明

定义音频帧结构体。

定义

```
typedef struct {
    hi_audio_bit_width bit_width; /* audio frame bit_width */
    hi_audio_snd_mode snd_mode; /* audio frame momo or stereo mode */
    hi_u8 ATTRIBUTE *virt_addr[HI_AUDIO_FRAME_CHN_NUM];
    hi_u64 ATTRIBUTE phys_addr[HI_AUDIO_FRAME_CHN_NUM];
    hi_u64 time_stamp; /* audio frame time stamp */
    hi_u32 seq; /* audio frame seq */
    hi_u32 len; /* data length per channel in frame */
    hi_u32 pool_id[HI_AUDIO_FRAME_CHN_NUM];
} hi_audio_frame;
```

成员

成员名称	描述
bit_width	音频采样精度。
snd_mode	音频声道模式。
virt_addr	音频帧数据虚拟地址。
phys_addr	音频帧数据物理地址。
time_stamp	音频帧时间戳（单位：μs）。
seq	音频帧序号。
len	单个声道音频帧长度（单位：byte）。
pool_id	音频帧缓存池ID。

说明

len（音频帧长度）指单个声道的数据长度。
单声道数据存放的虚拟地址为 virt_addr [0]，长度为 len；立体声数据按左右声道分开存放，虚拟地址 virt_addr [0]存放长度为 len 的左声道数据，虚拟地址virt_addr [1]存放长度为 len 的右声道数据。

1.9.20.2.9 hi_audio_stream

说明

定义音频码流结构体。

定义

```
typedef struct {
    hi_u8 ATTRIBUTE *stream; /* the virtual address of stream */
    hi_u64 ATTRIBUTE phys_addr; /* the physics address of stream */
}
```

```
hi_u32 len;          /* stream length, by bytes */  
hi_u64 time_stamp;   /* frame time stamp */  
hi_u32 seq;          /* frame seq, if stream is not a valid frame,seq is 0 */  
} hi_audio_stream;
```

成员

成员名称	描述
stream	音频码流数据指针。
phys_addr	音频码流的物理地址。
len	音频码流长度（单位：byte）。 - 调用hi_mpi_adec_create_chn接口设置pack方式解码，必须为一帧音频码流的长度（指码流数据+头长度），且一帧码流最大长度为16384。 - 调用hi_mpi_adec_create_chn接口设置stream方式解码： - 调用hi_mpi_adec_send_stream接口发送码流时，设置非阻塞方式时，len参数取值范围：(0, 16384]。 - 调用hi_mpi_adec_send_stream接口发送码流时，设置阻塞方式时，len参数取值范围在hi_u32数据类型的取值范围内，但不包括0。
time_stamp	音频码流时间戳。
seq	音频码流序号，如果stream无效，seq为0。

1.9.20.2.10 hi_aec_frame

说明

定义音频回声抵消参考帧信息结构体。

定义

```
typedef struct {  
    hi_audio_frame ref_frame; /* AEC reference audio frame */  
    hi_bool valid; /* whether frame is valid */  
    hi_bool sys_bind; /* whether is sysbind */  
} hi_aec_frame;
```

成员

成员名称	描述
ref_frame	回声抵消参考帧结构体。

成员名称	描述
valid	参考帧有效的标志。 HI_TRUE：参考帧有效。 HI_FALSE：参考帧无效，无效时不能使用此参考帧进行回声抵消。
sys_bind	AI和AENC是否系统绑定。

 说明

目前不支持回声抵消功能，该结构体设置为空即可。

1.9.20.2.11 hi_audio_frame_info

说明

定义解码后的音频帧信息结构体。

定义

```
typedef struct {  
    hi_audio_frame *frame;    /* frame pointer */  
    hi_u32 id;                /* frame id */  
} hi_audio_frame_info;
```

成员名称	描述
frame	音频帧指针。
id	音频帧的索引。

1.9.20.2.12 hi_acodec_volume_ctrl

说明

定义内置 Audio Codec 音量控制结构体。

定义

```
typedef struct {  
    /* volume control, 0x00~0x7e, 0x7F:mute */  
    hi_u32 volume_ctrl;  
    /* adc/dac mute control, 1:mute, 0:unmute */  
    hi_u32 volume_ctrl_mute;  
} hi_acodec_volume_ctrl;
```

成员

成员名称	描述
volume_ctrl	音量大小。 取值范围：0~0x7F，在0x7F时为静音。
volume_ctrl_mute	静音控制。 取值范围： 0：不静音 1：静音

1.9.20.2.13 音量调节宏

宏定义	取值
HI_ACODEC_SET_DACL_VOLUME	0xc008410d
HI_ACODEC_SET_DACR_VOLUME	0xc008410e
HI_ACODEC_GET_DACL_VOLUME	0xc0084111
HI_ACODEC_GET_DACR_VOLUME	0xc0084112
HI_ACODEC_SET_ADCL_VOLUME	0xc008410f
HI_ACODEC_SET_ADCR_VOLUME	0xc0084110
HI_ACODEC_GET_ADCL_VOLUME	0xc0084113
HI_ACODEC_GET_ADCR_VOLUME	0xc0084114

1.9.20.2.14 hi_aenc_chn_attr

说明

定义音频编码通道属性结构体。

定义

```
typedef struct {  
    hi_payload_type type;  
    hi_u32 point_num_per_frame;  
    hi_u32 buf_size;
```

```
hi_void ATTRIBUTE *value;  
} hi_aenc_chn_attr;
```

成员

成员名称	描述
type	音频编码协议类型，目前支持HI_PT_G711A、HI_PT_G711U、HI_PT_AAC。 静态属性。
point_num_per_frame	音频编码协议对应的帧长（编码时收到的音频帧长小于等于该帧长都可以进行编码）。 G.711a/G.711u编码时支持80/160/240/320/480，AAC编码只支持1024。
buf_size	音频编码缓存大小，以帧为单位。 G.711a/G.711u编码时，该参数取值范围：[2, 100]；AAC编码时，该参数取值范围：[10, 80]。 静态属性。
value	具体协议属性指针。 G.711a/G.711u编码时，该参数值当前无效，非空指针即可；AAC编码时，需传入hi_aenc_attr_aac结构体变量。 静态属性。

1.9.20.2.15 hi_adec_mode

说明

定义解码方式。

定义

```
typedef enum {  
    HI_ADEC_MODE_PACK = 0,  
    HI_ADEC_MODE_STREAM,  
    HI_ADEC_MODE_BUTT  
} hi_adec_mode;
```

成员名称	描述
HI_ADEC_MODE_PACK	pack方式解码。
HI_ADEC_MODE_STREAM	stream方式解码。
HI_ADEC_MODE_BUTT	预留值。

1.9.20.2.16 hi_adec_chn_attr

说明

定义音频解码通道属性结构体。

定义

```
typedef struct {  
    hi_payload_type type;  
    hi_u32 buf_size;  
    hi_adec_mode mode;  
    hi_void ATTRIBUTE *value;  
} hi_adec_chn_attr;
```

成员名称	描述
type	音频解码协议类型，当前仅支持HI_PT_G711A、HI_PT_G711U、HI_PT_AAC。
buf_size	音频解码缓存大小，以帧为单位。 G.711a/G.711u解码时，该参数取值范围：[2, 100]；AAC解码时，该参数取值范围：[10, 80]。
mode	解码方式。 G.711a/G.711u解码时，支持stream和pack解码方式；AAC解码时，仅支持stream模式。
value	具体协议属性指针。 G.711a/G.711u解码时该参数值当前无效，非空指针即可；AAC解码时，需传入hi_adec_attr_aac结构体变量。 静态属性。

1.9.20.2.17 hi_adec_attr_g711

说明

定义G.711解码协议属性结构体。预留。

定义

```
typedef struct {  
    hi_u32 reserved;  
}hi_adec_attr_g711;
```

成员

成员名称	描述
reserved	待扩展用（目前暂未使用）。

1.9.20.2.18 hi_ai_chn_mode

说明

定义AI通道的工作模式。

定义

```
typedef enum {  
    HI_AI_CHN_MODE_NORMAL = 0,  
    HI_AI_CHN_MODE_FAST,  
    HI_AI_CHN_MODE_BUTT  
} hi_ai_chn_mode;
```

成员名称	描述
HI_AI_CHN_MODE_N ORMAL	标准模式。
HI_AI_CHN_MODE_FA ST	快速模式。
HI_AI_CHN_MODE_BU TT	预留值。

1.9.20.2.19 hi_ai_chn_attr

说明

定义AI通道属性结构体。

定义

```
typedef struct {  
    hi_ai_chn_mode mode;  
} hi_ai_chn_attr;
```

成员名称	描述
mode	工作模式。

1.9.20.2.20 hi_aac_type

说明

AAC编解码协议类型。

定义

```
typedef enum {  
    HI_AAC_TYPE_AACLCLC    = 0,  
    HI_AAC_TYPE_EAAC        = 1,  
    HI_AAC_TYPE_EAACPLUS    = 2,  
    HI_AAC_TYPE_AACLCLD     = 3,  
}
```



```
HI_AAC_TYPE_AACELD = 4,  
HI_AAC_TYPE_BUTT,  
} hi_aac_type;
```

成员名称	描述
HI_AAC_TYPE_AACLC	AACLC格式，当前只支持单声道，数据位宽16bit，采样率48k的AAC编解码。
HI_AAC_TYPE_EAAC	eAAC格式（也称为HEAAC、AAC+或 aacPlusV1 ）。当前不支持。
HI_AAC_TYPE_EAACPLUS	eAACPLUS格式（也称为AAC++或 aacPlusV2 ）。当前不支持。
HI_AAC_TYPE_AACLD	ACLD格式。当前不支持。
HI_AAC_TYPE_AACELD	AACELD格式。当前不支持。
HI_AAC_TYPE_BUTT	预留值。

1.9.20.2.21 hi_aac_bps

说明

AAC编解码协议码率。

定义

```
typedef enum {  
    HI_AAC_BPS_8K = 8000,  
    HI_AAC_BPS_16K = 16000,  
    HI_AAC_BPS_22K = 22000,  
    HI_AAC_BPS_24K = 24000,  
    HI_AAC_BPS_32K = 32000,  
    HI_AAC_BPS_48K = 48000,  
    HI_AAC_BPS_64K = 64000,  
    HI_AAC_BPS_96K = 96000,  
    HI_AAC_BPS_128K = 128000,  
    HI_AAC_BPS_256K = 256000,  
    HI_AAC_BPS_320K = 320000,  
    HI_AAC_BPS_BUTT  
} hi_aac_bps;
```

成员名称	描述
HI_AAC_BPS_8K	码率8kbit/s。当前不支持。
HI_AAC_BPS_16K	码率16kbit/s。当前不支持。
HI_AAC_BPS_22K	码率22kbit/s。当前不支持。
HI_AAC_BPS_24K	码率24kbit/s。当前不支持。
HI_AAC_BPS_32K	码率32kbit/s。当前不支持。
HI_AAC_BPS_48K	码率48kbit/s。
HI_AAC_BPS_64K	码率64kbit/s。

成员名称	描述
HI_AAC_BPS_96K	码率96kbit/s。
HI_AAC_BPS_128K	码率128kbit/s。
HI_AAC_BPS_256K	码率256kbit/s。
HI_AAC_BPS_320K	码率320kbit/s。当前不支持。
HI_AAC_BPS_BUTT	预留值。

1.9.20.2.22 hi_aac_transport_type

说明

AAC编解码协议传输封装类型。

定义

```
typedef enum {  
    HI_AAC_TRANSPORT_TYPE_ADTS = 0,  
    HI_AAC_TRANSPORT_TYPE_LOAS = 1,  
    HI_AAC_TRANSPORT_TYPE_LATM_MCP1 = 2,  
    HI_AAC_TRANSPORT_TYPE_BUTT  
} hi_aac_transport_type;
```

成员名称	描述
AAC_TRANS_TYPE_ADTS	ADTS封装格式。当前仅支持AACLC。
AAC_TRANS_TYPE_LOAS	LOAS封装格式。当前不支持。
AAC_TRANS_TYPE_LATM_MCP1	LATM1封装格式。当前不支持。
HI_AAC_TRANSPORT_TYPE_BUTT	预留值。

1.9.20.2.23 hi_aenc_attr_aac

说明

AAC编码协议属性结构体。

定义

```
typedef struct {  
    hi_aac_type    aac_type;  
    hi_aac_bps     bit_rate;  
    hi_audio_sample_rate sample_rate;  
    hi_audio_bit_width  bit_width;  
    hi_audio_snd_mode  snd_mode;
```

```
hi_aac_transport_type transport_type;  
hi_s16 band_width;  
} hi_aenc_attr_aac;
```

成员

成员名称	描述
aac_type	AAC编解码协议类型。当前仅支持HI_AAC_TYPE_AACLC。
bit_rate	AAC编解码协议码率。目前由于采样率只支持48kHz，在单声道的情况下，支持的码率范围为48~256 kbps。
sample_rate	采样率。当前仅支持48kHz。
bit_width	位宽。当前仅支持16 bit。
snd_mode	声道。当前仅支持单声道。
transport_type	AAC编解码协议传输封装类型。当前仅支持HI_AAC_TRANSPORT_TYPE_ADTS类型。
band_width	频段范围，取值为0或[1000, 24000]。

1.9.20.2.24 hi_adec_attr_aac

说明

AAC解码协议属性结构体。

定义

```
typedef struct {  
    hi_aac_transport_type transport_type;  
} hi_adec_attr_aac;
```

成员

成员名称	描述
transport_type	AAC编解码协议传输封装类型。当前仅支持HI_AAC_TRANSPORT_TYPE_ADTS类型。

1.9.20.3 ISP 系统控制及 3A 算法注册

1.9.20.3.1 hi_sensor_id

说明

定义Sensor ID。

定义

```
typedef hi_s32 hi_sensor_id;
```

1.9.20.3.2 hi_isp_sns_attr_info

说明

定义ISP Sensor属性。

定义

```
typedef struct {  
    hi_sensor_id sensor_id;  
} hi_isp_sns_attr_info;
```

成员

成员名称	描述
sensor_id	Sensor ID号。

1.9.20.3.3 hi_isp_cmos_sensor_image_mode

说明

定义Sensor输出的宽高和帧率属性。

定义

```
typedef struct {  
    hi_u16 width;  
    hi_u16 height;  
    hi_float fps;  
    hi_u8 sns_mode;  
} hi_isp_cmos_sensor_image_mode;
```

成员

成员名称	描述
width	Sensor输出的宽度。
height	Sensor输出的高度。
fps	Sensor输出的帧率。
sns_mode	用于进行Sensor初始化序列的选择，在分辨率和帧率相同时，配置不同的sns_mode对应不同的初始化序列；其他情况，sns_mode默认为0。

1.9.20.3.4 hi_isp_cmos_alg_key

说明

定义ISP的各算法是否采用cmos中的默认配置的标志位。

定义

```
typedef union {
    hi_u64 key;
    struct {
        hi_u64 bit1_drc : 1; /* [0] */
        hi_u64 bit1_demosaic : 1; /* [1] */
        hi_u64 bit1_pregamma : 1; /* [2] */
        hi_u64 bit1_gamma : 1; /* [3] */
        hi_u64 bit1_sharpen : 1; /* [4] */
        hi_u64 bit1_edge_mark : 1; /* [5] */
        hi_u64 bit1_hlc : 1; /* [6] */
        hi_u64 bit1_ldci : 1; /* [7] */
        hi_u64 bit1_dpc : 1; /* [8] */
        hi_u64 bit1_lsc : 1; /* [9] */
        hi_u64 bit1_ge : 1; /* [10] */
        hi_u64 bit1_anti_false_color : 1; /* [11] */
        hi_u64 bit1_bayer_nr : 1; /* [12] */
        hi_u64 bit1_detail : 1; /* [13] */
        hi_u64 bit1_ca : 1; /* [14] */
        hi_u64 bit1_expander : 1; /* [15] */
        hi_u64 bit1_clut : 1; /* [16] */
        hi_u64 bit1_wdr : 1; /* [17] */
        hi_u64 bit1_dehaze : 1; /* [18] */
        hi_u64 bit1_lcac : 1; /* [19] */
        hi_u64 bit1_acs : 1; /* [20] */
        hi_u64 bit44_rsv : 43; /* [21:63] */
    };
} hi_isp_cmos_alg_key;
```

成员

成员名称	描述
bit1_drc	drc模块是否采用cmos默认配置的标志位。
bit1_demosaic	demosaic模块是否采用cmos默认配置的标志位。
bit1_pregamma	pregamma模块是否采用cmos默认配置的标志位。
bit1_gamma	gamma模块是否采用cmos默认配置的标志位。
bit1_sharpen	sharpen模块是否采用cmos默认配置的标志位。
bit1_edge_mark	edge mark模块是否采用cmos默认配置的标志位
bit1_hlc	hlc模块是否采用cmos默认配置的标志位。
bit1_ldci	ldci模块是否采用cmos默认配置的标志位。
bit1_dpc	dpc模块是否采用cmos默认配置的标志位。
bit1_lsc	lsc模块是否采用cmos默认配置的标志位。
bit1_ge	ge模块是否采用cmos默认配置的标志位。
bit1_anti_false_color	anti false color模块是否采用cmos默认配置的标志位。

成员名称	描述
bit1_bayer_nr	bayer nr模块是否采用cmos默认配置的标志位。
bit1_detail	detail enhance模块是否采用cmos默认配置的标志位
bit1_ca	ca模块是否采用cmos默认配置的标志位。
bit1_expander	expander模块是否采用cmos默认配置的标志位。 仅 sensor built-in模式有效。
bit1_clut	clut模块是否采用cmos默认配置的标志位。
bit1_wdr	wdr模块是否采用cmos默认配置的标志位。
bit1_dehaze	dehaze模块是否采用cmos默认配置的标志位。
bit1_lcac	Local cac模块是否采用cmos默认配置的标志位。
bit1_acs	acs模块是否采用cmos默认配置的标志位。

注意事项

如果ISP的某个算法模块要使用cmos中的配置，要将对应的标志位置为1，否则采用的是算法内部的默认配置。

1.9.20.3.5 hi_isp_cmos_noise_calibration

说明

定义噪声校正参数。

定义

```
typedef struct {  
    hi_u16 calibration_lut_num;  
    hi_float calibration_coef[HI_ISP_BAYER_CALIBTAION_MAX_NUM]  
[HI_ISP_BAYER_CALIBRATION_PARA_NUM];  
} hi_isp_cmos_noise_calibration;
```

成员

成员名称	描述
calibration_lut_num	标定序列数目。 取值范围：[0, 49]。
calibration_coef	噪声标定表。 HI_ISP_BAYER_CALIBTAION_MAX_NUM用于定义标定噪声模型参数的iso档位个数的最大值，当前为50。 HI_ISP_BAYER_CALIBRATION_PARA_NUM用于定义标定噪声模型参数的最大个数，当前为16。

1.9.20.3.6 hi_isp_cmos_sensor_max_resolution

说明

定义Sensor最大分辨率结构体。

定义

```
typedef struct {  
    hi_u32 max_width;  
    hi_u32 max_height;  
} hi_isp_cmos_sensor_max_resolution;
```

成员

成员名称	描述
max_width	最大宽度。
max_height	最大高度。

1.9.20.3.7 hi_isp_cmos_sensor_mode

说明

定义Sensor模式寄存器。

定义

```
typedef struct {  
    hi_u32 sensor_id;  
    hi_u8 sensor_mode;  
} hi_isp_cmos_sensor_mode;
```

成员

成员名称	描述
sensor_id	Sensor ID。
sensor_mode	Sensor自定义工作模式，不同分辨率与帧率对应不同的工作模式。

1.9.20.3.8 hi_isp_cmos_wdr_switch_attr

说明

定义WDR切换属性。

定义

```
typedef struct {  
    hi_u32 exp_ratio[HI_ISP_EXP_RATIO_NUM];  
} hi_isp_cmos_wdr_switch_attr;
```

成员

成员名称	描述
exp_ratio	曝光比期望值

注意事项

在 cmos_get_isp_default 函数里根据 WDR 模式给 exp_ratio 赋默认值，所赋值要与 cmos_get_ae_default 中 AE 初始化的曝光比保持一致性：

- 若 ae_sns_dft->man_ratio_enable 为 TRUE，exp_ratio 与 ae_sns_dft->arr_ratio 的值相同；
- 若 ae_sns_dft->man_ratio_enable 为 FALSE，exp_ratio 为 0x40。

1.9.20.3.9 hi_isp_cmos_default

说明

定义ISP基础算法库的初始化参数结构体。

定义

```
typedef struct {  
    hi_isp_cmos_alg_key key;  
    const hi_isp_cmos_drc *drc;  
    const hi_isp_cmos_demosaic *demosaic;  
    const hi_isp_cmos_pregamma *pregamma;  
    const hi_isp_cmos_gamma *gamma;  
    const hi_isp_cmos_sharpen *sharpen;  
    const hi_isp_cmos_edgemark *edge_mark;  
    const hi_isp_cmos_hlc *hlc;  
    const hi_isp_cmos_ldci *ldci;  
    const hi_isp_cmos_dpc *dpc;  
    const hi_isp_cmos_lsc *lsc;  
    const hi_isp_cmos_ge *ge;  
    const hi_isp_cmos_afc *anti_false_color;  
    const hi_isp_cmos_bayernr *bayer_nr;  
    const hi_isp_cmos_ca *ca;  
    const hi_isp_expander_attr *expander;  
    const hi_isp_cmos_clut *clut;  
    const hi_isp_cmos_wdr *wdr;  
    const hi_isp_cmos_dehaze *dehaze;  
    const hi_isp_cmos_lcac *lcac;  
    const hi_isp_cmos_acs *acs;  
    hi_isp_cmos_noise_calibration noise_calibration;  
    hi_isp_cmos_sensor_max_resolution sensor_max_resolution;  
    hi_isp_cmos_sensor_mode sensor_mode;  
    hi_isp_cmos_wdr_switch_attr wdr_switch_attr;  
} hi_isp_cmos_default;
```


成员

成员名称	描述
key	标识各算法是否采用 cmos 中默认配置的 key。
*drc	DRC结构体指针。 请参见 hi_isp_cmos_drc 。
*dmosaic	Demosaic结构体指针。 请参见 hi_isp_cmos_dmosaic 。
*pregamma	PreGamma结构体指针。 请参见 hi_isp_cmos_pregamma 。
*gamma	Gamma结构体指针。 请参见 hi_isp_cmos_gamma 。
*sharpen	Sharpen结构体指针。 请参见 hi_isp_cmos_sharpen 。
*edge_mark	EdgeMark结构体指针。 请参见 hi_isp_cmos_edgemark 。
*hlc	HLC结构体指针。 请参见 hi_isp_cmos_hlc 。
*ldci	LDCI结构体指针。 请参见 hi_isp_cmos_ldci 。
*dpc	DPC结构体指针。 请参见 hi_isp_cmos_dpc 。
*lsc	LSC结构体指针。 请参见 hi_isp_cmos_lsc 。
*ge	GE模块结构体指针。 请参见 hi_isp_cmos_ge 。
*anti_false_color	AntiFalse结构体指针。 请参见 hi_isp_cmos_afc 。
*bayer_nr	BayerNR结构体指针。 请参见 hi_isp_cmos_bayernr 。
*ca	CA模块结构体指针。 请参见 hi_isp_cmos_ca 。
*expander	Expander结构体指针。仅Sensor built-in模式有效。 请参见 hi_isp_expander_attr 。
*clut	Clut结构体指针。 请参见 hi_isp_cmos_clut 。

成员名称	描述
*wdr	WDR模式结构体指针。 请参见 hi_isp_cmos_wdr 。
*dehaze	Dehaze结构体指针。 请参见 hi_isp_cmos_dehaze 。
*lcac	Local cac结构体指针。 请参见 hi_isp_cmos_lcac 。
*acs	ACS结构体指针。 请参见 hi_isp_cmos_acs 。
noise_calibration	Noise校正结构体。
sensor_max_resolution	Sensor最大宽高结构体。
sensor_mode	Sensor模式结构体。
wdr_switch_attr	WDR切换属性。

注意事项

- ISP的各算法模块如果要采用cmos的默认配置，要在回调函数[pfn_cmos_get_isp_default](#)中将对应的标志位置1，并给该算法模块的cmos结构体指针赋值。如果cmos默认值配置的不合法，会导致算法初始化失败，同时run起来之后算法不能正常调节。

1.9.20.3.10 hi_isp_cmos_black_level

说明

定义sensor的黑电平结构体。

定义

```
typedef struct {  
    hi_bool update;  
    hi_u16 black_level[HI_ISP_BAYER_CHN_NUM];  
} hi_isp_cmos_black_level;
```

成员

成员名称	描述
update	Sensor的黑电平是否会动态根据增益改变。 取值范围： - HI_TRUE：是 - HI_FALSE：否 如果Sensor的黑电平不会动态根据增益改变，update配置为HI_FALSE 即可。
black_level	Sensor的黑电平数组，取值范围：[0, 4095]。 HI_ISP_BAYER_CHN_NUM用于定义Bayer数据的通道数目，当前为4。

1.9.20.3.11 hi_isp_sns_type

说明

定义Sensor与ISP的通信接口类型。

定义

```
typedef enum {  
    HI_ISP_SNS_I2C_TYPE = 0,  
    HI_ISP_SNS_SSP_TYPE,  
    HI_ISP_SNS_TYPE_BUTT,  
} hi_isp_sns_type;
```

成员

成员名称	描述
HI_ISP_SNS_I2C_TYPE	Sensor与ISP使用I2C接口通信。
HI_ISP_SNS_SSP_TYPE	Sensor与ISP使用SSP接口通信。
HI_ISP_SNS_TYPE_BUTT	预留。

1.9.20.3.12 hi_isp_sns_commbus

说明

定义与Sensor绑定的设备号信息。

定义

```
typedef union {  
    hi_s8 i2c_dev;  
    struct {
```

```
    hi_s8 bit4_ssp_dev    : 4;  
    hi_s8 bit4_ssp_cs    : 4;  
} ssp_dev;  
} hi_isp_sns_commbus;
```

成员

成员名称	描述
i2c_dev	Sensor绑定的I2C设备号。
bit4_ssp_dev	Sensor绑定的SPI设备号。
bit4_ssp_cs	Sensor绑定的SPI片选信号。

1.9.20.3.13 hi_isp_i2c_data

说明

定义I2C数据的参数。

定义

```
typedef struct {  
    hi_bool update; /* RW; Range: [0x0, 0x1]; Format:1.0;  
                    HI_TRUE: The sensor registers are written,  
                    HI_FALSE: The sensor registers are not written */  
    hi_u8 delay_frm_num; /* RW; Number of delayed frames for the sensor register */  
    hi_u8 int_pos; /* RW; Position where the configuration of the sensor register takes effect */  
    hi_u8 dev_addr; /* RW; Sensor device address */  
    hi_u32 reg_addr; /* RW; Sensor register address */  
    hi_u32 addr_byte_num; /* RW; Bit width of the sensor register address */  
    hi_u32 data; /* RW; Sensor register data */  
    hi_u32 data_byte_num; /* RW; Bit width of sensor register data */  
} hi_isp_i2c_data;
```

成员

成员名称	描述
update	数据是否会配置Sensor寄存器。 HI_TRUE：数据会配置Sensor寄存器； HI_FALSE：数据不会配置Sensor寄存器。
delay_frame_num	Sensor寄存器延迟配置的帧数。此变量的目的是保证曝光时间和增益同时生效。

成员名称	描述
interrupt_pos	Sensor寄存器的配置生效的位置。 • 设置为0x0时表示寄存器在超短帧起始中断配置生效，设置为0x 1时表示寄存器在超短帧结束中断配置生效。 • 设置为0x10时表示寄存器在短帧起始中断配置生效，设置为0x11时表示寄存器在短帧结束中断配置生效。 • 设置为0x20时表示c寄存器在中帧起始中断配置生效，设置为0x21时表示寄存器在中帧结束中断配置生效。 • 设置为0x30时表示寄存器在长帧起始中断配置生效，设置为0x31时表示寄存器在长帧结束中断配置生效。
dev_addr	Sensor设备地址。
reg_addr	Sensor寄存器地址。
addr_byte_num	Sensor寄存器地址位宽。
data	Sensor寄存器数据。
data_byte_num	Sensor寄存器数据位宽。

1.9.20.3.14 hi_isp_ssp_data

说明

定义SSP数据的参数。

定义

```
typedef struct {
    hi_bool update; /* RW; Range: [0x0, 0x1]; Format:1.0;
                    HI_TRUE: The sensor registers are written,
                    HI_FALSE: The sensor registers are not written */
    hi_u8 delay_frm_num; /* RW; Number of delayed frames for the sensor register */
    hi_u8 int_pos; /* RW;Position where the configuration of the sensor register takes effect */
    hi_u32 dev_addr; /* RW;Sensor device address */
    hi_u32 dev_addr_byte_num; /* RW;Bit width of the sensor device address */
    hi_u32 reg_addr; /* RW;Sensor register address */
    hi_u32 reg_addr_byte_num; /* RW;Bit width of the sensor register address */
    hi_u32 data; /* RW;Sensor register data */
    hi_u32 data_byte_num; /* RW;Bit width of sensor register data */
} hi_isp_ssp_data;
```

成员

成员名称	描述
update	数据是否会配置Sensor寄存器。 HI_TRUE：数据会配置Sensor寄存器； HI_FALSE：数据不会配置Sensor寄存器。

成员名称	描述
delay_frame_num	Sensor寄存器延迟配置的帧数。此变量的目的是保证曝光时间和增益同时生效。
interrupt_pos	Sensor寄存器的配置生效的位置。 • 设置为0x0时表示寄存器在帧起始中断配置生效，设置为1时表示寄存器在AF中断配置生效。 • 设置为0x10时表示寄存器在短帧起始中断配置生效，设置为0x11时表示寄存器在短帧结束中断配置生效。 • 设置为0x20时表示寄存器在中帧起始中断配置生效，设置为0x21时表示寄存器在中帧结束中断配置生效。 • 设置为0x30时表示寄存器在长帧起始中断配置生效，设置为0x31时表示寄存器在长帧结束中断配置生效。
dev_addr	Sensor设备地址。
dev_addr_byte_num	Sensor设备地址位宽。
reg_addr	Sensor寄存器地址。
reg_addr_byte_num	Sensor寄存器地址位宽。
data	Sensor寄存器数据。
data_byte_num	Sensor寄存器数据位宽。

1.9.20.3.15 hi_isp_sns_regs_info

说明

定义Sensor的寄存器信息。

定义

```
typedef struct {
    hi_isp_sns_type sns_type;
    hi_u32 reg_num;
    hi_u8 cfg2_valid_delay_max;
    hi_u32 exp_distance[HI_ISP_WDR_MAX_FRAME_NUM - 1]
    hi_isp_sns_commbus com_bus;
    union {
        hi_isp_i2c_data i2c_data[HI_ISP_MAX_SNS_REGS];
        hi_isp_ssp_data ssp_data[HI_ISP_MAX_SNS_REGS];
    };
    struct {
        hi_bool update;
        hi_u8 delay_frame_num;
        hi_u32 slave_vs_time;
        hi_u32 slave_bind_dev;
    } slv_sync;
    hi_bool config;
} hi_isp_sns_regs_info;
```

成员

成员名称	描述	
sns_type	Sensor与ISP的通信接口类型。	
reg_num	曝光结果写到Sensor时需要配置的寄存器个数，不支持动态修改。	
cfg2_valid_delay_max	所有Sensor寄存器从配置到生效延迟的帧数的最大值，单位为帧，用于保证Sensor寄存器和ISP寄存器的同步。一般情况下，cmos Sensor的曝光时间寄存器的延迟最大，为1~2帧，因此配置一般为1或2。	
exp_distance	Sensor在wdr模式下曝光长帧与中帧的行差，中帧与短帧的行差，短帧与短短帧的行差。预留属性，暂不支持。	
com_bus	与Sensor绑定的设备号信息	
i2c_data	I2C数据参数。	
ssp_data	SSP数据参数。	
update	数据是否会配置Sensor寄存器。 HI_TRUE：数据会配置Sensor寄存器； HI_FALSE：数据不会配置Sensor寄存器。	
delay_frame_num	Sensor寄存器延迟配置的帧数。此变量的目的是保证曝光时间和增益同时生效。	
slave_vs_time	XVS信号周期，单位：Sensor输入时钟周期。	
slave_bind_dev	Slave设备号与vi_pipe绑定关系。	
config	Sensor寄存器数据配置完成标志。 - TD_TRUE：完成配置。 - TD_FALSE：还未配置。	

注意事项

无

1.9.20.3.16 hi_isp_sensor_exp_func

说明

定义Sensor回调函数结构体。

定义

```
typedef struct {  
    hi_void (*pfn_cmos_sensor_init)(hi_vi_pipe vi_pipe);  
    hi_void (*pfn_cmos_sensor_exit)(hi_vi_pipe vi_pipe);  
    hi_void (*pfn_cmos_sensor_global_init)(hi_vi_pipe vi_pipe);  
    hi_s32 (*pfn_cmos_set_image_mode)(hi_vi_pipe vi_pipe, hi_isp_cmos_sensor_image_mode  
*sensor_image_mode);  
    hi_s32 (*pfn_cmos_set_wdr_mode)(hi_vi_pipe vi_pipe, hi_u8 mode);  
    hi_s32 (*pfn_cmos_get_isp_default)(hi_vi_pipe vi_pipe, hi_isp_cmos_default *def);  
    hi_s32 (*pfn_cmos_get_isp_black_level)(hi_vi_pipe vi_pipe, hi_isp_cmos_black_level *black_level);  
    hi_s32 (*pfn_cmos_get_sns_reg_info)(hi_vi_pipe vi_pipe, hi_isp_sns_regs_info *sns_regs_info);  
    hi_void (*pfn_cmos_set_pixel_detect)(hi_vi_pipe vi_pipe, hi_bool enable);  
    hi_void (*pfn_cmos_notify_event)(hi_vi_pipe vi_pipe, hi_u32 event_id, hi_void *value);  
} hi_isp_sensor_exp_func;
```

成员

成员名称	描述
pfn_cmos_sensor_init	初始化Sensor的回调函数指针。
pfn_cmos_sensor_exit	Sensor的回调退出函数指针。
pfn_cmos_sensor_global_init	初始化全局变量的回调函数指针。
pfn_cmos_set_image_mode	设置分辨率和帧率切换的回调函数指针。 返回值0表示Sensor模式发生改变，ISP会调用pfn_cmos_sensor_init 重新配置Sensor； 返回值-2表示Sensor模式没有变化，ISP不会重新配置Sensor。
pfn_cmos_set_wdr_mode	设置wdr模式的回调函数指针。
pfn_cmos_get_isp_default	获取ISP基础算法的初始值的回调函数指针。
pfn_cmos_get_isp_black_level	获取Sensor的黑电平值的回调函数指针，支持根据Sensor增益动态调整黑电平值。若此处动态调整黑电平值，则外部只能通过接口hi_mpi_isp_set_black_level_attr的手动模式设置黑电平。
pfn_cmos_get_sns_reg_info	获取Sensor寄存器信息的回调函数指针，用于实现内核态配置AE信息。
pfn_cmos_set_pixel_detect	设置坏点校正开关的回调函数指针。
pfn_cmos_notify_event	通知模组驱动特殊事件，如休眠后唤醒事件，便于模组做部分逻辑处理。 - event_id：必须为0，只支持1个事件，即休眠唤醒事件。 - value：必须为null，预留字段。

注意事项

- pfn_cmos_sensor_init, pfn_cmos_get_isp_default, pfn_cmos_get_isp_black_level, pfn_cmos_set_pixel_detect和 pfn_cmos_get_sns_reg_info 必须赋值，其他回调函数指针如果不需要赋值，应置为NULL。例如有的Sensor不支持切换分辨率，那么pfn_cmos_set_image_mode需要置为NULL。
- 不支持切换AWB增益配置位置。

1.9.20.3.17 hi_isp_sensor_register

说明

定义Sensor注册结构体。

定义

```
typedef struct {  
    hi_isp_sensor_exp_func sns_exp;  
} hi_isp_sensor_register;
```

成员

成员名称	描述
sns_exp	Sensor注册的回调函数结构体。

1.9.20.3.18 hi_isp_bayer_format

说明

定义输入Bayer图像数据格式。

定义

```
typedef enum {  
    HI_ISP_BAYER_RGGB = 0,  
    HI_ISP_BAYER_GRBG = 1,  
    HI_ISP_BAYER_GBRG = 2,  
    HI_ISP_BAYER_BGGR = 3,  
    HI_ISP_BAYER_BUTT  
} hi_isp_bayer_format;
```

成员

成员名称	描述
HI_ISP_BAYER_RGGB	RGGB排列方式。
HI_ISP_BAYER_GRBG	GRGB排列方式。
HI_ISP_BAYER_GBRG	GBRG排列方式。
HI_ISP_BAYER_BGGR	BGGR排列方式。

1.9.20.3.19 hi_isp_pub_attr

说明

定义 ISP 公共属性。

定义

```
typedef struct {  
    hi_rect      wnd_rect;  
    hi_size      sns_size;  
    hi_float     frame_rate;  
    hi_isp_bayer_format bayer_format;  
    hi_wdr_mode  wdr_mode;  
    hi_u8        sns_mode;  
} hi_isp_pub_attr;
```

成员

成员名称	描述
wnd_rect	裁剪窗口起始位置和图像宽高，必须设置为与vi pipe属性中的size保持一致，且不能比所绑定的dev属性中的宽高 大，如果配置不符，会导致出图异常。 - wnd_rect中设置的水平方向起始位置与图像宽度之和应小于Sensor输出的图像宽度。 - wnd_rect中设置的垂直方向起始位置与图像高度之和应小于Sensor输出的图像高度，由于无法检测Sensor实际输出的宽高，当不满足该条件时会返回不报错。
sns_size	Sensor输出的图像宽高。 宽或高的取值范围：[120, 16384]。 设置sns_size时，需与对应的Sensor采集属性值保持一致。
frame_rate	输入图像帧率，取值范围为(0.00, 65535.00]。
bayer_format	Bayer数据格式。
wdr_mode	WDR模式选择。
sns_mode	用于进行Sensor初始化序列的选择，在分辨率和帧率相同 同时，配置不同的sns_mode对应不同的初始化序列；其他 情况，sns_mode默认配置为0，可通过sns_size和 frame_rate进行初始化序列的选择。

1.9.20.3.20 hi_isp_frame_info

说明

定义 ISP 实时信息。

定义

```
typedef struct {  
    hi_u32    iso;  
    hi_u32    exposure_time;  
    hi_u32    isp_dgain;  
    hi_u32    again;  
    hi_u32    dgain;  
    hi_u32    ratio[3];  
    hi_u32    isp_nr_strength;  
    hi_u32    f_number;  
    hi_u32    sensor_id;  
    hi_u32    sensor_mode;  
    hi_u32    hmax_times;  
    hi_u32    vmax;  
    hi_u32    vc_num;  
    hi_u64    fsync_timestamp;  
    hi_u64    fs_timestamp;  
    hi_u64    fe_timestamp;  
    hi_u64    pe_timestamp;  
    hi_s64    motion_filter[9];  
    hi_u32    reserved[10];  
} hi_isp_frame_info;
```

成员

成员名称	描述
iso	当前Sensor模拟增益*Sensor数字增益*ISP数字增益*100。仅作PQ调测参考使用。
exposure_time	曝光时间，单位是微秒（us）。仅作PQ调测参考使用。
isp_dgain	ISP数字增益。仅作PQ调测参考使用。
again	Sensor的模拟增益。仅作PQ调测参考使用。
dgain	Sensor的数字增益。仅作PQ调测参考使用。
ratio	多帧合成WDR相邻2帧默认曝光比。仅作PQ调测参考使用。
isp_nr_strength	ISP的NR强度。当前未支持，默认值为0。
f_number	当前使用的镜头的F值。仅作PQ调测参考使用。
sensor_id	当前使用的sensorID。仅作PQ调测参考使用。
sensor_mode	当前使用的Sensor序列模式。仅作PQ调测参考使用。
hmax_times	当前使用的Sensor对应读出一行的时间，单位是纳秒（ns）。仅作PQ调测参考使用。
vmax	Sensor每帧实际生效的总行数，单位是行。仅作PQ调测参考使用。
vc_num	当前未支持，默认值为0。当前未支持，默认值为0。
fsync_timestamp	发出fsync曝光同步信号的时间。
fs_timestamp	收到Sensor模组FrameStart信号的时间，仅做参考，会受CPU负载以及调度而存在误差。

成员名称	描述
fe_timestamp	收到Sensor模组FrameEnd信号的时间，仅做参考，会受CPU负载以及调度而存在误差。
pe_timestamp	ISP处理完成后输出图像的时间，仅做维测参考使用。
motion_filter	数字防抖的3*3仿射变换矩阵，15bit 小数精度。该值无需用户设置，是开启数字防抖后自动生成的值。 用户不能直接使用该矩阵作为仿射变化矩阵，完成以下步骤后可得到仿射变换矩阵： （1）转换为浮点类型做归一化处理 $m = float([x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9])$ $m_1 = \begin{bmatrix} \frac{x_1}{x_9} & \frac{x_2}{x_9} & \frac{x_3}{x_9} \\ \frac{x_4}{x_9} & \frac{x_5}{x_9} & \frac{x_6}{x_9} \\ \frac{x_7}{x_9} & \frac{x_8}{x_9} & \frac{x_9}{x_9} \end{bmatrix}$ （2）求m1的逆矩阵，最终得到m2矩阵，m2矩阵可与图像直接进行运算 $m_2 = inv(m_1)$ （3）m2矩阵与输入图像进行相乘可得到校正后的图像 $img_{dst} = m_2 \otimes img_{src}$
reserved	预留字段，默认输出为0。

1.9.20.3.21 hi_slave_dev

说明

定义从信号设备号。

定义

```
typedef hi_s32 hi_slave_dev;
```

1.9.20.3.22 hi_isp_slave_sns_sync

说明

定义从模式Sensor同步信号配置。

定义

```
typedef struct {
    union {
        struct {
            hi_u32 bit16_reserved : 16;
            hi_u32 bit_h_inv : 1;
            hi_u32 bit_v_inv : 1;
            hi_u32 bit12_reserved : 12;
            hi_u32 bit_h_enable : 1;
            hi_u32 bit_v_enable : 1;
        } bits;
        hi_u32 bytes;
    } cfg;
    hi_u32 vs_time;
    hi_u32 hs_time;
    hi_u32 vs_cyc;
    hi_u32 hs_cyc;
    hi_u32 hs_dly_cyc;
    hi_u32 slave_mode_time;
} hi_isp_slave_sns_sync;
```

成员

成员名称	描述
bit16_reserved	保留字段。
bit_h_inv	XHS极性配置。 0：正极； 1：负极。
bit_v_inv	XVS极性配置。 0：正极； 1：负极。
bit12_reserved	保留字段。
bit_h_enable	XHS输出使能。
bit_v_enable	XVS输出使能。
vs_time	XVS信号周期，单位：Sensor输入时钟周期。
hs_time	XHS信号周期，单位：Sensor输入时钟周期。
vs_cyc	XVS有效电平宽度，单位：Sensor输入时钟周期。
hs_cyc	XHS有效电平宽度，单位：Sensor输入时钟周期。
hs_dly_cyc	XHS脉冲输出相对XVS脉冲的延迟周期配置，单位：Sensor输入时钟周期。

成员名称	描述
slave_mode_time	Sensor从模式时序配置选择寄存器： ● 0：选择SENSOR0 timing配置； ● 1：选择SENSOR1 timing配置； ● 2：选择SENSOR2 timing配置； ● 3：选择SENSOR3 timing配置。

注意事项

如图 2-4~图 2-6 所示，说明了同步信号发生模块配置参数的含义。

图2-4 同步信号配置时序图

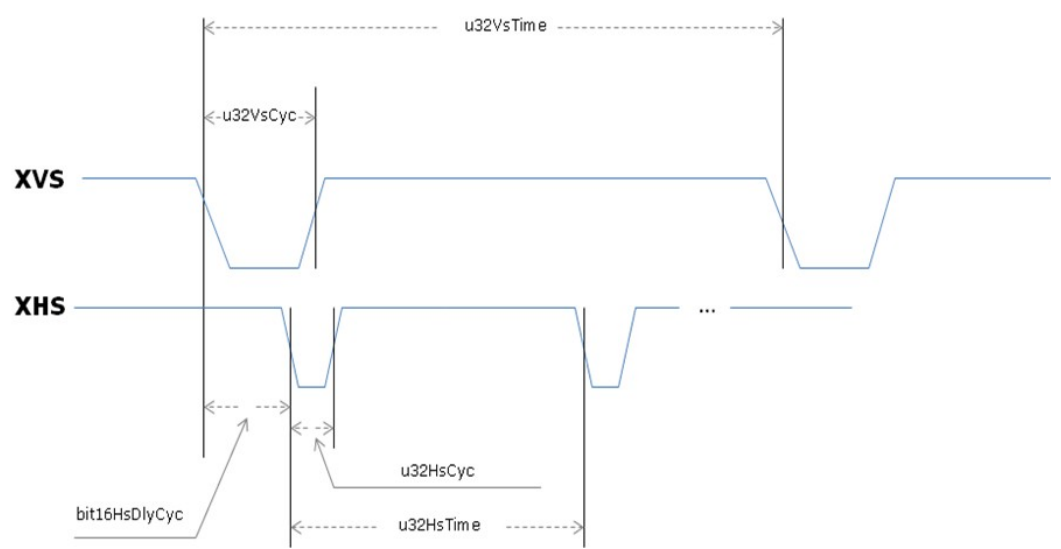


图2-5 同步信号极性翻转

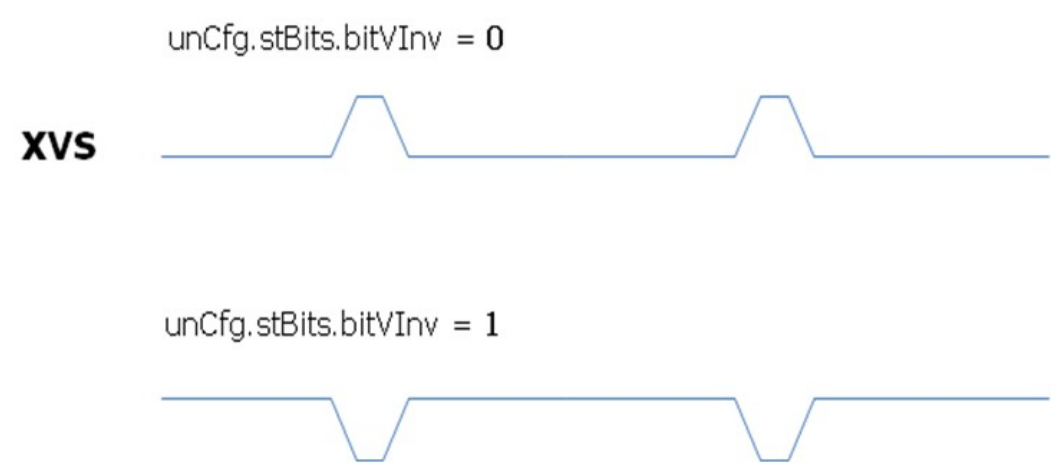
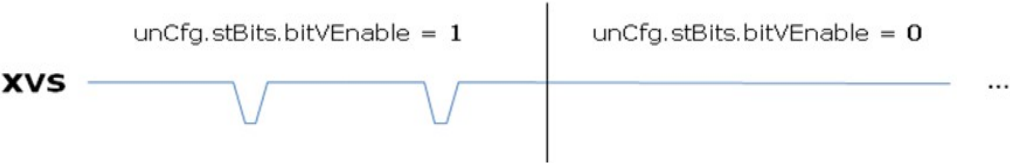


图2-6 同步信号使能



1.9.20.3.23 hi_isp_3a_alg_lib

说明

定义AE/AWB算法库结构体。

定义

```
typedef struct {  
    hi_s32 id;  
    hi_char lib_name[HI_ISP_ALG_LIB_NAME_SIZE_MAX];  
} hi_isp_3a_alg_lib;
```

成员

成员名称	描述
id	算法库实例的Id，用以支持运行同一个算法库的多个实例
lib_name	标识算法库名称的字符数组，用以区分不同的算法库。 HI_ISP_ALG_LIB_NAME_SIZE_MAX用于定义3A算法库名称的最大字符数，表示20。

1.9.20.3.24 hi_isp_ae_accuracy_type

说明

定义曝光时间、增益的精度类型的枚举。

定义

```
typedef enum {  
    HI_ISP_AE_ACCURACY_DB = 0,  
    HI_ISP_AE_ACCURACY_LINEAR,  
    HI_ISP_AE_ACCURACY_TABLE,  
    HI_ISP_AE_ACCURACY_BUTT,  
} hi_isp_ae_accuracy_type;
```

成员

成员名称	描述
HI_ISP_AE_ACCURACY_DB	DB精度类型。 DB精度指的是增益的倍数以倍增的形式增加。例如，Sensor支持的again 为0db，0.3db，0.6db……这种情况下，精度类型可以设定为HI_ISP_AE_ACCURACY_DB，精度值设置为0.3，在这种精度下，again为 80，则表示 $80 \times 0.3\text{db} = 24\text{db}$ ，24db是16倍增益；再例如，Sensor支持的again为1倍，2倍，4倍，8倍……这种情况下，对应为db则为0db，6db，12db，18db，所以精度类型设定为HI_ISP_AE_ACCURACY_DB，精度值设置为6。由于AE算法内部实现线性转DB时计算精度存在误差，当DB精度较高(小于1)时，建议采用表格精度来实现。
HI_ISP_AE_ACCURACY_LINEAR	线性精度类型。 线性精度指的是增益的倍数均匀增加。例如，Sensor能支持的again为 1 倍， $1(1+1/16)$ 倍， $1(1+2/16)$ 倍……这种情况下，精度类型可以设定为HI_ISP_AE_ACCURACY_LINEAR，精度值设置为0.0625，在这种精度下，again为32则表示 $32 \times 0.0625 = 2$ 倍增益。
HI_ISP_AE_ACCURACY_TABLE	表格类型。 表格精度指增益的倍数是通过查表的形式获得，表格统一使用10bit精度，即1024表示1倍增益。当某些Sensor的增益值的增加规律非线性或者DB精度较高时，可以使用表格精度方式，AE算法计算出需要的模拟增益/数字增益的数值，用查询Sensor增益表格中最接近的值作为数字增益/模拟增益的值。 使用该模式时，需要相应的初始化回调结构体hi_isp_ae_sensor_exp_func 中的回调函数。

1.9.20.3.25 hi_isp_ae_accuracy

说明

定义曝光时间、增益的精度结构体。

定义

```
typedef struct {  
    hi_isp_ae_accuracy_type accu_type;  
    float accuracy;  
    float offset;  
} hi_isp_ae_accuracy;
```


成员

成员名称	描述
accu_type	精度类型。
accuracy	精度值。
offset	曝光时间的偏移量，支持正负偏移量设置，以行为单位，该值配置与Sensor强相关。

注意事项

- 曝光时间的精度通常都是线性的，都是以行为单位。若曝光时间为线性精度，精度accuracy为大于1的整数，则意味着AE计算出来的曝光行数只能为最小曝光时间加上accuracy的整数倍。该功能可用于计算某些Sensor WDR模式下的长/短帧曝光时间，如某些Sensor曝光行数必须为 $2n+1$ ，则可以配置最小曝光时间为1或3，配置accuracy为2。
- 曝光时间与Sensor增益保持线性关系是AE曝光量分配的前提，即认为曝光量一定的情况下，曝光时间和Sensor增益可以相互转换而保持图像亮度基本不变。比如说曝光量为4096，那么分配为曝光时间(2)*增益(2048)，也可以分配为曝光时间(4)*增益(1024)，这两种情况下图像亮度应该基本不变。若不满足这种线性关系，在高亮环境下，由于曝光时间很短，1行曝光时间的变化也容易使画面发生闪烁。某些Sensor，如Panasoni MN34220采用1080p@30fps的初始化序列配置时，曝光时间必须偏移0.8018 行之后才与Sensor增益有线性关系，因此增加了offset这个变量，它为float型变量，以行为单位，将offset设置成0.8018，AE算法内部即可完成偏移处理。一般Sensor的曝光时间和增益不存在这种偏移关系，需在cmos.c中将该值配为0。

1.9.20.3.26 hi_isp_ae_sensor_default

说明

定义AE算法库的初始化参数结构体。

定义

```
typedef struct {  
    hi_u8 hist_thresh[HI_ISP_HIST_THRESH_NUM];  
    hi_u8 ae_compensation;  
    hi_u32 lines_per500ms;  
    hi_u32 flicker_freq;  
    hi_float fps;  
    hi_u32 hmax_times;  
    hi_u32 init_exposure;  
    hi_u32 init_ae_speed;  
    hi_u32 init_ae_tolerance;  
    hi_u32 full_lines_std;  
    hi_u32 full_lines_max;  
    hi_u32 full_lines;  
    hi_u32 binning_full_lines;  
    hi_u32 max_int_time;  
    hi_u32 min_int_time;  
    hi_u32 max_int_time_target;  
    hi_u32 min_int_time_target;  
    hi_isp_ae_accuracy int_time_accu;
```

```
hi_u32 max_again;
hi_u32 min_again;
hi_u32 max_again_target;
hi_u32 min_again_target;
hi_isp_ae_accuracy again_accu;
hi_u32 max_dgain;
hi_u32 min_dgain;
hi_u32 max_dgain_target;
hi_u32 min_dgain_target;
hi_isp_ae_accuracy dgain_accu;
hi_u32 max_isp_dgain_target;
hi_u32 min_isp_dgain_target;
hi_u32 isp_dgain_shift;
hi_u32 max_int_time_step;
hi_bool max_time_step_enable;
hi_u32 max_inc_time_step[HI_ISP_WDR_MAX_FRAME_NUM];
hi_u32 max_dec_time_step[HI_ISP_WDR_MAX_FRAME_NUM];
hi_u32 lf_max_short_time;
hi_u32 lf_min_exposure;
hi_isp_ae_route ae_route_attr;
hi_bool ae_route_ex_valid;
hi_isp_ae_route_ex ae_route_attr_ex;
hi_isp_ae_route ae_route_sf_attr;
hi_isp_ae_route_ex ae_route_sf_attr_ex;
hi_u16 man_ratio_enable;
hi_u32 arr_ratio[HI_ISP_EXP_RATIO_NUM];
hi_isp_iris_type iris_type;
hi_isp_piris_attr piris_attr;
hi_isp_iris_f_no max_iris_fno;
hi_isp_iris_f_no min_iris_fno;
hi_isp_ae_strategy ae_exp_mode;
hi_u16 iso_cal_coef;
hi_u8 ae_run_interval;
hi_u32 exp_ratio_max;
hi_u32 exp_ratio_min;
hi_bool diff_gain_support;
hi_isp_quick_start_param quick_start;
hi_isp_prior_frame prior_frame;
hi_bool ae_gain_sep_cfg;
hi_bool lhcg_support;
hi_u32 sns_lhcg_exp_ratio;
} hi_isp_ae_sensor_default;
```

成员

成员名称	描述
hist_thresh	五段直方图的分割门限值数组，数组内各元素的取值范围为[0,0xff]。推荐使用默认值，线性模式为{0xd,0x28,0x60,0x80}，BUILT_IN WDR模式为{0x20,0x40,0x60,0x80}，帧合成WDR模式为{0xc,0x18,0x60,0x80}。 #define HI_ISP_HIST_THRESH_NUM 4
ae_compensation	AE亮度目标值，取值范围为[0,255]，建议用0x38~0x40。
lines_per500ms	每500ms的总行数。
flicker_freq	抗闪频率，数值为当前电源频率的256倍。
fps	基准帧率。
hmax_times	Sensor读出一行的时间，单位：ns。

成员名称	描述
init_exposure	默认初始曝光量，等于曝光时间与曝光增益的乘积，其中曝光时间的单位为微秒(us)。AE算法采用该值作为初始5帧的曝光量，可用于运动DV加速启动。 建议关注快速启动的产品形态根据常用场景配置一个合适的初始曝光量，以达到AE快速收敛。线性/WDR模式切换时，也会采用该值作为切换后第一帧的曝光量。合理修改Sensor初始化序列的曝光时间和增益，将曝光时间(行数)*增益(6bit 小数精度)计算得到曝光量，并赋值给该变量，可使得切换更加平滑。若不设置，该值默认为0。
init_ae_speed	默认初始AE调节速度，AE算法采用该值作为初始100帧的调节速度，可用于运动 DV 加速启动。 建议关注快速启动的产品形态可将该值配置为128。若不设置，该值默认为0。AE算法内部会将该值钳位至[64,255]。
init_ae_tolerance	默认初始AE曝光容忍偏差，AE算法采用该值作为初始100帧的曝光容忍偏差值，可用于设置得到首次AE收敛稳定标志的亮度范围。AE统计亮度第一次落在[目标亮度-init_ae_tolerance, 目标亮度+init_ae_tolerance]范围内时，得到首次AE收敛稳定标志。若不设置，该值默认为0。若在cmos.c不对该值赋值或赋值为0，则AE算法内部将其置为ae_compensation/10。AE算法内部会将该值钳位至(0,255]。
full_lines_std	基准帧率时1帧的有效行数。
full_lines_max	Sensor降帧后1帧所能达到的最大行数，一般设为Sensor所支持的最大行数。
full_lines	Sensor每帧实际生效的总行数。使用AE算法时，回调cmos_fps_set和cmos_slow_framerate_set时必须给该值赋值，用于返回每帧实际生效的总行数。
binning_full_lines	Sensor在binning模式下的等效曝光总行数，以行为单位。
max_int_time	最大曝光时间，以行为单位。
min_int_time	最小曝光时间，以行为单位。
max_int_time_target	最大曝光时间目标值，以行为单位。
min_int_time_target	最小曝光时间目标值，以行为单位。
int_time_accu	曝光时间精度。
max_again	最大模拟增益，以倍为单位。
min_again	最小模拟增益，以倍为单位。
max_again_target	最大模拟增益目标值，以倍为单位。
min_again_target	最小模拟增益目标值，以倍为单位。
again_accu	模拟增益精度。

成员名称	描述
max_dgain	最大数字增益，以倍为单位。
min_dgain	最小数字增益，以倍为单位。
max_dgain_target	最大数字增益目标值，以倍为单位。
min_dgain_target	最小数字增益目标值，以倍为单位。
dgain_accu	数字增益精度。
max_isp_dgain_target	最大ISP数字增益目标值，以倍为单位。
min_isp_dgain_target	最小ISP数字增益目标值，以倍为单位。
isp_dgain_shift	ISP数字增益精度。
max_int_time_step	自动长帧模式下普通模式和长帧模式之前切换过程中，对短帧曝光时间减小的最大调整步长，以行数为单位。仅在自动长帧模式下有效。
max_time_step_enable	短帧曝光时间最大步长限定开关，该值为HI_TRUE时，短帧曝光时间最大步长限制功能生效。仅在2帧行合成WDR模式下生效。
max_inc_time_step	目前只开放max_inc_time_step[0]，max_inc_time_step[0]为短帧曝光时间增加的最大步长，以行为单位。即下一帧的曝光行数相比当前帧增加的最大曝光行数不超过max_inc_time_step [0]。仅在2帧行合成WDR模式下max_time_step_enable开关使能后生效。 HI_ISP_WDR_MAX_FRAME_NUM用于定义WDR合成的最大帧数，表示4。
max_dec_time_step	目前只开放max_dec_time_step[0]，max_dec_time_step[0]为短帧曝光时间减少的最大步长，以行为单位。即下一帧的曝光行数相比当前帧减少的最大曝光行数不超过max_dec_time_step [0]。仅在2帧行合成WDR模式下max_time_step_enable开关使能后生效。 HI_ISP_WDR_MAX_FRAME_NUM用于定义WDR合成的最大帧数，表示4。
lf_max_short_time	自动长帧模式下短帧最大曝光时间。
lf_min_exposure	自动长帧模式下强制输出长帧的最小曝光量。
ae_route_attr	AE默认分配路线。
ae_route_ex_valid	AE扩展分配路线是否生效开关，该值为HI_TRUE时使用扩展分配路线，否则使用普通AE分配路线。
ae_route_attr_ex	AE默认扩展分配路线，合理配置可用于优化WDR模式图像效果。
ae_route_sf_attr	AE短帧默认分配路线。

成员名称	描述
ae_route_sf_attr_ex	AE短帧默认扩展分配路线，合理配置可用于优化WDR模式图像效果。
man_ratio_enable	多帧合成WDR手动曝光比使能，该值为HI_TRUE时，曝光比固定，不会根据场景自动更新。
arr_ratio	多帧合成WDR相邻2帧默认曝光比。当man_ratio_enable为HI_TRUE时，采用arr_ratio[0]作为S/VS 默认曝光比，arr_ratio[1]作为M/S默认曝光比，arr_ratio[2]作为 L/M默认曝光比。上述VS，S，M，L 分别表示极短帧，短帧，中帧和长帧的曝光时间，2帧合成时只有VS和S；3帧合成时采用VS，S 和 M；4帧合成时采用VS，S，M 和 L。6bit小数精度。 数组内各元素取值范围：[0x40, 0xFFFF]。 HI_ISP_EXP_RATIO_NUM用于定义WDR曝光比的数目，表示3。
iris_type	默认镜头类型，DC-Iris 或 P-Iris。默认镜头类型与实际对接镜头类型不一致时，AI算法无法正常工作。
piris_attr	P-Iris参数，与具体镜头相关。只有默认镜头类型是P-Iris时，才需要用该结构体参数。
max_iris_fno	P-Iris可用的最大 F 值，与具体使用镜头相关，使用P-Iris时必须设置，用于限制实际生效的max_iris_fno_target和min_iris_fno_target，避免光圈目标值落到不合理的范围导致AE分配异常。 取值范围为[HI_ISP_IRIS_F_NO_32_0, HI_ISP_IRIS_F_NO_1_0]。
min_iris_fno	P-Iris可用的最小F值，与具体使用镜头相关，使用P-Iris时必须设置，用于限制实际生效的max_iris_fno_target和min_iris_fno_target，避免光圈目标值落到不合理的范围导致AE分配异常。 取值范围为[HI_ISP_IRIS_F_NO_32_0, HI_ISP_IRIS_F_NO_1_0]。
ae_exp_mode	默认曝光策略，高光优先或低光优先。建议FSWDR模式设置为低光优先，线性模式及BuiltIn WDR设置为高光优先。若不设置该值，默认为高光优先。
iso_cal_coef	ISO标定系数，用于保证拍照所需DCF信息中显示的ISO是标准的，8bit精度。 取值范围：[0x0, 0xFFFF]，默认值为 0x100。
ae_run_interval	默认AE算法运行间隔，以帧为单位。若不设置，则默认AE每帧执行一次。 取值范围：(0x0, 0xFF]

成员名称	描述
exp_ratio_max	默认最大曝光比，仅在多帧合成WDR模式下有效。 - 当man_ratio_enable为HI_FALSE时，exp_ratio_max表示最长帧与最短帧曝光时间比值的最大值。即2帧合成时表示S/VS曝光比的最大值，3帧合成时表示M/VS曝光比的最大值，4帧合成时表示L/VS曝光比的最大值。 - 当man_ratio_enable为HI_TRUE时，exp_ratio_max无效。6bit小数精度，0x40表示曝光比为1倍。 取值范围：[0x40, 0x4000]
exp_ratio_min	默认最小曝光比，仅在多帧合成WDR模式下有效。 - 当man_ratio_enable为HI_FALSE时，exp_ratio_min表示长帧曝光时间与短帧曝光时间比值的最小值。 - 当man_ratio_enable为HI_TRUE时，exp_ratio_min无效。6bit小数精度，0x40表示曝光比为1倍。 取值范围：[0x40, exp_ratio_max]
diff_gain_support	Sensor是否支持长短帧不同增益配置，该值为HI_TRUE时，Sensor支持长短帧不同Sensor模拟增益、Sensor数字增益配置，该值为HI_FALSE时，Sensor不支持长短帧不同Sensor模拟增益、Sensor数字增益配置。
quick_start	设置不带光敏快启相关参数。
prior_frame	WDR模式下曝光路线生效的优先帧，线性模式下此参数需配置为LONG_FRAME。
ae_gain_sep_cfg	长短帧增益是否分开分配。
lhcg_support	Sensor是否使用LCG+HCG模式，该值为HI_TRUE时，Sensor使用LCG+HCG模式，该值为HI_FALSE时，Sensor使用普通模式。
sns_lhcg_exp_ratio	LCG+HCG模式的基础曝光比。

注意事项

- 线性/WDR 模式切换时，会回调pfn_cmos_get_ae_default函数更新AE相关默认参数。若WDR模式要使用AE扩展分配路线而线性模式不需要，建议在cmos_get_ae_default函数里面先对AE路线清零：ae_route_ex_valid=HI_FALSE，ae_route_attr.total_num= 0，ae_route_attr_ex.total_num= 0，然后视需要在WDR分支赋值。
- AE算法采用init_exposure作为初始5帧的曝光量，可用于运动DV加速启动。建议关注快速启动的产品形态根据常用场景配置一个合适的初始曝光量，以达到AE快速收敛。FSWDR模式该值对应的是长帧曝光量，FSWDR模式若要快速启动，最好在cmos.c设置为固定曝光比，以减少曝光比调整的时间，待AE稳定后再根据需要设置为自动曝光比。若在cmos.c未给init_exposure赋值或将init_exposure赋值为0，则AE算法内部按起始曝光量为1024开始计算。

WDR模式切换时，AE算法内部会计算切换曝光量以保证切换平滑，如果希望在WDR模式切换时init_exposure生效，可以通过配置init_ae_speed为0xFFFFFFFF实现。

- 对于特定Sensor，如OV2718 DCG模式，内部仅支持固定曝光比，需要配置man_ratio_enable= HI_TRUE，并把ratio配置为Sensor支持的固定曝光比，如果使用自动曝光比或手动配置为不支持的曝光比，会使图像效果不正常。
- max_int_time_step自动长帧模式下普通模式和长帧模式之前切换过程中，对短帧曝光时间减小的最大调整步长，以行数为单位。仅在自动长帧模式下有效。此参数针对短帧曝光时间减小过快会出现坏帧的Sensor开放，一般Sensor可以设定为较大值，则可以受此参数限制。
- max_time_step_enable, max_inc_time_step[0], max_dec_time_step[0]，这部分参数针对2合1行合成WDR模式下，短帧曝光时间前后2帧之间增加或者减少有限制的Sensor开放。
- lf_max_short_time为自动长帧模式下短帧曝光时间的最大值，如果此参数设置过小会导致自动长帧模式下亮区噪声表现变差。
- 对于支持LCG+HCG模式的Sensor，如OV2775，如果使用该模式，需正确配置基础曝光比sns_lhcg_exp_ratio，并且必须配置prior_frame=HI_ISP_LONG_FRAME，ae_gain_sep_cfg= HI_TRUE，否则会使图像效果不正常。

1.9.20.3.27 hi_isp_ae_sensor_exp_func

说明定义

Sensor回调函数结构体。

定义

```
typedef struct {  
    hi_s32 (*pfn_cmos_get_ae_default)(hi_vi_pipe vi_pipe, hi_isp_ae_sensor_default *ae_sns_dft);  
    /* the function of sensor set fps */  
    hi_void (*pfn_cmos_fps_set)(hi_vi_pipe vi_pipe, hi_float f32_fps, hi_isp_ae_sensor_default *ae_sns_dft);  
    hi_void (*pfn_cmos_slow_framerate_set)(hi_vi_pipe vi_pipe, hi_u32 full_lines, hi_isp_ae_sensor_default  
    *ae_sns_dft);  
  
    /* while isp notify ae to update sensor regs, ae call these funcs. */  
    hi_void (*pfn_cmos_inttime_update)(hi_vi_pipe vi_pipe, hi_u32 int_time);  
    hi_void (*pfn_cmos_gains_update)(hi_vi_pipe vi_pipe, hi_u32 again, hi_u32 dgain);  
  
    hi_void (*pfn_cmos_again_calc_table)(hi_vi_pipe vi_pipe, hi_u32 *again_lin, hi_u32 *again_db);  
    hi_void (*pfn_cmos_dgain_calc_table)(hi_vi_pipe vi_pipe, hi_u32 *dgain_lin, hi_u32 *dgain_db);  
  
    hi_void (*pfn_cmos_get_inttime_max)(hi_vi_pipe vi_pipe, hi_u16 man_ratio_enable,  
    hi_u32 *ratio, hi_u32 *int_time_max, hi_u32 *int_time_min, hi_u32 *lf_max_int_time);  
  
    /* long frame mode set */  
    hi_void (*pfn_cmos_ae_fswdr_attr_set)(hi_vi_pipe vi_pipe, hi_isp_ae_fswdr_attr *ae_fswdr_attr);  
    hi_void (*pfn_cmos_ae_quick_start_status_set)(hi_vi_pipe vi_pipe, hi_bool quick_start_status);  
    hi_void (*pfn_cmos_exp_param_convert)(hi_vi_pipe vi_pipe, hi_isp_ae_convert_param *exp_param);  
} hi_isp_ae_sensor_exp_func;
```

成员

成员名称	描述
pfn_cmos_get_ae_default	获取AE算法库的初始值的回调函数指针。
pfn_cmos_fps_set	设置Sensor的帧率。
pfn_cmos_slow_frame_rate_set	设置Sensor的降帧。
pfn_cmos_inttime_update	设置Sensor的曝光时间。
pfn_cmos_gains_update	设置Sensor的模拟增益和数字增益。
pfn_cmos_again_calc_table	计算TABLE类型Sensor模拟增益。
pfn_cmos_dgain_calc_table	计算TABLE类型Sensor数字增益。
pfn_cmos_get_inttime_max	WDR模式下，计算短帧最大曝光时间的回调函数指针，与Sensor强相关。
pfn_cmos_ae_fswdr_attr_set	2to1LineWDR模式下，设置长帧模式。
pfn_cmos_ae_quick_start_status_set	设置AE无光敏快启收敛状态。
pfn_cmos_exp_param_convert	完成不同帧率等曝光量转换曝光参数属性，配套hi_mpi_isp_get_exp_convert提供。

注意事项

- 如果回调函数指针不需要赋值，需要置为NULL。
- 在hi_isp_ae_sensor_default中定义了曝光时间和增益的精度，pfn_cmos_inttime_update和pfn_cmos_gains_update中设置的曝光时间和增益都是带精度的值，如何转换成Sensor的配置值与Sensor强相关，请参阅Sensor手册。
- 不使用无光敏快启功能时，pfn_cmos_ae_quick_start_status_set需要设置为NULL。
- quick_start_status是AE无光敏快启收敛状态的标志位。该值为HI_TRUE时，无光敏快启收敛完成。

1.9.20.3.28 hi_isp_ae_sensor_register

说明

定义Sensor注册结构体。

定义

```
typedef struct {  
    hi_isp_ae_sensor_exp_func sns_exp;  
} hi_isp_ae_sensor_register;
```

成员

成员名称	描述
sns_exp	Sensor注册的回调函数结构体。

1.9.20.3.29 hi_isp_awb_agc_table

说明

定义饱和度初始化参数结构体。

定义

```
typedef struct {  
    hi_bool valid;  
    hi_u8 saturation[HI_ISP_AUTO_ISO_NUM];  
} hi_isp_awb_agc_table;
```

成员

成员名称	描述
valid	该结构体的数据是否有效，取值范围为 0、1。
saturation	根据增益动态调节饱和度的插值数组，数组内元素的取值范围为[0, 255]。 saturation为非单调递减序列。 HI_ISP_AUTO_ISO_NUM用于定义ISO档位数，表示16。

1.9.20.3.30 hi_isp_awb_ccm_tab

说明

定义不同色温下自动颜色校正矩阵系数。

定义

```
typedef struct {  
    hi_u16 color_temp;  
    hi_u16 ccm[HI_ISP_CCM_MATRIX_SIZE];  
} hi_isp_awb_ccm_tab;
```

成员

成员名称	描述
color_temp	当前配置的CCM对应的色温。 取值范围：[2000, 10000]
ccm	不同色温下的颜色校正矩阵，8bit小数精度。Bit[15]是符号位，0 表示正数，1 表示负数，例如0x8010表示-16。Bit[12],Bit[13],Bit[14]是无效的，即正数从0x0000到0x0FFF，负数从0x8000到0x8FFF，是CCM的合理参数。 取值范围：[0x0, 0xFFFF]。 HI_ISP_CCM_MATRIX_SIZE用于定义CCM矩阵参数个数，表示9。

1.9.20.3.31 hi_isp_awb_ccm

说明

定义CCM颜色校正矩阵属性。

定义

```
typedef struct {  
    hi_u16 ccm_tab_num;  
    hi_isp_awb_ccm_tab ccm_tab[HI_ISP_CCM_MATRIX_NUM];  
} hi_isp_awb_ccm;
```

成员

成员名称	描述
ccm_tab_num	当前配置的 CCM 的组数。 取值范围：[3, 7]。
ccm_tab	不同色温下的颜色校正矩阵和对应的色温值。 HI_ISP_CCM_MATRIX_NUM用于定义AWB可配置的CCM矩阵组数，表示7。

注意事项

如果选择三组CCM，推荐高色温CCM在D50光源标定，中色温CCM在TL84光源标定，低色温CCM在A光源标定。

1.9.20.3.32 hi_isp_awb_sensor_default

说明

定义AWB算法库的初始化参数结构体。

定义

```
typedef struct {
    hi_u16 wb_ref_temp;
    hi_u16 gain_offset[HI_ISP_BAYER_CHN_NUM];
    hi_s32 wb_para[HI_ISP_AWB_CURVE_PARA_NUM];
    hi_u16 golden_rgain;
    hi_u16 golden_bgain;
    hi_u16 sample_rgain;
    hi_u16 sample_bgain;
    hi_isp_awb_agc_table agc_tbl;
    hi_isp_awb_ccm ccm;
    hi_u16 init_rgain;
    hi_u16 init_ggain;
    hi_u16 init_bgain;
    hi_u8 awb_run_interval;
    hi_u16 init_ccm[HI_ISP_CCM_MATRIX_SIZE];
} hi_isp_awb_sensor_default;
```

成员

成员名称	描述
wb_ref_temp	静态白平衡校正色温，取值范围为[0,0xFFFF]。
gain_offset	静态白平衡的R、Gr、Gb、B颜色通道的增益值数组，数组内元素的取值范围为[0, 0xFFFF]。 HI_ISP_BAYER_CHN_NUM用于定义Bayer数据的通道数目，表示4。
wb_para	校正工具给出的白平衡参数数组，取值范围为[0, 0xFFFFFFFF]。 HI_ISP_AWB_CURVE_PARA_NUM用于定义昇腾AWB标定的Planck 曲线参数个数，表示6。
golden_rgain	Golden样机在线标定得到的G/R值。
golden_bgain	Golden样机在线标定得到的G/B值。
sample_rgain	当前样机在线标定得到的G/R值。
sample_bgain	当前样机在线标定得到的G/B值。
agc_tbl	该结构体的数据是否有效，取值范围为[0,1]。
	根据增益动态调节饱和度的插值数组，取值范围为[0, 255]。
ccm	当前配置的CCM的组数。 取值范围： [3, 7]
	不同色温下的颜色校正矩阵和对应的色温值。
init_rgain	ISP启动时R通道白平衡增益初始值。
init_ggain	ISP启动时G通道白平衡增益初始值。
init_bgain	ISP启动时B通道白平衡增益初始值。

成员名称	描述
awb_run_interval	ISP启动时AWB工作频率。 取值范围：[0x1, 0xFF]
init_ccm	ISP启动时CCM初始值数组。 HI_ISP_CCM_MATRIX_SIZE用于定义CCM矩阵参数个数，表示9。

注意事项

参考色温即静态白平衡校正的环境色温，需要提供色度计测量的实际值。

1.9.20.3.33 hi_isp_awb_sensor_exp_func

说明

定义Sensor回调函数结构体。

定义

```
typedef struct {  
    hi_s32 (*pfn_cmos_get_awb_default)(hi_vi_pipe vi_pipe, hi_isp_awb_sensor_default *awb_sns_dft);  
} hi_isp_awb_sensor_exp_func;
```

成员

成员名称	描述
vi_pipe	vi_pipe号。
pfn_cmos_get_awb_default	获取 AWB 算法库的初始值的回调函数指针。

1.9.20.3.34 hi_isp_awb_sensor_register

说明

定义Sensor注册结构体。

定义

```
typedef struct {  
    hi_isp_awb_sensor_exp_func sns_exp;  
} hi_isp_awb_sensor_register;
```

成员

成员名称	描述
sns_exp	Sensor注册的回调函数结构体。

1.9.20.4 MIPI Rx ioctl 命令字参数

1.9.20.4.1 combo_dev_t

说明

定义MIPI Rx、SLVS设备号。

定义

```
typedef unsigned int combo_dev_t;
```

注意事项

- MIPI RX设备有效范围为[0, 4)。
- SLVS-EC设备有效范围为[0, 2)。

1.9.20.4.2 sns_rst_source_t

说明

Sensor的复位信号线编号，软件上称为Sensor的复位源。

定义

```
typedef unsigned int sns_rst_source_t;
```

注意事项

- 有效值范围为[0, 2)。
- 每条Sensor复位信号线可以接多个Sensor，用户需要根据板子的连线确认Sensor复位信号线编号。

1.9.20.4.3 sns_clk_source_t

说明

Sensor的时钟信号线编号，软件上称为Sensor的时钟源。

定义

```
typedef unsigned int sns_clk_source_t;
```

注意事项

- 有效值范围为[0, 4)。
- 每条Sensor时钟信号线可以接多个Sensor，用户需要根据板子的连线确认Sensor时钟信号线编号。

1.9.20.4.4 lane_divide_mode_t

说明

定义MIPI Rx的LANE分布模式。各模式的含义请参见[表1-49](#)。

定义

```
typedef enum {  
    LANE_DIVIDE_MODE_0 = 0,  
    LANE_DIVIDE_MODE_1 = 1,  
    LANE_DIVIDE_MODE_2 = 2,  
    LANE_DIVIDE_MODE_3 = 3,  
    LANE_DIVIDE_MODE_BUTT  
} lane_divide_mode_t;
```

注意事项

只有MIPI Rx需要设置LANE的分布模式。

1.9.20.4.5 input_mode_t

说明

定义MIPI Rx输入接口类型。

定义

```
typedef enum {  
    INPUT_MODE_MIPI = 0x0, /* mipi */  
    INPUT_MODE_SUBLVDS = 0x1, /* SUB_LVDS */  
    INPUT_MODE_LVDS = 0x2, /* LVDS */  
    INPUT_MODE_HISPI = 0x3, /* HISPI */  
    INPUT_MODE_SLVS = 0x4,  
    INPUT_MODE_BUTT  
} input_mode_t;
```

1.9.20.4.6 img_rect_t

说明

定义MIPI的裁剪属性。

定义

```
typedef struct {  
    int x;  
    int y;  
    unsigned int width;  
    unsigned int height;  
} img_rect_t;
```

成员

成员名称	描述
x	裁剪起始位置x。
y	裁剪起始位置y。 SLV-EC的裁剪Y坐标必须要大于等于Sensor输出有效行的行号。
width	裁剪宽度，单位：像素。
height	裁剪高度，单位：像素。

1.9.20.4.7 slvs_lane_rate_t

说明

定义SLVS LANE的输入速率。

定义

```
typedef enum {  
    SLVS_LANE_RATE_LOW = 0,      /* 1152Mbps */  
    SLVS_LANE_RATE_HIGH = 1,    /* 2304Mbps */  
    SLVS_LANE_RATE_BUTT  
} slvs_lane_rate_t;
```

1.9.20.4.8 slvs_err_check_mode_t

说明

SLVS的CRC (Cyclic Redundancy Check) 和ECC (Error Control Coding) 模式。

定义

```
typedef enum {  
    SLVS_ERR_CHECK_MODE_NONE = 0x0, /* disable ECC & CRC */  
    SLVS_ERR_CHECK_MODE_CRC = 0x1,  /* enable  CRC */  
    SLVS_ERR_CHECK_MODE_ECC_2BYTE = 0x2, /* enable  2 Byte ECC */  
    SLVS_ERR_CHECK_MODE_ECC_4BYTE = 0x3, /* enable  4 Byte ECC */  
    SLVS_ERR_CHECK_MODE_BUTT  
} slvs_err_check_mode_t;
```

成员

成员名称	描述
SLVS_ERR_CHECK_MODE_NONE	ECC与CRC都不启用。
SLVS_ERR_CHECK_MODE_CRC	启用CRC。

成员名称	描述
SLVS_ERR_CHECK_MODE_ECC_2BYTE	启用2 Byte ECC。
SLVS_ERR_CHECK_MODE_ECC_4BYTE	启用4 Byte ECC。

注意事项

SLVS ECC与CRC功能是互斥的，不能同时打开。

1.9.20.4.9 data_type_t

说明

定义传输的数据类型。

定义

```
typedef enum {  
    DATA_TYPE_RAW_8BIT = 0,  
    DATA_TYPE_RAW_10BIT = 1,  
    DATA_TYPE_RAW_12BIT = 2,  
    DATA_TYPE_RAW_14BIT = 3,  
    DATA_TYPE_YUV420_8BIT_NORMAL = 5,  
    DATA_TYPE_YUV422_8BIT = 7,  
    DATA_TYPE_YUV422_PACKED = 8,  
    DATA_TYPE_BUTT  
} data_type_t;
```

成员

成员名称	描述
DATA_TYPE_RAW_8BIT	8BIT的RAW数据。
DATA_TYPE_RAW_10BIT	10BIT的RAW数据。
DATA_TYPE_RAW_12BIT	12BIT的RAW数据。
DATA_TYPE_RAW_14BIT	14BIT的RAW数据。
DATA_TYPE_YUV420_8BIT_NORMAL	8BIT的YUV420数据，NORMAL模式。
DATA_TYPE_YUV422_8BIT	8BIT的YUV422数据。
DATA_TYPE_YUV422_PACKED	YUV422数据在MIPI按16bit raw打包接收。

1.9.20.4.10 mipi_data_rate_t

说明

MIPI Rx输入速率。

定义

```
typedef enum
{
    MIPI_DATA_RATE_X1 = 0,      /* output 1 pixel per clock */
    MIPI_DATA_RATE_X2 = 1,      /* output 2 pixel per clock */
    MIPI_DATA_RATE_BUTT
} mipi_data_rate_t;
```

成员

成员名称	描述
MIPI_DATA_RATE_X1	1拍1像素。
MIPI_DATA_RATE_X2	1拍2像素。当前不支持该选项。

1.9.20.4.11 mipi_wdr_mode_t

说明

定义MIPI WDR模式。

定义

```
typedef enum {
    HI_MIPI_WDR_MODE_NONE = 0x0,
    HI_MIPI_WDR_MODE_VC = 0x1, /* Virtual Channel */
    HI_MIPI_WDR_MODE_DT = 0x2, /* Data Type */
    HI_MIPI_WDR_MODE_DOL = 0x3, /* DOL Mode */
    HI_MIPI_WDR_MODE_BUTT
} mipi_wdr_mode_t;
```

成员

成员名称	描述
HI_MIPI_WDR_MODE_NONE	线性模式。
HI_MIPI_WDR_MODE_VC	使用Packet header中的Virtual Channel区分长短曝光帧。
HI_MIPI_WDR_MODE_DT	使用Packet header中的自定义Data type区分长短曝光帧。
HI_MIPI_WDR_MODE_DOL	表示DOL模式WDR，使用Packet header之后的一个像素识别长短曝光帧

注意事项

需与当前对接的Sensor模组配置对齐，且需与后端VI、ISP的模式配置对齐。

1.9.20.4.12 slvs_wdr_mode_t

说明

定义SLVS的WDR模式。

定义

```
typedef enum {  
    HI_SLVS_WDR_MODE_NONE    = 0x0,  
    HI_SLVS_WDR_MODE_2F      = 0x1,  
    HI_SLVS_WDR_MODE_DOL_2F  = 0x4  
} slvs_wdr_mode_t;
```

成员

成员名称	描述
HI_SLVS_WDR_MODE_NONE	线性模式。
HI_SLVS_WDR_MODE_2F	2合1WDR。 SLVS接口不支持该选项。
HI_SLVS_WDR_MODE_DOL_2F	DOL模式2合1WDR。

注意事项

- Built-in WDR模式和帧合成WDR模式都需要配置为HI_SLVS_WDR_MODE_NONE。
- DOL WDR模式需要配置为HI_SLVS_WDR_MODE_DOL_2F。
- 需与当前对接的Sensor模组配置对齐，且需与后端VI、ISP的模式配置对齐。

1.9.20.4.13 lvds_wdr_mode_t

说明

LVDS的WDR模式。

定义

```
typedef enum {  
    HI_LVDS_WDR_MODE_NONE    = 0x0,  
    HI_LVDS_WDR_MODE_2F      = 0x1,  
    HI_LVDS_WDR_MODE_DOL_2F  = 0x4  
} lvds_wdr_mode_t;
```

成员

成员名称	描述
HI_LVDS_WDR_MODE_NONE	线性模式。
HI_LVDS_WDR_MODE_2F	2合1WDR。 SLVS接口不支持该选项。
HI_LVDS_WDR_MODE_DOL_2F	DOL模式2合1WDR。

注意事项

- Built-in WDR 模式和帧合成WDR 模式都需要配置为 HI_LVDS_WDR_MODE_NONE。
- DOL WDR 模式需要配置为 HI_LVDS_WDR_MODE_DOL_2F。
- 需与当前对接的Sensor模组配置对齐，且需与后端VI、ISP的模式配置对齐。

1.9.20.4.14 mipi_dev_attr_t

说明

定义MIPI设备属性。

定义

```
typedef struct {  
    data_type_t input_data_type; /* data type: 8/10/12/14/16 bit */  
    mipi_wdr_mode_t wdr_mode; /* MIPI WDR mode */  
    short lane_id[MIPI_LANE_NUM]; /* lane_id: -1 - disable */  
    union {  
        short data_type[WDR_VC_NUM]; /* attribute of MIPI WDR mode.  
AUTO:mipi_wdr_mode_t:HI_MIPI_WDR_MODE_DT; */  
    };  
} mipi_dev_attr_t;
```

成员

成员名称	描述
input_data_type	传输的数据类型。
lane_id	发送端(Sensor)和接收端(MIPI Rx) Lane的对应关系，未使用的Lane设置为-1。 MIPI_LANE_NUM表示8。
wdr_mode	MIPI WDR模式。
data_type	当wdr_mode为 HI_MIPI_WDR_MODE_DT时，需要设置 data_type，表示不同曝光长度数据对应的Data Type。 WDR_VC_NUM表示4。

注意事项

Atlas 200/500 A2推理产品上有8条LANE，如果工作模式采用2+2+2+2方式，则设备0需要使用lane0和lane2，所以lane_id配置为{0, 2, -1, -1, -1, -1, -1, -1}。

1.9.20.4.15 lvds_sync_mode_t

说明

定义LVDS同步方式。

定义

```
typedef enum {  
    LVDS_SYNC_MODE_SOF = 0, /* Sensor SOL, EOL, SOF, EOF */  
    LVDS_SYNC_MODE_SAV, /* SAV, EAV */  
    LVDS_SYNC_MODE_BUTT  
} lvds_sync_mode_t;
```

成员

成员名称	同步方式
LVDS_SYNC_MODE_SOF	SOF、EOF、SOL、EOL。
LVDS_SYNC_MODE_SAV	invalid SAV、invalid EAV、valid SAV、valid EAV。

1.9.20.4.16 lvds_bit_endian_t

说明

定义比特位大小端模式。

定义

```
typedef enum {  
    LVDS_ENDIAN_LITTLE = 0x0,  
    LVDS_ENDIAN_BIG = 0x1,  
    LVDS_ENDIAN_BUTT  
} lvds_bit_endian_t;
```

1.9.20.4.17 lvds_vsync_type_t

说明

定义LVDS VSync（垂直同步）类型。

定义

```
typedef enum {  
    LVDS_VSYNC_NORMAL = 0x00,  
    LVDS_VSYNC_SHARE = 0x01,  
    LVDS_VSYNC_HCONNECT = 0x02,  
}
```

```
LVDS_VSYNC_BUTT
} lvds_vsync_type_t;
```

成员

成员名称	描述
LVDS_VSYNC_NORMAL	长短曝光帧有独立的SOF-EOF、SOL-EOL或者invalid SAV-invalid EAV, valid SAV-valid EAV 。
LVDS_VSYNC_SHARE	长短曝光帧共用一对SOF-EOF标识，短曝光的起始几行用固定值填充。
LVDS_VSYNC_HCONNECT	长短曝光帧共用一对SAV-EAV标识，长短曝光帧之间是固定周期的消隐。

参考信息

- LVDS_VSYNC_SHARE同步方式：

SOF	Long Exposure	EOL	Horizontal Blanking	SOL	Padding	EOL	Horizontal Blanking
SOL	Padding				Short Exposure	EOF	
SOV	V.BLK	EOV	-	SOV	V.BLK	EOV	-

- LVDS_VSYNC_HCONNECT同步方式：

SA V	Long Exposure frame	Horizon tal Blankin g(fix period)	V.BLK	Horizon tal Blankin g(fix period)	V.BLK	EA V	Horizo ntal Blanki ng
			Short ExposureFra me1		Short Exposur e Frame2		
	V.BLK		V.BLK				
	V.BLK						-

1.9.20.4.18 lvds_vsync_attr_t

说明

定义LVDS VSync（垂直同步）参数。

定义

```
typedef struct {
    lvds_vsync_type_t sync_type;
    unsigned short hblank1;
```

```
    unsigned short hblank2;  
} lvds_vsync_attr_t;
```

成员

成员名称	描述
sync_type	长短曝光帧同步方式。
hblank1	当同步方式为LVDS_VSYNC_HCONNECT，需要配置，表示Hconnect得消隐区长度。
hblank2	当同步方式为LVDS_VSYNC_HCONNECT，需要配置，表示Hconnect得消隐区长度。

1.9.20.4.19 lvds_fid_type_t

说明

定义FID (Frame Identification Id) 的类型。

定义

```
typedef enum {  
    LVDS_FID_NONE = 0x00,  
    LVDS_FID_IN_SAV = 0x01, /* frame identification id in SAV 4th */  
    LVDS_FID_IN_DATA = 0x02, /* frame identification id in first data */  
    LVDS_FID_BUTT  
} lvds_fid_type_t;
```

成员

成员名称	描述
LVDS_FID_NONE	不使用FID。
LVDS_FID_IN_SAV	FID插入在SAV第4个字段中，DOL 4个字段的同步码需要将lvds_fid_attr_t.fid_type配置为LVDS_FID_IN_SAV。
LVDS_FID_IN_DATA	FID作为Frame Information Column插入在同步码之后的第一个像素之前，DOL 5 个字段的同步码需要将lvds_fid_attr_t.fid_type配置为LVDS_FID_IN_DATA。

1.9.20.4.20 lvds_fid_attr_t

说明

定义FID (Frame Identification Id) 属性的配置信息。

定义

```
typedef struct {  
    lvds_fid_type_t fid_type;
```

```
    unsigned char output_fil;  
} lvds_fid_attr_t;
```

成员

成员名称	描述
fid_type	LVDS DOL模式下的FID类型
output_fil	DOL模式中的Frame information line紧接在 V-Blanking 之后输出，Frame ID是Frame information line中的第一个像素值。 Frame information line中并不包含有效的视频数据： - 如果output_fil设置为HI_TRUE，Frame information line会输出到后端设备。 - 如果output_fil设置为HI_FALSE，MIPI Rx将丢弃这一行数据。

1.9.20.4.21 lvds_dev_attr_t

说明

定义LVDS/SubLVDS/HiSPi设备属性。

定义

```
typedef struct {  
    data_type_t input_data_type; /* data type: 8/10/12/14 bit */  
    lvds_wdr_mode_t wdr_mode; /* WDR mode */  
    lvds_sync_mode_t sync_mode; /* sync mode: SOF, SAV */  
    lvds_vsync_attr_t vsync_attr; /* normal, share, hconnect */  
    lvds_fid_attr_t fid_attr; /* frame identification code */  
    lvds_bit_endian_t data_endian; /* data endian: little/big */  
    lvds_bit_endian_t sync_code_endian; /* sync code endian: little/big */  
    short lane_id[LVDS_LANE_NUM]; /* lane_id: -1 - disable */  
    /* each vc has 4 params, sync_code[i]:  
    sync_mode is SYNC_MODE_SOF: SOF, EOF, SOL, EOL  
    sync_mode is SYNC_MODE_SAV: invalid sav, invalid eav, valid sav, valid eav */  
    unsigned short sync_code[LVDS_LANE_NUM][WDR_VC_NUM][SYNC_CODE_NUM];  
} lvds_dev_attr_t;
```

成员

成员名称	描述
input_data_type	传输的数据类型。
wdr_mode	WDR模式。
sync_mode	LVDS同步模式。
vsync_attr	vsync类型，当wdr_mod为DOL模式并且sync_mode 为 LVDS_SYNC_MODE_SAV 时，需要配置vsync的类型。

成员名称	描述
fid_attr	frame identification类型，当wdr_mode为DOL模式，并且sync_mode为 LVDS_SYNC_MODE_SAV时，需要配置。
data_endian	数据大小端模式。
sync_code_endian	同步码大小端模式。
lane_id	发送端(Sensor)和接收端(MIPI Rx) lane的对应关系，未使用的lane设置为-1。 LVDS_LANE_NUM表示8。
sync_code	同步码共4个，前3个同步码固定为全1、全0、全0，不需要配置，只需配置第 4 个同步码。每lane有16个配置项，共四行四列。每一行代表一个Virtual Channel，WDR模式时 4行的配置需要根据发送端配置不同的同步码，线性模式时4行应配置为相同的同步码。根据不同的同步模式，4列的同步码分别表示SOF/EOF/ SOL/EOL 的同步码或者invalid SAV/invalid EAV/ valid SAV/valid EAV的同步码。 LVDS_LANE_NUM表示8。 WDR_VC_NUM表示4。 SYNC_CODE_NUM表示4。

注意事项

使用该结构体，LVDS的LANE同步码同步到达。

1.9.20.4.22 slvs_dev_attr_t

说明

定义SLVS设备属性。

定义

```
typedef struct {
    data_type_t      input_data_type;
    slvs_wdr_mode_t  wdr_mode;
    slvs_lane_rate_t lane_rate;
    int              sensor_valid_width;
    short            lane_id[SLVS_LANE_NUM]; /* lane_id: -1 - disable */
    slvs_err_check_mode_t err_check_mode; /* ECC CRC mode */
} slvs_dev_attr_t;
```

成员

成员名称	描述
input_data_type	传输的数据类型。
wdr_mode	WDR模式。

成员名称	描述
lane_rate	SLVS Lane速率。
sensor_valid_width	一行数据包的像素个数，仅低14bit有效，即最大16383像素。
lane_id	发送端(Sensor)和接收端(SLVS) Lane的对应关系。未使用的Lane设置为-1。 SLVS_LANE_NUM表示8。
err_check_mode	SLVS的CRC、ECC 模式，必须与Sensor端匹配。

注意事项

- SLVS只能支持线性模式和2合1WDR模式。
- SLVS接口不受[HI_MIPI_SET_HS_MODE](#)的限制。
- SLVS 的设备0和1的Lane可以在lane0-lane7之间任意选择，两个设备的Lane不能重复。

1.9.20.4.23 combo_dev_attr_t

说明

combo设备属性，由于MIPI Rx能够对接CSI-2、LVDS、HiSPi等时序，所以将MIPI Rx称为combo设备。

定义

```
typedef struct {
    combo\_dev\_t devno;
    input\_mode\_t input_mode;
    mipi\_data\_rate\_t data_rate;
    img\_rect\_t img_rect;
    union {
        mipi\_dev\_attr\_t mipi_attr;
        lvds\_dev\_attr\_t lvds_attr;
        slvs\_dev\_attr\_t slvs_attr;
    };
} combo\_dev\_attr\_t;
```

成员

成员名称	描述
devno	MIPI Rx，SLVS设备号。
input_mode	输入接口类型。
img_rect	图像裁剪区域。
data_rate	接口传输速率。当前版本只支持1拍1像素，所以必须设置为MIPI_DATA_RATE_X1。

成员名称	描述
mipi_attr	如果input_mode配置为INPUT_MODE_MIPI，则必须配置mipi_attr。
lvds_attr	如果input_mode配置为INPUT_MODE_SUBLVDS/ INPUT_MODE_LVDS/ INPUT_MODE_HISPI，则必须配置lvds_attr。
slvs_attr	如果input_mode配置为INPUT_MODE_SLVS，则必须配置slvs_attr。
lvds_attr_ex	如果input_mode配置为INPUT_MODE_LVDS_EX，则必须配置lvds_attr_ex。

1.9.20.4.24 sns_clk_freq_t

说明

Sensor从模式下的时钟输入频率。

定义

```
typedef enum {  
    SENSOR_CLK_74P25MHz = 0x00,  
    SENSOR_CLK_72MHz     = 0x01,  
    SENSOR_CLK_54MHz     = 0x02,  
    SENSOR_CLK_50MHz     = 0x03,  
    SENSOR_CLK_24MHz     = 0x04,  
    SENSOR_CLK_37P125MHz = 0x05,  
    SENSOR_CLK_36MHz     = 0x06,  
    SENSOR_CLK_27MHz     = 0x07,  
    SENSOR_CLK_25MHz     = 0x08,  
    SENSOR_CLK_12MHz     = 0x09,  
    SENSOR_CLK_FREQ_BUTT  
} sns_clk_freq_t;
```

成员

成员名称	描述
SENSOR_CLK_74P25MHz	时钟频率为74.25MHz
SENSOR_CLK_72MHz	时钟频率为72MHz
SENSOR_CLK_54MHz	时钟频率为54MHz
SENSOR_CLK_50MHz	时钟频率为50MHz
SENSOR_CLK_24MHz	时钟频率为24MHz
SENSOR_CLK_37P125MHz	时钟频率为37.125MHz
SENSOR_CLK_36MHz	时钟频率为36MHz

成员名称	描述
SENSOR_CLK_27MHz	时钟频率为27MHz
SENSOR_CLK_25MHz	时钟频率为25MHz
SENSOR_CLK_12MHz	时钟频率为12MHz

1.9.20.4.25 sns_clk_cfg_t

说明

Sensor从模式下的Sensor时钟输入配置。

定义

```
typedef struct {  
    sns_clk_source_t clk_source;  
    sns_clk_freq_t clk_freq;  
} sns_clk_cfg_t;
```

成员

成员名称	描述
clk_source	时钟信号源序号。
clk_freq	时钟频率。

1.9.20.5 VI 视频输入

1.9.20.5.1 hi_vi_dev

说明

定义VI设备号。

定义

```
typedef hi_s32 hi_vi_dev;
```

1.9.20.5.2 hi_vi_pipe

说明

定义VI PIPE号。

定义

```
typedef hi_s32 hi_vi_pipe;
```

1.9.20.5.3 hi_vi_chn

说明

定义VI通道号。

定义

```
typedef hi_s32 hi_vi_chn;
```

1.9.20.5.4 hi_vi_intf_mode

说明

定义视频设备的接口模式 。

定义

```
typedef enum {  
    HI_VI_MODE_MIPI = 4,  
    HI_VI_MODE_MIPI_YUV420_NORM,  
    HI_VI_MODE_MIPI_YUV420_LEGACY,  
    HI_VI_MODE_MIPI_YUV422,  
    HI_VI_MODE_LVDS,  
    HI_VI_MODE_HISPI,  
    HI_VI_MODE_SLVS,  
    HI_VI_MODE_BUTT  
} hi_vi_intf_mode;
```

成员

表 1-81 视频设备接口模式

成员名称	描述
HI_VI_INTF_MODE_MIPI	输入数据符合MIPI协议，用于传输RAW数据。
HI_VI_INTF_MODE_MIPI_YUV420_NORM	输入数据符合MIPI协议，用于传输YUV420 normal模式的数据。
HI_VI_INTF_MODE_MIPI_YUV420_LEGACY	输入数据符合MIPI协议，用于传输YUV420 legacy模式的数据。当前版本不支持该模式。
HI_VI_INTF_MODE_MIPI_YUV422	输入数据符合MIPI协议，用于传输YUV422数据。
HI_VI_MODE_LVDS	输入数据符合LVDS协议。当前版本不支持该模式。
HI_VI_MODE_HISPI	输入数据符合HISPI协议。当前版本不支持该模式。
HI_VI_MODE_SLVS	输入数据符合SLVS-EC协议。
HI_VI_MODE_BUTT	预留值。

1.9.20.5.5 hi_data_rate

说明

定义设备的速率。

定义

```
typedef enum {  
    HI_DATA_RATE_X1 = 0,  
    HI_DATA_RATE_X2 = 1,  
    HI_DATA_RATE_BUTT  
} hi_data_rate;
```

成员

成员名称	描述
HI_DATA_RATE_X1	一拍一像素。
HI_DATA_RATE_X2	一拍两像素，当前版本不支持。
HI_DATA_RATE_BUTT	预留值。

1.9.20.5.6 hi_vi_scan_mode

说明

定义视频设备接收的是隔行或逐行输入图像。

定义

```
typedef enum {  
    HI_VI_SCAN_PROGRESSIVE = 0,  
    HI_VI_SCAN_INTERLACED,  
    HI_VI_SCAN_BUTT  
} hi_vi_scan_mode;
```

成员

表 1-82 视频设备接收扫描模式

成员名称	描述
HI_VI_SCAN_PROGRES SIVE	VI输入为逐行图像。
HI_VI_SCAN_INTERLA CED	VI输入为隔行图像。
HI_VI_SCAN_BUTT	预留值。

1.9.20.5.7 hi_vi_data_seq

说明

定义视频设备接收的YUV数据的数据排列顺序 。

定义

```
typedef enum {  
    HI_VI_DATA_SEQ_VUVU = 0,  
    HI_VI_DATA_SEQ_UVUV,  
    HI_VI_DATA_SEQ_UYVY,  
    HI_VI_DATA_SEQ_VYUY,  
    HI_VI_DATA_SEQ_YUYV,  
    HI_VI_DATA_SEQ_YVYU,  
    HI_VI_DATA_SEQ_BUTT  
} hi_vi_data_seq;
```

成员

成员名称	描述
HI_VI_DATA_SEQ_VUVU	YUV数据通过分离模式输入时，C分量的输入排列顺序为VUVU。
HI_VI_DATA_SEQ_UVUV	YUV数据通过分离模式输入时，C分量的输入排列顺序为UVUV。
HI_VI_DATA_SEQ_UYVY	YUV数据通过复合模式输入时，顺序为UYVY。当前版本不支持。
HI_VI_DATA_SEQ_VYUY	YUV数据通过复合模式输入时，顺序为VYUY。当前版本不支持。
HI_VI_DATA_SEQ_YUYV	YUV数据通过复合模式输入时，顺序为YUYV。当前版本不支持。
HI_VI_DATA_SEQ_YVYU	YUV数据通过复合模式输入时，顺序为YVYU。当前版本不支持。
HI_VI_DATA_SEQ_BUTT	预留值。

1.9.20.5.8 hi_vi_data_type

说明

VI输入数据类型枚举。

定义

```
typedef enum {  
    HI_VI_DATA_TYPE_RAW = 0,  
    HI_VI_DATA_TYPE_YUV,  
    HI_VI_DATA_TYPE_BUTT  
} hi_vi_data_type;
```

成员

成员名称	描述
HI_VI_DATA_TYPE_RAW	输入数据类型为 RGB，VI 前端一般接的是Sensor。
HI_VI_DATA_TYPE_YUV	输入数据类型为 YUV，VI 前端一般接的是 AD。
HI_VI_DATA_TYPE_BUFFER	预留值。

1.9.20.5.9 hi_vi_dev_attr

说明

定义视频输入设备的属性。

定义

```
typedef struct {  
    hi_vi_intf_mode    intf_mode;  
    hi_vi_scan_mode    scan_mode;  
    hi_vi_data_seq      data_seq;  
    hi_vi_data_type     data_type;  
    hi_size             in_size;  
    hi_vi_wdr_attr      wdr_attr;  
    hi_data_rate         data_rate;  
    hi_u32              reserved[30];  
} hi_vi_dev_attr;
```

成员

成员名称	描述
intf_mode	接口模式。
scan_mode	输入扫描模式 (逐行、隔行)。当前仅支持逐行。
data_seq	输入YUV数据的排列顺序 (仅支持YUV格式，如VUVU或UVUV等)。 当data_type为HI_VI_DATA_TYPE_YUV时，需要配置data_seq。
data_type	输入数据类型，Sensor输入一般为Raw，部分内置ISP的Sensor输入也可以为YUV。
in_size	VI设备可设置要捕获图像的高宽。 捕获图像的最小宽高与最大宽高： 宽度： [120, 16384] 高度： [120, 16384]

成员名称	描述
data_rate	设备的速率，可选择是1拍1像素还是1拍2像素。 当前仅支持1拍1像素。
wdr_addr	WDR属性。
reserved	预留参数，为保证后续版本兼容性，请务必使用memset结构体方式进行清零初始化，在代码中必须避免显式对reserved字段进行访问。

1.9.20.5.10 hi_vi_dev_bind_pipe

说明

定义VI DEV与PIPE 的绑定关系。

定义

```
typedef struct {  
    hi_u32 num;  
    hi_vi_pipe pipe_id[HI_VI_MAX_PHYS_PIPE_NUM];  
} hi_vi_dev_bind_pipe;
```

成员

成员名称	描述
num	该VI Dev所绑定的PIPE数量，取值范围[1, 4]。
pipe_id	该VI Dev绑定的PIPE号。 HI_VI_MAX_PHYS_PIPE_NUM表示4。

1.9.20.5.11 hi_vi_wdr_attr

说明

定义VI WDR融合组的属性 。

定义

```
typedef struct {  
    hi_wdr_mode wdr_mode;  
    hi_u32 cache_line;  
} hi_vi_wdr_attr;
```


成员

成员名称	描述
wdr_mode	WDR工作模式，分为帧模式、行模式、非WDR等三大类。
cache_line	预留参数。

说明

WDR模式配置需与Sensor模组配置、Mipi RX配置对齐，否则会导致出图异常。

1.9.20.5.12 hi_vi_pipe_attr

说明

定义VI PIPE属性。

定义

```
typedef struct {  
    hi_vi_pipe_bypass_mode pipe_bypass_mode;  
    hi_bool                 isp_bypass;  
    hi_size                 size;  
    hi_pixel_format         pixel_format;  
    hi_compress_mode        compress_mode;  
    hi_data_bit_width        bit_width;  
    hi_frame_rate_ctrl      frame_rate_ctrl;  
    hi_u32                   depth;  
    hi_u32                   reserved[10];  
} hi_vi_pipe_attr;
```

成员

成员名称	描述
pipe_bypass_mode	VI PIPE的bypass模式。静态属性，创建PIPE时设定，不可更改。
isp_bypass	ISP是否运行。静态属性，创建PIPE时设定，不可更改。 - HI_FALSE：ISP正常运行。 - HI_TRUE：ISP不运行。 当输入为Raw数据时，必须设置为HI_FALSE；当输入为YUV时则可以设置为HI_TRUE或者HI_FALSE。
size	输入图像宽高。需根据所绑定的VI设备属性的宽高或调用hi_mpi_vi_set_pipe_pre_crop接口配置的宽高，保证大小一致，需与ISP公共属性所设置的宽高需保持一致。静态属性，创建PIPE时设定，不可更改。 取值范围：[120, 16384]。

成员名称	描述
pixel_format	像素格式。静态属性，创建PIPE时设定，不可更改。 如果是YUV图像，则仅pipe2和pipe3支持，且仅支持输出HI_PIXEL_FORMAT_YVU_SEMIPLANAR_422、HI_PIXEL_FORMAT_YVU_SEMIPLANAR_420、HI_PIXEL_FORMAT_YUV_400。 如果是RAW图像，则仅支持位宽为8、10、12、14的RGB BAYER格式。
compress_mode	数据压缩格式。静态属性，创建PIPE时设定，不可更改。 - 仅支持帧压缩HI_COMPRESS_MODE_FRAME或行压缩HI_COMPRESS_MODE_LINE，且开启帧压缩时，分辨率需满足：总像素个数不少于307200个。 - 当输入数据格式为HI_PIXEL_FORMAT_RGB_BAYER_10BPP、HI_PIXEL_FORMAT_RGB_BAYER_12BPP、HI_PIXEL_FORMAT_RGB_BAYER_14BPP时，线性模式、WDR帧模式和行模式都支持HI_COMPRESS_MODE_LINE、HI_COMPRESS_MODE_FRAME；当输入数据格式为HI_PIXEL_FORMAT_RGB_BAYER_8BPP时，不支持压缩；当输入数据格式为YUV时，不支持压缩。
bit_width	输入图像的bit位宽。仅当像素格式pixel_format为YUV像素格式时有效。静态属性，创建PIPE时设定，不可更改。
frame_rate_ctrl	帧率控制。
depth	PIPE队列深度，用于缓存VI PIPE的临时数据、RAW/YUV图输入和输出数据。静态属性，创建 PIPE 时设定，不可更改。 临时缓存实际所能使用的最大队列深度为： hi_vi_pipe_attr.depth - hi_vi_frame_dump_attr.depth 取值范围：[0, 16]。
reserved	预留参数，为保证后续版本兼容性，请务必使用memset结构体方式进行清零初始化，在代码中必须避免显式对reserved字段进行访问。

1.9.20.5.13 hi_vi_vpss_mode

说明

定义VI各个PIPE和VPSS各个组的工作模式。

定义

```
typedef struct {  
    hi_vi_vpss_mode_type mode[VI_MAX_PIPE_NUM];  
} hi_vi_vpss_mode;
```

成员

成员名称	描述
mode	VI各个PIPE和VPSS各个组的工作模式。 数组内每个元素对应每个VI PIPE，例如数组内index为0的元素对应PIPE 0。 <code>#define VI_MAX_PIPE_NUM 12</code>

1.9.20.5.14 hi_vi_vpss_mode_type

说明

定义VI PIPE和VPSS组的工作模式。

定义

```
typedef enum {  
    HI_VI_OFFLINE_VPSS_OFFLINE = 0,  
    HI_VI_OFFLINE_VPSS_ONLINE,  
    HI_VI_ONLINE_VPSS_OFFLINE,  
    HI_VI_ONLINE_VPSS_ONLINE,  
    HI_VI_PARALLEL_VPSS_OFFLINE,  
    HI_VI_PARALLEL_VPSS_PARALLEL,  
    HI_VI_VPSS_MODE_BUTT  
} hi_vi_vpss_mode_type;
```

成员

成员名称	描述
HI_VI_OFFLINE_VPSS_OFFLINE	VI离线，VPSS离线。 使用该模式时，如果需要通过视频流自动流转到VPSS继续处理，则需要调用 hi_mpi_sys_bind 接口设定VI、VPSS绑定关系，并完成VI PIPE以及被绑VPSS Group的初始化。
HI_VI_OFFLINE_VPSS_ONLINE	VI离线，VPSS在线。 当使用该模式时，所有VI PIPE都需要使用该模式。该模式下，VI PIPE编号与VPSS GROUP号一一对应，数据从VI PIPE流动到VPSS GROUP，不需要通过 hi_mpi_sys_bind 接口设定VI、VPSS绑定关系，绑定关系内部自动维护为：VPSS GROUP ID = VI PIPE ID + 256。 该模式下，VI PIPE以及内部自动绑定的VPSS Group必须都完成初始化，否则没法正常出图。
HI_VI_ONLINE_VPSS_OFFLINE	VI在线，VPSS离线。当前版本不支持。
HI_VI_ONLINE_VPSS_ONLINE	VI在线，VPSS在线。当前版本不支持。
HI_VI_PARALLEL_VPSS_OFFLINE	VI并行，VPSS离线。当前版本不支持。

成员名称	描述
HI_VI_PARALLEL_VPS S_PARALLEL	VI并行，VPSS并行。当前版本不支持。

1.9.20.5.15 hi_vi_pipe_bypass_mode

说明

定义VI PIPE的bypass模式。

定义

```
typedef enum {  
    HI_VI_PIPE_BYPASS_NONE,  
    HI_VI_PIPE_BYPASS_FE,  
    HI_VI_PIPE_BYPASS_BE,  
    HI_VI_PIPE_BYPASS_BUTT  
} hi_vi_pipe_bypass_mode;
```

成员

成员名称	描述
HI_VI_PIPE_BYPASS_N ONE	VI 的数据经过FE与BE处理。
HI_VI_PIPE_BYPASS_FE	VI 的数据不经过FE处理，只经过BE处理。当前版本不支持。
HI_VI_PIPE_BYPASS_BE	VI 的数据经过FE处理，不经过BE处理。
HI_VI_PIPE_BYPASS_B UTT	预留值。

1.9.20.5.16 hi_vi_frame_dump_attr

说明

定义VI PIPE dump属性。

定义

```
typedef struct {  
    hi_bool    enable;  
    hi_u32     depth;  
} hi_vi_frame_dump_attr;
```

成员

成员名称	描述
enable	是否开启dump。 - HI_FALSE：不开启。 - HI_TRUE：开启。
depth	Dump数据的队列深度，该值不允许超过对应PIPE属性里的depth值。 取值范围：[0, 8]。

1.9.20.5.17 hi_vi_pipe_frame_source

说明

定义视频输入设备的属性。

定义

```
typedef enum {  
    HI_VI_PIPE_FRAME_SOURCE_FE = 0,  
    HI_VI_PIPE_FRAME_SOURCE_USER,  
    HI_VI_PIPE_FRAME_SOURCE_BUTT  
} hi_vi_pipe_frame_source;
```

成员

成员名称	描述
HI_VI_PIPE_FRAME_SOURCE_FE	数据来自于前端FE，默认模式。
HI_VI_PIPE_FRAME_SOURCE_USER	数据来自用户从BE送进来的数据。
HI_VI_PIPE_FRAME_SOURCE_BUTT	预留值。

1.9.20.5.18 hi_vi_chn_attr

说明

定义VI通道属性。

定义

```
typedef struct {  
    hi_size size;  
    hi_pixel_format pixel_format;  
    hi_dynamic_range dynamic_range;  
    hi_video_format video_format;
```

```
hi_compress_mode compress_mode;
hi_u32 depth;
hi_frame_rate_ctrl frame_rate_ctrl;
hi_u32 reserved[10];
} hi_vi_chn_attr;
```

成员

成员名称	描述
size	目标图像大小。静态属性，设置通道时设定，不可更改。 - 必须配置，且大小需要在VI支持的范围，详细范围说明参考1.9.9.2 约束说明。 - 通道不支持缩放，通道size必须和pipe属性配置的size一致。
pixel_format	目标图像像素格式。静态属性，设置通道时设定，不可更改。 当未开启DIS或LDC时，支持的像素格式为： HI_PIXEL_FORMAT_YVU_SEMIPLANAR_422、 HI_PIXEL_FORMAT_YVU_SEMIPLANAR_420、 HI_PIXEL_FORMAT_YUV_400。 当开启DIS或LDC时，支持的像素格式为： HI_PIXEL_FORMAT_YVU_SEMIPLANAR_420、 HI_PIXEL_FORMAT_YUV_400。
dynamic_range	目标图像动态范围。静态属性，设置通道时设定，不可更改。
video_format	目标图像视频数据格式。当前版本仅支持 HI_VIDEO_FORMAT_LINEAR格式。
compress_mode	目标图像压缩格式。静态属性，设置通道时设定，不可更改。当前版本不支持。
depth	用户获取图像的队列深度。静态属性，设置通道时设定，不可更改。 取值范围：[0, 8]。
frame_rate_ctrl	帧率控制。
reserved	预留参数，为保证后续版本兼容性，请务必使用memset结构体方式进行清零初始化，在代码中必须避免显式对reserved字段进行访问。

说明

- 当前版本，VI CHN通道不支持缩放，所以请确保将chn属性中的size大小与pipe属性中的size大小保持一致。
- 请根据实际负载和抖动，设置合理的depth值，内部实际申请的内存块数量为：调用[hi_mpi_vi_enable_chn](#)接口开启通道时的hi_vi_pipe_attr属性中的depth - hi_vi_frame_dump_attr属性中的depth + chn属性中的depth值。所以建议用户在开启通道前，确保已完成pipe dump属性的相关配置。

1.9.20.5.19 hi_dis_config

说明

定义DIS的配置信息。

定义

```
typedef struct {  
    hi_dis_mode      mode;  
    hi_dis_motion_level motion_level;  
    hi_dis_pdt_type   pdt_type;  
    hi_u32            buf_num;  
    hi_u32            crop_ratio;  
    hi_u32            frame_rate;  
    hi_bool camera_steady;  
    hi_bool scale;  
} hi_dis_config;
```

成员

成员名称	描述
mode	DIS中使用不同自由度防抖算法，共有3种不同算法。
motion_level	Camera的运动级别。
pdt_type	使用DIS的产品形态，当前支持IPC、DV和无人机。
buf_num	DIS用于缓存图像的内存块数目，在DIS输出帧率偶尔出现丢帧时，可以增加缓存buf数。 取值范围：[5, 10]。
crop_ratio	DIS输出图像的裁剪比例。 取值范围：[50, 98]。
frame_rate	设置为VI输出的实际帧率。取值范围：(0, 60]。 - 当防抖模式为HI_DIS_MODE_4_DOF_GME/HI_DIS_MODE_6_DOF_GME时，低帧率会导致两帧间隔变大，两帧之间抖动幅度增加，同时，果冻效应（rolling shutter）也更加严重，预期可能会出现防抖效果下降，特别是在抖动幅度大的场景； - 当防抖模式为HI_DIS_MODE_GYRO时，除了防抖效果的影响和GME一致，会导致算法支持的陀螺仪的最高采样率下降。当前3.5KHz的规格是基于最小25fps设计，如果下降，对应的支持的陀螺仪的最大采样率也会下降。
camera_steady	镜头是否固定静止的。 取值范围： - HI_FALSE：不固定静止。 - HI_TRUE：固定静止。

成员名称	描述
scale	Crop后的输出图像是否进行放大。当前DIS提供Crop 后的输出图像是否进行放大操作选择。如果用户不想采用在DIS放大功能，可以选择在后端的VPSS进行放大操作。 取值范围： • HI_FALSE：不放大。 • HI_TRUE：放大。 scale为HI_FALSE的情况下，输出的图像宽高为输入图像宽高乘以crop_ratio设置的裁剪比例，且值为2对齐。

1.9.20.5.20 hi_dis_attr

说明

定义DIS属性。

定义

```
typedef struct {  
    hi_bool enable;  
    hi_bool gdc_bypass;  
    hi_u32 moving_subject_level;  
    hi_s32 rolling_shutter_coef;  
    hi_s32 timelag;  
    hi_u32 view_angle;  
    hi_u32 horizontal_limit;  
    hi_u32 vertical_limit;  
    hi_bool still_crop;  
    hi_u32 strength;  
} hi_dis_attr;
```

成员

成员名称	描述
enable	是否开启DIS功能。 • HI_FALSE：不开启； • HI_TRUE：开启。
gdc_bypass	DIS功能中是否开启bypass GDC的开关。 • HI_FALSE：不开启bypass GDC功能，即正常调用GDC做PMF矫正； • HI_TRUE：开启bypass GDC功能，即不调用GDC做PMF矫正。
moving_subject_level	用于判断物体是否是运动的级别，值越小，运动过程中越稳定，但更容易出现偏移情况。值较大时，防抖效果越弱，运动过程中背景抖动幅度大，但能够改善图像偏移现象。 取值范围：[0, 6]。

成员名称	描述
rolling_shutter_coef	校正果冻效应（rolling shutter）现象的校正参数。 取值范围：[0, 1000]。
timelag	帧起始时间和陀螺仪数据采集时间的时间差，单位为微秒。 取值范围：[-2000000, 2000000]。
view_angle	预留参数。
horizontal_limit	水平偏移限制。当大面积物体经过引起背景拖拽的水平偏移超过一定幅度时就不进行防抖。偏移幅度计算： $2047 * horizontal\_limit / 1000$ 。 取值范围：[0,1000]。
vertical_limit	垂直偏移限制。当大面积物体经过引起背景拖拽的垂直偏移超过一定幅度时就不进行防抖。偏移幅度计算： $2047 * vertical\_limit / 1000$ 。 取值范围：[0,1000]。
still_crop	关闭DIS防抖效果，但图像依旧按照所配置的比例裁剪。
strength	陀螺仪防抖的强度控制。仅对GYRO模式有效，对GME模式无效，默认强度为1024。 取值范围：[0,1024]。

1.9.20.5.21 hi_dis_mode

说明

定义DIS防抖算法模式。

定义

```
typedef enum {  
    HI_DIS_MODE_4_DOF_GME = 0,  
    HI_DIS_MODE_6_DOF_GME,  
    HI_DIS_MODE_GYRO,  
    HI_DIS_MODE_DOF_BUTT,  
} hi_dis_mode;
```

成员

成员名称	描述
HI_DIS_MODE_4_DOF_GME	四自由度GME算法，不使用陀螺仪。
HI_DIS_MODE_6_DOF_GME	六自由度GME算法，不使用陀螺仪。
HI_DIS_MODE_GYRO	陀螺仪算法，不使用GME。

成员名称	描述
HI_DIS_MODE_DOF_BUTT	预留值。

1.9.20.5.22 hi_dis_motion_level

说明

定义镜头的运动级别。

定义

```
typedef enum {  
    HI_DIS_MOTION_LEVEL_LOW = 0,  
    HI_DIS_MOTION_LEVEL_NORMAL,  
    HI_DIS_MOTION_LEVEL_HIGH,  
    HI_DIS_MOTION_LEVEL_BUTT  
} hi_dis_motion_level;
```

成员

成员名称	描述
HI_DIS_MOTION_LEVEL_LOW	低级别运动，镜头小幅度运动。当前版本不支持。
HI_DIS_MOTION_LEVEL_NORMAL	正常级别运动，镜头正常幅度运动。
HI_DIS_MOTION_LEVEL_HIGH	高级别运动，镜头大幅度运动。
HI_DIS_MOTION_LEVEL_BUTT	预留值。

1.9.20.5.23 hi_dis_pdt_type

说明

定义使用DIS的产品形态。

定义

```
typedef enum {  
    HI_DIS_PDT_TYPE_IPC = 0,  
    HI_DIS_PDT_TYPE_DV,  
    HI_DIS_PDT_TYPE_DRONE,  
    HI_DIS_PDT_TYPE_BUTT  
} hi_dis_pdt_type;
```

成员

成员名称	描述
HI_DIS_PDT_TYPE_IPC	IPC产品形态。
HI_DIS_PDT_TYPE_DV	DV产品形态。
HI_DIS_PDT_TYPE_DRONE	无人机产品形态。
HI_DIS_PDT_TYPE_BUTT	预留值。

1.9.20.5.24 hi_dis_param

说明

定义DIS可选参数。

定义

```
typedef struct {  
    hi_u32 large_motion_stable_coef;  
    hi_u32 low_freq_motion_preserve;  
    hi_u32 low_freq_motion_freq;  
} hi_dis_param;
```

成员

成员名称	描述
large_motion_stable_coef	大幅度运动的防抖衰减参数。 取值范围：[0, 100]。
low_freq_motion_preserve	低频运动的保留程度。 取值范围：[0, 100]。
low_freq_motion_freq	低频运动的截止频率。 取值范围：[0, 100]。

1.9.20.5.25 hi_vi_ldc_attr

说明

定义镜头畸变校正属性结构体。

定义

```
typedef struct {  
    hi_bool enable;  
    hi_ldc_version ldc_version;
```

```
union {  
    hi_ldc_v1_attr ldc_v1_attr;  
    hi_ldc_v2_attr ldc_v2_attr;  
}  
} hi_vi_ldc_attr;
```

成员

成员名称	描述
enable	控制LDC是否开启。 - HI_FALSE：不开启； - HI_TRUE：开启。
ldc_version	LDC版本。
ldc_v1_attr	镜头畸变校正属性，ldc_version为HI_LDC_V1时才生效。
ldc_v2_attr	镜头畸变校正属性，ldc_version为HI_LDC_V2时才生效，必须跟Gyro DIS一起使用。

1.9.20.5.26 hi_ldc_v1_attr

说明

定义HI_LDC_V1 版本的 镜头畸变校正属性 。

定义

```
typedef struct {  
    hi_bool aspect;  
    hi_s32 x_ratio;  
    hi_s32 y_ratio;  
    hi_s32 xy_ratio;  
    hi_s32 center_x_offset;  
    hi_s32 center_y_offset;  
    hi_s32 distortion_ratio;  
} hi_ldc_v1_attr;
```

成员

成员名称	描述
aspect	畸变校正类型。 取值范围： - HI_FALSE：不保持幅形比。 - HI_TRUE：保持幅形比。
x_ratio	水平视场角大小，取值范围：[0, 100]，aspect=HI_FALSE时本参数有效。 其值为水平视场角介于最小视场角和最大视场角之间的一个比例。

成员名称	描述
y_ratio	垂直视场角大小，取值范围：[0, 100]，aspect=HI_FALSE时本参数有效； 其值为垂直视场角介于最小视场角和最大视场角之间的一个比例。
xy_ratio	整体视场角大小，包括水平视场角和垂直视场角，取值范围：[0, 100]，aspect=HI_FALSE时本参数有效； 其值为整体视场角介于最小视场角和最大视场角之间的一个比例。
center_x_offset	畸变中心点相对图像中心点水平偏移。 取值范围：[-511, 511]。
center_y_offset	畸变中心点相对图像中心点垂直偏移。 取值范围：[-511, 511]。
distortion_ratio	畸变程度，有效范围为[-300, 500]，负数为枕型，正数为桶型。

1.9.20.5.27 hi_ldc_v2_attr

说明

定义HI_LDC_V2版本的镜头畸变校正属性。

定义

```
typedef struct {  
    hi_s32 focal_len_x;  
    hi_s32 focal_len_y;  
    hi_s32 coor_shift_x;  
    hi_s32 coor_shift_y;  
    hi_s32 src_cali_ratio[HI_SRC_LENS_COEF_SEG][HI_SRC_LENS_COEF_NUM];  
    hi_s32 src_jun_pt;  
    hi_s32 dst_cali_ratio[HI_DST_LENS_COEF_SEG][HI_DST_LENS_COEF_NUM];  
    hi_s32 dst_jun_pt[HI_DST_LENS_COEF_SEG_POINT];  
    hi_s32 max_du;  
} hi_ldc_v2_attr;
```

成员

成员名称	描述
focal_len_x	水平方向镜头有效焦距。 取值范围：[6400, 117341700]。
focal_len_y	垂直方向镜头有效焦距。 取值范围：[6400, 117341700]。

成员名称	描述
coor_shift_x	光心X坐标。 取值范围：[35* Width, 65* Width], Width为图像宽。
coor_shift_y	光心Y坐标。 取值范围：[35* Height, 65* Height], Height为图像高。
src_cali_ratio	镜头畸变系数，最多支持分2 段。其中[0][0]固定为1*10^5，其他成员范围均为[-16*10^5, 16*10^5]。
src_jun_pt	镜头畸变系数。 取值范围：[0, 32*10^5]。
dst_cali_ratio	镜头畸变系数，最多支持分3段。每个成员范围均为[-16*10^5, 16*10^5]。
dst_jun_pt	镜头畸变系数。 取值范围：[0,16*10^5]。
max_du	镜头畸变系数。 取值范围：[0,1<<20]。

1.9.20.5.28 hi_ldc_version

说明

定义LDC 版本 。

定义

```
typedef enum {  
    HI_LDC_V1 = 1,  
    HI_LDC_V2 = 2,  
    HI_LDC_VERSION_BUTT  
} hi_ldc_version
```

成员

成员名称	描述
HI_LDC_V1	HI_LDC_V1版本。
HI_LDC_V2	HI_LDC_V2版本。 该版本的畸变矫正必须配合DIS陀螺仪算法使用。
HI_LDC_VERSION_BUTT	预留值。

1.9.20.5.29 hi_vi_low_delay_info

说明

定义低延时属性。

定义

```
typedef struct {  
    hi_bool enable;  
    hi_u32 line_cnt;  
} hi_vi_low_delay_info;
```

成员

成员名称	描述
enable	是否开启低延时开关。 取值范围： - HI_FALSE：不开启。 - HI_TRUE：开启。
line_cnt	低延时输出行号，最小值为32，最大值不超过图像高度，不建议设置行号太接近图像高度。

1.9.20.6 Region 区域管理

1.9.20.6.1 HI_RGN_CLUT_NUM

说明

定义RGB 颜色查表个数。

定义

```
#define HI_RGN_CLUT_NUM 16
```

1.9.20.6.2 HI_RGN_HANDLE_MAX

说明

定义区域的最大句柄个数。

定义

```
#define HI_RGN_HANDLE_MAX 1024
```

1.9.20.6.3 hi_rgn_handle

说明

定义区域句柄。

定义

```
typedef hi_u32 hi_rgn_handle
```

1.9.20.6.4 hi_phys_addr_t

说明

定义物理地址类型。

定义

```
typedef hi_u64 hi_phys_addr_t;
```

1.9.20.6.5 hi_rgn_type

说明

定义区域类型。

定义

```
typedef enum {  
    HI_RGN_OVERLAY = 0,  
    HI_RGN_COVER,  
    HI_RGN_OVERLAYEX,  
    HI_RGN_COVEREX,  
    HI_RGN_LINE,  
    HI_RGN_MOSAIC,  
    HI_RGN_MOSAICEX,  
    HI_RGN_CORNER_RECTEX,  
} hi_rgn_type;
```

成员

成员名称	描述
HI_RGN_OVERLAY	视频叠加区域。
HI_RGN_COVER	视频遮挡区域。
HI_RGN_OVERLAYEX	扩展视频叠加区域。
HI_RGN_COVEREX	扩展视频遮挡区域。
HI_RGN_LINE	扩展线形遮挡区域。
HI_RGN_MOSAIC	MOSAIC视频区域。
HI_RGN_MOSAICEX	扩展MOSAICEX视频区域。
HI_RGN_CORNER_RECTEX	角框区域。

1.9.20.6.6 hi_rgn_overlayex_attr

说明

定义扩展叠加区域属性结构体。

定义

```
typedef struct {  
    hi_pixel_format pixel_format;  
    hi_u32          bg_color;  
    hi_size         size;  
    hi_u32          canvas_num;  
    hi_u32          clut[HI_RGN_CLUT_NUM];  
} hi_rgn_overlayex_attr;
```

成员

成员名称	描述
pixel_format	像素格式 取值范围： HI_PIXEL_FORMAT_ARGB_1555 = 33; HI_PIXEL_FORMAT_ARGB_4444 = 25; HI_PIXEL_FORMAT_ARGB_8888 = 14; HI_PIXEL_FORMAT_ARGB_CLUT2 = 41; HI_PIXEL_FORMAT_ARGB_CLUT4 = 42。
bg_color	区域背景色。 当像素格式为HI_PIXEL_FORMAT_ARGB_1555 = 33，取值范围： [0, 0xffff]。 当像素格式为HI_PIXEL_FORMAT_ARGB_4444 = 25，取值范围： [0, 0xffff]。 当像素格式为HI_PIXEL_FORMAT_ARGB_8888 = 14，取值范围： [0, 0xffffffff]。 当像素格式为HI_PIXEL_FORMAT_ARGB_CLUT2 = 41，取值范围： [0, 0x3]。 当像素格式为HI_PIXEL_FORMAT_ARGB_CLUT4 = 42，取值范围： [0,0xf]。
size	区域的高宽 取值范围： 宽度： [2, 8192]，要求以2对齐。 高度： [2, 8192]，要求以2对齐。
canvas_num	区域的内存数量 取值范围： [1, 2]。

成员名称	描述
clut	定义RGB颜色值。HI_RGN_CLUT_NUM用于定义RGB颜色查表个数，当前表示16。 颜色值取值范围：[0x0, 0xffffffff]。 像素为HI_PIXEL_FORMAT_ARGB_CLUT2 = 41或者HI_PIXEL_FORMAT_ARGB_CLUT4 = 42时，才进行参数范围检查。

1.9.20.6.7 hi_rgn_overlayex_chn_attr

说明

定义扩展叠加区域的通道显示属性。

定义

```
typedef struct {  
    hi_point    point;  
    hi_u32      fg_alpha;  
    hi_u32      bg_alpha;  
    hi_u32      layer;  
} hi_rgn_overlayex_chn_attr;
```

成员

成员名称	描述
point	区域位置。动态属性。 区域位置坐标取值范围： 水平位置X：[0, 16382]，要求以2对齐。 垂直位置Y：[0, 16382]，要求以2对齐。
fg_alpha	Alpha位为1的像素点的透明度。也称前景Alpha。动态属性。 取值范围：[0, 255]。取值越小，越透明。 仅当被叠加图片的格式为HI_PIXEL_FORMAT_ARGB_1555格式时，需设置该参数。
bg_alpha	Alpha位为0的像素点的透明度。也称背景Alpha。动态属性。 取值范围：[0, 255]。取值越小，越透明。 仅当被叠加图片的格式为HI_PIXEL_FORMAT_ARGB_1555格式时，需设置该参数。
layer	区域层次。动态属性。 取值范围与各模块最多能支持的OVERLAYEX个数相关，从0开始到通道支持区域最大个数减1。值越大，层次越高。

1.9.20.6.8 hi_rgn_cover_chn_attr

说明

定义遮挡区域的通道显示属性。

定义

```
typedef struct {  
    hi_cover    cover;  
    hi_u32      layer;  
    hi_coord    coord;  
} hi_rgn_cover_chn_attr;
```

成员

成员名称	描述
cover	COVER属性。动态属性。 其中rect表示区域坐标位置和宽高 绝对坐标（ coord为0的情况 ）： - 区域位置坐标取值范围： 水平位置X： [-16384, 16382]，要求以2对齐。 垂直位置Y： [-16384, 16382]，要求以2对齐。 - 宽高取值范围： 宽度： [2, 16384]，要求以2对齐。 高度： [2, 16384]，要求以2对齐。
layer	区域层次。动态属性。 取值范围与各模块最多能支持的COVER个数相关，从0开始到通道支持区域最大个数减1。值越大，层次越高。
coord	区域坐标类型，当前只支持配置为0，表示绝对坐标。动态属性。

1.9.20.6.9 hi_rgn_mosaic_chn_attr

说明

定义mosaic区域属性结构体。

定义

```
typedef struct {  
    hi_rect    rect;  
    hi_blk_size blk_size;  
    hi_u32      layer;  
} hi_rgn_mosaic_chn_attr;
```

成员

成员名称	描述
rect	区域位置、宽高。动态属性。 - 区域位置坐标取值范围： 水平位置X: [-16384, 16384] 垂直位置Y: [-16384, 16384] - 宽高取值范围： 宽度: [8, 16384] 高度: [8, 16384]
blk_size	区域块大小。动态属性。 只支持HI_BLK_SIZE_8、HI_BLK_SIZE_16、HI_BLK_SIZE_32。
layer	MOSAIC区域层次。动态属性。 取值范围与各模块最多能支持的MOSAIC个数相关，从0开始到通道支持区域最大个数减1。值越大，层次越高。

1.9.20.6.10 hi_rgn_type_attr

说明

定义区域属性联合体。

定义

```
typedef union {  
    hi_rgn_overlayex_attr    overlayex;  
} hi_rgn_type_attr;
```

成员

成员名称	描述
overlayex	扩展叠加区域属性。

1.9.20.6.11 hi_rgn_type_chn_attr

说明

定义区域通道显示属性联合体。

定义

```
typedef union {  
    hi_rgn_cover_chn_attr    cover_chn;  
    hi_rgn_overlayex_chn_attr    overlayex_chn;
```

```
    hi_rgn_mosaic_chn_attr mosaic_chn;  
} hi_rgn_type_chn_attr;
```

成员

成员名称	描述
cover_chn	遮挡区域通道显示属性。
overlayex_chn	扩展叠加区域通道显示属性。
mosaic_chn	MOSAIC显示属性。

1.9.20.6.12 hi_rgn_attr

说明

定义区域属性结构体。

定义

```
typedef struct {  
    hi_rgn_type type;  
    hi_rgn_type_attr attr;  
} hi_rgn_attr;
```

成员

成员名称	描述
type	区域类型。
attr	区域属性。

1.9.20.6.13 hi_rgn_chn_attr

说明

定义区域通道显示属性结构体。

定义

```
typedef struct {  
    hi_bool is_show;  
    hi_rgn_type type;  
    hi_rgn_type_chn_attr attr;  
} hi_rgn_chn_attr;
```

成员

成员名称	描述
is_show	区域是否显示。动态属性。 取值范围： - HI_TRUE：显示 - HI_FALSE：不显示
type	区域类型。静态属性。
attr	区域通道显示属性。

1.9.20.6.14 hi_rgn_canvas_info

说明

定义画布信息结构体 。

定义

```
typedef struct {  
    hi_phys_addr_t    phys_addr;  
    hi_size           size;  
    hi_u32            stride;  
    hi_pixel_format    pixel_format;  
    hi_void ATTRIBUTE *virt_addr;  
} hi_rgn_canvas_info;
```

成员

成员名称	描述
phys_addr	画布物理地址。
size	画布尺寸。
stride	画布的stride。
pixel_format	画布的像素格式。
virt_addr	画布虚拟地址。 用户的位图数据拷贝到该地址指向的内存中，用于更新画布内容。

1.9.20.7 VPSS 视频处理

1.9.20.7.1 基本数据类型说明

```
typedef hi_s32 hi_vpss_grp;  
typedef hi_s32 hi_vpss_chn;
```

1.9.20.7.2 hi_vpss_crop_info

说明

定义裁剪功能所需信息。

定义

```
typedef struct {  
    hi_bool    enable;  
    hi_coord   crop_mode;  
    hi_rect    crop_rect;  
}hi_vpss_crop_info;
```

成员

成员名称	描述
enable	是否开启裁剪功能。 取值范围： - HI_FALSE：不开启。 - HI_TRUE：开启。
crop_mode	裁剪区域起始点坐标模式。
crop_rect	裁剪的矩形区域。 - 将crop_mode参数设置为相对坐标时，裁剪区域坐标取值范围为 [0, 999]，裁剪区域宽高取值范围为[1, 1000] 。 - 裁剪区域的尺寸不能小于VPSS支持的最小输入分辨率，不能超过VPSS支持的最大输入分辨率；如果裁剪宽度大于输入图像宽度，则裁剪输出宽度调整为输入图像宽度；如果裁剪高度大于输入图像高度，则裁剪输出高度调整为输入图像高度；裁剪区域起始点不支持负坐标，裁剪区域右边界不能超出VPSS支持的最大输入宽度，裁剪区域下边界不能超出VPSS支持的最大输入高度。 - 如果裁剪区域超出图像范围，裁剪坐标向原点方向移动，优先保证裁剪出的宽高与所设置的参数相同。

 说明

- 相对坐标模式下：
 - 横坐标计算公式： $X = \text{crop\_rect.x} * \text{实际图像宽度} / 1000$ ，计算完成后会进行取整操作和对齐操作。公式同样适用于纵坐标计算。
 - 区域宽度计算公式： $\text{Width} = \text{crop\_rect.width} * \text{实际图像宽度} / 1000$ ，计算完成后会进行取整操作和对齐操作。公式同样适用于区域高度计算。
 - 如果经过计算后，裁剪区域宽高小于VPSS支持的最小分辨率，则裁剪输出宽高调整为输入图像宽高，接口返回成功。
- 绝对坐标模式下：
 - 如果裁剪宽高小于VPSS支持的最小输入分辨率，则crop设置不生效，接口返回失败。
 - 如果裁剪宽高大于VPSS支持的最大输入分辨率，则crop设置不生效，接口返回失败。
 - 裁剪区域坐标和裁剪宽高之和超过VPSS支持的最大输入分辨率，则crop设置不生效，接口返回失败。

1.9.20.7.3 hi_vpss_nr_type

说明

NR (Noise Reduction) 类型枚举。

定义

```
typedef enum {  
    HI_VPSS_NR_TYPE_VIDEO_NORM = 0,  
    HI_VPSS_NR_TYPE_BUTT  
} hi_vpss_nr_type;
```

成员

成员名称	描述
HI_VPSS_NR_TYPE_VIDEO_NORM	视频NR类型。
HI_VPSS_NR_TYPE_BUTT	预留值。

1.9.20.7.4 hi_vpss_nr_motion_mode

说明

NR的运动矢量模式。

定义

```
typedef enum {  
    HI_VPSS_NR_MOTION_MODE_NORM = 0,  
    HI_VPSS_NR_MOTION_BUTT  
} hi_vpss_nr_motion_mode;
```


成员

成员名称	描述
HI_VPSS_NR_MOTION_MODE_NORM	普通模式。
HI_VPSS_NR_MOTION_BUTT	预留值。

1.9.20.7.5 hi_vpss_nr_attr

说明

定义VPSS GRP的3DNR属性。开3DNR时，压缩格式只支持帧压缩和非压，而且压缩方式不能动态改变。

定义

```
typedef struct {  
    hi_vpss_nr_type    nr_type;  
    hi_compress_mode    compress_mode;  
    hi_vpss_nr_motion_mode    nr_motion_mode;  
} hi_vpss_nr_attr;
```

成员

成员名称	描述
nr_type	NR类型。
compress_mode	重构帧压缩类型，仅支持帧压缩模式 HI_COMPRESS_MODE_FRAME，且开启帧压缩模式时， 宽度必须大于等于640，高度必须大于等于480并小于等于4096。
nr_motion_mode	NR运动矢量模式。

1.9.20.7.6 hi_vpss_grp_attr

说明

定义VPSS Group属性。

定义

```
typedef struct {  
    hi_u32    max_width;  
    hi_u32    max_height;  
    hi_pixel_format    pixel_format;  
    hi_dynamic_range    dynamic_range;  
    hi_frame_rate_ctrl    frame_rate;  
    hi_bool    nr_en;  
    hi_vpss_nr_attr    nr_attr;
```

```
    hi_u32 reserved[20];  
} hi_vpss_grp_attr;
```

成员

成员名称	描述
max_width	最大输入图像宽度。 静态属性，创建Group时设定，不可更改。
max_height	最大输入图像高度。 静态属性，创建Group时设定，不可更改。
pixel_format	输入图像像素格式。 静态属性，创建Group时设定，不可更改。 - VPSS Group ID在[0,256)范围内，支持以下格式： HI_PIXEL_FORMAT_YVU_SEMIPLANAR_420、 HI_PIXEL_FORMAT_YUV_SEMIPLANAR_420。 - VPSS Group ID在[256,264)范围内，支持以下格式： HI_PIXEL_FORMAT_YVU_SEMIPLANAR_422、 HI_PIXEL_FORMAT_YUV_SEMIPLANAR_422、 HI_PIXEL_FORMAT_YVU_SEMIPLANAR_420、 HI_PIXEL_FORMAT_YUV_SEMIPLANAR_420、 HI_PIXEL_FORMAT_YUV_400。
dynamic_range	保留参数，必须赋值为0。
frame_rate	组帧率。
nr_en	是否开启NR。 取值范围： - HI_FALSE：不开启。 - HI_TRUE：开启。
nr_attr	NR属性。
reserved	预留参数，为保证后续版本兼容性，请务必使用memset结构体方式进行清零初始化，在代码中必须避免显式对reserved字段进行访问。

1.9.20.7.7 hi_vpss_nr_version

说明

NR版本枚举。

定义

```
typedef enum {  
    HI_VPSS_NR_V3 = 3,  
    HI_VPSS_NR_BUTT  
} hi_vpss_nr_version;
```

成员

成员名称	描述
HI_VPSS_NR_V3	3DNR参数版本3。

1.9.20.7.8 hi_vpss_nrx_v3

说明

定义3DNR X接口版本V3的参数。

定义

```
typedef struct {  
    hi_v200_vpss_iey iey[5];  
    hi_v200_vpss_sfy sfy[5];  
    hi_v200_vpss_mdy mdy[2];  
    hi_v200_vpss_tfy tfy[3];  
    hi_v200_vpss_nr_c nr_c;  
} hi_vpss_nrx_v3;
```

成员

成员名称	描述
iey	增强模块参数。
sfy	空域滤波参数。
mdy	运动检测参数。
tfy	时域滤波参数。
nrc	色度滤波参数。

1.9.20.7.9 hi_v200_vpss_iey

说明

定义3DNR增强模块参数。

定义

```
typedef struct {  
    hi_u8 ies0, ies1, ies2, ies3;  
    hi_u16 iedz : 10, ie_en : 1, reserved : 5;  
} hi_v200_vpss_iey;
```

成员

成员名称	描述
ies0、ies1、ies2、ies3	边缘的绝对增强强度，0~3对应不同频段。 取值范围：[0, 255]。
iedz	噪声控制阈值。 取值范围：[0, 999]。
ie_en	ie模块开关。
reserved	预留参数，为保证后续版本兼容性，请务必使用memset结构体方式进行清零初始化，在代码中必须避免显式对reserved字段进行访问。

1.9.20.7.10 hi_v200_vpss_sfy

说明

定义3DNR空域滤波参数。

定义

```
typedef struct {  
    hi_u8 spn6 : 3, sfr : 5;  
    hi_u8 sbn6 : 3, pbr6 : 5;  
    hi_u16 srt0 : 5, srt1 : 5, j_mode : 3, de_idx : 3;  
    hi_u8 sfr6[4], sbr6[2], de_rate;  
    hi_u8 sfs1, sft1, sbr1;  
    hi_u8 sfs2, sft2, sbr2;  
    hi_u8 sfs4, sft4, sbr4;  
    hi_u16 sth1 : 9, sfn1 : 3, sfn0 : 3, nr_y_en : 1;  
    hi_u16 sth_d1 : 9, reserved : 7;  
    hi_u16 sth2 : 9, sfn2 : 3, k_mode : 3, reserved_1 : 1;  
    hi_u16 sth_d2 : 9, reserved_2 : 7;  
    hi_u16 sbs_k[32], sds_k[32];  
} hi_v200_vpss_sfy;
```

成员

成员名称	描述
j_mode	空域混合模式。 取值范围：[0,4]。
spn6、sbn6	混合模式滤波器选择。 取值范围：[0, 5]。
pbr6	表示spn6和sbn6滤波结果的混合比例，当混合模式j_mode为1时生效。 取值范围 [0,15]。

成员名称	描述
sfr6[4]	表示 由sbn6选择的滤波器产生的结果和spn6融合后的相对强度。 取值范围 [0, 31]。
de_idx	预留参数，当前只允许取默认值4。
de_rate	预留参数，当前只允许取默认值0。
srt0, srt1	预留参数，sfy[0]和sfy[1]下只允许取默认值16，sfy[2]~sfy[4]下只允许取0。
sfr	纯空域滤波器在SF _i 或者SF _k 模式下空域滤波结果通过4种滤波器审查后所能发挥的相对强度。 取值范围：[0,31]。
sfs1、sfs2、sfs4	表示1~4号滤波器强度（3和4号滤波器强度一样）。 取值范围：[0,255]。
sft1、sft2、sft4	表示1~4号滤波器附加强度。 取值范围：[0,255]。
sbr1、sbr2、sbr4、sbr6[2]	表示1~4、6号滤波器的滤波的不对称强度。 sbr1、sbr2、sbr4取值范围：[0,255]。 sbr6取值范围：[0,15]。
sth1、sth2、sth_d1、sth_d2	保边阈值上限和下限。值越小，越多的边缘被保留，噪声也会越大；值越大，保留的边缘越少，只有很强的边缘被保留住。 取值范围：[0,511]。
sfn0、sfn1、sfn2	对应 sth1，sth2不同图像特性选择不同滤波器的类型（编号）。 取值范围：[0,6]。
k_mode	选择根据绝对亮度调整去噪强度的模式。第2级和第3级有效。 取值范围：[0,3]。
nr_y_en	每一级去噪的使能开关。 0：关闭； 1：打开。
sbs_k[32]、sds_k[32]	根据画面的绝对亮度阈值划分的去相对暗区的去噪强度表。第2级和第3级有效。 取值范围 [0, 8191]。
reserved、reserved_1、reserved_2	预留参数，为保证后续版本兼容性，请务必使用memset结构体方式进行清零初始化，在代码中必须避免显式对reserved字段进行访问。

1.9.20.7.11 hi_v200_vpss_mdy

说明

定义3DNR运动检测参数。

定义

```
typedef struct {
    hi_u16 madz0 : 9, mai00 : 2, mai01 : 2, mai02 : 2, reserved : 1;
    hi_u16 madz1 : 9, mai10 : 2, mai11 : 2, mai12 : 2, reserved_1 : 1;
    hi_u8 mabr0, mabr1;
    hi_u16 math0 : 10, mate0 : 4, matw : 2;
    hi_u16 math_d0 : 10, reserved_2 : 6;
    hi_u16 math1 : 10, reserved_3 : 6;
    hi_u16 math_d1 : 10, reserved_4 : 6;
    hi_u8 masw : 4, mate1 : 4;
    hi_u8 mabw0 : 4, mabw1 : 4;
    hi_u16 adv_math : 1, adv_th : 12, reserved_5 : 3;
} hi_v200_vpss_mdy;
```

成员

成员名称	描述
mai00、mai01、mai02	选择采用哪种时域滤波器和空域的滤波器进行混合（通路0）。 取值范围 [0,3]。
mai10、mai11、mai12	选择采用哪种时域滤波器和空域的滤波器进行混合（通路1）。 取值范围 [0,3]。
madz0	控制根据图像的方差信息（madz0表示方差阈值的上限），在mai02中混入mai01的结果。 取值范围[0,511]。
madz1	控制根据图像的方差信息（madz1表示方差阈值的上限），在mai12中混入mai11的结果。 取值范围[0,511]。
mabr0	控制根据图像的亮度信息（mabr0表示亮度阈值），在mai02中混入mai01的结果。 取值范围[0, 255]。
mabr1	控制根据图像的亮度信息（mabr1表示亮度阈值），在mai12中混入mai11的结果。 取值范围[0, 255]。
math0、math_d0	动静判决阈值上下限（通路0），math0表示上限、math_d0表示下限，math0、math_d0的取值范围为[0, 999]，且要求math0必须大于math_d0，否则math0会默认与math_d0相等。

成员名称	描述
math1、math_d1	动静判决阈值上下限（通路1），math1表示上限、math_d1表示下限，math1、math_d1的取值范围为[0, 999]，且要求math1必须大于math_d1，否则math1会默认与math_d1相等。
mate0、mate1	表示通路0，1中平坦区域运动检测指数。 取值范围[0, 8]。
matw	表示时域滤波防运动拖尾指数，该值越大，运动拖尾收敛越快，反之，该值越小，运动拖尾收敛越慢，一般不建议调试，建议设置为默认值2。 取值范围[0, 3]。
masw	表示时域滤波防雨点指数，该值越大，有助于降低雨点噪声出现的概率，一般不建议调试，设置为默认值12。 取值范围[0, 15]。
mabw0、mabw1	表示通路0，1中运动检测内容窗口大小的选择。 取值范围[0, 9]。
adv_math	增强型MATH模式开关。0：关闭，1：打开。
adv_th	控制增强型MATH的效果。取值范围[0, 999]。
reserved、reserved_1、reserved_2、reserved_3、reserved_4、reserved_5	预留参数，为保证后续版本兼容性，请务必使用memset结构体方式进行清零初始化，在代码中必须避免显式对reserved字段进行访问。

1.9.20.7.12 hi_v200_vpss_tfy

说明

定义3DNR时域滤波参数。

定义

```
typedef struct {
    hi_u16 tfs0 : 4, tdz0 : 10, tdx0 : 2;
    hi_u16 tfs1 : 4, tdz1 : 10, tdx1 : 2;
    hi_u16 sdz0 : 10, str0 : 5, dz_mode0 : 1;
    hi_u16 sdz1 : 10, str1 : 5, dz_mode1 : 1;
    hi_u8 tss0 : 4, tsi0 : 4, tfr0[6];
    hi_u8 tss1 : 4, tsi1 : 4, tfr1[6];
    hi_u8 tfrs : 4, ted : 2, ref : 1, reserved : 1;
} hi_v200_vpss_tfy;
```

成员

成员名称	描述
tfs0、tfs1	通路0，1中时域滤波绝对强度。
dz_mode0、dz_mode1	通路0，1中tdz的模式选择。 取值范围[0,1]
tdz0、tdz1	当dz_mode0， dz_mode1为0时，保护运动区域的纹理不受时域滤波影响，将tdz调大时，运动区域的纹理可以得到保护，同时也会带来时域滤波强度的削弱；当dz_mode0， dz_mode1为1时，增加运动区域的时域滤波强度，将tdz调大时，运动区域时域去噪能力加强。 取值范围[0, 999]。
tdx0、tdx1	该参数建议使用默认值为2，不推荐调试。 取值范围：[0, 3]。
str0、str1	通路0，1中滤波器作用后叠加在结果的比例，值越大比例越高。 取值范围：[0, 31]。
tfr0[6]、tfr1[6]	通路0，1中静止区域时域滤波的相对强度。 取值范围[0, 31]。
tss0、tss1	通路0，1中时域静止区域混入空域的比例。 取值范围[0, 15]。
tsi0、tsi1	通路0，1中时混入空域滤波器的选择。 取值范围[0, 1]
ref	参考帧开关。 0：关闭 1：打开
ted	仅在时域生效时有效，用于控制运动后新内容的方法，0表示关闭，不做处理，1、2表示使用空域方法处理，3表示使用时域方法处理。 取值范围 [0,3]。
tfrs	tfr强度控制模式。 取值范围 [0, 15]。
sdz0、sdsz1	通路0，1中约束滤波器作用强度，值越小，作用强度越低。 取值范围：[0, 999]。
reserved	预留参数，为保证后续版本兼容性，请务必使用memset结构体方式进行清零初始化，在代码中必须避免显式对reserved字段进行访问。

1.9.20.7.13 hi_v200_vpss_nr_c

说明

定义3DNR视频色度滤波参数。

定义

```
typedef struct {  
    hi_u8 sfc, tfc : 6, reserved : 2;  
    hi_u8 trc, tpc : 6, reserved_1 : 2;  
    hi_u8 mode : 1, reserved_2 : 7;  
    hi_u8 presfc : 6, reserved_3 : 2;  
} hi_v200_vpss_nr_c;
```

成员

成员名称	描述
sfc	色度空域相对强度参数。 取值范围[0, 255]。
tpc	时域色噪滤波类型。 取值范围[0, 32]。
tfc	色度时域相对强度参数。 取值范围[0, 32]。
trc	用于抑制运动区域的色彩侵染现象。 取值范围[0, 255]。
mode	色度滤波模式。 取值范围[0,1]
presfc	色度空域预处理滤波器的强度。 取值范围[0,32]。
reserved、reserved_1、 reserved_2、reserved_3	预留参数，为保证后续版本兼容性，请务必使用 memset结构体方式进行清零初始化，在代码中必须避免 显式对reserved字段进行访问。

1.9.20.7.14 hi_vpss_nrx_param_manual_v3

说明

定义3DNR X接口版本V3的手动参数。

定义

```
typedef struct {  
    hi_vpss_nrx_v3 nrx_param;  
} hi_vpss_nrx_param_manual_v3;
```

成员

成员名称	描述
nrx_param	3DNR接口的标准参数。

1.9.20.7.15 hi_vpss_nrx_param_auto_v3

说明

定义3DNR X接口版本V3的自动参数。

定义

```
typedef struct {  
    hi_u32 param_num;  
    hi_u32 *iso;  
    hi_vpss_nrx_v3 *nrx_param;  
} hi_vpss_nrx_param_auto_v3;
```

成员

成员名称	描述
param_num	NR自动模式配置标准参数个数。 取值范围：[1, 16]。 约束说明：自动模式下的参数需要申请内存空间，自动参数的组数为param_num，需要为iso申请不小于param_num * sizeof(hi_u32)长度的内存，需要为nrx_param申请不小于param_num * sizeof(hi_vpss_nrx_v3)长度的内存。
iso	NR自动模式ISO数组指针。 ISO值的取值范围：[100, 3276800]，且数组中的ISO值是递增的。
nrx_param	NR 自动模式标准参数数组指针。

1.9.20.7.16 hi_vpss_nrx_param_v3

说明

定义3DNR X接口版本V3的属性。

定义

```
typedef struct {  
    hi_op_mode opt_mode;  
    hi_vpss_nrx_param_manual_v3 nrx_manual;  
    hi_vpss_nrx_param_auto_v3 nrx_auto;  
} hi_vpss_nrx_param_v3;
```

成员

成员名称	描述
opt_mode	NR选择模式。
nrx_manual	NR手动模式的标准参数。
nrx_auto	NR自动模式的标准参数。

1.9.20.7.17 hi_vpss_grp_nrx_param

说明

定义3DNR标准参数属性。

定义

```
typedef struct {  
    hi_vpss_nr_version nr_version;  
    union {  
        hi_vpss_nrx_param_v3 nrx_param_v3;  
    };  
} hi_vpss_grp_nrx_param;
```

成员

成员名称	描述
nr_version	NR接口版本枚举。
nrx_param_v3	当前芯片NR接口标准参数。

1.9.20.7.18 hi_vpss_grp_param

说明

定义VPSS的3DNR参数。

定义

```
typedef struct{  
    hi_vpss_grp_nrx_param nrx_param;  
    hi_u32 reserved[4];  
}hi_vpss_grp_param;
```

成员

成员名称	描述
nrx_param	3DNR标准参数。

成员名称	描述
reserved	预留参数，为保证后续版本兼容性，请务必使用memset结构体方式进行清零初始化，在代码中必须避免显式对reserved字段进行访问。

1.9.20.7.19 hi_fisheye_cfg

说明

定义FISHEYE的鱼镜头LMF参数配置。

定义

```
typedef struct {  
    hi_u16 lmf_coef[HI_FISHEYE_LMF_COEF_NUM]; /* RW; LMF coefficient of gdc len */  
} hi_fisheye_cfg;
```

成员

成员名称	描述
lmf_coef	鱼镜头LMF参数。 #define HI_FISHEYE_LMF_COEF_NUM 128

须知

LMF参数要按照镜头厂商的推荐参数进行转换后再配置（正确的LMF参数符合 $lmf\_coef[i+1] \geq lmf\_coef[i] + 5$ && $lmf\_coef[i+1] \leq lmf\_coef[i] + 31$ && $lmf\_coef[57] < 1024 < lmf\_coef[85]$ && $lmf\_coef[0] = 0$ 的规律），如果配置的参数不满足此规律则会报错，如果配置的参数有误则可能出现总线错误等异常现象，如果没有镜头厂商提供的参数建议关闭LMF功能。

1.9.20.7.20 hi_vpss_chn_mode

说明

定义VPSS通道的工作模式。

定义

```
typedef enum {  
    HI_VPSS_CHN_MODE_AUTO = 0,  
    HI_VPSS_CHN_MODE_USER,  
    HI_VPSS_CHN_MODE_BUTT  
} hi_vpss_chn_mode;
```

成员

成员名称	描述
HI_VPSS_CHN_MODE_USER	用户设置模式。 如果需要在支持从VPSS通道获取处理的图像数据，则必须将模式设置为HI_VPSS_CHN_MODE_USER。 若使用鱼眼矫正功能，则必须将模式设置为HI_VPSS_CHN_MODE_USER。
HI_VPSS_CHN_MODE_AUTO	自动模式。 - AUTO模式下，hi_vpss_chn_attr中的width、height、aspect_ratio、compress_mode、pixel_format、frame_rate、video_format、depth参数设置不生效，相应设置由后端模块确定。 - AUTO模式下，用户无法获取图像。 - AUTO模式下，各通道仅可与一个接收者绑定，主要用于预览和回放场景下做播放控制。
HI_VPSS_CHN_MODE_BUTT	预留值。

1.9.20.7.21 hi_vpss_chn_attr

说明

定义VPSS物理通道的属性。

定义

```
typedef struct {  
    hi_bool mirror_en;  
    hi_bool flip_en;  
    hi_vpss_chn_mode chn_mode; /* RW; vpss channel's work mode. */  
    hi_u32 width;  
    hi_u32 height;  
    hi_video_format video_format; /* RW; video format of target image. */  
    hi_pixel_format pixel_format; /* RW; pixel format of target image. */  
    hi_dynamic_range dynamic_range; /* RW; dynamic_range of target image. */  
    hi_compress_mode compress_mode; /* RW; compression mode of the output. */  
    hi_frame_rate_ctrl frame_rate; /* frame rate control info */  
    hi_u32 depth; /* RW; range: [0, 8]; user get list depth. */  
    hi_aspect_ratio aspect_ratio; /* aspect ratio info. */  
    hi_u32 reserved[10];  
} hi_vpss_chn_attr;
```

成员

成员名称	描述
mirror_en	预留参数，当前版本不支持。 需要用户手动设置为0，避免后续版本的兼容性问题。

成员名称	描述
flip_en	预留参数，当前版本不支持。 需要用户手动设置为0，避免后续版本的兼容性问题。
chn_mode	通道工作模式。
width	目标图像宽度。 - VPSS Group ID在[0,256)范围内，通道0和通道1都支持放大和缩小。 - VPSS Group ID在[256,264)范围内，不同通道支持的功能不同，约束也不同，如下： - 通道0仅支持放大，最大支持16倍放大。当缩放前图像宽度小于4096时，输出宽度不允许超过8192；当缩放前图像宽度大于4096时，最大放大倍数为2。 - 通道1仅支持缩小，最大支持15倍缩小。
height	目标图像高度。 - VPSS Group ID在[0,256)范围内，通道0和通道1都支持放大和缩小 - VPSS Group ID在[256,264)范围内，不同通道支持的功能不同，约束也不同，如下： - 通道0仅支持放大，最大支持16倍放大。 - 通道1仅支持缩小，最大支持15倍缩小。
video_format	视频格式，仅支持HI_VIDEO_FORMAT_LINEAR。
pixel_format	目标图像像素格式。 - VPSS Group ID在[0,256)范围内，支持以下格式：支持HI_PIXEL_FORMAT_YVU_SEMIPLANAR_420、HI_PIXEL_FORMAT_YUV_SEMIPLANAR_420。 如果开启鱼眼矫正，则只支持HI_PIXEL_FORMAT_YVU_SEMIPLANAR_420。 - VPSS Group ID在[256,264)范围内，支持以下格式： - 支持HI_PIXEL_FORMAT_YVU_SEMIPLANAR_422、HI_PIXEL_FORMAT_YUV_SEMIPLANAR_422、HI_PIXEL_FORMAT_YVU_SEMIPLANAR_420、HI_PIXEL_FORMAT_YUV_SEMIPLANAR_420、HI_PIXEL_FORMAT_YUV_400。 如果开启鱼眼矫正，则不支持HI_PIXEL_FORMAT_YVU_SEMIPLANAR_422、HI_PIXEL_FORMAT_YUV_SEMIPLANAR_422、HI_PIXEL_FORMAT_YUV_SEMIPLANAR_420。 - 支持semi-planar 422转semi-planar 420，支持通过设置通道输出像素格式实现UV反转。
dynamic_range	预留参数。
compress_mode	仅支持不压缩HI_COMPRESS_MODE_NONE。

成员名称	描述
frame_rate	帧率控制信息。 - 源帧率与目标帧率都为-1，则不进行帧率控制，在通道 AUTO 模式下通道帧率控制不生效。 - 目标帧率不能大于源帧率。
depth	用户获取通道图像的队列长度。若队列长度为0，HI_VPSS_CHN_MODE_USER 模式下用户无法获取图像。 开启鱼眼矫正功能时，需要额外多分配1个图像内存资源。 取值范围：[0, 8]。
aspect_ratio	幅形比参数。幅形比处理时，先做缩放，再加黑边，缩放比例越大，芯片处理时间越长，可能会导致帧率下降。 - 开启幅形比后，非视频区域的填充颜色，值不大于 0xFFFFFFFF。 - 如果输出图像存在缩放，则图像会缩放至幅形比视频区域的大小，缩放比例不能超过 VPSS 支持的缩放比例。 - ASPECT_RATIO_MANUAL 模式下，坐标值必须满足 x：大于等于 0，小于通道图像宽； y：大于等于 0，小于通道图像高；宽高满足 w：大于等于64，小于等于通道图像宽度； h：大于等于 64,小于等于通道图像高度，并且 x+w 不大于通道图像宽，y+h 不大于通道图像高。
reserved	预留参数，为保证后续版本兼容性，请务必使用memset 结构体方式进行清零初始化，在代码中必须避免显式对 reserved 字段进行访问。

1.9.20.7.22 hi_fisheye_correction_attr

说明

定义鱼眼矫正属性的相关配置。

定义

```
typedef struct {  
    hi_bool enable;  
    hi_fisheye_attr fisheye_attr;  
    hi_size dst_size;  
} hi_fisheye_correction_attr;
```

成员

成员名称	描述
enable	控制鱼眼矫正是否开启。若开启鱼眼矫正，输入图像分辨率增加额外约束为：宽在[1920,8192]范围内，高在[1080,8192]范围内。 取值范围： - HI_FALSE：不开启。 - HI_TRUE：开启。
fisheye_attr	鱼眼属性。
dst_size	鱼眼矫正后的输出图像大小。范围：[640*360, 8192*8192]

1.9.20.7.23 hi_fisheye_mount_mode

说明

定义FISHEYE属性中的安装模式。

定义

```
typedef enum {  
    HI_FISHEYE_MOUNT_MODE_DESKTOP = 0, /* Desktop mount mode */  
    HI_FISHEYE_MOUNT_MODE_CEILING = 1, /* Ceiling mount mode */  
    HI_FISHEYE_MOUNT_MODE_WALL = 2, /* wall mount mode */  
    HI_FISHEYE_MOUNT_MODE_BUTT  
} hi_fisheye_mount_mode;
```

成员

成员名称	描述
HI_FISHEYE_MOUNT_MODE_DESKTOP	地装模式。
HI_FISHEYE_MOUNT_MODE_CEILING	顶装模式。
HI_FISHEYE_MOUNT_MODE_WALL	壁装模式。
HI_FISHEYE_MOUNT_MODE_BUTT	预留值。

1.9.20.7.24 hi_fisheye_view_mode

说明

定义FISHEYE属性中的校正模式。

定义

```
typedef enum {  
    HI_FISHEYE_VIEW_MODE_360_PANORAMA = 0, /* 360 panorama mode of gdc correction */  
    HI_FISHEYE_VIEW_MODE_180_PANORAMA = 1, /* 180 panorama mode of gdc correction */  
    HI_FISHEYE_VIEW_MODE_NORM = 2, /* normal mode of gdc correction */  
    HI_FISHEYE_VIEW_MODE_NO_TRANS = 3, /* no gdc correction */  
    HI_FISHEYE_VIEW_MODE_BUTT  
} hi_fisheye_view_mode;
```

成员

成员名称	描述
HI_FISHEYE_VIEW_MODE_360_PANORAMA	360全景模式。
HI_FISHEYE_VIEW_MODE_180_PANORAMA	180全景模式。
HI_FISHEYE_VIEW_MODE_NORM	Normal校正模式。
HI_FISHEYE_VIEW_MODE_NO_TRANS	不做校正模式，输出原图。
HI_FISHEYE_VIEW_MODE_BUTT	预留值。

1.9.20.7.25 hi_fisheye_rgn_attr

说明

定义FISHEYE每个校正区域的属性配置。

定义

```
typedef struct {  
    hi_fisheye_view_mode view_mode;  
    hi_u32 in_radius;  
    hi_u32 out_radius;  
    hi_u32 pan;  
    hi_u32 tilt;  
    hi_u32 hor_zoom;  
    hi_u32 ver_zoom;  
    hi_rect out_rect;  
} hi_fisheye_rgn_attr;
```

成员

成员名称	描述
view_mode	该校正区域的校正模式。目前不支持设置为HI_FISHEYE_VIEW_MODE_NO_TRANS。

成员名称	描述
in_radius	360全景模式表示该校正区域所对应原图的内半径，其他模式无效。 取值范围：[0, out_radius)。
out_radius	360全景模式表示该校正区域所对应原图的外半径，其他模式为校正区域的可视半径。 取值范围：[1, 3 x max(width of input picture/4, height of input picture/4)]。
pan	该校正区域PTZ参数的Pan值。 取值范围：[0, 360]。
tilt	该校正区域PTZ参数的Tilt值。 取值范围：[0, 360]。
hor_zoom	该校正区域PTZ参数的水平Zoom值。 取值范围：[1, 4095]。
ver_zoom	该校正区域PTZ参数的垂直Zoom值。 取值范围：[1, 4095]。
out_rect	该校正区域的输出位置及宽高。 取值范围：位置及宽高不能超出整个鱼眼矫正输出图的范围；且起始x坐标需16像素的对齐，起始y坐标需2像素对齐。

1.9.20.7.26 hi_fisheye_attr

说明

定义FISHEYE属性的相关配置。

定义

```
typedef struct {
    hi_bool lmf_en;
    hi_bool bg_color_en;
    hi_u32 bg_color;
    hi_s32 hor_offset;
    hi_s32 ver_offset;
    hi_u32 trapezoid_coef;
    hi_s32 fan_strength;
    hi_fisheye_mount_mode mount_mode;
    hi_u32 rgn_num;
    hi_fisheye_rgn_attr fisheye_rgn_attr[HI_FISHEYE_MAX_RGN_NUM];
} hi_fisheye_attr;
```

成员

成员名称	描述
lmf_en	是否使用用户设置的鱼镜头LMF参数。取值范围： - HI_FALSE：不开启。 - HI_TRUE：开启。仅在VPSS组ID的取值范围满足[256, 268)下才可开启。
bg_color_en	是否在输出图像打上背景色。目前仅支持设置为HI_FALSE。 取值范围： - HI_FALSE：不开启。 - HI_TRUE：开启。
bg_color	背景色的颜色RGB888格式，取值范围[0, 0xFFFFFF]。
hor_offset	镜头中心点相对于Sensor中心点的水平偏移，取值范围[-511, 511]，单位为像素。镜头中心点必须为图像中的有效像素点。
ver_offset	镜头中心点相对于Sensor中心点的垂直偏移，取值范围[-511, 511]，单位为像素。镜头中心点必须为图像中的有效像素点。
trapezoid_coef	梯形校正强度系数。用于壁装时的俯仰角校正，取值范围[0, 32]。
fan_strength	扇形校正强度系数，仅在180模式时有效，取值范围[-760, 760]。
mount_mode	鱼眼校正安装模式。
rgn_num	一幅图像的校正区域数目，最多支持4个区域。
fisheye_rgn_attr	每个校正区域各自的属性配置。

1.9.20.8 VO 视频输出

1.9.20.8.1 hi_vo_dev

说明

定义设备号。

定义

```
typedef hi_s32 hi_vo_dev;
```

成员

成员名称	描述
hi_vo_dev	视频输出模块有2个视频输出设备，如下定义： 0：DHD0设备，即高清显示设备0。 1：DHD1设备，即高清显示设备1。

1.9.20.8.2 hi_vo_layer

说明

定义视频层号。

定义

```
typedef hi_s32 hi_vo_layer;
```

成员

成员名称	描述
hi_vo_layer	视频输出模块有4个视频层和5个图形层，如下定义： 0：HI_VO_LAYER_V0，即视频层 0； 1：HI_VO_LAYER_V1，即视频层 1； 2：HI_VO_LAYER_V2，即视频层 2，用作 PIP 层；当前版本不支持。 3：HI_VO_LAYER_V3，即视频层 3；当前版本不支持。 4：HI_VO_LAYER_G0，即图形层 0； 5：HI_VO_LAYER_G1，即图形层 1； 6：HI_VO_LAYER_G2，即图形层 2，用作鼠标层； 7：HI_VO_LAYER_G3，即图形层 3；当前版本不支持。 8：HI_VO_LAYER_G4，即图形层 4；当前版本不支持。

 说明

当前版本接口HI_VO_LAYER_G0到HI_VO_LAYER_G4配置无效。

1.9.20.8.3 hi_vo_chn

说明

定义通道备号。

定义

```
typedef hi_s32 hi_vo_chn;
```

成员

成员名称	描述
hi_vo_chn	支持最大64路通道。

1.9.20.8.4 hi_vo_intf_type

说明

定义接口类型。

定义

```
typedef hi_u32 hi_vo_intf_type;
```

成员

成员名称	描述
hi_vo_intf_type	接口类型支持： HI_VO_INTF_VGA：VGA接口输出，定义值为（0x01L << 1）。 HI_VO_INTF_HDMI：HDMI0接口输出，定义值为（0x01L << 4）。 HI_VO_INTF_HDMI1：HDMI1接口输出，定义值为（0x01L << 12）。 HI_VO_INTF_MIPI：MIPITX接口输出，定义值为（0x01L << 10）。

1.9.20.8.5 hi_vo_intf_sync

说明

定义显示输出时序。

定义

```
typedef enum {  
    HI_VO_OUT_PAL = 0, /* PAL standard */  
    HI_VO_OUT_NTSC = 1, /* NTSC standard */  
    HI_VO_OUT_960H_PAL = 2, /* ITU-R BT.1302 960 x 576 at 50 Hz (interlaced) */  
    HI_VO_OUT_960H_NTSC = 3, /* ITU-R BT.1302 960 x 480 at 60 Hz (interlaced) */  
    HI_VO_OUT_640x480_60 = 4, /* VESA 640 x 480 at 60 Hz (non-interlaced) CVT */  
    HI_VO_OUT_480P60 = 5, /* 720 x 480 at 60 Hz. */  
    HI_VO_OUT_576P50 = 6, /* 720 x 576 at 50 Hz. */  
    HI_VO_OUT_800x600_60 = 7, /* VESA 800 x 600 at 60 Hz (non-interlaced) */  
}
```

```
HI_VO_OUT_1024x768_60 = 8, /* VESA 1024 x 768 at 60 Hz (non-interlaced) */
HI_VO_OUT_720P50 = 9, /* 1280 x 720 at 50 Hz. */
HI_VO_OUT_720P60 = 10, /* 1280 x 720 at 60 Hz. */
HI_VO_OUT_1280x800_60 = 11, /* 1280*800@60Hz VGA@60Hz */
HI_VO_OUT_1280x1024_60 = 12, /* VESA 1280 x 1024 at 60 Hz (non-interlaced) */
HI_VO_OUT_1366x768_60 = 13, /* VESA 1366 x 768 at 60 Hz (non-interlaced) */
HI_VO_OUT_1400x1050_60 = 14, /* VESA 1400 x 1050 at 60 Hz (non-interlaced) CVT */
HI_VO_OUT_1440x900_60 = 15, /* VESA 1440 x 900 at 60 Hz (non-interlaced) CVT Compliant */
HI_VO_OUT_1680x1050_60 = 16, /* VESA 1680 x 1050 at 60 Hz (non-interlaced) */
HI_VO_OUT_1080P24 = 17, /* 1920 x 1080 at 24 Hz. */
HI_VO_OUT_1080P25 = 18, /* 1920 x 1080 at 25 Hz. */
HI_VO_OUT_1080P30 = 19, /* 1920 x 1080 at 30 Hz. */
HI_VO_OUT_1080I50 = 20, /* 1920 x 1080 at 50 Hz, interlaced. */
HI_VO_OUT_1080I60 = 21, /* 1920 x 1080 at 60 Hz, interlaced. */
HI_VO_OUT_1080P50 = 22, /* 1920 x 1080 at 50 Hz. */
HI_VO_OUT_1080P60 = 23, /* 1920 x 1080 at 60 Hz. */
HI_VO_OUT_1600x1200_60 = 24, /* VESA 1600 x 1200 at 60 Hz (non-interlaced) */
HI_VO_OUT_1920x1200_60 = 25, /* VESA 1920 x 1200 at 60 Hz (non-interlaced) CVT (Reduced Blanking)
*/
HI_VO_OUT_1920x2160_30 = 26, /* 1920x2160_30 */
HI_VO_OUT_2560x1440_30 = 27, /* 2560x1440_30 */
HI_VO_OUT_2560x1440_60 = 28, /* 2560x1440_60 */
HI_VO_OUT_2560x1600_60 = 29, /* 2560x1600_60 */
HI_VO_OUT_3840x2160_24 = 30, /* 3840x2160_24 */
HI_VO_OUT_3840x2160_25 = 31, /* 3840x2160_25 */
HI_VO_OUT_3840x2160_30 = 32, /* 3840x2160_30 */
HI_VO_OUT_3840x2160_50 = 33, /* 3840x2160_50 */
HI_VO_OUT_3840x2160_60 = 34, /* 3840x2160_60 */
HI_VO_OUT_4096x2160_24 = 35, /* 4096x2160_24 */
HI_VO_OUT_4096x2160_25 = 36, /* 4096x2160_25 */
HI_VO_OUT_4096x2160_30 = 37, /* 4096x2160_30 */
HI_VO_OUT_4096x2160_50 = 38, /* 4096x2160_50 */
HI_VO_OUT_4096x2160_60 = 39, /* 4096x2160_60 */
HI_VO_OUT_7680x4320_30 = 40, /* 7680x4320_30 */
HI_VO_OUT_240x320_50 = 41, /* 240x320_50 */
HI_VO_OUT_320x240_50 = 42, /* 320x240_50 */
HI_VO_OUT_240x320_60 = 43, /* 240x320_60 */
HI_VO_OUT_320x240_60 = 44, /* 320x240_60 */
HI_VO_OUT_800x600_50 = 45, /* 800x600_50 */
HI_VO_OUT_720x1280_60 = 46, /* For MIPI DSI Tx 720 x1280 at 60 Hz */
HI_VO_OUT_1080x1920_60 = 47, /* For MIPI DSI Tx 1080x1920 at 60 Hz */
HI_VO_OUT_800x480_60 = 48, /* For MIPI DSI Tx 800x480 at 60 Hz */
HI_VO_OUT_1080x1920_60_VIDEO = 49, /* For MIPI DSI Tx 1080x1920 at 60 Hz in video mode */
HI_VO_OUT_USER = 50, /* User timing. */
HI_VO_OUT_BUTT
} hi_vo_intf_sync;
```

成员

成员名称	描述
HI_VO_OUT_PAL	PAL (Phase Alternating Line) 制式。当前版本不支持。
HI_VO_OUT_NTSC	NTSC (National Television System Committee) 制式。当前版本不支持。
HI_VO_OUT_960H_PAL	PAL 960H制式。当前版本不支持。
HI_VO_OUT_960H_NTSC	NTSC 960H制式。当前版本不支持。
HI_VO_OUT_640x480_60	640x480 60帧 。当前版本不支持。
HI_VO_OUT_480P60	480P 60帧， HDMI/VGA支持。

成员名称	描述
HI_VO_OUT_576P50	576P 50帧。当前版本不支持。
HI_VO_OUT_800x600_60	800x600 60帧，HDMI/VGA支持。
HI_VO_OUT_1024x768_60	1024x768 60帧。当前版本不支持。
HI_VO_OUT_720P50	720P 50帧。当前版本不支持。
HI_VO_OUT_720P60	720P 60帧，HDMI/VGA支持。
HI_VO_OUT_1280x800_60	1280x800 60帧。当前版本不支持。
HI_VO_OUT_1280x1024_60	1280x1024 60帧。当前版本不支持。
HI_VO_OUT_1366x768_60	1366x768 60帧。当前版本不支持。
HI_VO_OUT_1400x1050_60	1400x1050 60帧。当前版本不支持。
HI_VO_OUT_1440x900_60	1440x900 60帧。当前版本不支持。
HI_VO_OUT_1680x1050_60	1680x1050 60帧。当前版本不支持。
HI_VO_OUT_1080P24	1080P 24帧。当前版本不支持。
HI_VO_OUT_1080P25	1080P 25帧。当前版本不支持。
HI_VO_OUT_1080P30	1080P 30帧，HDMI/VGA支持。当前版本不支持。
HI_VO_OUT_1080I50	1080I 50帧。当前版本不支持。
HI_VO_OUT_1080I60	1080I 60帧。当前版本不支持。
HI_VO_OUT_1080P50	1080P50帧，HDMI/VGA支持。
HI_VO_OUT_1080P60	1080P60帧，HDMI/VGA支持。
HI_VO_OUT_1600x1200_60	1600x1200 60帧，HDMI支持。
HI_VO_OUT_1920x1200_60	1920x1200 60帧，HDMI支持。
HI_VO_OUT_1920x2160_30	1920x2160 30帧。当前版本不支持。
HI_VO_OUT_2560x1440_30	2560x1440 30帧。当前版本不支持。
HI_VO_OUT_2560x1440_60	2560x1440 60帧。当前版本不支持。
HI_VO_OUT_2560x1600_60	2560x1600 60帧。当前版本不支持。

成员名称	描述
HI_VO_OUT_3840x2160_24	3840x2160 24帧，HDMI支持。
HI_VO_OUT_3840x2160_25	3840x2160 25帧，HDMI支持。
HI_VO_OUT_3840x2160_30	3840x2160 30帧，HDMI支持。
HI_VO_OUT_3840x2160_50	3840x2160 50帧，HDMI支持。
HI_VO_OUT_3840x2160_60	3840x2160 60帧，HDMI支持。
HI_VO_OUT_4096x2160_24	4096x2160 24帧。当前版本不支持。
HI_VO_OUT_4096x2160_25	4096x2160 25帧。当前版本不支持。
HI_VO_OUT_4096x2160_30	4096x2160 30帧。当前版本不支持。
HI_VO_OUT_4096x2160_50	4096x2160 50帧。当前版本不支持。
HI_VO_OUT_4096x2160_60	4096x2160 60帧。当前版本不支持。
HI_VO_OUT_7680x4320_30	7680x4320 30帧。当前版本不支持。
HI_VO_OUT_240x320_50	240x320 50帧。当前版本不支持。
HI_VO_OUT_320x240_50	320x240 50帧。当前版本不支持。
HI_VO_OUT_240x320_60	240x320 60帧。当前版本不支持。
HI_VO_OUT_320x240_60	320x240 60帧。当前版本不支持。
HI_VO_OUT_800x600_50	800x600 50帧。当前版本不支持。
HI_VO_OUT_720x1280_60	720x1280 60帧。当前版本不支持。
HI_VO_OUT_1080x1920_60	1080x1920 60帧，MIPITX Mate9屏cmd模式支持。
HI_VO_OUT_800x480_60	800x480 60帧，MIPITX 树莓派屏支持。
HI_VO_OUT_1080x1920_60_VIDEO	1080x1920 60帧，MIPITX Mate9屏video模式支持。
HI_VO_OUT_USER	用户配置规格。当前版本不支持。
HI_VO_OUT_BUTT	预留值。

1.9.20.8.6 hi_vo_sync_info

说明

定义时序信息结构体。

 说明

当前版本目前该结构体未使用，仅在[hi_vo_intf_sync](#)设置为HI_VO_OUT_USER时用户才感知。

定义

```
typedef struct {
    hi_bool syncm;
    hi_bool iop;
    hi_u8 intfbb;
    hi_u16 vact;
    hi_u16 vbb;
    hi_u16 vfb;
    hi_u16 hact;
    hi_u16 hbb;
    hi_u16 hfb;
    hi_u16 hmid;
    hi_u16 bvact;
    hi_u16 bvbb;
    hi_u16 bvfb;
    hi_u16 hpw;
    hi_u16 vpw;
    hi_bool idv;
    hi_bool ihs;
    hi_bool ivs;
} hi_vo_sync_info;
```

成员

成员名称	描述
syncm	同步模式，RGB接口选择1，表示信号同步。取值范围：[0, 1]
iop	0为隔行，1为逐行，RGB接口一般配置为 1。取值范围：[0, 1]
intfbb	无效参数，可以忽略。取值范围：[0, 255]
vact	垂直有效区，隔行输出时表示顶场垂直有效区。单位：行。取值范围：[100, 4096]
vbb	垂直消隐后肩，隔行输出时表示顶场垂直消隐后肩。单位：行。取值范围：[1, 256]
vfb	垂直消隐前肩，隔行输出时表示顶场垂直消隐前肩。单位：行。取值范围：[1, 256]
hact	水平有效区。单位：像素。取值范围：[1, 4096]
hbb	水平消隐后肩。单位：像素。取值范围：[1, 65535]
hfb	水平消隐前肩。单位：像素。取值范围：[1, 65535]
hmid	底场垂直同步有效像素值。取值范围：[0, 65535]

成员名称	描述
bvact	底场垂直有效区，隔行时有效。单位：行。取值范围：[0, 4096]
bvbb	底场垂直消隐后肩，隔行时有效。单位：行。取值范围：[0, 256]
bvfb	底场垂直消隐前肩，隔行时有效。单位：行。取值范围：[0, 256]。
hpw	水平同步信号的宽度。单位：像素。取值范围：[1, 65535]
vpw	垂直同步信号的宽度。单位：行。取值范围：[1, 256]
idv	数据有效信号的极性。配置0为高有效，配置1为低有效。取值范围：[0, 1]
ihs	水平有效信号的极性，配置0为高有效，配置1为低有效。取值范围：[0, 1]
ivs	垂直有效信号的极性，配置0为高有效，配置1为低有效。取值范围：[0, 1]

1.9.20.8.7 hi_vo_partition_mode

说明

定义视频层分割处理模式。

定义

```
typedef enum {  
    HI_VO_PARTITION_MODE_SINGLE = 0,  
    HI_VO_PARTITION_MODE_MULTI = 1,  
    HI_VO_PARTITION_MODE_BUTT  
} hi_vo_partition_mode;
```

成员

成员名称	描述
HI_VO_PARTITION_MODE_SINGLE	视频层按照单区域配置给硬件显示。
HI_VO_PARTITION_MODE_MULTI	视频层按照多区域配置给硬件显示。
HI_VO_PARTITION_MODE_BUTT	预留值。

1.9.20.8.8 hi_vo_pub_attr

说明

定义视频输出公共属性结构体。

定义

```
typedef struct {  
    hi_u32 bg_color;  
    hi_vo_intf_type intf_type;  
    hi_vo_intf_sync intf_sync;  
    hi_vo_sync_info sync_info;  
} hi_vo_pub_attr;
```

成员

成员名称	描述
bg_color	设备背景色，表示方法为RGB888。
intf_type	接口类型，可选择一种类型或多种类型的组合。
intf_sync	接口时序。
sync_info	接口时序信息。 intf_sync配置用户时序HI_VO_OUT_USER时，该结构体生效。

说明

当前版本支持组合场景如下：
HI_VO_INTF_HDMI绑定DHD0；不支持与HI_VO_INTF_HDMI1，HI_VO_INTF_VGA，HI_VO_INTF_MIPI同时送显
HI_VO_INTF_VGA绑定DHD0；不支持与HI_VO_INTF_HDMI，HI_VO_INTF_HDMI1，HI_VO_INTF_MIPI同时送显
HI_VO_INTF_MIPI绑定DHD0；不支持与HI_VO_INTF_HDMI，HI_VO_INTF_HDMI1，HI_VO_INTF_VGA同时送显
HI_VO_INTF_HDMI1绑定DHD1；不支持与HI_VO_INTF_HDMI，HI_VO_INTF_VGA，HI_VO_INTF_MIPI同时送显

1.9.20.8.9 hi_vo_video_layer_attr

说明

定义视频层属性。

定义

```
typedef struct {  
    hi_rect display_rect;  
    hi_size img_size;  
    hi_u32 display_frame_rate;  
    hi_pixel_format pixel_format;  
    hi_bool double_frame_en;
```

```
hi_bool cluster_mode_en;  
hi_dynamic_range dst_dynamic_range;  
hi_u32 display_buf_len;  
hi_vo_partition_mode partition_mode;  
hi_compress_mode compress_mode;  
} hi_vo_video_layer_attr;
```

成员

成员名称	描述
display_rect	视频显示区域矩形结构体， - 当partition_mode参数设置为HI_VO_PARTITION_MODE_SINGLE时，display_rect.x、display_rect.y为动态属性，display_rect.width、display_rect.height为静态属性。 - 当partition_mode参数设置为HI_VO_PARTITION_MODE_MULTI时，display_rect为静态属性。
img_size;	图像分辨率结构体，即合成画面尺寸，静态属性。
display_frame_rate	视频显示帧率，静态属性。
pixel_format	视频层输入像素格式，静态属性。 支持以下格式：YVU420 SEMIPLANAR、YVU422 SEMIPLANAR、YUV400。
double_frame_en	视频层倍帧开关，静态属性。当前版本不支持。
cluster_mode_en	视频层内存聚集使能开关，静态属性。当前版本不支持。
dst_dynamic_range	视频层输出动态范围类型，静态属性。
display_buf_len	视频层显示缓存的长度，静态属性。 当前场景无需关注。
partition_mode	视频层的分割模式，静态属性。
compress_mode	视频层支持压缩或解压模式，静态属性。 当前仅支持HI_COMPRESS_MODE_NONE。

1.9.20.8.10 hi_vo_chn_attr

说明

定义视频输出通道属性。

定义

```
typedef struct {  
    hi_u32 priority;  
    hi_rect rect;  
    hi_bool deflicker_en;  
} hi_vo_chn_attr;
```

成员

成员名称	描述
priority	视频通道叠加优先级，数值越大优先级越高，优先级高的在上层。 hi_vo_video_layer_attr.partition_mode参数设置为HI_VO_PARTITION_MODE_SINGLE时，priority参数值有效，且取值范围为：[0, 64)。 动态属性。
rect	通道矩形显示区域。 以屏幕的左上角为原点，其取值必须是2对齐，且该矩形区域必须在屏幕范围之内、矩形区域的宽高要求2对齐、矩形区域的宽高必须不小于64。 动态属性。
deflicker_en	是否使能抗闪烁，当前不支持。 - HI_TRUE：使能 - HI_FALSE：禁用

1.9.20.8.11 hi_vo_chn_param

说明

定义视频输出通道参数，用于幅型比功能。

定义

```
typedef struct {  
    hi_aspect_ratio aspect_ratio  
} hi_vo_chn_param;
```

成员

成员名称	描述
aspect_ratio	幅型比参数。

1.9.20.9 TDE 图形绘制

1.9.20.9.1 hi_tde_color_format

说明

颜色格式枚举。

定义

```
typedef enum {  
    HI_TDE_COLOR_FORMAT_RGB444 = 0,  
    HI_TDE_COLOR_FORMAT_BGR444,  
    HI_TDE_COLOR_FORMAT_RGB555,  
    HI_TDE_COLOR_FORMAT_BGR555,  
    HI_TDE_COLOR_FORMAT_RGB565,  
    HI_TDE_COLOR_FORMAT_BGR565,  
    HI_TDE_COLOR_FORMAT_RGB888,  
    HI_TDE_COLOR_FORMAT_BGR888,  
    HI_TDE_COLOR_FORMAT_ARGB4444,  
    HI_TDE_COLOR_FORMAT_ABGR4444,  
    HI_TDE_COLOR_FORMAT_RGBA4444,  
    HI_TDE_COLOR_FORMAT_BGRA4444,  
    HI_TDE_COLOR_FORMAT_ARGB1555,  
    HI_TDE_COLOR_FORMAT_ABGR1555,  
    HI_TDE_COLOR_FORMAT_RGBA1555,  
    HI_TDE_COLOR_FORMAT_BGRA1555,  
    HI_TDE_COLOR_FORMAT_ARGB8565,  
    HI_TDE_COLOR_FORMAT_ABGR8565,  
    HI_TDE_COLOR_FORMAT_RGBA8565,  
    HI_TDE_COLOR_FORMAT_BGRA8565,  
    HI_TDE_COLOR_FORMAT_ARGB8888,  
    HI_TDE_COLOR_FORMAT_ABGR8888,  
    HI_TDE_COLOR_FORMAT_RGBA8888,  
    HI_TDE_COLOR_FORMAT_BGRA8888,  
    HI_TDE_COLOR_FORMAT_RABG8888,  
    HI_TDE_COLOR_FORMAT_CLUT1,  
    HI_TDE_COLOR_FORMAT_CLUT2,  
    HI_TDE_COLOR_FORMAT_CLUT4,  
    HI_TDE_COLOR_FORMAT_CLUT8,  
    HI_TDE_COLOR_FORMAT_ACLUT44,  
    HI_TDE_COLOR_FORMAT_ACLUT88,  
    HI_TDE_COLOR_FORMAT_A1,  
    HI_TDE_COLOR_FORMAT_A8,  
    HI_TDE_COLOR_FORMAT_YCbCr888,  
    HI_TDE_COLOR_FORMAT_AYCbCr8888,  
    HI_TDE_COLOR_FORMAT_YCbCr422,  
    HI_TDE_COLOR_FORMAT_PKGVYUY,  
    HI_TDE_COLOR_FORMAT_BYTE,  
    HI_TDE_COLOR_FORMAT_HALFWORD,  
    HI_TDE_COLOR_FORMAT_JPG_YCbCr400MBP,  
    HI_TDE_COLOR_FORMAT_JPG_YCbCr422MBHP,  
    HI_TDE_COLOR_FORMAT_JPG_YCbCr422MBVP,  
    HI_TDE_COLOR_FORMAT_MP1_YCbCr420MBP,  
    HI_TDE_COLOR_FORMAT_MP2_YCbCr420MBP,  
    HI_TDE_COLOR_FORMAT_MP2_YCbCr420MBI,  
    HI_TDE_COLOR_FORMAT_JPG_YCbCr4,  
    HI_TDE_COLOR_FORMAT_JPG_YCbCr444MBP,  
    HI_TDE_COLOR_FORMAT_MAX  
} hi_tde_color_format;
```

成员

成员名称	描述
HI_TDE_COLOR_FORMAT_RGB444	RGB444颜色格式。Atlas 200/500 A2推理产品不支持。
HI_TDE_COLOR_FORMAT_BGR444	BGR444颜色格式。Atlas 200/500 A2推理产品不支持。
HI_TDE_COLOR_FORMAT_RGB555	RGB555颜色格式。Atlas 200/500 A2推理产品不支持。

成员名称	描述
HI_TDE_COLOR_FORMAT_BGR555	BGR555颜色格式。Atlas 200/500 A2推理产品不支持。
HI_TDE_COLOR_FORMAT_RGB565	RGB565颜色格式。Atlas 200/500 A2推理产品不支持。
HI_TDE_COLOR_FORMAT_BGR565	BGR565颜色格式。Atlas 200/500 A2推理产品不支持。
HI_TDE_COLOR_FORMAT_RGB888	RGB888颜色格式。Atlas 200/500 A2推理产品不支持。
HI_TDE_COLOR_FORMAT_BGR888	BGR888颜色格式。Atlas 200/500 A2推理产品不支持。
HI_TDE_COLOR_FORMAT_ARGB4444	ARGB4444颜色格式。Atlas 200/500 A2推理产品不支持。
HI_TDE_COLOR_FORMAT_ABGR4444	ABGR4444颜色格式。Atlas 200/500 A2推理产品不支持。
HI_TDE_COLOR_FORMAT_RGBA4444	RGBA4444颜色格式。Atlas 200/500 A2推理产品不支持。
HI_TDE_COLOR_FORMAT_BGRA4444	BGRA4444颜色格式。Atlas 200/500 A2推理产品不支持。
HI_TDE_COLOR_FORMAT_ARGB1555	ARGB1555颜色格式。Atlas 200/500 A2推理产品不支持。
HI_TDE_COLOR_FORMAT_ABGR1555	ABGR1555颜色格式。Atlas 200/500 A2推理产品不支持。
HI_TDE_COLOR_FORMAT_RGBA1555	RGBA1555颜色格式。Atlas 200/500 A2推理产品不支持。
HI_TDE_COLOR_FORMAT_BGRA1555	BGRA1555颜色格式。Atlas 200/500 A2推理产品不支持。
HI_TDE_COLOR_FORMAT_ARGB8565	ARGB8565颜色格式。Atlas 200/500 A2推理产品不支持。
HI_TDE_COLOR_FORMAT_ABGR8565	ABGR8565颜色格式。Atlas 200/500 A2推理产品不支持。
HI_TDE_COLOR_FORMAT_RGBA8565	RGBA8565颜色格式。Atlas 200/500 A2推理产品不支持。
HI_TDE_COLOR_FORMAT_BGRA8565	BGRA8565颜色格式。Atlas 200/500 A2推理产品不支持。
HI_TDE_COLOR_FORMAT_ARGB8888	ARGB8888颜色格式。
HI_TDE_COLOR_FORMAT_ABGR8888	ABGR8888颜色格式。Atlas 200/500 A2推理产品不支持。

成员名称	描述
HI_TDE_COLOR_FORMAT_RGBA8888	RGBA8888颜色格式。Atlas 200/500 A2推理产品不支持。
HI_TDE_COLOR_FORMAT_BGRA8888	BGRA8888颜色格式。Atlas 200/500 A2推理产品不支持。
HI_TDE_COLOR_FORMAT_RABG8888	RABG8888颜色格式。Atlas 200/500 A2推理产品不支持。
HI_TDE_COLOR_FORMAT_CLUT1	CLUT1颜色格式。Atlas 200/500 A2推理产品不支持。
HI_TDE_COLOR_FORMAT_CLUT2	CLUT2颜色格式。Atlas 200/500 A2推理产品不支持。
HI_TDE_COLOR_FORMAT_CLUT4	CLUT4颜色格式。Atlas 200/500 A2推理产品不支持。
HI_TDE_COLOR_FORMAT_CLUT8	CLUT8颜色格式。Atlas 200/500 A2推理产品不支持。
HI_TDE_COLOR_FORMAT_ACLUT44	ACLUT44颜色格式。Atlas 200/500 A2推理产品不支持。
HI_TDE_COLOR_FORMAT_ACLUT88	ACLUT88颜色格式。Atlas 200/500 A2推理产品不支持。
HI_TDE_COLOR_FORMAT_A1	A1颜色格式。Atlas 200/500 A2推理产品不支持。
HI_TDE_COLOR_FORMAT_A8	A8颜色格式。Atlas 200/500 A2推理产品不支持。
HI_TDE_COLOR_FORMAT_YCbCr888	YCbCr888颜色格式。Atlas 200/500 A2推理产品不支持。
HI_TDE_COLOR_FORMAT_AYCbCr8888	AYCbCr8888颜色格式。Atlas 200/500 A2推理产品不支持。
HI_TDE_COLOR_FORMAT_YCbCr422	YCbCr422颜色格式，YVYU排列。Atlas 200/500 A2推理产品不支持。
HI_TDE_COLOR_FORMAT_PKGVYUY	YCbCr422颜色格式，VYUY排列。Atlas 200/500 A2推理产品不支持。
HI_TDE_COLOR_FORMAT_BYTE	Byte格式。Atlas 200/500 A2推理产品不支持。
HI_TDE_COLOR_FORMAT_HALFWORD	halfword格式。Atlas 200/500 A2推理产品不支持。
HI_TDE_COLOR_FORMAT_JPG_YCbCr400MBP	YCbCr400MBP颜色格式。Atlas 200/500 A2推理产品不支持。
HI_TDE_COLOR_FORMAT_JPG_YCbCr422MBHP	YCbCr422MBHP颜色格式。Atlas 200/500 A2推理产品不支持。

成员名称	描述
HI_TDE_COLOR_FORMAT_JPG_YCbCr422MBVP	YCbCr422MBVP颜色格式。Atlas 200/500 A2推理产品不支持。
HI_TDE_COLOR_FORMAT_MP1_YCbCr420MBP	YCbCr420MBP颜色格式。Atlas 200/500 A2推理产品不支持。
HI_TDE_COLOR_FORMAT_MP2_YCbCr420MBP	YCbCr420MBP颜色格式。Atlas 200/500 A2推理产品不支持。
HI_TDE_COLOR_FORMAT_MP2_YCbCr420MBI	YCbCr420MBI颜色格式。Atlas 200/500 A2推理产品不支持。
HI_TDE_COLOR_FORMAT_JPG_YCbCr420MBP	YCbCr420MBP颜色格式。Atlas 200/500 A2推理产品不支持。
HI_TDE_COLOR_FORMAT_JPG_YCbCr444MBP	YCbCr444MBP颜色格式。Atlas 200/500 A2推理产品不支持。
TDE_COLOR_FMT_BUTT	无效的像素格式。

1.9.20.9.2 hi_tde_surface

说明

位图信息结构体。

定义

```
typedef struct {
    hi_u64 phys_addr;
    hi_u32 phys_len;
    hi_tde_color_format color_format;
    hi_u32 height;
    hi_u32 width;
    hi_u32 stride;
    hi_bool is_ycbcr_clut;
    hi_bool alpha_max_is_255;
    hi_u8 alpha0;
    hi_u8 alpha1;
    hi_u64 cbcr_phys_addr;
    hi_u32 cbcr_phys_len;
    hi_u32 cbcr_stride;
    hi_u64 clut_phys_addr;
    hi_u32 clut_phys_len;
} hi_tde_surface;
```

成员

成员名称	描述
phys_addr	位图或Y分量的起始地址。
phys_len	物理地址长度。
color_format	颜色格式。

成员名称	描述
height	位图高度。
width	位图宽度。
stride	位图或Y分量的步长。
is_ycbcr_clut	CLUT是否在YCbCr空间中。Atlas 200/500 A2推理产品不支持。
alpha_max_is_255	alpha最大值是128还是255。Atlas 200/500 A2推理产品不支持。
support_alpha_ex_1555	ARGB1555位图中是否支持alpha扩展。Atlas 200/500 A2推理产品不支持。
alpha0	alpha0，用于ARGB1555 format。Atlas 200/500 A2推理产品不支持。
alpha1	alpha1，用于ARGB1555 format。Atlas 200/500 A2推理产品不支持。
cbcr_phys_addr	CbCr分量的地址。Atlas 200/500 A2推理产品不支持。
cbcr_phys_len	CbCr分量的地址长度。Atlas 200/500 A2推理产品不支持。
cbcr_stride	CbCr分量的步长。Atlas 200/500 A2推理产品不支持。
clut_phys_addr	Clut表的地址。Atlas 200/500 A2推理产品不支持。
clut_phys_len	Clut表的地址长度。Atlas 200/500 A2推理产品不支持。

1.9.20.9.3 hi_tde_rect

说明

矩形信息结构体。

定义

```
typedef struct {  
    hi_s32 pos_x;  
    hi_s32 pos_y;  
    hi_u32 width;  
    hi_u32 height;  
} hi_tde_rect;
```

成员

成员名称	描述
pos_x	水平坐标。
pos_y	垂直坐标。

成员名称	描述
width	宽度。
height	高度。

1.9.20.9.4 hi_tde_none_src

说明

无源位图操作区域位图信息结构体，当前在快速填充功能中作为入参使用。

定义

```
typedef struct {  
    hi_tde_surface *dst_surface;  
    hi_tde_rect *dst_rect;  
} hi_tde_none_src;
```

成员

成员名称	描述
dst_surface	目标位图。
dst_rect	目标位图操作区域。

1.9.20.9.5 hi_tde_single_src

说明

单源位图区域信息和目标位图区域信息结构体，当前在快速拷贝功能中作为入参使用。

定义

```
typedef struct {  
    hi_tde_surface *src_surface;  
    hi_tde_surface *dst_surface;  
    hi_tde_rect *src_rect;  
    hi_tde_rect *dst_rect;  
} hi_tde_single_src;
```

成员

成员名称	描述
src_surface	源位图。
dst_surface	目标位图。
src_rect	源位图操作区域。

成员名称	描述
dst_rect	目标位图操作区域。

1.9.20.9.6 hi_tde_double_src

说明

双源位图区域信息和目标位图区域信息结构体，当前在模式填充中使用。

定义

```
typedef struct {  
    hi_tde_surface *bg_surface;  
    hi_tde_surface *fg_surface;  
    hi_tde_surface *dst_surface;  
    hi_tde_rect *bg_rect;  
    hi_tde_rect *fg_rect;  
    hi_tde_rect *dst_rect;  
} hi_tde_double_src;
```

成员

成员名称	描述
bg_surface	背景位图。
fg_surface	前景位图。
dst_surface	目标位图。
bg_rect	背景位图操作区域。
fg_rect	前景位图操作区域。
dst_rect	目标位图操作区域。

1.9.20.9.7 hi_tde_fill_color

说明

填充颜色信息结构体。

定义

```
typedef struct {  
    hi_tde_color_format color_format;  
    hi_u32 color_value;  
} hi_tde_fill_color;
```

成员

成员名称	描述
color_format	颜色格式。
color_value	颜色数据。

1.9.20.9.8 hi_tde_blend_cmd

说明

混合选项命令。

定义

```
typedef enum {  
    HI_TDE_BLEND_CMD_NONE = 0x0,  
    HI_TDE_BLEND_CMD_CLEAR,  
    HI_TDE_BLEND_CMD_SRC,  
    HI_TDE_BLEND_CMD_SRCOVER,  
    HI_TDE_BLEND_CMD_DSTOVER,  
    HI_TDE_BLEND_CMD_SRCIN,  
    HI_TDE_BLEND_CMD_DSTIN,  
    HI_TDE_BLEND_CMD_SRCOUT,  
    HI_TDE_BLEND_CMD_DSTOUT,  
    HI_TDE_BLEND_CMD_SRCATOP,  
    HI_TDE_BLEND_CMD_DSTATOP,  
    HI_TDE_BLEND_CMD_ADD,  
    HI_TDE_BLEND_CMD_XOR,  
    HI_TDE_BLEND_CMD_DST,  
    HI_TDE_BLEND_CMD_CONFIG,  
    HI_TDE_BLEND_CMD_MAX  
} hi_tde_blend_cmd;
```

成员

描述中有以下定义

ff: 前景图混合系数, 范围[0, 1]

bf: 背景图混合系数, 范围[0, 1]

fa: 前景图透明度, 范围[0, 1]

ba: 背景图透明度, 范围[0, 1]

成员名称	描述
HI_TDE_BLEND_CMD_NONE	ff取fa, bf取1.0 - fa。
HI_TDE_BLEND_CMD_CLEAR	ff取0.0, fd取0.0。
HI_TDE_BLEND_CMD_SRC	ff取1.0, fd取0.0。

成员名称	描述
HI_TDE_BLEND_CMD_SRC OVER	ff取1.0, bf取1.0 - fa。
HI_TDE_BLEND_CMD_DS TOVER	ff取1.0 - ba, bf取1.0。
HI_TDE_BLEND_CMD_SRC IN	ff取ba, bf取0.0。
HI_TDE_BLEND_CMD_DS TIN	ff取0.0, bf取fa。
HI_TDE_BLEND_CMD_SRC OUT	ff取1.0 - ba, bf取0.0。
HI_TDE_BLEND_CMD_DS TOUT	ff取0.0, bf取1.0 - fa。
HI_TDE_BLEND_CMD_SRC ATOP	ff取ba, bf取1.0 - fa。
HI_TDE_BLEND_CMD_DS TATOP	ff取1.0 - ba, bf取fa。
HI_TDE_BLEND_CMD_AD D	ff取1.0, bf取1.0。
HI_TDE_BLEND_CMD_XO R	ff取1.0 - ba, bf取1.0 - fa。
HI_TDE_BLEND_CMD_DS T	ff取0.0, bf取1.0。
HI_TDE_BLEND_CMD_CO NFIG	用户自定义参数规则。
HI_TDE_BLEND_CMD_MA X	无效混合选项。

1.9.20.9.9 hi_tde_blend_mode

说明

混合选项命令。

定义

```
typedef enum {  
    HI_TDE_BLEND_ZERO = 0x0,  
    HI_TDE_BLEND_ONE,  
    HI_TDE_BLEND_SRC2COLOR,  
    HI_TDE_BLEND_INVSRC2COLOR,  
    HI_TDE_BLEND_SRC2ALPHA,  
    HI_TDE_BLEND_INVSRC2ALPHA,  
    HI_TDE_BLEND_SRC1COLOR,  
    HI_TDE_BLEND_MAX
```

```
HI_TDE_BLEND_INVSRC1COLOR,  
HI_TDE_BLEND_SRC1ALPHA,  
HI_TDE_BLEND_INVSRC1ALPHA,  
HI_TDE_BLEND_SRC2ALPHASAT,  
HI_TDE_BLEND_MAX  
} hi_tde_blend_mode;
```

成员

描述中有以下定义

ff: 前景图混合系数, 范围[0, 1]

bf: 背景图混合系数, 范围[0, 1]

fa: 前景图透明度, 范围[0, 1]

ba: 背景图透明度, 范围[0, 1]

成员名称	描述
HI_TDE_BLEND_ZERO	混合系数取0。
HI_TDE_BLEND_ONE	混合系数取1。
HI_TDE_BLEND_SRC2COLOR	混合系数取fc。
HI_TDE_BLEND_INVSRC2COLOR	混合系数取1-fc。
HI_TDE_BLEND_SRC2ALPHA	混合系数取fa。
HI_TDE_BLEND_INVSRC2ALPHA	混合系数取1- fa。
HI_TDE_BLEND_SRC1COLOR	混合系数取bc。
HI_TDE_BLEND_INVSRC1COLOR	混合系数取1-bc。
HI_TDE_BLEND_SRC1ALPHA	混合系数取ba。
HI_TDE_BLEND_INVSRC1ALPHA	混合系数取1-ba。
HI_TDE_BLEND_SRC2ALPHASAT	混合系数取min(1 - ba, fa) + 1。
HI_TDE_BLEND_MAX	无效的alpha混合模式。

1.9.20.9.10 hi_tde_rop_mode

说明

TDE支持的ROP操作类型。

定义

```
typedef enum {  
    HI_TDE_ROP_BLACK = 0, /* Blackness */  
    HI_TDE_ROP_NOTMERGEPEN, /* ~(S2 | S1) */  
    HI_TDE_ROP_MASKNOTPEN, /* ~S2&S1 */  
    HI_TDE_ROP_NOTCOPYPEN, /* ~S2 */  
    HI_TDE_ROP_MASKPENNOT, /* S2&~S1 */  
    HI_TDE_ROP_NOT, /* ~S1 */  
    HI_TDE_ROP_XORPEN, /* S2^S1 */  
    HI_TDE_ROP_NOTMASKPEN, /* ~(S2 & S1) */  
    HI_TDE_ROP_MASKPEN, /* S2&S1 */  
    HI_TDE_ROP_NOTXORPEN, /* ~(S2^S1) */  
    HI_TDE_ROP_NOP, /* S1 */  
    HI_TDE_ROP_MERGEENOTPEN, /* ~S2|S1 */  
    HI_TDE_ROP_COPYPEN, /* S2 */  
    HI_TDE_ROP_MERGEENOT, /* S2|~S1 */  
    HI_TDE_ROP_MERGEEN, /* S2|S1 */  
    HI_TDE_ROP_WHITE, /* Whiteness */  
    HI_TDE_ROP_MAX  
} hi_tde_rop_mode;
```

成员

描述中有以下定义：S1表示位图1，S2表示位图2。

成员名称	描述
HI_TDE_ROP_BLACK	黑色。Atlas 200/500 A2推理产品不支持。
HI_TDE_ROP_NOTMERGEPEN	~(S1+S2) 。Atlas 200/500 A2推理产品不支持。
HI_TDE_ROP_MASKNOTPEN	~S2&S1。Atlas 200/500 A2推理产品不支持。
HI_TDE_ROP_NOTCOPYPEN	~S2。Atlas 200/500 A2推理产品不支持。
HI_TDE_ROP_MASKPENNOT	S2&~S1。Atlas 200/500 A2推理产品不支持。
HI_TDE_ROP_NOT	~S1。Atlas 200/500 A2推理产品不支持。
HI_TDE_ROP_XORPEN	S2^S1。Atlas 200/500 A2推理产品不支持。
HI_TDE_ROP_NOTMASKPEN	~(S2^S1) 。Atlas 200/500 A2推理产品不支持。
HI_TDE_ROP_MASKPEN	S2&S1。Atlas 200/500 A2推理产品不支持。
HI_TDE_ROP_NOTXORPEN	~(S2^S1)。Atlas 200/500 A2推理产品不支持。

成员名称	描述
HI_TDE_ROP_NOP	S1。Atlas 200/500 A2推理产品不支持。
HI_TDE_ROP_MERGENOTPEN	~S2 + S1。Atlas 200/500 A2推理产品不支持。
HI_TDE_ROP_COPYPEN	S2。Atlas 200/500 A2推理产品不支持。
HI_TDE_ROP_MERGEPENNOT	S2 + ~S1。Atlas 200/500 A2推理产品不支持。
HI_TDE_ROP_MERGEPEN	S2 + S1。Atlas 200/500 A2推理产品不支持。
HI_TDE_ROP_WHITE	白色。Atlas 200/500 A2推理产品不支持。
HI_TDE_ROP_MAX	无效ROP类型。Atlas 200/500 A2推理产品不支持。

1.9.20.9.11 hi_tde_clip_mode

说明

TDE裁剪的配置属性。

定义

```
typedef enum {  
    HI_TDE_CLIP_MODE_NONE = 0,  
    HI_TDE_CLIP_MODE_INSIDE,  
    HI_TDE_CLIP_MODE_OUTSIDE,  
    HI_TDE_CLIP_MODE_MAX  
} hi_tde_clip_mode;
```

成员

描述中有以下定义：S1表示位图1，S2表示位图2。

成员名称	描述
HI_TDE_CLIP_MODE_NONE	输出结果不做剪切。Atlas 200/500 A2推理产品不支持。
HI_TDE_CLIP_MODE_INSIDE	区域内剪切模式。Atlas 200/500 A2推理产品不支持。
HI_TDE_CLIP_MODE_OUTSIDE	区域外剪切模式。Atlas 200/500 A2推理产品不支持。
HI_TDE_CLIP_MODE_MAX	无效剪切模式。Atlas 200/500 A2推理产品不支持。

1.9.20.9.12 hi_tde_colorkey_mode

说明

TDE colorkey的配置属性。

定义

```
typedef enum {  
    HI_TDE_COLORKEY_MODE_NONE = 0,  
    HI_TDE_COLORKEY_MODE_FG,  
    HI_TDE_COLORKEY_MODE_BG,  
    HI_TDE_COLORKEY_MODE_MAX  
} hi_tde_colorkey_mode;
```

成员

成员名称	描述
HI_TDE_COLORKEY_MODE_NONE	不做colorkey操作。Atlas 200/500 A2推理产品不支持。
HI_TDE_COLORKEY_MODE_FG	对前景位图进行colorkey操作。Atlas 200/500 A2推理产品不支持。
HI_TDE_COLORKEY_MODE_BG	对背景位图进行colorkey操作。Atlas 200/500 A2推理产品不支持。
HI_TDE_COLORKEY_MODE_MAX	无效的colorkey模式。Atlas 200/500 A2推理产品不支持。

1.9.20.9.13 hi_tde_colorkey_component

说明

TDE单个颜色分量的关键色属性。

定义

```
typedef struct {  
    hi_u8 min_component;  
    hi_u8 max_component;  
    hi_u8 is_component_out;  
    hi_u8 is_component_ignore;  
    hi_u8 component_mask;  
    hi_u8 component_reserved;  
    hi_u8 component_reserved1;  
    hi_u8 component_reserved2;  
} hi_tde_colorkey_component;
```

成员

成员名称	描述
min_component	分量最小值。Atlas 200/500 A2推理产品不支持。

成员名称	描述
max_component	分量最大值。Atlas 200/500 A2推理产品不支持。
is_component_out	分量关键色在范围内或范围外。Atlas 200/500 A2推理产品不支持。
is_component_ignore	分量是否忽略。Atlas 200/500 A2推理产品不支持。
component_mask	分量掩码。Atlas 200/500 A2推理产品不支持。
component_reserved	保留值。Atlas 200/500 A2推理产品不支持。
component_reserved1	保留值。Atlas 200/500 A2推理产品不支持。
component_reserved2	保留值。Atlas 200/500 A2推理产品不支持。

1.9.20.9.14 hi_tde_colorkey

说明

TDE关键色的配置属性。

定义

```
typedef union {
    struct {
        hi_tde_colorkey_component alpha;
        hi_tde_colorkey_component red;
        hi_tde_colorkey_component green;
        hi_tde_colorkey_component blue;
    } argb_colorkey;
    struct {
        hi_tde_colorkey_component alpha;
        hi_tde_colorkey_component y;
        hi_tde_colorkey_component cb;
        hi_tde_colorkey_component cr;
    } ycbcr_colorkey;
    struct {
        hi_tde_colorkey_component alpha;
        hi_tde_colorkey_component clut;
    } clut_colorkey;
} hi_tde_colorkey;
```

成员

成员名称	描述
argb_colorkey.alpha	argb alpha 分量关键色属性。Atlas 200/500 A2推理产品不支持。
argb_colorkey.red	argb red 分量关键色属性。Atlas 200/500 A2推理产品不支持。
argb_colorkey.green	argb green 分量关键色属性。Atlas 200/500 A2推理产品不支持。

成员名称	描述
argb_colorkey.blue	argb r blue 分量关键色属性。Atlas 200/500 A2推理产品不支持。
ycbcr_colorkey.y	YCbCr Y 分量关键色属性。Atlas 200/500 A2推理产品不支持。
ycbcr_colorkey.cb	YCbCr Cb 分量关键色属性。Atlas 200/500 A2推理产品不支持。
ycbcr_colorkey.cr	YCbCr Cr 分量关键色属性。Atlas 200/500 A2推理产品不支持。
clut_colorkey.alpha	CLUT alpha 分量关键色属性。Atlas 200/500 A2推理产品不支持。
clut_colorkey.clut	CLUT CLUT 分量关键色属性。Atlas 200/500 A2推理产品不支持。

1.9.20.9.15 hi_tde_blend_opt

说明

模式填充混合选项结构体。

定义

```
typedef struct {  
    hi_bool global_alpha_en;  
    hi_bool pixel_alpha_en;  
    hi_bool src1_alpha_premulti;  
    hi_bool src2_alpha_premulti;  
    hi_tde_blend_cmd blend_cmd;  
    hi_tde_blend_mode src1_blend_mode;  
    hi_tde_blend_mode src2_blend_mode;  
} hi_tde_blend_opt;
```

成员

成员名称	描述
global_alpha_en	全局alpha是否使能。
pixel_alpha_en	像素alpha是否使能。
src1_alpha_premulti	src1 alpha预乘是否使能。
src2_alpha_premulti	src2 alpha预乘是否使能。
blend_cmd	Alpha 叠加指令。
src1_blend_mode	设置src1_blend模式。
src2_blend_mode	设置src2_blend模式。

1.9.20.9.16 hi_tde_alpha_blending

说明

透明度混合枚举。

定义

```
typedef enum {  
    HI_TDE_ALPHA_BLENDING_NONE = 0x0,  
    HI_TDE_ALPHA_BLENDING_BLEND = 0x1,  
    HI_TDE_ALPHA_BLENDING_ROP = 0x2,  
    HI_TDE_ALPHA_BLENDING_COLORIZE = 0x4,  
    HI_TDE_ALPHA_BLENDING_MAX = 0x8  
} hi_tde_alpha_blending;
```

成员

成员名称	描述
HI_TDE_ALPHA_BLENDING_NONE	不使用混合操作。
HI_TDE_ALPHA_BLENDING_BLEND	Alpha通道混合操作选项。
HI_TDE_ALPHA_BLENDING_ROP	混合ROP选项。Atlas 200/500 A2推理产品不支持。
HI_TDE_ALPHA_BLENDING_COLORIZE	混合colorize选项。Atlas 200/500 A2推理产品不支持。

1.9.20.9.17 hi_tde_out_alpha_from

说明

输出图像的alpha来源类型。

定义

```
typedef enum {  
    HI_TDE_OUT_ALPHA_FROM_NORM = 0, /* Output from the result of alpha blending or anti-flicker */  
    HI_TDE_OUT_ALPHA_FROM_BG, /* Output from the background bitmap */  
    HI_TDE_OUT_ALPHA_FROM_FG, /* Output from the foreground bitmap */  
    HI_TDE_OUT_ALPHA_FROM_GLOBALALPHA, /* Output from the global alpha */  
    HI_TDE_OUT_ALPHA_FROM_MAX  
} hi_tde_out_alpha_from;
```

成员

成员名称	描述
HI_TDE_OUT_ALPHA_FROM_NORM	输出图像的alpha来源于alpha blending结果。

成员名称	描述
HI_TDE_OUT_ALPHA_FROM_BG	输出图像alpha来源于背景位图。
HI_TDE_OUT_ALPHA_FROM_FG	输出图像alpha来源于前景位图。
HI_TDE_OUT_ALPHA_FROM_GLOBALALPHA	输出图像alpha来源于全局alpha。

1.9.20.9.18 hi_tde_csc_opt

说明

CSC参数选项。

定义

```
typedef struct {  
    hi_bool src_csc_user_en; /* User-defined ICSC parameter enable */  
    hi_bool src_csc_param_reload_en; /* User-defined ICSC parameter reload enable */  
    hi_bool dst_csc_user_en; /* User-defined OCSC parameter enable */  
    hi_bool dst_csc_param_reload_en; /* User-defined OCSC parameter reload enable */  
    hi_u64 src_csc_param_addr; /* ICSC parameter address. The address must be 128-bit aligned. */  
    hi_s32 src_csc_param_len;  
    hi_u64 dst_csc_param_addr; /* OCSC parameter address. The address must be 128-bit aligned. */  
    hi_s32 dst_csc_param_len;  
} hi_tde_csc_opt;
```

成员

成员名称	描述
src_csc_user_en	用户自定义ICSC参数使能。Atlas 200/500 A2推理产品不支持。
src_csc_param_reload_en	重新加载用户自定义ICSC参数使能。Atlas 200/500 A2推理产品不支持。
dst_csc_user_en	用户自定义OCSC参数使能。Atlas 200/500 A2推理产品不支持。
dst_csc_param_reload_en	重新加载用户自定义OCSC参数使能。Atlas 200/500 A2推理产品不支持。
src_csc_param_addr	ICSC参数地址，要求128bit对齐。Atlas 200/500 A2推理产品不支持。
src_csc_param_len	ICSC参数长度。Atlas 200/500 A2推理产品不支持。
dst_csc_param_addr	OCSC参数地址，要求128bit对齐。Atlas 200/500 A2推理产品不支持。
dst_csc_param_len	OCSC参数长度。Atlas 200/500 A2推理产品不支持。

1.9.20.9.19 hi_tde_pattern_fill_opt

说明

模式填充选项操作，在模式填充功能结构中使用。

定义

```
typedef struct {
    hi_tde_alpha_blending alpha_blending_cmd;
    hi_tde_rop_mode rop_color;
    hi_tde_rop_mode rop_alpha;
    hi_tde_colorkey_mode colorkey_mode;
    hi_tde_colorkey colorkey_value;
    hi_tde_clip_mode clip_mode;
    hi_tde_rect clip_rect;
    hi_bool clut_reload;
    hi_u8 global_alpha;
    hi_tde_out_alpha_from out_alpha_from;
    hi_u32 color_resize;
    hi_tde_blend_opt blend_opt;
    hi_tde_csc_opt csc_opt;
} hi_tde_pattern_fill_opt;
```

成员

成员名称	描述
alpha_blending_cmd	逻辑操作类型。
rop_color	颜色空间的rop类型。Atlas 200/500 A2推理产品不支持。
rop_alpha	alpha分量的rop类型。Atlas 200/500 A2推理产品不支持。
colorkey_mode	Colorkey模式。Atlas 200/500 A2推理产品不支持。
colorkey_value	Colorkey值。Atlas 200/500 A2推理产品不支持。
clip_mode	裁剪模式。Atlas 200/500 A2推理产品不支持。
clip_rect	裁剪区域。Atlas 200/500 A2推理产品不支持。
clut_reload	是否重载clut。Atlas 200/500 A2推理产品不支持。
global_alpha	全局alpha值。
out_alpha_from	输出alpha的源。
color_resize	Colorize的值。Atlas 200/500 A2推理产品不支持。
blend_opt	混合选项。
csc_opt	csc选项。Atlas 200/500 A2推理产品不支持。

1.9.20.10 HIFB 叠加图形层管理

1.9.20.10.1 fb_var_screeninfo

说明

定义HIFB的可变屏幕信息，此为Linux FrameBuffer原生结构体。

定义

```
struct fb_var_screeninfo {
    __u32 xres;          /* visible resolution */
    __u32 yres;          /* visible resolution */
    __u32 xres_virtual;  /* virtual resolution */
    __u32 yres_virtual;  /* virtual resolution */
    __u32 xoffset;       /* offset from virtual to visible */
    __u32 yoffset;       /* resolution */
    __u32 bits_per_pixel;
    __u32 grayscale;     /* 0 = color, 1 = grayscale, */
                        /* >1 = FOURCC */
    struct fb_bitfield red; /* bitfield in fb mem if true color, */
    struct fb_bitfield green; /* else only length is significant */
    struct fb_bitfield blue;
    struct fb_bitfield transp; /* transparency */
    __u32 nonstd;         /* != 0 Non standard pixel format */
    __u32 activate;       /* see FB_ACTIVATE_* */
    __u32 height;         /* height of picture in mm */
    __u32 width;          /* width of picture in mm */
    __u32 accel_flags;     /* (OBSOLETE) see fb_info.flags */

    /* Timing: All values in pixclocks, except pixclock (of course) */
    __u32 pixclock;       /* pixel clock in ps (pico seconds) */
    __u32 left_margin;    /* time from sync to picture */
    __u32 right_margin;   /* time from picture to sync */
    __u32 upper_margin;   /* time from sync to picture */
    __u32 lower_margin;
    __u32 hsync_len;      /* length of horizontal sync */
    __u32 vsync_len;      /* length of vertical sync */
    __u32 sync;           /* see FB_SYNC_* */
    __u32 vmode;          /* see FB_VMODE_* */
    __u32 rotate;         /* angle we rotate counter clockwise */
    __u32 colorspace;     /* colorspace for FOURCC-based modes */
    __u32 reserved[4];    /* Reserved for future compatibility */
};
```

成员

G0、G1图层，下表中的Gx_min、Gy_min为32；G2、G3、G4图层，下表中的Gx_min、Gy_min为8。Gx_min表示当前图层支持的最小宽度，Gy_min表示当前图层支持的最小高度。

成员名称	描述
xres	可见屏幕宽度（像素数），需要为偶数。 取值范围：[Gx_min, xres_virtual]。
yres	可见屏幕高度（像素数），需要为偶数。 取值范围：[Gy_min, yres_virtual]。
xres_virtual	虚拟屏幕宽度（显存中图像宽度）。 取值范围：大于或等于Gx_min。

成员名称	描述
yres_virtual	虚拟屏幕高度（显存中图像高度）。 取值范围：大于或等于Gy_min。
xoffset	在x方向上的偏移像素数。 取值范围：[0, xres_virtual - xres]。
yoffset	在y方向上的偏移像素数。 取值范围：[0, yres_virtual - yres]
bits_per_pixel	每个像素所占的比特数。 取值范围：32/16/4/2 注：4和2分别对应CLUT4和CLUT2模式，需要使能CMAP功能。
red	颜色分量中红色的位域信息。
green	颜色分量中绿色的位域信息。
blue	颜色分量中蓝色的位域信息。
transp	颜色分量中alpha分量的位域信息。
grayscale	颜色模式，当前版本不支持。
nonstd	是否为标准像素格式，当前版本不支持。 ● 0：是 ● 1：否
activate	设置生效的时刻，当前版本不支持。
height	屏幕高，单位为mm，当前版本不支持。
width	屏幕宽，单位为mm，当前版本不支持。
accel_flags	加速标志，当前版本不支持。
pixclock	显示一个点需要的时间，单位为ns，当前版本不支持。
left_margin	左消隐信号，当前版本不支持。
right_margin	右消隐信号，当前版本不支持。
upper_margin	上消隐信号，当前版本不支持。
lower_margin	下消隐信号，当前版本不支持。
hsync_len	水平同步时长，当前版本不支持。
vsync_len	垂直同步时长，当前版本不支持。
sync	同步信号方式，当前版本不支持。
vmode	扫描模式，当前版本不支持。
rotate	顺时针旋转的角度，当前版本不支持。

成员名称	描述
colorspace	FOURCC模式的色彩空间，当前版本不支持。
reserved[4]	预留参数，当前版本不支持。

1.9.20.10.2 fb_fix_screeninfo

说明

定义HIFB的固定屏幕信息，此为Linux FrameBuffer原生结构体。

定义

```
struct fb_fix_screeninfo {
    char id[16];          /* identification string eg "TT Builtin" */
    unsigned long smem_start; /* Start of frame buffer mem */
                          /* (physical address) */
    __u32 smem_len;       /* Length of frame buffer mem */
    __u32 type;           /* see FB_TYPE_* */
    __u32 type_aux;       /* Interleave for interleaved Planes */
    __u32 visual;         /* see FB_VISUAL_* */
    __u16 xpanstep;       /* zero if no hardware panning */
    __u16 ypanstep;       /* zero if no hardware panning */
    __u16 ywrapstep;      /* zero if no hardware ywrap */
    __u32 line_length;    /* length of a line in bytes */
    unsigned long mmio_start; /* Start of Memory Mapped I/O */
                          /* (physical address) */
    __u32 mmio_len;       /* Length of Memory Mapped I/O */
    __u32 accel;          /* Indicate to driver which */
                          /* specific chip/card we have */
    __u16 capabilities;   /* see FB_CAP_* */
    __u16 reserved[2];    /* Reserved for future compatibility */
};
```

成员

成员名称	描述
id[16]	设备驱动名称。
smem_start	显存起始内核态虚拟地址。
smem_len	显存大小，单位：字节。
type	显存类型，当前版本不支持。
type_aux	附加类型，当前版本不支持。
visual	色彩模式，当前版本不支持。
xpanstep	支持水平方向上的PAN显示： 0：不支持。 - 非0：支持，此时该值用于表示在水平方向上每步进的像素值。 当前版本不支持。

成员名称	描述
ypanstep	支持垂直方向上的PAN显示： 0：不支持。 - 非0：支持，此时该值用于表示在垂直方向上每步进的像素值。 当前版本不支持。
ywrapstep	类似于ypanstep，不同之处在于：当起显示当底部时，能回到显存的开始处进行显示。 当前版本不支持。
line_length	每行字节数，当前版本不支持。
mmio_start	显存起始用户态虚拟地址。
mmio_len	显存大小，单位：字节。
accel	显示所支持的硬件加速设备，当前版本不支持。
capabilities	预留参数，当前版本不支持。
reserved[2]	预留参数，当前版本不支持。

1.9.20.10.3 fb_cmap

说明

定义HIFB的颜色表信息，此为Linux FrameBuffer原生结构体。

定义

```
struct fb_cmap {
    __u32 start;          /* First entry */
    __u32 len;            /* Number of entries */
    __u16 *red;           /* Red values */
    __u16 *green;
    __u16 *blue;
    __u16 *transp;       /* transparency, can be NULL */
};
```

成员

成员名称	描述
start	颜色表入口索引。
len	颜色表长度（单分量颜色表数量）。 最大长度为256。
red	红色分量表。
green	绿色分量表。

成员名称	描述
blue	蓝色分量表。
transp	Alpha分量表。

1.9.20.10.4 hi_fb_point

说明

定义HIFB的起始坐标信息。

定义

```
typedef struct {  
    hi_s32 x_pos;  
    hi_s32 y_pos;  
} hi_fb_point;
```

成员

成员名称	描述
x_pos	起始点水平坐标。
y_pos	起始点垂直坐标。

1.9.20.10.5 hi_fb_alpha

说明

定义HIFB Alpha结构体。

定义

```
typedef struct {  
    hi_bool alpha_en;  
    hi_bool alpha_chn_en;  
    hi_u8 alpha0;  
    hi_u8 alpha1;  
    hi_u8 global_alpha;  
    hi_u8 reserved;  
} hi_fb_alpha;
```

成员

成员名称	描述
alpha_en	Alpha叠加使能。
alpha_chn_en	Alpha通道叠加使能。
alpha0	Alpha通道为0时选取的Alpha值，当前版本不支持。

成员名称	描述
alpha1	Alpha通道为1时选取的Alpha值，当前版本不支持。
global_alpha	全局Alpha值，范围为0~255，默认为255。在Alpha通道使能时起作用。
reserved	预留参数，当前版本不支持。

1.9.20.10.6 fb_bitfield

说明

位域信息，用于设置像素格式，此为Linux FrameBuffer原生结构体。

定义

```
struct fb_bitfield {
    __u32 offset;      /* beginning of bitfield */
    __u32 length;      /* length of bitfield */
    __u32 msb_right;    /* != 0 : Most significant bit is */
                      /* right */
};
```

成员

成员名称	描述
offset	颜色分量的偏移。
length	颜色分量所占bit位数，
msb_right	0：最高位在左侧。 - 非0：最高位在右侧。

1.9.20.11 HDMI 外设

1.9.20.11.1 hi_hdmi_id

说明

定义HDMI设备号。

定义

```
typedef enum {
    HI_HDMI_ID_0,
    HI_HDMI_ID_1,
    HI_HDMI_ID_BUTT
} hi_hdmi_id;
```

成员

成员名称	描述
HI_HDMI_ID_0	HDMI设备0。
HI_HDMI_ID_1	HDMI设备1。
HI_HDMI_ID_BUTT	预留值。

1.9.20.11.2 hi_hdmi_video_format

说明

定义HDMI视频格式。

定义

```
typedef enum {  
    HI_HDMI_VIDEO_FORMAT_1080P_60,  
    HI_HDMI_VIDEO_FORMAT_1080P_50,  
    HI_HDMI_VIDEO_FORMAT_1080P_30,  
    HI_HDMI_VIDEO_FORMAT_1080P_25,  
    HI_HDMI_VIDEO_FORMAT_1080P_24,  
    HI_HDMI_VIDEO_FORMAT_1080i_60,  
    HI_HDMI_VIDEO_FORMAT_1080i_50,  
    HI_HDMI_VIDEO_FORMAT_720P_60,  
    HI_HDMI_VIDEO_FORMAT_720P_50,  
    HI_HDMI_VIDEO_FORMAT_576P_50,  
    HI_HDMI_VIDEO_FORMAT_480P_60,  
    HI_HDMI_VIDEO_FORMAT_PAL,  
    HI_HDMI_VIDEO_FORMAT_NTSC,  
    HI_HDMI_VIDEO_FORMAT_861D_640X480_60,  
    HI_HDMI_VIDEO_FORMAT_VESA_800X600_60,  
    HI_HDMI_VIDEO_FORMAT_VESA_1024X768_60,  
    HI_HDMI_VIDEO_FORMAT_VESA_1280X800_60,  
    HI_HDMI_VIDEO_FORMAT_VESA_1280X1024_60,  
    HI_HDMI_VIDEO_FORMAT_VESA_1366X768_60,  
    HI_HDMI_VIDEO_FORMAT_VESA_1440X900_60,  
    HI_HDMI_VIDEO_FORMAT_VESA_1400X1050_60,  
    HI_HDMI_VIDEO_FORMAT_VESA_1600X1200_60,  
    HI_HDMI_VIDEO_FORMAT_VESA_1680X1050_60,  
    HI_HDMI_VIDEO_FORMAT_VESA_1920X1200_60,  
    HI_HDMI_VIDEO_FORMAT_2560x1440_30,  
    HI_HDMI_VIDEO_FORMAT_2560x1440_60,  
    HI_HDMI_VIDEO_FORMAT_2560x1600_60,  
    HI_HDMI_VIDEO_FORMAT_1920x2160_30,  
    HI_HDMI_VIDEO_FORMAT_3840X2160P_24,  
    HI_HDMI_VIDEO_FORMAT_3840X2160P_25,  
    HI_HDMI_VIDEO_FORMAT_3840X2160P_30,  
    HI_HDMI_VIDEO_FORMAT_3840X2160P_50,  
    HI_HDMI_VIDEO_FORMAT_3840X2160P_60,  
    HI_HDMI_VIDEO_FORMAT_4096X2160P_24,  
    HI_HDMI_VIDEO_FORMAT_4096X2160P_25,  
    HI_HDMI_VIDEO_FORMAT_4096X2160P_30,  
    HI_HDMI_VIDEO_FORMAT_4096X2160P_50,  
    HI_HDMI_VIDEO_FORMAT_4096X2160P_60,  
    HI_HDMI_VIDEO_FORMAT_3840X2160P_120,  
    HI_HDMI_VIDEO_FORMAT_4096X2160P_120,  
    HI_HDMI_VIDEO_FORMAT_7680X4320P_30,  
    HI_HDMI_VIDEO_FORMAT_VESA_CUSTOMER_DEFINE,  
    HI_HDMI_VIDEO_FORMAT_BUTT  
} hi_hdmi_video_format;
```

成员

成员名称	描述
HI_HDMI_VIDEO_FORMAT_1080P_60	1080P60帧。
HI_HDMI_VIDEO_FORMAT_1080P_50	1080P50帧。
HI_HDMI_VIDEO_FORMAT_1080P_30	1080P 30帧。
HI_HDMI_VIDEO_FORMAT_1080P_25	1080P 25帧。当前版本不支持。
HI_HDMI_VIDEO_FORMAT_1080P_24	1080P 24帧。当前版本不支持。
HI_HDMI_VIDEO_FORMAT_1080i_60	1080I 60帧。当前版本不支持。
HI_HDMI_VIDEO_FORMAT_1080i_50	1080I 50帧。当前版本不支持。
HI_HDMI_VIDEO_FORMAT_720P_60	720P 60帧。
HI_HDMI_VIDEO_FORMAT_720P_50	720P 50帧。当前版本不支持。
HI_HDMI_VIDEO_FORMAT_576P_50	576P 50帧。当前版本不支持。
HI_HDMI_VIDEO_FORMAT_480P_60	480P 60帧。
HI_HDMI_VIDEO_FORMAT_PAL	PAL (Phase Alternating Line) 制式。当前版本不支持。
HI_HDMI_VIDEO_FORMAT_NTSC	NTSC (National Television System Committee) 制式。当前版本不支持。
HI_HDMI_VIDEO_FORMAT_861D_640X480_60	640x480 60帧 。当前版本不支持。
HI_HDMI_VIDEO_FORMAT_VESA_800X600_60	800x600 60帧 。
HI_HDMI_VIDEO_FORMAT_VESA_1024X768_60	1024x768 60帧。当前版本不支持。
HI_HDMI_VIDEO_FORMAT_VESA_1280X800_60	1280x800 60帧。当前版本不支持。
HI_HDMI_VIDEO_FORMAT_VESA_1280X1024_60	1280x1024 60帧。当前版本不支持。

成员名称	描述
HI_HDMI_VIDEO_FORMAT_VESA_1366X768_60	1366x768 60帧。当前版本不支持。
HI_HDMI_VIDEO_FORMAT_VESA_1440X900_60	1440x900 60帧。当前版本不支持。
HI_HDMI_VIDEO_FORMAT_VESA_1400X1050_60	1400x1050 60帧。当前版本不支持。
HI_HDMI_VIDEO_FORMAT_VESA_1600X1200_60	1600x1200 60帧。
HI_HDMI_VIDEO_FORMAT_VESA_1680X1050_60	1680x1050 60帧。当前版本不支持。
HI_HDMI_VIDEO_FORMAT_VESA_1920X1200_60	1920x1200 60帧。
HI_HDMI_VIDEO_FORMAT_2560x1440_30	2560x1440 30帧。当前版本不支持。
HI_HDMI_VIDEO_FORMAT_2560x1440_60	2560x1440 60帧。当前版本不支持。
HI_HDMI_VIDEO_FORMAT_2560x1600_60	2560x1600 30帧。当前版本不支持。
HI_HDMI_VIDEO_FORMAT_1920x2160_30	1920x2160 30帧。当前版本不支持。
HI_HDMI_VIDEO_FORMAT_3840X2160P_24	3840x2160 24帧。
HI_HDMI_VIDEO_FORMAT_3840X2160P_25	3840x2160 25帧。
HI_HDMI_VIDEO_FORMAT_3840X2160P_30	3840x2160 30帧。
HI_HDMI_VIDEO_FORMAT_3840X2160P_50	3840x2160 50帧。
HI_HDMI_VIDEO_FORMAT_3840X2160P_60	3840x2160 60帧。
HI_HDMI_VIDEO_FORMAT_4096X2160P_24	4096x2160 24帧。当前版本不支持。
HI_HDMI_VIDEO_FORMAT_4096X2160P_25	4096x2160 25帧。当前版本不支持。
HI_HDMI_VIDEO_FORMAT_4096X2160P_30	4096x2160 30帧。当前版本不支持。
HI_HDMI_VIDEO_FORMAT_4096X2160P_50	4096x2160 50帧。当前版本不支持。

成员名称	描述
HI_HDMI_VIDEO_FORMAT_4096X2160P_60	4096x2160 60帧。当前版本不支持。
HI_HDMI_VIDEO_FORMAT_3840X2160P_120	3840X2160 120帧。当前版本不支持。
HI_HDMI_VIDEO_FORMAT_4096X2160P_120	4096x2160 120帧。当前版本不支持。
HI_HDMI_VIDEO_FORMAT_7680X4320P_30	7680x4320 30帧。当前版本不支持。
HI_HDMI_VIDEO_FORMAT_VESA_CUSTOMER_DEFINE	用户配置规格。
HI_HDMI_VIDEO_FORMAT_BUTT	预留值。

1.9.20.11.3 hi_hdmi_deep_color

说明

定义HDMI深色模式。

定义

```
typedef enum {  
    HI_HDMI_DEEP_COLOR_24BIT,  
    HI_HDMI_DEEP_COLOR_30BIT,  
    HI_HDMI_DEEP_COLOR_36BIT,  
    HI_HDMI_DEEP_COLOR_BUTT  
} hi_hdmi_deep_color;
```

成员

成员名称	描述
HI_HDMI_DEEP_COLOR_24BIT	HDMI Deep Color 24bit 模式
HI_HDMI_DEEP_COLOR_30BIT	HDMI Deep Color 30bit 模式。当前版本不支持。
HI_HDMI_DEEP_COLOR_36BIT	HDMI Deep Color 36bit 模式。当前版本不支持。
HI_HDMI_DEEP_COLOR_BUTT	预留值。

1.9.20.11.4 hi_hdmi_sample_rate

说明

定义 HDMI 音频输出采样率。

定义

```
typedef enum {  
    HI_HDMI_SAMPLE_RATE_UNKNOWN,  
    HI_HDMI_SAMPLE_RATE_8K,  
    HI_HDMI_SAMPLE_RATE_11K,  
    HI_HDMI_SAMPLE_RATE_12K,  
    HI_HDMI_SAMPLE_RATE_16K,  
    HI_HDMI_SAMPLE_RATE_22K,  
    HI_HDMI_SAMPLE_RATE_24K,  
    HI_HDMI_SAMPLE_RATE_32K,  
    HI_HDMI_SAMPLE_RATE_44K,  
    HI_HDMI_SAMPLE_RATE_48K,  
    HI_HDMI_SAMPLE_RATE_88K,  
    HI_HDMI_SAMPLE_RATE_96K,  
    HI_HDMI_SAMPLE_RATE_176K,  
    HI_HDMI_SAMPLE_RATE_192K,  
    HI_HDMI_SAMPLE_RATE_768K,  
    HI_HDMI_SAMPLE_RATE_BUTT  
} hi_hdmi_sample_rate;
```

成员

成员名称	描述
HI_HDMI_SAMPLE_RATE_8K	8K采样率。当前版本不支持。
HI_HDMI_SAMPLE_RATE_11K	11K采样率。当前版本不支持。
HI_HDMI_SAMPLE_RATE_12K	12K采样率。当前版本不支持。
HI_HDMI_SAMPLE_RATE_16K	16K采样率。当前版本不支持。
HI_HDMI_SAMPLE_RATE_22K	22K采样率。当前版本不支持。
HI_HDMI_SAMPLE_RATE_24K	24K采样率。当前版本不支持。
HI_HDMI_SAMPLE_RATE_32K	32K采样率。当前版本不支持。
HI_HDMI_SAMPLE_RATE_44K	44K采样率。当前版本不支持。
HI_HDMI_SAMPLE_RATE_48K	48K采样率。
HI_HDMI_SAMPLE_RATE_88K	88K采样率。当前版本不支持。

成员名称	描述
HI_HDMI_SAMPLE_RATE_96K	96K采样率。当前版本不支持。
HI_HDMI_SAMPLE_RATE_176K	176K采样率。当前版本不支持。
HI_HDMI_SAMPLE_RATE_192K	192K采样率。当前版本不支持。
HI_HDMI_SAMPLE_RATE_768K	768K采样率。当前版本不支持。
HI_HDMI_SAMPLE_RATE_BUTT	预留值。

1.9.20.11.5 hi_hdmi_bit_depth

说明

定义HDMI音频输出采样位宽。

定义

```
typedef enum {  
    HI_HDMI_BIT_DEPTH_UNKNOWN,  
    HI_HDMI_BIT_DEPTH_8,  
    HI_HDMI_BIT_DEPTH_16,  
    HI_HDMI_BIT_DEPTH_18,  
    HI_HDMI_BIT_DEPTH_20,  
    HI_HDMI_BIT_DEPTH_24,  
    HI_HDMI_BIT_DEPTH_32,  
    HI_HDMI_BIT_DEPTH_BUTT  
} hi_hdmi_bit_depth;
```

成员

成员名称	描述
HI_HDMI_BIT_DEPTH_8	8bit采样位宽。当前版本不支持。
HI_HDMI_BIT_DEPTH_16	16bit采样位宽。
HI_HDMI_BIT_DEPTH_18	18bit采样位宽。当前版本不支持。
HI_HDMI_BIT_DEPTH_20	20bit采样位宽。当前版本不支持。
HI_HDMI_BIT_DEPTH_24	24bit采样位宽。当前版本不支持。
HI_HDMI_BIT_DEPTH_32	32bit采样位宽。当前版本不支持。
HI_HDMI_BIT_DEPTH_BUTT	预留值。

1.9.20.11.6 hi_hdmi_attr

说明

定义HDMI输出属性结构体。

定义

```
typedef struct {  
    hi_bool hdmi_en;  
    hi_hdmi_video_format video_format;  
    hi_hdmi_deep_color deep_color_mode;  
    hi_bool audio_en;  
    hi_hdmi_sample_rate sample_rate;  
    hi_hdmi_bit_depth bit_depth;  
    hi_bool auth_mode_en;  
    hi_bool deep_color_adapt_en;  
    hi_u32 pix_clk;  
} hi_hdmi_attr;
```

成员

成员名称	描述
hdmi_en	是否强制HDMI输出。 - HI_TRUE：强制HDMI输出 - HI_FALSE：DVI输出 当前版本需设置为HI_TRUE
video_format	视频制式。建议用户设置为Sink能力集支持的制式。当前版本中此参数仅支持设置为HI_HDMI_VIDEO_FORMAT_VESA_CUSTOMER_DEFINE
deep_color_mode	DeepColor输出模式。 默认为HI_HDMI_DEEP_COLOR_24BIT（HI_HDMI_DEEP_COLOR_OFF）。 部分Sink 不支持HI_HDMI_DEEP_COLOR_30BIT 和 HI_HDMI_DEEP_COLOR_36BIT，设置此DeepColor可能引起异常，需要设置为默认值。
audio_en	是否使能音频：使用音频输出时必须设置为HI_TRUE;
sample_rate	音频采样率，此参数需要与AO的配置保持一致。当前版本目前不支持除48KHz 外的采样率，同时建议用户设置为Sink能力集支持的采样率。
bit_depth	音频位宽，默认为16，当前版本不支持更改为非默认值（此参数需要与AO的配置保持一致）。
auth_mode_en	使能该模式，HDMI强制输出，不再去根据显示或认证设备的EDID信息来自适应调整，主要针对认证场景。 - HI_TRUE：使能bAuthMode - HI_FALSE：不能使bAuthMode 当前版本不支持更改此参数为非默认值。

成员名称	描述
deep_color_adapt_en	当前版本中此参数仅支持设置为：HI_TRUE。
pix_clk	像素时钟，此参数需要与VO配置的输入时序制式对应的像素时钟保持一致，具体值可通过查询VESA对应标准获取。 当前版本不支持更改此参数为与VO输入时序对应的像素时钟不一致的值。

1.9.20.11.7 hi_hdmi_infoframe_type

说明

定义HDMI信息帧类型。

定义

```
typedef enum {  
    HI_INFOFRAME_TYPE_AVI,  
    HI_INFOFRAME_TYPE_AUDIO,  
    HI_INFOFRAME_TYPE_VENDORSPEC,  
    HI_INFOFRAME_TYPE_BUTT  
} hi_hdmi_infoframe_type;
```

成员

成员名称	描述
HI_INFOFRAME_TYPE_AVI	显示信息帧类型。
HI_INFOFRAME_TYPE_AUDIO	音频信息帧类型。
HI_INFOFRAME_TYPE_VENDORSPEC	供应商定制信息帧类型。当前版本不支持。
HI_INFOFRAME_TYPE_BUTT	预留给值。

 说明

当前版本对该结构体的部分参数赋值存在如上表所述的限制，用例中需要按照上述限制配置相关接口的入参值，否则属于未定义行为，影响送显、音频输出。

1.9.20.11.8 hi_hdmi_color_space

说明

定义HDMI颜色空间。

定义

```
typedef enum {  
    HI_HDMI_COLOR_SPACE_RGB444,  
    HI_HDMI_COLOR_SPACE_YCBCR422,  
    HI_HDMI_COLOR_SPACE_YCBCR444,  
    HI_HDMI_COLOR_SPACE_YCBCR420,  
    HI_HDMI_COLOR_SPACE_BUTT  
} hi_hdmi_color_space;
```

成员

成员名称	描述
HI_HDMI_COLOR_SPACE_RGB444	RGB444。当前版本不支持。
HI_HDMI_COLOR_SPACE_YCBCR422	YUV422。当前版本不支持。
HI_HDMI_COLOR_SPACE_YCBCR444	YUV444。
HI_HDMI_COLOR_SPACE_YCBCR420	YUV420。当前版本不支持。
HI_HDMI_COLOR_SPACE_BUTT	预留值。

说明

当前版本HDMI输入来自VO，VO输出格式仅支持YUV444。

1.9.20.11.9 hi_hdmi_bar_info

说明

定义HDMI面板信息。

定义

```
typedef enum {  
    HI_HDMI_BAR_INFO_NOT_VALID,  
    HI_HDMI_BAR_INFO_V,  
    HI_HDMI_BAR_INFO_H,  
    HI_HDMI_BAR_INFO_VH,  
    HI_HDMI_BAR_INFO_BUTT  
} hi_hdmi_bar_info;
```

成员

成员名称	描述
HI_HDMI_BAR_INFO_NOT_VALID	当前版本不支持HDMI 面板信息配置，需配置此项。

成员名称	描述
HI_HDMI_BAR_INFO_V	预留值。
HI_HDMI_BAR_INFO_H	预留值。
HI_HDMI_BAR_INFO_VH	预留值。
HI_HDMI_BAR_INFO_BUTT	预留值。

1.9.20.11.10 hi_hdmi_scan_info

说明

定义HDMI扫描信息。

定义

```
typedef enum {  
    HI_HDMI_SCAN_INFO_NO_DATA,  
    HI_HDMI_SCAN_INFO_OVERSCANNED,  
    HI_HDMI_SCAN_INFO_UNDERSCANNED,  
    HI_HDMI_SCAN_INFO_BUTT  
} hi_hdmi_scan_info;
```

成员

成员名称	描述
HI_HDMI_SCAN_INFO_NO_DATA	当前版本不支持HDMI 扫描信息配置，需配置此项。
HI_HDMI_SCAN_INFO_OVERSCANNED	预留值。
HI_HDMI_SCAN_INFO_UNDERSCANNED	预留值。
HI_HDMI_SCAN_INFO_BUTT	预留值。

1.9.20.11.11 hi_hdmi_colorimetry

说明

定义HDMI音频输出信号色域标准。

定义

```
typedef enum {  
    HI_HDMI_COMMON_COLORIMETRY_NO_DATA,  
    HI_HDMI_COMMON_COLORIMETRY_ITU601,  
}
```

```
HI_HDMI_COMMON_COLORIMETRY_ITU709,  
HI_HDMI_COMMON_COLORIMETRY_BUTT  
} hi_hdmi_colorimetry;
```

成员

成员名称	描述
HI_HDMI_COMMON_COLORIMETRY_NO_DATA	不配置色域。
HI_HDMI_COMMON_COLORIMETRY_ITU601	国际电信联盟601标准。
HI_HDMI_COMMON_COLORIMETRY_ITU709	国际电信联盟709标准。当前版本不支持。
HI_HDMI_COMMON_COLORIMETRY_BUTT	预留值。

1.9.20.11.12 hi_hdmi_ex_colorimetry

说明

定义HDMI音频输出信号色域标准扩展。

定义

```
typedef enum {  
    HI_HDMI_COMMON_COLORIMETRY_XVYCC_601,  
    HI_HDMI_COMMON_COLORIMETRY_XVYCC_709,  
    HI_HDMI_COMMON_COLORIMETRY_S_YCC_601,  
    HI_HDMI_COMMON_COLORIMETRY_ADOBE_YCC_601,  
    HI_HDMI_COMMON_COLORIMETRY_ADOBE_RGB,  
    HI_HDMI_COMMON_COLORIMETRY_2020_CONST_LUMINOUS,  
    HI_HDMI_COMMON_COLORIMETRY_2020_NON_CONST_LUMINOUS,  
    HI_HDMI_COMMON_COLORIMETRY_EXT_BUTT  
} hi_hdmi_ex_colorimetry;
```

成员

成员名称	描述
HI_HDMI_COMMON_COLORIMETRY_XVYCC_601	新一代601色域标准扩展
HI_HDMI_COMMON_COLORIMETRY_XVYCC_709	预留值。
HI_HDMI_COMMON_COLORIMETRY_S_YCC_601	预留值。
HI_HDMI_COMMON_COLORIMETRY_ADOBE_YCC_601	预留值。

成员名称	描述
HI_HDMI_COMMON_COLORIMETRY_ADOBE_RGB	预留值。
HI_HDMI_COMMON_COLORIMETRY_2020_CONST_LUMINOUS	预留值。
HI_HDMI_COMMON_COLORIMETRY_2020_NON_CONST_LUMINOUS	预留值。
HI_HDMI_COMMON_COLORIMETRY_EXT_BUTT	预留值。

1.9.20.11.13 hi_pic_aspect_ratio

说明

定义HDMI图像宽高比。

定义

```
typedef enum {  
    HI_HDMI_PIC_ASPECT_RATIO_NO_DATA,  
    HI_HDMI_PIC_ASPECT_RATIO_4TO3,  
    HI_HDMI_PIC_ASPECT_RATIO_16TO9,  
    HI_HDMI_PIC_ASPECT_RATIO_64TO27,  
    HI_HDMI_PIC_ASPECT_RATIO_256TO135,  
    HI_HDMI_PIC_ASPECT_RATIO_BUTT  
} hi_pic_aspect_ratio;
```

成员

成员名称	描述
HI_HDMI_PIC_ASPECT_RATIO_4TO3	宽高比4：3。
HI_HDMI_PIC_ASPECT_RATIO_16TO9	宽高比16：9。
HI_HDMI_PIC_ASPECT_RATIO_64TO27	宽高比64：27。当前版本不支持。
HI_HDMI_PIC_ASPECT_RATIO_256TO135	宽高比256：135。当前版本不支持。
HI_HDMI_PIC_ASPECT_RATIO_BUTT	预留值。

1.9.20.11.14 hi_hdmi_active_aspect_ratio

说明

定义HDMI实际图像有效宽高比。

定义

```
typedef enum {  
    HI_HDMI_ACTIVE_ASPECT_RATIO_16TO9_TOP = 2,  
    HI_HDMI_ACTIVE_ASPECT_RATIO_14TO9_TOP,  
    HI_HDMI_ACTIVE_ASPECT_RATIO_16TO9_BOX_CENTER,  
    HI_HDMI_ACTIVE_ASPECT_RATIO_SAME_PIC = 8,  
    HI_HDMI_ACTIVE_ASPECT_RATIO_4TO3_CENTER,  
    HI_HDMI_ACTIVE_ASPECT_RATIO_16TO9_CENTER,  
    HI_HDMI_ACTIVE_ASPECT_RATIO_14TO9_CENTER,  
    HI_HDMI_ACTIVE_ASPECT_RATIO_4TO3_14_9 = 13,  
    HI_HDMI_ACTIVE_ASPECT_RATIO_16TO9_14_9,  
    HI_HDMI_ACTIVE_ASPECT_RATIO_16TO9_4_3,  
    HI_HDMI_ACTIVE_ASPECT_RATIO_BUTT  
} hi_hdmi_active_aspect_ratio;
```

成员

成员名称	描述
HI_HDMI_ACTIVE_ASPECT_RATIO_16TO9_TOP	实际图像宽高比16：9，顶部区域。当前版本不支持。
HI_HDMI_ACTIVE_ASPECT_RATIO_14TO9_TOP	实际图像宽高比14：9，顶部区域。当前版本不支持。
HI_HDMI_ACTIVE_ASPECT_RATIO_16TO9_BOX_CENTER	实际图像宽高比16：9，盒型中心区域。当前版本不支持。
HI_HDMI_ACTIVE_ASPECT_RATIO_SAME_PIC	实际图像宽高比与图形一致。
HI_HDMI_ACTIVE_ASPECT_RATIO_4TO3_CENTER	实际图像宽高比4：3，中心区域。当前版本不支持。
HI_HDMI_ACTIVE_ASPECT_RATIO_16TO9_CENTER	实际图像宽高比16：9，中心区域。当前版本不支持。
HI_HDMI_ACTIVE_ASPECT_RATIO_14TO9_CENTER	实际图像宽高比14：9，中心区域。当前版本不支持。
HI_HDMI_ACTIVE_ASPECT_RATIO_4TO3_14_9	实际图像宽高比4：3，拉升为14：9。当前版本不支持。
HI_HDMI_ACTIVE_ASPECT_RATIO_16TO9_14_9	实际图像宽高比16：9，拉升为14：9。当前版本不支持。
HI_HDMI_ACTIVE_ASPECT_RATIO_16TO9_4_3	实际图像宽高比16：9，拉升为4：3。当前版本不支持。
HI_HDMI_ACTIVE_ASPECT_RATIO_BUTT	预留值。

1.9.20.11.15 hi_hdmi_pic_scaline

说明

定义HDMI图像扫描信息。

定义

```
typedef enum {  
    HI_HDMI_PIC_NON_UNIFORM_SCALING,  
    HI_HDMI_PIC_SCALING_H,  
    HI_HDMI_PIC_SCALING_V,  
    HI_HDMI_PIC_SCALING_HV,  
    HI_HDMI_PIC_SCALING_BUTT  
} hi_hdmi_pic_scaline;
```

成员

成员名称	描述
HI_HDMI_PIC_NON_UNIFORM_SCALING	不做特殊扫描处理。
HI_HDMI_PIC_SCALING_H	水平扫描。当前版本不支持。
HI_HDMI_PIC_SCALING_V	垂直扫描。当前版本不支持。
HI_HDMI_PIC_SCALING_HV	水平垂直同时扫描。当前版本不支持。
HI_HDMI_PIC_SCALING_BUTT	预留值。

1.9.20.11.16 hi_hdmi_rgb_quant_range

说明

定义HDMI RGB量化范围。

定义

```
typedef enum {  
    HI_HDMI_RGB_QUANT_DEFAULT_RANGE  
    HI_HDMI_RGB_QUANT_LIMITED_RANGE  
    HI_HDMI_RGB_QUANT_FULL_RANGE  
    HI_HDMI_RGB_QUANT_FULL_BUTT  
} hi_hdmi_rgb_quant_range;
```

成员

成员名称	描述
HI_HDMI_RGB_QUANT_DEFAULT_RANGE	RGB默认范围。

成员名称	描述
HI_HDMI_RGB_QUANT_LIMITED_RANGE	RGB有限范围。
HI_HDMI_RGB_QUANT_FULL_RANGE	RGB全量范围。
HI_HDMI_RGB_QUANT_FULL_BUTT	预留值。

1.9.20.11.17 hi_hdmi_pixel_repetition

说明

定义HDMI像素复制次数。

定义

```
typedef enum {  
    HI_HDMI_PIXEL_REPET_NO,  
    HI_HDMI_PIXEL_REPET_2_TIMES,  
    HI_HDMI_PIXEL_REPET_3_TIMES,  
    HI_HDMI_PIXEL_REPET_4_TIMES,  
    HI_HDMI_PIXEL_REPET_5_TIMES,  
    HI_HDMI_PIXEL_REPET_6_TIMES,  
    HI_HDMI_PIXEL_REPET_7_TIMES,  
    HI_HDMI_PIXEL_REPET_8_TIMES,  
    HI_HDMI_PIXEL_REPET_9_TIMES,  
    HI_HDMI_PIXEL_REPET_10_TIMES,  
    HI_HDMI_PIXEL_REPET_BUTT  
} hi_hdmi_pixel_repetition;
```

成员

成员名称	描述
HI_HDMI_PIXEL_REPET_NO	不做像素复制。
HI_HDMI_PIXEL_REPET_2_TIMES	像素复制2次。当前版本不支持。
HI_HDMI_PIXEL_REPET_3_TIMES	像素复制3次。当前版本不支持。
HI_HDMI_PIXEL_REPET_4_TIMES	像素复制4次。当前版本不支持。
HI_HDMI_PIXEL_REPET_5_TIMES	像素复制5次。当前版本不支持。
HI_HDMI_PIXEL_REPET_6_TIMES	像素复制6次。当前版本不支持。

成员名称	描述
HI_HDMI_PIXEL_REPET_7_TIMES	像素复制7次。当前版本不支持。
HI_HDMI_PIXEL_REPET_8_TIMES	像素复制8次。当前版本不支持。
HI_HDMI_PIXEL_REPET_9_TIMES	像素复制9次。当前版本不支持。
HI_HDMI_PIXEL_REPET_10_TIMES	像素复制10次。当前版本不支持。
HI_HDMI_PIXEL_REPET_BUTT	预留值。

1.9.20.11.18 hi_hdmi_content_type

说明

定义HDMI内容信息。

定义

```
typedef enum {  
    HI_HDMI_CONTNET_GRAPHIC,  
    HI_HDMI_CONTNET_PHOTO,  
    HI_HDMI_CONTNET_CINEMA,  
    HI_HDMI_CONTNET_GAME,  
    HI_HDMI_CONTNET_BUTT  
} hi_hdmi_content_type;
```

成员

成员名称	描述
HI_HDMI_CONTNET_GRAPHIC	图像信息。
HI_HDMI_CONTNET_PHOTO	照片信息。
HI_HDMI_CONTNET_CINEMA	相机信息。当前版本不支持。
HI_HDMI_CONTNET_GAME	游戏信息。当前版本不支持。
HI_HDMI_CONTNET_BUTT	预留值。

1.9.20.11.19 hi_hdmi_ycc_quant_range

说明

定义HDMI YUV量化范围。

定义

```
typedef enum {  
    HI_HDMI_YCC_QUANT_LIMITED_RANGE,  
    HI_HDMI_YCC_QUANT_FULL_RANGE,  
    HI_HDMI_YCC_QUANT_BUTT  
} hi_hdmi_ycc_quant_range;
```

成员

成员名称	描述
HI_HDMI_YCC_QUANT_LIMITED_RANGE	YUV分量有限范围。
HI_HDMI_YCC_QUANT_FULL_RANGE	YUV全量有限范围。
HI_HDMI_YCC_QUANT_BUTT	预留值。

1.9.20.11.20 hi_hdmi_avi_infotrame

说明

定义HDMI输出AVI（Audio Video Interleave）信息帧。

定义

```
typedef struct {  
    hi_hdmi_video_format timing_mode;  
    hi_hdmi_color_space color_space;  
    hi_bool active_info_present;  
    hi_hdmi_bar_info bar_info;  
    hi_hdmi_scan_info scan_info;  
    hi_hdmi_colorimetry colorimetry;  
    hi_hdmi_ex_colorimetry ex_colorimetry;  
    hi_pic_aspect_ratio aspect_ratio;  
    hi_hdmi_active_aspect_ratio active_aspect_ratio;  
    hi_hdmi_pic_scaline pic_scaling;  
    hi_hdmi_rgb_quant_range rgb_quant;  
    hi_bool is_it_content;  
    hi_hdmi_pixel_repetition pixel_repetition;  
    hi_hdmi_content_type content_type;  
    hi_hdmi_ycc_quant_range ycc_quant;  
    hi_u16 line_n_end_of_top_bar;  
    hi_u16 line_n_start_of_bot_bar;  
    hi_u16 pixel_n_end_of_left_bar;  
    hi_u16 pixel_n_start_of_right_bar;  
} hi_hdmi_avi_infotrame;
```

成员

成员名称	描述
timing_mode	视频时序，必须与vo输入时序对应,否则属于未定义行为，影响送显。
color_space	颜色空间，必须与vo输入颜色空间对应，用户送图时必须设置为HI_HDMI_COLOR_SPACE_YCBCR444。
active_info_present	信息是否有效：当前版本仅支持设置为HI_FALSE。
bar_info	面板信息：当前版本仅支持设置为HI_HDMI_BAR_INFO_NOT_VALID。
scan_info	扫描信息：当前版本仅支持设置为HI_HDMI_SCAN_INFO_NO_DATA。
colorimetry	色域：当前版本仅支持设置为HI_HDMI_COMMON_COLORIMETRY_ITU601。
ex_colorimetry	扩展色域：当前版本仅支持设置为HI_HDMI_COMMON_COLORIMETRY_XVYCC_601。
aspect_ratio;	图像宽高比：当前版本仅支持设置为HI_HDMI_PIC_ASPECT_RATIO_4TO3或者HI_HDMI_PIC_ASPECT_RATIO_16TO9。
active_aspect_ratio	实际图像有效宽高比：当前版本仅支持设置为HI_HDMI_ACTIVE_ASPECT_RATIO_SAME_PIC。
pic_scaling	图像扫描：当前版本仅支持设置为HI_HDMI_PIC_NON_UNIFORM_SCALING。
rgb_quant	RGB量化：当前版本仅支持设置为HI_HDMI_RGB_QUANT_FULL_RANGE。
is_it_content	判断内容是否有效：当前版本仅支持设置为HI_FALSE。
pixel_repetition	像素加倍：当前版本仅支持设置为HI_HDMI_PIXEL_REPET_NO。
content_type	内容信息类型：当前版本仅支持设置为HI_HDMI_CONTNET_PHOTO。
ycc_quant	YUV分量量化：当前版本仅支持设置为HI_HDMI_YCC_QUANT_FULL_RANGE。
line_n_end_of_top_bar	顶部终止行数：当前版本仅支持设置为0。
line_n_start_of_bot_bar	底部开始行数：当前版本仅支持设置为0。
pixel_n_end_of_left_bar	左侧终止像素数：当前版本仅支持设置为0。
pixel_n_start_of_right_bar	右侧开始像素数：当前版本仅支持设置为0。

1.9.20.11.21 hi_hdmi_audio_chn_cnt

说明

定义音频通道数。

定义

```
typedef enum {  
    HI_HDMI_AUDIO_CHN_CNT_STREAM,  
    HI_HDMI_AUDIO_CHN_CNT_2,  
    HI_HDMI_AUDIO_CHN_CNT_3,  
    HI_HDMI_AUDIO_CHN_CNT_4,  
    HI_HDMI_AUDIO_CHN_CNT_5,  
    HI_HDMI_AUDIO_CHN_CNT_6,  
    HI_HDMI_AUDIO_CHN_CNT_7,  
    HI_HDMI_AUDIO_CHN_CNT_8,  
    HI_HDMI_AUDIO_CHN_CNT_BUTT  
} hi_hdmi_audio_chn_cnt;
```

成员

成员名称	描述
HI_HDMI_AUDIO_CHN_CNT_STREAM	STREAM模式的音频通道形态。
HI_HDMI_AUDIO_CHN_CNT_2	2声道。
HI_HDMI_AUDIO_CHN_CNT_3	3声道。当前版本不支持。
HI_HDMI_AUDIO_CHN_CNT_4	4声道。当前版本不支持。
HI_HDMI_AUDIO_CHN_CNT_5	5声道。当前版本不支持。
HI_HDMI_AUDIO_CHN_CNT_6	6声道。当前版本不支持。
HI_HDMI_AUDIO_CHN_CNT_7	7声道。当前版本不支持。
HI_HDMI_AUDIO_CHN_CNT_8	8声道。当前版本不支持。
HI_HDMI_AUDIO_CHN_CNT_BUTT	预留值。

1.9.20.11.22 hi_hdmi_coding_type

说明

定义HDMI音频输出采样位宽。

定义

```
typedef enum {  
    HI_HDMI_AUDIO_CODING_REFER_STREAM_HEAD,  
    HI_HDMI_AUDIO_CODING_PCM,  
    HI_HDMI_AUDIO_CODING_AC3,  
    HI_HDMI_AUDIO_CODING_MPEG1,  
    HI_HDMI_AUDIO_CODING_MP3,  
    HI_HDMI_AUDIO_CODING_MPEG2,  
    HI_HDMI_AUDIO_CODING_AACLC,  
    HI_HDMI_AUDIO_CODING_DTS,  
    HI_HDMI_AUDIO_CODING_ATRAC,  
    HI_HDMI_AUDIO_CODING_ONE_BIT_AUDIO,  
    HI_HDMI_AUDIO_CODING_ENAHNCED_AC3,  
    HI_HDMI_AUDIO_CODING_DTS_HD,  
    HI_HDMI_AUDIO_CODING_MAT,  
    HI_HDMI_AUDIO_CODING_DST,  
    HI_HDMI_AUDIO_CODING_WMA_PRO,  
    HI_HDMI_AUDIO_CODING_BUTT  
} hi_hdmi_coding_type;
```

成员

成员名称	描述
HI_HDMI_AUDIO_CODING_REFER_STREAM_HEAD	REFER_STREAM_HEAD格式音频。当前版本不支持。
HI_HDMI_AUDIO_CODING_PCM	PCM格式音频。
HI_HDMI_AUDIO_CODING_AC3	AC3格式音频。当前版本不支持。
HI_HDMI_AUDIO_CODING_MPEG1	MPEG1格式音频。当前版本不支持。
HI_HDMI_AUDIO_CODING_MP3	MP3格式音频。当前版本不支持。
HI_HDMI_AUDIO_CODING_MPEG2	MPEG2格式音频。当前版本不支持。
HI_HDMI_AUDIO_CODING_AACLC	AACLC格式音频。当前版本不支持。
HI_HDMI_AUDIO_CODING_DTS	DTS格式音频。当前版本不支持。
HI_HDMI_AUDIO_CODING_ATRAC	ATRAC格式音频。当前版本不支持。
HI_HDMI_AUDIO_CODING_ONE_BIT_AUDIO	ONE_BIT_AUDIO格式音频。当前版本不支持。
HI_HDMI_AUDIO_CODING_ENAHNCED_AC3	ENAHNCED_AC3格式音频。当前版本不支持。
HI_HDMI_AUDIO_CODING_DTS_HD	DTS_HD格式音频。当前版本不支持。

成员名称	描述
HI_HDMI_AUDIO_CODING_MAT	MAT格式音频。当前版本不支持。
HI_HDMI_AUDIO_CODING_DST	DST格式音频。当前版本不支持。
HI_HDMI_AUDIO_CODING_WMA_PRO	WMA_PRO格式音频。当前版本不支持。
HI_HDMI_AUDIO_CODING_BUTT	预留给。

1.9.20.11.23 hi_hdmi_audio_sample_size

说明

定义音频采样大小。

定义

```
typedef enum {
    HI_HDMI_AUDIO_SAMPLE_SIZE_STREAM,
    HI_HDMI_AUDIO_SAMPLE_SIZE_16,
    HI_HDMI_AUDIO_SAMPLE_SIZE_20,
    HI_HDMI_AUDIO_SAMPLE_SIZE_24,
    HI_HDMI_AUDIO_SAMPLE_SIZE_BUTT
} hi_hdmi_audio_sample_size;
```

成员

成员名称	描述
HI_HDMI_AUDIO_SAMPLE_SIZE_STREAM	STREAM格式采样率。当前版本不支持。
HI_HDMI_AUDIO_SAMPLE_SIZE_16	音频采样深度为16。
HI_HDMI_AUDIO_SAMPLE_SIZE_20	音频采样深度为20。当前版本不支持。
HI_HDMI_AUDIO_SAMPLE_SIZE_24	音频采样深度为24。当前版本不支持。
HI_HDMI_AUDIO_SAMPLE_SIZE_BUTT	预留给。

1.9.20.11.24 hi_hdmi_audio_sample_freq

说明

定义HDMI 音频采样率。

定义

```
typedef enum {  
    HI_HDMI_AUDIO_SAMPLE_FREQ_STREAM,  
    HI_HDMI_AUDIO_SAMPLE_FREQ_32000,  
    HI_HDMI_AUDIO_SAMPLE_FREQ_44100,  
    HI_HDMI_AUDIO_SAMPLE_FREQ_48000,  
    HI_HDMI_AUDIO_SAMPLE_FREQ_88200,  
    HI_HDMI_AUDIO_SAMPLE_FREQ_96000,  
    HI_HDMI_AUDIO_SAMPLE_FREQ_176400,  
    HI_HDMI_AUDIO_SAMPLE_FREQ_192000,  
    HI_HDMI_AUDIO_SAMPLE_FREQ_BUTT  
} hi_hdmi_audio_sample_freq;
```

成员

成员名称	描述
HI_HDMI_AUDIO_SAMPL E_FREQ_STREAM	定义音频采样频率枚举类型：流控制。当前版本不支持。
HI_HDMI_AUDIO_SAMPL E_FREQ_32000	定义音频采样频率枚举类型：32000hz。当前版本不支持。
HI_HDMI_AUDIO_SAMPL E_FREQ_44100	定义音频采样频率枚举类型：32000hz。当前版本不支持。
HI_HDMI_AUDIO_SAMPL E_FREQ_48000	定义音频采样频率枚举类型：48000hz。
HI_HDMI_AUDIO_SAMPL E_FREQ_88200	定义音频采样频率枚举类型：88200hz。当前版本不支持。
HI_HDMI_AUDIO_SAMPL E_FREQ_96000	定义音频采样频率枚举类型：96000hz。当前版本不支持。
HI_HDMI_AUDIO_SAMPL E_FREQ_176400	定义音频采样频率枚举类型：176400hz。当前版本不支持。
HI_HDMI_AUDIO_SAMPL E_FREQ_192000	定义音频采样频率枚举类型：192000hz。当前版本不支持。
HI_HDMI_AUDIO_SAMPL E_FREQ_BUTT	预留值。

1.9.20.11.25 hi_hdmi_level_shift_val

说明

定义HDMI音频调测等级。

定义

```
typedef enum {
    HI_HDMI_LEVEL_SHIFT_VAL_0_DB,
    HI_HDMI_LEVEL_SHIFT_VAL_1_DB,
    HI_HDMI_LEVEL_SHIFT_VAL_2_DB,
    HI_HDMI_LEVEL_SHIFT_VAL_3_DB,
    HI_HDMI_LEVEL_SHIFT_VAL_4_DB,
    HI_HDMI_LEVEL_SHIFT_VAL_5_DB,
    HI_HDMI_LEVEL_SHIFT_VAL_6_DB,
    HI_HDMI_LEVEL_SHIFT_VAL_7_DB,
    HI_HDMI_LEVEL_SHIFT_VAL_8_DB,
    HI_HDMI_LEVEL_SHIFT_VAL_9_DB,
    HI_HDMI_LEVEL_SHIFT_VAL_10_DB,
    HI_HDMI_LEVEL_SHIFT_VAL_11_DB,
    HI_HDMI_LEVEL_SHIFT_VAL_12_DB,
    HI_HDMI_LEVEL_SHIFT_VAL_13_DB,
    HI_HDMI_LEVEL_SHIFT_VAL_14_DB,
    HI_HDMI_LEVEL_SHIFT_VAL_15_DB,
    HI_HDMI_LEVEL_SHIFT_VAL_BUTT
} hi_hdmi_level_shift_val;
```

成员

成员名称	描述
HI_HDMI_LEVEL_SHIFT_VAL_0_DB	左移值：0db。
HI_HDMI_LEVEL_SHIFT_VAL_1_DB	左移值：1db。当前版本不支持。
HI_HDMI_LEVEL_SHIFT_VAL_2_DB	左移值：2db。当前版本不支持。
HI_HDMI_LEVEL_SHIFT_VAL_3_DB	左移值：3db。当前版本不支持。
HI_HDMI_LEVEL_SHIFT_VAL_4_DB	左移值：4db。当前版本不支持。
HI_HDMI_LEVEL_SHIFT_VAL_5_DB	左移值：5db。当前版本不支持。
HI_HDMI_LEVEL_SHIFT_VAL_6_DB	左移值：6db。当前版本不支持。
HI_HDMI_LEVEL_SHIFT_VAL_7_DB	左移值：7db。当前版本不支持。
HI_HDMI_LEVEL_SHIFT_VAL_8_DB	左移值：8db。当前版本不支持。
HI_HDMI_LEVEL_SHIFT_VAL_9_DB	左移值：9db。当前版本不支持。
HI_HDMI_LEVEL_SHIFT_VAL_10_DB	左移值：10db。当前版本不支持。
HI_HDMI_LEVEL_SHIFT_VAL_11_DB	左移值：11db。当前版本不支持。

成员名称	描述
HI_HDMI_LEVEL_SHIFT_V AL_12_DB	左移值：12db。当前版本不支持。
HI_HDMI_LEVEL_SHIFT_V AL_13_DB	左移值：13db。当前版本不支持。
HI_HDMI_LEVEL_SHIFT_V AL_14_DB	左移值：14db。当前版本不支持。
HI_HDMI_LEVEL_SHIFT_V AL_15_DB	左移值：15db。当前版本不支持。
HI_HDMI_LEVEL_SHIFT_V AL_BUTT	预留值。

1.9.20.11.26 hi_hdmi_lfe_playback_level

说明

定义HDMI音频回放等级。

定义

```
typedef enum {  
    HI_HDMI_LFE_PLAYBACK_NO,  
    HI_HDMI_LFE_PLAYBACK_0_DB,  
    HI_HDMI_LFE_PLAYBACK_10_DB,  
    HI_HDMI_LFE_PLAYBACK_BUTT  
} hi_hdmi_lfe_playback_level;
```

成员

成员名称	描述
HI_HDMI_LFE_PLAYBACK_ NO	回放功能关闭。
HI_HDMI_LFE_PLAYBACK_ 0_DB	回放功能等级为0db。当前版本不支持。
HI_HDMI_LFE_PLAYBACK_ 10_DB	回放功能等级为10db。当前版本不支持。
HI_HDMI_LFE_PLAYBACK_ BUTT	预留值。

1.9.20.11.27 hi_hdmi_audio_infoframe

说明

定义HDMI AUDIO信息帧单元。

定义

```
typedef struct {  
    hi_hdmi_audio_chn_cnt chn_cnt;  
    hi_hdmi_coding_type coding_type;  
    hi_hdmi_audio_sample_size sample_size;  
    hi_hdmi_audio_sample_freq sampling_freq;  
    hi_u8 chn_alloc;  
    hi_hdmi_level_shift_val level_shift;  
    hi_hdmi_lfe_playback_level lfe_playback_level;  
    hi_bool down_mix_inhibit;  
} hi_hdmi_audio_infotrame;
```

成员

成员名称	描述
chn_cnt	音频通道个数：当前版本仅支持设置为 HI_HDMI_AUDIO_CHN_CNT_2。
coding_type	音频格式：当前版本仅支持设置为 HI_HDMI_AUDIO_CODING_PCM。
sample_size	音频采样深度（位宽）：当前版本仅支持设置为 HI_HDMI_AUDIO_SAMPLE_SIZE_16。
sampling_freq	enSamplingFrequency：当前版本仅支持设置为 HI_HDMI_AUDIO_SAMPLE_FREQ_48000。
chn_alloc	Channel/Speaker分配：当前版本仅支持设置为0。
level_shift	Level Shift Value，左移值：当前版本仅支持设置为 HI_HDMI_LEVEL_SHIFT_VAL_0_DB。
lfe_playback_level	LFE playback level information，LFE 播放等级信息。当前版本仅支持设置为 HI_HDMI_LFE_PLAYBACK_NO。
down_mix_inhibit	Down-mix Inhibit 标志位：当前版本仅支持设置为 HI_FALSE。

1.9.20.11.28 hi_hdmi_vendorspec_infotrame

说明

定义HDMI定制化信息帧单元结构体。

定义

```
typedef struct {  
    hi_u8 data_len;  
    hi_u8 user_data[HI_HDMI_VENDOR_USER_DATA_MAX_LEN];  
} hi_hdmi_vendorspec_infotrame;
```

成员

成员名称	描述
data_len	IEEE（ Institute of Electrical and Electronics Engineers ）注册码数组长度，最大值为22。
user_data	IEEE注册码，当前版本暂不支持使用。 #define HI_HDMI_VENDOR_USER_DATA_MAX_LEN 22

 说明

当前版本中该结构体暂未使用，不可赋值，否则属于未定义行为。

1.9.20.11.29 hi_hdmi_infotrame_unit

说明

定义HDMI信息帧类型结构。

定义

```
typedef union {  
    hi_hdmi_avi_infotrame avi_infotrame;  
    hi_hdmi_audio_infotrame audio_infotrame;  
    hi_hdmi_vendorspec_infotrame vendor_spec_infotrame;  
} hi_hdmi_infotrame_unit;
```

成员

成员名称	描述
avi_infotrame	显示信息帧单元。
audio_infotrame	音频信息帧单元。
vendor_spec_infotrame	供应商定制信息帧单元。当前版本不支持。

1.9.20.11.30 hi_hdmi_infotrame

说明

定义HDMI输出信息帧结构体。

定义

```
typedef struct {  
    hi_hdmi_infotrame_type infotrame_type;  
    hi_hdmi_infotrame_unit infotrame_unit;  
} hi_hdmi_infotrame;
```

成员

成员名称	描述
infoframe_type	信息帧类型，仅支持HI_INFOFRAME_TYPE_AUDIO和HI_INFOFRAME_TYPE_AVI信息帧。
infoframe_unit	信息帧单元(内容)结构体。

1.9.20.11.31 hi_hdmi_sink_capability

说明

定义HDMI Sink端能力集结构体。

定义

```
typedef struct {  
    hi_bool is_connected;  
    hi_bool support_hdmi;  
    hi_bool support_hdmi_2_0;  
} hi_hdmi_sink_capability;
```

成员

成员名称	描述
is_connected	设备是否连接。 - HI_TRUE：已连接 - HI_FALSE：未连接
support_hdmi	设备是否支持HDMI1.4。 - HI_TRUE：支持 - HI_FALSE：不支持
support_hdmi_2_0	设备是否支持HDMI2.0。 - HI_TRUE：支持 - HI_FALSE：不支持

1.9.20.11.32 hi_hdmi_event_type

说明

定义HDMI事件通知枚举。

定义

```
typedef enum {  
    HI_HDMI_EVENT_HOTPLUG = 0x10,  
    HI_HDMI_EVENT_NO_PLUG,
```



```
    HI_HDMI_EVENT_EDID_FAIL,  
    HI_HDMI_EVENT_BUTT  
} hi_hdmi_event_type;
```

成员

成员名称	描述
HI_HDMI_EVENT_HOTPLUG	HDMI Cable插入事件。
HI_HDMI_EVENT_NO_PLUG	HDMI Cable拔出事件。
HI_HDMI_EVENT_EDID_FAIL	HDMI EDID读取失败事件，当前预留，未使能。
HI_HDMI_EVENT_BUTT	预留给值。

1.9.20.11.33 hi_hdmi_callback

说明

定义HDMI回调函数指针类型。

定义

```
typedef void (*hi_hdmi_callback)(hi_hdmi_event_type event, void *private_data);
```

成员

成员名称	描述
event	HDMI事件通知类型。
private_data	事件私有数据。

1.9.20.11.34 hi_hdmi_callback_func

说明

定义HDMI回调函数结构体。

定义

```
typedef struct {  
    hi_hdmi_callback hdmi_event_callback;  
    void *private_data;  
} hi_hdmi_callback_func;
```

成员

成员名称	描述
hdm_i_event_callback	事件处理回调函数。
private_data	回调函数参数私有数据。

1.9.20.12 VPC 图像处理

1.9.20.12.1 hi_vpc_chn

说明

描述图片处理通道号。

定义

```
typedef hi_s32 hi_vpc_chn;
```

1.9.20.12.2 hi_vpc_pic_info

说明

图像信息结构体，用来描述VPC功能输入和输出图片信息。

定义

```
typedef struct {  
    hi_void* picture_address;  
    hi_u32 picture_buffer_size;  
    hi_u32 picture_width;  
    hi_u32 picture_height;  
    hi_u32 picture_width_stride;  
    hi_u32 picture_height_stride;  
    hi_pixel_format picture_format;  
} hi_vpc_pic_info;
```

成员

成员名称	描述
picture_address	存放图片数据的Device地址。
picture_buffer_size	存放图片数据的缓冲区大小。
picture_width	图片真实宽。
picture_height	图片真实高。
picture_width_stride	图片宽stride。
picture_height_stride	图片高stride。
picture_format	目标图片的格式。

1.9.20.12.3 hi_vpc_crop_region

说明

VPC抠图功能需要的参数。

定义

```
typedef struct {  
    hi_u32 top_offset;  
    hi_u32 left_offset;  
    hi_u32 crop_width;  
    hi_u32 crop_height;  
} hi_vpc_crop_region;
```

成员

成员名称	描述
top_offset	上偏移。
left_offset	左偏移。
crop_width	抠图区域宽。
crop_height	抠图区域高。

1.9.20.12.4 hi_vpc_resize_info

说明

VPC缩放功能需要的参数。

定义

```
typedef struct {  
    hi_u32 resize_width;  
    hi_u32 resize_height;  
    hi_u32 interpolation;  
} hi_vpc_resize_info;
```

成员

成员名称	描述
resize_width	缩放之后的宽。
resize_height	缩放之后的高。

成员名称	描述
interpolation	缩放算法。 Atlas 推理系列产品，支持如下缩放算法： 0：业界通用的Bilinear算法（与OpenCV算法的计算过程类似，当输入和输出图片格式都为RGB时，在[1/32, 512]的缩放范围内，与OpenCV的单个像素值最大差异为正负1） 1：业界通用的Nearest neighbor 算法（与OpenCV算法的计算过程类似） Atlas A2训练系列产品/Atlas 800I A2推理产品，支持如下缩放算法： 0：业界通用的Bilinear算法（与OpenCV-3.4.2版本算法的计算结果相同） 1：业界通用的Nearest neighbor 算法（与OpenCV-3.4.2版本算法的计算结果相同） Atlas 200/500 A2推理产品，支持如下缩放算法： 0：业界通用的Bilinear算法（与OpenCV-3.4.2版本算法的计算结果相同） 1：业界通用的Nearest neighbor 算法（与OpenCV-3.4.2版本算法的计算结果相同）

1.9.20.12.5 hi_vpc_crop_region_info

说明

VPC抠图功能需要的参数。

定义

```
typedef struct {  
    hi_vpc_pic_info dest_pic_info;  
    hi_vpc_crop_region crop_region;  
} hi_vpc_crop_region_info;
```

成员

成员名称	描述
dest_pic_info	抠图目标图片信息。
crop_region	抠图区域信息。

1.9.20.12.6 hi_vpc_crop_resize_region

说明

VPC抠图并缩放功能需要的参数。

定义

```
typedef struct {  
    hi_vpc_pic_info dest_pic_info;  
    hi_vpc_crop_region crop_region;  
    hi_vpc_resize_info resize_info;  
} hi_vpc_crop_resize_region;
```

成员

成员名称	描述
dest_pic_info	目标图片信息。
crop_region	抠图区域信息。
resize_info	图片缩放信息。

注意事项

处理图片的数量不能大于256个。缩放宽高必须与目标图片宽高一致。

1.9.20.12.7 hi_vpc_crop_resize_paste_region

说明

VPC抠图然后缩放并贴图到特定区域功能需要的参数。

定义

```
typedef struct {  
    hi_vpc_pic_info dest_pic_info;  
    hi_vpc_crop_region crop_region;  
    hi_vpc_resize_info resize_info;  
    hi_u32 dest_top_offset;  
    hi_u32 dest_left_offset;  
} hi_vpc_crop_resize_paste_region;
```

成员

成员名称	描述
dest_pic_info	目标图片信息。
crop_region	抠图区域信息。
resize_info	图片缩放信息。
dest_top_offset	贴图到另外一张图的顶部偏移。

成员名称	描述
dest_left_offset	贴图到另外一张图的左侧偏移。 注：Atlas 推理系列产品上，左偏移16像素对齐。

1.9.20.12.8 hi_vpc_bord_type

说明

定义边界填充的类型。

定义

```
typedef enum {  
    HI_BORDER_CONSTANT = 0,  
    HI_BORDER_REPLICATE,  
    HI_BORDER_REFLECT,  
    HI_BORDER_REFLECT_101  
} hi_vpc_bord_type;
```

成员

成员名称	描述
HI_BORDER_CONSTANT	添加有颜色的常数值边界。
HI_BORDER_REPLICATE	重复最后一个元素。 举例，其中*表示任意图像元素: aaaaaa a*****h hhhhhhh;
HI_BORDER_REFLECT	边界元素的镜像，镜像包括边界元素。硬件限制，不支持该选项。 举例，其中*表示任意图像元素: ba abc*****fgh hg;
HI_BORDER_REFLECT_101	边界元素的镜像，镜像不包括边界元素。硬件限制，不支持该选项。 举例，其中*表示任意图像元素: cb abc****fgh gf;

注意事项

版本	约束
Atlas 推理系列产品	支持HI_BORDER_CONSTANT和HI_BORDER_REPLICATE可以填充到4096*4096
Atlas 200/500 A2推理产品 Atlas A2训练系列产品/ Atlas 800I A2推理产品	支持HI_BORDER_CONSTANT和HI_BORDER_REPLICATE可以填充到4096*8192

1.9.20.12.9 hi_vpc_make_border_info

说明

VPC图像填充需要用的信息。

定义

```
typedef struct {  
    hi_u32 top;  
    hi_u32 bottom;  
    hi_u32 left;  
    hi_u32 right;  
    hi_vpc_bord_type border_type;  
    hi_vpc_scalar scalar_value;  
} hi_vpc_make_border_info;
```

成员

成员名称	描述
top	顶部填充像素数。
bottom	底部填充像素数。
left	左侧填充像素数。
right	右侧填充像素数。
border_type	扩充边缘像素的类型。
scalar_value	仅在填充类型为HI_BORDER_CONSTANT的时候有效，指定填充的像素值。

1.9.20.12.10 hi_vpc_histogram_config

说明

存放直方图统计结果的结构体。

定义

```
typedef struct {  
    hi_u32 histogram_y_or_r[256];  
    hi_u32 histogram_u_or_g[256];  
    hi_u32 histogram_v_or_b[256];  
} hi_vpc_histogram_config;
```

成员

成员名称	描述
histogram_y_or_r	Y或者R分量，像素分布情况。
histogram_u_or_g	U或者G分量，像素分布情况。

成员名称	描述
histogram_v_or_b	V或者B分量，像素分布情况。

1.9.20.12.11 hi_vpc_lut_remap

说明

配置图像各分量重映射信息的结构体。

定义

```
typedef struct {  
    hi_u8 map_value_y_or_r[256];  
    hi_u8 map_value_u_or_g[256];  
    hi_u8 map_value_v_or_b[256];  
} hi_vpc_lut_remap;
```

成员

成员名称	描述
map_value_y_or_r	Y或者R分量的重映射配置信息。
map_value_u_or_g	U或者G分量的重映射配置信息。
map_value_v_or_b	V或者B分量的重映射配置信息。

1.9.20.12.12 hi_vpc_scalar

说明

定义图像处理像素各分量结构体。

定义

```
typedef struct {  
    hi_double val[4];  
} hi_vpc_scalar;
```

成员

成员名称	描述
val	存放颜色分量的值，取值范围[0,255]。 当前支持用户手动按顺序存放R、G、B分量或Y、U、V分量的值，即val[0]存放R分量或Y分量，val[1]存放G分量或U分量，val[2]存放B分量或V分量，val[3]预留。 若输入、输出图片格式都是YUV格式，填充YUV分量的值；反之，填充的是RGB分量的值。

1.9.20.12.13 hi_vpc_chn_attr

说明

定义图像处理通道属性结构体。

定义

```
typedef struct {  
    hi_s32 attr;  
    hi_u32 pic_width;  
    hi_u32 pic_height;  
} hi_vpc_chn_attr;
```

成员

成员名称	描述
attr	Atlas 200/500 A2推理产品，预留参数，暂不支持。 Atlas A2训练系列产品/Atlas 800I A2推理产品，预留参数，暂不支持。 任务队列深度，取值范围：[0,350]，取值越大，队列深度越深，在一个通道内可下发的任务数量越多，但系统内部内存消耗线性增加。 attr参数取值为[0,350]之间，具体说明如下： - 取值为0，表示默认使用32。 - 取值为[1, 10)，VPC内部考虑批量任务的处理（例如 hi_mpi_vpc_batch_crop_resize_paste），自动将队列深度设置为10。 - 取值为[10, 350]，取用户所配置的值。 说明 任务队列深度取值大于350时，当前版本会自行采用350的取值，为避免后续版本的兼容性问题，建议用户在[队列深度最小值,350]范围内取值。 在一个VPC处理通道关联n个线程的场景下，防止通道队列满而阻塞任务下发，建议增加任务队列深度以保证性能，当线程数≤32时，可使用默认队列深度32；当线程数>32时，建议队列深度大于或等于线程数。
pic_width	通道支持的处理图像最大宽（以像素为单位）。预留字段，暂不支持。需要用户手动设置为0，避免后续版本的兼容性问题。 静态属性。
pic_height	通道支持的处理图像最大高（以像素为单位）。预留字段，暂不支持。需要用户手动设置为0，避免后续版本的兼容性问题。 静态属性。

1.9.20.12.14 hi_vpc_crop_resize_border_region

说明

VPC抠图缩放填充信息结构体。

定义

```
typedef struct {  
    hi_vpc_pic_info dest_pic_info;  
    hi_vpc_crop_region crop_region;  
    hi_vpc_resize_info resize_info;  
    hi_u32 dest_top_offset;  
    hi_u32 dest_left_offset;  
    hi_vpc_bord_type border_type;  
    hi_vpc_scalar scalar_value;  
} hi_vpc_crop_resize_border_region;
```

成员

成员名称	描述
dest_pic_info	目标图片信息。
crop_region	原图抠图区域信息。
resize_info	抠图缩放信息。
dest_top_offset	贴图到另外一张图的顶部偏移。
dest_left_offset	贴图到另外一张图的左侧偏移。
border_type	扩充边缘像素的类型。
scalar_value	仅在填充类型为HI_BORDER_CONSTANT的时候有效，指定填充的像素值。 当前支持用户手动按顺序存放R、G、B分量或Y、U、V分量的值。 若输入输出都是YUV格式，填充YUV分量的值；反之，填充的是RGB分量的值。

注意事项

默认抠图区域的图片格式会转成目标图片的格式，然后再缩放、填充至目标图片的宽高。

1.9.20.12.15 hi_csc_conf

说明

定义色域配置信息。

定义

```
type struct{  
    hi_u32 alpha;
```

```
hi_u32 attr[4];  
};
```

成员

成员名称	描述
alpha	用于设置图像的透明度，取值范围：[0, 255]，取值越小，图像越透明。 在查看图像时，无法直接感知透明度，一般在对图像进行融合、渲染等处理时才会使用透明度信息。
attr	预留字段，暂不支持。需要用户手动将数组中的每个元素值设置为0，避免后续版本的兼容性问题。

1.9.20.12.16 hi_roundview_stitching_param

说明

定义环视拼接参数的结构体。

定义

```
typedef struct {  
    hi_stiching_ipm_param imp_conf;  
    hi_stiching_gain_param gain_conf;  
    hi_u32 reserved[8];  
} hi_roundview_stitching_param;
```

成员

成员名称	描述
imp_conf	畸变矫正参数结构体。
gain_conf	增益补偿参数结构体。 若无需增益补偿，则需要给该结构体内的gain_type成员设置为GAIN_NONE。
reserved	预留参数，为保证后续版本兼容性，请务必使用memset结构体方式进行清零初始化，在代码中必须避免显式对reserved字段进行访问。

1.9.20.12.17 hi_roundview_stitching_pic_param

说明

定义环视拼接输入、输出图片信息的结构体。

定义

```
typedef struct {  
    hi_vpc_pic_info *dest_pic;
```

```
hi_u32 count;  
hi_vpc_pic_info *source_pic;  
} hi_roundview_stitching_pic_param;
```

成员

成员名称	描述
dest_pic	输出图片指针。
count	输入图片数量，当前固定配置为4，否则会返回报错。
source_pic	输入图片数组。 source_pic[0]存放第一张输入图片数据，source_pic[1]存放第二张输入图片数据，source_pic[2]存放第三张输入图片数据，source_pic[3]存放第四张输入图片数据。 4张输入图片的宽保持一致、高保持一致。

1.9.20.12.18 hi_stiching_ipm_param

说明

定义畸变矫正参数的结构体。

定义

```
typedef struct {  
    hi_u32 src_pic_width;  
    hi_u32 src_pic_height;  
    hi_u32 dest_pic_width;  
    hi_u32 dest_pic_height;  
    hi_u32 ipm_table_len;  
    hi_stiching_ipm_table *ipm_table_address;  
} hi_stiching_ipm_param;
```

成员

成员名称	描述
src_pic_width	输入图片宽。 4张输入图片的宽保持一致。
src_pic_height	输入图片高。 4张输入图片的高保持一致。
dest_pic_width	输出图片宽。
dest_pic_height	输出图片高。
ipm_table_len	数组长度，必须为dest_pic_width * dest_pic_height

成员名称	描述
ipm_table_address	畸变矫正结构体数组。 由用户提前申请内存存放数组中的数据，内存大小（单位为Byte）= sizeof(hi_stiching_ipm_table) * ipm_table_len，该内存大小可能比较大，如果直接使用C标准库中的malloc接口申请内存，会导致内存碎片化，大页个数减少，进而影响使用hi_mpi_dvpp_malloc接口申请DVPP各功能的输入/输出内存。因此推荐直接使用hi_mpi_dvpp_malloc接口申请存放ipm_table_address数组数据的内存。

1.9.20.12.19 hi_stiching_ipm_table

说明

定义畸变矫正表的结构体。即每个输出图片像素点，是怎么由2幅输入图片融合成的。

定义

```
typedef struct {  
    hi_u32 pic1_index;  
    hi_u32 pic2_index;  
    hi_float weight1;  
    hi_float weight2;  
    hi_float offset1_x;  
    hi_float offset1_y;  
    hi_float offset2_x;  
    hi_float offset2_y;  
}hi_stiching_ipm_table;
```

成员

设置输入图片编号时，如果pic1_index、pic2_index设置为同一个编号，则表示图像融合时参考的是同一张输入图片。

设置图像融合的比重时，要求weight2 + weight1 = 1。

如果weight1、weight2都设置为0，无论pic1_index、pic2_index配置什么，硬件都不会进行融合。

成员名称	描述
pic1_index	对应输入图片1的编号，取值范围[0, 3]。
pic2_index	对应输入图片2的编号，取值范围[0, 3]。
weight1	图像融合时图片1的比重，取值范围[0, 1]。
weight2	图像融合时图片2的比重，取值范围[0, 1]。
offset1_x	图片1插值点x坐标。
offset1_y	图片1插值点y坐标。
offset2_x	图片2插值点x坐标。

成员名称	描述
offset2_y	图片2插值点y坐标。

1.9.20.12.20 hi_stitching_gain_param

说明

定义增益补偿参数的结构体。

定义

```
typedef struct {  
    hi_stitching_histogram_type hist_type;  
    hi_stitching_gain_type gain_type;  
    hi_void *user_data;  
    hi_s32 (*calculate_gain_callback)(  
        hi_void *user_data,  
        hi_void *hist_config,  
        hi_u32 hist_count,  
        void *gain,  
        hi_u32 gain_size);  
} hi_stitching_gain_param;
```

成员

成员名称	描述
hist_type	直方图统计类型。
gain_type	增益补偿类型。
user_data	用户自定义数据。

成员名称	描述
calculate_gain_callback	回调函数指针。 gain_type选择GAIN_LUT时，必须配置回调函数指针。 回调函数内的参数说明如下： • user_data 系统将hi_stitching_gain_param.user_data用户自定义数据赋值给回调函数内的user_data。 • hist_config 回调函数中的hist_config参数，由系统内部管理，用户可以在下发环视拼接任务之后，释放环视拼接内部资源前，通过该参数获取直方图信息。 • hist_count hi_stitching_histogram_param结构体对象的个数。 当直方图统计类型hi_stitching_histogram_type为OVERLAP_HISTOGRAM 时，hist_count固定为8；当直方图统计类型hi_stitching_histogram_type为GLOBAL_HISTOGRAM时，hist_count固定为4。 • gain 回调函数中的gain参数，需由用户转换为hi_vpc_lut_remap结构体数组，数组长度为4： hi_vpc_lut_remap* lutRemap = (hi_vpc_lut_remap*)gain • gain_size hi_vpc_lut_remap结构体数组的长度，当前为4。

1.9.20.12.21 hi_stitching_histogram_type

说明

环视拼接的直方图统计类型。

定义

```
typedef enum {  
    OVERLAP_HISTOGRAM = 0, // 对8个重叠区域做直方图统计  
    GLOBAL_HISTOGRAM,      // 对4张输入图做直方图统计  
} hi_stitching_histogram_type;
```

1.9.20.12.22 hi_stitching_gain_type

说明

定义增益补偿类型的枚举。

定义

```
typedef enum {  
    GAIN_NONE = 0, // 无需增益补偿  
    GAIN_LUT,      // 由用户配置图像各分量重映射信息来实现增益补偿  
} hi_stitching_gain_type;
```

1.9.20.12.23 hi_stitching_histogram_param

说明

在用于增益补偿的回调函数中，将
hi_stitching_gain_param.calculate_gain_callback.hist_config字段转换为
hi_stitching_histogram_param结构体，可参见[转换示例代码](#)。

定义

```
typedef struct {  
    hi_u32 pic_index;  
    hi_u32 overlap_pic_index;  
    hi_vpc_histogram_config histogram_info;  
} hi_stitching_histogram_param;
```

成员

成员名称	描述
pic_index	输入图片编号。
overlap_pic_index	当直方图统计类型hi_stitching_histogram_type为OVERLAP_HISTOGRAM 时，overlap_pic_index参数表示与输入图片有重叠区域的图片的编号。
histogram_info	直方图统计结果。

转换示例代码

```
hi_stitching_histogram_param *hisData = (hi_stitching_histogram_param *)hist_config;
```

1.9.20.12.24 hi_vpc_crop_resize_resize_paste_region

说明

VPC抠图然后缩放并贴图到特定区域功能需要的参数，支持两次缩放。

定义

```
typedef struct {  
    hi_vpc_pic_info dest_pic_info;  
    hi_vpc_crop_region crop_region;  
    hi_vpc_resize_info resize_info1;  
    hi_vpc_resize_info resize_info2;  
    hi_u32 dest_top_offset;  
    hi_u32 dest_left_offset;  
    hi_u32 reserved[2];  
} hi_vpc_crop_resize_resize_paste_region;
```

成员

成员名称	描述
dest_pic_info	目标图片信息。

成员名称	描述
crop_region	抠图区域信息。
resize_info1	第一次图片缩放的信息。 缩放后的图片宽高需与输出图片的宽高对齐要求保持一致，请参见表1-59。
resize_info2	第二次图片缩放的信息。 缩放后的图片宽高需与输出图片的宽高对齐要求保持一致，请参见表1-59。
dest_top_offset	贴图到另外一张图的顶部偏移。
dest_left_offset	贴图到另外一张图的左侧偏移。 注：Atlas 推理系列产品上，左偏移16像素对齐。
reserved	预留参数，为保证后续版本兼容性，请务必使用memset结构体方式进行清零初始化，在代码中必须避免显式对reserved字段进行访问。

1.9.20.12.25 hi_vpc_workspace_param

说明

定义设置系统内部临时缓存相关的参数结构体。

定义

```
type struct {
    hi_workspace_func func;
    hi_u32 max_source_pic_width;
    hi_u32 max_source_pic_height;
    hi_u32 max_dest_pic_width;
    hi_u32 max_dest_pic_height;
    hi_u64 reserved;
} hi_vpc_workspace_param;
```

成员

输入、输出图片分辨率的取值范围为：10*6~4096*8192。

成员名称	描述
func	指定为哪种功能申请临时缓存。
max_source_pic_width	输入图片的最大宽度。
max_source_pic_height	输入图片的最大高度。
max_dest_pic_width	输出图片的最大宽度。

成员名称	描述
max_dest_pic_height	输出图片的最大高度。
reserved	预留参数，为保证后续版本兼容性，请务必使用memset结构体方式进行清零初始化，在代码中必须避免显式对reserved字段进行访问。

1.9.20.12.26 hi_workspace_func

说明

定义需申请临时缓存的功能枚举。

定义

```
typedef enum {  
    REMAP = 0,  
    AFFINE,  
} hi_workspace_func;
```

1.9.20.12.27 hi_remap_lut

说明

定义像素位置重映射表信息。

定义

```
typedef struct {  
    hi_u32 src_pic_width;  
    hi_u32 src_pic_height;  
    hi_u32 dest_pic_width;  
    hi_u32 dest_pic_height;  
    hi_void *lut;  
    hi_u32 lut_size;  
    hi_u64 reserved;  
} hi_remap_lut;
```

成员

输入、输出图片分辨率的取值范围为：10*6~4096*8192。

成员名称	描述
src_pic_width	输入图片宽度。
src_pic_height	输入图片高度。
dest_pic_width	输出图片宽度。
dest_pic_height	输出图片高度。

成员名称	描述
lut_size	像素位置重映射表的内存大小，单位Byte。 可调用 hi_mpi_vpc_get_lut_mem_size 接口获取内存大小。
lut	像素位置重映射表的内存地址指针。 该内存需由用户提前调用 hi_mpi_dvpp_malloc 接口申请。
reserved	预留参数，为保证后续版本兼容性，请务必使用memset结构体方式进行清零初始化，在代码中必须避免显式对reserved字段进行访问。

1.9.20.12.28 hi_point_pair_info

说明

定义输入、输出图片中的像素点坐标信息的结构体。

定义

```
typedef struct {  
    hi\_float\_point src_point[4];  
    hi\_u32 src_point_count;  
    hi\_float\_point dest_point[4];  
    hi\_u32 dest_point_count;  
} hi_point_pair_info;
```

成员

成员名称	描述
src_point	输入图片像素点坐标。
src_point_count	输入图片像素点坐标数量。
dest_point	输出图片像素点坐标。
dest_point_count	输出图片像素点坐标数量。

1.9.20.12.29 hi_float_point

说明

定义坐标信息的结构体。

定义

```
typedef struct {  
    hi\_float x;  
    hi\_float y;  
} hi_float_point;
```

成员

成员名称	描述
x	像素点x坐标。
y	像素点y坐标。

1.9.20.12.30 hi_map_param

说明

定义用户原始map表信息的结构体。

定义

```
typedef struct {  
    hi_void *map1;  
    hi_void *map2;  
    hi_u32 map_size;  
    hi_u32 src_pic_width;  
    hi_u32 src_pic_height;  
    hi_u32 dest_pic_width;  
    hi_u32 dest_pic_height;  
    hi_u64 reserved;  
} hi_map_param;
```

成员

成员名称	描述
map1	Remap变换使用的map1表, 表示(x,y)坐标的x, 类型为hi_float, 大小为: dest_pic_width * dest_pic_height * sizeof(hi_float)
map2	Remap变换使用的map2表, 表示(x,y)坐标的y, 类型为hi_float, 大小为: dest_pic_width * dest_pic_height * sizeof(hi_float)
map_size	map1或map2的内存大小, map1表和map2表的内存大小一样, 单位Byte。
src_pic_width	map表对应的输入图片的宽度。
src_pic_height	map表对应的输入图片的高度。
dest_pic_width	map表对应的输出图像的宽度。
dest_pic_height	map表对应的输出图像的高度。
reserved	预留参数, 为保证后续版本兼容性, 请务必使用memset结构体方式进行清零初始化, 在代码中必须避免显式对reserved字段进行访问。

1.9.20.12.31 hi_warp_transform_param

说明

定义Remap变换参数结构体。

定义

```
typedef struct {  
    hi_vpc_pic_info src;  
    hi_vpc_pic_info dst;  
    hi_transform_config transform_config;  
} hi_warp_transform_param;
```

成员

成员名称	描述
src	输入图片信息
dst	输出图片信息。
transform_config	仿射/透视/Remap变换配置信息。

1.9.20.12.32 hi_transform_config

说明

定义仿射/透视/Remap变换配置结构体。

定义

```
typedef struct {  
    hi_u32 interpolation;  
    hi_vpc_bord_type border_type;  
    hi_vpc_scalar scalar_value;  
} hi_transform_config;
```

成员

成员名称	描述
interpolation	缩放算法。 Atlas 200/500 A2推理产品支持如下缩放算法： 0：业界通用的Bilinear算法 1：业界通用的Nearest neighbor 算法 Atlas A2训练系列产品/Atlas 800I A2推理产品支持如下缩放算法： 0：业界通用的Bilinear算法 1：业界通用的Nearest neighbor 算法

成员名称	描述
border_type	边界填充的类型。 Atlas 200/500 A2推理产品支持如下取值： 0：寄存器固定值 1：边界复制模式 Atlas A2训练系列产品/Atlas 800I A2推理产品支持如下取值： 0：寄存器固定值 1：边界复制模式
scalar_value	图像填充的像素值，只支持3通道。 如果输入图片格式为YUV400，则val[0]存放Y分量，val[1]、val[2]、val[3]预留，需初始化为0。

1.9.20.12.33 hi_morph_shapes

说明

定义滤波卷积形状。

定义

```
typedef enum {  
    MORPH_RECT = 0,  
    MORPH_CROSS = 1,  
    MORPH_ELLIPSE = 2,  
    MORPH_MAX = 100  
} hi_morph_shapes;
```

成员

成员名称	描述
MORPH_RECT	矩形。
MORPH_CROSS	交叉形，预留字段，暂不支持。
MORPH_ELLIPSE	椭圆形，预留字段，暂不支持。
MORPH_MAX	最大可支持的卷积滤波形状，预留字段，暂不支持。

1.9.20.12.34 hi_rotation

说明

定义旋转角度。

定义

```
typedef enum {  
    HI_ROTATION_90 = 0,  
    HI_ROTATION_180 = 1,  
    HI_ROTATION_270 = 2,  
} hi_rotation;
```

成员

成员名称	描述
HI_ROTATION_90	顺时针旋转90度。
HI_ROTATION_180	顺时针旋转180度。
HI_ROTATION_270	顺时针旋转270度。

1.9.20.12.35 hi_blk_size

说明

定义单个马赛克框的大小。

定义

```
typedef enum {  
    HI_BLK_SIZE_4 = 0,  
    HI_BLK_SIZE_8,  
    HI_BLK_SIZE_16,  
    HI_BLK_SIZE_32,  
    HI_BLK_SIZE_64,  
    HI_BLK_SIZE_128,  
} hi_blk_size;
```

成员

成员名称	描述
HI_BLK_SIZE_4	预留字段，暂不支持。
HI_BLK_SIZE_8	单个马赛克是8*8的正方形。
HI_BLK_SIZE_16	单个马赛克是16*16的正方形。
HI_BLK_SIZE_32	单个马赛克是32*32的正方形。
HI_BLK_SIZE_64	预留字段，暂不支持。

成员名称	描述
HI_BLK_SIZE_1 28	预留字段，暂不支持。

1.9.20.12.36 hi_median_blur_param

说明

定义中值滤波信息结构体。

定义

```
typedef struct {  
    hi_vpc_pic_info src;  
    hi_vpc_pic_info dst;  
    hi_median_blur_config median_blur_cfg;  
} hi_median_blur_param;
```

成员

成员名称	描述
src	输入图片信息。
dst	输出图片信息。
median_blur_cfg	中值滤波配置参数。

1.9.20.12.37 hi_median_blur_config

说明

定义中值滤波参数结构体。

定义

```
typedef struct {  
    hi_u32 kernel_size;  
} hi_median_blur_config;
```

成员

成员名称	描述
kernel_size	中值滤波卷积核尺寸，只支持1，3或者5。

1.9.20.12.38 hi_blur_param

说明

定义滤波信息结构体。

定义

```
typedef struct {  
    hi_vpc_pic_info src;  
    hi_vpc_pic_info dst;  
    hi_blur_config blur_cfg;  
} hi_blur_param;
```

成员

成员名称	描述
src	输入图片信息。
dst	输出图片信息。
blur_cfg	滤波配置参数。

1.9.20.12.39 hi_blur_config

说明

定义滤波参数结构体。

定义

```
typedef struct {  
    hi_size kernel_size;  
    hi_morph_shapes morph_shapes;  
    hi_point anchor;  
    hi_u32 iterations;  
    hi_vpc_bord_type border_type;  
    hi_vpc_scalar scalar_value;  
} hi_blur_config;
```

成员

成员名称	描述
kernel_size	卷积核的尺寸，kernel_size的宽和高只能设置为1，3或者5。
morph_shapes	滤波的卷积形状。
anchor	卷积核锚点，需要设置成(-1, -1)，预留字段。
iterations	卷积迭代次数，范围[1, 100]，均值滤波处理只支持设置成1，腐蚀或膨胀处理时支持取[1, 100]范围内的值。
border_type	边界填充类型，只支持HI_BORDER_CONSTANT和HI_BORDER_REPLICATE两种填充模式。

成员名称	描述
scalar_value	边界填充值。

1.9.20.12.40 hi_gaussian_blur_param

说明

定义高斯滤波信息结构体。

定义

```
typedef struct {  
    hi_vpc_pic_info src;  
    hi_vpc_pic_info dst;  
    hi_gaussian_blur_config gaussian_blur_cfg;  
} hi_gaussian_blur_param;
```

成员

成员名称	描述
src	输入图片信息。
dst	输出图片信息。
gaussian_blur_cfg	高斯滤波配置参数。

1.9.20.12.41 hi_gaussian_blur_config

说明

定义高斯滤波参数结构体。

定义

```
typedef struct {  
    hi_size kernel_size;  
    hi_double sigma_x;  
    hi_double sigma_y;  
    hi_vpc_bord_type border_type;  
    hi_vpc_scalar scalar_value;  
    hi_u32 reserved[2];  
} hi_gaussian_blur_config;
```

成员

成员名称	描述
kernel_size	卷积核尺寸，kernel_size的宽和高只能设置为1，3或者5。

成员名称	描述
sigma_x	高斯滤波x轴sigma值。
sigma_y	高斯滤波y轴sigma值，如果sigma_y=0，则sigma_y=sigma_x。
border_type	边界填充类型，只支持HI_BORDER_CONSTANT和HI_BORDER_REPLICATE两种填充模式。
scalar_value	边界填充值。
reserved[2]	预留参数，为保证后续版本兼容性，请务必使用memset结构体方式进行清零初始化，在代码中必须避免显式对reserved字段进行访问。

1.9.20.12.42 hi_filter_2d_param

说明

定义2D卷积滤波信息结构体。

定义

```
typedef struct {  
    hi_vpc_pic_info src;  
    hi_vpc_pic_info dst;  
    hi_filter_2d_config filter_2d_cfg;  
} hi_filter_2d_param;
```

成员

成员名称	描述
src	输入图片信息。
dst	输出图片信息。
filter_2d_cfg	2D卷积滤波配置参数。

1.9.20.12.43 hi_filter_2d_config

说明

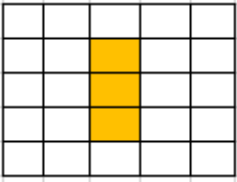
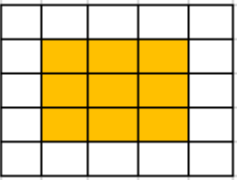
定义2D卷积滤波参数结构体。

定义

```
typedef struct {  
    hi_double filter[5][5];  
    hi_size kernel_size;  
    hi_point anchor;  
    hi_double delta;  
    hi_vpc_bord_type border_type;  
    hi_vpc_scalar scalar_value;
```

```
    hi_u32 reserved[2];  
} hi_filter_2d_config;
```

成员

成员名称	描述
filter	自定义卷积核。 由于filter是5*5的数组，当kernel_size内的宽、高不同时为5时，二维数组内的中间数据有效，例如： - 宽为1、高为3时，下图中的黄区区域数据有效：  - 宽为3、高为3时，下图中的黄色区域数据有效： 
kernel_size	卷积核尺寸，kernel_size的宽和高只能设置为1，3或者5。
anchor	卷积核锚点，预留字段，需要设置成(-1, -1)。
delta	增量值，会被加到执行卷积之后的每个像素值之上。
border_type	边界填充类型，只支持HI_BORDER_CONSTANT和HI_BORDER_REPLICATE两种填充模式。
scalar_value	边界填充值。
reserved[2]	预留参数，为保证后续版本兼容性，请务必使用memset结构体方式进行清零初始化，在代码中必须避免显式对reserved字段进行访问。

1.9.20.12.44 hi_mosaic_param

说明

定义马赛克信息结构体。

定义

```
typedef struct {  
    hi_vpc_pic_info src;  
    hi_vpc_pic_info dst;  
    hi_u32 count;  
    hi_mosaic *mosaic;  
} hi_mosaic_param;
```

成员

成员名称	描述
src	输入图片信息。
dst	输出图片信息。
count	马赛克区域个数，取值范围[1, 100]。
mosaic	马赛克区域配置参数数组指针。

1.9.20.12.45 hi_rotate_param

说明

定义固定角度旋转信息结构体。

定义

```
typedef struct {  
    hi_vpc_pic_info src;  
    hi_vpc_pic_info dst;  
    hi_rotation angle;  
} hi_rotate_param;
```

成员

成员名称	描述
src	输入图片信息。
dst	输出图片信息。
angle	旋转角度。

1.9.20.12.46 hi_mosaic

说明

定义马赛克参数结构体。

定义

```
typedef struct {  
    hi_blk_size blk_size;  
    hi_rect rect;  
} hi_mosaic;
```

成员

成员名称	描述
blk_size	单个马赛克的大小，只支持HI_BLK_SIZE_8、HI_BLK_SIZE_16、HI_BLK_SIZE_32。
rect	马赛克区域，其宽、高最小为8，可以部分区域在图片外。

1.9.20.12.47 hi_cover_type

说明

定义覆盖类型。

定义

```
typedef enum {  
    HI_COVER_RECT = 0,  
    HI_COVER_QUAD,  
} hi_cover_type;
```

成员

成员名称	描述
HI_COVER_RECT	矩形覆盖。
HI_COVER_QUAD	任意四边形覆盖。

1.9.20.12.48 hi_cover_param

说明

定义覆盖信息结构体。

定义

```
typedef struct {  
    hi_vpc_pic_info src;  
    hi_vpc_pic_info dst;  
    hi_u32 count;  
    hi_cover *cover;  
} hi_cover_param;
```

成员

成员名称	描述
src	输入图片信息。

成员名称	描述
dst	输出图片信息。
count	覆盖区域个数，取值范围[1, 100]。
cover	覆盖区域配置参数数组指针。

1.9.20.12.49 hi_cover

说明

定义覆盖参数结构体。

定义

```
typedef struct {
    hi_cover_type type;
    union {
        hi_rect rect;
        hi_quad_cover quad;
    };
    hi_u32 color;
} hi_cover;
```

成员

成员名称	描述
type	覆盖类型。
rect	覆盖区域。 当图片格式为YUV422SP、YUV420SP时，要求x、y、width、height值均为偶数。
quad	自定义覆盖区域。
color	覆盖区域的颜色值，低24位有效。

1.9.20.12.50 hi_quad_cover

说明

定义自定义覆盖区域结构体。

定义

```
typedef struct {
    hi_bool is_solid;
    hi_u32 thick;
    hi_point point[4];
} hi_quad_cover;
```

成员

成员名称	描述
is_solid	只支持is_solid设置为1。
thick	预留字段，暂不支持。
point[4]	四边形4个点的坐标。

1.9.20.12.51 hi_line_param

说明

定义画线信息结构体。

定义

```
typedef struct {  
    hi_vpc_pic_info src;  
    hi_vpc_pic_info dst;  
    hi_u32 count;  
    hi_line *line;  
} hi_line_param;
```

成员

成员名称	描述
src	输入图片信息。
dst	输出图片信息。
count	画线条数，取值范围[1, 100]。
line	画线配置参数数组指针。 当图片格式为YUV422SP、YUV420SP时，要求起始点、结束点坐标值为偶数。

1.9.20.12.52 hi_line

说明

定义画线参数结构体。

定义

```
typedef struct {  
    hi_point start_point;  
    hi_point end_point;  
    hi_u32 thick;  
    hi_u32 color;  
} hi_line;
```


成员

成员名称	描述
start_point	起始点坐标。
end_point	结束点坐标。 结束点坐标不能与起始点坐标相同。
thick	线的粗细值，取值需要2对齐。
color	线的颜色值，低24位有效。

1.9.20.12.53 hi_osd_param

说明

定义叠加信息结构体。

定义

```
typedef struct {  
    hi_vpc_pic_info src;  
    hi_vpc_pic_info dst;  
    hi_u32 count;  
    hi_osd *osd;  
    hi_u32 clut[16];  
} hi_osd_param;
```

成员

成员名称	描述
src	输入图片信息。
dst	输出图片信息。
count	叠加区域个数，取值范围[1, 100]。
osd	叠加区域配置参数数组指针。
clut	clut查找表系数。

1.9.20.12.54 hi_osd

说明

定义叠加参数结构体。

定义

```
typedef struct {  
    hi_rect rect;  
    hi_pixel_format pixel_format;
```

```
hi_void* picture_address;  
hi_u32 stride;  
hi_u32 bg_alpha;  
hi_u32 fg_alpha;  
hi_osd_inverted_color osd_inverted_color;  
} hi_osd;
```

成员

成员名称	描述
rect	叠加区域坐标。
pixel_format	被叠加图片的格式。 支持如下格式： HI_PIXEL_FORMAT_ARGB_8888 = 14, // ARGB8888 HI_PIXEL_FORMAT_ARGB_1555 = 33, // ARGB1555 A:1bit R:5bit G:5bit B:5bit HI_PIXEL_FORMAT_ARGB_4444 = 25, // ARGB4444 A:4bit R:4bit G:4bit B:4bit HI_PIXEL_FORMAT_ARGB_CLUT2 = 41, // ARGB Color Lookup Table 2bit HI_PIXEL_FORMAT_ARGB_CLUT4 = 42, // ARGB Color Lookup Table 4bit
picture_addresses	被叠加图片的地址。
stride	被叠加图片的stride。 - 对于HI_PIXEL_FORMAT_ARGB_8888格式，stride值为rect.width的4倍 - 对于HI_PIXEL_FORMAT_ARGB_1555/ HI_PIXEL_FORMAT_ARGB_4444格式，stride值为rect.width的2倍 - 对于HI_PIXEL_FORMAT_ARGB_CLUT2格式，stride值为rect.width的1/4倍 - 对于HI_PIXEL_FORMAT_ARGB_CLUT4格式，stride值为rect.width的1/2倍
bg_alpha	背景alpha。 取值范围[0,255]。 仅当被叠加图片的格式为HI_PIXEL_FORMAT_ARGB_1555格式时，需设置该参数。
fg_alpha	前景alpha。 取值范围[0,255]。 仅当被叠加图片的格式为HI_PIXEL_FORMAT_ARGB_1555格式时，需设置该参数。
osd_inverted_color	反色类型。

1.9.20.12.55 hi_osd_inverted_color

说明

定义OSD反色类型。

定义

```
typedef enum {  
    HI_OSD_INVERTED_COLOR_NONE = 0,  
    HI_OSD_INVERTED_COLOR_RGB,  
    HI_OSD_INVERTED_COLOR_ALPHA,  
    HI_OSD_INVERTED_COLOR_ALL,  
} hi_osd_inverted_color;
```

成员

成员名称	描述
HI_OSD_INVERTED_COLOR_NONE	不反色。
HI_OSD_INVERTED_COLOR_RGB	RGB通道反色。
HI_OSD_INVERTED_COLOR_ALPHA	ALPHA通道反色。
HI_OSD_INVERTED_COLOR_ALL	所有通道反色。

1.9.20.12.56 hi_transform_matrix

说明

定义仿射变换中的结构体。

定义

```
typedef struct {  
    hi_matrix_type matrix_type;  
    hi_float matrix[3][4];  
} hi_transform_matrix;
```

成员

成员名称	描述
matrix_type	矩阵类型。
matrix	仿射变换矩阵。

1.9.20.12.57 hi_matrix_type

说明

定义仿射变换中的矩阵类型。

定义

```
typedef enum {  
    HI_MATRIX_2_PLUS_3 = 0,  
    HI_MATRIX_3_PLUS_3,  
    HI_MATRIX_3_PLUS_4,  
} hi_matrix_type;
```

成员

成员名称	描述
HI_MATRIX_2_PLUS_3	2*3 Matrix。
HI_MATRIX_3_PLUS_3	3*3 Matrix。
HI_MATRIX_3_PLUS_4	3*4 Matrix。

1.9.20.12.58 hi_flip_param

说明

定义翻转信息结构体。

定义

```
typedef struct {  
    hi_vpc_pic_info src;  
    hi_vpc_pic_info dst;  
    hi_flip_mode flip_mode;  
} hi_flip_param;
```

成员

成员名称	描述
src	输入图片信息。
dst	输出图片信息。
flip_mode	翻转模式参数。

1.9.20.12.59 hi_flip_mode

说明

定义翻转模式类型。

定义

```
typedef enum {  
    HI_FLIP_HOR = 0,  
    HI_FLIP_VER = 1,  
    HI_FLIP_BOTH = 2  
} hi_flip_mode;
```

成员

成员名称	描述
HI_FLIP_HOR	进行水平方向翻转。
HI_FLIP_VER	进行竖直方向翻转。
HI_FLIP_BOTH	同时进行水平方向翻转和竖直方向翻转，相当于180度旋转。

1.9.20.12.60 hi_opt_attr

说明

定义通道属性信息。

定义

```
typedef struct {  
    hi_opt_attr_type attr_type;  
    hi_u32 len;  
    hi_void* value;  
} hi_opt_attr;
```

成员

成员名称	描述
attr_type	属性类型。
len	属性值数据的长度，当前固定为4。
value	属性值，当前仅支持设置为0。

1.9.20.12.61 hi_opt_attr_type

说明

定义属性类型。

定义

```
typedef enum {  
    HI_VPC_CHN_PRIORITY = 0  
} hi_opt_attr_type;
```

成员

成员名称	描述
HI_VPC_CHN_PRIORITY	VPC通道优先级属性，当前仅支持设置为0，表示将通道设置为高优先级

1.9.20.13 VDEC 视频/JPEGD 图像解码

1.9.20.13.1 hi_vdec_chn

说明

描述解码通道。

定义

```
typedef hi_s32 hi_vdec_chn;
```

1.9.20.13.2 hi_vdec_send_mode

说明

定义码流发送方式。

定义

```
typedef enum {  
    HI_VDEC_SEND_MODE_STREAM = 0,  
    HI_VDEC_SEND_MODE_FRAME ,  
    HI_VDEC_SEND_MODE_COMPAT,  
    HI_VDEC_SEND_MODE_BUTT  
}hi_vdec_send_mode;
```

成员

成员名称	描述
HI_VDEC_SEND_MODE_STREAM	流模式，当前不支持。
HI_VDEC_SEND_MODE_FRAME	帧模式，当前仅支持按帧发送。
HI_VDEC_SEND_MODE_COMPAT	兼容模式，当前不支持。
HI_VDEC_SEND_MODE_BUTT	保留值。

注意事项

1. 仅支持HI_VDEC_SEND_MODE_FRAME；
2. 当一帧码流的最后一包没有把end_of_frame为HI_TRUE时，还可以再发送一个end_of_frame为HI_TRUE的空包（注意带上当前帧的PTS（Presentation Time Stamp））给解码器内部标识当前帧码流已发送完毕。
3. 不支持一帧图像分多包进行发送的模式；

1.9.20.13.3 hi_vdec_frame_type

说明

定义码流发送方式。

定义

```
typedef enum {  
    HI_VDEC_FRAME_TYPE_I = 0,  
    HI_VDEC_FRAME_TYPE_P = 1,  
    HI_VDEC_FRAME_TYPE_B = 2,  
    HI_VDEC_FRAME_TYPE_BUTT  
}hi_vdec_frame_type;
```

成员

成员名称	描述
HI_VDEC_FRAME_TYPE_I	IDR帧。
HI_VDEC_FRAME_TYPE_P	P帧。
HI_VDEC_FRAME_TYPE_B	B帧。
HI_VDEC_FRAME_TYPE_BUTT	保留值。

1.9.20.13.4 hi_quick_mark_mode

说明

定义快速释放参考帧模式枚举。

定义

```
typedef enum {  
    HI_QUICK_MARK_ADAPT = 0,  
    HI_QUICK_MARK_FORCE,  
    HI_QUICK_MARK_NONE,  
    HI_QUICK_MARK_BUTT  
}hi_quick_mark_mode;
```

成员

成员名称	描述
HI_QUICK_MARK_ADAPT	自适应模式。预留参数，暂未支持。
HI_QUICK_MARK_FORCE	强制模式。预留参数，暂未支持。

成员名称	描述
HI_QUICK_MAR K_NONE	普通模式。预留参数，暂未支持。
HI_QUICK_MAR K_BUTT	保留值。

1.9.20.13.5 hi_vdec_pic_info

说明

定义视频原始图像帧结构。

定义

```
typedef struct {  
    hi_u32 width;  
    hi_u32 height;  
    hi_u32 width_stride;  
    hi_u32 height_stride;  
    hi_pixel_format pixel_format;  
    hi_u64 vir_addr;  
    hi_u32 buffer_size;  
    hi_s16 offset_top;  
    hi_s16 offset_bottom;  
    hi_s16 offset_left;  
    hi_s16 offset_right;  
}hi_vdec_pic_info;
```


成员

成员名称	描述
width	图像宽度，视频解码时使用该参数。 - Atlas 推理系列产品，视频解码场景下，支持对输出图片进行缩放，缩放算法默认为业界通用的Bilinear算法（与OpenCV算法的计算过程类似），解码图像经过缩放后的输出宽取值范围为[10, 4096]。 以下场景，VDEC不对输出图片进行缩放：在hi_vdec_pic_info结构体内，将width参数值和height参数值同时设置为0。 - Atlas A2训练系列产品/Atlas 800I A2推理产品，视频解码场景下，支持对输出图片进行缩放，缩放算法默认为业界通用的Bilinear算法（与OpenCV算法的计算过程类似），解码图像经过缩放后的输出宽取值范围为[10, 4096]。 以下场景，VDEC不对输出图片进行缩放：在hi_vdec_pic_info结构体内，将width参数值和height参数值同时设置为0。 - Atlas 200/500 A2推理产品，视频解码场景下，支持对输出图片进行缩放，缩放算法默认为业界通用的Bilinear算法（与OpenCV算法的计算过程类似），当输入码流宽度小于或等于4096时，解码图像经过缩放后的输出宽度范围为[10, 8192]；当输入码流宽度大于4096时，解码图像经过缩放后的输出宽度范围为[128, 8192]。 以下场景，VDEC不对输出图片进行缩放：在hi_vdec_pic_info结构体内，将width参数值和height参数值同时设置为0。
height	图像高度，视频解码时使用该参数。 - Atlas 推理系列产品，视频解码场景下，支持对输出图片进行缩放，缩放算法默认为业界通用的Bilinear算法（与OpenCV算法的计算过程类似），解码图像经过缩放后的输出高度范围为[6, 4096]。 - Atlas A2训练系列产品/Atlas 800I A2推理产品，视频解码场景下，支持对输出图片进行缩放，缩放算法默认为业界通用的Bilinear算法（与OpenCV算法的计算过程类似），解码图像经过缩放后的输出高度范围为[6, 4096]。 - Atlas 200/500 A2推理产品，视频解码场景下，支持对输出图片进行缩放，缩放算法默认为业界通用的Bilinear算法（与OpenCV算法的计算过程类似），当输入码流宽度小于或等于4096时，解码图像经过缩放后的输出高度范围为[6, 8192]；当输入码流宽度大于4096时，解码图像经过缩放后的输出高度范围为[128, 8192]。
width_stride	图像宽度步长，视频解码时，宽stride的配置约束请参见1.9.17.3 VDEC功能及约束说明。 取值范围：[32, 16384]。

成员名称	描述
height_stride	图像高度步长，视频解码时，高stride的配置约束请参见 1.9.17.3 VDEC功能及约束说明 。 取值范围：[6, 16384]。
pixel_format	图像像素格式。图像或视频解码时使用该参数。VDEC视频解码输出图片格式参见 1.9.17.3 VDEC功能及约束说明 ，JPEGD图片解码输出图片格式参见 1.9.17.1 JPEGD功能及约束说明 。
vir_addr	解码输出数据在内存中的起始地址。 隔行码流场景下，隔行码流每帧发送两场，带两块输出内存；解码时其中一块内存无图像输出，属于正常现象，用户需及时释放内存；隔行码流的解码输出数据都在奇数场对应的输出buffer中。
buffer_size	存放解码图像的缓冲区大小。
offset_top	解码区域相对原图的顶部偏移，JPEG图片解码场景下按指定区域解码时配置，如 图1-31 所示，若无需按指定区域解码，需将此值初始化为0。 Atlas 推理系列产品不支持该参数，为保证后续版本兼容性，建议此参数配置为0。
offset_bottom	解码区域相对原图的底部偏移，JPEG图片解码场景下按指定区域解码时配置，如 图1-31 所示，若无需按指定区域解码，需将此值初始化为0。 Atlas 推理系列产品不支持该参数，为保证后续版本兼容性，建议此参数配置为0。
offset_left	解码区域相对原图的左侧偏移，JPEG图片解码场景下按指定区域解码时配置，如 图1-31 所示，若无需按指定区域解码，需将此值初始化为0。 Atlas 推理系列产品不支持该参数，为保证后续版本兼容性，建议此参数配置为0。
offset_right	解码区域相对原图的右侧偏移，JPEG图片解码场景下按指定区域解码时配置，如 图1-31 所示，若无需按指定区域解码，需将此值初始化为0。 Atlas 推理系列产品不支持该参数，为保证后续版本兼容性，建议此参数配置为0。

注意事项

10bit的数据经过VPC处理（图像处理）之后均会变成8bit。

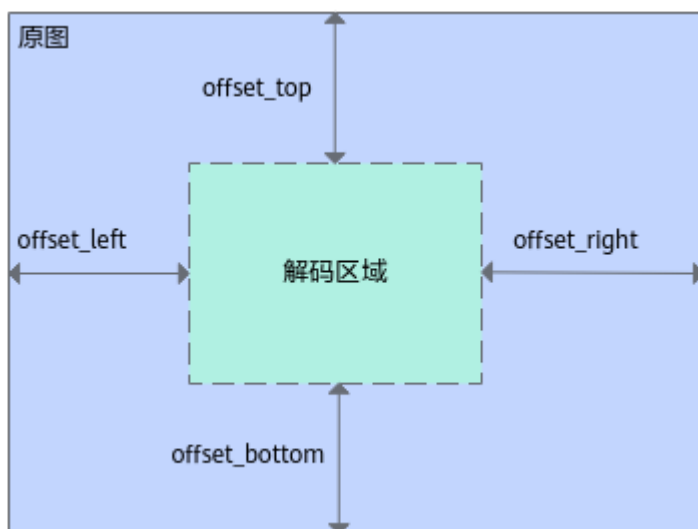
参考信息

JPEG图片解码场景下，按区域解码的功能示意图如下。

使用区域解码存在以下约束：

- 对于jpeg图片解码进行区域解码，不支持旋转、软件解码图片；
- 区域解码的范围不能超出原图大小，并且最小支持32*32，最大支持16384*16384；
- 使用以下图片格式时，对各偏移值的约束如下：
 - HI_PIXEL_FORMAT_YUV_SEMIPLANAR_420或HI_PIXEL_FORMAT_YVU_SEMIPLANAR_420格式：offset_top和offset_left必须为偶数，(height - offset_bottom) 和 (width - offset_right) 必须为偶数。
 - HI_PIXEL_FORMAT_YUV_SEMIPLANAR_422或HI_PIXEL_FORMAT_YVU_SEMIPLANAR_422格式：offset_left必须为偶数，(width - offset_right) 必须为偶数。
 - HI_PIXEL_FORMAT_YUV_SEMIPLANAR_440或HI_PIXEL_FORMAT_YVU_SEMIPLANAR_440：offset_top必须为偶数，(height - offset_bottom) 必须为偶数。

图 1-31 按区域解码功能示意图



1.9.20.13.6 hi_vdec_video_attr

说明

定义视频解码视频属性。

定义

```
typedef struct {  
    hi_u32 ref_frame_num;  
    hi_bool temporal_mvp_en;  
    hi_u32 tmv_buf_size;  
}hi_vdec_video_attr;
```

成员

成员名称	描述
ref_frame_num	参考帧的数目，用于决定解码时需要的参考帧个数，会较大的影响内存VB（ Video Buffer ）块占用，根据实际情况设置合适的值。如果此处设置的值大于实际码流中的参考帧数量，系统内部会根据实际码流的参考帧数量进行调整，以节省内存，但不超过当前所设置的参数值。 取值范围：[0, 16]，以帧为单位。 不知情的情况下，其他码流：推荐设为5。 测试码流：推荐设为16。 静态属性。
temporal_mvp_en	是否支持时域运动矢量预测。 取值范围：[0, 1]。 如果H.264解码不需要解码B帧，或者H.265解码不需要解码“支持时域运动矢量预测（sps_temporal_mvp_enabled_flag =1）的码流”，则配置temporal_mvp_en为0，否则配置为1。当配置为0时，可不分配输出TMV信息的VB块，节省内存。 注意：调用hi_mpi_vdec_set_chn_param接口设置通道参数时，只支持将hi_video_dec_mode设置为HI_VIDEO_DEC_MODE_IPB（表示解码I、P、B帧），因此解码H.264码流时只能将temporal_mvp_en配置为1。
tmv_buf_size	视频解码图像TMV（ Temporal Motion Vector ）缓存大小，以Byte为单位，temporal_mvp_en为1时有效。用户可先调用hi_vdec_get_tmv_buf_size接口获取视频解码图像TMV缓存大小。

1.9.20.13.7 hi_vdec_chn_attr

说明

定义解码通道属性结构体。

定义

```
typedef struct {
    hi_payload_type type;
    hi_vdec_send_mode mode;
    hi_u32 pic_width;
    hi_u32 pic_height;
    hi_u32 stream_buf_size;
    hi_u32 frame_buf_size;
    hi_u32 frame_buf_cnt;

    union {
        hi_vdec_video_attr video_attr;
    };
}hi_vdec_chn_attr;
```

成员

成员名称	描述
type	解码协议类型枚举值。 静态属性。
mode	码流发送方式。静态属性。 Atlas 推理系列产品上，H.264/H.265/JPEG仅支持按帧发送码流。 Atlas 200/500 A2推理产品上，VDEC视频解码场景下，H.264/H.265仅支持按帧发送码流；JPEG图片解码场景下，该参数预留。 Atlas A2训练系列产品/Atlas 800I A2推理产品上，VDEC视频解码场景下，H.264/H.265仅支持按帧发送码流；JPEG图片解码场景下，该参数预留
pic_width	通道支持的解码图像最大宽（以像素为单位）。静态属性。 Atlas 200/500 A2推理产品上，JPEG图片解码场景下，该参数预留。 Atlas A2训练系列产品/Atlas 800I A2推理产品上，JPEG图片解码场景下，该参数预留
pic_height	通道支持的解码图像最大高（以像素为单位）。静态属性。 Atlas 200/500 A2推理产品上，JPEG图片解码场景下，该参数预留。 Atlas A2训练系列产品/Atlas 800I A2推理产品上，JPEG图片解码场景下，该参数预留
stream_buf_size	输入码流缓存大小。 取值范围：大于或等于解码通道大小（宽*高）的 3/4倍，即YUV420图像大小的一半（宽*高*3/2*1/2），小于或等于1024*1024*1024，以 Byte 为单位。 推荐值：一幅YUV420解码图像大小。即：宽*高*1.5。 静态属性。
frame_buf_size	解码图像帧存大小，以Byte为单位。JPEGD图像解码不需要设置该参数。 取值范围：大于或等于0，取值为0时，VDEC内部将根据实际码流信息自动分配合适的帧存大小；取值大于0时，用户需保证（帧存大小*参考帧个数 >= 实际所需最小帧存大小*实际所需最小参考帧个数），否则不能正常解码。实际所需最小帧存大小，用户需先根据码流中的SPS（Sequence Parameter Set）信息获取输入码流的宽、高、比特位宽，再调用hi_vdec_get_pic_buf_size接口获取实际所需最小帧存大小。

成员名称	描述
frame_buf_cnt	解码图像帧存个数。JPEG图像解码不需要设置该参数。 取值范围：大于或等于0。 • 一般情况下，设置为0，VDEC内部将根据实际码流信息自动分配合适的帧存个数； • 如果客户需要限制VDEC解码帧存大小，则设置为大于0的值，用户需保证（帧存大小*参考帧个数 >= 实际需要最小帧存大小*实际需要最小参考帧个数），否则不能正常解码。实际所需最小参考帧个数，用户需先根据码流中的SPS（Sequence Parameter Set）信息获取码流参考帧个数，H.264/H.265解码所需帧存个数=参考帧+显示帧+1。
video_attr	视频(H.264/H.265)解码通道属性。

注意事项

设置通道属性的图像宽和高，若用户清楚码流的具体宽高，建议以实际值来设置。若是不清楚，建议以此码流的最大规格来创建。但是最大规格是4096*4096，不能超过此值。

1.9.20.13.8 hi_h264_protocol_param

说明

与 H.264 协议相关的内存分配参数。

定义

```
typedef struct {  
    hi_s32 max_slice_num;  
    hi_s32 max_sps_num;  
    hi_s32 max_pps_num;  
}hi_h264_protocol_param;
```

成员

成员名称	描述
max_slice_num	该通道解码支持的最大Slice个数。 取值范围：[1, 300]。 Default：16。
max_sps_num	该通道解码支持的最大SPS个数。 取值范围：[1, 32]。 Default：2。
max_pps_num	该通道解码支持的最大PPS个数。 取值范围：[1, 256]。 Default：2。

1.9.20.13.9 hi_h265_protocol_param

说明

定义与 H.265 协议相关的内存分配参数。

定义

```
typedef struct {  
    hi_s32 max_slice_segment_num;  
    hi_s32 max_vps_num;  
    hi_s32 max_sps_num;  
    hi_s32 max_pps_num;  
}hi_h265_protocol_param;
```

成员

成员名称	描述
max_slice_segment_num	该通道解码支持的最大SliceSegment个数。 取值范围：[1, 600]。 Default：16。
max_vps_num	该通道解码支持的最大VPS个数。 取值范围：[1, 16]。 Default：2。
max_sps_num	该通道解码支持的最大SPS个数。 取值范围：[1, 16]。 Default：2。
max_pps_num	该通道解码支持的最大PPS个数。 取值范围：[1, 64]。 Default：2。

1.9.20.13.10 hi_vdec_protocol_param

说明

定义与协议相关的内存分配参数。

定义

```
typedef struct {  
    hi_payload_type type;  
    union  
    {  
        hi_h264_protocol_param h264_param;  
        hi_h265_protocol_param h265_param;  
    };  
}hi_vdec_protocol_param;
```

成员

成员名称	描述
type	解码通道支持的协议。
h264_param	H.264协议参数。
h265_param	H.265协议参数。

1.9.20.13.11 hi_video_dec_mode

说明

定义视频解码模式枚举。

定义

```
typedef enum {  
    HI_VIDEO_DEC_MODE_IPB = 0,  
    HI_VIDEO_DEC_MODE_IP,  
    HI_VIDEO_DEC_MODE_I,  
    HI_VIDEO_DEC_MODE_BUTT  
}hi_video_dec_mode;
```

成员

成员名称	描述
HI_VIDEO_DEC_MODE_IPB	IPB模式，即I、P、B帧都解码。
HI_VIDEO_DEC_MODE_IP	IP模式，即只解码I帧和P帧。暂不支持。
HI_VIDEO_DEC_MODE_I	I模式，即只解码I帧。暂不支持。
HI_VIDEO_DEC_MODE_BUTT	保留值。

1.9.20.13.12 hi_video_out_order

说明

定义视频解码输出顺序枚举。

定义

```
typedef enum {  
    HI_VIDEO_OUT_ORDER_DISPLAY = 0,  
    HI_VIDEO_OUT_ORDER_DEC,  
    HI_VIDEO_OUT_ORDER_BUTT  
}hi_video_out_order;
```


成员

成员名称	描述
HI_VIDEO_OUT_ORDER_DISPLAY	显示序输出。
HI_VIDEO_OUT_ORDER_DEC	解码序输出。
HI_VIDEO_OUT_ORDER_BUTT	保留值。

注意事项

解码有 B 帧的码流应设置为显示序输出。

1.9.20.13.13 hi_vdec_video_param

说明

定义视频解码高级参数。

定义

```
typedef struct {  
    hi_bool composite_dec_en;  
    hi_bool slice_input_en;  
    hi_s32 err_threshold;  
    hi_video_dec_mode dec_mode;  
    hi_video_out_order out_order;  
    hi_compress_mode compress_mode;  
    hi_video_format video_format;  
    hi_quick_mark_mode quick_mark_mode;  
} hi_vdec_video_param;
```

成员

成员名称	描述
composite_dec_en	复合码流使能开关。预留参数，暂未支持。
slice_input_en	预留参数，暂未支持。
err_threshold	预留参数，暂未支持。
dec_mode	解码模式。 默认值：HI_VIDEO_DEC_MODE_IPB。
out_order	解码图像输出顺序。 默认值：HI_VIDEO_OUT_ORDER_DISPLAY。

成员名称	描述
compress_mode	解码图像压缩模式。支持HFBC压缩 HI_COMPRESS_MODE_HFBC。 默认值：HI_COMPRESS_MODE_HFBC。
video_format	解码图像数据格式。 当前仅支持HI_VIDEO_FORMAT_TILE_64x16和 HI_VIDEO_FORMAT_LINEAR。 默认值：HI_VIDEO_FORMAT_TILE_64x16。
quick_mark_mode	快速释放参考帧模式。预留参数，暂未支持。

1.9.20.13.14 hi_vdec_pic_param

说明

定义图像解码高级参数。

定义

```
typedef struct {  
    hi_pixel_format pixel_format;  
    hi_u32 alpha;  
}hi_vdec_pic_param;
```

成员

成员名称	描述
pixel_format	图片像素格式。预留参数。
alpha	RGB解码透明度。预留参数。

1.9.20.13.15 hi_vdec_chn_param

说明

定义解码通道高级参数。

定义

```
typedef struct {  
    hi_payload_type type;  
    hi_u32 display_frame_num;  
    union{  
        hi_vdec_video_param video_param;  
        hi_vdec_pic_param pic_param;  
    };  
}hi_vdec_chn_param;
```

成员

成员名称	描述
type	解码协议。
display_frame_num	解码后缓存图像帧数。 取值范围：[0, 16]。 默认值：2。
video_param	视频(H.264/H.265)解码高级参数。
pic_param	图片(JPEG)解码高级参数。

1.9.20.13.16 hi_vdec_dec_err

说明

定义解码错误信息结构体。

定义

```
typedef struct {  
    hi_s32 set_pic_size_err;  
    hi_s32 set_protocol_num_err;  
    hi_s32 set_ref_num_err;  
    hi_s32 set_pic_buf_size_err;  
    hi_s32 format_err;  
    hi_s32 stream_unsupport;  
    hi_s32 pack_err;  
    hi_s32 stream_size_over;  
    hi_s32 stream_not_release;  
} hi_vdec_dec_err;
```

成员

成员	描述
set_pic_size_err	图像的宽（或高）比通道的宽（或高）大，预留参数。
set_protocol_num_err	设置的协议参数个数不够。比如：Slice/PPS/SPS个数。此参数仅对H.264/H.265解码有效。
set_ref_num_err	设置的参考帧个数不够。此参数仅对H.264/H.265解码有效。
set_pic_buf_size_err	图像buffer内存大小不够。
format_err	不支持的格式。JPEG的Progressive码流。
stream_unsupport	不支持的规格（码流规格与昇腾AI处理器宣称支持的规格不一致）。
pack_err	码流有错。

成员	描述
stream_size_ove r	一帧码流太大了，当整个SCD（ Start Code Detection ） buffer都装不下一帧码流时，强制清空SCDbuffer。 此参数仅对H.264/H.265解码有效。
stream_not_rele ase	VFMW（ video firmware ）内部管理码流错误，出现长时间不释放码流的情况，预留参数。

1.9.20.13.17 hi_vdec_chn_status

说明

定义通道状态结构体。

定义

```
typedef struct {  
    hi_payload_type type;  
    hi_u32 left_stream_bytes;  
    hi_u32 left_stream_frames;  
    hi_u32 left_decoded_frames;  
    hi_bool is_started;  
    hi_u32 recv_stream_frames;  
    hi_u32 dec_stream_frames;  
    hi_vdec_dec_err dec_err;  
    hi_u32 width;  
    hi_u32 height;  
    hi_u64 latest_frame_pts;  
} hi_vdec_chn_status;
```

成员

成员名称	描述
type	解码协议。
left_stream_bytes	码流buffer中待解码的byte数，包括正在解码的当前帧中未解码的byte数。
left_stream_frames	码流buffer中待解码的帧数，不包括正在解码的当前帧。-1表示无效。仅按帧发送时有效。
left_decoded_frame s	图像buffer中剩余的图片数目。
is_started	解码器是否已经启动接收码流。
recv_stream_frames	码流buffer中已接收码流帧数。-1表示无效。仅按帧发送时有效。
dec_stream_frames	码流buffer中已解码帧数。仅按帧发送时有效。
dec_err	解码错误信息。
width	解码后输出图片的宽。

成员名称	描述
height	解码后输出图片的高。
latest_frame_pts	最新解码图像的时间戳。预留参数，暂不支持。

1.9.20.13.18 hi_vdec_stream

说明

定义视频/图像解码的码流结构体。

定义

```
typedef struct {  
    hi_bool end_of_frame;  
    hi_bool end_of_stream;  
    hi_bool need_display;  
    hi_u64 pts;  
    hi_u64 private_data;  
    hi_u32 len;  
    hi_u8 *ATTRIBUTE addr;  
} hi_vdec_stream;
```

成员

成员	描述
end_of_frame	当前帧是否结束（此值目前暂无效）。
end_of_stream	是否发完所有码流。
need_display	当前帧输出是否显示。 0：不显示，设置为0，输出Buffer无解码结果，且不显示帧按照显示序或解码序输出。 1：显示，设置为1，才能输出解码结果，且显示帧按照显示序或解码序输出。 但如果解码码流中同时存在不显示帧、显示帧，则系统不保证不显示帧与显示帧之间按照显示序或解码序输出。
pts	码流包的时间戳，以us为单位。仅按帧发送时有效。
private_data	私有数据。预留参数，暂不支持。
len	码流包的长度，以Byte为单位。
addr	码流包的地址。

注意事项

- 1. 按帧发送时，解码图像的时间戳等于码流包中的时间戳。
- 2. 按流发送时，解码图像的时间戳等于0。

3. 当发完所有码流后，把end_of_stream设置为1，表示码流文件结束，这时解码器会解完发送下来的所有码流并输出所有图像。如果发完所有码流后把end_of_stream设置为0，解码器内部可能残余大于等于一帧的图像未解码输出，因为解码器必须等到下一帧码流到来才能知道当前帧已经结束，送入解码。
4. VDEC支持发送一包end_of_stream为1的空码流包（地址为空或长度为 0）。

1.9.20.13.19 hi_vdec_supplement_info

说明

定义输出帧补充信息，暂不支持。

定义

```
typedef struct {  
    hi_payload_type type;  
    union {  
        hi_vdec_video_supplement_info video_supplement_info;  
    };  
} hi_vdec_supplement_info;
```

1.9.20.13.20 hi_vdec_video_supplement_info

说明

定义输出视频帧补充信息，暂不支持。

定义

```
typedef struct {  
    hi_vdec_frame_type frame_type;  
    hi_u32 err_rate;  
    hi_u32 poc;  
}hi_vdec_video_supplement_info;
```

成员

成员名称	描述
frame_type	解码帧类型。预留参数，暂不支持。
err_rate	错误率。预留参数，暂不支持。
poc	poc值。预留参数，暂不支持。

1.9.20.13.21 hi_jpegd_precision_mode

说明

定义JPEGD解码输出图片的宽、高对齐模式。

定义

```
typedef enum {  
    YUVOUT_ALIGN_DOWN = 0,
```

```
YUVOUT_ALIGN_UP = 1,  
YUVOUT_ALIGN_DOWN_COMPAT = 2,  
} hi_pegd_precision_mode;
```

对齐模式取值的含义如下：

- YUVOUT_ALIGN_DOWN：默认值
 - 解码后的输出图片格式为YUV420SP时，若源图的宽、高为奇数时，将输出图片的宽、高向下2对齐
 - 解码后的输出图片格式为YUV422SP时，若源图的宽为奇数时，将输出图片的宽向下2对齐
 - 解码后的输出图片格式为YUV440SP时，若源图的高为奇数时，将输出图片高向下2对齐
- YUVOUT_ALIGN_UP：
 - 解码后的输出图片格式为YUV420SP时，若源图的宽、高为奇数时，将输出图片的宽、高向上2对齐
 - 解码后的输出图片格式为YUV422SP时，若源图的宽为奇数时，将输出图片的宽向上2对齐
 - 解码后的输出图片格式为YUV440SP时，若源图的高为奇数时，将输出图片高向上2对齐
- YUVOUT_ALIGN_DOWN_COMPAT：
兼容旧版本，解码后的输出图片格式为YUV420SP/YUV422SP/YUV440SP时，若源图的宽、高为奇数时，将输出图片的宽、高向下2对齐。

1.9.20.14 VENC 视频/JPEGE 图像编码

1.9.20.14.1 hi_venc_chn

说明

描述编码通道。

定义

```
typedef hi\_s32 hi_venc_chn;
```

1.9.20.14.2 hi_venc_stream

说明

定义帧码流类型结构体。

定义

```
typedef struct {  
    hi\_venc\_pack ATTRIBUTE* pack;  
    hi\_u32 ATTRIBUTE pack_cnt;  
    hi\_u32 seq;  
  
    union {  
        hi\_venc\_h264\_stream\_info h264_info;  
        hi\_venc\_jpeg\_stream\_info jpeg_info;  
        hi\_venc\_h265\_stream\_info h265_info;  
    };  
};
```

```
    hi_venc_prores_stream_info prores_info;
};

union {
    hi_venc_h264_adv_stream_info h264_adv_info;
    hi_venc_h265_adv_stream_info h265_adv_info;
};
} hi_venc_stream;
```

成员

成员名称	描述
pack	帧码流包结构。
pack_cnt	一帧码流的所有包的个数。
seq	码流序列号。 按帧获取时，此处为帧序号；按包获取时，此处为包序号。
h264_info/h265_info	码流特征信息。 Atlas 推理系列产品不支持该参数。 Atlas A2训练系列产品/Atlas 800I A2推理产品不支持该参数。
jpeg_info/prores_info	码流特征信息，暂不支持。
h264_adv_info/ h265_adv_info	码流高级特征信息。 Atlas 推理系列产品不支持该参数。 Atlas A2训练系列产品/Atlas 800I A2推理产品不支持该参数。

1.9.20.14.3 hi_venc_pack

说明

定义帧码流包结构体。

定义

```
typedef struct {
    union {
        hi_u64 phys_addr;
        hi_u64 input_addr;
    };
    hi_u8 ATTRIBUTE* addr;
    hi_u32 ATTRIBUTE len;
    hi_u64 pts;
    hi_bool is_frame_end;
    hi_venc_data_type data_type;
    hi_u32 offset;
    hi_u32 data_num;
    hi_venc_pack_info pack_info[HI_VENC_MAX_PACK_INFO_NUM];
} hi_venc_pack;
```


成员

成员名称	描述
phys_addr	码流包物理地址。
input_addr	用户输入码流数据的内存地址。 如果用户需要回收输入码流数据对应的内存，在得到编码码流结果之后，可以通过该参数获取原始YUV码流对应数据的用户态虚拟地址。
addr	码流包首地址。
len	码流包长度。 当len=0，表示视频/图像编码失败。
pts	时间戳。单位： us。
is_frame_end	帧结束标识。 取值范围： - HI_TRUE：该码流包是该帧的最后一个包。 - HI_FALSE：该码流包不是该帧的最后一个包。
data_type	码流类型，支持 H.265/H.264/JPEG协议类型的数据包。
offset	码流包中有效数据与码流包首地址addr的偏移。
data_num	当前码流包（当前包的类型由 data_type 指定）数据中包含码流包的个数。
pack_info[HI_VENC_MAX_PACK_INFO_NUM]	当前码流包数据中包含码流包数据信息。 #define HI_VENC_MAX_PACK_INFO_NUM 8

1.9.20.14.4 hi_venc_chn_attr

说明

定义编码通道属性结构体。

定义

```
typedef struct {  
    hi_venc_attr venc_attr;  
    hi_venc_rc_attr rc_attr;  
    hi_venc_gop_attr gop_attr;  
}hi_venc_chn_attr;
```

成员

成员名称	描述
venc_attr	编码器属性。 编码器属性中除通道宽高（pic_width和pic_height）外都是静态属性，一旦创建编码通道成功，静态属性不支持被修改，除非该通道被销毁，重新创建。
rc_attr	码率控制器属性。 码率控制器属性首先需要配置RC（Rate Control）模式，JPEG不需要配置码率控制器属性，其它协议类型通道（H.264/H.265）都必须配置。码率控制器属性RC模式必须与编码器属性协议类型匹配。
gop_attr	GOP Mode类型的结构体。 对于H264、H265协议，需要配置该参数。

1.9.20.14.5 hi_venc_attr

说明

定义编码器属性结构体。

定义

```
typedef struct {  
    hi_payload_type type;  
    hi_u32 max_pic_width;  
    hi_u32 max_pic_height;  
    hi_u32 buf_size;  
    hi_u32 profile;  
    hi_bool is_by_frame;  
    hi_u32 pic_width;  
    hi_u32 pic_height;  
    union {  
        hi_venc_h264_attr h264_attr;  
        hi_venc_h265_attr h265_attr;  
        hi_venc_jpeg_attr jpeg_attr;  
        hi_venc_prores_attr prores_attr;  
    };  
}hi_venc_attr;
```

成员

成员名称	描述
type	编码协议类型。
max_pic_width	编码图像最大宽度，静态属性，必须是2的整数倍。 取值范围：[MIN_WIDTH, MAX_WIDTH]，以像素为单位。 MIN_WIDTH、MAX_WIDTH，分别表示编码通道支持的最小宽度、最大宽度，参见表1-83。

成员名称	描述
max_pic_height	除jpeg协议外，编码图像最大高度，静态属性，必须是2的整数倍。 取值范围：[MIN_HEIGHT, MAX_HEIGHT]，以像素为单位。MIN_HEIGHT、MAX_HEIGHT分别表示编码通道支持的最小高度、最大高度，参见表1-83。
buf_size	码流buffer大小，单位为Byte，静态属性。 - 调用hi_mpi_venc_send_jpege_frame接口发送原始图像进行图像编码时，在创建通道时，buf_size参数值必须设置为0。 - 调用hi_mpi_venc_send_frame接口发送原始图像进行视频或图像编码时，在创建通道时，buf_size参数值必须设置为64的整数倍，取值范围：[Min, Max]，以Byte为单位。 (1) 视频编码场景下，Min当前表示32*1024，Max当前表示1*1024*1024*1024。 用户可以在创建通道时，合理设置每个通道的buf_size，以节省内存使用开销，推荐将buf_size设置为：原图宽*原图高*3/2后再64对齐，但buf_size依然需要在[Min, Max]范围内。 视频编码场景下，若buf_size设置的不合理，可能出现以下异常情况： - 反复重编，进而导致编码时延变长、性能下降、图像质量下降； - 编码失败，获取不到编码结果。 (2) 图像编码场景下，如果原图格式为YUV420，Min值为原图宽16对齐*原图高16对齐*3/2；如果原图格式为YUV422 Packed，Min值为原图宽16对齐*原图高16对齐*2。Max当前表示1*1024*1024*1024。 用户可以在创建通道时根据业务场景中图片的实际最大分辨率，合理设置每个通道的buf_size，以节省内存使用开销，推荐将buf_size设置为：原图宽*原图高*倍数，倍数推荐为5。 例如：用户图片的分辨率有720p (1280*720)、1080p (1080*1920)、4K (3840*2160) 大小的图片，需要的buf_size应按照4k配置：3840*2160*5。 图片编码场景下，buf_size最小值为320*240*5，防止小分辨率场景下，buffer容易被占满，影响编码功能执行，例如：用户图片的最大分辨率为128*128，推荐用户将buf_size设置为320*240*5。

成员名称	描述
profile	编码的等级，数字越大编码质量越好，静态属性。 • H.264 取值范围：[0, 2]。 - 0: Baseline。当取值为该值时，不支持编码B帧。 - 1: Main Profile。 - 2: High Profile。 • H.265 取值范围：[0,1]。 - 0: Main Profile。 - 1: Main 10 Profile，预留给，暂不支持。 • Jpeg 取值范围：0（表示Baseline）。
is_by_frame	帧/包模式获取码流，静态属性。 • HI_TRUE：按帧获取。 • HI_FALSE：按包获取。
pic_width	编码通道宽度，必须是MIN_ALIGN的整数倍。通道创建后，VENC视频编码支持修改通道宽度，JPEGE图片编码不支持修改通道宽度。 取值范围：[MIN_WIDTH, max_pic_width]，以像素为单位。MIN_WIDTH、max_pic_width、MIN_ALIGN分别表示编码通道支持的最小宽度、设置的编码图像最大宽度、最小对齐单元（像素），参见表1-83。 JPEG通道宽高的设置要满足：$pic\_width * pic\_height \leq max\_pic\_width * max\_pic\_height$。
pic_height	除jpeg协议外，编码通道高度，必须是MIN_ALIGN的整数倍。通道创建后VENC视频编码支持修改通道高度，JPEGE图片编码不支持修改通道高度。 取值范围：[MIN_HEIGHT, max_pic_height]，以像素为单位。MIN_HEIGHT、max_pic_height、MIN_ALIGN分别表示编码通道支持的最小高度、设置的编码图像最大高度、最小对齐单元（像素），参见表1-83。 JPEG通道宽高的设置要满足：$pic\_width * pic\_height \leq max\_pic\_width * max\_pic\_height$。
h264_attr	编码协议的属性。 Atlas 推理系列产品，预留给参数，暂不支持。 Atlas A2训练系列产品/Atlas 800I A2推理产品，预留给参数，暂不支持。
h265_attr	编码协议的属性。 Atlas 推理系列产品，预留给参数，暂不支持。 Atlas A2训练系列产品/Atlas 800I A2推理产品，预留给参数，暂不支持。
jpeg_attr	编码协议的属性。 预留给参数，暂不支持。

成员名称	描述
prores_attr	编码协议的属性。 预留参数，暂不支持。

参考信息

MIN_WIDTH、MAX_WIDTH、MIN_HEIGHT、MAX_HEIGHT、MIN_ALIGN分别表示编码通道支持的最小宽度、最大宽度、最小高度、最大高度、最小对齐单元（像素）。

表 1-83 编码通道宽高

W/H (pixel)	H.264	H.265	JPEG
MIN_WIDTH	128	128	32
MAX_WIDTH	4096	4096	8192
MIN_HEIGHT	128	128	32
MAX_HEIGHT	4096	4096	8192
MIN_ALIGN	2	2	2

1.9.20.14.6 hi_venc_rc_attr

说明

定义编码通道码率控制器属性。

定义

```
typedef struct {
    hi_venc_rc_mode rc_mode;
    union {
        hi_venc_h264_cbr h264_cbr;
        hi_venc_h264_vbr h264_vbr;
        hi_venc_h264_avbr h264_avbr;
        hi_venc_h264_qvbr h264_qvbr;
        hi_venc_h264_cvbr h264_cvbr;
        hi_venc_h264_fixqp h264_fixqp;
        hi_venc_h264_qpmap h264_qpmap;

        hi_venc_mjpeg_cbr mjpeg_cbr;
        hi_venc_mjpeg_vbr mjpeg_vbr;
        hi_venc_mjpeg_fixqp mjpeg_fixqp;

        hi_venc_h265_cbr h265_cbr;
        hi_venc_h265_vbr h265_vbr;
        hi_venc_h265_avbr h265_avbr;
        hi_venc_h265_qvbr h265_qvbr;
        hi_venc_h265_cvbr h265_cvbr;
        hi_venc_h265_fixqp h265_fixqp;
        hi_venc_h265_qpmap h265_qpmap;
    };
};
```

```
};  
} hi_venc_rc_attr;
```

成员

成员名称	描述
rc_mode	RC模式。 • CBR (Constant Bit Rate) 固定比特率。即在码率统计时间内保证编码码率平稳。 • VBR (Variable Bit Rate) 可变比特率，即允许在码率统计时间内编码码率波动，从而保证编码图像质量平稳。 • AVBR (Adaptive Variable Bit Rate) 自适应可变比特率，即允许在码率统计时间内编码码率波动，从而保证编码图像质量平稳。码率控制内部会检测当前场景的运动静止状态，在运动时用较高码率编码，在静止时主动降低目标码率。 • QVBR (Quality Variable Bit Rate) 基于主观图像质量的可变比特率，该码控调节方式是利用实时统计的 PSNR (图像质量客观评价指标) 的大小来动态调整码率，从而保证编码图像质量平稳。在PSNR较小时主动升高目标码率，PSNR较大时主动降低目标码率。 • CVBR (Constrained Variable Bit Rate) 是以VBR为基础，旨在提供平稳的图像质量的码控算法，同时对VBR的码率进行限制，以满足传输带宽以及存储空间的要求。具体来说， CVBR设置了瞬时，短期与长期码率的限制。其中，瞬时码率的限制保证了网络带宽对传输的要求；长期码率限制保证了在长时间视频录制时，存储设备有足够的空间储存数据；同时，短期码率会根据长期码率的设置和实际使用情况进行调节，以在场景复杂的情况下提供更加平稳的图像质量，并在场景简单时节省码率。 • FIXQP (Fix Quantisation Parameter) 固定QP值。在码率统计时间内，编码图像所有宏块QP值相同，采用用户设定的图像QP值，I帧和P帧的QP值可以分别设置。 并且对于不同协议，相同 RC 模式的属性变量基本一致。表1-84介绍了几种RC模式的公共的属性。
h264_cbr	H.264协议编码通道CBR模式属性。
h264_vbr	H.264协议编码通道VBR模式属性。
h264_avbr	H.264协议编码通道AVBR模式属性。
h264_qvbr	H.264协议编码通道QVBR模式属性。
h264_cvbr	H.264协议编码通道CVBR模式属性。
h264_fixqp	H.264协议编码通道FIXQP模式属性。 Atlas 推理系列产品，预留参数，暂不支持。 Atlas A2训练系列产品/Atlas 800I A2推理产品，预留参数，暂不支持。
h264_qpmap	H.264协议编码通道QP MAP模式属性。预留参数，暂不支持。

成员名称	描述
mjpeg_cbr	Mjpeg协议编码通道CBR模式属性。预留参数，暂不支持。
mjpeg_vbr	Mjpeg协议编码通道VBR模式属性。预留参数，暂不支持。
mjpeg_fixqp	Mjpeg (Motion Joint Photographic Experts Group) 协议编码通道FIXQP模式属性。预留参数，暂不支持。
h265_cbr	H.265协议编码通道CBR模式属性。
h265_vbr	H.265协议编码通道VBR模式属性。
h265_avbr	H.265协议编码通道AVBR模式属性。
h265_qvbr	H.265协议编码通道QVBR模式属性。
h265_cvbr	H.265协议编码通道CVBR模式属性。
h265_fixqp	H.265协议编码通道FIXQP模式属性。 Atlas 推理系列产品，预留参数，暂不支持。 Atlas A2训练系列产品/Atlas 800I A2推理产品，预留参数，暂不支持。
h265_qpmap	H.265协议编码通道QPMP模式属性。预留参数，暂不支持。

参考信息

表 1-84 码率控制器属性的约束

RC 模式	GOP	StatTime(s)	FrmRate(src_frame_rate/ dst_frame_rate)
CBR	≥1	≥1	src_frame_rate ≥ dst_frame_rate
VBR	≥1	≥1	src_frame_rate ≥ dst_frame_rate
AVBR	≥1	≥1	src_frame_rate ≥ dst_frame_rate
FIXQP	≥1	≥1	src_frame_rate ≥ dst_frame_rate
QPMP	≥1	≥1	src_frame_rate ≥ dst_frame_rate

1.9.20.14.7 hi_venc_rc_mode

说明

定义编码通道码率控制器模式。

定义

```
typedef enum {  
    HI_VENC_RC_MODE_H264_CBR = 1,  
    ...  
}
```

```
HI_VENC_RC_MODE_H264_VBR,  
HI_VENC_RC_MODE_H264_AVBR,  
HI_VENC_RC_MODE_H264_QVBR,  
HI_VENC_RC_MODE_H264_CVBR,  
HI_VENC_RC_MODE_H264_FIXQP,  
HI_VENC_RC_MODE_H264_QPMAP,  
  
HI_VENC_RC_MODE_MJPEG_CBR,  
HI_VENC_RC_MODE_MJPEG_VBR,  
HI_VENC_RC_MODE_MJPEG_FIXQP,  
  
HI_VENC_RC_MODE_H265_CBR,  
HI_VENC_RC_MODE_H265_VBR,  
HI_VENC_RC_MODE_H265_AVBR,  
HI_VENC_RC_MODE_H265_QVBR,  
HI_VENC_RC_MODE_H265_CVBR,  
HI_VENC_RC_MODE_H265_FIXQP,  
HI_VENC_RC_MODE_H265_QPMAP,  
  
HI_VENC_RC_MODE_BUTT,  
} hi_venc_rc_mode;
```

成员

成员名称	描述
HI_VENC_RC_MODE_H264_CBR	H.264 CBR模式。
HI_VENC_RC_MODE_H264_VBR	H.264 VBR模式。
HI_VENC_RC_MODE_H264_AVBR	H.264 AVBR模式。
HI_VENC_RC_MODE_H264_QVBR	H.264 QVBR模式。
HI_VENC_RC_MODE_H264_CVBR	H.264 CVBR模式。
HI_VENC_RC_MODE_H264_FIXQP	H.264 FIXQP模式。 Atlas 推理系列产品，预留参数，暂不支持。 Atlas A2训练系列产品/Atlas 800I A2推理产品，预留参数，暂不支持。
HI_VENC_RC_MODE_H264_QPMAP	H.264 QPMAP模式。预留参数，暂不支持。
HI_VENC_RC_MODE_MJPEG_CBR	MJPEG CBR模式。预留参数，暂不支持。
HI_VENC_RC_MODE_MJPEG_VBR	MJPEG VBR模式。预留参数，暂不支持。
HI_VENC_RC_MODE_MJPEG_FIXQP	MJPEG FIXQP模式。预留参数，暂不支持。
HI_VENC_RC_MODE_H265_CBR	H.265 CBR模式。

成员名称	描述
HI_VENC_RC_MODE_H265_VBR	H.265 VBR模式。
HI_VENC_RC_MODE_H265_AVBR	H.265 AVBR模式。
HI_VENC_RC_MODE_H265_QVBR	H.265 QVBR模式。
HI_VENC_RC_MODE_H265_CVBR	H.265 CVBR模式。
HI_VENC_RC_MODE_H265_FIXQP	H.265 FIXQP模式。 Atlas 推理系列产品，预留参数，暂不支持。 Atlas A2训练系列产品/Atlas 800I A2推理产品，预留参数，暂不支持。
HI_VENC_RC_MODE_H265_QPMAP	H.265 QPMAP模式。预留参数，暂不支持。

1.9.20.14.8 hi_venc_h264_cbr

说明

定义H.264编码通道CBR属性结构。

定义

```
typedef struct {  
    hi_u32 gop;  
    hi_u32 stats_time;  
    hi_u32 src_frame_rate;  
    hi_u32 dst_frame_rate;  
    hi_u32 bit_rate;  
} hi_venc_h264_cbr;
```

成员

成员名称	描述
gop	Group Of Pictures，视频序列由若干时间连续的图像构成，在对其进行编码时，先将该视频序列分割成若干小的图像组，该参数即指一个图像组的帧数；由于GOP结构仅支持Normalp（第一帧为I帧，其余帧为P帧），此参数也为I帧间隔，默认值65535；由于编码I帧不需参考之前的帧，之前帧的编码质量影响从此帧开始不再传递，故该值越小，编码质量越好；但不建议小于帧率，因为I帧仅有帧内预测块，大部分场景需要有一定的P帧（有帧间预测块）才能保证编码质量。 取值范围：[1, 65536]。

成员名称	描述
stats_time	CBR码率统计时间，以秒为单位，一般场景下，stats_time= gop / dst_frame_rate，编码质量较好。 码率统计时间越长，每帧图像的码率波动对于码率调节的影响越弱，码率的调节会更缓慢，图像质量的波动会更轻微；码率统计时间越短，每帧图像的码率波动对于码率调节的影响越强，图像码率的调节会更灵敏，图像质量的波动会更剧烈。 取值范围：[1, 60]。
src_frame_rate	编码器输入帧率，以fps为单位，即每秒进入编码器的帧数，默认值为30；码率一定的情况下，该值越小，编码质量越好，但小于25播放时画面会不连贯。 该参数取值范围：[1, 240]。
dst_frame_rate	编码器输出帧率，以fps为单位，即每秒编码器输出的帧数，默认值为30；码率一定的情况下，该值越小，编码质量越好，但小于25播放时画面会不连贯。 取值范围：[1, src_frame_rate]。 暂不支持分数帧率。
bit_rate	编码器输出平均码率，以kbps为单位，即编码码流每秒比特数，该值越大，编码质量越好；默认值2000。 该参数取值范围：[2, 614400]。 平均比特率的设置与编码通道宽高以及图像帧率都有关系。典型的平均比特率的设置如表1-85所示。注意，表1-85中的平均比特率的设置是在通道编码帧率为满帧率（30fps）时的设置。当用户设置编码输出帧率不为满帧率时，可以对下表中的码率按用户设置帧率与满帧率（30fps）的比例进行换算。

表 1-85 典型平均比特率配置

图像宽高/码率等级	1080p(1920x1080)	4K(3840x2160)
低码率	< 2Mbps	< 8Mbps
中码率	2Mbps ~ 8Mbps	8Mbps ~ 30Mbps
高码率	>8Mbps	>30Mbps

1.9.20.14.9 hi_venc_h264_vbr

说明

定义H.264编码通道VBR属性结构。

定义

```
typedef struct {  
    hi_u32 gop;
```

```
hi_u32 stats_time;  
hi_u32 src_frame_rate;  
hi_u32 dst_frame_rate;  
hi_u32 max_bit_rate;  
jhi_venc_h264_vbr;
```

成员

成员名称	描述
gop	Group Of Pictures，视频序列由若干时间连续的图像构成，在对 其进行编码时，先将该视频序列分割成若干个小图像组，该参数 即指一个图像组的帧数；由于GOP结构仅支持Normalp（第一帧为I 帧，其余帧为P帧），此参数也为I帧间隔，默认值65535；由于编 码I帧不需参考之前的帧，之前帧的编码质量影响从此帧开始不再传 递，故该值越小，编码质量越好；但不建议小于帧率，因为I帧仅有 帧内预测块，大部分场景需要有一定的P帧（有帧间预测块）才能 保证编码质量。 取值范围：[1, 65536]。
stats_time	VBR码率统计时间，以秒为单位，一般场景下，stats_time= gop / dst_frame_rate，编码质量较好。 码率统计时间越长，每帧图像的码率波动对于码率调节的影响越 弱，码率的调节会更缓慢，图像质量的波动会更轻微；码率统计时 间越短，每帧图像的码率波动对于码率调节的影响越强，图像码率 的调节会更灵敏，图像质量的波动会更剧烈。 取值范围：[1, 60]。
src_frame_rate	编码器输入帧率，以fps为单位，即每秒进入编码器的帧数，默认值 为30；码率一定的情况下，该值越小，编码质量越好，但小于25播 放时画面会不连贯。 该参数取值范围：[1, 240]。
dst_frame_rate	编码器输出帧率，以fps为单位，即每秒编码器输出的帧数，默认值 为30；码率一定的情况下，该值越小，编码质量越好，但小于25播 放时画面会不连贯。 取值范围：[1, src_frame_rate]。 暂不支持分数帧率。
max_bit_rate	编码器输出最大码率，以kbps为单位，即编码码流每秒比特数，该 值越大，编码质量越好；默认值2000。 该参数取值范围：[2, 614400]。

1.9.20.14.10 hi_venc_h264_avbr

说明

定义 H.264 编码通道AVBR属性结构。

定义

```
typedef struct {  
    hi_u32 gop;
```

```
hi_u32 stats_time;  
hi_u32 src_frame_rate;  
hi_u32 dst_frame_rate;  
hi_u32 max_bit_rate;  
jhi_venc_h264_avbr;
```

成员

成员名称	描述
gop	Group Of Pictures，视频序列由若干时间连续的图像构成，在对其进行编码时，先将该视频序列分割成若干个小的图像组，该参数即指一个图像组的帧数；由于GOP结构仅支持Normalp（第一帧为I帧，其余帧为P帧），此参数也为I帧间隔，默认值65535；由于编码I帧不需参考之前的帧，之前帧的编码质量影响从此帧开始不再传递，故该值越小，编码质量越好；但不建议小于帧率，因为I帧仅有帧内预测块，大部分场景需要有一定的P帧（有帧间预测块块）才能保证编码质量。 取值范围：[1, 65536]。
stats_time	AVBR码率统计时间，以秒为单位，一般场景下，stats_time= gop / dst_frame_rate，编码质量较好。 码率统计时间越长，每帧图像的码率波动对于码率调节的影响越弱，码率的调节会更缓慢，图像质量的波动会更轻微；码率统计时间越短，每帧图像的码率波动对于码率调节的影响越强，图像码率的调节会更灵敏，图像质量的波动会更剧烈。 取值范围：[1, 60]。
src_frame_rate	编码器输入帧率，以fps为单位，即每秒进入编码器的帧数，默认值为30；码率一定的情况下，该值越小，编码质量越好，但小于25播放时画面会不连贯。 取值范围：[1, 240]。
dst_frame_rate	编码器输出帧率，以fps为单位，即每秒编码器输出的帧数，默认值为30；码率一定的情况下，该值越小，编码质量越好，但小于25播放时画面会不连贯。 取值范围：[1, src_frame_rate]。 暂不支持分数帧率。
max_bit_rate	编码器输出最大码率，以kbps为单位，即编码码流每秒比特数，该值越大，编码质量越好；默认值2000。 取值范围：[2, 614400]。

1.9.20.14.11 hi_venc_h264_qvbr

说明

定义H.264协议编码通道QVBR模式属性结构。

定义

```
typedef struct {  
    hi_u32 gop;
```

```
hi_u32 stats_time;  
hi_u32 src_frame_rate;  
hi_u32 dst_frame_rate;  
hi_u32 target_bit_rate;  
jhi_venc_h264_qvbr;
```

成员

成员名称	描述
gop	Group Of Pictures，视频序列由若干时间连续的图像构成，在对其进行编码时，先将该视频序列分割成若干个小的图像组，该参数即指一个图像组的帧数；由于GOP结构仅支持Normalp（第一帧为I帧，其余帧为P帧），此参数也为I帧间隔，默认值65535；由于编码I帧不需参考之前的帧，之前帧的编码质量影响从此帧开始不再传递，故该值越小，编码质量越好；但不建议小于帧率，因为I帧仅有帧内预测块，大部分场景需要有一定的P帧（有帧间预测块）才能保证编码质量。 取值范围：[1, 65536]。
stats_time	QVBR码率统计时间，以秒为单位，一般场景下，stats_time=gop / dst_frame_rate，编码质量较好。 码率统计时间越长，每帧图像的码率波动对于码率调节的影响越弱，码率的调节会更缓慢，图像质量的波动会更轻微；码率统计时间越短，每帧图像的码率波动对于码率调节的影响越强，图像码率的调节会更灵敏，图像质量的波动会更剧烈。 取值范围：[1, 60]。
src_frame_rate	编码器输入帧率，以fps为单位，即每秒进入编码器的帧数，默认值为30；码率一定的情况下，该值越小，编码质量越好，但小于25播放时画面会不连贯。 取值范围：[1, 240]。
dst_frame_rate	编码器输出帧率，以fps为单位，即每秒编码器输出的帧数，默认值为30；码率一定的情况下，该值越小，编码质量越好，但小于25播放时画面会不连贯。 取值范围：[1, src_frame_rate]。 暂不支持分数帧率。
target_bit_rate	编码器输出最大码率，以kbps为单位，即编码码流每秒比特数，该值越大，编码质量越好；默认值2000。 取值范围：[2, 614400]

1.9.20.14.12 hi_venc_h264_cvbr

说明

定义H.264协议编码通道CVBR模式属性结构。

定义

```
typedef struct {  
    hi_u32 gop;
```

```
hi_u32 stats_time;  
hi_u32 src_frame_rate;  
hi_u32 dst_frame_rate;  
hi_u32 max_bit_rate;  
hi_u32 short_term_stats_time;  
hi_u32 long_term_stats_time;  
hi_u32 long_term_max_bit_rate;  
hi_u32 long_term_min_bit_rate;  
} hi_venc_h264_cvbr;
```

成员

成员名称	描述
gop	Group Of Pictures，视频序列由若干时间连续的图像构成，在对 其进行编码时，先将该视频序列分割成若干小的图像组，该参数 即指一个图像组的帧数；由于GOP结构仅支持Normalp（第一帧为I 帧，其余帧为P帧），此参数也为I帧间隔，默认值65535；由于编 码I帧不需参考之前的帧，之前帧的编码质量影响从此帧开始不再传 递，故该值越小，编码质量越好；但不建议小于帧率，因为I帧仅有 帧内预测块，大部分场景需要有一定的P帧（有帧间预测块）才 能保证编码质量。 取值范围：[1, 65536]。
stats_time	CVBR码率统计时间，以秒为单位，一般场景下，stats_time= gop / dst_frame_rate，编码质量较好。 码率统计时间越长，每帧图像的码率波动对于码率调节的影响越 弱，码率的调节会更缓慢，图像质量的波动会更轻微；码率统计时 间越短，每帧图像的码率波动对于码率调节的影响越强，图像码率 的调节会更灵敏，图像质量的波动会更剧烈。 取值范围：[1, 60]。
src_frame_rate	编码器输入帧率，以fps为单位，即每秒进入编码器的帧数，默认值 为30；码率一定的情况下，该值越小，编码质量越好，但小于25播 放时画面会不连贯。 取值范围：[1, 240]。
dst_frame_rate	编码器输出帧率，以fps为单位，即每秒编码器输出的帧数，默认值 为30；码率一定的情况下，该值越小，编码质量越好，但小于25播 放时画面会不连贯。 取值范围：[1, src_frame_rate]。 暂不支持分数帧率。
max_bit_rate	编码器输出最大码率，以kbps为单位，即编码码流每秒比特数，该 值越大，编码质量越好；默认值2000。 取值范围：[2, 614400]。
short_term_stats_time	码率短期统计时间，以秒为单位。 取值范围：[1, 120]。
long_term_stats_time	码率长期统计时间，默认为分钟。 取值范围：[1, 1440]。

成员名称	描述
long_term_max_bit_rate	编码器输出长期最大码率，以kbps为单位。 取值范围：[2, max_bit_rate]。
long_term_min_bit_rate	编码器输出长期最小码率，以kbps为单位。 取值范围：[0, long_term_max_bit_rate] 。

1.9.20.14.13 hi_venc_h264_fixqp

说明

定义 H.264 编码通道FIXQP属性结构。暂不支持。

定义

```
typedef struct {  
    hi_u32 gop;  
    hi_u32 src_frame_rate;  
    hi_u32 dst_frame_rate;  
    hi_u32 i_qp;  
    hi_u32 p_qp;  
    hi_u32 b_qp;  
} hi_venc_h264_fixqp;
```

成员

成员名称	描述
gop	Group Of Pictures，视频序列由若干时间连续的图像构成，在对其进行编码时，先将该视频序列分割成若干小的图像组，该参数即指一个图像组的帧数；由于GOP结构仅支持Normalp（第一帧为I帧，其余帧为P帧），此参数也为I帧间隔，默认值65535；由于编码I帧不需参考之前的帧，之前帧的编码质量影响从此帧开始不再传递，故该值越小，编码质量越好；但不建议小于帧率，因为I帧仅有帧内预测块，大部分场景需要有一定的P帧（有帧间预测块块）才能保证编码质量。 取值范围：[1, 65536]。
src_frame_rate	编码器输入帧率，以fps为单位，即每秒进入编码器的帧数，默认值为30；码率一定的情况下，该值越小，编码质量越好，但小于25播放时画面会不连贯。 该参数取值范围：[1, 240]。
dst_frame_rate	编码器输出帧率，以fps为单位，即每秒编码器输出的帧数，默认值为30；码率一定的情况下，该值越小，编码质量越好，但小于25播放时画面会不连贯。 取值范围：[1, src_frame_rate]。 暂不支持分数帧率。

成员名称	描述
i_qp	I帧所有宏块Qp值。 取值范围：[0, 51]。
p_qp	P帧所有宏块Qp值。 取值范围：[0, 51]。
b_qp	B帧所有宏块Qp值，当前仅支持配置为0。

1.9.20.14.14 hi_venc_h264_qpmap

说明

定义 H.264 编码通道QPMAP属性结构。暂不支持。

定义

```
typedef struct {  
    hi_u32 gop;  
    hi_u32 stats_time;  
    hi_u32 src_frame_rate;  
    hi_u32 dst_frame_rate;  
}hi_venc_h264_qpmap;
```

1.9.20.14.15 hi_venc_mjpeg_cbr

说明

定义MJPEG编码通道CBR属性结构。暂不支持。

定义

```
typedef struct {  
    hi_u32 stats_time; // RW; Range:[1, 60]; the rate statistic time, the unit is senconds(s)  
    hi_u32 src_frame_rate; // RW; Range:[1, 240]; the input frame rate of the venc channel  
    hi_u32 dst_frame_rate; // RW; Range:[1, 240]; the target frame rate of the venc channel,  
    // can not be larger than src_frame_rate  
    hi_u32 bit_rate; // RW; Range:[2, 614400]; average bitrate  
} hi_venc_mjpeg_cbr;
```

1.9.20.14.16 hi_venc_mjpeg_vbr

说明

定义MJPEG编码通道VBR属性结构。暂不支持。

定义

```
typedef struct {  
    hi_u32 stats_time; // RW; Range:[1, 60]; the rate statistic time, the unit is senconds(s)  
    hi_u32 src_frame_rate; // RW; Range:[1, 240]; the input frame rate of the venc channel  
    hi_u32 dst_frame_rate; // RW; Range:[1, 240]; the target frame rate of the venc channel,  
    // can not be larger than src_frame_rate  
    hi_u32 max_bit_rate; // RW; Range:[2, 614400];the max bitrate  
}hi_venc_mjpeg_vbr;
```


1.9.20.14.17 hi_venc_mjpeg_fixqp

说明

定义MJPEG编码通道FIXQP属性结构。暂不支持。

定义

```
typedef struct {  
    hi_u32 src_frame_rate; // RW; Range:[1, 240]; the input frame rate of the venc channel  
    hi_u32 dst_frame_rate; // RW; Range:[1, 240]; the target frame rate of the venc channel,  
        // can not be larger than src_frame_rate  
    hi_u32 qfactor; // RW; Range:[1,99];image quality.  
} hi_venc_mjpeg_fixqp;
```

1.9.20.14.18 hi_venc_h265_cbr

说明

定义 H.265编码通道CBR属性结构。

定义

```
typedef struct {  
    hi_u32 gop;  
    hi_u32 stats_time;  
    hi_u32 src_frame_rate;  
    hi_u32 dst_frame_rate;  
    hi_u32 bit_rate;  
} hi_venc_h265_cbr;
```

成员

成员名称	描述
gop	Group Of Pictures，视频序列由若干时间连续的图像构成，在对 其进行编码时，先将该视频序列分割成若干个小图像组，该参数 即指一个图像组的帧数；由于GOP结构仅支持Normalp（第一帧为I 帧，其余帧为P帧），此参数也为I帧间隔，默认值65535；由于编 码I帧不需参考之前的帧，之前帧的编码质量影响从此帧开始不再传 递，故该值越小，编码质量越好；但不建议小于帧率，因为I帧仅有 帧内预测块，大部分场景需要有一定的P帧（有帧间预测块块）才 能保证编码质量。 取值范围：[1, 65536]。
stats_time	CBR码率统计时间，以秒为单位，一般场景下，stats_time= gop / dst_frame_rate，编码质量较好。 码率统计时间越长，每帧图像的码率波动对于码率调节的影响越 弱，码率的调节会更缓慢，图像质量的波动会更轻微；码率统计时 间越短，每帧图像的码率波动对于码率调节的影响越强，图像码率 的调节会更灵敏，图像质量的波动会更剧烈。 取值范围：[1, 60]。

成员名称	描述
src_frame_rate	编码器输入帧率，以fps为单位，即每秒进入编码器的帧数，默认值为30；码率一定的情况下，该值越小，编码质量越好，但小于25播放时画面会不连贯。 该参数取值范围：[1, 240]。
dst_frame_rate	编码器输出帧率，以fps为单位，即每秒编码器输出的帧数，默认值为30；码率一定的情况下，该值越小，编码质量越好，但小于25播放时画面会不连贯。 取值范围：[1, src_frame_rate]。 暂不支持分数帧率。
bit_rate	编码器输出平均码率，以kbps为单位，即编码码流每秒比特数，该值越大，编码质量越好；默认值2000。 该参数取值范围：[2, 614400]。 平均比特率的设置与编码通道宽高以及图像帧率都有关系。典型的平均比特率的设置如表1-86所示。注意，表1-86中的平均比特率的设置是在通道编码帧率为满帧率（30fps）时的设置。当用户设置编码输出帧率不为满帧率时，可以对下表中的码率按用户设置帧率与满帧率（30fps）的比例进行换算。

表 1-86 典型平均比特率配置

图像宽高/码率等级	1080p(1920x1080)	4K(3840x2160)
低码率	< 2Mbps	< 8Mbps
中码率	2Mbps ~ 8Mbps	8Mbps ~ 30Mbps
高码率	>8Mbps	>30Mbps

1.9.20.14.19 hi_venc_h265_vbr

说明

定义 H.265编码通道VBR属性结构。

定义

```
typedef struct {  
    hi_u32 gop;  
    hi_u32 stats_time;  
    hi_u32 src_frame_rate;  
    hi_u32 dst_frame_rate;  
    hi_u32 max_bit_rate;  
}hi_venc_h265_vbr;
```

成员

成员名称	描述
gop	Group Of Pictures，视频序列由若干时间连续的图像构成，在对其进行编码时，先将该视频序列分割成若干小的图像组，该参数即指一个图像组的帧数；由于GOP结构仅支持Normalp（第一帧为I帧，其余帧为P帧），此参数也为I帧间隔，默认值65535；由于编码I帧不需参考之前的帧，之前帧的编码质量影响从此帧开始不再传递，故该值越小，编码质量越好；但不建议小于帧率，因为I帧仅有帧内预测块，大部分场景需要有一定的P帧（有帧间预测块块）才能保证编码质量。 取值范围：[1, 65536]。
stats_time	VBR 码率统计时间，以秒为单位，一般场景下，stats_time= gop / dst_frame_rate，编码质量较好。 码率统计时间越长，每帧图像的码率波动对于码率调节的影响越弱，码率的调节会更缓慢，图像质量的波动会更轻微；码率统计时间越短，每帧图像的码率波动对于码率调节的影响越强，图像码率的调节会更灵敏，图像质量的波动会更剧烈。 取值范围：[1, 60]。
src_frame_rate	编码器输入帧率，以fps为单位，即每秒进入编码器的帧数，默认值为30；码率一定的情况下，该值越小，编码质量越好，但小于25播放时画面会不连贯。 该参数取值范围：[1, 240]。
dst_frame_rate	编码器输出帧率，以fps为单位，即每秒编码器输出的帧数，默认值为30；码率一定的情况下，该值越小，编码质量越好，但小于25播放时画面会不连贯。 取值范围：[1, src_frame_rate]。 暂不支持分数帧率。
max_bit_rate	编码器输出最大码率，以kbps为单位，即编码码流每秒比特数，该值越大，编码质量越好；默认值2000。 该参数取值范围：[2, 614400]。

注意事项

请参见hi_venc_h264_cbr中关于src_frame_rate和dst_frame_rate的说明。

1.9.20.14.20 hi_venc_h265_avbr

说明

定义H.265编码通道AVBR属性结构。

定义

```
typedef struct {  
    hi_u32 gop;
```

```
hi_u32 stats_time;  
hi_u32 src_frame_rate;  
hi_u32 dst_frame_rate;  
hi_u32 max_bit_rate;  
hi_venc_h265_avbr;
```

成员

成员名称	描述
gop	Group Of Pictures，视频序列由若干时间连续的图像构成，在对其进行编码时，先将该视频序列分割成若干小的图像组，该参数即指一个图像组的帧数；由于GOP结构仅支持Normalp（第一帧为I帧，其余帧为P帧），此参数也为I帧间隔，默认值65535；由于编码I帧不需参考之前的帧，之前帧的编码质量影响从此帧开始不再传递，故该值越小，编码质量越好；但不建议小于帧率，因为I帧仅有帧内预测块，大部分场景需要有一定的P帧（有帧间预测块块）才能保证编码质量。 取值范围：[1, 65536]。
stats_time	AVBR码率统计时间，以秒为单位，一般场景下，stats_time= gop / dst_frame_rate，编码质量较好。 码率统计时间越长，每帧图像的码率波动对于码率调节的影响越弱，码率的调节会更缓慢，图像质量的波动会更轻微；码率统计时间越短，每帧图像的码率波动对于码率调节的影响越强，图像码率的调节会更灵敏，图像质量的波动会更剧烈。 取值范围：[1, 60]。
src_frame_rate	编码器输入帧率，以fps为单位，即每秒进入编码器的帧数，默认值为30；码率一定的情况下，该值越小，编码质量越好，但小于25播放时画面会不连贯。 该参数取值范围：[1, 240]。
dst_frame_rate	编码器输出帧率，以fps为单位，即每秒编码器输出的帧数，默认值为30；码率一定的情况下，该值越小，编码质量越好，但小于25播放时画面会不连贯。 取值范围：[1, src_frame_rate]。 暂不支持分数帧率。
max_bit_rate	编码器输出最大码率，以kbps为单位，即编码码流每秒比特数，该值越大，编码质量越好；默认值2000。 该参数取值范围：[2, 614400]。

1.9.20.14.21 hi_venc_h265_qvbr

说明

定义H.265协议编码通道QVBR模式属性结构。

定义

```
typedef struct {  
    hi_u32 gop;
```

```
hi_u32 stats_time;  
hi_u32 src_frame_rate;  
hi_u32 dst_frame_rate;  
hi_u32 target_bit_rate;  
}hi_venc_h265_qvbr;
```

成员

成员名称	描述
gop	Group Of Pictures，视频序列由若干时间连续的图像构成，在对其进行编码时，先将该视频序列分割成若干小的图像组，该参数即指一个图像组的帧数；由于GOP结构仅支持Normalp（第一帧为I帧，其余帧为P帧），此参数也为I帧间隔，默认值65535；由于编码I帧不需参考之前的帧，之前帧的编码质量影响从此帧开始不再传递，故该值越小，编码质量越好；但不建议小于帧率，因为I帧仅有帧内预测块，大部分场景需要有一定的P帧（有帧间预测块块）才能保证编码质量。 取值范围：[1, 65536]。
stats_time	QVBR码率统计时间，以秒为单位，一般场景下，stats_time=gop / dst_frame_rate，编码质量较好。 码率统计时间越长，每帧图像的码率波动对于码率调节的影响越弱，码率的调节会更缓慢，图像质量的波动会更轻微；码率统计时间越短，每帧图像的码率波动对于码率调节的影响越强，图像码率的调节会更灵敏，图像质量的波动会更剧烈。 取值范围：[1, 60]。
src_frame_rate	编码器输入帧率，以fps为单位，即每秒进入编码器的帧数，默认值为30；码率一定的情况下，该值越小，编码质量越好，但小于25播放时画面会不连贯。 取值范围：[1, 240]。
dst_frame_rate	编码器输出帧率，以fps为单位，即每秒编码器输出的帧数，默认值为30；码率一定的情况下，该值越小，编码质量越好，但小于25播放时画面会不连贯。 取值范围：[1, src_frame_rate]。 暂不支持分数帧率。
target_bit_rate	编码器输出最大码率，以kbps为单位，即编码码流每秒比特数，该值越大，编码质量越好；默认值2000。 取值范围：[2, 614400]。

1.9.20.14.22 hi_venc_h265_cvbr

说明

定义H.265协议编码通道CVBR模式属性结构。

定义

```
typedef struct {  
    hi_u32 gop;
```

```
hi_u32 stats_time;  
hi_u32 src_frame_rate;  
hi_u32 dst_frame_rate;  
hi_u32 max_bit_rate;  
hi_u32 short_term_stats_time;  
hi_u32 long_term_stats_time;  
hi_u32 long_term_max_bit_rate;  
hi_u32 long_term_min_bit_rate;  
} hi_venc_h265_cvbr;
```

成员

成员名称	描述
gop	Group Of Pictures，视频序列由若干时间连续的图像构成，在对其进行编码时，先将该视频序列分割成若干小的图像组，该参数即指一个图像组的帧数；由于GOP结构仅支持Normalp（第一帧为I帧，其余帧为P帧），此参数也为I帧间隔，默认值65535；由于编码I帧不需参考之前的帧，之前帧的编码质量影响从此帧开始不再传递，故该值越小，编码质量越好；但不建议小于帧率，因为I帧仅有帧内预测块，大部分场景需要有一定的P帧（有帧间预测块块）才能保证编码质量。 取值范围：[1, 65536]。
stats_time	CVBR码率统计时间，以秒为单位，一般场景下，stats_time=gop / dst_frame_rate，编码质量较好。 码率统计时间越长，每帧图像的码率波动对于码率调节的影响越弱，码率的调节会更缓慢，图像质量的波动会更轻微；码率统计时间越短，每帧图像的码率波动对于码率调节的影响越强，图像码率的调节会更灵敏，图像质量的波动会更剧烈。 取值范围：[1, 60]。
src_frame_rate	编码器输入帧率，以fps为单位，即每秒进入编码器的帧数，默认值为30；码率一定的情况下，该值越小，编码质量越好，但小于25播放时画面会不连贯。 取值范围：[1, 240]。
dst_frame_rate	编码器输出帧率，以fps为单位，即每秒编码器输出的帧数，默认值为30；码率一定的情况下，该值越小，编码质量越好，但小于25播放时画面会不连贯。 取值范围：[1, src_frame_rate]。 暂不支持分数帧率。
max_bit_rate	编码器输出最大码率，以kbps为单位，即编码码流每秒比特数，该值越大，编码质量越好；默认值2000。 取值范围：[2, 614400]。
short_term_stats_time	码率短期统计时间，以秒为单位。 取值范围：[1, 120]。
long_term_stats_time	码率长期统计时间，默认为分钟。 取值范围：[1, 1440]。

成员名称	描述
long_term_max_bit_rate	编码器输出长期最大码率，以kbps为单位。 取值范围：[2, max_bit_rate]。
long_term_min_bit_rate	编码器输出长期最小码率，以kbps为单位。 取值范围：[0, long_term_max_bit_rate] 。

1.9.20.14.23 hi_venc_h265_fixqp

说明

定义H.265编码通道FIXQP属性结构。

Atlas 推理系列产品暂不支持该结构体。

定义

```
typedef struct {  
    hi_u32 gop;  
    hi_u32 src_frame_rate;  
    hi_u32 dst_frame_rate;  
    hi_u32 i_qp;  
    hi_u32 p_qp;  
    hi_u32 b_qp;  
} hi_venc_h265_fixqp;
```

成员

成员名称	描述
gop	Group Of Pictures，视频序列由若干时间连续的图像构成，在对其进行编码时，先将该视频序列分割成若干小的图像组，该参数即指一个图像组的帧数；由于GOP结构仅支持Normalp（第一帧为I帧，其余帧为P帧），此参数也为I帧间隔，默认值65535；由于编码I帧不需参考之前的帧，之前帧的编码质量影响从此帧开始不再传递，故该值越小，编码质量越好；但不建议小于帧率，因为I帧仅有帧内预测块，大部分场景需要有一定的P帧（有帧间预测块块）才能保证编码质量。 取值范围：[1, 65536]。
src_frame_rate	编码器输入帧率，以fps为单位，即每秒进入编码器的帧数，默认值为30；码率一定的情况下，该值越小，编码质量越好，但小于25播放时画面会不连贯。 该参数取值范围：[1, 240]。
dst_frame_rate	编码器输出帧率，以fps为单位，即每秒编码器输出的帧数，默认值为30；码率一定的情况下，该值越小，编码质量越好，但小于25播放时画面会不连贯。 取值范围：[1, src_frame_rate]。 暂不支持分数帧率。

成员名称	描述
i_qp	I帧所有宏块Qp值。 取值范围：[0, 51]。
p_qp	P帧所有宏块Qp值。 取值范围：[0, 51]。
b_qp	B帧所有宏块Qp值，当前仅支持配置为0。

1.9.20.14.24 hi_venc_h265_qpmap

说明

定义H.265编码通道QP MAP属性结构。暂不支持。

定义

```
typedef struct {  
    hi_u32 gop;  
    hi_u32 stats_time;  
    hi_u32 src_frame_rate;  
    hi_u32 dst_frame_rate;  
    hi_venc_rc_qpmap_mode qpmap_mode;  
}hi_venc_h265_qpmap;
```

1.9.20.14.25 hi_venc_gop_attr

说明

定义编码器GOP属性结构体。

定义

```
typedef struct {  
    hi_venc_gop_mode gop_mode;  
    union {  
        hi_venc_gop_normal_p normal_p;  
        hi_venc_gop_dual_p dual_p;  
        hi_venc_gop_smart_p smart_p;  
        hi_venc_gop_adv_smart_p adv_smart_p;  
        hi_venc_gop_bipred_b bipred_b;  
    };  
} hi_venc_gop_attr;
```

成员

成员名称	描述
gop_mode	编码GOP类型。
normal_p	编码单参考帧P帧GOP属性结构体。

成员名称	描述
dual_p	编码双参考帧P帧GOP属性结构体。 Atlas 推理系列产品，预留参数，暂不支持。 Atlas A2训练系列产品/Atlas 800I A2推理产品，预留参数，暂不支持。
smart_p	编码智能P帧GOP属性结构体。 Atlas 推理系列产品，预留参数，暂不支持。 Atlas A2训练系列产品/Atlas 800I A2推理产品，预留参数，暂不支持。
adv_smart_p	编码高级智能P帧GOP属性结构体，暂不支持。
bipred_b	编码B帧GOP属性结构体，暂不支持。

1.9.20.14.26 hi_venc_gop_mode

说明

定义 H.264/H.265 GOP类型。

定义

```
typedef enum {  
    HI_VENC_GOP_MODE_NORMAL_P    = 0,  
    HI_VENC_GOP_MODE_DUAL_P      = 1,  
    HI_VENC_GOP_MODE_SMART_P     = 2,  
    HI_VENC_GOP_MODE_ADV_SMART_P = 3,  
    HI_VENC_GOP_MODE_BIPRED_B    = 4,  
    HI_VENC_GOP_MODE_LOW_DELAY_B = 5,  
  
    HI_VENC_GOP_MODE_BUTT,  
} hi_venc_gop_mode;
```

成员

成员名称	描述
HI_VENC_GOP_MODE_NORMAL_P	编码单参考帧P帧GOP类型。
HI_VENC_GOP_MODE_DUAL_P	编码双参考帧P帧GOP类型。 Atlas 推理系列产品，预留参数，暂不支持。 Atlas A2训练系列产品/Atlas 800I A2推理产品，预留参数，暂不支持。
HI_VENC_GOP_MODE_SMART_P	编码智能P帧GOP类型。 Atlas 推理系列产品，预留参数，暂不支持。 Atlas A2训练系列产品/Atlas 800I A2推理产品，预留参数，暂不支持。

成员名称	描述
HI_VENC_GOP_MODE_ADV_SMART_P	编码高级智能P帧GOP类型，暂不支持。
HI_VENC_GOP_MODE_BIPRED_B	编码B帧GOP类型，暂不支持。
HI_VENC_GOP_MODE_LOW_DELAY_B	编码B帧GOP类型，其中B帧只有前向参考帧，暂不支持。
HI_VENC_GOP_MODE_BUTT	保留值

1.9.20.14.27 hi_venc_gop_normal_p

说明

定义编码单参考帧P帧GOP属性结构体。

定义

```
typedef struct {  
    hi_s32 ip_qp_delta;  
}hi_venc_gop_normal_p;
```

成员

成员名称	描述
ip_qp_delta	I帧相对P帧的QP（Quantisation Parameter）差值。 取值范围：[-10, 30]。

1.9.20.14.28 hi_venc_gop_dual_p

说明

定义编码双参考帧P帧GOP属性结构体。

Atlas 推理系列产品，预留结构体，暂不支持。

Atlas A2训练系列产品/Atlas 800I A2推理产品，预留结构体，暂不支持。

定义

```
typedef struct {  
    hi_u32 sp_interval;  
    hi_s32 sp_qp_delta;  
    hi_s32 ip_qp_delta;  
}hi_venc_gop_dual_p;
```

成员

成员名称	描述
sp_interval	Special P帧的间隔。 取值范围：[0, 1) U (1, gop-1]，gop是I帧间隔。
sp_qp_delta	Special P帧相对普通P帧的QP差值。 取值范围：[-10, 30]。
ip_qp_delta	I帧相对普通P帧的QP差值。 取值范围：[-10, 30]。

1.9.20.14.29 hi_venc_gop_smart_p

说明

定义编码智能P帧GOP属性结构体。

Atlas 推理系列产品，预留结构体，暂不支持。

Atlas A2训练系列产品/Atlas 800I A2推理产品，预留结构体，暂不支持。

定义

```
typedef struct {  
    hi_u32 bg_interval;  
    hi_s32 bg_qp_delta;  
    hi_s32 vi_qp_delta;  
}hi_venc_gop_smart_p;
```

成员

成员名称	描述
bg_interval	长期参考帧的间隔。 取值范围：[gop, 65536]，且必须是gop的整数倍，gop是I帧间隔。
bg_qp_delta	长期参考帧相对P帧的QP差值。 取值范围：[-10, 30]。 在调试信息中，该参数等同其它码控模式的ip_qp_delta。
vi_qp_delta	虚拟I帧相对普通P帧的QP差值。 取值范围：[-10, 30]。

1.9.20.14.30 hi_venc_gop_adv_smart_p

说明

定义编码高级智能P帧GOP属性结构体，暂不支持。

定义

```
typedef struct {  
    hi_u32 bg_interval;  
    hi_s32 bg_qp_delta;  
    hi_s32 vi_qp_delta;  
}hi_venc_gop_adv_smart_p;
```

1.9.20.14.31 hi_venc_gop_bipred_b

说明

定义编码B帧GOP属性结构体，暂不支持。

定义

```
typedef struct {  
    hi_u32 b_frame_num; // RW; Range:[1, 3]; Number of B frames  
    hi_s32 b_qp_delta; // RW; Range:[-10, 30]; QP variance between P frame and B frame  
    hi_s32 ip_qp_delta; // RW; Range:[-10, 30]; QP variance between P frame and I frame  
} hi_venc_gop_bipred_b;
```

1.9.20.14.32 hi_venc_start_param

说明

定义编码通道连续接收并编码的帧数结构体。

定义

```
typedef struct {  
    hi_s32 recv_pic_num;  
} hi_venc_start_param;
```

成员

成员名称	描述
recv_pic_num	编码通道连续接收并编码的帧数。 取值范围：-1或正整数。

1.9.20.14.33 hi_venc_chn_status

说明

定义编码通道的状态结构体。

定义

```
typedef struct {  
    hi_u32 left_pics;  
    hi_u32 left_stream_bytes;  
    hi_u32 left_stream_frames;  
    hi_u32 cur_packs;  
    hi_u32 left_recv_pics;  
    hi_u32 left_enc_pics;  
    hi_bool is_jpeg_snap_end;
```

```
hi_u64 release_pic_pts;  
hi_venc_stream_info stream_info;  
} hi_venc_chn_status;
```

成员

成员名称	描述
left_pics	待编码的图像数，暂不支持。
left_stream_bytes	码流buffer中用户暂未取走的byte数目，暂不支持。
left_stream_frames	码流buffer中用户暂未取走的帧数目，暂不支持。
cur_packs	当前帧的码流包个数。
left_recv_pics	剩余待接收的帧数，在用户调用接口hi_mpi_venc_start_chn设置接收帧数后有效，暂不支持。
left_enc_pics	剩余待编码的帧数，在用户调用接口hi_mpi_venc_start_chn设置接收帧数后有效，暂不支持。
is_jpeg_snap_end	Jpege抓拍模式下指示抓拍过程是否结束，暂不支持。
release_pic_pts	预留参数，暂不支持。
stream_info	码流信息。 Atlas 推理系列产品，预留参数，暂不支持。 Atlas A2训练系列产品/Atlas 800I A2推理产品，预留参数，暂不支持。

注意事项

如果venc接收的抓拍帧有丢帧出现，is_jpeg_snap_end会存在不能正确标记抓拍结束的情况。

1.9.20.14.34 hi_venc_h264_stream_info

说明

定义H.264协议码流特征信息。

Atlas 推理系列产品，预留结构体，暂不支持。

Atlas A2训练系列产品/Atlas 800I A2推理产品，预留结构体，暂不支持。

定义

```
typedef struct {  
    hi_u32 pic_bytes;  
    hi_u32 inter16x16_mb_num;  
    hi_u32 inter8x8_mb_num;  
    hi_u32 intra16_mb_num;  
    hi_u32 intra8_mb_num;  
    hi_u32 intra4_mb_num;  
    hi_venc_ref_type ref_type;  
    hi_u32 update_attr_cnt;
```

```
hi_u32 start_qp;  
hi_u32 mean_qp;  
hi_bool is_p_skip;  
} hi_venc_h264_stream_info;
```

成员

成员名称	描述
pic_bytes	编码当前帧的字节数。
inter16x16_mb_num	编码当前帧中采用Inter16*16预测模式的宏块数。 Atlas 200/500 A2推理产品，暂不支持该参数。
inter8x8_mb_num	编码当前帧中采用Inter8*8预测模式的宏块数。 Atlas 200/500 A2推理产品，暂不支持该参数。
intra16_mb_num	编码当前帧中采用Intra16预测模式的宏块数。 Atlas 200/500 A2推理产品，暂不支持该参数。
intra8_mb_num	编码当前帧中采用Intra8预测模式的宏块数。 Atlas 200/500 A2推理产品，暂不支持该参数。
intra4_mb_num	编码当前帧中采用Intra4预测模式的宏块数。 Atlas 200/500 A2推理产品，暂不支持该参数。
ref_type	高级跳帧参考下的编码帧类型。 Atlas 200/500 A2推理产品，暂不支持该参数。
update_attr_cnt	通道属性或参数(包含RC参数)被设置的次数。 Atlas 200/500 A2推理产品，暂不支持该参数。
start_qp	编码当前帧的startqp值。
mean_qp	编码当前帧的平均QP帧。
is_p_skip	标识当前帧是否为pskip帧。 Atlas 200/500 A2推理产品，暂不支持该参数。

1.9.20.14.35 hi_venc_ref_type

说明

定义H.264跳帧参考码流的帧类型和参考属性，暂不支持。

定义

```
typedef enum {  
    HI_VENC_BASE_IDR_SLICE = 0,  
    HI_VENC_BASE_P_SLICE_REF_TO_IDR,  
    HI_VENC_BASE_P_SLICE_REF_BY_BASE,  
}
```

```
HI_VENC_BASE_P_SLICE_REF_BY_ENHANCE,  
HI_VENC_ENHANCE_P_SLICE_REF_BY_ENHANCE,  
HI_VENC_ENHANCE_P_SLICE_NOT_FOR_REF,  
HI_VENC_P_SLICE_BUTT  
} hi_venc_ref_type;  
}
```

1.9.20.14.36 hi_venc_jpeg_stream_info

说明

定义JPEG协议码流特征信息， 暂不支持。

定义

```
typedef struct {  
    hi_u32 pic_bytes;  
    hi_u32 update_attr_cnt;  
    hi_u32 qfactor;  
}hi_venc_jpeg_stream_info;
```

成员

成员名称	描述
pic_bytes	一帧JPEG码流大小，以字节为单位。
update_attr_cnt	通道属性或参数(包含RC参数)被设置的次数。
qfactor	编码当前帧的Qfactor。

1.9.20.14.37 hi_venc_h265_stream_info

说明

定义H.265协议码流特征信息。

Atlas 推理系列产品，预留结构体，暂不支持。

Atlas A2训练系列产品/Atlas 800I A2推理产品，预留结构体，暂不支持。

定义

```
typedef struct {  
    hi_u32 pic_bytes;  
    hi_u32 inter64x64_cu_num;  
    hi_u32 inter32x32_cu_num;  
    hi_u32 inter16x16_cu_num;  
    hi_u32 inter8x8_cu_num;  
    hi_u32 intra32x32_cu_num;  
    hi_u32 intra16x16_cu_num;  
    hi_u32 intra8x8_cu_num;  
    hi_u32 intra4x4_cu_num;  
    hi_venc_ref_type ref_type;  
    hi_u32 update_attr_cnt;  
    hi_u32 start_qp;  
    hi_u32 mean_qp;  
    hi_bool is_p_skip;  
}hi_venc_h265_stream_info;
```

成员

成员名称	描述
pic_bytes	编码当前帧的字节数。
inter64x64_cu_num	编码当前帧中采用Inter64*64预测模式的CU块数。 Atlas 200/500 A2推理产品，暂不支持该参数。
inter32x32_cu_num	编码当前帧中采用Inter32*32预测模式的CU块数。 Atlas 200/500 A2推理产品，暂不支持该参数。
inter16x16_cu_num	编码当前帧中采用Inter16*16预测模式的CU块数。 Atlas 200/500 A2推理产品，暂不支持该参数。
inter8x8_cu_num	编码当前帧中采用Inter8*8预测模式的CU块数。 Atlas 200/500 A2推理产品，暂不支持该参数。
intra32x32_cu_num	编码当前帧中采用Intra32*32预测模式的CU块数。 Atlas 200/500 A2推理产品，暂不支持该参数。
intra16x16_cu_num	编码当前帧中采用Intra16*16预测模式的CU块数。 Atlas 200/500 A2推理产品，暂不支持该参数。
intra8x8_cu_num	编码当前帧中采用Intra8*8预测模式的CU块数。 Atlas 200/500 A2推理产品，暂不支持该参数。
intra4x4_cu_num	编码当前帧中采用Intra4*4预测模式的CU块数。 Atlas 200/500 A2推理产品，暂不支持该参数。
ref_type	高级跳帧参考下的编码帧类型。 Atlas 200/500 A2推理产品，暂不支持该参数。
update_attr_cnt	通道属性或参数(包含RC参数)被设置的次数。 Atlas 200/500 A2推理产品，暂不支持该参数。
start_qp	编码当前帧的startqp值。
mean_qp	编码当前帧的平均QP值。
is_p_skip	标识当前帧是否为pskip帧。 Atlas 200/500 A2推理产品，暂不支持该参数。

注意事项

ref_type请参见hi_venc_h264_stream_info中关于ref_type变量的说明。

1.9.20.14.38 hi_venc_prores_stream_info

说明

定义PRORES协议码流特征信息， 暂不支持。

定义

```
typedef struct {  
    hi_u32 pic_bytes;  
    hi_u32 update_attr_cnt;  
} hi_venc_prores_stream_info;
```

成员

成员名称	描述
pic_bytes	编码当前帧的字节数。
update_attr_cnt	通道属性或参数被设置的次数。

1.9.20.14.39 hi_venc_mod_param

说明

编码相关模块参数。

定义

```
typedef struct {  
    hi_venc_mod_type mod_type;  
    union {  
        hi_venc_venc_mod_param venc_mod_param;  
        hi_venc_h264_mod_param h264_mod_param;  
        hi_venc_h265_mod_param h265_mod_param;  
        hi_venc_jpeg_mod_param jpeg_mod_param;  
    };  
} hi_venc_mod_param;
```

成员

成员名称	描述
mod_type	设置或者获取模块参数的类型。
venc_mod_param/ h264_mod_param/ h265_mod_param/ jpeg_mod_param	VENCL内部各个模块的工作参数。

1.9.20.14.40 hi_venc_mod_type

说明

编码相关模块参数类型。

定义

```
typedef enum {  
    HI_VENC_MOD_VENC = 1,  
    HI_VENC_MOD_H264,  
    HI_VENC_MOD_H265,  
    HI_VENC_MOD_JPEG,  
    HI_VENC_MOD_RC,  
    HI_VENC_MOD_BUTT  
} hi_venc_mod_type;
```

成员

成员名称	描述
HI_VENC_MOD_VENC	VENC内公共模块参数类型。
HI_VENC_MOD_H264	VENC内h264e模块参数类型。
HI_VENC_MOD_H265	VENC内h265e模块参数类型。
HI_VENC_MOD_JPEG	VENC内jpege模块参数类型。
HI_VENC_MOD_RC	VENC内码控模块参数类型。暂不支持设置。
HI_VENC_MOD_BUTT	保留值。

1.9.20.14.41 hi_venc_venc_mod_param

说明

Device上的venc内公共模块参数。

定义

```
typedef struct {  
    hi_u32 buf_cache;  
    hi_u32 frame_buf_recycle;  
} hi_venc_venc_mod_param;
```

成员

成员名称	描述
buf_cache	码流获取是否支持cache方式，暂不支持，默认值为0。 0：关闭码流Buffer Cache。 1：打开码流Buffer Cache。

成员名称	描述
frame_buf_recycle	帧存是否回收，暂不支持，默认值为0。 0：关闭编码帧存回收。 1：打开编码帧存回收。

1.9.20.14.42 hi_venc_h264_mod_param

说明

Device上的venc内h264e模块参数。

定义

```
typedef struct {  
    hi_u32    one_stream_buf;  
    hi_u32    mini_buf_mode;  
    hi_u32    low_power_mode;  
    hi_vb_src vb_src;  
    hi_bool   qp_hist_en;  
    hi_u32    max_user_data_len;  
} hi_venc_h264_mod_param;
```

成员

成员名称	描述
one_stream_buf	编码码流帧配置模式，默认值为1。 0：多包模式。 1：单包模式。
mini_buf_mode	编码码流buffer配置模式，暂不支持, 默认值为1。 0：码流buffer根据分辨率分配。 1：码流buffer下限为32k，用户保证合理。
low_power_mode	低功耗模式，暂不支持，默认值为0。 0：关闭低功耗模式。 1：使能低功耗模式。 注意：低功耗或者极低功耗使能后会导致图像质量损失。
vb_src	参考帧和重构帧的帧存分配方式，默认值为HI_VB_SRC_PRIVATE。 - HI_VB_SRC_PRIVATE：私有VB方式，驱动内部完成帧存分配。 - HI_VB_SRC_USER：用户VB，需由用户分配帧存，暂不支持。

成员名称	描述
qp_hist_en	Qp 直方图输出控制模式，暂不支持，默认值为1。 0: Qp 直方图不输出，即 hi_venc_h264_adv_stream_info中的成员变量 qp_hist 为全0。 1: Qp 直方图输出，即 hi_venc_h264_adv_stream_info中的成员变量 qp_hist 为实际的当前帧的QP直方图。
max_user_data_len	用户数据所占内存的最大大小，单位Byte，暂不支持，默认值为1024。

1.9.20.14.43 hi_venc_h265_mod_param

说明

Device上的venc内h265e模块参数。

定义

```
typedef struct {  
    hi_u32 one_stream_buf;  
    hi_u32 mini_buf_mode;  
    hi_u32 low_power_mode;  
    hi_vb_src vb_src;  
    hi_bool qp_hist_en;  
    hi_u32 max_user_data_len;  
} hi_venc_h265_mod_param;
```

成员

成员名称	描述
one_stream_buf	编码码流帧配置模式，默认值为1。 0: 多包模式。 1: 单包模式。
mini_buf_mode	编码码流buffer配置模式，暂不支持，默认值为1。 0: 码流buffer 根据分辨率分配。 1: 码流buffer下限为32k，用户保证合理。
low_power_mode	低功耗模式，暂不支持，默认值为0。 0: 关闭低功耗模式。 1: 使能低功耗模式。 2: 使能极低功耗模式。 注意：低功耗或者极低功耗使能后会导致图像质量损失。

成员名称	描述
vb_src	参考帧和重构帧的帧存分配方式，默认值为HI_VB_SRC_PRIVATE。 - HI_VB_SRC_PRIVATE：私有VB方式，驱动内部完成帧存分配。 - HI_VB_SRC_USER：用户VB，需由用户分配帧存，暂不支持。
qp_hist_en	Qp 直方图输出控制模式，暂不支持，默认值为1。 0：Qp 直方图不输出，即 hi_venc_h265_adv_stream_info中的成员变量 qp_hist 为全0。 1：Qp 直方图输出，即 hi_venc_h265_adv_stream_info中的成员变量 qp_hist 为实际的当前帧的QP直方图 。
max_user_data_len	用户数据所占内存的最大大小，单位Byte，暂不支持，默认值为1024。

1.9.20.14.44 hi_venc_jpeg_mod_param

说明

Device上的venc内jpege模块中参数。

定义

```
typedef struct {  
    hi_u32 one_stream_buf;  
    hi_u32 mini_buf_mode;  
    hi_u32 clear_stream_buf;  
    hi_u32 dering_mode;  
} hi_venc_jpeg_mod_param;
```

成员

成员名称	描述
one_stream_buf	编码码流帧配置模式，默认值为1。 0：多包模式，暂不支持 1：单包模式
mini_buf_mode	编码码流buffer配置模式，暂不支持，默认值为0。 0：码流buffer根据分辨率分配。 1：码流buffer下限为32k，用户保证合理。

成员名称	描述
clear_stream_buf	JPEG 编码通道设置属性时是否清空码流buffer，暂不支持，默认值为1。 0：设置通道属性时保留码流buffer 和上下文计数。 1：设置通道属性时清空码流buffer。
dering_mode	预留参数，暂不使用，默认值为0。

1.9.20.14.45 hi_venc_jpeg_param

说明

用于设置或者获取指定JPEGE编码通道的量化参数的数据结构。

定义

```
typedef struct {
    hi_u32 qfactor;
    hi_u8 y_qt[64];
    hi_u8 cb_qt[64];
    hi_u8 cr_qt[64];
    hi_u32 mcu_per_ecs;
    hi_bool ecs_output_en;
} hi_venc_jpeg_param;
```

成员

成员名称	描述
qfactor	编码质量范围0xFFFFFFFF或[1, 100]，数值越小图片质量越差，默认100。 0xFFFFFFFF：配置成该值，则用户需要自行配置 y_qt、cb_qt、cr_qt，量化表的配置会影响最后的编码结果。 [1, 100]：配置成该范围内的值，表示y_qt、cb_qt、cr_qt参数不生效，以系统默认的为准。
y_qt	Y量化表。 取值范围：[1, 255]。
cb_qt	Cb量化表。 取值范围：[1, 255]。
cr_qt	Cr量化表。 取值范围：[1, 255]。
mcu_per_ecs	每个ECS中包含多少个MCU，系统默认为 0，表示不划分Ecs，预留参数，当前不支持。 mcu_per_ecs: [0, (picwidth+15)>>4*(picheight+15)>>4*2]

成员名称	描述
ecs_output_en	预留参数，暂不支持。

1.9.20.14.46 hi_venc_h264_adv_stream_info

说明

定义H.264协议码流高级特征信息。

Atlas 推理系列产品，预留结构体，暂不支持。

Atlas A2训练系列产品/Atlas 800I A2推理产品，预留结构体，暂不支持。

定义

```
typedef struct {  
    hi_u32 residual_bits;  
    hi_u32 head_bits;  
    hi_u32 madi_val;  
    hi_u32 madp_val;  
    hi_double psnr_val;  
    hi_u32 sse_lcu_cnt;  
    hi_u64 sse_sum;  
    hi_venc_sse_info sse_info[HI_VENC_MAX_SSE_NUM];  
    hi_u32 qp_hist[HI_VENC_QP_HIST_NUM];  
    hi_u32 move_scene16x16_num;  
    hi_u32 move_scene_bits;  
} hi_venc_h264_adv_stream_info;
```

成员

成员名称	描述
residual_bits	编码当前帧残差（bit）数。 Atlas 200/500 A2推理产品，暂不支持该参数。
head_bits	编码当前帧头信息的（bit）数。 Atlas 200/500 A2推理产品，暂不支持该参数。
madi_val	编码当前帧空域纹理复杂度Madi值。 Atlas 200/500 A2推理产品，暂不支持该参数。
madp_val	编码当前帧时域运动复杂度Madp值。 Atlas 200/500 A2推理产品，暂不支持该参数。
psnr_val	编码当前帧的PSNR（Peak signal-to-noise ratio）（峰值信噪比）值。 不同软件中关于峰值信噪比的计算公式略有不同，因此峰值信噪比值可能存在差异。 仅支持输入图片分辨率的乘积小于4096*2032时获取psnr_val值，否则psnr_val值无效。

成员名称	描述
sse_lcu_cnt	编码当前帧中LCU（ Largest Coding Unit ）个数 。
sse_sum	编码当前帧中SSE（ sum of squared errors ）（ 和方差 ）值。 不同软件中关于和方差的计算公式略有不同，因此和方差值可能存在差异。
sse_info	编码当前帧中8个区域的SSE（ 和方差 ）值。 不同软件中关于和方差的计算公式略有不同，因此和方差值可能存在差异。 #define HI_VENC_MAX_SSE_NUM 8
qp_hist	编码当前帧Qp直方图。 Atlas 200/500 A2推理产品，暂不支持该参数。 #define HI_VENC_QP_HIST_NUM 52
move_scene16x16_num	判断为图像前景的16*16块的数目，需要开启前景宏块级码控制。 Atlas 200/500 A2推理产品，暂不支持该参数。
move_scene_bits	判断为图像前景区域编码bit数，需要开启前景宏块级码控制。 Atlas 200/500 A2推理产品，暂不支持该参数。

1.9.20.14.47 hi_venc_h265_adv_stream_info

说明

定义H.265协议码流高级特征信息。

Atlas 推理系列产品，预留结构体，暂不支持。

Atlas A2训练系列产品/Atlas 800I A2推理产品，预留结构体，暂不支持。

定义

```
typedef struct {  
    hi_u32 residual_bits;  
    hi_u32 head_bits;  
    hi_u32 madi_val;  
    hi_u32 madp_val;  
    hi_double psnr_val;  
    hi_u32 sse_lcu_cnt;  
    hi_u64 sse_sum;  
    hi_venc_sse_info sse_info[HI_VENC_MAX_SSE_NUM];  
    hi_u32 qp_hist[HI_VENC_QP_HIST_NUM];  
    hi_u32 move_scene32x32_num;  
    hi_u32 move_scene_bits;  
} hi_venc_h265_adv_stream_info;
```


成员

成员名称	描述
residual_bits	编码当前帧残差（bit）数。 Atlas 200/500 A2推理产品，暂不支持该参数。
head_bits	编码当前帧头信息的（bit）数。 Atlas 200/500 A2推理产品，暂不支持该参数。
madi_val	编码当前帧空域纹理复杂度Madi值。 Atlas 200/500 A2推理产品，暂不支持该参数。
madp_val	编码当前帧时域运动复杂度Madp值。 Atlas 200/500 A2推理产品，暂不支持该参数。
psnr_val	编码当前帧的PSNR（峰值信噪比）值。 不同软件中关于峰值信噪比的计算公式略有不同，因此峰值信噪比值可能存在差异。 仅支持输入图片分辨率的乘积小于8192*4064时获取psnr_val值，否则psnr_val值无效。
sse_lcu_cnt	编码当前帧中LCU个数。
sse_sum	编码当前帧中SSE（和方差）值。 不同软件中关于和方差的计算公式略有不同，因此和方差值可能存在差异。
sse_info	编码当前帧中8个区域的SSE（和方差）值。 不同软件中关于和方差的计算公式略有不同，因此和方差值可能存在差异。 #define HI_VENC_MAX_SSE_NUM 8
qp_hist	编码当前帧Qp直方图。 Atlas 200/500 A2推理产品，暂不支持该参数。 #define HI_VENC_QP_HIST_NUM 52
move_scene32x32_num	判断为图像前景的32*32块的数目，需要开启前景宏块级码控制。 Atlas 200/500 A2推理产品，暂不支持该参数。
move_scene_bits	判断为图像前景区域编码bit数，需要开启前景宏块级码控制。 Atlas 200/500 A2推理产品，暂不支持该参数。

1.9.20.14.48 hi_venc_data_type

说明

定义码流结果类型。

定义

```
typedef union {  
    hi_venc_h264_nalu_type h264_type;  
    hi_venc_jpege_pack_type jpeg_type;  
    hi_venc_h265_nalu_type h265_type;  
    hi_venc_prores_pack_type prores_type;  
} hi_venc_data_type;
```

成员

成员名称	描述
h264_type	H.264码流包类型。
jpeg_type	JPEG码流包类型。
h265_type	H.265码流包类型。
prores_type	PRORES码流包类型。

1.9.20.14.49 hi_venc_h264_nalu_type

说明

定义H.264码流NALU类型。

定义

```
typedef enum {  
    HI_VENC_H264_NALU_B_SLICE = 0, // B SLICE types  
    HI_VENC_H264_NALU_P_SLICE = 1, // P SLICE types  
    HI_VENC_H264_NALU_I_SLICE = 2, // I SLICE types  
    HI_VENC_H264_NALU_IDR_SLICE = 5, // IDR SLICE types  
    HI_VENC_H264_NALU_SEI = 6, // SEI types  
    HI_VENC_H264_NALU_SPS = 7, // SPS types  
    HI_VENC_H264_NALU_PPS = 8, // PPS types  
    HI_VENC_H264_NALU_BUTT  
} hi_venc_h264_nalu_type;
```

1.9.20.14.50 hi_venc_jpege_pack_type

说明

定义JPEG码流的PACK类型。

定义

```
typedef enum {  
    HI_VENC_JPEG_PACK_ECS = 5, // ECS types  
    HI_VENC_JPEG_PACK_APP = 6, // APP types  
    HI_VENC_JPEG_PACK_VDO = 7, // VDO types  
    HI_VENC_JPEG_PACK_PIC = 8, // PIC types  
    HI_VENC_JPEG_PACK_DCF = 9, // DCF types  
    HI_VENC_JPEG_PACK_DCF_PIC = 10, // DCF PIC types  
    HI_VENC_JPEG_PACK_BUTT  
} hi_venc_jpege_pack_type;
```

1.9.20.14.51 hi_venc_h265_nalu_type

说明

定义H.265码流NALU类型。

定义

```
typedef enum {  
    HI_VENC_H265_NALU_B_SLICE = 0, // B SLICE types  
    HI_VENC_H265_NALU_P_SLICE = 1, // P SLICE types  
    HI_VENC_H265_NALU_I_SLICE = 2, // I SLICE types  
    HI_VENC_H265_NALU_IDR_SLICE = 19, // IDR SLICE types  
    HI_VENC_H265_NALU_VPS = 32, // VPS types  
    HI_VENC_H265_NALU_SPS = 33, // SPS types  
    HI_VENC_H265_NALU_PPS = 34, // PPS types  
    HI_VENC_H265_NALU_SEI = 39, // SEI types  
    HI_VENC_H265_NALU_ENHANCE = 64, // ENHANCE types  
    HI_VENC_H265_NALU_BUTT  
} hi_venc_h265_nalu_type;
```

1.9.20.14.52 hi_venc_prores_pack_type

说明

定义PRORES码流的PACK类型，暂不支持。

定义

```
typedef enum {  
    HI_VENC_PRORES_PACK_PIC = 1, // PIC types  
    HI_VENC_PRORES_PACK_BUTT  
} hi_venc_prores_pack_type;
```

1.9.20.14.53 hi_venc_chn_param

说明

定义Venc通道参数结构体。

定义

```
typedef struct {  
    hi_bool color_to_grey_en;  
    hi_u32 priority;  
    hi_u32 max_stream_cnt;  
    hi_u32 poll_wake_up_frame_cnt;  
    hi_crop_info crop_info;  
    hi_frame_rate_ctrl frame_rate;  
} hi_venc_chn_param;
```

成员

成员名称	描述
color_to_grey_en	开启或关闭一个通道的彩转灰功能。 Atlas 推理系列产品，预留参数，暂不支持。 Atlas A2训练系列产品/Atlas 800I A2推理产品，预留参数，暂不支持。

成员名称	描述
priority	编码通道优先级参数，默认为0，数字越大优先级越高。 取值范围：[0, 2)。
max_stream_cnt	最大码流缓存帧数，暂不支持。
poll_wake_up_frame_cnt	当通道使用超时或阻塞获取码流，编码指定的帧 poll_wake_up_frame_cnt之后唤醒阻塞接口，暂不支持。 取值范围：大于0 默认值：1
crop_info	预留参数，暂不支持。
frame_rate	通道帧率控制参数，暂不支持。

1.9.20.14.54 hi_venc_stream_info

说明

定义码流信息结构体。

Atlas 推理系列产品，预留结构体，暂不支持。

Atlas A2训练系列产品/Atlas 800I A2推理产品，预留结构体，暂不支持。

定义

```
typedef struct {  
    hi_venc_ref_type ref_type;  
    hi_u32 pic_bytes;  
    hi_u32 pic_cnt;  
    hi_u32 start_qp;  
    hi_u32 mean_qp;  
    hi_bool is_p_skip;  
  
    hi_u32 residual_bits;  
    hi_u32 head_bits;  
    hi_u32 madi_val;  
    hi_u32 madp_val;  
    hi_u64 sse_sum;  
    hi_u32 sse_lcu_cnt;  
    hi_double psnr_val;  
} hi_venc_stream_info;
```

成员

成员名称	描述
ref_type	高级跳帧参考下的编码帧类型。 Atlas 200/500 A2推理产品，暂不支持该参数。
pic_bytes	已编码码流的字节数。

成员名称	描述
pic_cnt	当通道属性is_by_frame == 1时，表示帧数；当is_by_frame == 0时，表示包数。 Atlas 200/500 A2推理产品，暂不支持该参数。
start_qp	编码帧的起始QP。
mean_qp	编码帧的平均QP。
is_p_skip	是否P帧是skip帧。 Atlas 200/500 A2推理产品，暂不支持该参数。
residual_bits	编码当前帧残差bit数。 Atlas 200/500 A2推理产品，暂不支持该参数。
head_bits	编码当前帧头信息bit数。 Atlas 200/500 A2推理产品，暂不支持该参数。
madi_val	编码当前帧空域纹理复杂度Madi值。 Atlas 200/500 A2推理产品，暂不支持该参数。
madp_val	编码当前帧时域运动复杂度Madp值。 Atlas 200/500 A2推理产品，暂不支持该参数。
sse_sum	编码当前帧中SSE（和方差）值。 不同软件中关于和方差的计算公式略有不同，因此和方差值可能存在差异。
sse_lcu_cnt	编码当前帧中LCU个数。
psnr_val	编码当前帧的PSNR（峰值信噪比）值。 不同软件中关于峰值信噪比的计算公式略有不同，因此峰值信噪比值可能存在差异。 对于H264，仅支持输入图片分辨率的乘积小于4096*2032时获取psnr_val值，否则psnr_val值无效。 对于H265，仅支持输入图片分辨率的乘积小于8192*4064时获取psnr_val值，否则psnr_val值无效。

1.9.20.14.55 hi_venc_pack_info

说明

定义当前码流包数据中包含的其他类型码流包数据的结构体。

定义

```
typedef struct {  
    hi_venc_data_type pack_type;  
    hi_u32 pack_offset;
```

```
hi_u32 pack_len;  
} hi_venc_pack_info;
```

成员

成员名称	描述
pack_type	当前码流包数据包含其他码流包的类型。
pack_offset	当前码流包数据包含其他码流包数据的偏移。
pack_len	当前码流包数据包含其他码流包数据的大小。

1.9.20.14.56 hi_frame_rate_ctrl

说明

定义通道帧率控制参数，暂不支持。

定义

```
typedef struct {  
    hi_s32 src_frame_rate;  
    hi_s32 dst_frame_rate;  
} hi_frame_rate_ctrl;
```

成员

成员名称	描述
src_frame_rate	进入通道的帧率，以fps为单位。 取值范围：-1 或[1, 240]。
dst_frame_rate	通道输出的帧率，以fps为单位。 取值范围： - Jpeg：-1 或[1, src_frame_rate]； - H.264/H.265：-1 或[1, 240]。

1.9.20.14.57 hi_venc_rc_qpmap_mode

说明

定义编码通道QP MAP模式下CU32、CU64 QP值的取值方式，暂不支持。

定义

```
typedef enum {  
    HI_VENC_RC_QPMAP_MODE_MEAN_QP = 0,  
    HI_VENC_RC_QPMAP_MODE_MIN_QP,  
    HI_VENC_RC_QPMAP_MODE_MAX_QP,  
}
```

```
    HI_VENC_RC_QPMAP_MODE_BUTT,  
} hi_venc_rc_qpmap_mode;
```

成员

成员名称	描述
HI_VENC_RC_QPMAP_MODE_MEAN_QP	CU32 QP值取4个16*16块Qp值的平均值； CU64 QP值取16个16*16块Qp值的平均值。
HI_VENC_RC_QPMAP_MODE_MIN_QP	CU32 QP值取4个16*16块Qp值的最小值； CU64 QP值取16个16*16块Qp值的最小值。
HI_VENC_RC_QPMAP_MODE_MAX_QP	CU32 QP值取4个16*16块Qp值的最大值； CU64 QP值取16个16*16块Qp值的最大值。
HI_VENC_RC_QPMAP_MODE_BUTT	保留值。

1.9.20.14.58 hi_venc_sse_info

说明

定义H264/H265协议SSE信息，暂不支持。

定义

```
typedef struct {  
    hi_bool enable;  
    hi_u64 sse_val;  
} hi_venc_sse_info;
```

成员

成员名称	描述
enable	区域SSE使能。1表示使能，0表示不使能。
sse_val	区域SSE值。

1.9.20.14.59 hi_venc_h264_attr

说明

定义H.264编码属性结构体。

Atlas 推理系列产品，预留结构体，暂不支持。

Atlas A2训练系列产品/Atlas 800I A2推理产品，预留结构体，暂不支持。

定义

```
typedef struct {  
    hi_bool rcn_ref_share_buf_en;  
    hi_u32 frame_buf_ratio;  
} hi_venc_h264_attr;
```

成员

成员名称	描述
rcn_ref_share_buf_en	是否使能参考帧和回写帧复用buffer。预留参数，暂不支持。 取值范围：[0,1]。1表示使能，0表示不使能。 静态属性。
frame_buf_ratio	帧buffer比例，设置该比例后，系统内部会根据该比例压缩帧buffer。 取值范围：[70,100]。比如将frame_buf_ratio设置为80，则表示帧buffer为原始大小的80%。 - 默认值为100，表示不压缩帧buffer； - 非取值范围内的值，表示不压缩帧buffer； [70,100)范围内的值，表示需按比例压缩帧buffer，可能会影响编码质量。建议在图片纹理不复杂、分辨率大于640*480等场景下，对于内存带宽有要求时，可以设置成[70,100)范围内的值。

1.9.20.14.60 hi_venc_h265_attr

说明

定义H.265编码属性结构体。

Atlas 推理系列产品，预留结构体，暂不支持。

Atlas A2训练系列产品/Atlas 800I A2推理产品，预留结构体，暂不支持。

定义

```
typedef struct {  
    hi_bool rcn_ref_share_buf_en;  
    hi_u32 frame_buf_ratio;  
} hi_venc_h265_attr;
```


成员

成员名称	描述
rcn_ref_share_buf_en	是否使能参考帧和回写帧复用buffer。预留参数，暂不支持。 取值范围：[0,1]。1表示使能，0表示不使能。 静态属性。
frame_buf_ratio	帧buffer比例，设置该比例后，系统内部会根据该比例压缩帧buffer。 取值范围：[70,100]。比如将frame_buf_ratio设置为80，则表示帧buffer为原始大小的80%。 - 默认值为100，表示不压缩帧buffer； - 非取值范围内的值，表示不压缩帧buffer； [70,100)范围内的值，表示需按比例压缩帧buffer，可能会影响编码质量。建议在图片纹理不复杂、分辨率大于640*480等场景下，对于内存带宽有要求时，可以设置成[70,100)范围内的值。

1.9.20.14.61 hi_venc_jpeg_attr

说明

定义JPEG编码属性结构体，暂不支持。

定义

```
typedef enum {  
    HI_VENC_PIC_RECV_SINGLE = 0,  
    HI_VENC_PIC_RECV_MULTIPLE,  
  
    HI_VENC_PIC_RECV_BUTT  
} hi_venc_pic_recv_mode;  
  
typedef struct {  
    hi_bool dcf_en;  
    hi_venc_mpf_cfg mpf_cfg;  
    hi_venc_pic_recv_mode recv_mode;  
} hi_venc_jpeg_attr;
```

1.9.20.14.62 hi_venc_prores_attr

说明

定义PRORES属性结构体，暂不支持。

定义

```
typedef enum {  
    HI_VENC_PRORES_FRAME_RATE_UNKNOWN = 0,  
    HI_VENC_PRORES_FRAME_RATE_23_976,  
    HI_VENC_PRORES_FRAME_RATE_24,  
    HI_VENC_PRORES_FRAME_RATE_25,  
}
```

```
HI_VENC_PRORES_FRAME_RATE_29_97,  
HI_VENC_PRORES_FRAME_RATE_30,  
HI_VENC_PRORES_FRAME_RATE_50,  
HI_VENC_PRORES_FRAME_RATE_59_94,  
HI_VENC_PRORES_FRAME_RATE_60,  
HI_VENC_PRORES_FRAME_RATE_100,  
HI_VENC_PRORES_FRAME_RATE_119_88,  
HI_VENC_PRORES_FRAME_RATE_120,  
HI_VENC_PRORES_FRAME_RATE_BUTT  
} hi_venc_prores_frame_rate;  
  
// the aspect ratio of PRORES  
typedef enum {  
    HI_VENC_PRORES_ASPECT_RATIO_UNKNOWN = 0,  
    HI_VENC_PRORES_ASPECT_RATIO_SQUARE,  
    HI_VENC_PRORES_ASPECT_RATIO_4_3,  
    HI_VENC_PRORES_ASPECT_RATIO_16_9,  
    HI_VENC_PRORES_ASPECT_RATIO_BUTT  
} hi_venc_prores_aspect_ratio;  
  
// the attribute of PRORES  
typedef struct {  
    hi_char identifier[HI_VENC_PRORES_MAX_ID_CHAR_NUM];  
    hi_venc_prores_frame_rate frame_rate_code;  
    hi_venc_prores_aspect_ratio aspect_ratio;  
} hi_venc_prores_attr;
```

1.9.20.14.63 hi_venc_scene_mode

说明

定义编码模式场景。

定义

```
typedef enum {  
    HI_VENC_SCENE_0 = 0, // 摄像机不运动或周期性连续运动的场景，支持h.264/h.265  
    HI_VENC_SCENE_1 = 1, // 高码率下运动场景，支持h.265  
    HI_VENC_SCENE_2 = 2, // 暂不支持  
    HI_VENC_SCENE_BUTT  
} hi_venc_scene_mode;
```

安防场景配置为HI_VENC_SCENE_0；智能驾驶、直播、游戏、动画、电影配置为HI_VENC_SCENE_1。

1.9.20.14.64 hi_venc_rc_param

说明

定义编码通道码率控制器的高级参数。

定义

```
typedef struct {  
    hi_u32 threshold_i[HI_VENC_TEXTURE_THRESHOLD_SIZE];  
    hi_u32 threshold_p[HI_VENC_TEXTURE_THRESHOLD_SIZE];  
    hi_u32 threshold_b[HI_VENC_TEXTURE_THRESHOLD_SIZE];  
    hi_u32 direction;  
    hi_u32 row_qp_delta;  
    hi_s32 first_frame_start_qp;  
    hi_venc_scene_chg_detect scene_chg_detect;  
    union {  
        hi_venc_h264_cbr_param h264_cbr_param;
```

```
hi_venc_h264_vbr_param h264_vbr_param;  
hi_venc_h264_avbr_param h264_avbr_param;  
hi_venc_h264_qvbr_param h264_qvbr_param;  
hi_venc_h264_cvbr_param h264_cvbr_param;  
hi_venc_h265_cbr_param h265_cbr_param;  
hi_venc_h265_vbr_param h265_vbr_param;  
hi_venc_h265_avbr_param h265_avbr_param;  
hi_venc_h265_qvbr_param h265_qvbr_param;  
hi_venc_h265_cvbr_param h265_cvbr_param;  
hi_venc_mjpeg_cbr_param mjpeg_cbr_param;  
hi_venc_mjpeg_vbr_param mjpeg_vbr_param;  
};  
}hi_venc_rc_param;
```

成员

成员名称	描述
threshold_i	I帧宏块级码率控制的Madi（用于度量当前帧的空域纹理复杂度）门限。 取值范围：[0, 255]。 默认值：[0,0,0,0,3,3,5,5,8,8,15,15,20,25,25]。 减方向的数值设置为0，表示关闭当前级；加方向的数值设置为255，表示关闭当前级。
threshold_p	P帧宏块级码率控制的Madi门限。 取值范围：[0, 255]。 默认值：[0,0,0,0,3,3,5,5,8,8,15,15,20,25,25]。 减方向的数值设置为0，表示关闭当前级；加方向的数值设置为255，表示关闭当前级。
threshold_b	B帧宏块级码率控制的Madi门限。 取值范围：[0, 255]。 默认值：[0,0,0,0,3,3,5,5,8,8,15,15,20,25,25]。 减方向的数值设置为0，表示关闭当前级；加方向的数值设置为255，表示关闭当前级。
direction	在基于纹理宏块级码率控制时，用于控制加减方向。 取值范围：[0, 16]。 默认值：8。 举例，direction=7 表示threshold_i、threshold_p、threshold_b的前7个参数用于QP减方向；后9个参数用于QP加方向。
row_qp_delta	行级码率控制调节幅度是一帧内行级调节的最大范围，其中行级以宏块行为单位。调节幅度越大，允许行级调整的QP范围越大，码率越平稳。对于图像复杂度分布不均匀的场景，行级码率控制调节幅度设置过大会带来图像质量不均匀。 取值范围：[0, 10]，设置为0表示关闭基于行的宏块级码率控制。 默认值：1。

成员名称	描述
first_frame_start_qp	设置第一帧的起始Qp值，CBR/VBR/AVBR/QVBR/CVBR模式下，该参数有效。此处第一帧的含义是：通道创建，GOP模式切换，RC模式切换，或分辨率切换后，序列的第一个IDR帧。需要注意如果第一帧编码完成后判定为重编，重编的帧不是第一帧，不受 first_frame_start_qp的约束。 取值范围：[MinIQP, MaxIQP]和-1。MinIQP参数值、MaxIQP参数值在码率控制模式的高级参数结构体内设置。 默认值：-1。如果为-1，则第一帧的起始QP由编码器内部计算，如果为其它合法值则由用户通过该参数设置第一帧的起始QP。 在低码率、少量运动场景下，如果编码效果不好，可以通过降低该参数值来提升图像质量和效果，例如，在“352*288分辨率，帧率25fps，GOP=50”场景下，将first_frame_start_qp设置为32，同时将max_reencode_times参数设置为0（表示不重编码，该参数是以下码率控制模式高级参数之一）。
scene_chg_detect	场景检测相关的控制参数，自适应检测当前编码场景是否发生变化，使能后会改善场景切换时的码率波动，VBR/AVBR/QVBR/CVBR 有效。使能场景切换检测时可使能自适应插入IDR帧，在场景切换时编码IDR帧，需要注意关闭重编后检测场景切换会失效。该参数建议与编码场景模式的SCENE_0模式配合使用。
h264_cbr_param	H.264通道CBR（Constant Bit Rate）码率控制模式高级参数。
h264_vbr_param	H.264通道VBR（Variable Bit Rate）码率控制模式高级参数。
h264_avbr_param	H.264通道AVBR（Adaptive Variable Bit Rate）码率控制模式高级参数。
h264_qvbr_param	H.264通道QVBR（Quality Variable Bit Rate）码率控制模式高级参数。
h264_cvbr_param	H.264通道CVBR（Constrained Variable Bit Rate）码率控制模式高级参数。
h265_cbr_param	H.265通道CBR（Constant Bit Rate）码率控制模式高级参数。
h265_vbr_param	H.265通道VBR（Variable Bit Rate）码率控制模式高级参数。
h265_avbr_param	H.265通道AVBR（Adaptive Variable Bit Rate）码率控制模式高级参数。
h265_qvbr_param	H.265通道QVBR（Quality Variable Bit Rate）码率控制模式高级参数。

成员名称	描述
h265_cvbr_param	H.265通道CVBR (Constrained Variable Bit Rate)码率控制模式高级参数。
mjpeg_cbr_param	MJPEG通道CBR (Constant Bit Rate) 码率控制模式高级参数。预留参数，暂不支持。
mjpeg_vbr_param	MJPEG通道VBR (Variable Bit Rate) 码率控制模式高级参数。预留参数，暂不支持。

1.9.20.14.65 hi_venc_h264_cbr_param

说明

定义H264协议编码通道CBR码率控制模式的高级参数。

定义

```
typedef struct {  
    hi_u32 max_i_proportion;  
    hi_u32 min_i_proportion;  
    hi_u32 max_qp;  
    hi_u32 min_qp;  
    hi_u32 max_i_qp;  
    hi_u32 min_i_qp;  
    hi_s32 max_reencode_times;  
    hi_bool qpmap_en;  
}hi_venc_h264_cbr_param;
```

成员

成员名称	描述
min_i_proportion	最小IP帧比例，预留参数，暂不支持。
max_i_proportion	最大IP帧比例。 取值范围：[1, 100]。 该参数默认值为100。
max_qp	帧最大QP，用于钳位质量。 取值范围：[0, 51]。 默认值：51。
min_qp	帧最小QP，用于钳位码率波动。 取值范围：[0, max_qp]。 默认值：10。
min_i_qp	I帧的最小QP。用于控制I帧的最大bits数。 取值范围：[0, max_i_qp]。 默认值：10。

成员名称	描述
max_i_qp	I帧的最大QP。用于控制I帧的最小bits数。 取值范围：[0, 51]。 默认值：51。
max_reencode_times	每帧重编码次数。0表示不进行重编码。 取值范围：[0, 3]。 默认值：2。
qpmap_en	预留参数，暂不支持。

1.9.20.14.66 hi_venc_h264_vbr_param

说明

定义H264协议编码通道VBR码率控制模式的高级参数。

定义

```
typedef struct {  
    hi_s32 chg_pos;  
    hi_u32 max_i_proportion;  
    hi_u32 min_i_proportion;  
    hi_s32 max_reencode_times;  
    hi_bool qpmap_en;  
    hi_u32 max_qp;  
    hi_u32 min_qp;  
    hi_u32 max_i_qp;  
    hi_u32 min_i_qp;  
} hi_venc_h264_vbr_param;
```

成员

max_qp、min_qp用于控制图像的质量范围，max_bit_rate用于钳位码率统计时间内的最大编码码率，chg_pos用于控制开始调整QP的码率基准线。

当编码码率大于max_bit_rate*chg_pos时，图像QP会逐步向max_qp调整，如果图像QP达到max_qp，QP会被钳位到最大值，max_bit_rate的钳位效果失效，编码码率有可能会超出max_bit_rate。

当编码码率小于max_bit_rate*chg_pos时，图像QP会逐步向min_qp调整，如果图像QP达到min_qp，此时编码的码率已经达到最大值，而且图像质量最好。

 说明

max_bit_rate参数在创建VENC通道时设置，是hi_venc_chn_attr.rc_attr结构体内的成员变量。

成员名称	描述
chg_pos	VBR开始调整Qp时的码率相对于最大码率的比例。 取值范围：[50, 100]。 默认值：90。

成员名称	描述
min_i_proportion	预留参数，暂不支持。
max_i_proportion	最大IP帧码率的比值。 取值范围：[1,100]。 该参数默认值为100。
max_reencode_times	每帧重编码次数。0 表示不进行重编码。 取值范围：[0, 3]。 默认值：2。
qpmap_en	预留参数，暂不支持。
max_qp	P、B帧的最大QP。取值范围：[0, 51]。 默认值：51。
min_qp	P、B帧的最小QP。 取值范围：[0, max_qp]。 默认值：24。
max_i_qp	I帧的最大QP。 取值范围：[0, 51]。 默认值：51。
min_i_qp	I帧的最小QP。 取值范围：[0, max_i_qp]。 默认值：24。

1.9.20.14.67 hi_venc_h264_avbr_param

说明

定义H264协议编码通道AVBR码率控制模式的高级参数。

定义

```
typedef struct {  
    hi_s32 chg_pos;  
    hi_u32 max_i_proportion;  
    hi_u32 min_i_proportion;  
    hi_s32 max_reencode_times;  
    hi_bool qpmap_en;  
    hi_s32 min_still_percent;  
    hi_u32 max_still_qp;  
    hi_u32 min_still_psnr;  
    hi_u32 max_qp;  
    hi_u32 min_qp;  
    hi_u32 max_i_qp;  
    hi_u32 min_i_qp;  
    hi_u32 min_qp_delta;  
    hi_u32 motion_sensitivity;  
    hi_bool save_bitrate_en;  
}hi_venc_h264_avbr_param;
```

成员

max_bit_rate表示运动场景下的最大码率，max_bit_rate*chg_pos*min_still_percent表示静止情况下的最小码率。

根据运动程度的不同，目标码率会在最大码率和最小码率间调整。max_qp，min_qp用于控制图像的质量范围，码率控制以QP钳位为最高优先级，超出min_qp，max_qp范围内码率控制将失效。

说明

max_bit_rate参数在创建VENC通道时设置，是hi_venc_chn_attr.rc_attr结构体内的成员变量。

成员名称	描述
chg_pos	AVBR开始调整Qp时的码率相对于最大码率的比例。 取值范围：[50, 100]。 默认值：65。
min_i_proportion	预留参数，暂不支持。
max_i_proportion	最大IP帧码率的比值。 取值范围：[1,100]。 默认值：100。
max_reencode_times	每帧重编码次数。0 表示不进行重编码。 取值范围：[0, 3]。 默认值：2。
qpmap_en	预留参数，暂不支持。
min_still_percent	静止状态下目标码率的最小百分比。此变量设置为100，AVBR将不会在判别为静止时主动调低目标码率。 取值范围：[5, 100]。 默认值：25。
max_still_qp	静止场景I帧QP的最大值。 取值范围：[min_i_qp, max_i_qp]。 默认值：35。
min_still_psnr	预留参数，暂不支持。
max_qp	P、B帧的最大QP。 取值范围：[0, 51]。 默认值：51。
min_qp	P、B帧的最小QP。 取值范围：[0, max_qp]。 默认值：24。

成员名称	描述
max_i_qp	I帧的最大QP。 取值范围：[0, 51]。 默认值：51。
min_i_qp	I帧的最小QP。 取值范围：[0, max_i_qp]。 默认值：24。
min_qp_delta	帧级QP最小值和CU级QP最小值的差值。 I帧，帧级QP最小值= min_qp_delta + min_i_qp P、B帧，帧级QP最小值= min_qp_delta+min_qp 取值范围：[0, 4]。 默认值：0。
motion_sensitivity	运动敏感度。 取值范围：[0, 100]。 默认值：100。
save_bitrate_en	预留参数，暂不支持。

1.9.20.14.68 hi_venc_h264_qvbr_param

说明

定义H264协议编码通道QVBR码率控制模式的高级参数。

定义

```
typedef struct {  
    hi_u32 max_i_proportion;  
    hi_u32 min_i_proportion;  
    hi_s32 max_reencode_times;  
    hi_bool qpmap_en;  
    hi_u32 max_qp;  
    hi_u32 min_qp;  
    hi_u32 max_i_qp;  
    hi_u32 min_i_qp;  
    hi_s32 max_bit_percent;  
    hi_s32 min_bit_percent;  
    hi_s32 max_psnr_fluctuate;  
    hi_s32 min_psnr_fluctuate;  
} hi_venc_h264_qvbr_param;
```

成员

当实时统计的PSNR小于max_psnr_fluctuate时，适当增加目标码率，最大码率=target_bit_rate* max_bit_percent。

当实时统计的PSNR大于max_psnr_fluctuate时，适当减小目标码率，最小码率=target_bit_rate* min_bit_percent。

根据当前PSNR的不同，目标码率会在最大码率和最小码率间调整，当PSNR值范围超过 $[\text{min\_psnr\_fluctuate}-4, \text{max\_psnr\_fluctuate}+4] \cap [20, 40]$ 时，PSNR不再起作用，码率会在最大码率和最小码率间调整。

max_qp, min_qp用于控制图像的质量范围，码率控制以QP钳位为最高优先级，超出min_qp, max_qp范围内码率控制将失效。

码率浮动上下限的优先级高于PSNR的优先级，例如，当码率浮动到上限依然不能满足PSNR要求，则码率不会再继续上调。

 说明

target_bit_rate参数在创建VENC通道时设置，是hi_venc_chn_attr.rc_attr结构体内的成员变量。

成员名称	描述
min_i_proportion	预留参数，暂不支持。
max_i_proportion	最大IP帧码率的比值。 取值范围：[1,100]。 默认值：100。
max_reencode_times	每帧重编码次数。0表示不进行重编码。 取值范围：[0, 3]。 默认值：2。
qpmap_en	预留参数，暂不支持。
max_qp	P、B帧的最大QP。 取值范围：[min_qp, 51]。 默认值：51。
min_qp	P、B帧的最小QP。 取值范围：[0, 51]。 默认值：16。
max_i_qp	I帧的最大QP。 取值范围：[min_i_qp, 51]。 默认值：51。
min_i_qp	I帧的最小QP。 取值范围：[0, 51]。 默认值：16。
max_bit_percent	码率百分比上限。 取值范围：[min_bit_percent, 180]。 默认值：110。
min_bit_percent	码率百分比下限。 取值范围：[30, 180]。 默认值：45。

成员名称	描述
max_psnr_fluctuate	Psnr上限。 取值范围：[min_psnr_fluctuate, 40]。 默认值：40。
min_psnr_fluctuate	Psnr下限。 取值范围：[18, 40]。 默认值：23。

1.9.20.14.69 hi_venc_h264_cvbr_param

说明

定义H264协议编码通道CVBR码率控制模式的高级参数。

定义

```
typedef struct {  
    hi_u32 max_i_proportion;  
    hi_u32 min_i_proportion;  
    hi_s32 max_reencode_times;  
    hi_bool qpmap_en;  
    hi_u32 max_qp;  
    hi_u32 min_qp;  
    hi_u32 max_i_qp;  
    hi_u32 min_i_qp;  
    hi_u32 min_qp_delta;  
    hi_u32 max_qp_delta;  
    hi_u32 extra_bit_percent;  
    hi_u32 long_term_stats_time_unit;  
    hi_bool save_bitrate_en;  
} hi_venc_h264_cvbr_param;
```

成员

成员名称	描述
min_i_proportion	预留参数，暂不支持。
max_i_proportion	最大IP帧码率的比值。 取值范围：[1,100]。 默认值：100。
max_reencode_times	每帧重编码次数。0 表示不进行重编码。 取值范围：[0, 3]。 默认值：2。
qpmap_en	预留参数，暂不支持。

成员名称	描述
max_qp	P、B帧的最大QP。 取值范围：[min_qp, 51]。 默认值：47。
min_qp	P、B帧的最小QP。 取值范围：[0, 51]。 默认值：22。
max_i_qp	I帧的最大QP。 取值范围：[min_i_qp, 51]。 默认值：47。
min_i_qp	I帧的最小QP。 取值范围：[0, 51]。 默认值：20。
min_qp_delta	帧级QP最小值和CU级QP最小值的差值。 I帧，帧级QP最小值 = min_qp_delta + min_i_qp P、B帧，帧级QP最小值= min_qp_delta+min_qp 取值范围：[0, 4]。 默认值：0。
max_qp_delta	帧级QP最大值和CU级QP最大值的差值。 I帧，帧级QP最大值 = max_i_qp - max_qp_delta P、B 帧：帧级QP最大值= max_qp - max_qp_delta 取值范围：[0, 4]。 默认值：0。
extra_bit_percent	编码器输出码流最大透支bit 数百分比。 在码率不足时，为保证图像质量，编码器会通过透支一定的bit数以提升图像质量，这部分透支的码率会在编码压力较小时进行偿还。 取值范围：[0, 1000]。 默认值：5。
long_term_stats_time_unit	码率长期统计时间参数（long_term_stats_time）的单位，本变量单位是秒（s），如配置long_term_stats_time_unit = 60，long_term_stats_time = 3，代表长期统计时间为3 分钟。 取值范围：[1, 1800]。 默认值：60。
save_bitrate_en	预留参数，暂不支持。

1.9.20.14.70 hi_venc_h265_vbr_param

说明

定义H265协议编码通道VBR码率控制模式的高级参数。

定义

```
typedef struct {  
    hi_s32 chg_pos;  
    hi_u32 max_i_proportion;  
    hi_u32 min_i_proportion;  
    hi_s32 max_reencode_times;  
    hi_u32 max_qp;  
    hi_u32 min_qp;  
    hi_u32 max_i_qp;  
    hi_u32 min_i_qp;  
    hi_bool qpmap_en;  
    hi_venc_rc_qpmap_mode qpmap_mode;  
}hi_venc_h265_vbr_param;
```

成员

max_qp、min_qp用于控制图像的质量范围，max_bit_rate用于钳位码率统计时间内的最大编码码率，chg_pos用于控制开始调整QP的码率基准线。

当编码码率大于max_bit_rate*chg_pos时，图像QP会逐步向max_qp调整，如果图像QP达到max_qp，QP会被钳位到最大值，max_bit_rate的钳位效果失效，编码码率有可能会超出max_bit_rate。

当编码码率小于max_bit_rate*chg_pos时，图像QP会逐步向min_qp调整，如果图像QP达到min_qp，此时编码的码率已经达到最大值，而且图像质量最好。

 说明

max_bit_rate参数在创建VENC通道时设置，是hi_venc_chn_attr.rc_attr结构体内的成员变量。

成员名称	描述
chg_pos	开始调整Qp时的码率相对于最大码率的比例。 取值范围：[50, 100]。 默认值：90。
max_i_proportion	最大IP 帧码率的比值。 取值范围：[1,100]。 该参数默认值为100。
min_i_proportion	预留参数，暂不支持。
max_reencode_times	每帧重编码次数。0 表示不进行重编码。 取值范围：[0, 3]。 默认值：2。
max_qp	P帧的最大QP。 取值范围：[0, 51]。 默认值：51。

成员名称	描述
min_qp	P帧的最小QP。 取值范围：[0, max_qp]。 默认值：24。
max_i_qp	I帧的最大QP。 取值范围：[0, 51]。 默认值：51。
min_i_qp	I帧的最小QP。 取值范围：[0, max_i_qp]。 默认值：24。
qpmap_en	预留参数，暂不支持。
qpmap_mode	预留参数，暂不支持。

1.9.20.14.71 hi_venc_h265_avbr_param

说明

定义H265协议编码通道AVBR码率控制模式的高级参数。

定义

```
typedef struct {  
    hi_s32 chg_pos;  
    hi_u32 max_i_proportion;  
    hi_u32 min_i_proportion;  
    hi_s32 max_reencode_times;  
    hi_s32 min_still_percent;  
    hi_u32 max_still_qp;  
    hi_u32 min_still_psnr;  
    hi_u32 max_qp;  
    hi_u32 min_qp;  
    hi_u32 max_i_qp;  
    hi_u32 min_i_qp;  
    hi_u32 min_qp_delta;  
    hi_u32 motion_sensitivity;  
    hi_bool qpmap_en;  
    hi_venc_rc_qpmap_mode qpmap_mode;  
}hi_venc_h265_avbr_param;
```

成员

max_bit_rate表示运动场景下的最大码率，max_bit_rate*chg_pos*min_still_percent表示静止情况下的最小码率。

根据运动程度的不同，目标码率会在最大码率和最小码率间调整。max_qp，min_qp用于控制图像的质量范围，码率控制以QP钳位为最高优先级，超出min_qp，max_qp范围内码率控制将失效。

 说明

max_bit_rate参数在创建VENC通道时设置，是hi_venc_chn_attr.rc_attr结构体内的成员变量。

成员名称	描述
chg_pos	开始调整Qp 时的码率相对于最大码率的比例。 取值范围：[50, 100]。 默认值：65。
min_i_proportion	预留参数，暂不支持。
max_i_proportion	最大IP帧码率的比值。 取值范围：[1,100]。 默认值：100。
max_reencode_times	每帧重编码次数。0 表示不进行重编码。 取值范围：[0, 3]。 默认值：2。
min_still_percent	静止状态下目标码率的最小百分比。此变量设置为100，AVBR 将不会在判别为静止时主动调低目标码率。 取值范围：[5, 100]。 默认值：25。
max_still_qp	静止场景I帧QP的最大值。 取值范围：[min_i_qp, max_i_qp]。 默认值：35。
min_still_psnr	预留参数，暂不支持。
max_qp	P帧的最大QP。 取值范围：[0, 51]。 默认值：51。
min_qp	P帧的最小QP。 取值范围：[0, max_qp]。 默认值：24。
max_i_qp	I帧的最大QP。 取值范围：[0, 51]。 默认值：51。
min_i_qp	I帧的最小QP。 取值范围：[0, max_i_qp]。 默认值：24。
min_qp_delta	帧级QP最小值和CU级QP最小值的差值。 I帧：FrameLevelMinQp = min_qp_delta + min_i_qp P、B 帧：FrameLevelMinQp= min_qp_delta+min_qp 取值范围：[0, 4]。 默认值：0。

成员名称	描述
motion_sensitivity	运动敏感度。 取值范围：[0, 100]。 默认值：100。
qpmap_en	预留参数，暂不支持。
qpmap_mode	预留参数，暂不支持。

1.9.20.14.72 hi_venc_h265_qvbr_param

说明

定义H265协议编码通道QVBR码率控制模式的高级参数。

定义

```
typedef struct {  
    hi_u32 max_i_proportion;  
    hi_u32 min_i_proportion;  
    hi_s32 max_reencode_times;  
    hi_bool qpmap_en;  
    hi_venc_rc_qpmap_mode qpmap_mode;  
    hi_u32 max_qp;  
    hi_u32 min_qp;  
    hi_u32 max_i_qp;  
    hi_u32 min_i_qp;  
    hi_s32 max_bit_percent;  
    hi_s32 min_bit_percent;  
    hi_s32 max_psnr_fluctuate;  
    hi_s32 min_psnr_fluctuate;  
} hi_venc_h265_qvbr_param;
```

成员

当实时统计的PSNR小于max_psnr_fluctuate时，适当增加目标码率，最大码率=target_bit_rate* max_bit_percent。

当实时统计的PSNR大于max_psnr_fluctuate时，适当减小目标码率，最小码率=target_bit_rate* min_bit_percent。

根据当前PSNR的不同，目标码率会在最大码率和最小码率间调整，当PSNR值范围超过[min_psnr_fluctuate-4,max_psnr_fluctuate+4]∩[20,40]时，PSNR不再起作用，码率会在最大码率和最小码率间调整。

max_qp，min_qp用于控制图像的质量范围，码率控制以QP钳位为最高优先级，超出min_qp，max_qp范围内码率控制将失效。

码率浮动上下限的优先级高于PSNR的优先级，例如，当码率浮动到上限依然不能满足PSNR要求，则码率不会再继续上调。

说明

target_bit_rate参数在创建VENC通道时设置，是hi_venc_chn_attr.rc_attr结构体内的成员变量。

成员名称	描述
min_i_proportion	预留参数，暂不支持。
max_i_proportion	最大IP 帧码率的比值。 取值范围：[1,100]。 默认值：100。
max_reencode_times	每帧重编码次数。0 表示不进行重编码。 取值范围：[0, 3]。 默认值：2。
qpmap_en	预留参数，暂不支持。
qpmap_mode	预留参数，暂不支持。
max_qp	P、B帧的最大QP。 取值范围：[min_qp, 51]。 默认值：51。
min_qp	P、B帧的最小QP。 取值范围：[0, 51]。 默认值：16。
max_i_qp	I帧的最大QP。 取值范围：[min_i_qp, 51]。 默认值：51。
min_i_qp	I帧的最小QP。 取值范围：[0, 51]。 默认值：16。
max_bit_percent	码率百分比上限。 取值范围：[min_bit_percent, 180]。 默认值：110。
min_bit_percent	码率百分比下限。 取值范围：[30, 180]。 默认值：45。
max_psnr_fluctuate	Psnr上限。 取值范围：[min_psnr_fluctuate, 40]。 默认值：40。
min_psnr_fluctuate	Psnr下限。 取值范围：[18, 40]。 默认值：23。

1.9.20.14.73 hi_venc_h265_cvbr_param

说明

定义H265协议编码通道CVBR码率控制模式的高级参数。

定义

```
typedef struct {  
    hi_u32 max_i_proportion;  
    hi_u32 min_i_proportion;  
    hi_s32 max_reencode_times;  
    hi_bool qpmap_en;  
    hi_venc_rc_qpmap_mode qpmap_mode;  
    hi_u32 max_qp;  
    hi_u32 min_qp;  
    hi_u32 max_i_qp;  
    hi_u32 min_i_qp;  
    hi_u32 min_qp_delta;  
    hi_u32 max_qp_delta;  
    hi_u32 extra_bit_percent;  
    hi_u32 long_term_stats_time_unit;  
} hi_venc_h265_cvbr_param;
```

成员

成员名称	描述
min_i_proportion	预留参数，暂不支持。
max_i_proportion	最大IP 帧码率的比值。 取值范围：[1,100]。 默认值：100。
max_reencode_times	每帧重编码次数。0 表示不进行重编码。 取值范围：[0, 3]。 默认值：2。
qpmap_en	预留参数，暂不支持。
qpmap_mode	预留参数，暂不支持。
max_qp	P、B帧的最大QP。 取值范围：[min_qp, 51]。 默认值：47。
min_qp	P、B帧的最小QP。 取值范围：[0, 51]。 默认值：22。
max_i_qp	I帧的最大QP。 取值范围：[min_i_qp, 51]。 默认值：47。

成员名称	描述
min_i_qp	I帧的最小QP。 取值范围：[0, 51]。 默认值：20。
min_qp_delta	帧级QP最小值和CU级QP最小值的差值。 I 帧：FrameLevelMinQp = min_qp_delta + min_i_qp P、B帧：FrameLevelMinQp= min_qp_delta+min_qp 取值范围：[0, 4]。 默认值：0。
max_qp_delta	帧级QP最大值和CU级QP最大值的差值。 I帧，帧级QP最大值 = max_i_qp - max_qp_delta P、B帧，帧级QP最大值= max_qp - max_qp_delta 取值范围：[0, 4]。 默认值：0。
extra_bit_percent	编码器输出码流最大透支bit 数百分比。 在码率不足时，为保证图像质量，编码器会通过透支一定的bit 数以提升图像质量，这部分透支的码率会在编码压力较小时进行偿还。 取值范围[0, 1000]。 默认值：5。
long_term_stats_time_unit	码率长期统计时间参数（long_term_stats_time）的单位，本变量单位是秒（s），如配置long_term_stats_time_unit = 60，long_term_stats_time =3，代表长期统计时间为3 分钟。 取值范围：[1, 1800]。 默认值：60。

1.9.20.14.74 hi_venc_h265_cbr_param

说明

定义H265协议编码通道CBR码率控制模式的高级参数。

定义

```
typedef struct {  
    hi_u32 max_i_proportion;  
    hi_u32 min_i_proportion;  
    hi_u32 max_qp;  
    hi_u32 min_qp;  
    hi_u32 max_i_qp;  
    hi_u32 min_i_qp;  
    hi_s32 max_reencode_times;  
    hi_bool qpmap_en;  
}
```

```
    hi_venc_rc_qpmap_mode qpmap_mode;  
}hi_venc_h265_cbr_param;
```

成员

成员名称	描述
min_i_proportion	预留参数，暂不支持。
max_i_proportion	最大IP帧比例。 取值范围：[1, 100]。 该参数默认值为100。
max_qp	帧最大QP，用于钳位质量。 取值范围：[0, 51]。 默认值：51。
min_qp	帧最小QP，用于钳位码率波动。 取值范围：[0, max_qp]。 默认值：10。
max_i_qp	I帧的最大QP。用于控制I帧的最小bits数。 取值范围：[0, 51]。 默认值：51。
min_i_qp	I帧的最小QP。用于控制I帧的最大bits数。 取值范围：[0, max_i_qp]。 默认值：10。
max_reencode_times	每帧重编码次数。0表示不进行重编码。 取值范围：[0, 3]。 默认值：2。
qpmap_en	预留参数，暂不支持。
qpmap_mode	预留参数，暂不支持。

1.9.20.14.75 hi_venc_scene_chg_detect

说明

定义编码场景检测控制参数。

定义

```
typedef struct {  
    hi_bool detect_scene_chg_en;  
    hi_bool adapt_insert_idr_frame_en;  
}hi_venc_scene_chg_detect;
```

成员

成员名称	描述
detect_scene_chg_en	是否使能场景切换检测。 取值范围如下： - HI_TRUE: 是，默认为该值 - HI_FALSE: 否
adapt_insert_idr_frame_en	是否使能自适应插入IDR帧。 取值范围如下： - HI_TRUE: 是 - HI_FALSE: 否，默认为该值

1.9.20.14.76 hi_venc_huffman_dc_table

说明

用于定义指定JPEGE编码通道的亮度DC huffman编码表参数的数据结构。

定义

```
typedef struct {  
    hi_u8 dc_bits[16];  
    hi_u8 dc_value[12];  
} hi_venc_huffman_dc_table;
```

成员

成员名称	描述
dc_bits[]	dc_bits数组存放的是长度从1bit到16bit的码字分别对应的符号个数。
dc_value[]	dc_value数据存放的是码字对应的符号值。

1.9.20.14.77 hi_venc_huffman_ac_table

说明

用于定义指定JPEGE编码通道的色度AC huffman编码表参数的数据结构。

定义

```
typedef struct {  
    hi_u8 ac_bits[16];  
    hi_u8 ac_value[162];  
} hi_venc_huffman_ac_table;
```

成员

成员名称	描述
ac_bits[]	ac_bits数组存放的是长度从1bit到16bit的码字分别对应的符号个数。
ac_value[]	ac_value数据存放的是码字对应的符号值。

1.9.20.14.78 hi_venc_jpeg_huffman_param

说明

用于定义指定JPEGE编码通道的huffman编码高级参数的数据结构。

定义

```
typedef struct {  
    hi_venc_huffman_dc_table dc_tables[3];  
    hi_venc_huffman_ac_table ac_tables[3];  
    hi_u32 reserved[2];  
} hi_venc_jpeg_huffman_param;
```

成员

成员名称	描述
dc_tables[]	亮度DC huffman编码表数据。 Cb分量和Cr分量需保持一致。
ac_tables[]	色度AC huffman编码表数据。 Cb分量和Cr分量需保持一致。
reserved[]	预留参数，为保证后续版本兼容性，请务必使用memset结构体方式进行清零初始化，在代码中必须避免显式对reserved字段进行访问。

1.9.20.14.79 hi_venc_mpf_cfg

说明

预留结构体，暂不支持。

定义

```
typedef struct {  
    hi_u8 large_thumbnail_num;  
    hi_video_size large_thumbnail_size[HI_VENC_MAX_MPF_NUM];  
} hi_venc_mpf_cfg;
```

1.9.20.14.80 hi_venc_mjpeg_cbr_param

说明

预留结构体，暂不支持。

定义

```
typedef struct {  
    hi_u32    max_qfactor;  
    hi_u32    min_qfactor;  
} hi_venc_mjpeg_cbr_param;
```

1.9.20.14.81 hi_venc_mjpeg_vbr_param

说明

预留结构体，暂不支持。

定义

```
typedef struct {  
    hi_s32    chg_pos;  
    hi_u32    max_qfactor;  
    hi_u32    min_qfactor;  
} hi_venc_mjpeg_vbr_param;
```

1.9.20.14.82 hi_venc_h264_ref_slice_type

说明

定义H.264 Slice码流的参考类型，预留，暂不支持。

定义

```
typedef enum {  
    HI_VENC_H264_REF_SLICE_FOR_1X = 1, // Reference slice for H264E_REF_MODE_1X  
    HI_VENC_H264_REF_SLICE_FOR_2X = 2, // Reference slice for H264E_REF_MODE_2X  
    HI_VENC_H264_REF_SLICE_FOR_4X = 5, // Reference slice for H264E_REF_MODE_4X  
    HI_VENC_H264_REF_SLICE_FOR_BUTT // slice not for reference  
} hi_venc_h264_ref_slice_type;
```

1.9.20.14.83 hi_venc_roi_attr

说明

定义编码感兴趣区域信息。

定义

```
typedef struct {  
    hi_u32    idx;  
    hi_bool    enable;  
    hi_bool    is_abs_qp;  
    hi_s32    qp;  
    hi_rect    rect;  
} hi_venc_roi_attr;
```

成员

成员名称	描述
idx	ROI区域的索引，系统支持的索引范围为[0,7]，不支持超出这个范围的索引。 系统支持每个通道可设置8个ROI区域，系统内部按照0~7的索引号对ROI区域进行管理。ROI区域之间可以互相叠加，且当发生叠加时，ROI区域之间的优先级按照索引号0~7依次提高。
enable	是否使能这个ROI区域。 Cb分量和Cr分量需保持一致。
is_abs_qp	表示ROI区域的QP模式，指定当前的ROI区域采用绝对QP方式或是相对QP。 - HI_FALSE：相对QP - HI_TURE：绝对QP
qp	QP值，当QP模式为HI_FALSE时，qp为QP偏移，qp范围[-51,51]，当QP模式为HI_TRUE时，qp为宏块QP值，qp范围[0,51]。
rect	指定当前的ROI区域的位置坐标和区域的大小。 ROI区域的起始点坐标必须在图像范围内，且必须16对齐；ROI区域必须在图像范围内，ROI区域的长宽必须是16对齐并且大于0。即，x、y、width、height必须是16对齐。

1.9.20.14.84 hi_venc_ref_param

说明

定义H.264/H.265编码的高级跳帧参考参数。

定义

```
typedef struct {  
    hi_u32 base;  
    hi_u32 enhance;  
    hi_bool pred_en;  
} hi_venc_ref_param;
```

成员

成员名称	描述
base	base层的周期。 取值范围：（ 0， +∞ ）。
enhance	enhance层的周期。 取值范围： [0, 255] 。

成员名称	描述
pred_en	代表base层的帧是否被base层其他帧用作参考。当pred_en设置为HI_FALSE时，等同于base设置为无限大，base 层的所有帧都参考IDR 帧。

1.9.20.14.85 hi_venc_h264_vui

说明

定义H.264协议编码通道Vui结构体。

定义

```
typedef struct {  
    hi_venc_vui_aspect_ratio    vui_aspect_ratio;  
    hi_venc_vui_h264_time_info  vui_time_info;  
    hi_venc_vui_video_signal    vui_video_signal;  
    hi_venc_vui_bitstream_restric    vui_bitstream_restric;  
} hi_venc_h264_vui;
```

成员

成员名称	描述
vui_aspect_ratio	预留参数，具体含义请参见H.264协议。
vui_time_info	Vui字段中时钟计数相关参数。
vui_video_signal	Vui字段中视频信号相关参数。
vui_bitstream_restric	预留参数，具体含义请参见H.264协议。

1.9.20.14.86 hi_venc_h265_vui

说明

定义H.265协议编码通道Vui结构体。

定义

```
typedef struct {  
    hi_venc_vui_aspect_ratio    vui_aspect_ratio;  
    hi_venc_vui_h265_time_info  vui_time_info;  
    hi_venc_vui_video_signal    vui_video_signal;  
    hi_venc_vui_bitstream_restric    vui_bitstream_restric;  
} hi_venc_h265_vui;
```

成员

成员名称	描述
vui_aspect_ratio	预留参数，具体含义请参见H.265协议。
vui_time_info	Vui字段中时钟计数相关参数。
vui_video_signal	Vui字段中视频信号相关参数。
vui_bitstream_restric	预留参数，具体含义请参见H.265协议。

1.9.20.14.87 hi_venc_vui_aspect_ratio

说明

定义H.264/H.265协议编码通道Vui中AspectRatio信息的结构体。预留结构体，暂不支持用户配置。

定义

```
typedef struct {  
    hi_u8 aspect_ratio_info_present_flag;  
    hi_u8 aspect_ratio_idc;  
    hi_u8 overscan_info_present_flag;  
    hi_u8 overscan_appropriate_flag;  
    hi_u16 sar_width;  
    hi_u16 sar_height;  
} hi_venc_vui_aspect_ratio;
```

成员

成员名称	描述
aspect_ratio_info_present_flag	具体含义请参见H.264/H.265协议，系统默认值为0。
aspect_ratio_idc	具体含义请参见H.264/H.265协议，系统默认值为1。
overscan_info_present_flag	具体含义请参见H.264/H.265协议，系统默认值为0。
overscan_appropriate_flag	具体含义请参见H.264/H.265协议，系统默认值为0。
sar_width	具体含义请参见H.264/H.265协议，系统默认值为1。
sar_height	具体含义请参见H.264/H.265协议，系统默认值为1。

1.9.20.14.88 hi_venc_vui_h264_time_info

说明

定义H.264协议编码通道Vui中时钟计数信息的结构体。

定义

```
typedef struct {  
    hi_u8 timing_info_present_flag;  
    hi_u8 fixed_frame_rate_flag;  
    hi_u32 num_units_in_tick;  
    hi_u32 time_scale;  
} hi_venc_vui_h264_time_info;
```

成员

成员名称	描述
timing_info_present_flag	Vui字段中时钟计数信息是否生效的标志位，系统默认值为0。取值范围：0或1。表格中其它参数只有在该标志为1时才生效。
fixed_frame_rate_flag	具体含义请参见H.264协议，系统默认值为1。取值范围：0或1。
num_units_in_tick	时钟计数器加1所经过的时间单元数，系统默认值为1。取值范围：大于0。
time_scale	一秒钟时长所经过的时间单元数，系统默认值为60。取值范围：大于0。

注意事项

时钟计数用于标记视频帧信号，在本H.264编码器中，视频显示帧率可由根据以下公式计算： $display\_rate = time\_scale / num\_units\_in\_tick / 2$ 。

由于显示帧率只与time_scale和num_units_in_tick的比值有关，建议用户设置时保持num_units_in_tick值为1，通过time_scale参数调整显示帧率，此时显示帧率 = $time\_scale / 2$ 。

显示帧率的最终决定权在于播放器，播放器也可以选择不参考TimeInfo信息。

1.9.20.14.89 hi_venc_vui_h265_time_info

说明

定义H.265协议编码通道Vui中时钟计数信息的结构体。

定义

```
typedef struct {  
    hi_u32 timing_info_present_flag;  
    hi_u32 num_units_in_tick;  
    hi_u32 time_scale;
```

```
hi_u32 num_ticks_poc_diff_one_minus1;  
} hi_venc_vui_h265_time_info;
```

成员

成员名称	描述
timing_info_present_flag	Vui字段中时钟计数信息是否生效的标志位，系统默认值为0。取值范围：0或1。表格中其它参数只有在该标志为1时才生效。
num_units_in_tick	时钟计数器加1所经过的时间单元数，系统默认值为1。取值范围：大于0。
time_scale	一秒钟时长所经过的时间单元数，系统默认值为60。取值范围：大于0。
num_ticks_poc_diff_one_minus1	具体含义请参见H.265协议，系统默认值为1。取值范围： $[0, 2^{32}-2]$ 。

注意事项

时钟计数用于标记视频帧信号，在本H.265编码器中，视频显示帧率可由以下公式计算： $display\_rate = time\_scale / num\_units\_in\_tick$

由于显示帧率只与time_scale和num_units_in_tick的比值有关，建议用户设置时保持num_units_in_tick值为1，通过time_scale参数调整显示帧率，此时显示帧率 = time_scale。

显示帧率的最终决定权在于播放器，播放器也可以选择不参考TimeInfo信息。

1.9.20.14.90 hi_venc_vui_video_signal

说明

定义H.264/H.265协议编码通道Vui中视频信号信息的结构体。

定义

```
typedef struct {  
    hi_u8 video_signal_type_present_flag;  
    hi_u8 video_format;  
    hi_u8 video_full_range_flag;  
    hi_u8 colour_description_present_flag;  
    hi_u8 colour_primaries;  
    hi_u8 transfer_characteristics;  
    hi_u8 matrix_coefficients;  
} hi_venc_vui_video_signal;
```

成员

成员名称	描述
video_signal_type_present_flag	Vui字段中视频信号信息是否生效的标志位，系统默认值为1。取值范围：0或1。表格中其它参数只有在该标志为1时才生效。
video_format	具体含义请参见H.264/H.265协议，系统默认值为5。取值范围：H.264:[0, 7]，H.265:[0, 5]。
video_full_range_flag	具体含义请参见H.264/H.265协议，取值范围：0或1。置0表示像素值域受限，亮度取值范围是16～235，色度的取值范围是16～240。 在Atlas 推理系列产品上，对于H.264、H.265码流，当前VENC编码时video_full_range_flag默认值为0（表示limited_range）。 在Atlas 200/500 A2推理产品上，对于H.265码流，当前VENC编码时video_full_range_flag默认值为0（表示limited_range）；对于H.264码流，当前VENC编码时video_full_range_flag默认值为1（表示full_range）。
colour_description_present_flag	具体含义请参见H.264/H.265协议，系统默认值为1，用户无需配置，系统内部根据当前图像实际信息填写。
colour_primaries	具体含义请参见H.264/H.265协议，系统默认值为1，用户无需配置，系统内部根据当前图像实际信息填写。
transfer_characteristics	具体含义请参见H.264/H.265协议，系统默认值为1，用户无需配置，系统内部根据当前图像实际信息填写。
matrix_coefficients	具体含义请参见H.264/H.265协议，系统默认值为1，用户无需配置，系统内部根据当前图像实际信息填写。

1.9.20.14.91 hi_venc_vui_bitstream_restric

说明

定义H.264/H.265协议编码通道Vui中Bitstream_Restriction信息的结构体。预留结构体，暂不支持用户配置。

定义

```
typedef struct {  
    hi_u8 bitstream_restriction_flag;  
} hi_venc_vui_bitstream_restric;
```

成员

成员名称	描述
bitstream_restriction_flag	具体含义请参见H.265协议，系统默认值为0。

1.9.20.14.92 hi_venc_cu_prediction

说明

定义CU模式选择的倾向性配置结构体。

定义

```
typedef struct {  
    hi_op_mode    pred_mode;  
    hi_u32    intra32_cost;  
    hi_u32    intra16_cost;  
    hi_u32    intra8_cost;  
    hi_u32    intra4_cost;  
    hi_u32    inter64_cost;  
    hi_u32    inter32_cost;  
    hi_u32    inter16_cost;  
    hi_u32    inter8_cost;  
} hi_venc_cu_prediction;
```

成员

成员名称	描述
pred_mode	倾向性选择模式，支持自动模式、手动模式两种。
intra32_cost	Intra32倾向性调节，对应帧内预测32*32像素块尺寸模式，该值增大表示选择该模式的倾向性越小。 取值范围：[0, 15]。 默认值：8。 H.264该值无效。
intra16_cost	Intra16倾向性调节，对应帧内预测16*16像素块尺寸模式，该值增大表示选择该模式的倾向性越小。 取值范围：[0, 15]。 默认值：8。
intra8_cost	Intra8倾向性调节，对应帧内预测8*8像素块尺寸模式，该值增大表示选择该模式的倾向性越小。 取值范围：[0, 15]。 默认值：8。
intra4_cost	Intra4倾向性调节，对应帧内预测4*4像素块尺寸模式，该值增大表示选择该模式的倾向性越小。 取值范围：[0, 15]。 默认值：8。

成员名称	描述
inter64_cost	Inter64倾向性调节，对应帧间预测64*64像素块尺寸模式，该值增大表示选择该模式的倾向性越小。 取值范围：[0, 15]。 默认值：8。 H.264该值无效。
inter32_cost	Inter32倾向性调节，对应帧间预测32*32像素块尺寸模式，该值增大表示选择该模式的倾向性越小。 取值范围：[0, 15]。 默认值：8。 H.264该值无效。
inter16_cost	Inter16倾向性调节，对应帧间预测16*16像素块尺寸模式，该值增大表示选择该模式的倾向性越小。 取值范围：[0, 15]。 默认值：8。
inter8_cost	Inter8倾向性调节，对应帧间预测8*8像素块尺寸模式，该值增大表示选择该模式的倾向性越小。 取值范围：[0, 15]。 默认值：8。

1.9.20.14.93 hi_venc_intra_refresh

说明

P帧刷Islice控制参数。

定义

```
typedef struct {  
    hi_bool refresh_enable;  
    hi_venc_intra_refresh_mode intra_refresh_mode;  
    hi_u32 refresh_num;  
    hi_u32 req_i_qp;  
} hi_venc_intra_refresh;
```

成员

成员名称	描述
refresh_enable	是否使能刷Islice功能。 0：不使能，默认值； 1：使能。
intra_refresh_mode	I宏块刷新模式，分为按行刷新和按列刷新。默认按行刷新。 Atlas 推理系列产品，仅支持按行刷新。

成员名称	描述
refresh_num	每次I宏块刷新行数或者列数，可以通过这个变量控制刷新的速度及码流的平稳程度。刷新行数或者列数越多，刷新的速度越快，但是码流平稳度越差；刷新的行数或者列数越少，刷新的速度越慢，但是码流平稳度越好。 需保证设置的refresh_num可以在一个GOP内完成Islice刷新，注意高级跳帧参考时只会在base层中的P帧（可被base层中其他帧的参考）进行刷新。refresh_num需满足表1-87中的计算公式。
req_i_qp	I帧QP值。 hi_mpi_venc_set_intra_refresh接口与hi_mpi_venc_request_idr接口配合使用时，该值用于控制插入的IDR帧的质量，该值越小，质量越好，插入的IDR帧大小越大。 取值范围：[0, 51]，默认值51。

参考信息

表 1-87 refresh_num 参数值计算公式

-	计算公式	备注
H.264，refresh_num 默认值为： $(pic\_height + lcu\_size - 1) >> 6$	- 行： $refresh\_num * (max\_refresh\_frame\_in\_gop - 1) >= (pic\_height + lcu\_size - 1) / lcu\_size$ - 列： $refresh\_num * (max\_refresh\_frame\_in\_gop - 1) >= (pic\_width + lcu\_size - 1) / lcu\_size$ 计算公式中的/表示向下取整。	- GOP内P帧刷Islice最大帧数 max_refresh_frame_in_gop参数值取值说明： - 无高级跳帧参考时： $max\_refresh\_frame\_in\_gop = gop$; - 有高级跳帧参考时： $max\_refresh\_frame\_in\_gop = (gop + (base * (enhance + 1) - 1)) / (base * (enhance + 1))$。 - 编码单元尺寸lcu_size参数值取值说明： - H.264时，该参数值固定为16； - H265时： - Atlas 推理系列产品，该参数值固定为64 - Atlas 200/500 A2推理产品，该参数值固定为32
H.265，refresh_num 默认值为： $((pic\_height + lcu\_size - 1) / lcu\_size) >> 2$		

1.9.20.14.94 hi_venc_intra_refresh_mode

说明

P帧刷Islice模式定义。

定义

```
typedef enum{
    INTRA_REFRESH_ROW = 0,           /* 按行刷新 */
    INTRA_REFRESH_COLUMN,           /* 按列刷新 */
    INTRA_REFRESH_BUTT
} hi_venc_intra_refresh_mode;
```

1.9.20.14.95 hi_venc_slice_split

说明

P帧刷Islice控制参数。

定义

```
typedef struct {
    hi_bool    split_enable;
    hi_u32     split_mode;
    hi_u32     split_size;
    hi_bool    slice_output_en;
    hi_u32     reserved[2];
} hi_venc_slice_split;
```

成员

成员名称	描述
split_enable	Slice分割是否使能。 0：不使能，默认值； 1：使能 。
split_mode	Slice分割模式。 当前仅支持设置为1，表示按宏块行或者LCU行分割。
split_size	- H.264表示每个slice占的宏块行数 最大值：(图像高+15)/16 最小值：(最大值+15)/16 - H.265表示每个slice占的LCU行数 最大值：(图像高+LCU -1)/LCU 最小值：(最大值+15)/16
slice_output_e n	当前仅支持设置为false。
reserved[2]	预留参数，当前仅支持设置为0。

注意事项

- split_mode=1时，对H265协议需要关注当前芯片的LCU大小。
- Atlas 推理系列产品，该参数值固定为64
 - Atlas 200/500 A2推理产品，该参数值固定为32

1.9.20.14.96 hi_venc_frame_type

说明

定义编码帧类型枚举。

定义

```
typedef enum {  
    VENC_FRAME_TYPE_NONE = 1,  
    VENC_FRAME_TYPE_IDR,  
    VENC_FRAME_TYPE_PSKIP,  
    VENC_FRAME_TYPE_BUTT  
} hi_venc_frame_type;
```

成员

成员名称	描述
HI_VENC_FRAME_TYPE_NONE	当前帧编码类型不指定，实际帧类型由驱动内部决定。
HI_VENC_FRAME_TYPE_IDR	指定当前帧编为IDR帧。
HI_VENC_FRAME_TYPE_P_SKIP	指定当前帧编为p_skip帧。

1.9.20.14.97 hi_user_rc_info

说明

定义用户码控信息结构体。

定义

```
typedef struct {  
    hi_bool qp_map_valid;  
    hi_bool skip_weight_valid;  
    hi_u32 blk_start_qp;  
    hi_u64 qp_map_phy_addr;  
    hi_u64 skip_weight_phy_addr;  
    hi_venc_frame_type frame_type;  
} hi_user_rc_info;
```

成员

成员名称	描述
qp_map_valid	QPMAP模式中,Qp表是否生效。
skip_weight_valid	QPMAP模式中,SkipWeight表是否生效, 预留参数, 暂不支持。
blk_start_qp	QPMAP模式中, 第一个16*16块的Qp值。 取值范围: [0, 51]。
qp_map_phy_addr	QPMAP模式中, Qp表的虚拟地址。
skip_weight_phy_addr	QPMAP模式中, SkipWeight表的虚拟地址, 预留参数, 暂不支持。
frame_type	指定当前帧编码帧类型, 预留参数, 暂不支持。

1.9.20.14.98 hi_user_roi_map

说明

定义用户ROI信息结构体, 预留结构体, 暂不支持。

定义

```
typedef struct {  
    hi_bool valid;  
    hi_u64 phy_addr;  
} hi_user_roi_map;
```

成员

成员名称	描述
valid	该图的ROIMap表是否有效。
phy_addr	该图的ROIMap表的虚拟地址。

1.9.20.14.99 hi_user_frame_info

说明

定义用户发送图像信息结构体。

定义

```
typedef struct {  
    hi_video_frame_info user_frame;  
    hi_user_rc_info user_rc_info;  
    hi_user_roi_map user_roi_map;  
} hi_user_frame_info;
```

成员

成员名称	描述
user_frame	用户发送图像帧信息结构体。具体描述请参见系统控制章节。
user_rc_info	用户码控信息结构体。
user_roi_map	用户ROIMap信息结构体，预留参数，暂不支持。

1.9.20.14.100 hi_venc_streamcopy_param

说明

定义用户帧拷贝参数信息结构体。

定义

```
typedef struct {  
    hi_bool copy_en;  
    hi_u32 copy_interval;  
} hi_venc_streamcopy_param;
```

成员

成员名称	描述
copy_en	用户设置是否使能帧拷贝。 取值范围如下： - HI_TRUE：使能 - HI_FALSE：不使能
copy_interval	帧拷贝使能时的拷贝间隔，取值范围[0, 10]。

1.9.20.15 PNGD 图像解码

1.9.20.15.1 hi_pngd_chn

说明

描述PNG图片解码通道。

定义

```
typedef hi_s32 hi_pngd_chn;
```

1.9.20.15.2 hi_pngd_chn_attr

说明

定义PNGD解码通道属性结构体。

定义

```
typedef struct {  
    hi_u32 stream_que_cnt;  
    hi_u64 reserved[4];  
} hi_pngd_chn_attr;
```

成员

成员名称	描述
stream_que_cnt	PNGD图片缓存队列大小。预留参数。
reserved	预留参数，为保证后续版本兼容性，请务必使用memset结构体方式进行清零初始化，在代码中必须避免显式对reserved字段进行访问。

1.9.21 返回码列表

1.9.21.1 公共返回码

错误代码	宏定义	描述
0xA0028001	-	无效的Device ID。
0xA0028002	-	无效的channel ID。
0xA0028003	HI_ERR_SYS_ILLEGAL_PARAM	参数设置无效。
0xA0028004	HI_ERR_SYS_EXIST	通道或资源已存在。
0xA0028005	HI_ERR_SYS_UNEXIST	通道或资源不存在。
0xA0028006	HI_ERR_SYS_NULL_PTR	空指针错误。
0xA0028007	-	使能系统、Device或通道前未配置对应的参数。
0xA0028008	HI_ERR_SYS_NOT_SUPPORT	不支持的功能。
0xA0028009	HI_ERR_SYS_NOT_PERM	操作不允许。
0xA002800C	HI_ERR_SYS_NO_MEM	分配内存失败，如系统内存不足。
0xA002800D	-	分配缓存失败，如申请的数据缓冲区太大。

错误代码	宏定义	描述
0xA002800E	-	缓冲区中无数据。
0xA002800F	-	缓冲区中数据满。
0xA0028010	HI_ERR_SYS_NOT_READY	系统控制属性未配置。
0xA0028011	-	地址错误。
0xA0028012	HI_ERR_SYS_BUSY	系统忙。
0xA0028013	-	缓存小于实际需要的大小。
0xA0028014	-	硬件或软件处理超时。
0xA0028015	HI_ERR_SYS_ERR	内部系统错误。
0xA002803F	-	最大的返回码，该模块的错误码必须小于该值。
0xA0018003	-	VB参数设置无效。
0xA0018005	-	不存在的内存池或者内存块。
0xA0018006	-	空指针错误。
0xA0018009	-	操作不允许。
0xA001800C	-	分配内存失败，如系统内存不足。
0xA001800D	-	没有满足条件的内存池或者内存块。
0xA0018010	-	系统控制属性未配置。
0xA0018012	-	VB系统忙。
0xA0018013	-	内存块小于实际需要的大小。

1.9.21.2 音频相关返回码

表 1-88 音频基础属性API 错误码

错误代码	宏定义	描述
0xa0148003	HI_ERR_AIO_ILLEGAL_PARAM	音频基础属性参数设置无效。
0xa0148006	HI_ERR_AIO_NULL_PTR	输入参数空指针错误。
0xa0148009	HI_ERR_AIO_NOT_PERM	操作不允许。

错误代码	宏定义	描述
0xa0148010	HI_ERR_AIO_NOT_READY	音频基础系统未初始化。
0xa0148012	HI_ERR_AIO_BUSY	音频基础系统忙。
0xa014801d	HI_ERR_AIO_REGISTER_ERR	注册失败。

表 1-89 音频输入错误码

错误代码	宏定义	描述
0xa0158001	HI_ERR_AI_INVALID_DEV_ID	音频输入设备号无效。
0xa0158002	HI_ERR_AI_INVALID_CHN_ID	音频输入通道号无效。
0xa0158003	HI_ERR_AI_ILLEGAL_PARAMETER	音频输入参数设置无效。
0xa0158006	HI_ERR_AI_NULL_PTR	输入参数空指针错误。
0xa0158007	HI_ERR_AI_NOT_CFG	音频输入设备属性未设置。
0xa0158009	HI_ERR_AI_NOT_PERM	操作不允许。
0xa015800c	HI_ERR_AI_NO_MEM	分配内存失败。
0xa015800d	HI_ERR_AI_NO_BUF	音频输入缓存不足。
0xa015800e	HI_ERR_AI_BUF_EMPTY	音频输入缓存为空。
0xa015800f	HI_ERR_AI_BUF_FULL	音频输入缓存为满。
0xa0158010	HI_ERR_AI_NOT_READY	音频输入系统未初始化。
0xa0158012	HI_ERR_AI_BUSY	音频输入系统忙。
0xa0158017	HI_ERR_AI_NOT_ENABLED	音频输入设备或通道没有使能。

表 1-90 音频输出错误码

错误代码	宏定义	描述
0xa0168001	HI_ERR_AO_INVALID_DEV_ID	音频输出设备号无效。
0xa0168002	HI_ERR_AO_INVALID_CHN_ID	音频输出通道号无效。

错误代码	宏定义	描述
0xa0168003	HI_ERR_AO_ILLEGAL_PA RAM	音频输出参数设置无效。
0xa0168006	HI_ERR_AO_NULL_PTR	输出空指针错误。
0xa0168007	HI_ERR_AO_NOT_CFG	音频输出设备属性未设置。
0xa0168009	HI_ERR_AO_NOT_PERM	操作不允许。
0xa016800c	HI_ERR_AO_NO_MEM	统内存不足。
0xa016800d	HI_ERR_AO_NO_BUF	音频输出缓存不足。
0xa016800e	HI_ERR_AO_BUF_EMPTY	音频输出缓存为空。
0xa016800f	HI_ERR_AO_BUF_FULL	音频输出缓存为满。
0xa0168010	HI_ERR_AO_NOT_READY	音频输出系统未初始化。
0xa0168012	HI_ERR_AO_BUSY	音频输出系统忙。
0xa0168017	HI_ERR_AO_NOT_ENABL ED	音频输出设备或通道没使能。

表 1-91 音频编码错误码

错误代码	宏定义	描述
0xa0178001	HI_ERR_AENC_INVALID_ DEV_ID	音频设备号无效。
0xa0178002	HI_ERR_AENC_INVALID_ CHN_ID	音频编码通道号无效。
0xa0178003	HI_ERR_AENC_ILLEGAL_P ARAM	音频编码参数设置无效。
0xa0178004	HI_ERR_AENC_EXIST	音频编码通道已经创建。
0xa0178005	HI_ERR_AENC_UNEXIST	音频编码通道未创建。
0xa0178006	HI_ERR_AENC_NULL_PTR	输入参数空指针错误。
0xa0178007	HI_ERR_AENC_NOT_CFG	编码通道未配置。
0xa0178008	HI_ERR_AENC_NOT_SUP PORT	操作不被支持。
0xa0178009	HI_ERR_AENC_NOT_PER M	操作不允许。
0xa017800c	HI_ERR_AENC_NO_MEM	系统内存不足。
0xa017800d	HI_ERR_AENC_NO_BUF	编码通道缓存分配失败。

错误代码	宏定义	描述
0xa017800e	HI_ERR_AENC_BUF_EMPTY	编码通道缓存空。
0xa017800f	HI_ERR_AENC_BUF_FULL	编码通道缓存满。
0xa0178010	HI_ERR_AENC_NOT_READY	系统没有初始化。
0xa0178040	HI_ERR_AENC_ENCODER_ERR	音频编码数据错误。

表 1-92 音频解码错误码

错误代码	宏定义	描述
0xa0188001	HI_ERR_ADEC_INVALID_DEV_ID	音频解码设备号无效。
0xa0188002	HI_ERR_ADEC_INVALID_CHN_ID	音频解码通道号无效。
0xa0188003	HI_ERR_ADEC_ILLEGAL_PARAMS	音频解码参数设置无效。
0xa0188004	HI_ERR_ADEC_EXIST	音频解码通道已经创建。
0xa0188005	HI_ERR_ADEC_UNEXIST	音频解码通道未创建。
0xa0188006	HI_ERR_ADEC_NULL_PTR	输入参数空指针错误。
0xa0188007	HI_ERR_ADEC_NOT_CFG	解码通道属性未配置。
0xa0188008	HI_ERR_ADEC_NOT_SUPPORT	操作不被支持。
0xa0188009	HI_ERR_ADEC_NOT_PERM	操作不允许。
0xa018800c	HI_ERR_ADEC_NO_MEM	系统内存不足。
0xa018800d	HI_ERR_ADEC_NO_BUF	解码通道缓存分配失败。
0xa018800e	HI_ERR_ADEC_BUF_EMPTY	解码通道缓存空。
0xa018800f	HI_ERR_ADEC_BUF_FULL	解码通道缓存满。
0xa0188010	HI_ERR_ADEC_NOT_READY	系统没有初始化。
0xa0188040	HI_ERR_ADEC_DECODER_ERR	音频解码数据错误。
0xa0188041	HI_ERR_ADEC_BUF_LACK	解码输入缓存空间不够。

1.9.21.3 ISP 系统控制返回码

表 1-93 ISP API 错误码

错误代码	宏定义	描述
0xA01C8003	HI_ERR_ISP_ILLEGAL_PARAM	输入参数无效。
0xA01C8006	HI_ERR_ISP_NULL_PTR	空指针错误。
0xA01C8008	HI_ERR_ISP_NOT_SUPPORT	当前ISP不支持。
0xA01C800C	HI_ERR_ISP_NO_MEM	内存不足。
0xA01C8040	HI_ERR_ISP_NOT_INIT	ISP没有初始化。
0xA01C8041	HI_ERR_ISP_MEM_NOT_INIT	外部寄存器没有初始化。
0xA01C8042	HI_ERR_ISP_ATTR_NOT_CFG	属性未配置。
0xA01C8043	HI_ERR_ISP_SNS_UNREGISTER	Sensor未注册。
0xA01C8044	HI_ERR_ISP_INVALID_ADDR	无效地址。
0xA01C8046	HI_ERR_ISP_NO_INT	ISP无中断。

1.9.21.4 VI 视频输入返回码

错误代码	宏定义	描述
0xA0108001	HI_ERR_VI_INVALID_DEV_ID	视频输入设备号无效。
0xA0108002	HI_ERR_VI_INVALID_CHN_ID	视频输入通道号无效。
0xA0108003	HI_ERR_VI_ILLEGAL_PARAM	视频输入参数设置无效。
0xA0108004	HI_ERR_VI_PIPE_EXIST	PIPE已存在。
0xA0108005	HI_ERR_VI_PIPE_UNEXIST	PIPE不存在。
0xA0108006	HI_ERR_VI_NULL_PTR	输入参数空指针错误。
0xA0108007	HI_ERR_VI_NOT_CFG	视频设备或通道属性未配置。
0xA0108008	HI_ERR_VI_NOT_SUPPORT	操作不支持。
0xA0108009	HI_ERR_VI_NOT_PERM	操作不允许。
0xA010800A	HI_ERR_VI_INVALID_PIPE_ID	PIPE号无效。
0xA010800C	HI_ERR_VI_NO_MEM	分配内存失败。
0xA010800E	HI_ERR_VI_BUF_EMPTY	视频输入缓存为空。
0xA010800F	HI_ERR_VI_BUF_FULL	视频输入缓存为满。

错误代码	宏定义	描述
0xA0108010	HI_ERR_VI_SYS_NOT_READY	视频输入系统未初始化。
0xA0108012	HI_ERR_VI_BUSY	视频输入系统忙。
0xA0108040	HI_ERR_VI_NOT_ENABLE	视频输入设备或通道未启用。
0xA0108041	HI_ERR_VI_NOT_DISABLE	视频输入设备或通道未禁用。
0xA0108043	HI_ERR_VI_TIME_OUT	视频配置属性超时。
0xA0108047	HI_ERR_VI_NOT_BINDED	视频通道未绑定。
0xA0108048	HI_ERR_VI_BINDED	视频通道已绑定。

1.9.21.5 区域管理返回码

错误代码	宏定义	描述
0xA0038001	HI_ERR_RGN_INVALID_DEV_ID	设备ID超出合法范围。
0xA0038002	HI_ERR_RGN_INVALID_CHN_ID	通道组号错误或无效区域句柄。
0xA0038003	HI_ERR_RGN_ILLEGAL_PARAM	参数超出合法范围。
0xA0038004	HI_ERR_RGN_EXIST	重复创建已存在的设备、通道或资源。
0xA0038005	HI_ERR_RGN_UNEXIST	试图使用或者销毁不存在的设备、通道或者资源。
0xA0038006	HI_ERR_RGN_NULL_PTR	函数参数中有空指针。
0xA0038007	HI_ERR_RGN_NOT_CFG	模块没有配置。
0xA0038008	HI_ERR_RGN_NOT_SUPPORT	不支持的参数或者功能。
0xA0038009	HI_ERR_RGN_NOT_PERM	该操作不允许，如试图修改静态配置参数。
0xA003800C	HI_ERR_RGN_NO_MEM	分配内存失败，如系统内存不足。
0xA003800D	HI_ERR_RGN_NO_BUF	分配缓存失败，如申请的数据缓冲区太大。
0xA003800E	HI_ERR_RGN_BUF_EMPTY	缓冲区中无数据。
0xA003800F	HI_ERR_RGN_BUF_FULL	缓冲区中数据满。
0xA0038010	HI_ERR_RGN_NOT_READY	系统没有初始化或没有加载相应模块。

错误代码	宏定义	描述
0xA0038011	HI_ERR_RGN_BAD_ADDR	地址非法。
0xA0038012	HI_ERR_RGN_BUSY	系统忙。

1.9.21.6 VPSS 视频处理子系统返回码

错误代码	宏定义	描述
0xA03E8001	HI_ERR_VPSS_INVALID_DEV_ID	VPSS组ID无效
0xA03E8002	HI_ERR_VPSS_INVALID_CHN_ID	VPSS通道号无效
0xA03E8003	HI_ERR_VPSS_ILLEGAL_PARAM	VPSS参数设置无效
0xA03E8004	HI_ERR_VPSS_EXIST	VPSS Group已创建
0xA03E8005	HI_ERR_VPSS_UNEXIST	VPSS Group未创建
0xA03E8006	HI_ERR_VPSS_NULL_PTR	输入参数空指针错误
0xA03E8008	HI_ERR_VPSS_NOT_SUPPORT	操作不支持
0xA03E8009	HI_ERR_VPSS_NOT_PERM	操作不允许
0xA03E800C	HI_ERR_VPSS_NO_MEM	分配内存失败
0xA03E800D	HI_ERR_VPSS_NO_BUF	分配BUF池失败
0xA03E800E	HI_ERR_VPSS_BUF_EMPTY	图像队列为空
0xA03E8010	HI_ERR_VPSS_NOT_READY	VPSS系统未初始化
0xA03E8012	HI_ERR_VPSS_BUSY	VPSS系统忙
0xA03E8013	HI_ERR_VPSS_SIZE_NOT_ENOUGH	VB块大小不够

1.9.21.7 VO 视频输出返回码

表 1-94 视频输出错误码

错误码	定义	描述
0xa00f8001	HI_ERR_VO_INVALID_DEV_ID	设备ID超出合法范围
0xa00f8003	HI_ERR_VO_INVALID_CHN_ID	通道ID超出合法范围
0xa00f8004	HI_ERR_VO_INVALID_LAYER_ID	层ID超出合法范围

错误码	定义	描述
0xa00f8007	HI_ERR_VO_ILLEGAL_PARAMETER	参数超出合法范围
0xa00f800a	HI_ERR_VO_NULL_PTR	函数参数中有空指针
0xa00f800b	HI_ERR_VO_NOT_CFG	未配置
0xa00f800c	HI_ERR_VO_NOT_SUPPORT	不支持的操作
0xa00f800d	HI_ERR_VO_NOT_PERM	操作不允许
0xa00f8010	HI_ERR_VO_NOT_ENABLE	未使能
0xa00f8011	HI_ERR_VO_NOT_DISABLE	未禁用
0xa00f8014	HI_ERR_VO_NO_MEM	分配内存失败
0xa00f8018	HI_ERR_VO_NOT_READY	系统未初始化
0xa00f8020	HI_ERR_VO_TIMEOUT	等待超时
0xa00f8022	HI_ERR_VO_BUSY	资源忙
0xa00f8024	HI_ERR_VO_NOT_BINDED	未被绑定
0xa00f8025	HI_ERR_VO_BINDED	已被绑

1.9.21.8 TDE 图形绘制返回码

表 1-95 图像处理错误码

错误码	定义	描述
0xA0648001	HI_ERR_TDE_DEV_NOT_OPEN	TDE设备未打开
0xA0648002	HI_ERR_TDE_DEV_OPEN_FAILED	TDE设备打开失败
0xA0648003	HI_ERR_TDE_NULL_PTR	输入参数包含空指针
0xA0648004	HI_ERR_TDE_NO_MEM	内存分配失败
0xA0648005	HI_ERR_TDE_INVALID_HANDLE	非法任务句柄
0xA0648006	HI_ERR_TDE_INVALID_PARAMETER	无效参数设置

错误码	定义	描述
0xA0648007	HI_ERR_TDE_NOT_ALIGN ED	参数值没有对齐
0xA0648008	HI_ERR_TDE_MINIFICATI ON	缩小倍数过大
0xA0648009	HI_ERR_TDE_CLIP_AREA	操作区域与clip区域没有交集，显示不会有更新
0xA0648010	HI_ERR_TDE_JOB_TIMEO UT	阻塞任务等待超时
0xA0648011	HI_ERR_TDE_UNSUPPOR TED_OPERATION	不支持的操作
0xA0648012	HI_ERR_TDE_QUERY_TIM EOUT	指定的任务超时未完成
0xA0648013	HI_ERR_TDE_INTERRUPT	阻塞任务被打断

1.9.21.9 HDMI 外设返回码

表 1-96 HDMI 错误码

错误码	定义	描述
0xA0288001	HI_ERR_HDMI_NOT_INIT	HDMI未初始化
0xA0288002	HI_ERR_HDMI_INVALID_ PARA	参数非法
0xA0288003	HI_ERR_HDMI_NUL_PTR	空指针
0xA0288004	HI_ERR_HDMI_DEV_NOT _OPEN	HDMI未打开
0xA0288005	HI_ERR_HDMI_DEV_NOT _CONNECT	HDMI设备未连接
0xA0288006	HI_ERR_HDMI_READ_SIN K_FAILED	HDMI读取Sink 端失败
0xA0288007	HI_ERR_HDMI_INIT_ALRE ADY	HDMI已经初始化
0xA0288008	HI_ERR_HDMI_CALLBAC K_ALREADY	HDMI回调已注册
0xA0288009	HI_ERR_HDMI_INVALID_ CALLBACK	回调函数无效
0xA028800A	HI_ERR_HDMI_FEATURE_ NO_USPPORT	功能不支持

错误码	定义	描述
0xA028800B	HI_ERR_HDMI_BUS_BUSY	总线忙
0xA028800C	HI_ERR_HDMI_READ_EVENT_FAILED	读取事件失败
0xA028800D	HI_ERR_HDMI_NOT_START	HDMI未工作
0xA028800E	HI_ERR_HDMI_READ_EDID_FAILED	EDID (Extended Display Identification Data) 读取失败
0xA028800F	HI_ERR_HDMI_INIT_FAILED	初始化失败
0xA0288010	HI_ERR_HDMI_CREATE_TASK_FAILED	创建任务失败
0xA0288011	HI_ERR_HDMI_MALLOC_FAILED	分配内存失败
0xA0288012	HI_ERR_HDMI_FREE_FAILED	释放内存失败
0xA0288013	HI_ERR_HDMI_PTHREAD_CREATE_FAILED	创建线程失败
0xA0288014	HI_ERR_HDMI_PTHREAD_JOIN_FAILED	加入线程失败
0xA0288015	HI_ERR_HDMI_STRATEGY_FAILED	工作策略失败
0xA0288016	HI_ERR_HDMI_SET_ATTR_FAILED	设置参数失败
0xA0288017	HI_ERR_HDMI_CALLBACK_NOT_REGISTER	回调未注册
0xA0288018	HI_ERR_HDMI_UNKNOWN_COMMAND	未知命令
0xA0288019	HI_ERR_HDMI_MUTEX_LOCK_FAILED	锁失败

1.9.21.10 VPC 图像处理返回码

错误代码	宏定义	描述
0xA0078001	-	无效的Device ID。
0xA0078002	HI_ERR_VPC_INVALID_CHN_ID	通道 ID 超出合法范围。

错误代码	宏定义	描述
0xA0078003	HI_ERR_VPC_ILLEGAL_PARAM	参数超出合法范围。
0xA0078004	HI_ERR_VPC_EXIST	试图创建已经存在的通道。
0xA0078005	HI_ERR_VPC_UNEXIST	通道未创建或已销毁。
0xA0078006	HI_ERR_VPC_NULL_PTR	函数参数中有空指针。
0xA0078007	HI_ERR_VPC_NOT_CFG	使用前未配置。
0xA0078008	HI_ERR_VPC_NOT_SUPPORT	不支持的参数或者功能。
0xA0078009	HI_ERR_VPC_NOT_PERM	该操作不允许。
0xA007800C	HI_ERR_VPC_NO_MEM	分配内存失败，如系统内存不足。
0xA007800D	HI_ERR_VPC_NO_BUF	分配缓存失败，如申请的数据缓冲区太大。
0xA007800E	HI_ERR_VPC_BUF_EMPTY	缓冲区中无数据。
0xA007800F	HI_ERR_VPC_BUF_FULL	缓冲区中数据满。
0xA0078010	HI_ERR_VPC_SYS_NOT_READY	系统没有初始化或者相关依赖的模块没有加载。
0xA0078011	HI_ERR_VPC_BAD_ADDR	内存地址错误。 媒体数据处理功能涉及的输入、输出内存需调用 hi_mpi_dvpp_malloc接口申请，或者内存大小配置错误，请检查代码逻辑。
0xA0078012	HI_ERR_VPC_BUSY	系统忙。
0xA0078013	-	缓存小于实际需要的大小。
0xA0078014	-	硬件或软件处理超时。
0xA0078015	HI_ERR_VPC_SYS_ERROR	内部系统错误。
0xA007803F	-	最大的返回码，该模块的错误码必须小于该值。

1.9.21.11 VDEC 视频解码/JPEGD 图片解码返回码

表1-97为VDEC视频解码和JPEGD图片解码共用的返回码。

表 1-97 共用返回码

错误代码	宏定义	描述
0xA0058001	-	无效的Device ID。
0xA0058002	HI_ERR_VDEC_INVALID_CHN_ID	通道 ID 超出合法范围。
0xA0058003	HI_ERR_VDEC_ILLEGAL_PARAM	参数超出合法范围。
0xA0058004	HI_ERR_VDEC_EXIST	试图创建已经存在的通道。
0xA0058005	HI_ERR_VDEC_UNEXIST	通道未创建或已销毁。
0xA0058006	HI_ERR_VDEC_NULL_PTR	函数参数中有空指针。
0xA0058007	HI_ERR_VDEC_NOT_CFG	使用前未配置。
0xA0058008	HI_ERR_VDEC_NOT_SUPPORT	不支持的参数或者功能。
0xA0058009	HI_ERR_VDEC_NOT_PERM	该操作不允许。
0xA005800C	HI_ERR_VDEC_NO_MEM	分配内存失败，如系统内存不足。
0xA005800D	HI_ERR_VDEC_NO_BUF	分配缓存失败，如申请的数据缓冲区太大。
0xA005800E	HI_ERR_VDEC_BUF_EMPTY	缓冲区中无数据。
0xA005800F	HI_ERR_VDEC_BUF_FULL	缓冲区中数据满。
0xA0058010	HI_ERR_VDEC_SYS_NOT_READY	系统没有初始化或者相关依赖的模块没有加载。
0xA0058011	HI_ERR_VDEC_BAD_ADDR	内存地址错误。 媒体数据处理功能涉及的输入、输出内存需调用 hi_mpi_dvpp_malloc接口申请，请检查代码逻辑。
0xA0058012	HI_ERR_VDEC_BUSY	系统忙。
0xA0058013	-	缓存小于实际需要的大小。
0xA0058014	-	硬件或软件处理超时。
0xA0058015	HI_ERR_VDEC_SYS_ERROR	内部系统错误。
0xA005803F	-	最大的返回码，该模块的错误码必须小于该值。

1.9.21.12 VENC 视频编码/JPEGE 图片编码返回码

表1-98为VENC视频编码和JPEGE图片编码共用的返回码。

表 1-98 共用返回码

错误代码	宏定义	描述
0xA0088001	-	无效的Device ID。
0xA0088002	HI_ERR_VENC_INVALID_CHN_ID	通道 ID 超出合法范围。
0xA0088003	HI_ERR_VENC_ILLEGAL_PARAM	参数超出合法范围。
0xA0088004	HI_ERR_VENC_EXIST	试图申请或者创建已经存在的设备、通道或者资源。
0xA0088005	HI_ERR_VENC_UNEXIST	试图使用或者销毁不存在的设备、通道或者资源。
0xA0088006	HI_ERR_VENC_NULL_PTR	函数参数中有空指针。
0xA0088007	HI_ERR_VENC_NOT_CFG	使用前未配置。
0xA0088008	HI_ERR_VENC_NOT_SUPPORT	不支持的参数或者功能。
0xA0088009	HI_ERR_VENC_NOT_PERM	该操作不允许，如试图修改静态配置参数。
0xA008800C	HI_ERR_VENC_NO_MEM	分配内存失败，如系统内存不足。
0xA008800D	HI_ERR_VENC_NO_BUF	分配缓存失败，如申请的数据缓冲区太大、频繁传入输入帧导致缓存区满等。
0xA008800E	HI_ERR_VENC_BUF_EMPTY	缓冲区中无数据。
0xA008800F	HI_ERR_VENC_BUF_FULL	缓冲区中数据满。
0xA0088010	HI_ERR_VENC_SYS_NOT_READY	系统没有初始化或没有加载相应模块。
0xA0088011	HI_ERR_VENC_BAD_ADDR	内存地址错误。 媒体数据处理功能涉及的输入、输出内存需调用 hi_mpi_dvpp_malloc 接口申请，请检查代码逻辑。
0xA0088012	HI_ERR_VENC_BUSY	系统忙。
0xA0088013	-	缓存小于实际需要的大小。

错误代码	宏定义	描述
0xA0088014	-	硬件或软件处理超时。
0xA0088015	HI_ERR_VENC_SYS_ERR OR	内部系统错误。
0xA008803F	-	最大的返回码，该模块的错误码必须小于该值。

1.9.21.13 PNGD 图像解码返回码

错误代码	宏定义	描述
0xA0408001	-	无效的Device ID。
0xA0408002	HI_ERR_PNGD_INVALID_CHN_ID	通道 ID 超出合法范围。
0xA0408003	HI_ERR_PNGD_ILLEGAL_PARAM	参数超出合法范围。
0xA0408004	HI_ERR_PNGD_EXIST	试图创建已经存在的通道。
0xA0408005	HI_ERR_PNGD_UNEXIST	通道未创建或已销毁。
0xA0408006	HI_ERR_PNGD_NULL_PTR	函数参数中有空指针。
0xA0408007	HI_ERR_PNGD_NOT_CFG	使用前未配置。
0xA0408008	HI_ERR_PNGD_NOT_SUPPORT	不支持的参数或者功能。
0xA0408009	HI_ERR_PNGD_NOT_PERM	该操作不允许。
0xA040800C	HI_ERR_PNGD_NO_MEM	分配内存失败，如系统内存不足。
0xA040800D	HI_ERR_PNGD_NO_BUF	分配缓存失败，如申请的数据缓冲区太大。
0xA040800E	HI_ERR_PNGD_BUF_EMPTY	缓冲区中无数据。
0xA040800F	HI_ERR_PNGD_BUF_FULL	缓冲区中数据满。
0xA0408010	HI_ERR_PNGD_SYS_NOT_READY	系统没有初始化或者相关依赖的模块没有加载。
0xA0408011	HI_ERR_PNGD_BAD_ADDR	内存地址错误。 媒体数据处理功能涉及的输入、输出内存需调用 hi_mpi_dvpp_malloc 接口申请，请检查代码逻辑。
0xA0408012	HI_ERR_PNGD_BUSY	系统忙。
0xA0408013	-	缓存小于实际需要的大小。

错误代码	宏定义	描述
0xA0408014	HI_ERR_PNGD_TIMEOUT	硬件或软件处理超时。
0xA0408015	HI_ERR_PNGD_SYS_ERROR	内部系统错误。
0xA040803F	-	最大的返回码，该模块的错误码必须小于该值。

1.10 Profiling 数据采集

1.10.1 Profiling AscendCL API (通过 Profiling AscendCL API 采集并落盘性能数据)

1.10.1.1 功能介绍

功能

该章节描述通过Profiling AscendCL API采集并落盘性能数据，实现将采集到的Profiling数据写入文件，再使用Profiling工具解析该文件（请参见《[性能调优工具使用指南](#)》下的“高级功能>数据解析与导出”），并展示性能分析数据。

包括以下两种接口调用方式：

- [aclprofInit](#)接口、[aclprofStart](#)接口、[aclprofStop](#)接口、[aclprofFinalize](#)接口配合使用，实现该方式的性能数据采集。该方式可获取AscendCL的接口性能数据、AI Core上算子的执行时间、AI Core性能指标数据等。目前这些接口为进程级控制，表示在进程内任意线程调用该接口，其它线程都会生效。
一个进程内，可以根据需求多次调用这些接口，基于不同的Profiling采集配置，采集数据。
- 调用[aclInit](#)接口，在AscendCL初始化阶段，通过*.json 文件传入要采集的Profiling数据。该方式可获取AscendCL的接口性能数据、AI Core上算子的执行时间、AI Core性能指标数据等。
一个进程内，只能调用一次[aclInit](#)接口，如果要修改Profiling采集配置，需修改*.json文件中的配置。详细使用说明请参见[aclInit](#)接口处的说明，不在本章节描述。

总体约束

不能与[1.10.3 Profiling AscendCL API for Subscription \(订阅算子信息的Profiling AscendCL API \)](#)的接口交叉调用：[aclprofInit](#)接口和[aclprofFinalize](#)接口之间不能调用[aclprofModelSubscribe](#)接口、[aclprofGet*](#)接口、[aclprofModelUnSubscribe](#)接口。

接口约束说明

- 调用接口要求：
 - [aclprofInit](#)接口必须在[aclInit](#)接口之后、模型加载之前调用。

如果已经通过[aclInit](#)接口配置了Profiling信息，则调用[aclprofInit](#)接口、[aclprofStart](#)接口、[aclprofStop](#)接口、[aclprofFinalize](#)时，会返回报错。

如果没有调用[aclprofInit](#)接口，调用[aclprofStart](#)接口、[aclprofStop](#)接口、[aclprofFinalize](#)时，会返回报错。

- [aclprofStart](#)接口在模型执行之前调用，若在模型执行过程中调用[aclprofStart](#)接口，Profiling采集到的数据为调用[aclprofStart](#)接口之后的数据，可能导致数据不完整。
调用[aclprofStart](#)接口时，可以指定从一个Device上采集性能数据，也可以指定从多个Device上采集性能数据。
一个用户APP进程内，如果连续调用多次[aclprofStart](#)接口，指定重复的Profiling配置，或指定的Device重复，会返回报错。
- 在用户APP的进程生命周期内，[aclprofInit](#)接口与[aclprofFinalize](#)接口配对使用，建议只调用一次，如该组合多次调用可以改变保存性能数据的文件的路径。
- [aclprofStart](#)接口与[aclprofStop](#)接口需配对使用。
- [aclprofSetConfig](#)接口必须在[aclprofStart](#)接口之前调用。一个APP进程内，可以根据需要选择一次或多次调用[aclprofSetConfig](#)接口。
- 调用[aclFinalize](#)并接收到正常退出码后为执行完毕，其他情况为非正常。由于性能数据采集不支持多进程并发执行，为确保驱动关闭正常，需要在前一个性能数据采集用例完全执行完毕之后再执行下一轮采集。建议在[aclFinalize](#)接口返回值上加入异常处理操作，方便展示执行状态与问题定位。

- 接口调用顺序：

- 建议的接口调用顺序如下，以“一个用户APP进程内采集多个模型推理时的性能数据”为例：
[aclInit](#)接口-->[aclprofInit](#)接口-->[aclprofStart](#)接口(指定Device 0和Device 1)-->模型1加载-->模型1执行-->[aclprofStop](#)接口(与[aclprofStart](#)接口的[aclprofConfig](#)数据保持一致)-->[aclprofStart](#)接口(指定Device 1和Device 2)-->模型2加载-->模型2执行-->[aclprofStop](#)接口(与[aclprofStart](#)接口的[aclprofConfig](#)数据保持一致)-->[aclprofFinalize](#)接口-->执行其它任务-->模型卸载-->[aclFinalize](#)接口
- 错误的接口调用顺序示例如下，以“一个用户APP进程内，如果连续调用多次[aclprofStart](#)接口，指定的Device重复”为例：
[aclInit](#)接口-->[aclprofInit](#)接口-->[aclprofStart](#)接口(指定Device 0和Device 1)-->[aclprofStart](#)接口(指定Device 1和Device 2)-->模型1加载-->模型1执行-->模型2加载-->模型2执行-->[aclprofStop](#)接口-->[aclprofStop](#)接口-->[aclprofFinalize](#)-->执行其它任务-->模型卸载-->[aclFinalize](#)接口

1.10.1.2 aclprofInit

函数功能

初始化Profiling，目前用于设置保存性能数据的文件的路径。

约束说明

与[aclprofFinalize](#)接口配对使用，先调用[aclprofInit](#)接口再调用[aclprofFinalize](#)接口。

函数原型

```
aclError aclprofInit(const char *profilerResultPath, size_t length)
```

参数说明

参数名	输入/输出	说明
profilerResultPath	输入	指定保存性能数据的文件的路径，此路径为绝对路径。
length	输入	profilerResultPath的长度，单位为Byte，最大长度不超过4096字节。

返回值说明

返回0表示成功，返回其他值表示失败。

参考资源

接口调用示例，参见Profiling性能数据采集。

1.10.1.3 aclprofSetConfig

函数功能

aclprofCreateConfig接口的扩展接口，用于设置性能数据采集参数。
该接口支持多次调用，用户需要保证数据的一致性和准确性。

约束说明

先调用aclprofSetConfig接口再调用[aclprofStart](#)接口，可根据需求选择调用该接口。

函数原型

```
aclError aclprofSetConfig(aclprofConfigType configType, const char *config, size_t configLength)
```

参数说明

参数名	输入/输出	说明
configType	输入	作为configType参数值。每个枚举表示不同采集配置，若要使用该接口下不同的选项采集多种性能数据，则需要多次调用该接口，详细说明如下： - ACL_PROF_STORAGE_LIMIT：指定落盘目录允许存放的最大文件容量，有效取值范围为[200, 4294967295]，单位为MB。 - ACL_PROF_SYS_HARDWARE_MEM_FREQ：片上内存带宽及内存、LLC的读写带宽数据、acc_pmu数据和SoC传输带宽信息、组件内存采集频率，范围[1,100]。 不同产品支持情况不同，请以实际实现为准。 说明 已知在安装有glibc<2.34的环境上采集memory数据，可能触发glibc的一个已知Bug 19329，通过升级环境的glibc版本可解决此问题。 - ACL_PROF_LLC_MODE：LLC Profiling采集事件。要求同时设置ACL_PROF_SYS_HARDWARE_MEM_FREQ。可以设置为： - Atlas 200/300/500 推理产品： - capacity：采集AI CPU和控制 CPU的LLC capacity数据。 - bandwidth：采集LLC bandwidth。默认值为capacity。 - Atlas 200/500 A2推理产品： - read：读事件，三级缓存读速率。 - write：写事件，三级缓存写速率。默认为read。 - Atlas 推理系列产品： - read：读事件，三级缓存读速率。 - write：写事件，三级缓存写速率。默认为read。 - Atlas 训练系列产品： - read：读事件，三级缓存读速率。 - write：写事件，三级缓存写速率。默认为read。 - Atlas A2训练系列产品/Atlas 800I A2推理产品： - read：读事件，三级缓存读速率。 - write：写事件，三级缓存写速率。默认为read。 - ACL_PROF_SYS_IO_FREQ：NIC、ROCE采集频率，范围[1,100]，单位hz。

参数名	输入/输出	说明
		- Atlas 200/300/500 推理产品：支持采集NIC。 - Atlas 200/500 A2推理产品：仅RC场景支持采集NIC，容器场景参数不生效。 - Atlas 训练系列产品：支持采集NIC和ROCE。 - Atlas A2训练系列产品/Atlas 800I A2推理产品：支持采集NIC和ROCE。 • ACL_PROF_SYS_INTERCONNECTION_FREQ：集合通信带宽数据（HCCS）、PCIe数据采集开关、片间传输带宽信息采集频率，范围[1,50]，单位hz。 - Atlas 推理系列产品：支持采集PCIe数据。 - Atlas 训练系列产品：支持采集HCCS、PCIe数据。 - Atlas A2训练系列产品/Atlas 800I A2推理产品：支持采集HCCS、PCIe数据、片间传输带宽信息。 • ACL_PROF_DVPP_FREQ：DVPP采集频率，范围[1,100]。 Atlas 推理系列产品：支持采集性能数据，暂不支持解析。 • ACL_PROF_HOST_SYS：Host侧进程级别的性能数据采集开关，取值包括cpu和mem。 • ACL_PROF_HOST_SYS_USAGE：Host侧系统和所有进程的性能数据采集开关，取值包括cpu和mem。 • ACL_PROF_HOST_SYS_USAGE_FREQ：CPU利用率、内存利用率的采集频率，范围[1,50]。
config	输入	指定配置项参数值。
configLength	输入	config的长度，单位为Byte，最大长度不超过256字节。

返回值说明

返回0表示成功，返回其他值表示失败。

1.10.1.4 aclprofStart

函数功能

下发Profiling请求，使能对应数据的采集。

用户可根据需要，在模型执行过程中按需调用aclprofStart接口，Profiling采集到的数据为调用该接口之后的数据。

约束说明

与[aclprofStop](#)接口配对使用，先调用[aclprofStart](#)接口再调用[aclprofStop](#)接口。

函数原型

aclError [aclprofStart](#)(const **aclprofConfig** *profilerConfig)

参数说明

参数名	输入/输出	说明
profilerConfig	输入	指定Profiling配置数据。 需提前调用 1.13.95.1 aclprofCreateConfig 接口创建 aclprofConfig 类型的数据。

返回值说明

返回0表示成功，返回其他值表示失败。

参考资源

接口调用示例，参见Profiling性能数据采集。

1.10.1.5 aclprofStop

函数功能

停止Profiling数据采集。

约束说明

与[aclprofStart](#)接口配对使用，先调用[aclprofStart](#)接口再调用[aclprofStop](#)接口。

函数原型

aclError [aclprofStop](#)(const **aclprofConfig** *profilerConfig)

参数说明

参数名	输入/输出	说明
profilerConfig	输入	指定停止Profiling数据采集的配置。 与 aclprofStart 接口中的 aclprofConfig 类型数据保持一致。

返回值说明

返回0表示成功，返回其他值表示失败。

参考资源

接口调用示例，参见Profiling性能数据采集。

1.10.1.6 aclprofFinalize

函数功能

结束Profiling。

约束说明

与[aclprofInit](#)接口配对使用，先调用[aclprofInit](#)接口再调用[aclprofFinalize](#)接口。

函数原型

[aclError](#) [aclprofFinalize\(\)](#)

参数说明

无

返回值说明

返回0表示成功，返回其他值表示失败。

参考资源

接口调用示例，参见Profiling性能数据采集。

1.10.2 Profiling AscendCL API for Extension (Profiling AscendCL API 扩展接口)

1.10.2.1 功能介绍

功能

当用户需要定位应用程序或上层框架程序的性能瓶颈时，可在Profiling采集进程内（[aclprofStart](#)接口、[aclprofStop](#)接口之间）调用Profiling AscendCL API扩展接口（统称为msproftx功能），开启记录应用程序执行期间特定事件发生的时间跨度，并将数据写入Profiling数据文件，再使用Profiling工具解析该文件，并导出展示性能分析数据。

Profiling工具解析导出操作请参见《[性能调优工具使用指南](#)》下的“高级功能>数据解析与导出”。

接口调用方式：在[aclprofStart](#)和[aclprofStop](#)接口之间调用[aclprofCreateStamp](#)、[1.10.2.3 aclprofSetStampTraceMessage](#)、[1.10.2.4 aclprofMark](#)、[1.10.2.5](#)

[aclprofMarkEx](#)、[aclprofPush](#)、[aclprofPop](#)、[aclprofRangeStart](#)、[aclprofRangeStop](#)、[aclprofDestroyStamp](#)接口。该方式可获取应用程序执行期间特定时间发生的事件并记录事件发生的时间跨度。

一个进程内，可以根据需求多次调用这些接口。

接口约束说明

- **调用接口要求：**msproftx功能相关接口须在[aclprofStart](#)接口与[aclprofStop](#)接口之间调用。其中配对使用的接口有：[aclprofCreateStamp](#)/[aclprofDestroyStamp](#)、[aclprofPush](#)/[aclprofPop](#)、[aclprofRangeStart](#)/[aclprofRangeStop](#)。
- **接口调用顺序：**[aclprofStart](#)接口(指定Device 0和Device 1)-->[aclprofCreateStamp](#)接口-->[aclprofSetStampTraceMessage](#)接口-->[aclprofMark](#)接口-->([aclprofPush](#)接口-->[aclprofPop](#)接口)或([aclprofRangeStart](#)接口-->[aclprofRangeStop](#)接口)-->[aclprofDestroyStamp](#)接口-->[aclprofStop](#)接口(与[aclprofStart](#)接口的[aclprofConfig](#)数据保持一致)。

1.10.2.2 aclprofCreateStamp

函数功能

创建msproftx事件标记。后续调用[aclprofMark](#)、[aclprofSetStampTraceMessage](#)、[aclprofPush](#)和[aclprofRangeStart](#)接口时需要以描述该事件的指针作为输入，表示记录该事件发生的时间跨度。

约束说明

与[1.10.2.10 aclprofDestroyStamp](#)接口配对使用，需提前调用[1.10.1.4 aclprofStart](#)接口。

函数原型

```
void *aclprofCreateStamp()
```

返回值说明

- 返回void类型的指针，表示成功。
- 返回nullptr，表示失败。

参考资源

接口调用示例，参见Profiling性能数据采集。

1.10.2.3 aclprofSetStampTraceMessage

函数功能

为msproftx事件标记携带字符串描述，在Profiling解析并导出结果中msprof_tx summary数据展示。

Profiling解析并导出结果请参见《[性能调优工具使用指南](#)》下的“Profiling数据说明>msproftx数据说明”。

约束说明

在[1.10.2.2 aclprofCreateStamp](#)接口和[1.10.2.10 aclprofDestroyStamp](#)接口之间调用。

函数原型

aclError aclprofSetStampTraceMessage(void *stamp, const char *msg, uint32_t msgLen)

参数说明

参数名	输入/输出	说明
stamp	输入	Stamp指针，指代msproftx事件标记。指定 aclprofCreateStamp 接口的指针。
msg	输入	Stamp信息字符串指针。
msgLen	输入	字符串长度。

返回值说明

返回0表示成功，返回其他值表示失败。

参考资源

接口调用示例，参见Profiling性能数据采集。

1.10.2.4 aclprofMark

函数功能

msproftx标记瞬时事件。

调用此接口后，Profiling自动在Stamp指针中加上当前时间戳，将Event type设置为Mark，表示开始一次msproftx采集。

约束说明

在[1.10.2.2 aclprofCreateStamp](#)接口和[1.10.2.10 aclprofDestroyStamp](#)接口之间调用。

函数原型

aclError aclprofMark(void *stamp)

参数说明

参数名	输入/输出	说明
stamp	输入	Stamp指针，指代msproftx事件标记。 指定 aclprofCreateStamp 接口的指针。

返回值说明

返回0表示成功，返回其他值表示失败。

参考资源

接口调用示例，参见Profiling性能数据采集。

1.10.2.5 aclprofMarkEx

函数功能

aclprofMarkEx打点接口。

调用此接口向配置的Stream流上下发打点任务，用于标识Host侧打点与Device侧打点任务的关系。

函数原型

aclError aclprofMarkEx(const char *msg, size_t msgLen, **aclrtStream** stream)

参数说明

参数名	输入/输出	说明
msg	输入	打点信息字符串指针。
msgLen	输入	字符串长度。最大支持127字符。
stream	输入	指定Stream。

返回值说明

返回0表示成功，返回其他值表示失败。

参考资源

接口调用示例，参见Profiling性能数据采集。

1.10.2.6 aclprofPush

函数功能

msproftx用于记录事件发生的时间跨度的开始时间。

调用此接口后，Profiling自动在Stamp指针中记录开始的时间戳，将Event type设置为Push/Pop。

约束说明

- 与[1.10.2.7 aclprofPop](#)接口成对使用，表示时间跨度的开始和结束。
- 在[1.10.2.2 aclprofCreateStamp](#)接口和[1.10.2.10 aclprofDestroyStamp](#)接口之间调用。
- 不能跨线程调用，若需要跨线程可使用[1.10.2.8 aclprofRangeStart](#)/[1.10.2.9 aclprofRangeStop](#)接口。

函数原型

aclError aclprofPush(void *stamp)

参数说明

参数名	输入/输出	说明
stamp	输入	Stamp指针，指代msproftx事件标记。 指定 aclprofCreateStamp 接口的指针。

返回值说明

返回0表示成功，返回其他值表示失败。

参考资源

接口调用示例，参见Profiling性能数据采集。

1.10.2.7 aclprofPop

函数功能

msproftx用于记录事件发生的时间跨度的结束时间。

调用此接口后，Profiling自动在Stamp指针中记录采集结束的时间戳。

约束说明

- 与[1.10.2.6 aclprofPush](#)接口成对使用，表示时间跨度的开始和结束。
- 在[1.10.2.2 aclprofCreateStamp](#)接口和[1.10.2.10 aclprofDestroyStamp](#)接口之间调用。

- 不能跨线程调用。若需要跨线程可使用[1.10.2.8 aclprofRangeStart](#)/[1.10.2.9 aclprofRangeStop](#)接口。

函数原型

aclError aclprofPop()

返回值说明

返回0表示成功，返回其他值表示失败。

参考资源

接口调用示例，参见Profiling性能数据采集。

1.10.2.8 aclprofRangeStart

函数功能

msproftx用于记录事件发生的时间跨度的开始时间。

调用此接口后，Profiling自动在Stamp指针记录采集开始的时间戳，将Event type设置为Start/Stop，生成一个进程唯一的id，并将Stamp保存在以进程粒度维护的一个map中。

约束说明

- 与[1.10.2.9 aclprofRangeStop](#)接口成对使用，表示时间跨度的开始和结束。
- 在[1.10.2.2 aclprofCreateStamp](#)接口和[1.10.2.10 aclprofDestroyStamp](#)接口之间调用。
- 可以跨线程调用。

函数原型

aclError aclprofRangeStart(void *stamp, uint32_t *rangeld)

参数说明

参数名	输入/输出	说明
stamp	输入	Stamp指针，指代msproftx事件标记。 指定 aclprofCreateStamp 接口的指针。
rangeld	输出	msproftx事件标记的唯一标识。用于在跨线程时区分。

返回值说明

返回0表示成功，返回其他值表示失败。

参考资源

接口调用示例，参见Profiling性能数据采集。

1.10.2.9 aclprofRangeStop

函数功能

msproftx用于记录事件发生的时间跨度的结束时间。
调用此接口后，Profiling自动在Stamp指针中记录采集结束的时间戳。

约束说明

- 与1.10.2.8 [aclprofRangeStart](#)接口成对使用，表示时间跨度的开始和结束。
- 在1.10.2.2 [aclprofCreateStamp](#)接口和1.10.2.10 [aclprofDestroyStamp](#)接口之间调用。
- 可以跨线程调用。

函数原型

aclError [aclprofRangeStop](#)(uint32_t rangeld)

参数说明

参数名	输入/输出	说明
rangeld	输出	msproftx事件标记的唯一标识。用于在跨线程时区分。

返回值说明

返回0表示成功，返回其他值表示失败。

参考资源

接口调用示例，参见Profiling性能数据采集。

1.10.2.10 aclprofDestroyStamp

函数功能

释放msproftx事件标记。

约束说明

与1.10.2.2 [aclprofCreateStamp](#)接口配对使用，在1.10.1.5 [aclprofStop](#)接口前调用。

函数原型

```
void aclprofDestroyStamp(void *stamp)
```

参数说明

参数名	输入/输出	说明
stamp	输入	Stamp指针，指代msproftx事件标记。 指定 aclprofCreateStamp 接口的指针。

参考资源

接口调用示例，参见Profiling性能数据采集。

1.10.3 Profiling AscendCL API for Subscription (订阅算子信息的 Profiling AscendCL API)

1.10.3.1 功能介绍

功能

该章节描述使用订阅算子信息的Profiling AscendCL API采集性能数据，实现将采集到的Profiling数据解析后写入管道，由用户读入内存，再由用户调用AscendCL的接口获取性能数据。

接口调用方式：[aclprofModelSubscribe](#)接口、aclprofGet*接口、[aclprofModelUnSubscribe](#)接口配合使用，实现该方式的性能数据采集，当前支持获取网络模型中算子的性能数据，包括算子名称、算子类型名称、算子执行时间等。

总体约束

不能与[1.10.1 Profiling AscendCL API \(通过Profiling AscendCL API采集并落盘性能数据 \)](#)的接口交叉调用：[aclprofModelSubscribe](#)接口和[aclprofModelUnSubscribe](#)接口之间不能调用[aclprofInit](#)接口、[aclprofStart](#)接口、[aclprofStop](#)接口和[aclprofFinalize](#)接口。

接口约束说明

- 接口调用要求：
 - [aclprofModelSubscribe](#)接口在模型执行之前调用，若在模型执行过程中调用[aclprofModelSubscribe](#)接口，Profiling采集到的数据为调用[aclprofModelSubscribe](#)接口之后的数据，可能导致数据不完整。
 - [aclprofModelSubscribe](#)接口需与[aclprofModelUnSubscribe](#)接口配对使用，不能在调用[aclprofModelUnSubscribe](#)接口前，多次调用[aclprofModelSubscribe](#)接口重复订阅相同的模型。

- 不能调用[aclprofModelSubscribe](#)接口订阅不存在的模型ID。
- 不能调用[aclprofModelUnSubscribe](#)接口取消订阅不存在的模型ID或未订阅过的模型ID。
- 如果在同一个Device上加载了多个模型，只能对多个模型下发同样的订阅配置。
- 接口调用顺序：
 - 建议的接口调用顺序如下：
模型加载-->[aclprofModelSubscribe](#)接口-->[aclprofGetOpDescSize](#)接口-->[aclprofGetOpNum](#)接口-->[aclprofGetOpType/aclprofGetOpName/aclprofGetOpStart/aclprofGetOpEnd/aclprofGetOpDuration/aclprofGetModelId](#)接口-->[aclprofModelUnSubscribe](#)接口
 - 错误的接口调用顺序示例如下，以重复定义同一个模型为例：
模型1加载-->[aclprofModelSubscribe](#)接口(指定模型1)-->[aclprofModelSubscribe](#)接口(指定模型1)-->[aclprofModelUnSubscribe](#)接口

1.10.3.2 aclprofModelSubscribe

函数功能

网络场景下，订阅算子的基本信息，包括算子名称、算子类型、算子执行耗时等。

约束说明

与[aclprofModelUnSubscribe](#)接口配对使用。

函数原型

```
aclError aclprofModelSubscribe(uint32_t modelId, const
aclprofSubscribeConfig *profSubscribeConfig)
```

参数说明

参数名	输入/输出	说明
modelId	输入	待订阅的网络模型的ID。 调用 aclmdlLoadFromFile 接口/ aclmdlLoadFromMem 接口/ aclmdlLoadFromFileWithMem 接口/ aclmdlLoadFromMemWithMem 接口加载模型成功后，会返回模型ID。
profSubscribeConfig	输入	待订阅的配置信息。 需提前调用 aclprofCreateSubscribeConfig 接口创建aclprofSubscribeConfig类型的数据。

返回值说明

返回0表示成功，返回其他值表示失败。

参考资源

接口调用示例，参见Profiling性能数据采集。

1.10.3.3 aclprofModelUnSubscribe

函数功能

网络场景下，取消订阅算子的基本信息，包括算子名称、算子类型、算子执行耗时等。

约束说明

与[aclprofModelSubscribe](#)接口配对使用。

函数原型

aclError aclprofModelUnSubscribe(uint32_t modelId)

参数说明

参数名	输入/输出	说明
modelId	输入	已订阅的模型的ID。

返回值说明

返回0表示成功，返回其他值表示失败。

参考资源

接口调用示例，参见Profiling性能数据采集。

1.10.3.4 aclprofGetOpDescSize

函数功能

获取单个算子数据结构的大小，单位为Byte。当前版本中约定每个算子数据结构的大小是一样的。

建议用户新建一个线程，在新线程内调用该接口，否则可能阻塞主线程中的其它任务调度。

函数原型

aclError aclprofGetOpDescSize(size_t *opDescSize)

参数说明

参数名	输入/输出	说明
opDescSize	输出	算子数据结构的大小。

返回值说明

返回0表示成功，返回其他值表示失败。

参考资源

接口调用示例，参见Profiling性能数据采集。

1.10.3.5 aclprofGetOpNum

函数功能

获取指定内存中算子的数量。

建议用户新建一个线程，在新线程内调用该接口，否则可能阻塞主线程中的其它任务调度。

函数原型

aclError aclprofGetOpNum(const void *opInfo, size_t opInfoLen, uint32_t *opNumber)

参数说明

参数名	输入/输出	说明
opInfo	输入	指定算子信息的内存地址。 调用 aclprofGetOpDescSize 接口获取到单个算子数据结构的大小后，用户需按照“单个算子数据结构的大小*整数系数”得到的数值申请内存，用于存放Profiling采集到的算子信息数据，作为本接口的输入。
opInfoLen	输入	算子信息的长度。
opNumber	输出	算子的数量。

返回值说明

返回0表示成功，返回其他值表示失败。

参考资源

接口调用示例，参见Profiling性能数据采集。

1.10.3.6 aclprofGetOpTypeLen

函数功能

获取算子类型的字符串长度，用于内存申请。

建议用户新建一个线程，在新线程内调用该接口，否则可能阻塞主线程中的其它任务调度。

函数原型

```
aclError aclprofGetOpTypeLen(const void *opInfo, size_t opInfoLen, uint32_t index, size_t *opTypeLen)
```

参数说明

参数名	输入/输出	说明
opInfo	输入	包含算子信息的地址。
opInfoLen	输入	算子信息的长度。
index	输入	指定获取第几个算子的算子类型名称。 用户调用 aclprofGetOpNum 接口获取算子数量后，这个index的取值范围：[0, (算子数量-1)]。
opTypeLen	输出	opType的长度。

返回值说明

返回0表示成功，返回其他值表示失败。

参考资源

接口调用示例，参见Profiling性能数据采集。

1.10.3.7 aclprofGetOpType

函数功能

获取指定算子的算子类型名称。

建议用户新建一个线程，在新线程内调用该接口，否则可能阻塞主线程中的其它任务调度。

函数原型

```
aclError aclprofGetOpType(const void *opInfo, size_t opInfoLen, uint32_t index, char *opType, size_t opTypeLen)
```

参数说明

参数名	输入/输出	说明
opInfo	输入	包含算子信息的地址。
opInfoLen	输入	算子信息的长度。
index	输入	指定获取第几个算子的算子类型名称。 用户调用 aclprofGetOpNum 接口获取算子数量后，这个index的取值范围：[0, (算子数量-1)]。
opType	输出	算子类型名称。
opTypeLen	输入	opType的实际内存申请长度。取值范围建议不小于 aclprofGetOpTypeLen ，否则内容会有截断。

返回值说明

返回0表示成功，返回其他值表示失败。

参考资源

接口调用示例，参见Profiling性能数据采集。

1.10.3.8 aclprofGetOpNameLen

函数功能

获取算子名称的字符串长度，用于内存申请。

建议用户新建一个线程，在新线程内调用该接口，否则可能阻塞主线程中的其它任务调度。

函数原型

```
aclError aclprofGetOpNameLen(const void *opInfo, size_t opInfoLen, uint32_t index, size_t *opNameLen)
```

参数说明

参数名	输入/输出	说明
opInfo	输入	包含算子信息的地址。

参数名	输入/输出	说明
opInfoLen	输入	算子信息的长度。
index	输入	指定获取第几个算子的算子名称长度。 用户调用 aclprofGetOpNum 接口获取算子数量后，这个index的取值范围：[0, (算子数量-1)]。
opNameLen	输出	opName的实际内存申请长度。

返回值说明

返回0表示成功，返回其他值表示失败。

参考资源

接口调用示例，参见Profiling性能数据采集。

1.10.3.9 aclprofGetOpName

函数功能

获取指定算子的算子名称。

建议用户新建一个线程，在新线程内调用该接口，否则可能阻塞主线程中的其它任务调度。

函数原型

aclError aclprofGetOpName(const void *opInfo, size_t opInfoLen, uint32_t index, char *opName, size_t opNameLen)

参数说明

参数名	输入/输出	说明
opInfo	输入	包含算子信息的地址。
opInfoLen	输入	算子信息的长度。
index	输入	指定获取第几个算子的算子名称。 用户调用 aclprofGetOpNum 接口获取算子数量后，这个index的取值范围：[0, (算子数量-1)]。
opName	输出	算子名称。
opNameLen	输入	opName的实际内存申请长度。取值范围建议不小于 aclprofGetOpNameLen ，否则内容会有截断。

返回值说明

返回0表示成功，返回其他值表示失败。

参考资源

接口调用示例，参见Profiling性能数据采集。

1.10.3.10 aclprofGetOpStart

函数功能

获取算子执行的开始时间，单位为ns。

建议用户新建一个线程，在新线程内调用该接口，否则可能阻塞主线程中的其它任务调度。

函数原型

uint64_t aclprofGetOpStart(const void *opInfo, size_t opInfoLen, uint32_t index)

参数说明

参数名	输入/输出	说明
opInfo	输入	包含算子信息的地址。
opInfoLen	输入	算子信息的长度。
index	输入	指定获取第几个算子执行的开始时间。 用户调用 aclprofGetOpNum 接口获取算子数量后，这个index的取值范围：[0, (算子数量-1)]。

返回值说明

算子执行的开始时间。

参考资源

接口调用示例，参见Profiling性能数据采集。

1.10.3.11 aclprofGetOpEnd

函数功能

获取算子执行的结束时间，单位为ns。

建议用户新建一个线程，在新线程内调用该接口，否则可能阻塞主线程中的其它任务调度。

函数原型

uint64_t aclprofGetOpEnd(const void *opInfo, size_t opInfoLen, uint32_t index)

参数说明

参数名	输入/输出	说明
opInfo	输入	包含算子信息的地址。
opInfoLen	输入	算子信息的长度。
index	输入	指定获取第几个算子执行的结束时间。 用户调用 aclprofGetOpNum 接口获取算子数量后，这个index的取值范围：[0, (算子数量-1)]。

返回值说明

算子执行结束时间。

参考资源

接口调用示例，参见Profiling性能数据采集。

1.10.3.12 aclprofGetOpDuration

函数功能

获取算子执行的耗时时间，单位为ns。

建议用户新建一个线程，在新线程内调用该接口，否则可能阻塞主线程中的其它任务调度。

函数原型

uint64_t aclprofGetOpDuration(const void *opInfo, size_t opInfoLen, uint32_t index)

参数说明

参数名	输入/输出	说明
opInfo	输入	包含算子信息的地址。
opInfoLen	输入	算子信息的长度。

参数名	输入/输出	说明
index	输入	指定获取第几个算子执行的耗时时间。 用户调用 aclprofGetOpNum 接口获取算子数量后，这个index的取值范围：[0, (算子数量-1)]。

返回值说明

算子执行的耗时时间。

参考资源

接口调用示例，参见Profiling性能数据采集。

1.10.3.13 aclprofGetModelId

函数功能

获取指定算子所在模型的ID。

建议用户新建一个线程，在新线程内调用该接口，否则可能阻塞主线程中的其它任务调度。

函数原型

size_t aclprofGetModelId(const void *opInfo, size_t opInfoLen, uint32_t index)

参数说明

参数名	输入/输出	说明
opInfo	输入	包含算子信息的地址。
opInfoLen	输入	算子信息的长度。
index	输入	指定获取第几个算子所在模型的ID。 用户调用 aclprofGetOpNum 接口获取算子数量后，这个index的取值范围：[0, (算子数量-1)]。

返回值说明

模型的ID。

参考资源

接口调用示例，参见Profiling性能数据采集。

1.10.4 Profiling AscendCL API (PyTorch 场景标记迭代时间)

1.10.4.1 aclprofGetStepTimestamp

函数功能

利用[单算子加载与执行接口](#)实现训练的场景下，使用本接口用于标记迭代开始与结束时间，为后续Profiling解析提供迭代标识，以便以迭代为粒度展示性能数据。

函数原型

aclError aclprofGetStepTimestamp(**aclprofStepInfo*** stepInfo, **aclprofStepTag** tag, **aclrtStream** stream)

参数说明

参数名	输入/输出	说明
stepinfo	输入	指定迭代信息。需提前调用 aclprofCreateStepInfo 接口创建 aclprofStepInfo 类型的数据。
tag	输入	用于标记迭代开始或结束。在迭代开始时传入枚举值 ACL_STEP_START，迭代结束时需传入枚举值 ACL_STEP_END。
stream	输入	指定Stream。

返回值说明

返回0表示成功，返回其他值表示失败。

1.11 数据传输

1.11.1 Tensor 数据传输

TDT (Tensor Data Transfer) 用于Host与Device之间传递Tensor数据，比如训练的预处理后数据，比如NN计算时的loss数据。

1.11.1.1 acltdtCreateChannel

- Atlas 200/300/500 推理产品，不支持该接口。
- Atlas 200/500 A2推理产品，不支持该接口。
- Atlas 推理系列产品，不支持该接口。
- Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

创建acldtdChannelHandle类型的数据，表示可以用于向Device发送数据或是从Device接收数据的通道。通道使用完成后，需及时依次调用[acldtdStopChannel](#)、[acldtdDestroyChannel](#)接口释放通道资源。

约束说明

仅支持在昇腾虚拟化实例（ Ascend Virtual Instance ）场景下使用本接口。

函数原型

acldtdChannelHandle *acldtdCreateChannel(uint32_t deviceId, const char *name)

参数说明

参数名	输入/输出	说明
deviceId	输入	Device ID。 用户调用 aclrtGetDeviceCount 接口获取可用的Device数量后，这个Device ID的取值范围： [0, (可用的Device数量-1)]
name	输入	队列通道名称的指针。

返回值说明

- 返回acldtdChannelHandle类型的指针，表示成功。
- 返回nullptr，表示失败。

1.11.1.2 acldtdSendTensor

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 200/500 A2推理产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

函数功能

从Host向Device发送预处理好的数据。

函数原型

[aclError](#) acldtdSendTensor(const acldtdChannelHandle *handle,
const [acldtdDataset](#) *dataset,
int32_t timeout)

参数说明

参数名	输入/输出	说明
handle	输入	指定通道。 需提前调用 acltdtCreateChannel 接口或 acltdtCreateChannelWithCapacity 接口创建acltdtChannelHandle类型的数据。
dataset	输入	向Device发送的数据的指针。
timeout	输入	等待超时时间。 该参数取值范围如下： • -1：阻塞方式，一直等待直到数据发送完成。 • 0：非阻塞方式，当通道满时，直接返回通道满这个错误，这时由用户自行设定重试间隔。 • >0：配置具体的超时时间，单位为毫秒。通道满时，等待达到超时时间后返回报错。超时时间受操作系统影响，一般偏差在操作系统的<code>一个时间片</code>内，例如，操作系统的<code>一个时间片</code>为4ms，用户设置的超时时间为1ms，则实际的超时时间在1ms到5ms范围内。在CPU负载高场景下，超时时间仍可能存在波动。

返回值说明

返回0表示成功，返回其他值表示失败。

1.11.1.3 acltdtReceiveTensor

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 200/500 A2推理产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

函数功能

在Host接收Device发过来的数据。

函数原型

```
aclError acltdtReceiveTensor(const acltdtChannelHandle *handle,  
acltdtDataset *dataset,  
int32_t timeout)
```

参数说明

参数名	输入/输出	说明
handle	输入	指定通道。 需提前调用 acltdtCreateChannel 接口或 acltdtCreateChannelWithCapacity 接口创建acltdtChannelHandle类型的数据。
dataset	输出	接收到的Device数据的指针。
timeout	输入	等待超时时间。 该参数取值范围如下： -1：阻塞方式，一直等待直到数据接收完成。 0：非阻塞方式，当通道空时，直接返回通道空这个错误，这时由用户自行设定重试间隔。 >0：配置具体的超时时间，单位为毫秒。通道空时，等待达到超时时间后返回报错。超时时间受操作系统影响，一般偏差在操作系统的 一个时间片内，例如，操作系统的 一个时间片为4ms，用户设置的超时时间为1ms，则实际的超时时间在1ms到5ms范围内。在CPU负载高场景下，超时时间仍可能存在波动。

返回值说明

返回0表示成功，返回其他值表示失败。

1.11.1.4 acltdtStopChannel

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 200/500 A2推理产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

函数功能

调用[acltdtSendTensor](#)接口发送数据时或调用[acltdtReceiveTensor](#)接口接收数据时，用户线程可能在没有数据时会卡住，此时如果需要退出的话，需要先将线程唤醒，该接口就是用来唤醒被卡住阻塞的线程用的。需要用户在发送、接收线程之外的一个线程里调用这个函数，来唤醒处于阻塞状态的发送/接收线程。

函数原型

[aclError](#) [acltdtStopChannel](#)([acltdtChannelHandle](#) *handle)

参数说明

参数名	输入/输出	说明
handle	输入	指定通道。 需提前调用 acltdtCreateChannel 接口或 acltdtCreateChannelWithCapacity 接口创建acltdtChannelHandle类型的数据。

返回值说明

返回0表示成功，返回其他值表示失败。

1.11.1.5 acltdtDestroyChannel

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 200/500 A2推理产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

函数功能

销毁acltdtChannelHandle类型的数据，只能销毁通过[acltdtCreateChannel](#)接口或[acltdtCreateChannelWithCapacity](#)接口创建的acltdtChannelHandle类型。

函数原型

[aclError](#) acltdtDestroyChannel(acltdtChannelHandle *handle)

参数说明

参数名	输入/输出	说明
handle	输入	待销毁的acltdtChannelHandle类型的指针。

返回值说明

返回0表示成功，返回非0表示失败。

1.11.1.6 acltdtCreateChannelWithCapacity

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 200/500 A2推理产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

函数功能

创建acldtdChannelHandle类型的数据，表示可以用于向Device发送数据或是从Device接收数据的通道，通道带容量。

函数原型

acldtdChannelHandle *acldtdCreateChannelWithCapacity(uint32_t deviceId, const char *name, size_t capacity)

参数说明

参数名	输入/输出	说明
deviceId	输入	Device ID。 用户调用 aclrtGetDeviceCount 接口获取可用的Device数量后，这个Device ID的取值范围：[0, (可用的Device数量-1)]
name	输入	队列通道名称的指针。
capacity	输入	队列通道容量，取值范围：[2, 8192]。

返回值说明

- 返回acldtdChannelHandle类型的指针，表示成功。
- 返回nullptr，表示失败。

1.11.1.7 acldtdQueryChannelSize

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 200/500 A2推理产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

函数功能

查询队列通道内的消息数量。

函数原型

aclError acldtdQueryChannelSize(const acldtdChannelHandle *handle, size_t *size)

参数说明

参数名	输入/输出	说明
handle	输入	指定通道。 需提前调用 acltdtCreateChannelWithCapacity 接口创建 acltdtChannelHandle 类型的数据。
size	输出	消息数量的指针。

返回值说明

返回0表示成功，返回其他值表示失败。

1.11.1.8 acltdtGetSliceInfoFromItem

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 200/500 A2推理产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

函数功能

用于输出Tensor分片信息。

使用场景：OutfeedEnqueueOpV2算子由于其功能要求需申请Device上的大块内存存放数据，在Device内存不足时，可能会导致内存申请失败，进而导致某些算子无法正常执行，该场景下，用户可以调用本接口获取Tensor分片信息（分片数量、分片索引），再根据分片信息拼接算子的Tensor数据。

函数原型

aclError [acltdtGetSliceInfoFromItem](#)(const [acltdtDataItem](#) *dataItem, size_t *sliceNum, size_t* sliceId)

参数说明

参数名	输入/输出	说明
dataItem	输入	acltdtDataItem 类型的指针。 acltdtDataItem 用于标识一个业务上的Tensor。 需提前调用 acltdtCreateDataItem 接口创建 acltdtDataItem 类型的数据。
sliceNum	输出	单个Tensor被切片的数量。
sliceId	输出	被切片Tensor的数据段索引。

返回值说明

返回0表示成功，返回其他值表示失败。

1.11.1.9 acltdtCleanChannel

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 200I/500 A2推理产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

函数功能

清空通道中的所有数据。

函数原型

```
aclError acltdtCleanChannel(acltdtChannelHandle *handle)
```

参数说明

参数名	输入/输出	说明
handle	输入	指定通道。 需提前调用 acltdtCreateChannelWithCapacity 接口创建acltdtChannelHandle类型的数据。

返回值说明

返回0表示成功，返回非0表示失败。

1.11.2 共享队列管理

1.11.2.1 功能及约束说明

功能说明

本章提供的接口用于实现Device侧的数据队列和数据转发功能，用户可以结合本章的队列管理和共享Buffer机制以实现数据驱动编程。

约束说明

队列只允许单个消费者或单个生产者操作，Enqueue（指向队列中添加数据）或Dequeue（指从队列中获取数据）时应单线程操作，不支持并发。

1.11.2.2 acltdtCreateQueue

函数功能

创建队列。

函数原型

aclError acltdtCreateQueue(const **acltdtQueueAttr** *attr, uint32_t *qid)

参数说明

参数名	输入/输出	说明
attr	输入	队列属性配置信息的指针。 需提前调用 acltdtCreateQueueAttr 接口创建acltdtQueueAttr类型的数据，再调用 acltdtSetQueueAttr 接口设置队列属性值（例如，队列名）。 若该参数为nullptr时，则使用队列属性默认值，即队列名系统自动分配，队列深度默认为8。
qid	输出	队列ID的指针。

返回值说明

返回0表示成功，返回其他值表示失败。

1.11.2.3 acltdtDestroyQueue

函数功能

销毁通过acltdtCreateQueue接口创建的队列。

函数原型

aclError acltdtDestroyQueue(uint32_t qid)

参数说明

参数名	输入/输出	说明
qid	输入	指定需销毁的队列。

返回值说明

返回0表示成功，返回其他值表示失败。

1.11.2.4 acltdtEnqueueData

函数功能

向队列中添加数据。

函数原型

aclError acltdtEnqueueData(uint32_t qid, const void *data, size_t dataSize, const void *userData, size_t userDataSize, int32_t timeout, uint32_t rsv);

参数说明

参数名	输入/输出	说明
qid	输入	需要添加数据的队列。 队列需提前调用 acltdtCreateQueue 接口创建。
data	输入	内存数据指针，支持Host侧或Device侧的内存。
dataSize	输入	内存数据大小，单位为Byte。
userData	输入	用户自定义数据指针。 若用户没有自定义数据，则传nullptr。
userDataSize	输入	用户自定义数据大小（<=96Byte）。 若用户没有自定义数据，则传0。
timeout	输入	等待超时时间。当队列满时，如果向队列中添加数据，系统内部会根据设置的等待超时时间来决定如何处理。 该参数取值范围如下： • -1：阻塞方式，一直等待直到数据成功加入队列。 • 0：非阻塞方式（仅支持device场景，host场景无效），当队列满时，直接返回队列满这个错误，这时由用户自行设定重试间隔。 • >0：配置具体的超时时间，单位为毫秒。队列满时，等待达到超时时间后返回报错。超时时间受操作系统影响，一般偏差在操作系统的—个时间片内，例如，操作系统的—个时间片为4ms，用户设置的超时时间为1ms，则实际的超时时间在1ms到5ms范围内。在CPU负载高场景下，超时时间仍可能存在波动。
rsv	输入	预留参数，暂不支持。当前可设置为0。

返回值说明

返回0表示成功，返回其他值表示失败。

1.11.2.5 acltdtDequeueData

函数功能

从队列中获取数据。

函数原型

aclError acltdtDequeueData(uint32_t qid, void *data, size_t dataSize, size_t *retDataSize, void *userData, size_t userDataSize, int32_t timeout);

参数说明

参数名	输入/输出	说明
qid	输入	需要从哪个队列中获取数据。 队列需提前调用 acltdtCreateQueue 接口创建。
data	输入	内存数据指针，支持Host侧或Device侧的内存。
dataSize	输入	内存数据大小，单位为Byte。
retDataSize	输入&输出	返回实际数据大小，单位为Byte。
userData	输入	用户自定义数据指针。 若用户没有自定义数据，则传nullptr。
userDataSize	输入	用户自定义数据大小（<=96Byte）。 若用户没有自定义数据，则传0。
timeout	输入	等待超时时间。当队列空时，如果从队列中获取数据，系统内部会根据设置的等待超时时间来决定如何处理。 该参数取值范围如下： • -1：阻塞方式，一直等待直到队列有数据后返回。 • 0：非阻塞方式（仅支持device场景，host场景无效），当队列空时，直接返回队列空这个错误，这时由用户自行设定重试间隔。 • >0：配置具体的超时时间，单位为毫秒。队列空时，等待达到超时时间后返回报错。超时时间受操作系统影响，一般偏差在操作系统的—个时间片内，例如，操作系统的—个时间片为4ms，用户设置的超时时间为1ms，则实际的超时时间在1ms到5ms范围内。在CPU负载高场景下，超时时间仍可能存在波动。

返回值说明

返回0表示成功，返回其他值表示失败。

1.11.2.6 acltdtEnqueue

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 200/500 A2推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

函数功能

向队列中添加数据，存放数据的内存必须调用[acltdtAllocBuf](#)接口申请。

函数原型

[aclError](#) [acltdtEnqueue](#)([uint32_t](#) qid, [acltdtBuf](#) buf, [int32_t](#) timeout)

参数说明

参数名	输入/输出	说明
qid	输入	需要添加数据的队列。 队列需提前调用 acltdtCreateQueue 接口创建。
buf	输入	共享Buffer指针。 该内存必须提前调用 acltdtAllocBuf 接口申请。
timeout	输入	等待超时时间。当队列满时，如果向队列中添加数据，系统内部会根据设置的等待超时时间来决定如何处理。 该参数取值范围如下： -1：阻塞方式，一直等待直到数据成功加入队列。 0：非阻塞方式，当队列满时，直接返回队列满这个错误，这时由用户自行设定重试间隔。 >0：配置具体的超时时间，单位为毫秒。队列满时，等待达到超时时间后返回报错。超时时间受操作系统影响，一般偏差在操作系统的 一个时间片内，例如，操作系统的 一个时间片为4ms，用户设置的超时时间为1ms，则实际的超时时间在1ms到5ms范围内。在CPU负载高场景下，超时时间仍可能存在波动。

返回值说明

返回0表示成功，返回其他值表示失败。

1.11.2.7 acltdtDequeue

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 200/500 A2推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

函数功能

从队列中获取数据。

函数原型

```
aclError acltdtDequeue(uint32_t qid, acltdtBuf *buf, int32_t timeout)
```

参数说明

参数名	输入/输出	说明
qid	输入	需要从哪个队列中获取数据。 队列需提前调用 acltdtCreateQueue 接口创建。
buf	输出	共享Buffer指针。
timeout	输入	等待超时时间。当队列空时，如果从队列中获取数据，系统内部会根据设置的等待超时时间来决定如何处理。 该参数取值范围如下： • -1：阻塞方式，一直等待直到队列有数据后返回。 • 0：非阻塞方式，当队列空时，直接返回队列空这个错误，这时由用户自行设定重试间隔。 • >0：配置具体的超时时间，单位为毫秒。队列空时，等待达到超时时间后返回报错。超时时间受操作系统影响，一般偏差在操作系统的 一个时间片内，例如，操作系统的 一个时间片为4ms，用户设置的超时时间为1ms，则实际的超时时间在1ms到5ms范围内。在CPU负载高场景下，超时时间仍可能存在波动。

返回值说明

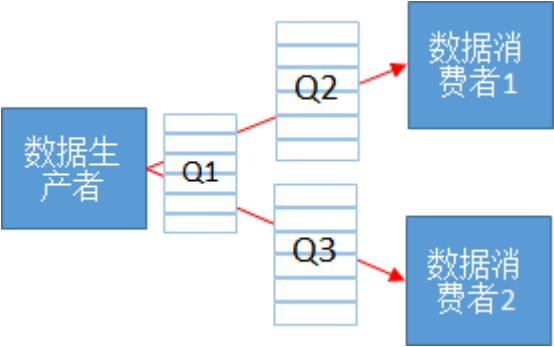
返回0表示成功，返回其他值表示失败。

1.11.2.8 acltdtBindQueueRoutes

函数功能

当应用存在数据一对多分发时，通过本接口绑定队列间数据转发路由关系。

如下图所示，可以建立两条路由关系Q1->Q2, Q1->Q3。



约束说明

- 数据一对多分发时，传递的共享Buffer数据是无锁的，消费者不能对数据进行inplace操作，如上图所示消费者1对共享数据的修改会导致消费者2访问的数据发生变化。
- 会对添加的队列路由关系的进行是否成环校验，不允许成环。
- 不支持多线程并发调用。

函数原型

```
aclError acltdtBindQueueRoutes(acltdtQueueRouteList *qRouteList)
```

参数说明

参数名	输入/输出	说明
qRouteList	输入/输出	路由关系数组的指针，接口调用完成后返回路由绑定结果。 需提前调用 acltdtCreateQueueRouteList 接口创建acltdtQueueRouteList类型的数据，再调用 acltdtAddQueueRoute 接口添加路由关系。

返回值说明

返回0表示成功，返回其他值表示失败。

只有当所有队列关系绑定成功且路由状态正常时，本接口才会返回成功；任何一条绑定失败，本接口返回失败，如果您需要知道具体哪个路由关系绑定失败，您可以先调用[acltdtGetQueueRoute](#)接口从路由关系数组中获取每一个路由关系，再调用[acltdtGetQueueRouteParam](#)接口查询绑定关系状态。

1.11.2.9 acltdtUnbindQueueRoutes

函数功能

解绑定数据队列路由关系。

函数原型

```
aclError acltdtUnbindQueueRoutes(acltdtQueueRouteList *qRouteList)
```

参数说明

参数名	输入/输出	说明
qRouteList	输入/输出	路由关系数组的指针，接口调用完成后返回路由去绑定结果。 可先通过 acltdtQueryQueueRoutes 获取路由关系数组。

返回值说明

返回0表示成功，返回其他值表示失败。

只有当所有队列关系解绑定成功，本接口才会返回成功；任何一条解绑定失败，本接口返回失败，如果您需要知道具体哪个路由关系解绑定失败，您可以先调用[acltdtGetQueueRoute](#)接口从路由关系数组中获取每一个路由关系，再调用[acltdtGetQueueRouteParam](#)接口查询绑定关系状态。

1.11.2.10 acltdtQueryQueueRoutes

函数功能

根据指定条件查询数据队列路由关系。

函数原型

```
aclError acltdtQueryQueueRoutes(const acltdtQueueRouteQueryInfo *queryInfo, acltdtQueueRouteList *qRouteList)
```

参数说明

参数名	输入/输出	说明
queryInfo	输入	查询条件的指针。 需提前调用 acltdtCreateQueueRouteQueryInfo 接口创建acltdtQueueRouteQueryInfo类型的数据。
qRouteList	输入&输出	路由关系数组的指针。 需提前调用 acltdtCreateQueueRouteList 接口创建acltdtQueueRouteList类型的数据。

返回值说明

返回0表示成功，返回其他值表示失败。

1.11.2.11 acltdtGrantQueue

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 200/500 A2推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

函数功能

进程间需要共享队列信息时，可以调用本接口给其它进程授予队列相关的权限，例如 Enqueue（指向队列中添加数据）权限、Dequeue（指从队列中获取数据）权限等。

约束说明

进程间传递队列相关信息时，安全性由用户保证。

函数原型

aclError acltdtGrantQueue(uint32_t qid, int32_t pid, uint32_t permission, int32_t timeout)

参数说明

参数名	输入/输出	说明
qid	输入	队列ID。
pid	输入	被授权进程的ID。
permission	输入	权限标识（队列生产者/消费者）。 用户选择如下多个宏进行逻辑或（例如： ACL_TDT_QUEUE_PERMISSION_DEQUEUE ACL_TDT_QUEUE_PERMISSION_ENQUEUE），作为 permission参数值。每个宏表示某一权限，详细说明如下： - ACL_TDT_QUEUE_PERMISSION_MANAGE：表示队列的管理权限。 - ACL_TDT_QUEUE_PERMISSION_DEQUEUE：表示 Dequeue权限。 - ACL_TDT_QUEUE_PERMISSION_ENQUEUE：表示 Enqueue权限。

参数名	输入/输出	说明
timeout	输入	等待超时时间，取值范围如下： • -1：阻塞方式，一直等待直到数据成功加入队列。 • 0：非阻塞方式，立即返回。 • >0：配置具体的超时时间，单位为毫秒，等到达超时时间后返回报错。超时时间受操作系统影响，一般偏差在操作系统的的一个时间片内，例如，操作系统的的一个时间片为4ms，用户设置的超时时间为1ms，则实际的超时时间在1ms到5ms范围内。在CPU负载高场景下，超时时间仍可能存在波动。

返回值说明

返回0表示成功，返回其他值表示失败。

1.11.2.12 acltdtAttachQueue

- Atlas 200/300/500 推理产品，不支持该接口。
- Atlas 200/500 A2推理产品，不支持该接口。
- Atlas 训练系列产品，不支持该接口。
- Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。
- Atlas 推理系列产品，不支持该接口。

函数功能

进程间需要共享队列信息时，在被授权的进程中调用本接口确认当前进程对队列有相应权限。

函数原型

```
aclError acltdtAttachQueue(uint32_t qid, int32_t timeout, uint32_t *permission)
```

参数说明

参数名	输入/输出	说明
qid	输入	队列ID。

参数名	输入/输出	说明
timeout	输入	等待超时时间，取值范围如下： • -1：阻塞方式，一直等待直到数据成功加入队列。 • 0：非阻塞方式，立即返回。 • >0：配置具体的超时时间，单位为毫秒，等到达超时时间后返回报错。超时时间受操作系统影响，一般偏差在操作系统的一个时间片内，例如，操作系统的一个时间片为4ms，用户设置的超时时间为1ms，则实际的超时时间在1ms到5ms范围内。在CPU负载高场景下，超时时间仍可能存在波动。
permission	输出	权限标识。 • 根据输出参数值的最后一个bit位判断是否有管理权限，0就是没有，1就是有。 • 根据输出参数值的倒数第二个bit位判断是否有从队列中获取数据的权限，0就是没有，1就是有。 • 根据输出参数值的倒数第三个bit位判断是否有向队列中添加数据的权限，0就是没有，1就是有。

返回值说明

返回0表示成功，返回其他值表示失败。

1.11.3 共享 Buffer 管理

本特性提供了跨进程共享Buffer管理能力，配合共享队列一起使用。共享Buffer的共享范围即是共享队列的授权确定，共享范围包括“一主多从”的这些进程。

1.11.3.1 acltdtAllocBuf

Atlas 训练系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

申请共享Buffer内存。

约束说明

- 使用acltdtAllocBuf接口申请内存后，数据区的长度为size参数的大小，在用户还未填入有效数据前，该内存的有效数据长度初始值为0，可在用户向内存中填入有效数据后，再通过[acltdtSetBufDataLen](#)接口设置有效数据长度。
- 使用acltdtAllocBuf接口申请的内存，需要通过[acltdtFreeBuf](#)接口释放内存。
- 在Atlas 200/300/500 推理产品上，仅支持在以下模式下使用该接口：
RC模式

- 在Atlas 200/500 A2推理产品，仅支持在RC模式下使用该接口。

函数原型

```
aclError acltdtAllocBuf(size_t size, uint32_t type, acltdtBuf *buf)
```

参数说明

参数名	输入/输出	说明
size	输入	用于指定申请数据区大小，单位Byte，不能超过4G。
type	输入	共享Buffer内存类型，支持设置如下枚举值。 typedef enum { ACL_TDT_NORMAL_MEM = 0, ACL_TDT_DVPP_MEM } acltdtAllocBufType;
buf	输出	申请成功，输出共享Buffer指针。

返回值说明

返回0表示成功，返回其他值表示失败。

参考资源

接口调用流程及示例，参见共享Buffer管理。

1.11.3.2 acltdtFreeBuf

Atlas 训练系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

释放通过[acltdtAllocBuf](#)接口申请的mbuf。

约束说明

- 在Atlas 200/300/500 推理产品上，仅支持在以下模式下使用该接口：
RC模式
- 在Atlas 200/500 A2推理产品，仅支持在RC模式下使用该接口。

函数原型

```
aclError acltdtFreeBuf(acltdtBuf buf)
```

参数说明

参数名	输入/输出	说明
buf	输入	指定要释放的mbuf。

返回值说明

返回0表示成功，返回其他值表示失败。

参考资源

接口调用流程及示例，参见共享Buffer管理。

1.11.3.3 acltdtGetBufData

Atlas 训练系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

获取共享Buffer的数据区指针和数据区长度，用户可以使用此指针填入数据。

约束说明

- 接口调用顺序：调用[acltdtAllocBuf](#)或[acltdtCopyBufRef](#)接口申请到共享Buffer后，因此需由用户调用[acltdtGetBufData](#)接口获取共享Buffer的内存指针及长度后，再自行向内存中填充有效数据，然后再调用[acltdtSetBufDataLen](#)接口设置共享Buffer中有效数据的长度，且长度必须小于[acltdtGetBufData](#)获取到的size大小。
- 在Atlas 200/300/500 推理产品上，仅支持在以下模式下使用该接口：
RC模式
- 在Atlas 200/500 A2推理产品，仅支持在RC模式下使用该接口。

函数原型

aclError [acltdtGetBufData](#)(const [acltdtBuf](#) buf, void **dataPtr, size_t *size)

参数说明

参数名	输入/输出	说明
buf	输入	共享Buffer指针，须通过 acltdtAllocBuf 接口申请获得。
dataPtr	输出	数据区指针（ device侧地址 ）。
size	输出	数据区的长度，单位为Byte。

返回值说明

返回0表示成功，返回其他值表示失败。

参考资源

接口调用流程及示例，参见共享Buffer管理。

1.11.3.4 acltdtSetBufUserData

Atlas 训练系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

设置共享Buffer的私有数据区数据，从用户内存拷贝到共享内存的私有数据区的指定偏移位置，用于设置控制信息做为上下文传递。

约束说明

- 当前默认私有数据区大小是96Byte，offset+size必须小于或等于96Byte，否则返回报错。
- 在Atlas 200/300/500 推理产品上，仅支持在以下模式下使用该接口：
RC模式
- 在Atlas 200/500 A2推理产品，仅支持在RC模式下使用该接口。

函数原型

aclError acltdtSetBufUserData(**acltdtBuf** buf, const void *dataPtr, size_t size, size_t offset)

参数说明

参数名	输入/输出	说明
buf	输入	共享Buffer指针，须通过 acltdtAllocBuf 或 acltdtCopyBufRef 接口申请获得。
dataPtr	输入	存放用户数据的内存地址指针。
size	输入	用户数据的长度，单位为Byte。 数据长度小于或等于96Byte。
offset	输入	地址偏移，单位为Byte。 偏移量小于或等于96Byte。

返回值说明

返回0表示成功，返回其他值表示失败。

参考资源

接口调用流程及示例，参见共享Buffer管理。

1.11.3.5 acltdtGetBufUserData

Atlas 训练系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

获取共享Buffer的私有数据区数据，偏移offset后并拷贝至用户申请的内存区域。

约束说明

- 当前默认私有数据区大小是96Byte，offset+size必须小于或等于96Byte，否则返回报错。
- 在Atlas 200/300/500 推理产品上，仅支持在以下模式下使用该接口：
RC模式
- 在Atlas 200/500 A2推理产品，仅支持在RC模式下使用该接口。

函数原型

aclError acltdtGetBufUserData(const **acltdtBuf** buf, void *dataPtr, size_t size, size_t offset)

参数说明

参数名	输入/输出	说明
buf	输入	共享Buffer指针，须通过 acltdtAllocBuf 或 acltdtCopyBufRef 接口申请获得。
dataPtr	输入	存放用户数据的内存地址指针。
size	输入	用户数据的长度，单位为Byte。 数据长度小于或等于96Byte。
offset	输入	地址偏移，单位为Byte。 偏移量小于或等于96Byte。

返回值说明

返回0表示成功，返回其他值表示失败。

1.11.3.6 acltdtSetBufDataLen

Atlas 训练系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

设置共享Buffer中有效数据的长度。

约束说明

- 接口调用顺序：调用[acltdtAllocBuf](#)或[acltdtCopyBufRef](#)接口申请到共享Buffer后，因此需由用户调用[acltdtGetBufData](#)接口获取共享Buffer的内存指针及长度后，再自行向内存中填充有效数据。
- 在Atlas 200/300/500 推理产品上，仅支持在以下模式下使用该接口：
RC模式
- 在Atlas 200/500 A2推理产品，仅支持在RC模式下使用该接口。

函数原型

aclError [acltdtSetBufDataLen](#)([acltdtBuf](#) buf, size_t len)

参数说明

参数名	输入/输出	说明
buf	输入	共享Buffer指针，须通过 acltdtAllocBuf 或 acltdtCopyBufRef 接口申请获得。
len	输入	有效数据的长度，单位为Byte。

返回值说明

返回0表示成功，返回其他值表示失败。

参考资源

接口调用流程及示例，参见[共享Buffer管理](#)。

1.11.3.7 [acltdtGetBufDataLen](#)

Atlas 训练系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

获取共享Buffer中有效数据的长度。

约束说明

- 通过[acltdtSetBufDataLen](#)接口设置共享Buffer中有效数据的长度后，可调用本接口获取有效数据的长度，否则，通过本接口获取到的长度为0。
- 在Atlas 200/300/500 推理产品上，仅支持在以下模式下使用该接口：
RC模式
- 在Atlas 200/500 A2推理产品，仅支持在RC模式下使用该接口。

函数原型

```
aclError acltdtGetBufDataLen(acltdtBuf buf, size_t *len)
```

参数说明

参数名	输入/输出	说明
buf	输入	共享Buffer指针，须通过 acltdtAllocBuf 或 acltdtCopyBufRef 接口申请获得。
len	输出	有效数据的长度，单位为Byte。

返回值说明

返回0表示成功，返回其他值表示失败。

1.11.3.8 acltdtCopyBufRef

Atlas 训练系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

对共享Buffer数据区的引用拷贝，创建并返回一个新的Buffer管理结构指向相同的数据区。

约束说明

- 在Atlas 200/300/500 推理产品上，仅支持在以下模式下使用该接口：
RC模式
- 在Atlas 200/500 A2推理产品，仅支持在RC模式下使用该接口。

函数原型

```
aclError acltdtCopyBufRef(const acltdtBuf buf, acltdtBuf *newBuf)
```

参数说明

参数名	输入/输出	说明
buf	输入	共享Buffer指针，须通过 acltdtAllocBuf 或 acltdtCopyBufRef 接口申请获得。
newBuf	输出	返回一个新的共享Buffer指针，指向相同的数据区。

返回值说明

返回0表示成功，返回其他值表示失败。

1.11.3.9 acltdtAppendBufChain

Atlas 训练系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

将某个共享Buffer挂接到headBuf指向的共享Buffer的尾部，headBuf指向的共享Buffer为共享Buffer链头部的第一个共享Buffer，可以是一个单独的共享Buffer（这相当于共享Buffer链只有一个节点）。

约束说明

- 共享Buffer链最大支持128个共享Buffer。
- 在Atlas 200/300/500 推理产品上，仅支持在以下模式下使用该接口：
RC模式
- 在Atlas 200/500 A2推理产品，仅支持在RC模式下使用该接口。

函数原型

aclError [acltdtAppendBufChain](#)([acltdtBuf](#) headBuf, [acltdtBuf](#) buf)

参数说明

参数名	输入/输出	说明
headBuf	输入	共享Buffer链头部的第一个共享Buffer的指针，须通过 acltdtAllocBuf 或 acltdtCopyBufRef 接口申请获得。
buf	输入	待Append的buf指针，须通过 acltdtAllocBuf 或 acltdtCopyBufRef 接口申请获得。

返回值说明

返回0表示成功，返回其他值表示失败。

1.11.3.10 acltdtGetBufChainNum

Atlas 训练系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

获取共享Buffer链中的共享Buffer数量。

约束说明

- 在Atlas 200/300/500 推理产品上，仅支持在以下模式下使用该接口：
RC模式
- 在Atlas 200/500 A2推理产品，仅支持在RC模式下使用该接口。

函数原型

aclError acltdtGetBufChainNum(**acltdtBuf** headBuf, uint32_t *num)

参数说明

参数名	输入/输出	说明
headBuf	输入	共享Buffer链头部的第一个共享Buffer的指针，须通过 acltdtAllocBuf 或 acltdtCopyBufRef 接口申请获得。
num	输出	共享Buffer链中的共享Buffer数量。

返回值说明

返回0表示成功，返回其他值表示失败。

1.11.3.11 acltdtGetBufFromChain

Atlas 训练系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数功能

获取Mbuf链中第index个Mbuf。

约束说明

- 在Atlas 200/300/500 推理产品上，仅支持在以下模式下使用该接口：
RC模式
- 在Atlas 200/500 A2推理产品，仅支持在RC模式下使用该接口。

函数原型

```
aclError acltdtGetBufFromChain(acltdtBuf headBuf, uint32_t index, acltdtBuf *buf)
```

参数说明

参数名	输入/输出	说明
headBuf	输入	共享Buffer链头部的第一个共享Buffer的指针，须通过 acltdtAllocBuf 或 acltdtCopyBufRef 接口申请获得。
index	输入	共享Buffer链中的共享Buffer序号（从0开始计数）。
buf	输出	返回第index个共享Buffer指针。

返回值说明

返回0表示成功，返回其他值表示失败。

1.12 特征向量检索

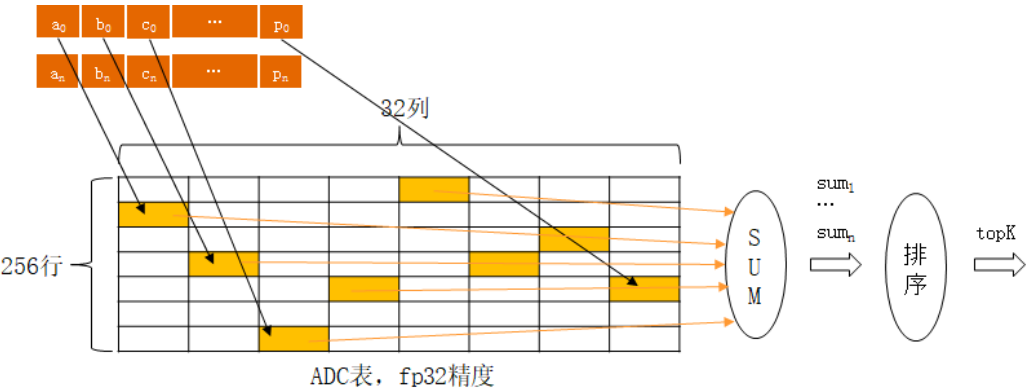
1.12.1 总体说明

在图像检索场景，通常使用深度学习算法处理感兴趣的图像时，提取出高维特征向量；并将一组感兴趣的图像提取得到的特征向量作为底库，待搜索的特征向量与底库中的向量依次进行相似度计算，按相似度排序返回检索结果。为了进一步降低存储和计算的开销，对浮点型长特征向量进行量化处理、转换为短特征再进行检索比对的技术被广泛应用。

昇腾AI处理器支持基于硬件加速的PQ短特征检索，提供特征底库的添加/删除接口，特征向量的修改/删除接口、以及执行检索接口；支持长度为32 Byte的短特征向量，支持设置返回TOP结果的数量，支持1:N和N:M检索模式。用户需要自行准备短特征底库，以及生成待检索向量的ADC表，流程如下：



检索比对过程示意图如下所示：



ADC表是大小为32KByte的查找表，使用时需要将表的内容以无符号字符的格式传入对应接口。

Atlas 推理系列产品的特征检索规格如下表所示：

表 1-99 规格说明

类型	规格说明
特征长度	32字节（因为数据结构管理开销，每条特征的平均内存开销约为36字节）
底库容量	最大5亿条（实际值取决于板卡的内存容量）
检索比对类型	支持1:N检索，支持N:M检索
底库数量	1:N底库最大100万个 N:M底库最大1个
底库容量	1:N每个特征库最大100万条特征 N:M底库最大1000万条特征
检索比对性能	每秒50亿次； 5亿底库下可达到10QPS，时延<1秒
检索返回候选特征数量	2~4800

1.12.2 aclfvlnit

- Atlas 200/300/500 推理产品不支持该接口。
- Atlas 200/500 A2推理产品不支持该接口。
- Atlas 训练系列产品不支持该接口。
- Atlas A2训练系列产品/Atlas 800I A2推理产品不支持该接口。
- Atlas 推理系列加速模块产品不支持该接口。

函数功能

初始化特征检索模块。
与[aclfvRelease](#)接口配套使用。

约束说明

一个进程内只能调用一次[aclfvInit](#)接口。

函数原型

aclError [aclfvInit](#)([aclfvInitPara](#) *initPara)

参数说明

参数名	输入/输出	说明
initPara	输入	特征向量检索初始化参数的指针。 - 调用aclfvCreateInitPara接口创建aclfvInitPara类型的数据。 1:N场景下调用aclfvSet1NTopNum接口设置返回的topK结果数量的最大值，不设置时，默认返回的topK结果数量的最大值为4800。 - N:M场景下调用aclfvSetNMTopNum接口设置返回的topK结果数量的最大值，不设置时，默认返回的topK结果数量的最大值为500。

返回值说明

返回0表示成功，返回其他值表示失败。

参考资源

接口调用流程及示例代码，参见特征向量检索。

1.12.3 [aclfvRelease](#)

Atlas 200/300/500 推理产品不支持该接口。
Atlas 200/500 A2推理产品不支持该接口。
Atlas 训练系列产品不支持该接口。
Atlas A2训练系列产品/Atlas 800I A2推理产品不支持该接口。
Atlas 推理系列加速模块产品不支持该接口。

函数功能

特征检索模块去初始化，释放内存空间，与[aclfvInit](#)接口配套使用。

函数原型

aclError **aclfvRelease()**

参数说明

无

返回值说明

返回0表示成功，返回其他值表示失败。

参考资源

接口调用流程及示例代码，参见特征向量检索。

1.12.4 aclfvRepoAdd

Atlas 200/300/500 推理产品不支持该接口。

Atlas 200/500 A2推理产品不支持该接口。

Atlas 训练系列产品不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品不支持该接口。

Atlas 推理系列加速模块产品不支持该接口。

函数功能

添加底库或向已存在底库中添加特征。该章节描述的1:N表示“检索请求的个数:底库的数量”，N:M用于碰撞两个库的相似性。

约束说明

- 1:N添加时，对于每个库，用户需要保证[aclfvFeatureInfo](#)结构体中的offset连续；N:M添加时，offset不为0会报错。
- 非线程安全，N:M场景不允许多线程同时添加\检索\删除。
- 调用本接口，内部会默认创建一个Stream，用于接口内任务同步执行，在接口执行完成后，该Stream资源会被自动释放。
受硬件限制，Stream总数有限，请参见[1.5.3.1 aclrtCreateStream](#)处的说明。

函数原型

aclError **aclfvRepoAdd(aclfvSearchType type, aclfvFeatureInfo *featureInfo)**

参数说明

参数名	输入/输出	说明
type	输入	检索类型。

参数名	输入/输出	说明
featureInfo	输入	特征描述信息的指针。 需提前调用 aclfvCreateFeatureInfo 接口创建 aclfvFeatureInfo 类型的数据。

返回值说明

返回0表示成功，返回其他值表示失败。

参考资源

接口调用流程及示例代码，参见特征向量检索。

1.12.5 aclfvRepoDel

Atlas 200/300/500 推理产品不支持该接口。

Atlas 200/500 A2推理产品不支持该接口。

Atlas 训练系列产品不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品不支持该接口。

Atlas 推理系列加速模块产品不支持该接口。

函数功能

删除底库。

约束说明

- 非线程安全，N:M场景不允许多线程同时添加\检索\删除。
- 调用本接口，内部会默认创建一个Stream，用于接口内任务同步执行，在接口执行完成后，该Stream资源会被自动释放。
受硬件限制，Stream总数有限，请参见[1.5.3.1 aclrtCreateStream](#)处的说明。

函数原型

aclError **aclfvRepoDel**(**aclfvSearchType** type, **aclfvRepoRange** *repoRange)

参数说明

参数名	输入/输出	说明
type	输入	检索类型。

参数名	输入/输出	说明
repoRange	输入	特征删除范围的指针。 需提前调用 aclfvCreateRepoRange 接口创建 aclfvRepoRange 类型的数据。

返回值说明

返回0表示成功，返回其他值表示失败。

参考资源

接口调用流程及示例代码，参见特征向量检索。

1.12.6 aclfvDel

Atlas 200/300/500 推理产品不支持该接口。

Atlas 200/500 A2推理产品不支持该接口。

Atlas 训练系列产品不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品不支持该接口。

Atlas 推理系列加速模块产品不支持该接口。

函数功能

精确删除底库中某个特征，一次只能删除底库中一个特征。N:M场景下不涉及这个接口。

约束说明

- 调用本接口，内部会默认创建一个Stream，用于接口内任务同步执行，在接口执行完成后，该Stream资源会被自动释放。
受硬件限制，Stream总数有限，请参见[1.5.3.1 aclrtCreateStream](#)处的说明。

函数原型

aclError [aclfvDel](#)([aclfvFeatureInfo](#) *featureInfo)

参数说明

参数名	输入/输出	说明
featureInfo	输入	特征描述信息的指针。 需提前调用 aclfvCreateFeatureInfo 接口创建 aclfvFeatureInfo 类型的数据。

返回值说明

返回0表示成功，返回其他值表示失败。

参考资源

接口调用流程及示例代码，参见特征向量检索。

1.12.7 aclfvModify

Atlas 200/300/500 推理产品不支持该接口。

Atlas 200/500 A2推理产品不支持该接口。

Atlas 训练系列产品不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品不支持该接口。

Atlas 推理系列加速模块产品不支持该接口。

函数功能

修改底库中某个特征，一次只能修改底库中一个特征，不允许修改底库中不存在的特征。N:M场景下不涉及这个接口。

约束说明

- 调用本接口，内部会默认创建一个Stream，用于接口内任务同步执行，在接口执行完成后，该Stream资源会被自动释放。
受硬件限制，Stream总数有限，请参见[1.5.3.1 aclrtCreateStream](#)处的说明。

函数原型

aclError **aclfvModify**(**aclfvFeatureInfo** *featureInfo)

参数说明

参数名	输入/输出	说明
featureInfo	输入	特征描述信息的指针。 需提前调用 aclfvCreateFeatureInfo 接口创建 aclfvFeatureInfo 类型的数据。

返回值说明

返回0表示成功，返回其他值表示失败。

参考资源

接口调用流程及示例代码，参见特征向量检索。

1.12.8 aclfvSearch

Atlas 200/300/500 推理产品不支持该接口。

Atlas 200/500 A2推理产品不支持该接口。

Atlas 训练系列产品不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品不支持该接口。

Atlas 推理系列加速模块产品不支持该接口。

函数功能

特征1:N或N:M检索。

约束说明

- 非线程安全，N:M场景不允许多线程同时添加\检索\删除。
- N:M场景或1:N场景，不添加底库，调用aclfvSearch接口检索时，返回结果数量为0。
- 调用本接口，内部会默认创建一个Stream，用于接口内任务同步执行，在接口执行完成后，该Stream资源会被自动释放。
受硬件限制，Stream总数有限，请参见[1.5.3.1 aclrtCreateStream](#)处的说明。

函数原型

aclError **aclfvSearch**(**aclfvSearchType** type, **aclfvSearchInput** *searchInput, **aclfvSearchResult** *searchRst)

参数说明

参数名	输入/输出	说明
type	输入	特征检索场景类型。
searchInput	输入	检索输入信息的指针。 需提前调用 aclfvCreateSearchInput 接口创建 aclfvSearchInput 类型的数据。

参数名	输入/输出	说明
searchRst	输出	检索输出结果的指针。 需提前调用aclfvCreateSearchResult接口创建aclfvSearchResult类型的数据，该数据中queryCnt参数应与检索输入信息searchInput中的queryCnt参数相同。 检索输出返回的结果数量通过aclfvCreateSearchInput接口的topk参数设置，但与调用aclfvInit接口进行初始化时设置的topK结果数量最大值有关，若调用aclfvCreateSearchInput接口设置的topk参数值大于或等于初始化时设置的topK结果数量最大值，则在返回检索输出结果时，按初始化时设置的topK结果数量最大值返回检索结果；若调用aclfvCreateSearchInput接口设置的topk参数值小于初始化时设置的topK结果数量最大值，则在返回检索输出结果时，按实际通过aclfvCreateSearchInput接口设置的topk参数值返回检索结果。

返回值说明

返回0表示成功，返回其他值表示失败。

参考资源

接口调用流程及示例代码，参见特征向量检索。

1.13 数据类型及其操作接口

1.13.1 aclError

typedef int aclError;

📖 说明

返回码定义规则：

- 规则1：开发人员的环境异常或者代码逻辑错误，可以通过优化环境或代码逻辑的方式解决问题，此时返回码定义为1XXXXX。
- 规则2：资源不足（Stream、内存等）、开发人员编程时使用的接口或参数与当前硬件不匹配，可以通过在编程时合理使用资源的方式解决，此时返回码定义为2XXXXX。
- 规则3：业务功能异常，比如队列满、队列空等，此时返回码定义为3XXXXX。
- 规则4：软硬件内部异常，包括软件内部错误、Device执行失败等，用户无法解决问题，需要将问题反馈给技术支持，此时返回码定义为5XXXXX。您可以获取日志后单击[Link](#)联系技术支持。
- 规则5：无法识别的错误，当前都映射为500000。您可以获取日志后单击[Link](#)联系技术支持。

表 1-100 返回码列表

返回码	含义	可能原因及解决方法
static const int ACL_SUCCESS = 0;	执行成功。	-
static const int ACL_ERROR_NONE = 0; 须知 此返回码后续版本会废弃， 请使用ACL_SUCCESS返回 码。	执行成功。	-
static const int ACL_ERROR_INVALID_PA RAM = 100000;	参数校验失败。	请检查接口的入参值是否 正确。
static const int ACL_ERROR_UNINITIALIZ E = 100001;	未初始化。	• 请检查是否已调用 aclInit接口进行初始 化，请确保已调用 aclInit接口，且在其它 AscendCL接口之前调 用。 • 请检查是否已调用对应 功能的初始化接口，例 如初始化Dump的 aclmdlInitDump接 口、初始化Profiling的 aclprofInit接口。
static const int ACL_ERROR_REPEAT_INI TIALIZE = 100002;	重复初始化或重复加载。	请检查是否调用对应的接 口重复初始化或重复加 载。
static const int ACL_ERROR_INVALID_FIL E = 100003;	无效的文件。	请检查文件是否存在、文 件是否能被访问等。
static const int ACL_ERROR_WRITE_FILE = 100004;	写文件失败。	请检查文件路径是否存 在、文件是否有写权限 等。
static const int ACL_ERROR_INVALID_FIL E_SIZE = 100005;	无效的文件大小。	请检查文件大小是否符合 接口要求。
static const int ACL_ERROR_PARSE_FILE = 100006;	解析文件失败。	请检查文件内容是否合 法。
static const int ACL_ERROR_FILE_MISSIN G_ATTR = 100007;	文件缺失参数。	请检查文件内容是否完 整。

返回码	含义	可能原因及解决方法
static const int ACL_ERROR_FILE_ATTR_I NVALID = 100008;	文件参数无效。	请检查文件中参数值是否正确。
static const int ACL_ERROR_INVALID_D UMP_CONFIG = 100009;	无效的Dump配置。	请检查Dump配置是否正确，详细配置请参见《 精度调试工具使用指南 》。
static const int ACL_ERROR_INVALID_PR OFILING_CONFIG = 100010;	无效的Profiling配置。	请检查Profiling配置是否正确。
static const int ACL_ERROR_INVALID_M ODEL_ID = 100011;	无效的模型ID。	请检查模型ID是否正确、模型是否正确加载。
static const int ACL_ERROR_DESERIALIZ E_MODEL = 100012;	反序列化模型失败。	模型可能与当前版本不匹配，请重新构建模型。
static const int ACL_ERROR_PARSE_MO DEL = 100013;	解析模型失败。	模型可能与当前版本不匹配，请重新构建模型。
static const int ACL_ERROR_READ_MOD EL_FAILURE = 100014;	读取模型失败。	请检查模型文件是否存在、模型文件是否能被访问等。
static const int ACL_ERROR_MODEL_SIZ E_INVALID = 100015;	无效的模型大小。	模型文件无效，请重新构建模型。
static const int ACL_ERROR_MODEL_MIS SING_ATTR = 100016;	模型缺少参数。	模型可能与当前版本不匹配，请重新构建模型。
static const int ACL_ERROR_MODEL_INP UT_NOT_MATCH = 100017;	模型的输入不匹配。	请检查模型的输入是否正确。
static const int ACL_ERROR_MODEL_OU TPUT_NOT_MATCH = 100018;	模型的输出不匹配。	请检查模型的输出是否正确。
static const int ACL_ERROR_MODEL_NO T_DYNAMIC = 100019;	非动态模型。	请检查当前模型是否支持动态场景，如不支持，请重新构建模型。
static const int ACL_ERROR_OP_TYPE_N OT_MATCH = 100020;	单算子类型不匹配。	请检查算子类型是否正确。

返回码	含义	可能原因及解决方法
static const int ACL_ERROR_OP_INPUT_ NOT_MATCH = 100021;	单算子的输入不匹配。	请检查算子的输入是否正确。
static const int ACL_ERROR_OP_OUTPU T_NOT_MATCH = 100022;	单算子的输出不匹配。	请检查算子的输出是否正确。
static const int ACL_ERROR_OP_ATTR_N OT_MATCH = 100023;	单算子的属性不匹配。	请检查算子的属性是否正确。
static const int ACL_ERROR_OP_NOT_FO UND = 100024;	单算子未找到。	请检查算子类型是否支持。
static const int ACL_ERROR_OP_LOAD_F AILED = 100025;	单算子加载失败。	模型可能与当前版本不匹配，请重新构建单算子模型。
static const int ACL_ERROR_UNSUPPOR TED_DATA_TYPE = 100026;	不支持的数据类型。	请检查数据类型是否存在或当前是否支持。
static const int ACL_ERROR_FORMAT_N OT_MATCH = 100027;	Format不匹配。	请检查Format是否正确。
static const int ACL_ERROR_BIN_SELECT OR_NOT_REGISTERED = 100028;	使用二进制选择方式编译算子接口时，算子未注册选择器。	请检查是否调用 aclopRegisterSelectKernelFunc 接口注册算子选择器。
static const int ACL_ERROR_KERNEL_NO T_FOUND = 100029;	编译算子时，算子Kernel未注册。	请检查是否调用 aclopCreateKernel 接口注册算子Kernel。
static const int ACL_ERROR_BIN_SELECT OR_ALREADY_REGISTERE D = 100030;	使用二进制选择方式编译算子接口时，算子重复注册。	请检查是否重复调用 aclopRegisterSelectKernelFunc 接口注册算子选择器。
static const int ACL_ERROR_KERNEL_AL READY_REGISTERED = 100031;	编译算子时，算子Kernel重复注册。	请检查是否重复调用 aclopCreateKernel 接口注册算子Kernel。
static const int ACL_ERROR_INVALID_Q UEUE_ID = 100032;	无效的队列ID。	请检查队列ID是否正确。

返回码	含义	可能原因及解决方法
static const int ACL_ERROR_REPEAT_SUBSCRIBE = 100033;	重复订阅。	请检查针对同一个Stream，是否重复调用 aclrtSubscribeReport 接口。
static const int ACL_ERROR_STREAM_NOT_SUBSCRIBE = 100034; 须知 此返回码后续版本会废弃，请使用 ACL_ERROR_RT_STREAM_NO_CB_REG 返回码。	Stream未订阅。	请检查是否已调用 aclrtSubscribeReport 接口。
static const int ACL_ERROR_THREAD_NOT_SUBSCRIBE = 100035; 须知 此返回码后续版本会废弃，请使用 ACL_ERROR_RT_THREAD_SUBSCRIBE 返回码。	线程未订阅。	请检查是否已调用 aclrtSubscribeReport 接口。
static const int ACL_ERROR_WAIT_CALLBACK_TIMEOUT = 100036; 须知 此返回码后续版本会废弃，请使用 ACL_ERROR_RT_REPORT_TIMEOUT 返回码。	等待callback超时。	请检查是否已调用 aclrtLaunchCallback 接口下发callback任务； 请检查 aclrtProcessReport 接口中超时时间是否合理； 请检查callback任务是否已经处理完成，如果已处理完成，但还调用 aclrtProcessReport 接口，则需优化代码逻辑。
static const int ACL_ERROR_REPEAT_FINALIZE = 100037;	重复去初始化。	请检查是否重复调用 aclFinalize 接口进行去初始化。
static const int ACL_ERROR_NOT_STATIC_AIPP = 100038; 须知 此返回码后续版本会废弃，请使用 ACL_ERROR_GE_AIPP_NOT_EXIST 返回码。	静态AIPP配置信息不存在。	调用 aclmdlGetFirstAippInfo 接口时，请传入正确的index值。
static const int ACL_ERROR_COMPILING_STUB_MODE = 100039;	运行应用前配置的动态库路径是编译桩的路径，不是正确的动态库路径。	请检查动态库路径的配置，确保使用运行模式的动态库。

返回码	含义	可能原因及解决方法
static const int ACL_ERROR_GROUP_NOT_SET = 100040; 须知 此返回码后续版本会废弃， 请使用 ACL_ERROR_RT_GROUP_NOT_SET 返回码。	未设置Group。	请检查是否已调用 aclrtSetGroup 接口。
static const int ACL_ERROR_GROUP_NOT_CREATE = 100041; 须知 此返回码后续版本会废弃， 请使用 ACL_ERROR_RT_GROUP_NOT_CREATE 返回码。	未创建对应的Group。	请检查调用接口时设置的 Group ID是否在支持的范围内， Group ID的取值范围： [0, (Group数量-1)]， 用户可调用 aclrtGetGroupCount 接口 获取Group数量。
static const int ACL_ERROR_PROF_ALREADY_RUN = 100042;	已存在采集Profiling数据的任务。	- 请检查代码逻辑，“通过调用AscendCL API方式采集Profiling数据”的配置不能与其它方式的Profiling配置并存，只能保留一种，各种方式的Profiling采集配置请参见《性能调优工具使用指南》。 - 请检查是否对同一个Device重复下发了多次Profiling配置，请参考1.10.1 Profiling AscendCL API (通过Profiling AscendCL API采集并落盘性能数据)中的接口调用要求、接口调用顺序调整代码逻辑。
static const int ACL_ERROR_PROF_NOT_RUN = 100043;	未使用 aclprofInit 接口先进行Profiling初始化。	请检查接口调用顺序，参考 1.10.1 Profiling AscendCL API (通过Profiling AscendCL API采集并落盘性能数据) 中的说明。

返回码	含义	可能原因及解决方法
static const int ACL_ERROR_DUMP_ALR EADY_RUN = 100044;	已存在获取Dump数据的任务。	请检查在调用 aclmdlInitDump 接口、 aclmdlSetDump 接口、 aclmdlFinalizeDump 接口配置Dump信息前，是否已调用 aclInit 接口配置Dump信息，如是，请调整代码逻辑，保留一种方式配置Dump信息即可。
static const int ACL_ERROR_DUMP_NOT _RUN = 100045;	未使用 aclmdlInitDump 接口先进行Dump初始化。	请检查获取Dump数据的接口调用顺序，参考 aclmdlInitDump 接口处的说明。
static const int ACL_ERROR_PROF_REPE AT_SUBSCRIBE = 148046;	重复订阅同一个模型。	请检查接口调用顺序，参考 1.10.1 Profiling AscendCL API (通过 Profiling AscendCL API 采集并落盘性能数据) 中的说明。
static const int ACL_ERROR_PROF_API_C ONFLICT = 148047;	采集性能数据的接口调用冲突。	两种方式的Profiling性能数据采集接口不能交叉调用， aclprofInit 接口和 aclprofFinalize 接口之间不能调用 aclprofModelSubscribe 接口、 aclprofGet* 接口、 aclprofModelUnSubscribe 接口， aclprofModelSubscribe 接口和 aclprofModelUnSubscribe 接口之间不能调用 aclprofInit 接口、 aclprofStart 接口、 aclprofStop 接口、 aclprofFinalize 。
static const int ACL_ERROR_INVALID_M AX_OPQUEUE_NUM_CO NFIG = 148048;	无效的算子缓存信息老化配置。	请检查算子缓存信息老化配置，参考 aclInit 处的配置说明及示例。
static const int ACL_ERROR_INVALID_OP P_PATH = 148049;	没有设置ASCEND_OPP_PATH环境变量，或该环境变量的值设置错误。	请检查是否设置ASCEND_OPP_PATH环境变量，且该环境变量的值是否为opp软件包的安装路径。

返回码	含义	可能原因及解决方法
static const int ACL_ERROR_OP_UNSUPPORTED_DYNAMIC = 148050;	算子不支持动态Shape。	• 请检查单算子模型文件中该算子的Shape是否为动态，如果是动态的，需要修改为固定Shape。 • 请检查编译算子时，aclTensorDesc的Shape是否为动态，如果是动态的，需要按照固定Shape重新创建aclTensorDesc。
static const int ACL_ERROR_RELATIVE_RESOURCE_NOT_CLEARED = 148051;	相关的资源尚未释放。	在销毁通道描述信息时，如果相关的通道尚未销毁则返回此错误码。请检查与此通道描述信息相关联的通道是否被销毁。
static const int ACL_ERROR_UNSUPPORTED_JPEG = 148052;	JPEGD功能不支持的输入图片编码格式（例如算术编码、渐进式编码等）。	实现JPEGD图片解码功能时，仅支持Huffman编码，压缩前的原图像色彩空间为YUV，像素的各分量比例为4:4:4或4:2:2或4:2:0或4:0:0或4:4:0，不支持算术编码、不支持渐进JPEG格式、不支持JPEG2000格式。
static const int ACL_ERROR_INVALID_BUNDLE_MODEL_ID = 148053;	无效的模型ID。	请检查模型ID是否正确、模型是否正确加载。
static const int ACL_ERROR_BAD_ALLOC = 200000;	申请内存失败。	请检查硬件环境上的内存剩余情况。
static const int ACL_ERROR_API_NOT_SUPPORTED = 200001;	接口不支持。	请检查调用的接口当前是否支持。
static const int ACL_ERROR_INVALID_DEVICE = 200002; 须知 此返回码后续版本会废弃，请使用 ACL_ERROR_RT_INVALID_DEVICEID 返回码。	无效的Device。	请检查Device是否存在。

返回码	含义	可能原因及解决方法
static const int ACL_ERROR_MEMORY_ADDRESS_UNALIGNED = 200003;	内存地址未对齐。	请检查内存地址是否符合接口要求。
static const int ACL_ERROR_RESOURCE_NOT_MATCH = 200004;	资源不匹配。	请检查调用接口时，是否传入正确的Stream、Context等资源。
static const int ACL_ERROR_INVALID_RESOURCE_HANDLE = 200005;	无效的资源句柄。	请检查调用接口时，传入的Stream、Context等资源是否已被销毁或占用。
static const int ACL_ERROR_FEATURE_UNSUPPORTED = 200006;	特性不支持。	您可以获取日志后单击 Link 联系技术支持。
static ACL_ERROR_PROF_MODULES_UNSUPPORTED = 200007;	下发了不支持的Profiling配置。	请参见 1.13.95.1 aclprofCreateConfig 中的说明检查Profiling的配置是否正确。
static const int ACL_ERROR_STORAGE_OVERFLOW_LIMIT = 300000;	超出存储上限。	请检查硬件环境上的存储剩余情况。
static const int ACL_ERROR_INTERNAL_ERROR = 500000;	未知内部错误。	您可以获取日志后单击 Link 联系技术支持。
static const int ACL_ERROR_FAILURE = 500001;	系统内部AscendCL的错误。	您可以获取日志后单击 Link 联系技术支持。
static const int ACL_ERROR_GE_FAILURE = 500002;	系统内部GE的错误。	您可以获取日志后单击 Link 联系技术支持。
static const int ACL_ERROR_RT_FAILURE = 500003;	系统内部RUNTIME的错误。	您可以获取日志后单击 Link 联系技术支持。
static const int ACL_ERROR_DRV_FAILURE = 500004;	系统内部DRV (Driver) 的错误。	您可以获取日志后单击 Link 联系技术支持。
static const int ACL_ERROR_PROFILING_FAILURE = 500005;	Profiling相关错误。	您可以获取日志后单击 Link 联系技术支持。

表 1-101 RUNTIME 的返回码列表

返回码	含义	可能原因及解决方法
static const int32_t ACL_RT_SUCCESS = 0;	执行成功。	-
static const int32_t ACL_ERROR_RT_PARAM_Invalid = 107000;	参数校验失败。	请检查接口入参是否正确。
static const int32_t ACL_ERROR_RT_INVALID_DEVICEID = 107001;	无效的Device ID。	请检查Device ID是否合法。
static const int32_t ACL_ERROR_RT_CONTEXT_NULL = 107002;	context为空。	请检查是否调用 aclrtSetCurrentContext 或 aclrtSetDevice 。
static const int32_t ACL_ERROR_RT_STREAM_CONTEXT = 107003;	stream不在当前context内。	请检查stream所在的context与当前context是否一致。
static const int32_t ACL_ERROR_RT_MODEL_CONTEXT = 107004;	model不在当前context内。	请检查加载的模型与当前context是否一致。
static const int32_t ACL_ERROR_RT_STREAM_MODEL = 107005;	stream不在当前model内。	请检查stream是否绑定过该模型。
static const int32_t ACL_ERROR_RT_EVENT_TIMESTAMP_INVALID = 107006;	event时间戳无效。	请检查event是否创建。
static const int32_t ACL_ERROR_RT_EVENT_TIMESTAMP_REVERSAL = 107007;	event时间戳反转。	请检查event是否创建。
static const int32_t ACL_ERROR_RT_ADDR_UNALIGNED = 107008;	内存地址未对齐。	请检查所申请的内存地址是否对齐，详细内存申请接口的约束请参见 1.5.5 内存管理 。
static const int32_t ACL_ERROR_RT_FILE_OPEN = 107009;	打开文件失败。	请检查文件是否存在。
static const int32_t ACL_ERROR_RT_FILE_WRITE = 107010;	写文件失败。	请检查文件是否存在或者是否具备写权限。
static const int32_t ACL_ERROR_RT_STREAM_SUBSCRIBE = 107011;	stream未订阅或重复订阅。	请检查当前stream是否订阅或重复订阅。

返回码	含义	可能原因及解决方法
static const int32_t ACL_ERROR_RT_THREAD_ SUBSCRIBE = 107012;	线程未订阅或重复订阅。	请检查当前线程是否订阅 或重复订阅。
static const int32_t ACL_ERROR_RT_GROUP_ NOT_SET = 107013;	未设置Group。	请检查是否已调用 aclrtSetGroup 接口。
static const int32_t ACL_ERROR_RT_GROUP_ NOT_CREATE = 107014;	未创建对应的Group。	请检查调用接口时设置的 Group ID是否在支持的范 围内，Group ID的取值范 围：[0, (Group数量-1)]， 用户可调用 aclrtGetGroupCount 接 口获取Group数量。
static const int32_t ACL_ERROR_RT_STREAM_ NO_CB_REG = 107015;	该callback对应的stream 未注册到线程。	请检查stream是否已经注 册到线程，检查是否调用 aclrtSubscribeReport 接 口。
static const int32_t ACL_ERROR_RT_INVALID_ MEMORY_TYPE = 107016;	无效的内存类型。	请检查内存类型是否合 法。
static const int32_t ACL_ERROR_RT_INVALID_ HANDLE = 107017;	无效的资源句柄。	检查对应输入和使用的参 数是否正确。
static const int32_t ACL_ERROR_RT_INVALID_ MALLOC_TYPE = 107018;	申请使用的内存类型不正 确。	检查对应输入和使用的内 存类型是否正确。
static const int32_t ACL_ERROR_RT_WAIT_TI MEOUT = 107019;	等待超时。	请尝试重新执行下发任务 的接口。
static const int32_t ACL_ERROR_RT_TASK_TI MEOUT = 107020;	执行任务超时。	请排查业务编排是否合理 或者设置合理的超时时 间。
static const int32_t ACL_ERROR_RT_SYSPARA MOPT_NOT_SET = 107021;	获取当前Context中的系统 参数值失败。	未设置当前Context中的系 统参数值，请参见 1.5.2.5 aclrtCtxSetSysParamOp t 。

返回码	含义	可能原因及解决方法
static const int32_t ACL_ERROR_RT_DEVICE_ TASK_ABORT = 107022;	Device上的任务丢弃操作 与其它操作冲突。	该错误码是由于调用 aclrtDeviceTaskAbort 接 口停止Device上的任务与 其它接口操作冲突，用户 需排查代码逻辑，等待 aclrtDeviceTaskAbort接口 执行完成后，才执行其它 操作。
static const int32_t ACL_ERROR_RT_FEATUR E_NOT_SUPPORT = 207000;	特性不支持。	您可以获取日志后单击 Link 联系技术支持。
static const int32_t ACL_ERROR_RT_MEMOR Y_ALLOCATION = 207001;	内存申请失败。	请检查硬件环境上的存储 剩余情况。
static const int32_t ACL_ERROR_RT_MEMOR Y_FREE = 207002;	内存释放失败。	您可以获取日志后单击 Link 联系技术支持。
static const int32_t ACL_ERROR_RT_AICORE_ OVER_FLOW = 207003;	aicore算子运算溢出。	请检查对应的aicore算子 运算是否有溢出，可以根 据dump数据找到对应溢 出的算子，再定位具体的 算子问题。
static const int32_t ACL_ERROR_RT_NO_DEV ICE = 207004;	Device不可用。	请检查Device是否正常运 行。
static const int32_t ACL_ERROR_RT_RESOUR CE_ALLOC_FAIL = 207005;	内存申请失败。	请检查硬件环境上的存储 剩余情况。
static const int32_t ACL_ERROR_RT_NO_PER MISSION = 207006;	没有操作权限。	请检查运行应用的用户权 限是否正确。
static const int32_t ACL_ERROR_RT_NO_EVE NT_RESOURCE = 207007;	Event资源不足。	请参考 1.5.4.1 aclrtCreateEvent 接口处 的说明检查Event数量是 否符合要求。
static const int32_t ACL_ERROR_RT_NO_STR EAM_RESOURCE = 207008;	Stream资源不足。	请参考 1.5.3.1 aclrtCreateStream 接口 处的说明检查Stream数 量是否符合要求。

返回码	含义	可能原因及解决方法
static const int32_t ACL_ERROR_RT_NO_NOTIFY_RESOURCE = 207009;	系统内部Notify资源不足。	媒体数据处理的并发任务太多或模型推理时消耗资源太多，建议尝试减少并发任务或卸载部分模型。
static const int32_t ACL_ERROR_RT_NO_MODEL_RESOURCE = 207010;	模型资源不足。	建议卸载部分模型。
static const int32_t ACL_ERROR_RT_NO_CDQ_RESOURCE = 207011;	Runtime内部资源不足。	您可以获取日志后单击 Link 联系技术支持。
static const int32_t ACL_ERROR_RT_OVER_LIMIT = 207012;	队列数目超出上限。	请销毁不需要的队列之后再创建新的队列。
static const int32_t ACL_ERROR_RT_QUEUE_EMPTY = 207013;	队列为空。	不能从空队列中获取数据，请先向队列中添加数据，再获取。
static const int32_t ACL_ERROR_RT_QUEUE_FULL = 207014;	队列已满。	不能向已满的队列中添加数据，请先从队列中获取数据，再添加。
static const int32_t ACL_ERROR_RT_REPEAT_INIT = 207015;	队列重复初始化。	建议初始化一次队列即可，不要重复初始化。
static const int32_t ACL_ERROR_RT_DEVICE_OOM = 207018;	Device侧内存耗尽。	排查Device上的内存使用情况，并根据Device上的内存规格合理规划内存的使用。
static const int32_t ACL_ERROR_RT_INTERNAL_ERROR = 507000;	runtime模块内部错误。	您可以获取日志后单击 Link 联系技术支持。
static const int32_t ACL_ERROR_RT_TASK_SCHEDULER_ERROR = 507001;	Device上的task scheduler模块内部错误。	您可以获取日志后单击 Link 联系技术支持。
static const int32_t ACL_ERROR_RT_STREAM_TASK_FULL = 507002;	stream上的task数量满。	您可以获取日志后单击 Link 联系技术支持。
static const int32_t ACL_ERROR_RT_STREAM_TASK_EMPTY = 507003;	stream上的task数量为空。	您可以获取日志后单击 Link 联系技术支持。

返回码	含义	可能原因及解决方法
static const int32_t ACL_ERROR_RT_STREAM_ _NOT_COMPLETE = 507004;	stream上的task未全部执 行完成。	您可以获取日志后单击 Link 联系技术支持。
static const int32_t ACL_ERROR_RT_END_OF _SEQUENCE = 507005;	AI CPU上的task执行完 成。	您可以获取日志后单击 Link 联系技术支持。
static const int32_t ACL_ERROR_RT_EVENT_ _NOT_COMPLETE = 507006;	event未完成。	您可以获取日志后单击 Link 联系技术支持。
static const int32_t ACL_ERROR_RT_CONTEX T_RELEASE_ERROR = 507007;	context释放失败。	您可以获取日志后单击 Link 联系技术支持。
static const int32_t ACL_ERROR_RT_SOC_VE RSION = 507008;	获取soc version失败。	您可以获取日志后单击 Link 联系技术支持。
static const int32_t ACL_ERROR_RT_TASK_TY PE_NOT_SUPPORT = 507009;	不支持的task类型。	您可以获取日志后单击 Link 联系技术支持。
static const int32_t ACL_ERROR_RT_LOST_H EARTBEAT = 507010;	task scheduler丢失心跳。	您可以获取日志后单击 Link 联系技术支持。
static const int32_t ACL_ERROR_RT_MODEL_ EXECUTE = 507011;	模型执行失败。	您可以获取日志后单击 Link 联系技术支持。
static const int32_t ACL_ERROR_RT_REPORT _TIMEOUT = 507012;	获取task scheduler的消息 失败。	排查接口的超时时间设置 是否过短，适当增长超时 时间。如果增长超时时间 后，依然有超时报错，再 排查日志。 您可以获取日志后单击 Link 联系技术支持。
static const int32_t ACL_ERROR_RT_SYS_DM A = 507013;	system dma (Direct Memory Access) 硬件执 行错误。	您可以获取日志后单击 Link 联系技术支持。
static const int32_t ACL_ERROR_RT_AICORE_ TIMEOUT = 507014;	aicore执行超时。	您可以获取日志后单击 Link 联系技术支持。 日志的详细介绍，请参见 《 日志参考 》。

返回码	含义	可能原因及解决方法
static const int32_t ACL_ERROR_RT_AICORE_ EXCEPTION = 507015;	aicore执行异常。	您可以获取日志后单击 Link 联系技术支持。
static const int32_t ACL_ERROR_RT_AICORE_ TRAP_EXCEPTION = 507016;	aicore trap执行异常。	您可以获取日志后单击 Link 联系技术支持。
static const int32_t ACL_ERROR_RT_AICPU_T IMEOUT = 507017;	AI CPU执行超时。	您可以获取日志后单击 Link 联系技术支持。
static const int32_t ACL_ERROR_RT_AICPU_E XCEPTION = 507018;	AI CPU执行异常。	您可以获取日志后单击 Link 联系技术支持。
static const int32_t ACL_ERROR_RT_AICPU_D ATADUMP_RSP_ERR = 507019;	AI CPU执行数据dump后 未给task scheduler返回响 应。	您可以获取日志后单击 Link 联系技术支持。
static const int32_t ACL_ERROR_RT_AICPU_ MODEL_RSP_ERR = 507020;	AI CPU执行模型后未给 task scheduler返回响应。	您可以获取日志后单击 Link 联系技术支持。
static const int32_t ACL_ERROR_RT_PROFI LING_ERROR = 507021;	profiling功能执行异常。	您可以获取日志后单击 Link 联系技术支持。
static const int32_t ACL_ERROR_RT_IPC_ERR OR = 507022;	进程间通信异常。	您可以获取日志后单击 Link 联系技术支持。
static const int32_t ACL_ERROR_RT_MODEL_ ABORT_NORMAL = 507023;	模型退出。	您可以获取日志后单击 Link 联系技术支持。
static const int32_t ACL_ERROR_RT_KERNEL_ UNREGISTERING = 507024;	算子正在去注册。	您可以获取日志后单击 Link 联系技术支持。
static const int32_t ACL_ERROR_RT_RINGBU FFER_NOT_INIT = 507025;	ringbuffer (环形缓冲区) 功能未初始化。	您可以获取日志后单击 Link 联系技术支持。
static const int32_t ACL_ERROR_RT_RINGBU FFER_NO_DATA = 507026;	ringbuffer (环形缓冲区) 没有数据。	您可以获取日志后单击 Link 联系技术支持。

返回码	含义	可能原因及解决方法
static const int32_t ACL_ERROR_RT_KERNEL_ LOOKUP = 507027;	RUNTIME内部的kernel未注册。	您可以获取日志后单击 Link 联系技术支持。
static const int32_t ACL_ERROR_RT_KERNEL_ DUPLICATE = 507028;	重复注册RUNTIME内部的kernel。	您可以获取日志后单击 Link 联系技术支持。
static const int32_t ACL_ERROR_RT_DEBUG_ REGISTER_FAIL = 507029;	debug功能注册失败。	您可以获取日志后单击 Link 联系技术支持。
static const int32_t ACL_ERROR_RT_DEBUG_ UNREGISTER_FAIL = 507030;	debug功能去注册失败。	您可以获取日志后单击 Link 联系技术支持。
static const int32_t ACL_ERROR_RT_LABEL_C ONTEXT = 507031;	标签不在当前context内。	您可以获取日志后单击 Link 联系技术支持。
static const int32_t ACL_ERROR_RT_PROGRA M_USE_OUT = 507032;	注册的program数量超过限制。	您可以获取日志后单击 Link 联系技术支持。
static const int32_t ACL_ERROR_RT_DEV_SET UP_ERROR = 507033;	Device启动失败。	您可以获取日志后单击 Link 联系技术支持。
static const int32_t ACL_ERROR_RT_VECTOR_ _CORE_TIMEOUT = 507034;	vector core执行超时。	您可以获取日志后单击 Link 联系技术支持。
static const int32_t ACL_ERROR_RT_VECTOR_ _CORE_EXCEPTION = 507035;	vector core执行异常。	您可以获取日志后单击 Link 联系技术支持。
static const int32_t ACL_ERROR_RT_VECTOR_ _CORE_TRAP_EXCEPTIO N = 507036;	vector core trap执行异常。	您可以获取日志后单击 Link 联系技术支持。
static const int32_t ACL_ERROR_RT_CDQ_BA TCH_ABNORMAL = 507037;	Runtime内部资源申请异常。	您可以获取日志后单击 Link 联系技术支持。
static const int32_t ACL_ERROR_RT_DIE_MO DE_CHANGE_ERROR = 507038;	die模式修改异常，不能修改die模式。	您可以获取日志后单击 Link 联系技术支持。

返回码	含义	可能原因及解决方法
static const int32_t ACL_ERROR_RT_DIE_SET _ERROR = 507039;	单die模式不能指定die。	您可以获取日志后单击 Link 联系技术支持。
static const int32_t ACL_ERROR_RT_INVALID _DIEID = 507040;	指定die id错误。	您可以获取日志后单击 Link 联系技术支持。
static const int32_t ACL_ERROR_RT_DIE_MO DE_NOT_SET = 507041;	die模式没有设置。	您可以获取日志后单击 Link 联系技术支持。
static const int32_t ACL_ERROR_RT_AICORE_ TRAP_READ_OVERFLOW = 507042;	aicore trap读越界异常。	您可以获取日志后单击 Link 联系技术支持。
static const int32_t ACL_ERROR_RT_AICORE_ TRAP_WRITE_OVERFLO W = 507043;	aicore trap写越界异常。	您可以获取日志后单击 Link 联系技术支持。
static const int32_t ACL_ERROR_RT_VECTOR_ _CORE_TRAP_READ_OVE RFLOW = 507044;	vector core trap读越界异常。	您可以获取日志后单击 Link 联系技术支持。
static const int32_t ACL_ERROR_RT_VECTOR_ _CORE_TRAP_WRITE_OV ERFLOW = 507045;	vector core trap写越界异常。	您可以获取日志后单击 Link 联系技术支持。
static const int32_t ACL_ERROR_RT_STREAM _SYNC_TIMEOUT = 507046;	在指定的超时等待事件 中，指定的stream中所有 任务还没有执行完成。	您可以获取日志后单击 Link 联系技术支持。
static const int32_t ACL_ERROR_RT_EVENT_S YNC_TIMEOUT = 507047;	在指定的Event同步等待 中，超过指定时间，该 Event还有没有执行完。	您可以获取日志后单击 Link 联系技术支持。
static const int32_t ACL_ERROR_RT_FFTS_PL US_TIMEOUT = 507048;	内部任务执行超时。	您可以获取日志后单击 Link 联系技术支持。
static const int32_t ACL_ERROR_RT_FFTS_PL US_EXCEPTION = 507049;	内部任务执行异常。	您可以获取日志后单击 Link 联系技术支持。
static const int32_t ACL_ERROR_RT_FFTS_PL US_TRAP_EXCEPTION = 507050;	内部任务trap异常。	您可以获取日志后单击 Link 联系技术支持。

返回码	含义	可能原因及解决方法
static const int32_t ACL_ERROR_RT_SEND_M SG = 507051;	数据入队过程中消息发送失败。	您可以获取日志后单击 Link 联系技术支持。
static const int32_t ACL_ERROR_RT_COPY_D ATA = 507052;	数据入队过程中内存拷贝失败。	您可以获取日志后单击 Link 联系技术支持。
static const int32_t ACL_ERROR_RT_DEVICE_ MEM_ERROR = 507053;	出现内存UCE（uncorrect error，指系统硬件不能直接处理恢复内存错误）的错误虚拟地址。	请参见 aclrtGetMemUceInfo 接口中的说明获取并修复内存UCE的错误虚拟地址。
static const int32_t ACL_ERROR_RT_DRV_INT ERNAL_ERROR = 507899;	Driver模块内部错误。	您可以获取日志后单击 Link 联系技术支持。
static const int32_t ACL_ERROR_RT_AICPU_I NTERNAL_ERROR = 507900;	AI CPU模块内部错误。	您可以获取日志后单击 Link 联系技术支持。
static const int32_t ACL_ERROR_RT_SOCKET _CLOSE = 507901;	内部HDC（Host Device Communication）会话链接断开。	您可以获取日志后单击 Link 联系技术支持。
static const int32_t ACL_ERROR_RT_AICPU_I NFO_LOAD_RSP_ERR = 507902;	AI CPU调度处理失败。	您可以获取日志后单击 Link 联系技术支持。

表 1-102 GE 的返回码列表

返回码	含义	可能原因及解决方法
uint32_t ACL_ERROR_GE_PARAM_ INVALID = 145000;	参数校验失败。	请检查接口的入参值是否正确。
uint32_t ACL_ERROR_GE_EXEC_N OT_INIT = 145001;	未初始化。	• 请检查是否已调用aclInit接口进行初始化，请确保已调用aclInit接口，且在其它AscendCL接口之前调用。 • 请检查是否已调用对应功能的初始化接口，例如初始化Dump的aclmdlInitDump接口、初始化Profiling的aclprofInit接口。

返回码	含义	可能原因及解决方法
uint32_t ACL_ERROR_GE_EXEC_M ODEL_PATH_INVALID = 145002;	无效的模型路径。	请检查模型路径是否正确。
uint32_t ACL_ERROR_GE_EXEC_M ODEL_ID_INVALID = 145003;	无效的模型ID。	请检查模型ID是否正确、 模型是否正确加载。
uint32_t ACL_ERROR_GE_EXEC_M ODEL_DATA_SIZE_INVALI D = 145006;	无效的模型大小。	无效的模型大小。
uint32_t ACL_ERROR_GE_EXEC_M ODEL_ADDR_INVALID = 145007;	无效的模型内存地址。	请检查模型地址是否有效。
uint32_t ACL_ERROR_GE_EXEC_M ODEL_QUEUE_ID_INVALI D = 145008;	无效的队列ID。	无效的队列ID。
uint32_t ACL_ERROR_GE_EXEC_LO AD_MODEL_REPEATED = 145009;	重复初始化或重复加载。	请检查是否调用对应的接 口重复初始化或重复加 载。
uint32_t ACL_ERROR_GE_DYNAMI C_INPUT_ADDR_INVALID = 145011;	无效的动态分档输入地 址。	请检查动态分档输入地 址。
uint32_t ACL_ERROR_GE_DYNAMI C_INPUT_LENGTH_INVAL ID = 145012;	无效的动态分档输入长 度。	请检查动态分档输入长 度。
uint32_t ACL_ERROR_GE_DYNAMI C_BATCH_SIZE_INVALID = 145013;	无效的动态分档Batch大 小。	请检查动态分档Batch大 小。
uint32_t ACL_ERROR_GE_AIPP_BA TCH_EMPTY = 145014;	无效的AIPP batch size。	请检查AIPP batch size是 否正确。
uint32_t ACL_ERROR_GE_AIPP_N OT_EXIST = 145015;	AIPP配置不存在。	请检查AIPP是否配置。

返回码	含义	可能原因及解决方法
uint32_t ACL_ERROR_GE_AIPP_MODE_INVALID = 145016;	无效的AIPP模式。	请检查模型转换时配置的AIPP模式是否正确。
uint32_t ACL_ERROR_GE_OP_TASK_TYPE_INVALID = 145017;	无效的任务类型。	请检查算子类型是否正确。
uint32_t ACL_ERROR_GE_OP_KERNEL_TYPE_INVALID = 145018;	无效的算子类型。	请检查算子类型是否正确。
uint32_t ACL_ERROR_GE_PLGMGR_PATH_INVALID = 145019;	无效的so文件，包括so文件的路径层级太深、so文件被误删除等情况。	请检查运行应用前配置的环境变量LD_LIBRARY_PATH是否正确，详细描述请参见编译运行处的操作指导。
uint32_t ACL_ERROR_GE_FORMAT_INVALID = 145020;	无效的format。	请检查Tensor数据的format是否有效。
uint32_t ACL_ERROR_GE_SHAPE_INVALID = 145021;	无效的shape。	请检查Tensor数据的shape是否有效。
uint32_t ACL_ERROR_GE_DATATYPE_INVALID = 145022;	无效的数据类型。	请检查Tensor数据的数据类型是否有效。
uint32_t ACL_ERROR_GE_MEMORY_ALLOCATION = 245000;	申请内存失败。	请检查硬件环境上的内存剩余情况。
uint32_t ACL_ERROR_GE_MEMORY_OPERATE_FAILED = 245001;	内存初始化、内存复制操作失败。	请检查内存地址是否正确、硬件环境上的内存是否足够等。
uint32_t ACL_ERROR_GE_DEVICE_MEMORY_ALLOCATION_FAILED = 245002;	申请Device内存失败。	Device内存已用完，无法继续申请，请释放部分Device内存，再重新尝试。
uint32_t ACL_ERROR_GE_SUBHEALTHY = 345102;	亚健康状态。	设备或进程异常触发的重部署动作完成后的状态为亚健康状态，亚健康状态下可以正常调用相关接口。

返回码	含义	可能原因及解决方法
static const uint32_t ACL_ERROR_GE_USER_R AISE_EXCEPTION = 345103;	用户自定义函数主动抛异常。	用户可根据DataFlowInfo中设置的UserData识别出来哪个输入的数据执行报错了，再根据报错排查问题。
static const uint32_t ACL_ERROR_GE_DATA_N OT_ALIGNED = 345104;	数据未对齐。	若用户自定义函数存在多个输出时，需排查用户代码中是否少设置输出，缺少输出可能会导致数据对齐异常。
uint32_t ACL_ERROR_GE_INTERN AL_ERROR = 545000;	未知内部错误。	您可以获取日志后单击 Link 联系技术支持。
uint32_t ACL_ERROR_GE_LOAD_M ODEL = 545001;	系统内部加载模型失败。	您可以获取日志后单击 Link 联系技术支持。
uint32_t ACL_ERROR_GE_EXEC_LO AD_MODEL_PARTITION_ FAILED = 545002;	系统内部加载模型失败。	您可以获取日志后单击 Link 联系技术支持。
uint32_t ACL_ERROR_GE_EXEC_LO AD_WEIGHT_PARTITION_ _FAILED = 545003;	系统内部加载模型权值失败。	您可以获取日志后单击 Link 联系技术支持。
uint32_t ACL_ERROR_GE_EXEC_LO AD_TASK_PARTITION_FAI LED = 545004;	系统内部加载模型任务失败。	您可以获取日志后单击 Link 联系技术支持。
uint32_t ACL_ERROR_GE_EXEC_LO AD_KERNEL_PARTITION_ FAILED = 545005;	系统内部加载模型算子失败。	您可以获取日志后单击 Link 联系技术支持。
uint32_t ACL_ERROR_GE_EXEC_RE LEASE_MODEL_DATA = 545006;	系统内释放模型空间失败。	您可以获取日志后单击 Link 联系技术支持。
uint32_t ACL_ERROR_GE_COMMA ND_HANDLE = 545007;	系统内命令操作失败。	您可以获取日志后单击 Link 联系技术支持。
uint32_t ACL_ERROR_GE_GET_TE NSOR_INFO = 545008;	系统内获取张量数据失败。	您可以获取日志后单击 Link 联系技术支持。

返回码	含义	可能原因及解决方法
uint32_t ACL_ERROR_GE_UNLOAD_MODEL = 545009;	系统内卸载模型空间失败。	您可以获取日志后单击 Link 联系技术支持。
uint32_t ACL_ERROR_GE_MODEL_EXECUTE_TIMEOUT = 545601;	模型执行超时	您可以获取日志后单击 Link 联系技术支持。
uint32_t ACL_ERROR_GE_REDEPLOYING = 545602;	正在重部署。	等待重部署动作完成后重新调用相关接口。

1.13.2 aclDataType

```
typedef enum {  
    ACL_DT_UNDEFINED = -1, //未知数据类型，默认值  
    ACL_FLOAT = 0,  
    ACL_FLOAT16 = 1,  
    ACL_INT8 = 2,  
    ACL_INT32 = 3,  
    ACL_UINT8 = 4,  
    ACL_INT16 = 6,  
    ACL_UINT16 = 7,  
    ACL_UINT32 = 8,  
    ACL_INT64 = 9,  
    ACL_UINT64 = 10,  
    ACL_DOUBLE = 11,  
    ACL_BOOL = 12,  
    ACL_STRING = 13,  
    ACL_COMPLEX64 = 16,  
    ACL_COMPLEX128 = 17,  
    ACL_BF16 = 27,  
    ACL_INT4 = 29,  
    ACL_UINT1 = 30,  
    ACL_COMPLEX32 = 33,  
} aclDataType;
```

1.13.3 aclFloat16

该数据类型仅用于数据传递，不能用于加减乘除等运算。

```
typedef uint16_t aclFloat16;
```

1.13.4 aclFormat

```
typedef enum {  
    ACL_FORMAT_UNDEFINED = -1,  
    ACL_FORMAT_NCHW = 0,  
    ACL_FORMAT_NHWC = 1,  
    ACL_FORMAT_ND = 2,  
    ACL_FORMAT_NC1HWC0 = 3,  
    ACL_FORMAT_FRACTAL_Z = 4,  
    ACL_FORMAT_NC1HWC0_C04 = 12,  
    ACL_FORMAT_HWCN = 16,  
    ACL_FORMAT_NDHWC = 27,  
    ACL_FORMAT_FRACTAL_NZ = 29,  
    ACL_FORMAT_NCDHW = 30,  
    ACL_FORMAT_NDC1HWC0 = 32,  
}
```

```
ACL_FRACTAL_Z_3D = 33,  
ACL_FORMAT_NC = 35,  
ACL_FORMAT_NCL = 47,  
} aclFormat;
```

- UNDEFINED: 未知格式, 默认值。
- NCHW: 4维数据格式。
- NHWC: 4维数据格式。
- ND: 表示支持任意格式, 仅有Square、Tanh等这些单输入对自身处理的算子外, 其它需要慎用。
- NC1HWC0: 5维数据格式。其中, C0与微架构强相关, 该值等于cube单元的size, 例如16; C1是将C维度按照C0切分: $C1=C/C0$, 若结果不整除, 最后一份数据需要padding到C0。
- FRACTAL_Z: 卷积的权重的格式。
- NC1HWC0_C04: 5维数据格式。其中, C0固定为4, C1是将C维度按照C0切分: $C1=C/C0$, 若结果不整除, 最后一份数据需要padding到C0。当前版本不支持。
- HWCN: 4维数据格式。
- NDHWC: NDHWC格式。对于3维图像就需要使用带D (Depth) 维度的格式。
- FRACTAL_NZ: 内部分形格式, 用户目前无需使用。
- NCDHW: NCDHW格式。对于3维图像就需要使用带D (Depth) 维度的格式。
- NDC1HWC0: 6维数据格式。相比于NC1HWC0, 仅多了D (Depth) 维度。
- FRACTAL_Z_3D: 3D卷积权重格式, 例如Conv3D/MaxPool3D/AvgPool3D这些算子都需要这种格式来表达。
- NC: 2维数据格式。
- NCL: 3维数据格式。

说明

各维度的含义如下: N (Batch) 表示批量大小、H (Height) 表示特征图高度、W (Width) 表示特征图宽度、C (Channels) 表示特征图通道、D (Depth) 表示特征图深度、L是特征图长度。

1.13.5 aclrtMemMallocPolicy

```
typedef enum aclrtMemMallocPolicy {  
    ACL_MEM_MALLOC_HUGE_FIRST,  
    ACL_MEM_MALLOC_HUGE_ONLY,  
    ACL_MEM_MALLOC_NORMAL_ONLY,  
    ACL_MEM_MALLOC_HUGE_FIRST_P2P,  
    ACL_MEM_MALLOC_HUGE_ONLY_P2P,  
    ACL_MEM_MALLOC_NORMAL_ONLY_P2P,  
    ACL_MEM_TYPE_LOW_BAND_WIDTH = 0x0100,  
    ACL_MEM_TYPE_HIGH_BAND_WIDTH = 0x1000  
} aclrtMemMallocPolicy;
```

此处支持单个枚举项, 也支持多个枚举项位或:

- 配置单个枚举项:
若配置ACL_MEM_MALLOC_HUGE_FIRST、ACL_MEM_MALLOC_HUGE_ONLY、ACL_MEM_MALLOC_NORMAL_ONLY、ACL_MEM_MALLOC_HUGE_FIRST_P2P、ACL_MEM_MALLOC_HUGE_ONLY_P2P、ACL_MEM_MALLOC_NORMAL_ONLY_P2P中的其中一个, 则系统内部会根据硬件支持情况选择从高带宽或低带宽物理内存申请内存;

若配置ACL_MEM_TYPE_LOW_BAND_WIDTH或
ACL_MEM_TYPE_HIGH_BAND_WIDTH，则系统内部会默认采取
ACL_MEM_MALLOC_HUGE_FIRST，优先申请大页。

- 配置多个枚举项位或：支持这三项（ACL_MEM_MALLOC_HUGE_FIRST、
ACL_MEM_MALLOC_HUGE_ONLY、ACL_MEM_MALLOC_NORMAL_ONLY）与
这两项（ACL_MEM_TYPE_LOW_BAND_WIDTH、
ACL_MEM_TYPE_HIGH_BAND_WIDTH）组合，例如：
ACL_MEM_MALLOC_HUGE_FIRST | ACL_MEM_TYPE_HIGH_BAND_WIDTH

表 1-103 枚举项说明

枚举项	说明
ACL_MEM_MALLOC_HUGE_FIRST	当申请的内存小于等于1M时，即使使用该内存分配规则，也是申请普通页的内存。当申请的内存大于1M时，优先申请大页内存，如果大页内存不够，则使用普通页的内存。
ACL_MEM_MALLOC_HUGE_ONLY	仅申请大页，如果大页内存不够，则返回错误。
ACL_MEM_MALLOC_NORMAL_ONLY	仅申请普通页，如果普通页内存不够，则返回错误。
ACL_MEM_MALLOC_HUGE_FIRST_P2P	仅Device之间内存复制场景下申请内存时使用该选项，表示优先申请大页内存，如果大页内存不够，则使用普通页的内存。 Atlas 200/300/500 推理产品，不支持该选项。 Atlas 200/500 A2推理产品，不支持该选项。
ACL_MEM_MALLOC_HUGE_ONLY_P2P	仅Device之间内存复制场景下申请内存时使用该选项，仅申请大页内存，如果大页内存不够，则返回错误。 Atlas 200/300/500 推理产品，不支持该选项。 Atlas 200/500 A2推理产品，不支持该选项。
ACL_MEM_MALLOC_NORMAL_ONLY_P2P	仅Device之间内存复制场景下申请内存时使用该选项，仅申请普通页的内存。 Atlas 200/300/500 推理产品，不支持该选项。 Atlas 200/500 A2推理产品，不支持该选项。
ACL_MEM_TYPE_LOW_BAND_WIDTH	从带宽低的物理内存上申请内存。 设置该选项无效，系统默认会根据硬件支持的内存类型选择。
ACL_MEM_TYPE_HIGH_BAND_WIDTH	从带宽高的物理内存上申请内存。 设置该选项无效，系统默认会根据硬件支持的内存类型选择。

1.13.6 acldvppPixelFormat

```
// Supported Pixel Format
enum acldvppPixelFormat {
    PIXEL_FORMAT_YUV_400 = 0, // 0, YUV400 8bit
    PIXEL_FORMAT_YUV_SEMIPLANAR_420 = 1, // 1, YUV420SP NV12 8bit
    PIXEL_FORMAT_YVU_SEMIPLANAR_420 = 2, // 2, YUV420SP NV21 8bit
    PIXEL_FORMAT_YUV_SEMIPLANAR_422 = 3, // 3, YUV422SP 8bit
    PIXEL_FORMAT_YVU_SEMIPLANAR_422 = 4, // 4, YVU422SP 8bit
    PIXEL_FORMAT_YUV_SEMIPLANAR_444 = 5, // 5, YUV444SP 8bit
    PIXEL_FORMAT_YVU_SEMIPLANAR_444 = 6, // 6, YVU444SP 8bit
    PIXEL_FORMAT_YUYV_PACKED_422 = 7, // 7, YUV422P YUYV 8bit
    PIXEL_FORMAT_UYVY_PACKED_422 = 8, // 8, YUV422P UYVY 8bit
    PIXEL_FORMAT_YVYU_PACKED_422 = 9, // 9, YUV422P YVYU 8bit
    PIXEL_FORMAT_VYUY_PACKED_422 = 10, // 10, YUV422P VYUY 8bit
    PIXEL_FORMAT_YUV_PACKED_444 = 11, // 11, YUV444P 8bit
    PIXEL_FORMAT_RGB_888 = 12, // 12, RGB888
    PIXEL_FORMAT_BGR_888 = 13, // 13, BGR888
    PIXEL_FORMAT_ARGB_8888 = 14, // 14, ARGB8888
    PIXEL_FORMAT_ABGR_8888 = 15, // 15, ABGR8888
    PIXEL_FORMAT_RGBA_8888 = 16, // 16, RGBA8888
    PIXEL_FORMAT_BGRA_8888 = 17, // 17, BGRA8888
    PIXEL_FORMAT_YUV_SEMI_PLANNER_420_10BIT = 18, // 18, YUV420SP 10bit
    PIXEL_FORMAT_YVU_SEMI_PLANNER_420_10BIT = 19, // 19, YVU420S 10bit
    PIXEL_FORMAT_YVU_PLANAR_420 = 20, // 20, YVU420P 8bit
    PIXEL_FORMAT_YVU_PLANAR_422, // YVU422P 8bit
    PIXEL_FORMAT_YVU_PLANAR_444, // YVU444P 8bit
    PIXEL_FORMAT_RGB_444 = 23, // RGB444 R:4bit G:4bit B:4bit, 当前不支持该格式
    PIXEL_FORMAT_BGR_444, // BGR444 R:4bit G:4bit B:4bit, 当前不支持该格式
    PIXEL_FORMAT_ARGB_4444, // ARGB4444 A:4bit R:4bit G:4bit B:4bit, 当前不支持该格式
    PIXEL_FORMAT_ABGR_4444, // ABGR4444 A:4bit R:4bit G:4bit B:4bit, 当前不支持该格式
    PIXEL_FORMAT_RGBA_4444, // RGBA4444 R:4bit G:4bit B:4bit A:4bit, 当前不支持该格式
    PIXEL_FORMAT_BGRA_4444, // BGRA4444 B:4bit G:4bit R:4bit A:4bit, 当前不支持该格式
    PIXEL_FORMAT_RGB_555, // RGB555 R:5bit G:5bit B:5bit, 当前不支持该格式
    PIXEL_FORMAT_BGR_555, // BGR555 B:5bit G:5bit R:5bit, 当前不支持该格式
    PIXEL_FORMAT_RGB_565, // RGB565 R:5bit G:6bit B:5bit, 当前不支持该格式
    PIXEL_FORMAT_BGR_565, // BGR565 B:5bit G:6bit R:5bit, 当前不支持该格式
    PIXEL_FORMAT_ARGB_1555, // ARGB1555 A:1bit R:5bit G:6bit B:5bit, 当前不支持该格式
    PIXEL_FORMAT_ABGR_1555, // ABGR1555 A:1bit B:5bit G:6bit R:5bit, 当前不支持该格式
    PIXEL_FORMAT_RGBA_1555, // RGBA1555 A:1bit B:5bit G:6bit R:5bit, 当前不支持该格式
    PIXEL_FORMAT_BGRA_1555, // BGRA1555 A:1bit B:5bit G:6bit R:5bit, 当前不支持该格式
    PIXEL_FORMAT_ARGB_8565, // ARGB8565 A:8bit R:5bit G:6bit B:5bit, 当前不支持该格式
    PIXEL_FORMAT_ABGR_8565, // ABGR8565 A:8bit B:5bit G:6bit R:5bit, 当前不支持该格式
    PIXEL_FORMAT_RGBA_8565, // RGBA8565 A:8bit R:5bit G:6bit B:5bit, 当前不支持该格式
    PIXEL_FORMAT_BGRA_8565, // BGRA8565 A:8bit B:5bit G:6bit R:5bit, 当前不支持该格式
    PIXEL_FORMAT_RGB_BAYER_8BPP = 50, // RGB Bayer 8bit, Bayer图像, 当前不支持该格式
    PIXEL_FORMAT_RGB_BAYER_10BPP, // RGB Bayer 10bit, Bayer图像, 当前不支持该格式
    PIXEL_FORMAT_RGB_BAYER_12BPP, // RGB Bayer 12bit, Bayer图像, 当前不支持该格式
    PIXEL_FORMAT_RGB_BAYER_14BPP, // RGB Bayer 14bit, Bayer图像, 当前不支持该格式
    PIXEL_FORMAT_RGB_BAYER_16BPP, // RGB Bayer 16bit, Bayer图像, 当前不支持该格式
    PIXEL_FORMAT_BGR_888_PLANAR = 70, // RGB888 Planar, 当前不支持该格式
    PIXEL_FORMAT_HSV_888_PACKAGE, // HSV Package, HSV图像package格式, 当前不支持该格式
    PIXEL_FORMAT_HSV_888_PLANAR, // HSV Planar, HSV图像Planar格式, 当前不支持该格式
    PIXEL_FORMAT_LAB_888_PACKAGE, // LAB Package, LAB图像package格式, 当前不支持该格式
    PIXEL_FORMAT_LAB_888_PLANAR, // LAB Planar, LAB图像Planar格式, 当前不支持该格式
    PIXEL_FORMAT_S8C1, // Signed 8bit for 1pixel 1Channel, 每个像素用1个8bit有符号数据表示的单通道图像, 当前不支持该格式
    PIXEL_FORMAT_S8C2_PACKAGE, // Signed 8bit for 1pixel 2Channel Package, 每个像素用2个8bit有符号数表示的双通道图像Package格式, 当前不支持该格式
    PIXEL_FORMAT_S8C2_PLANAR, // Signed 8bit for 1pixel 2Channel Planar, 每个像素用2个8bit有符号数据表的双通道图像Planar格式, 当前不支持该格式
    PIXEL_FORMAT_S16C1, // Signed 16bit 1pixel 1Channel, 每个像素用1个16bit有符号数据表示的单通道图像, 当前不支持该格式
    PIXEL_FORMAT_U8C1, // Unsigned 8bit 1pixel 1Channel, 每个像素用1个8bit无符号数据表示的单通道图像, 当前不支持该格式
    PIXEL_FORMAT_U16C1, // Unsigned 16bit 1pixel 1Channel, 每个像素用1个16bit无符号数据表示的单通道图像, 当前不支持该格式
    PIXEL_FORMAT_S32C1, // Signed 32bit 1pixel 1Channel, 每个像素用1个32bit有符号数据表示
```

```
的单通道图像，当前不支持该格式
PIXEL_FORMAT_U32C1, // Unsigned 32bit 1pixel 1Channel, 每个像素用1个32bit无符号数据
表示的单通道图像，当前不支持该格式
PIXEL_FORMAT_U64C1, // Unsigned 64bit 1pixel 1Channel, 每个像素用1个64bit无符号数据
表示的单通道图像，当前不支持该格式
PIXEL_FORMAT_S64C1, // Signed 64bit 1pixel 1Channel, 每个像素用1个64bit有符号数据表示
的单通道图像，当前不支持该格式
PIXEL_FORMAT_YUV_SEMIPLANAR_440 = 1000, // YUV440SP 8bit
PIXEL_FORMAT_YVU_SEMIPLANAR_440, // YVU440SP 8bit
PIXEL_FORMAT_FLOAT32, // Float 32bit for 1pixel, 每个像素用1个float32数据表示，当前不支
持该格式
PIXEL_FORMAT_BUTT,
PIXEL_FORMAT_UNKNOWN = 10000
};
```

1.13.7 acldvppStreamFormat

```
enum acldvppStreamFormat {
    H265_MAIN_LEVEL = 0,
    H264_BASELINE_LEVEL,
    H264_MAIN_LEVEL,
    H264_HIGH_LEVEL
};
```

1.13.8 acldvppJpegFormat

```
enum acldvppJpegFormat {
    ACL_JPEG_CSS_444 = 0,
    ACL_JPEG_CSS_422,
    ACL_JPEG_CSS_420,
    ACL_JPEG_CSS_GRAY,
    ACL_JPEG_CSS_440,
    ACL_JPEG_CSS_411,
    ACL_JPEG_CSS_UNKNOWN = 1000
};
```

1.13.9 acldvppCscMatrix

Atlas 200/300/500 推理产品，不支持该枚举值。

Atlas 训练系列产品，不支持该枚举值。

```
enum acldvppCscMatrix {
    ACL_DVPP_CSC_MATRIX_BT601_WIDE = 0, //基于BT601 wide标准的色域转换矩阵，默认值
    ACL_DVPP_CSC_MATRIX_BT601_NARROW, //基于BT601 narrow标准的色域转换矩阵
    ACL_DVPP_CSC_MATRIX_BT709_WIDE, //基于BT709 wide标准的色域转换矩阵
    ACL_DVPP_CSC_MATRIX_BT709_NARROW, //基于BT709 narrow标准的色域转换矩阵
    ACL_DVPP_CSC_MATRIX_BT2020_WIDE, //基于BT2020 wide标准的色域转换矩阵
    ACL_DVPP_CSC_MATRIX_BT2020_NARROW //基于BT2020 narrow标准的色域转换矩阵
};
```

基于以上各标准的色域转换矩阵，其各分量的转换公式如下：

```
# YUV转RGB:
# | R | | matrix_r0c0 matrix_r0c1 matrix_r0c2 | | Y - input_bias_0 |
# | G | = | matrix_r1c0 matrix_r1c1 matrix_r1c2 | * | U - input_bias_1 |
# | B | | matrix_r2c0 matrix_r2c1 matrix_r2c2 | | V - input_bias_2 |
# BGR转YUV:
# | Y | | output_bias_0 | | matrix_r0c0 matrix_r0c1 matrix_r0c2 | | R |
# | U | = | output_bias_1 | + | matrix_r1c0 matrix_r1c1 matrix_r1c2 | * | G |
# | V | | output_bias_2 | | matrix_r2c0 matrix_r2c1 matrix_r2c2 | | B |
```

基于不同标准的色域转换矩阵，各参数值如下表所示。

成员名称	描述
ACL_DVPP_CSC_MATRIX_BT601_WIDE	基于BT601 wide标准的色域转换矩阵。默认为ACL_DVPP_CSC_MATRIX_BT601_WIDE。 基于该标准的色域转换矩阵，各参数值如下： <pre># YUV转RGB: # R 1.000 0.000 1.402 Y - 0 # G = 1.000 -0.344 -0.714 * U - 128 # B 1.000 1.772 0.000 V - 128 # RGB转YUV: # Y -0.5 0.299 0.587 0.114 R # U = 127.5 + -0.168 -0.331 0.500 * G # V 127.5 0.500 -0.419 -0.081 B </pre>
ACL_DVPP_CSC_MATRIX_BT601_NARROW	基于BT601 narrow标准的色域转换矩阵。 基于该标准的色域转换矩阵，各参数值如下： <pre># YUV转RGB: # R 1.16438 0.00000 1.59602 Y - 16 # G = 1.16438 -0.39176 -0.81297 * U - 128 # B 1.16438 2.01723 0.00000 V - 128 # RGB转YUV: # Y 16 0.25679 0.51564 0.10014 R # U = 128 + -0.14491 -0.29099 0.43922 * G # V 128 0.42941 -0.36779 -0.07143 B </pre>
ACL_DVPP_CSC_MATRIX_BT709_WIDE	基于BT709 wide标准的色域转换矩阵。 基于该标准的色域转换矩阵，各参数值如下： <pre># YUV转RGB: # R 1.00000 0.00000 1.57480 Y - 0 # G = 1.00000 -0.18732 -0.46812 * U - 128 # B 1.00000 1.85560 0.00000 V - 128 # RGB转YUV: # Y 0 0.21260 0.71520 0.07220 R # U = 128 + -0.11457 -0.38543 0.50000 * G # V 128 0.50000 -0.45415 -0.04585 B </pre>
ACL_DVPP_CSC_MATRIX_BT709_NARROW	基于BT709 narrow标准的色域转换矩阵。 基于该标准的色域转换矩阵，各参数值如下： <pre># YUV转RGB: # R 1.16438 0.00000 1.79274 Y - 16 # G = 1.16438 -0.21325 -0.53291 * U - 128 # B 1.16438 2.11240 0.00000 V - 128 # RGB转YUV: # Y 16 0.18259 0.62825 0.06342 R # U = 128 + -0.09840 -0.33857 0.43922 * G # V 128 0.42941 -0.39894 -0.04027 B </pre>
ACL_DVPP_CSC_MATRIX_BT2020_WIDE	基于BT2020 wide标准的色域转换矩阵。 基于该标准的色域转换矩阵，各参数值如下： <pre># YUV转RGB: # R 1.00000 0.00000 1.47460 Y - 0 # G = 1.00000 -0.16455 -0.57135 * U - 128 # B 1.00000 1.88140 0.00000 V - 128 # RGB转YUV: # Y 0 0.26270 0.67800 0.05930 R # U = 128 + -0.13963 -0.36037 0.50000 * G # V 128 0.50000 -0.45979 -0.04021 B </pre>

成员名称	描述
ACL_DVPP_CSC_MATRIX_BT2020_NARROW	基于BT2020 narrow标准的色域转换矩阵。 基于该标准的色域转换矩阵，各参数值如下： # YUV转RGB： # R 1.16438 0.00000 1.67868 Y - 16 # G = 1.16438 -0.18733 -0.65042 * U - 128 # B 1.16438 2.14177 0.00000 V - 128 # RGB转YUV： # Y 16 0.22564 0.59558 0.05209 R # U = 128 + -0.11992 -0.31656 0.43922 * G # V 128 0.42941 -0.40389 -0.03533 B

1.13.10 aclrtContext

```
typedef void *aclrtContext;
```

1.13.11 aclrtStream

```
typedef void *aclrtStream;
```

1.13.12 aclrtEvent

```
typedef void *aclrtEvent;
```

1.13.13 aclrtEventStatus

```
typedef enum aclrtEventStatus {  
    ACL_EVENT_STATUS_COMPLETE = 0, //完成  
    ACL_EVENT_STATUS_NOT_READY = 1, //未完成  
    ACL_EVENT_STATUS_RESERVED = 2, //预留  
} aclrtEventStatus;
```

1.13.14 aclrtRunMode

```
typedef enum aclrtRunMode {  
    ACL_DEVICE, // AI软件栈运行在Device的Control CPU或板端环境上  
    ACL_HOST, // AI软件栈运行在Host侧  
} aclrtRunMode;
```

1.13.15 aclTransType

```
typedef enum aclTransType  
{  
    ACL_TRANS_N, //不转置，默认为不转置  
    ACL_TRANS_T, //转置  
    ACL_TRANS_NZ, //内部格式，能够提供更好的接口性能，建议使用场景为：某个输入矩阵作为底库，执行一次转NZ格式的操作后，多次使用NZ格式的算子，提升性能。  
    ACL_TRANS_NZ_T //内部格式转置，当前不支持  
}aclTransType;
```

1.13.16 aclComputeType

```
typedef enum aclComputeType  
{  
    ACL_COMPUTE_HIGH_PRECISION,  
    ACL_COMPUTE_LOW_PRECISION //当前不支持  
}aclComputeType;
```


1.13.17 aclfvSearchType

Atlas 200/300/500 推理产品，不支持该枚举值。

Atlas 200/500 A2推理产品，不支持该枚举值。

Atlas 训练系列产品，不支持该枚举值。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该枚举值。

```
enum aclfvSearchType {  
    SEARCH_1_N, // 1:N operation type  
    SEARCH_N_M // N:M operation type  
};
```

1.13.18 aclAippInputFormat

静态AIPP和动态AIPP支持的图片格式不同，详情如下：

```
typedef enum  
{  
    ACL_YUV420SP_U8 = 1, // YUV420SP_U8, 静态AIPP和动态AIPP都支持  
    ACL_XRGB8888_U8=2, // XRGB8888_U8, 静态AIPP和动态AIPP都支持  
    ACL_RGB888_U8=3, // RGB888_U8, 静态AIPP和动态AIPP都支持  
    ACL_YUV400_U8=4, // YUV400_U8, 静态AIPP和动态AIPP都支持  
    ACL_NC1HWC0DI_FP16 = 5, // 暂不支持  
    ACL_NC1HWC0DI_S8 = 6, // 暂不支持  
    ACL_ARGB8888_U8 = 7, // 暂不支持  
    ACL_YUYV_U8 = 8, // 暂不支持  
    ACL_YUV422SP_U8 = 9, // 暂不支持  
    ACL_AYUV444_U8 = 10, // 暂不支持  
    ACL_RAW10 = 11, // 暂不支持  
    ACL_RAW12 = 12, // 暂不支持  
    ACL_RAW16 = 13, // 暂不支持  
    ACL_RAW24 = 14, // 暂不支持  
    ACL_AIPP_RESERVED = 0xffff,  
} aclAippInputFormat;
```

1.13.19 aclAippInfo

```
typedef struct aclAippInfo {  
    aclAippInputFormat inputFormat;  
    int32_t srcImageSizeW;  
    int32_t srcImageSizeH;  
    int8_t cropSwitch;  
    int32_t loadStartPosW;  
    int32_t loadStartPosH;  
    int32_t cropSizeW;  
    int32_t cropSizeH;  
    int8_t resizeSwitch;  
    int32_t resizeOutputW;  
    int32_t resizeOutputH;  
    int8_t paddingSwitch;  
    int32_t leftPaddingSize;  
    int32_t rightPaddingSize;  
    int32_t topPaddingSize;  
    int32_t bottomPaddingSize;  
    int8_t cscSwitch;  
    int8_t rbufSwapSwitch;  
    int8_t axSwapSwitch;  
    int8_t singleLineMode;  
    int32_t matrixR0C0;  
    int32_t matrixR0C1;  
    int32_t matrixR0C2;  
    int32_t matrixR1C0;  
    int32_t matrixR1C1;
```

```
int32_t matrixR1C2;  
int32_t matrixR2C0;  
int32_t matrixR2C1;  
int32_t matrixR2C2;  
int32_t outputBias0;  
int32_t outputBias1;  
int32_t outputBias2;  
int32_t inputBias0;  
int32_t inputBias1;  
int32_t inputBias2;  
int32_t meanChn0;  
int32_t meanChn1;  
int32_t meanChn2;  
int32_t meanChn3;  
float minChn0;  
float minChn1;  
float minChn2;  
float minChn3;  
float varReciChn0;  
float varReciChn1;  
float varReciChn2;  
float varReciChn3;  
aclFormat srcFormat; // 模型转换前, 原始模型的输入format  
aclDataType srcDatatype; // 模型转换前, 原始模型的输入datatype  
size_t srcDimNum; // 模型转换前, 原始模型输入dims  
size_t shapeCount; // 动态Shape ( 动态Batch或动态分辨率 ) 场景下的档位数  
aclAippDims outDims[ACL_MAX_SHAPE_COUNT]; // 在多shape模型下, 各个分档对应的aipp输出dims信息, 需要在模型编译阶段按照shape推导出来, 保存在模型中, 输出的Format, datatype、输出dims等不随shape变化而变化  
aclAippExtendInfo *aippExtend; // 预留参数, 当前版本用户必须传入空指针  
} aclAippInfo;
```

1.13.20 aclAippDims

```
typedef struct aclAippDims {  
    aclmdlIODims srcDims; // 模型转换前的dims, 如[1,3,150,150]  
    size_t srcSize; // 模型转换前输入数据的大小  
    aclmdlIODims aippOutdims; // 内部数据格式的dims信息  
    size_t aippOutSize; // 经过AIPP处理后的输出数据的大小  
} aclAippDims;
```

1.13.21 aclmdlIODims

```
#define ACL_MAX_DIM_CNT 128  
#define ACL_MAX_TENSOR_NAME_LEN 128  
typedef struct aclmdlIODims {  
    char name[ACL_MAX_TENSOR_NAME_LEN]; /**< tensor name */  
    size_t dimCount; /**Shape中的维度个数, 如果为标量, 则维度个数为0*/  
    int64_t dims[ACL_MAX_DIM_CNT]; /**< 维度信息 */  
} aclmdlIODims;
```

1.13.22 aclrtGroupAttr

Atlas 200/300/500 推理产品, 不支持该枚举值。

Atlas 200/500 A2推理产品, 不支持该枚举值。

Atlas 训练系列产品, 不支该枚举值。

Atlas A2训练系列产品/Atlas 800I A2推理产品, 不支持该枚举值。

```
typedef enum aclrtGroupAttr  
{  
    ACL_GROUP_AICORE_INT, //指定Group对应的aicore个数, 属性值的数据类型为整型。  
    ACL_GROUP_AIV_INT, //指定Group对应的vector core个数, 属性值的数据类型为整型。
```

```
ACL_GROUP_AIC_INT, //指定Group对应的aicpu线程数, 属性值的数据类型为整型。
ACL_GROUP_SDMANUM_INT, //内存异步拷贝的通道数, 属性值的数据类型为整型。
ACL_GROUP_ASQNUM_INT, //指定Group下可以被同时调度执行的stream个数, 小于或等于32, 当前系统
级最大一共可以同时调度32个stream。属性值的数据类型为整型。
ACL_GROUP_GROUPID_INT //指定Group的ID。属性值的数据类型为整型。
} aclrtGroupAttr;
```

1.13.23 aclprofAicoreMetrics

```
typedef enum {
    ACL_AICORE_ARITHMETIC_UTILIZATION = 0, //各种计算类指标占比统计, 包括采集项mac_fp16_ratio、
    mac_int8_ratio、vec_fp32_ratio、vec_fp16_ratio、vec_int32_ratio、vec_misc_ratio
    ACL_AICORE_PIPE_UTILIZATION = 1, //计算单元和搬运单元耗时占比, 包括采集项vec_ratio、
    mac_ratio、scalar_ratio、mte1_ratio、mte2_ratio、mte3_ratio、icache_miss_rate、fixpipe_ratio ( Atlas A2训
    练系列产品/Atlas 800I A2推理产品 )
    ACL_AICORE_MEMORY_BANDWIDTH = 2, //外部内存读写类指令占比, 包括采集项ub_read_bw、
    ub_write_bw、l1_read_bw、l1_write_bw、l2_read_bw ( Atlas 200/300/500 推理产品 )、l2_write_bw ( Atlas
    200/300/500 推理产品 )、main_mem_read_bw、main_mem_write_bw
    ACL_AICORE_L0B_AND_WIDTH = 3, //内部内存读写类指令占比, 包括采集项scalar_ld_ratio、
    scalar_st_ratio、l0a_read_bw、l0a_write_bw、l0b_read_bw、l0b_write_bw、l0c_read_bw、l0c_write_bw、
    l0c_read_bw_cube、l0c_write_bw_cube
    ACL_AICORE_RESOURCE_CONFLICT_RATIO = 4, //流水线队列类指令占比, 包括采集项
    vec_bankgroup_cflt_ratio、vec_bank_cflt_ratio、vec_resc_cflt_ratio、mte1_iq_full_ratio、mte2_iq_full_ratio、
    mte3_iq_full_ratio、cube_iq_full_ratio、vec_iq_full_ratio、iq_full_ratio
    ACL_AICORE_MEMORY_UB = 5, //内部内存读写指令占比, 包括采集项ub_read_bw_mte ( Atlas
    200/300/500 推理产品 )、ub_write_bw_mte ( Atlas 200/300/500 推理产品 )、ub_read_bw_vector、
    ub_write_bw_vector、ub_read_bw_scalar、ub_write_bw_scalar
    ACL_AICORE_L2_CACHE = 6, // ( Atlas A2训练系列产品/Atlas 800I A2推理产品 ) 读写cache命中
    次数和缺失后重新分配次数, 包括采集项ai*_write_cache_hit、ai*_write_cache_miss_allocate、
    ai*_r*_read_cache_hit、ai*_r*_read_cache_miss_allocate
    ACL_AICORE_L2_CACHE = 6, // ( Atlas 200/500 A2推理产品 ) 读写cache命中次数和缺失后重新分
    配次数, 包括采集项write_cache_hit、write_cache_miss_allocate、r*_read_cache_hit、
    r*_read_cache_miss_allocate
    ACL_AICORE_PIPE_EXECUTE_UTILIZATION = 7, // ( Atlas 200/500 A2推理产品 ) 计算单元和搬运单元耗
    时占比, 包括采集项vec_exe_ratio、mac_exe_ratio、scalar_exe_ratio、mte1_exe_ratio、mte2_exe_ratio、
    mte3_exe_ratio、fixpipe_exe_ratio
    ACL_AICORE_NONE = 0xFF
}aclprofAicoreMetrics;
```

1.13.24 acldvppBorderType

Atlas 200/300/500 推理产品, 当前版本不支持该枚举值。

Atlas 训练系列产品, 当前版本不支持该枚举值。

```
enum acldvppBorderType {
    BORDER_CONSTANT = 0, //添加有颜色的常数值边界
    BORDER_REPLICATE, //重复最后一个元素。举例: aaaaaa|abcdefgh|hhhhhhh
    BORDER_REFLECT, //边界元素的镜像。举例: cba|abcde-fgh|hgf
    BORDER_REFLECT_101 //边界元素的镜像。举例: cba|abcde-fgh|gf
};
```

1.13.25 aclmdlInputAippType

```
typedef enum {
    ACL_DATA_WITHOUT_AIPP = 0, //该输入无AIPP信息
    ACL_DATA_WITH_STATIC_AIPP, //该输入带静态AIPP信息
    ACL_DATA_WITH_DYNAMIC_AIPP, //该输入有关联的动态AIPP输入
    ACL_DYNAMIC_AIPP_NODE //该输入本身就是动态AIPP输入, 用于配置动态AIPP参数
} aclmdlInputAippType;
```

1.13.26 acltdtTensorType

Atlas 200/300/500 推理产品, 当前版本不支持该枚举值。

Atlas 200/500 A2推理产品, 当前版本不支持该枚举值。

Atlas 推理系列产品，当前版本不支持该枚举值。

```
enum acltdtTensorType {
    ACL_TENSOR_DATA_UNDEFINED = -1,
    ACL_TENSOR_DATA_TENSOR,      // 正常tensor数据标识
    ACL_TENSOR_DATA_END_OF_SEQUENCE, // end数据标识
    ACL_TENSOR_DATA_ABNORMAL,    // 异常数据标识
    ACL_TENSOR_DATA_SLICE_TENSOR, // tensor分片场景下的tensor数据
    ACL_TENSOR_DATA_END_TENSOR   // tensor分片场景下标识最后一个tensor
};
```

1.13.27 aclCompileOpt

```
typedef enum {
    ACL_PRECISION_MODE,      // 网络模型的算子精度模式
    ACL_AICORE_NUM,          // 模型编译时使用的AI Core数量
    ACL_AUTO_TUNE_MODE,      // 算子的自动调优模式
    ACL_OP_SELECT_IMPL_MODE, // 选择算子是高精度实现还是高性能实现
    ACL_OPTYPELIST_FOR_IMPLMODE, // 列举算子类型的列表，该列表中的算子使用
    ACL_OP_SELECT_IMPL_MODE指定的模式
    ACL_OP_DEBUG_LEVEL,      // TBE算子编译debug功能开关
    ACL_DEBUG_DIR,           // 保存模型转换、网络迁移过程中算子编译生成的调试相关过程文件的路径，包
    括算子.o/.json/.cce等文件。
    ACL_OP_COMPILER_CACHE_MODE, // 算子编译磁盘缓存模式
    ACL_OP_COMPILER_CACHE_DIR,  // 算子编译磁盘缓存的目录
    ACL_OP_PERFORMANCE_MODE,    // 通过该选项设置是否按照算子执行高性能的方式编译算子
    ACL_OP_JIT_COMPILE,         // 选择是在线编译算子，还是使用已编译的算子二进制文件
    ACL_OP_DETERMINISTIC,       // 是否开启确定性计算
    ACL_CUSTOMIZE_DTYPES,       // 模型编译时自定义某个或某些算子的计算精度
    ACL_OP_PRECISION_MODE,      // 指定算子内部处理时的精度模式，支持指定一个算子或多个算子。
    ACL_ALLOW_HF32,             // hf32是昇腾推出的专门用于算子内部计算的精度类型，当前版本不支持
    ACL_PRECISION_MODE_V2,      // 网络模型的算子精度模式，相比ACL_PRECISION_MODE选项，
    ACL_PRECISION_MODE_V2是新版本中新增的，精度模式更多，同时原有精度模式选项值语义更清晰，便于理解
    ACL_OP_DEBUG_OPTION         // 当前仅支持配置为oom，表示开启Global Memory访问越界检测
} aclCompileOpt;
```

表 1-104 编译选项配置

编译选项	取值说明
ACL_PRECISION_MODE	用于配置网络模型的算子精度模式。如果不配置该编译选项，默认采用allow_fp32_to_fp16。 ● force_fp32/cube_fp16in_fp32out: 配置为force_fp32或cube_fp16in_fp32out，效果等同，系统内部都会根据矩阵类算子或矢量类算子，来选择不同的处理方式。cube_fp16in_fp32out为新版本中新增的，对于矩阵计算类算子，该选项语义更清晰。 - 对于矩阵计算类算子，系统内部会按算子实现的支持情况处理： 1. 优先选择输入数据类型为float16且输出数据类型为float32； 2. 如果1中的场景不支持，则选择输入数据类型为float32且输出数据类型为float32； 3. 如果2中的场景不支持，则选择输入数据类型为float16且输出数据类型为float16； 4. 如果3中的场景不支持，则报错。 - 对于矢量计算类算子，表示网络模型中算子支持float16和float32时，强制选择float32，若原图精度为float16，也会强制转为float32。 如果网络模型中存在部分算子，并且该算子实现不支持float32，比如某算子仅支持float16类型，则该参数不生效，仍然使用支持的float16；如果该算子不支持float32，且又配置了黑名单（precision_reduce = false），则会使用float32的AI CPU算子；如果AI CPU算子也不支持，则执行报错。 ● force_fp16: 表示网络模型中算子支持float16和float32时，强制选择float16。 ● allow_fp32_to_fp16: - 对于矩阵类算子，使用float16。 - 对于矢量类算子，优先保持原图精度，如果网络模型中算子支持float32，则保留原始精度float32，如果网络模型中算子不支持float32，则直接降低精度到float16。 ● must_keep_origin_dtype: 保持原图精度。 - 如果原图中某算子精度为float16，AI Core中该算子的实现不支持float16、仅支持float32和bfloat16，则系统内部会自动采用高精度float32。 - 如果原图中某算子精度为float16，AI Core中该算子的实现不支持float16、仅支持bfloat16，则会使

编译选项	取值说明
	用float16的AI CPU算子；如果AI CPU算子也不支持，则执行报错。 - 如果原图中某算子精度为float32，AI Core中该算子的实现不支持float32类型、仅支持float16类型，则会使用float32的AI CPU算子；如果AI CPU算子也不支持，则执行报错。 bfloat16类型仅在以下产品型号支持： Atlas A2训练系列产品/Atlas 800I A2推理产品 Atlas 200/500 A2推理产品 ● allow_mix_precision/allow_mix_precision_fp16: 配置为allow_mix_precision或allow_mix_precision_fp16，效果等同，均表示使用混合精度float16和float32数据类型来处理神经网络的过程。allow_mix_precision_fp16为新版本中新增的，语义更清晰，便于理解。 针对网络模型中float32数据类型的算子，按照内置的优化策略，自动将部分float32的算子降低精度到float16，从而在精度损失很小的情况下提升系统性能并减少内存使用。 若配置了该种模式，则可以在OPP软件包安装路径\${INSTALL_DIR}/opp/built-in/op_impl/ai_core/tbe/config/<soc_version>/aic-<soc_version>-ops-info.json内置优化策略文件中查看“precision_reduce”参数的取值： - 若取值为true（白名单），则表示允许将当前float32类型的算子，降低精度到float16。 - 若取值为false（黑名单），则不允许将当前float32类型的算子降低精度到float16，相应算子仍旧使用float32精度。 - 若网络模型中算子没有配置该参数（灰名单），当前算子的混合精度处理机制和前一个算子保持一致，即如果前一个算子支持降精度处理，当前算子也支持降精度；如果前一个算子不允许降精度，当前算子也不支持降精度。 ● allow_mix_precision_bf16: 表示使用混合精度bfloat16和float32数据类型来处理神经网络的过程。针对原始模型中float32数据类型的算子，按照内置的优化策略，自动将部分float32的算子降低精度到bfloat16，从而在精度损失很小的情况下提升系统性能并减少内存使用；如果算子不支持bfloat16和float32，则使用AI CPU算子进行计算；如果AI CPU算子也不支持，则执行报错。 若配置了该种模式，则可以在OPP软件包安装路径\${INSTALL_DIR}/opp/built-in/op_impl/ai_core/tbe/config/<soc_version>/aic-<soc_version>-ops-info.json内置优化策略文件中查看“precision_reduce”参数的取值：

编译选项	取值说明
	- 若取值为true（白名单），则表示允许将当前float32类型的算子，降低精度到bfloat16。 - 若取值为false（黑名单），则不允许将当前float32类型的算子降低精度到bfloat16，相应算子仍旧使用float32精度。 - 若网络模型中算子没有配置该参数（灰名单），当前算子的混合精度处理机制和前一个算子保持一致，即如果前一个算子支持降精度处理，当前算子也支持降精度；如果前一个算子不允许降精度，当前算子也不支持降精度。 该选项仅在以下产品型号支持： Atlas A2训练系列产品/Atlas 800I A2推理产品 Atlas 200/500 A2推理产品 ● allow_fp32_to_bf16: - 对于矩阵计算类算子，使用bfloat16。 - 对于矢量计算类算子，优先保持原图精度，如果原始模型中算子支持float32，则保留原始精度float32；如果原始模型中算子不支持float32，则直接降低精度到bfloat16；如果算子不支持bfloat16和float32，则使用AI CPU算子进行计算；如果AI CPU算子也不支持，则执行报错。 该选项仅在以下产品型号支持： Atlas A2训练系列产品/Atlas 800I A2推理产品 Atlas 200/500 A2推理产品
ACL_AICORE_NUM	用于配置模型编译时使用的AI Core数量。 当前版本设置无效。
ACL_AUTO_TUNE_MODE	该参数后续废弃，请勿配置，否则后续版本可能存在兼容性问题。若涉及调优，请参见《AOE工具使用指南》。 用于配置算子的自动调优模式。 ● GA（Genetic Algorithm）：遗传算法，用于设置Cube算子的调优性能。 ● RL（Reinforcement Learning）：强化学习，用于设置Vector算子的调优性能。

编译选项	取值说明
ACL_OP_SELECT_IMPL_MODE	用于选择算子是高精度实现还是高性能实现。如果不配置该编译选项，默认采用high_precision。 • high_precision：表示算子采用高精度实现模式。 该选项采用系统内置的配置文件设置算子实现模式，内置配置文件路径为\${INSTALL_DIR}/opp/built-in/op_impl/ai_core/tbe/impl_mode/high_precision.ini。 为保持兼容，该参数仅对high_precision.ini文件中算子列表生效，通过该列表可以控制算子生效的范围并保证之前版本的网络模型不受影响。 • high_performance：表示算子采用高性能实现模式。 该选项采用系统内置的配置文件设置算子实现模式，内置配置文件路径为\${INSTALL_DIR}/opp/built-in/op_impl/ai_core/tbe/impl_mode/high_performance.ini。 为保持兼容，该参数仅对high_performance.ini文件中算子列表生效，通过该列表可以控制算子生效的范围并保证之前版本的网络模型不受影响。 • high_precision_for_all：表示算子采用高精度实现模式。 该选项采用系统内置的配置文件设置算子实现模式，内置配置文件路径为\${INSTALL_DIR}/opp/built-in/op_impl/ai_core/tbe/impl_mode/high_precision_for_all.ini，该文件中列表后续可能会跟随版本更新。 该实现模式不保证兼容，如果后续新的软件包中有算子新增了实现模式（即配置文件中新增了某个算子的实现模式），之前版本使用high_precision_for_all的网络模型，在新版本上性能可能会下降。 • high_performance_for_all：表示算子采用高性能实现模式。 该选项采用系统内置的配置文件设置算子实现模式，内置配置文件路径为\${INSTALL_DIR}/opp/built-in/op_impl/ai_core/tbe/impl_mode/high_performance_for_all.ini，该文件中列表后续可能会跟随版本更新。 该实现模式不保证兼容，如果后续新的软件包中有算子新增了实现模式（即配置文件中新增了某个算子的实现模式），之前版本使用high_performance_for_all的网络模型，在新版本上精度可能会下降。
ACL_OPTYPELIST_FOR_IMPLMODE	通过ACL_OPTYPELIST_FOR_IMPLMODE选项设置算子类型的列表（多个算子使用英文逗号进行分隔），与ACL_OP_SELECT_IMPL_MODE选项配合使用，设置列表中的算子通过高精度实现或高性能实现。

编译选项	取值说明
ACL_OP_DEBUG_LEVEL	用于配置TBE算子编译debug功能开关。 0: 不开启算子debug功能, 在当前执行路径不生成算子编译目录kernel_meta。默认为0。 1: 开启算子debug功能, 在当前执行路径生成kernel_meta文件夹, 并在该文件夹下生成.o (算子二进制文件)、.json文件 (算子描述文件) 以及TBE指令映射文件 (算子cce文件*.cce和python-cce映射文件*_loc.json), 用于后续分析AICore Error问题。 2: 开启算子debug功能, 在当前执行路径生成kernel_meta文件夹, 并在该文件夹下生成.o (算子二进制文件)、.json文件 (算子描述文件) 以及TBE指令映射文件 (算子cce文件*.cce和python-cce映射文件*_loc.json), 用于后续分析AICore Error问题, 同时设置为2, 还会关闭编译优化开关、开启ccec调试功能 (ccec编译器选项设置为-O0-g)。 3: 不开启算子debug功能, 在当前执行路径生成kernel_meta文件夹, 并在该文件夹中生成.o (算子二进制文件) 和.json文件 (算子描述文件), 分析算子问题时可参考。 4: 不开启算子debug功能, 在当前执行路径生成kernel_meta文件夹, 并在该文件夹下生成.o (算子二进制文件) 和.json文件 (算子描述文件) 以及TBE指令映射文件 (算子cce文件*.cce) 和UB融合计算描述文件 ({kernel_name}_compute.json), 可在分析算子问题时进行问题复现、精度比对时使用。 说明 配置为2 (即开启ccec编译选项) 时, 会导致算子Kernel (*.o文件) 大小增大。动态Shape场景下, 由于算子编译时会遍历可能的Shape场景, 因此可能会导致算子Kernel文件过大而无法进行编译, 此种场景下, 建议不要配置ccec编译选项。 由于算子Kernel文件过大而无法编译的报错日志示例如下: message:link error ld.lld: error: InputSection too large for range extension thunk ./kernel_meta_xxxxx.o
ACL_DEBUG_DIR	用于配置保存模型转换、网络迁移过程中算子编译生成的调试相关过程文件的路径 (默认路径: 执行应用的当前路径/kernel_meta), 包括算子.o/.json/.cce等文件。具体生成哪些文件以ACL_OP_DEBUG_LEVEL选项设置的取值为准。 路径支持大小写字母 (a-z, A-Z)、数字 (0-9)、下划线 (_)、中划线 (-)、句点 (.)、中文字符。 说明 关于配置算子编译文件的缓存目录, 除此处设置枚举值的方式, 还可以配置环境变量ASCEND_WORK_PATH, 几种方式的优先级为: 本节设置枚举值的方式 > 设置环境变量 > 默认路径。 环境变量的详细配置说明请参见《环境变量参考》。

编译选项	取值说明
ACL_OP_COMPILER_CACHE_MODE	用于配置算子编译磁盘缓存模式。该编译选项需要与ACL_OP_COMPILER_CACHE_DIR配合使用。 • enable: 表示启用算子编译缓存。启用后可以避免针对相同编译参数及算子参数的算子重复编译, 从而提升编译速度。 • force: 启用算子编译缓存功能, 区别于enable模式, force模式下会强制刷新缓存, 即先删除已有缓存, 再重新编译并加入缓存。比如当用户的python变更、依赖库变更、算子调优后知识库变更等, 需要先指定为force用于先清理已有的缓存, 后续再修改为enable模式, 以避免每次编译时都强制刷新缓存。 • disable: 表示禁用算子编译缓存, 算子重新编译。 若同时打开调试开关(将ACL_OP_DEBUG_LEVEL选项配置为非0值), 则系统会忽略ACL_OP_COMPILER_CACHE_MODE选项的配置, 对调试场景的编译结果不做缓存。 启用算子编译缓存功能时, 可以通过配置文件(编译算子时在ACL_OP_COMPILER_CACHE_DIR选项指定的路径下自动生成op_cache.ini配置文件)、环境变量两种方式来设置缓存文件夹的磁盘空间大小: 1. 通过配置文件op_cache.ini设置 若op_cache.ini文件不存在, 则需要手动创建。打开该文件, 增加如下信息: #配置文件格式, 必须包含, 自动生成的文件中默认包括如下信息, 手动创建时, 需要输入 [op_compiler_cache] #限制某个芯片下缓存文件夹的磁盘空间的大小, 取值需为整数, 单位为MB max_op_cache_size=500 #设置需要保留缓存的空间大小比例, 取值范围: [1,100], 单位为百分比; 例如80表示缓存空间不足时, 删除缓存, 保留80% remain_cache_size_ratio=80 • 上述文件中的max_op_cache_size和remain_cache_size_ratio参数取值都有效时, op_cache.ini文件才会生效。 • 当编译缓存文件大小超过“max_op_cache_size”的设置值, 且超过半小时缓存文件未被访问时, 缓存文件就会老化(算子编译时, 不会因为编译缓存文件大小超过设置值而中断, 所以当“max_op_cache_size”设置过小时, 会出现实际编译缓存文件大小超过此设置值的情况)。 • 若需要关闭编译缓存老化功能, 可将“max_op_cache_size”设置为“-1”, 此时访问算子缓存时不会更新访问时间, 算子编译缓存不会老化, 磁盘空间使用默认大小500M。 • 若多个使用者使用相同的缓存路径, 建议使用配置文件的方式进行设置, 该场景下op_cache.ini文件会影响所有使用者。 2. 通过环境变量设置

编译选项	取值说明
	该场景下，开发者可以通过环境变量 <code>ASCEND_MAX_OP_CACHE_SIZE</code> 来限制某个芯片下缓存文件夹的磁盘空间的大小，当编译缓存空间大小达到 <code>ASCEND_MAX_OP_CACHE_SIZE</code> 设置的取值，且超过半个小时缓存文件未被访问时，缓存文件就会老化。可通过环境变量 <code>ASCEND_REMAIN_CACHE_SIZE_RATIO</code> 设置需要保留缓存的空间大小比例。 配置示例如下： <pre># ASCEND_MAX_OP_CACHE_SIZE环境变量默认值为500，取值需为整数，单位为MB export ASCEND_MAX_OP_CACHE_SIZE=500 # ASCEND_REMAIN_CACHE_SIZE_RATIO环境变量取值范围：[1,100]，默认值为50，单位为百分比；例如80表示缓存空间不足时，删除缓存，保留80% export ASCEND_REMAIN_CACHE_SIZE_RATIO=50</pre> - 通过环境变量配置，只对当前用户生效。 - 若需要关闭编译缓存老化功能，可将环境变量“<code>ASCEND_MAX_OP_CACHE_SIZE</code>”设置为“-1”，此时访问算子缓存时不会更新访问时间，算子编译缓存不会老化，磁盘空间使用默认大小500M。 若同时配置了 <code>op_cache.ini</code> 文件和环境变量，则优先读取 <code>op_cache.ini</code> 文件中的配置项，若 <code>op_cache.ini</code> 文件和环境变量都未设置，则读取系统默认值：默认磁盘空间大小500M，默认保留缓存的空间50%。
<code>ACL_OP_COMPILER_CACHE_DIR</code>	用于配置算子编译文件的缓存目录（默认路径：<code>\$HOME/atc_data</code>）。该编译选项需要与 <code>ACL_OP_COMPILER_CACHE_MODE</code> 配合使用。 路径支持大小写字母（a-z, A-Z）、数字（0-9）、下划线（_）、中划线（-）、句点（.）、中文字符。 如果设置了 <code>ACL_OP_DEBUG_LEVEL</code> 编译选项，则只有编译选项值为0才会启用编译缓存功能，其它取值禁用编译缓存功能。 说明 关于配置算子编译文件的缓存目录，除此设置枚举值的方式，还可以配置环境变量 <code>ASCEND_CACHE_PATH</code>，几种方式的优先级为：本节设置枚举值的方式 > 设置环境变量 > 默认路径。 环境变量的详细配置说明请参见《环境变量参考》。

编译选项	取值说明
ACL_OP_PERFORMANCE_MODE	该参数已废弃，请勿配置，否则后续版本可能存在兼容性问题。 通过该选项设置是否按照算子执行高性能的方式编译算子，默认采用normal方式。 取值范围： • normal：算子编译时按照编译性能最高的方式进行编译。 • high：算子编译时应该按照算子执行性能最好的方式去泛化编译。
ACL_OP_JIT_COMPILE	选择是在线编译算子，还是使用已编译的算子二进制文件。 • enable：在线编译算子，系统根据得到的算子信息进行优化，从而编译出运行性能更优的算子。固定Shape网络场景下，建议设置为enable。 • disable：优先查找系统中的已编译好的算子二进制文件，如果能查找到，则不再编译算子，编译性能更优；如果查找不到，则再编译算子。动态Shape网络场景下，建议设置为disable。若将本参数设置为disable，则需要安装算子二进制文件包，请参见《CANN软件安装指南》中的“常用操作 > 安装、升级和卸载二进制算子包”章节。 Atlas 200/300/500 推理产品，该选项默认值为enable。 Atlas 训练系列产品，该选项默认值为enable。 Atlas 推理系列产品，该选项默认值为enable。 Atlas 200/500 A2推理产品，该选项默认值为enable。 Atlas A2训练系列产品/Atlas 800I A2推理产品，该选项默认值为disable。
ACL_OP_DETERMINISTIC	是否开启确定性计算。 • 0：默认值，不开启确定性计算。配置该选项时，算子在相同的硬件和输入下，多次执行的结果可能不同。这个差异的来源，一般是因为在算子实现中，存在异步的多线程执行，会导致浮点数累加的顺序变化。 • 1：开启确定性计算。配置该选项时，算子在相同的硬件和输入下，多次执行将产生相同的输出。但启用确定性计算往往导致算子执行变慢。 通常建议不开启确定性计算，因为确定性计算往往会导致算子执行变慢，进而影响性能。当发现模型多次执行结果不同，或者是进行精度调优时，可开启确定性计算，辅助模型调试、调优。

编译选项	取值说明
ACL_CUSTOMIZE_DTYPES	*.cfg配置文件路径，包含文件名，配置文件中列举需要指定计算精度的算子名称或算子类型，每个算子单独一行。通过该配置，在模型编译时，可自定义某个或某些算子的计算精度。 配置约束： • 路径和文件名支持大小写字母（a-z，A-Z）、数字（0-9）、下划线（_）、中划线（-）、句点（.）、英文冒号(:)、中文字符。 • 配置文件中若为算子名称，以 Opname::InputDtype:dtype1,...,OutputDtype:dtype1,...格式进行配置，每个Opname单独一行，dtype1，dtype2..需要与可设置计算精度的算子输入，算子输出的个数一一对应。 • 配置文件中若为算子类型，以 OpType::TypeName:InputDtype:dtype1,...,OutputDtype:dtype1,...格式进行配置，每个OpType单独一行，dtype1，dtype2..需要与可设置计算精度的算子输入，算子输出的个数一一对应，且算子OpType必须为基于Ascend IR定义的算子的OpType，OpType可查阅《算子加速库接口参考》。 • 对于同一个算子，如果同时配置了Opname和OpType的配置项，编译时以Opname的配置项为准。 • 使用该参数指定某个算子的计算精度时，如果模型转换过程中该算子被融合掉，则该算子指定的计算精度不生效。

编译选项	取值说明
ACL_OP_PRECISION_MODE	设置算子精度模式的配置文件（.ini格式）路径以及文件名，路径和文件名：支持大小写字母（a-z，A-Z）、数字（0-9）、下划线（_）、中划线（-）、句点（.）、中文字符。 - 配置文件中支持设置如下精度模式： - high_precision：表示高精度。 - high_performance：表示高性能。 - enable_float_32_execution：算子内部计算时使用FP32数据类型。若使用HF32计算，精度损失超过预期时，可启用该配置，指定部分算子内部计算时使用FP32，保持精度。 Atlas A2训练系列产品/Atlas 800I A2推理产品支持此配置。 - enable_hi_float_32_execution：算子内部计算时使用HF32数据类型，启用该配置，FP32数据类型自动转换为HF32数据类型。通过该配置降低数据所占空间大小，实现性能提升。当前版本暂不支持。 - support_out_of_bound_index：表示对gather、scatter和segment类算子的indices输入进行越界校验，校验会降低算子的执行性能。 - 构造算子精度模式配置文件<code>op_precision.ini</code>，并在该文件中按照算子类型、节点名称设置精度模式，每一行设置一个算子类型或节点名称的精度模式，按节点名称设置精度模式的优先级高于按算子类型。 配置样例如下： <pre>[ByOpType] optype1=high_precision optype2=high_performance optype3=support_of_bound_index [ByNodeName] nodename1=high_precision nodename2=high_performance nodename3=support_of_bound_index</pre>

编译选项	取值说明
ACL_ALLOW_HF32	当前版本不支持该参数。 算子内部计算时是否允许HF32类型替换FP32类型，true允许，false不允许，当前版本该配置仅针对Conv类算子与Matmul类算子生效。默认针对Conv类算子，启用FP32转换为HF32；默认针对Matmul类算子，关闭FP32转换为HF32。 HF32是昇腾推出的专门用于算子内部计算的单精度浮点类型，与其他常用数据类型的比较如下图所示。HF32与FP32支持相同的数值范围，但尾数位精度（11位）却接近FP16（10位）。通过降低精度让HF32单精度数据类型代替原有的FP32单精度数据类型，可大大降低数据所占空间大小，实现性能的提升。 说明 - 配置为true时，哪些算子内部计算时允许HF32类型替换FP32类型，可在“CANN软件安装路径/opp/built-in/op_impl/ai_core/tbe/impl_mode/allow_hf32_matmul_t_conv_t.ini”文件中查看； - 配置为false时，哪些算子内部计算时不允许HF32类型替换FP32类型，可在“CANN软件安装路径/opp/built-in/op_impl/ai_core/tbe/impl_mode/allow_hf32_matmul_f_conv_f.ini”文件中查看。 - 针对同一个算子，使用ACL_OP_PRECISION_MODE选项中的enable_hi_float_32_execution或enable_float_32_execution时，不建议与ACL_ALLOW_HF32选项同时使用，若同时使用，则优先级如下： op_precision_mode(ByNodeName, 按节点名称设置精度模式) > allow_hf32 > op_precision_mode(ByOpType, 按算子类型设置精度模式)。

编译选项	取值说明
ACL_PRECISION_MODE_V2	网络模型的算子精度模式，如果不配置该编译选项，默认采用fp16。 相比ACL_PRECISION_MODE选项，ACL_PRECISION_MODE_V2是新版本中新增的，精度模式选项值语义更清晰，便于理解。 • fp16: 算子支持float16和float32数据类型时，强制选择float16。 • origin: 保持原图精度。 - 如果原图中某算子精度为float16，AI Core中该算子的实现不支持float16、仅支持float32和bfloat16，则系统内部会自动采用高精度float32。 - 如果原图中某算子精度为float16，AI Core中该算子的实现不支持float16、仅支持bfloat16，则会使用float16的AI CPU算子；如果AI CPU算子也不支持，则执行报错。 - 如果原图中某算子精度为float32，AI Core中该算子的实现不支持float32类型、仅支持float16类型，则会使用float32的AI CPU算子；如果AI CPU算子也不支持，则执行报错。 bfloat16类型仅在以下产品型号支持： Atlas A2训练系列产品/Atlas 800I A2推理产品 Atlas 200/500 A2推理产品 • cube_fp16in_fp32out: 算子既支持float32又支持float16数据类型时，系统内部根据算子类型不同，选择不同的处理方式。 - 对于矩阵计算类算子，系统内部会按算子实现的支持情况处理： 1. 优先选择输入数据类型为float16且输出数据类型为float32； 2. 如果1中的场景不支持，则选择输入数据类型为float32且输出数据类型为float32； 3. 如果2中的场景不支持，则选择输入数据类型为float16且输出数据类型为float16； 4. 如果3中的场景不支持，则报错。 - 对于向量计算类算子，表示网络模型中算子支持float16和float32时，强制选择float32，若原图精度为float16，也会强制转为float32。 如果网络模型中存在部分算子，并且该算子实现不支持float32，比如某算子仅支持float16类型，则该参数不生效，仍然使用支持的float16；如果该算子不支持float32，且又配置了黑名单（precision_reduce = false），则会使用float32的

编译选项	取值说明
	AI CPU算子；如果AI CPU算子也不支持，则执行报错。 • mixed_float16: 表示使用混合精度float16和float32数据类型来处理神经网络。针对网络模型中float32数据类型的算子，按照内置的优化策略，自动将部分float32的算子降低精度到float16，从而在精度损失很小的情况下提升系统性能并减少内存使用。 若配置了该种模式，则可以在OPP软件包安装路径\${INSTALL_DIR}/opp/built-in/op_impl/ai_core/tbe/config/<soc_version>/aic-<soc_version>-ops-info.json内置优化策略文件中查看“precision_reduce”参数的取值： - 若取值为true（白名单），则表示允许将当前float32类型的算子，降低精度到float16。 - 若取值为false（黑名单），则不允许将当前float32类型的算子降低精度到float16，相应算子仍旧使用float32精度。 - 若网络模型中算子没有配置该参数（灰名单），当前算子的混合精度处理机制和前一个算子保持一致，即如果前一个算子支持降精度处理，当前算子也支持降精度；如果前一个算子不允许降精度，当前算子也不支持降精度。 • mixed_bfloat16: 表示使用混合精度bfloat16和float32数据类型来处理神经网络。针对网络模型中float32数据类型的算子，按照内置的优化策略，自动将部分float32的算子降低精度到bfloat16，从而在精度损失很小的情况下提升系统性能并减少内存使用；如果算子不支持bfloat16和float32，则使用AI CPU算子进行计算；如果AI CPU算子也不支持，则执行报错。 若配置了该种模式，则可以在OPP软件包安装路径\${INSTALL_DIR}/opp/built-in/op_impl/ai_core/tbe/config/<soc_version>/aic-<soc_version>-ops-info.json内置优化策略文件中查看“precision_reduce”参数的取值： - 若取值为true（白名单），则表示允许将当前float32类型的算子，降低精度到bfloat16。 - 若取值为false（黑名单），则不允许将当前float32类型的算子降低精度到bfloat16，相应算子仍旧使用float32精度。 - 若网络模型中算子没有配置该参数（灰名单），当前算子的混合精度处理机制和前一个算子保持一致，即如果前一个算子支持降精度处理，当前算子也支持降精度；如果前一个算子不允许降精度，当前算子也不支持降精度。 该选项仅在以下产品型号支持：

编译选项	取值说明
	Atlas A2训练系列产品/Atlas 800I A2推理产品 Atlas 200/500 A2推理产品
ACL_OP_DEBUG_OPTION	当前仅支持配置为oom，表示开启Global Memory访问越界检测。 执行算子过程中，涉及从Global Memory中读写数据，例如读算子输入数据、写算子输出数据等，配置oom选项开启Global Memory访问越界检测，实际执行算子时，若出现内存越界，AscendCL会返回“EZ9999”错误码。

1.13.28 acldvppChannelDescParamType

```
enum acldvppChannelDescParamType {  
    ACL_DVPP_CSC_MATRIX_UINT32 = 0, // 色域转换矩阵属性，该属性类型对应的值为uint32_t类型  
    ACL_DVPP_MODE_UINT32,           // 通道描述信息中的通道模式，该属性类型对应的值为uint32_t类型  
    ACL_DVPP_CHANNEL_ID_UINT64,      // 通道ID，该属性类型对应的值为uint64_t类型  
    ACL_DVPP_CHANNEL_HEIGHT_UINT32,  // 通道的高度，该属性类型对应的值为uint32_t类型  
    ACL_DVPP_CHANNEL_WIDTH_UINT32,   // 通道的宽度，该属性类型对应的值为uint32_t类型  
    ACL_DVPP_JPEGD_PRECISION_MODE_ENUM // JPEGD解码输出图片的宽、高对齐模式，该属性类型对应的值为int32_t类型  
}
```

表 1-105 属性配置

属性类型	取值说明
ACL_DVPP_CSC_MATRIX_UINT32	色域转换矩阵属性，属性值请参见 acldvppCscMatrix 。 Atlas 200/300/500 推理产品，不支持该选项。 Atlas 训练系列产品，不支持该选项。
ACL_DVPP_MODE_UINT32	通道描述信息中的通道模式，用于明确图片数据处理通道用于实现哪种功能，目前支持 VPC 、 JPEGD 、 JPEGG 、 PNGD 功能。属性值请参见 acldvppChannelMode 。 Atlas 200/300/500 推理产品，不支持该选项。 Atlas 训练系列产品，不支持该选项。
ACL_DVPP_CHANNEL_ID_UINT64	通道ID。

属性类型	取值说明
ACL_DVPP_CHANNEL_HEIGHT_UINT32	通道的高度，目前仅支持在JPEGE功能中使用该属性，其它功能中设置该属性无效。 Atlas 200/300/500 推理产品，不支持该选项。 Atlas 训练系列产品，不支持该选项。 Atlas 推理系列产品上，默认值和最大值为8192。 Atlas 200/500 A2推理产品上，默认值为8192，最大值为16384，若输入图片高大于8192，则需要调用acldvppSetChannelDescParam接口设置通道高度。 Atlas A2训练系列产品/Atlas 800I A2推理产品上，默认值和最大值为8192。
ACL_DVPP_CHANNEL_WIDTH_UINT32	通道的宽度，目前仅支持在JPEGE功能中使用该属性，其它功能中设置该属性无效。 Atlas 200/300/500 推理产品，不支持该选项。 Atlas 训练系列产品，不支持该选项。 Atlas 推理系列产品上，默认值和最大值为8192。 默认值为8192，最大值为16384，若输入图片宽大于8192，则需要调用acldvppSetChannelDescParam接口设置通道宽度。 Atlas A2训练系列产品/Atlas 800I A2推理产品上，默认值和最大值为8192。
ACL_DVPP_JPEGD_PRECISION_MODE_ENUM	JPEGD解码输出图片的宽、高对齐模式，模式选项参见1.13.46 acldvppJpegdPrecisionMode。 目前仅支持在JPEGE功能中使用该属性，其它功能中设置该属性无效。 Atlas 200/300/500 推理产品，不支持该选项。 Atlas 训练系列产品，不支持该选项。

1.13.29 aclvdecChannelDescParamType

```
enum aclvdecChannelDescParamType {  
    ACL_VDEC_CSC_MATRIX_UINT32 = 0, // 色域转换矩阵属性，该属性类型对应的值为uint32_t类型  
    ACL_VDEC_OUT_MODE_UINT32, // 是否实时出帧，该属性类型对应的值为uint32_t类型  
    ACL_VDEC_THREAD_ID_UINT64, // 回调线程ID，该属性类型对应的值为uint64_t类型  
    ACL_VDEC_CALLBACK_PTR, // 回调函数，该属性类型对应的值是内存指针  
    ACL_VDEC_CHANNEL_ID_UINT32, // 通道号，该属性类型对应的值为uint32_t类型  
    ACL_VDEC_ENCODE_TYPE_UINT32, // 视频编码协议，该属性类型对应的值为uint32_t类型  
    ACL_VDEC_OUT_PIC_FORMAT_UINT32, // 输出图片格式，该属性类型对应的值为uint32_t类型  
    ACL_VDEC_OUT_PIC_WIDTH_UINT32, // 解码码流最大宽度，该属性类型对应的值为uint32_t类型  
    ACL_VDEC_OUT_PIC_HEIGHT_UINT32, // 解码码流最大高度，该属性类型对应的值为uint32_t类型  
    ACL_VDEC_REF_FRAME_NUM_UINT32, // 参考帧数量，该属性类型对应的值为uint32_t类型  
    ACL_VDEC_BIT_DEPTH_UINT32 // 视频位宽，该属性类型对应的值为uint32_t类型  
}
```

表 1-106 属性配置

属性类型	取值说明
ACL_VDEC_CSC_MATRIX_UINT32	色域转换矩阵属性，属性值请参见 aclvppCscMatrix 。 Atlas 200/300/500 推理产品，不支持该选项。 Atlas 训练系列产品，不支持该选项。
ACL_VDEC_OUT_MODE_UINT32	设置是否实时出帧（即发送一帧解码一帧，无需依赖后续帧的传入）。 取值范围如下： 0：默认出帧模式，由于解码过程中存在缓存帧，无法实时输出，因此VDEC需要在收到码流中的多帧数据后，才开始输出解码结果。 1：快速出帧模式，VDEC获取码流中的一帧数据后，就开始实时输出解码结果。只支持简单参考关系的H264/H265标准码流（无长期参考帧，无B帧）。
ACL_VDEC_THREAD_ID_UINT64	回调线程ID。 说明 同一个进程内，在不同的Device上注册VDEC解码回调函数的线程时，不能指定同一个线程ID。 同一个Device上，多路VDEC解码可以指定同一个线程ID，但相比一个线程处理一路VDEC解码任务来说，一个线程内串行处理多路VDEC解码任务时，VDEC解码性能可能下降。
ACL_VDEC_CALLBACK_PTR	解码回调函数。 请参见 1.8.9.5 aclvdecCallback 。
ACL_VDEC_CHANNEL_ID_UINT32	通道ID。 Atlas 200/300/500 推理产品，该参数值的取值范围[0, 31]。 Atlas 训练系列产品，该参数值的取值范围[0, 31]。 Atlas 推理系列产品，该参数值的取值范围[0, 255]。 Atlas 200/500 A2推理产品，该参数值的取值范围[0, 255]。 Atlas A2训练系列产品/Atlas 800I A2推理产品，该参数值的取值范围[0, 255]。
ACL_VDEC_ENCODE_TYPE_UINT32	视频编码协议H265-main level（0）、H264-baseline level（1）、H264-main level(2)、H264-high level（3）。 请参见 aclvppStreamFormat 。
ACL_VDEC_OUT_PIC_FORMAT_UINT32	输出图像存储格式，具体支持哪些格式请参见 码流/图片格式、宽高对齐、内存约束 。 如果不设置输出格式，默认使用YUV420SP NV12。

属性类型	取值说明
ACL_VDEC_OUT_PIC_WIDTH_UINT32	解码码流最大宽度。 Atlas 200/300/500 推理产品，如果不设置解码码流最大宽度，内部默认使用1920。 Atlas 训练系列产品，如果不设置解码码流最大宽度，内部默认使用1920。 Atlas 推理系列产品，如果不设置解码码流最大宽度，内部默认使用4096。 Atlas 200/500 A2推理产品，如果不设置解码码流最大宽度，内部默认使用4096。 Atlas A2训练系列产品/Atlas 800I A2推理产品，如果不设置解码码流最大宽度，内部默认使用4096。
ACL_VDEC_OUT_PIC_HEIGHT_UINT32	解码码流最大高度。 Atlas 200/300/500 推理产品，如果不设置解码码流最大高度，内部默认使用1120。 Atlas 训练系列产品，如果不设置解码码流最大高度，内部默认使用1120。 Atlas 推理系列产品，如果不设置解码码流最大高度，内部默认使用4096。 Atlas 200/500 A2推理产品，如果不设置解码码流最大高度，内部默认使用4096。 Atlas A2训练系列产品/Atlas 800I A2推理产品，如果不设置解码码流最大高度，内部默认使用4096。
ACL_VDEC_REF_FRAME_NUM_UINT32	参考帧数量，取值范围[0, 16]。 Atlas 200/300/500 推理产品，当前版本不支持用户设置该参数。 Atlas 训练系列产品，当前版本不支持用户设置该参数。 Atlas 推理系列产品，如果不设置参考帧数量，内部默认使用8。 Atlas 200/500 A2推理产品，如果不设置参考帧数量，内部默认使用8。 Atlas A2训练系列产品/Atlas 800I A2推理产品，如果不设置参考帧数量，内部默认使用8。
ACL_VDEC_BIT_DEPTH_UINT32	设置视频位宽，默认值为10-bit。 取值范围如下： 0: 8-bit 1: 10-bit(默认值) Atlas 200/300/500 推理产品，当前版本不支持用户设置该参数。 Atlas 训练系列产品，当前版本不支持用户设置该参数。

1.13.30 aclvencChannelDescParamType

Atlas 训练系列产品，不支持该枚举值。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该枚举值。

```
enum aclvencChannelDescParamType {
    ACL_VENC_THREAD_ID_UINT64 = 0, // 回调线程ID，该属性类型对应的值为uint64_t类型
    ACL_VENC_CALLBACK_PTR, // 回调函数，该属性类型对应的值是内存指针
    ACL_VENC_PIXEL_FORMAT_UINT32, // 输入图像格式，该属性类型对应的值为uint32_t类型
    ACL_VENC_ENCODE_TYPE_UINT32, // 视频编码协议，该属性类型对应的值为uint32_t类型
    ACL_VENC_PIC_WIDTH_UINT32, // 输入图片宽度，该属性类型对应的值为uint32_t类型
    ACL_VENC_PIC_HEIGHT_UINT32, // 输入图片高度，该属性类型对应的值为uint32_t类型
    ACL_VENC_KEY_FRAME_INTERVAL_UINT32, // 关键帧间隔，该属性类型对应的值为uint32_t类型
    ACL_VENC_BUF_ADDR_PTR, // 编码输出缓存地址
    ACL_VENC_BUF_SIZE_UINT32, // 编码输出缓存大小，该属性类型对应的值为uint32_t类型
    ACL_VENC_RC_MODE_UINT32, // 码率控制模式，该属性类型对应的值为uint32_t类型
    ACL_VENC_SRC_RATE_UINT32, // 输入码流帧率，该属性类型对应的值为uint32_t类型
    ACL_VENC_MAX_BITRATE_UINT32, // 输出码率，该属性类型对应的值为uint32_t类型
    ACL_VENC_MAX_IP_PROP_UINT32 // 一个GOP内单个I帧bit数和单个P帧bit数的比例，该属性类型对应的值为uint32_t类型
}
```

表 1-107 属性配置

属性类型	取值说明
ACL_VENC_THREAD_ID_UINT64	回调线程ID。 说明 同一个进程内，在不同的Device上注册VENC编码回调函数的线程时，不能指定同一个线程ID。
ACL_VENC_CALLBACK_PTR	编码回调函数。
ACL_VENC_PIXEL_FORMAT_UINT32	输入图像格式，支持如下格式： - PIXEL_FORMAT_YUV_SEMIPLANAR_420 - PIXEL_FORMAT_YVU_SEMIPLANAR_420
ACL_VENC_ENCODE_TYPE_UINT32	视频编码协议。 参见 1.13.7 acldvppStreamFormat 。
ACL_VENC_PIC_WIDTH_UINT32	图片宽度。
ACL_VENC_PIC_HEIGHT_UINT32	图片高度。
ACL_VENC_KEY_FRAME_INTERVAL_UINT32	关键帧间隔，取值范围[1,65536]。
ACL_VENC_BUF_ADDR_PTR	编码输出缓存指针。 说明 在Atlas 200/300/500 推理产品上，当前版本不支持用户设置该参数，由VENC内部管理该缓存地址。

属性类型	取值说明
ACL_VENC_BUF_SIZE_UINT32	编码输出缓存大小，单位为Byte。 说明 在Atlas 200/300/500 推理产品上，当前版本不支持用户设置该参数，内部默认使用3686400 Byte。 在Atlas 推理系列产品上，如果不设置该参数，参数值默认为8M；如果设置该参数，参数值最小为5M。 Atlas 200/500 A2推理产品，如果不设置该参数，参数值默认为8M；如果设置该参数，参数值最小为5M。 当前版本获取该参数的值为无效值，不支持获取该参数值。
ACL_VENC_RC_MODE_UINT32	指定码率控制模式。 ● 0表示使用默认值 ● 1表示变码率VBR模式 ● 2表示定码率CBR模式 说明 如果不设置该参数，则采用默认值0。 Atlas 200/300/500 推理产品，默认值0表示CBR模式。 Atlas 推理系列产品，默认值0表示VBR模式。 Atlas 200/500 A2推理产品，默认值0表示VBR模式。
ACL_VENC_SRC_RATE_UINT32	输入码流帧率，单位fps。 Atlas 200/300/500 推理产品，取值范围0或者[1,120]。 Atlas 推理系列产品，取值范围0或者[1,240]。 Atlas 200/500 A2推理产品，取值范围0或者[1,240]。 如果不设置该参数，默认为30；如果设置为0，表示使用默认值，即30。如果该值和实际输入码流帧率相差太大，会影响输出码率。
ACL_VENC_MAX_BITRATE_UINT32	输出码率，单位kbps。 Atlas 200/300/500 推理产品，取值范围0或[10,30000]。如果不设置该参数，默认为300；如果设置为0，表示使用默认值，即300。 Atlas 推理系列产品，取值范围[2,614400]，如果不设置该参数，默认为2000；如果设置为0，表示使用默认值，即2000。 Atlas 200/500 A2推理产品，取值范围[2,614400]，如果不设置该参数，默认为2000；如果设置为0，表示使用默认值，即2000。
ACL_VENC_MAX_IP_PROP_UINT32	一个GOP内单个I帧bit数和单个P帧bit数的比例，取值范围0或[1,100]。如果不设置该参数，VBR模式下此值默认为80，CBR模式下此值默认为70；如果设置为0，表示使用默认值，即VBR模式下此值默认为80，CBR模式下此值默认为70。

1.13.31 aclmdlConfigAttr

```
typedef enum {
    ACL_MDL_PRIORITY_INT32 = 0,
    ACL_MDL_LOAD_TYPE_SIZE_T,
    ACL_MDL_PATH_PTR,
    ACL_MDL_MEM_ADDR_PTR,
    ACL_MDL_MEM_SIZE_T,
    ACL_MDL_WEIGHT_ADDR_PTR,
    ACL_MDL_WEIGHT_SIZE_T,
    ACL_MDL_WORKSPACE_ADDR_PTR,
    ACL_MDL_WORKSPACE_SIZE_T,
    ACL_MDL_INPUTQ_NUM_SIZE_T,
    ACL_MDL_INPUTQ_ADDR_PTR,
    ACL_MDL_OUTPUTQ_NUM_SIZE_T,
    ACL_MDL_OUTPUTQ_ADDR_PTR,
    ACL_MDL_WORKSPACE_MEM_OPTIMIZE,
    ACL_MDL_WEIGHT_PATH_PTR,
    ACL_MDL_MODEL_DESC_PTR,
    ACL_MDL_MODEL_DESC_SIZE_T,
    ACL_MDL_KERNEL_PTR,
    ACL_MDL_KERNEL_SIZE_T,
    ACL_MDL_KERNEL_ARGS_PTR,
    ACL_MDL_KERNEL_ARGS_SIZE_T,
    ACL_MDL_STATIC_TASK_PTR,
    ACL_MDL_STATIC_TASK_SIZE_T,
    ACL_MDL_DYNAMIC_TASK_PTR,
    ACL_MDL_DYNAMIC_TASK_SIZE_T,
    ACL_MDL_MEM_MALLOCPOLICY_SIZE_T
} aclmdlConfigAttr;
```

表 1-108 模型加载选项配置

选项	取值说明
ACL_MDL_PRIORITY_INT32	模型执行的优先级，可选项。该选项对应的值为int32类型。 数字越小优先级越高，取值[0,7]，默认值为0。 Atlas 训练系列产品，docker容器内安装CANN，且配置算力分组的场景，该枚举值不生效。

选项	取值说明
ACL_MDL_LOAD_TYPE_SIZE_T	模型加载方式，必选项。该选项对应的值为size_t类型。 ACL_MDL_LOAD_TYPE_SIZE_T（表示模型加载方式）的取值如下： • ACL_MDL_LOAD_FROM_FILE #define ACL_MDL_LOAD_FROM_FILE 1 • ACL_MDL_LOAD_FROM_FILE_WITH_MEM #define ACL_MDL_LOAD_FROM_FILE_WITH_MEM 2 • ACL_MDL_LOAD_FROM_MEM #define ACL_MDL_LOAD_FROM_MEM 3 • ACL_MDL_LOAD_FROM_MEM_WITH_MEM #define ACL_MDL_LOAD_FROM_MEM_WITH_MEM 4 • ACL_MDL_LOAD_FROM_FILE_WITH_Q #define ACL_MDL_LOAD_FROM_FILE_WITH_Q 5 • ACL_MDL_LOAD_FROM_MEM_WITH_Q #define ACL_MDL_LOAD_FROM_MEM_WITH_Q 6 注意：如果将ACL_MDL_LOAD_TYPE_SIZE_T设置为ACL_MDL_LOAD_FROM_MEM，表示从内存加载模型数据，还支持使用ACL_MDL_WEIGHT_PATH_PTR选项指定权重文件目录。
ACL_MDL_PATH_PTR	离线模型文件路径的指针，如果选择从文件加载模型，则该选项必选。
ACL_MDL_MEM_ADDR_PTR	模型在内存中的起始地址，如果选择从内存加载模型，则该选项必选。
ACL_MDL_MEM_SIZE_T	模型在内存中的大小，如果选择从内存加载模型，则该选项必选，与ACL_MDL_MEM_ADDR_PTR选项配合使用。该选项对应的值为size_t类型。
ACL_MDL_WEIGHT_ADDR_PTR	Device上模型权值内存（存放权值数据）的指针，如果需要由用户管理权值内存，则该选项必选。若不配置该选项，则表示由系统管理内存。
ACL_MDL_WEIGHT_SIZE_T	权值内存大小，单位为Byte，如果需要由用户管理权值内存，则该选项必选，与ACL_MDL_WEIGHT_ADDR_PTR选项配合使用。该选项对应的值为size_t类型。
ACL_MDL_WORKSPACE_ADDR_PTR	Device上模型所需工作内存（存放模型执行过程中的临时数据）的指针，如果需要由用户管理工作内存，则该选项必选。若不配置该选项，则表示由系统管理内存。
ACL_MDL_WORKSPACE_SIZE_T	模型所需工作内存的大小，单位为Byte，如果需要由用户管理工作内存，则该选项必选，与ACL_MDL_WORKSPACE_ADDR_PTR选项配合使用。该选项对应的值为size_t类型。

选项	取值说明
ACL_MDL_INPUTQ_NUM_SIZET	模型输入队列大小,带队列加载模型时,该选项必选,与ACL_MDL_INPUTQ_ADDR_PTR选项配合使用。该选项对应的值为size_t类型。
ACL_MDL_INPUTQ_ADDR_PTR	模型输入队列ID的指针,带队列加载模型时,该选项必选,一个模型输入对应一个队列ID。
ACL_MDL_OUTPUTQ_NUM_SIZET	模型输出队列大小,带队列加载模型时,该选项必选,与ACL_MDL_OUTPUTQ_ADDR_PTR选项配置使用。该选项对应的值为size_t类型。
ACL_MDL_OUTPUTQ_ADDR_PTR	模型输出队列ID的指针,带队列加载模型时,该选项必选,一个模型输出对应一个队列ID。
ACL_MDL_WORKSPACE_MEM_OPTIMIZE	是否开启模型工作内存优化功能,1开启,0不开启。 若关注内存规划或内存资源有限时,建议在模型加载前,开启工作内存优化功能,此时工作内存中不包含存放模型输入、输出数据的内存,工作内存大小会减小,达到节省内存的目的。 在模型执行前,还需要由用户申请存放模型输入、输出数据的内存,因此即使在模型加载时开启工作内存优化功能,也不会影响后续的模式执行。
ACL_MDL_WEIGHT_PATH_PTR	权重文件所在目录的指针。对om模型文件大小有限制的场景下,通过本参数可实现权重文件外置功能。 如果将ACL_MDL_LOAD_TYPE_SIZET设置为ACL_MDL_LOAD_FROM_MEM,表示从内存加载模型数据,则支持使用ACL_MDL_WEIGHT_PATH_PTR指定权重文件目录。 一般对om模型文件大小有限制或模型文件加密的场景下,需单独指定权重文件目录,但前提是需在使用ATC工具生成om文件时,将--external_weight参数设置为1(1表示将原始网络中的Const/Constant节点的权重保存在单独的文件中)。
ACL_MDL_MODEL_DESC_PTR	存放模型描述信息的内存指针。 当前版本不支持该配置。
ACL_MDL_MODEL_DESC_SIZET	存放模型描述信息所需的内存大小,单位Byte。该选项对应的值为size_t类型。 可提前调用aclmdlQueryExeOMDesc接口获取存放模型描述信息所需的内存大小,且本选项需与ACL_MDL_MODEL_DESC_PTR选项配合使用。 当前版本不支持该配置。
ACL_MDL_KERNEL_PTR	存放TBE算子kernel(算子的*.o与*.json)的内存指针。 当前版本不支持该配置。

选项	取值说明
ACL_MDL_KERNEL_SIZE_T	存放TBE算子kernel（算子的*.o与*.json）所需的内存大小，单位Byte。该选项对应的值为size_t类型。 可提前调用 aclmdlQueryExeOMDesc 接口获取存放TBE算子kernel（算子的*.o与*.json）所需的内存大小，且本选项需与ACL_MDL_KERNEL_PTR选项配合使用。 当前版本不支持该配置。
ACL_MDL_KERNEL_ARGS_PTR	存放TBE算子kernel参数的内存指针。 当前版本不支持该配置。
ACL_MDL_KERNEL_ARGS_SIZE_T	存放TBE算子kernel参数所需的内存大小，单位Byte。该选项对应的值为size_t类型。 可提前调用 aclmdlQueryExeOMDesc 接口获取存放TBE算子kernel参数所需的内存大小，且本选项需与ACL_MDL_KERNEL_ARGS_PTR选项配合使用。 当前版本不支持该配置。
ACL_MDL_STATIC_TASK_PTR	存放静态shape任务描述信息的内存指针。 当前版本不支持该配置。
ACL_MDL_STATIC_TASK_SIZE_T	存放静态shape任务描述信息所需的内存大小，单位Byte。该选项对应的值为size_t类型。 可提前调用 aclmdlQueryExeOMDesc 接口获取存放静态shape任务描述信息所需的内存大小，且本选项需与ACL_MDL_STATIC_TASK_PTR选项配合使用。 当前版本不支持该配置。
ACL_MDL_DYNAMIC_TASK_PTR	存放动态shape任务描述信息的内存指针。 当前版本不支持该配置。
ACL_MDL_DYNAMIC_TASK_SIZE_T	存放动态shape任务描述信息所需的内存大小，单位Byte。该选项对应的值为size_t类型。 可提前调用 aclmdlQueryExeOMDesc 接口获取存放动态shape任务描述信息所需的内存大小，且本选项需与ACL_MDL_DYNAMIC_TASK_PTR选项配合使用。 当前版本不支持该配置。
ACL_MDL_MEM_MALLOC_POLICY_SIZE_T	内存分配规则，该选项对应的值为size_t类型。 当前版本不支持该配置。

说明

关于如何获取om文件，请参见《[ATC工具使用指南](#)》。

1.13.32 aclMemType

```
typedef enum {  
    ACL_MEMTYPE_DEVICE = 0, //Device内存  
    ACL_MEMTYPE_HOST = 1,  //Host内存  
    ACL_MEMTYPE_HOST_COMPILE_INDEPENDENT = 2  //Host内存  
}
```

ACL_MEMTYPE_HOST和ACL_MEMTYPE_HOST_COMPILE_INDEPENDENT都标识Host内存，但在使用上有区别：

- ACL_MEMTYPE_HOST：若通过[aclSetCompileopt](#)接口将ACL_OP_JIT_COMPILE设置为disable，设置该选项时，算子输入或输出的值的变化，不会触发算子重新编译；若通过[aclSetCompileopt](#)接口将ACL_OP_JIT_COMPILE设置为enable，算子输入或输出的值的变化，会触发算子重新编译。
- ACL_MEMTYPE_HOST_COMPILE_INDEPENDENT：设置该选项时，算子输入或输出的值的变化，都不会触发算子重新编译。若算子编译时依赖其输入或输出的值，此时如果设置为 ACL_MEMTYPE_HOST_COMPILE_INDEPENDENT，则可能会导致编译失败。

1.13.33 aclOpCompileFlag

该参数已废弃，请勿配置，否则后续版本可能存在兼容性问题。

```
typedef enum aclCompileFlag {  
    ACL_OP_COMPILE_DEFAULT, //精确编译，不设置编译flag时，默认是精确编译  
    ACL_OP_COMPILE_FUZZ  
}
```

1.13.34 aclrtEventWaitStatus

```
aclrtEventWaitStatus{  
    ACL_EVENT_WAIT_STATUS_COMPLETE = 0,    //完成  
    ACL_EVENT_WAIT_STATUS_NOT_READY = 1,   //未完成  
    ACL_EVENT_WAIT_STATUS_RESERVED = 0xffff, //预留  
}
```

1.13.35 aclprofStepTag

```
typedef enum{  
    ACL_STEP_START = 0, //step start  
    ACL_STEP_END = 1  //step end  
} aclprofStepTag
```

说明

同一个[aclprofStepInfo](#)对象、同一个tag只能设置一次，否则Profiling解析会出错。

1.13.36 acltdtBuf

```
typedef void *acltdtBuf;
```

1.13.37 acltdtQueueAttrType

```
typedef enum {  
    ACL_TDT_QUEUE_NAME_PTR = 0, //队列名  
    ACL_TDT_QUEUE_DEPTH_UINT32 //队列深度  
} acltdtQueueAttrType;
```

1.13.38 acltdtQueueRouteParamType

```
typedef enum {  
    ACL_TDT_QUEUE_ROUTE_SRC_UINT32 = 0, //源队列ID  
    ACL_TDT_QUEUE_ROUTE_DST_UINT32,    //目标队列ID  
    ACL_TDT_QUEUE_ROUTE_STATUS_INT32   //路由绑定状态, 0表示未绑定, 1表示绑定, 2表示绑定异常  
} acltdtQueueRouteParamType;
```

1.13.39 acltdtQueueRouteQueryMode

```
typedef enum {  
    ACL_TDT_QUEUE_ROUTE_QUERY_SRC = 0,    //指定为只根据源队列ID匹配查询  
    ACL_TDT_QUEUE_ROUTE_QUERY_DST = 1,    //指定为只根据目标队列ID匹配查询  
    ACL_TDT_QUEUE_ROUTE_QUERY_SRC_AND_DST = 2 //指定为同时根据源、目标队列ID匹配查询  
} acltdtQueueRouteQueryMode;
```

1.13.40 acltdtQueueRouteQueryInfoParamType

```
typedef enum {  
    ACL_TDT_QUEUE_ROUTE_QUERY_MODE_ENUM = 0, //查询匹配模式, 选择该参数类型后, 参数值来源于  
    acltdtQueueRouteQueryMode枚举值  
    ACL_TDT_QUEUE_ROUTE_QUERY_SRC_ID_UINT32, //指定要查询的源队列ID  
    ACL_TDT_QUEUE_ROUTE_QUERY_DST_ID_UINT32 //指定要查询的目标队列ID  
} acltdtQueueRouteQueryInfoParamType;
```

1.13.41 acldvppChannelMode

```
enum acldvppChannelMode {  
    DVPP_CHNMODE_VPC = 1, //指定图片数据处理通道用于实现VPC功能  
    DVPP_CHNMODE_JPEGD = 2, //指定图片数据处理通道用于实现JPEGD功能  
    DVPP_CHNMODE_JPEGE = 4, //指定图片数据处理通道用于实现JPEGE功能  
    DVPP_CHNMODE_PNGD = 8 //指定图片数据处理通道用于实现PNGD功能  
};
```

1.13.42 aclopEngineType

```
typedef enum aclEngineType {  
    ACL_ENGINE_SYS, // 不关心具体执行引擎时填写, 当前仅支持设置为该选项  
    ACL_ENGINE_AICORE, // 将算子编译成AI Core算子, 当前不支持  
    ACL_ENGINE_VECTOR // 将算子编译成Vector Core算子, 当前不支持  
} aclopEngineType;
```

1.13.43 aclopCompileType

```
typedef enum aclCompileType {  
    ACL_COMPILE_SYS, // 当前仅支持设置为该选项  
    ACL_COMPILE_UNREGISTERED // 当前不支持  
} aclopCompileType;
```

1.13.44 aclrtEventRecordedStatus

```
typedef enum aclrtEventRecordedStatus {  
    ACL_EVENT_RECORDED_STATUS_NOT_READY = 0, //Event未被记录到Stream中, 或记录到Stream中的  
    Event未被执行或执行失败  
    ACL_EVENT_RECORDED_STATUS_COMPLETE = 1, //记录到Stream中的Event执行成功  
} aclrtEventRecordedStatus;
```

1.13.45 aclrtFloatOverflowMode

```
typedef enum aclrtFloatOverflowMode {  
    ACL_RT_OVERFLOW_MODE_SATURATION = 0, // 溢出检测饱和模式, 设置成该模式, 计算精度可能存在误  
    差, 该模式仅为兼容旧版本, 后续不演进
```

```
ACL_RT_OVERFLOW_MODE_INFNaN, // 溢出检测Inf/NaN模式，默认值
ACL_RT_OVERFLOW_MODE_UNDEF,
} aclrtFloatOverflowMode;
```

1.13.46 aclvppJpegdPrecisionMode

```
enum aclvppJpegdPrecisionMode {
    ACL_YUVOUT_ALIGN_DOWN = 0,
    ACL_YUVOUT_ALIGN_UP = 1,
    ACL_YUVOUT_ALIGN_DOWN_COMPAT = 2
};
```

对齐模式取值的含义如下：

- ACL_YUVOUT_ALIGN_DOWN:
 - 解码后的输出图片格式为YUV420SP时，若源图的宽、高为奇数时，将输出图片的宽、高向下2对齐
 - 解码后的输出图片格式为YUV422SP时，若源图的宽为奇数时，将输出图片的宽向下2对齐
 - 解码后的输出图片格式为YUV440SP时，若源图的高为奇数时，将输出图片高向下2对齐
- ACL_YUVOUT_ALIGN_UP:
 - 解码后的输出图片格式为YUV420SP时，若源图的宽、高为奇数时，将输出图片的宽、高向上2对齐
 - 解码后的输出图片格式为YUV422SP时，若源图的宽为奇数时，将输出图片的宽向上2对齐
 - 解码后的输出图片格式为YUV440SP时，若源图的高为奇数时，将输出图片高向上2对齐
- ACL_YUVOUT_ALIGN_DOWN_COMPAT：默认值
兼容旧版本，解码后的输出图片格式为YUV420SP/YUV422SP/YUV440SP时，若源图的宽、高为奇数时，将输出图片的宽、高向下2对齐。

1.13.47 aclmdlExecConfigAttr

```
typedef enum {
    ACL_MDL_STREAM_SYNC_TIMEOUT = 0,
    ACL_MDL_EVENT_SYNC_TIMEOUT,
    ACL_MDL_WORK_ADDR_PTR,
    ACL_MDL_WORK_SIZE,
    ACL_MDL_MPAIMID_SIZE, /** param reserved */
    ACL_MDL_AICQOS_SIZE, /** param reserved */
    ACL_MDL_AICOST_SIZE, /** param reserved */
    ACL_MDL_MEC_TIMETHR_SIZE /** param reserved */
} aclmdlExecConfigAttr;
```

表 1-109 枚举项说明

枚举项	说明
ACL_MDL_STREAM_SYNC_TIMEOUT	在执行模型推理时控制Stream任务的超时时间。该属性值为INT32类型。 取值说明如下： -1：表示永久等待，默认永久等待。 >0：配置具体的超时时间，单位是毫秒。
ACL_MDL_EVENT_SYNC_TIMEOUT	在执行模型推理时控制Event任务的超时时间。该属性值为INT32类型。 取值说明如下： -1：表示永久等待，默认永久等待。 >0：配置具体的超时时间，单位是毫秒。
ACL_MDL_WORK_ADDR_PTR	模型所需工作内存（ Device上存放模型执行过程中的临时数据 ）的指针，由用户管理工作内存。一般用于模型一次加载、多并发执行的场景。 如果同时配置ACL_MDL_WORK_ADDR_PTR以及 aclrtStreamConfigAttr 中的ACL_RT_STREAM_WORK_ADDR_PTR（表示Stream上模型的工作内存），则以ACL_MDL_WORK_ADDR_PTR优先。 当前版本不支持该配置。
ACL_MDL_WORK_SIZE	模型所需工作内存的大小，单位为Byte。一般用于模型一次加载、多并发执行的场景。 当前版本不支持该配置。
ACL_MDL_MPAIMID_SIZE	预留配置。
ACL_MDL_AICQOS_SIZE	预留配置。
ACL_MDL_AICOST_SIZE	预留配置。
ACL_MDL_MEC_TIME_THR_SIZE	预留配置。

1.13.48 aclrtStreamConfigAttr

```
typedef enum {  
    ACL_RT_STREAM_WORK_ADDR_PTR = 0,  
    ACL_RT_STREAM_WORK_SIZE,  
    ACL_RT_STREAM_FLAG,  
    ACL_RT_STREAM_PRIORITY,  
} aclrtStreamConfigAttr;
```

表 1-110 枚举项说明

枚举项	说明
ACL_RT_STREAM_WORK_ADDR_PTR	某一个Stream上的模型所需工作内存（ Device上存放模型执行过程中的临时数据 ）的指针，由用户管理工作内存。该配置主要用于多模型在同一个Stream上串行执行时想共享工作内存的场景，此时需按多个模型中最大的工作内存来申请内存，可提前使用 aclmdlQuerySize 查询各模型所需的工作内存大小。 如果同时配置ACL_RT_STREAM_WORK_ADDR_PTR以及 aclmdlExecConfigAttr 中的ACL_MDL_WORK_ADDR_PTR（表示某个模型的工作内存），则以aclmdlExecConfigAttr中的ACL_MDL_WORK_ADDR_PTR优先。
ACL_RT_STREAM_WORK_SIZE	模型所需工作内存的大小，单位为Byte。
ACL_RT_STREAM_FLAG	预留配置，默认值为0。
ACL_RT_STREAM_PRIORITY	Stream的优先级，数字越小优先级越高，取值[0,7]。默认值为0。

1.13.49 aclSysParamOpt

```
typedef enum {  
    ACL_OPT_DETERMINISTIC = 0,  
} aclSysParamOpt;
```

表 1-111 枚举项说明

枚举项	说明
ACL_OPT_DETERMINISTIC	是否开启确定性计算。 0：默认值，不开启确定性计算。 1：开启确定性计算。 当开启确定性计算功能时，算子在相同的硬件和输入下，多次执行将产生相同的输出。但启用确定性计算往往导致算子执行变慢。 默认情况下，不开启确定性计算，算子在相同的硬件和输入下，多次执行的结果可能不同。这个差异的来源，一般是因为在算子实现中，存在异步的多线程执行，会导致浮点数累加的顺序变化。 通常建议不开启确定性计算，因为确定性计算往往会导致算子执行变慢，进而影响性能。当发现模型多次执行结果不同，或者是进行精度调优时，可开启确定性计算，辅助模型调试、调优。

1.13.50 aclrtUtilizationInfo

```
typedef struct aclrtUtilizationInfo {  
    int32_t cubeUtilization; // Cube利用率  
    int32_t vectorUtilization; // Vector利用率  
    int32_t aicpuUtilization; // AI CPU利用率  
    int32_t memoryUtilization; // Device内存利用率  
    aclrtUtilizationExtendInfo *utilizationExtend; // 预留参数，当前设置为null  
} aclrtUtilizationInfo;
```

1.13.51 aclrtStreamStatus

```
typedef enum aclrtStreamStatus {  
    ACL_STREAM_STATUS_COMPLETE = 0, // Stream上的所有任务已完成  
    ACL_STREAM_STATUS_NOT_READY = 1, // Stream上至少有一个任务未完成  
    ACL_STREAM_STATUS_RESERVED = 0xFFFF, // 预留  
} aclrtStreamStatus;
```

1.13.52 aclrtMemHandleType

```
typedef enum aclrtMemHandleType {  
    ACL_MEM_HANDLE_TYPE_NONE = 0,  
} aclrtMemHandleType;
```

1.13.53 aclrtMemAllocationType

```
typedef enum aclrtMemAllocationType {  
    ACL_MEM_ALLOCATION_TYPE_PINNED = 0,  
} aclrtMemAllocationType;
```

1.13.54 aclrtMemAttr

```
typedef enum aclrtMemAttr {  
    ACL_DDR_MEM, // DDR内存，DDR上所有大页内存+普通内存  
    ACL_HBM_MEM, // HBM内存，HBM上所有大页内存+普通内存  
    ACL_DDR_MEM_HUGE, // DDR大页内存  
    ACL_DDR_MEM_NORMAL, // DDR普通内存  
    ACL_HBM_MEM_HUGE, // HBM大页内存  
    ACL_HBM_MEM_NORMAL, // HBM普通内存  
    ACL_DDR_MEM_P2P_HUGE, // DDR中用于Device间数据复制的大页内存  
    ACL_DDR_MEM_P2P_NORMAL, // DDR中用于Device间数据复制的普通内存  
    ACL_HBM_MEM_P2P_HUGE, // HBM中用于Device间数据复制的大页内存  
    ACL_HBM_MEM_P2P_NORMAL, // HBM中用于Device间数据复制的普通内存  
} aclrtMemAttr;
```

1.13.55 aclrtMemLocationType

```
typedef enum aclrtMemLocationType {  
    ACL_MEM_LOCATION_TYPE_HOST = 0, // 预留，当前不支持  
    ACL_MEM_LOCATION_TYPE_DEVICE,  
} aclrtMemLocationType;
```

1.13.56 aclrtMemLocation

```
typedef struct aclrtMemLocation {  
    uint32_t id;  
    aclrtMemLocationType type; // 内存所在位置，当前仅支持ACL_MEM_LOCATION_TYPE_DEVICE（表示Device上的内存）  
} aclrtMemLocation;
```

1.13.57 aclrtPhysicalMemProp

```
typedef struct aclrtPhysicalMemProp {  
    aclrtMemHandleType handleType;
```

```
aclrtMemAllocationType allocationType;  
aclrtMemAttr memAttr;  
aclrtMemLocation location;  
uint64_t reserve;  
} aclrtPhysicalMemProp;
```

成员名称	描述
handleType	handle类型，当前仅支持ACL_MEM_HANDLE_TYPE_NONE 。
allocationType	内存分配类型，当前仅支持ACL_MEM_ALLOCATION_TYPE_PINNED。
memAttr	内存属性。
location	内存所在位置。
reserve	预留。

1.13.58 aclrtDrvMemHandle

```
typedef void* aclrtDrvMemHandle;
```

1.13.59 aclrtDeviceStatus

```
typedef enum aclrtDeviceStatus {  
    ACL_RT_DEVICE_STATUS_NORMAL = 0,    // Device正常可用  
    ACL_RT_DEVICE_STATUS_ABNORMAL,      // Device异常  
    ACL_RT_DEVICE_STATUS_END = 0xFFFF,  // 预留值  
} aclrtDeviceStatus;
```

1.13.60 aclrtMemGranularityOptions

```
typedef enum aclrtMemGranularityOptions {  
    ACL_RT_MEM_ALLOC_GRANULARITY_MINIMUM,  
    ACL_RT_MEM_ALLOC_GRANULARITY_RECOMMENDED,  
    ACL_RT_MEM_ALLOC_GRANULARITY_UNDEF = 0xFFFF,  
} aclrtMemGranularityOptions;
```

1.13.61 aclCannAttr

```
typedef enum {  
    ACL_CANN_ATTR_UNDEFINED = -1, // 未知特性  
    ACL_CANN_ATTR_INF_NAN = 0,    // 溢出检测Inf/NaN模式  
    ACL_CANN_ATTR_BF16 = 1,      // bf16数据类型  
    ACL_CANN_ATTR_JIT_COMPILE = 2 // 算子二进制特性  
} aclCannAttr;
```

1.13.62 aclDeviceInfo

```
typedef enum {  
    ACL_DEVICE_INFO_UNDEFINED = -1, // 未知规格  
    ACL_DEVICE_INFO_AI_CORE_NUM = 0, // AI Core数量  
    ACL_DEVICE_INFO_VECTOR_CORE_NUM = 1, // Vector Core数量  
    ACL_DEVICE_INFO_L2_SIZE = 2 // L2 Buffer大小，单位Byte  
} aclDeviceInfo;
```

1.13.63 aclmdlExeOMDesc

当前版本不支持该结构体。

```
typedef struct aclmdlExeOMDesc {
    size_t workSize;
    size_t weightSize;
    size_t modelDescSize;
    size_t kernelSize;
    size_t kernelArgsSize;
    size_t staticTaskSize;
    size_t dynamicTaskSize;
    size_t reserved[9];
} aclmdlExeOMDesc;
```

成员名称	描述
workSize	模型执行时所需的工作内存的大小，单位Byte。
weightSize	模型执行时所需的权值内存的大小，单位Byte。
modelDescSize	存放模型描述信息所需的内存大小，单位Byte。
kernelSize	存放TBE算子kernel（算子的*.o与*.json）所需的内存大小，单位Byte。
kernelArgsSize	存放TBE算子kernel参数所需的内存大小，单位Byte。
staticTaskSize	存放静态shape任务描述信息所需的内存大小，单位Byte。
dynamicTaskSize	存放动态shape任务描述信息所需的内存大小，单位Byte。
reserved	预留值。

1.13.64 aclrtMemUceInfo

```
#define MAX_MEM_UCE_INFO_ARRAY_SIZE 128
#define UCE_INFO_RESERVED_SIZE 14

typedef struct aclrtMemUceInfo {
    void* addr;
    size_t len;
    size_t reserved[UCE_INFO_RESERVED_SIZE];
} aclrtMemUceInfo;
```

成员名称	描述
addr	内存UCE的错误虚拟起始地址。
len	内存大小，单位Byte。 从addr开始的len大小范围内的内存都是异常的。
reserved	预留参数。

1.13.65 aclmdlIODimsRange

```
#define ACL_DIM_ENDPOINTS 2
#define ACL_MAX_DIM_CNT 128

typedef struct aclmdlIODimsRange {
    size_t rangeCount; /** 维度数量 */
}
```

```
int64_t range[ACL_MAX_DIM_CNT][ACL_DIM_ENDPOINTS]; /** Shape范围二维数组，第一维表示维度数量，第二维表示范围的上、下限 */  
} aclmdlIODimsRange;
```

1.13.66 aclrtCmoType

```
typedef enum aclrtCmoType {  
    ACL_RT_CMO_TYPE_PREFETCH = 0,    // 内存预取  
} aclrtCmoType;
```

1.13.67 aclrtLastErrLevel

```
typedef enum aclrtLastErrLevel {  
    ACL_RT_THREAD_LEVEL = 0,  
} aclrtLastErrLevel;
```

1.13.68 aclmdlAIPP

1.13.68.1 aclmdlCreateAIPP

函数功能

动态AIPP场景下，根据模型支持的batch size创建aclmdlAIPP类型的数据，用于存放动态AIPP的参数。

如需销毁aclmdlAIPP类型的数据，请参见[aclmdlDestroyAIPP](#)。

函数原型

[aclmdlAIPP](#) *aclmdlCreateAIPP(uint64_t batchSize)

参数说明

参数名	输入/输出	说明
batchSize	输入	原始模型的batch size。

返回值说明

返回aclmdlAIPP类型的指针。

1.13.68.2 aclmdlSetAIPPCscParams

函数功能

动态AIPP场景下，设置CSC色域转换相关的参数，若色域转换开关关闭，则调用该接口设置以下参数无效。

YUV转BGR:
| B | | cscMatrixR0C0 cscMatrixR0C1 cscMatrixR0C2 | | Y - cscInputBiasR0 |
| G | = | cscMatrixR1C0 cscMatrixR1C1 cscMatrixR1C2 | | U - cscInputBiasR1 | >> 8
| R | | cscMatrixR2C0 cscMatrixR2C1 cscMatrixR2C2 | | V - cscInputBiasR2 |
BGR转YUV:

```
| Y | | cscMatrixR0C0 cscMatrixR0C1 cscMatrixR0C2 | | B | | cscOutputBiasR0 |  
| U | = | cscMatrixR1C0 cscMatrixR1C1 cscMatrixR1C2 | | G | >> 8 + | cscOutputBiasR1 |  
| V | | cscMatrixR2C0 cscMatrixR2C1 cscMatrixR2C2 | | R | | cscOutputBiasR2 |
```

色域转换参数值与转换前图片的格式、转换后图片的格式强相关，您可以参考《[ATC 工具使用指南](#)》中的“[特性配置 > AIPP配置 > 色域转换配置说明](#)”，根据转换前图片格式、转换后图片格式来配置色域转换参数。如果手册中列举的图片格式不满足要求，您需自行根据实际需求配置色域转换参数。

约束说明

如果通过[aclmdlSetAIPPInputFormat](#)接口设置的原始图像格式为YUV400，则不支持通过本接口设置色域转换参数。

函数原型

```
aclError aclmdlSetAIPPCscParams(aclmdlAIPP *aippParamsSet, int8_t  
cscSwitch,  
  
int16_t cscMatrixR0C0, int16_t cscMatrixR0C1, int16_t cscMatrixR0C2,  
int16_t cscMatrixR1C0, int16_t cscMatrixR1C1, int16_t cscMatrixR1C2,  
int16_t cscMatrixR2C0, int16_t cscMatrixR2C1, int16_t cscMatrixR2C2,  
uint8_t cscOutputBiasR0, uint8_t cscOutputBiasR1, uint8_t cscOutputBiasR2,  
uint8_t cscInputBiasR0, uint8_t cscInputBiasR1, uint8_t cscInputBiasR2)
```

参数说明

参数名	输入/输出	说明
aippParamsSet	输出	动态AIPP参数对象的指针。 需提前调用 aclmdlCreateAIPP 接口创建aclmdlAIPP类型的数据。
cscSwitch	输入	色域转换开关，取值范围： 0：关闭色域转换开关，设置为0时，则设置以下参数无效 1：打开色域转换开关
cscMatrixR0C0	输入	色域转换矩阵参数。 取值范围：[-32677,32676]
cscMatrixR0C1	输入	色域转换矩阵参数。 取值范围：[-32677,32676]
cscMatrixR0C2	输入	色域转换矩阵参数。 取值范围：[-32677,32676]

参数名	输入/输出	说明
cscMatrixR1C0	输入	色域转换矩阵参数。 取值范围：[-32677,32676]
cscMatrixR1C1	输入	色域转换矩阵参数。 取值范围：[-32677,32676]
cscMatrixR1C2	输入	色域转换矩阵参数。 取值范围：[-32677,32676]
cscMatrixR2C0	输入	色域转换矩阵参数。 取值范围：[-32677,32676]
cscMatrixR2C1	输入	色域转换矩阵参数。 取值范围：[-32677,32676]
cscMatrixR2C2	输入	色域转换矩阵参数。 取值范围：[-32677,32676]
cscOutputBiasR0	输入	RGB转YUV时的输出偏移。 取值范围：[0, 255]
cscOutputBiasR1	输入	RGB转YUV时的输出偏移。 取值范围：[0, 255]
cscOutputBiasR2	输入	RGB转YUV时的输出偏移。 取值范围：[0, 255]
cscInputBiasR0	输入	YUV转RGB时的输入偏移。 取值范围：[0, 255]
cscInputBiasR1	输入	YUV转RGB时的输入偏移。 取值范围：[0, 255]
cscInputBiasR2	输入	YUV转RGB时的输入偏移。 取值范围：[0, 255]

返回值说明

返回0表示成功，返回其他值表示失败。

1.13.68.3 aclmdlSetAIPPInputFormat

函数功能

动态AIPP场景下，必须设置原始输入图像的格式。

函数原型

aclError aclmdlSetAIPPInputFormat(**aclmdlAIPP** *aippParamsSet,
aclAippInputFormat inputFormat)

参数说明

参数名	输入/输出	说明
aippParamsSet	输出	动态AIPP参数对象的指针。 需提前调用 aclmdlCreateAIPP 接口创建aclmdlAIPP类型的数据。
inputFormat	输入	表示原始输入图像的格式。

返回值说明

返回0表示成功，返回其他值表示失败。

1.13.68.4 aclmdlSetAIPPRbuvSwapSwitch

函数功能

动态AIPP场景下，设置是否交换R通道与B通道、或者是否交换U通道与V通道。

函数原型

aclError aclmdlSetAIPPRbuvSwapSwitch(**aclmdlAIPP** *aippParamsSet, int8_t
rbuvSwapSwitch)

参数说明

参数名	输入/输出	说明
aippParamsSet	输出	动态AIPP参数对象的指针。 需提前调用 aclmdlCreateAIPP 接口创建aclmdlAIPP类型的数据。

参数名	输入/输出	说明
rbuvSwapSwitch	输入	表示是否交换R通道与B通道、或者是否交换U通道与V通道的开关，取值范围： 0：不交换 1：交换

返回值说明

返回0表示成功，返回其他值表示失败。

1.13.68.5 aclmdlSetAIPPAxSwapSwitch

函数功能

动态AIPP场景下，设置RGBA->ARGB或者YUVA->AYUV的交换开关。

函数原型

aclError aclmdlSetAIPPAxSwapSwitch(**aclmdlAIPP** *aippParamsSet, int8_t axSwapSwitch)

参数说明

参数名	输入/输出	说明
aippParamsSet	输出	动态AIPP参数对象的指针。 需提前调用 aclmdlCreateAIPP 接口创建aclmdlAIPP类型的数据。
axSwapSwitch	输入	表示RGBA->ARGB或者YUVA->AYUV的交换开关，取值范围： 0：不交换 1：交换

返回值说明

返回0表示成功，返回其他值表示失败。

1.13.68.6 aclmdlSetAIPPSrcImageSize

函数功能

动态AIPP场景下，必须设置原始图片的宽和高。

函数原型

```
aclError aclmdlSetAIPPSrcImageSize(aclmdlAIPP *aippParamsSet, int32_t
srcImageSizeW, int32_t srcImageSizeH)
```

参数说明

参数名	输入/输出	说明
aippParamsSet	输出	动态AIPP参数对象的指针。 需提前调用 aclmdlCreateAIPP 接口创建aclmdlAIPP类型的数据。
srcImageSizeW	输入	原始图片的宽，对于YUV420SP_U8或YUV422SP_U8或YUYV_U8类型的图像，要求srcImageSizeW取值是偶数。 取值范围： [2,4096]
srcImageSizeH	输入	原始图片的高，对于YUV420SP_U8类型的图像，要求srcImageSizeH取值是偶数。 取值范围： [1,4096]

返回值说明

返回0表示成功，返回其他值表示失败。

1.13.68.7 aclmdlSetAIPPScfParams

- Atlas 200/300/500 推理产品，不支持该接口。
- Atlas 200/500 A2推理产品，不支持该接口。
- Atlas 训练系列产品，不支持该接口。
- Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。
- Atlas 推理系列产品，不支持该接口。

函数功能

动态AIPP场景下，设置缩放相关的参数。

函数原型

```
aclError aclmdlSetAIPPScfParams(aclmdlAIPP *aippParamsSet, int8_t scfSwitch,
int32_t scfInputSizeW, int32_t scfInputSizeH,
int32_t scfOutputSizeW, int32_t scfOutputSizeH,
uint64_t batchSize)
```

约束说明

缩放比例 $\text{scfOutputSizeW}/\text{scfInputSizeW} \in [1/16, 16]$ 、 $\text{scfOutputSizeH}/\text{scfInputSizeH} \in [1/16, 16]$ 。

参数说明

参数名	输入/输出	说明
aippParmsSet	输出	动态AIPP参数对象的指针。 需提前调用 aclmdlCreateAIPP 接口创建aclmdlAIPP类型的数据。
scfSwitch	输入	是否对图片执行缩放操作，取值范围： 0：不执行缩放操作，设置为0时，则设置scfInputSizeW、scfInputSizeH、scfOutputSizeW、scfOutputSizeH参数无效 1：执行缩放操作
scfInputSizeW	输入	缩放前图片的宽。 取值范围：[16,4096] 若开启了抠图功能，则缩放前图片的宽与 抠图区域的宽度 保持一致；若未开启抠图功能，则缩放前图片的宽与 原始图片的宽 保持一致。
scfInputSizeH	输入	缩放前图片的高。 取值范围：[16,4096] 若开启了抠图功能，则缩放前图片的高与 抠图区域的高度 保持一致；若未开启抠图功能，则缩放前图片的高与 原始图片的高 保持一致。
scfOutputSizeW	输入	缩放后图片的宽。 取值范围：[16,1920]
scfOutputSizeH	输入	缩放后图片的高。 取值范围：[16,4096]
batchIndex	输入	指定对第几个Batch上的图片执行缩放操作。 取值范围：[0, batchSize) batchSize是在调用 aclmdlCreateAIPP 接口创建aclmdlAIPP类型的数据时设置。

返回值说明

返回0表示成功，返回其他值表示失败。

1.13.68.8 aclmdlSetAIPPCropParams

函数功能

动态AIPP场景下，设置抠图相关的参数。

函数原型

```
aclError aclmdlSetAIPPCropParams(aclmdlAIPP *aippParamsSet, int8_t
cropSwitch,

int32_t cropStartPosW, int32_t cropStartPosH,

int32_t cropSizeW, int32_t cropSizeH,

uint64_t batchSize)
```

约束说明

若开启抠图功能，则通过[aclmdlSetAIPPSrcImageSize](#)接口设置的参数与通过[aclmdlSetAIPPCropParams](#)接口设置的参数之间必须满足以下公式：

- cropSizeW+cropStartPosW ≤ srcImageSizeW
- cropSizeH+cropStartPosH ≤ srcImageSizeH

参数说明

参数名	输入/输出	说明
aippParamsSet	输出	动态AIPP参数对象的指针。 需提前调用 aclmdlCreateAIPP 接口创建aclmdlAIPP类型的数据。
cropSwitch	输入	是否对图片执行抠图操作，取值范围： 0：不执行抠图操作，设置为0时，则设置cropStartPosW、cropStartPosH、cropSizeW、cropSizeH参数无效 1：执行抠图操作
cropStartPosW	输入	抠图时，坐标点起始位置在图中横向的坐标。 对于YUV420SP_U8格式的图像，参数取值要求是偶数。 取值范围：[0,4095]
cropStartPosH	输入	抠图时，坐标点起始位置在图中纵向的坐标。 对于YUV420SP_U8格式的图像，参数取值要求是偶数。 取值范围：[0,4095]
cropSizeW	输入	抠图区域的宽度。 取值范围：[1,4096]

参数名	输入/输出	说明
cropSizeH	输入	抠图区域的高度。 取值范围：[1,4096]
batchIndex	输入	指定对第几个Batch上的图片执行抠图操作。 取值范围：[0,batchSize) batchSize是在调用 aclmdlCreateAIPP 接口创建aclmdlAIPP类型的数据时设置。

返回值说明

返回0表示成功，返回其他值表示失败。

1.13.68.9 aclmdlSetAIPPPaddingParams

函数功能

动态AIPP场景下，设置补边相关的参数。

函数原型

```
aclError aclmdlSetAIPPPaddingParams(aclmdlAIPP *aippParamsSet, int8_t paddingSwitch,
int32_t paddingSizeTop, int32_t paddingSizeBottom,
int32_t paddingSizeLeft, int32_t paddingSizeRight,
uint64_t batchIndex)
```

约束说明

补边之后，图片的宽必须小于等于1080。

参数说明

参数名	输入/输出	说明
aippParamsSet	输出	动态AIPP参数对象的指针。 需提前调用 aclmdlCreateAIPP 接口创建aclmdlAIPP类型的数据。

参数名	输入/输出	说明
paddingSwitch	输入	是否对图片执行补边操作，取值范围： 0：不执行补边操作，设置为0时，则设置paddingSizeTop、paddingSizeBottom、paddingSizeLeft、paddingSizeRight参数无效 1：执行补边操作
paddingSizeTop	输入	在图片上方填充的值。 取值范围：[0, 32]
paddingSizeBottom	输入	在图片下方填充的值。 取值范围：[0, 32]
paddingSizeLeft	输入	在图片左方填充的值。 取值范围：[0, 32]
paddingSizeRight	输入	在图片右方填充的值。 取值范围：[0, 32]
batchIndex	输入	指定对第几个Batch上的图片执行补边操作。 取值范围：[0, batchSize) batchSize是在调用 aclmdlCreateAIPP 接口创建aclmdlAIPP类型的数据时设置。

返回值说明

返回0表示成功，返回其他值表示失败。

1.13.68.10 aclmdlSetAIPPDtcPixelMean

函数功能

动态AIPP场景下，设置通道的均值。

函数原型

```
aclError aclmdlSetAIPPDtcPixelMean(aclmdlAIPP *aippParamsSet,
int16_t dtcPixelMeanChn0,
int16_t dtcPixelMeanChn1,
int16_t dtcPixelMeanChn2,
int16_t dtcPixelMeanChn3,
uint64_t batchIndex)
```

参数说明

参数名	输入/输出	说明
aippParmsSet	输出	动态AIPP参数对象的指针。 需提前调用 aclmdlCreateAIPP 接口创建aclmdlAIPP类型的数据。
dtcPixelMeanChn0	输入	通道0的均值。 取值范围：[0, 255]
dtcPixelMeanChn1	输入	通道1的均值。 取值范围：[0, 255]
dtcPixelMeanChn2	输入	通道2的均值。 取值范围：[0, 255]
dtcPixelMeanChn3	输入	通道3的均值。 如果只有3个通道，将该参数设置为0。 取值范围：[0, 255]
batchIndex	输入	指定对第几个Batch上的图片设置通道均值。 取值范围：[0, batchSize) batchSize是在调用 aclmdlCreateAIPP 接口创建aclmdlAIPP类型的数据时设置。

返回值说明

返回0表示成功，返回其他值表示失败。

1.13.68.11 aclmdlSetAIPPDtcPixelMin

函数功能

动态AIPP场景下，设置通道的最小值。

函数原型

```
aclError aclmdlSetAIPPDtcPixelMin(aclmdlAIPP *aippParmsSet,  
float dtcPixelMinChn0,  
float dtcPixelMinChn1,  
float dtcPixelMinChn2,  
float dtcPixelMinChn3,  
uint64_t batchIndex)
```

参数说明

参数名	输入/输出	说明
aippParmsSet	输出	动态AIPP参数对象的指针。 需提前调用 aclmdlCreateAIPP 接口创建aclmdlAIPP类型的数据。
dtcPixelMinChn0	输入	通道0的最小值。 取值范围：[0, 255]
dtcPixelMinChn1	输入	通道1的最小值。 取值范围：[0, 255]
dtcPixelMinChn2	输入	通道2的最小值。 取值范围：[0, 255]
dtcPixelMinChn3	输入	通道3的最小值。如果只有3个通道，将该参数设置为0。 取值范围：[0, 255]
batchIndex	输入	指定对第几个Batch上的图片设置通道最小值。 取值范围：[0, batchSize) batchSize是在调用 aclmdlCreateAIPP 接口创建aclmdlAIPP类型的数据时设置。

返回值说明

返回0表示成功，返回其他值表示失败。

1.13.68.12 aclmdlSetAIPPPixelVarReci

函数功能

动态AIPP场景下，设置通道的方差。

函数原型

```
aclError aclmdlSetAIPPPixelVarReci(aclmdlAIPP *aippParmsSet,  
float dtcPixelVarReciChn0,  
float dtcPixelVarReciChn1,  
float dtcPixelVarReciChn2,  
float dtcPixelVarReciChn3,  
uint64_t batchIndex)
```

参数说明

参数名	输入/输出	说明
aippParmsSet	输出	动态AIPP参数对象的指针。 需提前调用 aclmdlCreateAIPP 接口创建aclmdlAIPP类型的数据。
dtcPixelVarReciChn0	输入	通道0的方差的倒数，默认值为1.0。 取值范围：[-65504, 65504]
dtcPixelVarReciChn1	输入	通道1的方差的倒数，默认值为1.0。 取值范围：[-65504, 65504]
dtcPixelVarReciChn2	输入	通道2的方差的倒数，默认值为1.0。 取值范围：[-65504, 65504]
dtcPixelVarReciChn3	输入	通道3的方差的倒数，默认值为1.0。如果只有3个通道，则将该参数设置为1.0。 取值范围：[-65504, 65504]
batchIndex	输入	指定对第几个Batch上的图片设置通道的方差。 取值范围：[0,batchSize) batchSize是在调用 aclmdlCreateAIPP 接口创建aclmdlAIPP类型的数据时设置。

返回值说明

返回0表示成功，返回其他值表示失败。

1.13.68.13 aclmdlGetAippDataSize

函数功能

根据batchSize获取动态AIPP输入数据所需的内存大小。

函数原型

aclError [aclmdlGetAippDataSize](#)(uint64_t batchSize, size_t *size)

参数说明

参数名	输入/输出	说明
batchSize	输入	原始模型的batch size。
size	输出	内存大小，单位Byte。

返回值说明

返回0表示成功，返回非0表示失败。

1.13.68.14 aclmdlDestroyAIPP

函数功能

销毁通过[aclmdlCreateAIPP](#)接口创建的aclmdlAIPP类型的数据。

函数原型

```
aclError aclmdlDestroyAIPP(const aclmdlAIPP *aippParamsSet)
```

参数说明

参数名	输入/输出	说明
aippParamsSet	输入	待销毁的aclmdlAIPP类型的指针。

返回值说明

返回0表示成功，返回非0表示失败。

1.13.69 aclopHandle

1.13.69.1 aclopCreateHandle

函数功能

创建一个执行算子的handle。
如需销毁handle，请参见[aclopDestroyHandle](#)。

约束说明

对于算子有constant输入的场景，如果未调用[aclSetTensorConst](#)接口设置constant输入，则需调用[aclSetTensorPlaceMent](#)设置TensorDesc的placement属性，将memType设置为Host内存。

函数原型

```
aclError aclopCreateHandle(const char *opType,  
int numInputs,  
const aclTensorDesc *const inputDesc[],  
int numOutputs,
```

```
const aclTensorDesc *const outputDesc[],  
const aclopAttr *opAttr,  
aclopHandle **handle);
```

参数说明

参数名	输入/输出	说明
opType	输入	算子类型名称的指针。
numInputs	输入	算子输入tensor的数量。
inputDesc	输入	算子输入tensor描述的指针数组。
numOutputs	输入	算子输出tensor的数量。
outputDesc	输入	算子输出tensor描述的指针数组。
opAttr	输入	算子属性的指针。
handle	输出	“aclopHandle数据指针”的指针。

返回值说明

返回0表示成功，返回其他值表示失败。

1.13.69.2 aclopDestroyHandle

函数功能

销毁通过aclopCreateHandle接口创建的执行算子的handle。

函数原型

```
void aclopDestroyHandle(aclopHandle *handle)
```

参数说明

参数名	输入/输出	说明
handle	输入	待销毁的aclopHandle类型的指针。

返回值说明

无

1.13.70 aclDataBuffer

1.13.70.1 aclCreateDataBuffer

函数功能

创建[aclDataBuffer](#)类型的数据，该数据类型用于描述内存地址、大小等内存信息。

如需销毁[aclDataBuffer](#)类型的数据，请参见[aclDestroyDataBuffer](#)。

函数原型

[aclDataBuffer](#) *[aclCreateDataBuffer](#)(void *data, size_t size)

参数说明

参数名	输入/输出	说明
data	输入	存放数据内存地址的指针。data参数支持传入nullptr，表示创建一个空的数据类型，此时size参数值必须设置为0。 该内存需由用户自行管理，调用 aclrtMalloc 接口/ aclrtFree 接口申请/释放内存，或调用 aclrtMallocHost 接口/ aclrtFreeHost 接口申请/释放内存。
size	输入	内存大小，单位Byte。 如果用户需要使用空tensor，则在申请内存时，内存大小最小为1Byte，以保障后续业务正常运行。

返回值说明

返回[aclDataBuffer](#)类型的指针。

1.13.70.2 aclDestroyDataBuffer

函数功能

销毁通过[aclCreateDataBuffer](#)接口创建的[aclDataBuffer](#)类型的数据。

此处仅销毁[aclDataBuffer](#)类型的数据，调用[aclCreateDataBuffer](#)接口创建[aclDataBuffer](#)类型数据时传入的data的内存需由用户自行释放。

函数原型

[aclError](#) [aclDestroyDataBuffer](#)(const [aclDataBuffer](#) *dataBuffer)

参数说明

参数名	输入/输出	说明
dataBuffer	输入	待销毁的aclDataBuffer类型的指针。

返回值说明

返回0表示成功，返回非0表示失败。

1.13.70.3 aclGetDataBufferAddr

函数功能

获取aclDataBuffer类型中的数据的内存地址。

函数原型

void *aclGetDataBufferAddr(const [aclDataBuffer](#) *dataBuffer)

参数说明

参数名	输入/输出	说明
dataBuffer	输入	aclDataBuffer类型的指针。 需提前调用 aclCreateDataBuffer 接口创建aclDataBuffer类型的数据。

返回值说明

返回aclDataBuffer类型中的数据的内存地址。

1.13.70.4 aclGetDataBufferSize (废弃)

须知

此接口后续版本会废弃，请使用[aclGetDataBufferSizeV2](#)接口。

函数功能

获取aclDataBuffer类型中数据的内存大小，单位Byte。

函数原型

uint32 aclGetDataBufferSize(const [aclDataBuffer](#) *dataBuffer)

参数说明

参数名	输入/输出	说明
dataBuffer	输入	aclDataBuffer类型的指针。 需提前调用 aclCreateDataBuffer 接口创建aclDataBuffer类型的数据。

返回值说明

aclDataBuffer类型中数据的内存大小。

1.13.70.5 aclGetDataBufferSizeV2

函数功能

获取aclDataBuffer类型中数据的内存大小，单位Byte。

函数原型

size_t aclGetDataBufferSizeV2(const [aclDataBuffer](#) *dataBuffer)

参数说明

参数名	输入/输出	说明
dataBuffer	输入	aclDataBuffer类型的指针。 需提前调用 aclCreateDataBuffer 接口创建aclDataBuffer类型的数据。

返回值说明

aclDataBuffer类型中数据的内存大小。

1.13.70.6 aclUpdateDataBuffer

函数功能

更新aclDataBuffer中数据的内存及大小。

更新aclDataBuffer后，之前aclDataBuffer中存放数据的内存如果不使用，需及时释放，否则可能会导致内存泄漏。

函数原型

[aclError](#) aclUpdateDataBuffer([aclDataBuffer](#) *dataBuffer, void *data, size_t size)

参数说明

参数名	输入/输出	说明
dataBuffer	输入	aclDataBuffer类型的指针。 需提前调用 aclCreateDataBuffer 接口创建aclDataBuffer类型的数据。 该内存需由用户自行管理，调用 aclrtMalloc 接口/ aclrtFree 接口申请/释放内存，或调用 aclrtMallocHost 接口/ aclrtFreeHost 接口申请/释放内存。
data	输入	存放数据内存地址的指针。
size	输入	内存大小，单位Byte。 如果用户需要使用空tensor，则在申请内存时，内存大小最小为1Byte，以保障后续业务正常运行。

返回值说明

返回0表示成功，返回非0表示失败。

1.13.71 aclmdlDataset

1.13.71.1 aclmdlCreateDataset

函数功能

创建aclmdlDataset类型的数据，该数据类型用于描述模型推理时的输入数据、输出数据，模型可能存在多个输入、多个输出，每个输入/输出的内存地址、内存大小用[aclDataBuffer](#)类型的数据来描述。

如需销毁aclmdlDataset类型的数据，请参见[aclmdlDestroyDataset](#)。

函数原型

[aclmdlDataset](#) *aclmdlCreateDataset()

参数说明

无

返回值说明

返回aclmdlDataset类型的指针。

1.13.71.2 aclmdlDestroyDataset

函数功能

销毁通过[aclmdlCreateDataset](#)接口创建的[aclmdlDataset](#)类型的数据。

函数原型

[aclError](#) [aclmdlDestroyDataset](#)(const [aclmdlDataset](#) *dataset)

参数说明

参数名	输入/输出	说明
dataset	输入	待销毁的 aclmdlDataset 类型的指针。

返回值说明

返回0表示成功，返回非0表示失败。

1.13.71.3 aclmdlAddDatasetBuffer

函数功能

向[aclmdlDataset](#)中增加[aclDataBuffer](#)。

函数原型

[aclError](#) [aclmdlAddDatasetBuffer](#)([aclmdlDataset](#) *dataset, [aclDataBuffer](#) *dataBuffer)

参数说明

参数名	输入/输出	说明
dataset	输出	待增加 aclDataBuffer 的 aclmdlDataset 地址指针。 需提前调用 aclmdlCreateDataset 接口创建 aclmdlDataset 类型的数据。
dataBuffer	输入	待增加的 aclDataBuffer 地址指针。 需提前调用 aclCreateDataBuffer 接口创建 aclDataBuffer 类型的数据。

返回值说明

返回0表示成功，返回非0表示失败。

1.13.71.4 aclmdlGetDatasetNumBuffers

函数功能

获取aclmdlDataset中aclDataBuffer的个数。

函数原型

`size_t aclmdlGetDatasetNumBuffers(const aclmdlDataset *dataset)`

参数说明

参数名	输入/输出	说明
dataset	输入	aclmdlDataset类型的指针。 需提前调用 aclmdlCreateDataset 接口创建aclmdlDataset类型的数据。

返回值说明

aclDataBuffer的个数。

1.13.71.5 aclmdlGetDatasetBuffer

函数功能

获取aclmdlDataset中的第n个aclDataBuffer。

函数原型

`aclDataBuffer* aclmdlGetDatasetBuffer(const aclmdlDataset *dataset, size_t index)`

参数说明

参数名	输入/输出	说明
dataset	输入	aclmdlDataset类型的指针。 需提前调用 aclmdlCreateDataset 接口创建aclmdlDataset类型的数据。
index	输入	表明获取的是第几个aclDataBuffer。

返回值说明

- 获取成功，返回aclDataBuffer的地址。

- 获取失败返回空地址。

1.13.71.6 aclmdlSetDatasetTensorDesc

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 200/500 A2推理产品，不支持该接口。

函数功能

如果模型输入或输出的Shape是动态的，在模型执行之前调用本接口设置模型输入或输出的tensor描述信息。

约束说明

对同一个模型，[aclmdlSetDynamicBatchSize](#)接口、[aclmdlSetDynamicHWSIZE](#)接口、[aclmdlSetInputDynamicDims](#)接口、[aclmdlSetDatasetTensorDesc](#)接口，只能调用其中一个接口。

函数原型

aclError [aclmdlSetDatasetTensorDesc](#)([aclmdlDataset](#) *dataset, [aclTensorDesc](#) *tensorDesc, size_t index)

参数说明

参数名	输入/输出	说明
dataset	输出	待增加 aclTensorDesc 的 aclmdlDataset 地址指针，表示模型执行的输入或输出数据结构。 需提前调用 aclmdlCreateDataset 接口创建 aclmdlDataset 类型的数据，再调用 aclmdlAddDatasetBuffer 接口向 aclmdlDataset 中增加 aclDataBuffer 。
tensorDesc	输入	待增加的 aclTensorDesc 地址指针，表示模型执行时对应的输入或输出的tensor描述。 需提前调用 aclCreateTensorDesc 接口创建 aclTensorDesc 类型的数据，当前只有设置模型输入、输出tensor描述信息中的维度信息有效（对应 aclCreateTensorDesc 接口中的代表维度个数的numDims参数、代表维度大小的dims参数），设置数据类型、Format无效。 此处设置的维度个数、维度大小必须在模型构建时设置的输入Shape范围内。
index	输入	表示第几个输入或输出的序号。 模型存在多个输入、输出时，为避免序号出错，可以先调用 aclmdlGetInputNameByIndex 、 aclmdlGetOutputNameByIndex 接口获取输入、输出的名称，根据输入、输出名称所对应的index来设置。

返回值说明

返回0表示成功，返回非0表示失败。

参考资源

接口调用示例，参见动态Shape输入（设置Shape范围）。

1.13.71.7 aclmdlGetDatasetTensorDesc

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 200/500 A2推理产品不支持该接口。

函数功能

如果模型输入或输出的Shape是动态的，模型在模型执行之后可以调用本接口从aclmdlDataset类型的数据（该类型用于描述模型推理时的输入数据、输出数据）中获取指定的输入或者输出的Tensor描述信息。

典型场景举例：如果模型输入Shape是动态的，在模型执行之前调用[aclmdlSetDatasetTensorDesc](#)设置该输入的tensor描述信息，在模型执行之后，调用[aclmdlGetDatasetTensorDesc](#)接口获取模型动态输出的Tensor描述信息。

函数原型

aclTensorDesc *aclmdlGetDatasetTensorDesc(const **aclmdlDataset** *dataset, size_t index)

参数说明

参数名	输入/输出	说明
dataset	输入	模型执行的输入或输出数据指针。
index	输入	表示第几个输入或输出的序号。 模型存在多个输入、输出时，为避免序号出错，可以先调用 aclmdlGetInputNameByIndex 、 aclmdlGetOutputNameByIndex 接口获取输入、输出的名称，根据输入、输出名称所对应的index来设置。

返回值说明

返回指定输入或输出的tensor描述信息。

参考资源

接口调用示例，参见动态Shape输入（设置Shape范围）。

1.13.72 aclmdlDesc

1.13.72.1 aclmdlCreateDesc

函数功能

创建[aclmdlDesc](#)类型的数据，表示模型描述信息。
如需销毁[aclmdlDesc](#)类型的数据，请参见[aclmdlDestroyDesc](#)。

函数原型

[aclmdlDesc](#)* [aclmdlCreateDesc](#)()

参数说明

无

返回值说明

返回[aclmdlDesc](#)类型的指针。

1.13.72.2 aclmdlDestroyDesc

函数功能

销毁通过[aclmdlCreateDesc](#)接口创建的[aclmdlDesc](#)类型的数据。

函数原型

[aclError](#) [aclmdlDestroyDesc](#)([aclmdlDesc](#) *modelDesc)

参数说明

参数名	输入/输出	说明
modelDesc	输入	待销毁的 aclmdlDesc 类型的指针。

返回值说明

返回0表示成功，返回非0表示失败。

1.13.72.3 aclmdlGetDesc

函数功能

根据模型ID获取该模型的模型描述信息。

函数原型

```
aclError aclmdlGetDesc(aclmdlDesc *modelDesc, uint32_t modelId)
```

参数说明

参数名	输入/输出	说明
modelDesc	输出	aclmdlDesc类型的指针。 需提前调用 aclmdlCreateDesc 接口创建aclmdlDesc类型的数据。
modelId	输入	模型ID。 调用 aclmdlLoadFromFile 接口/ aclmdlLoadFromMem 接口/ aclmdlLoadFromFileWithMem 接口/ aclmdlLoadFromMemWithMem 接口加载模型成功后，会返回模型ID。

返回值说明

返回0表示成功，返回非0表示失败。

1.13.72.4 aclmdlGetNumInputs

函数功能

根据模型描述信息获取模型的输入个数。

函数原型

```
size_t aclmdlGetNumInputs(aclmdlDesc *modelDesc)
```

参数说明

参数名	输入/输出	说明
modelDesc	输入	aclmdlDesc类型的指针。 需提前调用 aclmdlCreateDesc 接口创建aclmdlDesc类型的数据。

返回值说明

模型的输入个数。

1.13.72.5 aclmdlGetNumOutputs

函数功能

根据模型描述信息获取模型的输出个数。

函数原型

`size_t aclmdlGetNumOutputs(aclmdlDesc *modelDesc)`

参数说明

参数名	输入/输出	说明
modelDesc	输入	aclmdlDesc类型的指针。 需提前调用 aclmdlCreateDesc 接口创建aclmdlDesc类型的数据。

返回值说明

模型的输出个数。

1.13.72.6 aclmdlGetInputSizeByIndex

函数功能

根据模型描述信息获取指定输入的大小，单位为Byte。

约束说明

如果模型输入的Shape是动态的、且维度的取值为-1（表示此维度可以使用 ≥ 1 的任意取值），则通过本接口获取的大小为0，用户需根据实际数据占用的内存大小来申请内存。

函数原型

`size_t aclmdlGetInputSizeByIndex(aclmdlDesc *modelDesc, size_t index)`

参数说明

参数名	输入/输出	说明
modelDesc	输入	aclmdlDesc类型的指针。 需提前调用 aclmdlCreateDesc 接口创建aclmdlDesc类型的数据。
index	输入	指定获取第几个输入的大小，index值从0开始。

返回值说明

针对动态Batch、动态分辨率（宽高）的场景，返回最大档位对应的输入的大小；静态场景下，返回指定输入的大小。单位是Byte。

1.13.72.7 aclmdlGetOutputSizeByIndex

函数功能

根据模型描述信息获取指定输出的大小，单位为Byte。

约束说明

如果通过本接口获取的大小为0，有可能是由于输出Shape的范围不确定，当前支持以下两种处理方式：

- 方式一：**系统内部自行申请对应index的输出内存**，节省内存，但内存数据使用结束后，需由用户释放内存，同时，系统内部申请内存时涉及内存拷贝，可能涉及性能损耗。
调用[aclCreateDataBuffer](#)接口创建存放对应index输出数据的[aclDataBuffer](#)类型时，可在data参数处传入nullptr，表示创建一个空的[aclDataBuffer](#)类型，然后在模型执行过程中，系统内部自行计算并申请该index输出的内存。
- 方式二：**用户预估输出内存大小，并申请内存**，由用户自行管理内存，但内存大小可能不够或超出，不够时系统会校验报错，超出时会浪费内存。
用户需先根据实际情况预估一块较大的输出内存，在模型执行过程中，系统会校验用户指定的输出内存大小是否符合要求，如果不符合要求，系统会返回报错，并在报错信息中提示具体需要多大的输出内存。您可以通过以下两种方式查看报错：
 - 获取应用类日志，查看ERROR级别的报错，如何获取日志请参见《[日志参考](#)》。
 - 在应用程序中调用[aclGetRecentErrMsg](#)接口获取报错。

函数原型

size_t aclmdlGetOutputSizeByIndex([aclmdlDesc](#) *modelDesc, size_t index)

参数说明

参数名	输入/输出	说明
modelDesc	输入	aclmdlDesc 类型的指针。 需提前调用 aclmdlCreateDesc 接口创建 aclmdlDesc 类型的数据。
index	输入	指定获取第几个输出的大小，index值从0开始。

返回值说明

针对动态Batch、动态分辨率（宽高）的场景，返回最大档位对应的输出的大小；静态场景下，返回指定输出的大小。单位是Byte。

1.13.72.8 aclmdlGetInputDims

函数功能

根据模型描述信息获取模型的输入tensor的维度信息。

如果模型中含有静态AIPP配置信息，您可以根据实际需要选择[aclmdlGetInputDims](#)接口或[aclmdlGetInputDimsV2](#)接口查询维度信息，两者的区别在于：

- 通过[aclmdlGetInputDims](#)接口获取的维度信息，各维度的值与输入图像的各维度的值保持一致，详细规则如[表1-112](#)所示。
- 通过[aclmdlGetInputDimsV2](#)接口获取的维度信息，H维度的值 = 输入图像的H维度值*系数，不同图片格式下C维度的值不同，详细规则如[aclmdlGetInputDimsV2](#)接口处的[表1-113](#)所示，且各维度值相乘的结果值与通过[aclmdlGetInputSizeByIndex](#)接口获取的值保持一致。

函数原型

aclError aclmdlGetInputDims(const **aclmdlDesc** *modelDesc, size_t index, **aclmdlIODims** *dims)

参数说明

参数名	输入/输出	说明
modelDesc	输入	aclmdlDesc类型的指针。 需提前调用 aclmdlCreateDesc 接口创建aclmdlDesc类型的数据。
index	输入	指定获取第几个输入的Dims，index值从0开始。

参数名	输入/输出	说明
dims	输出	输入维度信息的指针。 针对动态Batch、动态分辨率（宽高）的场景，输入tensor的dims中batch size或宽高为-1，表示其动态可变。例如，输入tensor的format为NCHW，在动态Batch场景下，动态可变的输入tensor的dims为[-1,3,224,224]；在动态分辨率场景下，动态可变的输入tensor的dims为[1,3,-1,-1]。举例中的斜体部分以实际情况为准。 若tensor的name长度大于127，则在输出的dims.name时，AscendCL会将tensor的name转换为“acl_modelId_{\$id}_input_{\$index}_{\$随机字符串}”格式（如果转换后的tensor的name与模型中已有的tensor的name冲突，则会在转换后的name尾部增加“_{\$随机字符串}”，否则不会增加随机字符串），并在转换后的name与原name之间建立映射关系，用户可调用aclmdlGetTensorRealName接口，传入转换后的name，获取原name（若向接口传入原name，则获取的还是原name）；若tensor的name长度小于或等于127，则在输出的dims.name时，按tensor的name输出。 针对静态AIPP场景，本接口针对不同格式的图像，对应NHWC的Format格式，当前接口中明确各个维度的定义规则，如表1-112所示。

表 1-112 静态 AIPP 场景下的维度定义规则

图像格式	Format参考格式	维度定义规则
YUV420SP_U8	NHWC	n,h,w,c
XRGB8888_U8	NHWC	n,h,w,c
RGB888_U8	NHWC	n,h,w,c
YUV400_U8	NHWC	n,h,w,c
ARGB8888_U8	NHWC	n,h,w,c
YUYV_U8	NHWC	n,h,w,c
YUV422SP_U8	NHWC	n,h,w,c
AYUV444_U8	NHWC	n,h,w,c

返回值说明

返回0表示成功，返回非0表示失败。

1.13.72.9 aclmdlGetInputDimsV2

函数功能

根据模型描述信息获取模型的输入tensor的维度信息。

如果模型中含有静态AIPP配置信息，您可以根据实际需要选择[aclmdlGetInputDims](#)接口或[aclmdlGetInputDimsV2](#)接口查询维度信息，两者的区别在于：

- 通过[aclmdlGetInputDims](#)接口获取的维度信息，各维度的值与输入图像的各维度的值保持一致，详细规则如[表1-112](#)所示。
- 通过[aclmdlGetInputDimsV2](#)接口获取的维度信息，H维度的值 = 输入图像的H维度值*系数，不同图片格式下C维度的值不同，详细规则如[aclmdlGetInputDimsV2](#)接口处的[表1-113](#)所示，且各维度值相乘的结果值与通过[aclmdlGetInputSizeByIndex](#)接口获取的值保持一致。

函数原型

```
aclError aclmdlGetInputDimsV2(const aclmdlDesc *modelDesc, size_t index,
aclmdlIODims *dims)
```

参数说明

参数名	输入/输出	说明
modelDesc	输入	aclmdlDesc类型的指针。 需提前调用 aclmdlCreateDesc 接口创建aclmdlDesc类型的数据。
index	输入	指定获取第几个输入的Dims，index值从0开始。

参数名	输入/输出	说明
dims	输出	输入维度信息的指针。 针对动态Batch、动态分辨率（宽高）的场景，输入tensor的dims中batch size或宽高为-1，表示其动态可变。例如，输入tensor的format为NCHW，在动态Batch场景下，动态可变的输入tensor的dims为[-1,3,224,224]；在动态分辨率场景下，动态可变的输入tensor的dims为[1,3,-1,-1]。举例中的斜体部分以实际情况为准。 若tensor的name长度大于127，则在输出的dims.name时，AscendCL会将tensor的name转换为“acl_modelId_{\$id}_input_{\$index}_{\$随机字符串}”格式（如果转换后的tensor的name与模型中已有的tensor的name冲突，则会在转换后的name尾部增加“_{\$随机字符串}”，否则不会增加随机字符串），并在转换后的name与原name之间建立映射关系，用户可调用aclmdlGetTensorRealName接口，传入转换后的name，获取原name（若向接口传入原name，则获取的还是原name）；若tensor的name长度小于或等于127，则在输出的dims.name时，按tensor的name输出。 针对静态AIPP场景，本接口针对不同格式的图像，对应NHWC的Format格式，当前接口中明确各个维度的定义规则，如表1-113所示。

表 1-113 静态 AIPP 场景下的维度定义规则

图像格式	Format参考格式	维度定义规则
YUV420SP_U8	NHWC	n,h*1.5,w,1
XRGB8888_U8	NHWC	n,h,w,4
RGB888_U8	NHWC	n,h,w,3
YUV400_U8	NHWC	n,h,w,1
ARGB8888_U8	NHWC	n,h,w,4
YUYV_U8	NHWC	n,h,w,2
YUV422SP_U8	NHWC	n,h*2,w,1
AYUV444_U8	NHWC	n,h,w,4

返回值说明

返回0表示成功，返回非0表示失败。

1.13.72.10 aclmdlGetOutputDims

函数功能

根据模型描述信息获取指定的模型输出tensor的维度信息。

固定Shape场景下，通过该接口获取指定的模型输出tensor的维度信息。

动态Shape（动态Batch或动态分辨率）场景下，通过该接口获取最大档的维度信息。

函数原型

```
aclError aclmdlGetOutputDims(const aclmdlDesc *modelDesc, size_t index,
aclmdlIODims *dims)
```

参数说明

参数名	输入/输出	说明
modelDesc	输入	aclmdlDesc类型的指针。 需提前调用 aclmdlCreateDesc 接口创建aclmdlDesc类型的数据。
index	输入	指定获取第几个输出的Dims，index值从0开始。
dims	输出	输出维度信息的指针。 若tensor的name长度大于127，则在输出的dims.name时，系统会将tensor的name转换为“acl_modelId_{id}_output_{index}_{随机字符串}”格式（如果转换后的tensor的name与模型中已有的tensor的name冲突，则会在转换后的name尾部增加“_{随机字符串}”，否则不会增加随机字符串），并在转换后的name与原name之间建立映射关系，用户可调用 aclmdlGetTensorRealName 接口，传入转换后的name，获取原name（若向接口传入原name，则获取的还是原name）；若tensor的name长度小于或等于127，则在输出的dims.name时，按tensor的name输出。

返回值说明

返回0表示成功，返回非0表示失败。

1.13.72.11 aclmdlGetInputNameByIndex

函数功能

根据模型描述信息获取模型中指定输入的输入名称。

函数原型

```
const char *aclmdlGetInputNameByIndex(const aclmdlDesc *modelDesc, size_t index)
```

参数说明

参数名	输入/输出	说明
modelDesc	输入	aclmdlDesc类型的指针。 需提前调用 aclmdlCreateDesc 接口创建aclmdlDesc类型的数据。
index	输入	指定获取第几个输入的名称，index值从0开始。

返回值说明

返回指定输入的输入名称。

1.13.72.12 aclmdlGetOutputNameByIndex

函数功能

根据模型描述信息获取模型中指定输出的输出算子名称、算子输出边的下标、top名称或输出名称。

函数原型

```
const char *aclmdlGetOutputNameByIndex(const aclmdlDesc *modelDesc, size_t index)
```

参数说明

参数名	输入/输出	说明
modelDesc	输入	aclmdlDesc类型的指针。 需提前调用 aclmdlCreateDesc 接口创建aclmdlDesc类型的数据。
index	输入	指定获取第几个输出的输出算子名称、算子输出边下标、top名称或输出名称，index值从0开始。

返回值说明

返回指定输出的输出算子名称、算子输出边的下标、top名称或输出名称。不同原始网络、不同构建模型的方式，调用本接口获取的返回值格式不同。

- Caffe网络
返回值格式如下，各项之间以冒号分割，如果模型中包含top名称就返回，不包含就不返回：
输出算子名称: 算子输出边下标: top名称
- TensorFlow网络
 - 使用ATC工具构建om模型的场景下，返回值格式如下，各项之间以冒号分割：
输出算子名称: 算子输出边下标
 - 使用Ascend Graph接口构建om模型的场景下，返回值格式如下，各项之间以下划线分割：
output_网络输出下标_输出算子名称_算子输出边下标
- ONNX网络
 - 在构建模型时，不指定输出节点名称（ node_name ）或输出名称（ output的name ），或者仅指定输出名称，返回值格式如下，各项之间以冒号分割：
输出算子名称: 算子输出边下标: 输出名称
 - 在构建模型时，指定输出节点名称（ node_name ）：
输出算子名称可能是图融合后的算子名称，也可能是子图名称。
 - 使用ATC工具构建om模型的场景下，返回值格式如下，各项之间以冒号分割：
输出算子名称: 算子输出边下标
 - 使用Ascend Graph接口构建om模型的场景下，返回值格式如下，各项之间以下划线分割：
output_网络输出下标_输出算子名称_算子输出边下标
 - 同时指定输出节点名称（ node_name ）和输出名称（ output的name ），接口返回报错。

1.13.72.13 aclmdlGetInputFormat

函数功能

根据模型描述信息获取模型中指定输入的Format。

函数原型

aclFormat aclmdlGetInputFormat(const **aclmdlDesc** *modelDesc, size_t index)

参数说明

参数名	输入/输出	说明
modelDesc	输入	aclmdlDesc类型的指针。 需提前调用 aclmdlCreateDesc 接口创建aclmdlDesc类型的数据。
index	输入	指定获取第几个输入的Format，index值从0开始。

返回值说明

返回指定输入的Format。

1.13.72.14 aclmdlGetOutputFormat

函数功能

根据模型描述信息获取模型中指定输出的Format。

函数原型

```
aclFormat aclmdlGetOutputFormat(const aclmdlDesc *modelDesc, size_t index)
```

参数说明

参数名	输入/输出	说明
modelDesc	输入	aclmdlDesc类型的指针。 需提前调用 aclmdlCreateDesc 接口创建aclmdlDesc类型的数据。
index	输入	指定获取第几个输出的Format，index值从0开始。

返回值说明

返回指定输出的Format。

1.13.72.15 aclmdlGetInputDataType

函数功能

根据模型描述信息获取模型中指定输入的数据类型。

函数原型

```
aclDataType aclmdlGetInputDataType(const aclmdlDesc *modelDesc, size_t index)
```

参数说明

参数名	输入/输出	说明
modelDesc	输入	aclmdlDesc类型的指针。 需提前调用 aclmdlCreateDesc 接口创建aclmdlDesc类型的数据。

参数名	输入/输出	说明
index	输入	指定获取第几个输入的数据类型，index值从0开始。

返回值说明

返回指定输入的数据类型。

1.13.72.16 aclmdlGetOutputDataType

函数功能

根据模型描述信息获取模型中指定输出的数据类型。

函数原型

aclDataType aclmdlGetOutputDataType(const **aclmdlDesc** *modelDesc, size_t index)

参数说明

参数名	输入/输出	说明
modelDesc	输入	aclmdlDesc类型的指针。 需提前调用 aclmdlCreateDesc 接口创建aclmdlDesc类型的数据。
index	输入	指定获取第几个输出的数据类型，index值从0开始。

返回值说明

返回指定输出的数据类型。

1.13.72.17 aclmdlGetInputIndexByName

函数功能

根据模型中的输入名称获取对应输入的索引编号。

函数原型

aclError aclmdlGetInputIndexByName(const **aclmdlDesc** *modelDesc, const char *name, size_t *index)

参数说明

参数名	输入/输出	说明
modelDesc	输入	aclmdlDesc类型的指针。 需提前调用 aclmdlCreateDesc 接口创建aclmdlDesc类型的数据。
name	输入	输入名称的指针。
index	输出	输入的索引编号的指针。

返回值说明

返回0表示成功，返回非0表示失败。

1.13.72.18 aclmdlGetOutputIndexByName

函数功能

根据模型中的输出名称获取对应输出的索引编号。

函数原型

aclError aclmdlGetOutputIndexByName(const **aclmdlDesc** *modelDesc, const char *name, size_t *index)

参数说明

参数名	输入/输出	说明
modelDesc	输入	aclmdlDesc类型的指针。 需提前调用 aclmdlCreateDesc 接口创建aclmdlDesc类型的数据。
name	输入	输出名称的指针。
index	输出	输出的索引编号的指针。

返回值说明

返回0表示成功，返回非0表示失败。

1.13.72.19 aclmdlGetDynamicBatch

函数功能

根据模型描述信息获取模型支持的动态Batch信息。

函数原型

aclError aclmdlGetDynamicBatch(const **aclmdlDesc** *modelDesc, aclmdlBatch *batch)

参数说明

参数名	输入/输出	说明
modelDesc	输入	aclmdlDesc类型的指针。 需提前调用 aclmdlCreateDesc 接口创建aclmdlDesc类型的数据。
batch	输出	const int ACL_MAX_BATCH_NUM = 128; typedef struct aclmdlBatch { size_t batchCount; /**模型中支持的batch分档数 */ uint64_t batch[ACL_MAX_BATCH_NUM]; /**模型中支持的具体分档 */ } aclmdlBatch; batchCount等于0时，表示不支持设置档位信息，以模型中的档位为准。

返回值说明

返回0表示成功，返回非0表示失败。

1.13.72.20 aclmdlGetDynamicHW

函数功能

根据模型描述信息获取模型支持的动态宽高信息。

函数原型

aclError aclmdlGetDynamicHW(const **aclmdlDesc** *modelDesc, size_t index, aclmdlHW *hw)

参数说明

参数名	输入/输出	说明
modelDesc	输入	aclmdlDesc类型的指针。 需提前调用 aclmdlCreateDesc 接口创建aclmdlDesc类型的数据。
index	输入	预留参数，当前未使用，固定设置为-1。

参数名	输入/输出	说明
hw	输出	<pre>const int ACL_MAX_HW_NUM = 128; typedef struct aclmdlHW { size_t hwCount; /**模型中支持的宽高分档数 */ uint64_t hw[ACL_MAX_HW_NUM][2]; /**模型中支持的具体分档，每 组分档中，数组下标为0代表的是高，下标为1代表的是宽 */ } aclmdlHW; hwCount等于0时，表示不支持设置档位信息，以模型中的 档位为准。</pre>

返回值说明

返回0表示成功，返回非0表示失败。

1.13.72.21 aclmdlGetCurOutputDims

函数功能

根据模型描述信息获取指定的模型输出tensor的实际维度信息。

约束说明

当前仅支持通过本接口获取以下场景中的模型输出tensor的维度信息：

- 通过模型转换设置多档Batch size或分辨率或维度值，实现动态Batch或动态分辨率或动态维度（ND格式）时，如果用户已调用[aclmdlSetDynamicBatchSize](#)设置Batch、或调用[aclmdlSetDynamicHWSIZE](#)接口设置输入图片的宽高、或调用[aclmdlSetInputDynamicDims](#)接口设置某动态维度的值，则可通过该接口获取指定模型输出tensor的实际维度信息；如果用户未调用[aclmdlSetDynamicBatchSize](#)接口、或[aclmdlSetDynamicHWSIZE](#)接口、或[aclmdlSetInputDynamicDims](#)接口，则通过该接口可获取最大档的维度信息。
- 固定Shape场景下，通过该接口获取指定的模型输出tensor的维度信息。

函数原型

`aclError aclmdlGetCurOutputDims(const aclmdlDesc *modelDesc, size_t index, aclmdlIODims *dims)`

参数说明

参数名	输入/输出	说明
modelDesc	输入	aclmdlDesc类型的指针。 需提前调用 aclmdlCreateDesc 接口创建aclmdlDesc类型的数据。
index	输入	指定获取第几个输出的Dims，index值从0开始。

参数名	输入/输出	说明
dims	输出	输出实际维度信息的指针。 若tensor的name长度大于127，则在输出的dims.name时，系统会将tensor的name转换为“acl_modelId_\${id}_output_\${index}_\${随机字符串}”格式（如果转换后的tensor的name与模型中已有的tensor的name冲突，则会在转换后的name尾部增加“_\${随机字符串}”，否则不会增加随机字符串），并在转换后的name与原name之间建立映射关系，用户可调用 aclmdlGetTensorRealName 接口，传入转换后的name，获取原name（若向接口传入原name，则获取的还是原name）；若tensor的name长度小于或等于127，则在输出的dims.name时，按tensor的name输出。

返回值说明

返回0表示成功，返回非0表示失败。

1.13.72.22 aclmdlGetInputDynamicGearCount

函数功能

根据模型描述信息获取模型的输入所支持的动态维度档位数。

约束说明

如果模型构建时没有设置动态维度的分档，那么通过该接口获取的动态维度档位数为0。模型构建的详细说明请参见模型构建。

函数原型

```
aclError aclmdlGetInputDynamicGearCount(const aclmdlDesc *modelDesc,
size_t index, size_t *gearCount)
```

参数说明

参数名	输入/输出	说明
modelDesc	输入	aclmdlDesc类型的指针。 需提前调用 aclmdlCreateDesc 接口创建aclmdlDesc类型的数据。
index	输入	预留参数，当前未使用，固定设置为-1。

参数名	输入/输出	说明
gearCount	输出	动态维度档位数的指针。 例如，模型的输入tensor是4维，在模型转换时通过--dynamic_dims参数设置的分档为“1,3,224,224;2,3,224,224;1,3,256,256”，那么通过该接口获取的动态维度档位数为3。

返回值说明

返回0表示成功，返回非0表示失败。

1.13.72.23 aclmdlGetInputDynamicDims

函数功能

根据模型描述信息获取模型的输入所支持的动态维度信息。

约束说明

只有在模型构建时设置了动态维度的分档信息后，才可以调用该接口获取动态维度信息。模型构建的详细说明请参见模型构建。

例如，模型有三个输入，分别为data(1, 1, 40, -1)，label(1, -1)，mask(-1, -1)，其中-1表示动态可变。在模型转换时，dynamic_dims参数的配置示例为：--dynamic_dims="20,20,1,1;40,40,2,2;80,60,4,4"，则通过本接口获取的动态维度信息为（[aclmdlIODims](#)结构体内的name暂不使用）：

- 第0档：
 - [aclmdlIODims](#)结构体内dimCount：8，表示所有输入tensor的维度数量之和
 - [aclmdlIODims](#)结构体内的dims：“1,1,40,20,1,20,1,1”，表示data(1,1,40,20)+label(1,20)+mask(1,1)
- 第1档：
 - [aclmdlIODims](#)结构体内dimCount：8，表示所有输入tensor的维度数量之和
 - [aclmdlIODims](#)结构体内的dims：“1,1,40,40,1,40,2,2”，表示data(1,1,40,40)+label(1,40)+mask(2,2)
- 第2档：
 - [aclmdlIODims](#)结构体内dimCount：8，表示所有输入tensor的维度数量之和
 - [aclmdlIODims](#)结构体内的dims：“1,1,40,80,1,60,4,4”，表示data(1,1,40,80)+label(1,60)+mask(4,4)

函数原型

```
aclError aclmdlGetInputDynamicDims(const aclmdlDesc *modelDesc, size_t index, aclmdlIODims *dims, size_t gearCount)
```

参数说明

参数名	输入/输出	说明
modelDesc	输入	aclmdlDesc类型的指针。 需提前调用 aclmdlCreateDesc 接口创建aclmdlDesc类型的数据。
index	输入	预留参数，当前未使用，固定设置为-1。
gearCount	输入	模型支持的动态维度档位数，需要先通过 aclmdlGetInputDynamicGearCount 接口获取。
dims	输出	输入的动态维度信息的指针。 dims参数是一个数组，数组中的每个元素指向 aclmdlIODims 结构， aclmdlIODims 结构体中的dims参数也是一个数组，该数组中的每个元素对应每一档中的具体值。 例如： <pre>aclmdlIODims dims[gearCount]; aclmdlGetInputDynamicDims(model.modelDesc, -1, dims, gearCount);</pre>

返回值说明

返回0表示成功，返回非0表示失败。

1.13.72.24 aclmdlGetTensorRealName

函数功能

根据指定名称获取tensor的真实名称。

aclmdlGetTensorRealName接口需要与[aclmdlGetInputDims](#)/[aclmdlGetInputDimsV2](#)/[aclmdlGetOutputDims](#)/[aclmdlGetCurOutputDims](#)接口配合使用。

函数原型

```
const char *aclmdlGetTensorRealName(const aclmdlDesc *modelDesc, const char *name)
```

参数说明

参数名	输入/输出	说明
modelDesc	输入	aclmdlDesc类型的指针。 需提前调用 aclmdlCreateDesc 接口创建aclmdlDesc类型的数据。

参数名	输入/输出	说明
name	输入	名称的指针，用于根据该名称获取tensor的真实名称。

返回值说明

返回指向tensor真实名称的指针，该指针的生命周期与modelDesc相同，若modelDesc资源被销毁，则该指针指向的内容也会自动被销毁。

若modelDesc或name为空，则返回nullptr。

1.13.72.25 aclmdlGetOpAttr

函数功能

获取整网中某个模型中某个算子的属性的值。

函数原型

const char *aclmdlGetOpAttr(aclmdlDesc *modelDesc, const char *opName, const char *attr)

参数说明

参数名	输入/输出	说明
modelDesc	输入	aclmdlDesc类型的指针。 需提前调用 aclmdlCreateDesc 接口创建aclmdlDesc类型的数据，再调用 aclmdlGetDesc 接口根据模型ID获取到对应的aclmdlDesc类型的数据。
opName	输入	算子名称的指针。
attr	输入	算子属性的指针。 当前仅支持_datadump_original_op_names属性，用于记录某个算子是由哪些算子融合得到的。通过本接口获取到的_datadump_original_op_names属性值格式为[opName1_len]opName1...[opNameN_len]opNameN，opNameN_len表示算子名称字符串的长度。 _datadump_original_op_names属性值示例如下，表示某个融合算子由scale2c_branch2c、bn2c_branch2c、res2c_branch2c、res2c、res2c_relu这五个算子融合而成的，算子名称字符串的长度分别为16、13、14、5、10： [16]scale2c_branch2c[13]bn2c_branch2c[14]res2c_branch2c[5]res2c[10]res2c_relu

返回值说明

返回属性值的字符串指针，该指针的生命周期与modelDesc相同，若modelDesc资源被销毁，则该指针指向的内容也会自动被销毁。若opName或者attr属性不存在、或者attr属性值为空，均返回指向空字符串的指针。

若调用该接口失败，则返回nullptr。

1.13.72.26 aclmdlGetDescFromFile

函数功能

根据模型文件获取该模型的模型描述信息。

函数原型

aclError aclmdlGetDescFromFile(**aclmdlDesc** *modelDesc, const char *modelPath)

约束说明

通过本接口获取到的模型描述信息，无法应用于**aclmdlGetOpAttr**、**aclmdlGetCurOutputDims**接口。

参数说明

参数名	输入/输出	说明
modelDesc	输出	aclmdlDesc类型的指针。 需提前调用 aclmdlCreateDesc 接口创建aclmdlDesc类型的数据。
modelPath	输入	模型文件路径的指针，路径中包含文件名。运行程序（APP）的用户需要对该存储路径有访问权限。 此处的模型文件是适配昇腾AI处理器的离线模型，即*.om文件。关于如何获取om文件，请参见《 ATC工具使用指南 》。

返回值说明

返回0表示成功，返回非0表示失败。

1.13.72.27 aclmdlGetDescFromMem

函数功能

从内存获取该模型的模型描述信息。

函数原型

aclError aclmdlGetDescFromMem(**aclmdlDesc** *modelDesc, const void *model, size_t modelSize)

约束说明

通过本接口获取到的模型描述信息，无法应用于**aclmdlGetOpAttr**、**aclmdlGetCurOutputDims**接口。

参数说明

参数名	输入/输出	说明
modelDesc	输出	aclmdlDesc类型的指针。 需提前调用 aclmdlCreateDesc 接口创建aclmdlDesc类型的数据。
model	输入	存放模型数据的内存地址指针。 应用运行在Host时，此处需申请Host上的内存；应用运行在Device时，此处需申请Device上的内存。内存申请接口请参见 1.5.5 内存管理 。
modelSize	输入	内存中的模型数据长度，单位Byte。

返回值说明

返回0表示成功，返回非0表示失败。

1.13.72.28 aclmdlGetInputDimsRange

函数功能

根据模型描述信息获取模型的输入tensor的维度范围。

函数原型

aclError aclmdlGetInputDimsRange(const **aclmdlDesc** *modelDesc, size_t index, **aclmdlIODimsRange** *dimsRange)

参数说明

参数名	输入/输出	说明
modelDesc	输入	aclmdlDesc类型的指针。 需提前调用 aclmdlCreateDesc 接口创建aclmdlDesc类型的数据。

参数名	输入/输出	说明
index	输入	指定获取第几个输入的维度信息，index值从0开始。
dimsRange	输出	输入维度信息的指针。 - 动态Shape且配置Shape范围的场景下，返回Shape范围，动态维度上下限不相同，静态维度上下限相同。例如，输入tensor的format为NCHW，dims为[1,3,10~224,10~300]，N和C维上是固定值，H和W维上是动态范围，那么通过本接口aclmdlIODimsRange.rangeCount为4，aclmdlIODimsRange.range数组中有4个元素，值分别为：{1,1}、{3,3}、{10,224}、{10,300}。 - 动态Shape且配置分档的场景下，不支持使用本接口，aclmdlIODimsRange.rangeCount返回0。 - 静态Shape场景下，返回Shape范围，但各维度上下限相同。

返回值说明

返回0表示成功，返回非0表示失败。

1.13.73 aclTensorDesc

1.13.73.1 aclCreateTensorDesc

函数功能

创建aclTensorDesc类型的数据，该数据类型用于描述tensor的数据类型、shape、format等信息。

如需销毁aclTensorDesc类型的数据，请参见[aclDestroyTensorDesc](#)。

函数原型

```
aclTensorDesc *aclCreateTensorDesc(aclDataType dataType,
int numDims,
const int64_t *dims,
aclFormat format)
```

参数说明

参数名	输入/输出	说明
dataType	输入	tensor描述的数据类型。

参数名	输入/输出	说明
numDims	输入	tensor描述shape的维度个数。 numDims存在以下约束： - numDims必须大于或等于0； - numDims>0时，numDims的值必须与dims数组的长度保持一致； - numDims=0时，系统不使用dims数组值，dims参数值无效。
dims	输入	tensor描述维度大小的指针。 dims是一个数组，数组中的每个元素表示tensor中每个维度的大小，如果数组中某个元素的值为0，则为空tensor。 如果用户需要使用空tensor，则在申请内存时，内存大小最小为1Byte，以保障后续业务正常运行。
format	输入	tensor描述的format。

返回值说明

返回aclTensorDesc类型的指针。

1.13.73.2 aclDestroyTensorDesc

函数功能

销毁aclTensorDesc类型的数据。

函数原型

void aclDestroyTensorDesc(const [aclTensorDesc](#) *desc)

参数说明

参数名	输入/输出	说明
desc	输入	待销毁的aclTensorDesc类型的指针。

返回值说明

无

1.13.73.3 aclGetTensorDescType

函数功能

获取tensor描述中的数据类型。

函数原型

`aclDataType aclGetTensorDescType(const aclTensorDesc *desc)`

参数说明

参数名	输入/输出	说明
desc	输入	aclTensorDesc类型的指针。 需提前调用 aclCreateTensorDesc 接口创建aclTensorDesc类型。

返回值说明

返回指定tensor描述的数据类型。

1.13.73.4 aclGetTensorDescFormat

函数功能

获取tensor描述中的format。

函数原型

`aclFormat aclGetTensorDescFormat(const aclTensorDesc *desc)`

参数说明

参数名	输入/输出	说明
desc	输入	aclTensorDesc类型的指针。 需提前调用 aclCreateTensorDesc 接口创建aclTensorDesc类型。

返回值说明

返回指定tensor描述的format。

1.13.73.5 aclGetTensorDescSize

函数功能

获取tensor数据占用的空间大小。

函数原型

`size_t aclGetTensorDescSize(const aclTensorDesc *desc)`

参数说明

参数名	输入/输出	说明
desc	输入	aclTensorDesc类型的指针。 需提前调用 aclCreateTensorDesc 接口创建aclTensorDesc类型。

返回值说明

返回指定tensor描述占用的空间大小。

1.13.73.6 aclGetTensorDescElementCount

函数功能

获取tensor描述中的元素个数。

函数原型

`size_t aclGetTensorDescElementCount(const aclTensorDesc *desc)`

参数说明

参数名	输入/输出	说明
desc	输入	aclTensorDesc类型的指针。 需提前调用 aclCreateTensorDesc 接口创建aclTensorDesc类型。

返回值说明

返回指定tensor描述中的元素个数。

1.13.73.7 aclGetTensorDescNumDims

函数功能

获取tensor描述中shape的维度个数。

函数原型

```
size_t aclGetTensorDescNumDims(const aclTensorDesc *desc)
```

参数说明

参数名	输入/输出	说明
desc	输入	aclTensorDesc类型的指针。 需提前调用 aclCreateTensorDesc 接口创建aclTensorDesc类型。

返回值说明

返回tensor描述中shape的维度个数。若返回ACL_UNKNOWN_RANK (#define ACL_UNKNOWN_RANK 0xFFFFFFFFFFFFFFFE) 表示动态Shape场景下维度个数未知。

1.13.73.8 aclGetTensorDescDim (废弃)

须知

此接口后续版本会废弃，请使用[aclGetTensorDescDimV2](#)接口。

函数功能

获取tensor描述中指定维度的大小。

函数原型

```
int64_t aclGetTensorDescDim(const aclTensorDesc *desc, size_t index)
```

参数说明

参数名	输入/输出	说明
desc	输入	aclTensorDesc类型的指针。 需提前调用 aclCreateTensorDesc 接口创建aclTensorDesc类型。

参数名	输入/输出	说明
index	输入	指定获取第几个维度的大小，index值从0开始。 用户调用 aclGetTensorDescNumDims 接口获取shape维度个数，这个Index的取值范围：[0, (shape维度个数-1)]。

返回值说明

返回指定tensor描述中指定维度的大小。

1.13.73.9 aclGetTensorDescDimV2

函数功能

获取tensor描述中指定维度的大小。

约束说明

当[aclGetTensorDescNumDims](#)接口的返回值为ACL_UNKNOWN_RANK时，表示动态Shape场景下维度个数未知，则不能调用aclGetTensorDescDimV2接口获取指定维度的大小。

函数原型

aclError aclGetTensorDescDimV2(const **aclTensorDesc** *desc, size_t index, int64_t *dimSize)

参数说明

参数名	输入/输出	说明
desc	输入	aclTensorDesc类型的指针。 需提前调用 aclCreateTensorDesc 接口创建aclTensorDesc类型。
index	输入	指定获取第几个维度的大小，index值从0开始。 用户调用 aclGetTensorDescNumDims 接口获取shape维度个数，这个Index的取值范围：[0, (shape维度个数-1)]。
dimSize	输出	tensor描述中index指定维度大小的指针。 当返回的dimSize参数值为-1时，表示维度的大小是动态的。

返回值说明

返回0表示成功，返回其他值表示失败。

1.13.73.10 aclGetTensorDescDimRange

函数功能

获取tensor描述中指定维度的范围，[1,-1]表示全shape范围。

约束说明

当[aclGetTensorDescNumDims](#)接口的返回值为ACL_UNKNOWN_RANK时，表示动态Shape场景下维度个数未知，则不能调用[aclGetTensorDescDimRange](#)接口获取指定维度的范围。

函数原型

aclError [aclGetTensorDescDimRange](#)(const [aclTensorDesc](#) *desc, size_t index, size_t dimRangeNum, int64_t *dimRange)

参数说明

参数名	输入/输出	说明
desc	输入	aclTensorDesc类型的指针。 需提前调用 aclCreateTensorDesc 接口创建aclTensorDesc类型。
index	输入	指定获取第几个维度的大小，index值从0开始。 用户调用 aclGetTensorDescNumDims 接口获取shape维度个数，这个Index的取值范围：[0, (shape维度个数-1)]。
dimRangeNum	输入	dimRange的长度，该值必须大于等于2。
dimRange	输出	tensor描述中index指定维度的Shape范围。 dimRange是一个数组，数组的第一个元素值表示Shape范围的最小值，第二个元素值表示Shape范围的最大值。该数组中仅前2个元素值有效。 当dimRange数组的值为[1,-1]时，表示全Shape范围。

返回值说明

返回0表示成功，返回其他值表示失败。

1.13.73.11 aclSetTensorDescName

函数功能

设置TensorDesc的名称。

函数原型

```
void aclSetTensorDescName(aclTensorDesc *desc, const char *name)
```

参数说明

参数名	输入/输出	说明
desc	输出	aclTensorDesc类型的指针。 需提前调用 aclCreateTensorDesc 接口创建aclTensorDesc类型。
name	输入	TensorDesc名称的指针。

返回值说明

无

1.13.73.12 aclGetTensorDescName

函数功能

获取TensorDesc的名称。

函数原型

```
const char *aclGetTensorDescName(aclTensorDesc *desc)
```

参数说明

参数名	输入/输出	说明
desc	输入	aclTensorDesc类型的指针。 需提前调用 aclCreateTensorDesc 接口创建aclTensorDesc类型。

返回值说明

返回TensorDesc的名称。

1.13.73.13 aclTransTensorDescFormat

函数功能

按指定dstFormat转换源aclTensorDesc中的format，生成新的目标aclTensorDesc，源aclTensorDesc中的format保持不变。同步接口。不支持该接口，预留接口。

函数原型

aclError aclTransTensorDescFormat(const **aclTensorDesc** *srcDesc, **aclFormat** dstFormat, **aclTensorDesc** **dstDesc)

参数说明

参数名	输入/输出	说明
srcDesc	输入	源aclTensorDesc数据的指针。 需提前调用 aclCreateTensorDesc 接口创建aclTensorDesc类型。
dstFormat	输入	需要设置的目标format。
dstDesc	输出	“目标aclTensorDesc数据指针”的指针。 需提前调用 aclCreateTensorDesc 接口创建aclTensorDesc类型。

返回值说明

返回0表示成功，返回其他值表示失败。

1.13.73.14 aclSetTensorStorageShape（废弃）

须知

此接口后续版本会废弃，请使用**aclSetTensorShape**接口。

函数功能

调用**aclCreateTensorDesc**接口创建tensor描述信息后，可通过aclSetTensorStorageShape接口设置tensor的实际dims信息，当前主要用于pytorch场景下。

Atlas 训练系列产品，不支持该接口。

函数原型

aclError aclSetTensorStorageShape(**aclTensorDesc** *desc, int numDims, const **int64_t** *dims)

参数说明

参数名	输入/输出	说明
desc	输出	aclTensorDesc类型的指针。 需提前调用 aclCreateTensorDesc 接口创建aclTensorDesc类型。
numDims	输入	需要设置的dims维度数。
dims	输入	dims的指针。

返回值说明

返回0表示成功，返回其他值表示失败。

1.13.73.15 aclSetTensorStorageFormat（废弃）

须知

此接口后续版本会废弃，请使用[aclSetTensorFormat](#)接口。

函数功能

调用[aclCreateTensorDesc](#)接口创建tensor描述信息后，可通过aclSetTensorStorageFormat设置tensor的实际Format信息，当前主要用于pytorch场景下。

Atlas 训练系列产品，不支持该接口。

函数原型

[aclError](#) [aclSetTensorStorageFormat](#)([aclTensorDesc](#) *desc, [aclFormat](#) format)

参数说明

参数名	输入/输出	说明
desc	输出	aclTensorDesc类型的指针。 需提前调用 aclCreateTensorDesc 接口创建aclTensorDesc类型。
format	输入	需要设置的format。

返回值说明

返回0表示成功，返回其他值表示失败。

1.13.73.16 aclSetTensorOriginShape

函数功能

调用[aclCreateTensorDesc](#)接口创建tensor描述信息后，可通过[aclSetTensorOriginShape](#)接口设置tensor的原始dims信息，当前主要用于pytorch场景下。

 说明

例如，某算子的原始Format为NHWC（原始Shape为4维）或者NDHWC（原始Shape为5维），为了提高数据访问效率，在进行模型转换或者在线构建网络时会自动转换Format，将原始数据Format转化为昇腾AI处理器内部计算数据时用的Format（NC1HWC0或者NDC1HWC0），同时进行推导Shape用于内部计算。

函数原型

aclError [aclSetTensorOriginShape](#)([aclTensorDesc](#) *desc, int numDims, const int64_t *dims)

参数说明

参数名	输入/输出	说明
desc	输出	aclTensorDesc 类型的指针。 需提前调用 aclCreateTensorDesc 接口创建 aclTensorDesc 类型。
numDims	输入	需要设置的dims维度数。
dims	输入	dims的指针。

返回值说明

返回0表示成功，返回其他值表示失败。

1.13.73.17 aclSetTensorOriginFormat

函数功能

调用[aclCreateTensorDesc](#)接口创建tensor描述信息后，可通过[aclSetTensorOriginFormat](#)设置tensor的原始Format信息，当前主要用于pytorch场景下。

 说明

例如，某算子的原始Format为NHWC（原始Shape为4维）或者NDHWC（原始Shape为5维），为了提高数据访问效率，在进行模型转换或者在线构建网络时会自动转换Format，将原始数据Format转化为昇腾AI处理器内部计算数据时用的Format（NC1HWC0或者NDC1HWC0），同时进行推导Shape用于内部计算。

函数原型

aclError **aclSetTensorOriginFormat**(**aclTensorDesc** *desc, **aclFormat** format)

参数说明

参数名	输入/输出	说明
desc	输出	aclTensorDesc类型的指针。 需提前调用 aclCreateTensorDesc 接口创建aclTensorDesc类型。
format	输入	需要设置的format。

返回值说明

返回0表示成功，返回其他值表示失败。

1.13.73.18 **aclSetTensorShape**

函数功能

调用**aclCreateTensorDesc**接口创建tensor描述信息后，可通过**aclSetTensorShape**接口设置昇腾AI处理器内部处理tensor时的dims信息，当前主要用于pytorch场景下。

Atlas 训练系列产品，不支持该接口。

 说明

例如，某算子的原始Format为NHWC（原始Shape为4维）或者NDHWC（原始Shape为5维），为了提高数据访问效率，在进行模型转换或者在线构建网络时会自动转换Format，将原始数据Format转化为昇腾AI处理器内部计算数据时用的Format（NC1HWC0或者NDC1HWC0），同时进行推导Shape用于内部计算。

函数原型

aclError **aclSetTensorShape**(**aclTensorDesc** *desc, int numDims, const int64_t *dims)

参数说明

参数名	输入/输出	说明
desc	输出	aclTensorDesc类型的指针。 需提前调用 aclCreateTensorDesc 接口创建aclTensorDesc类型。
numDims	输入	需要设置的dims维度数。
dims	输入	dims的指针。

返回值说明

返回0表示成功，返回其他值表示失败。

1.13.73.19 aclSetTensorFormat

函数功能

调用[aclCreateTensorDesc](#)接口创建tensor描述信息后，可通过aclSetTensorFormat设置昇腾AI处理器内部处理tensor时的Format信息，当前主要用于pytorch场景下。

Atlas 训练系列产品，不支持该接口。

说明

例如，某算子的原始Format为NHWC（原始Shape为4维）或者NDHWC（原始Shape为5维），为了提高数据访问效率，在进行模型转换或者在线构建网络时会自动转换Format，将原始数据Format转化为昇腾AI处理器内部计算数据时用的Format（NC1HWC0或者NDC1HWC0），同时进行推导Shape用于内部计算。

函数原型

aclError [aclSetTensorFormat](#)([aclTensorDesc](#) *desc, [aclFormat](#) format)

参数说明

参数名	输入/输出	说明
desc	输出	aclTensorDesc类型的指针。 需提前调用 aclCreateTensorDesc 接口创建aclTensorDesc类型。
format	输入	需要设置的format。

返回值说明

返回0表示成功，返回其他值表示失败。

1.13.73.20 aclSetTensorShapeRange

函数功能

调用[aclCreateTensorDesc](#)接口创建tensor描述信息后，可通过本接口设置tensor的各个维度的取值范围。

使用场景：动态Shape的算子，其输入Shape中变化维度用-1表示，但每个变化维度的范围是不一样的，需要显式设置，如Shape为[16,-1,20,-1]，对应的shape range可以是[[16,16],[1,128],[20,20],[1,10]]，表示第一维的Shape范围固定为16，第二维的Shape范围为1到128，第三维的Shape范围固定为20，第四维的Shape范围为1到10。

函数原型

aclError **aclSetTensorShapeRange**(**aclTensorDesc*** desc, **size_t** dimsCount, **int64_t** dimsRange[][ACL_TENSOR_SHAPE_RANGE_NUM])

参数说明

参数名	输入/输出	说明
desc	输出	aclTensorDesc类型的指针。 需提前调用 aclCreateTensorDesc 接口创建aclTensorDesc类型。
dimsCount	输入	tensor中dims的维度个数。
dimsRange	输入	dimsRange为每个维度的取值范围，用二维数组表示范围。 如果Shape中的维度值不是-1时，表示固定维度，该维度对应的shape range中最大值和最小值相同；否则，表示动态维度，数组的两个维度分别表示对应tensor dims中的最小值和最大值。 #define ACL_TENSOR_SHAPE_RANGE_NUM 2

返回值说明

返回0表示成功，返回其他值表示失败。

1.13.73.21 aclSetTensorDynamicInput

函数功能

调用[aclCreateTensorDesc](#)接口创建tensor描述信息后，可通过[aclSetTensorDynamicInput](#)接口设置多个动态输入的分组标识，例如某个算子包括动态输入为x、必选输入y，其中动态输入有多个，调用[aclSetTensorDynamicInput](#)将**dynamicInputName**设置为x，AscendCL内部根据**dynamicInputName**索引到对应的多个动态输入，达到分组的目的。

函数原型

```
aclError aclSetTensorDynamicInput(aclTensorDesc *desc, const char *dynamicInputName)
```

参数说明

参数名	输入/输出	说明
desc	输出	aclTensorDesc类型的指针。 需提前调用 aclCreateTensorDesc 接口创建aclTensorDesc类型。
dynamicInputName	输入	desc中动态输入分组标识的指针。

返回值说明

返回0表示成功，返回其他值表示失败。

1.13.73.22 aclGetTensorDescByIndex

函数功能

获取算子的指定Index的tensor描述信息。

函数原型

```
aclTensorDesc *aclGetTensorDescByIndex(aclTensorDesc *desc, size_t index)
```

参数说明

参数名	输入/输出	说明
desc	输入	aclTensorDesc类型的指针。 调用 aclmdlCreateAndGetOpDesc 接口获取算子所有输入/输出的tensor描述，作为本接口的输入。
index	输入	指定获取第几个输入/输出的tensor描述，index值从0开始。 用户调用 aclmdlCreateAndGetOpDesc 接口获取的算子输入/输出的数量后，这个Index的取值范围：[0, (算子输入/输出的数量-1)]。

返回值说明

返回指定输入/输出的aclTensorDesc类型数据的指针。

1.13.73.23 aclGetTensorDescAddress

函数功能

获取指定算子输入/输出的tensor数据的内存地址。

函数原型

```
void *aclGetTensorDescAddress(const aclTensorDesc *desc)
```

参数说明

参数名	输入/输出	说明
desc	输入	aclTensorDesc类型的指针。 调用 aclGetTensorDescByIndex 接口获取算子的指定输入/输出的tensor描述，作为本接口的输入。

返回值说明

返回指定算子输入/输出的tensor数据的内存地址。

1.13.73.24 aclSetTensorConst

函数功能

为算子设置constant输入，指定存放输入tensor数据的内存地址和地址长度。

函数原型

```
aclError aclSetTensorConst(aclTensorDesc *desc, void *dataBuffer, size_t length)
```

参数说明

参数名	输入/输出	说明
desc	输出	aclTensorDesc类型的指针。 需提前调用 aclCreateTensorDesc 接口创建aclTensorDesc类型。
dataBuffer	输入	输入tensor数据内存地址指针。 此处算子输入tensor数据的内存必须根据应用运行模式来确定，应用运行在Host时，此处需申请Host上的内存；应用运行在Device时，此处需申请Device上的内存。内存申请接口请参见 1.5.5 内存管理 。
length	输入	内存地址的长度，单位是Byte。

返回值说明

返回0表示成功，返回其他值表示失败。

1.13.73.25 aclSetTensorPlaceMent

函数功能

设置TensorDesc的placement属性，用于标识存放Tensor数据的内存类型。

函数原型

aclError aclSetTensorPlaceMent(**aclTensorDesc** *desc, **aclMemType** memType)

参数说明

参数名	输入/输出	说明
desc	输出	aclTensorDesc类型的指针。 需提前调用 aclCreateTensorDesc 接口创建aclTensorDesc类型。
memType	输入	指定placement属性值。

返回值说明

返回0表示成功，返回其他值表示失败。

1.13.73.26 aclSetTensorValueRange

函数功能

调用**aclCreateTensorDesc**接口创建tensor描述信息后，可通过本接口设置tensor的数据值的范围。

使用场景：部分算子input的值就是该算子的输出Shape，在动态Shape场景下，这个Shape的值就有一个取值范围，在执行算子时，需要在input上设置Shape值的范围（例如：[[16,16],[1,128],[20,20],[1,10]]），AscendCL才能正常编译、执行算子。

函数原型

aclError aclSetTensorValueRange(**aclTensorDesc*** desc, **size_t** valueCount, **int64_t** valueRange[][ACL_TENSOR_VALUE_RANGE_NUM])

参数说明

参数名	输入/输出	说明
desc	输出	aclTensorDesc类型的指针。 需提前调用 aclCreateTensorDesc 接口创建aclTensorDesc类型。
valueCount	输入	需设置范围的数据值的个数。
valueRange	输入	valueRange为每个数据值的范围，用二维数组表示范围。 #define ACL_TENSOR_VALUE_RANGE_NUM 2

返回值说明

返回0表示成功，返回其他值表示失败。

1.13.74 aclopAttr

1.13.74.1 aclopCreateAttr

函数功能

创建aclopAttr类型的数据，该数据类型用于保存算子的属性。
如需销毁aclopAttr类型的数据，请参见[aclopDestroyAttr](#)。

函数原型

[aclopAttr](#) *aclopCreateAttr()

参数说明

无

返回值说明

返回aclopAttr类型的指针。

1.13.74.2 aclopDestroyAttr

函数功能

销毁通过[aclopCreateAttr](#)接口创建的aclopAttr类型的数据。

函数原型

void aclopDestroyAttr(const [aclopAttr](#) *attr)

参数说明

参数名	输入/输出	说明
attr	输入	待销毁的aclopAttr类型指针。

返回值说明

无

1.13.74.3 aclopSetAttrBool

函数功能

设置bool类型标量的属性值。

函数原型

```
aclError aclopSetAttrBool(aclopAttr *attr, const char *attrName, uint8_t attrValue)
```

参数说明

参数名	输入/输出	说明
attr	输出	aclopAttr类型的指针。 需提前调用 aclopCreateAttr 接口创建aclopAttr类型。
attrName	输入	属性名的指针。
attrValue	输入	属性值。

返回值说明

返回0表示成功，返回非0表示失败。

1.13.74.4 aclopSetAttrInt

函数功能

设置int64_t类型标量的属性值。

函数原型

```
aclError aclopSetAttrInt(aclopAttr *attr, const char *attrName, int64_t attrValue)
```

参数说明

参数名	输入/输出	说明
attr	输出	aclopAttr类型的指针。 需提前调用 aclopCreateAttr 接口创建aclopAttr类型。
attrName	输入	属性名的指针。
attrValue	输入	属性值。

返回值说明

返回0表示成功，返回非0表示失败。

1.13.74.5 aclopSetAttrFloat

函数功能

设置float类型标量的属性值。

函数原型

aclError aclopSetAttrFloat(**aclopAttr** *attr, const char *attrName, float attrValue)

参数说明

参数名	输入/输出	说明
attr	输出	aclopAttr类型的指针。 需提前调用 aclopCreateAttr 接口创建aclopAttr类型。
attrName	输入	属性名的指针。
attrValue	输入	属性值。

返回值说明

返回0表示成功，返回非0表示失败。

1.13.74.6 aclopSetAttrString

函数功能

设置字符串类型标量的属性值。

函数原型

aclError aclopSetAttrString(**aclopAttr** *attr, const char *attrName, const char *attrValue)

参数说明

参数名	输入/输出	说明
attr	输出	aclopAttr类型的指针。 需提前调用 aclopCreateAttr 接口创建aclopAttr类型。
attrName	输入	属性名的指针。
attrValue	输入	属性值的指针。

返回值说明

返回0表示成功，返回非0表示失败。

1.13.74.7 aclopSetAttrListBool

函数功能

设置bool类型列表的属性值。

函数原型

aclError aclopSetAttrListBool(**aclopAttr** *attr, const char *attrName, int numValues, const uint8_t *values)

参数说明

参数名	输入/输出	说明
attr	输出	aclopAttr类型的指针。 需提前调用 aclopCreateAttr 接口创建aclopAttr类型。
attrName	输入	属性名的指针。
numValues	输入	属性值数目。
values	输入	属性值的数组。

返回值说明

返回0表示成功，返回非0表示失败。

1.13.74.8 aclopSetAttrListInt

函数功能

设置int64_t类型列表的属性值。

函数原型

```
aclError aclopSetAttrListInt(aclopAttr *attr, const char *attrName, int numValues, const int64_t *values)
```

参数说明

参数名	输入/输出	说明
attr	输出	aclopAttr类型的指针。 需提前调用 aclopCreateAttr 接口创建aclopAttr类型。
attrName	输入	属性名的指针。
numValues	输入	属性值数目。
values	输入	属性值的数组。

返回值说明

返回0表示成功，返回非0表示失败。

1.13.74.9 aclopSetAttrListFloat

函数功能

设置float类型列表的属性值。

函数原型

```
aclError aclopSetAttrListFloat(aclopAttr *attr, const char *attrName, int numValues, const float *values)
```

参数说明

参数名	输入/输出	说明
attr	输出	aclopAttr类型的指针。 需提前调用 aclopCreateAttr 接口创建aclopAttr类型。
attrName	输入	属性名的指针。
numValues	输入	属性值数目。

参数名	输入/输出	说明
values	输入	属性值的数组。

返回值说明

返回0表示成功，返回非0表示失败。

1.13.74.10 aclopSetAttrListString

函数功能

设置字符串类型列表的属性值。

函数原型

aclError aclopSetAttrListString(**aclopAttr** *attr, const char *attrName, int numValues, const char **values)

参数说明

参数名	输入/输出	说明
attr	输出	aclopAttr类型的指针。 需提前调用 aclopCreateAttr 接口创建aclopAttr类型。
attrName	输入	属性名的指针。
numValues	输入	属性值数目。
values	输入	“属性值的数组”的指针。

返回值说明

返回0表示成功，返回非0表示失败。

1.13.74.11 aclopSetAttrListListInt

函数功能

设置int64_t类型列表的属性值。

函数原型

aclError aclopSetAttrListListInt(**aclopAttr** *attr,
const char *attrName,

```
int numLists,  
const int *numValues,  
const int64_t *const values[]);
```

参数说明

参数名	输入/输出	说明
attr	输出	aclopAttr类型的指针。 需提前调用 aclopCreateAttr 接口创建aclopAttr类型。
attrName	输入	属性名的指针。
numLists	输入	列表数目。
numValues	输入	列表中属性值数目。 numValues是一个数组，数组中的每个元素表示每个列表中的属性值数目。
values	输入	列表中属性值的指针数组。

返回值说明

返回0表示成功，返回非0表示失败。

1.13.74.12 aclopSetAttrDataType

函数功能

设置指定属性的数据类型。

函数原型

```
aclError aclopSetAttrDataType(aclopAttr *attr, const char *attrName,  
aclDataType attrValue)
```

参数说明

参数名	输入/输出	说明
attr	输出	aclopAttr类型的指针。 需提前调用 aclopCreateAttr 接口创建aclopAttr类型。
attrName	输入	属性名的指针。
attrValue	输入	属性值。

返回值说明

返回0表示成功，返回非0表示失败。

1.13.74.13 aclopSetAttrListDataType

函数功能

设置指定属性列表的数据类型。

函数原型

aclError aclopSetAttrListDataType(**aclOpAttr** *attr, const char *attrName, int numValues, const **aclDataType** values[])

参数说明

参数名	输入/输出	说明
attr	输出	aclopAttr类型的指针。 需提前调用 aclOpCreateAttr 接口创建aclopAttr类型。
attrName	输入	属性名的指针。
numValues	输入	属性值数目。
values	输入	属性值的数组。

返回值说明

返回0表示成功，返回非0表示失败。

1.13.75 acldvppChannelDesc

1.13.75.1 acldvppCreateChannelDesc

函数功能

创建acldvppChannelDesc类型的数据，表示创建图片数据处理通道时的通道描述信息。

函数原型

acldvppChannelDesc *acldvppCreateChannelDesc()

参数说明

无

返回值说明

- 返回acldvppChannelDesc类型的指针，表示成功。
- 返回nullptr，表示失败。

1.13.75.2 acldvppDestroyChannelDesc

函数功能

销毁通过[acldvppCreateChannelDesc](#)接口创建的acldvppChannelDesc类型的数据。

函数原型

[aclError](#) acldvppDestroyChannelDesc([acldvppChannelDesc](#) *channelDesc)

参数说明

参数名	输入/输出	说明
channelDesc	输入	待销毁的acldvppChannelDesc类型指针。

返回值说明

返回0表示成功，返回其他值表示失败。

1.13.75.3 acldvppGetChannelDescChannelId

函数功能

根据通道描述信息获取通道ID。

约束说明

接口调用顺序: [acldvppCreateChannelDesc](#) --> [acldvppCreateChannel](#) --> [acldvppGetChannelDescChannelId](#)

函数原型

uint64_t acldvppGetChannelDescChannelId(const [acldvppChannelDesc](#) *channelDesc)

参数说明

参数名	输入/输出	说明
channelDesc	输入	通道描述信息的指针。 需提前调用 acldvppCreateChannelDesc 接口创建acldvppChannelDesc类型的数据。

返回值说明

返回通道ID。

1.13.75.4 acldvppSetChannelDescMode

函数功能

指定通道描述信息中的通道模式，明确图片数据处理通道用于实现哪种功能，目前支持[VPC](#)、[JPEGD](#)、[JPEGE](#)、[PNGD](#)功能。

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

约束说明

接口调用顺序：[acldvppCreateChannelDesc](#) --> [acldvppSetChannelDescMode](#) --> [acldvppCreateChannel](#)

函数原型

[aclError](#) acldvppSetChannelDescMode([acldvppChannelDesc](#) *channelDesc, uint32_t mode)

参数说明

参数名	输入/输出	说明
channelDesc	输出	通道描述信息的指针。 需提前调用 acldvppCreateChannelDesc 接口创建acldvppChannelDesc类型的数据。
mode	输入	指定通道模式，各模式的取值请参见 acldvppChannelMode 枚举值。

返回值说明

返回0表示成功，返回其他值表示失败。

1.13.75.5 aclvppSetChannelDescParam

函数功能

设置图片数据处理通道的属性参数接口。

函数原型

aclError aclvppSetChannelDescParam(**aclvppChannelDesc** *channelDesc, **aclvppChannelDescParamType** paramType, size_t length, const void *param)

参数说明

参数名	输入/输出	说明
channelDesc	输出	通道描述信息的指针。 需提前调用 aclvppCreateChannelDesc 接口创建aclvppChannelDesc类型的数据。
paramType	输入	属性参数的类型。
length	输入	属性参数值的字节数。 - 属性参数的类型为*_UINT64时，此处配置为8。 - 属性参数的类型为*_UINT32时，此处配置为4。 - 属性参数的类型为*_ENUM时，此处配置为4。
param	输入	属性参数值的指针。

返回值说明

返回0表示成功，返回其他值表示失败。

1.13.75.6 aclvppGetChannelDescParam

函数功能

获取图片数据处理通道的属性参数接口。

函数原型

aclError aclvppGetChannelDescParam(const **aclvppChannelDesc** *channelDesc, **aclvppChannelDescParamType** paramType, size_t length, size_t *paramRetSize, void *param)

参数说明

参数名	输入/输出	说明
channelDesc	输入	通道描述信息的指针。 需提前调用 acldvppCreateChannelDesc 接口创建acldvppChannelDesc类型的数据，调用 acldvppSetChannelDescParam 接口设置通道属性值。
paramType	输入	属性参数的类型。
length	输入	属性参数值的字节数。 - 属性参数的类型为*_UINT64时，此处配置为8。 - 属性参数的类型为*_UINT32时，此处配置为4。 - 属性参数的类型为*_ENUM时，此处配置为4。
paramRetSize	输出	实际返回的属性参数值字节数的指针。
param	输出	属性参数值的指针。

返回值说明

返回0表示成功，返回其他值表示失败。

1.13.76 acldvppPicDesc

1.13.76.1 acldvppCreatePicDesc

函数功能

创建acldvppPicDesc类型的数据，表示图片描述信息。
如需销毁acldvppPicDesc类型的数据，请参见[acldvppDestroyPicDesc](#)。

函数原型

[acldvppPicDesc](#) *acldvppCreatePicDesc()

参数说明

无

返回值说明

- 返回acldvppPicDesc类型的指针，表示成功。
- 返回nullptr，表示失败。

1.13.76.2 aclvppSetPicDesc 系列接口

函数功能

设置图片描述参数。

函数原型

```
aclError aclvppSetPicDescData(aclvppPicDesc *picDesc, void *dataDev);
aclError aclvppSetPicDescSize(aclvppPicDesc *picDesc, uint32_t size);
aclError aclvppSetPicDescFormat(aclvppPicDesc *picDesc,
aclvppPixelFormat format);
aclError aclvppSetPicDescWidth(aclvppPicDesc *picDesc, uint32_t width);
aclError aclvppSetPicDescHeight(aclvppPicDesc *picDesc, uint32_t height);
aclError aclvppSetPicDescWidthStride(aclvppPicDesc *picDesc, uint32_t
widthStride);
aclError aclvppSetPicDescHeightStride(aclvppPicDesc *picDesc, uint32_t
heightStride);
aclError aclvppSetPicDescRetCode(aclvppPicDesc *picDesc,uint32_t retCode)
```

参数说明

参数名	输入/输出	说明
picDesc	输出	描述图片信息的指针。 需提前调用 aclvppCreatePicDesc 接口创建aclvppPicDesc类型的数据。
dataDev	输入	Device上图片数据内存的指针，必须是使用 aclvppMalloc 接口申请的内存。
size	输入	内存大小，单位Byte。
format	输入	图片格式。
width	输入	原始图片宽。
height	输入	原始图片高。
widthStride	输入	对齐后的宽。
heightStride	输入	对齐后的高。
retCode	输入	返回码（0成功，非0失败）。

返回值说明

返回0表示成功，返回其他值表示失败。

1.13.76.3 aclvppGetPicDesc 系列接口

函数功能

获取图片描述信息。

函数原型

```
void *aclvppGetPicDescData(const aclvppPicDesc *picDesc);  
uint32_t aclvppGetPicDescSize(const aclvppPicDesc *picDesc);  
aclvppPixelFormat aclvppGetPicDescFormat(const aclvppPicDesc *picDesc);  
uint32_t aclvppGetPicDescWidth(const aclvppPicDesc *picDesc);  
uint32_t aclvppGetPicDescHeight(const aclvppPicDesc *picDesc);  
uint32_t aclvppGetPicDescWidthStride(const aclvppPicDesc *picDesc);  
uint32_t aclvppGetPicDescHeightStride(const aclvppPicDesc *picDesc);  
uint32_t aclvppGetPicDescRetCode(const aclvppPicDesc *picDesc);
```

参数说明

参数名	输入/输出	说明
picDesc	输入	描述图片信息的指针。 需提前调用 aclvppCreatePicDesc 接口创建aclvppPicDesc类型的数据，调用 aclvppSetPicDesc系列接口 设置图片描述参数。

返回值说明

参数名	说明
data	图片数据的内存地址。
size	图片数据的内存的大小。
format	图片格式。
width	原始图片宽。
height	原始图片高。
widthStride	对齐后的宽。
heightStride	对齐后的高。

参数名	说明
retCode	返回码（0成功，非0失败）。 VDEC解码时，用户需调用 acldvppGetPicDescRetCode 接口获取retCode返回码判断是否解码成功，如果解码失败，需要根据日志中的返回码判断具体的问题，返回码请参见返回码说明。retCode中部分返回码是以十六进制形式定义的，如用户在编码时以十进制打印返回码的值（例如，65540），则需提前转换为十六进制（例如，0x10004），然后在返回码说明中查询。

1.13.76.4 acldvppDestroyPicDesc

函数功能

销毁通过[acldvppCreatePicDesc](#)接口创建的acldvppPicDesc类型的数据。

函数原型

aclError acldvppDestroyPicDesc([acldvppPicDesc](#) *picDesc)

参数说明

参数名	输入/输出	说明
picDesc	输入	待销毁的acldvppPicDesc类型指针。

返回值说明

返回0表示成功，返回其他值表示失败。

1.13.77 acldvppRoiConfig

1.13.77.1 acldvppCreateRoiConfig

函数功能

创建acldvppRoiConfig类型的数据，用于描述某个区域位置的数据。
如需销毁acldvppRoiConfig类型的数据，请参见[acldvppDestroyRoiConfig](#)。

函数原型

acldvppRoiConfig *acldvppCreateRoiConfig(
uint32_t left,


```
uint32_t right,  
uint32_t top,  
uint32_t bottom)
```

参数说明

参数名	输入/输出	说明
left	输入	左偏移。 - 在Atlas 200/300/500 推理产品上，该参数值必须为偶数。如果用做贴图区域，左偏移需要16对齐 - 在Atlas 训练系列产品上，该参数值必须为偶数。如果用做贴图区域，左偏移需要16对齐 - 在Atlas 推理系列产品上，该参数值无偶数要求。如果用做贴图区域，左偏移需要16对齐 - 在Atlas A2训练系列产品/Atlas 800I A2推理产品上，该参数值无偶数要求、无对齐要求。
right	输入	右偏移。 - 在Atlas 200/300/500 推理产品上，该参数值必须为奇数。 - 在Atlas 训练系列产品上，该参数值必须为奇数。 - 在Atlas 推理系列产品上，该参数值无奇数要求。 - 在Atlas A2训练系列产品/Atlas 800I A2推理产品上，该参数值无奇数要求。
top	输入	上偏移。 - 在Atlas 200/300/500 推理产品上，该参数值必须为偶数。 - 在Atlas 训练系列产品上，该参数值必须为偶数。 - 在Atlas 推理系列产品上，该参数值无偶数要求。 - 在Atlas A2训练系列产品/Atlas 800I A2推理产品上，该参数值无偶数要求。
bottom	输入	下偏移。 - 在Atlas 200/300/500 推理产品上，该参数值必须为奇数。 - 在Atlas 训练系列产品上，该参数值必须为奇数。 - 在Atlas 推理系列产品上，该参数值无奇数要求。 - 在Atlas A2训练系列产品/Atlas 800I A2推理产品上，该参数值无奇数要求。

返回值说明

- 返回acldvppRoiConfig类型的指针，表示成功。
- 返回nullptr，表示失败。

1.13.77.2 acldvppSetRoiConfig 系列接口

函数功能

设置某个区域的位置信息。

函数原型

```
aclError acldvppSetRoiConfig(acldvppRoiConfig *config, uint32_t left, uint32_t right, uint32_t top, uint32_t bottom);  
  
aclError acldvppSetRoiConfigLeft(acldvppRoiConfig *config, uint32_t left);  
  
aclError acldvppSetRoiConfigRight(acldvppRoiConfig *config, uint32_t right);  
  
aclError acldvppSetRoiConfigTop(acldvppRoiConfig *config, uint32_t top);  
  
aclError acldvppSetRoiConfigBottom(acldvppRoiConfig *config, uint32_t bottom);
```

参数说明

参数名	输入/输出	说明
config	输出	区域位置数据的指针。 需提前调用 acldvppCreateRoiConfig 接口创建acldvppRoiConfig类型的数据。
left	输入	左偏移。 • 在Atlas 200/300/500 推理产品上，该参数值必须为偶数。如果用做贴图区域，左偏移需要16对齐 • 在Atlas 训练系列产品上，该参数值必须为偶数。如果用做贴图区域，左偏移需要16对齐 • 在Atlas 推理系列产品上，该参数值无偶数要求。如果用做贴图区域，左偏移需要16对齐 • 在Atlas A2训练系列产品/Atlas 800I A2推理产品上，该参数值无偶数要求、无对齐要求。

参数名	输入/输出	说明
right	输入	右偏移。 - 在Atlas 200/300/500 推理产品上，该参数值必须为奇数。 - 在Atlas 训练系列产品上，该参数值必须为奇数。 - 在Atlas 推理系列产品上，该参数值无奇数要求。 - 在Atlas A2训练系列产品/Atlas 800I A2推理产品上，该参数值无奇数要求。
top	输入	上偏移。 - 在Atlas 200/300/500 推理产品上，该参数值必须为偶数。 - 在Atlas 训练系列产品上，该参数值必须为偶数。 - 在Atlas 推理系列产品上，该参数值无偶数要求。 - 在Atlas A2训练系列产品/Atlas 800I A2推理产品上，该参数值无偶数要求。
bottom	输入	下偏移。 - 在Atlas 200/300/500 推理产品上，该参数值必须为奇数。 - 在Atlas 训练系列产品上，该参数值必须为奇数。 - 在Atlas 推理系列产品上，该参数值无奇数要求。 - 在Atlas A2训练系列产品/Atlas 800I A2推理产品上，该参数值无奇数要求。

返回值说明

返回0表示成功，返回其他值表示失败。

1.13.77.3 acldvppDestroyRoiConfig

函数功能

销毁通过[acldvppCreateRoiConfig](#)接口创建的acldvppRoiConfig类型的数据。

函数原型

`aclError acldvppDestroyRoiConfig(acldvppRoiConfig *roiConfig)`

参数说明

参数名	输入/输出	说明
roiConfig	输入	待销毁的acldvppRoiConfig类型的指针。

返回值说明

返回0表示成功，返回其他值表示失败。

1.13.78 acldvppResizeConfig

1.13.78.1 acldvppCreateResizeConfig

函数功能

创建acldvppResizeConfig类型的数据，表示图片缩放配置数据。

如需销毁acldvppResizeConfig类型的数据，请参见[acldvppDestroyResizeConfig](#)。

Atlas 200/300/500 推理产品，默认缩放算法为“自研的高阶滤波算法”。

Atlas 训练系列产品，默认缩放算法为“自研的高阶滤波算法”。

Atlas 推理系列产品，默认缩放算法为“业界通用的Bilinear算法”。

Atlas A2训练系列产品/Atlas 800I A2推理产品，默认缩放算法为“业界通用的Bilinear算法”。

Atlas 200/500 A2推理产品，默认缩放算法为“业界通用的Bilinear算法”。

函数原型

[acldvppResizeConfig](#) *acldvppCreateResizeConfig()

参数说明

无

返回值说明

- 返回acldvppResizeConfig类型的指针，表示成功。
- 返回nullptr，表示失败。

1.13.78.2 acldvppSetResizeConfigInterpolation

函数功能

设置图片缩放算法。

函数原型

aclError **acldvppSetResizeConfigInterpolation**(**acldvppResizeConfig**
***resizeConfig**, **uint32_t** interpolation)

参数说明

参数名	输入/输出	说明
resizeConfig	输出	待设置的缩放配置数据的指针。 需提前调用 acldvppCreateResizeConfig 接口创建 acldvppResizeConfig类型的数据。

参数名	输入/输出	说明
interpolation	输入	指定缩放算法。此处配置的缩放算法建议与训练模型时的缩放算法保持一致。 Atlas 200/300/500 推理产品上，支持如下缩放算法： 0：默认值，表示自研的高阶滤波算法。 1：业界通用的Bilinear算法（与OpenCV算法的计算过程类似） 2：业界通用的Nearest neighbor 算法（与OpenCV算法的计算过程类似） 3：业界通用的Bilinear算法（与Tensorflow算法的计算过程类似） 4：业界通用的Nearest neighbor算法（与Tensorflow算法的计算过程类似） Atlas 训练系列产品上，支持如下缩放算法： 0：默认值，表示自研的高阶滤波算法。 1：业界通用的Bilinear算法（与OpenCV算法的计算过程类似） 2：业界通用的Nearest neighbor 算法（与OpenCV算法的计算过程类似） 3：业界通用的Bilinear算法（与Tensorflow算法的计算过程类似） 4：业界通用的Nearest neighbor算法（与Tensorflow算法的计算过程类似） Atlas 推理系列产品,上，支持如下缩放算法： 0：设置为0时，系统内部也会自动采用1。 1：业界通用的Bilinear算法（与OpenCV算法的计算过程类似）。 设置该算法，且当输入和输出图片格式都为RGB时，在[1/32, 512]的缩放范围内，与OpenCV的单个像素值最大差异为正负1。 2：业界通用的Nearest neighbor 算法（与OpenCV算法的计算过程类似） Atlas 200/500 A2推理产品，支持如下缩放算法： 0：设置为0时，系统内部也会自动采用1。 1：业界通用的Bilinear算法（与OpenCV-3.4.2版本算法的计算结果相同） 2：业界通用的Nearest neighbor 算法（与OpenCV-3.4.2版本算法的计算结果相同） 3：业界通用的Bilinear算法（与1相同） 4：业界通用的Nearest neighbor算法（与2相同） 5：自研的高阶滤波算法

参数名	输入/输出	说明
		当前该缩放算法仅支持输入图片分辨率范围为10*6~8192*8192、输出图片分辨率的范围为10*6~4096*8192；输入图片格式仅支持YUV、但不包括Planner格式、也不包括YUV400、YUV440，输出图片格式仅支持YUV420SP NV12 8bit、YUV420SP NV21 8bit；宽或高的缩放范围：[1/32, 16]。 Atlas A2训练系列产品/Atlas 800I A2推理产品，支持如下缩放算法： • 0：设置为0时，系统内部也会自动采用1。 • 1：业界通用的Bilinear算法（与OpenCV-3.4.2版本算法的计算结果相同） • 2：业界通用的Nearest neighbor 算法（与OpenCV-3.4.2版本算法的计算结果相同） • 3：业界通用的Bilinear算法（与1相同） • 4：业界通用的Nearest neighbor算法（与2相同） • 5：自研的高阶滤波算法 当前该缩放算法仅支持输入图片分辨率范围为10*6~8192*8192、输出图片分辨率的范围为10*6~4096*8192；输入图片格式仅支持YUV、但不包括Planner格式、也不包括YUV400、YUV440，输出图片格式仅支持YUV420SP NV12 8bit、YUV420SP NV21 8bit；宽或高的缩放范围：[1/32, 16]。

返回值说明

返回0表示成功，返回其他值表示失败。

1.13.78.3 acldvppGetResizeConfigInterpolation

函数功能

获取图片缩放算法。

函数原型

```
uint32 acldvppGetResizeConfigInterpolation(const acldvppResizeConfig *
resizeConfig)
```

参数说明

参数名	输入/输出	说明
resizeConfig	输入	待获取的缩放配置数据的指针。 需提前调用 acldvppCreateResizeConfig 接口创建acldvppResizeConfig类型的数据，调用 acldvppSetResizeConfigInterpolation 接口设置缩放算法。

返回值说明

返回0表示成功，返回其他值表示失败。

1.13.78.4 acldvppDestroyResizeConfig

函数功能

销毁通过[acldvppCreateResizeConfig](#)接口创建的acldvppResizeConfig类型的数据。

函数原型

aclError acldvppDestroyResizeConfig(**acldvppResizeConfig** *resizeConfig)

参数说明

参数名	输入/输出	说明
resizeConfig	输入	待销毁的acldvppResizeConfig类型的指针。

返回值说明

返回0表示成功，返回其他值表示失败。

1.13.79 acldvppJpegeConfig

1.13.79.1 acldvppCreateJpegeConfig

函数功能

创建acldvppJpegeConfig类型的数据，表示图片编码配置数据。
如需销毁acldvppJpegeConfig类型的数据，请参见[acldvppDestroyJpegeConfig](#)。

函数原型

```
acldvppJpegeConfig *acldvppCreateJpegeConfig()
```

参数说明

无

返回值说明

- 返回acldvppJpegeConfig类型的指针，表示成功。
- 返回nullptr，表示失败。

1.13.79.2 acldvppSetJpegeConfigLevel

函数功能

设置编码配置数据。

函数原型

```
aclError acldvppSetJpegeConfigLevel(acldvppJpegeConfig *jpegeConfig,
uint32_t level)
```

参数说明

参数名	输入/输出	说明
jpegeConfig	输出	编码配置数据的指针。 需提前调用acldvppCreateJpegeConfig接口创建acldvppJpegeConfig类型的数据。
level	输入	编码质量范围。 Atlas 200/300/500 推理产品，该参数取值范围：[0, 100]，其中level 0编码质量与level 100差不多，而在[1, 100]内数值越小输出图片质量越差。 Atlas 训练系列产品，该参数取值范围：[0, 100]，其中level 0编码质量与level 100差不多，而在[1, 100]内数值越小输出图片质量越差。 Atlas 推理系列产品，该参数取值范围：[1, 100]，数值越小输出图片质量越差。 Atlas 200/500 A2推理产品，该参数取值范围：[1, 100]，数值越小输出图片质量越差。 Atlas A2训练系列产品/Atlas 800I A2推理产品，该参数取值范围：[1, 100]，数值越小输出图片质量越差。

返回值说明

返回0表示成功，返回其他值表示失败。

1.13.79.3 acldvppGetJpegeConfigLevel

函数功能

获取编码配置数据。

函数原型

uint32_t acldvppGetJpegeConfigLevel(const [acldvppJpegeConfig](#) *jpegeConfig)

参数说明

参数名	输入/输出	说明
jpegeConfig	输入	编码配置数据的指针。 需提前调用 acldvppCreateJpegeConfig 接口创建 acldvppJpegeConfig 类型的数据，再调用 acldvppSetJpegeConfigLevel 接口设置编码配置数据。

返回值说明

返回编码质量范围。

Atlas 200/300/500 推理产品，该参数取值范围：[0, 100]，其中level 0编码质量与level 100差不多，而在[1, 100]内数值越小输出图片质量越差。

Atlas 训练系列产品，该参数取值范围：[0, 100]，其中level 0编码质量与level 100差不多，而在[1, 100]内数值越小输出图片质量越差。

Atlas 推理系列产品，该参数取值范围：[1, 100]，数值越小输出图片质量越差。

Atlas 200/500 A2推理产品，该参数取值范围：[1, 100]，数值越小输出图片质量越差。

Atlas A2训练系列产品/Atlas 800I A2推理产品，该参数取值范围：[1, 100]，数值越小输出图片质量越差。

1.13.79.4 acldvppDestroyJpegeConfig

函数功能

销毁通过[acldvppCreateJpegeConfig](#)接口创建的acldvppJpegeConfig类型的数据。

函数原型

```
aclError aclvppDestroyJpegeConfig(aclvppJpegeConfig *jpegeConfig)
```

参数说明

参数名	输入/输出	说明
jpegeConfig	输入	待销毁的aclvppJpegeConfig类型的指针。

返回值说明

返回0表示成功，返回其他值表示失败。

1.13.80 aclvdecChannelDesc

1.13.80.1 aclvdecCreateChannelDesc

函数功能

创建aclvdecChannelDesc类型的数据，表示创建视频解码处理通道时的通道描述信息。

如需销毁aclvdecChannelDesc类型的数据，请参见[aclvdecDestroyChannelDesc](#)。

函数原型

```
aclvdecChannelDesc *aclvdecCreateChannelDesc()
```

参数说明

无

返回值说明

- 返回aclvdecChannelDesc类型的指针，表示成功。
- 返回nullptr，表示失败。

1.13.80.2 aclvdecSetChannelDesc 系列接口

函数功能

设置视频解码处理通道描述信息的属性值。

函数原型

```
aclError aclvdecSetChannelDescChannelId(aclvdecChannelDesc *channelDesc, uint32_t channelId);
```

```
aclError aclvdecSetChannelDescThreadId(aclvdecChannelDesc *channelDesc,  
uint64_t threadId);
```

```
aclError aclvdecSetChannelDescCallback(aclvdecChannelDesc *channelDesc,  
aclvdecCallback callback);
```

```
aclError aclvdecSetChannelDescEnType(aclvdecChannelDesc *channelDesc,  
aclvppStreamFormat enType);
```

```
aclError aclvdecSetChannelDescOutPicFormat(aclvdecChannelDesc  
*channelDesc, aclvppPixelFormat outPicFormat);
```

```
aclError aclvdecSetChannelDescOutPicWidth(aclvdecChannelDesc  
*channelDesc, uint32_t outPicWidth);
```

```
aclError aclvdecSetChannelDescOutPicHeight(aclvdecChannelDesc  
*channelDesc, uint32_t outPicHeight);
```

```
aclError aclvdecSetChannelDescRefFrameNum(aclvdecChannelDesc  
*channelDesc, uint32_t refFrameNum);
```

```
aclError aclvdecSetChannelDescOutMode(aclvdecChannelDesc *channelDesc,  
uint32_t outMode)
```

```
aclError aclvdecSetChannelDescBitDepth(aclvdecChannelDesc *channelDesc,  
uint32_t bitDepth)
```

参数说明

参数名	输入/输出	说明
channelDesc	输出	视频解码处理通道描述信息的指针。 需提前调用 aclvdecCreateChannelDesc 接口创建 aclvdecChannelDesc 类型的数据。
channelId	输入	通道ID。 Atlas 200/300/500 推理产品，该参数值的取值范围[0, 31]。 Atlas 训练系列产品，该参数值的取值范围[0, 31]。 Atlas 推理系列产品，该参数值的取值范围[0, 255]。 Atlas 200/500 A2推理产品，该参数值的取值范围[0, 255]。 Atlas A2训练系列产品/Atlas 800I A2推理产品，该参数值的取值范围[0, 255]。
threadId	输入	回调线程ID。 说明 同一个进程内，在不同的Device上注册VDEC解码回调函数的线程时，不能指定同一个线程ID。 同一个Device上，多路VDEC解码可以指定同一个线程ID，但相比一个线程处理一路VDEC解码任务来说，一个线程内串行处理多路VDEC解码任务时，VDEC解码性能可能下降。

参数名	输入/输出	说明
callback	输入	解码回调函数。 请参见 1.8.9.5 aclvdecCallback 。
enType	输入	视频编码协议H265-main level (0)、H264-baseline level (1)、H264-main level(2)、H264-high level (3)。
outPicFormat	输入	输出图像存储格式，具体支持哪些格式请参见 码流/图片格式、宽高对齐、内存约束 。 如果不设置输出格式，默认使用YUV420SP NV12。
outPicWidth	输入	解码码流最大宽度。 Atlas 200/300/500 推理产品，如果不设置解码码流最大宽度，内部默认使用1920。 Atlas 训练系列产品，如果不设置解码码流最大宽度，内部默认使用1920。 Atlas 推理系列产品，如果不设置解码码流最大宽度，内部默认使用4096。 Atlas 200/500 A2推理产品，如果不设置解码码流最大宽度，内部默认使用4096。 Atlas A2训练系列产品/Atlas 800I A2推理产品，如果不设置解码码流最大宽度，内部默认使用4096。
outPicHeight	输入	解码码流最大高度。 Atlas 200/300/500 推理产品，如果不设置解码码流最大高度，内部默认使用1120。 Atlas 训练系列产品，如果不设置解码码流最大高度，内部默认使用1120。 Atlas 推理系列产品，如果不设置解码码流最大高度，内部默认使用4096。 Atlas 200/500 A2推理产品，如果不设置解码码流最大高度，内部默认使用4096。 Atlas A2训练系列产品/Atlas 800I A2推理产品，如果不设置解码码流最大高度，内部默认使用4096。
refFrameNum	输入	参考帧数量，取值范围[0, 16]。 Atlas 200/300/500 推理产品，当前版本不支持用户设置该参数。 Atlas 训练系列产品，当前版本不支持用户设置该参数。 Atlas 推理系列产品，如果不设置参考帧数量，内部默认使用8。 Atlas 200/500 A2推理产品，如果不设置参考帧数量，内部默认使用8。 Atlas A2训练系列产品/Atlas 800I A2推理产品，如果不设置参考帧数量，内部默认使用8。

参数名	输入/输出	说明
outMode	输入	设置是否实时出帧（即发送一帧解码一帧，无需依赖后续帧的传入）。 取值范围如下： 0：默认出帧模式，由于解码过程中存在缓存帧，无法实时输出，因此VDEC需要在收到码流中的多帧数据后，才开始输出解码结果。 1：快速出帧模式，VDEC获取码流中的一帧数据后，就开始实时输出解码结果。只支持简单参考关系的H264/H265标准码流（无长期参考帧，无B帧）。
bitDepth	输入	设置视频位宽，默认值为10-bit。 取值范围如下： 0：8-bit 1：10-bit(默认值) Atlas 200/300/500 推理产品，当前版本不支持用户设置该参数。 Atlas 训练系列产品，当前版本不支持用户设置该参数。

返回值说明

返回0表示成功，返回其他值表示失败。

1.13.80.3 aclvdecGetChannelDesc 系列接口

函数功能

获取视频解码处理通道的描述信息。

函数原型

```
uint32_t aclvdecGetChannelDescChannelId(const aclvdecChannelDesc
*channelDesc);

uint32_t aclvdecGetChannelDescThreadId(const aclvdecChannelDesc
*channelDesc);

aclvdecCallback aclvdecGetChannelDescCallback(const aclvdecChannelDesc
*channelDesc);

aclvppStreamFormat aclvdecGetChannelDescEnType(const
aclvdecChannelDesc *channelDesc);

aclvppPixelFormat aclvdecGetChannelDescOutPicFormat(const
aclvdecChannelDesc *channelDesc);

uint32_t aclvdecGetChannelDescOutPicWidth(const aclvdecChannelDesc
*channelDesc);
```

```
uint32_t aclvdecGetChannelDescOutPicHeight(const aclvdecChannelDesc
*channelDesc);

uint32_t aclvdecGetChannelDescRefFrameNum(const aclvdecChannelDesc
*channelDesc);

uint32_t aclvdecGetChannelDescOutMode(const aclvdecChannelDesc
*channelDesc)

uint32_t aclvdecGetChannelDescBitDepth(const aclvdecChannelDesc
*channelDesc)
```

参数说明

参数名	输入/输出	说明
channelDesc	输入	视频解码数据处理通道描述信息的指针。 需提前调用 aclvdecCreateChannelDesc 接口创建 aclvdecChannelDesc 类型的数据，调用 aclvdecSetChannelDesc 系列接口设置视频解码处理通道描述信息的属性值。

返回值说明

参数名	说明
channelId	解码通道号。 Atlas 200/300/500 推理产品，该参数值的取值范围[0, 31]。 Atlas 训练系列产品，该参数值的取值范围[0, 31]。 Atlas 推理系列产品，该参数值的取值范围[0, 255]。 Atlas 200/500 A2推理产品，该参数值的取值范围[0, 255]。 Atlas A2训练系列产品/Atlas 800I A2推理产品，该参数值的取值范围[0, 255]。
threadId	回调线程ID。
callback	回调函数。 请参见 1.8.9.5 aclvdecCallback 。
enType	视频编码协议H265-main level (0)、H264-baseline level (1)、H264-main level(2)、H264-high level (3)。
outPicFormat	YUV图像存储格式。
outPicWidth	图片宽度。
outPicHeight	图片高度。

参数名	说明
refFrameNum	参考帧数量，取值范围[0, 16]。 Atlas 200/300/500 推理产品，当前版本获取该参数的值为无效值，不支持获取该参数值。 Atlas 训练系列产品，当前版本获取该参数的值为无效值，不支持获取该参数值。
outMode	是否实时出帧（即发送一帧解码一帧，无需依赖后续帧的传入），取值范围如下： 0：默认出帧模式，由于解码过程中存在缓存帧，无法实时输出，因此VDEC需要在收到码流中的多帧数据后，才开始输出解码结果。 1：快速出帧模式，VDEC获取码流中的一帧数据后，就开始实时输出解码结果。只支持简单参考关系的H264/H265标准码流（无长期参考帧，无B帧）。
bitDepth	视频位宽。 取值范围如下： 0：8-bit 1：10-bit Atlas 200/300/500 推理产品，当前版本获取该参数的值为无效值，不支持获取该参数值。 Atlas 训练系列产品，当前版本获取该参数的值为无效值，不支持获取该参数值。

1.13.80.4 aclvdecSetChannelDescParam

函数功能

设置视频编码处理通道描述信息的属性值。

函数原型

aclError **aclvdecSetChannelDescParam**(**aclvdecChannelDesc** *channelDesc, **aclvdecChannelDescParamType** paramType, size_t length, const void *param)

参数说明

参数名	输入/输出	说明
channelDesc	输出	视频解码处理通道描述信息的指针。 需提前调用 aclvdecCreateChannelDesc 接口创建aclvdecChannelDesc类型的数据。
paramType	输入	属性参数的类型。

参数名	输入/输出	说明
length	输入	属性参数值的字节数。 - 属性参数的类型为*_UINT64时，此处配置为8。 - 属性参数的类型为*_UINT32时，此处配置为4。 - 属性参数的类型为*_PTR时，若操作系统是32位，则此处配置为4；若操作系统是64位，则配置为8。
param	输入	属性参数值的指针。

返回值说明

返回0表示成功，返回其他值表示失败。

1.13.80.5 aclvdecGetChannelDescParam

函数功能

获取视频解码处理通道的描述信息。

函数原型

aclError **aclvdecGetChannelDescParam**(const **aclvdecChannelDesc** *channelDesc, **aclvdecChannelDescParamType** paramType, size_t length, size_t *paramRetSize, void *param)

参数说明

参数名	输入/输出	说明
channelDesc	输入	视频解码数据处理通道描述信息的指针。 需提前调用 aclvdecCreateChannelDesc 接口创建aclvdecChannelDesc类型的数据，调用 aclvdecSetChannelDescParam 接口设置视频解码处理通道描述信息的属性值。
paramType	输入	属性参数的类型。
length	输入	属性参数值的字节数。 - 属性参数的类型为*_UINT64时，此处配置为8。 - 属性参数的类型为*_UINT32时，此处配置为4。 - 属性参数的类型为*_PTR时，若操作系统是32位，则此处配置为4；若操作系统是64位，则配置为8。

参数名	输入/输出	说明
paramRetSize	输出	实际返回的属性参数值字节数的指针。
param	输出	属性参数值的指针。

返回值说明

返回0表示成功，返回其他值表示失败。

1.13.80.6 aclvdecDestroyChannelDesc

函数功能

销毁通过[aclvdecCreateChannelDesc](#)接口创建的aclvdecChannelDesc类型的数据。

函数原型

aclError aclvdecDestroyChannelDesc([aclvdecChannelDesc](#) *channelDesc)

参数说明

参数名	输入/输出	说明
channelDesc	输入	待销毁的aclvdecChannelDesc类型的指针。

返回值说明

返回0表示成功，返回非0表示失败。

1.13.81 acldvppStreamDesc

1.13.81.1 acldvppCreateStreamDesc

函数功能

创建acldvppStreamDesc类型的数据，表示视频码流描述信息。
如需销毁acldvppStreamDesc类型的数据，请参见[acldvppDestroyStreamDesc](#)。

函数原型

acldvppStreamDesc *acldvppCreateStreamDesc()

参数说明

无

返回值说明

- 返回acldvppStreamDesc类型的指针，表示成功。
- 返回nullptr，表示失败。

1.13.81.2 acldvppSetStreamDesc 系列接口

函数功能

设置视频码流信息的属性值。

函数原型

```
aclError acldvppSetStreamDescData(acldvppStreamDesc *streamDesc, void *dataDev);

aclError acldvppSetStreamDescSize(acldvppStreamDesc *streamDesc, uint32_t size);

aclError acldvppSetStreamDescFormat(acldvppStreamDesc *streamDesc, acldvppStreamFormat format);

aclError acldvppSetStreamDescTimestamp(acldvppStreamDesc *streamDesc, uint64_t timestamp);

aclError acldvppSetStreamDescRetCode(acldvppStreamDesc *streamDesc, uint32_t retCode);

aclError acldvppSetStreamDescEos(acldvppStreamDesc *streamDesc, uint8_t eos)
```

参数说明

参数名	输入/输出	说明
streamDesc	输出	要设置的视频码流信息的指针。 需提前调用 acldvppCreateStreamDesc 接口创建acldvppStreamDesc类型的数据。
dataDev	输入	Device上码流数据内存的指针。
size	输入	内存大小，单位Byte。
format	输入	码流格式（ h264/h265 ）。
timestamp	输入	时间戳。
retCode	输入	返回码（0成功，非0失败）。
eos	输入	是否为最后一帧数据（ 0否，1是 ）。

返回值说明

返回0表示成功，返回其他值表示失败。

1.13.81.3 acldvppGetStreamDesc 系列接口

函数功能

获取视频码流信息的属性。

函数原型

```
void *acldvppGetStreamDescData(const acldvppStreamDesc *streamDesc);
uint32_t acldvppGetStreamDescSize(const acldvppStreamDesc *streamDesc);
acldvppStreamFormat acldvppGetStreamDescFormat(const acldvppStreamDesc *streamDesc);
uint64_t acldvppGetStreamDescTimestamp(const acldvppStreamDesc *streamDesc);
uint32_t acldvppGetStreamDescRetCode(const acldvppStreamDesc *streamDesc);
uint8_t acldvppGetStreamDescEos(const acldvppStreamDesc *streamDesc)
```

参数说明

参数名	输入/输出	说明
streamDesc	输入	要设置的视频码流信息的指针。 需提前调用acldvppCreateStreamDesc接口创建acldvppStreamDesc类型的数据，调用acldvppSetStreamDesc系列接口设置视频码流信息的属性值。

返回值说明

参数名	说明
data	Device上码流数据内存。
size	内存大小，单位Byte。
format	码流格式（ h264/h265 ）。
timestamp	时间戳。

参数名	说明
retCode	返回码 (0成功, 非0失败)。
eos	是否为最后一帧数据。

1.13.81.4 acldvppDestroyStreamDesc

函数功能

销毁通过[acldvppCreateStreamDesc](#)接口创建的acldvppStreamDesc类型的数据。

函数原型

aclError acldvppDestroyStreamDesc(**acldvppStreamDesc** *streamDesc)

参数说明

参数名	输入/输出	说明
streamDesc	输入	待销毁的acldvppStreamDesc类型的指针。

返回值说明

返回0表示成功, 返回非0表示失败。

1.13.82 aclvdecFrameConfig

1.13.82.1 aclvdecCreateFrameConfig

函数功能

创建aclvdecFrameConfig类型的数据, 表示创建VDEC解码时的单帧解码配置参数。
如需销毁aclvdecFrameConfig类型的数据, 请参见[aclvdecDestroyFrameConfig](#)。

函数原型

aclvdecFrameConfig *aclvdecCreateFrameConfig()

参数说明

无

返回值说明

- 返回[aclvdecFrameConfig](#)类型的指针，表示成功。
- 返回nullptr，表示失败。

1.13.82.2 [aclvdecDestroyFrameConfig](#)

函数功能

销毁通过[aclvdecCreateFrameConfig](#)接口创建的[aclvdecFrameConfig](#)类型的数据。

函数原型

[aclError](#) [aclvdecDestroyFrameConfig](#)([aclvdecFrameConfig](#) *vdecFrameConfig)

参数说明

参数名	输入/输出	说明
vdecFrameConfig	输入	待销毁的 aclvdecFrameConfig 类型的指针。

返回值说明

返回0表示成功，返回非0表示失败。

1.13.83 [acldvppBatchPicDesc](#)

1.13.83.1 [acldvppCreateBatchPicDesc](#)

函数功能

创建[acldvppBatchPicDesc](#)类型的数据，表示批量图片描述信息。
如需销毁[acldvppBatchPicDesc](#)类型的数据，请参见[acldvppDestroyBatchPicDesc](#)。

函数原型

[acldvppBatchPicDesc](#) *[acldvppCreateBatchPicDesc](#)(uint32_t batchSize)

参数说明

参数名	输入/输出	说明
batchSize	输入	图片数量。

返回值说明

- 返回acldvppBatchPicDesc类型的指针，表示成功。
- 返回nullptr，表示失败。

1.13.83.2 acldvppGetPicDesc

函数功能

获取指定图片的图片描述信息。

函数原型

acldvppPicDesc *acldvppGetPicDesc(**acldvppBatchPicDesc** *batchPicDesc, uint32_t index)

参数说明

参数名	输入/输出	说明
batchPicDesc	输入	批量图片描述信息的指针。 需提前调用 acldvppCreateBatchPicDesc 接口创建acldvppBatchPicDesc类型的数据。
index	输入	指定获取第几个图片的描述信息，index值从0开始。

返回值说明

返回指定图片的图片描述信息。

1.13.83.3 acldvppDestroyBatchPicDesc

函数功能

销毁通过**acldvppCreateBatchPicDesc**接口创建的acldvppBatchPicDesc类型的数据。

函数原型

aclError acldvppDestroyBatchPicDesc(**acldvppBatchPicDesc** *batchPicDesc)

参数说明

参数名	输入/输出	说明
batchPicDesc	输入	待销毁的acldvppBatchPicDesc类型的指针。

返回值说明

返回0表示成功，返回非0表示失败。

1.13.84 aclvencChannelDesc

1.13.84.1 aclvencCreateChannelDesc

函数功能

创建[aclvencChannelDesc](#)类型的数据，表示创建视频编码处理通道时的通道描述信息。

如需销毁[aclvencChannelDesc](#)类型的数据，请参见[aclvencDestroyChannelDesc](#)。

Atlas 训练系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数原型

```
aclvencChannelDesc *aclvencCreateChannelDesc()
```

参数说明

无

返回值说明

- 返回[aclvencChannelDesc](#)类型的指针，表示成功。
- 返回nullptr，表示失败。

1.13.84.2 aclvencSetChannelDesc 系列接口

函数功能

设置视频编码处理通道描述信息的属性值。

Atlas 训练系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数原型

```
aclError aclvencSetChannelDescThreadId(aclvencChannelDesc *channelDesc,  
uint64_t threadId);
```

```
aclError aclvencSetChannelDescCallback(aclvencChannelDesc *channelDesc,  
aclvencCallback callback);
```

```
aclError aclvencSetChannelDescEnType(aclvencChannelDesc *channelDesc,  
aclvppStreamFormat enType);
```



```
aclError aclvencSetChannelDescPicFormat(aclvencChannelDesc *channelDesc,  
aclvppPixelFormat picFormat);
```

```
aclError aclvencSetChannelDescPicWidth(aclvencChannelDesc *channelDesc,  
uint32_t picWidth);
```

```
aclError aclvencSetChannelDescPicHeight(aclvencChannelDesc *channelDesc,  
uint32_t picHeight);
```

```
aclError aclvencSetChannelDescKeyFrameInterval(aclvencChannelDesc  
*channelDesc, uint32_t keyFrameInterval);
```

```
aclError aclvencSetChannelDescBufAddr(aclvencChannelDesc *channelDesc,  
void *bufAddr);
```

```
aclError aclvencSetChannelDescBufSize(aclvencChannelDesc *channelDesc,  
uint32_t bufSize);
```

```
aclError aclvencSetChannelDescRcMode(aclvencChannelDesc *channelDesc,  
uint32_t rcMode);
```

```
aclError aclvencSetChannelDescSrcRate(aclvencChannelDesc *channelDesc,  
uint32_t srcRate);
```

```
aclError aclvencSetChannelDescMaxBitRate(aclvencChannelDesc  
*channelDesc, uint32_t maxBitRate)
```

参数说明

参数名	输入/输出	说明
channelDesc	输出	视频编码处理通道描述信息的指针。 需提前调用 aclvencCreateChannelDesc 接口创建 aclvencChannelDesc 类型的数据。
threadId	输入	回调线程ID。 说明 同一个进程内，在不同的Device上注册VENC编码回调函数的线程时，不能指定同一个线程ID。
callback	输入	编码回调函数。
enType	输入	视频编码协议。
picFormat	输入	输入图像格式，支持如下格式： - PIXEL_FORMAT_YUV_SEMIPLANAR_420 - PIXEL_FORMAT_YVU_SEMIPLANAR_420
picWidth	输入	图片宽度。
picHeight	输入	图片高度。
keyFrameInterval	输入	关键帧间隔，取值范围[1,65536]。

参数名	输入/输出	说明
bufAddr	输入	编码输出缓存指针。 说明 在Atlas 200/300/500 推理产品上，当前版本不支持用户设置该参数，由VENC内部管理该缓存地址。
bufSize	输入	编码输出缓存大小，单位为Byte。 说明 在Atlas 200/300/500 推理产品上，当前版本不支持用户设置该参数，内部默认使用3686400 Byte。 在Atlas 推理系列产品上，如果不设置该参数，参数值默认为8M；如果设置该参数，参数值最小为5M。 Atlas 200/500 A2推理产品，如果不设置该参数，参数值默认为8M；如果设置该参数，参数值最小为5M。
rcMode	输入	指定码率控制模式。 0表示使用默认值 1表示变码率VBR模式 2表示定码率CBR模式 说明 如果不设置该参数，则采用默认值0。 Atlas 200/300/500 推理产品，默认值0表示CBR模式。 Atlas 推理系列产品，默认值0表示VBR模式。 Atlas 200/500 A2推理产品，默认值0表示VBR模式。
srcRate	输入	输入码流帧率，单位fps。 Atlas 200/300/500 推理产品，取值范围0或者[1,120]。 Atlas 推理系列产品，取值范围0或者[1,240]。 Atlas 200/500 A2推理产品，取值范围0或者[1,240]。 如果不设置该参数，默认为30；如果设置为0，表示使用默认值，即30。如果该值和实际输入码流帧率相差太大，会影响输出码率。
maxBitRate	输入	输出码率，单位kbps。 Atlas 200/300/500 推理产品，取值范围0或[10,30000]。如果不设置该参数，默认为300；如果设置为0，表示使用默认值，即300。 Atlas 推理系列产品，取值范围[2,614400]，如果不设置该参数，默认为2000；如果设置为0，表示使用默认值，即2000。 Atlas 200/500 A2推理产品，取值范围[2,614400]，如果不设置该参数，默认为2000；如果设置为0，表示使用默认值，即2000。

返回值说明

返回0表示成功，返回其他值表示失败。

1.13.84.3 aclvencGetChannelDesc 系列接口

函数功能

获取视频编码处理通道的描述信息。

Atlas 训练系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数原型

```
uint32_t aclvencGetChannelDescChannelId(const aclvencChannelDesc
*channelDesc);

uint64_t aclvencGetChannelDescThreadId(const aclvencChannelDesc
*channelDesc);

aclvencCallback aclvencGetChannelDescCallback(const aclvencChannelDesc
*channelDesc);

acldvppStreamFormat aclvencGetChannelDescEnType(const
aclvencChannelDesc *channelDesc);

acldvppPixelFormat aclvencGetChannelDescPicFormat(const
aclvencChannelDesc *channelDesc);

uint32_t aclvencGetChannelDescPicWidth(const aclvencChannelDesc
*channelDesc);

uint32_t aclvencGetChannelDescPicHeight(const aclvencChannelDesc
*channelDesc);

uint32_t aclvencGetChannelDescKeyFrameInterval(const aclvencChannelDesc
*channelDesc);

void *aclvencGetChannelDescBufAddr(const aclvencChannelDesc
*channelDesc);

uint32_t aclvencGetChannelDescBufSize(const aclvencChannelDesc
*channelDesc);

uint32_t aclvencGetChannelDescRcMode(const aclvencChannelDesc
*channelDesc);

uint32_t aclvencGetChannelDescSrcRate(const aclvencChannelDesc
*channelDesc);

uint32_t aclvencGetChannelDescMaxBitRate(const aclvencChannelDesc
*channelDesc)
```

参数说明

参数名	输入/输出	说明
channelDesc	输入	视频编码处理通道描述信息的指针。 需提前调用 aclvncCreateChannelDesc 接口创建 aclvncChannelDesc 类型的数据，调用 aclvncSetChannelDesc 系列接口设置视频编码处理通道描述信息的属性值。

返回值说明

参数名	说明
channelId	通道号，默认为0。
threadId	回调线程ID。
callback	编码回调函数。
enType	视频编码协议。
picFormat	输入图像格式。
picWidth	图片宽度。
picHeight	图片高度。
keyFrameInterval	关键帧间隔，不能为0。
bufAddr	编码输出缓存地址。 在Atlas 200/300/500 推理产品上，当前版本获取该参数的值为无效值，不支持获取该参数值。
bufSize	编码输出缓存大小，单位为Byte。 在Atlas 200/300/500 推理产品上，当前版本获取该参数的值为无效值，不支持获取该参数值。
rcMode	指定码率控制模式。 1表示变码率VBR模式 2表示定码率CBR模式
srcRate	输入码流帧率，单位fps。
maxBitRate	输出码率，单位kbps。

1.13.84.4 aclvencSetChannelDescParam

函数功能

设置视频编码处理通道的属性参数接口。

Atlas 训练系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数原型

```
aclError aclvencSetChannelDescParam(aclvencChannelDesc *channelDesc,
aclvencChannelDescParamType paramType, size_t length, const void *param)
```

参数说明

参数名	输入/输出	说明
channelDesc	输入	视频编码处理通道描述信息的指针。 需提前调用 aclvencCreateChannelDesc 接口创建aclvencChannelDesc类型的数据。
paramType	输入	属性参数的类型。
length	输入	属性参数值的字节数。 - 属性参数的类型为*_UINT64时，此处配置为8。 - 属性参数的类型为*_UINT32时，此处配置为4。 - 属性参数的类型为*_PTR时，若操作系统是32位，则此处配置为4；若操作系统是64位，则配置为8。
param	输入	属性参数值的指针。

返回值说明

返回0表示成功，返回其他值表示失败。

1.13.84.5 aclvencGetChannelDescParam

函数功能

获取视频编码处理通道的描述信息。

Atlas 训练系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数原型

```
aclError aclvencGetChannelDescParam(const aclvencChannelDesc
*channelDesc, aclvencChannelDescParamType paramType, size_t length, size_t
*paramRetSize, void *param)
```

参数说明

参数名	输入/输出	说明
channelDesc	输入	视频编码处理通道描述信息的指针。 需提前调用 aclvencCreateChannelDesc 接口创建aclvencChannelDesc类型的数据，调用 aclvencSetChannelDescParam 接口设置视频编码处理通道描述信息的属性值。
paramType	输入	属性参数的类型。
length	输入	属性参数值的字节数。 - 属性参数的类型为*_UINT64时，此处配置为8； - 属性参数的类型为*_UINT32时，此处配置为4； - 属性参数的类型为*_PTR时，若操作系统是32位，则此处配置为4；若操作系统是64位，则配置为8。
paramRetSize	输出	实际返回的属性参数值字节数的指针。
param	输出	属性参数值的指针。

返回值说明

返回0表示成功，返回其他值表示失败。

1.13.84.6 aclvencDestroyChannelDesc

函数功能

销毁通过[aclvencCreateChannelDesc](#)接口创建的aclvencChannelDesc类型的数据。

Atlas 训练系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数原型

```
aclError aclvencDestroyChannelDesc(aclvencChannelDesc *channelDesc)
```

参数说明

参数名	输入/输出	说明
channelDesc	输入	待销毁的aclvencChannelDesc类型的指针。

返回值说明

返回0表示成功，返回非0表示失败。

1.13.85 aclvencFrameConfig

1.13.85.1 aclvencCreateFrameConfig

函数功能

创建aclvencFrameConfig类型的数据，表示VENC编码时的单帧编码配置参数。

如需销毁aclvencFrameConfig类型的数据，请参见[aclvencDestroyFrameConfig](#)。

Atlas 训练系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数原型

[aclvencFrameConfig](#) *aclvencCreateFrameConfig()

参数说明

无

返回值说明

- 返回aclvencFrameConfig类型的指针，表示成功。
- 返回nullptr，表示失败。

1.13.85.2 aclvencSetFrameConfig 系列接口

函数功能

设置单帧编码配置参数。

Atlas 训练系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数原型

```
aclError aclvencSetFrameConfigForcelFrame(aclvencFrameConfig *config,
uint8_t forcelFrame);

aclError aclvencSetFrameConfigEos(aclvencFrameConfig *config, uint8_t eos)
```

参数说明

参数名	输入/输出	说明
config	输出	单帧编码配置数据的指针。 需提前调用 aclvencCreateFrameConfig 接口创建 aclvencFrameConfig 类型的数据。
forcelFrame	输入	是否强制重新开始I帧间隔： 0：不强制 1：强制重新开始I帧
eos	输入	是否为结束帧： 0：不是 1：是结束帧

返回值说明

返回0表示成功，返回非0表示失败。

1.13.85.3 aclvencGetFrameConfig 系列接口

函数功能

获取单帧编码配置参数。

Atlas 训练系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数原型

```
uint8_t aclvencGetFrameConfigForcelFrame(const aclvencFrameConfig
*config);

uint8_t aclvencGetFrameConfigEos(const aclvencFrameConfig *config)
```


参数说明

参数名	输入/输出	说明
config	输入	单帧编码配置数据的指针。 需提前调用 aclvencCreateFrameConfig 接口创建 aclvencFrameConfig 类型的数据，调用 aclvencSetFrameConfig系列接口 设置单帧编码配置参数。

返回值说明

参数名	说明
forceIframe	是否强制重新开始I帧间隔： 0：不强制 1：强制重新开始I帧
eos	是否为结束帧： 0：不是 1：是结束帧

1.13.85.4 aclvencDestroyFrameConfig

函数功能

销毁通过[aclvencCreateFrameConfig](#)接口创建的[aclvencFrameConfig](#)类型的数据。

Atlas 训练系列产品，不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

函数原型

aclError [aclvencDestroyFrameConfig\(aclvencFrameConfig *config\)](#)

参数说明

参数名	输入/输出	说明
config	输入	待销毁的 aclvencFrameConfig 类型的指针。

返回值说明

返回0表示成功，返回非0表示失败。

1.13.86 acldvppLutMap

1.13.86.1 acldvppCreateLutMap

函数功能

创建acldvppLutMap类型的数据，表示色彩重映射表。在acldvppLutMap类型中，色彩重映射表的维度数量默认为3。

如需销毁acldvppLutMap类型的数据，请参见[acldvppDestroyLutMap](#)。

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

函数原型

[acldvppLutMap](#) *acldvppCreateLutMap()

参数说明

无

返回值说明

- 返回acldvppLutMap类型的指针，表示成功。
- 返回nullptr，表示失败。

1.13.86.2 acldvppDestroyLutMap

函数功能

销毁通过[acldvppCreateLutMap](#)接口创建的acldvppLutMap类型的数据。

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

函数原型

[aclError](#) acldvppDestroyLutMap([acldvppLutMap](#) *lutMap)

参数说明

参数名	输入/输出	说明
lutMap	输入	待销毁的acldvppLutMap类型的指针。

返回值说明

返回0表示成功，返回非0表示失败。

1.13.86.3 acldvppGetLutMapDims

函数功能

获取色彩重映射表的维度数量。

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

函数原型

uint32_t acldvppGetLutMapDims(const **acldvppLutMap** *lutMap)

参数说明

参数名	输入/输出	说明
lutMap	输入	色彩重映射表的指针。 需提前调用 acldvppCreateLutMap 接口创建acldvppLutMap类型的数据。

返回值说明

返回维度数量。

1.13.86.4 acldvppGetLutMapData

函数功能

根据指定维度获取色彩重映射表数据的内存指针及数组长度。

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

函数原型

```
aclError acldvppGetLutMapData(const acldvppLutMap *lutMap,  
                               uint32_t dim,  
                               uint8_t **data,  
                               uint32_t *len)
```

参数说明

参数名	输入/输出	说明
lutMap	输入	色彩重映射表的指针。 需提前调用 acldvppCreateLutMap 接口创建acldvppLutMap类型的数据。
dim	输入	指定色彩重映射表的维度。 调用 acldvppGetLutMapDims 接口获取色彩重映射表的维度数量，dim的取值范围：[0, (维度数量-1)]。
data	输出	“获取指定维度的色彩重映射表数据的内存指针”的指针。
len	输出	data数组长度的指针。

返回值说明

返回0表示成功，返回非0表示失败。

1.13.87 acldvppBorderConfig

1.13.87.1 acldvppCreateBorderConfig

函数功能

创建acldvppBorderConfig类型的数据，表示边界填充配置。

如需销毁acldvppBorderConfig类型的数据，请参见[acldvppDestroyBorderConfig](#)。

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

函数原型

```
acldvppBorderConfig *acldvppCreateBorderConfig()
```

参数说明

无

返回值说明

- 返回acldvppBorderConfig类型的指针，表示成功。
- 返回nullptr，表示失败。

1.13.87.2 acldvppSetBorderConfig 系列接口

函数功能

设置边界填充配置参数。

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

函数原型

```
aclError acldvppSetBorderConfigValue(acldvppBorderConfig *borderConfig,
uint32_t index, double value);

aclError acldvppSetBorderConfigBorderType(acldvppBorderConfig
*borderConfig, acldvppBorderType borderType);

aclError acldvppSetBorderConfigTop(acldvppBorderConfig *borderConfig,
uint32_t top);

aclError acldvppSetBorderConfigBottom(acldvppBorderConfig *borderConfig,
uint32_t bottom);

aclError acldvppSetBorderConfigLeft(acldvppBorderConfig *borderConfig,
uint32_t left);

aclError acldvppSetBorderConfigRight(acldvppBorderConfig *borderConfig,
uint32_t right)
```

参数说明

参数名	输入/输出	说明
borderConfig	输出	边界填充配置数据的指针。 需提前调用acldvppCreateBorderConfig接口创建acldvppBorderConfig类型的数据。
index	输入	表示颜色分量数组的下标。 0表示R或Y分量，1表示G或U分量，2表示B或V分量，3是预留值。

参数名	输入/输出	说明
value	输入	指定填充的各颜色分量的像素值。 仅当borderType设置为BORDER_CONSTANT时，该参数有效。 当前支持用户手动按顺序存放R、G、B分量或Y、U、V分量的值。若输入输出都是YUV格式，填充YUV分量的值；反之，填充的是RGB分量的值。
borderType	输入	填充枚举类型。
top	输入	图像上方填充的像素数。
bottom	输入	图像下方填充的像素数。
left	输入	图像左方填充的像素数。
right	输入	图像右方填充的像素数。

返回值说明

返回0表示成功，返回非0表示失败。

1.13.87.3 acldvppGetBorderConfig 系列接口

函数功能

获取边界填充配置参数。

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

函数原型

```
double acldvppGetBorderConfigValue(const acldvppBorderConfig
*borderConfig, uint32_t index);

acldvppBorderType acldvppGetBorderConfigBorderType(const
acldvppBorderConfig *borderConfig);

uint32_t acldvppGetBorderConfigTop(const acldvppBorderConfig
*borderConfig);

uint32_t acldvppGetBorderConfigBottom(const acldvppBorderConfig
*borderConfig);

uint32_t acldvppGetBorderConfigLeft(const acldvppBorderConfig
*borderConfig);

uint32_t acldvppGetBorderConfigRight(const acldvppBorderConfig
*borderConfig)
```

参数说明

参数名	输入/输出	说明
borderConfig	输入	边界填充配置数据的指针。 需先调用 acldvppCreateBorderConfig 接口创建acldvppBorderConfig类型的数据，调用 acldvppSetBorderConfig 系列接口设置边界填充数据。
index	输入	表示颜色分量数组的下标。 0表示R或Y分量，1表示G或U分量，2表示B或V分量，3是预留值。

返回值说明

参数名	说明
value	表示对应颜色分量的值。
borderType	填充枚举类型。
top	图像上方填充的像素数。
bottom	图像下方填充的像素数。
left	图像左方填充的像素数。
right	图像右方填充的像素数。

1.13.87.4 acldvppDestroyBorderConfig

函数功能

销毁通过[acldvppCreateBorderConfig](#)接口创建的acldvppBorderConfig类型的数据。

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

函数原型

[aclError](#) acldvppDestroyBorderConfig([acldvppBorderConfig](#) *borderConfig)

参数说明

参数名	输入/输出	说明
borderConfig	输入	待销毁的acldvppBorderConfig类型的指针。

返回值说明

返回0表示成功，返回非0表示失败。

1.13.88 acldvppHist

1.13.88.1 acldvppCreateHist

函数功能

创建acldvppHist类型的数据，用于描述颜色分布直方图信息。在acldvppHist类型中，直方图的维度数量默认为3。

如需销毁acldvppHist类型的数据，请参见[acldvppDestroyHist](#)。

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

函数原型

[acldvppHist*](#) acldvppCreateHist()

参数说明

无

返回值说明

- 返回acldvppHist类型的指针，表示成功。
- 返回nullptr，表示失败。

1.13.88.2 acldvppDestroyHist

函数功能

销毁通过[acldvppCreateHist](#)接口创建的acldvppHist类型的数据。

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

函数原型

aclError **acldvppDestroyHist**(**acldvppHist** *hist)

参数说明

参数名	输入/输出	说明
hist	输入	待销毁的acldvppHist类型的指针。

返回值说明

返回0表示成功，返回非0表示失败。

1.13.88.3 acldvppGetHistDims

函数功能

获取直方图数据的维度数量。

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

函数原型

uint32_t **acldvppGetHistDims**(**acldvppHist** *hist)

参数说明

参数名	输入/输出	说明
hist	输入	直方图描述信息的指针。 需提前调用 acldvppCreateHist 接口创建直方图描述信息。

返回值说明

返回维度数量。

1.13.88.4 acldvppGetHistData

函数功能

按指定维度获取直方图数据。

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

函数原型

```
aclError acldvppGetHistData(aclvppHist *hist, uint32_t dim, uint32_t **data,
uint16_t *len)
```

参数说明

参数名	输入/输出	说明
hist	输入	直方图描述信息的指针。 需提前调用 acldvppVpcCalcHistAsync 接口统计各维度直方图数据。
dim	输入	直方图维度。 调用 acldvppGetHistDims 接口获取直方图的维度数量，dim的取值范围：[0, (维度数量-1)]。
data	输出	“对应维度的直方图数据的首地址指针”的指针。
len	输出	直方图数据的长度的指针。

返回值说明

返回0表示成功，返回非0表示失败。

1.13.88.5 acldvppGetHistRetCode

函数功能

获取返回码。

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

函数原型

```
uint32_t acldvppGetHistRetCode(aclvppHist* hist)
```

参数说明

参数名	输入/输出	说明
hist	输入	直方图描述信息的指针。 需提前调用 acldvppVpcCalcHistAsync 接口统计各维度直方图数据。

返回值说明

获取直方图统计是否成功的返回码，0表示成功，非0表示失败。

1.13.88.6 acldvppClearHist

函数功能

将直方图统计数据清零。

使用场景：在统计完一张图像的颜色分布直方图信息后，如果想重复使用acldvppHist类型的变量，可以调用本接口先将acldvppHist类型中的直方图统计数据清零，再统计下一张图像的颜色分布直方图信息。

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 训练系列产品，不支持该接口。

函数原型

```
aclError acldvppClearHist(acldvppHist *hist)
```

参数说明

参数名	输入/输出	说明
hist	输入	直方图描述信息的指针。 需提前调用 acldvppCreateHist 接口创建acldvppHist类型的数据。

返回值说明

返回0表示成功，返回非0表示失败。

1.13.89 aclfvRepoRange

1.13.89.1 aclfvCreateRepoRange

函数功能

创建aclfvRepoRange类型的数据，表示创建特征库范围的参数。

如需销毁aclfvRepoRange类型的数据，请参见[aclfvDestroyRepoRange](#)。

Atlas 200/300/500 推理产品不支持该接口。

Atlas 200/500 A2推理产品不支持该接口。

Atlas 训练系列产品不支持该接口。
Atlas A2训练系列产品/Atlas 800I A2推理产品不支持该接口。

函数原型

aclfvRepoRange *aclfvCreateRepoRange(uint32_t id0Min, uint32_t id0Max, uint32_t id1Min, uint32_t id1Max)

参数说明

参数名	输入/输出	说明
id0Min	输入	id0起始值，取值范围：[0,1023]，需小于等于id0Max。
id0Max	输入	id0最大值，取值范围：[0,1023]。
id1Min	输入	id1起始值，取值范围：[0,1023]，需小于等于id1Max
id1Max	输入	id1最大值，取值范围：[0,1023]。

返回值说明

- 返回aclfvRepoRange类型的指针，表示成功。
- 返回nullptr，表示失败。

1.13.89.2 aclfvDestroyRepoRange

函数功能

销毁通过**aclfvCreateRepoRange**接口创建的aclfvRepoRange类型的数据。
Atlas 200/300/500 推理产品不支持该接口。
Atlas 200/500 A2推理产品不支持该接口。
Atlas 训练系列产品不支持该接口。
Atlas A2训练系列产品/Atlas 800I A2推理产品不支持该接口。

函数原型

aclError aclfvDestroyRepoRange(**aclfvRepoRange** *repoRange)

参数说明

参数名	输入/输出	说明
repoRange	输入	待销毁的aclfvRepoRange类型的指针。

返回值说明

返回0表示成功，返回非0表示失败。

1.13.90 aclfvFeatureInfo

1.13.90.1 aclfvCreateFeatureInfo

函数功能

创建aclfvFeatureInfo类型的数据，表示创建特征描述信息。

如需销毁aclfvFeatureInfo类型的数据，请参见[aclfvDestroyFeatureInfo](#)。

Atlas 200/300/500 推理产品不支持该接口。

Atlas 200/500 A2推理产品不支持该接口。

Atlas 训练系列产品不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品不支持该接口。

函数原型

aclfvFeatureInfo *aclfvCreateFeatureInfo(uint32_t id0, uint32_t id1, uint32_t offset, uint32_t featureLen, uint32_t featureCount, uint8_t *featureData, uint32_t featureDataLen)

参数说明

参数名	输入/输出	说明
id0	输入	一级库id，取值范围0-1023，N:M场景默认为0。通过id0、id1共同确定一个库。
id1	输入	二级库id，取值范围0-1023，N:M场景默认为0。通过id0、id1共同确定一个库。
offset	输入	添加的首个特征在库中的偏移，偏移值需要与库中已添加特征个数一致。N:M场景默认为0。
featureLen	输入	单个特征长度，当前固定为36Byte，系统内部会校验。
featureCount	输入	特征数量，1:N最大100万，N:M最大1000万。
featureData	输入	特征值指针，根据特征长度连续存储，每条特征前4Byte值为0。
featureDataLen	输入	featureData指针申请的内存长度，用于校验。

返回值说明

- 返回[aclfvFeatureInfo](#)类型的指针，表示成功。
- 返回nullptr，表示失败。

1.13.90.2 [aclfvDestroyFeatureInfo](#)

函数功能

销毁通过[aclfvCreateFeatureInfo](#)接口创建的[aclfvFeatureInfo](#)类型的数据。

Atlas 200/300/500 推理产品不支持该接口。

Atlas 200/500 A2推理产品不支持该接口。

Atlas 训练系列产品不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品不支持该接口。

函数原型

[aclError](#) [aclfvDestroyFeatureInfo](#)([aclfvFeatureInfo](#) *featureInfo)

参数说明

参数名	输入/输出	说明
featureInfo	输入	待销毁的 aclfvFeatureInfo 类型的指针。

返回值说明

返回0表示成功，返回非0表示失败。

1.13.91 [aclfvQueryTable](#)

1.13.91.1 [aclfvCreateQueryTable](#)

函数功能

创建[aclfvQueryTable](#)类型的数据，表示创建检索输入表信息。

如需销毁[aclfvQueryTable](#)类型的数据，请参见[aclfvDestroyQueryTable](#)。

Atlas 200/300/500 推理产品不支持该接口。

Atlas 200/500 A2推理产品不支持该接口。

Atlas 训练系列产品不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品不支持该接口。

函数原型

aclfvQueryTable *aclfvCreateQueryTable(uint32_t queryCnt, uint32_t tableLen, uint8_t *tableData, uint32_t tableDataLen)

参数说明

参数名	输入/输出	说明
queryCnt	输入	检索请求数量，1:N场景为1，N:M场景最大1024。
tableLen	输入	单个表长度，长度固定为32KB，系统内部会校验。
tableData	输入	特征值列表的指针，根据特征长度连续存储。
tableDataLen	输入	tableData指针申请的内存长度，用于校验。

返回值说明

- 返回aclfvQueryTable类型的指针，表示成功。
- 返回nullptr，表示失败。

1.13.91.2 aclfvDestroyQueryTable

函数功能

销毁通过[aclfvCreateQueryTable](#)接口创建的aclfvQueryTable类型的数据。

Atlas 200/300/500 推理产品不支持该接口。

Atlas 200/500 A2推理产品不支持该接口。

Atlas 训练系列产品不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品不支持该接口。

函数原型

aclError aclfvDestroyQueryTable(**aclfvQueryTable** *queryTable)

参数说明

参数名	输入/输出	说明
queryTable	输入	待销毁的aclfvQueryTable类型的指针。

返回值说明

返回0表示成功，返回非0表示失败。

1.13.92 aclfvSearchInput

1.13.92.1 aclfvCreateSearchInput

函数功能

创建aclfvSearchInput类型的数据，表示创建检索任务输入信息。

如需销毁aclfvSearchInput类型的数据，请参见[aclfvDestroySearchInput](#)。

Atlas 200/300/500 推理产品不支持该接口。

Atlas 200/500 A2推理产品不支持该接口。

Atlas 训练系列产品不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品不支持该接口。

函数原型

[aclfvSearchInput](#) *aclfvCreateSearchInput([aclfvQueryTable](#) *queryTable,
[aclfvRepoRange](#) *repoRange, uint32_t topk)

参数说明

参数名	输入/输出	说明
queryTable	输入	检索输入表信息的指针。 调用aclfvCreateQueryTable接口创建 aclfvQueryTable 类型的数据。
repoRange	输入	检索特征库范围的指针。 调用aclfvCreateRepoRange接口创建 aclfvRepoRange 类型的数据。
topk	输入	单个检索请求需要返回的结果数量。

返回值说明

- 返回aclfvSearchInput类型的指针，表示成功。
- 返回nullptr，表示失败。

1.13.92.2 aclfvDestroySearchInput

函数功能

销毁通过[aclfvCreateSearchInput](#)接口创建的aclfvSearchInput类型的数据。

Atlas 200/300/500 推理产品不支持该接口。

Atlas 200/500 A2推理产品不支持该接口。

Atlas 训练系列产品不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品不支持该接口。

函数原型

aclError [aclfvDestroySearchInput](#)([aclfvSearchInput](#) *searchInput)

参数说明

参数名	输入/输出	说明
searchInput	输入	待销毁的aclfvSearchInput类型的指针。

返回值说明

返回0表示成功，返回非0表示失败。

1.13.93 aclfvSearchResult

1.13.93.1 aclfvCreateSearchResult

函数功能

创建aclfvSearchResult类型的数据，表示创建检索结果信息。

如需销毁aclfvSearchResult类型的数据，请参见[aclfvDestroySearchResult](#)。

Atlas 200/300/500 推理产品不支持该接口。

Atlas 200/500 A2推理产品不支持该接口。

Atlas 训练系列产品不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品不支持该接口。

函数原型

aclfvSearchResult *[aclfvCreateSearchResult](#)(uint32_t queryCnt, uint32_t *resultNum, uint32_t resultNumDataLen, uint32_t *id0, uint32_t *id1, uint32_t *resultOffset, float *resultDistance, uint32_t dataLen)

参数说明

参数名	输入/输出	说明
queryCnt	输入	检索请求个数，1:N场景为1。
resultNum	输入	结果个数的指针。 每个检索请求的结果个数。
resultNumDataLen	输入	resultNum的内存总长度。
id0	输入	一级库id的指针，总个数为topK*queryCnt。
id1	输入	二级库id的指针，总个数为topK*queryCnt。
resultOffset	输入	偏移指针。 表示每个检索请求的检索结果对应底库的偏移，总个数为topK*queryCnt。
resultDistance	输入	每个检索结果与检索请求间的距离的指针。 总个数为topK*queryCnt。
dataLen	输入	申请的内存大小，计算公式为：topK*queryCnt*4Byte。

返回值说明

- 返回aclfvSearchResult类型的指针，表示成功。
- 返回nullptr，表示失败。

1.13.93.2 aclfvDestroySearchResult

函数功能

销毁通过[aclfvCreateSearchResult](#)接口创建的aclfvSearchResult类型的数据。

Atlas 200/300/500 推理产品不支持该接口。

Atlas 200/500 A2推理产品不支持该接口。

Atlas 训练系列产品不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品不支持该接口。

函数原型

aclError [aclfvDestroySearchResult](#)([aclfvSearchResult](#) *searchResult)

参数说明

参数名	输入/输出	说明
searchResult	输入	待销毁的aclfvSearchResult类型的指针。

返回值说明

返回0表示成功，返回非0表示失败。

1.13.94 aclfvInitPara

1.13.94.1 aclfvCreateInitPara

函数功能

创建aclfvInitPara类型的数据，表示创建特征向量检索的初始化参数。

如需销毁aclfvInitPara类型的数据，请参见[aclfvDestroyInitPara](#)。

Atlas 200/300/500 推理产品不支持该接口。

Atlas 200/500 A2推理产品不支持该接口。

Atlas 训练系列产品不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品不支持该接口。

函数原型

aclfvInitPara *aclfvCreateInitPara(uint64_t fsNum)

参数说明

参数名	输入/输出	说明
fsNum	输入	底库特征个数，用于系统内部给特征检索模块申请内存资源。 取值范围：[1,500000000]

返回值说明

- 返回aclfvInitPara类型的指针，表示成功。
- 返回nullptr，表示失败。

1.13.94.2 aclfvDestroyInitPara

函数功能

销毁通过[aclfvCreateInitPara](#)接口创建的[aclfvInitPara](#)类型的数据。

Atlas 200/300/500 推理产品不支持该接口。

Atlas 200/500 A2推理产品不支持该接口。

Atlas 训练系列产品不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品不支持该接口。

函数原型

aclError [aclfvDestroyInitPara](#)([aclfvInitPara](#) *initPara)

参数说明

参数名	输入/输出	说明
initPara	输入	待销毁的 aclfvInitPara 类型的指针。

返回值说明

返回0表示成功，返回非0表示失败。

1.13.94.3 aclfvSet1NTopNum

函数功能

设置1:N场景下返回的topK结果数量的最大值，如果不设置，默认topK结果数量的最大值为4800。

Atlas 200/300/500 推理产品不支持该接口。

Atlas 200/500 A2推理产品不支持该接口。

Atlas 训练系列产品不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品不支持该接口。

函数原型

aclError [aclfvSet1NTopNum](#)([aclfvInitPara](#) *initPara, uint32_t maxTopNumFor1N)

参数说明

参数名	输入/输出	说明
initPara	输入&输出	特征向量检索初始化参数的指针。 需提前调用 aclfvCreateInitPara 接口创建 aclfvInitPara 类型的数据。
maxTopNumFor1N	输入	1:N场景下返回的topK结果的最大值，取值范围：[2,4800]。

返回值说明

返回0表示成功，返回非0表示失败。

1.13.94.4 aclfvSetNMTopNum

函数功能

设置N:M场景下返回的topK结果数量的最大值，如果不设置，默认topK结果数量的最大值为500。

Atlas 200/300/500 推理产品不支持该接口。

Atlas 200/500 A2推理产品不支持该接口。

Atlas 训练系列产品不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品不支持该接口。

函数原型

aclError [aclfvSetNMTopNum](#)([aclfvInitPara](#) *initPara, uint32_t maxTopNumForNM)

参数说明

参数名	输入/输出	说明
initPara	输入&输出	特征向量检索初始化参数的指针。 需提前调用 aclfvCreateInitPara 接口创建 aclfvInitPara 类型的数据。
maxTopNumForNM	输入	N:M场景下返回的topK结果的最大值，取值范围：[500,4800]。

返回值说明

返回0表示成功，返回非0表示失败。

1.13.95 aclprofConfig

1.13.95.1 aclprofCreateConfig

函数功能

创建aclprofConfig类型的数据，表示创建Profiling配置数据。

aclProfConfig类型数据可以只创建一次、多处使用，用户需要保证数据的一致性和准确性。

如需销毁aclprofConfig类型的数据，请参见[aclprofDestroyConfig](#)。

约束说明

- 使用aclprofDestroyConfig接口销毁aclprofConfig类型的数据，如不销毁会导致内存未被释放。
- 与[aclprofDestroyConfig](#)接口配对使用，先调用aclprofCreateConfig接口再调用aclprofDestroyConfig接口。

函数原型

aclprofConfig *aclprofCreateConfig(uint32_t *deviceIdList, uint32_t deviceNums, **aclprofAicoreMetrics** aicoreMetrics, const aclprofAicoreEvents *aicoreEvents, uint64_t dataTypeConfig)

参数说明

参数名	输入/输出	说明
deviceIdList	输入	Device ID列表。须根据实际环境的Device ID配置。
deviceNums	输入	Device的个数。需由用户保证deviceIdList中的Device个数与deviceNums参数值一致，否则可能会导致后续业务异常。
aicoreMetrics	输入	表示AI Core性能指标采集项。
aicoreEvents	输入	表示AI Core事件，目前配置为NULL。

参数名	输入/输出	说明
dataTypeConfig	输入	用户选择如下多个宏进行逻辑或（例如： ACL_PROF_ACL_API ACL_PROF_AICORE_METRICS），作为dataTypeConfig 参数值。每个宏表示某一类性能数据，详细说明如下： • ACL_PROF_ACL_API：表示采集AscendCL接口的性能数据，包括Host与Device之间、Device间的同步异步内存复制时延等。 • ACL_PROF_TASK_TIME：采集算子下发耗时、算子执行耗时数据以及算子基本信息数据，提供更全面的性能分析数据。 • ACL_PROF_TASK_TIME_L0：采集算子下发耗时、算子执行耗时数据。与ACL_PROF_TASK_TIME相比，由于不采集算子基本信息数据，采集时性能开销较小，可更精准统计相关耗时数据。 • ACL_PROF_OP_ATTR：控制采集算子的属性信息，当前仅支持aclnn算子。 • ACL_PROF_AICORE_METRICS：表示采集AI Core性能指标数据，逻辑或时必须包括该宏，aicoreMetrics入参处配置的性能指标采集项才有效。 • ACL_PROF_TASK_MEMORY：控制CANN算子的内存占用情况采集开关，用于优化内存使用。单算子场景下，按照GE组件维度和算子维度采集算子内存大小及生命周期信息（单算子API执行方式不采集GE组件内存）；静态图和静态子图场景下，在算子编译阶段按照算子维度采集算子内存大小及生命周期信息。 • ACL_PROF_AICPU：表示采集AI CPU任务的开始、结束数据。 • ACL_PROF_L2CACHE：表示采集L2 Cache数据（Atlas 200/300/500 推理产品不支持）。 • ACL_PROF_HCCL_TRACE：控制HCCL数据采集开关。 • ACL_PROF_MSPROFTX：获取用户和上层框架程序输出的性能数据。可在采集进程内（aclprofStart接口、aclprofStop接口之间）调用如下两种接口开启记录应用程序执行期间特定事件发生的时间跨度，并写入性能数据文件，再使用msprof工具解析该文件，并导出展示性能分析数据： - mstx API（MindStudio Tools Extension API）接口介绍和示例代码请参见详细操作请参见mstx API参考。 - Profiling AscendCL API扩展接口（统称为msproftx接口）详细操作请参见Profiling性能数据采集。 • ACL_PROF_TRAINING_TRACE：控制迭代轨迹数据采集开关。

参数名	输入/输出	说明
		ACL_PROF_RUNTIME_API: 控制runtime api性能数据采集开关。

返回值说明

- 返回aclprofConfig类型的指针，表示成功。
- 返回nullptr，表示失败。

1.13.95.2 aclprofDestroyConfig

函数功能

销毁通过[aclprofCreateConfig](#)接口创建的aclprofConfig类型的数据。

约束说明

- 与[aclprofCreateConfig](#)接口配对使用，先调用aclprofCreateConfig接口再调用aclprofDestroyConfig接口。
- 同一aclprofConfig指针重复调用aclprofDestroyConfig接口，会出现重复释放内存的报错。

函数原型

aclError aclprofDestroyConfig(const **aclprofConfig** *profilerConfig)

参数说明

参数名	输入/输出	说明
profilerConfig	输入	待销毁的aclprofConfig类型的指针。

返回值说明

返回0表示成功，返回非0表示失败。

1.13.96 aclprofSubscribeConfig

1.13.96.1 aclprofCreateSubscribeConfig

函数功能

创建aclprofSubscribeConfig类型的数据，表示创建订阅配置信息。

如需销毁[aclprofSubscribeConfig](#)类型的数据，请参见[aclprofDestroySubscribeConfig](#)。

约束说明

- 使用[aclprofDestroySubscribeConfig](#)接口销毁[aclprofSubscribeConfig](#)类型的数据，如不销毁会导致内存未被释放。
- 与[aclprofDestroySubscribeConfig](#)接口配对使用，先调用[aclprofCreateSubscribeConfig](#)接口再调用[aclprofDestroySubscribeConfig](#)接口。

函数原型

[aclprofSubscribeConfig](#) *[aclprofCreateSubscribeConfig](#)(int8_t timeInfoSwitch, [aclprofAicoreMetrics](#) aicoreMetrics, void *fd)

参数说明

参数名	输入/输出	说明
timeInfoSwitch	输入	是否采集网络模型中算子的性能数据： 1：采集 0：不采集
aicoreMetrics	输入	表示AI Core性能指标采集项。 说明 订阅接口目前仅提供算子耗时统计的功能，暂时不支持AicoreMetrics采集功能。
fd	输入	用户创建的管道写指针。 用户在调用 aclprofModelUnSubscribe 接口后，系统内部会在数据发送结束后，关闭该模型的管道写指针。

返回值说明

- 返回[aclprofSubscribeConfig](#)类型的指针，表示成功。
- 返回nullptr，表示失败。

1.13.96.2 [aclprofDestroySubscribeConfig](#)

函数功能

销毁通过[aclprofCreateSubscribeConfig](#)接口创建的[aclprofSubscribeConfig](#)类型的数据。

约束说明

- 与[aclprofCreateSubscribeConfig](#)接口配对使用，先调用[aclprofCreateSubscribeConfig](#)接口再调用[aclprofDestroySubscribeConfig](#)接口。
- 同一[aclprofSubscribeConfig](#)指针重复调用[aclprofDestroySubscribeConfig](#)接口，会出现重复释放内存的报错。

函数原型

aclError aclprofDestroySubscribeConfig(const **aclprofSubscribeConfig** *profSubscribeConfig)

参数说明

参数名	输入/输出	说明
profSubscribeConfig	输入	待销毁的aclprofSubscribeConfig类型的指针。

返回值说明

返回0表示成功，返回非0表示失败。

1.13.97 aclprofStepInfo

1.13.97.1 aclprofCreateStepInfo

函数功能

创建aclprofStepInfo对象，用于描述迭代信息。

约束说明

- 使用aclprofDestroyStepInfo接口销毁aclprofStepInfo类型的数据，如不销毁会导致内存未被释放。
- 与[aclprofDestroyStepInfo](#)接口配对使用，先调用aclprofCreateStepInfo接口再调用aclprofDestroyStepInfo接口。

函数原型

aclprofStepInfo* aclprofCreateStepInfo()

返回值

- 返回aclprofStepInfo类型的指针，表示成功。
- 返回nullptr，表示失败。

说明

同一个aclprofStepInfo对象、同一个tag只能设置一次，否则Profiling解析会出错。

1.13.97.2 aclprofDestroyStepInfo

函数功能

销毁通过[aclprofCreateStepInfo](#)接口创建的aclprofStepInfo类型的数据。

约束说明

- 与[aclprofCreateStepInfo](#)接口配对使用，先调用[aclprofCreateStepInfo](#)接口再调用[aclprofDestroyStepInfo](#)接口。
- 同一[aclprofStepInfo](#)数据重复调用[aclprofDestroyStepInfo](#)接口，会出现重复释放内存的报错。

函数原型

void aclprofDestroyStepInfo (aclprofStepInfo* stepinfo)

参数说明

参数名	输入/输出	说明
stepinfo	输入	指定迭代信息。需提前调用 aclprofCreateStepInfo 接口创建 aclprofStepInfo 类型的数据。

1.13.98 acltdtDataItem

1.13.98.1 acltdtCreateDataItem

函数功能

创建[acltdtDataItem](#)类型的数据，代表一个业务上的Tensor。

如需销毁[acltdtDataItem](#)类型的数据，请参见[acltdtDestroyDataItem](#)。

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 200/500 A2推理产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

函数原型

**[acltdtDataItem](#) *acltdtCreateDataItem([acltdtTensorType](#) tdtType,
const int64_t *dims,
size_t dimNum,
[aclDataType](#) dataType,
void *data,
size_t size)**

参数说明

参数名	输入/输出	说明
tdtType	输入	tensor类型。
dims	输入	tensor的Shape。
dimNum	输入	tensor的Shape中的维度个数。
dataType	输入	正常数据里的数据类型。
data	输入	数据地址指针。
size	输入	数据长度。

返回值说明

- 返回acltdtDataItem类型的指针，表示成功。
- 返回nullptr，表示失败。

1.13.98.2 acltdtDestroyDataItem

函数功能

销毁通过[acltdtCreateDataItem](#)接口创建的acltdtDataItem类型的数据。

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 200/500 A2推理产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

函数原型

aclError [acltdtDestroyDataItem](#)([acltdtDataItem](#) *dataItem)

参数说明

参数名	输入/输出	说明
dataItem	输入	待销毁的acltdtDataItem类型的指针。

返回值说明

返回0表示成功，返回非0表示失败。

1.13.98.3 acltdtGetTensorTypeFromItem

函数功能

收到dataItem数据之后，从数据描述里首先check这个数据是个正常数据，还是一个end数据，还是一个异常数据。

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 200/500 A2推理产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

函数原型

acltdtTensorType acltdtGetTensorTypeFromItem(const **acltdtDataItem** *dataItem)

参数说明

参数名	输入/输出	说明
dataItem	输入	acltdtDataItem类型的指针。 需提前调用 acltdtCreateDataItem 接口创建acltdtDataItem类型的数据。

返回值说明

请参见[acltdtTensorType](#)。

1.13.98.4 acltdtGetDataItemFromItem

函数功能

获取正常数据的数据类型。

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 200/500 A2推理产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

函数原型

aclDataType acltdtGetDataItemFromItem(const **acltdtDataItem** *dataItem)

参数说明

参数名	输入/输出	说明
dataltem	输入	acldtdDataItem类型的指针。 需提前调用 acldtdCreateDataItem 接口创建acldtdDataItem类型的数据。

返回值说明

请参见[acldtdTensorType](#)。

1.13.98.5 acldtdGetDataAddrFromItem

函数功能

得到正常数据的数据地址。

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 200/500 A2推理产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

函数原型

void *acldtdGetDataAddrFromItem(const [acldtdDataItem](#) *dataltem)

参数说明

参数名	输入/输出	说明
dataltem	输入	acldtdDataItem类型的指针。 需提前调用 acldtdCreateDataItem 接口创建acldtdDataItem类型的数据。

返回值说明

返回正常数据的数据地址。

1.13.98.6 acldtdGetDataSizeFromItem

函数功能

得到正常数据的数据长度。

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 200/500 A2推理产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

函数原型

size_t acltdtGetDataSizeFromItem(const [acltdtDataItem](#) *dataItem)

参数说明

参数名	输入/输出	说明
dataItem	输入	acltdtDataItem类型的指针。 需提前调用 acltdtCreateDataItem 接口创建acltdtDataItem类型的数据。

返回值说明

请参见[acltdtTensorType](#)。

1.13.98.7 acltdtGetDimNumFromItem

函数功能

得到正常tensor数据的数据Shape中的维度个数。

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 200/500 A2推理产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

函数原型

size_t acltdtGetDimNumFromItem(const [acltdtDataItem](#) *dataItem)

参数说明

参数名	输入/输出	说明
dataItem	输入	acltdtDataItem类型的指针。 需提前调用 acltdtCreateDataItem 接口创建acltdtDataItem类型的数据。

返回值说明

请参见[acltdtTensorType](#)。

1.13.98.8 acltdtGetDimsFromItem

函数功能

得到正常tensor数据的数据Shape。

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 200/500 A2推理产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

函数原型

aclError acltdtGetDimsFromItem(const **acltdtDataItem** *dataItem, int64_t *dims, size_t dimNum)

参数说明

参数名	输入/输出	说明
dataItem	输入	acltdtDataItem类型的指针。 需提前调用 acltdtCreateDataItem 接口创建acltdtDataItem类型的数据。
dims	输入&输出	维度信息数组。
dimNum	输入	维度个数。

返回值说明

请参见[acltdtTensorType](#)。

1.13.99 acltdtDataset

1.13.99.1 acltdtCreateDataset

函数功能

创建acltdtDataset类型的数据，对等一个Vector<tensor>。

如需销毁acltdtDataset类型的数据，请参见[acltdtDestroyDataset](#)。

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 200/500 A2推理产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

函数原型

acltdtDataset *acltdtCreateDataset()

参数说明

无

返回值说明

- 返回acltdtDataset类型的指针，表示成功。
- 返回nullptr，表示失败。

1.13.99.2 acltdtDestroyDataset

函数功能

销毁通过**acltdtCreateDataset**接口创建的acltdtDataset类型的数据。

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 200/500 A2推理产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

函数原型

aclError acltdtDestroyDataset(**acltdtDataset** *dataset)

参数说明

参数名	输入/输出	说明
dataset	输入	待销毁的acltdtDataset类型的指针。

返回值说明

返回0表示成功，返回非0表示失败。

1.13.99.3 acltdtGetDataItem

函数功能

获取acltdtDataset中的第n个acltdtDataItem。

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 200/500 A2推理产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

函数原型

acldtdDataItem *acldtdGetDataItem(const **acldtdDataset** *dataset, size_t index)

参数说明

参数名	输入/输出	说明
dataset	输入	acldtdDataset类型的指针。 需提前调用 acldtdCreateDataset 接口创建acldtdDataset类型的数据，再调用 acldtdAddDataItem 接口添加acldtdDataItem。
index	输入	表明获取的是第几个acldtdDataItem。

返回值说明

- 获取成功，返回acldtdDataItem的地址。
- 获取失败返回空地址。

1.13.99.4 acldtdAddDataItem

函数功能

向acldtdDataset中增加acldtdDataItem。

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 200/500 A2推理产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

函数原型

aclError acldtdAddDataItem(**acldtdDataset** *dataset, **acldtdDataItem** *dataItem)

参数说明

参数名	输入/输出	说明
dataset	输出	待增加acldtdDataItem的acldtdDataset地址指针。 需提前调用 acldtdCreateDataset 接口创建acldtdDataset类型的数据。

参数名	输入/输出	说明
dataltem	输入	待增加的acltdtDataltem地址指针。 需提前调用 acltdtCreateDataltem 接口创建acltdtDataltem类型的数据。

返回值说明

返回0表示成功，返回非0表示失败。

1.13.99.5 acltdtGetDatasetSize

函数功能

获取acltdtDataset中acltdtDataltem的个数。

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 200/500 A2推理产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

函数原型

size_t acltdtGetDatasetSize(const [acltdtDataset](#) *dataset)

参数说明

参数名	输入/输出	说明
dataset	输入	acltdtDataset类型的指针。 需提前调用 acltdtCreateDataset 接口创建acltdtDataset类型的数据。

返回值说明

acltdtDataltem的个数。

1.13.99.6 acltdtGetDatasetName

函数功能

获取标识acltdtDataset（对等一个Vector<tensor>）的描述符。

Atlas 200/300/500 推理产品，不支持该接口。

Atlas 200/500 A2推理产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

函数原型

```
const char *acltdtGetDatasetName(const acltdtDataset *dataset)
```

参数说明

参数名	输入/输出	说明
dataset	输入	acltdtDataset类型的指针。 需提前调用 acltdtCreateDataset 接口创建acltdtDataset类型的数据。

返回值说明

标识acltdtDataset（对等一个Vector<tensor>）的描述符的指针。

1.13.100 aclmdlConfigHandle

1.13.100.1 aclmdlCreateConfigHandle

函数功能

创建aclmdlConfigHandle类型的数据，表示一个模型加载的配置对象。
如需销毁aclmdlConfigHandle类型的数据，请参见[aclmdlDestroyConfigHandle](#)。

函数原型

```
aclmdlConfigHandle *aclmdlCreateConfigHandle()
```

参数说明

无

返回值说明

- 返回aclmdlConfigHandle类型的指针，表示成功。
- 返回nullptr，表示失败。

1.13.100.2 aclmdlDestroyConfigHandle

函数功能

销毁通过[aclmdlCreateConfigHandle](#)接口创建的aclmdlConfigHandle类型的数据。

函数原型

```
aclError aclmdlDestroyConfigHandle(aclmdlConfigHandle *handle)
```

参数说明

参数名	输入/输出	说明
handle	输入	待销毁的aclmdlConfigHandle类型的指针。

返回值说明

返回0表示成功，返回其他值表示失败。

1.13.101 acltdtQueueAttr

1.13.101.1 acltdtCreateQueueAttr

函数功能

创建acltdtQueueAttr类型的数据，表示队列属性配置信息。

函数原型

```
acltdtQueueAttr *acltdtCreateQueueAttr()
```

参数说明

无

返回值说明

- 返回acltdtQueueAttr类型的指针，表示成功。
- 返回nullptr，表示失败。

1.13.101.2 acltdtDestroyQueueAttr

函数功能

销毁通过acltdtCreateQueueAttr接口创建的acltdtQueueAttr类型的数据。

函数原型

```
aclError acltdtDestroyQueueAttr(const acltdtQueueAttr *attr)
```

参数说明

参数名	输入/输出	说明
attr	输入	待销毁的acltdtQueueAttr类型的指针。

返回值说明

返回0表示成功，返回其他值表示失败。

1.13.101.3 acltdtSetQueueAttr

函数功能

设置队列属性值。

函数原型

aclError acltdtSetQueueAttr(**acltdtQueueAttr** *attr, **acltdtQueueAttrType** type, **size_t** len, **const void** *param)

参数说明

参数名	输入/输出	说明
attr	输入&输出	队列属性配置信息的指针。 需提前调用 acltdtCreateQueueAttr 接口创建acltdtQueueAttr类型的数据。
type	输入	属性类型。
len	输入	属性值的字节数。 - 属性参数的类型为*_UINT64时，此处配置为8； - 属性参数的类型为*_UINT32时，此处配置为4； - 属性参数的类型为*_PTR时，若操作系统是32位，则此处配置为4；若操作系统是64位，则配置为8。
param	输入	指向属性值的指针。

返回值说明

返回0表示成功，返回其他值表示失败。

1.13.101.4 acltdtGetQueueAttr

函数功能

获取队列属性值。

函数原型

```
aclError acltdtGetQueueAttr(const acltdtQueueAttr *attr,  
acltdtQueueAttrType type, size_t len, size_t *paramRetSize, void *param)
```

参数说明

参数名	输入/输出	说明
attr	输入	队列属性配置信息的指针。 需提前调用 acltdtCreateQueueAttr 接口创建acltdtQueueAttr类型的数据。
type	输入	属性类型。
len	输入	属性值的字节数。 - 属性参数的类型为*_UINT64时，此处配置为8； - 属性参数的类型为*_UINT32时，此处配置为4； - 属性参数的类型为*_PTR时，若操作系统是32位，则此处配置为4；若操作系统是64位，则配置为8。
paramRetSize	输出	实际返回的属性值字节数的指针。
param	输出	指向属性值的指针。

返回值说明

返回0表示成功，返回其他值表示失败。

1.13.102 acltdtQueueRoute

1.13.102.1 acltdtCreateQueueRoute

函数功能

创建acltdtQueueRoute类型的数据，表示创建队列路由配置。

函数原型

```
acltdtQueueRoute* acltdtCreateQueueRoute(uint32_t srcId, uint32_t dstId)
```

参数说明

参数名	输入/输出	说明
srcId	输入	源队列ID。
dstId	输入	目的队列ID。

返回值说明

- 返回[acltdtQueueRoute](#)类型的指针，表示成功。
- 返回[nullptr](#)，表示失败。

1.13.102.2 acltdtDestroyQueueRoute

函数功能

销毁通过[acltdtCreateQueueRoute](#)接口创建的[acltdtQueueRoute](#)类型的数据。

函数原型

[aclError](#) [acltdtDestroyQueueRoute](#)(const [acltdtQueueRoute](#) *route)

参数说明

参数名	输入/输出	说明
route	输入	待销毁的 acltdtQueueRoute 类型的指针。

返回值说明

返回0表示成功，返回其他值表示失败。

1.13.102.3 acltdtGetQueueRouteParam

函数功能

获取队列路由配置的相关信息，例如源队列ID、目标队列ID等。

函数原型

[aclError](#) [acltdtGetQueueRouteParam](#)(const [acltdtQueueRoute](#) *route, [acltdtQueueRouteParamType](#) type, size_t len, size_t *paramRetSize, void *param)

参数说明

参数名	输入/输出	说明
route	输入	队列路由配置信息的指针。 需提前调用 acltdtCreateQueueRoute 接口创建acltdtQueueRoute类型的数据。
type	输入	路由参数类型。
len	输入	参数值的字节数。 - 参数的类型为*_UINT64时，此处配置为8； - 参数的类型为*_UINT32时，此处配置为4； - 参数的类型为*_PTR时，若操作系统是32位，则此处配置为4；若操作系统是64位，则配置为8。
paramRetSize	输出	实际返回的参数值字节数的指针。
param	输出	指向参数值的指针。

返回值说明

返回0表示成功，返回其他值表示失败。

1.13.103 acltdtQueueRouteList

1.13.103.1 acltdtCreateQueueRouteList

函数功能

创建acltdtQueueRouteList类型的数据，表示队列路由配置数组。

函数原型

[acltdtQueueRouteList*](#) [acltdtCreateQueueRouteList\(\)](#)

参数说明

无

返回值说明

- 返回acltdtQueueRouteList类型的指针，表示成功。
- 返回nullptr，表示失败。

1.13.103.2 acltdtDestroyQueueRouteList

函数功能

销毁通过[acltdtCreateQueueRouteList](#)接口创建的acltdtQueueRouteList类型的数据。

函数原型

aclError acltdtDestroyQueueRouteList(const **acltdtQueueRouteList** *routeList)

参数说明

参数名	输入/输出	说明
routeList	输入	待销毁的acltdtQueueRouteList类型的指针。

返回值说明

返回0表示成功，返回其他值表示失败。

1.13.103.3 acltdtAddQueueRoute

函数功能

向队列路由配置数组中添加队列路由配置信息。

函数原型

aclError acltdtAddQueueRoute(**acltdtQueueRouteList** *routeList, const **acltdtQueueRoute** *route)

参数说明

参数名	输入/输出	说明
routeList	输入&输出	队列路由配置数组。 需提前调用 acltdtCreateQueueRouteList 接口创建acltdtQueueRouteList类型的数据。
route	输入	需添加的队列路由配置信息的指针。 需提前调用 acltdtCreateQueueRoute 接口创建acltdtQueueRoute类型的数据。

返回值说明

返回0表示成功，返回其他值表示失败。

1.13.103.4 acltdtGetQueueRoute

函数功能

从队列路由配置数组中获取指定的队列路由配置信息。

函数原型

```
aclError acltdtGetQueueRoute(const acltdtQueueRouteList *routeList, size_t index, acltdtQueueRoute *route)
```

参数说明

参数名	输入/输出	说明
routeList	输入	队列路由配置数组。 需提前调用 acltdtCreateQueueRouteList 接口创建acltdtQueueRouteList类型的数据。
index	输入	指定获取哪一个队列路由配置信息，index编号从0开始。
route	输入&输出	需添加的队列路由配置信息的指针。

返回值说明

返回0表示成功，返回其他值表示失败。

1.13.103.5 acltdtGetQueueRouteNum

函数功能

从队列路由配置数组中获取路由数量。

函数原型

```
size_t acltdtGetQueueRouteNum(const acltdtQueueRouteList *routeList)
```

参数说明

参数名	输入/输出	说明
routeList	输入	队列路由配置数组。 需提前调用 acltdtCreateQueueRouteList 接口创建acltdtQueueRouteList类型的数据。

返回值说明

返回路由数量。

1.13.104 acltdtQueueRouteQueryInfo

1.13.104.1 acltdtCreateQueueRouteQueryInfo

函数功能

创建acltdtQueueRouteQueryInfo类型的数据，表示队列路由关系查询条件。

函数原型

```
acltdtQueueRouteQueryInfo* acltdtCreateQueueRouteQueryInfo()
```

参数说明

无

返回值说明

- 返回acltdtQueueRouteQueryInfo类型的指针，表示成功。
- 返回nullptr，表示失败。

1.13.104.2 acltdtDestroyQueueRouteQueryInfo

函数功能

销毁通过[acltdtCreateQueueRouteQueryInfo](#)接口创建的acltdtQueueRouteQueryInfo类型的数据。

函数原型

```
aclError acltdtDestroyQueueRouteQueryInfo(const  
acltdtQueueRouteQueryInfo *info)
```

参数说明

参数名	输入/输出	说明
info	输入	待销毁的acltdtQueueRouteQueryInfo类型的指针。

返回值说明

返回0表示成功，返回其他值表示失败。

1.13.104.3 acltdtSetQueueRouteQueryInfo

函数功能

设置路由关系查询条件。

函数原型

```
aclError acltdtSetQueueRouteQueryInfo(acltdtQueueRouteQueryInfo *param,
acltdtQueueRouteQueryInfoParamType type, size_t len, const void *value)
```

参数说明

参数名	输入/输出	说明
param	输入&输出	队列路由关系查询条件的指针。 acltdtCreateQueueRouteQueryInfo
type	输入	参数类型。
len	输入	参数值的字节数。 - 属性参数的类型为*_UINT64时，此处配置为8； - 属性参数的类型为*_UINT32时，此处配置为4； - 属性参数的类型为*_PTR时，若操作系统是32位，则此处配置为4；若操作系统是64位，则配置为8。
value	输入	参数值的指针。

返回值说明

返回0表示成功，返回其他值表示失败。

1.13.105 aclGraphDumpOption

1.13.105.1 aclCreateGraphDumpOpt

函数功能

创建aclGraphDumpOption类型的数据，表示单算子图Dump配置信息，用于设置图Dump时的选项（当前还没有选项，预留该数据类型，便于后续扩展）。

函数原型

```
aclGraphDumpOption *aclCreateGraphDumpOpt()
```

参数说明

无

返回值说明

- 返回[aclGraphDumpOption](#)类型的指针，表示成功。
- 返回[nullptr](#)，表示失败。

1.13.105.2 aclDestroyGraphDumpOpt

函数功能

销毁通过[aclCreateGraphDumpOpt](#)接口创建的[aclGraphDumpOption](#)类型的数据。

函数原型

[aclError](#) [aclDestroyGraphDumpOpt](#)(const [aclGraphDumpOption](#)
[*graphDumpOpt](#))

参数说明

参数名	输入/输出	说明
graphDumpOpt	输入	待销毁的 aclGraphDumpOption 类型的指针。

返回值说明

返回0表示成功，返回其他值表示失败。

1.13.106 aclmdlExecConfigHandle

1.13.106.1 aclmdlCreateExecConfigHandle

函数功能

创建[aclmdlExecConfigHandle](#)类型的数据，表示一个模型执行的配置对象。

如需销毁[aclmdlExecConfigHandle](#)类型的数据，请参见[aclmdlDestroyExecConfigHandle](#)。

函数原型

[aclmdlExecConfigHandle](#) *[aclmdlCreateExecConfigHandle](#)()

参数说明

无

返回值说明

- 返回[aclmdlExecConfigHandle](#)类型的指针，表示成功。
- 返回NULL，表示失败。

1.13.106.2 aclmdlDestroyExecConfigHandle

函数功能

销毁通过[aclmdlCreateExecConfigHandle](#)接口创建的[aclmdlExecConfigHandle](#)类型的数据。

函数原型

aclError [aclmdlDestroyExecConfigHandle](#)(const [aclmdlExecConfigHandle](#) *handle)

参数说明

参数名	输入/输出	说明
handle	输入	待销毁的 aclmdlExecConfigHandle 类型的指针。

返回值说明

返回0表示成功，返回其他值表示失败。

1.13.107 aclrtStreamConfigHandle

1.13.107.1 aclrtCreateStreamConfigHandle

函数功能

创建[aclrtStreamConfigHandle](#)类型的数据，表示一个Stream的配置对象。

如需销毁[aclrtStreamConfigHandle](#)类型的数据，请参见[aclrtDestroyStreamConfigHandle](#)。

Atlas 200/300/500 推理产品不支持该接口。

Atlas 200/500 A2推理产品不支持该接口。

Atlas 训练系列产品不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

函数原型

aclrtStreamConfigHandle *[aclrtCreateStreamConfigHandle](#)(void)

参数说明

无

返回值说明

- 返回aclrtStreamConfigHandle类型的指针，表示成功。
- 返回NULL，表示失败。

1.13.107.2 aclrtDestroyStreamConfigHandle

函数功能

销毁通过[aclrtCreateStreamConfigHandle](#)接口创建的aclrtStreamConfigHandle类型的数据。

Atlas 200/300/500 推理产品不支持该接口。

Atlas 200/500 A2推理产品不支持该接口。

Atlas 训练系列产品不支持该接口。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

Atlas 推理系列产品，不支持该接口。

函数原型

aclError [aclrtDestroyStreamConfigHandle](#)([aclrtStreamConfigHandle](#) *handle)

参数说明

参数名	输入/输出	说明
handle	输入	待销毁的aclrtStreamConfigHandle类型的指针。

返回值说明

返回0表示成功，返回其他值表示失败。

1.13.108 aclrtAllocatorDesc

1.13.108.1 aclrtAllocatorCreateDesc

函数功能

创建aclrtAllocatorDesc类型的数据，表示Allocator描述信息，主要用于注册回调函数。

函数原型

```
aclrtAllocatorDesc aclrtAllocatorCreateDesc()
```

返回值说明

- 返回aclrtAllocatorDesc类型的指针，表示成功。
- 返回NULL，表示失败。

1.13.108.2 aclrtAllocatorDestroyDesc

函数功能

销毁通过aclrtAllocatorCreateDesc接口创建[的aclrtAllocatorDesc类型的数据](#)。

原型

```
aclError aclrtAllocatorDestroyDesc(aclrtAllocatorDesc allocatorDesc)
```

参数说明

参数名	输入/输出	说明
allocatorDesc	输入	待销毁的aclrtAllocatorDesc类型的数据。

返回值说明

返回0表示成功，返回其他值表示失败。

1.13.108.3 aclrtAllocatorSetObjToDesc

函数功能

使用用户提供的Allocator场景下，向Allocator描述信息中设置Allocator对象。

函数原型

```
aclError aclrtAllocatorSetObjToDesc(aclrtAllocatorDesc allocatorDesc, aclrtAllocator allocator)
```

参数说明

参数名	输入/输出	说明
allocatorDesc	输入	Allocator描述符指针。 需提前调用 aclrtAllocatorCreateDesc 接口设置Allocator描述信息。
allocator	输入	用户提供的Allocator对象指针。

返回值说明

返回0表示成功，返回其他值表示失败。

1.13.108.4 aclrtAllocatorSetAllocFuncToDesc

函数功能

使用用户提供的Allocator场景下，设置"申请内存block"的回调函数。

函数原型

[aclError](#) [aclrtAllocatorSetAllocFuncToDesc](#)([aclrtAllocatorDesc](#) allocatorDesc, [aclrtAllocatorAllocFunc](#) func)

参数说明

参数名	输入/输出	说明
allocatorDesc	输入	Allocator描述符指针。 需提前调用 aclrtAllocatorCreateDesc 接口设置Allocator描述信息。
func	输入	申请内存block的回调函数。 回调函数定义如下： typedef void *(*aclrtAllocatorAllocFunc)(aclrtAllocator allocator, size_t size); aclrtAllocator的定义如下： typedef void *aclrtAllocator;

返回值说明

返回0表示成功，返回其他值表示失败。

1.13.108.5 aclrtAllocatorSetAllocAdviseFuncToDesc

函数功能

使用用户提供的Allocator场景下，设置"根据建议地址申请内存block"的回调函数，一般用于内存复用场景。

函数原型

aclError aclrtAllocatorSetAllocAdviseFuncToDesc(**aclrtAllocatorDesc** allocatorDesc, **aclrtAllocatorAllocAdviseFunc** func)

参数说明

参数名	输入/输出	说明
allocatorDesc	输入	Allocator描述符指针。 需提前调用 aclrtAllocatorCreateDesc 接口设置Allocator描述信息。
func	输入	根据建议地址申请内存block的回调函数。 回调函数定义如下： typedef void *(*aclrtAllocatorAllocAdviseFunc)(aclrtAllocator allocator, size_t size, aclrtAllocatorAddr addr); aclrtAllocator、aclrtAllocatorAddr的定义如下： typedef void *aclrtAllocator; typedef void *aclrtAllocatorAddr;

返回值说明

返回0表示成功，返回其他值表示失败。

1.13.108.6 aclrtAllocatorSetFreeFuncToDesc

函数功能

使用用户提供的Allocator场景下，设置"释放内存block"的回调函数。

函数原型

aclError aclrtAllocatorSetFreeFuncToDesc(**aclrtAllocatorDesc** allocatorDesc, **aclrtAllocatorFreeFunc** func)

参数说明

参数名	输入/输出	说明
allocatorDesc	输入	Allocator描述符指针。 需提前调用 aclrtAllocatorCreateDesc 接口设置Allocator描述信息。
func	输入	释放内存block的回调函数。 回调函数定义如下： typedef void (*aclrtAllocatorFreeFunc)(aclrtAllocator allocator, aclrtAllocatorBlock block); aclrtAllocator、aclrtAllocatorBlock的定义如下： typedef void *aclrtAllocator; typedef void *aclrtAllocatorBlock;

返回值说明

返回0表示成功，返回其他值表示失败。

1.13.108.7 aclrtAllocatorSetGetAddrFromBlockFuncToDesc

函数功能

使用用户提供的Allocator场景下，设置"根据申请来的block获取device内存地址"的回调函数。

函数原型

aclError aclrtAllocatorSetGetAddrFromBlockFuncToDesc(**aclrtAllocatorDesc** allocatorDesc, **aclrtAllocatorGetAddrFromBlockFunc** func)

参数说明

参数名	输入/输出	说明
allocatorDesc	输入	Allocator描述符指针。 需提前调用 aclrtAllocatorCreateDesc 接口设置Allocator描述信息。
func	输入	根据申请来的block获取device内存地址的回调函数。 回调函数定义如下： typedef void *(*aclrtAllocatorGetAddrFromBlockFunc)(aclrtAllocatorBlock block); aclrtAllocatorBlock的定义如下： typedef void *aclrtAllocatorBlock;

返回值说明

返回0表示成功，返回其他值表示失败。

2 AscendCL API (Python)

- [2.1 废弃接口/返回码列表](#)
- [2.2 同步&异步API说明](#)
- [2.3 系统配置](#)
- [2.4 公共接口](#)
- [2.5 日志管理](#)
- [2.6 Util模块](#)
- [2.7 Runtime \(rt \)](#)
- [2.8 模型加载与执行 \(mdl \)](#)
- [2.9 媒体数据处理V1](#)
- [2.10 媒体数据处理V2](#)
- [2.11 算子加载与执行 \(op \)](#)
- [2.12 CBLAS接口 \(blas \)](#)
- [2.13 算子编译](#)
- [2.14 特征向量检索](#)
- [2.15 Profiling数据采集](#)
- [2.16 数据类型及其操作接口](#)

2.1 废弃接口/返回码列表

接口

- [acl.op.execute](#)接口
此接口后续版本会废弃，请使用[acl.op.execute_v2](#)接口。
- [acl.get_tensor_desc_dim](#)接口
此接口后续版本会废弃，请使用[acl.get_tensor_desc_dim_v2](#)接口。

- [acl.get_data_buffer_size](#)接口
此接口后续版本会废弃，请使用[acl.get_data_buffer_size_v2](#)接口。
- [acl.set_tensor_storage_shape](#)接口
此接口后续版本会废弃，请使用[acl.set_tensor_shape](#)接口。
- [acl.set_tensor_storage_format](#)接口
此接口后续版本会废弃，请使用[acl.set_tensor_format](#)接口。
- [acl.rt.query_event](#)接口
此接口后续版本会废弃，请使用[acl.rt.query_event_status](#)接口。
- [acl.util.numpy_to_ptr](#)接口
此接口后续版本会废弃，请使用[acl.util.bytes_to_ptr](#)接口。
- [acl.util.numpy_contiguous_to_ptr](#)接口
此接口后续版本会废弃，请使用[acl.util.bytes_to_ptr](#)接口。
- [acl.util.ptr_to_numpy](#)接口
此接口后续版本会废弃，请使用[acl.util.ptr_to_bytes](#)接口。
- [acl.himpi.vpc_calc_hist](#)接口
此接口后续版本会废弃，请使用[2.10.4.16 acl.himpi.vpc_calc_hist_v2](#)接口。

返回码

- [ACL_ERROR_NONE](#)
此返回码后续版本会废弃，请使用[ACL_SUCCESS](#)返回码。
- [ACL_ERROR_NOT_STATIC_AIPP](#)
此返回码后续版本会废弃，请使用[ACL_ERROR_GE_AIPP_NOT_EXIST](#)返回码。
- [ACL_ERROR_STREAM_NOT_SUBSCRIBE](#)
此返回码后续版本会废弃，请使用[ACL_ERROR_RT_STREAM_NO_CB_REG](#)返回码。
- [ACL_ERROR_THREAD_NOT_SUBSCRIBE](#)
此返回码后续版本会废弃，请使用[ACL_ERROR_RT_THREAD_SUBSCRIBE](#)返回码。
- [ACL_ERROR_WAIT_CALLBACK_TIMEOUT](#)
此返回码后续版本会废弃，请使用[ACL_ERROR_RT_REPORT_TIMEOUT](#)返回码。
- [ACL_ERROR_INVALID_DEVICE](#)
此返回码后续版本会废弃，请使用[ACL_ERROR_RT_INVALID_DEVICEID](#)返回码。
- [ACL_ERROR_GROUP_NOT_SET](#)
此返回码后续版本会废弃，请使用[ACL_ERROR_RT_GROUP_NOT_SET](#)返回码。
- [ACL_ERROR_GROUP_NOT_CREATE](#)
此返回码后续版本会废弃，请使用[ACL_ERROR_RT_GROUP_NOT_CREATE](#)返回码。

2.2 同步&异步 API 说明

CANN支持以下几类显式同步，调用此类接口后，主机线程会阻塞直到相关的任务执行完成。

- **设备同步：acl.rt.synchronize_device**
阻塞当前主机线程直到Device上所有显式或隐式创建的Stream都完成所有先前下发的任务。应该尽量少使用该函数，以免拖延主机运行。
- **流同步：acl.rt.synchronize_stream**
阻塞当前主机线程直到指定的Stream中完成所有下发的任务。
- **事件同步：acl.rt.synchronize_event**
阻塞当前主机线程直到指定的Event事件完成。属于更细粒度的同步。

对于异步接口，主机线程调用异步接口后仅代表下发任务，在任务未完成前，函数就已经返回给主机线程。用户需要调用以上显式同步接口阻塞主机线程，等待任务完成。

2.3 系统配置

2.3.1 函数：init

C函数原型	aclError aclInit(const char *configPath)
Python函数	ret = acl.init(config_path)
函数功能	pyACL初始化函数。

输入说明	config_path: str, 配置文件所在的路径, 包含文件名。 配置文件内容为JSON格式 (JSON文件内的“{”的层级最多为10, “[”的层级最多为10)。如果以下的默认配置已满足需求, 无需修改, 可直接调用acl.init接口不传入参数或者可将配置文件配置为空JSON串 (即配置文件中只有{})。 配置文件格式为JSON格式, 当前支持以下配置: • Dump信息配置, 包括以下配置。 - 模型Dump配置 (用于导出模型中每一层算子输入和输出数据)、单算子Dump配置 (用于导出单个算子的输入和输出数据), 导出的数据用于与指定模型或算子进行比对, 定位精度问题, 配置示例、说明及约束请参见配置文件示例 (模型Dump配置、单算子Dump配置)。默认不启用该dump配置。 - 异常算子Dump配置 (用于导出异常算子的输入输出数据、workspace信息、Tiling信息), 导出的数据用于分析AI Core Error问题, 配置示例请参见配置文件示例 (异常算子Dump配置)。默认不启用该dump配置。 - 溢出算子Dump配置 (用于导出模型中溢出算子的输入和输出数据), 导出的数据用于分析溢出原因, 定位模型精度的问题, 配置示例、说明及约束请参见配置文件示例 (溢出算子Dump配置)。默认不启用该dump配置。 • Profiling采集信息配置, 示例、配置说明及约束请参见《性能调优工具》。默认不启用Profiling采集信息配置。 • 算子缓存信息老化配置, 为节约内存和平衡调用性能, 可通过“max_opqueue_num”参数配置“算子类型 - 单算子模型”映射队列的最大长度, 如果长度达到最大, 则会先删除长期未使用的映射信息以及缓存中的单算子模型, 再加载最新的映射信息以及对应的单算子模型。如果不配置映射队列的最大长度, 则默认最大长度为“20000”。示例及约束说明请参见配置文件示例 (算子缓存信息老化配置)。 • 错误信息上报模式配置, 用于控制acl.get_recent_err_msg接口按进程或线程级别获取错误信息, 默认按线程级别。示例请参见配置文件示例 (错误信息上报模式配置)。 建议不要同时配置dump信息和Profiling采集信息, 否则dump操作会影响系统性能, 导致Profiling采集的性能数据指标不准确。
返回值说明	ret: int, 错误码。 • 返回0表示成功。 • 返回其它值表示失败。
约束说明	• 一个进程内只能调用一次acl.init接口, 且与acl.finalize去初始化接口配对使用。。 • 使用pyACL接口开发应用时, 必须先调用acl.init接口, 否则可能会导致后续系统内部资源初始化出错, 进而导致其它业务异常。

相关接口	pyACL还提供了其它使能Dump或Profiling的接口，如下，与acl.init不同的是，以下这些接口相对灵活，可以在一个进程内调用多次接口，每次调用接口时可以基于不同的Dump配置或Profiling配置。 - 获取Dump数据，参见acl.mdl.init_dump、acl.mdl.set_dump、acl.mdl.finalize_dump。 - 获取Profiling数据，参见2.15 Profiling数据采集。
参考资源	接口调用示例，参见pyACL初始化与去初始化。

配置文件示例（模型 Dump 配置、单算子 Dump 配置）

模型Dump配置示例如下：

```
{
  "dump":{
    "dump_list":[
      { "model_name":"ResNet-101"
      },
      {
        "model_name":"ResNet-50",
        "layer":[
          "conv1 conv1_relu",
          "res2a_branch2ares2a_branch2a_relu",
          "res2a_branch1",
          "pool1"
        ]
      }
    ],
    "dump_path": "$HOME/output",
    "dump_mode": "output",
    "dump_op_switch": "off",
    "dump_data": "tensor"
  }
}
```

单算子Dump配置示例如下：

```
{
  "dump":{
    "dump_path": "output",
    "dump_list": [],
    "dump_op_switch": "on",
    "dump_data": "tensor"
  }
}
```

表 2-1 acl.json 文件格式说明

配置项	参数说明	备注
dump	-	- 不具有输出的TBE算子、AI CPU算子，如StreamActive、Send、Recv、const等不会生成dump数据；编译后的模型中部分算子并不会在AI CPU或AI Core执行，如concatD类型算子，则无法生成dump数据。 - 采用dump部分算子场景下，因data算子不会在AI CPU或AI Core上执行，如果用户填写dump data节点算子时需要一并填写data节点算子的后继节点，才能dump出data节点算子数据。
dump_list	（必填）待dump数据的整网模型列表。 创建模型dump配置信息，当存在多个模型需要dump时，需要每个模型之间用英文逗号隔开。 在单算子执行场景下，dump_list为空。	-
model_name	模型名称。 - 模型加载方式为文件加载时，填入模型文件的名称，不需要带后缀名；也可以配置为ATC模型文件转换后的json文件里的最外层"name"字段对应值。 - 模型加载方式为内存加载时，配置为ATC模型文件转换后的json文件里的最外层"name"字段对应值。	- 各个模型的model_name值须唯一。 - 模型加载支持内存和文件两种加载方式，文件加载且配置为ATC模型文件转换后的json文件里的最外层"name"字段对应值时，如果acl.json文件里model_name配置项值同时包括模型文件名、本处获取的name值，以模型文件名的配置项生效。

配置项	参数说明	备注
layer	算子名。可以指定为ATC模型转换后的算子名，也支持指定为转换前的原始算子名。 - 当需要dump指定的部分算子时，按格式配置layer字段，每行配置模型中的一个算子名，且每个算子之间用英文逗号隔开。若指定的算子其输入涉及data算子，会同时将data算子信息dump出来。 - 当需要dump模型的所有算子时，不需要包含layer字段。 在指定算子dump时，用户可以无需设置model_name，此时会默认dump所有model下的相应算子。如果配置了model_name，则dump对应model下的相应算子。	IO性能相对较差时，可能会出现由于数据量过大导致执行超时，所以不建议全量dump，请指定算子进行dump。
dump_path	（必填）dump数据文件存储到运行环境的目录。 支持配置绝对路径或相对路径： - 绝对路径配置以“/”开头，例如：\$HOME/output。 - 相对路径配置直接以目录名开始，例如：output。	该参数指定的目录需要提前创建且确保安装时配置的运行用户具有读写权限。
dump_mode	dump数据模式。 - input: dump算子的输入数据。 - output: dump算子的输出数据，默认取值output。 - all: dump算子的输入、输出数据。	-
dump_op_switch	单算子模型dump数据开关。 - on: 开启单算子模型dump。 - off: 关闭单算子模型dump，默认取值off。	-

配置项	参数说明	备注
dump_step	指定采集哪些迭代的Dump数据。推理场景无需配置。 不配置该参数，默认所有迭代都会产生dump数据，数据量比较大，建议按需指定迭代。 多个迭代用“ ”分割，例如：0 5 10；也可以用“-”指定迭代范围，例如：0 3-5 10。 配置示例： <pre>{ "dump":{ "dump_list": [.....], "dump_path":"\$HOME/output", "dump_mode":"output", "dump_op_switch":"off" "dump_step": "0 3-5 10" } }</pre>	-
dump_data	指定算子dump内容类型，取值： • tensor: dump算子数据，默认为tensor。 • stats: dump算子统计数据，结果文件为csv格式，文件中包含算子名称、输入/输出的数据类型、最大值、最小值等。 通常dump数据量太大并且耗时长，可以先dump算子统计数据，根据统计数据识别可能异常的算子，然后再dump算子数据。 模型Dump场景下，会根据dump_mode处的配置统计算子输入或算子输出或算子输入&输出的信息。	-

配置文件示例（异常算子 Dump 配置）

将“dump_scene”参数设置为“lite_exception”，启用异常算子Dump功能，同时可配合使用“ASCEND_WORK_PATH”环境变量配置落盘路径，否则落盘在应用程序的当前执行目录下。

配置文件中的示例内容如下：

```
{
  "dump":{
    "dump_scene":"lite_exception"
  }
}
```

说明

异常算子Dump配置，不能与模型Dump配置或单算子Dump配置同时开启，否则模型Dump或单算子Dump不生效。

配置文件示例（溢出算子 Dump 配置）

溢出算子Dump配置的相关约束说明如下：

- 将“dump_debug”配置为“on”表示开启溢出算子配置，不配置“dump_debug”或将“dump_debug”配置为“off”表示不开启溢出算子配置。
- 若开启溢出算子配置，则“dump_path”必须配置，表示导出数据文件的存储路径。

支持配置绝对路径或相对路径：

- 绝对路径配置以“/”开头，例如：/home。
- 相对路径配置直接以目录名开始，例如：output。

获取导出的数据文件后，如何解析请参见溢出算子数据采集及分析。

- 溢出算子Dump配置，不能与模型Dump配置或单算子Dump配置同时开启，否则会返回报错。

配置文件中的示例内容如下：

```
{
  "dump":{
    "dump_path":"output",
    "dump_debug":"on"
  }
}
```

配置文件示例（算子缓存信息老化配置）

算子缓存信息老化配置的相关约束说明如下：

- 对于静态加载的算子，调用[acl.op.set_model_dir](#)接口加载指定目录下的单算子模型或调用[acl.op.load](#)接口加载指定单算子模型时，老化配置无效，不会对该部分的算子信息做老化。
- pyACL内部分开维护固定Shape和动态Shape算子的映射队列，最大长度都为“max_opqueue_num”参数值。
- “max_opqueue_num”参数值为静态加载算子的单算子模型个数和在线编译算子的单算子模型个数的总和，因此“max_opqueue_num”参数值应大于当前进程中可用的、静态加载算子的单算子模型个数，否则会导致在线编译算子的信息无法老化。

配置文件中的示例内容如下：

```
{
  "max_opqueue_num": "10000"
}
```

配置文件示例（错误信息上报模式配置）

“err_msg_mode” 参数取值范围：“0” 为默认值，表示按线程级别获取错误信息；
“1” 表示按进程级别获取错误信息。

配置文件中的示例内容如下：

```
{  
    "err_msg_mode": "1"  
}
```

2.3.2 函数：finalize

C函数原型	aclError aclFinalize()
Python函数	ret = acl.finalize()
函数功能	pyACL去初始化函数，用于释放进程内的pyACL相关资源。 在该接口内，默认增加2000ms延时（实际最大延时可达2000ms），用于Device业务日志回传，保证ERROR级别和EVENT级别日志不丢失，您可以将 “ASCEND_LOG_DEVICE_FLUSH_TIMEOUT” 环境变量设置为 “0”（命令示例：export ASCEND_LOG_DEVICE_FLUSH_TIMEOUT=0），去除该默认延时。 关于ASCEND_LOG_DEVICE_FLUSH_TIMEOUT环境变量的详细描述请参见《日志参考》。
输入说明	无
返回值说明	ret : int，错误码。 - 返回0表示成功。 - 返回其它值表示失败。
约束说明	应用的进程退出前，应显式调用该接口实现pyACL去初始化，否则可能会导致异常，例如应用进程退出时有异常报错。 不建议在析构函数中调用acl.finalize接口，否则在进程退出时可能由于单例析构顺序未知而导致进程异常退出的问题。
参考资源	接口调用示例，参见pyACL初始化与去初始化。

2.3.3 函数：get_version

C函数原型	aclError aclrtGetVersion(int32_t *majorVersion, int32_t *minorVersion, int32_t *patchVersion)
Python函数	major_version, minor_version, patch_version, ret = acl.get_version()
函数功能	查询接口版本号，pyACL接口版本号命名可以采用：A.B.C模式，其中，A表示有不兼容修改，B表示新增接口，C表示bug修复。

输入说明	无
返回值说明	major_version: int, 主版本号, 从1开始, 如果出现接口的不兼容变更时, 加1。 minor_version: int, 次版本号, 从0开始, 按照迭代周期, 有新增接口时加1。 patch_version: int, 补丁版本号, 从0开始, 表示本版本仅仅解决了问题, 在“majorVersion”、“minorVersion”不变的情况下加1。但“majorVersion”、“minorVersion”增加的时候, “patchVersion”一般为“0”。 ret: int, 错误码。 • 返回0表示成功。 • 返回其它值表示失败。
约束说明	无

2.3.4 函数: get_recent_err_msg

C函数原型	const char *aclGetRecentErrMsg()
Python函数	msg = acl.get_recent_err_msg()
函数功能	获取并清空与本接口在同一个线程中的其它pyACL接口调用失败时的错误描述信息。 获取进程级别、还是线程级别的错误描述信息由acl.init接口中的“err_msg_mode”配置控制, 默认线程级别。
输入说明	无
返回值说明	msg: str, 返回错误描述字符串。 如果通过本接口获取到多条错误描述信息, 最上面的错误描述信息为最新的。 获取错误描述信息失败时, 返回None。
约束说明	• 建议在每次调用pyACL接口失败时都调用acl.get_recent_err_msg接口, 以便获取调用pyACL接口异常时的错误描述信息, 用于定位问题。 • 同一个线程中多次调用acl.get_recent_err_msg接口后, 只有最后一次调用acl.get_recent_err_msg接口返回的错误描述字符串有效, 之前acl.get_recent_err_msg接口返回的错误描述字符串不能使用, 否则可能导致内存非法访问。 • 如果未在每次调用pyACL接口失败时调用acl.get_recent_err_msg接口可能导致错误信息堆积、丢失。

2.3.5 函数：get_cann_attribute_list

C函数原型	<code>aclError aclGetCannAttributeList(const aclCannAttr **cannAttrList, size_t *num)</code>
Python函数	<code>cann_attr_list, num, ret = acl.get_cann_attribute_list()</code>
函数功能	查询运行环境中CANN软件和对对应芯片支持的特性列表。
输入说明	无
返回值说明	cann_attr_list : list, 用于保存运行环境支持的特性枚举列表, 具体请参见 2.16.42 aclCannAttr 。 用户无需提前申请内存, 应用进程退出时, 内存自动释放。 num : int, 用于保存支持的特性数量, 与“cann_attr_list”列表长度保持一致。 ret : int, 错误码。 • 返回0表示成功。 • 返回其它值表示失败。
约束说明	无

2.3.6 函数：get_cann_attribute

C函数原型	<code>aclError aclGetCannAttribute(aclCannAttr cannAttr, int32_t *value)</code>
Python函数	<code>value, ret = acl.get_cann_attribute(cann_attr)</code>
函数功能	查询运行环境是否支持指定的CANN特性。
输入说明	cann_attr : int, 特性列表枚举值, 一次查询可指定其中一项, 具体请参见 2.16.42 aclCannAttr 。
返回值说明	value : int, 是否支持。 • 1: 支持 • 0: 不支持 ret : int, 错误码。 • 返回0表示成功。 • 返回其它值表示失败。
约束说明	无

2.3.7 函数：get_device_capability

C函数原型	<code>aclError aclGetDeviceCapability(uint32_t deviceId, aclDeviceInfo deviceInfo, int64_t *value)</code>
Python函数	<code>value, ret = acl.get_device_capability(device_id, device_info)</code>
函数功能	查询运行环境中对应Device的硬件规格大小。
输入说明	device_id : int, Device ID。 用户调用 acl.rt.get_device_count 接口获取可用的Device数量后, 这个Device ID的取值范围: [0, (可用的Device数量 - 1)]。 device_info : int, 指定Device上的硬件规格类型, 具体请参见 2.16.41 aclDeviceInfo 。
返回值说明	value : int, 硬件规格值。 ret : int, 错误码。 - 返回0表示成功。 - 返回其它值表示失败。
约束说明	无

2.4 公共接口

2.4.1 函数：float16_to_float

C函数原型	<code>float aclFloat16ToFloat(aclFloat16 value)</code>
Python函数	<code>output = acl.float16_to_float(value)</code>
函数功能	将 aclFloat16 类型的数据转换为float (指float32) 类型的数据。
输入说明	value : int, 待转化的数据, aclFloat16 类型的数据。
返回值说明	output : float, 浮点数类型。
约束说明	无

2.4.2 函数：float_to_float16

C函数原型	<code>aclFloat16 aclFloatToFloat16(float value)</code>
Python函数	<code>output = acl.float_to_float16(value)</code>
函数功能	将float (指float32) 类型的数据转换为 aclFloat16 类型的数据。
输入说明	value : float, 待转化的数据。

返回值说明	output : int, aclFloat16 类型的数据。
约束说明	无

2.4.3 函数：create_data_buffer

C函数原型	aclDataBuffer *aclCreateDataBuffer(void *data, size_t size)
Python函数	output = acl.create_data_buffer(data, size)
函数功能	创建 aclDataBuffer 类型的数据，该数据类型用于描述内存地址、大小等内存信息。 如需销毁 aclDataBuffer 类型的数据，请参见 2.4.4 函数：destroy_data_buffer 。
输入说明	data : int, 表述存储 aclDataBuffer 数据的指针地址。“data”参数支持传入0，表示创建一个空的数据类型，此时“size”参数值必须设置为0。 该内存需由用户自行管理，调用 acl.rt.malloc 接口/ acl.rt.free 接口申请/释放内存，或调用 acl.rt.malloc_host 接口/ acl.rt.free_host 接口申请/释放内存。 size : int, 内存大小。如果用户需要使用空Tensor，则在申请内存时，内存大小最小为1Byte，以保障后续业务正常运行。
返回值说明	output : int, 输出 aclDataBuffer 的指针地址。
约束说明	无

2.4.4 函数：destroy_data_buffer

C函数原型	aclError aclDestroyDataBuffer(const aclDataBuffer *dataBuffer)
Python函数	ret = acl.destroy_data_buffer(data_buffer)
函数功能	销毁 aclDataBuffer 类型的数据。 此处仅销毁 aclDataBuffer 类型的数据，调用 acl.create_data_buffer 接口创建 aclDataBuffer 类型数据时传入的data的内存需由用户自行释放。
输入说明	data_buffer : int, 表述 aclDataBuffer 的指针地址。
返回值说明	ret : int, 错误码。 • 返回0表示成功。 • 返回其它值表示失败。
约束说明	无

2.4.5 函数：get_data_buffer_addr

C函数原型	<code>void *aclGetDataBufferAddr(const aclDataBuffer *dataBuffer)</code>
Python函数	<code>output = acl.get_data_buffer_addr(data_buffer)</code>
函数功能	获取aclDataBuffer类型中的数据的内存地址。
输入说明	data_buffer : int, 表述aclDataBuffer的指针地址。 需提前调用 acl.create_data_buffer 接口创建aclDataBuffer类型的数据。
返回值说明	output : int, aclDataBuffer类型中的数据的指针地址。
约束说明	无

2.4.6 函数：get_data_buffer_size

须知

此接口后续版本会废弃, 请使用[acl.get_data_buffer_size_v2](#)接口。

C函数原型	<code>size_t aclGetDataBufferSize(const aclDataBuffer *dataBuffer)</code>
Python函数	<code>size = acl.get_data_buffer_size(data_buffer)</code>
函数功能	获取aclDataBuffer类型中数据的内存大小, 单位Byte。
输入说明	data_buffer : int, aclDataBuffer的指针地址。 需提前调用 acl.create_data_buffer 接口创建aclDataBuffer类型的数据。
返回值说明	size : int, aclDataBuffer类型中数据的内存大小。
约束说明	无

2.4.7 函数：get_data_buffer_size_v2

C函数原型	<code>size_t aclGetDataBufferSizeV2(const aclDataBuffer *dataBuffer)</code>
Python函数	<code>size = acl.get_data_buffer_size_v2(data_buffer)</code>
函数功能	获取aclDataBuffer类型中数据的内存大小, 单位Byte。

输入说明	data_buffer : int, aclDataBuffer的指针地址。 需提前调用 acl.create_data_buffer 接口创建aclDataBuffer类型的数据。
返回值说明	size : int, aclDataBuffer类型中数据的内存大小。
约束说明	无

2.4.8 函数：update_data_buffer

C函数原型	aclError aclUpdateDataBuffer(aclDataBuffer *dataBuffer, void *data, size_t size)
Python函数	ret = acl.update_data_buffer(data_buffer, data, size)
函数功能	更新aclDataBuffer中数据的内存及大小。 更新aclDataBuffer后，之前aclDataBuffer中存放数据的内存如果不使用，需及时释放，否则可能会导致内存泄漏。
输入说明	data_buffer : int, aclDataBuffer类型的指针地址。 需提前调用 acl.create_data_buffer 接口创建aclDataBuffer类型的数据。 该内存需由用户自行管理，调用 acl.rt.malloc 接口/ acl.rt.free 接口申请/释放内存，或调用 acl.rt.malloc_host 接口/ acl.rt.free_host 接口申请/释放内存。 data : int, 存放数据内存地址的指针地址。 size : int, 内存大小，单位Byte。如需使用空Tensor，在申请内存时，内存大小最小为1Byte。
返回值说明	ret : int, 错误码。 - 返回0表示成功。 - 返回其它值表示失败。
约束说明	无

2.4.9 函数：data_type_size

C函数原型	size_t aclDataTypeSize(aclDataType dataType)
Python函数	size = acl.data_type_size(data_type)
函数功能	获取 aclDataType 数据的大小，单位Byte。
输入说明	data_type : int, 指定要获取大小的aclDataType数据。
返回值说明	size : int, aclDataType数据的大小，单位Byte。
约束说明	无

2.4.10 函数：create_tensor_desc

C函数原型	<code>aclTensorDesc *aclCreateTensorDesc(aclDataType dataType, int numDims, const int64_t *dims, aclFormat format)</code>
Python函数	<code>tensor_desc = acl.create_tensor_desc(data_type, dims, format)</code>
函数功能	创建 aclTensorDesc 类型的数据，该数据类型用于描述Tensor的数据类型、Shape、Format等信息。 如需销毁 aclTensorDesc 类型的数据，请参见 2.4.11 函数：destroy_tensor_desc 。
输入说明	data_type : int，表述Tensor描述的数据类型。 dims : int list，Tensor描述指定维度大小。“dims”是一个int的列表，列表中的每项表示Tensor中每个维度的大小，如果列表中某个元素的值为0，则为空Tensor。如果用户需要使用空Tensor，则在申请内存时，内存大小最小为1Byte，以保障后续业务正常运行。 format : int，Tensor描述的Format。
返回值说明	tensor_desc : int，创建Tensor描述的地址对象。
约束说明	无

2.4.11 函数：destroy_tensor_desc

C函数原型	<code>void aclDestroyTensorDesc(const aclTensorDesc *desc)</code>
Python函数	<code>acl.destroy_tensor_desc(desc)</code>
函数功能	销毁 aclTensorDesc 类型的数据。
输入说明	desc : int，待销毁的 aclTensorDesc 类型的指针地址。
返回值说明	无
约束说明	无

2.4.12 函数：get_tensor_desc_type

C函数原型	<code>aclDataType aclGetTensorDescType(const aclTensorDesc *desc)</code>
Python函数	<code>type = acl.get_tensor_desc_type(desc)</code>
函数功能	获取Tensor描述中的数据类型。

输入说明	desc : int, aclTensorDesc类型的指针地址。 需提前调用 acl.create_tensor_desc 接口创建aclTensorDesc类型。
返回值说明	Type : int, 返回指定Tensor描述的数据类型。
约束说明	无

2.4.13 函数: get_tensor_desc_format

C函数原型	aclFormat aclGetTensorDescFormat(const aclTensorDesc *desc)
Python函数	format = acl.get_tensor_desc_format(desc)
函数功能	获取Tensor描述中的Format。
输入说明	desc : int, aclTensorDesc类型的指针地址。 需提前调用 acl.create_tensor_desc 接口创建aclTensorDesc类型。
返回值说明	format : int, 返回指定Tensor描述的格式。
约束说明	无

2.4.14 函数: get_tensor_desc_size

C函数原型	size_t aclGetTensorDescSize(const aclTensorDesc *desc)
Python函数	size = acl.get_tensor_desc_size(desc)
函数功能	获取Tensor描述占用的空间大小。
输入说明	desc : int, aclTensorDesc类型的指针地址。 需提前调用 acl.create_tensor_desc 接口创建aclTensorDesc类型。
返回值说明	size : int, Tensor描述占用的空间大小。
约束说明	无

2.4.15 函数: get_tensor_desc_element_count

C函数原型	size_t aclGetTensorDescElementCount(const aclTensorDesc *desc)
Python函数	size = acl.get_tensor_desc_element_count(desc)
函数功能	获取Tensor描述中的元素个数。

输入说明	desc : int, aclTensorDesc类型的指针地址。 需提前调用 acl.create_tensor_desc 接口创建aclTensorDesc类型。
返回值说明	size : int, 获取Tensor描述中的元素个数。
约束说明	无

2.4.16 函数: `get_tensor_desc_num_dims`

C函数原型	<code>size_t aclGetTensorDescNumDims(const aclTensorDesc *desc)</code>
Python函数	<code>size = acl.get_tensor_desc_num_dims(desc)</code>
函数功能	获取Tensor描述中Shape的维度个数。
输入说明	desc : int, aclTensorDesc类型的指针地址。 需提前调用 acl.create_tensor_desc 接口创建aclTensorDesc类型。
返回值说明	size : int, 获取Tensor描述中Shape的维度个数。 若返回 ACL_UNKNOWN_RANK (0xFFFFFFFFFFFFFFFF) 表示动态Shape场景下维度个数未知。
约束说明	无

2.4.17 函数: `get_tensor_desc_dim`

须知 此接口后续版本会废弃, 请使用acl.get_tensor_desc_dim_v2接口。	
C函数原型	<code>int64_t aclGetTensorDescDim(const aclTensorDesc *desc, size_t index)</code>
Python函数	<code>num = acl.get_tensor_desc_dim(desc, index)</code>
函数功能	获取Tensor描述中指定维度的大小。
输入说明	desc : int, aclTensorDesc类型的指针地址。 需提前调用 acl.create_tensor_desc 接口创建aclTensorDesc类型。 index : int, 指定获取第几个维度的大小, “index” 值从0开始。 用户调用 acl.get_tensor_desc_dim 接口获取Shape维度个数, “index” 的取值范围: [0, (<i>shape</i> 维度个数-1)]。
返回值说明	num : int, 获取Tensor描述中指定维度的大小。
约束说明	无

2.4.18 函数：get_tensor_desc_dim_v2

C函数原型	<code>aclError aclGetTensorDescDimV2(const aclTensorDesc *desc, size_t index, int64_t *dimSize)</code>
Python函数	<code>dim_size, ret = acl.get_tensor_desc_dim_v2(desc, index)</code>
函数功能	获取Tensor描述中指定维度的大小。
输入说明	desc : int, aclTensorDesc类型的指针地址。 需提前调用 acl.create_tensor_desc 接口创建aclTensorDesc类型。 index : int, 指定获取第几个维度的大小, “index” 值从0开始。 用户调用 acl.get_tensor_desc_num_dims 接口获取Shape维度个数, “index” 的取值范围: [0, (shape维度个数-1)]。
返回值说明	dim_size : int, Tensor描述中 “index” 指定维度的大小。当返回的 “dim_size” 参数值为 “-1” 时, 表示维度的大小是动态的。 ret : int, 错误码。 • 返回0表示成功。 • 返回其它值表示失败。
约束说明	当 acl.get_tensor_desc_num_dims 接口的返回值为 ACL_UNKNOWN_RANK (0xFFFFFFFFFFFFFFFE) 时, 表示动态Shape场景下维度个数未知, 则不能调用 <code>acl.get_tensor_desc_dim_v2</code> 接口获取指定维度的范围。

2.4.19 函数：get_tensor_desc_dim_range

C函数原型	<code>aclError aclGetTensorDescDimRange(const aclTensorDesc *desc, size_t index, size_t dimRangeNum, int64_t *dimRange)</code>
Python函数	<code>dim_range, ret = acl.get_tensor_desc_dim_range(desc, index, dim_range_num)</code>
函数功能	获取Tensor描述中指定维度的范围, [1, -1]表示全Shape范围。
输入说明	desc : int, aclTensorDesc类型的指针地址。 需提前调用 acl.create_tensor_desc 接口创建aclTensorDesc类型。 index : int, 指定获取第几个维度的大小, “index” 值从0开始。 用户调用 acl.get_tensor_desc_num_dims 接口获取Shape维度个数, 该 “index” 的取值范围: [0, (shape维度个数-1)]。 dim_range_num : int, “dim_range” 的长度, 该值必须大于或等于2。

返回值说明	dim_range: list, Tensor描述中“index”指定维度的Shape范围。 - 第一个元素值表示Shape范围的最小值，第二个元素值表示Shape范围的最大值。该数组中仅前2个元素值有效。 - 当“dim_range”数组的值为[1, -1]时，表示全Shape范围。 ret: int, 错误码。 - 返回0表示成功。 - 返回其它值表示失败。
约束说明	当 acl.get_tensor_desc_num_dims 接口的返回值为 ACL_UNKNOWN_RANK (0xFFFFFFFFFFFFFFFE) 时，表示动态Shape场景下维度个数未知，则不能调用 acl.get_tensor_desc_dim_range 接口获取指定维度的范围。

2.4.20 函数：set_tensor_desc_name

C函数原型	void aclSetTensorDescName(aclTensorDesc *desc, const char *name)
Python函数	acl.set_tensor_desc_name(desc, name)
函数功能	设置TensorDesc的名称。
输入说明	desc: int, aclTensorDesc类型的指针地址。 需提前调用acl.create_tensor_desc接口创建aclTensorDesc类型。 name: str, Tensor描述的名称。
返回值说明	无
约束说明	无

2.4.21 函数：get_tensor_desc_name

C函数原型	const char * aclGetTensorDescName(aclTensorDesc *desc)
Python函数	name = acl.get_tensor_desc_name(desc)
函数功能	获取Tensor描述的名称。
输入说明	desc: int, aclTensorDesc类型的指针地址。 需提前调用acl.create_tensor_desc接口创建aclTensorDesc类型。
返回值说明	name : str, Tensor描述的名称。
约束说明	无

2.4.22 函数：trans_tensor_desc_format

C函数原型	<code>aclError aclTransTensorDescFormat(const aclTensorDesc *srcDesc, aclFormat dstFormat, aclTensorDesc **dstDesc)</code>
Python函数	<code>dstDesc, ret = acl.trans_tensor_desc_format(srcDesc, dstFormat)</code>
函数功能	根据指定的“dstFormat”转换“srcDesc”（输入源aclTensorDesc中）的Format，生成新的目标aclTensorDesc，源aclTensorDesc中的Format保持不变。当前版本不支持该接口，预留接口。
输入说明	srcDesc : int，输入源aclTensorDesc数据的指针地址。需提前调用 2.4.10 函数：create_tensor_desc 接口创建aclTensorDesc类型。 dstFormat : int，需要设置的目标Format，具体请参见 2.16.4 aclFormat 。
返回值说明	dstDesc : int，输出aclTensorDesc的指针地址。需提前调用 2.4.10 函数：create_tensor_desc 接口创建aclTensorDesc类型。 ret : int，错误码。 - 返回0表示成功。 - 返回其它值表示失败。
约束说明	无

2.4.23 函数：set_tensor_storage_shape

须知 此接口后续版本会废弃，请使用acl.set_tensor_shape接口。	
C函数原型	<code>aclError aclSetTensorStorageShape(aclTensorDesc *desc, int numDims, const int64_t *dims)</code>
Python函数	<code>ret = acl.set_tensor_storage_shape(desc, dims)</code>
函数功能	调用 acl.create_tensor_desc 接口创建Tensor描述信息后，可通过 acl.set_tensor_storage_shape 设置Tensor的实际“dims”信息，当前主要用于PyTorch场景下。 Atlas 训练系列产品 ，不支持该接口。
输入说明	desc : int，aclTensorDesc类型的指针地址。 需提前调用 acl.create_tensor_desc 接口创建aclTensorDesc类型。 dims : list，要设置的“dims”维度数。

返回值说明	ret : int, 错误码。 - 返回0表示成功。 - 返回其它值表示失败。
约束说明	无

2.4.24 函数：set_tensor_storage_format

须知 此接口后续版本会废弃，请使用 acl.set_tensor_format 接口。	
C函数原型	aclError aclSetTensorStorageFormat(aclTensorDesc *desc, aclFormat format)
Python函数	ret = acl.set_tensor_storage_format(desc, format)
函数功能	调用 acl.create_tensor_desc 接口创建Tensor描述信息后，可通过 acl.set_tensor_storage_format 设置Tensor的实际Format信息，当前主要用于PyTorch场景下。 Atlas 训练系列产品 ，不支持该接口。
输入说明	desc : int, aclTensorDesc类型的指针地址。 需提前调用 acl.create_tensor_desc 接口创建aclTensorDesc类型。 format : int, 指定Tensor描述的格式。
返回值说明	ret : int, 错误码。 - 返回0表示成功。 - 返回其它值表示失败。
约束说明	无

2.4.25 函数：set_tensor_origin_shape

C函数原型	aclError aclSetTensorOriginShape(aclTensorDesc *desc, int numDims, const int64_t *dims)
Python函数	ret = acl.set_tensor_origin_shape(desc, dims)

函数功能	调用 acl.create_tensor_desc 接口创建Tensor描述信息后，可通过 acl.set_tensor_origin_shape 设置Tensor的原始维度信息，当前主要用于PyTorch场景下。 说明 例如，某算子的原始Format为NHWC（原始Shape为4维）或者NDHWC（原始Shape为5维），为了提高数据访问效率，在进行模型转换或者在线构建网络时会自动转换Format，将原始数据Format转化为昇腾AI处理器内部计算数据时用的Format（NC1HWC0或者NDC1HWC0），同时进行推导Shape用于内部计算。
输入说明	desc : int，Tensor描述数据的指针地址。 需提前调用 acl.create_tensor_desc 接口创建aclTensorDesc类型。 dims : list，待设置的维度。
返回值说明	ret : int，错误码。 • 返回0表示成功。 • 返回其它值表示失败。
约束说明	无

2.4.26 函数：set_tensor_origin_format

C函数原型	aclError aclSetTensorOriginFormat(aclTensorDesc *desc, aclFormat format)
Python函数	ret = acl.set_tensor_origin_format(desc, format)
函数功能	调用 acl.create_tensor_desc 接口创建Tensor描述信息后，可通过 acl.set_tensor_origin_format 设置Tensor的原始Format信息，当前主要用于PyTorch场景下。 说明 例如，某算子的原始Format为NHWC（原始Shape为4维）或者NDHWC（原始Shape为5维），为了提高数据访问效率，在进行模型转换或者在线构建网络时会自动转换Format，将原始数据Format转化为昇腾AI处理器内部计算数据时用的Format（NC1HWC0或者NDC1HWC0），同时进行推导Shape用于内部计算。
输入说明	desc : int，Tensor描述数据的指针地址。 需提前调用 acl.create_tensor_desc 接口创建aclTensorDesc类型。 format : int，指定Tensor描述的格式。
返回值说明	ret : int，错误码。 • 返回0表示成功。 • 返回其它值表示失败。
约束说明	无

2.4.27 函数：set_tensor_shape

C函数原型	<code>aclError aclSetTensorShape(aclTensorDesc *desc, int numDims, const int64_t *dims)</code>
Python函数	<code>ret = acl.set_tensor_shape(desc, dims)</code>
函数功能	调用acl.create_tensor_desc接口创建Tensor描述信息后，可通过acl.set_tensor_shape设置昇腾AI处理器内部处理Tensor时的维度信息，当前主要用于PyTorch场景下。 Atlas 训练系列产品，不支持该接口。 说明 例如，某算子的原始Format为NHWC（原始Shape为4维）或者NDHWC（原始Shape为5维），为了提高数据访问效率，在进行模型转换或者在线构建网络时会自动转换Format，将原始数据Format转化为昇腾AI处理器内部计算数据时用的Format（NC1HWC0或者NDC1HWC0），同时进行推导Shape用于内部计算。
输入说明	desc: int，Tensor描述数据的指针地址。 需提前调用acl.create_tensor_desc接口创建aclTensorDesc类型。 dims: list，要设置的维度。
返回值说明	ret: int，错误码。 • 返回0表示成功。 • 返回其它值表示失败。
约束说明	无

2.4.28 函数：set_tensor_format

C函数原型	<code>aclError aclSetTensorFormat(aclTensorDesc *desc, aclFormat format)</code>
Python函数	<code>ret = acl.set_tensor_format(desc, format)</code>
函数功能	调用acl.create_tensor_desc接口创建Tensor描述信息后，可通过acl.set_tensor_format设置昇腾AI处理器内部处理Tensor时的Format信息，当前主要用于PyTorch场景下。 Atlas 训练系列产品，不支持该接口。 说明 例如，某算子的原始Format为NHWC（原始Shape为4维）或者NDHWC（原始Shape为5维），为了提高数据访问效率，在进行模型转换或者在线构建网络时会自动转换Format，将原始数据Format转化为昇腾AI处理器内部计算数据时用的Format（NC1HWC0或者NDC1HWC0），同时进行推导Shape用于内部计算。
输入说明	desc: int，Tensor描述数据的指针地址。 需提前调用acl.create_tensor_desc接口创建aclTensorDesc类型。 format: int，指定Tensor描述的格式。

返回值说明	ret : int, 错误码。 - 返回0表示成功。 - 返回其它值表示失败。
约束说明	无

2.4.29 函数: set_tensor_shape_range

C函数原型	aclError aclSetTensorShapeRange(aclTensorDesc* desc, size_t dimsCount, int64_t dimsRange[] [ACL_TENSOR_SHAPE_RANGE_NUM])
Python函数	ret = acl.set_tensor_shape_range(desc, dims_range)
函数功能	调用 acl.create_tensor_desc 接口创建Tensor描述信息后, 可通过本接口设置Tensor的各个维度的取值范围。 使用场景: 动态Shape的算子, 其输入Shape中变化维度用“-1”表示, 但每个变化维度的范围是不一样的, 需要显式设置。 如Shape为[16, -1, 20, -1], 对应的shape range可以是[[16, 16], [1, 128], [20, 20], [1, 10]], 表示第一维的Shape范围固定为16, 第二维的Shape范围为1到128, 第三维的Shape范围固定为20, 第四维的Shape范围为1到10。
输入说明	desc : int, Tensor描述数据的指针地址。 需提前调用 acl.create_tensor_desc 接口创建aclTensorDesc类型。 dims_range : list, 用二维list表示每个维度的取值范围。
返回值说明	ret : int, 错误码。 - 返回0表示成功。 - 返回其它值表示失败。
约束说明	无

2.4.30 函数: set_tensor_dynamic_input

C函数原型	aclError aclSetTensorDynamicInput(aclTensorDesc *desc, const char *dynamicInputName)
Python函数	ret = acl.set_tensor_dynamic_input(desc, dynamic_input_name)
函数功能	调用 acl.create_tensor_desc 接口创建Tensor描述信息后, 可通过 acl.set_tensor_dynamic_input 接口设置算子动态输入的动态输入属性的名称。

输入说明	desc : int, Tensor描述数据的指针地址。 需提前调用 acl.create_tensor_desc 接口创建aclTensorDesc类型。 dynamic_input_name : int, 设置“desc”中的动态输入属性的名称。
返回值说明	ret : int, 错误码。 - 返回0表示成功。 - 返回其它值表示失败。
约束说明	无

2.4.31 函数：get_tensor_desc_by_index

C函数原型	<code>aclTensorDesc *aclGetTensorDescByIndex(aclTensorDesc *desc, size_t index)</code>
Python函数	<code>tensor_desc = acl.get_tensor_desc_by_index(desc, index)</code>
函数功能	获取算子的指定Index的Tensor描述信息。
输入说明	desc : int, Tensor描述数据的指针地址。 调用 acl.mdl.create_and_get_op_desc 接口, 获取算子所有输入/输出的Tensor描述作为本接口的输入。 index : int, 指定获取第几个维度的大小, “index” 值从0开始。 调用 acl.mdl.create_and_get_op_desc 接口获取的算子输入/输出的数量后, 该“index”的取值范围: [0, (算子输入/输出的数量 - 1)]。
返回值说明	tensor_desc : 返回指定输入/输出的Tensor描述信息的指针地址。
约束说明	无

2.4.32 函数：get_tensor_desc_address

C函数原型	<code>void *aclGetTensorDescAddress(const aclTensorDesc *desc)</code>
Python函数	<code>addr = acl.get_tensor_desc_address(desc)</code>
函数功能	获取指定算子输入/输出的Tensor数据的指针地址。
输入说明	desc : int, Tensor描述信息的指针地址。 调用 acl.get_tensor_desc_by_index 接口获取算子所有输入/输出的Tensor描述, 作为本接口的输入。
返回值说明	addr : int, 返回指定算子输入/输出的Tensor数据的指针地址。
约束说明	无

2.4.33 函数：set_tensor_const

C函数原型	aclError aclSetTensorConst(aclTensorDesc *desc, void *dataBuffer, size_t length)
Python函数	ret = acl.set_tensor_const(desc, dataBuffer, length)
函数功能	为算子设置constant输入，指定存放输入Tensor数据的内存地址和地址长度。
输入说明	desc : int, aclTensorDesc类型的指针地址。 需提前调用 acl.create_tensor_desc 接口创建aclTensorDesc类型。 dataBuffer : int, 存放constant输入Tensor数据的内存地址。 此处算子输入Tensor数据的内存必须根据应用运行模式来确定。内存申请接口请参见 2.7.6 内存管理 。 - 应用运行在Host时，此处需申请Host上的内存。 - 应用运行在Device时，此处需申请Device上的内存。 length : int, 内存地址的长度，单位是Byte。
返回值说明	ret : int, 错误码。 - 返回0表示成功。 - 返回其它值表示失败。
约束说明	无

2.4.34 函数：set_tensor_place_ment

C函数原型	aclError aclSetTensorPlaceMent(aclTensorDesc *desc, aclMemType memType)
Python函数	ret = acl.set_tensor_place_ment(desc, mem_type)
函数功能	设置Tensor描述的placement属性，用于标识存放Tensor数据的内存为Host内存或Device内存。
输入说明	desc : int, Tensor描述信息的指针地址。 需提前调用 acl.create_tensor_desc 接口创建算子输入/输出的Tensor描述，作为本接口的输入。 mem_type : int, 指定placement属性值，其取值参考 2.16.29 aclMemType 。
返回值说明	ret : int, 错误码。 - 返回0表示成功。 - 返回其它值表示失败。
约束说明	无

2.4.35 函数：set_tensor_value_range

C函数原型	<code>aclError aclSetTensorValueRange(aclTensorDesc* desc, size_t valueCount, int64_t valueRange[] [ACL_TENSOR_VALUE_RANGE_NUM])</code>
Python函数	<code>ret = acl.set_tensor_value_range(desc, value_range)</code>
函数功能	调用acl.create_tensor_desc接口创建Tensor描述信息后，可通过本接口设置Tensor的数据值的范围。 使用场景：部分算子input的值就是该算子的输出Shape，在动态Shape场景下，这个Shape的值就有一个取值范围，在执行算子时，需要在input上设置Shape值的范围（例如：[[16, 16], [1, 128], [20, 20], [1, 10]]），pyACL才能正常执行算子。
输入说明	desc: int, aclTensorDesc类型的指针地址。需提前调用2.4.10 函数：create_tensor_desc接口创建aclTensorDesc类型。 value_range: list, 用二维列表表示每个数据值的范围。 ACL_TENSOR_VALUE_RANGE_NUM = 2
返回值说明	ret: int, 错误码。 • 返回0表示成功。 • 返回其它值表示失败。
约束说明	无

2.5 日志管理

2.5.1 函数：app_log

C函数原型	<code>ACL_APP_LOG(aclLogLevel logLevel, const char *message)</code>
Python函数	<code>acl.app_log(log_level, message)</code>
函数功能	将日志记录到系统相应的日志文件中。
输入说明	log_level: int, 设定日志级别。 • 0: DEBUG • 1: INFO • 2: WARNING • 3: ERROR message: str, 记录的日志信息。
返回值说明	无
约束说明	无

2.6 Util 模块

2.6.1 函数：numpy_to_ptr

须知

该接口即将废弃，建议使用[acl.util.bytes_to_ptr](#)接口。

C函数原型	无
Python函数	<code>ptr = acl.util.numpy_to_ptr(data)</code>
函数功能	获取numpy.ndarry数组的内存数据指针地址。
输入说明	data : numpy.ndarray类型数据。
返回值说明	ptr : int, numpy数组的内存数据指针地址。
约束说明	无

注意事项	● 若要继续使用该接口，需要运行环境为python ≥ 3.8且numpy ≥ 1.22.0。 ● 修改示例如下： np_ptr = acl.util.numpy_to_ptr(np_arr_in) 修改后使用： bytes_in = np_arr_in.tobytes() bytes_ptr = acl.util.bytes_to_ptr(bytes_in) ● 使用本函数返回的指针地址前，需要确保传入的numpy.ndarray对象生命周期还没有结束（没有被删除或被Python的GC回收），否则将会产生未定行为。 ● 若传入的numpy.ndarray数组在内存上是非行连续的，在接口调用时会打印如下提示信息： Warning: The input ndarray is discontiguous. Please use acl.util.numpy_contiguous_to_ptr instead. 请使用acl.util.numpy_contiguous_to_ptr接口获取numpy数组的地址对象，否则在使用acl.rt.memcpy拷贝数据时，由于acl.rt.memcpy拷贝的是连续内存地址，会导致拷贝的数据和目标拷贝数据不一致的问题。 ● 可以通过numpy数组的“flags”属性来查看numpy数组在内存上是否连续： print(data.flags) 得到结果如下： C_CONTIGUOUS : True F_CONTIGUOUS : False OWNDATA : True WRITEABLE : True ALIGNED : True WRITEBACKIFCOPY : False UPDATEIFCOPY : False 其中“C_CONTIGUOUS: True”，则说明numpy数组在内存上行连续。“F_CONTIGUOUS: False”，则说明numpy数组在内存上列不连续。

2.6.2 函数：numpy_contiguous_to_ptr

须知	
该接口即将废弃，建议使用 acl.util.bytes_to_ptr 接口。	
C函数原型	无
Python函数	ptr, data_out = acl.util.numpy_contiguous_to_ptr(data_in)
函数功能	获取numpy.ndarray数组的内存数据指针地址。 若传入的numpy.ndarray数组在内存上是非行连续的，则先将其内存转行连续，然后再获取转换后的numpy.ndarray数组转换为内存数据指针地址。
输入说明	data_in：numpy类型数据。

返回值说明	ptr: int，可以被C语言访问的数据。 data_out: numpy类型数据，若传入的numpy数组“data_in”内存非行连续，则返回值“data_out”即为其内存转行连续后的numpy数组。
约束说明	访问的是int类型的数据，需要将numpy.ndarray对象的内存数据指针地址转换为int。
注意事项	- 修改示例如下： np_ptr, data_out = acl.util.numpy_contiguous_to_ptr(np_arr_in) 修改后使用： bytes_in = np_arr_in.tobytes() bytes_ptr = acl.util.bytes_to_ptr(bytes_in) - 若要继续使用该接口，需要运行环境为python ≥ 3.8且numpy ≥ 1.22.0。 - 使用本函数返回的指针地址前，需要确保传入的numpy.ndarry对象生命周期还没有结束（没有被删除或被Python的GC回收），否则将会产生未定行为。

2.6.3 函数： ptr_to_numpy

须知	
该接口即将废弃，建议使用 acl.util.ptr_to_bytes 接口。	
C函数原型	无
Python函数	output = acl.util.ptr_to_numpy(ptr, shape, type)
函数功能	将指针地址数据转换为numpy数组，可以使Python代码直接访问。

输入说明	ptr: int, C语言中的指针地址, 是能够访问的数据的首地址。 shape: tuple, 需要构造的numpy的Shape。 type: int, 表示ptr中数据的数据类型。 下面举例一些常用的类型: • 0: NPY_BOOL • 1: NPY_BYTE, NPY_INT8 • 2: NPY_UINT8 • 3: NPY_SHORT, NPY_INT16 • 4: NPY_USHORT, NPY_UINT16 • 5: NPY_INT, NPY_INT32 • 6: NPY_UINT, NPY_UINT32 • 7: NPY_INT64 • 8: NPY_UINT64 • 9: NPY_LONGLONG • 10: NPY_ULONGLONG • 11: NPY_FLOAT32 • 12: NPY_DOUBLE • 23: NPY_HALF, NPY_FLOAT16
返回值说明	output : numpy类型。
约束说明	无
注意事项	• 修改示例如下: <code>np_arr_out = acl.util.ptr_to_numpy(host_ptr, np_arr_in.shape, NPY_INT32)</code> 修改后使用: <code>bytes_out = acl.util.ptr_to_bytes(ptr, size)</code> <code>np_arr_out = np.frombuffer(bytes_out, dtype=np.int32).reshape(np_arr_in.shape)</code> • 若要继续使用该接口, 需要运行环境为python ≥ 3.8且numpy ≥ 1.22.0。

2.6.4 函数: bytes_to_ptr

C函数原型	无
Python函数	ptr = acl.util.bytes_to_ptr(data)
函数功能	将bytes对象转换成为void*数据, 可以将转换好的数据传递给C函数直接使用。
输入说明	data : bytes类型数据。
返回值说明	ptr : int, 可以被C语言访问的数据。

约束说明	该函数输入的生命周期需要大于输出的生命周期，否则可能会产生未定义行为。
注意事项	推荐使用 acl.util.bytes_to_ptr 替代 acl.util.ptr_to_bytes 。 修改示例： <pre>bytes_in = np_arr_in.tobytes() bytes_ptr = acl.util.bytes_to_ptr(bytes_in)</pre>

2.6.5 函数：ptr_to_bytes

C函数原型	无
Python函数	bytes_out = acl.util.ptr_to_bytes(ptr, size)
函数功能	将void*数据转换为bytes对象，可以使Python代码直接访问。
输入说明	ptr : int, C语言中的指针地址，能够访问的数据的首地址。 size : int, 数据的大小，单位Byte。
返回值说明	bytes_out : bytes对象。
约束说明	无
注意事项	由于 acl.util.ptr_to_bytes 会导致数据性能下降，建议使用 2.6.4 函数：bytes_to_ptr 。 <pre>bytes_out = acl.util.ptr_to_bytes(bytes_ptr, size)</pre>

2.6.6 thread

2.6.6.1 函数：start_thread

C函数原型	无
Python函数	thr_id, ret = acl.util.start_thread(thr_func, args_list)
函数功能	创建线程，在线程中执行传入的Python函数。
输入说明	thr_func : Python函数对象，表示用户自定义的Python函数。 例如： <pre>def thr_func(args_list): pass</pre> args_list : list, 表示用户自定义的Python函数（thr_func）需要的传入参数。
返回值说明	thr_id : int, 线程ID号。 ret : int, 错误码。 - 返回0表示成功。 - 返回其它值表示失败。

约束说明	新建线程，无法自动获取Context，需要调用 acl.rt.set_context 接口显式指定Context。

2.6.6.2 函数：stop_thread

C函数原型	无
Python函数	<code>ret = acl.util.stop_thread(thr_id)</code>
函数功能	等待由 acl.util.start_thread 接口创建的线程退出。
输入说明	thr_id : int，线程ID号。需指定由 acl.util.start_thread 接口获得的线程ID号。
返回值说明	ret : int，错误码。 • 返回0表示成功。 • 返回其它值表示失败。
约束说明	无

2.7 Runtime (rt)

2.7.1 Device 管理

2.7.1.1 函数：set_device

C函数原型	<code>aclError aclrtSetDevice(int32_t deviceId)</code>
Python函数	<code>ret = acl.rt.set_device(device_id)</code>
函数功能	指定当前线程中用于运算的Device，同时隐式创建默认Context。 对于Atlas 200/300/500 推理产品，该默认Context中包含2个Stream，1个默认Stream和1个执行内部同步的Stream。 对于Atlas 200I/500 A2推理产品，该默认Context中包含1个默认Stream。 对于Atlas 训练系列产品，该默认Context中包含1个默认Stream。 对于Atlas A2训练系列产品/Atlas 800I A2推理产品，该默认Context中包含1个默认Stream。 对于Atlas 推理系列产品，在标准形态下，该Context中包含2个Stream，1个默认Stream和1个执行内部同步的Stream。
输入说明	device_id : int，Device设备号。 用户调用 acl.rt.get_device_count 接口获取可用的Device数量后，取值范围：[0, (可用的Device数量 - 1)]。

返回值说明	ret : int, 错误码。 • 返回0表示成功。 • 返回其它值表示失败。
约束说明	如果多次调用acl.rt.set_device接口而不调用acl.rt.reset_device接口释放本进程使用的Device资源, 功能上不会有问题, 因为在进程退出时也会释放本进程使用的Device资源。建议acl.rt.set_device接口和acl.rt.reset_device接口配对使用, 在不使用Device上资源时, 通过调用acl.rt.reset_device接口及时释放本进程使用的Device资源。 在不同进程或线程中支持调用acl.rt.set_device接口指定同一个Device用于运算。在同一个进程中的多个线程中, 如果调用acl.rt.set_device接口指定同一个Device用于运算, 这时隐式创建的默认Context是同一个。
参考资源	接口调用流程与示例, 请参见运行管理资源申请与释放、同步等待。

2.7.1.2 函数: reset_device

C函数原型	aclError aclrtResetDevice(int32_t deviceId)
Python函数	ret = acl.rt.reset_device(device_id)
函数功能	复位当前运算的Device, 释放Device上的资源, 包括默认Context、默认Stream以及默认Context下创建的所有Stream。若默认Context或默认Stream下的任务还未完成, 系统会等待任务完成后释放。
输入说明	device_id : int, Device设备号。
返回值说明	ret : int, 错误码。 • 返回0表示成功。 • 返回其它值表示失败。
约束说明	若要复位的Device上存在显式创建的Context、Stream、Event, 在复位前, 建议遵循如下接口调用顺序, 否则可能会导致业务异常。 接口调用顺序: 调用acl.rt.destroy_event接口释放Event/调用acl.rt.destroy_stream接口释放显式创建的Stream-->调用acl.rt.destroy_context释放显式创建的Context-->调用acl.rt.reset_device接口。
参考资源	接口调用流程与示例, 请参见运行管理资源申请与释放、同步等待。

2.7.1.3 函数: get_device

C函数原型	<code>aclError aclrtGetDevice(int32_t *deviceId)</code>
Python函数	<code>device_id, ret = acl.rt.get_device()</code>
函数功能	获取当前正在使用的Device的ID。
输入说明	无
返回值说明	device_id : int, 正在使用的Device设备号。 ret : int, 错误码。 - 返回0表示成功。 - 返回其它值表示失败。
约束说明	如果没有调用 acl.rt.set_device 接口显式指定Device或没有调用 acl.rt.create_context 接口隐式指定Device, 则调用 acl.rt.get_device 接口时, 返回错误。

2.7.1.4 函数: get_run_mode

C函数原型	<code>aclError aclrtGetRunMode(aclrtRunMode *runMode)</code>
Python函数	<code>run_mode, ret = acl.rt.get_run_mode()</code>
函数功能	获取当前昇腾AI软件栈的运行模式。
输入说明	无
返回值说明	run_mode : int, 表示运行模式。 0: ACL_DEVICE, 昇腾AI软件栈运行在Device的Control CPU或板端环境上。 Atlas 训练系列产品, 不支持该选项。 Atlas A2训练系列产品/Atlas 800I A2推理产品, 不支持该选项。 1: ACL_HOST, 昇腾AI软件栈运行在Host CPU上。 ret : int, 错误码。 - 返回0表示成功。 - 返回其它值表示失败。
约束说明	无
参考资源	接口调用流程与示例, 请参见运行管理资源申请与释放。

2.7.1.5 函数: set_ts_device

C函数原型	<code>aclError aclrtSetTsDevice(aclrtTsId tsId)</code>
Python函数	<code>ret = acl.rt.set_ts_device(ts_id)</code>

函数功能	设置本次计算需要使用的Task Schedule。
输入说明	ts_id : int, 指定本次计算需要使用的Task Schedule。如果昇腾AI软件栈中仅有AICORE Task Schedule, 则设置该参数无效, 默认使用AICORE Task Schedule。 0: ACL_TS_ID_AICORE, 使用AICORE Task Schedule。 1: ACL_TS_ID_AIVECTOR, 使用AIVECTOR Task Schedule。 2: ACL_TS_ID_RESERVED。
返回值说明	ret : int, 错误码。 - 返回0表示成功。 - 返回其它值表示失败。
约束说明	无

2.7.1.6 函数: get_device_count

C函数原型	aclError aclrtGetDeviceCount(uint32_t *count)
Python函数	count, ret = acl.rt.get_device_count()
函数功能	获取可用Device的数量。
输入说明	无
返回值说明	count : int, 获取Device数量。 ret : int, 错误码。 - 返回0表示成功。 - 返回其它值表示失败。
约束说明	无

2.7.1.7 函数: get_device_utilization_rate

C函数原型	aclError aclrtGetDeviceUtilizationRate(int32_t deviceId, aclrtUtilizationInfo *utilizationInfo)
Python函数	utilization_info, ret = acl.rt.get_device_utilization_rate(device_id)
函数功能	查询Device上Cube、Vector、AI CPU等的利用率。
输入说明	device_id : int, Device设备号。 用户调用 acl.rt.get_device_count 接口获取可用的Device数量后, 该Device ID的取值范围为[0, (可用的Device数量-1)]。

返回值说明	utilization_info : dict, aclrtUtilizationInfo类型利用率信息字典，具体请参见 2.16.35 aclrtUtilizationInfo 。 ret : int, 错误码。 • 返回0表示成功。 • 返回其它值表示失败。
约束说明	• 昇腾虚拟化实例场景下，不支持调用本接口查询利用率，接口返回值无实际意义。 • 开启Profiling功能时，不支持调用本接口查询利用率，接口返回值无实际意义。 • 查询Device内存利用率为预留功能，当前版本不支持，若调用本接口查询内存利用率，查询到的利用率为-1。 • Atlas 200/300/500 推理产品上没有Vector，调用本接口查询Vector利用率时，查询到的利用率为-1。 • Atlas 训练系列产品上没有Vector，调用本接口查询Vector利用率时，查询到的利用率为-1。

2.7.1.8 函数：query_device_status

C函数原型	aclError aclrtQueryDeviceStatus(int32_t deviceId, aclrtDeviceStatus *deviceStatus)
Python函数	device_status, ret = acl.rt.query_device_status(device_id)
函数功能	查询Device状态是正常可用、还是异常不可用。
输入说明	device_id : int, Device设备号。 用户调用 acl.rt.get_device_count 接口获取可用的Device数量后，该Device ID的取值范围为[0, (<i>可用的Device数量</i> -1)]。
返回值说明	device_status : Device状态，具体请参见 2.16.38 aclrtDeviceStatus 。 ret : int, 错误码。 • 返回0表示成功。 • 返回其它值表示失败。
约束说明	无

2.7.1.9 函数：synchronize_device

C函数原型	aclError aclrtSynchronizeDevice(void)
Python函数	ret = acl.rt.synchronize_device()

函数功能	阻塞应用程序运行，直到正在运算中的Device完成运算。 多Device场景下，调用该接口等待的是当前Context对应的Device。
输入说明	无
返回值说明	ret : int, 错误码。 - 返回0表示成功。 - 返回其它值表示失败。
约束说明	无
参考资源	- 接口调用流程，参见多Device切换。 - 接口调用示例，参见关于Device的同步等待。

2.7.1.10 函数：get_soc_name

C函数原型	<code>const char *aclrtGetSocName()</code>
Python函数	<code>name = acl.get_soc_name()</code>
函数功能	查询当前运行环境的芯片版本。
输入说明	无
返回值说明	name : str, 返回芯片版本字符串。如果通过该接口获取芯片版本失败，则返回None。
约束说明	无

2.7.1.11 函数：set_device_sat_mode

C函数原型	<code>aclError aclrtSetDeviceSatMode(aclrtFloatOverflowMode mode)</code>
Python函数	<code>ret = acl.rt.set_device_sat_mode(mode)</code>
函数功能	设置当前Context对应的Device的浮点计算结果输出模式。 Atlas 200/300/500 推理产品 ，当前版本不支持该接口。 Atlas 200I/500 A2推理产品 ，当前版本不支持该接口。 Atlas 训练系列产品 ，当前版本不支持该接口。 Atlas 推理系列产品 ，当前版本不支持该接口。
输入说明	mode : int, 设置浮点计算结果输出模式，具体请参见 2.16.34 aclrtFloatOverflowMode 。

返回值说明	ret : int, 错误码。 - 返回0表示成功。 - 返回其它值表示失败。
约束说明	调用该接口成功后, 后续在该Device上新创建的Stream按设置的模式生效, 对之前已创建的Stream不生效。

2.7.1.12 函数: get_device_sat_mode

C函数原型	aclError aclrtGetDeviceSatMode(aclrtFloatOverflowMode *mode)
Python函数	mode, ret = acl.rt.get_device_sat_mode()
函数功能	查询当前Context对应的Device的浮点计算结果输出模式。 Atlas 200/300/500 推理产品 , 当前版本不支持该接口。 Atlas 200I/500 A2推理产品 , 当前版本不支持该接口。 Atlas 训练系列产品 , 当前版本不支持该接口。 Atlas 推理系列产品 , 当前版本不支持该接口。
输入说明	无
返回值说明	mode : int, 获取浮点计算结果输出模式, 具体请参见 2.16.34 aclrtFloatOverflowMode 。 ret : int, 错误码。 - 返回0表示成功。 - 返回其它值表示失败。
约束说明	无

2.7.2 Context 管理

2.7.2.1 函数: create_context

C函数原型	aclError aclrtCreateContext(aclrtContext *context, int32_t deviceId)
Python函数	context, ret = acl.rt.create_context(device_id)

函数功能	在当前进程或线程中显式创建一个Context。 对于Atlas 200/300/500 推理产品，该Context中包含2个Stream，1个默认Stream和1个执行内部同步的Stream。 对于Atlas 200I/500 A2推理产品，该Context中包含2个Stream，1个默认Stream和1个执行内部同步的Stream。 对于Atlas 训练系列产品，该Context中包含1个默认Stream。 对于Atlas A2训练系列产品/Atlas 800I A2推理产品，该Context中包含1个默认Stream。 对于Atlas 推理系列产品，在标准形态下，该Context中包含2个Stream，1个默认Stream和1个执行内部同步的Stream。 说明 如果在应用程序中没有调用acl.rt.set_device接口，那么在首次调用acl.rt.create_context接口时，系统内部会根据该接口传入的Device ID，为该Device绑定一个默认Stream（一个Device仅绑定一个默认Stream），因此在首次调用acl.rt.create_context接口时，占用的Stream数量 = Device上绑定的默认Stream + Context中包含的Stream。
输入说明	device_id : int，指定需要创建Context的Device设备号。
返回值说明	context : int，表示创建的Context的指针地址。 ret : int，错误码。 • 返回0表示成功。 • 返回其它值表示失败。
约束说明	支持以下使用场景： • 若不调用acl.rt.create_context接口显式创建Context，那系统会使用默认Context，该默认Context是在调用acl.rt.set_device接口时隐式创建的。 - 隐式创建Context：适合简单、无复杂交互逻辑的应用，但缺点在于，在多线程编程中，执行结果取决于线程调度的顺序。 - 显式创建Context：适合大型、复杂交互逻辑的应用，便于提高程序的可读性、可维护性。 • 在某一进程中指定Device，该进程内的多个线程可共用在此Device上显式创建的Context（调用acl.rt.create_context接口显式创建Context）。 • 若在某一进程内创建多个Context（Context的数量与Stream相关，Stream数量有限，请参见acl.rt.create_stream），当前线程在同一时刻内只能使用其中一个Context，建议通过acl.rt.set_context接口明确指定当前线程的Context，增加程序的可维护性。
参考资源	接口调用流程与示例，请参见运行管理资源申请与释放。

2.7.2.2 函数：destroy_context

C函数原型	<code>aclError aclrtDestroyContext(aclrtContext context)</code>

Python函数	ret = acl.rt.destroy_context(context)
函数功能	销毁一个Context，释放Context的资源。只能销毁通过 acl.rt.create_context 接口创建的Context。
输入说明	context : int，指定需要销毁的Context对象指针地址。
返回值说明	ret : int，错误码。 • 返回0表示成功。 • 返回其它值表示失败。
约束说明	无
参考资源	接口调用流程与示例，请参见运行管理资源申请与释放、同步等待。

2.7.2.3 函数：set_context

C函数原型	aclError aclrtSetCurrentContext(aclrtContext context)
Python函数	ret = acl.rt.set_context(context)
函数功能	设置线程的Context。
输入说明	context : int，线程当前的Context对象指针地址。
返回值说明	ret : int，错误码。 • 返回0表示成功。 • 返回其它值表示失败。
约束说明	• 支持以下场景： - 如果在某线程（例如：thread1）中调用acl.rt.create_context接口显式创建一个Context（例如：ctx1），则可以不调用acl.rt.set_context接口指定该线程的Context，系统默认将ctx1作为thread1的Context。 - 如果没有调用acl.rt.create_context接口显式创建Context，则系统将默认Context作为线程的Context，此时，不能通过acl.rt.destroy_context接口来释放默认Context。 - 如果多次调用acl.rt.set_context接口设置线程的Context，以最后一次为准。 • 若给线程设置的Context所对应的Device已经被复位，则不能将该Context设置为线程的Context，否则会导致业务异常。 • 推荐在某一线程中创建的Context，在该线程中使用。若在线程A中调用acl.rt.create_context接口创建Context，在线程B中使用该Context，则需由用户自行保证两个线程中同一个Context下同一个Stream中任务执行的顺序。
参考资源	接口调用流程，参见多Device切换。

2.7.2.4 函数：get_context

C函数原型	<code>aclError aclrtGetCurrentContext(aclrtContext *context)</code>
Python函数	<code>context, ret = acl.rt.get_context()</code>
函数功能	获取线程的Context。
输入说明	无
返回值说明	context : int, 表示创建的Context对象指针地址。 ret : int, 错误码。 • 返回0表示成功。 • 返回其它值表示失败。
约束说明	如果用户多次调用 acl.rt.set_context 接口设置当前线程的Context, 则获取的是最后一次设置的Context。

2.7.2.5 函数：get_overflow_status

C函数原型	<code>aclError aclrtGetOverflowStatus(void *outputAddr, size_t outputSize, aclrtStream stream)</code>
Python函数	<code>ret = acl.rt.get_overflow_status(outputAddr, outputSize, stream)</code>
函数功能	获取当前Context下所有Stream上任务的溢出状态, 并将状态值拷贝到用户申请的Device内存中。异步接口。 Atlas 200/300/500 推理产品 , 不支持该接口。
输入说明	outputAddr : int, 用户申请的Device内存, 需通过 acl.rt.malloc 接口申请。如果需要在Host侧查看数据, 可调用 acl.rt.memcpy 或 acl.rt.memcpy_async 接口, 将Device侧的数据传输到Host侧。 outputSize : int, 需申请的Device内存大小, 单位Byte, 固定大小为64Byte。 stream : int, 指定Stream的指针地址, 用于下发溢出状态查询任务。
返回值说明	ret : int, 错误码。 • 返回0表示成功。 • 返回其它值表示失败。
约束说明	该接口是异步接口, 调用接口成功仅代表任务下发成功, 不表示任务执行成功。调用该接口后, 需调用同步等待接口 (例如, acl.rt.synchronize_stream) 确保任务已执行完成。 Atlas A2训练系列产品/Atlas 800I A2推理产品 上, 调用本接口查询出来的溢出状态是进程级别的。

2.7.2.6 函数：reset_overflow_status

C函数原型	<code>aclError aclrtResetOverflowStatus(aclrtStream stream)</code>
Python函数	<code>ret = acl.rt.reset_overflow_status(stream)</code>
函数功能	清除当前Context下所有Stream上任务的溢出状态。异步接口。 Atlas 200/300/500 推理产品 ，不支持该接口。
输入说明	stream : int, 指定Stream的指针地址，用于下发溢出状态复位任务。
返回值说明	ret : int, 错误码。 • 返回0表示成功。 • 返回其它值表示失败。
约束说明	该接口是异步接口，调用接口成功仅表示任务下发成功，不表示任务执行成功。 调用该接口后，需调用同步等待接口（例如， acl.rt.synchronize_stream ）确保任务已执行完成。 Atlas A2训练系列产品/Atlas 800I A2推理产品 上，调用本接口清除的溢出状态是进程级别的。

2.7.3 Stream 管理

2.7.3.1 函数：create_stream

C函数原型	<code>aclError aclrtCreateStream(aclrtStream *stream)</code>
Python函数	<code>stream, ret = acl.rt.create_stream()</code>
函数功能	在当前进程或线程中创建一个Stream。
输入说明	无
返回值说明	stream : int, 返回创建的Stream的指针地址。 ret : int, 错误码。 • 返回0表示成功。 • 返回其它值表示失败。

约束说明	• 每个Context对应一个默认Stream，该默认Stream是调用acl.rt.set_device接口或acl.rt.create_context接口隐式创建的。推荐调用acl.rt.create_stream接口显式创建Stream。 - 隐式创建Stream：适合简单、无复杂交互逻辑的应用，但缺点在于，在多线程编程中，执行结果取决于线程调度的顺序。 - 显式创建Stream：推荐显式，适合大型、复杂交互逻辑的应用，且便于提高程序的可读性、可维护性。 • Atlas 200/300/500 推理产品 - 硬件资源最多支持1024个Stream。 - 如果已存在多个默认Stream，只能显式创建N个Stream（$N = 1024 - \text{默认Stream个数} - \text{执行内部同步的Stream个数}$）。 例如，若已存在一个默认Stream和一个执行内部同步的Stream，则只能显式创建1022个Stream。 • Atlas 200I/500 A2推理产品 - 硬件资源最多支持512个Stream。 - 如果已存在多个默认Stream，只能显式创建N个Stream（$N = 512 - \text{默认Stream个数} - \text{执行内部同步的Stream个数}$）。 例如，若已存在一个默认Stream和一个执行内部同步的Stream，则只能显式创建510个Stream。 • Atlas 推理系列产品 - 硬件资源最多支持1024个Stream。 - 标准形态下，如果已存在多个默认Stream，只能显式创建N个Stream（$N = 1024 - \text{默认Stream个数} - \text{执行内部同步的Stream个数}$）。 例如，若已存在一个默认Stream和一个执行内部同步的Stream，则只能显式创建1022个Stream。 • Atlas 训练系列产品 - 硬件资源最多支持2048个Stream。 - 如果已存在多个默认Stream，只能显式创建N个Stream（$N = 2048 - \text{默认Stream个数} - \text{执行内部同步的Stream个数}$）。 例如，若已存在一个默认Stream和一个执行内部同步的Stream，则只能显式创建2046个Stream。 - 多进程场景下，若一次性创建的Stream数量总和接近2048，可能会出现创建Stream失败的情况，此时，建议进行以下操作： 1. 清理冗余Stream，减少不必要的Stream。 2. 调整代码逻辑，分批创建Stream。 例如，第一批创建部分Stream，然后第二批再创建部分Stream，以此类推，直到Stream总数接近2048。 • Atlas A2训练系列产品/Atlas 800I A2推理产品 - 硬件资源最多支持2048个Stream。

	- 如果已存在多个默认Stream，只能显式创建N个Stream（N = 2048 - 默认Stream个数 - 执行内部同步的Stream个数）。 例如，若已存在一个默认Stream和一个执行内部同步的Stream，则只能显式创建2046个Stream。
参考资源	接口调用流程与示例，请参见运行管理资源申请与释放。

2.7.3.2 函数：create_stream_with_config

C函数原型	<code>aclError aclrtCreateStreamWithConfig(aclrtStream *stream, uint32_t priority, uint32_t flag)</code>
Python函数	<code>stream, ret = acl.rt.create_stream_with_config(priority, flag)</code>
函数功能	在当前进程或线程中创建一个Stream。 在Atlas 推理系列产品上，相比acl.rt.create_stream接口，使用本接口可以创建一个快速下发任务的Stream，但会增加内存消耗或CPU的性能消耗。 在Atlas 训练系列产品上，相比acl.rt.create_stream接口，使用本接口可以创建一个快速下发任务的Stream，但会增加内存消耗或CPU的性能消耗。 在Atlas A2训练系列产品/Atlas 800I A2推理产品上，相比acl.rt.create_stream接口，使用本接口可以创建一个快速下发任务的Stream，但会增加内存消耗或CPU的性能消耗。 在Atlas 200/300/500 推理产品上，使用本接口与acl.rt.create_stream接口相比是等价的。 在Atlas 200I/500 A2推理产品上，使用本接口与acl.rt.create_stream接口是等价的。

输入说明	priority: int, 优先级。 Atlas 200/300/500 推理产品上, 当前固定设置为0, 预留参数, 暂不使用。 Atlas 训练系列产品上, 当前固定设置为0, 预留参数, 暂不使用。 Atlas A2训练系列产品/Atlas 800I A2推理产品上, 当前固定设置为0, 预留参数, 暂不使用。 Atlas 200I/500 A2推理产品上, 当前固定设置为0, 预留参数, 暂不使用。 Atlas 推理系列产品上, 该参数取值范围: [0, 7], 总共最多支持8个优先级, 数字越小代表优先级越高, 其中, 0的优先级最高, 7的优先级最低。配置取值范围以外的值, 本接口返回报错。 flag: int, Stream的flag。 取值范围: ● ACL_STREAM_FAST_LAUNCH = 1, 使用该模式创建出来的Stream, 在使用Stream时, 下发任务的速度更快。 相比acl.rt.create_stream接口创建出来的Stream, 在使用Stream时才会申请系统内部资源, 导致下发任务的时长增加, 使用本接口的“ACL_STREAM_FAST_LAUNCH”模式创建Stream时, 会在创建Stream时预申请系统内部资源, 因此创建Stream的时长增加, 下发任务的时长缩短, 总体来说, 创建一次Stream, 使用多次的场景下, 总时长缩短, 但创建Stream时预申请内部资源会增加内存消耗。 ● ACL_STREAM_FAST_SYNC = 2, 使用该模式创建出来的Stream, 在调用acl.rt.synchronize_stream接口时, 会阻塞当前线程, 主动查询任务的执行状态, 一旦任务完成, 立即返回。 相比acl.rt.create_stream接口创建出来的Stream, 在调用acl.rt.synchronize_stream接口时, 会一直被动等待Device上任务执行完成的通知, 等待时间长, 使用本接口的“ACL_STREAM_FAST_SYNC”模式创建的Stream, 没有被动等待, 总时长缩短, 但主动查询的操作会增加CPU的性能消耗。 说明 配置取值范围之外的值, 本接口创建出来的Stream等同于acl.rt.create_stream接口。
返回值说明	stream: int, 表示创建的Stream对象的指针地址。 ret: int, 错误码。 ● 返回0表示成功。 ● 返回其它值表示失败。

2.7.3.3 函数: destroy_stream

C函数原型	aclError aclrtDestroyStream(aclrtStream stream)
Python函数	ret = acl.rt.destroy_stream(stream)

函数功能	销毁指定Stream，销毁通过 acl.rt.create_stream 或 acl.rt.create_stream_with_config 接口创建的Stream，若Stream上有未完成的任务，会等待任务完成后再销毁Stream。
输入说明	stream : int，待销毁的Stream的指针地址。
返回值说明	ret : int，错误码。 - 返回0表示成功。 - 返回其它值表示失败。
约束说明	- 调用acl.rt.destroy_stream接口销毁指定Stream前，需要先调用acl.rt.synchronize_stream接口确保Stream中的任务都已完成。 - 调用acl.rt.destroy_stream接口销毁指定Stream时，需确保该Stream在当前Context下。 - 调用acl.rt.destroy_stream接口销毁指定Stream时，需确保其它接口没有正在使用该Stream。
参考资源	接口调用流程与示例，请参见运行管理资源申请与释放、同步等待。

2.7.3.4 函数：destroy_stream_force

C函数原型	aclError aclrtDestroyStreamForce(aclrtStream stream)
Python函数	ret = acl.rt.destroy_stream_force(stream)
函数功能	销毁指定Stream，销毁通过 acl.rt.create_stream 或 acl.rt.create_stream_with_config 接口创建的Stream，若Stream上有未完成的任务，不会等待任务完成，直接强制销毁Stream。
输入说明	stream : int，待销毁的Stream的指针地址。
返回值说明	ret : int，错误码。 - 返回0表示成功。 - 返回其它值表示失败。
约束说明	调用本接口销毁指定Stream时，需确保该Stream在当前Context下。

2.7.3.5 函数：set_stream_overflow_switch

C函数原型	aclError aclrtSetStreamOverflowSwitch(aclrtStream stream, uint32_t flag)
Python函数	ret = acl.rt.set_stream_overflow_switch(stream, flag)

函数功能	饱和模式下，对接上层训练框架时（例如：PyTorch），针对指定Stream，打开或关闭溢出检测开关，关闭后无法通过溢出检测算子获取任务是否溢出。 Atlas 200/300/500 推理产品，当前版本不支持该接口。 Atlas 200I/500 A2推理产品，当前版本不支持该接口。 Atlas 训练系列产品，当前版本不支持该接口。 Atlas 推理系列产品，当前版本不支持该接口。
输入说明	stream: int，指定Stream的指针地址，若传入0，则操作默认Stream。 flag: int，溢出检测开关，取值范围如下。 0: 关闭 1: 打开
返回值说明	ret: int，错误码。 - 返回0表示成功。 - 返回其它值表示失败。
约束说明	- 在调用本接口前，可调用acl.rt.set_device_sat_mode接口设置饱和模式。 - 调用该接口打开或关闭溢出检测开关后，仅对后续新下的任务生效，已下发的任务仍维持原样。

2.7.3.6 函数：get_stream_overflow_switch

C函数原型	<code>aclError aclrtGetStreamOverflowSwitch(aclrtStream stream, uint32_t flag)</code>
Python函数	<code>flag, ret = acl.rt.get_stream_overflow_switch(stream)</code>
函数功能	针对指定Stream，获取其当前溢出检测开关是否打开。 Atlas 200/300/500 推理产品，当前版本不支持该接口。 Atlas 200I/500 A2推理产品，当前版本不支持该接口。 Atlas 训练系列产品，当前版本不支持该接口。 Atlas 推理系列产品，当前版本不支持该接口。
输入说明	stream: int，待操作Stream的指针地址，若传入0，则操作默认Stream。
返回值说明	flag: int，溢出检测开关，取值范围如下。 0: 关闭 1: 打开 ret: int，错误码。 - 返回0表示成功。 - 返回其它值表示失败。

2.7.3.7 函数：set_stream_failure_mode

C函数原型	<code>aclError aclrtSetStreamFailureMode(aclrtStream stream, uint64_t mode)</code>
Python函数	<code>ret = acl.rt.set_stream_failure_mode(stream, mode)</code>
函数功能	当一个Stream上下发了多个任务时，可通过本接口指定任务调度模式，以便控制某个任务失败后是否继续执行下一个任务。 Atlas 200/300/500 推理产品 ，当前版本不支持该接口。
输入说明	stream : int，待操作Stream的指针地址。 “stream”参数不支持指定默认Stream（即向该参数传入0）的任务调度模式。 mode : int，当一个Stream上下发了多个任务时，可通过本参数指定任务调度模式，以便控制某个任务失败后是否继续执行下一个任务。 取值范围如下： • ACL_CONTINUE_ON_FAILURE = 0：默认值，某个任务失败后，继续执行下一个任务。 • ACL_STOP_ON_FAILURE = 1：某个任务失败后，停止执行后续的任务。
返回值说明	ret : int，错误码。 • 返回0表示成功。 • 返回其它值表示失败。
约束说明	无

2.7.3.8 函数：stream_query

C函数原型	<code>aclError aclrtStreamQuery(aclrtStream stream, aclrtStreamStatus *status)</code>
Python函数	<code>status, ret = acl.rt.stream_query(stream)</code>
函数功能	查询指定Stream上的所有任务的执行状态。
输入说明	stream : int，待操作Stream的指针地址。
返回值说明	status : Stream上的任务状态，具体请参见 2.16.37 aclrtStreamStatus 。 ret : int，错误码。 • 返回0表示成功。 • 返回其它值表示失败。

约束说明	本接口支持在以下场景使用： ● Atlas 200/300/500 推理产品使用该接口查询任务的执行状态。 ● 昇腾虚拟化实例（Ascend Virtual Instance）场景下支持使用该接口查询任务的执行状态。

2.7.3.9 函数：synchronize_stream

C函数原型	<code>aclError aclrtSynchronizeStream(aclrtStream stream)</code>
Python函数	<code>ret = acl.rt.synchronize_stream(stream)</code>
函数功能	阻塞应用程序运行，直到指定Stream中的所有任务都完成。
输入说明	stream : int，指定需要完成所有任务的Stream对象的指针地址。
返回值说明	ret : int，错误码。 ● 返回0表示成功。 ● 返回其它值表示失败。
约束说明	无
参考资源	接口调用示例，参见关于Stream间任务的同步等待。

2.7.3.10 函数：synchronize_stream_with_timeout

C函数原型	<code>aclError aclrtSynchronizeStreamWithTimeout(aclrtStream stream, int32_t timeout)</code>
Python函数	<code>ret = acl.rt.synchronize_stream_with_timeout(stream, timeout)</code>
函数功能	阻塞应用程序运行，直到指定Stream中的所有任务都完成，该接口是在 acl.rt.synchronize_stream 接口基础上进行了增强，支持用户设置超时时间，当应用程序异常时可根据所设置的超时时间自行退出。
输入说明	stream : int，指定需要完成所有任务的Stream的指针地址。 timeout : int，接口的超时时间。取值说明如下： ● -1：表示永久等待，和接口acl.rt.synchronize_stream功能一样。 ● >0：配置具体的超时时间，单位是毫秒。
返回值说明	ret : int，错误码。 ● 返回0表示成功。 ● 返回其它值表示失败。
约束说明	无

2.7.4 Event 管理

2.7.4.1 函数：create_event

C函数原型	<code>aclError aclrtCreateEvent(aclrtEvent *event)</code>
Python函数	<code>event, ret = acl.rt.create_event()</code>
函数功能	创建一个Event，创建的Event可用于统计两个Event之间的耗时、多Stream场景下的同步等待等场景。
输入说明	无
返回值说明	event : int，创建的Event对象的指针地址。 ret : int，错误码。 - 返回0表示成功。 - 返回其它值表示失败。
约束说明	调用本接口创建Event后，后续调用 acl.rt.record_event 接口时，系统内部才会申请Event资源，因此会受Event数量的限制，Event达到上限后，系统内部会等待资源释放。 Event数量限制如下： Atlas 200/300/500 推理产品 ，单个Device上最多支持1023个Event。 Atlas 推理系列产品 ，单个Device上最多支持1023个Event。 Atlas 200I/500 A2推理产品 ，单个Device上最多支持65535个Event。 Atlas 训练系列产品 ，单个Device上最多支持65535个Event。 Atlas A2训练系列产品/Atlas 800I A2推理产品 ，单个Device上最多支持65535个Event。

2.7.4.2 函数：create_event_with_flag

C函数原型	<code>aclError aclrtCreateEventWithFlag(aclrtEvent *event, uint32_t flag)</code>
Python函数	<code>event, ret = acl.rt.create_event_with_flag(flag)</code>
函数功能	创建一个带flag的Event，不同flag的Event用于不同的功能。支持创建Event时携带多个flag（按位进行或操作），从而同时使能对应flag的功能。

输入说明	flag: int, Event的flag, 可支持以下选项。 • ACL_EVENT_TIME_LINE = 0x00000008, 表示创建的Event数量不受限制且创建出来的Event可用于统计两个Event之间的耗时。 • ACL_EVENT_SYNC = 0x00000001, 表示创建的Event数量受限且创建出来的可用于多Stream间的同步等待。 • ACL_EVENT_CAPTURE_STREAM_PROGRESS = 0x00000002, 表示创建的Event用于跟踪Stream的任务执行进度。
返回值说明	event: int, 创建的Event对象指针地址。 ret: int, 错误码。 • 返回0表示成功。 • 返回其它值表示失败。
约束说明	使用本接口创建Event时, “flag” 为bitmap, 支持对单个flag或者多个flag进行操作, 例如: acl.rt.create_event_with_flag(ACL_EVENT_TIME_LINE ACL_EVENT_SYNC)。 • 若 “flag” 参数值包含 “ACL_EVENT_SYNC”, 则创建出来的Event数量受限, 具体参考如下: Atlas 200/300/500 推理产品, 单个Device上最多支持1023个Event。 Atlas 200I/500 A2推理产品, 单个Device上最多支持1023个Event。 Atlas 训练系列产品, 单个Device上最多支持65535个Event。 Atlas A2训练系列产品/Atlas 800I A2推理产品, 单个Device上最多支持65535个Event。 • 若flag参数值不包含 “ACL_EVENT_SYNC”, 则不支持在以下API中使用本接口创建的Event: acl.rt.reset_event、acl.rt.stream_wait_event、acl.rt.query_event_wait_status。

2.7.4.3 函数: create_event_ex_with_flag

C函数原型	aclError aclrtCreateEventExWithFlag(aclrtEvent *event, uint32_t flag)
Python函数	event, ret = acl.rt.create_event_ex_with_flag(flag)
函数功能	创建一个带flag的Event, 不同flag的Event用于不同的功能。支持创建Event时携带多个flag (按位进行或操作), 从而同时使能对应flag的功能。创建Event时, Event资源不受硬件限制。

输入说明	flag: int, Event指针的flag, 可支持以下选项。 • ACL_EVENT_TIME_LINE = 0x00000008, 表示创建的Event数量不受限制且创建出来的Event可用于统计两个Event之间的耗时。 • ACL_EVENT_SYNC = 0x00000001, 表示创建的Event数量受限且创建出来的可用于多Stream间的同步等待。 • ACL_EVENT_CAPTURE_STREAM_PROGRESS = 0x00000002, 表示创建的Event用于跟踪Stream的任务执行进度。
返回值说明	event: int, 创建的Event对象指针地址。 ret: int, 错误码。 • 返回0表示成功。 • 返回其它值表示失败。
约束说明	• 采用本API创建的Event, 不支持在以下接口中使用: acl.rt.reset_event、acl.rt.query_event、acl.rt.query_event_wait_status。 • 调用本接口创建Event时, “flag” 为bitmap, 支持对单个flag或者多个flag进行操作, 例如: <code>acl.rt.create_event_ex_with_flag(ACL_EVENT_TIME_LINE ACL_EVENT_SYNC)</code>。 - 若“flag” 参数值包含“ACL_EVENT_SYNC”, 后续调用acl.rt.record_event接口时, 系统内部才会申请Event资源, 因此会受Event数量的限制, Event达到上限后, 系统内部会等待资源释放。 Event数量限制如下: Atlas 200/300/500 推理产品, 单个Device上最多支持1023个Event。 Atlas 推理系列产品, 单个Device上最多支持1023个Event。 Atlas 200I/500 A2推理产品, 单个Device上最多支持65535个Event。 Atlas 训练系列产品, 单个Device上最多支持65535个Event。 Atlas A2训练系列产品/Atlas 800I A2推理产品, 单个Device上最多支持65535个Event。 - 若“flag” 参数值不包含“ACL_EVENT_SYNC”, 则不支持在acl.rt.stream_wait_event接口中使用本接口创建的Event。

2.7.4.4 函数: `destroy_event`

C函数原型	<code>aclError aclrtDestroyEvent(aclrtEvent event)</code>
Python函数	<code>ret = acl.rt.destroy_event(event)</code>

函数功能	销毁一个Event，支持在Event未完成前调用本接口销毁Event。此时，本接口不会阻塞线程等Event完成，Event相关资源会在Event完成时被自动释放。
输入说明	event : int，待销毁的Event指针地址。
返回值说明	ret : int，错误码。 • 返回0表示成功。 • 返回其它值表示失败。
约束说明	在调用 acl.rt.destroy_event 接口销毁指定Event时，需确保其它接口没有正在使用该Event。
参考资源	接口调用示例，参见关于Event的同步等待、关于Stream间任务的同步等待。

2.7.4.5 函数：record_event

C函数原型	aclError aclrtRecordEvent(aclrtEvent event, aclrtStream stream)
Python函数	ret = acl.rt.record_event(event, stream)
函数功能	在Stream中记录一个Event。本接口被调用时，pyACL会捕获当前Stream上已下发的任务并记录到Event事件中，因此后续若调用acl.rt.query_event_status或acl.rt.stream_wait_event接口时，会检查或等待该Event事件中所捕获的任务都已经完成。 对于使用acl rt.create_event_ex_with_flag创建的Event： • 本接口支持对同一个Event多次record实现Event复用，每次Record会重新捕获当前Stream上已下发的任务，并覆盖保存到Event中。在调用acl.rt.stream_wait_event接口时，会使用最近一次Event中所保存的任务，且不会被后续的acl.rt.record_event调用影响。 • 在首次调用本接口前，由于Event中没有任务，因此调用acl.rt.query_event_status接口时会返回“ACL_EVENT_RECORDED_STATUS_COMPLETE”。
输入说明	event : int，待记录的Event的指针地址。 stream : int，将该Event记录在指定的Stream（指针地址）中，如果使用默认Stream，此处设置为0。
返回值说明	ret : int，错误码。 • 返回0表示成功。 • 返回其它值表示失败。

约束说明	- 该接口是异步接口，调用接口成功仅表示任务下发成功，不表示任务执行成功。调用该接口后，需调用同步等待接口（例如，acl.rt.synchronize_stream）确保任务已执行完成。 acl.rt.record_event接口与acl.rt.stream_wait_event接口配合使用时，主要用于多Stream之间同步的场景，在调用acl.rt.record_event接口时，系统内部会申请Event资源，在调用acl.rt.stream_wait_event接口之后，请及时调用acl.rt.reset_event接口释放Event资源。 接口调用顺序：acl.rt.create_event-->acl.rt.record_event-->acl.rt.stream_wait_event-->acl.rt.reset_event
参考资源	接口调用示例，参见关于Event的同步等待、关于Stream间任务的同步等待。

2.7.4.6 函数：reset_event

C函数原型	<code>aclError aclrtResetEvent(aclrtEvent event, aclrtStream stream)</code>
Python函数	<code>ret = acl.rt.reset_event(event, stream)</code>
函数功能	复位一个Event。用户需确保等待Stream中的任务都完成后，再复位Event，异步接口。
输入说明	event : int，待复位的Event对象的指针地址。 stream : int，指定Event所在的Stream的对象指针地址。
返回值说明	ret : int，错误码。 - 返回0表示成功。 - 返回其它值表示失败。
约束说明	- 通过acl.rt.create_event_ex_with_flag接口创建的Event，不支持调用本接口。 - 该接口是异步接口，调用接口成功仅表示任务下发成功，不表示任务执行成功。调用该接口后，需调用同步等待接口（例如，acl.rt.synchronize_stream）确保任务已执行完成。

2.7.4.7 函数：query_event

须知 此接口后续版本会废弃，请使用2.7.4.8 函数：query_event_status接口。	
C函数原型	<code>aclError aclrtQueryEvent(aclrtEvent event, aclrtEventStatus *status)</code>

Python函数	status, ret = acl.rt.query_event(event)
函数功能	查询与本接口在同一线程中的 acl.rt.record_event 接口所记录的Event是否执行完成。
输入说明	event : int, 指定待查询的Event对象指针地址。
返回值说明	status : int, 表示的是Event状态的指针地址。 • ACL_EVENT_RECORDED_STATUS_NOT_READY = 0, Event未被记录到Stream中, 或记录到Stream中的Event未被执行或执行失败。 • ACL_EVENT_RECORDED_STATUS_COMPLETE = 1, 记录到Stream中的Event执行成功。 ret : int, 错误码。 • 返回0表示成功。 • 返回其它值表示失败。
约束说明	无

2.7.4.8 函数: query_event_status

C函数原型	aclError aclrtQueryEventStatus(aclrtEvent event, aclrtEventRecordedStatus *status)
Python函数	status, ret = acl.rt.query_event_status(event)
函数功能	查询该Event捕获的所有任务的执行状态。具体见 acl.rt.record_event 接口参考Event捕获的细节。
输入说明	event : int, 指定待查询的Event对象的指针地址。
返回值说明	status : int, 表示的是Event状态的指针地址。 • ACL_EVENT_RECORDED_STATUS_NOT_READY = 0, Event未被记录到Stream中, 或记录到Stream中的Event未被执行或执行失败。 • ACL_EVENT_RECORDED_STATUS_COMPLETE = 1, 记录到Stream中的Event执行成功。 ret : int, 错误码。 • 返回0表示成功。 • 返回其它值表示失败。
约束说明	如果用户在不同线程上分别调用 acl.rt.record_event 和 acl.rt.query_event_status , 可能由于多线程导致这两个API的执行时间乱序, 进而导致查询到的Event对象的完成状态不符合预期。

2.7.4.9 函数：query_event_wait_status

C函数原型	<code>aclError aclrtQueryEventWaitStatus(aclrtEvent event, aclrtEventWaitStatus *status)</code>
Python函数	<code>status, ret=acl.rt.query_event_wait_status(event)</code>
函数功能	调用 acl.stream_wait_event 接口后查询该Event对应的等待任务是否都执行完成。
输入说明	event : int, 指定待查询的Event对象的指针地址。
返回值说明	status : int, Event状态的指针地址, 具体请参见 2.16.30 aclrtEventWaitStatus 。 • ACL_EVENT_RECORDED_STATUS_NOT_READY = 0, Event未被记录到Stream中, 或记录到Stream中的Event未被执行或执行失败。 • ACL_EVENT_RECORDED_STATUS_COMPLETE = 1, 记录到Stream中的Event执行成功。 ret : int, 错误码。 • 返回0表示成功。 • 返回其它值表示失败。
约束说明	通过 acl.rt.create_event_ex_with_flag 接口创建的Event, 不支持调用本接口。

2.7.4.10 函数：synchronize_event

C函数原型	<code>aclError aclrtSynchronizeEvent(aclrtEvent event)</code>
Python函数	<code>ret = acl.rt.synchronize_event(event)</code>
函数功能	阻塞当前线程运行直到Event捕获的所有任务都执行完成。具体见 acl.rt.record_event 接口参考Event捕获的细节。
输入说明	event : int, 指定需等待的Event对象的指针地址。
返回值说明	ret : int, 错误码。 • 返回0表示成功。 • 返回其它值表示失败。
约束说明	无
参考资源	接口调用示例, 参见关于Event的同步等待。

2.7.4.11 函数：synchronize_event_with_timeout

C函数原型	<code>aclError aclrtSynchronizeEventWithTimeout(aclrtEvent event, int32_t timeout)</code>

Python函数	ret = acl.rt.synchronize_event_with_timeout(event, timeout)
函数功能	阻塞当前线程运行直到Event捕获的所有任务都执行完成（具体见 acl.rt.record_event 接口参考Event捕获的细节）。 该接口在 acl.rt.synchronize_event 接口的基础上进行了增强，支持用户设置永久等待或配置具体的超时时间，若配置具体的超时时间，则当应用程序异常时可根据所设置的超时时间自行退出。
输入说明	event : int，指定需等待的Event对象的指针地址。 timeout : int，接口的超时时间。 取值说明如下： • -1：表示永久等待与接口acl.rt.synchronize_event功能一致。 • >0：配置具体的超时时间，单位是毫秒。
返回值说明	ret : int，错误码。 • 返回0表示成功。 • 返回其它值表示失败。
约束说明	Atlas 200/300/500 推理产品 ，不支持调用接口配置具体的超时时间，仅支持设置永久等待。
参考资源	接口调用示例，参见关于Event的同步等待。

2.7.4.12 函数：event_elapsed_time

C函数原型	aclError aclrtEventElapsedTime(float *ms, aclrtEvent startEvent, aclrtEvent endEvent)
Python函数	ms, ret = acl.rt.event_elapsed_time(start, end)
函数功能	统计两个Event之间的耗时。
输入说明	start : int，指定起始Event对象的指针地址。 end : int，指定结尾Event对象的指针地址。
返回值说明	ms : float，表示的是两个Event之间的耗时，单位是毫秒。 ret : int，错误码。 • 返回0表示成功。 • 返回其它值表示失败。
约束说明	接口调用顺序：调用 acl.rt.create_event / acl.rt.create_event_with_flag 接口创建event-->调用 acl.rt.record_event 接口在同一个stream中记录起始event、结尾event-->调用 acl.rt.synchronize_stream 接口阻塞应用程序运行，直到指定Stream中的所有任务都完成-->调用 acl.rt.event_elapsed_time 接口统计两个Event之间的耗时。

2.7.4.13 函数：stream_wait_event

C函数原型	<code>aclError aclrtStreamWaitEvent(aclrtStream stream, aclrtEvent event)</code>
Python函数	<code>ret = acl.rt.stream_wait_event(stream, event)</code>
函数功能	阻塞指定Stream的运行，直到指定的Event完成，支持多个Stream等待同一个Event的场景。异步接口。 提交到Stream上的所有后续任务都需要等待Event捕获的任务都完成后才能开始执行。具体见acl.rt.record_event接口了解Event捕获的细节。
输入说明	stream: int, 指定需要等待Event完成的Stream的指针地址。如果使用默认Stream，此处设置为0。 event: int, 需等待的Event的指针地址。
返回值说明	ret: int, 错误码。 • 返回0表示成功。 • 返回其它值表示失败。
约束说明	• 该接口是异步接口，调用接口成功仅表示任务下发成功，不表示任务执行成功。调用该接口后，需调用同步等待接口（例如，acl.rt.synchronize_stream）确保任务已执行完成。 • acl.rt.record_event接口与acl.rt.stream_wait_event接口配合使用时，主要用于多Stream之间同步的场景，在调用acl.rt.record_event接口时，系统内部会申请Event资源，在调用acl.rt.stream_wait_event接口之后，请及时调用acl.rt.reset_event接口释放Event资源。 接口调用顺序：acl.rt.create_event-->acl.rt.record_event-->acl.rt.stream_wait_event-->acl.rt.reset_event • 一个进程内，调用acl.init接口初始化pyACL后，调用acl.rt.set_op_wait_timeout接口设置超时时间，本进程内后续调用acl.rt.stream_wait_event接口下发的任务支持在所设置的超时时间内等待，若等待的时间超过所设置的超时时间，则pyACL会返回报错。 由于acl.rt.stream_wait_event接口是异步接口，调用接口成功仅表示任务下发成功，不表示任务执行成功，因此若等待的时间超过所设置的超时时间，则在调用acl.rt.synchronize_stream接口后，会返回报错。
参考资源	接口调用示例，参见关于Event的同步等待、关于Stream间任务的同步等待。

2.7.4.14 函数：set_op_wait_timeout

C函数原型	<code>aclError aclrtSetOpWaitTimeout(uint32_t timeout)</code>
Python函数	<code>ret = acl.rt.set_op_wait_timeout(timeout)</code>

函数功能	本接口用于设置等待Event完成的超时时间。 Atlas 推理系列产品 ，不支持该接口。
输入说明	timeout : 设置超时时间，单位为秒。将该参数设置为0时，表示不超时。
返回值说明	ret : int, 错误码。 - 返回0表示成功。 - 返回其它值表示失败。
约束说明	- 不调用本接口，则默认不超时，一个进程内多次调用本接口，则以最后一次设置的时间为准。 - 一个进程内，调用acl.init接口初始化pyACL后，调用本接口设置超时时间，本进程内后续调用acl.rt.stream_wait_event接口下发的任务支持在所设置的超时时间内等待。由于acl.rt.stream_wait_event接口是异步接口，调用接口成功仅表示任务下发成功，不表示任务执行成功，因此若等待的时间超过所设置的超时时间，则在调用acl.rt.synchronize_stream接口后，会返回报错。

2.7.5 执行控制

2.7.5.1 函数：subscribe_report

C函数原型	aclError aclrtSubscribeReport(uint64_t threadId, aclrtStream stream)
Python函数	ret = acl.rt.subscribe_report(thread_id, stream)
函数功能	异步任务场景下，指定处理Stream上回调函数的线程。 本接口需与以下其它接口配合使用，以便实现异步场景下的Callback功能： - 新建线程，在线程函数内，调用acl.rt.process_report接口设置超时时间（需循环调用），等待acl.rt.launch_callback接口下发的函数回调任务。 - 调用acl.rt.subscribe_report接口建立第1步中的线程和Stream的绑定关系，该Stream下发的函数回调任务将在绑定的线程中执行。 - 在指定Stream上执行异步任务（例如异步推理任务）。 - 定义并实现回调函数，调用acl.rt.launch_callback接口在Stream的任务队列中下发函数回调任务，触发acl.rt.subscribe_report订阅的线程处理回调函数，每调用一次acl.rt.launch_callback接口，就会下发一个回调函数任务。 - 异步任务全部执行完成后，调用acl.rt.unsubscribe_report接口取消线程订阅，解除线程和Stream的绑定关系。

输入说明	thread_id : int, 指定线程id。 stream : int, 指定需要处理的Stream的指针地址。
返回值说明	ret : int, 错误码。 • 返回0表示成功。 • 返回其它值表示失败。
约束说明	• 支持多次调用acl.rt.subscribe_report接口给多个Stream（仅支持同一Device内的多个Stream）订阅同一个处理回调函数的线程。 • 为确保Stream内的任务按调用顺序执行，不支持调用acl.rt.subscribe_report接口给同一个Stream订阅多个处理回调函数的线程。 • 在Atlas 200/300/500 推理产品上，单进程内调用acl.rt.subscribe_report接口订阅的线程数量如果超过128个，则接口返回失败。 • 在Atlas 训练系列产品上，单进程内调用acl.rt.subscribe_report接口订阅的线程数量如果超过1024个，则接口返回失败。 • 在Atlas A2训练系列产品/Atlas 800I A2推理产品上，单进程内调用acl.rt.subscribe_report接口订阅的线程数量如果超过1024个，则接口返回失败。 • 在Atlas 推理系列产品上，单进程内调用acl.rt.subscribe_report接口订阅的线程数量如果超过1024个，则接口返回失败。 • 在Atlas 200I/500 A2推理产品上，单进程内调用acl.rt.subscribe_report接口订阅的线程数量如果超过1024个，则接口返回失败。 • 考虑操作系统的线程切换性能开销，建议调用acl.rt.subscribe_report接口订阅的线程数量控制在32个以下（包括32）。 • 同一个进程内，在不同的Device上订阅回调函数的线程时，不能指定同一个线程ID。
参考资源	• 接口调用流程，参见接口调用流程。 • 接口调用示例，参见示例代码。

2.7.5.2 函数：launch_callback

C函数原型	aclError aclrtLaunchCallback(aclrtCallback fn, void *userData, aclrtCallbackBlockType blockType, aclrtStream stream)
Python函数	ret = acl.rt.launch_callback(fn, user_data_list, block_type, stream)

函数功能	异步任务场景下，在Stream的任务队列中下发一个函数回调任务，系统内部在执行到该回调任务时，会在Stream上订阅的线程（通过acl.rt.subscribe_report接口注册的线程）中执行回调函数。异步接口。 本接口需与以下其它接口配合使用，以便实现异步场景下的Callback功能： 1. 新建线程，在线程函数内，调用acl.rt.process_report接口设置超时时间（需循环调用），等待acl.rt.launch_callback接口下发的函数回调任务。 2. 调用acl.rt.subscribe_report接口建立第1步中的线程和Stream的绑定关系，该Stream下发的函数回调任务将在绑定的线程中执行。 3. 在指定Stream上执行异步任务（例如异步推理任务）。 4. 定义并实现回调函数，调用acl.rt.launch_callback接口在Stream的任务队列中下发函数回调任务，触发acl.rt.subscribe_report订阅的线程处理回调函数，每调用一次acl.rt.launch_callback接口，就会下发一个回调函数任务。 5. 异步任务全部执行完成后，调用acl.rt.unsubscribe_report接口取消线程订阅，解除线程和Stream的绑定关系。
输入说明	fn: function，表示Python侧的回调函数。 user_data_list: list，表示需要传递给回调函数的参数（目前传入的是list类型的数据）。 block_type: int，指定线程是否阻塞。 • 0：表示非阻塞。Atlas 200/300/500 推理产品，当前版本不支持该选项，该选项预留。 • 1：表示阻塞。 stream: int，表示指定的Stream。
返回值说明	ret: int，错误码。 • 返回0表示成功。 • 返回其它值表示失败。
约束说明	该接口是异步接口，调用接口成功仅表示任务下发成功，不表示任务执行成功。调用该接口后，需调用同步等待接口（例如，acl.rt.synchronize_stream）确保任务已执行完成。
参考资源	• 接口调用流程，参见接口调用流程。 • 接口调用示例，参见示例代码。

2.7.5.3 函数：process_report

C函数原型	<code>aclError aclrtProcessReport(int32_t timeout)</code>
Python函数	<code>ret = acl.rt.process_report(timeout)</code>

函数功能	异步任务场景下，调用本接口设置超时时间，等待acl.rt.launch_callback接口下发的回调任务执行。 本接口需与以下其它接口配合使用，以便实现异步场景下的Callback功能： 1. 新建线程，在线程函数内，调用acl.rt.process_report接口设置超时时间（需循环调用），等待acl.rt.launch_callback接口下发的函数回调任务。 2. 调用acl.rt.subscribe_report接口建立第1步中的线程和Stream的绑定关系，该Stream下发的函数回调任务将在绑定的线程中执行。 3. 在指定Stream上执行异步任务（例如异步推理任务）。 4. 定义并实现回调函数，调用acl.rt.launch_callback接口在Stream的任务队列中下发函数回调任务，触发acl.rt.subscribe_report订阅的线程处理回调函数，每调用一次acl.rt.launch_callback接口，就会下发一个回调函数任务。 5. 异步任务全部执行完成后，调用acl.rt.unsubscribe_report接口取消线程订阅，解除线程和Stream的绑定关系。
输入说明	timeout: int，指定超时时间，单位为ms。 取值范围为： • -1：表示无限等待。 • 大于0（不包含0）：表示等待的时间。
返回值说明	ret: int，错误码。 • 返回0表示成功。 • 返回其它值表示失败。
约束说明	无
资源参考	• 接口调用流程，参见接口调用流程。 • 接口调用示例，参见示例代码。

2.7.5.4 函数：unsubscribe_report

C函数原型	<code>aclError aclrtUnSubscribeReport(uint64_t threadId, aclrtStream stream)</code>
Python函数	<code>ret = acl.rt.unsubscribe_report(thread_id, stream)</code>

函数功能	异步任务场景下，取消线程订阅，Stream上的回调函数不再由指定线程处理。 本接口需与以下其它接口配合使用，以便实现异步场景下的Callback功能： 1. 新建线程，在线程函数内，调用acl.rt.process_report接口设置超时时间（需循环调用），等待acl.rt.launch_callback接口下发的函数回调任务。 2. 调用acl.rt.subscribe_report接口建立第1步中的线程和Stream的绑定关系，该Stream下发的函数回调任务将在绑定的线程中执行。 3. 在指定Stream上执行异步任务（例如异步推理任务）。 4. 定义并实现回调函数，调用acl.rt.launch_callback接口在Stream的任务队列中下发函数回调任务，触发acl.rt.subscribe_report订阅的线程处理回调函数，每调用一次acl.rt.launch_callback接口，就会下发一个回调函数任务。 5. 异步任务全部执行完成后，调用acl.rt.unsubscribe_report接口取消线程订阅，解除线程和Stream的绑定关系。
输入说明	thread_id: int，指定线程id。 stream: int，指定需要处理的Stream的指针地址。
返回值说明	ret: int，错误码。 • 返回0表示成功。 • 返回其它值表示失败。
约束说明	无
参考资源	• 接口调用流程，参见接口调用流程。 • 接口调用示例，参见示例代码。

2.7.6 内存管理

2.7.6.1 总体说明

- 若涉及媒体数据处理功能，关于内存使用，有以下注意事项：
 - a. 由于媒体数据处理功能对存放输入、输出数据的内存有更高的要求（例如，内存首地址128对齐），因此需调用专用的内存申请接口，如下。
 - 调用[媒体数据处理V1](#)版本的接口对图片进行抠图、缩放等操作时，调用[acl.media.dvpp_malloc](#)接口申请内存。
 - 调用[媒体数据处理V2](#)版本的接口对图片进行抠图、缩放等操作时，调用[acl.himpi.dvpp_malloc](#)接口申请内存。
 - b. 调用上述接口申请出来的内存可以满足媒体数据处理的要求，也可以在其它任务中使用，例如，从性能角度，为了减少拷贝，媒体数据处理的输出作为模型推理的输入，实现内存复用。

- c. 由于媒体数据处理访问的地址空间有限，为确保媒体数据处理时内存足够，除媒体数据处理功能外的其它功能（例如，模型加载），建议调用[内存管理](#)章节下的[acl.rt.malloc](#)接口、[acl.rt.malloc_host](#)接口或[acl.rt.malloc_cached](#)接口申请内存。

Atlas 推理系列产品上，媒体数据处理功能在每个进程内，可访问的地址空间最大为16GB。

Atlas A2训练系列产品/Atlas 800I A2推理产品上，媒体数据处理功能在每个进程内，可访问的地址空间最大为16GB。

- 对于**Atlas 200/300/500 推理产品**，若涉及申请Device上的大页内存，当前版本为防止大页内存不足，系统内部已提前预留（*Device操作系统上的可用内存* * 25% ≈ 1940M）大页内存，在申请大页内存时请优先使用这部分预留内存。
Device操作系统上的可用内存指使用**free**命令查询得到的**total**内存。
 - 在标准形态下，使用完预留的大页内存后，系统内部会根据Device上的内存动态扩展或缩减大页内存。
- 对于**Atlas A2训练系列产品/Atlas 800I A2推理产品**，若涉及申请Device上的大页内存，需注意，当前版本为防止大页内存不够用，系统内部已提前预留大页内存，用户在使用大页内存前，可调用[acl.rt.get_mem_info](#)接口，查询空闲的片上内存大页内存（对应ACL_HBM_MEM_HUGE类型）、片上内存普通内存（对应ACL_HBM_MEM_NORMAL类型）。
- Ascend EP形态下，Device启动后首次执行推理或训练任务时，会将AI CPU算子从Host搬迁到Device上并缓存在Device上，用于提升性能，因此会占用部分Device内存（100M~200M，不同芯片有所不同）。若Device重启，这部分AI CPU算子缓存才会释放。

以下产品涉及该约束：

- Atlas 推理系列产品
- Atlas 200I/500 A2推理产品
- Atlas 训练系列产品
- Atlas A2 训练系列产品

2.7.6.2 函数：malloc

C函数原型	<code>aclError aclrtMalloc(void **devPtr, size_t size, aclrtMemMallocPolicy policy)</code>
Python函数	<code>dev_ptr, ret = acl.rt.malloc(size, policy)</code>
函数功能	在Device上分配size大小的线性内存，并通过“dev_ptr”返回已分配内存的指针地址。本接口分配的内存会进行字节对齐，会对用户申请的size向上对齐成32字节整数倍后再多加32字节。
输入说明	size : int, 申请内存的大小，单位Byte，size不能为0。 policy : int, 内存分配规则，具体请参见 2.16.5 aclrtMemMallocPolicy 。
返回值说明	dev_ptr : int, 指向Device上已分配内存的指针地址。 ret : int, 错误码。 • 返回0表示成功。 • 返回其它值表示失败。

约束说明	• 由于媒体数据处理功能对存放输入、输出数据的内存有更高的要求（例如，内存首地址128对齐），因此需调用专用的内存申请接口，如下。 - 调用媒体数据处理V1版本的接口对图片进行抠图、缩放等操作时，调用acl.media.dvpp_malloc接口申请内存。 - 调用媒体数据处理V2版本的接口对图片进行抠图、缩放等操作时，调用acl.himpi.dvpp_malloc接口申请内存。 • 本接口分配的内存不会进行对内容的初始化，建议在使用内存前先调用acl.rt.memset接口先初始化内存，清除内存中的随机数。 • 本接口内部不会进行隐式的Device同步或流同步。如果申请内存成功或申请内存失败会立刻返回结果。 • 使用acl.rt.malloc接口申请的内存，需要通过acl.rt.free接口释放内存。 • 频繁调用acl.rt.malloc接口申请内存、调用acl.rt.free接口释放内存，会损耗性能，建议用户提前做内存预先分配或二次管理，避免频繁申请/释放内存。 • 调用acl.rt.malloc接口申请内存时，会对用户输入的size按向上对齐成32字节整数倍后，再加32字节。 若用户使用本接口申请大块内存并自行划分、管理内存时，每段内存需同时满足以下需求： - 内存大小向上对齐成32整数倍加32字节（$m = \text{ALIGN\_UP}[\text{len}, 32] + 32$字节）。 - 内存起始地址需满足64字节对齐（$\text{ALIGN\_UP}[m, 64]$）。 说明 len表示某段内存的大小，$\text{ALIGN\_UP}[\text{len}, k]$表示向上按k字节对齐：$((\text{len} - 1) / k + 1) * k$。
参考资源	接口调用示例，参见数据传输。

2.7.6.3 函数：malloc_cached

C函数原型	<code>aclError aclrtMallocCached(void **devPtr, size_t size, aclrtMemMallocPolicy policy)</code>
Python函数	<code>dev_ptr, ret = acl.rt.malloc_cached(size, policy)</code>
函数功能	申请Device上的内存，该接口在任何场景下申请的内存都是支持cache缓存。在Device上申请size大小的线性内存，通过dev_ptr返回已分配内存的指针地址。 使用acl.rt.malloc_cached接口申请的内存与使用acl.rt.malloc接口申请的内存是等价的，都支持cache缓存，不需要用户处理cpu与npu之间的cache一致性。 调用媒体数据处理的接口前，若需要申请Device上的内存存放输入或输出数据，需调用acl.media.dvpp_malloc申请内存。

输入说明	size : int, 申请内存的大小, 单位Byte。size不能为0。 policy : int, 内存分配规则, 具体请参见 2.16.5 aclrtMemMallocPolicy 。
返回值说明	dev_ptr : int, 指向Device上已分配内存的指针地址。 ret : int, 错误码。 - 返回0表示成功。 - 返回其它值表示失败。
约束说明	其它约束与 acl.rt.malloc 接口相同。

2.7.6.4 函数: mem_flush

C函数原型	aclError aclrtMemFlush(void *devPtr, size_t size)
Python函数	ret = acl.rt.mem_flush(dev_ptr, size)
函数功能	将cache中的数据刷新到DDR中并将cache中的内容设置成无效。 该版本用户无需处理CPU与NPU之间的cache一致性, 无需调用该接口。
输入说明	dev_ptr : int, 待flush的DDR内存起始地址。 size : int, 待flush的DDR内存大小, 单位Byte。“size”不能为0。
返回值说明	ret : int, 错误码。 - 返回0表示成功。 - 返回其它值表示失败。
约束说明	无

2.7.6.5 函数: mem_invalidate

C函数原型	aclError aclrtMemInvalidate(void *devPtr, size_t size)
Python函数	ret = acl.rt.mem_invalidate(dev_ptr, size)
函数功能	将cache中的数据设置成无效。 该版本不需要用户处理CPU与NPU之间的cache一致性, 无需调用该接口。
输入说明	dev_ptr : int, 需要指定DDR内存对应的cache无效。 size : int, DDR内存大小, 单位Byte。“size”不能为0。
返回值说明	ret : int, 错误码。 - 返回0表示成功。 - 返回其它值表示失败。

约束说明	无

2.7.6.6 函数：get_mem_info

C函数原型	<code>aclError aclrtGetMemInfo(aclrtMemAttr attr, size_t *free, size_t *total)</code>
Python函数	<code>free, total, ret = acl.rt.get_mem_info(attr)</code>
函数功能	获取指定属性的pyACL应用可用内存的空闲大小和总大小，不包括系统预留内存大小。
输入说明	attr : int，需要查询的内存的属性值，具体请参见 2.16.39 aclrtMemAttr 。
返回值说明	free : int，对应属性的内存的空闲大小，单位Byte。 total : int，对应属性的内存的总大小，单位Byte。 ret : int，错误码。 - 返回0表示成功。 - 返回其它值表示失败。
约束说明	- 调用本接口前是必须先指定用于计算的Device（例如调用acl.rt.set_device接口指定用于计算的Device），因此本接口中不体现Device ID。 Atlas 200/300/500 推理产品 请根据实际硬件支持的情况，选择对应属性的内存，否则调用本接口获取到的空闲大小和总大小都为0。 Atlas 训练系列产品 请根据实际硬件支持的情况，选择对应属性的内存，否则调用本接口获取到的空闲大小和总大小都为0。 Atlas A2训练系列产品/Atlas 800I A2推理产品 请根据实际硬件支持的情况，选择对应属性的内存，否则调用本接口获取到的空闲大小和总大小都为0。 Atlas 200I/500 A2推理产品 由于Device上没有片上内存相关的内存，若指定查询片上内存相关内存，则pyACL系统内部自动转换为查询DDR相关内存，例如，若查询ACL_HBM_MEM，则pyACL系统内部实际查询ACL_DDR_MEM；若查询ACL_HBM_MEM_HUGE，则pyACL系统内部实际查询ACL_DDR_MEM_HUGE。 Atlas 推理系列产品 由于Device上没有片上内存相关的内存，若指定查询片上内存相关内存，则pyACL系统内部自动转换为查询DDR相关内存，例如，若查询ACL_HBM_MEM，则pyACL系统内部实际查询ACL_DDR_MEM；若查询ACL_HBM_MEM_HUGE，则pyACL系统内部实际查询ACL_DDR_MEM_HUGE。

2.7.6.7 函数：free

C函数原型	<code>aclError aclrtFree(void *devPtr)</code>
Python函数	<code>ret= acl.rt.free(dev_ptr)</code>
函数功能	释放Device上的内存。
输入说明	dev_ptr : int, 待释放内存的指针地址。
返回值说明	ret : int, 错误码。 • 返回0表示成功。 • 返回其它值表示失败。
约束说明	• <code>acl.rt.free</code>接口只能释放通过acl.rt.malloc接口或acl.rt.malloc_cached接口。 • 本接口会立刻释放传入的内存，函数内部不会进行隐式的Device同步或流同步。用户需要确保调用完本接口后不再对该内存指针进行访问。
参考资源	接口调用示例，参见数据传输。

2.7.6.8 函数：malloc_host

C函数原型	<code>aclError aclrtMallocHost(void **hostPtr, size_t size)</code>
Python函数	<code>host_ptr, ret= acl.rt.malloc_host(size)</code>
函数功能	pyACL应用程序在Host上运行时，调用该接口申请的是Host内存（该内存是锁页内存），由系统保证内存首地址64字节对齐。 pyACL应用程序在Device上运行时，调用该接口申请的是Device内存，且Device上的内存按普通页申请，如需首地址64字节对齐，需要用户自行处理对齐。
输入说明	size : int, 申请内存的大小，单位Byte。size不能为0。
返回值说明	host_ptr : int, 指向Host上已分配内存的指针的指针地址。 ret : int, 错误码。 • 返回0表示成功。 • 返回其它值表示失败。

约束说明	● 本接口分配的内存不会对内容进行初始化, 建议在使用内存前先调用acl.rt.memset接口先初始化内存, 清除内存中的随机数。 ● 本接口内部不会进行隐式的Device同步或流同步。如果申请内存成功或申请内存失败会立刻返回结果。 ● 使用acl.rt.malloc_host接口申请的内存, 需要通过acl.rt.free_host接口释放内存。 ● 使用acl.rt.malloc_host接口分配过多的锁页内存, 将导致操作系统用于分页的物理内存变少, 从而降低系统整体的性能。 ● 频繁调用acl.rt.malloc_host接口申请内存、调用acl.rt.free_host接口释放内存, 会损耗性能, 建议用户提前做内存预先分配或二次管理, 避免频繁申请/释放内存。 ● 若用户使用本接口申请大块内存并自行划分、管理内存时, 每段内存需同时满足以下需求: - 内存大小向上对齐成32整数倍加32字节 ($m = \text{ALIGN\_UP}[\text{len}, 32] + 32$ 字节) 。 - 内存起始地址需满足64字节对齐 ($\text{ALIGN\_UP}[m, 64]$) 。 说明 len表示某段内存的大小, $\text{ALIGN\_UP}[\text{len}, k]$表示向上按k字节对齐: $((\text{len} - 1) / k + 1) * k$。
参考资源	接口调用示例, 参见数据传输。

2.7.6.9 函数: free_host

C函数原型	<code>aclError aclrtFreeHost(void *hostPtr)</code>
Python函数	<code>ret= acl.rt.free_host(host_ptr)</code>
函数功能	释放Host上的内存。
输入说明	host_ptr : int, 待释放内存的指针地址。
返回值说明	ret : int, 错误码。 ● 返回0表示成功。 ● 返回其它值表示失败。
约束说明	● 本接口会立刻释放传入的内存, 函数内部不会进行隐式的Device同步或流同步。用户需要确保调用完本接口后不再对该内存指针地址进行访问。 ● acl.rt.free_host接口只能释放通过acl.rt.malloc_host接口申请的内存。
参考资源	接口调用示例, 参见数据传输。

2.7.6.10 函数：memset

C函数原型	<code>aclError aclrtMemset(void *devPtr, size_t maxCount, int32_t value, size_t count)</code>
Python函数	<code>ret = acl.rt.memset(dev_ptr, max_count, value, count)</code>
函数功能	初始化内存，将内存中的内容设置为指定的值。 待初始化的内存支持在Host侧或Device侧，系统根据地址判定是Host还是Device。
输入说明	dev_ptr : int，内存的起始地址。 max_count : int，内存的最大长度，单位Byte。 value : int，设置的值。 count : int，需要设置为指定值的内存长度，单位Byte。
返回值说明	ret : int，错误码。 - 返回0表示成功。 - 返回其它值表示失败。
约束说明	本接口会立刻进行内存初始化，函数内部不会进行隐式的Device同步或流同步。

2.7.6.11 函数：memset_async

C函数原型	<code>aclError aclrtMemsetAsync(void *devPtr, size_t maxCount, int32_t value, size_t count, aclrtStream stream)</code>
Python函数	<code>ret = acl.rt.memset_async(dev_ptr, max_count, value, count, stream)</code>
函数功能	初始化内存，将内存中的内容设置为指定的值，异步接口。 待初始化的内存支持在Host侧或Device侧，系统根据地址判定是Host还是Device。
输入说明	dev_ptr : int，内存的起始地址。 max_count : int，内存的最大长度，单位Byte。 value : int，设置的值。 count : int，需要设置为指定值的内存长度，单位Byte。 stream : int，指定Stream的指针地址。
返回值说明	ret : int，错误码。 - 返回0表示成功。 - 返回其它值表示失败。
约束说明	该接口是异步接口，调用接口成功仅表示任务下发成功，不表示任务执行成功。调用该接口后，必须调用同步等待接口（例如， acl.rt.synchronize_stream ）确保内存初始化的任务已执行完成。

2.7.6.12 函数：memcpy

C函数原型	aclError aclrtMemcpy(void *dst, size_t destMax, const void *src, size_t count, aclrtMemcpyKind kind)
Python函数	ret = acl.rt.memcpy(dst, dest_max, src, count, kind)
函数功能	实现内存复制。
输入说明	dst : int, 目的内存地址指针地址。 dest_max : int, 目的内存地址的最大内存长度, 单位Byte。 src : int, 源内存地址指针地址。 count : int, 内存复制的长度, 单位Byte。 kind : int, 内存复制的类型, 预留参数, 配置枚举值中的值无效, 系统内部会根据源内存地址的指针地址、目的内存地址的指针地址判断是否可以将源地址的数据复制到目的地址, 如果不可以, 则系统会返回报错。 0: ACL_MEMCPY_HOST_TO_HOST, Host内的内存复制。 1: ACL_MEMCPY_HOST_TO_DEVICE, Host到Device的内存复制。 2: ACL_MEMCPY_DEVICE_TO_HOST, Device到Host的内存复制。 3: ACL_MEMCPY_DEVICE_TO_DEVICE, Device内的内存复制。
返回值说明	ret : int, 错误码。 - 返回0表示成功。 - 返回其它值表示失败。
约束说明	本接口会立刻进行内存复制, 函数内部不会进行隐式的Device同步或流同步。
参考资源	接口调用示例, 参见数据传输。

2.7.6.13 函数：memcpy_async

C函数原型	aclError aclrtMemcpyAsync(void *dst, size_t destMax, const void *src, size_t count, aclrtMemcpyKind kind, aclrtStream stream)
Python函数	ret = acl.rt.memcpy_async(dst, dest_max, src, count, kind, stream)
函数功能	实现内存复制, 异步接口。

输入说明	dst: int, 目的内存地址的指针地址。 dest_max: int, 目的内存地址的最大内存长度, 单位Byte。 src: int, 源内存地址的指针地址。 count: int, 内存复制的长度, 单位Byte。 kind: int, 内存复制的类型。 0: ACL_MEMCPY_HOST_TO_HOST, Host内的内存复制。 1: ACL_MEMCPY_HOST_TO_DEVICE, Host到Device的内存复制。 2: ACL_MEMCPY_DEVICE_TO_HOST, Device到Host的内存复制。 3: ACL_MEMCPY_DEVICE_TO_DEVICE, Device内的内存复制。 stream: int, stream ID。
返回值说明	ret: int, 错误码。 - 返回0表示成功。 - 返回其它值表示失败。
约束说明	- 调用本接口进行内存复制时, 源地址和目的地址都必须64字节对齐。 - 本接口是异步接口, 调用接口成功仅表示任务下发成功, 不表示任务执行成功。调用该接口后, 一定要调用acl.rt.synchronize_stream接口确保内存复制的任务已执行完成。 - 若调用本接口传入的kind为“ACL_MEMCPY_HOST_TO_DEVICE”或“ACL_MEMCPY_DEVICE_TO_HOST”, 并且Host内存不是通过acl.rt.malloc_host申请得到的, 本接口不会报错, 但是在实际执行到该任务时会发生不可预知的错误。 - 本接口不支持异步Host内的内存复制功能, 因此传入的kind为“ACL_MEMCPY_HOST_TO_HOST”时, 任务下发成功, 但系统内部处理该任务时会返回失败。 Atlas 200/300/500 推理产品 - Ascend RC形态下, 在板端运行应用时, 选择涉及Host的内存复制类型后, 系统内部会默认使用Device内的内存复制。
参考资源	接口调用示例, 参见数据传输。

2.7.6.14 函数: memcpy2d

C函数原型	<pre>aclError aclrtMemcpy2d(void *dst, size_t dpitch, const void *src, size_t spitch, size_t width, size_t height, aclrtMemcpyKind kind)</pre>

Python函数	ret = acl.rt.memcpy2d(dst, dpitch, src, spitch, width, height, kind)
函数功能	实现同步内存复制，主要用于矩阵数据的复制。 Atlas 推理系列加速模块产品不支持该接口。
输入说明	dst : 目的内存地址的指针地址。 dpitch : 目的内存中相邻两列向量的地址距离。 src : 源内存地址的指针地址。 spitch : 源内存中相邻两列向量的地址距离。 width : 待复制的矩阵宽度。 height : 待复制的矩阵高度。 kind : 内存复制的类型。 ACL_MEMCPY_HOST_TO_HOST = 0 // Host内的内存复制 ACL_MEMCPY_HOST_TO_DEVICE = 1 // Host到Device的内存复制 ACL_MEMCPY_DEVICE_TO_HOST = 2 // Device到Host的内存复制 ACL_MEMCPY_DEVICE_TO_DEVICE = 3 // Device内的内存复制
返回值说明	ret : int, 错误码。 - 返回0表示成功。 - 返回其它值表示失败。
约束说明	- 当前仅支持“ACL_MEMCPY_HOST_TO_DEVICE”类型和“ACL_MEMCPY_DEVICE_TO_HOST”类型的内存复制。 - Atlas 200/300/500 推理产品，Ascend RC形态下，不支持调用本接口。 - Atlas 200I/500 A2推理产品，Ascend RC形态下，不支持调用本接口。

2.7.6.15 函数：memcpy2d_async

C函数原型	aclError aclrtMemcpy2dAsync(void *dst, size_t dpitch, const void *src, size_t spitch, size_t width, size_t height, aclrtMemcpyKind kind, aclrtStream stream)
Python函数	ret = acl.rt.memcpy2d_async(dst, dpitch, src, spitch, width, height, kind, stream)
函数功能	实现异步内存复制，主要用于矩阵数据的复制。异步接口。 Atlas 推理系列加速模块产品不支持该接口。

输入说明	dst: 目的内存地址的指针地址。 dpitch: 目的内存中相邻两列向量的地址距离。 src: 源内存地址的指针地址。 spitch: 源内存中相邻两列向量的地址距离。 width: 待复制的矩阵宽度。 height: 待复制的矩阵高度。 “height”最大可设置为$5 * 1024 * 1024 = 5242880$，否则接口返回失败。 kind: 内存复制的类型。 ACL_MEMCPY_HOST_TO_HOST = 0 // Host内的内存复制 ACL_MEMCPY_HOST_TO_DEVICE = 1 // Host到Device的内存复制 ACL_MEMCPY_DEVICE_TO_HOST = 2 // Device到Host的内存复制 ACL_MEMCPY_DEVICE_TO_DEVICE = 3 // Device内的内存复制 stream: 指定Stream的指针地址。
返回值说明	ret: int, 错误码。 • 返回0表示成功。 • 返回其它值表示失败。
约束说明	• 该接口是异步接口，调用接口成功仅表示任务下发成功，不表示任务执行成功。调用该接口后，需调用同步等待接口（例如，acl.rt.synchronize_stream）确保内存复制的任务已执行完成。 • 当前仅支持“ACL_MEMCPY_HOST_TO_DEVICE”类型和“ACL_MEMCPY_DEVICE_TO_HOST”类型的内存复制。 • Atlas 200/300/500 推理产品，Ascend RC形态下，不支持调用本接口。 • Atlas 200I/500 A2推理产品，Ascend RC形态下，不支持调用本接口。

2.7.6.16 函数：malloc_physical

C函数原型	<code>aclError aclrtMallocPhysical(aclrtDrvMemHandle *handle, size_t size, const aclrtPhysicalMemProp *prop, uint64_t flags)</code>
Python函数	<code>handle, ret = acl.rt.malloc_physical(size, prop, flags)</code>

函数功能	申请Device物理内存并返回一个物理内存handle。 本接口可配合acl.rt.reserve_mem_address接口（申请虚拟内存）、acl.rt.map_mem接口（建立虚拟内存与物理内存之间的映射）使用，以便申请地址连续的虚拟内存、最大化利用物理内存地址的目的。 本接口可配合acl.rt.mem_export_to_shareable_handle接口（导出物理内存handle）、acl.rt.mem_import_from_shareable_handle（导入共享handle）使用，用于实现多进程之间的物理内存共享。同时，也支持在共享物理内存时，使用虚拟内存，请参见acl.rt.mem_export_to_shareable_handle接口处的说明。 Atlas 200/300/500 推理产品不支持该接口。
输入说明 acl.get_cann_attribute_list	size: int，物理地址空间大小，单位Byte。 先调用acl.rt.mem_get_allocation_granularity接口获取内存申请粒度，然后再调用本接口申请物理内存时size按获取到的内存申请粒度对齐，以便节约内存。 当前内存申请粒度固定2M，因此size当前需按照2M对齐。 prop: dict，物理内存属性信息，具体请参见2.16.36 aclrtPhysicalMemProp。 flags: 预留，当前仅支持设置为0。
返回值说明	ret: int，错误码。 • 返回0表示成功。 • 返回其它值表示失败。
约束说明	• Ascend RC形态不支持调用本接口。 • 当前版本仅支持“ACL_HBM_MEM_HUGE”（表示片上内存大页内存）、“ACL_HBM_MEM_NORMAL”（表示片上内存普通页内存）两种内存类型，若传入“ACL_HBM_MEM_NORMAL”类型，系统内部当前也按片上内存大页内存申请。 在Atlas 推理系列产品上，由于Device上没有片上内存内存，使用本接口申请“ACL_HBM_MEM_HUGE”类型的内存时，pyACL系统内部自动转换为申请“ACL_DDR_MEM_HUGE”类型的DDR内存。 在Atlas 200I/500 A2推理产品上，由于Device上没有片上内存内存，使用本接口申请“ACL_HBM_MEM_HUGE”类型的内存时，pyACL系统内部自动转换为申请“ACL_DDR_MEM_HUGE”类型的DDR内存。

2.7.6.17 函数：free_physical

C函数原型	aclError aclrtFreePhysical(aclrtDrvMemHandle handle)
Python函数	ret = acl.rt.free_physical(handle)

函数功能	释放通过 acl.rt.malloc_physical 接口申请的物理内存。 Atlas 200/300/500 推理产品 不支持该接口。
输入说明	handle : 待释放的物理地址信息handle。
返回值说明	ret : int, 错误码。 • 返回0表示成功。 • 返回其它值表示失败。
约束说明	• Ascend RC形态不支持调用本接口。 • 若该物理内存与虚拟内存存在映射关系, 则释放物理内存前, 需调用acl.rt.unmap_mem接口取消该物理内存与虚拟内存的映射。

2.7.6.18 函数: reserve_mem_address

C函数原型	aclError aclrtReserveMemAddress(void **virPtr, size_t size, size_t alignment, void *expectPtr, uint64_t flags)
Python函数	vir_ptr, ret = acl.rt.reserve_mem_address(size, alignment, expect_ptr, flags)
函数功能	预留虚拟内存。 本接口需与以下其它接口配合使用, 以便申请地址连续的虚拟内存、最大化利用物理内存的目的: 1. 申请虚拟内存 (acl.rt.reserve_mem_address接口) 。 2. 申请物理内存 (acl.rt.malloc_physical接口) 。 3. 将虚拟内存映射到物理内存 (acl.rt.map_mem接口) 。 4. 执行任务 (调用具体的任务接口) 。 5. 取消虚拟内存与物理内存的映射 (acl.rt.unmap_mem接口) 。 6. 释放物理内存 (acl.rt.free_physical接口) 。 7. 释放虚拟内存 (acl.rt.release_mem_address接口) 。
输入说明	size : int, 虚拟地址空间大小, 单位Byte。size不能为0。 alignment : int, 虚拟地址对齐值, 预留, 当前只能设置为0。 expect_ptr : 指定期望返回的虚拟地址空间起始地址, 预留, 当前只能传0。 flags : 大页/普通页标志, 此处的标志需与 acl.rt.malloc_physical 接口的内存类型保持一致。取值参考如下: • 0: 普通页。 • 1: 大页。

返回值说明	vir_ptr : int, 指向已分配的虚拟地址空间的地址。 ret : int, 错误码。 - 返回0表示成功。 - 返回其它值表示失败。
约束说明	Ascend RC形态不支持调用本接口。

2.7.6.19 函数: release_mem_address

C函数原型	aclError aclrtReleaseMemAddress(void *virPtr)
Python函数	ret = acl.rt.release_mem_address(vir_ptr)
函数功能	释放通过 acl.rt.reserve_mem_address 接口申请的虚拟内存。 本接口需与以下其它接口配合使用, 以便申请地址连续的虚拟内存、最大化利用物理内存的目的: 1. 申请虚拟内存 (acl.rt.reserve_mem_address接口) 。 2. 申请物理内存 (acl.rt.malloc_physical接口) 。 3. 将虚拟内存映射到物理内存 (acl.rt.map_mem接口) 。 4. 执行任务 (调用具体的任务接口) 。 5. 取消虚拟内存与物理内存的映射 (acl.rt.unmap_mem接口) 。 6. 释放物理内存 (acl.rt.free_physical接口) 。 7. 释放虚拟内存 (acl.rt.release_mem_address接口) 。
输入说明	vir_ptr : int, 指向待释放的虚拟地址空间的地址。
返回值说明	ret : int, 错误码。 - 返回0表示成功。 - 返回其它值表示失败。
约束说明	Ascend RC形态不支持调用本接口。 若该虚拟内存与物理内存存在映射关系, 则释放虚拟内存前, 需调用 acl.rt.unmap_mem 接口取消该虚拟内存与物理内存的映射。

2.7.6.20 函数: map_mem

C函数原型	aclError aclrtMapMem(void *virPtr, size_t size, size_t offset, aclrtDrvMemHandle handle, uint64_t flags)
Python函数	ret = acl.rt.map_mem(vir_ptr, size, offset, handle, flags)

函数功能	将虚拟内存映射到物理内存。 本接口需与以下其它接口配合使用，以便申请地址连续的虚拟内存、最大化利用物理内存的目的： 1. 申请虚拟内存（acl.rt.reserve_mem_address接口）。 2. 申请物理内存（acl.rt.malloc_physical接口）。 3. 将虚拟内存映射到物理内存（acl.rt.map_mem接口）。 4. 执行任务（调用具体的任务接口）。 5. 取消虚拟内存与物理内存的映射（acl.rt.unmap_mem接口）。 6. 释放物理内存（acl.rt.free_physical接口）。 7. 释放虚拟内存（acl.rt.release_mem_address接口）。
输入说明	vir_ptr: int，待映射的虚拟地址。该地址不一定为起始地址，用户也可以根据起始地址自行偏移后，再映射。虚拟地址当前需按照2M对齐。 size: int，待映射的内存大小，单位Byte。size当前需按照2M对齐。 offset: 物理内存偏移值，当前只能设置为0。 handle: 物理内存指针地址。 flags: 预留，当前仅支持设置为0。
返回值说明	ret: int，错误码。 • 返回0表示成功。 • 返回其它值表示失败。
约束说明	Ascend RC形态不支持调用本接口。

2.7.6.21 函数：unmap_mem

C函数原型	aclError aclrtUnmapMem(void *virPtr)
Python函数	ret = acl.rt.unmap_mem(vir_ptr)
函数功能	取消虚拟内存与物理内存之间的映射关系。 本接口需与以下其它接口配合使用，以便申请地址连续的虚拟内存、最大化利用物理内存的目的： 1. 申请虚拟内存（acl.rt.reserve_mem_address接口）。 2. 申请物理内存（acl.rt.malloc_physical接口）。 3. 将虚拟内存映射到物理内存（acl.rt.map_mem接口）。 4. 执行任务（调用具体的任务接口）。 5. 取消虚拟内存与物理内存的映射（acl.rt.unmap_mem接口）。 6. 释放物理内存（acl.rt.free_physical接口）。 7. 释放虚拟内存（acl.rt.release_mem_address接口）。

输入说明	vir_ptr : int，待取消映射的虚拟地址。
返回值说明	ret : int，错误码。 • 返回0表示成功。 • 返回其它值表示失败。
约束说明	Ascend RC形态不支持调用本接口。

2.7.6.22 函数：mem_export_to_shareable_handle

C函数原型	aclError aclrtMemExportToShareableHandle (aclrtDrvMemHandle handle, aclrtMemHandleType handleType, uint64_t flags, uint64_t *shareableHandle)
Python函数	shareable_handle, ret = acl.rt.mem_export_to_shareable_handle (handle, handle_type, flags)

函数功能	将本进程通过acl.rt.malloc_physical接口获取到的物理内存handle导出，以便后续将物理内存共享给其它进程。 本接口需与以下其它接口配合使用，以便实现内存共享的目的（此处以A、B进程为例，说明两个进程间的物理内存共享接口调用流程）： - 在A进程中： - 调用acl.rt.malloc_physical接口，申请物理内存。 先调用acl.rt.mem_get_allocation_granularity接口获取内存申请粒度，然后在调用acl.rt.malloc_physical接口申请物理内存时size按获取到的内存申请粒度对齐，以便节约内存。 若需申请地址连续的虚拟内存、最大化利用物理内存地址的目的，此处可配合acl.rt.reserve_mem_address、acl.rt.map_mem等接口申请虚拟内存、建立虚拟内存与物理内存之间的映射，请参见对应接口的说明。 - 调用acl.rt.mem_export_to_shareable_handle接口，导出物理内存handle，输出shareable_handle。 - 获取B进程的进程ID，并调用acl.rt.mem_set_pid_to_shareable_handle接口，将B进程的进程ID设置为白名单。 - 调用acl.rt.free_physical接口，释放物理内存。 涉及共享内存的进程都需要释放物理内存，所有涉及内存共享的进程都完成释放操作，物理内存才真正释放。释放物理内存后，引用的分配内存将释放回操作系统，之后使用handle将导致未定义的行为。 - 在B进程中： - 调用acl.rt.device_get_bare_tgid接口，获取B进程的进程ID。 本接口内部在获取进程ID时已适配物理机、虚拟机场景，用户只需调用本接口获取进程ID，再配置其它接口使用，达到物理内存共享的目的。若用户不调用本接口、自行获取进程ID，可能会导致后续使用进程ID异常。 - 调用acl.rt.mem_import_from_shareable_handle，获取shareable_handle里的信息，并返回本进程中的handle。 （在调用acl.rt.mem_import_from_shareable_handle接口前，需确保待共享的物理内存存在，不能提前释放。） 若需申请地址连续的虚拟内存、最大化利用物理内存地址的目的，此处可配合acl.rt.reserve_mem_address、acl.rt.map_mem等接口申请虚拟内存、建立虚拟内存与物理内存之间的映射，请参见对应接口的说明。 - 调用acl.rt.free_physical接口，释放物理内存。
输入说明	handle: int，存放物理内存信息的handle。 需先在本进程调用acl.rt.malloc_physical接口申请物理内存，该接口调用成功，会返回一个handle。 handle_type: int，预留参数，当前固定输入ACL_MEM_HANDLE_TYPE_NONE = 0。 flags: int，预留参数，当前固定填0。

返回值说明	shareable_handle : int, 标识共享给其它进程的shareable_handle。 ret : int, 错误码。 - 返回0表示成功。 - 返回其它值表示失败。
约束说明	- Ascend RC形态不支持调用本接口。 - 只支持Device侧的物理内存共享。 - 通过acl.rt.free_physical接口释放物理内存, 实现销毁shareable_handle。若有进程还在使用shareable_handle, 则等待shareable_handle使用完成后再执行销毁任务。 - handle与shareable_handle是一一对应的关系, 在同一个进程中, 不允许一对多、或多对一, 否则报错, 例如重复调用本接口导出时则会返回报错。 - 不支持算力切分和算力分组场景。

2.7.6.23 函数: device_get_bare_tgid

C函数原型	aclError aclrtDeviceGetBareTgid(int32_t *pid)
Python函数	pid, ret = acl.rt.device_get_bare_tgid()
函数功能	获取当前进程的进程ID。 本接口内部在获取进程ID时已适配物理机、虚拟机场景, 用户只需调用本接口获取进程ID, 再配置其它接口使用 (配合流程请参见 acl.rt.mem_export_to_shareable_handle), 达到物理内存共享的目的。若用户不调用本接口、自行获取进程ID, 可能会导致后续使用进程ID异常。
输入说明	无。
返回值说明	pid : int, 进程ID。 ret : int, 错误码。 - 返回0表示成功。 - 返回其它值表示失败。
约束说明	无

2.7.6.24 函数: mem_set_pid_to_shareable_handle

C函数原型	aclError aclrtMemSetPidToShareableHandle(uint64_t shareableHandle, int32_t *pid, size_t pidNum)
Python函数	ret = acl.rt.mem_set_pid_to_shareable_handle(shareable_handle, pid)

函数功能	设置可共享内存的进程白名单，只有配置了白名单的进程才能用 acl.rt.mem_export_to_shareable_handle 接口导出的shareable_handle。 本接口需与其它接口配合使用，以便实现内存共享的目的，请参见 acl.rt.mem_export_to_shareable_handle 接口处的说明。
输入说明	shareable_handle : int，通过 acl.rt.mem_export_to_shareable_handle 接口导出的shareable_handle。 pid : int，用于存放白名单进程ID的数组。 进程ID可调用 acl.rt.device_get_bare_tgid 接口获取，Docker场景下获取到的是物理机上的进程ID，非Docker场景下获取到的是进程ID。
返回值说明	ret : int，错误码。 - 返回0表示成功。 - 返回其它值表示失败。
约束说明	在调用 aclrtMemExportToShareableHandle 接口的进程中，调用本接口设置进程白名单。 Ascend RC形态不支持调用本接口。

2.7.6.25 函数：mem_import_from_shareable_handle

C函数原型	<code>aclError aclrtMemImportFromShareableHandle(uint64_t shareableHandle, int32_t deviceId, aclrtDrvMemHandle *handle)</code>
Python函数	<code>handle, ret = acl.rt.mem_import_from_shareable_handle(shareable_handle, device_id)</code>
函数功能	在本进程中获取shareable_handle里的信息，并返回本进程中的handle，用于在本进程中建立虚拟地址与物理地址之间的映射关系。本接口还支持生成指定Device上的handle。 本接口需与其它接口配合使用，以便实现内存共享的目的，配合使用流程请参见 acl.rt.mem_export_to_shareable_handle 接口处的说明。
输入说明	shareableHandle : int，待共享的shareable_handle。 与 acl.rt.mem_export_to_shareable_handle 接口中导出的shareable_handle保持一致。 deviceId : int，用于生成指定Device ID上的handle。 用户调用 acl.rt.get_device_count 接口获取可用的Device数量后，这个Device ID的取值范围：[0, (可用的Device数量 - 1)]

返回值说明	handle : int, 本进程的物理内存handle。 ret : int, 错误码。 • 返回0表示成功。 • 返回其它值表示失败。
约束说明	• Ascend RC形态不支持调用本接口。 • 在调用本接口前, 需确保待共享的物理内存存在, 不能提前释放。 • handle与shareable_handle是一一对应的关系, 在同一个进程中, 不允许一对多、或多对一, 否则报错, 例如重复调用本接口导入时则会返回报错。 • 不支持同一个进程中调用acl.rt.mem_import_from_shareable_handle、acl.rt.mem_export_to_shareable_handle这两个接口, 只支持跨进程调用。 • 支持在一个Device上调用acl.rt.mem_export_to_shareable_handle接口导出handle, 然后调用本接口生成另一个Device上的handle。 • 若调用acl.rt.free_physical销毁handle, 则需所有调用本接口的进程都销毁shareable_handle的情况下, handle才会正在销毁。

2.7.6.26 函数: mem_get_allocation_granularity

C函数原型	aclError aclrtMemGetAllocationGranularity(aclrtPhysicalMemProp *prop, aclrtMemGranularityOptions option, size_t *granularity)
Python函数	granularity, ret = acl.rt.mem_get_allocation_granularity(prop, option)
函数功能	查询内存申请粒度。系统内部计算给定分配规范的最小粒度或建议粒度, 并以granularity参数返回粒度。此粒度可用作对齐、地址大小或地址映射的倍数。
输入说明	prop : dict, 物理内存属性信息, 具体请参见 2.16.36 aclrtPhysicalMemProp 。 option : int, 最小粒度或推荐粒度, 具体请参见 2.16.43 aclrtMemGranularityOptions 。
返回值说明	granularity : int, 内存申请粒度, 单位为Byte, 当前只支持2M。 ret : int, 错误码。 • 返回0表示成功。 • 返回其它值表示失败。
约束说明	Ascend RC形态不支持调用本接口。

2.7.7 异常处理

2.7.7.1 函数：set_exception_info_callback

C函数原型	aclError aclrtSetExceptionInfoCallback(aclrtExceptionInfoCallback callback)
Python函数	ret = acl.rt.set_exception_info_callback(fn)
函数功能	设置异常回调函数。
输入说明	fn : function, 表示Python侧的回调函数, 格式如下: <pre>def exception_callback(exception_info) :exception_info: 表示异常信息aclrtExceptionInfo的指针地址 :return:</pre>
返回值说明	ret : int, 错误码。 • 返回0表示成功。 • 返回其它值表示失败。
约束说明	• 您需要在执行异步任务之前, 设置异常回调函数, 当Device上的任务执行异常时, 系统会向用户设置的异常回调函数中传入一个包含任务ID、Stream ID、线程ID、Device ID以及错误码的aclrtExceptionInfo的指针地址, 并执行回调函数, 用户可以再分别调用acl.rt.get_task_id_from_exception_info、acl.rt.get_stream_id_from_exception_info、acl.rt.get_thread_id_from_exception_info、acl.rt.get_device_id_from_exception_info、acl.rt.get_error_code_from_exception_info接口获取产生异常的任务ID、Stream ID、线程ID、Device ID以及错误码, 便于定位问题。 使用场景举例: 例如, 在调用acl.op.execute_v2接口前, 调用acl.rt.set_exception_info_callback接口设置异常回调函数, 当算子在Device执行异常时, 系统会向用户设置的异常回调函数中传入一个包含任务ID、Stream ID、线程ID、Device ID以及错误码的aclrtExceptionInfo的指针地址, 并执行回调函数。 • 如果多次设置异常回调函数, 以最后一次设置为准。 • 如果想清空回调函数, 可再次调用acl.rt.set_exception_info_callback接口进行重置 (传入None或不传入参数)。
参考资源	接口调用示例, 参见AI Core异常信息获取。

2.7.7.2 函数：get_task_id_from_exception_info

C函数原型	uint32_t aclrtGetTaskIdFromExceptionInfo(const aclrtExceptionInfo *info)

Python函数	task_id = acl.rt.get_task_id_from_exception_info(info)
函数功能	获取异常信息中的任务ID。该接口与 acl.rt.set_exception_info_callback 接口配合使用。
输入说明	info : int, 表示异常信息aclrtExceptionInfo的指针地址。 在执行任务之前调用 acl.rt.set_exception_info_callback 接口, 系统会将产生异常的任务ID、Stream ID、线程ID、Device ID存放在aclExceptionInfo中。
返回值说明	task_id : int, 返回异常信息中的任务ID, 返回值为“0xFFFFFFFF”（以十六进制为例）时表示异常信息为空。
约束说明	无
参考资源	接口调用示例, 参见AI Core异常信息获取。

2.7.7.3 函数: get_stream_id_from_exception_info

C函数原型	uint32_t aclrtGetStreamIdFromExceptionInfo(const aclrtExceptionInfo *info)
Python函数	stream_id = acl.rt.get_stream_id_from_exception_info(info)
函数功能	获取异常信息中的Stream ID。该接口与 acl.rt.set_exception_info_callback 接口配合使用。
输入说明	info : int, 表示异常信息aclrtExceptionInfo的指针地址。 在执行任务之前调用 acl.rt.set_exception_info_callback 接口, 系统会将产生异常的任务ID、Stream ID、线程ID、Device ID存放在aclExceptionInfo中。
返回值说明	stream_id : int, 返回异常信息中的Stream ID, 返回值为“0xFFFFFFFF”（以十六进制为例）时表示异常信息为空。
约束说明	无
参考资源	接口调用示例, 参见AI Core异常信息获取。

2.7.7.4 函数: get_thread_id_from_exception_info

C函数原型	uint32_t aclrtGetThreadIdFromExceptionInfo(const aclrtExceptionInfo *info)
Python函数	thread_id = acl.rt.get_thread_id_from_exception_info(info)
函数功能	获取异常信息中的线程ID。该接口与 acl.rt.set_exception_info_callback 接口配合使用。

输入说明	info : int, 表示异常信息 aclrtExceptionInfo 的指针地址。 在执行任务之前调用 acl.rt.set_exception_info_callback 接口, 系统会将产生异常的任务ID、Stream ID、线程ID、Device ID存放在 aclExceptionInfo 中。
返回值说明	thread_id : int, 返回异常信息中的线程ID, 返回值为“0xFFFFFFFF” (以十六进制为例) 时表示异常信息为空。
约束说明	无

2.7.7.5 函数: `get_device_id_from_exception_info`

C函数原型	uint32_t aclrtGetDeviceIdFromExceptionInfo(const aclrtExceptionInfo *info)
Python函数	device_id = acl.rt.get_device_id_from_exception_info(info)
函数功能	获取异常信息中的Device ID。
输入说明	info : int, 表示异常信息 aclrtExceptionInfo 的指针地址。 在执行任务之前调用 acl.rt.set_exception_info_callback 接口, 系统会将产生异常的任务ID、Stream ID、线程ID、Device ID存放在 aclExceptionInfo 中。
返回值说明	device_id : int, 返回异常信息中的Device设备号, 返回值为“0xFFFFFFFF” (以十六进制为例) 时表示异常信息为空。
约束说明	无
参考资源	接口调用示例, 参见AI Core异常信息获取。

2.7.7.6 函数: `get_error_code_from_exception_info`

C函数原型	uint32_t aclrtGetErrorCodeFromExceptionInfo(const aclrtExceptionInfo *info)
Python函数	ret = acl.rt.get_error_code_from_exception_info(info)
函数功能	获取异常信息中的错误码。该接口与 acl.rt.set_exception_info_callback 接口配合使用。
输入说明	info : int, 异常信息 aclrtExceptionInfo 的指针地址。 在执行任务之前调用 acl.rt.set_exception_info_callback 接口, 系统会将产生异常的任务ID、Stream ID、线程ID、Device ID、错误码存放在 aclExceptionInfo 中。
返回值说明	ret : int, 错误码。 - 返回0表示成功。 - 返回其它值表示失败, 返回值为“0xFFFFFFFF” (以十六进制为例) 时表示Device异常。

约束说明	无

2.8 模型加载与执行 (mdl)

2.8.1 模型加载与执行

2.8.1.1 函数: load_from_file

C函数原型	<code>aclError aclmdlLoadFromFile(const char *modelPath, uint32_t *modelId)</code>
Python函数	<code>model_id, ret = acl.mdl.load_from_file(model_path)</code>
函数功能	从文件加载离线模型数据（适配昇腾AI处理器的离线模型），由系统内部管理内存。 系统完成模型加载后，返回的模型ID，作为后续操作时用于识别模型的标志。
输入说明	model_path: str，离线模型文件的存储路径，路径中包含文件名。运行程序（APP）的用户需要对该存储路径有访问权限。 此处的离线模型文件是适配昇腾AI处理器的离线模型，即*.om文件。 说明 - 关于如何获取om文件，请参见模型构建。 - om模型文件大小有限制的场景下，使用ATC工具生成om文件时，如将“--external_weight”参数设置为1（1表示将原始网络中的Const/Constant节点的权重保存在单独的文件中，且该文件保存在与om文件同级的“weight”目录下），那么在使用本接口加载om文件时，需将“weight”目录与om文件放在同级目录下，pyACL会进入“weight”目录查找权重文件，未正确放置“weight”目录可能会导致单独的权重文件加载不成功。
返回值说明	model_id: int，系统完成模型加载后生成的模型ID对应的指针地址。 ret: int，错误码。 - 返回0表示成功。 - 返回其它值表示失败。
约束说明	- 在加载前，请先根据文件的大小评估内存空间是否足够，内存空间不足，会导致应用程序异常。 - 模型加载、模型执行、模型卸载的操作必须在同一个Context下（关于Context的创建请参见acl.rt.set_device、acl.rt.create_context）。

相关接口	当前pyACL还提供了 acl.mdl.set_config_opt 接口、 acl.mdl.load_with_config 接口来实现模型加载，通过配置对象中的属性来区分，在加载模型时是从文件加载，还是从内存加载，以及内存是由系统内部管理，还是由用户管理。
参考资源	接口调用流程，参见模型加载。

2.8.1.2 函数：load_from_mem

C函数原型	aclError aclmdlLoadFromMem(const void* model, size_t modelSize, uint32_t* modelId)
Python函数	model_id, ret = acl.mdl.load_from_mem(model, model_size)
函数功能	从内存加载离线模型数据，由系统内部管理模型运行的内存。系统完成模型加载后，返回的模型ID，作为后续操作时用于识别模型的标志。
输入说明	model : int，模型数据的内存地址对应的指针地址。 - 应用运行在Host时，此处需申请Host上的内存。 - 应用运行在Device时，此处需申请Device上的内存。 内存申请接口请参见 2.7.6 内存管理 。 model_size : int，内存中的模型数据长度，单位Byte。
返回值说明	model_id : int，系统完成模型加载后生成的模型ID。 ret : int，错误码。 - 返回0表示成功。 - 返回其它值表示失败。
约束说明	模型加载、模型执行、模型卸载的操作必须在同一个Context下（关于Context的创建请参见 acl.rt.set_device 、 acl.rt.create_context ）。
相关接口	当前pyACL还提供了 acl.mdl.set_config_opt 接口、 acl.mdl.load_with_config 接口来实现模型加载，通过配置对象中的属性来区分，在加载模型时是从文件加载，还是从内存加载，以及内存是由系统内部管理，还是由用户管理。
参考资源	接口调用流程，参见模型加载。

2.8.1.3 函数：load_from_file_with_mem

C函数原型	aclError aclmdlLoadFromFileWithMem(const char *modelPath, uint32_t *modelId, void *workPtr, size_t workSize, void *weightPtr, size_t weightSize)

Python函数	model_id, ret = acl.mdl.load_from_file_with_mem(model_path, work_ptr, work_size, weight_ptr, weight_size)
函数功能	从文件加载离线模型数据（适配昇腾AI处理器的离线模型），由用户自行管理模型运行的内存。 系统完成模型加载后，返回的模型ID，作为后续操作时用于识别模型的标志。
输入说明	model_path : str，离线模型文件的存储路径，路径中包含文件名。运行程序（APP）的用户需要对该存储路径有访问权限。 此处的离线模型文件是适配昇腾AI处理器的离线模型，即*.om文件。 说明 关于如何获取om文件，请参见模型构建。 work_ptr : int，Device上模型所需工作内存（存放模型输入/输出等数据）的指针地址，由用户自行管理，模型执行过程中不能释放该内存。如果在该参数处传入0，表示由系统管理内存。 说明 由 用户自行管理工作内存 时，如果多个模型串行执行，可共用同一个工作内存，但用户需确保模型的串行执行顺序，工作内存的大小需按多个模型中最大工作内存的大小来申请，例如通过以下方式保证串行： • 同步模型执行时，加锁，保证执行任务串行。 • 异步模型执行时，使用同一个Stream，保证执行任务串行。 work_size : int，模型所需工作内存的大小，单位Byte。 “work_ptr”为0时无效。 weight_ptr : int，Device上模型权值内存（存放权值数据）的指针地址，由用户自行管理，模型执行过程中不能释放该内存。如果在该参数处传入0，表示由系统管理内存。 说明 使用 用户自行管理权值内存 时，在多线程场景下，对于同一个模型，如果在每个线程中都加载了一次，可选择共用“weight_ptr”的方式，因为“weight_ptr”内存在推理过程中是只读的。 此处需注意，在共用“weight_ptr”期间，不能释放“weight_ptr”。 weight_size : int，模型所需权值内存的大小，单位Byte。 “weight_ptr”为0时无效。
返回值说明	model_id : int，系统完成模型加载后生成的模型ID。 ret : int，错误码。 • 返回0表示成功。 • 返回其它值表示失败。
约束说明	模型加载、模型执行、模型卸载的操作必须在同一个Context下（关于Context的创建请参见 acl.rt.set_device 、 acl.rt.create_context ）。
相关接口	当前pyACL还提供了 acl.mdl.set_config_opt 接口、 acl.mdl.load_with_config 接口来实现模型加载，通过配置对象中的属性来区分，在加载模型时是从文件加载，还是从内存加载，以及内存是由系统内部管理，还是由用户管理。

参考资源	接口调用流程，参见模型加载。

2.8.1.4 函数：load_from_mem_with_mem

C函数原型	aclError aclmdlLoadFromMemWithMem(const void *model, size_t modelSize, uint32_t *modelId, void *workPtr, size_t workSize, void *weightPtr, size_t weightSize)
Python函数	model_id, ret = acl.mdl.load_from_mem_with_mem(model, model_size, work_ptr, work_size, weight_ptr, weight_size)
函数功能	从内存加载离线模型数据，由用户自行管理模型运行的内存。 系统完成模型加载后，返回的模型ID，作为后续操作时用于识别模型的标志。
输入说明	model : int，模型数据的内存地址。 - 应用运行在Host时，此处需申请Host上的内存。 - 应用运行在Device时，此处需申请Device上的内存。 - 应用的运行模式可使用acl.rt.get_run_mode接口获取。 model_size : int，模型数据长度，单位Byte。 work_ptr : int，Device上模型所需工作内存（存放模型输入/输出等数据）的指针地址，由用户自行管理，模型执行过程中不能释放该内存。如果在该参数处传入0，表示由系统管理内存。 说明 由用户自行管理工作内存时，如果多个模型串行执行，可共用同一个工作内存，但用户需确保模型的串行执行顺序，工作内存的大小需按多个模型中最大工作内存的大小来申请，例如通过以下方式保证串行： - 同步模型执行时，加锁，保证执行任务串行。 - 异步模型执行时，使用同一个Stream，保证执行任务串行。 work_size : int，模型所需工作内存的大小，单位Byte。 “work_ptr”为0时无效。 weight_ptr : int，Device上模型权值内存（存放权值数据）的指针地址，由用户自行管理，模型执行过程中不能释放该内存。如果在“weight_ptr”参数处传入0，表示由系统管理内存。 说明 使用用户自行管理权值内存时，在多线程场景下，对于同一个模型，如果在每个线程中都加载了一次，可选择共用“weight_ptr”的方式，因为“weight_ptr”内存推理过程中是只读的。 此处需注意，在共用“weight_ptr”期间，不能释放“weight_ptr”。 weight_size : int，模型所需权值内存的大小，单位Byte。 “weight_ptr”为0时无效。
返回值说明	model_id : int，系统完成模型加载后生成的模型ID。 ret : int，错误码。 - 返回0表示成功。 - 返回其它值表示失败。

约束说明	模型加载、模型执行、模型卸载的操作必须在同一个Context下（关于Context的创建请参见 acl.rt.set_device 、 acl.rt.create_context ）。
相关接口	当前pyACL还提供了 acl.mdl.set_config_opt 接口、 acl.mdl.load_with_config 接口来实现模型加载，通过配置对象中的属性来区分，在加载模型时是从文件加载，还是从内存加载，以及内存是由系统内部管理，还是由用户管理。
参考资源	接口调用流程，参见模型加载。

2.8.1.5 函数：execute

C函数原型	aclError aclmdlExecute(uint32_t modelId, const aclmdlDataset *input, aclmdlDataset *output)
Python函数	ret = acl.mdl.execute(model_id, input, output)
函数功能	执行模型推理，直到返回推理结果。
输入说明	model_id: int，指定需要执行推理的模型的ID。 可在以下接口成功加载模型后获取到模型ID。 • acl.mdl.load_from_file • acl.mdl.load_from_mem • acl.mdl.load_from_file_with_mem • acl.mdl.load_from_mem_with_mem input: int，模型推理的输入数据对应的指针地址，具体请参见2.8.3 aclmdlDataset。 output: int，模型推理的输出数据对应的指针地址，具体请参见2.8.3 aclmdlDataset。 调用acl.create_data_buffer接口创建存放对应index输出数据的aclDataBuffer类型时，可通过“data”参数处传入0，同时设置“size”为0，创建一个空的aclDataBuffer类型，在模型执行过程中，系统内部自行计算并申请该index输出的内存。 使用该方式可节省内存，但在内存数据使用结束后，需由用户释放内存并重置aclDataBuffer，并且系统内部在申请内存时涉及内存拷贝，可能涉及性能损耗。 释放内存并重置aclDataBuffer的示例代码如下： <pre>data_buffer = acl.mdl.get_dataset_buffer(output, 0) // 根据index获取对应的dataBuffer data_addr = acl.get_data_buffer_addr(data_buffer) // 获取data的Device指针地址 acl.rt.free(data_addr) // 释放Device内存 acl.update_data_buffer(data_buffer, 0, 0) // 重置dataBuffer，以便下次推理</pre>
返回值说明	ret: int，错误码。 • 返回0表示成功。 • 返回其它值表示失败。

约束说明	● 若由于业务需求，必须在多线程中使用同一个“model_id”，则用户线程间需加锁，保证刷新输入输出内存、保证执行是连续操作，例如： // 线程A的接口调用顺序： lock(handle1) -> acl.rt.memcpy刷新输入输出内存 -> acl.mdl.execute执行推理 -> unlock(handle1) // 线程B的接口调用顺序： lock(handle1) -> acl.rt.memcpy刷新输入输出内存 -> acl.mdl.execute执行推理 -> unlock(handle1) ● 模型加载、模型执行、模型卸载的操作必须在同一个Context下（关于Context的创建请参见acl.rt.create_context）。 ● 存放模型输入/输出数据的内存，可以使用以下接口申请：acl.rt.malloc、acl.rt.malloc_host、acl.rt.malloc_cached接口、acl.media.dvpp_malloc接口或acl.himpi.dvpp_malloc接口。 其中： - 内存申请接口的具体使用场景及约束请参见相关接口的详细说明。 - 当pyACL应用程序在Device上调用acl.rt.malloc_host接口申请内存时，该内存为Device内存。 - acl.media.dvpp_malloc接口和acl.himpi.dvpp_malloc接口是媒体数据处理功能专用的内存申请接口，一般从性能角度，为了减少拷贝，媒体数据处理的输出作为模型推理的输入，实现内存复用。 - 由于硬件对内存有对齐和补齐要求，若用户使用这些接口申请大块内存并自行划分、管理内存时，需满足对应接口的对齐和补齐约束，请参见内存二次分配管理。
参考资源	接口调用流程及示例代码，参见模型执行。

2.8.1.6 函数：execute_v2

C函数原型	<code>aclError aclmdlExecuteV2(uint32_t modelId, const aclmdlDataset *input, aclmdlDataset *output, aclrtStream stream, const aclmdlExecConfigHandle *handle)</code>
Python函数	<code>ret = acl.mdl.execute_v2(model_id, input, output, stream, handle)</code>
函数功能	根据 acl.mdl.set_exec_config_opt 接口所配置的属性，执行模型推理，直到返回推理结果。该接口是在 acl.mdl.execute 接口基础上进行了增强，支持在执行模型推理时控制Stream任务的超时时间、Event任务的超时时间。

输入说明	model_id: int, 指定需要执行推理的模型的ID。 可在以下接口成功加载模型后获取到模型ID。 • acl.mdl.load_from_file • acl.mdl.load_from_mem • acl.mdl.load_from_file_with_mem • acl.mdl.load_from_mem_with_mem input: int, 模型推理的输入数据的指针地址, 具体请参见2.8.3 aclmdlDataset。 output: int, 模型推理的输出数据的指针地址, 具体请参见2.8.3 aclmdlDataset。 调用acl.create_data_buffer接口创建存放对应index输出数据的aclDataBuffer类型时, 可通过“data”参数处传入0, 同时设置“size”为0, 创建一个空的aclDataBuffer类型, 在模型执行过程中, 系统内部自行计算并申请该index输出的内存。 使用该方式可节省内存, 但内存数据使用结束后, 需由用户释放内存并重置aclDataBuffer, 并且系统内部申请内存时涉及内存拷贝, 可能涉及性能损耗。 释放内存并重置aclDataBuffer的示例代码如下: <pre>data_buffer = acl.mdl.get_dataset_buffer(output, 0) // 根据index获取对应的dataBuffer data_addr = acl.get_data_buffer_addr(data_buffer) // 获取data的Device指针地址 acl.rt.free(data_addr) // 释放Device内存 acl.update_data_buffer(data_buffer, 0, 0) // 重置dataBuffer, 以便下次推理</pre> stream: int, 指定已创建的Stream指针地址, 如需指定新的Stream, 可通过acl.rt.create_stream等接口创建并获取Stream指针地址。 handle: int, 模型执行的配置对象的指针地址, 与acl.mdl.set_exec_config_opt中的handle保持一致。
返回值说明	ret: int, 错误码。 • 返回0表示成功。 • 返回其它值表示失败。
约束说明	• 本接口需要配合其它接口一起使用, 实现模型执行, 接口调用顺序如下: 1. 调用acl.mdl.create_exec_config_handle接口创建模型执行的配置对象的指针地址。 2. 多次调用acl.mdl.set_exec_config_opt接口设置配置对象中每个属性的值。 3. 调用acl.mdl.execute_v2接口指定模型执行时需要的配置信息, 并进行模型执行。 4. 模型加载成功后, 调用acl.mdl.destroy_exec_config_handle接口销毁。 • 其他约束与acl.mdl.execute一致。
参考资源	• 接口调用流程及示例代码, 参见模型执行。 • Stream创建与管理, 请参见Stream管理。

2.8.1.7 函数: execute_async

C函数原型	<code>aclError aclmdlExecuteAsync(uint32_t modelId, const aclmdlDataset *input, aclmdlDataset *output, aclrtStream stream)</code>
Python函数	<code>ret = acl.mdl.execute_async(model_id, input, output, stream)</code>
函数功能	执行模型推理，异步接口。
输入说明	model_id: int，指定需要执行推理的模型的ID。 可在以下接口成功加载模型后获取到模型ID。 • acl.mdl.load_from_file • acl.mdl.load_from_mem • acl.mdl.load_from_file_with_mem • acl.mdl.load_from_mem_with_mem input: int，模型推理的输入数据的指针地址，具体请参见2.8.3 aclmdlDataset。 output: int，模型推理的输出数据的指针地址，具体请参见2.8.3 aclmdlDataset。 stream: int，指定已创建的Stream指针地址，如需指定新的Stream，可通过acl.rt.create_stream等接口创建并获取Stream指针地址。
返回值说明	ret: int，错误码。 • 返回0表示成功。 • 返回其它值表示失败。

约束说明	- 该接口是异步接口，调用接口成功仅表示任务下发成功，不表示任务执行成功。调用该接口后，需调用同步等待接口（例如，acl.rt.synchronize_stream）确保任务已执行完成。 - 对同一个“model_id”的模型，不可通过acl.mdl.execute_async接口执行多Stream并发场景下的模型推理。 - 如需在多线程中使用同一个“model_id”，则用户线程间需加锁，保证刷新输入输出内存、保证执行是连续操作，例如： // 线程A的接口调用顺序： lock(handle1) -> acl.rt.memcpy_async(stream1)刷新输入输出内存 -> acl.mdl.execute_async(modelId1,stream1)执行推理 -> unlock(handle1) // 线程B的接口调用顺序： lock(handle1) -> acl.rt.memcpy_async(stream1)刷新输入输出内存 -> acl.mdl.execute_async(modelId1,stream1)执行推理 -> unlock(handle1) - 模型加载、模型执行、模型卸载的操作必须在同一个Context下（关于Context的创建请参见acl.rt.create_context）。 - 存放模型输入/输出数据的内存，可以使用以下接口申请：acl.rt.malloc、acl.rt.malloc_host、acl.rt.malloc_cached接口、acl.media.dvpp_malloc接口或acl.himpi.dvpp_malloc接口。 其中，acl.media.dvpp_malloc接口和acl.himpi.dvpp_malloc接口是媒体数据处理功能专用的内存申请接口，一般从性能角度，为了减少拷贝，媒体数据处理的输出作为模型推理的输入，实现内存复用。 由于硬件对内存有对齐和补齐要求，若用户使用这些接口申请大块内存并自行划分、管理内存时，需满足对应接口的对齐和补齐约束，具体请参见内存二次分配管理。
参考资源	- 接口调用流程，参见接口调用流程。 - 接口调用示例，参见示例代码。 - Stream创建与管理，请参见Stream管理。

2.8.1.8 函数：unload

C函数原型	<code>aclError aclmdlUnload(uint32_t modelId)</code>
Python函数	<code>ret = acl.mdl.unload(model_id)</code>
函数功能	系统完成模型推理后，可调用该接口卸载模型，释放资源。
输入说明	model_id : int，指定需卸载的模型的ID。
返回值说明	ret : int，错误码。 - 返回0表示成功。 - 返回其它值表示失败。

约束说明	- 在调用acl.mdl.unload接口卸载指定模型时，需确保该模型当前未被其他接口使用。 - 模型加载、模型执行、模型卸载的操作必须在同一个Context下（关于Context的创建请参见acl.rt.set_device、acl.rt.create_context）。
参考资源	接口调用流程，参见pyACL接口调用流程。

2.8.1.9 函数：query_size

C函数原型	aclError aclmdlQuerySize(const char *fileName, size_t *workSize, size_t *weightSize)
Python函数	work_size, weight_size, ret = acl.mdl.query_size(file_name)
函数功能	根据模型文件获取模型执行时所需的权值内存大小、工作内存大小。
输入说明	file_name : str，需要获取内存信息的模型路径，路径中包含文件名。运行程序（APP）的用户需要对该路径有访问权限。
返回值说明	work_size : int，模型执行时所需的工作内存的大小，单位Byte。 weight_size : int，模型执行时所需权值内存的大小，单位Byte。 ret : int，错误码。 - 返回0表示成功。 - 返回其它值表示失败。
约束说明	当由用户管理内存时，为节省内存资源，在申请工作内存、权值内存前，需要调用 acl.mdl.query_size 接口查询模型运行时所需工作内存、权值内存的大小。 如果模型输入数据的Shape不确定，则不能调用 acl.mdl.query_size 接口查询内存大小，在加载模型时，就无法由用户管理内存，此时需选择由系统管理内存的模型加载接口（例如， acl.mdl.load_from_file 、 acl.mdl.load_from_mem ）。
参考资源	接口调用流程，参见模型加载。

2.8.1.10 函数：query_size_from_mem

C函数原型	aclError aclmdlQuerySizeFromMem(const void *model, size_t modelSize, size_t *workSize, size_t *weightSize)
Python函数	work_size, weight_size, ret = acl.mdl.query_size_from_mem(model, model_size)
函数功能	根据内存中的模型数据获取模型执行时所需的权值内存大小、内存大小。

输入说明	model : int, 需要获取内存信息的模型数据对应的指针地址。 model_size : int, 模型数据长度, 单位Byte。
返回值说明	work_size : int, 模型执行时所需的工作内存的大小, 单位Byte。 weight_size : int, 模型执行时所需权值内存的大小, 单位Byte。 ret : int, 错误码。 • 返回0表示成功。 • 返回其它值表示失败。
约束说明	执行和权重内存为Device内存, 而且需要用户申请和释放。

2.8.1.11 函数: set_dynamic_batch_size

C函数原型	aclError aclmdlSetDynamicBatchSize(uint32_t modelId, aclmdlDataset *dataset, size_t index, uint64_t batchSize)
Python函数	ret = acl.mdl.set_dynamic_batch_size(model_id, dataset, index, batch_size)
函数功能	在动态Batch场景下, 在模型执行前调用本接口设置模型推理时的批量大小Batch (每次处理图片的数量)。
输入说明	model_id : int, 模型ID。 可在以下接口成功加载模型后获取到模型ID。 • acl.mdl.load_from_file • acl.mdl.load_from_mem • acl.mdl.load_from_file_with_mem • acl.mdl.load_from_mem_with_mem dataset : int, 表示模型的输入数据对应的指针地址。 使用aclmdlDataset类型的数据描述模型推理时的输入数据, 输入的内存地址、内存大小用aclDataBuffer类型的数据来描述, 具体请参见 2.8.3 aclmdlDataset 。 index : int, 标识动态Batch输入的输入index, 需调用 acl.mdl.get_input_index_by_name 接口获取, 动态Batch和动态分辨率输入的名称固定为“ascend_mbatch_shape_data”, 动态AIPP输入的名称固定为“ascend_dynamic_aipp_data”。 batch_size : int, 指定模型推理时的批量大小Batch。 此处设置的“batch_size”只能为模型构建时设置的Batch档位中的其中一档, 也可调用 acl.mdl.get_dynamic_batch 接口获取指定模型支持的Batch档位数以及每一档中的Batch数。
返回值说明	ret : int, 错误码。 • 返回0表示成功。 • 返回其它值表示失败。
约束说明	无

参考资源	接口调用流程与示例，参见动态Batch/动态分辨率/动态维度（设置多档维度值）。

2.8.1.12 函数：set_dynamic_hw_size

C函数原型	<code>aclError aclmdlSetDynamicHWSize(uint32_t modelId, aclmdlDataset *dataset, size_t index, uint64_t height, uint64_t width)</code>
Python函数	<code>ret = acl.mdl.set_dynamic_hw_size(model_id, dataset, index, height, width)</code>
函数功能	动态分辨率场景下，在模型执行前调用本接口设置模型推理时输入图片的高和宽。
输入说明	model_id: int，模型ID。 可在以下接口成功加载模型后获取到模型ID。 • acl.mdl.load_from_file • acl.mdl.load_from_mem • acl.mdl.load_from_file_with_mem • acl.mdl.load_from_mem_with_mem dataset: 表示模型的输入数据。 使用aclmdlDataset类型的数据描述模型推理时的输入数据，输入的内存地址、内存大小用aclDataBuffer类型的数据来描述，具体请参见2.8.3 aclmdlDataset。 index: int，标识动态Batch输入的输入index，需调用acl.mdl.get_input_index_by_name接口获取，动态Batch和动态分辨率输入的名称固定为“ascend_mbatch_shape_data”，动态AIPP输入的名称固定为“ascend_dynamic_aipp_data”。 height: int，需设置的H值。 width: int，需设置的W值。 此处设置的分辨率（“height”与“width”）只能为模型构建时设置的分辨率档位中的其中一档，也可以调用acl.mdl.get_dynamic_hw接口获取指定模型支持的分辨率档位数以及每一档中的宽、高。
返回值说明	ret: int，错误码。 • 返回0表示成功。 • 返回其它值表示失败。
约束说明	无
参考资源	接口调用流程与示例，参见动态Batch/动态分辨率/动态维度（设置多档维度值）。

2.8.1.13 函数：create_and_get_op_desc

C函数原型	<code>aclError aclmdlCreateAndGetOpDesc(uint32_t deviceId, uint32_t streamId, uint32_t taskId, char *opName, size_t opNameLen, aclTensorDesc **inputDesc, size_t *numInputs, aclTensorDesc **outputDesc, size_t *numOutputs)</code>
Python函数	<code>op_name, input_desc, num_inputs, output_desc, num_outputs, ret = acl.mdl.create_and_get_op_desc(device_id, stream_id, task_id, op_name_len)</code>
函数功能	获取指定算子的描述信息，包括算子名、输入Tensor描述、输出Tensor描述，如果查询不到指定算子，则返回报错。该接口不支持动态Shape场景。
输入说明	device_id: int, 指定Device ID。 调用acl.rt.get_device_id_from_exception_info接口获取异常信息中的Device ID，作为本接口的输入。 stream_id: int, 指定Stream ID。 调用acl.rt.get_stream_id_from_exception_info接口获取异常信息中的Stream ID，作为本接口的输入。 task_id: int, 指定Task ID。 调用acl.rt.get_task_id_from_exception_info接口获取异常信息中的Task ID，作为本接口的输入。 op_name_len: int, 算子名称字符串长度。 若用户指定的长度比实际算子名称的长度短，则返回报错。
返回值说明	op_name: string, 算子名称字符串。 input_desc: int, 算子所有输入的Tensor描述，指向一块连续内存的首地址。 num_inputs: int, 输入个数。 output_desc: int, 算子所有输出的Tensor描述，指向一块连续内存的首地址。 num_outputs: int, 输出个数。 ret: int, 错误码。 • 返回0表示成功。 • 返回其它值表示失败。

约束说明	使用场景举例：执行整网模型推理时（不支持动态Shape场景），如果产生AI Core报错，可以调用本接口获取报错算子的描述信息，再做进一步错误排查。 推荐的接口调用顺序如下： 1. 定义并实现异常回调函数fn(aclrtExceptionInfoCallback类型)，回调函数原型请参见acl.rt.set_exception_info_callback。 实现回调函数的关键步骤如下： a. 在异常回调函数fn内调用acl.rt.get_device_id_from_exception_info、acl.rt.get_stream_id_from_exception_info、acl.rt.get_task_id_from_exception_info接口分别获取Device ID、Stream ID、Task ID。 b. 在异常回调函数fn内调用acl.mdl.create_and_get_op_desc接口获取算子的描述信息。 c. 在异常回调函数fn内调用acl.get_tensor_desc_by_index接口获取指定算子输入/输出的Tensor描述。 d. 在异常回调函数fn内参考如下接口获取Tensor描述中的数据，进行进一步分析。 例如，调用acl.get_tensor_desc_address接口获取Tensor数据的内存地址（用户可从该内存地址中获取Tensor数据）、调用acl.get_tensor_desc_type接口获取Tensor描述中的数据类型、调用acl.get_tensor_desc_format接口获取Tensor描述中的Format、调用acl.get_tensor_desc_num_dims接口获取Tensor描述中的Shape维度个数、调用acl.get_tensor_desc_dim_v2接口获取Shape中指定维度的大小。 2. 调用acl.rt.set_exception_info_callback接口设置异常回调函数。 3. 执行模型推理。 如果存在AI Core报错，则触发回调函数fn，获取算子的信息，进行进一步分析。
参考资源	接口调用示例，参见AI Core异常信息获取。

2.8.1.14 函数：init_dump

C函数原型	aclError aclmdlInitDump()
Python函数	ret = acl.mdl.init_dump()
函数功能	Dump初始化。
输入说明	无
返回值说明	ret : int，错误码。 • 返回0表示成功。 • 返回其它值表示失败。

约束说明	• acl.mdl.init_dump接口需要与acl.mdl.set_dump接口、acl.mdl.finalize_dump接口配合使用，用于将Dump数据记录到文件中。一个进程内，可以根据需求多次调用这些接口，基于不同的Dump配置信息，获取Dump数据。 • 场景举例： - 两次模型执行，需要设置不同的Dump配置信息，接口调用顺序：acl.init接口 --> acl.mdl.init_dump接口 --> acl.mdl.set_dump接口 --> 模型加载 --> 模型执行 --> acl.mdl.finalize_dump接口 --> 模型卸载 --> acl.mdl.init_dump接口 --> acl.mdl.set_dump接口 --> 模型加载 --> 模型执行 --> acl.mdl.finalize_dump接口 --> 模型卸载 --> 执行其它任务 --> acl.finalize接口 - 同一个模型执行两次，第一次需要Dump，第二次无需Dump，接口调用顺序：acl.init接口 --> acl.mdl.init_dump接口 --> acl.mdl.set_dump接口 --> 模型加载 --> 模型执行 --> acl.mdl.finalize_dump接口 --> 模型卸载 --> 模型加载 --> 模型执行 --> 执行其它任务 --> acl.finalize接口 • 对于模型Dump配置、单算子Dump配置、溢出算子Dump配置，如果已经通过acl.init接口配置了dump信息，则调用acl.mdl.init_dump接口时会返回失败。 • 必须在调用acl.init接口之后、模型加载接口之前调用acl.mdl.init_dump接口。
相关接口	pyACL还提供了 acl.init 接口，在pyACL初始化阶段，通过*.json文件传入Dump配置信息，运行应用后获取Dump数据的方式。该种方式，一个进程内，只能调用一次 acl.init 接口，如果要修改Dump配置信息，需修改*.json文件中的配置。

2.8.1.15 函数：set_dump

C函数原型	<code>aclError aclmdlSetDump(const char *dumpCfgPath)</code>
Python函数	<code>ret = acl.mdl.set_dump(dump_cfg_path)</code>
函数功能	设置dump参数。

输入说明	dump_cfg_path: string, 配置文件所在的路径, 包含文件名。 当前支持以下Dump信息配置: ● 模型Dump配置 (用于导出模型中每一层算子输入和输出数据)、单算子Dump配置 (用于导出单个算子的输入和输出数据), 导出的数据用于与指定模型或算子进行比对, 定位精度问题, 配置示例、说明及约束请参见配置文件示例 (模型Dump配置、单算子Dump配置)。默认不启用该dump配置。 ● 异常算子Dump配置 (用于导出异常算子的输入输出数据、workspace信息、Tiling信息), 导出的数据用于分析AI Core Error问题, 配置示例请参见配置文件示例 (异常算子Dump配置)。默认不启用该dump配置。 ● 溢出算子Dump配置 (用于导出模型中溢出算子的输入和输出数据), 导出的数据用于分析溢出原因, 定位模型精度的问题, 配置示例、说明及约束请参见配置文件示例 (溢出算子Dump配置)。默认不启用该dump配置。
返回值说明	ret: int, 错误码。 ● 返回0表示成功。 ● 返回其它值表示失败。

约束说明	• acl.mdl.init_dump接口需要与acl.mdl.set_dump接口、acl.mdl.finalize_dump接口配合使用，用于将Dump数据记录到文件中。一个进程内，可以根据需求多次调用这些接口，基于不同的Dump配置信息，获取Dump数据。 • 场景举例： - 两次模型执行，需要设置不同的Dump配置信息，接口调用顺序：acl.init接口 --> acl.mdl.init_dump接口 --> acl.mdl.set_dump接口 --> 模型加载 --> 模型执行 --> acl.mdl.finalize_dump接口 --> 模型卸载 --> acl.mdl.init_dump接口 --> acl.mdl.set_dump接口 --> 模型加载 --> 模型执行 --> acl.mdl.finalize_dump接口 --> 模型卸载 --> 执行其它任务 --> acl.finalize接口 - 同一个模型执行两次，第一次需要Dump，第二次无需Dump，接口调用顺序：acl.init接口 --> acl.mdl.init_dump接口 --> acl.mdl.set_dump接口 --> 模型加载 --> 模型执行 --> acl.mdl.finalize_dump接口 --> 模型卸载 --> 模型加载 --> 模型执行 --> 执行其它任务 --> acl.finalize接口 • 只有在调用本接口之后加载模型，配置的Dump信息有效。在调用本接口之前已经加载的模型不受影响，除非用户在调用本接口后重新加载该模型。 例如以下接口调用顺序中，加载的模型1不受影响，配置的Dump信息仅对加载的模型2有效： acl.mdl.init_dump接口 --> 模型1加载 --> acl.mdl.set_dump接口 --> 模型2加载 --> acl.mdl.finalize_dump接口 • 多次调用本接口对同一个模型配置了Dump信息，系统内处理时会采用覆盖策略。 例如以下接口调用顺序中，第二次调用本接口配置的Dump信息会覆盖第一次配置的Dump信息： acl.mdl.init_dump接口 --> acl.mdl.set_dump接口 --> acl.mdl.set_dump接口 --> 模型1加载 --> acl.mdl.finalize_dump接口
相关接口	pyACL还提供了 acl.init 接口，在pyACL初始化阶段，通过*.json文件传入Dump配置信息，运行应用后获取Dump数据的方式。该种方式，一个进程内，只能调用一次 acl.init 接口，如果要修改Dump配置信息，需修改*.json文件中的配置。

配置文件示例（模型 Dump 配置、单算子 Dump 配置）

模型Dump配置示例如下：

```
{
  "dump":{
    "dump_list":[
      {
        "model_name":"ResNet-101"
      },
      {
        "model_name":"ResNet-50",
        "layer":[
          "conv1conv1_relu",
          "res2a_branch2ares2a_branch2a_relu",
          "res2a_branch1",

```

```
        "pool1"  
    ]  
    }  
  ],  
  "dump_path": "$HOME/output",  
  "dump_mode": "output",  
  "dump_op_switch": "off",  
  "dump_data": "tensor"  
}  
}
```

单算子Dump配置示例如下：

```
{  
  "dump": {  
    "dump_path": "output",  
    "dump_list": [],  
    "dump_op_switch": "on",  
    "dump_data": "tensor"  
  }  
}
```

表 2-2 acl.json 文件格式说明

配置项	参数说明	备注
dump	-	- 不具有输出的TBE算子、AI CPU算子，如StreamActive、Send、Recv、const等不会生成dump数据；编译后的模型中部分算子并不会在AI CPU或AI Core执行，如concatD类型算子，则无法生成dump数据。 - 采用dump部分算子场景下，因data算子不会在AI CPU或AI Core上执行，如果用户填写dump data节点算子时需要一并填写data节点算子的后继节点，才能dump出data节点算子数据。
dump_list	（必填）待dump数据的整网模型列表。 创建模型dump配置信息，当存在多个模型需要dump时，需要每个模型之间用英文逗号隔开。 在单算子执行场景下，dump_list为空。	-

配置项	参数说明	备注
model_name	模型名称。 - 模型加载方式为文件加载时，填入模型文件的名称，不需要带后缀名；也可以配置为ATC模型文件转换后的json文件里的最外层"name"字段对应值。 - 模型加载方式为内存加载时，配置为ATC模型文件转换后的json文件里的最外层"name"字段对应值。	- 各个模型的model_name值须唯一。 - 模型加载支持内存和文件两种加载方式，文件加载且配置为ATC模型文件转换后的json文件里的最外层"name"字段对应值时，如果acl.json文件里model_name配置项值同时包括模型文件名、本处获取的name值，以模型文件名的配置项生效。
layer	算子名。可以指定为ATC模型转换后的算子名，也支持指定为转换前的原始算子名。 - 当需要dump指定的部分算子时，按格式配置layer字段，每行配置模型中的一个算子名，且每个算子之间用英文逗号隔开。若指定的算子其输入涉及data算子，会同时将data算子信息dump出来。 - 当需要dump模型的所有算子时，不需要包含layer字段。 在指定算子dump时，用户可以无需设置model_name，此时会默认dump所有model下的相应算子。如果配置了model_name，则dump对应model下的相应算子。	IO性能相对较差时，可能会出现由于数据量过大导致执行超时，所以不建议全量dump，请指定算子进行dump。
dump_path	（必填）dump数据文件存储到运行环境的目录。 支持配置绝对路径或相对路径： - 绝对路径配置以“/”开头，例如：\$HOME/output。 - 相对路径配置直接以目录名开始，例如：output。	该参数指定的目录需要提前创建且确保安装时配置的运行用户具有读写权限。

配置项	参数说明	备注
dump_mode	dump数据模式。 - input: dump算子的输入数据。 - output: dump算子的输出数据，默认取值output。 - all: dump算子的输入、输出数据。	-
dump_op_switch	单算子模型dump数据开关。 - on: 开启单算子模型dump。 - off: 关闭单算子模型dump，默认取值off。	-
dump_step	指定采集哪些迭代的Dump数据。推理场景无需配置。 不配置该参数，默认所有迭代都会产生dump数据，数据量比较大，建议按需指定迭代。 多个迭代用“ ”分割，例如：0 5 10；也可以用“-”指定迭代范围，例如：0 3-5 10。 配置示例： <pre>{ "dump":{ "dump_list": [.....], "dump_path":"\$HOME/output", "dump_mode":"output", "dump_op_switch":"off" "dump_step": "0 3-5 10" } }</pre>	-

配置项	参数说明	备注
dump_data	指定算子dump内容类型，取值： • tensor: dump算子数据，默认为tensor。 • stats: dump算子统计数据，结果文件为csv格式，文件中包含算子名称、输入/输出的数据类型、最大值、最小值等。 通常dump数据量太大并且耗时长，可以先dump算子统计数据，根据统计数据识别可能异常的算子，然后再dump算子数据。 模型Dump场景下，会根据dump_mode处的配置统计算子输入或算子输出或算子输入&输出的信息。	-

配置文件示例（异常算子 Dump 配置）

将“dump_scene”参数设置为“lite_exception”，启用异常算子Dump功能，同时可配合使用“ASCEND_WORK_PATH”环境变量配置落盘路径，否则落盘在应用程序的当前执行目录下。

配置文件中的示例内容如下：

```
{
  "dump":{
    "dump_scene":"lite_exception"
  }
}
```

说明

异常算子Dump配置，不能与模型Dump配置或单算子Dump配置同时开启，否则模型Dump或单算子Dump不生效。

配置文件示例（溢出算子 Dump 配置）

溢出算子Dump配置的相关约束说明如下：

- 将“dump_debug”配置为“on”表示开启溢出算子配置，不配置“dump_debug”或将“dump_debug”配置为“off”表示不开启溢出算子配置。
- 若开启溢出算子配置，则“dump_path”必须配置，表示导出数据文件的存储路径。

支持配置绝对路径或相对路径：

- 绝对路径配置以“/”开头，例如：/home。

- 相对路径配置直接以目录名开始，例如：output。

获取导出的数据文件后，如何解析请参见溢出算子数据采集及分析。

- 溢出算子Dump配置，不能与模型Dump配置或单算子Dump配置同时开启，否则会返回报错。

配置文件中的示例内容如下：

```
{
  "dump":{
    "dump_path":"output",
    "dump_debug":"on"
  }
}
```

2.8.1.16 函数：finalize_dump

C函数原型	<code>aclError aclmdlFinalizeDump()</code>
Python函数	<code>ret = acl.mdl.finalize_dump()</code>
函数功能	Dump去初始化。
输入说明	无
返回值说明	ret : int，错误码。 • 返回0表示成功。 • 返回其它值表示失败。
约束说明	• acl.mdl.init_dump接口需要与acl.mdl.set_dump接口、acl.mdl.finalize_dump接口配合使用，用于将Dump数据记录到文件中。一个进程内，可以根据需求多次调用这些接口，基于不同的Dump配置信息，获取Dump数据。 • 场景举例： - 两次模型执行，需要设置不同的的Dump配置信息，接口调用顺序：acl.init接口 --> acl.mdl.init_dump接口 --> acl.mdl.set_dump接口 --> 模型加载 --> 模型执行 --> acl.mdl.finalize_dump接口 --> 模型卸载 --> acl.mdl.init_dump接口 --> acl.mdl.set_dump接口 --> 模型加载 --> 模型执行 --> acl.mdl.finalize_dump接口 --> 模型卸载 --> 执行其它任务 --> acl.finalize接口 - 同一个模型执行两次，第一次需要Dump，第二次无需Dump，接口调用顺序：acl.init接口 --> acl.mdl.init_dump接口 --> acl.mdl.set_dump接口 --> 模型加载 --> 模型执行 --> acl.mdl.finalize_dump接口 --> 模型卸载 --> 模型加载 --> 模型执行 --> 执行其它任务 --> acl.finalize接口
相关接口	pyACL还提供了 acl.init 接口，在pyACL初始化阶段，通过*.json文件传入Dump配置信息，运行应用后获取Dump数据的方式。该种方式，一个进程内，只能调用一次 acl.init 接口，如果要修改Dump配置信息，需修改*.json文件中的配置。

2.8.1.17 函数：get_aipp_type

C函数原型	<code>aclError aclmdlGetAippType(uint32_t modelId, size_t index, aclmdlInputAippType *type, size_t *dynamicAttachedDataIndex)</code>
Python函数	<code>type, dynamic_attached_data_index, ret = acl.mdl.get_aipp_type(model_id, index)</code>
函数功能	获取指定模型的指定输入所支持的AIPP类型（动态AIPP或静态AIPP）及动态AIPP输入对应的index值。
输入说明	model_id : int, 指定模型的ID。 可在以下接口成功加载模型后获取到模型ID。 • acl.mdl.load_from_file • acl.mdl.load_from_mem • acl.mdl.load_from_file_with_mem • acl.mdl.load_from_mem_with_mem index : int, 模型中输入的index。
返回值说明	type : int, 表示指定模型输入的AIPP类型，具体请参见 2.16.23 aclmdlInputAippType 。 dynamic_attached_data_index : int, 返回信息。 • 当“type”不为ACL_DATA_WITH_DYNAMIC_AIPP时，该值返回“0xFFFFFFFF”，表示无效。 • 当“type”为ACL_DATA_WITH_DYNAMIC_AIPP时，该值返回动态AIPP输入（用于配置动态AIPP参数）的index。 ret : int, 错误码。 • 返回0表示成功。 • 返回其它值表示失败。
约束说明	无
参考资源	接口调用示例，参见动态AIPP（多个动态AIPP输入）。

2.8.1.18 函数：set_aipp_by_input_index

C函数原型	<code>aclError aclmdlSetAIPPByInputIndex(uint32_t modelId, aclmdlDataset *dataset, size_t index, const aclmdlAIPP *aippParmsSet)</code>
Python函数	<code>ret = acl.mdl.set_aipp_by_input_index(model_id, dataset_in, index, aipp)</code>

函数功能	动态AIPP场景下，根据指定的模型输入的输入index，设置模型推理时的AIPP参数值。 动态AIPP支持的几种操作的计算方式及其计算顺序如下： 1. 抠图 2. 色域转换 3. 减均值/归一化 4. 补边
输入说明	model_id: int，模型ID。 可在以下接口成功加载模型后获取到模型ID。 • acl.mdl.load_from_file • acl.mdl.load_from_mem • acl.mdl.load_from_file_with_mem • acl.mdl.load_from_mem_with_mem dataset_in: int，表示模型推理的输入数据的指针地址。 使用aclmdlDataset类型的数据描述模型推理时的输入数据，输入的内存地址、内存大小用aclDataBuffer类型的数据来描述，具体请参见2.8.3 aclmdlDataset。 index: int，表示在第几个输入上设置动态AIPP参数。如果该输入没有关联动态AIPP输入，则返回报错。 可调用acl.mdl.get_aipp_type查询指定模型的指定输入是否有关联的动态AIPP输入，如存在关联则本接口的index参数值与acl.mdl.get_aipp_type接口的index参数值保持一致；如不存在则无需设置动态AIPP参数。 aipp: int，动态AIPP参数对象的指针地址。 提前调用acl.mdl.create_aipp接口创建aclmdlAIPP类型的数据。
返回值说明	ret: int，错误码。 • 返回0表示成功。 • 返回其它值表示失败。
约束说明	• 经过动态AIPP处理后的图像的宽、高必须与原始模型输入Shape中的宽、高保持一致。 • 多Batch场景下，根据每个Batch的配置计算出动态AIPP后输出图片的宽、高，经过动态AIPP后每个Batch的输出图片宽、高必须是一致的。计算输出图片宽、高的计算公式如表2-3所示。 • 抠图或者缩放或者补边操作之后，对图片宽、高的校验规则如下，其中，aippOutputW、aippOutputH分别表示AIPP输出图片的宽、高，其它参数是acl.mdl.set_aipp_srcimage_size、acl.mdl.set_aipp_scf_params、acl.mdl.set_aipp_crop_params、acl.mdl.set_aipp_padding_params接口的入参，具体请参考表2-3。
参考资源	接口调用示例，参见动态AIPP（多个动态AIPP输入）。

表 2-3 输出图片宽、高计算公式

抠图	缩放	补边 (padding)	动态AIPP输出图片的宽、高
否	否	否	aippOutputW = srcImageSizeW aippOutputH = srcImageSizeH
是	否	否	aippOutputW = cropSizeW aippOutputH = cropSizeH
是	是	否	aippOutputW = scfOutputSizeW aippOutputH = scfOutputSizeH
是	否	是	aippOutputW = cropSizeW + paddingSizeLeft + paddingSizeRight aippOutputH = cropSizeH + paddingSizeTop + paddingSizeBottom
否	否	是	aippOutputW = srcImageSizeW + paddingSizeLeft + paddingSizeRight aippOutputH = srcImageSizeH + paddingSizeTop + paddingSizeBottom
否	是	是	aippOutputW = scfOutputSizeW + paddingSizeLeft + paddingSizeRight aippOutputH = scfOutputSizeH + paddingSizeTop + paddingSizeBottom
否	是	否	aippOutputW = scfOutputSizeW aippOutputH = scfOutputSizeH

抠图	缩放	补边 (padding)	动态AIPP输出图片的宽、高
是	是	是	aippOutputW = scfOutputSizeW + paddingSizeLeft + paddingSizeRight aippOutputH = scfOutputSizeH + paddingSizeTop + paddingSizeBottom

2.8.1.19 函数：set_input_dynamic_dims

C函数原型	aclError aclmdlSetInputDynamicDims(uint32_t modelId, aclmdlDataset *dataset, size_t index, const aclmdlIODims *dims)
Python函数	ret = acl.mdl.set_input_dynamic_dims(model_id, dataset_in, index, dims)
函数功能	如果模型输入的Shape是动态的、输入数据Format为ND格式（ND表示支持任意格式），在模型执行前调用本接口设置模型推理时具体维度的值。

输入说明	model_id: int, 模型ID。 可在以下接口成功加载模型后获取到模型ID。 • acl.mdl.load_from_file • acl.mdl.load_from_mem • acl.mdl.load_from_file_with_mem • acl.mdl.load_from_mem_with_mem dataset_in: int, 表示模型的输入数据。 使用aclmdlDataset类型的数据描述模型推理时的输入数据, 输入的内存地址、内存大小用aclDataBuffer类型的数据来描述, 2.8.3 aclmdlDataset。 index: int, 标识动态维度的输入index。 需调用acl.mdl.get_input_index_by_name接口获取, 输入名称固定为“ascend_mbatch_shape_data”。 dims: dict, 指定具体某一档的所有维度信息, 具体请参见2.16.18 aclmdlIODims。 可以调用acl.mdl.get_input_dynamic_dims接口获取指定模型支持的动态维度档位数以及每一档中的值。 例如: 在ATC工具进行模型转换时, 使用以下参数: input_shap="data:1,1,40,-1;label:1,-1;mask:-1,-1" , dynamic_dims="20,20,1,1; 40,40,2,2; 80,60,4,4" 若输入数据的真实维度为 (1,1,40,20,1,20,1,1), 则dims输入示例如下 (name暂不使用): dims = {'dimCount': 8, 'name': '', 'dims': [1,1,40,20,1,20,1,1]}
返回值说明	ret: int, 错误码 • 返回0表示成功。 • 返回其它值表示失败。
约束说明	无
参考资源	接口调用流程与示例, 参见 动态Batch/动态分辨率/动态维度 (设置多档维度值) 。

2.8.1.20 函数: create_config_handle

C函数原型	aclmdlConfigHandle *aclmdlCreateConfigHandle()
Python函数	config_handle = acl.mdl.create_config_handle()
函数功能	创建aclmdlConfigHandle类型的数据, 表示一个模型加载的配置对象。 如需销毁aclmdlConfigHandle类型的数据, 请参见acl.mdl.destroy_config_handle。
输入说明	无

返回值说明	config_handle : int, aclmdlConfigHandle类型的指针地址。 • 返回非0表示成功。 • 返回0表示失败。
约束说明	无

2.8.1.21 函数: destroy_config_handle

C函数原型	aclError aclmdlDestroyConfigHandle(aclmdlConfigHandle *handle)
Python函数	ret = acl.mdl.destroy_config_handle(config_handle)
函数功能	销毁通过 acl.mdl.create_config_handle 接口创建的aclmdlConfigHandle类型的数据。
输入说明	config_handle : int, 待销毁的aclmdlConfigHandle类型的指针地址。
返回值说明	ret : int, 错误码 • 返回0表示成功。 • 返回其它值表示失败。
约束说明	无

2.8.1.22 函数: set_config_opt

C函数原型	aclError aclmdlSetConfigOpt(aclmdlConfigHandle *handle, aclmdlConfigAttr attr, const void *attrValue, size_t valueSize)
Python函数	ret = acl.mdl.set_config_opt(config_handle, attr, attrValue)
函数功能	设置模型加载的配置对象中的各属性的取值, 包括模型执行的优先级、模型的文件路径或内存地址、内存大小等。
输入说明	config_handle : int, 指定模型加载的配置对象的指针地址。需提前调用 acl.mdl.create_config_handle 接口创建该对象的指针地址。 attr : int, 指定需设置的属性, 取值请参考 2.16.27 aclmdlConfigAttr 。 attrValue : 指定设置属性的值, 取值请参考 2.16.27 aclmdlConfigAttr 。
返回值说明	ret : int, 错误码。 • 返回0表示成功。 • 返回其它值表示失败。

约束说明	本接口需要与以下其它接口配合，实现模型加载功能： 1. 调用acl.mdl.create_config_handle接口创建模型加载的配置对象。 2. 多次调用acl.mdl.set_config_opt接口设置配置对象中每个属性的值。 3. 调用acl.mdl.load_with_config接口指定模型加载时需要的配置信息，并进行模型加载。 4. 模型加载成功后，调用acl.mdl.destroy_config_handle接口销毁。
相关接口	使用acl.mdl.set_config_opt接口、acl.mdl.load_with_config接口时，是通过配置对象中的属性来区分，在加载模型时是从文件加载，还是从内存加载，以及内存是由系统内部管理，还是由用户管理。 当前pyACL还提供了以下接口实现模型加载的功能，从使用的接口上区分从文件加载，还是从内存加载，以及内存是由系统内部管理，还是由用户管理。 • acl.mdl.load_from_file接口 • acl.mdl.load_from_mem接口 • acl.mdl.load_from_file_with_mem接口 • acl.mdl.load_from_mem_with_mem接口

2.8.1.23 函数：load_with_config

C函数原型	<code>aclError aclmdlLoadWithConfig(const aclmdlConfigHandle *handle, uint32_t *modelId)</code>
Python函数	<code>model_id, ret = acl.mdl.load_with_config(config_handle)</code>
函数功能	指定模型加载时需要的配置信息，并进行模型加载。
输入说明	config_handle : int，指定模型加载的配置对象。需提前调用 acl.mdl.create_config_handle 接口创建该对象，与 acl.mdl.set_config_opt 中的“config_handle”保持一致。
返回值说明	model_id : int，系统完成模型加载后生成的模型ID。 ret : int，错误码。 • 返回0表示成功。 • 返回其它值表示失败。

约束说明	在加载前，请先根据文件的大小评估内存空间是否足够，内存空间不足，会导致应用程序异常。 本接口需要与以下其它接口配合，实现模型加载功能： 1. 调用acl.mdl.create_config_handle接口创建模型加载的配置对象。 2. 多次调用acl.mdl.set_config_opt接口设置配置对象中每个属性的值。 3. 调用acl.mdl.load_with_config接口指定模型加载时需要的配置信息，并进行模型加载。 4. 模型加载成功后，调用acl.mdl.destroy_config_handle接口销毁。
相关接口	使用acl.mdl.set_config_opt接口、acl.mdl.load_with_config接口时，是通过配置对象中的属性来区分，在加载模型时是从文件加载，还是从内存加载，以及内存是由系统内部管理，还是由用户管理。 当前pyACL还提供了以下接口实现模型加载的功能，从使用的接口上区分从文件加载，还是从内存加载，以及内存是由系统内部管理，还是由用户管理。 • acl.mdl.load_from_file接口 • acl.mdl.load_from_mem接口 • acl.mdl.load_from_file_with_mem接口 • acl.mdl.load_from_mem_with_mem接口

2.8.1.24 函数：get_tensor_real_name

C函数原型	<code>const char *aclmdlGetTensorRealName(const aclmdlDesc *modelDesc, const char *name)</code>
Python函数	<code>real_name = acl.mdl.get_tensor_real_name(model_desc, name)</code>
函数功能	根据指定名称获取Tensor的真实名称。 <code>acl.mdl.get_tensor_real_name</code>接口需要与以下接口配合使用。 • acl.mdl.get_input_dims • acl.mdl.get_input_dims_v2 • acl.mdl.get_output_dims • acl.mdl.get_cur_output_dims
输入说明	model_desc: int, <code>aclmdlDesc</code>类型的指针地址。 需提前调用acl.mdl.create_desc接口创建<acode>aclmdlDesc类型的数据。</acode> name: str, 指定名称，用于根据该名称获取Tensor的真实名称。

返回值说明	real_name : str, 返回Tensor真实名称。 若“model_desc”或“name”为空, 则返回0。
约束说明	无

2.8.1.25 函数: create_exec_config_handle

C函数原型	aclmdlExecConfigHandle *aclmdlCreateExecConfigHandle()
Python函数	handle = acl.mdl.create_exec_config_handle()
函数功能	创建aclmdlExecConfigHandle类型的数据, 表示一个模型执行的配置对象的指针地址。 如需销毁aclmdlExecConfigHandle类型的数据, 请参见 2.8.1.26 函数: destroy_exec_config_handle 。
输入说明	无
返回值说明	handle : int, 返回aclmdlExecConfigHandle类型的指针地址。 • 返回非0表示成功。 • 返回0表示失败。

2.8.1.26 函数: destroy_exec_config_handle

C函数原型	aclError aclmdlDestroyExecConfigHandle(const aclmdlExecConfigHandle *handle)
Python函数	ret = acl.mdl.destroy_exec_config_handle(handle)
函数功能	销毁通过 acl.mdl.create_exec_config_handle 接口创建的aclmdlExecConfigHandle类型数据。
输入说明	handle : int, 待销毁的aclmdlExecConfigHandle数据类型的指针地址。
返回值说明	ret : int, 错误码。 • 返回0表示成功。 • 返回其它值表示失败。

2.8.1.27 函数: set_exec_config_opt

C函数原型	aclError aclmdlSetExecConfigOpt(aclmdlExecConfigHandle *handle, aclmdlExecConfigAttr attr, const void *attrValue, size_t valueSize)
Python函数	ret = acl.mdl.set_exec_config_opt(handle, attr, attrValue)

函数功能	设置模型执行的配置对象中的各属性的取值。
输入说明	handle : int, 模型执行的配置对象的指针地址。需提前调用 acl.mdl.create_exec_config_handle 接口创建该对象的指针地址。 attr : int, 指定需设置的属性。具体请参见 2.16.28 aclmdlExecConfigAttr 。 attrValue : int, “attr” 对应的属性取值。
返回值说明	ret : int, 错误码。 - 返回0表示成功。 - 返回其它值表示失败。
约束说明	本接口需要配合其它接口一起使用, 实现模型执行, 接口调用顺序如下: - 调用acl.mdl.create_exec_config_handle接口创建模型执行的配置对象。 - 多次调用acl.mdl.set_exec_config_opt接口设置配置对象中每个属性的值。 - 调用acl.mdl.execute_v2接口指定模型执行时需要的配置信息, 并进行模型执行。 - 模型加载成功后, 调用acl.mdl.destroy_exec_config_handle接口销毁。

2.8.2 aclmdlAIPP

2.8.2.1 函数: create_aipp

C函数原型	<code>aclmdlAIPP *aclmdlCreateAIPP(uint64_t batchSize)</code>
Python函数	<code>aipp_ptr = acl.mdl.create_aipp(batch_size)</code>
函数功能	动态AIPP场景下, 根据模型支持的batch size创建aclmdlAIPP类型的数据, 用于存放动态AIPP的参数。 如需销毁aclmdlAIPP类型的数据, 请参见 2.8.2.2 函数: destroy_aipp 。
输入说明	batch_size : int, 模型的Batch数。
返回值说明	aipp_ptr : int, aclmdlAIPP类型数据对象地址。
约束说明	无

2.8.2.2 函数：destroy_aipp

C函数原型	<code>aclError aclmdlDestroyAIPP(const aclmdlAIPP *aippParmsSet)</code>
Python函数	<code>ret = acl.mdl.destroy_aipp(aipp_parms_set)</code>
函数功能	销毁通过 acl.mdl.create_aipp 接口创建的aclmdlAIPP类型的数据。
输入说明	aipp_parms_set : int, 待销毁的aclmdlAIPP数据类型对象的指针地址。
返回值说明	ret : int, 错误码 - 返回0表示成功。 - 返回其它值表示失败。
约束说明	无

2.8.2.3 设置动态 AIPP 参数

2.8.2.3.1 函数：set_aipp_input_format

C函数原型	<code>aclError aclmdlSetAIPPInputFormat(aclmdlAIPP *aippParmsSet, aclAippInputFormat inputFormat)</code>
Python函数	<code>ret = acl.mdl.set_aipp_input_format(aipp_parms_set, input_format)</code>
函数功能	动态AIPP场景下，必须设置原始输入图像的格式。
输入说明	aipp_parms_set : int, 动态AIPP参数对象的指针地址，需提前调用 acl.mdl.create_aipp 接口创建aclmdlAIPP类型的数据。 input_format : int, 表示原始输入图像的格式，具体请参见 2.16.15 aclAippInputFormat 。
返回值说明	ret : int, 错误码 - 返回0表示成功。 - 返回其它值表示失败。
约束说明	无

2.8.2.3.2 函数：set_aipp_csc_params

C函数原型	aclError aclmdlSetAIPPCscParams(aclmdlAIPP *aippParmsSet, int8_t cscSwitch, int16_t cscMatrixR0C0, int16_t cscMatrixR0C1, int16_t cscMatrixR0C2, int16_t cscMatrixR1C0, int16_t cscMatrixR1C1, int16_t cscMatrixR1C2, int16_t cscMatrixR2C0, int16_t cscMatrixR2C1, int16_t cscMatrixR2C2, uint8_t cscOutputBiasR0, uint8_t cscOutputBiasR1, uint8_t cscOutputBiasR2, uint8_t cscInputBiasR0, uint8_t cscInputBiasR1, uint8_t cscInputBiasR2)
Python函数	ret = acl.mdl.set_aipp_csc_params(aipp_parms_set, csc_switch, csc_matrix_R0C0, csc_matrix_R0C1, csc_matrix_R0C2, csc_matrix_R1C0, csc_matrix_R1C1, csc_matrix_R1C2, csc_matrix_R2C0, csc_matrix_R2C1, csc_matrix_R2C2, csc_output_bias_R0, csc_output_bias_R1, csc_output_bias_R2, csc_input_bias_R0, csc_input_bias_R1, csc_input_bias_R2)
函数功能	动态AIPP场景下，设置CSC色域转换相关的参数，若色域转换开关关闭，则调用该接口设置的参数无效。 YUV转BGR: <pre> B cscMatrixR0C0 cscMatrixR0C1 cscMatrixR0C2 Y - cscInputBiasR0 G = cscMatrixR1C0 cscMatrixR1C1 cscMatrixR1C2 U - cscInputBiasR1 >> 8 R cscMatrixR2C0 cscMatrixR2C1 cscMatrixR2C2 V - cscInputBiasR2 </pre> BGR转YUV: <pre> Y cscMatrixR0C0 cscMatrixR0C1 cscMatrixR0C2 B cscOutputBiasR0 U = cscMatrixR1C0 cscMatrixR1C1 cscMatrixR1C2 G >> 8 + cscOutputBiasR1 V cscMatrixR2C0 cscMatrixR2C1 cscMatrixR2C2 R cscOutputBiasR2 </pre> 色域转换参数值与转换前图片的格式、转换后图片的格式强相关，您可以参见《ATC工具使用指南》下的色域转换配置说明，按照转换前图片格式、转换后图片格式来配置色域转换参数。如果手册中列举的图片格式不满足要求，您需自行根据实际需求配置色域转换参数。

输入说明	aipp_parms_set: int, 动态AIPP参数对象的指针地址, 需提前调用acl.mdl.create_aipp接口创建aclmdlAIPP类型的数据。 csc_switch: int, 色域转换开关, 取值范围: 0: 关闭色域转换开关, 默认为0。 1: 打开色域转换开关。 csc_matrix_R0C0: int, 色域转换矩阵参数, 取值范围: [-32677, 32676]。 csc_matrix_R0C1: int, 色域转换矩阵参数, 取值范围: [-32677, 32676]。 csc_matrix_R0C2: int, 色域转换矩阵参数, 取值范围: [-32677, 32676]。 csc_matrix_R1C0: int, 色域转换矩阵参数, 取值范围: [-32677, 32676]。 csc_matrix_R1C1: int, 色域转换矩阵参数, 取值范围: [-32677, 32676]。 csc_matrix_R1C2: int, 色域转换矩阵参数, 取值范围: [-32677, 32676]。 csc_matrix_R2C0: int, 色域转换矩阵参数, 取值范围: [-32677, 32676]。 csc_matrix_R2C1: int, 色域转换矩阵参数, 取值范围: [-32677, 32676]。 csc_matrix_R2C2: int, 色域转换矩阵参数, 取值范围: [-32677, 32676]。 csc_output_bias_R0: int, RGB转YUV时的输出偏移, 默认值为0, 取值范围: [0, 255]。 csc_output_bias_R1: int, RGB转YUV时的输出偏移, 默认值为0, 取值范围: [0, 255]。 csc_output_bias_R2: int, RGB转YUV时的输出偏移, 默认值为0, 取值范围: [0, 255]。 csc_input_bias_R0: int, YUV转RGB时的输入偏移, 默认值为0, 取值范围: [0, 255]。 csc_input_bias_R1: int, YUV转RGB时的输入偏移, 默认值为0, 取值范围: [0, 255]。 csc_input_bias_R2: int, YUV转RGB时的输入偏移, 默认值为0, 取值范围: [0, 255]。
返回值说明	ret: int, 错误码 - 返回0表示成功。 - 返回其它值表示失败。
约束说明	如果通过 acl.mdl.set_aipp_input_format 接口设置的原始图像格式为YUV400, 则不支持通过本接口设置色域转换参数。

2.8.2.3.3 函数：set_aipp_rbuvs_swap_switch

C函数原型	<code>aclError aclmdlSetAIPPRbuvsSwapSwitch(aclmdlAIPP *aippParamsSet, int8_t rbuvsSwapSwitch)</code>
Python函数	<code>ret = acl.mdl.set_aipp_rbuvs_swap_switch(aipp_params_set, rbuvs_swap_switch)</code>
函数功能	动态AIPP场景下，CSC色域转换前，设置是否交换R通道与B通道或者是否交换U通道与V通道。
输入说明	aipp_params_set : int，动态AIPP参数对象的指针地址，需提前调用 acl.mdl.create_aipp 接口创建aclmdlAIPP类型的数据。 rbuvs_swap_switch : int，表示是否交换R通道与B通道（或者是否交换U通道与V通道）的开关。 取值范围： 0：不交换，默认为0。 1：交换。
返回值说明	ret : int，错误码 - 返回0表示成功。 - 返回其它值表示失败。
约束说明	无

2.8.2.3.4 函数：set_aipp_ax_swap_switch

C函数原型	<code>aclError aclmdlSetAIPPAxSwapSwitch(aclmdlAIPP *aippParamsSet, int8_t axSwapSwitch)</code>
Python函数	<code>ret = acl.mdl.set_aipp_ax_swap_switch(aipp_params_set, ax_swap_switch)</code>
函数功能	动态AIPP场景下，设置RGBA->ARGB或者YUVA->AYUV的交换开关。
输入说明	aipp_params_set : int，动态AIPP参数对象的指针地址。需提前调用 acl.mdl.create_aipp 接口创建aclmdlAIPP类型的数据。 ax_swap_switch : int，表示RGBA->ARGB或者YUVA->AYUV的交换开关。 取值范围： 0：不交换，默认为0。 1：交换。
返回值说明	ret : int，错误码 - 返回0表示成功。 - 返回其它值表示失败。

约束说明	无

2.8.2.3.5 函数：set_aipp_src_image_size

C函数原型	<code>aclError aclmdlSetAIPPSrcImageSize(aclmdlAIPP *aippParmsSet, int32_t srcImageSizeW, int32_t srcImageSizeH)</code>
Python函数	<code>ret = acl.mdl.set_aipp_src_image_size(aipp_params_set, src_image_size_w, src_image_size_h)</code>
函数功能	动态AIPP场景下，必须设置原始图片的宽和高。
输入说明	aipp_params_set : int，动态AIPP参数对象，需提前调用 acl.mdl.create_aipp 接口创建 aclmdlAIPP 类型的数据。 src_image_size_w : int，原始图片的宽，对于YUV420SP_U8类型的图像，要求取值是偶数，取值范围：[2, 4096]。 src_image_size_h : int，原始图片的高，对于YUV420SP_U8类型的图像，要求取值是偶数，取值范围：[1, 4096]。
返回值说明	ret : int，错误码 • 返回0表示成功。 • 返回 其它值 表示失败。
约束说明	无

2.8.2.3.6 函数：set_aipp_crop_params

C函数原型	<code>aclError aclmdlSetAIPPCropParams(aclmdlAIPP *aippParmsSet, int8_t cropSwitch, int32_t cropStartPosW, int32_t cropStartPosH, int32_t cropSizeW, int32_t cropSizeH, uint64_t batchIndex)</code>
Python函数	<code>ret = acl.mdl.set_aipp_crop_params(aipp_params_set, crop_switch, crop_start_pos_w, crop_start_pos_h, crop_size_w, crop_size_h, batch_index)</code>
函数功能	动态AIPP场景下，设置抠图相关的参数。

输入说明	aipp_parms_set: int, 动态AIPP参数对象的指针地址, 需提前调用acl.mdl.create_aipp接口创建aclmdlAIPP类型的数据。 crop_switch: int, 是否对图片执行抠图操作。取值范围: 0: 不执行抠图操作, 设置为0时, 则设置cropStartPosW、cropStartPosH、cropSizeW、cropSizeH参数无效。 1: 执行抠图操作。 crop_start_pos_w: int, 抠图时, 坐标点起始位置在图中横向的坐标。对于YUV420SP_U8格式的图像, 参数取值要求是偶数, 取值范围: [0, 4095]。 crop_start_pos_h: int, 抠图时, 坐标点起始位置在图中纵向的坐标。对于YUV420SP_U8格式的图像, 参数取值要求是偶数, 取值范围: [0, 4095]。 crop_size_w: int, 抠图区域的宽度。取值范围: [1, 4096]。 crop_size_h: int, 抠图区域的高度。取值范围: [1, 4096]。 batch_index: int, 指定对第几个Batch上的图片执行抠图操作, 默认为0, 取值范围: [0, batch_size), batch_size是在调用acl.mdl.create_aipp接口创建aclmdlAIPP类型的数据时设置。
返回值说明	ret: int, 错误码 - 返回0表示成功。 - 返回其它值表示失败。
约束说明	若开启抠图功能, 则通过acl.mdl.set_aipp_src_image_size接口设置的参数与通过acl.mdl.set_aipp_crop_params接口设置的参数之间必须满足以下公式: $srcImageSizeW \geq cropSizeW + cropStartPosW$ $srcImageSizeH \geq cropSizeH + cropStartPosH$

2.8.2.3.7 函数: set_aipp_padding_params

C函数原型	aclError aclmdlSetAIPPPaddingParams(aclmdlAIPP *aippParmsSet, int8_t paddingSwitch,int32_t paddingSizeTop, int32_t paddingSizeBottom,int32_t paddingSizeLeft, int32_t paddingSizeRight,uint64_t batchSize)
Python函数	ret = acl.mdl.set_aipp_padding_params(aipp_parms_set, padding_switch, padding_size_top, padding_size_bottom, padding_size_left, padding_size_right, batch_index)
函数功能	动态AIPP场景下, 设置补边相关的参数。

输入说明	aipp_parms_set: int, 动态AIPP参数对象的指针地址, 需提前调用acl.mdl.create_aipp接口创建aclmdlAIPP类型的数据。 padding_switch: int, 是否对图片执行补边操作。 取值范围: 0: 不执行补边操作, 默认为0 (此时设置“padding_size_top”、“padding_size_bottom”、“padding_size_left”、“padding_size_right”参数无效)。 1: 执行补边操作。 padding_size_top: int, 在图片上方填充的值, 取值范围: [0, 32]。 padding_size_bottom: int, 在图片下方填充的值, 取值范围: [0, 32]。 padding_size_left: int, 在图片左方填充的值, 取值范围: [0, 32]。 padding_size_right: int, 在图片右方填充的值, 取值范围: [0, 32]。 batch_index: int, 指定对第几个Batch上的图片执行补边操作, 默认为0, 取值范围: [0, batch_size), batch_size是在调用acl.mdl.create_aipp接口创建aclmdlAIPP类型的数据时设置。
返回值说明	ret: int, 错误码 - 返回0表示成功。 - 返回其它值表示失败。
约束说明	补边之后, 图片的宽必须小于等于1080。

2.8.2.3.8 函数: set_aipp_dtc_pixel_mean

C函数原型	<pre>aclError aclmdlSetAIPPDtcPixelMean(aclmdlAIPP *aippParmsSet, int16_t dtcPixelMeanChn0, int16_t dtcPixelMeanChn1, int16_t dtcPixelMeanChn2, int16_t dtcPixelMeanChn3, uint64_t batchSize)</pre>
Python函数	<pre>ret = acl.mdl.set_aipp_dtc_pixel_mean(aipp_parms_set, dtc_pixel_mean_chn0, dtc_pixel_mean_chn1, dtc_pixel_mean_chn2, dtc_pixel_mean_chn3, batch_index)</pre>
函数功能	动态AIPP场景下, 设置通道的均值。

输入说明	aipp_parms_set: int, 动态AIPP参数对象的指针地址, 需提前调用acl.mdl.create_aipp接口创建aclmdlAIPP类型的数据。 dtc_pixel_mean_chn0: int, 通道0的均值, 取值范围: [0, 255]。 dtc_pixel_mean_chn1: int, 通道1的均值, 取值范围: [0, 255]。 dtc_pixel_mean_chn2: int, 通道2的均值, 取值范围: [0, 255]。 dtc_pixel_mean_chn3: int, 通道3的均值, 如果只有3个通道, 则该参数传默认值0。取值范围: [0, 255]。 batch_index: int, 指定对第几个Batch上的图片设置通道均值, 默认为0, 取值范围: [0, <i>batch_size</i>), <i>batch_size</i>是在调用acl.mdl.create_aipp接口创建aclmdlAIPP类型的数据时设置。
返回值说明	ret: int, 错误码 • 返回0表示成功。 • 返回其它值表示失败。
约束说明	无

2.8.2.3.9 函数: set_aipp_dtc_pixel_min

C函数原型	aclError aclmdlSetAIPPDtcPixelMin(aclmdlAIPP *aippParmsSet, float dtcPixelMinChn0, float dtcPixelMinChn1, float dtcPixelMinChn2, float dtcPixelMinChn3, uint64_t batchSize)
Python函数	ret = acl.mdl.set_aipp_dtc_pixel_min(aipp_parms_set, dtc_pixel_mean_chn0, dtc_pixel_mean_chn1, dtc_pixel_mean_chn2, dtc_pixel_mean_chn3, batch_index)
函数功能	动态AIPP场景下, 设置通道的最小值。
输入说明	aipp_parms_set: int, 动态AIPP参数对象的指针地址, 需提前调用acl.mdl.create_aipp接口创建aclmdlAIPP类型的数据。 dtc_pixel_mean_chn0: int, 通道0的均值, 取值范围: [0, 255]。 dtc_pixel_mean_chn1: int, 通道1的均值, 取值范围: [0, 255]。 dtc_pixel_mean_chn2: int, 通道2的均值, 取值范围: [0, 255]。 dtc_pixel_mean_chn3: int, 通道3的均值, 如果只有3个通道, 则该参数传默认值0。取值范围: [0, 255]。 batch_index: int, 指定对第几个Batch上的图片设置通道均值, 默认为0。取值范围: [0, <i>batch_size</i>), <i>batch_size</i>可在调用acl.mdl.create_aipp接口创建aclmdlAIPP类型的数据时设置。

返回值说明	ret : int, 错误码 - 返回0表示成功。 - 返回其它值表示失败。
约束说明	无

2.8.2.3.10 函数: set_aipp_pixel_var_reci

C函数原型	aclError aclmdlSetAIPPPixelVarReci(aclmdlAIPP *aippParmsSet, float dtcPixelVarReciChn0, float dtcPixelVarReciChn1, float dtcPixelVarReciChn2, float dtcPixelVarReciChn3, uint64_t batchSize)
Python函数	ret = acl.mdl.set_aipp_pixel_var_reci(aipp_parms_set, dtc_pixel_var_reci_chn0, dtc_pixel_var_reci_chn1, dtc_pixel_var_reci_chn2, dtc_pixel_var_reci_chn3, batch_index)
函数功能	动态AIPP场景下，设置通道的方差。
输入说明	aipp_parms_set : int, 动态AIPP参数对象的指针地址，需提前调用 acl.mdl.create_aipp 接口创建aclmdlAIPP类型的数据。 dtc_pixel_var_reci_chn0 : int, 通道0的方差，默认值为1.0，取值范围：[-65504, 65504]。 dtc_pixel_var_reci_chn1 : int, 通道1的方差，默认值为1.0，取值范围：[-65504, 65504]。 dtc_pixel_var_reci_chn2 : int, 通道2的方差，默认值为1.0，取值范围：[-65504, 65504]。 dtc_pixel_var_reci_chn3 : int, 通道3的方差，如果只有3个通道，则该参数传默认值1.0，取值范围：[-65504, 65504]。 batch_index : int, 指定对第几个Batch上的图片设置通道的方差，取值范围：[0, <i>batch_size</i>)， <i>batch_size</i> 是在调用 acl.mdl.create_aipp 接口创建aclmdlAIPP类型的数据时设置。
返回值说明	ret : int, 错误码 - 返回0表示成功。 - 返回其它值表示失败。
约束说明	无

2.8.2.3.11 函数: set_input_aipp

C函数原型	aclError aclmdlSetInputAIPP(uint32_t modelId, aclmdlDataset *dataset, size_t index, const aclmdlAIPP *aippParmsSet)

Python函数	ret = acl.mdl.set_input_aipp(model_id, dataset, index, aipp_parms_set)
函数功能	动态AIPP场景下，根据指定的动态AIPP输入的输入index，设置模型推理时的AIPP参数值。 动态AIPP支持的几种操作的计算方式及其计算顺序如下： 1. 抠图 2. 色域转换 3. 缩放（当前版本不支持缩放） 4. 减均值/归一化 5. 补边
输入说明	model_id: int，模型的ID，可在以下接口成功加载模型后获取到模型ID。 • acl.mdl.load_from_file • acl.mdl.load_from_mem • acl.mdl.load_from_file_with_mem • acl.mdl.load_from_mem_with_mem dataset: int，表示模型的输入数据的指针地址，使用aclmdlDataset类型的数据描述模型推理时的输入数据，输入的内存地址、内存大小用aclDataBuffer类型的数据来描述。 index: int，标识动态AIPP输入的输入index。 • 多个动态AIPP输入的场景下，用户可调用acl.mdl.get_aipp_type接口获取指定模型输入所关联的动态AIPP输入的输入index。 • 为保证向前兼容，如果明确只有一个动态AIPP输入，可调用acl.mdl.get_input_index_by_name接口获取，输入名称固定为“ascend_dynamic_aipp_data”。 aipp_parms_set: int，动态AIPP参数对象的指针地址，需提前调用acl.mdl.create_aipp接口创建aclmdlAIPP类型的数据。
返回值说明	ret: int，错误码 • 返回0表示成功。 • 返回其它值表示失败。
约束说明	• 经过动态AIPP处理后的图像的宽、高必须与原始模型中输入Shape中的宽、高保持一致。 • 多Batch场景下，根据每个Batch的配置计算出动态AIPP后输出图片的宽、高，经过动态AIPP后每个Batch的输出图片宽、高必须是一致的。计算输出图片宽、高的计算公式如表2-4所示。 • 抠图或者缩放或者补边之后，对图片宽、高的校验规则如下，其中，aippOutputW、aippOutputH分别表示AIPP输出图片的宽、高，其它参数是acl.mdl.set_aipp_src_image_size、acl.mdl.set_aipp_crop_params、acl.mdl.set_aipp_padding_params接口的入参。
参考资源	接口调用流程与示例，参见动态AIPP（单个动态AIPP输入）。

表 2-4 输出图片宽、高计算公式

抠图	缩放	补边 (padding)	动态AIPP输出图片的宽、高
否	否	否	aippOutputW = srcImageSizeW aippOutputH = srcImageSizeH
是	否	否	aippOutputW = cropSizeW aippOutputH = cropSizeH
是	是	否	aippOutputW = scfOutputSizeW aippOutputH = scfOutputSizeH
是	否	是	aippOutputW = cropSizeW + paddingSizeLeft + paddingSizeRight aippOutputH = cropSizeH + paddingSizeTop + paddingSizeBottom
否	否	是	aippOutputW = srcImageSizeW + paddingSizeLeft + paddingSizeRight aippOutputH = srcImageSizeH + paddingSizeTop + paddingSizeBottom
否	是	是	aippOutputW = scfOutputSizeW + paddingSizeLeft + paddingSizeRight aippOutputH = scfOutputSizeH + paddingSizeTop + paddingSizeBottom
否	是	否	aippOutputW = scfOutputSizeW aippOutputH = scfOutputSizeH

抠图	缩放	补边 (padding)	动态AIPP输出图片的宽、高
是	是	是	aippOutputW = scfOutputSizeW + paddingSizeLeft + paddingSizeRight aippOutputH = scfOutputSizeH + paddingSizeTop + paddingSizeBottom

2.8.3 aclmdlDataset

2.8.3.1 总体说明

您可以通过以下接口获取[aclmdlDataset](#)类型的数据并对[aclmdlDataset](#)的数据执行一些操作。该数据类型下的操作接口都是同步接口。

[aclmdlDataset](#)主要用于描述模型推理时的输入数据、输出数据，模型可能存在多个输入、多个输出，每个输入/输出的内存地址、内存大小用[aclDataBuffer](#)类型的数据来描述。

2.8.3.2 函数：create_dataset

C函数原型	aclmdlDataset *aclmdlCreateDataset()
Python函数	mdl_dataset = acl.mdl.create_dataset()
函数功能	创建 aclmdlDataset 类型的数据。 该数据类型用于描述模型推理时的输入数据、输出数据，模型可能存在多个输入、多个输出，每个输入/输出的内存地址、内存大小用 aclDataBuffer 类型的数据来描述。 如需销毁 aclmdlDataset 类型的数据，请参见 2.8.3.3 函数：destroy_dataset 。
输入说明	无
返回值说明	mdl_dataset : int, aclmdlDataset 类型的指针地址。
约束说明	无

2.8.3.3 函数：destroy_dataset

C函数原型	aclError aclmdlDestroyDataset(const aclmdlDataset *dataset)
Python函数	ret = acl.mdl.destroy_dataset(dataset)

函数功能	销毁通过 acl.mdl.create_dataset 接口创建的aclmdlDataset类型的数据。
输入说明	dataset : int, 待销毁的aclmdlDataset的指针地址。
返回值说明	ret : int, 错误码 - 返回0表示成功。 - 返回其它值表示失败。
约束说明	销毁类接口：调用该接口后，不能继续使用已释放或销毁的资源，建议调用该接口后，将相关资源设置为无效值（例如，置为0或None）。

2.8.3.4 函数：add_dataset_buffer

C函数原型	aclError aclmdlAddDatasetBuffer(aclmdlDataset *dataset, aclDataBuffer *dataBuffer)
Python函数	dataset, ret = acl.mdl.add_dataset_buffer(dataset, data_buffer)
函数功能	向aclmdlDataset中增加aclDataBuffer。
输入说明	dataset : int, 待增加aclDataBuffer的aclmdlDataset指针地址。 需提前调用 acl.mdl.create_dataset 接口创建aclmdlDataset类型的数据。 data_buffer : int, 待增加的aclDataBuffer指针地址。 需提前调用 acl.create_data_buffer 接口创建aclDataBuffer类型的数据。
返回值说明	dataset : int, 待增加aclDataBuffer的aclmdlDataset指针地址。 ret : int, 错误码。 - 返回0表示成功。 - 返回其它值表示失败。
约束说明	无

2.8.3.5 函数：get_dataset_num_buffers

C函数原型	size_t aclmdlGetDatasetNumBuffers(const aclmdlDataset *dataset)
Python函数	index = acl.mdl.get_dataset_num_buffers(dataset)
函数功能	获取aclmdlDataset中aclDataBuffer的个数。

输入说明	dataset : int, aclmdlDataset数据对象的指针地址。 需提前调用 acl.mdl.create_dataset 接口创建aclmdlDataset类型的数据。
返回值说明	index : int, aclDataBuffer数据类型的个数。
约束说明	无

2.8.3.6 函数: get_dataset_buffer

C函数原型	aclDataBuffer* aclmdlGetDatasetBuffer(const aclmdlDataset *dataset, size_t index)
Python函数	data_buffer = acl.mdl.get_dataset_buffer(dataset, index)
函数功能	获取aclmdlDataset中的第n个aclDataBuffer。
输入说明	dataset : int, aclmdlDataset数据对象的指针地址。 需提前调用 acl.mdl.create_dataset 接口创建aclmdlDataset类型的数据。 index : int, 表明获取的是第几个aclDataBuffer。
返回值说明	data_buffer : int, aclDataBuffer数据对象指针地址。 - 获取成功, 返回aclDataBuffer的地址。 - 获取失败返回空地址。
约束说明	无

2.8.3.7 函数: set_dataset_tensor_desc

C函数原型	aclError aclmdlSetDatasetTensorDesc(aclmdlDataset *dataset, aclTensorDesc *tensorDesc, size_t index)
Python函数	dataset, ret = acl.mdl.set_dataset_tensor_desc(dataset, tensorDesc, index)
函数功能	如果模型输入或输出的Shape是动态的, 在模型执行之前调用本接口设置模型输入或输出的Tensor描述信息。 Atlas 200/300/500 推理产品 , 不支持该接口。 Atlas 200I/500 A2推理产品 , 不支持该接口。

输入说明	dataset: int, 待增加aclTensorDesc的aclmdlDataset指针地址, 表示模型执行的输入或输出数据结构。 需提前调用acl.mdl.create_dataset接口创建aclmdlDataset类型的数据, 再调用acl.mdl.add_dataset_buffer接口向aclmdlDataset中增加aclDataBuffer。 tensorDesc: int, 待增加的aclTensorDesc指针地址, 表示模型执行时对应的输入或输出的Tensor描述。 需提前调用acl.create_tensor_desc接口创建aclTensorDesc类型的数据, 当前仅支持设置模型输入、输出Tensor描述信息中的维度信息 (对应acl.create_tensor_desc接口中的代表维度个数的numDims参数、代表维度大小的dims参数), 设置数据类型、Format无效。 此处设置的维度个数、维度大小必须在模型构建时设置的输入Shape范围内, 模型构建的详细说明请参见模型构建。 index: int, 表示第几个输入或输出的序号。 模型存在多个输入、输出时, 为避免序号出错, 可以先调用acl.mdl.get_input_name_by_index、acl.mdl.get_output_name_by_index接口获取输入、输出的名称, 然后根据输入、输出名称所对应的“index”来设置。
返回值说明	dataset: int, 待增加aclTensorDesc的aclmdlDataset指针地址, 表示模型执行的输入或输出数据结构。 ret: int, 错误码。 • 返回0表示成功。 • 返回其它值表示失败。
约束说明	对同一个模型, acl.mdl.set_dynamic_batch_size 接口、 acl.mdl.set_dynamic_hw_size 接口、 acl.mdl.set_input_dynamic_dims 接口、 acl.mdl.set_dataset_tensor_desc 接口, 只能调用其中一个接口。

2.8.3.8 函数: get_dataset_tensor_desc

C函数原型	<code>aclTensorDesc *aclmdlGetDatasetTensorDesc(const aclmdlDataset *dataset, size_t index)</code>
Python函数	<code>tensorDesc =acl.mdl.get_dataset_tensor_desc(dataset, index)</code>

函数功能	如果模型输入或输出的Shape是动态的，模型在模型执行之后可以调用本接口从aclmdlDataset类型的数据（该类型用于描述模型推理时的输入数据、输出数据）中获取指定的输入或者输出的Tensor描述信息。 典型场景举例：如果模型输入Shape是动态的，在模型执行之前调acl.mdl.set_dataset_tensor_desc设置该输入的Tensor描述信息，在模型执行之后，调用acl.mdl.get_dataset_tensor_desc接口获取模型动态输出的Tensor描述信息。 Atlas 200/300/500 推理产品，不支持该接口。 Atlas 200I/500 A2推理产品，不支持该接口。
输入说明	dataset: int，表示模型执行的输入或输出数据结构的指针地址。 index: int，表示第几个输入或输出的序号。 模型存在多个输入、输出时，为避免序号出错，可以先调用acl.mdl.get_input_name_by_index、acl.mdl.get_output_name_by_index接口获取输入、输出的名称，根据输入、输出名称所对应的index来设置。
返回值说明	tensorDesc: int，返回指定输入或输出Tensor描述信息的指针地址。

2.8.4 aclmdlDesc

2.8.4.1 函数：create_desc

C函数原型	aclmdlDesc* aclmdlCreateDesc()
Python函数	desc = acl.mdl.create_desc()
函数功能	创建aclmdlDesc类型的数据。 如需销毁aclmdlDesc类型的数据，请参见 acl.mdl.destroy_desc 。
输入说明	无
返回值说明	desc: int，aclmdlDesc类型对象的地址。
约束说明	无

2.8.4.2 函数：destroy_desc

C函数原型	aclError aclmdlDestroyDesc(aclmdlDesc *modelDesc)
Python函数	ret = acl.mdl.destroy_desc(model_desc)
函数功能	销毁通过 acl.mdl.create_desc 接口创建的aclmdlDesc类型的数据。
输入说明	model_desc: int，待销毁的aclmdlDesc的指针地址。

返回值说明	ret : int, 错误码 - 返回0表示成功。 - 返回其它值表示失败。
约束说明	无

2.8.4.3 函数：get_desc

C函数原型	aclError aclmdlGetDesc(aclmdlDesc *modelDesc, uint32_t modelId)
Python函数	ret = acl.mdl.get_desc(model_desc, model_id)
函数功能	根据模型ID获取该模型的aclmdlDesc类型数据。
输入说明	model_desc : int, aclmdlDesc类型数据的指针地址。 需提前调用 acl.mdl.create_desc 接口创建aclmdlDesc类型的数据。 model_id : int, 模型ID。
返回值说明	ret : int, 错误码 - 返回0表示成功。 - 返回其它值表示失败。
约束说明	无

2.8.4.4 函数：get_num_inputs

C函数原型	size_t aclmdlGetNumInputs(aclmdlDesc *modelDesc)
Python函数	size = acl.mdl.get_num_inputs(model_desc)
函数功能	根据aclmdlDesc类型的数据，获取模型的输入个数。
输入说明	model_desc : int, aclmdlDesc类型数据的指针地址。 需提前调用 acl.mdl.create_desc 接口创建aclmdlDesc类型的数据。
返回值说明	size : int, 模型的输入个数。
约束说明	无

2.8.4.5 函数：get_num_outputs

C函数原型	size_t aclmdlGetNumOutputs(aclmdlDesc *modelDesc)
Python函数	size = acl.mdl.get_num_outputs(model_desc)

函数功能	根据aclmdlDesc类型的数据，获取模型的输出个数。
输入说明	model_desc : int, aclmdlDesc类型数据的指针地址。 需提前调用 acl.mdl.create_desc 接口创建aclmdlDesc类型的数据。
返回值说明	size : int, 模型的输出个数。
约束说明	无

2.8.4.6 函数: get_input_size_by_index

C函数原型	size_t aclmdlGetInputSizeByIndex(aclmdlDesc *modelDesc, size_t index)
Python函数	size = acl.mdl.get_input_size_by_index(model_desc, index)
函数功能	根据aclmdlDesc类型的数据，获取指定输入的大小，单位为Byte。
输入说明	model_desc : int, aclmdlDesc类型数据的指针地址。 需提前调用 acl.mdl.create_desc 接口创建aclmdlDesc类型的数据。 index : int, 指定获取第几个输入的大小，“index”取值从0开始。
返回值说明	size : int, 返回指定输入的大小，单位是Byte。 - 针对动态Batch、动态分辨率（宽高）的场景，返回最大档位对应的输入的大小。 - 静态场景下，返回指定输入的大小。单位是Byte。
约束说明	如果模型输入的Shape是动态的、且维度的取值为-1（表示此维度可以使用大于或等于1的任意取值），则通过本接口获取的大小为0，用户需根据实际数据占用的内存大小来申请内存。

2.8.4.7 函数: get_output_size_by_index

C函数原型	size_t aclmdlGetOutputSizeByIndex(aclmdlDesc *modelDesc, size_t index)
Python函数	size = acl.mdl.get_output_size_by_index(model_desc, index)
函数功能	根据aclmdlDesc类型的数据，获取指定输出的大小，单位为Byte。
输入说明	model_desc : int, aclmdlDesc类型数据的指针地址。需提前调用 acl.mdl.create_desc 接口创建aclmdlDesc类型的数据。 index : int, 指定获取第几个输出的大小，index值从0开始。

返回值说明	size: int, 返回指定输出的大小, 单位是Byte。 - 针对动态Batch、动态分辨率(宽高)的场景, 返回最大档位对应的输出的大小。 - 静态场景下, 返回指定输出的大小。单位是Byte。
约束说明	如果通过本接口获取的大小为0, 有可能是由于输出Shape的范围不确定, 当前支持以下两种处理方式: 系统内部自行申请对应index的输出内存, 节省内存, 但内存数据使用结束后, 需由用户释放内存, 同时, 系统内部申请内存时涉及内存拷贝, 可能涉及性能损耗。 调用acl.create_data_buffer接口创建存放对应index输出数据的aclDataBuffer类型时, 可在data参数处传入0, 表示创建一个空的aclDataBuffer类型, 然后在模型执行过程中, 系统内部自行计算并申请该index输出的内存。 用户预估输出内存大小, 并申请内存, 由用户自行管理内存, 但内存大小可能不足或超出, 不足时系统会校验报错, 超出时会浪费内存。 用户需先根据实际情况预估一块较大的输出内存, 在模型执行过程中, 系统会校验用户指定的输出内存大小是否符合要求, 如果不符合要求, 系统会返回报错, 并在报错信息中提示具体需要多大的输出内存。您可以通过以下两种方式查看报错: - 获取应用类日志, 查看ERROR级别的报错, 如何获取日志请参见《日志参考》。 - 在应用程序中调用2.3.4 函数: get_recent_err_msg接口获取报错。

2.8.4.8 函数: get_input_dims

C函数原型	aclError aclmdlGetInputDims(const aclmdlDesc *modelDesc, size_t index, aclmdlIODims *dims)
Python函数	dims, ret = acl.mdl.get_input_dims(model_desc, index)
函数功能	根据模型描述获取模型的输入Tensor的维度信息。 如果模型中含有静态AIPP配置信息, 您可以根据实际需要选择acl.mdl.get_input_dims接口或acl.mdl.get.input_dims_v2接口查询维度信息, 两者的区别在于: - 通过acl.mdl.get_input_dims接口获取的维度信息, 各维度的值与输入图像的各维度的值保持一致, 详细规则请参见表2-5。 - 通过acl.mdl.get_input_dims_v2接口获取的维度信息, <i>H维度的值 = 输入图像的H维度值 * 系数</i>, 不同图片格式下C维度的值不同, 详细规则请参见acl.mdl.get_input_dims_v2接口处的表2-6, 且各维度值相乘的结果值与通过acl.mdl.get_input_size_by_index接口获取的值保持一致。

输入说明	model_desc: int, aclmdlDesc类型数据的指针地址。 需提前调用acl.mdl.create_desc接口创建aclmdlDesc类型的数据。 index: int, 指定获取第几个输入的dims, index值从0开始。
返回值说明	dims: dict, 针对多batch或多种宽高的场景, 按照该字典的数组返回其支持的最大档的组合, 具体请参见2.16.18 aclmdlIODims。 <pre>dims = { "name": xxx, #tensor name "dimCount":xxx, #shape中的维度个数 "dims": [xx, xx, xx] # 维度信息 }</pre> 针对动态Batch、动态分辨率（宽高）的场景, 输入Tensor的“dims”中Batch数或宽高为-1, 表示其动态可变。 例如, 输入Tensor的“format”为NCHW, 在动态Batch场景下, 动态可变的输入Tensor的“dims”为[-1, 3, 224, 224], 在动态分辨率场景下, 动态可变的输入Tensor的“dims”为[1, 3, -1, -1]。 (举例中的斜体部分以实际情况为准。) - 若Tensor的“name”长度大于127, 则在输出dims["name"]时, pyACL会将Tensor的“name”转换为“acl_modelId_\${id}_input_\${index}_\${随机字符串}”格式并在转换后的“name”与原“name”之间建立映射关系, 用户可调用acl.mdl.get_tensor_real_name接口, 传入转换后的“name”, 获取原“name” (若向接口传入原“name”, 则获取的还是原“name”)。 说明 如果转换后的Tensor的“name”与模型中已有的Tensor的“name”冲突, 则会在转换后的“name”尾部增加“_\${随机字符串}”, 否则不会增加随机字符串。 - 若Tensor的“name”长度小于或等于127, 则在输出的dims["name"]时, 按Tensor的“name”输出。 ret: int, 错误码。 - 返回0表示成功。 - 返回其它值表示失败。
约束说明	无

表 2-5 静态 AIPP 场景下的维度定义规则

图像格式	Format参考格式	维度定义规则
YUV420SP_U8	NHWC	n,h,w,c
XRGB8888_U8	NHWC	n,h,w,c
RGB888_U8	NHWC	n,h,w,c
YUV400_U8	NHWC	n,h,w,c
ARGB8888_U8	NHWC	n,h,w,c

图像格式	Format参考格式	维度定义规则
YUYV_U8	NHWC	n,h,w,c
YUV422SP_U8	NHWC	n,h,w,c
AYUV444_U8	NHWC	n,h,w,c

2.8.4.9 函数: `get_input_dims_v2`

C函数原型	<code>aclError aclmdlGetInputDimsV2(const aclmdlDesc *modelDesc, size_t index, aclmdlIODims *dims)</code>
Python函数	<code>dims, ret = acl.mdl.get_input_dims_v2(model_desc, index)</code>
函数功能	根据模型描述获取模型的输入Tensor的维度信息。 如果模型中含有静态AIPP配置信息，您可以根据实际需要选择acl.mdl.get_input_dims接口或acl.mdl.get_input_dims_v2接口查询维度信息，两者的区别在于： - 通过acl.mdl.get_input_dims接口获取的维度信息，各维度的值与输入图像的各维度的值保持一致，详细规则请参见表2-5。 - 通过acl.mdl.get_input_dims_v2接口获取的维度信息，<i>H维度的值 = 输入图像的H维度值 * 系数</i>，不同图片格式下C维度的值不同，详细规则请参见acl.mdl.get_input_dims_v2接口处的表2-6，且各维度值相乘的结果值与通过acl.mdl.get_input_size_by_index接口获取的值保持一致。
输入说明	model_desc: int，aclmdlDesc类型数据的指针地址。 需提前调用acl.mdl.create_desc接口创建aclmdlDesc类型的数据。 index: int，指定获取第几个输入的大小，index值从0开始。

返回值说明	dims: dict, 针对多batch或多种宽高的场景, 按照该字典的数组返回其支持的最大档的组合, 具体请参见2.16.18 aclmdlIODims。 <pre>dims = { "name": xxx, #tensor name "dimCount":xxx, #shape中的维度个数 "dims": [xx, xx, xx] # 维度信息 }</pre> 针对动态Batch、动态分辨率（宽高）的场景, 输入Tensor的“dims”中Batch数或宽高为-1, 表示其动态可变。 例如, 输入Tensor的“format”为NCHW, 在动态Batch场景下, 动态可变的输入Tensor的“dims”为[-1, 3, 224, 224], 在动态分辨率场景下, 动态可变的输入Tensor的“dims”为[1, 3, -1, -1]。举例中的斜体部分以实际情况为准。 - 若Tensor的“name”长度大于127, 则在输出dims["name"]时, pyACL会将Tensor的“name”转换为“acl_modelId_\${id}_input_\${index}_\${随机字符串}”格式并在转换后的“name”与原“name”之间建立映射关系, 用户可调用acl.mdl.get_tensor_real_name接口, 传入转换后的“name”, 获取原“name”（若向接口传入原“name”, 则获取的还是原“name”）。 说明 如果转换后的Tensor的“name”与模型中已有的Tensor的“name”冲突, 则会在转换后的“name”尾部增加“_\${随机字符串}”, 否则不会增加随机字符串。 - 若Tensor的“name”长度小于或等于127, 则在输出的dims["name"]时, 按Tensor的“name”输出。 针对静态AIPP场景, 本接口针对不同格式的图像, 对应NHWC的Format格式, 当前接口中明确各个维度的定义规则, 如表2-6所示。 ret: int, 错误码。 - 返回0表示成功。 - 返回其它值表示失败。
约束说明	无

表 2-6 动态 AIPP 场景下的维度定义规则

图像格式	Format参考格式	维度定义规则
YUV420SP_U8	NHWC	"n,h*1.5,w,1"
XRGB8888_U8	NHWC	"n,h,w,4"
RGB888_U8	NHWC	"n,h,w,3"
YUV400_U8	NHWC	"n,h,w,1"
ARGB8888_U8	NHWC	"n,h,w,4"
YUYV_U8	NHWC	"n,h,w,2"

图像格式	Format参考格式	维度定义规则
YUV422SP_U8	NHWC	"n,h*2,w,1"
AYUV444_U8	NHWC	"n,h,w,4"

2.8.4.10 函数：get_output_dims

C函数原型	aclError aclmdlGetOutputDims(const aclmdlDesc *modelDesc, size_t index, aclmdlIODims *dims)
Python函数	dims, ret = acl.mdl.get_output_dims(model_desc, index)
函数功能	根据模型描述获取指定的模型输出Tensor的维度信息。 - 固定Shape场景下，通过该接口获取指定的模型输出Tensor的维度信息。 - 动态Shape（动态Batch或动态分辨率）场景下，通过该接口获取最大档的维度信息。
输入说明	model_desc: int, aclmdlDesc类型数据的指针地址。 需提前调用acl.mdl.create_desc接口创建aclmdlDesc类型的数据。 index: int, 指定获取第几个输入的大小，index值从0开始。
返回值说明	dims: dict, 针对多batch或多种宽高的场景，按照该字典的数组返回其支持的最大档的组合，具体请参见2.16.18 aclmdlIODims。 <pre>dims = { "name": xxx, #tensor name "dimCount":xxx, #shape中的维度个数 "dims": [xx, xx, xx] # 维度信息 }</pre> - 若Tensor的“name”长度大于127，则在输出dims["name"]时，pyACL会将Tensor的“name”转换为“acl_modelId_{id}_input_{index}_{随机字符串}”格式并在转换后的“name”与原“name”之间建立映射关系，用户可调用acl.mdl.get_tensor_real_name接口，传入转换后的“name”，获取原“name”（若向接口传入原“name”，则获取的还是原“name”）。 说明 如果转换后的Tensor的“name”与模型中已有的Tensor的“name”冲突，则会在转换后的“name”尾部增加“_{随机字符串}”，否则不会增加随机字符串。 - 若Tensor的“name”长度小于或等于127，则在输出的dims["name"]时，按Tensor的“name”输出。 ret: int, 错误码。 - 返回0表示成功。 - 返回其它值表示失败。
约束说明	无

2.8.4.11 函数：get_input_name_by_index

C函数原型	<code>const char *aclmdlGetInputNameByIndex(const aclmdlDesc *modelDesc, size_t index)</code>
Python函数	<code>input_name = acl.mdl.get_input_name_by_index(model_desc, index)</code>
函数功能	获取模型中指定输入的输入名称。
输入说明	model_desc : int, aclmdlDesc类型数据的指针地址。 需提前调用 acl.mdl.create_desc 接口创建aclmdlDesc类型的数据。 index : int, 指定获取第几个输入的大小, “index” 值从0开始。
返回值说明	input_name : str, 返回指定输入的输入名称。
约束说明	无

2.8.4.12 函数：get_output_name_by_index

C函数原型	<code>const char *aclmdlGetOutputNameByIndex(const aclmdlDesc *modelDesc, size_t index)</code>
Python函数	<code>output_name = acl.mdl.get_output_name_by_index(model_desc, index)</code>
函数功能	根据模型描述信息获取模型中指定输出的输出算子名称、算子输出边的下标、top名称或输出名称。
输入说明	model_desc : int, aclmdlDesc类型数据的指针地址。 需提前调用 acl.mdl.create_desc 接口创建aclmdlDesc类型的数据。 index : int, 指定获取第几个输入的大小, “index” 值从0开始。

返回值说明	output_name: str, 返回指定输入的输入名称。 返回指定输出的输出算子名称、算子输出边的下标、top名称或输出名称。不同原始网络、不同构建模型的方式, 调用本接口获取的返回值格式不同。 • Caffe网络。 返回值格式如下, 各项之间以冒号分割, 如果模型中包含top名称就返回, 不包含就不返回: 输出算子名称: 算子输出边下标: top名称 • TensorFlow网络。 - 使用ATC工具构建om模型的场景下, 返回值格式如下, 各项之间以冒号分割: 输出算子名称: 算子输出边下标 - 使用Ascend Graph接口构建om模型的场景下, 返回值格式如下, 各项之间以下划线分割: output_网络输出下标_输出算子名称_算子输出边下标 • ONNX网络。 返回值格式如下, 各项之间以冒号分割: 输出算子名称: 算子输出边下标: 输出名称 如果在构建om模型时, 以下情况会在返回值中包含输出名称。 - 不指定网络模型中输出算子的名称或输出的名称(系统会自动从原始模型中获取输出的名称)。 - 仅指定网络模型中输出的名称。 使用Ascend Graph接口构建om模型时, 如果指定了网络模型中输出算子的名称, 则返回值格式如下, 各项之间以下划线分割: output_网络输出下标_输出算子名称_算子输出边下标
约束说明	无

2.8.4.13 函数: get_input_format

C函数原型	aclFormat aclmdlGetInputFormat (const aclmdlDesc * modelDesc , size_t index)
Python函数	format_info = acl.mdl.get_input_format (model_desc , index)
函数功能	根据模型描述信息获取模型中指定输入的Format。
输入说明	model_desc: int, aclmdlDesc类型数据的指针地址。 需提前调用acl.mdl.create_desc接口创建aclmdlDesc类型的数据。 index: int, 指定获取第几个输入的大小, “index”值从0开始。
返回值说明	format_info : int, 返回指定输入的Format, 具体请参见 2.16.4 aclFormat 。
约束说明	无

2.8.4.14 函数：get_output_format

C函数原型	aclFormat aclmdlGetOutputFormat(const aclmdlDesc *modelDesc, size_t index)
Python函数	format_info = acl.mdl.get_output_format(model_desc, index)
函数功能	根据模型描述信息获取模型中指定输出的Format。
输入说明	model_desc : int, aclmdlDesc类型数据的指针地址。 需提前调用 acl.mdl.create_desc 接口创建aclmdlDesc类型的数据。 index : int, 指定获取第几个输入的大小, “index” 值从0开始。
返回值说明	format_info : int, 返回指定输入的Format, 具体请参见 2.16.4 aclFormat 。
约束说明	无

2.8.4.15 函数：get_input_data_type

C函数原型	aclDataType aclmdlGetInputDataType(const aclmdlDesc *modelDesc, size_t index)
Python函数	data_type = acl.mdl.get_input_data_type(model_desc, index)
函数功能	根据模型描述信息获取模型中指定输入的数据类型。
输入说明	model_desc : int, aclmdlDesc类型数据的指针地址。 index : int, 指定获取第几个输入的大小, “index” 值从0开始。
返回值说明	data_type : int, 返回指定输入的数据类型, 具体请参见 2.16.2 aclDataType 。
约束说明	无

2.8.4.16 函数：get_output_data_type

C函数原型	aclDataType aclmdlGetOutputDataType(const aclmdlDesc *modelDesc, size_t index)
Python函数	data_type = acl.mdl.get_output_data_type(model_desc, index)
函数功能	根据模型描述信息获取模型中指定输出的数据类型。
输入说明	model_desc : int, aclmdlDesc类型数据的指针地址。 需提前调用 acl.mdl.create_desc 接口创建aclmdlDesc类型的数据。 index : int, 指定获取第几个输入的大小, “index” 值从0开始。

返回值说明	data_type : int, 返回指定输入的数据类型, 具体请参见 2.16.2 aclDataType 。
约束说明	无

2.8.4.17 函数: get_input_index_by_name

C函数原型	aclError aclmdlGetInputIndexByName(const aclmdlDesc *modelDesc, const char *name, size_t *index)
Python函数	index, ret = acl.mdl.get_input_index_by_name(model_desc, name)
函数功能	根据模型中的输入名称获取对应输入的索引编号。
输入说明	model_desc : int, aclmdlDesc类型数据的指针地址。 需提前调用 acl.mdl.create_desc 接口创建aclmdlDesc类型的数据。 name : str, 指定输入名称。
返回值说明	index : int, 获取对应输入的索引编号。 ret : int, 错误码。 • 返回0表示成功。 • 返回其它值表示失败。
约束说明	无

2.8.4.18 函数: get_output_index_by_name

C函数原型	aclError aclmdlGetOutputIndexByName(const aclmdlDesc *modelDesc, const char *name, size_t *index)
Python函数	index, ret = acl.mdl.get_output_index_by_name(model_desc, name)
函数功能	根据模型中的输出名称获取对应输出的索引编号。
输入说明	model_desc : int, aclmdlDesc类型数据的指针地址。 需提前调用 acl.mdl.create_desc 接口创建aclmdlDesc类型的数据。 name : str, 指定输入名称。
返回值说明	index : int, 获取对应输入的索引编号。 ret : int, 错误码。 • 返回0表示成功。 • 返回其它值表示失败。
约束说明	无

2.8.4.19 函数：get_dynamic_batch

C函数原型	<code>aclError aclmdlGetDynamicBatch(const aclmdlDesc *modelDesc, aclmdlBatch *batch)</code>
Python函数	<code>batch, ret = acl.mdl.get_dynamic_batch(model_desc)</code>
函数功能	根据模型描述信息获取模型支持的动态Batch信息。
输入说明	model_desc : int, aclmdlDesc类型数据的指针地址。 需提前调用 acl.mdl.create_desc 接口创建aclmdlDesc类型的数据。
返回值说明	batch : dict, 获取的模型支持的动态Batch信息。 - 模型中支持的最大batch数为128。 <pre>batch = { "batchCount": int, #模型中支持的batch分档数 "batch": [xx, xx,] # 模型中支持的具体分档 }</pre> “batchCount”等于0时, 表示不支持设置档位信息, 以模型中的档位为准。 ret : int, 错误码。 - 返回0表示成功。 - 返回其它值表示失败。
约束说明	无

2.8.4.20 函数：get_dynamic_hw

C函数原型	<code>aclError aclmdlGetDynamicHW(const aclmdlDesc *modelDesc, size_t index, aclmdlHW *hw)</code>
Python函数	<code>hw_info, ret = acl.mdl.get_dynamic_hw(model_desc, index)</code>
函数功能	根据模型描述信息获取模型支持的动态宽高信息。
输入说明	model_desc : int, aclmdlDesc类型数据的指针地址。 需提前调用 acl.mdl.create_desc 接口创建aclmdlDesc类型的数据。 index : 预留参数, 当前未使用, 固定设置为-1。

返回值说明	hw_info: dict, 获取的模型支持的宽高信息。 - 最大的HW数量为128。 <pre>hw_info = { "hwCount": int, #模型中支持的宽高分档数 "hw":[xx, xx,] # 模型中支持的具体分档, 每组分档中, 数组下标为0代表的是高, 下标为1代表的是宽 }</pre> “hwCount”等于0时, 表示不支持设置档位信息, 以模型中的档位为准。 ret: int, 错误码。 - 返回0表示成功。 - 返回其它值表示失败。
约束说明	无

2.8.4.21 函数: get_cur_output_dims

C函数原型	aclError aclmdlGetCurOutputDims(const aclmdlDesc *modelDesc, size_t index, aclIODims *dims)
Python函数	dims, ret = acl.mdl.get_cur_output_dims(model_desc, index)
函数功能	根据模型描述信息获取指定的模型输出Tensor的实际维度信息。
输入说明	model_desc: int, aclmdlDesc类型数据的指针地址。 需提前调用acl.mdl.create_desc接口创建aclmdlDesc类型的数据。 index: int, 指定获取第几个输出的Dims, index值从0开始。
返回值说明	dims: dict, 输出实际维度信息。 - 若Tensor的“name”长度大于127, 则在输出dims["name"]时, pyACL会将Tensor的“name”转换为“acl_modelId_{id}_input_{index}_{随机字符串}”格式并在转换后的“name”与原“name”之间建立映射关系, 用户可调用acl.mdl.get_tensor_real_name接口, 传入转换后的“name”, 获取原“name”(若向接口传入原“name”, 则获取的还是原“name”)。 说明 如果转换后的Tensor的“name”与模型中已有的Tensor的“name”冲突, 则会在转换后的“name”尾部增加“_{随机字符串}”, 否则不会增加随机字符串。 - 若Tensor的“name”长度小于或等于127, 则在输出的dims["name"]时, 按Tensor的“name”输出。 ret: int, 错误码。 - 返回0表示成功。 - 返回其它值表示失败。

约束说明	当前仅支持通过本接口获取以下场景中的模型输出Tensor的维度信息： - 通过模型转换设置多档Batch size或分辨率或维度值，实现动态Batch或动态分辨率或动态维度（ND格式）时。 - 如果用户已调用acl.mdl.set_dynamic_batch_size设置Batch或调用acl.mdl.set_dynamic_hw_size接口设置输入图片的宽高或调用acl.mdl.set_input_dynamic_dims接口设置某动态维度的值，则可通过该接口获取指定模型输出Tensor的实际维度信息。 - 如果用户未调用acl.mdl.set_dynamic_batch_size接口、acl.mdl.set_dynamic_hw_size接口或acl.mdl.set_input_dynamic_dims接口，则通过该接口可获取最大档的维度信息。 - 固定Shape场景下，通过该接口获取指定的模型输出Tensor的维度信息。

2.8.4.22 函数：get_first_aipp_info

C函数原型	<code>aclError aclmdlGetFirstAippInfo(uint32_t modelId, size_t index, aclAippInfo *aippInfo)</code>
Python函数	<code>aipp_info, ret = acl.mdl.get_first_aipp_info(model_id, index)</code>
函数功能	获取模型AIPP（包括静态AIPP和动态AIPP）的配置信息。 AIPP支持的几种操作的计算方式及其计算顺序如下： - 抠图 - 色域转换 - 缩放（当前版本不支持缩放） - 减均值/归一化 - 补边
输入说明	model_id: int，模型ID。 可在以下接口成功加载模型后获取到模型ID。 acl.mdl.load_from_file acl.mdl.load_from_mem acl.mdl.load_from_file_with_mem acl.mdl.load_from_mem_with_mem index: int，指定模型的第几个输入。

返回值说明	aipp_info: dict, 获取指定输入上静态AIPP的配置信息, 具体请参见2.16.16 aclAippInfo。 详细说明及参数解释, 请参考《ATC工具使用指南》。 ret: int, 错误码。 • 返回0表示成功。 • 返回其它值表示失败。
约束说明	如果使用本接口获取模型中动态AIPP的信息, 只能获取“aipp_info”中如下参数的值: srcFormat、srcDatatype、srcDimNum、shapeCount、outDims, 其它参数值无效。

2.8.4.23 函数: get_input_dynamic_gear_count

C函数原型	aclError aclmdlGetInputDynamicGearCount(const aclmdlDesc *modelDesc, size_t index, size_t *gearCount)
Python函数	gear_count, ret = acl.mdl.get_input_dynamic_gear_count(model_desc, index)
函数功能	根据模型描述信息获取模型的输入所支持的动态维度档位数。
输入说明	model_desc: int, aclmdlDesc类型模型描述信息数据的指针地址。 需提前调用acl.mdl.create_desc接口创建aclmdlDesc类型的数据。 index: int, 预留参数, 当前未使用, 固定设置为-1。
返回值说明	gear_count: int, 动态维度档位数。 例如, 模型的输入Tensor是4维, 在模型转换时通过“--dynamic_dims”参数设置的分档为“1, 3, 224, 224; 2, 3, 224, 224; 1, 3, 256, 256”, 那么通过该接口获取的动态维度档位数为3。 ret: int, 错误码。 • 返回0表示成功。 • 返回其它值表示失败。
约束说明	如果模型构建时没有设置动态维度的分档, 那么通过该接口获取的动态维度档位数为0。模型构建的详细说明请参见模型构建。

2.8.4.24 函数: get_input_dynamic_dims

C函数原型	aclError aclmdlGetInputDynamicDims(const aclmdlDesc *modelDesc, size_t index, aclmdlIODims *dims, size_t gearCount)
Python函数	dims_out, ret = acl.mdl.get_input_dynamic_dims(model_desc, index, gear_count)

函数功能	根据模型描述信息获取模型的输入所支持的动态维度信息。
输入说明	model_desc: int, aclmdlDesc类型数据的指针地址。 需提前调用acl.mdl.create_desc接口创建aclmdlDesc类型的数据。 index: int, 预留参数, 当前未使用, 固定设置为-1。 gear_count: int, 模型支持的动态维度档位数, 需要先通过acl.mdl.get_input_dynamic_gear_count接口获取。
返回值说明	dims_out: list, 获取输入的动态维度信息。 “dims_out”参数是一个列表, 模型有几个输入, 则该列表就有几个元素。列表中的每个元素都是一个aclmdlIODims的字典, “aclmdlIODims”字典中的dims是一个列表, 该列表中的每个元素对应每一档中的具体值。 例如: <pre>dims_out, ret = acl.mdl.get_input_dynamic_dims(model_desc, -1, gear_count)</pre> ret: int, 错误码。 • 返回0表示成功。 • 返回其它值表示失败。
约束说明	只有在模型转换时通过“--dynamic_dims”参数设置了分档信息后, 才可以调用该接口获取动态维度信息。 例如, 模型有三个输入, 分别为data(1, 1, 40, -1), label(1, -1), mask(-1, -1), 其中-1表示动态可变。在模型转换时, “--dynamic_dims”参数的配置示例为: --dynamic_dims="20,20,1,1; 40,40,2,2; 80,60,4,4", 则通过本接口获取的动态维度信息为 (aclmdlIODims字典内的“name”暂不使用): • 第0档: - aclmdlIODims内“dimCount”: 8, 表示所有输入Tensor的维度数量之和。 - aclmdlIODims内的“dims”: “1,1,40,20,1,20,1,1”, 表示data(1,1,40,20)+label(1,20)+mask(1,1) • 第1档: - aclmdlIODims内“dimCount”: 8, 表示所有输入Tensor的维度数量之和。 - aclmdlIODims内的“dims”: “1,1,40,40,1,40,2,2”, 表示data(1,1,40,40)+label(1,40)+mask(2,2) • 第2档: - aclmdlIODims内“dimCount”: 8, 表示所有输入Tensor的维度数量之和。 - aclmdlIODims内的“dims”: “1,1,40,80,1,60,4,4”, 表示data(1,1,40,80)+label(1,60)+mask(4,4)

2.8.4.25 函数：get_op_attr

C函数原型	<code>const char *aclmdlGetOpAttr(aclmdlDesc *modelDesc, const char *opName, const char *attr)</code>
Python函数	<code>attrValue = acl.mdl.get_op_attr(modelDesc, opName, attr)</code>
函数功能	获取整网中某个模型中某个算子的属性的值。
输入说明	modelDesc: int, aclmdlDesc类型的指针地址。 需提前调用 acl.mdl.create_desc接口创建aclmdlDesc类型的数据，再调用 acl.mdl.get_desc接口根据模型ID获取到对应的aclmdlDesc类型的数据。 opName: str, 算子名称。 attr: str, 算子属性。 当前仅支持 “_datadump_original_op_names” 属性，用于记录某个算子是由哪些算子融合得到的。通过本接口获取到的 “_datadump_original_op_names” 属性值格式为 [opName1_len]opName1...[opNameN_len]opNameN, “opNameN_len” 表示算子名称字符串的长度。 “_datadump_original_op_names” 属性值示例如下，表示某个融合算子由scale2c_branch2c、bn2c_branch2c、res2c_branch2c、res2c、res2c_relu这五个算子融合而成的，算子名称字符串的长度分别为16、13、14、5、10： [16]scale2c_branch2c[13]bn2c_branch2c[14]res2c_branch2c[5]res2c[10]res2c_relu
返回值说明	attrValue: str, 返回属性值的字符串，若opName或者attr属性不存在或者attr属性值为空，均返回空字符串。 若调用该接口失败，则返回None。
约束说明	无

2.9 媒体数据处理 V1

2.9.1 总体说明

多版本接口差异

- 本手册中媒体数据处理V1版本与媒体数据处理V2版本的接口功能范围相同，都是描述处理媒体数据的接口，用于实现抠图、图片缩放、格式转换等功能，但两套接口不能混用。
 - 建议使用V2版本中的接口，保证后续版本接口功能以及业务的连续演进。
 - V1版本中的接口是为了兼容旧版本，保证使用该部分接口的用户能继续使用，后续版本不再演进。

功能说明

本章节介绍媒体数据处理V1版本（DVPP，Digital Vision Pre-Processing）的功能，如[表2-7](#)所示。

表 2-7 功能说明

功能	说明
VPC (Vision Preprocessing Core)	负责图像处理功能，支持对图片做抠图、缩放、格式转换等操作，详细描述请参见 2.9.6.1 功能说明 。
JPEGD (JPEG Decoder)	负责完成图像解码功能，将.jpg、.jpeg、.JPG、.JPEG图片解码成YUV格式图片，详细描述请参见 2.9.7.1 功能及约束说明 。
JPEGE (JPEG Encoder)	负责完成图像编码功能，将YUV格式图片编码成.jpg图片，详细描述请参见 2.9.8.1 功能及约束说明 。
VDEC (Video Decoder)	负责视频解码，详细描述请参见 2.9.9.1 功能及约束说明 。
VENC (Video Encoder)	负责视频编码，详细描述请参见 2.9.10.1 功能及约束说明 。
PNGD (PNG Decoder)	负责PNG格式图片的解码，详细描述请参见 2.9.11.1 功能及约束说明 。

功能支持度说明

各产品对媒体数据处理V1版本各功能的支持度如[表2-8](#)所示。

各标识的含义如下：

- √：支持。
- x：不支持。

表 2-8 功能支持度说明

产品	VPC	JPEG D	JPEG E	PNG D	VDEC	VENC
Atlas 200/300/500 推理产品	√	√	√	√	√	√
Atlas 200I/500 A2推理产品	√	√	√	√	√	√
Atlas 训练系列产品	√	√	√	√	√	x
Atlas A2训练系列产品/Atlas 800I A2推理产品	√	√	√	√	√	x
Atlas 推理系列产品	√	√	√	√	√	√

整体约束说明

使用本章中介绍的接口，有以下注意点：

- 关于异步接口：

对于本章介绍的异步接口，调用接口成功仅表示任务下发成功，不表示任务执行成功，对于有依赖的接口，为确保能按序执行任务，建议用户在多个接口中指定同一个Stream，因为同一个Stream中的任务按接口调用顺序执行。

在调用异步接口对图片进行解码、抠图、缩放等操作时，如果任务之间有依赖，一定要调用[acl.rt.synchronize_stream](#)接口确保在同一个Stream中的任务按序执行。

从性能角度考虑，建议一个Stream上下发多个异步媒体数据处理任务后，执行一次[acl.rt.synchronize_stream](#)接口。

调用异步接口后，不能马上释放资源，需调用同步等待接口（例如，[acl.rt.synchronize_stream](#)）确保Device侧任务执行完成后才能释放。

- 关于内存申请/释放：
 - a. 实现媒体数据处理的VPC、JPEGD、JPEGE等功能前，若需要申请Device上的内存存放输入或输出数据，需调用[acl.media.dvpp_malloc](#)申请内存、调用[acl.media.dvpp_free](#)接口释放内存。
 - b. 调用1申请出来的内存可以满足媒体数据处理的要求，也可以在其它任务中使用，例如，从性能角度，为了减少拷贝，媒体数据处理的输出作为模型推理的输入，实现内存复用。
 - c. 由于媒体数据处理访问的地址空间有限，为确保媒体数据处理时内存足够，除媒体数据处理功能外的其它功能（例如，模型加载），建议调用[内存管理](#)章节下的[acl.rt.malloc](#)接口、[acl.rt.malloc_host](#)接口或[acl.rt.malloc_cached](#)接口申请内存。

Atlas 推理系列产品上，媒体数据处理功能在每个进程内，可访问的地址空间最大为16GB。

Atlas A2训练系列产品/Atlas 800I A2推理产品上，媒体数据处理功能在每个进程内，可访问的地址空间最大为16GB。

- 关于通道的要求：

实现媒体数据处理的各功能前，必须调用接口创建对应功能的通道，创建通道的接口请参见[2.9.4 通道创建与释放](#)。通道的创建与销毁会涉及资源的申请与释放，反复创建与销毁通道会影响业务性能，因此建议根据实际场景管理通道，例如，如果有持续VPC图片处理，则创建VPC的通道后，等到所有VPC功能调用完成后，再销毁该VPC通道。

通道数量多，会影响Device的CPU占用率和内存占用，通道数量建议参考各功能章节下的性能指标的路数。

Atlas 推理系列产品上，各功能的通道数有一定的限制，VPC的通道数最多256，JPEGD与VDEC共用通道且通道数最多256，JPEGE与VENC共用通道且通道数最多256，PNGD的通道数最多128。创建通道的接口请参见[2.9.4 通道创建与释放](#)。

Atlas A2训练系列产品/Atlas 800I A2推理产品上，各功能的通道数有一定的限制，VPC的通道数最多256，JPEGD的通道数最多256，VDEC的通道数最多128，JPEGE的通道数最多256，PNGD的通道数最多128。创建通道的接口请参见[2.9.4 通道创建与释放](#)。

Atlas 200I/500 A2推理产品上，各功能的通道数有一定的限制，VPC的通道数最多128，JPEGD的通道数最多128，VDEC的通道数最多128，JPEGE的通道数最多128，VENC的通道数最多128，PNGD的通道数最多128。创建通道的接口请参见[2.9.4 通道创建与释放](#)。

📖 说明

对于Atlas 推理系列产品，昇腾虚拟化实例（Ascend Virtual Instance）场景下，如果通道总数不为整数，则向下取整：

- $\text{VPC通道总数} = (\text{被分配的VPC硬件单元数量} / \text{VPC硬件单元总数}) * 256$
- $\text{VDEC通道和JPEGD通道总数} = (\text{被分配的VDEC硬件单元} + \text{JPEGD硬件单元数量之和} / \text{VDEC硬件单元} + \text{JPEGD硬件单元总数之和}) * 256$
- $\text{VENC通道和JPEGE通道总数} = (\text{被分配的VENC硬件单元} + \text{JPEGE硬件单元数量之和} / \text{VENC硬件单元} + \text{JPEGE硬件单元总数之和}) * 256$
- $\text{PNGD通道总数} = \text{切分规格} * 128$

对于PNGD功能，若使用以下昇腾虚拟化实例（Ascend Virtual Instance）模板，则对通道数的限制不同：

- 使用vir04_4c_dvpp模板时，通道总数固定为128。
- 使用vir04_3c_ndvpp模板时，表示不使用DVPP功能，因此通道总数为0。

对于Atlas A2训练系列产品/Atlas 800I A2推理产品，昇腾虚拟化实例（Ascend Virtual Instance）场景下，如果通道总数不为整数，则向下取整：

- VPC通道总数最多256。
- $\text{VDEC通道} = (\text{被分配的VDEC硬件单元} / \text{VDEC硬件单元}) * 32$ ，如果通道总数不为整数，则向下取整。JPEGD的通道数不受算力影响，但JPEGD+VDEC的总通道数最大256。
- JPEG通道总数最多256。
- $\text{PNGD通道总数} = \text{切分规格} * 128$

对于PNGD功能，若使用以下昇腾虚拟化实例（Ascend Virtual Instance）模板，则对通道数的限制不同：

- 使用vir12_3c_32g_m、vir10_3c_16g_m、vir10_3c_32g_m模板时，通道总数固定为128。
- 使用vir12_3c_32g_nm、vir10_3c_16g_nm、vir10_3c_32g_nm模板时，表示不使用DVPP功能，因此通道总数为0。

对于Atlas 200I/500 A2推理产品，昇腾虚拟化实例（Ascend Virtual Instance）场景下，如果通道总数不为整数，则向下取整：

- VPC通道总数最多128。
- $\text{VDEC通道} = (\text{被分配的VDEC硬件单元} / \text{VDEC硬件单元}) * 128$ ，如果通道总数不为整数，则向下取整。JPEGD的通道数不随算力影响，但JPEGD+VDEC的总通道数最大128。
- $\text{VENC通道} = (\text{被分配的VENC硬件单元} / \text{VENC硬件单元}) * 128$ ，如果通道总数不为整数，则向下取整。JPEGE的通道数不随算力影响，但JPEGE+VENC的总通道数最大128。

昇腾虚拟化实例（Ascend Virtual Instance）的详细描述，以及各场景下的算力资源规格请参见《昇腾虚拟化实例（AVI）用户指南》。

2.9.2 基本概念

表 2-9 关键概念列表

概念	描述
宽Stride (widthStride)	指一行图像跨距，表示输入/输出图片对齐后的宽，RGB格式和YUV格式的宽Stride计算方式不一样。 各功能对宽Stride的要求不同，请参见对应功能的约束说明： • VPC功能，请参见图片格式、宽高对齐、内存约束。 • JPEGD功能，请参见2.9.7.1 功能及约束说明。 • JPEGGE功能，请参见2.9.8.1 功能及约束说明。 • PNGD功能，请参见2.9.11.1 功能及约束说明。 • VDEC功能，请参见2.9.9.1 功能及约束说明。 • VENC功能，请参见2.9.10.1 功能及约束说明。
高Stride (heightStride)	指图像在内存中的行数，表示输入/输出图片对齐后的高。 各功能对高Stride的要求不同，请参见对应功能的约束说明： • VPC功能，请参见图片格式、宽高对齐、内存约束。 • JPEGD功能，请参见2.9.7.1 功能及约束说明。 • JPEGGE功能，请参见2.9.8.1 功能及约束说明。 • PNGD功能，请参见2.9.11.1 功能及约束说明。 • VDEC功能，请参见2.9.9.1 功能及约束说明。 • VENC功能，请参见2.9.10.1 功能及约束说明。
上/下/左/右偏移	通过配置上偏移、下偏移、左偏移、右偏移可以实现两个功能：指定抠图区域或贴图区域的位置，控制抠图或贴图区域的宽、高，$右偏移 - 左偏移 + 1 = 宽$，$下偏移 - 上偏移 + 1 = 高$。 Atlas 200/300/500 推理产品请参见图2-1。 Atlas 推理系列产品请参见图2-1。 Atlas 训练系列产品请参见图2-1。 Atlas 200I/500 A2推理产品请参见图2-6。 Atlas A2训练系列产品/Atlas 800I A2推理产品请参见图2-6。
抠图区域	指用户指定的需抠出的图片区域。 抠图区域最小分辨率为10 * 6，最大分辨率为4096 * 4096。 抠图区域的约束，请参见抠图、贴图约束。
贴图区域	指在输出图片中用户指定的区域，贴图区域最小分辨率为10 * 6，最大分辨率为4096 * 4096。 贴图区域的约束，请参见抠图、贴图约束。

概念	描述
非8K	非8K表示10 * 6 ~ 4096 * 4096（包括4096）分辨率范围的输入图片。 非8K场景下，输入图片宽Stride、高Stride的取值范围如下：（宽Stride/系数）在32 ~ 4096（包括4096）范围内，高Stride在6 ~ 4096。 系数与输入图片的格式有关： • YUV400、YUV420SP、YUV422SP、YUV444SP、YUV440SP：系数为1。 • YUV422packed：系数为2。 • YUV444packed、RGB888：系数为3。 • ARGB8888：系数为4。
8K	8K表示宽或高在4096 ~ 8192（不包括4096）范围内的输入图片。 8K场景下，输入图片宽Stride、高Stride的取值范围如下： Atlas 200/300/500 推理产品上，宽Stride在4096 ~ 8192范围内或高Stride在4096 ~ 8192范围内（不包括4096）。 Atlas 训练系列产品上，宽Stride在4096 ~ 8192范围内或高Stride在4096 ~ 8192范围内（不包括4096）。 Atlas 推理系列产品上，宽Stride在4096~16384范围内（不包括4096）。

2.9.3 内存申请与释放

2.9.3.1 函数：dvpp_malloc

C函数原型	aclError aclDvppMalloc(void **devPtr, size_t size)
Python函数	dev_ptr, ret = acl.media.dvpp_malloc(size)
函数功能	该接口主要用于分配内存给Device侧媒体数据处理时使用，申请的大页内存满足数据处理的要求（例如，内存首地址128对齐）。
输入说明	size: int，申请内存的大小，单位Byte。
返回值说明	dev_ptr: int，Device上已分配内存的指针地址。 ret: int，错误码。 • 返回0表示成功。 • 返回其它值表示失败。

约束说明	- 调用该接口申请内存后，如果内存不再使用，需及时调用acl.media.dvpp_free接口释放内存。 - 频繁调用acl.media.dvpp_malloc接口申请内存、调用acl.media.dvpp_free接口释放内存，会损耗性能，建议用户提前做内存预先分配或二次管理，避免频繁申请/释放内存。 - 调用acl.media.dvpp_malloc接口申请内存时，会对用户输入的“size”按向上对齐成32整数倍后，再多加32字节。 - 媒体数据处理的输出作为模型推理的输入时，若用户使用本接口申请大块内存并自行划分、管理内存时，每段内存需同时满足以下要求： - 内存大小向上对齐成32整数倍+32字节（$m = \text{ALIGN\_UP}[\text{len}, 32] + 32$字节）。 - 内存起始地址需满足128字节对齐（$\text{ALIGN\_UP}[m, 128]$）。 说明 “len”表示某段内存的大小，$\text{ALIGN\_UP}[\text{len}, k]$表示向上按“k”字节对齐：$((\text{len} - 1) / k + 1) * k$。 - 调用该接口申请大页内存失败，仅表示系统内的大页内存不够。 Atlas 200/300/500 推理产品关于系统内预留大页的说明请参见2.7.6.1 总体说明。 Atlas A2训练系列产品/Atlas 800I A2推理产品关于系统内预留大页的说明请参见2.7.6.1 总体说明。
参考资源	接口调用流程及示例，参见媒体数据处理V1。

2.9.3.2 函数：dvpp_free

C函数原型	<code>aclError aclDvppFree(void *devPtr)</code>
Python函数	<code>ret = acl.media.dvpp_free(dev_ptr)</code>
函数功能	释放通过 acl.media.dvpp_malloc 接口申请的内存。
输入说明	dev_ptr : int，待释放内存的指针地址。
返回值说明	ret : int，错误码。 - 返回0表示成功。 - 返回其它值表示失败。
约束说明	无
参考资源	接口调用流程及示例，参见媒体数据处理V1。

2.9.4 通道创建与释放

2.9.4.1 VPC/JPEGD/JPEGE/PNGD 图片处理通道

2.9.4.1.1 函数：dvpp_create_channel

C函数原型	aclError acldvppCreateChannel(aclDvppChannelDesc *channelDesc)
Python函数	ret = acl.media.dvpp_create_channel(dvpp_channel_desc)
函数功能	创建图片数据处理的通道，同一个通道可以重复使用，销毁后不再可用。
输入说明	dvpp_channel_desc : int，指定通道描述信息的指针地址。调用 acl.media.dvpp_create_channel_desc 接口创建该输入数据。 同时，对于Atlas 推理系列产品，您还需要调用 dvpp_set_channel_desc_mode 接口指定通道描述信息中的通道模式，明确图片数据处理通道用于实现哪种功能，目前支持 VPC 、 JPEGD 、 JPEGE 、 PNGD 功能。若不指定通道模式，则系统默认4种模式的通道都创建，可能会占用通道资源（通道数的限制，请参见 2.9.1 总体说明 ）。
返回值说明	ret : int，错误码。 • 返回0表示成功。 • 返回其它值表示失败。
约束说明	• 通道为非线程安全，即不同线程要求创建不同的通道。 • 同一个通道不能同时在多个Stream中并发使用。 • acl.media.dvpp_create_channel接口内部调用1次acl.rt.create_stream接口创建了1个Stream，该Stream用于创建通道的任务，创建通道的任务结束后，该Stream资源自动释放。 用户在创建通道时，acl.rt.create_stream接口的约束需要遵循，例如Stream的数量等。
参考资源	接口调用流程与示例，参见媒体数据处理V1。

2.9.4.1.2 函数：dvpp_destroy_channel

C函数原型	aclError aclDvppDestroyChannel(aclDvppChannelDesc *channelDesc)
Python函数	ret = acl.media.dvpp_destroy_channel(dvpp_channel_desc)
函数功能	销毁图片数据处理的通道，只能销毁通过 acl.media.dvpp_create_channel 接口创建的通道。
输入说明	dvpp_channel_desc : int，指定通道描述信息的指针地址。 与调用 acl.media.dvpp_create_channel 接口创建通道时指定的dvpp_channel_desc保持一致。

返回值说明	ret : int, 错误码。 • 返回0表示成功。 • 返回其它值表示失败。
约束说明	如果在调用 acl.media.dvpp_destroy_channel 接口销毁通道前, 已调用 acl.media.dvpp_destroy_channel_desc 接口销毁了通道描述信息, 那么在调用 acl.media.dvpp_destroy_channel 接口销毁通道时会报错。
参考资源	接口调用流程及示例, 参见媒体数据处理V1。

2.9.4.2 VDEC 视频解码通道

2.9.4.2.1 函数: vdec_create_channel

C函数原型	aclError aclvdecCreateChannel(aclvdecChannelDesc *channelDesc)
Python函数	ret = acl.media.vdec_create_channel (vdec_channel_desc)
函数功能	创建视频数据处理的通道, 同一个通道可以重复使用, 销毁后不再可用。
输入说明	vdec_channel_desc : int, 通道描述信息的指针地址。 调用 acl.media.vdec_create_channel_desc 接口创建 vdec_channel_desc 类型的数据, 调用 acl.media.vdec_set_channel_desc 系列接口设置通道描述信息的属性。
返回值说明	ret : int, 错误码。 • 返回0表示成功。 • 返回其它值表示失败。

约束说明	• 通道为非线程安全，即不同线程要求创建不同的通道。 • 通道数的限制，请参见2.9.1 总体说明。 • acl.media.vdec_create_channel接口内部封装了acl.rt.create_stream接口和acl.rt.subscribe_report： - 调用2次acl.rt.create_stream接口创建了2个Stream，一个Stream用于视频码流解码任务，另一个用于回调函数的处理。这些Stream资源只有在调用acl.media.vdec_destroy_channel接口销毁通道后才能自动释放。 - 调用acl.rt.subscribe_report接口指定处理Stream上回调函数的线程，回调函数和线程是由用户调用acl.media.vdec_set_channel_desc系列接口时指定的。 用户在实现VDEC功能时，无需再单独调用acl.rt.create_stream接口、acl.rt.subscribe_report接口，但acl.rt.create_stream接口、acl.rt.subscribe_report接口的约束需要遵循，例如Stream的数量、处理Stream上回调函数的线程数量等。
参考资源	接口调用流程及示例，参见媒体数据处理V1。

2.9.4.2.2 函数：vdec_destroy_channel

C函数原型	<code>aclError aclvdecDestroyChannel(aclvdecChannelDesc *channelDesc)</code>
Python函数	<code>ret = acl.media.vdec_destroy_channel(vdec_channel_desc)</code>
函数功能	销毁视频数据处理的通道，只能销毁通过 acl.media.vdec_create_channel 接口创建的通道。
输入说明	vdec_channel_desc : int，通道描述信息的指针地址。 与调用 acl.media.vdec_create_channel 接口创建通道时指定的vdec_channel_desc保持一致。
返回值说明	ret : int，错误码。 • 返回0表示成功。 • 返回其它值表示失败。

约束说明	- 销毁通道接口会等待已发送帧解码完成且用户的回调函数处理完成后再销毁通道。 <code>acl.media.vdec_destroy_channel</code>接口内部封装了acl.rt.unsubscribe_report接口取消线程注册（Stream上的回调函数不再由指定线程处理）、acl.rt.destroy_stream接口销毁Stream。用户在实现VDEC功能时，无需再单独调用acl.rt.unsubscribe_report接口、acl.rt.destroy_stream接口。 - 如果在调用acl.media.vdec_destroy_channel接口销毁通道前，已调用acl.media.vdec_destroy_channel_desc接口销毁了通道描述信息，那么在调用acl.media.vdec_destroy_channel接口销毁通道时会报错。
参考资源	接口调用流程及示例，参见媒体数据处理V1。

2.9.4.3 VENC 视频编码通道

2.9.4.3.1 函数：venc_create_channel

C函数原型	<code>aclError aclvencCreateChannel(aclvencChannelDesc *channelDesc)</code>
Python函数	<code>ret = acl.media.venc_create_channel(venc_channel_desc)</code>
函数功能	创建视频数据编码处理的通道，同一个通道可以重复使用，销毁后不再可用。 Atlas 训练系列产品 ，不支持该接口。 Atlas A2训练系列产品/Atlas 800I A2推理产品 ，不支持该接口。
输入说明	venc_channel_desc : int，指定通道描述信息。 调用 acl.media.venc_create_channel_desc 接口创建 aclvencChannelDesc 类型的数据，调用 acl.media.venc_set_channel_desc 系列接口设置通道描述信息的属性。
返回值说明	ret : int，错误码。 - 返回0表示成功。 - 返回其它值表示失败。

约束说明	• 通道为非线程安全，即不同线程要求创建不同的通道。 • 通道数的限制，请参见2.9.1 总体说明。 • <code>acl.media.venc_create_channel</code>接口内部调用2次acl.rt.create_stream接口创建了2个Stream，一个Stream用于视频码流编码任务，另一个Stream用于回调函数的处理。这些Stream资源只有在调用acl.media.venc_destroy_channel接口销毁通道后才能自动释放。 • 用户在实现VENC功能时，需要遵循acl.rt.create_stream接口的约束，如Stream的数量等。
参考资源	接口调用流程及示例，参见媒体数据处理V1。

2.9.4.3.2 函数：venc_destroy_channel

C函数原型	<code>aclError aclvencDestroyChannel(aclvencChannelDesc *channelDesc)</code>
Python函数	<code>ret = acl.media.venc_destroy_channel(venc_channel_desc)</code>
函数功能	销毁视频数据编码处理的通道，只能销毁通过 acl.media.venc_create_channel 接口创建的通道。 Atlas 训练系列产品，不支持该接口。 Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。
输入说明	<code>venc_channel_desc</code> : int，指定通道描述信息，与调用 acl.media.venc_create_channel 接口创建通道时指定的channelDesc保持一致。
返回值说明	<code>ret</code> : int，错误码。 • 返回0表示成功。 • 返回其它值表示失败。
约束说明	• 销毁通道接口会等待用户的回调函数处理完成后再销毁。 • 如果在调用acl.media.venc_destroy_channel接口销毁通道前，已调用acl.media.venc_destroy_channel_desc接口销毁了通道描述信息，那么在调用acl.media.venc_destroy_channel接口销毁通道时会报错。
参考资源	接口调用流程及示例，参见媒体数据处理V1。

2.9.5 common

2.9.5.1 函数：dvpp_create_channel_desc

C函数原型	<code>acldvppChannelDesc *acldvppCreateChannelDesc()</code>

Python函数	dvpp_channel_desc = acl.media.dvpp_create_channel_desc()
函数功能	创建aclDvppChannelDesc类型的数据，表示创建图片数据处理通道时的通道描述信息。
输入说明	无
返回值说明	dvpp_channel_desc : int，返回aclDvppChannelDesc类型的指针地址。 - 若是非0值，表示成功。 - 若是0，表示失败。
约束说明	无

2.9.5.2 函数：dvpp_destroy_channel_desc

C函数原型	aclError aclDvppDestroyChannelDesc(aclDvppChannelDesc *channelDesc)
Python函数	ret = acl.media.dvpp_destroy_channel_desc(dvpp_channel_desc)
函数功能	销毁通过 acl.media.dvpp_create_channel_desc 接口创建的aclDvppChannelDesc类型的数据。
输入说明	dvpp_channel_desc : int，指定需要销毁的通道描述信息的指针地址。
返回值说明	ret : int，错误码。 - 返回0表示成功。 - 返回其它值表示失败。

2.9.5.3 函数：dvpp_get_channel_desc_channel_id

C函数原型	uint64_t aclDvppGetChannelDescChannelId(const aclDvppChannelDesc *channelDesc)
Python函数	channel_id = acl.media.dvpp_get_channel_desc_channel_id(dvpp_channel_desc)
函数功能	根据通道描述信息获取通道ID。
输入说明	dvpp_channel_desc : int，指定通道描述信息的指针地址。 需提前调用 acl.media.dvpp_create_channel_desc 接口创建aclDvppChannelDesc类型的数据。
返回值说明	channel_id : int，返回通道ID。

约束说明	接口调用顺序： 1. acl.media.dvpp_create_channel_desc 2. acl.media.dvpp_create_channel 3. acl.media.dvpp_get_channel_desc_channel_id 。

2.9.5.4 函数：dvpp_set_channel_desc_mode

C函数原型	<code>aclError acldvppSetChannelDescMode(acldvppChannelDesc *channelDesc, uint32_t mode)</code>
Python函数	<code>ret = acl.media.dvpp_set_channel_desc_mode(dvpp_channel_desc, mode)</code>
函数功能	指定通道描述信息中的通道模式，明确图片数据处理通道用于实现哪种功能，目前支持VPC、JPEGD、JPEGE、PNGD功能。 Atlas 200/300/500 推理产品 ，当前版本不支持该接口。 Atlas 训练系列产品 ，当前版本不支持该接口。
输入说明	dvpp_channel_desc : int，指定通道描述信息的指针地址。 需提前调用 acl.media.dvpp_create_channel_desc 接口创建的acldvppChannelDesc数据。 mode : int，指定通道模式，具体请参见 2.16.33 acldvppChannelMode 。
返回值说明	ret : int，错误码。 - 返回0表示成功。 - 返回其它值表示失败。
约束说明	接口调用顺序： 1. acl.media.dvpp_create_channel_desc 2. acl.media.dvpp_set_channel_desc_mode 3. acl.media.dvpp_create_channel

2.9.5.5 函数：dvpp_set_channel_desc_param

C函数原型	<code>aclError acldvppSetChannelDescParam(acldvppChannelDesc *channelDesc, acldvppChannelDescParamType paramType, size_t length, const void *param)</code>
Python函数	<code>channelDesc, ret=acl.media.dvpp_set_channel_desc_param(channelDesc, paramType, param)</code>
函数功能	设置图片数据处理通道的属性参数接口。

输入说明	channelDesc : int, 指定通道描述信息的指针地址。 需提前调用 acl.media.dvpp_create_channel_desc 接口创建acldvppChannelDesc类型的数据。 paramType : int, 属性参数的类型, 具体请参见 2.16.24 acldvppChannelDescParamType 。 param : int, 指向属性参数值。
返回值说明	channelDesc : int, 指定通道描述信息的指针地址。 ret : int, 错误码。 - 返回0表示成功。 - 返回其它值表示失败。
约束说明	无

2.9.5.6 函数: dvpp_get_channel_desc_param

C函数原型	aclError acldvppGetChannelDescParam(const acldvppChannelDesc *channelDesc, acldvppChannelDescParamType paramType, size_t length, size_t *paramRetSize, void *param)
Python函数	paramRetSize, param, ret=acl.media.dvpp_get_channel_desc_param(channelDesc, paramType)
函数功能	获取图片数据处理通道的属性参数接口。
输入说明	channelDesc : int, 指定通道描述信息的指针地址。 需提前调用 acl.media.dvpp_create_channel_desc 接口创建acldvppChannelDesc类型的数据, 调用 acl.media.dvpp_set_channel_desc_param 接口设置通道属性值。 paramType : int, 属性参数的类型, 具体请参见 2.16.24 acldvppChannelDescParamType 。
返回值说明	paramRetSize : int, 实际返回的属性参数值的字节数。 param : int, 指向属性参数值的指针地址。 ret : int, 错误码。 - 返回0表示成功。 - 返回其它值表示失败。
约束说明	无

2.9.5.7 函数：vdec_set_channel_desc_param

C函数原型	<code>aclError aclvdecSetChannelDescParam(aclvdecChannelDesc *channelDesc, aclvdecChannelDescParamType paramType, size_t length, const void *param)</code>
Python函数	<code>channelDesc, ret = acl.media.vdec_set_channel_desc_param(channelDesc, paramType, param)</code>
函数功能	设置视频编码处理通道描述信息的属性。
输入说明	channelDesc : int, 视频解码处理通道的描述信息的指针地址。 需提前调用 acl.media.dvpp_create_channel_desc 接口创建 aclvdecChannelDesc 类型的数据。 paramType : int, 属性参数的类型, 具体请参见 2.16.25 aclvdecChannelDescParamType 。 param : int, 指向属性参数值的指针地址。
返回值说明	channelDesc : int, 表示视频解码处理通道的描述信息。 ret : int, 错误码。 - 返回0表示成功。 - 返回其它值表示失败。
约束说明	无

2.9.5.8 函数：vdec_get_channel_desc_param

C函数原型	<code>aclError aclvdecGetChannelDescParam(const aclvdecChannelDesc *channelDesc, aclvdecChannelDescParamType paramType, size_t length, size_t *paramRetSize, void *param)</code>
Python函数	<code>paramRetSize, param, ret = acl.media.vdec_get_channel_desc_param(channelDesc, paramType)</code>
函数功能	获取视频解码处理通道的描述信息。
输入说明	channelDesc : int, 视频解码数据处理通道的描述信息的指针地址。 需提前调用 acl.media.dvpp_create_channel_desc 接口创建 aclvdecChannelDesc 类型的数据, 调用 acl.media.vdec_set_channel_desc_param 接口设置视频编码处理通道描述信息的属性值。 paramType : int, 属性参数的类型, 具体请参见 2.16.25 aclvdecChannelDescParamType 。

返回值说明	paramRetSize : int, 实际返回的属性参数值的字节数。 param : int, 指向属性参数值。 ret : int, 错误码。 • 返回0表示成功。 • 返回其它值表示失败。
约束说明	无

2.9.5.9 函数: dvpp_create_pic_desc

C函数原型	acldvppPicDesc *acldvppCreatePicDesc()
Python函数	dvpp_pic_desc = acl.media.dvpp_create_pic_desc()
函数功能	创建acldvppPicDesc类型的数据, 表示图片描述信息。 如需销毁acldvppPicDesc类型的数据, 请参见 acl.media.dvpp_destroy_pic_desc 。
输入说明	无
返回值说明	dvpp_pic_desc : int, 返回acldvppPicDesc类型的指针地址。 • 非0, 表示成功。 • 0, 表示失败。
约束说明	无

2.9.5.10 函数: dvpp_destroy_pic_desc

C函数原型	aclError acldvppDestroyPicDesc(acldvppPicDesc *picDesc)
Python函数	ret = acl.media.dvpp_destroy_pic_desc(dvpp_pic_desc)
函数功能	销毁通过 acl.media.dvpp_create_pic_desc 接口创建的acldvppPicDesc类型的数据。
输入说明	dvpp_pic_desc : int, 指定待销毁的图片描述信息指针地址。
返回值说明	ret : int, 错误码。 • 返回0表示成功。 • 返回其它值表示失败。
约束说明	无

2.9.5.11 函数：dvpp_set_pic_desc_data

C函数原型	aclError aclDvppSetPicDescData(aclDvppPicDesc *picDesc, void *dataDev)
Python函数	ret = acl.media.dvpp_set_pic_desc_data(dvpp_pic_desc, data_dev)
函数功能	设置图片描述参数：图片数据内存。
输入说明	dvpp_pic_desc : int，图片描述信息的指针地址。 需提前调用 acl.media.dvpp_create_pic_desc 接口创建aclDvppPicDesc类型的数据。 data_dev : int，Device上图片数据内存的指针地址，必须是使用 acl.media.dvpp_malloc 接口申请的内存。
返回值说明	ret : int，错误码。 • 返回0表示成功。 • 返回其它值表示失败。
约束说明	无

2.9.5.12 函数：dvpp_set_pic_desc_size

C函数原型	aclError aclDvppSetPicDescSize(aclDvppPicDesc *picDesc, uint32_t size)
Python函数	ret = acl.media.dvpp_set_pic_desc_size(dvpp_pic_desc, size)
函数功能	设置图片描述参数：图片数据内存大小。
输入说明	dvpp_pic_desc : int，图片描述信息的指针地址。 需提前调用 acl.media.dvpp_create_pic_desc 接口创建aclDvppPicDesc类型的数据。 size : int，内存大小，单位Byte。
返回值说明	ret : int，错误码。 • 返回0表示成功。 • 返回其它值表示失败。
约束说明	无

2.9.5.13 函数：dvpp_set_pic_desc_format

C函数原型	aclError aclDvppSetPicDescFormat(aclDvppPicDesc *picDesc, aclDvppPixelFormat format)
Python函数	ret = acl.media.dvpp_set_pic_desc_format(dvpp_pic_desc, format)

函数功能	设置图片描述参数：图片格式。
输入说明	dvpp_pic_desc : int, 图片描述信息的指针地址。 需提前调用 acl.media.dvpp_create_pic_desc 接口创建acldvppPicDesc类型的数据。 format : int, 图片格式, 详细参数请参见 2.16.6 acldvppPixelFormat 。
返回值说明	ret : int, 错误码。 - 返回0表示成功。 - 返回其它值表示失败。
约束说明	无

2.9.5.14 函数：dvpp_set_pic_desc_width

C函数原型	aclError acldvppSetPicDescWidth(acldvppPicDesc *picDesc, uint32_t width)
Python函数	ret = acl.media.dvpp_set_pic_desc_width(dvpp_pic_desc, width)
函数功能	设置图片描述参数：原始图片宽。
输入说明	dvpp_pic_desc : int, 图片描述信息的指针地址。 需提前调用 acl.media.dvpp_create_pic_desc 接口创建acldvppPicDesc类型的数据。 width : int, 原始图片宽。
返回值说明	ret : int, 错误码。 - 返回0表示成功。 - 返回其它值表示失败。
约束说明	无

2.9.5.15 函数：dvpp_set_pic_desc_height

C函数原型	aclError acldvppSetPicDescHeight(acldvppPicDesc *picDesc, uint32_t height)
Python函数	ret = acl.media.dvpp_set_pic_desc_height(dvpp_pic_desc, height)
函数功能	设置图片描述参数：原始图片高。

输入说明	dvpp_pic_desc : int, 图片描述信息的指针地址。 需提前调用 acl.media.dvpp_create_pic_desc 接口创建aclDvppPicDesc类型的数据。 height : int, 原始图片高。
返回值说明	ret : int, 错误码。 - 返回0表示成功。 - 返回其它值表示失败。
约束说明	无

2.9.5.16 函数: dvpp_set_pic_desc_width_stride

C函数原型	aclError aclDvppSetPicDescWidthStride(aclDvppPicDesc *picDesc, uint32_t widthStride)
Python函数	ret = acl.media.dvpp_set_pic_desc_width_stride(dvpp_pic_desc, width_stride)
函数功能	设置图片描述参数: 对齐后的宽。
输入说明	dvpp_pic_desc : int, 图片描述信息的指针地址。 需提前调用 acl.media.dvpp_create_pic_desc 接口创建aclDvppPicDesc类型的数据。 width_stride : int, 对齐后的宽。
返回值说明	ret : int, 错误码。 - 返回0表示成功。 - 返回其它值表示失败。
约束说明	无

2.9.5.17 函数: dvpp_set_pic_desc_height_stride

C函数原型	aclError aclDvppSetPicDescHeightStride(aclDvppPicDesc *picDesc, uint32_t heightStride)
Python函数	ret = acl.media.dvpp_set_pic_desc_height_stride(dvpp_pic_desc, height_stride)
函数功能	设置图片描述参数: 对齐后的高。

输入说明	dvpp_pic_desc : int, 图片描述信息的指针地址。 需提前调用 acl.media.dvpp_create_pic_desc 接口创建acldvppPicDesc类型的数据。 height_stride : int, 对齐后的高。
返回值说明	ret : int, 错误码。 - 返回0表示成功。 - 返回其它值表示失败。
约束说明	无

2.9.5.18 函数: dvpp_set_pic_desc_ret_code

C函数原型	aclError acldvppSetPicDescRetCode(acldvppPicDesc *picDesc, uint32_t retCode)
Python函数	ret = acl.media.dvpp_set_pic_desc_ret_code(dvpp_pic_desc, ret_code)
函数功能	设置图片描述参数: 返回码。
输入说明	dvpp_pic_desc : int, 图片描述信息的指针地址。 需提前调用 acl.media.dvpp_create_pic_desc 接口创建acldvppPicDesc类型的数据。 ret_code : int, 返回码。 0成功。 - 非0失败。
返回值说明	ret : int, 错误码。 - 返回0表示成功。 - 返回其它值表示失败。
约束说明	无

2.9.5.19 函数: dvpp_get_pic_desc_data

C函数原型	void *acldvppGetPicDescData(const acldvppPicDesc *picDesc)
Python函数	data_dev = acl.media.dvpp_get_pic_desc_data(dvpp_pic_desc)
函数功能	获取图片描述参数: 图片数据内存。
输入说明	dvpp_pic_desc : int, 图片描述信息的指针地址。
返回值说明	data_dev : int, 图片数据内存。

约束说明	无

2.9.5.20 函数：dvpp_get_pic_desc_size

C函数原型	<code>uint32_t aclDvppGetPicDescSize(const aclDvppPicDesc *picDesc)</code>
Python函数	<code>size = acl.media.dvpp_get_pic_desc_size(dvpp_pic_desc)</code>
函数功能	获取图片描述参数：图片数据内存大小。
输入说明	<code>dvpp_pic_desc</code> ：int，图片描述信息的指针地址。
返回值说明	<code>size</code> ：int，图片数据内存大小。
约束说明	无

2.9.5.21 函数：dvpp_get_pic_desc_format

C函数原型	<code>aclDvppPixelFormat aclDvppGetPicDescFormat(const aclDvppPicDesc *picDesc)</code>
Python函数	<code>format = acl.media.dvpp_get_pic_desc_format(dvpp_pic_desc)</code>
函数功能	获取图片描述参数：图片格式。
输入说明	<code>dvpp_pic_desc</code> ：int，图片描述信息的指针地址。
返回值说明	<code>format</code> ：int，图片格式，具体请参见 2.16.6 aclDvppPixelFormat 。
约束说明	无

2.9.5.22 函数：dvpp_get_pic_desc_width

C函数原型	<code>uint32_t aclDvppGetPicDescWidth(const aclDvppPicDesc *picDesc)</code>
Python函数	<code>width = acl.media.dvpp_get_pic_desc_width(dvpp_pic_desc)</code>
函数功能	获取图片描述参数：原始图片宽。
输入说明	<code>dvpp_pic_desc</code> ：int，图片描述信息的指针地址。
返回值说明	<code>width</code> ：int，原始图片宽。
约束说明	无

2.9.5.23 函数：dvpp_get_pic_desc_height

C函数原型	uint32_t aclDvppGetPicDescHeight(const aclDvppPicDesc *picDesc)
Python函数	height = acl.media.dvpp_get_pic_desc_height(dvpp_pic_desc)
函数功能	获取图片描述参数：原始图片高。
输入说明	dvpp_pic_desc: int，图片描述信息的指针地址。
返回值说明	height: int，原始图片高。
约束说明	无

2.9.5.24 函数：dvpp_get_pic_desc_width_stride

C函数原型	uint32_t aclDvppGetPicDescWidthStride(const aclDvppPicDesc *picDesc)
Python函数	width_stride = acl.media.dvpp_get_pic_desc_width_stride(dvpp_pic_desc)
函数功能	获取图片描述参数：对齐后的宽。
输入说明	dvpp_pic_desc: int，图片描述信息的指针地址。
返回值说明	width_stride: int，对齐后的宽。
约束说明	无

2.9.5.25 函数：dvpp_get_pic_desc_height_stride

C函数原型	uint32_t aclDvppGetPicDescHeightStride(const aclDvppPicDesc *picDesc)
Python函数	height_stride = acl.media.dvpp_get_pic_desc_height_stride(dvpp_pic_desc)
函数功能	获取图片描述参数：对齐后的高。
输入说明	dvpp_pic_desc: int，图片描述信息的指针地址。
返回值说明	height_stride: int，对齐后的高。
约束说明	无

2.9.5.26 函数：dvpp_get_pic_desc_ret_code

C函数原型	<code>uint32_t aclDvppGetPicDescRetCode(const aclDvppPicDesc *picDesc)</code>
Python函数	<code>ret_code = acl.media.dvpp_get_pic_desc_ret_code(dvpp_pic_desc)</code>
函数功能	获取图片描述参数：返回码。
输入说明	<code>dvpp_pic_desc</code> : int, 图片描述信息的指针地址。
返回值说明	<code>ret_code</code> : int, 返回码。 • 0成功。 • 非0失败。 VDEC解码时，用户需调用 acl.media.dvpp_get_pic_desc_ret_code 接口获取“ret_code”返回码判断是否解码成功，如果解码失败，需要根据日志中的返回码判断具体的问题，返回码请参见返回码说明。“ret_code”中部分返回码是以十六进制形式定义的，如用户在编码时以十进制打印返回码的值（例如，65540），则需提前转换为十六进制（例如，0x10004），然后在返回码说明中查询。
约束说明	无

2.9.5.27 函数：dvpp_create_stream_desc

C函数原型	<code>aclDvppStreamDesc *aclDvppCreateStreamDesc()</code>
Python函数	<code>dvpp_stream_desc = acl.media.dvpp_create_stream_desc()</code>
函数功能	创建 aclDvppStreamDesc 类型的数据，表示视频码流描述信息。如需销毁 aclDvppStreamDesc 类型的数据，请参见 acl.media.dvpp_create_stream_desc 。
输入说明	无
返回值说明	<code>dvpp_stream_desc</code> : int, 返回 aclDvppStreamDesc 类型的指针地址。 • 非0，表示成功。 • 0，表示失败。
约束说明	无

2.9.5.28 函数：dvpp_destroy_stream_desc

C函数原型	<code>aclError aclDvppDestroyStreamDesc(aclDvppStreamDesc *streamDesc)</code>

Python函数	ret = acl.media.dvpp_destroy_stream_desc(dvpp_stream_desc)
函数功能	销毁视频码流描述信息，只能销毁通过 acl.media.dvpp_create_stream_desc接口创建的视频码流描述信息。
输入说明	dvpp_stream_desc: int，指定待销毁的视频码流描述信息的指针地址。
返回值说明	ret: int，错误码。 • 返回0表示成功。 • 返回其它值表示失败。
约束说明	无

2.9.5.29 函数：dvpp_set_stream_desc_data

C函数原型	aclError aclDvppSetStreamDescData(aclDvppStreamDesc *streamDesc, void *dataDev)
Python函数	ret = acl.media.dvpp_set_stream_desc_data(dvpp_stream_desc, data_dev)
函数功能	设置视频码流描述参数：Device上视频码流数据内存。
输入说明	dvpp_stream_desc: int，视频码流描述信息的指针地址。 需提前调用acl.media.dvpp_create_stream_desc接口创建aclDvppStreamDesc类型的数据。 data_dev: int，Device上视频码流数据内存。
返回值说明	ret: int，错误码。 • 返回0表示成功。 • 返回其它值表示失败。
约束说明	无

2.9.5.30 函数：dvpp_set_stream_desc_size

C函数原型	aclError aclDvppSetStreamDescSize(aclDvppStreamDesc *streamDesc, uint32_t size)
Python函数	ret = acl.media.dvpp_set_stream_desc_size(dvpp_stream_desc, size)
函数功能	设置视频码流描述参数：视频码流数据内存大小。

输入说明	dvpp_stream_desc : int, 视频码流描述信息的指针地址。 需提前调用 acl.media.dvpp_create_stream_desc 接口创建acldvppStreamDesc类型的数据。 size : int, 内存大小, 单位Byte。
返回值说明	ret : int, 错误码。 • 返回0表示成功。 • 返回其它值表示失败。
约束说明	无

2.9.5.31 函数: dvpp_set_stream_desc_format

C函数原型	aclError acldvppSetStreamDescFormat(acldvppStreamDesc *streamDesc, acldvppStreamFormat format)
Python函数	ret = acl.media.dvpp_set_stream_desc_format(dvpp_stream_desc, format)
函数功能	设置视频码流描述参数: 视频码流格式。
输入说明	dvpp_stream_desc : int, 视频码流描述信息的指针地址。 需提前调用 acl.media.dvpp_create_stream_desc 接口创建acldvppStreamDesc类型的数据。 format : int, 视频码流格式 (H.264/H.265), 具体请参见 2.16.7 acldvppStreamFormat 。
返回值说明	ret : int, 错误码。 • 返回0表示成功。 • 返回其它值表示失败。
约束说明	无

2.9.5.32 函数: dvpp_set_stream_desc_timestamp

C函数原型	aclError acldvppSetStreamDescTimestamp(acldvppStreamDesc *streamDesc, uint64_t timestamp)
Python函数	ret = acl.media.dvpp_set_stream_desc_timestamp(dvpp_stream_desc, timestamp)
函数功能	设置视频码流描述参数: 时间戳。

输入说明	dvpp_stream_desc : int, 视频码流描述信息的指针地址。 需提前调用 acl.media.dvpp_create_stream_desc 接口创建 aclDvppStreamDesc 类型的数据。 timestamp : int, 时间戳。
返回值说明	ret : int, 错误码。 - 返回0表示成功。 - 返回其它值表示失败。
约束说明	无

2.9.5.33 函数: dvpp_set_stream_desc_ret_code

C函数原型	<code>aclError aclDvppSetStreamDescRetCode(aclDvppStreamDesc *streamDesc, uint32_t retCode)</code>
Python函数	<code>ret = acl.media.dvpp_set_stream_desc_ret_code(dvpp_stream_desc, ret_code)</code>
函数功能	设置视频码流描述参数: 返回码。
输入说明	dvpp_stream_desc : int, 视频码流描述信息的指针地址。 需提前调用 acl.media.dvpp_create_stream_desc 接口创建 aclDvppStreamDesc 类型的数据。 ret_code : int, 返回码。 - 返回0成功。 - 返回非0失败。
返回值说明	ret : int, 错误码。 - 返回0表示成功。 - 返回其它值表示失败。
约束说明	无

2.9.5.34 函数: dvpp_set_stream_desc_eos

C函数原型	<code>aclError aclDvppSetStreamDescEos(aclDvppStreamDesc *streamDesc, uint8_t eos)</code>
Python函数	<code>ret = acl.media.dvpp_set_stream_desc_eos(dvpp_stream_desc, eos)</code>
函数功能	设置视频码流描述参数: 是否为最后一帧数据。

输入说明	dvpp_stream_desc : int, 视频码流描述信息的指针地址。 需提前调用 acl.media.dvpp_create_stream_desc 接口创建acldvppStreamDesc类型的数据。 eos : int, 是否为最后一帧数据。 0: 否 1: 是
返回值说明	ret : int, 错误码。 - 返回0表示成功。 - 返回其它值表示失败。
约束说明	无

2.9.5.35 函数: dvpp_get_stream_desc_data

C函数原型	void *acldvppGetStreamDescData(const acldvppStreamDesc *streamDesc)
Python函数	data_dev = acl.media.dvpp_get_stream_desc_data(dvpp_stream_desc)
函数功能	获取视频码流描述参数: Device上视频码流数据内存。
输入说明	dvpp_stream_desc : int, 视频码流描述信息的指针地址。
返回值说明	data_dev : int, Device上视频码流数据内存。
约束说明	无

2.9.5.36 函数: dvpp_get_stream_desc_size

C函数原型	uint32_t acldvppGetStreamDescSize(const acldvppStreamDesc *streamDesc)
Python函数	size = acl.media.dvpp_get_stream_desc_size(dvpp_stream_desc)
函数功能	获取视频码流描述参数: 视频码流数据内存大小。
输入说明	dvpp_stream_desc : int, 视频码流描述信息的指针地址。
返回值说明	size : int, 内存大小, 单位Byte。
约束说明	无

2.9.5.37 函数：dvpp_get_stream_desc_format

C函数原型	<code>acldvppStreamFormat acldvppGetStreamDescFormat(const acldvppStreamDesc *streamDesc)</code>
Python函数	<code>format = acl.media.dvpp_get_stream_desc_format(dvpp_stream_desc)</code>
函数功能	获取视频码流描述参数：视频码流格式。
输入说明	<code>dvpp_stream_desc</code> : int, 视频码流描述信息的指针地址。
返回值说明	<code>format</code> : int, 视频码流格式 (H.264/H.265), 具体请参见 2.16.7 acldvppStreamFormat 。
约束说明	无

2.9.5.38 函数：dvpp_get_stream_desc_timestamp

C函数原型	<code>uint64_t acldvppGetStreamDescTimestamp(const acldvppStreamDesc *streamDesc)</code>
Python函数	<code>timestamp = acl.media.dvpp_get_stream_desc_timestamp(dvpp_stream_desc)</code>
函数功能	获取视频码流描述参数：时间戳。
输入说明	<code>dvpp_stream_desc</code> : int, 视频码流描述信息的指针地址。
返回值说明	<code>timestamp</code> : int, 时间戳。
约束说明	无

2.9.5.39 函数：dvpp_get_stream_desc_ret_code

C函数原型	<code>uint32_t acldvppGetStreamDescRetCode(const acldvppStreamDesc *streamDesc)</code>
Python函数	<code>ret_code = acl.media.dvpp_get_stream_desc_ret_code(dvpp_stream_desc)</code>
函数功能	获取视频码流描述参数：返回码。
输入说明	<code>dvpp_stream_desc</code> : int, 视频码流描述信息的指针地址。
返回值说明	<code>ret_code</code> : int, 返回码。 • 返回0成功。 • 返回非0失败。
约束说明	无

2.9.5.40 函数：dvpp_get_stream_desc_eos

C函数原型	uint8_t aclDvppGetStreamDescEos(const aclDvppStreamDesc *streamDesc)
Python函数	eos = acl.media.dvpp_get_stream_desc_eos(dvpp_stream_desc)
函数功能	获取视频码流描述参数：是否为最后一帧数据。
输入说明	dvpp_stream_desc: int, 视频码流描述信息的指针地址。
返回值说明	eos: int, 是否为最后一帧数据。 • 0: 否 • 1: 是
约束说明	无

2.9.5.41 函数：dvpp_create_batch_pic_desc

C函数原型	aclDvppBatchPicDesc *aclDvppCreateBatchPicDesc(uint32_t batchSize)
Python函数	batch_pic_desc = acl.media.dvpp_create_batch_pic_desc(batch_size)
函数功能	创建aclDvppBatchPicDesc类型的数据，表示批量图片描述信息。如需销毁aclDvppBatchPicDesc类型的数据，请参见 acl.media.dvpp_destroy_batch_pic_desc 。
输入说明	batch_size: int, 图片数量。
返回值说明	batch_pic_desc: int, 返回创建描述信息的地址。 • 若是非0值，表示成功。 • 若是0，表示失败。
约束说明	无

2.9.5.42 函数：dvpp_destroy_batch_pic_desc

C函数原型	aclError aclDvppDestroyBatchPicDesc (aclDvppBatchPicDesc *batchPicDesc)
Python函数	ret = acl.media.dvpp_destroy_batch_pic_desc(batch_pic_desc)
函数功能	销毁通过 acl.media.dvpp_create_batch_pic_desc 接口创建的aclDvppBatchPicDesc类型的数据。

输入说明	batch_pic_desc : int, 指定需要销毁的acldvppBatchPicDesc类型数据的指针地址。
返回值说明	ret : int, 返回码。 • 返回0成功。 • 返回非0失败。
约束说明	无

2.9.5.43 函数: dvpp_get_pic_desc

C函数原型	acldvppPicDesc *acldvppGetPicDesc(acldvppBatchPicDesc *batchPicDesc, uint32_t index)
Python函数	pic_desc = acl.media.dvpp_get_pic_desc(batch_pic_desc, index)
函数功能	获取指定图片的图片描述信息。
输入说明	batch_pic_desc : int, 指定批量图片描述信息的指针地址。 需提前调用 acl.media.dvpp_create_batch_pic_desc 接口创建acldvppBatchPicDesc类型的数据。 index : int, 指定获取第几个图片的描述信息, index值从0开始。
返回值说明	pic_desc : 返回指定图片的图片描述信息。
约束说明	无

2.9.5.44 函数: dvpp_create_border_config

C函数原型	acldvppBorderConfig *acldvppCreateBorderConfig()
Python函数	pic_desc, ret = acl.media.dvpp_create_border_config()
函数功能	创建acldvppBorderConfig类型的数据, 表示边界填充配置。 如需销毁acldvppBorderConfig类型的数据, 请参见 acl.media.dvpp_destroy_border_config 。 Atlas 200/300/500 推理产品 , 当前版本不支持该接口。 Atlas 训练系列产品 , 当前版本不支持该接口。
输入说明	无
返回值说明	pic_desc : int, 返回指定图片的图片描述信息的指针地址。 ret : int, 返回码。 • 返回0成功。 • 返回非0失败。
约束说明	无

2.9.5.45 函数：dvpp_destroy_border_config

C函数原型	aclError aclDvppDestroyBorderConfig(aclDvppBorderConfig *borderConfig)
Python函数	ret = acl.media.dvpp_destroy_border_config(border_config)
函数功能	销毁通过 dvpp_create_border_config 接口创建的aclDvppBorderConfig类型的数据。 Atlas 200/300/500 推理产品 ，当前版本不支持该接口。 Atlas 训练系列产品 ，当前版本不支持该接口。
输入说明	border_config : int，待销毁的aclDvppBorderConfig类型的指针地址。
返回值说明	ret : int，错误码。 - 返回0表示成功。 - 返回其它值表示失败。
约束说明	无

2.9.5.46 函数：dvpp_set_border_config_value

C函数原型	aclError aclDvppSetBorderConfigValue(aclDvppBorderConfig *borderConfig, uint32_t index, double value);
Python函数	ret = acl.media.dvpp_set_border_config_value(border_config, index, value)
函数功能	设置边界填充配置参数：颜色分量对应像素值。 Atlas 200/300/500 推理产品 ，当前版本不支持该接口。 Atlas 训练系列产品 ，当前版本不支持该接口。
输入说明	border_config : int，表示边界填充配置数据的指针地址。需提前调用 acl.media.dvpp_create_border_config 接口创建aclDvppBorderConfig类型的数据。 index : int，表示颜色分量数组的下标。0表示R或Y分量，1表示G或U分量，2表示B或V分量，3为预留值。 value : int，指定填充的各颜色分量的像素值。仅当“border_type”（填充枚举类型，具体请参见 2.9.5.47 函数：dvpp_set_border_config_border_type ）设置为“BORDER_CONSTANT”时，该参数有效。当前支持用户手动按顺序存放R、G、B分量或Y、U、V分量的值。若输入输出都是YUV格式，填充YUV分量的值，反之，填充的是RGB分量的值。
返回值说明	ret : int，错误码。 - 返回0表示成功。 - 返回其它值表示失败。
约束说明	无

2.9.5.47 函数：dvpp_set_border_config_border_type

C函数原型	aclError acldvppSetBorderConfigBorderType(acldvppBorderConfig *borderConfig, acldvppBorderType borderType);
Python函数	ret = acl.media.dvpp_set_border_config_border_type(border_config, border_type)
函数功能	设置边界填充配置参数：填充枚举类型。 Atlas 200/300/500 推理产品 ，当前版本不支持该接口。 Atlas 训练系列产品 ，当前版本不支持该接口。
输入说明	border_config : int，表示边界填充配置数据的指针地址。需提前调用 acl.media.dvpp_create_border_config 接口创建acldvppBorderConfig类型的数据。 border_type : int，填充枚举类型，具体请参见 2.16.21 acldvppBorderType 。
返回值说明	ret : int，错误码。 • 返回0表示成功。 • 返回其它值表示失败。
约束说明	无

2.9.5.48 函数：dvpp_set_border_config_top

C函数原型	aclError acldvppSetBorderConfigTop(acldvppBorderConfig *borderConfig, uint32_t top);
Python函数	ret = acl.media.dvpp_set_border_config_top(border_config, top)
函数功能	设置边界填充配置参数：图像上方填充的像素数。 Atlas 200/300/500 推理产品 ，当前版本不支持该接口。 Atlas 训练系列产品 ，当前版本不支持该接口。
输入说明	border_config : int，表示边界填充配置数据的指针地址。需提前调用 acl.media.dvpp_create_border_config 接口创建acldvppBorderConfig类型的数据。 top : int，图像上方填充的像素数。
返回值说明	ret : int，错误码。 • 返回0表示成功。 • 返回其它值表示失败。
约束说明	无

2.9.5.49 函数：dvpp_set_border_config_bottom

C函数原型	<code>aclError aclDvppSetBorderConfigBottom(aclDvppBorderConfig *borderConfig, uint32_t bottom);</code>
Python函数	<code>ret = acl.media.dvpp_set_border_config_bottom(border_config, bottom)</code>
函数功能	设置边界填充配置参数：图像下方填充的像素数。 Atlas 200/300/500 推理产品 ，当前版本不支持该接口。 Atlas 训练系列产品 ，当前版本不支持该接口。
输入说明	border_config : int，表示边界填充配置数据的指针地址。需提前调用 acl.media.dvpp_create_border_config 接口创建 aclDvppBorderConfig 类型的数据。 bottom : int，图像下方填充的像素数。
返回值说明	ret : int，错误码。 • 返回0表示成功。 • 返回其它值表示失败。
约束说明	无

2.9.5.50 函数：dvpp_set_border_config_left

C函数原型	<code>aclError aclDvppSetBorderConfigLeft(aclDvppBorderConfig *borderConfig, uint32_t left);</code>
Python函数	<code>ret = acl.media.dvpp_set_border_config_left(border_config, left)</code>
函数功能	设置边界填充配置参数：图像左侧填充的像素数。 Atlas 200/300/500 推理产品 ，当前版本不支持该接口。 Atlas 训练系列产品 ，当前版本不支持该接口。
输入说明	border_config : int，表示边界填充配置数据。需提前调用 acl.media.dvpp_create_border_config 接口创建 aclDvppBorderConfig 类型的数据。 left : int，图像左侧填充的像素数。
返回值说明	ret : int，错误码。 • 返回0表示成功。 • 返回其它值表示失败。
约束说明	无

2.9.5.51 函数：dvpp_set_border_config_right

C函数原型	<code>aclError aclDvppSetBorderConfigRight(aclDvppBorderConfig *borderConfig, uint32_t right)</code>
Python函数	<code>ret = acl.media.dvpp_set_border_config_right(border_config, right)</code>
函数功能	设置边界填充配置参数：图像右侧填充的像素数。 Atlas 200/300/500 推理产品 ，当前版本不支持该接口。 Atlas 训练系列产品 ，当前版本不支持该接口。
输入说明	border_config : int，表示边界填充配置数据的指针地址。需提前调用 acl.media.dvpp_create_border_config 接口创建 aclDvppBorderConfig 类型的数据。 right : int，图像右侧填充的像素数。
返回值说明	ret : int，错误码。 - 返回0表示成功。 - 返回其它值表示失败。
约束说明	无

2.9.5.52 函数：dvpp_get_border_config_value

C函数原型	<code>double aclDvppGetBorderConfigValue(const aclDvppBorderConfig *borderConfig, uint32_t index);</code>
Python函数	<code>value = acl.media.dvpp_get_border_config_value(border_config, index)</code>
函数功能	获取边界填充配置参数：颜色分量的值。 Atlas 200/300/500 推理产品 ，当前版本不支持该接口。 Atlas 训练系列产品 ，当前版本不支持该接口。
输入说明	border_config : int，边界填充配置数据的指针地址。 index : int，0表示R或Y分量，1表示G或U分量，2表示B或V分量，3是预留值。
返回值说明	value : 表示对应颜色分量的值。
约束说明	无

2.9.5.53 函数：dvpp_get_border_config_border_type

C函数原型	<code>aclDvppBorderType aclDvppGetBorderConfigBorderType(const aclDvppBorderConfig *borderConfig);</code>

Python函数	border_type = acl.media.dvpp_get_border_config_border_type(border_config)
函数功能	获取边界填充配置参数：填充枚举类型。 Atlas 200/300/500 推理产品 ，当前版本不支持该接口。 Atlas 训练系列产品 ，当前版本不支持该接口。
输入说明	border_config : int，边界填充配置数据的指针地址。
返回值说明	border_type : int，填充枚举类型，具体请参见 2.16.21 acldvppBorderType 。
约束说明	无

2.9.5.54 函数：dvpp_get_border_config_top

C函数原型	uint32_t acldvppGetBorderConfigTop(const acldvppBorderConfig *borderConfig);
Python函数	top = acl.media.dvpp_get_border_config_top(border_config)
函数功能	获取边界填充配置参数：图像上方填充的像素数。 Atlas 200/300/500 推理产品 ，当前版本不支持该接口。 Atlas 训练系列产品 ，当前版本不支持该接口。
输入说明	border_config : int，边界填充配置数据的指针地址。
返回值说明	top : int，图像上方填充的像素数。
约束说明	无

2.9.5.55 函数：dvpp_get_border_config_bottom

C函数原型	uint32_t acldvppGetBorderConfigBottom(const acldvppBorderConfig *borderConfig);
Python函数	bottom = acl.media.dvpp_get_border_config_bottom(border_config)
函数功能	获取边界填充配置参数：图像下方填充的像素数。 Atlas 200/300/500 推理产品 ，当前版本不支持该接口。 Atlas 训练系列产品 ，当前版本不支持该接口。
输入说明	border_config : int，边界填充配置数据的指针地址。
返回值说明	bottom : int，图像下方填充的像素数。
约束说明	无

2.9.5.56 函数：dvpp_get_border_config_left

C函数原型	<code>uint32_t aclDvppGetBorderConfigLeft(const aclDvppBorderConfig *borderConfig);</code>
Python函数	<code>left= acl.media.dvpp_get_border_config_left(border_config)</code>
函数功能	获取边界填充配置参数：图像左侧填充的像素数。 Atlas 200/300/500 推理产品 ，当前版本不支持该接口。 Atlas 训练系列产品 ，当前版本不支持该接口。
输入说明	border_config ：int，边界填充配置数据的指针地址。
返回值说明	left ：int，图像左侧填充的像素数。
约束说明	无

2.9.5.57 函数：dvpp_get_border_config_right

C函数原型	<code>uint32_t aclDvppGetBorderConfigRight(const aclDvppBorderConfig *borderConfig)</code>
Python函数	<code>right = acl.media.dvpp_get_border_config_right(border_config)</code>
函数功能	获取边界填充配置参数：图像右侧填充的像素数。 Atlas 200/300/500 推理产品 ，当前版本不支持该接口。 Atlas 训练系列产品 ，当前版本不支持该接口。
输入说明	border_config ：int，边界填充配置数据的指针地址。
返回值说明	right ：int，图像右侧填充的像素数。
约束说明	无

2.9.6 VPC 功能

2.9.6.1 功能说明

功能说明

VPC (Vision Preprocessing Core) 各功能的介绍、各版本的功能支持度如[表2-10](#)、[表2-11](#)所示。

本章的接口均为异步接口，调用接口成功仅表示任务下发成功，不表示任务执行成功。调用该接口后，需调用同步等待接口（例如，[acl.rt.synchronize_stream](#)）确保任务已执行完成。

表 2-10 功能介绍

功能	描述
抠图	从输入图片中抠取需要用的图片区域，支持一图多框和多图多框。
缩放	针对不同分辨率的图像，支持8K缩放、非8K缩放。 支持单图裁剪缩放（支持非压缩格式）、一图多框裁剪缩放（支持非压缩格式）。 其它缩放方式，如：原图缩放、等比例缩放（缩放前后图片的宽高比例相同）。
叠加	对抠图操作得到的图片进行缩放后，根据用户设置，将图片放在输出图片的指定区域。 该输出图片支持空白图片（由用户申请的空输出内存产生的）或已有图片（由用户申请输出内存后将已有图片读入输出内存），只有当输出图片是已有图片时，才表示叠加。
拼接	从输入图片中抠多张图片，对抠出的图进行缩放后，放到输出图片的指定区域。
图像金字塔	对原图进行多次高斯滤波之后向下采样产生不同尺寸的图像。
直方图统计	统计图像每个通道（RGB/YUV）的像素值分布。
色彩重映射	根据配置信息将图片从原图映射为另一张图。
边界填充	对图像进行边界填充。
格式转换	支持RGB格式、YUV格式之间的格式转换。
图像灰度化	将彩色图像转化为灰度图像。需注意，输入为灰度图像、输出只能为灰度图像。

表 2-11 产品功能支持度

功能	Atlas 200/300/500 推理产品	Atlas 训练系列产品	Atlas 推理系列产品	Atlas 200I/500 A2推理产品	Atlas A2训练系列产品/Atlas 800I A2推理产品
抠图	√	√	√	√	√
缩放	√	√	√	√	√
叠加	√	√	√	√	√
拼接	√	√	√	√	√
图像金字塔	x	x	√	x	x
直方图统计	x	x	√	√	√
色彩重映射	x	x	√	√	√

功能	Atlas 200/300/500 推理产品	Atlas 训练系列产品	Atlas 推理系列产品	Atlas 200I/500 A2推理产品	Atlas A2训练系列产品/Atlas 800I A2推理产品
边界填充	x	x	√	√	√
格式转换	√	√	√	√	√
图像灰度化	√ 从YUV420SP格式的输出图像数据中只取Y分量的数据	√ 从YUV420SP格式的输出图像数据中只取Y分量的数据	√ 实现图像灰度化的操作是输出YUV400格式的输出图片	√ 实现图像灰度化的操作是输出YUV400格式的输出图片	√ 实现图像灰度化的操作是输出YUV400格式的输出图片

功能示意图(Atlas 200/300/500 推理产品)(Atlas 训练系列产品)(Atlas 推理系列产品)

图 2-1 VPC 功能示意图（抠图+缩放+叠加）

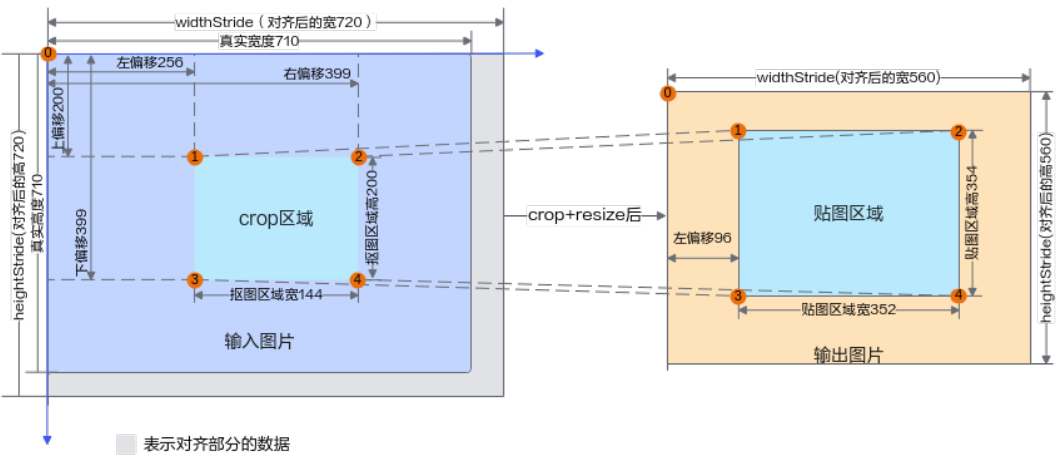


图 2-2 VPC 功能示意图（拼接）

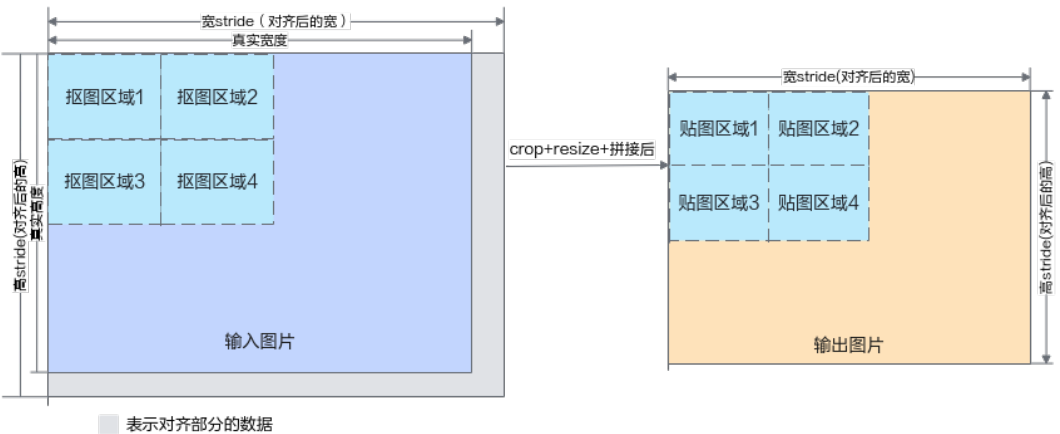


图 2-3 等比例缩放（贴图区域在输出图片的中心位置），即缩放前后图片的宽高比例相同

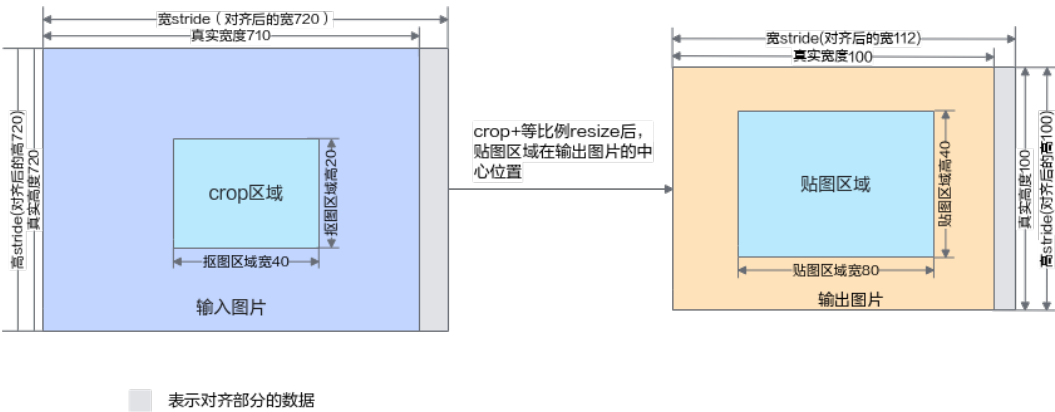


图 2-4 等比例缩放（贴图区域在输出图片的上下居中位置）

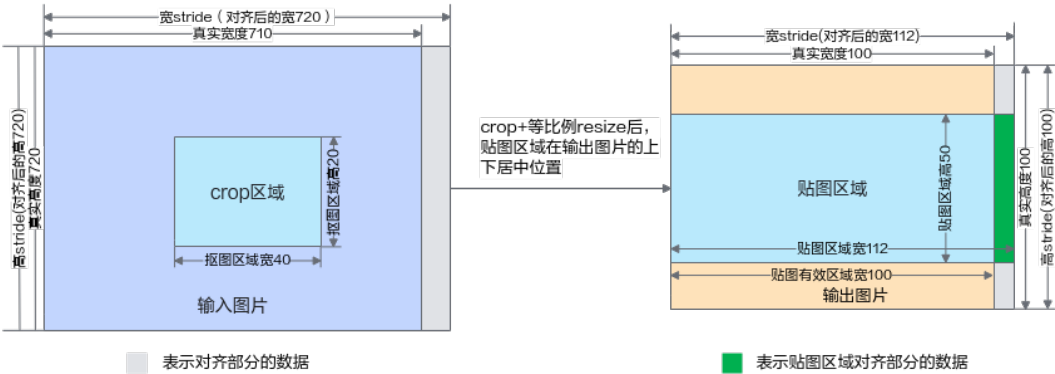
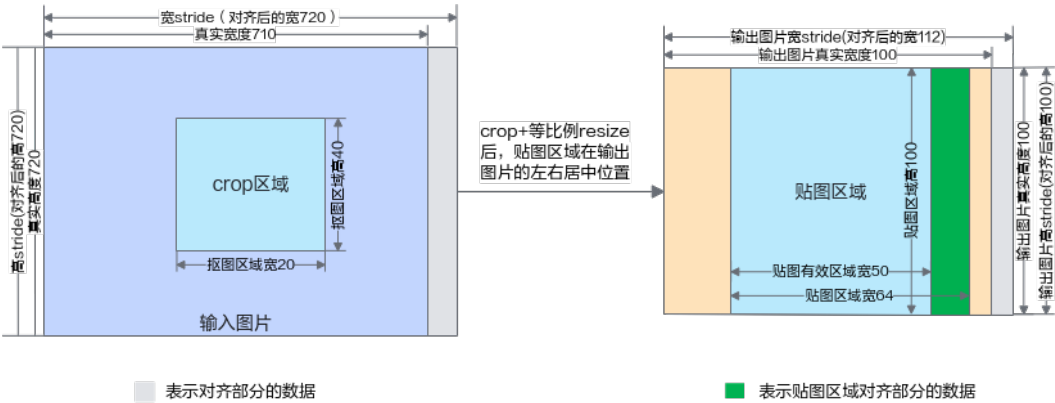


图 2-5 等比例缩放（贴图区域在输出图片的左右居中位置）



功能示意图(Atlas A2 训练系列产品/Atlas 800I A2 推理产品)(Atlas 200I/500 A2 推理产品)

图 2-6 VPC 功能示意图（抠图+缩放+叠加）

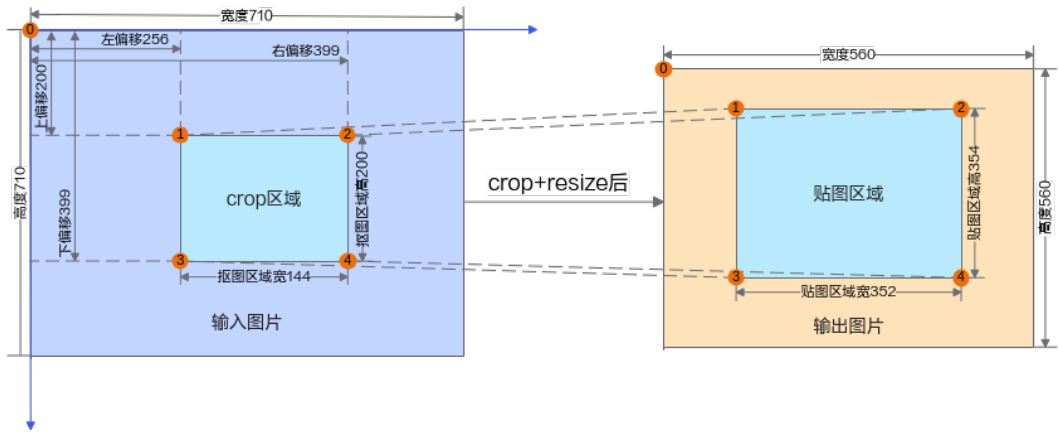


图 2-7 VPC 功能示意图（拼接）

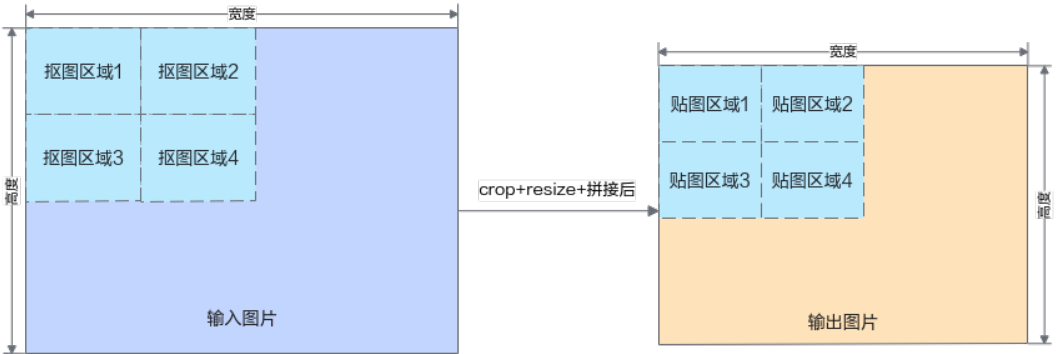
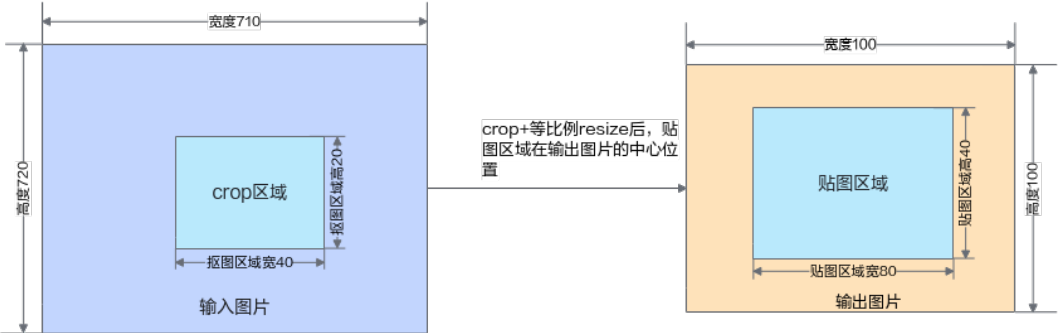


图 2-8 等比例缩放（贴图区域在输出图片的中心位置），即缩放前后图片的宽高比例相同



参考说明

RGB、YUV格式图像的各分量排布示意图。示例：SP图像以YUV420SP为例，Packed和RGB图像以ARGB图像为例。

格式	分辨率	宽stride	高stride	buffer大小					
yuv420sp	4*4	4	4	24					
内存排列									
y11	y12	y13	y14						
y21	y22	y23	y24						
y31	y32	y33	y34						
y41	y42	y43	y44						
u11	v11	u13	v13						
u31	v31	u33	v33						
格式	分辨率	宽stride	高stride	buffer大小					
yuv420sp	4*4	6	6	54					
内存排列 x为无效数据									
y11	y12	y13	y14	x	x				
y21	y22	y23	y24	x	x				
y31	y32	y33	y34	x	x				
y41	y42	y43	y44	x	x				
x	x	x	x	x	x				
x	x	x	x	x	x				
u11	v11	u13	v13	x	x				
u31	v31	u33	v33	x	x				
x	x	x	x	x	x				
格式	分辨率	宽stride	高stride	buffer大小					
argb	2*2	10	4	40					
内存排列 x为无效数据									
a11	r11	g11	b11	a12	r12	g12	b12	x	x
a21	r21	g21	b21	a22	r22	g22	b22	x	x
x	x	x	x	x	x	x	x	x	x
x	x	x	x	x	x	x	x	x	x

2.9.6.2 约束说明

- 本节中8K、非8K的概念如下：
- **非8K**：宽 * 高在10 * 6 ~ 4096 * 4096范围内，包括4096。
 - **8K**：宽或高在4096 ~ 8192，不包括4096。

图片分辨率约束

- 输入图片分辨率，支持如下分辨率的输入图片。
10 * 6 ~ 8192 * 8192
在调用接口实现VPC功能时，各接口对分辨率的要求可能不同，请参见[2.9.6 VPC功能](#)下各接口的说明。其中，当输入图片格式为YUV440SP、YUV440P时，输入图片的宽最大值为4096。
- 输出图片分辨率

版本	分辨率范围
Atlas 200/300/500 推理产品 Atlas 训练系列产品 Atlas 推理系列产品	10 * 6 ~ 4096 * 4096

版本	分辨率范围
Atlas A2训练系列产品/Atlas 800I A2推理产品 Atlas 200I/500 A2推理产品	10 * 6 ~ 4096 * 8192

图片格式、宽高对齐、内存约束

VPC在处理图片时，需调用[acl.media.dvpp_malloc](#)接口申请Device上的输入、输出内存，调用[acl.media.dvpp_free](#)接口释放输入、输出内存，这部分内存的生命周期由用户自行管理。

- 输入图片格式、宽高对齐约束请参见[表2-12](#)。
- 输出图片格式、宽高对齐约束请参见[表2-13](#)。

Atlas 200/300/500 推理产品，在8K场景下，输入图片格式仅支持YUV420SP NV12 8bit和YUV420SP NV21 8bit。

Atlas 训练系列产品，在8K场景下，输入图片格式仅支持YUV420SP NV12 8bit和YUV420SP NV21 8bit。

Atlas 推理系列产品，在8K场景下，输入图片格式支持如下格式：

- YUV400 8bit
- YUV420SP NV12 8bit、YUV420SP NV21 8bit
- YUV422SP 8bit、YVU422SP 8bit
- YUV444SP 8bit、YVU444SP 8bit
- YUV422P YUYV 8bit、YUV422P UYVY 8bit、YUV422P YVYU 8bit、YUV422P VYUY 8bit

Atlas A2训练系列产品/Atlas 800I A2推理产品，在8K场景下，输入图片格式支持如下格式：

- YUV400 8bit
- YUV420SP NV12 8bit、YUV420SP NV21 8bit
- YUV422SP 8bit、YVU422SP 8bit
- YUV444SP 8bit、YVU444SP 8bit
- YUV422P YUYV 8bit、YUV422P UYVY 8bit、YUV422P YVYU 8bit、YUV422P VYUY 8bit

Atlas 200I/500 A2推理产品，在8K场景下，输入图片格式支持如下格式：

- YUV400 8bit
- YUV420SP NV12 8bit、YUV420SP NV21 8bit
- YUV422SP 8bit、YVU422SP 8bit
- YUV444SP 8bit、YVU444SP 8bit
- YUV422P YUYV 8bit、YUV422P UYVY 8bit、YUV422P YVYU 8bit、YUV422P VYUY 8bit

 说明

在调用接口实现VPC功能时：

- 图片格式的定义请参见[acldvppPixelFormat](#)，宽Stride、高Stride、8K等概念请参见[2.9.2 基本概念](#)。
- 不同版本中，宽Stride、高Stride的最小值、最大值不同，具体参考如下：
 - Atlas 200/300/500 推理产品
 - 宽Stride最小32、最大16384（ $16384 = 4096 * 4$ ，宽是4096的argb格式的图像，1个像素占用4个字节，一行像素就占用 $4096 * 4$ ，即宽Stride）。
 - 高Stride最小6、最大16384。
 - Atlas 推理系列产品
 - 宽Stride最小32、最大16384（ $16384 = 4096 * 4$ ，宽是4096的argb格式的图像，1个像素占用4个字节，一行像素就占用 $4096 * 4$ ，即宽Stride）。
 - 高Stride最小6、最大16384。
 - Atlas 训练系列产品
 - 宽Stride最小32、最大16384（ $16384 = 4096 * 4$ ，宽是4096的argb格式的图像，1个像素占用4个字节，一行像素就占用 $4096 * 4$ ，即宽Stride）。
 - 高Stride最小6、最大16384。
 - Atlas 200I/500 A2推理产品
 - 宽Stride最小10、最大16384（ $16384 = 4096 * 4$ ，宽是4096的argb格式的图像，1个像素占用4个字节，一行像素就占用 $4096 * 4$ ，即宽Stride）。
 - 高Stride最小6、最大16384。
 - Atlas A2训练系列产品/Atlas 800I A2推理产品
 - 宽Stride最小10、最大16384（ $16384 = 4096 * 4$ ，宽是4096的argb格式的图像，1个像素占用4个字节，一行像素就占用 $4096 * 4$ ，即宽Stride）。
 - 高Stride最小6、最大16384。

表 2-12 输入图片格式、宽高对齐、内存大小约束

版本	输入图片格式	输入图片宽、高对齐要求	输入图片宽stride、高stride、内存大小要求 Atlas 200/300/500 推理产品 Atlas 训练系列产品	输入图片宽stride、高stride、内存大小要求 Atlas 推理系列产品	输入图片宽stride、高stride、内存大小要求 Atlas 200I/500 A2推理产品 Atlas A2训练系列产品/Atlas 800I A2推理产品
通用格式，各版本都支持	YUV400 8bit	无对齐要求	宽stride为宽16对齐后的值。 高stride为高2对齐后的值。 内存大小（单位Byte）= 宽stride * 高stride * 3/2	宽stride为宽16对齐后的值。 高stride无对齐要求，与高相同即可。 内存大小（单位Byte）= 宽stride * 高stride	宽stride无对齐要求，与宽相同即可。 高stride无对齐要求，与高相同即可。 内存大小（单位Byte）= 宽stride * 高stride

版本	输入图片格式	输入图片宽、高对齐要求	输入图片宽stride、高stride、内存大小要求 Atlas 200/300/500推理产品 Atlas 训练系列产品	输入图片宽stride、高stride、内存大小要求 Atlas 推理系列产品	输入图片宽stride、高stride、内存大小要求 Atlas 200I/500 A2推理产品 Atlas A2训练系列产品/Atlas 800I A2推理产品
	YUV420SP NV12 8bit	宽2对齐 高2对齐	宽stride为宽16对齐后的值。 高stride为高2对齐后的值。 内存大小 (单位Byte) = 宽stride * 高stride * 3/2	宽stride为宽16对齐后的值。 高stride为高2对齐后的值。 内存大小 (单位Byte) = 宽stride * 高stride * 3/2	宽stride无对齐要求，与宽相同即可。 高stride为高2对齐后的值。 内存大小 (单位Byte) = 宽stride * 高stride * 3/2
	YUV420SP NV21 8bit				
	YUV422SP 8bit	宽2对齐 高无对齐要求	宽stride为宽16对齐后的值。 高stride为高2对齐后的值。 内存大小 (单位Byte) = 宽stride * 高stride * 2	宽stride为宽16对齐后的值。 高stride无对齐要求，与高相同即可。 内存大小 (单位Byte) = 宽stride * 高stride * 2	宽stride无对齐要求，与宽相同即可。 高stride无对齐要求，与高相同即可。 内存大小 (单位Byte) = 宽stride * 高stride * 2
	YVU422SP 8bit				
	YUV444SP 8bit	无对齐要求	宽stride为宽16对齐后的值。 高stride为高2对齐后的值。 内存大小 (单位Byte) = 宽stride * 高stride * 3	宽stride为宽16对齐后的值。 高stride无对齐要求，与高相同即可。 内存大小 (单位Byte) = 宽stride * 高stride * 3	宽stride无对齐要求，与宽相同即可。 高stride无对齐要求，与高相同即可。 内存大小 (单位Byte) = 宽stride * 高stride * 3
	YVU444SP 8bit				
	YUV422Packed YUYV 8bit	宽2对齐 高无对齐要求	宽stride为宽16对齐后、再乘以2的值。 高stride为高2对齐后的值。 内存大小 (单位Byte) = 宽stride * 高stride	宽stride为宽16对齐后、再乘以2的值。 高stride无对齐要求，与高相同即可。 内存大小 (单位Byte) = 宽stride * 高stride	宽stride为宽乘以2的值。 高stride无对齐要求，与高相同即可。 内存大小 (单位Byte) = 宽stride * 高stride

版本	输入图片格式	输入图片宽、高对齐要求	输入图片宽stride、高stride、内存大小要求 Atlas 200/300/500推理产品 Atlas 训练系列产品	输入图片宽stride、高stride、内存大小要求 Atlas 推理系列产品	输入图片宽stride、高stride、内存大小要求 Atlas 200I/500 A2推理产品 Atlas A2训练系列产品/Atlas 800I A2推理产品
	YUV422Packe d UYVY 8bit				
	YUV422Packe d YVYU 8bit				
	YUV422Packe d VYUY 8bit				
	YUV444Packe d 8bit	无对齐要求	宽stride为宽16对齐后、再乘以3的值。 高stride为高2对齐后的值。 内存大小（单位Byte）= 宽stride * 高stride	宽stride为宽16对齐后、再乘以3的值。 高stride无对齐要求，与高相同即可。 内存大小（单位Byte）= 宽stride * 高stride	宽stride为宽乘以3的值。 高stride无对齐要求，与高相同即可。 内存大小（单位Byte）= 宽stride * 高stride
	RGB888				
	BGR888				
	ARGB8888	无对齐要求	宽stride为宽16对齐后、再乘以4的值。 高stride为高2对齐后的值。 内存大小（单位Byte）= 宽stride * 高stride	宽stride为宽16对齐后、再乘以4的值。 高stride无对齐要求，与高相同即可。 内存大小（单位Byte）= 宽stride * 高stride	宽stride为宽乘以4的。 高stride无对齐要求，与高相同即可。 内存大小（单位Byte）= 宽stride * 高stride
	ABGR8888				
	RGBA8888				
	BGRA8888				

版本	输入图片格式	输入图片宽、高对齐要求	输入图片宽stride、高stride、内存大小要求 Atlas 200/300/500推理产品 Atlas 训练系列产品	输入图片宽stride、高stride、内存大小要求 Atlas 推理系列产品	输入图片宽stride、高stride、内存大小要求 Atlas 200I/500 A2推理产品 Atlas A2训练系列产品/Atlas 800I A2推理产品
该部分格式仅以下版本支持： Atlas 推理系列产品 Atlas 200I/500 A2推理产品 Atlas A2训练系列产品/Atlas 800I A2推理产品	YUV440SP 8bit	宽无对齐要求 高2对齐	-	宽stride为宽16对齐后的值。 高stride为高2对齐后的值。 内存大小（单位Byte）= 宽stride * 高stride * 2	宽stride无对齐要求，与宽相同即可。 高stride为高2对齐后的值。 内存大小（单位Byte）= 宽stride * 高stride * 2
	YVU440SP 8bit				
该部分格式仅以下版本支持： Atlas 200I/500 A2推理产品 Atlas A2训练系列产品/Atlas 800I A2推理产品	YVU420Planar	宽2对齐，高2对齐	-	-	宽stride无对齐要求，与宽相同即可。 高stride无对齐要求，与高相同即可。 内存大小（单位Byte）= 宽stride * 高stride * 3/2
	YUV420Planar				
	YVU422Planar	无对齐要求			宽stride无对齐要求，与宽相同即可。 高stride无对齐要求，与高相同即可。 内存大小（单位Byte）= 宽stride * 高stride * 2
	YUV422Planar				

版本	输入图片格式	输入图片宽、高对齐要求	输入图片宽stride、高stride、内存大小要求 Atlas 200/300/500推理产品 Atlas 训练系列产品	输入图片宽stride、高stride、内存大小要求 Atlas 推理系列产品	输入图片宽stride、高stride、内存大小要求 Atlas 200I/500 A2推理产品 Atlas A2训练系列产品/Atlas 800I A2推理产品
	YVU444Planar	无对齐要求			宽stride无对齐要求，与宽相同即可。
	YUV444Planar				高stride无对齐要求，与高相同即可。
					内存大小（单位Byte）= 宽stride * 高stride * 3
	YVU444Packed 8bit	无对齐要求			宽stride无对齐要求，宽stride为宽乘以3的值。
					高stride无对齐要求，与高相同即可。
					内存大小（单位Byte）= 宽stride * 高stride
	YUV440Planar	宽无对齐要求，但宽≤4096 高2对齐			宽stride无对齐要求，与宽相同即可。
	YVU440Planar				高stride无对齐要求，与高相同即可。
					内存大小（单位Byte）= 宽stride * 高stride * 2
	RGB888Planar	无对齐要求			宽stride无对齐要求，与宽相同即可。
	BGR888Planar				高stride无对齐要求，与高相同即可。
					内存大小（单位Byte）= 宽stride * 高stride * 3

版本	输入图 片格式	输入图 片宽、 高对齐 要求	输入图片宽 stride、高 stride、内存 大小要求 Atlas 200/300/500 推理产品 Atlas 训练系 列产品	输入图片宽 stride、高 stride、内存大 小要求 Atlas 推理系列 产品	输入图片宽 stride、高 stride、内存大 小要求 Atlas 200I/500 A2推理产品 Atlas A2训练系 列产品/Atlas 800I A2推理产 品
Atlas A2训练 系列产 品/Atlas 800I A2 推理产 品	YUV40 0 FP32	无对齐 要求	-	-	宽stride无对齐要 求，宽stride为宽 乘以3的值；高 stride无对齐要 求，与高相同即 可。 内存大小（单位 Byte）= 宽stride * 高stride
	RGB88 8 FP32				
	BGR88 8 FP32				
	RGB88 8Plana r FP32 BGR88 8Plana r FP32	无对齐 要求			宽stride无对齐要 求，与宽相同即 可；高stride无对 齐要求，与高相 同即可。 内存大小（单位 Byte）= 宽stride * 高stride * 3

表 2-13 输出图片格式、宽高对齐、内存大小约束

版本	输出图片格式	输出图片宽、高对齐要求	输出图片宽stride、高stride、内存大小要求 Atlas 200/300/500 推理产品 Atlas 训练系列产品	输出图片宽stride、高stride、内存大小要求 Atlas 推理系列产品	输出图片宽stride、高stride、内存大小要求 Atlas 200I/500 A2推理产品 Atlas A2训练系列产品/Atlas 800I A2推理产品
通用格式，各版本都支持	YUV420 SP NV12 8bit	宽2对齐 高2对齐	宽stride为宽16对齐后的值。 高stride为高2对齐后的值。 内存大小（单位Byte）= 宽stride * 高stride * 3/2	宽stride为宽16对齐后的值。 高stride为高2对齐后的值。 内存大小（单位Byte）= 宽stride * 高stride * 3/2	宽stride无对齐要求，与宽相同即可。 高stride为高2对齐后的值。 内存大小（单位Byte）= 宽stride * 高stride * 3/2
	YUV420 SP NV21 8bit				
该部分格式仅以下版本支持： Atlas 推理系列产品 Atlas 200I/500 A2 推理产品 Atlas A2训练系列产品/Atlas 800I A2推理产品	YUV400 8bit	无对齐要求	-	宽stride为宽16对齐后的值。 高stride无对齐要求，与高相同即可。 内存大小（单位Byte）= 宽stride * 高stride	宽stride无对齐要求，与宽相同即可；高stride无对齐要求，与高相同即可。 内存大小（单位Byte）= 宽stride * 高stride
	YUV422 SP 8bit	宽2对齐		宽stride为宽16对齐后的值。 高stride无对齐要求，与高相同即可。 内存大小（单位Byte）= 宽stride * 高stride * 2	宽stride无对齐要求，与宽相同即可；高stride无对齐要求，与高相同即可。 内存大小（单位Byte）= 宽stride * 高stride * 2
	YVU422 SP 8bit	高无对齐要求			
	YUV444 Packed 8bit	无对齐要求		宽stride为宽16对齐后、再乘以3的值。 高stride无对齐要求，与高相同即可。 内存大小（单位Byte）= 宽stride * 高stride	宽stride为宽乘以3的值；高stride无对齐要求，与高相同即可。 内存大小（单位Byte）= 宽stride * 高stride
	RGB888				
	BGR888				

版本	输出图片格式	输出图片宽、高对齐要求	输出图片宽stride、高stride、内存大小要求 Atlas 200/300/500推理产品 Atlas 训练系列产品	输出图片宽stride、高stride、内存大小要求 Atlas 推理系列产品	输出图片宽stride、高stride、内存大小要求 Atlas 200I/500 A2推理产品 Atlas A2训练系列产品/Atlas 800I A2推理产品
	ARGB8888	无对齐要求		宽stride为宽16对齐后、再乘以4的值。 高stride无对齐要求，与高相同即可。 内存大小（单位Byte）= 宽stride * 高stride	宽stride为宽乘以4的值；高stride无对齐要求，与高相同即可。 内存大小（单位Byte）= 宽stride * 高stride
	ABGR8888				
	RGBA8888				
	BGRA8888				
该部分格式仅以下版本支持： Atlas 200I/500 A2推理产品 Atlas A2训练系列产品/ Atlas 800I A2推理产品	YVU444 Packed	无对齐要求	-	-	宽stride为宽乘以3的值；高stride无对齐要求，与高相同即可。 内存大小（单位Byte）= 宽stride * 高stride
	RGB888 Planar BGR888 Planar	无对齐要求	-	-	宽stride无对齐要求，与宽相同即可；高stride无对齐要求，与高相同即可。 内存大小（单位Byte）= 宽stride * 高stride * 3

版本	输出图片格式	输出图片宽、高对齐要求	输出图片宽stride、高stride、内存大小要求 Atlas 200/300/500推理产品 Atlas 训练系列产品	输出图片宽stride、高stride、内存大小要求 Atlas 推理系列产品	输出图片宽stride、高stride、内存大小要求 Atlas 200I/500A2推理产品 Atlas A2训练系列产品/Atlas 800I A2推理产品
该部分格式仅以下版本支持： Atlas A2训练系列产品/ Atlas 800I A2推理产品	YUV400 FP32 RGB888 FP32 BGR888 FP32	无对齐要求	-	-	宽stride无对齐要求，宽stride为宽乘以3的值；高stride无对齐要求，与高相同即可。 内存大小（单位Byte）= 宽stride * 高stride
	RGB888 Planar FP32 BGR888 Planar FP32	无对齐要求			宽stride无对齐要求，与宽相同即可；高stride无对齐要求，与高相同即可。 内存大小（单位Byte）= 宽stride * 高stride * 3

抠图、贴图约束

产品版本	抠图、贴图约束说明
Atlas 200/300/500 推理产品	• 抠图区域不超出输入图片区域。 • 贴图区域不超出输出图片区域，最大贴图个数256个。 • 贴图区域相对输出图片的左偏移16对齐。 - 贴图时可直接放置在输出图片的最左侧，即相对输出图片的左偏移为0，0也满足16对齐的要求。 - 输出图片的贴图宽度建议16对齐，如果不是16对齐，会多写一段无效数据使其16对齐。见图2-4、图2-5，贴图区域旁边的绿色框就表示无效数据。 - 在等比例缩放场景下，由于贴图左偏移16对齐的限制，如果直接调用acl.media.dvpp_vpc_crop_resize_paste_async接口实现缩放功能，贴图的位置可能不在输出图片的正中心位置，此时可通过抠图、缩放、填充几个功能配合使用（调用acl.media.dvpp_vpc_batch_crop_resize_make_border_async接口），实现等比例缩放。 • 抠图、贴图区域的奇数、偶数限制： 仅输出图片格式为YUV420SP，贴图区域奇数、偶数限制为：左偏移和上偏移为偶数、右偏移和下偏移为奇数。 • 缩放场景下，贴图/抠图的宽或高缩放比例范围： [1/32, 16]。 • 推理场景下，VPC的输出作为模型推理的输入时，由于贴图区域的宽 * 高有16 * 2对齐的约束，若贴图有效区域的宽 * 高不满足16 * 2对齐时，则贴图区域中会存在一些补边的无效数据，为保证推理精度，贴图有效区域的宽 * 高不满足16 * 2对齐时，用户需借助AIPP抠出有效区域，去除无效数据。

产品版本	抠图、贴图约束说明
Atlas 训练系列产品	● 抠图区域不超出输入图片区域。 ● 贴图区域不超出输出图片区域，最大贴图个数256个。 ● 贴图区域相对输出图片的左偏移16对齐。 - 贴图时可直接放置在输出图片的最左侧，即相对输出图片的左偏移为0，0也满足16对齐的要求。 - 输出图片的贴图宽度建议16对齐，如果不是16对齐，会多写一段无效数据使其16对齐。见图2-4、图2-5，贴图区域旁边的绿色框就表示无效数据。 - 在等比例缩放场景下，由于贴图左偏移16对齐的限制，如果直接调用acl.media.dvpp_vpc_crop_resize_paste_async接口实现缩放功能，贴图的位置可能不在输出图片的正中心位置，此时可通过抠图、缩放、填充几个功能配合使用（调用acl.media.dvpp_vpc_batch_crop_resize_make_border_async接口），实现等比例缩放。 ● 抠图、贴图区域的奇数、偶数限制： 仅输出图片格式为YUV420SP，贴图区域奇数、偶数限制为：左偏移和上偏移为偶数、右偏移和下偏移为奇数。 ● 缩放场景下，贴图/抠图的宽或高缩放比例范围： [1/32, 16]。 ● 推理场景下，VPC的输出作为模型推理的输入时，由于贴图区域的宽 * 高有16 * 2对齐的约束，若贴图有效区域的宽 * 高不满足16 * 2对齐时，则贴图区域中会存在一些补边的无效数据，为保证推理精度，贴图有效区域的宽 * 高不满足16 * 2对齐时，用户需借助AIPP抠出有效区域，去除无效数据。

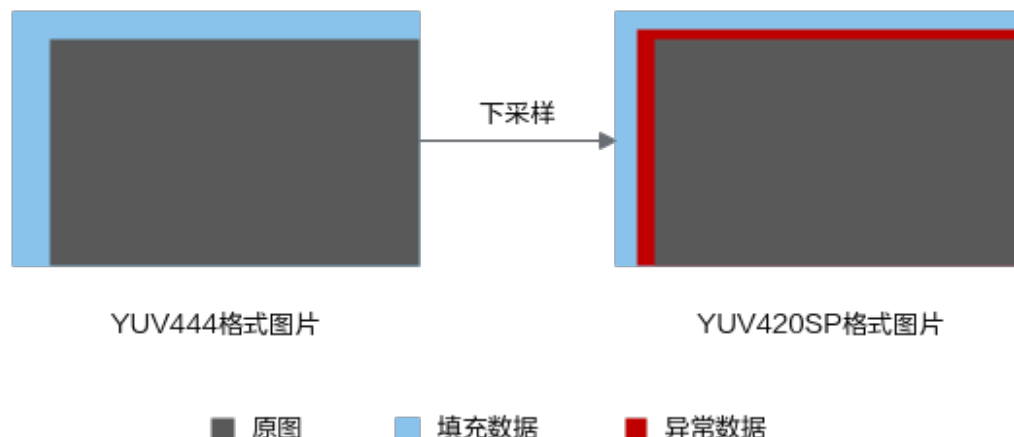
产品版本	抠图、贴图约束说明
Atlas 推理系列产品	• 抠图区域不超出输入图片区域。 • 贴图区域不超出输出图片区域，最大贴图个数256个。 • 贴图区域相对输出图片的左偏移16对齐。 - 贴图时可直接放置在输出图片的最左侧，即相对输出图片的左偏移为0，0也满足16对齐的要求。 - 输出图片的贴图宽度建议16对齐，如果不是16对齐，会多写一段无效数据使其16对齐。见图2-4、图2-5，贴图区域旁边的绿色框就表示无效数据。 - 在等比例缩放场景下，由于贴图左偏移16对齐的限制，如果直接调用acl.media.dvpp_vpc_crop_resize_paste_async接口实现缩放功能，贴图的位置可能不在输出图片的正中心位置，此时可通过抠图、缩放、填充几个功能配合使用（调用acl.media.dvpp_vpc_batch_crop_resize_make_border_async接口），实现等比例缩放。 • 抠图、贴图区域的奇数、偶数限制： 输出图片格式为YUV420SP，贴图区域奇数、偶数限制为：左偏移和上偏移为偶数、右偏移和下偏移为奇数。 输出图片格式为YUV422SP，贴图区域奇数、偶数限制为：左偏移为偶数、右偏移为奇数。 其它格式，贴图区域没有偏移奇偶数的限制。
Atlas 200I/500 A2推理产品	• 抠图区域不超出输入图片区域。 • 贴图区域不超出输出图片区域，最大贴图个数256个。 • 抠图、贴图区域的奇数、偶数限制： 抠图起始坐标无奇数、偶数限制。 当输出图片格式为YUV420时，贴图区奇数、偶数限制为：左偏移和上偏移为偶数、右偏移和下偏移为奇数。 当输出图片格式为YUV422时，贴图区奇数、偶数限制为：左偏移为偶数、右偏移为奇数。
Atlas A2训练系列产品/ Atlas 800I A2推理产品	• 抠图区域不超出输入图片区域。 • 贴图区域不超出输出图片区域，最大贴图个数256个。 • 抠图、贴图区域的奇数、偶数限制： 抠图起始坐标无奇数、偶数限制。 输出图片格式为YUV420时，贴图区奇数、偶数限制为：左偏移和上偏移为偶数、右偏移和下偏移为奇数。 输出图片格式为YUV422时，贴图区奇数、偶数限制为：左偏移为偶数、右偏移为奇数。

YUV 格式图像下采样约束

VPC在处理图片时，会根据输入或输出图片格式，将输入图片格式转换为YUV444或RGB用于内部处理，YUV444或RGB没有宽高奇偶数的限制，但当输出图片格式为

YUV420SP或YUV422SP格式时，会进行下采样处理，由于YUV420SP或YUV422SP格式本身的数据排布导致宽高存在奇偶数限制，因此输出图片的边缘可能存在异常数据。

图 2-9 异常效果图片举例



出现异常数据的根因在于，在计算过程中，出现输出图片位置处于奇数起始点时，此时YUV444格式的图片是正确的，但是下采样到YUV420SP格式时，由于奇数行和偶数行是共用同一个UV，导致图片起始行的Y与上一行的UV组成新的像素，产生异常数据。

2.9.6.3 性能指标说明

本节仅描述各产品对应的昇腾AI处理器的性能指标参考数据，各产品由于其内存、CPU等配置不同，在性能规格上会存在差异，详细产品的性能规格请查阅[昇腾社区](#)的官方说明。

本节以单个昇腾AI处理器为例，描述基本场景性能指标参考，其它分辨率可以等量估算。

性能指标说明（Atlas 200/300/500 推理产品）

以下性能数据，是基于一个Stream上下发10个异步媒体数据处理任务后，执行一次[acl.rt.synchronize_stream](#)接口。

- 对于非8K图像，单个Device的基本场景性能指标参考如下（1路对应一个通道，一个通道对应一个线程）：
对于1080p的图像，若存在Host -> Device的图片数据拷贝，由于拷贝带宽限制，最大总帧率约为1000fps。
对于4K的图像，若存在Host -> Device的图片数据拷贝，由于拷贝带宽限制，最大总帧率约为250fps。

表 2-14 Atlas 200/300/500 推理产品的昇腾 AI 处理器性能指标参考数据

场景举例	总帧率
• 输入图像分辨率：1080p （ 1920 * 1080 ） • 输出图像分辨率：1080p （ 1920 * 1080 ） • 输入/输出图片格式： YUV420SP • n路（ n < 4 ）	n * 360fps
• 输入图像分辨率：1080p （ 1920 * 1080 ） • 输出图像分辨率：1080p （ 1920 * 1080 ） • 输入/输出图片格式： YUV420SP • n路（ n ≥ 4 ）	1440fps
• 输入图像分辨率：4K图像 （ 3840 * 2160 ） • 输出图像分辨率：4K图像 （ 3840 * 2160 ） • 输入/输出图片格式： YUV420SP • n路（ n < 4 ）	n * 90fps
• 输入图像分辨率：4K图像 （ 3840 * 2160 ） • 输出图像分辨率：4K图像 （ 3840 * 2160 ） • 输入/输出图片格式： YUV420SP • n路（ n ≥ 4 ）	360fps

- 对于8K图像，单个Device的典型场景性能指标参考如下（ 1路对应一个通道，一个通道对应一个线程 ）：

表 2-15 Atlas 200/300/500 推理产品的昇腾 AI 处理器性能指标参考数据

场景举例	总帧率
• 输入图像分辨率：8K图像（ 7680 * 4320 ） • 输出图像分辨率：1080p（ 1920 * 1080 ） • 输入/输出图片格式：YUV420SP • n路（ n < 4 ）	n * 25fps
• 输入图像分辨率：8K图像（ 7680 * 4320 ） • 输出图像分辨率：1080p（ 1920 * 1080 ） • 输入/输出图片格式：YUV420SP • n路（ n ≥ 4 ）	100fps
• 输入图像分辨率：8K图像（ 7680 * 4320 ） • 输出图像分辨率：4K图像（ 3840 * 2160 ） • 输入/输出图片格式：YUV420SP • n路（ n < 4 ）	n * 25fps
• 输入图像分辨率：8K图像（ 7680 * 4320 ） • 输出图像分辨率：4K图像（ 3840 * 2160 ） • 输入/输出图片格式：YUV420SP • n路（ n ≥ 4 ）	100fps

📖 说明

- VPC处理性能与处理过程中的图像分辨率强相关，以输入图像和输出图像中的最大分辨率作为基准分辨率，基准分辨率越大，处理耗时越久，性能越低。
- 调用VPC批处理接口（接口命名中包含Batch，例如[acl.media.dvpp_vpc_batch_crop_async](#)接口）时，由于图像处理单元DVPP（Digital Video Pre-Processor）内部多个VPC硬件单元会并行处理图片任务，因此单路就可以达到最大总帧率。

性能指标说明（Atlas 推理系列产品）

以下性能数据，是基于一个Stream上下发64个异步媒体数据处理任务后，执行一次[acl.rt.synchronize_stream](#)接口，且Host服务器配置为：CPU主频为Intel(R) Xeon(R) CPU E5-2680 v4 @ 2.40GHz，CPU核心数为14。

单个Device的基本场景性能指标参考如下（1路对应一个通道，一个通道对应一个线程）：

表 2-16 Atlas 推理系列产品的昇腾 AI 处理器性能指标参考数据

场景举例	总帧率
- 输入图片分辨率：1080p（1920 * 1080） - 输出图片分辨率：1080p（1920 * 1080） - 输入/输出图片格式：YUV420SP - n路（1 ≤ n ≤ 12）	n * 360fps
- 输入图片分辨率：1080p（1920 * 1080） - 输出图片分辨率：1080p（1920 * 1080） - 输入/输出图片格式：YUV420SP - n路（n > 12）	4320fps
- 输入图片分辨率：4K图像（3840 * 2160） - 输出图片分辨率：4K图像（3840 * 2160） - 输入/输出图片格式：YUV420SP - n路（1 ≤ n ≤ 12）	n * 90fps
- 输入图片分辨率：4K图像（3840 * 2160） - 输出图片分辨率：4K图像（3840 * 2160） - 输入/输出图片格式：YUV420SP - n路（n > 12）	1080fps

场景举例	总帧率
• 输入图片分辨率：8K图像 (7680 * 4320) • 输出图片分辨率：4K图像 (3840 * 2160) • 输入/输出图片格式：YUV420SP • n路 ($1 \leq n \leq 12$)	n * 22.5fps
• 输入图片分辨率：8K图像 (7680 * 4320) • 输出图片分辨率：4K图像 (3840 * 2160) • 输入/输出图片格式：YUV420SP • n路 ($n > 12$)	270fps

 说明

调用VPC批处理接口（接口命名中包含Batch，例如[acl.media.dvpp_vpc_batch_crop_async](#)接口）时，由于图像处理单元DVPP（Digital Video Pre-Processor）内部多个VPC硬件单元会并行处理图片任务，因此单路就可以达到最大总帧率。

性能指标说明（Atlas 训练系列产品）

以下性能数据，是基于一个Stream上下发10个异步媒体数据处理任务后，执行一次[acl.rt.synchronize_stream](#)接口。

- 对于非8K图像，单个Device的基本场景性能指标参考如下（1路对应一个通道，一个通道对应一个线程）：

表 2-17 Atlas 训练系列产品的昇腾 AI 处理器性能指标参考数据

场景举例	总帧率
• 输入图像分辨率：1080p (1920 * 1080) • 输出图像分辨率：1080p (1920 * 1080) • 输入/输出图片格式：YUV420SP • n路 ($n < 16$)	n * 360fps

场景举例	总帧率
• 输入图像分辨率：1080p (1920 * 1080) • 输出图像分辨率：1080p (1920 * 1080) • 输入/输出图片格式：YUV420SP • n路 (n ≥ 16)	5760fps
• 输入图像分辨率：4K图像 (3840 * 2160) • 输出图像分辨率：4K图像 (3840 * 2160) • 输入/输出图片格式：YUV420SP • n路 (n < 16)	n * 90fps
• 输入图像分辨率：4K图像 (3840 * 2160) • 输出图像分辨率：4K图像 (3840 * 2160) • 输入/输出图片格式：YUV420SP • n路 (n ≥ 16)	1440fps

- 对于8K图像，单个Device的典型场景性能指标参考如下（1路对应一个通道，一个通道对应一个线程）：

表 2-18 Atlas 训练系列产品的昇腾 AI 处理器性能指标参考数据

场景举例	总帧率
• 输入图像分辨率：8K图像 (7680 * 4320) • 输出图像分辨率：1080p (1920 * 1080) • 输入/输出图片格式：YUV420SP • n路 (n < 16)	n * 25fps

场景举例	总帧率
• 输入图像分辨率：8K图像（ 7680 * 4320 ） • 输出图像分辨率：1080p（ 1920 * 1080 ） • 输入/输出图片格式：YUV420SP • n路（ n ≥ 16 ）	400fps
• 输入图像分辨率：8K图像（ 7680 * 4320 ） • 输出图像分辨率：4K图像（ 3840 * 2160 ） • 输入/输出图片格式：YUV420SP • n路（ n < 16 ）	n * 25fps
• 输入图像分辨率：8K图像（ 7680 * 4320 ） • 输出图像分辨率：4K图像（ 3840 * 2160 ） • 输入/输出图片格式：YUV420SP • n路（ n ≥ 16 ）	400fps

 说明

调用VPC批处理接口（接口命名中包含Batch，例如[acl.media.dvpp_vpc_batch_crop_async](#)接口）时，由于图像处理单元DVPP（Digital Video Pre-Processor）内部多个VPC硬件单元会并行处理图片任务，因此单路就可以达到最大总帧率。

性能指标说明（Atlas 200I/500 A2 推理产品）

单个Device场景下的性能指标参考如下（1路对应一个通道，一个通道对应一个线程，或者n路对应一个通道，一个通道对应n个线程）：

表 2-19 Atlas 200I/500 A2 推理产品的昇腾 AI 处理器性能指标参考数据

场景举例	总帧率
• 输入图片分辨率：1080p (1920 * 1080) • 输出图片分辨率：1080p (1920 * 1080) • 输入/输出图片格式：YUV420SP • n路 (n < 2)	n * 800fps
• 输入图片分辨率：1080p (1920 * 1080) • 输出图片分辨率：1080p (1920 * 1080) • 输入/输出图片格式：YUV420SP • n路 (n ≥ 2)	1600fps
• 输入图片分辨率：4K图像 (3840 * 2160) • 输出图片分辨率：4K图像 (3840 * 2160) • 输入/输出图片格式：YUV420SP • n路 (n < 2)	n * 200fps
• 输入图片分辨率：4K图像 (3840 * 2160) • 输出图片分辨率：4K图像 (3840 * 2160) • 输入/输出图片格式：YUV420SP • n路 (n ≥ 2)	400fps
• 输入图片分辨率：8K图像 (7680 * 4320) • 输出图片分辨率：4K图像 (3840 * 2160) • 输入/输出图片格式：YUV420SP • n路 (n < 2)	n * 100fps

场景举例	总帧率
- 输入图片分辨率：8K图像（7680 * 4320） - 输出图片分辨率：4K图像（3840 * 2160） - 输入/输出图片格式：YUV420SP - n路（$n \geq 2$）	200fps

说明

调用VPC批处理接口（接口命名中包含batch，例如[acl.media.dvpp_vpc_batch_crop_async](#)接口）时，由于图像处理单元DVPP（Digital Video Pre-Processing）内部多个VPC硬件单元会并行处理图片任务，因此单路就可以达到最大总帧率。

性能指标说明（Atlas A2 训练系列产品/Atlas 800I A2 推理产品）

表 2-20 Atlas A2 训练系列产品/Atlas 800I A2 推理产品的昇腾 AI 处理器性能指标参考数据

场景举例	总帧率
- 输入图片分辨率：1080p（1920 * 1080） - 输出图片分辨率：1080p（1920 * 1080） - 输入/输出图片格式：YUV420SP - n路（$1 \leq n \leq 10$）	$n * 1440\text{fps}$
- 输入图片分辨率：1080p（1920 * 1080） - 输出图片分辨率：1080p（1920 * 1080） - 输入/输出图片格式：YUV420SP - n路（$n > 10$）	14400fps
- 输入图片分辨率：4K图像（3840 * 2160） - 输出图片分辨率：4K图像（3840 * 2160） - 输入/输出图片格式：YUV420SP - n路（$1 \leq n \leq 10$）	$n * 360\text{fps}$

场景举例	总帧率
• 输入图片分辨率：4K图像（3840 * 2160） • 输出图片分辨率：4K图像（3840 * 2160） • 输入/输出图片格式：YUV420SP • n路（n > 10）	3600fps
• 输入图片分辨率：8K图像（7680 * 4320） • 输出图片分辨率：4K图像（3840 * 2160） • 输入/输出图片格式：YUV420SP • n路（1 ≤ n ≤ 10）	n * 90fps
• 输入图片分辨率：8K图像（7680 * 4320） • 输出图片分辨率：4K图像（3840 * 2160） • 输入/输出图片格式：YUV420SP • n路（n > 10）	900fps

2.9.6.4 函数：dvpp_vpc_resize_async

C函数原型	<code>aclError acldvppVpcResizeAsync(aclDvppChannelDesc *channelDesc, aclDvppPicDesc *inputDesc, aclDvppPicDesc *outputDesc, aclDvppResizeConfig *resizeConfig, aclrtStream stream)</code>
Python函数	<code>ret = acl.media.dvpp_vpc_resize_async(dvpp_channel_desc, input_dvpp_pic_desc, output_dvpp_pic_desc, dvpp_resize_config, stream)</code>
函数功能	将输入图片缩放到输出图片大小，异步接口。

输入说明	dvpp_channel_desc: int, 指定通道描述信息的指针地址, 与调用acl.media.dvpp_create_channel接口创建通道时指定的“dvpp_channel_desc”保持一致。 input_dvpp_pic_desc: int, 输入图片信息的指针地址。 - 调用acl.media.dvpp_create_pic_desc接口创建图片描述信息。 - 调用acl.media.dvpp_set_pic_desc系列接口设置图片描述参数（例如，图片格式、宽、高等）。 output_dvpp_pic_desc: int, 输出图片信息的指针地址。 output_dvpp_pic_desc参数作为输入时，需要用户调用如下接口： - 调用acl.media.dvpp_create_pic_desc接口创建图片描述信息。 - 调用acl.dvpp.set_pic_desc系列接口设置图片描述参数（例如，内存地址、内存大小、图片格式、宽、高等）。 dvpp_resize_config: int, 指定图片缩放配置数据的指针地址。 调用acl.media.dvpp_create_resize_config接口创建图片缩放配置数据。 stream: int, 指定Stream对象。
返回值说明	ret: int, 错误码。 - 返回0表示任务下发成功。 - 返回非0表示任务下发失败。

约束说明

版本	约束
Atlas 200/300/500 推理产品 Atlas 训练系列产品	- 关于输入图片分辨率、图片格式： - 输入图片分辨率在10 * 6 ~ 4096 * 4096（包括4096）范围内时，支持图片格式、宽高对齐、内存约束处说明的输入图片格式。 - 输入图片的宽或高在4096 ~ 8192（包括8192）范围内时，支持格式如下： 支持aclDvppPixelFormat枚举值中的如下枚举项： PIXEL_FORMAT_YUV_SEMIPLANAR_420 = 1, // YUV420SP NV12 8bit PIXEL_FORMAT_YVU_SEMIPLANAR_420 = 2, // YUV420SP NV21 8bit - 如果输入图片的宽高不是2对齐，VPC会自动处理成向下2对齐，例如输入图片的宽高为15 * 15，VPC会先处理成14 * 14，再进行缩放。 - 关于输出图片分辨率、图片格式请参见图片格式、宽高对齐、内存约束。 - 关于各图片格式的宽高对齐、内存大小约束，请参见图片格式、宽高对齐、内存约束。

版本	约束
Atlas 推理系列产品	- 关于输入图片分辨率、图片格式： - 输入图片分辨率在$10 * 6 \sim 4096 * 4096$（包括4096）范围内时，支持图片格式、宽高对齐、内存约束处说明的输入图片格式。 - 输入图片的宽或高在4096 ~ 8192（包括8192）范围内时，支持格式如下： 支持aclDvppPixelFormat枚举值中的如下枚举项： <pre>PIXEL_FORMAT_YUV_400 = 0, // YUV400 8bit PIXEL_FORMAT_YUV_SEMIPLANAR_420 = 1, // YUV420SP NV12 8bit PIXEL_FORMAT_YVU_SEMIPLANAR_420 = 2, // YUV420SP NV21 8bit PIXEL_FORMAT_YUV_SEMIPLANAR_422 = 3, // YUV422SP 8bit PIXEL_FORMAT_YVU_SEMIPLANAR_422 = 4, // YVU422SP 8bit PIXEL_FORMAT_YUV_SEMIPLANAR_444 = 5, // YUV444SP 8bit PIXEL_FORMAT_YVU_SEMIPLANAR_444 = 6, // YVU444SP 8bit PIXEL_FORMAT_YUYV_PACKED_422 = 7, // YUV422Packed YUYV 8bit PIXEL_FORMAT_UYVY_PACKED_422 = 8, // YUV422Packed UYVY 8bit PIXEL_FORMAT_VYVU_PACKED_422 = 9, // YUV422Packed VYVU 8bit PIXEL_FORMAT_VYUY_PACKED_422 = 10, // YUV422Packed VYUY 8bit</pre> - 如果输入图片的宽高不是2对齐，VPC会自动处理成向下2对齐，例如输入图片的宽高为$15 * 15$，VPC会先处理成$14 * 14$，再进行缩放。 - 关于输出图片分辨率、图片格式请参见图片格式、宽高对齐、内存约束。 - 关于各图片格式的宽高对齐、内存大小约束，请参见图片格式、宽高对齐、内存约束。 - 当输入、输出图片格式都为ARGB8888、ABGR8888、RGBA8888、BGRA8888格式时，在经过本接口缩放处理后，A分量值会变为0。
Atlas A2训练系列产品/ Atlas 800I A2推理产品 Atlas 200I/500 A2推理产品	- 关于输入、输出图片分辨率： - 输入图片分辨率：$10 * 6 \sim 8192 * 8192$ - 输出图片分辨率：$10 * 6 \sim 4096 * 8192$ - 关于图片格式、宽高对齐、内存等总体约束，请参见图片格式、宽高对齐、内存约束。 - 当输入、输出图片格式都为ARGB8888、ABGR8888、RGBA8888、BGRA8888格式时，在经过本接口缩放处理后，A分量值会变为0。

2.9.6.5 函数：dvpp_vpc_crop_async

C函数原型	<code>aclError aclDvppVpcCropAsync(aclDvppChannelDesc *channelDesc, aclDvppPicDesc *inputDesc, aclDvppPicDesc *outputDesc, aclDvppRoiConfig *cropArea, aclrtStream stream)</code>
Python函数	<code>ret = acl.media.dvpp_vpc_crop_async(dvpp_channel_desc, input_dvpp_pic_desc, output_dvpp_pic_desc, crop_area, stream)</code>
函数功能	按指定区域从输入图片中抠图，再将抠的图片存放到输出内存中，作为输出图片，异步接口。
输入说明	dvpp_channel_desc: int, 指定通道描述信息的指针地址，与调用acl.media.dvpp_create_channel接口创建通道时指定的“dvpp_channel_desc”保持一致。 input_dvpp_pic_desc: int, 输入图片信息的指针地址。 - 调用acl.media.dvpp_create_pic_desc接口创建图片描述信息。 - 调用acl.media.dvpp_set_pic_desc系列接口设置图片描述参数（例如，图片格式、宽、高等）。 output_dvpp_pic_desc: int, 输出图片信息的指针地址。 outputDesc参数作为输入时，需要用户调用如下接口： - 调用acl.media.dvpp_create_pic_desc接口创建图片描述信息。 - 调用acl.media.dvpp_set_pic_desc系列接口设置图片描述参数（例如，内存地址、内存大小、图片格式、宽、高等）。 crop_area: int, 抠图区域位置的指针地址。调用acl.media.dvpp_create_roi_config接口创建区域位置数据。 stream: int, 指定Stream对象。
返回值说明	ret: int, 错误码。 - 返回0，表示任务下发成功。 - 返回非0，表示任务下发失败。

约束说明

版本	约束
Atlas 200/300/500 推理产品 Atlas 训练系列产品	- 关于输入图片分辨率、图片格式： - 输入图片分辨率在10 * 6 ~ 4096 * 4096（包括4096）范围内时，支持图片格式、宽高对齐、内存约束处说明的输入图片格式。 - 输入图片的宽或高在4096 ~ 8192（包括8192）范围内时，支持格式如下： 支持aclDvppPixelFormat枚举值中的如下枚举项： PIXEL_FORMAT_YUV_SEMIPLANAR_420 = 1, // YUV420SP NV12 8bit PIXEL_FORMAT_YVU_SEMIPLANAR_420 = 2, // YUV420SP NV21 8bit - 关于输出图片分辨率、图片格式请参见图片格式、宽高对齐、内存约束。 - 关于各图片格式的宽高对齐、内存大小约束，请参见图片格式、宽高对齐、内存约束。 - 输出图片区域与“crop_area”不一致时会对图片再做一次缩放操作，默认缩放算法为“华为自研的高阶滤波算法”。 - 由于通过本接口抠的图直接作为输出图片，所以需满足输出图片最大分辨率4096 * 4096的限制，因此本接口中抠图区域最大分辨率为4096 * 4096。

版本	约束
Atlas 推理系列产品	- 关于输入图片分辨率、图片格式： - 输入图片分辨率在$10 * 6 \sim 4096 * 4096$（包括4096）范围内时，支持图片格式、宽高对齐、内存约束处说明的输入图片格式。 - 输入图片的宽或高在4096 ~ 8192（包括8192）范围内时，支持格式如下： 支持aclDvppPixelFormat枚举值中的如下枚举项： <pre> PIXEL_FORMAT_YUV_400 = 0, // YUV400 8bit PIXEL_FORMAT_YUV_SEMIPLANAR_420 = 1, // YUV420SP NV12 8bit PIXEL_FORMAT_YVU_SEMIPLANAR_420 = 2, // YUV420SP NV21 8bit PIXEL_FORMAT_YUV_SEMIPLANAR_422 = 3, // YUV422SP 8bit PIXEL_FORMAT_YVU_SEMIPLANAR_422 = 4, // YVU422SP 8bit PIXEL_FORMAT_YUV_SEMIPLANAR_444 = 5, // YUV444SP 8bit PIXEL_FORMAT_YVU_SEMIPLANAR_444 = 6, // YVU444SP 8bit PIXEL_FORMAT_YUYV_PACKED_422 = 7, // YUV422Packed YUYV 8bit PIXEL_FORMAT_UYVY_PACKED_422 = 8, // YUV422Packed UYVY 8bit PIXEL_FORMAT_VYU_PACKED_422 = 9, // YUV422Packed VYU 8bit PIXEL_FORMAT_VYUY_PACKED_422 = 10, // YUV422Packed VYUY 8bit </pre> - 关于输出图片分辨率、图片格式请参见图片格式、宽高对齐、内存约束。 - 关于各图片格式的宽高对齐、内存大小约束，请参见图片格式、宽高对齐、内存约束。 - 输出图片区域与“crop_area”不一致时会对图片再做一次缩放操作，默认缩放算法为“业界通用的Bilinear算法”。 - 由于通过本接口抠的图直接作为输出图片，所以需满足输出图片最大分辨率$4096 * 4096$的限制，因此本接口中抠图区域最大分辨率为$4096 * 4096$。
Atlas A2训练系列产品/ Atlas 800I A2推理产品 Atlas 200I/500 A2推理产品	- 关于输入、输出图片分辨率： - 输入图片分辨率：$10 * 6 \sim 8192 * 8192$ - 输出图片分辨率：$10 * 6 \sim 4096 * 8192$ - 关于图片格式、宽高对齐、内存等总体约束，请参见图片格式、宽高对齐、内存约束。 - 输出图片区域与“crop_area”不一致时会对图片再做一次缩放操作，默认缩放算法为“业界通用的Bilinear算法”。 - 由于通过本接口抠的图直接作为输出图片，所以需满足输出图片最大分辨率$4096 * 8192$的限制，因此本接口中抠图区域最大分辨率为$4096 * 8192$。

2.9.6.6 函数：dvpp_vpc_crop_resize_async

C函数原型	<code>aclError aclDvppVpcCropResizeAsync(aclDvppChannelDesc *channelDesc, aclDvppPicDesc *inputDesc, aclDvppPicDesc *outputDesc, aclDvppRoiConfig *cropArea, aclDvppResizeConfig *resizeConfig, aclrtStream stream)</code>
Python函数	<code>ret = acl.media.dvpp_vpc_crop_resize_async(dvpp_channel_desc, input_dvpp_pic_desc, output_dvpp_pic_desc, crop_area, resize_config, stream)</code>
函数功能	按指定区域从一张输入图片中抠出一张子图，再将抠的图片存放到输出内存中，作为输出图片，异步接口。 输出图片区域与“crop_area”不一致时会再对图片做一次缩放操作，按照“resize_config”处配置的缩放算法。
输入说明	dvpp_channel_desc: int, 指定通道描述信息的指针地址，与调用acl.media.dvpp_create_channel接口创建通道时指定的“dvpp_channel_desc”保持一致。 input_dvpp_pic_desc: int, 输入图片信息的指针地址。 - 调用acl.media.dvpp_create_pic_desc接口创建图片描述信息。 - 调用acl.media.dvpp_set_pic_desc系列接口设置图片描述参数（例如，内存地址、内存大小、图片格式、宽、高等）。 output_dvpp_pic_desc: int, 输出图片信息的指针地址。 - 调用acl.media.dvpp_create_pic_desc接口创建图片描述信息。 - 调用acl.media.dvpp_set_pic_desc系列接口设置图片描述参数（例如，内存地址、内存大小、图片格式、宽、高等）。 crop_area: int, 抠图区域位置的指针地址。 调用acl.media.dvpp_create_roi_config接口创建区域位置数据。 resize_config: int, 指定图片缩放配置数据的指针地址。 需提前调用acl.media.dvpp_create_resize_config接口创建图片缩放配置数据。 stream: int, 指定Stream对象。
返回值说明	ret: int, 错误码。 - 返回0表示任务下发成功。 - 返回非0表示任务下发失败。
参考资源	接口调用示例，参见抠图缩放（一图一框）示例代码。

约束说明

版本	约束
Atlas 200/300/500 推理产品 Atlas 训练系列产品	- 关于输入图片分辨率、图片格式： - 输入图片分辨率在$10 * 6 \sim 4096 * 4096$（包括4096）范围内时，支持图片格式、宽高对齐、内存约束处说明的输入图片格式。 - 输入图片的宽或高在$4096 \sim 8192$（包括8192）范围内时，支持格式如下： 支持acldvppPixelFormat枚举值中的如下枚举项： PIXEL_FORMAT_YUV_SEMIPLANAR_420 = 1, // YUV420SP NV12 8bit PIXEL_FORMAT_YVU_SEMIPLANAR_420 = 2, // YUV420SP NV21 8bit - 如果输入图片的宽高不是2对齐，VPC会自动处理成向下2对齐，例如输入图片的宽高为$15 * 15$，VPC会先处理成$14 * 14$，再进行缩放。 - 关于输出图片分辨率、图片格式请参见图片格式、宽高对齐、内存约束。 - 关于各图片格式的宽高对齐、内存大小约束，请参见图片格式、宽高对齐、内存约束。
Atlas 推理系列产品	- 关于输入图片分辨率、图片格式： - 输入图片分辨率在$10 * 6 \sim 4096 * 4096$（包括4096）范围内时，支持图片格式、宽高对齐、内存约束处说明的输入图片格式。 - 输入图片的宽或高在$4096 \sim 8192$（包括8192）范围内时，支持格式如下： 支持acldvppPixelFormat枚举值中的如下枚举项： PIXEL_FORMAT_YUV_400 = 0, // YUV400 8bit PIXEL_FORMAT_YUV_SEMIPLANAR_420 = 1, // YUV420SP NV12 8bit PIXEL_FORMAT_YVU_SEMIPLANAR_420 = 2, // YUV420SP NV21 8bit PIXEL_FORMAT_YUV_SEMIPLANAR_422 = 3, // YUV422SP 8bit PIXEL_FORMAT_YVU_SEMIPLANAR_422 = 4, // YVU422SP 8bit PIXEL_FORMAT_YUV_SEMIPLANAR_444 = 5, // YUV444SP 8bit PIXEL_FORMAT_YVU_SEMIPLANAR_444 = 6, // YVU444SP 8bit PIXEL_FORMAT_YUYV_PACKED_422 = 7, // YUV422Packed YUYV 8bit PIXEL_FORMAT_UYVY_PACKED_422 = 8, // YUV422Packed UYVY 8bit PIXEL_FORMAT_YVYU_PACKED_422 = 9, // YUV422Packed YVYU 8bit PIXEL_FORMAT_VYUY_PACKED_422 = 10, // YUV422Packed VYUY 8bit - 关于输出图片分辨率、图片格式请参见图片格式、宽高对齐、内存约束。 - 关于各图片格式的宽高对齐、内存大小约束，请参见图片格式、宽高对齐、内存约束。 - 当输入、输出图片格式都为ARGB8888、ABGR8888、RGBA8888、BGRA8888格式时，在经过本接口缩放处理后，A分量值会变为0。

版本	约束
Atlas A2训练系列产品/ Atlas 800I A2推理产品 Atlas 200I/500 A2推理 产品	- 关于输入、输出图片分辨率： - 输入图片分辨率：10 * 6 ~ 8192 * 8192 - 输出图片分辨率：10 * 6 ~ 4096 * 8192 - 关于图片格式、宽高对齐、内存等总体约束，请参见图片格式、宽高对齐、内存约束。 - 当输入、输出图片格式都为ARGB8888、ABGR8888、RGBA8888、BGRA8888格式时，在经过本接口缩放处理后，A分量值会变为0。

2.9.6.7 函数：dvpp_vpc_batch_crop_async

C函数原型	<code>aclError aclDvppVpcBatchCropAsync(aclDvppChannelDesc *channelDesc, aclDvppBatchPicDesc *srcBatchPicDescs, uint32_t *roiNums, uint32_t size, aclDvppBatchPicDesc *dstBatchPicDescs, aclDvppRoiConfig *cropAreas[], aclrtStream stream)</code>
Python函数	<code>dst_batch_pic_descs, ret = acl.media.dvpp_vpc_batch_crop_async(dvpp_channel_desc, src_batch_pic_descs, roi_nums_list, dst_batch_pic_descs, crop_area_list, stream)</code>
函数功能	对于一张或多张输入图片，支持从每个输入图片中抠出一张或多张子图并一一对应存放到多个输出区域，输出图片区域与抠图区域不一致时会对图片再做一次缩放操作。异步接口。

输入说明	dvpp_channel_desc: int, 指定通道描述信息的指针地址, 与调用acl.media.dvpp_create_channel接口创建通道时指定的“dvpp_channel_desc”保持一致。 src_batch_pic_descs: int, 指定批量输入图片描述信息的指针地址。 - 调用acl.media.dvpp_create_batch_pic_desc接口创建批量图片描述信息。 - 调用acl.media.dvpp_get_pic_desc接口获取指定图片的图片描述信息。 - 调用acl.media.dvpp_set_pic_desc系列接口设置图片描述参数（例如，内存地址、内存大小、图片格式、宽、高等）。 roi_nums_list: list, 表示抠图的数量列表, 当前最大抠图数量为256。 “roi_nums_list”总和小于等于256, 与“dst_batch_pic_descs”中的“batch_size”值保持一致。(roi_nums_list[0] + ... + roi_nums_list[size - 1] <= 256) dst_batch_pic_descs: int, 指定批量输出图片描述信息的指针地址。 - 用acl.media.dvpp_create_batch_pic_desc接口创建批量图片描述信息。 - 调用acl.media.dvpp_get_pic_desc接口获取指定图片的图片描述信息。 - 调用acl.media.dvpp_set_pic_desc系列接口设置图片描述参数（例如，内存地址、内存大小、图片格式、宽、高等）。 crop_area_list: list, 抠图区域位置列表。调用acl.media.dvpp_create_roi_config接口创建区域位置数据。 stream: int, 指定Stream。
返回值说明	dst_batch_pic_descs: int, 批量输出图片描述信息。 ret: int, 错误码。 - 返回0, 表示任务下发成功。 - 返回非0, 表示任务下发失败。

约束说明

版本	约束
Atlas 200/300/500 推理产品 Atlas 训练系列产品	- 关于输入图片分辨率、图片格式： - 输入图片分辨率在$10 * 6 \sim 4096 * 4096$（包括4096）范围内时，支持图片格式、宽高对齐、内存约束处说明的输入图片格式。 - 输入图片的宽或高在4096 ~ 8192（包括8192）范围内时，支持格式如下： 支持aclDvppPixelFormat枚举值中的如下枚举项： <pre>PIXEL_FORMAT_YUV_SEMIPLANAR_420 = 1, // YUV420SP NV12 8bit PIXEL_FORMAT_YVU_SEMIPLANAR_420 = 2, // YUV420SP NV21 8bit</pre> - 关于输出图片分辨率、图片格式请参见图片格式、宽高对齐、内存约束。 - 关于各图片格式的宽高对齐、内存大小约束，请参见图片格式、宽高对齐、内存约束。 - 输出图片区域与“crop_area”不一致时会对图片再做一次缩放操作，默认缩放算法为“华为自研的高阶滤波算法”。
Atlas 推理系列产品	- 关于输入图片分辨率、图片格式： - 输入图片分辨率在$10 * 6 \sim 4096 * 4096$（包括4096）范围内时，支持图片格式、宽高对齐、内存约束处说明的输入图片格式。 - 输入图片的宽或高在4096 ~ 8192（包括8192）范围内时，支持格式如下： 支持aclDvppPixelFormat枚举值中的如下枚举项： <pre>PIXEL_FORMAT_YUV_400 = 0, // YUV400 8bit PIXEL_FORMAT_YUV_SEMIPLANAR_420 = 1, // YUV420SP NV12 8bit PIXEL_FORMAT_YVU_SEMIPLANAR_420 = 2, // YUV420SP NV21 8bit PIXEL_FORMAT_YUV_SEMIPLANAR_422 = 3, // YUV422SP 8bit PIXEL_FORMAT_YVU_SEMIPLANAR_422 = 4, // YVU422SP 8bit PIXEL_FORMAT_YUV_SEMIPLANAR_444 = 5, // YUV444SP 8bit PIXEL_FORMAT_YVU_SEMIPLANAR_444 = 6, // YVU444SP 8bit PIXEL_FORMAT_YUYV_PACKED_422 = 7, // YUV422Packed YUYV 8bit PIXEL_FORMAT_UYVY_PACKED_422 = 8, // YUV422Packed UYVY 8bit PIXEL_FORMAT_VYU_PACKED_422 = 9, // YUV422Packed VYU 8bit PIXEL_FORMAT_VYUY_PACKED_422 = 10, // YUV422Packed VYUY 8bit</pre> - 关于输出图片分辨率、图片格式请参见图片格式、宽高对齐、内存约束。 - 关于各图片格式的宽高对齐、内存大小约束，请参见图片格式、宽高对齐、内存约束。 - 输出图片区域与“crop_area”不一致时会对图片再做一次缩放操作，默认缩放算法为“业界通用的Bilinear算法”。

版本	约束
Atlas A2训练系列产品/ Atlas 800I A2推理产品 Atlas 200I/500 A2推理 产品	- 关于输入、输出图片分辨率： - 输入图片分辨率：10 * 6 ~ 8192 * 8192 - 输出图片分辨率：10 * 6 ~ 4096 * 8192 - 关于图片格式、宽高对齐、内存等总体约束，请参见图片格式、宽高对齐、内存约束。 - 输出图片区域与“crop_area”不一致时会对图片再做一次缩放操作，默认缩放算法为“业界通用的Bilinear算法”。

2.9.6.8 函数：dvpp_vpc_batch_crop_resize_async

C函数原型	<code>aclError aclDvppVpcBatchCropResizeAsync(aclDvppChannelDesc *channelDesc, aclDvppBatchPicDesc *srcBatchPicDescs, uint32_t *roiNums, uint32_t size, aclDvppBatchPicDesc *dstBatchPicDescs, aclDvppRoiConfig *cropAreas[], aclDvppResizeConfig resizeConfig, aclrtStream stream)</code>
Python函数	<code>ret = acl.media.dvpp_vpc_batch_crop_resize_async(dvpp_channel_ desc, src_batch_pic_descs, roi_nums_list, dst_batch_pic_descs, crop_area_list, resize_config, stream)</code>
函数功能	对于一张或多张输入图片，支持从每个输入图片中抠出一张或多张子图并——对应存放到多个输出区域，输出图片区域与抠图区域不一致时会对图片再做一次缩放操作，按照“resize_config”处配置的缩放算法。异步接口。

输入说明	dvpp_channel_desc: int, 指定通道描述信息的指针地址, 与调用acl.media.dvpp_create_channel接口创建通道时指定的“dvpp_channel_desc”保持一致。 src_batch_pic_descs: int, 指定批量输入图片描述信息的指针地址。 - 调用acl.media.dvpp_create_batch_pic_desc接口创建批量图片描述信息。 - 调用acl.media.dvpp_get_pic_desc接口获取指定图片的图片描述信息。 - 调用acl.media.dvpp_set_pic_desc系列接口设置图片描述参数（例如，内存地址、内存大小、图片格式、宽、高等）。 roi_nums_list: list, 表示抠图的数量列表, 当前最大抠图数量为256。 列表总和小于等于256, 与“dst_batch_pic_descs”中的“batch_size”值保持一致。(roi_nums_list[0] + ... + roi_nums_list[size - 1] <= 256) dst_batch_pic_descs: int, 指定批量输出图片描述信息的指针地址。 - 调用acl.media.dvpp_create_batch_pic_desc接口创建批量图片描述信息。 - 调用acl.media.dvpp_get_pic_desc接口获取指定图片的图片描述信息。 - 调用acl.media.dvpp_set_pic_desc系列接口设置图片描述参数（例如，内存地址、内存大小、图片格式、宽、高等）。 crop_area_list: list, 抠图区域位置列表。 调用acl.media.dvpp_create_roi_config接口创建区域位置数据。 resize_config: int, 指定图片缩放配置数据的指针地址。 需提前调用acl.media.dvpp_create_resize_config接口创建图片缩放配置数据。 stream: int, 指定Stream。
返回值说明	dst_batch_pic_descs: int, 批量输出图片描述信息的指针地址。 ret: int, 错误码。 - 返回0, 表示任务下发成功。 - 返回非0, 表示任务下发失败。

约束说明

版本	约束
Atlas 200/300/500 推理产品 Atlas 训练系列产品	- 关于输入图片分辨率、图片格式： - 输入图片分辨率在$10 * 6 \sim 4096 * 4096$（包括4096）范围内时，支持图片格式、宽高对齐、内存约束处说明的输入图片格式。 - 输入图片的宽或高在$4096 \sim 8192$（包括8192）范围内时，支持格式如下： 支持acldvppPixelFormat枚举值中的如下枚举项： PIXEL_FORMAT_YUV_SEMIPLANAR_420 = 1, // YUV420SP NV12 8bit PIXEL_FORMAT_YVU_SEMIPLANAR_420 = 2, // YUV420SP NV21 8bit - 关于输出图片分辨率、图片格式请参见图片格式、宽高对齐、内存约束。 - 关于各图片格式的宽高对齐、内存大小约束，请参见图片格式、宽高对齐、内存约束。
Atlas 推理系列产品	- 关于输入图片分辨率、图片格式： - 输入图片分辨率在$10 * 6 \sim 4096 * 4096$（包括4096）范围内时，支持图片格式、宽高对齐、内存约束处说明的输入图片格式。 - 输入图片的宽或高在$4096 \sim 8192$（包括8192）范围内时，支持格式如下： 支持acldvppPixelFormat枚举值中的如下枚举项： PIXEL_FORMAT_YUV_400 = 0, // YUV400 8bit PIXEL_FORMAT_YUV_SEMIPLANAR_420 = 1, // YUV420SP NV12 8bit PIXEL_FORMAT_YVU_SEMIPLANAR_420 = 2, // YUV420SP NV21 8bit PIXEL_FORMAT_YUV_SEMIPLANAR_422 = 3, // YUV422SP 8bit PIXEL_FORMAT_YVU_SEMIPLANAR_422 = 4, // YVU422SP 8bit PIXEL_FORMAT_YUV_SEMIPLANAR_444 = 5, // YUV444SP 8bit PIXEL_FORMAT_YVU_SEMIPLANAR_444 = 6, // YVU444SP 8bit PIXEL_FORMAT_YUYV_PACKED_422 = 7, // YUV422Packed YUYV 8bit PIXEL_FORMAT_UYVY_PACKED_422 = 8, // YUV422Packed UYVY 8bit PIXEL_FORMAT_YVYU_PACKED_422 = 9, // YUV422Packed YVYU 8bit PIXEL_FORMAT_VYUY_PACKED_422 = 10, // YUV422Packed VYUY 8bit - 关于输出图片分辨率、图片格式请参见图片格式、宽高对齐、内存约束。 - 关于各图片格式的宽高对齐、内存大小约束，请参见图片格式、宽高对齐、内存约束。 - 当输入、输出图片格式都为ARGB8888、ABGR8888、RGBA8888、BGRA8888格式时，在经过本接口缩放处理后，A分量值会变为0。

版本	约束
Atlas A2训练系列产品/ Atlas 800I A2推理产品 Atlas 200I/500 A2推理 产品	- 关于输入、输出图片分辨率： - 输入图片分辨率：10 * 6 ~ 8192 * 8192 - 输出图片分辨率：10 * 6 ~ 4096 * 8192 - 关于图片格式、宽高对齐、内存等总体约束，请参见图片格式、宽高对齐、内存约束。 - 当输入、输出图片格式都为ARGB8888、ABGR8888、RGBA8888、BGRA8888格式时，在经过本接口缩放处理后，A分量值会变为0。

2.9.6.9 函数：dvpp_vpc_crop_and_paste_async

C函数原型	<code>aclError aclDvppVpcCropAndPasteAsync(aclDvppChannelDesc *channelDesc, aclDvppPicDesc *inputDesc, aclDvppPicDesc *outputDesc, aclDvppRoiConfig *cropArea, aclDvppRoiConfig *pasteArea, aclrtStream stream)</code>
Python函数	<code>ret = acl.media.dvpp_vpc_crop_and_paste_async(dvpp_channel_de sc, input_dvpp_pic_desc, output_dvpp_pic_desc, crop_area, paste_area, stream)</code>
函数功能	按指定区域从一个输入图片中抠出一张子图，再将抠的子图贴到目标图片的指定位置，作为输出图片。异步接口。

输入说明	dvpp_channel_desc: int, 指定通道描述信息的指针地址, 与调用acl.media.dvpp_create_channel接口创建通道时指定的dvpp_channel_desc保持一致。 input_dvpp_pic_desc: int, 输入图片信息的指针地址。 - 调用acl.media.dvpp_create_pic_desc接口创建图片描述信息。 - 调用acl.media.dvpp_set_pic_desc系列接口设置图片描述参数 (例如, 图片格式、宽、高等)。 output_dvpp_pic_desc: int, 输出图片信息的指针地址。 - 调用acl.media.dvpp_create_pic_desc接口创建图片描述信息。 - 调用acl.media.dvpp_set_pic_desc系列接口设置图片描述参数 (例如, 图片格式、宽、高等)。 crop_area: int, 抠图区域位置的指针地址。 调用acl.media.dvpp_create_roi_config接口创建区域位置数据。 paste_area: int, 贴图区域位置的指针地址。 - 调用acl.media.dvpp_create_roi_config接口创建区域位置数据。 - 贴图区域左偏移需要16对齐。 stream: int, 指定Stream对象。
返回值说明	ret: int, 错误码。 - 返回0, 表示任务下发成功。 - 返回非0, 表示任务下发失败。

约束说明

版本	约束
Atlas 200/300/500 推理产品 Atlas 训练系列产品	- 关于输入图片分辨率、图片格式： - 输入图片分辨率在$10 * 6 \sim 4096 * 4096$（包括4096）范围内时，支持图片格式、宽高对齐、内存约束处说明的输入图片格式。 - 输入图片的宽或高在4096 ~ 8192（包括8192）范围内时，支持格式如下： 支持aclDvppPixelFormat枚举值中的如下枚举项： <pre>PIXEL_FORMAT_YUV_SEMIPLANAR_420 = 1, // YUV420SP NV12 8bit PIXEL_FORMAT_YVU_SEMIPLANAR_420 = 2, // YUV420SP NV21 8bit</pre> - 关于输出图片分辨率、图片格式请参见图片格式、宽高对齐、内存约束。 - 关于各图片格式的宽高对齐、内存大小约束，请参见图片格式、宽高对齐、内存约束。 - 当“paste_area”的宽高与“crop_area”的宽高不一致时会对图片做一次缩放操作，默认缩放算法为“华为自研的高阶滤波算法”。
Atlas 推理系列产品	- 关于输入图片分辨率、图片格式： - 输入图片分辨率在$10 * 6 \sim 4096 * 4096$（包括4096）范围内时，支持图片格式、宽高对齐、内存约束处说明的输入图片格式。 - 输入图片的宽或高在4096 ~ 8192（包括8192）范围内时，支持格式如下： 支持aclDvppPixelFormat枚举值中的如下枚举项： <pre>PIXEL_FORMAT_YUV_400 = 0, // YUV400 8bit PIXEL_FORMAT_YUV_SEMIPLANAR_420 = 1, // YUV420SP NV12 8bit PIXEL_FORMAT_YVU_SEMIPLANAR_420 = 2, // YUV420SP NV21 8bit PIXEL_FORMAT_YUV_SEMIPLANAR_422 = 3, // YUV422SP 8bit PIXEL_FORMAT_YVU_SEMIPLANAR_422 = 4, // YVU422SP 8bit PIXEL_FORMAT_YUV_SEMIPLANAR_444 = 5, // YUV444SP 8bit PIXEL_FORMAT_YVU_SEMIPLANAR_444 = 6, // YVU444SP 8bit PIXEL_FORMAT_YUVV_PACKED_422 = 7, // YUV422Packed YUYV 8bit PIXEL_FORMAT_UYVY_PACKED_422 = 8, // YUV422Packed UYVY 8bit PIXEL_FORMAT_VYUY_PACKED_422 = 9, // YUV422Packed VYUY 8bit PIXEL_FORMAT_VYUY_PACKED_422 = 10, // YUV422Packed VYUY 8bit</pre> - 关于输出图片分辨率、图片格式请参见图片格式、宽高对齐、内存约束。 - 关于各图片格式的宽高对齐、内存大小约束，请参见图片格式、宽高对齐、内存约束。 - 当“paste_area”的宽高与“crop_area”的宽高不一致时会对图片做一次缩放操作，默认缩放算法为“业界通用的Bilinear算法”。

版本	约束
Atlas A2训练系列产品/ Atlas 800I A2推理产品 Atlas 200I/500 A2推理 产品	- 关于图片分辨率、图片格式、宽高对齐、内存等总体约束，请参见图片格式、宽高对齐、内存约束。 - 当“paste_area”的宽高与“crop_area”的宽高不一致时会对图片做一次缩放操作，默认缩放算法为“业界通用的Bilinear算法”。 - 由于YUV格式图像下采样约束，当输出图片格式为YUV420SP或YUV422SP格式时，贴图的顶部偏移、左侧偏移建议为偶数。

2.9.6.10 函数：dvpp_vpc_crop_resize_paste_async

C函数原型	<code>aclError aclDvppVpcCropResizePasteAsync(aclDvppChannelDesc *channelDesc, aclDvppPicDesc *inputDesc, aclDvppPicDesc *outputDesc, aclDvppRoiConfig *cropArea, aclDvppRoiConfig *pasteArea, aclDvppResizeConfig *resizeConfig, aclrtStream stream)</code>
Python函数	<code>ret = acl.media.dvpp_vpc_crop_resize_paste_async(dvpp_channel_ desc, input_dvpp_pic_desc, output_dvpp_pic_desc, crop_area, paste_area, resize_config, stream)</code>
函数功能	按指定区域从输入图片中抠图，再将抠的图片贴到目标图片的指定位置，作为输出图片。异步接口。 “crop_area”的宽高与“paste_area”宽高不一致时会对图片再做一次缩放操作，按照“resize_config”处配置的缩放算法。

输入说明	dvpp_channel_desc: int, 指定通道描述信息的指针地址, 与调用acl.media.dvpp_create_channel接口创建通道时指定的dvpp_channel_desc保持一致。 input_dvpp_pic_desc: int, 输入图片信息的指针地址。 - 调用acl.media.dvpp_create_pic_desc接口创建图片描述信息。 - 调用acl.media.dvpp_set_pic_desc系列接口设置图片描述参数 (例如, 内存地址、内存大小、图片格式、宽、高等)。 output_dvpp_pic_desc: int, 输出图片信息的指针地址。 - 调用acl.media.dvpp_create_pic_desc接口创建图片描述信息。 - 调用acl.media.dvpp_set_pic_desc系列接口设置图片描述参数 (例如, 内存地址、内存大小、图片格式、宽、高等)。 crop_area: int, 抠图区域位置的指针地址。 调用acl.media.dvpp_create_roi_config接口创建区域位置数据。 paste_area: int, 贴图区域位置的指针地址。 - 调用acl.media.dvpp_create_roi_config接口创建区域位置数据。 - 贴图区域左偏移需要16对齐。 resize_config: int, 指定图片缩放配置数据的指针地址。 需提前调用acl.media.dvpp_create_resize_config接口创建图片缩放配置数据。 stream: int, 指定Stream对象。
返回值说明	ret: int, 错误码。 - 返回0, 表示任务下发成功。 - 返回非0, 表示任务下发失败。
参考资源	接口调用示例, 参见抠图缩放 (一图一框) 示例代码。

约束说明

版本	约束
Atlas 200/300/500 推理产品 Atlas 训练系列产品	- 关于输入图片分辨率、图片格式： - 输入图片分辨率在$10 * 6 \sim 4096 * 4096$（包括4096）范围内时，支持图片格式、宽高对齐、内存约束处说明的输入图片格式。 - 输入图片的宽或高在$4096 \sim 8192$（包括8192）范围内时，支持格式如下： 支持acldvppPixelFormat枚举值中的如下枚举项： PIXEL_FORMAT_YUV_SEMIPLANAR_420 = 1, // YUV420SP NV12 8bit PIXEL_FORMAT_YVU_SEMIPLANAR_420 = 2, // YUV420SP NV21 8bit - 关于输出图片分辨率、图片格式请参见图片格式、宽高对齐、内存约束。 - 关于各图片格式的宽高对齐、内存大小约束，请参见图片格式、宽高对齐、内存约束。
Atlas 推理系列产品	- 关于输入图片分辨率、图片格式： - 输入图片分辨率在$10 * 6 \sim 4096 * 4096$（包括4096）范围内时，支持图片格式、宽高对齐、内存约束处说明的输入图片格式。 - 输入图片的宽或高在$4096 \sim 8192$（包括8192）范围内时，支持格式如下： 支持acldvppPixelFormat枚举值中的如下枚举项： PIXEL_FORMAT_YUV_400 = 0, // YUV400 8bit PIXEL_FORMAT_YUV_SEMIPLANAR_420 = 1, // YUV420SP NV12 8bit PIXEL_FORMAT_YVU_SEMIPLANAR_420 = 2, // YUV420SP NV21 8bit PIXEL_FORMAT_YUV_SEMIPLANAR_422 = 3, // YUV422SP 8bit PIXEL_FORMAT_YVU_SEMIPLANAR_422 = 4, // YVU422SP 8bit PIXEL_FORMAT_YUV_SEMIPLANAR_444 = 5, // YUV444SP 8bit PIXEL_FORMAT_YVU_SEMIPLANAR_444 = 6, // YVU444SP 8bit PIXEL_FORMAT_YUYV_PACKED_422 = 7, // YUV422Packed YUYV 8bit PIXEL_FORMAT_UYVY_PACKED_422 = 8, // YUV422Packed UYVY 8bit PIXEL_FORMAT_YVYU_PACKED_422 = 9, // YUV422Packed YVYU 8bit PIXEL_FORMAT_VYUY_PACKED_422 = 10, // YUV422Packed VYUY 8bit - 关于输出图片分辨率、图片格式请参见图片格式、宽高对齐、内存约束。 - 关于各图片格式的宽高对齐、内存大小约束，请参见图片格式、宽高对齐、内存约束。 - 当输入、输出图片格式都为ARGB8888、ABGR8888、RGBA8888、BGRA8888格式时，在经过本接口缩放处理后，A分量值会变为0。

版本	约束
Atlas A2训练系列产品/ Atlas 800I A2推理产品 Atlas 200I/500 A2推理 产品	- 关于图片分辨率、图片格式、宽高对齐、内存等总体约束，请参见图片格式、宽高对齐、内存约束。 - 由于YUV格式图像下采样约束，当输出图片格式为YUV420SP或YUV422SP格式时，贴图的顶部偏移、左侧偏移建议为偶数。 - 当输入、输出图片格式都为ARGB8888、ABGR8888、RGBA8888、BGRA8888格式时，在经过本接口缩放处理后，A分量值会变为0。

2.9.6.11 函数：dvpp_vpc_batch_crop_and_paste_async

C函数原型	<code>aclError aclDvppVpcBatchCropAndPasteAsync(aclDvppChannelDesc *channelDesc, aclDvppBatchPicDesc *srcBatchPicDesc, uint32_t *roiNums, uint32_t size, aclDvppBatchPicDesc *dstBatchPicDesc, aclDvppRoiConfig *cropAreas[], aclDvppRoiConfig *pasteAreas[], aclrtStream stream)</code>
Python函数	<code>dst_batch_pic_descs, ret = acl.media.dvpp_vpc_batch_crop_and_paste_async(dvpp_chan nel_desc, src_batch_pic_descs, roi_nums_list, dst_batch_pic_descs, crop_area_list, paste_area_list, stream)</code>
函数功能	按指定区域从一张或多张输入图片中抠出一个或多个子图，再将抠的子图贴到目标图片的指定位置，作为输出图片。异步接口。

输入说明	dvpp_channel_desc: int, 指定通道描述信息的指针地址, 与调用acl.media.dvpp_create_channel接口创建通道时指定的“dvpp_channel_desc”保持一致。 src_batch_pic_descs: int, 指定批量输入图片描述信息的指针地址。 - 调用acl.media.dvpp_create_batch_pic_desc接口创建批量图片描述信息。 - 调用acl.media.dvpp_get_pic_desc接口获取指定图片的图片描述信息。 - 调用acl.media.dvpp_set_pic_desc系列接口设置图片描述参数（例如，内存地址、内存大小、图片格式、宽、高等）。 roi_nums_list: list, 表示抠图的数量列表, 当前最大抠图数量为256。 “roi_nums_list”总和小于等于256, 与“dst_batch_pic_descs”中的“batch_size”值保持一致。（roi_nums_list[0] + ... + roi_nums_list[size - 1] <= 256） dst_batch_pic_descs: int, 指定批量输出图片描述信息的指针地址。 - 调用acl.media.dvpp_create_batch_pic_desc接口创建批量图片描述信息。 - 调用acl.media.dvpp_get_pic_desc接口获取指定图片的图片描述信息。 - 调用acl.media.dvpp_set_pic_desc系列接口设置图片描述参数（例如，内存地址、内存大小、图片格式、宽、高等）。 crop_area_list: list, 抠图区域位置列表。 - 调用acl.media.dvpp_create_roi_config接口创建区域位置数据。 “crop_area”、“paste_area”中元素个数与“dst_batch_pic_descs”中的“batch_size”值相等。 paste_area_list: list, 贴图区域位置列表。 - 调用acl.media.dvpp_create_roi_config接口创建区域位置数据。 - 贴图区域左偏移需要16对齐。 “crop_area”、“paste_area”中元素个数与“dst_batch_pic_descs”中的“batch_size”值相等。 stream: int, 指定Stream。
返回值说明	dst_batch_pic_descs: int, 批量输出图片描述信息的指针地址。 ret: int, 错误码。 - 返回0表示任务下发成功。 - 返回非0表示任务下发失败。
参考资源	接口调用示例, 参见抠图贴图（一图多框）示例代码。

约束说明

版本	约束
Atlas 200/300/500 推理产品 Atlas 训练系列产品	- 关于输入图片分辨率、图片格式： - 输入图片分辨率在$10 * 6 \sim 4096 * 4096$（包括4096）范围内时，支持图片格式、宽高对齐、内存约束处说明的输入图片格式。 - 输入图片的宽或高在4096 ~ 8192（包括8192）范围内时，支持格式如下： 支持aclDvppPixelFormat枚举值中的如下枚举项： PIXEL_FORMAT_YUV_SEMIPLANAR_420 = 1, // YUV420SP NV12 8bit PIXEL_FORMAT_YVU_SEMIPLANAR_420 = 2, // YUV420SP NV21 8bit - 关于输出图片分辨率、图片格式请参见图片格式、宽高对齐、内存约束。 - 关于各图片格式的宽高对齐、内存大小约束，请参见图片格式、宽高对齐、内存约束。 - 当“paste_area”的宽高与“crop_area”的宽高不一致时会对图片做一次缩放操作，默认缩放算法为“华为自研的高阶滤波算法”。
Atlas 推理系列产品	- 关于输入图片分辨率、图片格式： - 输入图片分辨率在$10 * 6 \sim 4096 * 4096$（包括4096）范围内时，支持图片格式、宽高对齐、内存约束处说明的输入图片格式。 - 输入图片的宽或高在4096 ~ 8192（包括8192）范围内时，支持格式如下： 支持aclDvppPixelFormat枚举值中的如下枚举项： PIXEL_FORMAT_YUV_400 = 0, // YUV400 8bit PIXEL_FORMAT_YUV_SEMIPLANAR_420 = 1, // YUV420SP NV12 8bit PIXEL_FORMAT_YVU_SEMIPLANAR_420 = 2, // YUV420SP NV21 8bit PIXEL_FORMAT_YUV_SEMIPLANAR_422 = 3, // YUV422SP 8bit PIXEL_FORMAT_YVU_SEMIPLANAR_422 = 4, // YVU422SP 8bit PIXEL_FORMAT_YUV_SEMIPLANAR_444 = 5, // YUV444SP 8bit PIXEL_FORMAT_YVU_SEMIPLANAR_444 = 6, // YVU444SP 8bit PIXEL_FORMAT_YUYV_PACKED_422 = 7, // YUV422Packed YUYV 8bit PIXEL_FORMAT_UYVY_PACKED_422 = 8, // YUV422Packed UYVY 8bit PIXEL_FORMAT_YVYU_PACKED_422 = 9, // YUV422Packed YVYU 8bit PIXEL_FORMAT_VYUY_PACKED_422 = 10, // YUV422Packed VYUY 8bit - 关于输出图片分辨率、图片格式请参见图片格式、宽高对齐、内存约束。 - 关于各图片格式的宽高对齐、内存大小约束，请参见图片格式、宽高对齐、内存约束。 - 当“paste_area”的宽高与“crop_area”的宽高不一致时会对图片做一次缩放操作，默认缩放算法为“业界通用的Bilinear算法”。

版本	约束
Atlas A2训练系列产品/ Atlas 800I A2推理产品 Atlas 200I/500 A2推理 产品	- 关于图片分辨率、图片格式、宽高对齐、内存等总体约束，请参见图片格式、宽高对齐、内存约束。 - 当“paste_area”的宽高与“crop_area”的宽高不一致时会对图片做一次缩放操作，默认缩放算法为“业界通用的Bilinear算法”。 - 由于YUV格式图像下采样约束，当输出图片格式为YUV420SP或YUV422SP格式时，贴图的顶部偏移、左侧偏移建议为偶数。

2.9.6.12 函数：dvpp_vpc_batch_crop_resize_paste_async

C函数原型	<code>aclError aclDvppVpcBatchCropResizePasteAsync(aclDvppChannelDesc *channelDesc, aclDvppBatchPicDesc *srcBatchPicDescs, uint32_t *roiNums, uint32_t size, aclDvppBatchPicDesc *dstBatchPicDescs, aclDvppRoiConfig *cropAreas[], aclDvppRoiConfig *pasteAreas[], aclDvppResizeConfig *resizeConfig, aclrtStream stream)</code>
Python函数	<code>ret = acl.media.dvpp_vpc_batch_crop_resize_paste_async(dvpp_ch annel_desc, src_batch_pic_descs, roi_nums_list, dst_batch_pic_descs, crop_area_list, paste_area_list, resize_config, stream)</code>
函数功能	按指定区域从一张或多张输入图片中抠出一个或多个子图，再将抠的子图贴到目标图片的指定位置，作为输出图片。异步接口。 当“paste_area”的宽高与“crop_area”的宽高不一致时会对图片做一次缩放操作，按照“resize_config”处配置的缩放算法。

输入说明	dvpp_channel_desc: int, 指定通道描述信息的指针地址, 与调用acl.media.dvpp_create_channel接口创建通道时指定的“dvpp_channel_desc”保持一致。 src_batch_pic_descs: int, 指定批量输入图片描述信息的指针地址。 - 调用acl.media.dvpp_create_batch_pic_desc接口创建批量图片描述信息。 - 调用acl.media.dvpp_get_pic_desc接口获取指定图片的图片描述信息。 - 调用acl.media.dvpp_set_pic_desc系列接口设置图片描述参数（例如，内存地址、内存大小、图片格式、宽、高等）。 roi_nums_list: list, 指定每个输入图片的抠图数量, 当前最大抠图数量为256。 列表总和小于等于256, 与“dst_batch_pic_descs”中的“batch_size”值保持一致。（roi_nums_list[0] + ... + roi_nums_list[size - 1] <= 256） dst_batch_pic_descs: int, 指定批量输出图片描述信息的指针地址。 - 调用acl.media.dvpp_create_batch_pic_desc接口创建批量图片描述信息。 - 调用acl.media.dvpp_get_pic_desc接口获取指定图片的图片描述信息。 - 调用acl.media.dvpp_set_pic_desc系列接口设置图片描述参数（例如，内存地址、内存大小、图片格式、宽、高等）。 crop_area_list: list, 抠图区域位置列表。 - 调用acl.media.dvpp_create_roi_config接口创建区域位置数据。 “crop_area_list”、“paste_area_list”中元素个数与“dst_batch_pic_descs”中的“batch_size”值相等。 paste_area_list: list, 贴图区域位置列表。 - 调用acl.media.dvpp_create_roi_config接口创建区域位置数据。 - 贴图区域左偏移需要16对齐。 “crop_area_list”、“paste_area_list”中元素个数与“dst_batch_pic_descs”中的“batch_size”值相等。 resize_config: int, 指定图片缩放配置数据的指针地址。 需提前调用acl.media.dvpp_create_resize_config接口创建图片缩放配置数据。 stream: int, 指定Stream。
返回值说明	ret: int, 错误码。 - 返回0表示任务下发成功。 - 返回非0表示任务下发失败。

约束说明

版本	约束
Atlas 200/300/500 推理产品 Atlas 训练系列产品	- 关于输入图片分辨率、图片格式： - 输入图片分辨率在$10 * 6 \sim 4096 * 4096$（包括4096）范围内时，支持图片格式、宽高对齐、内存约束处说明的输入图片格式。 - 输入图片的宽或高在$4096 \sim 8192$（包括8192）范围内时，支持格式如下： 支持acldvppPixelFormat枚举值中的如下枚举项： PIXEL_FORMAT_YUV_SEMIPLANAR_420 = 1, // YUV420SP NV12 8bit PIXEL_FORMAT_YVU_SEMIPLANAR_420 = 2, // YUV420SP NV21 8bit - 关于输出图片分辨率、图片格式请参见图片格式、宽高对齐、内存约束。 - 关于各图片格式的宽高对齐、内存大小约束，请参见图片格式、宽高对齐、内存约束。
Atlas 推理系列产品	- 关于输入图片分辨率、图片格式： - 输入图片分辨率在$10 * 6 \sim 4096 * 4096$（包括4096）范围内时，支持图片格式、宽高对齐、内存约束处说明的输入图片格式。 - 输入图片的宽或高在$4096 \sim 8192$（包括8192）范围内时，支持格式如下： 支持acldvppPixelFormat枚举值中的如下枚举项： PIXEL_FORMAT_YUV_400 = 0, // YUV400 8bit PIXEL_FORMAT_YUV_SEMIPLANAR_420 = 1, // YUV420SP NV12 8bit PIXEL_FORMAT_YVU_SEMIPLANAR_420 = 2, // YUV420SP NV21 8bit PIXEL_FORMAT_YUV_SEMIPLANAR_422 = 3, // YUV422SP 8bit PIXEL_FORMAT_YVU_SEMIPLANAR_422 = 4, // YVU422SP 8bit PIXEL_FORMAT_YUV_SEMIPLANAR_444 = 5, // YUV444SP 8bit PIXEL_FORMAT_YVU_SEMIPLANAR_444 = 6, // YVU444SP 8bit PIXEL_FORMAT_YUYV_PACKED_422 = 7, // YUV422Packed YUYV 8bit PIXEL_FORMAT_UYVY_PACKED_422 = 8, // YUV422Packed UYVY 8bit PIXEL_FORMAT_YVYU_PACKED_422 = 9, // YUV422Packed YVYU 8bit PIXEL_FORMAT_VYUY_PACKED_422 = 10, // YUV422Packed VYUY 8bit - 关于输出图片分辨率、图片格式请参见图片格式、宽高对齐、内存约束。 - 关于各图片格式的宽高对齐、内存大小约束，请参见图片格式、宽高对齐、内存约束。 - 当输入、输出图片格式都为ARGB8888、ABGR8888、RGBA8888、BGRA8888格式时，在经过本接口缩放处理后，A分量值会变为0。

版本	约束
Atlas A2训练系列产品/ Atlas 800I A2推理产品 Atlas 200I/500 A2推理产品	- 关于图片分辨率、图片格式、宽高对齐、内存等总体约束，请参见图片格式、宽高对齐、内存约束。 - 由于YUV格式图像下采样约束，当输出图片格式为YUV420SP或YUV422SP格式时，贴图的顶部偏移、左侧偏移建议为偶数。 - 当输入、输出图片格式都为ARGB8888、ABGR8888、RGBA8888、BGRA8888格式时，在经过本接口缩放处理后，A分量值会变为0。

2.9.6.13 函数：dvpp_vpc_batch_crop_resize_make_border_async

C函数原型	<code>aclError aclDvppVpcBatchCropResizeMakeBorderAsync(aclDvppChannelDesc *channelDesc, aclDvppBatchPicDesc *srcBatchPicDescs, uint32_t *roiNums, uint32_t size, aclDvppBatchPicDesc *dstBatchPicDescs, aclDvppRoiConfig *cropAreas[], aclDvppBorderConfig *borderCfgs[], aclDvppResizeConfig *resizeConfig, aclrtStream stream)</code>
Python函数	<code>ret = acl.media.dvpp_vpc_batch_crop_resize_make_border_async(vpp_channel_desc, src_batch_pic_descs, roi_nums_list, dst_batch_pic_descs, crop_area_list, border_cfgs_list, resize_config, stream)</code>
函数功能	支持从一张或多张输入图片中抠出一张或多张子图，对每个子图进行进行缩放，扩边填充后输出至目标图像内存地址。异步接口。 Atlas 200/300/500 推理产品 ，当前版本不支持该接口。 Atlas 训练系列产品 ，当前版本不支持该接口。

输入说明	dvpp_channel_desc: int, 指定通道描述信息的指针地址, 与调用acl.media.dvpp_create_channel接口创建通道时指定的“dvpp_channel_desc”保持一致。 src_batch_pic_descs: int, 指定批量输入图片描述信息的指针地址。 - 调用acl.media.dvpp_create_batch_pic_desc接口创建批量图片描述信息。 - 调用acl.media.dvpp_get_pic_desc接口获取指定图片的图片描述信息。 - 调用acl.media.dvpp_set_pic_desc系列接口设置图片描述参数（例如，内存地址、内存大小、图片格式、宽、高等）。 roi_nums_list: list, 指定每个输入图片的抠图数量, 当前最大抠图数量为256。 列表总和小于等于256, 与“dst_batch_pic_descs”中的“batch_size”值保持一致。（roi_nums_list[0] + ... + roi_nums_list[size - 1] <= 256） dst_batch_pic_descs: int, 指定批量输出图片描述信息的指针地址。 - 调用acl.media.dvpp_create_batch_pic_desc接口创建批量图片描述信息。 - 调用acl.media.dvpp_get_pic_desc接口获取指定图片的图片描述信息。 - 调用acl.media.dvpp_set_pic_desc系列接口设置图片描述参数（例如，内存地址、内存大小、图片格式、宽、高等）。 crop_area_list: list, 抠图区域位置列表。 - 调用acl.media.dvpp_create_roi_config接口创建区域位置数据。 “crop_area_list”、“paste_area_list”中元素个数与“dst_batch_pic_descs”中的“batch_size”值相等。 border_cfgs_list: list, 扩边填充配置列表。 “border_cfgs”数组中的元素与“crop_areas”数组中的元素一一对应。 - 扩边填充配置参数中的“border_type”参数值仅支持“BORDER_CONSTANT”和“BORDER_REPLICATE”。 resize_config: int, 指定图片缩放配置数据的指针地址。 需提前调用acl.media.dvpp_create_resize_config接口创建图片缩放配置数据。 VPC进行缩放处理时, 缩放后的宽/高是输出图片的宽/高减去扩边的值。 stream: int, 指定Stream。
返回值说明	ret: int, 错误码。 - 返回0, 表示任务下发成功。 - 返回非0, 表示任务下发失败。

约束说明

版本	约束
Atlas 推理系列产品	- 关于输入图片分辨率、图片格式： - 输入图片分辨率在10 * 6 ~ 4096 * 4096（包括4096）范围内时，支持图片格式、宽高对齐、内存约束处说明的输入图片格式。 - 输入图片的宽或高在4096 ~ 8192（包括8192）范围内时，支持格式如下： 支持acldvppPixelFormat枚举值中的如下枚举项：<pre>PIXEL_FORMAT_YUV_400 = 0, // YUV400 8bit PIXEL_FORMAT_YUV_SEMIPLANAR_420 = 1, // YUV420SP NV12 8bit PIXEL_FORMAT_YVU_SEMIPLANAR_420 = 2, // YUV420SP NV21 8bit PIXEL_FORMAT_YUV_SEMIPLANAR_422 = 3, // YUV422SP 8bit PIXEL_FORMAT_YVU_SEMIPLANAR_422 = 4, // YVU422SP 8bit PIXEL_FORMAT_YUV_SEMIPLANAR_444 = 5, // YUV444SP 8bit PIXEL_FORMAT_YVU_SEMIPLANAR_444 = 6, // YVU444SP 8bit PIXEL_FORMAT_YUYV_PACKED_422 = 7, // YUV422Packed YUYV 8bit PIXEL_FORMAT_UYVY_PACKED_422 = 8, // YUV422Packed UYVY 8bit PIXEL_FORMAT_YVYU_PACKED_422 = 9, // YUV422Packed YVYU 8bit PIXEL_FORMAT_VYUY_PACKED_422 = 10, // YUV422Packed VYUY 8bit</pre> - 关于输出图片分辨率、图片格式请参见图片格式、宽高对齐、内存约束。 - 关于各图片格式的宽高对齐、内存大小约束，请参见图片格式、宽高对齐、内存约束。 - 当输入、输出图片格式都为ARGB8888、ABGR8888、RGBA8888、BGRA8888格式时，在经过本接口缩放处理后，A分量值会变为0。 - 使用本接口实现等比例缩放功能时，相关的描述及约束请参见抠图、贴图约束。 - 由于YUV格式图像下采样约束，当输出图片格式为YUV420SP或YUV422SP格式时，需注意： - 对于YUV420SP输出格式，上下左右填充的尺寸建议为偶数。 - 对于YUV422SP输出格式，左右填充的尺寸建议为偶数。

版本	约束
Atlas A2训练系列产品/ Atlas 800I A2推理产品 Atlas 200I/500 A2推理 产品	- 关于图片分辨率、图片格式、宽高对齐、内存等总体约束，请参见图片格式、宽高对齐、内存约束。 - 当输入、输出图片格式都为ARGB8888、ABGR8888、RGBA8888、BGRA8888格式时，在经过本接口缩放处理后，A分量值会变为0。 - 使用本接口实现等比例缩放功能时，相关的描述及约束请参见抠图、贴图约束。 - 由于YUV格式图像下采样约束，当输出图片格式为YUV420SP或YUV422SP格式时，需注意： - 对于YUV420SP输出格式，上下左右填充的尺寸建议为偶数。 - 对于YUV422SP输出格式，左右填充的尺寸建议为偶数。

2.9.6.14 函数：dvpp_vpc_convert_color_async

C函数原型	aclError aclDvppVpcConvertColorAsync(aclDvppChannelDesc *channelDesc, aclDvppPicDesc *inputDesc, aclDvppPicDesc *outputDesc, aclrtStream stream)
Python函数	ret = acl.media.dvpp_vpc_convert_color_async(dvpp_channel_desc , input_dvpp_pic_desc, output_dvpp_pic_desc, stream)
函数功能	转换图片的格式，输出图片的宽高须与输入图片的宽高一致。异步接口。 Atlas 200/300/500 推理产品 ，当前版本不支持该接口。 Atlas 训练系列产品 ，当前版本不支持该接口。

输入说明	dvpp_channel_desc: int, 指定通道描述信息的指针地址, 与调用acl.media.dvpp_create_channel接口创建通道时指定的“dvpp_channel_desc”保持一致。 input_dvpp_pic_desc: int, 输入图片信息的指针地址。 - 调用acl.media.dvpp_create_pic_desc接口创建图片描述信息。 - 调用acl.media.dvpp_set_pic_desc系列接口设置图片描述参数（例如，内存地址、内存大小、图片格式、宽、高等）。 output_dvpp_pic_desc: int, 输出图片信息的指针地址。 作为输入时，需要用户调用如下接口： - 调用acl.media.dvpp_create_pic_desc接口创建图片描述信息。 - 调用acl.media.dvpp_set_pic_desc系列接口设置图片描述参数（例如，内存地址、内存大小、图片格式、宽、高等）。需指定输出图片的宽、高、宽Stride、高Stride、内存地址、内存大小、格式等。 - 配置输出图片的宽、高时，需与输入图片的宽、高保持一致，否则会返回报错。 如果将输出图片的宽或高配置为0，则VPC内部会将输入图片的宽、高分别作为输出图片的宽、高；同时，VPC内部会按对齐要求计算宽Stride、高Stride，不同图片格式的对齐要求不同，请参见图片格式、宽高对齐、内存约束。 - 配置输出图片的宽Stride、高Stride时，满足对齐要求即可，不同图片格式的对齐要求不同，请参见图片格式、宽高对齐、内存约束。 stream: int, 指定Stream对象。
返回值说明	ret: int, 错误码。 - 返回0表示任务下发成功。 - 返回非0表示任务下发失败。
约束说明	- 输入、输出图片分辨率为10 * 6 ~ 4096 * 4096（包括4096）。 - 输入、输出图片的格式请参见图片格式、宽高对齐、内存约束。 Atlas 推理系列产品上，当输入格式是YUV400时，输出格式只能是YUV400。
参考资源	接口调用示例，参见格式转换示例代码。

2.9.6.15 函数：dvpp_vpc_pyr_down_async

C函数原型	<pre>aclError aclDvppVpcPyrDownAsync(aclDvppChannelDesc *channelDesc, aclDvppPicDesc *inputDesc, aclDvppPicDesc *outputDesc, void *reserve, aclrtStream stream)</pre>

Python函数	ret = acl.media.dvpp_vpc_pyr_down_async(dvpp_channel_desc, input_dvpp_pic_desc, output_dvpp_pic_desc, reserve, stream)
函数功能	对图像进行金字塔缩放，将输出图片缩放为输入图片的一半。异步接口。 Atlas 200/300/500 推理产品 ，不支持该接口。 Atlas 训练系列产品 ，不支持该接口。 Atlas 200I/500 A2推理产品 ，不支持该接口。 Atlas A2训练系列产品/Atlas 800I A2推理产品 ，不支持该接口。
输入说明	dvpp_channel_desc : int, 指定通道描述信息的指针地址，与调用 acl.media.dvpp_create_channel 接口创建通道时指定的“dvpp_channel_desc”保持一致。 input_dvpp_pic_desc : int, 输入图片信息的指针地址。 - 调用acl.media.dvpp_create_pic_desc接口创建图片描述信息。 - 调用acl.media.dvpp_set_pic_desc系列接口设置图片描述参数（例如，内存地址、内存大小、图片格式、宽、高等）。 output_dvpp_pic_desc : int, 输出图片信息的指针地址。 作为输入时，需要用户调用如下接口： - 调用acl.media.dvpp_create_pic_desc接口创建图片描述信息。 - 调用acl.media.dvpp_set_pic_desc系列接口设置图片描述参数（例如，内存地址、内存大小、图片格式、宽、高等）。 reserve : int, 金字塔缩放配置参数的指针地址，预留参数。 stream : int, 指定Stream。
返回值说明	ret : int, 错误码。 - 返回0，表示任务下发成功。 - 返回非0，表示任务下发失败。
约束说明	- 关于输入图片 - 输入图片分辨率：20 * 12 ~ 2048 * 2048 - 输入图片格式：PIXEL_FORMAT_YUV_400 - 关于输出图片 - 输出图像分辨率：输入图像分辨率 / 2 - 图像格式：PIXEL_FORMAT_YUV_400 “PIXEL_FORMAT_YUV_400”图片格式的对齐要求请参见图片格式、宽高对齐、内存约束。

2.9.6.16 函数：dvpp_create_roi_config

C函数原型	<code>acldvppRoiConfig *acldvppCreateRoiConfig(uint32_t left, uint32_t right, uint32_t top, uint32_t bottom)</code>
Python函数	<code>dvpp_roi_config = acl.media.dvpp_create_roi_config(left, right, top, bottom)</code>
函数功能	创建acldvppRoiConfig类型的数据，用于描述某个区域位置的数据。 如需销毁acldvppRoiConfig类型的数据，请参见acl.media.dvpp.destroy_roi_config。
输入说明	left: int, 左偏移。 - 在Atlas 200/300/500 推理产品上，该参数值必须为偶数。如果用做贴图区域，左偏移需要16对齐。 - 在Atlas 训练系列产品上，该参数值必须为偶数。如果用做贴图区域，左偏移需要16对齐。 - 在Atlas 推理系列产品上，该参数值无偶数要求。如果用做贴图区域，左偏移需要16对齐。 - 在Atlas A2训练系列产品/Atlas 800I A2推理产品上，该参数值无偶数要求、无对齐要求。 right: int, 右偏移。 - 在Atlas 200/300/500 推理产品上，该参数值必须为奇数。 - 在Atlas 训练系列产品上，该参数值必须为奇数。 - 在Atlas 推理系列产品上，该参数值无奇数要求。 - 在Atlas A2训练系列产品/Atlas 800I A2推理产品上，该参数值无奇数要求。 top: int, 上偏移。 - 在Atlas 200/300/500 推理产品上，该参数值必须为偶数。 - 在Atlas 训练系列产品上，该参数值必须为偶数。 - 在Atlas 推理系列产品上，该参数值无偶数要求。 - 在Atlas A2训练系列产品/Atlas 800I A2推理产品上，该参数值无偶数要求。 bottom: int, 下偏移。 - 在Atlas 200/300/500 推理产品上，该参数值必须为奇数。 - 在Atlas 训练系列产品上，该参数值必须为奇数。 - 在Atlas 推理系列产品上，该参数值无奇数要求。 - 在Atlas A2训练系列产品/Atlas 800I A2推理产品上，该参数值无奇数要求。
返回值说明	dvpp_roi_config : int, acldvppRoiConfig类型数据的指针地址。 - 返回非0值，表示成功。 - 返回0，表示失败。

2.9.6.17 函数：dvpp_destroy_roi_config

C函数原型	<code>aclError aclDvppDestroyRoiConfig(aclDvppRoiConfig *roiConfig)</code>
Python函数	<code>ret = acl.media.dvpp_destroy_roi_config(roi_config)</code>
函数功能	销毁通过 acl.media.dvpp_create_roi_config 接口创建的数据。
输入说明	<code>roi_config</code> : int, 指定待销毁的数据的指针地址。
返回值说明	<code>ret</code> : int, 错误码。 • 返回0表示成功。 • 返回其它值表示失败。

2.9.6.18 函数：dvpp_set_roi_config

C函数原型	<code>aclError aclDvppSetRoiConfig(aclDvppRoiConfig *config, uint32_t left, uint32_t right, uint32_t top, uint32_t bottom)</code>
Python函数	<code>ret = acl.media.dvpp_set_roi_config(roi_config, left, right, top, bottom)</code>
函数功能	设置某个区域的位置信息。

输入说明	roi_config: int, 指定区域位置数据的指针地址。需提前调用acl.media.dvpp_create_roi_config接口创建的数据。 left: int, 左偏移。 - 在Atlas 200/300/500 推理产品上, 该参数值必须为偶数。如果用做贴图区域, 左偏移需要16对齐。 - 在Atlas 训练系列产品上, 该参数值必须为偶数。如果用做贴图区域, 左偏移需要16对齐。 - 在Atlas 推理系列产品上, 该参数值无偶数要求。如果用做贴图区域, 左偏移需要16对齐。 - 在Atlas A2训练系列产品/Atlas 800I A2推理产品上, 该参数值无偶数要求、无对齐要求。 right: int, 右偏移。 - 在Atlas 200/300/500 推理产品上, 该参数值必须为奇数。 - 在Atlas 训练系列产品上, 该参数值必须为奇数。 - 在Atlas 推理系列产品上, 该参数值无奇数要求。 - 在Atlas A2训练系列产品/Atlas 800I A2推理产品上, 该参数值无奇数要求。 top: int, 上偏移。 - 在Atlas 200/300/500 推理产品上, 该参数值必须为偶数。 - 在Atlas 训练系列产品上, 该参数值必须为偶数。 - 在Atlas 推理系列产品上, 该参数值无偶数要求。 - 在Atlas A2训练系列产品/Atlas 800I A2推理产品上, 该参数值无偶数要求。 bottom: int, 下偏移。 - 在Atlas 200/300/500 推理产品上, 该参数值必须为奇数。 - 在Atlas 训练系列产品上, 该参数值必须为奇数。 - 在Atlas 推理系列产品上, 该参数值无奇数要求。 - 在Atlas A2训练系列产品/Atlas 800I A2推理产品上, 该参数值无奇数要求。
返回值说明	ret: int, 错误码。 - 返回0表示成功。 - 返回其它值表示失败。

2.9.6.19 函数: dvpp_set_roi_config_left

C函数原型	<code>aclError aclDvppSetRoiConfigLeft(aclDvppRoiConfig *config, uint32_t left)</code>
Python函数	<code>ret = acl.media.dvpp_set_roi_config_left(roi_config, left)</code>
函数功能	设置某个区域的位置信息: 左偏移。

输入说明	roi_config: int, 描述某个区域位置数据的指针地址。需提前调用acl.media.dvpp_create_roi_config接口创建的数据。 left: int, 左偏移。 - 在Atlas 200/300/500 推理产品上, 该参数值必须为偶数。如果用做贴图区域, 左偏移需要16对齐。 - 在Atlas 训练系列产品上, 该参数值必须为偶数。如果用做贴图区域, 左偏移需要16对齐。 - 在Atlas 推理系列产品上, 该参数值无偶数要求。如果用做贴图区域, 左偏移需要16对齐。 - 在Atlas A2训练系列产品/Atlas 800I A2推理产品上, 该参数值无偶数要求、无对齐要求。
返回值说明	ret: int, 错误码。 - 返回0表示成功。 - 返回其它值表示失败。

2.9.6.20 函数: dvpp_set_roi_config_right

C函数原型	aclError aclDvppSetRoiConfigRight (aclDvppRoiConfig *config, uint32_t right)
Python函数	ret = acl.media.dvpp_set_roi_config_right (roi_config, right)
函数功能	设置某个区域的位置信息: 右偏移。
输入说明	roi_config: int, 描述某个区域位置数据的指针地址。需提前调用acl.media.dvpp_create_roi_config接口创建的数据。 right: int, 右偏移。 - 在Atlas 200/300/500 推理产品上, 该参数值必须为奇数。 - 在Atlas 训练系列产品上, 该参数值必须为奇数。 - 在Atlas 推理系列产品上, 该参数值无奇数要求。 - 在Atlas A2训练系列产品/Atlas 800I A2推理产品上, 该参数值无奇数要求。
返回值说明	ret: int, 错误码。 - 返回0表示成功。 - 返回其它值表示失败。

2.9.6.21 函数: dvpp_set_roi_config_top

C函数原型	aclError aclDvppSetRoiConfigTop(aclDvppRoiConfig *config, uint32_t top)
Python函数	ret = acl.media.dvpp_set_roi_config_top(roi_config, top)

函数功能	设置某个区域的位置信息：上偏移。
输入说明	roi_config : int, 描述某个区域位置数据的指针地址。需提前调用 acl.media.dvpp_create_roi_config 接口创建的数据。 top : int, 上偏移。 - 在Atlas 200/300/500 推理产品上, 该参数值必须为偶数。 - 在Atlas 训练系列产品上, 该参数值必须为偶数。 - 在Atlas 推理系列产品上, 该参数值无偶数要求。 - 在Atlas A2训练系列产品/Atlas 800I A2推理产品上, 该参数值无偶数要求。
返回值说明	ret : int, 错误码。 - 返回0表示成功。 - 返回其它值表示失败。

2.9.6.22 函数：dvpp_set_roi_config_bottom

C函数原型	aclError aclDvppSetRoiConfigBottom(aclDvppRoiConfig *config, uint32_t bottom)
Python函数	ret = acl.media.dvpp_set_roi_config_bottom(roi_config, bottom)
函数功能	设置某个区域的位置信息：下偏移。
输入说明	roi_config : int, 描述某个区域位置数据的指针地址。需提前调用 acl.media.dvpp_create_roi_config 接口创建的数据。 bottom : int, 下偏移。 - 在Atlas 200/300/500 推理产品上, 该参数值必须为奇数。 - 在Atlas 训练系列产品上, 该参数值必须为奇数。 - 在Atlas 推理系列产品上, 该参数值无奇数要求。 - 在Atlas A2训练系列产品/Atlas 800I A2推理产品上, 该参数值无奇数要求。
返回值说明	ret : int, 错误码。 - 返回0表示成功。 - 返回其它值表示失败。

2.9.6.23 函数：dvpp_create_resize_config

C函数原型	aclDvppResizeConfig *aclDvppCreateResizeConfig()
Python函数	dvpp_resize_config = acl.media.dvpp_create_resize_config()

函数功能	创建acldvppResizeConfig类型的数据，表示图片缩放配置数据。 如需销毁acldvppResizeConfig类型的数据，请参见acl.media.dvpp_destroy_resize_config。 Atlas 200/300/500 推理产品，默认缩放算法为“华为自研的高阶滤波算法”。 Atlas 训练系列产品，默认缩放算法为“华为自研的高阶滤波算法”。 Atlas 推理系列产品，默认缩放算法为“业界通用的Bilinear算法”。 Atlas A2训练系列产品/Atlas 800I A2推理产品，默认缩放算法为“业界通用的Bilinear算法”。 Atlas 200I/500 A2推理产品，默认缩放算法为“业界通用的Bilinear算法”。
返回值说明	dvpp_resize_config: int，返回acldvppResizeConfig类型数据的指针地址。 • 返回非0值，表示成功。 • 返回0，表示失败。

2.9.6.24 函数：dvpp_destroy_resize_config

C函数原型	aclError acldvppDestroyResizeConfig(acldvppResizeConfig *resizeConfig)
Python函数	ret = acl.media.dvpp_destroy_resize_config(dvpp_resize_config)
函数功能	销毁通过 acl.media.dvpp_create_resize_config 接口创建的缩放配置数据。
输入说明	dvpp_resize_config : int，指定待销毁的acldvppResizeConfig类型数据的指针地址。
返回值说明	ret : int，错误码。 • 返回0表示成功。 • 返回其它值表示失败。

2.9.6.25 函数：dvpp_set_resize_config_interpolation

C函数原型	aclError acldvppSetResizeConfigInterpolation(acldvppResizeConfig *resizeConfig, uint32_t interpolation)
Python函数	ret = acl.media.dvpp_set_resize_config_interpolation(dvpp_resize_config, interpolation)

函数功能	设置图片缩放算法。

输入说明	dvpp_resize_config: int, 指定待设置的缩放配置数据的指针地址。需提前调用acl.media.dvpp_create_resize_config接口创建aclDvppResizeConfig类型的数据。 interpolation: int, 指定缩放算法。此处配置的缩放算法建议与训练模型时的缩放算法保持一致。 Atlas 200/300/500 推理产品, 支持如下缩放算法: • 0: 默认值, 表示华为自研的高阶滤波算法。 • 1: 业界通用的Bilinear算法 (与OpenCV算法的计算过程类似)。 • 2: 业界通用的Nearest neighbor 算法 (与OpenCV算法的计算过程类似)。 • 3: 业界通用的Bilinear算法 (与Tensorflow算法的计算过程类似)。 • 4: 业界通用的Nearest neighbor算法 (与Tensorflow算法的计算过程类似)。 Atlas 训练系列产品, 支持如下缩放算法: • 0: 默认值, 表示华为自研的高阶滤波算法。 • 1: 业界通用的Bilinear算法 (与OpenCV算法的计算过程类似)。 • 2: 业界通用的Nearest neighbor 算法 (与OpenCV算法的计算过程类似)。 • 3: 业界通用的Bilinear算法 (与Tensorflow算法的计算过程类似)。 • 4: 业界通用的Nearest neighbor算法 (与Tensorflow算法的计算过程类似)。 Atlas 推理系列产品, 支持如下缩放算法: • 0: 设置为0时, 系统内部也会自动采用1。 • 1: 业界通用的Bilinear算法 (与OpenCV算法的计算过程类似)。 设置该算法, 且当输入和输出图片格式都为RGB时, 在[1/32, 512]的缩放范围内, 与OpenCV的单个像素值最大差异为正负1。 • 2: 业界通用的Nearest neighbor 算法 (与OpenCV算法的计算过程类似)。 Atlas 200I/500 A2推理产品, 支持如下缩放算法: • 0: 默认值, 表示华为自研的高阶滤波算法。 • 1: 业界通用的Bilinear算法 (与OpenCV-3.4.2版本算法的计算结果相同)。 • 2: 业界通用的Nearest neighbor 算法 (与OpenCV-3.4.2版本算法的计算结果相同)。 • 3: 业界通用的Bilinear算法 (与1相同)。 • 4: 业界通用的Nearest neighbor算法 (与2相同)。 • 5: 华为自研的高阶滤波算法。

	当前该缩放算法仅支持输入图片分辨率范围为10 * 6 ~ 8192 * 8192、输出图片分辨率的范围为10 * 6 ~ 4096 * 8192。 输入图片格式仅支持YUV、但不包括Planner格式、YUV400与YUV440，输出图片格式仅支持YUV420SP NV12 8bit、YUV420SP NV21 8bit。 宽或高的缩放范围：[1/32, 16]。 Atlas A2训练系列产品/Atlas 800I A2推理产品，支持如下缩放算法： • 0：默认值，表示华为自研的高阶滤波算法。 • 1：业界通用的Bilinear算法（与OpenCV-3.4.2版本算法的计算结果相同）。 • 2：业界通用的Nearest neighbor 算法（与OpenCV-3.4.2版本算法的计算结果相同）。 • 3：业界通用的Bilinear算法（与1相同）。 • 4：业界通用的Nearest neighbor算法（与2相同）。 • 5：华为自研的高阶滤波算法。 当前该缩放算法仅支持输入图片分辨率范围为10 * 6 ~ 8192 * 8192、输出图片分辨率的范围为10 * 6 ~ 4096 * 8192。 输入图片格式仅支持YUV、但不包括Planner格式、YUV400与YUV440，输出图片格式仅支持YUV420SP NV12 8bit、YUV420SP NV21 8bit。 宽或高的缩放范围：[1/32, 16]。
返回值说明	ret: int，错误码。 • 返回0表示成功。 • 返回其它值表示失败。

2.9.6.26 函数：dvpp_get_resize_config_interpolation

C函数原型	uint32 aclDvppGetResizeConfigInterpolation(const aclDvppResizeConfig * resizeConfig)
Python函数	interpolation = acl.media.dvpp_get_resize_config_interpolation(dvpp_resize_config)
函数功能	获取图片缩放算法。
输入说明	dvpp_resize_config: int，指定待获取的缩放配置数据的指针地址。 需提前调用acl.media.dvpp_create_resize_config接口创建aclDvppResizeConfig类型的数据，调用acl.media.dvpp_set_resize_config_interpolation接口设置缩放算法。
返回值说明	interpolation: int，缩放算法，具体请参见acl.media.dvpp_create_resize_config。

2.9.7 JPEGD 功能

2.9.7.1 功能及约束说明

功能说明

JPEGD (JPEG Decoder) 实现.jpg、.jpeg、.JPG、.JPEG图片文件的解码。

- **JPEGD在解码图片时，支持对图片进行旋转。**
如果输入图片的码流中包含Orientation信息（代表捕获图像时摄像机相对于场景的方向），则JPEGD在解码时会解析Orientation信息，将图片进行90度、180度、270度或镜像旋转。旋转后输出图片的宽Stride、高Stride、输出内存仍需满足[图片格式、宽高对齐、内存约束](#)中的要求。
如果输入图片的码流异常，导致JPEGD解码时无法读取Orientation信息，则不能实现图片旋转的功能。
以下产品在使用图片旋转功能时，存在约束，请参见[表2-21](#)。

表 2-21 JPEGD 解码图片，旋转功能约束

产品版本	约束
Atlas 200/300/500 推理产品	多路并发解码JPEG图片场景下，如果解码的JPEG图片中，包含带旋转信息的大分辨率图片，例如3840 * 2160分辨率及以上的图片，则可能导致图片解码时间过长，从而导致用户进程超时退出。
Atlas 200I/500 A2推理产品 Atlas A2训练系列产品/Atlas 800I A2推理产品	● JPEGD 422格式旋转图片，若原图高为奇数，并且旋转方式为宽高对换，则旋转后宽存在黑边，建议用户将宽向下对齐到偶数使用（即去除黑边），比如原图为200 * 101，若旋转后为101 * 200，则建议用户实际使用区域为100 * 200。 ● JPEGD 440格式旋转图片，若原图宽为奇数，并且旋转方式为宽高对换，则旋转后高存在黑边，建议用户将高向下对齐到偶数使用（即去除黑边），比如原图为201 * 101，若旋转后为100 * 201，则建议用户实际使用区域为100 * 200。

- **JPEGD在解码图片时，支持按源图片格式解码。**
源图片格式解码是指解码前后图片的编码格式保持一致，例如解码前输入图片为jpeg(440)，解码后输出图片为YUV440SP V在前U在后或YUV440SP U在前V在后。
使用源图片格式解码，有以下方式：
 - 调用[acl.media.dvpp_jpeg_get_image_info_v2](#)接口，获取JPEGD解码前输入图片的编码格式，在调用JPEGD解码接口时，将输出图片格式设置的与输入图片编码格式一致。

- 在调用JPEG解码接口时，将输出图片格式配置为“PIXEL_FORMAT_UNKNOWN”，输出格式默认按源图片格式输出、且是V在前U在后的Semi-Planar格式。

例如，JPEG输入图片编码格式为jpeg(440)，输出图片格式配置为“PIXEL_FORMAT_UNKNOWN”，JPEG解码后，实际输出图片格式为YUV440SP V在前U在后。

 说明

JPEG解码后的输出图片，如果要直接作为模型推理的输入，建议将输出图片格式配置为“PIXEL_FORMAT_UNKNOWN”，此时JPEG使用源图片格式解码（但需要确保解码后的图片格式模型是支持的），保证模型推理的精度。

JPEG解码后的输出图片，如果直接作为VPC的输入，该场景下若使用源图片格式解码时，则需要关注解码后的输出图片格式VPC是否支持（VPC输入图片的格式请参见[2.9.6.2 约束说明](#)），如果VPC不支持，则用户需按VPC支持的情况指定JPEG的输出图片格式。

图片分辨率约束

- 输入图片分辨率

版本	分辨率范围
Atlas 200/300/500 推理产品 Atlas 训练系列产品 Atlas 推理系列产品	最大分辨率：8192 * 8192，最小分辨率：32 * 32。
Atlas A2训练系列产品/Atlas 800I A2推理产品 Atlas 200I/500 A2推理产品	最大分辨率：16384 * 16384，最小分辨率：32 * 32。

- 输出图片分辨率

JPEG只对图片解码，不会改变图片分辨率，因此输出与输入的图片分辨率保持一致。

图片格式、宽高对齐、内存约束

实现JPEG图片解码功能时，需调用[acl.media.dvpp_malloc](#)接口申请Device上的输入、输出内存，调用[acl.media.dvpp_free](#)接口释放输入、输出内存，这部分内存的生命周期由用户自行管理。

- 输入内存的大小就是指实际的输入图片所占用的大小。
- 输出内存的大小，可调用[acl.media.dvpp_jpeg_predict_dec_size](#)接口预估，计算公式参见下表。

实现JPEG图片解码功能时，仅支持Huffman编码，压缩前的原图像色彩空间为YUV，像素的各分量比例为4：4：4或4：2：2或4：2：0或4：0：0或4：4：0，不支持算术编码、不支持渐进JPEG格式、不支持JPEG2000格式。

 说明

输出图片格式的定义请参见[aclDvppPixelFormat](#)，宽Stride、高Stride等概念请参见[2.9.2 基本概念](#)。

表 2-22 图片格式、宽高对齐、内存大小约束

输入图片格式 (YUV 分量比例)	输出图片格式	输出图片宽、高对齐要求	输出图片宽stride、高stride、内存大小要求 Atlas 200/300/500 推理产品 Atlas 训练系列产品	输出图片宽stride、高stride、内存大小要求 Atlas 推理系列产品 Atlas 200I/500 A2推理产品 Atlas A2训练系列产品/Atlas 800I A2推理产品
jpeg(444)	YVU444SP 8bit	无对齐要求	宽stride为宽128对齐后的值。	宽stride为宽64对齐后的值。
	YUV444SP 8bit		高stride为高16对齐后的值。	高stride为高16对齐后的值。
	YUV420SP NV12 8bit	宽2对齐 高2对齐	内存大小 (单位 Byte) ≥ 宽stride * 高stride * 3	内存大小 (单位 Byte) ≥ 宽stride * 高stride * 3
	YUV420SP NV21 8bit		宽stride为宽128对齐后的值。	宽stride为宽64对齐后的值。
jpeg(422)	YVU422SP 8bit	宽2对齐 高2对齐	高stride为高16对齐后的值。	高stride为高16对齐后的值。
	YUV422SP 8bit		内存大小 (单位 Byte) ≥ 宽stride * 高stride * 3/2	内存大小 (单位 Byte) ≥ 宽stride * 高stride * 3/2
	YUV420SP NV12 8bit	宽2对齐 高2对齐	宽stride为宽128对齐后的值。	宽stride为宽64对齐后的值。
	YUV420SP NV21 8bit		高stride为高16对齐后的值。	高stride为高16对齐后的值。
			内存大小 (单位 Byte) ≥ 宽stride * 高stride * 2	内存大小 (单位 Byte) ≥ 宽stride * 高stride * 2
			宽stride为宽128对齐后的值。	宽stride为宽64对齐后的值。
			高stride为高16对齐后的值。	高stride为高16对齐后的值。
			内存大小 (单位 Byte) ≥ 宽stride * 高stride * 3/2	内存大小 (单位 Byte) ≥ 宽stride * 高stride * 3/2

输入图片格式 (YUV 分量比例)	输出图片格式	输出图片宽、高对齐要求	输出图片宽stride、高stride、内存大小要求 Atlas 200/300/500 推理产品 Atlas 训练系列产品	输出图片宽stride、高stride、内存大小要求 Atlas 推理系列产品 Atlas 200I/500 A2推理产品 Atlas A2训练系列产品/Atlas 800I A2推理产品
jpeg(420)	YUV420SP NV12 8bit	宽2对齐 高2对齐	宽stride为宽128对齐后的值。 高stride为高16对齐后的值。 内存大小 (单位 Byte) $\geq$ 宽stride * 高stride * 3/2	宽stride为宽64对齐后的值。 高stride为高16对齐后的值。 内存大小 (单位 Byte) $\geq$ 宽stride * 高stride * 3/2
	YUV420SP NV21 8bit			
jpeg(400)	YUV420SP NV12 8bit	宽2对齐 高2对齐	宽stride为宽128对齐后的值。 高stride为高16对齐后的值。 内存大小 (单位 Byte) $\geq$ 宽stride * 高stride * 3/2	宽stride为宽64对齐后的值。 高stride为高16对齐后的值。 内存大小 (单位 Byte) $\geq$ 宽stride * 高stride * 3/2
	YUV420SP NV21 8bit			
	YUV400 8bit Atlas 200/300/500 推理产品和 Atlas 训练系列产品不支持该格式	无对齐要求	宽stride为宽128对齐后的值。 高stride为高16对齐后的值。 内存大小 (单位 Byte) $\geq$ 宽stride * 高stride * 2	宽stride为宽64对齐后的值。 高stride为高16对齐后的值。 内存大小 (单位 Byte) $\geq$ 宽stride * 高stride
jpeg(440)	YVU440SP 8bit	宽2对齐 高2对齐	宽stride为宽128对齐后的值。 高stride为高16对齐后的值。 内存大小 (单位 Byte) $\geq$ 宽stride * 高stride * 2	宽stride为宽64对齐后的值。 高stride为高16对齐后的值。 内存大小 (单位 Byte) $\geq$ 宽stride * 高stride * 2
	YUV440SP 8bit			
	YUV420SP NV12 8bit	宽2对齐 高2对齐	宽stride为宽128对齐后的值。 高stride为高16对齐后的值。 内存大小 (单位 Byte) $\geq$ 宽stride * 高stride * 3/2	宽stride为宽64对齐后的值。 高stride为高16对齐后的值。 内存大小 (单位 Byte) $\geq$ 宽stride * 高stride * 3/2
	YUV420SP NV21 8bit			

软、硬件约束

- **硬件约束：**
 - 最多支持4张Huffman表，其中包括2张DC（直流）表和2张AC（交流）表。
 - 最多支持3张量化表。
 - 只支持8bit采样精度。
 - 只支持对顺序式编码的图片进行解码。
 - 只支持基于DCT（Discrete Cosine Transform）变换的JPEG格式解码。
 - 只支持一个SOS（Start of Scan）标志的图片解码。
- **软件约束：**
 - 支持3个SOS标志的图片解码。
 - 支持mcu（Minimum Coded Unit）数据不足的异常图片解码。

精度相关约束

在**Atlas 200/300/500 推理产品**上，JPEG+VPC串联使用时，由于JPEG解码后的输出图片的宽 $Stride * 高Stride$ 有 $128 * 16$ 对齐的约束，因此解码后的输出图片的宽、高有一些补边的无效数据，所以在执行VPC功能时（例如缩放时），需调用和接口正确设置输入图片的原图宽高，这样VPC在缩放图片前会先根据原图宽高自行抠图，目的是去除无效数据对图像精度的影响。

在**Atlas 训练系列产品**上，JPEG+VPC串联使用时，由于JPEG解码后的输出图片的宽 $Stride * 高Stride$ 有 $128 * 16$ 对齐的约束，因此解码后的输出图片的宽、高有一些补边的无效数据，所以在执行VPC功能时（例如缩放时），需调用和接口正确设置输入图片的原图宽高，这样VPC在缩放图片前会先根据原图宽高自行抠图，目的是去除无效数据对图像精度的影响。

在**Atlas 200I/500 A2推理产品**上，JPEG+VPC串联使用时，由于JPEG解码后的输出图片的宽 $Stride * 高Stride$ 有 $64 * 16$ 对齐的约束，因此解码后的输出图片的宽、高有一些补边的无效数据，所以在执行VPC功能时（例如缩放时），需调用和接口正确设置输入图片的原图宽高，这样VPC在缩放图片前会先根据原图宽高自行抠图，目的是去除无效数据对图像精度的影响。

在**Atlas 推理系列产品**上，JPEG+VPC串联使用时，由于JPEG解码后的输出图片的宽 $Stride * 高Stride$ 有 $64 * 16$ 对齐的约束，因此解码后的输出图片的宽、高有一些补边的无效数据，所以在执行VPC功能时（例如缩放时），需调用和接口正确设置输入图片的原图宽高，这样VPC在缩放图片前会先根据原图宽高自行抠图，目的是去除无效数据对图像精度的影响。

在**Atlas A2训练系列产品/Atlas 800I A2推理产品**上，JPEG+VPC串联使用时，由于JPEG解码后的输出图片的宽 $Stride * 高Stride$ 有 $64 * 16$ 对齐的约束，因此解码后的输出图片的宽、高有一些补边的无效数据，所以在执行VPC功能时（例如缩放时），需调用和接口正确设置输入图片的原图宽高，这样VPC在缩放图片前会先根据原图宽高自行抠图，目的是去除无效数据对图像精度的影响。

2.9.7.2 性能指标说明

本节仅描述各产品对应的昇腾AI处理器的性能指标参考数据，各产品由于其内存、CPU等配置不同，在性能规格上会存在差异，详细产品的性能规格请查阅[昇腾社区](#)的官方说明。

本节以单个昇腾AI处理器为例，描述基本场景性能指标参考，其它分辨率可以等量估算。

下表中的1路对应一个通道，一个通道对应一个线程；各分辨率的含义为：1080p指分辨率为1920 * 1080的图片，4K指分辨率为3840 * 2160的图片。

PEGD性能指标是基于硬件解码的性能，JPEG硬件解码不支持3个SOS的图片解码，对于硬件不支持的格式，会使用软件解码，软件解码性能参考为1080P 15fps。JPEG解码的输出图片如果涉及旋转，则性能指标低于软件解码的参考值，例如对于1080P的图片，性能指标低于15fps。

性能指标说明（Atlas 200/300/500 推理产品）

以下性能数据，是基于一个Stream上下发1个异步媒体数据处理任务后，执行一次[acl.rt.synchronize_stream](#)接口。

场景举例	总帧率
1080p * 1路	128fps
1080p * n路 ($n \geq 2$)	256fps
4k * 1路	32fps
4k * n路 ($n \geq 2$)	64fps

性能指标说明（Atlas 推理系列产品）

以下性能数据，是基于一个stream上下发10个异步媒体数据处理任务后，执行一次[acl.rt.synchronize_stream](#)接口，且Host服务器配置为：CPU主频为Intel(R) Xeon(R) CPU E5-2680 v4 @ 2.40GHz，CPU核心数为14。

场景举例	总帧率
1080p * n路 ($1 \leq n \leq 16$)	$n * 128\text{fps}$
1080p * n路 ($n > 16$)	2048fps
4k * n路 ($1 \leq n \leq 16$)	$n * 32\text{fps}$
4k * n路 ($n > 16$)	512fps

性能指标说明（Atlas 训练系列产品）

以下性能数据，是基于一个stream上下发1个异步媒体数据处理任务后，执行一次[acl.rt.synchronize_stream](#)接口。

场景举例	总帧率
1080p * n路 ($1 \leq n \leq 16$)	$n * 128\text{fps}$
1080p * n路 ($n > 16$)	2048fps
4k * n路 ($1 \leq n \leq 16$)	$n * 32\text{fps}$
4k * n路 ($n > 16$)	512fps

性能指标说明（Atlas 200I/500 A2 推理产品）

以下性能数据，是基于一个stream上下发10个异步媒体数据处理任务后，执行一次[acl.rt.synchronize_stream](#)接口。

场景举例	总帧率
1080p * n路 ($1 \leq n \leq 2$)	$n * 256\text{fps}$
1080p * n路 ($n > 2$)	512fps
4k * n路 ($1 \leq n \leq 2$)	$n * 64\text{fps}$
4k * n路 ($n > 2$)	128fps

性能指标数据（Atlas A2 训练系列产品/Atlas 800I A2 推理产品）

以下性能数据，是基于一个stream上下发10个异步媒体数据处理任务后，执行一次[acl.rt.synchronize_stream](#)接口。

输入图片格式	输入图片文件大小	场景举例	总帧率
YUV400/420	$\leq 500\text{KB}$	1080p * n路 ($1 \leq n \leq 28$)	$n * 512\text{fps}$
		1080p * n路 ($n > 28$)	14336fps
YUV440/422/444	$\leq 500\text{KB}$	1080p * n路 ($1 \leq n \leq 28$)	$n * 256\text{fps}$

输入图片格式	输入图片文件大小	场景举例	总帧率
		1080p * n路 (n > 28)	7168fps
YUV400/420	≤ 2MB	4k * n路 (1 ≤ n ≤ 28)	n * 128fps
		4k * n路 (n > 28)	3584fps
YUV440/422/444	≤ 2MB	4k * n路 (1 ≤ n ≤ 28)	n * 64fps
		4k * n路 (n > 28)	1792fps

2.9.7.3 函数：dvpp_jpeg_decode_async

C函数原型	<code>aclError aclDvppJpegDecodeAsync(aclDvppChannelDesc *channelDesc, const void *data, uint32_t size, aclDvppPicDesc *outputDesc, aclrtStream stream)</code>
Python函数	<code>ret = acl.media.dvpp_jpeg_decode_async(dvpp_channel_desc, data, size, output_dvpp_pic_desc, stream)</code>
函数功能	解码.jpg、.jpeg、.JPG、.JPEG图片，异步接口。

输入说明	dvpp_channel_desc: int, 指定通道描述信息的指针地址, 与调用acl.media.dvpp_create_channel接口创建通道时指定的dvpp_channel_desc保持一致。 data: int, 输入图片的内存地址的指针地址。 size: int, 输入图片的实际数据大小, 单位Byte。 output_dvpp_pic_desc: int, 输出图片信息的指针地址。 - 调用acl.media.dvpp_create_pic_desc接口创建图片描述信息。 - 调用acl.media.dvpp_set_pic_desc系列接口设置输出图片的内存地址、内存大小、格式、width_stride、height_stride。 - 输出图片的内存大小可提前调用acl.media.dvpp_jpeg_predict_dec_size接口预估。 - JPEG原图的宽、高可通过acl.media.dvpp_jpeg_get_image_info/acl.media.dvpp_jpeg_get_image_info_v2接口获取。 - 输出图片格式、输出图片的宽高对齐要求, 请参见2.9.7.1 功能及约束说明。 作为输出时, 用户可以从内存地址中获取解码后的输出图片数据、可以调用acl.media.dvpp_get_pic_desc系列接口获取输出图片的宽/高。 说明 Atlas 推理系列产品上, 如果解码后的输出图片数据需要在后续操作 (例如, 使用VPC实现抠图、缩放等功能) 中使用, 建议在解码后调用acl.media.dvpp_get_pic_desc获取输出图片的宽/高, 因为解码过程中可能会对输出图片的宽高执行对齐操作。 例如, jpeg(444) 源码的图片解码成YUV420SP格式的输出图片时, 当jpeg(444) 源码图片的宽/高为奇数时, 解码出来的YUV420SP格式的输出图片的宽/高理论上应该为奇数, 但是YUV420SP格式本身要求图片的宽/高都为偶数, 这时JPEGD会对奇数宽/高做向下2对齐的操作。 Atlas 200I/500 A2推理产品上, 如果解码后的输出图片数据需要在后续操作 (例如, 使用VPC实现抠图、缩放等功能) 中使用, 建议在解码后调用acl.media.dvpp_get_pic_desc获取输出图片的宽/高, 因为解码过程中可能会对输出图片的宽高执行对齐操作。 例如, jpeg(444) 源码的图片解码成YUV420SP格式的输出图片时, 当jpeg(444) 源码图片的宽/高为奇数时, 解码出来的YUV420SP格式的输出图片的宽/高理论上应该为奇数, 但是YUV420SP格式本身要求图片的宽/高都为偶数, 这时JPEGD会对奇数宽/高做向下2对齐的操作。 Atlas A2训练系列产品/Atlas 800I A2推理产品上, 如果解码后的输出图片数据需要在后续操作 (例如, 使用VPC实现抠图、缩放等功能) 中使用, 建议在解码后调用acl.media.dvpp_get_pic_desc获取输出图片的宽/高, 因为解码过程中可能会对输出图片的宽高执行对齐操作。 例如, jpeg(444) 源码的图片解码成YUV420SP格式的输出图片时, 当jpeg(444) 源码图片的宽/高为奇数时, 解码出来的YUV420SP格式的输出图片的宽/高理论上应该为奇数, 但是YUV420SP格式本身要求图片的宽/高都为偶数, 这时JPEGD会对奇数宽/高做向下2对齐的操作。 stream: int, 指定Stream对象。
返回值说明	ret: int, 错误码。 - 返回0, 表示任务下发成功。 - 返回非0, 表示任务下发失败。

约束说明	本接口是异步接口，调用接口成功仅表示任务下发成功，不表示任务执行成功。调用该接口后，需调用同步等待接口（例如， acl.rt.synchronize_stream ）确保任务已执行完成。
参考资源	接口调用流程及示例，参见JPEGD图片解码。

2.9.7.4 函数：dvpp_jpeg_get_image_info

C函数原型	<code>aclError aclDvppJpegGetImageInfo(const void *data, uint32_t size, uint32_t *width, uint32_t *height, int32_t *components)</code>
Python函数	<code>width, height, components, ret = acl.media.dvpp_jpeg_get_image_info(data, size)</code>
函数功能	从存放JPEG图片数据的内存中读取JPEG图片的宽、高、颜色通道个数。 该接口会检查图片格式是否支持解码，如果遇到不支持的码流则该接口会返回ACL_ERROR_UNSUPPORTED_JPEG错误码，该错误码的详细描述请参见 2.16.1 aclError 。
输入说明	data : int，存放JPEG图片数据的内存地址。 size : int，内存大小，单位为Byte。
返回值说明	width : int，图片的宽的指针地址。 height : int，图片的高的指针地址。 components : int，颜色通道个数的指针地址。 ret : int，错误码。 - 返回0表示成功。 - 返回其它值表示失败。

2.9.7.5 函数：dvpp_jpeg_get_image_info_v2

C函数原型	<code>aclError aclDvppJpegGetImageInfoV2(const void *data, uint32_t size, uint32_t *width, uint32_t *height, int32_t *components, aclDvppJpegFormat *format)</code>
Python函数	<code>width, height, components, format, ret = acl.media.dvpp_jpeg_get_image_info_v2(data, size)</code>
函数功能	从存放JPEG图片数据的内存中读取JPEG图片的宽、高、颜色通道个数、JPEG原图编码格式。 该接口会检查图片格式是否支持解码，如果遇到不支持的码流则该接口会返回ACL_ERROR_UNSUPPORTED_JPEG错误码，该错误码的详细描述请参见 2.16.1 aclError 。

输入说明	data : int, 存放JPEG图片数据的内存地址的指针地址。 size : int, 内存大小, 单位为Byte。
返回值说明	width : int, 图片的宽的指针地址。 height : int, 图片的高的指针地址。 components : int, 颜色通道个数的指针地址。 format : int, JPEG原图编码格式的指针地址。 ret : int, 错误码。 • 返回0表示成功。 • 返回其它值表示失败。

2.9.7.6 函数: dvpp_jpeg_predict_dec_size

C函数原型	<code>aclError aclDvppJpegPredictDecSize(const void *data, uint32_t dataSize, aclDvppPixelFormat outputPixelFormat, uint32_t *decSize)</code>
Python函数	<code>dec_size, ret = acl.media.dvpp_jpeg_predict_dec_size(data, data_size, format)</code>
函数功能	根据存放JPEG图片数据的内存预估JPEG图片解码后所需的输出内存的大小。
输入说明	data : int, Host上存放JPEG图片数据的内存地址的指针地址。 data_size : int, 内存大小, 单位为Byte。 format : int, 解码后的输出图片的格式。具体请参见 2.16.6 aclDvppPixelFormat 。
返回值说明	dec_size : int, 预估JPEG图片解码后所需的输出内存的大小的指针地址, 单位为Byte。 ret : int, 错误码。 • 返回0表示成功。 • 返回其它值表示失败。
参考资源	接口调用流程及示例, 参见JPEG图片解码。

2.9.8 JPEG 功能

2.9.8.1 功能及约束说明

功能说明

JPEG (JPEG Encoder) 将YUV格式图片编码成JPEG压缩格式的图片文件, 例如 *.jpg。

图片分辨率约束

- 输入图片分辨率：

版本	分辨率范围
Atlas 200/300/500 推理产品 Atlas 训练系列产品 Atlas 推理系列产品 Atlas A2训练系列产品/Atlas 800I A2推理产品	最大分辨率：8192 * 8192，最小分辨率：32 * 32。
Atlas 200I/500 A2推理产品	最大分辨率：16384 * 16384，最小分辨率：32 * 32。
- 输出图片分辨率
JPEGE只对图片编码，不会改变图片分辨率，因此输出与输入的图片分辨率保持一致。

图片格式、宽高对齐、内存约束

- 实现JPEGE图片编码功能时，需调用[acl.media.dvpp_malloc](#)接口申请Device上的输入、输出内存，调用[acl.media.dvpp_free](#)接口释放输入、输出内存，这部分内存的生命周期由用户自行管理。
- 输入内存的大小请参见下表中的计算公式。
 - 输出内存的大小就是指实际的编码后图片所占用的大小，可调用[acl.media.dvpp_jpeg_predict_enc_size](#)接口预估输出内存大小。

📖 说明

输入图片格式的定义请参见[aclDvppPixelFormat](#)，宽Stride、高Stride等概念请参见[2.9.2 基本概念](#)。

表 2-23 图片格式、宽高对齐、内存大小约束

输入图片格式	输入图片宽、高对齐要求	输入图片宽Stride、高Stride、内存大小要求	输出图片格式
YUV420SP NV12 8bit	宽2对齐 高2对齐	宽Stride为宽16对齐后的值，兼容对齐到16的倍数如128（按128对齐时，性能更优）。 高Stride支持以下两种配置： - 配置为与输入图片的高度相同的数值。 - 或配置为输入图片的高度向上对齐到16的数值（最小为32）。 该方式的使用场景举例：JPEGD的输出图片直接作为JPEGE的输入（JPEGD输出图片高度是向上对齐到16的）。 内存大小（单位Byte）= 宽Stride * 高Stride * 3/2	JPEG压缩格式的图片文件，例如*.jpg。 只支持huffman编码，不支持算术编码，不支持渐进编码。
YUV420SP NV21 8bit			
YUV422Packed YUYV 8bit	宽2对齐 高无对齐要求	宽Stride为宽的两倍再16对齐后的值。 高Stride支持以下两种配置： - 配置为与输入图片的高度相同的数值。 - 或配置为输入图片的高度向上对齐到16的数值（最小为32）。 该方式的使用场景举例：JPEGD的输出图片直接作为JPEGE的输入（JPEGD输出图片高度是向上对齐到16的）。 内存大小（单位Byte）= 宽Stride * 高Stride	
YUV422Packed UYVY 8bit			
YUV422Packed VYU 8bit			
YUV422Packed VYUY 8bit			

2.9.8.2 性能指标说明

本节仅描述各产品对应的昇腾AI处理器的性能指标参考数据，各产品由于其内存、CPU等配置不同，在性能规格上会存在差异，详细产品的性能规格请查阅[昇腾社区](#)的官方说明。

本节以单个昇腾AI处理器为例，描述基本场景性能指标参考，其它分辨率可以等量估算。

下表中的1路对应一个通道，一个通道对应一个线程；各分辨率的含义为：1080p指分辨率为1920 * 1080的图片，4K指分辨率为3840 * 2160的图片。

性能指标说明（Atlas 200/300/500 推理产品）

以下性能数据，是基于一个Stream上下发10个异步媒体数据处理任务后，执行一次[acl.rt.synchronize_stream](#)接口。

场景举例	总帧率
1080p * n路 ($n \geq 1$)	64fps
4k * n路 ($n \geq 1$)	16fps

性能指标说明（Atlas 推理系列产品）

以下性能数据，是基于一个Stream上下发10个异步媒体数据处理任务后，执行一次[acl.rt.synchronize_stream](#)接口，且Host服务器配置为：CPU主频为Intel(R) Xeon(R) CPU E5-2680 v4 @ 2.40GHz，CPU核心数为14。

场景举例（输入图片格式：YUV420 8bit）	总帧率
1080p * n路 ($1 \leq n \leq 8$)	$n * 128\text{fps}$
1080p * n路 ($n > 8$)	1024fps
4k * n路 ($1 \leq n \leq 8$)	$n * 32\text{fps}$
4k * n路 ($n > 8$)	256fps

性能指标说明（Atlas 训练系列产品）

以下性能数据，是基于一个Stream上下发10个异步媒体数据处理任务后，执行一次[acl.rt.synchronize_stream](#)接口。

场景举例（输入图片格式：YUV420 8bit）	总帧率
1080p * n路 ($1 \leq n \leq 8$)	$n * 128\text{fps}$
1080p * n路 ($n > 8$)	1024fps
4k * n路 ($1 \leq n \leq 8$)	$n * 32\text{fps}$
4k * n路 ($n > 8$)	256fps

性能指标说明（Atlas 200I/500 A2 推理产品）

以下性能数据，是基于一个Stream上下发10个异步媒体数据处理任务后，执行一次[acl.rt.synchronize_stream](#)接口。

场景举例（输入图片格式：YUV420 8bit）	总帧率
1080p * n路	256fps
4k * n路	64fps

性能指标说明（Atlas A2 训练系列产品/Atlas 800I A2 推理产品）

以下性能数据，是基于一个Stream上下发10个异步媒体数据处理任务后，执行一次[acl.rt.synchronize_stream](#)接口。

场景举例（输入图片格式：YUV420 8bit）	总帧率
1080p * n路（ $1 \leq n \leq 4$ ）	$n * 256\text{fps}$
1080p * n路（ $n > 4$ ）	1024fps
4k * n路（ $1 \leq n \leq 4$ ）	$n * 64\text{fps}$
4k * n路（ $n > 4$ ）	256fps

2.9.8.3 函数：dvpp_jpeg_encode_async

C函数原型	<code>aclError aclDvppJpegEncodeAsync(aclDvppChannelDesc *channelDesc, aclDvppPicDesc *inputDesc, const void *data, uint32_t *size, aclDvppJpegeConfig *config, aclrtStream stream)</code>
Python函数	<code>ret = acl.media.dvpp_jpeg_encode_async(dvpp_channel_desc, input_dvpp_pic_desc, data, size_ptr, dvpp_jpege_config, stream)</code>
函数功能	将YUV格式图片编码成.jpg图片，异步接口。

输入说明	dvpp_channel_desc: int, 指定通道描述信息的指针地址, 与调用acl.media.dvpp_create_channel接口创建通道时指定的dvpp_channel_desc保持一致。 input_dvpp_pic_desc: int, 输入图片描述信息的指针地址。 - 调用acl.media.dvpp_create_pic_desc接口创建图片描述信息。 - 调用acl.media.dvpp_set_pic_desc系列接口设置图片描述参数 (例如, 图片格式、宽、高等)。输入图片分辨率 (≤ 通道的最大宽度 * 高度)、图像格式的要求, 请参见2.9.8.1 功能及约束说明。 data: int, 输出内存地址的指针地址, 存放编码后的数据。 size_ptr: int, 指向输出内存大小的指针地址, 内存大小单位Byte。 - 可提前调用acl.media.dvpp_jpeg_predict_enc_size接口预估输出内存大小 “out_buffer_size”。 - 调用np_out_size = np.array([out_buffer_size], dtype=np.int32)创建numpy array类型数据。再调用bytes_data = np_out_size.tobytes()-> \->size_ptr = acl.util.bytes_to_ptr(bytes_data), 将数据转换为对象作为入参。 注意: 函数调用成功后, size_ptr指向的数据 (即np_out_size) 会发生改变。 dvpp_jpege_config: int, 表示图片编码配置数据的指针地址。 - 调用acl.media.dvpp_create_jpege_config接口创建图片编码配置数据, 调用acl.media.dvpp_set_jpege_config_level接口设置编码配置数据。 stream: int, 指定Stream对象。
返回值说明	ret: int, 错误码。 - 返回0, 表示任务下发成功。 - 返回非0, 表示任务下发失败。
约束说明	本接口是异步接口, 调用接口成功仅表示任务下发成功, 不表示任务执行成功。调用该接口后, 需调用同步等待接口 (例如, acl.rt.synchronize_stream) 确保任务已执行完成。
参考资源	接口调用流程及示例, 参见JPEG图片编码。

2.9.8.4 函数: dvpp_jpeg_predict_enc_size

C函数原型	<code>aclError aclDvppJpegPredictEncSize(const aclDvppPicDesc *inputDesc, const aclDvppJpegeConfig *config, uint32_t *size)</code>
Python函数	<code>size, ret = acl.media.dvpp_jpeg_predict_enc_size(input_dvpp_pic_desc, config)</code>

函数功能	根据输入图片描述信息、图片编码配置数据预估图片编码后所需的输出内存的大小。
输入说明	input_dvpp_pic_desc : int, 输入图片描述信息的指针地址。 - 调用acl.media.dvpp_create_pic_desc接口创建图片描述信息。 - 调用acl.media.dvpp_set_pic_desc系列接口设置图片描述参数（例如，图片格式、宽、高等）。 config : int, 图片编码配置数据的指针地址。 调用 acl.media.dvpp_create_jpege_config 接口创建图片编码配置数据。
返回值说明	size : int, 预估JPEG图片编码后所需的输出内存的大小，单位为Byte。 预估的输出内存会大于实际输出内存，实际输出内存需从 acl.media.dvpp_jpeg_encode_async 接口的输出参数size获取。 ret : int, 错误码。 - 返回0表示成功。 - 返回其它值表示失败。
参考资源	接口调用流程及示例，参见JPEG图片编码。

2.9.8.5 函数：dvpp_create_jpege_config

C函数原型	acldvppJpegeConfig *acldvppCreateJpegeConfig()
Python函数	dvpp_jpege_config = acl.media.dvpp_create_jpege_config()
函数功能	创建acldvppJpegeConfig类型的数据，表示图片编码配置数据。 如需销毁acldvppJpegeConfig类型的数据，请参见 acl.media.dvpp_destroy_jpege_config 。
返回值说明	dvpp_jpege_config : int, acldvppJpegeConfig类型的指针地址。 - 返回非0值，表示成功。 - 返回0，表示失败。

2.9.8.6 函数：dvpp_destroy_jpege_config

C函数原型	aclError acldvppDestroyJpegeConfig(acldvppJpegeConfig *jpegeConfig)
Python函数	ret = acl.media.dvpp_destroy_jpege_config(dvpp_jpege_config)
函数功能	销毁通过 acl.media.dvpp_create_jpege_config 接口创建的编码配置数据。

输入说明	dvpp_jpege_config : int, 指定待销毁数据的指针地址。
返回值说明	ret : int, 错误码。 - 返回0表示成功。 - 返回其它值表示失败。

2.9.8.7 函数: dvpp_set_jpege_config_level

C函数原型	aclError aclDvppSetJpegeConfigLevel(aclDvppJpegeConfig *jpegeConfig, uint32_t level)
Python函数	ret = acl.media.dvpp_set_jpege_config_level(dvpp_jpege_config, level)
函数功能	设置编码配置数据。
输入说明	dvpp_jpege_config : int, 指定编码配置数据的指针地址。调用 acl.media.dvpp_create_jpege_config 接口创建的编码配置数据。 level : int, 编码质量范围。 Atlas 200/300/500 推理产品 , 该参数取值范围: [0, 100], 其中 level 0 编码质量与 level 100 差距较小, 而在 [1, 100] 内数值越小输出图片质量越差。 Atlas 训练系列产品 , 该参数取值范围: [0, 100], 其中 level 0 编码质量与 level 100 差距较小, 而在 [1, 100] 内数值越小输出图片质量越差。 Atlas 推理系列产品 , 该参数取值范围: [1, 100], 数值越小输出图片质量越差。 Atlas A2训练系列产品/Atlas 800I A2推理产品 , 该参数取值范围: [1, 100], 数值越小输出图片质量越差。 Atlas 200I/500 A2推理产品 , 该参数取值范围: [1, 100], 数值越小输出图片质量越差。
返回值说明	ret : int, 错误码。 - 返回0表示成功。 - 返回其它值表示失败。

2.9.8.8 函数: dvpp_get_jpege_config_level

C函数原型	uint32_t aclDvppGetJpegeConfigLevel(const aclDvppJpegeConfig *jpegeConfig)
Python函数	level = acl.media.dvpp_get_jpege_config_level(dvpp_jpege_config)
函数功能	获取编码配置数据。

输入说明	dvpp_jpege_config : int, 指定编码配置数据的指针地址。调用 acl.media.dvpp_create_jpege_config 接口创建aclDvppJpegeConfig类型的数据, 调用 acl.media.dvpp_set_jpege_config_level 接口设置编码配置数据。
返回值说明	level : int, 返回编码质量范围。 Atlas 200/300/500 推理产品 , 该参数取值范围: [0, 100], 其中level 0编码质量与level 100差距较小, 而在[1, 100]内数值越小输出图片质量越差。 Atlas 训练系列产品 , 该参数取值范围: [0, 100], 其中level 0编码质量与level 100差距较小, 而在[1, 100]内数值越小输出图片质量越差。 Atlas 推理系列产品 , 该参数取值范围: [1, 100], 数值越小输出图片质量越差。 Atlas A2训练系列产品/Atlas 800I A2推理产品 , 该参数取值范围: [1, 100], 数值越小输出图片质量越差。 Atlas 200I/500 A2推理产品 , 该参数取值范围: [1, 100], 数值越小输出图片质量越差。

2.9.9 VDEC 功能

2.9.9.1 功能及约束说明

功能说明

VDEC (Video Decoder) 实现视频的解码, VDEC内部经过VPC处理后, 输出指定格式的图片, 参见[码流/图片格式](#)、[宽高对齐](#)、[内存约束](#)。

分辨率约束

- 输入码流分辨率:

版本	分辨率范围
Atlas 200/300/500 推理产品 Atlas 训练系列产品 Atlas 推理系列产品 Atlas A2训练系列产品/Atlas 800I A2推理产品	最大分辨率4096 * 4096, 最小分辨率128 * 128。
Atlas 200I/500 A2推理产品	对于H.264输入码流, 最大分辨率8192 * 8192, 最小分辨率128 * 128。 对于H.265输入码流, 最大分辨率5400 * 8192, 最小分辨率128 * 128。

- 输出图片分辨率：

版本	分辨率范围
Atlas 200/300/500 推理产品 Atlas 训练系列产品 Atlas 推理系列产品 Atlas A2训练系列产品/Atlas 800I A2推理产品	最大分辨率4096 * 4096，最小分辨率10 * 6。
Atlas 200I/500 A2推理产品	输出图片最大分辨率8192 * 8192，当输入码流宽度小于或等于4096时，输出图片最小分辨率为10 * 6，当输入码流宽度大于4096时，输出图片最小分辨率为128 * 128。

码流/图片格式、宽高对齐、内存约束

实现VDEC视频解码功能时，输入或输出内存的生命周期由用户自行管理：

- 输入内存
调用[acl.media.dvpp_malloc](#)/[acl.media.dvpp_free](#)接口申请/释放内存或调用[acl.rt.malloc](#)/[acl.rt.free](#)接口申请/释放内存，内存大小就是指实际的输入码流所占用的大小。
调用[acl.media.dvpp_malloc](#)接口申请的内存为媒体数据处理的专用内存，但专用内存的地址空间有限，若关注内存规划或内存资源有限时，使用VDEC视频解码功能时，建议调用[acl.rt.malloc](#)接口申请内存。
- 输出内存
需调用[acl.media.dvpp_malloc](#)接口申请Device上的输出内存，调用[acl.media.dvpp_free](#)接口释放输出内存，内存大小请参见下表中的计算公式。

说明

输出图片格式的定义请参见[aclDvppPixelFormat](#)，宽Stride、高Stride等概念请参见[2.9.2 基本概念](#)。

VDEC视频解码时，输入码流支持如下格式：

- H.264 bp/mp/hp level5.1 YUV420编码的码流，当前只支持annex-B的裸码流。
- H.265 8/10bit level5.1 YUV420编码的码流，当前只支持annex-B的裸码流。

VDEC视频解码时，输出图片格式、宽高对齐要求等如下表所示：

表 2-24 码流/图片格式、宽高对齐、内存大小约束

版本	输出图 片格式	输出图 片宽、 高对齐 要求	输出图片宽 stride、高 stride、内存大小 要求 Atlas 200/300/500 推理 产品 Atlas 训练系列产 品 Atlas 推理系列产 品	输出图片宽 stride、高 stride、内存大小 要求 Atlas 200I/500 A2推理产品 Atlas A2训练系列 产品/Atlas 800I A2推理产品
通用格式，各版本 都支持	YUV420 SP NV12 8bit	宽2对齐 高2对齐	宽stride为宽16对 齐后的值。如果用 户设置的输出图片 宽度小于16，则宽 stride最小为32。 高stride为高2对齐 后的值。 内存大小（单位 Byte）≥ 宽stride * 高stride * 3/2	宽stride无对齐要 求，与宽相同即可，最小10，最大 24576。 高stride无对齐要 求，与高相同即可，最小6，最大 24576。 内存大小（单位 Byte）≥ 宽stride * 高stride * 3/2
	YUV420 SP NV21 8bit			
该部分格式仅以下 版本支持： Atlas 推理系列产 品 Atlas 200I/500 A2 推理产品 Atlas A2训练系列 产品/Atlas 800I A2推理产品	RGB888	无对齐 要求	宽stride为宽16对 齐后再乘以3的 值。 高stride无对齐要 求。 内存大小（单位 Byte）≥ 宽stride * 高stride	宽stride为宽乘以3 的值，最小30，最 大24576。 高stride无对齐要 求，最小6，最大 24576。 内存大小（单位 Byte）≥ 宽stride * 高stride
	BGR888			

其它约束

- VDEC只支持按帧输入码流进行解码。
- 若码流中有坏帧、缺帧等情况，解码器VDEC可能会丢帧。
- 通过隔行扫描方式编码出来的码流，VDEC仅支持解码H.264 8bit编码的码流。

2.9.9.2 性能指标说明

本节仅描述各产品对应的昇腾AI处理器的性能指标参考数据，各产品由于其内存、CPU等配置不同，在性能规格上会存在差异，详细产品的性能规格请查阅[昇腾社区](#)的官方说明。

本节以单个昇腾AI处理器为例，描述基本场景性能指标参考，其它分辨率可以等量估算。

下表中的1路对应一个通道，一个通道对应一个线程；各分辨率的含义为：720p指分辨率为1280 * 720的图片，1080p指分辨率为1920 * 1080的图片，4K指分辨率为3840 * 2160的图片。

性能指标说明（Atlas 200/300/500 推理产品）

- 针对如下典型场景下各分辨率的码流，单个Device的VDEC解码的路数推荐如下值：

表 2-25 Atlas 200/300/500 推理产品的昇腾 AI 处理器性能指标参考数据

分辨率	总性能	单路性能（n 路）	每路VDEC解码的最大内存消耗（此处以H.265格式的输入码流、参考帧数量9个为例，作为参考值）
4k	120fps	120fps / n （推荐n = 4，每路30fps）	约182MB
1080p	480fps	480fps / n （推荐n = 16，每路30fps）	约56MB
≤ 720p	960fps	960fps / n （推荐n = 32，每路30fps）	约35MB

下表以1080P分辨率的输入码流为例，说明每路VDEC解码的最大内存消耗的计算公式，在计算公式中：

- 输入码流缓存大小：1080P分辨率及以下的输入码流，输入码流缓存大小默认为6M；1080P分辨率以上的输入码流，输入码流缓存大小默认为9M。
- 解码图像帧存大小：1080P分辨率的输入码流，该参数值为3MB。其它分辨率时参数值可等量折算。
- 视频解码图像Tmv缓存大小：H.264格式、1080P分辨率的输入码流，该参数值为0.5MB；H.265格式、1080P分辨率的输入码流，该参数值为1MB。其它分辨率时参数值可等量折算。
- 参考帧数量：以最大参考帧个数为例，系统内部会根据实际码流的参考帧个数自适应。

每路VDEC解码的内存消耗计算公式	场景举例	内存消耗（单位为MB）
4MB + 输入码流缓存大小 * 2 + (解码图像帧存大小 + 视频解码图像Tmv缓存大小) * (参考帧数量 + 1)	• 输入码流格式H.264 • 输入码流分辨率1080P • 输入码流缓存大小为6M • 解码图像帧存大小为3MB • 视频解码图像Tmv缓存大小为0.5MB	51MB（参考帧数量9个） 26.5MB（参考帧数量2个）
	• 输入码流格式H.265 • 输入码流分辨率1080P • 输入码流缓存大小为6M • 解码图像帧存大小为3MB • 视频解码图像Tmv缓存大小为1MB	56MB（参考帧数量9个） 28MB（参考帧数量2个）

• 关于解码路数与帧率的建议

 说明

下表中给出的规格建议供参考，如果单进程内启动的路数超过下表中的建议，则可能出现内存不足或性能不够的问题，进而导致创建解码通道失败或执行解码缓慢。

- 720p指分辨率为1280 * 720的图片。
- 1080p指分辨率为1920 * 1080的图片。
- 4K指分辨率为3840 * 2160的图片。

典型分辨率	单进程内各启动路数时的规格建议（依据输入帧率得出）				
-	输入帧率 ≥ 25fps	20fps < 输入帧率 < 25fps	15fps < 输入帧率 ≤ 20fps	10fps < 输入帧率 ≤ 15fps	输入帧率 ≤ 10fps
≤ 720p	32路	32路	32路	32路	32路
1080p	16路	19路	24路	32路	32路
4K	4路	4路	6路	8路	12路

性能指标说明（Atlas 推理系列产品）

以下性能数据，基于以下Host服务器配置：CPU主频为Intel(R) Xeon(R) CPU E5-2680 v4 @ 2.40GHz，CPU核心数为14。

表 2-26 Atlas 推理系列产品的昇腾 AI 处理器性能指标参考数据

场景举例	总帧率
720p * n路(1 ≤ n ≤ 12)	n * 480fps 说明 针对 Atlas 300V 视频解析卡，当1 < n ≤ 12时，总帧率都为n * 400fps。
720p * n路(n > 12)	5760fps 说明 针对 Atlas 300V 视频解析卡，当n > 12时，总帧率都为4800fps。
1080p * n路(1 ≤ n ≤ 12)	n * 240fps 说明 针对 Atlas 300V 视频解析卡，当1 < n ≤ 12时，总帧率都为n * 200fps。
1080p * n路(n > 12)	2880fps 说明 针对 Atlas 300V 视频解析卡，当n > 12时，总帧率都为2400fps。
4k * n路(1 ≤ n ≤ 12)	n * 60fps
4k * n路(n > 12)	720fps

下表说明了典型场景下每路VDEC解码的内存消耗，在内存消耗的计算公式中：

- 输入码流缓存大小：等于输入码流的宽 * 高 * 2。
- 解码图像帧存大小：1080P分辨率的输入码流，该参数值为3MB。其它分辨率时参数值可等量折算。
- 视频解码图像Tmv缓存大小。
 - H.264格式、1080P分辨率的输入码流，该参数值为0.5MB。
 - H.265格式、1080P分辨率的输入码流，该参数值为1MB。其它分辨率时参数值可等量折算。
- 参考帧数量可由用户调用[acl.media.vdec_set_channel_desc_ref_frame_num](#)接口设置。
- 解码后缓存图像帧数固定为2个。

每路VDEC解码的内存消耗计算公式	场景举例	内存消耗 (单位为MB)
6MB + 输入码流缓存大小 * 2 + (解码图像帧存大小 + 视频解码图像Tmv缓存大小) * (参考帧数量 + 解码后缓存图像帧数 + 1)	• 输入码流格式H.264 • 输入码流分辨率1080P • 输入码流缓存大小为4MB • 解码图像帧存大小为3MB • 视频解码图像Tmv缓存大小为0.5MB • 解码后缓存图像帧数2个	52.5MB (参考帧数量8个) 31.5MB (参考帧数量2个)
	• 输入码流格式H.265 • 输入码流分辨率1080P • 输入码流缓存大小为4MB • 解码图像帧存大小为3MB • 视频解码图像Tmv缓存大小为1MB • 解码后缓存图像帧数2个	58MB (参考帧数量8个) 34MB (参考帧数量2个)

性能指标说明 (Atlas 训练系列产品)

场景举例	总帧率
1080p * n路 * 4进程 (n ≤ 32)	n * 30 * 4fps
4k * n路 * 4进程 (n ≤ 8)	n * 30 * 4fps

下表以1080P分辨率的输入码流为例，说明每路VDEC解码的最大内存消耗的计算公式，在计算公式中：

- 输入码流缓存大小：1080P分辨率及以下的输入码流，输入码流缓存大小默认为6M；1080P分辨率以上的输入码流，输入码流缓存大小默认为9M。
- 解码图像帧存大小：1080P分辨率的输入码流，该参数值为3MB。其它分辨率时参数值可等量折算。
- 视频解码图像Tmv缓存大小：H.264格式、1080P分辨率的输入码流，该参数值为0.5MB；H.265格式、1080P分辨率的输入码流，该参数值为1MB。其它分辨率时参数值可等量折算。

- 参考帧数量：以最大参考帧个数为例，系统内部会根据实际码流的参考帧个数自适应。

每路VDEC解码的内存消耗计算公式	场景举例	内存消耗（单位为MB）
4MB + 输入码流缓存大小 * 2 + (解码图像帧存大小 + 视频解码图像Tmv缓存大小) * (参考帧数量 + 1)	● 输入码流格式H.264 ● 输入码流分辨率1080P ● 输入码流缓存大小为6M ● 解码图像帧存大小为3MB ● 视频解码图像Tmv缓存大小为0.5MB	51MB（参考帧数量9个） 26.5MB（参考帧数量2个）
	● 输入码流格式H.265 ● 输入码流分辨率1080P ● 输入码流缓存大小为6M ● 解码图像帧存大小为3MB ● 视频解码图像Tmv缓存大小为1MB	56MB（参考帧数量9个） 28MB（参考帧数量2个）

性能指标说明（Atlas 200I/500 A2 推理产品）

表 2-27 Atlas 200I/500 A2 推理产品的昇腾 AI 处理器性能指标参考数据

场景举例	总帧率
720p * n路(1 ≤ n ≤ 4)	n * 600fps
720p * n路(n > 4)	2400fps
1080p * n路(1 ≤ n ≤ 4)	n * 300fps
1080p * n路(n > 4)	1200fps
4k * n路(1 ≤ n ≤ 4)	n * 75fps
4k * n路(n > 4)	300fps

下表说明了典型场景下每路VDEC解码的内存消耗，在内存消耗的计算公式中：

- 输入码流缓存大小：等于输入码流的宽 * 高 * 2。
- 解码图像帧存大小：1080P分辨率的输入码流，该参数值为3MB。其它分辨率时参数值可等量折算。

- 视频解码图像Tmv缓存大小：H.264格式、1080P分辨率的输入码流，该参数值为0.5MB；H.265格式、1080P分辨率的输入码流，该参数值为1MB。其它分辨率时参数值可等量折算。
- 参考帧数量可由用户调用[acl.media.vdec_set_channel_desc_ref_frame_num](#)接口设置。
- 解码后缓存图像帧数固定为2个。

每路VDEC解码的内存消耗计算公式	场景举例	内存消耗（单位为MB）
16MB + 输入码流缓存大小 * 2 + (解码图像帧存大小 + 视频解码图像Tmv缓存大小) * (参考帧数量 + 解码后缓存图像帧数 + 1)	• 输入码流格式H.264 • 输入码流分辨率1080P • 输入码流缓存大小为4MB • 解码图像帧存大小为3MB • 视频解码图像Tmv缓存大小为0.5MB • 解码后缓存图像帧数2个	52.5MB（参考帧数量8个） 31.5MB（参考帧数量2个）
	• 输入码流格式H.265 • 输入码流分辨率1080P • 输入码流缓存大小为4MB • 解码图像帧存大小为3MB • 视频解码图像Tmv缓存大小为1MB • 解码后缓存图像帧数2个	58MB（参考帧数量8个） 34MB（参考帧数量2个）

性能指标说明（Atlas A2 训练系列产品/Atlas 800I A2 推理产品）

表 2-28 Atlas A2 训练系列产品/Atlas 800I A2 推理产品的昇腾 AI 处理器性能指标参考数据

场景举例	总帧率
720p * n路(1 ≤ n ≤ 2)	n * 480fps
720p * n路(n>2)	960fps
1080p * n路(1 ≤ n ≤ 2)	n * 240fps
1080p * n路(n > 2)	480fps

场景举例	总帧率
4k * n路($1 \leq n \leq 2$)	$n * 60\text{fps}$
4k * n路($n > 2$)	120fps

下表说明了典型场景下每路VDEC解码的内存消耗，在内存消耗的计算公式中：

- 输入码流缓存大小：等于输入码流的宽 * 高 * 2。
- 解码图像帧存大小：1080P分辨率的输入码流，该参数值为3MB。其它分辨率时参数值可等量折算。
- 视频解码图像Tmv缓存大小：
 - H.264格式、1080P分辨率的输入码流，该参数值为0.5MB
 - H.265格式、1080P分辨率的输入码流，该参数值为1MB。其它分辨率时参数值可等量折算。
- 参考帧数量可由用户调用[acl.media.vdec_set_channel_desc_ref_frame_num](#)接口设置。
- 解码后缓存图像帧数固定为2个。

每路VDEC解码的内存消耗计算公式	场景举例	内存消耗（单位为MB）
$6\text{MB} + \text{输入码流缓存大小} * 2 + (\text{解码图像帧存大小} + \text{视频解码图像Tmv缓存大小}) * (\text{参考帧数量} + \text{解码后缓存图像帧数} + 1)$	• 输入码流格式H.264 • 输入码流分辨率1080P • 输入码流缓存大小为4MB • 解码图像帧存大小为3MB • 视频解码图像Tmv缓存大小为0.5MB • 解码后缓存图像帧数2个	52.5MB（参考帧数量8个） 31.5MB（参考帧数量2个）
	• 输入码流格式H.265 • 输入码流分辨率1080P • 输入码流缓存大小为4MB • 解码图像帧存大小为3MB • 视频解码图像Tmv缓存大小为1MB • 解码后缓存图像帧数2个	58MB（参考帧数量8个） 34MB（参考帧数量2个）

2.9.9.3 函数：vdec_send_frame

C函数原型	<code>aclError aclvdecSendFrame(aclvdecChannelDesc *channelDesc, acldvppStreamDesc *input, acldvppPicDesc *output, aclvdecFrameConfig *config, void* userData)</code>
Python函数	<code>ret = acl.media.vdec_send_frame(vdec_channel_desc, dvpp_stream_desc, dvpp_pic_desc, vdec_frame_config, user_data)</code>
函数功能	将待解码的输入内存和解码输出内存一起传到解码器进行解码，异步接口。

输入说明	vdec_channel_desc: int, 指定通道描述信息的指针地址。 与调用acl.media.dvpp_create_channel接口创建通道时指定的“dvpp_channel_desc”保持一致。 调用acl.media.vdec_set_channel_desc系列接口设置通道描述信息的属性, 包括解码通道号、线程、回调函数、视频编码协议等。 说明 对于Atlas 200/300/500 推理产品, 由于软件约束, 不支持调用acl.media.vdec_set_channel_desc_ref_frame_num接口设置通道描述信息的对应属性, 若调用则会返回报错。 对于Atlas 训练系列产品, 由于软件约束, 不支持调用acl.media.vdec_set_channel_desc_ref_frame_num接口设置通道描述信息的对应属性, 若调用则会返回报错。 dvpp_stream_desc: int, 输入码流描述信息的指针地址, 输入内存用户需提前申请。 - 调用acl.media.dvpp_create_stream_desc接口创建码流描述信息。调用acl.media.dvpp_set_stream_desc系列接口设置视频码流信息的属性(例如, 码流格式)。 输入码流格式以通道描述信息中设置的格式为准, 调用acl.media.dvpp_set_stream_desc_format接口设置输入码流格式无效。 - 输入视频码流的分辨率、视频格式的要求, 请参见2.9.9.1 功能及约束说明。 dvpp_pic_desc: int, 输出图片描述信息的指针地址, 输出内存用户需提前申请。 “dvpp_pic_desc”参数作为输入时, 需要用户调用如下接口: - 调用acl.media.dvpp_create_pic_desc接口创建图片描述信息。 - 调用acl.media.dvpp_set_pic_desc系列接口设置输出图片的数据内存、内存大小、宽、高、格式。 - 如果此处设置输出图片的宽高, 则需要同时设置width、height、width_stride、height_stride这4个参数, 且当width < 16时, “width_stride”必须对齐到32, 不能对齐到16; 如果这4个参数都不设置, 则以输入码流的宽高为准; 如果设置部分参数, 则接口返回错误。 - 如果此处设置输出图片的格式, 则以此处为准; 如果此处不设置, 则以通道描述信息中设置的为准。 - 输出图像格式的要求, 请参见2.9.9.1 功能及约束说明。 “dvpp_pic_desc”参数作为输出时, 用户需在回调函数中调用acl.media.dvpp_get_pic_desc系列接口获取解码后的输出图片数据。隔行码流场景下, 隔行码流每帧发送两场, 解码时其中一场无图像输出, 属于正常现象, 会返回ERR_DECODE_NOPIC = 0x20000错误码; 隔行码流的解码输出数据都在奇数场对应的输出Buffer中。 vdec_frame_config: int, 解码配置的指针地址, 预留, 当前可填0。 user_data: python对象, 用户自定义数据的指针地址。

	如果用户需要获取解码的帧序号，则可以在“user_data”参数处定义，然后解码的帧序号可以通过“user_data”参数传递给VDEC的回调函数，用于确定回调函数中处理的是第几帧数据。
返回值说明	ret : int，错误码。 • 返回0表示成功。 • 返回其它值表示失败。
约束说明	• 本接口是异步接口，调用接口成功仅表示任务下发成功，不表示任务执行成功。调用该接口后，需调用同步等待接口（例如，acl.rt.synchronize_stream）确保任务已执行完成。 • 发送数据前必须保证通道已经被创建，否则返回错误。 • 发送码流时须按帧发送，一次只发送完整的一帧码流。 • 不能发送eos为0的空码流包（码流长度为0或码流地址为空）。 • 发送完所有码流后，可以发送eos为1的空码流包，表示当前码流文件结束。发送eos后，本接口会等待已发送帧全部解码并且用户的回调函数处理完成后才返回。 • 由于码流异常、解码超时等原因，可能导致acl.media.vdec_send_frame接口发送帧或发送eos失败，建议用户在编写代码时，获取该接口的返回码，当该接口调用失败时，进行异常处理。 • acl.media.vdec_send_frame接口内部封装了acl.rt.launch_callback接口，用于在Stream的任务队列中增加一个需要执行的回调函数。用户在实现VDEC功能时，无需再单独调用acl.rt.launch_callback接口。
参考资源	接口调用流程及示例，参见VDEC视频解码。

2.9.9.4 函数：vdec_send_skipped_frame

C函数原型	aclError aclvdecSendSkippedFrame(aclvdecChannelDesc *channelDesc, acldvppStreamDesc *input, aclvdecFrameConfig *config, void *userData)
Python函数	ret = acl.media.vdec_send_skipped_frame (vdec_channel_desc, dvpp_stream_desc, vdec_frame_config, user_data)
函数功能	如果不想获取某一帧的解码结果，可以调用本接口，将待解码的码流（输入内存）传到解码器进行解码，此时，解码结果最终不会输出，解码完成的回调函数中返回的dvpp_pic_desc为0。异步接口。

输入说明	vdec_channel_desc: int, 指定通道描述信息的指针地址。 与调用acl.media.dvpp_create_channel接口创建通道时指定的dvpp_channel_desc保持一致。 调用acl.media.vdec_set_channel_desc系列接口设置通道描述信息的属性, 包括解码通道号、线程、回调函数、视频编码协议等。 说明 对于Atlas 200/300/500 推理产品, 由于软件约束, 不支持调用acl.media.vdec_set_channel_desc_ref_frame_num接口设置通道描述信息的对应属性, 若调用则会返回报错。 对于Atlas 训练系列产品, 由于软件约束, 不支持调用acl.media.vdec_set_channel_desc_ref_frame_num接口设置通道描述信息的对应属性, 若调用则会返回报错。 dvpp_stream_desc: int, 输入码流描述信息的指针地址, 输入内存用户需提前申请。 - 调用acl.media.dvpp_create_stream_desc接口创建码流描述信息, 调用acl.media.dvpp_set_stream_desc系列接口设置视频码流信息的属性(例如, 码流格式)。 输入码流格式以通道描述信息中设置的格式为准, 调用acl.media.dvpp_set_stream_desc_format接口设置输入码流格式无效。 - 输入视频码流的分辨率、视频格式的要求, 请参见2.9.9.1 功能及约束说明。 vdec_frame_config: int, 解码配置的指针地址, 预留, 当前可填0。 user_data: python对象, 用户自定义数据的指针地址。 如果用户需要获取解码的帧序号, 则可以在“user_data”参数处定义, 然后解码的帧序号可以通过“user_data”参数传递给VDEC的回调函数, 用于确定回调函数中处理的是第几帧数据。
返回值说明	ret: int, 错误码。 - 返回0表示成功。 - 返回其它值表示失败。

约束说明	• 本接口是异步接口，调用接口成功仅表示任务下发成功，不表示任务执行成功。调用该接口后，需调用同步等待接口（例如，acl.rt.synchronize_stream）确保任务已执行完成。 • 发送数据前必须保证通道已经被创建，否则返回错误。 • 发送码流时须按帧发送，一次只发送完整的一帧码流。 • 不能发送eos为0的空码流包（码流长度为0或码流地址为空）。 • 不可以发送eos为1的帧。 • 由于码流异常、解码超时等原因，可能导致acl.media.vdec_send_frame接口发送帧或发送eos失败，建议用户在编写代码时，获取该接口的返回码，当该接口调用失败时，进行异常处理。 • acl.media.vdec_send_skipped_frame接口内部封装了acl.rt.launch_callback接口，用于在Stream的任务队列中增加一个需要在Host上执行的回调函数。用户在实现VDEC功能时，无需再单独调用acl.rt.launch_callback接口。

2.9.9.5 函数：vdec_create_frame_config

C函数原型	<code>aclvdecFrameConfig *aclvdecCreateFrameConfig()</code>
Python函数	<code>vdec_frame_config = acl.media.vdec_create_frame_config()</code>
函数功能	创建 aclvdecFrameConfig 类型的数据，表示创建VDEC解码时的单帧解码配置参数。 如需销毁 aclvdecFrameConfig 类型的数据，请参见 acl.media.vdec_destroy_frame_config 。
返回值说明	vdec_frame_config : int，返回 aclvdecFrameConfig 类型数据的指针地址。 • 返回非0值，表示成功。 • 返回0，表示失败。

2.9.9.6 函数：vdec_destroy_frame_config

C函数原型	<code>aclError aclvdecDestroyFrameConfig(aclvdecFrameConfig *vdecFrameConfig)</code>
Python函数	<code>ret = acl.media.vdec_destroy_frame_config(vdec_frame_config)</code>
函数功能	销毁通过 acl.media.vdec_create_frame_config 接口创建的配置数据。
输入说明	vdec_frame_config : int，指定待销毁的 aclvdecFrameConfig 类型数据的指针地址。

返回值说明	ret : int, 错误码。 • 返回0表示成功。 • 返回其它值表示失败。

2.9.9.7 函数: `vdec_create_channel_desc`

C函数原型	<code>aclvdecChannelDesc *aclvdecCreateChannelDesc()</code>
Python函数	<code>vdec_channel_desc = acl.media.vdec_create_channel_desc()</code>
函数功能	创建vdec_channel_desc类型的数据, 表示创建视频数据解码处理通道时的通道描述信息。 如需销毁aclvdecChannelDesc类型的数据, 请参见 acl.media.vdec_destroy_channel_desc 。
返回值说明	vdec_channel_desc : int, 返回aclvdecChannelDesc类型数据的指针地址。 • 返回非0数据, 表示成功。 • 返回0, 表示失败。

2.9.9.8 函数: `vdec_destroy_channel_desc`

C函数原型	<code>aclError aclvdecDestroyChannelDesc(aclvdecChannelDesc *channelDesc)</code>
Python函数	ret = <code>acl.media.vdec_destroy_channel_desc(vdec_channel_desc)</code>
函数功能	销毁通过 acl.media.vdec_create_channel_desc 接口创建的aclvdecChannelDesc类型的数据。
输入说明	vdec_channel_desc : int, 待销毁的aclvdecChannelDesc类型数据的指针地址。
返回值说明	ret : int, 错误码。 • 返回0表示成功。 • 返回其它值表示失败。

2.9.9.9 函数: `vdec_set_channel_desc_channel_id`

C函数原型	<code>aclError aclvdecSetChannelDescChannelId(aclvdecChannelDesc *channelDesc, uint32_t channelId)</code>

Python函数	ret = acl.media.vdec_set_channel_desc_channel_id(vdec_channel_desc, channel_id)
函数功能	设置视频解码处理通道描述信息的属性：解码通道号。
输入说明	vdec_channel_desc : int，视频解码处理通道描述信息的指针地址。 需提前调用 acl.media.vdec_create_channel_desc 接口创建 aclvdecChannelDesc 类型的数据。 channel_id : int，通道ID。 Atlas 200/300/500 推理产品 ，该参数值的取值范围[0, 31]。 Atlas 训练系列产品 ，该参数值的取值范围[0, 31]。 Atlas 200I/500 A2推理产品 ，该参数值的取值范围[0, 255]。 Atlas A2训练系列产品/Atlas 800I A2推理产品 ，该参数值的取值范围[0, 255]。 Atlas 推理系列产品 ，该参数值的取值范围[0, 255]。
返回值说明	ret : int，错误码。 - 返回0表示成功。 - 返回其它值表示失败。

2.9.9.10 函数：vdec_set_channel_desc_thread_id

C函数原型	aclError aclvdecSetChannelDescThreadId(aclvdecChannelDesc *channelDesc, uint64_t threadId)
Python函数	ret = acl.media.vdec_set_channel_desc_thread_id(vdec_channel_desc, thread_id)
函数功能	设置视频解码处理通道描述信息的属性：回调线程ID。
输入说明	vdec_channel_desc : int，视频解码处理通道描述信息的指针地址。 需提前调用 acl.media.vdec_create_channel_desc 接口创建 aclvdecChannelDesc 类型的数据。 thread_id : int，回调线程ID。 说明 同一个进程内，在不同的Device上注册VDEC解码回调函数的线程时，不能指定同一个线程ID。 同一个Device上，多路VDEC解码可以指定同一个线程ID，但相比一个线程处理一路VDEC解码任务来说，一个线程内串行处理多路VDEC解码任务时，VDEC解码性能可能下降。

返回值说明	ret : int, 错误码。 - 返回0表示成功。 - 返回其它值表示失败。

2.9.9.11 函数: vdec_set_channel_desc_callback

C函数原型	aclError aclvdecSetChannelDescCallback(aclvdecChannelDesc *channelDesc, aclvdecCallback callback)
Python函数	ret = acl.media.vdec_set_channel_desc_callback(vdec_channel_desc, callback)
函数功能	设置视频解码处理通道描述信息的属性：回调函数。
输入说明	vdec_channel_desc : int, 视频解码处理通道描述信息的指针地址。 需提前调用 acl.media.vdec_create_channel_desc 接口创建 aclvdecChannelDesc 类型的数据。 callback : Python函数对象, 回调函数。 Python侧格式如下: <pre>def vdec_call_back_func(dvpp_stream_desc, dvpp_pic_desc, user_data): pass</pre>
返回值说明	ret : int, 错误码。 - 返回0表示成功。 - 返回其它值表示失败。

2.9.9.12 函数: vdec_set_channel_desc_entype

C函数原型	aclError aclvdecSetChannelDescEnType(aclvdecChannelDesc *channelDesc, acldvppStreamFormat enType)
Python函数	ret = acl.media.vdec_set_channel_desc_entype(vdec_channel_desc, entype)
函数功能	设置视频解码处理通道描述信息的属性：视频编码协议。
输入说明	vdec_channel_desc : int, 视频解码处理通道描述信息的指针地址。 需提前调用 acl.media.vdec_create_channel_desc 接口创建 aclvdecChannelDesc 类型的数据。 entype : int, 视频编码协议, 具体请参见 2.16.7 acldvppStreamFormat 。

返回值说明	ret : int, 错误码。 - 返回0表示成功。 - 返回其它值表示失败。

2.9.9.13 函数: vdec_set_channel_desc_out_pic_format

C函数原型	aclError aclvdecSetChannelDescOutPicFormat (aclvdecChannelDesc *channelDesc, aclvppPixelFormat outPicFormat)
Python函数	ret = acl.media.vdec_set_channel_desc_out_pic_format (vdec_channel_desc, out_pic_format)
函数功能	设置视频解码处理通道描述信息的属性: YUV图像存储格式。
输入说明	vdec_channel_desc : int, 视频解码处理通道描述信息的指针地址。 需提前调用 acl.media.vdec_create_channel_desc 接口创建aclvdecChannelDesc类型的数据。 out_pic_format : int, YUV图像存储格式, 支持格式请参见 表 2-24 。 如果不设置输出格式, 默认使用YUV420SP NV12。
返回值说明	ret : int, 错误码。 - 返回0表示成功。 - 返回其它值表示失败。

2.9.9.14 函数: vdec_set_channel_desc_out_pic_width

C函数原型	aclError aclvdecSetChannelDescOutPicWidth (aclvdecChannelDesc *channelDesc, uint32_t outPicWidth)
Python函数	ret = acl.media.vdec_set_channel_desc_out_pic_width (vdec_channel_desc, out_pic_width)
函数功能	设置视频解码处理通道描述信息的属性: 图片宽度。

输入说明	vdec_channel_desc: int, 视频解码处理通道描述信息的指针地址。 out_pic_width: int, 解码码流最大宽度。 Atlas 200/300/500 推理产品, 如果不设置解码码流最大宽度, 内部默认使用1920。 Atlas 训练系列产品, 如果不设置解码码流最大宽度, 内部默认使用1920。 Atlas 200I/500 A2推理产品, 如果不设置解码码流最大宽度, 内部默认使用4096。 Atlas A2训练系列产品/Atlas 800I A2推理产品, 如果不设置解码码流最大宽度, 内部默认使用4096。 Atlas 推理系列产品, 如果不设置解码码流最大宽度, 内部默认使用4096。
返回值说明	ret: int, 错误码。 • 返回0表示成功。 • 返回其它值表示失败。

2.9.9.15 函数: vdec_set_channel_desc_out_pic_height

C函数原型	aclError aclvdecSetChannelDescOutPicHeight(aclvdecChannelDesc *channelDesc, uint32_t outPicHeight)
Python函数	ret = acl.media.vdec_set_channel_desc_out_pic_height(vdec_channel_desc, out_pic_height)
函数功能	设置视频解码处理通道描述信息的属性: 图片高度。
输入说明	vdec_channel_desc: int, 视频数据解码处理通道的描述信息。需提前调用acl.media.vdec_create_channel_desc接口创建aclvdecChannelDesc类型的数据。 out_pic_height: int, 解码码流最大高度。 Atlas 200/300/500 推理产品, 如果不设置解码码流最大高度, 内部默认使用1120。 Atlas 训练系列产品, 如果不设置解码码流最大高度, 内部默认使用1120。 Atlas 200I/500 A2推理产品, 如果不设置解码码流最大高度, 内部默认使用4096。 Atlas A2训练系列产品/Atlas 800I A2推理产品, 如果不设置解码码流最大高度, 内部默认使用4096。 Atlas 推理系列产品, 如果不设置解码码流最大高度, 内部默认使用4096。

返回值说明	ret : int, 错误码。 - 返回0表示成功。 - 返回其它值表示失败。

2.9.9.16 函数: vdec_set_channel_desc_ref_frame_num

C函数原型	aclError aclvdecSetChannelDescRefFrameNum (aclvdecChannelDesc *channelDesc, uint32_t refFrameNum)
Python函数	ret = acl.media.vdec_set_channel_desc_ref_frame_num (vdec_channel_desc, ref_frame_num)
函数功能	设置视频解码处理通道描述信息的属性: 参考帧数量。
输入说明	vdec_channel_desc : int, 视频数据解码处理通道的描述信息。 需提前调用 acl.media.vdec_create_channel_desc 接口创建aclvdecChannelDesc类型的数据。 ref_frame_num : int, 参考帧数量, 取值范围[0, 16]。 Atlas 200/300/500 推理产品 , 当前版本不支持用户设置该参数。 Atlas 训练系列产品 , 当前版本不支持用户设置该参数。 Atlas 200I/500 A2推理产品 , 如果不设置参考帧数量, 内部默认使用8。 Atlas A2训练系列产品/Atlas 800I A2推理产品 , 如果不设置参考帧数量, 内部默认使用8。 Atlas 推理系列产品 , 如果不设置参考帧数量, 内部默认使用8。
返回值说明	ret : int, 错误码。 - 返回0表示成功。 - 返回其它值表示失败。

2.9.9.17 函数: vdec_set_channel_desc_out_mode

C函数原型	aclError aclvdecSetChannelDescOutMode (aclvdecChannelDesc *channelDesc, uint32_t outMode)
Python函数	ret = acl.media.vdec_set_channel_desc_out_mode (vdec_channel_desc, out_mode)
函数功能	设置视频解码处理通道描述信息的属性: 是否实时出帧。

输入说明	vdec_channel_desc: int, 视频数据解码处理通道的描述信息。需提前调用acl.media.vdec_create_channel_desc接口创建aclvdecChannelDesc类型的数据。 out_mode: int, 设置是否实时出帧（即发送一帧解码一帧，无需依赖后续帧的传入）。取值范围如下： 0: 默认出帧模式，由于解码过程中存在缓存帧，无法实时输出，因此VDEC需要在收到码流中的多帧数据后，才开始输出解码结果。 1: 快速出帧模式，VDEC获取码流中的一帧数据后，就开始实时输出解码结果。只支持简单参考关系的H.264/H.265标准码流（无长期参考帧，无B帧）。
返回值说明	ret: int, 错误码。 - 返回0表示成功。 - 返回其它值表示失败。

2.9.9.18 函数：vdec_set_channel_desc_bit_depth

C函数原型	aclError aclvdecSetChannelDescBitDepth(aclvdecChannelDesc *channelDesc, uint32_t bitDepth)
Python函数	ret = acl.media.vdec_set_channel_desc_bit_depth(vdec_channel_desc, desc_bit_depth)
函数功能	设置视频解码处理通道描述信息的属性。
输入说明	vdec_channel_desc: int, 视频解码处理通道的描述信息。需提前调用acl.media.vdec_create_channel_desc接口创建aclvdecChannelDesc类型的数据。 desc_bit_depth: int, 设置视频位宽，默认值为10-bit。取值范围如下： 0: 8-bit 1: 10-bit（默认值） Atlas 200/300/500 推理产品，当前版本不支持用户设置该参数。 Atlas 训练系列产品，当前版本不支持用户设置该参数。
返回值说明	ret: int, 错误码。 - 返回0表示成功。 - 返回其它值表示失败。

2.9.9.19 函数：vdec_get_channel_desc_channel_id

C函数原型	<code>uint32_t aclvdecGetChannelDescChannelId(const aclvdecChannelDesc *channelDesc)</code>
Python函数	<code>channel_id = acl.media.vdec_get_channel_desc_channel_id(vdec_channel_desc)</code>
函数功能	获取视频解码处理通道描述信息的属性：解码通道号。
输入说明	<code>vdec_channel_desc</code> : int, 视频数据解码处理通道的描述信息。
返回值说明	<code>channel_id</code> : int, 解码通道号。 Atlas 200/300/500 推理产品, 该参数值的取值范围[0, 31]。 Atlas 训练系列产品, 该参数值的取值范围[0, 31]。 Atlas A2训练系列产品/Atlas 800I A2推理产品, 该参数值的取值范围[0, 255]。 Atlas 200I/500 A2推理产品, 该参数值的取值范围[0, 255]。 Atlas 推理系列产品, 该参数值的取值范围[0, 255]。

2.9.9.20 函数：vdec_get_channel_desc_thread_id

C函数原型	<code>uint32_t aclvdecGetChannelDescThreadId(const aclvdecChannelDesc *channelDesc);</code>
Python函数	<code>thread_id = acl.media.vdec_get_channel_desc_thread_id(vdec_channel_desc)</code>
函数功能	获取视频解码处理通道描述信息的属性：回调线程ID。
输入说明	<code>vdec_channel_desc</code> : int, 视频数据解码处理通道的描述信息。
返回值说明	<code>thread_id</code> : int, 回调线程ID。

2.9.9.21 函数：vdec_get_channel_desc_callback

C函数原型	<code>aclvdecCallback aclvdecGetChannelDescCallback(const aclvdecChannelDesc *channelDesc)</code>
Python函数	<code>callback = acl.media.vdec_get_channel_desc_callback(vdec_channel_desc)</code>
函数功能	获取视频解码处理通道描述信息的属性：回调函数。
输入说明	<code>vdec_channel_desc</code> : int, 视频数据解码处理通道的描述信息。

返回值说明	callback : python函数对象, 回调函数。 Python侧格式如下: <pre>def vdec_call_back_func(dvpp_stream_desc, dvpp_pic_desc, user_data): pass</pre>

2.9.9.22 函数: vdec_get_channel_desc_entype

C函数原型	aclDvppStreamFormat aclVdecGetChannelDescEnType (const aclVdecChannelDesc *channelDesc);
Python函数	entype = acl.media.vdec_get_channel_desc_entype (vdec_channel_desc)
函数功能	获取视频解码处理通道描述信息的属性: 视频编码协议。
输入说明	vdec_channel_desc : int, 视频数据解码处理通道的描述信息。
返回值说明	entype : int, 视频编码协议, 具体请参见 2.16.7 aclDvppStreamFormat 。

2.9.9.23 函数: vdec_get_channel_desc_out_pic_format

C函数原型	aclDvppPixelFormat aclVdecGetChannelDescOutPicFormat (const aclVdecChannelDesc *channelDesc);
Python函数	out_pic_format = acl.media.vdec_get_channel_desc_out_pic_format (vdec_channel_desc)
函数功能	获取视频解码处理通道描述信息的属性: YUV图像存储格式。
输入说明	vdec_channel_desc : int, 视频数据解码处理通道的描述信息。
返回值说明	out_pic_format : int, YUV图像存储格式。

2.9.9.24 函数: vdec_get_channel_desc_out_pic_width

C函数原型	uint32_t aclVdecGetChannelDescOutPicWidth (const aclVdecChannelDesc *channelDesc);
Python函数	out_pic_width = acl.media.vdec_get_channel_desc_out_pic_width (vdec_channel_desc)
函数功能	获取视频解码处理通道描述信息的属性: 图片宽度。
输入说明	vdec_channel_desc : int, 视频数据解码处理通道的描述信息。

返回值说明	out_pic_width: int, 图片宽度。

2.9.9.25 函数: vdec_get_channel_desc_out_pic_height

C函数原型	uint32_t aclvdecGetChannelDescOutPicHeight(const aclvdecChannelDesc *channelDesc);
Python函数	out_pic_height = acl.media.vdec_get_channel_desc_out_pic_height(vdec_channel_desc)
函数功能	获取视频解码处理通道描述信息的属性: 图片高度。
输入说明	vdec_channel_desc: int, 视频数据解码处理通道的描述信息。
返回值说明	out_pic_height: int, 图片高度。

2.9.9.26 函数: vdec_get_channel_desc_ref_frame_num

C函数原型	uint32_t aclvdecGetChannelDescRefFrameNum(const aclvdecChannelDesc *channelDesc)
Python函数	ref_frame_num = acl.media.vdec_get_channel_desc_ref_frame_num(vdec_channel_desc)
函数功能	获取视频解码处理通道描述信息的属性: 参考帧数量。
输入说明	vdec_channel_desc: int, 视频数据解码处理通道的描述信息。
返回值说明	ref_frame_num: int, 参考帧数量, 取值范围[0, 16]。 Atlas 200/300/500 推理产品 , 当前版本获取该参数的值为无效值, 不支持获取该参数值。 Atlas 训练系列产品 , 当前版本获取该参数的值为无效值, 不支持获取该参数值。

2.9.9.27 函数: vdec_get_channel_desc_out_mode

C函数原型	uint32_t aclvdecGetChannelDescOutMode(const aclvdecChannelDesc *channelDesc)
Python函数	out_mode = acl.media.vdec_get_channel_desc_out_mode(vdec_channel_desc)
函数功能	获取视频解码处理通道的描述信息: 是否实时出帧。
输入说明	vdec_channel_desc: int, 视频数据解码处理通道的描述信息。

返回值说明	out_mode: int, 设置是否实时出帧（即发送一帧解码一帧，无需依赖后续帧的传入）。 取值范围如下： 0: 默认出帧模式，由于解码过程中存在缓存帧，无法实时输出，因此VDEC需要在收到码流中的多帧数据后，才开始输出解码结果。 1: 快速出帧模式，VDEC获取码流中的一帧数据后，就开始实时输出解码结果。只支持简单参考关系的H.264/H.265标准码流（无长期参考帧，无B帧）。

2.9.9.28 函数：vdec_get_channel_desc_bit_depth

C函数原型	uint32_t aclvdecGetChannelDescBitDepth(const aclvdecChannelDesc *channelDesc)
Python函数	desc_bit_depth = acl.media.vdec_get_channel_desc_bit_depth(vdec_channel_desc)
函数功能	获取视频解码处理通道的描述信息。
输入说明	vdec_channel_desc: int, 表视频解码数据处理通道的描述信息。
返回值说明	desc_bit_depth: int, 视频位宽。 取值范围如下： 0: 8-bit 1: 10-bit Atlas 200/300/500 推理产品，当前版本获取该参数的值为无效值，不支持获取该参数值。 Atlas 训练系列产品，当前版本获取该参数的值为无效值，不支持获取该参数值。

2.9.10 VENC 功能

2.9.10.1 功能及约束说明

Atlas 训练系列产品上，不支持该功能。

Atlas A2训练系列产品/Atlas 800I A2推理产品上，不支持该功能。

功能说明

VENC (Video Encoder) 将YUV420SP NV12/NV21-8bit图片数据编码成H.264/H.265格式的视频码流。

在Atlas 200/300/500 推理产品上，不支持单进程多线程场景。

分辨率约束

- 输入图片分辨率

版本	分辨率范围
Atlas 200/300/500 推理产品	最大分辨率1920 * 1920，最小分辨率128 * 128。
Atlas 推理系列产品	宽和高的范围都是128 ~ 4096，对于H.264格式的码流，最大分辨率的乘积不超过4096 * 2304。
Atlas 200I/500 A2推理产品	最大分辨率8192 * 8192，最小分辨率114 * 114。

- 输出码流分辨率
VENC只对图片编码，不会改变图片分辨率，因此输出与输入的图片分辨率保持一致。

码流/图片格式、宽高对齐、内存约束

实现VENC视频编码功能时：

- 输入内存
需调用[acl.media.dvpp_malloc](#)接口申请Device上的输入内存，调用[acl.media.dvpp_free](#)接口释放输入内存，这部分内存的生命周期由用户自行管理。内存大小参见下表中的计算公式。
- 输出内存
不需要用户管理输出内存，由系统管理内存。

说明

输入图片格式的定义请参见[aclDvppPixelFormat](#)，宽Stride、高Stride等概念请参见[2.9.2 基本概念](#)。

表 2-29 图片格式、宽高对齐、内存大小约束

输入图片格式	输入图片宽、高对齐要求	输入图片宽Stride、高Stride、内存大小要求	输出码流格式
YUV420SP NV12 8bit	宽2对齐 高2对齐	宽Stride为宽16对齐后的值。 高Stride无需设置。 内存大小（单位Byte） = 宽Stride * 高Stride * 3/2	- H.264 BP/MP/HP - H.265 MP（仅支持Slice码流）
YUV420SP NV21 8bit			

2.9.10.2 性能指标说明

- Atlas 训练系列产品上，不支持该功能。
- Atlas A2训练系列产品/Atlas 800I A2推理产品上，不支持该功能。

本节仅描述各产品对应的昇腾AI处理器的性能指标参考数据，各产品由于其内存、CPU等配置不同，在性能规格上会存在差异，详细产品的性能规格请查阅[昇腾社区](#)的官方说明。

本节以单个昇腾AI处理器为例，描述基本场景性能指标参考，其它分辨率可以等量估算。

下表中的1路对应一个通道，一个通道对应一个线程；各分辨率的含义为：720p指分辨率为1280 * 720的图片，1080p指分辨率为1920 * 1080的图片，4K指分辨率为3840 * 2160的图片。

性能指标说明（Atlas 200/300/500 推理产品）

表 2-30 Atlas 200/300/500 推理产品的昇腾 AI 处理器性能指标参考数据

场景举例	总帧率
1080p * n路（一个进程对应一路）	30fps

性能指标说明（Atlas 推理系列产品）

以下性能数据，基于以下Host服务器配置：CPU主频为Intel(R) Xeon(R) CPU E5-2680 v4 @ 2.40GHz，CPU核心数为14。

表 2-31 Atlas 推理系列产品的昇腾 AI 处理器性能指标参考数据

场景举例	总帧率
720p * n路($1 \leq n \leq 3$)	$n * 480\text{fps}$
720p * n路($n > 3$)	14400fps
1080p * n路($1 \leq n \leq 3$)	$n * 240\text{fps}$
1080p * n路($n > 3$)	720fps
4k * n路($1 \leq n \leq 3$)	$n * 60\text{fps}$
4k * n路($n > 3$)	180fps

性能指标说明（Atlas 200I/500 A2 推理产品）

表 2-32 Atlas 200I/500 A2 推理产品的昇腾 AI 处理器性能指标参考数据

场景举例	总帧率
720p * n路($1 \leq n \leq 3$)	$n * 400\text{fps}$
720p * n路($n > 3$)	1200fps

场景举例	总帧率
1080p * n路($1 \leq n \leq 3$)	$n * 200\text{fps}$
1080p * n路($n > 3$)	600fps
4k * n路($1 \leq n \leq 3$)	$n * 50\text{fps}$
4k * n路($n > 3$)	150fps

2.9.10.3 函数：venc_send_frame

C函数原型	<code>aclError aclvencSendFrame(aclvencChannelDesc *channelDesc, acldvppPicDesc *input, void *reserve, aclvencFrameConfig *config, void *userdata)</code>
Python函数	<code>ret = acl.media.venc_send_frame(venc_channel_desc, input, reserve, config, userdata)</code>
函数功能	将待编码的图片传到编码器进行编码。异步接口。 Atlas 训练系列产品，不支持该接口。 Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。
输入说明	venc_channel_desc : int，指定通道描述的指针地址。 与调用 acl.media.venc_create_channel 接口创建通道时指定的channelDesc保持一致。 在通道描述信息中指定视频编码的回调函数。 input : int，输入图片描述信息的指针地址，输入内存用户需提前申请。 - 调用acl.media.dvpp_create_pic_desc接口创建图片描述信息，调用acl.media.dvpp_set_pic_desc系列接口设置图片描述参数（例如，图片格式、宽、高等）。输入图片格式以通道描述信息中设置的格式为准，调用acl.media.dvpp_set_pic_desc_format接口设置输入图片格式无效。 - 输入图片的分辨率、格式的要求，请参见2.9.10.1 功能及约束说明。 reserve : int，预留参数，暂不使用，设为0。 config : int，单帧配置数据的指针地址。 userdata : Python对象，用户自定义数据。
返回值说明	ret : int，错误码。 - 返回0表示成功。 - 返回其它值表示失败。

约束说明	- 本接口是异步接口，调用接口成功仅表示任务下发成功，不表示任务执行成功。调用该接口后，需调用同步等待接口（例如，acl.rt.synchronize_stream）确保任务已执行完成。 - 发送数据前必须保证通道已经被创建，否则返回错误。 - 不能发送eos为0的空码流包（码流长度为0或码流地址为空）。 - 结束视频编码时可以发送eos为1的空图片，表示当前编码结束。
参考资源	接口调用流程及示例，参见VENC视频编码。

2.9.10.4 函数：venc_create_channel_desc

C函数原型	<code>aclvencChannelDesc *aclvencCreateChannelDesc()</code>
Python函数	<code>venc_channel_desc = acl.media.venc_create_channel_desc()</code>
函数功能	创建aclvencChannelDesc类型的数据，表示创建视频数据编码处理通道时的通道描述信息。 如需销毁aclvencChannelDesc类型的数据，请参见acl.media.venc_destroy_channel_desc。 Atlas 训练系列产品，不支持该接口。 Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。
返回值说明	<code>venc_channel_desc</code>: int，返回aclvencChannelDesc类型数据的指针地址。 - 返回非0值，表示成功。 - 返回0，表示失败。

2.9.10.5 函数：venc_destroy_channel_desc

C函数原型	<code>aclError aclvencDestroyChannelDesc(aclvencChannelDesc *channelDesc)</code>
Python函数	<code>ret = acl.media.venc_destroy_channel_desc(venc_channel_desc)</code>
函数功能	销毁通过acl.media.venc_create_channel_desc接口创建的aclvencChannelDesc类型的数据。 Atlas 训练系列产品，不支持该接口。 Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。
输入说明	<code>venc_channel_desc</code>: int，指定需要销毁的aclvencChannelDesc类型数据的指针地址。

返回值说明	ret : int, 错误码。 - 返回0表示成功。 - 返回其它值表示失败。

2.9.10.6 函数: venc_set_channel_desc_thread_id

C函数原型	aclError aclvencSetChannelDescThreadId(aclvencChannelDesc *channelDesc, uint64_t threadId)
Python函数	ret = acl.media.venc_set_channel_desc_thread_id(venc_channel_desc, thread_id)
函数功能	设置视频编码处理通道描述信息的属性: 回调线程ID。 Atlas 训练系列产品 , 不支持该接口。 Atlas A2训练系列产品/Atlas 800I A2推理产品 , 不支持该接口。
输入说明	venc_channel_desc : int, 视频数据编码处理通道的描述信息的指针地址。 需提前调用 acl.media.venc_create_channel_desc 接口创建aclvencChannelDesc类型的数据。 thread_id : int, 回调线程ID。 说明 同一个进程内, 在不同的Device上注册VENC编码回调函数的线程时, 不能指定同一个线程ID。
返回值说明	ret : int, 错误码。 - 返回0表示成功。 - 返回其它值表示失败。

2.9.10.7 函数: venc_set_channel_desc_callback

C函数原型	aclError aclvencSetChannelDescCallback(aclvencChannelDesc *channelDesc, aclvencCallback callback)
Python函数	ret = acl.media.venc_set_channel_desc_callback(venc_channel_desc, callback)
函数功能	设置视频编码处理通道描述信息的属性: 回调函数。 Atlas 训练系列产品 , 不支持该接口。 Atlas A2训练系列产品/Atlas 800I A2推理产品 , 不支持该接口。

输入说明	venc_channel_desc: int，视频数据编码处理通道的描述信息的指针地址。 需提前调用acl.media.venc_create_channel_desc接口创建aclvencChannelDesc类型的数据。 callback: python函数对象，编码回调函数。 Python侧格式如下： <pre>def venc_call_back_func(dvpp_pic_desc, dvpp_stream_desc, user_data): pass</pre>
返回值说明	ret: int，错误码。 • 返回0表示成功。 • 返回其它值表示失败。

2.9.10.8 函数：venc_set_channel_desc_entype

C函数原型	<code>aclError aclvencSetChannelDescEnType(aclvencChannelDesc *channelDesc, acldvppStreamFormat enType);</code>
Python函数	<code>ret = acl.media.venc_set_channel_desc_entype(venc_channel_desc, venc_entype)</code>
函数功能	设置视频编码处理通道描述信息的属性：视频编码协议。 Atlas 训练系列产品，不支持该接口。 Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。
输入说明	venc_channel_desc: int，视频数据编码处理通道的描述信息的指针地址。 需提前调用acl.media.venc_create_channel_desc接口创建aclvencChannelDesc类型的数据。 venc_entype: int，视频编码协议，具体请参见2.16.7 acldvppStreamFormat。
返回值说明	ret: int，错误码。 • 返回0表示成功。 • 返回其它值表示失败。

2.9.10.9 函数：venc_set_channel_desc_pic_format

C函数原型	<code>aclError aclvencSetChannelDescPicFormat(aclvencChannelDesc *channelDesc, acldvppPixelFormat picFormat);</code>
Python函数	<code>ret = acl.media.venc_set_channel_desc_pic_format(venc_channel_desc, venc_format)</code>

函数功能	设置视频编码处理通道描述信息的属性：输入图像格式。 Atlas 训练系列产品 ，不支持该接口。 Atlas A2训练系列产品/Atlas 800I A2推理产品 ，不支持该接口。
输入说明	venc_channel_desc : int，视频数据编码处理通道的描述信息的指针地址。 需提前调用 acl.media.venc_create_channel_desc 接口创建aclvencChannelDesc类型的数据。 venc_format : int，输入图像格式，支持如下格式： - PIXEL_FORMAT_YUV_SEMIPLANAR_420 - PIXEL_FORMAT_YVU_SEMIPLANAR_420
返回值说明	ret : int，错误码。 - 返回0表示成功。 - 返回其它值表示失败。

2.9.10.10 函数：venc_set_channel_desc_pic_width

C函数原型	aclError aclvencSetChannelDescPicWidth(aclvencChannelDesc *channelDesc, uint32_t picWidth);
Python函数	ret = acl.media.venc_set_channel_desc_pic_width(venc_channel_desc, in_pic_width)
函数功能	设置视频编码处理通道描述信息的属性：图像宽度。 Atlas 训练系列产品 ，不支持该接口。 Atlas A2训练系列产品/Atlas 800I A2推理产品 ，不支持该接口。
输入说明	venc_channel_desc : int，视频数据编码处理通道的描述信息的指针地址。 需提前调用 acl.media.venc_create_channel_desc 接口创建aclvencChannelDesc类型的数据。 in_pic_width : int，图片宽度。
返回值说明	ret : int，错误码。 - 返回0表示成功。 - 返回其它值表示失败。

2.9.10.11 函数：venc_set_channel_desc_pic_height

C函数原型	aclError aclvencSetChannelDescPicHeight(aclvencChannelDesc *channelDesc, uint32_t picHeight);

Python函数	ret = acl.media.venc_set_channel_desc_pic_height(venc_channel_desc, in_pic_height)
函数功能	设置视频编码处理通道描述信息的属性：图像高度。 Atlas 训练系列产品 ，不支持该接口。 Atlas A2训练系列产品/Atlas 800I A2推理产品 ，不支持该接口。
输入说明	venc_channel_desc : int，视频数据编码处理通道的描述信息的指针地址。 需提前调用 acl.media.venc_create_channel_desc 接口创建aclvencChannelDesc类型的数据。 in_pic_height : int，图片高度。
返回值说明	ret : int，错误码。 - 返回0表示成功。 - 返回其它值表示失败。

2.9.10.12 函数：venc_set_channel_desc_key_frame_interval

C函数原型	aclError aclvencSetChannelDescKeyFrameInterval(aclvencChannelDesc *channelDesc, uint32_t keyFrameInterval);
Python函数	ret = acl.media.venc_set_channel_desc_key_frame_interval(venc_channel_desc, key_frame_interval)
函数功能	设置视频编码处理通道描述信息的属性：关键帧间隔。 Atlas 训练系列产品 ，不支持该接口。 Atlas A2训练系列产品/Atlas 800I A2推理产品 ，不支持该接口。
输入说明	venc_channel_desc : int，视频数据编码处理通道的描述信息的指针地址。 需提前调用 acl.media.venc_create_channel_desc 接口创建aclvencChannelDesc类型的数据。 key_frame_interval : int，关键帧间隔，取值范围[1, 65536]。
返回值说明	ret : int，错误码。 - 返回0表示成功。 - 返回其它值表示失败。

2.9.10.13 函数: venc_set_channel_desc_buf_addr

C函数原型	aclError aclvencSetChannelDescBufAddr(aclvencChannelDesc *channelDesc, void *bufAddr)
Python函数	ret = acl.media.venc_set_channel_desc_buf_addr(venc_channel_desc, buf_addr)
函数功能	设置视频编码处理通道描述信息的属性: 编码输出缓存地址。 Atlas 训练系列产品 , 不支持该接口。 Atlas A2训练系列产品/Atlas 800I A2推理产品 , 不支持该接口。
输入说明	venc_channel_desc : int, 视频数据编码处理通道的描述信息的指针地址。 需提前调用 acl.media.venc_create_channel_desc 接口创建aclvencChannelDesc类型的数据。 buf_addr : int, 编码输出缓存地址。 说明 在 Atlas 200/300/500 推理产品 上, 当前版本不支持用户设置该参数, 由VENC内部管理该缓存地址。
返回值说明	ret : int, 错误码。 - 返回0表示成功。 - 返回其它值表示失败。

2.9.10.14 函数: venc_set_channel_desc_buf_size

C函数原型	aclError aclvencSetChannelDescBufSize(aclvencChannelDesc *channelDesc, uint32_t bufSize)
Python函数	ret = acl.media.venc_set_channel_desc_buf_size(venc_channel_desc, buf_size)
函数功能	设置视频编码处理通道描述信息的属性: 编码输出缓存大小。 Atlas 训练系列产品 , 不支持该接口。 Atlas A2训练系列产品/Atlas 800I A2推理产品 , 不支持该接口。

输入说明	venc_channel_desc: int, 视频数据编码处理通道的描述信息的指针地址。 需提前调用acl.media.venc_create_channel_desc接口创建aclvencChannelDesc类型的数据。 buf_size: int, 编码输出缓存大小, 单位为Byte。 说明 在Atlas 200/300/500 推理产品上, 当前版本不支持用户设置该参数, 内部默认使用3686400Byte。 在Atlas 推理系列产品上, 如果不设置该参数, 参数值默认为8M; 如果设置该参数, 参数值最小为5M。 在Atlas 200I/500 A2推理产品上, 如果不设置该参数, 参数值默认为8M; 如果设置该参数, 参数值最小为5M。
返回值说明	ret: int, 错误码。 • 返回0表示成功。 • 返回其它值表示失败。

2.9.10.15 函数: venc_set_channel_desc_rc_mode

C函数原型	aclError aclvencSetChannelDescRcMode(aclvencChannelDesc *channelDesc, uint32_t rcMode)
Python函数	ret = acl.media.venc_set_channel_desc_rc_mode(venc_channel_desc, rc_mode)
函数功能	设置视频编码处理通道描述信息的属性。 Atlas 训练系列产品, 不支持该接口。 Atlas A2训练系列产品/Atlas 800I A2推理产品, 不支持该接口。
输入说明	venc_channel_desc: int, 视频编码处理通道的描述信息的指针地址。 需提前调用acl.media.venc_create_channel_desc接口创建aclvencChannelDesc类型的数据。 rc_mode: int, 指定码率控制模式。 • 0表示使用默认值, 即CBR模式。 • 1表示变码率VBR模式。 • 2表示定码率CBR模式。 说明 如果不设置该参数, 则采用默认值0。 Atlas 200/300/500 推理产品, 默认值0表示CBR模式。 Atlas 200I/500 A2推理产品, 默认值0表示VBR模式。 Atlas 推理系列产品, 默认值0表示VBR模式。

返回值说明	ret : int, 错误码。 - 返回0表示成功。 - 返回其它值表示失败。
约束说明	无

2.9.10.16 函数: venc_set_channel_desc_src_rate

C函数原型	aclError aclvencSetChannelDescSrcRate(aclvencChannelDesc *channelDesc, uint32_t srcRate)
Python函数	ret = acl.media.venc_set_channel_desc_src_rate(venc_channel_desc, src_rate)
函数功能	设置视频编码处理通道描述信息的属性。 Atlas 训练系列产品 , 不支持该接口。 Atlas A2训练系列产品/Atlas 800I A2推理产品 , 不支持该接口。
输入说明	venc_channel_desc : int, 视频编码处理通道的描述信息的指针地址。 需提前调用 acl.media.venc_create_channel_desc 接口创建aclvencChannelDesc类型的数据。 src_rate : int, 输入码流帧率, 单位fps。 Atlas 200/300/500 推理产品 , 取值范围0或者[1, 120]。 Atlas 200I/500 A2推理产品 , 取值范围0或者[1, 240]。 Atlas 推理系列产品 , 取值范围0或者[1, 240]。 如果不设置该参数, 默认为30; 如果设置为0, 表示使用默认值, 即30。如果该值和实际输入码流帧率相差太大, 会影响输出码率。
返回值说明	ret : int, 错误码。 - 返回0表示成功。 - 返回其它值表示失败。
约束说明	无

2.9.10.17 函数: venc_set_channel_desc_max_bit_rate

C函数原型	aclError aclvencSetChannelDescMaxBitRate(aclvencChannelDesc *channelDesc, uint32_t maxBitRate)
Python函数	ret = acl.media.venc_set_channel_desc_max_bit_rate(venc_channel_desc, max_bit_rate)

函数功能	设置视频编码处理通道描述信息的属性。 Atlas 训练系列产品 ，不支持该接口。 Atlas A2训练系列产品/Atlas 800I A2推理产品 ，不支持该接口。
输入说明	venc_channel_desc : int，视频编码处理通道的描述信息的指针地址。 需提前调用 acl.media.venc_create_channel_desc 接口创建 aclvencChannelDesc 类型的数据。 max_bit_rate : int，输出码率，单位kbps。 Atlas 200/300/500 推理产品 ，取值范围0或[10, 30000]。如果不设置该参数，默认为300；如果设置为0，表示使用默认值，即300。 Atlas 200I/500 A2推理产品 ，取值范围[2, 614400]，如果不设置该参数，默认为2000；如果设置为0，表示使用默认值，即2000。 Atlas 推理系列产品 ，取值范围[2, 614400]，如果不设置该参数，默认为2000；如果设置为0，表示使用默认值，即2000。
返回值说明	ret : int，错误码。 - 返回0表示成功。 - 返回其它值表示失败。
约束说明	无

2.9.10.18 函数：venc_get_channel_desc_channel_id

C函数原型	<code>uint32_t aclvencGetChannelDescChannelId(const aclvencChannelDesc *channelDesc);</code>
Python函数	<code>channel_id = acl.media.venc_get_channel_desc_channel_id(venc_channel_desc)</code>
函数功能	获取视频编码处理通道的描述信息：通道号。 Atlas 训练系列产品 ，不支持该接口。 Atlas A2训练系列产品/Atlas 800I A2推理产品 ，不支持该接口。
输入说明	venc_channel_desc : int，视频数据编码处理通道的描述信息的指针地址。
返回值说明	channel_id : int，通道号，默认为0。

2.9.10.19 函数：venc_get_channel_desc_thread_id

C函数原型	<code>uint64_t aclvencGetChannelDescThreadId(const aclvencChannelDesc *channelDesc);</code>

Python函数	<code>thread_id = acl.media.venc_get_channel_desc_thread_id(venc_channel_desc)</code>
函数功能	获取视频编码处理通道的描述信息：回调线程ID。 Atlas 训练系列产品 ，不支持该接口。 Atlas A2训练系列产品/Atlas 800I A2推理产品 ，不支持该接口。
输入说明	venc_channel_desc : int，视频数据编码处理通道的描述信息的指针地址。
返回值说明	thread_id : int，回调线程ID。

2.9.10.20 函数：venc_get_channel_desc_callback

C函数原型	<code>aclvencCallback aclvencGetChannelDescCallback(const aclvencChannelDesc *channelDesc);</code>
Python函数	<code>callback = acl.media.venc_get_channel_desc_callback(venc_channel_desc)</code>
函数功能	获取视频编码处理通道的描述信息：编码回调函数。 Atlas 训练系列产品 ，不支持该接口。 Atlas A2训练系列产品/Atlas 800I A2推理产品 ，不支持该接口。
输入说明	venc_channel_desc : int，视频数据编码处理通道的描述信息的指针地址。
返回值说明	callback : python函数对象，编码回调函数。 Python侧格式如下： <pre>def venc_call_back_func(dvpp_pic_desc, dvpp_stream_desc, user_data): pass</pre>

2.9.10.21 函数：venc_get_channel_desc_entype

C函数原型	<code>acldvppStreamFormat aclvencGetChannelDescEnType(const aclvencChannelDesc *channelDesc);</code>
Python函数	<code>entype = acl.media.venc_get_channel_desc_entype(venc_channel_desc)</code>
函数功能	获取视频编码处理通道的描述信息：视频编码协议。 Atlas 训练系列产品 ，不支持该接口。 Atlas A2训练系列产品/Atlas 800I A2推理产品 ，不支持该接口。
输入说明	venc_channel_desc : int，视频数据编码处理通道的描述信息的指针地址。

返回值说明	entype : int, 视频编码协议, 具体请参见 2.16.7 acldvppStreamFormat 。

2.9.10.22 函数: venc_get_channel_desc_pic_format

C函数原型	<code>acldvppPixelFormat aclvencGetChannelDescPicFormat(const aclvencChannelDesc *channelDesc);</code>
Python函数	<code>in_pic_format = acl.media.venc_get_channel_desc_pic_format(venc_channel_desc)</code>
函数功能	获取视频编码处理通道的描述信息: 输入图像格式。 Atlas 训练系列产品 , 不支持该接口。 Atlas A2训练系列产品/Atlas 800I A2推理产品 , 不支持该接口。
输入说明	venc_channel_desc : int, 视频数据编码处理通道的描述信息的指针地址。
返回值说明	in_pic_format : int, 输入图像格式。

2.9.10.23 函数: venc_get_channel_desc_pic_width

C函数原型	<code>uint32_t aclvencGetChannelDescPicWidth(const aclvencChannelDesc *channelDesc);</code>
Python函数	<code>in_pic_width = acl.media.venc_get_channel_desc_pic_width(venc_channel_desc)</code>
函数功能	获取视频编码处理通道的描述信息: 图片宽度。 Atlas 训练系列产品 , 不支持该接口。 Atlas A2训练系列产品/Atlas 800I A2推理产品 , 不支持该接口。
输入说明	venc_channel_desc : int, 视频数据编码处理通道的描述信息的指针地址。
返回值说明	in_pic_width : int, 图片宽度。

2.9.10.24 函数: venc_get_channel_desc_pic_height

C函数原型	<code>uint32_t aclvencGetChannelDescPicHeight(const aclvencChannelDesc *channelDesc);</code>
Python函数	<code>in_pic_height = acl.media.venc_get_channel_desc_pic_height(venc_channel_desc)</code>

函数功能	获取视频编码处理通道的描述信息：图片高度。 Atlas 训练系列产品 ，不支持该接口。 Atlas A2训练系列产品/Atlas 800I A2推理产品 ，不支持该接口。
输入说明	venc_channel_desc : int，视频数据编码处理通道的描述信息的指针地址。
返回值说明	in_pic_height : int，图片高度。

2.9.10.25 函数：venc_get_channel_desc_key_frame_interval

C函数原型	<code>uint32_t aclvencGetChannelDescKeyFrameInterval(const aclvencChannelDesc *channelDesc)</code>
Python函数	<code>key_frame_interval = acl.media.venc_get_channel_desc_key_frame_interval(venc_channel_desc)</code>
函数功能	获取视频编码处理通道的描述信息：关键帧间隔。 Atlas 训练系列产品 ，不支持该接口。 Atlas A2训练系列产品/Atlas 800I A2推理产品 ，不支持该接口。
输入说明	venc_channel_desc : int，视频数据编码处理通道的描述信息的指针地址。
返回值说明	key_frame_interval : int，关键帧间隔，不能为0。

2.9.10.26 函数：venc_get_channel_desc_buf_addr

C函数原型	<code>void *aclvencGetChannelDescBufAddr(const aclvencChannelDesc *channelDesc)</code>
Python函数	<code>buf_addr = acl.media.venc_get_channel_desc_buf_addr(venc_channel_desc)</code>
函数功能	获取视频编码处理通道描述信息的属性：编码输出缓存地址。 Atlas 训练系列产品 ，不支持该接口。 Atlas A2训练系列产品/Atlas 800I A2推理产品 ，不支持该接口。
输入说明	venc_channel_desc : int，视频数据编码处理通道的描述信息的指针地址。
返回值说明	buf_addr : int，编码输出缓存地址。 在 Atlas 200/300/500 推理产品 上，当前版本获取该参数的值为无效值，不支持获取该参数值。

2.9.10.27 函数: venc_get_channel_desc_buf_size

C函数原型	<code>uint32_t aclvencGetChannelDescBufSize(const aclvencChannelDesc *channelDesc)</code>
Python函数	<code>buf_size = acl.media.venc_get_channel_desc_buf_size(venc_channel_desc)</code>
函数功能	获取视频编码处理通道描述信息的属性：编码输出缓存大小。 Atlas 训练系列产品 ，不支持该接口。 Atlas A2训练系列产品/Atlas 800I A2推理产品 ，不支持该接口。
输入说明	venc_channel_desc : int，视频数据编码处理通道的描述信息的指针地址。
返回值说明	buf_size : int，编码输出缓存大小，单位为Byte。 在 Atlas 200/300/500 推理产品 上，当前版本获取该参数的值为无效值，不支持获取该参数值。

2.9.10.28 函数: venc_get_channel_desc_rc_mode

C函数原型	<code>uint32_t aclvencGetChannelDescRcMode(const aclvencChannelDesc *channelDesc)</code>
Python函数	<code>rc_mode = acl.media.venc_get_channel_desc_rc_mode(venc_channel_desc)</code>
函数功能	获取视频编码处理通道的描述信息。 Atlas 训练系列产品 ，不支持该接口。 Atlas A2训练系列产品/Atlas 800I A2推理产品 ，不支持该接口。
输入说明	venc_channel_desc : int，视频数据编码处理通道的描述信息的指针地址。
返回值说明	rc_mode : int，指定码率控制模式。 • 1表示变码率VBR模式。 • 2表示定码率CBR模式。

2.9.10.29 函数: venc_get_channel_desc_src_rate

C函数原型	<code>uint32_t aclvencGetChannelDescSrcRate(const aclvencChannelDesc *channelDesc)</code>
Python函数	<code>src_rate = acl.media.venc_get_channel_desc_src_rate(venc_channel_desc)</code>

函数功能	获取视频编码处理通道的描述信息。 Atlas 训练系列产品 ，不支持该接口。 Atlas A2训练系列产品/Atlas 800I A2推理产品 ，不支持该接口。
输入说明	venc_channel_desc : int，视频数据编码处理通道的描述信息的指针地址。
返回值说明	src_rate : int，输入码流帧率，单位fps。

2.9.10.30 函数: venc_get_channel_desc_max_bit_rate

C函数原型	<code>uint32_t aclvencGetChannelDescMaxBitRate(const aclvencChannelDesc *channelDesc)</code>
Python函数	<code>max_bit_rate = acl.media.venc_get_channel_desc_max_bit_rate(venc_channel_desc)</code>
函数功能	获取视频编码处理通道的描述信息。 Atlas 训练系列产品 ，不支持该接口。 Atlas A2训练系列产品/Atlas 800I A2推理产品 ，不支持该接口。
输入说明	venc_channel_desc : int，视频数据编码处理通道的描述信息的指针地址。
返回值说明	max_bit_rate : int，输出码率，单位kbps。

2.9.10.31 函数: venc_set_channel_desc_param

C函数原型	<code>aclError aclvencSetChannelDescParam(aclvencChannelDesc *channelDesc, aclvencChannelDescParamType paramType, size_t length, const void *param)</code>
Python函数	<code>ret = acl.media.venc_set_channel_desc_param(venc_channel_desc, param_type, param)</code>
函数功能	设置视频编码处理通道的属性参数接口。 Atlas 训练系列产品 ，不支持该接口。 Atlas A2训练系列产品/Atlas 800I A2推理产品 ，不支持该接口。

输入说明	venc_channel_desc: int, 视频数据编码处理通道的描述信息的指针地址。 param_type: int, 属性参数的类型, 参考2.16.26 aclvencChannelDescParamType。 param: 属性参数值。 - 属性参数的类型为uint64_t或uint32_t时, 此处为对应的参数值。 - 属性参数的类型为回调函数时, 此处为python函数对象, 编码回调函数。 python侧格式如下: <pre>def venc_call_back_func(dvpp_pic_desc, dvpp_stream_desc, user_data): pass</pre> - 属性参数的类型为指针时, 此处为指针指向的地址。
返回值说明	ret: int, 错误码。 - 返回0表示成功。 - 返回其它值表示失败。

2.9.10.32 函数: venc_get_channel_desc_param

C函数原型	aclError aclvencGetChannelDescParam(const aclvencChannelDesc *channelDesc, aclvencChannelDescParamType paramType, size_t length, size_t *paramRetSize, void *param)
Python函数	attr, ret = acl.media.venc_get_channel_desc_param(venc_channel_desc, param_type)
函数功能	获取视频编码处理通道的描述信息。 Atlas 训练系列产品, 不支持该接口。 Atlas A2训练系列产品/Atlas 800I A2推理产品, 不支持该接口。
输入说明	venc_channel_desc: int, 视频数据编码处理通道的描述信息的指针地址。 param_type: int, 属性参数的类型, 参考2.16.26 aclvencChannelDescParamType。

返回值说明	attr: 属性值。 - 属性参数的类型为uint64_t或uint32_t时，此处为对应的属性值。 - 属性参数的类型为回调函数时，此处为python函数对象，编码回调函数。 python侧格数如下： <pre>def venc_call_back_func(dvpp_pic_desc, dvpp_stream_desc, user_data): pass</pre> - 属性参数的类型为指针时，此处为指针指向的地址。 ret: int，错误码。 - 返回0表示成功。 - 返回其它值表示失败。

2.9.10.33 函数：venc_create_frame_config

C函数原型	aclvencFrameConfig *aclvencCreateFrameConfig()
Python函数	frame_config = acl.media.venc_create_frame_config()
函数功能	创建aclvencFrameConfig类型的数据，创建VENC编码时的单帧编码配置参数。 如需销毁aclvencFrameConfig类型的数据，请参见acl.media.venc_destroy_frame_config。 Atlas 训练系列产品，不支持该接口。 Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。
返回值说明	frame_config: int。 - 返回非0值，表示成功。 - 返回0，表示失败。

2.9.10.34 函数：venc_destroy_frame_config

C函数原型	aclError aclvencDestroyFrameConfig(aclvencFrameConfig *config)
Python函数	ret = acl.media.venc_destroy_frame_config(frame_config)
函数功能	销毁通过acl.media.venc_create_frame_config接口创建的aclvencFrameConfig类型的数据。 Atlas 训练系列产品，当前版本不支持该接口。 Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。
输入说明	frame_config: int，指定需要销毁的aclvencFrameConfig类型数据的指针地址。

返回值说明	ret : int, 错误码。 - 返回0表示成功。 - 返回其它值表示失败。

2.9.10.35 函数: venc_set_frame_config_force_i_frame

C函数原型	aclError aclvencSetFrameConfigForceIframe(aclvencFrameConfig *config, uint8_t forceIframe);
Python函数	ret = acl.media.venc_set_frame_config_force_i_frame(config, force_i_frame)
函数功能	设置单帧编码配置参数。 Atlas 训练系列产品 , 当前版本不支持该接口。 Atlas A2训练系列产品/Atlas 800I A2推理产品 , 不支持该接口。
输入说明	config : int, 表示单帧编码配置数据的指针地址。 需提前调用 acl.media.venc_create_frame_config 接口创建aclvencFrameConfig类型的数据。 force_i_frame : int, 是否强制重新开始I帧间隔。 0: 不强制。 1: 强制重新开始I帧。
返回值说明	ret : int, 错误码。 - 返回0表示成功。 - 返回其它值表示失败。

2.9.10.36 函数: venc_set_frame_config_eos

C函数原型	aclError aclvencSetFrameConfigEos(aclvencFrameConfig *config, uint8_t eos)
Python函数	ret = acl.media.venc_set_frame_config_eos(config, eos)
函数功能	设置单帧编码配置参数。 Atlas 训练系列产品 , 当前版本不支持该接口。 Atlas A2训练系列产品/Atlas 800I A2推理产品 , 不支持该接口。

输入说明	config : int, 表示单帧编码配置数据的指针地址。 需提前调用 acl.media.venc_create_frame_config 接口创建aclvencFrameConfig类型的数据。 eos : int, 是否为结束帧。 0: 不是结束帧。 1: 是结束帧。
返回值说明	ret : int, 错误码。 - 返回0表示成功。 - 返回其它值表示失败。

2.9.10.37 函数: venc_get_frame_config_force_i_frame

C函数原型	uint8_t aclvencGetFrameConfigForceIframe(const aclvencFrameConfig *config);
Python函数	force_i_frame = acl.media.venc_get_frame_config_force_i_frame (config)
函数功能	获取单帧编码配置参数。 Atlas 训练系列产品 , 当前版本不支持该接口。 Atlas A2训练系列产品/Atlas 800I A2推理产品 , 不支持该接口。
输入说明	config : int, 单帧编码配置数据的指针地址。
返回值说明	force_i_frame : int, 是否强制重新开始I帧间隔。 0: 不强制重新开始。 1: 强制重新开始I帧。

2.9.10.38 函数: venc_get_frame_config_eos

C函数原型	uint8_t aclvencGetFrameConfigEos(const aclvencFrameConfig *config)
Python函数	eos = acl.media.venc_get_frame_config_eos (config)
函数功能	获取单帧编码配置参数。 Atlas 训练系列产品 , 当前版本不支持该接口。 Atlas A2训练系列产品/Atlas 800I A2推理产品 , 不支持该接口。
输入说明	config : int, 单帧编码配置数据的指针地址。
返回值说明	eos : int, 是否为结束帧。 0: 不是结束帧。 1: 是结束帧。

2.9.11 PNGD 功能

2.9.11.1 功能及约束说明

功能说明

PNGD (PNG Decoder) 功能：实现PNG格式图片的硬件解码。

PNGD在解码图片时，支持按源图片格式解码。在调用PNGD解码接口时，直接将输出图片格式配置为“PIXEL_FORMAT_UNKNOWN”，输出格式默认按源图片格式输出。

源图片格式解码是指解码前后图片的编码格式保持一致，例如解码前输入图片格式为RGB，解码后输出图片格式为RGB888。

图片分辨率约束

- 输入图片分辨率
最大分辨率4096 * 4096，最小分辨率32 * 32。
- 输出图片分辨率
PNGD只对图片解码，不会改变图片分辨率，因此输出与输入的图片分辨率保持一致。

图片格式、宽高对齐、内存约束

- 实现图片解码功能时，需调用[acl.media.dvpp_malloc](#)接口申请Device上的输入、输出内存，调用[acl.media.dvpp_free](#)接口释放输入、输出内存，这部分内存的生命周期由用户自行管理。
- 输入内存的大小就是指实际的输入图片所占用的大小。
 - 输出内存的大小可调用[acl.media.dvpp_png_predict_dec_size](#)接口预估。

 说明

输出图片格式的定义请参见[2.16.6 aclDvppPixelFormat](#)，宽Stride、高Stride等概念请参见[2.9.2 基本概念](#)。

表 2-33 图片格式、宽高对齐、内存大小约束

输入图片格式	输出图片格式	输出图片宽、高对齐要求	输出图片宽Stride、高Stride、内存大小要求
RGB	RGB888	无对齐要求	宽Stride为宽128对齐后再乘以3的值。 高Stride为高16对齐后的值。 输出内存的大小可调用 acl.media.dvpp_png_predict_dec_size 接口预估。

输入图片格式	输出图片格式	输出图片宽、高对齐要求	输出图片宽Stride、高Stride、内存大小要求
GRAY	RGB888	无对齐要求	宽Stride为宽128对齐后再乘以3的值。 高Stride为高16对齐后的值。 输出内存的大小可调用 acl.media.dvpp_png_predict_dec_size 接口预估。
RGBA	RGB888	无对齐要求	宽Stride为宽128对齐后再乘以3的值。 高Stride为高16对齐后的值。 输出内存的大小可调用 acl.media.dvpp_png_predict_dec_size 接口预估。
	RGBA8888 8bit	无对齐要求	宽Stride为宽128对齐后再乘以4的值。 高Stride为高16对齐后的值。 输出内存的大小可调用 acl.media.dvpp_png_predict_dec_size 接口预估。
AGRAY	RGB888	无对齐要求	宽Stride为宽128对齐后再乘以3的值。 高Stride为高16对齐后的值。 输出内存的大小可调用 acl.media.dvpp_png_predict_dec_size 接口预估。
	RGBA8888 8bit	无对齐要求	宽Stride为宽128对齐后再乘以4的值。 高Stride为高16对齐后的值。 输出内存的大小可调用 acl.media.dvpp_png_predict_dec_size 接口预估。

2.9.11.2 性能指标说明

本节仅描述各产品对应的昇腾AI处理器的性能指标参考数据，各产品由于其内存、CPU等配置不同，在性能规格上会存在差异，详细产品的性能规格请查阅[昇腾社区](#)的官方说明。

本节以单个昇腾AI处理器为例，描述基本场景性能指标参考，其它分辨率可以等量估算。

下表中的1路对应一个通道，一个通道对应一个线程；各分辨率的含义为：1080p指分辨率为1920 * 1080的图片，4K指分辨率为3840 * 2160的图片。

性能指标说明（Atlas 200/300/500 推理产品）

以下性能数据，是基于一个Stream上下发10个异步媒体数据处理任务后，执行一次[acl.rt.synchronize_stream](#)接口。

表 2-34 Atlas 200/300/500 推理产品的昇腾 AI 处理器性能指标参考数据

场景举例	总帧率
1080p * n路 (1 ≤ n ≤ 5)	n * 4fps
1080p * n路 (n ≥ 6)	24fps
4k * n路 (1 ≤ n ≤ 5)	n * 1fps
4k * n路 (n ≥ 6)	6fps

性能指标说明（Atlas 推理系列产品）

以下性能数据，是基于一个Stream上下发10个异步媒体数据处理任务后，执行一次[acl.rt.synchronize_stream](#)接口，且Host服务器配置为：CPU主频为Intel(R) Xeon(R) CPU E5-2680 v4 @ 2.40GHz，CPU核心数为14。

表 2-35 Atlas 推理系列产品的昇腾 AI 处理器性能指标参考数据

场景举例	总帧率
1080p * n路 (1 ≤ n ≤ 5)	n * 4fps
1080p * n路 (n ≥ 6)	24fps
4k * n路 (1 ≤ n ≤ 5)	n * 1fps
4k * n路 (n ≥ 6)	6fps

性能指标说明（Atlas 训练系列产品）

以下性能数据，是基于一个Stream上下发10个异步媒体数据处理任务后，执行一次[acl.rt.synchronize_stream](#)接口。

表 2-36 Atlas 训练系列产品的昇腾 AI 处理器性能指标参考数据

场景举例	总帧率
1080p * n路 (1 ≤ n ≤ 24)	n * 4fps
1080p * n路 (n > 24)	96fps

场景举例	总帧率
4k * n路 ($1 \leq n \leq 24$)	$n * 1\text{fps}$
4k * n路 ($n > 24$)	24fps

性能指标说明 (Atlas 200I/500 A2 推理产品)

以下性能数据，是基于一个Stream上下发10个异步媒体数据处理任务后，执行一次[acl.rt.synchronize_stream](#)接口。

表 2-37 Atlas 200I/500 A2 推理产品的昇腾 AI 处理器性能指标参考数据

场景举例	总帧率
1080p * n路 ($1 \leq n \leq 5$)	$n * 4\text{fps}$
1080p * n路 ($n \geq 6$)	24fps
4k * n路 ($1 \leq n \leq 5$)	$n * 1\text{fps}$
4k * n路 ($n \geq 6$)	6fps

性能指标说明 (Atlas A2 训练系列产品/Atlas 800I A2 推理产品)

以下性能数据，是基于一个Stream上下发10个异步媒体数据处理任务后，执行一次[acl.rt.synchronize_stream](#)接口。

表 2-38 Atlas A2 训练系列产品/Atlas 800I A2 推理产品的昇腾 AI 处理器性能指标参考数据

场景举例	总帧率
1080p * n路 ($1 \leq n \leq 5$)	$n * 4\text{fps}$
1080p * n路 ($n \geq 6$)	24fps
4k * n路 ($1 \leq n \leq 5$)	$n * 1\text{fps}$
4k * n路 ($n \geq 6$)	6fps

2.9.11.3 函数：dvpp_png_decode_async

C函数原型	<code>aclError aclDvppPngDecodeAsync(aclDvppChannelDesc *channelDesc, const void *data, uint32_t size, aclDvppPicDesc *outputDesc, aclrtStream stream)</code>

Python函数	<code>ret = acl.media.dvpp_png_decode_async(dvpp_channel_desc, data, size, output_desc, stream)</code>
函数功能	解码PNG图片，异步接口，支持格式请参见表2-39。
输入说明	dvpp_channel_desc: int，指定通道描述信息的指针地址。与调用acl.media.dvpp_create_channel接口创建通道时指定的“dvpp_channel_desc”保持一致。 data: int，输入图片的内存地址的指针地址。 size: int，输入图片的实际数据大小，单位Byte。 output_desc: int，输出图片信息的指针地址。 作为输入时，需要用户调用如下接口： - 调用acl.media.dvpp_create_pic_desc接口创建图片描述信息。 - 调用acl.media.dvpp_set_pic_desc系列接口设置输出图片的内存地址、内存大小、格式。 - 输出图片的内存大小可提前调用acl.media.dvpp_png_predict_dec_size接口获取。 - PNG原图信息（例如宽、高）可通过acl.media.dvpp_png_get_image_info接口获取。 - 输出图片格式、内存等要求，请参见2.9.11.1 功能及约束说明。 作为输出时，用户可以从内存地址中获取解码后的输出图片数据。 stream: int，指定Stream。
返回值说明	ret: int，错误码。 - 返回0表示成功。 - 返回其它值表示失败。
约束说明	本接口是异步接口，调用接口成功仅表示任务下发成功，不表示任务执行成功。调用该接口后，需调用同步等待接口（例如， acl.rt.synchronize_stream ）确保任务已执行完成。
参考资源	接口调用流程及示例，参见PNGD图片解码。

表 2-39 支持格式

功能	输入图片格式	输出图片格式
调用本接口时将输出图片格式设置为PIXEL_FORMAT_UNKNOWN 表示不指定输出图片格式，输出图片格式与输入图片格式一致。	RGB/GRAY	RGB

功能	输入图片格式	输出图片格式
调用本接口时将输出图片格式设置为 PIXEL_FORMAT_UNKNOWN 表示不指定输出图片格式，输出图片格式与输入 图片格式一致。	RGBA/AGRAY	RGBA
调用本接口时将输出图片格式设置为 PIXEL_FORMAT_RGB_888，表示指定输出图片格 式，将输入图片格式转换为输出图片格式。	RGB/RGBA/ GRAY/AGRAY	RGB

2.9.11.4 函数：dvpp_png_get_image_info

C函数原型	<code>aclError aclDvppPngGetImageInfo(const void *data, uint32_t dataSize, uint32_t *width, uint32_t *height, int32_t *components)</code>
Python函数	<code>width, height, components, ret = acl.media.dvpp_png_get_image_info(data, data_size)</code>
函数功能	从存放PNG图片数据的内存中读取PNG图片的宽、高。
输入说明	data : int，存放PNG图片数据内存地址的指针地址。 data_size : int，内存大小，单位为Byte。
返回值说明	width : int，图片的宽。 height : int，图片的高。 components : int，颜色通道个数。 ret : int，错误码。 • 返回0表示成功。 • 返回其它值表示失败。

2.9.11.5 函数：dvpp_png_predict_dec_size

C函数原型	<code>aclError aclDvppPngPredictDecSize(const void *data, uint32_t dataSize, aclDvppPixelFormat outputPixelFormat, uint32_t *decSize)</code>
Python函数	<code>dec_size ,ret = acl.media.dvpp_png_predict_dec_size(data, data_size, output_pixel_format)</code>
函数功能	根据存放PNG图片数据的内存计算出PNG图片解码后所需的输出内存的大小。

输入说明	data: int, 存放PNG图片数据的内存的地址的指针地址。 data_size: int, 内存大小, 单位为Byte。 output_pixel_format: int, 解码后的输出图片的格式, 支持设置如下格式: • PIXEL_FORMAT_UNKNOWN: 按占用内存大小最大的PNG解码输出格式计算内存大小。 • PIXEL_FORMAT_RGB_888 • PIXEL_FORMAT_RGBA_8888
返回值说明	dec_size: int, PNG图片解码后所需的输出内存的大小, 单位为Byte。 ret: int, 错误码。 • 返回0表示成功。 • 返回其它值表示失败。
参考资源	接口调用流程及示例, 参见PNGD图片解码。

2.10 媒体数据处理 V2

2.10.1 总体说明

多版本接口差异

- 本手册中媒体数据处理V1版本与媒体数据处理V2版本的接口都是描述处理媒体数据的接口, 用于实现抠图、图片缩放、格式转换等功能, 但两套接口不能混用。
- V2版本的功能比V1版本更多, 如下:
 - JPEGE: V2版本接口支持高级的参数配置, 如huffman表配置。
 - VENC: V2版本接口支持更加细化的码控参数配置和效果调优, 如I/P帧QP、宏块码控等。
 - VDEC: V2版本接口支持更细化的内存控制, 如设置输入码流缓存。
 - 建议使用V2版本中的接口, 保证后续版本接口功能以及业务的连续演进。
 - V1版本中的接口是为了兼容旧版本, 保证使用该部分接口的用户能继续使用, 后续版本不再演进。

典型功能介绍

CANN提供了AIPP和DVPP两种处理图像/视频数据的方式, 本章主要介绍基于DVPP的图像/视频数据处理。

处理方式	描述
AIPP (Artificial Intelligence Pre-Processing)	AIPP人工智能预处理，在AI Core上完成数据预处理，主要功能包括改变图像尺寸（抠图、填充等）、色域转换（转换图像格式）、减均值/乘系数（改变图像像素）等。 AIPP区分为静态AIPP和动态AIPP。您只能选择静态AIPP或动态AIPP中的一种来处理图片，不能同时配置静态AIPP和动态AIPP两种方式。 ● 静态AIPP：模型转换时设置AIPP模式为静态，同时设置AIPP参数，模型生成后，AIPP参数值被保存在离线模型（*.om）中，每次模型推理过程采用固定的AIPP预处理参数（无法修改）。 如果使用静态AIPP方式，多Batch情况下共用同一份AIPP参数，AIPP参数值在使用ATC工具进行模型转换时设置，ATC工具的详细说明请参见《ATC工具使用指南》。 ● 动态AIPP：模型转换时仅设置AIPP模式为动态，每次模型推理前，根据需求，在执行模型前设置动态AIPP参数值，然后在模型执行时可使用不同的AIPP参数。 如果使用动态AIPP方式，多Batch可使用不同的AIPP参数，各Batch所使用的AIPP参数值通过pyACL提供的接口来设置，请参见模型动态AIPP推理中的介绍。
DVPP (Digital Vision Pre-Processing)	DVPP是昇腾AI处理器内置的图像处理单元，通过pyACL媒体数据处理接口提供强大的媒体处理硬加速能力，主要功能包括以下功能： ● VPC (Vision Preprocessing Core)：处理YUV、RGB等格式的图片，包括缩放、抠图、色域转换等。 ● JPEGD (JPEG Decoder)：JPEG压缩格式-->YUV格式的图片解码。 ● JPEGE (JPEG Encoder)：YUV格式-->JPEG压缩格式的图片编码。 ● VDEC (Video Decoder)：H.264/H.265格式-->YUV/RGB格式的视频码流解码。 ● VENC (Video Encoder)：YUV420SP格式-->H.264/H.265格式的视频码流编码。 ● PNGD (PNG Decoder)：PNG格式-->RGB格式的图片解码。 说明 AIPP、DVPP可以分开独立使用，也可以组合使用。组合使用场景下，一般先使用DVPP对图片/视频进行解码、抠图、缩放等基本处理，但由于DVPP硬件上的约束，DVPP处理后的图片格式、分辨率有可能不满足模型的要求，因此还需要再经过AIPP进一步做色域转换、抠图、填充等处理。 例如，在Atlas 200/300/500 推理产品和Atlas 训练系列产品上，由于DVPP视频解码仅支持输出YUV格式的图片，如果模型需要RGB格式的图片，则需要再经过AIPP做色域转换的处理。

功能支持度说明

产品对媒体数据处理V2版本各功能的支持度如下表所示。

产品	VPC	JPEGD	JPEGE	PNGD	VDEC	VENC
Atlas 200/300/500推理产品	x	x	x	x	x	x
Atlas 训练系列产品	x	x	x	x	x	x
Atlas 推理系列产品	√	√	√	√	√	√
Atlas 200I/500 A2推理产品	√	√	√	√	√	√
Atlas A2训练系列产品/Atlas 800I A2推理产品	√	√	√	√	√	x

整体约束说明

使用本章中介绍的接口，有以下注意点：

- 关于内存申请/释放：
 - 实现媒体数据处理的VPC、JPEGD、JPEGE等功能前，若需要申请内存存放输入或输出数据，需调用[2.10.3.3 acl.himpi.dvpp_malloc](#)申请内存、调用[2.10.3.4 acl.himpi.dvpp_free](#)接口释放内存。
 - 调用[a](#)申请出来的内存可以满足媒体数据处理的要求，也可以在其它任务中使用，例如，从性能角度，为了减少拷贝，媒体数据处理的输出作为模型推理的输入，实现内存复用。
 - 但由于媒体数据处理访问的地址空间有限，为确保媒体数据处理时内存足够，除媒体数据处理功能外的其它功能（例如，模型加载），建议调用[内存管理](#)章节下的[acl.rt.malloc](#)接口、[acl.rt.malloc_host](#)接口或[acl.rt.malloc_cached](#)接口申请内存。
对于**Atlas 推理系列产品**：媒体数据处理功能在每个进程内，可访问的地址空间最大为16GB。
对于**Atlas A2训练系列产品/Atlas 800I A2推理产品**：媒体数据处理功能在每个进程内，可访问的地址空间最大为16GB。
- 关于通道的要求：

实现媒体数据处理的各功能前，必须调用接口创建对应功能的通道，请分别参见[2.10.4.1 VPC功能](#)、[2.10.5 VDEC视频解码/JPEGD图像解码接口](#)、[2.10.6 VENC视频/JPEGE图像编码接口](#)、[2.10.7 PNGD图像处理接口](#)章节下的通道创建与销毁接口，查看接口说明以及通道数的最大限制。

通道的创建与销毁会涉及资源的申请与释放，反复创建与销毁通道会影响业务性能，因此建议根据实际场景管理通道，例如，如果有持续VPC图片处理，则创建VPC的通道后，等到所有VPC功能调用完成后，再销毁该VPC通道。

通道数量多，会影响Device的CPU占用率和内存占用，通道数量建议参考各功能章节下的性能指标的路数。

2.10.2 基本概念

表 2-40 概念解释

概念	描述
宽Stride (widthStride)	指一行图像跨距，表示输入图片对齐后的宽，RGB格式和YUV格式的宽Stride计算方式不一样。 各功能对宽Stride的要求不同，请参见对应功能的约束说明： • VPC功能，请参见图片格式、宽高对齐、内存约束。 • JPEGD功能，请参见2.10.5.1 JPEGD功能及约束说明。 • JPEGG功能，请参见2.10.6.1 JPEGG功能及约束说明。 • PNGD功能，请参见2.10.7.1 功能及约束说明。 • VDEC功能，请参见2.10.5.3 VDEC功能及约束说明。 • VENC功能，请参见2.10.6.3 VENC功能及约束说明。
高Stride (heightStride)	指图像在内存中的行数，表示输入图片对齐后的高。 各功能对高Stride的要求不同，请参见对应功能的约束说明： • VPC功能，请参见图片格式、宽高对齐、内存约束。 • JPEGD功能，请参见2.10.5.1 JPEGD功能及约束说明。 • JPEGG功能，请参见2.10.6.1 JPEGG功能及约束说明。 • PNGD功能，请参见2.10.7.1 功能及约束说明。 • VDEC功能，请参见2.10.5.3 VDEC功能及约束说明。 • VENC功能，请参见2.10.6.3 VENC功能及约束说明。
抠图区域	指用户指定的需抠出的图片区域。抠图起始坐标无奇数、偶数限制。 抠图区域最小分辨率为10 * 6，最大分辨率为8192 * 8192。 抠图区域的约束请参见 抠图、贴图约束 。
贴图区域	指在输出图片中用户指定的区域。 贴图区域最小分辨率为10 * 6，最大分辨率为4096 * 4096。 贴图区域的约束请参见 抠图、贴图约束 。
上/左偏移	通过配置上偏移、左偏移、抠图/贴图区域的宽和高可以指定抠图区域或贴图区域的位置。参见 图2-10图2-15 。

2.10.3 公共接口

2.10.3.1 acl.himpi.sys_init

c函数原型	hi_s32 hi_mpi_sys_init(hi_void)
Python函数	ret = acl.himpi.sys_init()

函数功能	用户处理图像、视频等媒体数据之前，需先调用本接口对媒体数据处理系统底层进行初始化。 Atlas 200/300/500 推理产品 ，不支持该接口。 Atlas 训练系列产品 ，不支持该接口。
返回值说明	ret : int, 错误码。 返回0表示成功。 返回非0 表示失败，参见 2.10.9.1 公共返回码 。
约束说明	- 需在申请pyACL运行管理资源后调用本接口。pyACL运行管理资源申请流程请参见运行管理资源申请流程。 - 多个进程同时运行情况下，每个进程都必须调用本接口初始化。 - 一个进程内，可以调用本接口反复初始化，不返回失败。 - 多Device场景下，针对每一个Device，都需要调用acl.himpi.sys_init接口进行初始化。
参考资源	接口调用流程示例，参见媒体数据处理V2。

2.10.3.2 acl.himpi.sys_exit

c函数原型	hi_s32 hi_mpi_sys_exit()
Python函数	ret = acl.himpi.sys_exit()
函数功能	用户处理完图像、视频等媒体数据之后，需先调用本接口对媒体数据处理系统底层进行去初始化。 Atlas 200/300/500 推理产品 ，不支持该接口。 Atlas 训练系列产品 ，不支持该接口。
返回值说明	ret : int, 错误码。 返回0表示成功。 返回非0表示失败，具体请参见 2.10.9.1 公共返回码 。
约束说明	- 可以调用接口反复进行去初始化，不返回失败。 - 调用本接口去初始化之后，会释放系统内部媒体数据处理的资源，因此不能再调用媒体数据处理V2章节下的其它接口（除acl.himpi.sys_init接口外），否则可能出现接口调用异常、返回码错误等情况。 - 需在释放pyACL运行管理资源前调用本接口。pyACL运行管理资源申请流程请参见运行管理资源申请流程。
参考资源	接口调用流程示例，参见媒体数据处理V2。

2.10.3.3 acl.himpi.dvpp_malloc

c函数原型	hi_s32 hi_mpi_dvpp_malloc(hi_u32 dev_id, hi_void **dev_ptr, hi_u64 size)
Python函数	dev_ptr, ret = acl.himpi.dvpp_malloc(dev_id, size)
函数功能	申请Device上的内存，申请的内存满足媒体数据处理的要求（例如，内存首地址128字节对齐）。 Atlas 200/300/500 推理产品，不支持该接口。 Atlas 训练系列产品，不支持该接口。
输入说明	dev_id: Device设备号，预留参数，该参数配置无效。 标准形态下，在Host上调用本接口时，DVPP内部根据pyACL接口acl.rt.set_device的device申请内存。 size: 申请内存的大小，单位Byte。
返回值说明	ret: int，错误码。 • 返回0表示成功。 • 返回非0表示失败，参见2.10.9.1 公共返回码。 dev_ptr: int，指向Device上已分配内存的指针地址。
约束说明	• 调用该接口申请内存后，这部分内存的生命周期由用户自行管理，如果不使用，必须调用acl.himpi.dvpp_free接口及时释放内存。 • 通过该接口申请的内存仅支持在单个进程内使用，不能在多个进程之间共享。 • 调用acl.himpi.dvpp_malloc接口申请内存时，会对用户输入的size按向上对齐成32整数倍后，再多加32字节。 • 媒体数据处理的输出作为模型推理的输入时，若用户使用本接口申请大块内存并自行划分、管理内存时，每段内存需同时满足以下要求： – 内存大小向上对齐成32整数倍 + 32字节（$m = \text{ALIGN\_UP}[\text{len}, 32] + 32$字节）。 – 内存起始地址需满足128字节对齐（$\text{ALIGN\_UP}[m, 128]$）。 说明 len表示某段内存的大小，$\text{ALIGN\_UP}[\text{len}, k]$表示向上按k字节对齐：$((\text{len} - 1) / k + 1) * k$。
参考资源	接口调用流程示例，参见媒体数据处理V2。

2.10.3.4 acl.himpi.dvpp_free

c函数原型	hi_s32 hi_mpi_dvpp_free(hi_void *dev_ptr)
Python函数	ret = acl.himpi.dvpp_free(dev_ptr)

函数功能	释放Device上的内存。 acl.himpi.dvpp_free 接口只能释放通过 acl.himpi.dvpp_malloc 接口申请的内存。 Atlas 200/300/500 推理产品 ，不支持该接口。 Atlas 训练系列产品 ，不支持该接口。
输入说明	dev_ptr : int，待释放内存的指针地址。
返回值说明	ret : int 错误码。 • 返回0表示成功。 • 返回非0表示失败。参见2.10.9.1 公共返回码
参考资源	接口调用流程示例，参见媒体数据处理V2。

2.10.3.5 acl.himpi.dvpp_get_image_info

c函数原型	hi_s32 hi_mpi_dvpp_get_image_info(hi_payload_type img_type, const hi_vdec_stream *stream, hi_img_info *img_info);
Python函数	img_info, ret = acl.himpi.dvpp_get_image_info(img_type, stream)
函数功能	根据输入码流，获取按源图格式输出时的图片宽、高、宽Stride、高Stride、解码后占用的内存大小、源图片格式等信息。 该接口会检查图片格式是否支持解码，如果遇到不支持的码流则该接口会返回HI_ERR_VDEC_NOT_SUPPORT错误码，该错误码的详细描述请参见 2.10.9.1 公共返回码 。 Atlas 200/300/500 推理产品 ，不支持该接口。 Atlas 训练系列产品 ，不支持该接口。
输入说明	img_type : int，要解析的码流类型。当前仅支持码流类型为HI_PT_JPEG，具体请参见 2.10.8.2.3 hi_payload_type 。 stream : int，输入码流信息的指针地址。 Atlas 推理系列产品 ，标准形态下，在Host上调用本接口时，该结构体内的addr参数配置的地址必须是Host上的内存地址。 Atlas A2训练系列产品/Atlas 800I A2推理产品 ，在Host上调用本接口时，该结构体内的addr参数配置的地址必须是Host上的内存地址。 Atlas 200I/500 A2推理产品 ，EP模式下，在Host上调用本接口时，该结构体内的addr参数配置的地址必须是Host上的内存地址。 Atlas 200I/500 A2推理产品 ，RC模式下，在Device上调用本接口时，该结构体内的addr参数配置的地址必须是Device上的内存地址。

返回值说明	img_info: dict, 图片信息字典, 包含图片宽、高、宽Stride、高Stride、解码后占用的内存大小、源图片格式等信息。其中, 源图格式通过“pixel_format”参数返回, 具体请参见2.10.8.2.14 hi_img_info。 ret: int, 错误码。 • 返回0, 表示成功。 • 返回非0, 表示失败。参见2.10.9.1 公共返回码。

2.10.3.6 acl.himpi.dvpp_get_version

c函数原型	hi_s32 hi_mpi_dvpp_get_version(hi_s32 *major_version, hi_s32 *minor_version, hi_s32 *patch_version)
Python函数	major_version, minor_version, patch_version, ret = acl.himpi.dvpp_get_version()
函数功能	查询接口版本号, 版本号命名采用: A.B.C模式, 其中, A表示有不兼容修改, B表示新增接口, C表示bug修复。 Atlas 200/300/500 推理产品, 当前版本该接口预留。 Atlas 训练系列产品, 当前版本该接口预留。
返回值说明	major_version: 主版本号, 从1开始, 如果出现接口的不兼容变更时, 加1。 minor_version: 次版本号, 从0开始, 按照迭代周期, 有新增接口时加1。 patch_version: 补丁版本号, 从0开始, 表示本版本仅仅解决了问题, 在“major_version”、“minor_version”不变的情况下加1; 但“major_version”、“minor_version”增加的时候, “patch_version”一般为0。 ret: int, 错误码。 • 返回0表示成功。 • 返回非0表示失败。参见2.10.9.1 公共返回码。

2.10.3.7 acl.himpi.sys_create_epoll

c函数原型	hi_s32 hi_mpi_sys_create_epoll(hi_s32 size, hi_s32 *epoll_fd);
Python函数	epoll_fd, ret = acl.himpi.sys_create_epoll(size)
函数功能	创建新的媒体数据处理的Epoll实例, 使用Epoll编程模型操作媒体数据处理通道的输入/输出数据。 Atlas 200/300/500 推理产品, 当前版本不支持该接口。 Atlas 训练系列产品, 当前版本不支持该接口。
输入说明	size: int, Epoll实例最大可处理的DVPP通道文件句柄个数, 当前该字段为预留, 传入一个正数即可。

返回值说明	epoll_fd: int, Epoll实例句柄的指针地址。句柄个数受限Linux操作系统的open files（一个任务最多可以同时打开的文件个数）资源限制，您可以使用ulimit -al命令查看open files。 ret: int, 错误码。 • 返回0表示成功。 • 返回非0表示失败，参见2.10.9.1 公共返回码。
约束说明	• acl.himpi.sys_create_epoll、acl.himpi.sys_ctl_epoll、acl.himpi.sys_wait_epoll、acl.himpi.sys_close_epoll这几个接口要配合使用。使用epoll编程模型操作DVPP通道的输入/输出数据。 • Atlas 推理系列产品，标准形态下，仅能使用pyACL提供的Epoll接口（例如，acl.himpi.sys_create_epoll），不能使用Linux操作系统的Epoll函数。 • Atlas A2训练系列产品/Atlas 800I A2推理产品，仅能使用pyACL提供的Epoll接口（例如，acl.himpi.sys_create_epoll），不能使用Linux操作系统的Epoll函数。 • Atlas 200I/500 A2推理产品，仅能使用pyACL提供的Epoll接口（例如，acl.himpi.sys_create_epoll），不能使用Linux操作系统的Epoll函数。 • VPC功能不支持使用该接口。

2.10.3.8 acl.himpi.sys_ctl_epoll

c函数原型	hi_s32 hi_mpi_sys_ctl_epoll(hi_s32 epoll_fd, hi_s32 operation, hi_s32 fd, hi_dvpp_epoll_event *event);
Python函数	ret = acl.himpi.sys_ctl_epoll(epoll_fd, operation, fd, event)
函数功能	在媒体数据处理Epoll实例中，对对应的媒体数据处理通道进行控制操作（增、删、改）。 Atlas 200/300/500 推理产品，不支持该接口。 Atlas 训练系列产品，不支持该接口。
输入说明	epoll_fd: int, Epoll实例句柄的指针地址，句柄需提前通过acl.himpi.sys_create_epoll接口创建。 operation: int, 操作类型（增、删、改），请参见2.10.8.2.17 hi_dvpp_epoll_ctl_op。 fd: int, 媒体数据处理通道的文件句柄。VENC场景下，您可以调用acl.himpi.venc_get_fd接口提前获取该文件句柄。VDEC场景下，您可以调用acl.himpi.vdec_get_fd接口提前获取该文件句柄。 event: int, Epoll事件信息的指针地址。该参数用于描述对应媒体数据处理通道事件类型以及关联的用户数据。进行删除操作时，该字段无效。

返回值说明	ret : int, 错误码。 • 返回0, 表示成功。 • 返回非0, 表示失败, 参见2.10.9.1 公共返回码。
约束说明	• acl.himpi.sys_create_epoll、acl.himpi.sys_ctl_epoll、acl.himpi.sys_wait_epoll、acl.himpi.sys_close_epoll这几个接口要配合使用, 实现使用epoll编程模型操作媒体数据处理通道的输入/输出数据。 • VPC功能不支持使用该接口。
参考资源	接口调用流程及示例, 参见JPEG图片编码、VENC视频编码。 接口调用流程及示例, 参见JPEG图片编码。

2.10.3.9 acl.himpi.sys_wait_epoll

c函数原型	hi_s32 hi_mpi_sys_wait_epoll(hi_s32 epoll_fd, hi_dvpp_epoll_event *events, hi_s32 max_events, hi_s32 timeout, hi_s32 *event_num);
Python函数	event, event_num, ret = acl.himpi.sys_wait_epoll(epoll_fd, max_events, timeout)
函数功能	使用Epoll模型进行媒体数据处理通道输入输出操作, 等待媒体数据处理Epoll实例产生事件。 Atlas 200/300/500 推理产品 , 不支持该接口。 Atlas 训练系列产品 , 不支持该接口。
输入说明	epoll_fd : int, 媒体数据处理Epoll实例句柄的指针地址, 句柄通过 acl.himpi.sys_create_epoll 接口创建。 max_events : int, 最大可返回的事件数, 不应超过events数组的元素个数。 timeout : int, 超时时间, 单位是毫秒。 • -1: 永久阻塞。 • 0: 立即返回。 • >0: 配置具体的超时时间。
返回值说明	event : list, 保存产生的事件信息, 具体请参见 2.10.8.2.19 hi_dvpp_epoll_event 。 event_num : int, Epoll实际产生的事件数。 ret : int, 错误码。 • 返回0表示成功。 • 返回非0表示失败, 参见2.10.9.1 公共返回码。

约束说明	• acl.himpi.sys_create_epoll、acl.himpi.sys_ctl_epoll、acl.himpi.sys_wait_epoll、acl.himpi.sys_close_epoll这几个接口要配合使用，实现使用Epoll编程模型操作媒体数据处理通道的输入/输出数据。 • VPC功能不支持使用该接口。
参考资源	接口调用流程及示例，参见JPEG图片编码、VENC视频编码。 接口调用流程及示例，参见JPEG图片编码。

2.10.3.10 acl.himpi.sys_close_epoll

c函数原型	hi_s32 hi_mpi_sys_close_epoll(hi_s32 epoll_fd)
Python函数	ret = acl.himpi.sys_close_epoll(epoll_fd)
函数功能	使用Epoll模型进行媒体数据处理通道输入输出操作，销毁媒体数据处理Epoll实例。 Atlas 200/300/500 推理产品 ，不支持该接口。 Atlas 训练系列产品 ，不支持该接口。
输入说明	epoll_fd : int，待销毁媒体数据处理Epoll实例句柄的指针地址，句柄通过 acl.himpi.sys_create_epoll 接口创建。
返回值说明	ret : int，错误码。 • 返回0表示成功。 • 返回非0表示失败，参见2.10.9.1 公共返回码。
约束说明	• acl.himpi.sys_create_epoll、acl.himpi.sys_ctl_epoll、acl.himpi.sys_wait_epoll、acl.himpi.sys_close_epoll这几个接口要配合使用，实现使用epoll编程模型操作媒体数据处理通道的输入/输出数据。 • VPC功能不支持使用该接口。
参考资源	接口调用流程及示例，参见JPEG图片编码、VENC视频编码。 接口调用流程及示例，参见JPEG图片编码。

2.10.3.11 acl.himpi.sys_get_image_align_info

c函数原型	hi_s32 hi_mpi_sys_get_image_align_info(const hi_module_type mod_type[], const hi_u32 mod_num, const hi_img_base_info *img_base_info, hi_img_align_info *img_align_info)
Python函数	img_align_info, ret = acl.himpi.sys_get_image_align_info(mod_type, mod_num, img_base_info)

函数功能	根据模块、图片宽高计算出宽高对齐后的值以及存放图片数据的内存大小。 Atlas 200/300/500 推理产品，不支持该接口。 Atlas 训练系列产品，不支持该接口。
输入说明	mod_type: list, 模块类型列表。模块类型是整数型常量，具体请参见2.10.8.2.29 hi_module_type。 - 如果数组中只有一个模块，则系统内部按该模块的对齐值来计算。 - 如果数组中有多个模块，则系统内部按多个模块中的最大对齐值来计算。 - 如果将模块类型设置为“HI_MOD_ALL”，则系统内部会从所有模块中选取最大对齐值来计算，便于简化管理各模块的对齐差异，操作简便，但可能占用内存较大。 mod_num: int, 模块类型数组长度。 img_base_info: dict, 图片基本信息字典，具体请参见2.10.8.2.30 hi_img_base_info。
返回值说明	img_align_info: dict, 图片对齐信息、内存大小的字典，具体请参见2.10.8.2.31 hi_img_align_info。 ret: int, 错误码。 - 返回0表示成功。 - 返回非0表示失败，参见2.10.9.1 公共返回码。
约束说明	- 图片宽、高不能小于该模块最小宽高、不能大于最大宽高，图片格式必须是该模块所支持的。 - 各模块支持的图片格式、对齐要求请参见2.10.4 VPC图像处理接口、2.10.5 VDEC视频解码/JPEGD图像解码接口、2.10.6 VENC视频/JPEGE图像编码接口、2.10.7 PNGD图像处理接口下的约束说明。图片格式不支持设置“HI_PIXEL_FORMAT_UNKNOWN”。

2.10.3.12 acl.himpi.sys_set_chn_csc_matrix

c函数原型	hi_s32 hi_mpi_sys_set_chn_csc_matrix(hi_mod_id mode, hi_s32 chn, hi_csc_matrix csc_matrix, hi_csc_coefficient *csc_coefficient);
Python函数	ret = acl.himpi.sys_set_chn_csc_matrix(mode, chn, csc_matrix, csc_coefficient)
函数功能	在处理图像或视频前，YUV与RGB之间格式转换场景下，如果需要调整色域转换参数时，在创建对应图像或视频处理通道后，可调用该接口设置色域转换参数。 Atlas 200/300/500 推理产品，当前版本不支持该接口。 Atlas 训练系列产品，当前版本不支持该接口。

输入说明	mode: int, 模块ID, 当前只支持“HI_ID_VDEC”和“HI_ID_VPC”。 chn: int, 通道号。 - 当“mode”设置为“HI_ID_VDEC”时, 通道号范围请参见2.10.5.5 acl.himpi.vdec_create_chn处“chn”参数的取值范围。 - 当“mode”设置为“HI_ID_VPC”时, 通道号范围请参见2.10.4.4 acl.himpi.vpc_create_chn处“chn”参数的取值范围。 csc_matrix: int, 色域转换矩阵的数据标准, 具体请参见2.10.8.2.20 hi_csc_matrix。 csc_coefficient: dict, 色域转换矩阵参数值字典, 具体请参见2.10.8.2.22 hi_csc_coefficient。 如果将“csc_matrix”参数设置为“HI_CSC_MATRIX_USER”, 则需要使用“csc_coefficient”参数设置色域转换矩阵参数, 否则, 此处传任意值。
返回值说明	ret: int, 错误码。 - 返回0, 表示成功。 - 返回非0, 表示失败, 参见2.10.9.1 公共返回码。
约束说明	- 如果不调用本接口设置色域转换参数, 默认使用BT601 wide标准的色域转换矩阵参数值。 - 如果默认的色域转换参数不适用, 仅需修改一次, 该通道内的其它任务都使用修改后的色域转换参数, 则在创建对应图像或视频处理通道后, 可调用本接口设置色域转换参数, 再调用具体的图像或视频处理接口, 然后再调用对应的图像或视频处理结果获取接口。 - 如果默认的色域转换参数不适用, 不同任务使用不同的色域转换参数, 需多次修改, 则在创建对应图像或视频处理通道后, 在上一次任务执行结束后、下一次任务执行前调用本接口设置色域转换参数(如果上一次任务没有执行结束, 就再次调用本接口设置色域转换参数, 可能会导致任务处理的结果不正确), 再调用具体的图像或视频处理接口, 然后再调用对应的图像或视频处理结果获取接口。 调用顺序示例如下: acl.himpi.sys_set_chn_csc_matrix (第一次修改色域转换参数) acl.himpi.vpc_crop (抠图) acl.himpi.vpc_get_process_result (等待抠图任务结束) acl.himpi.sys_set_chn_csc_matrix (第二次修改色域转换参数) acl.himpi.vpc_resize (缩放) acl.himpi.vpc_get_process_result (等待缩放任务结束)

2.10.3.13 acl.himpi.sys_get_chn_csc_matrix

c函数原型	hi_s32 hi_mpi_sys_get_chn_csc_matrix(hi_mod_id mode, hi_s32 chn, hi_csc_matrix *csc_matrix, hi_csc_coefficient *csc_coefficient);
Python函数	csc_matrix, csc_coefficient, ret = acl.himpi.sys_get_chn_csc_matrix(mode, chn)
函数功能	用户使用该接口获取通道的色域转换矩阵参数值。 Atlas 200/300/500 推理产品 ，不支持该接口。 Atlas 训练系列产品 ，不支持该接口。
输入说明	mode : int, 模块ID, 当前只支持“HI_ID_VDEC”和“HI_ID_VPC”。 chn : int, 通道号。 - 当“mode”设置为“HI_ID_VDEC”时, 通道号范围请参见acl.himpi.vdec_create_chn处“chn”参数的取值范围。 - 当“mode”设置为“HI_ID_VPC”时, 通道号范围请参见acl.himpi.vpc_create_chn处“chn”参数的取值范围。
返回值说明	ret : int, 错误码。 - 返回0, 表示成功。 - 返回非0, 表示失败, 参见2.10.9.1 公共返回码。 csc_matrix : int, 色域转换矩阵的数据标准, 具体请参见 2.10.8.2.20 hi_csc_matrix 。 csc_coefficient : dict, 色域转换矩阵参数值, 具体请参见 2.10.8.2.22 hi_csc_coefficient 。
约束说明	- 在创建通道成功后, 才可以调用本接口获取通道的色域转换矩阵参数值。 - 不调用acl.himpi.sys_set_chn_csc_matrix接口设置色域转换矩阵参数时, 返回默认的BT601 wide标准的色域转换矩阵参数值。

2.10.4 VPC 图像处理接口

2.10.4.1 VPC 功能

Atlas 200/300/500 推理产品上, 不支持本章中的VPC功能。

Atlas 训练系列产品上, 不支持本章中的VPC功能。

功能说明

VPC (Vision Preprocessing Core) 功能如[表2-41](#)所示, 各产品功能支持度请参见[表2-42](#)。

表 2-41 功能介绍

功能	描述
抠图	从输入图片中抠出需要用的图片区域，支持一图多框和多图多框。
缩放	针对不同分辨率的图像，支持8K缩放、非8K缩放。 支持单图裁剪缩放（支持非压缩格式）、一图多框裁剪缩放（支持非压缩格式）。 其它缩放方式，如：原图缩放、等比例缩放（缩放前后图片的宽高比例相同）。
叠加	对抠图操作得到的图片进行缩放后，根据用户设置，将图片放在输出图片的指定区域。 该输出图片支持空白图片（由用户申请的空输出内存产生的）或已有图片（由用户申请输出内存后将已有图片读入输出内存），只有当输出图片是已有图片时，才表示叠加。
拼接	从输入图片中抠多张图片，对抠出的图进行缩放后，放到输出图片的指定区域。
图像金字塔	对原图进行多次高斯滤波之后向下采样产生不同尺寸的图像。
直方图统计	统计图像每个通道（RGB/YUV）的像素值分布。
色彩重映射	根据配置信息将图片从原图映射为另一张图。
边界填充	对图像进行边界填充。
格式转换	支持RGB格式、YUV格式之间的格式转换。
图像灰度化	将彩色图像转化为灰度图像。需注意，输入为灰度图像、输出只能为灰度图像。 实现图像灰度化的操作是输出YUV400格式的输出图片。
Remap变换	根据像素位置LUT对输入图像进行几何形变，典型的用途包括：镜头畸变校正、仿射变换、透视变换，功能示意图请参见图2-18。 形变方式可以用下述公式描述： $dst(x, y) = src(LUT(x, y))$ 。 其中， $dst(x, y)$ 为输出图像在坐标 (x, y) 处的像素值，像素位置 $LUT(x, y)$ 是输出图像在 (x, y) 处的像素对应输入图像中的横、纵坐标值， $src(LUT(x, y))$ 为输入图像在坐标 $LUT(x, y)$ 处的像素值。 通用的像素位置LUT可通过用户提供的map1（横坐标映射矩阵）和map2（纵坐标映射矩阵）生成，仿射变换或透视变换的像素位置LUT也可通过用户提供的3个点对或4个点对信息计算生成。
滤波	对输入图片做滤波处理，当前可支持中值滤波/腐蚀/膨胀/高斯滤波/均值滤波/卷积滤波。
旋转	对输入图片做固定角度的旋转，支持90度/180度/270度。
马赛克	对输入图片做马赛克处理。
覆盖	对输入图片做局部覆盖操作。
画线	对输入图片做画线处理。

功能	描述
添加水印	对输入图片做添加水印的处理。

表 2-42 产品功能支持度

功能	Atlas 推理系列产品	Atlas 200I/500 A2推理产品	Atlas A2训练系列产品/Atlas 800I A2推理产品
抠图	√	√	√
缩放	√	√	√
叠加	√	√	√
拼接	√	√	√
图像金字塔	√	x	x
直方图统计	√	√	√
色彩重映射	√	√	√
边界填充	√	√	√
格式转换	√	√	√
图像灰度化	√	√	√
环视拼接	x	x	x
Remap变换	x	√	√
滤波	x	√	√
旋转	x	√	√
马赛克	x	√	x
覆盖	x	√	x
画线	x	√	x
添加水印	x	√	x

功能示意图（Atlas 推理系列产品）

图 2-10 VPC 功能示意图（抠图+缩放+叠加）

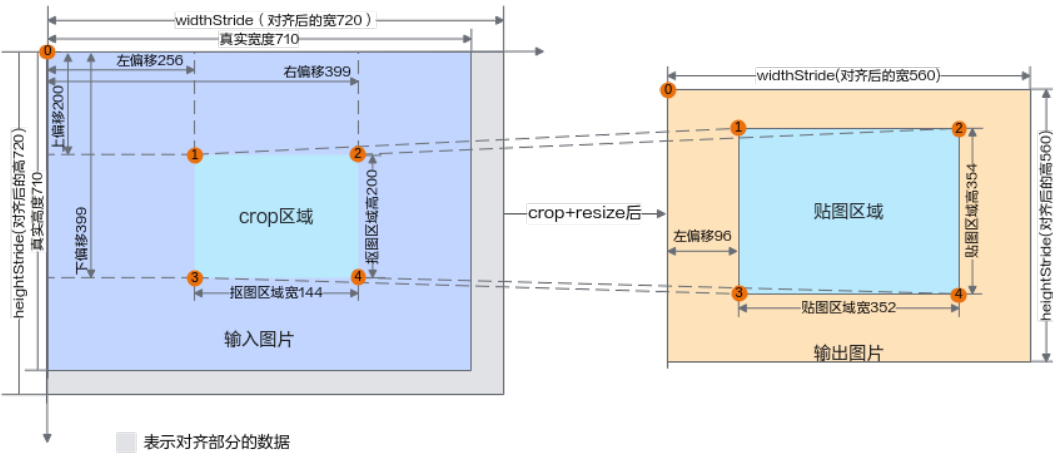


图 2-11 VPC 功能示意图（拼接）

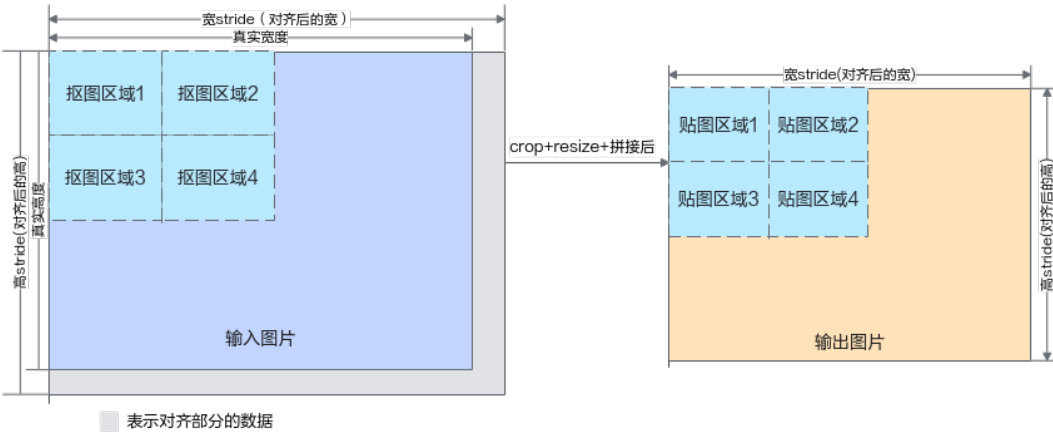


图 2-12 等比例缩放（贴图区域在输出图片的中心位置），即缩放前后图片的宽高比例相同

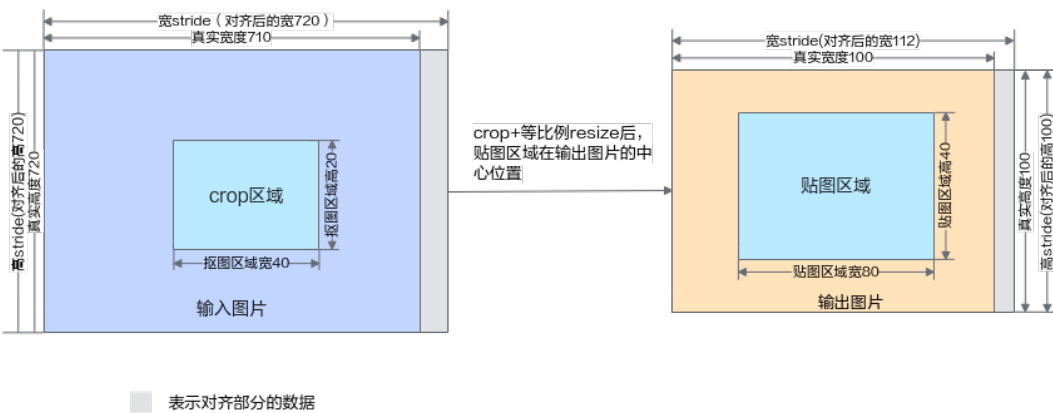


图 2-13 等比例缩放（贴图区域在输出图片的上下居中位置）

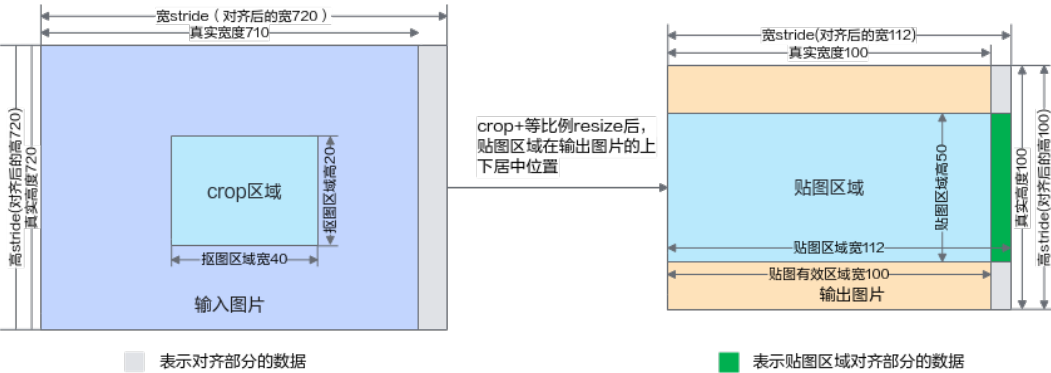
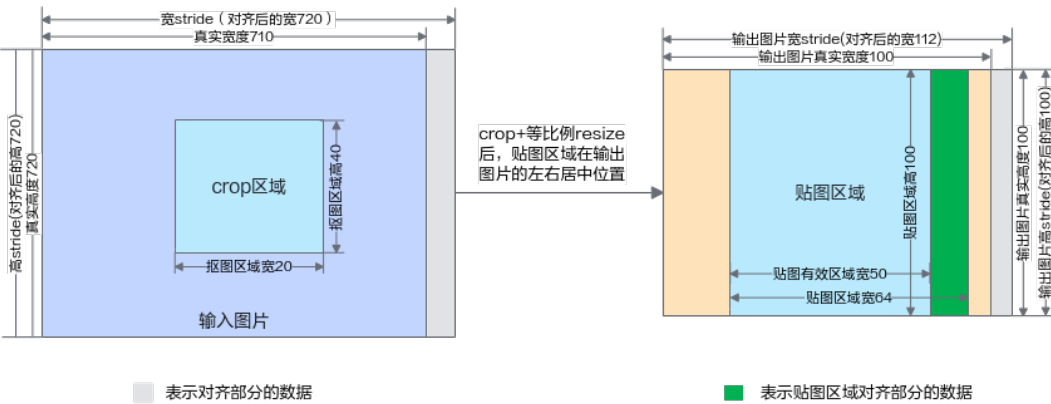


图 2-14 等比例缩放（贴图区域在输出图片的左右居中位置）



功能示意图（Atlas A2 训练系列产品/Atlas 800I A2 推理产品）(Atlas 200I/500 A2 推理产品)

图 2-15 VPC 功能示意图（抠图+缩放+叠加）

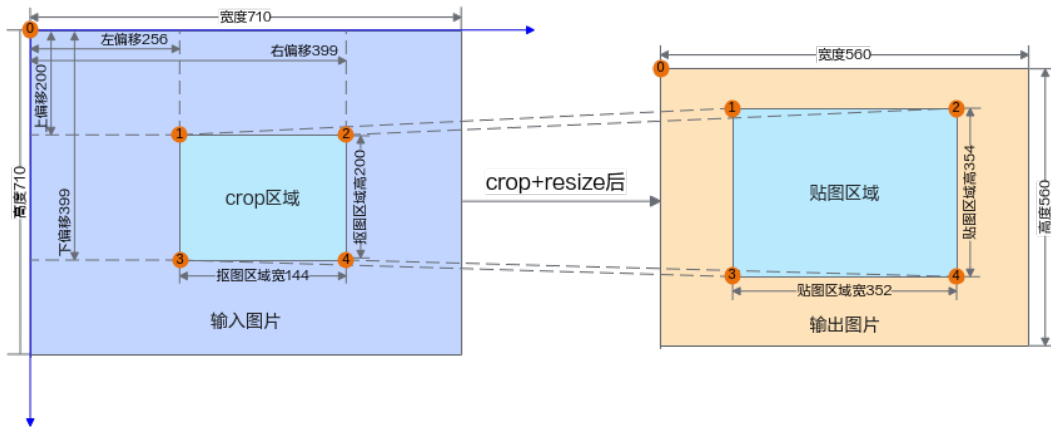


图 2-16 VPC 功能示意图（拼接）

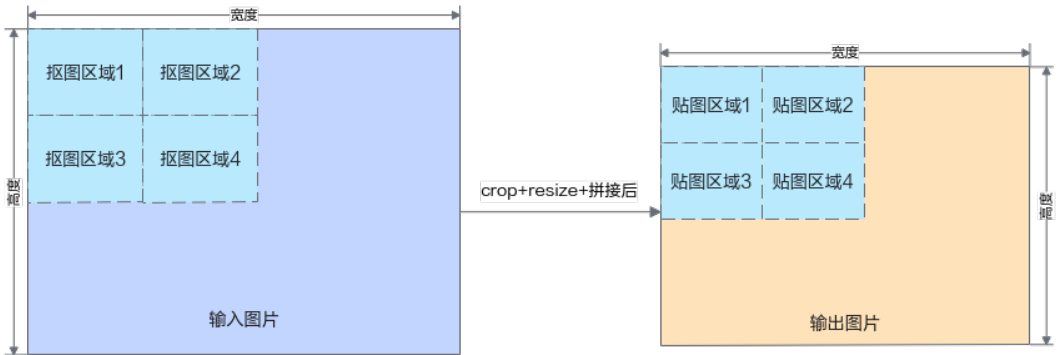


图 2-17 等比例缩放（贴图区域在输出图片的中心位置），即缩放前后图片的宽高比例相同

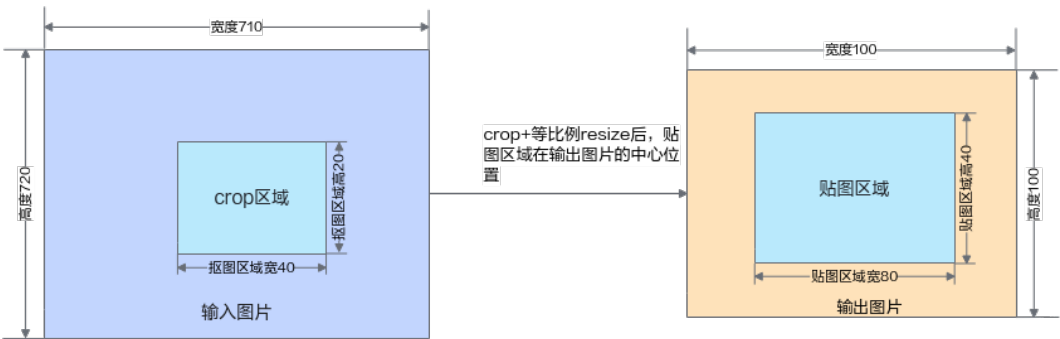
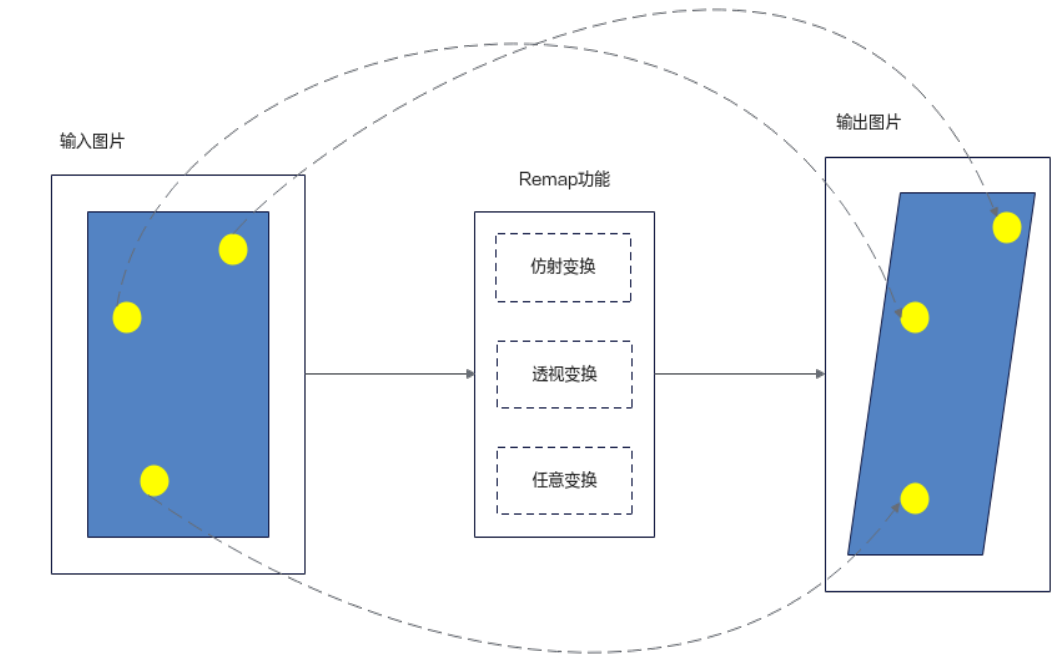


图 2-18 Remap 功能



参考说明

RGB、YUV格式图像的各分量排布示意图。示例：SP图像以YUV420SP为例，Packed和RGB图像以ARGB图像为例。

格式	分辨率	宽stride	高stride	buffer大小					
yuv420sp	4*4	4	4	24					
内存排列									
y11	y12	y13	y14						
y21	y22	y23	y24						
y31	y32	y33	y34						
y41	y42	y43	y44						
u11	v11	u13	v13						
u31	v31	u33	v33						
格式	分辨率	宽stride	高stride	buffer大小					
yuv420sp	4*4	6	6	54					
内存排列 x为无效数据									
y11	y12	y13	y14	x	x				
y21	y22	y23	y24	x	x				
y31	y32	y33	y34	x	x				
y41	y42	y43	y44	x	x				
x	x	x	x	x	x				
x	x	x	x	x	x				
u11	v11	u13	v13	x	x				
u31	v31	u33	v33	x	x				
x	x	x	x	x	x				
格式	分辨率	宽stride	高stride	buffer大小					
argb	2*2	10	4	40					
内存排列 x为无效数据									
a11	r11	g11	b11	a12	r12	g12	b12	x	x
a21	r21	g21	b21	a22	r22	g22	b22	x	x
x	x	x	x	x	x	x	x	x	x
x	x	x	x	x	x	x	x	x	x

2.10.4.2 约束说明

Atlas 200/300/500 推理产品上，该功能为预留功能，暂不支持。

Atlas 训练系列产品上，该功能为预留功能，暂不支持。

图片分辨率约束

- 输入图片分辨率
VPC处理图片时，通用的输入分辨率范围为10 * 6 ~ 8192 * 8192，其中部分接口的分辨率限制可能不同，请以具体接口内的说明为准，请参见[2.10.4 VPC图像处理接口](#)下各接口的说明。其中，当输入图片格式为YUV440SP、YUV440P时，输入图片的宽最大值为4096。
- 输出图片分辨率

产品	分辨率范围
Atlas 推理系列产品	10 * 6 ~ 4096 * 4096

产品	分辨率范围
Atlas A2训练系列产品/Atlas 800I A2推理产品 Atlas 200I/500 A2推理产品	10 * 6 ~ 4096 * 8192

内存申请/释放接口说明

输入、输出内存大小要求请参见[图片格式](#)、[宽高对齐](#)、[内存约束](#)中的计算公式。

产品	内存申请/释放接口
Atlas 推理系列产品	VPC在处理图片时，需调用 acl.himpi.dvpp_malloc 接口申请Device上的输入、输出内存，调用 acl.himpi.dvpp_free 接口释放输入、输出内存，这部分内存的生命周期由用户自行管理。
Atlas A2训练系列产品/ Atlas 800I A2推理产品 Atlas 200I/500 A2推理产品	VPC在处理图片时，支持以下两种内存申请接口： - 调用acl.rt.malloc\acl.rt.free接口申请\释放Device上的输入、输出内存，这部分内存的生命周期由用户自行管理。 - 调用acl.himpi.dvpp_malloc\acl.himpi.dvpp_free接口申请\释放Device上的输入、输出内存，这部分内存的生命周期由用户自行管理。 调用acl.himpi.dvpp_malloc接口申请的内存为媒体数据处理的专用内存，但专用内存的地址空间有限，若关注内存规划或内存资源有限时，建议调用acl.rt.malloc接口申请内存。

图片格式、宽高对齐、内存约束

说明

在调用接口实现VPC功能时：

- 图片格式的定义请参见[hi_pixel_format](#)，宽Stride、高Stride等概念请参见[2.10.2 基本概念](#)。
- 不同版本中，宽Stride、高Stride的最小值、最大值不同，如下：
 - Atlas 推理系列产品**上，宽Stride最小32、最大16384（16384 = 4096 * 4，宽是4096的argb格式的图像，1个像素占用4个字节，一行像素就占用4096 * 4，即宽Stride）；高Stride最小6、最大16384。
 - Atlas 200I/500 A2推理产品**上，宽Stride最小10、高Stride最小6。
 - Atlas A2训练系列产品/Atlas 800I A2推理产品**上，宽Stride最小10、高Stride最小6。
- 各接口对图片格式的要求可能不同，请参见[2.10.4 VPC图像处理接口](#)下各接口的说明。

表 2-43 输入图片格式、宽高对齐、内存大小约束

产品	图片格式	图片宽、高 对齐要求	图片宽 Stride、高 Stride对齐 要求 Atlas 推理 系列产品	图片宽 Stride、高 Stride对齐 要求 Atlas 200I/500 A2推理产品 Atlas A2训 练系列产 品/Atlas 800I A2推 理产品	内存大小要 求 (单位 Byte)
通用格式， 各版本都支持	YUV400 8bit	无对齐要求	宽Stride为 宽16对齐后 的值； 高Stride无 对齐要求， 与高相同即可。	宽Stride无 对齐要求， 与宽相同即可； 高Stride无 对齐要求， 与高相同即可。	宽Stride * 高Stride
	YUV420SP NV12 8bit YUV420SP NV21 8bit	宽2对齐 高2对齐	宽Stride为 宽16对齐后 的值； 高Stride为 高2对齐后 的值。	宽Stride无 对齐要求， 与宽相同即可； 高stride无 对齐要求， 与高相同即可。	宽Stride * 高Stride * 3/2
	YUV422SP 8bit YVU422SP 8bit	宽2对齐 高无对齐要求	宽Stride为 宽16对齐后 的值； 高Stride无 对齐要求， 与高相同即可。	宽Stride无 对齐要求， 与宽相同即可； 高Stride无 对齐要求， 与高相同即可。	宽Stride * 高Stride * 2
	YUV444SP 8bit YVU444SP 8bit	无对齐要求	宽Stride为 宽16对齐后 的值； 高Stride无 对齐要求， 与高相同即可。	宽Stride无 对齐要求， 与宽相同即可； 高Stride无 对齐要求， 与高相同即可。	宽Stride * 高Stride * 3

产品	图片格式	图片宽、高 对齐要求	图片宽 Stride、高 Stride对齐 要求 Atlas 推理 系列产品	图片宽 Stride、高 Stride对齐 要求 Atlas 200I/500 A2推理产品 Atlas A2训 练系列产 品/Atlas 800I A2推 理产品	内存大小要 求 (单位 Byte)
	YUV422Packed YUYV 8bit YUV422Packed UYVY 8bit YUV422Packed YVYU 8bit YUV422Packed VYUY 8bit	宽2对齐 高无对齐要求	宽Stride为宽16对齐后、再乘以2的值； 高Stride无对齐要求，与高相同即可。	宽Stride为宽 * 2的值； 高Stride无对齐要求，与高相同即可。	宽Stride * 高Stride
	YUV444Packed 8bit RGB888 BGR888	无对齐要求	宽Stride为宽16对齐后、再乘以3的值； 高Stride无对齐要求，与高相同即可。	宽Stride为宽 * 3的值； 高Stride无对齐要求，与高相同即可。	宽Stride * 高Stride
	ARGB8888 ABGR8888 RGBA8888 BGRA8888	无对齐要求	宽Stride为宽16对齐后、再乘以4的值； 高Stride无对齐要求，与高相同即可。	宽Stride为宽 * 4的值； 高Stride无对齐要求，与高相同即可。	宽Stride * 高Stride
	YUV440SP 8bit YVU440SP 8bit	宽无对齐要求，但宽 ≤4096 高2对齐	宽Stride为宽16对齐后的值； 高Stride为高2对齐后的值。	宽Stride无对齐要求，与宽相同即可； 高Stride为高2对齐后的值。	宽Stride * 高Stride * 2

产品	图片格式	图片宽、高 对齐要求	图片宽 Stride、高 Stride对齐 要求 Atlas 推理 系列产品	图片宽 Stride、高 Stride对齐 要求 Atlas 200I/500 A2推理产品 Atlas A2训 练系列产 品/Atlas 800I A2推 理产品	内存大小要 求 (单位 Byte)
该部分格式 仅以下版本 支持： Atlas 200I/500 A2推理产品 Atlas A2训 练系列产 品/Atlas 800I A2推 理产品	YVU420Pla nar YUV420Pla nar	宽2对齐， 高2对齐	-	宽Stride无 对齐要求， 与宽相同即 可；高 Stride无对 齐要求，与 高相同即 可。	宽Stride * 高Stride * 3/2
	YVU422Pla nar YUV422Pla nar	无对齐要求	-	宽Stride无 对齐要求， 与宽相同即 可；高 Stride无对 齐要求，与 高相同即 可。	宽Stride * 高Stride * 2
	YVU444Pla nar YUV444Pla nar	无对齐要求	-	宽Stride无 对齐要求， 与宽相同即 可；高 Stride无对 齐要求，与 高相同即 可。	宽Stride * 高Stride * 3
	YVU444Pac ked 8bit	无对齐要求	-	宽Stride无 对齐要求， 宽Stride为 宽 * 3的 值；高 Stride无对 齐要求，与 高相同即 可。	宽Stride * 高Stride

产品	图片格式	图片宽、高 对齐要求	图片宽 Stride、高 Stride对齐 要求 Atlas 推理 系列产品	图片宽 Stride、高 Stride对齐 要求 Atlas 200I/500 A2推理产品 Atlas A2训 练系列产 品/Atlas 800I A2推 理产品	内存大小要 求 (单位 Byte)
	YUV440Pla nar YVU440Pla nar	宽无对齐要 求， 但宽 ≤ 4096 高2对齐	-	宽Stride无 对齐要求， 与宽相同即 可； 高 Stride无对 齐要求， 与 高相同即 可。	宽Stride * 高Stride * 2
	RGB888Pla nar BGR888Pla nar	无对齐要求	-	宽Stride无 对齐要求， 与宽相同即 可； 高 Stride无对 齐要求， 与 高相同即 可。	宽Stride * 高Stride * 3
该部分格式 仅以下版本 支持： Atlas A2训 练系列产 品/Atlas 800I A2推 理产品	YUV400 FP32	无对齐要求	-	宽Stride无 对齐要求， 宽Stride为 宽 * 4 (4表 示FP32占4 个字节) 的 值； 高 Stride无对 齐要求， 与 高相同即 可。	宽Stride * 高Stride

产品	图片格式	图片宽、高 对齐要求	图片宽 Stride、高 Stride对齐 要求 Atlas 推理 系列产品	图片宽 Stride、高 Stride对齐 要求 Atlas 200I/500 A2推理产品 Atlas A2训 练系列产 品/Atlas 800I A2推 理产品	内存大小要 求 (单位 Byte)
	RGB888 FP32 BGR888 FP32	无对齐要求	-	宽Stride无 对齐要求， 宽Stride为 宽 * 3 * 4 (4表示 FP32占4个 字节) 的 值；高 Stride无对 齐要求，与 高相同即可。	宽Stride * 高Stride
	RGB888Pla nar FP32 BGR888Pla nar FP32	无对齐要求	-	宽Stride无 对齐要求， 宽Stride为 宽 * 4 (4表 示FP32占4 个字节) 的 值；高 Stride无对 齐要求，与 高相同即可。	宽Stride * 高Stride * 3

表 2-44 输出图片格式、宽高对齐、内存大小约束

产品	图片格式	图片宽、高 对齐要求	图片宽 Stride、高 Stride对齐 要求 Atlas 推理 系列产品	图片宽 Stride、高 Stride对齐 要求 Atlas 200I/500 A2推理产品 Atlas A2训 练系列产 品/Atlas 800I A2推 理产品	内存大小要 求 (单位 Byte)
通用格式， 各版本都支持	YUV400 8bit	无对齐要求	宽Stride为 宽16对齐后 的值； 高Stride无 对齐要求， 与高相同即可。	宽Stride无 对齐要求， 与宽相同即可； 高Stride无 对齐要求， 与高相同即可。	宽Stride * 高Stride
	YUV420SP NV12 8bit YUV420SP NV21 8bit	宽2对齐 高2对齐	宽Stride为 宽16对齐后 的值； 高Stride为 高2对齐后 的值。	宽Stride无 对齐要求， 与宽相同即可； 高stride无 对齐要求， 与高相同即可。	宽Stride * 高Stride * 3/2
	YUV422SP 8bit YVU422SP 8bit	宽2对齐 高无对齐要求	宽Stride为 宽16对齐后 的值； 高Stride无 对齐要求， 与高相同即可。	宽Stride无 对齐要求， 与宽相同即可； 高Stride无 对齐要求， 与高相同即可。	宽Stride * 高Stride * 2
	YUV444Pac ked 8bit RGB888 BGR888	无对齐要求	宽Stride为 宽16对齐 后、再乘以 3的值； 高Stride无 对齐要求， 与高相同即可。	宽Stride为 宽 * 3的 值； 高Stride无 对齐要求， 与高相同即可。	宽Stride * 高Stride

产品	图片格式	图片宽、高 对齐要求	图片宽 Stride、高 Stride对齐 要求 Atlas 推理 系列产品	图片宽 Stride、高 Stride对齐 要求 Atlas 200I/500 A2推理产品 Atlas A2训 练系列产 品/Atlas 800I A2推 理产品	内存大小要 求 (单位 Byte)
	ARGB8888 ABGR8888 RGBA8888 BGRA8888	无对齐要求	宽Stride为 宽16对齐 后、再乘以 4的值； 高Stride无 对齐要求， 与高相同即 可。	宽Stride为 宽 * 4的 值； 高Stride无 对齐要求， 与高相同即 可。	宽Stride * 高Stride
该部分格式 仅以下版本 支持： Atlas 200I/500 A2推理产品 Atlas A2训 练系列产 品/Atlas 800I A2推 理产品	YUV422Pac ked YUYV 8bit YUV422Pac ked UYVY 8bit YUV422Pac ked YVYU 8bit YUV422Pac ked VYUY 8bit	宽无对齐要 求 高2对齐	-	宽Stride为 宽 * 2的 值； 高Stride无 对齐要求， 与高相同即 可。	宽Stride * 高Stride
该部分格式 仅以下版本 支持： Atlas 200I/500 A2推理产品 Atlas A2训 练系列产 品/Atlas 800I A2推 理产品	YVU444Pac ked	无对齐要求	-	宽Stride为 宽 * 3的 值； 高Stride无 对齐要求， 与高相同即 可。	宽Stride * 高Stride
	RGB888Pla nar BGR888Pla nar	无对齐要求	-	宽Stride无 对齐要求， 与宽相同即 可；高 Stride无对 齐要求，与 高相同即 可。	宽Stride * 高Stride * 3

产品	图片格式	图片宽、高 对齐要求	图片宽 Stride、高 Stride对齐 要求 Atlas 推理 系列产品	图片宽 Stride、高 Stride对齐 要求 Atlas 200I/500 A2推理产品 Atlas A2训 练系列产 品/Atlas 800I A2推 理产品	内存大小要 求 (单位 Byte)
该部分格式 仅以下版本 支持： Atlas A2训 练系列产 品/Atlas 800I A2推 理产品	YUV400 FP32	无对齐要求	-	宽Stride无 对齐要求， 宽Stride为 宽 * 4 (4表 示FP32占4 个字节) 的 值；高 Stride无对 齐要求，与 高相同即 可。	宽Stride * 高Stride
	RGB888 FP32 BGR888 FP32	无对齐要求	-	宽Stride无 对齐要求， 宽Stride为 宽 * 3 * 4 (4表示 FP32占4个 字节) 的 值；高 Stride无对 齐要求，与 高相同即 可。	宽Stride * 高Stride
	RGB888Pla nar FP32 BGR888Pla nar FP32	无对齐要求	-	宽Stride无 对齐要求， 宽Stride为 宽 * 4 (4表 示FP32占4 个字节) 的 值；高 Stride无对 齐要求，与 高相同即 可。	宽Stride * 高Stride * 3

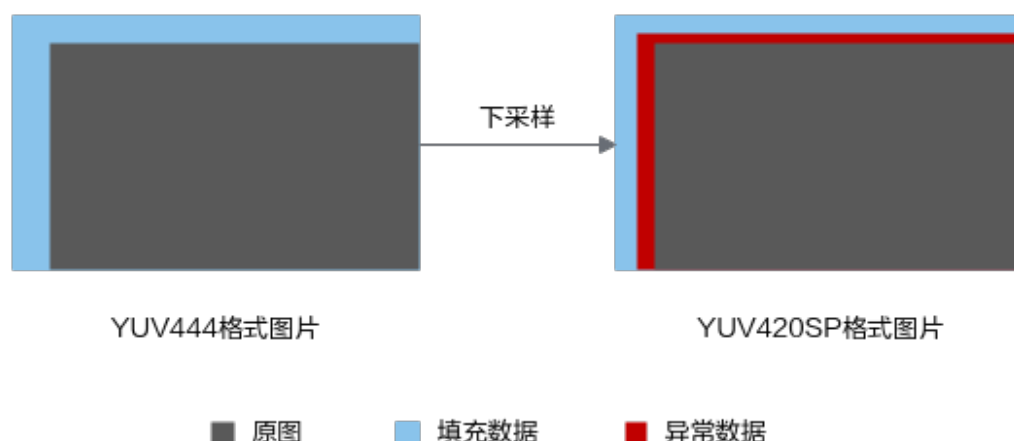
抠图、贴图约束

- 抠图区域不超出输入图片区域。
- 抠图、贴图区域的奇数、偶数限制为：
输出图片格式为YUV420SP，贴图区域奇数、偶数限制为：左偏移和上偏移为偶数、右偏移和下偏移为奇数。
输出图片格式为YUV422SP，贴图区域奇数、偶数限制为：左偏移为偶数、右偏移为奇数。
其它格式，贴图区域没有偏移奇偶数的限制。
- 贴图区域不超出输出图片区域，最大贴图个数256个。
在Atlas 推理系列产品上，由于16对齐的要求，还存在以下约束：
 - 贴图区域相对输出图片的左偏移16对齐。贴图时可直接放置在输出图片的最左侧，即相对输出图片的左偏移为0，0也满足16对齐的要求。
 - 输出图片的贴图宽度建议16对齐，如果不是16对齐，会多写一段无效数据使其16对齐，如图2-13、图2-14所示，贴图区域旁边的绿色框就表示无效数据。
 - 等比例缩放场景下，由于贴图左偏移16对齐的限制，如果直接调用[acl.himpi.vpc_crop_resize_paste](#)接口实现缩放功能，贴图的位置可能不在输出图片的正中心位置，此时可通过抠图、缩放、填充几个功能配合使用（调用[acl.himpi.vpc_crop_resize_make_border](#)或[acl.himpi.vpc_batch_crop_resize_make_border](#)接口），实现等比例缩放。

YUV 格式图像下采样约束

VPC在处理图片时，会根据输入或输出图片格式，将输入图片格式转换为YUV444或RGB用于内部处理，YUV444或RGB没有宽高奇偶数的限制，但当输出图片格式为YUV420SP或YUV422SP格式时，会进行下采样处理，由于YUV420SP或YUV422SP格式本身的数据排布导致宽高存在奇偶数限制，因此输出图片的边缘可能存在异常数据。

图 2-19 异常效果图片举例



出现异常数据的根因在于，在计算过程中，出现输出图片位置处于奇数起始点时，此时YUV444格式的图片是正确的，但是下采样到YUV420SP格式时，由于奇数行和偶数

行是共用同一个UV的，导致图片起始行的Y与上一行的UV组成新的像素，产生异常数据。

精度相关约束（Atlas 推理系列产品）

VPC功能中，贴图区域的宽有16对齐的约束，贴图有效区域的宽不满足16对齐时，贴图区域中会存在一些补边的无效数据，为保证推理精度，贴图有效区域的宽不满足16对齐时，在贴图前用户可先将贴图有效区域缩放成16 * 2对齐，或在推理前借助AIPP抠出有效区域，去除无效数据。

2.10.4.3 性能指标说明

Atlas 200/300/500 推理产品上，该功能为预留功能，暂不支持。

Atlas 训练系列产品上，该功能为预留功能，暂不支持。

性能指标说明

本节仅描述各产品对应的昇腾AI处理器的性能指标参考数据，各产品由于其内存、CPU等配置不同，在性能规格上会存在差异，详细产品的性能规格请查阅[昇腾社区](#)的官方说明。

本节以单个昇腾AI处理器为例，描述基本场景性能指标参考，其它分辨率可以等量估算。

下表中的1路对应一个通道，一个通道对应一个线程；各分辨率的含义为：720p指分辨率为1280 * 720的图片，1080p指分辨率为1920 * 1080的图片，4K指分辨率为3840 * 2160的图片。

表 2-45 Atlas 推理系列产品的昇腾 AI 处理器性能指标参考数据

场景举例	总帧率
● 输入图片分辨率：1080p（1920 * 1080） ● 输出图片分辨率：1080p（1920 * 1080） ● 输入/输出图片格式：YUV420SP ● n路（1 ≤ n ≤ 12）	n * 360fps
● 输入图片分辨率：1080p（1920 * 1080） ● 输出图片分辨率：1080p（1920 * 1080） ● 输入/输出图片格式：YUV420SP ● n路（n > 12）	4320fps

场景举例	总帧率
• 输入图片分辨率：4K图像 (3840 * 2160) • 输出图片分辨率：4K图像 (3840 * 2160) • 输入/输出图片格式：YUV420SP • n路 (1 ≤ n ≤ 12)	n * 90fps
• 输入图片分辨率：4K图像 (3840 * 2160) • 输出图片分辨率：4K图像 (3840 * 2160) • 输入/输出图片格式：YUV420SP • n路 (n > 12)	1080fps
• 输入图片分辨率：8K图像 (7680 * 4320) • 输出图片分辨率：4K图像 (3840 * 2160) • 输入/输出图片格式：YUV420SP • n路 (1 ≤ n ≤ 12)	n * 22.5fps
• 输入图片分辨率：8K图像 (7680 * 4320) • 输出图片分辨率：4K图像 (3840 * 2160) • 输入/输出图片格式：YUV420SP • n路 (n > 12)	270fps

表 2-46 Atlas 200I/500 A2 推理产品的昇腾 AI 处理器性能指标参考数据

场景举例	总帧率
• 输入图片分辨率：1080p (1920 * 1080) • 输出图片分辨率：1080p (1920 * 1080) • 输入/输出图片格式：YUV420SP • n路 (n < 2)	n * 800fps

场景举例	总帧率
• 输入图片分辨率：1080p (1920 * 1080) • 输出图片分辨率：1080p (1920 * 1080) • 输入/输出图片格式： YUV420SP • n路 ($n \geq 2$)	1600fps
• 输入图片分辨率：4K图像 (3840 * 2160) • 输出图片分辨率：4K图像 (3840 * 2160) • 输入/输出图片格式： YUV420SP • n路 ($n < 2$)	$n * 200\text{fps}$
• 输入图片分辨率：4K图像 (3840 * 2160) • 输出图片分辨率：4K图像 (3840 * 2160) • 输入/输出图片格式： YUV420SP • n路 ($n \geq 2$)	400fps
• 输入图片分辨率：8K图像 (7680 * 4320) • 输出图片分辨率：4K图像 (3840 * 2160) • 输入/输出图片格式： YUV420SP • n路 ($n < 2$)	$n * 100\text{fps}$
• 输入图片分辨率：8K图像 (7680 * 4320) • 输出图片分辨率：4K图像 (3840 * 2160) • 输入/输出图片格式： YUV420SP • n路 ($n \geq 2$)	200fps

表 2-47 Atlas A2 训练系列产品/Atlas 800I A2 推理产品的昇腾 AI 处理器性能指标参考数据

场景举例	总帧率
- 输入图片分辨率：1080p (1920 * 1080) - 输出图片分辨率：1080p (1920 * 1080) - 输入/输出图片格式：YUV420SP - n路 ($1 \leq n \leq 10$)	$n * 1440\text{fps}$
- 输入图片分辨率：1080p (1920 * 1080) - 输出图片分辨率：1080p (1920 * 1080) - 输入/输出图片格式：YUV420SP - n路 ($n > 10$)	14400fps
- 输入图片分辨率：4K图像 (3840 * 2160) - 输出图片分辨率：4K图像 (3840 * 2160) - 输入/输出图片格式：YUV420SP - n路 ($1 \leq n \leq 10$)	$n * 360\text{fps}$
- 输入图片分辨率：4K图像 (3840 * 2160) - 输出图片分辨率：4K图像 (3840 * 2160) - 输入/输出图片格式：YUV420SP - n路 ($n > 10$)	3600fps
- 输入图片分辨率：8K图像 (7680 * 4320) - 输出图片分辨率：4K图像 (3840 * 2160) - 输入/输出图片格式：YUV420SP - n路 ($1 \leq n \leq 10$)	$n * 90\text{fps}$

场景举例	总帧率
● 输入图片分辨率：8K图像（7680 * 4320） ● 输出图片分辨率：4K图像（3840 * 2160） ● 输入/输出图片格式：YUV420SP ● n路（n > 10）	900fps

 说明

- VPC处理性能与处理过程中的图像分辨率强相关，以输入图像和输出图像中的最大分辨率作为基准分辨率，基准分辨率越大，处理耗时越久，性能越低。
- 调用VPC批处理接口（接口命名中包含batch，例如acl.himpi.vpc_batch_crop_resize_paste接口）时，由于图像处理单元DVPP（Digital Video Pre-Processing）内部多个VPC硬件单元会并行处理图片任务，因此单路就可以达到最大总帧率。

2.10.4.4 acl.himpi.vpc_create_chn

c函数原型	hi_s32 hi_mpi_vpc_create_chn(hi_vpc_chn chn, const hi_vpc_chn_attr *attr);
Python函数	ret = acl.himpi.vpc_create_chn(chn, attr)
函数功能	根据设置的通道属性创建图像处理通道，由用户指定通道号。 Atlas 200/300/500 推理产品 ，当前版本该接口预留。 Atlas 训练系列产品 ，当前版本该接口预留。
输入说明	chn: int，图片处理通道号。 ● Atlas 推理系列产品，该参数的取值范围：[0, 256)，通道总数最多256。 ● Atlas A2训练系列产品/Atlas 800I A2推理产品，该参数的取值范围：[0, 256)，通道总数最多256。 ● Atlas 200I/500 A2推理产品，该参数的取值范围：[0, 128)，通道总数最多128。 说明 对于Atlas 推理系列产品，昇腾虚拟化实例（Ascend Virtual Instance）场景下，通道总数 = （被分配的VPC硬件单元 / VPC硬件单元总数）* 256，如果通道总数不为整数，则向下取整。 对于Atlas A2训练系列产品/Atlas 800I A2推理产品，昇腾虚拟化实例（Ascend Virtual Instance）场景下，通道总数最多256。 对于Atlas 200I/500 A2推理产品，昇腾虚拟化实例（Ascend Virtual Instance）场景下，通道总数最多128。 昇腾虚拟化实例（Ascend Virtual Instance）的详细描述，以及各场景下的算力资源规格请参见《昇腾虚拟化实例（AVI）用户指南》。 attr: dict，图片处理通道属性字典，具体请参见2.10.8.5.13 hi_vpc_chn_attr。

返回值说明	ret : int, 错误码。 - 返回0, 表示成功。 - 返回非0, 表示失败, 参见2.10.9.2 VPC图像处理返回码。
约束说明	- Atlas 推理系列产品上, 单个Device上的通道号不能超出最大的通道号范围。用户在使用VPC功能或VDEC功能(内部会使用VPC功能)时, 都会占用VPC通道号, 为防止VPC功能与VDEC功能的通道号冲突, 用户在使用VPC功能时, 通道号的取值范围为[0, 255]。在使用VDEC功能时, VDEC内部使用的VPC通道号范围为[256, 511]。 - 如果参数“attr”为空, 会返回错误码“HI_ERR_VPC_NULL_PTR”。 - 在创建图像处理通道之前必须保证通道未创建(或者已经销毁), 否则会直接返回失败。
参考资源	接口调用流程, 参见典型功能接口调用流程(以缩放为例)。

2.10.4.5 acl.himpi.vpc_destroy_chn

c函数原型	hi_s32 hi_mpi_vpc_destroy_chn(hi_vpc_chn chn);
Python函数	ret = acl.himpi.vpc_destroy_chn(chn)
函数功能	销毁创建的图像处理通道。 Atlas 200/300/500 推理产品 , 当前版本不支持该接口。 Atlas 训练系列产品 , 当前版本不支持该接口。
输入说明	chn : int, 图片处理通道号。 Atlas 推理系列产品, 该参数的取值范围: [0, 256), 通道总数最多256。 Atlas A2训练系列产品/Atlas 800I A2推理产品, 该参数的取值范围: [0, 256), 通道总数最多256。 Atlas 200I/500 A2推理产品, 该参数的取值范围: [0, 128), 通道总数最多128。
返回值说明	ret : int, 错误码。 - 返回0, 表示成功。 - 返回非0, 表示失败, 参见2.10.9.2 VPC图像处理返回码。
约束说明	通道号不能超出最大的通道号范围。
参考资源	接口调用流程, 参见VPC图片处理典型功能。

2.10.4.6 acl.himpi.vpc_resize

c函数原型	hi_s32 hi_mpi_vpc_resize (hi_vpc_chn chn, const hi_vpc_pic_info *source_pic, hi_vpc_pic_info *dest_pic, hi_double fx, hi_double fy, hi_u32 interpolation, hi_u32 *task_id, hi_s32 milli_sec);
Python函数	task_id, ret = acl.himpi.vpc_resize(chn, source_pic, dest_pic, fx, fy, interpolation, milli_sec)
函数功能	对图像进行按照一定比例进行缩放或者缩放到固定尺寸。异步接口。 Atlas 200/300/500 推理产品 ，当前版本不支持该接口。 Atlas 训练系列产品 ，当前版本不支持该接口。

输入说明	chn: int, 图片处理通道号。 ● Atlas 推理系列产品, 该参数的取值范围: [0, 256), 通道总数最多256。 ● Atlas A2训练系列产品/Atlas 800I A2推理产品, 该参数的取值范围: [0, 256), 通道总数最多256。 ● Atlas 200I/500 A2推理产品, 该参数的取值范围: [0, 128), 通道总数最多128。 source_pic: dict, 原始图片信息, 相关约束请参见约束说明。 dest_pic: dict, 目标图片信息。 fx: 宽的缩放比例。 fy: 高的缩放比例。 interpolation: int, 缩放算法。此处配置的缩放算法建议与训练模型时的缩放算法保持一致。 Atlas 推理系列产品, 支持如下缩放算法: ● 0: 业界通用的Bilinear算法 (与OpenCV算法的计算过程类似, 当输入和输出图片格式都为RGB时, 在[1/32, 512]的缩放范围内, 与OpenCV的单个像素值最大差异为正负1)。 ● 1: 业界通用的Nearest neighbor 算法 (与OpenCV算法的计算过程类似)。 Atlas A2训练系列产品/Atlas 800I A2推理产品, 支持如下缩放算法: ● 0: 业界通用的Bilinear算法 (与OpenCV-3.4.2版本算法的计算结果相同)。 ● 1: 业界通用的Nearest neighbor 算法 (与OpenCV-3.4.2版本算法的计算结果相同)。 Atlas 200I/500 A2推理产品, 支持如下缩放算法: ● 0: 业界通用的Bilinear算法 (与OpenCV-3.4.2版本算法的计算结果相同)。 ● 1: 业界通用的Nearest neighbor 算法 (与OpenCV-3.4.2版本算法的计算结果相同)。 milli_sec: 超时时间配置, 单位是毫秒, 取值范围如下。 ● “-1”: 阻塞方式。 ● “0”: 非阻塞方式。 ● >0: 超时方式, 配置具体的超时时间。超时时间受操作系统影响, 一般偏差在操作系统的一个时间片内, 例如, 操作系统的 一个时间片为4ms, 用户设置的milli_sec参数值为1, 则实际的超时时间在1ms到5ms范围内。在CPU负载高场景下, 超时时间仍可能存在波动。
返回值说明	ret: int, 错误码。 ● 返回0表示成功。 ● 返回非0表示失败, 参见2.10.9.2 VPC图像处理返回码。 task_id: int, 此次任务分配的ID, 用来区分任务。

参考资源

接口调用流程，参见VPC图片处理典型功能。

约束说明

版本	约束
Atlas 推理系列产品	- 关于输入图片分辨率、图片格式： - 输入图片分辨率在10 * 6 ~ 4096 * 4096（包括4096）范围内时，支持图片格式、宽高对齐、内存约束处说明的输入图片格式。 - 输入图片的宽或高在4096 ~ 8192（包括8192）范围内时，支持格式如下： 支持hi_pixel_format枚举值中的如下枚举项：<pre>HI_PIXEL_FORMAT_YUV_400 = 0, // YUV400 8bit HI_PIXEL_FORMAT_YUV_SEMIPLANAR_420 = 1, // YUV420SP NV12 8bit HI_PIXEL_FORMAT_YVU_SEMIPLANAR_420 = 2, // YUV420SP NV21 8bit HI_PIXEL_FORMAT_YUV_SEMIPLANAR_422 = 3, // YUV422SP 8bit HI_PIXEL_FORMAT_YVU_SEMIPLANAR_422 = 4, // YVU422SP 8bit HI_PIXEL_FORMAT_YUV_SEMIPLANAR_444 = 5, // YUV444SP 8bit HI_PIXEL_FORMAT_YVU_SEMIPLANAR_444 = 6, // YVU444SP 8bit HI_PIXEL_FORMAT_YUYV_PACKED_422 = 7, // YUV422Packed YUYV 8bit HI_PIXEL_FORMAT_UYVY_PACKED_422 = 8, // YUV422Packed UYVY 8bit HI_PIXEL_FORMAT_YVYU_PACKED_422 = 9, // YUV422Packed YVYU 8bit HI_PIXEL_FORMAT_VYUY_PACKED_422 = 10, // YUV422Packed VYUY 8bit</pre> - 关于各图片格式的宽高对齐、内存大小约束，请参见图片格式、宽高对齐、内存约束。 - 关于输出图片分辨率、图片格式、宽高对齐、内存大小约束请参见2.10.4.2 约束说明。 - 当输入、输出图片格式都为ARGB8888、ABGR8888、RGBA8888、BGRA8888格式时，在经过本接口缩放处理后，A分量值会变为0。 - 设置fx / fy之后，缩放后的图片的分辨率必须在[10 * 6 ~ 4096 * 4096]范围内，否则返回错误。 - 缩放可以通过fx / fy或者目标图片宽高两种方式来指定，但是两者均配置以目标尺寸缩放优先。两者不可同时为0。

版本	约束
Atlas 200I/500 A2推理产品 Atlas A2训练系列产品/ Atlas 800I A2推理产品	- 使用本接口对输入、输出图片分辨率的要求如下： - 输入图片分辨率：10 * 6 ~ 16384 * 16384 当输出图片宽度大于4096时，输入图片最小分辨率为128 * 16。 当输入图片宽度大于8192，输入图片高度最小为16。 - 输出图片分辨率：10 * 6 ~ 16384 * 16384 - 关于图片格式、宽高对齐、内存等约束，请参见图片格式、宽高对齐、内存约束。 - 当输入、输出图片格式都为ARGB8888、ABGR8888、RGBA8888、BGRA8888格式时，在经过本接口缩放处理后，A分量值会变为0。 - 设置fx / fy之后，缩放后的图片的分辨率必须在[10 * 6 ~ 16384 * 16384]范围内，否则返回错误。 - 缩放可以通过fx / fy或者目标图片宽高两种方式来指定，但是两者均配置以目标尺寸缩放优先。两者不可同时为0。

2.10.4.7 acl.himpi.vpc_crop

c函数原型	hi_s32 hi_mpi_vpc_crop(hi_vpc_chn chn, const hi_vpc_pic_info *source_pic, hi_vpc_crop_region_info crop_info[], hi_u32 count, hi_u32 *task_id, hi_s32 milli_sec);
Python函数	task_id, ret = acl.himpi.vpc_crop(chn, source_pic, crop_info, count, milli_sec)
函数功能	按照指定区域从一张输入图片中抠出一张或多张子图。异步接口。 Atlas 200/300/500 推理产品 ，当前版本不支持该接口。 Atlas 训练系列产品 ，当前版本不支持该接口。

输入说明	chn: int, 图片处理通道号。 • Atlas 推理系列产品, 该参数的取值范围: [0, 256), 通道总数最多256。 • Atlas A2训练系列产品/Atlas 800I A2推理产品, 该参数的取值范围: [0, 256), 通道总数最多256。 • Atlas 200I/500 A2推理产品, 该参数的取值范围: [0, 128), 通道总数最多128。 source_pic: dict, 原始图片信息, 具体请参见2.10.8.5.1 hi_vpc_pic_info, 相关约束请参见约束说明。 crop_info: list, 抠图图片信息列表, 长度与“count”参数值保持一致, 具体请参见2.10.8.5.4 hi_vpc_crop_region_info。 count: int, 抠图区域的数量, 取值范围[1, 256]。 milli_sec: int, 超时时间配置, 单位是毫秒, 取值范围如下。 • “-1”: 阻塞方式。 • “0”: 非阻塞方式。 • >0: 超时方式, 配置具体的超时时间。超时时间受操作系统影响, 一般偏差在操作系统的—个时间片内, 例如, 操作系统的—个时间片为4ms, 用户设置的milli_sec参数值为1, 则实际的超时时间在1ms到5ms范围内。在CPU负载高场景下, 超时时间仍可能存在波动。
返回值说明	task_id: int, 此次任务分配的ID, 用来区分任务。 ret: int, 错误码。 • 返回0表示成功。 • 返回非0表示失败, 参见2.10.9.2 VPC图像处理返回码。
参考资源	接口调用流程, 参见典型功能接口调用流程(以缩放为例)。

约束说明

版本	约束
Atlas 推理系列产品	- 关于输入图片分辨率、图片格式： - 输入图片分辨率在$10 * 6 \sim 4096 * 4096$（包括4096）范围内时，支持图片格式、宽高对齐、内存约束处说明的输入图片格式。 - 输入图片的宽或高在$4096 \sim 8192$（包括8192）范围内时，支持格式如下： 支持hi_pixel_format枚举值中的如下枚举项： <pre>HI_PIXEL_FORMAT_YUV_400 = 0, // YUV400 8bit HI_PIXEL_FORMAT_YUV_SEMIPLANAR_420 = 1, // YUV420SP NV12 8bit HI_PIXEL_FORMAT_YVU_SEMIPLANAR_420 = 2, // YUV420SP NV21 8bit HI_PIXEL_FORMAT_YUV_SEMIPLANAR_422 = 3, // YUV422SP 8bit HI_PIXEL_FORMAT_YVU_SEMIPLANAR_422 = 4, // YVU422SP 8bit HI_PIXEL_FORMAT_YUV_SEMIPLANAR_444 = 5, // YUV444SP 8bit HI_PIXEL_FORMAT_YVU_SEMIPLANAR_444 = 6, // YVU444SP 8bit HI_PIXEL_FORMAT_YUYV_PACKED_422 = 7, // YUV422Packed YUYV 8bit HI_PIXEL_FORMAT_UYVY_PACKED_422 = 8, // YUV422Packed UYVY 8bit HI_PIXEL_FORMAT_VYVU_PACKED_422 = 9, // YUV422Packed VYVU 8bit HI_PIXEL_FORMAT_VYUY_PACKED_422 = 10, // YUV422Packed VYUY 8bit</pre> - 关于各图片格式的宽高对齐、内存大小约束，请参见图片格式、宽高对齐、内存约束。 - 关于输出图片分辨率、图片格式、宽高对齐、内存大小约束请参见2.10.4.2 约束说明。 - 单张1080P的图像处理超时时间建议设置3ms，以此类推，如果是2张1080P的图像，建议设置超时间为6ms；如果是3张1080P图片，建议设置超时时间为9ms。其他分辨率，按照图像尺寸进行近似的等价折算，例如，$3840 * 2160$分辨率的图像建议超时时间设置为12ms。
Atlas 200I/500 A2推理产品 Atlas A2训练系列产品/ Atlas 800I A2推理产品	- 关于图片分辨率、图片格式、宽高对齐、内存等约束，请参见2.10.4.2 约束说明。 - 单张1080P的图像处理超时时间建议设置3ms，以此类推，如果是2张1080P的图像，建议设置超时间为6ms；如果是3张1080P图片，建议设置超时时间为9ms。其他分辨率，按照图像尺寸进行近似的等价折算，例如，$3840 * 2160$分辨率的图像建议超时时间设置为12ms。

2.10.4.8 acl.himpi.vpc_crop_resize

c函数原型	hi_s32 hi_mpi_vpc_crop_resize(hi_vpc_chn chn, const hi_vpc_pic_info *source_pic, hi_vpc_crop_resize_region crop_resize_info[], hi_u32 count, hi_u32 *task_id, hi_s32 milli_sec);
Python函数	task_id, ret = acl.himpi.vpc_crop_resize(chn, source_pic, crop_resize_info, count, milli_sec)
函数功能	从一张大图扣出一张或多张子图，并缩放到指定尺寸。异步接口。 Atlas 200/300/500 推理产品，当前版本不支持该接口。 Atlas 训练系列产品，当前版本不支持该接口。
输入说明	chn: int，图片处理通道号。 • Atlas 推理系列产品，该参数的取值范围：[0, 256)，通道总数最多256。 • Atlas A2训练系列产品/Atlas 800I A2推理产品，该参数的取值范围：[0, 256)，通道总数最多256。 • Atlas 200I/500 A2推理产品，该参数的取值范围：[0, 128)，通道总数最多128。 source_pic: dict，原始图片信息，具体请参见2.10.8.5.1 hi_vpc_pic_info，相关约束请参见约束说明。 crop_resize_info: list，抠图并缩放的图片信息列表，长度与“count”参数值保持一致，具体请参见2.10.8.5.5 hi_vpc_crop_resize_region。 count: int，抠图并缩放的图片数量，取值范围[1, 256]。 milli_sec: int，超时时间配置，单位是毫秒，取值范围如下。 • “-1”：阻塞方式。 • “0”：非阻塞方式。 • >0：超时方式，配置具体的超时时间。超时时间受操作系统影响，一般偏差在操作系统的一个时间片内，例如，操作系统的—个时间片为4ms，用户设置的milli_sec参数值为1，则实际的超时时间在1ms到5ms范围内。在CPU负载高场景下，超时时间仍可能存在波动。
返回值说明	ret: int，错误码。 • 返回0表示成功。 • 返回非0表示失败，参见2.10.9.2 VPC图像处理返回码 task_id: int，任务分配的ID，用来区分任务。

约束说明

版本	约束
Atlas 推理系列产品	- 关于输入图片分辨率、图片格式： - 输入图片分辨率在10 * 6 ~ 4096 * 4096（包括4096）范围内时，支持图片格式、宽高对齐、内存约束处说明的输入图片格式。 - 输入图片的宽或高在4096 ~ 8192（包括8192）范围内时，支持格式如下： 支持hi_pixel_format枚举值中的如下枚举项： HI_PIXEL_FORMAT_YUV_400 = 0, // YUV400 8bit HI_PIXEL_FORMAT_YUV_SEMIPLANAR_420 = 1, // YUV420SP NV12 8bit HI_PIXEL_FORMAT_YVU_SEMIPLANAR_420 = 2, // YUV420SP NV21 8bit HI_PIXEL_FORMAT_YUV_SEMIPLANAR_422 = 3, // YUV422SP 8bit HI_PIXEL_FORMAT_YVU_SEMIPLANAR_422 = 4, // YVU422SP 8bit HI_PIXEL_FORMAT_YUV_SEMIPLANAR_444 = 5, // YUV444SP 8bit HI_PIXEL_FORMAT_YVU_SEMIPLANAR_444 = 6, // YVU444SP 8bit HI_PIXEL_FORMAT_YUYV_PACKED_422 = 7, // YUV422Packed YUYV 8bit HI_PIXEL_FORMAT_UYVY_PACKED_422 = 8, // YUV422Packed UYVY 8bit HI_PIXEL_FORMAT_YVYU_PACKED_422 = 9, // YUV422Packed YVYU 8bit HI_PIXEL_FORMAT_VYUY_PACKED_422 = 10, // YUV422Packed VYUY 8bit - 关于各图片格式的宽高对齐、内存大小约束，请参见图片格式、宽高对齐、内存约束。 - 关于输出图片分辨率、图片格式、宽高对齐、内存大小约束请参见2.10.4.2 约束说明。 - 当输入、输出图片格式都为ARGB8888、ABGR8888、RGBA8888、BGRA8888格式时，在经过本接口缩放处理后，A分量值会变为0。 - 若是crop的宽高与resize之后的宽高一致，则不进行缩放；resize宽高必须与输出宽高一致。
Atlas 200I/500 A2推理产品 Atlas A2训练系列产品/ Atlas 800I A2推理产品	- 关于图片分辨率、图片格式、宽高对齐、内存等约束，请参见2.10.4.2 约束说明。 - 当输入、输出图片格式都为ARGB8888、ABGR8888、RGBA8888、BGRA8888格式时，在经过本接口缩放处理后，A分量值会变为0。 - 若是crop的宽高与resize之后的宽高一致，则不进行缩放；resize宽高必须与输出宽高一致。

2.10.4.9 acl.himpi.vpc_crop_resize_paste

c函数原型	hi_s32 hi_mpi_vpc_crop_resize_paste (hi_vpc_chn chn, const hi_vpc_pic_info *source_pic, hi_vpc_crop_resize_paste_region crop_resize_paste_info[], hi_u32 count, hi_u32 *task_id, hi_s32 milli_sec);
Python函数	task_id, ret = acl.himpi.vpc_crop_resize_paste(chn, source_pic, crop_resize_paste_info, count, milli_sec)
函数功能	从输入图片中抠图，对抠出的子图进行缩放后，放在用户输出图片的指定区域。异步接口。 Atlas 200/300/500 推理产品，当前版本不支持该接口。 Atlas 训练系列产品，当前版本不支持该接口。
输入说明	chn: int，图片处理通道号。 • Atlas 推理系列产品，该参数的取值范围：[0, 256)，通道总数最多256。 • Atlas A2训练系列产品/Atlas 800I A2推理产品，该参数的取值范围：[0, 256)，通道总数最多256。 • Atlas 200I/500 A2推理产品，该参数的取值范围：[0, 128)，通道总数最多128。 source_pic: dict，原始图片信息，具体请参见2.10.8.5.1 hi_vpc_pic_info，相关约束请参见约束说明。 crop_resize_paste_info: list，抠图缩放并贴图的图片信息列表，长度与“count”参数值保持一致，具体请参见2.10.8.5.6 hi_vpc_crop_resize_paste_region。 count: int，抠图缩放贴图的图片数量，取值范围[1, 256]。 milli_sec: int，超时时间配置，单位是毫秒，取值范围如下。 • “-1”：阻塞方式。 • “0”：非阻塞方式。 • >0：超时方式，配置具体的超时时间。超时时间受操作系统影响，一般偏差在操作系统的一个时间片内，例如，操作系统的—个时间片为4ms，用户设置的milli_sec参数值为1，则实际的超时时间在1ms到5ms范围内。在CPU负载高场景下，超时时间仍可能存在波动。
返回值说明	ret: int，错误码。 • 返回0表示成功。 • 返回非0表示失败，参见2.10.9.2 VPC图像处理返回码。 task_id: int，任务分配的ID，用来区分任务。

约束说明

版本	约束
Atlas 推理系列产品	- 关于输入图片分辨率、图片格式： - 输入图片分辨率在10 * 6 ~ 4096 * 4096（包括4096）范围内时，支持图片格式、宽高对齐、内存约束处说明的输入图片格式。 - 输入图片的宽或高在4096 ~ 8192（包括8192）范围内时，支持格式如下： 支持hi_pixel_format枚举值中的如下枚举项： HI_PIXEL_FORMAT_YUV_400 = 0, // YUV400 8bit HI_PIXEL_FORMAT_YUV_SEMIPLANAR_420 = 1, // YUV420SP NV12 8bit HI_PIXEL_FORMAT_YVU_SEMIPLANAR_420 = 2, // YUV420SP NV21 8bit HI_PIXEL_FORMAT_YUV_SEMIPLANAR_422 = 3, // YUV422SP 8bit HI_PIXEL_FORMAT_YVU_SEMIPLANAR_422 = 4, // YVU422SP 8bit HI_PIXEL_FORMAT_YUV_SEMIPLANAR_444 = 5, // YUV444SP 8bit HI_PIXEL_FORMAT_YVU_SEMIPLANAR_444 = 6, // YVU444SP 8bit HI_PIXEL_FORMAT_YUYV_PACKED_422 = 7, // YUV422Packed YUYV 8bit HI_PIXEL_FORMAT_UYVY_PACKED_422 = 8, // YUV422Packed UYVY 8bit HI_PIXEL_FORMAT_YVYU_PACKED_422 = 9, // YUV422Packed YVYU 8bit HI_PIXEL_FORMAT_VYUY_PACKED_422 = 10, // YUV422Packed VYUY 8bit - 关于各图片格式的宽高对齐、内存大小约束，请参见图片格式、宽高对齐、内存约束。 - 关于输出图片分辨率、图片格式、宽高对齐、内存大小约束请参见2.10.4.2 约束说明。 - 当输入、输出图片格式都为ARGB8888、ABGR8888、RGBA8888、BGRA8888格式时，在经过本接口缩放处理后，A分量值会变为0。 - 输出图片的数量最多等于抠图区域的数量，最少是一张图片。也就是说，如果从输入图片中扣n张子图，那么贴图的目标图片可以是同一张大图，也可以是多张大图。取决于hi_vpc_crop_resize_paste_region字典中指定的地址。 - 由于“hi_vpc_crop_resize_paste_region”字典内容比较丰富，使用者可以自由组合功能。例如：若是crop的宽高与resize之后的宽高一致，则不进行缩放。

版本	约束
Atlas 200I/500 A2推理产品 Atlas A2训练系列产品/ Atlas 800I A2推理产品	- 关于图片分辨率、图片格式、宽高对齐、内存等约束，请参见2.10.4.2 约束说明。 - 由于YUV格式图像下采样约束，当输出图片格式为YUV420SP或YUV422SP格式时，贴图的顶部偏移、左侧偏移需为偶数。 - 当输入、输出图片格式都为ARGB8888、ABGR8888、RGBA8888、BGRA8888格式时，在经过本接口缩放处理后，A分量值会变为0。 - 输出图片的数量最多等于抠图区域的数量，最少是一张图片。也就是说，如果从输入图片中扣n张子图，那么贴图的目标图片可以是同一张大图，也可以是多张大图。取决于“hi_vpc_crop_resize_paste_region”字典中指定的地址。 - 由于“hi_vpc_crop_resize_paste_region”字典内容比较丰富，使用者可以自由组合功能。例如：若是crop的宽高与resize之后的宽高一致，则不进行缩放。

2.10.4.10 acl.himpi.vpc_convert_color

c函数原型	hi_s32 hi_mpi_vpc_convert_color (hi_vpc_chn chn, const hi_vpc_pic_info *source_pic, hi_vpc_pic_info *dest_pic, hi_u32 *task_id, hi_s32 milli_sec);
Python函数	task_id, ret = acl.himpi.vpc_convert_color(chn, source_pic, dest_pic, milli_sec)
函数功能	转换图片的格式。异步接口。 Atlas 200/300/500 推理产品 ，当前版本不支持该接口。 Atlas 训练系列产品 ，当前版本不支持该接口。

输入说明	chn: int, 图片处理通道号。 ● Atlas 推理系列产品, 该参数的取值范围: [0, 256), 通道总数最多256。 ● Atlas A2训练系列产品/Atlas 800I A2推理产品, 该参数的取值范围: [0, 256), 通道总数最多256。 ● Atlas 200I/500 A2推理产品, 该参数的取值范围: [0, 128), 通道总数最多128。 source_pic: dict, 原始图片信息, 具体请参见2.10.8.5.1 hi_vpc_pic_info, 相关约束请参见约束说明。 dest_pic: dict, 目标图片信息, 具体请参见2.10.8.5.1 hi_vpc_pic_info。 需指定输出图片的宽、高、宽Stride、高Stride、内存地址、内存大小、格式等。 ● 配置输出图片的宽、高时, 需与输入图片的宽、高保持一致, 否则会返回报错。 如果将输出图片的宽或高配置为0, 则VPC内部会将输入图片的宽、高分别作为输出图片的宽、高; 同时, VPC内部会按对齐要求计算宽Stride、高Stride, 不同图片格式的对齐要求不同, 请参见图片格式、宽高对齐、内存约束。 ● 配置输出图片的宽Stride、高Stride时, 满足对齐要求即可, 不同图片格式的对齐要求不同, 请参见图片格式、宽高对齐、内存约束。 milli_sec: int, 超时时间配置, 单位是毫秒, 取值范围如下。 ● “-1”: 阻塞方式。 ● “0”: 非阻塞方式。 ● >0: 超时方式, 配置具体的超时时间。超时时间受操作系统影响, 一般偏差在操作系统的—个时间片内, 例如, 操作系统的—个时间片为4ms, 用户设置的milli_sec参数值为1, 则实际的超时时间在1ms到5ms范围内。在CPU负载高场景下, 超时时间仍可能存在波动。
返回值说明	ret: int, 错误码。 ● 返回0表示成功。 ● 返回非0表示失败, 参见2.10.9.2 VPC图像处理返回码。 task_id: 此次任务分配的ID, 用来区分任务。

约束说明

版本	约束
Atlas 推理系列产品	● 输入、输出图片分辨率: 10 * 6 ~ 4096 * 4096 (包括 4096 * 4096) 。 ● 输入图片格式、输出图片格式、宽高对齐约束、内存约束请参见图片格式、宽高对齐、内存约束。当输入格式是YUV400时, 输出格式只能是YUV400。

版本	约束
Atlas 200I/500 A2推理产品	- 输入、输出图片分辨率为10 * 6 ~ 4096 * 8192 (包括 4096 * 8192) 。 - 输入图片格式、输出图片格式、宽高对齐约束、内存约束请参见图片格式、宽高对齐、内存约束。
Atlas A2训练系列产品/ Atlas 800I A2推理产品	

2.10.4.11 acl.himpi.vpc_convert_color_v2

c函数原型	hi_s32 hi_mpi_vpc_convert_color_v2(hi_vpc_chn chn, const hi_vpc_pic_info *source_pic, hi_vpc_pic_info *dest_pic, hi_csc_conf *conf, hi_u32 *task_id, hi_s32 milli_sec)
Python函数	task_id, ret = acl.himpi.vpc_convert_color_v2(chn, source_pic, dest_pic, conf, milli_sec)
函数功能	转换图片的格式，在2.10.4.10 acl.himpi.vpc_convert_color接口的基础上扩展功能，支持设置RGBA格式图片的透明度。异步接口。 Atlas 200/300/500 推理产品，当前版本不支持该接口。 Atlas 训练系列产品，当前版本不支持该接口。

输入说明	chn: int, 图片处理通道号。 ● Atlas 推理系列产品, 该参数的取值范围: [0, 256), 通道总数最多256。 ● Atlas A2训练系列产品/Atlas 800I A2推理产品, 该参数的取值范围: [0, 256), 通道总数最多256。 ● Atlas 200I/500 A2推理产品, 该参数的取值范围: [0, 128), 通道总数最多128。 source_pic: dict, 原始图片信息, 具体请参见2.10.8.5.1 hi_vpc_pic_info, 相关约束请参见约束说明。 dest_pic: dict, 目标图片信息, 具体请参见2.10.8.5.1 hi_vpc_pic_info。 需指定输出图片的宽、高、宽Stride、高Stride、内存地址、内存大小、格式等。 ● 配置输出图片的宽、高时, 需与输入图片的宽、高保持一致, 否则会返回报错。 如果将输出图片的宽或高配置为0, 则VPC内部会将输入图片的宽、高分别作为输出图片的宽、高; 同时, VPC内部会按对齐要求计算宽Stride、高Stride, 不同图片格式的对齐要求不同, 请参见图片格式、宽高对齐、内存约束。 ● 配置输出图片的宽Stride、高Stride时, 满足对齐要求即可, 不同图片格式的对齐要求不同, 请参见图片格式、宽高对齐、内存约束。 conf: int, 输出图片透明度值。 透明度只针对以下图像格式有效: <pre>PIXEL_FORMAT_ARGB_8888 = 14, PIXEL_FORMAT_ABGR_8888 = 15, PIXEL_FORMAT_RGBA_8888 = 16, PIXEL_FORMAT_BGRA_8888 = 17,</pre> milli_sec: int, 超时时间配置, 单位是毫秒, 取值范围如下。 ● “-1”: 阻塞方式。 ● “0”: 非阻塞方式。 ● >0: 超时方式, 配置具体的超时时间。超时时间受操作系统影响, 一般偏差在操作系统的-一个时间片内, 例如, 操作系统的-一个时间片为4ms, 用户设置的milli_sec参数值为1, 则实际的-超时时间在1ms到5ms范围内。在CPU负载高场景下, 超时时间-仍可能存在波动。
返回值说明	ret: int, 错误码。 ● 返回0表示成功。 ● 返回非0表示失败, 参见2.10.9.2 VPC图像处理返回码。 task_id: int, 任务分配的ID, 用来区分任务。

约束说明

版本	约束
Atlas 推理系列产品	- 输入、输出图片分辨率：10 * 6 ~ 4096 * 4096（包括 4096 * 4096）。 - 输入图片格式、输出图片格式、宽高对齐约束、内存约束请参见图片格式、宽高对齐、内存约束。当输入格式是YUV400时，输出格式只能是YUV400。
Atlas 200I/500 A2推理产品 Atlas A2训练系列产品/ Atlas 800I A2推理产品	- 输入、输出图片分辨率为10 * 6 ~ 4096 * 8192（包括 4096 * 8192）。 - 输入图片格式、输出图片格式、宽高对齐约束、内存约束请参见图片格式、宽高对齐、内存约束。

2.10.4.12 acl.himpi.vpc_convert_color_to_yuv420

c函数原型	hi_s32 hi_mpi_vpc_convert_color_to_yuv420 (hi_vpc_chn chn, const hi_vpc_pic_info *source_pic, hi_vpc_pic_info *dest_pic, hi_u32 *task_id, hi_s32 milli_sec);
Python函数	task_id, ret = acl.himpi.vpc_convert_color_to_yuv420(chn, source_pic, dest_pic, milli_sec)
函数功能	转换图片的格式，且转换后的格式仅支持YUV420 semi-planar/YVU420 semi-planar。异步接口。 Atlas 200/300/500 推理产品，当前版本不支持该接口。 Atlas 训练系列产品，当前版本不支持该接口。

输入说明	chn: int, 图片处理通道号。 • Atlas 推理系列产品, 该参数的取值范围: [0, 256), 通道总数最多256。 • Atlas A2训练系列产品/Atlas 800I A2推理产品, 该参数的取值范围: [0, 256), 通道总数最多256。 • Atlas 200I/500 A2推理产品, 该参数的取值范围: [0, 128), 通道总数最多128。 source_pic: 原始图片信息, 具体请参见2.10.8.5.1 hi_vpc_pic_info, 相关约束请参见约束说明。 dest_pic: 目标图片信息, 具体请参见2.10.8.5.1 hi_vpc_pic_info。 需指定输出图片的宽、高、宽Stride、高Stride、内存地址、内存大小、格式等。 • 配置输出图片的宽、高时, 需与输入图片的宽、高保持一致, 否则会返回报错。 如果将输出图片的宽或高配置为0, 则VPC内部会将输入图片的宽、高分别作为输出图片的宽、高; 同时, VPC内部会按对齐要求计算宽Stride、高Stride, 不同图片格式的对齐要求不同, 请参见图片格式、宽高对齐、内存约束。 • 配置输出图片的宽Stride、高Stride时, 满足对齐要求即可, 不同图片格式的对齐要求不同, 请参见图片格式、宽高对齐、内存约束。 milli_sec: int, 超时时间配置, 单位是毫秒, 取值范围如下。 • “-1”: 阻塞方式。 • “0”: 非阻塞方式。 • >0: 超时方式, 配置具体的超时时间。超时时间受操作系统影响, 一般偏差在操作系统的—个时间片内, 例如, 操作系统的—个时间片为4ms, 用户设置的milli_sec参数值为1, 则实际的超时时间在1ms到5ms范围内。在CPU负载高场景下, 超时时间仍可能存在波动。
返回值说明	ret: int, 错误码。 • 返回0表示成功。 • 返回非0表示失败, 参见2.10.9.2 VPC图像处理返回码。 task_id: 此次任务分配的ID, 用来区分任务。

约束说明

- 本接口兼容旧昇腾AI处理器版本上的格式转换接口, 兼容场景下的输入、输出图片格式如下, 兼容场景下输出的二进制相同:
 - 输入图片格式: YUV420 semi-planar/YVU420 semi-planar/YUV422 semi-planar/YVU422 semi-planar/YUV444 packed/YUV444 semi-planar/YVU444 semi-planar
 - 输出图片格式: YUV420 semi-planar/YVU420 semi-planar
 - 在以上输入图片格式、输出图片格式场景下, 本接口与[2.10.4.10 acl.himpi.vpc_convert_color](#)接口输出的二进制不一致。

- 关于输入分辨率的要求：

版本	约束
Atlas 推理系列产品	输入图片分辨率：10 * 6 ~ 4096 * 4096（包括4096 * 4096）。
Atlas 200I/500 A2推理产品 Atlas A2训练系列产品/Atlas 800I A2推理产品	输入图片分辨率：10 * 6 ~ 4096 * 8192（包括4096 * 8192）。

- 输入图片格式支持[hi_pixel_format](#)枚举值中的如下枚举项：

```
HI_PIXEL_FORMAT_YUV_SEMIPLANAR_420 = 1, // YUV420SP NV12 8bit
HI_PIXEL_FORMAT_YVU_SEMIPLANAR_420 = 2, // YUV420SP NV21 8bit
HI_PIXEL_FORMAT_YUV_SEMIPLANAR_422 = 3, // YUV422SP 8bit
HI_PIXEL_FORMAT_YVU_SEMIPLANAR_422 = 4, // YVU422SP 8bit
HI_PIXEL_FORMAT_YUV_SEMIPLANAR_444 = 5, // YUV444SP 8bit
HI_PIXEL_FORMAT_YVU_SEMIPLANAR_444 = 6, // YVU444SP 8bit
HI_PIXEL_FORMAT_YUYV_PACKED_422 = 7, // YUV422Packed YUYV 8bit
HI_PIXEL_FORMAT_UYVY_PACKED_422 = 8, // YUV422Packed UYVY 8bit
HI_PIXEL_FORMAT_YVYU_PACKED_422 = 9, // YUV422Packed YVYU 8bit
HI_PIXEL_FORMAT_VYUY_PACKED_422 = 10, // YUV422Packed VYUY 8bit
HI_PIXEL_FORMAT_YUV_PACKED_444 = 11, // YUV444Packed 8bit
HI_PIXEL_FORMAT_RGB_888 = 12, // RGB888
HI_PIXEL_FORMAT_BGR_888 = 13, // BGR888
HI_PIXEL_FORMAT_ARGB_8888 = 14, // ARGB8888
HI_PIXEL_FORMAT_ABGR_8888 = 15, // ABGR8888
HI_PIXEL_FORMAT_RGBA_8888 = 16, // RGBA8888
HI_PIXEL_FORMAT_BGRA_8888 = 17, // BGRA8888
HI_PIXEL_FORMAT_YUV_SEMIPLANAR_440 = 1000, // YUV440SP 8bit
HI_PIXEL_FORMAT_YVU_SEMIPLANAR_440 = 1001, // YVU440SP 8bit
```

- 输出图片格式支持[hi_pixel_format](#)枚举值中的如下枚举项，设置其它图片格式无效，接口会返回失败：

```
HI_PIXEL_FORMAT_YUV_SEMIPLANAR_420 = 1, // YUV420SP NV12 8bit
HI_PIXEL_FORMAT_YVU_SEMIPLANAR_420 = 2, // YUV420SP NV21 8bit
```

- 关于各图片格式的宽高对齐、内存大小约束，请参见[图片格式、宽高对齐、内存约束](#)。

2.10.4.13 acl.himpi.vpc_copy_make_border

c函数原型	<code>hi_s32 hi_mpi_vpc_copy_make_border(hi_vpc_chn chn, const hi_vpc_pic_info *source_pic, hi_vpc_pic_info *dest_pic, hi_vpc_make_border_info make_border_info, hi_u32 *task_id, hi_s32 milli_sec);</code>
Python函数	<code>task_id, ret = acl.himpi.vpc_copy_make_border(chn, source_pic, dest_pic, make_border_info, milli_sec)</code>
函数功能	对输入图像进行边界填充。异步接口。 Atlas 200/300/500 推理产品 ，当前版本不支持该接口。 Atlas 训练系列产品 ，当前版本不支持该接口。

输入说明	chn: int, 图片处理通道号。 ● Atlas 推理系列产品, 该参数的取值范围: [0, 256), 通道总数最多256。 ● Atlas A2训练系列产品/Atlas 800I A2推理产品, 该参数的取值范围: [0, 256), 通道总数最多256。 ● Atlas 200I/500 A2推理产品, 该参数的取值范围: [0, 128), 通道总数最多128。 source_pic: dict, 原始图片信息, 具体请参见2.10.8.5.1 hi_vpc_pic_info, 相关约束请参见约束说明。 dest_pic: dict, 目标图片信息, 具体请参见2.10.8.5.1 hi_vpc_pic_info。 make_border_info: dict, 边界填充信息, 具体请参见2.10.8.5.8 hi_vpc_make_border_info。 milli_sec: int, 超时时间配置, 单位是毫秒, 取值范围如下。 ● “-1”: 阻塞方式。 ● “0”: 非阻塞方式。 ● >0: 超时方式, 配置具体的超时时间。超时时间受操作系统影响, 一般偏差在操作系统的一个时间片内, 例如, 操作系统的 一个时间片为4ms, 用户设置的milli_sec参数值为1, 则实际的 超时时间在1ms到5ms范围内。在CPU负载高场景下, 超时时间 仍可能存在波动。
返回值说明	ret: int, 错误码。 ● 返回0表示成功。 ● 返回非0表示失败, 参见2.10.9.2 VPC图像处理返回码。 task_id: int, 任务分配的ID, 用来区分任务。

约束说明

版本	约束
Atlas 推理系列产品	● 输入、输出图片分辨率为10 * 6 ~ 4096 * 4096。 ● 输入、输出图片的格式、各图片格式的宽高对齐、内存大小约束, 请参见图片格式、宽高对齐、内存约束。扩充图像的宽没有16对齐的要求, 但是内存申请仍然要按照16的对齐要求进行申请, 即Stride要按照16对齐。 ● 由于YUV格式图像下采样约束, 当输出图片格式为YUV420SP或YUV422SP格式时, 需注意: - 对于YUV420SP输出格式, 上下左右填充的尺寸建议为偶数。 - 对于YUV422SP输出格式, 左右填充的尺寸建议为偶数。

版本	约束
Atlas 200I/500 A2推理产品 Atlas A2训练系列产品/ Atlas 800I A2推理产品	• 输入、输出图片分辨率为10 * 6 ~ 4096 * 8192。 • 输入、输出图片的格式、各图片格式的宽高对齐、内存大小约束，请参见图片格式、宽高对齐、内存约束。 • 根据make_border_info["border_type"]参数设置的扩充边缘像素的类型，本接口支持的输入、输出图片格式有所不同： - 当参数值设置为“HI_BORDER_CONSTANT”或“HI_BORDER_REPLICATE”时，支持的输入、输出图片格式请参见图片格式、宽高对齐、内存约束。 - 当参数值设置为“HI_BORDER_REFLECT”或“HI_BORDER_REFLECT_101”时，支持的输入、输出图片格式如下所示：<pre>HI_PIXEL_FORMAT_YUV_400 = 0, // YUV400 8bit HI_PIXEL_FORMAT_YUYV_PAKED_422 = 7, // YUV422Packed YUYV 8bit HI_PIXEL_FORMAT_UYVY_PAKED_422 = 8, // YUV422Packed UYVY 8bit HI_PIXEL_FORMAT_YVYU_PAKED_422 = 9, // YUV422Packed YVYU 8bit HI_PIXEL_FORMAT_VYUY_PAKED_422 = 10, // YUV422Packed VYUY 8bit HI_PIXEL_FORMAT_YUV_PAKED_444 = 11, // YUV444 Package 8bit HI_PIXEL_FORMAT_RGB_888 = 12, // RGB888 HI_PIXEL_FORMAT_BGR_888 = 13, // BGR888 HI_PIXEL_FORMAT_ARGB_8888 = 14, // ARGB8888 HI_PIXEL_FORMAT_ABGR_8888 = 15, // ABGR8888 HI_PIXEL_FORMAT_RGBA_8888 = 16, // RGBA8888 HI_PIXEL_FORMAT_BGRA_8888 = 17, // BGRA8888 HI_PIXEL_FORMAT_YVU_PAKED_444 = 58, // YVU444 Package 8bit</pre> • 各图片格式的宽高对齐、内存大小约束，请参见图片格式、宽高对齐、内存约束。 • 由于YUV格式图像下采样约束，当输出图片格式为YUV420SP或YUV422SP格式时，需注意： - 对于YUV420SP输出格式，上下左右填充的尺寸建议为偶数； - 对于YUV422SP输出格式，左右填充的尺寸建议为偶数。

2.10.4.14 acl.himpi.vpc_pyrdown

c函数原型	hi_s32 hi_mpi_vpc_pyrdown(hi_vpc_chn chn, const hi_vpc_pic_info *source_pic, hi_vpc_pic_info dest_pic[], hi_u32 filter_level, hi_s8 gaussian_filter[][5], hi_u16 divisor, hi_vpc_make_border_info make_border_info, hi_u32 *task_id, hi_s32 milli_sec);

Python函数	<code>task_id, ret = acl.himpi.vpc_pyrdown(chn, source_pic, dest_pic, filter_level, gaussian_filter, divisor, make_border_info, milli_sec)</code>
函数功能	对图像进行金字塔缩放，当前仅支持YUV400的图片格式。异步接口。 Atlas 200/300/500 推理产品，当前版本不支持该接口。 Atlas 训练系列产品，当前版本不支持该接口。 Atlas 200I/500 A2推理产品，不支持该接口。 Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

输入说明	chn: int, 图片处理通道号。 Atlas 推理系列产品, 该参数的取值范围: [0, 256), 通道总数最多256。 source_pic: dict, 输入图片信息, 具体请参见2.10.8.5.1 hi_vpc_pic_info。输入图片的分辨率最大支持2048 * 2048, 仅支持YUV 400格式输入。根据“filter_level”, 最小输入分辨率限制如下: - filter_level = 1, 最小输入分辨率为20 * 12。 - filter_level = 2, 最小输入分辨率为40 * 24。 - filter_level = 3, 最小输入分辨率为80 * 48。 - filter_level = 4, 最小输入分辨率为160 * 96。 dest_pic: dict, 目标图片信息, 具体请参见2.10.8.5.1 hi_vpc_pic_info, 该数组长度与filter_level参数值保持一致。 - dest_pic[0]保存的是原图宽高都缩小为1 / 2的图。 - dest_pic[1]保存的是原图宽高都缩小为1 / 4的图。 - dest_pic[2]保存的是原图宽高都缩小为1 / 8的图。 - dest_pic[3]保存的是原图宽高都缩小为1 / 16的图。 filter_level: int, 参数有效范围为[1, 4], 指定金字塔图像层数以及图像数量。 gaussian_filter: 高斯滤波参数, 需传入5 * 5的列表, 可通过传入“None”使用默认值, 具体参见如下: <pre>gaussian_filter = [[1, 4, 6, 4, 1], [4, 16, 24, 16, 4], [6, 24, 36, 24, 6], [4, 16, 24, 16, 4], [1, 4, 6, 4, 1]]</pre> divisor: int, 滤波器除数, 必须是2的幂次方。 make_border_info: dict, 边界填充信息, 具体请参见2.10.8.5.8 hi_vpc_make_border_info。填充固定填充2个像素, 支持的填充类型为: “HI_BORDER_CONSTANT”、 “HI_BOARD_REPLICATE” 和 “HI_BOARD_REFLECT”。 milli_sec: int, 超时时间配置, 单位是毫秒, 取值范围如下。 “-1”: 阻塞方式。 “0”: 非阻塞方式。 >0: 超时方式, 配置具体的超时时间。超时时间受操作系统影响, 一般偏差在操作系统的一个时间片内, 例如, 操作系统的 一个时间片为4ms, 用户设置的milli_sec参数值为1, 则实际的 超时时间在1ms到5ms范围内。在CPU负载高场景下, 超时时间 仍可能存在波动。
返回值说明	ret: int, 错误码。 - 返回0表示成功。 - 返回非0表示失败, 参见2.10.9.2 VPC图像处理返回码。 task_id: int, 任务分配的ID, 用来区分任务。

2.10.4.15 acl.himpi.vpc_calc_hist

须知	
此接口后续版本会废弃，请使用 2.10.4.16 acl.himpi.vpc_calc_hist_v2 接口。	
c函数原型	hi_s32 hi_mpi_vpc_calc_hist(hi_vpc_chn chn, const hi_vpc_pic_info *source_pic, hi_vpc_histogram_config *hist_config, hi_u32 *task_id, hi_s32 milli_sec);
Python函数	hist_config, task_id, ret = acl.himpi.vpc_calc_hist(chn, source_pic, milli_sec)
函数功能	统计图像每个通道的像素值分布。异步接口。 Atlas 200/300/500 推理产品，当前版本不支持该接口。 Atlas 训练系列产品，当前版本不支持该接口。

输入说明	chn: int, 图片处理通道号。 ● Atlas 推理系列产品, 该参数的取值范围: [0, 256), 通道总数最多256。 ● Atlas A2训练系列产品/Atlas 800I A2推理产品, 该参数的取值范围: [0, 256), 通道总数最多256。 ● Atlas 200I/500 A2推理产品, 该参数的取值范围: [0, 128), 通道总数最多128。 source_pic: 输入图片信息。 ● Atlas 推理系列产品, 输入图片分辨率: 10 * 6 ~ 4096 * 4096 (包括4096)。 ● Atlas A2训练系列产品/Atlas 800I A2推理产品, 输入图片分辨率: 10 * 6 ~ 4096 * 8192 (包括4096 * 8192)。 ● Atlas 200I/500 A2推理产品, 输入图片分辨率: 10 * 6 ~ 4096 * 8192 (包括4096 * 8192)。 ● 输入图片格式支持hi_pixel_format枚举值中的如下枚举项: <pre>HI_PIXEL_FORMAT_YUV_400 = 0, // YUV400 8bit HI_PIXEL_FORMAT_YUV_SEMIPLANAR_420 = 1, // YUV420SP NV12 8bit HI_PIXEL_FORMAT_YVU_SEMIPLANAR_420 = 2, // YUV420SP NV21 8bit HI_PIXEL_FORMAT_YUV_SEMIPLANAR_422 = 3, // YUV422SP 8bit HI_PIXEL_FORMAT_YVU_SEMIPLANAR_422 = 4, // YVU422SP 8bit HI_PIXEL_FORMAT_YUV_SEMIPLANAR_444 = 5, // YUV444SP 8bit HI_PIXEL_FORMAT_YVU_SEMIPLANAR_444 = 6, // YVU444SP 8bit HI_PIXEL_FORMAT_YUYV_PACKED_422 = 7, // YUV422Packed YUYV 8bit HI_PIXEL_FORMAT_UYVY_PACKED_422 = 8, // YUV422Packed UYVY 8bit HI_PIXEL_FORMAT_YVYU_PACKED_422 = 9, // YUV422Packed YVYU 8bit HI_PIXEL_FORMAT_VYUY_PACKED_422 = 10, // YUV422Packed VYUY 8bit HI_PIXEL_FORMAT_YUV_PACKED_444 = 11, // YUV444P 8bit HI_PIXEL_FORMAT_RGB_888 = 12, // RGB888 HI_PIXEL_FORMAT_BGR_888 = 13, // BGR888 HI_PIXEL_FORMAT_YUV_SEMIPLANAR_440 = 1000, // YUV440SP 8bit HI_PIXEL_FORMAT_YVU_SEMIPLANAR_440 = 1001, // YVU440SP 8bit</pre> 说明 当图片格式为YUV440SP 8bit、YVU440SP 8bit时, Y分量统计准确, UV分量统计上存在误差。 milli_sec: int, 超时时间配置, 单位是毫秒, 取值范围如下。 ● “-1”: 阻塞方式。 ● “0”: 非阻塞方式。 ● >0: 超时方式, 配置具体的超时时间。超时时间受操作系统影响, 一般偏差在操作系统的 一个时间片内, 例如, 操作系统的 一个时间片为4ms, 用户设置的milli_sec参数值为1, 则实际的 超时时间在1ms到5ms范围内。在CPU负载高场景下, 超时时间 仍可能存在波动。

返回值说明	hist_config: 统计结果。 “hist_config”是一个包含3个数组的结构体，用来存放每张图像3个分量的0-255的分布情况。 该接口即将废弃，当前使用本接口获取到的“hist_config”为全0信息，请使用2.10.4.16 acl.himpi.vpc_calc_hist_v2代替。 task_id: 任务分配的ID，用来区分任务。 ret: int, 错误码。 • 返回0表示成功。 • 返回非0表示失败，参见2.10.9.2 VPC图像处理返回码。 • 该接口即将废弃，当前使用本接口获取到的ret为0，请使用2.10.4.16 acl.himpi.vpc_calc_hist_v2代替。

2.10.4.16 acl.himpi.vpc_calc_hist_v2

c函数原型	<pre>hi_s32 hi_mpi_vpc_calc_hist(hi_vpc_chn chn, const hi_vpc_pic_info *source_pic, hi_vpc_histogram_config *hist_config, hi_u32 *task_id, hi_s32 milli_sec);</pre>
Python函数	<pre>task_id, ret = acl.himpi.vpc_calc_hist_v2(chn, source_pic, milli_sec, hist_config)</pre>
函数功能	统计图像每个通道的像素值分布。异步接口。 Atlas 200/300/500 推理产品，当前版本不支持该接口。 Atlas 训练系列产品，当前版本不支持该接口。

输入说明	chn: int, 图片处理通道号。 ● Atlas 推理系列产品, 该参数的取值范围: [0, 256), 通道总数最多256。 ● Atlas A2训练系列产品/Atlas 800I A2推理产品, 该参数的取值范围: [0, 256), 通道总数最多256。 ● Atlas 200I/500 A2推理产品, 该参数的取值范围: [0, 128), 通道总数最多128。 source_pic: dict, 输入图片信息, 具体请参见2.10.8.5.1 hi_vpc_pic_info, 相关约束请参见约束说明。 说明 当图片格式为YUV440SP 8bit、YVU440SP 8bit时, Y分量统计准确, UV分量统计上存在误差。 milli_sec: int, 超时时间配置, 单位是毫秒, 取值范围如下。 ● “-1”: 阻塞方式。 ● “0”: 非阻塞方式。 ● >0: 超时方式, 配置具体的超时时间。超时时间受操作系统影响, 一般偏差在操作系统的—个时间片内, 例如, 操作系统的—个时间片为4ms, 用户设置的milli_sec参数值为1, 则实际的超时时间在1ms到5ms范围内。在CPU负载高场景下, 超时时间仍可能存在波动。 hist_config: 统计结果指针地址。 “hist_config” 中的地址指向包含统计结果的结构体, 用来存放每张图像3个分量0~255的分布情况。 调用np_out_size = np.array([out_buffer_size], dtype=np.int32)创建numpy array类型数据。 再调用bytes_data = np_out_size.tobytes()-> \->size_ptr = acl.util.bytes_to_ptr(bytes_data), 将数据转换为对象作为入参。函数调用成功后, “size_ptr” 指向的数据 (即np_out_size) 会发生变化。
返回值说明	ret: int, 错误码。 ● 返回0表示成功。 ● 返回非0表示失败, 参见2.10.9.2 VPC图像处理返回码。 task_id: 任务分配的ID, 用来区分任务。

约束说明

- 关于输入分辨率的要求:

版本	约束
Atlas 推理系列产品	输入图片分辨率: 10 * 6 ~ 4096 * 4096 (包括4096 * 4096)。

版本	约束
Atlas 200I/500 A2推理产品 Atlas A2训练系列产品/Atlas 800I A2推理产品	输入图片分辨率：10 * 6 ~ 4096 * 8192（包括4096 * 8192）。

- 输入图片格式支持[hi_pixel_format](#)枚举值中的如下枚举项：

HI_PIXEL_FORMAT_YUV_400 = 0, // YUV400 8bit

HI_PIXEL_FORMAT_YUV_SEMIPLANAR_420 = 1, // YUV420SP NV12 8bit

HI_PIXEL_FORMAT_YVU_SEMIPLANAR_420 = 2, // YUV420SP NV21 8bit

HI_PIXEL_FORMAT_YUV_SEMIPLANAR_422 = 3, // YUV422SP 8bit

HI_PIXEL_FORMAT_YVU_SEMIPLANAR_422 = 4, // YVU422SP 8bit

HI_PIXEL_FORMAT_YUV_SEMIPLANAR_444 = 5, // YUV444SP 8bit

HI_PIXEL_FORMAT_YVU_SEMIPLANAR_444 = 6, // YVU444SP 8bit

HI_PIXEL_FORMAT_YUYV_PACKED_422 = 7, // YUV422Packed YUYV 8bit

HI_PIXEL_FORMAT_UYVY_PACKED_422 = 8, // YUV422Packed UYVY 8bit

HI_PIXEL_FORMAT_YVYU_PACKED_422 = 9, // YUV422Packed YVYU 8bit

HI_PIXEL_FORMAT_VYUY_PACKED_422 = 10, // YUV422Packed VYUY 8bit

HI_PIXEL_FORMAT_YUV_PACKED_444 = 11, // YUV444P 8bit

HI_PIXEL_FORMAT_RGB_888 = 12, // RGB888

HI_PIXEL_FORMAT_BGR_888 = 13, // BGR888

HI_PIXEL_FORMAT_YUV_SEMIPLANAR_440 = 1000, // YUV440SP 8bit

HI_PIXEL_FORMAT_YVU_SEMIPLANAR_440 = 1001, // YVU440SP 8bit
- 关于各图片格式的宽高对齐、内存大小约束，请参见[图片格式](#)、[宽高对齐](#)、[内存约束](#)。

2.10.4.17 acl.himpi.vpc_equalize_hist

c函数原型	hi_s32 hi_mpi_vpc_equalize_hist(hi_vpc_chn chn, const hi_vpc_pic_info* source_pic, hi_vpc_pic_info* dest_pic, const hi_vpc_lut_remap *lut_remap, hi_u32 *task_id, hi_s32 milli_sec);
Python函数	task_id, ret = acl.himpi.vpc_equalize_hist(chn, source_pic, dest_pic, lut_remap, milli_sec)
函数功能	将输入图片中指定分量的指定像素值重映射成其它像素值，作为输出图片，主要用户图像增强的场景。通过图片的色彩重映射以达到图片对比度更好，清晰度更好的效果。异步接口。 Atlas 200/300/500 推理产品，当前版本不支持该接口。 Atlas 训练系列产品，当前版本不支持该接口。

输入说明	chn: int, 图片处理通道号。 • Atlas 推理系列产品, 该参数的取值范围: [0, 256), 通道总数最多256。 • Atlas A2训练系列产品/Atlas 800I A2推理产品, 该参数的取值范围: [0, 256), 通道总数最多256。 • Atlas 200I/500 A2推理产品, 该参数的取值范围: [0, 128), 通道总数最多128。 source_pic: dict, 输入图片信息, 具体请参见2.10.8.5.1 hi_vpc_pic_info, 相关约束请参见约束说明。 dest_pic: dict, 目标图片信息, 具体请参见2.10.8.5.1 hi_vpc_pic_info。输出图片格式与输入图片格式保持一致。 lut_remap: dict, 图像“lut_remap”的配置, 具体请参见2.10.8.5.10 hi_vpc_lut_remap。“lut_remap”是一个768 Byte的数组, 配置给硬件进行图片像素重映射。 milli_sec: int, 超时时间配置, 单位是毫秒, 取值范围如下。 • “-1”: 阻塞方式。 • “0”: 非阻塞方式。 • >0: 超时方式, 配置具体的超时时间。超时时间受操作系统影响, 一般偏差在操作系统的—个时间片内, 例如, 操作系统的—个时间片为4ms, 用户设置的milli_sec参数值为1, 则实际的超时时间在1ms到5ms范围内。在CPU负载高场景下, 超时时间仍可能存在波动。
返回值说明	ret: int, 错误码。 • 返回0表示成功。 • 返回非0表示失败, 参见2.10.9.2 VPC图像处理返回码。 task_id: 任务分配的ID, 用来区分任务。

约束说明

- 输入图片分辨率为10 * 6 ~ 4096 * 4096 (包括4096)。
- 输入图片格式支持[hi_pixel_format](#)枚举值中的如下枚举项:

```
HI_PIXEL_FORMAT_YUV_400 = 0,           // YUV400 8bit
HI_PIXEL_FORMAT_YUV_SEMIPLANAR_420 = 1, // YUV420SP NV12 8bit
HI_PIXEL_FORMAT_YVU_SEMIPLANAR_420 = 2, // YUV420SP NV21 8bit
HI_PIXEL_FORMAT_YUV_SEMIPLANAR_422 = 3, // YUV422SP 8bit
HI_PIXEL_FORMAT_YVU_SEMIPLANAR_422 = 4, // YVU422SP 8bit
HI_PIXEL_FORMAT_YUV_PACKED_444 = 11,    // YUV444P 8bit
HI_PIXEL_FORMAT_RGB_888 = 12,           // RGB888
HI_PIXEL_FORMAT_BGR_888 = 13,           // BGR888
```
- 输出图片格式与输入图片格式保持一致。
- 关于各图片格式的宽高对齐、内存大小约束, 请参见[图片格式、宽高对齐、内存约束](#)。

2.10.4.18 acl.himpi.vpc_get_process_result

c函数原型	hi_s32 hi_mpi_vpc_get_process_result(hi_vpc_chn chn, hi_u32 task_id, hi_s32 milli_sec);
Python函数	ret = acl.himpi.vpc_get_process_result(chn, task_id, milli_sec)
函数功能	等待任务处理完成，再获取图片处理结果。 由于VPC功能接口是异步接口（例如：2.10.4.7 acl.himpi.vpc_crop），因此在调用VPC功能接口之后，还需要调用本接口等待任务处理完成，然后再获取处理结果。 Atlas 200/300/500 推理产品，当前版本不支持该接口。 Atlas 训练系列产品，当前版本不支持该接口。
输入说明	chn: int，图片处理通道号。 • Atlas 推理系列产品，该参数的取值范围：[0, 256)，通道总数最多256。 • Atlas A2训练系列产品/Atlas 800I A2推理产品，该参数的取值范围：[0, 256)，通道总数最多256。 • Atlas 200I/500 A2推理产品，该参数的取值范围：[0, 128)，通道总数最多128。 milli_sec: int，超时时间配置，单位是毫秒，取值范围如下。 • “-1”：阻塞方式。 • “0”：非阻塞方式。 • >0：超时方式，配置具体的超时时间。超时时间受操作系统影响，一般偏差在操作系统的一个时间片内，例如，操作系统的—个时间片为4ms，用户设置的milli_sec参数值为1，则实际的超时时间在1ms到5ms范围内。在CPU负载高场景下，超时时间仍可能存在波动。 task_id: 任务ID。 • 此处通过任务ID来明确需要获取哪次任务执行后的结果数据。 • 任务ID可以在调用抠图、缩放等接口后获取，请参见acl.himpi.vpc_resize~acl.himpi.vpc_batch_crop_resize_make_border。
返回值说明	ret: int，错误码。 • 返回0表示成功。 • 返回非0表示失败，参见2.10.9.2 VPC图像处理返回码。
参考资源	接口调用流程，参见VPC图片处理典型功能。

2.10.4.19 acl.himpi.vpc_sys_create_chn

c函数原型	hi_s32 hi_mpi_vpc_sys_create_chn(hi_vpc_chn *chnl, const hi_vpc_chn_attr *attr);

Python函数	<code>chnl, ret = acl.himpi.vpc_sys_create_chn(attr)</code>
函数功能	根据设置的通道属性创建图像处理通道，由系统返回可用通道号给用户。 Atlas 200/300/500 推理产品，当前版本不支持该接口。 Atlas 训练系列产品，当前版本不支持该接口。
输入说明	attr: dict，图片处理通道属性字典，具体请参见2.10.8.5.13 hi_vpc_chn_attr。如果参数“attr”为空，会返回错误码“HI_ERR_VPC_NULL_PTR”。
返回值说明	ret: int，错误码。 • 返回0表示成功。 • 返回非0表示失败，参见2.10.9.2 VPC图像处理返回码。 chnl: int，图片处理通道号，调用该接口后，由系统返回可用通道号给用户。 • Atlas 推理系列产品，该参数的取值范围：[0, 256)，通道总数最多256。 • Atlas A2训练系列产品/Atlas 800I A2推理产品，该参数的取值范围：[0, 256)，通道总数最多256。 • Atlas 200I/500 A2推理产品，该参数的取值范围：[0, 128)，通道总数最多128。

2.10.4.20 acl.himpi.vpc_crop_resize_make_border

c函数原型	<code>hi_s32 hi_mpi_vpc_crop_resize_make_border(hi_vpc_chn chn, const hi_vpc_pic_info *source_pic, hi_vpc_crop_resize_border_region crop_resize_make_border_info[], hi_u32 count, hi_u32 *task_id, hi_s32 milli_sec);</code>
Python函数	<code>task_id, ret = acl.himpi.vpc_crop_resize_make_border(chn, source_pic, crop_resize_make_border_info, count, milli_sec)</code>
函数功能	按指定区域从一张输入图片中抠出一个或多个子图，对子图缩放后，再将每个子图按指定类型填充，作为一张或多张目标图片输出，主要用于等比例缩放场景。异步接口。 Atlas 200/300/500 推理产品，当前版本不支持该接口。 Atlas 训练系列产品，当前版本不支持该接口。

输入说明	chn: int, 图片处理通道号。 • Atlas 推理系列产品, 该参数的取值范围: [0, 256), 通道总数最多256。 • Atlas A2训练系列产品/Atlas 800I A2推理产品, 该参数的取值范围: [0, 256), 通道总数最多256。 • Atlas 200I/500 A2推理产品, 该参数的取值范围: [0, 128), 通道总数最多128。 source_pic: dict, 原始图片信息, 具体请参见2.10.8.5.1 hi_vpc_pic_info, 相关约束请参见约束说明。 crop_resize_make_border_info: list, 抠图缩放填充的图片信息列表, 需要用户设置抠图、缩放、填充信息以及目标图片的内存地址, VPC将抠图缩放填充后的结果数据存放在用户指定的内存地址中, 具体请参见2.10.8.5.14 hi_vpc_crop_resize_border_region。 如果多个填充结果的数据都指向同一个内存地址, 后一次的结果数据会覆盖前一次的结果数据, 导致输出图片跟预期不一致。 当前填充类型仅支持“HI_BORDER_CONSTANT”和“HI_BORDER_REPLICATE”, 可以补边到4096 * 4096。 “crop_resize_make_border_info”数组的长度与“count”参数值保持一致。 count: int, 抠图缩放填充的图片数量, 取值范围[1, 256]。 milli_sec: int, 超时时间配置, 单位是毫秒, 取值范围如下。 • “-1”: 阻塞方式。 • “0”: 非阻塞方式。 • >0: 超时方式, 配置具体的超时时间。超时时间受操作系统影响, 一般偏差在操作系统的一个时间片内, 例如, 操作系统的-一个时间片为4ms, 用户设置的milli_sec参数值为1, 则实际的超时时间在1ms到5ms范围内。在CPU负载高场景下, 超时时间仍可能存在波动。
返回值说明	ret: int, 错误码。 • 返回0表示成功。 • 返回非0表示失败, 参见2.10.9.2 VPC图像处理返回码。 task_id: 任务分配的ID, 用来区分任务。

约束说明

版本	约束
Atlas 推理系列产品	- 关于输入图片分辨率、图片格式： - 输入图片分辨率在$10 * 6 \sim 4096 * 4096$（包括4096）范围内时，支持图片格式、宽高对齐、内存约束处说明的输入图片格式。 - 输入图片的宽或高在$4096 \sim 8192$（包括8192）范围内时，支持格式如下： 支持hi_pixel_format枚举值中的如下枚举项： <pre>HI_PIXEL_FORMAT_YUV_400 = 0, // YUV400 8bit HI_PIXEL_FORMAT_YUV_SEMIPLANAR_420 = 1, // YUV420SP NV12 8bit HI_PIXEL_FORMAT_YVU_SEMIPLANAR_420 = 2, // YUV420SP NV21 8bit HI_PIXEL_FORMAT_YUV_SEMIPLANAR_422 = 3, // YUV422SP 8bit HI_PIXEL_FORMAT_YVU_SEMIPLANAR_422 = 4, // YVU422SP 8bit HI_PIXEL_FORMAT_YUV_SEMIPLANAR_444 = 5, // YUV444SP 8bit HI_PIXEL_FORMAT_YVU_SEMIPLANAR_444 = 6, // YVU444SP 8bit HI_PIXEL_FORMAT_YUYV_PACKED_422 = 7, // YUV422Packed YUYV 8bit HI_PIXEL_FORMAT_UYVY_PACKED_422 = 8, // YUV422Packed UYVY 8bit HI_PIXEL_FORMAT_YVYU_PACKED_422 = 9, // YUV422Packed YVYU 8bit HI_PIXEL_FORMAT_VYUY_PACKED_422 = 10, // YUV422Packed VYUY 8bit</pre> - 关于各图片格式的宽高对齐、内存大小约束，请参见图片格式、宽高对齐、内存约束。 - 关于输出图片分辨率、图片格式、宽高对齐、内存大小约束请参见2.10.4.2 约束说明。 - 当输入、输出图片格式都为ARGB8888、ABGR8888、RGBA8888、BGRA8888格式时，在经过本接口处理后，A分量值会变为0。 - 由于“hi_vpc_crop_resize_border_region”字典内容比较丰富，使用者可以自由组合功能。但存在一些约束，例如：若是crop的宽高与resize之后的宽高一致，则不进行缩放；若偏移值为0，则不进行填充。 - 使用本接口实现等比例缩放功能时，相关的描述及约束请参见抠图、贴图约束。 - 由于YUV格式图像下采样约束，当输出图片格式为YUV420SP或YUV422SP格式时，需注意： - 对于YUV420SP输出格式，上下左右填充的尺寸建议为偶数。 - 对于YUV422SP输出格式，左右填充的尺寸建议为偶数。

版本	约束
Atlas 200I/500 A2推理产品 Atlas A2训练系列产品/ Atlas 800I A2推理产品	- 关于图片分辨率、图片格式、宽高对齐、内存等约束，请参见2.10.4.2 约束说明。 - 设置fx/fy之后，缩放后的图片的分辨率必须在[10 * 6 ~ 16384 * 16384]范围内，否则返回错误。 - 当输入、输出图片格式都为ARGB8888、ABGR8888、RGBA8888、BGRA8888格式时，在经过本接口缩放处理后，A分量值会变为0。 - 由于“hi_vpc_crop_resize_border_region”字典内容比较丰富，使用者可以自由组合功能。但存在一些约束，例如：若是crop的宽高与resize之后的宽高一致，则不进行缩放；若偏移值为0，则不进行填充。 - 使用本接口实现等比例缩放功能时，相关的描述及约束请参见抠图、贴图约束。 - 由于YUV格式图像下采样约束，当输出图片格式为YUV420SP或YUV422SP格式时，需注意： - 对于YUV420SP输出格式，上下左右填充的尺寸建议为偶数。 - 对于YUV422SP输出格式，左右填充的尺寸建议为偶数。

2.10.4.21 acl.himpi.vpc_batch_crop_resize_paste

c函数原型	hi_s32 hi_mpi_vpc_batch_crop_resize_paste (hi_vpc_chn chn, const hi_vpc_pic_info *source_pic[], hi_u32 pic_num, hi_vpc_crop_resize_paste_region crop_resize_paste_info[], hi_u32 count[], hi_u32 *task_id, hi_s32 milli_sec);
Python函数	task_id, ret =acl.himpi.vpc_batch_crop_resize_paste(chn, source_pic, pic_num, crop_resize_paste_info, count, milli_sec)
函数功能	acl.himpi.vpc_crop_resize_paste的扩展接口，支持一次处理多张输入图片。可输入多张原图，从每张原图中抠一张或多张子图缩放并贴图到每张目标图上，目标图片的数量与子图数量保持一致。异步接口。 Atlas 200/300/500 推理产品，当前版本不支持该接口。 Atlas 训练系列产品，当前版本不支持该接口。

输入说明	chn: int, 图片处理通道号。 ● Atlas 推理系列产品, 该参数的取值范围: [0, 256), 通道总数最多256。 ● Atlas A2训练系列产品/Atlas 800I A2推理产品, 该参数的取值范围: [0, 256), 通道总数最多256。 ● Atlas 200I/500 A2推理产品, 该参数的取值范围: [0, 128), 通道总数最多128。 source_pic: dict, 原始图片信息, 具体请参见2.10.8.5.1 hi_vpc_pic_info, 相关约束请参见约束说明。 pic_num: int, 原始图片数量, 与“source_pic”列表长度、“count”数组长度保持一致。 crop_resize_paste_info: 抠图缩放并贴图信息字典链表, 需要用户设置抠图、缩放、贴图信息以及目标图片的内存地址, VPC将抠图缩放贴图后的结果数据存放在用户指定的内存地址中, 具体请参见2.10.8.5.6 hi_vpc_crop_resize_paste_region。 count: int, 每张原图抠图缩放贴图的图片数量, 数组内第一个元素的值表示第一张原图的抠图缩放贴图的图片数量, 第二个元素的值表示第二张原图的抠图缩放贴图的图片数量, 以此类推。 “count”数组内元素值之和等于“crop_resize_paste_info”数组的长度。“count”数组内元素值之和的取值范围[1, 256]。 milli_sec: int, 超时时间配置, 单位是毫秒, 取值范围如下。 ● “-1”: 阻塞方式。 ● “0”: 非阻塞方式。 ● >0: 超时方式, 配置具体的超时时间。超时时间受操作系统影响, 一般偏差在操作系统的-一个时间片内, 例如, 操作系统的-一个时间片为4ms, 用户设置的milli_sec参数值为1, 则实际的-超时时间在1ms到5ms范围内。在CPU负载高场景下, 超时时间-仍可能存在波动。
返回值说明	ret: int, 错误码。 ● 返回0表示成功。 ● 返回非0表示失败, 参见2.10.9.2 VPC图像处理返回码。 task_id: 任务分配的ID, 用来区分任务。

约束说明

版本	约束
Atlas 推理系列产品	- 关于输入图片分辨率、图片格式： - 输入图片分辨率在10 * 6 ~ 4096 * 4096（包括4096）范围内时，支持图片格式、宽高对齐、内存约束处说明的输入图片格式。 - 输入图片的宽或高在4096 ~ 8192（包括8192）范围内时，支持格式如下： 支持hi_pixel_format枚举值中的如下枚举项：<pre>HI_PIXEL_FORMAT_YUV_400 = 0, // YUV400 8bit HI_PIXEL_FORMAT_YUV_SEMIPLANAR_420 = 1, // YUV420SP NV12 8bit HI_PIXEL_FORMAT_YVU_SEMIPLANAR_420 = 2, // YUV420SP NV21 8bit HI_PIXEL_FORMAT_YUV_SEMIPLANAR_422 = 3, // YUV422SP 8bit HI_PIXEL_FORMAT_YVU_SEMIPLANAR_422 = 4, // YVU422SP 8bit HI_PIXEL_FORMAT_YUV_SEMIPLANAR_444 = 5, // YUV444SP 8bit HI_PIXEL_FORMAT_YVU_SEMIPLANAR_444 = 6, // YVU444SP 8bit HI_PIXEL_FORMAT_YUYV_PACKED_422 = 7, // YUV422Packed YUYV 8bit HI_PIXEL_FORMAT_UYVY_PACKED_422 = 8, // YUV422Packed UYVY 8bit HI_PIXEL_FORMAT_YVYU_PACKED_422 = 9, // YUV422Packed YVYU 8bit HI_PIXEL_FORMAT_VYUY_PACKED_422 = 10, // YUV422Packed VYUY 8bit</pre> - 关于各图片格式的宽高对齐、内存大小约束，请参见图片格式、宽高对齐、内存约束。 - 关于输出图片分辨率、图片格式、宽高对齐、内存大小约束请参见2.10.4.2 约束说明。 - 当输入、输出图片格式都为ARGB8888、ABGR8888、RGBA8888、BGRA8888格式时，在经过本接口缩放处理后，A分量值会变为0。 - 由于“hi_vpc_crop_resize_paste_region”字典内容比较丰富，使用者可以自由组合功能。但存在一些通用约束，例如： - 若是crop的宽高和原图宽高一致，则不进行crop。 - 若是crop的宽高与resize之后的宽高一致，则不进行缩放。 - 若是crop、resize之后的宽高和目标图片一致，并且贴图左上角坐标等于(0, 0)，则不进行贴图。

版本	约束
Atlas 200I/500 A2推理产品 Atlas A2训练系列产品/ Atlas 800I A2推理产品	- 关于图片分辨率、各图片格式的宽高对齐、内存大小约束，请参见2.10.4.2 约束说明。 - 当输入、输出图片格式都为ARGB8888、ABGR8888、RGBA8888、BGRA8888格式时，在经过本接口缩放处理后，A分量值会变为0。 - 由于YUV格式图像下采样约束，当输出图片格式为YUV420SP或YUV422SP格式时，贴图的顶部偏移、左侧偏移建议为偶数。 - 由于“hi_vpc_crop_resize_paste_region”字典内容比较丰富，使用者可以自由组合功能。但存在一些通用约束，例如： - 若是crop的宽高和原图宽高一致，则不进行crop。 - 若是crop的宽高与resize之后的宽高一致，则不进行缩放。 - 若是crop、resize之后的宽高和目标图片一致，并且贴图左上角坐标等于(0, 0)，则不进行贴图。

2.10.4.22 acl.himpi.vpc_batch_crop_resize_make_border

c函数原型	<pre>hi_s32 hi_mpi_vpc_batch_crop_resize_make_border(hi_vpc_chn chn, const hi_vpc_pic_info *source_pic[], hi_u32 pic_num, hi_vpc_crop_resize_border_region crop_resize_make_border_info[], hi_u32 count[], hi_u32 *task_id, hi_s32 milli_sec);</pre>
Python函数	<pre>task_id, ret = acl.himpi.vpc_batch_crop_resize_make_border(chn, source_pic, pic_num, crop_resize_make_border_info, count, milli_sec)</pre>
函数功能	acl.himpi.vpc_crop_resize_make_border的扩展接口，支持批处理模式。可输入多张原图，按指定区域从每张输入图片中抠出一个或多个子图，对子图缩放后，再将子图贴到每张目标图片（目标图片的数量与子图数量保持一致）的指定位置，区域之外按指定类型填充。异步接口。 Atlas 200/300/500 推理产品，当前版本不支持该接口。 Atlas 训练系列产品，当前版本不支持该接口。

输入说明	chn: int, 图片处理通道号。 ● Atlas 推理系列产品, 该参数的取值范围: [0, 256), 通道总数最多256。 ● Atlas A2训练系列产品/Atlas 800I A2推理产品, 该参数的取值范围: [0, 256), 通道总数最多256。 ● Atlas 200I/500 A2推理产品, 该参数的取值范围: [0, 128), 通道总数最多128。 source_pic: dict, 原始图片信息, 具体请参见2.10.8.5.1 hi_vpc_pic_info, 相关约束请参见约束说明。 pic_num: int, 原始图片数量, 与“source_pic”列表长度、“count”数组长度保持一致。 crop_resize_make_border_info: dict, 抠图缩放填充的信息字典, 需要用户设置抠图、缩放、填充信息以及目标图片的内存地址, VPC将抠图缩放填充后的结果数据存放在用户指定的内存地址中。当前填充类型仅支持“HI_BORDER_CONSTANT”, 可以填充到4096 * 4096, 具体请参见2.10.8.5.14 hi_vpc_crop_resize_border_region。 count: int, 每张原图抠图缩放填充的图片数量, “count”数组内元素值之和等于“crop_resize_make_border_info”数组的长度。“count”数组内元素值之和的取值范围[1, 256]。 milli_sec: int, 超时时间配置, 单位是毫秒, 取值范围如下。 ● “-1”: 阻塞方式。 ● “0”: 非阻塞方式。 ● >0: 超时方式, 配置具体的超时时间。超时时间受操作系统影响, 一般偏差在操作系统的-一个时间片内, 例如, 操作系统的-一个时间片为4ms, 用户设置的milli_sec参数值为1, 则实际的-超时时间在1ms到5ms范围内。在CPU负载高场景下, 超时时间-仍可能存在波动。
返回值说明	ret: int, 错误码。 ● 返回0表示成功。 ● 返回非0表示失败, 参见2.10.9.2 VPC图像处理返回码。 task_id: 任务分配的ID, 用来区分任务。

约束说明

版本	约束
Atlas 推理系列产品	- 关于输入图片分辨率、图片格式： - 输入图片分辨率在10 * 6 ~ 4096 * 4096（包括4096）范围内时，支持图片格式、宽高对齐、内存约束处说明的输入图片格式。 - 输入图片的宽或高在4096 ~ 8192（包括8192）范围内时，支持格式如下： 支持hi_pixel_format枚举值中的如下枚举项： HI_PIXEL_FORMAT_YUV_400 = 0, // YUV400 8bit HI_PIXEL_FORMAT_YUV_SEMIPLANAR_420 = 1, // YUV420SP NV12 8bit HI_PIXEL_FORMAT_YVU_SEMIPLANAR_420 = 2, // YUV420SP NV21 8bit HI_PIXEL_FORMAT_YUV_SEMIPLANAR_422 = 3, // YUV422SP 8bit HI_PIXEL_FORMAT_YVU_SEMIPLANAR_422 = 4, // YVU422SP 8bit HI_PIXEL_FORMAT_YUV_SEMIPLANAR_444 = 5, // YUV444SP 8bit HI_PIXEL_FORMAT_YVU_SEMIPLANAR_444 = 6, // YVU444SP 8bit HI_PIXEL_FORMAT_YUYV_PACKED_422 = 7, // YUV422Packed YUYV 8bit HI_PIXEL_FORMAT_UYVY_PACKED_422 = 8, // YUV422Packed UYVY 8bit HI_PIXEL_FORMAT_YVYU_PACKED_422 = 9, // YUV422Packed YVYU 8bit HI_PIXEL_FORMAT_VYUY_PACKED_422 = 10, // YUV422Packed VYUY 8bit - 关于各图片格式的宽高对齐、内存大小约束，请参见图片格式、宽高对齐、内存约束。 - 关于输出图片分辨率、图片格式、宽高对齐、内存大小约束请参见2.10.4.2 约束说明。 - 当输入、输出图片格式都为ARGB8888、ABGR8888、RGBA8888、BGRA8888格式时，在经过本接口缩放处理后，A分量值会变为0。 - 由于“hi_vpc_crop_resize_border_region”字典内容比较丰富，使用者可以自由组合功能。但存在一些通用约束，例如： - 若是crop的宽高和原图宽高一致，则不进行crop。 - 若是crop的宽高与resize之后的宽高一致，则不进行缩放。 - 若是crop、resize之后的宽高和目标图片的宽高一致，并且左上角坐标等于(0, 0)，则不进行填充。 - 使用本接口实现等比例缩放功能时，相关的描述及约束请参见抠图、贴图约束。 - 由于YUV格式图像下采样约束，当输出图片格式为YUV420SP或YUV422SP格式时，需注意： - 对于YUV420SP输出格式，上下左右填充的尺寸建议为偶数。

版本	约束
	- 对于YUV422SP输出格式，左右填充的尺寸建议为偶数。
Atlas 200I/500 A2推理产品 Atlas A2训练系列产品/ Atlas 800I A2推理产品	• 关于图片分辨率、各图片格式的宽高对齐、内存大小约束，请参见2.10.4.2 约束说明。 • 当输入、输出图片格式都为ARGB8888、ABGR8888、RGBA8888、BGRA8888格式时，在经过本接口缩放处理后，A分量值会变为0。 • 由于“hi_vpc_crop_resize_border_region”字典内容比较丰富，使用者可以自由组合功能。但存在一些通用约束，例如： - 若是crop的宽高和原图宽高一致，则不进行crop。 - 若是crop的宽高与resize之后的宽高一致，则不进行缩放。 - 若是crop、resize之后的宽高和目标图片的宽高一致，并且左上角坐标等于(0, 0)，则不进行填充。 • 使用本接口实现等比例缩放功能时，相关的描述及约束请参见抠图、贴图约束。 • 由于YUV格式图像下采样约束，当输出图片格式为YUV420SP或YUV422SP格式时，需注意： - 对于YUV420SP输出格式，上下左右填充的尺寸建议为偶数。 - 对于YUV422SP输出格式，左右填充的尺寸建议为偶数。

2.10.4.23 acl.himpi.vpc_blur

c函数原型	hi_s32 hi_mpi_vpc_blur(hi_vpc_chn chn, const hi_blur_param* blur_param, hi_u32 *task_id, hi_s32 milli_sec)
Python函数	task_id, ret = acl.himpi.vpc_blur(chn, blur_param, milli_sec)
函数功能	对输入图片做均值滤波处理。 Atlas 200/300/500 推理产品 ，不支持该接口。 Atlas 训练系列产品 ，不支持该接口。 Atlas 推理系列产品 ，不支持该接口。

输入说明	chn: int, 图片处理通道号。 ● Atlas A2训练系列产品/Atlas 800I A2推理产品, 该参数的取值范围: [0, 256), 通道总数最多256。 ● Atlas 200I/500 A2推理产品, 该参数的取值范围: [0, 128), 通道总数最多128。 blur_param: dict, 滤波信息字典, 具体请参见2.10.8.5.28 hi_blur_param, 相关约束请参见约束说明。 milli_sec: int, 超时时间配置, 单位是毫秒, 取值范围如下。 ● “-1”: 阻塞方式。 ● “0”: 非阻塞方式。 ● >0: 超时方式, 配置具体的超时时间。超时时间受操作系统影响, 一般偏差在操作系统的 一个时间片内, 例如, 操作系统的 一个时间片为4ms, 用户设置的milli_sec参数值为1, 则实际的 超时时间在1ms到5ms范围内。在CPU负载高场景下, 超时时间 仍可能存在波动。
返回值说明	ret: int, 错误码。 ● 返回0表示成功。 ● 返回非0表示失败, 参见2.10.9.2 VPC图像处理返回码。 task_id: 任务分配的ID, 用来区分任务。

约束说明

版本	约束
Atlas 200I/500 A2推理产品	- 输入、输出图片分辨率的取值范围为：10 * 6 ~ 4096 * 8192，且输入、输出图片的分辨率需保持一致。 - 支持如下输入图片格式（输出图片格式与输入保持一致）： HI_PIXEL_FORMAT_YUV_400 = 0, // YUV400 8bit HI_PIXEL_FORMAT_YUV_SEMIPLANAR_420 = 1, // YUV420SP NV12 8bit HI_PIXEL_FORMAT_YVU_SEMIPLANAR_420 = 2, // YUV420SP NV21 8bit HI_PIXEL_FORMAT_YUV_SEMIPLANAR_422 = 3, // YUV422SP 8bit HI_PIXEL_FORMAT_YVU_SEMIPLANAR_422 = 4, // YVU422SP 8bit HI_PIXEL_FORMAT_YUYV_PACKED_422 = 7, // YUV422Packed YUYV 8bit HI_PIXEL_FORMAT_YVYU_PACKED_422 = 9, // YUV422Packed YVYU 8bit HI_PIXEL_FORMAT_YUV_PACKED_444 = 11, // YUV444 Package 8bit HI_PIXEL_FORMAT_RGB_888 = 12, // RGB888 HI_PIXEL_FORMAT_BGR_888 = 13, // BGR888 HI_PIXEL_FORMAT_YVU_PACKED_444 = 58, // YVU444 Package 8bit HI_PIXEL_FORMAT_RGB_888_PLANAR = 69, // RGB888 Planar HI_PIXEL_FORMAT_BGR_888_PLANAR = 70, // BGR888 Planar - 实现原理和OpenCV一致，浮点转定点运算部分有差异，精度不能完全对标。 - 由于YUV格式图像下采样约束，当输出图片格式为YUV420SP或YUV422SP格式时，在配置边界填充类型时，建议设置为“边界复制模式”，避免输出图片边缘异常数据。

版本	约束
Atlas A2训练系列产品/ Atlas 800I A2推理产品	- 输入、输出图片分辨率的取值范围为：10 * 6 ~ 4096 * 8192，且输入、输出图片的分辨率需保持一致。 - 支持如下输入图片格式（输出图片格式与输入保持一致）： HI_PIXEL_FORMAT_YUV_400 = 0, // YUV400 8bit HI_PIXEL_FORMAT_YUV_SEMIPLANAR_420 = 1, // YUV420SP NV12 8bit HI_PIXEL_FORMAT_YVU_SEMIPLANAR_420 = 2, // YUV420SP NV21 8bit HI_PIXEL_FORMAT_YUV_SEMIPLANAR_422 = 3, // YUV422SP 8bit HI_PIXEL_FORMAT_YVU_SEMIPLANAR_422 = 4, // YVU422SP 8bit HI_PIXEL_FORMAT_YUYV_PACKED_422 = 7, // YUV422Packed YUYV 8bit HI_PIXEL_FORMAT_YVYU_PACKED_422 = 9, // YUV422Packed YVYU 8bit HI_PIXEL_FORMAT_YUV_PACKED_444 = 11, // YUV444 Package 8bit HI_PIXEL_FORMAT_RGB_888 = 12, // RGB888 HI_PIXEL_FORMAT_BGR_888 = 13, // BGR888 HI_PIXEL_FORMAT_YVU_PACKED_444 = 58, // YVU444 Package 8bit HI_PIXEL_FORMAT_RGB_888_PLANAR = 69, // RGB888 Planar HI_PIXEL_FORMAT_BGR_888_PLANAR = 70, // BGR888 Planar HI_PIXEL_FORMAT_YUV400_FP32 = 138, // YUV400 Package 每个像素用1个float32数据表示 HI_PIXEL_FORMAT_RGB_888_PLANAR_FP32 = 1006, // RGB888 Planar 每个像素用1个float32数据表示 HI_PIXEL_FORMAT_BGR_888_PLANAR_FP32 = 1007, // BGR888 Planar 每个像素用1个float32数据表示 HI_PIXEL_FORMAT_RGB_888_FP32 = 1012, // RGB888 Package, 每个像素用1个float32数据表示 HI_PIXEL_FORMAT_BGR_888_FP32 = 1013, // BGR888 Package, 每个像素用1个float32数据表示 - 由于YUV格式图像下采样约束，当输出图片格式为YUV420SP或YUV422SP格式时，在配置边界填充类型时，建议设置为“边界复制模式”，避免输出图片边缘异常数据。

2.10.4.24 acl.himpi.vpc_crop_resize_resize_paste

c函数原型	hi_s32 hi_mpi_vpc_crop_resize_resize_paste(hi_vpc_chn chn, const hi_vpc_pic_info *source_pic, hi_vpc_crop_resize_resize_paste_region crop_resize_resize_paste_info[], hi_u32 count, hi_u32 *task_id, hi_s32 milli_sec)
Python函数	task_id, ret = acl.himpi.vpc_crop_resize_resize_paste(chn, source_pic, crop_resize_resize_paste_info, count, milli_sec)

函数功能	从输入图片中抠图，对抠出的子图进行两次缩放后（两次缩放目的是对敏感图像信息模糊脱敏），贴在用户输出图片的指定区域。异步接口。 Atlas 200/300/500 推理产品，不支持该接口。 Atlas 200I/500 A2推理产品，不支持该接口。 Atlas 训练系列产品，不支持该接口。 Atlas 推理系列产品，不支持该接口。 Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。
输入说明	chn: int，图片处理通道号。 source_pic: dict，原始图片信息字典，请参见2.10.8.5.1 hi_vpc_pic_info。 输入图片分辨率在10 * 6 ~ 4096 * 4096（包括4096）范围内时，支持图片格式、宽高对齐、内存约束处说明的输入图片格式。 crop_resize_resize_paste_info: list，抠图缩放并贴图的图片信息列表，该列表长度与“count”参数值保持一致。元素类型参见2.10.8.5.15 hi_vpc_crop_resize_resize_paste_region。 count: int，抠图缩放贴图的图片数量，取值范围[1, 256]。 输入图片分辨率在10 * 6 ~ 4096 * 4096（包括4096）范围内时，支持图片格式、宽高对齐、内存约束处说明的输入图片格式。 milli_sec: int，超时时间配置，单位是毫秒，取值范围如下。 “-1”：阻塞方式。 “0”：非阻塞方式。 >0：超时方式，配置具体的超时时间。超时时间受操作系统影响，一般偏差在操作系统的一个时间片内，例如，操作系统的—个时间片为4ms，用户设置的milli_sec参数值为1，则实际的超时时间在1ms到5ms范围内。在CPU负载高场景下，超时时间仍可能存在波动。
返回值说明	ret: int，错误码。 - 返回0表示成功。 - 返回非0表示失败，参见2.10.9.2 VPC图像处理返回码。 task_id: 任务分配的ID，用来区分任务。

约束说明

- 输出图片的数量最多等于抠图区域的数量，最少是一张图片。也就是说，如果从输入图片中扣n张子图，那么贴图的目标图片可以是同一张大图，也可以是多张大图。取决于“hi_vpc_crop_resize_resize_paste_region”字典中指定的地址。
- 当输入、输出图片格式都为ARGB8888、ABGR8888、RGBA8888、BGRA8888格式时，在经过本接口缩放处理后，A分量值会变为0。
- VPC功能的约束请参见[约束说明](#)。

2.10.4.25 acl.himpi.vpc_dilate

c函数原型	hi_s32 hi_mpi_vpc_dilate(hi_vpc_chn chn, const hi_blur_param* dilate_param, hi_u32 *task_id, hi_s32 milli_sec)
Python函数	task_id, ret = acl.himpi.vpc_dilate(chn, dilate_param, milli_sec)
函数功能	对输入图片做膨胀处理。 Atlas 200/300/500 推理产品 ，不支持该接口。 Atlas 训练系列产品 ，不支持该接口。 Atlas 推理系列产品 ，不支持该接口。
输入说明	chn : int, 图片处理通道号。 • Atlas A2训练系列产品/Atlas 800I A2推理产品，该参数的取值范围：[0, 256)，通道总数最多256。 • Atlas 200I/500 A2推理产品，该参数的取值范围：[0, 128)，通道总数最多128。 dilate_param : dict, 膨胀信息字典，具体请参见 2.10.8.5.28 hi_blur_param ，相关约束请参见 约束说明 。 milli_sec : int, 超时时间配置，单位是毫秒，取值范围如下。 • “-1”：阻塞方式。 • “0”：非阻塞方式。 • >0：超时方式，配置具体的超时时间。超时时间受操作系统影响，一般偏差在操作系统的一个时间片内，例如，操作系统的-一个时间片为4ms，用户设置的milli_sec参数值为1，则实际的超时时间在1ms到5ms范围内。在CPU负载高场景下，超时时间仍可能存在波动。
返回值说明	ret : int, 错误码。 • 返回0表示成功。 • 返回非0表示失败，参见2.10.9.2 VPC图像处理返回码。 task_id : 任务分配的ID，用来区分任务。

约束说明

- 输入、输出图片分辨率的取值范围为：10 * 6 ~ 4096 * 8192，且输入、输出图片的分辨率需保持一致。
- 支持如下输入图片格式（输出图片格式与输入保持一致）：

```
HI_PIXEL_FORMAT_YUV_400 = 0,           // YUV400 8bit
HI_PIXEL_FORMAT_YUV_SEMIPLANAR_420 = 1, // YUV420SP NV12 8bit
HI_PIXEL_FORMAT_YVU_SEMIPLANAR_420 = 2, // YUV420SP NV21 8bit
HI_PIXEL_FORMAT_YUV_SEMIPLANAR_422 = 3, // YUV422SP 8bit
HI_PIXEL_FORMAT_YVU_SEMIPLANAR_422 = 4, // YVU422SP 8bit
HI_PIXEL_FORMAT_YUYV_PACKED_422 = 7,    // YUV422Packed YUYV 8bit
HI_PIXEL_FORMAT_YVYU_PACKED_422 = 9,    // YUV422Packed YVYU 8bit
HI_PIXEL_FORMAT_YUV_PACKED_444 = 11,    // YUV444 Package 8bit
HI_PIXEL_FORMAT_RGB_888 = 12,           // RGB888
HI_PIXEL_FORMAT_BGR_888 = 13,           // BGR888
HI_PIXEL_FORMAT_YVU_PACKED_444 = 58,    // YVU444 Package 8bit
```

```
HI_PIXEL_FORMAT_RGB_888_PLANAR = 69,    // RGB888 Planar
HI_PIXEL_FORMAT_BGR_888_PLANAR = 70,    // BGR888 Planar
```

- 由于YUV格式图像下采样约束，当输出图片格式为YUV420SP或YUV422SP格式时，在配置边界填充类型时，建议设置为“边界复制模式”，避免输出图片边缘异常数据。

2.10.4.26 acl.himpi.vpc_draw_cover

c函数原型	<code>hi_s32 hi_mpi_vpc_draw_cover(hi_vpc_chn chn, const hi_cover_param* cover_param, hi_u32 *task_id, hi_s32 milli_sec)</code>
Python函数	<code>task_id, ret = acl.himpi.vpc_draw_cover(chn, cover_param, milli_sec)</code>
函数功能	对输入图片的部分区域做覆盖处理。 Atlas 200/300/500 推理产品 ，不支持该接口。 Atlas 训练系列产品 ，不支持该接口。 Atlas A2训练系列产品/Atlas 800I A2推理产品 ，不支持该接口。 Atlas 推理系列产品 ，不支持该接口。
输入说明	chn : int, 图片处理通道号。 Atlas 200I/500 A2推理产品 ，该参数的取值范围：[0, 128)，通道总数最多128。 cover_param : dict, 覆盖信息字典，具体请参见 2.10.8.5.38 hi_cover_param 。 milli_sec : int, 超时时间配置，单位是毫秒，取值范围如下。 • “-1”：阻塞方式。 • “0”：非阻塞方式。 • >0：超时方式，配置具体的超时时间。超时时间受操作系统影响，一般偏差在操作系统的—个时间片内，例如，操作系统的—个时间片为4ms，用户设置的milli_sec参数值为1，则实际的超时时间在1ms到5ms范围内。在CPU负载高场景下，超时时间仍可能存在波动。
返回值说明	ret : int, 错误码。 • 返回0表示成功。 • 返回非0表示失败，参见2.10.9.2 VPC图像处理返回码。 task_id : 任务分配的ID，用来区分任务。

约束说明

- 输入、输出图片分辨率的取值范围为：10 * 6 ~ 4096 * 8192，且输入、输出图片的分辨率需保持一致。
- 支持如下输入图片格式（输出图片格式与输入保持一致）：

```
HI_PIXEL_FORMAT_YUV_400 = 0,          // YUV400 8bit
HI_PIXEL_FORMAT_YUV_SEMIPLANAR_420 = 1, // YUV420SP NV12 8bit
HI_PIXEL_FORMAT_YVU_SEMIPLANAR_420 = 2, // YUV420SP NV21 8bit
HI_PIXEL_FORMAT_YUV_SEMIPLANAR_422 = 3, // YUV422SP 8bit
```

```
HI_PIXEL_FORMAT_YVU_SEMIPLANAR_422 = 4, // YVU422SP 8bit
HI_PIXEL_FORMAT_YUYV_PACKED_422 = 7, // YUV422Packed YUYV 8bit
HI_PIXEL_FORMAT_YVYU_PACKED_422 = 9, // YUV422Packed YVYU 8bit
HI_PIXEL_FORMAT_YUV_PACKED_444 = 11, // YUV444 Package 8bit
HI_PIXEL_FORMAT_RGB_888 = 12, // RGB888
HI_PIXEL_FORMAT_BGR_888 = 13, // BGR888
HI_PIXEL_FORMAT_YVU_PACKED_444 = 58, // YVU444 Package 8bit
HI_PIXEL_FORMAT_RGB_888_PLANAR = 69, // RGB888 Planar
HI_PIXEL_FORMAT_BGR_888_PLANAR = 70, // BGR888 Planar
```

2.10.4.27 acl.himpi.vpc_draw_line

c函数原型	hi_s32 hi_mpi_vpc_draw_line(hi_vpc_chn chn, const hi_line_param* cover_param, hi_u32 *task_id, hi_s32 milli_sec)
Python函数	task_id, ret = acl.himpi.vpc_draw_line(chn, cover_param, milli_sec)
函数功能	对输入图片做画线处理。 Atlas 200/300/500 推理产品 ，不支持该接口。 Atlas 训练系列产品 ，不支持该接口。 Atlas A2训练系列产品/Atlas 800I A2推理产品 ，不支持该接口。 Atlas 推理系列产品 ，不支持该接口。
输入说明	chn : int, 图片处理通道号。 Atlas 200I/500 A2推理产品 ，该参数的取值范围：[0, 128)，通道总数最多128。 line_param : dict, 画线信息字典，具体请参见 2.10.8.5.41 hi_line_param 。 milli_sec : int, 超时时间配置，单位是毫秒，取值范围如下。 “-1”：阻塞方式。 “0”：非阻塞方式。 >0: 超时方式，配置具体的超时时间。超时时间受操作系统影响，一般偏差在操作系统的一个时间片内，例如，操作系统的-一个时间片为4ms，用户设置的milli_sec参数值为1，则实际的超时时间在1ms到5ms范围内。在CPU负载高场景下，超时时间仍可能存在波动。
返回值说明	ret : int, 错误码。 - 返回0表示成功。 - 返回非0表示失败，参见2.10.9.2 VPC图像处理返回码。 task_id : 任务分配的ID，用来区分任务。

约束说明

- 输入、输出图片分辨率的取值范围为：10 * 6 ~ 4096 * 8192，且输入、输出图片的分辨率需保持一致。
- 支持如下输入图片格式（输出图片格式与输入保持一致）：

```
HI_PIXEL_FORMAT_YUV_400 = 0, // YUV400 8bit
HI_PIXEL_FORMAT_YUV_SEMIPLANAR_420 = 1, // YUV420SP NV12 8bit
```



```
HI_PIXEL_FORMAT_YUV_SEMIPLANAR_420 = 2, // YUV420SP NV21 8bit
HI_PIXEL_FORMAT_YUV_SEMIPLANAR_422 = 3, // YUV422SP 8bit
HI_PIXEL_FORMAT_YUV_SEMIPLANAR_422 = 4, // YVU422SP 8bit
HI_PIXEL_FORMAT_YUYV_PACKED_422 = 7, // YUV422Packed YUYV 8bit
HI_PIXEL_FORMAT_VYU_PACKED_422 = 9, // YUV422Packed VYU 8bit
HI_PIXEL_FORMAT_YUV_PACKED_444 = 11, // YUV444 Package 8bit
HI_PIXEL_FORMAT_RGB_888 = 12, // RGB888
HI_PIXEL_FORMAT_BGR_888 = 13, // BGR888
HI_PIXEL_FORMAT_YUV_PACKED_444 = 58, // YVU444 Package 8bit
HI_PIXEL_FORMAT_RGB_888_PLANAR = 69, // RGB888 Planar
HI_PIXEL_FORMAT_BGR_888_PLANAR = 70, // BGR888 Planar
```

2.10.4.28 acl.himpi.vpc_draw_mosaic

c函数原型	hi_s32 hi_mpi_vpc_draw_mosaic(hi_vpc_chn chn, const hi_mosaic_param* mosaic_param, hi_u32 *task_id, hi_s32 milli_sec)
Python函数	task_id, ret = acl.himpi.vpc_draw_mosaic(chn, mosaic_param, milli_sec)
函数功能	对输入图片的部分区域做马赛克处理。 Atlas 200/300/500 推理产品 ，不支持该接口。 Atlas 训练系列产品 ，不支持该接口。 Atlas A2训练系列产品/Atlas 800I A2推理产品 ，不支持该接口。 Atlas 推理系列产品 ，不支持该接口。
输入说明	chn : int, 图片处理通道号。 Atlas 200I/500 A2推理产品 ，该参数的取值范围: [0, 128)，通道总数最多128。 mosaic_param : dict, 马赛克信息，具体请参见 2.10.8.5.34 hi_mosaic_param 。 milli_sec : int, 超时时间配置，单位是毫秒，取值范围如下。 “-1”：阻塞方式。 “0”：非阻塞方式。 >0: 超时方式，配置具体的超时时间。超时时间受操作系统影响，一般偏差在操作系统的一个时间片内，例如，操作系统的-一个时间片为4ms，用户设置的milli_sec参数值为1，则实际的超时时间在1ms到5ms范围内。在CPU负载高场景下，超时时间仍可能存在波动。
返回值说明	ret : int, 错误码。 - 返回0表示成功。 - 返回非0表示失败，参见2.10.9.2 VPC图像处理返回码。 task_id : 任务分配的ID，用来区分任务。

约束说明

- 输入、输出图片分辨率的取值范围为：10 * 6 ~ 4096 * 8192，且输入、输出图片的分辨率需保持一致。

- 支持如下输入图片格式（输出图片格式与输入保持一致）：

```
HI_PIXEL_FORMAT_YUV_400 = 0, // YUV400 8bit
HI_PIXEL_FORMAT_YUV_SEMIPLANAR_420 = 1, // YUV420SP NV12 8bit
HI_PIXEL_FORMAT_YVU_SEMIPLANAR_420 = 2, // YUV420SP NV21 8bit
HI_PIXEL_FORMAT_YUV_SEMIPLANAR_422 = 3, // YUV422SP 8bit
HI_PIXEL_FORMAT_YVU_SEMIPLANAR_422 = 4, // YVU422SP 8bit
HI_PIXEL_FORMAT_YUYV_PACKED_422 = 7, // YUV422Packed YUYV 8bit
HI_PIXEL_FORMAT_YVYU_PACKED_422 = 9, // YUV422Packed YVYU 8bit
HI_PIXEL_FORMAT_YUV_PACKED_444 = 11, // YUV444 Package 8bit
HI_PIXEL_FORMAT_RGB_888 = 12, // RGB888
HI_PIXEL_FORMAT_BGR_888 = 13, // BGR888
HI_PIXEL_FORMAT_YVU_PACKED_444 = 58, // YVU444 Package 8bit
HI_PIXEL_FORMAT_RGB_888_PLANAR = 69, // RGB888 Planar
HI_PIXEL_FORMAT_BGR_888_PLANAR = 70, // BGR888 Planar
```

2.10.4.29 acl.himpi.vpc_draw_osd

c函数原型	<code>hi_s32 hi_mpi_vpc_draw_osd(hi_vpc_chn chn, const hi_osd_param* osd_param, hi_u32 *task_id, hi_s32 milli_sec)</code>
Python函数	<code>task_id, ret = acl.himpi.vpc_draw_osd(chn, osd_param, task_id, milli_sec)</code>
函数功能	对输入图片做叠加处理，可用于添加水印的场景。 Atlas 200/300/500 推理产品 ，不支持该接口。 Atlas 训练系列产品 ，不支持该接口。 Atlas A2训练系列产品/Atlas 800I A2推理产品 ，不支持该接口。 Atlas 推理系列产品 ，不支持该接口。
输入说明	chn : int，图片处理通道号。 Atlas 200I/500 A2推理产品 ，该参数的取值范围：[0, 128)，通道总数最多128。 osd_param : dict，叠加信息字典，具体请参见 2.10.8.5.43 hi_osd_param 。 milli_sec : int，超时时间配置，单位是毫秒，取值范围如下。 “-1”：阻塞方式。 “0”：非阻塞方式。 >0：超时方式，配置具体的超时时间。超时时间受操作系统影响，一般偏差在操作系统的—个时间片内，例如，操作系统的—个时间片为4ms，用户设置的milli_sec参数值为1，则实际的超时时间在1ms到5ms范围内。在CPU负载高场景下，超时时间仍可能存在波动。
返回值说明	ret : int，错误码。 - 返回0表示成功。 - 返回非0表示失败，参见2.10.9.2 VPC图像处理返回码。 task_id : 任务分配的ID，用来区分任务。

约束说明

- 输入、输出图片分辨率的取值范围为：10 * 6 ~ 4096 * 8192，且输入、输出图片的分辨率需保持一致。

- 支持如下输入图片格式（输出图片格式与输入保持一致）：

```
HI_PIXEL_FORMAT_YUV_400 = 0,           // YUV400 8bit
HI_PIXEL_FORMAT_YUV_SEMIPLANAR_420 = 1, // YUV420SP NV12 8bit
HI_PIXEL_FORMAT_YVU_SEMIPLANAR_420 = 2, // YUV420SP NV21 8bit
HI_PIXEL_FORMAT_YUV_SEMIPLANAR_422 = 3, // YUV422SP 8bit
HI_PIXEL_FORMAT_YVU_SEMIPLANAR_422 = 4, // YVU422SP 8bit
HI_PIXEL_FORMAT_YUYV_PACKED_422 = 7,    // YUV422Packed YUYV 8bit
HI_PIXEL_FORMAT_YVYU_PACKED_422 = 9,    // YUV422Packed YVYU 8bit
HI_PIXEL_FORMAT_YUV_PACKED_444 = 11,    // YUV444 Package 8bit
HI_PIXEL_FORMAT_RGB_888 = 12,           // RGB888
HI_PIXEL_FORMAT_BGR_888 = 13,           // BGR888
HI_PIXEL_FORMAT_YVU_PACKED_444 = 58,    // YVU444 Package 8bit
HI_PIXEL_FORMAT_RGB_888_PLANAR = 69,    // RGB888 Planar
HI_PIXEL_FORMAT_BGR_888_PLANAR = 70,    // BGR888 Planar
```

2.10.4.30 acl.himpi.vpc_erode

c函数原型	hi_s32 hi_mpi_vpc_erode(hi_vpc_chn chn, const hi_blur_param* erode_param, hi_u32 *task_id, hi_s32 milli_sec)
Python函数	task_id, ret = acl.himpi.vpc_erode(chn, erode_param, milli_sec)
函数功能	对输入图片做腐蚀处理。 Atlas 200/300/500 推理产品 ，不支持该接口。 Atlas 训练系列产品 ，不支持该接口。 Atlas 推理系列产品 ，不支持该接口。
输入说明	chn : int, 图片处理通道号。 Atlas A2训练系列产品/Atlas 800I A2推理产品，该参数的取值范围：[0, 256)，通道总数最多256。 Atlas 200I/500 A2推理产品，该参数的取值范围：[0, 128)，通道总数最多128。 erode_param : dict, 腐蚀信息字典，具体请参见 2.10.8.5.28 hi_blur_param 。 milli_sec : int, 超时时间配置，单位是毫秒，取值范围如下。 “-1”：阻塞方式。 “0”：非阻塞方式。 >0: 超时方式，配置具体的超时时间。超时时间受操作系统影响，一般偏差在操作系统的 一个时间片内，例如，操作系统的 一个时间片为4ms，用户设置的milli_sec参数值为1，则实际的超时时间在1ms到5ms范围内。在CPU负载高场景下，超时时间仍可能存在波动。
返回值说明	ret : int, 错误码。 - 返回0表示成功。 - 返回非0表示失败，参见2.10.9.2 VPC图像处理返回码。 task_id : 任务分配的ID，用来区分任务。

约束说明

- 输入、输出图片分辨率的取值范围为：10 *6 ~ 4096 * 8192，且输入、输出图片的分辨率需保持一致。
- 支持如下输入图片格式（输出图片格式与输入保持一致）：

```
HI_PIXEL_FORMAT_YUV_400 = 0,           // YUV400 8bit
HI_PIXEL_FORMAT_YUV_SEMIPLANAR_420 = 1, // YUV420SP NV12 8bit
HI_PIXEL_FORMAT_YVU_SEMIPLANAR_420 = 2, // YUV420SP NV21 8bit
HI_PIXEL_FORMAT_YUV_SEMIPLANAR_422 = 3, // YUV422SP 8bit
HI_PIXEL_FORMAT_YVU_SEMIPLANAR_422 = 4, // YVU422SP 8bit
HI_PIXEL_FORMAT_YUYV_PACKED_422 = 7,   // YUV422Packed YUYV 8bit
HI_PIXEL_FORMAT_YVYU_PACKED_422 = 9,   // YUV422Packed YVYU 8bit
HI_PIXEL_FORMAT_YUV_PACKED_444 = 11,   // YUV444 Package 8bit
HI_PIXEL_FORMAT_RGB_888 = 12,          // RGB888
HI_PIXEL_FORMAT_BGR_888 = 13,          // BGR888
HI_PIXEL_FORMAT_YVU_PACKED_444 = 58,   // YVU444 Package 8bit
HI_PIXEL_FORMAT_RGB_888_PLANAR = 69,   // RGB888 Planar
HI_PIXEL_FORMAT_BGR_888_PLANAR = 70,   // BGR888 Planar
```
- 由于YUV格式图像下采样约束，当输出图片格式为YUV420SP或YUV422SP格式时，在配置边界填充类型时，建议设置为“边界复制模式”，避免输出图片边缘异常数据。

2.10.4.31 acl.himpi.vpc_filter2d

c函数原型	hi_s32 hi_mpi_vpc_filter2d(hi_vpc_chn chn, const hi_filter_2d_param* filter_2d_param, hi_u32 *task_id, hi_s32 milli_sec)
Python函数	task_id, ret = acl.himpi.vpc_filter2d(chn, filter_2d_param, milli_sec)
函数功能	对输入图片做2D卷积滤波处理。 Atlas 200/300/500 推理产品，不支持该接口。 Atlas 训练系列产品，不支持该接口。 Atlas 推理系列产品，不支持该接口。
输入说明	chn: int，图片处理通道号。 • Atlas A2训练系列产品/Atlas 800I A2推理产品，该参数的取值范围：[0, 256)，通道总数最多256。 • Atlas 200I/500 A2推理产品，该参数的取值范围：[0, 128)，通道总数最多128。 filter_2d_param: dict，2D卷积滤波信息字典，具体请参见2.10.8.5.32 hi_filter_2d_param。 milli_sec: int，超时时间配置，单位是毫秒，取值范围如下。 • “-1”：阻塞方式。 • “0”：非阻塞方式。 • >0：超时方式，配置具体的超时时间。超时时间受操作系统影响，一般偏差在操作系统的一个时间片内，例如，操作系统的 一个时间片为4ms，用户设置的milli_sec参数值为1，则实际的超时时间在1ms到5ms范围内。在CPU负载高场景下，超时时间仍可能存在波动。

返回值说明	ret: int，错误码。 • 返回0表示成功。 • 返回非0表示失败，参见2.10.9.2 VPC图像处理返回码。 task_id: 任务分配的ID，用来区分任务。

约束说明

版本	约束
Atlas 200I/500 A2推理产品	• 输入、输出图片分辨率的取值范围为：10 * 6 ~ 4096 * 8192，且输入、输出图片的分辨率需保持一致。 • 支持如下输入图片格式（输出图片格式与输入保持一致）： <pre>HI_PIXEL_FORMAT_YUV_400 = 0, // YUV400 8bit HI_PIXEL_FORMAT_YUV_SEMIPLANAR_420 = 1, // YUV420SP NV12 8bit HI_PIXEL_FORMAT_YVU_SEMIPLANAR_420 = 2, // YUV420SP NV21 8bit HI_PIXEL_FORMAT_YUV_SEMIPLANAR_422 = 3, // YUV422SP 8bit HI_PIXEL_FORMAT_YVU_SEMIPLANAR_422 = 4, // YVU422SP 8bit HI_PIXEL_FORMAT_YUYV_PACKED_422 = 7, // YUV422Packed YUYV 8bit HI_PIXEL_FORMAT_YVYU_PACKED_422 = 9, // YUV422Packed YVYU 8bit HI_PIXEL_FORMAT_YUV_PACKED_444 = 11, // YUV444 Package 8bit HI_PIXEL_FORMAT_RGB_888 = 12, // RGB888 HI_PIXEL_FORMAT_BGR_888 = 13, // BGR888 HI_PIXEL_FORMAT_YVU_PACKED_444 = 58, // YVU444 Package 8bit HI_PIXEL_FORMAT_RGB_888_PLANAR = 69, // RGB888 Planar HI_PIXEL_FORMAT_BGR_888_PLANAR = 70, // BGR888 Planar</pre> • 实现原理和OpenCV一致，浮点转定点运算部分有差异，精度不能完全对标。 • 由于YUV格式图像下采样约束，当输出图片格式为YUV420SP或YUV422SP格式时，在配置边界填充类型时，建议设置为“边界复制模式”，避免输出图片边缘异常数据。

版本	约束
Atlas A2训练系列产品/ Atlas 800I A2推理产品	- 输入、输出图片分辨率的取值范围为：10 * 6 ~ 4096 * 8192，且输入、输出图片的分辨率需保持一致。 - 支持如下输入图片格式（输出图片格式与输入保持一致）： HI_PIXEL_FORMAT_YUV_400 = 0, // YUV400 8bit HI_PIXEL_FORMAT_YUV_SEMIPLANAR_420 = 1, // YUV420SP NV12 8bit HI_PIXEL_FORMAT_YVU_SEMIPLANAR_420 = 2, // YUV420SP NV21 8bit HI_PIXEL_FORMAT_YUV_SEMIPLANAR_422 = 3, // YUV422SP 8bit HI_PIXEL_FORMAT_YVU_SEMIPLANAR_422 = 4, // YVU422SP 8bit HI_PIXEL_FORMAT_YUYV_PACKED_422 = 7, // YUV422Packed YUYV 8bit HI_PIXEL_FORMAT_YVYU_PACKED_422 = 9, // YUV422Packed YVYU 8bit HI_PIXEL_FORMAT_YUV_PACKED_444 = 11, // YUV444 Package 8bit HI_PIXEL_FORMAT_RGB_888 = 12, // RGB888 HI_PIXEL_FORMAT_BGR_888 = 13, // BGR888 HI_PIXEL_FORMAT_YVU_PACKED_444 = 58, // YVU444 Package 8bit HI_PIXEL_FORMAT_RGB_888_PLANAR = 69, // RGB888 Planar HI_PIXEL_FORMAT_BGR_888_PLANAR = 70, // BGR888 Planar HI_PIXEL_FORMAT_YUV400_FP32 = 138, // YUV400 Package 每个像素用1个float32数据表示 HI_PIXEL_FORMAT_RGB_888_PLANAR_FP32 = 1006, // RGB888 Planar 每个像素用1个float32数据表示 HI_PIXEL_FORMAT_BGR_888_PLANAR_FP32 = 1007, // BGR888 Planar 每个像素用1个float32数据表示 HI_PIXEL_FORMAT_RGB_888_FP32 = 1012, // RGB888 Package, 每个像素用1个float32数据表示 HI_PIXEL_FORMAT_BGR_888_FP32 = 1013, // BGR888 Package, 每个像素用1个float32数据表示 - 由于YUV格式图像下采样约束，当输出图片格式为YUV420SP或YUV422SP格式时，在配置边界填充类型时，建议设置为“边界复制模式”，避免输出图片边缘异常数据。

2.10.4.32 acl.himpi.vpc_flip

c函数原型	hi_s32 hi_mpi_vpc_flip(hi_vpc_chn chn, hi_flip_param* flip_param, hi_u32 *task_id, hi_s32 milli_sec)
Python函数	task_id, ret = acl.himpi.vpc_flip(chn, flip_param, milli_sec)
函数功能	对输入图片做翻转处理，支持水平方向翻转、竖直方向翻转和水平竖直方向同时翻转。 Atlas 200/300/500 推理产品 ，不支持该接口。 Atlas 训练系列产品 ，不支持该接口。 Atlas 推理系列产品 ，不支持该接口。

输入说明	chn: int, 图片处理通道号。 ● Atlas A2训练系列产品/Atlas 800I A2推理产品, 该参数的取值范围: [0, 256), 通道总数最多256。 ● Atlas 200I/500 A2推理产品, 该参数的取值范围: [0, 128), 通道总数最多128。 flip_param: dict, 翻转信息字典, 具体请参见2.10.8.5.48 hi_flip_param。 milli_sec: int, 超时时间配置, 单位是毫秒, 取值范围如下。 ● “-1”: 阻塞方式。 ● “0”: 非阻塞方式。 ● >0: 超时方式, 配置具体的超时时间。超时时间受操作系统影响, 一般偏差在操作系统的一个时间片内, 例如, 操作系统的 一个时间片为4ms, 用户设置的milli_sec参数值为1, 则实际的 超时时间在1ms到5ms范围内。在CPU负载高场景下, 超时时间 仍可能存在波动。
返回值说明	ret: int, 错误码。 ● 返回0表示成功。 ● 返回非0表示失败, 参见2.10.9.2 VPC图像处理返回码。 task_id: 任务分配的ID, 用来区分任务。

约束说明

- 输入、输出图片分辨率的取值范围为: 10 * 6 ~ 4096 * 8192, 且输入、输出图片的分辨率需保持一致。
- 支持如下输入图片格式: (输出格式需与输入保持一致)

```
HI_PIXEL_FORMAT_YUV_400 = 0,           // YUV400 8bit
HI_PIXEL_FORMAT_YUV_SEMIPLANAR_420 = 1, // YUV420SP NV12 8bit
HI_PIXEL_FORMAT_YVU_SEMIPLANAR_420 = 2, // YUV420SP NV21 8bit
HI_PIXEL_FORMAT_YUV_SEMIPLANAR_422 = 3, // YUV422SP 8bit
HI_PIXEL_FORMAT_YVU_SEMIPLANAR_422 = 4, // YVU422SP 8bit
HI_PIXEL_FORMAT_YUYV_PACKED_422 = 7,    // YUV422Packed YUYV 8bit
HI_PIXEL_FORMAT_UYVY_PACKED_422 = 8,    // YUV422Packed UYVY 8bit
HI_PIXEL_FORMAT_YVYU_PACKED_422 = 9,    // YUV422Packed YVYU 8bit
HI_PIXEL_FORMAT_VYUY_PACKED_422 = 10,   // YUV422Packed VYUY 8bit
HI_PIXEL_FORMAT_YUV_PACKED_444 = 11,    // YUV444 Package 8bit
HI_PIXEL_FORMAT_RGB_888 = 12,           // RGB888
HI_PIXEL_FORMAT_BGR_888 = 13,           // BGR888
HI_PIXEL_FORMAT_ARGB_8888 = 14,         // ARGB_8888
HI_PIXEL_FORMAT_ABGR_8888 = 15,         // ABGR_8888
HI_PIXEL_FORMAT_RGBA_8888 = 16,         // RGBA_8888
HI_PIXEL_FORMAT_BGRA_8888 = 17,         // BGRA_8888
HI_PIXEL_FORMAT_YVU_PLANAR_444 = 22,   // YVU444 Planar 8bit
HI_PIXEL_FORMAT_YVU_PACKED_444 = 58,    // YVU444 Package 8bit
HI_PIXEL_FORMAT_RGB_888_PLANAR = 69,    // RGB888 Planar
HI_PIXEL_FORMAT_BGR_888_PLANAR = 70,    // BGR888 Planar
```

2.10.4.33 acl.himpi.vpc_gaussian_blur

c函数原型	hi_s32 hi_mpi_vpc_gaussian_blur(hi_vpc_chn chn, const hi_gaussian_blur_param* gaussian_blur_param, hi_u32 *task_id, hi_s32 milli_sec)
Python函数	task_id, ret = acl.himpi.vpc_gaussian_blur(chn, gaussian_blur_param, milli_sec)
函数功能	对输入图片做高斯滤波处理。 Atlas 200/300/500 推理产品，不支持该接口。 Atlas 训练系列产品，不支持该接口。 Atlas 推理系列产品，不支持该接口。
输入说明	chn: int，图片处理通道号。 • Atlas A2训练系列产品/Atlas 800I A2推理产品，该参数的取值范围：[0, 256)，通道总数最多256。 • Atlas 200I/500 A2推理产品，该参数的取值范围：[0, 128)，通道总数最多128。 gaussian_blur_param: dict，高斯滤波信息字典，具体请参见2.10.8.5.30 hi_gaussian_blur_param。 milli_sec: int，超时时间配置，单位是毫秒，取值范围如下。 • “-1”：阻塞方式。 • “0”：非阻塞方式。 • >0：超时方式，配置具体的超时时间。超时时间受操作系统影响，一般偏差在操作系统的一个时间片内，例如，操作系统的—个时间片为4ms，用户设置的milli_sec参数值为1，则实际的超时时间在1ms到5ms范围内。在CPU负载高场景下，超时时间仍可能存在波动。
返回值说明	ret: int，错误码。 • 返回0表示成功。 • 返回非0表示失败，参见2.10.9.2 VPC图像处理返回码。 task_id: 任务分配的ID，用来区分任务。

约束说明

版本	约束
Atlas 200I/500 A2推理产品	- 输入、输出图片分辨率的取值范围为：10 * 6 ~ 4096 * 8192，且输入、输出图片的分辨率需保持一致。 - 支持如下输入图片格式（输出图片格式与输入保持一致）： HI_PIXEL_FORMAT_YUV_400 = 0, // YUV400 8bit HI_PIXEL_FORMAT_YUV_SEMIPLANAR_420 = 1, // YUV420SP NV12 8bit HI_PIXEL_FORMAT_YVU_SEMIPLANAR_420 = 2, // YUV420SP NV21 8bit HI_PIXEL_FORMAT_YUV_SEMIPLANAR_422 = 3, // YUV422SP 8bit HI_PIXEL_FORMAT_YVU_SEMIPLANAR_422 = 4, // YVU422SP 8bit HI_PIXEL_FORMAT_YUYV_PACKED_422 = 7, // YUV422Packed YUYV 8bit HI_PIXEL_FORMAT_YVYU_PACKED_422 = 9, // YUV422Packed YVYU 8bit HI_PIXEL_FORMAT_YUV_PACKED_444 = 11, // YUV444 Package 8bit HI_PIXEL_FORMAT_RGB_888 = 12, // RGB888 HI_PIXEL_FORMAT_BGR_888 = 13, // BGR888 HI_PIXEL_FORMAT_YVU_PACKED_444 = 58, // YVU444 Package 8bit HI_PIXEL_FORMAT_RGB_888_PLANAR = 69, // RGB888 Planar HI_PIXEL_FORMAT_BGR_888_PLANAR = 70, // BGR888 Planar - 实现原理和OpenCV一致，浮点转定点运算部分有差异，精度不能完全对标。 - 由于YUV格式图像下采样约束，当输出图片格式为YUV420SP或YUV422SP格式时，在配置边界填充类型时，建议设置为“边界复制模式”，避免输出图片边缘异常数据。

版本	约束
Atlas A2训练系列产品/ Atlas 800I A2推理产品	- 输入、输出图片分辨率的取值范围为：10 * 6 ~ 4096 * 8192，且输入、输出图片的分辨率需保持一致。 - 支持如下输入图片格式（输出图片格式与输入保持一致）： HI_PIXEL_FORMAT_YUV_400 = 0, // YUV400 8bit HI_PIXEL_FORMAT_YUV_SEMIPLANAR_420 = 1, // YUV420SP NV12 8bit HI_PIXEL_FORMAT_YVU_SEMIPLANAR_420 = 2, // YUV420SP NV21 8bit HI_PIXEL_FORMAT_YUV_SEMIPLANAR_422 = 3, // YUV422SP 8bit HI_PIXEL_FORMAT_YVU_SEMIPLANAR_422 = 4, // YVU422SP 8bit HI_PIXEL_FORMAT_YUYV_PACKED_422 = 7, // YUV422Packed YUYV 8bit HI_PIXEL_FORMAT_YVYU_PACKED_422 = 9, // YUV422Packed YVYU 8bit HI_PIXEL_FORMAT_YUV_PACKED_444 = 11, // YUV444 Package 8bit HI_PIXEL_FORMAT_RGB_888 = 12, // RGB888 HI_PIXEL_FORMAT_BGR_888 = 13, // BGR888 HI_PIXEL_FORMAT_YVU_PACKED_444 = 58, // YVU444 Package 8bit HI_PIXEL_FORMAT_RGB_888_PLANAR = 69, // RGB888 Planar 每个像素用1个float32数据表示 HI_PIXEL_FORMAT_BGR_888_PLANAR = 70, // BGR888 Planar 每个像素用1个float32数据表示 HI_PIXEL_FORMAT_YUV400_FP32 = 138, // YUV400 Package 每个像素用1个float32数据表示 HI_PIXEL_FORMAT_RGB_888_PLANAR_FP32 = 1006, // RGB888 Planar 每个像素用1个float32数据表示 HI_PIXEL_FORMAT_BGR_888_PLANAR_FP32 = 1007, // BGR888 Planar 每个像素用1个float32数据表示 HI_PIXEL_FORMAT_RGB_888_FP32 = 1012, // RGB888 Package, 每个像素用1个float32数据表示 HI_PIXEL_FORMAT_BGR_888_FP32 = 1013, // BGR888 Package, 每个像素用1个float32数据表示 - 由于YUV格式图像下采样约束，当输出图片格式为YUV420SP或YUV422SP格式时，在配置边界填充类型时，建议设置为“边界复制模式”，避免输出图片边缘异常数据。

2.10.4.34 acl.himpi.vpc_get_affine_lut

c函数原型	hi_s32 hi_mpi_vpc_get_affine_lut(hi_point_pair_info *point_pair_info, hi_u32 interpolation, hi_remap_lut *remap_lut)
Python函数	remap_lut, ret = acl.himpi.vpc_get_affine_lut(point_pair_info, interpolation, remap_lut)
函数功能	根据输入、输出图片中的像素位置坐标点，获取仿射变换像素位置重映射表。 Atlas 200/300/500 推理产品，不支持该接口。 Atlas 训练系列产品，不支持该接口。 Atlas 推理系列产品，不支持该接口。

输入说明	point_pair_info: dict, 包含三对输入、输出图片中的像素点坐标信息的字典, 每对中分别包含一个输入图片像素点坐标、一个输出图片像素点坐标, 具体请参见2.10.8.5.19 hi_point_pair_info。 interpolation: int, 缩放算法。支持如下缩放算法: 0: 业界通用的Bilinear算法 (与OpenCV算法的计算过程类似)。 1: 业界通用的Nearest neighbor 算法 (与OpenCV算法的计算过程类似)。 remap_lut: dict, 仿射变换像素位置重映射表信息字典, 具体请参见2.10.8.5.18 hi_remap_lut。字典中的lut属性需由用户提前申请内存, 内存大小“lut_size”属性可调用acl.himpi.vpc_get_lut_mem_size接口获取。
返回值说明	remap_lut: dict, 仿射变换像素位置重映射表信息字典。在成功调用本接口后, 用户可通过lut属性中的地址获取仿射变换LUT表信息。 ret: int, 错误码。 - 返回0表示成功。 - 返回非0表示失败, 参见2.10.9.2 VPC图像处理返回码。

2.10.4.35 acl.himpi.vpc_get_affine_transform

c函数原型	hi_s32 hi_mpi_vpc_get_affine_transform(hi_point_pair_info *point_pair_info, hi_transform_matrix *matrix)
Python函数	matrix, ret = acl.himpi.vpc_get_affine_transform(point_pair_info)
函数功能	获取仿射变换需要的变换矩阵。 Atlas 200/300/500 推理产品, 不支持该接口。 Atlas 训练系列产品, 不支持该接口。 Atlas 推理系列产品, 不支持该接口。
输入说明	point_pair_info: dict, 包含三对输入、输出图片中的像素点坐标信息的字典, 每对中分别包含一个输入图片像素点坐标、一个输出图片像素点坐标, 具体请参见2.10.8.5.19 hi_point_pair_info。
返回值说明	matrix: dict, 仿射变换需要的变换矩阵信息字典, 其中“matrix_type”参数值请参考实际输入矩阵进行配置, 具体参见2.10.8.5.46 hi_transform_matrix。 ret: int, 错误码。 - 返回0表示成功。 - 返回非0表示失败, 参见2.10.9.2 VPC图像处理返回码。

2.10.4.36 acl.himpi.vpc_get_lut_mem_size

c函数原型	hi_s32 hi_mpi_vpc_get_lut_mem_size(hi_u32 width, hi_u32 height, hi_u32 *lut_size)
Python函数	lut_size, ret = acl.himpi.vpc_get_lut_mem_size(width, height)
函数功能	根据输出图片宽高信息获取像素位置重映射表所需的内存大小。 Atlas 200/300/500 推理产品 ，不支持该接口。 Atlas 训练系列产品 ，不支持该接口。 Atlas 推理系列产品 ，不支持该接口。
输入说明	width : int，输出图像宽度。 height : int，输出图像高度。
返回值说明	lut_size : int，像素位置重映射表的内存大小，单位Byte。 ret : int，错误码。 • 返回0表示成功。 • 返回非0表示失败，参见2.10.9.2 VPC图像处理返回码。
约束说明	调用本接口时，width * height的取值范围为：10 * 6 ~ 4096 * 8192。

2.10.4.37 acl.himpi.vpc_get_perspective_lut

c函数原型	hi_s32 hi_mpi_vpc_get_perspective_lut(hi_point_pair_info *point_pair_info, hi_u32 interpolation, hi_remap_lut *remap_lut)
Python函数	remap_lut, ret = acl.himpi.vpc_get_perspective_lut(point_pair_info, interpolation, remap_lut)
函数功能	根据输入、输出图片中的像素位置坐标点，获取透视变换像素位置重映射表。 Atlas 200/300/500 推理产品 ，不支持该接口。 Atlas 训练系列产品 ，不支持该接口。 Atlas 推理系列产品 ，不支持该接口。

输入说明	point_pair_info: dict, 包含四对输入、输出图片中的像素点坐标信息字典, 每对中分别包含一个输入图片像素点坐标、一个输出图片像素点坐标, 具体请参见2.10.8.5.19 hi_point_pair_info。 interpolation: int, 支持如下缩放算法。 0: 业界通用的Bilinear算法 (与OpenCV算法的计算过程类似)。 1: 业界通用的Nearest neighbor 算法 (与OpenCV算法的计算过程类似)。 remap_lut: dict, 透视变换像素位置重映射表字典, 具体请参见2.10.8.5.18 hi_remap_lut。字典中的lut属性需由用户提前申请内存, 内存大小“lut_size”属性可调用acl.himpi.vpc_get_lut_mem_size接口获取。
返回值说明	remap_lut: dict, 透视变换像素位置重映射表字典。在成功调用本接口后, 用户可通过lut属性中的地址获取仿射变换LUT表信息。 ret: int, 错误码。 - 返回0表示成功。 - 返回非0表示失败, 参见2.10.9.2 VPC图像处理返回码。

2.10.4.38 acl.himpi.vpc_get_remap_lut

c函数原型	hi_s32 hi_mpi_vpc_get_remap_lut(hi_map_param *map_param, hi_remap_lut *remap_lut)
Python函数	remap_lut, ret =acl.himpi.vpc_get_remap_lut(map_param, remap_lut)
函数功能	根据用户输入的Remap表 (例如仿射、透视、鱼眼等), 采用业界通用的Bilinear算法 (与OpenCV算法的计算过程类似), 获取系统使用的像素位置重映射表。同步接口。 Atlas 200/300/500 推理产品, 不支持该接口。 Atlas 训练系列产品, 不支持该接口。 Atlas 推理系列产品, 不支持该接口。
输入说明	map_param: dict, 用户原始map表信息字典。请参见2.10.8.5.50 hi_map_param。 remap_lut: dict, 透视变换像素位置重映射表字典, 具体请参见2.10.8.5.18 hi_remap_lut。字典中的lut属性需由用户提前申请内存, 内存大小“lut_size”属性可调用acl.himpi.vpc_get_lut_mem_size接口获取。
返回值说明	remap_lut: dict, 透视变换像素位置重映射表字典。在成功调用本接口后, 用户可通过lut属性中的地址获取仿射变换LUT表信息。 ret: int, 错误码。 - 返回0表示成功。 - 返回非0表示失败, 参见2.10.9.2 VPC图像处理返回码。

2.10.4.39 acl.himpi.vpc_get_rotation_matrix

c函数原型	hi_s32 hi_mpi_vpc_get_rotation_matrix(hi_float_point center_point, hi_double angle, hi_double scale, hi_transform_matrix *matrix)
Python函数	matrix, ret = acl.himpi.vpc_get_rotation_matrix(center_point, angle, scale)
函数功能	获取仿射变换需要的旋转矩阵。 Atlas 200/300/500 推理产品 ，不支持该接口。 Atlas 训练系列产品 ，不支持该接口。 Atlas 推理系列产品 ，不支持该接口。
输入说明	center_point : dict，输入图像的旋转中心点字典，具体请参见 2.10.8.5.20 hi_float_point 。 angle : float，旋转角度，正值表示逆时针旋转。 scale : float，缩放参数，“scale”参数值大于1表示放大图片，参数值小于1表示缩小图片。
返回值说明	matrix : dict，仿射变换需要的变换矩阵信息字典，其中“matrix_type”参数值请参考实际输入矩阵进行配置，具体参见 2.10.8.5.46 hi_transform_matrix 。 ret : int，错误码。 • 返回0表示成功。 • 返回非0表示失败，参见2.10.9.2 VPC图像处理返回码。

2.10.4.40 acl.himpi.vpc_lut_remap

c函数原型	hi_s32 hi_mpi_vpc_lut_remap(hi_vpc_chn chn, hi_warp_transform_param* transform_param, hi_remap_lut *remap_lut, hi_u32 *task_id, hi_s32 milli_sec)
Python函数	task_id, ret = acl.himpi.vpc_lut_remap(chn, transform_param, remap_lut, milli_sec)
函数功能	基于输入像素位置重映射表的方式对图片进行Remap处理。异步接口。 Atlas 200/300/500 推理产品 ，不支持该接口。 Atlas 训练系列产品 ，不支持该接口。 Atlas 推理系列产品 ，不支持该接口。

输入说明	chn: int, 图片处理通道号。 ● Atlas A2训练系列产品/Atlas 800I A2推理产品, 该参数的取值范围: [0, 256), 通道总数最多256。 ● Atlas 200I/500 A2推理产品, 该参数的取值范围: [0, 128), 通道总数最多128。 transform_param: dict, Remap变换参数字典, 具体请参见2.10.8.5.21 hi_warp_transform_param。 remap_lut: 像素位置重映射表信息, 具体请参见2.10.8.5.18 hi_remap_lut。 需提前调用以下get接口获取像素位置重映射表信息: acl.himpi.vpc_get_affine_lut、 acl.himpi.vpc_get_perspective_lut或 acl.himpi.vpc_get_remap_lut, 但本接口的缩放算法需与get接口的缩放算法保持一致, 否则可能导致remap结果不符合预期。 milli_sec: int, 超时时间配置, 单位是毫秒, 取值范围如下。 ● “-1”: 阻塞方式。 ● “0”: 非阻塞方式。 ● >0: 超时方式, 配置具体的超时时间。超时时间受操作系统影响, 一般偏差在操作系统的<code>一个时间片</code>内, 例如, 操作系统的<code>一个时间片</code>为4ms, 用户设置的<code>milli_sec</code>参数值为1, 则实际的超时时间在1ms到5ms范围内。在CPU负载高场景下, 超时时间仍可能存在波动。
返回值说明	ret: int, 错误码。 ● 返回0表示成功。 ● 返回非0表示失败, 参见2.10.9.2 VPC图像处理返回码。 task_id: 任务分配的ID, 用来区分任务。

约束说明

版本	约束
Atlas 200I/500 A2推理产品	● 输入、输出图片分辨率的取值范围为：10 * 6 ~ 4096 * 8192。 ● 支持如下输入图片格式： HI_PIXEL_FORMAT_YUV_400 = 0, // YUV400 8bit HI_PIXEL_FORMAT_YUV_SEMIPLANAR_420 = 1, // YUV420SP NV12 8bit HI_PIXEL_FORMAT_YVU_SEMIPLANAR_420 = 2, // YUV420SP NV21 8bit HI_PIXEL_FORMAT_RGB_888 = 12, // RGB888 HI_PIXEL_FORMAT_BGR_888 = 13, // BGR888 HI_PIXEL_FORMAT_YUV_PACKED_444 = 11, // YUV444P 8bit HI_PIXEL_FORMAT_YVU_PACKED_444 = 58, // YVU444P 8bit ● 支持如下输出图片格式： HI_PIXEL_FORMAT_YUV_400 = 0, // YUV400 8bit HI_PIXEL_FORMAT_YUV_SEMIPLANAR_420 = 1, // YUV420SP NV12 8bit HI_PIXEL_FORMAT_YVU_SEMIPLANAR_420 = 2, // YUV420SP NV21 8bit HI_PIXEL_FORMAT_RGB_888 = 12, // RGB888 HI_PIXEL_FORMAT_BGR_888 = 13, // BGR888 HI_PIXEL_FORMAT_YUV_PACKED_444 = 11, // YUV444P 8bit HI_PIXEL_FORMAT_YVU_PACKED_444 = 58, // YVU444P 8bit HI_PIXEL_FORMAT_YUYV_PACKED_422 = 7, // YUV422P YUYV 8bit HI_PIXEL_FORMAT_UYVY_PACKED_422 = 8, // YUV422P UYVY 8bit HI_PIXEL_FORMAT_YVYU_PACKED_422 = 9, // YUV422P YVYU 8bit HI_PIXEL_FORMAT_VYUY_PACKED_422 = 10, // YUV422P VYUY 8bit ● 关于图片宽高对齐、内存等约束，请参见约束说明。 ● 在任务执行过程中，需要临时缓存，因此每个通道第一次执行本接口前需先调用acl.himpi.vpc_set_chn_workspace接口申请内部临时缓存。 由于硬件约束，除YUV420SP NV12、YUV420SP NV21图片格式外，其它格式在输入与输出保持一致的情况下，无需调用acl.himpi.vpc_set_chn_workspace接口申请内部临时缓存，达到节省内存的目的。 ● 单张1080P（1920 * 1080）的图像处理超时时间建议设置10ms，其他分辨率，按照图像尺寸进行近似的等价折算，例如，4096 * 4096分辨率的图像建议超时时间设置为40ms。 ● 由于YUV格式图像下采样约束，当输出图片格式为YUV420SP或YUV422SP格式时，在配置边界填充类型时，建议设置为“边界复制模式”，避免输出图片边缘异常数据。

版本	约束
Atlas A2训练系列产品/ Atlas 800I A2推理产品	- 输入、输出图片分辨率的取值范围为：10 * 6 ~ 4096 * 8192。 - 支持如下输入图片格式： HI_PIXEL_FORMAT_YUV_400 = 0, // YUV400 8bit HI_PIXEL_FORMAT_YUV_SEMIPLANAR_420 = 1, // YUV420SP NV12 8bit HI_PIXEL_FORMAT_YVU_SEMIPLANAR_420 = 2, // YUV420SP NV21 8bit HI_PIXEL_FORMAT_RGB_888 = 12, // RGB888 HI_PIXEL_FORMAT_BGR_888 = 13, // BGR888 HI_PIXEL_FORMAT_YUV_PACKED_444 = 11, // YUV444P 8bit HI_PIXEL_FORMAT_YVU_PACKED_444 = 58, // YVU444P 8bit HI_PIXEL_FORMAT_YUV400_FP32 = 138, // YUV400 FP32, 选择该输入图片格式，则输出图片格式必须与输入保持一致 HI_PIXEL_FORMAT_RGB_888_FP32 = 1012, // RGB888 FP32, 选择该输入图片格式，则输出图片格式必须与输入保持一致 HI_PIXEL_FORMAT_BGR_888_FP32 = 1013, // BGR888 FP32, 选择该输入图片格式，则输出图片格式必须与输入保持一致 - 支持如下输出图片格式： HI_PIXEL_FORMAT_YUV_400 = 0, // YUV400 8bit HI_PIXEL_FORMAT_YUV_SEMIPLANAR_420 = 1, // YUV420SP NV12 8bit HI_PIXEL_FORMAT_YVU_SEMIPLANAR_420 = 2, // YUV420SP NV21 8bit HI_PIXEL_FORMAT_RGB_888 = 12, // RGB888 HI_PIXEL_FORMAT_BGR_888 = 13, // BGR888 HI_PIXEL_FORMAT_YUV_PACKED_444 = 11, // YUV444P 8bit HI_PIXEL_FORMAT_YVU_PACKED_444 = 58, // YVU444P 8bit HI_PIXEL_FORMAT_YUYV_PACKED_422 = 7, // YUV422P YUYV 8bit HI_PIXEL_FORMAT_UYVY_PACKED_422 = 8, // YUV422P UYVY 8bit HI_PIXEL_FORMAT_YVYU_PACKED_422 = 9, // YUV422P YVYU 8bit HI_PIXEL_FORMAT_VYUY_PACKED_422 = 10, // YUV422P VYUY 8bit HI_PIXEL_FORMAT_YUV400_FP32 = 138, // YUV400 FP32, 选择该输出图片格式，则输入图片格式必须与输出保持一致 HI_PIXEL_FORMAT_RGB_888_FP32 = 1012, // RGB888 FP32, 选择该输出图片格式，则输入图片格式必须与输出保持一致 HI_PIXEL_FORMAT_BGR_888_FP32 = 1013, // BGR888 FP32, 选择该输出图片格式，则输入图片格式必须与输出保持一致 - 关于图片宽高对齐、内存等约束，请参见约束说明。 - 在任务执行过程中，需要临时缓存，因此每个通道第一次执行本接口前需先调用acl.himpi.vpc_set_chn_workspace接口申请内部临时缓存。 由于硬件约束，除YUV420SP NV12、YUV420SP NV21图片格式外，其它格式在输入与输出保持一致的情况下，无需调用acl.himpi.vpc_set_chn_workspace接口申请内部临时缓存，达到节省内存的目的。 - 单张1080P（1920 * 1080）的图像处理超时时间建议设置10ms，其他分辨率，按照图像尺寸进行近似的等价折算，例如，4096 * 4096分辨率的图像建议超时时间设置为40ms。 - 由于YUV格式图像下采样约束，当输出图片格式为YUV420SP或YUV422SP格式时，在配置边界填充类型时，建议设置为“边界复制模式”，避免输出图片边缘异常数据。

2.10.4.41 acl.himpi.vpc_median_blur

c函数原型	hi_s32 hi_mpi_vpc_median_blur(hi_vpc_chn chn, const hi_median_blur_param* median_blur_param, hi_u32*task_id, hi_s32 milli_sec)
Python函数	task_id, ret = acl.himpi.vpc_median_blur(chn, median_blur_param, milli_sec)
函数功能	对输入图片做中值滤波处理。 Atlas 200/300/500 推理产品 ，不支持该接口。 Atlas 训练系列产品 ，不支持该接口。 Atlas 推理系列产品 ，不支持该接口。
输入说明	chn : int, 图片处理通道号。 • Atlas A2训练系列产品/Atlas 800I A2推理产品，该参数的取值范围：[0, 256)，通道总数最多256。 • Atlas 200I/500 A2推理产品，该参数的取值范围：[0, 128)，通道总数最多128。 median_blur_param : dict, 中值滤波信息字典，具体请参见 2.10.8.5.26 hi_median_blur_param 。 milli_sec : int, 超时时间配置，单位是毫秒，取值范围如下。 • “-1”：阻塞方式。 • “0”：非阻塞方式。 • >0：超时方式，配置具体的超时时间。超时时间受操作系统影响，一般偏差在操作系统的一个时间片内，例如，操作系统的-一个时间片为4ms，用户设置的milli_sec参数值为1，则实际的超时时间在1ms到5ms范围内。在CPU负载高场景下，超时时间仍可能存在波动。
返回值说明	ret : int, 错误码。 • 返回0表示成功。 • 返回非0表示失败，参见2.10.9.2 VPC图像处理返回码。 task_id : 任务分配的ID，用来区分任务。

约束说明

- 输入、输出图片分辨率的取值范围为：10 * 6 ~ 4096 * 8192，且输入、输出图片的分辨率需保持一致。
- 支持如下输入图片格式（输出图片格式与输入保持一致）：

```
HI_PIXEL_FORMAT_YUV_400 = 0,           // YUV400 8bit
HI_PIXEL_FORMAT_YUV_SEMIPLANAR_420 = 1, // YUV420SP NV12 8bit
HI_PIXEL_FORMAT_YVU_SEMIPLANAR_420 = 2, // YUV420SP NV21 8bit
HI_PIXEL_FORMAT_YUV_SEMIPLANAR_422 = 3, // YUV422SP 8bit
HI_PIXEL_FORMAT_YVU_SEMIPLANAR_422 = 4, // YVU422SP 8bit
HI_PIXEL_FORMAT_YUYV_PACKED_422 = 7,    // YUV422Packed YUYV 8bit
HI_PIXEL_FORMAT_YVYU_PACKED_422 = 9,    // YUV422Packed YVYU 8bit
HI_PIXEL_FORMAT_YUV_PACKED_444 = 11,    // YUV444 Package 8bit
HI_PIXEL_FORMAT_RGB_888 = 12,           // RGB888
HI_PIXEL_FORMAT_BGR_888 = 13,           // BGR888
HI_PIXEL_FORMAT_YVU_PACKED_444 = 58,    // YVU444 Package 8bit
```

HI_PIXEL_FORMAT_RGB_888_PLANAR = 69, // RGB888 Planar
HI_PIXEL_FORMAT_BGR_888_PLANAR = 70, // BGR888 Planar

2.10.4.42 acl.himpi.vpc_rotate

c函数原型	hi_s32 hi_mpi_vpc_rotate(hi_vpc_chn chn, const hi_rotate_param* rotate_param, hi_u32 *task_id, hi_s32 milli_sec)
Python函数	task_id, ret = acl.himpi.vpc_rotate(chn, rotate_param, milli_sec)
函数功能	对输入图片做90度/180度/270度操作，输出图片宽高也要随旋转角度置换。 Atlas 200/300/500 推理产品 ，不支持该接口。 Atlas 训练系列产品 ，不支持该接口。 Atlas 推理系列产品 ，不支持该接口。
输入说明	chn : int，图片处理通道号。 Atlas A2训练系列产品/Atlas 800I A2推理产品，该参数的取值范围：[0, 256)，通道总数最多256。 Atlas 200I/500 A2推理产品，该参数的取值范围：[0, 128)，通道总数最多128。 rotate_param : dict，旋转信息字典，具体请参见 2.10.8.5.35 hi_rotate_param 。 milli_sec : int，超时时间配置，单位是毫秒，取值范围如下。 “-1”：阻塞方式。 “0”：非阻塞方式。 >0：超时方式，配置具体的超时时间。超时时间受操作系统影响，一般偏差在操作系统的一个时间片内，例如，操作系统的 一个时间片为4ms，用户设置的milli_sec参数值为1，则实际的超时时间在1ms到5ms范围内。在CPU负载高场景下，超时时间仍可能存在波动。
返回值说明	ret : int，错误码。 - 返回0表示成功。 - 返回非0表示失败，参见2.10.9.2 VPC图像处理返回码。 task_id : 任务分配的ID，用来区分任务。

约束说明

版本	约束
Atlas 200I/500 A2推理产品	- 输入、输出图片分辨率的取值范围为：10 * 6 ~ 4096 * 8192。 - 支持如下输入图片格式（输出图片格式与输入保持一致）： HI_PIXEL_FORMAT_YUV_400 = 0, // YUV400 8bit HI_PIXEL_FORMAT_YUV_SEMIPLANAR_420 = 1, // YUV420SP NV12 8bit HI_PIXEL_FORMAT_YVU_SEMIPLANAR_420 = 2, // YUV420SP NV21 8bit HI_PIXEL_FORMAT_YUV_PACKED_444 = 11, // YUV444 Package 8bit HI_PIXEL_FORMAT_RGB_888 = 12, // RGB888 HI_PIXEL_FORMAT_BGR_888 = 13, // BGR888 HI_PIXEL_FORMAT_YVU_PACKED_444 = 58, // YVU444 Package 8bit HI_PIXEL_FORMAT_RGB_888_PLANAR = 69, // RGB888 Planar HI_PIXEL_FORMAT_BGR_888_PLANAR = 70, // BGR888 Planar
Atlas A2训练系列产品/ Atlas 800I A2推理产品	- 输入、输出图片分辨率的取值范围为：10 * 6 ~ 4096 * 8192。 - 支持如下输入图片格式（输出图片格式与输入保持一致）： HI_PIXEL_FORMAT_YUV_400 = 0, // YUV400 8bit HI_PIXEL_FORMAT_YUV_PACKED_444 = 11, // YUV444 Package 8bit HI_PIXEL_FORMAT_RGB_888 = 12, // RGB888 HI_PIXEL_FORMAT_BGR_888 = 13, // BGR888 HI_PIXEL_FORMAT_YVU_PACKED_444 = 58, // YVU444 Package 8bit HI_PIXEL_FORMAT_YUV400_FP32 = 138, // YUV400 Package 每个像素用1个float32数据表示 HI_PIXEL_FORMAT_RGB_888_FP32 = 1012, // RGB888 Package, 每个像素用1个float32数据表示 HI_PIXEL_FORMAT_BGR_888_FP32 = 1013, // BGR888 Package, 每个像素用1个float32数据表示

2.10.4.43 acl.himpi.vpc_set_chn_workspace

c函数原型	hi_s32 hi_mpi_vpc_set_chn_workspace(hi_vpc_chn chn, hi_vpc_workspace_param *workspace_param, hi_u32 param_cnt)
Python函数	ret = acl.himpi.vpc_set_chn_workspace(chn, workspace_param, param_cnt)
函数功能	调用本接口后，系统内部会根据输入图片、输出图片的最大宽高信息申请需使用的内部临时缓存，通过该接口申请的临时缓存，在销毁通道时会自动释放。 Atlas 200/300/500 推理产品，不支持该接口。 Atlas 训练系列产品，不支持该接口。 Atlas 推理系列产品，不支持该接口。

输入说明	chn: int, 图片处理通道号。 • Atlas A2训练系列产品/Atlas 800I A2推理产品, 该参数的取值范围: [0, 256), 通道总数最多256。 • Atlas 200I/500 A2推理产品, 该参数的取值范围: [0, 128), 通道总数最多128。 workspace_param: list, 元素类型为2.10.8.5.16 hi_vpc_workspace_param字典的数组。 此处需根据所使用的Remap功能或仿射变换功能来设置2.10.8.5.16 hi_vpc_workspace_param["func"]参数值。 若在一个应用程序中, 涉及Remap、仿射变换两种功能, 则可以在2.10.8.5.16 hi_vpc_workspace_param数组中包含2个元素, 一个元素的hi_vpc_workspace_param["func"]设置为REMAP, 一个元素的hi_vpc_workspace_param["func"]设置为AFFINE。 param_cnt: int, 数组长度, 需与“workspace_param”数组长度保持一致。
返回值说明	ret: int, 错误码。 • 返回0表示成功。 • 返回非0表示失败, 参见2.10.9.2 VPC图像处理返回码。
约束说明	对于每个通道, 该接口只能设置一次, 后续不能修改。

2.10.4.44 acl.himpi.vpc_warp_affine

c函数原型	hi_s32 hi_mpi_vpc_warp_affine(hi_vpc_chn chn, hi_transform_matrix *matrix, hi_warp_transform_param *transform_param, hi_u32 *task_id, hi_s32 milli_sec)
Python函数	task_id, ret = acl.himpi.vpc_warp_affine(chn, matrix, transform_param, milli_sec)
函数功能	对输入图片做仿射变换。 Atlas 200/300/500 推理产品, 不支持该接口。 Atlas 训练系列产品, 不支持该接口。 Atlas 推理系列产品, 不支持该接口。

输入说明	chn: int, 图片处理通道号。 • Atlas A2训练系列产品/Atlas 800I A2推理产品, 该参数的取值范围: [0, 256), 通道总数最多256。 • Atlas 200I/500 A2推理产品, 该参数的取值范围: [0, 128), 通道总数最多128。 matrix: dict, 仿射变换需要的变换矩阵信息字典, 具体参见2.10.8.5.46 hi_transform_matrix。使用acl.himpi.vpc_get_affine_transform或者acl.himpi.vpc_get_rotation_matrix接口获取的仿射变换矩阵信息字典。 transform_param: dict, 仿射变换参数, 在2.10.8.5.21 hi_warp_transform_param字典内配置输入、输出图片信息, 相关约束请参见约束说明。 milli_sec: int, 超时时间配置, 单位是毫秒, 取值范围如下。 • “-1”: 阻塞方式。 • “0”: 非阻塞方式。 • >0: 超时方式, 配置具体的超时时间。超时时间受操作系统影响, 一般偏差在操作系统的一个时间片内, 例如, 操作系统的-一个时间片为4ms, 用户设置的milli_sec参数值为1, 则实际的超时时间在1ms到5ms范围内。在CPU负载高场景下, 超时时间仍可能存在波动。
返回值说明	ret: int, 错误码。 • 返回0表示成功。 • 返回非0表示失败, 参见2.10.9.2 VPC图像处理返回码。 task_id: 任务分配的ID, 用来区分任务。

约束说明

- 输入、输出图片分辨率的取值范围为: 10 * 6 ~ 4096 * 8192。
- 关于输入、输出图片格式的要求如下 (关于图片宽高对齐、内存等约束, 请参见[约束说明](#)) :

版本	约束
Atlas 200I/500 A2推理产品	- 支持如下输入图片格式： HI_PIXEL_FORMAT_YUV_400 = 0, // YUV400 8bit HI_PIXEL_FORMAT_YUV_SEMIPLANAR_420 = 1, // YUV420SP NV12 8bit HI_PIXEL_FORMAT_YVU_SEMIPLANAR_420 = 2, // YUV420SP NV21 8bit HI_PIXEL_FORMAT_RGB_888 = 12, // RGB888 HI_PIXEL_FORMAT_BGR_888 = 13, // BGR888 HI_PIXEL_FORMAT_YUV_PACKED_444 = 11, // YUV444P 8bit HI_PIXEL_FORMAT_YVU_PACKED_444 = 58, // YVU444P 8bit - 支持如下输出图片格式： HI_PIXEL_FORMAT_YUV_400 = 0, // YUV400 8bit HI_PIXEL_FORMAT_YUV_SEMIPLANAR_420 = 1, // YUV420SP NV12 8bit HI_PIXEL_FORMAT_YVU_SEMIPLANAR_420 = 2, // YUV420SP NV21 8bit HI_PIXEL_FORMAT_RGB_888 = 12, // RGB888 HI_PIXEL_FORMAT_BGR_888 = 13, // BGR888 HI_PIXEL_FORMAT_YUV_PACKED_444 = 11, // YUV444P 8bit HI_PIXEL_FORMAT_YVU_PACKED_444 = 58, // YVU444P 8bit HI_PIXEL_FORMAT_YUYV_PACKED_422 = 7, // YUV422P YUYV 8bit HI_PIXEL_FORMAT_UYVY_PACKED_422 = 8, // YUV422P UYVY 8bit HI_PIXEL_FORMAT_YVYU_PACKED_422 = 9, // YUV422P YVYU 8bit HI_PIXEL_FORMAT_VYUY_PACKED_422 = 10, // YUV422P VYUY 8bit

版本	约束
Atlas A2训练系列产品/Atlas 800I A2推理产品	- 支持如下输入图片格式： HI_PIXEL_FORMAT_YUV_400 = 0, // YUV400 8bit HI_PIXEL_FORMAT_YUV_SEMIPLANAR_420 = 1, // YUV420SP NV12 8bit HI_PIXEL_FORMAT_YVU_SEMIPLANAR_420 = 2, // YUV420SP NV21 8bit HI_PIXEL_FORMAT_RGB_888 = 12, // RGB888 HI_PIXEL_FORMAT_BGR_888 = 13, // BGR888 HI_PIXEL_FORMAT_YUV_PACKED_444 = 11, // YUV444P 8bit HI_PIXEL_FORMAT_YVU_PACKED_444 = 58, // YVU444P 8bit HI_PIXEL_FORMAT_YUV400_FP32 = 138, // YUV400 FP32, 选择该输入图片格式，则输出图片格式必须与输入保持一致 HI_PIXEL_FORMAT_RGB_888_FP32 = 1012, // RGB888 FP32, 选择该输入图片格式，则输出图片格式必须与输入保持一致 HI_PIXEL_FORMAT_BGR_888_FP32 = 1013, // BGR888 FP32, 选择该输入图片格式，则输出图片格式必须与输入保持一致 - 支持如下输出图片格式： HI_PIXEL_FORMAT_YUV_400 = 0, // YUV400 8bit HI_PIXEL_FORMAT_YUV_SEMIPLANAR_420 = 1, // YUV420SP NV12 8bit HI_PIXEL_FORMAT_YVU_SEMIPLANAR_420 = 2, // YUV420SP NV21 8bit HI_PIXEL_FORMAT_RGB_888 = 12, // RGB888 HI_PIXEL_FORMAT_BGR_888 = 13, // BGR888 HI_PIXEL_FORMAT_YUV_PACKED_444 = 11, // YUV444P 8bit HI_PIXEL_FORMAT_YVU_PACKED_444 = 58, // YVU444P 8bit HI_PIXEL_FORMAT_YUYV_PACKED_422 = 7, // YUV422P YUYV 8bit HI_PIXEL_FORMAT_UYVY_PACKED_422 = 8, // YUV422P UYVY 8bit HI_PIXEL_FORMAT_VYVU_PACKED_422 = 9, // YUV422P VYVU 8bit HI_PIXEL_FORMAT_VYUY_PACKED_422 = 10, // YUV422P VYUY 8bit HI_PIXEL_FORMAT_YUV400_FP32 = 138, // YUV400 FP32, 选择该输出图片格式，则输入图片格式必须与输出保持一致 HI_PIXEL_FORMAT_RGB_888_FP32 = 1012, // RGB888 FP32, 选择该输出图片格式，则输入图片格式必须与输出保持一致 HI_PIXEL_FORMAT_BGR_888_FP32 = 1013, // BGR888 FP32, 选择该输出图片格式，则输入图片格式必须与输出保持一致

- 在任务执行过程中，需要临时缓存，因此每个通道第一次执行本接口前需先调用[acl.himpi.vpc_set_chn_workspace](#)接口申请内部临时缓存。
- 由于[YUV格式图像下采样约束](#)，当输出图片格式为YUV420SP或YUV422SP格式时，在配置边界填充类型时，建议设置为“边界复制模式”，避免输出图片边缘异常数据。

2.10.5 VDEC 视频解码/JPEGD 图像解码接口

2.10.5.1 JPEGD 功能及约束说明

Atlas 200/300/500 推理产品上，该功能为预留功能，暂不支持。

Atlas 训练系列产品上，该功能为预留功能，暂不支持。

功能说明

JPEGD (JPEG Decoder) 实现.jpg、.jpeg、.JPG、.JPEG图片文件的解码。

● **JPEGD在解码图片时，支持对图片进行旋转。**

如果输入图片的码流中包含**Orientation信息**（代表捕获图像时摄像机相对于场景的方向），则JPEGD在解码时会解析Orientation信息，将图片进行90度、180度、270度或镜像旋转。旋转后输出图片的宽Stride、高Stride、输出内存仍需满足**图片格式、宽高对齐、内存约束**中的要求。如果输入图片的码流异常，导致JPEGD解码时无法读取Orientation信息，则不能实现图片旋转的功能。

针对以下产品版本，旋转图片时存在约束：

产品版本	约束
Atlas 200I/500 A2推理产品 Atlas A2训练系列产品/Atlas 800I A2推理产品	JPEGD 422格式 旋转图片，若原图高为奇数，并且旋转方式为宽高对换，则旋转后宽存在黑边，建议用户将宽向下对齐到偶数使用（即去除黑边），比如原图为200 * 101，若旋转后为101 * 200，则建议用户实际使用区域为100 * 200。 JPEGD 440格式 旋转图片，若原图宽为奇数，并且旋转方式为宽高对换，则旋转后高存在黑边，建议用户将高向下对齐到偶数使用（即去除黑边），比如原图为201 * 101，若旋转后为100 * 201，则建议用户实际使用区域为100 * 200。

● **JPEGD在解码图片时，支持按源图片格式解码。**

源图片格式解码是指解码前后图片的编码格式保持一致，例如解码前输入图片为jpeg(440)，解码后输出图片为YUV440SP V在前U在后或YUV440SP U在前V在后。

使用源图片格式解码，有以下方式：

- 在调用JPEGD解码接口时，直接将输出图片格式配置为“HI_PIXEL_FORMAT_UNKNOWN”，输出格式默认按源图片格式输出、且是V在前U在后的Semi-Planar格式。例如，JPEG源图片格式为jpeg(440)，输出图片格式配置为“HI_PIXEL_FORMAT_UNKNOWN”，JPEGD解码后，实际输出图片格式为YUV440SP V在前U在后。
此种方式，由于不知道输出图片格式，因此需要用户申请尽量大的内存或调用[acl.himpi.dvpp_get_image_info](#)接口获取解码输出内存大小，防止内存不够，无法存放输出图片。
- 先调用[acl.himpi.dvpp_get_image_info](#)接口根据传入的jpeg源图片，获取按源图解码时的输出图片的宽、高、宽Stride、高Stride、解码输出内存大小、图片格式等信息后，再调用JPEGD解码接口，使用通过[acl.himpi.dvpp_get_image_info](#)接口获取的图片格式来设置输出图片格式。

 说明

JPEGD解码后的输出图片，如果要直接作为模型推理的输入，建议将输出图片格式配置为“HI_PIXEL_FORMAT_UNKNOWN”，这时JPEGD使用源图片格式解码（但这里要确保解码后的图片格式模型是支持的），保证模型推理的精度。

JPEGD解码后的输出图片，如果直接作为VPC的输入，该场景下若使用源图片格式解码时，则需要关注解码后的输出图片格式VPC是否支持（VPC输入图片的格式请参见[2.10.4.2 约束说明](#)），如果VPC不支持，则用户需按VPC支持的情况指定JPEGD的输出图片格式。

图片分辨率约束

- 输入图片分辨率

版本	分辨率范围
Atlas 推理系列产品	最大分辨率：8192 * 8192，最小分辨率：32 * 32。
Atlas A2训练系列产品/Atlas 800I A2推理产品 Atlas 200I/500 A2推理产品	最大分辨率：16384 * 16384，最小分辨率：32 * 32。
- 输出图片分辨率

JPEGD只对图片解码，不会改变图片分辨率，因此输出与输入的图片分辨率保持一致。

内存申请/释放接口说明

输入内存的大小就是指实际的输入图片所占用的大小，输出内存的大小参见表2-48中的计算公式。

版本	内存申请/释放接口
Atlas 推理系列产品	JPEGD图片解码时，调用 acl.himpi.dvpp_malloc \ acl.himpi.dvpp_free 接口申请\释放Device上的输入、输出内存，这部分内存的生命周期由用户自行管理。
Atlas A2训练系列产品/ Atlas 800I A2推理产品 Atlas 200I/500 A2推理产品	JPEGD图片解码时，支持以下两种内存申请接口： - 调用acl.rt.malloc\acl.rt.free接口申请\释放Device上的输入、输出内存，这部分内存的生命周期由用户自行管理。 - 调用acl.himpi.dvpp_malloc\acl.himpi.dvpp_free接口申请\释放Device上的输入、输出内存，这部分内存的生命周期由用户自行管理。 调用 acl.himpi.dvpp_malloc 接口申请的内存为媒体数据处理的专用内存，但专用内存的地址空间有限，若关注内存规划或内存资源有限时，建议调用 acl.rt.malloc 接口申请内存。

图片格式、宽高对齐、内存约束

实现JPEGD图片解码功能时，仅支持Huffman编码，压缩前的原图像色彩空间为YUV，像素的各分量比例为4:4:4、4:2:2、4:2:0、4:0:0或4:4:0，不支持算术编码、不支持渐进JPEG格式、不支持JPEG2000格式。

说明

输出图片格式的定义请参见[hi_pixel_format](#)，宽Stride、高Stride等概念请参见[2.10.2 基本概念](#)。

表 2-48 图片格式、宽高对齐、内存大小约束

输入图片格式 (YUV分量比例)	输出图片格式	输出图片宽、高对齐要求	输出图片宽Stride、高Stride、内存大小要求
jpeg(444)	YVU444SP 8bit	无对齐要求	宽Stride为宽64对齐后的值。 高Stride为高16对齐后的值。 内存大小 (单位Byte) ≥ 宽Stride * 高Stride * 3
	YUV444SP 8bit		
	YUV420SP NV12 8bit	宽2对齐 高2对齐	宽Stride为宽64对齐后的值。 高Stride为高16对齐后的值。 内存大小 (单位Byte) ≥ 宽Stride * 高Stride * 3/2
	YUV420SP NV21 8bit		
jpeg(422)	YVU422SP 8bit	宽2对齐 高无对齐要求	宽Stride为宽64对齐后的值。 高Stride为高16对齐后的值。 内存大小 (单位Byte) ≥ 宽Stride * 高Stride * 2
	YUV422SP 8bit		
	YUV420SP NV12 8bit	宽2对齐 高2对齐	宽Stride为宽64对齐后的值。 高Stride为高16对齐后的值。 内存大小 (单位Byte) ≥ 宽Stride * 高Stride * 3/2
	YUV420SP NV21 8bit		
jpeg(420)	YUV420SP NV12 8bit	宽2对齐 高2对齐	宽Stride为宽64对齐后的值。 高Stride为高16对齐后的值。 内存大小 (单位Byte) ≥ 宽Stride * 高Stride * 3/2
	YUV420SP NV21 8bit		
jpeg(400)	YUV420SP NV12 8bit	宽2对齐 高2对齐	宽Stride为宽64对齐后的值。 高Stride为高16对齐后的值。 内存大小 (单位Byte) ≥ 宽Stride * 高Stride * 3/2
	YUV420SP NV21 8bit		
	YUV400 8bit	无对齐要求	宽Stride为宽64对齐后的值。 高Stride为高16对齐后的值。 内存大小 (单位Byte) ≥ 宽Stride * 高Stride
jpeg(440)	YVU440SP 8bit	宽没有对齐要求 高2对齐	宽Stride为宽64对齐后的值。 高Stride为高16对齐后的值。 内存大小 (单位Byte) ≥ 宽Stride * 高Stride * 2
	YUV440SP 8bit		
	YUV420SP NV12 8bit	宽2对齐 高2对齐	宽Stride为宽64对齐后的值。 高Stride为高16对齐后的值。 内存大小 (单位Byte) ≥ 宽Stride * 高Stride * 3/2

输入图片格式 (YUV分量比例)	输出图片格式	输出图片宽、高 对齐要求	输出图片宽Stride、高Stride、 内存大小要求
	YUV420SP NV21 8bit		

软、硬件约束

- 硬件约束：
 - 最多支持4张Huffman表，其中包括2张DC（ Direct Current ）（ 直流 ）表和2张AC（ Alternating Current ）（ 交流 ）表。
 - 最多支持3张量化表。
 - 只支持8bit采样精度。
 - 只支持对顺序式编码的图片进行解码。
 - 只支持基于DCT（ Discrete Cosine Transform ）变换的JPEG 格式解码。
 - 只支持一个SOS（ Start of Scan ）标志的图片解码。
- 软件约束：
 - 支持3个SOS标志的图片解码。
 - 支持MCU（ Minimum Coded Unit ）数据不足的异常图片解码。

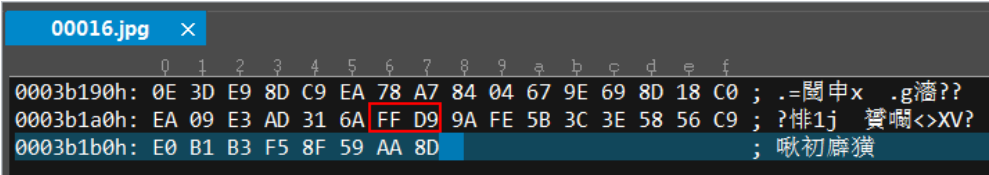
精度相关约束

JPEGD+VPC串联使用时，由于JPEGD解码后的输出图片的宽Stride * 高Stride有64 * 16对齐的约束，因此解码后的输出图片的宽、高有一些补边的无效数据，所以在执行VPC功能时（例如缩放时），需在输入图片的hi_vpc_pic_info["picture_width"]、hi_vpc_pic_info["picture_height"]属性处正确设置输入图片的原图宽高，这样VPC在缩放图片前会先根据原图宽高自行抠图，目的是去除无效数据对图像精度的影响。

其它注意事项

若图片内EOI（ End Of Image，标记代码为0xFFD9）之后，还有用户自定义的数据，则JPEGD在对图片进行解码时，会直接清零EOI之后的8字节数据，若用户需要保留这些自定义的数据，则将图片数据读入内存之后，需要提前备份这部分数据，再传给JPEGD处理。

若需要查看图片内EOI之后是否存在自定义数据，可以使用二进制查看工具打开图片查看，例如下图中的FFD9标记符之后就存在自定义数据。



2.10.5.2 JPEGD 性能指标数据

Atlas 200/300/500 推理产品上，该功能为预留功能，暂不支持。

Atlas 训练系列产品上，该功能为预留功能，暂不支持。

性能指标说明

本节仅描述各产品对应的昇腾AI处理器的性能指标参考数据，各产品由于其内存、CPU等配置不同，在性能规格上会存在差异，详细产品的性能规格请查阅[昇腾社区](#)的官方说明。

本节以单个昇腾AI处理器为例，描述基本场景性能指标参考，其它分辨率可以等量估算。

下表中的1路对应一个通道，一个通道对应一个线程；各分辨率的含义为：720p指分辨率为1280 * 720的图片，1080p指分辨率为1920 * 1080的图片，4K指分辨率为3840 * 2160的图片。

JPEGD性能指标是基于硬件解码的性能，JPEGD硬件解码不支持3个SOS的图片解码，对于硬件不支持的格式，会使用软件解码，软件解码性能参考为1080P 15fps。JPEGD解码的输出图片如果涉及旋转，则性能指标低于软件解码的参考值，例如对于1080P的图片，性能指标低于15fps。

表 2-49 Atlas 推理系列产品的昇腾 AI 处理器性能指标参考数据

场景举例	总帧率
1080p * n路 (1 ≤ n ≤ 16)	n * 128fps
1080p * n路 (n > 16)	2048fps
4k * n路 (1 ≤ n ≤ 16)	n * 32fps
4k * n路 (n > 16)	512fps

表 2-50 Atlas 200I/500 A2 推理产品的昇腾 AI 处理器性能指标参考数据

场景举例	总帧率
1080p * n路 (1 ≤ n ≤ 2)	n * 256fps
1080p * n路 (n > 2)	512fps
4k * n路 (1 ≤ n ≤ 2)	n * 64fps
4k * n路 (n > 2)	128fps

表 2-51 Atlas A2 训练系列产品/Atlas 800I A2 推理产品性能规格数据

输入图片格式	输入图片文件大小	场景举例	总帧率
YUV400/420	≤ 500KB	1080p * n路 (1 ≤ n ≤ 28)	n * 512fps
		1080p * n路 (n > 28)	14336fps
YUV440/422/444	≤ 500KB	1080p * n路 (1 ≤ n ≤ 28)	n * 256fps
		1080p * n路 (n > 28)	7168fps
YUV400/420	≤ 2MB	4k * n路 (1 ≤ n ≤ 28)	n * 128fps
		4k * n路 (n > 28)	3584fps
YUV440/422/444	≤ 2MB	4k * n路 (1 ≤ n ≤ 28)	n * 64fps
		4k * n路 (n > 28)	1792fps

2.10.5.3 VDEC 功能及约束说明

Atlas 200/300/500 推理产品上，不支持本章中的VDEC功能。

Atlas 训练系列产品上，不支持本章中的VDEC功能。

功能说明

VDEC (Video Decoder) 实现视频的解码。

分辨率约束

- 输入码流分辨率：

版本	分辨率范围
Atlas 推理系列产品 Atlas A2训练系列产品/Atlas 800I A2推理产品	最大分辨率4096 * 4096，最小分辨率128 * 128。
Atlas 200I/500 A2推理产品	对于H.264输入码流，最大分辨率8192 * 8192，最小分辨率128 * 128。 对于H.265输入码流，最大分辨率5400 * 8192，最小分辨率128 * 128。

- 输出图片分辨率：

版本	分辨率范围
Atlas 推理系列产品 Atlas A2训练系列产品/Atlas 800I A2推理产品	最大分辨率4096 * 4096，最小分辨率10 * 6。
Atlas 200I/500 A2推理产品	输出图片最大分辨率8192 * 8192。 - 当输入码流宽度小于或等于4096时，输出图片最小分辨率为10 * 6。 - 当输入码流宽度大于4096时，输出图片最小分辨率为128 * 128。

内存申请/释放接口说明

输入内存的大小就是指实际的输入码流所占用的大小，输出内存的大小参见表2-52中的计算公式。

版本	内存申请接口
Atlas 推理系列产品	VDEC视频解码时，调用 acl.himpi.dvpp_malloc \ acl.himpi.dvpp_free 接口申请\释放Device上的输入、输出内存，这部分内存的生命周期由用户自行管理。
Atlas A2训练系列产品/ Atlas 800I A2推理产品 Atlas 200I/500 A2推理产品	VDEC视频解码时，支持以下两种内存申请接口： - 调用acl.rt.malloc \ acl.rt.free接口申请\释放Device上的输入、输出内存，这部分内存的生命周期由用户自行管理。 - 调用acl.himpi.dvpp_malloc \ acl.himpi.dvpp_free接口申请\释放Device上的输入、输出内存，这部分内存的生命周期由用户自行管理。 注意：调用acl.himpi.dvpp_malloc接口申请的内存为媒体数据处理的专用内存，但专用内存的地址空间有限，若关注内存规划或内存资源有限时，建议调用acl.rt.malloc接口申请内存。

码流/图片格式、宽高对齐、内存约束

- 输入码流格式。
 - H.264 bp/mp/hp level5.1 YUV420编码的码流，当前只支持annex-B的裸码流。
 - H.265 8/10bit level5.1 YUV420编码的码流，当前只支持annex-B的裸码流。
- 输出图片要求：

表 2-52 码流/图片格式、宽高对齐、内存大小约束

输出图片格式	输出图片宽、高 对齐要求	输出图片宽 Stride、高 Stride、内存大小 要求 Atlas 推理系列产 品	输出图片宽 Stride、高 Stride、内存大小 要求 Atlas 200I/500 A2推理产品 Atlas A2训练系 列产品/Atlas 800I A2推理产品
YUV420SP NV12 8bit	宽2对齐 高2对齐	宽Stride为宽16对 齐后的值，最小 32，最大16384。 高Stride为高2对 齐后的值，最小 6，最大16384。 内存大小（单位 Byte）≥ 宽Stride * 高Stride * 3/2	宽Stride无对齐要 求，与宽相同即 可，最小10，最 大24576。 高Stride无对齐要 求，与高相同即 可，最小6，最大 24576。 内存大小（单位 Byte）≥ 宽Stride * 高Stride * 3/2
YUV420SP NV21 8bit	宽2对齐 高2对齐		
RGB888	无对齐要求	宽Stride为宽16对 齐后再乘以3的 值，最小48，最 大16384。 高Stride无对齐要 求，最小6，最大 16384。 内存大小（单位 Byte）≥ 宽Stride * 高Stride	宽Stride为宽乘以 3的值，最小30， 最大24576。 高Stride无对齐要 求，最小6，最大 16384。 内存大小（单位 Byte）≥ 宽Stride * 高Stride
BGR888	无对齐要求		
RGB888 Planar Atlas A2训练系列 产品/Atlas 800I A2推理产品支持 该格式。	无对齐要求		宽stride无对齐要 求，与宽相同即 可，最小10，最 大24576。 高stride无对齐要 求，与高相同即 可，最小6，最大 24576。 内存大小（单位 Byte）≥ 宽stride * 高stride
BGR888 Planar Atlas A2训练系列 产品/Atlas 800I A2推理产品支持 该格式。	无对齐要求		

 说明

输出图片格式的定义请参见[hi_pixel_format](#)，宽Stride、高Stride等概念请参见[2.10.2 基本概念](#)。

其它约束

- **Atlas A2训练系列产品/Atlas 800I A2推理产品**，若输出图片格式为YUV420SP NV12 10bit或YUV420SP NV21 10bit，使用VDEC功能时，不支持缩放图片、色域转换。
- VDEC只支持对按帧输入码流进行解码。
- 若码流中有坏帧、缺帧等情况，解码器VDEC解码时会将该帧标记为解码失败，并上报异常。
- 通过隔行扫描方式编码出来的码流，VDEC仅支持解码H.264 8bit编码的码流。
- 同时配置按帧发送码流与按解码序输出，可以达到快速解码和快速输出的目的，但这种场景不支持解码含有B帧的码流。

2.10.5.4 VDEC 性能指标数据

Atlas 200/300/500 推理产品上，不支持本章中的VDEC功能。

Atlas 训练系列产品上，不支持本章中的VDEC功能。

性能指标说明

本节仅描述各产品对应的昇腾AI处理器的性能指标参考数据，各产品由于其内存、CPU等配置不同，在性能规格上会存在差异，详细产品的性能规格请查阅[昇腾社区](#)的官方说明。

本节以单个昇腾AI处理器为例，描述基本场景性能指标参考，其它分辨率可以等量估算。

下表中的1路对应一个通道，一个通道对应一个线程；各分辨率的含义为：720p指分辨率为1280 * 720的图片，1080p指分辨率为1920 * 1080的图片，4K指分辨率为3840 * 2160的图片。

表 2-53 Atlas 推理系列产品性能规格数据

场景举例	总帧率
720p * n路($1 \leq n \leq 12$)	n * 480fps 说明 针对 Atlas 300V 视频解析卡，当 $1 < n \leq 12$ 时，总帧率都为n * 400fps。
720p * n路($n > 12$)	5760fps 说明 针对 Atlas 300V 视频解析卡，当 $n > 12$ 时，总帧率都为4800fps。
1080p * n路($1 \leq n \leq 12$)	n * 240fps 说明 针对 Atlas 300V 视频解析卡，当 $1 < n \leq 12$ 时，总帧率都为n * 200fps。

场景举例	总帧率
1080p * n路($n > 12$)	2880fps 说明 针对 Atlas 300V 视频解析卡，当 $n > 12$ 时，总帧率都为2400fps。
4k * n路($1 \leq n \leq 12$)	$n * 60\text{fps}$
4k * n路($n > 12$)	720fps

表 2-54 Atlas 200I/500 A2 推理产品性能规格数据

场景举例	总帧率
720p * n路($1 \leq n \leq 4$)	$n * 600\text{fps}$
720p * n路($n > 4$)	2400fps
1080p * n路($1 \leq n \leq 4$)	$n * 300\text{fps}$
1080p * n路($n > 4$)	1200fps
4k * n路($1 \leq n \leq 4$)	$n * 75\text{fps}$
4k * n路($n > 4$)	300fps

表 2-55 Atlas A2 训练系列产品/Atlas 800I A2 推理产品性能规格数据

场景举例	总帧率
720p * n路($1 \leq n \leq 2$)	$n * 480\text{fps}$
720p * n路($n > 2$)	960fps
1080p * n路($1 \leq n \leq 2$)	$n * 240\text{fps}$
1080p * n路($n > 2$)	480fps
4k * n路($1 \leq n \leq 2$)	$n * 60\text{fps}$
4k * n路($n > 2$)	120fps

下表以1080P分辨率的输入码流为例，说明每路VDEC解码的最大内存消耗的计算公式，在计算公式中：

- 输入码流缓存大小：大于或等于解码通道大小（宽 * 高）的3/4倍。
- 解码图像帧存大小：可调用[acl.himpi.vdec_get_pic_buf_size](#)接口获取解码图像帧存大小，该参数值跟输入码流分辨率相关。
- 视频解码图像Tmv缓存大小：可调用[acl.himpi.vdec_get_tmv_buf_size](#)接口获取视频解码图像Tmv缓存大小，该参数值跟输入码流分辨率相关。

- 输入码流缓存大小、解码图像帧存大小、视频解码图像Tmv缓存大小、参考帧数量均由用户调用[acl.himpi.vdec_create_chn](#)接口创建通道时设置。
- 解码后缓存图像帧数由用户在调用[acl.himpi.vdec_set_chn_param](#)接口时设置。

每路VDEC解码的内存消耗计算公式	场景举例	内存消耗（单位为MB）
6MB + 输入码流缓存大小 * 2 + (解码图像帧存大小 + 视频解码图像Tmv缓存大小) * (参考帧数量 + 解码后缓存图像帧数 + 1)	• 输入码流格式H.264 • 输入码流分辨率1080P • 输入码流缓存大小为4MB • 解码图像帧存大小为3MB • 视频解码图像Tmv缓存大小为0.5MB • 解码后缓存图像帧数2个	52.5MB（参考帧数量为8） 31.5MB（参考帧数量为2）
	• 输入码流格式H.265 • 输入码流分辨率1080P • 输入码流缓存大小为4MB • 解码图像帧存大小为3MB • 视频解码图像Tmv缓存大小为1MB • 解码后缓存图像帧数2个	58MB（参考帧数量为8） 34MB（参考帧数量为2）

2.10.5.5 [acl.himpi.vdec_create_chn](#)

c函数原型	hi_s32 hi_mpi_vdec_create_chn(hi_vdec_chn chn, const hi_vdec_chn_attr *attr);
Python函数	ret = acl.himpi.vdec_create_chn (chn, attr)
函数功能	根据设置的通道属性创建解码通道。 Atlas 200/300/500 推理产品 ，当前版本不支持该接口。 Atlas 训练系列产品 ，当前版本不支持该接口。

输入说明	chn: int, 解码通道号。 ● Atlas 推理系列产品, 该参数的取值范围: [0, 256), JPEGD功能和VDEC功能共用通道, 且通道总数最多256。 ● Atlas A2训练系列产品/Atlas 800I A2推理产品, 该参数的取值范围: [0, 256), JPEGD功能和VDEC功能共用通道, 且通道总数最多256, 其中, JPEGD解码通道总数最多256, VDEC解码通道总数最多32。 ● Atlas 200I/500 A2推理产品, 该参数的取值范围: [0, 128), JPEGD功能和VDEC功能共用通道, 且通道总数最多128。 说明 对于Atlas 推理系列产品, 昇腾虚拟化实例 (Ascend Virtual Instance) 场景下, VDEC通道和JPEGD通道总数 = (被分配的VDEC硬件单元+JPEGD硬件单元数量之和 / VDEC硬件单元+JPEGD硬件单元总数之和) * 256, 如果通道总数不为整数, 则向下取整。 对于Atlas A2训练系列产品/Atlas 800I A2推理产品, 昇腾虚拟化实例 (Ascend Virtual Instance) 场景下, VDEC通道 = (被分配的VDEC硬件单元 / VDEC硬件单元) * 32, 如果通道总数不为整数, 则向下取整。JPEGD的通道数不受算力影响, 但JPEGD+VDEC的总通道数最大256。 对于Atlas 200I/500 A2推理产品, 昇腾虚拟化实例 (Ascend Virtual Instance) 场景下, VDEC通道 = (被分配的VDEC硬件单元 / VDEC硬件单元) * 128, 如果通道总数不为整数, 则向下取整。JPEGD的通道数不随算力影响, 但JPEGD+VDEC的总通道数最大128。 昇腾虚拟化实例 (Ascend Virtual Instance) 的详细描述, 以及各场景下的算力资源规格请参见《昇腾虚拟化实例 (AVI) 用户指南》。 attr: dict, 解码通道属性字典, 具体请参见2.10.8.3.6 hi_vdec_chn_attr。 ● 如果参数“attr”为空, 会返回“HI_ERR_VDEC_NULL_PTR”的错误码。 ● 当通道属性“attr”中的值超过解码能力集时, 会返回“HI_ERR_VDEC_ILLEGAL_PARAM”的错误码。
返回值说明	ret: int, 错误码。 ● 返回0表示成功。 ● 返回非0表示失败, 参见2.10.9.1 公共返回码。

约束说明	• 单个Device上的通道号不能超出最大的通道号范围。 • 在创建解码通道之前必须保证通道未创建（或者已经销毁），否则会直接返回错误。 • 系统内存不足时会返回“HI_ERR_VDEC_NO_MEM”的错误码，可考虑扩展操作系统内存。 • 如果需要解码的H.264码流有B帧，或者需要解码的H.265码流支持时域运动矢量预测（sps_temporal_mvp_enabled_flag = 1），则创建通道时需要设置此通道支持时域运动矢量预测（temporal_mvp_en 设置为 1），还需要为其分配输出每一帧Tmv（Temporal Motion Vector）信息的VB（Vedio Buffer）块，该VB块的大小比图像VB块小很多，所需个数为RefFrameNum + 1，具体大小通过调用函数acl.himpi.vdec_get_tmv_buf_size获取，否则会导致解码出现花屏等错误。 • H.264、H.265解码每个解码通道所需VB个数至少为参考帧+显示帧+1，JPEG解码每个解码通道所需VB个数至少为显示帧+1。不同协议解码所需的图像VB块大小不同，具体大小通过调用函数acl.himpi.vdec_get_pic_buf_size获取。 • 如果 H.264解码不需要解码B帧，或者 H.265解码不需要解码支持时域运动矢量预测（sps_temporal_mvp_enabled_flag = 1）的码流，则创建通道时可设置此通道不支持时域运动矢量预测（“temporal_mvp_en” 设置为“0”），此种情况不输出Tmv信息，可以不用创建Tmv VB池，节省内存。 • 用户需要根据解码码流配置解码所需的帧存VB大小“frame_buf_size”和个数“frame_buf_cnt”，以及存放Tmv信息的VB大小“tmv_buf_size”，解码器内部按此配置创建相应的私有VB池。 • 只要帧存Buffer和Tmvbuffer足够大，解码通道能解码在最大最小分辨率范围内的任意分辨率码流。通道宽高当前只与码流Buffer、SCDbuffer（start code detector buffer）大小有关。
参考资源	接口调用流程，参见JPEG图片解码、VDEC视频解码。

2.10.5.6 acl.himpi.vdec_destroy_chn

c函数原型	hi_s32 hi_mpi_vdec_destroy_chn(hi_vdec_chn chn);
Python函数	ret = acl.himpi.vdec_destroy_chn(chn)
函数功能	销毁解码通道，以释放相关资源。 Atlas 200/300/500 推理产品 ，当前版本不支持该接口。 Atlas 训练系列产品 ，当前版本不支持该接口。

输入说明	chn: int, 解码通道号。 • Atlas 推理系列产品, 该参数的取值范围: [0, 256), JPEGD功能和VDEC功能共用通道, 且通道总数最多256。 • Atlas A2训练系列产品/Atlas 800I A2推理产品, 该参数的取值范围: [0, 256), JPEGD功能和VDEC功能共用通道, 且通道总数最多256, 其中, JPEGD解码通道总数最多256, VDEC解码通道总数最多32。 • Atlas 200I/500 A2推理产品, 该参数的取值范围: [0, 128), JPEGD功能和VDEC功能共用通道, 且通道总数最多128。
返回值说明	ret: int, 错误码。 • 返回0表示成功。 • 返回非0表示失败, 参见2.10.9.3 VDEC视频/JPEGD图像解码返回码。
约束说明	• 销毁前必须保证通道已创建, 否则会返回通道未创建的错误。 • 销毁前必须停止接收码流 (或者尚未开始接收码流), 否则返回错误码 “HI_ERR_VDEC_NOT_PERM” 。
参考资源	接口调用流程, 参见JPEGD图片解码、VDEC视频解码。

2.10.5.7 acl.himpi.vdec_get_chn_attr

c函数原型	hi_s32 hi_mpi_vdec_get_chn_attr(hi_vdec_chn chn, hi_vdec_chn_attr *attr);
Python函数	attr, ret= acl.himpi.vdec_get_chn_attr(chn)
函数功能	获取视频解码通道属性。 Atlas 200/300/500 推理产品, 当前版本不支持该接口。 Atlas 训练系列产品, 当前版本不支持该接口。
输入说明	chn: int, 解码通道号。 • Atlas 推理系列产品, 该参数的取值范围: [0, 256), JPEGD功能和VDEC功能共用通道, 且通道总数最多256。 • Atlas A2训练系列产品/Atlas 800I A2推理产品, 该参数的取值范围: [0, 256), JPEGD功能和VDEC功能共用通道, 且通道总数最多256, 其中, JPEGD解码通道总数最多256, VDEC解码通道总数最多32。 • Atlas 200I/500 A2推理产品, 该参数的取值范围: [0, 128), JPEGD功能和VDEC功能共用通道, 且通道总数最多128。

返回值说明	ret : int, 错误码。 - 返回0表示成功。 - 返回非0表示失败, 参见2.10.9.3 VDEC视频/JPEGD图像解码返回码。 attr : dict, 解码通道属性字典, 具体请参见 2.10.8.3.6 hi_vdec_chn_attr 。
约束说明	获取属性前必须保证通道已创建, 否则会返回通道未创建的错误码“HI_ERR_VDEC_UNEXIST”。

2.10.5.8 acl.himpi.vdec_set_chn_attr

c函数原型	hi_s32 hi_mpi_vdec_set_chn_attr(hi_vdec_chn chn, const hi_vdec_chn_attr *attr);
Python函数	ret = acl.himpi.vdec_set_chn_attr(chn, attr)
函数功能	设置解码通道属性。不支持该接口。
输入说明	chn : int, 解码通道号。 Atlas 推理系列产品, 该参数的取值范围: [0, 256), JPEGD功能和VDEC功能共用通道, 且通道总数最多256。 Atlas A2训练系列产品/Atlas 800I A2推理产品, 该参数的取值范围: [0, 256), JPEGD功能和VDEC功能共用通道, 且通道总数最多256, 其中, JPEGD解码通道总数最多256, VDEC解码通道总数最多32。 Atlas 200I/500 A2推理产品, 该参数的取值范围: [0, 128), JPEGD功能和VDEC功能共用通道, 且通道总数最多128。 attr : dict, 解码通道属性字典, 具体请参见 2.10.8.3.6 hi_vdec_chn_attr 。
返回值说明	ret : int, 错误码。 - 返回0表示成功。 - 返回非0表示失败, 参见2.10.9.3 VDEC视频/JPEGD图像解码返回码。
约束说明	- 获取属性前必须保证通道已创建, 否则会返回通道未创建的错误码“HI_ERR_VDEC_UNEXIST”。 - 可以改变的属性有“type”、“mode”、“ref_frame_num”, 其它属性不能改变。其中, “type”只支持“HI_PT_H264”、“HI_PT_H265”。 - 解码器内部会在改变属性时会自动复位解码通道。

2.10.5.9 acl.himpi.vdec_start_rcv_stream

c函数原型	hi_s32 hi_mpi_vdec_start_rcv_stream(hi_vdec_chn chn);

Python函数	ret = acl.himpi.vdec_start_recv_stream(chn)
函数功能	创建通道之后，启动解码之前，解码器开始接收用户发送的码流。 Atlas 200/300/500 推理产品，当前版本不支持该接口。 Atlas 训练系列产品，当前版本不支持该接口。
输入说明	chn : int，解码通道号。 ● Atlas 推理系列产品，该参数的取值范围：[0, 256)，JPEGD功能和VDEC功能共用通道，且通道总数最多256。 ● Atlas A2训练系列产品/Atlas 800I A2推理产品，该参数的取值范围：[0, 256)，JPEGD功能和VDEC功能共用通道，且通道总数最多256，其中，JPEGD解码通道总数最多256，VDEC解码通道总数最多32。 ● Atlas 200I/500 A2推理产品，该参数的取值范围：[0, 128)，JPEGD功能和VDEC功能共用通道，且通道总数最多128。
返回值说明	ret : int，错误码。 ● 返回0表示成功。 ● 返回非0表示失败，参见2.10.9.3 VDEC视频/JPEGD图像解码返回码。
约束说明	● 启动接收码流前必须保证通道已创建，否则会返回通道未创建的错误码“HI_ERR_VDEC_UNEXIST”。 ● 启动接收码流之后，才能调用acl.himpi.vdec_send_stream发送码流。 ● 重复调用启动接收码流接口时，返回成功。
参考资源	接口调用流程，参见JPEGD图片解码、VDEC视频解码。

2.10.5.10 acl.himpi.vdec_stop_recv_stream

c函数原型	hi_s32 hi_mpi_vdec_stop_recv_stream(hi_vdec_chn chn);
Python函数	ret = acl.himpi.vdec_stop_recv_stream(chn)
函数功能	码流发送结束之后，解码器停止接收用户发送的码流，在销毁解码通道之前必须调用此接口。 Atlas 200/300/500 推理产品 ，当前版本不支持该接口。 Atlas 训练系列产品 ，当前版本不支持该接口。

输入说明	chn: int, 解码通道号。 • Atlas 推理系列产品, 该参数的取值范围: [0, 256), JPEGD功能和VDEC功能共用通道, 且通道总数最多256。 • Atlas A2训练系列产品/Atlas 800I A2推理产品, 该参数的取值范围: [0, 256), JPEGD功能和VDEC功能共用通道, 且通道总数最多256, 其中, JPEGD解码通道总数最多256, VDEC解码通道总数最多32。 • Atlas 200I/500 A2推理产品, 该参数的取值范围: [0, 128), JPEGD功能和VDEC功能共用通道, 且通道总数最多128。
返回值说明	ret: int, 错误码。 • 返回0表示成功。 • 返回非0表示失败, 参见2.10.9.3 VDEC视频/JPEGD图像解码返回码。
约束说明	• 停止接收码流前必须保证通道已创建, 否则会返回通道未创建的错误码“HI_ERR_VDEC_UNEXIST”。 • 调用此接口后, 调用发送码流接口acl.himpi.vdec_send_stream会返回失败。 • 重复调用停止接收码流接口时, 返回成功。
参考资源	接口调用流程, 参见JPEGD图片解码、VDEC视频解码。

2.10.5.11 acl.himpi.vdec_query_status

c函数原型	hi_s32 hi_mpi_vdec_query_status(hi_vdec_chn chn, hi_vdec_chn_status *status);
Python函数	status, ret = acl.himpi.vdec_query_status(chn)
函数功能	解码过程中通过此接口来查询解码通道状态。 Atlas 200/300/500 推理产品, 当前版本不支持该接口。 Atlas 训练系列产品, 当前版本不支持该接口。
输入说明	chn: int, 解码通道号。 • Atlas 推理系列产品, 该参数的取值范围: [0, 256), JPEGD功能和VDEC功能共用通道, 且通道总数最多256。 • Atlas A2训练系列产品/Atlas 800I A2推理产品, 该参数的取值范围: [0, 256), JPEGD功能和VDEC功能共用通道, 且通道总数最多256, 其中, JPEGD解码通道总数最多256, VDEC解码通道总数最多32。 • Atlas 200I/500 A2推理产品, 该参数的取值范围: [0, 128), JPEGD功能和VDEC功能共用通道, 且通道总数最多128。

返回值说明	ret: int, 错误码。 • 返回0表示成功。 • 返回非0表示失败, 参见2.10.9.3 VDEC视频/JPEGD图像解码返回码。 status: dict, 获取的通道状态, 具体请参见2.10.8.3.16 hi_vdec_chn_status。
约束说明	查询通道状态前必须保证通道已创建, 否则会返回通道未创建的错误码“HI_ERR_VDEC_UNEXIST”。

2.10.5.12 acl.himpi.vdec_reset_chn

c函数原型	hi_s32 hi_mpi_vdec_reset_chn(hi_vdec_chn chn);
Python函数	ret = acl.himpi.vdec_reset_chn(chn)
函数功能	解码通道长时间没有反应的情况下或者解码失败等情况下, 可以调用此接口复位通道, 尝试恢复。 Atlas 200/300/500 推理产品, 当前版本不支持该接口。 Atlas 训练系列产品, 当前版本不支持该接口。
输入说明	chn: int, 解码通道号。 • Atlas 推理系列产品, 该参数的取值范围: [0, 256), JPEGD功能和VDEC功能共用通道, 且通道总数最多256。 • Atlas A2训练系列产品/Atlas 800I A2推理产品, 该参数的取值范围: [0, 256), JPEGD功能和VDEC功能共用通道, 且通道总数最多256, 其中, JPEGD解码通道总数最多256, VDEC解码通道总数最多32。 • Atlas 200I/500 A2推理产品, 该参数的取值范围: [0, 128), JPEGD功能和VDEC功能共用通道, 且通道总数最多128。
返回值说明	ret: int, 错误码。 • 返回0表示成功。 • 返回非0表示失败, 参见2.10.9.3 VDEC视频/JPEGD图像解码返回码。
约束说明	• 复位前必须保证通道已创建, 否则会返回错误码“HI_ERR_VDEC_UNEXIST”。 • 复位前必须停止接收码流, 否则返回错误码“HI_ERR_VDEC_NOT_PERM”。

2.10.5.13 acl.himpi.vdec_set_chn_param

c函数原型	hi_s32 hi_mpi_vdec_set_chn_param(hi_vdec_chn chn, const hi_vdec_chn_param *chn_param);

Python函数	<code>ret = acl.himpi.vdec_set_chn_param(chn, chn_param)</code>
函数功能	改变通道参数的时候调用此接口，设置解码通道参数。 Atlas 200/300/500 推理产品 ，当前版本不支持该接口。 Atlas 训练系列产品 ，当前版本不支持该接口。
输入说明	chn : int，解码通道号。 • Atlas 推理系列产品，该参数的取值范围：[0, 256)，JPEGD功能和VDEC功能共用通道，且通道总数最多256。 • Atlas A2训练系列产品/Atlas 800I A2推理产品，该参数的取值范围：[0, 256)，JPEGD功能和VDEC功能共用通道，且通道总数最多256，其中，JPEGD解码通道总数最多256，VDEC解码通道总数最多32。 • Atlas 200I/500 A2推理产品，该参数的取值范围：[0, 128)，JPEGD功能和VDEC功能共用通道，且通道总数最多128。 chn_param : dict，解码通道参数字典，具体请参见 2.10.8.3.14 hi_vdec_chn_param 。
返回值说明	ret : int，错误码。 • 返回0表示成功。 • 返回非0表示失败，参见2.10.9.3 VDEC视频/JPEGD图像解码返回码。
约束说明	• 必须保证通道已创建，否则会返回错误码“HI_ERR_VDEC_UNEXIST”。 • 该接口可设置通道的一些高级属性，创建通道后有默认值，默认值见数据类型hi_vdec_chn_param的说明。 • 各项参数如超过合法范围，会返回“HI_ERR_VDEC_ILLEGAL_PARAM”的错误码，各参数范围见数据类型hi_vdec_chn_param的说明。 • 调用acl.himpi.vdec_create_chn接口创建通道时设置按帧发送码流（目前仅支持按帧发送），同时，调用本接口设置解码序输出码流，即chn_param["video_param"]["out_order"] = HI_VIDEO_OUT_ORDER_DEC，可以达到快速解码和快速输出的目的，但这种场景不支持解码含有B帧的码流。
参考资源	接口调用流程，参见JPEGD图片解码、VDEC视频解码。

2.10.5.14 acl.himpi.vdec_get_chn_param

c函数原型	<code>hi_s32 hi_mpi_vdec_get_chn_param(hi_vdec_chn chn, hi_vdec_chn_param *chn_param);</code>
Python函数	<code>chn_param, ret = acl.himpi.vdec_get_chn_param(chn)</code>

函数功能	修改通道参数前可先调用此接口获取通道参数。 Atlas 200/300/500 推理产品 ，当前版本不支持该接口。 Atlas 训练系列产品 ，当前版本不支持该接口。
输入说明	chn : int，解码通道号。 • Atlas 推理系列产品，该参数的取值范围：[0, 256)，JPEG功能 and VDEC功能共用通道，且通道总数最多256。 • Atlas A2训练系列产品/Atlas 800I A2推理产品，该参数的取值范围：[0, 256)，JPEG功能 and VDEC功能共用通道，且通道总数最多256，其中，JPEG解码通道总数最多256，VDEC解码通道总数最多32。 • Atlas 200I/500 A2推理产品，该参数的取值范围：[0, 128)，JPEG功能 and VDEC功能共用通道，且通道总数最多128。
返回值说明	chn_param : dict，解码通道参数字典，具体请参见 2.10.8.3.14 hi_vdec_chn_param 。 ret : int，错误码。 • 返回0表示成功。 • 返回非0表示失败，参见2.10.9.3 VDEC视频/JPEG图像解码返回码。
约束说明	必须保证通道已创建，否则会返回错误码“HI_ERR_VDEC_UNEXIST”。
参考资源	接口调用流程，参见JPEG图片解码、VDEC视频解码。

2.10.5.15 acl.himpi.vdec_set_protocol_param

c函数原型	hi_s32 hi_mpi_vdec_set_protocol_param(hi_vdec_chn chn, const hi_vdec_protocol_param* protocol_param);
Python函数	ret = acl.himpi.vdec_set_protocol_param(chn, protocol_param)
函数功能	设置协议相关的内存分配通道参数，如slice、pps（Picture Parameter Set）、sps（Sequence Parameter Set）等。 当前版本不支持该接口。

输入说明	chn: int, 解码通道号。 • Atlas 推理系列产品, 该参数的取值范围: [0, 256), JPEGD功能和VDEC功能共用通道, 且通道总数最多256。 • Atlas A2训练系列产品/Atlas 800I A2推理产品, 该参数的取值范围: [0, 256), JPEGD功能和VDEC功能共用通道, 且通道总数最多256, 其中, JPEGD解码通道总数最多256, VDEC解码通道总数最多32。 • Atlas 200I/500 A2推理产品, 该参数的取值范围: [0, 128), JPEGD功能和VDEC功能共用通道, 且通道总数最多128。 protocol_param: dict, 解码通道参数字典, 具体请参见2.10.8.3.9 hi_vdec_protocol_param。
返回值说明	ret: int, 错误码。 • 返回0表示成功。 • 返回非0表示失败, 参见2.10.9.3 VDEC视频/JPEGD图像解码返回码。
约束说明	• 必须保证通道已创建, 否则会返回错误码“HI_ERR_VDEC_UNEXIST”。 • 设置协议参数前必须先停止接收码流, 否则返回错误码“HI_ERR_VDEC_NOT_PERM”。 • 切换协议参数时, 解码器内部会复位解码通道。 • 协议参数所需的内存是解码器内部根据当前码流实际vps (Video Parameter Set) /sps/pps/slice个数使用vmalloc动态分配的, 最大上限不超过此接口设置的最大vps/sps/pps/slice个数。

2.10.5.16 acl.himpi.vdec_get_protocol_param

c函数原型	hi_s32 hi_mpi_vdec_get_protocol_param(hi_vdec_chn chn, hi_vdec_protocol_param* protocol_param);
Python函数	protocol_param, ret = acl.himpi.vdec_get_protocol_param(chn)
函数功能	获取协议相关的内存分配通道参数, 如slice、pps (Picture Parameter Set)、sps (Sequence Parameter Set) 等。 Atlas 200/300/500 推理产品, 当前版本不支持该接口。 Atlas 训练系列产品, 当前版本不支持该接口。

输入说明	chn: int, 解码通道号。 • Atlas 推理系列产品, 该参数的取值范围: [0, 256), JPEGD功能和VDEC功能共用通道, 且通道总数最多256。 • Atlas A2训练系列产品/Atlas 800I A2推理产品, 该参数的取值范围: [0, 256), JPEGD功能和VDEC功能共用通道, 且通道总数最多256, 其中, JPEGD解码通道总数最多256, VDEC解码通道总数最多32。 • Atlas 200I/500 A2推理产品, 该参数的取值范围: [0, 128), JPEGD功能和VDEC功能共用通道, 且通道总数最多128。
返回值说明	protocol_param: dict, 解码通道参数字典, 具体请参见2.10.8.3.9 hi_vdec_protocol_param。 ret: int, 错误码。 • 返回0表示成功。 • 返回非0表示失败, 参见2.10.9.3 VDEC视频/JPEGD图像解码返回码。
约束说明	必须保证通道已创建, 否则会返回错误码“HI_ERR_VDEC_UNEXIST”。

2.10.5.17 acl.himpi.vdec_send_stream

c函数原型	hi_s32 hi_mpi_vdec_send_stream(hi_vdec_chn chn, const hi_vdec_stream *stream, hi_vdec_pic_info *vdec_pic_info, hi_s32 milli_sec);
Python函数	ret = acl.himpi.vdec_send_stream(chn, stream, vdec_pic_info, milli_sec)
函数功能	解码前, 向解码通道发送码流数据及存放解码结果的Buffer。 Atlas 200/300/500 推理产品 , 当前版本不支持该接口。 Atlas 训练系列产品 , 当前版本不支持该接口。

输入说明	chn: int, 解码通道号。 ● Atlas 推理系列产品, 该参数的取值范围: [0, 256), JPEGD功能和VDEC功能共用通道, 且通道总数最多256。 ● Atlas A2训练系列产品/Atlas 800I A2推理产品, 该参数的取值范围: [0, 256), JPEGD功能和VDEC功能共用通道, 且通道总数最多256, 其中, JPEGD解码通道总数最多256, VDEC解码通道总数最多32。 ● Atlas 200I/500 A2推理产品, 该参数的取值范围: [0, 128), JPEGD功能和VDEC功能共用通道, 且通道总数最多128。 stream: dict, 输入码流信息。该字典内的“addr”参数配置的地址为Device上的内存地址, 具体请参见2.10.8.3.17 hi_vdec_stream。 vdec_pic_info: dict, 输出图片信息字典, 具体请参见2.10.8.3.4 hi_vdec_pic_info。 milli_sec: int, 超时时间, 单位是毫秒。 ● “-1”: 阻塞方式。 ● “0”: 非阻塞方式。 ● >0: 超时方式, 配置具体的超时时间。超时时间受操作系统影响, 一般偏差在操作系统的片内, 例如, 操作系统的片为4ms, 用户设置的“milli_sec”参数值为“1”, 则实际的超时时间在1ms到5ms范围内。在CPU负载高场景下, 超时时间仍可能存在波动。
返回值说明	ret: int, 错误码。 ● 返回0表示成功。 ● 返回非0表示失败, 参见2.10.9.3 VDEC视频/JPEGD图像解码返回码。

约束说明	• 发送数据前必须保证已经调用acl.himpi.vdec_start_rcv_stream接口启动接收码流，否则直接返回该操作不允许的错误码“HI_ERR_VDEC_NOT_PERM”。如果在发送数据过程中停止接收码流，就会立刻返回“HI_ERR_VDEC_NOT_PERM”。 • 发送数据前必须保证通道已经被创建，否则直接返回通道未创建的错误码“HI_ERR_VDEC_UNEXIST”。如果在发送码流过程中复位通道或者销毁通道，就会立刻返回错误码“HI_ERR_VDEC_UNEXIST”。 • 发送码流时需要按照创建解码通道时设置的发送方式（仅支持按帧发送）进行发送。按帧发送时，调用此接口一次，必须发送完整的一帧码流，否则，解码会出现错误。 • 不能发送“end_of_stream”为0的空码流包（码流长度为0或码流地址为空），否则返回错误码。发送“end_of_stream”为1的空码流包时，解码器会把所有码流全部解完并输出全部图像。除此之外，其它情况应该把“end_of_stream”置为0。 • 当码流buffer为空且装不下当前包码流时，会返回参数超出范围的错误码“HI_ERR_VDEC_ILLEGAL_PARAM”。 • 以非阻塞方式发送码流，如果码流缓冲区已满，会立刻返回错误码“HI_ERR_VDEC_BUF_FULL”。 • 以超时方式发送码流，到达设定的超时时间还不能成功发送码流会返回错误码“HI_ERR_VDEC_BUF_FULL”。 • 视频解码时支持设置是否将解码结果写入输出内存中，hi_vdec_stream字典中将“need_display”设置为“0”时，此帧码流的解码结果将不会写入到hi_vdec_pic_info字典中配置的输出内存中，此时允许将输出内存地址配置为“None”。acl.himpi.vdec_get_frame接口仍然能获取到字典hi_video_frame_info和hi_vdec_stream的信息，但是此时输出内存中无解码结果数据，需要用户释放输出内存（如果用户传入的输出内存地址是“None”，则无需释放）。 • 解码时，输入输出内存均需要在调用acl.himpi.vdec_get_frame接口获取结果之后才能进行释放。 • 视频解码，输入输出内存要求、输入码流格式、输出图片格式请参见2.10.5.3 VDEC功能及约束说明。 • 图像解码，输入输出内存要求、输入图片格式、输出图片格式请参见2.10.6.1 JPEG功能及约束说明。
参考资源	接口调用流程，参见JPEGD图片解码、VDEC视频解码。

2.10.5.18 acl.himpi.vdec_get_frame

c函数原型	<pre>hi_s32 hi_mpi_vdec_get_frame(hi_vdec_chn chn, hi_video_frame_info *frame_info, hi_vdec_supplement_info *supplement, hi_vdec_stream *stream, hi_s32 milli_sec);</pre>
Python函数	<pre>frame_info, supplement, stream, ret = acl.himpi.vdec_get_frame(chn, milli_sec)</pre>

函数功能	解码后，获取视频解码通道的解码图像及输入Stream。 JPEGD图像解码场景下，在调用本接口时，如果硬件解码失败，则会在本接口内部触发软件解码。 Atlas 200/300/500 推理产品，当前版本不支持该接口。 Atlas 训练系列产品，当前版本不支持该接口。
输入说明	chn: int，解码通道号。 • Atlas 推理系列产品，该参数的取值范围：[0, 256)，JPEGD功能和VDEC功能共用通道，且通道总数最多256。 • Atlas A2训练系列产品/Atlas 800I A2推理产品，该参数的取值范围：[0, 256)，JPEGD功能和VDEC功能共用通道，且通道总数最多256，其中，JPEGD解码通道总数最多256，VDEC解码通道总数最多32。 • Atlas 200I/500 A2推理产品，该参数的取值范围：[0, 128)，JPEGD功能和VDEC功能共用通道，且通道总数最多128。 milli_sec: int，超时时间，单位是毫秒。 • “-1”：阻塞方式。 • “0”：非阻塞方式。 • >0：超时方式，配置具体的超时时间。超时时间受操作系统影响，一般偏差在操作系统的一个时间片内，例如，操作系统的 一个时间片为4ms，用户设置的“milli_sec”参数值为“1”，则实际的超时时间在1ms到5ms范围内。在CPU负载高场景下，超时时间仍可能存在波动。 说明 JPEGD图像解码场景下，此处配置的超时时间仅指获取硬件解码结果的等待时间，不包含软件解码的时间。

返回值说明	ret: int, 错误码。 • 返回0表示成功，获取解码数据成功后，需调用acl.himpi.vdec_release_frame接口释放解码相关资源。本接口虽返回成功，但有可能是解码失败的数据，解码成功或失败，需要查看hi_video_frame字典（hi_video_frame_info字典的成员）中的“frame_flag”参数。 • 返回非0表示失败，参见2.10.9.3 VDEC视频/JPEGD图像解码返回码。 说明 • 如果在解码图像时未创建通道，则会返回错误码“HI_ERR_VDEC_UNEXIST”。 • 如果在获取图像的过程中销毁通道，则会返回错误码“HI_ERR_VDEC_UNEXIST”。 • 如果在获取解码图像的过程中复位通道，则会返回错误码“HI_ERR_VDEC_UNEXIST”。 • 以非阻塞方式获取解码图像，如果缓冲区内无图像，则会返回错误码“HI_ERR_VDEC_BUF_EMPTY”。 • 以超时方式获取解码图像，到达设定的超时时间还不能获取到硬件解码结果则会返回错误码“HI_ERR_VDEC_BUF_EMPTY”，此时则可增大超时时间，或循环调用本接口等待解码结果。 stream: dict, 获取已完成解码的输入码流信息字典，具体请参见2.10.8.3.17 hi_vdec_stream。 frame_info: dict, 获取已经完成解码的图像信息。解码后的数据存放在Device内存中，具体请参见2.10.8.2.9 hi_video_frame_info。 supplement: dict, 获取的解码图像补充信息。预留参数，暂未使用。
约束说明	• 向VDEC获取解码结果，输入Buffer和输出Buffer一起获取。 • JPEGD图片解码时，默认对于jpeg(420)、jpeg(422)、jpeg(440)格式的输入图片，如果输入图片的宽高为奇数，JPEGD解码输出图片的宽高为向下2对齐；默认对于jpeg(444)、jpeg(400)格式的输入图片，JPEGD解码输出图片的宽高与输入图片保持一致。若需调整输出图片的宽、高对齐模式，可调用acl.himpi.vdec_set_jpegd_precision_mode接口。
参考资源	接口调用流程，参见JPEGD图片解码、VDEC视频解码。

2.10.5.19 acl.himpi.vdec_release_frame

c函数原型	hi_s32 hi_mpi_vdec_release_frame(hi_vdec_chn chn, const hi_video_frame_info *frame_info);
Python函数	ret = acl.himpi.vdec_release_frame(chn, frame_info)
函数功能	解码之后，释放资源。 Atlas 200/300/500 推理产品，当前版本不支持该接口。 Atlas 训练系列产品，当前版本不支持该接口。

输入说明	chn: int, 解码通道号。 • Atlas 推理系列产品, 该参数的取值范围: [0, 256), JPEGD功能和VDEC功能共用通道, 且通道总数最多256。 • Atlas A2训练系列产品/Atlas 800I A2推理产品, 该参数的取值范围: [0, 256), JPEGD功能和VDEC功能共用通道, 且通道总数最多256, 其中, JPEGD解码通道总数最多256, VDEC解码通道总数最多32。 • Atlas 200I/500 A2推理产品, 该参数的取值范围: [0, 128), JPEGD功能和VDEC功能共用通道, 且通道总数最多128。 frame_info: dict, 解码后的图像信息字典, 调用2.10.5.18 acl.himpi.vdec_get_frame接口获取, 具体请参见2.10.8.2.9 hi_video_frame_info。
返回值说明	ret: int, 错误码。 • 返回0表示成功。 • 返回非0表示失败, 参见2.10.9.3 VDEC视频/JPEGD图像解码返回码。 - 如果在解码图像时未创建通道, 则会返回“HI_ERR_VDEC_UNEXIST”的错误码。 - 如果在获取图像的过程中销毁通道, 则会返回“HI_ERR_VDEC_UNEXIST”的错误码。 - 如果在获取解码图像的过程中复位通道, 则会返回“HI_ERR_VDEC_UNEXIST”的错误码。 - 以非阻塞方式获取解码图像, 如果缓冲区内无图像, 则会返回“HI_ERR_VDEC_BUF_EMPTY”的错误码。 - 以超时方式获取解码图像, 到达设定的超时时间还不能获取到硬件解码结果则会返回“HI_ERR_VDEC_BUF_EMPTY”的错误码, 此时则可增大超时时间, 或循环调用本接口等待解码结果。
约束说明	• 此接口需与acl.himpi.vdec_get_frame配对使用, 获取的数据应当在使用完之后立即释放, 如果不及时释放, 会导致解码过程阻塞等待资源。 • 释放的数据必须是调用acl.himpi.vdec_get_frame接口从该通道获取的数据, 不得对数据信息字典进行任何修改。
参考资源	接口调用流程, 参见JPEGD图片解码、VDEC视频解码。

2.10.5.20 acl.himpi.vdec_get_fd

c函数原型	hi_s32 hi_mpi_vdec_get_fd(hi_vdec_chn chn);
Python函数	ret = acl.himpi.vdec_get_fd(chn)
函数功能	获取视频解码通道的设备文件句柄。 Atlas 200/300/500 推理产品, 当前版本不支持该接口。 Atlas 训练系列产品, 当前版本不支持该接口。

输入说明	chn : int, 解码通道号。 • Atlas 推理系列产品, 该参数的取值范围: [0, 256), JPEGD功能和VDEC功能共用通道, 且通道总数最多256。 • Atlas A2训练系列产品/Atlas 800I A2推理产品, 该参数的取值范围: [0, 256), JPEGD功能和VDEC功能共用通道, 且通道总数最多256, 其中, JPEGD解码通道总数最多256, VDEC解码通道总数最多32。 • Atlas 200I/500 A2推理产品, 该参数的取值范围: [0, 128), JPEGD功能和VDEC功能共用通道, 且通道总数最多128。
返回值说明	ret : int, 错误码。 • 0或正数值: 有效返回值。 • 非正数值: 无效返回值。
约束说明	在进程退出时会自动关闭文件句柄, 回收资源。

2.10.5.21 acl.himpi.vdec_close_fd

c函数原型	hi_s32 hi_mpi_vdec_close_fd(hi_vdec_chn chn);
Python函数	ret = acl.himpi.vdec_close_fd(chn)
函数功能	关闭视频解码的设备文件句柄。 当前版本不支持该接口。
输入说明	chn : int, 解码通道号。 • Atlas 推理系列产品, 该参数的取值范围: [0, 256), JPEGD功能和VDEC功能共用通道, 且通道总数最多256。 • Atlas A2训练系列产品/Atlas 800I A2推理产品, 该参数的取值范围: [0, 256), JPEGD功能和VDEC功能共用通道, 且通道总数最多256, 其中, JPEGD解码通道总数最多256, VDEC解码通道总数最多32。 • Atlas 200I/500 A2推理产品, 该参数的取值范围: [0, 128), JPEGD功能和VDEC功能共用通道, 且通道总数最多128。
返回值说明	ret : int, 错误码。 • 返回0表示成功。 • 返回非0表示失败, 参见2.10.9.3 VDEC视频/JPEGD图像解码返回码。

2.10.5.22 acl.himpi.vdec_get_pic_buf_size

c函数原型	hi_u32 hi_vdec_get_pic_buf_size(hi_payload_type type, hi_pic_buf_attr *buf_attr);
Python函数	ret = acl.himpi.vdec_get_pic_buf_size(type, buf_attr)

函数功能	获取解码图像需要的Buffer大小。 用户在以下场景可使用该函数：计算视频解码需要的内部缓冲区的大小，在初始化的时候配置对应的通道参数、计算图像解码的输出内存。 Atlas 200/300/500 推理产品，当前版本不支持该接口。 Atlas 训练系列产品，当前版本不支持该接口。
输入说明	type: int，解码视频类型，当前支持“HI_PT_H264”、“HI_PT_H265”、“HI_PT_JPEG”，具体请参见2.10.8.2.3 hi_payload_type。 buf_attr: dict，图片参数字典，具体请参见2.10.8.2.15 hi_pic_buf_attr。
返回值说明	ret: int，图像存储需要的图像大小。 - 非0为需要存储的图像大小。 - 若解码视频类型为“HI_PT_JPEG”（表示JPEGD解码）时，“pixel_format”传入不支持的格式或者“HI_PIXEL_FORMAT_UNKNOWN”，返回值为“0”。
约束说明	提供给用户来计算输出图片缓冲区大小。
参考资源	接口调用流程及示例，参见VDEC视频解码。

2.10.5.23 acl.himpi.vdec_get_tmv_buf_size

c函数原型	hi_u32 hi_vdec_get_tmv_buf_size(hi_payload_type type, hi_u32 width, hi_u32 height);
Python函数	ret = acl.himpi.vdec_get_tmv_buf_size(type, width, height)
函数功能	取解码图像需要的矢量预测缓冲区的大小。 Atlas 200/300/500 推理产品，当前版本不支持该接口。 Atlas 训练系列产品，当前版本不支持该接口。
输入说明	type: int，解码视频类型，具体请参见2.10.8.2.3 hi_payload_type。 width: int，解码图像宽。 height: int，解码图像高。
返回值说明	图像存储需要矢量预测缓冲区大小。
约束说明	提供给用户来计算矢量预测缓冲区大小，在创建通道的时候初始化对应的参数。
参考资源	接口调用流程及示例，参见VDEC视频解码。

2.10.5.24 acl.himpi.vdec_set_jpegd_precision_mode

c函数原型	hi_u32 hi_mpi_vdec_set_jpegd_precision_mode(hi_vdec_chn chn, const hi_jpegd_precision_mode mode)
Python函数	ret = acl.himpi.vdec_set_jpegd_precision_mode(chn, mode)
函数功能	设置JPEGD解码输出图片的宽、高对齐模式。 Atlas 200/300/500 推理产品 ，当前版本不支持该接口。 Atlas 训练系列产品 ，当前版本不支持该接口。
输入说明	chn : int, 解码通道号。 • Atlas 推理系列产品，该参数的取值范围：[0, 256)，JPEGD功能和VDEC功能共用通道，且通道总数最多256。 • Atlas A2训练系列产品/Atlas 800I A2推理产品，该参数的取值范围：[0, 256)，JPEGD功能和VDEC功能共用通道，且通道总数最多256，其中，JPEGD解码通道总数最多256，VDEC解码通道总数最多32。 • Atlas 200I/500 A2推理产品，该参数的取值范围：[0, 128)，JPEGD功能和VDEC功能共用通道，且通道总数最多128。 mode : int, 设置JPEGD解码输出图片的宽、高对齐模式，具体请参见 2.10.8.3.20 hi_jpegd_precision_mode 。
返回值说明	ret : int, 错误码。 • 0: 成功 • 非0: 失败，参见2.10.9.3 VDEC视频/JPEGD图像解码返回码
约束说明	• 调用本接口设置对齐模式前，必须保证通道已创建，否则会返回通道未创建的错误码“HI_ERR_VDEC_UNEXIST”。 • 调用本接口设置对齐模式仅对JPEGD解码通道有效，其它类型通道返回操作不支持。 • 在调用acl.himpi.vdec_start_rcv_stream接口之前调用本接口设置对齐模式，在调用acl.himpi.vdec_start_rcv_stream接口之后调用本接口，系统会返回操作不允许的错误提示。

2.10.5.25 acl.himpi.vdec_get_jpegd_output_info

c函数原型	hi_s32 hi_mpi_vdec_get_jpegd_output_info(const hi_vdec_stream *stream, const hi_pixel_format output_format, hi_img_info *img_info)
Python函数	img_info, ret = acl.himpi.vdec_get_jpegd_output_info(stream, output_format)

函数功能	图片解码场景下，根据输入码流，获取按源图格式或指定图片格式输出时的图片宽、高、宽Stride、高Stride、解码后占用的内存大小、图片格式等信息。 该接口会检查图片格式是否支持解码，如果遇到不支持的码流则该接口会返回“HI_ERR_VDEC_NOT_SUPPORT”错误码，该错误码的详细描述请参见2.10.9.1 公共返回码。 Atlas 200/300/500 推理产品，不支持该接口。 Atlas 训练系列产品，不支持该接口。
输入说明	stream: dict，输入码流信息字典。具体参见2.10.8.3.17 hi_vdec_stream。 Atlas 推理系列产品，标准形态下，在Host上调用本接口时，该字典内的addr参数配置的地址必须是Host上的内存地址。 Atlas A2训练系列产品/Atlas 800I A2推理产品，在Host上调用本接口时，该字典内的addr参数配置的地址必须是Host上的内存地址。 Atlas 200I/500 A2推理产品，EP模式下，在Host上调用本接口时，该字典内的addr参数配置的地址必须是Host上的内存地址。 Atlas 200I/500 A2推理产品，RC模式下，在Device上调用本接口时，该字典内的addr参数配置的地址必须是Device上的内存地址。 output_format: int，输出图片格式。 若按源图格式获取相关信息，则将该参数值设置为HI_PIXEL_FORMAT_UNKNOWN = 0。若按指定格式获取相关信息，则将该参数值设置为对应的图片格式，支持的图片格式请参见图片格式、宽高对齐、内存约束。
返回值说明	img_info: dict，输出图片信息字典，包含图片宽、高、宽Stride、高Stride、解码后占用的内存大小、图片格式等信息。其中，输出图片格式，通过“pixel_format”参数返回。具体参见2.10.8.2.14 hi_img_info。 ret: int，错误码。 0: 成功。 - 非0: 失败，参见2.10.9.3 VDEC视频/JPEGD图像解码返回码。

2.10.6 VENC 视频/JPEGE 图像编码接口

2.10.6.1 JPEG 功能及约束说明

Atlas 200/300/500 推理产品上，不支持本章中的JPEG功能。

Atlas 训练系列产品上，不支持本章中的JPEG功能。

功能说明

JPEGE (JPEG Encoder) 将YUV格式图片编码成JPEG压缩格式的图片文件，例如*.jpg。

图片分辨率约束

- 输入图片分辨率

版本	分辨率范围
Atlas 推理系列产品 Atlas A2训练系列产品/Atlas 800I A2推理产品	最大分辨率：8192 * 8192，最小分辨率：32 * 32。
Atlas 200I/500 A2推理产品	最大分辨率：16384 * 16384，最小分辨率：32 * 32。
- 输出图片分辨率

JPEGE只对图片编码，不会改变图片分辨率，因此输出与输入的图片分辨率保持一致。

内存申请/释放接口说明

输入内存的大小就是指实际的输入码流所占用的大小，输出内存的大小参见表2-56中的计算公式。

版本	输入内存申请/释放接口	输出内存申请/释放接口
Atlas 推理系列产品	JPEGE图片编码功能时，调用 acl.himpi.dvpp_malloc \ acl.himpi.dvpp_free 接口申请\释放Device上的输入内存，这部分内存的生命周期由用户自行管理。	支持以下两种方式： - 不需要用户管理、由DVPP内部管理输出内存时，调用acl.himpi.venc_send_frame接口发送原始图像进行图像编码。 在调用acl.himpi.venc_create_chn接口创建通道时，必须正确设置 <code>hi_venc_chn_attr["venc_attr"]["buf_size"]</code>属性值（参数描述请参见hi_venc_attr）。 - 该方式下，相比由用户管理内存，输出结果数据的JPEGE头中不存在COM注释字段，数据长度会短一点，但需要用户从DVPP返回的内存中拷贝输出结果数据到指定内存。 - 由用户自行管理输出内存、管理内存的生命周期，调用acl.himpi.venc_send_jpege_frame接口发送原始图像进行图像编码。 在调用acl.himpi.venc_create_chn接口创建通道时，需将 <code>hi_venc_chn_attr["venc_attr"]["buf_size"]</code>属性值设置为0（参数描述请参见hi_venc_attr），然后调用acl.himpi.venc_get_jpege_predicted_size接口预估输出内存大小，调用acl.himpi.dvpp_malloc\acl.himpi.dvpp_free接口申请\释放输出内存。 - 该方式下，直接在调用acl.himpi.venc_send_jpege_frame接口时，设置输出内存地址，输出

版本	输入内存申请/释放接口	输出内存申请/释放接口
		结果数据直接存放到用户设置的内存中，相比由系统管理内存的方式，用户可减少一次“从DVPP返回的内存中拷贝输出结果数据到指定内存”的操作，但输出结果数据的JPEG头中可能会存在COM注释字段（字段长度范围4~19Byte），数据长度会长一点。

版本	输入内存申请/释放接口	输出内存申请/释放接口
Atlas A2训练系列产品/ Atlas 800I A2推理产品 Atlas 200I/500 A2推理 产品	JPEGE图片编码功能时，支持以下两种内存申请接口： - 调用acl.rt.malloc \ acl.rt.free接口申请\释放Device上的输入内存，这部分内存的生命周期由用户自行管理。 - 调用 acl.himpi.dvpp_malloc \ acl.himpi.dvpp_free接口申请\释放Device上的输入内存，这部分内存的生命周期由用户自行管理。 注意：调用 acl.himpi.dvpp_malloc接口申请的内存为媒体数据处理的专用内存，但专用内存的地址空间有限，若关注内存规划或内存资源有限时，建议调用acl.rt.malloc接口申请内存。	支持以下两种方式： - 不需要用户管理、由DVPP内部管理输出内存时，调用 acl.himpi.venc_send_frame接口发送原始图像进行图像编码。 在调用 acl.himpi.venc_create_chn接口创建通道时，必须正确设置 <code>hi_venc_chn_attr["venc_attr"]["buf_size"]</code>属性值（参数描述请参见hi_venc_attr）。 - 该方式下，相比由用户管理内存，输出结果数据的JPEG头中不存在COM注释字段，数据长度会短一点，但需要用户从DVPP返回的内存中拷贝输出结果数据到指定内存。 - 由用户自行管理输出内存、管理内存的生命周期，调用 acl.himpi.venc_send_jpege_frame接口发送原始图像进行图像编码。 在调用 acl.himpi.venc_create_chn接口创建通道时，需将 <code>hi_venc_chn_attr["venc_attr"]["buf_size"]</code>属性值设置为0（参数描述请参见hi_venc_attr），然后调用 acl.himpi.venc_get_jpege_predicted_size接口预估输出内存大小，调用 acl.himpi.dvpp_malloc \ acl.himpi.dvpp_free接口申请\释放输出内存（也支持调用 acl.rt.malloc \ acl.rt.free接口申请\释

版本	输入内存申请/释放接口	输出内存申请/释放接口
		放Device上的输出内存)。 该方式下，直接在调用 acl.himpi.venc_send_jpege_frame 接口时，设置输出内存地址，输出结果数据直接存放到用户设置的内存中，相比由系统管理内存的方式，用户可减少一次“从DVPP返回的内存中拷贝输出结果数据到指定内存”的操作，但输出结果数据的JPEG头中可能会存在COM注释字段（字段长度范围4~19Byte），数据长度会长一点。

图片格式、宽高对齐、内存约束

说明

输入图片格式的定义请参见[hi_pixel_format](#)，宽Stride、高Stride等概念请参见[2.10.2 基本概念](#)。

表 2-56 图片格式、宽高对齐、内存大小约束

输入图片格式	输入图片宽、高对齐要求	输入图片宽Stride、高Stride、内存大小要求	输出图片格式
YUV420SP NV12 8bit	宽2对齐 高2对齐	宽Stride为宽16对齐后的值，兼容对齐到16的倍数如128（按128对齐时，性能更优）。 高Stride无需设置。 内存大小（单位Byte）= 宽Stride * 高 * 3/2	JPEG压缩格式的图片文件，例如*.jpg。 只支持huffman编码，不支持算术编码，不支持渐进编码。
YUV420SP NV21 8bit			
YUV422Packed YUYV 8bit	宽2对齐 高无对齐要求		
YUV422Packed UYVY 8bit			
YUV422Packed YVYU 8bit			

输入图片格式	输入图片宽、高对齐要求	输入图片宽Stride、高Stride、内存大小要求	输出图片格式
YUV422Packed VYUY 8bit			

2.10.6.2 JPEG 性能指标说明

Atlas 200/300/500 推理产品上，该功能为预留功能，暂不支持。

Atlas 训练系列产品上，该功能为预留功能，暂不支持。

性能指标说明

本节仅描述各产品对应的昇腾AI处理器的性能指标参考数据，各产品由于其内存、CPU等配置不同，在性能规格上会存在差异，详细产品的性能规格请查阅[昇腾社区](#)的官方说明。

本节以单个昇腾AI处理器为例，描述基本场景性能指标参考，其它分辨率可以等量估算。

下表中的1路对应一个通道，一个通道对应一个线程；各分辨率的含义为：1080p指分辨率为1920 * 1080的图片，4K指分辨率为3840 * 2160的图片。

表 2-57 Atlas 推理系列产品性能规格数据

场景举例（输入图片格式：YUV420 8bit）	总帧率
1080p * n路（1 ≤ n ≤ 8）	n * 128fps
1080p * n路（n > 8）	1024fps
4k * n路（1 ≤ n ≤ 8）	n * 32fps
4k * n路（n > 8）	256fps

表 2-58 Atlas 200I/500 A2 推理产品性能规格数据

场景举例（输入图片格式：YUV420 8bit）	总帧率
1080p * n路	256fps
4k * n路	64fps

表 2-59 Atlas A2 训练系列产品/Atlas 800I A2 推理产品性能规格数据

场景举例（输入图片格式：YUV420 8bit）	总帧率
1080p * n路（ $1 \leq n \leq 4$ ）	$n * 256\text{fps}$
1080p * n路（ $n > 4$ ）	1024fps
4k * n路（ $1 \leq n \leq 4$ ）	$n * 64\text{fps}$
4k * n路（ $n > 4$ ）	256fps

2.10.6.3 VENC 功能及约束说明

Atlas 200/300/500 推理产品上，不支持本章中的VENC功能。

Atlas 训练系列产品上，不支持本章中的VENC功能。

Atlas A2训练系列产品/Atlas 800I A2推理产品上，不支持该功能。

功能说明

VENC（Video Encoder）将YUV420SP NV12/NV21-8bit图片数据编码成H.264/H.265格式的视频码流。

分辨率约束

- 输入图片分辨率

版本	分辨率范围
Atlas 推理系列产品	宽和高的范围都是128 ~ 4096，对于H.264格式的码流，最大分辨率的乘积不超过4096 * 2304。
Atlas 200I/500 A2推理产品	最大分辨率8192 * 8192，最小分辨率114 * 114

- 输出码流分辨率

VENC只对图片编码，不会改变图片分辨率，因此输出与输入的图片分辨率保持一致。

内存申请/释放接口说明

输入内存的大小参见表2-60中的计算公式，输出内存无需用户管理、由系统管理。

版本	内存申请/释放接口
Atlas 推理系列产品	VENC视频编码时，调用 acl.himpi.dvpp_malloc \ acl.himpi.dvpp_free 接口申请/释放Device上的输入内存，这部分内存的生命周期由用户自行管理。

版本	内存申请/释放接口
Atlas 200I/500 A2推理产品	VENC视频编码时，支持以下两种内存申请接口： - 调用acl.rt.malloc\acl.rt.free接口申请\释放Device上的输入内存，这部分内存的生命周期由用户自行管理。 - 调用acl.himpi.dvpp_malloc\acl.himpi.dvpp_free接口申请\释放Device上的输入内存，这部分内存的生命周期由用户自行管理。 调用acl.himpi.dvpp_malloc接口申请的内存为媒体数据处理的专用内存，但专用内存的地址空间有限，若关注内存规划或内存资源有限时，建议调用acl.rt.malloc接口申请内存。

码流/图片格式、宽高对齐、内存约束

📖 说明

输入图片格式的定义请参见[hi_pixel_format](#)，宽Stride、高Stride等概念请参见[2.10.2 基本概念](#)。

表 2-60 图片格式、宽高对齐、内存大小约束

输入图片格式	输入图片宽、高对齐要求	输入图片宽Stride、高Stride、内存大小要求	输出码流格式
YUV420SP NV12 8bit	宽2对齐 高2对齐	宽Stride为宽16对齐后的值。 高Stride无需设置。 内存大小（单位Byte） = 宽Stride * 高Stride * 3/2	- H.264 BP/MP/HP - H.265 MP（仅支持Slice码流）
YUV420SP NV21 8bit			

2.10.6.4 VENC 性能指标说明

Atlas 200/300/500 推理产品上，该功能为预留功能，暂不支持。

Atlas 训练系列产品上，不支持该功能。

Atlas A2训练系列产品/Atlas 800I A2推理产品上，不支持该功能。

性能指标说明

本节仅描述各产品对应的昇腾AI处理器的性能指标参考数据，各产品由于其内存、CPU等配置不同，在性能规格上会存在差异，详细产品的性能规格请查阅[昇腾社区](#)的官方说明。

本节以单个昇腾AI处理器为例，描述基本场景性能指标参考，其它分辨率可以等量估算。

下表中的1路对应一个通道，一个通道对应一个线程；各分辨率的含义为：720p指分辨率为1280 * 720的图片，1080p指分辨率为1920 * 1080的图片，4K指分辨率为3840 * 2160的图片。

表 2-61 Atlas 推理系列产品参考性能指标

场景举例	总帧率
720p * n路($1 \leq n \leq 3$)	$n * 480\text{fps}$
720p * n路($n > 3$)	1440fps
1080p * n路($1 \leq n \leq 3$)	$n * 240\text{fps}$
1080p * n路($n > 3$)	720fps
4k * n路($1 \leq n \leq 3$)	$n * 60\text{fps}$
4k * n路($n > 3$)	180fps

表 2-62 Atlas 200I/500 A2 推理产品参考性能指标

场景举例	总帧率
720p * n路($1 \leq n \leq 3$)	$n * 400\text{fps}$
720p * n路($n > 3$)	1200fps
1080p * n路($1 \leq n \leq 3$)	$n * 200\text{fps}$
1080p * n路($n > 3$)	600fps
4k * n路($1 \leq n \leq 3$)	$n * 50\text{fps}$
4k * n路($n > 3$)	150fps

2.10.6.5 acl.himpi.venc_create_chn

c函数原型	<code>hi_s32 hi_mpi_venc_create_chn(hi_venc_chn chn, const hi_venc_chn_attr *attr);</code>
Python函数	<code>ret = acl.himpi.venc_create_chn(chn, attr)</code>
函数功能	创建编码通道。 Atlas 200/300/500 推理产品 ，当前版本不支持该接口。 Atlas 训练系列产品 ，当前版本不支持该接口。

输入说明	chn: int, 编码通道号。 • Atlas 推理系列产品, 编码通道号的取值范围: [0, 256), JPEG功能 and VENC功能共用通道, 且通道总数最多256。 • Atlas A2训练系列产品/Atlas 800I A2推理产品, 编码通道号的取值范围: [0, 256), JPEG功能通道总数最多256。 • Atlas 200I/500 A2推理产品, 编码通道号的取值范围: [0, 128), JPEG功能和VENC功能共用通道, 且通道总数最多128。 说明 对于Atlas 推理系列产品, 昇腾虚拟化实例 (Ascend Virtual Instance) 场景下, VENC通道和JPEG通道总数 = (被分配的VENC硬件单元+JPEG硬件单元数量之和 / VENC硬件单元+JPEG硬件单元总数之和) * 256, 如果通道总数不为整数, 则向下取整。 对于Atlas A2训练系列产品/Atlas 800I A2推理产品, 昇腾虚拟化实例 (Ascend Virtual Instance) 场景下, JPEG通道总数最多256。 对于Atlas 200I/500 A2推理产品, 昇腾虚拟化实例 (Ascend Virtual Instance) 场景下, VENC通道 = (被分配的VENC硬件单元 / VENC硬件单元) * 128, 如果通道总数不为整数, 则向下取整。JPEG的通道数不随算力影响, 但JPEG+VENC的总通道数最大128。 昇腾虚拟化实例 (Ascend Virtual Instance) 的详细描述, 以及各场景下的算力资源规格请参见《昇腾虚拟化实例 (AVI) 用户指南》。 attr: dict, 编码通道属性字典, 具体请参见2.10.8.4.3 hi_venc_chn_attr。
返回值说明	ret: int, 错误码。 • 返回0表示成功。 • 返回非0表示失败, 参见2.10.9.4 VENC视频/JPEG图像编码返回码。

约束说明	• 单个Device上的通道号不能超出最大的通道号范围。 • 编码通道属性由三部分组成，编码器属性、码率控制器属性和帧结构类型属性，帧结构类型属性简称GOP（ Group of Pictures ）类型属性。 • 编码器属性必须设置编码码流Buffer深度、获取码流方式、编码profile等，表2-63详细描述了各种协议的各项属性的特性。 • 推荐的编码宽高为：3840 * 2160(4k)、1920 * 1080(1080P)、1280 * 720(720P)。 • H.264/H.265编码帧存由YHeaderSize、CHeaderSize、YSize、CSize、PmeSize、PmeInfoSize和TmvSize共同构成，编码器默认根据最大宽高计算帧存进行内存分配，在设置通道宽高时需保证根据通道宽高计算的每一部分帧存大小都不能大于最大宽高计算的帧存大小。 • 码率控制器属性中的输入/输出帧率（src_frame_rate/dst_frame_rate）是用于配合编码过程中的码率（bit_rate）分配，不是最终的播放器显示帧率，若想要设置显示帧率请参考acl.himpi.venc_set_h264_vui或acl.himpi.venc_set_h265_vui接口。 建议用户将码控帧率与显示帧率设置一致，否则播放时的码率可能与用户设置的编码码率不同，播放时的码率会随着所设置的显示帧率而变化。
参考资源	接口调用流程，参见JPEGE图片编码、VENC视频编码。

参考信息

表 2-63 编码器属性的约束

编码协议	编码方式	码流 Buffer 深度	获取码流模式	编码 profile
H.264	Frame	• mini_buf_mode = 0时，Buffer ≥ max_pic_width * max_pic_height * 3/4。 • mini_buf_mode = 1时，Buffer ≥ 32 * 1024 Byte。	Frame/ Slice	Baseline Mainprofile Highprofile
JPEG	Frame	• mini_buf_mode = 0时，如果原图格式为YUV420，Buffer最小值为原图宽16对齐 * 原图高16对齐 * 3/2。如果原图格式为YUV422 Packed，Buffer最小值为原图宽16对齐 * 原图高16对齐 * 2 • mini_buf_mode = 1时，Buffer ≥ 32 * 1024 Byte。	Frame/Ecs	Baseline

编码协议	编码方式	码流 Buffer 深度	获取码流模式	编码 profile
H.265	Frame	- mini_buf_mode = 0时, Buffer $\geq \text{max\_pic\_width} * \text{max\_pic\_height} * 3/4$。 - mini_buf_mode = 1时, Buffer $\geq 32 * 1024$ Byte。	Frame/ Slice	Main profile Main 10 profile (暂不支持)
		- mini_buf_mode = 0时, Buffer $\geq \text{max\_pic\_width} * \text{max\_pic\_height} * (3/4) * (5/4)$ 。 - mini_buf_mode = 1时, Buffer $\geq 32 * 1024$ Byte	Frame/ Slice	Main 10 profile (暂不支持)

2.10.6.6 acl.himpi.venc_destroy_chn

c函数原型	hi_s32 hi_mpi_venc_destroy_chn(hi_venc_chn chn);
Python函数	ret = acl.himpi.venc_destroy_chn(chn)
函数功能	用户退出编码流程时，需要关闭之前打开的编码通道，以释放相关资源。 Atlas 200/300/500 推理产品 ，当前版本不支持该接口。 Atlas 训练系列产品 ，当前版本不支持该接口。
输入说明	chn : int，编码通道号。 Atlas 推理系列产品，编码通道号的取值范围：[0, 256)，JPEGE功能和VENC功能共用通道，且通道总数最多256。 Atlas A2训练系列产品/Atlas 800I A2推理产品，编码通道号的取值范围：[0, 256)，JPEGE功能通道总数最多256。 Atlas 200I/500 A2推理产品，编码通道号的取值范围：[0, 128)，JPEGE功能和VENC功能共用通道，且通道总数最多128。
返回值说明	ret : int，错误码。 - 返回0表示成功。 - 返回非0表示失败，参见2.10.9.4 VENC视频/JPEGE图像编码返回码。
约束说明	- 销毁并不存在的通道，返回失败。 - 销毁前必须停止接收图像，否则返回失败。
参考资源	接口调用流程，参见JPEGE图片编码、VENC视频编码。

2.10.6.7 acl.himpi.venc_start_chn

c函数原型	hi_s32 hi_mpi_venc_start_chn (hi_venc_chn chn, const hi_venc_start_param *recv_param);
Python函数	ret = acl.himpi.venc_start_chn(chn, recv_param)
函数功能	开启编码通道接收输入图片，允许指定接收帧数，超出指定的帧数后自动停止接收图像。 Atlas 200/300/500 推理产品 ，当前版本不支持该接口。 Atlas 训练系列产品 ，当前版本不支持该接口。
输入说明	chn : int，编码通道号。 • Atlas 推理系列产品，编码通道号的取值范围：[0, 256)，JPEGE功能和VENC功能共用通道，且通道总数最多256。 • Atlas A2训练系列产品/Atlas 800I A2推理产品，编码通道号的取值范围：[0, 256)，JPEGE功能通道总数最多256。 • Atlas 200I/500 A2推理产品，编码通道号的取值范围：[0, 128)，JPEGE功能和VENC功能共用通道，且通道总数最多128。 recv_param : dict，接收图像参数字典，用于指定需要接收的图像帧数，具体请参见 2.10.8.4.31 hi_venc_start_param 。
返回值说明	ret : int，错误码。 • 返回0表示成功。 • 返回非0表示失败，参见2.10.9.4 VENC视频/JPEGE图像编码返回码。
约束说明	• 如果通道未创建，则返回错误码“HI_ERR_VENC_UNEXIST”。 • 需要接收的图像帧数设置为-1时表示不指定帧数。 • 如果通道已经开始接收图像，没有停止接收图像前再一次调用此接口指定接收帧数，返回操作不允许。 • 如果通道已经开始接收图像，没有停止接收图像前允许再一次调用此接口不指定接收帧数。 • 只有开启接收之后编码器才开始接收图像编码。
参考资源	接口调用流程，参见JPEGE图片编码、VENC视频编码。

2.10.6.8 acl.himpi.venc_stop_chn

c函数原型	hi_s32 hi_mpi_venc_stop_chn(hi_venc_chn chn);
Python函数	ret = acl.himpi.venc_stop_chn(chn)
函数功能	停止编码通道接收输入图片。 Atlas 200/300/500 推理产品 ，当前版本不支持该接口。 Atlas 训练系列产品 ，当前版本不支持该接口。

输入说明	chn: int, 编码通道号。 • Atlas 推理系列产品, 编码通道号的取值范围: [0, 256), JPEG功能 and VENC功能共用通道, 且通道总数最多256。 • Atlas A2训练系列产品/Atlas 800I A2推理产品, 编码通道号的取值范围: [0, 256), JPEG功能通道总数最多256。 • Atlas 200I/500 A2推理产品, 编码通道号的取值范围: [0, 128), JPEG功能和VENC功能共用通道, 且通道总数最多128。
返回值说明	ret: int, 错误码。 • 返回0表示成功。 • 返回非0表示失败, 参见2.10.9.4 VENC视频/JPEG图像编码返回码。
约束说明	• 如果通道未创建, 则返回失败。 • 此接口并不判断当前是否停止接收, 即允许重复停止接收不返回错误。 • 此接口用于编码通道停止接收图像来编码, 在编码通道销毁或复位前必须停止接收图像。 • 调用此接口仅停止接收原始数据编码, 码流Buffer并不会被清除。
参考资源	接口调用流程, 参见JPEG图片编码、VENC视频编码。

2.10.6.9 acl.himpi.venc_query_status

c函数原型	hi_s32 hi_mpi_venc_query_status(hi_venc_chn chn, hi_venc_chn_status *status);
Python函数	status, ret = acl.himpi.venc_query_status(chn)
函数功能	查询编码通道状态。 Atlas 200/300/500 推理产品, 当前版本不支持该接口。 Atlas 训练系列产品, 当前版本不支持该接口。
输入说明	chn: int, 编码通道号。 • Atlas 推理系列产品, 编码通道号的取值范围: [0, 256), JPEG功能和VENC功能共用通道, 且通道总数最多256。 • Atlas A2训练系列产品/Atlas 800I A2推理产品, 编码通道号的取值范围: [0, 256), JPEG功能通道总数最多256。 • Atlas 200I/500 A2推理产品, 编码通道号的取值范围: [0, 128), JPEG功能和VENC功能共用通道, 且通道总数最多128。

返回值说明	ret: int, 错误码。 • 返回0表示成功。 • 返回非0表示失败, 参见2.10.9.4 VENC视频/JPEGE图像编码返回码。 status: dict, 编码通道的状态2.10.8.4.32 hi_venc_chn_status。
约束说明	• 如果通道未创建, 则返回失败。 • 此接口用于查询此函数调用时刻的编码器状态, “status” 包含以下信息: - 在编码通道状态字典中, “left_pics” 表示待编码的帧个数。在复位通道前, 可以通过查询是否还有图像待编码来决定复位时机, 防止复位时将可能需要编码的帧清理出去。 - 在编码通道状态字典中, “left_stream_bytes” 表示码流Buffer中剩余的Byte数目。 在复位通道前, 可以通过查询是否还有码流没有被处理来决定复位时机, 防止复位时将可能需要的码流清理出去。 - 在编码通道状态字典中, “left_stream_frames” 表示码流Buffer中剩余的帧数目。 在复位通道前, 可以通过查询是否还有图像的码流没有被取走来决定复位时机, 防止复位时将可能需要的码流清理出去。 - 在编码通道状态字典中, “cur_packs” 表示当前帧的码流包个数。在调用acl.himpi.venc_get_stream之前应确保“cur_packs” 大于 0。 在按包获取时当前帧可能不是一个完整帧 (被取走一部分), 按帧获取时表示当前一个完整帧的包个数 (如果没有一帧数据则为 0)。用户在需要按帧获取码流时, 需要查询一个完整帧的包个数, 在这种情况下, 通常可以在select成功后执行query操作, 此时 “cur_packs” 是当前完整帧中包的个数。 - 在编码通道状态字典中, “left_recv_pics” 表示调用acl.himpi.venc_start_chn接口后剩余等待接收的帧数目。 - 在编码通道状态字典中, “left_enc_pics” 表示调用acl.himpi.venc_start_chn接口后剩余等待编码的帧数目。 - 如果调用acl.himpi.venc_start_chn但没有指定接收帧数, “left_recv_pics” 和 “left_enc_pics” 数目始终为0。
参考资源	接口调用流程, 参见JPEGE图片编码、VENC视频编码。

2.10.6.10 acl.himpi.venc_get_stream

c函数原型	<code>hi_s32 hi_mpi_venc_get_stream(hi_venc_chn chn, hi_venc_stream *stream, hi_s32 milli_sec);</code>
Python函数	<code>stream, ret = acl.himpi.venc_get_stream(chn, stream, milli_sec)</code>

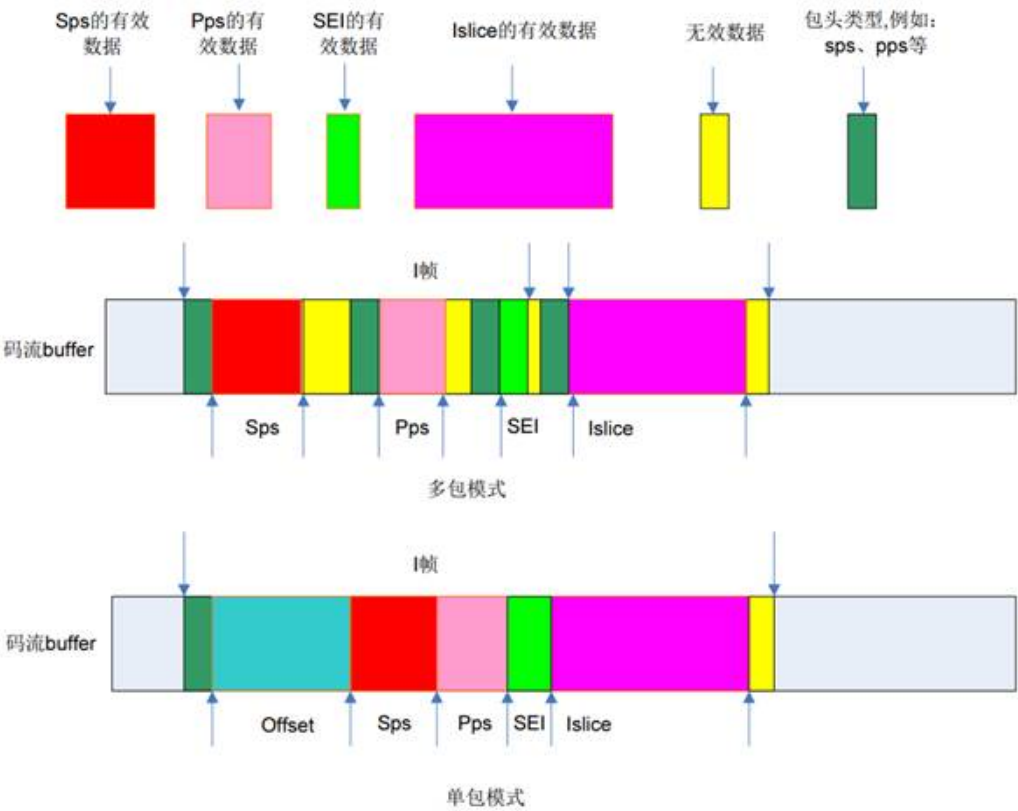
函数功能	获取编码的码流。 Atlas 200/300/500 推理产品，当前版本不支持该接口。 Atlas 训练系列产品，当前版本不支持该接口。
输入说明	chn: int, 编码通道号。 • Atlas 推理系列产品，编码通道号的取值范围：[0, 256)，JPEG功能 and VENC功能共用通道，且通道总数最多256。 • Atlas A2训练系列产品/Atlas 800I A2推理产品，编码通道号的取值范围：[0, 256)，JPEG功能通道总数最多256。 • Atlas 200I/500 A2推理产品，编码通道号的取值范围：[0, 128)，JPEG功能和VENC功能共用通道，且通道总数最多128。 stream: dict, “pack_cnt” 传入 “cur_packs” 个包信息字典，自动分配内存，具体请参见2.10.8.4.1 hi_venc_stream。 milli_sec: int, 超时时间，单位是毫秒。 • “-1”：阻塞方式。 • “0”：非阻塞方式。 • >0: 超时方式，配置具体的超时时间。超时时间受操作系统影响，一般偏差在操作系统的一个时间片内，例如，操作系统的 一个时间片为4ms，用户设置的“milli_sec”参数值为“1”，则实际的超时时间在1ms到5ms范围内。在CPU负载高场景下，超时时间仍可能存在波动。
返回值说明	ret: int, 错误码。 • 返回0表示成功。 • 返回非0表示失败，参见2.10.9.4 VENC视频/JPEG图像编码返回码。 stream: dict, 码流字典。该字典内的“addr”属性配置的地址为Device上的内存地址，表示输出的码流存放在Device的内存中。 • Atlas 推理系列产品，标准形态下，用户在Host上调用该接口后，需要访问输出码流数据，则需要将Device中的数据传输回Host。 • Atlas A2训练系列产品/Atlas 800I A2推理产品，用户在Host上调用该接口后，需要访问输出码流数据，则需要将Device中的数据传输回Host。 • Atlas 200I/500 A2推理产品，EP模式下，用户在Host上调用该接口后，需要访问输出码流数据，则需要将Device中的数据传回Host。

约束说明	• 如果通道未创建，返回失败。 • 如果“stream”为空，返回错误码“HI_ERR_VENC_NULL_PTR”。 • 如果“milli_sec”小于-1，返回错误码“HI_ERR_VENC_ILLEGAL_PARAM”。 • 支持超时方式获取。支持select/poll系统调用。 - milli_sec = 0 时，则为非阻塞获取，即如果缓冲无数据，则返回错误码“HI_ERR_VENC_BUF_EMPTY”。 - milli_sec = -1 时，则为阻塞，即如果缓冲无数据，则会等待有数据时才返回获取成功。 - milli_sec > 0 时，则为超时，即如果缓冲无数据，则会等待用户设定的超时时间，若在设定的时间内有数据则返回获取成功，否则返回超时失败。 • 支持按包或按帧方式获取码流。如果按包获取，则： - 对于JPEG编码协议，每次获取的是一个参数包或数据包。 - 对于H.265编码协议，每次获取的是一个NAL单元。 • 码流字典“hi_venc_stream”包含以下部分： - 码流包信息列表“pack” 在函数调用成功后，将在返回值中“stream”字典填充一组hi_venc_pack字典的列表。 - 码流包个数“pack_cnt” 在输入时，此值指定pack中要获取的“hi_venc_pack”字典的最大数量。按包获取时，“pack_cnt”必须不小于1；按帧获取时，“pack_cnt”必须不小于当前帧的包个数（可通过acl.himpi.venc_query_status接口获得）。 在函数调用成功后，返回值中“stream”字典的“pack_cnt”为实际填充pack的包个数。 - 序列号seq按帧获取时是帧序列号；按包获取时为包序列号。 • 如果用户长时间不获取码流，码流缓冲区就会满。一个编码通道如果发生码流缓冲区满，就会不再启动编码，直到用户获取码流并释放码流，从而有足够的码流缓冲可以用于编码时，才开始继续编码。 • 建议用户获取码流接口调用与释放码流的接口调用成对出现，且尽快释放码流，防止出现由于用户获取码流，释放不及时而导致的码流Buffer满，停止编码。 • 建议用户使用select方式获取码流，且按照如下的流程： 1. 调用acl.himpi.venc_query_status函数查询编码通道状态。 2. 确保“cur_packs”和“left_stream_frames”同时大于0。 3. 构造一个“stream”字典。<pre>stream["pack_cnt"] = cur_packs</pre> 4. 调用acl.himpi.venc_get_stream传入“stream”字典，获取编码码流。<pre>stream, ret = acl.himpi.venc_get_stream(chn, stream)</pre>

	5. 调用acl.himpi.venc_release_stream传入上一步返回的“stream”字典，释放码流缓存。 - 当编码模块参数中的“one_stream_buf”为1时，用户获取一帧码流时只会得到一个码流包（不包含用户数据）的地址，即“pack_cnt”为1，地址为pack[0] ["addr"], 在“hi_venc_pack”字典中增加offset，用来指出一帧码流中有效数据的地址和pack[0] ["addr"]的偏移。如图2-20所示（以H.264单包为例），pack[0] ["data_num"] = 3，即3种NAL包，分别为Sps、Pps、SEI。 pack[0] ["pack_info"] [0] ["pack_type"] ["h264_type"] = HI_VENC_H264_NALU_SPS pack[0] ["pack_info"] [1] ["pack_type"] ["h264_type"] = HI_VENC_H264_NALU_PPS pack[0] ["pack_info"] [2] ["pack_type"] ["h264_type"] = HI_VENC_H264_NALU_SEI 其他协议以此类推，“data_num”和“pack_info”只在单包模式才有意义。 - 两种模式可通过调用接口acl.himpi.venc_set_mod_param设置h.265e/h264e/jpeg模块参数“one_stream_buf”来选择。 - one_stream_buf = 1 表示单包模式（JPEG目前只支持单帧单包模式）。 - one_stream_buf = 0 表示多包模式，系统默认one_stream_buf = 0。
参考资源	接口调用流程，参见JPEG图片编码、VENC视频编码。

其他参考信息

图 2-20 码流包结构图



2.10.6.11 acl.himpi.venc_release_stream

c函数原型	hi_s32 hi_mpi_venc_release_stream(hi_venc_chn chn, hi_venc_stream *stream);
Python函数	ret = acl.himpi.venc_release_stream(chn, stream)
函数功能	释放码流缓存。 Atlas 200/300/500 推理产品，不支持该接口。 Atlas 训练系列产品，不支持该接口。
输入说明	chn: int，编码通道号。 - Atlas 推理系列产品，编码通道号的取值范围：[0, 256)，JPEG功能 and VENC功能共用通道，且通道总数最多256。 - Atlas A2训练系列产品/Atlas 800I A2推理产品，编码通道号的取值范围：[0, 256)，JPEG功能通道总数最多256。 - Atlas 200I/500 A2推理产品，编码通道号的取值范围：[0, 128)，JPEG功能和VENC功能共用通道，且通道总数最多128。 stream: dict，码流字典。如果“stream”为空，则返回错误码“HI_ERR_VENC_NULL_PTR”。

返回值说明	ret : int, 错误码。 • 返回0表示成功。 • 返回非0表示失败, 参见2.10.9.4 VENC视频/JPEGE图像编码返回码。
约束说明	• 如果通道未创建, 则返回错误码“HI_ERR_VENC_UNEXIST”。 • 此接口应当和acl.himpi.venc_get_stream配对起来使用, 用户获取码流后必须及时释放已经获取的码流缓存, 否则可能会导致码流 Buffer 满, 影响编码器编码, 并且用户必须按先获取先释放的顺序释放已经获取的码流缓存。 • 在编码通道复位以后, 所有未释放的码流包均无效, 不能再使用或者释放这部分无效的码流缓存。 • 释放无效的码流会返回失败“HI_ERR_VENC_ILLEGAL_PARAM”。
参考资源	接口调用流程, 参见JPEGE图片编码、VENC视频编码。

2.10.6.12 acl.himpi.venc_send_frame

c函数原型	hi_s32 hi_mpi_venc_send_frame(hi_venc_chn chn, const hi_video_frame_info *frame, hi_s32 milli_sec)
Python函数	ret = acl.himpi.venc_send_frame(chn, frame, milli_sec)
函数功能	发送原始图像进行视频或图像编码。 Atlas 200/300/500 推理产品 , 不支持该接口。 Atlas 训练系列产品 , 不支持该接口。

输入说明	chn: int, 编码通道号。 • Atlas 推理系列产品, 编码通道号的取值范围: [0, 256), JPEG功能 and VENC功能共用通道, 且通道总数最多256。 • Atlas A2训练系列产品/Atlas 800I A2推理产品, 编码通道号的取值范围: [0, 256), JPEG功能通道总数最多256。 • Atlas 200I/500 A2推理产品, 编码通道号的取值范围: [0, 128), JPEG功能和VENC功能共用通道, 且通道总数最多128。 frame: dict, 原始图像信息字典, 具体请参见2.10.8.2.9 hi_video_frame_info。 milli_sec: int, 超时时间, 单位是毫秒。 • “-1”: 阻塞方式。 • “0”: 非阻塞方式。 • >0: 超时方式, 配置具体的超时时间。超时时间受操作系统影响, 一般偏差在操作系统的 一个时间片内, 例如, 操作系统的 一个时间片为4ms, 用户设置的 “milli_sec” 参数值为 “1”, 则实际的超时时间在1ms到5ms范围内。在CPU负载高场景下, 超时时间仍可能存在波动。
返回值说明	ret: int, 错误码。 • 返回0表示成功。 • 返回非0表示失败, 参见2.10.9.4 VENC视频/JPEG图像编码返回码。
约束说明	• 视频输入的原始图像大小必须与编码通道的大小保持一致, JPEG输入的原始图像大小小于或等于编码通道的大小。 • 调用该接口发送图像, 用户需要保证编码通道已创建且开启接收输入图片。 • 编码时, 输入内存需要在调用acl.himpi.venc_get_stream接口获取结果之后才能进行释放。 • 视频编码, 对输入、输出的约束请参见2.10.6.3 VENC功能及约束说明。 • 图像编码, 对输入、输出的约束请参见2.10.6.1 JPEG功能及约束说明。
参考资源	接口调用流程, 参见JPEG图片编码、VENC视频编码。

2.10.6.13 acl.himpi.venc_set_mod_param

c函数原型	hi_s32 hi_mpi_venc_set_mod_param(const hi_venc_mod_param *mod_param)
Python函数	ret = acl.himpi.venc_set_mod_param(mod_param)

函数功能	设置编码相关的模块参数。 Atlas 200/300/500 推理产品 ，当前版本不支持该接口。 Atlas 训练系列产品 ，当前版本不支持该接口。 Atlas A2训练系列产品/Atlas 800I A2推理产品 ，当前版本不支持该接口。
输入说明	mod_param : dict, 编码模块参数字典，具体请参见 2.10.8.4.38 hi_venc_mod_param 。
返回值说明	ret : int, 错误码。 - 返回0表示成功。 - 返回非0表示失败，参见2.10.9.4 VENC视频/JPEGE图像编码返回码。
约束说明	此接口在通道创建前调用，如果通道已经创建，则返回错误码“HI_ERR_VENC_NOT_PERM”。 建议用户在调用此接口之前先调用 acl.himpi.venc_get_mod_param 接口，获取当前通道的编码模块参数配置，然后再根据需求设置相应参数。
参考资源	接口调用流程，参见JPEGE图片编码、VENC视频编码。

2.10.6.14 acl.himpi.venc_get_mod_param

c函数原型	hi_s32 hi_mpi_venc_get_mod_param(hi_venc_mod_param *mod_param)
Python函数	mod_param, ret = acl.himpi.venc_get_mod_param(mod_param)
函数功能	获取编码相关的模块参数。 Atlas 200/300/500 推理产品 ，当前版本不支持该接口。 Atlas 训练系列产品 ，当前版本不支持该接口。 Atlas A2训练系列产品/Atlas 800I A2推理产品 ，当前版本不支持该接口。
输入说明	mod_param : dict, 编码模块类型，具体请参见 2.10.8.4.38 hi_venc_mod_param 。
返回值说明	ret : int, 错误码。 - 返回0表示成功。 - 返回非0表示失败，参见2.10.9.4 VENC视频/JPEGE图像编码返回码。 mod_param : dict, 编码模块参数。
约束说明	建议用户在调用此接口之前先设置 hi_venc_mod_param 字典中的“mod_type”参数，接口会根据不同的模块类型去获取相应模块参数。

参考资源	接口调用流程，参见JPEG图片编码、VENC视频编码。

2.10.6.15 acl.himpi.venc_request_idr

c函数原型	hi_s32 hi_mpi_venc_request_idr(hi_venc_chn chn, hi_bool instant);
Python函数	ret = acl.himpi.venc_request_idr(chn, instant)
函数功能	请求IDR帧。 Atlas 200/300/500 推理产品 ，当前版本不支持该接口。 Atlas 训练系列产品 ，当前版本不支持该接口。 Atlas A2训练系列产品/Atlas 800I A2推理产品 ，当前版本不支持该接口。
输入说明	chn : 编码通道号。 Atlas 推理系列产品，编码通道号的取值范围: [0, 256)，JPEG功能和VENC功能共用通道，且通道总数最多256。 Atlas 200I/500 A2推理产品，编码通道号的取值范围: [0, 128)，JPEG功能和VENC功能共用通道，且通道总数最多128。 instant : 是否使能立即编码IDR帧。
返回值说明	ret : int, 错误码。 - 返回0表示成功。 - 返回非0表示失败，参见2.10.9.4 VENC视频/JPEG图像编码返回码。
约束说明	- 如果通道未创建，则返回失败。 - 接受IDR (Instantaneous Decoder Refresh) 帧请求后，当 instant = 0时，则在帧率控制的下一帧编出IDR帧，当 instant = 1 时，则立即编出IDR帧，不受帧率控制约束。 - IDR帧请求，只支持H.264/H.265编码协议。 - 由于目前的使用场景目标帧率与源帧率的值一致，所以此接口不受帧率控制影响当“instant”设置为0或1时，都是每调用一次接口即编出一个IDR，调用频繁会影响码流帧率和码率的稳定，使用时需要注意。 - 当GOP模式为Smartp或B帧模式（暂不支持这两种模式暂不支持B帧模式）下，请求IDR帧会延时生效。

2.10.6.16 acl.himpi.venc_get_fd

c函数原型	hi_s32 hi_mpi_venc_get_fd(hi_venc_chn chn);
Python函数	ret = acl.himpi.venc_get_fd(chn)

函数功能	获取编码通道对应的文件句柄。 Atlas 200/300/500 推理产品 ，当前版本不支持该接口。 Atlas 训练系列产品 ，当前版本不支持该接口。
输入说明	chn : int，编码通道号。 • Atlas 推理系列产品，编码通道号的取值范围：[0, 256)，JPEGE功能和VENC功能共用通道，且通道总数最多256。 • Atlas A2训练系列产品/Atlas 800I A2推理产品，编码通道号的取值范围：[0, 256)，JPEGE功能通道总数最多256。 • Atlas 200I/500 A2推理产品，编码通道号的取值范围：[0, 128)，JPEGE功能和VENC功能共用通道，且通道总数最多128。
返回值说明	ret : int，编码通道对应的文件句柄。 • 0或正数值：有效返回值。 • 非正数值：无效返回值。
约束说明	在进程退出时会自动关闭文件句柄，回收资源。
参考资源	接口调用流程，参见JPEGE图片编码、VENC视频编码。

2.10.6.17 acl.himpi.venc_close_fd

c函数原型	hi_s32 hi_mpi_venc_close_fd(hi_venc_chn chn);
Python函数	ret = acl.himpi.venc_close_fd(chn)
函数功能	关闭编码通道对应的设备文件句柄。 当前版本不支持该接口。
输入说明	chn : int，编码通道号。
返回值说明	ret : int，错误码。 • 0：关闭成功。 • -1：关闭失败。

2.10.6.18 acl.himpi.venc_set_jpeg_param

c函数原型	hi_s32 hi_mpi_venc_set_jpeg_param(hi_venc_chn chn, const hi_venc_jpeg_param *jpeg_param)
Python函数	ret = acl.himpi.venc_set_jpeg_param(chn, jpeg_param)
函数功能	设置 JPEG 协议编码通道的量化表高级配置参数，用户需要调整 JPEGE量化参数表时才需要使用该接口。 Atlas 200/300/500 推理产品 ，当前版本不支持该接口。 Atlas 训练系列产品 ，当前版本不支持该接口。

输入说明	chn: int, 编码通道号。 • Atlas 推理系列产品, 编码通道号的取值范围: [0, 256), JPEG功能 and VENC功能共用通道, 且通道总数最多256。 • Atlas A2训练系列产品/Atlas 800I A2推理产品, 编码通道号的取值范围: [0, 256), JPEG功能通道总数最多256。 • Atlas 200I/500 A2推理产品, 编码通道号的取值范围: [0, 128), JPEG功能和VENC功能共用通道, 且通道总数最多128。 jpeg_param: dict, JPEG协议编码通道的高级参数集合, 具体请参见2.10.8.4.44 hi_venc_jpeg_param。
返回值说明	ret: int, 错误码。 • 返回0表示成功。 • 返回非0表示失败, 参见2.10.9.4 VENC视频/JPEG图像编码返回码。
约束说明	本接口在编码通道创建之后、编码通道销毁之前调用。 用户可以先调用acl.himpi.venc_get_jpeg_param获取当前参数, 然后对该参数进行修改后再调用本接口设置。 当hi_venc_jpeg_param字典内的“qfactor”属性值配置为0xFFFFFFFF, QT (Quantisation Table) 表的Cb分量 (Cb表示蓝色分量) 和Cr分量 (Cr表示红色分量) 需保持一致。

2.10.6.19 acl.himpi.venc_get_jpeg_param

c函数原型	hi_s32 hi_mpi_venc_get_jpeg_param(hi_venc_chn chn, hi_venc_jpeg_param *jpeg_param)
Python函数	jpeg_param, ret = acl.himpi.venc_get_jpeg_param(chn)
函数功能	获取JPEG协议编码通道的量化表高级配置参数。 Atlas 200/300/500 推理产品, 当前版本不支持该接口。 Atlas 训练系列产品, 当前版本不支持该接口。
输入说明	chn: int, 编码通道号。 • Atlas 推理系列产品, 编码通道号的取值范围: [0, 256), JPEG功能和VENC功能共用通道, 且通道总数最多256。 • Atlas A2训练系列产品/Atlas 800I A2推理产品, 编码通道号的取值范围: [0, 256), JPEG功能通道总数最多256。 • Atlas 200I/500 A2推理产品, 编码通道号的取值范围: [0, 128), JPEG功能和VENC功能共用通道, 且通道总数最多128。

返回值说明	ret: int, 错误码。 • 返回0表示成功。 • 返回非0表示失败, 参见2.10.9.4 VENC视频/JPEGE图像编码返回码。 jpeg_param: dict, JPEG协议编码通道的高级参数集合, 具体请参见2.10.8.4.44 hi_venc_jpeg_param。
约束说明	本接口在编码通道创建之后、编码通道销毁之前调用。 用户在调用acl.himpi.venc_set_jpeg_param前, 可以调用该接口获取当前JPEGE的量化表参数。

2.10.6.20 acl.himpi.venc_set_chn_param

c函数原型	hi_s32 hi_mpi_venc_set_chn_param(hi_venc_chn chn, const hi_venc_chn_param *chn_param);
Python函数	ret = acl.himpi.venc_set_chn_param(chn, chn_param)
函数功能	设置通道参数。 Atlas 200/300/500 推理产品, 当前版本不支持该接口。 Atlas 训练系列产品, 当前版本不支持该接口。 Atlas A2训练系列产品/Atlas 800I A2推理产品, 当前版本不支持该接口。 Atlas 200I/500 A2推理产品, JPEGE图片编码场景下, 不支持该接口。
输入说明	chn: int, 编码通道号。 • Atlas 推理系列产品, 编码通道号的取值范围: [0, 256), JPEGE功能和VENC功能共用通道, 且通道总数最多256。 • Atlas 200I/500 A2推理产品, 编码通道号的取值范围: [0, 128), JPEGE功能和VENC功能共用通道, 且通道总数最多128。 chn_param: dict, VENC的通道参数, 具体请参见2.10.8.4.52 hi_venc_chn_param。
返回值说明	ret: int, 错误码。 • 返回0表示成功。 • 返回非0表示失败, 参见2.10.9.4 VENC视频/JPEGE图像编码返回码。
约束说明	本接口在编码通道创建之后、编码通道销毁之前调用。如果针对未创建的通道, 调用本接口设置参数, 则返回失败。

2.10.6.21 acl.himpi.venc_get_chn_param

c函数原型	hi_s32 hi_mpi_venc_get_chn_param(hi_venc_chn chn, hi_venc_chn_param *chn_param);
Python函数	chn_param, ret = acl.himpi.venc_get_chn_param(chn)
函数功能	获取通道参数。 Atlas 200/300/500 推理产品 ，当前版本不支持该接口。 Atlas 训练系列产品 ，当前版本不支持该接口。 Atlas A2训练系列产品/Atlas 800I A2推理产品 ，当前版本不支持该接口。 Atlas 200I/500 A2推理产品 ，JPEG图片编码场景下，不支持该接口。
输入说明	chn : int，编码通道号。 Atlas 推理系列产品，编码通道号的取值范围：[0, 256)，JPEG功能和VENC功能共用通道，且通道总数最多256。 Atlas 200I/500 A2推理产品，编码通道号的取值范围：[0, 128)，JPEG功能和VENC功能共用通道，且通道总数最多128。
返回值说明	ret : int，错误码。 - 返回0表示成功。 - 返回非0表示失败，参见2.10.9.4 VENC视频/JPEG图像编码返回码。 chn_param : dict，VENC通道参数，具体请参见 2.10.8.4.52 hi_venc_chn_param 。
约束说明	本接口在编码通道创建之后、编码通道销毁之前调用。如果针对未创建的通道，调用本接口设置参数，则返回失败。

2.10.6.22 acl.himpi.venc_set_scene_mode

c函数原型	hi_s32 hi_mpi_venc_set_scene_mode(hi_venc_chn chn, const hi_venc_scene_mode scene_mode);
Python函数	ret = acl.himpi.venc_set_scene_mode(chn, scene_mode)
函数功能	设置场景模式。 Atlas 200/300/500 推理产品 ，当前版本不支持该接口。 Atlas 训练系列产品 ，当前版本不支持该接口。 Atlas A2训练系列产品/Atlas 800I A2推理产品 ，当前版本不支持该接口。

输入说明	chn: int, 编码通道号。 • Atlas 推理系列产品, 编码通道号的取值范围: [0, 256), JPEG功能 and VENC功能共用通道, 且通道总数最多256。 • Atlas 200I/500 A2推理产品, 编码通道号的取值范围: [0, 128), JPEG功能和VENC功能共用通道, 且通道总数最多128。 scene_mode: int, 场景模式, 具体请参见2.10.8.4.66 hi_venc_scene_mode。
返回值说明	ret: int, 错误码。 • 返回0表示成功。 • 返回非0表示失败, 参见2.10.9.4 VENC视频/JPEG图像编码返回码。
约束说明	• 不同的编码模式场景对图像质量有影响, 默认场景模式为“HI_VENC_SCENE_0”。 • 本接口在编码通道创建之后、编码通道销毁之前调用。

2.10.6.23 acl.himpi.venc_get_scene_mode

c函数原型	hi_s32 hi_mpi_venc_get_scene_mode(hi_venc_chn chn, hi_venc_scene_mode *scene_mode)
Python函数	scene_mode, ret = acl.himpi.venc_get_scene_mode(chn)
函数功能	获取场景模式。 Atlas 200/300/500 推理产品, 不支持该接口。 Atlas 训练系列产品, 不支持该接口。 Atlas A2训练系列产品/Atlas 800I A2推理产品, 不支持该接口。
输入说明	chn: int, 编码通道号。 • Atlas 推理系列产品, 编码通道号的取值范围: [0, 256), JPEG功能和VENC功能共用通道, 且通道总数最多256。 • Atlas A2训练系列产品/Atlas 800I A2推理产品, 编码通道号的取值范围: [0, 256), JPEG功能通道总数最多256。 • Atlas 200I/500 A2推理产品, 编码通道号的取值范围: [0, 128), JPEG功能和VENC功能共用通道, 且通道总数最多128。
返回值说明	scene_mode: int, 当前场景模式。详情请参见2.10.8.4.66 hi_venc_scene_mode。 ret: int, 错误码。 • 返回0表示成功。 • 返回非0表示失败, 参见2.10.9.4 VENC视频/JPEG图像编码返回码。
约束说明	本接口在编码通道创建之后、编码通道销毁之前调用。

2.10.6.24 acl.himpi.venc_set_rc_param

c函数原型	hi_s32 hi_mpi_venc_set_rc_param(hi_venc_chn chn, const hi_venc_rc_param *rc_param);
Python函数	ret = acl.himpi.venc_set_rc_param(chn, rc_param)
函数功能	设置编码通道码率控制器的高级参数的值。 Atlas 200/300/500 推理产品 ，当前版本不支持该接口。 Atlas 训练系列产品 ，当前版本不支持该接口。 Atlas A2训练系列产品/Atlas 800I A2推理产品 ，当前版本不支持该接口。
输入说明	chn : int，编码通道号。 • Atlas 推理系列产品，编码通道号的取值范围：[0, 256)，JPEG功能和VENC功能共用通道，且通道总数最多256。 • Atlas 200I/500 A2推理产品，编码通道号的取值范围：[0, 128)，JPEG功能和VENC功能共用通道，且通道总数最多128。 rc_param : dict，编码通道码率控制器的高级参数字典，具体请参见 2.10.8.4.67 hi_venc_rc_param 。
返回值说明	ret : int，错误码。 • 返回0表示成功。 • 返回非0表示失败，参见2.10.9.4 VENC视频/JPEG图像编码返回码。
约束说明	• 本接口在编码通道创建之后、编码通道销毁之前设置。 • 如果不调用本接口设置编码通道码率控制器的高级参数的值，就会默认采用这些高级参数的默认值。 • 建议用户先调用acl.himpi.venc_get_rc_param接口获取编码通道码率控制器的高级参数的值，再调用本接口对高级参数进行设置。

2.10.6.25 acl.himpi.venc_get_rc_param

c函数原型	hi_s32 hi_mpi_venc_get_rc_param(hi_venc_chn chn, hi_venc_rc_param *rc_param);
Python函数	rc_param, ret = acl.himpi.venc_get_rc_param(chn, mode)

函数功能	获取编码通道码率控制器的高级参数的值。 Atlas 200/300/500 推理产品 ，当前版本不支持该接口。 Atlas 训练系列产品 ，当前版本不支持该接口。 Atlas A2训练系列产品/Atlas 800I A2推理产品 ，当前版本不支持该接口。
输入说明	chn : int, 编码通道号。 • Atlas 推理系列产品，编码通道号的取值范围：[0, 256)，JPEG功能和VENC功能共用通道，且通道总数最多256。 • Atlas 200I/500 A2推理产品，编码通道号的取值范围：[0, 128)，JPEG功能和VENC功能共用通道，且通道总数最多128。 mode : int, 编码通道码率控制器模式，具体请参见 2.10.8.4.6 hi_venc_rc_mode 。
返回值说明	ret : int, 错误码。 • 返回0表示成功。 • 返回非0表示失败，参见2.10.9.4 VENC视频/JPEG图像编码返回码。 rc_param : dict, 编码通道码率控制器的高级参数字典，具体请参见 2.10.8.4.67 hi_venc_rc_param 。
约束说明	本接口在编码通道创建之后、编码通道销毁之前调用。

2.10.6.26 acl.himpi.venc_set_jpeg_huffman_param

c函数原型	hi_s32 hi_mpi_venc_set_jpeg_huffman_param(hi_venc_chn chn, const hi_venc_jpeg_huffman_param *jpeg_huffman_param);
Python函数	ret = acl.himpi.venc_set_jpeg_huffman_param(chn, jpeg_huffman_param)
函数功能	设置编码通道的JPEG huffman编码高级参数。 Atlas 200/300/500 推理产品 ，当前版本不支持该接口。 Atlas 训练系列产品 ，当前版本不支持该接口。
输入说明	chn : int, 编码通道号。 • Atlas 推理系列产品，编码通道号的取值范围：[0, 256)，JPEG功能和VENC功能共用通道，且通道总数最多256。 • Atlas A2训练系列产品/Atlas 800I A2推理产品，编码通道号的取值范围：[0, 256)，JPEG功能通道总数最多256。 • Atlas 200I/500 A2推理产品，编码通道号的取值范围：[0, 128)，JPEG功能和VENC功能共用通道，且通道总数最多128。 jpeg_huffman_param : dict, huffman编码参数字典，具体请参见 2.10.8.4.81 hi_venc_jpeg_huffman_param 。

返回值说明	ret : int, 错误码。 - 返回0表示成功。 - 返回非0表示失败, 参见2.10.9.4 VENC视频/JPEGE图像编码返回码。
约束说明	本接口在编码通道创建之后、编码通道销毁之前调用。

2.10.6.27 acl.himpi.venc_get_jpeg_huffman_param

c函数原型	hi_s32 hi_mpi_venc_get_jpeg_huffman_param(hi_venc_chn chn, hi_venc_jpeg_huffman_param *jpeg_huffman_data);
Python函数	jpeg_huffman_data, ret = acl.himpi.venc_get_jpeg_huffman_param(chn)
函数功能	获取编码通道的JPEGE huffman编码高级参数。 Atlas 200/300/500 推理产品 , 当前版本不支持该接口。 Atlas 训练系列产品 , 当前版本不支持该接口。
输入说明	chn : int, 编码通道号。 Atlas 推理系列产品, 编码通道号的取值范围: [0, 256), JPEGE功能和VENC功能共用通道, 且通道总数最多256。 Atlas A2训练系列产品/Atlas 800I A2推理产品, 编码通道号的取值范围: [0, 256), JPEGE功能通道总数最多256。 Atlas 200I/500 A2推理产品, 编码通道号的取值范围: [0, 128), JPEGE功能和VENC功能共用通道, 且通道总数最多128。
返回值说明	ret : int, 错误码。 - 返回0表示成功。 - 返回非0表示失败, 参见2.10.9.4 VENC视频/JPEGE图像编码返回码。 jpeg_huffman_data : dict, huffman高级参数字典, 具体请参见 2.10.8.4.81 hi_venc_jpeg_huffman_param 。
约束说明	本接口在编码通道创建之后、编码通道销毁之前调用。 不调用 acl.himpi.venc_set_jpeg_huffman_param 接口设置huffman编码高级参数时, 返回默认的标准huffman编码参数。

2.10.6.28 acl.himpi.venc_compact_jpeg_tables

c函数原型	hi_s32 hi_mpi_venc_compact_jpeg_tables(hi_venc_chn chn, hi_u32 table_type, hi_bool enable);
Python函数	ret = acl.himpi.venc_compact_jpeg_tables(chn, table_type, enable)

函数功能	配置是否对高级参数huffman或QT表进行压缩。 Atlas 200/300/500 推理产品 ，当前版本不支持该接口。 Atlas 训练系列产品 ，当前版本不支持该接口。
输入说明	chn : int, 编码通道号。 • Atlas 推理系列产品，编码通道号的取值范围：[0, 256)，JPEG功能 and VENC功能共用通道，且通道总数最多256。 • Atlas A2训练系列产品/Atlas 800I A2推理产品，编码通道号的取值范围：[0, 256)，JPEG功能通道总数最多256。 • Atlas 200I/500 A2推理产品，编码通道号的取值范围：[0, 128)，JPEG功能和VENC功能共用通道，且通道总数最多128。 table_type : int, 压缩类型，仅支持设置为0，表示huffman表数据和QT量化表数据都压缩。 enable : int, 是否压缩，0表示不压缩，1表示压缩。
返回值说明	ret : int, 错误码。 • 返回0表示成功。 • 返回非0表示失败，参见2.10.9.4 VENC视频/JPEG图像编码返回码。
约束说明	本接口在编码通道创建之后、编码通道销毁之前调用。 调用此接口后，如果huffman表或QT表的Y、Cb、Cr分量内容一致时，支持压缩。

2.10.6.29 acl.himpi.venc_send_jpege_frame

c函数原型	hi_s32 hi_mpi_venc_send_jpege_frame(hi_venc_chn chn, const hi_video_frame_info *frame, const hi_img_stream *jpege_stream, hi_s32 milli_sec)
Python函数	ret = acl.himpi.venc_send_jpege_frame(chn, frame, jpege_stream, milli_sec)
函数功能	发送原始图像进行图像编码，并支持配置编码结果的输出内存地址。 Atlas 200/300/500 推理产品 ，当前版本不支持该接口。 Atlas 训练系列产品 ，当前版本不支持该接口。

输入说明	chn: int, 编码通道号。 • Atlas 推理系列产品, 编码通道号的取值范围: [0, 256), JPEG功能 and VENC功能共用通道, 且通道总数最多256。 • Atlas A2训练系列产品/Atlas 800I A2推理产品, 编码通道号的取值范围: [0, 256), JPEG功能通道总数最多256。 • Atlas 200I/500 A2推理产品, 编码通道号的取值范围: [0, 128), JPEG功能和VENC功能共用通道, 且通道总数最多128。 frame: dict, 原始图像信息字典, 具体请参见2.10.8.2.9 hi_video_frame_info。 jpege_stream: dict, 用于指定编码结果的输出内存地址和长度, 具体请参见2.10.8.6.2 hi_img_stream。 输出内存首地址需满足16字节对齐, 内存大小可以调用acl.himpi.venc_get_jpege_predicted_size获取, 再可调用acl.himpi.dvpp_malloc接口申请输出内存。 milli_sec: int, 超时时间, 单位是毫秒。 • “-1”: 阻塞方式。 • “0”: 非阻塞方式。 • >0: 超时方式, 配置具体的超时时间。超时时间受操作系统影响, 一般偏差在操作系统的一个时间片内, 例如, 操作系统的一个时间片为4ms, 用户设置的“milli_sec”参数值为“1”, 则实际的超时时间在1ms到5ms范围内。在CPU负载高场景下, 超时时间仍可能存在波动。
返回值说明	ret: int, 错误码。 • 返回0表示成功。 • 返回非0表示失败, 参见2.10.9.4 VENC视频/JPEG图像编码返回码。
约束说明	• 在调用2.10.6.5 acl.himpi.venc_create_chn接口创建通道时, 将参数hi_venc_chn_attr.get("venc_attr")["buf_size"]参数值设置为0, 本接口才有效, 否则返回错误码“HI_ERR_VENC_NOT_SUPPORT”。 • JPEG输入图像分辨率必须小于或等于编码通道的宽、高。 • 调用该接口发送图像, 用户需要保证编码通道已创建且开启接收输入图片。 • 编码时, 输入、输出内存均需要在调用acl.himpi.venc_get_stream接口获取结果之后才能进行释放。 • 图像编码, 对输入、输出的约束请参见2.10.6.1 JPEG功能及约束说明。
参考资源	接口调用流程及示例, 参见JPEG图片编码、VENC视频编码。

2.10.6.30 acl.himpi.venc_get_jpege_predicted_size

c函数原型	hi_s32 hi_mpi_venc_get_jpege_predicted_size(const hi_video_frame_info *frame, const hi_venc_jpeg_param *jpeg_param, hi_u32 *size)
Python函数	size, ret = acl.himpi.venc_get_jpege_predicted_size(frame, jpeg_param)
函数功能	预估JPEG编码一帧图片所需的输出缓冲区大小。 Atlas 200/300/500 推理产品 ，当前版本不支持该接口。 Atlas 训练系列产品 ，当前版本不支持该接口。
输入说明	frame : int，原始图像信息字典，具体请参见 2.10.8.2.9 hi_video_frame_info 。 jpeg_param : dict，JPEG协议编码通道的高级参数集合字典，预留参数。
返回值说明	ret : int，错误码。 - 返回0表示成功。 - 返回非0表示失败，参见2.10.9.4 VENC视频/JPEG图像编码返回码。 size : 输出缓冲区大小，单位为Byte。

2.10.6.31 acl.himpi.venc_get_cu_pred

c函数原型	hi_s32 hi_mpi_venc_get_cu_pred(hi_venc_chn chn, hi_venc_cu_prediction *cu_pred)
Python函数	cu_pred, ret = acl.himpi.venc_get_cu_pred(chn)
函数功能	获取通道CU/MB的模式选择的倾向性设置。 Atlas 200/300/500 推理产品 ，不支持该接口。 Atlas 训练系列产品 ，不支持该接口。 Atlas A2训练系列产品/Atlas 800I A2推理产品 ，不支持该接口。
输入说明	chn : int，编码通道号。 Atlas 推理系列产品，编码通道号的取值范围：[0, 256)，JPEG功能和VENC功能共用通道，且通道总数最多256。 Atlas 200I/500 A2推理产品，编码通道号的取值范围：[0, 128)，JPEG功能和VENC功能共用通道，且通道总数最多128。

返回值说明	cu_pred: dict, CU模式选择的倾向性字典。具体参见2.10.8.4.94 hi_venc_cu_prediction。 ret: int, 错误码。 • 返回0表示成功。 • 返回非0表示失败, 参见2.10.9.4 VENC视频/JPEGE图像编码返回码。
约束说明	• 本接口用于获取CU模式选择的倾向性配置。 • 本接口在编码通道创建之后, 编码通道销毁之前调用。 • 仅H.265/H.264编码通道支持。

2.10.6.32 acl.himpi.venc_set_cu_pred

c函数原型	hi_s32 hi_mpi_venc_set_cu_pred(hi_venc_chn chn, const hi_venc_cu_prediction *cu_pred)
Python函数	ret = acl.himpi.venc_set_cu_pred(chn, cu_pred)
函数功能	本接口用于调节通道CU/MB的模式选择的倾向性, 支持调节帧间预测 (Inter) 和帧内预测 (Intra) 的倾向性配置以及编码块大小的倾向性配置。 编码协议以像素块作为基本处理单元, H.265/H.264中分别称为CU(Coding Unit)/MB(MacroBlock)。本接口中的CU模式指Inter/Intra预测模式中对不同尺寸像素块的选择。 Atlas 200/300/500 推理产品, 不支持该接口。 Atlas 训练系列产品, 不支持该接口。 Atlas A2训练系列产品/Atlas 800I A2推理产品, 不支持该接口。
输入说明	chn: int, 编码通道号。 • Atlas 推理系列产品, 编码通道号的取值范围: [0, 256), JPEGE功能和VENC功能共用通道, 且通道总数最多256。 • Atlas 200I/500 A2推理产品, 编码通道号的取值范围: [0, 128), JPEGE功能和VENC功能共用通道, 且通道总数最多128。 cu_pred: dict, CU模式选择的倾向性参数字典。具体参见2.10.8.4.94 hi_venc_cu_prediction。 该字典内需包含以下参数: • pred_mode: 倾向性选择模式, 支持auto和manual两种模式。auto模式下由驱动内部完成倾向性配置, 即用户的配置参数不生效; manual模式允许用户根据实际场景完成自定义的倾向性配置。 • intra32_cost/intra16_cost/intra8_cost/intra4_cost/inter64_cost/inter32_cost/inter16_cost/inter8_cost分别设置各自模式的倾向性大小。默认每个cost值为8, 即不做倾向性配置。每个模式的cost设置越大意味着选择该模式的倾向性越小。

返回值说明	ret : int, 错误码。 • 返回0表示成功。 • 返回非0表示失败, 参见2.10.9.4 VENC视频/JPEGE图像编码返回码。
约束说明	• 仅H.265/H.264编码通道支持。 • 建议用户在创建通道之后, 启动编码之前调用本接口, 减少编码过程中调用本接口的次数。若在编码过程中调用本接口, 则下一帧编码时配置生效。 • 建议用户在调用此接口之前先调用acl.himpi.venc_get_cu_pred接口, 获取CU模式选择的倾向性配置, 然后再进行设置。 • 注意某些场景下特定模式不使能 (例如, H.264中最大MB尺寸为16*16, 且baseline和main profile下inter16_cost不使能, high profile下inter8_cost不使能), 即使用户设置了这些特定模式的倾向性参数也不生效。

2.10.6.33 acl.himpi.venc_get_h264_vui

c函数原型	hi_s32 hi_mpi_venc_get_h264_vui(hi_venc_chn chn, hi_venc_h264_vui *h264_vui)
Python函数	h264_vui, ret = acl.himpi.venc_get_h264_vui(chn)
函数功能	获取H.264协议编码通道的VUI配置。 Atlas 200/300/500 推理产品 , 不支持该接口。 Atlas 训练系列产品 , 不支持该接口。 Atlas A2训练系列产品/Atlas 800I A2推理产品 , 不支持该接口。
输入说明	chn : int, 编码通道号。 • Atlas 推理系列产品, 编码通道号的取值范围: [0, 256), JPEG功能 and VENC功能共用通道, 且通道总数最多256。 • Atlas 200I/500 A2推理产品, 编码通道号的取值范围: [0, 128), JPEG功能 and VENC功能共用通道, 且通道总数最多128。
返回值说明	h264_vui : dict, H.264协议编码通道的VUI属性。具体请参见 2.10.8.4.87 hi_venc_h264_vui 。 ret : int, 错误码。 • 返回0表示成功。 • 返回非0表示失败, 参见2.10.9.4 VENC视频/JPEGE图像编码返回码。

约束说明	● 本接口用于获取H.264协议编码通道的VUI配置。 ● 本接口在编码通道创建之后，编码通道销毁之前调用。 ● 建议用户在创建通道之后，启动编码之前调用此接口，减少在编码过程中调用的次数。 ● 用户应当仅在H.264协议编码通道中调用本接口。

2.10.6.34 acl.himpi.venc_set_h264_vui

c函数原型	hi_s32 hi_mpi_venc_set_h264_vui(hi_venc_chn chn, const hi_venc_h264_vui *h264_vui)
Python函数	ret = acl.himpi.venc_set_h264_vui(chn, h264_vui)
函数功能	设置H.264协议编码通道的VUI参数。 VUI (Video usability information) 参数主要包含解码图像的格式信息，比如采样横纵比、光电转换特性、颜色空间等。VUI参数在协议中属于可选参数，不影响视频编码过程，但会建议解码器/播放器做一些矫正处理。 Atlas 200/300/500 推理产品，不支持该接口。 Atlas 训练系列产品，不支持该接口。 Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。
输入说明	chn: int，编码通道号。 ● Atlas 推理系列产品，编码通道号的取值范围：[0, 256)，JPEG功能和VENC功能共用通道，且通道总数最多256。 ● Atlas 200I/500 A2推理产品，编码通道号的取值范围：[0, 128)，JPEG功能和VENC功能共用通道，且通道总数最多128。 h264_vui: dict，H.264协议编码通道的VUI参数字典，具体请参见2.10.8.4.87 hi_venc_h264_vui。
返回值说明	ret: int，错误码。 ● 返回0表示成功。 ● 返回非0表示失败，参见2.10.9.4 VENC视频/JPEG图像编码返回码。

约束说明	● 本接口用于设置H.264协议编码通道的VUI配置。 ● 本接口在编码通道创建之后，编码通道销毁之前设置。此接口在编码过程中被调用时，等到下一个I帧时生效。 ● 建议用户在创建通道之后，启动编码之前调用此接口，减少在编码过程中调用的次数。 ● 用户应当仅在H.264协议编码通道中调用本接口。 ● 目前仅开放“hi_venc_h264_vui”中的“hi_venc_vui_video_signal”属性供用户配置，其它参数预留，均需配置为默认值，否则会导致编码失败。建议用户在调用此接口之前先调用acl.himpi.venc_get_h264_vui接口，获取当前编码通道的VUI默认配置，然后再进行设置，避免预留参数出错。 ● hi_venc_vui_h264_time_info包含时钟计数相关参数，默认情况下为不生效状态，用户想要控制显示帧率时需要主动配置相关参数。注意，显示帧率只影响播放器的播放速度，若用户想设置编码过程中的码控帧率需配置acl.himpi.venc_create_chn接口中的输入/输出帧率参数。 建议用户将码控帧率与显示帧率设置一致，否则播放时的码率可能与用户设置的编码码率不同，播放时的码率会随着所设置的显示帧率而变化。 如果用户需要调整播放速度，则建议在合理范围内调整显示帧率，一般正常视频播放速度为30fps，表示显示帧率为30fps。 - 若将显示帧率调大，则播放速度加快，例如3倍速播放，则显示帧率设置为90fps。 - 若将显示帧率调小，则播放速度减慢，例如0.5倍速播放，则显示帧率设置为15fps。 ● 由于VUI只是协议可选参数而非强制参数，调用本接口设置参数（如显示帧率、像素值域范围等）的最终效果取决于解码器/播放器行为，或者用户自定义的解码/播放行为。

2.10.6.35 `acl.himpi.venc_get_h265_vui`

c函数原型	<code>hi_s32 hi_mpi_venc_get_h265_vui(hi_venc_chn chn, hi_venc_h265_vui *h265_vui)</code>
Python函数	<code>h265_vui, ret = acl.himpi.venc_get_h265_vui(chn)</code>
函数功能	获取H.265协议编码通道的VUI配置。 Atlas 200/300/500 推理产品 ，不支持该接口。 Atlas 训练系列产品 ，不支持该接口。 Atlas A2训练系列产品/Atlas 800I A2推理产品 ，不支持该接口。

输入说明	chn: int, 编码通道号。 • Atlas 推理系列产品, 编码通道号的取值范围: [0, 256), JPEG功能 and VENC功能共用通道, 且通道总数最多256。 • Atlas 200I/500 A2推理产品, 编码通道号的取值范围: [0, 128), JPEG功能 and VENC功能共用通道, 且通道总数最多128。
返回值说明	h265_vui: dict, H.265协议编码通道的VUI属性字典。参见2.10.8.4.88 hi_venc_h265_vui。 ret: int, 错误码。 • 返回0表示成功。 • 返回非0表示失败, 参见2.10.9.4 VENC视频/JPEG图像编码返回码。
约束说明	• 本接口用于获取H.265协议编码通道的VUI配置。 • 本接口在编码通道创建之后, 编码通道销毁之前调用。 • 建议用户在创建通道之后, 启动编码之前调用此接口, 减少在编码过程中调用的次数。 • 用户应当仅在H.265协议编码通道中调用本接口。

2.10.6.36 acl.himpi.venc_set_h265_vui

c函数原型	hi_s32 hi_mpi_venc_set_h265_vui(hi_venc_chn chn, const hi_venc_h265_vui *h265_vui)
Python函数	ret = acl.himpi.venc_set_h265_vui(chn, h265_vui)
函数功能	设置H.265协议编码通道的VUI参数。 VUI (Video usability information) 参数主要包含解码图像的格式信息, 比如采样纵横比、光电转换特性、颜色空间等。VUI参数在协议中属于可选参数, 不影响视频编码过程, 但会建议解码器/播放器做一些矫正处理。 Atlas 200/300/500 推理产品, 不支持该接口。 Atlas 训练系列产品, 不支持该接口。 Atlas A2训练系列产品/Atlas 800I A2推理产品, 不支持该接口。
输入说明	chn: int, 编码通道号。 • Atlas 推理系列产品, 编码通道号的取值范围: [0, 256), JPEG功能 and VENC功能共用通道, 且通道总数最多256。 • Atlas 200I/500 A2推理产品, 编码通道号的取值范围: [0, 128), JPEG功能 and VENC功能共用通道, 且通道总数最多128。 h265_vui: dict, H.265协议编码通道的VUI属性字典, 参见2.10.8.4.88 hi_venc_h265_vui。

返回值说明	ret: int, 错误码。 • 返回0表示成功。 • 返回非0表示失败, 参见2.10.9.4 VENC视频/JPEGE图像编码返回码。
约束说明	• 本接口用于设置H.265协议编码通道的VUI配置。 • 本接口在编码通道创建之后, 编码通道销毁之前设置。此接口在编码过程中被调用时, 等到下一个I帧时生效。 • 建议用户在创建通道之后, 启动编码之前调用此接口, 减少在编码过程中调用的次数。 • 用户应当仅在H.265协议编码通道中调用本接口。 • 目前仅开放hi_venc_h265_vui中的“hi_venc_vui_video_signal”属性和“hi_venc_vui_h265_time_info”属性供用户配置, 其它属性预留, 均需配置为默认值, 否则会导致编码失败。建议用户在调用此接口之前先调用acl.himpi.venc_get_h265_vui接口, 获取当前编码通道的VUI默认配置, 然后再进行设置, 避免预留属性出错。 • hi_venc_vui_h265_time_info参数包含时钟计数相关参数, 默认情况下为不生效状态, 用户想要控制显示帧率时需要主动配置相关参数。注意, 此处的显示帧率只影响播放器的播放速度, 用户想控制编码过程中的码控帧率需配置acl.himpi.venc_create_chn接口中的“hi_venc_rc_attr”属性。 建议用户将码控帧率与显示帧率设置一致, 否则播放时的码率可能与用户设置的编码码率不同, 播放时的码率会随着所设置的显示帧率而变化。 如果用户需要调整播放速度, 则建议在合理范围内调整显示帧率, 一般正常视频播放速度为30fps, 表示显示帧率为30fps。 - 若将显示帧率调大, 则播放速度加快, 例如3倍速播放, 则显示帧率设置为90fps。 - 若将显示帧率调小, 则播放速度减慢, 例如0.5倍速播放, 则显示帧率设置为15fps。 • 由于VUI只是协议可选参数而非强制参数, 本接口设置参数(如显示帧率、像素值域范围等)的最终效果取决于解码器/播放器行为, 或者用户自定义的解码/播放行为。

2.10.6.37 [acl.himpi.venc_get_intra_refresh](#)

c函数原型	<code>hi_s32 hi_mpi_venc_get_intra_refresh(hi_venc_chn chn, hi_venc_intra_refresh *intra_refresh)</code>
Python函数	<code>intra_refresh, ret = acl.himpi.venc_get_intra_refresh(chn)</code>

函数功能	获取P帧刷新Islice的设置参数。 Atlas 200/300/500 推理产品 ，不支持该接口。 Atlas 训练系列产品 ，不支持该接口。 Atlas A2训练系列产品/Atlas 800I A2推理产品 ，不支持该接口。
输入说明	chn : int, 编码通道号。 • Atlas 推理系列产品，编码通道号的取值范围：[0, 256)，JPEG功能和VENC功能共用通道，且通道总数最多256。 • Atlas 200I/500 A2推理产品，编码通道号的取值范围：[0, 128)，JPEG功能和VENC功能共用通道，且通道总数最多128。
返回值说明	intra_refresh : dict, 刷Islice的参数字典。参见 2.10.8.4.95 hi_venc_intra_refresh 。 ret : int, 错误码。 • 返回0表示成功。 • 返回非0表示失败，参见2.10.9.4 VENC视频/JPEG图像编码返回码。
约束说明	• 本接口必须在编码通道创建之后，编码通道销毁之前调用。 • 切换GOP模式时，获取的属性是已设置的最新的属性。

2.10.6.38 acl.himpi.venc_set_intra_refresh

c函数原型	hi_s32 hi_mpi_venc_set_intra_refresh(hi_venc_chn chn, const hi_venc_intra_refresh *intra_refresh)
Python函数	ret = acl.himpi.venc_set_intra_refresh(chn, intra_refresh)
函数功能	设置P帧刷新Islice的参数。 Atlas 200/300/500 推理产品 ，不支持该接口。 Atlas 训练系列产品 ，不支持该接口。 Atlas A2训练系列产品/Atlas 800I A2推理产品 ，不支持该接口。
输入说明	chn : int, 编码通道号。 • Atlas 推理系列产品，编码通道号的取值范围：[0, 256)，JPEG功能和VENC功能共用通道，且通道总数最多256。 • Atlas 200I/500 A2推理产品，编码通道号的取值范围：[0, 128)，JPEG功能和VENC功能共用通道，且通道总数最多128。 intra_refresh : dict, 刷Islice的参数字典。参见 2.10.8.4.95 hi_venc_intra_refresh 。

返回值说明	ret : int, 错误码。 • 返回0表示成功。 • 返回非0表示失败, 参见2.10.9.4 VENC视频/JPEGE图像编码返回码。
约束说明	• 仅支持H.265/H.264、GOP模式为“HI_VENC_GOP_MODE_NORMAL_P”的编码通道。 • 调用acl.himpi.venc_set_ref_param接口设置高级跳帧参考后需要重新进行设置, 不支持高级跳帧参考“pred_en”为0。 • 建议在创建通道后、启动编码前先调用acl.himpi.venc_get_intra_refresh接口获取P帧刷Islice的设置参数后, 再调用本接口设置P帧刷Islice的参数。 • 调用本接口使能P帧刷Islice时, 会生成一个I帧, 且需等下一个GOP的IDR帧才能生效; 关闭P帧刷Islice参数时会立即生效。

2.10.6.39 acl.himpi.venc_get_ref_param

c函数原型	hi_s32 hi_mpi_venc_get_ref_param(hi_venc_chn chn, const hi_venc_ref_param *ref_param)
Python函数	ref_param, ret = acl.himpi.venc_get_ref_param(chn)
函数功能	获取H.264/H.265编码通道高级跳帧参考参数。关于高级跳帧模式的详细说明请参见 高级跳帧模式说明 。 Atlas 200/300/500 推理产品 , 不支持该接口。 Atlas 训练系列产品 , 不支持该接口。 Atlas A2训练系列产品/Atlas 800I A2推理产品 , 不支持该接口。
输入说明	chn : int, 编码通道号。 • Atlas 推理系列产品, 编码通道号的取值范围: [0, 256), JPEGE功能和VENC功能共用通道, 且通道总数最多256。 • Atlas 200I/500 A2推理产品, 编码通道号的取值范围: [0, 128), JPEGE功能和VENC功能共用通道, 且通道总数最多128。
返回值说明	ref_param : dict, H.264/H.265编码通道高级跳帧参考参数字典。参见 2.10.8.4.86 hi_venc_ref_param 。 ret : int, 错误码。 • 返回0表示成功。 • 返回非0表示失败, 参见2.10.9.4 VENC视频/JPEGE图像编码返回码。
约束说明	本接口在编码通道创建之后、编码通道销毁之前调用。

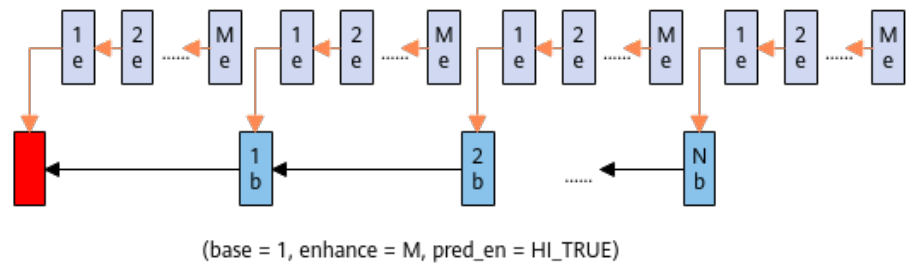
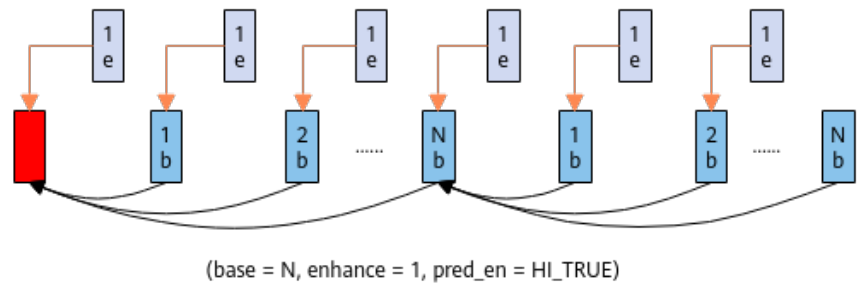
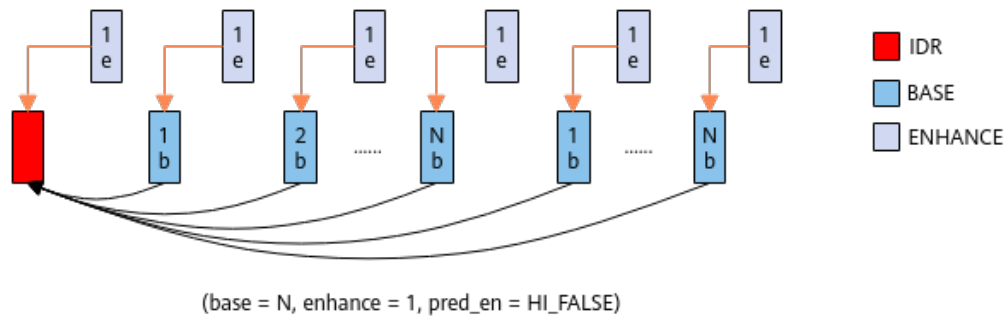
2.10.6.40 acl.himpi.venc_set_ref_param

c函数原型	hi_s32 hi_mpi_venc_set_ref_param(hi_venc_chn chn, const hi_venc_ref_param *ref_param)
Python函数	ret = acl.himpi.venc_set_ref_param(chn, ref_param)
函数功能	设置H.264/H.265编码通道高级跳帧参考参数，用于改变帧间参考关系。关于高级跳帧模式的详细说明请参见高级跳帧模式说明。 Atlas 200/300/500 推理产品，不支持该接口。 Atlas 训练系列产品，不支持该接口。 Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。
输入说明	chn: int，编码通道号。 • Atlas 推理系列产品，编码通道号的取值范围：[0, 256)，JPEG功能 and VENC功能共用通道，且通道总数最多256。 • Atlas 200I/500 A2推理产品，编码通道号的取值范围：[0, 128)，JPEG功能 and VENC功能共用通道，且通道总数最多128。 ref_param: dict，H.264/H.265编码通道高级跳帧参考参数字典。参见2.10.8.4.86 hi_venc_ref_param。
返回值说明	ret: int，错误码。 • 返回0表示成功。 • 返回非0表示失败，参见2.10.9.4 VENC视频/JPEG图像编码返回码。
约束说明	• 创建H.264/H.265协议编码通道时，默认跳帧参考模式是1倍跳帧参考模式。如果用户需要修改变码通道的跳帧参考，建议在创建通道之后，启动编码之前调用此接口，减少在编码过程中调用的次数。 • 本接口在编码过程中被调用时，等到下一个I帧时生效。 • 如果用户设置1倍跳帧参考模式，对应的配置为：pred_en = HI_TRUE，enhance = 0，base = 1；如果设置2倍跳帧参考模式，对应的配置为：pred_en = HI_TRUE，enhance = 1，base = 1；如果设置4倍跳帧参考模式，对应的配置为：pred_en = HI_TRUE，enhance = 1，base = 2。 • 如果用户设置通道的Gop Mode为“HI_VENC_GOP_MODE_DUAL_P”时，“sp_interval”只支持设置为sp_interval = (enhance + 1) * base，其中，当“sp_interval”不为0时，“pred_en”只支持设置为“HI_TRUE”。 • 如果用户设置通道的Gop Mode为“HI_VENC_GOP_MODE_SMART_P”时，“pred_en”只支持设置为“HI_TRUE”。 • 如果用户设置通道的Gop Mode为“HI_VENC_GOP_MODE_BIPREDB”时，“enhance”必须等于“b_frame_num”。

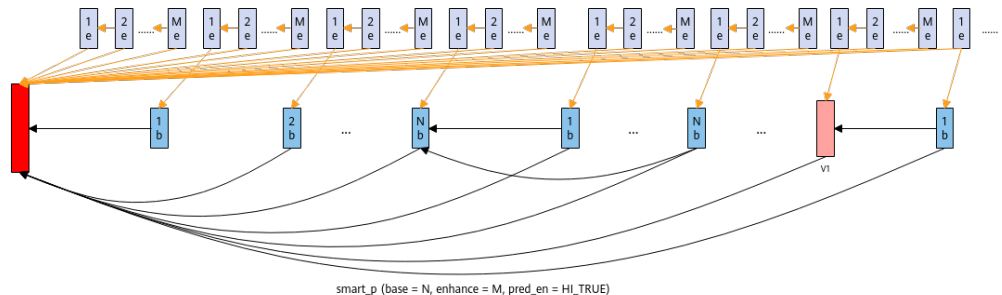
高级跳帧模式说明

高级跳帧参考模式涉及3个参数：base、enhance 和pred_en，其含义请参考2.10.8.4.86 hi_venc_ref_param。高级跳帧参考模式示意图如下图所示。

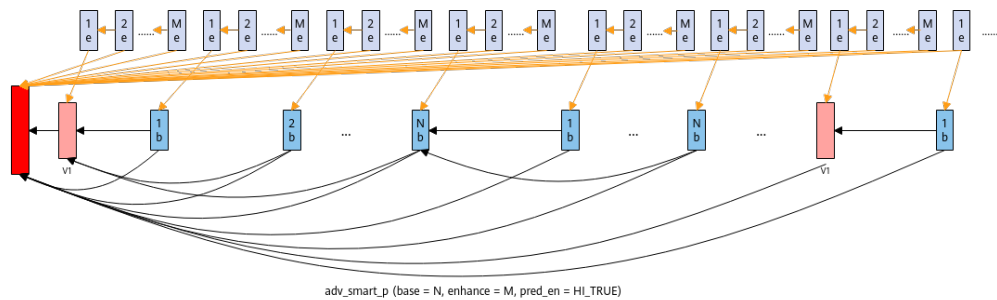
- 注意：dual_p 模式且sp_interval不等于0下配置高级跳帧参考需要满足如下条件： $base * (enhance + 1) = sp\_interval$ ，例如：4 倍高级跳帧参考（base=2，enhance=1），那么dual_p的sp_interval = $2 * (1 + 1) = 4$ 。
- normal_p高级跳帧参考示意图



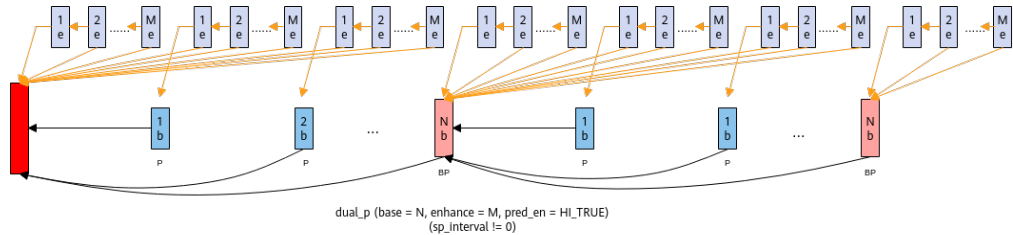
- smart_p高级跳帧参考示意图



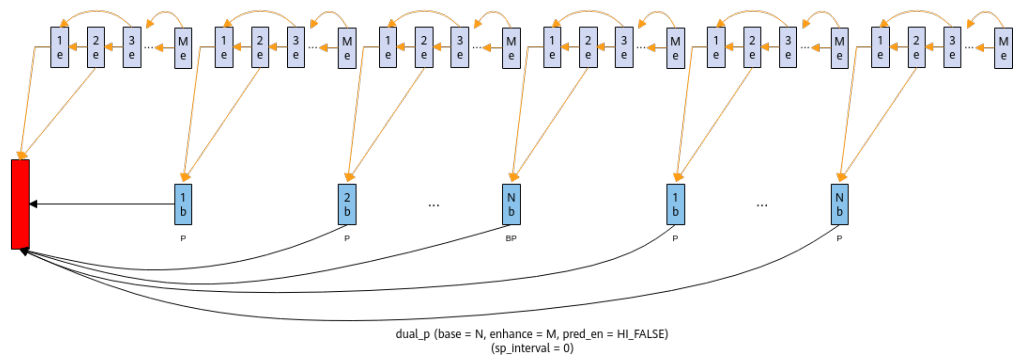
- adv_smart_p高级跳帧参考示意图



• dual_p (sp_interval!=0)高级跳帧参考示意图



• dual_p (sp_interval=0)高级跳帧参考示意图



2.10.6.41 acl.himpi.venc_get_roi_attr

c函数原型	hi_s32 hi_mpi_venc_get_roi_attr(hi_venc_chn chn, hi_u32 idx, hi_venc_roi_attr *roi_attr)
Python函数	roi_attr, ret = acl.himpi.venc_get_roi_attr(chn, idx)
函数功能	获取H.264/H.265通道的ROI配置属性。关于ROI编码功能的详细说明请参见 ROI编码功能说明 。 Atlas 200/300/500 推理产品，不支持该接口。 Atlas 训练系列产品，不支持该接口。 Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。

输入说明	chn: int, 编码通道号。 • Atlas 推理系列产品, 编码通道号的取值范围: [0, 256), JPEG功能 and VENC功能共用通道, 且通道总数最多256。 • Atlas 200I/500 A2推理产品, 编码通道号的取值范围: [0, 128), JPEG功能 and VENC功能共用通道, 且通道总数最多128。 idx: int, H.264/H.265协议编码通道ROI区域索引。
返回值说明	roi_attr: dict, ROI区域参数字典。参见2.10.8.4.85 hi_venc_roi_attr。 ret: int, 错误码。 • 返回0表示成功。 • 返回非0表示失败, 参见2.10.9.4 VENC视频/JPEG图像编码返回码。
约束说明	• 本接口用于获取H.264/H.265协议编码通道的指定索引号的ROI配置。 • 本接口在编码通道创建之后, 编码通道销毁之前调用。 • 建议用户在创建通道之后, 启动编码之前调用此接口, 减少在编码过程中调用的次数。

2.10.6.42 acl.himpi.venc_set_roi_attr

c函数原型	hi_s32 hi_mpi_venc_set_roi_attr(hi_venc_chn chn, const hi_venc_roi_attr *roi_attr)
Python函数	ret = acl.himpi.venc_set_roi_attr(chn, roi_attr)
函数功能	设置H.264/H.265通道的ROI属性, 用户可以通过配置ROI区域, 对该区域的图像QP进行限制, 从而实现图像中该区域的QP与其他图像区域的差异化。关于ROI编码功能的详细说明请参见ROI编码功能说明。 Atlas 200/300/500 推理产品, 不支持该接口。 Atlas 训练系列产品, 不支持该接口。 Atlas A2训练系列产品/Atlas 800I A2推理产品, 不支持该接口。
输入说明	chn: int, 编码通道号。 • Atlas 推理系列产品, 编码通道号的取值范围: [0, 256), JPEG功能 and VENC功能共用通道, 且通道总数最多256。 • Atlas 200I/500 A2推理产品, 编码通道号的取值范围: [0, 128), JPEG功能 and VENC功能共用通道, 且通道总数最多128。 roi_attr: dict, ROI区域参数字典。参见2.10.8.4.85 hi_venc_roi_attr。

返回值说明	ret: int, 错误码。 • 返回0表示成功。 • 返回非0表示失败, 参见2.10.9.4 VENC视频/JPEGE图像编码返回码。
约束说明	• 本接口用于设置H.264/H.265通道的ROI属性。 • 本接口在编码通道创建之后, 编码通道销毁之前设置。此接口在编码过程中被调用时, 等到下一个帧时生效。 • 建议用户在创建通道之后, 启动编码之前调用此接口, 减少在编码过程中调用的次数。 • 建议用户在调用此接口之前先调用acl.himpi.venc_get_roi_attr接口, 获取当前通道的ROI配置, 然后再进行设置。 • 设置该接口后, 如果当前帧判断编码为pskip帧, 以pskip帧效果优先。 • 设置该接口后, 可能会出现感兴趣区域QP值不全为设置值的现象, 这是因为在编码过程中, 有些编码块的残差会为0, 没有数据需要被量化, QP值不生效所致, 属于正常现象, 不影响编码效果。

ROI 编码功能说明

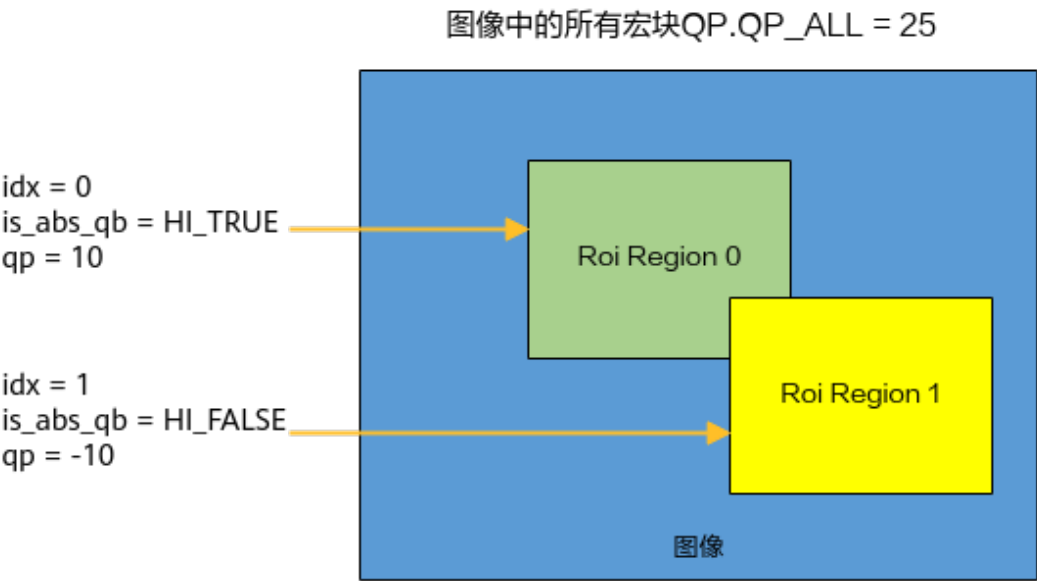
ROI (Region Of Interest) 编码：对感兴趣区域编码。

用户可以通过配置ROI区域, 对该区域的图像QP进行限制, 从而实现图像中该区域的QP与其它图像区域的差异化。系统现仅支持对H.264/H.265通道进行ROI设置。系统提供了8个感兴趣区域, 可供用户同时使用。

8个区域可以互相叠加, 且叠加时的优先级按照0 ~ 7的索引号依次提高, 这里, 叠加优先级是指发生叠加时, 图像区域的最终QP值的判定, 最终的区域QP值按照优先级最高的区域设定。ROI区域可配置绝对QP和相对QP两种模式。

- 绝对QP: ROI区域的QP为用户设定的QP值。
- 相对QP: ROI区域的QP为码率控制产生的QP与用户设定的QP偏移值的和。

以下图为例, 假如图像QP为25, 即图像中所有宏块QP值为25。Roi区域0设置为绝对QP模式, QP值为10, 索引为0; Roi区域1设置为相对QP模式, QP为-10, 索引为1。区域0的index小于区域1的index, 所以在发生互相重叠的图像区域按高优先级的区域 (区域1) QP设置。区域0除了发生重叠的部分的QP值等于10。区域1 的QP值为25-10=15。



2.10.6.43 acl.himpi.venc_set_chn_attr

c函数原型	hi_s32 hi_mpi_venc_set_chn_attr(hi_venc_chn chn, const hi_venc_chn_attr *chn_attr);
Python函数	ret = acl.himpi.venc_set_chn_attr(chn, chn_attr)
函数功能	设置编码通道的属性。 Atlas 200/300/500 推理产品，不支持该接口。 Atlas 训练系列产品，不支持该接口。 Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该接口。
输入说明	chn: int, 编码通道号。 - Atlas 推理系列产品，编码通道号的取值范围：[0, 256)，JPEG功能和VENC功能共用通道，且通道总数最多256。 - Atlas 200I/500 A2推理产品，编码通道号的取值范围：[0, 128)，JPEG功能和VENC功能共用通道，且通道总数最多128。 chn_attr: dict, 编码通道属性，具体请参见 2.10.8.4.3 hi_venc_chn_attr 。
返回值说明	ret: int, 错误码。 - 返回0表示成功。 - 返回非0表示失败，参见2.10.9.4 VENC视频/JPEG图像编码返回码。

约束说明	- 仅H.265/H.264编码通道支持。 - 建议用户在调用此接口之前先调用acl.himpi.venc_get_chn_attr接口，获取当前通道属性，然后再进行设置。 - 编码通道属性包括了编码器属性、帧结构GOP属性和码率控制器RC属性三部分。 编码通道属性分为动态属性和静态属性两种，其中，静态属性的属性值在通道创建时配置，在通道创建之后不能被修改；动态属性的属性值在通道创建时配置，有部分参数不支持修改，支持修改的参数在通道销毁之前可以被修改。 如果设置静态属性或者不支持修改的参数，则返回失败。静态属性、属性是否支持修改、属性取值范围具体请参见hi_venc_chn_attr。 - 设置编码通道属性，未修改“gop_mode”时，则通道属性立即生效；修改“gop_mode”时，则通道属性延时生效，最大延时生效时间如下所示（当前不支持HI_VENC_GOP_MODE_LOW_DELAY_B模式）： “HI_VENC_GOP_MODE_LOW_DELAY_B”模式，在修改为非“HI_VENC_GOP_MODE_LOW_DELAY_B”模式后，对应最大延时变化（以编码帧的个数为单位）为1 + b_frame_num。 - 非“HI_VENC_GOP_MODE_LOW_DELAY_B”模式，在修改为其他模式后，对应最大延时变化（以编码帧的个数为单位）为1。 - 修改“gop_mode”时，通道的P帧刷Islice、跳帧参考的属性（H.264/H.265）、pic_order_cnt_type（H.264）会恢复默认值，如需继续使用需要重新配置。 - 如果修改GOP、RC属性中的一种，RC模块会重新初始化，所有RC高级参数都恢复到默认值。如果客户程序中有对RC高级参数的调整，需要重新设置。RC高级参数请参考acl.himpi.venc_set_rc_param等。 - 如果同时修改“gop_mode”和“rc_mode”时，由于存在延迟生效（延时参见上表），为了防止调用acl.himpi.venc_get_rc_param获取的RC参数值不准确，建议等待延迟参数生效后再执行相应操作。

2.10.6.44 acl.himpi.venc_get_chn_attr

c函数原型	<code>hi_s32 hi_mpi_venc_get_chn_attr(hi_venc_chn chn, hi_venc_chn_attr *chn_attr);</code>
Python函数	<code>chn_attr, ret = acl.himpi.venc_get_chn_attr(chn)</code>
函数功能	获取编码通道当前的属性。 Atlas 200/300/500 推理产品 ，不支持该接口。 Atlas 训练系列产品 ，不支持该接口。 Atlas A2训练系列产品/Atlas 800I A2推理产品 ，不支持该接口。

输入说明	chn: int, 编码通道号。 • Atlas 推理系列产品, 编码通道号的取值范围: [0, 256), JPEG功能 and VENC功能共用通道, 且通道总数最多256。 • Atlas 200I/500 A2推理产品, 编码通道号的取值范围: [0, 128), JPEG功能和VENC功能共用通道, 且通道总数最多128。
返回值说明	ret: int, 错误码。 • 返回0表示成功。 • 返回非0表示失败, 参见2.10.9.4 VENC视频/JPEG图像编码返回码。 chn_attr: dict, 编码通道属性, 具体请参见2.10.8.4.3 hi_venc_chn_attr。
约束说明	• 本接口在编码通道创建之后、编码通道销毁之前调用。 • 仅H.265/H.264编码通道支持。

2.10.6.45 acl.himpi.venc_set_slice_split

c函数原型	hi_s32 hi_mpi_venc_set_slice_split(hi_venc_chn chn, const hi_venc_slice_split *slice_split)
Python函数	ret = acl.himpi.venc_set_slice_split(chn, slice_split)
函数功能	本接口用于设置通道的slice分割配置。 Atlas 200/300/500 推理产品, 不支持该接口。 Atlas 训练系列产品, 不支持该接口。 Atlas A2训练系列产品/Atlas 800I A2推理产品, 不支持该接口。
输入说明	chn: int, 编码通道号。 • Atlas 推理系列产品, 编码通道号的取值范围: [0, 256), JPEG功能和VENC功能共用通道, 且通道总数最多256。 • Atlas 200I/500 A2推理产品, 编码通道号的取值范围: [0, 128), JPEG功能和VENC功能共用通道, 且通道总数最多128。 slice_split: dict, H.264/H.265协议编码的码流分割参数, 具体请参见2.10.8.4.97 hi_venc_slice_split。
返回值说明	ret: int, 错误码。 • 返回0表示成功。 • 返回非0表示失败, 参见2.10.9.4 VENC视频/JPEG图像编码返回码。

约束说明	- 仅H.265/H.264编码通道支持。 - 本接口在编码通道创建之后，编码通道销毁之前设置。若在编码过程中调用本接口，则下一帧编码时配置生效。 - 建议用户在调用此接口之前先调用acl.himpi.venc_get_slice_split接口，获取当前通道的slice_split配置，然后再进行设置。

2.10.6.46 `acl.himpi.venc_get_slice_split`

c函数原型	<code>hi_s32 hi_mpi_venc_get_slice_split(hi_venc_chn chn, hi_venc_slice_split *slice_split)</code>
Python函数	<code>slice_split, ret = acl.himpi.venc_get_slice_split(chn)</code>
函数功能	本接口用于获取通道的slice分割配置。 Atlas 200/300/500 推理产品 ，不支持该接口。 Atlas 训练系列产品 ，不支持该接口。 Atlas A2训练系列产品/Atlas 800I A2推理产品 ，不支持该接口。
输入说明	chn : int，编码通道号。 Atlas 推理系列产品，编码通道号的取值范围：[0, 256)，JPEG功能 and VENC功能共用通道，且通道总数最多256。 Atlas 200I/500 A2推理产品，编码通道号的取值范围：[0, 128)，JPEG功能 and VENC功能共用通道，且通道总数最多128。
返回值说明	slice_split : dict，H.264/H.265协议编码的码流分割参数。 ret : int，错误码。 - 返回0表示成功。 - 返回非0表示失败，参见2.10.9.4 VENC视频/JPEG图像编码返回码。
约束说明	- 仅H.265/H.264编码通道支持。 - 本接口在编码通道创建之后、编码通道销毁之前调用。

2.10.7 PNGD 图像处理接口

2.10.7.1 功能及约束说明

Atlas 200/300/500 推理产品上，不支持本章中的PNGD功能。

Atlas 训练系列产品上，不支持本章中的PNGD功能。

功能说明

PNGD (PNG Decoder) 功能：实现PNG格式图片的硬件解码。

PNGD在解码图片时，支持按源图片格式解码。源图片格式解码是指解码前后图片的编码格式保持一致，例如解码前输入图片格式为RGB，解码后输出图片格式为RGB888。

如果不清楚PNG源图片格式、但想使用源格式解码时，有以下两种方式：

- 在调用PNGD解码接口时，直接将输出图片格式配置为“HI_PIXEL_FORMAT_UNKNOWN”，输出格式默认按源图片格式输出。此种方式，由于不知道输出图片格式，因此需要用户申请尽量大的内存，防止内存不够，无法存放输出图片。

输入图片格式	操作	实际输出图片格式
RGB/GRAY	将输出图片格式设置为“HI_PIXEL_FORMAT_UNKNOWN”	RGB888
RGBA/AGRAY	将输出图片格式设置为“HI_PIXEL_FORMAT_UNKNOWN”	RGBA8888

- 先调用[acl.himpi.png_get_image_info](#)接口根据传入的PNG源图片，获取按源图解码时的输出图片的宽、高、宽Stride、高Stride、解码输出内存大小、图片格式等信息后，再调用PNGD解码接口，使用通过[acl.himpi.png_get_image_info](#)接口获取的图片格式来设置输出图片格式。

图片分辨率约束

- 输入图片分辨率
最大分辨率4096 * 4096，最小分辨率32 * 32。
- 输出图片分辨率
PNGD只对图片解码，不会改变图片分辨率，因此输出与输入的图片分辨率保持一致。

内存申请/释放接口说明

输入内存的大小就是指实际的输入图片所占用的大小，输出内存的大小参见[表2-64](#)中的计算公式。

版本	内存申请/释放接口
Atlas 推理系列产品	PNG图片解码时，调用 acl.himpi.dvpp_malloc \ acl.himpi.dvpp_free 接口申请\释放Device上的输入、输出内存，这部分内存的生命周期由用户自行管理。

版本	内存申请/释放接口
Atlas A2训练系列产品/ Atlas 800I A2推理产品 Atlas 200I/500 A2推理 产品	PNG图片解码时，支持以下两种内存申请接口： - 调用acl.rt.malloc\acl.rt.free接口申请\释放Device上的输入、输出内存，这部分内存的生命周期由用户自行管理。 - 调用acl.himpi.dvpp_malloc\acl.himpi.dvpp_free接口申请\释放Device上的输入、输出内存，这部分内存的生命周期由用户自行管理。 调用acl.himpi.dvpp_malloc接口申请的内存为媒体数据处理的专用内存，但专用内存的地址空间有限，若关注内存规划或内存资源有限时，建议调用acl.rt.malloc接口申请内存。

图片格式、宽高对齐、内存约束

若不确定输出格式、将输出图片格式设置为“HI_PIXEL_FORMAT_UNKNOWN”时：

- 宽Stride：
如果输入图片是RGB或GRAY格式，则宽Stride为输出图片的宽先向上128对齐后再乘以3的值；如果输入图片是RGBA或AGRAY格式，则宽Stride为输出图片的宽先向上128对齐后再乘以4的值。
- 高Stride：
为输出图片的高向上16对齐后的值。

表 2-64 图片格式、宽高对齐、内存大小约束

输入图片格式	输出图片格式	输出图片宽、高对齐要求	输出图片宽Stride、高Stride、内存大小要求
RGB	RGB888	无对齐要求	宽Stride为宽向上1对齐、16对齐或128对齐后再乘以3的值。 高Stride的取值范围：[输出图片的高, 输出图片的高向上128对齐]。 内存大小（单位Byte）= 宽Stride * 高Stride
GRAY	RGB888		
RGBA	RGB888		宽Stride为宽向上1对齐、16对齐或128对齐后再乘以3的值。 高Stride的取值范围：[输出图片的高, 输出图片的高向上128对齐]。 内存大小（单位Byte）= 宽Stride * 高Stride

输入图片格式	输出图片格式	输出图片宽、高对齐要求	输出图片宽Stride、高Stride、内存大小要求
	RGBA8888 8bit		宽Stride为宽向上1对齐、16对齐或128对齐后再乘以4的值。 高Stride的取值范围：[输出图片的高, 输出图片的高向上128对齐]。 内存大小（单位Byte）= 宽Stride * 高Stride
AGRAY	RGB888		宽Stride为宽向上1对齐、16对齐或128对齐后再乘以3的值。 高Stride的取值范围：[输出图片的高, 输出图片的高向上128对齐]。 内存大小（单位Byte）= 宽Stride * 高Stride
	RGBA8888 8bit		宽Stride为宽向上1对齐、16对齐或128对齐后再乘以4的值。 高Stride的取值范围：[输出图片的高, 输出图片的高向上128对齐]。 内存大小（单位Byte）= 宽Stride * 高Stride

 说明

- 输出图片格式的定义请参见[hi_pixel_format](#)，宽Stride、高Stride等概念请参见[2.10.2 基本概念](#)。
- 如果用户将宽Stride、高Stride设置为0，传入对应的接口，PNGD内部在处理时，会默认根据输出图片格式将宽Stride向上128对齐、高Stride向上16对齐。

其它约束

PNGD只支持对完整PNG图片进行解码，不支持将一张PNG图片分割成多个数据包后，由PNGD解码。

2.10.7.2 性能指标说明

Atlas 200/300/500 推理产品上，当前版本不支持该功能。

Atlas 训练系列产品上，当前版本不支持该功能。

性能指标说明

本节以单个昇腾AI处理器为例，描述基本场景性能指标参考，其它分辨率可以等量估算。

1080p指分辨率为1920 * 1080的图片；4K指分辨率为3840 * 2160的图片。单个Device的基本场景性能指标参考如下（1路对应一个通道，一个通道对应一个线程）：

场景举例	总帧率
1080p * n路 (1 ≤ n ≤ 5)	n * 4fps
1080p * n路 (n ≥ 6)	24fps
4k * n路 (1 ≤ n ≤ 5)	n * 1fps
4k * n路 (n ≥ 6)	6fps

2.10.7.3 acl.himpi.pngd_create_chn

c函数原型	hi_s32 hi_mpi_pngd_create_chn(hi_pngd_chn chn, const hi_pngd_chn_attr *attr)
Python函数	ret = acl.himpi.pngd_create_chn(chn, attr)
函数功能	根据设置的通道属性创建解码通道。 Atlas 200/300/500 推理产品 ，当前版本不支持该接口。 Atlas 训练系列产品 ，当前版本不支持该接口。
输入说明	chn : int，解码通道号。 该参数的取值范围为[0, 128)，通道总数最多128。 说明 对于Atlas 推理系列产品 ，昇腾虚拟化实例（Ascend Virtual Instance）场景下，使用不同的昇腾虚拟化实例（Ascend Virtual Instance）模板，对通道数的限制不同： - 使用vir04_4c_dvpp模板时，通道总数固定为128。 - 使用vir04_3c_ndvpp模板时，表示不使用DVPP功能，因此通道总数为0。 - 使用其它模板时，通道总数 = 切分规格 * 128，如果通道总数不为整数，则向下取整。 对于Atlas A2训练系列产品/Atlas 800I A2推理产品 ，昇腾虚拟化实例（Ascend Virtual Instance）场景下，使用不同的昇腾虚拟化实例（Ascend Virtual Instance）模板，对通道数的限制不同： - 使用vir12_3c_32g_m、vir10_3c_16g_m、vir10_3c_32g_m模板时，通道总数固定为128。 - 使用vir12_3c_32g_nm、vir10_3c_16g_nm、vir10_3c_32g_nm模板时，表示不使用DVPP功能，因此通道总数为0。 - 使用其它模板时，通道总数 = 切分规格 * 128，如果通道总数不为整数，则向下取整。 昇腾虚拟化实例（Ascend Virtual Instance）的详细描述，以及各场景下的算力资源规格请参见《昇腾虚拟化实例（AVI）用户指南》。 attr : dict，解码通道属性字典，具体请参见 2.10.8.6.1 hi_pngd_chn_attr 。

返回值说明	ret : int, 错误码。 • 返回0表示成功。 • 返回非0表示失败, 参见2.10.9.5 PNGD图像解码返回码。
约束说明	• 单个Device上的通道号不能超出最大的通道号范围。 • 如果参数“attr”为空, 会返回错误码“HI_ERR_PNGD_NULL_PTR”。 • 在创建PNG解码通道之前必须保证通道未创建(或者已经销毁), 否则会直接返回错误。 • 系统内存不足时会返回“HI_ERR_PNGD_NO_MEM”的错误码, 可考虑扩展操作系统内存。 • 当通道属性“attr”中的值超过解码能力集时, 会返回“HI_ERR_PNGD_ILLEGAL_PARAM”的错误码。
参考场景	接口调用流程, 参见 2.10.7.1 功能及约束说明 。

2.10.7.4 acl.himpi.pngd_destroy_chn

c函数原型	hi_s32 hi_mpi_pngd_destroy_chn(hi_pngd_chn chn)
Python函数	ret = acl.himpi.pngd_destroy_chn(chn)
函数功能	用户退出流程时, 需要关闭之前创建的解码通道, 以释放相关资源。 Atlas 200/300/500 推理产品 , 当前版本不支持该接口。 Atlas 训练系列产品 , 当前版本不支持该接口。
输入说明	chn : int, 解码通道号。 该参数的取值范围为[0, 128), 通道总数最多128。
返回值说明	ret : int, 错误码。 • 返回0表示成功。 • 返回非0表示失败, 参见2.10.9.5 PNGD图像解码返回码。
约束说明	销毁前必须保证通道已创建, 否则会返回通道未创建的错误。
参考资源	接口调用流程, 参见 2.10.7.1 功能及约束说明 。

2.10.7.5 acl.himpi.pngd_send_stream

c函数原型	hi_s32 hi_mpi_pngd_send_stream(hi_pngd_chn chn, const hi_img_stream *stream, hi_pic_info *png_pic_info, hi_s32 milli_sec)
Python函数	ret = acl.himpi.pngd_send_stream(chn, stream, png_pic_info, milli_sec)

函数功能	解码前，向解码通道发送码流数据及存放解码结果的Buffer。 Atlas 200/300/500 推理产品 ，当前版本不支持该接口。 Atlas 训练系列产品 ，当前版本不支持该接口。
输入说明	chn : int，解码通道号。 该参数的取值范围为[0, 128)，通道总数最多128。 stream : dict，输入码流信息，具体请参见 2.10.8.6.2 hi_img_stream 。该字典内的“addr”参数配置的地址为Device上的内存地址。 png_pic_info : dict，输出图片信息字典， 2.10.8.6.3 hi_pic_info 。 milli_sec : int，超时时间，单位是毫秒。 • -1：阻塞方式 • 0：非阻塞方式 • >0：超时方式，配置具体的超时时间。超时时间受操作系统影响，一般偏差在操作系统的一个时间片内，例如，操作系统的-一个时间片为4ms，用户设置的milli_sec参数值为1，则实际的超时时间在1ms到5ms范围内。在CPU负载高场景下，超时时间仍可能存在波动。
返回值说明	ret : int，错误码。 • 返回0表示成功。 • 返回非0表示失败，参见2.10.9.5 PNGD图像解码返回码。
约束说明	• 发送数据前必须保证通道已经被创建，否则直接返回通道未创建的错误码“HI_ERR_PNGD_UNEXIST”。如果在发送码流过程中销毁通道，就会立刻返回错误码“HI_ERR_PNGD_UNEXIST”。 • 发送码流时需要送入整张PNG图片。否则，解码会出现错误。 • 以非阻塞方式发送码流，如果码流缓冲区已满，会立刻返回错误码“HI_ERR_PNGD_BUF_FULL”。 • 以超时方式发送码流，到达设定的超时时间还不能成功发送码流会返回错误码“HI_ERR_PNGD_BUF_FULL”。 • 图像解码时，对输入、输出图片的要求请参见2.10.7.1 功能及约束说明。 • 解码时，输入输出内存必须使用acl.himpi.dvpp_malloc接口和acl.himpi.dvpp_free接口进行申请和释放，输入输出内存均需要在调用2.10.7.6 acl.himpi.pngd_get_image_data接口获取结果之后才能进行释放。
参考资源	接口调用流程，参见 2.10.7.1 功能及约束说明 。

2.10.7.6 acl.himpi.pngd_get_image_data

c函数原型	hi_s32 hi_mpi_pngd_get_image_data(hi_pngd_chn chn, hi_pic_info *png_pic_info, hi_img_stream *stream, hi_s32 milli_sec)
Python函数	png_pic_info, stream, ret = acl.himpi.pngd_get_image_data(chn, milli_sec)
函数功能	解码后，获取解码通道的解码图像及输入Stream。 Atlas 200/300/500 推理产品 ，当前版本不支持该接口。 Atlas 训练系列产品 ，当前版本不支持该接口。
输入说明	chn : int，解码通道号。 该参数的取值范围为[0, 128)，通道总数最多128。 milli_sec : int，超时时间，单位是毫秒。 • -1：阻塞方式 • 0：非阻塞方式 • >0：超时方式，配置具体的超时时间。超时时间受操作系统影响，一般偏差在操作系统的-一个时间片内，例如，操作系统的-一个时间片为4ms，用户设置的milli_sec参数值为1，则实际的-超时时间在1ms到5ms范围内。在CPU负载高场景下，超时时间-仍可能存在波动。
返回值说明	stream : dict，获取已完成解码的输入码流信息字典，具体请参见 2.10.8.6.2 hi_img_stream 。 png_pic_info : dict，获取已经完成解码的图像信息字典，具体请参见 2.10.8.6.3 hi_pic_info 。解码后的数据存放在Device内存中。 ret : int，错误码。 • 返回0表示获取数据成功。 • 返回非0表示失败，参见2.10.9.5 PNGD图像解码返回码。 “HI_ERR_PNGD_BUF_EMPTY”表示无数据，其他错误码表示解码异常。
约束说明	• 此接口通过改变“milli_sec”值支持阻塞方式、非阻塞方式、超时方式获取解码图像。 • 获取解码图像时必须保证通道已经被创建，否则直接返回通道未创建的错误码“HI_ERR_PNGD_UNEXIST”。如果在获取图像的过程中销毁通道，就会立刻返回错误码“HI_ERR_PNGD_UNEXIST”。 • 以非阻塞方式获取解码图像，如果缓冲区内无图像，会立刻返回错误码“HI_ERR_PNGD_BUF_EMPTY”。 • 以超时方式获取解码图像，到达设定的超时时间还不能获取到图像则会返回错误码“HI_ERR_PNGD_BUF_EMPTY”。 • 向PNGD获取解码结果，输入Buffer和输出Buffer一起获取。 • PNGD图片解码时，关于输入、输出图片的要求请参见2.10.7.1 功能及约束说明。
参考资源	接口调用流程，参见 2.10.7.1 功能及约束说明 。

2.10.7.7 acl.himpi.png_get_image_info

c函数原型	hi_s32 hi_mpi_png_get_image_info(const hi_img_stream *png_stream, hi_img_info *img_info)
Python函数	img_info, ret = acl.himpi.png_get_image_info(png_stream)
函数功能	根据输入码流，获取PNG源图片的宽、高、源格式及源格式解码时输出图片的宽Stride、高Stride、以及所需的内存大小等信息。 Atlas 200/300/500 推理产品 ，当前版本不支持该接口。 Atlas 训练系列产品 ，当前版本不支持该接口。
输入说明	png_stream : dict，输入码流信息，具体请参见 2.10.8.6.2 hi_img_stream 。 Atlas 推理系列产品 标准形态下，在Host上调用本接口时，该字典内的addr参数配置的地址必须是Host上的内存地址。
返回值说明	img_info : dict，图片信息字典，具体请参见 2.10.8.2.14 hi_img_info 。 ret : int，错误码。 • 返回0表示成功。 • 返回非0表示失败，参见2.10.9.5 PNGD图像解码返回码。
约束说明	提供给用户来计算解码后输出图片占用的内存大小。当前仅支持解析PNG图片。通过本接口获取的“width_stride”向上128对齐、“height_stride”向上16对齐。

2.10.8 数据类型

2.10.8.1 总体说明

- 本章节描述DVPP各个模块所用到的字典、全局变量等，预留的字段需要手动设置为0，避免后续版本的兼容性问题。
- 命名规则：
- 各模块均需要的模块放在公共部分，各模块特性部分尽量放在各模块内，并且以模块名开头。
 - 例如“HI_COMPRESS_MODE_BUTT”，表示保留值。

2.10.8.2 公共

此部分作为公共部分，描述了本模块所支持以及未来可能支持的图片格式，以及支持的Payload（数据内容）类型等等。

2.10.8.2.1 基本数据类型说明

```
HI_FALSE = 0
HI_TRUE = 1
```

2.10.8.2.2 hi_pixel_format

说明

DVPP所支持的各种图片格式。

定义

```
HI_PIXEL_FORMAT_YUV_400           = 0// YUV400 8bit
HI_PIXEL_FORMAT_YUV_SEMIPLANAR_420 = 1// YUV420SP NV12 8bit
HI_PIXEL_FORMAT_YVU_SEMIPLANAR_420 = 2// YUV420SP NV21 8bit
HI_PIXEL_FORMAT_YUV_SEMIPLANAR_422 = 3// YUV422SP 8bit
HI_PIXEL_FORMAT_YVU_SEMIPLANAR_422 = 4// YVU422SP 8bit
HI_PIXEL_FORMAT_YUV_SEMIPLANAR_444 = 5// YUV444SP 8bit
HI_PIXEL_FORMAT_YVU_SEMIPLANAR_444 = 6// YVU444SP 8bit
HI_PIXEL_FORMAT_YUYV_PACKED_422    = 7// YUV422 Package YUYV 8bit
HI_PIXEL_FORMAT_UYVY_PACKED_422    = 8// YUV422 Package UYVY 8bit
HI_PIXEL_FORMAT_YVYU_PACKED_422    = 9// YUV422 Package YVYU 8bit
HI_PIXEL_FORMAT_VYUY_PACKED_422    = 10// YUV422 Package VYUY 8bit
HI_PIXEL_FORMAT_YUV_PACKED_444     = 11// YUV444 Package 8bit
HI_PIXEL_FORMAT_RGB_888            = 12// RGB888
HI_PIXEL_FORMAT_BGR_888            = 13// BGR888
HI_PIXEL_FORMAT_ARGB_8888         = 14// ARGB8888
HI_PIXEL_FORMAT_ABGR_8888         = 15// ABGR8888
HI_PIXEL_FORMAT_RGBA_8888         = 16// RGBA8888
HI_PIXEL_FORMAT_BGRA_8888         = 17// BGRA8888
HI_PIXEL_FORMAT_YUV_SEMI_PLANNER_420_10BIT = 18 // YUV420SP 10bit
HI_PIXEL_FORMAT_YVU_SEMI_PLANNER_420_10BIT = 19 // YVU420sp 10bit
HI_PIXEL_FORMAT_YVU_PLANAR_420    = 20// YVU420P 8bit
HI_PIXEL_FORMAT_YVU_PLANAR_422    = 21// YVU422P 8bit
HI_PIXEL_FORMAT_YVU_PLANAR_444    = 22// YVU444P 8bit
HI_PIXEL_FORMAT_RGB_444           = 23// RGB444 R:4bit G:4bit B:4bit, 当前不支持该格式
HI_PIXEL_FORMAT_BGR_444           = 24// BGR444 R:4bit G:4bit B:4bit, 当前不支持该格式
HI_PIXEL_FORMAT_ARGB_4444         = 25// ARGB4444 A:4bit R:4bit G:4bit B:4bit, 当前不支持该格式
HI_PIXEL_FORMAT_ABGR_4444         = 26// ABGR4444 A:4bit B:4bit G:4bit R:4bit, 当前不支持该格式
HI_PIXEL_FORMAT_RGBA_4444         = 27// RGBA4444 R:4bit G:4bit B:4bit A:4bit, 当前不支持该格式
HI_PIXEL_FORMAT_BGRA_4444         = 28// BGRA4444 B:4bit G:4bit R:4bit A:4bit, 当前不支持该格式
HI_PIXEL_FORMAT_RGB_555           = 29// RGB555 R:5bit G:5bit B:5bit, 当前不支持该格式
HI_PIXEL_FORMAT_BGR_555           = 30// BGR555 B:5bit G:5bit R:5bit, 当前不支持该格式
HI_PIXEL_FORMAT_RGB_565           = 31// RGB565 R:5bit G:6bit B:5bit, 当前不支持该格式
HI_PIXEL_FORMAT_BGR_565           = 32// BGR565 B:5bit G:6bit R:5bit, 当前不支持该格式
HI_PIXEL_FORMAT_ARGB_1555         = 33// ARGB1555 A:1bit R:5bit G:6bit B:5bit, 当前不支持该格式
HI_PIXEL_FORMAT_ABGR_1555         = 34// ABGR1555 A:1bit B:5bit G:6bit R:5bit, 当前不支持该格式
HI_PIXEL_FORMAT_RGBA_1555         = 35// RGBA1555 A:1bit B:5bit G:6bit R:5bit, 当前不支持该格式
HI_PIXEL_FORMAT_BGRA_1555         = 36// BGRA1555 A:1bit B:5bit G:6bit R:5bit, 当前不支持该格式
HI_PIXEL_FORMAT_ARGB_8565         = 37// ARGB8565 A:8bit R:5bit G:6bit B:5bit, 当前不支持该格式
HI_PIXEL_FORMAT_ABGR_8565         = 38// ABGR8565 A:8bit B:5bit G:6bit R:5bit, 当前不支持该格式
HI_PIXEL_FORMAT_RGBA_8565         = 39// RGBA8565 A:8bit R:5bit G:6bit B:5bit, 当前不支持该格式
HI_PIXEL_FORMAT_BGRA_8565         = 40// BGRA8565 A:8bit B:5bit G:6bit R:5bit, 当前不支持该格式
HI_PIXEL_FORMAT_ARGB_CLUT2        = 41// ARGB Color Lookup Table 2bit
HI_PIXEL_FORMAT_ARGB_CLUT4        = 42// ARGB Color Lookup Table 4bit

HI_PIXEL_FORMAT_RGB_BAYER_8BPP     = 50
HI_PIXEL_FORMAT_RGB_BAYER_10BPP    = 51
HI_PIXEL_FORMAT_RGB_BAYER_12BPP    = 52
HI_PIXEL_FORMAT_RGB_BAYER_14BPP    = 53
HI_PIXEL_FORMAT_RGB_BAYER_16BPP    = 54// RGB Bayer 16bit, Bayer图像, 当前不支持该格式
HI_PIXEL_FORMAT_YUV_PLANAR_420     = 55// YUV420P 8bit
HI_PIXEL_FORMAT_YUV_PLANAR_422     = 56// YUV422P 8bit
HI_PIXEL_FORMAT_YUV_PLANAR_444     = 57// YUV444P 8bit
HI_PIXEL_FORMAT_YVU_PACKED_444     = 58// YVU444 Package 8bit
HI_PIXEL_FORMAT_XYUV_PACKED_444    = 59// AYUV444 Package 8bit
```

HI_PIXEL_FORMAT_XYVU_PACKED_444	= 60// AYVU444 Package 8bit
HI_PIXEL_FORMAT_YUV_SEMIPLANAR_411	= 61// YUV411SP 8bit
HI_PIXEL_FORMAT_YVU_SEMIPLANAR_411	= 62// YVU411SP 8bit
HI_PIXEL_FORMAT_YUV_PLANAR_411	= 63// YUV411P 8bit
HI_PIXEL_FORMAT_YVU_PLANAR_411	= 64// YVU411P 8bit
HI_PIXEL_FORMAT_YUV_PLANAR_440	= 65// YUV440P 8bit
HI_PIXEL_FORMAT_YVU_PLANAR_440	= 66// YVU440P 8bit
HI_PIXEL_FORMAT_RGB_888_PLANAR	= 69// RGB888 Planar
HI_PIXEL_FORMAT_BGR_888_PLANAR	= 70// BGR888 Planar
HI_PIXEL_FORMAT_HSV_888_PACKAGE	= 71// HSV Package, HSV图像package格式, 当前不支持该格式
HI_PIXEL_FORMAT_HSV_888_PLANAR	= 72// HSV Planar, HSV图像Planar格式, 当前不支持该格式
HI_PIXEL_FORMAT_LAB_888_PACKAGE	= 73// LAB Package, LAB图像package格式, 当前不支持该格式
HI_PIXEL_FORMAT_LAB_888_PLANAR	= 74// LAB Planar, LAB图像Planar格式, 当前不支持该格式
HI_PIXEL_FORMAT_S8C1	= 75// Signed 8bit for 1pixel 1Channel, 每个像素用1个8bit有符号数表示的单通道图像, 当前不支持该格式
HI_PIXEL_FORMAT_S8C2_PACKAGE	= 76// Signed 8bit for 1pixel 2Channel Package, 每个像素用2个8bit有符号数表示的双通道图像Package格式, 当前不支持该格式
HI_PIXEL_FORMAT_S8C2_PLANAR	= 77// Signed 8bit for 1pixel 2Channel Planar, 每个像素用2个8bit有符号数表示的双通道图像Planar格式, 当前不支持该格式
HI_PIXEL_FORMAT_S16C1	= 78// Signed 16bit 1pixel 1Channel, 每个像素用1个16bit有符号数表示的单通道图像, 当前不支持该格式
HI_PIXEL_FORMAT_U8C1	= 79// Unsigned 8bit 1pixel 1Channel, 每个像素用1个8bit无符号数表示的单通道图像, 当前不支持该格式
HI_PIXEL_FORMAT_U16C1	= 80// Unsigned 16bit 1pixel 1Channel, 每个像素用1个16bit无符号数表示的单通道图像, 当前不支持该格式
HI_PIXEL_FORMAT_S32C1	= 81// Signed 32bit 1pixel 1Channel, 每个像素用1个32bit有符号数表示的单通道图像, 当前不支持该格式
HI_PIXEL_FORMAT_U32C1	= 82// Unsigned 32bit 1pixel 1Channel, 每个像素用1个32bit无符号数表示的单通道图像, 当前不支持该格式
HI_PIXEL_FORMAT_U64C1	= 83// Unsigned 64bit 1pixel 1Channel, 每个像素用1个64bit无符号数表示的单通道图像, 当前不支持该格式
HI_PIXEL_FORMAT_S64C1	= 84// Signed 64bit 1pixel 1Channel, 每个像素用1个64bit有符号数表示的单通道图像, 当前不支持该格式
HI_PIXEL_FORMAT_RGB_888_INT8	= 110// RGB888 Package 每个像素的单分量占用1个8bit有符号数
HI_PIXEL_FORMAT_BGR_888_INT8	= 111// BGR888 Package 每个像素的单分量占用1个8bit有符号数
HI_PIXEL_FORMAT_RGB_888_INT16	= 112// RGB888 Package 每个像素的单分量占用1个16bit有符号数
HI_PIXEL_FORMAT_BGR_888_INT16	= 113// BGR888 Package 每个像素的单分量占用1个16bit有符号数
HI_PIXEL_FORMAT_RGB_888_INT32	= 114// RGB888 Package 每个像素的单分量占用1个32bit有符号数
HI_PIXEL_FORMAT_BGR_888_INT32	= 115// BGR888 Package 每个像素的单分量占用1个32bit有符号数
HI_PIXEL_FORMAT_RGB_888_UINT16	= 116// RGB888 Package 每个像素的单分量占用1个16bit无符号数
HI_PIXEL_FORMAT_BGR_888_UINT16	= 117// BGR888 Package 每个像素的单分量占用1个16bit无符号数
HI_PIXEL_FORMAT_RGB_888_UINT32	= 118// RGB888 Package 每个像素的单分量占用1个32bit无符号数
HI_PIXEL_FORMAT_BGR_888_UINT32	= 119// BGR888 Package 每个像素的单分量占用1个32bit无符号数
HI_PIXEL_FORMAT_RGB_888_PLANAR_INT8	= 120// RGB888 Planar 每个像素的单分量占用1个8bit有符号数
HI_PIXEL_FORMAT_BGR_888_PLANAR_INT8	= 121// BGR888 Planar 每个像素的单分量占用1个8bit有符号数
HI_PIXEL_FORMAT_RGB_888_PLANAR_INT16	= 122// RGB888 Planar 每个像素的单分量占用1个16bit有符号数
HI_PIXEL_FORMAT_BGR_888_PLANAR_INT16	= 123// BGR888 Planar 每个像素的单分量占用1个16bit有符号数
HI_PIXEL_FORMAT_RGB_888_PLANAR_INT32	= 124// RGB888 Planar 每个像素的单分量占用1个32bit有符号数
HI_PIXEL_FORMAT_BGR_888_PLANAR_INT32	= 125// BGR888 Planar 每个像素的单分量占用1个32bit有符号数
HI_PIXEL_FORMAT_RGB_888_PLANAR_UINT16	= 126// RGB888 Planar 每个像素的单分量占用1个16bit无

符号数	
HI_PIXEL_FORMAT_BGR_888_PLANAR_UINT16	= 127// BGR888 Planar 每个像素的单分量占用1个16bit无符号数
HI_PIXEL_FORMAT_RGB_888_PLANAR_UINT32	= 128// RGB888 Planar 每个像素的单分量占用1个32bit无符号数
HI_PIXEL_FORMAT_BGR_888_PLANAR_UINT32	= 129// BGR888 Planar 每个像素的单分量占用1个32bit无符号数
HI_PIXEL_FORMAT_YUV400_UINT16	= 130// YUV400 Package 每个像素的单分量占用1个16bit无符号数
HI_PIXEL_FORMAT_YUV400_UINT32	= 131// YUV400 Package 每个像素的单分量占用1个32bit无符号数
HI_PIXEL_FORMAT_YUV400_UINT64	= 132// YUV400 Package 每个像素的单分量占用1个64bit无符号数
HI_PIXEL_FORMAT_YUV400_INT8	= 133// YUV400 Package 每个像素的单分量占用1个8bit有符号数
HI_PIXEL_FORMAT_YUV400_INT16	= 134// YUV400 Package 每个像素的单分量占用1个16bit有符号数
HI_PIXEL_FORMAT_YUV400_INT32	= 135// YUV400 Package 每个像素的单分量占用1个32bit有符号数
HI_PIXEL_FORMAT_YUV400_INT64	= 136// YUV400 Package 每个像素的单分量占用1个64bit有符号数
HI_PIXEL_FORMAT_YUV400_FP16	= 137// YUV400 Package 每个像素用1个float16数据表示
HI_PIXEL_FORMAT_YUV400_FP32	= 138// YUV400 Package 每个像素用1个float32数据表示
HI_PIXEL_FORMAT_YUV400_FP64	= 139// YUV400 Package 每个像素用1个float64数据表示
HI_PIXEL_FORMAT_YUV400_BF16	= 140// YUV400 Package 每个像素用1个BFloat16数据表示，当前不支持该格式
HI_PIXEL_FORMAT_YUV_SEMIPLANAR_440	= 1000// YUV440SP 8bit
HI_PIXEL_FORMAT_YVU_SEMIPLANAR_440	= 1001// YVU440SP 8bit
HI_PIXEL_FORMAT_FLOAT32	= 1002// Float 32bit for 1pixel，每个像素用1个float32数据表示，当前不支持该格式
HI_PIXEL_FORMAT_BUTT	= 1003
HI_PIXEL_FORMAT_RGB_888_PLANAR_FP16	= 1004// RGB888 Planar 每个像素用1个float16数据表示
HI_PIXEL_FORMAT_BGR_888_PLANAR_FP16	= 1005// BGR888 Planar 每个像素用1个float16数据表示
HI_PIXEL_FORMAT_RGB_888_PLANAR_FP32	= 1006// RGB888 Planar 每个像素用1个float32数据表示
HI_PIXEL_FORMAT_BGR_888_PLANAR_FP32	= 1007// BGR888 Planar 每个像素用1个float32数据表示
HI_PIXEL_FORMAT_RGB_888_PLANAR_BF16	= 1008// RGB888 Planar 每个像素用1个BFloat16数据表示，当前不支持该格式
HI_PIXEL_FORMAT_BGR_888_PLANAR_BF16	= 1009// BGR888 Planar 每个像素用1个BFloat16数据表示，当前不支持该格式
HI_PIXEL_FORMAT_RGB_888_FP16	= 1010// RGB888 Package，每个像素用1个float16数据表示
HI_PIXEL_FORMAT_BGR_888_FP16	= 1011// BGR888 Package，每个像素用1个float16数据表示
HI_PIXEL_FORMAT_RGB_888_FP32	= 1012// RGB888 Package，每个像素用1个float32数据表示
HI_PIXEL_FORMAT_BGR_888_FP32	= 1013// BGR888 Package，每个像素用1个float32数据表示
HI_PIXEL_FORMAT_RGB_888_BF16	= 1014// RGB888 Package 每个像素用1个BFloat16数据表示，当前不支持该格式
HI_PIXEL_FORMAT_BGR_888_BF16	= 1015// BGR888 Package 每个像素用1个BFloat16数据表示，当前不支持该格式
HI_PIXEL_FORMAT_UNKNOWN	= 10000

参考信息

下文以HI_PIXEL_FORMAT_S8C1、HI_PIXEL_FORMAT_S8C2_PACKAGE、HI_PIXEL_FORMAT_S8C2_PLANAR为例说明排布格式，供参考。

图 2-21 HI_PIXEL_FORMAT_S8C1

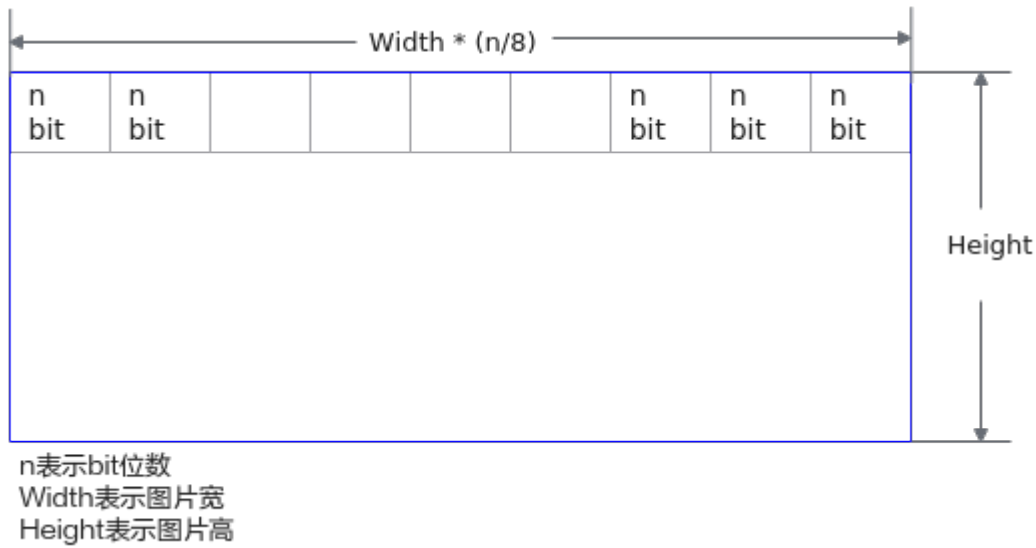


图 2-22 HI_PIXEL_FORMAT_S8C2_PACKAGE

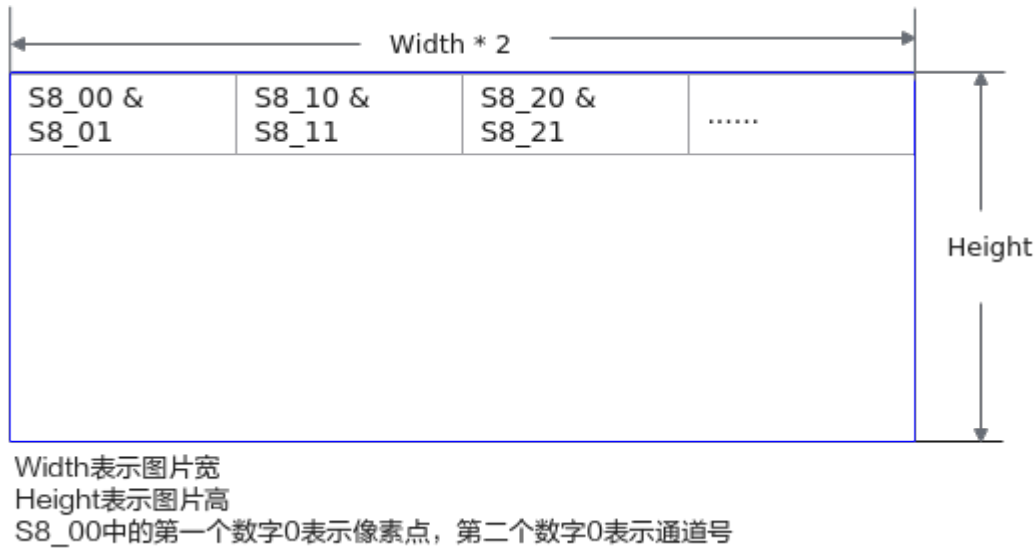
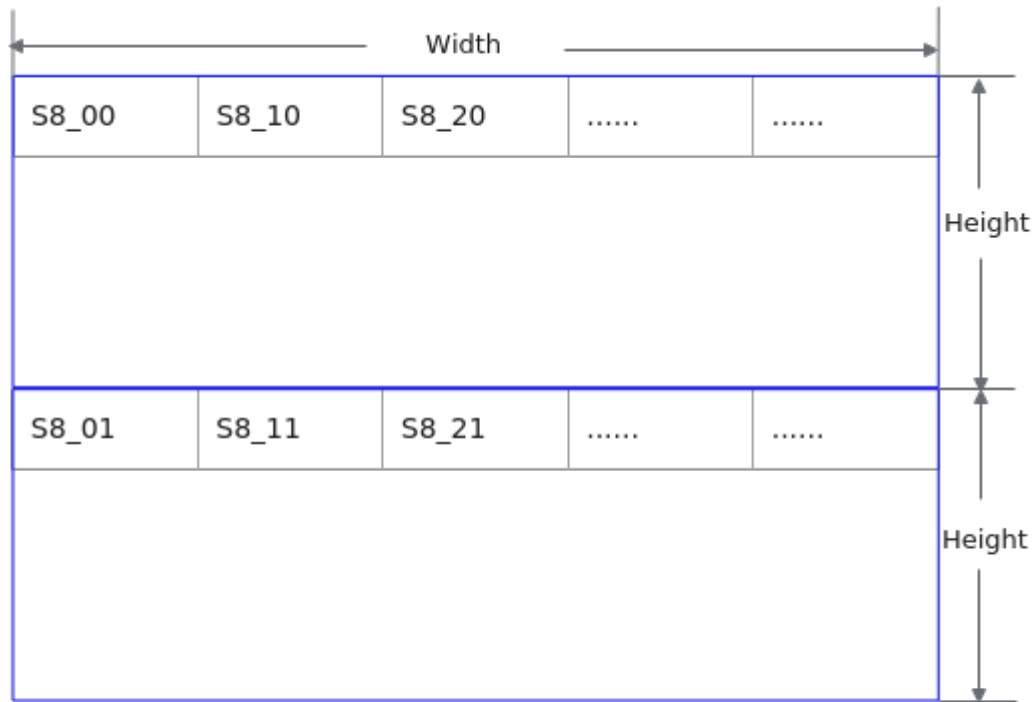


图 2-23 HI_PIXEL_FORMAT_S8C2_PLANAR



Width表示图片宽
Height表示图片高
S8_00中的第一个数字0表示像素点，第二个数字0表示通道号

2.10.8.2.3 hi_payload_type

说明

定义视频类型。

目前DVPP模块仅支持HI_PT_H264、HI_PT_H265、HI_PT_JPEG、HI_PT_VPC，HI_PT_PNG，其中各功能支持的视频类型如下：

- VPC: HI_PT_VPC
- JPEGD: HI_PT_JPEG
- JPEGE: HI_PT_JPEG
- VDEC: HI_PT_H264、HI_PT_H265
- VENC: HI_PT_H264、HI_PT_H265
- PNGD: HI_PT_PNG

定义

HI_PT_PCMU	= 0
HI_PT_1016	= 1
HI_PT_G721	= 2
HI_PT_GSM	= 3
HI_PT_G723	= 4
HI_PT_DVI4_8K	= 5
HI_PT_DVI4_16K	= 6
HI_PT_LPC	= 7
HI_PT_PCMA	= 8

```
HI_PT_G722          = 9
HI_PT_S16BE_STEREO = 10
HI_PT_S16BE_MONO   = 11
HI_PT_QCELP        = 12
HI_PT_CN           = 13
HI_PT_MPEGAUDIO    = 14
HI_PT_G728         = 15
HI_PT_DVI4_3       = 16
HI_PT_DVI4_4       = 17
HI_PT_G729         = 18
HI_PT_G711A        = 19
HI_PT_G711U        = 20
HI_PT_G726         = 21
HI_PT_G729A        = 22
HI_PT_LPCM         = 23
HI_PT_CelB         = 25
HI_PT_JPEG         = 26
HI_PT_CUSM         = 27
HI_PT_NV           = 28
HI_PT_PICW         = 29
HI_PT_CPV          = 30
HI_PT_H261         = 31
HI_PT_MPEGVIDEO    = 32
HI_PT_MPEG2TS      = 33
HI_PT_H263         = 34
HI_PT_SPEG         = 35
HI_PT_MPEG2VIDEO   = 36
HI_PT_AAC          = 37
HI_PT_WMA9STD      = 38
HI_PT_HEAAC        = 39
HI_PT_PCM_VOICE    = 40
HI_PT_PCM_AUDIO    = 41
HI_PT_MP3          = 43
HI_PT_ADPCMA       = 49
HI_PT_AEC          = 50
HI_PT_X_LD         = 95
HI_PT_H264         = 96
HI_PT_D_GSM_HR     = 200
HI_PT_D_GSM_EFR    = 201
HI_PT_D_L8         = 202
HI_PT_D_RED        = 203
HI_PT_D_VDVI       = 204
HI_PT_D_BT656      = 220
HI_PT_D_H263_1998  = 221
HI_PT_D_MP1S       = 222
HI_PT_D_MP2P       = 223
HI_PT_D_BMPEG      = 224
HI_PT_MP4VIDEO     = 230
HI_PT_MP4AUDIO     = 237
HI_PT_VC1          = 238
HI_PT_JVC_ASF      = 255
HI_PT_D_AVI        = 256
HI_PT_DIVX3        = 257
HI_PT_AVS          = 258
HI_PT_REAL8        = 259
HI_PT_REAL9        = 260
HI_PT_VP6          = 261
HI_PT_VP6F         = 262
HI_PT_VP6A         = 263
HI_PT_SOIRENSEN    = 264
HI_PT_H265         = 265
HI_PT_VP8          = 266
HI_PT_MVC          = 267
HI_PT_PNG          = 268
# add by hisilicon
HI_PT_AMR          = 1001
HI_PT_MJPEG        = 1002
HI_PT_AMRWB        = 1003
HI_PT_PRORES       = 1006
```

```
HI_PT_OPUS      = 1007
HI_PT_VPC       = 2000
HI_PT_BUTT      = 2001
```

2.10.8.2.4 hi_video_field

说明

定义视频图像帧场类型。

定义

```
HI_VIDEO_FIELD_TOP      = 0x1, # even field
HI_VIDEO_FIELD_BOTTOM   = 0x2, # odd field
HI_VIDEO_FIELD_INTERLACED = 0x3, # two interlaced fields
HI_VIDEO_FIELD_FRAME     = 0x4, # frame
HI_VIDEO_FIELD_BUTT     = 0x5
```

成员

成员名称	描述
HI_VIDEO_FIELD_TOP	顶场类型。
HI_VIDEO_FIELD_BOTTOM	底场类型。
HI_VIDEO_FIELD_INTERLACED	两场间插类型。
HI_VIDEO_FIELD_FRAME	帧类型。
HI_VIDEO_FIELD_BUTT	保留值。

2.10.8.2.5 hi_dynamic_range

说明

定义动态范围。

定义

```
HI_DYNAMIC_RANGE_SDR8 = 0
HI_DYNAMIC_RANGE_SDR10 = 1
HI_DYNAMIC_RANGE_HDR10 = 2
HI_DYNAMIC_RANGE_HLG = 3
HI_DYNAMIC_RANGE_SLF = 4
HI_DYNAMIC_RANGE_XDR = 5
HI_DYNAMIC_RANGE_BUTT = 6
```


成员

成员名称	描述
HI_DYNAMIC_RANGE_SDR8	8bit 数据的标准动态范围。
HI_DYNAMIC_RANGE_SDR10	10bit 数据的标准动态范围。
HI_DYNAMIC_RANGE_HDR10	10bit 数据的高动态范围。
HI_DYNAMIC_RANGE_HLG	带有混合对数的Gamma。
HI_DYNAMIC_RANGE_SLF	暂时无效。
HI_DYNAMIC_RANGE_XDR	10bit 数据，算法处理的一个中间类型数据，用户无需关心。
HI_DYNAMIC_RANGE_BUTT	保留值。

注意事项

各个动态范围对应的曲线如下：

Dynamic Range	Transfer Characteristic
SDR8/ SDR10	$V = \alpha * L_c^{0.45} - (\alpha - 1)$ for $1 \geq L_c \geq \beta$ $V = 4.500 * L_c$ for $\beta > L_c \geq 0$
HDR10	对于 L_c 的所有值， $V = ((c_1 + c_2 * L_c^n) \div (1 + c_3 * L_c^n))$ $c_1 = c_3 - c_2 + 1 = 3424 \div 4096 = 0.8359375$ $c_2 = 32 * 2413 \div 4096 = 18.8515625$ $c_3 = 32 * 2392 \div 4096 = 18.6875$ $m = 128 * 2523 \div 4096 = 78.84375$ $n = 0.25 * 2610 \div 4096 = 0.1593017578125$ 其中，峰白区 L_c 等于1通常是为了表示参考输出发光亮度，即10000坎德拉每平方米。
HLG	$V = a * \ln(12 * L_c - b) + c$ for $1 \geq L_c > 1 \div 12$ $V = \text{Sqrt}(3) * L_c^{0.5}$ for $1 \div 12 \geq L_c \geq 0$ $a = 0.17883277, b = 0.28466892, c = 0.55991073$

2.10.8.2.6 hi_color_gamut

说明

定义色域范围。

定义

```
HI_COLOR_GAMUT_BT601 = 0
HI_COLOR_GAMUT_BT709 = 1
HI_COLOR_GAMUT_BT2020 = 2
HI_COLOR_GAMUT_USER = 3
HI_COLOR_GAMUT_BUTT = 4
```

成员

成员名称	描述
HI_COLOR_GAMUT_BT601	BT601色域。
HI_COLOR_GAMUT_BT709	BT709色域。
HI_COLOR_GAMUT_BT2020	BT2020色域。
HI_COLOR_GAMUT_USER	用户自定义的色域，非标准色域。
HI_COLOR_GAMUT_BUTT	保留值。

2.10.8.2.7 hi_video_supplement

说明

定义视频图像帧补充信息。

定义

```
hi_video_supplement = {"misc_info_phys_addr": misc_info_phys_addr, # default allocated buffer
                        "jpeg_dcf_phys_addr": jpeg_dcf_phys_addr,
                        "isp_info_phys_addr": isp_info_phys_addr,
                        "low_delay_phys_addr": low_delay_phys_addr,
                        "bnr_rnt_phys_addr": bnr_rnt_phys_addr,
                        "motion_data_phys_addr": motion_data_phys_addr,
                        "frame_dng_phys_addr": frame_dng_phys_addr,
                        "misc_info_virt_addr": misc_info_virt_addr, # misc info
                        "jpeg_dcf_virt_addr": jpeg_dcf_virt_addr, # jpeg_dcf, used in JPEG DCF
                        "isp_info_virt_addr": isp_info_virt_addr, # isp_frame_info, used in ISP debug, when get raw
                        "low_delay_virt_addr": low_delay_virt_addr, # used in low delay
                        "bnr_mot_virt_addr": bnr_mot_virt_addr, # used for 3dnr from bnr mot
                        "motion_data_virt_addr": motion_data_virt_addr, # vpss 3dnr use: gme motion data, filter
                        "frame_dng_virt_addr": frame_dng_virt_addr}
```

成员

成员名称	描述
misc_info_phys_addr	整型，视频扩展信息的物理地址。暂不支持。
jpeg_dcf_phys_addr	整型，JPEG DCF (Design rule for Camera File system) 信息的物理地址。暂不支持，用户不能配置，系统使用默认值0。
isp_info_phys_addr	整型，ISP辅助信息的物理地址。用户不能配置修改。
low_delay_phys_addr	整型，低延时信息物理地址。暂不支持。
bnr_rnt_phys_addr	整型，预留参数。暂不支持，用户不能配置修改。
motion_data_phys_addr	整型，运动数据物理地址。暂不支持，用户不能配置修改。
frame_dng_phys_addr	整型，预留参数。暂不支持，用户不能配置修改。
misc_info_virt_addr	整型，视频扩展信息的虚拟地址。暂不支持，用户不能配置修改。
jpeg_dcf_virt_addr	整型，JPEG DCF信息的虚拟地址。暂不支持，用户不能配置修改。
isp_info_virt_addr	整型，ISP辅助信息的内核态虚拟地址。暂不支持。
low_delay_virt_addr	整型，低延时信息虚拟地址。暂不支持。
bnr_mot_virt_addr	整型，BNR信息虚拟地址。暂不支持。
motion_data_virt_addr	整型，运动信息虚拟地址。暂不支持。
frame_dng_virt_addr	整型，DNG格式信息的虚拟地址。暂不支持。

2.10.8.2.8 hi_video_frame

说明

定义视频原始图像帧字典。

定义

```
hi_video_frame = {"width": width,
                  "height": height,
                  "field": hi_video_field,
                  "pixel_format": hi_pixel_format,
                  "video_format": hi_video_format,
                  "compress_mode": hi_compress_mode,
                  "dynamic_range": hi_dynamic_range,
                  "color_gamut": hi_color_gamut,
                  "header_stride": [header_stride, header_stride, header_stride],
                  "width_stride": [width_stride, width_stride, width_stride],
                  "height_stride": [height_stride, height_stride, height_stride],
                  "header_phys_addr": [header_phys_addr, header_phys_addr, header_phys_addr],
                  "phys_addr": [phys_addr, phys_addr, phys_addr],
                  "header_virt_addr": [header_virt_addr_list, header_virt_addr_list, header_virt_addr_list],
```

```
"virt_addr": [virt_addr, virt_addr, virt_addr],  
"time_ref": time_ref,  
"pts": pts,  
"user_data": [user_data],  
"frame_flag": frame_flag, # frame_flag, can be OR operation.  
"supplement": hi\_video\_supplement}
```

成员

成员名称	描述
width	整型，图像宽度。
height	整型，图像高度。
field	整型，帧场模式。预留参数。
pixel_format	整型，视频图像像素格式。
video_format	整型，视频图像格式。
compress_mode	整型，视频压缩模式。
dynamic_range	整型，动态范围。预留参数。
color_gamut	整型，色域范围。预留参数。
header_stride	列表，图像压缩头跨距。预留参数。
width_stride	列表，输出图像分量的宽度数据跨距。YUV图像则为Y、U、V分量的数据跨距。RGB图像则为R、G、B分量的数据跨距。
height_stride	列表，输出图像分量的高度数据跨距。YUV图像则为Y、U、V分量的数据跨距。RGB图像则为R、G、B分量的数据跨距。 若涉及VENC视频编码、JPEG图片编码功能时，该参数作为预留参数，暂不支持。
header_phys_addr	列表，压缩头物理地址。预留参数。
phys_addr	列表，物理地址。预留参数。
header_virt_addr	列表，压缩头虚拟地址。预留参数。
virt_addr	列表，图像在Device内存中的起始虚拟地址。 作为输入图像时，对于YUV图像，virt_addr[0]为图像起始地址，也是Y分量起始地址，virt_addr[1]为U分量起始地址，virt_addr[2]为V分量起始地址； 作为输出图像时，对于YUV和RGB图像，virt_addr[0]为图像在device上的起始地址，virt_addr[1]和virt_addr[2]为保留字段。

成员名称	描述
time_ref	整型，图像帧序列号。 解码场景下，该参数预留。 JPEGE编码场景下，该参数预留。 VENC编码场景下，针对每一帧设置该参数时，按照帧的顺序，该参数值需递增且为偶数，例如，第一帧时该参数值设置为2，第二帧时该参数值设置为4，以此类推。
pts	整型，图像时间戳。
user_data	列表，私有数据。预留参数。

成员名称	描述
frame_flag	整形。 - 在VDEC解码场景下，该参数表示该帧解码是否成功，不管成功或失败，用户都需及时释放输入、输出内存，对于解码失败的帧，系统不保证按照显示序或解码序按顺序输出，本参数取值范围： 0：解码成功。 1：解码失败，可能由于输入码流异常、设置的码流格式与实际码流不一致等问题导致。 2：隔行码流场景下使用，隔行码流每帧发送两场，带两块输出内存；解码时其中一块内存无图像输出，属于正常现象，用户需及时释放内存；隔行码流的解码输出数据都在奇数场对应的输出Buffer中。 3：参考帧个数设置错误导致解码失败。 4：VDEC解码帧存大小设置错误导致解码失败。 - 在JPEGD解码场景下，该参数表示该帧解码是否成功，不管成功或失败，用户都需及时释放输入、输出内存，本参数取值范围： 0：解码成功。 1：解码失败，可能由于输入码流异常、设置的码流格式与实际码流不一致等问题导致。 4：JPEGD解码输出内存大小过小导致解码失败，可提前调用acl.himpi.vdec_get_jpegd_output_info接口获取输出内存大小。 5：仅JPEGD解码场景使用，表示JPEGD解码配置参数设置错误，例如，设置的宽高超出限制、设置的目标解码格式不支持等。 - 在VENC编码场景下，不同bit位用于使能不同的功能： - 该参数的bit0表示是否使能 slice_temporal_mvp_enabled_flag语法，表示帧间预测是否使用时域MV（MotionVector）预测，取值范围如下，但仅H.265协议下该参数生效： 0：slice_temporal_mvp_enabled_flag为True，表示使用时域MV预测。 1：slice_temporal_mvp_enabled_flag为False，表示不使用时域MV预测。 - 该参数的bit1表示是否使能图像扩边，取值范围如下，H.264/H.265协议下该参数生效： 0：表示使能图像扩边。 1：表示关闭图像扩边。 - 在JPEGE编码场景下，取值0或1，但当前版本暂未使用该参数，参数值无效。

成员名称	描述
supplement	字典，图像的补充信息。 Atlas 200I/500 A2推理产品 上，该参数为透传参数。 Atlas 推理系列产品 上，该参数为透传参数。 Atlas A2训练系列产品/Atlas 800I A2推理产品 上，该参数为透传参数。

注意事项

- 10bit 数据不压缩时在内存中的存储方式是是紧凑排列的。
- 10bit YUV 段压缩数据存储是前 8bit 和后 2bit 分开存储。
- 10bit tile 64x16 数据（即 VDH（video decoder hardware）解码之后的数据）不管是压缩还是非压缩，在内存中的存储方式都是前 8bit 和后 2bit 分开存储。

2.10.8.2.9 hi_video_frame_info

说明

定义视频图像帧信息字典。

定义

```
hi_video_frame_info = {"v_frame": hi\_video\_frame, "pool_id" pool_id, "mod_id": hi\_mod\_id}
```

成员

成员名称	描述
v_frame	字典，视频图像帧属性字典。
pool_id	整型，视频缓存池ID，预留参数。
mod_id	整型，DVPP内的子模块ID，预留参数。

2.10.8.2.10 hi_vb_src

说明

定义VB来源选择。

定义

```
HI_VB_SRC_COMMON = 0 #公共VB  
HI_VB_SRC_MOD    = 1 #模块VB  
HI_VB_SRC_PRIVATE = 2 #私有VB  
HI_VB_SRC_USER   = 3 #用户VB  
HI_VB_SRC_BUTT   = 4
```

2.10.8.2.11 hi_mod_id

说明

定义模块ID类型。

定义

```
HI_ID_CMPI      = 0
HI_ID_VB        = 1
HI_ID_SYS       = 2
HI_ID_RGN       = 3
HI_ID_CHNL      = 4
HI_ID_VDEC      = 5
HI_ID_AVS       = 6
HI_ID_VPC       = 7
HI_ID_VENC      = 8
HI_ID_SVP       = 9
HI_ID_H264E     = 10
HI_ID_JPEG      = 11
HI_ID_MPEG4E    = 12
HI_ID_H265E     = 13
HI_ID_JPEGD     = 14
HI_ID_VO        = 15
HI_ID_VI        = 16
HI_ID_DIS       = 17
HI_ID_VALG      = 18
HI_ID_RC        = 19
HI_ID_AIO       = 20
HI_ID_AI        = 21
HI_ID_AO        = 22
HI_ID_AENC      = 23
HI_ID_ADEC      = 24
HI_ID_VPU       = 25
HI_ID_PCIV      = 26
HI_ID_PCIVFMW   = 27
HI_ID_ISP       = 28
HI_ID_IVE       = 29
HI_ID_USER      = 30
HI_ID_DCCM      = 31
HI_ID_DCCS      = 32
HI_ID_PROC      = 33
HI_ID_LOG       = 34
HI_ID_VFMW      = 35
HI_ID_H264D     = 36
HI_ID_GDC       = 37
HI_ID_PHOTO     = 38
HI_ID_FB        = 39
HI_ID_HDMI      = 40
HI_ID_VOIE      = 41
HI_ID_TDE       = 42
HI_ID_HDR       = 43
HI_ID_PRORES    = 44
HI_ID_VGS       = 45
HI_ID_FD        = 47
HI_ID_ODT       = 48 # Object detection trace
HI_ID_VQA       = 49 # Video quality analysis
HI_ID_LPR       = 50 # Object detection trace
HI_ID_SVP_NNIE  = 51
HI_ID_SVP_DSP   = 52
HI_ID_DPU_RECT  = 53
HI_ID_DPU_MATCH = 54
HI_ID_MOTIONSENSOR = 55
HI_ID_MOTIONFUSION = 56
HI_ID_GYRODIS   = 57
HI_ID_PM        = 58
HI_ID_SVP_ALG   = 59
HI_ID_IVP       = 60
```



```
HI_ID_MCF      = 61
HI_ID_VPSS     = 62
HI_ID_DRV_VPC  = 63
HI_ID_PNGD     = 64
HI_ID_BUTT     = 256
```

2.10.8.2.12 hi_data_bit_width

说明

定义数据比特位宽类型。

目前DVPP模块仅支持“HI_DATA_BIT_WIDTH_8”和“HI_DATA_BIT_WIDTH_10”两个数据位宽格式。

定义

```
HI_DATA_BIT_WIDTH_8  = 0
HI_DATA_BIT_WIDTH_10 = 1
HI_DATA_BIT_WIDTH_12 = 2
HI_DATA_BIT_WIDTH_14 = 3
HI_DATA_BIT_WIDTH_16 = 4
HI_DATA_BIT_WIDTH_BUTT = 5
```

2.10.8.2.13 hi_video_format

说明

定义视频格式。

定义

```
HI_VIDEO_FORMAT_LINEAR = 0 # nature video line
HI_VIDEO_FORMAT_TILE_64x16 = 1 # tile cell: 64pixel x 16line
HI_VIDEO_FORMAT_BUTT = 2
```

成员

成员名称	描述
HI_VIDEO_FORMAT_LINEAR	线性存储的视频格式。
HI_VIDEO_FORMAT_TILE_64x16	TILE 格式存储的视频格式，其中，tile的块大小为64像素宽，16行像素高，视频/图像编码暂不支持。
HI_VIDEO_FORMAT_BUTT	保留值。

2.10.8.2.14 hi_img_info

说明

定义图片信息的字典。

定义

```
hi_img_info = {"width": width,  
               "height": height,  
               "width_stride": width_stride,  
               "height_stride": height_stride,  
               "img_buf_size": img_buf_size,  
               "pixel_format": hi\_jpeg\_raw\_format/hi\_png\_color\_format,  
               "reserved": [4]}
```

成员

成员名称	描述
width	整型，图片宽。
height	整型，图片高。
width_stride	整型，图片宽Stride。
height_stride	整型，图片高Stride。
img_buf_size	整型，存放图片的内存大小，单位是byte。
pixel_format	整型，图片格式。
reserved	预留参数，当前版本请设置为全0列表，且不应显式使用该字段。

2.10.8.2.15 hi_pic_buf_attr

说明

定义图片缓冲区信息的字典。

定义

```
hi_pic_buf_attr = {"width": width,  
                   "height": height,  
                   "align": align,  
                   "bit_width": hi\_data\_bit\_width,  
                   "pixel_format": hi\_pixel\_format,  
                   "compress_mode": hi\_compress\_mode}
```

成员

成员名称	描述
width	整型，图片宽。
height	整型，图片高。
align	整型，对齐参数，预留参数，暂不支持。
bit_width	整型，图片位宽。

成员名称	描述
pixel_format	整型，图片格式。VDEC视频解码时，暂不使用该参数；JPEGD图像解码时，表示解码输出图像格式，当前支持的输出图像格式请参见 2.10.5.1 JPEGD功能及约束说明 。
compress_mode	整型，压缩模式。

2.10.8.2.16 hi_jpeg_raw_format

说明

定义JPEG源图片格式。

定义

```
HI_JPEG_RAW_FORMAT_YUV444 = 0
HI_JPEG_RAW_FORMAT_YUV422 = 1
HI_JPEG_RAW_FORMAT_YUV420 = 2
HI_JPEG_RAW_FORMAT_YUV440 = 3
HI_JPEG_RAW_FORMAT_YUV400 = 4
HI_JPEG_RAW_FORMAT_YUV411 = 5
HI_JPEG_RAW_FORMAT_MAX   = 100
```

2.10.8.2.17 hi_dvpp_epoll_ctl_op

说明

定义DVPP epoll操作类型。

定义

```
HI_DVPP_EPOLL_CTL_ADD = 1
HI_DVPP_EPOLL_CTL_MOD = 2
HI_DVPP_EPOLL_CTL_DEL = 3
HI_DVPP_EPOLL_CTL_BUTT = 4
```

成员

成员名称	描述
HI_DVPP_EPOLL_CTL_ADD	向epoll实例中添加新的DVPP通道的文件句柄。
HI_DVPP_EPOLL_CTL_MOD	修改epoll实例中已有DVPP通道的处理事件类型或对应的用户数据。
HI_DVPP_EPOLL_CTL_DEL	从epoll实例中删除DVPP通道的文件句柄。
HI_DVPP_EPOLL_CTL_BUTT	保留值。

2.10.8.2.18 hi_dvpp_epoll_event_type

说明

定义DVPP epoll事件类型。

定义

```
HI_DVPP_EPOLL_IN = 1
HI_DVPP_EPOLL_OUT = 1 << 1
HI_DVPP_EPOLL_ET = 1 << 31
```

成员

成员名称	描述
HI_DVPP_EPOLL_IN	可读事件，标识DVPP通道中存在输出数据。
HI_DVPP_EPOLL_OUT	可写事件，标识DVPP通道可以进行新的数据输入。预留功能，暂不支持。
HI_DVPP_EPOLL_ET	边缘触发方式，不设定则DVPP epoll默认为水平触发。

2.10.8.2.19 hi_dvpp_epoll_event

说明

定义DVPP epoll事件信息。

定义

```
hi_dvpp_epoll_event = {"events": events, "data": data}
```

成员

成员名称	描述
events	整型，DVPP epoll事件类型，由 hi_dvpp_epoll_event_type 中定义的类型组成的位掩码（bitmask），例如处理可读和可写两种事件时，此处配置为：DVPP_EPOLL_IN DVPP_EPOLL_OUT。
data	整型，关联到DVPP通道的用户自定义数据。

2.10.8.2.20 hi_csc_matrix

说明

定义色域转换矩阵的数据标准。

基于各标准的色域转换矩阵，其各分量的转换公式如下：

```
# YUV转RGB:
# | R | | matrix_r0c0 matrix_r0c1 matrix_r0c2 | | Y - input_bias_0 |
# | G | = | matrix_r1c0 matrix_r1c1 matrix_r1c2 | * | U - input_bias_1 |
# | B | | matrix_r2c0 matrix_r2c1 matrix_r2c2 | | V - input_bias_2 |
# BGR转YUV:
# | Y | | output_bias_0 | | matrix_r0c0 matrix_r0c1 matrix_r0c2 | | R |
# | U | = | output_bias_1 | + | matrix_r1c0 matrix_r1c1 matrix_r1c2 | * | G |
# | V | | output_bias_2 | | matrix_r2c0 matrix_r2c1 matrix_r2c2 | | B |
```

定义

```
HI_CSC_MATRIX_BT601_WIDE = 0
HI_CSC_MATRIX_BT601_NARROW = 1
HI_CSC_MATRIX_BT709_WIDE = 2
HI_CSC_MATRIX_BT709_NARROW = 3
HI_CSC_MATRIX_BT2020_WIDE = 4
HI_CSC_MATRIX_BT2020_NARROW = 5
HI_CSC_MATRIX_USER = 100
HI_CSC_MATRIX_BUTT = 101
```

成员

成员名称	描述
HI_CSC_MATRIX_BT601_WIDE	基于BT601 wide标准的色域转换矩阵。默认为“HI_CSC_MATRIX_BT601_WIDE”。 基于该标准的色域转换矩阵，各参数值如下： # YUV转RGB: # R 1.000 0.000 1.402 Y - 0 # G = 1.000 -0.344 -0.714 * U - 128 # B 1.000 1.772 0.000 V - 128 # RGB转YUV: # Y -0.5 0.299 0.587 0.114 R # U = 127.5 + -0.168 -0.331 0.500 * G # V 127.5 0.500 -0.419 -0.081 B
HI_CSC_MATRIX_BT601_NARROW	基于BT601 narrow标准的色域转换矩阵。 基于该标准的色域转换矩阵，各参数值如下： # YUV转RGB: # R 1.16438 0.00000 1.59602 Y - 16 # G = 1.16438 -0.39176 -0.81297 * U - 128 # B 1.16438 2.01723 0.00000 V - 128 # RGB转YUV: # Y 16 0.25679 0.51564 0.10014 R # U = 128 + -0.14491 -0.29099 0.43922 * G # V 128 0.42941 -0.36779 -0.07143 B
HI_CSC_MATRIX_BT709_WIDE	基于BT709 wide标准的色域转换矩阵。 基于该标准的色域转换矩阵，各参数值如下： # YUV转RGB: # R 1.00000 0.00000 1.57480 Y - 0 # G = 1.00000 -0.18732 -0.46812 * U - 128 # B 1.00000 1.85560 0.00000 V - 128 # RGB转YUV: # Y 0 0.21260 0.71520 0.07220 R # U = 128 + -0.11457 -0.38543 0.50000 * G # V 128 0.50000 -0.45415 -0.04585 B

成员名称	描述
HI_CSC_MATRIX_BT709_NARROW	基于BT709 narrow标准的色域转换矩阵。 基于该标准的色域转换矩阵，各参数值如下： # YUV转RGB： # R 1.16438 0.00000 1.79274 Y - 16 # G = 1.16438 -0.21325 -0.53291 * U - 128 # B 1.16438 2.11240 0.00000 V - 128 # RGB转YUV： # Y 16 0.18259 0.62825 0.06342 R # U = 128 + -0.09840 -0.33857 0.43922 * G # V 128 0.42941 -0.39894 -0.04027 B
HI_CSC_MATRIX_BT2020_WIDE	基于BT2020 wide标准的色域转换矩阵。 基于该标准的色域转换矩阵，各参数值如下： # YUV转RGB： # R 1.00000 0.00000 1.47460 Y - 0 # G = 1.00000 -0.16455 -0.57135 * U - 128 # B 1.00000 1.88140 0.00000 V - 128 # RGB转YUV： # Y 0 0.26270 0.67800 0.05930 R # U = 128 + -0.13963 -0.36037 0.50000 * G # V 128 0.50000 -0.45979 -0.04021 B
HI_CSC_MATRIX_BT2020_NARROW	基于BT2020 narrow标准的色域转换矩阵。 基于该标准的色域转换矩阵，各参数值如下： # YUV转RGB： # R 1.16438 0.00000 1.67868 Y - 16 # G = 1.16438 -0.18733 -0.65042 * U - 128 # B 1.16438 2.14177 0.00000 V - 128 # RGB转YUV： # Y 16 0.22564 0.59558 0.05209 R # U = 128 + -0.11992 -0.31656 0.43922 * G # V 128 0.42941 -0.40389 -0.03533 B
HI_CSC_MATRIX_USER	用户自定义色域转换矩阵。
HI_CSC_MATRIX_BUTT	保留值。

2.10.8.2.21 hi_coefficient

说明

定义色域转换矩阵参数。

定义

```
hi_coefficient = {"csc_matrix_r0_c0": csc_matrix_r0_c0,  
                  "csc_matrix_r0_c1": csc_matrix_r0_c1  
                  "csc_matrix_r0_c2": csc_matrix_r0_c2  
                  "csc_matrix_r1_c0": csc_matrix_r1_c0  
                  "csc_matrix_r1_c1": csc_matrix_r1_c1  
                  "csc_matrix_r1_c2": csc_matrix_r1_c2  
                  "csc_matrix_r2_c0": csc_matrix_r2_c0  
                  "csc_matrix_r2_c1": csc_matrix_r2_c1  
                  "csc_matrix_r2_c2": csc_matrix_r2_c2  
                  "csc_bias_r0": csc_bias_r0  
                  "csc_bias_r1": csc_bias_r1  
                  "csc_bias_r2": csc_bias_r2}
```

成员

成员名称	描述
csc_matrix_r0_c0	整型，设置色域转换矩阵参数R0C0。
csc_matrix_r0_c1	整型，设置色域转换矩阵参数R0C1。
csc_matrix_r0_c2	整型，设置色域转换矩阵参数R0C2。
csc_matrix_r1_c0	整型，设置色域转换矩阵参数R1C0。
csc_matrix_r1_c1	整型，设置色域转换矩阵参数R1C1。
csc_matrix_r1_c2	整型，设置色域转换矩阵参数R1C2。
csc_matrix_r2_c0	整型，设置色域转换矩阵参数R2C0。
csc_matrix_r2_c1	整型，设置色域转换矩阵参数R2C1。
csc_matrix_r2_c2	整型，设置色域转换矩阵参数R2C2。
csc_bias_r0	整型，设置偏移R0。
csc_bias_r1	整型，设置偏移R1。
csc_bias_r2	整型，设置偏移R2。

2.10.8.2.22 hi_csc_coefficient

说明

定义色域转换矩阵参数的字典。

定义

```
hi_csc_coefficient = {"yuv_to_rgb_coefficient": hi_coefficient,  
                    "rgb_to_yuv_coefficient": hi_coefficient}
```

成员

成员名称	描述
yuv_to_rgb_coefficient	字典，YUV转RGB的色域转换参数。
rgb_to_yuv_coefficient	字典，RGB转YUV的色域转换参数。

2.10.8.2.23 hi_video_size

说明

预留字典，暂不支持。

定义

```
hi_video_size = {"width": width, "height": height}
```

2.10.8.2.24 hi_rect

说明

定义矩形宽高和左上角坐标。

定义

```
hi_rect = {"x": x,  
           "y": y,  
           "width": width,  
           "height": height}
```

成员

成员名称	描述
x	整型，左上角x轴。
y	整型，左上角y轴。
width	整型，矩形宽度。
height	整型，矩形高度。

2.10.8.2.25 hi_op_mode

说明

定义操作模式枚举。

定义

```
HI_OP_MODE_AUTO = 0 # 自动模式，一般此模式使用程序内部的默认参数  
HI_OP_MODE_MANUAL = 1 # 手动模式，一般此模式使用用户配置参数
```

2.10.8.2.26 hi_point

说明

定义坐标点字典。

定义

```
hi_point = {"x": x,  
            "y": y}
```


成员

成员名称	描述
x	整型，x轴坐标值。
y	整型，y轴坐标值。

2.10.8.2.27 hi_size

说明

定于矩形尺寸的字典。

定义

```
hi_size = {"width": width,  
           "height": height}
```

成员

成员名称	描述
width	整型，宽度。
height	整型，高度。

2.10.8.2.28 hi_compress_mode

说明

定义视频压缩数据格式枚举。

定义

```
HI_COMPRESS_MODE_NONE = 0  
HI_COMPRESS_MODE_SEG = 1  
HI_COMPRESS_MODE_TILE = 2  
HI_COMPRESS_MODE_HFBC = 3  
HI_COMPRESS_MODE_LINE = 4  
HI_COMPRESS_MODE_FRAME = 5  
HI_COMPRESS_MODE_BUTT = 6
```

成员

成员名称	描述
HI_COMPRESS_MODE_NONE	不压缩。
HI_COMPRESS_MODE_SEG	段压缩，以256字节为一个压缩片段。

成员名称	描述
HI_COMPRESS_MODE_TILE	Tile压缩，以一个tile为一个压缩片段。
HI_COMPRESS_MODE_HFBC	本昇腾AI处理器特有的HFBC压缩模式，压缩单位是64 * 8。
HI_COMPRESS_MODE_LINE	行压缩，以一整行为一个压缩片段进行压缩。如ISP WDR模式下内部的RAW图像采用行压缩。
HI_COMPRESS_MODE_FRAME	帧压缩，以一整帧为一个压缩片段进行压缩。如ISP线性模式下内部的RAW图像采用帧压缩，或者3DNR内部参考帧和重构帧采用帧压缩。
HI_COMPRESS_MODE_BUTT	预留值。

2.10.8.2.29 hi_module_type

说明

定义模块类型。

定义

```
HI_MOD_VPC = 0
HI_MOD_VENC = 1
HI_MOD_VDEC = 2
HI_MOD_JPEGE = 3
HI_MOD_JPEGD = 4
HI_MOD_PNGD = 5
HI_MOD_ALL = 100
HI_MOD_BUTT = 255 // 预留值
```

2.10.8.2.30 hi_img_base_info

说明

定义图片基本信息。

定义

```
hi_img_base_info = {"width": width,
                    "height": height,
                    "pixel_format": hi_pixel_format,
                    "reserved": [2]}
```

成员

成员名称	描述
width	整型，图片宽度。
height	整型，图片高度。

成员名称	描述
pixel_format	整型，图片格式。
reserved	列表，预留属性，为保证后续版本兼容性，请务必传入[0, 0]进行清零初始化，在代码中必须避免显式对reserved字段进行访问。

2.10.8.2.31 hi_img_align_info

说明

定义图片对齐信息。

定义

```
hi_img_align_info = {"width_stride": width_stride,  
                    "height_stride": height_stride,  
                    "img_buf_size": img_buf_size,  
                    "reserved": [2]}
```

成员

成员名称	描述
width_stride	整型，图片宽度对齐后的值。
height_stride	整型，图片高度对齐后的值。
img_buf_size	整型，存放图片数据所需的内存大小，单位Byte。
reserved	列表，预留属性，为保证后续版本兼容性，请务必传入[0, 0]进行清零初始化，在代码中必须避免显式对“reserved”字段进行访问。

2.10.8.3 视频/图像解码

2.10.8.3.1 hi_vdec_send_mode

说明

定义码流发送方式。

定义

```
HI_VDEC_SEND_MODE_STREAM = 0 # send by stream  
HI_VDEC_SEND_MODE_FRAME = 1 # send by frame  
HI_VDEC_SEND_MODE_COMPAT = 2 # One frame supports multiple packets sending.  
HI_VDEC_SEND_MODE_BUTT = 3 # reserved
```

成员

成员名称	描述
HI_VDEC_SEND_MODE_STREAM	流模式，当前不支持。
HI_VDEC_SEND_MODE_FRAME	帧模式，当前仅支持按帧发送。
HI_VDEC_SEND_MODE_COMPAT	兼容模式，当前不支持。
HI_VDEC_SEND_MODE_BUTT	保留值。

注意事项

- 仅支持“HI_VDEC_SEND_MODE_FRAME”。
- 当一帧码流的最后一包没有把“end_of_frame”为“HI_TRUE”时，还可以再发送一个“end_of_frame”为“HI_TRUE”的空包（注意带上当前帧的PTS（Presentation Time Stamp））给解码器内部标识当前帧码流已发送完毕。
- 不支持一帧图像分多包进行发送的模式。

2.10.8.3.2 hi_vdec_frame_type

说明

定义码流发送方式。

定义

```
HI_VDEC_FRAME_TYPE_I = 0
HI_VDEC_FRAME_TYPE_P = 1
HI_VDEC_FRAME_TYPE_B = 2
HI_VDEC_FRAME_TYPE_BUTT = 3
```

成员

成员名称	描述
HI_VDEC_FRAME_TYPE_I	IDR帧。
HI_VDEC_FRAME_TYPE_P	P帧。
HI_VDEC_FRAME_TYPE_B	B帧。
HI_VDEC_FRAME_TYPE_BUTT	保留值。

2.10.8.3.3 hi_quick_mark_mode

说明

定义快速释放参考帧模式。

定义

```
HI_QUICK_MARK_ADAPT = 0
HI_QUICK_MARK_FORCE = 1
HI_QUICK_MARK_NONE = 2
HI_QUICK_MARK_BUTT = 3
```

成员

成员名称	描述
HI_QUICK_MARK_ADAPT	自适应模式。预留参数，暂未支持。
HI_QUICK_MARK_FORCE	强制模式。预留参数，暂未支持。
HI_QUICK_MARK_NONE	普通模式。预留参数，暂未支持。
HI_QUICK_MARK_BUTT	保留值。

2.10.8.3.4 hi_vdec_pic_info

说明

定义视频原始图像帧字典。

定义

```
hi_vdec_pic_info = {"width": width,
                    "height": height,
                    "width_stride": width_stride,
                    "height_stride": height_stride,
                    "pixel_format": hi_pixel_format,
                    "vir_addr": vir_addr,
                    "buffer_size": buffer_size,
                    "offset_top": offset_top,
                    "offset_bottom": offset_bottom,
                    "offset_left": offset_left,
                    "offset_right": offset_right}
```

成员

成员名称	描述
width	整型，图像宽度，视频解码时使用该参数。 ● Atlas 推理系列产品，视频解码场景下，支持对输出图片进行缩放，缩放算法默认为业界通用的Bilinear算法（与OpenCV算法的计算过程类似），解码图像经过缩放后的输出宽取值范围为[10, 4096]。 以下场景，VDEC不对输出图片进行缩放：在“hi_vdec_pic_info”字典内，将“width”属性值和“height”属性值同时设置为0。 ● Atlas A2训练系列产品/Atlas 800I A2推理产品，视频解码场景下，支持对输出图片进行缩放，缩放算法默认为业界通用的Bilinear算法（与OpenCV算法的计算过程类似），解码图像经过缩放后的输出宽取值范围为[10, 4096]。 以下场景，VDEC不对输出图片进行缩放：在“hi_vdec_pic_info”字典内，将“width”属性值和“height”属性值同时设置为0。 ● Atlas 200I/500 A2推理产品，视频解码场景下，支持对输出图片进行缩放，缩放算法默认为业界通用的Bilinear算法（与OpenCV算法的计算过程类似），当输入码流宽度小于或等于4096时，解码图像经过缩放后的输出宽度范围为[10, 8192]；当输入码流宽度大于4096时，解码图像经过缩放后的输出宽度范围为[128, 8192]。 以下场景，VDEC不对输出图片进行缩放：在“hi_vdec_pic_info”字典内，将“width”属性值和“height”属性值同时设置为0。
height	整型，图像高度，视频解码时使用该参数。 ● Atlas 推理系列产品，视频解码场景下，支持对输出图片进行缩放，缩放算法默认为业界通用的Bilinear算法（与OpenCV算法的计算过程类似），解码图像经过缩放后的输出高度范围为[6, 4096]。 ● Atlas A2训练系列产品/Atlas 800I A2推理产品，视频解码场景下，支持对输出图片进行缩放，缩放算法默认为业界通用的Bilinear算法（与OpenCV算法的计算过程类似），解码图像经过缩放后的输出高度范围为[6, 4096]。 ● Atlas 200I/500 A2推理产品，视频解码场景下，支持对输出图片进行缩放，缩放算法默认为业界通用的Bilinear算法（与OpenCV算法的计算过程类似），当输入码流宽度小于或等于4096时，解码图像经过缩放后的输出高度范围为[6, 8192]；当输入码流宽度大于4096时，解码图像经过缩放后的输出高度范围为[128, 8192]。

成员名称	描述
width_stride	整型，图像宽度步长，视频解码时，YUV420 semi-planar/YVU420 semi-planar输出时，该值为宽向上16对齐；RGB888/BGR888输出时，该值为宽向上16对齐，再乘3。 取值范围：[32, 16384]。
height_stride	整型，图像高度步长，视频解码时，YUV420 semi-planar/YVU420 semi-planar输出时，该值为高向上2对齐；RGB888/BGR888输出时，该值等于高。 取值范围：[6, 16384]。
pixel_format	整型，图像像素格式。图像或视频解码时使用该参数。VDEC视频解码输出图片格式参见 2.10.5.3 VDEC功能及约束说明 ，JPEGD图片解码输出图片格式参见 2.10.5.1 JPEGD功能及约束说明 。
vir_addr	整型，解码输出数据在内存中的起始地址。 隔行码流场景下，隔行码流每帧发送两场，带两块输出内存；解码时其中一块内存无图像输出，属于正常现象，用户需及时释放内存；隔行码流的解码输出数据都在奇数场对应的输出Buffer中。
buffer_size	整型，存放解码图像的缓冲区大小。
offset_top	整型，解码区域相对原图的顶部偏移，JPEG图片解码场景下按指定区域解码时配置，如 图2-24 所示，若无需按指定区域解码，需将此值初始化为0。 Atlas 推理系列产品 不支持该参数，为保证后续版本兼容性，建议此参数配置为0。
offset_bottom	整型，解码区域相对原图的底部偏移，JPEG图片解码场景下按指定区域解码时配置，如 图2-24 所示，若无需按指定区域解码，需将此值初始化为0。 Atlas 推理系列产品 不支持该参数，为保证后续版本兼容性，建议此参数配置为0。
offset_left	整型，解码区域相对原图的左侧偏移，JPEG图片解码场景下按指定区域解码时配置，如 图2-24 所示，若无需按指定区域解码，需将此值初始化为0。 Atlas 推理系列产品 不支持该参数，为保证后续版本兼容性，建议此参数配置为0。
offset_right	整型，解码区域相对原图的右侧偏移，JPEG图片解码场景下按指定区域解码时配置，如 图2-24 所示，若无需按指定区域解码，需将此值初始化为0。 Atlas 推理系列产品 不支持该参数，为保证后续版本兼容性，建议此参数配置为0。

注意事项

10bit的数据经过VPC处理（图像处理）之后均会变成8bit。

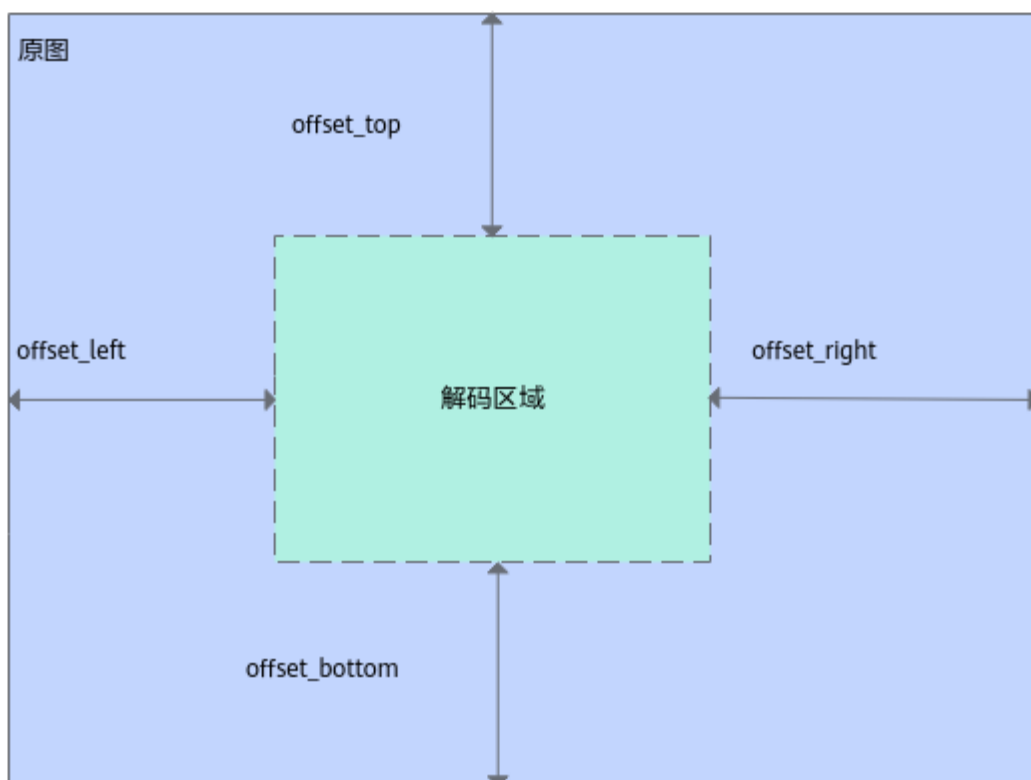
参考信息

JPEG图片解码场景下，按区域解码的功能示意图如下。

使用区域解码存在以下约束：

- 对于JPEG图片解码进行区域解码，不支持旋转、软件解码图片。
- 区域解码的范围不能超出原图大小，并且最小支持32 * 32，最大支持16384 * 16384。
- 使用以下图片格式时，对各偏移值的约束如下：
 - “HI_PIXEL_FORMAT_YUV_SEMIPLANAR_420” 或
“HI_PIXEL_FORMAT_YVU_SEMIPLANAR_420” 格式：“offset_top” 和
“offset_left” 必须为偶数，（height - offset_bottom）和（width -
offset_right）必须为偶数。
 - “HI_PIXEL_FORMAT_YUV_SEMIPLANAR_422” 或
“HI_PIXEL_FORMAT_YVU_SEMIPLANAR_422” 格式：“offset_left” 必须
为偶数，（width - offset_right）必须为偶数。
 - “HI_PIXEL_FORMAT_YUV_SEMIPLANAR_440” 或
“HI_PIXEL_FORMAT_YVU_SEMIPLANAR_440”：“offset_top” 必须为偶
数，（height - offset_bottom）必须为偶数。

图 2-24 按区域解码功能示意图



2.10.8.3.5 hi_vdec_video_attr

说明

定义视频解码视频属性。

定义

```
hi_vdec_video_attr = {"ref_frame_num": ref_frame_num, "temporal_mvp_en": temporal_mvp_en,  
"tmv_buf_size": tmv_buf_size}
```

成员

成员名称	描述
ref_frame_num	整型，参考帧的数目，用于决定解码时需要的参考帧个数，会较大的影响内存VB块占用，根据实际情况设置合适的值。如果此处设置的值大于实际码流中的参考帧数量，系统内部会根据实际码流的参考帧数量进行调整，以节省内存，但不超过当前所设置的参数值。 取值范围：[0, 16]，以帧为单位。 不知情的情况下，其他码流：推荐设为 5。 测试码流：推荐设为 16。 静态属性。
temporal_mvp_en	整型，是否支持时域运动矢量预测。 取值范围：[0, 1]。 如果H.264解码不需要解码B帧，或者H.265解码不需要解码支持时域运动矢量预测（sps_temporal_mvp_enabled_flag =1）的码流，则配置“temporal_mvp_en”为0，否则配置为1。 当配置为0时，可不分配输出Tmv信息的VB块，节省内存。 调用acl.himpi.vdec_set_chn_param接口设置通道参数时，只支持将“hi_video_dec_mode”设置为“HI_VIDEO_DEC_MODE_IPB”（表示解码I、P、B帧），因此解码H.264码流时只能将“temporal_mvp_en”配置为1。
tmv_buf_size	整型，视频解码图像Tmv（Temporal Motion Vector）Buffer大小，“temporal_mvp_en”为1时有效。 用户可先调用acl.himpi.vdec_get_tmv_buf_size接口获取视频解码图像TMV缓存大小。

2.10.8.3.6 hi_vdec_chn_attr

说明

定义解码通道属性字典。

定义

```
hi_vdec_chn_attr= {"type": hi_payload_type, # RW; video type to be decoded  
                  "mode": hi_vdec_send_mode, # RW; send by stream or by frame  
                  "pic_width": pic_width, # RW; max pic width  
                  "pic_height": pic_height, # RW; max pic height  
                  "stream_buf_size": stream_buf_size, # stream buf size(Byte)  
                  "frame_buf_size": frame_buf_size, # RW; frame buf size(Byte)  
                  "frame_buf_cnt": frame_buf_cnt, # RW; frame buf num  
                  "video_attr": hi_vdec_video_attr} # RW; structure with video
```

成员

成员名称	描述
type	整型，解码协议类型值，静态属性。
mode	整型，码流发送方式，静态属性。 Atlas 推理系列产品上，H.264/H.265/JPEG仅支持按帧发送码流。 Atlas 200I/500 A2推理产品上，VDEC视频解码场景下，H.264/H.265仅支持按帧发送码流；JPEG图片解码场景下，该参数预留。 Atlas A2训练系列产品/Atlas 800I A2推理产品上，VDEC视频解码场景下，H.264/H.265仅支持按帧发送码流；JPEG图片解码场景下，该参数预留。
pic_width	整型，通道支持的解码图像最大宽（以像素为单位），静态属性。 Atlas 200I/500 A2推理产品上，JPEG图片解码场景下，该参数预留。 Atlas A2训练系列产品/Atlas 800I A2推理产品上，JPEG图片解码场景下，该参数预留。
pic_height	整型，通道支持的解码图像最大高（以像素为单位），静态属性。 Atlas 200I/500 A2推理产品上，JPEG图片解码场景下，该参数预留。 Atlas A2训练系列产品/Atlas 800I A2推理产品上，JPEG图片解码场景下，该参数预留。
stream_buf_size	整型，码流缓存的大小，静态属性。 取值范围：大于或等于解码通道大小（宽 * 高）的 3/4 倍，即YUV420图像大小的一半（宽 * 高 * 3/2 * 1/2），小于或等于1024 * 1024 * 1024，以Byte为单位。 推荐值：一幅YUV420解码图像大小。即：宽 * 高 * 1.5。

成员名称	描述
frame_buf_size	整型，解码图像帧存Buffer大小，以Byte为单位。JPEGD图像解码不需要设置该参数。 取值范围：大于或等于0。 - 取值为0时，VDEC内部将根据实际码流信息自动分配合适的帧存大小。 - 取值大于0时，用户需保证（“帧存大小” * 参考帧个数 $\geq$ 实际所需最小帧存大小 * 实际所需最小参考帧个数），否则不能正常解码。 实际所需最小帧存大小，用户需先根据码流中的SPS（Sequence Parameter Set）信息获取输入码流的宽、高、比特位宽，再调用acl.himpi.vdec_get_pic_buf_size接口获取实际所需最小帧存大小。
frame_buf_cnt	整型，解码图像帧存个数。JPEGD图像解码不需要设置该参数。 取值范围：大于或等于0。 - 一般情况下，设置为0，VDEC内部将根据实际码流信息自动分配合适的帧存个数。 - 如果客户需要限制VDEC解码帧存大小，则设置为大于0的值，用户需保证（帧存大小 * 参考帧个数 $\geq$ 实际需要最小帧存大小 * 实际需要最小参考帧个数），否则不能正常解码。 实际所需最小参考帧个数，用户需先根据码流中的SPS（Sequence Parameter Set）信息获取码流参考帧个数，$H.264/H.265$解码所需帧存个数 = 参考帧 + 显示帧 + 1。
video_attr	字典，视频(H.264/H.265)解码通道属性。

注意事项

设置通道属性的图像宽和高，若用户清楚码流的具体宽高，建议以实际值来设置。若是不清楚，建议以此码流的最大规格来创建。但是最大规格是4096 * 4096，不能超过此值。

2.10.8.3.7 hi_h264_protocol_param

说明

与H.264协议相关的内存分配参数。

定义

```
hi_h264_protocol_param = {"max_slice_num": max_slice_num, # RW, Range: [1, 300]; max slice num support
                           "max_sps_num": max_sps_num, # RW, Range: [1, 32]; max sps num support
                           "max_pps_num": max_pps_num } # RW, Range: [1, 256]; max pps num support
```

成员

成员名称	描述
max_slice_num	整型，该通道解码支持的最大Slice个数。 取值范围：[1, 300]。 Default：16。
max_sps_num	整型，该通道解码支持的最大SPS个数。 取值范围：[1, 32]。 Default：2。
max_pps_num	整型，该通道解码支持的最大PPS个数。 取值范围：[1, 256]。 Default：2。

2.10.8.3.8 hi_h265_protocol_param

说明

定义与H.265协议相关的内存分配参数。

定义

```
hi_h265_protocol_param = {"max_slice_segment_num": max_slice_segment_num, # RW; max slice segment num support
                          "max_vps_num": max_vps_num, # RW; max vps num support
                          "max_sps_num": max_sps_num, # RW; max sps num support
                          "max_pps_num": max_pps_num } # RW; max pps num support
```

成员

成员名称	描述
max_slice_segment_num	整型，该通道解码支持的最大SliceSegment个数。 取值范围：[1, 600] Default：16。
max_vps_num	整型，该通道解码支持的最大VPS个数。 取值范围：[1, 16]。 Default：2。
max_sps_num	整型，该通道解码支持的最大SPS个数。 取值范围：[1, 16]。 Default：2。
max_pps_num	整型，该通道解码支持的最大PPS个数。 取值范围：[1, 64]。 Default：2。

2.10.8.3.9 hi_vdec_protocol_param

说明

定义与协议相关的内存分配参数。

定义

```
hi_vdec_protocol_param = {"type": type,          # RW; video type to be decoded, only h264 and h265 supported
                          "h264_param": hi_h264_protocol_param, # protocol param structure for h264
                          "h265_param": hi_h265_protocol_param } # protocol param structure for h265
```

成员

成员名称	描述
type	整型，解码通道支持的协议。
h264_param	字典，H.264协议参数。
h265_param	字典，H.265协议参数。

2.10.8.3.10 hi_video_dec_mode

说明

定义视频解码模式。

定义

```
HI_VIDEO_DEC_MODE_IPB = 0
HI_VIDEO_DEC_MODE_IP  = 1
HI_VIDEO_DEC_MODE_I   = 2
HI_VIDEO_DEC_MODE_BUTT = 3
```

成员

成员名称	描述
HI_VIDEO_DEC_MODE_IPB	IPB模式，即I、P、B帧都解码。
HI_VIDEO_DEC_MODE_IP	IP模式，即只解码I帧和P帧。暂不支持。
HI_VIDEO_DEC_MODE_I	I模式，即只解码I帧。暂不支持。
HI_VIDEO_DEC_MODE_BUTT	保留值。

2.10.8.3.11 hi_video_out_order

说明

定义视频解码输出顺序。

定义

```
HI_VIDEO_OUT_ORDER_DISPLAY = 0
HI_VIDEO_OUT_ORDER_DEC      = 1
HI_VIDEO_OUT_ORDER_BUTT     = 2
```

成员

成员名称	描述
HI_VIDEO_OUT_ORDE R_DISPLAY	显示序输出。
HI_VIDEO_OUT_ORDE R_DEC	解码序输出。
HI_VIDEO_OUT_ORDE R_BUTT	保留值。

注意事项

解码有B帧的码流应设置为显示序输出。

2.10.8.3.12 hi_vdec_video_param

说明

定义视频解码高级参数。

定义

```
hi_vdec_video_param = {"composite_dec_en": composite_dec_en,
                        "slice_input_en": slice_input_en,
                        "err_threshold": err_threshold,
                        "dec_mode": hi_video_dec_mode, # frames output order ,0: the same with display order , 1:
the same with decoder order
                        "out_order": hi_video_out_order,
                        "compress_mode": hi_compress_mode, # RW; compress mode
                        "video_format": hi_video_format, # RW; video format
                        "quick_mark_mode": hi_quick_mark_mode }
```

成员

成员名称	描述
composite_dec_en	整型，复合码流使能开关。预留参数，暂未支持。
slice_input_en	预留参数，暂未支持。

成员名称	描述
err_threshold	预留参数，暂未支持。
dec_mode	整型, 解码模式。 默认值：HI_VIDEO_DEC_MODE_IPB。
out_order	整型，解码图像输出顺序。 默认值：HI_VIDEO_OUT_ORDER_DISPLAY
compress_mode	整型，解码图像压缩模式。支持HFBC压缩 “HI_COMPRESS_MODE_HFBC”。 默认值：HI_COMPRESS_MODE_HFBC
video_format	整型，解码图像数据格式。 当前仅支持“HI_VIDEO_FORMAT_TILE_64x16”和 “HI_VIDEO_FORMAT_LINEAR”。 默认值：HI_VIDEO_FORMAT_TILE_64x16
quick_mark_mode	整型，快速释放参考帧模式。预留参数，暂未支持。

2.10.8.3.13 hi_vdec_param_picture

说明

定义图像解码高级参数。

定义

```
hi_vdec_param_picture = {"pixel_format": hi\_pixel\_format, # RW; out put pixel format  
                        "u32Alpha": u32Alpha }
```

成员

成员名称	描述
pixel_format	整型，图片像素格式。保留参数。
u32Alpha	整型，RGB解码透明度。保留参数。

2.10.8.3.14 hi_vdec_chn_param

说明

定义解码通道高级参数。

定义

```
hi_vdec_chn_param = {"type": hi\_payload\_type , # RW; video type to be decoded  
                    "display_frame_num": display_frame_num, # RW, Range: [0, 16]; display frame num  
                    "video_param": hi\_vdec\_video\_param } # structure with video ( h265/h264/VP9)
```

```
hi_vdec_chn_param = {"type": hi_payload_type, # RW; video type to be decoded  
                    "display_frame_num": display_frame_num, # RW, Range: [0, 16]; display frame num  
                    "pic_param": hi_vdec_param_picture } # structure with picture (jpeg)
```

成员

成员名称	描述
type	整型，解码协议。
display_frame_num	整型，解码缓存图像的最小帧数。 取值范围：[0, 16]。 默认值：2。
video_param	字典，视频(H.264/H.265)解码高级参数。
pic_param	字典，图片(JPEG)解码高级参数。

2.10.8.3.15 hi_vdec_dec_err

说明

定义解码错误信息字典。

定义

```
hi_vdec_dec_err = {"set_pic_size_err": set_pic_size_err, # R; Picture width or height is larger than channel  
width or height.  
                  "set_protocol_num_err": set_protocol_num_err, # R; Protocol num is not enough. eg: slice, pps,  
sps.  
                  "set_ref_num_err": set_ref_num_err, # R; Reference num is not enough.  
                  "set_pic_buf_size_err": set_pic_buf_size_err, # R; The buffer size of picture is not enough.  
                  "format_err": format_err, # R; Format error. eg: do not support filed.  
                  "stream_unsupport": stream_unsupport, # R; Unsupported the stream specification.  
                  "pack_err": pack_err # R; Stream package error.  
                  "stream_size_over": stream_size_over, # R; the stream size is too big and force discard stream  
                  "stream_not_release": stream_not_release } # R; the stream not released for too long time
```

成员

成员	描述
set_pic_size_err	整型，图像的宽（或高）比通道的宽（或高）大，预留参数。
set_protocol_num_err	整型，设置的协议参数个数不足。比如：Slice/Pps/Sps个数。此参数仅对H.264/H.265解码有效。
set_ref_num_err	整型，设置的参考帧个数不足。此参数仅对H.264/H.265解码有效。
set_pic_buf_size_err	整型，图像Buffer内存大小不足。
format_err	整型，不支持的格式。JPEG的Progressive码流。

成员	描述
stream_unsupport	整型，不支持的规格（码流规格与昇腾AI处理器宣称支持的规格不一致）。
pack_err	整型，码流存在错误。
stream_size_over	整型，在一帧中的码流过大，当整个SCDbuffer都装不下一帧码流时，强制清空SCDbuffer。 此参数仅对H.264/H.265解码有效。
stream_not_release	整型，VFMW（video firmware）内部管理码流错误，出现长时间不释放码流的情况，预留参数。

2.10.8.3.16 hi_vdec_chn_status

说明

定义通道状态字典。

定义

```
hi_vdec_chn_status = {"type": hi_payload_type, # R; video type to be decoded
    "left_stream_bytes": left_stream_bytes, # R; left stream bytes waiting for decode
    "left_stream_frames": left_stream_frames, # R; left frames waiting for decode,only valid for
HI_VDEC_SEND_MODE_FRAME
    "left_decoded_frames": left_decoded_frames, # R; pics waiting for output.
    "is_started": is_started, # R; had started rcv stream?
    "rcv_stream_frames": rcv_stream_frames, # R; how many frames of stream has been
received. valid when send by frame.
    "dec_stream_frames": dec_stream_frames, # R; how many frames of stream has been
decoded. valid when send by frame.
    "dec_err": hi_vdec_dec_err, # R; information about decode error.
    "width": width,
    "height": height,
    "latest_frame_pts": latest_frame_pts }
```

成员

成员名称	描述
type	整型，解码协议。
left_stream_bytes	整型，码流Buffer中待解码的Byte数，包括正在解码的当前帧中未解码的Byte数。
left_stream_frames	整型，码流Buffer中待解码的帧数，不包括正在解码的当前帧。-1表示无效。仅按帧发送时有效。
left_decoded_frames	整型，图像Buffer中剩余的图片数目。
is_started	整型，解码器是否已经启动接收码流。
rcv_stream_frames	整型，码流Buffer中已接收码流帧数。-1表示无效。仅按帧发送时有效。

成员名称	描述
dec_stream_frames	整型，码流Buffer中已解码帧数。仅按帧发送时有效。
dec_err	字典，解码错误信息。
width	整型，解码后输出图片的宽。
height	整型，解码后输出图片的高。
latest_frame_pts	最新解码图像的时间戳。预留参数，暂不支持。

2.10.8.3.17 hi_vdec_stream

说明

定义视频/图像解码的码流字典。

定义

```
hi_vdec_stream = {"end_of_frame": end_of_frame, # W; is the end of a frame.  
                  "end_of_stream": end_of_stream, # W; is the end of all stream.  
                  "need_display": need_display, # W; is the current frame displayed. only valid by  
HI_VDEC_SEND_MODE_FRAME.  
                  "pts": pts, # W; time stamp  
                  "private_data": private_data, # W; private data,reserved  
                  "len": len, # W; length of stream  
                  "addr": addr} # W; stream address
```

成员

成员	描述
end_of_frame	整型，当前帧是否结束（此值目前暂无效）。
end_of_stream	整型，是否发完所有码流。
need_display	整型，当前帧是否输出显示。 0：不显示，设置为0，输出Buffer无解码结果，且不显示帧按照显示序或解码序输出。 1：显示，设置为1，才能输出解码结果，且显示帧按照显示序或解码序输出。 但如果解码码流中同时存在不显示帧、显示帧，则系统不保证不显示帧与显示帧之间按照显示序或解码序输出。
pts	整型，码流包的时间戳，以us为单位。仅按帧发送时有效。
private_data	私有数据。预留参数，暂不支持。
len	整型，码流包的长度，以Byte为单位。
addr	整型，码流包的地址。

注意事项

- 按帧发送时，解码图像的时间戳等于码流包中的时间戳。
- 按流发送时，解码图像的时间戳等于 0。
- 当发完所有码流后，把“end_of_stream”设置为1，表示码流文件结束，这时解码器会解完发送下来的所有码流并输出所有图像。如果发完所有码流后把“end_of_stream”设置为0，解码器内部可能残余大于等于一帧的图像未解码输出，因为解码器必须等到下一帧码流到来才能知道当前帧已经结束，送入解码。
- VDEC支持发送一包“end_of_stream”为1的空码流包（地址为空或长度为 0）。

2.10.8.3.18 hi_vdec_supplement_info

说明

定义输出帧补充信息，暂不支持。

定义

```
hi_vdec_supplement_info = {"type": hi_payload_type, # RW; Video type to be decoded.  
                           "video_supplement_info": hi_vdec_video_supplement_info} # Structure with video (h265/  
h264)
```

2.10.8.3.19 hi_vdec_video_supplement_info

说明

定义输出视频帧补充信息，暂不支持。

定义

```
hi_vdec_video_supplement_info = {"frame_type": hi_vdec_frame_type,  
                                 "err_rate": err_rate,  
                                 "poc": poc}
```

成员

成员名称	描述
frame_type	整型，解码帧类型。预留参数，暂未支持。
err_rate	整型，错误率。预留参数，暂未支持。
poc	整型，poc值。预留参数，暂未支持。

2.10.8.3.20 hi_jpegd_precision_mode

说明

定义JPEGD解码输出图片的宽、高对齐模式。

定义

YUVOUT_ALIGN_DOWN = 0
YUVOUT_ALIGN_UP = 1
YUVOUT_ALIGN_DOWN_COMPAT = 2

成员名称	描述
YUVOUT_ALIGN_DOWN	默认值。 - 解码后的输出图片格式为YUV420SP时，若源图的宽、高为奇数时，将输出图片的宽、高向下2对齐。 - 解码后的输出图片格式为YUV422SP时，若源图的宽为奇数时，将输出图片的宽向下2对齐。 - 解码后的输出图片格式为YUV440SP时，若源图的高为奇数时，将输出图片高向下2对齐。
YUVOUT_ALIGN_UP	- 解码后的输出图片格式为YUV420SP时，若源图的宽、高为奇数时，将输出图片的宽、高向上2对齐。 - 解码后的输出图片格式为YUV422SP时，若源图的宽为奇数时，将输出图片的宽向上2对齐。 - 解码后的输出图片格式为YUV440SP时，若源图的高为奇数时，将输出图片高向上2对齐。
YUVOUT_ALIGN_DOWN_COMPAT	兼容旧版本，解码后的输出图片格式为YUV420SP/YUV422SP/YUV440SP时，若源图的宽、高为奇数时，将输出图片的宽、高向下2对齐。

2.10.8.4 视频/图像编码

2.10.8.4.1 hi_venc_stream

说明

定义帧码流类型字典。

定义

```
hi_venc_stream = {"pack": hi_venc_pack, # R; stream pack attribute
                  "pack_cnt": pack_cnt, # R; the pack number of one frame stream
                  "seq": seq} # R; the list number of stream

hi_venc_stream = {"pack": hi_venc_pack, # R; stream pack attribute
                  "pack_cnt": pack_cnt, # R; the pack number of one frame stream
                  "seq": seq, # R; the list number of stream
                  "h264_info": hi_venc_h264_stream_info, # R; the stream info of h264
                  "h264_adv_info": hi_venc_h264_adv_stream_info} # R; the stream info of h264

hi_venc_stream = {"pack": hi_venc_pack, # R; stream pack attribute
                  "pack_cnt": pack_cnt, # R; the pack number of one frame stream
                  "seq": seq, # R; the list number of stream
                  "h264_info": hi_venc_h264_stream_info} # R; the stream info of h264

hi_venc_stream = {"pack": hi_venc_pack, # R; stream pack attribute
                  "pack_cnt": pack_cnt, # R; the pack number of one frame stream
                  "seq": seq, # R; the list number of stream
```

```
"jpeg_info": hi_venc_jpeg_stream_info}

hi_venc_stream = {"pack": hi_venc_pack, # R; stream pack attribute
                  "pack_cnt": pack_cnt, # R; the pack number of one frame stream
                  "seq": seq, # R; the list number of stream
                  "h265_info": hi_venc_h265_stream_info} # R; the stream info of h265

hi_venc_stream = {"pack": hi_venc_pack, # R; stream pack attribute
                  "pack_cnt": pack_cnt, # R; the pack number of one frame stream
                  "seq": seq, # R; the list number of stream
                  "h265_info": hi_venc_h265_stream_info, # R; the stream info of h265
                  "h265_adv_info": hi_venc_h265_adv_stream_info} # R; the stream info of h265

hi_venc_stream = {"pack": hi_venc_pack, # R; stream pack attribute
                  "pack_cnt": pack_cnt, # R; the pack number of one frame stream
                  "seq": seq, # R; the list number of stream
                  "prores_info": hi_venc_prores_stream_info}
```

成员

成员名称	描述
pack	整型，帧码流包结构。
pack_cnt	整型，一帧码流的所有包的个数。
seq	整型，码流序列号。 按帧获取时，此处为帧序号；按包获取时，此处为包序号。
h264_info/h265_info	字典，码流特征信息。 Atlas 推理系列产品 不支持该参数。 Atlas A2训练系列产品/Atlas 800I A2推理产品 不支持该参数。
jpeg_info/prores_info	码流特征信息，暂不支持。
h264_adv_info/ h265_adv_info	字典，码流高级特征信息。 Atlas 推理系列产品 不支持该参数。 Atlas A2训练系列产品/Atlas 800I A2推理产品 不支持该参数。

2.10.8.4.2 hi_venc_pack

说明

定义帧码流包字典。

定义

```
hi_venc_pack = {"phys_addr": phys_addr,
                "addr": addr,
                "len": len,
                "pts": pts,
                "is_frame_end": is_frame_end,
                "data_type": hi_venc_data_type,
                "offset": offset,
```

```
        "data_num": data_num
        "pack_info": [hi_venc_pack_info]]

hi_venc_pack = {"input_addr": input_addr,
               "addr": addr,
               "len": len,
               "pts": pts,
               "is_frame_end": is_frame_end,
               "data_type": hi_venc_data_type,
               "offset": offset,
               "data_num": data_num
               "pack_info": [hi_venc_pack_info]}
```

成员

成员名称	描述
addr	整型，码流包首地址。
phys_addr	整型，码流包物理地址。
input_addr	整型，用户输入码流数据的内存地址。
len	整型，码流包长度。 当len = 0，表示视频/图像编码失败。
data_type	整型，码流类型，支持 H.265/H.264/JPEG协议类型的数据包。
pts	整型，时间戳。单位：us。
is_frame_end	整型，帧结束标识。 取值范围： - HI_TRUE：该码流包是该帧的最后一个包。 - HI_FALSE：该码流包不是该帧的最后一个包。
offset	整型，码流包中有效数据与码流包首地址“addr”的偏移。
data_num	整型，当前码流包（当前包的类型由“data_type”指定）数据中包含码流包的个数。
pack_info	列表，当前码流包数据中包含码流包数据信息。

2.10.8.4.3 hi_venc_chn_attr

说明

定义编码通道属性字典。

定义

```
hi_venc_chn_attr = {"venc_attr": hi_venc_attr, "rc_attr": hi_venc_rc_attr, "gop_attr": hi_venc_gop_attr}
```

成员

成员名称	描述
venc_attr	字典，编码器属性。 编码器属性中除通道宽高（“pic_width”和“pic_height”）外都是静态属性，一旦创建编码通道成功，静态属性不支持被修改，除非该通道被销毁，重新创建。
rc_attr	字典，码率控制器属性。 码率控制器属性首先需要配置RC（Rate Control）模式，JPEG不需要配置码率控制器属性，其它协议类型通道（H.264/H.265）都必须配置。码率控制器属性RC模式必须与编码器属性协议类型匹配。
gop_attr	字典，Gop Mode类型的字典。 对于H.264、H.265协议，需要配置该参数。

2.10.8.4.4 hi_venc_attr

说明

定义编码器属性字典。

定义

```
hi_venc_attr = {"type": hi_payload_type,
               "max_pic_width": max_pic_width,
               "max_pic_height": max_pic_height,
               "buf_size": buf_size,
               "profile": profile,
               "is_by_frame": is_by_frame,
               "pic_width": pic_width,
               "pic_height": pic_height,
               "h264_attr":hi_venc_h264_attr} # attributes of H264e

hi_venc_attr = {"type": type,
               "max_pic_width": max_pic_width,
               "max_pic_height": max_pic_height,
               "buf_size": buf_size,
               "profile": profile,
               "is_by_frame": is_by_frame,
               "pic_width": pic_width,
               "pic_height": pic_height,
               "h265_attr":hi_venc_h265_attr} # attributes of H265e

hi_venc_attr = {"type": hi_payload_type,
               "max_pic_width": max_pic_width,
               "max_pic_height": max_pic_height,
               "buf_size": buf_size,
               "profile": profile,
               "is_by_frame": is_by_frame,
               "pic_width": pic_width,
               "pic_height": pic_height,
               "jpeg_attr":hi_venc_jpeg_attr} # attributes of jpeg

hi_venc_attr = {"type": hi_payload_type,
               "max_pic_width": max_pic_width,
```

```
"max_pic_height": max_pic_height,
"buf_size": buf_size,
"profile": profile,
"is_by_frame": is_by_frame,
"pic_width": pic_width,
"pic_height": pic_height,
"prores_attr": hi\_venc\_prores\_attr} # attributes of prores
```

成员

成员名称	描述
type	整型，编码协议类型。
max_pic_width	整型，编码图像最大宽度，静态属性，必须是2的整数倍。 取值范围：[MIN_WIDTH, MAX_WIDTH]，以像素为单位。“MIN_WIDTH”与“MAX_WIDTH”，分别表示编码通道支持的最小宽度、最大宽度，参见 表2-65 。
max_pic_height	整型，除JPEG协议外，编码图像最大高度，静态属性，必须是2的整数倍。 取值范围：[MIN_HEIGHT, MAX_HEIGHT]，以像素为单位。“MIN_HEIGHT”、“MAX_HEIGHT”分别表示编码通道支持的最小高度、最大高度，参见 表2-65 。

成员名称	描述
buf_size	整型，码流Buffer大小，单位为Byte，静态属性。 - 调用acl.himpi.venc_send_jpege_frame接口发送原始图像进行图像编码时，在创建通道时，“buf_size”参数值必须设置为0。 - 调用acl.himpi.venc_send_frame接口发送原始图像进行视频或图像编码时，在创建通道时，“buf_size”参数值必须设置为64的整数倍，取值范围：[Min, Max]，以Byte为单位。 视频编码场景下，Min当前表示$32 * 1024$，Max当前表示$1 * 1024 * 1024 * 1024$。 用户可以在创建通道时，合理设置每个通道的“buf_size”，以节省内存使用开销，推荐将“buf_size”设置为：$原图宽 * 原图高 * 3/2$后再64对齐，但“buf_size”依然需要在[Min, Max]范围内。 视频编码场景下，若“buf_size”设置的不合理，可能出现以下异常情况： - 反复重编，进而导致编码时延变长、性能下降、图像质量下降。 - 编码失败，获取不到编码结果。 图像编码场景下，如果原图格式为YUV420，Min值为$原图宽16对齐 * 原图高16对齐 * 3/2$，如果原图格式为YUV422 Packed，Min值为$原图宽16对齐 * 原图高16对齐 * 2$。Max当前表示$1 * 1024 * 1024 * 1024$。 用户可以在创建通道时根据业务场景中图片的实际最大分辨率，合理设置每个通道的“buf_size”，以节省内存使用开销，推荐将“buf_size”设置为：$原图宽 * 原图高 * 倍数$，倍数推荐为5。 例如：用户图片的分辨率有720p ($1280 * 720$)、1080p ($1080 * 1920$)、4K ($3840 * 2160$) 大小的图片，需要的“buf_size”应按照4k配置：$3840 * 2160 * 5$。 图片编码场景下，“buf_size”最小值为$320 * 240 * 5$，防止小分辨率场景下，Buffer容易被占满，影响编码功能执行，例如：用户图片的最大分辨率为$128 * 128$，推荐用户将“buf_size”设置为$320 * 240 * 5$。

成员名称	描述
profile	整型，编码的等级，静态属性。 ● H.264 取值范围：[0, 3]。 – 0：Baseline。当取值为该值时，不支持编码B帧。 – 1：Main Profile。 – 2：High Profile。 ● H.265 取值范围：[0, 1]。 – 0：Main Profile。 – 1：Main 10 Profile，预留给，暂不支持。 ● JPEG取值范围：0（表示Baseline）。
is_by_frame	整型，帧/包模式获取码流，静态属性。 ● HI_TRUE：按帧获取。 ● HI_FALSE：按包获取。
pic_width	编码通道宽度，必须是“MIN_ALIGN”的整数倍。通道创建后，VENC视频编码支持修改通道宽度，JPEG图片编码不支持修改通道宽度。 取值范围：[MIN_WIDTH, MAX_WIDTH]，以像素为单位。“MIN_WIDTH”、“MAX_WIDTH”、“MIN_ALIGN”分别表示编码通道支持的最小宽度、最大宽度、最小对齐单元（像素），参见表2-65。 JPEG通道宽高的设置要满足：$\text{pic\_width} * \text{pic\_height} \leq \text{max\_pic\_width} * \text{max\_pic\_height}$。
pic_height	除JPEG协议外，编码通道高度，必须是“MIN_ALIGN”的整数倍。通道创建后VENC视频编码支持修改通道高度，JPEG图片编码不支持修改通道高度。 取值范围：[MIN_HEIGHT, MAX_HEIGHT]，以像素为单位。“MIN_HEIGHT”、“MAX_HEIGHT”、“MIN_ALIGN”分别表示编码通道支持的最小高度、最大高度、最小对齐单元（像素），参见表2-65。 JPEG通道宽高的设置要满足：$\text{pic\_width} * \text{pic\_height} \leq \text{max\_pic\_width} * \text{max\_pic\_height}$。
h264_attr	编码协议的属性。 Atlas 推理系列产品，预留参数，暂不支持。 Atlas A2训练系列产品/Atlas 800I A2推理产品，预留参数，暂不支持。
h265_attr	编码协议的属性。 Atlas 推理系列产品，预留参数，暂不支持。 Atlas A2训练系列产品/Atlas 800I A2推理产品，预留参数，暂不支持。
jpeg_attr	编码协议的属性。 预留参数，暂不支持。

成员名称	描述
prores_attr	编码协议的属性。 预留参数，暂不支持。

参考信息

MIN_WIDTH、MAX_WIDTH、MIN_HEIGHT、MAX_HEIGHT、MIN_ALIGN分别表示编码通道支持的最小宽度、最大宽度、最小高度、最大高度、最小对齐单元（像素）。

表 2-65 编码通道宽高

W/H (pixel)	H.264	H.265	JPEG
MIN_WIDTH	128	128	32
MAX_WIDTH	4096	4096	8192
MIN_HEIGHT	128	128	32
MAX_HEIGHT	4096	4096	8192
MIN_ALIGN	2	2	2

2.10.8.4.5 hi_venc_rc_attr

说明

定义编码通道码率控制器属性。

定义

```
hi_venc_rc_attr = {"rc_mode": rc_mode, # RW; the type of rc  
                  "h264_cbr": hi_venc_h264_cbr}  
  
hi_venc_rc_attr = {"rc_mode": rc_mode, # RW; the type of rc  
                  "h264_vbr": hi_venc_h264_vbr}  
  
hi_venc_rc_attr = {"rc_mode": rc_mode, # RW; the type of rc  
                  "h264_avbr": hi_venc_h264_avbr}  
  
hi_venc_rc_attr = {"rc_mode": rc_mode, # RW; the type of rc  
                  "h264_qvbr": hi_venc_h264_qvbr}  
  
hi_venc_rc_attr = {"rc_mode": rc_mode, # RW; the type of rc  
                  "h264_cvbr": hi_venc_h264_cvbr}  
  
hi_venc_rc_attr = {"rc_mode": rc_mode, # RW; the type of rc  
                  "h264_fixqp": hi_venc_h264_fixqp}  
  
hi_venc_rc_attr = {"rc_mode": rc_mode, # RW; the type of rc  
                  "h264_qpmap": hi_venc_h264_qpmap}  
  
hi_venc_rc_attr = {"rc_mode": rc_mode, # RW; the type of rc  
                  "mjpeg_cbr": hi_venc_mjpeg_cbr}
```

```
hi_venc_rc_attr = {"rc_mode": rc_mode, # RW; the type of rc
                  "mjpeg_vbr": hi\_venc\_mjpeg\_vbr}

hi_venc_rc_attr = {"rc_mode": rc_mode, # RW; the type of rc
                  "mjpeg_fixqp": hi\_venc\_mjpeg\_fixqp}

hi_venc_rc_attr = {"rc_mode": rc_mode, # RW; the type of rc
                  "h265_cbr": hi\_venc\_h265\_cbr}

hi_venc_rc_attr = {"rc_mode": rc_mode, # RW; the type of rc
                  "h265_vbr": hi\_venc\_h265\_vbr}

hi_venc_rc_attr = {"rc_mode": rc_mode, # RW; the type of rc
                  "h265_avbr": hi\_venc\_h265\_avbr}

hi_venc_rc_attr = {"rc_mode": rc_mode, # RW; the type of rc
                  "h265_qvbr": hi\_venc\_h265\_qvbr}

hi_venc_rc_attr = {"rc_mode": rc_mode, # RW; the type of rc
                  "h265_cvbr": hi\_venc\_h265\_cvbr}

hi_venc_rc_attr = {"rc_mode": rc_mode, # RW; the type of rc
                  "h265_fixqp": hi\_venc\_h265\_fixqp}

hi_venc_rc_attr = {"rc_mode": rc_mode, # RW; the type of rc
                  "h265_qpmap": hi\_venc\_h265\_qpmap}
```

成员

成员名称	描述
rc_mode	RC模式。 • CBR (Constant Bit Rate) 固定比特率。即在码率统计时间内保证编码码率平稳。 • VBR (Variable Bit Rate) 可变比特率，即允许在码率统计时间内编码码率波动，从而保证编码图像质量平稳。 • AVBR (Adaptive Variable Bit Rate) 自适应可变比特率，即允许在码率统计时间内编码码率波动，从而保证编码图像质量平稳。码率控制内部会检测当前场景的运动静止状态，在运动时用较高码率编码，在静止时主动降低目标码率。 • QVBR (Quality Variable Bit Rate) 基于主观图像质量的可变比特率，该码控调节方式是利用实时统计的PSNR (图像质量客观评价指标) 的大小来动态调整码率，从而保证编码图像质量平稳。在PSNR较小时主动升高目标码率，PSNR较大时主动降低目标码率。 • CVBR (Constrained Variable Bit Rate) 是以VBR为基础，旨在提供平稳的图像质量的码控算法，同时对VBR的码率进行限制，以满足传输带宽以及存储空间的要求。具体来说，CVBR设置了瞬时、短期与长期码率的限制。其中，瞬时码率的限制保证了网络带宽对传输的要求；长期码率限制保证了在长时间视频录制时，存储设备有足够的空间储存数据；同时，短期码率会根据长期码率的设置和实际使用情况进行调节，以在场景复杂的情况下提供更加平稳的图像质量，并在场景简单时节省码率。 • FIXQP (Fix Quantisation Parameter) 固定QP值。在码率统计时间内，编码图像所有宏块QP值相同，采用用户设定的图像QP值，I帧和P帧的QP值可以分别设置。 对于不同协议，相同 RC 模式的属性变量基本一致。表 2-66介绍了几种RC模式的公共的属性。
h264_cbr	字典，H.264协议编码通道CBR模式属性。
h264_vbr	字典，H.264协议编码通道VBR模式属性。
h264_avbr	字典，H.264协议编码通道AVBR模式属性。
h264_qvbr	字典，H.264协议编码通道QVBR模式属性。
h264_cvbr	字典，H.264协议编码通道CVBR模式属性。
h264_fixqp	字典，H.264协议编码通道FIXQP模式属性。 Atlas 推理系列产品，预留参数，暂不支持。 Atlas A2训练系列产品/Atlas 800I A2推理产品，预留参数，暂不支持。

成员名称	描述
h264_qpmap	字典，H.264协议编码通道QPMap模式属性。预留参数，暂不支持。
mjpeg_fixqp	字典，MJPEG（Motion Joint Photographic Experts Group）协议编码通道FIXQP模式属性。预留参数，暂不支持。
mjpeg_cbr	字典，MJPEG协议编码通道CBR模式属性。预留参数，暂不支持。
mjpeg_vbr	字典，MJPEG协议编码通道VBR模式属性。预留参数，暂不支持。
h265_cbr	字典，H.265协议编码通道CBR模式属性。
h265_vbr	字典，H.265协议编码通道VBR模式属性。
h265_avbr	字典，H.265协议编码通道AVBR模式属性。
h265_qvbr	字典，H.265协议编码通道QVBR模式属性。
h265_cvbr	字典，H.265协议编码通道CVBR模式属性。
h265_fixqp	字典，H.265协议编码通道FIXQP模式属性。 Atlas 推理系列产品 ，预留参数，暂不支持。 Atlas A2训练系列产品/Atlas 800I A2推理产品 ，预留参数，暂不支持。
h265_qpmap	字典，H.265协议编码通道QPMap模式属性。预留参数，暂不支持。

参考信息

表 2-66 码率控制器属性的约束

RC 模式	GOP	StatTime(s)	FrmRate(src_frame_rate/ dst_frame_rate)
CBR	≥ 1	≥ 1	src_frame_rate ≥ dst_frame_rate
VBR	≥ 1	≥ 1	src_frame_rate ≥ dst_frame_rate
AVBR	≥ 1	≥ 1	src_frame_rate ≥ dst_frame_rate
FIXQP	≥ 1	≥ 1	src_frame_rate ≥ dst_frame_rate
QPMap	≥ 1	≥ 1	src_frame_rate ≥ dst_frame_rate

2.10.8.4.6 hi_venc_rc_mode

说明

定义编码通道码率控制器模式。

定义

```
HI_VENC_RC_MODE_H264_CBR    = 1
HI_VENC_RC_MODE_H264_VBR    = 2
HI_VENC_RC_MODE_H264_AVBR   = 3
HI_VENC_RC_MODE_H264_QVBR   = 4
HI_VENC_RC_MODE_H264_CVBR   = 5
HI_VENC_RC_MODE_H264_FIXQP  = 6
HI_VENC_RC_MODE_H264_QPMAP  = 7
HI_VENC_RC_MODE_MJPEG_CBR   = 8
HI_VENC_RC_MODE_MJPEG_VBR   = 9
HI_VENC_RC_MODE_MJPEG_FIXQP = 10
HI_VENC_RC_MODE_H265_CBR    = 11
HI_VENC_RC_MODE_H265_VBR    = 12
HI_VENC_RC_MODE_H265_AVBR   = 13
HI_VENC_RC_MODE_H265_QVBR   = 14
HI_VENC_RC_MODE_H265_CVBR   = 15
HI_VENC_RC_MODE_H265_FIXQP  = 16
HI_VENC_RC_MODE_H265_QPMAP  = 17
HI_VENC_RC_MODE_BUTT        = 18
```

成员

成员名称	描述
HI_VENC_RC_MODE_H264_CBR	H.264 CBR模式。
HI_VENC_RC_MODE_H264_VBR	H.264 VBR模式。
HI_VENC_RC_MODE_H264_AVBR	H.264 AVBR模式。
HI_VENC_RC_MODE_H264_QVBR	H.264 QVBR模式。
HI_VENC_RC_MODE_H264_CVBR	H.264 CVBR模式。
HI_VENC_RC_MODE_H264_FIXQP	H.264 FIXQP模式。 Atlas 推理系列产品 ，预留参数，暂不支持。 Atlas A2训练系列产品/Atlas 800I A2推理产品 ，预留参数，暂不支持。
HI_VENC_RC_MODE_H264_QPMAP	H.264 QPMAP模式。预留参数，暂不支持。
HI_VENC_RC_MODE_MJPEG_CBR	MJPEG CBR模式。预留参数，暂不支持。
HI_VENC_RC_MODE_MJPEG_VBR	MJPEG VBR模式。预留参数，暂不支持。

成员名称	描述
HI_VENC_RC_MODE_MJPEG_FIXQP	MJPEG FIXQP模式。预留参数，暂不支持。
HI_VENC_RC_MODE_H265_CBR	H.265 CBR模式。
HI_VENC_RC_MODE_H265_VBR	H.265 VBR模式。
HI_VENC_RC_MODE_H265_AVBR	H.265 AVBR模式。
HI_VENC_RC_MODE_H265_QVBR	H.265 QVBR模式。
HI_VENC_RC_MODE_H265_CVBR	H.265 CVBR模式。
HI_VENC_RC_MODE_H265_FIXQP	H.265 FIXQP模式。 Atlas 推理系列产品 ，预留参数，暂不支持。 Atlas A2训练系列产品/Atlas 800I A2推理产品 ，预留参数，暂不支持。
HI_VENC_RC_MODE_H265_QPMAP	H.265 QPMAP模式。预留参数，暂不支持。

2.10.8.4.7 hi_venc_h264_cbr

说明

定义H.264编码通道CBR属性字典。

定义

```
hi_venc_h264_cbr = {"gop": gop,
                    "stats_time": stats_time,
                    "src_frame_rate": src_frame_rate,
                    "dst_frame_rate": dst_frame_rate,
                    "bit_rate": bit_rate }
```


成员

成员名称	描述
<code>gop</code>	整型，Group Of Pictures，视频序列由若干时间连续的图像构成，在对其进行编码时，先将该视频序列分割成若干小的图像组，该参数即指一个图像组的帧数。 由于GOP结构仅支持Normalp（第一帧为I帧，其余帧为P帧），此参数也为I帧间隔，默认值65535。 由于编码I帧不需参考之前的帧，之前帧的编码质量影响从此帧开始不再传递，故该值越小，编码质量越好，但不建议小于帧率，因为I帧仅有帧内预测块，大部分场景需要有一定的P帧（有帧间预测块）才能保证编码质量。 取值范围：[1, 65536]。
<code>stats_time</code>	整型，CBR码率统计时间，以秒为单位，一般场景下，$stats\_time = gop / dst\_frame\_rate$，编码质量较好。 码率统计时间越长，每帧图像的码率波动对于码率调节的影响越弱，码率的调节会更缓慢，图像质量的波动会更轻微；码率统计时间越短，每帧图像的码率波动对于码率调节的影响越强，图像码率的调节会更灵敏，图像质量的波动会更剧烈。 取值范围：[1, 60]。
<code>src_frame_rate</code>	整型，输入帧率，以fps为单位，即每秒进入编码器的帧数，默认值为30；码率一定的情况下，该值越小，编码质量越好，但小于25播放时画面会不连贯。 该参数取值范围：[1, 240]。
<code>dst_frame_rate</code>	整型，编码器输出帧率，以fps为单位，即每秒编码器输出的帧数，默认值为30；码率一定的情况下，该值越小，编码质量越好，但小于25播放时画面会不连贯。 取值范围：[1, <code>src_frame_rate</code>]。 暂不支持分数帧率。
<code>bit_rate</code>	整型，编码器输出平均码率，以kbps为单位，即编码码流每秒比特数，该值越大，编码质量越好；默认值2000。 该参数取值范围：[2, 614400]。 平均比特率的设置与编码通道宽高以及图像帧率都有关系。典型的平均比特率的设置如表2-67所示。注意，表2-67中的平均比特率的设置是在通道编码帧率为满帧率（30fps）时的设置。当用户设置编码输出帧率不为满帧率时，可以对下表中的码率按用户设置帧率与满帧率（30fps）的比例进行换算。

表 2-67 典型平均比特率配置

图像宽高/码率等级	1080p(1920 x 1080)	4K(3840 x 2160)
低码率	< 2Mbps	< 8Mbps
中码率	2Mbps ~ 8Mbps	8Mbps ~ 30Mbps

图像宽高/码率等级	1080p(1920 x 1080)	4K(3840 x 2160)
高码率	> 8Mbps	> 30Mbps

2.10.8.4.8 hi_venc_h264_vbr

说明

定义H.264编码通道VBR属性字典。

定义

```
hi_venc_h264_vbr = {"gop": gop,  
                    "stats_time": stats_time,  
                    "src_frame_rate": src_frame_rate,  
                    "dst_frame_rate": dst_frame_rate,  
                    "max_bit_rate": max_bit_rate}
```

成员

成员名称	描述
gop	整型，Group Of Pictures，视频序列由若干时间连续的图像构成，在对其进行编码时，先将该视频序列分割成若干个小的图像组，该参数即指一个图像组的帧数。 由于GOP结构仅支持Normalp（第一帧为I帧，其余帧为P帧），此参数也为I帧间隔，默认值65535。 由于编码I帧不需参考之前的帧，之前帧的编码质量影响从此帧开始不再传递，故该值越小，编码质量越好，但不建议小于帧率，因为I帧仅有帧内预测块，大部分场景需要有一定的P帧（有帧间预测块）才能保证编码质量。 取值范围：[1, 65536]。
stats_time	整型，VBR码率统计时间，以秒为单位，一般场景下，stats_time = gop / dst_frame_rate，编码质量较好。 码率统计时间越长，每帧图像的码率波动对于码率调节的影响越弱，码率的调节会更缓慢，图像质量的波动会更轻微；码率统计时间越短，每帧图像的码率波动对于码率调节的影响越强，图像码率的调节会更灵敏，图像质量的波动会更剧烈。 取值范围：[1, 60]。
src_frame_rate	整型，输入帧率，以fps为单位，即每秒进入编码器的帧数，默认值为30；码率一定的情况下，该值越小，编码质量越好，但小于25播放时画面会不连贯。 该参数取值范围：[1, 240]。

成员名称	描述
dst_frame_rate	整型，编码器输出帧率，以fps为单位，即每秒编码器输出的帧数，默认值为30；码率一定的情况下，该值越小，编码质量越好，但小于25播放时画面会不连贯。 取值范围：[1, src_frame_rate]。 暂不支持分数帧率。
max_bit_rate	整型，编码器输出最大码率，以kbps为单位，即编码码流每秒比特数，该值越大，编码质量越好；默认值2000。 该参数取值范围：[2, 614400]。

2.10.8.4.9 hi_venc_h264_avbr

说明

定义 H.264 编码通道AVBR属性字典。

定义

```
hi_venc_h264_avbr = {"gop": gop,  
                    "stats_time": stats_time,  
                    "src_frame_rate": src_frame_rate,  
                    "dst_frame_rate": dst_frame_rate,  
                    "max_bit_rate": max_bit_rate}
```

成员

成员名称	描述
gop	整型，Group Of Pictures，视频序列由若干时间连续的图像构成，在对其进行编码时，先将该视频序列分割成若干个小的图像组，该参数即指一个图像组的帧数。 由于GOP结构仅支持Normalp（第一帧为I帧，其余帧为P帧），此参数也为I帧间隔，默认值65535。 由于编码I帧不需参考之前的帧，之前帧的编码质量影响从此帧开始不再传递，故该值越小，编码质量越好，但不建议小于帧率，因为I帧仅有帧内预测块，大部分场景需要有一定的P帧（有帧间预测块）才能保证编码质量。 取值范围：[1, 65536]。
stats_time	整型，AVBR码率统计时间，以秒为单位，一般场景下，stats_time = gop / dst_frame_rate，编码质量较好。 码率统计时间越长，每帧图像的码率波动对于码率调节的影响越弱，码率的调节会更缓慢，图像质量的波动会更轻微；码率统计时间越短，每帧图像的码率波动对于码率调节的影响越强，图像码率的调节会更灵敏，图像质量的波动会更剧烈。 取值范围：[1, 60]。

成员名称	描述
src_frame_rate	整型，输入帧率，以fps为单位，即每秒进入编码器的帧数，默认值为30；码率一定的情况下，该值越小，编码质量越好，但小于25播放时画面会不连贯。 取值范围：[1, 240]。
dst_frame_rate	整型，编码器输出帧率，以fps为单位，即每秒编码器输出的帧数，默认值为30；码率一定的情况下，该值越小，编码质量越好，但小于25播放时画面会不连贯。 取值范围：[1, src_frame_rate]。 暂不支持分数帧率。
max_bit_rate	整型，编码器输出最大码率，以kbps为单位，即编码码流每秒比特数，该值越大，编码质量越好；默认值2000。 取值范围：[2, 614400]

2.10.8.4.10 hi_venc_h264_qvbr

说明

定义H.264协议编码通道QVBR模式属性字典。

定义

```
hi_venc_h264_qvbr = {"gop": gop,
                    "stats_time": stats_time,
                    "src_frame_rate": src_frame_rate,
                    "dst_frame_rate": dst_frame_rate,
                    "target_bit_rate": target_bit_rate}
```

成员

成员名称	描述
gop	整型，Group Of Pictures，视频序列由若干时间连续的图像构成，在对其进行编码时，先将该视频序列分割成若干个小的图像组，该参数即指一个图像组的帧数。 由于GOP结构仅支持Normalp（第一帧为I帧，其余帧为P帧），此参数也为I帧间隔，默认值65535。 由于编码I帧不需参考之前的帧，之前帧的编码质量影响从此帧开始不再传递，故该值越小，编码质量越好，但不建议小于帧率，因为I帧仅有帧内预测块，大部分场景需要有一定的P帧（有帧间预测块）才能保证编码质量。 取值范围：[1, 65536]。

成员名称	描述
stats_time	整型，QVBR码率统计时间，以秒为单位，一般场景下，stats_time= gop / dst_frame_rate，编码质量较好。 码率统计时间越长，每帧图像的码率波动对于码率调节的影响越弱，码率的调节会更缓慢，图像质量的波动会更轻微；码率统计时间越短，每帧图像的码率波动对于码率调节的影响越强，图像码率的调节会更灵敏，图像质量的波动会更剧烈。 取值范围：[1, 60]。
src_frame_rate	整型，输入帧率，以fps为单位，即每秒进入编码器的帧数，默认值为30；码率一定的情况下，该值越小，编码质量越好，但小于25播放时画面会不连贯。 取值范围：[1, 240]。
dst_frame_rate	整型，编码器输出帧率，以fps为单位，即每秒编码器输出的帧数，默认值为30；码率一定的情况下，该值越小，编码质量越好，但小于25播放时画面会不连贯。 取值范围：[1, src_frame_rate]。 暂不支持分数帧率。
target_bit_rate	整型，编码器输出最大码率，以kbps为单位，即编码码流每秒比特数，该值越大，编码质量越好；默认值2000。 取值范围：[2, 614400]

2.10.8.4.11 hi_venc_h264_cvbr

说明

定义H.264协议编码通道CVBR模式属性字典。

定义

```
hi_venc_h264_cvbr = {"gop": gop,
    "stats_time": stats_time,
    "src_frame_rate": src_frame_rate,
    "dst_frame_rate": dst_frame_rate,
    "max_bit_rate": max_bit_rate,
    "short_term_stats_time": short_term_stats_time,
    "long_term_stats_time": long_term_stats_time,
    "long_term_max_bit_rate": long_term_max_bit_rate,
    "long_term_min_bit_rate": long_term_min_bit_rate}
```

成员

成员名称	描述
gop	整型，Group Of Pictures，视频序列由若干时间连续的图像构成，在对其进行编码时，先将该视频序列分割成若干小的图像组，该参数即指一个图像组的帧数。 由于GOP结构仅支持Normalp（第一帧为I帧，其余帧为P帧），此参数也为I帧间隔，默认值65535。 由于编码I帧不需参考之前的帧，之前帧的编码质量影响从此帧开始不再传递，故该值越小，编码质量越好，但不建议小于帧率，因为I帧仅有帧内预测块，大部分场景需要有一定的P帧（有帧间预测块）才能保证编码质量。 取值范围：[1, 65536]。
stats_time	整型，CVBR码率统计时间，以秒为单位，一般场景下，$stats\_time = gop / dst\_frame\_rate$，编码质量较好。 码率统计时间越长，每帧图像的码率波动对于码率调节的影响越弱，码率的调节会更缓慢，图像质量的波动会更轻微；码率统计时间越短，每帧图像的码率波动对于码率调节的影响越强，图像码率的调节会更灵敏，图像质量的波动会更剧烈。 取值范围：[1, 60]。
src_frame_rate	整型，输入帧率，以fps为单位，即每秒进入编码器的帧数，默认值为30；码率一定的情况下，该值越小，编码质量越好，但小于25播放时画面会不连贯。 取值范围：[1, 240]。
dst_frame_rate	整型，编码器输出帧率，以fps为单位，即每秒编码器输出的帧数，默认值为30；码率一定的情况下，该值越小，编码质量越好，但小于25播放时画面会不连贯。 取值范围：[1, src_frame_rate]。 暂不支持分数帧率。
max_bit_rate	整型，编码器输出最大码率，以kbps为单位，即编码码流每秒比特数，该值越大，编码质量越好；默认值2000。 取值范围：[2, 614400]
short_term_stats_time	整型，码率短期统计时间，以秒为单位。 取值范围：[1, 120]。
long_term_stats_time	整型，码率长期统计时间，默认为分钟。 取值范围：[1, 1440]。
long_term_max_bit_rate	整型，编码器输出长期最大码率，以kbps为单位。 取值范围：[2, max_bit_rate]。
long_term_min_bit_rate	整型，编码器输出长期最小码率，以kbps为单位。 取值范围：[0, long_term_max_bit_rate]。

2.10.8.4.12 hi_venc_h264_fixqp

说明

定义 H.264 编码通道FIXQP属性字典。

Atlas 推理系列产品暂不支持。

定义

```
hi_venc_h264_fixqp = {"gop": gop,  
                      "src_frame_rate": src_frame_rate,  
                      "dst_frame_rate": dst_frame_rate,  
                      "i_qp": i_qp,  
                      "p_qp": p_qp,  
                      "b_qp": b_qp}
```

成员

成员名称	描述
gop	Group Of Pictures，视频序列由若干时间连续的图像构成，在对其进行编码时，先将该视频序列分割成若干个小的图像组，该参数即指一个图像组的帧数。 由于GOP结构仅支持Normalp（第一帧为I帧，其余帧为P帧），此参数也为I帧间隔，默认值65535。 由于编码I帧不需参考之前的帧，之前帧的编码质量影响从此帧开始不再传递，故该值越小，编码质量越好，但不建议小于帧率，因为I帧仅有帧内预测块，大部分场景需要有一定的P帧（有帧间预测块）才能保证编码质量。 取值范围：[1, 65536]。
src_frame_rate	输入帧率，以fps为单位，即每秒进入编码器的帧数，默认值为30，码率一定的情况下，该值越小，编码质量越好，但小于25时播放画面会不连贯。 该参数取值范围：[1, 240]。
dst_frame_rate	编码器输出帧率，以fps为单位，即每秒编码器输出的帧数，默认值为30，码率一定的情况下，该值越小，编码质量越好，但小于25时播放画面会不连贯。 取值范围：[1, src_frame_rate]。 暂不支持分数帧率。
i_qp	I帧所有宏块QP值。 取值范围：[0, 51]。
p_qp	P帧所有宏块QP值。 取值范围：[0, 51]。
b_qp	B帧所有宏块QP值，当前仅支持配置为0。

2.10.8.4.13 hi_venc_h264_qpmap

说明

定义 H.264 编码通道QPMap属性字典。暂不支持。

定义

```
hi_venc_h264_cvbr = {"gop": gop,  
                    "stats_time": stats_time,  
                    "src_frame_rate": src_frame_rate,  
                    "dst_frame_rate": dst_frame_rate}
```

2.10.8.4.14 hi_venc_mjpeg_cbr

说明

定义MJPEG编码通道CBR属性字典。暂不支持。

定义

```
hi_venc_mjpeg_cbr = {"stats_time": stats_time, # RW; Range:[1, 60]; the rate statistic time, the unit is  
senconds(s)  
                    "src_frame_rate": src_frame_rate, # RW; Range:[1, 240]; the input frame rate of the venc  
channel  
                    "dst_frame_rate": dst_frame_rate, # RW; Range:[1, 240]; the target frame rate of the venc  
channel,  
                    # can not be larger than src_frame_rate  
                    "bit_rate": bit_rate} # RW; Range:[2, 614400]; average bitrate} hi_venc_mjpeg_cbr;
```

2.10.8.4.15 hi_venc_mjpeg_vbr

说明

定义MJPEG编码通道VBR属性字典。暂不支持。

定义

```
hi_venc_mjpeg_cbr = {"stats_time": stats_time, # RW; Range:[1, 60]; the rate statistic time, the unit is  
senconds(s)  
                    "src_frame_rate": src_frame_rate, # RW; Range:[1, 240]; the input frame rate of the venc  
channel  
                    "dst_frame_rate": dst_frame_rate, # RW; Range:[1, 240]; the target frame rate of the venc  
channel,  
                    # can not be larger than src_frame_rate  
                    "max_bit_rate": max_bit_rate} # RW; Range:[2, 614400];the max bitrate}hi_venc_mjpeg_vbr;
```

2.10.8.4.16 hi_venc_mjpeg_fixqp

说明

定义MJPEG编码通道FIXQP属性字典。暂不支持。

定义

```
hi_venc_mjpeg_fixqp = {"src_frame_rate": src_frame_rate, # RW; Range:[1, 240]; the input frame rate of the  
venc channel  
                    "dst_frame_rate": dst_frame_rate, # RW; Range:[1, 240]; the target frame rate of the venc  
channel,
```



```
# can not be larger than src_frame_rate
"qfactor": qfactor} # RW; Range:[1,99];image quality.
```

2.10.8.4.17 hi_venc_h265_cbr

说明

定义 H.265编码通道CBR属性字典。

定义

```
hi_venc_h265_cbr = {"gop": gop,
                    "stats_time": stats_time,
                    "src_frame_rate": src_frame_rate,
                    "dst_frame_rate": dst_frame_rate,
                    "bit_rate": bit_rate}
```

成员

成员名称	描述
gop	整型，Group Of Pictures，视频序列由若干时间连续的图像构成，在对其进行编码时，先将该视频序列分割成若干个小的图像组，该参数即指一个图像组的帧数。 由于GOP结构仅支持Normalp（第一帧为I帧，其余帧为P帧），此参数也为I帧间隔，默认值65535。 由于编码I帧不需参考之前的帧，之前帧的编码质量影响从此帧开始不再传递，故该值越小，编码质量越好，但不建议小于帧率，因为I帧仅有帧内预测块，大部分场景需要有一定的P帧（有帧间预测块）才能保证编码质量。 取值范围：[1, 65536]。
stats_time	整型，CBR码率统计时间，以秒为单位，一般场景下，stats_time = gop / dst_frame_rate，编码质量较好。 码率统计时间越长，每帧图像的码率波动对于码率调节的影响越弱，码率的调节会更缓慢，图像质量的波动会更轻微；码率统计时间越短，每帧图像的码率波动对于码率调节的影响越强，图像码率的调节会更灵敏，图像质量的波动会更剧烈。 取值范围：[1, 60]。
src_frame_rate	整型，输入帧率，以fps为单位，即每秒进入编码器的帧数，默认值为30；码率一定的情况下，该值越小，编码质量越好，但小于25播放时画面会不连贯。 该参数取值范围：[1, 240]。
dst_frame_rate	整型，编码器输出帧率，以fps为单位，即每秒编码器输出的帧数，默认值为30；码率一定的情况下，该值越小，编码质量越好，但小于25播放时画面会不连贯。 取值范围：[1, src_frame_rate]。 暂不支持分数帧率。

成员名称	描述
bit_rate	整型，编码器输出平均码率，以kbps为单位，即编码码流每秒比特数，该值越大，编码质量越好；默认值2000。 该参数取值范围：[2, 614400]。 平均比特率的设置与编码通道宽高以及图像帧率都有关系。典型的平均比特率的设置如表2-68所示。注意，表2-68中的平均比特率的设置是在通道编码帧率为满帧率（30fps）时的设置。当用户设置编码输出帧率不为满帧率时，可以对下表中的码率按用户设置帧率与满帧率（30fps）的比例进行换算。

表 2-68 典型平均比特率配置

图像宽高/码率等级	1080p(1920 x 1080)	4K(3840 x 2160)
低码率	< 2Mbps	< 8Mbps
中码率	2Mbps ~ 8Mbps	8Mbps ~ 30Mbps
高码率	> 8Mbps	> 30Mbps

2.10.8.4.18 hi_venc_h265_vbr

说明

定义H.265编码通道VBR属性字典。

定义

```
hi_venc_h265_vbr = {"gop": gop,
                    "stats_time": stats_time,
                    "src_frame_rate": src_frame_rate,
                    "dst_frame_rate": dst_frame_rate,
                    "max_bit_rate": max_bit_rate}
```

成员

成员名称	描述
gop	整型，Group Of Pictures，视频序列由若干时间连续的图像构成，在对其进行编码时，先将该视频序列分割成若干个小的图像组，该参数即指一个图像组的帧数。 由于GOP结构仅支持Normalp（第一帧为I帧，其余帧为P帧），此参数也为I帧间隔，默认值65535。 由于编码I帧不需参考之前的帧，之前帧的编码质量影响从此帧开始不再传递，故该值越小，编码质量越好，但不建议小于帧率，因为I帧仅有帧内预测块，大部分场景需要有一定的P帧（有帧间预测块）才能保证编码质量。 取值范围：[1, 65536]。

成员名称	描述
stats_time	整型，VBR 码率统计时间，以秒为单位，一般场景下，stats_time = gop / dst_frame_rate，编码质量较好。 码率统计时间越长，每帧图像的码率波动对于码率调节的影响越弱，码率的调节会更缓慢，图像质量的波动会更轻微；码率统计时间越短，每帧图像的码率波动对于码率调节的影响越强，图像码率的调节会更灵敏，图像质量的波动会更剧烈。 取值范围：[1, 60]。
src_frame_rate	整型，输入帧率，以fps为单位，即每秒进入编码器的帧数，默认值为30；码率一定的情况下，该值越小，编码质量越好，但小于25播放时画面会不连贯。 该参数取值范围：[1, 240]。
dst_frame_rate	整型，编码器输出帧率，以fps为单位，即每秒编码器输出的帧数，默认值为30；码率一定的情况下，该值越小，编码质量越好，但小于25播放时画面会不连贯。 取值范围：[1, src_frame_rate]。 暂不支持分数帧率。
max_bit_rate	整型，编码器输出最大码率，以kbps为单位，即编码码流每秒比特数，该值越大，编码质量越好；默认值2000。 该参数取值范围：[2, 614400]。

注意事项

请参见[hi_venc_h264_cbr](#)中关于“src_frame_rate”和“dst_frame_rate”的说明。

2.10.8.4.19 hi_venc_h265_avbr

说明

定义H.265编码通道AVBR属性字典。

定义

```
hi_venc_h265_avbr = {"gop": gop,
                    "stats_time": stats_time,
                    "src_frame_rate": src_frame_rate,
                    "dst_frame_rate": dst_frame_rate,
                    "max_bit_rate": max_bit_rate }
```

成员

成员名称	描述
<code>gop</code>	整型，Group Of Pictures，视频序列由若干时间连续的图像构成，在对其进行编码时，先将该视频序列分割成若干小的图像组，该参数即指一个图像组的帧数。 由于GOP结构仅支持Normalp（第一帧为I帧，其余帧为P帧），此参数也为I帧间隔，默认值65535。 由于编码I帧不需参考之前的帧，之前帧的编码质量影响从此帧开始不再传递，故该值越小，编码质量越好，但不建议小于帧率，因为I帧仅有帧内预测块，大部分场景需要有一定的P帧（有帧间预测块）才能保证编码质量。 取值范围：[1, 65536]。
<code>stats_time</code>	整型，AVBR码率统计时间，以秒为单位，一般场景下，$stats\_time = gop / dst\_frame\_rate$，编码质量较好。 码率统计时间越长，每帧图像的码率波动对于码率调节的影响越弱，码率的调节会更缓慢，图像质量的波动会更轻微；码率统计时间越短，每帧图像的码率波动对于码率调节的影响越强，图像码率的调节会更灵敏，图像质量的波动会更剧烈。 取值范围：[1, 60]。
<code>src_frame_rate</code>	整型，输入帧率，以fps为单位，即每秒进入编码器的帧数，默认值为30；码率一定的情况下，该值越小，编码质量越好，但小于25播放时画面会不连贯。 该参数取值范围：[1, 240]。
<code>dst_frame_rate</code>	整型，编码器输出帧率，以fps为单位，即每秒编码器输出的帧数，默认值为30；码率一定的情况下，该值越小，编码质量越好，但小于25播放时画面会不连贯。 取值范围：[1, <code>src_frame_rate</code>]。 暂不支持分数帧率。
<code>max_bit_rate</code>	整型，编码器输出最大码率，以kbps为单位，即编码码流每秒比特数，该值越大，编码质量越好；默认值2000。 该参数取值范围：[2, 614400]。

2.10.8.4.20 hi_venc_h265_qvbr

说明

定义H.265协议编码通道QVBR模式属性字典。

定义

```
hi_venc_h265_qvbr = {"gop": gop,  
                     "stats_time": stats_time,  
                     "src_frame_rate": src_frame_rate,  
                     "dst_frame_rate": dst_frame_rate,  
                     "target_bit_rate": target_bit_rate }
```

成员

成员名称	描述
<code>gop</code>	整型，Group Of Pictures，视频序列由若干时间连续的图像构成，在对其进行编码时，先将该视频序列分割成若干个小的图像组，该参数即指一个图像组的帧数。 由于GOP结构仅支持Normalp（第一帧为I帧，其余帧为P帧），此参数也为I帧间隔，默认值65535。 由于编码I帧不需参考之前的帧，之前帧的编码质量影响从此帧开始不再传递，故该值越小，编码质量越好，但不建议小于帧率，因为I帧仅有帧内预测块，大部分场景需要有一定的P帧（有帧间预测块）才能保证编码质量。 取值范围：[1, 65536]。
<code>stats_time</code>	整型，QVBR码率统计时间，以秒为单位，一般场景下，$stats\_time = gop / dst\_frame\_rate$，编码质量较好。 码率统计时间越长，每帧图像的码率波动对于码率调节的影响越弱，码率的调节会更缓慢，图像质量的波动会更轻微；码率统计时间越短，每帧图像的码率波动对于码率调节的影响越强，图像码率的调节会更灵敏，图像质量的波动会更剧烈。 取值范围：[1, 60]。
<code>src_frame_rate</code>	整型，输入帧率，以fps为单位，即每秒进入编码器的帧数，默认值为30；码率一定的情况下，该值越小，编码质量越好，但小于25播放时画面会不连贯。 取值范围：[1, 240]。
<code>dst_frame_rate</code>	整型，编码器输出帧率，以fps为单位，即每秒编码器输出的帧数，默认值为30；码率一定的情况下，该值越小，编码质量越好，但小于25播放时画面会不连贯。 取值范围：[1, <code>src_frame_rate</code>]。 暂不支持分数帧率。
<code>target_bit_rate</code>	整型，编码器输出最大码率，以kbps为单位，即编码码流每秒比特数，该值越大，编码质量越好；默认值2000。 取值范围：[2, 614400]。

2.10.8.4.21 hi_venc_h265_cvbr

说明

定义H.265协议编码通道CVBR模式属性字典。

定义

```
hi_venc_h265_cvbr = {"gop": gop,
                    "stats_time": stats_time,
                    "src_frame_rate": src_frame_rate,
                    "dst_frame_rate": dst_frame_rate,
                    "max_bit_rate": max_bit_rate,
```

```
"short_term_stats_time": short_term_stats_time,  
"long_term_stats_time": long_term_stats_time,  
"long_term_max_bit_rate": long_term_max_bit_rate,  
"long_term_min_bit_rate": long_term_min_bit_rate}
```

成员

成员名称	描述
<code>gop</code>	整型，Group Of Pictures，视频序列由若干时间连续的图像构成，在对其进行编码时，先将该视频序列分割成若干个小的图像组，该参数即指一个图像组的帧数。 由于GOP结构仅支持Normalp（第一帧为I帧，其余帧为P帧），此参数也为I帧间隔，默认值65535。 由于编码I帧不需参考之前的帧，之前帧的编码质量影响从此帧开始不再传递，故该值越小，编码质量越好，但不建议小于帧率，因为I帧仅有帧内预测块，大部分场景需要有一定的P帧（有帧间预测块）才能保证编码质量。 取值范围：[1, 65536]。
<code>stats_time</code>	整型，CVBR码率统计时间，以秒为单位，一般场景下，$stats\_time = gop / dst\_frame\_rate$，编码质量较好。 码率统计时间越长，每帧图像的码率波动对于码率调节的影响越弱，码率的调节会更缓慢，图像质量的波动会更轻微；码率统计时间越短，每帧图像的码率波动对于码率调节的影响越强，图像码率的调节会更灵敏，图像质量的波动会更剧烈。 取值范围：[1, 60]。
<code>src_frame_rate</code>	整型，输入帧率，以fps为单位，即每秒进入编码器的帧数，默认值为30；码率一定的情况下，该值越小，编码质量越好，但小于25播放时画面会不连贯。 取值范围：[1, 240]。
<code>dst_frame_rate</code>	整型，编码器输出帧率，以fps为单位，即每秒编码器输出的帧数，默认值为30；码率一定的情况下，该值越小，编码质量越好，但小于25播放时画面会不连贯。 取值范围：[1, <code>src_frame_rate</code>]。 暂不支持分数帧率。
<code>max_bit_rate</code>	整型，编码器输出最大码率，以kbps为单位，即编码码流每秒比特数，该值越大，编码质量越好；默认值2000。 取值范围：[2, 614400]。
<code>short_term_stats_time</code>	整型，码率短期统计时间，以秒为单位。 取值范围：[1, 120]。
<code>long_term_stats_time</code>	整型，码率长期统计时间，默认为分钟。 取值范围：[1, 1440]。
<code>long_term_max_bit_rate</code>	整型，编码器输出长期最大码率，以kbps为单位。 取值范围：[2, <code>max_bit_rate</code>]。

成员名称	描述
long_term_min_bit_rate	整型，编码器输出长期最小码率，以kbps为单位。 取值范围：[0, long_term_max_bit_rate]。

2.10.8.4.22 hi_venc_h265_fixqp

说明

定义H.265编码通道FIXQP属性字典。
Atlas 推理系列产品暂不支持。

定义

```
hi_venc_h265_fixqp = {"gop": gop,  
                      "src_frame_rate": src_frame_rate,  
                      "dst_frame_rate": dst_frame_rate,  
                      "i_qp": i_qp,  
                      "p_qp": p_qp,  
                      "b_qp": b_qp}
```

成员

成员名称	描述
gop	Group Of Pictures，视频序列由若干时间连续的图像构成，在对其进行编码时，先将该视频序列分割成若干个小的图像组，该参数即指一个图像组的帧数。 由于GOP结构仅支持Normalp（第一帧为I帧，其余帧为P帧），此参数也为I帧间隔，默认值65535。 由于编码I帧不需参考之前的帧，之前帧的编码质量影响从此帧开始不再传递，故该值越小，编码质量越好，但不建议小于帧率，因为I帧仅有帧内预测块，大部分场景需要有一定的P帧（有帧间预测块）才能保证编码质量。 取值范围：[1, 65536]。
src_frame_rate	输入帧率，以fps为单位，即每秒进入编码器的帧数，默认值为30，码率一定的情况下，该值越小，编码质量越好，但小于25播放时画面会不连贯。 该参数取值范围：[1, 240]。
dst_frame_rate	编码器输出帧率，以fps为单位，即每秒编码器输出的帧数，默认值为30，码率一定的情况下，该值越小，编码质量越好，但小于25播放时画面会不连贯。 取值范围：[1, src_frame_rate]。 暂不支持分数帧率。
i_qp	I帧所有宏块QP值。 取值范围：[0, 51]。

成员名称	描述
p_qp	P帧所有宏块QP值。 取值范围：[0, 51]。
b_qp	B帧所有宏块QP值，当前仅支持配置为0。

2.10.8.4.23 hi_venc_h265_qpmap

说明

定义H.265编码通道QPMAP属性字典。暂不支持。

定义

```
hi_venc_h265_qpmap = {"gop": gop,
    "stats_time": stats_time,
    "src_frame_rate": src_frame_rate,
    "dst_frame_rate": dst_frame_rate,
    "hi_venc_rc_qpmap_mode": hi_venc_rc_qpmap_mode}
```

2.10.8.4.24 hi_venc_gop_attr

说明

定义编码器GOP属性字典。

定义

```
hi_venc_gop_attr = {"gop_mode": hi_venc_gop_mode,
    "normal_p": hi_venc_gop_normal_p}

hi_venc_gop_attr = {"gop_mode": hi_venc_gop_mode,
    "dual_p": hi_venc_gop_dual_p }

hi_venc_gop_attr = {"gop_mode": hi_venc_gop_mode,
    "smart_p": hi_venc_gop_smart_p }

hi_venc_gop_attr = {"gop_mode": hi_venc_gop_mode,
    "adv_smart_p": hi_venc_gop_adv_smart_p}

hi_venc_gop_attr = {"gop_mode": hi_venc_gop_mode,
    "bipred_b": hi_venc_gop_bipred_b}
```

成员

成员名称	描述
gop_mode	整型，编码GOP类型。
normal_p	字典，编码单参考帧P帧GOP属性字典。

成员名称	描述
dual_p	字典，编码双参考帧P帧GOP属性字典。 Atlas 推理系列产品 ，预留参数，暂不支持。 Atlas A2训练系列产品/Atlas 800I A2推理产品 ，预留参数，暂不支持。
smart_p	字典，编码智能P帧GOP属性字典。 Atlas 推理系列产品 ，预留参数，暂不支持。 Atlas A2训练系列产品/Atlas 800I A2推理产品 ，预留参数，暂不支持。
adv_smart_p	字典，编码高级智能P帧GOP属性字典，暂不支持。
bipred_b	字典，编码B帧GOP属性字典，暂不支持。

2.10.8.4.25 hi_venc_gop_mode

说明

定义 H.264/H.265 GOP类型。

定义

```
HI_VENC_GOP_MODE_NORMAL_P    = 0 # NORMALP
HI_VENC_GOP_MODE_DUAL_P      = 1 # DUALP
HI_VENC_GOP_MODE_SMART_P      = 2 # SMARTP
HI_VENC_GOP_MODE_ADV_SMART_P = 3 # ADVSMARTP
HI_VENC_GOP_MODE_BIPRED_B     = 4 # BIPREDB
HI_VENC_GOP_MODE_LOW_DELAY_B = 5 # LOWDELAYB
HI_VENC_GOP_MODE_BUTT         = 6
```

成员

成员名称	描述
HI_VENC_GOP_MODE_NORMAL_P	编码单参考帧P帧GOP类型。
HI_VENC_GOP_MODE_DUAL_P	编码双参考帧P帧GOP类型。 Atlas 推理系列产品 ，预留参数，暂不支持。 Atlas A2训练系列产品/Atlas 800I A2推理产品 ，预留参数，暂不支持。
HI_VENC_GOP_MODE_SMART_P	编码智能P帧GOP类型。 Atlas 推理系列产品 ，预留参数，暂不支持。 Atlas A2训练系列产品/Atlas 800I A2推理产品 ，预留参数，暂不支持。
HI_VENC_GOP_MODE_ADV_SMART_P	编码高级智能P帧GOP类型，暂不支持。

成员名称	描述
HI_VENC_GOP_MODE_BIPRED_B	编码B帧GOP类型，暂不支持。
HI_VENC_GOP_MODE_LOW_DELAY_B	编码B帧GOP类型，其中B帧只有前向参考帧，暂不支持。
HI_VENC_GOP_MODE_BUTT	保留值。

2.10.8.4.26 hi_venc_gop_normal_p

说明

定义编码单参考帧P帧GOP属性字典。

定义

```
hi_venc_gop_normal_p = {"ip_qp_delta": ip_qp_delta}
```

成员

成员名称	描述
ip_qp_delta	整型，I帧相对P帧的QP（Quantisation Parameter）差值。 取值范围：[-10, 30]。

2.10.8.4.27 hi_venc_gop_dual_p

说明

定义编码双参考帧 P 帧 GOP 属性字典。

Atlas 推理系列产品，预留字典，暂不支持。

Atlas A2训练系列产品/Atlas 800I A2推理产品，预留字典，暂不支持。

定义

```
hi_venc_gop_dual_p = {"sp_interval": sp_interval,  
                      "sp_qp_delta": sp_qp_delta,  
                      "ip_qp_delta": ip_qp_delta}
```

成员

成员名称	描述
sp_interval	Special P帧的间隔。 取值范围：[0, 1) U (1, gop-1]，gop是I帧间隔。
sp_qp_delta	Special P帧相对普通P帧的QP差值。 取值范围：[-10, 30]。
ip_qp_delta	I帧相对普通P帧的QP差值。 取值范围：[-10, 30]。

2.10.8.4.28 hi_venc_gop_smart_p

说明

定义编码智能 P 帧 GOP 属性字典。

Atlas 推理系列产品，预留字典，暂不支持。

Atlas A2训练系列产品/Atlas 800I A2推理产品，预留字典，暂不支持。

定义

```
hi_venc_gop_smart_p = {"bg_interval": bg_interval,
                        "bg_qp_delta": bg_qp_delta,
                        "vi_qp_delta": vi_qp_delta}
```

成员

成员名称	描述
bg_interval	长期参考帧的间隔。 取值范围：[gop, 65536]，且必须是gop的整数倍，gop是I帧间隔。
bg_qp_delta	长期参考帧相对P帧的QP差值。 取值范围：[-10, 30]。 在调试信息中，该参数等同其它码控模式的ip_qp_delta。
vi_qp_delta	虚拟I帧相对普通P帧的QP差值。 取值范围：[-10, 30]。

2.10.8.4.29 hi_venc_gop_adv_smart_p

说明

定义编码高级智能 P 帧 GOP 属性字典，暂不支持。

定义

```
hi_venc_gop_adv_smart_p= {"bg_interval": bg_interval,
                           "bg_qp_delta": bg_qp_delta,
                           "vi_qp_delta": vi_qp_delta}
```

2.10.8.4.30 hi_venc_gop_bipred_b

说明

定义编码B帧GOP属性字典，暂不支持。

定义

```
hi_venc_gop_bipred_b = {"b_frame_num": b_frame_num, # RW; Range:[1, 3]; Number of B frames
                        "b_qp_delta": b_qp_delta, # RW; Range:[-10, 30]; QP variance between P frame and B frame
                        "ip_qp_delta": ip_qp_delta} # RW; Range:[-10, 30]; QP variance between P frame and I
frame
```

2.10.8.4.31 hi_venc_start_param

说明

定义编码通道连续接收并编码的帧数字典。

定义

```
hi_venc_start_param = {"recv_pic_num": recv_pic_num}
```

成员

成员名称	描述
recv_pic_num	整型，编码通道连续接收并编码的帧数。 取值范围：-1或正整数。

2.10.8.4.32 hi_venc_chn_status

说明

定义编码通道的状态字典。

定义

```
hi_venc_chn_status = {"left_pics": left_pics, # R; left picture number
                      "left_stream_bytes": left_stream_bytes, # R; left stream bytes
                      "left_stream_frames": left_stream_frames, # R; left stream frames
                      "cur_packs": cur_packs, # R; pack number of current frame
                      "left_rcv_pics": left_rcv_pics, # R; Number of frames to be received.
                      "left_enc_pics": left_enc_pics, # R; Number of frames to be encoded.
                      "is_jpeg_snap_end": is_jpeg_snap_end, # R; the end of Snap.
                      "release_pic_pts": release_pic_pts,
                      "stream_info": hi\_venc\_stream\_info}
```

成员

成员名称	描述
left_pics	整型，待编码的图像数，暂不支持。
left_stream_bytes	整型，码流Buffer剩余的Byte数，暂不支持。
left_stream_frames	整型，码流Buffer剩余的帧数，暂不支持。
cur_packs	整型，当前帧的码流包个数。
left_recv_pics	整型，剩余待接收的帧数，在用户调用接口 2.10.6.7 acl.himpi.venc_start_chn 设置接收帧数后有效，暂不支持。
left_enc_pics	整型，剩余待编码的帧数，在用户调用接口 2.10.6.7 acl.himpi.venc_start_chn 设置接收帧数后有效，暂不支持。
is_jpeg_snap_end	整型，Jpege抓拍模式下指示抓拍过程是否结束，暂不支持。
release_pic_pts	整型，预留参数，暂不支持。
stream_info	整型，码流信息。 Atlas 推理系列产品 ，预留参数，暂不支持。 Atlas A2训练系列产品/Atlas 800I A2推理产品 ，预留参数，暂不支持。

注意事项

如果VENC接收的抓拍帧有丢帧出现，“is_jpeg_snap_end”会存在不能正确标记抓拍结束的情况。

2.10.8.4.33 hi_venc_h264_stream_info

说明

定义H.264协议码流特征信息。

Atlas 推理系列产品，预留字典，暂不支持。

Atlas A2训练系列产品/Atlas 800I A2推理产品，预留字典，暂不支持。

定义

```
hi_venc_h264_stream_info = {"pic_bytes": pic_bytes,  
                             "inter16x16_mb_num": inter16x16_mb_num,  
                             "inter8x8_mb_num": inter8x8_mb_num,  
                             "intra16_mb_num": intra16_mb_num,  
                             "intra8_mb_num": intra8_mb_num,  
                             "intra4_mb_num": intra4_mb_num,  
                             "ref_type": hi\_venc\_ref\_type,  
                             "update_attr_cnt": update_attr_cnt,  
                             "start_qp": start_qp,
```

```
"mean_qp": mean_qp,  
"is_p_skip": is_p_skip }
```

成员

成员名称	描述
pic_bytes	整型，编码当前帧的字节（Byte）数。
inter16x16_mb_num	整型，编码当前帧中采用 Inter16x16 预测模式的宏块数。 Atlas 200I/500 A2推理产品，暂不支持该参数。
inter8x8_mb_num	整型，编码当前帧中采用 Inter8x8 预测模式的宏块数。 Atlas 200I/500 A2推理产品，暂不支持该参数。
intra16_mb_num	整型，编码当前帧中采用 Intra16 预测模式的宏块数。 Atlas 200I/500 A2推理产品，暂不支持该参数。
intra8_mb_num	整型，编码当前帧中采用 Intra8 预测模式的宏块数。 Atlas 200I/500 A2推理产品，暂不支持该参数。
intra4_mb_num	整型，编码当前帧中采用 Intra4 预测模式的宏块数。 Atlas 200I/500 A2推理产品，暂不支持该参数。
ref_type	整型，高级跳帧参考下的编码帧类型。 Atlas 200I/500 A2推理产品，暂不支持该参数。
update_attr_cnt	整型，通道属性或参数(包含 RC 参数)被设置的次数。 Atlas 200I/500 A2推理产品，暂不支持该参数。
start_qp	整型，编码当前帧的 startqp 值。
mean_qp	整型，编码当前帧的平均 QP 帧。
is_p_skip	整型，标识当前帧是否为 pskip 帧。 Atlas 200I/500 A2推理产品，暂不支持该参数。

2.10.8.4.34 hi_venc_ref_type

说明

定义H.264跳帧参考码流的帧类型和参考属性，暂不支持。

定义

```
HI_VENC_BASE_IDR_SLICE           = 0 # the Idr frame at Base layer  
HI_VENC_BASE_P_SLICE_REF_TO_IDR  = 1 # the P frame at Base layer referenced by other frames at  
Base layer and reference to Idr frame  
HI_VENC_BASE_P_SLICE_REF_BY_BASE  = 2 # the P frame at Base layer referenced by other frames at  
Base layer  
HI_VENC_BASE_P_SLICE_REF_BY_ENHANCE = 3 # the P frame at Base layer referenced by other frames at  
Enhance layer  
HI_VENC_ENHANCE_P_SLICE_REF_BY_ENHANCE = 4 #the P frame at Enhance layer referenced by other  
frames at Enhance layer
```

```
HI_VENC_ENHANCE_P_SLICE_NOT_FOR_REF = 5 # the P frame at Enhance layer not referenced
HI_VENC_P_SLICE_BUTT = 6
```

2.10.8.4.35 hi_venc_jpeg_stream_info

说明

定义JPEG协议码流特征信息， 暂不支持。

定义

```
hi_venc_jpeg_stream_info = {"pic_bytes": pic_bytes,
                           "update_attr_cnt": update_attr_cnt,
                           "qfactor": qfactor}
```

成员

成员名称	描述
pic_bytes	整型，一帧JPEG码流大小，以字节（Byte）为单位。
update_attr_cnt	整型，通道属性或参数（包含 RC 参数）被设置的次数。
qfactor	整型，编码当前帧的 Qfactor。

2.10.8.4.36 hi_venc_h265_stream_info

说明

定义H.265协议码流特征信息。

Atlas 推理系列产品，预留字典，暂不支持。

Atlas A2训练系列产品/Atlas 800I A2推理产品，预留字典，暂不支持。

定义

```
hi_venc_h265_stream_info = {"pic_bytes": pic_bytes,
                             "inter64x64_cu_num": inter64x64_cu_num,
                             "inter32x32_cu_num": inter32x32_cu_num,
                             "inter16x16_cu_num": inter16x16_cu_num,
                             "inter8x8_cu_num": inter8x8_cu_num,
                             "intra32x32_cu_num": intra32x32_cu_num,
                             "intra16x16_cu_num": intra16x16_cu_num,
                             "intra8x8_cu_num": intra8x8_cu_num,
                             "intra4x4_cu_num": intra4x4_cu_num,
                             "ref_type": hi\_venc\_ref\_type,
                             "update_attr_cnt": update_attr_cnt,
                             "start_qp": start_qp,
                             "mean_qp": mean_qp,
                             "is_p_skip": is_p_skip}
```

成员

成员名称	描述
pic_bytes	整型，编码当前帧的字节数。

成员名称	描述
inter64x64_cu_num	整型，编码当前帧中采用 Inter64x64 预测模式的 CU 块数。 Atlas 200I/500 A2推理产品，暂不支持该参数。
inter32x32_cu_num	整型，编码当前帧中采用 Inter32x32 预测模式的 CU 块数。 Atlas 200I/500 A2推理产品，暂不支持该参数。
inter16x16_cu_num	整型，编码当前帧中采用 Inter16x16 预测模式的 CU 块数。 Atlas 200I/500 A2推理产品，暂不支持该参数。
inter8x8_cu_num	整型，编码当前帧中采用 Inter8x8 预测模式的 CU 块数。 Atlas 200I/500 A2推理产品，暂不支持该参数。
intra32x32_cu_num	整型，编码当前帧中采用 Intra32x32 预测模式的 CU 块数。 Atlas 200I/500 A2推理产品，暂不支持该参数。
intra16x16_cu_num	整型，编码当前帧中采用 Intra16x16 预测模式的 CU 块数。 Atlas 200I/500 A2推理产品，暂不支持该参数。
intra8x8_cu_num	整型，编码当前帧中采用 Intra8x8 预测模式的 CU 块数。 Atlas 200I/500 A2推理产品，暂不支持该参数。
intra4x4_cu_num	整型，编码当前帧中采用 Intra4x4 预测模式的 CU 块数。 Atlas 200I/500 A2推理产品，暂不支持该参数。
ref_type	整型，高级跳帧参考下的编码帧类型。 Atlas 200I/500 A2推理产品，暂不支持该参数。
update_attr_cnt	整型，通道属性或参数（包含 RC 参数）被设置的次数。 Atlas 200I/500 A2推理产品，暂不支持该参数。
start_qp	整型，编码当前帧的 startqp 值。
mean_qp	整型，编码当前帧的平均 QP 值。
is_p_skip	整型，标识当前帧是否为 pskip 帧。 Atlas 200I/500 A2推理产品，暂不支持该参数。

注意事项

“ref_type”请参见[acl.himpi.venc_h264_stream_info](#)中关于“ref_type”变量的说明。

2.10.8.4.37 hi_venc_prores_stream_info

说明

定义PRORES协议码流特征信息， 暂不支持。

定义

```
hi_venc_prores_stream_info= {"pic_bytes": pic_bytes,  
                             "update_attr_cnt": update_attr_cnt}
```

成员

成员名称	描述
pic_bytes	整型，编码当前帧的字节数。
update_attr_cnt	整型，通道属性或参数被设置的次数。

2.10.8.4.38 hi_venc_mod_param

说明

编码相关模块参数。

定义

```
hi_venc_mod_param = {"mod_type": hi\_venc\_mod\_type,  
                     "venc_mod_param": hi\_venc\_venc\_mod\_param}  
  
hi_venc_mod_param = {"mod_type": hi\_venc\_mod\_type,  
                     "h264_mod_param": hi\_venc\_h264\_mod\_param}  
  
hi_venc_mod_param = {"mod_type": hi\_venc\_mod\_type,  
                     "h265_mod_param": hi\_venc\_h265\_mod\_param}  
  
hi_venc_mod_param = {"mod_type": hi\_venc\_mod\_type,  
                     "jpeg_mod_param": hi\_venc\_jpeg\_mod\_param}
```

成员

成员名称	描述
mod_type	整型，设置或者获取模块参数的类型。
venc_mod_param h264_mod_param h265_mod_param jpeg_mod_param	字典，VENC内部各个模块的工作参数。

2.10.8.4.39 hi_venc_mod_type

说明

编码相关模块参数类型。

定义

```
HI_VENC_MOD_VENC = 1
HI_VENC_MOD_H264 = 2
HI_VENC_MOD_H265 = 3
HI_VENC_MOD_JPEG = 4
HI_VENC_MOD_RC = 5
HI_VENC_MOD_BUTT = 6
```

成员

成员名称	描述
HI_VENC_MOD_VENC	VENC内公共模块参数类型。
HI_VENC_MOD_H264	VENC内h264e模块参数类型。
HI_VENC_MOD_H265	VENC内h265e模块参数类型。
HI_VENC_MOD_JPEG	VENC内jpege模块参数类型。
HI_VENC_MOD_RC	VENC内码控模块参数类型。暂不支持设置。
HI_VENC_MOD_BUTT	保留值。

2.10.8.4.40 hi_venc_venc_mod_param

说明

Device上的VENC内公共模块参数。

定义

```
hi_venc_venc_mod_param = {"buf_cache": buf_cache,
                           "frame_buf_recycle": frame_buf_recycle}
```

成员

成员名称	描述
buf_cache	整型，码流获取是否支持Cache方式，暂不支持，默认值为0。 0：关闭码流Buffer Cache。 1：打开码流Buffer Cache。

成员名称	描述
frame_buf_recycle	整型，帧存是否回收，暂不支持，默认值为0。 0：关闭编码帧存回收。 1：打开编码帧存回收。

2.10.8.4.41 hi_venc_h264_mod_param

说明

Device上的VENC内h264e模块参数。

定义

```
hi_venc_h264_mod_param = {"one_stream_buf": one_stream_buf, # RW; Range:[0, 1]; one stream buffer
                           "mini_buf_mode": mini_buf_mode, # RW; Range:[0, 1]; H264e MiniBufMode
                           "low_power_mode": low_power_mode, # RW; Range:[0, 1]; H264e PowerSaveEn
                           "vb_src": hi_vb_src, # RW; H264e VBSource
                           "qp_hist_en": qp_hist_en, # RW; Range:[0, 1]
                           "max_user_data_len": max_user_data_len} # RW; Range:[0, 65536]; one user data buffer
len
```

成员

成员名称	描述
one_stream_buf	整型，编码码流帧配置模式，默认值为1。 0：多包模式。 1：单包模式。
mini_buf_mode	整型，编码码流Buffer配置模式，暂不支持, 默认值为1。 0：码流Buffer根据分辨率分配。 1：码流Buffer下限为32k，用户保证合理。
low_power_mode	整型，低功耗模式，暂不支持，默认值为0。 0：关闭低功耗模式。 1：使能低功耗模式。 注意：低功耗使能后会导致图像质量损失。
vb_src	整型，参考帧和重构帧的帧存分配方式，默认值为“HI_VB_SRC_PRIVATE”。 - HI_VB_SRC_PRIVATE：私有VB方式，驱动内部完成帧存分配。 - HI_VB_SRC_USER：用户VB，需由用户分配帧存，暂不支持

成员名称	描述
qp_hist_en	整型，QP 直方图输出控制模式，暂不支持，默认值为1。 0：QP 直方图不输出，即 hi_venc_h264_adv_stream_info 中的成员变量“qp_hist”为全0。 1：QP 直方图输出，即 hi_venc_h264_adv_stream_info 中的成员变量“qp_hist”为实际的当前帧的QP直方图。
max_user_data_len	整型，用户数据所占内存的最大大小，单位Byte，暂不支持，默认值为1024。

2.10.8.4.42 hi_venc_h265_mod_param

说明

Device上的VENC内h265e模块参数。

定义

```
hi_venc_h265_mod_param = {"one_stream_buf": one_stream_buf, # RW; Range:[0, 1]; one stream buffer
                           "mini_buf_mode": mini_buf_mode, # RW; Range:[0, 1]; H265e MiniBufMode
                           "low_power_mode": low_power_mode, # RW; Range:[0, 2]; H265e PowerSaveEn
                           "vb_src": hi\_vb\_src, # RW; H265e VBSource
                           "qp_hist_en": qp_hist_en, # RW; Range:[0, 1]
                           "max_user_data_len": max_user_data_len} # RW; Range:[0, 65536]; one user data buffer
len
```

成员

成员名称	描述
one_stream_buf	整型，编码码流帧配置模式，默认值为1。 0：多包模式。 1：单包模式。
mini_buf_mode	整型，编码码流Buffer配置模式，暂不支持，默认值为1。 0：码流Buffer根据分辨率分配。 1：码流Buffer下限为32k，用户保证合理。
low_power_mode	整型，低功耗模式，暂不支持，默认值为0。 0：关闭低功耗模式。 1：使能低功耗模式。 2：使能极低功耗模式。 注意：低功耗或者极低功耗使能后会导致图像质量损失。

成员名称	描述
vb_src	整型，参考帧和重构帧的帧存分配方式，默认值为“HI_VB_SRC_PRIVATE”。 - HI_VB_SRC_PRIVATE：私有VB方式，驱动内部完成帧存分配。 - HI_VB_SRC_USER：用户VB，需由用户分配帧存，暂不支持。
qp_hist_en	整型，QP 直方图输出控制模式，暂不支持，默认值为1。 0：QP 直方图不输出，即 hi_venc_h265_adv_stream_info 中的成员变量“qp_hist”为全0。 1：QP 直方图输出，即 hi_venc_h265_adv_stream_info 中的成员变量“qp_hist”为实际的当前帧的QP直方图。
max_user_data_len	整型，用户数据所占内存的最大大小，单位Byte，暂不支持，默认值为1024。

2.10.8.4.43 hi_venc_jpeg_mod_param

说明

Device上的VENC内jpeg模块中参数。

定义

```
hi_venc_jpeg_mod_param = {"one_stream_buf": one_stream_buf, # RW; Range:[0, 1]; one stream buffer  
"mini_buf_mode": mini_buf_mode, # RW; Range:[0, 1]; Jpege MiniBufMode  
"clear_stream_buf": clear_stream_buf, # RW; Range:[0, 1]; JpegClearStreamBuf  
"dering_mode": dering_mode} # RW; Range:[0, 1]; Jpege Dering Mode ;
```

成员

成员名称	描述
one_stream_buf	整型，编码码流帧配置模式，默认值为1。 0：多包模式，暂不支持。 1：单包模式。
mini_buf_mode	整型，编码码流Buffer配置模式，暂不支持，默认值为0。 0：码流Buffer根据分辨率分配。 1：码流Buffer下限为32k，用户保证合理。
clear_stream_buf	整型，JPEG 编码通道设置属性时是否清空码流Buffer，暂不支持，默认值为1。 0：设置通道属性时保留码流Buffer和上下文计数。 1：设置通道属性时清空码流Buffer。

成员名称	描述
dering_mode	预留参数，暂不使用，默认值为0。

2.10.8.4.44 hi_venc_jpeg_param

说明

用于设置或者获取指定JPEG编码通道的量化参数的数据字典。

定义

```
hi_venc_jpeg_param = {"qfactor": qfactor, # RW; Range:0xFFFFFFFF or [1,100]; Qfactor value
    "y_qt": y_qt[64], # RW; Range:[1, 255]; Y quantization table
    "cb_qt": cb_qt[64], # RW; Range:[1, 255]; Cb quantization table
    "cr_qt": cr_qt[64], # RW; Range:[1, 255]; Cr quantization table
    "mcu_per_ecs": mcu_per_ecs, # RW; the max MCU number is (picwidth + 15) >> 4 x (picheight +15) >> 4 x 2]; MCU number of one ECS
    "ecs_output_en": ecs_output_en}
```

成员

成员名称	描述
qfactor	整型，编码质量范围0xFFFFFFFF或[1, 100]，数值越小图片质量越差，默认100。 0xFFFFFFFF：配置为该值，则用户需要自行配置 y_qt、cb_qt、cr_qt，量化表的配置会影响最后的编码结果。 [1, 100]：配置为该范围内的值，表示y_qt、cb_qt、cr_qt参数不生效，以系统默认的为准。
y_qt	列表，Y量化表。 取值范围：[1, 255]。
cb_qt	列表，Cb量化表。 取值范围：[1, 255]。
cr_qt	列表，Cr量化表。 取值范围：[1, 255]。
mcu_per_ecs	整型，每个ECS中包含多少个MCU，系统默认为 0，表示不划分ECS，预留参数，当前不支持。 mcu_per_ecs: [0, (picwidth+15)>>4 x (picheight +15)>>4 x 2]
ecs_output_en	整型，预留参数，暂不支持。

2.10.8.4.45 hi_venc_h264_adv_stream_info

说明

定义H.264协议码流高级特征信息。

Atlas 推理系列产品，预留字典，暂不支持。

Atlas A2训练系列产品/Atlas 800I A2推理产品，预留字典，暂不支持。

定义

```
hi_venc_h264_adv_stream_info = {"residual_bits": residual_bits, # R; the residual num
                                "head_bits": head_bits, # R; the head bit num
                                "madi_val": madi_val, # R; the madi value
                                "madp_val": madp_val, # R; the madp value
                                "psnr_val": psnr_val, # R; the PSNR value
                                "sse_lcu_cnt": sse_lcu_cnt, # R; the lcu cnt of the sse
                                "sse_sum": sse_sum, # R; the sum of the sse
                                "sse_info": [hi_venc_sse_info], # R; the information of the sse
                                "qp_hist": [qp_hist], # R; the Qp histogram value
                                "move_scene16x16_num": move_scene16x16_num, # R; the 16x16 cu num of the
move scene
                                "move_scene_bits": move_scene_bits} # R; the stream bit num of the move scene
```

成员

成员名称	描述
residual_bits	整型，编码当前帧残差（bit）数。 Atlas 200I/500 A2推理产品 ，暂不支持该参数。
head_bits	整型，编码当前帧头信息的（bit）数。 Atlas 200I/500 A2推理产品 ，暂不支持该参数。
madi_val	整型，编码当前帧空域纹理复杂度Madi值。 Atlas 200I/500 A2推理产品 ，暂不支持该参数。
madp_val	整型，编码当前帧时域运动复杂度Madp值。 Atlas 200I/500 A2推理产品 ，暂不支持该参数。
psnr_val	整型，编码当前帧的PSNR（Peak signal-to-noise ratio，峰值信噪比）值。 不同软件中关于峰值信噪比的计算公式略有不同，因此峰值信噪比值可能存在差异。 仅支持输入图片分辨率小于或等于3840 * 2160分辨率时获取“psnr_val”值，否则“psnr_val”值无效。
sse_lcu_cnt	整型，编码当前帧中LCU（Largest Coding Unit）个数。
sse_sum	整型，编码当前帧中SSE（Sum of Squared Errors，和方差）值。 不同软件中关于和方差的计算公式略有不同，因此和方差值可能存在差异。

成员名称	描述
sse_info	列表，编码当前帧中 8个区域的SSE（和方差）值。 不同软件中关于和方差的计算公式略有不同，因此和方差值可能存在差异。
qp_hist	列表，编码当前帧QP直方图。 Atlas 200I/500 A2推理产品 ，暂不支持该参数。
move_scene16x16_num	整型，判断为图像前景的16*16块的数目，需要开启前景宏块级码控制。 Atlas 200I/500 A2推理产品 ，暂不支持该参数。
move_scene_bits	整型，判断为图像前景区域编码bit数，需要开启前景宏块级码控制。 Atlas 200I/500 A2推理产品 ，暂不支持该参数。

2.10.8.4.46 hi_venc_h265_adv_stream_info

说明

定义H.265协议码流高级特征信息。

Atlas 推理系列产品，预留字典，暂不支持。

Atlas A2训练系列产品/Atlas 800I A2推理产品，预留字典，暂不支持。

定义

```
hi_venc_h265_adv_stream_info = {"residual_bits": residual_bits, # R; the residual num
                                "head_bits": head_bits, # R; the head bit num
                                "madi_val": madi_val, # R; the madi value
                                "madp_val": madp_val, # R; the madp value
                                "psnr_val": psnr_val, # R; the PSNR value
                                "sse_lcu_cnt": sse_lcu_cnt, # R; the lcu cnt of the sse
                                "sse_sum": sse_sum, # R; the sum of the sse
                                "sse_info": [hi_venc_sse_info], # R; the information of the sse
                                "qp_hist": [qp_hist], # R; the Qp histogram value
                                "move_scene32x32_num": move_scene32x32_num, # R; the 32x32 cu num of the
move scene
                                "move_scene_bits": move_scene_bits} # R; the stream bit num of the move scene
```

成员

成员名称	描述
residual_bits	整型，编码当前帧残差（bit）数。 Atlas 200I/500 A2推理产品 ，暂不支持该参数。
head_bits	整型，编码当前帧头信息的（bit）数。 Atlas 200I/500 A2推理产品 ，暂不支持该参数。

成员名称	描述
madi_val	整型，编码当前帧空域纹理复杂度Madi值。 Atlas 200I/500 A2推理产品，暂不支持该参数。
madp_val	整型，编码当前帧时域运动复杂度Madp值。 Atlas 200I/500 A2推理产品，暂不支持该参数。
psnr_val	整型，编码当前帧的PSNR（峰值信噪比）值。 不同软件中关于峰值信噪比的计算公式略有不同，因此峰值信噪比值可能存在差异。 仅支持输入图片分辨率小于或等于3840 * 2160分辨率时获取“psnr_val”值，否则“psnr_val”值无效。
sse_lcu_cnt	整型，编码当前帧中LCU个数。
sse_sum	编码当前帧中SSE（和方差）值。 不同软件中关于和方差的计算公式略有不同，因此和方差值可能存在差异。
sse_info	列表，编码当前帧中 8个区域的SSE（和方差）值。 不同软件中关于和方差的计算公式略有不同，因此和方差值可能存在差异。
qp_hist	列表，编码当前帧QP直方图。 Atlas 200I/500 A2推理产品，暂不支持该参数。
move_scene32x32_num	整型，判断为图像前景的32 * 32块的数目，需要开启前景宏块级码控制。 Atlas 200I/500 A2推理产品，暂不支持该参数。
move_scene_bits	整型，判断为图像前景区域编码bit数，需要开启前景宏块级码控制。 Atlas 200I/500 A2推理产品，暂不支持该参数。

2.10.8.4.47 hi_venc_data_type

说明

定义码流结果类型。

定义

```
hi_venc_data_type = {"h264_type": hi_venc_h264_nalu_type, # R; H264E NALU types
                    "jpeg_type": hi_venc_jpege_pack_type, # R; JPEG pack types
                    "h265_type": hi_venc_h265_nalu_type, # R; H265E NALU types
                    "prores_type": hi_venc_prores_pack_type}
```

成员

成员名称	描述
h264_type	整型，H.264码流包类型。
jpeg_type	整型，JPEG码流包类型。
h265_type	整型，H.265码流包类型。
prores_type	整型，PRORES码流包类型。

2.10.8.4.48 hi_venc_h264_nalu_type

说明

定义H.264码流NALU类型。

定义

```
HI_VENC_H264_NALU_B_SLICE = 0 # B SLICE types
HI_VENC_H264_NALU_P_SLICE = 1 # P SLICE types
HI_VENC_H264_NALU_I_SLICE = 2 # I SLICE types
HI_VENC_H264_NALU_IDR_SLICE = 5 # IDR SLICE types
HI_VENC_H264_NALU_SEI = 6 # SEI types
HI_VENC_H264_NALU_SPS = 7 # SPS types
HI_VENC_H264_NALU_PPS = 8 # PPS types
HI_VENC_H264_NALU_BUTT = 9
```

2.10.8.4.49 hi_venc_jpege_pack_type

说明

定义JPEG码流的PACK类型。

定义

```
HI_VENC_JPEG_PACK_ECS = 5 # ECS types
HI_VENC_JPEG_PACK_APP = 6 # APP types
HI_VENC_JPEG_PACK_VDO = 7 # VDO types
HI_VENC_JPEG_PACK_PIC = 8 # PIC types
HI_VENC_JPEG_PACK_DCF = 9 # DCF types
HI_VENC_JPEG_PACK_DCF_PIC = 10 # DCF PIC types
HI_VENC_JPEG_PACK_BUTT = 11
```

2.10.8.4.50 hi_venc_h265_nalu_type

说明

定义H.265码流NALU类型。

定义

```
HI_VENC_H265_NALU_B_SLICE = 0 # B SLICE types
HI_VENC_H265_NALU_P_SLICE = 1 # P SLICE types
HI_VENC_H265_NALU_I_SLICE = 2 # I SLICE types
HI_VENC_H265_NALU_IDR_SLICE = 19 # IDR SLICE types
```

```
HI_VENC_H265_NALU_VPS      = 32 # VPS types
HI_VENC_H265_NALU_SPS      = 33 # SPS types
HI_VENC_H265_NALU_PPS      = 34 # PPS types
HI_VENC_H265_NALU_SEI      = 39 # SEI types
HI_VENC_H265_NALU_ENHANCE  = 64 # ENHANCE typesHI_VENC_H265_NALU_BUTT
```

2.10.8.4.51 hi_venc_prores_pack_type

说明

定义PRORES码流的PACK类型，暂不支持。

定义

```
HI_VENC_PRORES_PACK_PIC = 1 # PIC types
HI_VENC_PRORES_PACK_BUTT = 2
```

2.10.8.4.52 hi_venc_chn_param

说明

定义VENC通道参数字典。

定义

```
hi_venc_chn_param = {"color_to_grey_en": color_to_grey_en,
                      "priority": priority,
                      "max_stream_cnt": max_stream_cnt,
                      "poll_wake_up_frame_cnt": poll_wake_up_frame_cnt,
                      "crop_info": hi_crop_info,
                      "frame_rate": hi_frame_rate_ctrl}
```

成员

成员名称	描述
color_to_grey_en	整型，开启或关闭一个通道的彩转灰功能。 Atlas 推理系列产品 ，预留参数，暂不支持。 Atlas A2训练系列产品/Atlas 800I A2推理产品 ，预留参数，暂不支持。
priority	整型，编码通道优先级参数，默认为0，数字越大优先级越高。 取值范围：[0, 2)。
max_stream_cnt	整型，最大码流缓存帧数，暂不支持。
poll_wake_up_frame_cnt	整型，当通道使用超时或阻塞获取码流，编码指定的帧“poll_wake_up_frame_cnt”之后唤醒阻塞接口，暂不支持。 取值范围：大于0 默认值：1
crop_info	字典，预留参数，暂不支持。
frame_rate	字典，通道帧率控制参数，暂不支持。

2.10.8.4.53 hi_venc_stream_info

说明

定义码流信息字典。

Atlas 推理系列产品，预留字典，暂不支持。

Atlas A2训练系列产品/Atlas 800I A2推理产品，预留字典，暂不支持。

定义

```
hi_venc_stream_info = {"ref_type": hi_venc_ref_type, # R;Type of encoded frames in advanced frame  
skipping reference mode  
    "pic_bytes": pic_bytes, # R;the coded picture stream byte number  
    "pic_cnt": pic_cnt, # R;When channel attributes 'is_by_frame == 1', it means count of frames.  
    # When channel attributes 'is_by_frame == 0', it means count of packets  
    "start_qp": start_qp, # R;the start Qp of encoded frames  
    "mean_qp": mean_qp, # R;the mean Qp of encoded frames  
    "is_p_skip": is_p_skip,  
    "residual_bits": residual_bits, # R;residual  
    "head_bits": head_bits, # R;head information  
    "madi_val": madi_val, # R;madi  
    "madp_val": madp_val, # R;madp  
    "sse_sum": sse_sum, # R;Sum of MSE value  
    "sse_lcu_cnt": sse_lcu_cnt, # R;Sum of LCU number  
    "psnr_val": psnr_val} # R;PSNR
```

成员

成员名称	描述
ref_type	整型，高级跳帧参考下的编码帧类型。 Atlas 200I/500 A2推理产品 ，暂不支持该参数。
pic_bytes	整型，已编码码流的字节数。
pic_cnt	整型，当通道属性is_by_frame == 1时，表示帧数；当is_by_frame == 0时，表示包数。 Atlas 200I/500 A2推理产品 ，暂不支持该参数。
start_qp	整型，编码帧的起始QP。
mean_qp	整型，编码帧的平均QP。
is_p_skip	整型，是否P帧是skip帧。 Atlas 200I/500 A2推理产品 ，暂不支持该参数。
residual_bits	整型，编码当前帧残差bit数。 Atlas 200I/500 A2推理产品 ，暂不支持该参数。
head_bits	整型，编码当前帧头信息bit数。 Atlas 200I/500 A2推理产品 ，暂不支持该参数。

成员名称	描述
madi_val	整型，编码当前帧空域纹理复杂度Madi值。 Atlas 200I/500 A2推理产品 ，暂不支持该参数。
madp_val	整型，编码当前帧时域运动复杂度Madp值。 Atlas 200I/500 A2推理产品 ，暂不支持该参数。
sse_sum	整型，编码当前帧中SSE（和方差）值。 不同软件中关于和方差的计算公式略有不同，因此和方差值可能存在差异。
sse_lcu_cnt	整型，编码当前帧中LCU个数。
psnr_val	整型，编码当前帧的PSNR（峰值信噪比）值。 不同软件中关于峰值信噪比的计算公式略有不同，因此峰值信噪比值可能存在差异。 - 对于H.264，仅支持输入图片分辨率的乘积小于4096 * 2032时获取“psnr_val”值，否则“psnr_val”值无效。 - 对于H.265，仅支持输入图片分辨率的乘积小于8192 * 4064时获取“psnr_val”值，否则“psnr_val”值无效。

2.10.8.4.54 hi_venc_pack_info

说明

定义当前码流包数据中包含的其他类型码流包数据的字典。

定义

```
hi_venc_pack_info = {"pack_type": hi_venc_data_type, # R; the pack type
                    "pack_offset": pack_offset,
                    "pack_len": pack_len}
```

成员

成员名称	描述
pack_type	整型，当前码流包数据包含其他码流包的类型。
pack_offset	整型，当前码流包数据包含其他码流包数据的偏移。
pack_len	整型，当前码流包数据包含其他码流包数据的大小。

2.10.8.4.55 hi_frame_rate_ctrl

说明

定义通道帧率控制参数，暂不支持。

定义

```
hi_frame_rate_ctrl = {"src_frame_rate": src_frame_rate, # RW; Range:[0, 240]; Input frame rate of a channel  
"dst_frame_rate": dst_frame_rate} # RW; Range:[0, 240]; Output frame rate of a channel}
```

成员

成员名称	描述
src_frame_rate	整型，进入通道的帧率，以fps为单位。 取值范围：-1 或[1, 240]。
dst_frame_rate	整型，通道输出的帧率，以fps为单位。 取值范围： • JPEG：-1 或[1, src_frame_rate]; • H.264/H.265：-1 或[1, 240]。

2.10.8.4.56 hi_venc_rc_qpmap_mode

说明

定义编码通道QPMAP模式下CU32、CU64 QP值的取值方式，暂不支持。

定义

```
HI_VENC_RC_QPMAP_MODE_MEAN_QP = 0  
HI_VENC_RC_QPMAP_MODE_MIN_QP  = 1  
HI_VENC_RC_QPMAP_MODE_MAX_QP  = 2  
HI_VENC_RC_QPMAP_MODE_BUTT    = 3
```

成员

成员名称	描述
HI_VENC_RC_QPMAP_MODE_MEAN_QP	CU32 QP值取4个16 * 16块QP值的平均值。 CU64 QP值取16个16 * 16块QP值的平均值。
HI_VENC_RC_QPMAP_MODE_MIN_QP	CU32 QP值取4个16 * 16块QP值的最小值。 CU64 QP值取16个16 * 16块QP值的最小值。
HI_VENC_RC_QPMAP_MODE_MAX_QP	CU32 QP值取4个16 * 16块QP值的最大值。 CU64 QP值取16个16 * 16块QP值的最大值。
HI_VENC_RC_QPMAP_MODE_BUTT	保留值。

2.10.8.4.57 hi_venc_sse_info

说明

定义H.264/H.265协议SSE信息，暂不支持。

定义

```
hi_venc_sse_info = {"enable": enable, # RW; Range:[0,1]; Region SSE enable  
                   "sse_val": sse_val} # R; Region SSE value
```

成员

成员名称	描述
enable	整型，区域SSE使能。1表示使能，0表示不使能。
sse_val	整型，区域SSE值。

2.10.8.4.58 hi_venc_h264_attr

说明

定义H.264编码属性字典。

Atlas 推理系列产品，预留字典，暂不支持。

Atlas A2训练系列产品/Atlas 800I A2推理产品，预留字典，暂不支持。

定义

```
hi_venc_h264_attr = {"rcn_ref_share_buf_en": rcn_ref_share_buf_en, # RW; Range:[0, 1];Whether to enable  
                    the Share Buf of Rcn and Ref  
                    "frame_buf_ratio": frame_buf_ratio}
```

成员

成员名称	描述
rcn_ref_share_buf_en	是否使能参考帧和回写帧复用Buffer。预留参数，暂不支持。 取值范围：[0, 1]。1表示使能，0表示不使能。 静态属性。

成员名称	描述
frame_buf_ratio	帧Buffer比例，设置该比例后，系统内部会根据该比例压缩帧Buffer。 取值范围：[70, 100]。比如将“frame_buf_ratio”设置为80，则表示帧Buffer为原始大小的80%。 - 默认值为100，表示不压缩帧Buffer。 - 非取值范围内的值，表示不压缩帧Buffer。 [70, 100)范围内的值，表示需按比例压缩帧Buffer，可能会影响编码质量。建议在图片纹理不复杂、分辨率大于640 * 480等场景下，对于内存带宽有要求时，可以设置成[70, 100)范围内的值。

2.10.8.4.59 hi_venc_h265_attr

说明

定义H.265编码属性字典。

Atlas 推理系列产品，预留字典，暂不支持。

Atlas A2训练系列产品/Atlas 800I A2推理产品，预留字典，暂不支持。

定义

```
hi_venc_h265_attr = {"rcn_ref_share_buf_en": rcn_ref_share_buf_en, # RW; Range:[0, 1];Whether to enable the Share Buf of Rcn and Ref  
                    "frame_buf_ratio": frame_buf_ratio}
```

成员

成员名称	描述
rcn_ref_share_buf_en	是否使能参考帧和回写帧复用Buffer。预留参数，暂不支持。 取值范围：[0, 1]。1表示使能，0表示不使能。 静态属性。
frame_buf_ratio	帧Buffer比例，设置该比例后，系统内部会根据该比例压缩帧Buffer。 取值范围：[70, 100]。比如将“frame_buf_ratio”设置为80，则表示帧Buffer为原始大小的80%。 - 默认值为100，表示不压缩帧Buffer。 - 非取值范围内的值，表示不压缩帧Buffer。 [70, 100)范围内的值，表示需按比例压缩帧Buffer，可能会影响编码质量。建议在图片纹理不复杂、分辨率大于640 * 480等场景下，对于内存带宽有要求时，可以设置成[70, 100)范围内的值。

2.10.8.4.60 hi_venc_pic_recv_mode

说明

定义JPEG编码属性字典，暂不支持。

定义

```
HI_VENC_PIC_RECV_SINGLE = 0
HI_VENC_PIC_RECV_MULTI = 1
HI_VENC_PIC_RECV_BUTT = 2
```

2.10.8.4.61 hi_venc_jpeg_attr

说明

定义JPEG编码属性字典，暂不支持。

定义

```
hi_venc_jpeg_attr = {"dcf_en": dcf_en, # RW; Range:[0, 1]; support dcf
                    "mpf_cfg": hi_venc_mpf_cfg, # RW; Range:[0, 1]; config of Mpf
                    "recv_mode": hi_venc_pic_recv_mode} # RW; Config the receive mode;
```

2.10.8.4.62 hi_venc_prores_frame_rate

说明

PRORES帧率全局，暂不支持。

定义

```
HI_VENC_PRORES_FRAME_RATE_UNKNOWN = 0
HI_VENC_PRORES_FRAME_RATE_23_976 = 1
HI_VENC_PRORES_FRAME_RATE_24 = 2
HI_VENC_PRORES_FRAME_RATE_25 = 3
HI_VENC_PRORES_FRAME_RATE_29_97 = 4
HI_VENC_PRORES_FRAME_RATE_30 = 5
HI_VENC_PRORES_FRAME_RATE_50 = 6
HI_VENC_PRORES_FRAME_RATE_59_94 = 7
HI_VENC_PRORES_FRAME_RATE_60 = 8
HI_VENC_PRORES_FRAME_RATE_100 = 9
HI_VENC_PRORES_FRAME_RATE_119_88 = 10
HI_VENC_PRORES_FRAME_RATE_120 = 11
HI_VENC_PRORES_FRAME_RATE_BUTT = 12
```

2.10.8.4.63 hi_venc_prores_aspect_ratio

说明

PRORES宽高比格式，暂不支持。

定义

```
# the aspect ratio of PRORES
HI_VENC_PRORES_ASPECT_RATIO_UNKNOWN = 0
HI_VENC_PRORES_ASPECT_RATIO_SQUARE = 1
HI_VENC_PRORES_ASPECT_RATIO_4_3 = 2
HI_VENC_PRORES_ASPECT_RATIO_16_9 = 3
HI_VENC_PRORES_ASPECT_RATIO_BUTT = 4
```

2.10.8.4.64 hi_venc_prores_attr

说明

定义PRORES属性字典，暂不支持。

定义

```
hi_venc_prores_attr = {"identifier": [identifier],  
                       "frame_rate_code": hi_venc_prores_frame_rate,  
                       "aspect_ratio": hi_venc_prores_aspect_ratio}
```

2.10.8.4.65 hi_crop_info

说明

裁剪crop属性字典。

定义

```
hi_crop_info= {"enable": enable, # RW; Range:[0, 1]; Crop region enable  
               "rect": hi_rect} # RW; Crop region, note: s32X must be multi of 16
```

成员名称	描述
enable	是否开启裁剪功能。 取值范围： 0：不开启。 1：开启。
rect	裁剪起始位置与宽高信息。

2.10.8.4.66 hi_venc_scene_mode

说明

定义编码模式场景。

定义

```
HI_VENC_SCENE_0 = 0 # 摄像机不运动或周期性连续运动的场景，支持h.264/h.265  
HI_VENC_SCENE_1 = 1 # 高码率下运动场景，支持h.265  
HI_VENC_SCENE_2 = 2 # 暂不支持  
HI_VENC_SCENE_BUTT = 3
```

安防场景配置为“HI_VENC_SCENE_0”；自动驾驶、直播、游戏、动画、电影配置为“HI_VENC_SCENE_1”。

2.10.8.4.67 hi_venc_rc_param

说明

定义编码通道码率控制器的高级参数。

定义

```
hi_venc_rc_param = {"threshold_i": threshold_i[HI_VENC_TEXTURE_THRESHOLD_SIZE],
                    "threshold_p": threshold_p[HI_VENC_TEXTURE_THRESHOLD_SIZE],
                    "threshold_b": threshold_b[HI_VENC_TEXTURE_THRESHOLD_SIZE],
                    "direction": direction,
                    "row_qp_delta": row_qp_delta,
                    "first_frame_start_qp": first_frame_start_qp,
                    "scene_chg_detect": scene_chg_detect,
                    "h264_cbr_param": hi\_venc\_h264\_cbr\_param }
```

```
hi_venc_rc_param = {"threshold_i": threshold_i[HI_VENC_TEXTURE_THRESHOLD_SIZE],
                    "threshold_p": threshold_p[HI_VENC_TEXTURE_THRESHOLD_SIZE],
                    "threshold_b": threshold_b[HI_VENC_TEXTURE_THRESHOLD_SIZE],
                    "direction": direction,
                    "row_qp_delta": row_qp_delta,
                    "first_frame_start_qp": first_frame_start_qp,
                    "scene_chg_detect": scene_chg_detect,
                    "h264_vbr_param": hi\_venc\_h264\_vbr\_param}
```

```
hi_venc_rc_param = {"threshold_i": threshold_i[HI_VENC_TEXTURE_THRESHOLD_SIZE],
                    "threshold_p": threshold_p[HI_VENC_TEXTURE_THRESHOLD_SIZE],
                    "threshold_b": threshold_b[HI_VENC_TEXTURE_THRESHOLD_SIZE],
                    "direction": direction,
                    "row_qp_delta": row_qp_delta,
                    "first_frame_start_qp": first_frame_start_qp,
                    "scene_chg_detect": scene_chg_detect,
                    "h264_avbr_param": hi\_venc\_h264\_avbr\_param}
```

```
hi_venc_rc_param = {"threshold_i": threshold_i[HI_VENC_TEXTURE_THRESHOLD_SIZE],
                    "threshold_p": threshold_p[HI_VENC_TEXTURE_THRESHOLD_SIZE],
                    "threshold_b": threshold_b[HI_VENC_TEXTURE_THRESHOLD_SIZE],
                    "direction": direction,
                    "row_qp_delta": row_qp_delta,
                    "first_frame_start_qp": first_frame_start_qp,
                    "scene_chg_detect": scene_chg_detect,
                    "h264_qvbr_param": hi\_venc\_h264\_qvbr\_param}
```

```
hi_venc_rc_param = {"threshold_i": threshold_i[HI_VENC_TEXTURE_THRESHOLD_SIZE],
                    "threshold_p": threshold_p[HI_VENC_TEXTURE_THRESHOLD_SIZE],
                    "threshold_b": threshold_b[HI_VENC_TEXTURE_THRESHOLD_SIZE],
                    "direction": direction,
                    "row_qp_delta": row_qp_delta,
                    "first_frame_start_qp": first_frame_start_qp,
                    "scene_chg_detect": scene_chg_detect,
                    "h264_cvbr_param": hi\_venc\_h264\_cvbr\_param}
```

```
hi_venc_rc_param = {"threshold_i": threshold_i[HI_VENC_TEXTURE_THRESHOLD_SIZE],
                    "threshold_p": threshold_p[HI_VENC_TEXTURE_THRESHOLD_SIZE],
                    "threshold_b": threshold_b[HI_VENC_TEXTURE_THRESHOLD_SIZE],
                    "direction": direction,
                    "row_qp_delta": row_qp_delta,
                    "first_frame_start_qp": first_frame_start_qp,
                    "scene_chg_detect": scene_chg_detect,
                    "h265_cbr_param": hi\_venc\_h265\_cbr\_param}
```

```
hi_venc_rc_param = {"threshold_i": threshold_i[HI_VENC_TEXTURE_THRESHOLD_SIZE],
                    "threshold_p": threshold_p[HI_VENC_TEXTURE_THRESHOLD_SIZE],
                    "threshold_b": threshold_b[HI_VENC_TEXTURE_THRESHOLD_SIZE],
                    "direction": direction,
                    "row_qp_delta": row_qp_delta,
                    "first_frame_start_qp": first_frame_start_qp,
                    "scene_chg_detect": scene_chg_detect,
                    "h265_vbr_param": hi\_venc\_h265\_vbr\_param}
```

```
hi_venc_rc_param = {"threshold_i": threshold_i[HI_VENC_TEXTURE_THRESHOLD_SIZE],
                    "threshold_p": threshold_p[HI_VENC_TEXTURE_THRESHOLD_SIZE],
                    "threshold_b": threshold_b[HI_VENC_TEXTURE_THRESHOLD_SIZE],
                    "direction": direction,
                    "row_qp_delta": row_qp_delta,
```

```
"first_frame_start_qp": first_frame_start_qp,
"scene_chg_detect": scene_chg_detect,
"h265_avbr_param": hi\_venc\_h265\_avbr\_param}

hi_venc_rc_param = {"threshold_i": threshold_i[HI_VENC_TEXTURE_THRESHOLD_SIZE],
                    "threshold_p": threshold_p[HI_VENC_TEXTURE_THRESHOLD_SIZE],
                    "threshold_b": threshold_b[HI_VENC_TEXTURE_THRESHOLD_SIZE],
                    "direction": direction,
                    "row_qp_delta": row_qp_delta,
                    "first_frame_start_qp": first_frame_start_qp,
                    "scene_chg_detect": scene_chg_detect,
                    "h265_qvbr_param": hi\_venc\_h265\_qvbr\_param}

hi_venc_rc_param = {"threshold_i": threshold_i[HI_VENC_TEXTURE_THRESHOLD_SIZE],
                    "threshold_p": threshold_p[HI_VENC_TEXTURE_THRESHOLD_SIZE],
                    "threshold_b": threshold_b[HI_VENC_TEXTURE_THRESHOLD_SIZE],
                    "direction": direction,
                    "row_qp_delta": row_qp_delta,
                    "first_frame_start_qp": first_frame_start_qp,
                    "scene_chg_detect": scene_chg_detect,
                    "h265_cvbr_param": hi\_venc\_h265\_cvbr\_param}

hi_venc_rc_param = {"threshold_i": threshold_i[HI_VENC_TEXTURE_THRESHOLD_SIZE],
                    "threshold_p": threshold_p[HI_VENC_TEXTURE_THRESHOLD_SIZE],
                    "threshold_b": threshold_b[HI_VENC_TEXTURE_THRESHOLD_SIZE],
                    "direction": direction,
                    "row_qp_delta": row_qp_delta,
                    "first_frame_start_qp": first_frame_start_qp,
                    "scene_chg_detect": scene_chg_detect,
                    "mpeg_cbr_param": hi\_venc\_mjpeg\_cbr\_param}

hi_venc_rc_param = {"threshold_i": threshold_i[HI_VENC_TEXTURE_THRESHOLD_SIZE],
                    "threshold_p": threshold_p[HI_VENC_TEXTURE_THRESHOLD_SIZE],
                    "threshold_b": threshold_b[HI_VENC_TEXTURE_THRESHOLD_SIZE],
                    "direction": direction,
                    "row_qp_delta": row_qp_delta,
                    "first_frame_start_qp": first_frame_start_qp,
                    "scene_chg_detect": scene_chg_detect,
                    "mpeg_vbr_param": hi\_venc\_mjpeg\_vbr\_param}
```

成员

成员名称	描述
threshold_i	列表， I帧宏块级码率控制的Madi（用于度量当前帧的空域纹理复杂度）门限。 取值范围：[0, 255]。 默认值：[0, 0, 0, 0, 3, 3, 5, 5, 8, 8, 8, 15, 15, 20, 25, 25]。 减方向的数值设置为0，表示关闭当前级；加方向的数值设置为255，表示关闭当前级。
threshold_p	列表， P帧宏块级码率控制的Madi门限。 取值范围：[0, 255]。 默认值：[0, 0, 0, 0, 3, 3, 5, 5, 8, 8, 8, 15, 15, 20, 25, 25]。 减方向的数值设置为0，表示关闭当前级；加方向的数值设置为255，表示关闭当前级。

成员名称	描述
threshold_b	列表， B帧宏块级码率控制的Madi门限。 取值范围： [0, 255]。 默认值： [0, 0, 0, 0, 3, 3, 5, 5, 8, 8, 8, 15, 15, 20, 25, 25]。 减方向的数值设置为0，表示关闭当前级；加方向的数值设置为255，表示关闭当前级。
direction	整型，在基于纹理宏块级码率控制时，用于控制加减方向。。 取值范围： [0, 16]。 默认值： 8。 举例， direction = 7表示threshold_i、 threshold_p、 threshold_b的前7个参数用于 QP减方向；后9个参数用于 QP加方向。
row_qp_delta	行级码率控制调节幅度是一帧内行级调节的最大范围，其中行级以宏块行为单位。调节幅度越大，允许行级调整的QP范围越大，码率越平稳。对于图像复杂度分布不均匀的场景，行级码率控制调节幅度设置过大会带来图像质量不均匀。 取值范围： [0, 10]，设置为0表示关闭基于行的宏块级码率控制。 默认值： 1。
first_frame_start_qp	整型，设置第一帧的起始QP值，CBR/VBR/AVBR/QVBR/CVBR模式下，该参数有效。此处第一帧的含义是：通道创建，Gop 模式切换，RC 模式切换，或分辨率切换后，序列的第一个 IDR帧。需要注意如果第一帧编码完成后判定为重编，重编的帧不是第一帧，不受“first_frame_start_qp”的约束。 取值范围： [MinIQP, MaxIQP]和-1。“MinIQP”参数值、“MaxIQP”参数值在码率控制模式的高级参数字典内设置。 默认值： -1。如果为-1，则第一帧的起始QP由编码器内部计算，如果为其它合法值则由用户通过该参数设置第一帧的起始QP。 在低码率、少量运动场景下，如果编码效果不好，可以通过降低该参数值来提升图像质量和效果。 例如，在“352*288分辨率，帧率25fps，GOP=50”场景下，将“first_frame_start_qp”设置为32，同时将max_reencode_times参数设置为0（表示不重编码，该参数是以下码率控制模式高级参数之一）。

成员名称	描述
scene_chg_detect	整型，场景检测相关的控制参数，自适应检测当前编码场景是否发生变化，使能后会改善场景切换时的码率波动，VBR/AVBR/QVBR/CVBR 有效。使能场景切换检测时可使能自适应插入 IDR 帧，在场景切换时编码 IDR 帧，需要注意关闭重编后检测场景切换会失效。该参数建议与编码场景模式的SCENE_0模式配合使用。
h264_cbr_param	字典，H.264通道CBR（Constant Bit Rate）码率控制模式高级参数。
h264_vbr_param	字典，H.264通道VBR（Variable Bit Rate）码率控制模式高级参数。
h264_avbr_param	字典，H.264通道AVBR（Adaptive Variable Bit Rate）码率控制模式高级参数。
h264_qvbr_param	字典，H.264通道QVBR（Quality Variable Bit Rate）码率控制模式高级参数。
h264_cvbr_param	字典，H.264通道CVBR（Constrained Variable Bit Rate）码率控制模式高级参数。
h265_cbr_param	字典，H.265通道CBR（Constant Bit Rate）码率控制模式高级参数。
h265_vbr_param	字典，H.265通道VBR（Variable Bit Rate）码率控制模式高级参数。
h265_avbr_param	字典，H.265通道AVBR（Adaptive Variable Bit Rate）码率控制模式高级参数。
h265_qvbr_param	字典，H.265通道QVBR（Quality Variable Bit Rate）码率控制模式高级参数。
h265_cvbr_param	字典，H.265通道CVBR（Constrained Variable Bit Rate）码率控制模式高级参数。
mjpeg_cbr_param	字典，MJPEG通道CBR（Constant Bit Rate）码率控制模式高级参数。预留参数，暂不支持。
mjpeg_vbr_param	字典，MJPEG通道VBR（Variable Bit Rate）码率控制模式高级参数。预留参数，暂不支持。

2.10.8.4.68 hi_venc_h264_cbr_param

说明

定义H.264协议编码通道CBR码率控制模式的高级参数。

定义

```
hi_venc_h264_cbr_param = {"max_i_proportion": max_i_proportion,
                           "min_i_proportion": min_i_proportion,
                           "max_qp": max_qp,
```

```
"min_qp": min_qp,  
"max_i_qp": max_i_qp,  
"min_i_qp": min_i_qp,  
"max_reencode_times": max_reencode_times,  
"qpmap_en": qpmap_en}
```

成员

成员名称	描述
min_i_proportion	整型，最小IP帧比例，预留参数，暂不支持。
max_i_proportion	最大IP帧比例。 取值范围：[1, 100]。 该参数默认值为100。
max_qp	整型，帧最大QP，用于钳位质量。 取值范围：[0, 51]。 默认值：51。
min_qp	整型，帧最小QP，用于钳位码率波动。 取值范围：[0, max_qp]。 默认值：10。
max_reencode_times	整型，每帧重编码次数。0表示不进行重编码。 取值范围：[0, 3]。 默认值：2。
min_i_qp	整型，I帧的最小QP。用于控制I帧的最大bits数。 取值范围：[0, max_i_qp]。 默认值：10。
max_i_qp	整型，I帧的最大QP。用于控制I帧的最小bits数。 取值范围：[0, 51]。 默认值：51。
qpmap_en	整型，预留参数，暂不支持。

2.10.8.4.69 hi_venc_h264_vbr_param

说明

定义H.264协议编码通道VBR码率控制模式的高级参数。

定义

```
hi_venc_h264_vbr_param = {"chg_pos": chg_pos,  
                           "max_i_proportion": max_i_proportion,  
                           "min_i_proportion": min_i_proportion,  
                           "max_reencode_times": max_reencode_times,  
                           "qpmap_en": qpmap_en,  
                           "max_qp": max_qp,  
                           "min_qp": min_qp,
```

```
"max_i_qp": max_i_qp,  
"min_i_qp": min_i_qp}
```

成员

- “max_qp”、“min_qp”用于控制图像的质量范围，“max_bit_rate”用于钳位码率统计时间内的最大编码码率，“chg_pos”用于控制开始调整QP的码率基准线。
- 当编码码率大于max_bit_rate * chg_pos时，图像QP会逐步向“max_qp”调整，如果图像QP达到“max_qp”，QP会被钳位到最大值，“max_bit_rate”的钳位效果失效，编码码率有可能会超出“max_bit_rate”。
 - 当编码码率小于max_bit_rate * chg_pos时，图像QP会逐步向“min_qp”调整，如果图像QP达到“min_qp”，此时编码的码率已经达到最大值，而且图像质量最好。

说明

“max_bit_rate”属性在创建VENC通道时设置，是hi_venc_chn_attr["rc_attr"]内的成员变量。

成员名称	描述
chg_pos	整型，VBR开始调整QP时的码率相对于最大码率的比例。 取值范围：[50, 100]。 默认值：90。
min_i_proportion	整型，预留参数，暂不支持。
max_i_proportion	最大IP帧码率的比值。 取值范围：[1,100]。 该参数默认值为100。
max_reencode_times	整型，每帧重编码次数。0 表示不进行重编码。 取值范围：[0, 3]。 默认值：2。
qpmap_en	整型，预留参数，暂不支持。
max_qp	整型，P、B帧的最大QP。 取值范围：[0, 51]。 默认值：51。
min_qp	整型，P、B帧的最小QP。 取值范围：[0, max_qp]。 默认值：24。
max_i_qp	整型，I帧的最大QP。 取值范围：[0, 51]。 默认值：51。
min_i_qp	整型，I帧的最小QP。 取值范围：[0, max_i_qp]。 默认值：24。

2.10.8.4.70 hi_venc_h264_avbr_param

说明

定义H.264协议编码通道AVBR码率控制模式的高级参数。

定义

```
hi_venc_h264_avbr_param = {"chg_pos": chg_pos,
                            "max_i_proportion": max_i_proportion,
                            "min_i_proportion": min_i_proportion,
                            "max_reencode_times": max_reencode_times,
                            "qpmap_en": qpmap_en,
                            "min_still_percent": min_still_percent,
                            "max_still_qp": max_still_qp,
                            "min_still_psnr": min_still_psnr,
                            "max_qp": max_qp,
                            "min_qp": min_qp,
                            "max_i_qp": max_i_qp,
                            "min_i_qp": min_i_qp,
                            "min_qp_delta": min_qp_delta,
                            "motion_sensitivity": motion_sensitivity,
                            "save_bitrate_en": save_bitrate_en}
```

成员

“max_bit_rate”表示运动场景下的最大码率， $\text{max\_bit\_rate} * \text{chg\_pos} * \text{min\_still\_percent}$ 表示静止情况下的最小码率。

根据运动程度的不同，目标码率会在最大码率和最小码率间调整。“max_qp”，“min_qp”用于控制图像的质量范围，码率控制以QP钳位为最高优先级，超出“min_qp”，“max_qp”范围内码率控制将失效。

 说明

“max_bit_rate”属性在创建VENC通道时设置，是hi_venc_chn_attr["rc_attr"]内的成员变量。

成员名称	描述
chg_pos	整型，AVBR开始调整QP时的码率相对于最大码率的比例。 取值范围：[50, 100]。 默认值：65。
min_i_proportion	整型，预留参数，暂不支持。
max_i_proportion	整型，最大IP帧码率的比值。 取值范围：[1,100]。 默认值：100。
max_reencode_times	整型，每帧重编码次数。0 表示不进行重编码。 取值范围：[0, 3]。 默认值：2。
qpmap_en	整型，预留参数，暂不支持。

成员名称	描述
min_still_percent	整型，静止状态下目标码率的最小百分比。此变量设置为100，AVBR将不会在判别为静止时主动调低目标码率。 取值范围：[5, 100]。 默认值：25。
max_still_qp	整型，静止场景I帧QP的最大值。 取值范围：[MinIqp, MaxIqp]。 默认值：35。
min_still_psnr	整型，预留参数，暂不支持。
max_qp	整型，P、B帧的最大QP。 取值范围：[0, 51]。 默认值：51。
min_qp	整型，P、B帧的最小QP。 取值范围：[0, max_qp]。 默认值：24。
max_i_qp	整型，I帧的最大QP。 取值范围：[0, 51]。 默认值：51。
min_i_qp	整型，I帧的最小QP。 取值范围：[0, max_i_qp]。 默认值：24。
min_qp_delta	整型，帧级QP最小值和CU级QP最小值的差值。 I帧，帧级QP最小值 = min_qp_delta + min_i_qp P、B帧，帧级QP最小值 = min_qp_delta + min_qp 取值范围：[0, 4]； 默认值：0。
motion_sensitivity	整型，运动敏感度。 取值范围：[0, 100]。 默认值：100。
save_bitrate_en	整型，预留参数，暂不支持。

2.10.8.4.71 hi_venc_h264_qvbr_param

说明

定义H.264协议编码通道QVBR码率控制模式的高级参数。

定义

```
hi_venc_h264_qvbr_param = {"max_i_proportion": max_i_proportion,
                            "min_i_proportion": min_i_proportion,
                            "max_reencode_times": max_reencode_times,
                            "qpmap_en": qpmap_en,
                            "max_qp": max_qp,
                            "min_qp": min_qp,
                            "max_i_qp": max_i_qp,
                            "min_i_qp": min_i_qp,
                            "max_bit_percent": max_bit_percent,
                            "min_bit_percent": min_bit_percent,
                            "max_psnr_fluctuate": max_psnr_fluctuate,
                            "min_psnr_fluctuate": min_psnr_fluctuate}
```

成员

当实时统计的PSNR小于“max_psnr_fluctuate”时，适当增加目标码率，最大码率 = target_bit_rate * max_bit_percent。

当实时统计的PSNR大于“max_psnr_fluctuate”时，适当减小目标码率，最小码率 = target_bit_rate * min_bit_percent。

根据当前PSNR的不同，目标码率会在最大码率和最小码率间调整，当PSNR值范围超过[min_psnr_fluctuate - 4, max_psnr_fluctuate + 4] ∩ [20, 40]时，PSNR不再起作用，码率会在最大码率和最小码率间调整。

“max_qp”，“min_qp”用于控制图像的质量范围，码率控制以QP钳位为最高优先级，超出“min_qp”，“max_qp”范围内码率控制将失效。

码率浮动上下限的优先级高于PSNR的优先级，例如，当码率浮动到上限依然不能满足PSNR要求，则码率不会再继续上调。

说明

“target_bit_rate”属性在创建VENC通道时设置，是hi_venc_chn_attr["rc_attr"]内的成员变量。

成员名称	描述
min_i_proportion	整型，预留参数，暂不支持。
max_i_proportion	整型，最大IP帧码率的比值。 取值范围：[1, 100]。 默认值：100。
max_reencode_times	整型，每帧重编码次数。0表示不进行重编码。 取值范围：[0, 3]。 默认值：2。
qpmap_en	整型，预留参数，暂不支持。
max_qp	整型，P、B帧的最大QP。 取值范围：[MinQp, 51]。 默认值：51。

成员名称	描述
min_qp	整型，P、B帧的最小QP。 取值范围：[0, 51]。 默认值：16。
max_i_qp	整型，I帧的最大QP。 取值范围：[MinIqp, 51]。 默认值：51。
min_i_qp	整型，I帧的最小QP。 取值范围：[0, 51]。 默认值：16。
max_bit_percent	整型，码率百分比上限。 取值范围：[min_bit_percent, 180]。 默认值：110。
min_bit_percent	整型，码率百分比下限。 取值范围：[30, 180]。 默认值：45。
max_psnr_fluctuate	整型，PSNR上限。 取值范围：[min_psnr_fluctuate, 40]。 默认值：40。
min_psnr_fluctuate	整型，PSNR下限。 取值范围：[18, 40]。 默认值：23。

2.10.8.4.72 hi_venc_h264_cvbr_param

说明

定义H.264协议编码通道CVBR码率控制模式的高级参数。

定义

```
hi_venc_h264_cvbr_param = {"max_i_proportion": max_i_proportion,
    "min_i_proportion": min_i_proportion,
    "max_reencode_times": max_reencode_times,
    "qpmap_en": qpmap_en,
    "max_qp": max_qp,
    "min_qp": min_qp,
    "max_i_qp": max_i_qp,
    "min_i_qp": min_i_qp,
    "min_qp_delta": min_qp_delta,
    "max_qp_delta": max_qp_delta,
    "extra_bit_percent": extra_bit_percent,
    "long_term_stats_time_unit": long_term_stats_time_unit,
    "save_bitrate_en": save_bitrate_en}
```

成员

成员名称	描述
min_i_proportion	整型，预留参数，暂不支持。
max_i_proportion	整型，最大IP帧码率的比值。 取值范围：[1, 100]。 默认值：100。
max_reencode_times	整型，每帧重编码次数。0 表示不进行重编码。 取值范围：[0, 3]。 默认值：2。
qpmap_en	整型，预留参数，暂不支持。
max_qp	整型，P、B帧的最大QP。 取值范围：[min_qp, 51]。 默认值：47。
min_qp	整型，P、B帧的最小QP。 取值范围：[0, 51]。 默认值：22。
max_i_qp	整型，I帧的最大QP。 取值范围：[min_i_qp, 51]。 默认值：47。
min_i_qp	整型，I帧的最小QP。 取值范围：[0, 51]。 默认值：20。
min_qp_delta	整型，帧级QP最小值和CU级QP最小值的差值。 I帧，帧级QP最小值 = min_qp_delta + min_i_qp P、B帧，帧级QP最小值 = min_qp_delta + min_qp 取值范围：[0, 4]。 默认值：0。
max_qp_delta	整型，帧级QP最大值和CU级QP最大值的差值。 I帧，帧级QP最大值 = max_i_qp - max_qp_delta P、B帧，帧级QP最大值 = max_qp - max_qp_delta 取值范围：[0, 4]。 默认值：0。

成员名称	描述
extra_bit_percent	整型，编码器输出码流最大透支bit数百分比。 在码率不足时，为保证图像质量，编码器会通过透支一定的bit数以提升图像质量，这部分透支的码率会在编码压力较小时进行偿还。 取值范围：[0, 1000]。 默认值：5。
long_term_stats_time_unit	整型，码率长期统计时间参数（long_term_stats_time）的单位，本变量单位是秒（s），如配置long_term_stats_time_unit = 60，long_term_stats_time = 3，代表长期统计时间为3 分钟。 取值范围：[1, 1800]。 默认值：60。
save_bitrate_en	整型，预留参数，暂不支持。

2.10.8.4.73 hi_venc_h265_vbr_param

说明

定义H.265协议编码通道VBR码率控制模式的高级参数。

定义

```
hi_venc_h264_cvbr_param = {"chg_pos": chg_pos,
    "max_i_proportion": max_i_proportion,
    "min_i_proportion": min_i_proportion,
    "max_reencode_times": max_reencode_times,
    "max_qp": max_qp,
    "min_qp": min_qp,
    "max_i_qp": max_i_qp,
    "min_i_qp": min_i_qp,
    "qpmap_en": qpmap_en,
    "qpmap_mode": hi_venc_rc_qpmap_mode}
```

成员

“max_qp”、“min_qp”用于控制图像的质量范围，“max_bit_rate”用于钳位码率统计时间内的最大编码码率，“chg_pos”用于控制开始调整QP的码率基准线。

当编码码率大于max_bit_rate * chg_pos时，图像QP会逐步向“max_qp”调整，如果图像QP达到“max_qp”，QP会被钳位到最大值，“max_bit_rate”的钳位效果失效，编码码率有可能会超出max_bit_rate。

当编码码率小于max_bit_rate * chg_pos时，图像QP会逐步向“min_qp”调整，如果图像QP达到“min_qp”，此时编码的码率已经达到最大值，而且图像质量最好。

 说明

“max_bit_rate”属性在创建VENC通道时设置，是hi_venc_chn_attr["rc_attr"]内的成员变量。

成员名称	描述
chg_pos	整型，开始调整QP时的码率相对于最大码率的比例。 取值范围：[50, 100]。 默认值：90。
min_i_proportion	整型，预留参数，暂不支持。
max_i_proportion	最大IP帧码率的比值。 取值范围：[1, 100]。 该参数默认值为100。
max_reencode_times	整型，每帧重编码次数。0 表示不进行重编码。 取值范围：[0, 3]。 默认值：2。
max_qp	整型，P帧的最大QP。 取值范围：[0, 51]。 默认值：51。
min_qp	整型，P帧的最小QP。 取值范围：[0, max_qp]。 默认值：24。
max_i_qp	整型，I帧的最大QP。 取值范围：[0, 51]。 默认值：51。
min_i_qp	整型，I帧的最小QP。 取值范围：[0, max_i_qp]。 默认值：24。
qpmap_en	整型，预留参数，暂不支持。
qpmap_en	整型，预留参数，暂不支持。

2.10.8.4.74 hi_venc_h265_avbr_param

说明

定义H.265协议编码通道AVBR码率控制模式的高级参数。

定义

```
hi_venc_h264_cvbr_param = {"chg_pos": chg_pos,
                           "max_i_proportion": max_i_proportion,
                           "min_i_proportion": min_i_proportion,
                           "max_reencode_times": max_reencode_times,
                           "min_still_percent": min_still_percent,
                           "max_still_qp": max_still_qp,
                           "min_still_psnr": min_still_psnr,
```

```
"max_qp": max_qp,  
"min_qp": min_qp,  
"max_i_qp": max_i_qp,  
"min_i_qp": min_i_qp,  
"min_qp_delta": min_qp_delta,  
"motion_sensitivity": motion_sensitivity,  
"qpmap_en": qpmap_en,  
"qpmap_mode": hi_venc_rc_qpmap_mode}
```

成员

“max_bit_rate”表示运动场景下的最大码率， $\text{max\_bit\_rate} * \text{chg\_pos} * \text{min\_still\_percent}$ 表示静止情况下的最小码率。

根据运动程度的不同，目标码率会在最大码率和最小码率间调整。“max_qp”，“min_qp”用于控制图像的质量范围，码率控制以QP钳位为最高优先级，超出“min_qp”，“max_qp”范围内码率控制将失效。

说明

“max_bit_rate”属性在创建VENC通道时设置，是hi_venc_chn_attr["rc_attr"]内的成员变量。

成员名称	描述
chg_pos	整型，开始调整QP时的码率相对于最大码率的比例。 取值范围：[50, 100]。 默认值：65。
min_i_proportion	整型，预留参数，暂不支持。
max_i_proportion	整型，最大IP帧码率的比值。 取值范围：[1, 100]。 默认值：100。
max_reencode_times	整型，每帧重编码次数。0表示不进行重编码。 取值范围：[0, 3]。 默认值：2。
min_still_percent	整型，静止状态下目标码率的最小百分比。此变量设置为100，AVBR将不会在判别为静止时主动调低目标码率。 取值范围：[5, 100]。 默认值：25。
max_still_qp	整型，静止场景I帧QP的最大值。 取值范围：[MinIQp, MaxIQp]。 默认值：35。
min_still_psnr	整型，预留参数，暂不支持。
max_qp	P帧的最大QP。 取值范围：[0, 51]。 默认值：51。

成员名称	描述
min_qp	整型，P帧的最小QP。 取值范围：[0, max_qp]。 默认值：24。
max_i_qp	整型，I帧的最大QP。 取值范围：[0, 51]。 默认值：51。
min_i_qp	整型，I帧的最小QP。 取值范围：[0, max_i_qp]。 默认值：24。
min_qp_delta	整型，帧级QP最小值和CU级QP最小值的差值。 I帧：FrameLevelMinQp = min_qp_delta + min_i_qp P、B 帧：FrameLevelMinQp = min_qp_delta + min_qp 取值范围：[0, 4]。 默认值：0。
motion_sensitivity	整型，运动敏感度。 取值范围：[0, 100]。 默认值：100。
qpmmap_en	整型，预留参数，暂不支持。
qpmmap_en	整型，预留参数，暂不支持。

2.10.8.4.75 hi_venc_h265_qvbr_param

说明

定义H.265协议编码通道QVBR码率控制模式的高级参数。

定义

```
hi_venc_h265_qvbr_param = {"max_i_proportion": max_i_proportion,
    "min_i_proportion": min_i_proportion,
    "max_reencode_times": max_reencode_times,
    "qpmmap_en": qpmmap_en,
    "qpmmap_mode": hi_venc_rc_qpmmap_mode,
    "max_qp": max_qp,
    "min_qp": min_qp,
    "max_i_qp": max_i_qp,
    "min_i_qp": min_i_qp,
    "max_bit_percent": max_bit_percent,
    "min_bit_percent": min_bit_percent,
    "max_psnr_fluctuate": max_psnr_fluctuate,
    "min_psnr_fluctuate": min_psnr_fluctuate}
```

成员

当实时统计的PSNR小于“max_psnr_fluctuate”时，适当增加目标码率，最大码率 = target_bit_rate * max_bit_percent。

当实时统计的PSNR大于“max_psnr_fluctuate”时，适当减小目标码率，最小码率 = target_bit_rate * min_bit_percent。

根据当前PSNR的不同，目标码率会在最大码率和最小码率间调整，当PSNR值范围超过[min_psnr_fluctuate - 4, max_psnr_fluctuate + 4] ∩ [20, 40]时，PSNR不再起作用，码率会在最大码率和最小码率间调整。

“max_qp”，“min_qp”用于控制图像的质量范围，码率控制以QP钳位为最高优先级，超出“min_qp”，“max_qp”范围内码率控制将失效。

码率浮动上下限的优先级高于PSNR的优先级，例如，当码率浮动到上限依然不能满足PSNR要求，则码率不会再继续上调。

说明

“target_bit_rate”属性在创建VENC通道时设置，是hi_venc_chn_attr["rc_attr"]内的成员变量。

成员名称	描述
min_i_proportion	整型，预留参数，暂不支持。
max_i_proportion	整型，最大IP帧码率的比值。 取值范围：[1,100]。 默认值：100。
max_reencode_times	整型，每帧重编码次数。0表示不进行重编码。 取值范围：[0, 3]。 默认值：2。
qpmap_en	整型，预留参数，暂不支持。
qpmap_en	整型，预留参数，暂不支持。
max_qp	整型，P、B帧的最大QP。 取值范围：[MinQp, 51]。 默认值：51。
min_qp	整型，P、B帧的最小QP。 取值范围：[0, 51]。 默认值：16。
max_i_qp	整型，I帧的最大QP。 取值范围：[MinIQp, 51]。 默认值：51。
min_i_qp	整型，I帧的最小QP。 取值范围：[0, 51]。 默认值：16。

成员名称	描述
max_bit_percent	整型，码率百分比上限。 取值范围：[min_bit_percent, 180]。 默认值：110。
min_bit_percent	整型，码率百分比下限。 取值范围：[30, 180]。 默认值：45。
max_psnr_fluctuate	整型，PSNR上限。 取值范围：[min_psnr_fluctuate, 40]。 默认值：40。
min_psnr_fluctuate	整型，PSNR下限。 取值范围：[18, 40]。 默认值：23。

2.10.8.4.76 hi_venc_h265_cvbr_param

说明

定义H.265协议编码通道CVBR码率控制模式的高级参数。

定义

```
hi_venc_h265_cvbr_param = {"max_i_proportion": max_i_proportion,
                           "min_i_proportion": min_i_proportion,
                           "max_reencode_times": max_reencode_times,
                           "qpmap_en": qpmap_en,
                           "qpmap_mode": hi_venc_rc_qpmap_mode,
                           "max_qp": max_qp,
                           "min_qp": min_qp,
                           "max_i_qp": max_i_qp,
                           "min_i_qp": min_i_qp,
                           "min_qp_delta": min_qp_delta,
                           "max_qp_delta": max_qp_delta,
                           "extra_bit_percent": extra_bit_percent,
                           "long_term_stats_time_unit": long_term_stats_time_unit}
```

成员

成员名称	描述
min_i_proportion	整型，预留参数，暂不支持。
max_i_proportion	整型，最大IP帧码率的比值。 取值范围：[1, 100]。 默认值：100。

成员名称	描述
max_reencode_times	整型，每帧重编码次数。0表示不进行重编码。 取值范围：[0, 3]。 默认值：2。
qpmap_en	整型，预留参数，暂不支持。
qpmap_en	整型，预留参数，暂不支持。
max_qp	整型，P、B帧的最大QP。 取值范围：[MinQp, 51]。 默认值：47。
min_qp	整型，P、B帧的最小QP。 取值范围：[0, 51]。 默认值：22。
max_i_qp	整型，I帧的最大QP。 取值范围：[MinIQp, 51]。 默认值：47。
min_i_qp	整型，I帧的最小QP。 取值范围：[0, 51]。 默认值：20。
min_qp_delta	整型，帧级QP最小值和CU级QP最小值的差值。 I 帧：FrameLevelMinQp = min_qp_delta + min_i_qp P、B帧：FrameLevelMinQp = min_qp_delta + min_qp 取值范围：[0, 4]。 默认值：0。
max_qp_delta	整型，帧级QP最大值和CU级QP最大值的差值。 I帧，帧级QP最大值 = max_i_qp - max_qp_delta P、B帧，帧级QP最大值 = max_qp - max_qp_delta 取值范围：[0, 4]。 默认值：0。
extra_bit_percent	整型，编码器输出码流最大透支bit数百分比。 在码率不足时，为保证图像质量，编码器会通过透支一定的bit数以提升图像质量，这部分透支的码率会在编码压力较小时进行偿还。 取值范围[0, 1000]。 默认值：5。

成员名称	描述
long_term_stats_time_unit	整型，码率长期统计时间参数（long_term_stats_time）的单位，本变量单位是秒（s），如配置long_term_stats_time_unit = 60，long_term_stats_time = 3，代表长期统计时间为3分钟。 取值范围：[1, 1800]。 默认值：60。

2.10.8.4.77 hi_venc_h265_cbr_param

说明

定义H.265协议编码通道CBR码率控制模式的高级参数。

定义

```
hi_venc_h265_qvbr_param = {"max_i_proportion": max_i_proportion,
                           "min_i_proportion": min_i_proportion,
                           "max_qp": max_qp,
                           "min_qp": min_qp,
                           "max_i_qp": max_i_qp,
                           "min_i_qp": min_i_qp,
                           "max_reencode_times": max_reencode_times,
                           "qpmap_en": qpmap_en,
                           "qpmap_mode": hi_venc_rc_qpmap_mode}
```

成员

成员名称	描述
min_i_proportion	整型，预留参数，暂不支持。
max_i_proportion	最大IP帧比例。 取值范围：[1, 100]。 该参数默认值为100。
max_qp	整型，帧最大QP，用于钳位质量。 取值范围：[0, 51]。 默认值：51。
min_qp	整型，帧最小QP，用于钳位码率波动。 取值范围：[0, max_qp]。 默认值：10。
max_i_qp	整型，I帧的最大QP。用于控制I帧的最小bits数。 取值范围：[0, 51]。 默认值：51。

成员名称	描述
min_i_qp	整型，I帧的最小QP。用于控制I帧的最大bits数。 取值范围：[0, max_i_qp]。 默认值：10。
max_reencode_times	整型，每帧重编码次数。0表示不进行重编码。 取值范围：[0, 3]。 默认值：2。
qpmap_en	整型，预留参数，暂不支持。
qpmap_en	整型，预留参数，暂不支持

2.10.8.4.78 hi_venc_scene_chg_detect

说明

定义编码场景检测控制参数。

定义

```
hi_venc_scene_chg_detect = {"detect_scene_chg_en": detect_scene_chg_en,  
                             "adapt_insert_idr_frame_en": adapt_insert_idr_frame_en}
```

成员

成员名称	描述
detect_scene_chg_en	整型，是否使能场景切换检测。 取值范围如下： - HI_TRUE：是，默认为该值 - HI_FALSE：否
adapt_insert_idr_frame_en	整型，是否使能自适应插入IDR帧。 取值范围如下： - HI_TRUE：是 - HI_FALSE：否，默认为该值

2.10.8.4.79 hi_venc_huffman_dc_table

说明

用于定义指定JPEG编码通道的亮度DC huffman编码表参数的数据结构。

定义

```
hi_venc_huffman_dc_table = {"dc_bits": dc_bits[16], # RW; Range:[0, 255]; len: 16  
                             "dc_value": dc_value[12]} # RW; Range:[0, 255];len 12
```

成员

成员名称	描述
dc_bits[]	列表，“dc_bits”数组存放的是长度从1bit到16bit的码字分别对应的符号个数。
dc_value[]	列表，“dc_value”数据存放的是码字对应的符号值。

2.10.8.4.80 hi_venc_huffman_ac_table

说明

用于定义指定JPEGE编码通道的色度AC huffman编码表参数的数据结构。

定义

```
hi_venc_huffman_ac_table = {"ac_bits": ac_bits[16],  
                             "ac_value": ac_value[162]}
```

成员

成员名称	描述
ac_bits	列表，“ac_bits”数组存放的是长度从1bit到16bit的码字分别对应的符号个数。
ac_value	列表，“ac_value”数据存放的是码字对应的符号值。

2.10.8.4.81 hi_venc_jpeg_huffman_param

说明

用于定义指定JPEGE编码通道的huffman编码高级参数的数据结构。

定义

```
hi_venc_jpeg_huffman_param = {"dc_tables": hi\_venc\_huffman\_dc\_table[3], # y, Cb, Cr  
                              "ac_tables": hi\_venc\_huffman\_ac\_table[3], # y, Cb, Cr  
                              "reserved": reserved[2]}
```

成员

成员名称	描述
dc_tables	列表，亮度DC huffman编码表数据。 Cb分量和CR分量需保持一致。
ac_tables	列表，色度AC huffman编码表数据。 Cb分量和CR分量需保持一致。
reserved	列表，预留属性，为保证后续版本兼容性，请务必传入[0, 0]进行清零初始化，在代码中必须避免显示对“reserved”字段进行访问。

2.10.8.4.82 hi_venc_mpf_cfg

说明

预留字典，暂不支持。

定义

```
hi_venc_mpf_cfg = {"large_thumbnail_num": large_thumbnail_num,  
                  "large_thumbnail_size": hi_video_size[HI_VENC_MAX_MPF_NUM]}
```

2.10.8.4.83 hi_venc_mjpeg_cbr_param

说明

预留字典，暂不支持。

定义

```
hi_venc_mjpeg_cbr_param = {"max_qfactor": max_qfactor,  
                           "min_qfactor": min_qfactor}
```

2.10.8.4.84 hi_venc_mjpeg_vbr_param

说明

预留字典，暂不支持。

定义

```
hi_venc_mjpeg_vbr_param = {"chg_pos": chg_pos,  
                           "max_qfactor": max_qfactor,  
                           "min_qfactor": min_qfactor}
```

2.10.8.4.85 hi_venc_roi_attr

定义编码感兴趣区域信息。

定义

```
hi_venc_roi_attr = {"idx": idx,  
                    "enable": enable,  
                    "is_abs_qp": is_abs_qp,  
                    "qp": qp,  
                    "rect": hi_rect}
```

成员

成员名称	描述
idx	整型，ROI区域的索引，系统支持的索引范围为[0, 7]，不支持超出这个范围的索引。 系统支持每个通道可设置8个ROI区域，系统内部按照0 ~ 7的索引号对ROI区域进行管理。ROI区域之间可以互相叠加，且当发生叠加时，ROI区域之间的优先级按照索引号0 ~ 7依次提高。
enable	整型，是否使能这个ROI区域。 Cb分量和Cr分量需保持一致。
is_abs_qp	整型，表示ROI区域的QP模式，指定当前的ROI区域采用绝对QP方式或是相对QP。 - HI_FALSE：相对QP - HI_TURE：绝对QP
qp	整型，QP值。 - 当QP模式为“HI_FALSE”时，“qp”为QP偏移，“qp”范围[-51, 51]。 - 当QP模式为“HI_TRUE”时，“qp”为宏块QP值，“qp”范围[0, 51]。
rect	字典，指定当前的ROI区域的位置坐标和区域的大小。 ROI区域的起始点坐标必须在图像范围内，且必须16对齐。 ROI区域必须在图像范围内，ROI区域的长宽必须是16对齐并且大于0。即，x、y、width、height必须是16对齐。

2.10.8.4.86 hi_venc_ref_param

说明

定义H.264/H.265编码的高级跳帧参考参数。

定义

```
hi_venc_ref_param = {"base": base,  
                     "enhance": enhance,  
                     "pred_en": pred_en}
```

成员

成员名称	描述
base	整型，base层的周期。 取值范围：(0, +∞)。
enhance	整型，enhance层的周期。 取值范围：[0, 255]。
pred_en	整型，代表base层的帧是否被base层其他帧用作参考。当“pred_en”设置为“HI_FALSE”时，等同于“base”设置为无限大，base层的所有帧都参考IDR帧。

2.10.8.4.87 hi_venc_h264_vui

说明

定义H.264协议编码通道VUI字典。

定义

```
hi_venc_h264_vui = {"vui_aspect_ratio": hi_venc_vui_aspect_ratio,  
                    "vui_time_info": hi_venc_vui_aspect_ratio,  
                    "vui_video_signal": hi_venc_vui_video_signal,  
                    "vui_bitstream_restric": hi_venc_vui_bitstream_restric}
```

成员

成员名称	描述
vui_aspect_ratio	字典，预留参数，具体含义请参见H.264协议。
vui_time_info	字典，VUI字段中时钟计数相关参数。
vui_video_signal	字典，VUI字段中视频信号相关参数。
vui_bitstream_restric	字典，预留参数，具体含义请参见H.264协议。

2.10.8.4.88 hi_venc_h265_vui

说明

定义H.265协议编码通道VUI字典。

定义

```
hi_venc_h265_vui = {"vui_aspect_ratio": hi_venc_vui_aspect_ratio,  
                    "vui_time_info": hi_venc_vui_aspect_ratio,  
                    "vui_video_signal": hi_venc_vui_video_signal,  
                    "vui_bitstream_restric": hi_venc_vui_bitstream_restric}
```

成员

成员名称	描述
vui_aspect_ratio	字典，预留参数，具体含义请参见H.265协议。
vui_time_info	字典，VUI字段中时钟计数相关参数。
vui_video_signal	字典，VUI字段中视频信号相关参数。
vui_bitstream_restric	字典，预留参数，具体含义请参见H.265协议。

2.10.8.4.89 hi_venc_vui_aspect_ratio

说明

定义H.264/H.265协议编码通道VUI中AspectRatio信息的字典。预留字典，暂不支持用户配置。

定义

```
hi_venc_vui_aspect_ratio = {"aspect_ratio_info_present_flag": aspect_ratio_info_present_flag,
                             "aspect_ratio_idc": aspect_ratio_idc,
                             "overscan_info_present_flag": overscan_info_present_flag,
                             "overscan_appropriate_flag": overscan_appropriate_flag,
                             "sar_width": sar_width,
                             "sar_height": sar_height}
```

成员

成员名称	描述
aspect_ratio_info_present_flag	整型，具体含义请参见H.264/H.265协议，系统默认值为0。
aspect_ratio_idc	整型，具体含义请参见H.264/H.265协议，系统默认值为1。
overscan_info_present_flag	整型，具体含义请参见H.264/H.265协议，系统默认值为0。
overscan_appropriate_flag	整型，具体含义请参见H.264/H.265协议，系统默认值为0。
sar_width	整型，具体含义请参见H.264/H.265协议，系统默认值为1。
sar_height	整型，具体含义请参见H.264/H.265协议，系统默认值为1。

2.10.8.4.90 hi_venc_vui_h264_time_info

说明

定义H.264协议编码通道VUI中TimeInfo信息的字典。

定义

```
hi_venc_vui_h264_time_info = {"timing_info_present_flag": timing_info_present_flag,
                              "fixed_frame_rate_flag": fixed_frame_rate_flag,
                              "num_units_in_tick": num_units_in_tick,
                              "time_scale": time_scale}
```

成员

成员名称	描述
timing_info_present_flag	整型，VUI字段中时钟计数信息是否生效的标志位，系统默认值为0。取值范围：0或1。表格中其它参数只有在该标志为1时才生效。
fixed_frame_rate_flag	整型，具体含义请参见H.264协议，系统默认值为1。取值范围：0或1。
num_units_in_tick	整型，时钟计数器加1所经过的时间单元数，系统默认值为1。取值范围：大于0。
time_scale	整型，一秒钟时长所经过的时间单元数，系统默认值为60。取值范围：大于0。

注意事项

- 时钟计数用于标记视频帧信号，在本H.264编码器中，视频显示帧率可由根据以下公式计算： $display\_rate = time\_scale / num\_units\_in\_tick / 2$ 。
- 由于显示帧率只与“time_scale”和“num_units_in_tick”的比值有关，建议用户设置时保持“num_units_in_tick”值为1，通过“time_scale”属性调整显示帧率，此时**显示帧率 = time_scale / 2**。
- 显示帧率的最终决定权在于播放器，播放器也可以选择不参考TimeInfo信息。

2.10.8.4.91 hi_venc_vui_h265_time_info

说明

定义H.264协议编码通道VUI中TimeInfo信息的字典。

定义

```
hi_venc_vui_h264_time_info = {"timing_info_present_flag": timing_info_present_flag,
                              "fixed_frame_rate_flag": fixed_frame_rate_flag,
                              "num_units_in_tick": num_units_in_tick,
                              "time_scale": time_scale}
```

成员

成员名称	描述
timing_info_present_flag	整型，VUI字段中时钟计数信息是否生效的标志位，系统默认值为0。取值范围：0或1。表格中其它参数只有在该标志为1时才生效。
fixed_frame_rate_flag	整型，时钟计数器加1所经过的时间单元数，系统默认值为1。取值范围：大于0。
num_units_in_tick	整型，一秒钟时长所经过的时间单元数，系统默认值为60。取值范围：大于0。
time_scale	整型，具体含义请参见H.265协议，系统默认值为1。取值范围： $[0, 2^{32}-2]$ 。

注意事项

- 时钟计数用于标记视频帧信号，在本H.265编码器中，视频显示帧率可由以下公式计算： $display\_rate = time\_scale / num\_units\_in\_tick$ 。
- 由于显示帧率只与“time_scale”和“num_units_in_tick”的比值有关，建议用户设置时保持“num_units_in_tick”值为1，通过“time_scale”属性调整显示帧率，此时显示帧率 = time_scale。
- 显示帧率的最终决定权在于播放器，播放器也可以选择不参考TimeInfo信息。

2.10.8.4.92 hi_venc_vui_video_signal

说明

定义H.264/H.265协议编码通道VUI中VideoSignal信息的字典。

定义

```
hi_venc_vui_video_signal = {"video_signal_type_present_flag": video_signal_type_present_flag,
                             "video_format": video_format,
                             "video_full_range_flag": video_full_range_flag,
                             "colour_description_present_flag": colour_description_present_flag,
                             "colour_primaries": colour_primaries,
                             "transfer_characteristics": transfer_characteristics,
                             "matrix_coefficients": matrix_coefficients}
```

成员

成员名称	描述
video_signal_type_present_flag	整型，VUI字段中视频信号信息是否生效的标志位，系统默认值为1。取值范围：0或1。表格中其它参数只有在该标志为1时才生效。
video_format	整型，具体含义请参见H.264/H.265协议，系统默认值为5。取值范围：H.264为 $[0, 7]$ ，H.265为 $[0, 5]$ 。

成员名称	描述
video_full_range_flag	整型，具体含义请参见H.264/H.265协议，系统默认值为0。取值范围：0或1。置0表示像素值域受限，亮度取值范围是16~235，色度的取值范围是16~240。 在Atlas 推理系列产品上，对于H.264、H.265码流，当前VENC编码时“video_full_range_flag”默认值为0（表示limited_range）。 在Atlas 200I/500 A2推理产品上，对于H.265码流，当前VENC编码时“video_full_range_flag”默认值为0（表示limited_range）；对于H.264码流，当前VENC编码时“video_full_range_flag”默认值为1（表示full_range）。
colour_description_present_flag	整型，具体含义请参见H.264/H.265协议，系统默认值为1，用户无需配置，系统内部根据当前图像实际信息填写。
colour_primaries	整型，具体含义请参见H.264/H.265协议，系统默认值为1，用户无需配置，系统内部根据当前图像实际信息填写。
transfer_characteristics	整型，具体含义请参见H.264/H.265协议，系统默认值为1，用户无需配置，系统内部根据当前图像实际信息填写。
matrix_coefficients	整型，具体含义请参见H.264/H.265协议，系统默认值为1，用户无需配置，系统内部根据当前图像实际信息填写。

2.10.8.4.93 hi_venc_vui_bitstream_restric

说明

定义H.264/H.265协议编码通道VUI中Bitstream_Restriction信息的字典。预留字典，暂不支持用户配置。

定义

```
hi_venc_vui_bitstream_restric = {"bitstream_restriction_flag": bitstream_restriction_flag}
```

成员

成员名称	描述
bitstream_restriction_flag	整型，具体含义请参见H.265协议，系统默认值为0。

2.10.8.4.94 hi_venc_cu_prediction

说明

定义CU模式选择的倾向性配置字典。

定义

```
hi_venc_cu_prediction = {"pred_mode": hi_op_mode,
    "intra32_cost": intra32_cost,
    "intra16_cost": intra16_cost,
    "intra8_cost": intra8_cost,
    "intra4_cost": intra4_cost,
    "inter64_cost": inter64_cost,
    "inter32_cost": inter32_cost,
    "inter16_cost": inter16_cost,
    "inter8_cost": inter8_cost}
```

成员

成员名称	描述
pred_mode	整型，倾向性选择模式，支持自动模式、手动模式两种。
intra32_cost	整型，Intra32倾向性调节，对应帧内预测32 * 32像素块尺寸模式，该值增大表示选择该模式的倾向性越小。 取值范围：[0, 15]。 默认值：8。 H.264该值无效。
intra16_cost	整型，Intra16倾向性调节，对应帧内预测16 * 16像素块尺寸模式，该值增大表示选择该模式的倾向性越小。 取值范围：[0, 15]。 默认值：8。
intra8_cost	整型，Intra8倾向性调节，对应帧内预测8 * 8像素块尺寸模式，该值增大表示选择该模式的倾向性越小。 取值范围：[0, 15]。 默认值：8。
intra4_cost	整型，Intra4倾向性调节，对应帧内预测4 * 4像素块尺寸模式，该值增大表示选择该模式的倾向性越小。 取值范围：[0, 15]。 默认值：8。
inter64_cost	整型，Inter64倾向性调节，对应帧间预测64 * 64像素块尺寸模式，该值增大表示选择该模式的倾向性越小。 取值范围：[0, 15]。 默认值：8。 H.264该值无效。

成员名称	描述
inter32_cost	整型，Inter32倾向性调节，对应帧间预测32 * 32像素块尺寸模式，该值增大表示选择该模式的倾向性越小。 取值范围：[0, 15]。 默认值：8。 H.264该值无效。
inter16_cost	整型，Inter16倾向性调节，对应帧间预测16 * 16像素块尺寸模式，该值增大表示选择该模式的倾向性越小。 取值范围：[0, 15]。 默认值：8。
inter8_cost	整型，Inter8倾向性调节，对应帧间预测8 * 8像素块尺寸模式，该值增大表示选择该模式的倾向性越小。 取值范围：[0, 15]。 默认值：8。

2.10.8.4.95 hi_venc_intra_refresh

说明

P帧刷Islice控制参数。

定义

```
hi_venc_intra_refresh = {"refresh_enable": refresh_enable,
                        "intra_refresh_mode": hi_venc_intra_refresh_mode,
                        "refresh_num": refresh_num,
                        "req_i_qp": req_i_qp}
```

成员

成员名称	描述
refresh_enable	整型，是否使能刷Islice功能。 0：不使能，默认值。 1：使能。
intra_refresh_mode	整型，I宏块刷新模式，分为按行刷新和按列刷新。默认按行刷新。 Atlas 推理系列产品 ，仅支持按行刷新。

成员名称	描述
refresh_num	每次I宏块刷新行数或者列数，可以通过这个变量控制刷新的速度及码流的平稳程度。 刷新行数或者列数越多，刷新的速度越快，但是码流平稳度越差；刷新的行数或者列数越少，刷新的速度越慢，但是码流平稳度越好。 需保证设置的“refresh_num”可以在一个GOP内完成Islice刷新，注意高级跳帧参考时只会在base层中的P帧（可被base层中其他帧的参考）进行刷新。 “refresh_num”需满足表2-69中的计算公式。
req_i_qp	整型，I帧QP值。 acl.himpi.venc_set_intra_refresh接口与acl.himpi.venc_request_idr接口配合使用时，该值用于控制插入的IDR帧的质量，该值越小，质量越好，插入的IDR帧大小越大。 取值范围：[0, 51]，默认值51。

参考信息

表 2-69 refresh_num 参数值计算公式

默认值	计算公式	备注
H.264 , refresh_num默认值为: $(pic\_height + lcu\_size - 1) \gg 6$	行: $refresh\_num * (max\_refresh\_frame\_i\_n\_gop - 1) \geq (pic\_height + lcu\_size - 1) // lcu\_size$	- GOP内P帧刷新slice最大帧数 “max_refresh_frame_in_gop”属性值取值说明: - 无高级跳帧参考时: $max\_refresh\_frame\_in\_gop = gop$; - 有高级跳帧参考时: $max\_refresh\_frame\_in\_gop = (gop + (base * (enhance + 1) - 1)) / (base * (enhance + 1))$。
H.265 , refresh_num默认值为: $((pic\_height + lcu\_size - 1) / lcu\_size) \gg 2$	列: $refresh\_num * (max\_refresh\_frame\_i\_n\_gop - 1) \geq (pic\_width + lcu\_size - 1) // lcu\_size$ 计算公式中的“//”表示向下取整。	- 编码单元尺寸 “lcu_size”属性值取值说明: - H.264时, 该参数值固定为16。 - H.265时: Atlas 推理系列产品, 该参数值固定为64。 Atlas 200I/500 A2推理产品, 该参数值固定为32。

2.10.8.4.96 hi_venc_intra_refresh_mode

说明

P帧刷新slice模式定义。

定义

```
INTRA_REFRESH_ROW    = 0      # 按行刷新
INTRA_REFRESH_COLUMN = 1      # 按列刷新
INTRA_REFRESH_BUTT   = 2
```

2.10.8.4.97 hi_venc_slice_split

说明

P帧刷Islice控制参数。

定义

```
hi_venc_slice_split = {"split_enable":split_enable,
                       "split_mode":split_mode,
                       "split_size":split_size,
                       "slice_output_en":slice_output_en,
                       "reserved":reserved[2]}
```

成员

成员名称	描述
split_enable	Slice分割是否使能。 0：不使能，默认值。 1：使能。
split_mode	Slice分割模式。 当前仅支持设置为1，表示按宏块行或者LCU行分割。
split_size	- H.264表示每个slice占的宏块行数。 最大值：(图像高 + 15) // 16 最小值：(最大值 + 15) // 16 - H.265表示每个slice占的LCU行数。 最大值：(图像高 + LCU - 1) // LCU 最小值：(最大值 + 15) // 16 说明 “split_size” 需以整数形式输入。
slice_output_en	当前仅支持设置为0。
reserved	预留参数，当前仅支持设置为0。

注意事项

当split_mode = 1时，使用H.265协议需要关注当前芯片的LCU大小。

Atlas 推理系列产品，该参数值固定为64。

Atlas 200I/500 A2推理产品，该参数值固定为32。

2.10.8.5 图像处理

2.10.8.5.1 hi_vpc_pic_info

说明

图像信息字典，用来描述VPC功能输入和输出图片信息。

定义

```
hi_vpc_pic_info = {"picture_address": picture_address,  
                  "picture_buffer_size": picture_buffer_size,  
                  "picture_width": picture_width,  
                  "picture_height": picture_height,  
                  "picture_width_stride": picture_width_stride,  
                  "picture_height_stride": picture_height_stride,  
                  "picture_format": hi_pixel_format}
```

成员

成员名称	描述
picture_address	整型，存放图片数据的Device地址。
picture_buffer_size	整型，存放图片数据的缓冲区大小。
picture_width	整型，图片真实宽。
picture_height	整型，图片真实高。
picture_width_stride	整型，图片宽Stride。
picture_height_stride	整型，图片高Stride。
picture_format	整型，目标图片的格式。

2.10.8.5.2 hi_vpc_crop_region

说明

VPC抠图功能需要的参数。

定义

```
hi_vpc_crop_region = {"top_offset": top_offset,  
                      "left_offset": left_offset,  
                      "crop_width": crop_width,  
                      "crop_height": crop_height}
```

成员

成员名称	描述
top_offset	整型，上偏移。
left_offset	整型，左偏移。
crop_width	整型，抠图区域宽。

成员名称	描述
crop_height	整型，抠图区域高。

2.10.8.5.3 hi_vpc_resize_info

说明

VPC缩放功能需要的参数。

定义

```
hi_vpc_resize_info = {"resize_width": resize_width,
                      "resize_height": resize_height,
                      "interpolation": interpolation}
```

成员

成员名称	描述
resize_width	整型，缩放之后的宽。
resize_height	整型，缩放之后的高。
interpolation	整型，缩放算法。 Atlas 推理系列产品，支持如下缩放算法： 0：业界通用的Bilinear算法（与OpenCV算法的计算过程类似，当输入和输出图片格式都为RGB时，在[1/32, 512]的缩放范围内，与OpenCV的单个像素值最大差异为正负1）。 1：业界通用的Nearest neighbor 算法（与OpenCV算法的计算过程类似）。 Atlas A2训练系列产品/Atlas 800I A2推理产品，支持如下缩放算法： 0：业界通用的Bilinear算法（与OpenCV-3.4.2版本算法的计算结果相同）。 1：业界通用的Nearest neighbor 算法（与OpenCV-3.4.2版本算法的计算结果相同）。 Atlas 200I/500 A2推理产品，支持如下缩放算法： 0：业界通用的Bilinear算法（与OpenCV-3.4.2版本算法的计算结果相同）。 1：业界通用的Nearest neighbor 算法（与OpenCV-3.4.2版本算法的计算结果相同）。

2.10.8.5.4 hi_vpc_crop_region_info

说明

VPC抠图功能需要的参数。

定义

```
hi_vpc_crop_region_info = {"dest_pic_info": hi_vpc_pic_info,  
                           "crop_region": hi_vpc_crop_region}
```

成员

成员名称	描述
dest_pic_info	字典，抠图目标图片信息。
crop_region	字典，抠图区域信息。

2.10.8.5.5 hi_vpc_crop_resize_region

说明

VPC抠图并缩放功能需要的参数。

定义

```
hi_vpc_crop_resize_region = {"dest_pic_info": hi_vpc_pic_info,  
                              "crop_region": hi_vpc_crop_region,  
                              "resize_info": hi_vpc_resize_info}
```

成员

成员名称	描述
dest_pic_info	字典，目标图片信息。
crop_region	字典，抠图区域信息。
resize_info	字典，图片缩放信息。

注意事项

处理图片的数量不能大于256个。缩放宽高必须与目标图片宽高一致。

2.10.8.5.6 hi_vpc_crop_resize_paste_region

说明

VPC抠图然后缩放并贴图到特定区域功能需要的参数。

定义

```
hi_vpc_crop_resize_paste_region = {"dest_pic_info": hi_vpc_pic_info,  
                                   "crop_region": hi_vpc_crop_region,  
                                   "resize_info": hi_vpc_resize_info,  
                                   "dest_top_offset": dest_top_offset,  
                                   "dest_left_offset": dest_left_offset}
```

成员

成员名称	描述
dest_pic_info	字典，目标图片信息。
crop_region	字典，抠图区域信息。
resize_info	字典，图片缩放信息。
dest_top_offset	整型，贴图到另外一张图的顶部偏移。
dest_left_offset	整型，贴图到另外一张图的左侧偏移。 在Atlas 推理系列产品上，左偏移16像素对齐。

2.10.8.5.7 hi_vpc_bord_type

说明

定义边界填充的类型。

定义

```
HI_BORDER_CONSTANT    = 0  
HI_BORDER_REPLICATE   = 1  
HI_BORDER_REFLECT     = 2  
HI_BORDER_REFLECT_101 = 3
```

成员

成员名称	描述
HI_BORDER_CONSTANT	添加有颜色的常数值边界。
HI_BORDER_REPLICATE	重复最后一个元素。 举例，其中*表示任意图像元素：aaaaaa a*****h hhhhhhh。
HI_BORDER_REFLECT	边界元素的镜像，镜像包括边界元素。 Atlas 200I/500 A2推理产品 硬件限制，不支持该选项。 举例，其中*表示任意图像元素：ba abc*****fgh hg。
HI_BORDER_REFLECT_101	边界元素的镜像，镜像不包括边界元素。 Atlas 200I/500 A2推理产品 硬件限制，不支持该选项。 举例，其中*表示任意图像元素：cb abc****fgh gf。

注意事项

版本	约束
Atlas 推理系列产品	支持 “HI_BORDER_CONSTANT” 和 “HI_BORDER_REPLICATE” 可以填充到4096 * 4096。
Atlas 200I/500 A2推理产品 Atlas A2训练系列产品/ Atlas 800I A2推理产品	支持 “HI_BORDER_CONSTANT” 和 “HI_BORDER_REPLICATE” 可以填充到4096 * 8192。

2.10.8.5.8 hi_vpc_make_border_info

说明

VPC图像填充需要用的信息。

定义

```
hi_vpc_crop_resize_paste_region = {"top": top,  
                                   "bottom": bottom,  
                                   "left": left,  
                                   "right": right,  
                                   "border_type": hi_vpc_bord_type,  
                                   "scalar_value": hi_vpc_scalar}
```

成员

成员名称	描述
top	整型，顶部填充像素数。
bottom	整型，底部填充像素数。
left	整型，左侧填充像素数。
right	整型，右侧填充像素数。
border_type	整型，扩充边缘像素的类型。
scalar_value	字典，仅在填充类型为 “HI_BORDER_CONSTANT” 的时候有效，指定填充的像素值。

2.10.8.5.9 hi_vpc_histogram_config

说明

存放直方图统计结果的字典。

定义

```
hi_vpc_histogram_config = {"histogram_y_or_r": histogram_y_or_r[256], # 256 level statistics of Y or R  
                           component
```



```
"histogram_u_or_g": histogram_u_or_g[256], # 256 level statistics of U or G component  
"histogram_v_or_b": histogram_v_or_b[256]} # 256 level statistics of V or B component
```

成员

成员名称	描述
histogram_y_or_r	列表，Y或者R分量，像素分布情况。
histogram_u_or_g	列表，U或者G分量，像素分布情况。
histogram_v_or_b	列表，V或者B分量，像素分布情况。

2.10.8.5.10 hi_vpc_lut_remap

说明

配置图像各分量重映射信息的字典。

定义

```
hi_vpc_lut_remap = {"map_value_y_or_r": map_value_y_or_r[256], # remap value of Y or R component  
"map_value_u_or_g": map_value_u_or_g[256], # remap value of U or G component  
"map_value_v_or_b": map_value_v_or_b[256]} # remap value of V or B component
```

成员

成员名称	描述
map_value_y_or_r	列表，Y或者R分量的重映射配置信息。
map_value_u_or_g	列表，U或者G分量的重映射配置信息。
map_value_v_or_b	列表，V或者B分量的重映射配置信息。

2.10.8.5.11 hi_csc_conf

说明

定义色域配置信息。

定义

```
hi_csc_conf = {"alpha": alpha,  
"attr": attr[4]}
```

成员

成员名称	描述
alpha	用于设置图像的透明度，取值范围：[0, 255]，取值越小，图像越透明。 在查看图像时，无法直接感知透明度，一般在对图像进行融合、渲染等处理时才会使用透明度信息。
attr	预留字段，暂不支持。需要用户手动将数组中的每个元素值设置为0，避免后续版本的兼容性问题。

2.10.8.5.12 hi_vpc_scalar

说明

定义图像处理像素各分量字典。

定义

```
hi_vpc_scalar = {"val": val[4]}
```

成员

成员名称	描述
val	列表，存放颜色分量的值，取值范围[0, 255]。 当前仅支持用户手动按顺序存放R、G、B分量或Y、U、V分量的值，即val[0]存放R分量或Y分量，val[1]存放G分量或U分量，val[2]存放B分量或V分量，val[3]预留。 若输入输出都是YUV格式，填充YUV分量的值，反之，填充的是RGB分量的值。

2.10.8.5.13 hi_vpc_chn_attr

说明

定义图像处理通道属性字典。

定义

```
hi_vpc_chn_attr = {"attr": attr,  
                  "pic_width": pic_width, # RW; reserved  
                  "pic_height": pic_height} # RW; reserved
```

成员

成员名称	描述
attr	Atlas 200I/500 A2推理产品，预留参数，暂不支持。 Atlas A2训练系列产品/Atlas 800I A2推理产品，预留参数，暂不支持。 任务队列深度，取值范围：[0, 350]，取值越大，队列深度越深，在一个通道内可下发的任务数量越多，但系统内部内存消耗线性增加。 “attr”属性取值为[0, 350]之间，具体说明如下： - 取值为0，表示默认使用32。 - 取值为[1, 10)，VPC内部考虑批量任务的处理（例如2.10.4.21 acl.himpi.vpc_batch_crop_resize_paste），自动将队列深度设置为10。 - 取值为[10, 350]，取用户所配置的值。 说明 任务队列深度取值大于350时，当前版本会自行采用350的取值，为避免后续版本的兼容性问题，建议用户在[队列深度最小值, 350]范围内取值。 在一个VPC处理通道关联n个线程的场景下，防止通道队列满而阻塞任务下发，建议增加任务队列深度以保证性能，当线程数 ≤ 32 时，可使用默认队列深度32。当线程数 > 32时，建议队列深度大于或等于线程数。
pic_width	整型，通道支持的处理图像最大宽（以像素为单位）。预留字段，暂不支持。需要用户手动设置为0，避免后续版本的兼容性问题。 静态属性。
pic_height	整型，通道支持的处理图像最大高（以像素为单位）。预留字段，暂不支持。需要用户手动设置为0，避免后续版本的兼容性问题。 静态属性。

2.10.8.5.14 hi_vpc_crop_resize_border_region

说明

VPC抠图缩放填充信息字典。

定义

```
hi_vpc_crop_resize_border_region = {"dest_pic_info": hi\_vpc\_pic\_info,
                                     "crop_region": hi\_vpc\_crop\_region,
                                     "resize_info": hi\_vpc\_resize\_info,
                                     "dest_top_offset": dest_top_offset,
                                     "dest_left_offset": dest_left_offset,
                                     "border_type": hi\_vpc\_bord\_type,
                                     "scalar_value": hi\_vpc\_scalar}
```

成员

成员名称	描述
dest_pic_info	字典，目标图片信息。
crop_region	字典，原图抠图区域信息。
resize_info	字典，抠图缩放信息。
dest_top_offset	整型，贴图到另外一张图的顶部偏移。
dest_left_offset	整型，贴图到另外一张图的左侧偏移。
border_type	整型，扩充边缘像素的类型。
scalar_value	字典，仅在填充类型为“HI_BORDER_CONSTANT”的时候有效，指定填充的像素值。 当前支持用户手动按顺序存放R、G、B分量或Y、U、V分量的值。 若输入输出都是YUV格式，填充YUV分量的值；反之，填充的是RGB分量的值。

注意事项

默认抠图区域的图片格式会转成目标图片的格式，然后再缩放、填充至目标图片的宽高。

2.10.8.5.15 hi_vpc_crop_resize_resize_paste_region

说明

VPC抠图然后缩放并贴图到特定区域功能需要的参数，支持两次缩放。

定义

```
hi_vpc_crop_resize_resize_paste_region = {"dest_pic_info": hi_vpc_pic_info,  
                                           "crop_region": hi_vpc_crop_region,  
                                           "resize_info1": hi_vpc_resize_info,  
                                           "resize_info2": hi_vpc_resize_info,  
                                           "dest_top_offset": dest_top_offset,  
                                           "dest_left_offset": dest_left_offset,  
                                           "reserved": [2]}
```

成员

成员名称	描述
dest_pic_info	字典，目标图片信息。
crop_region	字典，抠图区域信息。
resize_info1	字典，第一次图片缩放的信息。
resize_info2	字典，第二次图片缩放的信息。

成员名称	描述
dest_top_offset	整型，贴图到另外一张图的顶部偏移。
dest_left_offset	整型，贴图到另外一张图的左侧偏移。 Atlas 推理系列产品 上，左偏移16像素对齐。
reserved	列表，预留属性，为保证后续版本兼容性，请务必传入[0, 0]清零初始化，在代码中必须避免显式对reserved字段进行访问。

2.10.8.5.16 hi_vpc_workspace_param

说明

定义设置系统内部临时缓存相关的参数。

定义

```
hi_vpc_workspace_param = {"func": hi_workspace_func,
                           "max_source_pic_width": max_source_pic_width,
                           "max_source_pic_height": max_source_pic_height,
                           "max_dest_pic_width": max_dest_pic_width,
                           "max_dest_pic_height": max_dest_pic_height,
                           "reserved": reserved}
```

成员

Atlas 200I/500 A2推理产品，输入、输出图片分辨率的取值范围为：10 * 6 ~ 4096 * 8192。

Atlas A2训练系列产品/Atlas 800I A2推理产品，输入、输出图片分辨率的取值范围为：10 * 6 ~ 4096 * 8192。

参数名	输入/输出	说明
func	输入	整型，指定为哪种功能申请临时缓存。
max_source_pic_width	输入	整型，输入图片的最大宽度。
max_source_pic_height	输入	整型，输入图片的最大高度。
max_dest_pic_width	输入	整型，输出图片的最大宽度。
max_dest_pic_height	输入	整型，输出图片的最大高度。
reserved	输入	整型，预留参数，为保证后续版本兼容性，请务必使用memset配置方式进行清零初始化，在代码中必须避免显式对reserved字段进行访问。

2.10.8.5.17 hi_workspace_func

说明

定义需申请临时缓存的功能枚举。

定义

```
REMAP = 0,  
AFFINE = 1
```

2.10.8.5.18 hi_remap_lut

说明

定义像素位置重映射表信息。

定义

```
hi_remap_lut = {"src_pic_width": src_pic_width,  
                "src_pic_height": src_pic_height,  
                "dest_pic_width": dest_pic_width,  
                "dest_pic_height": dest_pic_height,  
                "lut": lut,  
                "lut_size": lut_size,  
                "reserved": reserved}
```

成员

Atlas 200I/500 A2推理产品，输入、输出图片分辨率的取值范围为：10 * 6 ~ 4096 * 8192。

Atlas A2训练系列产品/Atlas 800I A2推理产品，输入、输出图片分辨率的取值范围为：10 * 6 ~ 4096 * 8192。

成员名称	描述
src_pic_width	整型，输入图片宽度。
src_pic_height	整型，输入图片高度。
dest_pic_width	整型，输出图片宽度。
dest_pic_height	整型，输出图片高度。
lut_size	整型，像素位置重映射表的内存大小，单位Byte。 可调用 2.10.4.36 acl.himpi.vpc_get_lut_mem_size 接口获取内存大小。
lut	整型，像素位置重映射表的内存地址。 Atlas 200I/500 A2推理产品 ，该内存需由用户提前调用 acl.himpi.dvpp_malloc 接口申请。 Atlas A2训练系列产品/Atlas 800I A2推理产品 ，该内存需由用户提前调用 acl.himpi.dvpp_malloc 接口申请。

成员名称	描述
reserved	整型，预留参数，为保证后续版本兼容性，请务必使用memset配置方式进行清零初始化，在代码中必须避免显式对reserved字段进行访问。

2.10.8.5.19 hi_point_pair_info

说明

输入、输出图片中的像素点坐标信息。

定义

```
hi_point_pair_info = {"src_point": hi_point[4],  
                      "src_point_count": src_point_count,  
                      "dest_point": hi_point[4],  
                      "dest_point_count": dest_point_count}
```

成员

成员名称	描述
src_point	字典，输入图片像素点坐标。
src_point_count	整型，输入图片像素点坐标数量。
dest_point	字典，输出图片像素点坐标。
dest_point_count	整型，输出图片像素点坐标数量。

2.10.8.5.20 hi_float_point

说明

坐标信息。

定义

```
hi_float_point = {"x": x,  
                  "y": y}
```

成员

成员名称	描述
x	浮点型，像素点x坐标。
y	浮点型，像素点y坐标。

2.10.8.5.21 hi_warp_transform_param

说明

Remap变换参数。

定义

```
hi_warp_transform_param = {"src": hi_vpc_pic_info,  
                           "dst": hi_vpc_pic_info,  
                           "transform_config": hi_transform_config}
```

成员

成员名称	描述
src	字典，输入图片信息
dst	字典，输出图片信息。
transform_config	字典，仿射/透视/Remap变换配置信息。

2.10.8.5.22 hi_transform_config

说明

定义仿射/透视/Remap变换配置信息。

定义

```
hi_transform_config = {"interpolation": interpolation,  
                       "border_type": hi_vpc_bord_type,  
                       "scalar_value": hi_vpc_scalar}
```

成员

成员名称	描述
interpolation	整型，支持如下缩放算法： Atlas 200I/500 A2推理产品 支持如下缩放算法： 0：业界通用的Bilinear算法 1：业界通用的Nearest neighbor 算法 Atlas A2训练系列产品/Atlas 800I A2推理产品 支持如下缩放算法： 0：业界通用的Bilinear算法 1：业界通用的Nearest neighbor 算法

成员名称	描述
border_type	整型，边界填充的类型。 Atlas 200I/500 A2推理产品 支持如下取值： 0：寄存器固定值 1：边界复制模式 Atlas A2训练系列产品/Atlas 800I A2推理产品 支持如下取值： 0：寄存器固定值 1：边界复制模式
scalar_value	字典，图像填充的像素值，只支持3通道。 Atlas 200I/500 A2推理产品 ，如果输入图片格式为YUV400，则val[0]存放Y分量，val[1]、val[2]、val[3]预留，需初始化为0。 Atlas A2训练系列产品/Atlas 800I A2推理产品 ，如果输入图片格式为YUV400，则val[0]存放Y分量，val[1]、val[2]、val[3]预留，需初始化为0。

2.10.8.5.23 hi_morph_shapes

说明

定义滤波卷积形状。

定义

```
MORPH_RECT    = 0,  
MORPH_CROSS   = 1,  
MORPH_ELLIPSE = 2,  
MORPH_MAX     = 100
```

成员

成员名称	描述
MORPH_RECT	矩形。
MORPH_CROSS	交叉形，预留字段，暂不支持。
MORPH_ELLIPSE	椭圆形，预留字段，暂不支持。
MORPH_MAX	最大可支持的卷积滤波形状，预留字段，暂不支持。

2.10.8.5.24 hi_rotation

说明

定义旋转角度。

定义

```
HI_ROTATION_90 = 0,  
HI_ROTATION_180 = 1,  
HI_ROTATION_270 = 2
```

成员

成员名称	描述
HI_ROTATION_90	顺时针旋转90度。
HI_ROTATION_180	顺时针旋转180度。
HI_ROTATION_270	顺时针旋转270度。

2.10.8.5.25 hi_blk_size

说明

定义单个马赛克框的大小。

定义

```
HI_BLK_SIZE_4 = 0,  
HI_BLK_SIZE_8 = 1,  
HI_BLK_SIZE_16 = 2,  
HI_BLK_SIZE_32 = 3,  
HI_BLK_SIZE_64 = 4,  
HI_BLK_SIZE_128 = 5
```

成员

成员名称	描述
HI_BLK_SIZE_4	预留字段，暂不支持。
HI_BLK_SIZE_8	单个马赛克是8 * 8的正方形。
HI_BLK_SIZE_16	单个马赛克是16 * 16的正方形。
HI_BLK_SIZE_32	单个马赛克是32 * 32的正方形。
HI_BLK_SIZE_64	预留字段，暂不支持。
HI_BLK_SIZE_128	预留字段，暂不支持。

2.10.8.5.26 hi_median_blur_param

说明

中值滤波信息。

定义

```
hi_median_blur_param = {"src": hi_vpc_pic_info,  
                        "dst": hi_vpc_pic_info,  
                        "median_blur_cfg": hi_median_blur_config}
```

成员

成员名称	描述
src	字典，输入图片信息。
dst	字典，输出图片信息。
median_blur_cfg	字典，中值滤波配置参数。

2.10.8.5.27 hi_median_blur_config

说明

中值滤波参数。

定义

```
hi_median_blur_config = {"kernel_size": kernel_size}
```

成员

成员名称	描述
kernel_size	整型，中值滤波卷积核尺寸，只支持1、3或者5。

2.10.8.5.28 hi_blur_param

说明

滤波信息。

定义

```
hi_blur_param = {"src": hi_vpc_pic_info,  
                 "dst": hi_vpc_pic_info,  
                 "blur_config": hi_blur_config}
```

成员

成员名称	描述
src	字典，输入图片信息。
dst	字典，输出图片信息。

成员名称	描述
blur_cfg	字典，滤波配置参数。

2.10.8.5.29 hi_blur_config

说明

滤波参数。

定义

```
hi_blur_config = {"kernel_size": hi_size,  
                  "morph_shapes": hi_morph_shapes,  
                  "anchor": hi_point,  
                  "iterations": iterations,  
                  "border_type": hi_vpc_bord_type,  
                  "scalar_value": hi_vpc_scalar}
```

成员

成员名称	描述
kernel_size	字典，卷积核的尺寸，“kernel_size”的宽和高只能设置为1、3或者5。
morph_shapes	整型，滤波的卷积形状。
anchor	字典，卷积核锚点，需要设置成(-1, -1)，预留字段。
iterations	整型，卷积迭代次数，范围[1, 100]，均值滤波处理只支持设置成1，腐蚀或膨胀处理时支持取[1, 100]范围内的值。
border_type	整型，边界填充类型，仅支持“HI_BORDER_CONSTANT”和“HI_BORDER_REPLICATE”两种填充模式。
scalar_value	字典，边界填充值。

2.10.8.5.30 hi_gaussian_blur_param

说明

高斯滤波信息。

定义

```
hi_gaussian_blur_param = {"src": hi_vpc_pic_info,  
                           "dst": hi_vpc_pic_info,  
                           "gaussian_blur_cfg": hi_gaussian_blur_config}
```

成员

成员名称	描述
src	字典，输入图片信息。
dst	字典，输出图片信息。
gaussian_blur_cfg	字典，高斯滤波配置参数。

2.10.8.5.31 hi_gaussian_blur_config

说明

高斯滤波参数。

定义

```
hi_gaussian_blur_config = {"kernel_size": hi_size,  
                           "sigma_x": sigma_x,  
                           "sigma_y": sigma_y,  
                           "border_type": hi_vpc_bord_type,  
                           "scalar_value": hi_vpc_scalar,  
                           "reserved": [2]}
```

成员

成员名称	描述
kernel_size	字典，卷积核尺寸，“kernel_size”的宽和高只能设置为1、3或者5。
sigma_x	整型，高斯滤波 x 轴sigma值。
sigma_y	整型，高斯滤波 y 轴sigma值，如果sigma_y = 0，则sigma_y = sigma_x。
border_type	整型，边界填充类型，仅支持“HI_BORDER_CONSTANT”和“HI_BORDER_REPLICATE”两种填充模式。
scalar_value	字典，边界填充值。
reserved[2]	列表，预留参数，代码中必须避免显式对reserved字段进行访问。

2.10.8.5.32 hi_filter_2d_param

说明

2D卷积滤波信息。

定义

```
hi_filter_2d_param = {"src": hi_vpc_pic_info,  
                     "dst": hi_vpc_pic_info,  
                     "filter_2d_cfg": hi_filter_2d_config}
```

成员

成员名称	描述
src	字典，输入图片信息。
dst	字典，输出图片信息。
filter_2d_cfg	字典，2D卷积滤波配置参数。

2.10.8.5.33 hi_filter_2d_config

说明

2D卷积滤波参数。

定义

```
hi_filter_2d_config = {"filter": [h][w],  
                      "kernel_size": hi_size,  
                      "anchor": hi_point,  
                      "delta": delta,  
                      "border_type": hi_vpc_bord_type,  
                      "scalar_value": hi_vpc_scalar,  
                      "reserved": [2]}
```

成员

成员名称	描述
filter	列表，自定义卷积核。 “w”列“h”行二维列表，“w”与“h”分别与“kernel_size”属性字典中的“width”属性和“height”属性对应。 例如：[[1,2,3]]为一个3列1行的列表，w和h分别为3和1。
kernel_size	字典，卷积核尺寸，“kernel_size”的宽和高只能设置为1、3或者5。
anchor	字典，卷积核锚点，预留字段，需要设置成(-1, -1)。
delta	整型，增量值，会被加到执行卷积之后的每个像素值之上。
border_type	整型，边界填充类型，仅支持“HI_BORDER_CONSTANT”和“HI_BORDER_REPLICATE”两种填充模式。

成员名称	描述
scalar_value	字典，边界填充值。
reserved[2]	列表，预留参数，在代码中必须避免显式对reserved字段进行访问。

2.10.8.5.34 hi_mosaic_param

说明

马赛克信息。

定义

```
hi_mosaic_param = {"src": hi_vpc_pic_info,  
                  "dst": hi_vpc_pic_info,  
                  "count": count,  
                  "mosaic": hi_mosaic}
```

成员

成员名称	描述
src	字典，输入图片信息。
dst	字典，输出图片信息。
count	整型，马赛克区域个数，取值范围[1, 100]。
mosaic	列表，马赛克区域配置参数列表。

2.10.8.5.35 hi_rotate_param

说明

固定角度旋转信息。

定义

```
hi_rotate_param = {"src": hi_vpc_pic_info,  
                  "dst": hi_vpc_pic_info,  
                  "angle": hi_rotation}
```

成员

成员名称	描述
src	字典，输入图片信息。
dst	字典，输出图片信息。

成员名称	描述
angle	整型，旋转角度。

2.10.8.5.36 hi_mosaic

说明

马赛克参数。

定义

```
hi_mosaic = {"blk_size": hi_blk_size,  
            "rect": hi_rect}
```

成员

成员名称	描述
blk_size	整型，单个马赛克的大小，仅支持“HI_BLK_SIZE_8”、“HI_BLK_SIZE_16”、“HI_BLK_SIZE_32”。
rect	字典，马赛克区域，其宽、高最小为8，可以部分区域在图片外。

2.10.8.5.37 hi_cover_type

说明

定义覆盖类型。

定义

```
HI_COVER_RECT = 0  
HI_COVER_QUAD = 1
```

成员

成员名称	描述
HI_COVER_RECT	矩形覆盖。
HI_COVER_QUAD	任意四边形覆盖。

2.10.8.5.38 hi_cover_param

说明

覆盖信息。

定义

```
hi_cover_param = {"src": hi_vpc_pic_info,  
                  "dst": hi_vpc_pic_info,  
                  "count": count,  
                  "cover": hi_cover}
```

成员

成员名称	描述
src	字典，输入图片信息。
dst	字典，输出图片信息。
count	整型，覆盖区域个数，取值范围[1, 100]。
cover	列表，覆盖区域配置参数列表。

2.10.8.5.39 hi_cover

说明

覆盖参数。

定义

```
hi_cover = {"type": hi_cover_type,  
            "rect"/>"quad": hi_rect/hi_quad_cover,  
            "color": color}
```

成员

成员名称	描述
type	整型，覆盖类型。
rect	字典，覆盖区域。 当图片格式为YUV422SP、YUV420SP时，要求x、y、width、height值均为偶数。
quad	字典，自定义覆盖区域。
color	整型，覆盖区域的颜色值，低24位有效。

2.10.8.5.40 hi_quad_cover

说明

自定义覆盖区域信息。

定义

```
hi_quad_cover = {"is_solid": is_solid,  
                 "thick": thick,  
                 "point": hi_point[4]}
```

成员

成员名称	描述
is_solid	整型，当前只支持设置为1。
thick	整型，预留字段，暂不支持。
point[4]	列表，四边形4个点的坐标。

2.10.8.5.41 hi_line_param

说明

定义画线信息。

定义

```
hi_line_param = {"src": hi_vpc_pic_info,  
                 "dst": hi_vpc_pic_info,  
                 "count": count,  
                 "line": hi_line}
```

成员

成员名称	描述
src	字典，输入图片信息。
dst	字典，输出图片信息。
count	整型，画线条数，取值范围[1, 100]。
line	列表，画线配置参数列表。 当图片格式为YUV422SP、YUV420SP时，要求起始点、结束点坐标值为偶数。

2.10.8.5.42 hi_line

说明

画线参数。

定义

```
hi_line = {"start_point": hi_point,  
           "end_point": hi_point,
```

```
"thick": thick,  
"color": color}
```

成员

成员名称	描述
start_point	字典，起始点坐标。
end_point	字典，结束点坐标。 结束点坐标不能起始点坐标相同。
thick	整型，线的粗细值，取值需要2对齐。
color	整型，线的颜色值，低24位有效。

2.10.8.5.43 hi_osd_param

说明

叠加信息。

定义

```
hi_osd_param = {"src": hi_vpc_pic_info,  
                "dst": hi_vpc_pic_info,  
                "count": count,  
                "osd": hi_osd,  
                "clut": clut[16]}
```

成员

成员名称	描述
src	字典，输入图片信息。
dst	字典，输出图片信息。
count	整型，叠加区域个数，取值范围[1, 100]。
osd	列表，叠加区域配置参数类别。
clut	列表，clut查找表系数。

2.10.8.5.44 hi_osd

说明

叠加参数。

定义

```
hi_osd = {"rect": hi_rect,  
          "pixel_format": hi_pixel_format,
```

```
"picture_address": picture_address,  
"stride": stride,  
"bg_alpha": bg_alpha,  
"fg_alpha": fg_alpha,  
"osd_inverted_color": hi_osd_inverted_color}
```

成员

成员名称	描述
rect	字典，叠加区域坐标。
pixel_format	整型，被叠加图片的格式。 支持如下格式： HI_PIXEL_FORMAT_ARGB_8888 = 14, // ARGB8888 HI_PIXEL_FORMAT_ARGB_1555 = 33, // ARGB1555 A:1bit R:5bit G:6bit B:5bit HI_PIXEL_FORMAT_ARGB_4444 = 25, // ARGB4444 A:4bit R:4bit G:4bit B:4bit HI_PIXEL_FORMAT_ARGB_CLUT2 = 41, // ARGB Color Lookup Table 2bit HI_PIXEL_FORMAT_ARGB_CLUT4 = 42, // ARGB Color Lookup Table 4bit
picture_address	整型，被叠加图片的地址。
stride	整型，被叠加图片的Stride。 - 对于“HI_PIXEL_FORMAT_ARGB_8888”格式，“stride”值为rect["width"]的4倍。 - 对于“HI_PIXEL_FORMAT_ARGB_1555”或“HI_PIXEL_FORMAT_ARGB_4444”格式，“stride”值为rect["width"]的2倍。 - 对于“HI_PIXEL_FORMAT_ARGB_CLUT2”格式，“stride”值为rect["width"]的1/4倍。 - 对于“HI_PIXEL_FORMAT_ARGB_CLUT4”格式，“stride”值为rect["width"]的1/2倍。
bg_alpha	整型，背景alpha。 取值范围[0, 255]。 仅当被叠加图片的格式为“HI_PIXEL_FORMAT_ARGB_1555”格式时，需设置该参数。
fg_alpha	整型，前景alpha。 取值范围[0, 255]。 仅当被叠加图片的格式为“HI_PIXEL_FORMAT_ARGB_1555”格式时，需设置该参数。
osd_inverted_color	整型，反色类型。

2.10.8.5.45 hi_osd_inverted_color

说明

定义OSD反色类型。

定义

```
HI_OSD_INVERTED_COLOR_NONE = 0,  
HI_OSD_INVERTED_COLOR_RGB  = 1,  
HI_OSD_INVERTED_COLOR_ALPHA = 2,  
HI_OSD_INVERTED_COLOR_ALL  = 3
```

成员

成员名称	描述
HI_OSD_INVERTED_COLOR_NONE	不反色。
HI_OSD_INVERTED_COLOR_RGB	RGB通道反色。
HI_OSD_INVERTED_COLOR_ALPHA	ALPHA通道反色。
HI_OSD_INVERTED_COLOR_ALL	所有通道反色。

2.10.8.5.46 hi_transform_matrix

说明

定义仿射变换矩阵参数。

定义

```
{"matrix_type": hi\_matrix\_type,  
 "matrix": matrix[3][4]}
```

成员

成员名称	描述
matrix_type	整型，矩阵类型。
matrix	列表，仿射变换矩阵。

2.10.8.5.47 hi_matrix_type

说明

定义仿射变换中的矩阵类型。

定义

```
HI_MATRIX_2_PLUS_3 = 0,  
HI_MATRIX_3_PLUS_3 = 1,  
HI_MATRIX_3_PLUS_4 = 2
```

成员

成员名称	描述
HI_MATRIX_2_PLUS_3	2 * 3 Matrix。
HI_MATRIX_3_PLUS_3	3 * 3 Matrix。
HI_MATRIX_3_PLUS_4	3 * 4 Matrix。

2.10.8.5.48 hi_flip_param

说明

翻转信息。

定义

```
hi_flip_param = {"src": hi\_vpc\_pic\_info,  
                 "dst": hi\_vpc\_pic\_info,  
                 "flip_mode": hi\_flip\_mode}
```

成员

成员名称	描述
src	字典，输入图片信息。
dst	字典，输出图片信息。
flip_mode	整型，翻转模式参数。

2.10.8.5.49 hi_flip_mode

说明

定义翻转模式类型。

定义

```
HI_FLIP_HOR = 0,  
HI_FLIP_VER = 1,  
HI_FLIP_BOTH = 2
```

成员

成员名称	描述
HI_FLIP_HOR	进行水平方向翻转。
HI_FLIP_VER	进行竖直方向翻转。
HI_FLIP_BOTH	同时进行水平方向翻转和竖直方向翻转，相当于180度旋转。

2.10.8.5.50 hi_map_param

说明

用户原始map表信息。

定义

```
hi_map_param = {"map1": map1,  
                "map2": map2,  
                "map_size": map_size,  
                "src_pic_width": src_pic_width,  
                "src_pic_height": src_pic_height,  
                "dest_pic_width": dest_pic_width,  
                "dest_pic_height": dest_pic_height,  
                "reserved": reserved}
```

成员

成员名称	描述
map1	整型，Remap变换使用的“map1”表, 表示(x, y)坐标的x，类型为hi_float, 大小为：dest_pic_width * dest_pic_height * sizeof(hi_float)
map2	整型，Remap变换使用的“map2”表, 表示(x, y)坐标的y，类型为hi_float, 大小为：dest_pic_width * dest_pic_height * sizeof(hi_float)
map_size	整型，“map1”或“map2”的内存大小，“map1”表和“map2”表的内存大小一样，单位Byte。
src_pic_width	整型，map表对应的输入图片的宽度。
src_pic_height	整型，map表对应的输入图片的高度。
dest_pic_width	整型，map表对应的输出图像的宽度。
dest_pic_height	整型，map表对应的输出图像的高度。

成员名称	描述
reserved	整型，预留参数，在代码中必须避免显式对reserved字段进行访问。

2.10.8.6 PNGD 图像解码

2.10.8.6.1 hi_pngd_chn_attr

说明

定义PNGD解码通道属性字典。

定义

```
hi_pngd_chn_attr = {"stream_que_cnt": stream_que_cnt, "reserved": reserved[4]}
```

成员

成员名称	描述
stream_que_cnt	整型，PNGD图片缓存队列大小。预留参数。
reserved	列表，预留参数，在代码中必须避免显示对reserved字段进行访问。

2.10.8.6.2 hi_img_stream

说明

定义图片码流信息。

定义

```
hi_img_stream = {"type": hi\_payload\_type, "addr": addr, "len": len, "pts": pts, "reserved": reserved[2]}
```

成员

成员名称	描述
type	整型，图片类型。
addr	整型，图片在内存中的起始虚拟地址。
len	整型，存放图片的内存大小，单位Byte。
pts	整型，图片时间戳，预留参数。
reserved	预留参数，在代码中必须避免显示对reserved字段进行访问。

2.10.8.6.3 hi_pic_info

说明

定义图片信息。

定义

```
hi_pic_info = {"picture_address": picture_address,
               "picture_buffer_size": picture_buffer_size,
               "picture_width": picture_width,
               "picture_height": picture_height,
               "picture_width_stride": picture_width_stride,
               "picture_height_stride": picture_height_stride,
               "picture_format": hi_pixel_format}
```

成员

成员名称	描述
picture_address	整型，图片在Device内存中的起始虚拟地址。
picture_buffer_size	整型，存放图片的内存大小，单位Byte。
picture_width	整型，图片宽。
picture_height	整型，图片高。
picture_width_stride	整型，图片宽Stride。
picture_height_stride	整型，图片高Stride。
picture_format	整型，图片格式。

2.10.8.6.4 hi_png_color_format

说明

定义PNG源图片格式的枚举。

定义

```
HI_PNG_COLOR_FORMAT_GRAY = 0x0, # gray bitmap
HI_PNG_COLOR_FORMAT_RGB = 0x2, # RGB bitmap
HI_PNG_COLOR_FORMAT_CLUT = 0x3, # PLTE调色板数据
HI_PNG_COLOR_FORMAT_AGRAY = 0x4, # gray bitmap with alpha
HI_PNG_COLOR_FORMAT_ARGB = 0x6, # RGB bitmap with alpha
HI_PNG_COLOR_FORMAT_BUTT = 0x100
```

2.10.9 返回码列表

2.10.9.1 公共返回码

错误代码	宏定义	描述
0xA0028001	-	无效的Device ID。
0xA0028002	-	无效的channel ID。
0xA0028003	HI_ERR_SYS_ILLEGAL_PARAM	参数设置无效。
0xA0028004	HI_ERR_SYS_EXIST	通道或资源已存在。
0xA0028005	HI_ERR_SYS_UNEXIST	通道或资源不存在。
0xA0028006	HI_ERR_SYS_NULL_PTR	指针地址为0。
0xA0028007	-	使能系统、Device或通道前未配置对应的参数。
0xA0028008	HI_ERR_SYS_NOT_SUPPORT	不支持的功能。
0xA0028009	HI_ERR_SYS_NOT_PERM	操作不允许。
0xA002800C	HI_ERR_SYS_NO_MEM	分配内存失败，如系统内存不足。
0xA002800D	-	分配缓存失败，如申请的数据缓冲区太大。
0xA002800E	-	缓冲区中无数据。
0xA002800F	-	缓冲区中数据满。
0xA0028010	HI_ERR_SYS_NOT_READY	系统控制属性未配置。
0xA0028011	-	地址错误。
0xA0028012	HI_ERR_SYS_BUSY	系统忙。
0xA0028013	-	缓存小于实际需要的大小。
0xA0028014	-	硬件或软件处理超时。
0xA0028015	HI_ERR_SYS_ERR	内部系统错误。
0xA002803F	-	最大的返回码，该模块的错误码必须小于该值。

2.10.9.2 VPC 图像处理返回码

错误代码	宏定义	描述
0xA0078001	-	无效的Device ID。
0xA0078002	HI_ERR_VPC_INVALID_CHN_ID	通道 ID 超出合法范围。
0xA0078003	HI_ERR_VPC_ILLEGAL_PARAM	参数超出合法范围。

错误代码	宏定义	描述
0xA0078004	HI_ERR_VPC_EXIST	试图创建已经存在的通道。
0xA0078005	HI_ERR_VPC_UNEXIST	通道未创建或已销毁。
0xA0078006	HI_ERR_VPC_NULL_PTR	函数参数中指针地址为0。
0xA0078007	HI_ERR_VPC_NOT_CFG	使用前未配置。
0xA0078008	HI_ERR_VPC_NOT_SUPPORT	不支持的参数或者功能。
0xA0078009	HI_ERR_VPC_NOT_PERM	该操作不允许。
0xA007800C	HI_ERR_VPC_NO_MEM	分配内存失败，如系统内存不足。
0xA007800D	HI_ERR_VPC_NO_BUF	分配缓存失败，如申请的数据缓冲区太大。
0xA007800E	HI_ERR_VPC_BUF_EMPTY	缓冲区中无数据。
0xA007800F	HI_ERR_VPC_BUF_FULL	缓冲区中数据满。
0xA0078010	HI_ERR_VPC_SYS_NOT_READY	系统没有初始化或者相关依赖的模块没有加载。
0xA0078011	HI_ERR_VPC_BAD_ADDR	内存地址错误。 媒体数据处理功能涉及的输入、输出内存需调用 acl.himpi.dvpp_malloc 接口申请，请检查代码逻辑。
0xA0078012	HI_ERR_VPC_BUSY	系统忙。
0xA0078013	-	缓存小于实际需要的大小。
0xA0078014	-	硬件或软件处理超时。
0xA0078015	HI_ERR_VPC_SYS_ERROR	内部系统错误。
0xA007803F	-	最大的返回码，该模块的错误码必须小于该值。

2.10.9.3 VDEC 视频/JPEGD 图像解码返回码

错误代码	宏定义	描述
0xA0058001	-	无效的Device ID。
0xA0058002	HI_ERR_VDEC_INVALID_CHN_ID	通道 ID 超出合法范围。
0xA0058003	HI_ERR_VDEC_ILLEGAL_PARAM	参数超出合法范围。

错误代码	宏定义	描述
0xA0058004	HI_ERR_VDEC_EXIST	试图创建已经存在的通道。
0xA0058005	HI_ERR_VDEC_UNEXIST	通道未创建或已销毁。
0xA0058006	HI_ERR_VDEC_NULL_PTR	函数参数中指针地址为0。
0xA0058007	HI_ERR_VDEC_NOT_CFG	使用前未配置。
0xA0058008	HI_ERR_VDEC_NOT_SUPPORT	不支持的参数或者功能。
0xA0058009	HI_ERR_VDEC_NOT_PERM	该操作不允许。
0xA005800C	HI_ERR_VDEC_NO_MEM	分配内存失败，如系统内存不足。
0xA005800D	HI_ERR_VDEC_NO_BUF	分配缓存失败，如申请的数据缓冲区太大。
0xA005800E	HI_ERR_VDEC_BUF_EMPTY	缓冲区中无数据。
0xA005800F	HI_ERR_VDEC_BUF_FULL	缓冲区中数据满。
0xA0058010	HI_ERR_VDEC_SYS_NOT_READY	系统没有初始化或者相关依赖的模块没有加载。
0xA0058011	HI_ERR_VDEC_BAD_ADDR	内存地址错误。 媒体数据处理功能涉及的输入、输出内存需调用 acl.himpi.dvpp_malloc 接口申请，请检查代码逻辑。
0xA0058012	HI_ERR_VDEC_BUSY	系统忙。
0xA0058013	-	缓存小于实际需要的大小。
0xA0058014	-	硬件或软件处理超时。
0xA0058015	-	内部系统错误。
0xA005803F	-	最大的返回码，该模块的错误码必须小于该值。

2.10.9.4 VENC 视频/JPEGE 图像编码返回码

错误代码	宏定义	描述
0xA0088001	-	无效的Device ID。
0xA0088002	HI_ERR_VENC_INVALID_CHN_ID	通道 ID 超出合法范围。
0xA0088003	HI_ERR_VENC_ILLEGAL_PARAM	参数超出合法范围。

错误代码	宏定义	描述
0xA0088004	HI_ERR_VENC_EXIST	试图申请或者创建已经存在的设备、通道或者资源。
0xA0088005	HI_ERR_VENC_UNEXIST	试图使用或者销毁不存在的设备、通道或者资源。
0xA0088006	HI_ERR_VENC_NULL_PTR	函数参数中指针地址为0。
0xA0088007	HI_ERR_VENC_NOT_CFG	使用前未配置。
0xA0088008	HI_ERR_VENC_NOT_SUPPORT	不支持的参数或者功能。
0xA0088009	HI_ERR_VENC_NOT_PERM	该操作不允许，如试图修改静态配置参数。
0xA008800C	HI_ERR_VENC_NO_MEM	分配内存失败，如系统内存不足。
0xA008800D	HI_ERR_VENC_NO_BUF	分配缓存失败，如申请的数据缓冲区太大、频繁传入输入帧导致缓存区满等。
0xA008800E	HI_ERR_VENC_BUF_EMPTY	缓冲区中无数据。
0xA008800F	HI_ERR_VENC_BUF_FULL	缓冲区中数据满。
0xA0088010	HI_ERR_VENC_SYS_NOT_READY	系统没有初始化或没有加载相应模块。
0xA0088011	HI_ERR_VENC_BAD_ADDR	内存地址错误。 媒体数据处理功能涉及的输入、输出内存需调用 acl.himpi.dvpp_malloc 接口申请，请检查代码逻辑。
0xA0088012	HI_ERR_VENC_BUSY	系统忙。
0xA0088013	-	缓存小于实际需要的大小。
0xA0088014	-	硬件或软件处理超时。
0xA0088015	-	内部系统错误。
0xA008803F	-	最大的返回码，该模块的错误码必须小于该值。

2.10.9.5 PNGD 图像解码返回码

错误代码	宏定义	描述
0xA0408001	-	无效的Device ID。

错误代码	宏定义	描述
0xA0408002	HI_ERR_PNGD_INVALID_CHN_ID	通道 ID 超出合法范围。
0xA0408003	HI_ERR_PNGD_ILLEGAL_PARAM	参数超出合法范围。
0xA0408004	HI_ERR_PNGD_EXIST	试图创建已经存在的通道。
0xA0408005	HI_ERR_PNGD_UNEXIST	通道未创建或已销毁。
0xA0408006	HI_ERR_PNGD_NULL_PTR	函数参数中指针地址为0。
0xA0408007	HI_ERR_PNGD_NOT_CFG	使用前未配置。
0xA0408008	HI_ERR_PNGD_NOT_SUPPORT	不支持的参数或者功能。
0xA0408009	HI_ERR_PNGD_NOT_PERM	该操作不允许。
0xA040800C	HI_ERR_PNGD_NO_MEM	分配内存失败，如系统内存不足。
0xA040800D	HI_ERR_PNGD_NO_BUF	分配缓存失败，如申请的数据缓冲区太大。
0xA040800E	HI_ERR_PNGD_BUF_EMPTY	缓冲区中无数据。
0xA040800F	HI_ERR_PNGD_BUF_FULL	缓冲区中数据满。
0xA0408010	HI_ERR_PNGD_SYS_NOT_READY	系统没有初始化或者相关依赖的模块没有加载。
0xA0408011	HI_ERR_PNGD_BAD_ADDR	内存地址错误。 媒体数据处理功能涉及的输入、输出内存需调用 acl.himpi.dvpp_malloc 接口申请，请检查代码逻辑。
0xA0408012	HI_ERR_PNGD_BUSY	系统忙。
0xA0408013	-	缓存小于实际需要的大小。
0xA0408014	HI_ERR_PNGD_TIMEOUT	硬件或软件处理超时。
0xA0408015	HI_ERR_PNGD_SYS_ERROR	内部系统错误。
0xA040803F	-	最大的返回码，该模块的错误码必须小于该值。

2.11 算子加载与执行 (op)

2.11.1 函数：set_model_dir

C函数原型	<code>aclError aclopSetModelDir(const char *modelDir)</code>
Python函数	<code>ret = acl.op.set_model_dir(model_dir)</code>
函数功能	设置加载模型文件的目录，目录下存放单算子模型文件（*.om文件）。
输入说明	model_dir : str，表示指定模型文件所在目录。 此处可设置多级目录，但系统最多从多级目录的最后一级开始，读取三级目录下的模型文件。 例如，将modelDir设置为"dir0/dir1"，dir1下的目录层级为“dir2/dir3/dir4”，这时系统只支持读取“dir1”、“dir1/dir2”、“dir1/dir2/dir3”目录下的模型文件。
返回值说明	ret : int，错误码。 - 返回0表示成功。 - 返回其它值表示失败。
约束说明	在加载前，请先根据文件的大小评估内存空间是否足够，内存空间不足，会导致应用程序异常。 Atlas 200/300/500 推理产品 ，在Host上运行应用时，一个进程内正在执行的算子的最大个数上限是40000000，在Device上运行应用时，一个进程内正在执行的算子的最大个数上限是2000000。 Atlas 200I/500 A2推理产品 ，在Host上运行应用时，一个进程内正在执行的算子的最大个数上限是40000000。 Atlas 训练系列产品 ，在Host上运行应用时，一个进程内正在执行的算子的最大个数上限是40000000。 Atlas A2训练系列产品/Atlas 800I A2推理产品 ，在Host上运行应用时，一个进程内正在执行的算子的最大个数上限是2000000。 Atlas 推理系列产品 ，在Host上运行应用时，一个进程内正在执行的算子的最大个数上限是40000000，在Device上运行应用时，一个进程内正在执行的算子的最大个数上限是2000000。
参考资源	接口调用流程、示例，参见单算子调用。

2.11.2 函数：load

C函数原型	<code>aclError aclopLoad(const void *model, size_t modelSize)</code>
Python函数	<code>ret = acl.op.load(model, model_size)</code>
函数功能	从内存中加载单算子模型数据，由用户管理内存。
输入说明	model : int，表示单算子模型数据的地址对象。 model_size : int，表示内存中的模型数据长度，单位Byte。

返回值说明	ret : int, 错误码。 - 返回0表示成功。 - 返回其它值表示失败。
约束说明	在加载前, 请先根据单算子om文件的大小评估内存空间是否足够, 内存空间不足, 会导致应用程序异常。 Atlas 200/300/500 推理产品, 在Host上运行应用时, 一个进程内正在执行的算子的最大个数上限是40000000, 在Device上运行应用时, 一个进程内正在执行的算子的最大个数上限是2000000。 Atlas 200I/500 A2推理产品, 在Host上运行应用时, 一个进程内正在执行的算子的最大个数上限是40000000。 Atlas 训练系列产品, 在Host上运行应用时, 一个进程内正在执行的算子的最大个数上限是40000000。 Atlas A2训练系列产品/Atlas 800I A2推理产品, 在Host上运行应用时, 一个进程内正在执行的算子的最大个数上限是2000000。 Atlas 推理系列产品, 在Host上运行应用时, 一个进程内正在执行的算子的最大个数上限是40000000, 在Device上运行应用时, 一个进程内正在执行的算子的最大个数上限是2000000。
参考资源	接口调用流程, 参见单算子调用。

2.11.3 函数: set_op_execute_time_out

C函数原型	aclError aclrtSetOpExecuteTimeOut(uint32_t timeout)
Python函数	ret = acl.rt.set_op_execute_time_out(timeout)
函数功能	设置算子执行的超时时间。 Atlas 200/300/500 推理产品 , 不支持该接口。
输入说明	timeout : int, 设置超时时间, 单位为秒。将该参数设置为0时, 表示使用默认超时时间。
返回值说明	ret : int, 错误码。 - 返回0表示成功。 - 返回其它值表示失败。

约束说明

- 不调用本接口, 算子执行的默认超时时间如下:

版本	AI Core算子的默认超时时间	AI CPU算子的默认超时时间
Atlas 推理系列产品	547秒	28秒
Atlas 训练系列产品	68秒	28秒

版本	AI Core算子的默认超时时间	AI CPU算子的默认超时时间
Atlas A2训练系列产品/Atlas 800I A2推理产品	1091秒	28秒
Atlas 200I/500 A2推理产品	1091秒	28秒

- 一个进程内多次调用本接口，则以最后一次设置的时间为准，允许设置的最大时间如下：

版本	AI Core算子的最大超时时间	AI CPU算子的最大超时时间
Atlas 推理系列产品	547秒	不支持设置
Atlas 训练系列产品	2176秒	2176秒
Atlas A2训练系列产品/Atlas 800I A2推理产品	1091秒	1091秒
Atlas 200I/500 A2推理产品	1091秒	1091秒

2.11.4 函数：execute

须知

此接口后续版本会废弃，请使用[acl.op.execute_v2](#)接口。

C函数原型	<code>aclError aclOpExecute(const char *opType, int numInputs, const aclTensorDesc *const inputDesc[], const aclDataBuffer *const inputs[], int numOutputs, const aclTensorDesc *const outputDesc[], aclDataBuffer *const outputs[], const aclOpAttr *attr, aclrtStream stream);</code>
Python函数	<code>ret = acl.op.execute(op_type, input_desc, inputs, output_desc, outputs, attr, stream)</code>
函数功能	执行指定的算子。异步接口。

输入说明	op_type: str, 指定算子类型名称。 input_desc: list, 表示算子输入Tensor的描述, 整形列表, 包含多个acl的Tensor描述地址对象。 inputs: list, 表示算子输入Tensor, 整形列表, 包含多个aclDataBuffer数据地址对象。 output_desc: list, 表示算子输出Tensor的描述, 整形列表, 包含多个acl的Tensor描述地址对象。 outputs: list, 表示算子输出Tensor, 整形列表, 包含多个aclDataBuffer数据地址对象。 attr: int, 算子的属性地址对象。 stream: int, 该算子需要加载的Stream对象。
返回值说明	ret: int, 错误码。 • 返回0表示成功。 • 返回其它值表示失败。
约束说明	• 该接口是异步接口, 调用接口成功仅表示任务下发成功, 不表示任务执行成功。调用该接口后, 需调用同步等待接口 (例如, acl.rt.synchronize_stream) 确保任务已执行完成。 • 多线程场景下, 不支持调用本接口时指定同一个Stream或使用默认Stream, 否则可能任务执行异常。 • 每个算子的输入、输出、属性不同, 需要应用在调用时严格按照算子输入、输出、属性来组织算子。用户在调用acl.op.execute接口时, pyACL根据op_type、输入Tensor的描述、输出Tensor的描述、attr等信息查找对应的任务, 并下发执行。

2.11.5 函数: execute_v2

C函数原型	<pre>aclError aclOpExecuteV2(const char *opType, int numInputs, aclTensorDesc *inputDesc[], aclDataBuffer *inputs[], int numOutputs, aclTensorDesc *outputDesc[], aclDataBuffer *outputs[], aclOpAttr *attr, aclrtStream stream);</pre>
Python函数	<pre>ret = acl.op.execute_v2(op_type, input_desc, inputs, output_desc, outputs, attr, stream)</pre>
函数功能	执行指定的算子。

输入说明	op_type: str, 指定算子类型名称。 input_desc: list, 表示算子输入Tensor的描述, 整形列表, 包含多个acl的Tensor描述地址对象。 inputs: list, 表示算子输入Tensor, 整形列表, 包含多个aclDataBuffer数据地址对象。 output_desc: list, 表示算子输出Tensor的描述, 整形列表, 包含多个acl的Tensor描述地址对象。 outputs: list, 表示算子输出Tensor, 整形列表, 包含多个aclDataBuffer数据地址对象。 attr: int, 算子的属性地址对象。 stream: int, 该算子需要加载的Stream对象。
返回值说明	ret: int, 错误码。 • 返回0表示成功。 • 返回其它值表示失败。

约束说明	该接口是异步接口，调用接口成功仅表示任务下发成功，不表示任务执行成功。调用该接口后，需调用同步等待接口（例如，acl.rt.synchronize_stream）确保任务已执行完成。 多线程场景下，不支持调用本接口时指定同一个Stream或使用默认Stream，否则可能任务执行异常。 每个算子的输入、输出、属性不同，需要应用在调用时严格按照算子输入、输出、属性来组织算子。用户在调用acl.op.execute_v2接口时，pyACL根据op_type、输入Tensor的描述、输出Tensor的描述、attr等信息查找对应的任务，并下发执行。 对于支持动态Shape的算子，无法明确算子输出Shape时，可调用acl.op.infer_shape接口获取算子的输出Shape： • 若可获取到准确的输出Shape，则使用准确的输出Shape来构造outputDesc参数，作为acl.op.execute_v2的输入。该场景下，acl.op.execute_v2接口是异步接口，对于异步接口，调用接口成功仅表示任务下发成功，不表示任务执行成功。调用该接口后，需调用同步等待接口（例如，acl.rt.synchronize_stream）确保任务已执行完成。 • 若无法获取准确的输出Shape，仅能获取输出Shape范围，则使用Shape最大值来构造outputDesc参数，作为acl.op.execute_v2的输入。该场景下，在调用acl.op.execute_v2接口执行算子后，系统内部会计算出准确的输出Shape，通过acl.op.execute_v2接口的outputDesc参数输出，此时acl.op.execute_v2接口是同步接口。 • （该场景预留）若无法获取准确的输出Shape以及Shape范围，则需由用户预估一个最大的Shape来构造outputDesc参数，作为acl.op.execute_v2的输入。该场景下，在调用acl.op.execute_v2接口执行算子后，系统内部会计算出准确的输出Shape，通过acl.op.execute_v2接口的outputDesc参数输出，此时acl.op.execute_v2接口是同步接口。 执行有可选输入的算子时，如果可选输入不使用： • 则需按此种方式创建acl.TensorDesc类型的数据：acl.create_tensor_desc(ACL_DT_UNDEFINED, 0, [], ACL_FORMAT_UNDEFINED)，表示数据类型设置为ACL_DT_UNDEFINED，Format设置为ACL_FORMAT_UNDEFINED，Shape信息为[]。 • 则需按此种方式创建acl.DataBuffer类型的数据：acl.create_data_buffer([], 0)，同时acl.DataBuffer中的数据不需要释放。 执行有constant输入的算子时，需要先调用acl.set_tensor_const接口设置constant输入。 调用acl.op.execute_v2接口时，设置的constant输入数据必须保持一致。 对于算子有constant输入的场景，如果未调用acl.set_tensor_const接口设置constant输入，则需调用acl.set_tensor_placement设置TensorDesc的placement属性，将memType设置为Host内存。 在执行单算子时，一般要求用户传入的输入/输出Tensor数据是存放在Device内存的，比如两个Tensor相加的场景。但是，也存在部

	分算子，除了将featureMap、weight等Device内存中的Tensor数据作为输入外，还需Tensor shape信息、learningRate、Tensor的dims信息等通常在Host内存中的Tensor数据也作为输入，此时，用户不需要额外将这些Host内存中的Tensor数据拷贝到Device上作为输入，只需要调用 acl.set_tensor_placement 接口设置对应TensorDesc的placement属性为Host内存，在执行单算子时，设置了placement属性为Host内存的TensorDesc，其对应的Tensor数据必须存放在Host内存中，pyACL内部会自动将Host上的Tensor数据拷贝到Device上。
参考资源	接口调用流程、示例，参见单算子调用。

2.11.6 函数：execute_with_handle

C函数原型	<code>aclError aclOpExecWithHandle(aclOpHandle *handle, int numInputs, const aclDataBuffer *const inputs[], int numOutputs, aclDataBuffer *const outputs[], aclrtStream stream);</code>
Python函数	<code>ret = acl.op.execute_with_handle(handle, inputs, outputs, stream)</code>
函数功能	以Handle方式调用一个算子，不支持动态Shape算子，动态Shape算子请使用 acl.op.execute_v2 。异步接口。
输入说明	handle: int，指定执行算子的handle，指针地址。 需提前调用acl.op.create_handle接口创建aclOpHandle类型的数据。 inputs: list，算子输入Tensor，整形列表，包含多个aclDataBuffer数据地址对象。 需提前调用acl.create_data_buffer接口创建aclDataBuffer类型的数据。 outputs: list，算子输出Tensor，整形列表，包含多个aclDataBuffer数据地址对象。 需提前调用acl.create_data_buffer接口创建aclDataBuffer类型的数据。 stream: int，执行算子所在的Stream，stream地址对象。
返回值说明	ret: int，错误码。 - 返回0表示成功。 - 返回其它值表示失败。

约束说明	- 该接口是异步接口，调用接口成功仅表示任务下发成功，不表示任务执行成功。调用该接口后，需调用同步等待接口（例如，acl.rt.synchronize_stream）确保任务已执行完成。 - 多线程场景下，不支持调用本接口时指定同一个Stream或使用默认Stream，否则可能任务执行异常。 - 执行有可选输入的算子时，如果可选输入不使用，则需参考以下方式创建aclDataBuffer类型的数据。 <code>acl.create_data_buffer(nullptr, 0)</code> 同时aclDataBuffer中的数据不需要释放，因为是空的数据类型。

2.11.7 函数：cast

C函数原型	<code>aclError aclopCast(const aclTensorDesc *srcDesc, const aclDataBuffer *srcBuffer, const aclTensorDesc *dstDesc, aclDataBuffer *dstBuffer, uint8_t truncate, aclrtStream stream)</code>
Python函数	<code>ret = acl.op.cast(src_desc, src_buffer, dst_desc, dst_buffer, truncate, stream)</code>
函数功能	转换输入数据的数据类型，异步接口。
输入说明	src_desc: int，输入Tensor描述指针地址。 src_buffer: int，输入Tensor，调用acl.create_data_buffer创建的aclDataBuffer指针地址。 dst_desc: int，输出Tensor的描述指针地址。 dst_buffer: int，输出Tensor，调用acl.create_data_buffer创建的aclDataBuffer指针地址。 truncate: int，预留。 stream: int，执行算子所在的Stream，Stream地址对象。
返回值说明	ret: int，错误码。 - 返回0表示成功。 - 返回其它值表示失败。
约束说明	无
参考资源	接口调用示例，参见接口调用流程。

2.11.8 函数：create_handle_for_cast

C函数原型	<code>aclError aclopCreateHandleForCast(const aclTensorDesc *srcDesc, const aclTensorDesc *dstDesc, uint8_t truncate, aclopHandle **handle)</code>

Python函数	handle, ret = acl.op.create_handle_for_cast(src_desc, dst_desc, truncate)
函数功能	创建数据类型转换的handle。 创建handle成功后，需调用 acl.op.execute_with_handle 接口执行算子。
输入说明	src_desc : int, 输入Tensor描述指针地址。 dst_desc : int, 输出Tensor描述指针地址。 truncate : int, 预留。
返回值说明	handle : int, 输出的handle的指针地址。 ret : int, 错误码。 • 返回0表示成功。 • 返回其它值表示失败
约束说明	无

2.11.9 函数：create_attr

C函数原型	aclopAttr *aclopCreateAttr()
Python函数	attr = acl.op.create_attr()
函数功能	创建aclopAttr类型的数据，该数据类型用于保存算子的属性。 如需销毁aclopAttr类型的数据，请参见 acl.op.destroy_attr 。
输入说明	无
返回值说明	attr : int, 返回aclopAttr类型数据的指针地址。
约束说明	无

2.11.10 函数：destroy_attr

C函数原型	void aclopDestroyAttr(const aclopAttr *attr)
Python函数	acl.op.destroy_attr(attr)
函数功能	销毁通过 acl.op.create_attr 接口创建的aclopAttr类型的数据。
输入说明	attr : int, 待销毁的aclopAttr类型数据指针地址，此对象使用调用 acl.op.create_attr 创建。
返回值说明	无
约束说明	无

2.11.11 函数：set_attr_bool

C函数原型	<code>aclError aclopSetAttrBool(aclopAttr *attr, const char *attrName, uint8_t attrValue)</code>
Python函数	<code>ret = acl.op.set_attr_bool(attr, attr_name, attr_value)</code>
函数功能	设置bool类型标量的属性值。
输入说明	attr : int, aclopAttr类型数据的指针地址。 需提前调用 acl.op.create_attr 接口创建aclopAttr类型数据。 attr_name : str, 属性名。 attr_value : int, 属性值。
返回值说明	ret : int, 错误码。 - 返回0表示成功。 - 返回其它值表示失败。
约束说明	无

2.11.12 函数：set_attr_int

C函数原型	<code>aclError aclopSetAttrInt(aclopAttr *attr, const char *attrName, int64_t attrValue);</code>
Python函数	<code>ret = acl.op.set_attr_int(attr, attr_name, attr_value)</code>
函数功能	设置int64_t类型标量的属性值。
输入说明	attr : int, aclopAttr类型数据的指针地址。 需提前调用 acl.op.create_attr 接口创建aclopAttr类型数据。 attr_name : str, 属性名。 attr_value : int, 属性值。
返回值说明	ret : int, 错误码。 - 返回0表示成功。 - 返回其它值表示失败。
约束说明	无

2.11.13 函数：set_attr_float

C函数原型	<code>aclError aclopSetAttrFloat(aclopAttr *attr, const char *attrName, float attrValue);</code>
Python函数	<code>ret = acl.op.set_attr_float(attr, attr_name, attr_value)</code>

函数功能	设置float类型标量的属性值。
输入说明	attr : int, aclopAttr类型数据的指针地址。 需提前调用 acl.op.create_attr 接口创建aclopAttr类型数据。 attr_name : str, 属性名。 attr_value : float, 属性值。
返回值说明	ret : int, 错误码。 - 返回0表示成功。 - 返回其它值表示失败。
约束说明	无

2.11.14 函数：set_attr_string

C函数原型	aclError aclopSetAttrString(aclopAttr *attr, const char *attrName, const char* attrValue);
Python函数	ret = acl.op.set_attr_string(attr, attr_name, attr_value)
函数功能	设置字符串类型标量的属性值。
输入说明	attr : int, aclopAttr类型数据的指针地址。 需提前调用 acl.op.create_attr 接口创建aclopAttr类型数据。 attr_name : str, 属性名。 attr_value : str, 属性值。
返回值说明	ret : int, 错误码。 - 返回0表示成功。 - 返回其它值表示失败。
约束说明	无

2.11.15 函数：set_attr_list_bool

C函数原型	aclError aclopSetAttrListBool(aclopAttr *attr, const char *attrName, int numValues, const uint8_t *attrValue);
Python函数	ret = acl.op.set_attr_list_bool(attr, attr_name, attr_value)
函数功能	设置bool类型列表的属性值。

输入说明	attr : int, aclopAttr类型数据的指针地址。 需提前调用 acl.op.create_attr 接口创建aclopAttr类型数据。 attr_name : str, 属性名。 attr_value : 同时支持两种数据类型作为输入。 - list, list中每个元素的数据类型为int, 取值范围为0 ~ 255。 - numpy, 一维数组numpy对象, 类型为uint8。
返回值说明	ret : int, 错误码。 - 返回0表示成功。 - 返回其它值表示失败。
约束说明	- 建议“attr_value”使用list数据类型, 后续将不支持numpy作为输入。 - 若“attr_value”需使用numpy数组, 需要运行环境为python ≥ 3.8且numpy ≥ 1.22.0。

2.11.16 函数: set_attr_list_int

C函数原型	aclError aclopSetAttrListInt (aclopAttr *attr, const char *attrName, int numValues, const int64_t *attrValue);
Python函数	ret = acl.op.set_attr_list_int (attr, attr_name, attr_value)
函数功能	设置int64_t类型列表的属性值。
输入说明	attr : int, aclopAttr类型数据的指针地址。 需提前调用 acl.op.create_attr 接口创建aclopAttr类型数据。 attr_name : str, 属性名。 attr_value : 同时支持两种数据类型作为输入。 - list, list中每个元素的数据类型为int。 - numpy, 一维数组numpy对象, 类型为int64。
返回值说明	ret : int, 错误码。 - 返回0表示成功。 - 返回其它值表示失败。
约束说明	- 建议“attr_value”使用list数据类型, 后续将不支持numpy作为输入。 - 若“attr_value”需使用numpy数组, 需要运行环境为python ≥ 3.8且numpy ≥ 1.22.0。

2.11.17 函数：set_attr_list_float

C函数原型	<code>aclError aclOpSetAttrListFloat(aclOpAttr *attr, const char *attrName, int numValues, const float *attrValue);</code>
Python函数	<code>ret = acl.op.set_attr_list_float(attr, attr_name, num_values , attr_value)</code>
函数功能	设置float类型列表的属性值。
输入说明	attr : int, aclOpAttr类型数据的指针地址。 需提前调用 acl.op.create_attr 接口创建aclOpAttr类型数据。 attr_name : str, 属性名。 attr_value : 同时支持两种数据类型作为输入。 - list, list中每个元素的数据类型为float。 - numpy, 一维数组numpy对象, 类型为float32。
返回值说明	ret : int, 错误码。 - 返回0表示成功。 - 返回其它值表示失败。
约束说明	- 建议“attr_value”使用list数据类型, 后续将不支持numpy作为输入。 - 若“attr_value”需使用numpy数组, 需要运行环境为python ≥ 3.8且numpy ≥ 1.22.0。

2.11.18 函数：set_attr_list_string

C函数原型	<code>aclError aclOpSetAttrListString(aclOpAttr *attr, const char *attrName, int numValues, const char **attrValue);</code>
Python函数	<code>ret = acl.op.set_attr_list_string(attr, attr_name , attr_value)</code>
函数功能	设置字符串类型列表的属性值。
输入说明	attr : int, aclOpAttr类型数据的指针地址。 需提前调用 acl.op.create_attr 接口创建aclOpAttr类型数据。 attr_name : str, 属性名。 attr_value : list, 每个元素是字符串的列表。
返回值说明	ret : int, 错误码。 - 返回0表示成功。 - 返回其它值表示失败。
约束说明	无

2.11.19 函数：set_attr_list_list_int

C函数原型	<code>aclError aclopSetAttrListListInt(aclopAttr *attr, const char *attrName, int numLists, const int *numValues, const int64_t *const values[]);</code>
Python函数	<code>ret = acl.op.set_attr_list_list_int(attr, attr_name , values)</code>
函数功能	设置int64_t类型列表的列表属性值。
输入说明	attr : int, aclopAttr类型数据的指针地址。 需提前调用 acl.op.create_attr 接口创建aclopAttr类型数据。 attr_name : str, 属性名。 values : list, 列表中每个元素是一个list (list的每个元素数据类型为int) 或一个一维数组numpy对象 (numpy对象的类型为int64) 。
返回值说明	ret : int, 错误码。 • 返回0表示成功。 • 返回其它值表示失败。
约束说明	• 建议“values”使用list数据类型, 后续将不支持numpy作为输入。 • 若“values”需使用numpy数组, 需要运行环境为python ≥ 3.8 且numpy ≥ 1.22.0。

2.11.20 函数：create_handle

C函数原型	<code>aclError aclopCreateHandle(const char *opType, int numInputs,int numInputs, const aclTensorDesc *const inputDesc[], int numOutputs, const aclTensorDesc *const outputDesc[], const aclopAttr *opAttr, aclopHandle **handle);</code>
Python函数	<code>handle, ret = acl.op.create_handle(op_type, input_desc, output_desc, op_attr)</code>
函数功能	创建一个执行算子的handle。 如需销毁handle, 请参见 acl.op.destroy_handle 。
输入说明	op_type : str, 指定算子类型名称字符串类型。 input_desc : list, 算子输入Tensor的描述, 整形列表。 output_desc : list, 算子输出Tensor的描述, 整形列表。 op_attr : int, 算子属性指针地址, 此对象使用调用 acl.op.create_attr 创建。

返回值说明	handle : int, 算子handle地址对象。 ret : int, 错误码。 • 返回0表示成功。 • 返回其它值表示失败。
约束说明	对于算子有constant输入的场景, 如果未调用 acl.set_tensor_const 接口设置constant输入, 则需调用 acl.set_tensor_placement 设置TensorDesc的placement属性, 将“mem_type”设置为Host内存。

2.11.21 函数: destroy_handle

C函数原型	void aclOpDestroyHandle(aclOpHandle *handle)
Python函数	acl.op.destroy_handle (handle)
函数功能	销毁一个执行算子的handle。
输入说明	handle : int, 待销毁的acl的算子Handle的指针地址。
返回值说明	无
约束说明	无

2.11.22 函数: infer_shape

C函数原型	aclError aclOpInferShape(const char *opType, int numInputs, aclTensorDesc *inputDesc[], aclDataBuffer *inputs[], int numOutputs, aclTensorDesc *outputDesc[], aclOpAttr *attr)
Python函数	ret = acl.op.infer_shape (op_type, in_desc_list, in_list, num_outputs, out_desc_list, attr)
函数功能	根据算子的输入Shape、输入值推导出算子的输出Shape。
输入说明	op_type : int, 指定算子类型名称。 in_desc_list : list, 算子输入Tensor的描述。需提前调用 acl.create_tensor_desc 接口创建aclTensorDesc类型。 in_list : list, 算子输入Tensor。 此处算子输入Tensor数据的内存必须根据应用运行模式来确定, 应用运行在Host时, 此处需申请Host上的内存。应用运行在Device时, 此处需申请Device上的内存。内存申请接口请参见 内存管理 。 num_outputs : int, 算子输出Tensor的数量。 attr : int, 算子的属性地址对象。

返回值说明	out_desc_list: list, 算子输出Tensor的描述。需提前调用acl.create_tensor_desc接口创建aclTensorDesc类型。 “out_desc_list”列表中的元素个数必须与“num_outputs”参数值保持一致。 ret: int, 错误码。 • 返回0表示成功。 • 返回其它值表示失败。
约束说明	• 推导算子的输出Shape包含三种场景： - 根据Shape推导, 可以得到算子的准确输出Shape, 则返回准确输出Shape。 - 根据Shape推导, 无法得到算子的准确输出Shape, 但可以得到输出Shape的范围, 则在输出参数out_desc_list中将算子输出Tensor描述中的动态维度的维度值记为-1。该场景下, 用户可调用get_tensor_desc_dim_range接口获取Tensor描述中指定维度的范围值。 - (该场景预留) 根据Shape推导, 无法得到算子的准确输出Shape以及Shape范围, 则在输出参数out_desc_list中将算子输出Tensor描述中的动态维度的维度值记为-2。 • 如果算子有动态输入或动态输出, 在调用acl.op.infer_shape接口推导算子的输出Shape前, 需先调用set_tensor_desc_name接口设置所有输入和输出的Tensor描述的名称, 且名称必须按照如下要求(算子IR原型中定义的输入/输出名称请参见《算子清单》。) - 对于必选输入、可选输入、必选输出, 名称必须与算子IR原型中定义的输入/输出名称保持一致。 - 对于动态输入、动态输出, 名称必须是: 算子IR原型中定义的输入/输出名称+编号。编号根据动态输入/输出的个数确定, 从0开始, 0对应第一个动态输入/输出, 1对应第二个动态输入/输出, 以此类推。 例如某个算子有2个输入(第1个是必选输入x, 第二个是动态输入y且输入个数为2)、1个必选输出z, 则调用set_tensor_desc_name接口设置名称的代码示例如下: <pre>acl.set_tensor_desc_name(input_tensor_desc[0], "x"); acl.set_tensor_desc_name(input_tensor_desc[1], "y0"); acl.set_tensor_desc_name(input_tensor_desc[2], "y1"); acl.set_tensor_desc_name(input_tensor_desc[3], "z");</pre>

2.11.23 函数：set_attr_data_type

C函数原型	<code>aclError aclopSetAttrDataType(aclopAttr *attr, const char *attrName, aclDataType attrValue)</code>
Python函数	<code>attr, ret = acl.op.set_attr_data_type(attr, attr_name, attr_value)</code>
函数功能	设置指定属性的数据类型。

输入说明	attr_name : str, 属性名。 attr_value : int, 属性值。
返回值说明	attr : int, aclopAttr类型的指针地址。需提前调用 acl.op.create_attr 接口创建aclopAttr类型。 ret : int, 错误码。 - 返回0表示成功。 - 返回其它值表示失败。
约束说明	无

2.11.24 函数: set_attr_list_data_type

C函数原型	aclError aclopSetAttrListDataType (aclopAttr *attr, const char *attrName, int numValues, const aclDataType values[])
Python函数	attr, ret = acl.op.set_attr_list_data_type (attr, attr_name, values)
函数功能	设置指定属性列表的数据类型。
输入说明	attr_name : str, 属性名。 values : list, 属性值的列表。
返回值说明	attr : int, aclopAttr类型的指针地址。需提前调用 acl.op.create_attr 接口创建aclopAttr类型。 ret : int, 错误码。 - 返回0表示成功。 - 返回其它值表示失败。
约束说明	无

2.12 CBLAS 接口 (blas)

2.12.1 总体说明

- 调用本章节接口实现“矩阵-矩阵”或“矩阵-向量”相乘的运算，本章中的接口内部封装了系统内置的矩阵乘算子GEMM，接口调用流程参见接口调用流程。
按照接口调用流程，在使用本章节的接口前，需要先使用ATC（Ascend Tensor Compiler）工具将内置的矩阵乘算子GEMM的算子描述信息（包括输入输出Tensor描述、算子属性等）编译成适配昇腾AI处理器的离线模型（*.om文件），用于验证矩阵乘算子GEMM的运行结果。
- 本章的CBLAS接口都是异步接口，调用接口成功仅表示任务下发成功，不表示任务执行成功。调用该接口后，需调用同步等待接口（例如，[acl.rt.synchronize_stream](#)）确保任务已执行完成。

2.12.2 关于 A、B、C 矩阵的 Shape 及内存大小计算公式

- A、B、C 矩阵是否转置的标记如果设置为 ACL_TRANS_N 或 ACL_TRANS_T，则用户在申请内存存放 A、B、C 矩阵数据时，实际申请的内存要和实际数据大小匹配，Shape 及内存大小的计算公式如下：
 - A 矩阵：shape = (m, k)；内存大小 = $m * k * \text{acl.data\_type\_size}(\text{data\_type\_a})$
 - B 矩阵：shape = (k, n)；内存大小 = $k * n * \text{acl.data\_type\_size}(\text{data\_type\_b})$
 - C 矩阵：shape = (m, n)；内存大小 = $m * n * \text{acl.data\_type\_size}(\text{data\_type\_c})$
- A、B、C 矩阵是否转置的标记如果设置为 ACL_TRANS_NZ，表示采用内部数据格式，矩阵 Shape 为 4 维，Shape 及内存大小的计算公式如下（假设 m, k, n 分别为原始轴）：
 - 当矩阵 A 和矩阵 B 中数据的类型为 aclFloat16，在计算实际内存大小时，m、k、n 均按 16 对齐向上取整计算：
 - A 矩阵：shape = ([k/16], [m/16], 16, 16)；内存大小 = $[m/16] * 16 * [k/16] * 16 * \text{acl.data\_type\_size}(\text{data\_type\_a})$
 - B 矩阵：shape = ([n/16], [k/16], 16, 16)；内存大小 = $[k/16] * 16 * [n/16] * 16 * \text{acl.data\_type\_size}(\text{data\_type\_b})$
 - C 矩阵：shape = ([n/16], [m/16], 16, 16)；内存大小 = $[m/16] * 16 * [n/16] * 16 * \text{acl.data\_type\_size}(\text{data\_type\_c})$
 - 当矩阵 A 和矩阵 B 中数据的类型为 int8_t，在计算实际内存大小时，reduce 轴按 32 对齐向上取整计算，非 reduce 轴按 16 对齐向上取整计算：
 - A 矩阵：shape = ([k/32], [m/16], 16, 32)；内存大小 = $[m/16] * 16 * [k/32] * 32 * \text{acl.data\_type\_size}(\text{data\_type\_a})$
 - B 矩阵：shape = ([k/32], [n/16], 32, 16)；内存大小 = $[k/32] * 32 * [n/16] * 16 * \text{acl.data\_type\_size}(\text{data\_type\_b})$
 - C 矩阵：shape = ([n/16], [m/16], 16, 16)；内存大小 = $[m/16] * 16 * [n/16] * 16 * \text{acl.data\_type\_size}(\text{data\_type\_c})$

说明

[] 表示向上对齐。

2.12.3 函数：gemv_ex

C函数原型	<pre>aclError aclblasGemvEx(aclTransType transA, int m, int n, const void *alpha, const void *a, int lda, aclDataType dataTypeA, const void *x, int incx, aclDataType dataTypeX, const void *beta, void *y, int incy, aclDataType dataTypeY, aclComputeType type, aclrtStream stream);</pre>
Python函数	<pre>ret = acl.blas.gemv_ex(trans_a, m, n,alpha, a, lda, data_type_a, x, incx, data_type_x, beta, y, incy, data_type_y, type, stream)</pre>
函数功能	执行矩阵-向量的乘法， $y = \alpha Ax + \beta y$ ，输入数据、输出数据的数据类型通过入参设置，异步接口。

输入说明	trans_a: int, 矩阵A是否转置的标记。 m: int, 矩阵A的行数, 存储矩阵乘数据时, 行优先。 n: int, 矩阵A的列数。 alpha: int, 用于执行乘操作的标量alpha数据指针地址。 a: int, 矩阵A的数据指针地址。 lda: int, A矩阵的主位, 此时选择转置, 按行优先, 则lda为A的列数。预留参数, 当前只能设置为-1。 data_type_a: int, 矩阵A的数据类型。 x: int, 向量x的数据指针地址。 incx: int, x连续元素之间的步长。预留参数, 当前只能设置为-1。 data_type_x: int, 向量x的数据类型。 beta: int, 用于执行乘操作的标量beta的数据指针地址。 y: int, 向量y的数据指针地址。 incy: int, y连续元素之间的步长。预留参数, 当前只能设置为-1。 data_type_y: int, 向量y的数据类型。 type: int, 计算精度, 默认高精度。 stream: int, 执行算子所在的Stream。
返回值说明	ret: int, 错误码。 • 返回0表示成功。 • 返回其它值表示失败。
约束说明	A、x、y的数据类型支持仅支持以下组合, α和β的数据类型与y一致。 A的数据类型 x的数据类型 y的数据类型 aclFloat16 aclFloat16 aclFloat16 aclFloat16 aclFloat16 float(float32) int8_t int8_t float(float32) int8_t int8_t int32_t
参考资源	接口调用流程, 参见接口调用流程。

2.12.4 函数：create_handle_for_gemv_ex

C函数原型	aclError aclblasCreateHandleForGemvEx(aclTransType transA, int m, int n, aclDataType dataTypeA, aclDataType dataTypeX, aclDataType dataTypeY, aclComputeType type, aclopHandle **handle)
Python函数	handle, ret = acl.blas.create_handle_for_gemv_ex(trans_a, m, n, data_type_a, data_type_x, data_type_y, type)
函数功能	创建矩阵-向量乘的handle，输入数据、输出数据的数据类型通过入参设置。 创建handle成功后，需调用 acl.op.execute_with_handle 接口执行算子。
输入说明	trans_a : int，矩阵A是否转置的标记。 m : int，矩阵A的行数，存储矩阵乘数据时，行优先。 n : int，矩阵A的列数。 data_type_a : int，矩阵A的数据类型。 data_type_x : int，向量x的数据类型。 data_type_y : int，向量y的数据类型。 type : int，计算精度。
返回值说明	handle : int，执行算子的handle数据的指针地址。 ret : int，错误码。 - 返回0表示成功。 - 返回其它值表示失败。
约束说明	A、x、y的数据类型支持仅支持以下组合： A的数据类型 x的数据类型 y的数据类型 aclFloat16 aclFloat16 aclFloat16 aclFloat16 aclFloat16 float(float32) int8_t int8_t float(float32) int8_t int8_t int32_t

2.12.5 函数：hgemv

C函数原型	aclError aclblasHgemv(aclTransType transA, int m, int n, const aclFloat16 *alpha, const aclFloat16 *a, int lda, const aclFloat16 *x, int incx, const aclFloat16 *beta, aclFloat16 *y, int incy, aclComputeType type, aclrtStream stream)
Python函数	ret = acl.blas.hgemv(trans_a, m, n, alpha, a, lda, x, incx, beta, y, incy, type, stream)
函数功能	执行矩阵-向量的乘法， $y = \alpha Ax + \beta y$ ，输入数据和输出数据的数据类型为 aclFloat16 ，异步接口。
输入说明	trans_a : int，矩阵A是否转置的标记。 m : int，矩阵A的行数，存储矩阵乘数据时，行优先。 n : int，矩阵A的列数。 alpha : int，用于执行乘操作的标量alpha数据指针地址。 a : int，矩阵A的数据指针地址。 lda : int，A矩阵的主位，此时选择转置，按行优先，则lda为A的列数。预留参数，当前只能设置为-1。 x : int，向量x的数据指针地址。 incx : int，x连续元素之间的步长。预留参数，当前只能设置为-1。 beta : int，用于执行乘操作的标量beta的数据指针地址。 y : int，向量y的数据指针地址。 incy : int，y连续元素之间的步长。预留参数，当前只能设置为-1。 type : int，计算精度。 stream : int，执行算子所在的Stream。
返回值说明	ret : int，错误码。 • 返回0表示成功。 • 返回其它值表示失败。
约束说明	无

参考资源	接口调用流程，参见接口调用流程。

2.12.6 函数：create_handle_for_hgemv

C函数原型	<code>aclError aclblasCreateHandleForHgemv(aclTransType transA, int m, int n, aclComputeType type, aclophandle **handle)</code>
Python函数	<code>handle, ret = acl.blas.create_handle_for_hgemv(trans_a, m, n, type)</code>
函数功能	创建矩阵-向量乘的handle，输入数据和输出数据的数据类型为 aclFloat16 。 创建handle成功后，需调用 acl.op.execute_with_handle 接口执行算子。
输入说明	trans_a : int，矩阵A是否转置的标记。 m : int，矩阵A的行数，存储矩阵乘数据时，行优先。 n : int，矩阵A的列数。 type : int，计算精度。
返回值说明	handle : int，执行算子的handle数据的指针地址。 ret : int，错误码。 - 返回0表示成功。 - 返回其它值表示失败。
约束说明	无

2.12.7 函数：s8gemv

C函数原型	aclError aclblasS8gemv(aclTransType transA, int m, int n, const int32_t *alpha, const int8_t *a, int lda, const int8_t *x, int incx, const int32_t *beta, int32_t *y, int incy, aclComputeType type, aclrtStream stream)
Python函数	ret = acl.blas.s8gemv(trans_a, m, n, alpha, a, lda, x, incx, beta, y, incy, type, stream)
函数功能	执行矩阵-向量的乘法， $y = \alpha Ax + \beta y$ ，输入数据的数据类型为int8_t，输出数据的数据类型为int32_t，异步接口。
输入说明	trans_a : int，矩阵A是否转置的标记。 m : int，矩阵A的行数，存储矩阵乘数据时，行优先。 n : int，矩阵A的列数。 alpha : int，用于执行乘操作的标量alpha数据指针地址。 a : int，矩阵A的数据指针地址。 lda : int，A矩阵的主位，此时选择转置，按行优先，则lda为A的列数。预留参数，当前只能设置为-1。 x : int，向量x的数据指针地址。 incx : int，x连续元素之间的步长。预留参数，当前只能设置为-1。 beta : int，用于执行乘操作的标量beta的数据指针地址。 y : int，向量y的数据指针地址。 incy : int，y连续元素之间的步长。预留参数，当前只能设置为-1。 type : int，计算精度。 stream : int，执行算子所在的Stream。
返回值说明	ret : int，错误码。 • 返回0表示成功。 • 返回其它值表示失败。
约束说明	无

参考资源	接口调用流程，参见接口调用流程。

2.12.8 函数：create_handle_for_s8gemv

C函数原型	<code>aclError aclblasCreateHandleForS8gemv(aclTransType transA, int m, int n, aclComputeType type, aclopHandle **handle)</code>
Python函数	<code>handle, ret = acl.blas.create_handle_for_s8gemv(trans_a, m, n, type)</code>
函数功能	创建矩阵-向量乘的handle，输入数据的数据类型为int8_t，输出数据的数据类型为int32_t。 创建handle成功后，需调用 acl.op.execute_with_handle 接口执行算子。
输入说明	trans_a : int，矩阵A是否转置的标记。 m : int，矩阵A的行数，存储矩阵乘数据时，行优先。 n : int，矩阵A的列数。 type : int，计算精度。
返回值说明	handle : int，执行算子的handle数据的指针地址。 ret : int，错误码。 - 返回0表示成功。 - 返回其它值表示失败。
约束说明	无

2.12.9 函数：gemm_ex

C函数原型	<code>aclError aclblasGemmEx(aclTransType transA, aclTransType transB, aclTransType transC, int m, int n, int k, const void *alpha, const void *matrixA, int lda, aclDataType dataTypeA, const void *matrixB, int ldb, aclDataType dataTypeB, const void *beta, void *matrixC, int ldc, aclDataType dataTypeC, aclComputeType type, aclrtStream stream)</code>
Python函数	<code>ret = acl.blas.gemm_ex(trans_a, trans_b, trans_c, m, n, k, alpha, matrix_a, lda, data_type_a, matrix_b, ldb, data_type_b, beta, matrix_c, ldc, data_type_c, type, stream)</code>
函数功能	执行矩阵-矩阵的乘法， $C = \alpha AB + \beta C$ ，输入数据、输出数据的数据类型通过入参设置，异步接口。

输入说明	trans_a: int, 矩阵A是否转置的标记。 trans_b: int, B矩阵是否转置的标记。 trans_c: int, C矩阵的标记, 当前仅支持aclTrans_N。 m: int, 矩阵A的行数与矩阵C的行数。 n: int, 矩阵B的列数与矩阵C的列数。 k: int, 矩阵A的列数与矩阵B的行数。 alpha: int, 用于执行乘操作的标量alpha数据指针地址。 matrix_a: int, 矩阵A的数据指针地址。 lda: int, A矩阵的主维, 此时选择转置, 按行优先, 则lda为A的列数。预留参数, 当前只能设置为-1。 data_type_a: int, 矩阵A的数据类型。 matrix_b: int, 矩阵B的数据指针地址。 ldb: int, B矩阵的主维, 此时选择转置, 按行优先, 则ldb为B的列数。预留参数, 当前只能设置为-1。 data_type_b: int, 矩阵B的数据类型。 beta: int, 用于执行乘操作的标量beta的数据指针地址。 matrix_c: int, 矩阵C的数据指针地址。 ldc: int, C矩阵的主维。预留参数, 当前只能设置为-1。 data_type_c: int, 矩阵C的数据类型。 type: int, 计算精度。 stream: int, 执行算子所在的Stream。
返回值说明	ret: int, 错误码。 • 返回0表示成功。 • 返回其它值表示失败。
约束说明	A、B、C的数据类型支持仅支持以下组合, α和β的数据类型与C一致。 A的数据类型 x的数据类型 y的数据类型 aclFloat16 aclFloat16 aclFloat16 aclFloat16 aclFloat16 float(float32) int8_t int8_t float(float32) int8_t int8_t int32_t
参考资源	接口调用流程、示例, 参见接口调用流程。

2.12.10 函数：create_handle_for_gemm_ex

C函数原型	aclError aclblasCreateHandleForGemmEx(aclTransType transA, aclTransType transB, aclTransType transC, int m, int n, int k, aclDataType dataTypeA, aclDataType dataTypeB, aclDataType dataTypeC, aclComputeType type, aclopHandle **handle)
Python函数	handle, ret = acl.blas.create_handle_for_gemm_ex(trans_a, trans_b, trans_c, m, n, k, data_type_a, data_type_b, data_type_c, type)
函数功能	创建矩阵-矩阵乘的handle，输入数据、输出数据的数据类型通过入参设置。 创建handle成功后，需调用 acl.op.execute_with_handle 接口执行算子。
输入说明	trans_a : int，矩阵A是否转置的标记。 trans_b : int，B矩阵是否转置的标记。 trans_c : int，C矩阵的标记，当前仅支持aclTrans_N。 m : int，矩阵A的行数与矩阵C的行数。 n : int，矩阵B的列数与矩阵C的列数。 k : int，矩阵A的列数与矩阵B的行数。 data_type_a : int，矩阵A的数据类型。 data_type_b : int，矩阵B的数据类型。 data_type_c : int，矩阵C的数据类型。 type : int，计算精度。
返回值说明	handle : int，执行算子的handle数据的指针地址。 ret : int，错误码。 - 返回0表示成功。 - 返回其它值表示失败。

约束说明	A、B、C的数据类型支持仅支持以下组合： A的数据类型 x的数据类型 y的数据类型 aclFloat16 aclFloat16 aclFloat16 aclFloat16 aclFloat16 float(float32) int8_t int8_t float(float32) int8_t int8_t int32_t

2.12.11 函数：hgemm

C函数原型	aclError aclblasHgemm(aclTransType transA, aclTransType transB, aclTransType transC, int m, int n, int k, const aclFloat16 *alpha, const aclFloat16 *matrixA, int lda, const aclFloat16 *matrixB, int ldb, const aclFloat16 *beta, aclFloat16 *matrixC, int ldc, aclComputeType type, aclrtStream stream)
Python函数	ret = acl.blas.hgemm(trans_a, trans_b, trans_c, m, n, k, alpha, matrix_a, lda, matrix_b, ldb, beta, matrix_c, ldc, type, stream)
函数功能	执行矩阵-矩阵的乘法， $C = \alpha AB + \beta C$ ，输入数据和输出数据的数据类型为 aclFloat16 ，异步接口。

输入说明	trans_a: int, 矩阵A是否转置的标记。 trans_b: int, B矩阵是否转置的标记。 trans_c: int, C矩阵的标记, 当前仅支持aclTrans_N。 m: int, 矩阵A的行数与矩阵C的行数。 n: int, 矩阵B的列数与矩阵C的列数。 k: int, 矩阵A的列数与矩阵B的行数。 alpha: int, 用于执行乘操作的标量alpha数据指针地址。 matrix_a: int, 矩阵A的数据指针地址。 lda: int, A矩阵的主维, 此时选择转置, 按行优先, 则lda为A的列数。 matrix_b: int, 矩阵B的数据指针地址。 ldb: int, B矩阵的主维, 此时选择转置, 按行优先, 则leading dimension为B的列数。 beta: int, 用于执行乘操作的标量beta的数据指针地址。 matrix_c: int, 矩阵C的数据指针地址。 ldc: int, C矩阵的主维, 当前预留。 type: int, 计算精度。 stream: int, 执行算子所在的Stream。
返回值说明	ret: int, 错误码。 • 返回0表示成功。 • 返回其它值表示失败。
约束说明	无
参考资源	接口调用流程, 参见接口调用流程。

2.12.12 函数: create_handle_for_hgemm

C函数原型	<pre>aclError aclblasCreateHandleForHgemm(aclTransType transA, aclTransType transB, aclTransType transC, int m, int n, int k, aclComputeType type, aclopHandle **handle)</pre>
Python函数	<pre>handle, ret = acl.blas.create_handle_for_hgemm(trans_a, trans_b, trans_c, m, n, k, type)</pre>

函数功能	创建矩阵-矩阵乘的handle，输入数据和输出数据的数据类型为aclFloat16。 创建handle成功后，需调用 acl.op.execute_with_handle 接口执行算子。
输入说明	trans_a : int，矩阵A是否转置的标记。 trans_b : int，B矩阵是否转置的标记。 trans_c : int，C矩阵的标记，当前仅支持aclTrans_N。 m : int，矩阵A的行数与矩阵C的行数。 n : int，矩阵B的列数与矩阵C的列数。 k : int，矩阵A的列数与矩阵B的行数。 type : int，计算精度。
返回值说明	handle : int，执行算子的handle数据的指针地址。 ret : int，错误码。 - 返回0表示成功。 - 返回其它值表示失败。
约束说明	无

2.12.13 函数：s8gemm

C函数原型	aclError aclblasS8gemm(aclTransType transA, aclTransType transB, aclTransType transC, int m, int n, int k, const int32_t *alpha, const int8_t *matrixA, int lda, const int8_t *matrixB, int ldb, const int32_t *beta, int32_t *matrixC, int ldc, aclComputeType type, aclrtStream stream)

Python函数	ret = acl.blas.s8gemm(trans_a, trans_b, trans_c, m, n, k, alpha, matrix_a, lda, matrix_b, ldb, beta, matrix_c, ldc, type, stream)
函数功能	执行矩阵-矩阵的乘法， $C = \alpha AB + \beta C$ ，输入数据的数据类型为int8_t，输出数据的数据类型为int32_t，异步接口。
输入说明	trans_a : int，矩阵A是否转置的标记。 trans_b : int，B矩阵是否转置的标记。 trans_c : int，C矩阵的标记，当前仅支持aclTrans_N。 m : int，矩阵A的行数与矩阵C的行数。 n : int，矩阵B的列数与矩阵C的列数。 k : int，矩阵A的列数与矩阵B的行数。 alpha : int，用于执行乘操作的标量alpha数据指针地址。 matrix_a : int，矩阵A的数据指针地址。 lda : int，A矩阵的主维，此时选择转置，按行优先，则lda为A的列数。 matrix_b : int，矩阵B的数据指针地址。 ldb : int，B矩阵的主维，此时选择转置，按行优先，则leading dimension为B的列数。 beta : int，用于执行乘操作的标量beta的数据指针地址。 matrix_c : int，矩阵C的数据指针地址。 ldc : int，C矩阵的主维，当前预留。 type : int，计算精度。 stream : int，执行算子所在的Stream。
返回值说明	ret : int，错误码。 - 返回0表示成功。 - 返回其它值表示失败。
约束说明	无
参考资源	接口调用流程，参见接口调用流程。

2.12.14 函数：create_handle_for_s8gemm

C函数原型	aclError aclblasCreateHandleForS8gemm(aclTransType transA, aclTransType transB, aclTransType transC, int m, int n, int k, aclComputeType type, aclopHandle **handle)
Python函数	handle, ret = acl.blas.create_handle_for_s8gemm(trans_a, trans_b, trans_c, m, n, k, type)
函数功能	创建矩阵-矩阵乘的handle，输入数据的数据类型为int8_t，输出数据的数据类型为int32_t。 创建handle成功后，需调用 acl.op.execute_with_handle 接口执行算子。
输入说明	trans_a : int，矩阵A是否转置的标记。 trans_b : int，B矩阵是否转置的标记。 trans_c : int，C矩阵的标记，当前仅支持aclTrans_N。 m : int，矩阵A的行数与矩阵C的行数。 n : int，矩阵B的列数与矩阵C的列数。 k : int，矩阵A的列数与矩阵B的行数。 type : int，计算精度。
返回值说明	handle : int，执行算子的handle数据的指针地址。 ret : int，错误码。 - 返回0表示成功。 - 返回其它值表示失败。
约束说明	无

2.13 算子编译

2.13.1 函数：register_compile_func

C函数原型	aclError aclopRegisterCompileFunc(const char *opType, aclopCompileFunc func)
Python函数	ret = acl.op.register_compile_func(op_type, func)

函数功能	动态Shape场景下，注册算子选择器，用于在算子执行时，能针对不同shape，选择相应的Tiling策略。 如果某算子已注册算子选择器，则不允许重新注册，如果需要变更算子选择器，必须先调用 acl.op.unregister_compile_func 接口取消注册，然后再调用 acl.op.register_compile_func 接口重新注册。
输入说明	op_type : str，算子类型。 func : python函数对象，算子选择器回调函数，函数定义： Python侧格式如下： <pre>def call_back_func(num_inputs, input_desc, num_outputs, output_desc, op_attr, acl_op_kernel_desc): pass</pre>
返回值说明	ret : int，错误码。 - 返回0表示成功。 - 返回其它值表示失败。
约束说明	无
参考资源	接口调用流程、示例，参见单算子调用。

2.13.2 函数：unregister_compile_func

C函数原型	<code>aclError aclOpUnregisterCompileFunc(const char *opType)</code>
Python函数	<code>ret = acl.op.unregister_compile_func(op_type)</code>
函数功能	动态Shape场景下，取消注册算子选择器。
输入说明	op_type : str，算子类型。
返回值说明	ret : int，错误码。 - 返回0表示成功。 - 返回其它值表示失败。
约束说明	无

2.13.3 函数：create_kernel

C函数原型	<code>aclError aclOpCreateKernel(const char *opType, const char *kernelId, const char *kernelName, void *binData, int binSize, aclOpEngineType enginetype, aclDataDeallocator deallocator)</code>

Python函数	ret = acl.op.create_kernel(op_type, kernel_id, kernel_name, bin_data, bin_size, enginetype, deallocator)
函数功能	动态Shape场景下，将算子注册到系统内部，运行算子时使用。
输入说明	op_type : str，算子类型。 kernel_id : str，算子执行时要指定的Kernel ID。 kernel_name : str，算子Kernel名称，和算子二进制文件中的kernelName保持一致。 bin_data : int，算子Kernel文件的内存地址。 bin_size : int，算子Kernel文件的内存大小，单位为Byte。 enginetype : int，表示算子执行引擎，该参数只有acl.op.update.params接口的“compile_flag”参数值为“ACL_COMPILE_SYS”时有效。 • 0: ACL_ENGINE_SYS，不关心具体执行引擎时填写。 • 1: ACL_ENGINE_AICORE，将算子编译成AI Core算子。 • 2: ACL_ENGINE_VECTOR，将算子编译成Vector Core算子。 deallocator : int，指定是否自动释放bin_data内存。 • 0: 不自动释放，数据由调用者自行编写代码进行释放。 • 1: 自动释放，内部会设置回调函数释放bin_data内存。
返回值说明	ret : int，错误码。 • 返回0表示成功。 • 返回其它值表示失败。
约束说明	无
参考资源	接口调用流程及示例，参见动态Shape算子（注册算子选择器）。

2.13.4 函数：set_kernel_args

C函数原型	aclError aclopSetKernelArgs(aclopKernelDesc *kernelDesc, const char *kernelId, uint32_t blockDim, const void *args, uint32_t argSize)
Python函数	ret = acl.op.set_kernel_args(kernel_desc, kernel_id, block_dim, args, arg_size)
函数功能	动态Shape场景下，设置算子Tiling参数、执行并发数。

输入说明	kernel_desc: int, Kernel描述缓存, aclopKernelDesc类型的指针地址。 kernel_id: str, 算子执行时要指定的Kernel ID, 与调用acl.op.create.kernel时传递的kernel_id一致。 block_dim: int, Kernel执行的并发数。 bin_data: int, 算子Kernel文件的内存地址。 bin_size: int, 算子Kernel文件的内存大小, 单位为Byte。 args: int, Tiling参数, 需要通过numpy数组tobytes方法获取bytes对象, 然后使用acl.utils.bytes_to_ptr()接口获取Tiling参数指针地址。 arg_size: int, Tiling参数内存大小, 单位为Byte, 为numpy.tobytes()转换得到bytes的对象长度, 可用len函数获取。
返回值说明	ret: int, 错误码。 • 返回0表示成功。 • 返回其它值表示失败。
约束说明	无
参考资源	接口调用流程及示例, 参见动态Shape算子 (注册算子选择器)。

2.13.5 函数: set_kernel_workspace_sizes

C函数原型	aclError aclopSetKernelWorkspaceSizes(aclopKernelDesc *kernelDesc, int numWorkspaces, size_t *workspaceSizes)
Python函数	ret = acl.op.set_kernel_workspace_sizes(kernel_desc, num_workspaces, workspace_sizes)
函数功能	动态Shape场景下, 设置算子Workspace参数。
输入说明	kernel_desc: int, Kernel描述缓存, aclopKernelDesc类型的指针地址。 num_workspaces: int, Workspaces个数。 workspace_sizes: int, Workspaces大小的数组地址。
返回值说明	ret: int, 错误码。 • 返回0表示成功。 • 返回其它值表示失败。
约束说明	为非必须接口, 根据算子情况可选。

2.13.6 函数：update_params

C函数原型	<code>aclError aclOpUpdateParams(const char *opType, int numInputs, const aclTensorDesc *const inputDesc[], int numOutputs, const aclTensorDesc *const outputDesc[], const aclOpAttr *attr)</code>
Python函数	<code>ret = acl.op.update_params(op_type, input_desc, output_desc, attr)</code>
函数功能	动态Shape场景下，设置算子Tiling参数、执行并发数。
输入说明	op_type : str，算子类型名称。 input_desc : list，算子输入Tensor的描述。 output_desc : list，算子输出Tensor的描述。 attr : int，算子属性。
返回值说明	ret : int，错误码。 - 返回0表示成功。 - 返回其它值表示失败。
约束说明	无
参考资源	接口调用流程及示例，参见动态Shape算子（注册算子选择器）。

2.14 特征向量检索

2.14.1 总体说明

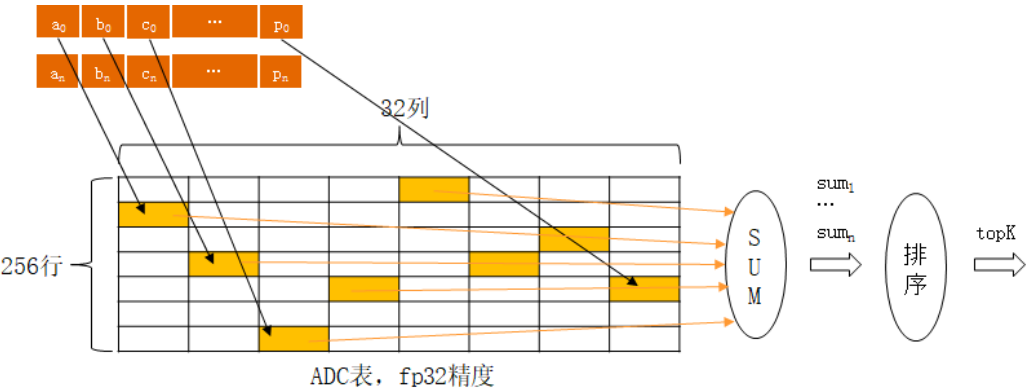
在图像检索场景，通常使用深度学习算法处理目标等感兴趣的图像，提取出高维特征向量，并将一组感兴趣的图像提取得到的特征向量作为底库，待搜索的特征向量与底库中的向量依次进行相似度计算，按相似度排序返回检索结果。为了进一步降低存储和计算的开销，对浮点型长特征向量进行量化处理、转换为短特征再进行检索比对的技术被广泛应用。

昇腾AI处理器支持基于硬件加速的PQ短特征检索，提供特征底库的添加/删除接口，特征向量的修改/删除接口、以及执行检索接口，支持长度为32 Byte的短特征向量，支持设置返回TOP结果的数量，支持1:N和N:M检索模式。用户需要自行准备短特征底库，以及生成待检索向量的ADC表，流程如下：

图 2-25 流程示意



检索比对过程示意图如下所示：



ADC表是大小为32KByte的查找表，使用时需要将表的内容以无符号字符的格式传入对应接口。

Atlas 推理系列产品的特征检索规格如下表所示：

表 2-70 规格说明

类型	规格说明
特征长度	32字节（因为数据结构管理开销，每条特征的平均内存开销约为36字节）。
底库容量	最大5亿条（实际值取决于板卡的内存容量）。
检索比对类型	支持1：N检索，支持N：M检索。
底库数量	1：N底库最大100万个。 N：M底库最大1个。
底库容量	1：N每个特征库最大100万条特征。 N：M底库最大1000万条特征。
检索比对性能	每秒50亿次。 5亿底库下可达到10QPS，时延<1秒
检索返回候选特征数量	2~4800

2.14.2 函数：init

C函数原型	aclError aclfvInit(aclfvInitPara *initPara)
Python函数	ret = acl.fv.init(init_para)

函数功能	初始化特征检索模块。与 acl.fv.release 接口配套使用。 Atlas 200/300/500 推理产品不支持该接口。 Atlas 200I/500 A2推理产品不支持该接口。 Atlas 训练系列产品不支持该接口。 Atlas A2训练系列产品/Atlas 800I A2推理产品不支持该接口。 Atlas 推理系列加速模块产品不支持该接口。
输入说明	init_para : int, 指定特征向量检索的初始化参数。 - 调用acl.fv.create_init_para接口创建aclfvInitPara类型的数据。 1 : N场景下调用acl.fv.set_1n_top_num接口设置返回的topK结果数量的最大值, 不设置时, 默认返回的topK结果数量的最大值为4800。 - N : M场景下调用acl.fv.set_nm_top_num接口设置返回的topK结果数量的最大值, 不设置时, 默认返回的topK结果数量的最大值为500。
返回值说明	ret : int, 错误码。 - 返回0表示成功。 - 返回其它值表示失败。
约束说明	一个进程内只能调用一次 acl.fv.init 接口。
参考资源	接口调用流程及示例代码, 参见特征向量检索。

2.14.3 函数: release

C函数原型	aclError aclfvRelease()
Python函数	ret = acl.fv.release()
函数功能	特征检索模块去初始化, 释放内存空间, 与 acl.fv.init 接口配套使用。 Atlas 200/300/500 推理产品不支持该接口。 Atlas 200I/500 A2推理产品不支持该接口。 Atlas 训练系列产品不支持该接口。 Atlas A2训练系列产品/Atlas 800I A2推理产品不支持该接口。 Atlas 推理系列加速模块产品不支持该接口。
输入说明	无
返回值说明	ret : int, 错误码。 - 返回0表示成功。 - 返回其它值表示失败。
约束说明	无

参考资源	接口调用流程及示例代码，参见特征向量检索。

2.14.4 函数：repo_add

C函数原型	aclError aclfvRepoAdd(aclfvSearchType type, aclfvFeatureInfo *featureInfo)
Python函数	ret = acl.fv.repo_add(type, feature_info)
函数功能	添加底库或向已存在底库中添加特征。该章节描述的1:N表示“检索请求的个数:底库的数量”，N:M用于碰撞两个库的相似性。 Atlas 200/300/500 推理产品不支持该接口。 Atlas 200I/500 A2推理产品不支持该接口。 Atlas 训练系列产品不支持该接口。 Atlas A2训练系列产品/Atlas 800I A2推理产品不支持该接口。 Atlas 推理系列加速模块产品不支持该接口。
输入说明	• type: int，检索类型，请参见2.16.14 aclfvSearchType处说明。 • feature_info: int，指针地址，指定特征描述信息。需提前调用acl.fv.create_feature_info接口创建aclfvFeatureInfo类型的数据。
返回值说明	ret: int，错误码。 • 返回0表示成功。 • 返回其它值表示失败。
约束说明	• 1:N添加时，对于每个库，用户需要保证aclfvFeatureInfo字典中的offset连续，N:M添加时，offset不为0会报错。 • 非线程安全，N:M场景不允许多线程同时添加\检索\删除。 • 调用本接口，内部会默认创建一个Stream，用于接口内任务同步执行，在接口执行完成后，该Stream资源会被自动释放。受硬件限制，Stream总数有限，请参见2.7.3.1 函数：create_stream处的说明。
参考资源	接口调用流程及示例代码，参见特征向量检索。

2.14.5 函数：repo_del

C函数原型	aclError aclfvRepoDel(aclfvSearchType type, aclfvRepoRange *repoRange)
Python函数	ret = acl.fv.repo_del(type, repo_range)

函数功能	删除底库。 Atlas 200/300/500 推理产品不支持该接口。 Atlas 200I/500 A2推理产品不支持该接口。 Atlas 训练系列产品不支持该接口。 Atlas A2训练系列产品/Atlas 800I A2推理产品不支持该接口。 Atlas 推理系列加速模块产品不支持该接口。
输入说明	• type: int, 检索类型, 请参见2.16.14 aclfvSearchType处说明。 • repo_range: int, 指针地址, 指定特征删除范围。需提前调用acl.fv.create_repo_range接口创建aclfvRepoRange类型的数据。
返回值说明	ret : int, 错误码。 • 返回0表示成功。 • 返回其它值表示失败。
约束说明	• 非线程安全, N:M场景不允许多线程同时添加\检索\删除。 • 调用本接口, 内部会默认创建一个Stream, 用于接口内任务同步执行, 在接口执行完成后, 该Stream资源会被自动释放。受硬件限制, Stream总数有限, 请参见2.7.3.1 函数: create_stream处的说明。
参考资源	接口调用流程及示例代码, 参见特征向量检索。

2.14.6 函数: delete

C函数原型	aclError aclfvDel(aclfvFeatureInfo *featureInfo)
Python函数	ret = acl.fv.delete(feature_info)
函数功能	精确删除底库中某个特征, 一次只能删除底库中一个特征。N:M场景下不涉及这个接口。 Atlas 200/300/500 推理产品不支持该接口。 Atlas 200I/500 A2推理产品不支持该接口。 Atlas 训练系列产品不支持该接口。 Atlas A2训练系列产品/Atlas 800I A2推理产品不支持该接口。 Atlas 推理系列加速模块产品不支持该接口。
输入说明	feature_info : int, 指针地址, 指定特征描述信息。 需提前调用 acl.fv.create_feature_info 接口创建aclfvFeatureInfo类型的数据。
返回值说明	ret : int, 错误码。 • 返回0表示成功。 • 返回其它值表示失败。

约束说明	调用本接口，内部会默认创建一个Stream，用于接口内任务同步执行，在接口执行完成后，该Stream资源会被自动释放。受硬件限制，Stream总数有限，请参见2.7.3.1 函数：create_stream处的说明。
参考资源	接口调用流程及示例代码，参见特征向量检索。

2.14.7 函数：modify

C函数原型	aclError aclfvModify(aclfvFeatureInfo *featureInfo)
Python函数	ret = acl.fv.modify(feature_info)
函数功能	修改底库中某个特征，一次只能修改底库中一个特征，不允许修改底库中不存在的特征。N:M场景下不涉及这个接口。 Atlas 200/300/500 推理产品不支持该接口。 Atlas 200I/500 A2推理产品不支持该接口。 Atlas 训练系列产品不支持该接口。 Atlas A2训练系列产品/Atlas 800I A2推理产品不支持该接口。 Atlas 推理系列加速模块产品不支持该接口。
输入说明	feature_info : int，指针地址，指定特征描述信息。需提前调用 acl.fv.create_feature_info 接口创建aclfvFeatureInfo类型的数据。
返回值说明	ret : int，错误码。 - 返回0表示成功。 - 返回其它值表示失败。
约束说明	调用本接口，内部会默认创建一个Stream，用于接口内任务同步执行，在接口执行完成后，该Stream资源会被自动释放。受硬件限制，Stream总数有限，请参见2.7.3.1 函数：create_stream处的说明。
参考资源	接口调用流程及示例代码，参见特征向量检索。

2.14.8 函数：search

C函数原型	aclError aclfvSearch(aclfvSearchType type, aclfvSearchInput *searchInput, aclfvSearchResult *searchRst)
Python函数	ret = acl.fv.search(type, search_input, search_result)

函数功能	特征1:N或N:M检索。 Atlas 200/300/500 推理产品不支持该接口。 Atlas 200I/500 A2推理产品不支持该接口。 Atlas 训练系列产品不支持该接口。 Atlas A2训练系列产品/Atlas 800I A2推理产品不支持该接口。 Atlas 推理系列加速模块产品不支持该接口。
输入说明	• type: int, 指定特征描述信息, 请参见2.16.14 aclfvSearchType处说明。 • search_input: int, 指针地址, 检索输入信息。需提前调用acl.fv.create_search_input接口创建aclfvSearchInput类型的数据。 • search_result: int, 指针地址, 检索输出结果。需提前调用acl.fv.create_search_result接口创建aclfvSearchResult类型的数据, 该数据中queryCnt参数应与检索输入信息searchInput中的queryCnt参数相同。 检索输出返回的结果数量通过acl.fv.create_search_input接口的topk参数设置, 但与调用acl.fv.init接口进行初始化时设置的topK结果数量最大值有关, 若调用acl.fv.create_search_input接口设置的topk参数值大于或等于初始化时设置的topK结果数量最大值, 则在返回检索输出结果时, 按初始化时设置的topK结果数量最大值返回检索结果, 若调用acl.fv.create_search_input接口设置的topk参数值小于初始化时设置的topK结果数量最大值, 则在返回检索输出结果时, 按实际通过acl.fv.create_search_input接口设置的topk参数值返回检索结果。
返回值说明	ret: int, 错误码。 • 返回0表示成功。 • 返回其它值表示失败。
约束说明	• 非线程安全, N:M场景不允许多线程同时添加\检索\删除。 • N:M场景或1:N场景, 不添加底库, 调用acl.fv.search接口检索时, 返回结果数量为0。 • 调用本接口, 内部会默认创建一个Stream, 用于接口内任务同步执行, 在接口执行完成后, 该Stream资源会被自动释放。受硬件限制, Stream总数有限, 请参见2.7.3.1 函数: create_stream处的说明。
参考资源	接口调用流程及示例代码, 参见特征向量检索。

2.14.9 函数: create_init_para

C函数原型	<code>aclfvInitPara *aclfvCreateInitPara(uint64_t fsNum)</code>
Python函数	<code>init_para = acl.fv.create_init_para(fs_num)</code>

函数功能	创建aclfvInitPara类型的数据，表示创建特征向量检索的初始化参数。 如需销毁aclfvInitPara类型的数据，请参见 2.14.10 函数：destroy_init_para 。 Atlas 200/300/500 推理产品 不支持该接口。 Atlas 200I/500 A2推理产品 不支持该接口。 Atlas 训练系列产品 不支持该接口。 Atlas A2训练系列产品/Atlas 800I A2推理产品 不支持该接口。
输入说明	fs_num : int，底库特征个数，用于系统内部给特征检索模块申请内存资源，取值范围：[1,500000000]。
返回值说明	init_para :int。 - 返回aclfvInitPara类型，表示成功。 - 返回0，表示失败。
约束说明	无
参考资源	接口调用流程及示例代码，参见特征向量检索。

2.14.10 函数：destroy_init_para

C函数原型	aclError aclfvDestroyInitPara(aclfvInitPara *initPara)
Python函数	ret = acl.fv.destroy_init_para(init_para)
函数功能	销毁通过 acl.fv.create_init_para 接口创建的aclfvInitPara类型的数据。 Atlas 200/300/500 推理产品 不支持该接口。 Atlas 200I/500 A2推理产品 不支持该接口。 Atlas 训练系列产品 不支持该接口。 Atlas A2训练系列产品/Atlas 800I A2推理产品 不支持该接口。
输入说明	init_para : int，待销毁的aclfvInitPara类型。
返回值说明	ret : int，错误码。 - 返回0表示成功。 - 返回其它值表示失败。
约束说明	无
参考资源	接口调用流程及示例代码，参见特征向量检索。

2.14.11 函数：set_1n_top_num

C函数原型	<code>aclError aclfvSet1NTopNum(aclfvInitPara *initPara, uint32_t maxTopNumFor1N)</code>
Python函数	<code>ret = acl.fv.set_1n_top_num(init_para, max_top_num_for_1n)</code>
函数功能	设置1:N场景下返回的topK结果数量的最大值，如果不设置，默认topK结果数量的最大值为4800。 Atlas 200/300/500 推理产品不支持该接口。 Atlas 200I/500 A2推理产品不支持该接口。 Atlas 训练系列产品不支持该接口。 Atlas A2训练系列产品/Atlas 800I A2推理产品不支持该接口。
输入说明	init_para: int, 指定特征向量检索的初始化参数。 需提前调用acl.fv.create_init_para接口创建aclfvInitPara类型的数据。 max_top_num_for_1n: int, 1:N场景下返回的topK结果的最大值，取值范围：[2, 4800]。
返回值说明	ret: int, 错误码。 - 返回0表示成功。 - 返回其它值表示失败。
约束说明	无
参考资源	接口调用流程及示例代码，参见特征向量检索。

2.14.12 函数：set_nm_top_num

C函数原型	<code>aclError aclfvSetNMTopNum(aclfvInitPara *initPara, uint32_t maxTopNumForNM)</code>
Python函数	<code>ret = acl.fv.set_nm_top_num(init_para, max_top_num_for_nm)</code>
函数功能	设置N:M场景下返回的topK结果数量的最大值，如果不设置，默认topK结果数量的最大值为500。 Atlas 200/300/500 推理产品不支持该接口。 Atlas 200I/500 A2推理产品不支持该接口。 Atlas 训练系列产品不支持该接口。 Atlas A2训练系列产品/Atlas 800I A2推理产品不支持该接口。
输入说明	init_para: int, 指定特征向量检索的初始化参数。 需提前调用acl.fv.create_init_para接口创建aclfvInitPara类型的数据。 max_top_num_for_nm: int, N:M场景下返回的topK结果的最大值，取值范围：[500,4800]。

返回值说明	ret : int, 错误码。 - 返回0表示成功。 - 返回其它值表示失败。
约束说明	无
参考资源	接口调用流程及示例代码，参见特征向量检索。

2.14.13 函数：create_feature_info

C函数原型	aclfvFeatureInfo *aclfvCreateFeatureInfo (uint32_t id0, uint32_t id1, uint32_t offset, uint32_t featureLen, uint32_t featureCount, uint8_t *featureData, uint32_t featureDataLen)
Python函数	feature_info = acl.fv.create_feature_info (id0, id1, offset, feature_len, feature_count, feature_data, feature_len)
函数功能	创建aclfvFeatureInfo类型的数据，表示创建特征描述信息。 如需销毁aclfvFeatureInfo类型的数据，请参见2.14.14 函数：destroy_feature_info。 Atlas 200/300/500 推理产品不支持该接口。 Atlas 200I/500 A2推理产品不支持该接口。 Atlas 训练系列产品不支持该接口。 Atlas A2训练系列产品/Atlas 800I A2推理产品不支持该接口。
输入说明	id0: int, 一级库id, 取值范围0-1023, N:M场景默认为0。通过id0、id1共同确定一个库。 id1: int, 二级库id, 取值范围0-1023, N:M场景默认为0。通过id0、id1共同确定一个库。 offset: int, 添加的首个特征在库中的偏移，偏移值需要与库中已添加特征个数一致。N:M场景默认为0。 feature_len: int, 单个特征长度，当前固定为36Byte，系统内部会校验。 feature_count: int, 特征数量，1:N最大100万，N:M最大1000万。 feature_data: int, 特征值，根据特征长度连续存储，每条特征前4Byte值为0。 feature_len: int, featureData申请的内存长度，用于校验。
返回值说明	feature_info : int, 指针地址。 - 返回aclfvFeatureInfo类型，表示成功。 - 返回0，表示失败。
约束说明	无

参考资源	接口调用流程及示例代码，参见特征向量检索。

2.14.14 函数：destroy_feature_info

C函数原型	<code>aclError aclfvDestroyFeatureInfo(aclfvFeatureInfo *featureInfo)</code>
Python函数	<code>ret = acl.fv.destroy_feature_info(feature_info)</code>
函数功能	销毁通过acl.fv.create_feature_info接口创建的<acode>aclfvFeatureInfo类型的数据。</acode> Atlas 200/300/500 推理产品不支持该接口。 Atlas 200I/500 A2推理产品不支持该接口。 Atlas 训练系列产品不支持该接口。 Atlas A2训练系列产品/Atlas 800I A2推理产品不支持该接口。
输入说明	<code>feature_info</code> : int, 指针地址，待销毁的 <acode>aclfvFeatureInfo类型。</acode>
返回值说明	<code>ret</code>: int, 错误码。 • 返回0表示成功。 • 返回其它值表示失败。
约束说明	无
参考资源	接口调用流程及示例代码，参见特征向量检索。

2.14.15 函数：create_repo_range

C函数原型	<code>aclfvRepoRange *aclfvCreateRepoRange(uint32_t id0Min, uint32_t id0Max, uint32_t id1Min, uint32_t id1Max)</code>
Python函数	<code>repo_range = acl.fv.create_repo_range(id0min, id0max, id1min, id1max)</code>
函数功能	创建<acode>aclfvRepoRange类型的数据，表示创建特征库范围的参数。</acode> 如需销毁<acode>aclfvRepoRange类型的数据，请参见2.14.16 函数：destroy_repo_range。</acode> Atlas 200/300/500 推理产品不支持该接口。 Atlas 200I/500 A2推理产品不支持该接口。 Atlas 训练系列产品不支持该接口。 Atlas A2训练系列产品/Atlas 800I A2推理产品不支持该接口。

输入说明	• id0min: int, id0起始值, 取值范围: [0,1023], 需小于等于id0Max。 • id0max: int, id0最大值, 取值范围: [0,1023]。 • id1min: int, id1起始值, 取值范围: [0,1023], 需小于等于id1Max • id1max: int, id1最大值, 取值范围: [0,1023]。
返回值说明	repo_range : int, 指针地址。 • 返回aclfvRepoRange类型, 表示成功。 • 返回0, 表示失败。
约束说明	无
参考资源	接口调用流程及示例代码, 参见特征向量检索。

2.14.16 函数: destroy_repo_range

C函数原型	aclError aclfvDestroyRepoRange(aclfvRepoRange *repoRange)
Python函数	ret = acl.fv.destroy_repo_range(repo_range)
函数功能	销毁通过 acl.fv.create_repo_range 接口创建的aclfvRepoRange类型的数据。 Atlas 200/300/500 推理产品 不支持该接口。 Atlas 200I/500 A2推理产品 不支持该接口。 Atlas 训练系列产品 不支持该接口。 Atlas A2训练系列产品/Atlas 800I A2推理产品 不支持该接口。
输入说明	repo_range : int, 指针地址, 待销毁的aclfvRepoRange类型。
返回值说明	ret : int, 错误码。 • 返回0表示成功。 • 返回其它值表示失败。
约束说明	无
参考资源	接口调用流程及示例代码, 参见特征向量检索。

2.14.17 函数: create_query_table

C函数原型	aclfvQueryTable *aclfvCreateQueryTable(uint32_t queryCnt, uint32_t tableLen, uint8_t *tableData, uint32_t tableDataLen)

Python函数	<code>query_table = acl.fv.create_query_table(query_cnt, table_len, table_data, table_data_len)</code>
函数功能	创建aclfvQueryTable类型的数据，表示创建检索输入表信息。 如需销毁aclfvQueryTable类型的数据，请参见2.14.18 函数：destroy_query_table。 Atlas 200/300/500 推理产品不支持该接口。 Atlas 200I/500 A2推理产品不支持该接口。 Atlas 训练系列产品不支持该接口。 Atlas A2训练系列产品/Atlas 800I A2推理产品不支持该接口。
输入说明	• query_cnt: int，检索请求数量，1:N场景为1，N:M场景最大1024。 • table_len: int，单个表长度，长度固定为32KB，系统内部会校验。 • table_data: int，特征值列表，根据特征长度连续存储。 • table_data_len: int，tableData申请的内存长度，用于校验。
返回值说明	query_table: int，指针地址。 • 返回aclfvQueryTable类型，表示成功。 • 返回0，表示失败。
约束说明	无
参考资源	接口调用流程及示例代码，参见 特征向量检索 。

2.14.18 函数：destroy_query_table

C函数原型	<code>aclError aclfvDestroyQueryTable(aclfvQueryTable *queryTable)</code>
Python函数	<code>ret = acl.fv.destroy_query_table(query_table)</code>
函数功能	销毁通过acl.fv.create_query_table接口创建的aclfvQueryTable类型的数据。 Atlas 200/300/500 推理产品不支持该接口。 Atlas 200I/500 A2推理产品不支持该接口。 Atlas 训练系列产品不支持该接口。 Atlas A2训练系列产品/Atlas 800I A2推理产品不支持该接口。
输入说明	query_table: int，指针地址，待销毁的aclfvQueryTable类型。
返回值说明	ret: int，错误码。 • 返回0表示成功。 • 返回其它值表示失败。
约束说明	无

参考资源	接口调用流程及示例代码，参见特征向量检索。

2.14.19 函数：create_search_input

C函数原型	<code>aclfvSearchInput *aclfvCreateSearchInput(aclfvQueryTable *queryTable, aclfvRepoRange *repoRange, uint32_t topk)</code>
Python函数	<code>search_input = acl.fv.create_search_input(query_table, repo_range, topk)</code>
函数功能	创建<acode>aclfvSearchInput类型的数据，表示创建检索任务输入信息。如需销毁<acode>aclfvSearchInput类型的数据，请参见2.14.20 函数：destroy_search_input。</acode></acode> Atlas 200/300/500 推理产品不支持该接口。 Atlas 200I/500 A2推理产品不支持该接口。 Atlas 训练系列产品不支持该接口。 Atlas A2训练系列产品/Atlas 800I A2推理产品不支持该接口。
输入说明	query_table: int，指针地址，检索的输入表信息。调用acl.fv.create_query_table接口创建<acode>aclfvQueryTable类型的数据。</acode> repo_range: int，指针地址，检索的特征库范围。调用acl.fv.create_repo_range接口创建<acode>aclfvRepoRange类型的数据。</acode> topk: 单个检索请求需要返回的结果数量。
返回值说明	search_input: int，指针地址。 - 返回<acode>aclfvSearchInput类型，表示成功。</acode> - 返回0，表示失败。
约束说明	无
参考资源	接口调用流程及示例代码，参见特征向量检索。

2.14.20 函数：destroy_search_input

C函数原型	<code>aclError aclfvDestroySearchInput(aclfvSearchInput *searchInput)</code>
Python函数	<code>ret = acl.fv.destroy_search_input(search_input)</code>

函数功能	销毁通过 acl.fv.create_search_input 接口创建的aclfvSearchInput类型的数据。 Atlas 200/300/500 推理产品 不支持该接口。 Atlas 200I/500 A2推理产品 不支持该接口。 Atlas 训练系列产品 不支持该接口。 Atlas A2训练系列产品/Atlas 800I A2推理产品 不支持该接口。
输入说明	search_input : int, 指针地址, 待销毁的aclfvSearchInput类型。
返回值说明	ret : int, 错误码。 - 返回0表示成功。 - 返回其它值表示失败。
约束说明	无
参考资源	接口调用流程及示例代码, 参见特征向量检索。

2.14.21 函数: create_search_result

C函数原型	aclfvSearchResult *aclfvCreateSearchResult(uint32_t queryCnt, uint32_t *resultNum, uint32_t resultNumDataLen, uint32_t *id0, uint32_t *id1, uint32_t *resultOffset, float *resultDistance, uint32_t dataLen)
Python函数	search_result = acl.fv.create_search_result(query_cnt, result_num, result_num_data_len, id_0, id_1, result_offset, result_distance, data_len)
函数功能	创建aclfvSearchResult类型的数据, 表示创建检索结果信息。 如需销毁aclfvSearchResult类型的数据, 请参见 2.14.22 函数: destroy_search_result 。 Atlas 200/300/500 推理产品 不支持该接口。 Atlas 200I/500 A2推理产品 不支持该接口。 Atlas 训练系列产品 不支持该接口。 Atlas A2训练系列产品/Atlas 800I A2推理产品 不支持该接口。

输入说明	• query_cnt: int, 检索请求个数, 1:N场景为1。 • result_num: int, 每个检索请求的结果个数。 • result_num_data_len: int, resultNum的内存总长度。 • id_0: int, 一级库id, 总个数为topK*queryCnt。 • id_1: int, 二级库id, 总个数为topK*queryCnt。 • result_offset: int, 每个检索请求的检索结果对应底库的偏移, 总个数为topK*queryCnt。 • result_distance: int, 每个检索结果与检索请求间的距离, 总个数为topK*queryCnt。 • data_len: int, 申请的内存大小, 计算公式为: topK*queryCnt*4Byte。
返回值说明	search_result : int, 指针地址。 • 返回aclfvSearchResult类型, 表示成功。 • 返回0, 表示失败。
约束说明	无
参考资源	接口调用流程及示例代码, 参见特征向量检索。

2.14.22 函数: `destroy_search_result`

C函数原型	<code>aclError aclfvDestroySearchResult(aclfvSearchResult *searchResult)</code>
Python函数	<code>ret = acl.fv.destroy_search_result(search_result)</code>
函数功能	销毁通过 acl.fv.create_search_result 接口创建的aclfvSearchResult类型的数据。 Atlas 200/300/500 推理产品 不支持该接口。 Atlas 200I/500 A2推理产品 不支持该接口。 Atlas 训练系列产品 不支持该接口。 Atlas A2训练系列产品/Atlas 800I A2推理产品 不支持该接口。
输入说明	search_result : int, 指针地址, 待销毁的aclfvSearchResult类型。
返回值说明	ret : int, 错误码。 • 返回0表示成功。 • 返回其它值表示失败。
约束说明	无
参考资源	接口调用流程及示例代码, 参见特征向量检索。

2.15 Profiling 数据采集

2.15.1 Profiling pyACL API (通过 Profiling pyACL API 采集并落盘性能数据)

2.15.1.1 功能介绍

功能

该章节描述通过Profiling pyACL API采集并落盘性能数据，实现将采集到的Profiling数据写入文件，再使用Profiling工具解析该文件（请参见《[性能调优工具](#)》下的“数据解析与导出”），并展示性能分析数据。

包括以下两种接口调用方式：

- [acl.prof.init](#)接口、[acl.prof.start](#)接口、[acl.prof.stop](#)接口、[acl.prof.finalize](#)接口配合使用，实现该方式的性能数据采集。该方式可获取pyACL的接口性能数据、AI Core上算子的执行时间、AI Core性能指标数据等。目前这些接口为进程级控制，表示在进程内任意线程调用该接口，其它线程都会生效。
一个进程内，可以根据需求多次调用这些接口，基于不同的Profiling采集配置，采集数据。
- 调用[acl.init](#)接口，在pyACL初始化阶段，通过*.json文件传入要采集的Profiling数据。该方式可获取pyACL的接口性能数据、AI Core上算子的执行时间、AI Core性能指标数据等。
一个进程内，只能调用一次[acl.init](#)接口，如果要修改Profiling采集配置，需修改*.json文件中的配置。详细使用说明请参见[acl.init](#)接口处的说明，不在本章节描述。

总体约束

不能与[2.15.3 Profiling pyACL API for Subscription \(订阅算子信息的Profiling pyACL API \)](#)的接口交叉调用：[acl.prof.init](#)接口和[acl.prof.finalize](#)接口之间不能调用[acl.prof.model_subscribe](#)接口、[aclprofGet*](#)接口、[acl.prof.model_unsubscribe](#)接口。

接口约束说明

- 调用接口要求：
 - [acl.prof.init](#)接口必须在[acl.init](#)接口之后、模型加载之前调用。
如果已经通过[acl.init](#)接口配置了Profiling信息，则调用[acl.prof.init](#)接口、[acl.prof.start](#)接口、[acl.prof.stop](#)接口、[acl.prof.finalize](#)时，会返回报错。
如果没有调用[acl.prof.init](#)接口，调用[acl.prof.start](#)接口、[acl.prof.stop](#)接口、[acl.prof.finalize](#)时，会返回报错。
 - [acl.prof.start](#)接口在模型执行之前调用，若在模型执行过程中调用[acl.prof.start](#)接口，Profiling采集到的数据为调用[acl.prof.start](#)接口之后的数据，可能导致数据不完整。

调用[acl.prof.start](#)接口时，可以指定从一个Device上采集性能数据，也可以指定从多个Device上采集性能数据。

一个用户APP进程内，如果连续调用多次[acl.prof.start](#)接口，指定重复的Profiling配置，或指定的Device重复，会返回报错。

- 在用户APP的进程生命周期内，[acl.prof.init](#)接口与[acl.prof.finalize](#)接口配对使用，建议只调用一次，如该组合多次调用可以改变保存性能数据的文件的路径。
 - [acl.prof.start](#)接口与[acl.prof.stop](#)接口需配对使用。
 - Profiling msproftx采集功能相关接口须在[acl.prof.start](#)接口与[acl.prof.stop](#)接口之间调用。其中配对使用的接口有：[acl.prof.create_stamp](#)/[acl.prof.destroy_stamp](#)、[acl.prof.push](#)/[acl.prof.pop](#)、[acl.prof.range_start](#)/[acl.prof.range_stop](#)。
 - 调用[acl.finalize](#)并接收到正常退出码后为执行完毕，其他情况为非正常。由于性能数据采集不支持多进程并发执行，为确保驱动关闭正常，需要在前一个性能数据采集用例完全执行完毕之后再执行下一轮采集。建议在[acl.finalize](#)接口返回值上加入异常处理操作，方便展示执行状态与问题定位。
- 接口调用顺序：
 - 建议的接口调用顺序如下，以“一个用户APP进程内采集多个模型推理时的性能数据”为例：
[acl.init](#)接口-->[acl.prof.init](#)接口-->[acl.prof.start](#)接口(指定Device 0和Device 1)-->模型1加载-->模型1执行-->[acl.prof.stop](#)接口(与[acl.prof.start](#)接口的aclprofConfig数据保持一致)-->[acl.prof.start](#)接口(指定Device 1和Device 2)-->模型2加载-->模型2执行-->[acl.prof.stop](#)接口(与[acl.prof.start](#)接口的aclprofConfig数据保持一致)-->[acl.prof.finalize](#)接口-->执行其它任务-->模型卸载-->[acl.finalize](#)接口
 - 错误的接口调用顺序示例如下，以“一个用户APP进程内，如果连续调用多次[acl.prof.start](#)接口，指定的Device重复”为例：
[acl.init](#)接口-->[acl.prof.init](#)接口-->[acl.prof.start](#)接口(指定Device 0和Device 1)-->[acl.prof.start](#)接口(指定Device 1和Device 2)-->模型1加载-->模型1执行-->模型2加载-->模型2执行-->[acl.prof.stop](#)接口-->[acl.prof.stop](#)接口-->[acl.prof.finalize](#)-->执行其它任务-->模型卸载-->[acl.finalize](#)接口

2.15.1.2 函数：create_config

C函数原型	<code>aclprofConfig *aclprofCreateConfig(uint32_t *deviceIdlist, uint32_t deviceNums, aclprofAicoreMetrics aicoreMetrics, aclprofAicoreEvents *aicoreEvents, uint64_t dataTypeConfig)</code>
Python函数	<code>prof_config = acl.prof.create_config(device_list, aicore_metrics, aicore_events, data_type_config)</code>
函数功能	创建aclprofConfig类型的数据，表示创建Profiling配置数据。 aclProfConfig类型数据可以只创建一次、多处使用，用户需要保证数据的一致性和准确性。 如需销毁aclprofConfig类型的数据，请参见 2.15.1.4 函数：destroy_config 。

输入说明	device_list: Device ID列表。须根据实际环境的Device ID配置。 aicore_metrics: 表示aclprofAicoreMetrics。 aicore_events: 表示AI Core事件, 目前配置为0。 data_type_config: 用户选择如下多个值进行逻辑或 (例如: ACL_PROF_ACL_API ACL_PROF_AICORE_METRICS), 作为data_type_config参数值。 每个值表示某一类性能数据, 详细说明如下: • ACL_PROF_ACL_API: 表示采集pyACL接口的性能数据, 包括Host与Device之间的同步异步内存复制时延等。 • ACL_PROF_TASK_TIME: 采集算子下发耗时、算子执行耗时数据以及算子基本信息数据, 提供更全面的性能分析数据。 • ACL_PROF_TASK_TIME_L0: 采集算子下发耗时、算子执行耗时数据。与ACL_PROF_TASK_TIME相比, 由于不采集算子基本信息数据, 采集时性能开销较小, 可更精准统计相关耗时数据。 • ACL_PROF_AICORE_METRICS: 表示采集AI Core性能指标数据, 逻辑或时必须包括该宏, aicore_metrics入参处配置的性能指标采集项才有效。 • ACL_PROF_TASK_MEMORY: 控制CANN算子的内存占用情况采集开关。仅采集GE组件算子。 • ACL_PROF_AICPU: 表示采集AI CPU任务的开始、结束数据。 • ACL_PROF_L2CACHE: 表示采集L2 Cache数据。 • ACL_PROF_HCCL_TRACE: 控制hccl数据采集开关。 • ACL_PROF_MSPROFTX: 获取用户和上层框架程序输出的性能数据。需要先在应用程序脚本中添加2.15.2 Profiling pyACL API for Extension (Profiling pyACL API扩展接口)。 • ACL_PROF_TRAINING_TRACE: 控制迭代轨迹数据采集开关。 • ACL_PROF_RUNTIME_API: 控制runtime api性能数据采集开关。
返回值说明	prof_config: int。 • 返回非0值表示成功, 返回值为返回aclprofConfig类型的指针地址。 • 返回0表示失败。
约束说明	• 使用acl.prof.destroy_config接口销毁aclprofConfig类型的数据, 如不销毁会导致内存未被释放。 • 与acl.prof.destroy_config接口配对使用, 先调用acl.prof.create_config接口再调用acl.prof.destroy_config接口。

2.15.1.3 函数: set_config

C函数原型	aclError aclprofSetConfig(aclprofConfigType configType, const char *config, size_t configLength)

Python函数	ret = acl.prof.set_config(config_type, config)
函数功能	acl.prof.create_config 接口的扩展接口，用于设置性能数据采集参数。 该接口支持多次调用，用户需要保证数据的一致性和准确性。

输入说明	config_type: int, 作为config_type参数值。每个常量表示不同采集配置, 若要使用该接口下不同的选项采集多种性能数据, 则需要多次调用该接口, 详细说明如下: - ACL_PROF_STORAGE_LIMIT = 1: 指定落盘目录允许存放的最大文件容量, 有效取值范围为[200, 4294967295], 单位为MB。 - ACL_PROF_SYS_HARDWARE_MEM_FREQ = 3: 片上内存带宽及内存、LLC的读写带宽数据、acc_pmu数据和SoC传输带宽信息、组件内存采集频率, 范围[1,100]。 不同产品支持情况不同, 请以实际实现为准。 说明 已知在安装有glibc<2.34的环境上采集memory数据, 可能触发glibc的一个已知Bug 19329, 通过升级环境的glibc版本可解决此问题。 - ACL_PROF_LLC_MODE = 4: LLC Profiling采集事件。要求同时设置ACL_PROF_SYS_HARDWARE_MEM_FREQ。可以设置为: Atlas 200/300/500 推理产品: - capacity: 采集AI CPU和Control CPU的LLC capacity数据。 - bandwidth: 采集LLC bandwidth。默认值为capacity。 Atlas 推理系列产品: - read: 读事件, 三级缓存读速率。 - write: 写事件, 三级缓存写速率。默认为read。 Atlas 训练系列产品: - read: 读事件, 三级缓存读速率。 - write: 写事件, 三级缓存写速率。默认为read。 Atlas 200I/500 A2推理产品: - read: 读事件, 三级缓存读速率。 - write: 写事件, 三级缓存写速率。默认为read。 Atlas A2训练系列产品/Atlas 800I A2推理产品: - read: 读事件, 三级缓存读速率。 - write: 写事件, 三级缓存写速率。默认为read。 - ACL_PROF_SYS_IO_FREQ = 5: NIC、ROCE采集频率, 范围[1,100], 单位hz。 Atlas 200/300/500 推理产品: 支持采集NIC。 Atlas 200I/500 A2推理产品: 支持采集NIC, 容器场景参数不生效。 Atlas 训练系列产品: 支持采集NIC和和ROCE。 Atlas A2训练系列产品/Atlas 800I A2推理产品: 支持采集NIC和ROCE。 - ACL_PROF_SYS_INTERCONNECTION_FREQ = 6: 集合通信带宽数据 (HCCS)、PCIe数据采集开关、片间传输带宽信息采集频率, 范围[1,50], 单位hz。 Atlas 推理系列产品: 支持采集PCIe数据。

	- Atlas 训练系列产品：支持采集HCCS、PCIe数据。 - Atlas A2训练系列产品/Atlas 800I A2推理产品：支持采集HCCS、PCIe数据、片间传输带宽信息。 • ACL_PROF_DVPP_FREQ：DVPP采集频率，范围[1,100]。 Atlas 推理系列产品：支持采集性能数据，暂不支持解析。 • ACL_PROF_HOST_SYS = 8：Host侧进程级别的性能数据采集开关，取值包括cpu和mem。 • ACL_PROF_HOST_SYS_USAGE = 9：Host侧系统和所有进程的性能数据采集开关，取值包括cpu和mem。 • ACL_PROF_HOST_SYS_USAGE_FREQ = 10：CPU利用率、内存利用率的采集频率，范围[1,50]。 config ：str，配置项的参数值。
返回值说明	ret ：int，错误码。 • 返回0表示成功。 • 返回其它值表示失败。
约束说明	先调用acl.prof.set_config接口再调用 acl.prof.start 接口，可根据需求选择调用该接口。

2.15.1.4 函数：destroy_config

C函数原型	aclError aclprofDestroyConfig(const aclprofConfig *profilerConfig)
Python函数	ret = acl.prof.destroy_config(profiler_config)
函数功能	销毁aclprofConfig类型的数据，只能销毁通过 acl.prof.create_config 接口创建的aclprofConfig类型。
输入说明	profiler_config ：待销毁的aclprofConfig类型的指针地址。
返回值说明	ret ：int，错误码。 • 返回0表示成功。 • 返回其它值表示失败。
约束说明	• 与acl.prof.create_config接口配对使用，先调用acl.prof.create_config接口再调用aclprofDestroyConfig接口。 • 同一aclprofConfig数据重复调用acl.prof.destroy_config接口，会出现重复释放内存的报错。

2.15.1.5 函数：init

C函数原型	aclError aclprofInit(const char *profilerResultPath, size_t length)

Python函数	ret = acl.prof.init(profiler_result_path)
函数功能	初始化Profiling，目前用于设置保存性能数据的文件的路径。
输入说明	profiler_result_path : str, 指定保存性能数据的文件的路径，此路径为绝对路径。
返回值说明	ret : int, 错误码。 • 返回0表示成功。 • 返回其它值表示失败。
参考资源	接口调用示例，参见Profiling性能数据采集。

2.15.1.6 函数：start

C函数原型	aclError aclprofStart(const aclprofConfig *profilerConfig)
Python函数	ret = acl.prof.start(profiler_config)
函数功能	下发Profiling请求，使能对应数据的采集。 用户可根据需要，在模型执行过程中按需调用 acl.prof.start 接口，Profiling采集到的数据为调用该接口之后的数据。
输入说明	profiler_config : int, 指定Profiling配置数据。 需提前调用 acl.prof.create_config 接口创建aclprofConfig类型的数据。
返回值说明	ret : int, 错误码。 • 返回0表示成功。 • 返回其它值表示失败。
约束说明	与 acl.prof.stop 接口配对使用，先调用acl.prof.start接口再调用acl.prof.stop接口。
参考资源	接口调用示例，参见Profiling性能数据采集。

2.15.1.7 函数：stop

C函数原型	aclError aclprofStop(const aclprofConfig *profilerConfig)
Python函数	ret = acl.prof.stop(profiler_config)
函数功能	停止Profiling数据采集。
输入说明	profiler_config : int, 指定停止Profiling数据采集的配置的指针地址。 与 acl.prof.start 接口中的aclprofConfig类型数据保持一致。

返回值说明	ret : int, 错误码。 - 返回0表示成功。 - 返回其它值表示失败。
约束说明	与 acl.prof.start 接口配对使用, 先调用 acl.prof.start 接口再调用 acl.prof.stop 接口。
参考资源	接口调用示例, 参见Profiling性能数据采集。

2.15.1.8 函数: finalize

C函数原型	aclError aclprofFinalize()
Python函数	ret = acl.prof.finalize()
函数功能	结束Profiling。
输入说明	无。
返回值说明	ret : int, 错误码。 - 返回0表示成功。 - 返回其它值表示失败。
约束说明	与 acl.prof.init 接口配对使用, 先调用 acl.prof.init 接口再调用 acl.prof.finalize 接口。
参考资源	接口调用示例, 参见Profiling性能数据采集。

2.15.2 Profiling pyACL API for Extension (Profiling pyACL API 扩展接口)

2.15.2.1 功能介绍

功能

当用户需要定位应用程序或上层框架程序的性能瓶颈时, 可在Profiling采集进程内 ([acl.prof.start](#)接口、[acl.prof.stop](#)接口之间) 调用Profiling pyACL API扩展接口 (统称为msproftx功能), 开启记录应用程序执行期间特定事件发生的时间跨度, 并将数据写入Profiling数据文件, 再使用Profiling工具解析该文件, 并导出展示性能分析数据。

Profiling工具解析导出操作请参见《 [性能调优工具](#) 》下的“Profiling数据解析”和“Profiling数据导出”。

接口调用方式: 在[acl.prof.start](#)和[acl.prof.stop](#)接口之间调用[acl.prof.create_stamp](#)、[acl.prof.set_stamp_trace_message](#)、[acl.prof.mark](#)、[acl.prof.push](#)、[acl.prof.pop](#)、[acl.prof.range_start](#)、[acl.prof.range_stop](#)、[acl.prof.destroy_stamp](#)接口。该方式可获取应用程序执行期间特定时间发生的事件并记录事件发生的时间跨度。

一个进程内，可以根据需求多次调用这些接口。

接口约束说明

- **调用接口要求：**msproftx功能相关接口须在[acl.prof.start](#)接口与[acl.prof.stop](#)接口之间调用。其中配对使用的接口有：[acl.prof.create_stamp](#)/[acl.prof.destroy_stamp](#)、[acl.prof.push](#)/[acl.prof.pop](#)、[acl.prof.range_start](#)/[acl.prof.range_stop](#)。
- **接口调用顺序：**[acl.prof.start](#)接口(指定Device 0和Device 1)-->[acl.prof.create_stamp](#)接口-->[acl.prof.set_stamp_trace_message](#)接口-->[acl.prof.mark](#)接口-->([acl.prof.push](#)接口-->[acl.prof.pop](#)接口)或([acl.prof.range_start](#)接口-->[acl.prof.range_stop](#)接口)-->[acl.prof.destroy_stamp](#)接口-->[acl.prof.stop](#)接口(与[acl.prof.start](#)接口的aclprofConfig数据保持一致)。

2.15.2.2 函数：create_stamp

C函数原型	<code>void *aclprofCreateStamp(void)</code>
Python函数	<code>stamp = acl.prof.create_stamp()</code>
函数功能	创建msproftx事件标记。后续调用 2.15.2.4 函数：mark 、 2.15.2.3 函数：set_stamp_trace_message 、 2.15.2.5 函数：push 和 2.15.2.7 函数：range_start 接口时需要以描述该事件的指针地址作为输入，表示记录该事件发生的时间跨度。 与 2.15.2.9 函数：destroy_stamp 接口配对使用，需提前调用 2.15.1.6 函数：start 接口。
输入说明	无
返回值说明	stamp: int, 指针地址。 • 返回0表示失败。 • 返回void类型的指针地址表示成功。
约束说明	无
参考资源	接口调用示例，参见Profiling性能数据采集。

2.15.2.3 函数：set_stamp_trace_message

C函数原型	<code>aclError aclprofSetStampTraceMessage(void *stamp, const char *msg, uint32_t msgLen)</code>
Python函数	<code>ret = acl.prof.set_stamp_trace_message(stamp, msg, msg_len)</code>
函数功能	为msproftx事件标记携带字符串描述，在Profiling解析并导出结果中msprof_tx summary数据展示。 Profiling解析并导出结果请参见《 性能调优工具 》下的“性能数据文件参考>msproftx数据说明”。

输入说明	stamp : int, Stamp指针地址, 指代msproftx事件标记。指定 2.15.2.2 函数: create_stamp 接口的指针地址。 msg : int, 字符串内容地址。 msg_len : int, 字符串长度。
返回值说明	ret : int, 错误码。 • 返回0表示成功。 • 返回其它值表示失败。
约束说明	在 2.15.2.2 函数: create_stamp 接口和 2.15.2.9 函数: destroy_stamp 接口之间调用。
参考资源	接口调用示例, 参见Profiling性能数据采集。

2.15.2.4 函数: mark

C函数原型	aclError aclprofMark(void *stamp)
Python函数	ret = acl.prof.mark(stamp)
函数功能	msproftx标记瞬时事件。 调用此接口后, Profiling自动在Stamp中加上当前时间戳, 将Event type设置为Mark, 表示开始一次msproftx采集。
输入说明	stamp : int, Stamp指针地址, 指代msproftx事件标记。指定 2.15.2.2 函数: create_stamp 接口的指针地址。
返回值说明	ret : int, 错误码。 • 返回0表示成功。 • 返回其它值表示失败。
约束说明	在 2.15.2.2 函数: create_stamp 接口和 2.15.2.9 函数: destroy_stamp 接口之间调用。
参考资源	接口调用示例, 参见Profiling性能数据采集。

2.15.2.5 函数: push

C函数原型	aclError aclprofPush(void *stamp)
Python函数	ret = acl.prof.push(stamp)
函数功能	msproftx用于记录事件发生的时间跨度的开始时间。 调用此接口后, Profiling自动在Stamp中记录开始的时间戳, 将Event type设置为Push/Pop。
输入说明	stamp : int, Stamp指针地址, 指代msproftx事件标记。指定 acl.prof.create_stamp 接口的指针地址。

返回值说明	ret : int, 错误码。 • 返回0表示成功。 • 返回其它值表示失败。
约束说明	• 与2.15.2.6 函数: pop接口成对使用, 表示时间跨度的开始和结束。 • 不能跨线程调用, 若需要跨线程可使用2.15.2.7 函数: range_start/2.15.2.8 函数: range_stop接口。 • 在2.15.2.2 函数: create_stamp接口和2.15.2.9 函数: destroy_stamp接口之间调用。
参考资源	接口调用示例, 参见Profiling性能数据采集。

2.15.2.6 函数: pop

C函数原型	aclError aclprofPop()
Python函数	ret = acl.prof.pop()
函数功能	msproftx用于记录事件发生的时间跨度的结束时间。 调用此接口后, Profiling自动在Stamp中记录采集结束的时间戳。
输入说明	无
返回值说明	ret : int, 错误码。 • 返回0表示成功。 • 返回其它值表示失败。
约束说明	• 与2.15.2.5 函数: push接口成对使用, 表示时间跨度的开始和结束。 • 不能跨线程调用。若需要跨线程可使用2.15.2.7 函数: range_start/2.15.2.8 函数: range_stop接口。 • 在2.15.2.2 函数: create_stamp接口和2.15.2.9 函数: destroy_stamp接口之间调用。
参考资源	接口调用示例, 参见Profiling性能数据采集。

2.15.2.7 函数: range_start

C函数原型	aclError aclprofRangeStart(void *stamp, uint32_t *rangeld)
Python函数	range_id, ret = acl.prof.range_start(stamp)
函数功能	msproftx用于记录事件发生的时间跨度的开始时间。 调用此接口后, Profiling自动在Stamp记录采集开始的时间戳, 将Event type设置为Start/Stop, 生成一个进程唯一的id, 并将Stamp保存在以进程粒度维护的一个map中。

输入说明	stamp : int, Stamp指针地址, 指代msproftx事件标记。指定 2.15.2.2 函数: create_stamp 接口的指针地址。
返回值说明	range_id : int msproftx事件标记的唯一标识。用于在跨线程时区分。 ret : int, 错误码。 • 返回0表示成功。 • 返回其它值表示失败。
约束说明	• 与2.15.2.8 函数: range_stop接口成对使用, 表示时间跨度的开始和结束。 • 可以跨线程调用。 • 在2.15.2.2 函数: create_stamp接口和2.15.2.9 函数: destroy_stamp接口之间调用。
参考资源	接口调用示例, 参见Profiling性能数据采集。

2.15.2.8 函数: range_stop

C函数原型	aclError aclprofRangeStop(uint32_t *rangeld)
Python函数	ret = acl.prof.range_stop(range_id)
函数功能	msproftx用于记录事件发生的时间跨度的结束时间。 调用此接口后, Profiling自动在Stamp中记录采集结束的时间戳。
输入说明	range_id : int, msproftx事件标记的唯一标识。用于在跨线程时区分。
返回值说明	ret : int, 错误码。 • 返回0表示成功。 • 返回其它值表示失败。
约束说明	• 与2.15.2.8 函数: range_stop接口成对使用, 表示时间跨度的开始和结束。 • 可以跨线程调用。 • 在2.15.2.2 函数: create_stamp接口和2.15.2.9 函数: destroy_stamp接口之间调用。
参考资源	接口调用示例, 参见Profiling性能数据采集。

2.15.2.9 函数: destroy_stamp

C函数原型	void aclprofDestroyStamp(void *stamp)
Python函数	acl.prof.destroy_stamp(stamp)

函数功能	释放msproftx事件标记。
输入说明	stamp : int, Stamp指针地址, 指代msproftx事件标记。指定 2.15.2.2 函数: create_stamp 接口的指针地址。
返回值说明	无
约束说明	与 2.15.2.2 函数: create_stamp 接口配对使用, 在 2.15.1.7 函数: stop 接口前调用。
参考资源	接口调用示例, 参见Profiling性能数据采集。

2.15.3 Profiling pyACL API for Subscription (订阅算子信息的 Profiling pyACL API)

2.15.3.1 功能介绍

功能

该章节描述使用订阅算子信息的Profiling pyACL API采集性能数据, 实现将采集到的Profiling数据解析后写入管道, 由用户读入内存, 再由用户调用pyACL的接口获取性能数据。

接口调用方式: [acl.prof.model_subscribe](#)接口、[acl.prof.get*](#)接口、[acl.prof.model_unsubscribe](#)接口配合使用, 实现该方式的性能数据采集, 当前支持获取网络模型中算子的性能数据, 包括算子名称、算子类型名称、算子执行时间等。

总体约束

不能与[2.15.1 Profiling pyACL API \(通过Profiling pyACL API采集并落盘性能数据 \)](#)的接口交叉调用: [acl.prof.model_subscribe](#)接口和[acl.prof.model_unsubscribe](#)接口之间不能调用[acl.prof.init](#)接口、[acl.prof.start](#)接口、[acl.prof.stop](#)接口、[acl.prof.finalize](#)接口。

接口约束说明

- 接口调用要求:
 - [acl.prof.model_subscribe](#)接口在模型执行之前调用, 若在模型执行过程中调用[acl.prof.model_subscribe](#)接口, Profiling采集到的数据为调用[acl.prof.model_subscribe](#)接口之后的数据, 可能导致数据不完整。
 - [acl.prof.model_subscribe](#)接口需与[acl.prof.model_unsubscribe](#)接口配对使用, 不能在调用[acl.prof.model_unsubscribe](#)接口前, 多次调用[acl.prof.model_subscribe](#)接口重复订阅相同的模型。
 - 不能调用[acl.prof.model_subscribe](#)接口订阅不存在的模型ID。
 - 不能调用[acl.prof.model_unsubscribe](#)接口取消订阅不存在的模型ID或未订阅过的模型ID。
 - 如果在同一个Device上加载了多个模型, 只能对多个模型下发同样的订阅配置。

- 接口调用顺序：
 - 建议的接口调用顺序如下：
模型加载-->[acl.prof.model_subscribe](#)接口-->[acl.prof.get_op_desc_size](#)接口-->[acl.prof.get_op_num](#)接口-->[acl.prof.get_op_type/](#)
[acl.prof.get_op_name/acl.prof.get_op_start/acl.prof.get_op_end/](#)
[acl.prof.get_op_duration/acl.prof.get_model_id](#)接口-->[acl.prof.model_unsubscribe](#)接口
 - 错误的接口调用顺序示例如下，以重复定义同一个模型为例：
模型1加载-->[acl.prof.model_subscribe](#)接口(指定模型1)-->[acl.prof.model_subscribe](#)接口(指定模型1)-->[acl.prof.model_unsubscribe](#)接口

2.15.3.2 函数：create_subscribe_config

C函数原型	<code>aclprofSubscribeConfig *aclprofCreateSubscribeConfig(int8_t timeInfoSwitch, aclprofAicoreMetrics aicoreMetrics, void *fd)</code>
Python函数	<code>subscribe_config = acl.prof.create_subscribe_config(time_info_switch, aicore_metrics, fd)</code>
函数功能	创建 aclprofSubscribeConfig 类型的数据，表示创建订阅配置信息。 如需销毁 aclprofSubscribeConfig 类型的数据，请参见 acl.prof.destroy_subscribe_config 。
输入说明	time_info_switch : int，是否采集网络模型中算子的性能数据： ● 1：采集 ● 0：不采集 aicore_metrics : int，表示AI Core性能指标采集项，参考 aclprofAicoreMetrics 。 说明 订阅接口目前仅提供算子耗时统计的功能，暂时不支持AicoreMetrics采集功能。 fd : int，用户创建管道的写文件描述符。 用户在调用 acl.prof.model_unsubscribe 接口后，系统内部会在数据发送结束后，关闭该模型的管道写文件描述符。
返回值说明	subscribe_config : int。 ● 返回非0值表示成功，返回aclprofSubscribeConfig的地址对象。 ● 返回0表示失败。

约束说明	- 使用acl.prof.destroy_subscribe_config接口销毁aclprofSubscribeConfig类型的数据，如不销毁会导致内存未被释放。 - 与acl.prof.destroy_subscribe_config接口配对使用，先调用acl.prof.create_subscribe_config接口再调用acl.prof.destroy_subscribe_config接口。

2.15.3.3 函数：destroy_subscribe_config

C函数原型	<code>aclError aclprofDestroySubscribeConfig(const aclprofSubscribeConfig *profSubscribeConfig)</code>
Python函数	<code>ret = acl.prof.destroy_subscribe_config(subscribe_config)</code>
函数功能	销毁 aclprofSubscribeConfig 类型的数据，只能销毁通过 acl.prof.create_subscribe_config 接口创建的 aclprofSubscribeConfig 类型。
输入说明	subscribe_config : int，待销毁的 aclprofSubscribeConfig 类型的地址对象。
返回值说明	ret : int，错误码。 - 返回0表示成功。 - 返回其它值表示失败。
约束说明	- 与acl.prof.create_subscribe_config接口配对使用，先调用acl.prof.create_subscribe_config接口再调用acl.prof.destroy_subscribe_config接口。 - 同一aclprofSubscribeConfig重复调用acl.prof.destroy_subscribe_config接口，会出现重复释放内存的报错。

2.15.3.4 函数：model_subscribe

C函数原型	<code>aclError aclprofModelSubscribe(uint32_t modelId, const aclprofSubscribeConfig *profSubscribeConfig)</code>
Python函数	<code>ret = acl.prof.model_subscribe(model_id, subscribe_config)</code>
函数功能	网络场景下，订阅算子的基本信息，包括算子名称、算子类型、算子执行耗时等。

输入说明	model_id : int, 待订阅的网络模型的ID。调用 acl.mdl.load_from_file 接口/ acl.mdl.load_from_mem 接口/ acl.mdl.load_from_file_with_mem 接口/ acl.mdl.load_from_mem_with_mem 接口加载模型成功后, 会返回模型ID。 subscribe_config : int, 待订阅的配置信息。调用 acl.prof.create_subscribe_config 接口创建配置数据。
返回值说明	ret : int, 错误码。 - 返回0表示成功。 - 返回其它值表示失败。
约束说明	需要与 acl.prof.model_unsubscribe 接口配对使用。
参考资源	接口调用示例, 参见Profiling性能数据采集。

2.15.3.5 函数: model_un_subscribe

C函数原型	aclError aclprofModelUnSubscribe(uint32_t modelId)
Python函数	ret = acl.prof.model_un_subscribe(model_id)
函数功能	网络场景下, 取消订阅算子的基本信息, 包括算子名称、算子类型、算子执行耗时等。
输入说明	model_id : int, 已订阅的网络模型的ID。
返回值说明	ret : int, 错误码。 - 返回0表示成功。 - 返回其它值表示失败。
约束说明	需要与 acl.prof.model_subscribe 接口配对使用。
参考资源	接口调用示例, 参见Profiling性能数据采集。

2.15.3.6 函数: model_unsubscribe

C函数原型	aclError aclprofModelUnSubscribe(uint32_t modelId)
Python函数	ret = acl.prof.model_unsubscribe(model_id)
函数功能	网络场景下, 取消订阅算子的基本信息, 包括算子名称、算子类型、算子执行耗时等。
输入说明	model_id : int, 已订阅的网络模型的ID。
返回值说明	ret : int, 错误码。 - 返回0表示成功。 - 返回其它值表示失败。

约束说明	需要与 acl.prof.model_subscribe 接口配对使用。
参考资源	接口调用示例，参见Profiling性能数据采集。

2.15.3.7 函数：get_op_desc_size

C函数原型	<code>aclError aclprofGetOpDescSize(size_t *opDescSize)</code>
Python函数	<code>op_desc_size, ret = acl.prof.get_op_desc_size()</code>
函数功能	获取单个算子数据结构的大小，单位为Byte。当前版本中约定每个算子数据结构的大小是一样的。 建议用户新建一个线程，在新线程内调用该接口，否则可能阻塞主线程中的其它任务调度。
输入说明	无
返回值说明	op_desc_size : int，算子数据结构的大小。 ret : int，错误码。 - 返回0表示成功。 - 返回其它值表示失败。
约束说明	无
参考资源	接口调用示例，参见Profiling性能数据采集。

2.15.3.8 函数：get_op_num

C函数原型	<code>aclError aclprofGetOpNum(const void *opInfo, size_t opInfoLen, uint32_t *opNumber)</code>
Python函数	<code>op_num, ret = acl.prof.get_op_num(op_info, op_info_len)</code>
函数功能	获取指定内存中算子的数量。 建议用户新建一个线程，在新线程内调用该接口，否则可能阻塞主线程中的其它任务调度。
输入说明	op_info : int，指定算子信息的内存地址。 调用 acl.prof.get_op_desc_size 接口获取到单个算子数据结构的大小后，用户需按照“单个算子数据结构的大小*整数系数”得到的数值申请内存，用于存放Profiling采集到的算子信息数据，作为本接口的输入。 op_info_len : int，算子信息的长度。
返回值说明	op_num : int，算子的数量。 ret : int，错误码。 - 返回0表示成功。 - 返回其它值表示失败。

约束说明	无
参考资源	接口调用示例，参见Profiling性能数据采集。

2.15.3.9 函数：get_op_type_len

C函数原型	<code>aclError aclprofGetOpTypeLen(const void *opInfo, size_t opInfoLen, uint32_t index, size_t *opTypeLen)</code>
Python函数	<code>opTypeLen, ret = acl.prof.get_op_type_len(opInfo, opInfoLen, index)</code>
函数功能	获取算子类型的字符串长度，用于内存申请。 建议用户新建一个线程，在新线程内调用该接口，否则可能阻塞主线程中的其它任务调度。
输入说明	opInfo ：包含算子信息的地址。 opInfoLen ：算子信息的长度。 index ：指定获取第几个算子的算子类型名称。 用户调用 acl.prof.get_op_num 接口获取算子数量后，这个index的取值范围：[0, (算子数量-1)]。
返回值说明	opTypeLen ：opType的长度。 ret ：int，错误码。 - 返回0表示成功。 - 返回其它值表示失败。
参考资源	接口调用示例，参见Profiling性能数据采集。

2.15.3.10 函数：get_op_name_len

C函数原型	<code>aclError aclprofGetOpNameLen(const void *opInfo, size_t opInfoLen, uint32_t index, size_t *opNameLen)</code>
Python函数	<code>opNameLen, ret = acl.prof.get_op_name_len(opInfo, opInfoLen, index)</code>
函数功能	获取算子名称的字符串长度，用于内存申请。 建议用户新建一个线程，在新线程内调用该接口，否则可能阻塞主线程中的其它任务调度。
输入说明	opInfo ：包含算子信息的地址。 opInfoLen ：算子信息的长度。 index ：指定获取第几个算子的算子名称长度。 用户调用 acl.prof.get_op_num 接口获取算子数量后，这个index的取值范围：[0, (算子数量-1)]。

返回值说明	opNameLen : opName的实际内存申请长度。 ret : int, 错误码。 - 返回0表示成功。 - 返回其它值表示失败。
参考资源	接口调用示例, 参见Profiling性能数据采集。

2.15.3.11 函数: get_op_type

C函数原型	aclError aclprofGetOpType(const void *opInfo, size_t opInfoLen, uint32_t index, char *opType, size_t opTypeLen)
Python函数	op_type, ret = acl.prof.get_op_type(op_info, op_info_len, index, op_type_len)
函数功能	获取指定算子的算子类型名称。 建议用户新建一个线程, 在新线程内调用该接口, 否则可能阻塞主线程中的其它任务调度。 acl.prof.get_op_type 和 acl.prof.get_op_type_v2 接口功能一致, 但 acl.prof.get_op_type_v2 接口会获取算子类型长度, 并分配相应的空间, 不需要用户传参来指定算子类型所需空间大小, 建议优先使用 acl.prof.get_op_type_v2 接口。
输入说明	op_info : int, 指定算子信息的内存地址。 op_info_len : int, 算子信息的长度。 index : int, 指定获取第几个算子的算子类型名称。 用户调用 acl.prof.get_op_num 接口获取算子数量后, 这个index的取值范围: [0, (算子数量-1)]。 op_type_len : int, op_type的长度。取值范围: [1, 65]。
返回值说明	op_type : str, 算子类型名称。 ret : int, 错误码。 - 返回0表示成功。 - 返回其它值表示失败。
参考资源	接口调用示例, 参见Profiling性能数据采集。

2.15.3.12 函数: get_op_type_v2

C函数原型	aclError aclprofGetOpType(const void *opInfo, size_t opInfoLen, uint32_t index, char *opType, size_t opTypeLen)
Python函数	op_type, ret = acl.prof.get_op_type_v2(op_info, op_info_len, index)

函数功能	获取指定算子的算子类型名称。 建议用户新建一个线程，在新线程内调用该接口，否则可能阻塞主线程中的其它任务调度。 acl.prof.get_op_type_v2和acl.prof.get_op_type接口功能一致，但acl.prof.get_op_type_v2接口会获取算子类型长度，并分配相应的空间，不需要用户传参来指定算子类型所需空间大小，建议优先使用acl.prof.get_op_type_v2接口。
输入说明	op_info: int，指定算子信息的内存地址。 op_info_len: int，算子信息的长度。 index: int，指定获取第几个算子的算子类型名称。 用户调用acl.prof.get_op_num接口获取算子数量后，这个index的取值范围：[0, (算子数量-1)]。
返回值说明	op_type: str，算子类型名称。 ret: int，错误码。 • 返回0表示成功。 • 返回其它值表示失败。
参考资源	接口调用示例，参见Profiling性能数据采集。

2.15.3.13 函数：get_op_name

C函数原型	<code>aclError aclprofGetOpName(const void *opInfo, size_t opInfoLen, uint32_t index, char *opName, size_t opNameLen)</code>
Python函数	<code>op_name, ret = acl.prof.get_op_name(op_info, op_info_len, index, op_name_len)</code>
函数功能	获取指定算子的算子名称。 建议用户新建一个线程，在新线程内调用该接口，否则可能阻塞主线程中的其它任务调度。 acl.prof.get_op_name和acl.prof.get_op_name_v2接口功能一致，但acl.prof.get_op_name_v2接口会获取算子名称长度，并分配相应的空间，不需要用户传参来指定算子名称所需空间大小，建议优先使用acl.prof.get_op_name_v2接口。
输入说明	op_info: int，指定算子信息的内存地址。 op_info_len: int，算子信息的长度。 index: int，指定获取第几个算子的算子名称。 用户调用acl.prof.get_op_num接口获取算子数量后，这个index的取值范围：[0, (算子数量-1)]。 op_name_len: int，op_name的长度。取值范围：[1, 257]。

返回值说明	op_name : str, 算子名称。 ret : int, 错误码。 - 返回0表示成功。 - 返回其它值表示失败。
参考资源	接口调用示例, 参见Profiling性能数据采集。

2.15.3.14 函数: get_op_name_v2

C函数原型	aclError aclprofGetOpName(const void *opInfo, size_t opInfoLen, uint32_t index, char *opName, size_t opNameLen)
Python函数	op_name, ret = acl.prof.get_op_name_v2(op_info, op_info_len, index)
函数功能	获取指定算子的算子名称。 建议用户新建一个线程, 在新线程内调用该接口, 否则可能阻塞主线程中的其它任务调度。 acl.prof.get_op_name_v2 和 acl.prof.get_op_name 接口功能一致, 但 acl.prof.get_op_name_v2 接口会获取算子名称长度, 并分配相应的空间, 不需要用户传参来指定算子名称所需空间大小, 建议优先使用 acl.prof.get_op_name_v2 接口。
输入说明	op_info : int, 指定算子信息的内存地址。 op_info_len : int, 算子信息的长度。 index : int, 指定获取第几个算子的算子名称。 用户调用 acl.prof.get_op_num 接口获取算子数量后, 这个index的取值范围: [0, (算子数量-1)]。
返回值说明	op_name : str, 算子名称。 ret : int, 错误码。 - 返回0表示成功。 - 返回其它值表示失败。
参考资源	接口调用示例, 参见Profiling性能数据采集。

2.15.3.15 函数: get_op_start

C函数原型	uint64_t aclprofGetOpStart(const void *opInfo, size_t opInfoLen, uint32_t index)
Python函数	op_start = acl.prof.get_op_start(op_info, op_info_len, index)

函数功能	获取算子执行的开始时间，单位为ns。 建议用户新建一个线程，在新线程内调用该接口，否则可能阻塞主线程中的其它任务调度。
输入说明	op_info : int，指定算子信息的内存地址。 op_info_len : int，算子信息的长度。 index : int，指定获取第几个算子的算子名称。 用户调用 acl.prof.get_op_num 接口获取算子数量后，这个index的取值范围：[0, (算子数量-1)]。
返回值说明	op_start : int，算子执行的开始时间。
参考资源	接口调用示例，参见Profiling性能数据采集。

2.15.3.16 函数：get_op_end

C函数原型	uint64_t aclprofGetOpEnd(const void *opInfo, size_t opInfoLen, uint32_t index)
Python函数	op_end = acl.prof.get_op_end(op_info, op_info_len, index)
函数功能	获取算子执行的结束时间，单位为ns。 建议用户新建一个线程，在新线程内调用该接口，否则可能阻塞主线程中的其它任务调度。
输入说明	op_info : int，指定算子信息的内存地址。 op_info_len : int，算子信息的长度。 index : int，指定获取第几个算子的算子名称。 用户调用 acl.prof.get_op_num 接口获取算子数量后，这个index的取值范围：[0, (算子数量-1)]。
返回值说明	op_end : int，算子执行的结束时间。
参考资源	接口调用示例，参见Profiling性能数据采集。

2.15.3.17 函数：get_op_duration

C函数原型	uint64_t aclprofGetOpDuration(const void *opInfo, size_t opInfoLen, uint32_t index)
Python函数	op_duration = acl.prof.get_op_duration(op_info, op_info_len, index)
函数功能	获取算子执行的耗时时间，单位为ns。 建议用户新建一个线程，在新线程内调用该接口，否则可能阻塞主线程中的其它任务调度。

输入说明	op_info : int, 指定算子信息的内存地址。 op_info_len : int, 算子信息的长度。 index : int, 指定获取第几个算子的算子名称。 用户调用 acl.prof.get_op_num 接口获取算子数量后, 这个index的取值范围: [0, (算子数量-1)]。
返回值说明	op_duration : int, 算子执行的耗时时间。
约束说明	无
参考资源	接口调用示例, 参见Profiling性能数据采集。

2.15.3.18 函数: get_model_id

C函数原型	size_t aclprofGetModelId(const void *opInfo, size_t opInfoLen, uint32_t index)
Python函数	model_id = acl.prof.get_model_id(op_info, op_info_len, index)
函数功能	获取指定算子所在模型的ID。 建议用户新建一个线程, 在新线程内调用该接口, 否则可能阻塞主线程中的其它任务调度。
输入说明	op_info : int, 指定算子信息的内存地址。 op_info_len : int, 算子信息的长度。 index : int, 指定获取第几个算子的算子名称。 用户调用 acl.prof.get_op_num 接口获取算子数量后, 这个index的取值范围: [0, (算子数量-1)]。
返回值说明	model_id : int, 模型的ID。
参考资源	接口调用示例, 参见Profiling性能数据采集。

2.15.4 Profiling pyACL API (PyTorch 场景标记迭代时间)

2.15.4.1 函数: get_step_timestamp

C函数原型	aclError aclprofGetStepTimestamp(aclprofStepInfo* stepInfo, aclprofStepTag tag, aclrtStream stream)
Python函数	ret =acl.prof.get_step_timestamp(stepinfo, tag, stream)
函数功能	利用 单算子加载与执行接口 实现训练的场景下, 使用本接口用于标记迭代开始与结束时间, 为后续Profiling解析提供迭代标识, 以便以迭代为粒度展示性能数据。

输入说明	stepinfo : 指定迭代信息。需提前调用 acl.prof.create_step_info 接口创建aclprofStepInfo类型的数据。 tag : 用于标记迭代开始或结束。在迭代开始时传入ACL_STEP_START, 迭代结束时需传入ACL_STEP_END。 stream : 指定Stream。
返回值说明	ret : int, 错误码。 - 返回0表示成功。 - 返回其它值表示失败。

2.15.4.2 函数: create_step_info

C函数原型	aclprofStepInfo* aclprofCreateStepInfo(void)
Python函数	info = acl.prof.create_step_info()
函数功能	创建aclprofStepInfo对象, 用于描述迭代信息。
输入说明	无
返回值说明	info : aclprofStepInfo类型指针地址。 - 返回aclprofStepInfo类型, 表示成功。 - 返回None, 表示失败。
约束说明	- 使用acl.prof.destroy_step_info接口销毁aclprofStepInfo类型的数据, 如不销毁会导致内存未被释放。 - 与acl.prof.destroy_step_info接口配对使用, 先调用acl.prof.create_step_info接口再调用acl.prof.destroy_step_info接口。 - 同一个aclprofStepInfo对象、同一个tag只能设置一次, 否则Profiling解析会出错。

2.15.4.3 函数: destroy_step_info

C函数原型	void aclprofDestroyStepInfo (aclprofStepInfo * stepinfo)
Python函数	acl.prof.destroy_step_info(stepinfo)
函数功能	销毁通过 acl.prof.create_step_info 接口创建的aclprofStepInfo类型的数据。
输入说明	stepinfo : 指定迭代信息, 需提前调用 acl.prof.create_step_info 接口创建aclprofStepInfo类型的数据。
返回值说明	无

约束说明	- 与acl.prof.create_step_info接口配对使用，先调用acl.prof.create_step_info接口再调用acl.prof.destroy_step_info接口。 - 同一aclprofStepInfo数据重复调用acl.prof.destroy_step_info接口，会出现重复释放内存的报错。

2.16 数据类型及其操作接口

2.16.1 aclError

 说明

返回码定义规则：

- 规则1：开发人员的环境异常或者代码逻辑错误，可以通过优化环境或代码逻辑的方式解决问题，此时返回码定义为：1XXXXX。
- 规则2：资源不足（Stream、内存等）、开发人员编程时使用的接口或参数与当前硬件不匹配，可以通过在编程时合理使用资源的方式解决，此时返回码定义为：2XXXXX。
- 规则3：业务功能异常，比如队列满、队列空等，此时返回码定义为3XXXXX。
- 规则4：软硬件内部异常，包括软件内部错误、Device执行失败等，用户无法解决问题，需要将问题反馈给技术支持，此时返回码定义为：5XXXXX。请获取日志后单击[Link](#)联系技术支持。
- 规则5：无法识别的错误，当前都映射为500000。请获取日志后单击[Link](#)联系技术支持。

表 2-71 返回码列表

返回码	说明	可能原因及解决方法
ACL_SUCCESS = 0	执行成功。	-
ACL_ERROR_NONE = 0 须知 此返回码后续版本会废弃， 请使用ACL_SUCCESS返回码。	执行成功。	-
ACL_ERROR_INVALID_PARAMETER = 100000	参数校验失败。	请检查接口的入参值是否正确。
ACL_ERROR_UNINITIALIZED = 100001	ACL未初始化。	- 请检查是否已调用acl.init接口进行初始化，请确保已调用acl.init接口，且在其它pyACL接口之前调用。 - 请检查是否已调用对应功能的初始化接口，例如初始化Dump的acl.mdl.init_dump接口、初始化Profiling的acl.prof.init接口。

返回码	说明	可能原因及解决方法
ACL_ERROR_REPEAT_INITIALIZE = 100002	重复初始化或重复加载。	请检查是否调用对应的接口重复初始化或重复加载。
ACL_ERROR_INVALID_FILE = 100003	无效的文件。	请检查文件是否存在、文件是否能被访问等。
ACL_ERROR_WRITE_FILE = 100004	写文件失败。	请检查文件路径是否存在、文件是否有写权限等。
ACL_ERROR_INVALID_FILE_SIZE = 100005	无效的文件大小。	请检查文件大小是否符合接口要求。
ACL_ERROR_PARSE_FILE = 100006	解析文件失败。	请检查文件内容是否合法。
ACL_ERROR_FILE_MISSING_ATTR = 100007	文件缺失参数。	请检查文件内容是否完整。
ACL_ERROR_FILE_ATTR_INVALID = 100008	文件参数无效。	请检查文件中参数值是否正确。
ACL_ERROR_INVALID_DUMP_CONFIG = 100009	无效的Dump配置。	请检查Dump配置是否正确，详细配置请参见《精度比对工具使用指南》。
ACL_ERROR_INVALID_PROFILING_CONFIG = 100010	无效的Profiling配置。	请检查Profiling配置是否正确。
ACL_ERROR_INVALID_MODEL_ID = 100011	无效的模型ID。	请检查模型ID是否正确、模型是否正确加载。
ACL_ERROR_DESERIALIZE_MODEL = 100012	反序列化模型失败。	模型可能与当前版本不匹配，请重新构建模型。
ACL_ERROR_PARSE_MODEL = 100013	解析模型失败。	模型可能与当前版本不匹配，请重新构建模型。
ACL_ERROR_READ_MODEL_FAILURE = 100014	读取模型失败。	请检查模型文件是否存在、模型文件是否能被访问等。
ACL_ERROR_MODEL_SIZE_INVALID = 100015	无效的模型大小。	模型文件无效，请重新构建模型。
ACL_ERROR_MODEL_MISSING_ATTR = 100016	模型缺少参数。	模型可能与当前版本不匹配，请重新构建模型。
ACL_ERROR_MODEL_INPUT_NOT_MATCH = 100017	模型的输入不匹配。	请检查模型的输入是否正确。

返回码	说明	可能原因及解决方法
ACL_ERROR_MODEL_OUTPUT_NOT_MATCH = 100018	模型的输出不匹配。	请检查模型的输出是否正确。
ACL_ERROR_MODEL_NOT_DYNAMIC = 100019	非动态模型。	请检查当前模型是否支持动态场景，如不支持，请重新构建模型。
ACL_ERROR_OP_TYPE_NOT_MATCH = 100020	单算子类型不匹配。	请检查算子类型是否正确。
ACL_ERROR_OP_INPUT_NOT_MATCH = 100021	单算子的输入不匹配。	请检查算子的输入是否正确。
ACL_ERROR_OP_OUTPUT_NOT_MATCH = 100022	单算子的输出不匹配。	请检查算子的输出是否正确。
ACL_ERROR_OP_ATTR_NOT_MATCH = 100023	单算子的属性不匹配。	请检查算子的属性是否正确。
ACL_ERROR_OP_NOT_FOUND = 100024	单算子未找到。	请检查算子类型是否支持。
ACL_ERROR_OP_LOAD_FAILED = 100025	单算子加载失败。	模型可能与当前版本不匹配，请重新构建单算子模型。
ACL_ERROR_UNSUPPORTED_DATA_TYPE = 100026	不支持的数据类型。	请检查数据类型是否存在或当前是否支持。
ACL_ERROR_FORMAT_NOT_MATCH = 100027	Format不匹配。	请检查Format是否正确。
ACL_ERROR_BIN_SELECTOR_NOT_REGISTERED = 100028	使用二进制选择方式编译算子接口时，算子未注册选择器。	请检查是否调用 acl.op.register_compile_func 接口注册算子选择器。
ACL_ERROR_KERNEL_NOT_FOUND = 100029	编译算子时，算子Kernel未注册。	请检查是否调用 acl.op.create_kernel 接口注册算子Kernel。
ACL_ERROR_BIN_SELECTOR_ALREADY_REGISTERED = 100030	使用二进制选择方式编译算子接口时，算子重复注册。	请检查是否重复调用 acl.op.register_compile_func 接口注册算子选择器。
ACL_ERROR_KERNEL_ALREADY_REGISTERED = 100031	编译算子时，算子Kernel重复注册。	请检查是否重复调用 acl.op.create_kernel 接口注册算子Kernel。
ACL_ERROR_INVALID_QUEUE_ID = 100032	无效的队列ID。	请检查队列ID是否正确。

返回码	说明	可能原因及解决方法
ACL_ERROR_REPEAT_SUBSCRIBE = 100033	重复订阅。	请检查针对同一个Stream，是否重复调用 acl.rt.subscribe_report 接口。
ACL_ERROR_STREAM_NOT_SUBSCRIBE = 100034 须知 此返回码后续版本会废弃，请使用 ACL_ERROR_RT_STREAM_NO_CB_REG 返回码。	Stream未订阅。	请检查是否已调用 acl.rt.subscribe_report 接口。
ACL_ERROR_THREAD_NOT_SUBSCRIBE = 100035 须知 此返回码后续版本会废弃，请使用 ACL_ERROR_RT_THREAD_SUBSCRIBE 返回码。	线程未订阅。	请检查是否已调用 acl.rt.subscribe_report 接口。
ACL_ERROR_WAIT_CALLBACK_TIMEOUT = 100036 须知 此返回码后续版本会废弃，请使用 ACL_ERROR_RT_REPORT_TIMEOUT 返回码。	等待callback超时。	请检查是否已调用 acl.rt.launch_callback 接口下发callback任务。 请检查 acl.rt.process_report 接口中超时时间是否合理。 请检查callback任务是否已经处理完成，如果已处理完成，但还调用 acl.rt.process_report 接口，则需优化代码逻辑。
ACL_ERROR_REPEAT_FINALIZE = 100037	重复去初始化。	请检查是否重复调用 acl.finalize 接口进行去初始化。
ACL_ERROR_NOT_STATIC_AIPP = 100038 须知 此返回码后续版本会废弃，请使用 ACL_ERROR_GE_AIPP_NOT_EXIST 返回码。	静态AIPP配置信息不存在。	调用 acl.mdl.get_first_aipp_info 接口时，请传入正确的index值。
ACL_ERROR_COMPILING_STUB_MODE = 100039	运行应用前配置的动态库路径是编译桩的路径，不是正确的动态库路径。	请检查动态库路径的配置，确保使用运行模式的动态库。

返回码	说明	可能原因及解决方法
ACL_ERROR_PROF_ALREADY_RUN = 100042	已存在采集Profiling数据的任务。	- 请检查代码逻辑，“通过调用pyACL API方式采集Profiling数据”的配置不能与其它方式的Profiling配置并存，只能保留一种，各种方式的Profiling采集配置请参见《性能调优工具》。 - 请检查是否对同一个Device重复下发了多次Profiling配置，请参考2.15.1 Profiling pyACL API (通过Profiling pyACL API采集并落盘性能数据)中的接口调用要求、接口调用顺序调整代码逻辑。
ACL_ERROR_PROF_NOT_RUN = 100043	未使用 acl.prof.init 接口先进行Profiling初始化。	请检查采集Profiling数据的接口调用顺序，参考 2.15.1 Profiling pyACL API (通过Profiling pyACL API采集并落盘性能数据) 中的说明。
ACL_ERROR_DUMP_ALREADY_RUN = 100044	已存在获取Dump数据的任务。	请检查在调用 acl.mdl.init_dump 接口、 acl.mdl.set_dump 接口、 acl.mdl.finalize_dump 接口配置Dump信息前，是否已调用 acl.init 接口配置Dump信息，如是，请调整代码逻辑，保留一种方式配置Dump信息即可。
ACL_ERROR_DUMP_NOT_RUN = 100045	未使用 acl.mdl.init_dump 接口先进行Dump初始化。	请检查获取Dump数据的接口调用顺序，参考 acl.mdl.init_dump 接口处的说明。
ACL_ERROR_PROF_REPEAT_SUBSCRIBE = 148046	重复订阅同一个模型。	请检查接口调用顺序，参考 2.15.1 Profiling pyACL API (通过Profiling pyACL API采集并落盘性能数据) 中的说明。

返回码	说明	可能原因及解决方法
ACL_ERROR_PROF_API_CONFLICT = 148047	采集性能数据的接口调用冲突。	两种方式的Profiling性能数据采集接口不能交叉调用， acl.prof.init 接口和 acl.prof.finalize 接口之间不能调用 acl.prof.model_subscribe 接口、 acl.prof.get_op_* 接口、 acl.prof.model_unsubscribe 接口， acl.prof.model_subscribe 接口和 acl.prof.model_unsubscribe 接口之间不能调用 acl.prof.init 接口、 acl.prof.start 接口、 acl.prof.stop 接口、 acl.prof.finalize 接口。
ACL_ERROR_INVALID_MAX_OPQUEUE_NUM_CONFIG = 148048	无效的算子缓存信息老化配置。	请检查算子缓存信息老化配置，参考 acl.init 处的配置说明及示例。
ACL_ERROR_INVALID_OPP_PATH = 148049	没有设置ASCEND_OPP_PATH环境变量，或该环境变量的值设置错误。	请检查是否设置ASCEND_OPP_PATH环境变量，且该环境变量的值是否为opp软件包的安装路径。
ACL_ERROR_OP_UNSUPPORTED_DYNAMIC = 148050;	算子不支持动态Shape。	- 请检查单算子模型文件中该算子的Shape是否为动态，如果是动态的，需要修改为固定Shape。 - 请检查编译算子时，aclTensorDesc的Shape是否为动态，如果是动态的，需要按照固定Shape重新创建aclTensorDesc。
ACL_ERROR_RELATIVE_RESOURCE_NOT_CLEARED = 148051;	相关的资源尚未释放。	在销毁通道描述信息时，如果相关的通道尚未销毁则返回此错误码。请检查与此通道描述信息相关联的通道是否被销毁。

返回码	说明	可能原因及解决方法
ACL_ERROR_UNSUPPORTED_JPEG = 148052;	JPEGD功能不支持的输入图片编码格式（例如算术编码、渐进式编码等）。	实现JPEGD图片解码功能时，仅支持Huffman编码，压缩前的原图像色彩空间为YUV，像素的各分量比例为4:4:4或4:2:2或4:2:0或4:0:0或4:4:0，不支持算术编码、不支持渐进JPEG格式、不支持JPEG2000格式。
ACL_ERROR_BAD_ALLOC = 200000	申请内存失败。	请检查硬件环境上的内存剩余情况。
ACL_ERROR_API_NOT_SUPPORTED = 200001	接口不支持。	请检查调用的接口当前是否支持。
ACL_ERROR_INVALID_DEVICE = 200002 须知 此返回码后续版本会废弃，请使用 ACL_ERROR_RT_INVALID_DEVICEID 返回码。	无效的Device。	请检查Device是否存在。
ACL_ERROR_MEMORY_ADDRESS_UNALIGNED = 200003	内存地址未对齐。	请检查内存地址是否符合接口要求。
ACL_ERROR_RESOURCE_NOT_MATCH = 200004	资源不匹配。	请检查调用接口时，是否传入正确的Stream、Context等资源。
ACL_ERROR_INVALID_RESOURCE_HANDLE = 200005	无效的资源句柄。	请检查调用接口时，传入的Stream、Context等资源是否已被销毁或占用。
ACL_ERROR_FEATURE_UNSUPPORTED = 200006	特性不支持。	请获取日志后单击 Link 联系技术支持。
ACL_ERROR_PROF_MODULES_UNSUPPORTED = 200007	下发了不支持的Profiling配置。	请参见 acl.prof.create_config 中的说明检查Profiling的配置是否正确。
ACL_ERROR_STORAGE_OVERFLOW_LIMIT = 300000	超出存储上限。	请检查硬件环境上的存储剩余情况。
ACL_ERROR_INTERNAL_ERROR = 500000	未知内部错误。	请获取日志后单击 Link 联系技术支持。
ACL_ERROR_FAILURE = 500001	系统内部ACL的错误。	请获取日志后单击 Link 联系技术支持。

返回码	说明	可能原因及解决方法
ACL_ERROR_GE_FAILURE = 500002	系统内部GE的错误。	请获取日志后单击 Link 联系技术支持。
ACL_ERROR_RT_FAILURE = 500003	系统内部RUNTIME的错误。	请获取日志后单击 Link 联系技术支持。
ACL_ERROR_DRV_FAILURE = 500004	系统内部DRV (Driver) 的错误。	请获取日志后单击 Link 联系技术支持。
ACL_ERROR_PROFILING_FAILURE = 500005	Profiling相关错误。	请获取日志后单击 Link 联系技术支持。

表 2-72 内部 RUNTIME 的返回码列表

返回码	含义	可能原因及解决方法
ACL_RT_SUCCESS = 0	执行成功。	-
ACL_ERROR_RT_PARAM_INVALID = 107000	参数校验失败。	请检查接口入参是否正确。
ACL_ERROR_RT_INVALID_DEVICEID = 107001	无效的Device ID。	请检查Device ID是否合法。
ACL_ERROR_RT_CONTEXT_NULL = 107002	context为空。	请检查是否调用 acl.rt.set_context 或 acl.rt.set_device 。
ACL_ERROR_RT_STREAM_CONTEXT = 107003	stream不在当前Context内。	请检查Stream所在的Context与当前Context是否一致。
ACL_ERROR_RT_MODEL_CONTEXT = 107004	model不在当前Context内。	请检查加载的模型与当前Context是否一致。
ACL_ERROR_RT_STREAM_MODEL = 107005	stream不在当前model内。	请检查Stream是否绑定过该模型。
ACL_ERROR_RT_EVENT_TIMESTAMP_INVALID = 107006	event时间戳无效。	请检查Event是否创建。
ACL_ERROR_RT_EVENT_TIMESTAMP_REVERSAL = 107007	event时间戳反转。	请检查Event是否创建。
ACL_ERROR_RT_ADDR_UNALIGNED = 107008	内存地址未对齐。	请检查所申请的内存地址是否对齐，详细内存申请接口的约束请参见 2.7.6 内存管理 。
ACL_ERROR_RT_FILE_OPEN = 107009	打开文件失败。	请检查文件是否存在。

返回码	含义	可能原因及解决方法
ACL_ERROR_RT_FILE_WRITE = 107010	写文件失败。	请检查文件是否存在或者是否具备写权限。
ACL_ERROR_RT_STREAM_SUBSCRIBE = 107011	stream未订阅或重复订阅。	请检查当前stream是否订阅或重复订阅。
ACL_ERROR_RT_THREAD_SUBSCRIBE = 107012	线程未订阅或重复订阅。	请检查当前线程是否订阅或重复订阅。
ACL_ERROR_RT_GROUP_NOT_SET = 107013	未设置Group。	-
ACL_ERROR_RT_GROUP_NOT_CREATE = 107014	未创建对应的Group。	请检查调用接口时设置的Group ID是否在支持的范围内，Group ID的取值范围：[0, (Group数量-1)]。
ACL_ERROR_RT_STREAM_NO_CB_REG = 107015	该callback对应的Stream未注册到线程。	请检查stream是否已经注册到线程，检查是否调用 acl.rt.subscribe_report 接口。
ACL_ERROR_RT_INVALID_MEMORY_TYPE = 107016	无效的内存类型。	请检查内存类型是否合法。
ACL_ERROR_RT_INVALID_HANDLE = 107017	无效的资源句柄。	检查对应输入和使用的参数是否正确。
ACL_ERROR_RT_INVALID_MALLOC_TYPE = 107018	申请使用的内存类型不正确。	检查对应输入和使用的内存类型是否正确。
ACL_ERROR_RT_WAIT_TIMEOUT = 107019	执行任务超时。	请尝试重新执行下发任务的接口。
ACL_ERROR_RT_TASK_TIMEOUT = 107020	执行任务超时。	请排查业务编排是否合理或者设置合理的超时时间。
ACL_ERROR_RT_SYSPARAMOPT_NOT_SET = 107021	获取当前Context中的系统参数值失败。	未设置当前Context中的系统参数值。
ACL_ERROR_RT_FEATURE_NOT_SUPPORT = 207000	特性不支持。	请获取日志后单击 Link 联系技术支持。
ACL_ERROR_RT_MEMORY_ALLOCATION = 207001	内存申请失败。	请检查硬件环境上的存储剩余情况。
ACL_ERROR_RT_MEMORY_FREE = 207002	内存释放失败。	请获取日志后单击 Link 联系技术支持。

返回码	含义	可能原因及解决方法
ACL_ERROR_RT_AICORE_OVER_FLOW = 207003	aicore算子运算溢出。	请检查对应的aicore算子运算是否有溢出。
ACL_ERROR_RT_NO_DEVICE = 207004	Device不可用。	请检查Device是否正常运行。
ACL_ERROR_RT_RESOURCE_ALLOC_FAIL = 207005	内存申请失败。	请检查硬件环境上的存储剩余情况。
ACL_ERROR_RT_NO_PERMISSION = 207006	没有操作权限。	请检查运行应用的用户权限是否正确。
ACL_ERROR_RT_NO_EVENT_RESOURCE = 207007	Event资源不足。	请参考 acl.rt.create_event 接口处的说明检查Event数量是否符合要求。
ACL_ERROR_RT_NO_STREAM_RESOURCE = 207008	Stream资源不足。	请参考 acl.rt.create_stream 接口处的说明检查Stream数量是否符合要求。
ACL_ERROR_RT_NO_NOTIFY_RESOURCE = 207009	系统内部Notify资源不足。	数据预处理的并发任务太多或模型推理时消耗资源太多，建议尝试减少并发任务或卸载部分模型。
ACL_ERROR_RT_NO_MODEL_RESOURCE = 207010	模型资源不足。	建议卸载部分模型。
ACL_ERROR_RT_NO_RUNTIME_RESOURCE = 207011;	Runtime内部资源不足。	请获取日志后单击 Link 联系技术支持。
ACL_ERROR_RT_OVER_LIMIT = 207012;	队列数目超出上限。	请销毁不需要的队列之后再创建新的队列。
ACL_ERROR_RT_QUEUE_EMPTY = 207013;	队列为空。	不能从空队列中获取数据，请先向队列中添加数据，再获取。
ACL_ERROR_RT_QUEUE_FULL = 207014;	队列已满。	不能向已满的队列中添加数据，请先从队列中获取数据，再添加。
ACL_ERROR_RT_REPEATED_INIT = 207015;	队列重复初始化。	建议初始化一次队列即可，不要重复初始化。
static const int32_t ACL_ERROR_RT_DEVICE_OOM = 207018;	Device侧内存耗尽。	排查Device上的内存使用情况，并根据Device上的内存规格合理规划内存的使用。

返回码	含义	可能原因及解决方法
ACL_ERROR_RT_INTERNAL_ERROR = 507000	Host上的runtime模块内部错误。	请获取日志后单击 Link 联系技术支持。
ACL_ERROR_RT_TASK_SCHEDULER_ERROR = 507001	Device上的task scheduler模块内部错误。	请获取日志后单击 Link 联系技术支持。
ACL_ERROR_RT_STREAM_TASK_FULL = 507002	stream上的task数量满。	请获取日志后单击 Link 联系技术支持。
ACL_ERROR_RT_STREAM_TASK_EMPTY = 507003	stream上的task数量为空。	请获取日志后单击 Link 联系技术支持。
ACL_ERROR_RT_STREAM_TASK_NOT_COMPLETE = 507004	stream上的task未全部执行完成。	请获取日志后单击 Link 联系技术支持。
ACL_ERROR_RT_END_OF_SEQUENCE = 507005	AI CPU上的task执行完成。	请获取日志后单击 Link 联系技术支持。
ACL_ERROR_RT_EVENT_NOT_COMPLETE = 507006	event未完成。	请获取日志后单击 Link 联系技术支持。
ACL_ERROR_RT_CONTEXT_RELEASE_ERROR = 507007	context释放失败。	请获取日志后单击 Link 联系技术支持。
ACL_ERROR_RT_SOC_VERSION = 507008	获取soc version失败。	请获取日志后单击 Link 联系技术支持。
ACL_ERROR_RT_TASK_TYPE_NOT_SUPPORT = 507009	不支持的task类型。	请获取日志后单击 Link 联系技术支持。
ACL_ERROR_RT_LOST_HEARTBEAT = 507010	task scheduler丢失心跳。	请获取日志后单击 Link 联系技术支持。
ACL_ERROR_RT_MODEL_EXECUTE = 507011	模型执行失败。	请获取日志后单击 Link 联系技术支持。
ACL_ERROR_RT_REPORT_TIMEOUT = 507012	获取task scheduler的消息失败。	排查接口的超时时间设置是否过短，适当增长超时时间。如果增长超时时间后，依然有超时报错，再排查日志。 请获取日志后单击 Link 联系技术支持。
ACL_ERROR_RT_SYS_DMA = 507013	system dma (Direct Memory Access) 硬件执行错误。	请获取日志后单击 Link 联系技术支持。
ACL_ERROR_RT_AICORE_TIMEOUT = 507014	aicore执行超时。	请获取日志后单击 Link 联系技术支持。

返回码	含义	可能原因及解决方法
ACL_ERROR_RT_AICORE_EXCEPTION = 507015	aicore执行异常。	请获取日志后单击 Link 联系技术支持。
ACL_ERROR_RT_AICORE_TRAP_EXCEPTION = 507016	aicore trap执行异常。	请获取日志后单击 Link 联系技术支持。
ACL_ERROR_RT_AICPU_TIMEOUT = 507017	aicpu执行超时。	请获取日志后单击 Link 联系技术支持。
ACL_ERROR_RT_AICPU_EXCEPTION = 507018	aicpu执行异常。	请获取日志后单击 Link 联系技术支持。
ACL_ERROR_RT_AICPU_DATA_DUMP_RSP_ERR = 507019	aicpu执行数据dump后未给task scheduler返回响应。	请获取日志后单击 Link 联系技术支持。
ACL_ERROR_RT_AICPU_MODEL_RSP_ERR = 507020	aicpu执行模型后未给task scheduler返回响应。	请获取日志后单击 Link 联系技术支持。
ACL_ERROR_RT_PROFILING_ERROR = 507021	profiling功能执行异常。	请获取日志后单击 Link 联系技术支持。
ACL_ERROR_RT_IPC_ERROR = 507022	进程间通信异常。	请获取日志后单击 Link 联系技术支持。
ACL_ERROR_RT_MODEL_ABORT_NORMAL = 507023	模型退出。	请获取日志后单击 Link 联系技术支持。
ACL_ERROR_RT_KERNEL_UNREGISTERING = 507024	算子正在去注册。	请获取日志后单击 Link 联系技术支持。
ACL_ERROR_RT_RINGBUFFER_NOT_INIT = 507025	ringbuffer（环形缓冲区）功能未初始化。	请获取日志后单击 Link 联系技术支持。
ACL_ERROR_RT_RINGBUFFER_NO_DATA = 507026	ringbuffer（环形缓冲区）没有数据。	请获取日志后单击 Link 联系技术支持。
ACL_ERROR_RT_KERNEL_LOOKUP = 507027	RUNTIME内部的kernel未注册。	请获取日志后单击 Link 联系技术支持。
ACL_ERROR_RT_KERNEL_DUPLICATE = 507028	重复注册RUNTIME内部的kernel。	请获取日志后单击 Link 联系技术支持。
ACL_ERROR_RT_DEBUG_REGISTER_FAIL = 507029	debug功能注册失败。	请获取日志后单击 Link 联系技术支持。
ACL_ERROR_RT_DEBUG_UNREGISTER_FAIL = 507030	debug功能去注册失败。	请获取日志后单击 Link 联系技术支持。

返回码	含义	可能原因及解决方法
ACL_ERROR_RT_LABEL_CONTEXT = 507031	标签不在当前Context内。	请获取日志后单击 Link 联系技术支持。
ACL_ERROR_RT_PROGRAM_USE_OUT = 507032	注册的program数量超过限制。	请获取日志后单击 Link 联系技术支持。
ACL_ERROR_RT_DEVICE_SETUP_ERROR = 507033	Device启动失败。	请获取日志后单击 Link 联系技术支持。
ACL_ERROR_RT_VECTOR_CORE_TIMEOUT = 507034;	vector core执行超时。	请获取日志后单击 Link 联系技术支持。
ACL_ERROR_RT_VECTOR_CORE_EXCEPTION = 507035;	vector core执行异常。	请获取日志后单击 Link 联系技术支持。
ACL_ERROR_RT_VECTOR_CORE_TRAP_EXCEPTION = 507036;	vector core trap执行异常。	请获取日志后单击 Link 联系技术支持。
ACL_ERROR_RT_CDQ_BATCH_ABNORMAL = 507037;	Runtime内部资源申请异常。	请获取日志后单击 Link 联系技术支持。
ACL_ERROR_RT_DIE_MODE_CHANGE_ERROR = 507038;	die模式修改异常，不能修改die模式。	请获取日志后单击 Link 联系技术支持。
ACL_ERROR_RT_DIE_SET_ERROR = 507039;	单die模式不能指定die。	请获取日志后单击 Link 联系技术支持。
ACL_ERROR_RT_INVALID_DIEID = 507040;	指定die id错误。	请获取日志后单击 Link 联系技术支持。
ACL_ERROR_RT_DIE_MODE_NOT_SET = 507041;	die模式没有设置。	请获取日志后单击 Link 联系技术支持。
ACL_ERROR_RT_AICORE_TRAP_READ_OVERFLOW = 507042;	aicore trap读越界异常。	请获取日志后单击 Link 联系技术支持。
ACL_ERROR_RT_AICORE_TRAP_WRITE_OVERFLOW = 507043;	aicore trap写越界异常。	请获取日志后单击 Link 联系技术支持。
ACL_ERROR_RT_VECTOR_CORE_TRAP_READ_OVERFLOW = 507044;	vector core trap读越界异常。	请获取日志后单击 Link 联系技术支持。
ACL_ERROR_RT_VECTOR_CORE_TRAP_WRITE_OVERFLOW = 507045;	vector core trap写越界异常。	请获取日志后单击 Link 联系技术支持。

返回码	含义	可能原因及解决方法
ACL_ERROR_RT_STREAM_SYNC_TIMEOUT = 507046;	在指定的超时等待事件中，指定的stream中所有任务还没有执行完成。	请获取日志后单击 Link 联系技术支持。
ACL_ERROR_RT_EVENT_SYNC_TIMEOUT = 507047;	在指定的Event同步等待中，超过指定时间，该Event还有没有执行完。	请获取日志后单击 Link 联系技术支持。
ACL_ERROR_RT_FFTS_PLUS_TIMEOUT = 507048;	内部任务执行超时。	请获取日志后单击 Link 联系技术支持。
ACL_ERROR_RT_FFTS_PLUS_EXCEPTION = 507049;	内部任务执行异常。	请获取日志后单击 Link 联系技术支持。
ACL_ERROR_RT_FFTS_PLUS_TRAP_EXCEPTION = 507050;	内部任务trap异常。	请获取日志后单击 Link 联系技术支持。
static const int32_t ACL_ERROR_RT_SEND_MSG = 507051;	数据入队过程中消息发送失败。	请获取日志后单击 Link 联系技术支持。
static const int32_t ACL_ERROR_RT_COPY_DATA = 507052;	数据入队过程中内存拷贝失败。	请获取日志后单击 Link 联系技术支持。
ACL_ERROR_RT_DRV_INTERNAL_ERROR = 507899	Driver模块内部错误。	请获取日志后单击 Link 联系技术支持。
ACL_ERROR_RT_AICPU_INTERNAL_ERROR = 507900;	AI CPU模块内部错误。	请获取日志后单击 Link 联系技术支持。
ACL_ERROR_RT_SOCKET_CLOSE = 507901;	内部HDC（Host Device Communication）会话链接断开。	请获取日志后单击 Link 联系技术支持。

表 2-73 内部 GE 的返回码列表

返回码	含义	可能原因及解决方法
ACL_ERROR_GE_PARAM_INVALID = 145000	参数校验失败。	请检查接口的入参值是否正确。

返回码	含义	可能原因及解决方法
ACL_ERROR_GE_EXEC_NOT_INIT = 145001	未初始化。	- 请检查是否已调用acl.init接口进行初始化，请确保已调用acl.init接口，且在其它pyACL接口之前调用。 - 请检查是否已调用对应功能的初始化接口，例如初始化Dump的acl.mdl.init_dump接口、初始化Profiling的acl.prof.init接口。
ACL_ERROR_GE_EXEC_MODEL_PATH_INVALID = 145002	无效的模型路径。	请检查模型路径是否正确。
ACL_ERROR_GE_EXEC_MODEL_ID_INVALID = 145003	无效的模型ID。	请检查模型ID是否正确、模型是否正确加载。
ACL_ERROR_GE_EXEC_MODEL_DATA_SIZE_INVALID = 145006	无效的模型大小。	无效的模型大小。
ACL_ERROR_GE_EXEC_MODEL_ADDR_INVALID = 145007	无效的模型内存地址。	请检查模型地址是否有效。
ACL_ERROR_GE_EXEC_MODEL_QUEUE_ID_INVALID = 145008	无效的队列ID。	无效的队列 ID。
ACL_ERROR_GE_EXEC_LOAD_MODEL_REPEATED = 145009	重复初始化或重复加载。	请检查是否调用对应的接口重复初始化或重复加载。
ACL_ERROR_GE_DYNAMIC_INPUT_ADDR_INVALID = 145011	无效的动态分档输入地址。	请检查动态分档输入地址。
ACL_ERROR_GE_DYNAMIC_INPUT_LENGTH_INVALID = 145012	无效的动态分档输入长度。	请检查动态分档输入长度。
ACL_ERROR_GE_DYNAMIC_BATCH_SIZE_INVALID = 145013	无效的动态分档Batch大小。	请检查动态分档Batch大小。
ACL_ERROR_GE_AIPP_BATCH_EMPTY = 145014	无效的AIPP Batch数。	请检查AIPP Batch数是否正确。

返回码	含义	可能原因及解决方法
ACL_ERROR_GE_AIPP_NOT_EXIST = 145015	AIPP配置不存在。	请检查AIPP是否配置。
ACL_ERROR_GE_AIPP_MODE_INVALID = 145016	无效的AIPP模式。	请检查模型转换时配置的AIPP模式是否正确。
ACL_ERROR_GE_OP_TASK_TYPE_INVALID = 145017	无效的任务类型。	请检查算子类型是否正确。
ACL_ERROR_GE_OP_KERNEL_TYPE_INVALID = 145018	无效的算子类型。	请检查算子类型是否正确。
ACL_ERROR_GE_PLGMR_PATH_INVALID = 145019	无效的so文件，包括so文件的路径层级太深、so文件被误删除等情况。	请检查运行应用前配置的环境变量LD_LIBRARY_PATH是否正确，详细描述请参见编译运行处的操作指导。
ACL_ERROR_GE_FORMAT_INVALID = 145020	无效的format。	请检查Tensor数据的format是否有效。
ACL_ERROR_GE_SHAPE_INVALID = 145021	无效的shape。	请检查Tensor数据的shape是否有效。
ACL_ERROR_GE_DATATYPE_INVALID = 145022	无效的数据类型。	请检查Tensor数据的数据类型是否有效。
ACL_ERROR_GE_MEMORY_ALLOCATION = 245000	申请内存失败。	请检查硬件环境上的内存剩余情况。
ACL_ERROR_GE_MEMORY_OPERATE_FAILED = 245001;	内存初始化、内存复制操作失败。	请检查内存地址是否正确、硬件环境上的内存是否足够等。
ACL_ERROR_GE_DEVICE_MEMORY_ALLOCATION_FAILED = 245002;	申请Device内存失败。	Device内存已用完，无法继续申请，请释放部分Device内存，再重新尝试。
ACL_ERROR_GE_INTERNAL_ERROR = 545000	未知内部错误。	请获取日志后单击 Link 联系技术支持。
ACL_ERROR_GE_LOAD_MODEL = 545001	系统内部加载模型失败。	请获取日志后单击 Link 联系技术支持。
ACL_ERROR_GE_EXEC_LOAD_MODEL_PARTITION_FAILED = 545002	系统内部加载模型失败。	请获取日志后单击 Link 联系技术支持。
ACL_ERROR_GE_EXEC_LOAD_WEIGHT_PARTITION_FAILED = 545003	系统内部加载模型权值失败。	请获取日志后单击 Link 联系技术支持。

返回码	含义	可能原因及解决方法
ACL_ERROR_GE_EXEC_LOAD_TASK_PARTITION_FAILED = 545004	系统内部加载模型任务失败。	请获取日志后单击 Link 联系技术支持。
ACL_ERROR_GE_EXEC_LOAD_KERNEL_PARTITION_FAILED = 545005	系统内部加载模型算子失败。	请获取日志后单击 Link 联系技术支持。
ACL_ERROR_GE_EXEC_RELEASE_MODEL_DATA = 545006	系统内释放模型空间失败。	请获取日志后单击 Link 联系技术支持。
ACL_ERROR_GE_COMMAND_HANDLE = 545007	系统内命令操作失败。	请获取日志后单击 Link 联系技术支持。
ACL_ERROR_GE_GET_TENSOR_INFO = 545008	系统内获取张量数据失败。	请获取日志后单击 Link 联系技术支持。
ACL_ERROR_GE_UNLOAD_MODEL = 545009	系统内卸载模型空间失败。	请获取日志后单击 Link 联系技术支持。
ACL_ERROR_GE_MODEL_EXECUTE_TIMEOUT = 545601	模型执行超时。	请获取日志后单击 Link 联系技术支持。

2.16.2 aclDataType

数据类型	说明
ACL_DT_UNDEFINED = -1	未知数据类型，默认值
ACL_FLOAT = 0	float
ACL_FLOAT16 = 1	float16
ACL_INT8 = 2	int8
ACL_INT32 = 3	int32
ACL_UINT8 = 4	uint8
ACL_INT16 = 6	int16
ACL_UINT16 = 7	uint16
ACL_UINT32 = 8	uint32
ACL_INT64 = 9	int64
ACL_UINT64 = 10	uint64
ACL_DOUBLE = 11	double

数据类型	说明
ACL_BOOL = 12	bool
ACL_STRING = 13	string
ACL_COMPLEX64 = 16	complex64
ACL_COMPLEX128 = 17	complex128
ACL_BF16 = 27	bf16
ACL_INT4 = 29	int4
ACL_UINT1 = 30	uint1
ACL_COMPLEX32 = 33	complex32

2.16.3 aclFloat16

数据格式	说明
aclFloat16	如果需要传递aclFloat16类型数据，即指传递uint16_t类型数据。

 说明

该数据类型仅用于数据传递，不能用于加减乘除等运算。

2.16.4 aclFormat

数据格式	说明
ACL_FORMAT_UNDEF INED = -1	未知数据类型，默认值。
ACL_FORMAT_NCHW = 0	NCHW格式。
ACL_FORMAT_NHWC = 1	NHWC格式。
ACL_FORMAT_ND = 2	表示支持任意格式，仅有Square、Tanh等这些单输入对自身处理的算子外，其它需要慎用。
ACL_FORMAT_NC1H WC0 = 3	5维数据格式。其中，C0与微架构强相关，该值等于cube单元的size，例如16；C1是将C维度按照C0切分：C1=C/C0，若结果不整除，最后一份数据需要填充到C0。
ACL_FORMAT_FRACTA L_Z = 4	卷积的权重的格式。

数据格式	说明
ACL_FORMAT_NC1H WC0_C04 = 12	5维数据格式。其中，C0固定为4，C1是将C维度按照C0切分： $C1=C/C0$ ，若结果不整除，最后一份数据需要padding到C0。当前版本不支持。
ACL_FORMAT_HWCN = 16	HWCN格式。
ACL_FORMAT_NDHW C = 27	NDHWC格式。对于3维图像就需要使用带D（Depth）维度的格式。
ACL_FORMAT_FRACTAL_NZ = 29	内部格式，用户目前无需使用。
ACL_FORMAT_NCDH W = 30	NCDHW格式。对于3维图像就需要使用带D（Depth）维度的格式。
ACL_FORMAT_NDC1H WC0 = 32	6维数据格式。相比于NC1HWC0，仅多了D（Depth）维度。
ACL_FRACTAL_Z_3D = 33	3D卷积权重格式，例如Conv3D/MaxPool3D/AvgPool3D这些算子均需以这种格式来表达。
ACL_FORMAT_NC = 35	2维数据格式。
ACL_FORMAT_NCL = 47	3维数据格式。

 说明

各维度的含义如下：N（Batch）表示批量大小、H（Height）表示特征图高度、W（Width）表示特征图宽度、C（Channels）表示特征图通道、D（Depth）表示特征图深度。

2.16.5 aclrtMemMallocPolicy

此处支持单个枚举项，也支持多个枚举项位或：

- 配置单个枚举项：
若配置以下其中一项，系统内部会根据硬件支持情况选择从高带宽或低带宽物理内存申请内存：
 - ACL_MEM_MALLOC_HUGE_FIRST
 - ACL_MEM_MALLOC_HUGE_ONLY
 - ACL_MEM_MALLOC_NORMAL_ONLY
 - ACL_MEM_MALLOC_HUGE_FIRST_P2P
 - ACL_MEM_MALLOC_HUGE_ONLY_P2P
 - ACL_MEM_MALLOC_NORMAL_ONLY_P2P若配置ACL_MEM_TYPE_LOW_BAND_WIDTH或ACL_MEM_TYPE_HIGH_BAND_WIDTH，则系统内部会默认采取ACL_MEM_MALLOC_HUGE_FIRST，优先申请大页。

● 配置多个枚举项位或：

支持以下两组各选一项组合配置（例如：ACL_MEM_MALLOC_HUGE_FIRST | ACL_MEM_TYPE_HIGH_BAND_WIDTH）

- ACL_MEM_MALLOC_HUGE_FIRST、ACL_MEM_MALLOC_HUGE_ONLY、ACL_MEM_MALLOC_NORMAL_ONLY
- ACL_MEM_TYPE_LOW_BAND_WIDTH、ACL_MEM_TYPE_HIGH_BAND_WIDTH

枚举项	说明
ACL_MEM_MALLOC_HUGE_FIRST = 0	当申请的内存小于等于1M时，即使使用该内存分配规则，也是申请普通页的内存。当申请的内存大于1M时，优先申请大页内存，如果大页内存不够，则使用普通页的内存。
ACL_MEM_MALLOC_HUGE_ONLY	仅申请大页，如果大页内存不够，则返回错误。
ACL_MEM_MALLOC_NORMAL_ONLY	仅申请普通页，如果普通页内存不够，则返回错误。
ACL_MEM_MALLOC_HUGE_FIRST_P2P	仅Device之间内存复制场景下申请内存时使用该选项，表示优先申请大页内存，如果大页内存不够，则使用普通页的内存。 Atlas 200/300/500 推理产品 ，当前版本不支持该选项。 Atlas 200I/500 A2推理产品 ，当前版本不支持该选项。
ACL_MEM_MALLOC_HUGE_ONLY_P2P	仅Device之间内存复制场景下申请内存时使用该选项，仅申请大页内存，如果大页内存不够，则返回错误。 Atlas 200/300/500 推理产品 ，当前版本不支持该选项。 Atlas 200I/500 A2推理产品 ，当前版本不支持该选项。
ACL_MEM_MALLOC_NORMAL_ONLY_P2P	仅Device之间内存复制场景下申请内存时使用该选项，仅申请普通页的内存。 Atlas 200/300/500 推理产品 ，当前版本不支持该选项。 Atlas 200I/500 A2推理产品 ，当前版本不支持该选项。
ACL_MEM_TYPE_LOW_BAND_WIDTH	从带宽高的物理内存上申请内存。 Atlas 200I/500 A2推理产品 ，设置该选项无效，系统默认会根据硬件支持的内存类型选择。
ACL_MEM_TYPE_HIGH_BAND_WIDTH	从带宽低的物理内存上申请内存。 Atlas 200I/500 A2推理产品 ，设置该选项无效，系统默认会根据硬件支持的内存类型选择。

2.16.6 acldvppPixelFormat

像素格式	说明
PIXEL_FORMAT_YUV_400 = 0	YUV400 8bit
PIXEL_FORMAT_YUV_SEMIPLA_NAR_420 = 1	YUV420SP NV12 8bit
PIXEL_FORMAT_YVU_SEMIPLA_NAR_420 = 2	YUV420SP NV21 8bit
PIXEL_FORMAT_YUV_SEMIPLA_NAR_422 = 3	YUV422SP NV12 8bit
PIXEL_FORMAT_YVU_SEMIPLA_NAR_422 = 4	YUV422SP NV21 8bit
PIXEL_FORMAT_YUV_SEMIPLA_NAR_444 = 5	YUV444SP NV12 8bit
PIXEL_FORMAT_YVU_SEMIPLA_NAR_444 = 6	YUV444SP NV21 8bit
PIXEL_FORMAT_YUYV_PACKED_422 = 7	YUV422P YUYV 8bit
PIXEL_FORMAT_UYVY_PACKED_422 = 8	YUV422P UYVY 8bit
PIXEL_FORMAT_YVYU_PACKED_422 = 9	YUV422P VYU 8bit
PIXEL_FORMAT_VYUY_PACKED_422 = 10	YUV422P VYUY 8bit
PIXEL_FORMAT_YUV_PACKED_444 = 11	YUV444P 8bit
PIXEL_FORMAT_RGB_888 = 12	RGB888
PIXEL_FORMAT_BGR_888 = 13	BGR888
PIXEL_FORMAT_ARGB_8888 = 14	ARGB8888
PIXEL_FORMAT_ABGR_8888 = 15	ABGR8888
PIXEL_FORMAT_RGBA_8888 = 16	RGBA8888
PIXEL_FORMAT_BGRA_8888 = 17	BGRA8888
PIXEL_FORMAT_YUV_SEMI_PLANNER_420_10BIT = 18	YUV420SP 10bit

像素格式	说明
PIXEL_FORMAT_YVU_SEMI_PLANNER_420_10BIT = 19	YVU420sp 10bit
PIXEL_FORMAT_YVU_PLANAR_420 = 20	YUV420P 8bit
PIXEL_FORMAT_YVU_PLANAR_422 = 21	YUV422P 8bit
PIXEL_FORMAT_YVU_PLANAR_444 = 22	YUV444P 8bit
PIXEL_FORMAT_RGB_444 = 23	RGB444 R:4bit G:4bit B:4bit, 当前不支持该格式
PIXEL_FORMAT_BGR_444 = 24	BGR444 R:4bit G:4bit B:4bit, 当前不支持该格式
PIXEL_FORMAT_ARGB_4444 = 25	ARGB4444 A:4bit R:4bit G:4bit B:4bit, 当前不支持该格式
PIXEL_FORMAT_ABGR_4444 = 26	ABGR4444 A:4bit B:4bit G:4bit R:4bit, 当前不支持该格式
PIXEL_FORMAT_RGBA_4444 = 27	RGBA4444 R:4bit G:4bit B:4bit A:4bit, 当前不支持该格式
PIXEL_FORMAT_BGRA_4444 = 28	BGRA4444 B:4bit G:4bit R:4bit A:4bit, 当前不支持该格式
PIXEL_FORMAT_RGB_555 = 29	RGB555 R:5bit G:5bit B:5bit, 当前不支持该格式
PIXEL_FORMAT_BGR_555 = 30	BGR555 B:5bit G:5bit R:5bit, 当前不支持该格式
PIXEL_FORMAT_RGB_565 = 31	RGB565 R:5bit G:6bit B:5bit, 当前不支持该格式
PIXEL_FORMAT_BGR_565 = 32	BGR565 B:5bit G:6bit R:5bit, 当前不支持该格式
PIXEL_FORMAT_ARGB_1555 = 33	ARGB1555 A:1bit R:5bit G:6bit B:5bit, 当前不支持该格式
PIXEL_FORMAT_ABGR_1555 = 34	ABGR1555 A:1bit B:5bit G:6bit R:5bit, 当前不支持该格式
PIXEL_FORMAT_RGBA_1555 = 35	RGBA1555 A:1bit B:5bit G:6bit R:5bit, 当前不支持该格式
PIXEL_FORMAT_BGRA_1555 = 36	BGRA1555 A:1bit B:5bit G:6bit R:5bit, 当前不支持该格式
PIXEL_FORMAT_ARGB_8565 = 37	ARGB8565 A:8bit R:5bit G:6bit B:5bit, 当前不支持该格式
PIXEL_FORMAT_ABGR_8565 = 38	ABGR8565 A:8bit B:5bit G:6bit R:5bit, 当前不支持该格式
PIXEL_FORMAT_RGBA_8565 = 39	RGBA8565 A:8bit R:5bit G:6bit B:5bit, 当前不支持该格式

像素格式	说明
PIXEL_FORMAT_BGRA_8565 = 40	BGRA8565 A:8bit B:5bit G:6bit R:5bit, 当前不支持该格式
PIXEL_FORMAT_RGB_BAYER_8BPP = 50	RGB Bayer 8bit, Bayer图像, 当前不支持该格式
PIXEL_FORMAT_RGB_BAYER_10BPP = 51	RGB Bayer 10bit, Bayer图像, 当前不支持该格式
PIXEL_FORMAT_RGB_BAYER_12BPP = 52	RGB Bayer 12bit, Bayer图像, 当前不支持该格式
PIXEL_FORMAT_RGB_BAYER_14BPP = 53	RGB Bayer 14bit, Bayer图像, 当前不支持该格式
PIXEL_FORMAT_RGB_BAYER_16BPP = 54	RGB Bayer 16bit, Bayer图像, 当前不支持该格式
PIXEL_FORMAT_BGR_888_PLANAR = 70	RGB888 Planar, 当前不支持该格式
PIXEL_FORMAT_HSV_888_PACKAGE = 71	HSV Package, HSV图像package格式, 当前不支持该格式
PIXEL_FORMAT_HSV_888_PLANAR = 72	HSV Planar, HSV图像Planar格式, 当前不支持该格式
PIXEL_FORMAT_LAB_888_PACKAGE = 73	LAB Package, LAB图像package格式, 当前不支持该格式
PIXEL_FORMAT_LAB_888_PLANAR = 74	LAB Planar, LAB图像Planar格式, 当前不支持该格式
PIXEL_FORMAT_S8C1 = 75	Signed 8bit for 1pixel 1Channel, 每个像素用1个8bit有符号数据表示的单通道图像, 当前不支持该格式
PIXEL_FORMAT_S8C2_PACKAGE = 76	Signed 8bit for 1pixel 2Channel Package, 每个像素用2个8bit有符号数表示的双通道图像Package格式, 当前不支持该格式
PIXEL_FORMAT_S8C2_PLANAR = 77	Signed 8bit for 1pixel 2Channel Planar, 每个像素用2个8bit有符号数据表的双通道图像Planar格式, 当前不支持该格式
PIXEL_FORMAT_S16C1 = 78	Signed 16bit 1pixel 1Channel, 每个像素用1个16bit有符号数据表示的单通道图像, 当前不支持该格式
PIXEL_FORMAT_U8C1 = 79	Unsigned 8bit 1pixel 1Channel, 每个像素用1个8bit无符号数据表示的单通道图像, 当前不支持该格式

像素格式	说明
PIXEL_FORMAT_U16C1 = 80	Unsigned 16bit 1pixel 1Channel，每个像素用1个16bit无符号数据表示的单通道图像，当前不支持该格式
PIXEL_FORMAT_S32C1 = 81	Signed 32bit 1pixel 1Channel，每个像素用1个32bit有符号数据表示的单通道图像，当前不支持该格式
PIXEL_FORMAT_U32C1 = 82	Unsigned 32bit 1pixel 1Channel，每个像素用1个32bit无符号数据表示的单通道图像，当前不支持该格式
PIXEL_FORMAT_U64C1 = 83	Unsigned 64bit 1pixel 1Channel，每个像素用1个64bit无符号数据表示的单通道图像，当前不支持该格式
PIXEL_FORMAT_S64C1 = 84	Signed 64bit 1pixel 1Channel，每个像素用1个64bit有符号数据表示的单通道图像，当前不支持该格式
PIXEL_FORMAT_YUV_SEMIPLANAR_440 = 1000	YUV440SP 8bit
PIXEL_FORMAT_YVU_SEMIPLANAR_440 = 1001	YVU440SP 8bit
PIXEL_FORMAT_FLOAT32 = 1002	Float 32bit for 1pixel，每个像素用1个float32数据表示，当前不支持该格式
PIXEL_FORMAT_BUTT = 1003	-
PIXEL_FORMAT_UNKNOWN = 10000	-

2.16.7 acldvppStreamFormat

H265_MAIN_LEVEL = 0
H264_BASELINE_LEVEL = 1
H264_MAIN_LEVEL = 2
H264_HIGH_LEVEL = 3

2.16.8 acldvppJpegFormat

ACL_JPEG_CSS_444 = 0
ACL_JPEG_CSS_422 = 1
ACL_JPEG_CSS_420 = 2
ACL_JPEG_CSS_GRAY = 3
ACL_JPEG_CSS_440 = 4
ACL_JPEG_CSS_411 = 5
ACL_JPEG_CSS_UNKNOWN = 1000

2.16.9 acldvppCscMatrix

Atlas 200/300/500 推理产品，不支持该值。

Atlas 训练系列产品，不支持该值。

数据格式	说明
ACL_DVPP_CSC_MATRIX_BT601_WIDE = 0	基于BT601 wide标准的色域转换矩阵，默认值。
ACL_DVPP_CSC_MATRIX_BT601_NARROW = 1	基于BT601 narrow标准的色域转换矩阵。
ACL_DVPP_CSC_MATRIX_BT709_WIDE = 2	基于BT709 wide标准的色域转换矩阵。
ACL_DVPP_CSC_MATRIX_BT709_NARROW = 3	基于BT709 narrow标准的色域转换矩阵。
ACL_DVPP_CSC_MATRIX_BT2020_WIDE = 4	基于BT2020 wide标准的色域转换矩阵。
ACL_DVPP_CSC_MATRIX_BT2020_NARROW = 5	基于BT2020 narrow标准的色域转换矩阵。

基于以上各标准的色域转换矩阵，其各分量的转换公式如下：

```
# YUV转RGB:
# | R | | matrix_r0c0 matrix_r0c1 matrix_r0c2 | | Y - input_bias_0 |
# | G | = | matrix_r1c0 matrix_r1c1 matrix_r1c2 | * | U - input_bias_1 |
# | B | | matrix_r2c0 matrix_r2c1 matrix_r2c2 | | V - input_bias_2 |
# BGR转YUV:
# | Y | | output_bias_0 | | matrix_r0c0 matrix_r0c1 matrix_r0c2 | | R |
# | U | = | output_bias_1 | + | matrix_r1c0 matrix_r1c1 matrix_r1c2 | * | G |
# | V | | output_bias_2 | | matrix_r2c0 matrix_r2c1 matrix_r2c2 | | B |
```

基于不同标准的色域转换矩阵，各参数值如下表所示。

成员名称	描述
ACL_DVPP_CSC_MATRIX_BT601_WIDE	基于BT601 wide标准的色域转换矩阵。默认为ACL_DVPP_CSC_MATRIX_BT601_WIDE。 基于该标准的色域转换矩阵，各参数值如下： # YUV转RGB: # R 1.000 0.000 1.402 Y - 0 # G = 1.000 -0.344 -0.714 * U - 128 # B 1.000 1.772 0.000 V - 128 # RGB转YUV: # Y -0.5 0.299 0.587 0.114 R # U = 127.5 + -0.168 -0.331 0.500 * G # V 127.5 0.500 -0.419 -0.081 B
ACL_DVPP_CSC_MATRIX_BT601_NARROW	基于BT601 narrow标准的色域转换矩阵。 基于该标准的色域转换矩阵，各参数值如下： # YUV转RGB: # R 1.16438 0.00000 1.59602 Y - 16 # G = 1.16438 -0.39176 -0.81297 * U - 128 # B 1.16438 2.01723 0.00000 V - 128 # RGB转YUV: # Y 16 0.25679 0.51564 0.10014 R # U = 128 + -0.14491 -0.29099 0.43922 * G # V 128 0.42941 -0.36779 -0.07143 B

成员名称	描述
ACL_DVPP_CSC_MATR IX_BT709_WIDE	基于BT709 wide标准的色域转换矩阵。 基于该标准的色域转换矩阵，各参数值如下： # YUV转RGB： # R 1.00000 0.00000 1.57480 Y - 0 # G = 1.00000 -0.18732 -0.46812 * U - 128 # B 1.00000 1.85560 0.00000 V - 128 # RGB转YUV： # Y 0 0.21260 0.71520 0.07220 R # U = 128 + -0.11457 -0.38543 0.50000 * G # V 128 0.50000 -0.45415 -0.04585 B
ACL_DVPP_CSC_MATR IX_BT709_NARROW	基于BT709 narrow标准的色域转换矩阵。 基于该标准的色域转换矩阵，各参数值如下： # YUV转RGB： # R 1.16438 0.00000 1.79274 Y - 16 # G = 1.16438 -0.21325 -0.53291 * U - 128 # B 1.16438 2.11240 0.00000 V - 128 # RGB转YUV： # Y 16 0.18259 0.62825 0.06342 R # U = 128 + -0.09840 -0.33857 0.43922 * G # V 128 0.42941 -0.39894 -0.04027 B
ACL_DVPP_CSC_MATR IX_BT2020_WIDE	基于BT2020 wide标准的色域转换矩阵。 基于该标准的色域转换矩阵，各参数值如下： # YUV转RGB： # R 1.00000 0.00000 1.47460 Y - 0 # G = 1.00000 -0.16455 -0.57135 * U - 128 # B 1.00000 1.88140 0.00000 V - 128 # RGB转YUV： # Y 0 0.26270 0.67800 0.05930 R # U = 128 + -0.13963 -0.36037 0.50000 * G # V 128 0.50000 -0.45979 -0.04021 B
ACL_DVPP_CSC_MATR IX_BT2020_NARROW	基于BT2020 narrow标准的色域转换矩阵。 基于该标准的色域转换矩阵，各参数值如下： # YUV转RGB： # R 1.16438 0.00000 1.67868 Y - 16 # G = 1.16438 -0.18733 -0.65042 * U - 128 # B 1.16438 2.14177 0.00000 V - 128 # RGB转YUV： # Y 16 0.22564 0.59558 0.05209 R # U = 128 + -0.11992 -0.31656 0.43922 * G # V 128 0.42941 -0.40389 -0.03533 B

2.16.10 aclrtEventStatus

数据格式	说明
ACL_EVENT_STATUS_COMPLE TE = 0	完成
ACL_EVENT_STATUS_NOT_REA DY = 1	未完成
ACL_EVENT_STATUS_RESERVE D = 2	预留

2.16.11 aclrtRunMode

数据格式	说明
ACL_DEVICE = 0	昇腾AI软件栈运行在Device的Control CPU或板端环境上。
ACL_HOST = 1	昇腾AI软件栈运行在Host侧。

2.16.12 aclTransType

转置方式	说明
ACL_TRANS_N = 0	不转置，默认为不转置。
ACL_TRANS_T = 1	转置。
ACL_TRANS_NZ = 2	内部格式，能够提供更好的接口性能，建议使用场景为：某个输入矩阵作为底库，执行一次转NZ格式的操作后，多次使用NZ格式的算子，提升性能。
ACL_TRANS_NZ_T = 3	内部格式转置，当前不支持。

2.16.13 aclComputeType

计算精度类型	说明
ACL_COMPUTE_HIGH_PRECISION = 0	高精度。
ACL_COMPUTE_LOW_PRECISION = 1	当前不支持。

2.16.14 aclfvSearchType

Atlas 200/300/500 推理产品，不支持该枚举值。

Atlas 200I/500 A2推理产品，不支持该枚举值。

Atlas 训练系列产品，不支持该枚举值。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该枚举值。

特征向量搜索方式	说明
SEARCH_1_N = 0	1 : N operation type

特征向量搜索方式	说明
SEARCH_N_M = 1	N : M operation type

2.16.15 aclAippInputFormat

Aipp输入图像格式	说明
ACL_YUV420SP_U8 = 1	静态AIPP和动态AIPP都支持。
ACL_XRGB8888_U8 = 2	静态AIPP和动态AIPP都支持
ACL_RGB888_U8 = 3	静态AIPP和动态AIPP都支持。
ACL_YUV400_U8 = 4	静态AIPP和动态AIPP都支持
ACL_NC1HWC0DI_FP16 = 5	暂不支持
ACL_NC1HWC0DI_S8 = 6	暂不支持
ACL_ARGB8888_U8 = 7	暂不支持。
ACL_YUYV_U8 = 8	暂不支持。
ACL_YUV422SP_U8 = 9	暂不支持。
ACL_AYUV444_U8 = 10	暂不支持。
ACL_RAW10 = 11	暂不支持。
ACL_RAW12 = 12	暂不支持。
ACL_RAW16 = 13	暂不支持。
ACL_RAW24 = 14	暂不支持。
ACL_AIPP_RESERVED = 0xffff	暂不支持。

2.16.16 aclAippInfo

静态Aipp配置信息	含义
inputFormat	AIPP输入图像格式（aclAippInputFormat类型数据）。
srcImageSizeW	原始图片的宽，对于YUV420SP_U8类型的图像，要求取值是偶数。 取值范围： [1,4096]
srcImageSizeH	原始图片的高，对于YUV420SP_U8类型的图像，要求取值是偶数。 取值范围： [1,4096]

静态Aipp配置信息	含义
cropSwitch	是否对图片执行抠图操作，取值范围： 0：不执行抠图操作，设置为0时，则设置cropStartPosW、cropStartPosH、cropSizeW、cropSizeH参数无效。 1：执行抠图操作。
loadStartPosW	抠图时，坐标点起始位置在图中横向的坐标。 对于YUV420SP_U8格式的图像，参数取值要求是偶数。 取值范围：[0, 4095]
loadStartPosH	抠图时，坐标点起始位置在图中纵向的坐标。 对于YUV420SP_U8格式的图像，参数取值要求是偶数。 取值范围：[0, 4095]
cropSizeW	抠图区域的宽度。 对于YUV420SP_U8格式的图像，参数取值要求是偶数。 取值范围：[1, 4096]
cropSizeH	抠图区域的高度。 对于YUV420SP_U8格式的图像，参数取值要求是偶数。 取值范围：[1, 4096]
resizeSwitch	AIPP处理图片时是否支持缩放，保留字段，暂不支持该功能。
resizeOutputW	缩放后图像的宽度。
resizeOutputH	缩放后图像的高度。
paddingSwitch	AIPP处理图片时补边使能开关。
leftPaddingSize	W左填充。
rightPaddingSize	W右填充。
topPaddingSize	H上填充。
bottomPaddingSize	H下填充。
cscSwitch	色域转换开关，静态AIPP配置。
rbuvSwapSwitch	色域转换前，R通道与B通道交换开关/U通道与V通道交换开关。
axSwapSwitch	色域转换前，RGBA->ARGB, YUVA->AYUV交换开关。
singleLineMode	单行处理模式（只处理抠图后的第一行）开关。
matrixR0C0	3 * 3 CSC矩阵元素。
matrixR0C1	3 * 3 CSC矩阵元素。
matrixR0C2	3 * 3 CSC矩阵元素。

静态Aipp配置信息	含义
matrixR1C0	3 * 3 CSC矩阵元素。
matrixR1C1	3 * 3 CSC矩阵元素。
matrixR1C2	3 * 3 CSC矩阵元素。
matrixR2C0	3 * 3 CSC矩阵元素。
matrixR2C1	3 * 3 CSC矩阵元素。
matrixR2C2	3 * 3 CSC矩阵元素。
outputBias0	RGB转YUV时的输出偏移。
outputBias1	RGB转YUV时的输出偏移。
outputBias2	RGB转YUV时的输出偏移。
inputBias0	YUV转RGB时的输入偏移。
inputBias1	YUV转RGB时的输入偏移。
inputBias2	YUV转RGB时的输入偏移。
meanChn0	每个通道的均值。
meanChn1	每个通道的均值。
meanChn2	每个通道的均值。
meanChn3	每个通道的均值。
minChn0	每个通道的最小值。
minChn1	每个通道的最小值。
minChn2	每个通道的最小值。
minChn3	每个通道的最小值。
varReciChn0	每个通道的方差。
varReciChn1	每个通道的方差。
varReciChn2	每个通道的方差。
varReciChn3	每个通道的方差。
srcFormat	模型转换前，原始模型的输入format（aclFormat类型数据）。
srcDatatype	模型转换前，原始模型的输入datatype（aclDataType类型数据）。
srcDimNum	模型转换前，原始模型输入dims。
shapeCount	动态Shape（动态Batch或动态分辨率）场景下的档位数。
outDims	输出数据维度（aclAippDims类型）。

静态Aipp配置信息	含义
aippExtend	预留参数。

2.16.17 aclAippDims

aipp维度信息	含义
srcDims	模型输入输出维度信息（aclmdlIODims类型）数据，模型转换前的dims，如[1, 3, 150, 150]。
srcSize	模型转换前输入数据的大小。
aippOutdims	内部数据格式的dims信息。
aippOutSize	经过AIPP处理后的输出数据的大小。

2.16.18 aclmdlIODims

模型输入输出维度信息	含义
name	Tensor名称。
dimCount	Shape中的维度个数。
dims	维度信息。

2.16.19 aclrtGroupAttr

Atlas 200/300/500 推理产品，不支持该枚举值。

Atlas 200I/500 A2推理产品，不支持该枚举值。

Atlas 训练系列产品，不支该枚举值。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该枚举值。

数据格式	说明
ACL_GROUP_AICORE_INT= 0	指定Group对应的aicore个数，属性值的数据类型为整型。
ACL_GROUP_AIV_INT= 1	指定Group对应的vector core个数，属性值的数据类型为整型。
ACL_GROUP_AIC_INT= 2	指定Group对应的aicpu线程数，属性值的数据类型为整型。

数据格式	说明
ACL_GROUP_SDMANUM_INT = 3	内存异步拷贝的通道数，属性值的数据类型为整型。
ACL_GROUP_SDMANUM_INT = 4	指定Group下可以被同时调度执行的Stream个数，小于或等于32，当前系统级最大一共可以同时调度32个Stream。属性值的数据类型为整型。
ACL_GROUP_GROUPID_INT = 5	指定Group的ID。属性值的数据类型为整型。

2.16.20 aclprofAicoreMetrics

AI Core性能指标采集项	含义
ACL_AICORE_ARITHMETIC_UTILIZATION = 0	各种计算类指标占比统计，包括采集项mac_fp16_ratio、mac_int8_ratio、vec_fp32_ratio、vec_fp16_ratio、vec_int32_ratio、vec_misc_ratio。
ACL_AICORE_PIPE_UTILIZATION = 1	计算单元和搬运单元耗时占比，包括采集项vec_ratio、mac_ratio、scalar_ratio、mte1_ratio、mte2_ratio、mte3_ratio、icache_miss_rate、fixpipe_ratio（ Atlas A2训练系列产品/Atlas 800I A2推理产品 ）。
ACL_AICORE_MEMORY_BANDWIDTH = 2	外部内存读写类指令占比，包括采集项ub_read_bw、ub_write_bw、l1_read_bw、l1_write_bw、l2_read_bw（ Atlas 200/300/500 推理产品 ）、l2_write_bw（ Atlas 200/300/500 推理产品 ）、main_mem_read_bw、main_mem_write_bw。
ACL_AICORE_L0B_AND_WIDTH = 3	内部内存读写类指令占比，包括采集项scalar_ld_ratio、scalar_st_ratio、l0a_read_bw、l0a_write_bw、l0b_read_bw、l0b_write_bw、l0c_read_bw、l0c_write_bw、l0c_read_bw_cube、l0c_write_bw_cube。
ACL_AICORE_RESOURCE_CONFLICT_RATIO = 4	流水线队列类指令占比，包括采集项vec_bankgroup_cflt_ratio、vec_bank_cflt_ratio、vec_resc_cflt_ratio、mte1_iq_full_ratio、mte2_iq_full_ratio、mte3_iq_full_ratio、cube_iq_full_ratio、vec_iq_full_ratio、iq_full_ratio。
ACL_AICORE_MEMORY_UB = 5	内部内存读写指令占比，包括采集项ub_read_bw_mte（ Atlas 200/300/500 推理产品 ）、ub_write_bw_mte（ Atlas 200/300/500 推理产品 ）、ub_read_bw_vector、ub_write_bw_vector、ub_read_bw_scalar、ub_write_bw_scalar。

AI Core性能指标采集项	含义
ACL_AICORE_L2_CACHE = 6	(Atlas A2训练系列产品/Atlas 800I A2推理产品) 读写cache命中次数和缺失后重新分配次数， 包括采集项ai*_write_cache_hit、ai*_write_cache_miss_allocate、ai*_r*_read_cache_hit、ai*_r*_read_cache_miss_allocate
ACL_AICORE_L2_CACHE = 6	(Atlas 200I/500 A2推理产品) 读写cache命中次数和缺失后重新分配次数， 包括采集项write_cache_hit、write_cache_miss_allocate、r*_read_cache_hit、r*_read_cache_miss_allocate。
ACL_AICORE_PIPE_EXECUTE_UTILIZATION = 7	(Atlas 200I/500 A2推理产品) 计算单元和搬运单元耗时占比，包括采集项vec_exe_ratio、mac_exe_ratio、scalar_exe_ratio、mte1_exe_ratio、mte2_exe_ratio、mte3_exe_ratio、fixpipe_exe_ratio。
ACL_AICORE_NONE = 0xFF	表示不采集数据。

2.16.21 acldvppBorderType

Atlas 200/300/500 推理产品，当前版本不支持该值。

Atlas 训练系列产品，当前版本不支持该值。

数据格式	说明
BORDER_CONSTANT= 0	添加有颜色的常数值边界。
BORDER_REPLICATE = 1	重复最后一个元素。举例: aaaaaa abcdefgh hhhhhhh
BORDER_REFLECT= 2	边界元素的镜像。举例： cba abcde-fgh hgf
BORDER_REFLECT_101 = 3	边界元素的镜像。举例： cba abcde-fgh gf

2.16.22 acldvppJpegdPrecisionMode

数据格式	说明
ACL_YUVOUT_ALIGN_DOWN = 0	ACL_YUVOUT_ALIGN_DOWN： - 解码后的输出图片格式为YUV420SP时，若源图的宽、高为奇数时，将输出图片的宽、高向下2对齐。 - 解码后的输出图片格式为YUV422SP时，若源图的宽为奇数时，将输出图片的宽向下2对齐。 - 解码后的输出图片格式为YUV440SP时，若源图的高为奇数时，将输出图片高向下2对齐。
ACL_YUVOUT_ALIGN_UP = 1	ACL_YUVOUT_ALIGN_UP： - 解码后的输出图片格式为YUV420SP时，若源图的宽、高为奇数时，将输出图片的宽、高向上2对齐。 - 解码后的输出图片格式为YUV422SP时，若源图的宽为奇数时，将输出图片的宽向上2对齐。 - 解码后的输出图片格式为YUV440SP时，若源图的高为奇数时，将输出图片高向上2对齐。
ACL_YUVOUT_ALIGN_DOWN_COMPAT = 2	ACL_YUVOUT_ALIGN_DOWN_COMPAT：默认值 兼容旧版本，解码后的输出图片格式为YUV420SP/YUV422SP/YUV440SP时，若源图的宽、高为奇数时，将输出图片的宽、高向下2对齐。

2.16.23 aclmdlInputAippType

模型输入的AIPP类型	含义
ACL_DATA_WITHOUT_AIPP = 0	该输入无AIPP信息。
ACL_DATA_WITH_STATIC_AIPP = 1	该输入带静态AIPP信息。
ACL_DATA_WITH_DYNAMIC_AIPP = 2	该输入有关联的动态AIPP输入。
ACL_DYNAMIC_AIPP_NODE = 3	该输入本身就是动态AIPP输入，用于配置动态AIPP参数。

2.16.24 acldvppChannelDescParamType

数据格式	说明
ACL_DVPP_CSC_MATRIX_UINT32 = 0	色域转换矩阵属性，该属性类型对应的值为uint32_t类型，属性值请参见 aclDvppCscMatrix 。 Atlas 200/300/500 推理产品 ，不支持该选项。 Atlas 训练系列产品 ，不支持该选项。
ACL_DVPP_MODE_UINT32	通道描述信息中的通道模式，用于明确图片数据处理通道用于实现哪种功能，目前支持 VPC 、 JPEGD 、 JPEGE 、 PNGD 功能。属性值请参见 aclDvppChannelMode 。 Atlas 200/300/500 推理产品 ，不支持该选项。 Atlas 训练系列产品 ，不支持该选项。
ACL_DVPP_CHANNEL_ID_UINT64	通道ID。
ACL_DVPP_CHANNEL_HEIGHT_UINT32	通道的高度，目前仅支持在 JPEGE 功能中使用该属性，其它功能中设置该属性无效。 Atlas 推理系列产品 ，默认值和最大值为8192。 Atlas A2训练系列产品/Atlas 800I A2推理产品 ，默认值和最大值为8192。 Atlas 200I/500 A2推理产品 ，默认值为8192，最大值为16384，若输入图片高大于8192，则需要调用 acl.media.dvpp_set_channel_desc_param 接口设置通道高度。 Atlas 200/300/500 推理产品 ，不支持该选项。 Atlas 训练系列产品 ，不支持该选项。
ACL_DVPP_CHANNEL_WIDTH_UINT32	通道的宽度，目前仅支持在 JPEGE 功能中使用该属性，其它功能中设置该属性无效。 Atlas 推理系列产品 ，默认值和最大值为8192。 Atlas A2训练系列产品/Atlas 800I A2推理产品 ，默认值和最大值为8192。 Atlas 200I/500 A2推理产品 ，默认值为8192，最大值为16384，若输入图片宽大于8192，则需要调用 acl.media.dvpp_set_channel_desc_param 接口设置通道宽度。 Atlas 200/300/500 推理产品 ，不支持该选项。 Atlas 训练系列产品 ，不支持该选项。
ACL_DVPP_JPEGD_PRECISION_MODE_ENUM	JPEGD解码输出图片的宽、高对齐模式，模式选项参见 2.16.22 aclDvppJpegdPrecisionMode 。 目前仅支持在JPEGD功能中使用该属性，其它功能中设置该属性无效。 Atlas 200/300/500 推理产品 ，不支持该选项。 Atlas 训练系列产品 ，不支持该选项。

2.16.25 aclvdecChannelDescParamType

数据格式	说明
ACL_VDEC_CSC_MATRIX_UINT32 = 0	色域转换矩阵属性，属性值请参见 aclvppCscMatrix 。 Atlas 200/300/500 推理产品 ，不支持该选项。 Atlas 训练系列产品 ，不支持该选项。
ACL_VDEC_OUT_MODE_UINT32 = 1	设置是否实时出帧（即发送一帧解码一帧，无需依赖后续帧的传入）。 取值范围如下： 0：默认出帧模式，由于解码过程中存在缓存帧，无法实时输出，因此VDEC需要在收到码流中的多帧数据后，才开始输出解码结果。 1：快速出帧模式，VDEC获取码流中的一帧数据后，就开始实时输出解码结果。只支持简单参考关系的H.264/H.265标准码流（无长期参考帧，无B帧）。
ACL_VDEC_THREAD_ID_UINT64 = 2	回调线程ID。 说明 同一个进程内，在不同的Device上注册VDEC解码回调函数的线程时，不能指定同一个线程ID。 同一个Device上，多路VDEC解码可以指定同一个线程ID，但相比一个线程处理一路VDEC解码任务来说，一个线程内串行处理多路VDEC解码任务时，VDEC解码性能可能下降。
ACL_VDEC_CALLBACK_PTR = 3	解码回调函数，格式如下： <pre>def vdec_call_back_func(dvpp_stream_desc, dvpp_pic_desc, user_data): pass</pre>
ACL_VDEC_CHANNEL_ID_UINT32 = 4	通道ID。 Atlas 200/300/500 推理产品 ，该参数值的取值范围[0, 31]。 Atlas 训练系列产品 ，该参数值的取值范围[0, 31]。 Atlas 200I/500 A2推理产品 ，该参数值的取值范围[0, 255]。 Atlas A2训练系列产品/Atlas 800I A2推理产品 ，该参数值的取值范围[0, 255]。 Atlas 推理系列产品 ，该参数值的取值范围[0, 255]。
ACL_VDEC_ENCODE_TYPE_UINT32 = 5	视频编码协议。 请参见 2.16.7 aclvppStreamFormat 。
ACL_VDEC_OUT_PICTURE_FORMAT_UINT32 = 6	YUV图像存储格式，具体请参见 表2-24 如果不设置输出格式，默认使用YUV420SP NV12。

数据格式	说明
ACL_VDEC_OUT_PIC_WIDTH_UINT32 = 7	解码码流最大宽度。 Atlas 200/300/500 推理产品，如果不设置解码码流最大宽度，内部默认使用1920。 Atlas 训练系列产品，如果不设置解码码流最大宽度，内部默认使用1920。 Atlas 200I/500 A2推理产品，如果不设置解码码流最大宽度，内部默认使用4096。 Atlas A2训练系列产品/Atlas 800I A2推理产品，如果不设置解码码流最大宽度，内部默认使用4096。 Atlas 推理系列产品，如果不设置解码码流最大宽度，内部默认使用4096。
ACL_VDEC_OUT_PIC_HEIGHT_UINT32 = 8	解码码流最大高度。 Atlas 200/300/500 推理产品，如果不设置解码码流最大高度，内部默认使用1120。 Atlas 训练系列产品，如果不设置解码码流最大高度，内部默认使用1120。 Atlas 200I/500 A2推理产品，如果不设置解码码流最大高度，内部默认使用4096。 Atlas A2训练系列产品/Atlas 800I A2推理产品，如果不设置解码码流最大高度，内部默认使用4096。 Atlas 推理系列产品，如果不设置解码码流最大高度，内部默认使用4096。
ACL_VDEC_REF_FRAME_NUM_UINT32 = 9	参考帧数量，取值范围[0, 16]。 Atlas 200/300/500 推理产品，当前版本不支持用户设置该参数。 Atlas 训练系列产品，当前版本不支持用户设置该参数。 Atlas 200I/500 A2推理产品，如果不设置解码码流最大高度，内部默认使用4096。 Atlas A2训练系列产品/Atlas 800I A2推理产品，如果不设置参考帧数量，内部默认使用8。 Atlas 推理系列产品，如果不设置参考帧数量，内部默认使用8。
ACL_VDEC_BIT_DEPTH_UINT32 = 10	设置视频位宽，默认值为10-bit。 取值范围如下： 0: 8-bit 1: 10-bit (默认值) Atlas 200/300/500 推理产品，当前版本不支持用户设置该参数。 Atlas 训练系列产品，当前版本不支持用户设置该参数。

2.16.26 aclvencChannelDescParamType

Atlas 训练系列产品，不支持该枚举值。

Atlas A2训练系列产品/Atlas 800I A2推理产品，不支持该枚举值。

视频编码处理通道属性类型	说明
ACL_VENC_THREAD_ID_UINT64 = 0	回调线程ID。 说明 同一个进程内，在不同的Device上注册VENC编码回调函数的线程时，不能指定同一个线程ID。
ACL_VENC_CALLBACK_PTR = 1	回调函数。
ACL_VENC_PIXEL_FORMAT_UINT32 = 2	输入图像格式，支持如下格式： - PIXEL_FORMAT_YUV_SEMIPLANAR_420 - PIXEL_FORMAT_YVU_SEMIPLANAR_420
ACL_VENC_ENCODE_TYPE_UINT32 = 3	视频编码协议。 请参见 2.16.7 aclvppStreamFormat 。
ACL_VENC_PICTURE_WIDTH_UINT32 = 4	输入图片宽度。
ACL_VENC_PICTURE_HEIGHT_UINT32 = 5	输入图片高度。
ACL_VENC_KEYFRAME_INTERVAL_UINT32 = 6	关键帧间隔，取值范围[1, 65536]。
ACL_VENC_BUFFER_ADDR_PTR = 7	编码输出缓存指针地址。 说明 在Atlas 200/300/500 推理产品上，当前版本不支持用户设置该参数，由VENC内部管理该缓存地址。
ACL_VENC_BUFFER_SIZE_UINT32 = 8	编码输出缓存大小，单位为Byte。 说明 在Atlas 200/300/500 推理产品上，当前版本不支持用户设置该参数，内部默认使用3686400Byte。 在Atlas 推理系列产品上，如果不设置该参数，参数值默认为8M；如果设置该参数，参数值最小为5M。 在Atlas 200I/500 A2推理产品上，如果不设置该参数，参数值默认为8M；如果设置该参数，参数值最小为5M。 Atlas 200/300/500 推理产品 ，当前版本获取该参数的值为无效值，不支持获取该参数值。

视频编码处理通道属性类型	说明
ACL_VENC_RC_MODE_UINT32 = 9	指定码率控制模式。 1表示变码率VBR模式。 2表示定码率CBR模式。 说明 如果不设置该参数，则采用默认值0。 Atlas 200/300/500 推理产品，默认值0表示CBR模式。 Atlas 200I/500 A2推理产品，默认值0表示VBR模式。 Atlas 推理系列产品，默认值0表示VBR模式。
ACL_VENC_SRC_RATE_UINT32 = 10	输入码流帧率，单位fps。 Atlas 200/300/500 推理产品，取值范围0或者[1, 120]。 Atlas 200I/500 A2推理产品，取值范围0或者[1, 240]。 Atlas 推理系列产品，取值范围0或者[1, 240]。 如果不设置该参数，默认为30；如果设置为0，表示使用默认值，即30。如果该值和实际输入码流帧率相差太大，会影响输出码率。
ACL_VENC_MAX_BITRATE_UINT32 = 11	输出码率，单位kbps。 Atlas 200/300/500 推理产品，取值范围0或[10, 30000]。如果不设置该参数，默认为300；如果设置为0，表示使用默认值，即300。 Atlas 200I/500 A2推理产品，取值范围[2, 614400]，如果不设置该参数，默认为2000；如果设置为0，表示使用默认值，即2000。 Atlas 推理系列产品，取值范围[2, 614400]，如果不设置该参数，默认为2000；如果设置为0，表示使用默认值，即2000。
ACL_VENC_MAX_IP_RATIO_UINT32 = 12	一个GOP内单个I帧bit数和单个P帧bit数的比例，取值范围0或[1,100]。如果不设置该参数，VBR模式下此值默认为80，CBR模式下此值默认为70；如果设置为0，表示使用默认值，即VBR模式下此值默认为80，CBR模式下此值默认为70。

2.16.27 aclmdlConfigAttr

模型加载配置类型	含义
ACL_MDLM_PRIORITY_UINT32 = 0	模型执行的优先级，数字越小优先级越高，取值[0, 7]，可选项。默认值为0。 Atlas 训练系列产品，docker容器内安装CANN，且配置算力分组的场景，该枚举值不生效。

模型加载配置类型	含义
ACL_MDL_LOAD_TYPE_SIZET = 1	模型加载方式，具体请参见表2-74，必选项。
ACL_MDL_PATH_PTR = 2	离线模型文件路径的指针地址，如果选择从文件加载模型，则该选项必选。
ACL_MDL_MEM_ADDR_PTR = 3	模型在内存中的起始地址，如果选择从内存加载模型，则该选项必选。
ACL_MDL_MEM_SIZET = 4	模型在内存中的大小，如果选择从内存加载模型，则该选项必选，与ACL_MDL_MEM_ADDR_PTR选项配合使用。
ACL_MDL_WEIGHT_ADDR_PTR = 5	Device上模型权值内存（存放权值数据）的指针地址，如果需要由用户管理权值内存，则该选项必选。若不配置该选项，则表示由系统管理内存。
ACL_MDL_WEIGHT_SIZET = 6	权值内存大小，单位为Byte，如果需要由用户管理权值内存，则该选项必选，与ACL_MDL_WEIGHT_ADDR_PTR选项配合使用。
ACL_MDL_WORKSPACE_ADDR_PTR = 7	Device上模型所需工作内存（存放模型输入/输出等数据）的指针地址，如果需要由用户管理工作内存，则该选项必选。若不配置该选项，则表示由系统管理内存。
ACL_MDL_WORKSPACE_SIZET = 8	模型所需工作内存的大小，单位为Byte，如果需要由用户管理权值内存，则该选项必选，与ACL_MDL_WORKSPACE_ADDR_PTR选项配合使用。
ACL_MDL_INPUTQ_NUM_SIZET = 9	模型输入队列大小，带队列加载模型时，该选项必选，与ACL_MDL_INPUTQ_ADDR_PTR选项配合使用。
ACL_MDL_INPUTQ_ADDR_PTR = 10	模型输入队列ID的指针地址，带队列加载模型时，该选项必选，一个模型输入对应一个队列ID。
ACL_MDL_OUTPUTQ_NUM_SIZET = 11	模型输出队列大小，带队列加载模型时，该选项必选，与ACL_MDL_OUTPUTQ_ADDR_PTR选项配合使用。
ACL_MDL_OUTPUTQ_ADDR_PTR = 12	模型输出队列ID的指针地址，带队列加载模型时，该选项必选，一个模型输出对应一个队列ID。
ACL_MDL_WORKSPACE_MEM_OPTIMIZE = 13	是否开启模型工作内存优化功能，1开启，0不开启。 若关注内存规划或内存资源有限时，建议在模型加载前，开启工作内存优化功能，此时工作内存中不包含存放模型输入、输出数据的内存，工作内存大小会减小，达到节省内存的目的。 在模型执行前，还需要由用户申请存放模型输入、输出数据的内存，因此即使在模型加载时开启工作内存优化功能，也不会影响后续的模式执行。

模型加载配置类型	含义
ACL_MDL_WEIGHT_PATH_PTR = 14	权重文件所在目录的指针地址。对om模型文件大小有限的场景下，通过本参数可实现权重文件外置功能。 如果将ACL_MDL_LOAD_TYPE_SIZET设置为ACL_MDL_LOAD_FROM_MEM，表示从内存加载模型数据，则支持使用ACL_MDL_WEIGHT_PATH_PTR指定权重文件目录。 一般对om模型文件大小有限制或模型文件加密的场景下，需单独指定权重文件目录，但前提是需在使用ATC工具生成om文件时，将--external_weight参数设置为1（1表示将原始网络中的Const/Constant节点的权重保存在单独的文件中）。

表 2-74 ACL_MDL_LOAD_TYPE_SIZET 对应的属性值

模型加载方式	含义
ACL_MDL_LOAD_FROM_FILE = 1	从文件加载离线模型数据，由系统内部管理内存。
ACL_MDL_LOAD_FROM_FILE_WITH_MEM = 2	从文件加载离线模型数据，由用户自行管理模型运行的内存。
ACL_MDL_LOAD_FROM_MEM = 3	从内存加载离线模型数据，由系统内部管理模型运行的内存。
ACL_MDL_LOAD_FROM_MEM_WITH_MEM = 4	从内存加载离线模型数据，由用户自行管理模型运行的内存。
ACL_MDL_LOAD_FROM_FILE_WITH_Q = 5	从文件加载离线模型数据，模型的输入、输出数据都存放在队列中。

 说明

- 关于如何获取om文件，请参见模型构建。
- om模型文件大小有限制的场景下，使用ATC工具生成om文件时，如将“--external_weight”参数设置为1（1表示将原始网络中的Const/Constant节点的权重保存在单独的文件中，且该文件保存在与om文件同级的“weight”目录下），那么在使用本接口加载om文件时，需将“weight”目录与om文件放在同级目录下，pyACL会进入“weight”目录查找权重文件，未正确放置“weight”目录可能会导致单独的权重文件加载不成功。

2.16.28 aclmdlExecConfigAttr

参数项	含义
ACL_MDL_STREAM_SYNC_TIMEOUT = 0	在执行模型推理时控制Stream任务的超时时间。该属性值为INT32类型。
ACL_MDL_STREAM_SYNC_TIMEOUT = 1	在执行模型推理时控制Event任务的超时时间。该属性值为INT32类型。
ACL_MDL_WORK_ADDRESS_PTR = 2	模型所需工作内存（Device上存放模型执行过程中的临时数据）的指针地址，由用户管理工作内存。一般用于模型一次加载、多并发执行的场景。 当前版本不支持该配置。
ACL_MDL_WORK_SIZE_T = 3	模型所需工作内存的大小，单位为Byte。一般用于模型一次加载、多并发执行的场景。 当前版本不支持该配置。
ACL_MDL_MPAAIMID_SIZE_T = 4	预留配置。
ACL_MDL_AICQOS_SIZE_T = 5	预留配置。
ACL_MDL_AICOST_SIZE_T = 6	预留配置。
ACL_MDL_MEC_TIMER_SIZE_T = 7	预留配置。

2.16.29 aclMemType

内存类型	含义
ACL_MEMTYPE_DEVICE = 0	Device内存。
ACL_MEMTYPE_HOST = 1	Host内存。
ACL_MEMTYPE_HOST_COMPILED_INDEPENDENT = 2	Host内存。

 说明

ACL_MEMTYPE_HOST和ACL_MEMTYPE_HOST_COMPILE_INDEPENDENT都标识Host内存，但在使用上有区别：

- ACL_MEMTYPE_HOST：设置该选项时，模糊编译场景下，算子输入或输出的值的变化，不会触发算子重新编译；精确编译场景下，算子输入或输出的值的变化，会触发算子重新编译。
- ACL_MEMTYPE_HOST_COMPILE_INDEPENDENT：设置该选项时，无论是模糊编译，还是精确编译，算子输入或输出的值的变化，都不会触发算子重新编译。若算子编译时依赖其输入或输出的值，此时如果设置为ACL_MEMTYPE_HOST_COMPILE_INDEPENDENT，则可能会导致编译失败。

2.16.30 aclrtEventWaitStatus

事件状态	说明
ACL_EVENT_STATUS_COMPLETE = 0	完成。
ACL_EVENT_STATUS_NOT_READY = 1	未完成。
ACL_EVENT_STATUS_RESERVED = 2	预留。

2.16.31 aclrtEventRecordedStatus

事件状态	说明
ACL_EVENT_RECORDED_STATUS_NOT_READY = 0	Event未被记录到Stream中，或记录到Stream中的Event未被执行或执行失败。
ACL_EVENT_RECORDED_STATUS_COMPLETE = 1	记录到Stream中的Event执行成功。

2.16.32 aclprofStepTag

数据格式	说明
ACL_STEP_START = 0	迭代开始。
ACL_STEP_END = 1	迭代结束。

 说明

同一个aclprofStepInfo对象、同一个tag只能设置一次，否则Profiling解析会出错。

2.16.33 acldvppChannelMode

数据格式	说明
DVPP_CHNMODE_VPC = 1	指定图片数据处理通道用于实现VPC功能。
DVPP_CHNMODE_JPEGD = 2	指定图片数据处理通道用于实现JPEGD功能。
DVPP_CHNMODE_JPEGGE = 4	指定图片数据处理通道用于实现JPEGE功能。
DVPP_CHNMODE_PNGD = 8	指定图片数据处理通道用于实现PNGD功能。

2.16.34 aclrtFloatOverflowMode

数据格式	说明
ACL_RT_OVERFLOW_MODE_SATURATION = 0	饱和模式，设置成该模式，计算精度可能存在误差。
ACL_RT_OVERFLOW_MODE_INF_NAN = 1	Inf/NaN模式（符合IEEE 754标准），默认值。
ACL_RT_OVERFLOW_MODE_UNDEF = 2	预留参数。

2.16.35 aclrtUtilizationInfo

数据格式	说明
cube_utilization	整型，Cube算子使用率。
vector_utilization	整型，Vector算子使用率。
aicpu_utilization	整型，AI CPU算子使用率。
memory_utilization	整型，Device内存使用率。
utilization_extend	整型，预留扩展字段，当前版本不支持。

2.16.36 aclrtPhysicalMemProp

成员名称	描述
handleType	handle类型，当前仅支持“ACL_MEM_HANDLE_TYPE_NONE”。
allocationType	内存分配类型，当前仅支持“ACL_MEM_ALLOCATION_TYPE_PINNED”。
memAttr	内存属性，具体请参见 2.16.39 aclrtMemAttr 。
location	设置内存所在位置。请参考以下方式传入，其中“id”表示Device的ID，“type”为内存所在位置类型。 location = { 'id' : 0, 'type' : ACL_MEM_LOCATION_TYPE_DEVICE }
reserve	预留。

2.16.37 aclrtStreamStatus

数据格式	说明
ACL_STREAM_STATUS_COMPLETE = 0	Stream上的所有任务已完成。
ACL_STREAM_STATUS_NOT_READY = 1	Stream上至少有一个任务未完成。
ACL_STREAM_STATUS_RESERVED = 0xFFFF	预留。

2.16.38 aclrtDeviceStatus

数据格式	说明
ACL_RT_DEVICE_STATUS_NORMAL = 0	Device正常可用。
ACL_RT_DEVICE_STATUS_ABNORMAL = 1	Device异常。
ACL_RT_DEVICE_STATUS_END = 0xFFFF	预留给值。

2.16.39 aclrtMemAttr

数据格式	说明
ACL_DDR_MEM = 0	DDR内存，DDR上所有大页内存 + 普通内存。
ACL_HBM_MEM = 1	片上内存，片上内存上所有大页内存 + 普通内存。
ACL_DDR_MEM_HUGE = 2	DDR大页内存。
ACL_DDR_MEM_NORMAL = 3	DDR普通内存。
ACL_HBM_MEM_HUGE = 4	片上内存大页内存。
ACL_HBM_MEM_NORMAL = 5	片上内存普通内存。

2.16.40 aclrtMemHandleType

数据格式	说明
ACL_MEM_HANDLE_TYPE_NONE = 0	-

2.16.41 aclDeviceInfo

数据格式	说明
ACL_DEVICE_INFO_UNDEFINED = -1	未知规格。
ACL_DEVICE_INFO_AI_CORE_NUM = 0	AI Core数量。
ACL_DEVICE_INFO_VECTOR_CORE_NUM = 1	Vector Core数量。
ACL_DEVICE_INFO_L2_SIZE = 2	L2 Buffer大小，单位Byte。

2.16.42 aclCannAttr

数据格式	说明
ACL_CANN_ATTR_UNDEFINED = -1	未知特性。
ACL_CANN_ATTR_INF_NAN = 0	溢出检测Inf/NaN模式。

数据格式	说明
ACL_CANN_ATTR_BF16 = 1	bf16数据类型。
ACL_CANN_ATTR_JIT_COMPILE = 2	算子二进制特性。

2.16.43 aclrtMemGranularityOptions

数据格式	说明
ACL_RT_MEM_ALLOC_GRANULARITY_MINIMUM = 0	最小内存粒度。
ACL_RT_MEM_ALLOC_GRANULARITY_RECOMMENDED = 1	推荐内存粒度。
ACL_RT_MEM_ALLOC_GRANULARITY_UNDEF = 0xFFFF	未定义。

3 ISP API

- 3.1 接口简介
- 3.2 ISP系统控制及3A算法注册
- 3.3 ISP系统控制高级参数配置
- 3.4 AE
- 3.5 AWB
- 3.6 CCM
- 3.7 Auto Color Shading
- 3.8 查询内部状态信息（当前版本不支持）
- 3.9 PreGamma
- 3.10 CLUT
- 3.11 CSC
- 3.12 LDCI
- 3.13 ColorTone
- 3.14 WDR
- 3.15 CAC
- 3.16 CA
- 3.17 去FPN
- 3.18 黑电平
- 3.19 Demosaic
- 3.20 Anti FalseColor
- 3.21 NR
- 3.22 Crosstalk Removal
- 3.23 Defect Pixel

- 3.24 Mesh Shading
- 3.25 DRC
- 3.26 Gamma
- 3.27 HLC
- 3.28 EdgeMark
- 3.29 Sharpen
- 3.30 Dehaze
- 3.31 Auto Flicker Type Detection
- 3.32 MotionFusion
- 3.33 统计信息
- 3.34 错误码

3.1 接口简介

ISP提供了各种图像调优算法API，供开发者进行ISP二次开发及图像质量调试。

ISP模块支持标准的Sensor图像数据处理，包括自动白平衡、自动曝光、Demosaic、坏点矫正及镜头阴影矫正等基本功能，也支持WDR、DRC、降噪等高级处理功能。ISP主要支持的图像处理功能如下表所示。

表 3-1 ISP 算法功能

算法	功能	描述
BLC	黑电平校正	提供Sensor相关的黑电平校正。
DPC	坏点矫正	提供对静态坏点和动态坏点的检测和校正功能。
BNR	Bayer降噪	提供在Bayer domain中实现对图像的去噪，目的是去除噪声的同时，保留细节。
FPN	固定噪声消除	通过标定的黑帧或黑行对Sensor输入的图像进行校正，达到去除Sensor FPN的目的。
DEMOSAIC	解马赛克	将Bayer格式的Raw图像转换到RGB图像。
CAC	紫边校正	用来校正由镜头引入的轴向色差（紫边）与横向色差（物体相对两侧带有不同颜色的彩色边缘）。
GAMMA	图像亮度调整	该模块根据伽马曲线分（R，G，B）三个颜色通道调整亮度。
DRC	动态范围压缩	调整图像的显示动态范围，使之在显示设备上的显示效果与人眼感知一致。

算法	功能	描述
WDR	宽动态功能	提供多帧合成宽动态功能： - 支持Sensor内部合成宽动态功能（ Sensor Built-in WDR ） - 最大支持2合1宽动态功能（ WDR ）
AWB	自动白平衡	该模块输出全局统计信息和区域统计信息，软件基于统计信息完成自动白平衡功能。
AE	自动曝光	该模块输出自动曝光信息的统计，软件根据统计信息调节Sensor可实现自动曝光的功能。
AF	自动对焦	当前不支持AF算法，仅支持AF统计信息，统计图像清晰度评价信息，用于完成支持自动对焦功能。
LSC	镜头阴影校正	用于镜头阴影校正。当前只支持mesh shading。
SHARPEN	图像锐化	实现图像的锐化，提高图像的清晰度。
DEHAZE	自动去雾处理	提供强大的分区域去雾能力以改善雾霾场景下视频的对比度和清晰度。
CLUT	颜色三维查找表增强	利用17*17*17大小的3D LUT实现复杂的颜色调整操作，比如亮度的调整，饱和度的调整，阴影区域，中间亮度区域，高亮区域分别调整。
LDCI	局部对比度增强	基于局域直方图均衡的方法来增强局部的对比度，提升暗区细节，同时对图像中高频进行一定的增强，提升对比度。
CA	色彩自适应	饱和度调整及热成像上色，当前不支持热成像上色。
3DNR	3D降噪	通过参数配置，把图像中的高斯噪声去除，使得图像变得平滑，同时降低了编码码率。
HLC	高光屏蔽	将图像中亮度最高的区域拉灰，以减少类似于车灯等高亮光源对人眼的刺激。
Crop	图像裁剪	实现对输入图像裁剪的功能。
GE	图像质量提升	校正Gr与Gb两个通道的失衡，提高部分场景的图像质量。
Expander	解压缩数据	将Sensor内部压缩的数据，进行解压缩。
DG	数字增益	提供分通道的数字增益功能。
MG	最大增益统计	MG模块统计DRC后分块均值，与AE统计分块均值相比，可以得出分块均值增益最大值。MG统计信息包含8bit精度分块R/Gr/Gb/B均值统计，分块最大支持17*15
CCM	颜色校正	通过标准3*3的矩阵和矢量偏移量可完成颜色空间的线性校正。

算法	功能	描述
CSC	色域转换	通过标准3*3的矩阵和矢量偏移量将输入{R, G, B}转换为{Y, U, V}。
DIS	防抖	通过比较两帧之间的差异，计算出帧间的偏移量，根据偏移量进行裁剪，消除图像抖动。
LDC	镜头畸变矫正	对一帧图像进行镜头畸变矫正和展宽功能。展宽功能当前不支持。
FISHEYE	鱼眼矫正	对一帧图像进行鱼眼校正（包括360全景、180全景和Normal三种校正模式）

3.2 ISP 系统控制及 3A 算法注册

本文中接口，如无特殊说明，不支持多进程。

3.2.1 hi_mpi_isp_sensor_reg_callback

函数功能

ISP提供的Sensor注册的回调接口，用于获取差异化的初始化参数，并控制Sensor。

函数原型

```
hi_s32 hi_mpi_isp_sensor_reg_callback(hi_vi_pipe vi_pipe, const hi_isp_sns_attr_info *sns_attr_info, const hi_isp_sensor_register *sns_register)
```

约束说明

此接口不支持多进程操作。

参数说明

参数名	输入/输出	说明
vi_pipe	输入	VI PIPE号。 取值范围：[0, 12)。
sns_attr_info	输入	向ISP注册的Sensor的属性。 该结构体内的sensor_id是Sensor库中自定义的值，需确保唯一，主要用于校对向ISP注册的Sensor和向3A注册的Sensor是否为同一个Sensor。
sns_register	输入	Sensor注册结构体指针。

返回值说明

- 0：成功
- 非0：失败，参见3.34 错误码。

3.2.2 hi_mpi_isp_sensor_unreg_callback

函数功能

ISP提供的取消Sensor注册的回调接口。

函数原型

```
hi_s32 hi_mpi_isp_sensor_unreg_callback(hi_vi_pipe vi_pipe, hi_sensor_id sensor_id)
```

约束说明

此接口不支持多进程操作。

参数说明

参数名	输入/输出	说明
vi_pipe	输入	VI PIPE号。 取值范围：[0, 12)。
sensor_id	输入	向ISP注册的Sensor的Id。 sensor_id是Sensor库中自定义的值，需确保唯一，主要用于校对向ISP反注册的Sensor和向3A反注册的Sensor是否为同一个Sensor。

返回值说明

- 0：成功
- 非0：失败，参见3.34 错误码。

3.2.3 hi_mpi_isp_mem_init

函数功能

初始化ISP内部资源。

函数原型

```
hi_s32 hi_mpi_isp_mem_init(hi_vi_pipe vi_pipe)
```

约束说明

- 调用该接口前，必须先调用[hi_mpi_isp_sensor_reg_callback](#)接口向ISP注册Sensor。
- 不支持多进程，必须要与hi_mpi_isp_sensor_reg_callback、hi_mpi_isp_ae_lib_reg_callback、hi_mpi_isp_awb_lib_reg_callback、hi_mpi_isp_init、hi_mpi_isp_run、hi_mpi_isp_exit、hi_mpi_ae_register、hi_mpi_awb_register接口在同一个进程调用。
- 当前业务正在运行hi_mpi_isp_run接口时，不能调用本接口。
- 推荐调用hi_mpi_isp_exit接口后，再调用本接口重新初始化。
- 不支持相同vi_pipe时，多线程执行ISP创建和销毁（即多线程同时调用hi_mpi_isp_sensor_reg_callback、hi_mpi_isp_ae_lib_reg_callback、hi_mpi_isp_awb_lib_reg_callback、hi_mpi_isp_init、hi_mpi_isp_run、hi_mpi_isp_exit、hi_mpi_ae_register、hi_mpi_awb_register接口）

参数说明

参数名	输入/输出	说明
vi_pipe	输入	VI PIPE号。 取值范围：[0, 12)。

返回值说明

- 0：成功
- 非0：失败，参见[3.34 错误码](#)。

3.2.4 hi_mpi_isp_set_pub_attr

函数功能

设置ISP公共属性，例如裁剪分辨率、帧率等。

函数原型

```
hi_s32 hi_mpi_isp_set_pub_attr(hi_vi_pipe vi_pipe, const hi_isp_pub_attr *pub_attr)
```

约束说明

- 调用本接口前，必须先调用[hi_mpi_isp_mem_init](#)接口初始化ISP内部资源。
- ISP支持运行过程中动态裁剪图像的起始位置。
- 调用本接口后ISP内的处理流程：
ISP firmware判断图像WDR模式、分辨率、帧率是否变化，若都不变则直接返回；否则，ISP firmware会调用通过[hi_mpi_isp_sensor_reg_callback](#)接口注册的回调函数[pfn_cmos_set_wdr_mode](#)、[pfn_cmos_set_image_mode](#)改变Sensor模式；
 - 若Sensor模式不改变（回调函数返回值-2），判断ISP的裁剪宽高是否变化，若有变化，ISP firmware切换分辨率，并调用通过

[hi_mpi_isp_sensor_reg_callback](#)接口注册的回调函数
[pfn_cmos_sensor_init](#)重新配置Sensor；若无变化，就不变。

- 若Sensor模式改变（回调函数返回值为0），则ISP firmware会调用通过[hi_mpi_isp_sensor_reg_callback](#)接口注册的回调函数
[pfn_cmos_sensor_init](#)重新配置Sensor；
- 用户可以修改回调函数[pfn_cmos_set_image_mode](#)的实现来调整Sensor模式切换的顺序。如只提供了5M30fps和1080P60fps初始化序列的Sensor，若要运行1080P30fps，可以从5M30fps裁剪得到，也可以从1080P60fps降帧得到，修改[pfn_cmos_set_image_mode](#)函数实现即可。
- 切换线性模式和帧WDR模式时，同样会判断[pfn_cmos_set_image_mode](#)的返回值，因此线性模式和帧WDR模式应该采用不同的 image_mode，才能保证切换成功。

参数说明

参数名	输入/输出	说明
vi_pipe	输入	VI PIPE号。 取值范围：[0, 12)。
pub_attr	输入	ISP公共属性。

返回值说明

- 0：成功
- 非0：失败，参见[3.34 错误码](#)。

3.2.5 hi_mpi_isp_get_pub_attr

函数功能

获取ISP公共属性。

函数原型

[hi_s32](#) hi_mpi_isp_get_pub_attr([hi_vi_pipe](#) vi_pipe, [hi_isp_pub_attr](#) *pub_attr)

参数说明

参数名	输入/输出	说明
vi_pipe	输入	VI PIPE号。 取值范围：[0, 12)。
pub_attr	输出	ISP公共属性。

返回值说明

- 0：成功
- 非0：失败，参见[3.34 错误码](#)。

3.2.6 hi_mpi_isp_init

函数功能

初始化ISP firmware。

函数原型

```
hi_s32 hi_mpi_isp_init(hi_vi_pipe vi_pipe)
```

约束说明

- 调用本接口前，必须先调用[hi_mpi_isp_set_pub_attr](#)接口设置图像公共属性。
- 为确保前一次进程退出前资源已完全清理，推荐先调用hi_mpi_isp_exit接口，再调用本接口初始化ISP firmware。
- 不支持多进程，必须要与[hi_mpi_isp_sensor_reg_callback](#)、[hi_mpi_isp_run](#)、[hi_mpi_isp_exit](#)、[hi_mpi_ae_register](#)、[hi_mpi_awb_register](#)接口在同一个进程调用。
- 不支持重复调用本接口。
- 不支持相同vi_pipe时，多线程执行ISP创建和销毁（即多线程同时调用[hi_mpi_isp_sensor_reg_callback](#)、[hi_mpi_isp_run](#)、[hi_mpi_isp_exit](#)、[hi_mpi_ae_register](#)、[hi_mpi_awb_register](#)接口）。
- ISP初始化后，需要一帧时间给硬件读取算法系数表。所以调用本接口后一帧时间内，不能调用hi_mpi_vi_stop_pipe接口停止VI PIPE。

参数说明

参数名	输入/输出	说明
vi_pipe	输入	VI PIPE号。 取值范围：[0, 12)。

返回值说明

- 0：成功
- 非0：失败，参见[3.34 错误码](#)。

3.2.7 hi_mpi_isp_run

函数功能

运行ISP firmware。

函数原型

hi_s32 hi_mpi_isp_run(**hi_vi_pipe** vi_pipe)

约束说明

- 调用本接口前，必须先调用**hi_mpi_isp_init**接口初始化ISP firmware。
- 不支持多进程，必须要与**hi_mpi_isp_sensor_reg_callback**、**hi_mpi_isp_init**、**hi_mpi_isp_exit**、**hi_mpi_ae_register**、**hi_mpi_awb_register**接口在同一个进程调用。
- 该接口是阻塞接口，建议用户采用单独的线程调用本接口。

参数说明

参数名	输入/输出	说明
vi_pipe	输入	PIPE号。 取值范围：[0, 12)。

返回值说明

- 0：成功
- 非0：失败，参见**3.34 错误码**。

3.2.8 hi_mpi_isp_exit

函数功能

退出ISP firmware。

函数原型

hi_s32 hi_mpi_isp_exit(**hi_vi_pipe** vi_pipe)

约束说明

- 调用**hi_mpi_isp_run**之后，再调用本接口退出ISP firmware。
- 不支持多进程，必须要与**hi_mpi_isp_sensor_reg_callback**、**hi_mpi_isp_init**、**hi_mpi_isp_run**、**hi_mpi_ae_register**、**hi_mpi_awb_register**接口在同一个进程调用。
- 支持重复调用本接口。
- 不支持相同vi_pipe时，多线程执行ISP创建和销毁（即多线程同时调用**hi_mpi_isp_sensor_reg_callback**、**hi_mpi_isp_run**、**hi_mpi_isp_exit**、**hi_mpi_ae_register**、**hi_mpi_awb_register**接口）。

参数说明

参数名	输入/输出	说明
vi_pipe	输入	PIPE号。 取值范围：[0, 12)。

返回值说明

- 0：成功
- 非0：失败，参见[3.34 错误码](#)。

3.2.9 hi_mpi_isp_set_sns_slave_attr

函数功能

设置从模式Sensor行场同步信号。

从模式Sensor需要其提供行同步信号XHS和场同步信号XVS，在这两个信号的控制下进行曝光与数据读出，这个接口主要配置同步信号发生模块，使其输出Sensor要求的行场时序。此接口一般在Sensor库里调用。

函数原型

hi_s32 hi_mpi_isp_set_sns_slave_attr(**hi_slave_dev** slave_dev, const **hi_isp_slave_sns_sync** *sns_sync)

约束说明

从信号有两组绑定关系，首先pipe 和vsync之间有绑定关系，其次vsync内部还可选择不同的信号源slave，即不同pipe可以选择不同的从信号设备vsync，不同的从信号设备vsync又可以选择不同的信号源slave。

参数说明

参数名	输入/输出	说明
slave_dev	输入	从信号设备号。
sns_sync	输入	同步信号配置。

返回值说明

- 0：成功
- 非0：失败，参见[3.34 错误码](#)。

3.2.10 hi_mpi_isp_get_sns_slave_attr

函数功能

获取从模式Sensor行场同步信号。

函数原型

hi_s32 hi_mpi_isp_get_sns_slave_attr(**hi_slave_dev** slave_dev,
hi_isp_slave_sns_sync *sns_sync)

参数说明

参数名	输入/输出	说明
slave_dev	输入	从信号设备号。
sns_sync	输出	同步信号配置。

返回值说明

- 0：成功
- 非0：失败，参见[3.34 错误码](#)。

3.2.11 hi_mpi_ae_register

函数功能

向ISP注册昇腾AE库。

函数原型

hi_s32 hi_mpi_ae_register(**hi_vi_pipe** vi_pipe, **hi_isp_3a_alg_lib** *ae_lib)

约束说明

- AE库可以注册多个实例。
- 此接口不支持多进程操作。

参数说明

参数名	输入/输出	说明
vi_pipe	输入	VI PIPE号。 取值范围：[0, 12)。
ae_lib	输入	AE算法库结构体指针。

返回值说明

- 0：成功
- 非0：失败，参见[3.34 错误码](#)。

3.2.12 hi_mpi_ae_unregister

函数功能

向ISP取消注册昇腾AE库。

函数原型

```
hi_s32 hi_mpi_ae_unregister(hi_vi_pipe vi_pipe, hi_isp_3a_alg_lib *ae_lib)
```

约束说明

此接口不支持多进程操作。

参数说明

参数名	输入/输出	说明
vi_pipe	输入	VI PIPE号。 取值范围：[0, 12)。
ae_lib	输入	AE 算法库结构体指针。

返回值说明

- 0：成功
- 非0：失败，参见[3.34 错误码](#)。

3.2.13 hi_mpi_ae_sensor_reg_callback

函数功能

AE库提供的Sensor注册的回调接口。AE通过Sensor注册的一系列回调接口，获取差异化的初始化参数，并控制Sensor。

函数原型

```
hi_s32 hi_mpi_ae_sensor_reg_callback(hi_vi_pipe vi_pipe, hi_isp_3a_alg_lib *ae_lib, const hi_isp_sns_attr_info *sns_attr_info, const hi_isp_ae_sensor_register *pregister)
```

约束说明

此接口不支持多进程操作。

参数说明

参数名	输入/输出	说明
vi_pipe	输入	VI PIPE号。 取值范围：[0, 12)。
ae_lib	输入	AE算法库结构体指针。
sns_attr_info	输入	向AE注册的Sensor的属性。 该结构体内的sensor_id是Sensor 库中自定义的值，需确保唯一，主要用于校对向ISP注册的Sensor和向3A注册的Sensor是否为同一个Sensor。
pregister	输入	Sensor注册结构体指针。

返回值说明

- 0：成功
- 非0：失败，参见[3.34 错误码](#)。

3.2.14 hi_mpi_ae_sensor_unreg_callback

函数功能

AE库提供的Sensor注销的回调接口。

函数原型

[hi_s32](#) hi_mpi_ae_sensor_unreg_callback([hi_vi_pipe](#) vi_pipe, [hi_isp_3a_alg_lib](#) *ae_lib, [hi_sensor_id](#) sensor_id)

约束说明

此接口不支持多进程操作。

参数说明

参数名	输入/输出	说明
vi_pipe	输入	VI PIPE号。 取值范围：[0, 12)。
ae_lib	输入	AE算法库结构体指针。
sensor_id	输入	向AE反注册的Sensor的Id。 sensor_id是Sensor库中自定义的值，主要用于校对向ISP反注册的Sensor和向3A反注册的Sensor是否为同一个Sensor。

返回值说明

- 0：成功
- 非0：失败，参见[3.34 错误码](#)

3.2.15 hi_mpi_awb_register

函数功能

向ISP注册昇腾AWB库。

函数原型

[hi_s32](#) hi_mpi_awb_register([hi_vi_pipe](#) vi_pipe, [hi_isp_3a_alg_lib](#) *awb_lib)

约束说明

- AWB库可以注册多个实例。
- 此接口不支持多进程操作。

参数说明

参数名	输入/输出	说明
vi_pipe	输入	VI PIPE号。 取值范围：[0, 12)。
awb_lib	输入	AWB算法库结构体指针。

返回值说明

- 0：成功
- 非0：失败，参见[3.34 错误码](#)。

3.2.16 hi_mpi_awb_unregister

函数功能

向ISP取消注册昇腾AWB库。

函数原型

[hi_s32](#) hi_mpi_awb_unregister([hi_vi_pipe](#) vi_pipe, [hi_isp_3a_alg_lib](#) *awb_lib)

约束说明

此接口不支持多进程操作。

参数说明

参数名	输入/输出	说明
vi_pipe	输入	VI PIPE号。 取值范围：[0, 12)。
awb_lib	输入	AWB算法库结构体指针。

返回值说明

- 0：成功
- 非0：失败，参见[3.34 错误码](#)。

3.2.17 hi_mpi_awb_sensor_reg_callback

函数功能

AWB库提供的Sensor注册的回调接口。AWB通过Sensor注册的一系列回调接口，获取差异化的初始化参数，并控制Sensor。

函数原型

```
hi_s32 hi_mpi_awb_sensor_reg_callback(hi_vi_pipe vi_pipe, hi_isp_3a_alg_lib
*awb_lib, const hi_isp_sns_attr_info *sns_attr_info, const
hi_isp_awb_sensor_register *awb_sns_register);
```

约束说明

此接口不支持多进程操作。

参数说明

参数名	输入/输出	说明
vi_pipe	输入	VI PIPE号。 取值范围：[0, 12)。
awb_lib	输入	AWB算法库结构体指针。
sns_attr_info	输入	向AWB注册的Sensor的属性。 该结构体内的sensor_id是Sensor库中自定义的值，需确保唯一，主要用于校对向ISP注册的Sensor和向3A注册的Sensor是否为同一个Sensor。
awb_sns_register	输入	Sensor注册结构体指针。

返回值说明

- 0：成功
- 非0：失败，参见3.34 错误码。

参考样例

```
hi_isp_3a_alg_lib awb_lib;
hi_isp_awb_sensor_register awb_register;
hi_isp_sns_attr_info sns_attr_info;
hi_isp_awb_sensor_exp_func *exp_funcs = &awb_register.sns_exp;
memset(exp_funcs, 0, sizeof(hi_isp_awb_sensor_exp_func));
exp_funcs->pfn_cmos_get_awb_default = cmos_get_awb_default;
hi_vi_pipe vi_pipe = 0;
awb_lib.id = 0;
sns_attr_info.sensor_id = IMX178_ID;
strcpy(awb_lib.lib_name, HI_AWB_LIB_NAME);
ret = hi_mpi_awb_sensor_reg_callback(vi_pipe, &awb_lib, &sns_attr_info, &awb_register);
if (ret) {
    printf("Sensor register callback function to awb lib failed!\n");
}
return ret;
}
```

3.2.18 hi_mpi_awb_sensor_unreg_callback

函数功能

AWB库提供的Sensor注销的回调接口。

函数原型

```
hi_s32 hi_mpi_awb_sensor_unreg_callback(hi_vi_pipe vi_pipe, hi_isp_3a_alg_lib
*awb_lib, hi_sensor_id sensor_id)
```

约束说明

- 此接口不支持多进程操作。

参数说明

参数名	输入/输出	说明
vi_pipe	输入	VI PIPE号。 取值范围：[0, 12)。
awb_lib	输入	AWB算法库结构体指针。
sensor_id	输入	向AWB注册的Sensor的Id。

返回值说明

- 0：成功
- 非0：失败，参见3.34 错误码。

3.2.19 公共数据类型

此部分作为公共部分，描述了本模块所支持以及未来可能支持的图片格式，以及支持的Payload（数据内容）类型等等。

3.2.19.1 基本数据类型说明

```
typedef unsigned char hi_u8;
typedef signed char hi_s8;
typedef unsigned short hi_u16;
typedef short hi_s16;
typedef unsigned int hi_u32;
typedef int hi_s32;
typedef unsigned long long hi_u64;
typedef long long hi_s64;
typedef char hi_char;
typedef double hi_double;
typedef hi_u32 hi_fr32;

typedef enum {
    HI_FALSE = 0,
    HI_TRUE = 1,
} hi_bool;

#define hi_void void
#define HI_NULL 0L
#define HI_SUCCESS 0
#define HI_FAILURE (-1)
```

3.2.19.2 hi_video_field

说明

定义视频图像帧场类型。

定义

```
typedef enum {
    HI_VIDEO_FIELD_TOP = 0x1,
    HI_VIDEO_FIELD_BOTTOM = 0x2,
    HI_VIDEO_FIELD_INTERLACED = 0x3,
    HI_VIDEO_FIELD_FRAME = 0x4,
    HI_VIDEO_FIELD_BUTT
} hi_video_field;
```

成员

成员名称	描述
HI_VIDEO_FIELD_TOP	顶场类型。
HI_VIDEO_FIELD_BOTTOM	底场类型。
HI_VIDEO_FIELD_INTERLACED	两场间插类型。
HI_VIDEO_FIELD_FRAME	帧类型。
HI_VIDEO_FIELD_BUTT	保留值。

3.2.19.3 hi_pixel_format

说明

DVPP所支持的各种图片格式的枚举。

定义

```
typedef enum {  
    HI_PIXEL_FORMAT_YUV_400 = 0,           // YUV400 8bit  
    HI_PIXEL_FORMAT_YUV_SEMIPLANAR_420 = 1, // YUV420SP NV12 8bit  
    HI_PIXEL_FORMAT_YVU_SEMIPLANAR_420 = 2, // YUV420SP NV21 8bit  
    HI_PIXEL_FORMAT_YUV_SEMIPLANAR_422 = 3, // YUV422SP 8bit  
    HI_PIXEL_FORMAT_YVU_SEMIPLANAR_422 = 4, // YVU422SP 8bit  
    HI_PIXEL_FORMAT_YUV_SEMIPLANAR_444 = 5, // YUV444SP 8bit  
    HI_PIXEL_FORMAT_YVU_SEMIPLANAR_444 = 6, // YVU444SP 8bit  
    HI_PIXEL_FORMAT_YUYV_PACKED_422 = 7,    // YUV422 Package YUYV 8bit  
    HI_PIXEL_FORMAT_UYVY_PACKED_422 = 8,    // YUV422 Package UYVY 8bit  
    HI_PIXEL_FORMAT_YVYU_PACKED_422 = 9,    // YUV422 Package YVYU 8bit  
    HI_PIXEL_FORMAT_VYUY_PACKED_422 = 10,   // YUV422 Package VYUY 8bit  
    HI_PIXEL_FORMAT_YUV_PACKED_444 = 11,    // YUV444 Package 8bit  
    HI_PIXEL_FORMAT_RGB_888 = 12,           // RGB888  
    HI_PIXEL_FORMAT_BGR_888 = 13,          // BGR888  
    HI_PIXEL_FORMAT_ARGB_8888 = 14,        // ARGB8888  
    HI_PIXEL_FORMAT_ABGR_8888 = 15,        // ABGR8888  
    HI_PIXEL_FORMAT_RGBA_8888 = 16,        // RGBA8888  
    HI_PIXEL_FORMAT_BGRA_8888 = 17,        // BGRA8888  
    HI_PIXEL_FORMAT_YUV_SEMI_PLANNER_420_10BIT = 18, // YUV420SP 10bit  
    HI_PIXEL_FORMAT_YVU_SEMI_PLANNER_420_10BIT = 19, // YVU420sp 10bit  
    HI_PIXEL_FORMAT_YVU_PLANAR_420 = 20,    // YVU420P 8bit  
    HI_PIXEL_FORMAT_YVU_PLANAR_422 = 21,    // YVU422P 8bit  
    HI_PIXEL_FORMAT_YVU_PLANAR_444 = 22,    // YVU444P 8bit  
    HI_PIXEL_FORMAT_RGB_444 = 23,           // RGB444 R:4bit G:4bit B:4bit, 当前不支持该格式  
    HI_PIXEL_FORMAT_BGR_444 = 24,           // BGR444 R:4bit G:4bit B:4bit, 当前不支持该格式  
    HI_PIXEL_FORMAT_ARGB_4444 = 25,        // ARGB4444 A:4bit R:4bit G:4bit B:4bit  
    HI_PIXEL_FORMAT_ABGR_4444 = 26,        // ABGR4444 A:4bit B:4bit G:4bit R:4bit, 当前不支持该格式  
    HI_PIXEL_FORMAT_RGBA_4444 = 27,        // RGBA4444 R:4bit G:4bit B:4bit A:4bit, 当前不支持该格式  
    HI_PIXEL_FORMAT_BGRA_4444 = 28,        // BGRA4444 B:4bit G:4bit R:4bit A:4bit, 当前不支持该格式  
    HI_PIXEL_FORMAT_RGB_555 = 29,          // RGB555 R:5bit G:5bit B:5bit, 当前不支持该格式  
    HI_PIXEL_FORMAT_BGR_555 = 30,          // BGR555 B:5bit G:5bit R:5bit, 当前不支持该格式  
    HI_PIXEL_FORMAT_RGB_565 = 31,          // RGB565 R:5bit G:6bit B:5bit, 当前不支持该格式  
    HI_PIXEL_FORMAT_BGR_565 = 32,          // BGR565 B:5bit G:6bit R:5bit, 当前不支持该格式  
    HI_PIXEL_FORMAT_ARGB_1555 = 33,        // ARGB1555 A:1bit R:5bit G:6bit B:5bit  
    HI_PIXEL_FORMAT_ABGR_1555 = 34,        // ABGR1555 A:1bit B:5bit G:6bit R:5bit, 当前不支持该格式  
    HI_PIXEL_FORMAT_RGBA_1555 = 35,        // RGBA1555 A:1bit B:5bit G:6bit R:5bit, 当前不支持该格式  
    HI_PIXEL_FORMAT_BGRA_1555 = 36,        // BGRA1555 A:1bit B:5bit G:6bit R:5bit, 当前不支持该格式  
    HI_PIXEL_FORMAT_ARGB_8565 = 37,        // ARGB8565 A:8bit R:5bit G:6bit B:5bit, 当前不支持该格式  
    HI_PIXEL_FORMAT_ABGR_8565 = 38,        // ABGR8565 A:8bit B:5bit G:6bit R:5bit, 当前不支持该格式  
    HI_PIXEL_FORMAT_RGBA_8565 = 39,        // RGBA8565 A:8bit R:5bit G:6bit B:5bit, 当前不支持该格式  
    HI_PIXEL_FORMAT_BGRA_8565 = 40,        // BGRA8565 A:8bit B:5bit G:6bit R:5bit, 当前不支持该格式  
    HI_PIXEL_FORMAT_ARGB_CLUT2 = 41,       // ARGB Color Lookup Table 2bit  
    HI_PIXEL_FORMAT_ARGB_CLUT4 = 42,       // ARGB Color Lookup Table 4bit  
  
    HI_PIXEL_FORMAT_RGB_BAYER_8BPP = 50,  
    HI_PIXEL_FORMAT_RGB_BAYER_10BPP = 51,  
    HI_PIXEL_FORMAT_RGB_BAYER_12BPP = 52,  
    HI_PIXEL_FORMAT_RGB_BAYER_14BPP = 53,  
    HI_PIXEL_FORMAT_RGB_BAYER_16BPP = 54, // RGB Bayer 16bit, Bayer图像, 当前不支持该格式  
    HI_PIXEL_FORMAT_YUV_PLANAR_420 = 55, // YUV420P 8bit  
    HI_PIXEL_FORMAT_YUV_PLANAR_422 = 56, // YUV422P 8bit  
    HI_PIXEL_FORMAT_YUV_PLANAR_444 = 57, // YUV444P 8bit  
    HI_PIXEL_FORMAT_YVU_PACKED_444 = 58, // YVU444 Package 8bit  
    HI_PIXEL_FORMAT_XYUV_PACKED_444 = 59, // AYUV444 Package 8bit  
    HI_PIXEL_FORMAT_XYVU_PACKED_444 = 60, // AYVU444 Package 8bit  
    HI_PIXEL_FORMAT_YUV_SEMIPLANAR_411 = 61, // YUV411SP 8bit  
    HI_PIXEL_FORMAT_YVU_SEMIPLANAR_411 = 62, // YVU411SP 8bit  
    HI_PIXEL_FORMAT_YUV_PLANAR_411 = 63, // YUV411P 8bit
```

```
HI_PIXEL_FORMAT_YVU_PLANAR_411 = 64, // YVU411P 8bit
HI_PIXEL_FORMAT_YUV_PLANAR_440 = 65, // YUV440P 8bit
HI_PIXEL_FORMAT_YVU_PLANAR_440 = 66, // YVU440P 8bit

HI_PIXEL_FORMAT_RGB_888_PLANAR = 69, // RGB888 Planar
HI_PIXEL_FORMAT_BGR_888_PLANAR = 70, // BGR888 Planar
HI_PIXEL_FORMAT_HSV_888_PACKAGE = 71, // HSV Package, HSV图像package格式, 当前不支持该格式
HI_PIXEL_FORMAT_HSV_888_PLANAR = 72, // HSV Planar, HSV图像Planar格式, 当前不支持该格式
HI_PIXEL_FORMAT_LAB_888_PACKAGE = 73, // LAB Package, LAB图像package格式, 当前不支持该格式
HI_PIXEL_FORMAT_LAB_888_PLANAR = 74, // LAB Planar, LAB图像Planar格式, 当前不支持该格式
HI_PIXEL_FORMAT_S8C1 = 75, // Signed 8bit for 1pixel 1Channel, 每个像素用1个8bit有符号数表示的单通道图像, 当前不支持该格式
HI_PIXEL_FORMAT_S8C2_PACKAGE = 76, // Signed 8bit for 1pixel 2Channel Package, 每个像素用2个8bit有符号数表示的双通道图像Package格式, 当前不支持该格式
HI_PIXEL_FORMAT_S8C2_PLANAR = 77, // Signed 8bit for 1pixel 2Channel Planar, 每个像素用2个8bit有符号数表示的双通道图像Planar格式, 当前不支持该格式
HI_PIXEL_FORMAT_S16C1 = 78, // Signed 16bit 1pixel 1Channel, 每个像素用1个16bit有符号数表示的单通道图像, 当前不支持该格式
HI_PIXEL_FORMAT_U8C1 = 79, // Unsigned 8bit 1pixel 1Channel, 每个像素用1个8bit无符号数表示的单通道图像, 当前不支持该格式
HI_PIXEL_FORMAT_U16C1 = 80, // Unsigned 16bit 1pixel 1Channel, 每个像素用1个16bit无符号数表示的单通道图像, 当前不支持该格式
HI_PIXEL_FORMAT_S32C1 = 81, // Signed 32bit 1pixel 1Channel, 每个像素用1个32bit有符号数表示的单通道图像, 当前不支持该格式
HI_PIXEL_FORMAT_U32C1 = 82, // Unsigned 32bit 1pixel 1Channel, 每个像素用1个32bit无符号数表示的单通道图像, 当前不支持该格式
HI_PIXEL_FORMAT_U64C1 = 83, // Unsigned 64bit 1pixel 1Channel, 每个像素用1个64bit无符号数表示的单通道图像, 当前不支持该格式
HI_PIXEL_FORMAT_S64C1 = 84, // Signed 64bit 1pixel 1Channel, 每个像素用1个64bit有符号数表示的单通道图像, 当前不支持该格式

HI_PIXEL_FORMAT_RGB_888_INT8 = 110, // RGB888 Package 每个像素的单分量占用1个8bit有符号数
HI_PIXEL_FORMAT_BGR_888_INT8 = 111, // BGR888 Package 每个像素的单分量占用1个8bit有符号数
HI_PIXEL_FORMAT_RGB_888_INT16 = 112, // RGB888 Package 每个像素的单分量占用1个16bit有符号数
HI_PIXEL_FORMAT_BGR_888_INT16 = 113, // BGR888 Package 每个像素的单分量占用1个16bit有符号数
HI_PIXEL_FORMAT_RGB_888_INT32 = 114, // RGB888 Package 每个像素的单分量占用1个32bit有符号数
HI_PIXEL_FORMAT_BGR_888_INT32 = 115, // BGR888 Package 每个像素的单分量占用1个32bit有符号数
HI_PIXEL_FORMAT_RGB_888_UINT16 = 116, // RGB888 Package 每个像素的单分量占用1个16bit无符号数
HI_PIXEL_FORMAT_BGR_888_UINT16 = 117, // BGR888 Package 每个像素的单分量占用1个16bit无符号数
HI_PIXEL_FORMAT_RGB_888_UINT32 = 118, // RGB888 Package 每个像素的单分量占用1个32bit无符号数
HI_PIXEL_FORMAT_BGR_888_UINT32 = 119, // BGR888 Package 每个像素的单分量占用1个32bit无符号数
HI_PIXEL_FORMAT_RGB_888_PLANAR_INT8 = 120, // RGB888 Planar 每个像素的单分量占用1个8bit有符号数
HI_PIXEL_FORMAT_BGR_888_PLANAR_INT8 = 121, // BGR888 Planar 每个像素的单分量占用1个8bit有符号数
HI_PIXEL_FORMAT_RGB_888_PLANAR_INT16 = 122, // RGB888 Planar 每个像素的单分量占用1个16bit有符号数
HI_PIXEL_FORMAT_BGR_888_PLANAR_INT16 = 123, // BGR888 Planar 每个像素的单分量占用1个16bit有符号数
HI_PIXEL_FORMAT_RGB_888_PLANAR_INT32 = 124, // RGB888 Planar 每个像素的单分量占用1个32bit有符号数
HI_PIXEL_FORMAT_BGR_888_PLANAR_INT32 = 125, // BGR888 Planar 每个像素的单分量占用1个32bit有符号数
HI_PIXEL_FORMAT_RGB_888_PLANAR_UINT16 = 126, // RGB888 Planar 每个像素的单分量占用1个16bit无符号数
HI_PIXEL_FORMAT_BGR_888_PLANAR_UINT16 = 127, // BGR888 Planar 每个像素的单分量占用1个16bit无符号数
HI_PIXEL_FORMAT_RGB_888_PLANAR_UINT32 = 128, // RGB888 Planar 每个像素的单分量占用1个32bit无
```

```
符号数
HI_PIXEL_FORMAT_BGR_888_PLANAR_UINT32 = 129, // BGR888 Planar 每个像素的单分量占用1个32bit无
符号数
HI_PIXEL_FORMAT_YUV400_UINT16 = 130, // YUV400 Package 每个像素的单分量占用1个16bit无符号
数
HI_PIXEL_FORMAT_YUV400_UINT32 = 131, // YUV400 Package 每个像素的单分量占用1个32bit无符号
数
HI_PIXEL_FORMAT_YUV400_UINT64 = 132, // YUV400 Package 每个像素的单分量占用1个64bit无符号
数
HI_PIXEL_FORMAT_YUV400_INT8 = 133, // YUV400 Package 每个像素的单分量占用1个8bit有符号数
HI_PIXEL_FORMAT_YUV400_INT16 = 134, // YUV400 Package 每个像素的单分量占用1个16bit有符号数
HI_PIXEL_FORMAT_YUV400_INT32 = 135, // YUV400 Package 每个像素的单分量占用1个32bit有符号数
HI_PIXEL_FORMAT_YUV400_INT64 = 136, // YUV400 Package 每个像素的单分量占用1个64bit有符号数
HI_PIXEL_FORMAT_YUV400_FP16 = 137, // YUV400 Package 每个像素用1个float16数据表示
HI_PIXEL_FORMAT_YUV400_FP32 = 138, // YUV400 Package 每个像素用1个float32数据表示
HI_PIXEL_FORMAT_YUV400_FP64 = 139, // YUV400 Package 每个像素用1个float64数据表示
HI_PIXEL_FORMAT_YUV400_BF16 = 140, // YUV400 Package 每个像素用1个BFloat16数据表示

HI_PIXEL_FORMAT_YUV_SEMIPLANAR_440 = 1000, // YUV440SP 8bit
HI_PIXEL_FORMAT_YVU_SEMIPLANAR_440 = 1001, // YVU440SP 8bit
HI_PIXEL_FORMAT_FLOAT32 = 1002, // Float 32bit for 1pixel, 每个像素用1个float32数据表示, 当
前不支持该格式
HI_PIXEL_FORMAT_BUTT = 1003,

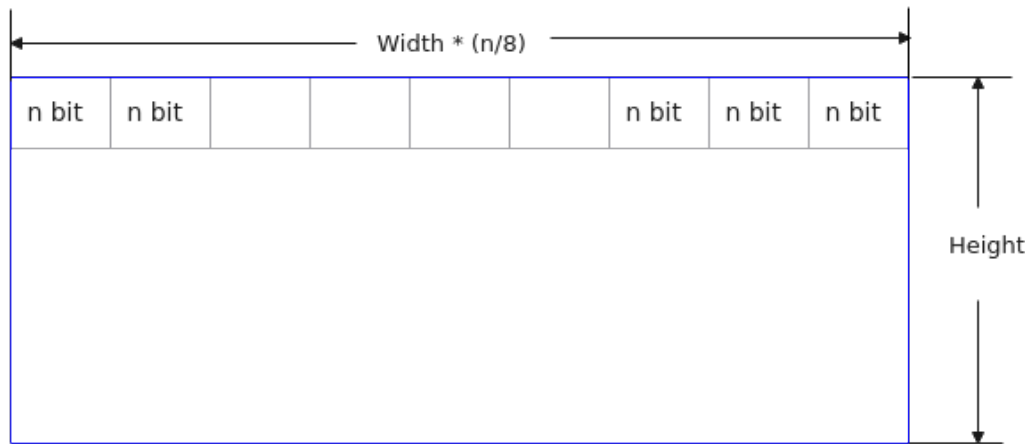
HI_PIXEL_FORMAT_RGB_888_PLANAR_FP16 = 1004, // RGB888 Planar 每个像素用1个float16数据表示
HI_PIXEL_FORMAT_BGR_888_PLANAR_FP16 = 1005, // BGR888 Planar 每个像素用1个float16数据表示
HI_PIXEL_FORMAT_RGB_888_PLANAR_FP32 = 1006, // RGB888 Planar 每个像素用1个float32数据表示
HI_PIXEL_FORMAT_BGR_888_PLANAR_FP32 = 1007, // BGR888 Planar 每个像素用1个float32数据表示
HI_PIXEL_FORMAT_RGB_888_PLANAR_BF16 = 1008, // RGB888 Planar 每个像素用1个BFloat16数据表示
HI_PIXEL_FORMAT_BGR_888_PLANAR_BF16 = 1009, // BGR888 Planar 每个像素用1个BFloat16数据表示
HI_PIXEL_FORMAT_RGB_888_FP16 = 1010, // RGB888 Package, 每个像素用1个float16数据表示
HI_PIXEL_FORMAT_BGR_888_FP16 = 1011, // BGR888 Package, 每个像素用1个float16数据表示
HI_PIXEL_FORMAT_RGB_888_FP32 = 1012, // RGB888 Package, 每个像素用1个float32数据表示
HI_PIXEL_FORMAT_BGR_888_FP32 = 1013, // BGR888 Package, 每个像素用1个float32数据表示
HI_PIXEL_FORMAT_RGB_888_BF16 = 1014, // RGB888 Package 每个像素用1个BFloat16数据表示
HI_PIXEL_FORMAT_BGR_888_BF16 = 1015, // BGR888 Package 每个像素用1个BFloat16数据表示

HI_PIXEL_FORMAT_UNKNOWN = 10000
} hi_pixel_format;
```

参考信息

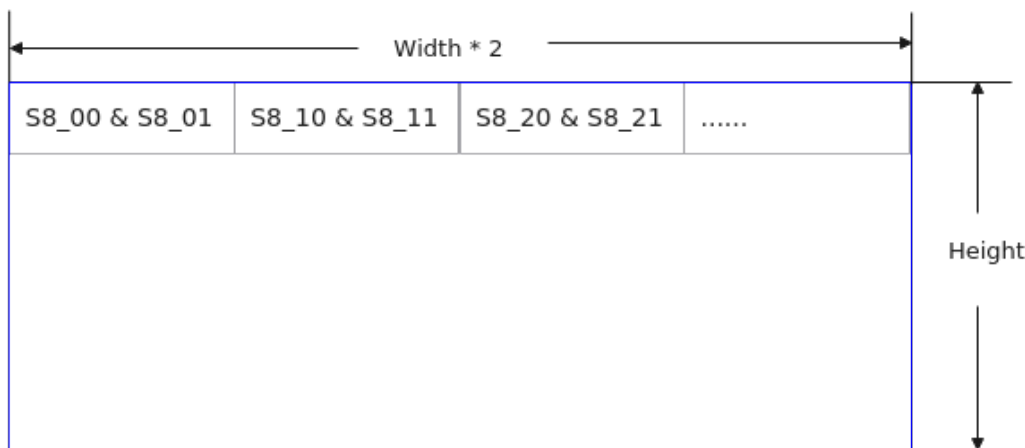
下文以HI_PIXEL_FORMAT_S8C1、HI_PIXEL_FORMAT_S8C2_PACKAGE、HI_PIXEL_FORMAT_S8C2_PLANAR为例说明排布格式，供参考。

图 3-1 HI_PIXEL_FORMAT_S8C1



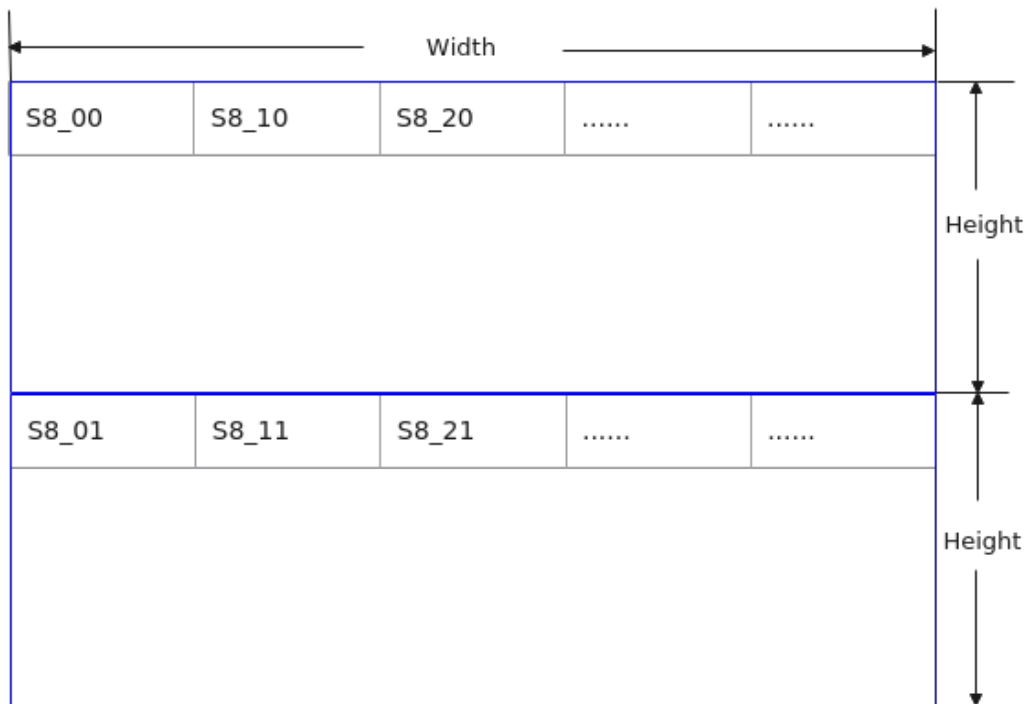
n表示bit位数
Width表示图片宽
Height表示图片高

图 3-2 HI_PIXEL_FORMAT_S8C2_PACKAGE



Width表示图片宽
Height表示图片高
S8_00中的第一个数字0表示像素点，第二个数字0表示通道号

图 3-3 HI_PIXEL_FORMAT_S8C2_PLANAR



Width表示图片宽
Height表示图片高
S8_00中的第一个数字0表示像素点，第二个数字0表示通道号

3.2.19.4 hi_video_format

说明

定义视频格式枚举。

定义

```
typedef enum {  
    HI_VIDEO_FORMAT_LINEAR = 0,  
    HI_VIDEO_FORMAT_TILE_64x16,  
    HI_VIDEO_FORMAT_BUTT  
} hi_video_format;
```

成员

成员名称	描述
HI_VIDEO_FORMAT_LINEAR	线性存储的视频格式。
HI_VIDEO_FORMAT_TILE_64x16	TILE 格式存储的视频格式，其中，tile的块大小为64像素宽，16行像素高，视频/图像编码暂不支持。
HI_VIDEO_FORMAT_BUTT	保留值。

3.2.19.5 hi_compress_mode

说明

定义视频压缩数据格式枚举。

定义

```
typedef enum {  
    HI_COMPRESS_MODE_NONE = 0,  
    HI_COMPRESS_MODE_SEG,  
    HI_COMPRESS_MODE_TILE,  
    HI_COMPRESS_MODE_HFBC,  
    HI_COMPRESS_MODE_LINE,  
    HI_COMPRESS_MODE_FRAME,  
    HI_COMPRESS_MODE_BUTT  
} hi_compress_mode;
```

成员

成员名称	描述
HI_COMPRESS_MODE_NO NE	不压缩。
HI_COMPRESS_MODE_SEG	段压缩，以256字节为一个压缩片段。
HI_COMPRESS_MODE_TILE	Tile压缩，以一个tile为一个压缩片段。
HI_COMPRESS_MODE_HFB C	本昇腾AI处理器特有的HFBC压缩模式，压缩单位是64*8。
HI_COMPRESS_MODE_LIN E	行压缩，以一整行为一个压缩片段进行压缩。如ISP WDR模式下内部的RAW图像采用行压缩。
HI_COMPRESS_MODE_FRA ME	帧压缩，以一整帧为一个压缩片段进行压缩。如ISP 线性模式下内部的RAW图像采用帧压缩，或者3DNR 内部参考帧和重构帧采用帧压缩。
HI_COMPRESS_MODE_BUT T	预留值。

3.2.19.6 hi_dynamic_range

说明

定义动态范围枚举。

定义

```
typedef enum {  
    HI_DYNAMIC_RANGE_SDR8 = 0,  
    HI_DYNAMIC_RANGE_SDR10,  
    HI_DYNAMIC_RANGE_HDR10,  
}
```



```
HI_DYNAMIC_RANGE_HLG,  
HI_DYNAMIC_RANGE_SLF,  
HI_DYNAMIC_RANGE_XDR,  
HI_DYNAMIC_RANGE_BUTT  
} hi_dynamic_range;
```

成员

成员名称	描述
HI_DYNAMIC_RANGE_SDR8	8bit 数据的标准动态范围。
HI_DYNAMIC_RANGE_SDR10	10bit 数据的标准动态范围。
HI_DYNAMIC_RANGE_HDR10	10bit 数据的高动态范围。
HI_DYNAMIC_RANGE_HLG	带有混合对数的Gamma。
HI_DYNAMIC_RANGE_SLF	暂时无效。
HI_DYNAMIC_RANGE_XDR	10bit 数据，算法处理的一个中间类型数据，用户无需关心。
HI_DYNAMIC_RANGE_BUTT	保留值。

注意事项

各个动态范围对应的曲线如下：

Dynamic Range	Transfer Characteristic
SDR8/ SDR10	$V = \alpha * L_c^{0.45} - (\alpha - 1)$ for $1 \geq L_c \geq \beta$ $V = 4.500 * L_c$ for $\beta > L_c \geq 0$
HDR10	对于 L_c 的所有值， $V = ((c_1 + c_2 * L_c^n) \div (1 + c_3 * L_c^n))$ $c_1 = c_3 - c_2 + 1 = 3424 \div 4096 = 0.8359375$ $c_2 = 32 * 2413 \div 4096 = 18.8515625$ $c_3 = 32 * 2392 \div 4096 = 18.6875$ $m = 128 * 2523 \div 4096 = 78.84375$ $n = 0.25 * 2610 \div 4096 = 0.1593017578125$ 其中，峰白区 L_c 等于1通常是为了表示参考输出发光亮度，即10000坎德拉每平方米。
HLG	$V = a * \text{Ln}(12 * L_c - b) + c$ for $1 \geq L_c > 1 \div 12$ $V = \text{Sqrt}(3) * L_c^{0.5}$ for $1 \div 12 \geq L_c \geq 0$ $a = 0.17883277, b = 0.28466892, c = 0.55991073$

3.2.19.7 hi_color_gamut

说明

定义色域范围枚举。

定义

```
typedef enum {  
    HI_COLOR_GAMUT_BT601 = 0,  
    HI_COLOR_GAMUT_BT709,  
    HI_COLOR_GAMUT_BT2020,  
    HI_COLOR_GAMUT_USER,  
    HI_COLOR_GAMUT_BUTT  
} hi_color_gamut;
```

成员

成员名称	描述
HI_COLOR_GAMUT_BT601	BT601色域
HI_COLOR_GAMUT_BT709	BT709色域。
HI_COLOR_GAMUT_BT2020	BT2020色域。当前不支持。
HI_COLOR_GAMUT_USER	用户自定义的色域，非标准色域。
HI_COLOR_GAMUT_BUTT	保留值。

3.2.19.8 hi_video_supplement

说明

定义视频图像帧补充信息。

定义

```
typedef struct {  
    hi_u64 misc_info_phys_addr;  
    hi_u64 jpeg_dcf_phys_addr;  
    hi_u64 isp_info_phys_addr;  
    hi_u64 low_delay_phys_addr;  
    hi_u64 bnr_rnt_phys_addr;  
    hi_u64 motion_data_phys_addr;  
    hi_u64 frame_dng_phys_addr;  
  
    hi_void* ATTRIBUTE misc_info_virt_addr;  
    hi_void* ATTRIBUTE jpeg_dcf_virt_addr;  
    hi_void* ATTRIBUTE isp_info_virt_addr;  
    hi_void* ATTRIBUTE low_delay_virt_addr;  
    hi_void* ATTRIBUTE bnr_mot_virt_addr;  
    hi_void* ATTRIBUTE motion_data_virt_addr;  
    hi_void* ATTRIBUTE frame_dng_virt_addr;  
} hi_video_supplement;
```

成员

成员名称	描述
misc_info_phys_addr	视频扩展信息的物理地址。暂不支持。
jpeg_dcf_phys_addr	Jpeg DCF (Design rule for Camera File system) 信息的物理地址。暂不支持。
isp_info_phys_addr	ISP (Image Signal Processor) 辅助信息的物理地址。暂不支持。
low_delay_phys_addr	低延时信息物理地址。暂不支持。
bnr_rnt_phys_addr	预留参数，暂不支持。
motion_data_phys_addr	运动数据物理地址。暂不支持。
frame_dng_phys_addr	预留参数，暂不支持。
misc_info_virt_addr	视频扩展信息的虚拟地址。暂不支持。
jpeg_dcf_virt_addr	Jpeg DCF信息的虚拟地址。暂不支持。
isp_info_virt_addr	ISP辅助信息的内核态虚拟地址。暂不支持。
low_delay_virt_addr	低延时信息虚拟地址。暂不支持。
bnr_mot_virt_addr	预留参数，暂不支持。
motion_data_virt_addr	运动信息虚拟地址。暂不支持。
frame_dng_virt_addr	预留参数，暂不支持。

3.2.19.9 hi_video_frame

说明

定义视频原始图像帧结构。

定义

```
typedef struct {
    hi_u32      width;
    hi_u32      height;
    hi_video_field field;
    hi_pixel_format pixel_format;
    hi_video_format video_format;
    hi_compress_mode compress_mode;
    hi_dynamic_range dynamic_range;
    hi_color_gamut color_gamut;

    hi_u32      header_stride[HI_MAX_COLOR_COMPONENT];
    hi_u32      width_stride[HI_MAX_COLOR_COMPONENT];
    hi_u32      height_stride[HI_MAX_COLOR_COMPONENT];

    hi_u64      header_phys_addr[HI_MAX_COLOR_COMPONENT];
    hi_u64      phys_addr[HI_MAX_COLOR_COMPONENT];
    hi_void* ATTRIBUTE header_virt_addr[HI_MAX_COLOR_COMPONENT];
    hi_void* ATTRIBUTE virt_addr[HI_MAX_COLOR_COMPONENT];
}
```

```
hi_u32      time_ref;
hi_u64      pts;

hi_u64      user_data[HI_MAX_USER_DATA_NUM];
hi_u32      frame_flag;
hi_video_supplement supplement;
} hi_video_frame;
```

成员

成员名称	描述
width	图像宽度。
height	图像高度。
field	帧场模式。预留参数。
pixel_format	视频图像像素格式。
video_format	视频图像格式。
compress_mode	视频压缩模式。
dynamic_range	动态范围。预留参数。
color_gamut	色域范围。预留参数。
header_stride	图像压缩头跨距。预留参数。
width_stride	输出图像分量的宽度数据跨距。YUV图像则为Y、U、V分量的数据跨距。RGB图像则为R、G、B分量的数据跨距。
height_stride	输出图像分量的高度数据跨距。YUV图像则为Y、U、V分量的数据跨距。RGB图像则为R、G、B分量的数据跨距。 若涉及VI视频采集、ISP系统控制、VPSS视频处理、VENC视频编码、JPEG图片编码功能时，该参数作为预留参数，暂不支持。
header_phys_addr	压缩头物理地址。预留参数。
phys_addr	物理地址。预留参数。
header_virt_addr	压缩头虚拟地址。预留参数。
virt_addr	图像在Device内存中的起始虚拟地址。 作为输入图像时，对于YUV图像，virt_addr[0]为图像起始地址，也是Y分量起始地址，virt_addr[1]为U分量起始地址，virt_addr[2]为V分量起始地址； 作为输出图像时，对于YUV和RGB图像，virt_addr[0]为图像在device上的起始地址，virt_addr[1]和virt_addr[2]为保留字段。
time_ref	图像帧序列号。
pts	图像时间戳。

成员名称	描述
user_data	私有数据。预留参数。
frame_flag	预留参数。
supplement	图像的补充信息。预留参数。

注意事项

10bit数据不压缩时在内存中的存储方式是紧凑排列的。

10bit YUV段压缩数据存储是前8bit和后2bit分开存储。

10bit tile 64*16 数据（即VDH（video decoder hardware）解码之后的数据）不管是压缩还是非压缩，在内存中的存储方式都是前8bit和后2bit分开存储。

3.2.19.10 hi_mod_id

说明

定义模块ID枚举类型。

定义

```
typedef enum {
    HI_ID_CMPI = 0,
    HI_ID_VB = 1,
    HI_ID_SYS = 2,
    HI_ID_RGN = 3,
    HI_ID_CHNL = 4,
    HI_ID_VDEC = 5,
    HI_ID_AVS = 6,
    HI_ID_VPC = 7,
    HI_ID_VENC = 8,
    HI_ID_SVP = 9,
    HI_ID_H264E = 10,
    HI_ID_JPEGE = 11,
    HI_ID_MPEG4E = 12,
    HI_ID_H265E = 13,
    HI_ID_JPEGD = 14,
    HI_ID_VO = 15,
    HI_ID_VI = 16,
    HI_ID_DIS = 17,
    HI_ID_VALG = 18,
    HI_ID_RC = 19,
    HI_ID_AIO = 20,
    HI_ID_AI = 21,
    HI_ID_AO = 22,
    HI_ID_AENC = 23,
    HI_ID_ADEC = 24,
    HI_ID_VPU = 25,
    HI_ID_PCIV = 26,
    HI_ID_PCIVFMW = 27,
    HI_ID_ISP = 28,
    HI_ID_IVE = 29,
    HI_ID_USER = 30,
    HI_ID_DCCM = 31,
    HI_ID_DCCS = 32,
    HI_ID_PROC = 33,
    HI_ID_LOG = 34,
```

```
HI_ID_VFMW = 35,  
HI_ID_H264D = 36,  
HI_ID_GDC = 37,  
HI_ID_PHOTO = 38,  
HI_ID_FB = 39,  
HI_ID_HDMI = 40,  
HI_ID_VOIE = 41,  
HI_ID_TDE = 42,  
HI_ID_HDR = 43,  
HI_ID_PRORES = 44,  
HI_ID_VGS = 45,  
  
HI_ID_FD = 47,  
HI_ID_ODT = 48,  
HI_ID_VQA = 49,  
HI_ID_LPR = 50,  
HI_ID_SVP_NNIE = 51,  
HI_ID_SVP_DSP = 52,  
HI_ID_DPU_RECT = 53,  
HI_ID_DPU_MATCH = 54,  
  
HI_ID_MOTIONSENSOR = 55,  
HI_ID_MOTIONFUSION = 56,  
  
HI_ID_GYRODIS = 57,  
HI_ID_PM = 58,  
HI_ID_SVP_ALG = 59,  
HI_ID_IVP = 60,  
HI_ID_MCF = 61,  
HI_ID_VPSS = 62,  
HI_ID_DRV_VPC = 63,  
HI_ID_PNGD = 64,  
HI_ID_BUTT = 0x100,  
} hi_mod_id;
```

3.2.19.11 hi_video_frame_info

说明

定义视频图像帧信息结构体。

定义

```
typedef struct {  
    hi_video_frame v_frame;  
    hi_u32 pool_id;  
    hi_mod_id mod_id;  
} hi_video_frame_info;
```

成员

成员名称	描述
v_frame	视频图像帧属性结构体。
pool_id	视频缓存池ID，预留参数。
mod_id	DVPP内的子模块ID，预留参数。

3.2.19.12 hi_rect

说明

定义矩形宽高和左上角坐标

定义

```
typedef struct {  
    hi_s32 x;  
    hi_s32 y;  
    hi_s32 width;  
    hi_s32 height;  
} hi_rect;
```

成员

成员名称	描述
x	左上角x轴
y	左上角y轴
width	矩形宽度
height	矩形高度

3.2.19.13 hi_op_mode

说明

定义操作模式枚举。

定义

```
typedef enum {  
    HI_OP_MODE_AUTO = 0, // 自动模式，一般此模式使用程序内部的默认参数  
    HI_OP_MODE_MANUAL = 1 // 手动模式，一般此模式使用用户配置的参数  
} hi_op_mode;
```

3.2.19.14 hi_point

说明

定义坐标点结构体。

定义

```
typedef struct {  
    hi_s32 x;  
    hi_s32 y;  
} hi_point;
```

成员

成员名称	描述
x	x轴坐标值。
y	y轴坐标值。

3.2.19.15 hi_size

说明

定于矩形尺寸的结构体。

定义

```
typedef struct {  
    hi_u32 width;  
    hi_u32 height;  
} hi_size;
```

成员

成员名称	描述
width	宽度
height	高度

3.2.19.16 hi_wdr_mode

说明

定义宽动态模式。

定义

```
typedef enum {  
    HI_WDR_MODE_NONE = 0,  
    HI_WDR_MODE_BUILT_IN,  
    HI_WDR_MODE_QUDRA,  
    HI_WDR_MODE_2To1_LINE,  
    HI_WDR_MODE_2To1_FRAME,  
    HI_WDR_MODE_3To1_LINE,  
    HI_WDR_MODE_3To1_FRAME,  
    HI_WDR_MODE_4To1_LINE,  
    HI_WDR_MODE_4To1_FRAME,  
    HI_WDR_MODE_BUTT,  
} hi_wdr_mode;
```


成员

成员名称	描述
HI_WDR_MODE_NONE	线性模式。
HI_WDR_MODE_BUILT_IN	Sensor合成WDR模式。
HI_WDR_MODE_QUDRA	QUDRA WDR模式。当前版本不支持。
HI_WDR_MODE_2To1_LINE	2 帧合成行WDR模式。
HI_WDR_MODE_2To1_FRAME	2 帧合成帧WDR模式。
HI_WDR_MODE_3To1_LINE	3 帧合成行WDR模式。当前版本不支持。
HI_WDR_MODE_3To1_FRAME	3 帧合成帧WDR模式。当前版本不支持。
HI_WDR_MODE_4To1_LINE	4 帧合成行WDR模式。当前版本不支持。
HI_WDR_MODE_4To1_FRAME	4 帧合成帧WDR模式。当前版本不支持。
HI_WDR_MODE_BUTT	预留值。

说明

此处的WDR模式配置，需要与VI、Sensor、MIPI RX配置保持一致，否则会导致出图异常。

3.2.19.17 hi_vi_pipe

说明

定义VI PIPE号。

定义

```
typedef hi_s32 hi_vi_pipe;
```

3.2.20 ISP 系统控制及 3A 算法注册数据类型

3.2.20.1 hi_sensor_id

说明

定义Sensor ID。

定义

```
typedef hi_s32 hi_sensor_id;
```

3.2.20.2 hi_isp_sns_attr_info

说明

定义ISP Sensor属性。

定义

```
typedef struct {  
    hi_sensor_id sensor_id;  
} hi_isp_sns_attr_info;
```

成员

成员名称	描述
sensor_id	Sensor ID号。

3.2.20.3 hi_isp_cmos_sensor_image_mode

说明

定义Sensor输出的宽高和帧率属性。

定义

```
typedef struct {  
    hi_u16 width;  
    hi_u16 height;  
    hi_float fps;  
    hi_u8 sns_mode;  
} hi_isp_cmos_sensor_image_mode;
```

成员

成员名称	描述
width	Sensor输出的宽度。
height	Sensor输出的高度。
fps	Sensor输出的帧率。
sns_mode	用于进行Sensor初始化序列的选择，在分辨率和帧率相同时，配置不同的sns_mode对应不同的初始化序列；其他情况，sns_mode默认为0。

3.2.20.4 hi_isp_cmos_alg_key

说明

定义 ISP 的各算法是否采用 cmos 中的默认配置的标志位。

定义

```
typedef union {
    hi_u64 key;
    struct {
        hi_u64 bit1_drc : 1; /* [0] */
        hi_u64 bit1_demosaic : 1; /* [1] */
        hi_u64 bit1_pregamma : 1; /* [2] */
        hi_u64 bit1_gamma : 1; /* [3] */
        hi_u64 bit1_sharpen : 1; /* [4] */
        hi_u64 bit1_edge_mark : 1; /* [5] */
        hi_u64 bit1_hlc : 1; /* [6] */
        hi_u64 bit1_ldci : 1; /* [7] */
        hi_u64 bit1_dpc : 1; /* [8] */
        hi_u64 bit1_lsc : 1; /* [9] */
        hi_u64 bit1_ge : 1; /* [10] */
        hi_u64 bit1_anti_false_color : 1; /* [11] */
        hi_u64 bit1_bayer_nr : 1; /* [12] */
        hi_u64 bit1_detail : 1; /* [13] */
        hi_u64 bit1_ca : 1; /* [14] */
        hi_u64 bit1_expander : 1; /* [15] */
        hi_u64 bit1_clut : 1; /* [16] */
        hi_u64 bit1_wdr : 1; /* [17] */
        hi_u64 bit1_dehaze : 1; /* [18] */
        hi_u64 bit1_lcac : 1; /* [19] */
        hi_u64 bit1_acs : 1; /* [20] */
        hi_u64 bit44_rsv : 43; /* [21:63] */
    };
} hi_isp_cmos_alg_key;
```

成员

成员名称	描述
bit1_drc	drc 模块是否采用 cmos 默认配置的标志位。
bit1_demosaic	demosaic 模块是否采用 cmos 默认配置的标志位。
bit1_pregamma	pregamma 模块是否采用 cmos 默认配置的标志位。
bit1_gamma	gamma 模块是否采用 cmos 默认配置的标志位。
bit1_sharpen	sharpen 模块是否采用 cmos 默认配置的标志位。
bit1_edge_mark	edge mark模块是否采用cmos默认配置的标志位
bit1_hlc	hlc模块是否采用cmos默认配置的标志位。
bit1_ldci	ldci 模块是否采用 cmos 默认配置的标志位。
bit1_dpc	dpc 模块是否采用 cmos 默认配置的标志位。
bit1_lsc	lsc 模块是否采用 cmos 默认配置的标志位。
bit1_ge	ge 模块是否采用 cmos 默认配置的标志位。
bit1_anti_false_color	anti false color 模块是否采用 cmos 默认配置的标志位。
bit1_bayer_nr	bayer nr 模块是否采用 cmos 默认配置的标志位。
bit1_detail	detail enhance模块是否采用cmos默认配置的标志位
bit1_ca	ca 模块是否采用 cmos 默认配置的标志位。

成员名称	描述
bit1_expander	expander 模块是否采用 cmos 默认配置的标志位。仅 sensor built-in 模式有效。
bit1_clut	clut 模块是否采用 cmos 默认配置的标志位。
bit1_wdr	wdr 模块是否采用 cmos 默认配置的标志位。
bit1_dehaze	dehaze 模块是否采用 cmos 默认配置的标志位。
bit1_lcac	Local cac 模块是否采用 cmos 默认配置的标志位。
bit1_acs	acs 模块是否采用 cmos 默认配置的标志位。

注意事项

如果ISP的某个算法模块要使用cmos中的配置，要将对应的标志位置为1，否则采用的是算法内部的默认配置。

3.2.20.5 hi_isp_cmos_noise_calibration

说明

定义噪声校正参数。

定义

```
typedef struct {  
    hi_u16 calibration_lut_num;  
    hi_float calibration_coef[HI_ISP_BAYER_CALIBTAION_MAX_NUM]  
[HI_ISP_BAYER_CALIBRATION_PARA_NUM];  
} hi_isp_cmos_noise_calibration;
```

成员

成员名称	描述
calibration_lut_num	标定序列数目。 取值范围：[0, 49]。
calibration_coef	噪声标定表。 HI_ISP_BAYER_CALIBTAION_MAX_NUM用于定义标定噪声模型参数的iso档位个数的最大值，当前为50。 HI_ISP_BAYER_CALIBRATION_PARA_NUM用于定义标定噪声模型参数的最大个数，当前为3。

3.2.20.6 hi_isp_cmos_sensor_max_resolution

说明

定义Sensor最大分辨率结构体。

定义

```
typedef struct {  
    hi_u32 max_width;  
    hi_u32 max_height;  
} hi_isp_cmos_sensor_max_resolution;
```

成员

成员名称	描述
max_width	最大宽度。
max_height	最大高度。

3.2.20.7 hi_isp_dng_blc_repeat_dim

说明

保留结构体。

定义

```
typedef struct {  
    hi_u16 blc_repeat_rows;  
    hi_u16 blc_repeat_cols;  
} hi_isp_dng_blc_repeat_dim;
```

成员

成员名称	描述
blc_repeat_rows	保留成员变量。
blc_repeat_cols	保留成员变量。

3.2.20.8 hi_isp_dng_repeat_pattern_dim

说明

保留结构体。

定义

```
typedef struct {  
    hi_u16 repeat_pattern_dim_rows;
```

```
    hi_u16 repeat_pattern_dim_cols;  
} hi_isp_dng_repeat_pattern_dim;
```

成员

成员名称	描述
repeat_pattern_dim_rows	保留成员变量。
repeat_pattern_dim_cols	保留成员变量。

3.2.20.9 hi_isp_dng_srational

说明

保留结构体。

定义

```
typedef struct {  
    hi_s32 numerator;  
    hi_s32 denominator;  
} hi_isp_dng_srational;
```

成员

成员名称	描述
numerator	保留成员变量。
denominator	保留成员变量。

3.2.20.10 hi_isp_dng_default_scale

说明

保留结构体。

定义

```
typedef struct {  
    hi_isp_dng_srational default_scale_h;  
    hi_isp_dng_srational default_scale_v;  
} hi_isp_dng_default_scale;
```

成员

成员名称	描述
default_scale_h	保留成员变量。
default_scale_v	保留成员变量。

3.2.20.11 hi_isp_dng_cfa_layout_type

说明

保留枚举定义。

定义

```
typedef enum hiDNG_CFALAYOUT_TYPE_E {  
    CFALAYOUT_TYPE_RECTANGULAR = 1,  
    CFALAYOUT_TYPE_A,  
    CFALAYOUT_TYPE_B,  
    CFALAYOUT_TYPE_C,  
    CFALAYOUT_TYPE_D,  
    CFALAYOUT_TYPE_E,  
    CFALAYOUT_TYPE_F,  
    CFALAYOUT_TYPE_G,  
    CFALAYOUT_TYPE_H,  
    CFALAYOUT_TYPE_BUTT  
} hi_isp_dng_cfa_layout_type;
```

成员

成员名称	描述
CFALAYOUT_TYPE_RECTANGULAR	保留成员变量。
CFALAYOUT_TYPE_A	保留成员变量。
CFALAYOUT_TYPE_B	保留成员变量。
CFALAYOUT_TYPE_C	保留成员变量。
CFALAYOUT_TYPE_D	保留成员变量。
CFALAYOUT_TYPE_E	保留成员变量。
CFALAYOUT_TYPE_F	保留成员变量。
CFALAYOUT_TYPE_G	保留成员变量。
CFALAYOUT_TYPE_H	保留成员变量。
CFALAYOUT_TYPE_BUTT	保留成员变量。

3.2.20.12 hi_isp_dng_raw_format

说明

保留结构体。

定义

```
typedef struct {  
    hi_u8 bits_per_sample;
```

```
hi_u8 cfa_plane_color[3];  
hi_isp_dng_cfa_layout_type cfa_layout;  
hi_isp_dng_blc_repeat_dim blc_repeat_dim;  
hi_u32 white_level;  
hi_isp_dng_default_scale default_scale;  
hi_isp_dng_repeat_pattern_dim cfa_repeat_pattern_dim;  
hi_u8 cfa_pattern[4];  
} hi_isp_dng_raw_format;
```

成员

成员名称	描述
bits_per_sample	保留成员变量。
cfa_plane_color	保留成员变量。
cfa_layout	保留成员变量。
blc_repeat_dim	保留成员变量。
white_level	保留成员变量。
default_scale	保留成员变量。
cfa_repeat_pattern_dim	保留成员变量。
cfa_pattern	保留成员变量。

3.2.20.13 hi_isp_cmos_sensor_mode

说明

定义Sensor模式寄存器。

定义

```
typedef struct {  
    hi_u32 sensor_id;  
    hi_u8 sensor_mode;  
    hi_bool valid_dng_raw_format;  
    hi_isp_dng_raw_format dng_raw_format;  
} hi_isp_cmos_sensor_mode;
```

成员

成员名称	描述
sensor_id	Sensor ID。
sensor_mode	Sensor自定义工作模式，不同分辨率与帧率对应不同的工作模式。
valid_dng_raw_format	预留字段，暂不使用，建议配置为0。

成员名称	描述
dng_raw_format	预留字段，暂不使用，建议配置为0。

3.2.20.14 hi_isp_cmos_dng_color_param

说明

定义DNG白平衡校正系数。

定义

```
typedef struct {  
    hi_isp_dng_wb_gain wb_gain1;  
    hi_isp_dng_wb_gain wb_gain2;  
} hi_isp_cmos_dng_color_param;
```

成员

成员名称	描述
wb_gain1	DNG白平衡校正系数1。
wb_gain2	DNG白平衡校正系数2。

3.2.20.15 hi_isp_cmos_wdr_switch_attr

说明

定义WDR切换属性。

定义

```
typedef struct {  
    hi_u32 exp_ratio[HI_ISP_EXP_RATIO_NUM];  
} hi_isp_cmos_wdr_switch_attr;
```

成员

成员名称	描述
exp_ratio	曝光比期望值

注意事项

在 `cmos_get_isp_default` 函数里根据 WDR 模式给 `exp_ratio` 赋默认值，所赋值要与 `cmos_get_ae_default` 中 AE 初始化的曝光比保持一致性：

- 若 `ae_sns_dft->man_ratio_enable` 为 TRUE，`exp_ratio` 与 `ae_sns_dft->arr_ratio` 的值相同；

- 若 ae_sns_dft->man_ratio_enable 为 FALSE，exp_ratio 为 0x40。

3.2.20.16 hi_isp_cmos_default

说明

定义ISP基础算法库的初始化参数结构体。

定义

```
typedef struct {  
    hi_isp_cmos_alg_key    key;  
    const hi_isp_cmos_drc  *drc;  
    const hi_isp_cmos_demosaic *demosaic;  
    const hi_isp_cmos_pregamma *pregamma;  
    const hi_isp_cmos_gamma *gamma;  
    const hi_isp_cmos_sharpen *sharpen;  
    const hi_isp_cmos_edgemark *edge_mark;  
    const hi_isp_cmos_hlc *hlc;  
    const hi_isp_cmos_ldci *ldci;  
    const hi_isp_cmos_dpc *dpc;  
    const hi_isp_cmos_lsc *lsc;  
    const hi_isp_cmos_ge *ge;  
    const hi_isp_cmos_afc *anti_false_color;  
    const hi_isp_cmos_bayernr *bayer_nr;  
    const hi_isp_cmos_ca *ca;  
    const hi_isp_expander_attr *expander;  
    const hi_isp_cmos_clut *clut;  
    const hi_isp_cmos_wdr *wdr;  
    const hi_isp_cmos_dehaze *dehaze;  
    const hi_isp_cmos_lcac *lcac;  
    const hi_isp_cmos_acs *acs;  
    hi_isp_cmos_noise_calibration noise_calibration;  
    hi_isp_cmos_sensor_max_resolution sensor_max_resolution;  
    hi_isp_cmos_sensor_mode sensor_mode;  
    hi_isp_cmos_wdr_switch_attr wdr_switch_attr;  
} hi_isp_cmos_default;
```

成员

成员名称	描述
key	标识各算法是否采用 cmos 中默认配置的 key。
*drc	DRC结构体指针。 结构体说明请参见《ISP图像调优指南》。
*demosaic	Demosaic结构体指针。 结构体说明请参见《ISP图像调优指南》。
*pregamma	PreGamma结构体指针。 结构体说明请参见《ISP图像调优指南》。
*gamma	Gamma结构体指针。 结构体说明请参见《ISP图像调优指南》。
*sharpen	Sharpen结构体指针。 结构体说明请参见《ISP图像调优指南》。

成员名称	描述
*edge_mark	EdgeMark结构体指针。 结构体说明请参见《ISP图像调优指南》。
*hlc	HLC结构体指针。 结构体说明请参见《ISP图像调优指南》。
*ldci	LDCI结构体指针。 结构体说明请参见《ISP图像调优指南》。
*dpc	DPC结构体指针。 结构体说明请参见《ISP图像调优指南》。
*lsc	LSC结构体指针。 结构体说明请参见《ISP图像调优指南》。
*ge	GE模块结构体指针。 结构体说明请参见《ISP图像调优指南》。
*anti_false_color	AntiFalse结构体指针。 结构体说明请参见《ISP图像调优指南》。
*bayer_nr	BayerNR结构体指针。 结构体说明请参见《ISP图像调优指南》。
*ca	CA模块结构体指针。 结构体说明请参见《ISP图像调优指南》。
*expander	Expander结构体指针。仅Sensor built-in模式有效。 结构体说明请参见《ISP图像调优指南》。
*clut	Clut结构体指针。 结构体说明请参见《ISP图像调优指南》。
*wdr	WDR模式结构体指针。 结构体说明请参见《ISP图像调优指南》。
*dehaze	Dehaze结构体指针。 结构体说明请参见《ISP图像调优指南》。
*lcac	Local cac结构体指针。 结构体说明请参见《ISP图像调优指南》。
*acs	ACS结构体指针。 结构体说明请参见《ISP图像调优指南》。
noise_calibration	Noise校正结构体。
sensor_max_resolution	Sensor最大宽高结构体。
sensor_mode	Sensor模式结构体。

成员名称	描述
wdr_switch_attr	WDR切换属性。

注意事项

- ISP的各算法模块如果要采用cmos的默认配置，要在回调函数 [pfn_cmos_get_isp_default](#)中将对应的标志位置1，并给该算法模块的cmos结构体指针赋值。如果cmos默认值配置的不合法，会导致算法初始化失败，同时run起来之后算法不能正常调节。

3.2.20.17 hi_isp_cmos_black_level

说明

定义Sensor的黑电平结构体。

定义

```
typedef struct {  
    hi_bool update;  
    hi_u16 black_level[HI_ISP_BAYER_CHN_NUM];  
} hi_isp_cmos_black_level;
```

成员

成员名称	描述
update	Sensor的黑电平是否会动态根据增益改变。 取值范围： - HI_TRUE：是 - HI_FALSE：否 如果Sensor的黑电平不会动态根据增益改变，update配置为HI_FALSE 即可。
black_level	Sensor的黑电平数组，取值范围：[0, 4095]。 HI_ISP_BAYER_CHN_NUM用于定义Bayer数据的通道数目，当前为4。

3.2.20.18 hi_isp_sns_type

说明

定义Sensor与ISP的通信接口类型。

定义

```
typedef enum {  
    HI_ISP_SNS_I2C_TYPE = 0,
```

```
HI_ISP_SNS_SSP_TYPE,  
HI_ISP_SNS_TYPE_BUTT,  
} hi_isp_sns_type;
```

成员

成员名称	描述
HI_ISP_SNS_I2C_TYPE	Sensor与ISP使用I2C接口通信。
HI_ISP_SNS_SSP_TYPE	Sensor与ISP使用SSP接口通信。
HI_ISP_SNS_TYPE_BUTT	预留。

3.2.20.19 hi_isp_sns_commbus

说明

定义与Sensor绑定的设备号信息。

定义

```
typedef union {  
    hi_s8 i2c_dev;  
    struct {  
        hi_s8 bit4_ssp_dev    : 4;  
        hi_s8 bit4_ssp_cs    : 4;  
    } ssp_dev;  
} hi_isp_sns_commbus;
```

成员

成员名称	描述
i2c_dev	Sensor绑定的I2C设备号。
bit4_ssp_dev	Sensor绑定的SPI设备号。
bit4_ssp_cs	Sensor绑定的SPI片选信号。

3.2.20.20 hi_isp_i2c_data

说明

定义I2C数据参数。

定义

```
typedef struct {  
    hi_bool update; /* RW; Range: [0x0, 0x1]; Format:1.0;  
                    HI_TRUE: The sensor registers are written,  
                    HI_FALSE: The sensor registers are not written */  
    hi_u8 delay_frm_num; /* RW; Number of delayed frames for the sensor register */  
    hi_u8 int_pos; /* RW; Position where the configuration of the sensor register takes effect */
```

```
hi_u8 dev_addr; /* RW;Sensor device address */
hi_u32 reg_addr; /* RW;Sensor register address */
hi_u32 addr_byte_num; /* RW;Bit width of the sensor register address */
hi_u32 data; /* RW;Sensor register data */
hi_u32 data_byte_num; /* RW;Bit width of sensor register data */
} hi_isp_i2c_data;
```

成员

成员名称	描述
update	数据是否会配置sensor寄存器。 HI_TRUE：数据会配置sensor寄存器； HI_FALSE：数据不会配置sensor寄存器。
delay_frame_num	Sensor寄存器延迟配置的帧数。此变量的目的是保证曝光时间和增益同时生效。
interrupt_pos	Sensor寄存器的配置生效的位置。 ● 设置为0x0时表示寄存器在超短帧起始中断配置生效，设置为0x1时表示寄存器在超短帧结束中断配置生效。 ● 设置为0x10时表示寄存器在短帧起始中断配置生效，设置为0x11时表示寄存器在短帧结束中断配置生效。 ● 设置为0x20时表示寄存器在中帧起始中断配置生效，设置为0x21时表示寄存器在中帧结束中断配置生效。 ● 设置为0x30时表示寄存器在长帧起始中断配置生效，设置为0x31时表示寄存器在长帧结束中断配置生效。
dev_addr	Sensor设备地址。
reg_addr	Sensor寄存器地址。
addr_byte_num	Sensor寄存器地址位宽。
data	Sensor寄存器数据。
data_byte_num	Sensor寄存器数据位宽。

3.2.20.21 hi_isp_ssp_data

说明

定义SSP数据参数。

定义

```
typedef struct {
    hi_bool update; /* RW; Range: [0x0, 0x1]; Format:1.0;
                    HI_TRUE: The sensor registers are written,
                    HI_FALSE: The sensor registers are not written */
    hi_u8 delay_frm_num; /* RW; Number of delayed frames for the sensor register */
    hi_u8 int_pos; /* RW;Position where the configuration of the sensor register takes effect */
    hi_u32 dev_addr; /* RW;Sensor device address */
    hi_u32 dev_addr_byte_num; /* RW;Bit width of the sensor device address */
    hi_u32 reg_addr; /* RW;Sensor register address */
    hi_u32 reg_addr_byte_num; /* RW;Bit width of the sensor register address */
    hi_u32 data; /* RW;Sensor register data */
    hi_u32 data_byte_num; /* RW;Bit width of sensor register data */
} hi_isp_ssp_data;
```

成员

成员名称	描述
update	数据是否会配置sensor寄存器。 HI_TRUE：数据会配置sensor寄存器； HI_FALSE：数据不会配置sensor寄存器。
delay_frame_num	sensor寄存器延迟配置的帧数。此变量的目的是保证曝光时间和增益同时生效。
interrupt_pos	Sensor寄存器的配置生效的位置。 ● 设置为0x0时表示寄存器在帧起始中断配置生效，设置为1时表示寄存器在AF中断配置生效。 ● 设置为0x10时表示寄存器在短帧起始中断配置生效，设置为0x11时表示寄存器在短帧结束中断配置生效。 ● 设置为0x20时表示寄存器在中帧起始中断配置生效，设置为0x21时表示寄存器在中帧结束中断配置生效。 ● 设置为0x30时表示寄存器在长帧起始中断配置生效，设置为0x31时表示寄存器在长帧结束中断配置生效。
dev_addr	Sensor设备地址。

成员名称	描述
dev_addr_byte_num	Sensor设备地址位宽。
reg_addr	Sensor寄存器地址。
reg_addr_byte_num	Sensor寄存器地址位宽。
data	Sensor寄存器数据。
data_byte_num	Sensor寄存器数据位宽。

3.2.20.22 hi_isp_sns_regs_info

说明

定义sensor的寄存器信息。

定义

```
typedef struct {
    hi_isp_sns_type sns_type;
    hi_u32 reg_num;
    hi_u8 cfg2_valid_delay_max;
    hi_u32 exp_distance[HI_ISP_WDR_MAX_FRAME_NUM - 1]
    hi_isp_sns_commbus com_bus;
    union {
        hi_isp_i2c_data i2c_data[HI_ISP_MAX_SNS_REGS];
        hi_isp_ssp_data ssp_data[HI_ISP_MAX_SNS_REGS];
    };
    struct {
        hi_bool update;
        hi_u8 delay_frame_num;
        hi_u32 slave_vs_time;
        hi_u32 slave_bind_dev;
    } slv_sync;
    hi_bool config;
} hi_isp_sns_regs_info;
```

成员

成员名称	描述
sns_type	Sensor与ISP的通信接口类型。
reg_num	曝光结果写到sensor时需要配置的寄存器个数，不支持动态修改。
cfg2_valid_delay_max	所有Sensor寄存器从配置到生效延迟的帧数的最大值，单位为帧，用于保证sensor寄存器和ISP寄存器的同步。一般情况下，cmos sensor的曝光时间寄存器的延迟最大，为1~2帧，因此配置一般为1或2。
exp_distance	Sensor在wdr模式下曝光长帧与中帧的行差，中帧与短帧的行差，短帧与短短帧的行差。预留属性，暂不支持。
com_bus	与Sensor绑定的设备号信息

成员名称	描述	
i2c_data	IC2数据参数。	
ssp_data	SSP数据参数。	
update	数据是否会配置sensor寄存器。 HI_TRUE：数据会配置sensor寄存器； HI_FALSE：数据不会配置sensor寄存器。	
delay_frame_num	sensor寄存器延迟配置的帧数。此变量的目的是保证曝光时间和增益同时生效。	
slave_vs_time	XVS信号周期，单位：sensor输入时钟周期。	
slave_bind_dev	Slave设备号与vi_pipe绑定关系。	
config	Sensor寄存器数据配置完成标志。 - TD_TRUE：完成配置。 - TD_FALSE：还未配置。	

注意事项

无

相关数据类型及接口

hi_isp_sensor_exp_func

3.2.20.23 hi_isp_sensor_exp_func

说明

定义sensor回调函数结构体。

定义

```
typedef struct {
    hi_void (*pfn_cmos_sensor_init)(hi_vi_pipe vi_pipe);
    hi_void (*pfn_cmos_sensor_exit)(hi_vi_pipe vi_pipe);
    hi_void (*pfn_cmos_sensor_global_init)(hi_vi_pipe vi_pipe);
    hi_s32 (*pfn_cmos_set_image_mode)(hi_vi_pipe vi_pipe, hi_isp_cmos_sensor_image_mode
*sensor_image_mode);
    hi_s32 (*pfn_cmos_set_wdr_mode)(hi_vi_pipe vi_pipe, hi_u8 mode);
    hi_s32 (*pfn_cmos_get_isp_default)(hi_vi_pipe vi_pipe, hi_isp_cmos_default *def);
    hi_s32 (*pfn_cmos_get_isp_black_level)(hi_vi_pipe vi_pipe, hi_isp_cmos_black_level *black_level);
    hi_s32 (*pfn_cmos_get_sns_reg_info)(hi_vi_pipe vi_pipe, hi_isp_sns_regs_info *sns_regs_info);
    hi_void (*pfn_cmos_set_pixel_detect)(hi_vi_pipe vi_pipe, hi_bool enable);
    hi_s32 (*pfn_cmos_get_awb_gains)(hi_vi_pipe vi_pipe, hi_u32 *sensor_awb_gain);
}
```

```
hi_void (*pfn_cmos_notify_event)(hi_vi_pipe vi_pipe, hi_u32 event_id, hi_void *value);  
} hi_isp_sensor_exp_func;
```

成员

成员名称	描述
pfn_cmos_sensor_init	初始化sensor的回调函数指针。
pfn_cmos_sensor_exit	sensor的回调退出函数指针。
pfn_cmos_sensor_global_init	初始化全局变量的回调函数指针。
pfn_cmos_set_image_mode	设置分辨率和帧率切换的回调函数指针。 返回值0表示sensor模式发生改变，ISP会调用 pfn_cmos_sensor_init 重新配置sensor； 返回值-2表示sensor模式没有变化，ISP不会重新配置sensor。
pfn_cmos_set_wdr_mode	设置wdr模式的回调函数指针。
pfn_cmos_get_isp_default	获取ISP基础算法的初始值的回调函数指针。
pfn_cmos_get_isp_black_level	获取sensor的黑电平值的回调函数指针，支持根据sensor增益动态调整黑电平值。若此处动态调整黑电平值，则外部只能通过接口 hi_mpi_isp_set_black_level_attr 的手动模式设置黑电平。
pfn_cmos_get_sns_reg_info	获取sensor寄存器信息的回调函数指针，用于实现内核态配置AE信息。
pfn_cmos_set_pixel_detect	设置坏点校正开关的回调函数指针。
pfn_cmos_get_awb_gains	配置awb增益回调函数。
pfn_cmos_notify_event	通知模组驱动特殊事件，如休眠后唤醒事件，便于模组做部分逻辑处理。 event_id：必须为0，只支持1个事件，即休眠唤醒事件。 value：必须为null，预留字段。

注意事项

- pfn_cmos_sensor_init, pfn_cmos_get_isp_default, pfn_cmos_get_isp_black_level, pfn_cmos_set_pixel_detect 和 pfn_cmos_get_sns_reg_info 必须赋值，其他回调函数指针如果不需要赋值，应置为 NULL。例如有的 sensor 不支持切换分辨率，那么 pfn_cmos_set_image_mode 需要置为 NULL。
- 不支持切换AWB增益配置位置。

3.2.20.24 hi_isp_sensor_register

说明

定义sensor注册结构体。

定义

```
typedef struct {  
    hi_isp_sensor_exp_func sns_exp;  
} hi_isp_sensor_register;
```

成员

成员名称	描述
sns_exp	Sensor注册的回调函数结构体。

3.2.20.25 hi_isp_bayer_format

说明

定义输入Bayer图像数据格式。

定义

```
typedef enum {  
    HI_ISP_BAYER_RGGB = 0,  
    HI_ISP_BAYER_GRBG = 1,  
    HI_ISP_BAYER_GBRG = 2,  
    HI_ISP_BAYER_BGGR = 3,  
    HI_ISP_BAYER_BUTT  
} hi_isp_bayer_format;
```

成员

成员名称	描述
HI_ISP_BAYER_RGGB	RGGB排列方式。
HI_ISP_BAYER_GRBG	GRGB排列方式。
HI_ISP_BAYER_GBRG	GBRG排列方式。
HI_ISP_BAYER_BGGR	BGGR排列方式。

3.2.20.26 hi_isp_pub_attr

说明

定义 ISP 公共属性。

定义

```
typedef struct {  
    hi_rect      wnd_rect;  
    hi_size      sns_size;  
    hi_float     frame_rate;  
    hi_isp_bayer_format bayer_format;  
    hi_wdr_mode  wdr_mode;  
    hi_u8        sns_mode;  
} hi_isp_pub_attr;
```

成员

成员名称	描述
wnd_rect	裁剪窗口起始位置和图像宽高，必须设置为与vi pipe属性中的size保持一致，且不能比所绑定的dev属性中的宽高大，如果配置不符，会导致出图异常。 - wnd_rect中设置的水平方向起始位置与图像宽度之和应小于sensor输出的图像宽度。 - wnd_rect中设置的垂直方向起始位置与图像高度之和应小于sensor输出的图像高度，由于无法检测sensor实际输出的宽高，当不满足该条件时会返回不报错。
sns_size	Sensor输出的图像宽高。 宽或高的取值范围：[120, 16384]。 设置sns_size时，需与对应的sensor采集属性值保持一致。
frame_rate	输入图像帧率，取值范围为(0.00, 65535.00]。
bayer_format	Bayer数据格式。
wdr_mode	WDR模式选择。
sns_mode	用于进行Sensor初始化序列的选择，在分辨率和帧率相同时，配置不同的sns_mode对应不同的初始化序列；其他情况，sns_mode默认配置为0，可通过sns_size和frame_rate进行初始化序列的选择。

3.2.20.27 hi_isp_frame_info

说明

定义 ISP 实时信息。

定义

```
typedef struct {  
    hi_u32      iso;  
    hi_u32      exposure_time;  
    hi_u32      isp_dgain;  
    hi_u32      again;  
    hi_u32      dgain;  
    hi_u32      ratio[3];  
    hi_u32      isp_nr_strength;  
    hi_u32      f_number;
```

```
hi_u32 sensor_id;  
hi_u32 sensor_mode;  
hi_u32 hmax_times;  
hi_u32 vmax;  
hi_u32 vc_num;  
hi_u64 fsync_timestamp;  
hi_u64 fs_timestamp;  
hi_u64 fe_timestamp;  
hi_u64 pe_timestamp;  
hi_s64 motion_filter[9];  
hi_u32 reserved[10];  
} hi_isp_frame_info;
```

成员

成员名称	描述
iso	当前sensor模拟增益*sensor数字增益*ISP数字增益*100。仅作参考PQ调测参考使用。
exposure_time	曝光时间，单位是微秒（us）。仅作参考PQ调测参考使用。
isp_dgain	ISP数字增益。仅作参考PQ调测参考使用。
again	Sensor的模拟增益。仅作参考PQ调测参考使用。
dgain	Sensor的数字增益。仅作参考PQ调测参考使用。
ratio	多帧合成WDR相邻2帧默认曝光比。仅作参考PQ调测参考使用。
isp_nr_strength	ISP的NR强度。当前未支持，默认值为0。
f_number	当前使用的镜头的F值。仅作参考PQ调测参考使用。
sensor_id	当前使用的sensorID。仅作参考PQ调测参考使用。
sensor_mode	当前使用的sensor序列模式。仅作参考PQ调测参考使用。
hmax_times	当前使用的sensor对应读出一行的时间，单位是纳秒（ns）。仅作参考PQ调测参考使用。
vmax	sensor每帧实际生效的总行数，单位是行。仅作参考PQ调测参考使用。
vc_num	当前未支持，默认值为0。当前未支持，默认值为0。
fsync_timestamp	发出fsync曝光同步信号的时间。
fs_timestamp	收到sensor模组FrameStart信号的时间，仅做参考，会受CPU负载以及调度而存在误差。
fe_timestamp	收到sensor模组FrameEnd信号的时间，仅做参考，会受CPU负载以及调度而存在误差。
pe_timestamp	ISP处理完成后输出图像的时间，仅做维测参考使用。

成员名称	描述
motion_filter	数字防抖的3*3仿射变换矩阵，15bit 小数精度。该值无需用户设置，是开启数字防抖后自动生成的值。用户不能直接使用该矩阵作为仿射变化矩阵，完成以下步骤后可得到仿射变换矩阵： （1）转换为浮点类型做归一化处理 $m = float([x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9])$ $m_1 = \begin{bmatrix} \frac{x_1}{x_9} & \frac{x_2}{x_9} & \frac{x_3}{x_9} \\ \frac{x_4}{x_9} & \frac{x_5}{x_9} & \frac{x_6}{x_9} \\ \frac{x_7}{x_9} & \frac{x_8}{x_9} & \frac{x_9}{x_9} \end{bmatrix}$ （2）求m1的逆矩阵，最终得到m2矩阵，m2矩阵可与图像直接进行运算 $m_2 = inv(m_1)$ （3）m2矩阵与输入图像进行相乘可得到校正后的图像 $img_{dst} = m_2 \otimes img_{src}$
reserved	预留字段，默认输出为0。

3.2.20.28 hi_slave_dev

说明

定义从信号设备号。

定义

```
typedef hi_s32 hi_slave_dev;
```

3.2.20.29 hi_isp_slave_sns_sync

说明

定义从模式sensor同步信号配置。

定义

```
typedef struct {
    union {
        struct {
            hi_u32 bit16_reserved : 16;
            hi_u32 bit_h_inv : 1;
            hi_u32 bit_v_inv : 1;
            hi_u32 bit12_reserved : 12;
            hi_u32 bit_h_enable : 1;
            hi_u32 bit_v_enable : 1;
        } bits;
        hi_u32 bytes;
    } cfg;
    hi_u32 vs_time;
    hi_u32 hs_time;
    hi_u32 vs_cyc;
    hi_u32 hs_cyc;
    hi_u32 hs_dly_cyc;
    hi_u32 slave_mode_time;
} hi_isp_slave_sns_sync;
```

成员

成员名称	描述
bit16_reserved	保留字段。
bit_h_inv	XHS极性配置。 0：正极； 1：负极。
bit_v_inv	XVS极性配置。 0：正极； 1：负极。
bit12_reserved	保留字段。
bit_h_enable	XHS输出使能。
bit_v_enable	XVS输出使能。
vs_time	XVS信号周期，单位：sensor输入时钟周期。
hs_time	XHS信号周期，单位：sensor输入时钟周期。
vs_cyc	XVS有效电平宽度，单位：sensor输入时钟周期。
hs_cyc	XHS有效电平宽度，单位：sensor输入时钟周期。
hs_dly_cyc	XHS脉冲输出相对XVS脉冲的延迟周期配置，单位：sensor输入时钟周期。
slave_mode_time	Sensor从模式时序配置选择寄存器： 0：选择SENSOR0 timing配置； 1：选择SENSOR1 timing配置； 2：选择SENSOR2 timing配置； 3：选择SENSOR3 timing配置。

注意事项

如图 2-4~图 2-6 所示，说明了同步信号发生模块配置参数的含义。

图2-4 同步信号配置时序图

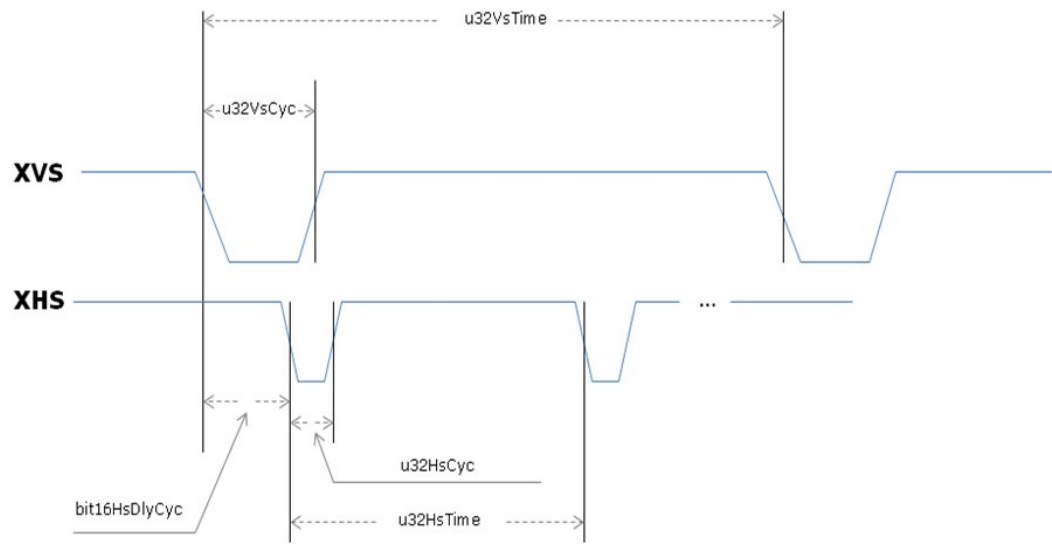


图2-5 同步信号极性翻转



图2-6 同步信号使能



3.2.20.30 hi_isp_3a_alg_lib

说明

定义AE/AWB算法库结构体。

定义

```
typedef struct {  
    hi_s32 id;  
    hi_char lib_name[HI_ISP_ALG_LIB_NAME_SIZE_MAX];  
} hi_isp_3a_alg_lib;
```

成员

成员名称	描述
id	算法库实例的Id，用以支持运行同一个算法库的多个实例
lib_name	标识算法库名称的字符数组，用以区分不同的算法库。 HI_ISP_ALG_LIB_NAME_SIZE_MAX用于定义3A算法库名称的最大字符数，表示20。

3.2.20.31 hi_isp_ae_accuracy_type

说明

定义曝光时间、增益的精度类型的枚举。

定义

```
typedef enum {  
    HI_ISP_AE_ACCURACY_DB = 0,  
    HI_ISP_AE_ACCURACY_LINEAR,  
    HI_ISP_AE_ACCURACY_TABLE,  
    HI_ISP_AE_ACCURACY_BUTT,  
} hi_isp_ae_accuracy_type;
```

成员

成员名称	描述
HI_ISP_AE_ACCURACY_DB	DB精度类型。 DB精度指的是增益的倍数以倍增的形式增加。例如，sensor支持的again 为0db，0.3db，0.6db……这种情况下，精度类型可以设定为HI_ISP_AE_ACCURACY_DB，精度值设置为0.3，在这种精度下，again为 80，则表示 $80 \times 0.3\text{db} = 24\text{db}$，24db是16倍增益；再例如，sensor支持的again为1倍，2倍，4倍，8倍……这种情况下，对应为db则为0db，6db，12db，18db，所以精度类型设定为HI_ISP_AE_ACCURACY_DB，精度值设置为6。由于AE算法内部实现线性转DB时计算精度存在误差，当DB精度较高(小于1)时，建议采用表格精度来实现。
HI_ISP_AE_ACCURACY_LINEAR	线性精度类型。 线性精度指的是增益的倍数均匀增加。例如，sensor能支持的again为 1 倍，$1(1+1/16)$倍，$1(1+2/16)$倍……这种情况下，精度类型可以设定为HI_ISP_AE_ACCURACY_LINEAR，精度值设置为0.0625，在这种精度下，again为32则表示 $32 \times 0.0625 = 2$倍增益。
HI_ISP_AE_ACCURACY_TABLE	表格类型。 表格精度指增益的倍数是通过查表的形式获得，表格统一使用10bit精度，即1024表示1倍增益。当某些sensor的增益值的增加规律非线性或者DB精度较高时，可以使用表格精度方式，AE算法计算出需要的模拟增益/数字增益的数值，用查询sensor增益表格中最接近的值作为数字增益/模拟增益的值。 使用该模式时，需要相应的初始化回调结构体hi_isp_ae_sensor_exp_func 中的回调函数。

3.2.20.32 hi_isp_ae_accuracy

说明

定义曝光时间、增益的精度的结构体。

定义

```
typedef struct {  
    hi_isp_ae_accuracy_type accu_type;  
    float accuracy;  
    float offset;  
} hi_isp_ae_accuracy;
```

成员

成员名称	描述
accu_type	精度类型。
accuracy	精度值。
offset	曝光时间的偏移量，支持正负偏移量设置，以行为单位，该值配置与sensor强相关。

注意事项

- 曝光时间的精度通常都是线性的，都是以行为单位。若曝光时间为线性精度，精度accuracy为大于1的整数，则意味着AE计算出来的曝光行数只能为最小曝光时间加上accuracy的整数倍。该功能可用于计算某些sensor WDR模式下的长/短帧曝光时间，如某些sensor曝光行数必须为 $2n+1$ ，则可以配置最小曝光时间为1或3，配置accuracy为2。
- 曝光时间与sensor增益保持线性关系是AE曝光量分配的前提，即认为曝光量一定的情况下，曝光时间和sensor增益可以相互转换而保持图像亮度基本不变。比如说曝光量为4096，那么分配为曝光时间(2)*增益(2048)，也可以分配为曝光时间(4)*增益(1024)，这两种情况下图像亮度应该基本不变。若不满足这种线性关系，在高亮环境下，由于曝光时间很短，1行曝光时间的变化也容易使画面发生闪烁。某些sensor，如Panasoni MN34220采用1080p@30fps的初始化序列配置时，曝光时间必须偏移0.8018行之后才与sensor增益有线性关系，因此增加了offset这个变量，它为float型变量，以行为单位，将offset设置成0.8018，AE算法内部即可完成偏移处理。一般sensor的曝光时间和增益不存在这种偏移关系，需在cmos.c中将该值配为0。

3.2.20.33 hi_isp_ae_sensor_default

说明

定义AE算法库的初始化参数结构体。

定义

```
typedef struct {
    hi_u8 hist_thresh[HI_ISP_HIST_THRESH_NUM];
    hi_u8 ae_compensation;
    hi_u32 lines_per500ms;
    hi_u32 flicker_freq;
    hi_float fps;
    hi_u32 hmax_times;
    hi_u32 init_exposure;
    hi_u32 init_ae_speed;
    hi_u32 init_ae_tolerance;
    hi_u32 full_lines_std;
    hi_u32 full_lines_max;
    hi_u32 full_lines;
    hi_u32 binning_full_lines;
    hi_u32 max_int_time;
    hi_u32 min_int_time;
    hi_u32 max_int_time_target;
    hi_u32 min_int_time_target;
    hi_isp_ae_accuracy int_time_accu;
```

```
hi_u32 max_again;
hi_u32 min_again;
hi_u32 max_again_target;
hi_u32 min_again_target;
hi_isp_ae_accuracy again_accu;
hi_u32 max_dgain;
hi_u32 min_dgain;
hi_u32 max_dgain_target;
hi_u32 min_dgain_target;
hi_isp_ae_accuracy dgain_accu;
hi_u32 max_isp_dgain_target;
hi_u32 min_isp_dgain_target;
hi_u32 isp_dgain_shift;
hi_u32 max_int_time_step;
hi_bool max_time_step_enable;
hi_u32 max_inc_time_step[HI_ISP_WDR_MAX_FRAME_NUM];
hi_u32 max_dec_time_step[HI_ISP_WDR_MAX_FRAME_NUM];
hi_u32 lf_max_short_time;
hi_u32 lf_min_exposure;
hi_isp_ae_route ae_route_attr;
hi_bool ae_route_ex_valid;
hi_isp_ae_route_ex ae_route_attr_ex;
hi_isp_ae_route ae_route_sf_attr;
hi_isp_ae_route_ex ae_route_sf_attr_ex;
hi_u16 man_ratio_enable;
hi_u32 arr_ratio[HI_ISP_EXP_RATIO_NUM];
hi_isp_iris_type iris_type;
hi_isp_piris_attr piris_attr;
hi_isp_iris_f_no max_iris_fno;
hi_isp_iris_f_no min_iris_fno;
hi_isp_ae_strategy ae_exp_mode;
hi_u16 iso_cal_coef;
hi_u8 ae_run_interval;
hi_u32 exp_ratio_max;
hi_u32 exp_ratio_min;
hi_bool diff_gain_support;
hi_isp_quick_start_param quick_start;
hi_isp_prior_frame prior_frame;
hi_bool ae_gain_sep_cfg;
hi_bool lhcg_support;
hi_u32 sns_lhcg_exp_ratio;
} hi_isp_ae_sensor_default;
```

成员

成员名称	描述
hist_thresh	五段直方图的分割门限值数组，数组内各元素的取值范围为[0,0xff]。推荐使用默认值，线性模式为{0xd,0x28,0x60,0x80}，BUILT_IN WDR模式为{0x20,0x40,0x60,0x80}，帧合成WDR模式为{0xc,0x18,0x60,0x80}。 #define HI_ISP_HIST_THRESH_NUM 4
ae_compensation	AE亮度目标值，取值范围为[0,255]，建议用0x38~0x40。
lines_per500ms	每 500ms 的总行数。
flicker_freq	抗闪频率，数值为当前电源频率的 256 倍。
fps	基准帧率。
hmax_times	Sensor 读出一行的时间，单位：ns。

成员名称	描述
init_exposure	默认初始曝光量，等于曝光时间与曝光增益的乘积，其中曝光时间的单位为微秒(us)。AE 算法采用该值作为初始 5 帧的曝光量，可用于运动 DV 加速启动。 建议关注快速启动的产品形态根据常用场景配置一个合适的初始曝光量，以达到 AE 快速收敛。线性/WDR 模式切换时，也会采用该值作为切换后第一帧的曝光量。合理修改sensor初始化序列的曝光时间和增益，将曝光时间(行数)*增益(6bit 小数精度)计算得到曝光量，并赋值给该变量，可使得切换更加平滑。若不设置，该值默认为 0。
init_ae_speed	默认初始AE调节速度，昇腾AE算法采用该值作为初始100帧的调节速度，可用于运动DV加速启动。 建议关注快速启动的产品形态可将该值配置为128。若不设置，该值默认为0。AE算法内部会将该值钳位至 [64,255]。
init_ae_tolerance	默认初始AE曝光容忍偏差，昇腾AE算法采用该值作为初始100帧的曝光容忍偏差值，可用于设置得到首次AE收敛稳定标志的亮度范围。AE统计亮度第一次落在[目标亮度-init_ae_tolerance, 目标亮度+init_ae_tolerance]范围内时，得到首次AE收敛稳定标志。若不设置，该值默认为 0。若在cmos.c 不对该值赋值或赋值为0，则AE算法内部将其置为ae_compensation/10。AE算法内部会将该值钳位至(0,255]。
full_lines_std	基准帧率时1帧的有效行数。
full_lines_max	sensor降帧后1帧所能达到的最大行数，一般设为sensor所支持的最大行数。
full_lines	sensor每帧实际生效的总行数。使用昇腾AE算法时，回调cmos_fps_set和cmos_slow_framerate_set时必须给该值赋值，用于返回每帧实际生效的总行数。
binning_full_lines	sensor在binning模式下的等效曝光总行数，以行为单位。
max_int_time	最大曝光时间，以行为单位。
min_int_time	最小曝光时间，以行为单位。
max_int_time_target	最大曝光时间目标值，以行为单位。
min_int_time_target	最小曝光时间目标值，以行为单位。
int_time_accu	曝光时间精度。
max_again	最大模拟增益，以倍为单位。
min_again	最小模拟增益，以倍为单位。
max_again_target	最大模拟增益目标值，以倍为单位。
min_again_target	最小模拟增益目标值，以倍为单位。
again_accu	模拟增益精度。

成员名称	描述
max_dgain	最大数字增益，以倍为单位。
min_dgain	最小数字增益，以倍为单位。
max_dgain_target	最大数字增益目标值，以倍为单位。
min_dgain_target	最小数字增益目标值，以倍为单位。
dgain_accu	数字增益精度。
max_isp_dgain_target	最大ISP数字增益目标值，以倍为单位。
min_isp_dgain_target	最小ISP数字增益目标值，以倍为单位。
isp_dgain_shift	ISP数字增益精度。
max_int_time_step	自动长帧模式下普通模式和长帧模式之前切换过程中，对短帧曝光时间减小的最大调整步长，以行数为单位。仅在自动长帧模式下有效。
max_time_step_enable	短帧曝光时间最大步长限定开关，该值为HI_TRUE时，短帧曝光时间最大步长限制功能生效。仅在2帧行合成WDR模式下生效。
max_inc_time_step	目前只开放max_inc_time_step [0]，max_inc_time_step[0]为短帧曝光时间增加的最大步长，以行为单位。即下一帧的曝光行数相比当前帧增加的最大曝光行数不超过max_inc_time_step [0]。仅在2帧行合成WDR模式下max_time_step_enable开关使能后生效。 HI_ISP_WDR_MAX_FRAME_NUM用于定义WDR合成的最大帧数，表示4。
max_dec_time_step	目前只开放max_dec_time_step [0]，max_dec_time_step [0]为短帧曝光时间减少的最大步长，以行为单位。即下一帧的曝光行数相比当前帧减少的最大曝光行数不超过max_dec_time_step [0]。仅在2帧行合成WDR模式下max_time_step_enable开关使能后生效。 HI_ISP_WDR_MAX_FRAME_NUM用于定义WDR合成的最大帧数，表示4。
lf_max_short_time	自动长帧模式下短帧最大曝光时间。
lf_min_exposure	自动长帧模式下强制输出长帧的最小曝光量。
ae_route_attr	AE默认分配路线。
ae_route_ex_valid	AE扩展分配路线是否生效开关，该值为HI_TRUE时使用扩展分配路线，否则使用普通AE分配路线。
ae_route_attr_ex	AE默认扩展分配路线，合理配置可用于优化WDR模式图像效果。
ae_route_sf_attr	AE短帧默认分配路线。

成员名称	描述
ae_route_sf_attr_ex	AE短帧默认扩展分配路线，合理配置可用于优化WDR模式图像效果。
man_ratio_enable	多帧合成WDR手动曝光比使能，该值为HI_TRUE时，曝光比固定，不会根据场景自动更新。
arr_ratio	多帧合成WDR相邻2帧默认曝光比。当man_ratio_enable为HI_TRUE时，采用arr_ratio[0]作为S/VS 默认曝光比，arr_ratio[1]作为M/S默认曝光比，arr_ratio[2]作为 L/M默认曝光比。上述VS，S，M，L分别表示极短帧，短帧，中帧和长帧的曝光时间，2帧合成时只有VS和S；3帧合成时采用VS，S 和 M；4帧合成时采用VS，S，M 和 L。6bit 小数精度。 数组内各元素取值范围：[0x40, 0xFFFF]。 HI_ISP_EXP_RATIO_NUM用于定义WDR曝光比的数目，表示3。
iris_type	默认镜头类型，DC-Iris 或 P-Iris。默认镜头类型与实际对接镜头类型不一致时，AI算法无法正常工作。
piris_attr	P-Iris参数，与具体镜头相关。只有默认镜头类型是P-Iris时，才需要用该结构体参数。
max_iris_fno	P-Iris可用的最大F值，与具体使用镜头相关，使用P-Iris时必须设置，用于限制实际生效的max_iris_fno_target和min_iris_fno_target，避免光圈目标值落到不合理的范围导致AE分配异常。 取值范围为[HI_ISP_IRIS_F_NO_32_0, HI_ISP_IRIS_F_NO_1_0]。
min_iris_fno	P-Iris可用的最小F值，与具体使用镜头相关，使用P-Iris时必须设置，用于限制实际生效的max_iris_fno_target和min_iris_fno_target，避免光圈目标值落到不合理的范围导致AE分配异常。 取值范围为[HI_ISP_IRIS_F_NO_32_0, HI_ISP_IRIS_F_NO_1_0]。
ae_exp_mode	默认曝光策略，高光优先或低光优先。建议 FSWDR 模式设置为低光优先，线性模式及BuiltIn WDR设置为高光优先。若不设置该值，默认为高光优先。
iso_cal_coef	ISO标定系数，用于保证拍照所需DCF信息中显示的ISO是标准的，8bit精度。 取值范围：[0x0, 0xFFFF]，默认值为0x100。
ae_run_interval	默认AE算法运行间隔，以帧为单位。若不设置，则默认AE每帧执行一次。 取值范围：(0x0, 0xFF]

成员名称	描述
exp_ratio_max	默认最大曝光比，仅在多帧合成WDR模式下有效。 - 当man_ratio_enable为HI_FALSE时，exp_ratio_max表示最长帧与最短帧曝光时间比值的最大值。即2帧合成时表示S/VS曝光比的最大值，3帧合成时表示M/VS曝光比的最大值，4帧合成时表示L/VS曝光比的最大值。 - 当man_ratio_enable为HI_TRUE时，exp_ratio_max无效。6bit小数精度，0x40表示曝光比为1倍。 取值范围：[0x40, 0x4000]
exp_ratio_min	默认最小曝光比，仅在多帧合成 WDR 模式下有效。 - 当man_ratio_enable为HI_FALSE时，exp_ratio_min表示长帧曝光时间与短帧曝光时间比值的最小值。 - 当man_ratio_enable为HI_TRUE时，exp_ratio_min无效。6bit 小数精度，0x40 表示曝光比为 1 倍。 取值范围：[0x40, exp_ratio_max]
diff_gain_support	sensor是否支持长短帧不同增益配置，该值为HI_TRUE时，sensor支持长短帧不同sensor模拟增益、sensor数字增益配置，该值为HI_FALSE时，sensor不支持长短帧不同sensor模拟增益、sensor数字增益配置。
quick_start	设置不带光敏快启相关参数。
prior_frame	WDR模式下曝光路线生效的优先帧，线性模式下此参数需配置为LONG_FRAME。
ae_gain_sep_cfg	长短帧增益是否分开分配。
lhcg_support	Sensor是否使用LCG+HCG模式，该值为HI_TRUE时，Sensor使用LCG+HCG模式，该值为HI_FALSE 时， Sensor使用普通模式。
sns_lhcg_exp_ratio	LCG+HCG模式的基础曝光比。

注意事项

- 线性/WDR模式切换时，会回调pfn_cmos_get_ae_default函数更新AE相关默认参数。若WDR模式要使用AE扩展分配路线而线性模式不需要，建议在cmos_get_ae_default函数里面先对AE路线清零：ae_route_ex_valid=HI_FALSE，ae_route_attr.total_num= 0，ae_route_attr_ex.total_num= 0，然后视需要在WDR分支赋值。
- 昇腾AE算法采用init_exposure作为初始5帧的曝光量，可用于运动DV加速启动。建议关注快速启动的产品形态根据常用场景配置一个合适的初始曝光量，以达到AE快速收敛。FSWDR模式该值对应的是长帧曝光量，FSWDR模式若要快速启动，最好在cmos.c设置为固定曝光比，以减少曝光比调整的时间，待AE稳定后再根据需要设置为自动曝光比。若在cmos.c未给init_exposure赋值或将init_exposure赋值为0，则AE算法内部按起始曝光量为1024开始计算。

WDR模式切换时，昇腾AE算法内部会计算切换曝光量以保证切换平滑，如果希望在WDR模式切换时init_exposure生效，可以通过配置 init_ae_speed为 0xFFFFFFFF实现。

- 对于特定Sensor，如OV2718 DCG模式，内部仅支持固定曝光比，需要配置 man_ratio_enable= HI_TRUE，并把ratio配置为Sensor支持的固定曝光比，如果使用自动曝光比或手动配置为不支持的曝光比，会使图像效果不正常。
- max_int_time_step自动长帧模式下普通模式和长帧模式之前切换过程中，对短帧曝光时间减小的最大调整步长，以行数为单位。仅在自动长帧模式下有效。此参数针对短帧曝光时间减小过快会出现坏帧的sensor开放，一般sensor可以设定为较大值，则可以受此参数限制。
- max_time_step_enable, max_inc_time_step [0], max_dec_time_step [0]，这部分参数针对2合1行合成WDR模式下，短帧曝光时间前后2帧之间增加或者减少有限制的sensor开放。
- lf_max_short_time为自动长帧模式下短帧曝光时间的最大值，如果此参数设置过小会导致自动长帧模式下亮区噪声表现变差。
- 对于支持LCG+HCG模式的Sensor，如OV2775，如果使用该模式，需正确配置基础曝光比sns_lhcg_exp_ratio，并且必须配置 prior_frame=HI_ISP_LONG_FRAME，ae_gain_sep_cfg= HI_TRUE，否则会使图像效果不正常。

3.2.20.34 hi_isp_ae_sensor_exp_func

说明定义

sensor回调函数结构体。

定义

```
typedef struct {
    hi_s32 (*pfn_cmos_get_ae_default)(hi_vi_pipe vi_pipe, hi_isp_ae_sensor_default *ae_sns_dft);
    /* the function of sensor set fps */
    hi_void (*pfn_cmos_fps_set)(hi_vi_pipe vi_pipe, hi_float f32_fps, hi_isp_ae_sensor_default *ae_sns_dft);
    hi_void (*pfn_cmos_slow_framerate_set)(hi_vi_pipe vi_pipe, hi_u32 full_lines, hi_isp_ae_sensor_default *ae_sns_dft);

    /* while isp notify ae to update sensor regs, ae call these funcs. */
    hi_void (*pfn_cmos_inttime_update)(hi_vi_pipe vi_pipe, hi_u32 int_time);
    hi_void (*pfn_cmos_gains_update)(hi_vi_pipe vi_pipe, hi_u32 again, hi_u32 dgain);

    hi_void (*pfn_cmos_again_calc_table)(hi_vi_pipe vi_pipe, hi_u32 *again_lin, hi_u32 *again_db);
    hi_void (*pfn_cmos_dgain_calc_table)(hi_vi_pipe vi_pipe, hi_u32 *dgain_lin, hi_u32 *dgain_db);

    hi_void (*pfn_cmos_get_inttime_max)(hi_vi_pipe vi_pipe, hi_u16 man_ratio_enable,
        hi_u32 *ratio, hi_u32 *int_time_max, hi_u32 *int_time_min, hi_u32 *lf_max_int_time);

    /* long frame mode set */
    hi_void (*pfn_cmos_ae_fswdr_attr_set)(hi_vi_pipe vi_pipe, hi_isp_ae_fswdr_attr *ae_fswdr_attr);
    hi_void (*pfn_cmos_ae_quick_start_status_set)(hi_vi_pipe vi_pipe, hi_bool quick_start_status);
    hi_void (*pfn_cmos_exp_param_convert)(hi_vi_pipe vi_pipe, hi_isp_ae_convert_param *exp_param);
} hi_isp_ae_sensor_exp_func;
```

成员

成员名称	描述
pfn_cmos_get_ae_default	获取AE算法库的初始值的回调函数指针。

成员名称	描述
pfn_cmos_fps_set	设置sensor的帧率。
pfn_cmos_slow_framerate_set	设置sensor的降帧。 full_lines: 输入参数, 设置的曝光行。
pfn_cmos_inttime_update	设置sensor的曝光时间。 int_time: 输入参数, 曝光时间, 以曝光行为单位。
pfn_cmos_gains_update	设置sensor的模拟增益和数字增益。 - again: 输入参数, 模拟曝光增益。 - dgain: 输入参数, 数字曝光增益。
pfn_cmos_again_calc_table	计算TABLE类型sensor模拟增益。 - again_lin: 输入参数, 计算出的模拟曝光增益。 - again_lin: 输出参数, 实际与table匹配到的模拟曝光增益。 - again_db: 输出参数, 模拟曝光增益对应table值。
pfn_cmos_dgain_calc_table	计算TABLE类型sensor数字增益。 - dgain_lin: 输入参数, 计算出的数字曝光增益。 - dgain_lin: 输出参数, 实际与table匹配到的数字曝光增益。 - dgain_db: 输出参数, 数字曝光增益对应table值。
pfn_cmos_get_inttime_max	WDR模式下, 计算短帧最大曝光时间的回调函数指针, 与sensor强相关。 - man_ratio_enable: 输入参数, 手动曝光比使能。 - ratio: 输出参数, 曝光时间比。 - int_time_max: 输出参数, 最大曝光时间, 以曝光行为单位。 - int_time_min: 输出参数, 最小曝光时间, 以曝光行为单位。 - lf_max_int_time: 输出参数, 长帧模式最大曝光时间, 以曝光行为单位。
pfn_cmos_ae_fswdr_attr_set	2to1LineWDR模式下, 设置长帧模式。
pfn_cmos_ae_quick_start_status_set	设置AE无光敏快启收敛状态。 不使用无光敏快启功能时, pfn_cmos_ae_quick_start_status_set需要设置为NULL。 quick_start_status: 输入参数, AE无光敏快启收敛状态的标志位, HI_TRUE表示无光敏快启收敛完成, HI_FALSE表示无光敏快启收敛未完成。

成员名称	描述
pfn_cmos_exp_param_convert	完成不同帧率等曝光量转换曝光参数属性，配套 hi_mpi_isp_get_exp_convert 提供。

注意事项

- 如果回调函数指针不需要赋值，需要置为NULL。
- 在hi_isp_ae_sensor_default中定义了曝光时间和增益的精度，pfn_cmos_inttime_update和pfn_cmos_gains_update中设置的曝光时间和增益都是带精度的值，如何转换成sensor的配置值与sensor强相关，请参阅sensor手册。

3.2.20.35 hi_isp_ae_sensor_register

说明

定义sensor注册结构体。

定义

```
typedef struct {  
    hi_isp_ae_sensor_exp_func sns_exp;  
} hi_isp_ae_sensor_register;
```

成员

成员名称	描述
sns_exp	Sensor注册的回调函数结构体。

3.2.20.36 hi_isp_awb_agc_table

说明

定义饱和度初始化参数结构体。

定义

```
typedef struct {  
    hi_bool valid;  
    hi_u8 saturation[HI_ISP_AUTO_ISO_NUM];  
} hi_isp_awb_agc_table;
```

成员

成员名称	描述
valid	该结构体的数据是否有效，取值范围为 0、1。

成员名称	描述
saturation	根据增益动态调节饱和度的插值数组，数组内元素的取值范围为[0, 255]。 saturation为非单调递减序列。 HI_ISP_AUTO_ISO_NUM用于定义ISO档位数，表示16。

3.2.20.37 hi_isp_awb_ccm_tab

说明

定义不同色温下自动颜色校正矩阵系数。

定义

```
typedef struct {  
    hi_u16 color_temp;  
    hi_u16 ccm[HI_ISP_CCM_MATRIX_SIZE];  
} hi_isp_awb_ccm_tab;
```

成员

成员名称	描述
color_temp	当前配置的CCM对应的色温。 取值范围：[2000, 10000]
ccm	不同色温下的颜色校正矩阵，8bit小数精度。Bit[15]是符号位，0表示正数，1表示负数，例如0x8010表示-16。 Bit[12],Bit[13],Bit[14]是无效的，即正数从0x0000到0x0FFF，负数从0x8000到0x8FFF，是CCM的合理参数。 取值范围：[0x0, 0xFFFF]。 HI_ISP_CCM_MATRIX_SIZE用于定义CCM矩阵参数个数，表示9。

3.2.20.38 hi_isp_awb_ccm

说明

定义CCM颜色校正矩阵属性。

定义

```
typedef struct {  
    hi_u16 ccm_tab_num;  
    hi_isp_awb_ccm_tab ccm_tab[HI_ISP_CCM_MATRIX_NUM];  
} hi_isp_awb_ccm;
```

成员

成员名称	描述
ccm_tab_num	当前配置的 CCM 的组数。 取值范围：[3, 7]。
ccm_tab	不同色温下的颜色校正矩阵和对应的色温值。 HI_ISP_CCM_MATRIX_NUM用于定义昇腾AWB可配置的CCM矩阵组数，表示7。

注意事项

如果选择三组CCM，推荐高色温CCM在D50光源标定，中色温CCM在TL84光源标定，低色温CCM在A光源标定。

3.2.20.39 hi_isp_awb_sensor_default

说明

定义AWB算法库的初始化参数结构体。

定义

```
typedef struct {  
    hi_u16 wb_ref_temp;  
    hi_u16 gain_offset[HI_ISP_BAYER_CHN_NUM];  
    hi_s32 wb_para[HI_ISP_AWB_CURVE_PARA_NUM];  
    hi_u16 golden_rgain;  
    hi_u16 golden_bgain;  
    hi_u16 sample_rgain;  
    hi_u16 sample_bgain;  
    hi_isp_awb_agc_table agc_tbl;  
    hi_isp_awb_ccm ccm;  
    hi_u16 init_rgain;  
    hi_u16 init_ggain;  
    hi_u16 init_bgain;  
    hi_u8 awb_run_interval;  
    hi_u16 init_ccm[HI_ISP_CCM_MATRIX_SIZE];  
} hi_isp_awb_sensor_default;
```

成员

成员名称	描述
wb_ref_temp	静态白平衡校正色温，取值范围为[0,0xFFFF]。
gain_offset	静态白平衡的R、Gr、Gb、B颜色通道的增益值数组，数组内元素的取值范围为[0, 0xFFFF]。 HI_ISP_BAYER_CHN_NUM用于定义Bayer数据的通道数目，表示4。

成员名称	描述
wb_para	校正工具给出的白平衡参数数组，取值范围为[0, 0xFFFFFFFF]。 HI_ISP_AWB_CURVE_PARA_NUM用于定义昇腾AWB标定的Planck 曲线参数个数，表示6。
golden_rgain	Golden样机在线标定得到的G/R值。
golden_bgain	Golden样机在线标定得到的G/B值。
sample_rgain	当前样机在线标定得到的G/R值。
sample_bgain	当前样机在线标定得到的G/B值。
agc_tbl	该结构体的数据是否有效，取值范围为[0,1]。
	根据增益动态调节饱和度的插值数组，取值范围为[0, 255]。
ccm	当前配置的CCM的组数。 取值范围： [3, 7]
	不同色温下的颜色校正矩阵和对应的色温值。
init_rgain	ISP启动时R通道白平衡增益初始值。
init_ggain	ISP启动时G通道白平衡增益初始值。
init_bgain	ISP启动时B通道白平衡增益初始值。
awb_run_interval	ISP启动时AWB工作频率。 取值范围： [0x1, 0xFF]
init_ccm	ISP启动时CCM初始值数组。 HI_ISP_CCM_MATRIX_SIZE用于定义CCM矩阵参数个数，表示9。

注意事项

参考色温即静态白平衡校正的环境色温，需要提供色度计测量的实际值。

3.2.20.40 hi_isp_awb_sensor_exp_func

说明

定义sensor回调函数结构体。

定义

```
typedef struct {  
    hi_s32 (*pfn_cmos_get_awb_default)(hi_vi_pipe vi_pipe, hi_isp_awb_sensor_default *awb_sns_dft);  
} hi_isp_awb_sensor_exp_func;
```

成员

成员名称	描述
vi_pipe	vi_pipe号。
pfn_cmos_get_awb_default	获取 AWB 算法库的初始值的回调函数指针。

3.2.20.41 hi_isp_awb_sensor_register

说明

定义sensor注册结构体。

定义

```
typedef struct {  
    hi_isp_awb_sensor_exp_func sns_exp;  
} hi_isp_awb_sensor_register;
```

成员

成员名称	描述
sns_exp	sensor注册的回调函数结构体。

3.3 ISP 系统控制高级参数配置

3.3.1 hi_mpi_isp_run_once

函数功能

运行ISP firmware一次。

约束说明

- 运行前需要确保已调用hi_mpi_isp_init初始化 ISP。
- 不支持多进程，必须要与hi_mpi_isp_sensor_reg_callback、hi_mpi_isp_ae_lib_reg_callback、hi_mpi_isp_awb_lib_reg_callback、hi_mpi_isp_init、hi_mpi_isp_run、hi_mpi_isp_exit、hi_mpi_ae_register、hi_mpi_awb_register接口在同一个进程调用。
- 该接口是阻塞接口。
- 该接口主要用于用户给 BE 灌 RAW。在使用时，要等待上一次发送的 RAW 数据处理完成之后才能进行下一次的 hi_mpi_isp_run_once 接口调用。发送RAW数据是调用 hi_mpi_vi_send_pipe_raw接口，处理RAW数据是调用接口 hi_mpi_vi_get_chn_frame。
- 使用hi_mpi_isp_run_once接口时处理视频流时，支持模式切换及分辨率切换，切换流程与使用hi_mpi_isp_run处理视频流类似：即切换过程中ISP模块无需退出，

VI模块需要销毁重建。区别在于：使用hi_mpi_isp_run_once处理视频流时需要用户创建一个线程。

- 对于同一个VI PIPE，不能同时使用hi_mpi_isp_run和hi_mpi_isp_run_once接口。

函数原型

```
hi_s32 hi_mpi_isp_run_once(hi_vi_pipe vi_pipe)
```

参数说明

参数名	输入/输出	说明
vi_pipe	输入	VI PIPE号。 取值范围：[0, 12)。

返回值说明

- 0：成功
- 非0：失败，参见[ISP返回码](#)。

3.3.2 hi_mpi_isp_set_module_ctrl

函数功能

设置是否启用ISP各功能模块。

约束说明

该接口对应的寄存器与各模块的使能寄存器复用。

函数原型

```
hi_s32 hi_mpi_isp_set_module_ctrl(hi_vi_pipe vi_pipe, const hi_isp_module_ctrl *mod_ctrl)
```

参数说明

参数名	输入/输出	说明
vi_pipe	输入	VI PIPE号。 取值范围：[0, 12)。
mod_ctrl	输入	模块控制值。每个比特位控制着 ISP 中的一个功能模块的使能。 0：开启该模块 1：关闭该模块

返回值说明

- 0：成功
- 非0：失败，参见[ISP返回码](#)。

3.3.3 hi_mpi_isp_get_module_ctrl

函数功能

获取ISP各功能模块是否启用的开关。

函数原型

[hi_s32](#) hi_mpi_isp_get_module_ctrl([hi_vi_pipe](#) vi_pipe, [hi_isp_module_ctrl](#) *mod_ctrl)

参数说明

参数名	输入/输出	说明
vi_pipe	输入	VI PIPE号。 取值范围：[0, 12)。
mod_ctrl	输出	模块控制值。每个比特位控制着 ISP 中的一个功能模块的使能。 • 0：开启该模块 • 1：关闭该模块

返回值说明

- 0：成功
- 非0：失败，参见[ISP返回码](#)。

3.3.4 hi_mpi_isp_set_ctrl_param

函数功能

设置 ISP 控制参数。

函数原型

[hi_s32](#) hi_mpi_isp_set_ctrl_param([hi_vi_pipe](#) vi_pipe, const [hi_isp_ctrl_param](#) *isp_ctrl_param)

参数说明

参数名	输入/输出	说明
vi_pipe	输入	VI PIPE号。 取值范围：[0, 12)。
isp_ctrl_param	输入	ISP 控制参数结构体指针。

返回值说明

- 0：成功
- 非0：失败，参见ISP返回码。

3.3.5 hi_mpi_isp_get_ctrl_param

函数功能

获取 ISP 控制参数。

函数原型

```
hi_s32 hi_mpi_isp_get_ctrl_param(hi_vi_pipe vi_pipe, hi_isp_ctrl_param *isp_ctrl_param)
```

参数说明

参数名	输入/输出	说明
vi_pipe	输入	VI PIPE号。 取值范围：[0, 12)。
isp_ctrl_param	输出	ISP 控制参数结构体指针。

返回值说明

- 0：成功
- 非0：失败，参见ISP返回码。

3.3.6 hi_mpi_isp_ae_lib_reg_callback

函数功能

用于向ISP注册AE算法库回调函数。

约束说明

- 使用昇腾内置的AE算法库时，不需要关注此接口；使用用户自定义的AE算法库时，需要调用此接口向ISP注册回调函数。

- 此接口不支持多进程操作。
- 最大支持2个AE库注册。

函数原型

```
hi_s32 hi_mpi_isp_ae_lib_reg_callback(hi_vi_pipe vi_pipe, const  
hi_isp_3a_alg_lib *ae_lib, const hi_isp_ae_register *ae_register)
```

参数说明

参数名	输入/输出	说明
vi_pipe	输入	VI PIPE号。 取值范围：[0, 12)。
ae_lib	输入	AE 库结构体指针。
ae_register	输入	AE 库注册结构体指针。

返回值说明

- 0：成功
- 非0：失败，参见[ISP返回码](#)。

参考样例

```
hi_isp_ae_register ae_register;  
hi_s32 ret = HI_SUCCESS;  
ae_register.ae_exp_func.pfn_ae_init = ae_init;  
ae_register.ae_exp_func.pfn_ae_run = ae_run;  
ae_register.ae_exp_func.pfn_ae_ctrl = ae_ctrl;  
ae_register.ae_exp_func.pfn_ae_exit = ae_exit;  
ret = hi_mpi_isp_ae_lib_reg_callback(vi_pipe, ae_lib, &ae_register);  
if (HI_SUCCESS != ret) {  
    printf("Hi_ae register failed!\n");  
}
```

3.3.7 hi_mpi_isp_ae_lib_unreg_callback

函数功能

ISP 提供的 AE 库反注册的回调接口。

函数原型

```
hi_s32 hi_mpi_isp_ae_lib_unreg_callback(hi_vi_pipe vi_pipe, const  
hi_isp_3a_alg_lib *ae_lib)
```

参数说明

参数名	输入/输出	说明
vi_pipe	输入	VI PIPE号。 取值范围：[0, 12)。
ae_lib	输入	AE 库结构体指针。

返回值说明

- 0：成功
- 非0：失败，参见[ISP返回码](#)。

参考样例

```
hi_s32 ret = HI_SUCCESS;
ret = hi_mpi_isp_ae_lib_unreg_callback(vi_pipe, ae_lib);
if (HI_SUCCESS != ret) {
    printf("Hi_ae unregister failed!\n");
}
return ret;
```

3.3.8 hi_mpi_isp_awb_lib_reg_callback

函数功能

ISP 提供的 AWB 库注册的回调接口。

约束说明

- ISP提供统一的 AWB 算法库接口，初始化、运行、控制、销毁AWB算法库。使用昇腾内置的AWB算法库时，不需要关注此接口；使用用户自定义的AWB算法库时，需要调用此接口向 ISP 注册回调函数。
- 此接口不支持多进程操作。
- 最大支持 2 个AWB库注册。

函数原型

hi_s32 hi_mpi_isp_awb_lib_reg_callback(**hi_vi_pipe** vi_pipe, const **hi_isp_3a_alg_lib** *awb_lib, const **hi_isp_awb_register** *awb_register)

参数说明

参数名	输入/输出	说明
vi_pipe	输入	VI PIPE号。 取值范围：[0, 12)。
awb_lib	输入	AWB 库结构体指针。
awb_register	输入	AWB 库注册结构体指针。

返回值说明

- 0：成功
- 非0：失败，参见ISP返回码。

3.3.9 hi_mpi_isp_awb_lib_unreg_callback

函数功能

ISP 提供的 AWB 库反注册的回调接口。

约束说明

- 使用昇腾内置的AWB算法库时，不需要关注此接口；使用用户自定义的AWB算法库时，需要调用此接口向ISP取消注册回调函数。
- 此接口不支持多进程操作。

函数原型

```
hi_s32 hi_mpi_isp_awb_lib_unreg_callback(hi_vi_pipe vi_pipe, const hi_isp_3a_alg_lib *awb_lib)
```

参数说明

参数名	输入/输出	说明
vi_pipe	输入	VI PIPE号。 取值范围：[0, 12)。
awb_lib	输入	AWB 库结构体指针。

返回值说明

- 0：成功
- 非0：失败，参见ISP返回码。

3.3.10 hi_mpi_isp_set_bind_attr

函数功能

设置ISP库与 3A 库、sensor的绑定关系。

函数原型

```
hi_s32 hi_mpi_isp_set_bind_attr(hi_vi_pipe vi_pipe, const hi_isp_bind_attr *bind_attr)
```

约束说明

- 不是必须调用的接口，仅当注册多个AE/AWB库，并希望切换算法库时才需要调用。当注册多个AE/AWB库时，默认绑定的为最后一个注册的AE库和AWB库。
- 此接口不支持多进程操作。

参数说明

参数名	输入/输出	说明
vi_pipe	输入	VI PIPE号。 取值范围：[0, 12)。
bind_attr	输入	绑定结构体指针。

返回值说明

- 0：成功
- 非0：失败，参见[ISP返回码](#)。

3.3.11 hi_mpi_isp_get_bind_attr

函数功能

获取ISP库与3A库、sensor的绑定关系。

函数原型

[hi_s32](#) hi_mpi_isp_get_bind_attr([hi_vi_pipe](#) vi_pipe, [hi_isp_bind_attr](#) *bind_attr)

约束说明

此接口不支持多进程操作。

参数说明

参数名	输入/输出	说明
vi_pipe	输入	VI PIPE号。 取值范围：[0, 12)。
bind_attr	输出	绑定结构体指针。

返回值说明

- 0：成功
- 非0：失败，参见[ISP返回码](#)。

3.3.12 hi_mpi_isp_get_vd_time_out

函数功能

获取ISP中断信息。

约束说明

- 该接口表示获取ISP产生中断的相关信息，包括是否产生了中断，中断产生时的 当前 ISP 帧信息，中断是采用的帧开始中断。
- 在极端情况下（cpu 占用率高等），使用HI_ISP_VD_FE_END方式获取ISP中断信息并读取统计信息，会出现读取统计信息不及时的情况，因此建议使用HI_ISP_VD_FE_START方式获取ISP中断来读取统计信息。
- N个进程同时调用该接口获取同样的isp_vd_type时，每个进程获取到的中断信息为实际中断信息的(1/N)，比如在灌raw场景中会使用HI_ISP_VD_FE_START的方式来发送raw数据，如果此时其它进程再使用HI_ISP_VD_FE_START方式运行业务，会导致灌raw业务的帧率减半。

函数原型

hi_s32 hi_mpi_isp_get_vd_time_out(**hi_vi_pipe** vi_pipe, **hi_isp_vd_type** isp_vd_type, **hi_u32** milli_sec)

参数说明

参数名	输入/输出	说明
vi_pipe	输入	VI PIPE号。 取值范围：[0, 12)。
isp_vd_type	输入	场同步信号。
milli_sec	输入	超时时间，单位是毫秒。 - 在指定时间内，如果获取不到 ISP 中断，则函数返回。 - 当milli_sec设为 0 时，表示阻塞模式，程序一直等待，直到获取到 ISP 中断才返回。

返回值说明

- 0：成功
- 非0：失败，参见ISP返回码。

3.3.13 hi_mpi_isp_set_fmw_state

函数功能

设置ISP firmware状态。

函数原型

```
hi_s32 hi_mpi_isp_set_fmw_state(hi_vi_pipe vi_pipe, const hi_isp_fmw_state state)
```

参数说明

参数名	输入/输出	说明
vi_pipe	输入	VI PIPE号。 取值范围：[0, 12)。
state	输入	ISP firmware 状态。 - 当state值为 HI_ISP_FMW_STATE_FREEZE，ISP Firmware的 3A 算法，Sharpen算法，DRC算法，Crosstalk removal 算法，NR算法，去雾算法，去马赛克算法，黑电平算法，去FPN算法，WDR算法等会冻结，Sensor 的寄存器也会停止配置，并保持冻结前的值。 - 当state值为 HI_ISP_FMW_STATE_RUN，ISP firmware正常运行。

返回值说明

- 0：成功
- 非0：失败，参见ISP返回码。

3.3.14 hi_mpi_isp_get_fmw_state

函数功能

获取ISP firmware状态。

函数原型

```
hi_s32 hi_mpi_isp_get_fmw_state(hi_vi_pipe vi_pipe, hi_isp_fmw_state *state)
```

参数说明

参数名	输入/输出	说明
vi_pipe	输入	VI PIPE号。 取值范围：[0, 12)。
state	输出	ISP firmware状态。

返回值说明

- 0：成功
- 非0：失败，参见ISP返回码。

3.3.15 数据类型

本文档中变量，如未明确指定取值范围，则默认是数据类型对应的取值范围。例如 hi_u8数据类型的变量取值范围为[0, 255]。本文档中变量，如未明确指定数据精度，则默认是1。

3.3.15.1 hi_isp_bayer_raw_bit

说明

定义输入Bayer图像数据位宽。

定义

```
typedef enum {  
    HI_ISP_BAYER_RAW_BIT_8BIT    = 8,  
    HI_ISP_BAYER_RAW_BIT_10BIT   = 10,  
    HI_ISP_BAYER_RAW_BIT_12BIT   = 12,  
    HI_ISP_BAYER_RAW_BIT_14BIT   = 14,  
    HI_ISP_BAYER_RAW_BIT_16BIT   = 16,  
    HI_ISP_BAYER_RAW_BIT_BUTT  
} hi_isp_bayer_raw_bit;
```

成员

成员名称	描述
HI_ISP_BAYER_RAWBIT_8BIT	Bayer数据位宽：8bit。
HI_ISP_BAYER_RAWBIT_10BIT	Bayer数据位宽：10bit。
HI_ISP_BAYER_RAWBIT_12BIT	Bayer数据位宽：12bit。
HI_ISP_BAYER_RAWBIT_14BIT	Bayer数据位宽：14bit。
HI_ISP_BAYER_RAWBIT_16BIT	Bayer数据位宽：16bit。

注意事项

该格式可以从所使用sensor的DataSheet上获取，并和裁剪起始位置相关。

3.3.15.2 hi_isp_fmw_state

说明

定义ISPfirmware状态。

定义

```
typedef enum {  
    HI_ISP_FMW_STATE_RUN = 0,  
    HI_ISP_FMW_STATE_FREEZE,  
    HI_ISP_FMW_STATE_BUTT  
} hi_isp_fmw_state;
```

成员

成员名称	描述
HI_ISP_FMW_STATE_RUN	Firmware正常运行状态。
HI_ISP_FMW_STATE_FREEZE	Firmware冻结状态。
HI_ISP_FMW_STATE_BUTT	预留值。

3.3.15.3 hi_isp_wdr_mode

说明

定义ISP宽动态模式。

定义

```
typedef struct {  
    hi_wdr_mode wdr_mode;  
} hi_isp_wdr_mode;
```

成员

成员名称	描述
wdr_mode	宽动态模式。

3.3.15.4 hi_isp_module_ctrl

说明

定义ISP功能模块的控制。

定义

```
typedef union {  
    hi_u64 key;  
    struct {  
        hi_u64 bit_bypass_isp_d_gain : 1; /* RW:[0] */  
        hi_u64 bit_bypass_anti_false_color : 1; /* RW:[1] */  
        hi_u64 bit_bypass_crosstalk_removal : 1; /* RW:[2] */  
        hi_u64 bit_bypass_dpc : 1; /* RW:[3] */  
        hi_u64 bit_bypass_nr : 1; /* RW:[4] */  
        hi_u64 bit_bypass_dehaze : 1; /* RW:[5] */  
        hi_u64 bit_bypass_wb_gain : 1; /* RW:[6] */  
    };  
};
```

```
hi_u64 bit_bypass_mesh_shading : 1; /* RW;[7] */
hi_u64 bit_bypass_drc : 1; /* RW;[8] */
hi_u64 bit_bypass_demosaic : 1; /* RW;[9] */
hi_u64 bit_bypass_color_matrix : 1; /* RW;[10] */
hi_u64 bit_bypass_gamma : 1; /* RW;[11] */
hi_u64 bit_bypass_fswdr : 1; /* RW;[12] */
hi_u64 bit_bypass_ca : 1; /* RW;[13] */
hi_u64 bit_bypass_csc : 1; /* RW;[14] */
hi_u64 bit_reserved15_1 : 1; /* H; [15];
hi_u64 bit_bypass_sharpen : 1; /* RW;[16] */
hi_u64 bit_bypass_local_cac : 1; /* RW;[17] */
hi_u64 bit_bypass_global_cac : 1; /* RW;[18];
hi_u64 bit2_chn_select : 2; /* RW;[19:20] */
hi_u64 bit_bypass_ldci : 1; /* RW;[21] */
hi_u64 bit_bypass_pregamma : 1; /* RW;[22] */
hi_u64 bit_reserved23_1 : 1; /* H; [23] */
hi_u64 bit_bypass_ae_stat_fe : 1; /* RW;[24] */
hi_u64 bit_bypass_ae_stat_be : 1; /* RW;[25] */
hi_u64 bit_bypass_mg_stat : 1; /* RW;[26] */
hi_u64 bit_bypass_de : 1; /* RW;[27] */
hi_u64 bit_bypass_af_stat_fe : 1; /* RW;[28] */
hi_u64 bit_bypass_af_stat_be : 1; /* RW;[29] */
hi_u64 bit_bypass_awb_stat : 1; /* RW;[30] */
hi_u64 bit_bypass_clut : 1; /* RW;[31] */
hi_u64 bit_bypass_hlc : 1; /* RW;[32] */
hi_u64 bit_bypass_edge_mark : 1; /* RW;[33] */
};
hi_u64 bit_bypass_rgbir : 1; /* RW;[34] */
hi_u64 bit_reserved29 : 29; /* H; [35:63] */
} hi_isp_module_ctrl;
```

成员

成员名称	描述
key	结构体枚举的整型值。
bit_bypass_isp_d_gain	旁路数字增益。
bit_bypass_anti_false_color	旁路去伪彩功能。
bit_bypass_crosstalk_removal	旁路Crosstalk Removal。
bit_bypass_dpc	旁路坏点校正。
bit_bypass_nr	旁路去噪。
bit_bypass_dehaze	旁路去雾。
bit_bypass_wb_gain	旁路白平衡增益和偏移量。
bit_bypass_mesh_shading	旁路镜头阴影校正。
bit_bypass_drc	旁路动态范围压缩。
bit_bypass_demosaic	旁路去马赛克模块。
bit_bypass_color_matrix	旁路颜色矩阵。
bit_bypass_gamma	旁路伽马校正。
bit_bypass_fswdr	旁路多帧合成宽动态范围WDR。

成员名称	描述
bit_bypass_ca	旁路颜色调整模块CA。
bit_bypass_csc	旁路色彩空间转换CSC。
bit_bypass_sharpen	旁路锐化Sharpen。
bit_bypass_local_cac	旁路色差Local CAC。
bit2_chn_select	宽动态范围WDR模式主路数据来源，一般在旁路多帧合成WDR模块后用于debug。 0：主路数据来源于超短帧； 1：主路数据来源于短帧； 2：主路数据来源于中帧； 3：主路数据来源于长帧。
bit_bypass_ldci	旁路局域自动对比度增强Local DCI。
bit_bypass_pregamma	旁路预伽马PreGamma。
bit_bypass_ae_stat_fe	旁路位于FE的AE统计信息。
bit_bypass_ae_stat_be	旁路位于BE的AE统计信息。
bit_bypass_mg_stat	旁路MG统计信息。
bit_bypass_de	暂不支持。
bit_bypass_af_stat_fe	暂不支持。
bit_bypass_af_stat_be	旁路位于BE的AF统计信息。
bit_bypass_awb_stat	旁路AWB统计信息。
bit_bypass_clut	旁路CLUT。
bit_bypass_hlc	旁路高光遮蔽HLC
bit_bypass_edge_mark	旁路EDGE_MARK
bit_bypass_rgbir	暂不支持。
bit_reserved29	预留参数。

注意事项

WDR模式下，开关WDR模块的使能，图像会有几帧颜色表现异常。

3.3.15.5 hi_isp_vd_type

说明

定义ISP场同步信号。

定义

```
typedef enum {  
    HI_ISP_VD_FE_START = 0,  
    HI_ISP_VD_FE_END,  
    HI_ISP_VD_BE_END,  
    HI_ISP_VD_BUTT  
} hi_isp_vd_type;
```

成员

成员名称	描述
HI_ISP_VD_FE_START	FE 帧起始。
HI_ISP_VD_FE_END	FE 帧结束。
HI_ISP_VD_BE_END	BE 帧结束。
HI_ISP_VD_BUTT	预留值。

注意事项

在线和并行模式下不支持HI_ISP_VD_BE_END方式。

3.3.15.6 hi_isp_cmos_lsc

说明

定义镜头阴影校正LSC的CMOS参数。

定义

```
typedef struct {  
    hi_u16 mesh_strength;  
    hi_u8 mesh_scale;  
    hi_isp_lsc_cabli_table lsc_calib_table[HI_ISP_MLSC_GAIN_LUT_NUM];  
    hi_isp_bnr_lsc_calib_table bnr_lsc_calib_table;  
} hi_isp_cmos_lsc;
```

成员

成员名称	描述
mesh_strength	用来全局纠正Mesh Shading标定后的强度。 取值范围[0, 65535]。
mesh_scale	增益表精度控制参数。 取值范围： [0, 7]
lsc_calib_table	校正表格。 #define HI_ISP_MLSC_Y_HALF_GRID_NUM ((HI_ISP_LSC_GRID_ROW - 1) / 2)

成员名称	描述
bnr_lsc_calib_table	Bayer NR模块中所用的LSC表格。

3.3.15.7 hi_isp_acs_y_shading_lut

说明

定义Auto Color Shading亮度分量上的校正强度表，也就是Gr/Gb分量的校正强度。

定义

```
typedef struct {  
    hi_u16 g_param_high_ct[HI_ISP_LSC_GRID_POINTS];  
    hi_u16 g_param_low_ct[HI_ISP_LSC_GRID_POINTS];  
} hi_isp_acs_y_shading_lut;
```

成员

成员名称	描述
g_param_high_ct	Gr/Gb分量上的校正强度表，校正强度较大。 #define HI_ISP_LSC_GRID_COL 33 #define HI_ISP_LSC_GRID_ROW 33 #define HI_ISP_LSC_GRID_POINTS (HI_ISP_LSC_GRID_COL * HI_ISP_LSC_GRID_ROW)
g_param_low_ct	Gr/Gb分量上的校正强度表，校正强度较小。

注意事项

算法根据场景从g_param_high_ct和g_param_low_ct两张表中进行插值。

3.3.15.8 hi_isp_acs_color_shading_lut

说明

定义Auto Color Shading颜色分量上的Lut表，算法会根据R/B分量上的Lut表，动态生成适合当前场景的Lut表。

定义

```
typedef struct {  
    hi_float avg_rg_map[HI_ISP_LSC_GRID_POINTS];  
    hi_float avg_bg_map[HI_ISP_LSC_GRID_POINTS];  
    hi_float prof_rg_map[HI_ISP_LSC_GRID_POINTS];  
    hi_float prof_bg_map[HI_ISP_LSC_GRID_POINTS];  
} hi_isp_acs_color_shading_lut;
```

成员

成员名称	描述
avg_rg_map[HI_ISP_LSC_GRID_POINTS]	R分量上的Color Shading表。 #define HI_ISP_LSC_GRID_COL 33 #define HI_ISP_LSC_GRID_ROW 33 #define HI_ISP_LSC_GRID_POINTS (HI_ISP_LSC_GRID_COL * HI_ISP_LSC_GRID_ROW)
avg_bg_map	B分量上的Color Shading表。
prof_rg_map	R分量上的Color Shading表。
prof_bg_map	B分量上的Color Shading表。

3.3.15.9 hi_isp_acs_calib_param

说明

定义Auto Color Shading的标定参数，用标定工具生成。

定义

```
typedef struct {  
    hi_s16 light_index[HI_ISP_ACS_LIGHT_NUM * HI_ISP_ACS_CHN_NUM];  
    hi_float model_ar_min;  
    hi_float model_ar_step;  
    hi_float model_ab_min;  
    hi_float model_ab_step;  
    hi_s16 light_type_g_high;  
    hi_s16 light_type_g_low;  
} hi_isp_acs_calib_param;
```

成员

成员名称	描述
light_index	标定的光源在算法模型当中的光源坐标。 #define HI_ISP_ACS_LIGHT_NUM 32 #define HI_ISP_ACS_CHN_NUM 2
model_ar_min	标定得到的算法模型参数。
model_ar_step	标定得到的算法模型参数。
model_ab_min	标定得到的算法模型参数。
model_ab_step	标定得到的算法模型参数。
light_type_g_high	对于g_param_high_ct表对应的光源坐标。
light_type_g_low	对于g_param_low_ct表对应的光源坐标。

3.3.15.10 hi_isp_cmos_acs

说明

定义Auto Color Shading的CMOS参数。

定义

```
typedef struct {
    hi_bool          enable;
    hi_u16           acs_shading_str;
    hi_isp_acs_calib_param acs_calib_param;
    hi_isp_acs_y_shading_lut acs_y_shading_lut;
    hi_isp_acs_color_shading_lut acs_color_shading_lut;
} hi_isp_cmos_acs;
```

成员

成员名称	描述
enable	acs模块使能。
acs_shading_str	acs算法luma shading校正强度。
acs_calib_param	Auto Color Shading的标定参数。
acs_y_shading_lut	Auto Color Shading亮度分量上的校正强度表，也就是Gr/Gb分量的校正强度。
acs_color_shading_lut	Auto Color Shading颜色分量上的Lut表，算法会根据R/B分量上的Lut表，动态生成适合当前场景的Lut表。

注意事项

- 增益表的默认配置与hi_isp_cmos_alg_key中的bit1_acs标志位有关，如果bit1_acs=1，则采用cmos_ex.h中的配置值作为默认值；否则默认配置都为0。
- ACS模块的otp通过hi_isp_cmos_lsc中的lsc_lut.lsc_gain_lut接口实现，其中lsc_lut.lsc_gain_lut[0]配置为golden sample在D50下标定的表，lsc_lut.lsc_gain_lut[1]配置为当前镜头模组在D50下标定的表，可以解决镜头的一致性问题，模组与golden之间的差异越小，校正的效果越好。也可以在ISP启动后配置接口hi_mpi_isp_set_mesh_shading_gain_lut_attr中的lsc_gain_lut[0]和lsc_gain_lut[1]，用法与上面描述一致。

3.3.15.11 hi_isp_lsc_cabli_table

说明

定义LSC的校正参数。

定义

```
typedef struct {
    hi_u16 r_gain[HI_ISP_LSC_GRID_POINTS];
}
```



```
hi_u16 gr_gain[HI_ISP_LSC_GRID_POINTS];  
hi_u16 gb_gain[HI_ISP_LSC_GRID_POINTS];  
hi_u16 b_gain[HI_ISP_LSC_GRID_POINTS];  
} hi_isp_lsc_cabli_table;
```

成员

成员名称	描述
r_gain[HI_ISP_LSC_GRID_POINTS]	R通道校正增益表。取值范围：[0, 1023] #define HI_ISP_LSC_GRID_COL 33 #define HI_ISP_LSC_GRID_ROW 33 #define HI_ISP_LSC_GRID_POINTS (HI_ISP_LSC_GRID_COL * HI_ISP_LSC_GRID_ROW)
gr_gain	Gr通道校正增益表。取值范围：[0, 1023]
gb_gain	Gb通道校正增益表。取值范围：[0, 1023]
b_gain	B通道校正增益表。取值范围：[0, 1023]

3.3.15.12 hi_isp_cmos_expander_point

说明

定义Expander结构体。

定义

```
typedef struct {  
    hi_u16 x;  
    hi_u32 y;  
} hi_isp_cmos_expander_point;
```

成员

成员名称	描述
x	x轴坐标值。 取值范围：[0x0,0x100]。
y	y轴坐标值。 取值范围：[0x0, 0x1000000]。

3.3.15.13 hi_isp_expander_attr

说明

定义Expander结构体。

定义

```
typedef struct {
    hi_bool enable;
    hi_u8 bit_depth_in; /* RW;Range:[0xC,0x14];Format:5.0;The Bit depth of input */
    hi_u8 bit_depth_out; /* RW;Range:[0xC,0x14];Format:5.0;The Bit depth of output */
    hi_u16 expander_point_num; /* Range:[1, 256] */
    hi_isp_cmos_expander_point expander_point[HI_ISP_EXPANDER_POINT_NUM_MAX];
} hi_isp_expander_attr;
```

成员

成员名称	描述
enable	expander模块的使能。 - HI_FALSE：关闭； - HI_TRUE：使能。 只能在sensor built-in模式下使能expander。
bit_depth_in	输入数据位宽。 取值范围：[0xC,0x14]，只能配置为偶数。
bit_depth_out	输出数据位宽。 取值范围：[0xC,0x14]，只能配置为偶数。
expander_point_num	拐点坐标的数目。取值范围：[1,256]。
expander_point	解压的拐点（包括横纵坐标）。 坐标必须是单调递增的。 #define HI_ISP_EXPANDER_POINT_NUM_MAX (HI_ISP_EXPANDER_NODE_NUM - 1)

注意事项

在sensor手册中会给出sensor在sensor-built-in模式下内部压缩时使用的拐点配置，需要将这几个拐点配置按照对应的原则进行转换，然后配置到expander_point中即可。

- expander_point的横坐标x需要根据sensor压缩曲线转换到[0,256]之间（8bit），例如sensor压缩输出的后有效数据位宽是12bit，则需要将sensor压缩曲线拐点的纵坐标右移4bit得到expander_point的横坐标x；
- expander_point的纵坐标y需要根据sensor压缩曲线转换到[0,1048576]之间（20bit），例如sensor合成有效数据未压缩之前有效位宽是16bit，则需要将sensor压缩曲线的拐点的横坐标左移4bit，得到expander_point的纵坐标y。
- 不必配置(0,0)的拐点坐标，代码内部在进行插值时，会自动将(0,0)作为第一个拐点坐标

3.3.15.14 hi_isp_cmos_clut

说明

定义CLUT结构体。

定义

```
typedef struct {  
    hi_bool enable;  
    hi_u32 gain_r;  
    hi_u32 gain_g;  
    hi_u32 gain_b;  
    hi_isp_clut_lut clut_lut;  
} hi_isp_cmos_clut;
```

成员

成员名称	描述
enable	CLUT使能。 取值范围： - HI_FALSE：关闭； - HI_TRUE：使能。
gain_r	R通道增益，取值范围：[0,4095]。
gain_g	G通道增益，取值范围：[0,4095]。
gain_b	B通道增益，取值范围：[0,4095]。
clut_lut	定义CLUT的查找表。

3.3.15.15 hi_isp_cmos_drc

说明

定义DRC的CMOS参数。

定义

```
typedef struct {  
    hi_bool enable;  
    hi_op_mode op_type;  
  
    hi_u16 manual_strength;  
    hi_u16 auto_strength;  
  
    hi_u8 spatialflt_coef;  
    hi_u8 rangeflt_coef;  
    hi_u8 contrast_control;  
    hi_s8 detail_adjust_factor;  
    hi_u8 range_ada_max;  
  
    hi_u8 grad_rev_max;  
    hi_u8 grad_rev_thr;  
  
    hi_u8 bright_gain_lmt;  
    hi_u8 bright_gain_lmt_step;  
    hi_u8 dark_gain_lmt_y;  
    hi_u8 dark_gain_lmt_c;  
  
    hi_u8 purple_reduction_strength;  
    hi_u8 local_mixing_bright_max;  
    hi_u8 local_mixing_bright_min;
```

```
hi_u8 local_mixing_bright_thr;  
hi_s8 local_mixing_bright_slo;  
hi_u8 local_mixing_dark_max;  
hi_u8 local_mixing_dark_min;  
hi_u8 local_mixing_dark_thr;  
hi_s8 local_mixing_dark_slo;  
hi_u16 color_correction_lut[HI_ISP_DRC_CC_NODE_NUM];  
hi_u16 tone_mapping_value[HI_ISP_DRC_TM_NODE_NUM];  
  
hi_u8 asymmetry;  
hi_u8 second_pole;  
hi_u8 stretch;  
hi_u8 compress;  
  
hi_u8 curve_sel;  
}  
hi_isp_cmos_drc;
```

成员

成员名称	描述
enable	使能开关。 取值范围： HI_FALSE：关闭； HI_TRUE：使能。
op_type	DRC工作类型。 取值范围： HI_OP_MODE_AUTO：自动； HI_OP_MODE_MANUAL：手动。
manual_strength	手动模式下DRC的强度，值越大，阴影和高光之间的差分增益就越高。 取值范围：[0x0, 0x3FF]。
auto_strength	自动模式DRC的强度，参数可以读写，值越大，整体强度越大。 取值范围：[0x0, 0x3FF]。
spatialflt_coef	Filter空域滤波系数。值越大，运动halo越不明显，细节越少，值越小，细节表现越好，运动halo越明显。 取值范围：[0x0, 0x5]
rangeflt_coef	Filter值域滤波系数。值越大，halo越明显；值越小，halo表现越好，但是在强边缘处可能会出现描边现象。 取值范围：[0x0, 0xA]
contrast_control	局部对比度控制，参数效果与图像亮度分布相关，一般场景下建议值为6到10之间。 取值范围：[0x0, 0xF]
detail_adjust_factor	Filter细节微调系数。值越大，整体细节越强。 取值范围：[-0xF, 0xF]

成员名称	描述
range_ada_max	控制值域滤波系数的自适应范围。实际生效滤波系数在 [rangeflt_coef, MIN(rangeflt_coef+range_ada_max, 0xA)] 范围内。调大该值有助于减弱边线，但同时可能造成背光物体细节损失。 取值范围：[0x0, 0x8]
grad_rev_max	去边线强度。值越大，边线减弱越明显，但是可能会带来细节损失。 取值范围：[0x0, 0x40]
grad_rev_thr	边线检测阈值。值越大，边线减弱越明显，但是可能会带来细节损失。 取值范围：[0x0, 0x80]
bright_gain_lmt	亮区亮度增益限制目标值；值越大，限制越大。 取值范围：[0x0, 0xF]
bright_gain_lmt_step	亮区亮度增益限制自适应步长；值越小，限制越大。 取值范围：[0x0, 0xF]
dark_gain_lmt_y	暗区亮度增益限制；值越大，限制越大。 取值范围：[0x0, 0x85]
dark_gain_lmt_c	暗区色度增益限制；值越大，限制越大。 取值范围：[0x0, 0x85]
purple_reduction_strength	紫边校正强度，值越大，紫边校正强度越大。 取值范围：[0x0, 0x80]
local_mixing_brightness_max	线性细节增强：用来控制正向细节的增益的最大值。 取值范围：[0x0, 0x80]
local_mixing_brightness_min	线性细节增强：用来控制正向细节的增益的最小值。 取值范围：[0x0, 0x40]
local_mixing_brightness_thr	线性细节增强：用来控制正向细节增益随亮度自适应的阈值。当像素亮度超过该阈值，增益将从最大值向最小值过渡（或反之，取决于斜率参数的符号）。 取值范围：[0x0, 0xFF]
local_mixing_brightness_slo	线性细节增强：用来控制正向细节增益随亮度自适应的斜率。绝对值越大，从最小值到最大值（或反之）过渡越快。 取值范围：[-0x7, 0x7]
local_mixing_dark_max	线性细节增强：用来控制负向细节的增益最大值。 取值范围：[0x0, 0x80]
local_mixing_dark_min	线性细节增强：用来控制负向细节的增益最小值。 取值范围：[0x0, 0x40]

成员名称	描述
local_mixing_dark_thr	线性细节增强：用来控制负向细节增益随亮度自适应的阈值。当像素亮度超过该阈值，增益将从最小值向最大值过渡（或反之，取决于斜率参数的符号）。 取值范围：[0x0, 0xFF]
local_mixing_dark_slo	线性细节增强：用来控制负向细节增益随亮度自适应的斜率。绝对值越大，从最小值到最大值（或反之）过渡越快。 取值范围：[-0x7, 0x7]
color_correction_lut	颜色校正系数LUT，值越小，饱和度越低。 取值范围：[0x0, 0x400] #define HI_ISP_DRC_CC_NODE_NUM 33
tone_mapping_value	用户自定义Tone Mapping曲线LUT。 取值范围：[0x0, 0xFFFF] #define HI_ISP_DRC_TM_NODE_NUM 200
asymmetry	用于生成Asymmetry Curve，值越小，暗区亮度拉伸越明显。 取值范围：[1, 30]
second_pole	用于生成Asymmetry Curve，值越小，亮区亮度压制越强，有利于保留高亮细节。 取值范围：[150, 210]
stretch	用于生成Asymmetry Curve，值越小，整体亮度越亮。 取值范围：[30, 60]
compress	用于生成Asymmetry Curve，值越小，整体亮度越亮。 取值范围：[100, 200]
curve_sel	DRC Tone mapping曲线选择。 取值范围： 0：Asymmetry curve； 1：用户自定义曲线

3.3.15.16 hi_isp_cmos_demosaic

说明

定义demosaic的CMOS参数。

定义

```
typedef struct {  
    hi_bool enable;  
    hi_u8 non_dir_str[HI_ISP_AUTO_ISO_NUM];  
    hi_u8 non_dir_mf_detail_ehc_str[HI_ISP_AUTO_ISO_NUM];  
    hi_u8 non_dir_hf_detail_ehc_str[HI_ISP_AUTO_ISO_NUM];  
    hi_u8 detail_smooth_range[HI_ISP_AUTO_ISO_NUM];  
} hi_isp_cmos_demosaic;
```

成员

成员名称	描述
enable	Demosaic模块使能。取值范围：[0, 1]。 0：禁止； 1：使能。
non_dir_str[HI_ISP_AUTO_ISO_NUM]	无方向插值强度，值越大，无方向插值所占比重越大。 取值范围：[0x0,0xFF]。 #define HI_ISP_AUTO_ISO_NUM 16
non_dir_mf_detail_ehc_str	无方向中频纹理增强强度，值越大，无方向中频纹理细节增强越强，对噪声同样有增强。 取值范围：[0x0, 0x7f]。
non_dir_hf_detail_ehc_str	无方向高频纹理增强强度，值越大，无方向高频纹理细节增强越强，对噪声均匀性有提升。 取值范围：[0x0,0x10]。
detail_smooth_range	细节平滑范围，值越大，做平滑处理的细节范围越大，能够抑制更多伪细节。 取值范围：[0x1, 0x7]。

3.3.15.17 hi_isp_cmos_pregamma

说明

定义pregamma的CMOS参数。

定义

```
typedef struct {  
    hi_bool enable;  
    hi_u32 pregamma_lut[HI_ISP_PREGAMMA_NODE_NUM];  
} hi_isp_cmos_pregamma;
```

成员

成员名称	描述
enable	PreGamma使能开关。 取值范围： - HI_FALSE：关闭 - HI_TRUE：使能
pregamma_lut	257 段LUT表，用来表示PreGamma输出值大小。 取值范围：[0x0, 0xFFFFF]。 #define HI_ISP_PREGAMMA_NODE_NUM 257

3.3.15.18 hi_isp_cmos_gamma

说明

定义gamma的CMOS参数。

定义

```
typedef struct {  
    hi_u16 gamma_lut[HI_ISP_GAMMA_EXT_NODE_NUMBER];  
} hi_isp_cmos_gamma;
```

成员

成员名称	描述
gamma_lut	1024 段LUT表，用来表示输入输出值大小。 取值范围：[0x0, 0xFFFF] #define HI_ISP_GAMMA_EXT_NODE_NUMBER 1025

3.3.15.19 hi_isp_cmos_sharpen_manual

说明

定义sharpen manual档的CMOS参数。

定义

```
typedef struct {  
    hi_u8 luma_wgt[HI_ISP_SHARPEN_LUMA_NUM];  
    hi_u16 texture_strength[HI_ISP_SHARPEN_GAIN_NUM];  
    hi_u16 edge_strength[HI_ISP_SHARPEN_GAIN_NUM];  
    hi_u16 texture_freq;  
    hi_u16 edge_freq;  
    hi_u8 over_shoot;  
    hi_u8 under_shoot;  
    hi_u8 shoot_sup_strength;  
    hi_u8 shoot_sup_adj;  
    hi_u8 detail_ctrl;  
    hi_u8 detail_ctrl_thr;  
    hi_u8 edge_filt_str;  
    hi_u8 edge_filt_max_cap;  
    hi_u8 r_gain;  
    hi_u8 g_gain;  
    hi_u8 b_gain;  
    hi_u8 skin_gain;  
    hi_u16 max_sharp_gain;  
} hi_isp_cmos_sharpen_manual;
```


成员

成员名称	描述
<code>luma_wgt[HI_ISP_SHARPEN_LUMA_NUM]</code>	亮度锐化权重。 满量程0-255的亮度被32个等分点平均分为32段亮度区间，每一段亮度区间对应一个亮度权重。比如0-7的亮度区间的权重是<code>luma_wgt[0]</code>，8-15的亮度区间的权重是<code>luma_wgt[1]</code>，依次类推。如图3-7所示。值越大，图像锐化程度越高，反之，越弱。 取值范围：[0, 127]，建议值 127。 #define HI_ISP_SHARPEN_LUMA_NUM32
<code>texture_strength[HI_ISP_SHARPEN_GAIN_NUM]</code>	该参数用于设置图像无方向的细节纹理的锐度。 该值越大，无方向的细节纹理的清晰度越高。 该参数是一个长度为32的数组，是一个32段的连续的强度曲线，如图3-8所示。 取值范围：[0, 4095]，建议值300。 #define HI_ISP_SHARPEN_GAIN_NUM32
<code>edge_strength[HI_ISP_SHARPEN_GAIN_NUM]</code>	该参数用于设置图像带方向的边缘的锐度。 该值越大，带方向的边缘的锐度越高。 该参数是一个长度为32的数组，是一个32段的连续的强度曲线，如图3-9所示。 取值范围：[0, 4095]，建议值400。
<code>texture_freq</code>	该参数用于设置图像的无方向细节纹理增强的频率。 该值越大，细节纹理的增强就越偏向于高频增强，细节纹理就越细碎。反之，该值越小，细节纹理就越粗越圆润。 <code>texture_freq</code>对应于强度参数<code>texture_strength</code>。 <code>texture_freq</code>越大，图像的细节纹理就越细碎，该值过大，会导致图像的细节纹理过于细碎而不自然，甚至感觉模糊。 取值范围：[0, 4095]，建议值128。
<code>edge_freq</code>	该参数用于设置图像的带方向的边缘增强频率。 该值越大，边缘的增强就越偏向于高频增强，图像的边缘就越纤薄越窄。反之，该值越小，边缘就越粗越圆润。 <code>edge_freq</code>对应于强度参数<code>edge_strength</code>。 <code>edge_freq</code>越大，图像的边缘就越纤薄越窄，该值过大，会导致图像的边缘过于纤薄而出现虚边现象。 取值范围：[0, 4095]，建议值96。
<code>over_shoot</code>	设置图像的overshoot（锐化后的白边白点）的强度。 该值越小，锐化后的白边白点越弱，清晰度也会下降。该值过小，图像会呈油画效果。 取值范围：[0, 127]，建议值100。

成员名称	描述
under_shoot	设置图像的undershoot（锐化后的黑边黑点）的强度。 该值越小，锐化后的黑边黑点越弱，清晰度也会下降。该值过小，图像会呈油画效果。 取值范围：[0, 127]，建议值127。
shoot_sup_strength	设置图像锐化后的overshoot和undershoot的抑制强度。用于在保证清晰度不明显下降的前提下，抑制锐化后的图像的overshoot和undershoot的宽度和幅度。 该值越大，锐化后的图像的overshoot和undershoot的宽度越窄、强度越小。该值变大，理论上不会影响图像的清晰度，只是黑白边变窄以后，会减弱人眼的锐度感受。 该参数需要和shoot_sup_adj配合使用。 取值范围：[0, 255]，建议值8。
shoot_sup_adj	用于对图像锐化后的overshoot和undershoot的抑制强度进行调节。 该参数配合shoot_sup_strength使用，用于调节shoot_sup_strength作用的区域范围。该值越小，则越多的纹理区域的shoot会被shoot_sup_strength抑制；该值越大，则只有很强的边缘的shoot会被shoot_sup_strength抑制，纹理区域的shoot不会被抑制。 取值范围：[0, 15]，建议值9。
detail_ctrl	用于控制图像的细节纹理区域的shoot强度，shoot越大，细节纹理区的清晰度越高。 • 若该值等于128，则图像的细节纹理区域的shoot强度和大边缘的shoot强度一致，都分别等于over_shoot和under_shoot。 • 若该值大于128，则图像的细节纹理的shoot强度大于大边缘，大边缘的shoot强度分别等于over_shoot和under_shoot。 • 若该值小于128，则图像的细节纹理的shoot强度小于大边缘，大边缘的shoot强度分别等于over_shoot和under_shoot。 取值范围：[0, 255]。
detail_ctrl_thr	图像的细节纹理区的shoot强度的控制阈值。 该值配合detail_ctrl使用，用于区分detail_ctrl所控制shoot的纹理区和边缘，也即纹理区和边缘的区分阈值。小于该值的区域为纹理区，该纹理区域的shoot会被detail_ctrl单独控制，而大于该阈值的边缘区域的shoot依然等于over_shoot和under_shoot。 取值范围：[0, 255]，建议值160。

成员名称	描述
edge_filt_str	边缘滤波强度调试参数，实现图像锐化边缘的范围和边缘平滑强度的控制。 该值越大，判为边缘的区域越多、也越宽，edge_strength起作用的图像边缘就越多，而且，该值越大，沿着边缘方向的平滑滤波强度也越大，边缘就越平滑。反之，判为边缘的区域越少、也越窄，edge_strength起作用的图像区域越少，边缘平滑就越弱。 取值范围：[0, 63]，建议值53。
edge_filt_max_cap	边缘滤波强度范围的调试参数，该值越大，边缘滤波的最大强度也最大，edge_filt_strength的可调试范围也越大。 一般建议该值大小控制30以内。 取值范围：[0, 47]，建议值18。
r_gain	深红色区域的锐化增益控制参数。该值越大，则深红色区域的锐化强度越大。 取值范围：[0, 31]，建议值28。
g_gain	绿色区域的锐化增益控制参数。该值越大，则绿色区域的锐化强度越大。 取值范围：[0, 255]，建议值32。
b_gain	深蓝色区域的锐化增益控制参数。该值越大，则深蓝色区域的锐化强度越大。 取值范围：[0, 31]，建议值 28。
skin_gain	肤色区域的锐化增益控制参数。该值越大，则肤色区域的锐化强度越大。 取值范围：[0, 31]，建议值23。
max_sharp_gain	图像锐化的最大增益限制值参数。该值越大，图像的锐化幅度越大，反之，锐化幅度越小。适当的调小该参数，可以减少图像的过锐化，减少图像锐化后的黑白点。 取值范围：[0, 0x7FF]，建议值0xA0。

3.3.15.20 hi_isp_cmos_sharpen_auto

说明

定义sharpen auto档的CMOS参数。

定义

```
typedef struct {  
    hi_u8  luma_wgt[HI_ISP_SHARPEN_LUMA_NUM][HI_ISP_AUTO_ISO_NUM];  
    hi_u16 texture_strength[HI_ISP_SHARPEN_GAIN_NUM][HI_ISP_AUTO_ISO_NUM];  
    hi_u16 edge_strength[HI_ISP_SHARPEN_GAIN_NUM][HI_ISP_AUTO_ISO_NUM];  
    hi_u16 texture_freq[HI_ISP_AUTO_ISO_NUM];  
    hi_u16 edge_freq[HI_ISP_AUTO_ISO_NUM];  
    hi_u8  over_shoot[HI_ISP_AUTO_ISO_NUM];  
};
```

```
hi_u8 under_shoot[HI_ISP_AUTO_ISO_NUM];  
hi_u8 shoot_sup_strength[HI_ISP_AUTO_ISO_NUM];  
hi_u8 shoot_sup_adj[HI_ISP_AUTO_ISO_NUM];  
hi_u8 detail_ctrl[HI_ISP_AUTO_ISO_NUM];  
hi_u8 detail_ctrl_thr[HI_ISP_AUTO_ISO_NUM];  
hi_u8 edge_filt_str[HI_ISP_AUTO_ISO_NUM];  
hi_u8 edge_filt_max_cap[HI_ISP_AUTO_ISO_NUM];  
hi_u8 r_gain[HI_ISP_AUTO_ISO_NUM];  
hi_u8 g_gain[HI_ISP_AUTO_ISO_NUM];  
hi_u8 b_gain[HI_ISP_AUTO_ISO_NUM];  
hi_u8 skin_gain[HI_ISP_AUTO_ISO_NUM];  
hi_u16 max_sharp_gain[HI_ISP_AUTO_ISO_NUM];  
} hi_isp_cmos_sharpen_auto;
```

成员

成员名称	描述
<code>luma_wgt[HI_ISP_SHARPEN_LUMA_NUM][HI_ISP_AUTO_ISO_NUM]</code>	亮度锐化权重。 满量程0-255的亮度被32个等分点平均分为32段亮度区间，每一段亮度区间对应一个亮度权重。比如0-7的亮度区间的权重是<code>luma_wgt[0]</code>，8-15的亮度区间的权重是<code>luma_wgt[1]</code>，依次类推。如图3-7所示。值越大，图像锐化程度越高，反之，越弱。 取值范围：[0, 127]，建议值127。 <pre>#define HI_ISP_SHARPEN_LUMA_NUM 32</pre>
<code>texture_strength[HI_ISP_SHARPEN_GAIN_NUM][HI_ISP_AUTO_ISO_NUM]</code>	该参数用于设置图像无方向的细节纹理的锐度。 该值越大，无方向的细节纹理的清晰度越高。该参数是一个<code>HI_ISP_SHARPEN_GAIN_NUM=32</code>的数组，是一个32段的连续的强度曲线，如图3-8所示。 取值范围：[0, 4095]，建议值300。 <pre>#define HI_ISP_SHARPEN_GAIN_NUM 32</pre>
<code>edge_strength[HI_ISP_SHARPEN_GAIN_NUM][HI_ISP_AUTO_ISO_NUM]</code>	该参数用于设置图像带方向的边缘的锐度。 该值越大，带方向的边缘的锐度越高。该参数是一个<code>HI_ISP_SHARPEN_GAIN_NUM=32</code>的数组，是一个32段的连续的强度曲线，如图3-9所示。取值范围：[0, 4095]，建议值400。
<code>texture_freq[HI_ISP_AUTO_ISO_NUM]</code>	该参数用于设置图像的无方向细节纹理的增强频率。 该值越大，细节纹理的增强就越偏向于高频增强，细节纹理就越细碎。反之，该值越小，细节纹理就越粗越圆润。 <code>texture_freq</code>对应于强度参数<code>texture_strength</code>。<code>texture_freq</code>越大，图像的细节纹理就越细碎，该值过大，会导致图像的细节纹理过于细碎而不自然，甚至感觉模糊。 取值范围：[0, 4095]，建议值128。

成员名称	描述
edge_freq[HI_ISP_AUTO_ISO_NUM]	该参数用于控制图像的带方向的边缘增强频段。 该值越大，边缘的增强就越偏向于高频增强，图像的边缘就越纤薄越窄。反之，该值越小，边缘就越粗越圆润。 edge_freq对应于强度参数edge_strength。edge_freq越大，图像的边缘就越纤薄越窄，该值过大，会导致图像的边缘过于纤薄而出现虚边现象。 取值范围：[0, 4095]，建议值96。
over_shoot[HI_ISP_AUTO_ISO_NUM]	设置图像的overshoot（锐化后的白边白点）的强度。 该值越小，锐化后的白边白点越弱，清晰度也会下降。该值过小，图像会呈油画效果。 取值范围：[0, 127]，建议值100。
under_shoot[HI_ISP_AUTO_ISO_NUM]	设置图像的undershoot（锐化后的黑边黑点）的强度。 该值越小，锐化后的黑边黑点越弱，清晰度也会下降。该值过小，图像会呈油画效果。 取值范围：[0, 127]，建议值127。
shoot_sup_strength[HI_ISP_AUTO_ISO_NUM]	该参数用于设置图像锐化后的overshoot和undershoot的抑制强度。用于在保证清晰度不明显下降的前提下，抑制锐化后的图像的overshoot和undershoot的宽度和幅度。 该值越大，锐化后的图像的overshoot和undershoot的宽度越窄、强度越小。该值变大，理论上不会影响图像的清晰度，只是黑白边变窄以后，会减弱人眼的锐度感受。 该参数需要和shoot_sup_adj配合使用。 取值范围：[0, 255]，建议值8。
shoot_sup_adj[HI_ISP_AUTO_ISO_NUM]	该参数用于对图像锐化后的overshoot和undershoot的抑制强度进行调节。 该参数配合shoot_sup_strength使用，用于调节shoot_sup_strength作用的区域范围。该值越小，则越多的纹理区域的shoot会被shoot_sup_strength抑制；该值越大，则只有很强的边缘的shoot会被shoot_sup_strength抑制，纹理区域的shoot不会被抑制。 取值范围：[0, 15]，建议值9。

成员名称	描述
detail_ctrl[HI_ISP_AUTO_ISO_NUM]	该参数用于控制图像的细节纹理区域的shoot强度，shoot越大，细节纹理区的清晰度越高。 - 若该值等于128，则图像的细节纹理区域的shoot强度和大边缘的shoot强度一致，都分别等于over_shoot和under_shoot。 - 若该值大于128，则图像的细节纹理的shoot强度大于大边缘，大边缘的shoot强度分别等于over_shoot和under_shoot。 - 若该值小于128，则图像的细节纹理的shoot强度小于大边缘，大边缘的shoot强度分别等于over_shoot和under_shoot。 取值范围：[0, 255]。
detail_ctrl_thr[HI_ISP_AUTO_ISO_NUM]	图像的细节纹理区的shoot强度的控制阈值。 该值配合detail_ctrl使用，用于区分detail_ctrl所控制shoot的纹理区和边缘，也即纹理区和边缘的区分阈值。小于该值的区域为纹理区，该纹理区域的shoot会被detail_ctrl单独控制，而大于该阈值的边缘区域的shoot依然等于over_shoot和under_shoot。 取值范围：[0, 0xFF]，建议值0xA0。
edge_filt_str[HI_ISP_AUTO_ISO_NUM]	边缘滤波强度调试参数：实现图像锐化边缘的范围和边缘平滑强度的控制。 该值越大，判为边缘的区域越多、也越宽，edge_strength起作用的图像边缘就越多，而且，该值越大，沿着边缘方向的平滑滤波强度也越大，边缘就越平滑。反之，判为边缘的区域越少、也越窄，edge_strength起作用的图像区域越少，边缘平滑就越弱。 取值范围：[0, 63]，建议值53。
edge_filt_max_cap[HI_ISP_AUTO_ISO_NUM]	边缘滤波强度范围的调试参数：该值越大，边缘滤波的最大强度也最大，edge_filt_strength的可调试范围也越大；一般建议该值大小控制30以内。 取值范围：[0, 47]；建议值18。
r_gain[HI_ISP_AUTO_ISO_NUM]	深红色区域的锐化增益控制参数。该值越大，则深红色区域的锐化强度越大。 取值范围：[0, 31]；建议值28。
g_gain[HI_ISP_AUTO_ISO_NUM]	绿色区域的锐化增益控制参数。该值越大，则绿色区域的锐化强度越大。 取值范围：[0, 255]；建议值32。
b_gain[HI_ISP_AUTO_ISO_NUM]	深蓝色区域的锐化增益控制参数。该值越大，则深蓝色区域的锐化强度越大。 取值范围：[0, 31]；建议值28。

成员名称	描述
skin_gain[HI_ISP_AUTO_ISO_NUM]	肤色区域的锐化增益控制参数。该值越大，则肤色区域的锐化强度越大。 取值范围：[0, 31]；建议值23。
max_sharp_gain[HI_ISP_AUTO_ISO_NUM]	图像锐化的最大增益限制值。该值越大，图像的锐化幅度越大，反之，锐化幅度越小。适当的调小该参数，可以减少图像的过锐化，可以减少图像锐化后的黑白点。 取值范围：[0, 0x7FF]，建议值0xA0。

3.3.15.21 hi_isp_cmos_sharpen

说明

定义sharpen的CMOS参数。

定义

```
typedef struct {  
    hi_u8 skin_umin;  
    hi_u8 skin_vmin;  
    hi_u8 skin_umax;  
    hi_u8 skin_vmax;  
    hi_isp_cmos_sharpen_manual manual_attr;  
    hi_isp_cmos_sharpen_auto auto_attr;  
} hi_isp_cmos_sharpen;
```

成员

成员名称	描述
skin_umin	肤色区域范围矩形窗的左下角的最小坐标的U值，取值范围：[0, 255]。
skin_vmin	肤色区域范围矩形窗的左下角的最小坐标的V值，取值范围：[0, 255]。
skin_umax	肤色区域范围矩形窗的右上角的最大坐标的U值，取值范围：[0, 255]。
skin_vmax	肤色区域范围矩形窗的右上角的最大坐标的V值，取值范围：[0, 255]。
manual_attr	Sharpen手动参数。
auto_attr	Sharpen自动参数。

3.3.15.22 hi_isp_cmos_edgemark

说明

定义edgemark的CMOS参数。

定义

```
typedef struct {  
    hi_bool enable;  
    hi_u8 threshold;  
    hi_u32 color;  
} hi_isp_cmos_edgemark;
```

成员

成员名称	描述
enable	EdgeMark边缘标记功能的使能。 - HI_FALSE：关闭； - HI_TRUE：使能。
threshold	边缘标记的阈值。计算得到的图像的局部统计量大于该阈值，则认为是强边缘或者强的纹理特征，将大于该阈值的强边缘和强的纹理特征标记上色。该值越小，则被标记的边缘和强的纹理特征越多。 取值范围：[0, 255]。
color	边缘标记的颜色设置值。该值设置的是RGB三分量值的大小。该值的0-7bit的低8位是表示B分量的大小，8-15bit的中间8位是G分量的大小，16-23bit的高8位是R分量的大小，24-31bit的最高8位保留不用。通过设置color可以为边缘标记的边缘标记上色选择合适的颜色，一般所选的颜色最好和图像的背景颜色形成明显的反差对比。 取值范围：[0, 0xFFFFFFFF]。默认值0xFF0000。

3.3.15.23 hi_isp_cmos_hlc

说明

定义hlc的CMOS参数。

定义

```
typedef struct {  
    hi_bool enable;  
    hi_u8 luma_thr;  
    hi_u8 luma_target;  
} hi_isp_cmos_hlc;
```


成员

成员名称	描述
enable	HLC高光遮蔽功能的使能。 - HI_FALSE：关闭； - HI_TRUE：使能。
luma_thr	高光遮蔽的阈值。亮度值大于该值的区域将会被拉灰成目标值为luma_target的低亮度值。该值一般都设置的比较大，否则，图像的很多区域都会被拉灰。 取值范围：[0, 255]。
luma_target	高光遮蔽拉灰的目标值。亮度值大于luma_thr的区域将被拉灰成目标值为luma_target的低亮度值，也即该区域的亮度将会被明显拉低拉暗。该值一般都设置的比较小，以减少高亮光源对人眼的刺激。 取值范围：[0, 255]。

3.3.15.24 hi_isp_cmos_ldci

说明

定义ldci的CMOS参数。

定义

```
typedef struct {  
    hi_bool enable;  
    hi_u8 gauss_lpf_sigma;  
    hi_u8 he_pos_wgt[HI_ISP_AUTO_ISO_NUM];  
    hi_u8 he_pos_sigma[HI_ISP_AUTO_ISO_NUM];  
    hi_u8 he_pos_mean[HI_ISP_AUTO_ISO_NUM];  
    hi_u8 he_neg_wgt[HI_ISP_AUTO_ISO_NUM];  
    hi_u8 he_neg_sigma[HI_ISP_AUTO_ISO_NUM];  
    hi_u8 he_neg_mean[HI_ISP_AUTO_ISO_NUM];  
    hi_u16 blc_ctrl[HI_ISP_AUTO_ISO_NUM];  
} hi_isp_cmos_ldci;
```

成员

成员名称	描述
enable	LDCI功能使能开关。 - HI_FALSE：关闭； - HI_TRUE：使能。
gauss_lpf_sigma	LPF滤波强度，其值越小说明LDCI作用越局域。 取值范围：[0x1, 0xFF]。

成员名称	描述
he_pos_wgt	直方图均衡结果与原图融合权重曲线，可根据原图亮度控制提亮程度；曲线呈高斯分布，横坐标为亮度。 #define HI_ISP_AUTO_ISO_NUM 16
he_pos_sigma	局域直方图均衡结果与原图融合权重曲线，根据亮度控制提亮程度，曲线呈GAUSS分布，该参数为GAUSS分布均方差，取值范围：[0x1, 0xFF]。
he_pos_mean	局域直方图均衡结果与原图融合权重曲线，根据亮度控制提亮程度，曲线呈GAUSS分布，该参数为GAUSS分布数学期望，取值范围：[0x0, 0xFF]。
he_neg_wgt	直方图均衡结果与原图融合权重曲线，可根据原图亮度控制减暗程度；曲线呈高斯分布，横坐标为亮度。
he_neg_sigma	局域直方图均衡结果与原图融合权重曲线，根据亮度控制减暗程度，曲线呈GAUSS分布，该参数为GAUSS分布均方差，取值范围：[0x1, 0xFF]。
he_neg_mean	局域直方图均衡结果与原图融合权重曲线，根据亮度控制减暗程度，曲线呈GAUSS分布，该参数为GAUSS分布数学期望，取值范围：[0x0, 0xFF]。
bhc_ctrl	手动模式下暗区增益控制强度，用于抑制暗区噪声，取值范围：[0x0, 0x1FF]。

3.3.15.25 hi_isp_cmos_dpc

说明

定义dpc的CMOS参数。

定义

```
typedef struct {  
    hi_u16 strength[HI_ISP_AUTO_ISO_NUM];  
    hi_u16 blend_ratio[HI_ISP_AUTO_ISO_NUM];  
} hi_isp_cmos_dpc;
```

成员

成员名称	描述
strength	DPC动态坏点处理强度，取值越大，DPC处理强度越强，可以校正的动态坏点越多。取值范围：[0x0, 0xFF]。 #define HI_ISP_AUTO_ISO_NUM 16
blend_ratio	坏点校正过程所需的融合比率。 取值范围：[0x0, 0x80]。

3.3.15.26 hi_isp_cmos_ge

说明

定义ge的CMOS参数。

定义

```
typedef struct {  
    hi_bool enable;  
    hi_u8 slope;  
    hi_u8 sensi_slope;  
    hi_u16 sensi_thr;  
    hi_u16 threshold[HI_ISP_AUTO_ISO_NUM];  
    hi_u16 strength[HI_ISP_AUTO_ISO_NUM];  
    hi_u16 np_offset[HI_ISP_AUTO_ISO_NUM];  
} hi_isp_cmos_ge;
```

成员

成员名称	描述
enable	使能Crosstalk功能。 - HI_FALSE：关闭； - HI_TRUE：使能。
slope	设置Crosstalk门限之上的处理强度。值越大，表示在门限值threshold之上整体处理强度衰减得越慢。 取值范围：[0, 14]。
sensi_slope	设置Crosstalk敏感度值。默认值 9。值越大，表示在门限值sensi_threshold之上，边沿上的处理强度衰减得越慢。 取值范围：[0, 14]。
sensi_thr	设置Crosstalk敏感度门限值。值越大，表示在更强的边沿上也会有绿平衡处理。 取值范围：[0, 0x3FFF]。
strength	设置Crosstalk强度值。值越大，处理强度越大。取值范围：[0, 256]。该数组的 16 个值分别对应的sensor在不同增益情况下不同的设置值。 #define HI_ISP_AUTO_ISO_NUM 16
threshold	设置Crosstalk门限值。值越大，表示整体处理的强度越大。该数组的 16 个值分别对应的sensor在不同增益情况下不同的设置值。 取值范围：[0, 0x3FFF]。
np_offset	设置Noise Profile值。值越大，表示抗噪能力越大。该数组的 16 个值分别对应的sensor在不同增益情况下不同的设置值。 取值范围：[0x200, 0x3FFF]。

3.3.15.27 hi_isp_cmos_afc

说明

定义afc的CMOS参数。

定义

```
typedef struct {  
    hi_bool enable;  
    hi_u8 threshold[HI_ISP_AUTO_ISO_NUM];  
    hi_u8 strength[HI_ISP_AUTO_ISO_NUM];  
} hi_isp_cmos_afc;
```

成员

成员名称	描述
enable	ANTIFALSECOLOR模块使能。 - HI_FALSE：关闭； - HI_TRUE：使能。
threshold	去伪彩阈值，值越大，伪彩去除的区域越多。 取值范围：[0, 0x20]。 #define HI_ISP_AUTO_ISO_NUM 16
strength	去伪彩强度，值越大，去伪彩强度越大，伪彩现象消除越好，但可能导致正常颜色的改变。 取值范围：[0, 0x1F]。

3.3.15.28 hi_isp_cmos_bayernr

说明

定义bayernr的CMOS参数。

定义

```
typedef struct {  
    hi_bool enable;  
    hi_bool bnr_mono_sensor_en;  
    hi_bool nr_lsc_enable;  
    hi_u8 bnr_lsc_max_gain;  
    hi_u16 bnr_lsc_cmp_strength;  
  
    hi_u8 lut_fine_str[HI_ISP_AUTO_ISO_NUM];  
    hi_u8 chroma_strength[HI_ISP_BAYER_CHN_NUM][HI_ISP_AUTO_ISO_NUM];  
    hi_u8 wdr_frame_strength[HI_ISP_WDR_MAX_FRAME_NUM];  
    hi_u8 fusion_frame_strength[HI_ISP_WDR_MAX_FRAME_NUM];  
    hi_u16 coarse_strength[HI_ISP_BAYER_CHN_NUM][HI_ISP_AUTO_ISO_NUM];  
    hi_u16 lut_coring_wgt[HI_ISP_AUTO_ISO_NUM];  
    hi_u16 lut_coring_ratio[HI_ISP_BAYERNR_LUT_LENGTH];  
} hi_isp_cmos_bayernr;
```

成员

成员名称	描述
enable	NR模块使能。 - HI_FALSE：关闭； - HI_TRUE：使能。
bnr_mono_sensor_en	黑白sensor模式使能。 - HI_FALSE：关闭； - HI_TRUE：使能。
nr_lsc_enable	NR参考LSC增益功能使能。 - HI_FALSE：关闭； - HI_TRUE：使能。
bnr_lsc_max_gain	参考LSC增益最大倍数。 取值范围[0x0, 0xBF]。
bnr_lsc_cmp_strength	参考LSC增益强度。 取值范围[0x0,0x100]。
lut_fine_str	亮噪去除强度，值越大，亮噪去除强度越大。取值范围：[0x0,0x80]。 <code>#define HI_ISP_AUTO_ISO_NUM 16</code>
chroma_strength	色噪去除强度选择，值越大，去色噪强度越大。四通道分别可调。 取值范围：[0x0, 0x3]。 <code>#define HI_ISP_AUTO_ISO_NUM 16</code> <code>#define HI_ISP_BAYER_CHN_NUM 4</code>
wdr_frame_strength	WDR模式下每一合成帧分别对应的全局亮度去噪强度。值越大，去噪强度越大，值越小，去噪强度越弱。 取值范围：[0x0, 0x50]。 <code>#define HI_ISP_WDR_MAX_FRAME_NUM 4</code>
fusion_frame_strength	Fusion模式下每一融合帧分别对应的全局亮度去噪强度。 取值范围：[0x0, 0x50]。
coarse_strength	控制亮度噪声整体去噪强度，四通道强度独立可调，值越大，亮噪去噪强度越大。 取值范围：[0x0, 0x360]。
lut_coring_wgt	随机噪声保留强度，值越大，保留的随机噪声越多。 取值范围：[0x0,0xC80]。

成员名称	描述
lut_coring_ratio	噪声保留强度，根据亮度在coring_ratio数组里查找对应的噪声保留强度，值越大，亮度噪声保留越多，值越小，亮度噪声保留越少。 取值范围[0x0, 0x3FF]。 #define HI_ISP_BAYERNR_LUT_LENGTH 33

3.3.15.29 hi_isp_cmos_ca

说明

定义ca的CMOS参数。

定义

```
typedef struct {  
    hi_bool enable;  
    hi_u16 y_ratio_lut[HI_ISP_CA_YRATIO_LUT_LENGTH];  
    hi_s16 iso_ratio[HI_ISP_AUTO_ISO_NUM];  
} hi_isp_cmos_ca;
```

成员

成员名称	描述
enable	色彩调整模块的使能。 - HI_FALSE：关闭； - HI_TRUE：使能。
y_ratio_lut	CA模式，根据亮度Y查找的调整色度UV的增益。意思是将亮度划分成256等分，对应的设定256个调整系数A，用于调整UV的值。建议调整的时候Y值越小（也就是比较暗的地方）对应的增益A越小，这样可以有效的抑制暗区的色噪。亮度的增益设置大一些，亮区的颜色会鲜艳一些。 取值范围：[0,2047] #define HI_ISP_CA_YRATIO_LUT_LENGTH 256
iso_ratio[HI_ISP_AUTO_ISO_NUM]	CA模式，根据ISO值查找的调整色度UV的增益B。这是一个全局的增益，也就是ISO固定，图像所有像素点的UV的调整增益都是B。建议在低ISO的时候对应的增益B可以设置大一些，高ISO（低照度）对应的增益B可以设置小一些，可以降低低照度下的暗处色噪的目的。 取值范围：[0,2047] #define HI_ISP_AUTO_ISO_NUM 16

3.3.15.30 hi_isp_cmos_wdr

说明

定义wdr的CMOS参数。

定义

```
typedef struct {  
    hi_bool fusion_mode;  
    hi_bool motion_comp;  
  
    hi_u16 short_thr;  
    hi_u16 long_thr;  
  
    hi_bool force_long;  
    hi_u16 force_long_low_thr;  
    hi_u16 force_long_hig_thr;  
  
    hi_bool short_expo_chk;  
    hi_u16 short_check_thd;  
    hi_bool md_ref_flicker;  
  
    hi_u8 md_thr_low_gain[HI_ISP_AUTO_ISO_NUM];  
    hi_u8 md_thr_hig_gain[HI_ISP_AUTO_ISO_NUM];  
  
    hi_isp_bnr_mode bnr_mode;  
    hi_u16 fusion_thr[HI_ISP_WDR_MAX_FRAME];  
    hi_u8 mdt_still_thd;  
    hi_u8 mdt_long_blend;  
} hi_isp_cmos_wdr;
```

成员

成员名称	描述
fusion_mode	Fusion合成开关。 - HI_FALSE：关闭； - HI_TRUE：使能。
motion_comp	WDR运动补偿使能。取值范围： - HI_FALSE：关闭； - HI_TRUE：使能。 在某些室内日光灯场景下，运动补偿会加重频闪带来的横条纹现象，此时可以关闭运动补偿。
short_thr	短曝光门限值，超过门限值的图像数据只选择短曝光数据。 取值范围：[0,0xFFFF]。
long_thr	长曝光门限值，低于门限值的图像数据只选择长曝光数据。 取值范围：[0, short_threshold]。

成员名称	描述
force_long	强制输出长帧使能。取值范围： - HI_FALSE：关闭； - HI_TRUE：使能。
force_long_low_thre shold	强制输出长帧的低阈值，亮度低于该阈值的运动区域强制输出长帧 取值范围：[0x0, 0xFF]。
force_long_hig_thre shold	强制输出长帧的高阈值，亮度大于该阈值的运动区域不强制
short_expo_chk	控制运动检测是否检查短帧数据大小，当短帧数据接近 0 时， WDR融合强制选长帧。 取值范围： - HI_FALSE：WDR融合时不检查短帧数据大小； - HI_TRUE：WDR融合时检查短帧数据大小。
short_check_threshold	短帧检查阈值。当短帧亮度小于该阈值时，WDR强制输出长帧。 取值范围：[0x,0xFF]
md_ref_flicker	控制运动判断是否参考Flicker检测结果。 取值范围： HI_FALSE：做运动判断时不参考Flicker检测结果； HI_TRUE：做运动判断时参考Flicker检测结果。
md_thr_low_gain	手动模式下判断是否运动的低阈值系数。值越大，运动程度越小。取值范围：[0, md_thr_hig_gain]。
md_thr_hig_gain	手动模式下判断是否运动的高阈值系数。值越大，运动程度越小。取值范围：[0, 0xFF]。
fusion_thr[HI_WDR_MAX_FRAME]	对应Fusion模式下超短帧，短帧，中帧，长帧的阈值，将输入数据限制在阈值以下。 取值范围：[0,0x3FFF]。 #define HI_WDR_MAX_FRAME 2
mdt_still_thd	运动检测中判断为静止的阈值，当运动程度小于该阈值时，判断该像素点为静止。 取值范围：[0,0xFE]。
mdt_long_blend	运动区域叠加长帧的权重，当该值为0xFE时，运动区域全部选择长帧。 取值范围：[0x0,0xFE]。

3.3.15.31 hi_isp_cmos_dehaze

说明

定义dehaze的CMOS参数。

定义

```
typedef struct {  
    hi_bool    enable;  
    hi_bool    user_lut_enable;  
    hi_op_mode op_type;  
    hi_u8      auto_strength;  
    hi_u8      manual_strength;  
} hi_isp_cmos_dehaze;
```

成员

成员名称	描述
enable	使能Dehaze功能。
user_lut_enable	用户自定义去雾曲线使能。
op_type	DEHAZE工作类型。默认值为HI_OP_MODE_AUTO。
manual_strength	手动模式下去雾强度，取值范围：[0,255]
auto_strength	自动模式下去雾强度，取值范围：[0,255]

3.3.15.32 hi_isp_cmos_lcac_manual

说明

定义手动模式下配置紫边校正强度参数。

定义

```
typedef struct {  
    hi_u8 de_purple_cr_str;  
    hi_u8 de_purple_cb_str;  
} hi_isp_cmos_lcac_manual;
```

成员

成员名称	描述
de_purple_cr_str	R通道的校正强度，取值范围：[0，8]。
de_purple_cb_str	B通道的校正强度，取值范围：[0，8]。

3.3.15.33 hi_isp_cmos_lcac_auto

说明

定义自动模式下配置紫边校正强度参数。

定义

```
typedef struct {  
    hi_u8 de_purple_cr_str[HI_ISP_EXP_RATIO_STRENGTH_NUM];  
    hi_u8 de_purple_cb_str[HI_ISP_EXP_RATIO_STRENGTH_NUM];  
} hi_isp_cmos_lcac_auto;
```

宏定义如下：

```
#define HI_ISP_EXP_RATIO_STRENGTH_NUM 16
```

成员

成员名称	描述
de_purple_cr_str	自动模式下，R通道的校正强度。该组的16个值分别对应16个不同的曝光比，一般情况下，曝光比越大，配置的去紫边的强度值越大，对应关系如表1所示。若连续两档的处理强度不相等，则通过线性插值的方法确定两档中间某个曝光比值对应的处理强度。 取值范围：[0, 8]。
de_purple_cb_str	自动模式下，B通道的校正强度。 取值范围：[0, 8]。

3.3.15.34 hi_isp_cmos_lcac

说明

定义lcac的CMOS参数。

定义

```
typedef struct {  
    hi_bool enable;  
    hi_op_mode op_mode;  
    hi_u16 var_thr;  
    hi_u16 r_luma_thd[HI_ISP_LCAC_DET_NUM];  
    hi_u16 g_luma_thd[HI_ISP_LCAC_DET_NUM];  
    hi_u16 b_luma_thd[HI_ISP_LCAC_DET_NUM];  
    hi_u16 y_luma_thd[HI_ISP_LCAC_DET_NUM];  
    hi_s16 cbcr_ratio[HI_ISP_LCAC_DET_NUM];  
    hi_u16 purple_det_range;  
    hi_isp_cmos_lcac_manual manual_attr;  
    hi_isp_cmos_lcac_auto auto_attr;  
} hi_isp_cmos_lcac;
```

成员

成员名称	描述
enable	enable紫边校正使能。 HI_FALSE：禁止； HI_TRUE：使能。
op_mode	紫边校正工作模式。
var_thr	var_threshold边缘检测阈值取值范围：[0, 4095]。
r_luma_thd	它分了3段，每一段的值表示高亮检测模块里面R分量阈值。实际中生效的值取决于purple_detect_range这个参数的大小。purple_detect_range越小，越倾向于r_detect_threshold第一段的值，purple_detect_range越大，越倾向于r_detect_threshold第3段的值。取值范围：[0, 4095]。 #define HI_ISP_LCAC_DET_NUM3
g_luma_thd	它分了3段，每一段的值表示高亮检测模块里面G分量阈值。实际中生效的值取决于purple_detect_range这个参数的大小。purple_detect_range越小，越倾向于g_detect_threshold第一段的值，purple_detect_range越大，越倾向于g_detect_threshold第3段的值。 取值范围：[0, 4095]。
b_luma_thd	它分了3段，每一段的值表示高亮检测模块里面B分量阈值。实际中生效的值取决于purple_detect_range这个参数的大小。purple_detect_range越小，越倾向于b_detect_threshold第一段的值，purple_detect_range越大，越倾向于b_detect_threshold第3段的值。 取值范围：[0, 4095]。
y_luma_thd	它分了3段，每一段的值表示高亮检测模块里面Luma分量阈值。实际中生效的值取决于purple_detect_range这个参数的大小。purple_detect_range越小，越倾向于luma_detect_threshold第一段的值，purple_detect_range越大，越倾向于luma_detect_threshold第3段的值。 取值范围：[0, 4095]

成员名称	描述
cocr_ratio	它分了3段，每一段的值表示紫色检测模块蓝色程度。实际中生效的值取决于purple_detect_range这个参数的大小。purple_detect_range越小，越倾向于cb_cr_ratio第一段的值，purple_detect_range越大，越倾向于cb_cr_ratio第3段的值。 取值范围：[-2048, 2047]。
purple_det_range	紫色检测的范围，该值控制的是r_detect_threshold, g_detect_threshold, b_detect_threshold, luma_detect_threshold, cb_cr_ratio这几个参数的生效情况。值越大，越多非高亮区域的紫色被界定为紫边区域。purple_detect_range可能会引入紫边去除不平滑的问题，可以检查r_detect_threshold, g_detect_threshold, b_detect_threshold, luma_detect_threshold, cb_cr_ratio这几个参数的配置。 取值范围：[0,410]。
manual_attr	手动模式下配置紫边校正强度参数。
auto_attr	自动模式下配置紫边校正强度参数。

3.3.15.35 hi_isp_bnr_lsc_calib_table

说明

定义Bayer NR LSC的标定LUT 。

定义

```
typedef struct {  
    hi_u16 r_gain[HI_ISP_RLSC_POINTS];  
    hi_u16 gr_gain[HI_ISP_RLSC_POINTS];  
    hi_u16 gb_gain[HI_ISP_RLSC_POINTS];  
    hi_u16 b_gain[HI_ISP_RLSC_POINTS];  
} hi_isp_bnr_lsc_calib_table;
```

成员

成员名称	描述
r_gain	R通道校正增益表。取值范围：[0, 1023] #define HI_ISP_RLSC_POINTS 129
gr_gain	Gr通道校正增益表。取值范围：[0, 1023]
gb_gain	Gb通道校正增益表。取值范围：[0, 1023]
b_gain	B通道校正增益表。取值范围：[0, 1023]

3.3.15.36 hi_isp_3a_alg_lib

说明

定义AE/AWB算法库结构体。

定义

```
typedef struct {  
    hi_s32 id;  
    hi_char lib_name[HI_ISP_ALG_LIB_NAME_SIZE_MAX];  
} hi_isp_3a_alg_lib;
```

成员

成员名称	描述
id	算法库实例的Id，用以支持运行同一个算法库的多个实例
lib_name	标识算法库名称的字符数组，用以区分不同的算法库。 HI_ISP_ALG_LIB_NAME_SIZE_MAX用于定义3A算法库名称的最大字符数，当前为20。

3.3.15.37 hi_isp_bind_attr

说明

定义ISP库与Sensor、3A库之间绑定关系的结构体。

定义

```
typedef struct {  
    hi_sensor_id sensor_id;  
    hi_isp_3a_alg_lib ae_lib;  
    hi_isp_3a_alg_lib af_lib;  
    hi_isp_3a_alg_lib awb_lib;  
} hi_isp_bind_attr;
```

成员

成员名称	描述
sensor_id	Sensor的id。
ae_lib	AE库结构体。
af_lib	AF库结构体。
awb_lib	AWB库结构体。

3.3.15.38 hi_isp_ctrl_proc_write

说明

定义ISP的PROC信息。

定义

```
typedef struct {  
    hi_char *proc_buff;  
    hi_u32 buff_len;  
    hi_u32 write_len;  
} hi_isp_ctrl_proc_write;
```

成员

成员名称	描述
proc_buff	ISP传给当前算法的Proc信息Buffer指针。
buff_len	ISP传给当前算法的Proc信息Buffer当前剩余字节数。Buffer总大小为8K字节。
write_len	当前算法传给ISP的Proc信息字节数。

注意事项

用户使用自己的3A算法，并需要支持3A算法的proc信息功能时，才需要关注此接口。

3.3.15.39 hi_isp_ctrl_cmd

说明

定义ISP对3A的控制命令。

定义

```
typedef enum {  
    HI_ISP_WDR_MODE_SET = 8000,  
    HI_ISP_PROC_WRITE,  
    HI_ISP_AE_FPS_BASE_SET,  
    HI_ISP_AE_BLC_SET,  
    HI_ISP_AE_RC_SET,  
    HI_ISP_AE_BAYER_FORMAT_SET,  
    HI_ISP_AE_INIT_INFO_GET,  
    HI_ISP_AWB_ISO_SET,  
    HI_ISP_CHANGE_IMAGE_MODE_SET,  
    HI_ISP_UPDATE_INFO_GET,  
    HI_ISP_FRAMEINFO_GET,  
    HI_ISP_ATTACHINFO_GET,  
    HI_ISP_COLORGAMUTINFO_GET,  
    HI_ISP_AWB_INTTIME_SET,  
    HI_ISP_BAS_MODE_SET,  
    HI_ISP_PROTRIGGER_SET,  
    HI_ISP_AWB_PIRIS_SET,  
    HI_ISP_AWB_SNAP_MODE_SET,  
    HI_ISP_AWB_ZONE_ROW_SET,  
    HI_ISP_AWB_ZONE_COL_SET,  
}
```

```
HI_ISP_AWB_ZONE_BIN_SET,  
HI_ISP_AWB_ERR_GET,  
HI_ISP_CTRL_CMD_BUTT,  
} hi_isp_ctrl_cmd;
```

成员

成员名称	描述
HI_ISP_WDR_MODE_SET	设置WDR模式，将ISP控制单元的WDR模式配置到算法模块，此命令对应的参数数据类型是hi_wdr_mode。
HI_ISP_PROC_WRITE	设置写PROC信息，将算法模块的PROC信息配置到ISP控制单元，此命令对应的参数数据类型是hi_isp_ctrl_proc_write。
HI_ISP_AE_FPS_BASE_SET	设置帧率，将ISP控制单元的帧率信息配置到AE算法模块，此命令对应的参数与hi_isp_pub_attr里面的frame_rate一样。
HI_ISP_AE_BLC_SET	设置黑电平，将黑电平信息配置到AE算法模块。
HI_ISP_AE_RC_SET	设置RC模块使能，将Radial Crop模块是否使能信息传入AE算法模块。
HI_ISP_AE_BAYER_FORMAT_SET	设置Bayer图像数据格式，将Bayer图像数据格式信息传入AE算法模块。
HI_ISP_AWB_ISO_SET	设置ISO值，将AE当前的ISO值配置到AWB模块，用于自动调整饱和度，此命令对应的参数与hi_isp_ae_result里面的iso一样。
HI_ISP_CHANGE_IMAGE_MODE_SET	设置图像分辨率切换标识，将ISP控制单元的图像分辨率标识配置到算法模块，此命令对应的参数数据类型为hi_u8，参数值为0表示图像分辨率未切换，其他值表示图像分辨率已切换。
HI_ISP_UPDATE_INFO_GET	AE，AWB状态信息更新，将算法获取运行过程中的状态信息，包括 AE，AWB的状态信息。
HI_ISP_FRAMEINFO_GET	获取ISP帧信息，包括ISO、去噪强度，与编码模块配合使用。
HI_ISP_ATTACHINFO_GET	获取ISP帧额外信息，包括ISO、各个模块的算法参数等。
HI_ISP_COLORGAMUTINFO_GET	获取通道色域属性。
HI_ISP_AWB_INTTIME_SET	设置曝光量值，将AE当前的曝光量值配置到AWB模块，用于室内外检测，此命令对应的参数与hi_isp_ae_result里面的int_time一样。
HI_ISP_BAS_MODE_SET	设置BAS模式。

成员名称	描述
HI_ISP_PROTRIGGER_SET	专业拍照触发信号设置，当用户触发一次专业拍照时，ISP将此信号设置给AE，AE启动专业曝光控制。当前不支持。
HI_ISP_AWB_PIRIS_SET	设置piris运行的增益信息，用于获取当前piris实际的状态信息。
HI_ISP_AWB_SNAP_MODE_SET	设置是否为抓拍模式，将ISP当前是否为抓拍模式的信息配置到AWB 模块。
HI_ISP_AWB_ZONE_ROW_SET	设置AWB分块统计信息的行数。拼接、Crop等处理导致AWB统计信息的行数不固定，需要将行数配置到AWB模块。
HI_ISP_AWB_ZONE_COL_SET	设置AWB分块统计信息的列数。拼接、Crop等处理导致AWB统计信息的列数不固定，需要将列数配置到AWB模块。
HI_ISP_AWB_ZONE_BIN_SET	设置AWB分块统计信息的亮度分组数。
HI_ISP_AWB_ERR_GET	获取AWB的报错信息。
HI_ISP_CTRL_CMD_BUTT	预留值。

3.3.15.40 hi_isp_stitch_attr

说明

定义ISP拼接结构体。

定义

```
typedef struct {  
    hi_bool stitch_enable;  
    hi_bool main_pipe;  
    hi_u8  stitch_pipe_num;  
    hi_s8  stitch_bind_id[HI_VI_MAX_PIPE_NUM];  
} hi_isp_stitch_attr;
```

成员

成员名称	描述
stitch_enable	拼接使能。
main_pipe	是否为主Pipe。
stitch_pipe_num	拼接总路数。
stitch_bind_id	拼接绑定的pipe的号。 HI_VI_MAX_PIPE_NUM定义最大的PIPE数量，当前为4。

3.3.15.41 hi_isp_ae_register

说明

定义AE注册结构体。

定义

```
typedef struct {  
    hi_isp_ae_exp_func ae_exp_func;  
} hi_isp_ae_register;
```

成员

成员名称	描述
ae_exp_func	AE注册的回调函数结构体。

3.3.15.42 hi_isp_ae_exp_func

说明

定义AE回调函数结构体。

定义

```
typedef struct {  
    hi_s32 (*pfn_ae_init)(hi_s32 handle, const hi_isp_ae_param *ae_param);  
    hi_s32 (*pfn_ae_run)(hi_s32 handle,  
        const hi_isp_ae_info *ae_info,  
        hi_isp_ae_result *ae_result,  
        hi_s32 reserved);  
    hi_s32 (*pfn_ae_ctrl)(hi_s32 handle, hi_u32 cmd, hi_void *value);  
    hi_s32 (*pfn_ae_exit)(hi_s32 handle);  
} hi_isp_ae_exp_func;
```

成员

成员名称	描述
pfn_ae_init	初始化AE的回调函数指针。
pfn_ae_run	运行AE的回调函数指针。
pfn_ae_ctrl	控制AE内部状态的回调函数指针。 cmd值需参见 hi_isp_ctrl_cmd 。
pfn_ae_exit	销毁AE的回调函数指针。

注意事项

- 调用hi_mpi_isp_init时将调用pfn_ae_init回调函数，以初始化AE算法库。
- 调用hi_mpi_isp_run时将调用pfn_ae_run回调函数，以运行AE算法库，计算得到sensor的曝光时间和增益、ISP的数字增益。
- 设计思路中，算法库实现ctrl接口用以改变内部运行状态，ctrl接口提供一个参数传输命令，提供一个VOID类型的指针传输数据。ctrl接口一方面以回调函数指针的形式注册给ISP库，ISP控制单元隐式调用一些命令控制算法库内部运行状态，另一方面，以算法库的用户接口的形式，从而用户可以改变算法库内部运行状态。示例：

```
hi_s32 ae_ctrl_cmd(hi_s32 handle, hi_u32 cmd, hi_void *value)
{
    ae_check_pointer_return(value);
    switch (cmd) {
        case HI_ISP_WDR_MODE_SET:
            .....
            break;
            .....
    }
    return HI_SUCCESS;
}
```

运行时ISP控制单元会隐式调用pfn_ae_ctrl回调函数，通知AE算法库切换WDR和线性模式、设置FPS、通知配置sensor。

当前Firmware定义的ctrl命令详参hi_isp_ctrl_cmd。

- 调用hi_mpi_isp_exit时将调用pfn_ae_exit回调函数，以销毁AE算法库。
- 一个算法库支持初始化和运行多个实例，参数handle以区分不同的算法库实例。如果需要支持多个实例，可以用不同的alg_lib. id多次注册算法库。例如：

```
hi_isp_3a_alg_lib ae_lib;
ae_lib.id = 0;
hi_vi_pipe vi_pipe = 0;
strncpy(ae_lib.lib_name, HI_AE_LIB_NAME, sizeof(HI_AE_LIB_NAME));
hi_mpi_ae_register(vi_pipe,&ae_lib);
ae_lib.id = 1;
hi_mpi_ae_register(vi_pipe,&ae_lib);
```

3.3.15.43 hi_isp_ae_param

说明

定义ISP提供给AE的初始化参数结构体。

定义

```
typedef struct {
    hi_sensor_id sensor_id;
    hi_u8 wdr_mode;
    hi_u8 hdr_mode;
    hi_u16 black_level;
    hi_float fps;
    hi_isp_bayer_format bayer;
    hi_isp_stitch_attr stitch_attr;
    hi_s32 reserved;
} hi_isp_ae_param;
```

成员

成员名称	描述
sensor_id	向ISP注册的sensor的id，用以检查向ISP注册的sensor和向AE注册的sensor是否一致。
wdr_mode	宽动态模式，ISP向AE提供宽动态模式信息。
hdr_mode	HDR模式，ISP向AE提供HDR模式信息。不支持。
black_level	黑电平值，12bit精度，ISP向AE提供黑电平信息。
fps	帧率，ISP向AE提供帧率信息。
bayer	Sensor Bayer Pattern，包括RGGB、GRBG、GBRG、BGGR四种格式。
stitch_attr	拼接模式，ISP向AE提供拼接模式信息。

3.3.15.44 hi_isp_fe_ae_stat_1

说明

定义AE统计信息结构体。

定义

```
typedef struct {  
    hi_u32 pixel_count[HI_ISP_CHN_MAX_NUM];  
    hi_u32 pixel_weight[HI_ISP_CHN_MAX_NUM];  
    hi_u32 histogram_mem_array[HI_ISP_CHN_MAX_NUM][HI_ISP_HIST_NUM];  
} hi_isp_fe_ae_stat_1;
```

宏定义如下：

```
#define HI_ISP_HIST_NUM            1024  
#define HI_ISP_CHN_MAX_NUM        4
```

成员

成员名称	描述
pixel_count	统计的像素点总个数。
pixel_weight	带权重统计的像素点总个数。
histogram_mem_array	1024段直方图的统计信息数组，取值范围：[0, 0xFFFFFFFF]。

3.3.15.45 hi_isp_fe_ae_stat_2

说明

定义AE统计信息结构体。

定义

```
typedef struct {  
    hi_u16 global_avg_r[HI_ISP_CHN_MAX_NUM];  
    hi_u16 global_avg_gr[HI_ISP_CHN_MAX_NUM];  
    hi_u16 global_avg_gb[HI_ISP_CHN_MAX_NUM];  
    hi_u16 global_avg_b[HI_ISP_CHN_MAX_NUM];  
} hi_isp_fe_ae_stat_2;
```

宏定义如下：

```
#define HI_ISP_CHN_MAX_NUM 4
```

成员

成员名称	描述
global_avg_r	全局R分量平均值，取值范围：[0, 0xFFFF]。
global_avg_gr	全局Gr分量平均值，取值范围：[0, 0xFFFF]。
global_avg_gb	全局Gb分量平均值，取值范围：[0, 0xFFFF]。
global_avg_b	全局B分量平均值，取值范围：[0, 0xFFFF]。

3.3.15.46 hi_isp_fe_ae_stat_3

说明

定义AE统计信息结构体。

定义

```
typedef struct {  
    hi_u16 zone_avg[HI_ISP_CHN_MAX_NUM][HI_ISP_AE_ZONE_ROW][HI_ISP_AE_ZONE_COLUMN]  
[HI_ISP_BAYER_PATTERN_NUM];  
} hi_isp_fe_ae_stat_3;
```

宏定义如下：

```
#define HI_ISP_CHN_MAX_NUM 4  
#define HI_ISP_AE_ZONE_ROW 15  
#define HI_ISP_AE_ZONE_COLUMN 17  
#define HI_ISP_BAYER_PATTERN_NUM 4
```

成员

成员名称	描述
zone_avg	分区间R、Gr、Gb、B分量平均值，取值范围：[0, 0xFFFF]。

3.3.15.47 hi_isp_fe_ae_stitch_stat_3

说明

定义AE统计信息结构体。

定义

```
typedef struct {
    hi_u16 zone_avg[HI_VI_MAX_PIPE_NUM][HI_ISP_CHN_MAX_NUM][HI_ISP_AE_ZONE_ROW]
    [HI_ISP_AE_ZONE_COLUMN][HI_ISP_BAYER_PATTERN_NUM];
} hi_isp_fe_ae_stitch_stat_3;
```

宏定义如下：

```
#define HI_VI_MAX_PHY_PIPE_NUM      4
#define HI_VI_MAX_VIR_PIPE_NUM      8
#define HI_VI_MAX_PIPE_NUM          (HI_VI_MAX_PHY_PIPE_NUM + HI_VI_MAX_VIR_PIPE_NUM)
#define HI_ISP_CHN_MAX_NUM          4
#define HI_ISP_AE_ZONE_ROW          15
#define HI_ISP_AE_ZONE_COLUMN       17
```

成员

成员名称	描述
zone_avg	拼接模式下拼接后分区间R、Gr、Gb、B分量平均值，取值范围：[0, 0xFFFF]。 只有参与拼接的Pipe分区间均值有效，其余Pipe的均值无效。

3.3.15.48 hi_isp_ae_info

说明

定义ISP提供给AE的统计信息结构体。

定义

```
typedef struct {
    hi_u32 frame_cnt; /* the counting of frame */
    hi_isp_fe_ae_stat_1 *fe_ae_stat1;
    hi_isp_fe_ae_stat_2 *fe_ae_stat2;
    hi_isp_fe_ae_stat_3 *fe_ae_stat3;
    hi_isp_fe_ae_stitch_stat_3 *fe_ae_sti_stat;
    hi_isp_fe_ae_stat_1 *be_ae_stat1;
    hi_isp_fe_ae_stat_2 *be_ae_stat2;
```

```
hi_isp_fe_ae_stat_3 *be_ae_stat3;  
hi_isp_fe_ae_stitch_stat_3 *be_ae_sti_stat;  
} hi_isp_ae_info;
```

成员

成员名称	描述
frame_cnt	帧的累加计数，取值范围：[0, 0xFFFFFFFF]。
fe_ae_stat1	统计的像素点总个数。
	带权重统计的像素点总个数。
	1024段直方图的统计信息数组，取值范围：[0, 0xFFFFFFFF]。
fe_ae_stat2	全局R分量平均值，取值范围：[0, 0xFFFF]。
	全局Gr分量平均值，取值范围：[0, 0xFFFF]。
	全局Gb分量平均值，取值范围：[0, 0xFFFF]。
	全局B分量平均值，取值范围：[0, 0xFFFF]。
fe_ae_stat3	分区间R、Gr、Gb、B分量平均值，取值范围：[0, 0xFFFF]。
fe_ae_sti_stat	拼接模式下拼接后分区间R、Gr、Gb、B分量平均值，取值范围：[0, 0xFFFF]。 只有参与拼接的Pipe分区间均值有效，其余Pipe的均值无效。
be_ae_stat1	统计的像素点总个数。
	带权重统计的像素点总个数。
	1024段直方图的统计信息数组，取值范围：[0, 0xFFFFFFFF]。
be_ae_stat2	全局R分量平均值，取值范围：[0, 0xFFFF]。
	全局Gr分量平均值，取值范围：[0, 0xFFFF]。
	全局Gb分量平均值，取值范围：[0, 0xFFFF]。
	全局B分量平均值，取值范围：[0, 0xFFFF]。
be_ae_stat3	分区间R、Gr、Gb、B分量平均值，取值范围：[0, 0xFFFF]。
be_ae_sti_stat	拼接模式下拼接后分区间R、Gr、Gb、B分量平均值，取值范围：[0, 0xFFFF]。 只有进行拼接对应Pipe的分区间均值有效，其余Pipe的均值无效。

注意事项

- AE库可以根据frame_cnt控制运算频率，例如两帧运算一次。

- fe_ae_stat1、be_ae_stat1分别表示位于FE及BE的全局1024段直方图统计信息。该统计信息是取输入数据流中的高10bit数据统计得到的，每个bin中数据表示该灰度值对应的像素个数。全局1024段直方图会受到分区间权重的影响，1024个bin的数据之和即为带权重参与统计的像素点个数。目前，AE算法默认只用了Gr通道的统计信息，在大面积红色时，会采用R和Gb通道的统计信息，在大面积蓝色时，会采用B和Gr通道的统计信息。
- 当MIPI、VI DEV的data_rate配置为DATA_RATE_X2时，fe_ae_stat1中的pixel_count、pixel_weight数目会减半。
- fe_ae_stat2、be_ae_stat2分别表示位于FE及BE的全局R/Gr/Gb/B 4分量取高16bit统计得到的均值，取值范围：[0, 0xFFFF]。全局4分量平均值会受分区间权重影响。
- fe_ae_stat3、be_ae_stat3分别表示位于FE及BE的15*17区间每个区间R/Gr/Gb/B 4分量取高16bit统计得到的均值，取值范围：[0, 0xFFFF]。
- AE统计模块可将输入数据开方后再进行统计。所谓数据开方，指的是对输入数据归一化至1后做开方处理。以1024段直方图统计为例，若输入数据为12bit，某个像素值为2048，若数据开方关闭，直接取高10bit数据进行统计，为灰度值512对应的bin像素个数加1；若数据开方使能，像素值2048归一化至1后为0.5，0.5开方为0.707，用10bit表示0.707为724，此时在直方图上表现为灰度值724对应的bin像素个数加1。由此可见，像素值较小的数据开方处理后像素值明显变大，相当于统计信息通过压缩亮区的统计精度来提升暗区的统计精度。建议在WDR模式下数据开方使能，线性模式下数据开方关闭。在开方模式下，统计精度为11bit，低5bit值为0，因此16bit最大值为0xFFE0。此外，AE统计模块在ISP pipeline的位置可以变化，具体可参考[3.33 统计信息](#)章节部分相关描述。

表 3-2 统计信息 hi_isp_ae_info 的成员变量默认配置说明

成员名称	主要统计信息	默认位置	黑电平	权重表影响
fe_ae_stat1	WDR合成前1024段直方图	FE-WB之后	未减	是
fe_ae_stat2	WDR合成前全局均值	FE-WB之后	未减	是
fe_ae_stat3	WDR合成前分块均值	FE-WB之后	未减	否
be_ae_stat1	WDR合成后1024段直方图	BE-WB之后	已减	是
be_ae_stat2	WDR合成后全局均值	BE-WB之后	已减	是
be_ae_stat3	WDR合成后分块均值	BE-WB之后	已减	否

 说明

- 表中说明仅在ISP默认配置下生效，实际会受到黑电平配置及AE统计信息位置的影响。
- WDR合成前（FE）统计信息为固定在WB模块之后，不可配置。使用FE统计信息时需要减去黑电平，直方图应减去10bit黑电平，均值应减去16bit黑电平。WDR合成前(FE)统计信息受到位于FE AE之前处理模块（在Atlas 200/500 A2推理产品中，FE AE之前的处理模块有DG/WB）增益影响，这些模块的FE增益值由算法内部保证与BE保持一致，无需单独配置。
- 线性模式推荐使用WDR合成前(FE)1024段直方图。WDR模式推荐使用WDR合成前(FE)1024段直方图，或者使用WDR合成后(BE)开方模式的1024段直方图。业务压力大时，WDR合成前(FE)实时性更好，因此推荐使用FE统计信息。SDK提供的AE算法在如果采用的是BE统计信息，默认在线性模式下使用（BE）不开方模式的统计信息，在WDR模式下使用（BE）开方模式的统计信息。如果线性模式下使用（BE）开方模式的统计信息，或者WDR模式下使用（BE）不开方模式的统计信息，会导致SDK提供AE算法的出现异常。

3.3.15.49 hi_isp_ae_stat_attr

说明

定义AE库返回给ISP的配置寄存器结构体。

定义

```
typedef struct {  
    hi_bool change;  
    hi_bool hist_adjust;  
    hi_u8 ae_be_sel;  
    hi_u8 four_plane_mode;  
    hi_u8 hist_offset_x;  
    hi_u8 hist_offset_y;  
    hi_u8 hist_skip_x;  
    hi_u8 hist_skip_y;  
    hi_bool mode_update;  
    hi_u8 hist_mode;  
    hi_u8 aver_mode;  
    hi_u8 max_gain_mode;  
    hi_bool wight_table_update;  
    hi_u8 weight_table[HI_VI_MAX_PIPE_NUM][HI_ISP_AE_ZONE_ROW][HI_ISP_AE_ZONE_COLUMN];  
} hi_isp_ae_stat_attr;
```

成员

子成员名称	描述
change	该结构体中的值是否需要配置寄存器。 • HI_FALSE：否； • HI_TRUE：是。

子成员名称	描述
hist_adjust	是否使能AE算法直方图调整。 • HI_FALSE: 否; • HI_TRUE: 是。 影响ae_be_sel/four_plane_mode/hist_offset_x/ hist_offset_y/ hist_skip_x/hist_skip_y六个参数的配置。 hist_adjust使能时, 以上六个参数以hi_isp_ae_result中的值为准配置芯片寄存器; hist_adjust不使能时, 以上六个参数以外部寄存器的值(MPI配置)为准配置芯片寄存器。
ae_be_sel	BE中AE统计模块在ISP pipeline的位置, 默认值为1。FE中AE统计模块固定在WB之后, 位置不可移动。 0: After ISP digital gain; 1: After static WB; 2: After DRC。
four_plane_mode	是否使能FourPlaneMode, 默认值为0, 使能时1024段直方图为BGGR四通道256段直方图。 • 0: 否; • 1: 是。
hist_skip_x	直方图统计时水平方向采样点设置, 0=every pixel; 1=every 2nd pixel; 2=every 3rd pixel; 3=every 4th pixel; 4=every 5th pixel; 5=every 8th pixel; 6+=every 9th pixel。 即取值为0时表示每1个像素采样一个进行统计; 取值为1时表示每2个像素采样一个进行统计, 以此类推。 0仅支持FourPlaneMode使能时配置。
hist_skip_y	直方图统计时垂直方向采样点设置, 0=every pixel; 1=every 2nd pixel; 2=every 3rd pixel; 3=every 4th pixel; 4=every 5th pixel; 5=every 8th pixel; 6+=every 9th pixel。
hist_offset_x	直方图统计时水平方向起始点设置。 0表示从第一列开始统计; 1表示从第二列开始统计。
hist_offset_y	直方图统计时垂直方向起始点设置。 0表示从第一行开始统计; 1表示从第二行开始统计。
mode_update	AE算法开方模式配置使能, 影响hist_mode/ aver_mode/ max_gain_mode三个参数的配置。 mode_update不为0时, 以上三个参数以hi_isp_ae_result中的值为准配置逻辑寄存器; mode_update为0时, 以上三个参数以外部寄存器的值(MPI配置)为准配置逻辑寄存器。

子成员名称	描述
hist_mode	全局1024段直方图开方模式。 0为不开方； 1为开方。 仅影响BE直方图统计信息。
aver_mode	均值开方模式。 0为不开方； 1为开方。 仅影响BE均值统计信息。
max_gain_mode	MG模块开方模式。 0为不开方； 1为开方。 仅影响MG模块统计信息。为了与AE分块统计信息进行比较，建议与aver_mode配置为同一模式。
wight_table_update	AE算法权重表配置使能，影响weight_table的配置。 - wight_table_update不为0时，权重表以hi_isp_ae_result中的值为准配置芯片寄存器； - wight_table_update为0时，权重表以外部寄存器的值(MPI配置)为准配置芯片寄存器。
weight_table[HI_ISP_AE_ZONE_ROW][HI_ISP_AE_ZONE_COLUMN]	15x17个区间的AE权重表，取值范围：[0, 15]。 #define HI_ISP_AE_ZONE_ROW 15 #define HI_ISP_AE_ZONE_COLUMN 17

注意事项

- weight_table支持拼接模式下在主路配置多路权重，在主路对所有支路对应Pipe下标的权重表进行赋值即可。在非拼接模式下，只有对应Pipe下标的权重表生效。
- SDK提供的AE算法只支持1024段直方图，在使用AE算法情况下使能four_plane_mode会导致AE算法异常。

3.3.15.50 hi_isp_ae_result

说明

定义AE库返回给ISP的配置寄存器结构体。

定义

```
typedef struct {  
    hi_u32 int_time[4];  
    hi_u32 isp_dgain;
```

```
hi_u32 again;
hi_u32 dgain;
hi_u32 iso;
hi_u32 isp_dgain_sf;
hi_u32 again_sf;
hi_u32 dgain_sf;
hi_u32 iso_sf;
hi_u8 ae_run_interval;
hi_bool piris_valid;
hi_s32 piris_pos;
hi_u32 piris_gain;
hi_u32 sns_lhcg_exp_ratio;
hi_isp_fswdr_mode fswdr_mode;
hi_u32 wdr_gain[HI_ISP_WDR_MAX_FRAME_NUM];
hi_u32 hmax_times;
hi_u32 vmax;
hi_isp_ae_stat_attr stat_attr;
hi_isp_dcf_update_info update_info;
} hi_isp_ae_result;
```

成员

成员名称	描述
int_time	AE计算得出的曝光时间，以1/16 us为单位，将曝光时间由行数转换成us时需要考虑cmos.c中offset的影响。 线性模式和sensor built-in WDR模式下，只有int_time[0]有效，int_time[1:3]建议配置等于int_time[0]；N帧合成WDR模式下，int_time[0:(N-1)]有效，配置值由小到大，依次表示最短到最长的曝光时间，用于计算长短帧曝光比，int_time[(N-1):3]建议配置等于int_time[(N-1)]。int_time[0]还需传递至其他模块用于与曝光时间相关的联动控制，会影响SDK提供的AWB效果。使用AWB算法和多帧WDR模式时必须配置该结构体。
isp_dgain	ISP的数字增益，8bit精度。使用ISP数字增益时必须配置；不使用时配置为0x100。
again	Sensor的模拟增益，10bit精度。使用多帧WDR算法时必须配置；不使用时配置为0x400。
dgain	Sensor的数字增益，10bit精度。使用多帧WDR算法时必须配置；不使用时配置为0x400。
iso	AE计算得出的总增益值，ISO表示系统增益，以常数100乘以倍数为单位,例如系统中sensor的增益为2倍，ISP的增益为1倍，那么整个系统的ISO值计算方式为：2*1*100=200，即系统ISO为200，本文中涉及到的ISO都是采用这种计算方法。此变量影响ISP的去噪，sharpen等自适应效果，必须配置。在2合1WDR模式下，ISO计算包含WDRGain。
isp_dgain_sf	短帧的ISP的数字增益，8bit精度。使用WDR算法时必须配置；不使用时配置为0x400。
again_sf	短帧的Sensor的模拟增益，10bit精度。使用WDR算法时必须配置；不使用时配置为0x400。
dgain_sf	短帧的Sensor的数字增益，10bit精度。使用WDR算法时必须配置；不使用时配置为0x400。

成员名称	描述
iso_sf	AE计算得出的短帧的总增益值。在2合1WDR模式下，ISO计算包含WDRGain。
ae_run_interval	AE算法运行的间隔，取值范围：[1, 255]。 取值为1时表示每帧都运行AE算法； 取值为2时表示每2帧运行1次AE算法，依此类推。 建议该值设置不要大于2，否则AE调节速度会受到影响。WDR模式时，该值建议设置为1，这样AE收敛会更加平滑。此变量决定AE计算结果配置到sensor和ISP寄存器的间隔帧数，必须配置。
piris_valid	Piris是否有效的标志。 - 取值为HI_TRUE时在内核态回调Piris驱动配置步进电机的位置。 - 取值为HI_FALSE时不回调。 - 使用AE算法和Piris驱动对接Piris镜头时，该值须置为HI_TRUE，对接非Piris镜头时该值须置为HI_FALSE。
piris_pos	Piris步进电机的位置，取值范围与具体Piris镜头相关。使用Piris驱动对接Piris镜头时，该值必须配置。
piris_gain	Piris光圈等效增益，取值范围与具体Piris镜头相关。可用于计算Piris工作时的等效曝光量，供其他模块参考。取值范围：[0, 1024]。对接非Piris镜头时建议将该值配置为512。
sns_lhcg_exp_ratio	LCG+HCG模式的基础曝光比，仅Sensor支持LCG+HCG模式时生效，不使用时需将该值配置为64。
fswdr_mode	WDR合成模式。 0：表示普通多帧合成WDR模式； 1：表示长帧模式； 2：表示自动长帧模式。
wdr_gain[HI_ISP_WDR_MAX_FRAME_NUM]	WDR合成前多路的数字增益，8bit精度。使用WDR合成前多路数字ISP数字增益时必须配置；不使用时配置为0x100。 <code>#define HI_ISP_WDR_MAX_FRAME_NUM 4</code>
hmax_times	Sensor对应读出一行的时间，单位：ns。
vmax	Sensor每帧实际生效的总行数，单位：行。
stat_attr	AE库返回给ISP的配置寄存器结构体
update_info	用于传递AE相关DCF信息，仅曝光相关参数生效。

注意事项

- ISP基本算法模块将会根据AE计算得出的总增益值调节配置参数，例如锐化、去噪等。
- weight_table支持拼接模式下在主路配置多路权重，在主路对所有支路对应Pipe下标的权重表进行赋值即可。在非拼接模式下，只有对应Pipe下标的权重表生效。

- 将曝光时间int_time由行转换成us时，可以通过cmos.c中的lines_per500ms进行，转换关系如下：
$$\text{int\_time}[0] = (((\text{hi\_u64})\text{int\_time\_rst}[0] * 1024 - \text{offset}) * 500000 / \text{ae\_sns\_dft->lines\_per500ms}) >> 10$$

上式中int_time_rst[0]是以行数为单位的曝光时间，offset=offset* 1024，offset即为曝光时间的偏移量，详见hi_isp_ae_accuracy的描述。
- 为了保证计算曝光比的精度，int_time中曝光时间精度为1/16us，计算曝光时间时需要保证在us的基础上左移四位，否则曝光比计算可能会有误差，其他模块获取的曝光时间也会偏小。
- 客户在使用非昇腾AE算法时，参数int_time、isp_dgain、again、dgain、iso、hmax_times、vmax必须配置，否则会影响其他模块的联动控制，其余参数则可根据需要选择性配置。

3.3.15.51 hi_isp_awb_register

说明

定义AWB注册结构体。

定义

```
typedef struct {  
    hi_isp_awb_exp_func awb_exp_func;  
} hi_isp_awb_register;
```

成员

成员名称	描述
awb_exp_func	AWB 注册的回调函数结构体。

3.3.15.52 hi_isp_awb_exp_func

说明

定义AWB回调函数结构体。

定义

```
typedef struct {  
    hi_s32 (*pfn_awb_init)(hi_s32 handle, const hi_isp_awb_param *awb_param, hi_isp_awb_result *awb_result);  
    hi_s32 (*pfn_awb_run)(hi_s32 handle, const hi_isp_awb_info *awb_info, hi_isp_awb_result *awb_result, hi_s32 reserved);  
    hi_s32 (*pfn_awb_ctrl)(hi_s32 handle, hi_u32 cmd, hi_void *value);  
    hi_s32 (*pfn_awb_exit)(hi_s32 handle);  
} hi_isp_awb_exp_func;
```

成员

成员名称	描述
pfn_awb_init	初始化AWB的回调函数指针。
pfn_awb_run	运行AWB的回调函数指针。
pfn_awb_ctrl	控制AWB内部状态的回调函数指针。 cmd值需参见 hi_isp_ctrl_cmd 。
pfn_awb_exit	销毁AWB的回调函数指针。

注意事项

- 调用hi_mpi_isp_init时将调用pfn_awb_init回调函数，以初始化AWB算法库。
- pfn_awb_init回调函数的hi_isp_awb_result参数返回ISP启动时的初始AWB增益和初始色彩校正矩阵。
- 调用hi_mpi_isp_run时将调用pfn_awb_run回调函数，以运行AWB算法库，计算得到白平衡增益、色彩校正矩阵。
- 运行时ISP控制单元会隐式调用pfn_awb_ctrl回调函数，通知AWB算法库切换WDR和线性模式、设置ISO和曝光时间（曝光时间单位是 us）。设置ISO的目的是为了实现ISO与饱和度的联动，增益大时色度噪声也会比较大，所以需要调节饱和度。设置曝光时间是为了辅助室内外判断。
当前Firmware定义的ctrl命令详细描述参见hi_isp_ctrl_cmd。
- 调用hi_mpi_isp_exit时将调用pfn_awb_exit回调函数，以销毁AWB算法库。

3.3.15.53 hi_isp_awb_param

说明

定义ISP提供给AWB的初始化参数结构体。

定义

```
typedef struct {  
    hi_sensor_id sensor_id;  
    hi_u8 wdr_mode;  
    hi_u8 awb_zone_row;  
    hi_u8 awb_zone_col;  
    hi_u8 awb_zone_bin;  
    hi_isp_stitch_attr stitch_attr;  
    hi_u16 awb_width;  
    hi_u16 awb_height;  
    hi_u32 init_iso;  
    hi_s8 reserved;  
} hi_isp_awb_param;
```

成员

成员名称	描述
sensor_id	向ISP注册的sensor的id，用以检查向ISP注册的sensor和向AWB注册的sensor是否一致。
wdr_mode	宽动态模式，ISP向AWB提供宽动态模式信息。
awb_zone_row	AWB统计结果行数。
awb_zone_col	AWB统计结果列数。
awb_zone_bin	AWB统计结果亮度分组个数。
stitch_attr	拼接信息结构体
awb_width	Firmware传递给AWB算法库的图像宽度。
awb_height	Firmware传递给AWB算法库的图像高度。
init_iso	Firmware传递给AWB算法库的AE 曝光ISO值。
reserved	保留参数。

3.3.15.54 hi_isp_awb_stat_result

说明

定义AWB统计信息结构体。

定义

```
typedef struct {  
    hi_u16 *zone_avg_r;  
    hi_u16 *zone_avg_g;  
    hi_u16 *zone_avg_b;  
    hi_u16 *zone_count;  
} hi_isp_awb_stat_result;
```

成员

成员名称	描述
zone_avg_r	Bayer域分区间统计信息中白点的 R 分量平均值数组起始地址。 R 分量均值取值范围为[0, 0xFFFF]。
zone_avg_g	Bayer域分区间统计信息中白点的 G 分量平均值数组起始地址。 G 分量均值取值范围为[0, 0xFFFF]。
zone_avg_b	Bayer域分区间统计信息中白点的 B 分量平均值数组起始地址。 B 分量均值取值范围为[0, 0xFFFF]。

成员名称	描述
zone_count	Bayer域分区间统计信息中白点的个数数组起始地址。已做归一化。 白点个数取值范围为[0, 0xFFFF]。

3.3.15.55 hi_isp_awb_stat_1

说明

定义AWB统计信息结构体。

定义

```
typedef struct {  
    hi_u16 metering_awb_avg_r;  
    hi_u16 metering_awb_avg_g;  
    hi_u16 metering_awb_avg_b;  
    hi_u16 metering_awb_count_all;  
} hi_isp_awb_stat_1;
```

成员

成员名称	描述
metering_awb_avg_r	Bayer域全局统计信息中白点的 R 分量平均值。 R 分量均值取值范围为[0, 0xFFFF]。
metering_awb_avg_g	Bayer域全局统计信息中白点的 G 分量平均值。 G 分量均值取值范围为[0, 0xFFFF]。
metering_awb_avg_b	Bayer域全局统计信息中白点的 B 分量平均值。 B 分量均值取值范围为[0, 0xFFFF]。
metering_awb_count_all	Bayer域全局统计信息中白点的个数。已做归一化。白点个数取值范围为[0, 0xFFFF]。

3.3.15.56 hi_isp_awb_info

说明

定义ISP提供给AWB的统计信息结构体。

定义

```
typedef struct {  
    hi_u32 frame_cnt;  
    hi_isp_awb_stat_1 *awb_stat1;  
    hi_isp_awb_stat_result awb_stat2;  
    hi_u8 awb_gain_switch;  
    hi_u8 awb_stat_switch;  
    hi_bool wb_gain_in_sensor;
```



```
hi_u32 wdr_wb_gain[HI_ISP_BAYER_CHN_NUM];  
} hi_isp_awb_info;
```

成员

成员名称	描述
frame_cnt	帧的累加计数。 取值范围为[0, 0xFFFFFFFF]。
awb_stat1	Awb统计信息 1
awb_stat2	Awb统计信息 2
awb_gain_switch	白平衡增益在ISP的位置。取值范围为[0,1]。 0：白平衡增益配置在WDR合成前的DG1处。 1：白平衡增益配置在WB处。
awb_stat_switch	白平衡统计模块在ISP的位置。取值范围为[0, 1, 2]。 0：白平衡统计模块在DG后。 1：白平衡统计模块在EXPANDER 后。 2：白平衡统计模块在DRC后。
wb_gain_in_sensor	白平衡增益是否在sensor配置。取值范围为[0, 1]。 0：白平衡增益在ISP配置。 1：白平衡增益在sensor配置。
wdr_wb_gain	在WDR合成前的DG1处配置的白平衡增益值。 #define HI_ISP_BAYER_CHN_NUM 4

注意事项

- AWB库可以根据frame_cnt控制运算频率，例如两帧运算一次。
- hi_isp_awb_info提供了全局统计信息和分块统计信息，不同模式下水平、垂直分块个数不固定，可通过hi_isp_awb_param的awb_zone_row、awb_zone_col两个参数获取。AWB分块个数 = awb_zone_row* awb_zone_col。
- 切换白平衡统计模块位置，会导致 2 帧统计信息错误，建议在启动时配置合理值，避免切换。如果必须进行位置切换，建议切换后，AWB算法冻结 2 帧以上，待统计信息正确后，再重新计算。
- 通过hi_mpi_isp_set_stats_cfg接口配置白平衡统计模块位置在DRC后，通过hi_mpi_isp_set_wb_attr接口关闭亮度对白平衡权重的影响，可以优化 WDR模式下暗区偏红问题。

3.3.15.57 hi_isp_awb_raw_stat_attr

说明

定义AWB库返回给ISP的配置寄存器结构体。

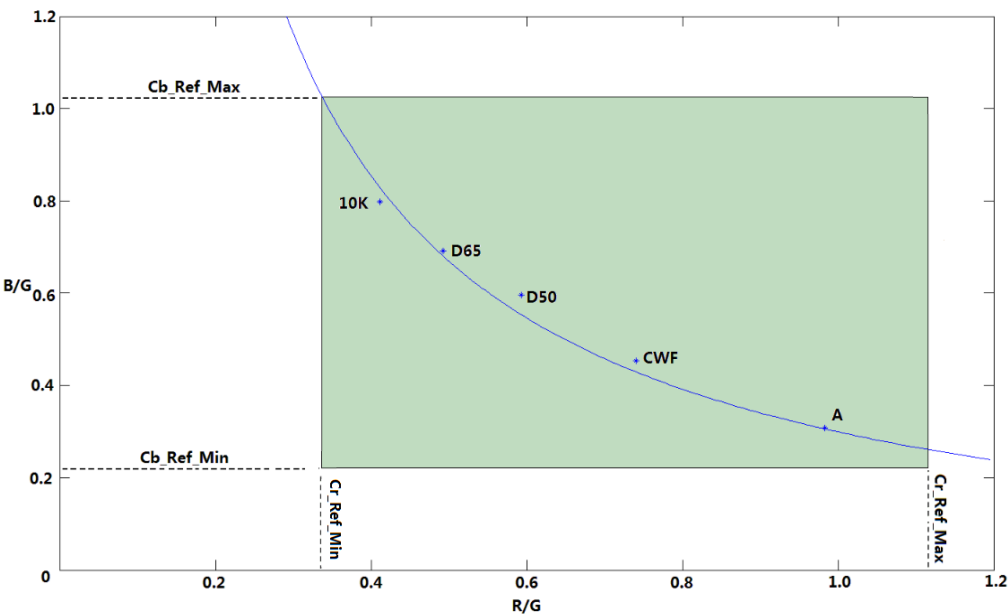
定义

```
typedef struct {  
    hi_bool stat_cfg_update;  
    hi_u16  metering_white_level_awb;  
    hi_u16  metering_black_level_awb;  
    hi_u16  metering_cr_ref_max_awb;  
    hi_u16  metering_cb_ref_max_awb;  
    hi_u16  metering_cr_ref_min_awb;  
    hi_u16  metering_cb_ref_min_awb;  
} hi_isp_awb_raw_stat_attr;
```

成员

成员名称	描述
stat_cfg_update	该结构体中的值是否需要配置寄存器
metering_white_level_awb	Bayer域统计白点信息时，找白点的亮度上限。取值范围：[0x0, 0xFFFF]，默认值 0xFFFF。
metering_black_level_awb	Bayer域统计白点信息时，找白点的亮度下限。取值范围：[0x0, 0xFFFF]，默认值 0x0。
metering_cr_ref_max_awb	Bayer域统计白点信息时，色差 R/G 的最大值，8bit 精度，默认值 512。
metering_cb_ref_max_awb	Bayer域统计白点信息时，色差 B/G 的最大值，8bit 精度，默认值 512。
metering_cr_ref_min_awb	Bayer域统计白点信息时，色差 R/G 的最小值，8bit 精度，默认值 128。
metering_cb_ref_min_awb	Bayer域统计白点信息时，色差 B/G 的最小值，8bit 精度，默认值 128。

图 3-4 白色区域选择相关参数



注意事项

- hi_isp_awb_raw_stat_attr结构体中的信息决定什么样的像素点被认为是白点，从而参与统计。用户开发新的AWB算法时可以使用默认值，也可以自定义配置，stat_cfg_update标识表明运行时当前帧是否需要配置stat_attr结构体中的值到寄存器。
- AWB统计模块在ISP Pipeline位置支持可配置，可配置为Digital Gain后，Expander后、或DRC后。默认在Digital Gain后。
- 只支持Bayer域统计信息。

3.3.15.58 hi_isp_awb_result

说明

定义AWB库返回给ISP的配置寄存器结构体。

定义

```
typedef struct {
    hi_u32 white_balance_gain[HI_ISP_BAYER_CHN_NUM];
    hi_u16 color_matrix[HI_ISP_CCM_MATRIX_SIZE];
    hi_u32 color_temp;
    hi_u8 saturation;
    hi_isp_awb_raw_stat_attr raw_stat_attr;
} hi_isp_awb_result;
```

成员

成员名称	描述
white_balance_gain	白平衡算法得出的 R、Gr、Gb、B 颜色通道的增益，16bit 精度表示。 #define HI_ISP_BAYER_CHN_NUM 4
color_matrix	色彩还原矩阵，8bit 精度表示。 #define HI_ISP_CCM_MATRIX_NUM 7
color_temp	AWB当前的色温。
saturation	当前的饱和度。
raw_stat_attr	定义AWB库返回给ISP的配置寄存器结构体。

注意事项

- AWB算法首先将会计算出R、Gr、Gb、B颜色通道的增益，以校正出白色。
- 16bit精度表明最后16位为小数。
- WDR模式，AWB统计结果和增益不需要做特别处理，和Linear模式一致。
- AWB算法其次会计算一个3x3的颜色校正矩阵，以还原出真实的色彩，8bit精度表明最后8位为小数。

3.3.15.59 hi_isp_dcf_update_info

说明

定义DCF信息中ISP实时更新配置参数。

定义

```
typedef struct {  
    hi_u32 iso_speed_ratings;  
    hi_u32 exposure_time;  
    hi_u32 exposure_bias_value;  
    hi_u8 exposure_program;  
    hi_u32 f_number;  
    hi_u32 max_aperture_value;  
    hi_u8 exposure_mode;  
    hi_u8 white_balance;  
} hi_isp_dcf_update_info;
```

成员

成员名称	描述
iso_speed_ratings	感光度。只读。
exposure_time	曝光时间 (快门速度的倒数)，单位是秒。 高16位为分子，低16位为分母。 只读。
exposure_bias_value	照片拍摄时的曝光补偿.，单位是APEX(EV)。高16 位为分子，低16位为分母。 只读。
exposure_program	拍照时相机使用的曝光程序。 1：手动曝光； 2：正常程序曝光； 3：光圈优先曝光； 4：快门优先曝光； 5：创意程序(慢速程序)； 6：动作程序(高速程序)； 7：肖像模式； 8：风景模式。只读。
f_number	光圈系数。 高16位为分子，低16位为分母。只读。
max_aperture_value	镜头的最大光圈值。 高16位为分子，低16位为分母。只读。

成员名称	描述
exposure_mode	曝光模式。 0：自动曝光； 1：手动曝光； 2：自动包围曝光。只读。
white_balance	白平衡。 0：自动白平衡； 1：手动白平衡。只读。

3.3.15.60 hi_isp_ctrl_param

说明

定义ISP控制参数结构体。

定义

```
typedef struct {  
    hi_u8    be_buf_num;  
    hi_u32   proc_param;  
    hi_u32   stat_interval;  
    hi_u32   update_pos;  
    hi_u32   interrupt_time_out;  
    hi_u32   pwm_num;  
    hi_u32   port_interrupt_delay;  
    hi_bool  ldci_tprflt_en;  
} hi_isp_ctrl_param;
```

成员

成员名称	描述
be_buf_num	ISP BE config buffer的数目。取值范围：[2, 9]，默认值为4
proc_param	ISP的PROC信息更新频率。默认值30; proc打印生效时最小值为1，无上限。proc_param为n表示每隔n帧更新一次ISP的PROC信息。配置0时表示不启用proc信息打印。
stat_interval	ISP统计信息更新频率。注意：在高帧率业务时，如120fps及其以上帧率，可通过参数stat_interval设置其ISP统计信息更新频率，从而可降低ISP的CPU占用率，同时降低性能消耗。 取值范围：(0,0xffffffff]

成员名称	描述
update_pos	默认值为0。根据hi_isp_sns_regs_info结构体中interrupt_pos变量的配置值。 0: sensor寄存器在帧起始中断或帧结束中断配置; - 其他值: sensor寄存器在帧结束中断配置。 取值范围: [0,1]
interrupt_time_out	表示中断超时的时间 (ms)。
pwm_num	表示PWM号, 配套dc-iris光圈控制场景使用。
port_interrupt_delay	表示Port中断延时时间, 默认值为0。 解决部分sensor (如: imx323) Half WDR模式下, 在前面几行配置sensor寄存器会出现闪烁, 需要延时。 port_init_delay是按照VI的工作时钟频率计算, 以时钟为单位。如: VI时钟是500MHz。延时 1ms, port_init_delay的计算如下: $\text{port\_init\_delay} (1\text{ms}) = 500\text{M}/1000\text{ms} = 500000$ 注意: port_init_delay仅在Half WDR模式下生效, 因为其他模式配置sensor不用Port中断。
ldci_tprflt_en	表示LDCI是否使能时域滤波, 默认值为0。

注意事项

- proc_param的默认值为30, 即每30帧更新一次ISP Proc信息, 如果要关闭ISP Proc信息, 则在hi_mpi_isp_mem_init之前通过接口hi_mpi_isp_set_ctrl_param设置ISP控制参数proc_param为 0, 不分配存储ISP Proc信息的内存, 而且后续不能将proc_param设置为非0值。
- 通过接口hi_mpi_isp_set_ctrl_param第一次设置proc_param为非0时, 必需在hi_mpi_isp_mem_init接口调用之前, 因为需要分配Proc信息存储的内存, 后续设置只能在非0值之间动态切换。
- ISP Proc信息频繁更新会消耗CPU资源。推荐设为30帧更新一次, 或仅Debug时开启。
- 只能在hi_mpi_isp_mem_init之前通过接口hi_mpi_isp_set_ctrl_param设置update_pos、pwm_num、port_interrupt_delay、ldci_tprflt_en、be_buf_num, 在hi_mpi_isp_mem_init之后不可再更改上述5个参数的配置值; 而hi_mpi_isp_get_ctrl_param接口调用顺序没有限制。
- 可通过接口hi_mpi_isp_set_ctrl_param动态更改proc_param、stat_interval、interrupt_time_out的配置值。
- 不支持加载ko时设置ISP控制参数。
- 离线模式下, VI缓存RAW数据的队列深度为8, 若业务量较大时, 采用默认的be_buf_num, 有可能会不丢帧但是ISP中有“get FreeBeBuf is fail”错误打印的问题, 此时可以增大be_buf_num的配置值, 缓解此问题; 另外对于业务量不繁重并且对于内存占用要求较高的应用场景, 可以适当的减小be_buf_num的配置值。

3.3.15.61 hi_isp_quick_start_param

说明

设定不带光敏AE快速启动参数结构体。

定义

```
typedef struct {  
    hi_bool quick_start_enable;  
    hi_u8 black_frame_num;  
    hi_bool ir_mode_en;  
    hi_u32 init_exposure_ir;  
    hi_u32 iso_thr_ir;  
    hi_u16 ir_cut_delay_time;  
} hi_isp_quick_start_param;
```

成员

成员名称	描述
quick_start_enable	设置是否打开AE启动快速收敛模式。该值为HI_TRUE时， 打开AE开机快速收敛模式， AE启动时会以最快速度进行收敛（对于大部分场景可在 10 帧内收敛完成）。此模式可以在没有光敏的时候达到启动AE快速收敛的需求。
black_frame_num	Sensor初始输出的坏帧数。此参数在quick_start_enable为HI_TRUE的情况下生效。
ir_mode_en	AE快速收敛模式下，支持红外模式开关。该值为HI_TRUE时， AE快速收敛支持红外模式。此参数在quick_start_enable为HI_TRUE的情况下生效。
init_exposure_ir	红外模式下的初始曝光量，等于曝光时间与曝光增益的乘积，其中曝光时间的单位为微秒(us)。此参数在quick_start_enable和ir_mode_en都为HI_TRUE的情况下生效。
iso_thr_ir	在AE快速收敛模式下，设定切换到红外模式（打开 IR CUT、IR LED）的ISO阈值，此参数在quick_start_enable为HI_TRUE的情况下生效。
ir_cut_delay_time	设置打开IR CUT物理上需要的时间。单位为ms。此参数在quick_start_enable为HI_TRUE的情况下生效。

注意事项

- black_frame_num填入sensor初始输出的坏帧数。因部分sensor在启动时，会输出若干坏帧。启动时没有坏帧的sensor则填入值为 0。
- 打开AE快速收敛模式状态下，如果外部硬件上没有IR CUT和IR LED，建议将ir_mode_en设定为HI_FALSE。此时AE启动快速收敛模式，不会进行相应的IR快速收敛模式。
- 在打开IR CUT过程中，对sensor的产生较大的亮度变化，此时AE算法内部会等待IR CUT切换完成。为了保证AE收敛的速度，加快快速收敛的速度，ir_cut_delay_time的值要设定合适，不能远大于实际IR CUT切换的时间。

3.3.15.62 hi_isp_awb_alg

说明

定义AWB算法类型。

定义

```
typedef enum {  
    HI_ISP_ALG_AWB_GW    = 0,  
    HI_ISP_ALG_AWB_SPEC = 1,  
    HI_ISP_ALG_BUTT  
} hi_isp_awb_alg;
```

成员

成员名称	描述
HI_ISP_ALG_AWB_GW	灰度世界AWB算法。
HI_ISP_ALG_AWB_SPEC	机器学习AWB算法。当前不支持。
HI_ISP_ALG_BUTT	预留值。

3.3.15.63 hi_isp_init_attr

说明

定义ISP第一次启动时AE/AWB的初始化参数结构体。

定义

```
typedef struct {  
    hi_bool is_ir_mode;  
    hi_u32 exp_time;  
    hi_u32 a_gain;  
    hi_u32 d_gain;  
    hi_u32 ispd_gain;  
    hi_u32 exposure;  
    hi_u32 init_iso;  
    hi_u32 lines_per500ms;  
    hi_u32 piris_fno;  
    hi_u16 wb_r_gain;  
    hi_u16 wb_g_gain;  
    hi_u16 wb_b_gain;  
    hi_u16 sample_r_gain;  
    hi_u16 sample_b_gain;  
    hi_u16 init_ccm[HI_ISP_CCM_MATRIX_SIZE];  
    hi_bool ae_route_ex_valid;  
    hi_bool quick_start_en;  
    hi_isp_ae_route ae_route;  
    hi_isp_ae_route_ex ae_route_ex;  
    hi_isp_ae_route ae_route_sf;  
    hi_isp_ae_route_ex ae_route_sf_ex;  
} hi_isp_init_attr;
```


成员

成员名称	描述
is_ir_mode	设置ISP启动状态是否红外模式。
exp_time	设置ISP第一次启动时AE初始曝光时间，单位为us，FSWDR模式下，表示当前最短帧(VS)曝光时间。不支持。
a_gain	设置ISP第一次启动时AE初始sensor模拟增益，精度为10bit，不支持。
d_gain	设置ISP第一次启动时AE初始sensor数字增益，精度为10bit，不支持。
ispd_gain	设置ISP第一次启动时AE初始ISP数字增益，精度为10bit，不支持。
exposure	设置ISP第一次启动时AE初始曝光量，等于曝光时间与曝光增益的乘积，其中曝光时间的单位为微秒(us)。
init_iso	设置ISP第一次启动时AE初始ISO值。
lines_per500ms	设置每500ms对应的曝光行数，用于计算AE初始曝光量。
piris_fno	P-Iris光圈F值对应的等效增益，不支持。
wb_r_gain	设置ISP第一次启动时AWB的 R 通道增益。
wb_g_gain	设置ISP第一次启动时AWB的 G 通道增益。
wb_b_gain	设置ISP第一次启动时AWB的 B 通道增益。
sample_r_gain	设置当前样机AWB在线标定的 G/R 的值。
sample_b_gain	设置当前样机AWB在线标定的 G/B 的值。
init_ccm[HI_ISP_CCM_MATRIX_SIZE]	设置ISP第一次启动时CCM值。 #define HI_ISP_CCM_MATRIX_NUM 7
ae_route_ex_valid	设置ISP第一次启动时AE扩展分配路线生效开关, 该值为HI_TRUE时使用扩展分配路线，否则使用普通分配路线。
quick_start_en	设置不带光敏快启相关参数。
ae_route	设置ISP第一次启动时AE的曝光分配路线。
ae_route_ex	设置ISP第一次启动时AE的扩展曝光分配路线。
ae_route_sf	设置ISP第一次启动时AE短帧的曝光分配路线，仅在WDR模式下使用。
ae_route_sf_ex	设置ISP第一次启动时AE短帧的扩展曝光分配路线，仅在WDR模式下使用。

注意事项

- 在ISP启动前，设置初始AWB增益和CCM系数，可改善前后帧颜色的一致性。
- 在ISP启动前，设置初始的AE曝光路线，可以实现帧率改变后，AE算法能够将初始设置的AE route保留并自动生效。

3.4 AE

3.4.1 AE 控制模块

3.4.1.1 hi_mpi_isp_set_exposure_attr

函数功能

设定AE曝光属性。

函数原型

```
hi_s32 hi_mpi_isp_set_exposure_attr(hi_vi_pipe vi_pipe, const  
hi_isp_exposure_attr *exp_attr)
```

约束说明

- AE曝光控制类型为自动时，曝光时间，曝光增益都由AE算法自动控制，可以通过配置自动曝光属性结构体auto_attr里面的参数得到不同的曝光效果。
- AE曝光控制类型为手动时，可以通过配置手动曝光属性结构体manual_attr控制使能类型（曝光时间使能、sensor模拟增益使能、sensor数字增益使能、ISP数字增益使能）及相应的曝光参数（曝光时间、sensor模拟增益、sensor数字增益、ISP数字增益）。
- AE曝光控制类型为自动时，配置手动曝光属性的参数无效。同理，AE曝光控制类型为手动时，配置自动曝光属性的参数无效。
- AE曝光控制类型为手动时，若曝光参数设置超出最大（小）值，将使用sensor支持的最大（小）值代替。
- 无论是自动曝光还是手动曝光，曝光时间的单位为微秒(us)，曝光增益的单位为10bit精度的倍数，即1024代表1倍，2048代表2倍等。
- WDR模式下，优先帧设置为长帧时，优先根据长帧的曝光路线进行曝光，其中2合1WDR模式下当增益分开配置时，短帧曝光路线根据长帧的曝光参数进行调整；优先帧设置为短帧时，优先根据短帧的曝光路线进行曝光，其中2合1WDR模式下当增益分开配置时，长帧曝光路线根据短帧的曝光参数进行调整。
- 2合1WDR模式下，设置增益分开配置，若sensor支持长短帧不同增益，可实现长短帧不同的sensor模拟增益、sensor数字增益、WDR增益，若sensor不支持长短帧不同增益，可实现长短帧不同的WDR增益。
- 使用灌图接口从BE进行回灌时，需要将AE曝光控制类型设置为手动。

参数说明

参数名	输入/输出	说明
vi_pipe	输入	VI PIPE号。 取值范围：[0, 12)。
exp_attr	输入	AE曝光属性结构体指针。

返回值说明

- 0：成功
- 非0：失败，参见[ISP返回码](#)。

参考样例

自动曝光属性设置：

```
hi_vi_pipe vi_pipe = 0;
hi_isp_exposure_attr exp_attr;

hi_mpi_isp_get_exposure_attr(vi_pipe, &exp_attr);

exp_attr.bypass = HI_FALSE;
exp_attr.prior_frame = HI_ISP_LONG_FRAME;
exp_attr.ae_gain_sep_cfg = HI_FALSE;
exp_attr.op_type = HI_OP_MODE_AUTO;
exp_attr.auto_attr.exp_time_range.max = 40000;
exp_attr.auto_attr.exp_time_range.min = 10;
hi_mpi_isp_set_exposure_attr(vi_pipe, &exp_attr);

exp_attr.auto_attr.speed = 0x80;
hi_mpi_isp_set_exposure_attr(vi_pipe, &exp_attr);

exp_attr.auto_attr.exp_attr = HI_ISP_AE_EXP_HIGHLIGHT_PRIOR;
exp_attr.auto_attr.hist_ratio_slope = 0x100;
exp_attr.auto_attr.max_hist_offset = 0x40;
hi_mpi_isp_set_exposure_attr(vi_pipe, &exp_attr);

exp_attr.auto_attr.antiflicker.enable = HI_TRUE;
exp_attr.auto_attr.antiflicker.frequency = 50;
exp_attr.auto_attr.antiflicker.mode = HI_ISP_ANTIFLICKER_NORMAL_MODE;
hi_mpi_isp_set_exposure_attr(vi_pipe, &exp_attr);

exp_attr.auto_attr.ae_delay_attr.black_delay_frame = 10;
exp_attr.auto_attr.ae_delay_attr.white_delay_frame = 0; hi_mpi_isp_set_exposure_attr(vi_pipe, &exp_attr);
```

手动曝光属性设置：

```
hi_vi_pipe vi_pipe = 0;
hi_isp_exposure_attr exp_attr;

hi_mpi_isp_get_exposure_attr(vi_pipe, &exp_attr);
exp_attr.bypass = HI_FALSE;
exp_attr.op_type = HI_OP_MODE_MANUAL;
exp_attr.manual_attr.a_gain_op_type = HI_OP_MODE_MANUAL;
exp_attr.manual_attr.d_gain_op_type = HI_OP_MODE_MANUAL;
exp_attr.manual_attr.ispd_gain_op_type = HI_OP_MODE_MANUAL;
exp_attr.manual_attr.exp_time_op_type = HI_OP_MODE_MANUAL;
exp_attr.manual_attr.a_gain = 0x400;
exp_attr.manual_attr.d_gain = 0x400;
exp_attr.manual_attr.ispd_gain = 0x400;
```

```
exp_attr.manual_attr.exp_time = 0x40000;  
hi_mpi_isp_get_exposure_attr(vi_pipe, &exp_attr);
```

3.4.1.2 hi_mpi_isp_get_exposure_attr

函数功能

获取AE曝光属性。

函数原型

hi_s32 hi_mpi_isp_get_exposure_attr(**hi_vi_pipe** vi_pipe, **hi_isp_exposure_attr** *exp_attr)

参数说明

参数名	输入/输出	说明
vi_pipe	输入	VI PIPE号。 取值范围：[0, 12)。
exp_attr	输出	AE曝光属性结构体指针。

返回值说明

- 0：成功
- 非0：失败，参见[ISP返回码](#)。

3.4.1.3 hi_mpi_isp_set_wdr_exposure_attr

函数功能

设置WDR模式下的AE。

函数原型

hi_s32 hi_mpi_isp_set_wdr_exposure_attr(**hi_vi_pipe** vi_pipe, const **hi_isp_wdr_exposure_attr** *wdr_exp_attr)

参数说明

参数名	输入/输出	说明
vi_pipe	输入	VI PIPE号。 取值范围：[0, 12)。
wdr_exp_attr	输入	WDR模式下的AE曝光属性结构体指针。

3.4.1.4 hi_mpi_isp_get_wdr_exposure_attr

函数功能

获取WDR模式下的AE。

函数原型

```
hi_s32 hi_mpi_isp_get_wdr_exposure_attr(hi_vi_pipe vi_pipe,
hi_isp_wdr_exposure_attr *wdr_exp_attr)
```

参数说明

参数名	输入/输出	说明
vi_pipe	输入	VI PIPE号。 取值范围：[0, 12)。
wdr_exp_attr	输出	WDR模式下的AE曝光属性结构体指针。

返回值说明

- 0：成功
- 非0：失败，参见ISP返回码。

3.4.1.5 hi_mpi_isp_set_ae_route_attr

函数功能

设置AE曝光分配策略属性。

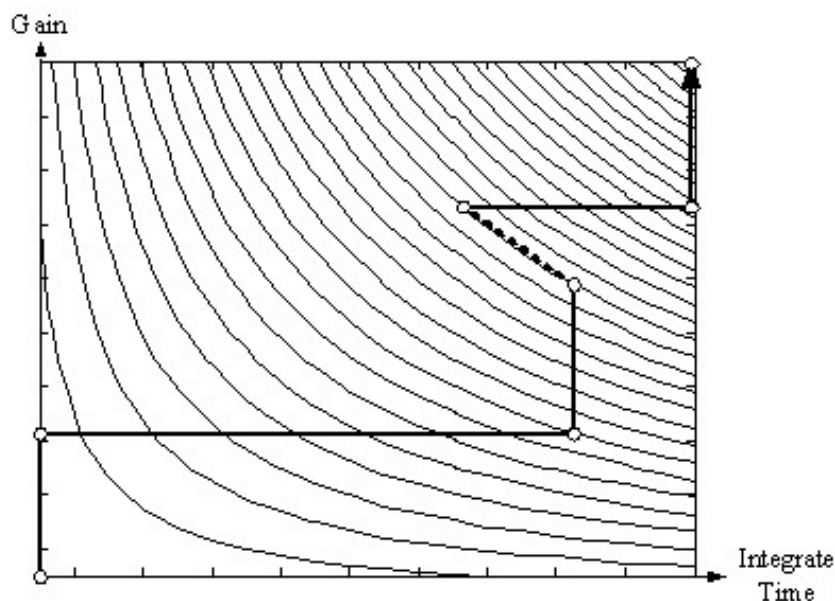
函数原型

```
hi_s32 hi_mpi_isp_set_ae_route_attr(hi_vi_pipe vi_pipe, const hi_isp_ae_route
*ae_route_attr)
```

约束说明

- 此接口用于设定AE曝光分配路线，AE计算得到的曝光量将按照设定的路线进行分配，用户可以根据自己的需求设定曝光优先、增益优先、光圈优先。
- AE分配路线示意图如下图所示。

图 3-5 AE 分配路线示意图



AE分配路线遵循以下限定：

- 最大支持 16 个节点，每个节点有曝光时间、增益、光圈三个分量，增益包含模拟增益、数字增益、ISP数字增益。
- 节点中曝光时间的单位为微秒(us)，不能设置为 0，也不能设置太小导致实际对应的曝光行数为 0，否则可能产生异常。
- 光圈分量仅支持P-Iris，不支持DC-Iris，因为DC-Iris无法精确控制，所以DCIris和手动光圈镜头光圈分量是无效的。即光圈类型为DC-Iris时，节点光圈分量不会对曝光量分配产生任何影响。
- 节点的曝光量是曝光时间、增益和光圈的乘积，节点曝光量单调递增，后一个节点的曝光量大于或等于前一个节点的曝光量，第一个节点的曝光量最小，最后一个节点的曝光量最大。
- 如果相邻节点的曝光量增加，那么应该有一个分量增加，其他分量固定，增加的分量决定该段路线的分配策略。例如增益分量增加，那么该段路线的分配策略是增益优先。
- 不支持设置等曝光量节点。
- 用户可以根据不同的场景设置不同的路线，分配路线支持动态切换。
- AE分配路线不能用于限制曝光参数的最大值和最小值。如果当前曝光量不在用户设定的路线范围当中，即曝光量超出了第一个节点或最后一个节点，曝光参数分配会超出节点限制，但不超过参数本身范围限制。
- 针对DC-Iris和手动光圈镜头，默认AE分配策略是首先分配曝光时间，其次分配增益。针对P-Iris镜头，默认AE分配策略是首先调节光圈，将光圈调至最大后调节曝光时间，最后再分配增益。如果当前曝光量不在用户设定的路线范围当中，按默认策略分配。
- 在线进行DC-Iris和P-Iris切换，AE route会重置为与光圈类型相匹配的默认分配策略，用户可以根据需要在切换光圈类型时自行设置AE route。
- 2 合 1WDR模式下，优先帧为短帧且增益分开配置不使能时，AE route不生效。

- 自动降帧时，若cmos.c里面设置了AE route，则在切换后会采用cmos.c中的AE route，否则采用已生效的AE route。同时，最大曝光时间的改变会更新到AE route中。
- 线性模式与WDR模式切换时，若cmos.c里面设置了AE route，则在切换后会采用cmos.c中的route，否则采用默认的AE route。
- 帧率或分辨率切换时，若用户设置的最大曝光目标时间大于切换后 1 帧所允许的最大曝光时间，那么分配路线的最大曝光时间会更新为切换后 1 帧所允许的最大曝光时间。
- 发生自动降帧、线性与WDR模式切换、帧率或分辨率切换、优先帧切换、增益分开配置切换、限制曝光时间或增益的最大最小值等情况时，实际生效的AE route可能与MPI设置的不一致，此时可以通过hi_mpi_isp_query_exposure_info获取实际生效的AE route。

参数说明

参数名	输入/输出	说明
vi_pipe	输入	VI PIPE号。 取值范围：[0, 12)。
ae_route_attr	输入	AE曝光分配策略结构体指针。

返回值说明

- 0：成功
- 非0：失败，参见ISP返回码。

3.4.1.6 hi_mpi_isp_get_ae_route_attr

函数功能

获取AE曝光分配策略属性。

函数原型

```
hi_s32 hi_mpi_isp_get_ae_route_attr(hi_vi_pipe vi_pipe, hi_isp_ae_route *ae_route_attr)
```

参数说明

参数名	输入/输出	说明
vi_pipe	输入	VI PIPE号。 取值范围：[0, 12)。
ae_route_attr	输出	AE曝光分配策略结构体指针。

返回值说明

- 0：成功
- 非0：失败，参见[ISP返回码](#)。

3.4.1.7 hi_mpi_isp_set_ae_route_attr_ex

函数功能

设置AE曝光分配扩展属性，支持分别设置AE分配策略中的sensor模拟增益，sensor数字增益和ISP数字增益。

函数原型

```
hi_s32 hi_mpi_isp_set_ae_route_attr_ex(hi_vi_pipe vi_pipe, const  
hi_isp_ae_route_ex *ae_route_attr_ex)
```

约束说明

- 此接口用于设定AE曝光分配扩展属性，AE计算得到的曝光量将按照设定的路线进行分配，用户可以根据自己的需求设定曝光时间优先、sensor模拟增益优先、sensor数字增益优先、ISP数字增益优先和光圈优先。该接口可用于设置WDR模式下的曝光分配路线，减轻正常室内照度多帧合成WDR产生的工频闪现象，优化WDR模式图像效果。
- AE曝光分配扩展属性是否生效可通过配置hi_mpi_isp_set_exposure_attr接口中的ae_route_ex_valid来实现。ae_route_ex_valid为HI_TRUE时使用扩展AE route，否则使用正常AE route。
- AE扩展分配路线遵循以下限定：
 - 最大支持 16 个节点，每个节点有曝光时间、sensor模拟增益、sensor数字增益、ISP数字增益和光圈五个分量。
 - 节点中曝光时间的单位为us，不能设置为 0，也不能设置太小导致实际对应的曝光行数为 0，否则可能产生异常。
 - 光圈分量仅支持P-Iris，不支持DC-Iris，因为DC-Iris无法精确控制，所以DCIris和手动光圈镜头光圈分量是无效的。即光圈类型为DC-Iris时，节点光圈分量不会对曝光量分配产生任何影响。
 - 节点的曝光量是曝光时间、sensor模拟增益、sensor数字增益、ISP数字增益和光圈的乘积，节点曝光量单调递增，后一个节点的曝光量大于或等于前一个节点的曝光量，第一个节点的曝光量最小，最后一个节点的曝光量最大。
 - 如果相邻节点的曝光量增加，那么应该有一个分量增加，其他分量固定，增加的分量决定该段路线的分配策略。例如sensor模拟增益分量增加，那么该段路线的分配策略是sensor模拟增益优先。
 - 不支持设置等曝光量节点。
 - 用户可以根据不同的场景设置不同的路线，分配路线支持动态切换。
 - AE分配路线不能用于限制曝光参数的最大值和最小值。如果当前曝光量不在用户设定的路线范围当中，即曝光量超出了第一个节点或最后一个节点，曝光参数分配会超出节点限制，但不超过参数本身范围限制。
 - 针对DC-Iris和手动光圈镜头，默认AE扩展分配策略是先分配曝光时间，再分配sensor模拟增益、sensor数字增益，最后分配ISP数字增益。针对P-Iris镜头，默认AE扩展分配策略是先调节光圈，将光圈调至最大后调节曝光时间，

最后再分配sensor模拟增益、sensor数字增益和ISP 数字增益。如果当前曝光量不在用户设定的路线范围当中，按默认策略分配。

- 在线进行DC-Iris和P-Iris切换，扩展AE route会重置为与光圈类型相匹配的默认分配策略，用户可以根据需要在切换光圈类型时自行设置扩展 AE route。
- 2 合 1WDR模式下，优先帧为短帧且增益分开配置不使能时，扩展AE route 不生效。
- 自动降帧时，最大曝光时间的改变会更新到分配路线中。
- 线性模式与WDR模式切换时，若cmos.c里面设置了扩展AE route，且cmos.c中的扩展AE route标识符ae_route_ex_valid为HI_TRUE，则在切换后会采用cmos.c中的扩展AE route，否则采用默认的AE route。
- 帧率或分辨率切换时，若用户设置的最大曝光目标时间大于切换后 1 帧所允许的最大曝光时间，那么分配路线的最大曝光时间会更新为切换后 1 帧所允许的最大曝光时间。
- 发生自动降帧、线性与WDR模式切换、帧率或分辨率切换、优先帧切换、增益分开配置切换、限制曝光时间或增益的最大最小值等情况时，实际生效的扩展AE route可能与MPI设置的不一致，此时可以通过hi_mpi_isp_query_exposure_info获取实际生效的扩展AE route。

参数说明

参数名	输入/输出	说明
vi_pipe	输入	VI PIPE号。 取值范围：[0, 12)。
ae_route_attr_ex	输入	AE 曝光分配策略扩展属性结构体指针。

返回值说明

- 0：成功
- 非0：失败，参见[ISP返回码](#)。

参考样例

```
hi_vi_pipe vi_pipe = 0;
hi_isp_exposure_attr exp_attr;
hi_isp_ae_route_ex ae_route_attr_ex;
hi_u32 route_ex_node [6][5]
= {{ 30, 1024, 1024, 1024, 0},
{ 30, 1024, 1024, 1024, 10},
{ 30, 16384, 1024, 1024, 10},
{1000000, 16384, 1024, 1024, 10},
{1000000, 16384, 16384, 1024, 10},
{1000000, 16384, 16384, 4096, 10}};
hi_mpi_isp_get_ae_route_attr_ex(vi_pipe,&ae_route_attr_ex);
hi_mpi_isp_get_exposure_attr(vi_pipe, &exp_attr);
exp_attr.ae_route_ex_valid = HI_TRUE;
ae_route_attr_ex.total_num = 6;
memcpy(ae_route_attr_ex.route_ex_node, route_ex_node, sizeof(route_ex_node));
hi_mpi_isp_set_ae_route_attr_ex(vi_pipe, & ae_route_attr_ex);
hi_mpi_isp_set_exposure_attr(vi_pipe, & ae_route_attr_ex);
```

3.4.1.8 hi_mpi_isp_get_ae_route_attr_ex

函数功能

获取AE曝光分配策略扩展属性。

函数原型

```
hi_s32 hi_mpi_isp_get_ae_route_attr_ex(hi_vi_pipe vi_pipe, hi_isp_ae_route_ex *ae_route_attr_ex)
```

参数说明

参数名	输入/输出	说明
vi_pipe	输入	VI PIPE号。 取值范围：[0, 12)。
ae_route_attr_ex	输出	AE曝光分配策略扩展属性结构体指针。

3.4.1.9 hi_mpi_isp_set_ae_route_sf_attr

函数功能

WDR模式下，设置AE短帧的曝光分配策略属性。

函数原型

```
hi_s32 hi_mpi_isp_set_ae_route_sf_attr(hi_vi_pipe vi_pipe, const hi_isp_ae_route *ae_route_sf_attr)
```

约束说明

- 此接口用于WDR模式下设定AE短帧的曝光分配路线，AE计算得到的短帧的曝光量将按照设定的路线进行分配，用户可以根据自己的需求设定曝光优先、增益优先、光圈优先。
- AE分配路线遵循以下限定：
 - 最大支持 16 个节点，每个节点有曝光时间、增益、光圈三个分量，增益包含模拟增益、数字增益、ISP数字增益。
 - 节点中曝光时间的单位为us，不能设置为 0，也不能设置太小导致实际对应的曝光行数为 0，否则可能产生异常。
 - 光圈分量仅支持P-Iris，不支持DC-Iris，因为DC-Iris 无法精确控制，所以DCIris和手动光圈镜头光圈分量是无效的。即光圈类型为DC-Iris时，节点光圈分量不会对曝光量分配产生任何影响。光圈分量仅在优先帧为短帧时可控。
 - 节点的曝光量是曝光时间、增益和光圈的乘积，节点曝光量单调递增，后一个节点的曝光量大于或等于前一个节点的曝光量，第一个节点的曝光量最小，最后一个节点的曝光量最大。

- 如果相邻节点的曝光量增加，那么应该有一个分量增加，其他分量固定，增加的分量决定该段路线的分配策略。例如增益分量增加，那么该段路线的分配策略是增益优先。
- 不支持设置等曝光量节点。
- 用户可以根据不同的场景设置不同的路线，分配路线支持动态切换。
- AE分配路线不能用于限制曝光参数的最大值和最小值。如果当前曝光量不在用户设定的路线范围当中，即曝光量超出了第一个节点或最后一个节点，曝光参数分配会超出节点限制，但不超过参数本身范围限制。
- 针对DC-Iris和手动光圈镜头，默认AE分配策略是首先分配曝光时间，其次分配增益。针对P-Iris镜头，默认AE分配策略是首先调节光圈，将光圈调至最大后调节曝光时间，最后再分配增益。如果当前曝光量不在用户设定的路线范围当中，按默认策略分配。
- 在线进行DC-Iris和P-Iris切换，短帧AE route会重置为与光圈类型相匹配的默认分配策略，用户可以根据需要在切换光圈类型时自行设置短帧AE route。
- 优先帧为长帧且增益分开配置不使能、以及线性模式下，短帧AE route不生效。
- 自动降帧时，最大曝光时间的改变会更新到分配路线中。
- 线性模式与WDR模式切换时，若cmos.c里面设置了短帧AE route，则在切换后会采用cmos.c中的route，否则采用默认的短帧AE route。
- 2合1WDR模式下，帧率或分辨率切换时，若用户设置的最大曝光目标时间大于切换后1帧所允许的最大曝光时间的一半，那么分配路线的最大曝光时间会更新为切换后1帧所允许的最大曝光时间的一半。
- 发生自动降帧、线性与2合1WDR模式切换、帧率或分辨率切换、优先帧切换、增益分开配置切换、限制曝光时间或增益的最大最小值等情况时，实际生效的AE route可能与MPI设置的不一致，此时可以通过
hi_mpi_isp_query_exposure_info获取实际生效的短帧AE route。

参数说明

参数名	输入/输出	说明
vi_pipe	输入	VI PIPE号。 取值范围：[0, 12)。
ae_route_sf_attr	输入	AE短帧的曝光分配策略结构体指针。

返回值说明

- 0：成功
- 非0：失败，参见[ISP返回码](#)。

参考样例

```
hi_vi_pipe vi_pipe = 0;
hi_isp_ae_route ae_route_sf_attr;
hi_u32 route_node [3][3] = {{100,1024,1},{20000,1024,1},{20000,16384,1}};
hi_mpi_isp_set_ae_route_sf_attr(vi_pipe, & ae_route_sf_attr);
ae_route_sf_attr.total_num = 3;
memcpy(ae_route_sf_attr.route_node, route_node, sizeof(route_node));
hi_mpi_isp_set_ae_route_sf_attr(vi_pipe, & ae_route_sf_attr);
```

3.4.1.10 hi_mpi_isp_get_ae_route_sf_attr

函数功能

获取AE短帧曝光分配策略属性。

函数原型

```
hi_s32 hi_mpi_isp_get_ae_route_sf_attr(hi_vi_pipe vi_pipe, hi_isp_ae_route
*ae_route_sf_attr)
```

参数说明

参数名	输入/输出	说明
vi_pipe	输入	VI PIPE号。 取值范围：[0, 12)。
ae_route_sf_attr	输出	AE短帧的曝光分配策略结构体指针。

返回值说明

- 0：成功
- 非0：失败，参见[ISP返回码](#)。

3.4.1.11 hi_mpi_isp_set_ae_route_sf_attr_ex

函数功能

此接口用于WDR模式下设定AE短帧的曝光分配扩展属性，AE计算得到的短帧的曝光量将按照设定的路线进行分配，用户可以根据自己的需求设定曝光时间优先、sensor模拟增益优先、sensor数字增益优先、ISP数字增益优先和光圈优先。

函数原型

```
hi_s32 hi_mpi_isp_set_ae_route_sf_attr_ex(hi_vi_pipe vi_pipe, const
hi_isp_ae_route_ex *ae_route_sf_attr_ex)
```

约束说明

- 此接口用于WDR模式下设定AE短帧的曝光分配扩展属性，AE计算得到的短帧的曝光量将按照设定的路线进行分配，用户可以根据自己的需求设定曝光时间优先、sensor模拟增益优先、sensor数字增益优先、ISP数字增益优先和光圈优先。
- AE曝光分配扩展属性是否生效可通过配置hi_mpi_isp_set_exposure_attr接口中的ae_route_ex_valid来实现。ae_route_ex_valid为HI_TRUE时使用扩展AE route，否则使用正常AE route。
- AE扩展分配路线遵循以下限定：
 - 最大支持 16 个节点，每个节点有曝光时间、sensor模拟增益、sensor数字增益、ISP数字增益和光圈五个分量。

- 节点中曝光时间的单位为us，不能设置为 0，也不能设置太小导致实际对应的曝光行数为 0，否则可能产生异常。节点曝光时间应设置不超过sensor最大曝光时间的一半。
- 光圈分量仅支持P-Iris，不支持DC-Iris，因为DC-Iris无法精确控制，所以DCIris和手动光圈镜头光圈分量是无效的。即光圈类型为DC-Iris时，节点光圈分量不会对曝光量分配产生任何影响。光圈分量仅在优先帧为短帧时可控。
- 节点的曝光量是曝光时间、sensor模拟增益、sensor数字增益、ISP数字增益和光圈的乘积，节点曝光量单调递增，后一个节点的曝光量大于或等于前一个节点的曝光量，第一个节点的曝光量最小，最后一个节点的曝光量最大。
- 如果相邻节点的曝光量增加，那么应该有一个分量增加，其他分量固定，增加的分量决定该段路线的分配策略。例如sensor 模拟增益分量增加，那么该段路线的分配策略是sensor模拟增益优先。
- 不支持设置等曝光量节点。
- 用户可以根据不同的场景设置不同的路线，分配路线支持动态切换。
- AE分配路线不能用于限制曝光参数的最大值和最小值。如果当前曝光量不在用户设定的路线范围当中，即曝光量超出了第一个节点或最后一个节点，曝光参数分配会超出节点限制，但不超过参数本身范围限制。
- 针对DC-Iris和手动光圈镜头，默认 AE 扩展分配策略是先分配曝光时间，再分配sensor模拟增益、sensor数字增益，最后分配ISP数字增益。针对P-Iris镜头，默认AE扩展分配策略是先调节光圈，将光圈调至最大后调节曝光时间，最后再分配sensor模拟增益、sensor数字增益和ISP数字增益。如果当前曝光量不在用户设定的路线范围当中，按默认策略分配。
- 在线进行DC-Iris和P-Iris切换，扩展AE route会重置为与光圈类型相匹配的默认分配策略，用户可以根据需要在切换光圈类型时自行设置短帧扩展AE route。
- 优先帧为长帧且增益分开配置不使能、以及线性模式下，短帧扩展AE route不生效。
- 自动降帧时，最大曝光时间的改变会更新到分配路线中。
- 线性模式与WDR模式切换时，若cmos.c里面设置了短帧扩展AE route，且cmos.c中的扩展AE route标识符 ae_route_ex_valid为HI_TRUE，则在切换后会采用cmos.c中的短帧扩展AE route，否则采用默认的短帧AE route。
- 2 合 1WDR模式下，帧率或分辨率切换时，若用户设置的最大曝光目标时间大于切换后 1 帧所允许的最大曝光时间，那么分配路线的最大曝光时间会更新为切换后 1 帧所允许的最大曝光时间。
- 发生自动降帧、线性与WDR模式切换、帧率或分辨率切换、优先帧切换、增益分开配置切换、限制曝光时间或增益的最大最小值等情况时，实际生效的扩展AE route可能与MPI设置的不一致，此时可以通过hi_mpi_isp_query_exposure_info获取实际生效的短帧扩展AE route。

参数说明

参数名	输入/输出	说明
vi_pipe	输入	VI PIPE号。 取值范围：[0, 12)。
ae_route_sf_attr_ex	输入	AE短帧的曝光分配策略扩展属性结构体指针。

返回值说明

- 0：成功
- 非0：失败，参见ISP返回码。

参考样例

```
hi_vi_pipe vi_pipe = 0;
hi_isp_exposure_attr exp_attr;
hi_isp_ae_route_ex ae_route_sf_attr_ex;
hi_u32 route_ex_node [6][5]
= {{ 30, 1024, 1024, 1024, 0},
{ 30, 1024, 1024, 1024, 10},
{ 30, 16384, 1024, 1024, 10},
{20000, 16384, 1024, 1024, 10},
{20000, 16384, 16384, 1024, 10},
{20000, 16384, 16384, 4096, 10}};
hi_mpi_isp_get_ae_route_sf_attr_ex(vi_pipe, & ae_route_sf_attr_ex);
hi_mpi_isp_get_exposure_attr(vi_pipe, &exp_attr);
exp_attr.ae_route_ex_valid = HI_TRUE;
ae_route_sf_attr_ex.total_num = 6;
memcpy(ae_route_sf_attr_ex.route_ex_node, route_ex_node, sizeof(route_ex_node));
hi_mpi_isp_get_ae_route_sf_attr_ex(vi_pipe, & ae_route_sf_attr_ex);
hi_mpi_isp_set_exposure_attr(vi_pipe, &exp_attr);
```

3.4.1.12 hi_mpi_isp_get_ae_route_sf_attr_ex

函数功能

获取AE短帧曝光分配策略扩展属性。

函数原型

```
hi_s32 hi_mpi_isp_get_ae_route_sf_attr_ex(hi_vi_pipe vi_pipe,
hi_isp_ae_route_ex *ae_route_sf_attr_ex)
```

参数说明

参数名	输入/输出	说明
vi_pipe	输入	VI PIPE号。 取值范围：[0, 12)。
ae_route_sf_attr_ex	输出	AE曝光分配策略扩展属性结构体指针。

返回值说明

- 0：成功
- 非0：失败，参见ISP返回码。

3.4.1.13 hi_mpi_isp_query_exposure_info

函数功能

获取AE内部状态信息，包括全局 5 段直方图、1024 段直方图和平均亮度等统计信息，同时还可获取AE运行状态中的曝光时间、增益、曝光量和实际生效的AE route等信息。

约束说明

- 获取的曝光时间以微秒(us)为单位，获取的sensor模拟增益、sensor数字增益和ISP数字增益以倍数为单位，精度是10bit。
- 获取的曝光量=(曝光时间 * 曝光增益)，未考虑光圈状态，其中曝光时间以行数为单位，曝光增益包括sensor模拟增益、sensor数字增益和ISP数字增益，以倍数为单位，6bit小数精度。若该值的精度不足以满足需求，可以根据上面高精度的曝光时间(us)和增益(10bit 小数精度)重新计算一个曝光量。
- 可通过查询hist_error来获取AE是否稳定的信息，若hist_error的绝对值小于曝光容忍偏差值，意味着当前AE不会动作。
- 通过该接口获取的AE route与Proc信息中的AE route都是实际生效的值。只不过该接口的节点曝光时间以us为单位，而 Proc信息中的曝光时间以行数为单位；该接口的光圈分量为 F 值大小，取值范围为[0,10]，而Proc信息中的光圈分量为 F 值等效增益，大小为(1<=F 值)。扩展AE route以及短帧AE route和短帧扩展AE route也是同样的对应关系。
- 调用该接口需要确保系统已经运行起来并且统计信息已经生成；当系统刚运行起来第一帧，AE内部还没有统计信息时，调用该接口统计信息会返回全 0，这时候系统内存是足够的。
- 手动模式下，实际生效的again、dgain等其他值可能和hi_mpi_set_exposure_attr存在一定偏差，和具体sensor相关。

函数原型

hi_s32 hi_mpi_isp_query_exposure_info(**hi_vi_pipe** vi_pipe, **hi_isp_exp_info** *exp_info)

参数说明

参数名	输入/输出	说明
vi_pipe	输入	VI PIPE号。 取值范围：[0, 12)。
exp_info	输出	曝光内部状态信息结构体指针。

返回值说明

- 0：成功
- 非0：失败，参见[ISP返回码](#)。

参考样例

```
hi_vi_pipe vi_pipe = 0;
hi_isp_exp_info exp_info;
hi_mpi_isp_query_exposure_info(vi_pipe, &exp_info);
printf("Sensor exposure time: %d\n",exp_info.exp_time);
printf("Analog Gain: %d\n",exp_info.a_gain);
printf("Digital Gain: %d\n",exp_info.d_gain);
printf("ISP Gain: %d\n",exp_info.isp_d_gain);
printf("Exposure: %d\n",exp_info.exposure);
printf("Average Luminance: %d\n",exp_info.ave_lum);
printf("Hist error: %d\n",exp_info.hist_error);
exp_info.exposure_is_max? printf("Exposure is MAX!\n"):printf("Exposure is NOT MAX!\n");
for(i = 0; i < 1024; i++)
{
    printf("Hist1024Value[%d]: %d\n",i, exp_info.ae_hist1024_value [i]);
}
```

3.4.1.14 hi_mpi_isp_set_exp_convert

函数功能

设定等曝光量转换的曝光参数相关属性。

函数原型

```
hi_s32 hi_mpi_isp_set_exp_convert(hi_vi_pipe vi_pipe, hi_isp_exp_conv_param
*conv_param)
```

约束说明

作为输入的值，除了值要设定输入的vi_pipe号，还需对conv_param中的成员变量tar_fps进行设置目标帧率即可。

参数说明

参数名	输入/输出	说明
vi_pipe	输入	VI PIPE号。 取值范围：[0, 12)。
conv_param	输入	等曝光量转换相关曝光量属性指针。

3.4.1.15 hi_mpi_isp_get_exp_convert

函数功能

设定等曝光量转换的曝光参数相关属性。

函数原型

```
hi_s32 hi_mpi_isp_get_exp_convert(hi_vi_pipe vi_pipe, hi_isp_exp_conv_param
*conv_param)
```


约束说明

- 转换出来的sensor 曝光时间、模拟增益、数字增益均为sensor 寄存器的值并有对应的寄存器地址，可以直接配给sensor 寄存器使用。转换出来的ISP数字增益以倍数为单位，精度是 10bit。
- 转换出来的sensor曝光时间、模拟增益、数字增益分别最多为 10 个寄存器值和 10 个寄存器地址。

参数说明

参数名	输入/输出	说明
vi_pipe	输入	VI PIPE号。 取值范围：[0, 12)。
conv_param	输出	等曝光量转换相关曝光量属性指针。

3.4.2 AI 控制模块

3.4.2.1 hi_mpi_isp_set_iris_attr

函数功能

设定光圈的控制属性，该函数可实现手动光圈属性和光圈类型等参数的设置。

函数原型

```
hi_s32 hi_mpi_isp_set_iris_attr(hi_vi_pipe vi_pipe, const hi_isp_iris_attr *iris_attr)
```

约束说明

- 根据实际对接镜头光圈类型，设置正确的光圈类型属性，由此再去设置相关的DC-Iris/P-Iris控制属性。若对接的是手动光圈镜头，可将光圈类型设置为 HI_ISP_IRIS_DC_TYPE，建议此时关闭 AI 使能。
- 手动光圈属性主要用于调试，可通过该MPI进行设置。对于P-Iris镜头，手动 iris_fno值会受到最大、最小光圈目标值的影响。自动光圈属性的更多参数需要调用hi_mpi_isp_set_dciris_attr和hi_mpi_isp_get_piris_attr进行设置。

参数说明

参数名	输入/输出	说明
vi_pipe	输入	VI PIPE号。 取值范围：[0, 12)。
iris_attr	输入	光圈控制属性结构体指针。

返回值说明

- 0：成功
- 非0：失败，参见ISP返回码。

3.4.2.2 hi_mpi_isp_get_iris_attr

函数功能

获取光圈的控制属性。

函数原型

```
hi_s32 hi_mpi_isp_get_iris_attr(hi_vi_pipe vi_pipe, hi_isp_iris_attr *iris_attr)
```

参数说明

参数名	输入/输出	说明
vi_pipe	输入	VI PIPE号。 取值范围：[0, 12)。
iris_attr	输出	光圈控制属性结构体指针。

返回值说明

- 0：成功
- 非0：失败，参见ISP返回码。

3.4.2.3 hi_mpi_isp_set_dciris_attr

函数功能

设定DC-Iris AI算法的控制属性，该函数可实现DC-Iris自动光圈的参数设置。

函数原型

```
hi_s32 hi_mpi_isp_set_dciris_attr(hi_vi_pipe vi_pipe, const hi_isp_dciris_attr *dciris_attr)
```

约束说明

- DC-Iris光圈控制采用PID算法，算法根据画面亮度，调节PWM占空比对光圈大小进行控制。当曝光时间和增益达到最小目标值之后，会进入光圈控制区域。当光圈控制能满足目标亮度的要求时，AE直接返回，保持曝光时间和增益不变。当画面亮度稳定且PWM占空比维持在打开值一段时间后，AI算法会认为光圈已经打开至最大，退出光圈控制区，将控制权交还给AE。处于光圈控制区时，更改AE算法参数，如最大/最小曝光时间、最大/最小增益和抗闪等需要即时生效的参数，AE会即时响应，根据新设定的参数和环境亮度，AI算法重新决定是否要进入光圈控制区。由于进入光圈控制区域和退出光圈控制区域需要短暂时间，针对手动光圈镜头建议关闭AI功能，否则AE的调节速度会受到一点影响。针对DC-Iris镜头建议

一直打开AI功能，随意开关AI容易导致光圈控制出现异常。针对某些长焦的DC-Iris镜头，默认参数可能会导致光圈打开/关闭速度过快，此时可以调节相关参数来解决，详见hi_isp_dciris_attr部分描述。

- 关闭AI功能，对于DC-Iris镜头，光圈会打开到最大。

参数说明

参数名	输入/输出	说明
vi_pipe	输入	VI PIPE号。 取值范围：[0, 12)。
dciris_attr	输入	DC-Iris自动光圈控制属性结构体指针。

返回值说明

- 0：成功
- 非0：失败，参见[ISP返回码](#)。

3.4.2.4 hi_mpi_isp_get_dciris_attr

函数功能

获取DC-Iris自动光圈的控制属性。

函数原型

```
hi_s32 hi_mpi_isp_get_dciris_attr(hi_vi_pipe vi_pipe, hi_isp_dciris_attr *dciris_attr)
```

参数说明

参数名	输入/输出	说明
vi_pipe	输入	VI PIPE号。 取值范围：[0, 12)。
dciris_attr	输出	DC-Iris自动光圈控制属性结构体指针。

返回值说明

- 0：成功
- 非0：失败，参见[ISP返回码](#)。

3.4.2.5 hi_mpi_isp_set_piris_attr

函数功能

设定P-Iris自动光圈的属性。

函数原型

```
hi_s32 hi_mpi_isp_set_piris_attr(hi_vi_pipe vi_pipe, const hi_isp_piris_attr *piris_attr)
```

约束说明

- P-Iris自动光圈控制属性包含一个只写参数step_fno_table_change，建议先对该结构体赋值，set MPI接口一次之后再get MPI接口，直接get再set MPI可能会报错。
- P-Iris镜头光圈控制通过AE分配路线进行。P-Iris对接重点在于正确设置镜头相关参数和合理设置AE分配路线，详见 hi_isp_piris_attr和hi_mpi_isp_set_ae_route_attr部分描述，才能保证P-Iris正常工作。由于不同P-Iris的驱动方式可能会有差别，用户可以自行修改P-Iris驱动以适配不同镜头。
关闭AI功能，对于P-Iris镜头，光圈会打开到最大光圈目标值对应步进电机位置。

参数说明

参数名	输入/输出	说明
vi_pipe	输入	VI PIPE号。 取值范围：[0, 12)。
piris_attr	输入	P-Iris自动光圈控制属性结构体指针。

返回值说明

- 0：成功
- 非0：失败，参见[ISP返回码](#)。

参考样例

```
hi_vi_pipe vi_pipe = 0;
hi_isp_piris_attr piris_attr, piris_attr_def;
hi_u16 total_step_def = 93;
hi_u16 step_count_def = 62;
hi_u16 step_fno_table_def[1024] =
{30,35,40,45,50,56,61,67,73,79,85,92,98,105,112,120,127,135,143,150,158,166,174,183,191,200,20
8,217,225,234,243,252,261,270,279,289,298,307,316,325,335,344,353,362,372,381,390,399,408,41
7,426,435,444,453,462,470,478,486,493,500,506,512};
hi_isp_iris_f_no max_iris_fno_target_def = 9;
hi_isp_iris_f_no min_iris_fno_target_def = 5;
piris_attr_def.step_fno_table_change= HI_TRUE;
piris_attr_def.zero_is_max= HI_TRUE;
piris_attr_def.step_count= step_count_def;
piris_attr_def.total_step= total_step_def;
piris_attr_def.max_iris_fno_target = max_iris_fno_target_def;
piris_attr_def.min_iris_fno_target = min_iris_fno_target_def;
memcpy(piris_attr_def.step_fno_table, step_fno_table_def, sizeof(piris_attr_def.step_fno_table));
```

```
hi_mpi_isp_set_piris_attr(vi_pipe, &piris_attr_def);
hi_mpi_isp_get_piris_attr(vi_pipe, &piris_attr);
```

3.4.2.6 hi_mpi_isp_get_piris_attr

函数功能

获取P-Iris自动光圈的属性。

函数原型

```
hi_s32 hi_mpi_isp_get_piris_attr(hi_vi_pipe vi_pipe, hi_isp_piris_attr *piris_attr)
```

参数说明

参数名	输入/输出	说明
vi_pipe	输入	VI PIPE号。 取值范围：[0, 12)。
piris_attr	输出	P-Iris自动光圈控制属性结构体指针。

返回值说明

- 0：成功
- 非0：失败，参见ISP返回码。

3.4.3 数据类型

3.4.3.1 hi_isp_ae_mode

说明

定义自动曝光的模式。

定义

```
typedef enum {
    HI_ISP_AE_MODE_SLOW_SHUTTER = 0,
    HI_ISP_AE_MODE_FIX_FRAME_RATE = 1,
    HI_ISP_AE_MODE_BUTT
} hi_isp_ae_mode;
```

成员

成员名称	描述
HI_ISP_AE_MODE_SLOW_SHUTTER	自动降帧模式，即SLOW_SHUTTER模式。
HI_ISP_AE_MODE_FIX_FRAME_RATE	固定帧率模式。

注意事项

- 自动降帧模式是指自动曝光调节时会优先增大曝光时间来尽量减小增益。当sensor增益达到用户设置的最大值时，自动曝光调节会逐渐降帧率并延长曝光时间，持续到曝光时间等于自动曝光的最大时间为止。低照度环境下噪声较小但帧率会降低。
- 固定帧率模式是指自动曝光调节时保持帧率不变，低照度环境下噪声会较大。
- Built-In WDR模式和多帧合成WDR模式支持降帧。若是客户自己对接的LineWDR sensor，支持自动降帧处理需要修改 `cmos.c`，修改方法可以参考imx290的`cmos.c`文件。另外，在`cmos.c`中Built-In WDR模式降帧设置应和线性一样。

3.4.3.2 hi_isp_ae_strategy

说明

定义AE曝光策略模式。

定义

```
typedef enum {  
    HI_ISP_AE_EXP_HIGHLIGHT_PRIOR = 0,  
    HI_ISP_AE_EXP_LOWLIGHT_PRIOR = 1,  
    HI_ISP_AE_STRATEGY_MODE_BUTT  
} hi_isp_ae_strategy;
```

成员

成员名称	描述
HI_ISP_AE_EXP_HIGHLIGHT_PRIOR	高光优先曝光模式。
HI_ISP_AE_EXP_LOWLIGHT_PRIOR	低光优先曝光模式。

注意事项

- 线性模式下，昇腾AE算法默认曝光策略为高光优先，尽量避免画面过曝，但在宽动态场景时，暗处亮度较低。此时若主要关注暗处区域，可采用低光优先模式，但亮处容易过曝。
- WDR模式下，昇腾AE算法默认使用低光优先模式。在低光优先模式和高光优先模式都是控制长帧曝光，手动模式曝光时间、最大最小曝光时间、AE_Route、AE_Route_Ex中曝光时间参数影响长帧曝光时间。

3.4.3.3 hi_isp_ae_delay

说明

定义AE延时属性。

定义

```
typedef struct {  
    hi_u16 black_delay_frame;
```

```
hi_u16 white_delay_frame;  
} hi_isp_ae_delay;
```

成员

成员名称	描述
black_delay_frame	图像亮度小于目标亮度的时间超过black_delay_frame帧时，AE开始调节。
white_delay_frame	图像亮度大于目标亮度的时间超过white_delay_frame帧时，AE开始调节。

注意事项

- black_delay_frame/white_delay_frame设置越大，在AE调节步长相同的情况下，调节至目标亮度需要越长时间。
- 人眼对于过曝比较敏感，建议white_delay_frame要设置得比black_delay_frame小一些。
- 固定降帧时，若帧率较低，black_delay_frame/white_delay_frame要做出相应调整，建议不要设置过大，否则AE收敛时间会较长。

3.4.3.4 hi_isp_ae_range

说明

定义曝光时间或增益的最大值和最小值。

定义

```
typedef struct {  
    hi_u32 max;  
    hi_u32 min;  
} hi_isp_ae_range;
```

成员

成员名称	描述
max	最大值。
min	最小值。 最小值不能大于最大值。

3.4.3.5 hi_isp_antiflicker_mode

说明

定义抗闪模式。

定义

```
typedef enum {  
    HI_ISP_ANTIFLICKER_NORMAL_MODE = 0x0,  
    HI_ISP_ANTIFLICKER_AUTO_MODE= 0x1,  
    HI_ISP_ANTIFLICKER_MODE_BUTT  
} hi_isp_antiflicker_mode;
```

成员

成员名称	描述
HI_ISP_ANTIFLICKER_NORMAL_MODE	普通抗闪模式。
HI_ISP_ANTIFLICKER_NORMAL_MODE	自动抗闪模式。

注意事项

- HI_ISP_ANTIFLICKER_NORMAL_MODE为普通抗闪模式，曝光时间可以根据亮度进行调节，最小曝光时间固定为 1/120 sec（60Hz）或 1/100 sec(50Hz)，不受曝光时间最小值的限制。
 - 有灯光的环境：曝光时间可与光源频率相匹配，能够防止图像闪烁。
 - 高亮度的环境：亮度越高，要求曝光时间就最短。而普通抗闪模式的最小曝光时间不能匹配光源频率，产生过曝。
- HI_ISP_ANTIFLICKER_AUTO_MODE为自动抗闪模式，曝光时间可以根据亮度进行调节，最小曝光时间可达到sensor的最小曝光时间。与普通抗闪模式的差别主要在高亮度的环境体现。高亮度的环境：最小曝光时间可以达到sensor的最小曝光时间，能够有效抑制过曝，但此时抗闪失效。

3.4.3.6 hi_isp_antiflicker

说明

定义抗闪属性。

定义

```
typedef struct {  
    hi_bool enable;  
    hi_u8 frequency;  
    hi_isp_antiflicker_mode mode;  
} hi_isp_antiflicker;
```

成员

成员名称	描述
enable	enable为 HI_TRUE时使能图像抗闪，为HI_FALSE时不使能图像抗闪。
frequency	抗闪频率值。 取值范围：[0x0, 0xFF]，默认值为 0x0。
mode	抗闪模式，普通抗闪或自动抗闪。

注意事项

抗闪开启后曝光时间会受到最大/最小曝光时间的限制，若最小抗闪时间大于最大曝光时间，那么开启抗闪后曝光时间将被限制为最大曝光时间。开启抗闪时，AE Route中各节点的Again、DGain、ISPDGain参数至少有一个需小于对应参数范围的上限，避免特定场景下无法进行补偿导致图像闪烁。

3.4.3.7 hi_isp_subflicker

说明

定义ISP图像亚抗闪属性。

定义

```
typedef struct {  
    hi_bool enable;  
    hi_u8 luma_diff;  
} hi_isp_subflicker;
```

成员

成员名称	描述
enable	enable 为 HI_TRUE 时使能图像亚抗闪功能，为 HI_FALSE 时不使能图像亚抗闪。
luma_diff	抗闪程度设置，范围 [0x0, 0x64]。亚抗闪功能生效时，该值越大越接近于抗闪。

注意事项

- 强制抗闪模式时，最小曝光时间固定为 1/120 s(60Hz) 或 1/100 s(50Hz)，在某些场景（如室内对准室外窗户的背光场景）画面可能会过曝严重，但不抗闪画面工频闪又比较厉害。在这种情况下引入亚抗闪模式是为了取得过曝与闪烁之间的平衡。在强制抗闪模式下，当亚抗闪功能生效时，若画面亮度小于 (ae_compensation + luma_diff)，那么曝光时间仍会固定为最小抗闪时间 1/120 s (60Hz) 或 1/100 s (50Hz)以防止图像闪烁。若画面亮度大于 (ae_compensation + luma_diff)，则取消抗闪，调整画面目标亮度为 (ae_compensation + luma_diff)，通过引入一定程度的画面闪烁来避免画面过曝严重。

 **注意**

非背光场景 luma_diff 较小时（如小于 10）可能会导致曝光时间在最小抗闪点左右来回变化，导致画面轻微闪烁，建议设置该值不小于 20。实际上，若 luma_diff 设置较小，背光场景下相当于取消了抗闪功能，与自动抗闪模式下的效果相近，达不到亚抗闪的目的，因此该值只有设置一个稍大些的值才有意义。

- 只有在满足打开抗闪、强制抗闪模式且抗闪频率值不等于 0 的前提下，亚抗闪功能才能生效。

- 自动抗闪模式不支持亚抗闪。在自动抗闪模式下打开亚抗闪，再将其切换到抗闪模式时，在极亮场景时可能有一帧过曝。

3.4.3.8 hi_isp_fswdr_mode

说明

定义ISP FSWDR运行模式。

定义

```
typedef enum {  
    HI_ISP_FSWDR_NORMAL_MODE      = 0x0,  
    HI_ISP_FSWDR_LONG_FRAME_MODE  = 0x1,  
    HI_ISP_FSWDR_AUTO_LONG_FRAME_MODE = 0x2,  
    HI_ISP_FSWDR_MODE_BUTT  
} hi_isp_fswdr_mode;
```

成员

成员名称	描述
HI_ISP_FSWDR_NORMAL_MODE	正常FSWDR模式，此时AE和合成模块按照自动/手动曝光比进行工作。
HI_ISP_FSWDR_LONG_FRAME_MODE	长帧模式，此时AE将短帧曝光时间设置为最小值，长帧曝光时间接近 1 帧所允许的最大值，合成模块只输出长帧数据，可用于优化弱宽动态场景或低照度时的图像质量。
HI_ISP_FSWDR_AUTO_LONG_FRAME_MODE	自动长帧模式，此时AE短帧曝光时间不会超过sensor限制的长帧模式下短帧最大曝光时间，在曝光量超过sensor设置的阈值时，自动切换为长帧模式，合成模块只输出长帧数据。

注意事项

- 长帧模式仅在行模式WDR长帧优先模式下有效，自动长帧模式仅在 2to1 行模式WDR长帧优先模式下有效，并且为了保证图像质量，WDR模块的运动检测及长短帧融合阈值均为自动配置，不支持手动配置。在线将模式切换到WDR模式，默认为正常 FSWDR模式。
- 自动长帧模式下，手动曝光比生效，如果手动曝光比大于 1，即使曝光量大于设置的阈值也不会进入长帧模式；如果手动曝光比为 1，则会自动进入长帧模式。
- 长帧模式下，如果WDR模块开启FUSION模式，可能会导致亮区颜色偏粉或变灰，因此不推荐长帧模式与WDR模块 FUSION模式同时开启。
- 长帧模式和自动长帧模式仅支持在优先帧为长帧且增益分开配置不使能的情况下使用。

相关数据类型及接口

无

3.4.3.9 hi_isp_ae_fswdr_attr

说明

定义 isp fswdr 运行模式

定义

```
typedef struct {  
    hi_isp_fswdr_mode fswdr_mode;  
} hi_isp_ae_fswdr_attr;
```

成员

成员名称	描述
fswdr_mode	定义WDR的合成模式，详见 hi_isp_fswdr_mode 类型定义

3.4.3.10 hi_isp_ae_attr

说明

定义自动曝光属性。

定义

```
typedef struct {  
    hi_isp_ae_range exp_time_range;  
    hi_isp_ae_range a_gain_range;  
    hi_isp_ae_range d_gain_range;  
    hi_isp_ae_range ispd_gain_range;  
    hi_isp_ae_range sys_gain_range;  
    hi_u32 gain_threshold;  
    hi_u8 speed;  
    hi_u16 black_speed_bias;  
    hi_u8 tolerance;  
    hi_u8 compensation;  
    hi_u16 ev_bias;  
    hi_isp_ae_strategy ae_strategy_mode;  
    hi_u16 hist_ratio_slope;  
    hi_u8 max_hist_offset;  
    hi_isp_ae_mode ae_mode;  
    hi_isp_antiflicker antiflicker;  
    hi_isp_subflicker subflicker;  
    hi_isp_ae_delay ae_delay_attr;  
    hi_bool manual_exp_value;  
    hi_u32 exp_value;  
    hi_isp_fswdr_mode fswdr_mode;  
    hi_bool wdr_quick;  
    hi_u16 iso_cal_coef;  
} hi_isp_ae_attr;
```

成员

成员名称	描述
exp_time_range	曝光时间范围，设置最大值和最小值，以微秒(us)为单位。取值范围：[0x0, 0xFFFFFFFF]，具体范围与sensor相关。
a_gain_range	sensor模拟增益范围，设置最大值和最小值，10bit 小数精度。取值范围：[0x400, 0xFFFFFFFF]，具体范围与sensor相关。
d_gain_range	sensor数字增益范围，设置最大值和最小值，10bit 小数精度。取值范围：[0x400, 0xFFFFFFFF]，具体范围与sensor相关。
ispd_gain_range	ISP数字增益范围，设置最大值和最小值，10bit 小数精度。取值范围：[0x400, 0x40000]。
sys_gain_range	系统增益范围，设置最大值和最小值，10bit 小数精度。取值范围：[0x400, 0xFFFFFFFF]，具体范围与sensor相关。
gain_threshold	自动降帧时的系统增益门限值，10bit 小数精度。取值范围：[0x400, 0xFFFFFFFF]。
speed	自动曝光调整时的速度。 取值范围：[0x0, 0xFF]，默认值为 0x40。
black_speed_bias	画面由暗到亮AE调节速度的偏差值，该值越大，画面从暗到亮的速度越快。 取值范围：[0x0, 0xFFFF]，默认值为 0x90。
tolerance	自动曝光调整时对画面亮度的容忍偏差。取值范围：[0x0, 0xFF]，默认值为 0x2。
compensation	自动曝光调整时的目标亮度。 取值范围：[0x0, 0xFF]，默认值为 0x38。
ev_bias	自动曝光调整时的目标亮度偏差值，10bit小数精度。取值范围：[0x0, 0xFFFF]，默认值为0x400。
ae_strategy_mode	自动曝光策略，高光优先或低光优先。
hist_ratio_slope	感兴趣区域的权重。 取值范围：[0x0, 0xFFFF]，默认值为 0x80。
max_hist_offset	感兴趣区域对统计平均值影响的最大程度。 取值范围：[0x0, 0xFF]，默认值为 0x10。
ae_mode	自动曝光模式，自动降帧模式或固定帧率模式。
antiflicker	抗闪属性设置。默认抗闪不使能。
subflicker	亚抗闪属性设置。默认亚抗闪不使能。
ae_delay_attr	延时属性设置。默认black_delay_frame=8，white_delay_frame=0。

成员名称	描述
manual_exp_value	手动曝光量使能，该值为HI_TRUE时，AE算法采用exp_value作为当前曝光量进行曝光时间和增益等的分配，为HI_FALSE时采用自动计算的曝光量进行分配。默认为HI_FALSE。
exp_value	手动曝光量值，等于曝光时间*系统增益，其中曝光时间的单位为微秒(us)。 取值范围：(0x0, 0xFFFFFFFF]。
fswdr_mode	FSWDR运行模式。默认为HI_ISP_FSWDR_NORMAL_MODE。
wdr_quick	WDR模式下，AE算法从稳定状态（亮度误差小于等于容忍偏差值tolerance）重新调整时，默认前 50 帧调整会进行时域滤波，以调整更加平滑。该值为HI_TRUE时，取消 50 帧时域滤波，使 AE 收敛速度更快。默认为HI_FALSE。
iso_cal_coef	ISO标定系数，用于保证拍照所需DCF信息中显示的ISO是标准的，8bit精度。 取值范围：[0x0, 0xFFFF]，默认值为 0x100。

注意事项

- 自动模式下，更改手动曝光属性的值不会生效。
 - 自动曝光的最大最小时间及增益。
可根据不同的场景对曝光时间及增益进行限定，如有高速运动物体场景可限定最大曝光时间值为较小值，这样可减轻运动物体拖影现象。
-  注意

为了防止图像过曝区域颜色不饱和，ISP内部会根据sensor黑电平自动修正lsp_dgain值，因此即使限制lsp_dgain最大值为 1024，lsp_dgain实际生效值仍会大于1024。如果需要强制ISPDGain为 1024, 可以通过手动模式实现，但是此时图像可能会出现亮区偏粉的问题。
- 自动曝光的系统增益
非扩展路径下，若(sensor模拟增益最小值*sensor数字增益最小值*ISP数字增益最小值)小于系统增益最小值，则AE算法内部计算时最小增益会被限制到系统增益的最小值。若(sensor模拟增益最大值*sensor数字增益最大值*ISP数字增益最大值)大于系统增益最大值，则AE算法内部计算时最大增益会被限制到系统增益的最大值。推荐通过设置系统增益的最大、最小值进行增益限制，分别限制sensor模拟增益、sensor数字增益和ISP数字增益时，若把较高精度的ISP数字增益限制到 1 倍，容易导致闪烁。
扩展路径下，采用系统增益的最大/最小值结合曝光时间的范围来限制曝光量的范围，sensor模拟增益、sensor数字增益及ISP 数字增益不直接受系统增益的影响。实际上，AE算法内部利用系统增益来计算最大/最小曝光量，而并不是直接限制某项增益的值。比如说系统增益最大值限制为 1 倍，但 sensor模拟增益的最小值限制为 2 倍，则实际生效的结果以sensor模拟增益的限制为准，sensor数字增益及ISP数字增益的限制也同理。手动模式下，系统增益不起作用，不对某项增益的值进行限制。

- 自动降帧时的系统增益门限值
在SLOW_SHUTTER模式下，当系统增益达到所设置的门限值时，系统将自动进入SLOW_SHUTTER模式。
- speed用于设定自动曝光时的收敛速度，该值越大曝光的收敛速度越快，但也会导致收敛过程中出现反复震荡。对于 Panasonic mn34220这款sensor，受sensor性能限制，在 2 帧合成WDR模式下，建议speed设置不能超过 64，否则在光线剧烈变化场景时可能出现闪烁。
- black_speed_bias用于设定画面由暗到亮AE调节速度的偏差值。默认speed下，画面从亮到暗的速度会快于画面从暗到亮的速度，若想双方向的AE调节速度差不多，可以调高black_speed_bias。
- 亮度补偿属性compensation用于调节曝光的目标亮度。曝光亮度补偿值越大则图像亮度越高。
- 曝光偏差属性ev_bias用于在特殊场景下对画面的目标亮度进行微调，也可认为是更高精度的亮度补偿值，通过调节该值来改变画面目标亮度，真实生效的AE目标亮度为 $\text{compensation} * \text{ev\_bias} / 1024$ 。compensation不变时，该值越大则图像亮度越高。
- 曝光容忍偏差属性tolerance用于调节曝光对环境的灵敏度，曝光容忍偏差值越大则曝光越不敏感，且可能导致同一目标亮度值多次调节得到的亮度差异越大，所以该属性推荐不能设定过大。尤其当目标亮度较低时，tolerance较大，最终画面收敛亮度可能会有明显差异。通过改变ev_bias对画面目标亮度进行微调时，若tolerance较大，过小的ev_bias变化量可能会看不出调整效果。
- 曝光策略属性 ae_strategy_mode 用于选择对高光优先或低光优先的曝光策略。高光优先意味着对高光敏感，尽量避免画面过曝。低光优先意味着对低光敏感，尽量看清楚暗处区域，不管画面是否过曝。默认的曝光策略是高光优先，用户可根据场景需要进行调整。
- hist_ratio_slope用于设定感兴趣区域的权重。若是高光优先曝光模式，则该值设置的是高光区域的权重，该值越大，则意味着对高光区域越敏感。反之，若是低光优先模式，则该值设置的是低光区域的权重，该值越大，则意味着对暗处区域越敏感，建议 hist_ratio_slope的值设置不要超过 0x100。高光优先曝光模式下，如果该值过大，只要画面中有一小部分过曝区域，AE算法就会认为画面亮度远大于目标亮度，这样曝光量就会不断往下调，导致画面变得很黑，同时画面中的过曝区域也在快速变小，达到某个程度的话AE算法又会认为画面亮度比目标亮度小了，曝光量又会往上调，如此反复出现来回调的现象，看起来像是画面闪烁。类似的，在低光优先模式下，如果AE目标亮度设置得比较低，若画面中有黑色物体来回移动，对画面亮度也会有较大影响，此时调低曝光调整速度和提高容忍偏差tolerance能够缓解该问题。
- 自动曝光时，可设定感兴趣区域对统计平均值影响的最大程度max_hist_offset。
该值相当于对提高hist_ratio_slope时增加的权重做限制，若该值为0，无论hist_ratio_slope多大，也不会对高光或低光区域做特殊处理，此时的统计平均值就是原始值。通过合理设置该值，可以保证任何场景AE稳定后画面平均亮度都在一定范围内，在高光优先曝光模式下，如果该值设置较大，在对对比度稍高一些的场景，如晴天室外场景，有天空有树木，则可能导致整体画面亮度偏低，因为此时优先保证了亮区天空的效果，通过限制该值从而限制对亮区的倾斜程度可以解决该问题。
- 曝光策略切换时，最好同时更新hist_ratio_slope和max_hist_offset的值，否则这两个值会采用默认配置，效果可能与预期不符。
- 在做强光抑制方案时，例如在RGBIR模式下，建议设置成高光优先曝光，通过降低AE目标亮度，同时适当设置 hist_ratio_slope和max_hist_offset为较大值来抑制

强光，暗区则可以通过使能DRC来看清楚。低光优先则可以用于实现非指定区域的背光补偿。

- AE曝光控制类型为自动时，可设定曝光模式ae_mode。该值可设定为慢快门模式(SLOW_SHUTTER)或固定帧率模式。慢快门模式通常用于低照度场景下进行自动降帧，以减少画面噪声。
- 抗闪属性结构体antiflicker可用于设定抗闪使能，抗闪频率和抗闪模式等属性。FSWDR模式下，长帧抗闪有效，在优先帧为短帧且增益分开配置时，长短帧抗闪都有效。由于随着曝光比的变化，短帧曝光时间的最大值会受到限制，建议FSWDR模式使用自动抗闪模式。

注意：OV2718 WDR模式下只曝光一次，曝光时间只有短帧有效，因此抗闪及亚抗闪效果并不能达到预期。

- 亚抗闪属性结构体subflicker可用于设定亚抗闪使能及抗闪程度属性。若有自动光圈，建议关闭亚抗闪功能。
- AE曝光控制类型为自动时，可设定AE延时生效属性结构体ae_delay_attr。该值的合理设置可提高画面亮度的稳定性，防止快速运动物体经过导致画面亮度发生变化。低码率设置时可适当增加该值，以避免AE调节时出现块效应。
- 用户可以通过将manual_exp_value置为HI_TRUE，手动设置曝光量exp_value来屏蔽昇腾AE算法的曝光量调整部分，只用到曝光量分配部分。exp_value的值会受到最大、最小曝光量的限制。最大、最小曝光量分别对应最大、最小曝光时间与增益的乘积。
- WDR模式下，优先帧为长帧时，exp_time_range设置的是长帧的曝光时间范围，优先帧为短帧时，exp_time_range设置的是短帧的曝光时间范围。
- FSWDR运行模式改变时，需要相应修改cmos.c中的以下回调函数：
cmos_long_frame_mode_set, cmos_get_ae_default, cmos_fps_set,
cmos_get_inttime_max, 以保证长、短帧曝光时间在合理范围内。正常WDR和长帧模式切换时，会回调 cmos_get_ae_default, 更新曝光时间最大/最小值，增益最大/最小值，光圈最大/最小值和compensation等参数，同时，AE内部会将hist_ratio_slope和max_hist_offset更新为算法默认值。需要注意的是，此时曝光比相关设置不会更新，仍会保持上一次的值。另外，长帧模式打开时，DRC强度会逐渐收敛至Auto目标值。客户可以根据需要，在切换完成后调用MPI接口修改参数。

注意

平滑切换的前提是切换前/后短帧曝光量与长帧曝光量相等。当曝光比较大时，由于正常WDR模式会限制增益，而长帧模式为了保证低照度效果会放开增益限制，可能会导致切换前/后短帧曝光量与长帧曝光量不等，这样切换就会有几帧突然之间变亮或者变暗，属于正常现象。若要避免这种现象，可以在切换完成之后再放开增益限制。

- 长帧模式仅在行模式WDR下才有效，帧模式WDR把曝光比设置为1:1，关闭运动检测，即可达到与行模式WDR长帧模式一样的效果。建议非行模式WDR统一将fswdr_mode设置为HI_ISP_FSWDR_NORMAL_MODE。
- 行模式WDR只曝光一次的sensor不支持长帧模式，如OV2718。

3.4.3.11 hi_isp_me_attr

说明

定义手动曝光属性。

定义

```
typedef struct {  
    hi_op_mode exp_time_op_type;  
    hi_op_mode a_gain_op_type;  
    hi_op_mode d_gain_op_type;  
    hi_op_mode ispd_gain_op_type;  
    hi_u32 exp_time;  
    hi_u32 a_gain;  
    hi_u32 d_gain;  
    hi_u32 isp_d_gain;  
} hi_isp_me_attr;
```

成员

成员名称	描述
exp_time_op_type	手动曝光时间使能，默认值为HI_OP_MODE_AUTO。
a_gain_op_type	手动sensor模拟增益使能，默认值为HI_OP_MODE_AUTO。
d_gain_op_type	手动sensor数字增益使能，默认值为HI_OP_MODE_AUTO。
ispd_gain_op_type	手动ISP数字增益使能，默认值为HI_OP_MODE_AUTO。
exp_time	手动曝光时间，以微秒(us)为单位，默认值为0x4000。取值范围：[0x0, 0xFFFFFFFF]，具体范围与sensor相关。
a_gain	手动sensor模拟增益，10bit小数精度，默认值为0x400。取值范围：[0x400, 0xFFFFFFFF]，具体范围与sensor相关。
d_gain	手动sensor数字增益，10bit小数精度，默认值为0x400。取值范围：[0x400, 0xFFFFFFFF]，具体范围与sensor相关。
isp_d_gain	手动ISP数字增益，10bit小数精度，默认值为0x400。取值范围：[0x400, 0x40000]，具体范围与sensor相关。

注意事项

- 手动模式下，手动曝光时间和增益的值会受到自动模式下最大/最小曝光时间和增益的限制。
- 手动曝光使能参数有效时，必须设置相应的手动曝光参数，若不设置，则采用系统默认值。
- 增益单位为10bit小数精度的倍数，即1024代表1倍。
- 若曝光参数设置超出最大（小）值，将使用sensor支持的最大（小）值代替。

3.4.3.12 hi_isp_exposure_attr

说明

定义ISP曝光属性。

定义

```
typedef struct {  
    hi_bool    bypass;  
    hi_op_mode op_type;  
    hi_u8      ae_run_interval;  
    hi_bool    hist_stat_adjust;  
    hi_bool    ae_route_ex_valid;  
    hi_isp_me_attr manual_attr;  
    hi_isp_ae_attr auto_attr;  
    hi_isp_prior_frame prior_frame;  
    hi_bool    ae_gain_sep_cfg;  
    hi_bool    advance_ae;  
} hi_isp_exposure_attr;
```

成员

成员名称	描述
bypass	AE模块bypass功能使能，默认为HI_FALSE。
op_type	自动曝光或手动曝光开关，默认为HI_OP_MODE_AUTO。
ae_run_interval	AE算法运行的间隔，取值范围为[1, 255]。 - 取值为1时表示每帧都运行AE算法； - 取值为2时表示每2帧运行1次 AE 算法，依此类推。建议该值设置不要大于2，否则AE 调节速度会受到影响。WDR模式时，该值建议设置为1，这样AE收敛会更加平滑。该值默认为1。
hist_stat_adjust	1024段直方图统计方式是否根据场景进行调整开关，默认为HI_TRUE。
ae_route_ex_valid	AE扩展分配路线是否生效开关，HI_TRUE时使用AE扩展分配路线，否则使用普通AE分配路线。默认为HI_FALSE。
manual_attr	手动曝光属性结构体。
auto_attr	自动曝光属性结构体。
prior_frame	WDR模式下曝光路线生效的优先帧，线性模式下此参数需配置为LONG_FRAME。
ae_gain_sep_cfg	长短帧增益是否分开分配。
advance_ae	是否使能advance AE。

注意事项

- AE bypass为HI_TRUE时，AE模块被bypass，任何AE配置都不会对图像亮度产生影响。hi_isp_ae_result保持为AE bypass前一帧的值。
- 昇腾AE主要基于全局1024段直方图信息进行亮度调整，正常场景1024段直方图只统计Gb或Gr分量。hist_stat_adjust为 HI_TRUE时，若进入了大面积单色场景（如大面积红色或蓝色），AE算法会根据全局平均值信息对1024段直方图统计方式进行调整，把R分量或B分量也考虑进来，如此可以防止大面积红色或蓝色场景画面亮度偏高。hist_stat_adjust为HI_FALSE 时，AE算法不会调整1024段直方图统计方式。红外场景建议将hist_stat_adjust设置为HI_FALSE。
- WDR/线性模式切换时，ae_run_interval会重置为1，用户可以在切换完成后，根据需要修改该值。
- ae_gain_sep_cfg仅在 2to1WDR模式下配置生效。
- 手动模式不支持长短帧增益分开分配。
- 曝光比变化过程中切换ae_gain_sep_cfg会交叉影响长短帧的曝光时间限制，可能会导致图像亮度突变，建议在曝光比收敛状态下切换ae_gain_sep_cfg。
- Advance AE目前可在正常FSWDR模式下生效（长帧模式不生效），使能advance_ae，可使图像暗区和亮区的曝光更加合理、准确，当场景发生亮暗比例变化时，图像亮暗区曝光更加稳定，在其他模式下advance_ae使能与否效果无差异。

3.4.3.13 hi_isp_wdr_exposure_attr

说明

定义WDR模式下的曝光属性。

定义

```
typedef struct {  
    hi_op_mode exp_ratio_type;  
    hi_u32 exp_ratio[HI_ISP_EXP_RATIO_NUM];  
    hi_u32 exp_ratio_max;  
    hi_u32 exp_ratio_min;  
    hi_u16 tolerance;  
    hi_u16 speed;  
    hi_u16 ratio_bias;  
    hi_u16 high_light_target;  
    hi_u16 exp_coef_min;  
} hi_isp_wdr_exposure_attr;
```

成员

成员名称	描述
exp_ratio_type	仅在多帧合成WDR模式下有效。

成员名称	描述
exp_ratio[HI_ISP_EXP_RATIO_NUM]	仅在多帧合成WDR模式下有效。 - 当exp_ratio_type为HI_OP_MODE_AUTO时，exp_ratio无效。 - 当exp_ratio_type为HI_OP_MODE_MANUAL时，exp_ratio为可读写，表示多帧合成WDR相邻2帧曝光比期望值。其中exp_ratio[0]作为S/VS曝光比，exp_ratio[1]作为M/S曝光比，exp_ratio[2]作为L/M曝光比。 上述VS，S，M，L分别表示极短帧，短帧，中帧和长帧的曝光时间，2帧合成时只有VS和S；3帧合成时采用VS，S和M；4帧合成时采用VS，S，M和L。6bit小数精度，0x40表示曝光比为1倍。 取值范围：[0x40, 0xFFFF] #define HI_ISP_EXP_RATIO_NUM 3
exp_ratio_max	仅在多帧合成WDR模式下有效。 - 当exp_ratio_type为HI_OP_MODE_AUTO时，exp_ratio_max表示最长帧与最短帧曝光时间比值的最大值。即2帧合成时表示S/VS曝光比的最大值，3帧合成时表示M/VS曝光比的最大值，4帧合成时表示L/VS曝光比的最大值。 - 当exp_ratio_type为HI_OP_MODE_MANUAL时，exp_ratio_max无效。 6bit小数精度，0x40表示曝光比为1倍。 取值范围：[0x40, 0x4000]
exp_ratio_min	仅在多帧合成WDR模式下有效。 - 当exp_ratio_type为HI_OP_MODE_AUTO时，exp_ratio_min表示长帧曝光时间与短帧曝光时间比值的最小值。 - 当exp_ratio_type为HI_OP_MODE_MANUAL时，exp_ratio_min无效。 格式为无符号6.6bit定点，0x40表示长帧曝光时间与短帧曝光时间的比值为1倍。默认值为0x40。 取值范围：[0x40, exp_ratio_max]
tolerance	曝光比容忍值，仅在两帧合成WDR模式下有效。 当exp_ratio_type为HI_OP_MODE_AUTO时，该值越大，表示场景动态范围变化在一定范围内时，曝光比保持不变。默认值为0xC。 取值范围：[0x0, 0xFF]
speed	自动曝光比调节速度，仅在两帧合成WDR模式下有效。 当exp_ratio_type为HI_OP_MODE_AUTO时，该值越大，自动曝光比调节速度越快。默认值为0x20。 取值范围：[0x0, 0xFF]

成员名称	描述
ratio_bias	曝光比偏差值，仅在多帧合成WDR模式下有效。 当exp_ratio_type为HI_OP_MODE_AUTO时，该值越大，自动曝光比越大。默认值为0x400，表示不对自动曝光比算法的计算结果进行调整。经过该值调整的曝光比会受到曝光比最大/最小值的限制。 取值范围：[0x0, 0xFFFF]
high_light_target	图像过曝程度的目标值，仅在使能Advance ae时有效。该值越大，图像过曝程度越大。默认值为 0x30。 取值范围：[0x0, 0x400]
exp_coef_min	曝光系数最小值，仅在使能advance ae时有效。 10bit小数精度，0x400表示曝光系数最小值为1倍，默认值为0x400。 取值范围：[0x0, 0x400]

注意事项

- 建议tolerance不要设置为太大的值，避免曝光比不能正确的反映场景动态范围变化。
- 针对某些对短帧最大曝光时间有限制的sensor，曝光比较小时，长帧最大曝光时间较短，图像动态范围及噪声表现较差，导致自动曝光比计算不准确，此时建议限制最小曝光比，保证长帧最大曝光时间至少达到3ms。如SonyIMX274@30fps，短帧最大曝光时间为680us，此时建议限制最小曝光比为1:5，即设置exp_ratio_min为0x140，以保证图像质量。
- 建议speed不要设置小于0x8，避免某些场景由于计算精度不足导致曝光比调节太慢甚至不调。speed过大可能导致曝光比变化太快，造成画面亮度出现振荡。
- 建议手动模式下exp_ratio不要设置大于0x400。如果曝光比大于0x400，在较亮的超宽动态场景，适当调高曝光比会优化长帧图像噪声表现。但是在较暗或低宽动态场景，如果曝光比过大，由于短帧最大曝光时间会被压缩，导致图像噪声表现会变差，而且会出现明显的噪声不连续，运动表现也会变差。
- 使能Advance ae，当曝光比达到设置的最大曝光比，图像过曝程度仍未达到目标值时，会适当降低AE亮度目标值，以换取图像整体过曝程度的降低。通过合理设置exp_coef_min，可使AE亮度目标值最多可以下降到compensation * exp_coef_min/1024。

3.4.3.14 hi_isp_ae_route_node

说明

定义AE分配路线节点属性。

定义

```
typedef struct {  
    hi_u32 int_time;  
    hi_u32 sys_gain;  
    hi_isp_iris_f_no iris_fno;
```

```
hi_u32 iris_fno_lin;  
} hi_isp_ae_route_node;
```

成员

成员名称	描述
int_time	节点曝光时间，单位为微秒(us)。 取值范围：(0x0, 0xFFFFFFFF]
sys_gain	节点增益，包括sensor模拟增益，sensor数字增益和ISP数字增益，10bit精度。 取值范围：[0x400, 0xFFFFFFFF]
iris_fno	节点光圈F值大小，仅支持P-Iris，不支持DC-Iris。取值范围： [HI_ISP_IRIS_F_NO_32_0, HI_ISP_IRIS_F_NO_1_0]。
iris_fno_lin	节点光圈 F 值等效增益大小，仅支持P-Iris，不支持DC-Iris。 取值范围：[1, 1024]

注意事项

- 节点的曝光量是曝光时间、增益和光圈的乘积，节点曝光量必须单调递增，即后一个节点的曝光量大于或等于前一个节点的曝光量，第一个节点的曝光量最小，最后一个节点的曝光量最大。在计算曝光量时，光圈 F 值要等效成一个增益，公式如下：等效增益FNO = 1 <<HI_ISP_IRIS_F_NO_XX_XX，所支持的最小光圈为 F32.0。由此可知F32.0对应增益1，F22.0对应增益2，F16.0对应增益4，以此类推，F1.0对应增益1024。
- 可以通过设置结构体 [hi_isp_piris_attr](#)中的fno_ex_valid来决定实际生效的AE route节点光圈值采用iris_fno或 iris_fno_lin。fno_ex_valid为HI_TRUE时采用高精度的iris_fno_lin，默认采用iris_fno。iris_fno会受到hi_isp_piris_attr 中 max_iris_fno_target/min_iris_fno_target的限制，iris_fno_lin会受到 hi_isp_piris_attr中 max_iris_fno_target_linear/min_iris_fno_target_linear的限制。另外，iris_fno和iris_fno_lin还会受到 hi_isp_ae_sensor_default中 max_iris_fno/max_iris_fno的限制，所以对接P-Iris时要在cmos.c中给 max_iris_fno/min_iris_fno赋合适的值。
- 不支持设置等曝光量节点。
- 为了保证曝光节点的曝光量，曝光节点的一个分量发生限制时，会对其他未达到最大值的分量进行调整，实际生效路径可能与设定路径不一致。为了防止曝光量溢出，如果光圈分量使能，则同一节点的曝光时间与10bit精度系统增益的乘积最大值不能超过0x1FFFFFFFFFFFFFFF，如果光圈分量不使能，则同一节点的曝光时间与10bit精度系统增益的乘积最大值不能超过 0x7FFFFFFFFFFFFFFF。
- 如果相邻节点的曝光量增加，则应该有一个分量增加，其他分量固定，增加的分量决定该段路线的分配策略。例如增益分量增加，那么该段路线的分配策略是增益优先。
- 光圈分量仅支持P-Iris，不支持DC-Iris，因为DC-Iris无法精确控制。
- 针对P-Iris，建议将第一个节点的曝光时间、增益和光圈F值都设置为相应的最小目标值，最后一个节点的曝光时间、增益和光圈F值都设置为相应的最大目标值。

3.4.3.15 hi_isp_ae_route

说明

定义AE曝光分配策略属性。

定义

```
typedef struct {  
    hi_u32 total_num;  
    hi_isp_ae_route_node route_node[HI_ISP_AE_ROUTE_MAX_NODES];  
} hi_isp_ae_route;
```

成员

成员名称	描述
total_num	曝光分配路线节点数目，目前最大为16。
route_node[HI_ISP_AE_ROUTE_MAX_NODES]	曝光分配路线节点属性。 #define HI_ISP_AE_ROUTE_MAX_NODES 16

注意事项

- 最大支持16个节点，每个节点有曝光时间、增益、光圈三个分量，增益包含sensor模拟增益、sensor数字增益、ISP数字增益。
- 用户可以根据不同的场景设置不同的路线，分配路线支持动态切换。
- 针对DC-Iris和手动光圈镜头，默认AE分配策略是曝光时间优先，其次分配增益。如果当前曝光量不在用户设定的路线范围当中，按默认策略分配。在DC_Iris调节时，曝光量调节并不会参考AE_ROUTE，因此实际生效的AE_ROUTE不会更新。
- 针对P-Iris，默认AE分配策略是首先调节光圈，将光圈调至最大后调节曝光时间，最后再分配增益。如果当前曝光量不在用户设定的路线范围当中，按默认策略分配。
- 在线进行DC-Iris和P-Iris切换，AE route会重置为与光圈类型相匹配的默认分配策略，用户可以根据需要在切换光圈类型时自行设置AE route。默认分配策略的AE route参数与切换时的AE曝光时间最小/最大值，增益最大/最小值和光圈分量最大/最小值相对应。注意：由于cmos.c也可以设置AE分配路线，若要在线进行DC-Iris和P-Iris切换，请不要在cmos.c中给iris_type赋值，这样AE算法设置的光圈类型就会传递到cmos.c中，此时回调cmos_get_ae_default或cmos_fps_set等函数，才能得到与光圈类型相匹配的AE route。

3.4.3.16 hi_isp_ae_route_ex_node

说明

定义AE扩展分配路线节点属性。

定义

```
typedef struct {  
    hi_u32 int_time;
```

```
hi_u32 a_gain;  
hi_u32 d_gain;  
hi_u32 isp_d_gain;  
hi_isp_iris_f_no iris_fno;  
hi_u32 iris_fno_lin;  
} hi_isp_ae_route_ex_node;
```

成员

成员名称	描述
int_time	节点曝光时间，单位为微秒(us)。 取值范围：(0x0, 0xFFFFFFFF]
a_gain	sensor模拟增益，10bit精度。取值范围：[0x400, 0x3FFFFFF]
d_gain	sensor数字增益，10bit精度。取值范围：[0x400, 0x3FFFFFF]
isp_d_gain	ISP数字增益，10bit精度。 取值范围：[0x400, 0x40000]
iris_fno	节点光圈 F 值大小，仅支持P-Iris，不支持DC-Iris。 取值范围：[HI_ISP_IRIS_F_NO_32_0, HI_ISP_IRIS_F_NO_1_0]
iris_fno_lin	节点光圈F值等效增益大小，仅支持P-Iris，不支持DC-Iris。 取值范围：[1, 1024]

注意事项

- 节点的曝光量是曝光时间、sensor模拟增益、sensor数字增益、ISP数字增益和光圈的乘积，节点曝光量必须单调递增，即后一个节点的曝光量大于或等于前一个节点的曝光量，第一个节点的曝光量最小，最后一个节点的曝光量最大。在计算曝光量时，光圈 F 值要等效成一个增益，公式如下：等效增益FNO = 1 << HI_ISP_IRIS_F_NO_XX_XX。由此可知F32.0对应增益1，F22.0对应增益2，F16.0对应增益4，以此类推，F1.0对应增益1024。
- 可以通过设置结构体hi_isp_piris_attr中的fno_ex_valid来决定实际生效的扩展AE route节点光圈值采用iris_fno或 iris_fno_lin。fno_ex_valid为HI_TRUE时采用高精度的iris_fno_lin，默认采用iris_fno。iris_fno会受到hi_isp_piris_attr中max_iris_fno_target/ min_iris_fno_target的限制，iris_fno_lin会受到hi_isp_piris_attr中max_iris_fno_target/ min_iris_fno_target的限制。另外，iris_fno和iris_fno_lin还会受到hi_isp_ae_sensor_default中max_iris_fno/ min_iris_fno的限制，所以对接P-Iris时要在cmos.c中给max_iris_fno/ min_iris_fno赋合适的值。
- 不支持设置等曝光量节点。
- 为了保证曝光节点的曝光量，曝光节点的一个分量发生限制时，会对其他未达到最大值的分量进行调整，实际生效路径可能与设定路径不一致。
- 如果相邻节点的曝光量增加，则应该有一个分量增加，其他分量固定，增加的分量决定该段路线的分配策略。例如sensor模拟增益分量增加，那么该段路线的分配策略是sensor模拟增益优先。
- 光圈分量仅支持P-Iris，不支持DC-Iris，因为DC-Iris无法精确控制。

- 针对P-Iris，建议将第一个节点的曝光时间、增益和光圈 F 值都设置为相应的最小目标值，最后一个节点的曝光时间、增益和光圈 F 值都设置为相应的最大目标值。
- 节点的曝光时间不能设置为0，也不能设置过小，导致以us为单位的曝光时间对应的曝光行数为0，否则可能产生异常。
- 节点增益的范围相比非扩展AE route有所缩小。此外，为防止曝光量溢出，sensor模拟增益、sensor数字增益和ISP数字增益的乘积等效为的10bit精度系统增益不能大于0xFFFFFFFF。如果光圈分量使能，则同一节点的曝光时间与10bit精度系统增益的乘积最大值不能超过0x1FFFFFFFFFFFFF，如果光圈分量不使能，则同一节点的曝光时间与10bit精度系统增益的乘积最大值不能超过0x7FFFFFFFFFFFFF。
- 若某段路线需要设置ISP数字增益优先，请注意不要把ISP数字增益用满，因为ISP数字增益还有弥补分配精度的作用，建议至少留下2倍ISP数字增益的余量，供弥补精度使用，即分配路线中节点的lsp_d_gain不要大于max_lsp_d_gain/2。
- 开启增益分开配置功能时，曝光节点中的ISP数字增益在实际分配时可能有部分被分配到WDRGain。

3.4.3.17 hi_isp_ae_route_ex

说明

定义AE曝光分配策略扩展属性。

定义

```
typedef struct {  
    hi_u32 total_num;  
    hi_isp_ae_route_ex_node route_ex_node[HI_ISP_AE_ROUTE_EX_MAX_NODES];  
} hi_isp_ae_route_ex;
```

成员

成员名称	描述
total_num	曝光扩展分配路线节点数目，目前最大为16。
route_ex_node[HI_ISP_AE_ROUTE_EX_MAX_NODES]	曝光扩展分配路线节点属性。 #define HI_ISP_AE_ROUTE_EX_MAX_NODES 16

注意事项

- 最大支持16个节点，每个节点有曝光时间、sensor模拟增益、sensor数字增益、ISP数字增益和光圈五个分量。
- 用户可以根据不同的场景设置不同的路线，分配路线支持动态切换。
- 针对DC-Iris和手动光圈镜头，默认AE扩展分配策略是曝光时间优先，再分配sensor模拟增益、sensor数字增益、最后分配ISP数字增益。如果当前曝光量不在用户设定的路线范围当中，按默认策略分配。在DC_Iris调节时，曝光量调节并不会参考AE_ROUTE_EX，因此实际生效的AE_ROUTE_EX不会更新。

- 针对P-Iris，默认AE扩展分配策略是首先调节光圈，将光圈调至最大后调节曝光时间，最后再分配sensor模拟增益、sensor 数字增益和ISP数字增益。如果当前曝光量不在用户设定的路线范围当中，按默认策略分配。
- 在线进行DC-Iris和P-Iris切换，扩展AE route会重置为与光圈类型相匹配的默认分配策略，用户可以根据需要在切换光圈类型时自行设置扩展AE route。默认分配策略的AE route参数与切换时的AE曝光时间最小/最大值，增益最大/最小值和光圈分量最大/最小值相对应。注意：由于cmos.c也可以设置AE分配路线，若要在线进行DC-Iris和P-Iris切换，请不要在cmos.c 中给iris_type赋值，这样AE算法设置的光圈类型就会传递到cmos.c中，此时回调cmos_get_ae_default或cmos_fps_set等函数，才能得到与光圈类型相匹配的扩展AE route。

3.4.3.18 hi_isp_exp_info

说明

定义ISP曝光内部状态信息。

定义

```
typedef struct {
    hi_u32 exp_time;
    hi_u32 short_exp_time;
    hi_u32 median_exp_time;
    hi_u32 long_exp_time;
    hi_u32 a_gain;
    hi_u32 d_gain;
    hi_u32 a_gain_sf;
    hi_u32 d_gain_sf;
    hi_u32 isp_d_gain;
    hi_u32 exposure;
    hi_bool exposure_is_max;
    hi_s16 hist_error;
    hi_u32 ae_hist1024_value[HI_ISP_HIST_NUM];
    hi_u8 ave_lum;
    hi_u32 lines_per500ms;
    hi_u32 piris_fno;
    hi_u32 fps;
    hi_u32 iso;
    hi_u32 isosf;
    hi_u32 iso_calibrate;
    hi_u32 ref_exp_ratio;
    hi_u32 wdr_exp_coef;
    hi_u32 first_stable_time;
    hi_u32 quick_star_iso;
    hi_isp_ae_route ae_route;
    hi_isp_ae_route_ex ae_route_ex;
    hi_isp_ae_route ae_route_sf;
    hi_isp_ae_route_ex ae_route_sf_ex;
} hi_isp_exp_info;
```

成员

成员名称	描述
exp_time	当前曝光时间，单位为微秒(us)。 取值范围：[0x0, 0xFFFFFFFF]

成员名称	描述
short_exp_time	FSWDR模式下，表示当前短帧(S)曝光时间，单位为微秒(us)。线性模式不用关注该值。 取值范围：[0x0, 0xFFFFFFFF]
median_exp_time	FSWDR模式下，表示当前中帧(M)曝光时间，单位为微秒(us)。线性模式不用关注该值。 取值范围：[0x0, 0xFFFFFFFF]
long_exp_time	FSWDR模式下，表示当前长帧曝光时间，单位为微秒(us)。 取值范围：[0x0, 0xFFFFFFFF]
a_gain	当前sensor模拟增益，10bit小数精度。 取值范围：[0x400, 0xFFFFFFFF]
d_gain	当前sensor数字增益，10bit小数精度。 取值范围：[0x400, 0xFFFFFFFF]
a_gain_sf	当前sensor短帧模拟增益，10bit小数精度。 取值范围：[0x400, 0xFFFFFFFF]
d_gain_sf	当前sensor短帧数字增益，10bit小数精度。 取值范围：[0x400, 0xFFFFFFFF]
isp_d_gain	当前ISP数字增益，10bit小数精度。 取值范围：[0x400, 0xFFFFFFFF]
exposure	当前曝光量，等于曝光时间与曝光增益的乘积，其中曝光时间的单位为微秒(us)。 取值范围：[0x0, 0xFFFFFFFF]
exposure_is_max	0：ISP未达到最大曝光水平； 1：ISP达到最大曝光水平。
hist_error	统计信息，AE的目标亮度值与实际值的差，该值为正表示当前期望的亮度信息大于实际的亮度信息，该值为负表示期望的亮度信息小于实际的亮度信息。 取值范围：[-0x8000, 0x7FFF]
ae_hist1024_value [HI_ISP_HIST_NUM]	全局1024段直方图统计信息 取值范围：[0x0, 0xFFFFFFFF] #define HI_ISP_HIST_NUM 1024
ave_lum	平均亮度信息 取值范围：[0x0, 0xFF]
lines_per500ms	当前每500ms对应的曝光行数，可用于将曝光时间的单位由us转换成行数。 取值范围：[0x0, 0xFFFFFFFF]

成员名称	描述
piris_fno	当前P-Iris光圈 F 值对应的等效增益。 取值范围：[0x0, 0x400]
fps	实际图像帧率 * 100。 取值范围：[0x0, 0xFFFFFFFF]
iso	当前sensor模拟增益*sensor数字增益*ISP数字增益 *100，其中增益的精度都为 10bit。 取值范围：[0x64, 0xFFFFFFFF]
isof	当前sensor短帧模拟增益*sensor 短帧数字增益*ISP短帧数字增益*100，其中增益的精度都为10bit。 取值范围：[0x64, 0xFFFFFFFF]
iso_calibrate	标准ISO，用于拍照DCF信息显示。 $\text{iso\_calibrate} = \text{iso} * 256 / \text{iso\_cal\_coef}$ 。 取值范围：[0x0,0xFFFFFFFF]
ref_exp_ratio	参考曝光比，用于估计当前场景的动态范围，会受到 hi_isp_wdr_exposure_attr 中 tolerance 和 speed 等值的影响。 取值范围：[0x40, 0x4000]
wdr_exp_coef	基于原始亮度目标值调整的曝光系数，仅在使能advance ae时有效，值越小，图像整体亮度越低。AE实际亮度目标值为 $\text{compensation} * \text{wdr\_exp\_coef} / 1024$ 。 取值范围：[0x0, 0x400]
quick_star_iso	不带光敏AE启动快速收敛模式下，AE快速计算出当前亮度环境下，收敛完成后大致的 ISO 值，用户可根据此ISO值决定是否要打开IR CUT和IR LED。 取值范围：[0x64, 0xFFFFFFFF]
first_stable_time	首次AE收敛稳定的时间，单位为微秒(us)。 取值范围：[0x0, 0xFFFFFFFF]
ae_route	实际生效的AE route，各个节点中的曝光时间以us为单位，增益为10bit精度，光圈取值范围为 [HI_ISP_IRIS_F_NO_32_0, HI_ISP_IRIS_F_NO_1_0]。光圈类型为DC-Iris时，节点光圈值不会对曝光量分配产生影响。
ae_route_ex	实际生效的扩展AE route，各个节点中的曝光时间以us为单位，增益为10bit精度，光圈取值范围为 [HI_ISP_IRIS_F_NO_32_0, HI_ISP_IRIS_F_NO_1_0]。光圈类型为DC-Iris 时，节点光圈值不会对曝光量分配产生影响。

成员名称	描述
ae_route_sf	实际生效的短帧AE route，各个节点中的曝光时间以us为单位，增益为10bit精度，光圈取值范围为[HI_ISP_IRIS_F_NO_32_0, HI_ISP_IRIS_F_NO_1_0]。光圈类型为DC-Iris 时，节点光圈值不会对曝光量分配产生影响。
ae_route_sf_ex	实际生效的短帧扩展AE route，各个节点中的曝光时间以us为单位，增益为10bit精度，光圈取值范围为[HI_ISP_IRIS_F_NO_32_0, HI_ISP_IRIS_F_NO_1_0]。光圈类型为DC-Iris时，节点光圈值不会对曝光量分配产生影响。

注意事项

- 曝光量的计算未考虑光圈状态，为曝光时间、sensor模拟增益、sensor数字增益、ISP数字增益4者乘积，其中曝光时间的单位为曝光行数，曝光增益为6bit小数精度。若该值的精度不足以满足需求，可以根据高精度的曝光时间(us)和增益(10bit小数精度)重新计算一个曝光量。
- ISP是否达到最大曝光水平的计算未考虑光圈状态。如果当前曝光时间大于等于最大目标曝光时间，当前增益大于等于最大目标增益，那么就认为ISP达到最大曝光水平，否则就未达到最大曝光水平。
- 图像平均亮度经过归一化处理，取值范围为0~255。
- fps可用于查询自动降帧状态下的实际帧率。
- iso和ref_exp_ratio都可作为FSWDR模式下切换到长帧模式的参考量。不过需要注意的是，若以iso作为参考量，需要考虑到曝光比对于切换阈值的影响，避免来回在长帧模式和正常WDR模式之间切换；若以ref_exp_ratio作为参考量，长帧模式下亮区可能是过曝的，ref_exp_ratio已不能正确表示场景动态范围，这会影响到长帧模式切换回正常WDR模式的判别。线性模式下ref_exp_ratio值可作为场景动态范围参考，当前场景动态范围越小，ref_exp_ratio越小；当前场景动态范围越大，ref_exp_ratio越大。
- a_gain_sf、d_gain_sf、isosf在线性模式下默认为1024、1024、100，但从其他模式切换到线性模式时保留上一次的值不更新。
- 当长短帧增益分开配置时，iso、isosf的结果还包含对应帧的WDRGain。
- WDR模式下，优先帧为长帧时，exposure_is_max表示长帧是否达到最大曝光量，优先帧为短帧时，exposure_is_max表示短帧是否达到最大曝光量。

3.4.3.19 hi_isp_prior_frame

说明

WDR模式下的优先帧。

定义

```
typedef enum {  
    HI_ISP_LONG_FRAME    = 0,  
    HI_ISP_SHORT_FRAME   = 1,  
}
```

```
HI_ISP_PRIOR_FRAME_BUTT  
} hi_isp_prior_frame;
```

成员

成员名称	描述
HI_ISP_LONG_FRAME	优先帧为长帧。
HI_ISP_SHORT_FRAME	优先帧为短帧。

注意事项

- 优先帧为长帧时，优先根据长帧的曝光路线设置长帧的曝光参数，其中2合1WDR模式下当增益分开配置时，短帧的曝光路线会根据长帧的曝光参数进行调整，优先帧为短帧时同理。默认优先帧为长帧。
- 优先帧切换时，会回调cmos_get_ae_default，更新曝光时间最大/最小值，增益最大/最小值，光圈最大/最小值和AE compensation等参数，同时，AE内部会将hist_ratio_slope和max_hist_offset更新为算法默认值。客户可以根据需要，在切换完成后调用MPI接口修改参数。
- 线性模式下优先帧设置无效。

3.4.3.20 hi_isp_iris_status

说明

定义ISP光圈状态。

定义

```
typedef enum {  
    HI_ISP_IRIS_KEEP = 0,  
    HI_ISP_IRIS_OPEN = 1,  
    HI_ISP_IRIS_CLOSE = 2,  
    HI_ISP_IRIS_BUTT  
} hi_isp_iris_status;
```

成员

成员名称	描述
HI_ISP_IRIS_KEEP	光圈保持当前状态。
HI_ISP_IRIS_OPEN	光圈全开。
HI_ISP_IRIS_CLOSE	光圈全关。

注意事项

该值设置为HI_ISP_IRIS_OPEN或HI_ISP_IRIS_CLOSE时，光圈处于全开或全关的状态，可用于测试AI电路和驱动是否正确。OPEN和CLOSE的优先级高于AI使能和手动/自动模式，AI算法运行时，为保证其能够正常工作，需将该值设置为

HI_ISP_IRIS_KEEP。

3.4.3.21 hi_isp_iris_type

说明

定义 ISP 光圈类型。

定义

```
typedef enum {  
    HI_ISP_IRIS_FIXED_TYPE = 0,  
    HI_ISP_IRIS_DC_TYPE = 1,  
    HI_ISP_IRIS_P_TYPE = 2,  
    HI_ISP_IRIS_TYPE_BUTT,  
} hi_isp_iris_type;
```

成员

成员名称	描述
HI_ISP_IRIS_FIXED_TYPE	固定光圈
HI_ISP_IRIS_DC_TYPE	DC-Iris光圈。
HI_ISP_IRIS_P_TYPE	P-Iris光圈。

注意事项

- 必须设置正确的光圈类型，AI算法才能正常工作。
- 若对接的是手动光圈镜头，可将该值设置为HI_ISP_IRIS_DC_TYPE，建议此时关闭AI使能。
- 由DC-Iris切换至P-Iris，需提前设置好P-Iris镜头相关参数，并且将光圈状态设置为HI_ISP_IRIS_KEEP，再进行切换

3.4.3.22 hi_isp_iris_f_no

说明

定义ISP光圈F值。

定义

```
typedef enum {  
    HI_ISP_IRIS_F_NO_32_0 = 0,  
    HI_ISP_IRIS_F_NO_22_0,  
    HI_ISP_IRIS_F_NO_16_0,  
    HI_ISP_IRIS_F_NO_11_0,  
    HI_ISP_IRIS_F_NO_8_0,  
    HI_ISP_IRIS_F_NO_5_6,  
    HI_ISP_IRIS_F_NO_4_0,  
    HI_ISP_IRIS_F_NO_2_8,  
    HI_ISP_IRIS_F_NO_2_0,  
    HI_ISP_IRIS_F_NO_1_4,  
    HI_ISP_IRIS_F_NO_1_0,  
}
```

```
HI_ISP_IRIS_F_NO_BUTT,  
} hi_isp_iris_f_no;
```

成员

成员名称	描述	等效增益
HI_ISP_IRIS_F_NO_32_0	光圈F32.0。	1
HI_ISP_IRIS_F_NO_22_0	光圈F22.0。	2
HI_ISP_IRIS_F_NO_16_0	光圈F16.0。	4
HI_ISP_IRIS_F_NO_11_0	光圈F11.0。	8
HI_ISP_IRIS_F_NO_8_0	光圈F8.0。	16
HI_ISP_IRIS_F_NO_5_6	光圈F5.6。	32
HI_ISP_IRIS_F_NO_4_0	光圈F4.0。	64
HI_ISP_IRIS_F_NO_2_8	光圈F2.8。	128
HI_ISP_IRIS_F_NO_2_0	光圈F2.0。	256
HI_ISP_IRIS_F_NO_1_4	光圈F1.4。	512
HI_ISP_IRIS_F_NO_1_0	光圈F1.0。	1024

注意事项

针对P-Iris，AE算法根据分配路线计算曝光量时，光圈 F 值要等效成一个增益，公式如下：等效增益FNO = 1 << HI_ISP_IRIS_F_NO_XX_XX。由此可知F32.0对应增益1，F22.0对应增益2，F16.0对应增益4，以此类推，F1.0对应增益1024。

3.4.3.23 hi_isp_mi_attr

说明

定义手动光圈属性。

定义

```
typedef struct {  
    hi_u32 hold_value;  
    hi_isp_iris_f_no iris_fno;  
} hi_isp_mi_attr;
```

成员

成员名称	描述
hold_value	AI校正值，用于DC-Iris的调试。 取值范围为[0x0, 0x3E8]。

成员名称	描述
iris_fno	手动光圈大小，根据光圈F值进行区分，仅支持P-Iris，不支持DC-Iris。

注意事项

- 对接DC-Iris镜头时，若hi_isp_iris_status设置为HI_ISP_IRIS_KEEP，手动光圈使能，hold_value可用于DC-Iris的调试，此时PWM的占空比即为hold_value。
- 对接P-Iris镜头时，若hi_isp_iris_status设置为HI_ISP_IRIS_KEEP，手动光圈使能，iris_fno可用于P-Iris的调试，此时会控制P-Iris步进电机走到光圈F值与iris_fno最接近的位置。自动曝光模式下，P-Iris手动光圈不生效，此时若要固定光圈为某个F值，可以将max_iris_fnomax_iris_fno_target/min_iris_fnomin_iris_fno_target设置为相同值来实现。

3.4.3.24 hi_isp_dciris_attr

说明

定义DC-Iris AI算法属性。

定义

```
typedef struct {  
    hi_s32 kp;  
    hi_s32 ki;  
    hi_s32 kd;  
    hi_u32 min_pwm_duty;  
    hi_u32 max_pwm_duty;  
    hi_u32 open_pwm_duty;  
} hi_isp_dciris_attr;
```

成员

成员名称	描述
kp	比例增益，用于调节光圈的开关速度，该值越大光圈打开和关闭的速度越快。该值过小光线剧烈变化收敛过程中容易出现振荡，该值过大容易出现超调，也会导致振荡。该值的合理设置与电路特性和镜头相关。建议值为7000。 取值范围为[0, 100000]。
ki	积分增益，用于调节光圈的开关速度，该值越大光圈打开和关闭的速度越快。该值较小时，碰到强光收敛后画面会稳定在一个比较低的亮度上；该值较大时可能会导致强光场景光圈无法关闭。该值的合理设置与电路特性和镜头相关。建议值为100，该值一般不需要修改。 取值范围为[0, 1000]。

成员名称	描述
kd	微分增益，用于限制光线剧烈变化时光圈的开关速度，该值越大光线剧烈变化时光圈打开和关闭的速度越慢。该值过大对于瞬间变化的亮度过于敏感，会导致场景亮度快速变化时画面出现振荡。该值的合理设置与电路特性和镜头相关。建议值为3000。 取值范围为[0, 100000]。
min_pwm_duty	最小PWM占空比。该值越小过曝时光圈关闭速度越快，但容易导致光圈来回震荡。该值的合理设置与电路特性和镜头相关。建议值为250。 取值范围为[0, 1000]。
max_pwm_duty	最大PWM占空比。该值越大画面全黑时光圈打开速度越快，该值过小则可能导致退出光圈控制区域时光圈仍未达到最大，造成画面噪声严重。该值的合理设置与电路特性和镜头相关。建议值为 950。 取值范围为[0, 1000]。
open_pwm_duty	光圈打开时的PWM占空比。当画面亮度稳定并且PWM占空比大于该值一段时间后，退出光圈控制区域。所以该值不能太小，否则容易导致光圈未达到最大就退出了光圈控制区域，造成画面噪声严重。该值的合理设置与电路特性和镜头相关。建议值为800。 取值范围为[0, 1000]。

注意事项

- 当光圈关闭出现震荡时，一般意味着光圈关闭速度太快了，可以通过适当减小kp和增大min_pwm_duty来解决。
- open_pwm_duty的取值要求在min_pwm_duty和max_pwm_duty之间，必须确保该值能将光圈较快的打开。

3.4.3.25 hi_isp_piris_attr

说明

定义P-Iris属性。

定义

```
typedef struct {
    hi_bool step_fno_table_change;
    hi_bool zero_is_max;
    hi_u16 total_step;
    hi_u16 step_count;
    hi_u16 step_fno_table[HI_ISP_AI_MAX_STEP_FNO_NUM];
    hi_isp_iris_f_no max_iris_fno_target;
    hi_isp_iris_f_no min_iris_fno_target;
    hi_bool fno_ex_valid;
    hi_u32 max_iris_fno_target_linear;
    hi_u32 min_iris_fno_target_linear;
} hi_isp_piris_attr;
```

成员

成员名称	描述
step_fno_table_change	P-Iris步进电机位置与光圈 F 值映射表是否更新标志。该值为HI_TRUE时会更新P-Iris步进电机位置与F值映射表，为HI_FALSE时不更新。
zero_is_max	P-Iris步进电机Step 0是否对应最大光圈位置标志，取值与P-Iris镜头相关。该值为HI_TRUE时表示步进电机处于位置0时，光圈打开至最大，为HI_FALSE时表示步进电机处于位置0时光圈关闭。
total_step	P-Iris步进电机的总步数，具体大小与P-Iris镜头相关。取值范围为[1, 1024]。
step_count	P-Iris步进电机的可用步数，具体大小与P-Iris镜头相关。取值范围为[1, 1024]。
step_fno_table[HI_ISP_AI_MAX_STEP_FNO_NUM]	P-Iris步进电机位置与F值映射表，具体数据与P-Iris镜头相关。 取值范围为[1, 1024]。 #define HI_ISP_AI_MAX_STEP_FNO_NUM 1024
max_iris_fno_target	最大光圈目标值，可用于控制AE分配路线，实际生效光圈大小与P-Iris镜头相关。 取值范围为[HI_ISP_IRIS_F_NO_32_0, HI_ISP_IRIS_F_NO_1_0]。
min_iris_fno_target	最小光圈目标值，可用于控制AE分配路线，实际生效光圈大小与P-Iris镜头相关。 取值范围为[HI_ISP_IRIS_F_NO_32_0, HI_ISP_IRIS_F_NO_1_0]。
fno_ex_valid	对接P-Iris时，AE分配路线是否采用更高精度的光圈F值等效增益标志。该值为 HI_TRUE时表示AE分配路线采用高精度的光圈F值等效增益。默认为HI_FALSE。
max_iris_fno_target_linear	最大光圈F值等效增益目标值，可用于控制AE分配路线，实际生效光圈大小与P-Iris镜头相关。取值范围为[1, 1024]。
min_iris_fno_target_linear	最小光圈F值等效增益目标值，可用于控制AE分配路线，实际生效光圈大小与P-Iris镜头相关。取值范围为[1, 1024]。

注意事项

- 在线更换P-Iris镜头时，可以通过MPI设置该结构体，将新镜头的参数传递给AE算法和P-Iris驱动。

- `step_fno_table_change`为只写寄存器，通过MPI获取该参数的值始终为 `HI_FALSE`。
- `total_step`表示P-Iris步进电机的总步数，可用于标定光圈的起始位置。
- `step_count`表示P-Iris步进电机的可用步数，一般小于`total_step`，因为靠近光圈关闭端的位置在将光圈孔径转换为F值时误差较大，光圈调节过程中容易出现振荡，所以通常不会利用光圈关闭端附近的那些位置。当前最多支持1024步的PIris镜头，若P-Iris镜头的精度很高，可调节步数超过1024，建议舍去光圈关闭端附近的那些位置。
- P-Iris步进电机位置与光圈F值映射表`step_fno_table`一般根据镜头原厂提供的步进电机位置与光圈孔径对应关系制作。由于P-Iris的控制通过AE分配路线进行，这就要求光圈F值与曝光时间和增益有较好的线性关系，因此映射表需要较高的精度。这里用1024表示F1.0，512表示F1.4，以此类推，1表示F32.0。制作映射表时，可以根据最大光圈孔径对应的实际F值来规定映射表的最大值，也可以不用管光圈达到最大时的实际F值是多少，统一将最大光圈孔径映射为1024。同一款镜头处于不同焦距时，`step_fno_table`可能需要修改。
- 以福光NV03105P这款P-Iris镜头为例说明与镜头相关的参数该如何设置。表1中步进电机位置与孔径面积对应关系是镜头原厂提供的。该款镜头步进电机总步数为93，步进电机位置为0时，光圈孔径面积最大，标称最大相对孔径为F1.4，据此规定映射表F值最大值为512。规定好最大值后，就可以计算出其他孔径面积对应的F值，因为孔径面积与F值成线性关系。比如说步进电机位置1对应的孔径面积为48.835，得到F值为 $(48.835/49.366)*512=506$ ，以此类推，可以得到所有有效步进电机位置对应的F值。由下表可以看到，当镜头靠近关闭端时，孔径面积很小，与最大孔径面积相差上万倍，映射表F值精度已不足以表现出孔径面积的变化，得到的F值都为0。实际上，哪怕提高映射表精度，由于镜头关闭端附近标称孔径面积与实际孔径面积一般相差较大，得到的F值也是不正确的，建议不要使用，否则在光圈调节过程中容易出现振荡。对于该款镜头，建议制作映射表时只取前62步，因此得到镜头相关的参数如下表所示。
- 步进电机位置与光圈F值映射表`step_fno_table`的前`step_count`个元素才是有效的，会被AE算法计算时用到，需要保证数组座标小于`step_count`时，值单调递增，当数组座标为 $(step\_count-1)$ 时，达到光圈最大F值。通过MPI设置`step_fno_table`时，只有前`step_count`个元素才会被写入寄存器保存；通过MPI获取`step_fno_table`时，只有前`step_count`个元素才会读出有效数值，其他值为0。
- 光圈类型为P-Iris时，昇腾AE算法计算最大/最小曝光量会参考`max_iris_fno_target/min_iris_fno_target`(若`fno_ex_valid`为true，则参考`max_iris_fno_target_linear/min_iris_fno_target_linear`)的值，因此这几个值要与`step_fno_table`的值相匹配，即最大/最小值都在`step_fno_table`有效数据范围内。
- 在手动配置DCF光圈相关信息时，需要关闭AI。如果需要配置的最大及实际光圈值为`hi_isp_iris_f_no`中定义的10个标准值，配置`max_iris_fno_target`为最大光圈值，配置`min_iris_fno_target`为实际光圈值即可。如果光圈不为10个标准值，则需要配置`fno_ex_valid`为`HI_TRUE`，以F32.0等效增益为1倍为基数计算实际光圈及最大光圈等效增益，并配置`max_iris_fno_target_linear`为最大光圈等效增益，配置`min_iris_fno_target_linear`为实际光圈等效增益。如果AI使能，则根据实际光圈生效的大小计算光圈值，并更新DCF信息。光圈最小可支持到F32.0。

表 3-3 P-Iris 步进电机位置与 F 值映射表，以福光 NV03105P 为例

步进电机位置	孔径面积	映射表 F 值	步进电机位置	孔径面积	映射表 F 值	步进电机位置	孔径面积	映射表 F 值
0	49.366	512	35	20.07	208	70	0.136	1

步进电机位置	孔径面积	映射表 F 值	步进电机位置	孔径面积	映射表 F 值	步进电机位置	孔径面积	映射表 F 值
1	48.835	506	36	19.241	200	71	0.095	1
2	48.234	500	37	18.42	191	72	0.067	1
3	47.571	493	38	17.609	183	73	0.045	0
4	46.856	486	39	16.808	174	74	0.028	0
5	46.11	478	40	16.017	166	75	0.016	0
6	45.324	470	41	15.237	158	76	0.008	0
7	44.511	462	42	14.469	150	77	0.004	0
8	43.674	453	43	13.788	143	78	0.003	0
9	42.822	444	44	12.972	135	79	0.003	0
10	41.963	435	45	12.254	127	80	0.002	0
11	41.099	426	46	11.541	120	81	0.002	0
12	40.231	417	47	10.843	112	82	0.001	0
13	39.357	408	48	10.162	105	83	0.001	0
14	38.478	399	49	9.497	98	84	0.001	0
15	37.608	390	50	8.851	92	85	close	0
16	36.721	381	51	8.222	85	86	close	0
17	35.832	372	52	7.611	79	87	close	0
18	34.94	362	53	7.018	73	88	close	0
19	34.047	353	54	6.443	67	89	close	0
20	33.153	344	55	5.893	61	90	close	0
21	32.259	335	56	5.354	56	91	close	0
22	31.365	325	57	4.832	50	92	close	0
23	30.473	316	58	4.329	45	93	M-stop	0
24	29.582	307	59	3.843	40	-	-	-
25	28.706	298	60	3.376	35	-	-	-
26	27.82	289	61	2.926	30	-	-	-
27	26.937	279	62	2.494	26	-	-	-
28	26.059	270	63	2.08	22	-	-	-

步进电机位置	孔径面积	映射表 F 值	步进电机位置	孔径面积	映射表 F 值	步进电机位置	孔径面积	映射表 F 值
29	25.184	261	64	1.684	17	-	-	-
30	24.315	252	65	1.305	14	-	-	-
31	23.451	243	66	0.949	10	-	-	-
32	22.593	234	67	0.607	6	-	-	-
33	21.741	225	68	0.374	4	-	-	-
34	20.896	217	69	0.225	2	-	-	-

表 3-4 P-Iris 镜头相关参数，以福光 NV03105P 为例

参数名	参数值	备注
zero_is_max	HI_TRUE	步进电机位置0对应光圈最大孔径，因此该值为 HI_TRUE
total_step	93	步进电机总步数为93
step_count	62	步进电机可用步数为62
step_fno_table	{30,35,40,45,50,56,61,67,73,79,85,92,98,105,112,120,127,135,143,150,158,166,174,183,191,200,208,217,225,234,243,252,261,270,279,289,298,307,316,325,335,344,353,362,372,381,390,399,408,417,426,435,444,453,462,470,478,486,493,500,506,512}	根据表3-3，取步进电机前62步对应的F值，按照从小到大的排列顺序，制作映射表。AE算法计算时，只会用到 step_fno_table数组中的前step_count个元素。当数组座标小于step_count时，值必须保证是单调递增的。当数组座标为 (step_count-1)时，达到光圈最大F值512。
max_iris_fno_target	HI_ISP_IRIS_F_NO_1_4	映射表最大值512对应F1.4
min_iris_fno_target	HI_ISP_IRIS_F_NO_5_6	映射表最小值30接近对应F5.6
fno_ex_valid	HI_FALSE	默认不采用高精度的光圈F值等效增益
max_iris_fno_target_linear	512	取step_fno_table有效数据范围内的最大值512
min_iris_fno_target_linear	30	取step_fno_table有效数据范围内的最小值30

3.4.3.26 hi_isp_iris_attr

说明

定义ISP光圈属性。

定义

```
typedef struct {  
    hi_bool enable;  
    hi_op_mode op_type;  
    hi_isp_iris_type iris_type;  
    hi_isp_iris_status iris_status;  
    hi_isp_mi_attr mi_attr;  
} hi_isp_iris_attr;
```

成员

成员名称	描述
enable	自动光圈使能。
op_type	自动光圈或手动光圈模式选择。
iris_type	光圈类型，DC-Iris或P-Iris。
iris_status	光圈状态。
mi_attr	手动光圈属性设置结构体。

注意事项

- 进行AI算法测试前，建议确认AI电路特性是否符合昇腾IPC要求。
- 针对DC-Iris镜头，AI算法会根据画面亮度，调节PWM占空比对光圈进行控制。当曝光时间和增益达到最小目标值之后，会进入光圈控制区域。当光圈控制能满足目标亮度的要求时，AE直接返回，保持曝光时间和增益不变。当画面亮度稳定且PWM 占空比维持在打开值一段时间后，AI算法会认为光圈已经打开至最大，退出光圈控制区，将控制权交还给AE。处于光圈控制区时，更改AE算法参数，如最大/最小曝光时间、最大/最小增益和抗闪等需要即时生效的参数，AE会即时响应，根据新设定的参数和环境亮度，AI算法重新决定是否要进入光圈控制区。由于进入光圈控制区域和退出光圈控制区域需要短暂时间，针对手动光圈镜头建议关闭AI功能，否则AE的调节速度会受到一点影响。针对DC-Iris镜头建议一直打开AI功能，随意开关AI容易导致光圈控制出现异常。针对某些长焦的DC-Iris镜头，默认参数可能会导致光圈打开/关闭速度过快，此时可以调节相关参数来解决，详见hi_isp_dciris_attr部分描述。
- 针对P-Iris镜头，光圈控制通过AE分配路线进行。P-Iris对接重点在于正确设置镜头相关参数和合理设置AE分配路线，详见 hi_isp_piris_attr和 hi_mpi_isp_set_ae_route_attr部分描述，才能保证P-Iris正常工作。由于不同P-Iris的驱动方式可能会有差别，用户可以自行修改P-Iris驱动以适配不同镜头。
- 关闭AI功能，对于DC-Iris镜头，光圈会打开到最大；对于P-Iris镜头，光圈会打开到最大光圈目标值对应步进电机位置，但此时曝光分配仍会参考AE路线，可能导致画面亮度异常，因此对接P-Iris镜头时，若不想使能AI，为保证曝光正常，需要把光圈类型切换至HI_ISP_IRIS_DC_TYPE。

- 由DC-Iris切换至P-Iris，需提前设置好P-Iris镜头相关参数，并且将光圈状态设置为HI_ISP_IRIS_KEEP，再进行切换。
 - 利用昇腾Demo板或Ref板进行DC-Iris测试时，load ko时要配置以下2个寄存器，用于配置管脚复用和时钟：himm 0x102F0078 0x12F5；himm0x11014590 0x0010。
 - 利用昇腾Demo板或Ref板进行P-Iris测试时，load ko时要配置以下1个寄存器，用于配置管脚复用和时钟：himm 0x102F0108 0x1101。
- 注意：采用昇腾AE库时，AE_init会打开一次默认的PWM3信号，若客户需要采用PWM3做其他处理，需要指定AI对应的 PWM Number为其他值。

3.4.3.27 hi_isp_exp_conv_param

说明

AE算法中定义不同帧率下等曝光量转换相关曝光参数属性。

定义

```
typedef struct {  
    hi_u32 tar_fps;  
    hi_u32 tar_isp_dgain;  
    hi_isp_ae_param_reg time_reg[10];  
    hi_isp_ae_param_reg again_reg[10];  
    hi_isp_ae_param_reg dgain_reg[10];  
} hi_isp_exp_conv_param;
```

成员

成员名称	描述
tar_fps	输入参数，等曝光量转换目标帧率 * 100。 取值范围：[0x0, 0xFFFFFFFF]
tar_isp_dgain	等曝光量转换后的ISP数字增益，精度10bit。
time_reg [10]	等曝光量转换后的sensor 曝光时间的寄存器值和寄存器地址。其成员变量 reg_addr为sensor寄存器的地址，reg_value为寄存器的值。最大可转换输出 10个寄存器值和地址。
again_reg [10]	等曝光量转换后的sensor 模拟增益的寄存器值和寄存器地址。其成员变量 reg_addr为sensor寄存器的地址，reg_value为寄存器的值。最大可转换输出 10个寄存器值和地址。
dgain_reg [10]	等曝光量转换后的sensor 数字增益的寄存器值和寄存器地址。其成员变量 reg_addr为sensor寄存器的地址，reg_value为寄存器的值。最大可转换输出 10个寄存器值和地址。

3.4.3.28 hi_isp_ae_convert_param

说明

对接驱动时，定义不同帧率下等曝光量转换相关曝光参数属性。

定义

```
typedef struct {
    hi_u32 tar_fps;
    hi_u32 exp_time;
    hi_u32 exp_again;
    hi_u32 exp_dgain;
    hi_u32 exp_isp_dgain;
    hi_isp_ae_param_reg time_reg[10];
    hi_isp_ae_param_reg again_reg[10];
    hi_isp_ae_param_reg dgain_reg[10];
} hi_isp_ae_convert_param;
```

成员

成员名称	描述
tar_fps	输入参数，等曝光量转换目标帧率 * 100。 取值范围：[0x0, 0xFFFFFFFF]
exp_time	等曝光量转换后的曝光时间，单位为微秒（us）。
exp_again	等曝光量转换后的again，精度10bit。
exp_dgain	等曝光量转换后的dgain，精度10bit。
exp_isp_dgain	等曝光量转换后的ISP数字增益，精度10bit。
time_reg [10]	等曝光量转换后的sensor 曝光时间的寄存器值和寄存器地址。其成员变量 reg_addr为sensor寄存器的地址，reg_value为寄存器的值。最大可转换输出 10个寄存器值和地址。
again_reg [10]	等曝光量转换后的sensor 模拟增益的寄存器值和寄存器地址。其成员变量 reg_addr为sensor寄存器的地址，reg_value为寄存器的值。最大可转换输出 10个寄存器值和地址。
dgain_reg [10]	等曝光量转换后的sensor 数字增益的寄存器值和寄存器地址。其成员变量 reg_addr为sensor寄存器的地址，reg_value为寄存器的值。最大可转换输出 10个寄存器值和地址。

3.4.3.29 hi_isp_ae_param_reg

说明

定义寄存器状态。

定义

```
typedef struct {  
    hi_u32 reg_addr;  
    hi_u32 reg_value;  
} hi_isp_ae_param_reg;
```

成员

成员名称	描述
reg_addr	寄存器的地址。
reg_value	寄存器的值。

3.5 AWB

3.5.1 概述

色温随可见光的光谱成分变化而变化，在低色温光源下，白色物体偏红，在高色温光源下，白色物体偏蓝，人眼可根据大脑的记忆判断，识别物体的真实颜色。AWB（Automatic White Balance）算法的功能是降低外界光源对物体真实颜色的影响，使得我们采集的颜色信息转变为在理想日光光源下的无偏色信息。

3.5.2 AWB 控制模块

3.5.2.1 hi_mpi_isp_set_wb_attr

函数功能

设置白平衡属性。

函数原型

```
hi_s32 hi_mpi_isp_set_wb_attr(hi_vi_pipe vi_pipe, const hi_isp_wb_attr  
*wb_attr)
```

约束说明

- 白平衡控制类型为自动时，AWB算法自动调节白平衡系数。
- 白平衡控制类型为手动时，AWB算法失效，需自行设定Rgain、Ggain、Bgain。
环境色温、照度会影响白点的分布范围，昇腾AWB算法在运行过程中，会根据环境参数自动刷新AWB Bayer域统计信息的白点参数。用户希望在昇腾AWB算法运行时修改AWB统计参数，需要调用hi_mpi_isp_set_wb_attr接口，关闭统计参数自动刷新功能。用户关闭统计参数自动刷新功能，到ISP收到配置并响应，有2帧的延时。因此，用户设置后需要等待2帧时间再修改AWB统计参数。

参数说明

参数名称	输入/输出	描述
vi_pipe	输入	Vi PIPE号。 取值范围：[0, 12)。
wb_attr	输入	白平衡的参数属性

返回值说明

- 0：成功
- 非0：失败，参见[ISP返回码](#)。

3.5.2.2 hi_mpi_isp_get_wb_attr

函数功能

获取白平衡属性。

函数原型

[hi_s32](#) hi_mpi_isp_get_wb_attr([hi_vi_pipe](#) vi_pipe, [hi_isp_wb_attr](#) *wb_attr)

参数说明

参数名称	输入/输出	描述
vi_pipe	输入	Vi PIPE号。 取值范围：[0, 12)。
wb_attr	输出	白平衡的参数属性。

返回值说明

- 0：成功
- 非0：失败，参见[ISP返回码](#)。

3.5.2.3 hi_mpi_isp_set_awb_attr_ex

函数功能

设置自动白平衡扩展属性。

函数原型

[hi_s32](#) hi_mpi_isp_set_awb_attr_ex([hi_vi_pipe](#) vi_pipe, [hi_isp_awb_attr_ex](#) *awb_attr_ex)

约束说明

当hi_mpi_isp_set_wb_attr接口成员wb_attr. auto_attr. alg_type为HI_ISP_AWB_ALG_ADVANCE时，此接口才有效。

参数说明

参数名称	输入/输出	描述
vi_pipe	输入	Vi PIPE号。 取值范围：[0, 12)。
awb_attr_ex	输入	自动白平衡扩展属性。

返回值说明

- 0：成功
- 非0：失败，参见[ISP返回码](#)。

3.5.2.4 hi_mpi_isp_get_awb_attr_ex

函数功能

获取自动白平衡扩展属性。

函数原型

```
hi_s32 hi_mpi_isp_get_awb_attr_ex(hi_vi_pipe vi_pipe, hi_isp_awb_attr_ex *awb_attr_ex)
```

参数说明

参数名称	输入/输出	描述
vi_pipe	输入	Vi PIPE号。 取值范围：[0, 12)。
awb_attr_ex	输出	自动白平衡扩展属性。

返回值说明

- 0：成功
- 非0：失败，参见[ISP返回码](#)。

3.5.2.5 hi_mpi_isp_query_wb_info

函数功能

获取当前白平衡增益系数，检测色温，饱和度值，颜色校正矩阵系数等信息。

函数原型

hi_s32 hi_mpi_isp_query_wb_info(**hi_vi_pipe** vi_pipe, **hi_isp_wb_info** *wb_info)

参数说明

参数名称	输入/输出	描述
vi_pipe	输入	Vi PIPE号。 取值范围：[0, 12)。
wb_info	输出	颜色相关状态参数。

返回值说明

- 0：成功
- 非0：失败，参见[ISP返回码](#)。

3.5.2.6 hi_mpi_isp_cal_gain_by_temp

函数功能

计算特定色温下的白平衡增益系数。

函数原型

hi_s32 hi_mpi_isp_cal_gain_by_temp(**hi_vi_pipe** vi_pipe, const **hi_isp_wb_attr** *wb_attr, **hi_u16** color_temp, **hi_s16** shift, **hi_u16** *awb_gain)

约束说明

- 白平衡标定参数决定预设色温增益计算的准确性，因此，在调用该接口前，必须完成标定，且标定参数配置到 ISP。用户先通过 hi_mpi_isp_set_wb_attr 接口设置标定结果，再调用 hi_mpi_isp_cal_gain_by_temp 接口计算预设色温的增益，过程中不能再修改标定结果。
- hi_mpi_isp_cal_gain_by_temp 功能基于 AWB 标定参数实现，因此，该接口仅对 hi_isp_awb_attr 结构体的标定参数 static_wb，curve_para 做有效范围检查，不对其他参数做约束。
- shift 参数决定光源点与 Planckian 曲线的位置关系，shift 为负数时，光源点位于 Planckian 曲线左侧，计算的白平衡增益在预设色温下会稍微偏红；shift 为正数时，光源点位于 Planckian 曲线右侧，计算的白平衡增益在预设色温下会稍微偏绿。

参数说明

参数名称	输入/输出	描述
vi_pipe	输入	vi_pipe 号。 取值范围：[0, 12)。

参数名称	输入/输出	描述
wb_attr	输入	白平衡的参数属性。需要用到 hi_isp_awb_attr 中的 static_wb，curve_para 两组标定参数。
color_temp	输入	色温值，单位为 Kelvin。 取值范围：[1500, 15000]
shift	输入	白点与 Planckian 曲线的位置和距离。取值范围：[-64, 64]
awb_gain	输出	预设色温下的 R, Gr, Gb, B 四个通道增益。

返回值说明

- 0：成功
- 非0：失败，参见[ISP返回码](#)。

3.5.3 数据类型

3.5.3.1 hi_isp_awb_attr

说明

定义ISP自动白平衡属性。

定义

```
typedef struct {
    hi_bool enable;
    hi_u16 ref_color_temp;
    hi_u16 static_wb[HI_ISP_BAYER_CHN_NUM];
    hi_s32 curve_para[HI_ISP_AWB_CURVE_PARA_NUM];
    hi_isp_awb_alg_type alg_type;
    hi_u8 rg_strength;
    hi_u8 bg_strength;
    hi_u16 speed;
    hi_u16 zone_sel;
    hi_u16 high_color_temp;
    hi_u16 low_color_temp;
    hi_isp_awb_ct_limit_attr ct_limit;
    hi_bool shift_limit_en;
    hi_u8 shift_limit;
    hi_bool gain_norm_en;
    hi_bool natural_cast_en;
    hi_isp_awb_cbcr_track_attr cb_cr_track;
    hi_isp_awb_lum_histogram_attr luma_hist;
    hi_bool awb_zone_wt_en;
    hi_u8 zone_wt[HI_ISP_AWB_ZONE_NUM];
} hi_isp_awb_attr;
```

成员

成员名称	描述
enable	自动白平衡使能。 取值范围： - HI_FALSE: 关闭 - HI_TRUE: 开启
ref_color_temp	校正静态白平衡系数的光源色温值。 推荐选用5000K附近光源进行静态白平衡校正。 取值范围：[0x0, 0xFFFF]。
static_wb	静态白平衡系数，由AWB标定工具给出。 取值范围：[0x0, 0xFFFF]。 #define HI_ISP_BAYER_CHN_NUM 4
curve_para	校准曲线的系数。取值范围：curve_para [3]!=0; curve_para [4]=128。 #define HI_ISP_AWB_CURVE_PARA_NUM 6
alg_type	自动白平衡算法类型选择，支持 HI_ISP_AWB_ALG_LOW COST和 HI_ISP_AWB_ALG_ADVANCE可选。 HI_ISP_AWB_ALG_LOW COST CPU占用较少，对光源的 适应性较好。HI_ISP_AWB_ALG_ADVANCE提升了AWB 精度。 取值范围：HI_ISP_AWB_ALG_LOW COST、 HI_ISP_AWB_ALG_ADVANCE。
rg_strength	自动白平衡 R 通道校准强度。 取值范围：[0x0, 0xFF]。
bg_strength	自动白平衡 B 通道校准强度。 取值范围：[0x0, 0xFF]。
speed	自动白平衡算法收敛速度,值越大，AWB收敛越快。 取值范围：[0x0, 0xFFFF]。
zone_sel	自动白平衡算法全局或分区域计算的选择，取值范围： [0, 255]。
high_color_temp	自动白平衡算法的色温上限，色温上限越大，蓝色物体 对AWB的干扰越大。 推荐范围：[8500, 10000]。
low_color_temp	自动白平衡算法的色温下限，色温下限越小，橙色、红 色物体对AWB的干扰越大。取值范围：[0x0, high_color_temp)。
ct_limit	白平衡算法环境色温超过预设色温范围后，增益计算模 式及手动增益值设定。

成员名称	描述
shift_limit_en	自动白平衡算法的将超出白点范围的增益映射回白点范围内的开关。 取值范围： - HI_FALSE: 关闭 - HI_TRUE: 开启
shift_limit	室内场景或室内外检测关闭时，自动白平衡的白点范围参数。 取值范围：[0x0, 0xFF]
gain_norm_en	自动白平衡算法的增益归一化的开关。使能后，可提高低照、低色温下的信噪比。 取值范围： - HI_FALSE: 关闭 - HI_TRUE: 开启
natural_cast_en	低色温下AWB风格喜好开关。 取值范围： - HI_FALSE: 关闭 - HI_TRUE: 开启
cb_cr_track	Bayer域统计信息与ISO的联动参数。
luma_hist	白平衡的亮度直方图统计参数。
awb_zone_wt_en	白平衡的分块权重使能开关。 取值范围： - HI_FALSE: 关闭 - HI_TRUE: 开启
zone_wt[HI_ISP_AWB_ZONE_NUM]	白平衡的1024分块权重表，取值范围：[0x0, 0xFF]。 #define HI_ISP_AWB_ZONE_NUM (HI_ISP_AWB_ZONE_ORIG_ROW * HI_ISP_AWB_ZONE_ORIG_COLUMN)

注意事项

- enable为HI_TRUE时，AWB正常工作，RGB通道增益系数由AWB根据环境色温计算；enable为HI_FALSE时，AWB不工作，RGB通道增益系数固定，为校正的静态白平衡系数。
- ref_color_temp、static_wb、curve_para是用户通过校正工具得到的AWB参数，是AWB准确的前提。更换光学器件后，建议重新校正这三组参数。ref_color_temp是进行静态白平衡校正光源的真实色温，推荐在D50光源下做静态白平衡校正。curve_para[0]，curve_para[1]，curve_para[2]三个参数确定Planckian曲线，curve_para[3]，curve_para[4]，curve_para[5]三个参数确定色温曲线，curve_para[4]取值固定为128。
- 自动白平衡通过curve_para等标定参数计算环境色温。在[2500K, 10000K]范围内，色温计算准确度较高，超出以上范围后，色温计算误差较大。

- 自动白平衡R/B通道校准强度，一般情况不建议调整。
可通过调整R/B通道校准强度使AWB校准偏强或者偏弱，0x80表示标准强度。建议rg_strength、bg_strength两个参数取相同值，且小于等于0x80。在低色温场景，校准强度小于0x80时图像偏红，大于0x80时图像偏蓝；在高色温场景，校准强度小于0x80时图像偏蓝，大于0x80时图像偏红。
- speed控制AWB增益的收敛速度。speed越大，切换光源时，AWB收敛的速度越快，但帧间的波动幅度也变大；speed越小，切换光源时，AWB收敛的速度越慢，但帧间的波动幅度变小，稳定性提高。
- zone_sel参数为0或255时，采用了近似灰度世界的算法；zone_sel为其他值时，AWB算法会对所有统计块进行分类筛选，提高AWB精度。
- 自动白平衡算法的色温上限/下限，以Kelvin为单位。
AWB算法支持的最高/最低色温。若实际场景中的色温大于色温上限或小于色温下限，则AWB不能完全恢复，图像会偏向光源色，低色温下偏黄，高色温下偏蓝。
注意：色温计算在高、低色温段误差较大，设置色温上限小于1000K时，可能出现随参数值递增或递减，颜色表现不平滑的现象。
- shift_limit_en决定AWB增益是否向Planckian曲线做映射。shift_limit控制白色块的范围，取值较小时，对光源的支持范围较窄，精度较高；取值较大时，支持更多的光源，精度降低。shift_limit_en仅控制AWB增益是否向Planckian曲线做映射，无论shift_limit_en打开或关闭，shift_limit取值的改变都会影响AWB结果。校正静态白平衡系数的光源色温值，要求在 AWB算法设置的色温上下限范围内。
- gain_norm_en使能，对RGB通道增益进行限制，可以改善低色温、低照度场景的信噪比。
- natural_cast_en使能，低色温下AWB保留光源色，图像颜色更自然。注意在极高ISO场景下，RGB通道增益系数计算的中间值会被内部截断，导致natural_cast_en使能后对最终AWB环境色温计算和RGB通道增益系数结果无影响，相似情况也出现在HI_ISP_AWB_ALG_LOW COST和HI_ISP_AWB_ALG_ADVANCE切换等场景中。
- 权重表与AWB统计分块的对应顺序如下：
 - 非拼接模式下，AWB统计信息分块个数是 32x32，zone_wt[0]对应第一行第一列，zone_wt[1]对应第一行第二列，zone_wt[31]对应第一行第 32 列，zone_wt[32]对应第二行第一列，依次类推。
 - 不支持拼接模式下配置AWB统计信息权重。
- 特定应用产品，客户使能awb_zone_wt_en后，可通过设置权重表改变每个区域的权重，优化AWB表现。在shading较严重时，可加大中心区域的权重，保证中间区域AWB准确还原，降低shading对AWB的影响。在行车记录仪应用，感兴趣区域一般在画面的中心偏下区域，可加大此区域的权重，降低天空、树木等区域对AWB的干扰。

3.5.3.2 hi_isp_awb_cbcr_track_attr

说明

定义Bayer域统计信息的联动参数。

定义

```
typedef struct {  
    hi_bool enable;  
    hi_u16 cr_max[HI_ISP_AUTO_ISO_NUM];  
}
```



```
hi_u16 cr_min[HI_ISP_AUTO_ISO_NUM];  
hi_u16 cb_max[HI_ISP_AUTO_ISO_NUM];  
hi_u16 cb_min[HI_ISP_AUTO_ISO_NUM];  
} hi_isp_awb_cbc_r_track_attr;
```

成员

成员名称	描述
enable	Bayer域统计信息参数与环境照度、色温的联动使能开关。 取值范围： - HI_FALSE:关闭 - HI_TRUE:开启
cr_max[HI_ISP_AUTO_ISO_NUM]	不同照度下的cr_max取值，取值范围： [0x0,0xFFFF] #define HI_ISP_AUTO_ISO_NUM 16
cr_min[HI_ISP_AUTO_ISO_NUM]	不同照度下的cr_min取值，取值范围：[0x0, cr_max]
cb_max[HI_ISP_AUTO_ISO_NUM]	不同照度下的cb_max取值，取值范围： [0x0,0xFFFF]
cb_min[HI_ISP_AUTO_ISO_NUM]	不同照度下的cb_min取值，取值范围：[0x0, cb_max]

注意事项

- 统计参数联动使能后，AWB算法会根据环境照度、色温、用户设置的cr_max数组等实时计算cr_max、cr_min、cb_max和cb_min四个参数，并配置相应的逻辑寄存器，此时用户通过PQ Tools设置以上四个统计参数不生效。
- 统计参数联动使能，AWB算法计算的cr_max等统计参数在根据ISO插值的基础上还和环境色温相关。低色温时，白点范围较宽；中高色温时，白点范围较窄。
- 用户手动配置cr_max、cr_min、cb_max、cb_min四个参数时，需要先关闭联动功能。
- cr_max[0]、cr_min[0]、cb_max[0]、cb_min[0]的取值可在AWB参数标定时确定。用户确定了支持的色温范围后，捕获高低色温的RAW图片，计算白色区域的R/G、B/G值。cr_max[0]和cb_min[0]分别对应低色温的R/G、B/G；cr_min[0]和cb_max[0]分别对应高色温的R/G、B/G。推荐用户设置的Cr、Cb范围稍大于RAW图统计的R/G、B/G取值范围。
- AWB算法计算的cr_max等统计参数在根据ISO插值的时候，会基于cr_max[0]、cr_min[0]、cb_max[0]、cb_min[0]的值。所以不同ISO下的取值设定应以ISO100的值为基础，单调递增或递减。
- 建议在低色温(钠灯)环境标定cr_max、cb_min数组。用户统计不同照度下白色区域的R/G、B/G值，设置cr_max、cb_min数组。推荐用户设置的Cr、Cb范围稍大于RAW图片统计的R/G、B/G取值范围。
- 因低照度下环境色温多在5000K以下，cr_min、cb_max两个数组的取值可设为常数。

- 当AWB统计信息配置在DRC后的时候，hi_isp_awb_cbc_r_track_attr参数不生效，cr_min、cb_min固定为128，cr_max、cb_max固定为512。

表cr_max[16]在不同的增益情况下的设置值（仅供参考）

cr_max	Again*Dgain*ISPDgain(倍数)	设置值
cr_max [0]	1	0x150
cr_max [1]	2	0x150
cr_max [2]	4	0x150
cr_max [3]	8	0x160
cr_max [4]	16	0x170
cr_max [5]	32	0x180
cr_max [6]	64	0x190
cr_max [7]	128	0x1A0
cr_max [8]	256	0x1B0
cr_max [9]	512	0x1C0
cr_max [10]	1024	0x1D0
cr_max [11]	2048	0x1E0
cr_max [12]	4096	0x1F0
cr_max [13]	8192	0x1F0
cr_max [14]	16384	0x1F0
cr_max [15]	32768	0x1F0

表 cr_min [16]在不同的增益情况下的设置值（仅供参考）

cr_min	Again*Dgain*ISPDgain(倍数)	设置值
cr_min [0]	1	0x30
cr_min [1]	2	0x30
cr_min [2]	4	0x30
cr_min [3]	8	0x30
cr_min [4]	16	0x30
cr_min [5]	32	0x2C
cr_min [6]	64	0x2A

cr_min	Again*Dgain*ISPDgain(倍数)	设置值
cr_min [7]	128	0x28
cr_min [8]	256	0x26
cr_min [9]	512	0x24
cr_min [10]	1024	0x22
cr_min [11]	2048	0x20
cr_min [12]	4096	0x1F
cr_min [13]	8192	0x1E
cr_min [14]	16384	0x1D
cr_min [15]	32768	0x1C

表cb_max[16]在不同的增益情况下的设置值（仅供参考）

cb_max	Again*Dgain*ISPDgain(倍数)	设置值
cb_max [0]	1	0x130
cb_max [1]	2	0x130
cb_max [2]	4	0x130
cb_max [3]	8	0x130
cb_max [4]	16	0x140
cb_max [5]	32	0x150
cb_max [6]	64	0x160
cb_max [7]	128	0x170
cb_max [8]	256	0x180
cb_max [9]	512	0x190
cb_max [10]	1024	0x1A0
cb_max [11]	2048	0x1B0
cb_max [12]	4096	0x1B0
cb_max [13]	8192	0x1B0
cb_max [14]	16384	0x1B0
cb_max [15]	32768	0x1B0

表cb_min[16]在不同的增益情况下的设置值（仅供参考）

cb_min	Again*Dgain*ISPDgain(倍数)	设置值
cb_min [0]	1	0x30
cb_min [1]	2	0x30
cb_min [2]	4	0x30
cb_min [3]	8	0x30
cb_min [4]	16	0x30
cb_min [5]	32	0x2C
cb_min [6]	64	0x2A
cb_min [7]	128	0x28
cb_min [8]	256	0x26
cb_min [9]	512	0x24
cb_min [10]	1024	0x22
cb_min [11]	2048	0x20
cb_min [12]	4096	0x1F
cb_min [13]	8192	0x1E
cb_min [14]	16384	0x1D
cb_min [15]	32768	0x1C

3.5.3.3 hi_isp_awb_lum_histogram_attr

说明

定义白平衡的亮度直方图统计参数。

定义

```
typedef struct {  
    hi_bool enable;  
    hi_op_mode op_type;  
    hi_u8 hist_thresh[HI_ISP_AWB_LUM_HIST_NUM];  
    hi_u16 hist_wt[HI_ISP_AWB_LUM_HIST_NUM];  
} hi_isp_awb_lum_histogram_attr;
```

成员

成员名称	描述
enable	亮度是否影响分块的权重。 取值范围： - HI_FALSE:关闭 - HI_TRUE:开启
op_type	自动模式下，AWB算法对分块统计结果做亮度直方图统计，自动分配亮度权重。手动模式下，用户设置亮度直方图的门限和权重。 取值范围： - HI_ISP_OP_TYPE_AUTO：自动模式 - HI_ISP_OP_TYPE_MANUAL：手动模式
hist_thresh[HI_ISP_AWB_LUM_HIST_NUM]	用户设置亮度直方图的门限，仅手动模式有效。 取值范围：[0x0,0xFF] #define HI_ISP_AWB_LUM_HIST_NUM 6
hist_wt[HI_ISP_AWB_LUM_HIST_NUM]	用户设置亮度直方图的权重，自动模式和手动模式下均有效。8bit小数精度。 取值范围：[0x0,0xFFFF] #define HI_ISP_AWB_LUM_HIST_NUM 6

注意事项

- hist_thresh[0]固定为 0，hist_thresh[5]固定为0xFF。hist_thresh应严格单调递增。
- hist_wt参数用来设置hist_thresh的权重。对应用户可通过hist_wt设置实现亮区优先、暗区优先。

3.5.3.4 hi_isp_awb_alg_type

说明

定义白平衡的算法类型属性。

定义

```
typedef enum {  
    HI_ISP_AWB_ALG_LOWCOST = 0,  
    HI_ISP_AWB_ALG_ADVANCE = 1,  
    HI_ISP_AWB_ALG_BUTT  
} hi_isp_awb_alg_type;
```

成员

成员名称	描述
HI_ISP_AWB_ALG_LOWCOST	改进的灰度世界算法，根据统计信息自动计算区间权重的AWB算法。
HI_ISP_AWB_ALG_ADVANCE	对统计信息分类，再次筛选白色块的AWB算法。
HI_ISP_AWB_ALG_BUTT	预留值。

注意事项

- HI_ISP_AWB_ALG_LOWCOST CPU占用低，AWB不同照度的稳定性较好，对光源的适应性稍好。
- HI_ISP_AWB_ALG_ADVANCE CPU占用高，高低色温的极低照度情况下，保留轻微光源色；提高了大面积纯色场景（室外大面积绿色、室内大面积肤色等）AWB算法准确度。

3.5.3.5 hi_isp_awb_ct_limit_attr

说明

定义白平衡的增益范围限制属性。

定义

```
typedef struct {  
    hi_bool enable;  
    hi_op_mode op_type;  
    hi_u16 high_rg_limit;  
    hi_u16 high_bg_limit;  
    hi_u16 low_rg_limit;  
    hi_u16 low_bg_limit;  
} hi_isp_awb_ct_limit_attr;
```

成员

成员名称	描述
enable	白平衡的增益范围限制开关。 取值范围： - HI_FALSE:关闭 - HI_TRUE:开启
op_type	自动或手动设定白平衡的增益限制。 取值范围： - HI_ISP_OP_TYPE_AUTO：自动模式 - HI_ISP_OP_TYPE_MANUAL：手动模式
high_rg_limit	手动模式下用户设定高色温下的最大 R 增益，8bit小数精度，取值范围：[0x0, 0xFFF]。

成员名称	描述
high_bg_limit	手动模式下用户设定高温下的最小 B 增益，8bit小数精度，取值范围：[0x0, 0xFFFF]。
low_rg_limit	手动模式下用户设定低温下的最小 R 增益，8bit小数精度，取值范围：[0x0, high_rg_limit)。
low_bg_limit	手动模式下用户设定低温下的最大 B 增益，8bit小数精度，取值范围：(high_bg_limit, 0xFFFF]

注意事项

- hi_isp_awb_ct_limit_attr结构体定义环境色温超过用户设定的色温范围时，AWB应采取的动作。AWB检测到环境色温超过用户设置的上下限时，相关参数才会生效。
- 支持自动模式和手动模式选择。自动模式下，AWB根据用户校正的色温曲线计算色温上下限色温的AWB增益，以限制 R, B 通道增益；手动模式下，高温时，AWB调用high_rg_limit、high_bg_limit参数限制 R, B 通道增益，低温时，AWB调用 low_rg_limit、low_bg_limit参数限制 R,B 通道增益。
- 建议用户在进行AWB色温曲线校正时，确定了AWB工作的色温范围后，再确定 high_rg_limit、high_bg_limit、low_rg_limit、low_bg_limit四个参数取值。Rg 对应Planckian曲线的横坐标值，Bg对应Planckian曲线的纵坐标值。

3.5.3.6 hi_isp_mwb_attr

说明

定义ISP手动白平衡属性。

定义

```
typedef struct {  
    hi_u16 r_gain;  
    hi_u16 gr_gain;  
    hi_u16 gb_gain;  
    hi_u16 b_gain;  
} hi_isp_mwb_attr;
```

成员

成员名称	描述
r_gain	手动白平衡红色通道增益，8bit小数精度。取值范围：[0x0,0xFFFF]。
gr_gain	手动白平衡绿色通道增益，8bit小数精度。取值范围：[0x0,0xFFFF]。
gb_gain	手动白平衡蓝色通道增益，8bit小数精度。取值范围：[0x0,0xFFFF]。

成员名称	描述
b_gain	手动白平衡蓝色通道增益，8bit小数精度。 取值范围：[0x0,0xFFF]。

3.5.3.7 hi_isp_wb_attr

说明

定义白平衡属性。

定义

```
typedef struct {  
    hi_bool bypass;  
    hi_u8 awb_run_interval;  
    hi_op_mode op_type;  
    hi_isp_mwb_attr manual_attr;  
    hi_isp_awb_attr auto_attr;  
    hi_isp_awb_alg alg_type;  
} hi_isp_wb_attr;
```

成员

成员名称	描述
bypass	白平衡模块Bypass使能。 取值范围： - HI_FALSE: 关闭 - HI_TRUE: 开启
awb_run_interval	白平衡模块工作频率。取值范围：[0x1, 0xFF]。
op_type	自动白平衡和手动白平衡切换。 取值范围： - HI_ISP_OP_TYPE_AUTO: 自动模式 - HI_ISP_OP_TYPE_MANUAL: 手动模式
manual_attr	手动白平衡参数。
auto_attr	自动白平衡参数。
alg_type	GW AWB或SpecAWB(机器学习)算法类型选择。当前仅支持HI_ISP_ALG_AWB_GW。

注意事项

- bypass为HI_TRUE 时，WB的其它参数设置不生效，RGB通道增益系数固定为0x100。

- awb_run_interval默认为 2，表示每2帧运行一次AWB。客户可根据业务需求降低AWB的运行频率。

3.5.3.8 hi_isp_awb_multi_ls_type

说明

定义混合光源下的AWB策略。

定义

```
typedef enum {  
    HI_ISP_AWB_MULTI_LS_SAT = 0,  
    HI_ISP_AWB_MULTI_LS_CCM = 1,  
    HI_ISP_AWB_MULTI_LS_BUTT  
} hi_isp_awb_multi_ls_type;
```

成员

成员名称	描述
HI_ISP_AWB_MULTI_LS_SAT	混合光源下自动调整饱和度。
HI_ISP_AWB_MULTI_LS_CCM	混合光源下自动调整CCM。
HI_ISP_AWB_MULTI_LS_BUTT	预留值。

注意事项

- HI_ISP_AWB_MULTI_LS_CCM对图像饱和度损失较少，但会改变颜色的色调。
- HI_ISP_AWB_MULTI_LS_SAT方式降低图像整体饱和度，改善混合光源下的偏色程度。

3.5.3.9 hi_isp_awb_indoor_outdoor_status

说明

定义AWB室内外状态。

定义

```
typedef enum {  
    HI_ISP_AWB_INDOOR_MODE = 0,  
    HI_ISP_AWB_OUTDOOR_MODE = 1,  
    HI_ISP_AWB_INDOOR_OUTDOOR_BUTT  
} hi_isp_awb_indoor_outdoor_status;
```

成员

成员名称	描述
HI_ISP_AWB_INDOOR_MODE	室内模式。

成员名称	描述
HI_ISP_AWB_OUTDOOR_MODE	室外模式。
HI_ISP_AWB_INDOOR_OUTDOOR_BUTT	无效。

3.5.3.10 hi_isp_awb_attr_ex

说明

定义自动白平衡扩展属性。

定义

```
typedef struct {  
    hi_u8 tolerance;  
    hi_u8 zone_radius;  
    hi_u16 curve_l_limit;  
    hi_u16 curve_r_limit;  
    hi_bool extra_light_en;  
    hi_isp_awb_extra_light_source_info light_info[HI_ISP_AWB_LS_NUM];  
    hi_isp_awb_in_out_attr in_or_out;  
    hi_bool multi_light_source_en;  
    hi_isp_awb_multi_ls_type multi_ls_type;  
    hi_u16 multi_ls_scaler;  
    hi_u16 multi_ct_bin[HI_ISP_AWB_MULTI_CT_NUM];  
    hi_u16 multi_ct_wt[HI_ISP_AWB_MULTI_CT_NUM];  
    hi_bool fine_tun_en;  
    hi_u8 fine_tun_strength;  
} hi_isp_awb_attr_ex;
```

成员

成员名称	描述
tolerance	自动白平衡调整的偏差范围，检测误差在门限范围内时，AWB不动作。 取值范围：[0x0,0xFF]
zone_radius	自动白平衡统计中对像素分类时用的距离范围。该值越小，AWB精度越高，但会降低 AWB算法稳定性。 取值范围：[0x0,0xFF]
curve_l_limit	自动白平衡色温曲线的左边界限。取值范围：[0x0,0x100]
curve_r_limit	自动白平衡色温曲线的右边界限。取值范围：[0x100,0xFFF]
extra_light_en	自动白平衡计算时是否考虑色温曲线外的独立光源点。 取值范围： - HI_FALSE: 关闭 - HI_TRUE: 开启

成员名称	描述
light_info[HI_ISP_AWB_LS_NUM]	色温曲线外的独立光源点的信息, 最多可以添加 4 个。 #define HI_ISP_AWB_LS_NUM 4
in_or_out	自动白平衡对场景做室内外判断的参数。
multi_light_source_en	自动白平衡检测当前场景是否为混合光源, 根据混合光源程度调整饱和度或CCM。 取值范围: - HI_FALSE: 关闭 - HI_TRUE: 开启
multi_ls_type	混合光源调整策略选择, 支持调整饱和度或CCM。 取值范围: - AWB_MULTI_LS_SAT: 调整饱和度 - AWB_MULTI_LS_CCM: 调整CCM
multi_ls_scaler	混合光源下, 饱和度或CCM最大调整幅度。实际调整幅度还和场景混合光源程度有关。 取值范围: [0x0, 0x100]
multi_ct_bin[HI_ISP_AWB_MULTI_CT_NUM]	混合光源下的色温分段参数。 取值范围: [0x0, 0xFFFF], 要求为单调递增序列。 #define HI_ISP_AWB_MULTI_CT_NUM 8
multi_ct_wt[HI_ISP_AWB_MULTI_CT_NUM]	混合光源下的色温权重参数。 取值范围: [0x0, 0x400] #define HI_ISP_AWB_MULTI_CT_NUM 8
fine_tun_en	自动白平衡特殊色检测开关, 包括肤色检测等。 取值范围: - HI_FALSE: 关闭 - HI_TRUE: 开启
fine_tun_strength	肤色、蓝色等单色检测的强度。仅在fine_tun_en使能时有效。取值范围: [0x0, 0xFF]

图 色温曲线的参数示意

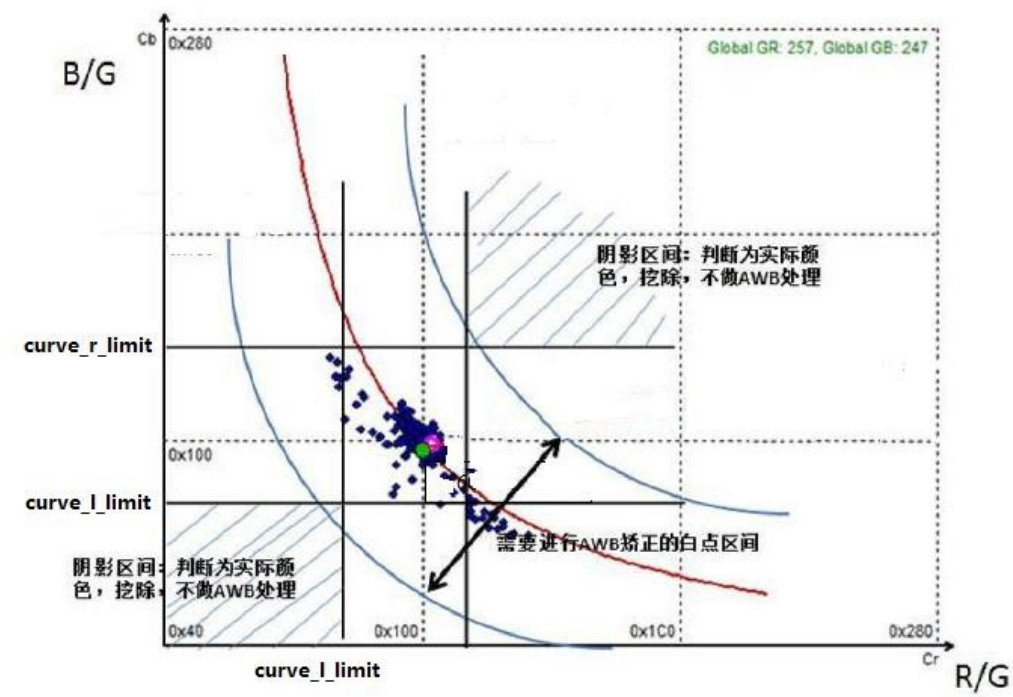
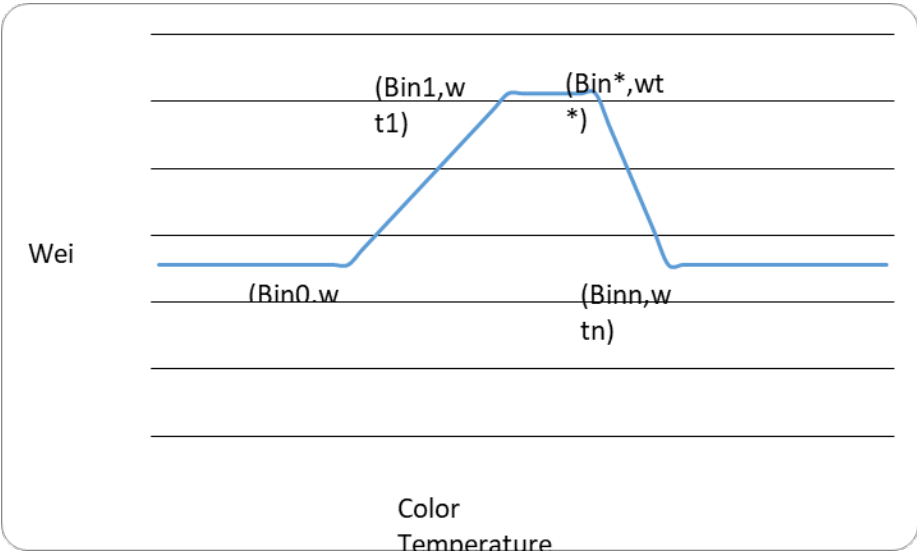


图 混合光源场景色温权重设置示例说明（n 为色温分段点个数）



注意事项

- tolerance是AWB的灵敏度参数。该值为0时，AWB系数实时更新，CPU占用较高；该值较大时，AWB在检测到环境色温变化时再更新AWB系数，CPU占用较低，但在环境色温微调时，AWB可能出现轻微的偏色。推荐室外应用时tolerance置为 0，室内应用时置为 2。
- zone_radius是选择光源环境白色块的色差半径。zone_radius越大，白色块的限制条件越宽，找到的白色块较多，稳定性稍好，精度稍低；zone_radius越小，白色块的限制条件越严格，找到的白色块较少，AWB精度更高。在混合光源的应用场景，推荐zone_radius比普通场景稍大。

- `curve_l_limit`取值 $\leq 0x100$, `curve_r_limit`取值 $\geq 0x100$ 。`curve_l_limit`和`curve_r_limit`两个参数用来排除颜色块。如图4-3所示, `curve_l_limit`排除了左侧的阴影区域, `curve_r_limit`排除了右侧的阴影区域。
- AWB算法支持在色温曲线外, 再增加独立光源点信息, 提升特殊光源下的AWB表现。
- `multi_light_source_en`使能后, AWB算法判断场景是否混合光源, 在混合光源场景, 自动降低饱和度或调整CCM, 以减弱偏色程度。最大调整幅度由`multi_ls_scaler`决定。调整CCM方式会改变色调。
- `multi_ls_type=HI_ISP_AWB_MULTI_LS_SAT`时, 调整饱和度来改善混合光源场景的偏色。`multi_ls_scaler/0x100`表示饱和度增益系数的下限。
`multi_ls_scaler=0xC0`, 表示饱和度的增益系数下限是 $0xC0/0x100 = 0.75$, 图像饱和度的波动范围是 $[0.75, 1.0]$ (1.0表示关闭`multi_light_source_en`功能的图像饱和度); `multi_ls_scaler=0x0`, 表示饱和度的增益系数下限是 $0x0/0x100 = 0$, 图像饱和度的波动范围是 $[0, 1]$, 在混合光源下图像可能变黑白。`multi_ls_scaler`取值越小, 图像饱和度的波动范围越大, 颜色稳定性下降。
- `multi_ls_type=HI_ISP_AWB_MULTI_LS_CCM`时, 调整CCM来改善混合光源场景的偏色。`multi_ls_scaler/0x100`表示CCM的调整幅度, 取值越大, 图像饱和度越低。`multi_ls_scaler=0xC0`, 表示CCM的最大调整幅度为 $0xC0$; `multi_ls_scaler=0x0`, 表示CCM的最大调整幅度为0, 等效于`multi_light_source_en=0`。
- 昇腾AWB算法在WDR模式自动关闭混合光源判断功能。
- `multi_ct_bin`、`multi_ct_wt`两组参数的合理配置, 可以改善混合光源高色温或低色温区域颜色表现。在非混合光源场景, 参数配置不生效。图4-3说明色温和权重配置的映射关系。
- 如果混合光源场景希望使能色温权重来改善AWB表现, 但不做饱和度或CCM调解, 可配置如下: `multi_light_source_en=HI_TRUE`, `multi_ls_type=HI_ISP_AWB_MULTI_LS_SAT`, `multi_ls_scaler=0x100`。
- `fine_tun_en`打开, AWB会自动检测肤色等特殊色, 改善肤色场景AWB表现, 提高AWB精度。但可能在蓝色背景+3500K光源时发生误判, 导致图像轻微偏黄。
- `fine_tun_strength`调整肤色检测的强度, 取值越大, 肤色场景AWB表现越好, 但肤色误判时的副作用越明显。推荐采用默认值 $0x80$ 。

3.5.3.11 hi_isp_awb_extra_light_source_info

说明

定义独立光源点的信息。

定义

```
typedef struct {  
    hi_u16 white_r_gain;  
    hi_u16 white_b_gain;  
    hi_u16 exp_quant;  
    hi_u8 light_status;  
    hi_u8 radius;  
} hi_isp_awb_extra_light_source_info;
```

成员

成员名称	描述
white_r_gain	增加或删除区域的中心Cr坐标，8bit小数精度。 取值范围：[0x0, 0xFF]
white_b_gain	增加或删除区域的中心Cb坐标，8bit小数精度。 取值范围：[0x0, 0xFF]
exp_quant	对外界环境亮度做判断，暂不支持，不需要配置。
light_status	光源点状态。 0：不使能； 1：增加光源点； 2：删除干扰色；
radius	增加或删除区域的半径。 取值范围：[0x0, 0xFF]

注意事项

- 独立光源点实现以下功能：用户设定CbCr坐标的某个区域，指定该区域为特殊光源下的白色表现，增加为独立光源点；或指定该区域为特定光源下的干扰色，进行删除。
- 通过light_status参数的设置实现功能选择。light_status设置为1时，在用户校正的Plankcian曲线外，增加特殊光源点，以优化该光源下颜色表现，不影响设备在其它光源下的表现；light_status设置为2时，支持用户删除特定光源的某个干扰色(肤色、绿色、蓝天等)。
- 建议删除干扰色仅用来优化特定环境的颜色表现，否则会影响其它光源的表现。在CbCr坐标，5500K光源下的肤色与3500K 光源的白色重合度很高，在5500K环境删除肤色的干扰，会导致3500K光源下偏色。
- white_r_gain、white_b_gain的取值。在该光源下捕获ColorChecker Raw数据，通过Calibration Tool Staitic WB选项校准得到的Rgain、Bgain即white_r_gain、white_b_gain坐标值。删除干扰色，可选择干扰色区域进行静态白平衡校正，来确定 white_r_gain、white_b_gain坐标值。
- radius取值由用户设置，增加光源点，建议取值在[0x8, 0x10]范围内，删除干扰色，尽量避免[white_r_gain、white_b_gain]为中心，radius为半径的圆形区域与Plankcian曲线重合。增加光源点或删除干扰色，设置radius半径太大时(覆盖场景内大多数统计块)，会导致功能失效，甚至起反作用。
- 当前支持定义最多4个光源色，4个相互独立。4个光源点需要避免交集，特别需要避免以下情况：CbCr坐标的单个点，既指定为增加的独立光源点，又指定为删除的干扰色。

3.5.3.12 hi_isp_awb_in_out_attr

说明

定义对场景做室内外判断的参数。

定义

```
typedef struct {  
    hi_bool enable;  
    hi_op_mode op_type;  
    hi_isp_awb_indoor_outdoor_status outdoor_status;  
    hi_u32 out_thresh;  
    hi_u16 low_start;  
    hi_u16 low_stop;  
    hi_u16 high_start;  
    hi_u16 high_stop;  
    hi_bool green_enhance_en;  
    hi_u8 out_shift_limit;  
} hi_isp_awb_in_out_attr;
```

成员

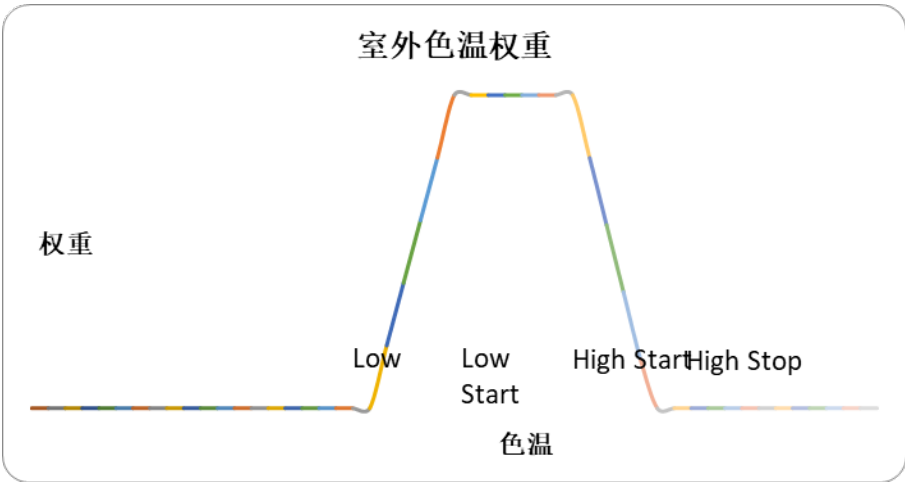
成员名称	描述
enable	对场景做室内外判断开关。 取值范围： - HI_FALSE: 关闭 - HI_TRUE: 开启
op_type	对场景做室内外判断类型(自动或手动)。 取值范围： - HI_ISP_OP_TYPE_AUTO: 自动模式 - HI_ISP_OP_TYPE_MANUAL: 手动模式
outdoor_status	手动模式下，用户输入室内外状态信息。 取值范围： - AWB_INDOOR_MODE: 室内模式 - AWB_OUTDOOR_MODE: 室外模式
out_thresh	室内外场景判定设置的阈值，以单位为us的曝光时间，小于该阈值则认为是室外。
low_start	将位于低色温范围内的像素权重调低，低色温区起始值，建议5000K。
low_stop	将位于低色温范围内的像素权重调低，低色温区终止值，建议4500K。 取值范围：(0, low_start)。
high_start	将位于高色温范围内的像素权重调低，高色温区起始值，建议6500K。 取值范围：(low_start, 0xFFFF]。
high_stop	将位于高色温范围内的像素权重调低，高色温区终止值，建议8000K。 取值范围：(high_start, 0xFFFF]。

成员名称	描述
green_enhance_en	在绿色植物场景下是否对绿色通道增强。 取值范围： - HI_FALSE: 关闭 - HI_TRUE: 开启
out_shift_limit	判断为室外场景时，AWB算法的白点范围限制shift=out_shift_limit。取值范围：[0x0, 0xFF]。

注意事项

- out_thresh需要根据sensor的感光特性进行调整。等效曝光时间=曝光时间*模拟增益*数字增益*ISP数字增益。
- 用户调用自己的AE库，且室内外判断类型为自动时，请务必将当前曝光时间和ISO传递给AWB库。
- 设置色温范围的low_start、low_stop、high_start、high_stop等4个参数，和sensor校正相关，建议用户针对sensor做微调。四个参数的取值范围要求：low_stop<low_start<high_start<high_stop。如果设置的室外色温范围[low_start, high_start]较宽，室外的大面积纯色场景颜色表现可能会稍差；如果设置的室外色温范围[low_start, high_start]较窄，在日落前等高低色温场景，AWB不能完全恢复，低色温下稍微偏黄，高色温下稍微偏蓝。
- AWB校正参数调整后，low_start、low_stop、high_start、high_stop等4个参数要再次进行调整，以达到最佳效果。
- shift取值影响AWB算法表现。普通场景shift_limit、out_shift_limit两个参数的取值大小对AWB结果影响不大。特殊光源、大面积单色场景，shift_limit、out_shift_limit两个参数的取值大小对AWB结果影响较大。
 - 两个Shift参数取值较小时，单色表现好，但特殊光源下容易偏色；
 - 两个shift参数取值较大时，AWB表现稳定性好，但大面积绿色、黄色等场景可能偏色。

图 室外色温范围参数的意义



3.5.3.13 hi_isp_wb_info

说明

定义白平衡，饱和度，颜色校正信息。

定义

```
typedef struct {
    hi_u16 r_gain;
    hi_u16 gr_gain;
    hi_u16 gb_gain;
    hi_u16 b_gain;
    hi_u16 saturation;
    hi_u16 color_temp;
    hi_u16 ccm[HI_ISP_CCM_MATRIX_SIZE];
    hi_u16 ls0_ct;
    hi_u16 ls1_ct;
    hi_u16 ls0_area;
    hi_u16 ls1_area;
    hi_u8 multi_degree;
    hi_u16 active_shift;
    hi_u32 first_stable_time;
    hi_isp_awb_indoor_outdoor_status in_out_status;
    hi_s16 bv;
} hi_isp_wb_info;
```

成员

成员名称	描述
r_gain	当前 R 通道增益值，8bit/小数精度。 取值范围：[0x0, 0xFFFF]。
gr_gain	当前 Gr 通道增益值，8bit/小数精度。 取值范围：[0x0, 0xFFFF]。
gb_gain	当前 Gb 通道增益值，8bit/小数精度。 取值范围：[0x0, 0xFFFF]。
b_gain	当前 B 通道增益值，8bit/小数精度。 取值范围：[0x0, 0xFFFF]。
saturation	当前饱和度值，有效范围为[0x0, 0xFF]。
color_temp	当前色温值。 取值范围：[0x0, 0xFFFF]。
ccm[HI_ISP_CCM_MATRIX_SIZE]	当前颜色校正矩阵值，8bit/小数精度。bit 15是符号位，0表示正数，1 表示负数，例如0x8010表示-16。 取值范围：[0x0, 0xFFFF]。 #define HI_ISP_CCM_MATRIX_NUM 7
ls0_ct	混合光源场景，主光源色温。 取值范围：[0x0, 0xFFFF]。

成员名称	描述
ls1_ct	混合光源场景，次要光源色温。 取值范围：[0x0, 0xFFFF]。
ls0_area	混合光源场景，主光源面积。取值范围为 [0x0,0xFF]。
ls1_area	混合光源场景，次要光源面积。取值范围为 [0x0, 0xFF]。
multi_degree	当前场景是混合光源的概率。取值范围为[0x0, 0xFF]。
first_stable_time	首次 AWB 收敛稳定的时间，以帧为计数单位。 取值范围为[0x0, 0xFFFFFFFF]。
active_shift	当前场景最终生效的白点范围限制shift值。取 值范围为[0x0,0xFF]。
in_out_status	室内外检测结果。取值范围： AWB_INDOOR_MODE、 AWB_OUTDOOR_MODE
bv	当前环境bv值。取值范围为[-32768, 32767]。

注意事项

- multi_degree非0时，表示场景是混合光源。该值越大，场景为混合光源的概率越大。
- ls0_ct、ls1_ct、ls0_area、ls1_area等查询结果仅在multi_degree非 0 时有效。
- 混合光源检测仅在室内场景打开，AWB判断为室外场景后，逐渐减弱混合光源检测的强度，以避免饱和度降低。
- 混合光源概率说明：主光源和次要光源的亮度、色温差等条件影响混合光源的概率计算。检测到场景内存在主光源和次要光源两种光源后，主光源和次要光源的色温差异越大、亮度差异越小，混合光源概率越大。
- ls0_area和 ls1_area之和有可能大于白平衡分区个数，原因是部分区域可能受到两种光源的影响，算法不将重叠区域严格识别，会叠加到主光源、次要光源。
- first_stable_time是AWB首次收敛稳定后所对应的帧数，非0表示AWB已稳定。ISP系统启动后，1-64帧AWB固定每帧执行一次，以加速AWB收敛。

3.6 CCM

3.6.1 概述

sensor对光谱的响应，在RGB各分量上与人眼对光谱的响应通常是有偏差的，通常通过一个色彩校正矩阵校正光谱响应的交叉效应和响应强度，使前端捕获的图片与人眼视觉在色彩上保持一致。

在ISP运行时，FW根据当前的光照强度，调整饱和度，实现CCM（Color Correction Matrix）矩阵系数的动态调整。CCM矩阵如下图所示。

图

$$\begin{pmatrix} R' \\ G' \\ B' \end{pmatrix} = \begin{pmatrix} m_{RR} & m_{RG} & m_{RB} \\ m_{GR} & m_{GG} & m_{GB} \\ m_{BR} & m_{BG} & m_{BB} \end{pmatrix} \times \begin{pmatrix} R \\ G \\ B \end{pmatrix}$$

关键概念如下：

- 色彩还原：通常通过一个色彩校正矩阵校正光谱响应的交叉效应和响应强度，使ISP处理后的图片与人眼视觉在色彩上保持一致。
- 饱和度：也称色彩的纯度。取决于该色中含色成分和消色成分(灰色)的比例。含色成分越大，饱和度越大；消色成分越大，饱和度越小

3.6.2 hi_mpi_isp_set_saturation_attr

函数功能

设置颜色饱和度属性。

函数原型

```
hi_s32 hi_mpi_isp_set_saturation_attr(hi_vi_pipe vi_pipe, const hi_isp_saturation_attr *sat_attr)
```

参数说明

参数名称	输入/输出	描述
vi_pipe	输入	Vi PIPE号 取值范围：[0, 12)。
sat_attr	输入	颜色饱和度属性。

返回值说明

- 0：成功
- 非0：失败，参见[ISP返回码](#)。

3.6.3 hi_mpi_isp_get_saturation_attr

函数功能

获取颜色饱和度属性。

函数原型

```
hi_s32 hi_mpi_isp_get_saturation_attr(hi_vi_pipe vi_pipe, hi_isp_saturation_attr *sat_attr)
```

参数说明

参数名称	输入/输出	描述
vi_pipe	输入	Vi PIPE号。 取值范围：[0, 12)。
sat_attr	输出	颜色饱和度属性。

返回值说明

- 0：成功
- 非0：失败，参见ISP返回码。

3.6.4 hi_mpi_isp_set_ccm_attr

函数功能

设置颜色矩阵。

函数原型

```
hi_s32 hi_mpi_isp_set_ccm_attr(hi_vi_pipe vi_pipe, const hi_isp_color_matrix_attr *ccm_attr)
```

约束说明

- 颜色校正矩阵的数据格式，应与校正工具提供的保持一致。
- 可根据当前色温，设置不同的CCM，从而在高低色温下都达到较好的颜色还原。
- 该MPI支持最大七个最小三个不同色温下的色彩还原矩阵，要按照色温递减的方式配置CCM，即0-6组CCM矩阵由高色温到低色温。典型的三组CCM为D50，TL84，A三个光源下的CCM。典型的五组CCM为10K，D65，D50，TL84，A五个光源下的CCM。前一组的色温值与后一组的色温值需符合如下规则： $T_{pre} * (100 - 6.25) > T_{post} * (100 + 6.25)$ 。
- 手动模式下，ISP系统生效的CCM即用户手动配置的CCM，有利于CCM精调，因此建议在手动模式下校正不同色温光源下的 CCM矩阵，得到多组CCM矩阵后再通过MPI写入，验证多个光源下的CCM组和效果。手动CCM模式下，饱和度调整有效。在特定光源下做CCM精调时，需要先确认饱和度是否期望值。

参数说明

参数名称	输入/输出	描述
vi_pipe	输入	Vi PIPE号。 取值范围：[0, 12)。
ccm_attr	输入	颜色矩阵。

返回值说明

- 0：成功
- 非0：失败，参见ISP返回码。

3.6.5 hi_mpi_isp_get_ccm_attr

函数功能

获取颜色矩阵。

函数原型

hi_s32 hi_mpi_isp_get_ccm_attr(**hi_vi_pipe** vi_pipe, **hi_isp_color_matrix_attr** *ccm_attr)

参数说明

参数名称	输入/输出	描述
vi_pipe	输入	Vi PIPE号。 取值范围：[0, 12)。
ccm_attr	输出	颜色矩阵。

返回值说明

- 0：成功
- 非0：失败，参见ISP返回码。

3.6.6 数据类型

3.6.6.1 hi_isp_saturation_attr

说明

定义ISP颜色饱和度属性。

定义

```
typedef struct {  
    hi_op_mode op_type;  
    hi_isp_saturation_manual manual_attr;  
    hi_isp_saturation_auto auto_attr;  
} hi_isp_saturation_attr;
```

成员

成员名称	描述
op_type	设置饱和度类型(手动或自动)。取值范围： HI_OP_MODE_AUTO、HI_OP_MODE_MANUAL
manual_attr	手动饱和度参数。
auto_attr	自动饱和度参数。

注意事项

设置饱和度功能分为自动和手动：

- op_type为HI_OP_MODE_AUTO，使用自动饱和度调节功能。
此时饱和度值会根据系统增益自动调节，饱和度与系统增益的关系请参见成员变量sat [16]的描述。
在自动饱和度模式下，系统最终的饱和度由系统增益对应的sat [16]变量，AWB控制的混合光源下自动调节饱和度 multi_ls_scaler，极低照度下CCM Bypass功能 iso_act_en，高、低色温下CCM Bypass功能temp_act_en，几个因素共同决定。
- op_type为HI_OP_MODE_MANUAL，使用手动饱和度调节功能。此时变量sat[16]无效。在手动饱和度模式下，系统最终的饱和度由用户完全控制。

3.6.6.2 hi_isp_saturation_manual

说明

定义手动饱和度属性。

定义

```
typedef struct {  
    hi_u8 saturation;  
} hi_isp_saturation_manual;
```

成员

成员名称	描述
saturation	手动饱和度值。 取值范围：[0x0, 0xFF]

3.6.6.3 hi_isp_saturation_auto

说明

定义自动饱和度属性。

定义

```
typedef struct {  
    hi_u8 sat[HI_ISP_AUTO_ISO_NUM];  
} hi_isp_saturation_auto;
```

成员

成员名称	描述
sat[HI_ISP_AUTO_ISO_NUM]	自动饱和度值。 推荐配置为递减序列。取值范围：[0x0, 0xFF] #define HI_ISP_AUTO_ISO_NUM 16

表sat[16]在不同的增益情况下的设置值，以mn34220 为例

sat	Again*Dgain*ISPDgain(倍数)	推荐设置值
sat [0]	1	0x74
sat [1]	2	0x74
sat [2]	4	0x70
sat [3]	8	0x6b
sat [4]	16	0x64
sat [5]	32	0x5b
sat [6]	64	0x52
sat [7]	128	0x44
sat [8]	256	0x3b
sat [9]	512	0x38
sat [10]	1024	0x38
sat [11]	2048	0x38
sat [12]	4096	0x38
sat [13]	8192	0x38
sat [14]	16384	0x38

sat	Again*Dgain*ISPDgain(倍数)	推荐设置值
sat [15]	32768	0x38

3.6.6.4 hi_isp_color_matrix_attr

说明

定义ISP颜色矩阵属性。

定义

```
typedef struct {  
    hi_op_mode op_type;  
    hi_isp_color_matrix_manual manual_attr;  
    hi_isp_color_matrix_auto auto_attr;  
} hi_isp_color_matrix_attr;
```

成员

成员名称	描述
op_type	颜色校正矩阵类型(手动或自动)。
manual_attr	手动颜色校正矩阵。
auto_attr	自动颜色校正矩阵。

3.6.6.5 hi_isp_color_matrix_auto

说明

定义自动颜色校正矩阵属性。

定义

```
typedef struct {  
    hi_bool iso_act_en;  
    hi_bool temp_act_en;  
    hi_u16 ccm_tab_num;  
    hi_isp_color_matrix_param ccm_tab[HI_ISP_CCM_MATRIX_NUM];  
} hi_isp_color_matrix_auto;
```

成员

成员名称	描述
iso_act_en	是否使能低照度下CCM Bypass功能。 取值范围：[0,1]。

成员名称	描述
temp_act_en	是否使能高、低色温下CCM Bypass功能。 取值范围：[0,1]。
ccm_tab_num	当前配置的CCM的组数。 取值范围：[3,7]。
ccm_tab[HI_ISP_CCM_MATRIX_NUM]	不同色温下的颜色校正矩阵和对应的色温值。 #define HI_ISP_CCM_MATRIX_NUM 7

注意事项

- 当hi_isp_saturation_attr中的op_type为HI_OP_MODE_AUTO，使用自动饱和度调节功能时，iso_act_en和temp_act_en功能生效。当op_type为HI_OP_MODE_MANUAL，使用手动饱和度调节功能时，iso_act_en和temp_act_en功能不生效。
- iso_act_en功能指的是极低照度下（ISO 大于 12800）逐渐Bypass CCM，将由饱和度数组sat[16]控制得到的CCM矩阵逐渐调向接近单位阵，这时CCM的实际饱和度会和饱和度数组中设定的不一致。用户关闭该功能后，极低照度下，饱和度数组仍生效。
- temp_act_en功能指的是在高色温(大于 8000K)和低色温场景(小于 2500K)，AWB可能未完全恢复，此时采用高饱和度CCM 矩阵，会加剧AWB的偏色程度。比如钠灯场景，AWB校正后白色区域蓝色稍有不足，采用高饱和度CCM矩阵，白色区域的蓝色分量可能为0，图像整体偏黄。使能temp_act_en，在高低色温下逐渐BypassCCM，将由饱和度数组sat[16]控制得到的 CCM矩阵逐渐调向接近单位阵，优化类似场景颜色表现。
- 颜色校正矩阵的数据格式，应与校正工具提供的保持一致。

3.6.6.6 hi_isp_color_matrix_param

说明

定义不同色温下自动颜色校正矩阵系数。

定义

```
typedef struct {  
    hi_u16 color_temp;  
    hi_u16 ccm[HI_ISP_CCM_MATRIX_SIZE];  
} hi_isp_color_matrix_param;
```

成员

成员名称	描述
color_temp	当前配置的CCM对应的色温。 取值范围：[500, 30000]。

成员名称	描述
ccm[HI_ISP_CCM_MATRIX_SIZE]	不同色温下的颜色校正矩阵，8bit小数精度。bit15是符号位，0表示正数，1表示负数，例如0x8010表示-16。bit12,13,14是无效的，即正数从0x0000到0x0FFF，负数从0x8000到0x8FFF，是CCM的合理参数。 取值范围：[0x0,0xFFFF]。 #define HI_ISP_CCM_MATRIX_SIZE 9

注意事项

- 颜色校正矩阵的数据格式，应与校正工具提供的保持一致。
- 当饱和度为自动的时候，实际生效的CCM参数和AWB算法反馈的当前色温值，自动饱和度参数有关。当饱和度为手动的时候，实际生效的CCM参数和AWB算法反馈的当前色温值有关。

3.6.6.7 hi_isp_color_matrix_manual

说明

定义手动颜色校正矩阵属性。

定义

```
typedef struct {  
    hi_bool sat_en;  
    hi_u16 ccm[HI_ISP_CCM_MATRIX_SIZE];  
} hi_isp_color_matrix_manual;
```

成员

成员名称	描述
sat_en	手动CCM模式下，饱和度是否生效。 取值范围：[0,1]。
ccm[HI_ISP_CCM_MATRIX_SIZE]	手动颜色校正矩阵，8bit小数精度。bit 15是符号位，0表示正数，1表示负数，例如0x8010表示-16。 取值范围：[0x0, 0xFFFF]。 #define HI_ISP_CCM_MATRIX_SIZE 9

注意事项

- 手动CCM模式下，用户可通过sat_en参数选择饱和度是否生效。
 - sat_en = 1，生效的CCM = Manual CCM * Saturation CCM，用户可用该功能确定不同照度下的饱和度数组；
 - sat_en = 0，生效的CCM = Manual CCM。

- `sat_en = 1`，生效的CCM会自动修正，保持每行参数的和为0x100。`sat_en = 0`，生效的CCM为手动设定参数值。

3.7 Auto Color Shading

3.7.1 hi_mpi_isp_get_auto_color_shading_attr

函数功能

获取Auto Color Shading属性参数。

函数原型

`hi_s32 hi_mpi_isp_get_auto_color_shading_attr(hi_vi_pipe vi_pipe, hi_isp_acs_attr *acs_attr)`

参数说明

参数名	输入/输出	说明
vi_pipe	输入	VI PIPE号。 取值范围：[0, 12)。
acs_attr	输出	Auto Color Shading 属性参数。

返回值说明

- 0：成功
- 非0：失败，参见[ISP返回码](#)

3.7.2 hi_mpi_isp_set_auto_color_shading_attr

函数功能

设置 Auto Color Shading 属性参数。

函数原型

`hi_s32 hi_mpi_isp_set_auto_color_shading_attr(hi_vi_pipe vi_pipe, const hi_isp_acs_attr *acs_attr)`

参数说明

参数名	输入/输出	说明
vi_pipe	输入	VI PIPE号。 取值范围：[0, 12)。

参数名	输入/输出	说明
acs_attr	输入	Auto Color Shading 属性参数。

返回值说明

- 0：成功
- 非0：失败，参见[ISP返回码](#)

3.7.3 数据类型

3.7.3.1 hi_isp_acs_attr

说明

定义Auto Color Shading属性参数。

定义

```
typedef struct {  
    hi_bool enable;  
    hi_u16 y_strength;  
    hi_u16 run_interval;  
    hi_bool lock_enable;  
} hi_isp_acs_attr;
```

成员

成员名称	描述
enable	ACS模块使能。 取值范围：HI_FALSE：关闭， HI_TRUE：使能。
y_strength	控制ACS算法Luma Shading也就是Y Shading的校正强度。 取值范围：[0x0, 0x100]。
run_interval	控制ACS算法多少帧运行一次，默认2帧运行一次。增大该值可以降低ACS算法的CPU占用率。 取值范围：[0x1, 0xff]。
lock_enable	固定ACS产生的Lut表，保持当前的LSC校正表。 取值范围：HI_FALSE：关闭， HI_TRUE：使能。

注意事项

- ACS模块开关与LSC的模块开关互不影响。ACS算法通过调用LSC的接口来配置Lut校正表，需要看到ACS模块效果时，Mesh Shading模块开关必须使能。
- 关闭ACS模块使能开关时，LSC的Lut表会恢复到未开ACS前的状态。

- ACS会参考mesh_scale的值来配置LSC相应的Lut表，cmos中bit1_lsc使能时会参考cmos中LSC的mesh_scale配置。
- 推荐mesh_scale的值取0~3，而不是4~7，因为4~7无法表示小于1倍的值，当y_strength的值较小时，校正Color Shading可能需要小于1倍的值。

3.8 查询内部状态信息（当前版本不支持）

3.8.1 功能描述

Inner State Information接口的作用是输出数个与ISO相关的参数其逻辑寄存器当前配置的真实值。

用户在调试过程中可以通过MPI接口读取这些真实值参数，观察参数强度是否正确配置。通过该接口用户只可以获取参数，而非操作改变该寄存器里的值。

3.8.2 hi_mpi_isp_query_inner_state_info

函数功能

获取内部寄存器实际强度配置和内部状态。

函数原型

hi_s32 hi_mpi_isp_query_inner_state_info(**hi_vi_pipe** vi_pipe,
hi_isp_inner_state_info *inner_state_info)

参数说明

参数名	输入/输出	说明
vi_pipe	输入	VI PIPE号。 取值范围：[0, 12)。
inner_state_info	输出	内部寄存器实际配置信息和状态值。

返回值说明

- 0：成功
- 非0：失败，参见[ISP返回码](#)

约束说明

- 在ISP算法未正常运行时，调用该接口可能无法获取到正确的实际强度配置和内部状态。
- 当用户切换分辨率或者在线性模式和WDR模式之间切换时候，可以调用这个接口来查询切换是否已经完成。
- 分辨率/帧率切换时，只查询res_switch_finish标记。模式切换时，只查询wdr_switch_finish标记。

- 业务正常运转时，在不改变WDR模式的情况下，合法调用hi_mpi_isp_set_pub_attr，查询到的wdr_switch_finish始终为 1。

3.8.3 数据类型

3.8.3.1 hi_isp_inner_state_info

说明

定义内部寄存器实际配置信息参数。

定义

```
typedef struct {  
    hi_u16 texture_strength[HI_ISP_SHARPEN_GAIN_NUM];  
    hi_u16 edge_strength[HI_ISP_SHARPEN_GAIN_NUM];  
    hi_u16 texture_freq;  
    hi_u16 edge_freq;  
    hi_u8 over_shoot;  
    hi_u8 under_shoot;  
    hi_u8 shoot_sup_strength;  
    hi_u16 coarse_strength[HI_ISP_BAYER_CHN_NUM];  
    hi_u8 wdr_frame_strength[HI_ISP_WDR_MAX_FRAME_NUM];  
    hi_u8 fine_strength;  
    hi_u16 coring_wgt;  
    hi_u16 dehaze_strength_actual;  
    hi_u16 drc_strength_actual;  
    hi_u32 wdr_exp_ratio_actual[HI_ISP_WDR_MAX_FRAME_NUM - 1];  
    hi_bool wdr_switch_finish;  
    hi_bool res_switch_finish;  
    hi_u16 black_level_actual[HI_ISP_BAYER_CHN_NUM];  
} hi_isp_inner_state_info;
```

成员

成员名称	描述
texture_strength[HI_ISP_SHARPEN_GAIN_NUM]	无方向的细节纹理的锐化强度，图像无方向的细节纹理的锐度。 取值范围：[0, 4095]。
edge_strength[HI_ISP_SHARPEN_GAIN_NUM]	带方向的边缘的锐化强度，图像带方向的边缘的锐度。 取值范围：[0, 4095]。
texture_freq	图像的无方向细节纹理的增强频段控制。图像的 细节纹理增强的频率。 取值范围：[0, 4095]。
edge_freq	图像的带方向的边缘的增强频段控制。图像 边缘增强的频率。 取值范围：[0, 4095]。
over_shoot	图像的overshoot（锐化后的白边白点）的 强度。 取值范围：[0, 127]。

成员名称	描述
under_shoot	图像的undershoot（锐化后的黑边黑点）的强度。 取值范围：[0, 127]。
shoot_sup_strength	图像锐化后的overshoot和undershoot的抑制强度。用于在保证清晰度不明显下降的前提下，抑制锐化后的图像的overshoot和undershoot的宽度和幅度。
coarse_strength	线性模式下控制亮度噪声整体去噪强度，四通道强度独立可调，值越大，亮噪去噪强度越大。 取值范围：[0, 0x360]。 #define HI_ISP_BAYER_CHN_NUM 4
wdr_frame_strength	WDR模式下每一融合帧分别对应的全局亮度去噪强度。 取值范围：[0, 0x50]。 #define HI_ISP_WDR_MAX_FRAME_NUM 4
fine_strength	亮噪去除强度，值越大，亮噪去除强度越大。 取值范围：[0x0,0x80]。
coring_wgt	随机噪声保留强度，值越大，保留的随机噪声越多。取值范围：[0x0,0xC80]。
dehaze_strength_actual	去雾强度值
drc_strength_actual	DRC强度生效值。取值范围：[0x0, 0x3FF]。
wdr_exp_ratio_actual	WDR曝光比 2to1时用wdr_exp_ratio_actual[0]表示曝光比 3to1时用wdr_exp_ratio_actual[0]及wdr_exp_ratio_actual[1]表示曝光比 4to1时用wdr_exp_ratio_actual[0]，wdr_exp_ratio_actual[1]，wdr_exp_ratio_actual[2]分别表示曝光比。
wdr_switch_finish	WDR切换是否完成标志。
res_switch_finish	分辨率切换是否完成标志。
black_level_actual	实际生效的黑电平的值，分别表示不同WDR通道R、Gr、Gb、B分量的黑电平。 取值范围：[0x0, 0xFFFF]。

注意事项

sensor-built-in模式下，查询到的实际生效的黑电平值为14bits；其他模式下，实际生效的黑电平值均为12bits。

3.9 PreGamma

3.9.1 hi_mpi_isp_get_pregamma_attr

PreGamma 函数功能

获取PreGamma属性参数。

函数原型

```
hi_s32 hi_mpi_isp_get_pregamma_attr(hi_vi_pipe vi_pipe,
hi_isp_pregamma_attr *pregamma_attr)
```

约束说明

PreGamma模块和DRC模块存在耦合，其只在DRC使能时才生效。PreGamma不支持乒乓切换，切换过程可能导致图像异常，该属性接口仅用来支持调试。

参数说明

参数名	输入/输出	说明
vi_pipe	输入	VI PIPE号。 取值范围：[0, 12)。
pregamma_attr	输入	PreGamma属性参数。

返回值说明

- 0：成功
- 非0：失败，参见ISP返回码。

3.9.2 hi_mpi_isp_set_pregamma_attr

函数功能

设置PreGamma属性参数。

函数原型

```
hi_s32 hi_mpi_isp_set_pregamma_attr(hi_vi_pipe vi_pipe, const
hi_isp_pregamma_attr *pregamma_attr)
```


约束说明

PreGamma模块和DRC模块存在耦合，其只在DRC使能时才生效。PreGamma不支持乒乓切换，切换过程可能导致图像异常，该属性接口仅用来支持调试；

参数说明

参数名	输入/输出	说明
vi_pipe	输入	VI PIPE号。 取值范围：[0, 12)。
pregamma_attr	输入	PreGamma 属性参数。

返回值说明

- 0：成功
- 非0：失败，参见[ISP返回码](#)。

3.9.3 数据类型

3.9.3.1 hi_isp_pregamma_attr

说明

定义PreGamma属性参数。

定义

```
typedef struct {  
    hi_bool enable;  
    hi_u32 table[HI_ISP_PREGAMMA_NODE_NUM];  
} hi_isp_pregamma_attr;
```

成员

成员名称	描述
enable	PreGamma使能开关。 取值范围：HI_FALSE：关闭，HI_TRUE：使能。
table[HI_ISP_PREGAMMA_NODE_NUM]	LUT表，用来表示PreGamma输出值大小。 取值范围：[0x0, 0xFFFFF]。 #define HI_ISP_PREGAMMA_NODE_NUM 257

注意事项

PreGamma模块和DRC模块存在耦合，其只在DRC使能时才生效。

3.10 CLUT

3.10.1 hi_mpi_isp_get_clut_coeff

函数功能

获取CLUT系数。

函数原型

hi_s32 hi_mpi_isp_get_clut_coeff(**hi_vi_pipe** vi_pipe, **hi_isp_clut_lut** *clut_lut)

参数说明

参数名	输入/输出	说明
vi_pipe	输入	VI PIPE号。 取值范围：[0, 12)。
clut_lut	输入	CLUT 系数。

返回值说明

- 0：成功
- 非0：失败，参见[ISP返回码](#)

3.10.2 hi_mpi_isp_set_clut_coeff

函数功能

设置CLUT系数。

函数原型

hi_s32 hi_mpi_isp_set_clut_coeff(**hi_vi_pipe** vi_pipe, const **hi_isp_clut_lut** *clut_lut)

参数说明

参数名	输入/输出	说明
vi_pipe	输入	VI PIPE号。 取值范围：[0, 12)。
clut_lut	输入	CLUT 系数。

返回值说明

- 0：成功
- 非0：失败，参见ISP返回码

3.10.3 hi_mpi_isp_get_clut_attr

函数功能

获取CLUT属性。

函数原型

```
hi_s32 hi_mpi_isp_get_clut_attr(hi_vi_pipe vi_pipe, hi_isp_clut_attr *clut_attr)
```

参数说明

参数名	输入/输出	说明
vi_pipe	输入	VI PIPE号。 取值范围：[0, 12)。
clut_attr	输入	CLUT属性。

返回值说明

- 0：成功
- 非0：失败，参见ISP返回码

3.10.4 hi_mpi_isp_set_clut_attr

函数功能

设置CLUT属性。

函数原型

```
hi_s32 hi_mpi_isp_set_clut_attr(hi_vi_pipe vi_pipe, const hi_isp_clut_attr *clut_attr)
```

参数说明

参数名	输入/输出	说明
vi_pipe	输入	VI PIPE号。 取值范围：[0, 12)。
clut_attr	输入	CLUT 属性。

返回值说明

- 0：成功
- 非0：失败，参见[ISP返回码](#)

3.10.5 数据类型

3.10.5.1 hi_isp_clut_attr

说明

定义CLUT的增益。

定义

```
typedef struct {  
    hi_bool enable;  
    hi_u32 gain_r;  
    hi_u32 gain_g;  
    hi_u32 gain_b;  
} hi_isp_clut_attr;
```

成员

成员名称	描述
enable	CLUT使能， 取值范围：HI_FALSE：关闭， HI_TRUE：使能。
gain_r	R通道增益，取值范围：[0,4095]。
gain_g	G通道增益，取值范围：[0,4095]。
gain_b	B通道增益，取值范围：[0,4095]。

注意事项

增益设定为512时，相当于增益是1.0。增益和LUT查表得出的调整量相乘，得到实际调整量。实际调整量和输入RGB值相加得到输出RGB值。

3.10.5.2 hi_isp_clut_lut

说明

定义CLUT的查找表。

定义

```
typedef struct {  
    hi_u32 lut[HI_ISP_CLUT_LUT_LENGTH];  
} hi_isp_clut_lut;
```

成员

成员名称	描述
lut	查找表取值范围：[0, 1073741823]。 注：lut表中每个元素的有效范围是30bit，若设置超过取值范围的 值，实际生效范围仍是30bit #define HI_ISP_CLUT_LUT_LENGTH5508

注意事项

用户需要自行生成3D LUT表，将17x17x17x3的LUT放入8x3的LUT，然后将8x3的LUT紧凑排列填入长度为5508的LUT。

17x17x17x3的LUT每个元素的范围为[-511, 511]，使用10bit有符号数表示。将17x17x17x3的LUT取成8x3的LUT的过程为：

三维表的下标为偶数时定义为0，下标为奇数时定义为1。R[i][j][k]，G[i][j][k]， B[i][j][k]中的i代表r的分格，j代表b的分格，k代表g的分格，即ijk代表三维表的地址坐标，对应的是rbg的排列顺序。R[i][j][k]，G[i][j][k]， B[i][j][k]的值是deltaR，deltaG，deltaB的值。用C代表RGB，真值表如下所示：

k	j	i	C
0	0	0	0
0	0	1	1
0	1	0	2
0	1	1	3
1	0	0	4
1	0	1	5
1	1	0	6
1	1	1	7

上图对应的伪代码如下所示：

```
for(k = 0; k < 17; k++) {
    for(j = 0; j < 17; j++) {
        for(i = 0; i < 17; i++) {
            switch((i&1) + ((j&1)<<1) + ((k&1)<<2)) {
                case 0:
                    R0[n0]=R[i][j][k];
                    G0[n0]=G[i][j][k];
                    B0[n0]=B[i][j][k];
                case 1:
                    R1[n1]=R[i][j][k];
                    G1[n1]=G[i][j][k];
                    B1[n1]=B[i][j][k];
                case 2:
                    R2[n2]=R[i][j][k];
                    G2[n2]=G[i][j][k];
                    B2[n2]=B[i][j][k];
                case 3:
                    R3[n3]=R[i][j][k];
                    G3[n3]=G[i][j][k];
                    B3[n3]=B[i][j][k];
                case 4:
                    R4[n4]=R[i][j][k];
                    G4[n4]=G[i][j][k];
                    B4[n4]=B[i][j][k];
            }
        }
    }
}
```

```
case 5:
    R5[n5]=R[i][j][k];
    G5[n5]=G[i][j][k];
    B5[n5]=B[i][j][k];
case 6:
    R6[n6]=R[i][j][k];
    G6[n6]=G[i][j][k];
    B6[n6]=B[i][j][k];
case 7:
    R7[n7]=R[i][j][k];
    G7[n7]=G[i][j][k];
    B7[n7]=B[i][j][k];
    }
    }
}
```

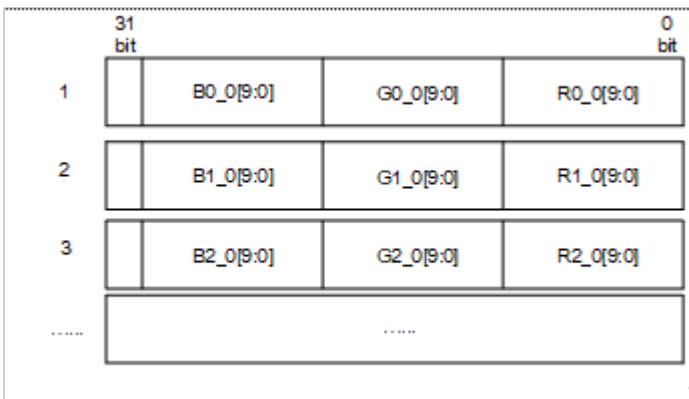
用C代表RGB，取成的8个LUT分别为c0[729]、c1[648]、c2[648]、c3[576]、c4[648]、c5[576]、c6[576]、c7[512]。

下面的描述用_0代表一维数组c0[729]第0个数据，_1代表一维数组c0[729]第1个数据。

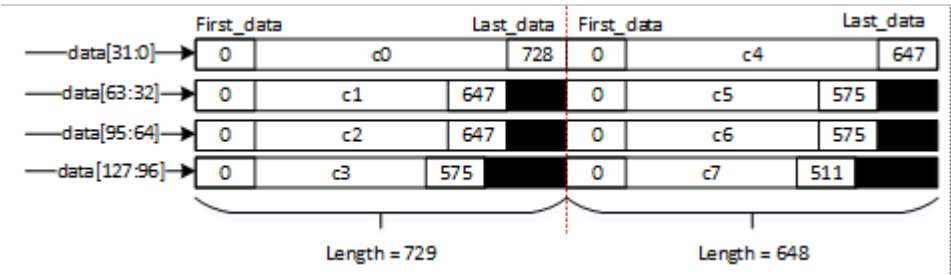
8x3的LUT按照每3个数据紧凑排列到1个32bit数据内的格式定义为：

- {2' b00,B0_0,G0_0,R0_0},
- {2' b00,B1_0,G1_0,R1_0},
- {2' b00,B2_0,G2_0,R2_0},
- {2' b00,B3_0,G3_0,R3_0},
- {2' b00,B0_1,G0_1,R0_1},
- {2' b00,B1_1,G1_1,R1_1},
- {2' b00,B2_1,G2_1,R2_1},
- {2' b00,B3_1,G3_1,R3_1},
- {.....},

上述格式可视化如下图所示：



整体看如下图所示：



8x3的LUT按照c0[729] 占用空间729 x 4 = 2916和c4[648] 占用空间648 x 4 = 2592连续存储成一个长度为5508的一维数组。

3.11 CSC

色彩空间转换模块(Color Space Conversion)，用来将RGB信号转换成符合标准的YUV信号。

3.11.1 hi_mpi_isp_set_csc_attr

函数功能

设定CSC（色彩空间转换）属性。

函数原型

```
hi_s32 hi_mpi_isp_set_csc_attr(hi_vi_pipe vi_pipe, const hi_isp_csc_attr *csc_attr)
```

参数说明

参数名	输入/输出	说明
vi_pipe	输入	VI PIPE号。 取值范围：[0, 12)。
csc_attr	输入	CSC 属性。

返回值说明

- 0：成功
- 非0：失败，参见ISP返回码

3.11.2 hi_mpi_isp_get_csc_attr

函数功能

获取CSC（色彩空间转换）属性。

函数原型

```
hi_s32 hi_mpi_isp_get_csc_attr(hi_vi_pipe vi_pipe, hi_isp_csc_attr *csc_attr)
```

参数说明

参数名	输入/输出	说明
vi_pipe	输入	VI PIPE号。 取值范围：[0, 12)。
csc_attr	输入	CSC属性。

返回值说明

- 0：成功
- 非0：失败，参见ISP返回码

3.11.3 数据类型

3.11.3.1 hi_isp_csc_attr

说明定义ISP色彩空间转换模块属性。

定义

```
typedef struct {  
    hi_bool enable;  
    hi_color_gamut color_gamut;  
    hi_u8 hue;  
    hi_u8 luma;  
    hi_u8 contr;  
    hi_u8 satu;  
    hi_bool limited_range_en;  
    hi_bool ext_csc_en;  
    hi_bool ct_mode_en;  
    hi_isp_csc_matrix csc_magtrx;  
} hi_isp_csc_attr;
```

成员

成员名称	描述
enable	是否使能CSC功能。 取值范围： - HI_FALSE：否 - HI_TRUE：是
color_gamut	色彩空间类型枚举。
hue	目标YUV空间色调。 取值范围：[0,100]。

成员名称	描述
luma	目标YUV空间亮度。 取值范围：[0,100]。
contr	目标YUV空间对比度。 取值范围：[0,100]。
satu	目标YUV空间饱和度。 取值范围：[0,100]。
limited_range_en	目标YUV空间动态范围选择。 取值范围： - HI_FALSE：输出为Full Range - HI_TRUE：输出为Limited Range
ext_csc_en	选择亮度调整模式（与luma配合）。 - HI_FALSE：表示luma不可以调节更宽的亮度变化 - HI_TRUE：表示luma可以调节更宽的亮度变化
ct_mode_en	选择输出的对比度调整模式。 - HI_FALSE：改变contr画面亮度会变化 - HI_TRUE：改变contr画面亮度不会变化
csc_magtrx	色彩空间转换矩阵系数。

注意事项

- color_gamut中可选择HI_COLOR_GAMUT_BT601，HI_COLOR_GAMUT_BT709，HI_COLOR_GAMUT_USER三种模式。在HI_COLOR_GAMUT_BT601和HI_COLOR_GAMUT_BT709模式下，csc模块采用特定的csc_magtrx，用户无法在这两个模式下修改csc_magtrx；只有在HI_COLOR_GAMUT_USER模式下，用户可根据实际情况自由设定csc_magtrx的参数来进行色彩空间转换
- 如果将contr值调小（20左右）时，在平坦区域容易出现等高线，出现的原因是因为contr值调小后，会影响画面的动态范围，导致原本0~255（或者10bit 0~1023）的动态范围缩小到原来的20%左右，因此在平坦区域如果有亮度变化，则非常容易产生量化误差出现等高线。

3.11.3.2 hi_isp_csc_matrix

说明

定义色彩空间转换矩阵系数。

定义

```
typedef struct {  
    hi_s16 csc_in_dc[HI_ISP_CSC_DC_NUM];  
    hi_s16 csc_out_dc[HI_ISP_CSC_DC_NUM];  
}
```

```
hi_s16 csc_coef[HI_ISP_CSC_COEF_NUM];  
} hi_isp_csc_matrix;
```

成员

成员名称	描述
csc_in_dc[HI_ISP_CSC_DC_NUM]	CSC转换矩阵，输入直流分量。 取值范围： [-1024,1023]。 #define HI_ISP_CSC_DC_NUM 3
csc_out_dc[HI_ISP_CSC_DC_NUM]	CSC转换矩阵，输出直流分量。 取值范围： [-1024,1023]。
csc_coef[HI_ISP_CSC_COEF_NUM]	CSC转换矩阵，3x3系数。取值范围： [-4096, 4095]。 #define HI_ISP_CSC_COEF_NUM 9

3.12 LDCI

3.12.1 hi_mpi_isp_get_ldci_attr

函数功能

获取LDCI属性参数。

函数原型

```
hi_s32 hi_mpi_isp_get_ldci_attr(hi_vi_pipe vi_pipe, hi_isp_ldci_attr *ldci_attr)
```

约束说明

开启LDCI的图像分辨率高度最大支持8192，当高度超过8192时，自动关闭LDCI功能。

参数说明

参数名	输入/输出	说明
vi_pipe	输入	VI PIPE号。 取值范围： [0, 12)。
ldci_attr	输入	LDCI 属性参数。

返回值说明

- 0：成功
- 非0：失败，参见ISP返回码

3.12.2 hi_mpi_isp_set_ldci_attr

函数功能

设置LDCI属性参数。

函数原型

```
hi_s32 hi_mpi_isp_set_ldci_attr(hi_vi_pipe vi_pipe, const hi_isp_ldci_attr *ldci_attr)
```

约束说明

开启LDCI的图像分辨率高度最大支持8192，当高度超过8192时，自动关闭LDCI功能。

参数说明

参数名	输入/输出	说明
vi_pipe	输入	VI PIPE号。 取值范围：[0, 12)。
ldci_attr	输入	LDCI 属性参数。

返回值说明

- 0：成功
- 非0：失败，参见[ISP返回码](#)。

3.12.3 数据类型

3.12.3.1 hi_isp_ldci_gauss_coef_attr

说明

定义GAUSS分布属性参数。

定义

```
typedef struct {  
    hi_u8 wgt;  
    hi_u8 sigma;  
    hi_u8 mean;  
} hi_isp_ldci_gauss_coef_attr;
```

成员

成员名称	描述
wgt	局域直方图均衡结果与原图融合权重曲线，根据亮度控制提亮程度，曲线呈GAUSS分布，该参数为GAUSS分布最大值，取值范围：[0x0, 0xFF]。
sigma	局域直方图均衡结果与原图融合权重曲线，根据亮度控制提亮程度，曲线呈GAUSS分布，该参数为GAUSS分布均方差。取值范围：[0x1, 0xFF]。
mean	局域直方图均衡结果与原图融合权重曲线，根据亮度控制提亮程度，曲线呈GAUSS分布，该参数为GAUSS分布数学期望。取值范围：[0x0, 0xFF]。

3.12.3.2 hi_isp_ldci_he_wgt_attr

说明

定义直方图均衡与原图融合权重属性参数。

定义

```
typedef struct {  
    hi_isp_ldci_gauss_coef_attr he_pos_wgt;  
    hi_isp_ldci_gauss_coef_attr he_neg_wgt;  
} hi_isp_ldci_he_wgt_attr;
```

成员

成员名称	描述
he_pos_wgt	直方图均衡结果与原图融合权重曲线，可根据原图亮度控制提亮程度；曲线呈高斯分布，横坐标为亮度。
he_neg_wgt	直方图均衡结果与原图融合权重曲线，可根据原图亮度控制减暗程度；曲线呈高斯分布，横坐标为亮度。

3.12.3.3 hi_isp_ldci_manual_attr

说明

定义LDCI手动模式属性参数。

定义

```
typedef struct {  
    hi_isp_ldci_he_wgt_attr he_wgt;  
    hi_u16 blc_ctrl;  
} hi_isp_ldci_manual_attr;
```

成员

成员名称	描述
he_wgt	手动模式下控制直方图均衡结果与原图融合权重曲线，可根据原图亮度分开控制提亮减暗程度；曲线呈高斯分布，横坐标为亮度。
blc_ctrl	手动模式下暗区增益控制强度，用于抑制暗区噪声。 取值范围：[0x0, 0x1FF]。

3.12.3.4 hi_isp_ldci_auto_attr

说明

定义LDCI自动模式下属性参数。

定义

```
typedef struct {  
    hi_isp_ldci_he_wgt_attr he_wgt[HI_ISP_AUTO_ISO_NUM];  
    hi_u16 blc_ctrl[HI_ISP_AUTO_ISO_NUM];  
} hi_isp_ldci_auto_attr;
```

成员

成员名称	描述
he_wgt[HI_ISP_AUTO_ISO_NUM]	自动模式下控制直方图均衡结果与原图融合权重曲线，可根据原图亮度分开控制提亮减暗程度；曲线呈高斯分布，横坐标为亮度。 #define HI_ISP_AUTO_ISO_NUM 16
blc_ctrl[HI_ISP_AUTO_ISO_NUM]	自动模式下暗区增益控制强度，用于抑制暗区噪声。 取值范围：[0x0, 0x1FF]。

3.12.3.5 hi_isp_ldci_attr

说明

定义LDCI属性参数。

定义

```
typedef struct {  
    hi_bool enable;  
    hi_u8 gauss_lpf_sigma;  
    hi_op_mode op_type;  
    hi_isp_ldci_manual_attr manual_attr;  
    hi_isp_ldci_auto_attr auto_attr;  
    hi_u16 tpr_incr_coef;  
    hi_u16 tpr_decr_coef;  
} hi_isp_ldci_attr;
```

成员

成员名称	描述
enable	LDCI功能使能开关。 取值范围： - HI_FALSE：关闭 - HI_TRUE：使能
gauss_lpf_sigma	LPF滤波强度，其值越小说明LDCI作用越局域。 取值范围：[0x1, 0xFF]。
op_type	LDCI工作类型。 - HI_OP_MODE_AUTO：自动 - HI_OP_MODE_MANUAL：手动
manual_attr	LDCI手动参数。
auto_attr	LDCI自动参数。
tpr_incr_coef	画面由暗到亮变化时域滤波系数，值越大，当前帧像素比重越大。 取值范围：[0x0, 0x100]。
tpr_decr_coef	画面由亮到暗变化时域滤波系数，值越大，当前帧像素比重越大。 取值范围：[0x0, 0x100]。

注意事项

开启LDCI的图像分辨率高度最大支持8192，当高度超过8192时，自动关闭LDCI功能。

3.13 ColorTone

3.13.1 hi_mpi_isp_set_color_tone_attr

函数功能

设置色调的参数。

函数原型

```
hi_s32 hi_mpi_isp_set_color_tone_attr(hi_vi_pipe vi_pipe, const hi_isp_color_tone_attr *ct_attr)
```

参数说明

参数名	输入/输出	说明
vi_pipe	输入	VI PIPE号。 取值范围：[0, 12)。
ct_attr	输入	色调的参数。

返回值说明

- 0：成功
- 非0：失败，参见[ISP返回码](#)

3.13.2 hi_mpi_isp_get_color_tone_attr

函数功能

获取色调的参数。

函数原型

[hi_s32](#) hi_mpi_isp_get_color_tone_attr([hi_vi_pipe](#) vi_pipe,
[hi_isp_color_tone_attr](#) *ct_attr)

参数说明

参数名	输入/输出	说明
vi_pipe	输入	VI PIPE号。 取值范围：[0, 12)。
ct_attr	输入	色调的参数。

返回值说明

- 0：成功
- 非0：失败，参见[ISP返回码](#)

3.13.3 数据类型

3.13.3.1 hi_isp_color_tone_attr

说明

定义色调调整属性。

定义

```
typedef struct {  
    hi_u16 red_cast_gain;  
    hi_u16 green_cast_gain;  
    hi_u16 blue_cast_gain;  
} hi_isp_color_tone_attr;
```

成员

成员名称	描述
red_cast_gain	R通道增益，8bit小数精度。取值范围：[0x100, 0x180]。
green_cast_gain	G通道增益，8bit小数精度。取值范围：[0x100, 0x180]。
blue_cast_gain	B通道增益，8bit小数精度。取值范围：[0x100, 0x180]。

 **注意**

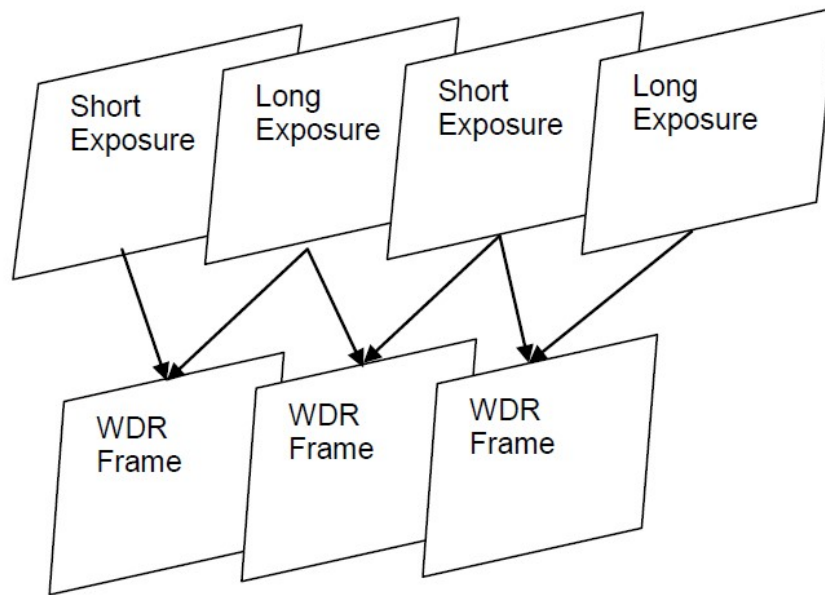
ColorTone支持客户调节固定的颜色风格，与色温无关。

3.14 WDR

由于传统的图像传感器的动态范围比较低，因此，如果用传统的图像传感器拍摄高动态范围的场景，就会要么亮处过曝，丢失很多细节；要么暗处曝光不足，细节难以分辨。为了记录高动态范围场景的每一个细节，可以采用多次曝光图像合成的方法。

WDR算法的目的就是将从sensor输出的长短帧数据合成为一帧20bit的数据。如图3-6所示，长短帧数据是先缓存在DDR中，然后同时给WDR模块进行合成。合成的原则是择优选取，即对于比较亮的区域，采用短帧数据，而对于比较暗的区域，采用长帧数据。

图 3-6 WDR 算法合成模型



- 本模块可以通过 `hi_isp_wdr_merge_mode` 来决定合成的方式。若该值为0，则用传统的WDR功能合成；若该值为1，则打开 Fusion功能，即用来改善Flicker现象；其中 `hi_isp_wdr_combine_attr`结构体用来控制传统的WDR功能，而 `hi_isp_fusion_attr`结构体用来控制Fusion功能。
- 传统的WDR，包括以下功能：
 - 可以去除鬼影。通过 `hi_isp_wdr_combine_attr`来选择融合区域是采用短帧还是长帧；
 - 可以通过`short_threshold`和`long_threshold`来控制选择长短帧的范围。即对于大于`short_threshold`的像素选择短帧，小于`long_threshold`的像素选择长帧，而对于在两者之间的像素采用长短帧融合的方式；
 - 可以通过`force_long_low_threshold`和`force_long_hig_threshold`来控制运动区域选择长帧，对于亮度小于`force_long_low_threshold`的运动区域强制选择长帧，亮度大于`force_long_hig_threshold`的运动区域不强制选择长帧，对于亮度在两者之间的运动区域采用长短帧融合的方式；
 - 可以通过`md_thr_low_gain`，`md_thr_hig_gain`来控制运动检测；
 - 可以通过`md_ref_flicker`来决定运动检测是否参考Flicker检测结果；
 - 可以通过`short_expo_chk`来控制WDR融合是否检查短帧数据大小。
- Fusion功能指适用于一般的室内场景（非宽动态）以及夜晚场景，用来改善工频闪问题。其中`fusion_threshold` [4]是用来控制Fusion选择时每帧的阈值。

3.14.1 hi_mpi_isp_get_fswdr_attr

函数功能

获取帧合成参数。

函数原型

```
hi_s32 hi_mpi_isp_get_fswdr_attr(hi_vi_pipe vi_pipe, hi_isp_wdr_fs_attr *fswdr_attr)
```

参数说明

参数名	输入/输出	说明
vi_pipe	输入	VI PIPE号。 取值范围：[0, 12)。
fswdr_attr	输入	帧合成参数。

返回值说明

- 0：成功
- 非0：失败，参见[ISP返回码](#)

3.14.2 hi_mpi_isp_set_fswdr_attr

函数功能

设置帧合成参数。

函数原型

```
hi_s32 hi_mpi_isp_set_fswdr_attr(hi_vi_pipe vi_pipe, const hi_isp_wdr_fs_attr *fswdr_attr)
```

参数说明

参数名	输入/输出	说明
vi_pipe	输入	VI PIPE号。 取值范围：[0, 12)。
fswdr_attr	输入	帧合成参数。

返回值说明

- 0：成功
- 非0：失败，参见[ISP返回码](#)

3.14.3 数据类型

3.14.3.1 hi_isp_wdr_merge_mode

说明

定义ISP合成方式属性。

定义

```
typedef enum {  
    HI_ISP_MERGE_WDR_MODE    = 0,  
    HI_ISP_MERGE_FUSION_MODE = 1,  
    HI_ISP_MERGE_BUTT  
} hi_isp_wdr_merge_mode;
```

成员

成员名称	描述
HI_ISP_MERGE_WDR_MODE	运行在传统WDR模式下。
HI_ISP_MERGE_FUSION_MODE	运行在FUSION模式下。
HI_ISP_MERGE_BUTT	无效值。

3.14.3.2 hi_isp_fswdr_mdt_attr

说明

定义ISPFSDR运动检测属性。

定义

```
typedef struct {  
    hi_bool short_expo_chk;  
    hi_u16  short_check_thd;  
    hi_bool md_ref_flicker;  
    hi_u8   mdt_still_thd;  
    hi_u8   mdt_full_thd;  
    hi_u8   mdt_long_blend;  
    hi_op_mode op_type;  
    hi_isp_fswdr_manual_attr manual_attr;  
    hi_isp_fswdr_auto_attr  auto_attr;  
} hi_isp_fswdr_mdt_attr;
```

成员

成员名称	描述
short_expo_chk	控制运动检测是否检查短帧数据大小，当短帧数据接近 0 时，WDR融合强制选长帧。 取值范围： HI_FALSE：WDR融合时不检查短帧数据大小； HI_TRUE：WDR融合时检查短帧数据大小。

成员名称	描述
short_check_thd	短帧检查阈值。当短帧亮度小于该阈值时，WDR强制输出长帧。 取值范围：[0x,0xFFF]
md_ref_flicker	控制运动判断是否参考Flicker检测结果。 取值范围： HI_FALSE：做运动判断时不参考Flicker检测结果； HI_TRUE：做运动判断时参考Flicker检测结果。
mdt_still_thd	运动检测中判断为静止的阈值，当运动程度小于该阈值时，判断该像素点为静止。 取值范围：[0,0xFE]。
mdt_full_thd	运动检测中判断为全运动的阈值，当运动程度大于该阈值时，判断该像素点为全运动。仅支持3合1WDR。 取值范围：[0,0xFE]。
mdt_long_blend	运动区域叠加长帧的权重，当该值为0xFE 时，运动区域全部选择长帧。 取值范围：[0x0,0xFE]。
op_type	FSWDR工作类型。 - HI_OP_MODE_AUTO：自动； - HI_OP_MODE_MANUAL：手动。
manual_attr	FSWDR手动参数。
auto_attr	FSWDR自动参数。

注意事项

WDR功能分为自动和手动：

- op_type为HI_OP_MODE_AUTO，使用自动WDR功能。
- op_type设置为HI_OP_MODE_MANUAL，使用手动WDR功能。

3.14.3.3 hi_isp_wdr_combine_attr

说明

定义ISP FSWDR combine属性。

定义

```
typedef struct {
    hi_bool motion_comp;
    hi_u16 short_thr;
    hi_u16 long_thr;
    hi_bool force_long;
    hi_u16 force_long_low_thr;
    hi_u16 force_long_hig_thr;
```

```
hi_isp_fswdr_mdt_attr wdr_mdt;  
} hi_isp_wdr_combine_attr;
```

成员

成员名称	描述
motion_comp	WDR运动补偿使能。取值范围： HI_FALSE：关闭； HI_TRUE：使能。 在某些室内日光灯场景下，运动补偿会加重频闪带来的横条纹现象，此时可以关闭运动补偿。
short_thr	短曝光门限值，超过门限值的图像数据只选择短曝光数据。 取值范围：[0,0xFFF]。
long_thr	长曝光门限值，低于门限值的图像数据只选择长曝光数据。 取值范围：[0, short_thr]。
force_long	强制输出长帧使能。 取值范围： HI_FALSE：关闭； HI_TRUE：使能。
force_long_low_thr	强制输出长帧的低阈值，亮度低于该阈值的运动区域强制输出长帧。 取值范围：[0x0, 0xFFF]。
force_long_hig_thr	强制输出长帧的高阈值，亮度大于该阈值的运动区域不强制。 输出长帧取值范围：[0x0, 0xFFF]
wdr_mdt	FSWDR运动检测参数，具体参见hi_isp_fswdr_mdt_attr。

注意事项

注意force_long_low_thr应小于force_long_hig_thr。

3.14.3.4 hi_isp_fusion_attr

说明

定义ISP FUSION属性。

定义

```
typedef struct {  
    hi_u16 fusion_thr[HI_ISP_WDR_MAX_FRAME_NUM];  
} hi_isp_fusion_attr;
```

成员

成员名称	描述
fusion_thr[HI_ISP_WDR_MAX_FRAME_NUM]	fusion_thr[0]，fusion_thr[1]，fusion_thr[2]，fusion_thr[3]分别 对应Fusion模式下超短帧，短帧，中帧，长帧的阈值，将输入数据限制在阈值以下。 当前仅支持超短帧和短帧。 取值范围：[0,0x3FFF]。 #define HI_ISP_WDR_MAX_FRAME_NUM 4

3.14.3.5 hi_isp_fswdr_manual_attr

说明

定义ISP FSWDR手动属性。

定义

```
typedef struct {  
    hi_u8 md_thr_low_gain;  
    hi_u8 md_thr_hig_gain;  
} hi_isp_fswdr_manual_attr;
```

成员

成员名称	描述
md_thr_low_gain	手动模式下判断是否运动的低阈值系数。值越大，运动程度越小。 取值范围：[0, md_thr_hig_gain]。
md_thr_hig_gain	手动模式下判断是否运动的高阈值系数。值越大，运动程度越小。 取值范围：[0, 0xFF]。

3.14.3.6 hi_isp_fswdr_auto_attr

说明

定义ISP FSWDR自动属性。

定义

```
typedef struct {  
    hi_u8 md_thr_low_gain[HI_ISP_AUTO_ISO_NUM];  
    hi_u8 md_thr_hig_gain[HI_ISP_AUTO_ISO_NUM];  
} hi_isp_fswdr_auto_attr;
```

成员

成员名称	描述
md_thr_low_gain[HI_ISP_AUTO_ISO_NUM]	自动模式下判断是否运动的低阈值系数。值越大，运动程度越小。 取值范围：[0,0xFF]。 #define HI_ISP_AUTO_ISO_NUM 16
md_thr_hig_gain[HI_ISP_AUTO_ISO_NUM]	自动模式下判断是否运动的高阈值系数。值越大，运动程度越小。 取值范围：[0,0xFF]。

3.14.3.7 hi_isp_wdr_fs_attr

说明

定义ISP FSWDR属性。

定义

```
typedef struct {  
    hi_isp_wdr_merge_mode wdr_merge_mode;  
    hi_isp_wdr_combine_attr wdr_combine;  
    hi_isp_fusion_attr fusion_attr;  
} hi_isp_wdr_fs_attr;
```

成员

成员名称	描述
wdr_merge_mode	合成方式。 取值范围： HI_ISP_MERGE_WDR_MODE：传统的WDR合成； HI_ISP_MERGE_FUSION_MODE：Fusion合成。 在室内正常照度下存在工频闪时或者夜晚场景，建议采用Fusion合成，具体参见hi_isp_wdr_merge_mode。
wdr_combine	传统的WDR合成结构体，具体参见hi_isp_wdr_combine_attr。
fusion_attr	Fusion合成结构体，具体参见hi_isp_fusion_attr。

3.15 CAC

3.15.1 hi_mpi_isp_get_global_cac_attr

函数功能

获取色差校正参数。

函数原型

hi_s32 hi_mpi_isp_get_global_cac_attr(**hi_vi_pipe** vi_pipe,
hi_isp_global_cac_attr *global_cac_attr)

参数说明

参数名	输入/输出	说明
vi_pipe	输入	VI PIPE号。 取值范围：[0, 12)。
global_cac_attr	输入	横向色差校正参数。

返回值说明

- 0：成功
- 非0：失败，参见[ISP返回码](#)

3.15.2 hi_mpi_isp_set_global_cac_attr

函数功能

设置横向色差校正参数。

函数原型

hi_s32 hi_mpi_isp_set_global_cac_attr(**hi_vi_pipe** vi_pipe, const
hi_isp_global_cac_attr *global_cac_attr)

参数说明

参数名	输入/输出	说明
vi_pipe	输入	VI PIPE号。 取值范围：[0, 12)。
global_cac_attr	输入	横向色差校正参数。

返回值说明

- 0：成功

- 非0：失败，参见[ISP返回码](#)

3.15.3 hi_mpi_isp_get_local_cac_attr

函数功能

获取紫边检测校正参数。

函数原型

[hi_s32](#) hi_mpi_isp_get_local_cac_attr([hi_vi_pipe](#) vi_pipe, [hi_isp_local_cac_attr](#) *local_cac_attr)

参数说明

参数名	输入/输出	说明
vi_pipe	输入	VI PIPE号。 取值范围：[0, 12)。
local_cac_attr	输入	紫边检测校正参数。

返回值说明

- 0：成功
- 非0：失败，参见[ISP返回码](#)

3.15.4 hi_mpi_isp_set_local_cac_attr

函数功能

设置紫边检测校正参数。

函数原型

[hi_s32](#) hi_mpi_isp_set_local_cac_attr([hi_vi_pipe](#) vi_pipe, const [hi_isp_local_cac_attr](#) *local_cac_attr)

参数说明

参数名	输入/输出	说明
vi_pipe	输入	VI PIPE号。 取值范围：[0, 12)。
local_cac_attr	输入	紫边检测校正参数。

返回值说明

- 0：成功
- 非0：失败，参见[ISP返回码](#)

3.15.5 数据类型

3.15.5.1 hi_isp_depurplestr_manual_attr

说明

定义紫边校正手动属性。

定义

```
typedef struct {  
    hi_u8 de_purple_cr_str;  
    hi_u8 de_purple_cb_str;  
} hi_isp_depurplestr_manual_attr;
```

成员

成员名称	描述
de_purple_cr_str	R通道的校正强度，取值范围：[0，8]。
de_purple_cb_str	B通道的校正强度，取值范围：[0，8]。

注意事项

de_purple_cb_str和de_purple_cr_str矫正强度过大，可能出现边界不连续现象或闪烁现象。

3.15.5.2 hi_isp_depurplestr_auto_attr

说明

定义紫边校正自动属性。

定义

```
typedef struct {  
    hi_u8 de_purple_cr_str[HI_ISP_LCAC_EXP_RATIO_NUM];  
    hi_u8 de_purple_cb_str[HI_ISP_LCAC_EXP_RATIO_NUM];  
} hi_isp_depurplestr_auto_attr;
```

成员

成员名称	描述
de_purple_cr_str[HI_ISP_LCAC_EXP_RATIO_NUM]	自动模式下，R通道的校正强度。该组的16个值分别对应16个不同的曝光比，一般情况下，曝光比越大，配置的去紫边的强度值越大，对应关系如表1所示。若连续两档的处理强度不相等，则通过线性插值的方法确定两档中间某个曝光比值对应的处理强度。 取值范围：[0, 8]。 #define HI_ISP_LCAC_EXP_RATIO_NUM 16
de_purple_cb_str[HI_ISP_LCAC_EXP_RATIO_NUM]	自动模式下，B通道的校正强度。 取值范围：[0, 8]。

表 3-5 de_purple_cr_str[16]在不同曝光比情况下对应的设置值

de_purple_cr_strength	曝光比
de_purple_cr_str[0]	64
de_purple_cr_str[1]	128
de_purple_cr_str[2]	192
de_purple_cr_str[3]	256
de_purple_cr_str[4]	320
de_purple_cr_str[5]	384
de_purple_cr_str[6]	448
de_purple_cr_str[7]	512
de_purple_cr_str[8]	576
de_purple_cr_str[9]	640
de_purple_cr_str[10]	704
de_purple_cr_str[11]	768
de_purple_cr_str[12]	832
de_purple_cr_str[13]	896
de_purple_cr_str[14]	960
de_purple_cr_str[15]	1023

注意事项

- 曝光比是WDR模式下才有的概念，线性模式下获取的曝光比值总为1，因此在线性模式下虽然有联动，生效的总是第一组配置值。
- 在built-in WDR模式下，本参数并不和曝光比联动，为与线性模式区分，生效的总是最后一组即曝光比为1023的配置值。
- 在WDR自动长帧模式和WDR长帧模式下，曝光比生效的总是第一组配置，不会发生变化。

3.15.5.3 hi_isp_local_cac_attr

说明

紫边检测校正属性。

定义

```
typedef struct {  
    hi_bool enable;  
    hi_u16 purple_detect_range;  
    hi_u16 var_threshold;  
    hi_u16 r_detect_threshold[HI_ISP_LCAC_DET_NUM];  
    hi_u16 g_detect_threshold[HI_ISP_LCAC_DET_NUM];  
    hi_u16 b_detect_threshold[HI_ISP_LCAC_DET_NUM];  
    hi_u16 luma_detect_threshold[HI_ISP_LCAC_DET_NUM];  
    hi_s16 cb_cr_ratio[HI_ISP_LCAC_DET_NUM];  
    hi_op_mode op_type;  
    hi_isp_depurplestr_manual_attr manual_attr;  
    hi_isp_depurplestr_auto_attr auto_attr;  
} hi_isp_local_cac_attr;
```

成员

成员名称	描述
enable	紫边校正使能。 取值范围：HI_FALSE：禁止；HI_TURE：使能。
purple_detect_range	紫色检测的范围，该值控制的是r_detect_threshold，g_detect_threshold，b_detect_threshold，luma_detect_threshold，cb_cr_ratio这几个参数的生效情况。值越大，越多非高亮区域的紫色被界定为紫边区域。 purple_detect_range可能会引入紫边去除不平滑的问题，可以检查r_detect_threshold，g_detect_threshold，b_detect_threshold，luma_detect_threshold，cb_cr_ratio这几个参数的配置。 取值范围：[0,410]。
var_threshold	边缘检测阈值。 取值范围：[0, 4095]。

成员名称	描述
r_detect_threshold[HI_ISP_LCAC_DET_NUM]	它分了3段，每一段的值表示高亮检测模块里面R分量阈值。 实际中生效的值取决于purple_detect_range这个参数的大小。purple_detect_range越小，越倾向于r_detect_threshold第一段的值，purple_detect_range越大，越倾向于r_detect_threshold第3段的值。 取值范围：[0, 4095]。 #define HI_ISP_LCAC_DET_NUM 3
g_detect_threshold[HI_ISP_LCAC_DET_NUM]	它分了3段，每一段的值表示高亮检测模块里面G分量阈值。 实际中生效的值取决于purple_detect_range这个参数的大小。purple_detect_range越小，越倾向于g_detect_threshold第一段的值，purple_detect_range越大，越倾向于g_detect_threshold第3段的值。 取值范围：[0, 4095]。
b_detect_threshold[HI_ISP_LCAC_DET_NUM]	它分了 3 段，每一段的值表示高亮检测模块里面B分量阈值。 实际中生效的值取决于purple_detect_range这个参数的大小。purple_detect_range越小，越倾向于b_detect_threshold第一段的值，purple_detect_range越大，越倾向于b_detect_threshold第3段的值。 取值范围：[0, 4095]。
luma_detect_threshold[HI_ISP_LCAC_DET_NUM]	它分了3段，每一段的值表示高亮检测模块里面Luma分量阈值。实际中生效的值取决于purple_detect_range这个参数的大小。purple_detect_range越小，越倾向于luma_detect_threshold第一段的值，purple_detect_range越大，越倾向于 luma_detect_threshold第 3 段的值。 取值范围：[0, 4095]
cb_cr_ratio[HI_ISP_LCAC_DET_NUM]	它分了3段，每一段的值表示紫色检测模块蓝色程度。实际中生效的值取决于 purple_detect_range这个参数的大小。purple_detect_range越小，越倾向于cb_cr_ratio 第一段的值，purple_detect_range越大，越倾向于cb_cr_ratio第3段的值。 取值范围：[-2048, 2047]。
op_type	紫边校正工作模式。 - HI_OP_MODE_AUTO：自动； - HI_OP_MODE_MANUAL：手动。
manual_attr	手动模式下配置紫边校正强度参数。
auto_attr	自动模式下配置紫边校正强度参数。

注意事项

- 由于算法设计，LocalCAC矫正强度会受到Demosaic结果影响。因此将LocalCAC强度设为最小时依然会保留一定去紫边能力，对高光处紫边表现有一定影响。如果想完全关闭LocalCAC效果需直接将使能开关enable置为HI_FALSE。
- 在紫边很宽的时候，若cb_cr_ratio的值未达到完全检测出紫边，会导致部分紫边去除，部分紫边残留而形成锯齿。
- 有一些紫边比较严重的场景，R或者B容易出现饱和的情况，这时，b_detect_threshold或者r_detect_threshold在最大值附近有去除紫边不平滑的现象。需要跟de_purple_cr_str和 de_purple_cb_str联合调整。

3.15.5.4 hi_isp_global_cac_attr

说明

定义色差校正属性。

定义

```
typedef struct {  
    hi_bool enable;  
    hi_u16 ver_coordinate;  
    hi_u16 hor_coordinate;  
    hi_s16 param_red_a;  
    hi_s16 param_red_b;  
    hi_s16 param_red_c;  
    hi_s16 param_blue_a;  
    hi_s16 param_blue_b;  
    hi_s16 param_blue_c;  
    hi_u8 ver_norm_shift;  
    hi_u8 ver_norm_factor;  
    hi_u8 hor_norm_shift;  
    hi_u8 hor_norm_factor;  
    hi_u16 cor_var_thr;  
} hi_isp_global_cac_attr;
```

成员

成员名称	描述
enable	色差校正使能。 取值范围： • HI_FALSE：关闭 • HI_TRUE：使能
ver_coordinate	光学中心垂直坐标。 取值范围：[0, 8191]且小于pipe设置的图像高度范围
hor_coordinate	光学中心水平坐标。 取值范围：[0, 8191]且小于pipe设置的图像宽度范围
param_red_a	R通道对应的半径多项式系数a。 取值范围：[-256, 255]

成员名称	描述
param_red_b	R通道对应的半径多项式系数b。 取值范围：[-256, 255]
param_red_c	R通道对应的半径多项式系数c。 取值范围：[-256, 255]
param_blue_a	B通道对应的半径多项式系数a。 取值范围：[-256, 255]
param_blue_b	B通道对应的半径多项式系数b。 取值范围：[-256, 255]
param_blue_c	B通道对应的半径多项式系数c。 取值范围：[-256, 255]
ver_norm_shift	垂直方向的归一化移位参数。 取值范围：[0, 7]
ver_norm_factor	垂直方向的归一化系数。 取值范围：[0, 31]
hor_norm_shift	水平方向的归一化移位参数。 取值范围：[0, 7]
hor_norm_factor	水平方向的归一化系数。 取值范围：[0, 31]
cor_var_thr	横向色差校正的方差阈值。 取值范围：[0, 4095]

3.16 CA

3.16.1 hi_mpi_isp_get_ca_attr

函数功能

获取CA属性。

约束说明

HDR模式，默认关闭CA，客户可以自行设置打开。

函数原型

[hi_s32](#) hi_mpi_isp_get_ca_attr([hi_vi_pipe](#) vi_pipe, [hi_isp_ca_attr](#) *ca_attr)

参数说明

参数名	输入/输出	说明
vi_pipe	输入	VI PIPE号。 取值范围：[0, 12)。
ca_attr	输入	CA 属性。

返回值说明

- 0：成功
- 非0：失败，参见[ISP返回码](#)

3.16.2 hi_mpi_isp_set_ca_attr

函数功能

设置CA属性。

函数原型

[hi_s32](#) hi_mpi_isp_set_ca_attr([hi_vi_pipe](#) vi_pipe, const [hi_isp_ca_attr](#) *ca_attr)

参数说明

参数名	输入/输出	说明
vi_pipe	输入	VI PIPE号。 取值范围：[0, 12)。
ca_attr	输入	CA 属性。

返回值说明

- 0：成功
- 非0：失败，参见[ISP返回码](#)

3.16.3 数据类型

3.16.3.1 hi_isp_ca_lut

说明

定义CA模式的属性。

定义

```
typedef struct {  
    hi_u32 y_ratio_lut[HI_ISP_CA_YRATIO_LUT_LENGTH];  
    hi_s32 iso_ratio[HI_ISP_AUTO_ISO_NUM];  
} hi_isp_ca_lut;
```

成员

成员名称	描述
y_ratio_lut	CA模式，根据亮度Y查找的调整色度UV的增益。意思是将亮度划分成256等分，对应的设定256个调整系数A，用于调整UV的值。建议调整的时候Y值越小（也就是比较暗的地方）对应的增益A越小，这样可以有效的抑制暗区的色噪。亮度的增益设置大一些，亮区的颜色会鲜艳一些。 取值范围：[0, 2047] #define HI_ISP_CA_YRATIO_LUT_LENGTH 128
iso_ratio[HI_ISP_AUTO_ISO_NUM]	CA模式，根据ISO值查找的调整色度UV的增益B。这是一个全局的增益，也就是ISO固定，图像所有像素点的UV的调整增益都是B。建议在低ISO的时候对应的增益B可以设置大一些，高ISO（低照度）对应的增益B可以设置小一些，可以降低低照度下的暗处色噪的目的。 取值范围：[0, 2047] #define HI_ISP_AUTO_ISO_NUM 16

3.16.3.2 hi_isp_ca_attr

说明

定义颜色调整模块的属性。

定义

```
typedef struct {  
    hi_bool enable;  
    hi_isp_ca_lut ca;  
} hi_isp_ca_attr;
```

成员

成员名称	描述
enable	色彩调整模块的使能。 取值范围： HI_FALSE：关闭 HI_TRUE：使能
ca	CA模式的属性。

3.17 去 FPN

3.17.1 hi_mpi_isp_set_fpn_attr

函数功能

设置去FPN属性。

函数原型

```
hi_s32 hi_mpi_isp_set_fpn_attr(hi_vi_pipe vi_pipe, const hi_isp_fpn_attr *fpn_attr)
```

约束说明

- 不支持分块模式下校正。
- 在WDR场景下不支持FPN校正。
- FPN校正不支持压缩，只能是非压缩模式。
- 校正用的黑帧最小分辨率为120*120，最大分辨率为4096*4096。
- 在校正时，用户需要输入标定时保存的黑帧信息、OFFSET、ISO。
- 在校正时，需要调用hi_mpi_vi_pipe_get_buffer申请内存，申请的内存长度不得小于标定时保存的黑帧长度。

参数说明

参数名	输入/输出	说明
vi_pipe	输入	VI PIPE号。 取值范围：[0, 12)。
fpn_attr	输入	去FPN属性指针。

返回值说明

- 0：成功
- 非0：失败，参见[ISP返回码](#)

3.17.2 hi_mpi_isp_get_fpn_attr

函数功能

获取去FPN属性。

函数原型

```
hi_s32 hi_mpi_isp_get_fpn_attr(hi_vi_pipe vi_pipe, hi_isp_fpn_attr *fpn_attr)
```

约束说明

无

参数说明

参数名	输入/输出	说明
vi_pipe	输入	VI PIPE号。 取值范围：[0, 12)。
fpn_attr	输入	去FPN属性指针。

返回值说明

- 0：成功
- 非0：失败，参见[ISP返回码](#)

3.17.3 hi_mpi_isp_fpn_calibrate

函数功能

设置去FPN标定属性。

函数原型

hi_s32 hi_mpi_isp_fpn_calibrate(**hi_vi_pipe** vi_pipe, **hi_isp_fpn_calibrate_attr** *calibrate_attr)

约束说明

- 该接口一般用于调试场景下标定FPN算法的参数，商用场景下不建议使用，可能会引起内存异常、图片效果异常等问题。
- 标定时用的内存注意要调用acldtAllocBuf接口，按16bit进行分配，需要保证传入的黑帧地址有效。标定完成之后要把黑帧，ISO，黑帧的长度、OFFSET、是否压缩标志等保存到外部存储介质，在校正的时候要用到这些信息。
acldtAllocBuf接口的说明请参见《[AscendCL应用软件开发指南 \(C&C++\)](#)》的"AscendCL API参考>共享Buffer管理"章节。
- 不支持分块模式下标定。
- FPN标定不支持压缩，只能是非压缩模式。
- 在WDR场景下，不支持开启FPN标定。
- VI物理通道使能时，不支持开启FPN标定。
- 在调用接口禁用VI物理通道前，建议适当延时保证任务下发一次。
- 只支持单PIPE的FPN标定，且单个PIPE的FPN标定和校正不能同时进行。
- 标定的黑帧最小分辨率为120*120，最大分辨率为4096*4096。
- 当VI模块的PIPE有销毁操作时（例如：分辨率切换、WDR模式切换等情况），保存在VI模块的FPN属性会丢失，FPN功能需要在VI模块的PIPE创建后重新配置才能恢复。

参数说明

参数名	输入/输出	说明
vi_pipe	输入	VI PIPE号。 取值范围：[0, 12)。
calibrate_attr	输入	去FPN标定属性指针。

返回值说明

- 0：成功
- 非0：失败，参见[ISP返回码](#)

3.17.4 数据类型

3.17.4.1 hi_isp_fpn_frame_info

说明

定义去FPN标定的黑帧信息。

定义

```
typedef struct
{
    hi_u32      iso;
    hi_u32      offset[HI_ISP_STRIPING_MAX_NUM];
    hi_u32      frm_size;
    hi_video_frame_info fpn_frame;
} hi_isp_fpn_frame_info;
```

成员

成员名称	描述
iso	标定时是输出，校正时是输入。 ISO值必须大于0。
offset[HI_ISP_STRIPING_MAX_NUM]	黑帧所有像素的平均值，来自于标定时的OFFSET。 取值范围：[0, 0xFFF]。 <code>#define HI_ISP_STRIPING_MAX_NUM 5</code>
frm_size	帧的大小，来自于标定时保存的黑帧大小。
fpn_frame	黑帧帧信息，首先分配内存，然后把标定时的黑帧从外存读入此内存，同时对此结构体成员进行赋值。

注意事项

通常情况下offset比较接近sensor的黑电平，若差别较大，可能会造成图像偏色，需要排查原因，offset可以在“/proc/umap/isp”文件中查询。

3.17.4.2 hi_isp_fpn_calibrate_attr

说明

定义去FPN的标定属性。

定义

```
typedef struct {  
    hi_u32 threshold;  
    hi_u32 frame_num;  
    hi_isp_fpn_type fpn_type;  
    hi_isp_fpn_frame_info fpn_cali_frame;  
} hi_isp_fpn_calibrate_attr;
```

成员

成员名称	描述
threshold	标定时的阈值，在标定时如果像素的值大于该值，则认为是坏点，该像素点不参与标定。 取值范围：[0x1, 0xFFF]
frame_num	标定的帧数。 取值范围为{1, 2, 4, 8, 16}，即为2的整数次幂。
fpn_type	标定的类型，当前只支持帧模式。
fpn_cali_frame	标定出来的黑帧信息。

注意事项

threshold与标定时保存黑帧的比特数相关，假设标定时保存的比特数是N，则阈值一般取 $(2^N) >> 2$ 。

3.17.4.3 hi_isp_fpn_attr

说明

定义去FPN的校正属性。

定义

```
typedef struct {  
    hi_bool enable;  
    hi_op_mode op_type;  
    hi_isp_fpn_type fpn_type;  
    hi_isp_fpn_frame_info fpn_frm_info;
```

```
hi_isp_fpn_manual_attr manual_attr;  
hi_isp_fpn_auto_attr auto_attr;  
} hi_isp_fpn_attr;
```

成员

成员名称	描述
enable	校正使能。 取值范围：HI_FALSE：关闭， HI_TRUE：使能。
op_type	校正模式，包括手动模式与自动模式。手动模式需要用户设置校正强度，而自动模式不需要设置校正强度，系统会根据当前图像的ISO计算出校正强度，进行动态校正。 取值范围： HI_OP_MODE_AUTO：自动 HI_OP_MODE_MANUAL：手动
fpn_type	校正类型，当前只支持帧模式。
fpn_frm_info	黑帧帧信息。
manual_attr	手动校正属性。
auto_attr	自动校正属性。

3.17.4.4 hi_isp_fpn_manual_attr

说明

定义去FPN在手动模式下的校正属性。

定义

```
typedef struct {  
    hi_u32 strength;  
} hi_isp_fpn_manual_attr;
```

成员

成员名称	描述
strength	手动模式时校正强度，1倍强度为256。 取值范围：[0, 1023]

3.17.4.5 hi_isp_fpn_auto_attr

说明

定义去FPN在自动模式下的校正属性。

定义

```
typedef struct {  
    hi_u32 strength;  
} hi_isp_fpn_auto_attr;
```

成员

成员名称	描述
strength	自动模式实时校正强度，只读属性（ISP Firmware根据ISO自动调整）。strength根据标定时的ISO与当前ISO动态调整，当前ISO大于标定时的ISO，strength大于 256；当前ISO小于标定时的ISO，strength小于 256。

3.17.4.6 hi_isp_fpn_type

说明

定义去FPN类型。

定义

```
typedef enum {  
    HI_ISP_FPN_TYPE_FRAME = 0,  
    HI_ISP_FPN_TYPE_BUTT  
} hi_isp_fpn_type;
```

成员

成员名称	描述
HI_ISP_FPN_TYPE_FRAME	帧模式校正。

3.18 黑电平

3.18.1 hi_mpi_isp_get_black_level_attr

函数功能

获取黑电平属性。

函数原型

```
hi_s32 hi_mpi_isp_get_black_level_attr(hi_vi_pipe vi_pipe,  
    hi_isp_black_level_attr *black_level)
```

参数说明

参数名	输入/输出	说明
vi_pipe	输入	VI PIPE号。 取值范围：[0, 12)。
black_level	输入	黑电平属性。

返回值说明

- 0：成功
- 非0：失败，参见[ISP返回码](#)

3.18.2 hi_mpi_isp_set_black_level_attr

函数功能

设置黑电平属性。

函数原型

[hi_s32](#) hi_mpi_isp_set_black_level_attr([hi_vi_pipe](#) vi_pipe, const [hi_isp_black_level_attr](#) *black_level)

约束说明

- 如果接口配置为自动模式：若cmos.c里面的ISP_CMOS_BLACK_LEVEL_S结构体成员bUpdate设置为HI_TRUE，则表示始终使用cmos.c内联动的黑电平配置，而bUpdate设置为HI_FALSE，则表示使用cmos.c非联动的黑电平配置；如果接口配置为手动模式：此时接口配置的黑电平配置会生效，ISP内部采用接口配置值。
- 线性模式下，black level模块集成在isp_dgain模块中，所以bypass isp_dgain会同时关闭isp_dgain模块中的black level功能。对图像质量影响较大。
- WDR模式下，black level模块集成在isp_dgain及WDR模块中，bypass isp_dgain也会对图像质量有部分影响。

参数说明

参数名	输入/输出	说明
vi_pipe	输入	VI PIPE号。 取值范围：[0, 12)。
black_level	输入	黑电平属性。

返回值说明

- 0：成功

- 非0：失败，参见[ISP返回码](#)

3.18.3 数据类型

3.18.3.1 hi_isp_black_level_attr

说明

定义ISP黑电平属性。

定义

```
typedef struct {  
    hi\_op\_mode op_type;  
    hi\_u16 black_level[HI_ISP_BAYER_CHN_NUM];  
} hi_isp_black_level_attr;
```

成员

成员名称	描述
op_type	黑电平工作类型。 • HI_OP_MODE_AUTO：自动 • HI_OP_MODE_MANUAL：手动 默认：HI_OP_MODE_AUTO模式。
black_level	黑电平的值，分别表示R、Gr、Gb、B分量的黑电平 取值范围：[0x0, 0xFFFF] <code>#define HI_ISP_BAYER_CHN_NUM 4</code>

3.19 Demosaic

3.19.1 hi_mpi_isp_get_demosaic_attr

函数功能

获取Demosaic参数。

函数原型

```
hi\_s32 hi_mpi_isp_get_demosaic_attr(hi\_vi\_pipe vi_pipe, hi\_isp\_demosaic\_attr  
*demosaic_attr)
```

参数说明

参数名	输入/输出	说明
vi_pipe	输入	VI PIPE号。 取值范围：[0, 12)。
demosaic_attr	输入	Demosaic参数。

返回值说明

- 0：成功
- 非0：失败，参见[ISP返回码](#)

3.19.2 hi_mpi_isp_set_demosaic_attr

函数功能

设置Demosaic参数。

函数原型

[hi_s32](#) hi_mpi_isp_set_demosaic_attr([hi_vi_pipe](#) vi_pipe, const [hi_isp_demosaic_attr](#) *demosaic_attr)

参数说明

参数名	输入/输出	说明
vi_pipe	输入	VI PIPE号。 取值范围：[0, 12)。
demosaic_attr	输入	Demosaic参数。

返回值说明

- 0：成功
- 非0：失败，参见[ISP返回码](#)

3.19.3 数据类型

3.19.3.1 hi_isp_demosaic_manual_attr

说明

定义ISP DEMOSAIC手动属性。

定义

```
typedef struct {  
    hi_u8  nddm_strength;  
    hi_u8  nddm_mf_detail_strength;  
    hi_u8  nddm_hf_detail_strength;  
    hi_u8  detail_smooth_range;  
  
    hi_u8  color_noise_f_threshold;  
    hi_u8  color_noise_f_strength;  
    hi_u8  color_noise_y_threshold;  
    hi_u8  color_noise_y_strength;  
} hi_isp_demosaic_manual_attr;
```

成员

成员名称	描述
nddm_strength	无方向插值强度，值越大，无方向插值所占比重越大。 取值范围：[0x0,0xFF]。
nddm_mf_detail_strength	无方向中频纹理增强强度，值越大，无方向中频纹理细节增强越强，对噪声同样有增强。 取值范围：[0x0, 0x7F]。
nddm_hf_detail_strength	无方向高频纹理增强强度，值越大，无方向高频纹理细节增强越强，对噪声均匀性有提升。 取值范围：[0x0,0x10]。
detail_smooth_range	细节平滑范围，值越大，做平滑处理的细节范围越大，能够抑制更多伪细节。 取值范围：[0x1, 0x7]。
color_noise_f_threshold	根据画面高频程度降低颜色噪声，值越大，则越容易在平坦区域进行去色噪功能。值越小，则影响的像素越少。 取值范围为[0x0, 0xFF]。
color_noise_f_strength	根据画面高频程度降低颜色噪声的强度，值越大，则降饱和度强度越高。 取值范围为[0x0, 0x8]。
color_noise_y_threshold	根据亮度和饱和度降低颜色噪声，值越大，对暗处影响越大，对饱和度高的像素影响越大。值越小，影响的像素越少。 取值范围：[0x0, 0xFF]。
color_noise_y_strength	根据亮度和饱和度降低颜色噪声的强度，值越大，对影响的像素降饱和度越多，值越小，降饱和度程度越少。 取值范围：[0x0, 0xFF]。

注意事项

- 在红外模式下，或者在CCM饱和度设定成0，输出灰度图的情况下，需要将color_noise_y_threshold和color_noise_y_strength参数都设定成0，关闭降低颜色噪声的功能。

- color_noise_y_threshold和color_noise_y_strength参数受降低颜色噪声的功能影响 会导致CCM饱和度设定成0时图像不会完全灰色，调节CCM饱和度时，图像饱和度不是线性变化。
- color_noise_f_threshold由于算法采取的是硬钳位方法进行判断是否需要降低饱和度，因此如果这个参数设置的过大，在有一定亮噪的情况下，会出现颜色过渡异常的情况。

3.19.3.2 hi_isp_demosaic_auto_attr

说明

定义ISP DEMOSAIC自动属性。

定义

```
typedef struct {  
    hi_u8 nddm_strength[HI_ISP_AUTO_ISO_NUM];  
    hi_u8 nddm_mf_detail_strength[HI_ISP_AUTO_ISO_NUM];  
    hi_u8 nddm_hf_detail_strength[HI_ISP_AUTO_ISO_NUM];  
    hi_u8 detail_smooth_range[HI_ISP_AUTO_ISO_NUM];  
    hi_u8 color_noise_f_threshold[HI_ISP_AUTO_ISO_NUM];  
    hi_u8 color_noise_f_strength[HI_ISP_AUTO_ISO_NUM];  
    hi_u8 color_noise_y_threshold[HI_ISP_AUTO_ISO_NUM];  
    hi_u8 color_noise_y_strength[HI_ISP_AUTO_ISO_NUM];  
} hi_isp_demosaic_auto_attr;
```

成员

成员名称	描述
nddm_strength[HI_ISP_AUTO_ISO_NUM]	无方向插值强度，值越大，无方向插值所占比重越大。 取值范围：[0x0,0xFF]。 #define HI_ISP_AUTO_ISO_NUM 16
nddm_mf_detail_strength[HI_ISP_AUTO_ISO_NUM]	无方向中频纹理增强强度，值越大，无方向中频纹理细节增强越强，对噪声同样有增强。 取值范围：[0x0, 0x7F]。
nddm_hf_detail_strength[HI_ISP_AUTO_ISO_NUM]	无方向高频纹理增强强度，值越大，无方向高频纹理细节增强越强，对噪声均匀性有提升。 取值范围：[0x0,0x10]。
detail_smooth_range[HI_ISP_AUTO_ISO_NUM]	细节平滑范围，值越大，做平滑处理的细节范围越大，能够抑制更多伪细节。 取值范围：[0x1, 0x7]。
color_noise_f_threshold[HI_ISP_AUTO_ISO_NUM]	根据画面高频程度降低颜色噪声，值越大，则越容易在平坦区域进行去色噪功能。值越小，则影响的像素越少。 取值范围：[0x0, 0xFF]。
color_noise_f_strength[HI_ISP_AUTO_ISO_NUM]	根据画面高频程度降低颜色噪声的强度，值越大，则降低饱和度强度越高。 取值范围：[0x0, 0x8]。

成员名称	描述
color_noise_y_threshold[HI_ISP_AUTO_ISO_NUM]	根据亮度和饱和度降低颜色噪声，值越大，对暗处影响越大，对饱和度高的像素影响越大。值越小，影响的像素越少。 取值范围：[0x0, 0xFF]。
color_noise_y_strength[HI_ISP_AUTO_ISO_NUM]	根据亮度和饱和度降低颜色噪声的强度，值越大，对影响的像素降饱和度越多，值越小，降饱和度程度越少。 取值范围：[0x0, 0xFF]。

3.19.3.3 hi_isp_demosaic_attr

说明

定义ISP DEMOSAIC属性。

定义

```
typedef struct {  
    hi_bool enable;  
    hi_op_mode op_type;  
    hi_isp_demosaic_manual_attr manual_attr;  
    hi_isp_demosaic_auto_attr auto_attr;  
} hi_isp_demosaic_attr;
```

成员

成员名称	描述
enable	是否使能Demosaic模块。取值如下。 取值范围：HI_FALSE：关闭， HI_TRUE：使能。
op_type	Demosaic工作类型。取值如下。 - HI_OP_MODE_AUTO：自动 - HI_OP_MODE_MANUAL：手动
manual_attr	Demosaic手动参数。
auto_attr	Demosaic自动参数。

3.20 Anti FalseColor

3.20.1 hi_mpi_isp_get_anti_false_color_attr

函数功能

获取Anti FalseColor参数。

函数原型

[hi_s32](#) hi_mpi_isp_get_anti_false_color_attr([hi_vi_pipe](#) vi_pipe,
hi_isp_anti_false_color_attr *anti_false_color)

参数说明

参数名	输入/输出	说明
vi_pipe	输入	VI PIPE号。 取值范围：[0, 12)。
anti_false_color	输入	Anti FalseColor参数

返回值说明

- 0：成功
- 非0：失败，参见[ISP返回码](#)

3.20.2 hi_mpi_isp_set_anti_false_color_attr

函数功能

设置Anti FalseColor参数。

函数原型

[hi_s32](#) hi_mpi_isp_set_anti_false_color_attr([hi_vi_pipe](#) vi_pipe, const
hi_isp_anti_false_color_attr *anti_false_color)

参数说明

参数名	输入/输出	说明
vi_pipe	输入	VI PIPE号。 取值范围：[0, 12)。
anti_false_color	输入	Anti FalseColor参数

返回值说明

- 0：成功
- 非0：失败，参见[ISP返回码](#)

3.20.3 数据类型

3.20.3.1 hi_isp_anti_false_color_manual_attr

说明

定义ISP Anti FalseColor手动属性。

定义

```
typedef struct {  
    hi_u8 threshold;  
    hi_u8 strength;  
} hi_isp_anti_false_color_manual_attr;
```

成员

成员名称	描述
threshold	去伪彩阈值，值越大，伪彩去除的区域越多。 取值范围：[0, 0x20]。
strength	去伪彩强度，值越大，去伪彩强度越大，伪彩现象消除越好，但可能导致正常颜色的改变。 取值范围：[0, 0x1F]。

3.20.3.2 hi_isp_anti_false_color_auto_attr

说明

定义ISP Anti FalseColor自动属性。

定义

```
typedef struct {  
    hi_u8 threshold[HI_ISP_AUTO_ISO_NUM];  
    hi_u8 strength[HI_ISP_AUTO_ISO_NUM];  
} hi_isp_anti_false_color_auto_attr;
```

成员

成员名称	描述
threshold[HI_ISP_AUTO_ISO_NUM]	去伪彩阈值，值越大，伪彩去除的区域越多。 取值范围：[0, 0x20]。 #define HI_ISP_AUTO_ISO_NUM 16
strength	去伪彩强度，值越大，去伪彩强度越大，伪彩现象消除越好，但可能导致正常颜色的改变。 取值范围：[0, 0x1F]。

3.20.3.3 hi_isp_anti_false_color_attr

说明

定义Anti FalseColor属性。

定义

```
typedef struct {  
    hi_bool enable;  
    hi_op_mode op_type;  
    hi_isp_anti_false_color_auto_attr auto_attr;  
    hi_isp_anti_false_color_manual_attr manual_attr;  
} hi_isp_anti_false_color_attr;
```

成员

成员名称	描述
enable	是否使能Anti FalseColor模块。 取值范围：HI_FALSE：关闭， HI_TRUE：使能。
op_type	Anti FalseColor工作类型。 - HI_OP_MODE_AUTO：自动 - HI_OP_MODE_MANUAL：手动
auto_attr	Anti FalseColor自动参数。
manual_attr	Anti FalseColor手动参数。

注意事项

- Anti FalseColor功能分为自动和手动参数：
- enable为HI_TRUE，op_type为HI_OP_MODE_AUTO，使用自动Anti FalseColor功能。
 - enable设置为HI_TRUE，op_type设置为HI_OP_MODE_MANUAL使用手动Anti FalseColor功能。

3.21 NR

3.21.1 hi_mpi_isp_get_nr_attr

函数功能

获取BNR参数。

函数原型

```
hi_s32 hi_mpi_isp_get_nr_attr(hi_vi_pipe vi_pipe, hi_isp_nr_attr *nr_attr)
```


参数说明

参数名	输入/输出	说明
vi_pipe	输入	VI PIPE号。 取值范围：[0, 12)。
nr_attr	输入	噪声抑制参数。

返回值说明

- 0：成功
- 非0：失败，参见[ISP返回码](#)

3.21.2 hi_mpi_isp_set_nr_attr

函数功能

设置BNR参数。

函数原型

[hi_s32](#) hi_mpi_isp_set_nr_attr([hi_vi_pipe](#) vi_pipe, const [hi_isp_nr_attr](#) *nr_attr)

参数说明

参数名	输入/输出	说明
vi_pipe	输入	VI PIPE号。 取值范围：[0, 12)。
nr_attr	输入	噪声抑制参数。

返回值说明

- 0：成功
- 非0：失败，参见[ISP返回码](#)

3.21.3 数据类型

3.21.3.1 hi_isp_nr_manual_attr

说明

定义ISP NR手动属性。

定义

```
typedef struct {  
    hi_u8  chroma_strength[HI_ISP_BAYER_CHN_NUM];  
    hi_u8  fine_strength;  
    hi_u16 coring_wgt;  
    hi_u16 coarse_strength[HI_ISP_BAYER_CHN_NUM];  
} hi_isp_nr_manual_attr;
```

成员

成员名称	描述
chroma_strength	色噪去除强度选择，值越大，去色噪强度越大。四通道分别可调。 取值范围：[0, 0x3]。 #define HI_ISP_BAYER_CHN_NUM 4
fine_strength	亮噪去除强度，值越大，亮噪去除强度越大。 取值范围：[0x0, 0x80]。
coring_wgt	随机噪声保留强度，值越大，保留的随机噪声越多。 取值范围：[0x0, 0xC80]。
coarse_strength	控制亮度噪声整体去噪强度，四通道强度独立可调，值越大，亮噪去噪强度越大。 取值范围：[0x0, 0x360]。

注意事项

低照度下，参数coarse_strength与fine_strength同时调到最大值时，容易出现由于去噪强度过大导致的边缘虚化，建议根据效果调到合适取值。

3.21.3.2 hi_isp_nr_auto_attr

说明

定义ISP NR自动属性。

定义

```
typedef struct {  
    hi_u8  chroma_strength[HI_ISP_BAYER_CHN_NUM][HI_ISP_AUTO_ISO_NUM];  
    hi_u8  fine_strength[HI_ISP_AUTO_ISO_NUM];  
    hi_u16 coring_wgt[HI_ISP_AUTO_ISO_NUM];  
    hi_u16 coarse_strength[HI_ISP_BAYER_CHN_NUM][HI_ISP_AUTO_ISO_NUM];  
} hi_isp_nr_auto_attr;
```

成员

成员名称	描述
chroma_strength	色噪去除强度选择，值越大，去色噪强度越大。四通道分别可调。 取值范围：[0x0, 0x3]。 #define HI_ISP_BAYER_CHN_NUM 4 #define HI_ISP_AUTO_ISO_NUM 16
fine_strength	亮噪去除强度，值越大，亮噪去除强度越大。 取值范围：[0x0, 0x80]。
coring_wgt	随机噪声保留强度，值越大，保留的随机噪声越多。 取值范围：[0x0, 0xC80]。
coarse_strength	控制亮噪整体去噪强度，四通道强度独立可调，值越大，亮噪去噪强度越大。 取值范围：[0x0, 0x360]。

注意事项

低照度下，参数coarse_strength与fine_strength同时调到最大值时，容易出现由于去噪强度过大导致的边缘虚化，建议根据效果调到合适取值。

3.21.3.3 hi_isp_nr_wdr_attr

说明

定义ISP NR在WDR模式下属性，与自动/手动模式并行。

定义

```
typedef struct {  
    hi_u8 wdr_frame_strength[HI_ISP_WDR_MAX_FRAME_NUM];  
    hi_u8 fusion_frame_strength[HI_ISP_WDR_MAX_FRAME_NUM];  
} hi_isp_nr_wdr_attr;
```

成员

成员名称	描述
wdr_frame_strength[HI_ISP_WDR_MAX_FRAME_NUM]	WDR模式下每一合成帧分别对应的全局亮度去噪强度。值越大，去噪强度越大，值越小，去噪强度越弱。 取值范围：[0x0, 0x50]。 #define HI_ISP_WDR_MAX_FRAME_NUM 4
fusion_frame_strength[HI_ISP_WDR_MAX_FRAME_NUM]	Fusion模式下每一融合帧分别对应的全局亮度去噪强度。 取值范围：[0x0, 0x50]。

3.21.3.4 hi_isp_nr_attr

说明

定义 ISP NR属性。

定义

```
typedef struct {  
    hi_bool enable;  
  
    hi_bool nr_lsc_enable;  
  
    hi_u8 bnr_lsc_max_gain;  
    hi_u16 bnr_lsc_cmp_strength;  
    hi_u16 coring_ratio[HI_ISP_BAYERNR_LUT_LENGTH];  
    hi_op_mode op_type;  
    hi_isp_nr_auto_attr auto_attr;  
    hi_isp_nr_manual_attr manual_attr;  
    hi_isp_nr_wdr_attr wdr_attr;  
} hi_isp_nr_attr;
```

成员

成员名称	描述
enable	是否使能NR模块 取值范围： - HI_FALSE：关闭 - HI_TRUE：使能
nr_lsc_enable	是否使能NR参考LSC增益功能。NR参考LSC去噪功能需要在LSC模块打开下调试，当LSC模块本身关闭时，NR参考LSC不生效。 取值范围： - HI_FALSE：关闭 - HI_TRUE：使能
bnr_lsc_max_gain	参考LSC增益最大倍数。 取值范围[0x0, 0xBF]。
bnr_lsc_cmp_strength	参考LSC增益强度。 取值范围[0x0, 0x100]。
coring_ratio[HI_ISP_BAYERNR_LUT_LENGTH]	噪声保留强度，根据亮度在coring_ratio数组里查找对应的噪声保留强度，值越大，亮度噪声保留越多，值越小，亮度噪声保留越少。 取值范围[0x0, 0x3FF]。 #define HI_ISP_BAYERNR_LUT_LENGTH 33
op_type	NR工作类型。 - HI_OP_MODE_AUTO：自动 - HI_OP_MODE_MANUAL：手动

成员名称	描述
auto_attr	NR自动参数。
manual_attr	NR手动参数。
wdr_attr	NR WDR模式下参数。 当工作在WDR模式下，NR整体去噪强度参数wdr_attr生效。

3.22 Crosstalk Removal

3.22.1 hi_mpi_isp_get_crosstalk_attr

函数功能

获取Crosstalk属性。

函数原型

[hi_s32](#) hi_mpi_isp_get_crosstalk_attr([hi_vi_pipe](#) vi_pipe, [hi_isp_cr_attr](#) *cr_attr)

参数说明

参数名	输入/输出	说明
vi_pipe	输入	VI PIPE号。 取值范围：[0, 12)。
cr_attr	输入	Crosstalk属性。

返回值说明

- 0：成功
- 非0：失败，参见[ISP返回码](#)

3.22.2 hi_mpi_isp_set_crosstalk_attr

函数功能

设置Crosstalk属性。

函数原型

[hi_s32](#) hi_mpi_isp_set_crosstalk_attr([hi_vi_pipe](#) vi_pipe, const [hi_isp_cr_attr](#) *cr_attr)

参数说明

参数名	输入/输出	说明
vi_pipe	输入	VI PIPE号。 取值范围：[0, 12)。
cr_attr	输入	Crosstalk属性。

返回值说明

- 0：成功
- 非0：失败，参见[ISP返回码](#)

3.22.3 数据类型

3.22.3.1 hi_isp_cr_attr

说明

定义ISP Crosstalk属性。

定义

```
typedef struct {  
    hi_bool enable;  
    hi_u8 slope;  
    hi_u8 sensi_slope;  
    hi_u16 sensi_threshold;  
    hi_u16 strength[HI_ISP_AUTO_ISO_NUM];  
    hi_u16 threshold[HI_ISP_AUTO_ISO_NUM];  
    hi_u16 np_offset[HI_ISP_AUTO_ISO_NUM];  
} hi_isp_cr_attr;
```

成员

成员名称	描述
enable	是否使能Crosstalk功能。 取值范围：HI_FALSE：关闭， HI_TRUE：使能。
slope	设置Crosstalk门限之上的处理强度。值越大，表示在门限值threshold之上整体处理强度衰减得越慢。 取值范围：[0, 14]。
sensi_slope	设置Crosstalk敏感度值。值越大，表示在门限值sensi_threshold之上，边沿上的处理强度衰减得越慢。 取值范围：[0, 14]。

成员名称	描述
sensi_threshold	设置Crosstalk敏感度门限值。值越大，表示在更强的边沿上也会有绿平衡处理。 取值范围：[0, 16383]。
strength[HI_ISP_AUTO_ISO_NUM]	设置Crosstalk强度值。值越大，处理强度越大。该数组的16个值分别对应的sensor在不同增益情况下不同的设置值。具体请参见表3-6。 取值范围：[0, 0x100]。 #define HI_ISP_AUTO_ISO_NUM 16
threshold[HI_ISP_AUTO_ISO_NUM]	设置Crosstalk门限值。值越大，表示整体处理的强度越大。该数组的16个值分别对应的sensor在不同增益情况下不同的设置值。 取值范围：[0, 16383]。
np_offset[HI_ISP_AUTO_ISO_NUM]	设置Noise Profile值。值越大，表示抗噪能力越大。该数组的16个值分别对应的sensor在不同增益情况下不同的设置值。 取值范围：[512, 16383]。

表 3-6 strength[HI_ISP_AUTO_ISO_NUM]在不同的增益情况下的设置值

strength	a_gain*d_gain*isp_d_gain (times)
strength [0]	100
strength [1]	200
strength [2]	400
strength [3]	800
strength [4]	1600
strength [5]	3200
strength [6]	6400
strength [7]	12800
strength [8]	25600
strength [9]	51200
strength [10]	102400
strength [11]	204800
strength [12]	409600
strength [13]	819200

strength	a_gain*d_gain*isp_d_gain (times)
strength [14]	1638400
strength [15]	3276800

注意事项

sensi_threshold参数变化会影响threshold参数的表现，建议先调节sensi_threshold，再调节threshold。

3.23 Defect Pixel

3.23.1 hi_mpi_isp_get_dp_dynamic_attr

函数功能

获取动态坏点调试属性。

函数原型

hi_s32 hi_mpi_isp_get_dp_dynamic_attr(hi_vi_pipe vi_pipe,
hi_isp_dp_dynamic_attr *dp_dynamic_attr)

参数说明

参数名	输入/输出	说明
vi_pipe	输入	VI PIPE号。 取值范围：[0, 12)。
dp_dynamic_attr	输入	动态坏点属性。

返回值说明

- 0：成功
- 非0：失败，参见ISP返回码

3.23.2 hi_mpi_isp_set_dp_dynamic_attr

函数功能

设置动态坏点调试属性。

函数原型

hi_s32 hi_mpi_isp_set_dp_dynamic_attr(**hi_vi_pipe** vi_pipe, const **hi_isp_dp_dynamic_attr** *dp_dynamic_attr)

参数说明

参数名	输入/输出	说明
vi_pipe	输入	VI PIPE号。 取值范围：[0, 12)。
dp_dynamic_attr	输入	动态坏点属性

返回值说明

- 0：成功
- 非0：失败，参见[ISP返回码](#)

3.23.3 hi_mpi_isp_get_dp_static_attr

函数功能

获取静态坏点属性。

函数原型

hi_s32 hi_mpi_isp_get_dp_static_attr(**hi_vi_pipe** vi_pipe, **hi_isp_dp_static_attr** *dp_static_attr)

约束说明

在设置静态坏点属性中，亮点坏点表中表示亮坏点坐标信息，暗坏点表中则保存的是暗坏点坐标信息；而获取静态坏点属性时，亮点坏点表中保存的是所有坏点的坐标信息。

参数说明

参数名	输入/输出	说明
vi_pipe	输入	VI PIPE号。 取值范围：[0, 12)。
dp_static_attr	输入	静态坏点属性

返回值说明

- 0：成功

- 非0：失败，参见[ISP返回码](#)

3.23.4 hi_mpi_isp_set_dp_static_attr

函数功能

设置静态坏点属性，合并坏点表，将坏点坐标信息与坏点数目写入系统内存。

函数原型

```
hi_s32 hi_mpi_isp_set_dp_static_attr(hi_vi_pipe vi_pipe, const
hi_isp_dp_static_attr *dp_static_attr)
```

约束说明

每一路ISP BE所能支持的最大静态坏点个数为HI_ISP_STATIC_DP_COUNT_NORMAL(HI_ISP_STATIC_DP_COUNT_NORMAL = 2048)，因此DPC模块所能支持的最大静态坏点个数为count = HI_ISP_STATIC_DP_COUNT_NORMAL * (当前业务场景下 ISP BE的分块数目)。则亮点数目bright_count <= count 且 dark_count <= count，若亮点表和暗点表中有same个重复的坏点，则 bright_count + dark_count - same <= count。ISP BE的分块数目参考ISP Proc信息中的“block_num”。

参数说明

参数名	输入/输出	说明
vi_pipe	输入	VI PIPE号。 取值范围：[0, 12)。
dp_static_attr	输入	静态坏点属性

返回值说明

- 0：成功
- 非0：失败，参见[ISP返回码](#)

3.23.5 hi_mpi_isp_get_dp_calibrate

函数功能

获取静态坏点标定结果。

约束说明

该接口一般用于调试场景下标定Defect Pixel算法的参数，商用场景下不建议使用，可能会引起内存异常、图片效果异常等问题。

函数原型

```
hi_s32 hi_mpi_isp_get_dp_calibrate(hi_vi_pipe vi_pipe,
hi_isp_dp_static_calibrate *dp_calibrate)
```

参数说明

参数名	输入/输出	说明
vi_pipe	输入	VI PIPE号。 取值范围：[0, 12)。
dp_calibrate	输入	静态坏点标定参数。

返回值说明

- 0：成功
- 非0：失败，参见[ISP返回码](#)

3.23.6 hi_mpi_isp_set_dp_calibrate

函数功能

设置静态坏点标定参数。

函数原型

```
hi_s32 hi_mpi_isp_set_dp_calibrate(hi_vi_pipe vi_pipe, const
hi_isp_dp_static_calibrate *dp_calibrate)
```

约束说明

- 该接口一般用于调试场景下标定Defect Pixel算法的参数，商用场景下不建议使用，可能会引起内存异常、图片效果异常等问题。
- 该接口为静态坏点标定接口，坏点标定只需启动一次，标定完成之后自动关闭标定使能，恢复为正常的坏点校正模式。亮坏点检测的环境为：使用最小的模拟增益和数字增益，降帧率到5~6fps，使曝光时间为200ms（系统自动完成），遮黑镜头或关闭光圈（需手动操作）；暗坏点检测的环境为：光圈处于正常打开状态，要求平坦背景，最好使用辉度箱固定光源，图像整体平均亮度大约为最大亮度50%，或Bayer格式中B通道亮度为最大亮度20%左右。
- 支持WDR模式下静态坏点标定。WDR模式下默认选择短帧进行静态坏点标定，推荐在标定前，通过hi_mpi_isp_set_module_ctrl接口的bit2_chn_select变量选择长帧，再启动静态坏点标定。一般情况下，长帧的标定即可满足需求。若特殊情况下，长帧标定后仍然有坏点残留，可通过bit2_chn_select变量选择中/短帧再次进行标定，并将多个坏点表合并成一个表。

参数说明

参数名	输入/输出	说明
vi_pipe	输入	VI PIPE号。 取值范围：[0, 12)。
dp_calibrate	输入	静态坏点标定参数。

返回值说明

- 0：成功
- 非0：失败，参见[ISP返回码](#)

3.23.7 数据类型

3.23.7.1 hi_isp_static_dp_type

说明

定义静态坏点标定类型。

定义

```
typedef enum {  
    HI_ISP_STATIC_DP_BRIGHT = 0x0,  
    HI_ISP_STATIC_DP_DARK,  
    HI_ISP_STATIC_DP_BUTT  
} hi_isp_static_dp_type;
```

成员

成员名称	描述
HI_ISP_STATIC_DP_BRIGHT	0x0：选择亮点标定
HI_ISP_STATIC_DP_DARK	0x1：选择暗点标定
HI_ISP_STATIC_DP_BUTT	预留参数。

3.23.7.2 hi_isp_status

说明

定义ISP校正（检测）状态。

定义

```
typedef enum {  
    HI_ISP_STATE_INIT    = 0,  
    HI_ISP_STATE_SUCCESS = 1,
```

```
HI_ISP_STATE_TIMEOUT = 2,  
HI_ISP_STATE_BUTT  
} hi_isp_status;
```

成员

成员名称	描述
HI_ISP_STATE_INIT	0：初始状态，未标定。
HI_ISP_STATE_SUCCESS	1：静态坏点标定成功结束。
HI_ISP_STATE_TIMEOUT	2：静态坏点标定超时结束。
HI_ISP_STATE_BUTT	预留参数。

3.23.7.3 hi_isp_dp_static_calibrate

说明

定义静态坏点标定的属性。

定义

```
typedef struct {  
    hi_bool enable_detect;  
    hi_isp_static_dp_type static_dp_type;  
    hi_u8 start_thresh;  
    hi_u16 count_max;  
    hi_u16 count_min;  
    hi_u16 time_limit;  
    hi_u32 table[HI_ISP_STATIC_DP_COUNT_MAX];  
    hi_u8 finish_thresh;  
    hi_u16 count;  
    hi_isp_status status;  
} hi_isp_dp_static_calibrate;
```

成员

成员名称	描述
enable_detect	是否使能静态坏点标定。取值如下。 - HI_FALSE：否 - HI_TRUE：是
static_dp_type	静态坏点标定类型。取值如下。 - HI_ISP_STATIC_DP_BRIGHT：亮点 - HI_ISP_STATIC_DP_DARK：暗点
start_thresh	静态坏点标定开始时的检测门限值，该值的设置与sensor相关。取值范围：[1, 0xFF]。

成员名称	描述
count_max	允许静态坏点的最大个数。 取值范围与vi pipe设置的宽度有关。 宽度[120, 3840]：取值范围 (0, HI_ISP_STATIC_DP_COUNT_NORMAL]。 宽度[3841, 7680]：取值范围 (0, 2 * HI_ISP_STATIC_DP_COUNT_NORMAL]。 宽度[7681, 11520]：取值范围 (0, 3 * HI_ISP_STATIC_DP_COUNT_NORMAL]。 宽度[11521, 15360]：取值范围 (0, 4 * HI_ISP_STATIC_DP_COUNT_NORMAL]。 宽度[15361, 16384]：取值范围 (0, 5 * HI_ISP_STATIC_DP_COUNT_NORMAL]。 #define HI_ISP_STATIC_DP_COUNT_NORMAL 2048
count_min	允许静态坏点的最小个数。 取值范围：[0, count_max)。
time_limit	允许标定超时门限值。 取值范围：[0, 0x640]。
table[HI_ISP_STATIC_DP_COUNT_MAX]	只读，亮暗坏点坐标值查找表，低29bit有效，[12:0]bit为坏点的水平坐标，[28:16]bit为坏点的垂直坐标。 每个坐标取值范围：[0, 0x1FFF1FFF]。 #define HI_ISP_STATIC_DP_COUNT_NORMAL 2048 #define HI_ISP_STRIPING_MAX_NUM 5 #define HI_ISP_STATIC_DP_COUNT_MAX (HI_ISP_STATIC_DP_COUNT_NORMAL * HI_ISP_STRIPING_MAX_NUM)
finish_thresh	只读，静态坏点标定结束时的检测门限值。 取值范围：[0, 0xFF]。
count	只读，标定出的静态坏点的个数。 取值范围：[0, HI_ISP_STATIC_DP_COUNT_MAX]。
status	只读，静态坏点标定结果状态信息。 取值范围： - HI_ISP_STATE_INIT：初始状态，未标定。 - HI_ISP_STATE_SUCCESS：静态坏点标定成功结束。 - HI_ISP_STATE_TIMEOUT：静态坏点标定超时结束。

注意事项

- 坏点检测算法检测成功的标准：检测出的坏点数量是否在[count_min, count_max)。所以不同类型的sensor在做坏点检测时需微调这两个值。
- 系统支持的坏点个数上限是HI_ISP_STATIC_DP_COUNT_MAX。当sensor坏点个数大于HI_ISP_STATIC_DP_COUNT_MAX时, 系统反馈坏点个数等于HI_ISP_STATIC_DP_COUNT_MAX。因此，坏点个数等于HI_ISP_STATIC_DP_COUNT_MAX不能反映当前实际坏点检况。

- finish_thresh只作为输出。针对同类型的sensor，参考finish_thresh值，设置合理的start_thresh，能加快静态坏点校正过程。
- 每次调用标定接口hi_mpi_isp_set_dp_calibrate时，算法内部会自动将status恢复为HI_ISP_STATE_INIT状态。
- 标定亮坏点时，推荐降帧，亮坏点超时时间近似等于time_limit/8 秒。标定暗坏点时，不做降帧动作，暗坏点超时时间近似等于time_limit/48 秒。

3.23.7.4 hi_isp_dp_static_attr

说明

定义静态坏点校正属性。

定义

```
typedef struct {  
    hi_bool enable;  
    hi_u16  bright_count;  
    hi_u16  dark_count;  
    hi_u32  bright_table[HI_ISP_STATIC_DP_COUNT_MAX];  
    hi_u32  dark_table[HI_ISP_STATIC_DP_COUNT_MAX];  
    hi_bool show;  
} hi_isp_dp_static_attr;
```

成员

成员名称	描述
enable	是否使能静态坏点校正。 • HI_FALSE： 否 • HI_TRUE： 是
bright_count	当该变量作为输入时，表示亮坏点的个数；该变量作为输出时，表示总的坏点个数。 取值范围与vi pipe设置的宽度有关。 宽度[120, 3840]：取值范围 [0, HI_ISP_STATIC_DP_COUNT_NORMAL]。 宽度[3841, 7680]：取值范围 [0, 2 * HI_ISP_STATIC_DP_COUNT_NORMAL]。 宽度[7681, 11520]：取值范围 [0, 3 * HI_ISP_STATIC_DP_COUNT_NORMAL]。 宽度[11521, 15360]：取值范围 [0, 4 * HI_ISP_STATIC_DP_COUNT_NORMAL]。 宽度[15361, 16384]：取值范围 [0, 5 * HI_ISP_STATIC_DP_COUNT_NORMAL]。 #define HI_ISP_STATIC_DP_COUNT_NORMAL 2048

成员名称	描述
dark_count	当该变量作为输入时，表示暗坏点的个数；作为输出时无效。 取值范围与vi pipe设置的宽度有关。 宽度[120, 3840]：取值范围 [0, HI_ISP_STATIC_DP_COUNT_NORMAL]。 宽度[3841, 7680]：取值范围 [0, 2 * HI_ISP_STATIC_DP_COUNT_NORMAL]。 宽度[7681, 11520]：取值范围 [0, 3 * HI_ISP_STATIC_DP_COUNT_NORMAL]。 宽度[11521, 15360]：取值范围 [0, 4 * HI_ISP_STATIC_DP_COUNT_NORMAL]。 宽度[15361, 16384]：取值范围 [0, 5 * HI_ISP_STATIC_DP_COUNT_NORMAL]。
bright_table [HI_ISP_STATIC_DP_COUNT_MAX]	亮坏点坐标信息，低29bit有效，[12:0]bit为坏点水平坐标，[28:16]bit为坏点垂直坐标。 当该变量作为输入时，表示亮坏点的坐标值查找表；作为输出时，表示所有坏点的坐标值查找表。 取值范围：[0, 0xFFF1FFF]。 数组的有效长度与vi pipe设置的宽度有关。 宽度[120, 3840]：有效长度 HI_ISP_STATIC_DP_COUNT_NORMAL。 宽度[3841, 7680]：有效长度 2 * HI_ISP_STATIC_DP_COUNT_NORMAL。 宽度[7681, 11520]：有效长度 3 * HI_ISP_STATIC_DP_COUNT_NORMAL。 宽度[11521, 15360]：有效长度 4 * HI_ISP_STATIC_DP_COUNT_NORMAL。 宽度[15361, 16384]：有效长度 5 * HI_ISP_STATIC_DP_COUNT_NORMAL。 <pre>#define HI_ISP_STATIC_DP_COUNT_NORMAL 2048 #define HI_ISP_STRIPING_MAX_NUM 5 #define HI_ISP_STATIC_DP_COUNT_MAX (HI_ISP_STATIC_DP_COUNT_NORMAL * HI_ISP_STRIPING_MAX_NUM)</pre>

成员名称	描述
dark_table [HI_ISP_STATIC_DP_COUNT_MAX]	暗的坏点坐标值，低29bit 有效，[12:0]bit为坏点水平坐标，[28:16]bit为坏点垂直坐标。 当该变量作为输入时，表示暗坏点的坐标值查找表；作为输出时无效。 取值范围：[0, 0x1FFF1FFF]。 数组的有效长度与vi pipe设置的宽度有关。 宽度[120, 3840]：有效长度 HI_ISP_STATIC_DP_COUNT_NORMAL。 宽度[3841, 7680]：有效长度 2 * HI_ISP_STATIC_DP_COUNT_NORMAL。 宽度[7681, 11520]：有效长度 3 * HI_ISP_STATIC_DP_COUNT_NORMAL。 宽度[11521, 15360]：有效长度 4 * HI_ISP_STATIC_DP_COUNT_NORMAL。 宽度[15361, 16384]：有效长度 5 * HI_ISP_STATIC_DP_COUNT_NORMAL。
show	是否使能静态坏点显示。 - HI_FALSE：否 - HI_TRUE：是

注意事项

- enable_detect、show与enable是3个互斥的设置，并且对应的优先级是逐次降低的。

3.23.7.5 hi_isp_dp_dynamic_manual_attr

说明

定义动态坏点校正的手动属性。

定义

```
typedef struct {  
    hi_u16 strength;  
    hi_u16 blend_ratio;  
} hi_isp_dp_dynamic_manual_attr;
```

成员

成员名称	描述
strength	DPC动态坏点处理强度，取值越大，DPC处理强度越强，可以校正的动态坏点越多。strength的选择与iso值有很大的关系，iso越高，对应的strength应该越大，在图像中坏点校正较好时，图像中偏红严重，此时可以再调节blend_ratio改善图像质量。 取值范围：[0x0, 0xFF]。
blend_ratio	坏点校正过程所需的融合比率。blend_ratio值越大，图像越模糊。 取值范围：[0x0, 0x80]。

3.23.7.6 hi_isp_dp_dynamic_auto_attr

说明

定义动态坏点校正的自动属性。

定义

```
typedef struct {  
    hi_u16 strength[HI_ISP_AUTO_ISO_NUM];  
    hi_u16 blend_ratio[HI_ISP_AUTO_ISO_NUM];  
} hi_isp_dp_dynamic_auto_attr;
```

成员

成员名称	描述
strength[HI_ISP_AUTO_ISO_NUM]	自动模式下的DPC的处理强度，该数组的16个值分别对应sensor在不同增益下的DPC的处理强度值，一般情况下，增益越大，配置的DPC处理强度值越大，对应关系如表表3-7所示。若连续两档的处理强度不相等，则通过线性插值的方法确定两档中间某个增益值对应的处理强度。 取值范围：[0, 0xFF]。 #define HI_ISP_AUTO_ISO_NUM 16
blend_ratio[HI_ISP_AUTO_ISO_NUM]	自动模式下的融合比率。其值越大，图像越偏绿。若连续两档的融合比率不相等，则通过线性插值的方法确定两档中间的某个增益值对应的融合比率。 取值范围：[0,0x80]。

表 3-7 strength[16]在不同增益情况下对应的设置值

strength	a_gain*d_gain*isp_d_gain(times)
strength[0]	100

strength	a_gain*d_gain*isp_d_gain(times)
strength[1]	200
strength[2]	400
strength[3]	800
strength[4]	1600
strength[5]	3200
strength[6]	6400
strength[7]	12800
strength[8]	25600
strength[9]	51200
strength[10]	102400
strength[11]	204800
strength[12]	409600
strength[13]	819200
strength[14]	1638400
strength[15]	3276800

注意事项

- 该结构体中的默认值均在sensor_cmos_ex.h中，如果用户需要修改默认值，请修改相应参数。如果用户需要对接新的sensor，请参考已经提供的其它sensor的默认值。
- 在 WDR模式与线性模式之下，a_gain*d_gain*isp_d_gain没有差异。

3.23.7.7 hi_isp_dp_dynamic_attr

说明

定义动态坏点校正属性。

定义

```
typedef struct {
    hi_bool enable;
    hi_bool sup_twinkle_en;
    hi_s8 soft_thr;
    hi_u8 soft_slope;
    hi_op_mode op_type;
    hi_isp_dp_dynamic_manual_attr manual_attr;
    hi_isp_dp_dynamic_auto_attr auto_attr;
} hi_isp_dp_dynamic_attr;
```

成员

成员名称	描述
enable	是否使能动态坏点校正功能。 - HI_FALSE: 否 - HI_TRUE: 是
op_type	DPC动态坏点校正工作模式。 - HI_OP_MODE_AUTO: 自动 - HI_OP_MODE_MANUAL: 手动
manual_attr	手动模式下配置动态坏点校正参数。请参见 hi_isp_dp_dynamic_manual_attr 。
auto_attr	自动模式下配置动态坏点校正参数。请参见 hi_isp_dp_dynamic_auto_attr 。
sup_twinkle_en	是否使能闪烁抑制功能。 - HI_FALSE: 否 - HI_TRUE: 是
soft_thr	闪烁抑制的阈值，值越小，所能处理的像素点越多，图像会越模糊。 取值范围：[0, 127]。
soft_slope	闪烁抑制模块中，控制偏差小于soft_thr的像素点的校正情况。 值越大，参与校正处理的像素点越多，图像会越模糊。 取值范围：[0, 255]。

注意事项

- 闪烁抑制功能与hi_isp_dp_dynamic_auto_attr或hi_isp_dp_dynamic_manual_attr中的strength参数存在联动关系，soft_thr和soft_slope取值不合理时，调节strength，有可能在strength的某些档位之间存在校正效果的突变，因此建议在保证soft_thr和soft_slope的配置值不明显影响图像的清晰度的前提下，调节strength，看坏点的校正以及闪烁抑制的效果。
- soft_slope参数设置过大，图像会模糊，同时可能会出现伪彩等副作用，因此soft_slope应设置为不明显影响图像清晰度的参数。

3.24 Mesh Shading

3.24.1 hi_mpi_isp_set_mesh_shading_attr

函数功能

设置Mesh Shading算法参数。

函数原型

hi_s32 hi_mpi_isp_set_mesh_shading_attr(**hi_vi_pipe** vi_pipe, const **hi_isp_shading_attr** *shading_attr)

参数说明

参数名	输入/输出	说明
vi_pipe	输入	VI PIPE号。 取值范围：[0, 12)。
shading_attr	输入	Mesh Shading 算法参数。

返回值说明

- 0：成功
- 非0：失败，参见[ISP返回码](#)

3.24.2 hi_mpi_isp_get_mesh_shading_attr

函数功能

获取Mesh Shading算法参数。

函数原型

hi_s32 hi_mpi_isp_get_mesh_shading_attr(**hi_vi_pipe** vi_pipe, **hi_isp_shading_attr** *shading_attr)

参数说明

参数名	输入/输出	说明
vi_pipe	输入	VI PIPE号。 取值范围：[0, 12)。
shading_attr	输入	Mesh Shading算法参数。

返回值说明

- 0：成功
- 非0：失败，参见[ISP返回码](#)

3.24.3 hi_mpi_isp_set_mesh_shading_gain_lut_attr

函数功能

设置Mesh Shading增益表属性。

函数原型

hi_s32 hi_mpi_isp_set_mesh_shading_gain_lut_attr(**hi_vi_pipe** vi_pipe, const **hi_isp_shading_lut_attr** *shading_lut_attr)

参数说明

参数名	输入/输出	说明
vi_pipe	输入	VI PIPE号。 取值范围：[0, 12)。
shading_lut_attr	输入	Mesh Shading增益表属性。

返回值说明

- 0：成功
- 非0：失败，参见[ISP返回码](#)

3.24.4 hi_mpi_isp_get_mesh_shading_gain_lut_attr

函数功能

获取Mesh Shading增益表属性。

函数原型

hi_s32 hi_mpi_isp_get_mesh_shading_gain_lut_attr(**hi_vi_pipe** vi_pipe, **hi_isp_shading_lut_attr** *shading_lut_attr)

参数说明

参数名	输入/输出	说明
vi_pipe	输入	VI PIPE号。 取值范围：[0, 12)。
shading_lut_attr	输入	Mesh Shading增益表属性。

返回值说明

- 0：成功
- 非0：失败，参见[ISP返回码](#)

3.24.5 数据类型

3.24.5.1 hi_isp_shading_attr

说明

定义Mesh Shading算法参数。

定义

```
typedef struct {  
    hi_bool enable;  
    hi_u16 mesh_strength;  
    hi_u16 blend_ratio;  
} hi_isp_shading_attr;
```

成员

成员名称	描述
enable	是否使能Mesh Shading。 - HI_FALSE：否 - HI_TRUE：是
mesh_strength	用来全局纠正Mesh Shading标定后的强度。当镜头shading很严重的时候，画面四个角补偿的增益很大，容易导致噪声很大，而且画面有格子现象。这个调整mesh_strength至小于一倍默认值，可以减少Mesh Shading的补偿值，让四个角的补偿增益较小一些，达到优化噪声和画面格子问题。 范围[0, 65535]。
blend_ratio	两张增益表的融合比例，表示为第 0 张表的权重值，256 表示只选择第 0 张表，0 表示只选择第 1 张表。 取值范围： [0,256]

3.24.5.2 hi_isp_shading_gain_lut

说明

定义Mesh Shading增益表参数。

定义

```
typedef struct {  
    hi_u16 r_gain[HI_ISP_LSC_GRID_POINTS];  
    hi_u16 gr_gain[HI_ISP_LSC_GRID_POINTS];  
    hi_u16 gb_gain[HI_ISP_LSC_GRID_POINTS];  
    hi_u16 b_gain[HI_ISP_LSC_GRID_POINTS];  
} hi_isp_shading_gain_lut;
```

成员

成员名称	描述
r_gain[HI_ISP_LSC_GRID_POINTS]	用来储存Mesh Shading所用R通道标定数据。该数组表示画面从左至右、从上至下的网格交点处的R分量阴影矫正增益数据值。 取值范围：[0, 1023]。 #define HI_ISP_LSC_GRID_POINTS (HI_ISP_LSC_GRID_COL * HI_ISP_LSC_GRID_ROW)
gr_gain[HI_ISP_LSC_GRID_POINTS]	用来储存Mesh Shading所用Gr通道标定数据。该数组表示画面从左至右、从上至下的网格交点处的Gr分量阴影矫正增益数据值。 取值范围：[0, 1023]。
gb_gain[HI_ISP_LSC_GRID_POINTS]	用来储存Mesh Shading所用Gb通道标定数据。该数组表示画面从左至右、从上至下的网格交点处的Gb分量阴影矫正增益数据值。 取值范围：[0, 1023]。
b_gain[HI_ISP_LSC_GRID_POINTS]	用来储存Mesh Shading所用B通道标定数据。该数组表示画面从左至右、从上至下的网格交点处的B分量阴影矫正增益数据值。 取值范围：[0, 1023]。

注意事项

- 增益表的默认配置与hi_isp_cmos_alg_key中的bit1(Lsc标志位)有关，如果bit1(Lsc=1)，则采用cmos_ex.h中的配置值作为默认值；否则默认配置为1倍增益。
- 增益表的精度与mesh_scale的值有关系，详见hi_isp_shading_lut_attr这一节中的[表1 mesh_scale与增益精度对应关系](#)。在mesh_scale配置为默认值4的情况下，增益表配置为0，表示增益为1倍。

3.24.5.3 hi_isp_bnr_lsc_gain_lut

说明

定义BayerNR所参考的LSC(lens shading correction, 镜头阴影校正)增益表参数。

定义

```
typedef struct {  
    hi_u16 r_gain[HI_ISP_RLSC_POINTS];  
    hi_u16 gr_gain[HI_ISP_RLSC_POINTS];  
    hi_u16 gb_gain[HI_ISP_RLSC_POINTS];  
    hi_u16 b_gain[HI_ISP_RLSC_POINTS];  
} hi_isp_bnr_lsc_gain_lut;
```


成员

成员名称	描述
r_gain[HI_ISP_RLS_C_POINTS]	用来储存BNR LSC所用R通道标定数据。 取值范围：[0, 65535]。 #define HI_ISP_RLSC_POINTS 129
gr_gain[HI_ISP_RLS_C_POINTS]	用来储存BNR LSC所用Gr通道标定数据。 取值范围：[0, 65535]。 #define HI_ISP_RLSC_POINTS 129
gb_gain[HI_ISP_RLSC_POINTS]	用来储存BNR LSC所用 Gb 通道标定数据。 取值范围：[0, 65535]。 #define HI_ISP_RLSC_POINTS 129
b_gain[HI_ISP_RLS_C_POINTS]	用来储存BNR LSC所用B通道标定数据。 取值范围：[0, 65535]。 #define HI_ISP_RLSC_POINTS 129

3.24.5.4 hi_isp_shading_lut_attr

说明

定义Mesh Shading增益表属性。

定义

```
typedef struct {  
    hi_u8 mesh_scale;  
    hi_u16 x_grid_width[HI_ISP_MLSC_X_HALF_GRID_NUM];  
    hi_u16 y_grid_width[HI_ISP_MLSC_Y_HALF_GRID_NUM];  
    hi_isp_shading_gain_lut lsc_gain_lut[HI_ISP_MLSC_GAIN_LUT_NUM];  
    hi_isp_bnr_lsc_gain_lut bnr_lsc_gain_lut;  
} hi_isp_shading_lut_attr;
```

成员

成员名称	描述
mesh_scale	增益表精度控制参数。当选择的精度较低时（比如 mesh_scale为3），画面衰减幅度大，可能会出现格子现象，可以提高精度（比如7）重新标定。 取值范围：[0, 7]。
x_grid_width[HI_ISP_MLSC_X_HALF_GRID_NUM]	用来储存各grid分区宽度大小信息。该接口各分量最小值为4，总和应为原画面宽度的四分之一。（例如原画面大小为1080p，则该接口各参数总和应为480） 取值范围：[4,width/4 - 60]，width为原画面的宽度。 #define HI_ISP_LSC_GRID_COL 33 #define HI_ISP_MLSC_X_HALF_GRID_NUM ((HI_ISP_LSC_GRID_COL - 1) / 2)

成员名称	描述
y_grid_width[HI_ISP_MLSC_Y_HALF_GRID_NUM]	用来储存各 GRID 分区高度大小信息。该接口各分量最小值为 4，总和应为原画面高度的四分之一。（例如原画面大小为 1080p，则该接口各参数总和应为 270） 取值范围：[4,height/4 - 60]，height为原画面的高度。 #define HI_ISP_LSC_GRID_ROW 33 #define HI_ISP_MLSC_Y_HALF_GRID_NUM ((HI_ISP_LSC_GRID_ROW - 1) / 2)
lsc_gain_lut[HI_ISP_MLSC_GAIN_LUT_NUM]	两组色温下的增益表配置。硬件基于这两组表以及 blend_ratio进行当前色温下校正增益表的计算。 取值范围：[0, 1023]。 #define HI_ISP_MLSC_GAIN_LUT_NUM 2
bnr_lsc_gain_lut	用于BNR LSC参考所用的增益表。 取值范围：[0, 65535]。

注意事项

- 调整x_grid_width或y_grid_width里的值即调整分区宽度或者高度。注意由于整体画面为上下、左右对称，因此每当设置一个分区的宽度或者高度时，其对称分区的宽度或高度也会相应发生变化。改变后的分区宽度与高度请满足总和为原画面大小宽度或高度的四分之一。由于调节比较复杂并容易造成Color Shading现象，一般不建议手动调节。分辨率切换的时候，x_grid_width与 y_grid_width自动重新配置。
- 硬件基于配置的两组色温下的增益表以及blend_ratio进行插值得到当前色温下的校正增益表的计算，从而校正整幅图像的shading现象，其中blend_ratio为 lsc_gain_lut[0]对应的权重。
- 在单光源模式下，lsc_gain_lut结构体需要配置成一致，此时需要配置成标定的第一个光源。
- 增益位宽为10bit，即最大值为1023。
 - 当mesh_scale为 0 至 3 时，增益的表示与其他算法一致，增益精度从高到低，取值范围从小变大，1倍增益不为 0；
 - 当mesh_scale为 4 至 7 时，精度从高到低，取值范围从小变大，增益的表示方式略有差异，实际的增益计算方式为通用增益减去对应精度的 1倍增益，比如 mesh_scale为 5时，按照增益精度为1.9，若通用增益为886，1倍对应增益512，那么实际的增益为886-512=374，这种表示方式下，最小的增益为1倍，但是表示的范围比上一种要大。
- 在能够完全校正的情况下，增益精度mesh_scale应该下尽量选择精度高的档位，提高校正的准确度，例如当mesh_scale为0或4时都能校正，应当选择4。对于某些镜头阴影特别严重的极端情况下，存在无法完全校正的可能，当选择较大增益倍数（如mesh_scale为3）校正时，可能会在边缘处出现暗条纹，此时需要降低增益倍数来提高精度（设 mesh_scale为2或 7），重新标定，直到暗条纹消失为止。

表 3-8 mesh_scale 与增益精度对应关系

mesh_scale	增益表的数据格式	增益取值范围	1 倍增益对应增益配置值
0	1.9	[0,2)	512
1	2.8	[0,4)	256
2	3.7	[0,8)	128
3	4.6	[0,16)	64
4	0.10	[1,2)	0
5	1.9	[1,3)	0
6	2.8	[1,5)	0
7	3.7	[1,9)	0

3.25 DRC

3.25.1 hi_mpi_isp_set_drc_attr

函数功能

设置动态范围压缩参数。

函数原型

```
hi_s32 hi_mpi_isp_set_drc_attr(hi_vi_pipe vi_pipe, const hi_isp_drc_attr *drc)
```

参数说明

参数名	输入/输出	说明
vi_pipe	输入	VI PIPE号。 取值范围：[0, 12)。
drc	输入	动态范围压缩参数。

返回值说明

- 0：成功
- 非0：失败，参见[ISP返回码](#)

3.25.2 hi_mpi_isp_get_drc_attr

函数功能

获取动态范围压缩参数。

函数原型

hi_s32 hi_mpi_isp_get_drc_attr(**hi_vi_pipe** vi_pipe, **hi_isp_drc_attr** *drc)

参数说明

参数名	输入/输出	说明
vi_pipe	输入	VI PIPE号。 取值范围：[0, 12)。
drc	输入	动态范围压缩参数。

返回值说明

- 0：成功
- 非0：失败，参见[ISP返回码](#)。

3.25.3 数据类型

3.25.3.1 hi_isp_drc_manual_attr

说明

定义ISP DRC手动属性。

定义

```
typedef struct {  
    hi_u16 strength;  
} hi_isp_drc_manual_attr;
```

成员

成员名称	描述
strength	手动模式下DRC的强度，值越大，阴影和高光之间的差分增益就越高。 取值范围：[0x0, 0x3FF]。

注意事项

在线性和WDR模式下均可以达到最大值。

3.25.3.2 hi_isp_drc_auto_attr

说明

定义ISP DRC自动属性。

定义

```
typedef struct {  
    hi_u16 strength;  
    hi_u16 strength_max;  
    hi_u16 strength_min;  
} hi_isp_drc_auto_attr;
```

成员

成员名称	描述
strength	自动模式下DRC的强度，参数可以读写，值越大，整体强度越大。 取值范围：[0x0, 0x3FF]。
strength_max	自动模式下DRC实际生效强度值的上限。 取值范围：[0x0, 0x3FF]。
strength_min	自动模式下DRC实际生效强度值的下限。 取值范围：[0x0, 0x3FF]。

注意事项

- 实际生效强度为：算法AUTO模式计算出的强度 * strength / 512。
- 注意strength_max必须大于等于strength_min，否则两个参数均不生效。
- 线性和WDR自动模式下DRC强度的建议取值范围不同，strength_max/min的建议取值范围也不同：
 - 线性自动模式下，strength_max和strength_min的建议取值范围为：[0x0, 0x200]，其中strength_max默认值为 0x200，strength_min默认值为0x0。
 - 在WDR自动模式下，strength_max和strength_min的建议取值范围为：[0x200, 0x3FF]，其中strength_max默认值为0x3FF，strength_min默认值为0x200。

3.25.3.3 hi_isp_drc_asymmetry_curve_attr

说明

定义ISP DRC Asymmetry Curve属性。

定义

```
typedef struct {  
    hi_u8 asymmetry;  
    hi_u8 second_pole;  
    hi_u8 stretch;  
    hi_u8 compress;  
} hi_isp_drc_asymmetry_curve_attr;
```

成员

成员名称	描述
asymmetry	用于生成Asymmetry Curve, 值越小，暗区亮度拉伸越明显。 取值范围：[1, 30]
second_pole	用于生成Asymmetry Curve，值越小，亮区亮度压制越强，有利于保留高亮细节。 取值范围：[150, 210]
stretch	用于生成Asymmetry Curve，值越小，整体亮度越亮。 取值范围：[30, 60]
compress	用于生成Asymmetry Curve，值越小，整体亮度越亮。 取值范围：[100, 200]

3.25.3.4 hi_isp_drc_curve_select

说明

定义ISP DRC Tone Mapping曲线类型。

定义

```
typedef enum {  
    HI_ISP_DRC_CURVE_ASYMMETRY = 0x0,  
    HI_ISP_DRC_CURVE_USER,  
    HI_ISP_DRC_CURVE_BUTT  
} hi_isp_drc_curve_select;
```

成员

成员名称	描述
HI_ISP_DRC_CURVE_ASYMMETRY	选择 Asymmetry Curve。
HI_ISP_DRC_CURVE_USER	选择用户自定义曲线。
HI_ISP_DRC_CURVE_BUTT	无效值。

3.25.3.5 hi_isp_drc_attr

说明

定义 ISP DRC 属性。

定义

```
typedef struct {  
    hi_bool enable;  
    hi_isp_drc_curve_select curve_select;  
    hi_u8 purple_reduction_strength;  
    hi_u8 local_mixing_bright_max;  
    hi_u8 local_mixing_bright_min;  
    hi_u8 local_mixing_bright_thr;  
    hi_s8 local_mixing_bright_slo;  
    hi_u8 local_mixing_dark_max;  
    hi_u8 local_mixing_dark_min;  
    hi_u8 local_mixing_dark_thr;  
    hi_s8 local_mixing_dark_slo;  
  
    hi_u8 bright_gain_lmt;  
    hi_u8 bright_gain_lmt_step;  
    hi_u8 dark_gain_lmt_y;  
    hi_u8 dark_gain_lmt_c;  
    hi_u16 color_correction_lut[HI_ISP_DRC_CC_NODE_NUM];  
    hi_u16 tone_mapping_value[HI_ISP_DRC_TM_NODE_NUM];  
  
    hi_u8 contrast_control;  
    hi_s8 detail_adjust_factor;  
  
    hi_u8 spatialflt_coef;  
    hi_u8 rangeflt_coef;  
    hi_u8 rangeada_max;  
  
    hi_u8 grad_rev_max;  
    hi_u8 grad_rev_thr;  
  
    hi_op_mode op_type;  
    hi_isp_drc_manual_attr manual_attr;  
    hi_isp_drc_auto_attr auto_attr;  
    hi_isp_drc_asymmetry_curve_attr asymmetry_curve;  
} hi_isp_drc_attr;
```

成员

成员名称	描述
enable	DRC 动态范围压缩使能。 取值范围： - HI_FALSE：关闭 - HI_TRUE：使能
curve_select	DRC Tone mapping 曲线选择。 取值范围： - HI_ISP_DRV_CURVE_ASYMMETRY：Asymmetry curve - HI_ISP_DRC_CURVE_USER：用户自定义曲线

成员名称	描述
purple_reduction_strength	紫边校正强度，值越大，紫边校正强度越大。 取值范围：[0x0, 0x80]
local_mixing_bright_max	线性细节增强：用来控制正向细节的增益的最大值。 取值范围：[0x0, 0x80]
local_mixing_bright_min	线性细节增强：用来控制正向细节的增益的最小值。 取值范围：[0x0, 0x40]
local_mixing_bright_thr	线性细节增强：用来控制正向细节增益随亮度自适应的阈值。当像素亮度超过该阈值，增益将从最大值向最小值过渡（或反之，取决于斜率参数的符号）。 取值范围：[0x0, 0xFF]
local_mixing_bright_slo	线性细节增强：用来控制正向细节增益随亮度自适应的斜率。绝对值越大，从最小值到最大值（或反之）过渡越快。 取值范围：[-0x7, 0x7]
local_mixing_dark_max	线性细节增强：用来控制负向细节的增益最大值。 取值范围：[0x0, 0x80]
local_mixing_dark_min	线性细节增强：用来控制负向细节的增益最小值。 取值范围：[0x0, 0x40]
local_mixing_dark_thr	线性细节增强：用来控制负向细节增益随亮度自适应的阈值。当像素亮度超过该阈值，增益将从最小值向最大值过渡（或反之，取决于斜率参数的符号）。 取值范围：[0x0, 0xFF]
local_mixing_dark_slo	线性细节增强：用来控制负向细节增益随亮度自适应的斜率。绝对值越大，从最小值到最大值（或反之）过渡越快。 取值范围：[-0x7, 0x7]
bright_gain_lmt	亮区亮度增益限制目标值，值越大，限制越大。 取值范围：[0x0, 0xF]
bright_gain_lmt_step	亮区亮度增益限制自适应步长，值越小，限制越大。 取值范围：[0x0, 0xF]
dark_gain_lmt_y	暗区亮度增益限制，值越大，限制越大。 取值范围：[0x0, 0x85]

成员名称	描述
dark_gain_lmt_c	暗区色度增益限制，值越大，限制越大。 取值范围：[0x0, 0x85]
color_correction_lut[HI_ISP_DRC_CC_NODE_NUM]	颜色校正系数LUT，值越小，饱和度越低。 取值范围：[0x0, 0x400] #define HI_ISP_DRC_CC_NODE_NUM 33
tone_mapping_value[HI_ISP_DRC_TM_NODE_NUM]	用户自定义Tone Mapping曲线 LUT。 取值范围：[0x0, 0xFFFF] #define HI_ISP_DRC_TM_NODE_NUM 200
contrast_control	局部对比度控制，参数效果与图像亮度分布相关，一般场景下建议值为6到10 之间。 取值范围：[0x0, 0xF]
detail_adjust_factor	Filter细节微调系数。值越大，整体细节越强。 取值范围：[-0xF, 0xF]
spatial_ftl_coef	Filter 空域滤波系数。值越大，运动光晕越不明显，细节越少；值越小，细节表现越好，运动halo越明显。 取值范围：[0x0, 0x5]
range_ftl_coef	Filter 值域滤波系数。值越大，halo越明显；值越小，halo表现越好，但是在强边缘处可能会出现描边现象。 取值范围：[0x0, 0xA]
range_ada_max	控制值域滤波系数的自适应范围。实际生效滤波系数在[range_ftl_coef, MIN(range_ftl_coef +range_ada_max, 0xA)]范围内。调大该值有助于减弱边线，但同时可能造成背光物体细节损失。 取值范围：[0x0, 0x8]
grad_rev_max	去边线强度。值越大，边线减弱越明显，但是可能会带来细节损失。 取值范围：[0x0, 0x40]
grad_rev_thr	边线检测阈值。值越大，边线减弱越明显，但是可能会带来细节损失。 取值范围：[0x0, 0x80]
op_type	DRC工作类型。 取值范围： - HI_OP_MODE_AUTO：使用自动 DRC 功能。 - HI_OP_MODE_MANUAL：使用手动 DRC 功能。
manual_attr	DRC 手动参数。

成员名称	描述
auto_attr	DRC 自动参数。
asymmetry_curve	DRC Asymmetry Curve 属性。

注意事项

- 建议color_correction_lut 的值不要调试过小，因为降低饱和度的同时会带来高频细节的损失，用户可以配合CA模块来调试整体饱和度。
- dark_gain_lmt_y和dark_gain_lmt_c两个接口的实际效果不完全平滑，即参数在两个相邻值之间切换时，效果可能存在跳变。

3.26 Gamma

3.26.1 hi_mpi_isp_get_gamma_attr

函数功能

获取Gamma属性。

函数原型

[hi_s32](#) hi_mpi_isp_get_gamma_attr([hi_vi_pipe](#) vi_pipe, [hi_isp_gamma_attr](#) *gamma_attr)

参数说明

参数名	输入/输出	说明
vi_pipe	输入	VI PIPE号。 取值范围：[0, 12)。
gamma_attr	输入	Gamma 属性。

返回值说明

- 0：成功
- 非0：失败，参见[ISP返回码](#)。

3.26.2 hi_mpi_isp_set_gamma_attr

函数功能

设置Gamma属性。

函数原型

```
hi_s32 hi_mpi_isp_set_gamma_attr(hi_vi_pipe vi_pipe, const
hi_isp_gamma_attr *gamma_attr)
```

参数说明

参数名	输入/输出	说明
vi_pipe	输入	VI PIPE号。 取值范围：[0, 12)。
gamma_attr	输入	Gamma 属性。

返回值说明

- 0：成功。
- 非0：失败，参见ISP返回码。

3.26.3 数据类型

3.26.3.1 hi_isp_gamma_attr

说明

定义ISP Gamma校正属性。

定义

```
typedef struct {
    hi_bool enable;
    hi_u16 table[HI_ISP_GAMMA_NODE_NUM];
    hi_isp_gamma_curve_type curve_type;
} hi_isp_gamma_attr;
```

成员

成员名称	描述
enable	Gamma校正功能使能。 HI_FALSE：关闭； HI_TRUE：使能。
table[HI_ISP_GAMMA_NODE_NUM]	LUT表，用来表示输入输出值大小。 取值范围：[0x0, 0xFFFF] #define HI_ISP_GAMMA_NODE_NUM 1025

成员名称	描述
curve_type	Gamma曲线选择。 • 默认值为： HI_ISP_GAMMA_CURVE_DEFAULT。 • SDR模式为： HI_ISP_GAMMA_CURVE_SRGB • HDR模式为： HI_ISP_GAMMA_CURVE_HDR，当前版本不支持该选项 • 用户自定义模式为： HI_ISP_GAMMA_CURVE_USER_DEFINE

注意事项

- Gamma校正 R、G、B 时调用的是同一组Gamma Table。
- curve_type不支持配置为HI_ISP_GAMMA_CURVE_HDR，此时会返回参数错误码HI_ERR_ISP_ILLEGAL_PARAM。选择HI_ISP_GAMMA_CURVE_DEFAULT、HI_ISP_GAMMA_CURVE_SRGB 等模式时，系统会自动配置预设的曲线，此时通过工具拖拉曲线均不生效。只有选择HI_ISP_GAMMA_CURVE_USER_DEFINE时，才能通过工具进行曲线修改。
- 使用任意类型的预设曲线后，可以切换成HI_ISP_GAMMA_CURVE_USER_DEFINE并以预设曲线为基础进行Gamma调试。

3.26.3.2 hi_isp_gamma_curve_type

说明

定义 Gamma 曲线类型。

定义

```
typedef enum {  
    HI_ISP_GAMMA_CURVE_DEFAULT = 0x0,  
    HI_ISP_GAMMA_CURVE_SRGB,  
    HI_ISP_GAMMA_CURVE_HDR,  
    HI_ISP_GAMMA_CURVE_USER_DEFINE,  
    HI_ISP_GAMMA_CURVE_BUTT  
} hi_isp_gamma_curve_type;
```

成员

成员名称	描述
HI_ISP_GAMMA_CURVE_DEFAULT	默认传输曲线。
HI_ISP_GAMMA_CURVE_SRGB	默认SDR传输曲线，由 BT.709 标准定义。

成员名称	描述
HI_ISP_GAMMA_CURVE_HDR	默认HDR传输曲线，由 SMPTE.2084 标准定义，不支持。
HI_ISP_GAMMA_CURVE_USER_DEFINE	用户自定义传输曲线。
HI_ISP_GAMMA_CURVE_BUTT	传输曲线类型枚举终止位指示。

注意事项

用户自定义Gamma曲线时，必须确保Gamma表配置正确。WDR模式下的Gamma 曲线与线性模式不一样，可以通过PQ Tools工具生成。

3.27 HLC

3.27.1 hi_mpi_isp_get_hlc_attr

函数功能

获取图像的高光遮蔽属性。

函数原型

```
hi_s32 hi_mpi_isp_get_hlc_attr(hi_vi_pipe vi_pipe, hi_isp_hlc_attr *hlc_attr)
```

参数说明

参数名	输入/输出	说明
vi_pipe	输入	VI PIPE号。 取值范围：[0, 12)。
hlc_attr	输入	图像高光遮蔽属性。

返回值说明

- 0：成功
- 非0：失败，参见ISP返回码。

3.27.2 hi_mpi_isp_set_hlc_attr

函数功能

设置图像的高光遮蔽属性。

函数原型

```
hi_s32 hi_mpi_isp_set_hlc_attr(hi_vi_pipe vi_pipe, const hi_isp_hlc_attr
*hlc_attr)
```

参数说明

参数名	输入/输出	说明
vi_pipe	输入	VI PIPE号。 取值范围：[0, 12)。
hlc_attr	输入	图像高光遮蔽属性。

返回值说明

- 0：成功
- 非0：失败，参见ISP返回码。

3.27.3 数据类型

3.27.3.1 hi_isp_hlc_attr

说明

定义ISP HLC属性。

定义

```
typedef struct {
    hi_bool enable;
    hi_u8 luma_thr;
    hi_u8 luma_target;
} hi_isp_hlc_attr
```

成员

成员名称	描述
enable	HLC高光遮蔽功能的使能。 - HI_FALSE：关闭； - HI_TRUE：使能。
luma_thr	高光遮蔽的阈值。亮度值大于该值的区域将会被拉灰成目标值为luma_target的低亮度值。该值一般都设置的比较大，否则，图像的很多区域都会被拉灰。 取值范围：[0, 255]。

成员名称	描述
luma_target	高光遮蔽拉灰的目标值。亮度值大于 luma_thr 的区域将被拉灰成目标值为 luma_target 的低亮度值，也即该区域的亮度将会被明显拉低拉暗。该值一般都设置的比较小，以减少高亮光源对人眼的刺激。 取值范围：[0, 255]。

3.28 EdgeMark

3.28.1 hi_mpi_isp_get_edge_mark_attr

函数功能

获取图像边缘标记属性。

函数原型

hi_s32 hi_mpi_isp_get_edge_mark_attr(**hi_vi_pipe** vi_pipe,
hi_isp_edge_mark_attr *edge_mark_attr)

参数说明

参数名	输入/输出	说明
vi_pipe	输入	VI PIPE号。 取值范围：[0, 12)。
edge_mark_attr	输入	图像边缘标记属性。

返回值说明

- 0：成功
- 非0：失败，参见[ISP返回码](#)。

3.28.2 hi_mpi_isp_set_edge_mark_attr

函数功能

设置图像边缘标记属性。

函数原型

hi_s32 hi_mpi_isp_set_edge_mark_attr(**hi_vi_pipe** vi_pipe, const
hi_isp_edge_mark_attr *edge_mark_attr)

参数说明

参数名	输入/输出	说明
vi_pipe	输入	VI PIPE号。 取值范围：[0, 12)。
edge_mark_attr	输入	图像边缘标记属性。

返回值说明

- 0：成功
- 非0：失败，参见[ISP返回码](#)。

3.28.3 数据类型

3.28.3.1 hi_isp_edge_mark_attr

说明

定义ISP EdgeMark属性。

定义

```
typedef struct {  
    hi_bool enable;  
    hi_u8 threshold;  
    hi_u32 color;  
} hi_isp_edge_mark_attr;
```

成员

成员名称	描述
enable	EdgeMark边缘标记功能的使能。 - HI_FALSE：关闭 - HI_TRUE：使能
threshold	边缘标记的阈值。计算得到的图像的局部统计量大于该阈值，则认为是强边缘或者强的纹理特征，将大于该阈值的强边缘和强的纹理特征标记上色。该值越小，则被标记的边缘和强的纹理特征越多。 取值范围：[0, 255]。

成员名称	描述
color	边缘标记的颜色设置值。该值设置的是RGB三分量值的大小。该值的0-7bit的低8位是表示B分量的大小，8-15bit的中间8位是G分量的大小，16-23bit的高8位是R分量的大小，24-31bit的最高8位保留不用。通过设置color可以为边缘标记的边缘标记上色选择合适的颜色，一般所选的颜色最好和图像的背景颜色形成明显的反差对比。 取值范围：[0, 0xFFFFFFFF]。 默认值0xFF0000。

3.29 Sharpen

3.29.1 hi_mpi_isp_set_sharpen_attr

函数功能

设定图像锐化属性。

函数原型

hi_s32 hi_mpi_isp_set_sharpen_attr(**hi_vi_pipe** vi_pipe, const **hi_isp_sharpen_attr** *shp_attr)

参数说明

参数名	输入/输出	说明
vi_pipe	输入	VI PIPE号。 取值范围：[0, 12)。
shp_attr	输入	图像锐化属性。

返回值说明

- 0：成功
- 非0：失败，参见[ISP返回码](#)。

3.29.2 hi_mpi_isp_get_sharpen_attr

函数功能

获取图像锐化属性。

函数原型

hi_s32 hi_mpi_isp_get_sharpen_attr(**hi_vi_pipe** vi_pipe, **hi_isp_sharpen_attr** *shp_attr)

参数说明

参数名	输入/输出	说明
vi_pipe	输入	VI PIPE号。 取值范围：[0, 12)。
shp_attr	输入	图像锐化属性。

返回值说明

- 0：成功
- 非0：失败，参见[ISP返回码](#)。

3.29.3 数据类型

3.29.3.1 hi_isp_sharpen_manual_attr

说明

定义图像锐化的手动属性。

定义

```
typedef struct {  
    hi_u8 luma_wgt[HI_ISP_SHARPEN_LUMA_NUM];  
    hi_u16 texture_strength[HI_ISP_SHARPEN_GAIN_NUM];  
    hi_u16 edge_strength[HI_ISP_SHARPEN_GAIN_NUM];  
    hi_u16 texture_freq;  
    hi_u16 edge_freq;  
    hi_u8 over_shoot;  
    hi_u8 under_shoot;  
    hi_u8 shoot_sup_strength;  
    hi_u8 shoot_sup_adj;  
    hi_u8 detail_ctrl;  
    hi_u8 detail_ctrl_threshold;  
    hi_u8 edge_filt_strength;  
    hi_u8 edge_filt_max_cap;  
    hi_u8 r_gain;  
    hi_u8 g_gain;  
    hi_u8 b_gain;  
    hi_u8 skin_gain;  
    hi_u16 max_sharp_gain;  
} hi_isp_sharpen_manual_attr;
```

成员

成员名称	描述
<code>luma_wgt[HI_ISP_SHARPEN_LUMA_NUM]</code>	亮度锐化权重。 满量程0-255的亮度被32个等分点平均分为32段亮度区间，每一段亮度区间对应一个亮度权重。比如0-7的亮度区间的权重是<code>luma_wgt[0]</code>，8-15的亮度区间的权重是<code>luma_wgt[1]</code>，依次类推。如图3-7所示。值越大，图像锐化程度越高，反之，越弱。 取值范围：[0, 127]，建议值 127。 <pre>#define HI_ISP_SHARPEN_LUMA_NUM 32</pre>
<code>texture_strength[HI_ISP_SHARPEN_GAIN_NUM]</code>	该参数用于设置图像无方向的细节纹理的锐度。 该值越大，无方向的细节纹理的清晰度越高。 该参数是一个长度为32的数组，是一个32段的连续的强度曲线，如图3-8所示。 取值范围：[0, 4095]，建议值300。 <pre>#define HI_ISP_SHARPEN_GAIN_NUM 32</pre>
<code>edge_strength[HI_ISP_SHARPEN_GAIN_NUM]</code>	该参数用于设置图像带方向的边缘的锐度。 该值越大，带方向的边缘的锐度越高。 该参数是一个长度为32的数组，是一个32段的连续的强度曲线，如图3-9所示。 取值范围：[0, 4095]，建议值400。
<code>texture_freq</code>	该参数用于设置图像的无方向细节纹理增强的频率。 该值越大，细节纹理的增强就越偏向于高频增强，细节纹理就越细碎。反之，该值越小，细节纹理就越粗越圆润。 <code>texture_freq</code>对应于强度参数<code>texture_strength</code>。<code>texture_freq</code>越大，图像的细节纹理就越细碎，该值过大，会导致图像的细节纹理过于细碎而不自然，甚至感觉模糊。 取值范围：[0, 4095]，建议值128。
<code>edge_freq</code>	该参数用于设置图像的带方向的边缘增强频率。 该值越大，边缘的增强就越偏向于高频增强，图像的边缘就越纤薄越窄。反之，该值越小，边缘就越粗越圆润。 <code>edge_freq</code>对应于强度参数<code>edge_strength</code>。<code>edge_freq</code>越大，图像的边缘就越纤薄越窄，该值过大，会导致图像的边缘过于纤薄而出现虚边现象。 取值范围：[0, 4095]，建议值96。
<code>over_shoot</code>	设置图像的overshoot（锐化后的白边白点）的强度。 该值越小，锐化后的白边白点越弱，清晰度也会下降。该值过小，图像会呈油画效果。 取值范围：[0, 127]，建议值100。

成员名称	描述
under_shoot	设置图像的undershoot（锐化后的黑边黑点）的强度。 该值越小，锐化后的黑边黑点越弱，清晰度也会下降。该值过小，图像会呈油画效果。 取值范围：[0, 127]，建议值127。
shoot_sup_strength	设置图像锐化后的overshoot和undershoot的抑制强度。用于在保证清晰度不明显下降的前提下，抑制锐化后的图像的overshoot和undershoot的宽度和幅度。 该值越大，锐化后的图像的overshoot和undershoot的宽度越窄、强度越小。该值变大，理论上不会影响图像的清晰度，只是黑白边变窄以后，会减弱人眼的锐度感受。 该参数需要和shoot_sup_adj配合使用。 取值范围：[0, 255]，建议值8。
shoot_sup_adj	用于对图像锐化后的overshoot和undershoot的抑制强度进行调节。 该参数配合shoot_sup_strength使用，用于调节shoot_sup_strength作用的区域范围。该值越小，则越多的纹理区域的shoot会被shoot_sup_strength抑制；该值越大，则只有很强的边缘的shoot会被shoot_sup_strength抑制，纹理区域的shoot不会被抑制。 取值范围：[0, 15]，建议值9。
detail_ctrl	用于控制图像的细节纹理区域的shoot强度，shoot越大，细节纹理区的清晰度越高。 • 若该值等于128，则图像的细节纹理区域的shoot强度和大边缘的shoot强度一致，都分别等于over_shoot和under_shoot。 • 若该值大于128，则图像的细节纹理的shoot强度大于大边缘，大边缘的shoot强度分别等于over_shoot和under_shoot。 • 若该值小于128，则图像的细节纹理的shoot强度小于大边缘，大边缘的shoot强度分别等于over_shoot和under_shoot。 取值范围：[0, 255]。
detail_ctrl_threshold	图像的细节纹理区的shoot强度的控制阈值。 该值配合detail_ctrl使用，用于区分detail_ctrl所控制shoot的纹理区和边缘，也即纹理区和边缘的区分阈值。小于该值的区域为纹理区，该纹理区域的shoot会被detail_ctrl单独控制，而大于该阈值的边缘区域的shoot依然等于over_shoot和under_shoot。 取值范围：[0, 255]，建议值160。

成员名称	描述
edge_filt_strength	边缘滤波强度调试参数，实现图像锐化边缘的范围和边缘平滑强度的控制。 该值越大，判为边缘的区域越多、也越宽，edge_strength起作用的图像边缘就越多，而且，该值越大，沿着边缘方向的平滑滤波强度也越大，边缘就越平滑。反之，判为边缘的区域越少、也越窄，edge_strength起作用的图像区域越少，边缘平滑就越弱。 取值范围：[0, 63]，建议值53。
edge_filt_max_cap	边缘滤波强度范围的调试参数，该值越大，边缘滤波的最大强度也最大，edge_filt_strength的可调试范围也越大。 一般建议该值大小控制30以内。 取值范围：[0, 47]，建议值18。
r_gain	深红色区域的锐化增益控制参数。该值越大，则深红色区域的锐化强度越大。 取值范围：[0, 31]，建议值28。
g_gain	绿色区域的锐化增益控制参数。该值越大，则绿色区域的锐化强度越大。 取值范围：[0, 255]，建议值32。
b_gain	深蓝色区域的锐化增益控制参数。该值越大，则深蓝色区域的锐化强度越大。 取值范围：[0, 31]，建议值 28。
skin_gain	肤色区域的锐化增益控制参数。该值越大，则肤色区域的锐化强度越大。 取值范围：[0, 31]，建议值23。
max_sharp_gain	图像锐化的最大增益限制值参数。该值越大，图像的锐化幅度越大，反之，锐化幅度越小。适当的调小该参数，可以减少图像的过锐化，减少图像锐化后的黑白点。 取值范围：[0, 0x7FF]，建议值0xA0。

图 3-7 Sharpen 的像素亮度 luma 和锐化强度 luma_wgt 的关系曲线

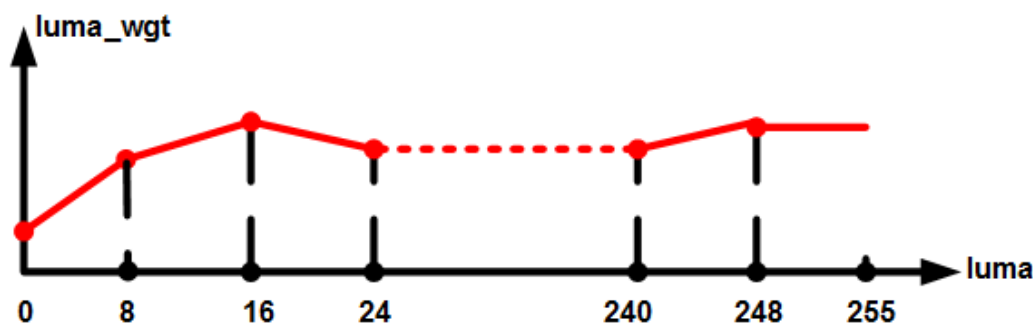
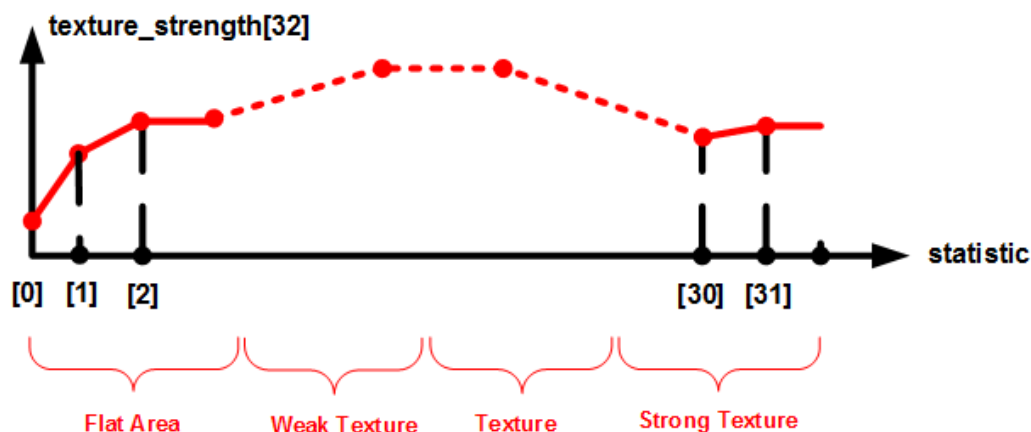
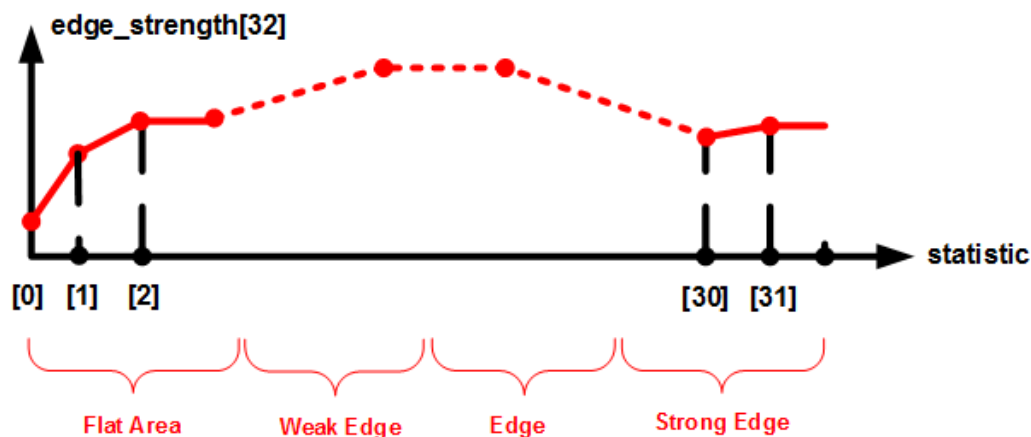


图 3-8 texture_strength[HI_ISP_SHARPEN_GAIN_NUM]强度曲线示意图



强度曲线的横坐标var是从图像中提取的方差统计特征，横坐标var被均分为32段，用于区分出图像的Flat Area（平坦区域）、Weak Texture（弱纹理）、Texture（纹理）和Strong Texture（强纹理）。纵坐标就是强度参数texture_strength的32个强度值，用户可以通过设置该曲线上的32个强度值来为平坦区域、弱纹理区域、纹理区域和强纹理区域设置不同的锐化强度。这4个区域并没有明显的区分界限，都是连续过渡，用户可以通过实际效果来调整纵坐标强度来为不同的区域设置不同的强度。

图 3-9 edge_strength[HI_ISP_SHARPEN_GAIN_NUM]强度曲线示意图



强度曲线的横坐标var是从图像中提取的方差统计值，横坐标var被均分为32段，用于区分出图像的Flat Area（平坦区域）、Weak Edge（弱边缘）、Edge（边缘）和Strong Edge（强边缘）。纵坐标是强度参数edge_strength的32个强度值，用户可以通过设置该曲线上的32个强度值来为平坦区域、弱边缘、边缘和强边缘设置不同的锐化强度。这4个区域并没有明显的区分界限，都是连续过渡，用户可以通过实际效果来调整纵坐标强度来为不同的区域设置不同的强度。

3.29.3.2 hi_isp_sharpen_auto_attr

说明

定义图像锐化的自动属性。

定义

```
typedef struct {
    hi_u8 luma_wgt[HI_ISP_SHARPEN_LUMA_NUM][HI_ISP_AUTO_ISO_NUM];
    hi_u16 texture_strength[HI_ISP_SHARPEN_GAIN_NUM][HI_ISP_AUTO_ISO_NUM];
    hi_u16 edge_strength[HI_ISP_SHARPEN_GAIN_NUM][HI_ISP_AUTO_ISO_NUM];
    hi_u16 texture_freq[HI_ISP_AUTO_ISO_NUM];
    hi_u16 edge_freq[HI_ISP_AUTO_ISO_NUM];
    hi_u8 over_shoot[HI_ISP_AUTO_ISO_NUM];
    hi_u8 under_shoot[HI_ISP_AUTO_ISO_NUM];
    hi_u8 shoot_sup_strength[HI_ISP_AUTO_ISO_NUM];
    hi_u8 shoot_sup_adj[HI_ISP_AUTO_ISO_NUM];
    hi_u8 detail_ctrl[HI_ISP_AUTO_ISO_NUM];
    hi_u8 detail_ctrl_threshold[HI_ISP_AUTO_ISO_NUM];
    hi_u8 edge_filt_strength[HI_ISP_AUTO_ISO_NUM];
    hi_u8 edge_filt_max_cap[HI_ISP_AUTO_ISO_NUM];
    hi_u8 r_gain[HI_ISP_AUTO_ISO_NUM];
    hi_u8 g_gain[HI_ISP_AUTO_ISO_NUM];
    hi_u8 b_gain[HI_ISP_AUTO_ISO_NUM];
    hi_u8 skin_gain[HI_ISP_AUTO_ISO_NUM];
    hi_u16 max_sharp_gain[HI_ISP_AUTO_ISO_NUM];
} hi_isp_sharpen_auto_attr;
```

成员

成员名称	描述
luma_wgt[HI_ISP_SHARPEN_LUMA_NUM][HI_ISP_AUTO_ISO_NUM]	亮度锐化权重。 满量程0-255的亮度被32个等分点平均分为32段亮度区间，每一段亮度区间对应一个亮度权重。比如0-7的亮度区间的权重是luma_wgt[0]，8-15的亮度区间的权重是luma_wgt[1]，依次类推。如图3-7所示。值越大，图像锐化程度越高，反之，越弱。 取值范围：[0, 127]，建议值127。 #define HI_ISP_AUTO_ISO_NUM 16 #define HI_ISP_SHARPEN_LUMA_NUM 32
texture_strength[HI_ISP_SHARPEN_GAIN_NUM][HI_ISP_AUTO_ISO_NUM]	该参数用于设置图像无方向的细节纹理的锐度。 该值越大，无方向的细节纹理的清晰度越高。该参数是一个HI_ISP_SHARPEN_GAIN_NUM=32的数组，是一个32段的连续的强度曲线，如图3-8所示。 取值范围：[0, 4095]，建议值300。 #define HI_ISP_SHARPEN_GAIN_NUM 32
edge_strength[HI_ISP_SHARPEN_GAIN_NUM][HI_ISP_AUTO_ISO_NUM]	该参数用于设置图像带方向的边缘的锐度。 该值越大，带方向的边缘的锐度越高。该参数是一个HI_ISP_SHARPEN_GAIN_NUM=32的数组，是一个32段的连续的强度曲线，如图3-9所示。取值范围：[0, 4095]，建议值400。

成员名称	描述
texture_freq[HI_ISP_AUTO_ISO_NUM]	该参数用于设置图像的无方向细节纹理的增强频率。 该值越大，细节纹理的增强就越偏向于高频增强，细节纹理就越细碎。反之，该值越小，细节纹理就越粗越圆润。 texture_freq对应于强度参数texture_strength。 texture_freq越大，图像的细节纹理就越细碎，该值过大，会导致图像的细节纹理过于细碎而不自然，甚至感觉模糊。 取值范围：[0, 4095]，建议值128。
edge_freq[HI_ISP_AUTO_ISO_NUM]	该参数用于控制图像的带方向的边缘增强频段。 该值越大，边缘的增强就越偏向于高频增强，图像的边缘就越纤薄越窄。反之，该值越小，边缘就越粗越圆润。 edge_freq对应于强度参数edge_strength。edge_freq越大，图像的边缘就越纤薄越窄，该值过大，会导致图像的边缘过于纤薄而出现虚边现象。 取值范围：[0, 4095]，建议值96。
over_shoot[HI_ISP_AUTO_ISO_NUM]	设置图像的overshoot（锐化后的白边白点）的强度。 该值越小，锐化后的白边白点越弱，清晰度也会下降。该值过小，图像会呈油画效果。 取值范围：[0, 127]，建议值100。
under_shoot[HI_ISP_AUTO_ISO_NUM]	设置图像的undershoot（锐化后的黑边黑点）的强度。 该值越小，锐化后的黑边黑点越弱，清晰度也会下降。该值过小，图像会呈油画效果。 取值范围：[0, 127]，建议值127。
shoot_sup_strength[HI_ISP_AUTO_ISO_NUM]	该参数用于设置图像锐化后的overshoot和undershoot的抑制强度。用于在保证清晰度不明显下降的前提下，抑制锐化后的图像的overshoot和undershoot的宽度和幅度。 该值越大，锐化后的图像的overshoot和undershoot的宽度越窄、强度越小。该值变大，理论上不会影响图像的清晰度，只是黑白边变窄以后，会减弱人眼的锐度感受。 该参数需要和shoot_sup_adj配合使用。 取值范围：[0, 255]，建议值8。
shoot_sup_adj[HI_ISP_AUTO_ISO_NUM]	该参数用于对图像锐化后的overshoot和undershoot的抑制强度进行调节。 该参数配合shoot_sup_strength使用，用于调节shoot_sup_strength作用的区域范围。该值越小，则越多的纹理区域的shoot会被shoot_sup_strength抑制；该值越大，则只有很强的边缘的shoot会被shoot_sup_strength抑制，纹理区域的shoot不会被抑制。 取值范围：[0, 15]，建议值9。

成员名称	描述
detail_ctrl[HI_ISP_AUTO_ISO_NUM]	该参数用于控制图像的细节纹理区域的shoot强度，shoot越大，细节纹理区的清晰度越高。 - 若该值等于128，则图像的细节纹理区域的shoot强度和大边缘的shoot强度一致，都分别等于over_shoot和under_shoot。 - 若该值大于128，则图像的细节纹理的shoot强度大于大边缘，大边缘的shoot强度分别等于over_shoot 和 under_shoot。 - 若该值小于128，则图像的细节纹理的shoot强度小于大边缘，大边缘的shoot强度分别等于over_shoot和 under_shoot。 取值范围：[0, 255]。
detail_ctrl_threshold[HI_ISP_AUTO_ISO_NUM]	图像的细节纹理区的shoot强度的控制阈值。 该值配合 detail_ctrl使用，用于区分detail_ctrl所控制shoot的纹理区和边缘，也即纹理区和边缘的区分阈值。小于该值的区域为纹理区，该纹理区域的shoot会被detail_ctrl单独控制，而大于该阈值的边缘区域的shoot依然等于 over_shoot和under_shoot。 取值范围：[0, 0xFF]，建议值0xA0。
edge_filt_strength[HI_ISP_AUTO_ISO_NUM]	边缘滤波强度调试参数：实现图像锐化边缘的范围和边缘平滑强度的控制。 该值越大，判为边缘的区域越多、也越宽，edge_strength起作用的图像边缘就越多，而且，该值越大，沿着边缘方向的平滑滤波强度也越大，边缘就越平滑。反之，判为边缘的区域越少、也越窄，edge_strength 起作用的图像区域越少，边缘平滑就越弱。 取值范围：[0, 63]，建议值 53。
edge_filt_max_cap[HI_ISP_AUTO_ISO_NUM]	边缘滤波强度范围的调试参数：该值越大，边缘滤波的最大强度也最大，edge_filt_strength的可调试范围也越大；一般建议该值大小控制30以内。 取值范围：[0, 47]；建议值 18。
r_gain[HI_ISP_AUTO_ISO_NUM]	深红色区域的锐化增益控制参数。该值越大，则深红色区域的锐化强度越大。 取值范围：[0, 31]；建议值 28。
g_gain[HI_ISP_AUTO_ISO_NUM]	绿色区域的锐化增益控制参数。该值越大，则绿色区域的锐化强度越大。 取值范围：[0, 255]；建议值 32。
b_gain[HI_ISP_AUTO_ISO_NUM]	深蓝色区域的锐化增益控制参数。该值越大，则深蓝色区域的锐化强度越大。 取值范围：[0, 31]；建议值 28。

成员名称	描述
skin_gain[HI_ISP_AUTO_ISO_NUM]	肤色区域的锐化增益控制参数。该值越大，则肤色区域的锐化强度越大。 取值范围：[0, 31]；建议值 23。
max_sharp_gain[HI_ISP_AUTO_ISO_NUM]	图像锐化的最大增益限制值。该值越大，图像的锐化幅度越大，反之，锐化幅度越小。适当的调小该参数，可以减少图像的过锐化，可以减少图像锐化后的黑白点。 取值范围：[0, 0x7FF]，建议值 0xA0。

注意事项

Auto 档参数分别对应sensor在16种不同增益（ $a\_gain \times d\_gain \times isp\_d\_gain(times)$ ）情况下的设置值，成员参数的含义与Manual档cmos成员参数一致。

16 种增益为：100、200、400、800、1600、3200、6400、12800、25600、51200、102400、204800、409600、819200、1638400、3276800。

3.29.3.3 hi_isp_sharpen_attr

说明

定义图像锐化的属性。

定义

```
typedef struct {  
    hi_bool enable;  
    hi_u8 skin_umin;  
    hi_u8 skin_vmin;  
    hi_u8 skin_umax;  
    hi_u8 skin_vmax;  
    hi_op_mode op_type;  
    hi_isp_sharpen_manual_attr manual_attr;  
    hi_isp_sharpen_auto_attr auto_attr;  
} hi_isp_sharpen_attr;
```

成员

成员名称	描述
enable	该参数用于控制是否使能图像锐化功能。 - HI_FALSE：关闭 - HI_TRUE：使能
skin_umin	肤色区域范围矩形窗的左下角的最小坐标的 U 值，取值范围：[0, 255]。
skin_vmin	肤色区域范围矩形窗的左下角的最小坐标的 V 值，取值范围：[0, 255]。

成员名称	描述
skin_umax	肤色区域范围矩形窗的右上角的最大坐标的 U 值，取值范围：[0, 255]。
skin_vmax	肤色区域范围矩形窗的右上角的最大坐标的 V 值，取值范围：[0, 255]。
op_type	Sharpen 工作类型。 - HI_OP_MODE_AUTO：自动 - HI_OP_MODE_MANUAL：手动
manual_attr	Sharpen 手动参数，详细请参见 3.29.3.1 hi_isp_sharpen_manual_attr 。
auto_attr	Sharpen 自动参数，详细请参见 3.29.3.2 hi_isp_sharpen_auto_attr 。

3.30 Dehaze

3.30.1 hi_mpi_isp_get_dehaze_attr

函数功能

获取 Dehaze 属性。

函数原型

hi_s32 hi_mpi_isp_get_dehaze_attr([hi_vi_pipe](#) vi_pipe, [hi_isp_dehaze_attr](#) *dehaze_attr)

参数说明

参数名	输入/输出	说明
vi_pipe	输入	VI PIPE号。 取值范围：[0, 12)。
dehaze_attr	输入	去雾属性。

返回值说明

- 0：成功
- 非0：失败，参见[ISP返回码](#)

3.30.2 hi_mpi_isp_set_dehaze_attr

函数功能

设置 Dehaze 属性。

函数原型

```
hi_s32 hi_mpi_isp_set_dehaze_attr(hi_vi_pipe vi_pipe, const hi_isp_dehaze_attr *dehaze_attr)
```

参数说明

参数名	输入/输出	说明
vi_pipe	输入	VI PIPE号。 取值范围：[0, 12)。
dehaze_attr	输入	去雾属性。

返回值说明

- 0：成功
- 非0：失败，参见[ISP返回码](#)。

3.30.3 数据类型

3.30.3.1 hi_isp_dehaze_manual_attr

定义去雾手动模式属性。

定义

```
typedef struct {  
    hi_u8 strength;  
} hi_isp_dehaze_manual_attr;
```

成员

成员名称	描述
strength	手动模式下去雾强度。 取值范围：[0, 255]

3.30.3.2 hi_isp_dehaze_auto_attr

说明

定义去雾自动模式属性。

定义

```
typedef struct {  
    hi_u8 strength;  
} hi_isp_dehaze_auto_attr;
```

成员

成员名称	描述
strength	自动模式下去雾强度。 取值范围：[0, 255]

3.30.3.3 hi_isp_dehaze_attr

说明

定义 ISP 去雾属性。

定义

```
typedef struct {  
    hi_bool enable;  
    hi_bool user_lut_enable;  
    hi_u8 dehaze_lut[HI_ISP_DEHAZE_LUT_SIZE];  
    hi_op_mode op_type;  
    hi_isp_dehaze_manual_attr manual_attr;  
    hi_isp_dehaze_auto_attr auto_attr;  
    hi_u16 tmprflt_incr_coef;  
    hi_u16 tmprflt_decr_coef;  
} hi_isp_dehaze_attr;
```

成员

成员名称	描述
enable	使能 Dehaze 功能。 取值范围：HI_FALSE：关闭， HI_TRUE：使能。
user_lut_enable	用户自定义去雾曲线使能。
dehaze_lut[HI_ISP_DEHAZE_LUT_SIZE]	用户使用自定义的曲线参数，曲线是根据亮度做去雾的强度，一共有256个点，也就是亮度区分256阶。 #define HI_ISP_DEHAZE_LUT_SIZE 256

成员名称	描述
op_type	DEHAZE工作类型。 - HI_OP_MODE_AUTO：自动模式； - HI_OP_MODE_MANUAL：手动模式。
manual_attr	手动模式，详细请参见 3.30.3.1 hi_isp_dehaze_manual_attr 。
auto_attr	自动模式，详细请参见 3.30.3.2 hi_isp_dehaze_auto_attr 。
tmprflt_incr_coef	时域滤波递增系数，值越大，画面由暗到亮收敛速度越快。 取值范围：[0x0, 0x80]。
tmprflt_decr_coef	时域滤波递减系数，值越大，画面由亮到暗收敛速度越快。 取值范围：[0x0, 0x80]。

注意事项

去雾使能后，为了避免画面异常，延迟两帧生效。

3.31 Auto Flicker Type Detection

3.31.1 功能描述

Auto Flicker Type Detection（自动工频闪类型检测）是一个结合软件和硬件一起实现的算法。该算法能够计算出当前环境光源的频闪频率是50Hz还是60Hz(当前版本只支持50HZ检测)。

须知：

- 在某些环境中，当存在有50Hz光源和60Hz光源来回变动的情况时（比如在室内50Hz的灯光下，用户可能会关闭灯光，打开60Hz光源的电视），Camera无法应用合适的抗闪参数。所以此时需要检测出当前环境光源的工频闪频率，Camera根据当前环境光源的工频闪频率适配合适的抗闪参数。
- 工频闪类型计算建议在vpss节点获取图像数据进行计算，前端的DIS、LDC、鱼眼校正等模块如果对图像的内容形态改变过大会影响工频闪类型计算的准确性。另外图像噪声过大，也会影响工频闪类型计算的准确性。
- 输入的图像数据须存在频闪效果，否则无法检测出光源的频闪频率

3.31.2 hi_mpi_isp_calc_flicker_type

函数功能

输入帧信息，获取工频闪类型检测的结果。

函数原型

hi_s32 hi_mpi_isp_calc_flicker_type(**hi_vi_pipe** vi_pipe, **hi_isp_calc_flicker_input** *input_param, **hi_isp_calc_flicker_output** *output_param, **hi_video_frame_info** frame[], **hi_u32** array_size)

约束说明

- 必须使用连续的3帧数据进行计算，输入的array_size帧数需要等于3。同时frame[]指向的数据必须保证有效。
- 计算的帧数据的分辨率不能超过ISP模块最大能够处理的图像分辨率。
- 为了加快计算速度，接口内部使用VGS硬件模块参与计算，所以用于计算工频闪的图像分辨率不能超过VGS的最大处理分辨率范围。
- 在不同频闪混合的光源下，比如光源环境同时存在50Hz和60Hz的光源，或者工频闪光源只占整个环境光源非常小的一部分，容易出现误判。
- 可以参考附录工频闪类型自适应Sample用例处理流程。

参数说明

参数名	输入/输出	说明
vi_pipe	输入	VI PIPE号。 取值范围：[0, 12)。
input_param	输入	自动计算工频闪输入相关参数指针。
output_param	输出	自动计算工频闪输出结果参数指针。
frame[]	输入	输入需要计算的连续帧数据。
array_size	输入	输入需要计算连续帧的个数。

返回值说明

- 0：成功
- 非0：失败，参见[ISP返回码](#)

参考样例

```
hi_vi_pipe vi_pipe = 0;
hi_u16 frame_num = 3;
hi_video_frame_info frame[3];
hi_isp_calc_flicker_input input_param;
hi_isp_calc_flicker_output output_param;
hi_isp_exp_info exp_info;
hi_mpi_isp_query_exposure_info(vi_pipe, &exp_info);
input_param.lines_per_second = exp_info.lines_per500ms*2
... ..
获取3帧连续帧数据给 frame[3](此处略)
... ..
ret = hi_mpi_isp_calc_flicker_type(vi_pipe, input_param, output_param, frame, frame_num);
```

3.31.3 数据类型

3.31.3.1 hi_isp_calc_flicker_input

说明

定义计算工频闪输入参数。

定义

```
typedef struct {  
    hi_u32 lines_per_second;  
} hi_isp_calc_flicker_input;
```

成员

成员名称	描述
lines_per_second	当前序列1秒的最大曝光总行数。 取值范围：[0x1F4,0xFFFFFFFF]。默认值0

注意事项

- lines_per_second等于1秒的最大总曝光行数，比如一帧的最大曝光时间为1125行，帧率为30fps，那么lines_per_second = 1125*30。通常，可以使用hi_isp_exp_info这个结构体成员变量lines_per500ms乘以 2。
- 需要注意这里的1行时间需要和1行数据的Readout Timing的时间相等，一般sensor的曝光行数单位时间和Readout 1行的时间相同，需要注意少部分sensor的Readout 1行时间等于2行，那么计算lines_per_second时需要除以2。其他等效倍率则除以相应的倍率。

3.31.3.2 hi_isp_calc_flicker_output

说明

定义自动工频闪类型检测输出结果。

定义

```
typedef struct {  
    hi_isp_calc_flicker_type flicker_type; /*The calculate result of flicker type*/  
} hi_isp_calc_flicker_output;
```

成员

成员名称	描述
flicker_type	hi_isp_calc_flicker_type: 工频闪计算结果。

3.31.3.3 hi_isp_calc_flicker_type

说明

自动工频闪检测结果类型

定义

```
typedef enum {  
    HI_ISP_FLICKER_TYPE_NONE = 0,  
    HI_ISP_FLICKER_TYPE_UNKNOW,  
    HI_ISP_FLICKER_TYPE_50HZ,  
    HI_ISP_FLICKER_TYPE_60HZ,  
    HI_ISP_FLICKER_TYPE_BUTT,  
} hi_isp_calc_flicker_type;
```

成员

成员名称	描述
HI_ISP_FLICKER_TYPE_NONE	无工频闪。
HI_ISP_FLICKER_TYPE_UNKNOW	有工频闪，但不确定是哪种类型。
HI_ISP_FLICKER_TYPE_50HZ	50HZ的工频闪。
HI_ISP_FLICKER_TYPE_60HZ	60HZ的工频闪（当前版本不支持）。
HI_ISP_FLICKER_TYPE_BUTT	预留值。

3.32 MotionFusion

3.32.1 hi_mpi_mfusion_set_attr

函数功能

设置motionfusion属性。

函数原型

```
hi_s32 hi_mpi_mfusion_set_attr(const hi_u32 fusion_id, const hi_mfusion_attr  
*mfusion_attr)
```

约束说明

- 调用该接口前，必须调用hi_mpi_sys_init进行系统初始化。

参数说明

参数名	输入/输出	说明
fusion_id	输入	fusion设备ID号。 取值范围：0（目前fusion_id仅有0有效）。
mfusion_attr	输入	motionfusion属性指针。

返回值说明

- 0：成功
- 非0：失败，参见[ISP返回码](#)

3.32.2 数据类型

3.32.2.1 hi_mfusion_steady_detect_attr

说明

定义IMU噪声和静止检测灵敏度参数类型。

定义

```
typedef struct {  
    hi_u32 steady_time_thr;  
    hi_s32 gyro_offset;  
    hi_s32 acc_offset;  
    hi_s32 gyro_rms;  
    hi_s32 acc_rms;  
    hi_s32 gyro_offset_factor;  
    hi_s32 acc_offset_factor;  
    hi_s32 gyro_rms_factor;  
    hi_s32 acc_rms_factor;  
} hi_mfusion_steady_detect_attr;
```

成员

成员名称	描述
steady_time_thr	静止状态时间阈值，单位为秒。连续检测到超过此配置时长的静止状态才启动在线温飘标定。 范围: [0, (1<<16)-1] 默认值: 3（录像机）或1（DV）

成员名称	描述
gyro_offset	GYRO零偏典型值（取绝对值，为正数），从数据手册中获取（Max value of Initial ZRO Tolerance），单位为dps (degree per second), 精度同GYRO数据精度，即ADC WORD LENGTH -1。 范围: [0, 100 * (1<<15)] 默认值: 327680
acc_offset	ACC零偏典型值（取绝对值，为正数），从数据手册中获取（Max value of Initial ZRO Tolerance），单位为g（一个重力加速度），精度同ACC数据精度，即ADC WORD LENGTH -1。 范围: [0, 0.5 * (1<<15)] 默认值: 3276
gyro_rms	GYRO在当前低通滤波带宽下的RMS噪声典型值，可从数据手册中Rate Noise Spectral Density和当前配置低通滤波带宽计算得出。单位为dps-rms, 精度同GYRO数据精度，即ADC WORD LENGTH -1。 范围: [0, 0.5 * (1<<15)] 默认值: 1769
acc_rms	ACC在当前低通滤波带宽下的RMS噪声典型值，可从数据手册中Rate Noise Spectral Density和当前配置低通滤波带宽计算得出。单位为g-rms, 精度同ACC数据精度，即ADC WORD LENGTH -1。 范围: [0, 0.005 * (1<<15)] 默认值: 44
gyro_offset_factor	静止判断使用gyro_offset的灵敏度（放大倍数），值越大，对运动越不敏感。精度为4bit。 范围: [0, 1000 * (1<<4)] 默认值: 32
acc_offset_factor	静止判断使用acc_offset的灵敏度（放大倍数），值越大，对运动越不敏感。精度为4bit。 范围: [0, 1000 * (1<<4)] 默认值: 32
gyro_rms_factor	静止判断使用gyro_rms的灵敏度（放大倍数），值越大，对运动越不敏感。精度为4bit。 范围: [0, 1000 * (1<<4)] 默认值: 128（录像机）或 200（DV）
acc_rms_factor	静止判断使用acc_rms的灵敏度（放大倍数），值越大，对运动越不敏感。精度为4bit。 范围: [0, 1000 * (1<<4)] 默认值: 160（录像机）或1600（DV）

3.32.2.2 hi_mfusion_attr

说明

定义motionfusion属性。

定义

```
typedef struct {  
    hi_u32 device_mask;  
    hi_u32 temperature_mask;  
    hi_mfusion_steady_detect_attr steady_detect_attr;  
} hi_mfusion_attr;
```

成员

成员名称	描述
device_mask	设备属性掩码。
temperature_mask	温度计掩码。
steady_detect_attr	IMU噪声和静止检测灵敏度参数。

3.33 统计信息

3.33.1 概述

ISP提供的3A统计信息及相关配置。

3.33.2 AE 统计信息

实现自动曝光信息的统计，软件根据统计信息调节sensor从而实现自动曝光的功能。AE统计信息包含分区间的 R/Gr/Gb/B 均值统计、带权重的全局 R/Gr/Gb/B 均值、1024 段直方图信息。

3.33.2.1 AE 区域统计信息

AE按支持 17×15 分块进行统计，每个分块均可以输出16bit R，Gr， Gb， B 均值 (Average R/Gr/Gb/B)。以 R、Gr 为例，区块的统计信息读取地址与数据的对应关系如下表所示。m 表示水平分区个数，n 表示垂直分区个数。

表 3-9 AE 区域统计信息

Memory address	Zone	Memory Read Data	
		[31:16]	[15:0]
0	0	Average R	Average Gr

Memory address	Zone	Memory Read Data	
		[31:16]	[15:0]
1	1	Average R	Average Gr
2	2	Average R	Average Gr
3	3	Average R	Average Gr
.....			
m*n-3	m*n-3	Average R	Average Gr
m*n-2	m*n-2	Average R	Average Gr
m*n-1	m*n-1	Average R	Average Gr

3.33.2.2 带权重的 AE 全局统计信息

与 AE 的区域统计信息基本一致，对于整幅图像，AE 也提供了 4 个全局统计信息。分别为 HI_ISP_AE_TOTAL_R_AVER，HI_ISP_AE_TOTAL_GR_AVER，HI_ISP_AE_TOTAL_GB_AVER，HI_ISP_AE_TOTAL_B_AVER。其物理意义与分块统计一致。

3.33.2.3 带权重的 AE 直方图信息

AE 提供 1024 阶直方图统计 0-1023，直方图信息读取地址与数据的对应关系如下表所示。

表 3-10 AE 1024 段直方图统计信息

Memory address	Zone	Memory Read Data	
		[31:29]	[28:0]
0	0	0	Pixel Value 为 0 的个数
1	1	0	Pixel Value 为 1 的个数
2	2	0	Pixel Value 为 2 的个数
3	3	0	Pixel Value 为 3 的个数
.....			
1022	1022	0	Pixel Value 为 1022 的个数
1023	1023	0	Pixel Value 为 1023 的个数

AE 支持 FourPlaneMode 模式，在 FourPlaneMode 模式下，AE 提供四通道 256 段直方图统计 0~255，直方图信息读取地址与数据的对应关系如下表所示。

表 3-11 AE 256 段直方图统计信息

Memory address	Zone	Memory Read Data	
		[31:29]	[28:0]
0	0	0	B 通道 Pixel Value 为 0 的个数
1	1	0	B 通道 Pixel Value 为 1 的个数
.....			
255	255	0	B 通道 Pixel Value 为 255 的个数
256	256	0	Gb 通道 Pixel Value 为 0 的个数
.....			
511	511	0	Gb 通道 Pixel Value 为 255 的个数
512	512	0	Gr 通道 Pixel Value 为 0 的个数
.....			
767	767	0	Gr 通道 Pixel Value 为 255 的个数
768	768	0	R 通道 Pixel Value 为 0 的个数
.....			
1022	1022	0	R 通道 Pixel Value 为 254 的个数
1023	1023	0	R 通道 Pixel Value 为 255 的个数

3.33.3 MG 统计信息

MG模块统计 DRC 后分块均值，与 AE 统计分块均值相比，可以得出分块均值增益最大值。MG 统计信息包含全局分区间 R/Gr/Gb/B 均值统计。

3.33.3.1 MG 区域统计信息

MG按 17×15 分块统计，每个分块均可以输出 8bit R，Gr，Gb，B 均值(Average R/Gr/Gb/B)。ISP统计算法内部区块的统计信息读取地址与数据的对应关系如下表所示。m 表示水平分区个数，n 表示垂直分区个数。

表 3-12 MG 区域统计信息

Memory address	Zone	Memory Read Data			
		[31:24]	[23:16]	[15:8]	[7:0]
0	0	Average R	Average Gr	Average Gb	Average B
1	1	Average R	Average Gr	Average Gb	Average B

Memory address	Zone	Memory Read Data			
		[31:24]	[23:16]	[15:8]	[7:0]
2	2	Average R	Average Gr	Average Gb	Average B
3	3	Average R	Average Gr	Average Gb	Average B
.....					
m*n-3	m*n-3	Average R	Average Gr	Average Gb	Average B
m*n-2	m*n-2	Average R	Average Gr	Average Gb	Average B
m*n-1	m*n-1	Average R	Average Gr	Average Gb	Average B

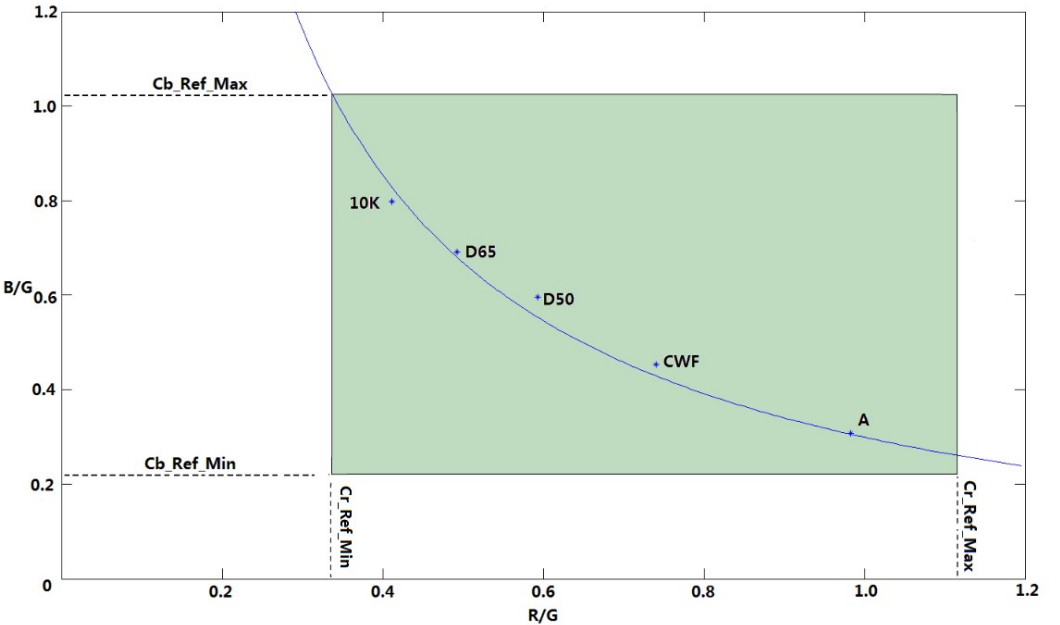
3.33.4 AWB 统计信息

AWB 统计信息包含全局统计信息和区域统计信息。

- 全局统计信息：整幅图像的 R、G、B 均值，以及有效统计点的个数。
- 区域统计信息：最大支持图像的 32x32 分块，每个分块均可以输出 R、G、B 均值，以及有效统计点的个数。

AWB 中统计的有效点，除了要满足配置的 RGB 上限与下限要求外（black_level, white_level），像素点颜色还需要满足配置的有效像素颜色空间，具体如下图所示，绿色区块内的为有效像素的颜色范围。

图 3-10 AWB 有效像素颜色空间



AWB 统计的有效像素区域是一个四边形，其中上下左右四个边界由 cr_ref_min/max, cb_ref_min/max 这 4 个参数决定，它们代表了允许的 r/g, b/g 的最大与最小值。

3.33.4.1 AWB 区域统计信息

AWB中最大支持图像的 32x32 分块，每个分块均可以输出 R、G、B 均值(Average R/G/B)。其中统计的像素点均为 16bit，分块内统计有效点个数 Count All，点数统计值均已根据图像大小做了归一化处理，归一化到 16bit。同时，每一个分块支持分亮度区间进行统计，最大支持分 4 个亮度区间。

须知

- 当分辨率较小，分块数较多时，如果每块的分辨率小于 60x14，那么需要减少分块数，以免每块过小。建议在分辨率小的情况下，使用较少的分块数。
- 如果要输出低分辨率，需要在业务启动的时候设置一个比较小的 AWB 统计信息分块数。

区块的统计信息读取地址与数据的对应关系下表所示。

表 3-13 AWB 区域统计信息

Memory address	Zone	Memory Read Data	
		[31:16]	[15:0]
0	0	Average G	Average R
1	0	Count All	Average B
2	1	Average G	Average R
3	1	Count All	Average B
.....			
2047	1023	Count All	Average B

3.33.4.2 AWB 全局统计信息

与 AWB 的区域统计信息基本一致，对于整幅图像，AWB 也提供了四个全局统计信息。分别为 ISP_AWB_AVG_R/G/B，ISP_AWB_CNT_ALL。其物理意义与分块统计一致。

3.33.5 API 参考

3.33.5.1 hi_mpi_isp_set_stats_cfg

函数功能

设置ISP 3A统计信息配置。

函数原型

hi_s32 hi_mpi_isp_set_stats_cfg(**hi_vi_pipe** vi_pipe, const **hi_isp_stats_cfg** *stat_cfg)

约束说明

- 用户可以关闭不需要的统计信息，以提高系统性能。
- 该接口必须在调用**hi_mpi_isp_init**接口之后才能调用。
- 如果要变更ae统计算法配置，图像宽度不能小于256，图像高度不能小于120。

参数说明

参数名	输入/输出	说明
vi_pipe	输入	VI PIPE号。 取值范围：[0, 12)。
stat_cfg	输入	ISP 统计信息配置。

返回值说明

- 0：成功
- 非0：失败，参见**ISP返回码**。

3.33.5.2 hi_mpi_isp_get_stats_cfg

函数功能

获取 ISP 3A 统计信息配置。

函数原型

hi_s32 hi_mpi_isp_get_stats_cfg(**hi_vi_pipe** vi_pipe, **hi_isp_stats_cfg** *stat_cfg)

约束说明

该接口必须在调用**hi_mpi_isp_init**接口之后才能调用。

参数说明

参数名称	描述	输入/输出
vi_pipe	VI PIPE 号。	输入
stat_cfg	ISP 统计信息配置。	输出

3.33.5.3 hi_mpi_isp_get_ae_stats

函数功能

获取 AE 统计信息。

函数原型

```
hi_s32 hi_mpi_isp_get_ae_stats(hi_vi_pipe vi_pipe, hi_isp_ae_stats *ae_stat)
```

约束说明

该接口必须在调用[hi_mpi_isp_init](#)接口之后才能调用。

参数说明

参数名	输入/输出	说明
vi_pipe	输入	VI PIPE号。 取值范围：[0, 12)。
ae_stat	输入	AE 统计信息。

返回值说明

- 0：成功
- 非0：失败，参见[ISP返回码](#)。

3.33.5.4 hi_mpi_isp_get_mg_stats

函数功能

获取 MG 统计信息。

函数原型

```
hi_s32 hi_mpi_isp_get_mg_stats(hi_vi_pipe vi_pipe, hi_isp_mg_stats *mg_stat)
```

约束说明

该接口必须在调用[hi_mpi_isp_init](#)接口之后才能调用。

参数说明

参数名	输入/输出	说明
vi_pipe	输入	VI PIPE号。 取值范围：[0, 12)。
mg_stat	输出	MG 统计信息。

返回值说明

- 0：成功
- 非0：失败，参见ISP返回码。

3.33.5.5 hi_mpi_isp_get_wb_stats

函数功能

获取WB统计信息。

函数原型

```
hi_s32 hi_mpi_isp_get_wb_stats(hi_vi_pipe vi_pipe, hi_isp_wb_stats *wb_stat)
```

约束说明

该接口必须在调用hi_mpi_isp_init接口之后才能调用。

参数说明

参数名	输入/输出	说明
vi_pipe	输入	VI PIPE号。 取值范围：[0, 12)。
wb_stat	输出	WB 统计信息。

返回值说明

- 0：成功
- 非0：失败，参见ISP返回码。

3.33.6 数据类型

3.33.6.1 hi_isp_stats_cfg

说明

定义 ISP 3A 统计信息配置。

定义

```
typedef struct {  
    hi_isp_stats_ctrl    key;  
    hi_isp_ae_stats_cfg  ae_cfg;  
    hi_isp_wb_stats_cfg  wb_cfg;  
}
```

```
hi_isp_focus_stats_cfg focus_cfg;
} hi_isp_stats_cfg;
```

成员

成员名称	描述
key	统计信息使能。
ae_cfg	AE 统计信息配置。
wb_cfg	AWB 统计信息配置。
focus_cfg	AF 统计信息配置。

3.33.6.2 hi_isp_stats_ctrl

说明

定义 ISP 3A 统计信息使能。

定义

```
typedef union {
    hi_u64 key;
    struct {
        hi_u64 bit1_fe_ae_global_stat : 1; /* [0] */
        hi_u64 bit1_fe_ae_local_stat : 1; /* [1] */
        hi_u64 bit1_fe_ae_stitch_global_stat : 1; /* [2] */
        hi_u64 bit1_fe_ae_stitch_local_stat : 1; /* [3] */
        hi_u64 bit1_be_ae_global_stat : 1; /* [4] */
        hi_u64 bit1_be_ae_local_stat : 1; /* [5] */
        hi_u64 bit1_be_ae_stitch_global_stat : 1; /* [6] */
        hi_u64 bit1_be_ae_stitch_local_stat : 1; /* [7] */
        hi_u64 bit1_awb_stat1 : 1; /* [8] */
        hi_u64 bit1_awb_stat2 : 1; /* [9] */
        hi_u64 bit2_reserved0 : 2; /* [10:11] */
        hi_u64 bit1_fe_af_stat : 1; /* [12] */
        hi_u64 bit1_be_af_stat : 1; /* [13] */
        hi_u64 bit2_reserved1 : 2; /* [14:15] */
        hi_u64 bit1_dehaze : 1; /* [16] */
        hi_u64 bit1_mg_stat : 1; /* [17] */
        hi_u64 bit14_reserved : 14; /* [18:31] */
        hi_u64 bit32_isr_access : 32; /* [32:63] */
    };
} hi_isp_stats_ctrl;
```

成员

成员名称	描述
bit1_fe_ae_global_stat	位于 FE 的 AE 全局统计信息使能。包括直方图和全局均值。
bit1_fe_ae_local_stat	位于 FE 的 AE 分块均值统计信息使能。
bit1_fe_ae_stitch_global_stat	拼接后位于 FE 的 AE 全局统计信息使能。包括直方图和全局均值。仅在拼接模式下生效。

成员名称	描述
bit1_fe_ae_stitch_local_stat	拼接后位于 FE 的 AE 分块均值统计信息使能。仅在拼接模式下生效。
bit1_be_ae_global_stat	位于 BE 的 AE 全局统计信息使能。包括直方图和全局均值。
bit1_be_ae_local_stat	位于 BE 的 AE 分块均值统计信息使能。
bit1_be_ae_stitch_global_stat	拼接后位于 BE 的 AE 全局统计信息使能。包括直方图和全局均值。仅在拼接模式下生效。
bit1_be_ae_stitch_local_stat	拼接后位于 BE 的 AE 分块均值统计信息使能。仅在拼接模式下生效。
bit1_awb_stat1	Bayer 域 AWB 全局统计信息使能。
bit1_awb_stat2	Bayer 域 AWB 分区间统计信息使能。
bit2_reserved0	保留位。
bit1_fe_af_stat	位于 FE 的 AF 统计信息使能。
bit1_be_af_stat	位于 BE 的 AF 统计信息使能。
bit2_reserved1	保留位。
bit1_dehaze	Dehaze 统计信息使能。
bit1_mg_stat	MG 统计信息使能。
bit14_reserved	保留位。
bit32_isr_access	中断服务程序中更新统计信息开关。

注意事项

- 用户可以关闭不使用的统计信息，如此就不会读相应统计信息，从而提高系统性能。
- bit1_fe_ae_global_stat~ bit14_reserved 32个bit 为使用hi_mpi_isp_get_xx_stats 接口（xx 表示 AE/AWB/AF）获取统计信息时相应的开关。isp firmware运转需要获取统计信息供昇腾相关算法使用。
- 如果用户使用hi_mpi_isp_get_xx_stats（xx 表示 AE/AWB/AF）实现自己的3A算法，可以通过disable bit32_isr_access中相应的 bit来关掉中断服务程序中相关统计信息的更新以达到降低CPU占用率的目的。bit32_isr_access中bit顺序定义也是 bit1_fe_ae_global_stat~ bit14_reserved。
例如：可以置零bit32_isr_access 的 bit0来停止更新fe_ae_global_stat统计信息。

3.33.6.3 hi_isp_ae_stats_cfg

说明

定义 AE 统计信息配置。

定义

```
typedef struct {  
    hi_isp_ae_switch      ae_switch;  
    hi_isp_ae_hist_config hist_config;  
    hi_isp_ae_four_plane_mode four_plane_mode;  
    hi_isp_ae_stat_mode   hist_mode;  
    hi_isp_ae_stat_mode   aver_mode;  
    hi_isp_ae_stat_mode   max_gain_mode;  
    hi_isp_ae_crop        crop;  
    hi_u8 weight[HI_ISP_AE_ZONE_ROW][HI_ISP_AE_ZONE_COLUMN];  
} hi_isp_ae_stats_cfg;
```

成员

成员名称	描述
ae_switch	WDR 合成后 AE 统计模块在 ISP pipeline 的位置，默认值为 1。
hist_config	全局直方图统计时的采样点方式配置。
four_plane_mode	four_plane_mode 使能模式，默认值为 0，使能时 1024 段直方图为BGGR 四通道 256 段直方图。
hist_mode	全局 1024 段直方图开方模式。仅影响BE 直方图统计信息。
aver_mode	均值开方模式，仅影响 BE 均值统计信息。
max_gain_mode	MG 模块开方模式。仅影响 MG 模块统计信息。为了与 AE 分块统计信息进行比较，建议与 aver_mode 配置为同一模式。
crop	AE 输入图像裁剪配置。
weight	15x17 个区间的 AE 权重表，取值范围为[0, 15]。 #define HI_ISP_AE_ZONE_COLUMN 17 #define HI_ISP_AE_ZONE_ROW 15

注意事项

- 线性模式数据流走的是 channel 1，默认使用全局 1024 段直方图进行 AE 控制，此时 channel 2/3/4 WDR 合成前全局 1024 段直方图全为 0。
- 全局 4 分量统计平均值会受 15*17 权重表的影响。
- Channel 1/2/3/4 全局 1024 段直方图会受 15*17 权重表的影响，WDR 模式可以使用这几路直方图信息进行 AE 控制。如 2 合 1 可使用 channel 1/2，3 合 1 可使用 channel 1/2/3，4 合 1 可使用 channel 1/2/3/4 等。
- 配置 ae_switch/hist_config/four_plane_mode 相关参数时，需要把昇腾 AE 的大面积单色判断功能关闭，关闭方式参考hi_isp_exposure_attr的 hist_stat_adjust属性。
- 自动曝光的静态统计信息分为 15*17 个区域，可通过设定权重表改变每个区域的权重。如可使中心区域的权重加大，则中心区域的亮度变化会使图像的全局直方图统计信息产生更多的变化。该属性可调节感兴趣区域的曝光权重，可用于实现指定区域的背光补偿。若能确定具体产品形态感兴趣区域的位置，可将这些位置的权重加大。如对于运动 DV 和行车记录仪，感兴趣区域一般在画面的中下部分，那么可适当加大场景中下部分区域的权重，使得感兴趣区域得到合理曝光。

3.33.6.4 hi_isp_ae_switch

说明

定义 AE 统计模块在 ISP pipeline 的位置。

定义

```
typedef enum {  
    HI_ISP_AE_AFTER_DG = 0,  
    HI_ISP_AE_AFTER_WB,  
    HI_ISP_AE_AFTER_DRC,  
    HI_ISP_AE_SWITCH_BUTT  
} hi_isp_ae_switch;
```

成员

成员名称	描述
HI_ISP_AE_AFTER_DG	After ISP digital gain。
HI_ISP_AE_AFTER_WB	After static WB。
HI_ISP_AE_AFTER_DRC	After DRC。
HI_ISP_AE_SWITCH_BUTT	预留值

注意事项

配置此参数时，需要把昇腾 AE 的大面积单色判断功能关闭。

3.33.6.5 hi_isp_ae_four_plane_mode

说明

FourPlaneMode 使能。

定义

```
typedef enum {  
    HI_ISP_AE_FOUR_PLANE_MODE_DISABLE = 0,  
    HI_ISP_AE_FOUR_PLANE_MODE_ENABLE,  
    HI_ISP_AE_FOUR_PLANE_MODE_BUTT  
} hi_isp_ae_four_plane_mode;
```

成员

成员名称	描述
HI_ISP_AE_FOUR_PLANE_MODE_DISABLE	1024 段直方图。
HI_ISP_AE_FOUR_PLANE_MODE_ENABLE	BGGR 四通道 256 段直方图。
HI_ISP_AE_FOUR_PLANE_MODE_BUTT	预留值

注意事项

配置此参数时，需要把昇腾 AE 的大面积单色判断功能关闭。

3.33.6.6 hi_isp_ae_hist_skip

说明

直方图统计时采样点设置。

定义

```
typedef enum {  
    HI_ISP_AE_HIST_SKIP_EVERY_PIXEL = 0,  
    HI_ISP_AE_HIST_SKIP_EVERY_2ND_PIXEL,  
    HI_ISP_AE_HIST_SKIP_EVERY_3RD_PIXEL,  
    HI_ISP_AE_HIST_SKIP_EVERY_4TH_PIXEL,  
    HI_ISP_AE_HIST_SKIP_EVERY_5TH_PIXEL,  
    HI_ISP_AE_HIST_SKIP_EVERY_8TH_PIXEL,  
    HI_ISP_AE_HIST_SKIP_EVERY_9TH_PIXEL,  
    HI_ISP_AE_HIST_SKIP_BUTT  
} hi_isp_ae_hist_skip;
```

成员

成员名称	描述
HI_ISP_AE_HIST_SKIP_EVERY_PIXEL	图像中每个点都统计。
HI_ISP_AE_HIST_SKIP_EVERY_2ND_PIXEL	图像在水平或垂直方向上每两个点统计一次。
HI_ISP_AE_HIST_SKIP_EVERY_3RD_PIXEL	图像在水平或垂直方向上每三个点统计一次。
HI_ISP_AE_HIST_SKIP_EVERY_4TH_PIXEL	图像在水平或垂直方向上每四个点统计一次。
HI_ISP_AE_HIST_SKIP_EVERY_5TH_PIXEL	图像在水平或垂直方向上每五个点统计一次。
HI_ISP_AE_HIST_SKIP_EVERY_8TH_PIXEL	图像在水平或垂直方向上每八个点统计一次。
HI_ISP_AE_HIST_SKIP_EVERY_9TH_PIXEL	图像在水平或垂直方向上每九个点统计一次。

3.33.6.7 hi_isp_ae_hist_offset_x

说明

直方图统计时水平方向起始点设置。

定义

```
typedef enum {  
    HI_ISP_AE_HIST_START_FORM_FIRST_COLUMN = 0,
```



```
HI_ISP_AE_HIST_START_FORM_SECOND_COLUMN,  
HI_ISP_AE_HIST_OFFSET_X_BUTT  
} hi_isp_ae_hist_offset_x;
```

成员

成员名称	描述
HI_ISP_AE_HIST_START_FORM_FIRST_COLUMN	从图像的第一列开始统计。
HI_ISP_AE_HIST_START_FORM_SECOND_COLUMN	从图像的第二列开始统计。

3.33.6.8 hi_isp_ae_hist_offset_y

说明

直方图统计时垂直方向起始点设置。

定义

```
typedef enum {  
    HI_ISP_AE_HIST_START_FORM_FIRST_ROW = 0,  
    HI_ISP_AE_HIST_START_FORM_SECOND_ROW,  
    HI_ISP_AE_HIST_OFFSET_Y_BUTT  
} hi_isp_ae_hist_offset_y;
```

成员

成员名称	描述
HI_ISP_AE_HIST_START_FORM_FIRST_ROW	从图像的第一行开始统计。
HI_ISP_AE_HIST_START_FORM_SECOND_ROW	从图像的第二行开始统计。

3.33.6.9 hi_isp_ae_hist_config

说明

定义全局直方图统计时的采样点方式。

定义

```
typedef struct {  
    hi_isp_ae_hist_skip hist_skip_x;  
    hi_isp_ae_hist_skip hist_skip_y;  
    hi_isp_ae_hist_offset_x hist_offset_x;  
    hi_isp_ae_hist_offset_y hist_offset_y;  
} hi_isp_ae_hist_config;
```

成员

成员名称	描述
hist_skip_x	直方图统计时水平方向采样点设置，取值为 0 时表示每 1 个像素采样一个进行统计，取值为 1 时表示每 2 个像素采样一个进行统计，以此类推。0 仅支持 FourPlaneMode 使能时配置。
hist_skip_y	直方图统计时垂直方向采样点设置。
hist_offset_x	直方图统计时水平方向起始点设置。
hist_offset_y	直方图统计时垂直方向起始点设置。

注意事项

- 可根据 Sensor 输入的 Bayer 数据格式，合理配置该结构体，使得全局直方图统计时只计入某单个分量数据。
- RGBIR 模式推荐全局直方图统计时只计入 G 分量，可根据 RGBIR Sensor 的 Bayer 数据格式合理配置该结构体实现。
- 配置 hist_config 相关参数时，需要把昇腾 AE 的大面积单色判断功能关闭。

3.33.6.10 hi_isp_ae_stat_mode

说明

定义统计模块的开方模式。

定义

```
typedef enum {  
    HI_ISP_AE_NORMAL = 0,  
    HI_ISP_AE_ROOT,  
    HI_ISP_AE_STAT_MODE_BUTT  
} hi_isp_ae_stat_mode;
```

成员

成员名称	描述
HI_ISP_AE_NORMAL	普通模式，不做处理。
HI_ISP_AE_ROOT	开方模式。

3.33.6.11 hi_isp_ae_crop

说明

对 AE 模块输入图像进行裁剪。

定义

```
typedef struct {  
    hi_bool enable;  
    hi_u16 x;  
    hi_u16 y;  
    hi_u16 w;  
    hi_u16 h;  
} hi_isp_ae_crop;
```

成员

成员名称	描述
enable	使能Crop，0为Crop不使能，1为Crop使能。
x	Crop X起始位置取值范围：[0, 图像宽度 - 256]
y	Crop Y起始位置取值范围：[0, 图像高度 - 120]
w	Crop宽度。取值范围：[256, 图像宽度]
h	Crop高度取值范围：[120, 图像高度]

注意事项

- 当裁剪图像宽小于600或高小于200时，AE统计信息仅全局直方图可用。
- 水平方向起始位置与Crop宽度之和应小于图像宽度，垂直方向起始位置与Crop高度之和应小于图像高度。
- 对于BE的AE统计信息，ISP BE分块场景下，不支持AE CROP（输入图像裁剪）功能。
- AE CROP（输入图像裁剪）参数会同步对MG模块生效。

3.33.6.12 hi_isp_awb_switch

说明

定义AWB统计模块在ISP pipeline的位置。

定义

```
typedef enum {  
    HI_ISP_AWB_AFTER_DG = 0,  
    HI_ISP_AWB_AFTER_EXPANDER,  
    HI_ISP_AWB_AFTER_DRC,  
    HI_ISP_AWB_SWITCH_BUTT  
} hi_isp_awb_switch;
```

成员

成员名称	描述
HI_ISP_AWB_AFTER_DG	AWB统计模块位置在Ditial Gain后。
HI_ISP_AWB_AFTER_EXPANDER	AWB统计模块位置在Expander后。

成员名称	描述
HI_ISP_AWB_AFTER_DRC	AWB统计模块位置在DRC后。

3.33.6.13 hi_isp_awb_gain_switch

说明

定义AWB增益的位置。

定义

```
typedef enum {  
    HI_ISP_AWB_GAIN_IN_ISP = 0,  
    HI_ISP_AWB_GAIN_IN_SENSOR,  
    HI_ISP_AWB_GAIN_IN_BUTT  
} hi_isp_awb_gain_switch;
```

成员

成员名称	描述
HI_ISP_AWB_GAIN_IN_ISP	AWB增益在ISP生效。
HI_ISP_AWB_GAIN_IN_SENSOR	AWB增益在sensor生效。

3.33.6.14 hi_isp_awb_crop

说明

对AWB模块输入图像进行裁剪。

定义

```
typedef struct {  
    hi_bool enable;  
    hi_u16 x;  
    hi_u16 y;  
    hi_u16 w;  
    hi_u16 h;  
} hi_isp_awb_crop;
```

成员

成员名称	描述
enable	使能Crop，0为Crop不使能，1为Crop使能。
x	Crop X起始位置取值范围：[0, 图像宽度 - 60 * zone_col]
y	Crop Y起始位置取值范围：[0, 图像高度 - 14 * zone_row]。

成员名称	描述
w	Crop宽度取值范围：[60 * zone_col, 图像宽度]。
h	Crop高度取值范围：[14 * zone_row, 图像高度]。

注意事项

- AWB要求每个分块的最小分辨率为60*14，当裁剪后的图像分辨率较小，分块数较多时，如果每块的分辨率小于60x14，那么需要减少分块数，以免每块过小。
- 水平方向起始位置与Crop宽度之和应小于图像宽度，垂直方向起始位置与Crop高度之和应小于图像高度。
- ISP BE分块场景下，不支持AWB CROP（输入图像裁剪）功能。

3.33.6.15 hi_isp_wb_stats_cfg

说明

定义AWB统计信息配置。

定义

```
typedef struct {  
    hi_isp_awb_switch awb_switch;  
    hi_u16 zone_row;  
    hi_u16 zone_col;  
    hi_u16 zone_bin;  
    hi_u16 hist_bin_thresh[HI_ISP_AWB_ZONE_BIN_MAX];  
    hi_u16 white_level;  
    hi_u16 black_level;  
    hi_u16 cb_max;  
    hi_u16 cb_min;  
    hi_u16 cr_max;  
    hi_u16 cr_min;  
    hi_isp_awb_crop crop;  
} hi_isp_wb_stats_cfg;
```

成员

成员名称	描述
awb_switch	WB统计模块位置，默认值为0，在Digital Gain后。
zone_row	WB统计垂直方向分块个数。取值范围：[0x1, min(0x20, ImageHeight / 14)]。 默认值与图像高度相关(保证每个分块的高度不小于 14)：如果图像高度大于等于448，默认值为0x20 如果图像高度小于448，默认值为ImageHeight/14。

成员名称	描述
zone_col	WB统计水平方向分块个数，取值范围：[BlkNum, min(0x20, ImageWidth/60)], BlkNum为ISP proc信息中ISP BE的分块数目。 默认值与图像的宽度相关(保证每个分块的宽度不小于 60)：如果图像宽度大于1920，默认值为0x20。 如果图像宽度小于1920，默认值为ImageWidth/60
zone_bin	亮度区域。取值范围：[1,1]。
hist_bin_thresh[HI_ISP_AWB_ZONE_BIN_MAX]	统计白点时，R,Gr,Gb,B四分量在各分块中的数量阈值。取值范围：[0x0, 0xFFFF]。 <code>#define HI_ISP_AWB_ZONE_BIN_MAX 4</code>
white_level	统计白点信息时，找白点的亮度上限。取值范围：[0x0, 0xFFFF]，默认值0xFFFF。
black_level	统计白点信息时，找白点的亮度下限。取值范围：[0x0, white_level]，默认值0x0。
cb_max	统计白点信息时，色差 B/G 的最大值，8bit精度，默认值512。取值范围：[0x0, 0xFFF]。
cb_min	统计白点信息时，色差 B/G 的最小值，8bit精度，默认值128。取值范围：[0x0, 0xFFF]。
cr_max	统计白点信息时，色差 R/G 的最大值，8bit精度，默认值512。取值范围：[0x0, 0xFFF]。
cr_min	统计白点信息时，色差 R/G 的最小值，8bit精度，默认值128。取值范围：[0x0, 0xFFF]。
crop	AWB输入图像裁剪配置。

注意事项

拼接模式下，每一路的参数是独立配置的。每一路的分块个数都需要配置，而且需要相同。即每一路的awb_switch，zone_row，zone_col需要相同。

3.33.6.16 hi_isp_ae_grid_info

说明

定义AE分区间统计信息的坐标信息。

定义

```
typedef struct {  
    hi_u16 grid_y_pos[HI_ISP_AE_ZONE_ROW + 1];  
    hi_u16 grid_x_pos[HI_ISP_AE_ZONE_COLUMN + 1];  
    hi_u8 status;  
} hi_isp_ae_grid_info;
```

成员

成员名称	描述
grid_y_pos[HI_ISP_AE_ZONE_ROW + 1]	AE每个分区 Y 方向坐标值。 #define HI_ISP_AE_ZONE_ROW 15
grid_x_pos[HI_ISP_AE_ZONE_COLUMN + 1]	AE每个分区 X 方向坐标值。 #define HI_ISP_AE_ZONE_COLUMN 17
status	AE统计信息坐标更新状态。

注意事项

- 非分块场景下，AE分区间统计信息为15*17个均匀分区的统计信息。
- BE分块场景下，AE分区间统计信息的分区大小是不均匀的，若按照均匀分区间的方式显示AE统计信息，会出现图像与统计信息不对应的情况，为避免这种问题，故提供每个分区的起始坐标信息。

3.33.6.17 hi_isp_ae_stats

说明

定义AE统计信息。

定义

```
typedef struct {  
    hi_u32 fe_hist1024_value[HI_ISP_WDR_MAX_FRAME_NUM][HI_ISP_HIST_NUM];  
    hi_u16 fe_global_avg[HI_ISP_WDR_MAX_FRAME_NUM][HI_ISP_BAYER_CHN_NUM];  
    hi_u16 fe_zone_avg[HI_ISP_WDR_MAX_FRAME_NUM][HI_ISP_AE_ZONE_ROW]  
[HI_ISP_AE_ZONE_COLUMN][HI_ISP_BAYER_CHN_NUM];  
    hi_u32 be_hist1024_value[HI_ISP_HIST_NUM];  
    hi_u16 be_global_avg[HI_ISP_BAYER_CHN_NUM];  
    hi_u16 be_zone_avg[HI_ISP_AE_ZONE_ROW][HI_ISP_AE_ZONE_COLUMN][HI_ISP_BAYER_CHN_NUM];  
    hi_isp_ae_grid_info fe_grid_info;  
    hi_isp_ae_grid_info be_grid_info;  
} hi_isp_ae_stats;
```

成员

```
#define HI_ISP_WDR_MAX_FRAME_NUM 4  
#define HI_ISP_HIST_NUM 1024  
#define HI_ISP_BAYER_CHN_NUM 4  
#define HI_ISP_AE_ZONE_ROW 15  
#define HI_ISP_AE_ZONE_COLUMN 17
```

成员名称	描述
fe_hist1024_value	FE全局1024段直方图信息。取值范围：[0x0, 0xFFFFFFFF]
fe_global_avg	FE全局统计平均值，按顺序分别表示 R,Gr,Gb,B 分量的平均值。 取值范围：[0x0, 0xFFFF]

成员名称	描述
fe_zone_avg	FE分区间统计平均值，按顺序分别表示 R,Gr,Gb,B 分量的平均值。 取值范围：[0x0, 0xFFFF]
be_hist1024_value	BE 全局1024段直方图信息。 取值范围：[0x0, 0xFFFFFFFF]
be_global_avg	BE 全局统计平均值，按顺序分别表示 R,Gr,Gb,B 分量的平均值。 取值范围：[0x0, 0xFFFF]
be_zone_avg	BE 分区间统计平均值，按顺序分别表示 R,Gr,Gb,B 分量的平均值。 取值范围：[0x0, 0xFFFF]
fe_grid_info	FE 分区间统计信息的坐标信息。
be_grid_info	BE 分区间统计信息的坐标信息。

注意事项

- WDR 模式下只在主路即最短帧所在 Pipe 才可以获取所有长短帧的统计信息，以 WDR2TO1 为例，HI_ISP_WDR_MAX_FRAME_NUM 对应下标为 0 时表示短帧统计信息，为 1 时表示长帧信息。
- WDR 合成前(FE)统计信息受到位于 FE AE 之前处理模块（FE AE 之前的模块有 DG/WB）增益影响，这些模块的 FE 增益值由算法内部保证与 BE 保持一致，无需单独配置。
- 当图像宽小于 600 或高小于 200 时，分区间统计信息不可用。

3.33.6.18 hi_isp_ae_stitch_stats

说明

定义拼接模式 AE 统计信息。当前版本不支持该功能。

定义

```
typedef struct {  
    hi_u32 fe_hist1024_value[HI_ISP_WDR_MAX_FRAME_NUM][HI_ISP_HIST_NUM];  
    hi_u16 fe_global_avg[HI_ISP_WDR_MAX_FRAME_NUM][HI_ISP_BAYER_CHN_NUM];  
    hi_u16 fe_zone_avg[HI_ISP_MAX_PIPE_NUM][HI_ISP_WDR_MAX_FRAME_NUM][HI_ISP_AE_ZONE_ROW]  
[HI_ISP_AE_ZONE_COLUMN][HI_ISP_BAYER_CHN_NUM];  
    hi_u32 be_hist1024_value[HI_ISP_HIST_NUM];  
    hi_u16 be_global_avg[HI_ISP_BAYER_CHN_NUM];  
    hi_u16 be_zone_avg[HI_ISP_MAX_PIPE_NUM][HI_ISP_AE_ZONE_ROW][HI_ISP_AE_ZONE_COLUMN]  
[HI_ISP_BAYER_CHN_NUM];  
}hi_isp_ae_stitch_stats;
```


成员

成员名称	描述
fe_hist1024_value[HI_ISP_WDR_MAX_FRAME_NUM][HI_ISP_HIST_NUM]	FE全局1024段直方图信息。 取值范围：[0x0, 0xFFFFFFFF] #define HI_ISP_WDR_MAX_FRAME_NUM 4 #define HI_ISP_HIST_NUM 1024
fe_global_avg	FE全局统计平均值，按顺序分别表示 R,Gr,Gb,B 分量的平均值。 取值范围：[0x0, 0xFFFF]
fe_zone_avg[HI_ISP_MAX_PIPE_NUM]	FE分区间统计平均值，按顺序分别表示 R,Gr,Gb,B 分量的平均值。 取值范围：[0x0, 0xFFFF]
be_hist1024_value[HI_ISP_HIST_NUM]	BE全局1024段直方图信息。 取值范围：[0x0, 0xFFFFFFFF] #define HI_ISP_HIST_NUM 1024
be_global_avg	BE全局统计平均值，按顺序分别表示 R,Gr,Gb,B 分量的平均值。 取值范围：[0x0, 0xFFFF]
be_zone_avg[HI_ISP_MAX_PIPE_NUM]	BE分区间统计平均值，按顺序分别表示 R,Gr,Gb,B 分量的平均值。 取值范围：[0x0, 0xFFFF] #define HI_ISP_MAX_PHY_PIPE_NUM 4 #define HI_ISP_MAX_VIR_PIPE_NUM 0 #define HI_ISP_MAX_PIPE_NUM (HI_ISP_MAX_PHY_PIPE_NUM + HI_ISP_MAX_VIR_PIPE_NUM)

注意事项

- 拼接后统计信息仅在拼接模式下生效，并且仅在拼接主路可以获取。
- 拼接后的分块均值只有参与拼接的Pipe分区间均值有效，其余Pipe的均值无效。如Pipe0和Pipe1进行拼接，则 HI_ISP_MAX_PIPE_NUM对应下标为0,1时 fe_zone_avg/be_zone_avg才有有效统计信息。
- WDR模式下只在拼接主路即最短帧所在Pipe才可以获取所有长短帧的统计信息，以WDR2TO1为例，HI_ISP_WDR_MAX_FRAME_NUM对应下标为0时表示短帧统计信息，为1时表示长帧信息。
- WDR合成前(FE)统计信息受到位于FE AE之前处理模块（FE AE之前的模块有DG/WB）增益影响，这些模块的 FE 增益值由算法内部保证与BE保持一致，无需单独配置。

3.33.6.19 hi_isp_mg_grid_info

说明定义

MG分区统计信息的坐标信息。

定义

```
typedef struct {  
    hi_u16 grid_y_pos[HI_ISP_MG_ZONE_ROW + 1];  
    hi_u16 grid_x_pos[HI_ISP_MG_ZONE_COLUMN + 1];  
    hi_u8 status;  
} hi_isp_mg_grid_info;
```

成员

成员名称	描述
grid_y_pos[HI_ISP_MG_ZONE_ROW + 1]	MG每个分区Y方向坐标值。 #define HI_ISP_MG_ZONE_ROW 15
grid_x_pos[HI_ISP_MG_ZONE_COLUMN + 1]	MG每个分区X方向坐标值。 #define HI_ISP_MG_ZONE_COLUMN 17
status	MG统计信息坐标更新状态。

注意事项

- 非分块场景下，MG分区统计信息为15*17个均匀分区的统计信息。
- BE分块场景下，MG分区统计信息的分区大小是不均匀的，若按照均匀分区的方式显示MG统计信息，会出现图像与统计信息不对应的情况，为避免这种问题，故提供每个分区的起始坐标信息。

3.33.6.20 hi_isp_mg_stats

说明定义

MG统计信息。

定义

```
typedef struct {  
    hi_u16 zone_avg[HI_ISP_MG_ZONE_ROW][HI_ISP_MG_ZONE_COLUMN][HI_ISP_BAYER_CHN_NUM];  
    hi_isp_mg_grid_info grid_info;  
} hi_isp_mg_stats;
```

成员

成员名称	描述
zone_avg[HI_ISP_MG_ZONE_ROW][HI_ISP_MG_ZONE_COLUMN]	分区间统计平均值，按顺序分别表示 R,Gr,Gb,B 分量的平均值。 取值范围：[0x0, 0xFFFF] #define HI_ISP_MG_ZONE_ROW 15 #define HI_ISP_MG_ZONE_COLUMN 17
grid_info	MG分区间统计信息的坐标值。

3.33.6.21 hi_isp_awb_grid_info

说明

定义AWB分区间统计信息的坐标信息。

定义

```
typedef struct {  
    hi_u16 grid_y_pos[HI_ISP_AWB_ZONE_ORIG_ROW + 1];  
    hi_u16 grid_x_pos[HI_ISP_AWB_ZONE_ORIG_COLUMN + 1];  
    hi_u8 status;  
} hi_isp_awb_grid_info;
```

成员

成员名称	描述
grid_y_pos[HI_ISP_AWB_ZONE_ORIG_ROW + 1]	AWB每个分区Y方向坐标值。 #define HI_ISP_AWB_ZONE_ORIG_ROW 32
grid_x_pos[HI_ISP_AWB_ZONE_ORIG_COLUMN + 1]	AWB每个分区X方向坐标值。 #define HI_ISP_AWB_ZONE_ORIG_COLUMN 32
status	AWB统计信息坐标更新状态。 0:not update 1:update

注意事项

- 非分块场景下，AWB分区间统计信息为zone_row* zone_col个均匀分区的统计信息。

- BE分块场景下，AWB分区统计信息的分区大小是不均匀的，若按照均匀分区的方式显示AWB统计信息，会出现图像与统计信息不对应的情况，为避免这种问题，故提供每个分区的起始坐标信息。

3.33.6.22 hi_isp_wb_stats

说明

定义AWB统计信息。

定义

```
typedef struct {  
    hi_u16 global_r;  
    hi_u16 global_g;  
    hi_u16 global_b;  
    hi_u16 count_all;  
    hi_u16 zone_avg_r[HI_ISP_AWB_ZONE_NUM];  
    hi_u16 zone_avg_g[HI_ISP_AWB_ZONE_NUM];  
    hi_u16 zone_avg_b[HI_ISP_AWB_ZONE_NUM];  
    hi_u16 zone_count_all[HI_ISP_AWB_ZONE_NUM];  
    hi_isp_awb_grid_info grid_info;  
} hi_isp_wb_stats;
```

成员

成员名称	描述
global_r	Bayer域全局统计的 R 分量平均值，取值范围：[0x0, 0xFFFF]。
global_g	Bayer域全局统计的 G 分量平均值，取值范围：[0x0, 0xFFFF]。
global_b	Bayer域全局统计的 B 分量平均值，取值范围：[0x0, 0xFFFF]。
count_all	全局统计的正方形灰色区域的像素个数，已做归一化，取值范围：[0x0, 0xFFFF]。
zone_avg_r[HI_ISP_AWB_ZONE_NUM]	Bayer域分区统计的 R 分量平均值，取值范围：[0x0, 0xFFFF]。 #define HI_ISP_AWB_ZONE_NUM (HI_ISP_AWB_ZONE_ORIG_ROW * HI_ISP_AWB_ZONE_ORIG_COLUMN)
zone_avg_g[HI_ISP_AWB_ZONE_NUM]	Bayer域分区统计的 G 分量平均值，取值范围：[0x0, 0xFFFF]。 #define HI_ISP_AWB_ZONE_NUM (HI_ISP_AWB_ZONE_ORIG_ROW * HI_ISP_AWB_ZONE_ORIG_COLUMN)
zone_avg_b[HI_ISP_AWB_ZONE_NUM]	Bayer域分区统计的 B 分量平均值，取值范围：[0x0, 0xFFFF]。 #define HI_ISP_AWB_ZONE_NUM (HI_ISP_AWB_ZONE_ORIG_ROW * HI_ISP_AWB_ZONE_ORIG_COLUMN)
zone_count_all	分区统计的正方形灰色区域的像素个数，已做归一化，取值范围：[0x0, 0xFFFF]。
grid_info	Awb分区统计信息的坐标信息。

注意事项

表 3-14 AWB 区域统计信息（以 zone_avg_r 为例，zone_row=32，zone_col=32）

Index	Zone	Bin	读取数据说明
0	0	0	第 0 个分块的 Average R
1	1	0	第 1 个分块的 Average R
2	2	0	第 2 个分块的 Average R
3	3	0	第 3 个分块的 Average R
4	4	0	第 4 个分块的 Average R
.....			
1023	1023	0	第 1023 个分块的 Average R

zone_count_all做归一化是为了消除分辨率差异对灰点个数的影响。归一化公式：
CountAll = (Count of Gray Pixels << 16) / (Count of All Pixels)。

3.33.6.23 hi_isp_wb_stitch_stats

说明

定义拼接模式WB统计信息。

定义

```
typedef struct {  
    hi_u16 zone_row;  
    hi_u16 zone_col;  
    hi_u16 zone_avg_r[HI_ISP_AWB_ZONE_STITCH_MAX];  
    hi_u16 zone_avg_g[HI_ISP_AWB_ZONE_STITCH_MAX];  
    hi_u16 zone_avg_b[HI_ISP_AWB_ZONE_STITCH_MAX];  
    hi_u16 zone_count_all[HI_ISP_AWB_ZONE_STITCH_MAX];  
} hi_isp_wb_stitch_stats;
```

成员

成员名称	描述
zone_row	拼接统计分块行数，取值范围：[0x0, 0x20]。
zone_col	拼接统计分块列数，取值范围：[0x0, 0x80]。
zone_avg_r[HI_ISP_AWB_ZONE_STITCH_MAX]	Bayer域分区间统计的 R 分量平均值，取值范围：[0x0,0xFFFF]。 #define HI_ISP_AWB_ZONE_STITCH_MAX (HI_ISP_AWB_ZONE_NUM * HI_ISP_MAX_STITCH_NUM)

成员名称	描述
zone_avg_r[HI_ISP_AWB_ZONE_STITCH_MAX]	Bayer域分区间统计的 G 分量平均值，取值范围：[0x0,0xFFFF]。 #define HI_ISP_AWB_ZONE_STITCH_MAX (HI_ISP_AWB_ZONE_NUM * HI_ISP_MAX_STITCH_NUM)
zone_avg_b[HI_ISP_AWB_ZONE_STITCH_MAX]	Bayer域分区间统计的 B 分量平均值，取值范围：[0x0,0xFFFF]。 #define HI_ISP_AWB_ZONE_STITCH_MAX (HI_ISP_AWB_ZONE_NUM * HI_ISP_MAX_STITCH_NUM)
zone_count_all[HI_ISP_AWB_ZONE_STITCH_MAX]	分区间统计的正方形灰色区域的像素个数，已做归一化，取值范围：[0x0, 0xFFFF]。 #define HI_ISP_AWB_ZONE_STITCH_MAX (HI_ISP_AWB_ZONE_NUM * HI_ISP_MAX_STITCH_NUM)

3.33.6.24 hi_isp_focus_stats_cfg

说明

定义AF统计信息配置。

定义

```
typedef struct {  
    hi_isp_af_cfg      config;  
    hi_isp_af_h_param  h_param_iir0;  
    hi_isp_af_h_param  h_param_iir1;  
    hi_isp_af_v_param  v_param_fir0;  
    hi_isp_af_v_param  v_param_fir1;  
    hi_isp_af_fv_param fv_param;  
} hi_isp_focus_stats_cfg;
```

成员

成员名称	描述
config	AF全局配置参数。
h_param_iir0	奇数行水平滤波器IIR参数设置。
h_param_iir1	偶数行水平滤波器IIR参数设置。
v_param_fir0	奇数行垂直滤波器FIR参数设置。
v_param_fir1	偶数行垂直滤波器FIR参数设置。
fv_param	统计结果输出格式参数配置。

注意事项

当分辨率较小，分块数较多时，如果每块的分辨率小于32x32，那么需要减少分块数，以免每块过小。建议在分辨率小的情况下，使用较少的分块数。

3.33.6.25 hi_isp_af_cfg

说明

定义AF统计信息配置参数。

定义

```
typedef struct {  
    hi_bool      af_en;  
    hi_u16       zone_col;  
    hi_u16       zone_row;  
    hi_isp_af_peak_mode peak_mode;  
    hi_isp_af_squ_mode  squ_mode;  
    hi_isp_af_crop      crop;  
    hi_isp_af_stats_pos stats_pos;  
    hi_isp_af_raw_cfg    raw_cfg;  
    hi_isp_af_pre_filter_cfg pre_flt_cfg;  
    hi_u16      high_luma_threshold  
} hi_isp_af_cfg;
```

成员

成员名称	描述
af_en	AF使能。取值范围：0、1。
zone_col	AF统计水平方向窗口个数，取值范围：[1, 17]。
zone_row	AF统计垂直方向窗口个数，取值范围：[1, 15]。
peak_mode	PEAK模式，决定分区间统计值是否进行求峰值处理。与square_mode共同影响分区间统计值。 0:ISP_AF_STA_NORM 1:ISP_AF_STA_PEAK
square_mode	平方模式，决定分区间水平和垂直方向统计值是否进行平方处理。与peak_mode共同影响分区间统计值。 0:ISP_AF_STA_SUM_NORM 1:ISP_AF_STA_SUM_SQU
crop	AF输入图像的裁剪配置。
stats_pos	AF统计信息位置配置，决定是在Bayer还是YUV域统计。 0:ISP_AF_STATISTICS_AFTER_DGAIN 1:ISP_AF_STATISTICS_AFTER_DRC 2:ISP_AF_STATISTICS_AFTER_CSC
raw_cfg	AF Bayer域配置，如果用户选择将AF模块放在Bayer域，那么需要配置这里。
pre_flt_cfg	AF模块预滤波配置，此模块可以帮助消除画面中的椒盐噪声。
high_luma_threshold	AF高亮点统计值Threshold设置，取值范围：[0,0xFF]。

注意事项

- 分区间统计值的计算方式为：
 - peak_mode为HI_ISP_AF_STA_NORM且square_mode为HI_ISP_AF_STA_SUM_NORM模式时，对分区间内每个像素的滤波器输出值求和即为分区间统计值；
 - peak_mode为HI_ISP_AF_STA_NORM且square_mode为HI_ISP_AF_STA_SUM_SQUARE模式时，对分区间内每个像素的滤波器输出值求平方和即为分区间统计值；
 - peak_mode为HI_ISP_AF_STA_PEAK且square_mode为HI_ISP_AF_STA_SUM_NORM模式时，先对分区间内一行中每个像素的滤波器输出值取最大值，再对多行的最大值求和即为分区间统计值；
 - peak_mode为HI_ISP_AF_STA_PEAK且square_mode为HI_ISP_AF_STA_SUM_SQUARE模式时，先对分区间内一行中每个像素的滤波器输出值取最大值，再对多行的最大值求平方和即为分区间统计值。
- 分区间亮度统计值（y）的计算方式为：对分区间内每个像素的亮度求和即为分区间亮度统计值（y）。
- 当ISP BE处于分块方式时，AF统计信息的水平方向窗口个数最小为BlockNum（BlockNum 可以参考 ISP 的 proc 信息）分块的个数，此时取值范围为 [BlockNum, 17]。

3.33.6.26 hi_isp_af_peak_mode

说明

PEAK模式，决定分区间统计值是否进行求峰值处理。

定义

```
typedef enum {  
    HI_ISP_AF_STA_NORM      = 0,  
    HI_ISP_AF_STA_PEAK,  
    HI_ISP_AF_STA_BUTT  
} hi_isp_af_peak_mode;
```

成员

成员名称	描述
HI_ISP_AF_STA_NORM	- 当square_mode为 HI_ISP_AF_STA_SUM_NORM 模式时，对分区间内每个像素的输出值求和即为分区间统计值； - 当 square_mode为 HI_ISP_AF_STA_SUM_SQUARE 模式时，对分区间内每个像素的滤波器输出值求平方和即为分区间统计值。

成员名称	描述
HI_ISP_AF_STA_PEAK	- 当 square_mode 为 HI_ISP_AF_STA_SUM_NORM 模式时，先对分区间内一行中每个像素的滤波器输出值取最大值，再对多行的最大值求和即为分区间统计值； - 当 square_mode 为 HI_ISP_AF_STA_SUM_SQUARE 模式时，先对分区间内一行中每个像素的滤波器输出值取最大值，再对多行的最大值求平方和即为分区间统计值。

3.33.6.27 hi_isp_af_squ_mode

说明

平方模式，决定分区间统计值是否进行平方处理。

定义

```
typedef enum {  
    HI_ISP_AF_STA_SUM_NORM    = 0,  
    HI_ISP_AF_STA_SUM_SQUARE,  
    HI_ISP_AF_STA_SUM_BUTT  
} hi_isp_af_squ_mode;
```

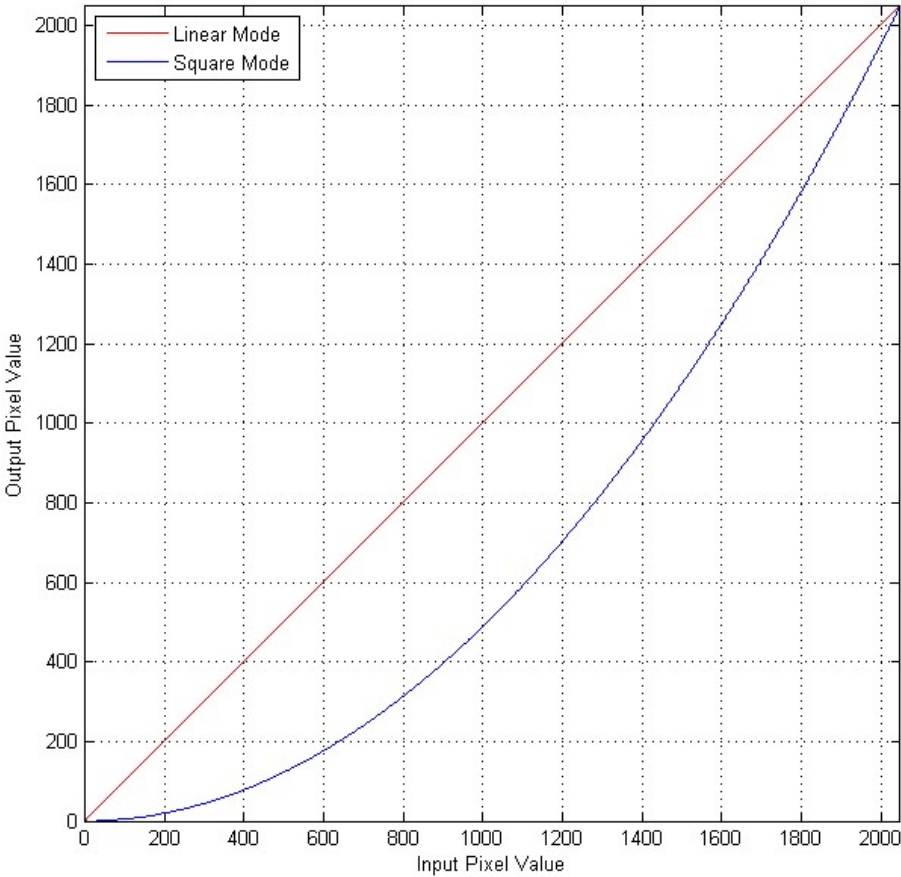
成员

成员名称	描述
HI_ISP_AF_STA_SUM_NORM	- 当 peak_mode 为 HI_ISP_AF_STA_NORM 模式时，对分区间内每个像素的滤波器输出值求和即为分区间统计值； - 当 peak_mode 为 HI_ISP_AF_STA_PEAK 模式时，先对分区间内一行中每个像素的滤波器输出值取最大值，再对多行的最大值求和即为分区间 IIR 统计值。
HI_ISP_AF_STA_SUM_SQUARE	- 当 peak_mode 为 HI_ISP_AF_STA_NORM 模式时，对分区间内每个像素的滤波器输出值求平方和即为分区间统计值； - 当 peak_mode 为 HI_ISP_AF_STA_PEAK 模式时，先对分区间内一行中每个像素的滤波器输出值取最大值，再对多行的最大值求平方和即为分区间统计值。

注意事项

采用 Square 模式，会对滤波器输出归一化后平方再做统计，如下图所示，input 为滤波后输出像素值，output 为处理后输出值，相比 normal 模式，在临近焦点附近可以得到比较陡峭的 FV 曲线，并且对幅值较小的噪声信号有一定的抑制作用，但也会使 FV 曲线平坦区域更加平坦，用户应该根据实际的场景需求来设定这个参数。

图 3-11 Square 模式



3.33.6.28 hi_isp_af_crop

说明

对 AF 模块输入图像进行裁剪。

定义

```
typedef struct {  
    hi_bool enable;  
    hi_u16 x;  
    hi_u16 y;  
    hi_u16 w;  
    hi_u16 h;  
} hi_isp_af_crop;
```

成员

成员名称	描述
enable	使能 Crop，取值范围：[0, 1]。
x	Crop X 起始位置，取值范围：[0, ImageWidth - 256]。
y	Crop Y 起始位置，取值范围：[0, ImageHeight - 120]。

成员名称	描述
w	Crop 宽度，8 对齐，取值范围：[256, ImageWidth]。
h	Crop 高度，2 对齐，取值范围：[120, ImageHeight]。

注意事项

- 水平方向起始位置与 Crop 宽度之和应小于图像宽度，垂直方向起始位置与 Crop 高度之和应小于图像高度。
- 对于 BE 的 AF 统计信息，ISP BE 分块场景下，不支持 AF CROP（输入图像裁剪）功能。

3.33.6.29 hi_isp_af_stats_pos

说明

配置 AF 模块统计位置。

定义

```
typedef enum {  
    HI_ISP_AF_STATS_AFTER_DGAIN = 0,  
    HI_ISP_AF_STATS_AFTER_DRC,  
    HI_ISP_AF_STATS_AFTER_CSC,  
    HI_ISP_AF_STATS_BUTT  
} hi_isp_af_stats_pos;
```

成员

成员名称	描述
HI_ISP_AF_STATS_AFTER_DGAIN	将 AF 模块放在 DGain 后进行统计，属于 RAW 域。
HI_ISP_AF_STATS_AFTER_DRC	将 AF 模块放在 DRC 及 Dither 后进行统计，属于 RAW 域。
HI_ISP_AF_STATS_AFTER_CSC	将 AF 模块放在 CSC 后进行统计，属于 YUV 域。

注意事项

- WDR 模式下，非宽动态场景的 Bayer 域（即 RAW 域）统计信息无明显变化趋势，推荐使用 YUV 域统计信息。
- AF 的 FE 统计信息只支持 DGain 之后进行统计，而 BE 统计信息支持 DGain、DRC 和 CSC 之后进行统计。

3.33.6.30 hi_isp_af_raw_cfg

说明

AF 模块 Bayer 域配置。

定义

```
typedef struct {  
    hi_u8      gamma_gain_limit;  
    hi_u8      gamma_value;  
    hi_isp_bayer_format bayer_format;  
} hi_isp_af_raw_cfg;
```

成员

成员名称	描述
gamma_gain_limit	AF 模块 gamma 限制。取值范围：[0x0, 0x5]。
gamma_value	AF 模块 gamma 类型。取值范围：[0x0, 0x6]。
bayer_format	设置 Bayer pattern，支持 BAYER_RGGB，BAYER_GRGB，BAYER_GBRG，BAYER_BGGR 四种。

3.33.6.31 hi_isp_af_pre_filter_cfg

说明

AF 模块预滤波器配置。

定义

```
typedef struct {  
    hi_bool    en;  
    hi_u16     strength;  
} hi_isp_af_pre_filter_cfg;
```

成员

成员名称	描述
en	使能 AF 预滤波模块。取值范围：0、1
strength	设置滤波强度，取值范围：[0x0, 0xFFFF]。

3.33.6.32 hi_isp_af_level_depend

说明

AF Level Depend Gain 模块配置。

定义

```
typedef struct {  
    hi_bool    en;  
    hi_u16     threshold_low;  
    hi_u16     gain_low;  
    hi_u16     slope_low;  
    hi_u16     threshold_high;  
    hi_u16     gain_high;  
    hi_u16     slope_high;  
} hi_isp_af_level_depend;
```

成员

成员名称	描述
en	使能 Level Depend Gain 模块。
threshold_low	设置 Dark Side Threshold，取值范围：[0x0, 0xFF]。
gain_low	设置 Dark Side Minimum Gain，取值范围：[0x0, 0xFF]。
slope_low	设置 Dark Side Slope，取值范围：[0x0, 0xF]。
threshold_high	设置 Bright Side Threshold，取值范围：[0x0, 0xFF]。
gain_high	设置 Bright Side Minimum Gain，取值范围：[0x0, 0xFF]。
slope_high	设置 Bright Side Slope，取值范围：[0x0, 0xF]。

3.33.6.33 hi_isp_af_coring

说明

AF Coring 模块配置。

定义

```
typedef struct {  
    hi_u16     threshold;  
    hi_u16     slope;  
    hi_u16     limit;  
} hi_isp_af_coring;
```

成员

成员名称	描述
threshold	设置 Coring Threshold，取值范围：[0x0, 0x7FF]。
slope	设置 Coring Slope，取值范围：[0x0, 0xF]。
limit	设置 Coring Maximum Output，取值范围：[0x0, 0x7FF]。

3.33.6.34 hi_isp_af_h_param

说明

AF 统计水平滤波器 IIR 参数设置。

定义

```
typedef struct {  
    hi_bool    narrow_band_en;  
    hi_bool    iir_en[HI_ISP_IIR_EN_NUM];  
    hi_u8      iir_shift;  
    hi_s16     iir_gain[HI_ISP_IIR_GAIN_NUM];  
    hi_u16     iir_shift_lut[HI_ISP_IIR_SHIFT_NUM];  
    hi_isp_af_level_depend level_depend;  
    hi_isp_af_coring    coring;  
} hi_isp_af_h_param;
```

成员

成员名称	描述
narrow_band_en	IIR 滤波器窄带宽使能。
iir_en	IIR 级联中 IIR 的使能。
iir_shift	IIR 滤波器整体位移量,取值范围：[0, 63]。
iir_gain	IIR 滤波器系数，用于控制 IIR 滤波器的频率响应。 取值范围：Gain0:[0, 255]；Others:[-511, 511]。
iir_shift_lut	IIR 滤波器移位调整，取值范围：[0, 0x7]。
level_depend	IIR 滤波器 Level Depend Gain 模块配置。
coring	IIR 滤波器 Coring 模块配置。

注意事项

iir_gain[0]为无符号 8bit 数，其他 iir_gain[1]-[6]为有符号 10bit 数，且不能等于-512。IIR 使能、narrow_band_en 使能、滤波器系数以及 iir_shift 通过 PQ Tools 可以生成，用户可以根据实际需要调节 LDG 和 Coring 模块以获取最优 FV 统计值输出。

3.33.6.35 hi_isp_af_v_param

说明

AF 统计垂直滤波器 FIR 参数设置。

定义

```
typedef struct {  
    hi_s16 fir_gain[HI_ISP_FIR_GAIN_NUM];  
    hi_isp_af_level_depend level_depend;
```

```
hi_isp_af_coring coring;  
} hi_isp_af_v_param;
```

成员

成员名称	描述
fir_gain[HI_ISP_FIR_GAIN_NUM]	FIR 滤波器系数，取值范围：[-31, 31]。用于控制 FIR 滤波器的频率响应。
level_depend	FIR 滤波器 Level Depend Gain 模块配置。
coring	FIR 滤波器 Coring 模块配置。

注意事项

FIR 滤波器系数通过 PQ Tools 可以生成，用户可以根据实际需要调节 LDG 和 Coring 模块以获取最优垂直 FV 统计值输出。

3.33.6.36 hi_isp_af_fv_param

说明

AF 统计结果输出格式参数。

定义

```
typedef struct {  
    hi_u16 acc_shift_y;  
    hi_u16 acc_shift_h[HI_ISP_ACC_SHIFT_H_NUM];  
    hi_u16 acc_shift_v[HI_ISP_ACC_SHIFT_V_NUM];  
    hi_u16 hl_cnt_shift;  
} hi_isp_af_fv_param;
```

成员

成员名称	描述
acc_shift_y	亮度 Y 统计值移位寄存器值，取值范围：[0, 0xF]。
acc_shift_h[HI_ISP_ACC_SHIFT_H_NUM]	水平 IIR 滤波值统计值移位寄存器值，取值范围：[0, 0xF]。
acc_shift_v[HI_ISP_ACC_SHIFT_V_NUM]	垂直 FIR 滤波值统计值移位寄存器值，取值范围：[0, 0xF]。
hl_cnt_shift	高亮计数器统计值移位寄存器值，取值范围：[0, 0xF]。

注意事项

FV 值用 unsinged 16bit 表示，当场景细节较多或噪声较多时，FV 值可能会溢出，这就需要对输出进行向右移位以保证输出在合理范围。用户根据自己的需求对相应的 shift 值进行调节。

3.33.6.37 hi_isp_focus_zone

说明

定义 AF 统计信息参数。

定义

```
typedef struct {  
    hi_u16 v1;  
    hi_u16 h1;  
    hi_u16 v2;  
    hi_u16 h2;  
    hi_u16 y;  
    hi_u16 hl_cnt;  
} hi_isp_focus_zone;
```

成员

成员名称	描述
v1	分区间统计的 AF 垂直方向奇数列 FIR 滤波器的统计值。取值范围：[0, 0xFFFF]。
h1	分区间统计的 AF 水平方向奇数行 IIR 滤波器的统计值。取值范围：[0, 0xFFFF]。
v2	分区间统计的 AF 垂直方向偶数列 FIR 滤波器的统计值。取值范围：[0, 0xFFFF]。
h2	分区间统计的 AF 水平方向偶数行 IIR 滤波器的统计值。取值范围：[0, 0xFFFF]。
y	分区间统计的 AF 亮度统计值，区块中每个像素点的亮度累加值。取值范围：[0, 0xFFFF]。
hl_cnt	分区间统计的 AF 亮度超阈值计数器值。取值范围：[0, 0xFFFF]。

注意事项

y 为每个块的 Y 分量累加值，用户可以用此值辅助 AF 算法，比如某些块中含有高亮光源，容易对 FV 值产生干扰，可根据亮度情况降低块 FV 对全局 FV 的影响，亦可用这些块实现一个移动侦测算法，用于检测聚焦过程中的一些运动干扰，用户可以根据自己 AF 算法的需求使用这个字段的统计信息。

3.33.6.38 hi_isp_fe_focus_stats

说明

定义 FE AF 统计信息，当前不支持。

定义

```
typedef struct {  
    hi_isp_focus_zone zone_metrics[HI_ISP_WDR_MAX_FRAME_NUM][HI_ISP_AF_ZONE_ROW]  
    [HI_ISP_AF_ZONE_COLUMN];  
} hi_isp_fe_focus_stats;
```

成员

成员名称	描述
zone_metrics[HI_ISP_WDR_MAX_FRAME_NUM]	ISP AF FE 分块统计信息。 #define HI_ISP_WDR_MAX_FRAME_NUM 4

3.33.6.39 hi_isp_be_focus_stats

说明

定义 BE AF 统计信息。

定义

```
typedef struct {  
    hi_isp_focus_zone zone_metrics[HI_ISP_AF_ZONE_ROW][HI_ISP_AF_ZONE_COLUMN];  
} hi_isp_be_focus_stats;
```

成员

成员名称	描述
zone_metrics	ISP AF BE 分块统计信息。

3.33.6.40 hi_isp_focus_grid_info

说明

定义 AF 分区间统计信息的坐标信息。

定义

```
typedef struct {  
    hi_u16 grid_y_pos[HI_ISP_AF_ZONE_ROW + 1];  
    hi_u16 grid_x_pos[HI_ISP_AF_ZONE_COLUMN + 1];  
    hi_u8 status;  
} hi_isp_focus_grid_info;
```

成员

成员名称	描述
grid_y_pos	AF 每个分区 Y 方向坐标值。
grid_x_pos	AF 每个分区 X 方向坐标值。
status	AF 统计信息坐标更新状态。 0: not update 1: update

注意事项

- 非分块场景下，AF 分区间统计信息为 zone_col * zone_row 个均匀分区的统计信息。
- BE 分块场景下，AF 分区间统计信息的分区大小是不均匀的，若按照均匀分区间的方式显示 AF 统计信息，会出现图像与统计信息不对应的情况，为避免这种问题，故提供每个分区的起始坐标信息。

3.33.6.41 hi_isp_af_stats

说明

定义 FE 和 BE 的 AF 统计信息。

定义

```
typedef struct {  
    hi_isp_fe_focus_stats fe_af_stat;  
    hi_isp_be_focus_stats be_af_stat;  
    hi_isp_focus_grid_info fe_af_grid_info;  
    hi_isp_focus_grid_info be_af_grid_info;  
} hi_isp_af_stats;
```

成员

成员名称	描述
fe_af_stat	ISP FE 的 AF 统计信息。
be_af_stat	ISP BE 的 AF 统计信息。
fe_af_grid_info	ISP FE 的 AF 分区间统计信息的坐标。
be_af_grid_info	ISP BE 的 AF 分区间统计的信息坐标。

注意事项

- 当前不支持从 FE 通道获取统计信息。
- 线性模式下离线模式推荐使用 FE 的统计信息(实时性更佳)。

- WDR 模式下推荐使用 BE DRC 后的统计信息(参考 hi_isp_af_stats_pos)。

3.34 错误码

表 3-15 ISP API 错误码

错误代码	宏定义	描述
0xA01C8003	HI_ERR_ISP_ILLEGAL_PARAM	输入参数无效
0xA01C8006	HI_ERR_ISP_NULL_PTR	空指针错误
0xA01C8008	HI_ERR_ISP_NOT_SUPPORT	当前 ISP 不支持
0xA01C800C	HI_ERR_ISP_NO_MEM	内存不足
0xA01C8040	HI_ERR_ISP_NOT_INIT	ISP 没有初始化
0xA01C8041	HI_ERR_ISP_MEM_NOT_INIT	外部寄存器没有初始化
0xA01C8042	HI_ERR_ISP_ATTR_NOT_CFG	属性未配置
0xA01C8043	HI_ERR_ISP_SNS_UNREGISTER	Sensor 未注册
0xA01C8044	HI_ERR_ISP_INVALID_ADDR	无效地址
0xA01C8046	HI_ERR_ISP_NO_INT	ISP 无中断
0xA0388002	HI_ERR_MOTIONFUSION_INVALID_CHNID	MOTIONFUSION无效的通道ID
0xA0388003	HI_ERR_MOTIONFUSION_ILLEGAL_PARAM	MOTIONFUSION参数设置无效
0xA0388006	HI_ERR_MOTIONFUSION_NULL_PTR	MOTIONFUSION空指针错误
0xA0388010	HI_ERR_MOTIONFUSION_SYS_NOT_READY	MOTIONFUSION系统未初始化

4 OpenHiva API

- 4.1 接口简介
- 4.2 返回码列表
- 4.3 类型定义
- 4.4 初始化、退出和调度接口
- 4.5 OpenHiva::Node接口
- 4.6 OpenHiva::TopicOptions接口
- 4.7 OpenHiva::Publisher接口
- 4.8 OpenHiva::Subscriber接口
- 4.9 OpenHiva::ServiceServer接口
- 4.10 OpenHiva::ServiceClient接口
- 4.11 NnEngine接口
- 4.12 定时器接口
- 4.13 配置管理相关接口

4.1 接口简介

数据面通信接口是业务和基础平台约定的实时通信类编程接口。它约束了用户的实时编程模型，指定了实时接口使用规范和规格。在接口的约束下，以平滑牵引当前业务代码满足机器人系统实时性为目标，最终实现基本的实时通信框架。

本章主要针对基础平台数据面子系统接口提供了说明，对系统软件的通信接口、定时器、配置管理等进行展开描述，助力用户更快、更高效地进行机器人业务应用开发。

须知

当前版本提供的API参考为试用接口，后续版本会存在变更，暂不支持应用于商用产品中。

表 4-1 关键接口概览

接口分类	关键接口	接口简介	头文件	依赖的库文件
初始化、退出和调度接口	OpenHiva::Init	初始化框架资源，如节点、线程组等。	Robot Middleware 安装目录/ robotmiddle ware/sdk/ include/ hiva/cm/ open/init.h	Robot Middleware 安装目录/ robotmiddle ware/sdk/lib/ hiva/cm/ libHivaOpen. so
	OpenHiva::Ready	判断OpenHiva框架状态。		
	OpenHiva::Shutdown	终止框架运行，释放资源。		
	OpenHiva::WaitForShutdown	阻塞式调用，防止主线程提前退出，框架退出前会释放资源。		
	OpenHiva::SpinOnce	调度groupName指定的group，订阅的Topic在指定group内会被调度处理消息。		
OpenHiva::Node接口	OpenHiva::Node	构造函数，完成Node的初始化工作。	Robot Middleware 安装目录/ robotmiddle ware/sdk/ include/ hiva/cm/ open/ node.h	
	OpenHiva::Node::CreatePublisher	创建发布者。		
	OpenHiva::Node::CreateSubscriber	创建订阅者。		
	OpenHiva::Node::CreateServer	创建Service的服务端。		
	OpenHiva::Node::CreateClient	创建Service的客户端。		
	OpenHiva::Node::Ready	判断Node是否关闭。		
	OpenHiva::Node::GetName	获取Node的名字。		
	OpenHiva::Node::GetNameSpace	获取Node的域名。		

接口分类	关键接口	接口简介	头文件	依赖的库文件
OpenHiva::TopicOptions接口	OpenHiva::TopicOptions::BuildShmOptions	设置SHM或DSF通信分配的内存块大小和内存块个数。	Robot Middleware 安装目录/robotmiddleware/sdk/include/hiva/cm/open/topic_options.h	
	OpenHiva::TopicOptions::BuildQueueOptions	设置DSF通信的队列参数。		
	OpenHiva::TopicOptions::BuildGroupName	设置Topic所属的groupName。		
	OpenHiva::TopicOptions::BuildTransportOptions	设置Topic的通信方式。		
	OpenHiva::TopicOptions::BuildDDSQos	设置Topic所属DDS通信的QoS策略。		
OpenHiva::Publisher接口	OpenHiva::Publisher::Publisher	发布端的默认构造函数。	Robot Middleware 安装目录/robotmiddleware/sdk/include/hiva/cm/open/publisher.h	
	OpenHiva::Publisher::Publish	消息发布函数。		
	OpenHiva::Publisher::Ready	判断Publisher是否处于工作状态。		
OpenHiva::Subscriber接口	OpenHiva::Subscriber::Subscriber	订阅端的默认构造函数。	Robot Middleware 安装目录/robotmiddleware/sdk/include/hiva/cm/open/subscriber.h	
	OpenHiva::Subscriber::Ready	判断Subscriber是否处于工作状态。		
	OpenHiva::Subscriber::StartReadMessage	启动接收消息。		
	OpenHiva::Subscriber::StopReadMessage	停止接收消息。		

接口分类	关键接口	接口简介	头文件	依赖的库文件
OpenHiva::ServiceServer接口	OpenHiva::ServiceServer::Enable	使能服务端。	Robot Middleware 安装目录/robotmiddle ware/sdk/ include/ hiva/cm/ open/ service_serv er.h	
	OpenHiva::ServiceServer::Disable	去使能服务端。		
	OpenHiva::ServiceServer::Destroy	释放 OpenHiva::Node::CreateServer申请的资源。		
OpenHiva::ServiceClient接口	OpenHiva::ServiceClient::InvokeService	客户端向服务端发送请求。	Robot Middleware 安装目录/robotmiddle ware/sdk/ include/ hiva/cm/ open/ service_clie nt.h	
	OpenHiva::ServiceClient::Destroy	释放 OpenHiva::Node::CreateClient申请的资源。		
NnEngine接口	Hiva::CreateNnEngine	创建NN引擎。	Robot Middleware 安装目录/robotmiddle ware/sdk/ include/ hiva/cm/ inner_topic/ hiva_nn_top ic.h	Robot Middleware 安装目录/robotmiddle ware/sdk/lib/ hiva/cm/ libHivaNnen gine.so
	Hiva::DestroyNnEngine	销毁NN引擎。		
定时器接口	Hiva::CreateTimer	创建Timer定时器。	Robot Middleware 安装目录/robotmiddle ware/sdk/ include/ hiva/cm/ hiva_timer/ timer.h	Robot Middleware 安装目录/robotmiddle ware/sdk/lib/ hiva/cm/ libHivaTimer. so
	Hiva::Timer::Start	启动Timer定时器。		
	Hiva::Timer::IsValid	判断Timer定时器是否有效。		
	Hiva::Timer::SetPeriod	重置Timer定时器周期。		
	Hiva::Timer::Stop	停止Timer定时器。		

接口分类	关键接口	接口简介	头文件	依赖的库文件
	Hiva::CreateWallTimer	创建WallTimer定时器。	Robot Middleware 安装目录/ robotmiddleware/sdk/ include/ hiva/cm/ hiva_timer/ wall_timer.h	
	Hiva::WallTimer::Start	启动WallTimer定时器。		
	Hiva::WallTimer::IsValid	判断WallTimer定时器是否有效。		
	Hiva::WallTimer::SetPeriod	重置WallTimer定时器周期。		
	Hiva::WallTimer::Stop	停止WallTimer定时器。		
	Hiva::CreateSteadyTimer	创建SteadyTimer定时器。	Robot Middleware 安装目录/ robotmiddleware/sdk/ include/ hiva/cm/ hiva_timer/ steady_time r.h	
	Hiva::SteadyTimer::Start	启动SteadyTimer定时器。		
	Hiva::SteadyTimer::IsValid	判断SteadyTimer定时器是否有效。		
	Hiva::SteadyTimer::SetPeriod	重置SteadyTimer定时器周期。		
	Hiva::SteadyTimer::Stop	停止SteadyTimer定时器。		
配置管理 相关接口	CfgMgr::BatchReset	批量删除持久化数据。	Robot Middleware 安装目录/ robotmiddleware/sdk/ include/ hiva/cm/ cfgmgr/ param.h	Robot Middleware 安装目录/ robotmiddleware/sdk/lib/ hiva/cm/ libcfgmgr_client.so
	CfgMgr::CancelMonitor	取消检查。		
	CfgMgr::CleanStartConfig	清理指定启动描述文件的数据。		
	CfgMgr::CreateSnapshot	配置参数快照，记录当前所有持久化参数。		
	CfgMgr::CreateSnapshotToFile	指定快照路径进行快照。		
	CfgMgr::DeleteSnapshot	删除快照文件。		
	CfgMgr::Get	读取配置参数。		
	CfgMgr::GetAllStartConfig	获取当前所有的启动描述文件。		

接口分类	关键接口	接口简介	头文件	依赖的库文件
	CfgMgr::GetAllSnapshot	获取当前所有的快照文件名称及对应启动描述文件。		
	CfgMgr::GetStartConfig	获取当前正在使用的启动描述文件。		
	CfgMgr::Has	判断某参数是否存在。		
	CfgMgr::Load	按yaml文件加载参数并进行持久化。		
	CfgMgr::Monitor	检查指定参数。		
	CfgMgr::Param	获取参数。		
	CfgMgr::Reload	重新加载yaml文件。		
	CfgMgr::Reset	删除持久化数据。		
	CfgMgr::RevertSnapshot	按名称还原快照文件。		
	CfgMgr::RevertSnapshotFromFile	指定路径进行恢复快照。		
	CfgMgr::Set	设置参数。		
	CfgMgr::SetNextStartConfig	设置启动参考描述文件。		
	CfgMgr::SetUpgrade	使能系统重启后重新加载yaml文件。		

4.2 返回码列表

表 4-2 返回码列表

返回码（16进制）	含义	可能原因及解决方法
HIVA_SUCCESS=0x00	通用成功错误码	无
HIVA_FAILED=0x01	通用失败错误码	可能原因：公共错误码 解决方法：要具体分析，当前无明确的错误含义

返回码（16进制）	含义	可能原因及解决方法
HIVA_INPUT_NODE_NAME_ERROR=0xF20004	node名称异常	可能原因：输入的节点名不规范 解决方法：检查输入的节点名称信息
HIVA_NNENGINE_PARAMCHECK_FAILED=0xF20350	参数校验失败	可能原因：输入参数不正确 解决方法：检查参数的范围，模型的路径，模型文件的权限等
HIVA_NNENGINE_ACLINIT_FAILED=0xF20351	AscendCL初始化失败	可能原因：AscendCL模块内部出错 解决方法：重启进程进行重试
HIVA_NNENGINE_ACLSETDEVICE_FAILED=0xF20352	AscendCL setDevice失败	可能原因：AscendCL模块内部出错 解决方法：重启进程进行重试
HIVA_SERVICE_SUB_NULL=0xF20401	sub初始化失败	可能原因： - 指定的blockNum或dataSize异常 - 对应的共享内存初始化失败 解决方法：重启节点
HIVA_SERVICE_CALL_REQUEST_NULL=0xF20402	请求数据指针为空	可能原因：请求时传入的数据指针异常 解决方法：检查参数值，重新进行请求
HIVA_SERVICE_CALL_RESULT_NULL=0xF20403	响应数据指针为空	可能原因：请求时传入的数据指针异常 解决方法：检查参数值，重新进行请求
HIVA_SERVICE_CALL_RESULT_FAILED=0xF20404	服务端处理返回失败	可能原因：服务端的回调处理返回失败 解决方法：重新进行请求
HIVA_SERVICE_CALL_TIMEOUT=0xF20405	请求超时	可能原因：请求发出后等待时间超时 解决方法：重新进行请求

返回码（16进制）	含义	可能原因及解决方法
HIVA_NODE_PARAM_INVALID=0xF20500	发布节点初始化参数无效（序列化参数空、消息大小错误等）	可能原因： - 序列化函数赋值为空 - 标准消息的消息发布大小，大于最大值小于最小值 - BufferMessage消息mbuf指针为空 解决方法：检查参数值
HIVA_INPUT_NODE_NAME_ERROR=0xF20501	输入节点名字非法	可能原因：节点名称超长、remap之后超长 解决方法：检查参数值
HIVA_NODE_STRCPY_ERROR=0xF20502	注册DataMaster时节点名称拷贝失败	可能原因：创建队列时构造注册消息，字符串拷贝失败 解决方法：重启节点
HIVA_NODE_SERIALIZE_FAILED=0xF20510	发布者序列化失败	可能原因：数据与消息类型不匹配，mbuf数据被破坏 解决方法：发布者检查代码，注意拦截
HIVA_NODE_MSG_SIZE_INVALID=0xF20511	BufferMessage类型的消息/管理面类型消息发布大小不在正常范围内	可能原因：BufferMessage类型的消息/管理面类型消息发布大小不在正常范围内 解决方法：检查参数值
HIVA_NODE_CONNECT_DataMaster_FAILED=0xF20517	节点初始化时注册DataMaster失败	可能原因：DataMaster退出或与DataMaster通信失败 解决方法：重启节点
HIVA_NODE_GENERA_MSG_FAILED=0xF20518	节点初始化生成注册消息失败	可能原因： - 注册消息字符串拼接失败 - 平面选择环境变量错误 /proc/self/exe链接读取错误 解决方法：重启节点
HIVA_NODE_HAL_CREATE_GROUP_FAILED=0xF20520	节点初始化创建组失败	可能原因：驱动加载失败 解决方法：权限错误、进程退出

返回码（16进制）	含义	可能原因及解决方法
HIVA_NODE_BIND_CPU_FAIL D=0xF20521	节点初始化绑核失败	可能原因：硬件初始化错误：detach device失败，OS绑核失败 解决方法：重启节点
HIVA_CORE_SERIALIZE_FUNC EMPTY=0xF20A05	publish消息时序列化函数为空	可能原因：序列化函数为空 解决方法：重启节点
HIVA_CORE_BUFF_PTR_EMPTY Y=0xF20A06	publish消息时buffer的起始地址为空	可能原因：buffer为空 解决方法：重启节点
HIVA_TRANSPORT_MSG_TYPE _INVALID=0xF20B07	消息类型不正确	可能原因：消息类型不正确 解决方法：检查BufferMessage消息判断是否正确
HIVA_TRANSPORT_QUEUE_IN VALID=0xF20B08	获取queue info失败	可能原因：hiva not OK 解决方法：重启节点
HIVA_TRANSPORT_VFS_NOT SUPPORT_BUFF_MSG=0xF20B 09	vfs不支持buffer msg	可能原因：vfs不支持BufferMessage 解决方法：重新配置通信方式或者消息类型
HIVA_SERVICE_CLIENT_PTR_E MPTY=0xF20D00	service的客户端指针为空	可能原因：service的客户端指针为空 解决方法：重启节点
HIVA_SERVICE_SEVER_PTR_E MPTY=0xF20D01	service的服务端指针为空	可能原因：service的服务端指针为空 解决方法：重启节点
HIVA_SERVICE_STATE_INVALI D=0xF20D02	service的状态不正确	可能原因：service的状态不正确 解决方法：检查状态转换流程是否正确
HIVA_SERVICE_NODEPTR_EM PTY=0xF20D03	node指针为空	可能原因：node指针为空 解决方法：重启节点

4.3 类型定义

4.3.1 OpenHiva::Node 类型定义

OpenHiva::Node主要用来创建发布者（Publisher）、订阅者（Subscriber）、服务端（ServiceServer）、客户端（ServiceClient）对象，Node类接口定义如下：

```
namespace OpenHiva {
using RemappingsString = std::unordered_map<std::string, std::string>;
class Node {
public:
    Node(const std::string &name, const std::string &ns = std::string(), const RemappingsString&
remappingsStr = RemappingsString());
    ~Node();
    template <typename M>
    std::shared_ptr<Publisher> CreatePublisher(const std::string &topicName, const TopicOptions
&topicOptions = TopicOptions());
    template <typename M>
    std::shared_ptr<Subscriber> CreateSubscriber(const std::string &topicName, const
std::function<void(M)> &callback, const TopicOptions & topicOptions = TopicOptions());
    template <typename ReqType, typename ResType>
    std::shared_ptr<ServiceServer> CreateServer(const std::string &serviceName, const ServiceOptions
&serviceOptions, const std::function<bool(ReqType, ResType)> &serviceCallback);
    template <typename ReqType, typename ResType>
    std::shared_ptr<ServiceClient> CreateClient(const std::string &serviceName, const ServiceOptions
&serverOptions);
    std::vector<std::shared_ptr<Subscriber>> GetSubscribers();
    bool Ready();
    const std::string &GetName();
    const std::string &GetNamespace();
private:
    void Shutdown();
    std::shared_ptr<TopicManager> topicManagerPtr_;
    std::shared_ptr<NodeUri> nodeUriPtr_;
};
}
```

4.3.2 OpenHiva::TopicOptions 类型定义

用户在调用[4.5.2 OpenHiva::Node::CreatePublisher](#)和[4.5.3 OpenHiva::Node::CreateSubscriber](#)时都需要TopicOptions，TopicOptions是Topic消息一系列属性值的集合，包括话题通信方式以及各个通信方式下的属性配置。

除了通过yaml文件进行参数配置，可以通过一系列Build接口进行参数配置，定义如下：

```
namespace OpenHiva {
using TransportOptions = std::unordered_map<std::string, std::string>;
class TopicOptions {
public:
    TopicOptions();
    ~TopicOptions();
    template <typename M>
    TopicOptions &BuildMessageTraits();
    // maxMsgSize在SHM通信模式下，不能超过30000000，单位是bytes
    TopicOptions &BuildShmOptions(const uint32_t maxMsgSize, const uint32_t blockNum);
    TopicOptions &BuildQueueOptions(const uint32_t queueSize, const bool overwrite = false, const bool
queueFCFlag = false, const uint16_t queueTTL = 0U);
    TopicOptions &BuildGroupName(const std::string &groupName);
    TopicOptions &BuildTransportOptions(const std::string &transType, const TransportOptions &opts);
    TopicOptions &BuildTopicName(const std::string &topicName);
    template <typename M, enable_if_t<self_topic_options<!is_shared_ptr_topic_options<M>::value, int> = 0>
TopicOptions &BuildCallback(const std::function<void(M)> &callback); // 设置回调函数，本模板函数中M必
须为普通消息类型
    template <typename M, enable_if_t<self_topic_options<is_shared_ptr_topic_options<M>::value, int> = 0>
TopicOptions &BuildCallback(const std::function<void(M)> &callback); // 设置回调函数，本模板函数中M必
须为std::shared_ptr消息类型
    const std::vector<std::string> &GetTransportTypes();
    const std::unordered_map<std::string, OpenHiva::TransportOptions> &GetTransportOptions() const;
```

```
Hiva::AdvertiseOptions CreateAdvertiseOps() const;  
Hiva::SubscribeOptions CreateSubscribeOps() const;  
};  
}
```

4.3.3 OpenHiva::Publisher 类型定义

OpenHiva::Publisher由[4.5.2 OpenHiva::Node::CreatePublisher](#)接口创建，该接口返回的OpenHiva::Publisher对象已完成初始化。

OpenHiva::Publisher作为发布端，部分接口和成员变量定义如下：

```
namespace OpenHiva {  
class Publisher {  
public:  
    uint32_t Publish(const HivaMessage &message); // HivaMessage接口，HivaBuffer通过BufferMessage去发布  
    bool Ready() const; // 判断Publisher是否有效，若无效，调用Publish接口会返回错误码  
};  
}
```

4.3.4 OpenHiva::Subscriber 类型定义

OpenHiva::Subscriber由[4.5.3 OpenHiva::Node::CreateSubscriber](#)接口创建，该接口返回的OpenHiva::Subscriber对象已完成初始化。

OpenHiva::Subscriber作为订阅端，部分接口和成员变量定义如下：

```
namespace OpenHiva {  
class Subscriber {  
public:  
    bool Ready() const; // 判断Subscriber是否有效  
    uint32_t StartReadMessage(); // 开始调用回调处理消息  
    uint32_t StopReadMessage(); // 停止调用回调处理消息  
};  
}
```

4.3.5 OpenHiva::ServiceOptions 类型定义

OpenHiva::ServiceOptions用来描述Server-Client的属性，Server和Client端必须保持一致。

说明

ServiceOptions入参中的请求数据大小（reqMsgSize）、响应数据大小（resMsgSize）、数据缓存量（blockNum）不能为0，消息大小msgSize=reqDataSize或resDataSize+消息头（64字节），单个消息大小msgSize不能超过30M，blockNum*msgSize需小于300M。

```
namespace OpenHiva {  
class ServiceOptions {  
public:  
    ServiceOptions &BuildShmOptions(const uint32_t reqMsgSize, const uint32_t resMsgSize, const uint32_t blockNum);  
private:  
    template <typename ReqType, typename ResType>  
    ServiceOptions &BuildCallBack(const std::function<bool(ReqType, ResType)> &serviceCallback); // 设置Service回调  
    ServiceOptions &BuildServiceName(const std::string &srvName); // 设置Service名  
    std::string serviceName_; // service名  
    uint32_t reqDataSize_{0U}; // Request消息最大长度，不能超过30000000bytes  
    uint32_t resDataSize_{0U}; // Response消息最大长度，不能超过30000000bytes  
    uint32_t blockNum_{0U}; // 通信时缓存的block个数，描述可以最大缓存多少帧消息  
    std::function<bool(char8_t *, char8_t *)> callback_{nullptr};  
    friend class Node;  
};  
}
```

4.3.6 OpenHiva::ServiceServer 类型定义

OpenHiva::ServiceServer用作Server-Client的服务端，用来接收客户端发布的消息并将消息处理后返回消息给客户端。

OpenHiva::ServiceServer由[4.5.4 OpenHiva::Node::CreateServer](#)接口创建，该接口返回的OpenHiva::ServiceServer对象已完成初始化。

```
namespace OpenHiva {
class ServiceServer {
public:
    uint32_t Enable(); // 使能Service服务端，使服务端能接收客户端的消息并将处理结果返回
    uint32_t Disable(); // 去使能Service服务端，此时服务端不再接收客户端消息
    void Destroy(); // 释放ServiceServer申请的资源
private:
    std::shared_ptr<Hiva::ServiceServer> coreServerPtr_;
    friend class Node;
};
}
```

4.3.7 OpenHiva::ServiceClient 类型定义

OpenHiva::ServiceClient用作Server-Client的客户端，向服务端发送请求消息，并等待服务端处理结果。

OpenHiva::ServiceClient由[4.5.5 OpenHiva::Node::CreateClient](#)接口创建，该接口返回的OpenHiva::ServiceClient对象已完成初始化。

```
namespace OpenHiva {
class ServiceClient {
public:
    template <typename ReqType, typename ResType>
    uint32_t InvokeService(ReqType req, ResType &res, const uint32_t timeout = 5000U); // 向Server发布
    ReqType类型的消息并等待ResType类型的消息返回
    void Destroy(); // 释放ServiceClient申请的资源
};
}
```

4.3.8 OpenHiva::ScheduleGroup 类型定义

调用[4.4.1 OpenHiva::Init](#)接口时需传入线程组参数，用户需提前创建线程组（ScheduleGroup）。OpenHiva::ScheduleGroup定义如下，包含了线程组名（groupName）和调度类型（scheduleType）。

```
namespace OpenHiva {
// 线程组调度类型
enum ScheduleType {
    UNBIND_AICPU = 0, // 非绑核线程，非确定性调度
    BIND_AICPU, // DSF-AICPU绑核线程，确定性调度
    USER_DEFINED, // 用户自己起线程，Hiva框架不起，与SpinOnce配合使用
};
// 线程组结构体
struct ScheduleGroup {
    std::string groupName; //线程组名称，需要在每个进程中保证线程组名称唯一
    ScheduleType scheduleType; // 调度类型
};
}
```

4.3.9 OpenHiva::BufferMessage 类型定义

OpenHiva::BufferMessage是一种特殊的HivaMessage，里面封装了OpenHiva::HivaBuffer的序列化和反序列化，可用于发布-订阅。

```
namespace Hiva {
namespace StdMsgs {
```

```
class BufferMessage : public Hiva::HivaMessage {
public:
    BufferMessage() {}
    BufferMessage(const OpenHiva::HivaBuffer &hivaBuffer) : data(hivaBuffer) {}
    ~BufferMessage() override {}
    bool Serialize(Hiva::HivaSerializer &serializer) const override
    {
        serializer << reinterpret_cast<uint64_t>(data.mbufPtr_);
        return serializer.Ok();
    }
    bool Deserialize(Hiva::HivaSerializer &serializer) override
    {
        uint64_t bufferAddr;
        serializer >> bufferAddr;
        data.mbufPtr_ = reinterpret_cast<Mbuf *>(bufferAddr);
        return serializer.Ok();
    }
    uint32_t GetSerializedLength(Hiva::HivaSerializer &serializer) const override
    {
        (void)serializer;
        return 8U;          // sizeof hivabuffer
    }
    const Hiva::HivaTime &GetTimeStamp() const override
    {
        static Hiva::HivaTime timestamp{0, 0};
        return timestamp;
    }
    virtual uint32_t GetSequenceld() const override
    {
        return 0U;
    }
    static bool HasHeader()
    {
        return false;
    }
    static std::string GetDataType()
    {
        return "StdMsgs/BufferMessage";
    }
    static std::string GetMD5Sum()
    {
        return "992ce8a1687cec8c8bd883ec73ca41d1";
    }
    static std::string GetDefinition()
    {
        return "Mbuf *mbufPtr";
    }
    OpenHiva::HivaBuffer data;
};
using OpenHiva::BufferMessage = Hiva::StdMsgs::BufferMessage;
```

4.3.10 OpenHiva::HivaBuffer 类型定义

HivaBuffer是确定性调度框架特有的内存类型，用户可将数据拷贝到HivaBuffer，并调用消息发布Publish接口将HivaBuffer发布出去。

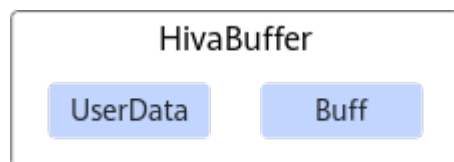
```
namespace OpenHiva {
class HivaBuffer{
public:
    explicit HivaBuffer ();
    void ConstructFromMbuf(struct Mbuf *mbufPtr);          // 内部使用，构造HivaBuffer
    uint32_t GetBuff (void *&dataPtr, size_t &dataLen);    // 获取Buff数据区写入数据的起始地址和长度
    uint32_t SetBuffDataLen(const size_t dataLen);        // 设置已写入Buff数据区的数据长度
    uint32_t SetUserData (const UserData &data);          // 往UserData数据区写入数据
    uint32_t GetUserData (UserData &data);                // 获取UserData数据区的数据
    uint32_t CopyHivaBuffer(HivaBuffer &hivaBuffer);      // 拷贝HivaBuffer输出新对象newBuffer
    uint32_t FreeHivaBuffer();                             // HivaBuffer释放
};
```



```
private:
    Mbuf * mbufPtr_;
    friend HivaBuffPool;
    friend Publisher;
    friend BufferMessage;
};
}
```

其内存模型如图4-1：

图 4-1 HivaBuffer 内存示意图



- UserData类型定义如下：

```
struct OpenHiva::UserData {
    char[96] userData; // 96bytes
};
```

用户可通过SetUserData接口设置HivaBuffer内的UserData数据，也可通过GetUserData获取HivaBuffer内的UserData数据。

- Buffer使用注意事项：

- 用户可通过GetBuff获取HivaBuffer内Buff数据区写入数据的起始地址和长度。
- 用户可以通过往Buff数据区写入数据后再通过SetBuffDataLen设置已写入数据的长度。
- 用户可以调用CopyHivaBuffer，输出新的HivaBuffer对象newBuffer。newBuffer与原HivaBuffer共享Buff数据区，此时Buff数据区的引用计数在原基础上加1；newBuffer与原HivaBuffer拥有各自的UserData数据区，newBuffer不会拷贝原HivaBuffer的UserData数据区。
- Buff数据区可以在HivaBuffer之间共享，每多一个用户，Buff数据区引用计数加1；Buff数据区与HivaBuffer生命周期不同，当Buff数据区引用计数为0时，Buff数据区被释放。
- UserData数据区不能在HivaBuffer之间共享，每个HivaBuffer都拥有各自的UserData数据区，UserData数据区与HivaBuffer生命周期相同。
- 当用户需要释放HivaBuffer时，可调用FreeHivaBuffer接口，此接口会将HivaBuffer释放（包括UserData数据区），并将HivaBuffer所包含的Buff数据区引用计数减1，当引用计数为0时，Buff数据区才会被释放。

4.3.11 OpenHiva::HivaBufferPool 类型定义

OpenHiva::HivaBufferPool是用于申请OpenHiva::HivaBuffer的类，定义如下：

```
class OpenHiva::HivaBufferPool {
public:
    HivaBufferPool();
    ~HivaBufferPool() = default;
    uint32_t InitMemoryPool(const std::string &poolName, const uint32_t blockSize, const uint32_t
blockNum);
    void DestroyMemoryPool();
    uint32_t Allocate(HivaBuffer &hivaBuffer) const;    // 从内存池中分配HivaBuffer
private:
    mempool_t *pHandle_;
};
```

HivaBufferPool使用注意事项:

- 用户可以通过HivaBufferPool的构造函数传入poolName（内存池命名）、blockSize（内存块大小，与HivaBuffer的Buff数据区大小一致）、blockNum（内存块个数）。
- 用户可以通过Allocate接口从内存池申请HivaBuffer。

4.3.12 OpenHiva::HivaMessage 类型定义

OpenHiva::HivaMessage是框架发布-订阅通用的消息类型，里面包含了消息的序列化和反序列化接口。

```
namespace Hiva {
namespace StdMsgs {
class HivaMessage {
public:
    HivaMessage() = default;
    virtual ~HivaMessage() = default;
    virtual bool Serialize(HivaSerializer &serializer) const = 0;
    virtual bool Deserialize(HivaSerializer &serializer) = 0;
    virtual uint32_t GetSerializedLength(HivaSerializer &serializer) const = 0;
    virtual const Hiva::HivaTime &GetTimeStamp() const = 0; // 获取消息生成时的时间戳，保存在消息头里，
    只有在HasHeader接口为true时才有意义
    virtual uint32_t GetSequenceId() const = 0; // 获取消息生成的序列号，保存在消息头里，只有在
    HasHeader接口为true时才有意义
    // 如下接口描述消息属性，必须在各个子类实现
    static bool HasHeader(); // 是否存在消息头
    static std::string GetDataType(); // 消息类型
    static std::string GetMD5Sum(); // 消息的md5值，每个消息的md5值都不同
    static std::string GetDefinition(); // 消息的定义，定义里面详细的描述了消息里面的成员变量
};
}
}
using OpenHiva::HivaMessage = Hiva::StdMsgs::HivaMessage;
```

4.3.13 OpenHiva::HivaMessageTraits 类型定义

OpenHiva::HivaMessageTraits结构体用来描述消息属性，每个OpenHiva::HivaMessage都有相应的MessageTraits。

```
struct OpenHiva::HivaMessageTraits {
    std::string md5sum;
    std::string dataType;
    bool hasHeader;
    std::string msgDef;
};
```

- md5sum：描述消息的md5值。
- datatype：描述消息类型。
- hasHeader：描述消息是否存在消息头。
- msgDef：描述消息类型的消息定义。

4.3.14 Hiva::Time 类型定义

- Hiva::BaseTime
Time（时间点）基类，实现了基础运算接口。HivaTime、WallTime、SteadyTime均继承BaseTime，支持BaseTime的所有运算操作。
BaseDuration类的定义请参见[4.3.15 Hiva::Duration类型定义](#)。

```
namespace Hiva {
/* Base class for Time implementations. */
class BaseTime {
```

```
public:
    BaseTime();
    BaseTime(const uint32_t secVal, const uint32_t nsecVal);
    explicit BaseTime(const double doubleSec);
    virtual ~BaseTime() = default;
    double ToSec() const;
    bool FromSec(const double doubleSec);
    bool FromTimespec(const timespec &timeVal);
    uint64_t ToNSec() const;
    bool FromNSec(const uint64_t uint64Nsec);
    bool IsZero() const;
    BaseDuration operator-(const BaseTime &rhs) const;
    BaseTime operator+(const BaseDuration &rhs) const;
    BaseTime operator-(const BaseDuration &rhs) const;
    BaseTime &operator+=(const BaseDuration &rhs);
    BaseTime &operator-=(const BaseDuration &rhs);
    bool operator==(const BaseTime &rhs) const;
    bool operator!=(const BaseTime &rhs) const;
    bool operator<(const BaseTime &rhs) const;
    bool operator>(const BaseTime &rhs) const;
    bool operator<=(const BaseTime &rhs) const;
    bool operator>=(const BaseTime &rhs) const;
    uint32_t sec;
    uint32_t nsec;
};
std::ostream &operator<<(std::ostream &outStream, const BaseTime &rhs);
}
```

- **Hiva::HivaTime**

继承BaseTime，支持BaseTime的所有运算操作。在使能仿真时间的时候为仿真时间，在不使能仿真时间的时候为WallTime（墙上时间）。

```
namespace Hiva {
/* Time may either represent wall clock time or simulation time. */
class HivaTime : public BaseTime {
public:
    HivaTime();
    HivaTime(const uint32_t secVal, const uint32_t nsecVal) noexcept;
    explicit HivaTime(const double doubleSec) noexcept;
    HivaTime(const BaseTime &timeVal);
    ~HivaTime() override = default;
    static bool UseSystemTime();
    static bool IsSimTime();
    static bool IsSystemTime();
    static HivaTime Now();
    static void SetNow(const HivaTime &newSimTime);
    static void Init();
    static void Shutdown();
    static bool GetState();
    static bool IsValid();
    static bool WaitForValid();
    static bool WaitForValid(const Hiva::WallDuration &timeout);
    static bool SleepUntil(const HivaTime &endTime);
private:
    static bool stopped;
    static std::mutex simTimeMutex;
    static bool initialized;
    static bool useSimTime;
    static HivaTime simTime;
};
}
```

- **Hiva::WallTime**

墙上时间，即CLOCK_REALTIME，继承BaseTime，支持BaseTime的所有运算操作。

```
namespace Hiva {
class WallTime : public BaseTime {
public:
    WallTime();
};
}
```

```
WallTime(const uint32_t secVal, const uint32_t nsecVal);
explicit WallTime(const double doubleSec);
WallTime(const BaseTime &timeVal);
~WallTime() override = default;
static WallTime Now();
static bool SleepUntil(const WallTime &endTime);
static bool IsSystemTime();
};
}
```

- Hiva::SteadyTime

单独递增时间，即CLOCK_MONOTONIC，不受时间跳变影响。继承BaseTime，支持BaseTime的所有运算操作。

```
namespace Hiva {
class SteadyTime : public BaseTime {
public:
    SteadyTime();
    SteadyTime(const uint32_t secVal, const uint32_t nsecVal);
    explicit SteadyTime(const double doubleSec);
    SteadyTime(const BaseTime &timeVal);
    ~SteadyTime() override = default;
    static SteadyTime Now();
    static bool SleepUntil(const SteadyTime &endTime);
    static bool IsSystemTime();
};
}
```

4.3.15 Hiva::Duration 类型定义

- Hiva::BaseDuration

duration（时长）基类，实现了基础运算接口，duration=time1-time2。HivaDuration、WallDuration均继承BaseDuration，支持BaseTime的所有运算操作，SteadyTime也是使用WallDuration。

```
namespace Hiva {
/* Base class for Duration implementations. */
class BaseDuration {
public:
    BaseDuration();
    explicit BaseDuration(const double doubleSec);
    BaseDuration(const int32_t secVal, const int32_t nsecVal);
    virtual ~BaseDuration() = default;
    bool FromSec(const double doubleSec);
    bool FromNSec(const int64_t int64Nsec);
    double ToSec() const;
    int64_t ToNSec() const;
    bool IsZero() const;
    BaseDuration operator+(const BaseDuration &rhs) const;
    BaseDuration operator-(const BaseDuration &rhs) const;
    BaseDuration operator-() const;
    BaseDuration operator*(const double scale) const;
    BaseDuration &operator+=(const BaseDuration &rhs);
    BaseDuration &operator-=(const BaseDuration &rhs);
    BaseDuration &operator*=(const double scale);
    bool operator==(const BaseDuration &rhs) const;
    bool operator!=(const BaseDuration &rhs) const;
    bool operator>(const BaseDuration &rhs) const;
    bool operator<(const BaseDuration &rhs) const;
    bool operator>=(const BaseDuration &rhs) const;
    bool operator<=(const BaseDuration &rhs) const;
    int32_t sec;
    int32_t nsec;
};
std::ostream &operator<<(std::ostream &outStream, const BaseDuration &rhs);
}
```

- Hiva::HivaDuration

Hiva时长，继承BaseDuration，支持BaseDuration的所有运算操作。

```
namespace Hiva {  
/* Duration representation for use with the Time class. */  
class HivaDuration : public BaseDuration {  
public:  
    HivaDuration();  
    explicit HivaDuration(const double doubleSec);  
    explicit HivaDuration(const BaseRate &rateVal);  
    HivaDuration(const BaseDuration &durationVal);  
    HivaDuration(const int32_t secVal, const int32_t nsecVal);  
    ~HivaDuration() override = default;  
    bool Sleep() const;  
};  
}
```

- Hiva::WallDuration

系统时长，继承BaseDuration，支持BaseDuration的所有运算操作。

```
namespace Hiva {  
/* Duration representation for use with the WallTime class. */  
class WallDuration : public BaseDuration {  
public:  
    WallDuration();  
    explicit WallDuration(const double doubleSec);  
    explicit WallDuration(const BaseRate &rateVal);  
    WallDuration(const BaseDuration &durationVal);  
    WallDuration(const int32_t secVal, const int32_t nsecVal);  
    ~WallDuration() override = default;  
    bool Sleep() const;  
};  
}
```

4.3.16 Hiva::Rate 类型定义

- Hiva::BaseRate

Rate（周期）基类，常用于循环中的sleep，使得任务按固定周期执行（sleep时会减去任务执行耗时）。

```
namespace Hiva {  
/* Class to help run loops at a desired frequency */  
class BaseRate {  
public:  
    BaseRate(const double frequency);  
    explicit BaseRate(const BaseDuration &durationVal);  
    virtual ~BaseRate() = default;  
    virtual bool Sleep() = 0;  
    virtual void Reset() = 0;  
    BaseDuration CycleTime() const;  
    BaseDuration ExpectedCycleTime() const;  
protected:  
    BaseDuration expectedCycleTime_;  
    BaseDuration actualCycleTime_;  
};  
}
```

- Hiva::HivaRate

继承BaseRate，基于HivaTime进行sleep等待。

```
namespace Hiva {  
/* Class to help run loops at a desired frequency */  
class HivaRate : public BaseRate {  
public:  
    HivaRate(const double frequency);  
    explicit HivaRate(const BaseDuration &durationVal);  
    ~HivaRate() override = default;  
    bool Sleep() override;  
};  
}
```

```
void Reset() override;
private:
    HivaTime start_;
};
}

• Hiva::WallRate
继承BaseRate，基于WallTime进行sleep等待。

namespace Hiva {
/* Class to help run loops at a desired frequency */
class WallRate : public BaseRate {
public:
    WallRate(const double frequency);
    explicit WallRate(const BaseDuration &durationVal);
    ~WallRate() override = default;
    bool Sleep() override;
    void Reset() override;
private:
    WallTime start_;
};
}
```

4.4 初始化、退出和调度接口

4.4.1 OpenHiva::Init

函数功能

初始化框架资源，如节点、线程组等。

函数原型

```
uint32_t OpenHiva::Init(const int &argc, char **argv, const
std::vector<ScheduleGroup> &schedGroupVec)
```

参数说明

参数名	输入/输出	说明
argc	输入	传入的参数个数。
argv	输入	传入的字符串数组地址。
schedGroup Vec	输入	线程组，ScheduleGroup类型，详见 4.3.8 OpenHiva::ScheduleGroup类型定义 。

返回值

返回uint32_t类型，成功返回HIVA_SUCCESS，该值为0；其他值为失败。

使用说明

由APP调用初始化接口执行OpenHiva节点的初始化动作，包括框架运行所需的各类资源申请、数据的读取、参数解析等。

注意事项

- 本接口不可重入，一个进程只能调用一次。
- 支持同步调用。
- 需要在Node实例化之前调用。
- 未调用OpenHiva::Init初始化或调用OpenHiva::Init失败时，OpenHiva::Ready返回false；调用OpenHiva::Init成功后，OpenHiva::Ready返回true。

4.4.2 OpenHiva::Ready

函数功能

判断OpenHiva框架状态。

函数原型

bool OpenHiva::Ready()

参数说明

无

返回值

返回bool类型，当进程处于使能状态，返回true；当节点Shutdown或初始化失败时，返回false。

使用说明

可用来判断进程中节点状态是否正常。

注意事项

- 可重入，同步接口。
- 未调用OpenHiva::Init初始化或调用OpenHiva::Init失败时，OpenHiva::Ready返回false；调用OpenHiva::Init成功后，OpenHiva::Ready返回true。
- 调用OpenHiva::Shutdown后，OpenHiva::Ready返回false。

4.4.3 OpenHiva::Shutdown

函数功能

终止OpenHiva框架运行，退出进程并释放资源。

函数原型

void OpenHiva::Shutdown()

参数说明

无

返回值

无

使用说明

无

注意事项

- 本接口不可重入，同步接口。
- 进程退出的接口，调用该接口后，[4.4.4 OpenHiva::WaitForShutdown](#)接口会终止，框架会释放资源并退出。
- 调用OpenHiva::Shutdown后，OpenHiva::Ready返回false，框架发布和订阅会终止。

4.4.4 OpenHiva::WaitForShutdown

函数功能

阻塞式调用，防止主线程提前退出，框架退出前会释放资源。

函数原型

`void OpenHiva::WaitForShutdown()`

参数说明

无

返回值

无

使用说明

无

注意事项

主线程中，调用WaitForShutdown接口释放资源，防止主线程提前退出。当进程异常终止时，WaitForShutdown接口内部会自动调用[4.4.3 OpenHiva::Shutdown](#)接口进行资源清理。

4.4.5 OpenHiva::SpinOnce

函数功能

调度groupName指定的group，订阅的Topic在指定group内会被调度处理消息。

函数原型

`uint32_t OpenHiva::SpinOnce(const std::string &groupName)`

参数说明

参数名	输入/输出	说明
groupName	输入	线程组名字。

返回值

返回uint32_t类型，成功返回HIVA_SUCCESS，该值为0；其他值为失败。

使用说明

无

注意事项

- groupName必须与[4.4.1 OpenHiva::Init](#)接口入参schedGroupVec里某个ScheduleGroup成员的groupName匹配，该groupName对应的scheduleType必须为USER_DEFINED，否则SpinOnce会返回失败。
- 建议不要直接在主线程中调用。消息比较大的场景下，调用可能会频繁触发缺页，导致CPU飙升，推荐起线程进行循环调用。

4.5 OpenHiva::Node 接口

4.5.1 OpenHiva::Node

函数功能

构造函数，完成Node的初始化工作。

函数原型

`void OpenHiva::Node(const std::string &nodeName, const std::string& nodeNameSpace = std::string(), const M_string& remappings = M_string())`

参数说明

参数名	输入/输出	说明
nodeName	输入	设置节点的名字。
nodeNameSpace	输入	设置节点的命名空间。

参数名	输入/输出	说明
remappings	输入	map<string, string>类型的参数，若其中包含__name:="a"，__ns:="b"，则节点名为"/b/a"。

返回值

无

使用说明

无

注意事项

- 本接口不可重入，同步接口。
- 在调用CreatePublisher、CreateSubscriber、CreateServiceServer、CreateServiceClient接口之前必须先创建Node。

4.5.2 OpenHiva::Node::CreatePublisher

函数功能

创建发布者。

函数原型

```
template <typename M> std::shared_ptr<OpenHiva::Publisher>  
OpenHiva::Node::CreatePublisher(const std::string &topicName, const  
TopicOptions &topicOptions = TopicOptions())
```

参数说明

参数名	输入/输出	说明
topicName	输入	待发布的Topic名称，应小于128字节。
topicOptions	输入	创建发布者的参数列表，详见 4.3.2 OpenHiva::TopicOptions类型定义 。

返回值

返回OpenHiva::Publisher对象，可以用来发布消息。

使用说明

无

注意事项

- 本接口不可重入，同步接口。
- 同一个Topic只能有一个OpenHiva::Publisher，不允许对同一个Topic重复调用CreatePublisher接口。
- 必须在执行完OpenHiva::Init后使用，通过OpenHiva::Node调用。
- M表示消息类型，必须是HivaMessage的子类。
- CreatePublisher和CreateSubscriber注册的同一Topic的消息类型应该保持一致。
- Topic数量不超过500个，全系统CreatePublisher和CreateSubscriber的数量不允许超过2048个。
- TopicName长度不允许超过128字节，首字符只能为字母、'/'，其他字符只能为字母、数字、'/'、'_'。

4.5.3 OpenHiva::Node::CreateSubscriber

函数功能

创建订阅者。

函数原型

```
template <typename M> std::shared_ptr<OpenHiva::Subscriber>  
OpenHiva::Node::CreateSubscriber(const std::string &topicName, const  
std::function<void(M)> &callback, const OpenHiva::TopicOptions  
&topicOptions = TopicOptions())
```

参数说明

参数名	输入/输出	说明
topicName	输入	待订阅的Topic名称，应小于128字节。
callback	输入	注册Subscriber的回调函数，M为消息类型。一般在收到消息后调用callback。
topicOptions	输入	Topic属性配置，详见 4.3.2 OpenHiva::TopicOptions类型定义 。

返回值

返回OpenHiva::Subscriber对象，可以用来订阅消息。

使用说明

APP通过CreateSubscriber函数，订阅指定Topic的发布数据，并为数据创建指定深度的缓存队列。在该Topic的数据发布后，OpenHiva收到数据后，会在groupName所在的线程组内，调用回调函数。

注意事项

- 本接口不可重入，同步接口。
- 必须在执行完OpenHiva::Init后使用，通过OpenHiva::Node调用。
- 对于相同Topic的数据，OpenHiva保证顺序处理；对于不同Topic的数据，OpenHiva不保证处理时序。
- 对于确定性调度的Topic回调，不允许在回调中（包括回调中调用的其它函数）加锁。
- 用户在调用Hiva::Node::CreateSubscriber接口时会设置TopicOptions的成员groupName。
- M表示消息类型，必须是HivaMessage的子类。
- CreatePublisher和CreateSubscriber注册的同一Topic的消息类型应该保持一致。
- Topic数量不超过500个，全系统CreatePublisher和CreateSubscriber的数量不允许超过2048个。
- TopicName长度不允许超过128字节，首字符只能为字母、'/'，其他字符只能为字母、数字、'/'、'_'。

4.5.4 OpenHiva::Node::CreateServer

函数功能

创建Service的服务端。

函数原型

```
template <typename ReqType, typename ResType>
std::shared_ptr<OpenHiva::ServiceServer> CreateServer(const std::string
&serviceName, const OpenHiva::ServiceOptions &serviceOptions, const
std::function<bool(ReqType, ResType)> &serviceCallback)
```

参数说明

参数名	输入/输出	说明
serviceName	输入	服务名。
serviceOptions	输入	Service属性配置，详见 4.3.5 OpenHiva::ServiceOptions类型定义 。
serviceCallback	输入	Server收到消息后调用的回调函数，用于处理Client端发布的消息。

返回值

返回OpenHiva::ServiceServer对象，可通过该对象完成服务端的初始化及资源释放。

使用说明

无

注意事项

- 服务名必须全局唯一，不可重复不可同名。
- 对于同一个服务，客户端和服务端的参数serviceName、serviceOptions必须完全一致。
- 调用时必须确保入参正确，按约束使用，回调函数必须判断长度，不可越界。
- 按serviceName拼接其他字符串来创建共享内存文件，需符合文件命名规则。
- 一个服务端只起一个线程，回调处理是串行的，不同服务的回调处理是并行的。
- 服务端处理太慢可能会导致客户端超时。
- 由于涉及共享内存文件读写，服务端进程和客户端进程的用户权限必须相同，否则可能通信失败。
- ReqType描述请求的消息类型，必须是HivaMessage的子类。
- ResType描述响应的消息类型，必须是HivaMessage的子类。

4.5.5 OpenHiva::Node::CreateClient

函数功能

创建Service的客户端。

函数原型

```
template <typename ReqType, typename ResType>
std::shared_ptr<OpenHiva::ServiceClient> CreateClient(const std::string
&serviceName, const ServiceOptions &serviceOptions)
```

参数说明

参数名	输入/输出	说明
serviceName	输入	服务名。
serviceOptions	输入	Service属性配置，详见 4.3.5 OpenHiva::ServiceOptions类型定义 。

返回值

返回OpenHiva::ServiceClient对象，可通过该对象完成客户端的初始化及资源释放，也可向服务端发送消息并获取服务端响应的结果。

使用说明

无

注意事项

- 本接口不可重入，同步接口。
- ReqType描述请求的消息类型，必须是HivaMessage的子类。

- ResType描述响应的消息类型，必须是HivaMessage的子类。

4.5.6 OpenHiva::Node::Ready

函数功能

判断Node是否关闭。

函数原型

```
bool OpenHiva::Node::Ready()
```

参数说明

无

返回值

返回bool类型，若为true，表示Node处于使能状态；若为false，表示Node处于去使能状态。

使用说明

无

注意事项

可重入，同步接口。

4.5.7 OpenHiva::Node::GetName

函数功能

获取Node的名字。

函数原型

```
const std::string &OpenHiva::Node::GetName()
```

参数说明

无

返回值

返回节点名（std::string类型），在所有的框架进程中全局唯一。

使用说明

无

注意事项

可重入，同步接口。

4.5.8 OpenHiva::Node::GetNameSpace

函数功能

获取Node的域名。

函数原型

const std::string &OpenHiva::Node::GetNameSpace()

参数说明

无

返回值

返回节点的NameSpace（std::string类型）。

使用说明

无

注意事项

可重入，同步接口。

4.6 OpenHiva::TopicOptions 接口

4.6.1 OpenHiva::TopicOptions::BuildShmOptions

函数功能

设置SHM或DSF通信分配的内存块大小和内存块个数。

函数原型

OpenHiva::TopicOptions &OpenHiva::TopicOptions::BuildShmOptions(const uint32_t blockSize, const uint32_t blockNum)

参数说明

参数名	输入/输出	说明
blockSize	输入	分配的内存块大小，blockSize必须≥最大消息长度。SHM和DSF通信会根据blockSize分配内存块。

参数名	输入/输出	说明
blockNum	输入	分配的内存块个数，内存块大小是blockSize。

返回值

返回设置blockSize和blockNum后的TopicOptions对象。

使用说明

无

注意事项

- 本接口不可重入，同步接口。
- 一个Topic发布和订阅的blockSize和blockNum必须相同。
- 创建Publisher或Subscriber时，若用户未调用该接口，会从配置文件中查找Topic相关配置，若配置文件也无配置，则使用默认值。
- SHM通信方式，blockSize不能超过30000000（bytes）。

4.6.2 OpenHiva::TopicOptions::BuildQueueOptions

函数功能

设置DSF通信的队列参数。

函数原型

OpenHiva::TopicOptions &OpenHiva::TopicOptions::BuildQueueOptions(const uint32_t queueDepth, const bool overwrite = true, const bool queueFCFlag = false, const uint32_t queueTTL = 1000U)

参数说明

参数名	输入/输出	说明
queueDepth	输入	设置DSF通信发布或订阅队列长度。
overwrite	输入	DSF通信是否会在队列里面写覆盖。若为true，队列满了后会丢弃队首的消息，新消息会进入队尾。
queueFCFlag	输入	DSF通信是否流控。若为false，则不开启流控；否则开启流控。
queueTTL	输入	仅当queueFCFlag为true时，开启流控，该参数生效。 描述消息从入队到出队的时间差，若时间差≤queueTTL，会在出队时将消息丢弃。

返回值

返回设置QueueOptions后的TopicOptions对象。

使用说明

该接口用于设置DSF通信队列相关的属性。

注意事项

- 本接口不可重入，同步接口。
- 使用DSF通信时必须调用该接口设置队列长度。

4.6.3 OpenHiva::TopicOptions::BuildGroupName

函数功能

设置Topic所属的groupName。

函数原型

OpenHiva::TopicOptions &OpenHiva::TopicOptions::BuildGroupName(const std::string &groupName)

参数说明

参数名	输入/输出	说明
groupName	输入	组名。

返回值

返回设置groupName后的TopicOptions对象。

使用说明

无

注意事项

- 本接口不可重入，同步接口。
- 若用户不调用该接口，会默认为DSF通信。

4.6.4 OpenHiva::TopicOptions::BuildTransportOptions

函数功能

设置Topic的通信方式。

函数原型

```
OpenHiva::TopicOptions
&OpenHiva::TopicOptions::BuildTransportOptions(const std::string
&transType, const TransportOptions &opts = TransportOptions())
```

参数说明

参数名	输入/输出	说明
transType	输入	通信类型，目前支持以下几种： • DDS通信：使用fastdds通信。 • DSF通信：OpenHiva特有的高性能零拷贝的通信方式。 • SHM通信：使用posix共享内存接口进行通信。
opts	输入	TransportOptions为std::unordered_map<std::string, std::string>类型，当前版本可忽略。

返回值

返回设置通信方式后的TopicOptions对象。

使用说明

无

注意事项

- 本接口不可重入，同步接口。
- 若用户不调用该接口，会默认为DSF通信。

4.6.5 OpenHiva::TopicOptions::BuildDDSQos

函数功能

设置Topic所属DDS通信的QoS策略。

函数原型

```
OpenHiva::TopicOptions &OpenHiva::TopicOptions::BuildDDSQos(const
DdsQosOptions & qosOptions)
```

参数说明

参数名	输入/输出	说明
qosOptions	输入	DDS消息保存策略参数，数据结构为DdsQosOptions，具体定义如下： <pre>struct DdsQosOptions { QosHistoryPolicy history; std::size_t depth; QosReliabilityPolicy reliability; QosDurabilityPolicy durability; QosDuration deadline; QosDuration lifespan; QosLivelinessPolicy liveliness; QosDuration leaseDuration; };</pre> 相关的数据类型或数据结构如下： QosHistoryPolicy enum class QosHistoryPolicy: uint32_t { DDS_QOS_HISTORY_POLICY_KEEP_LAST, DDS_QOS_HISTORY_POLICY_KEEP_ALL, DDS_QOS_HISTORY_POLICY_SYSTEM_DEFAULT, DDS_QOS_HISTORY_POLICY_UNKNOWN }; QosReliabilityPolicy enum class QosReliabilityPolicy: uint32_t { DDS_QOS_RELIABILITY_POLICY_RELIABLE, DDS_QOS_RELIABILITY_POLICY_BEST_EFFORT, DDS_QOS_RELIABILITY_POLICY_SYSTEM_DEFAULT, DDS_QOS_RELIABILITY_POLICY_UNKNOWN }; QosDurabilityPolicy enum class QosDurabilityPolicy: uint32_t { DDS_QOS_DURABILITY_POLICY_VOLATILE, DDS_QOS_DURABILITY_POLICY_TRANSIENT_LOCAL, DDS_QOS_DURABILITY_POLICY_SYSTEM_DEFAULT, DDS_QOS_DURABILITY_POLICY_UNKNOWN }; QosLivelinessPolicy enum class QosLivelinessPolicy: uint32_t { DDS_QOS_LIVELINESS_POLICY_AUTOMATIC, DDS_QOS_LIVELINESS_POLICY_MANUAL_BY_TOPIC, DDS_QOS_LIVELINESS_POLICY_SYSTEM_DEFAULT, DDS_QOS_LIVELINESS_POLICY_UNKNOWN }; QosDuration struct QosDuration { // Seconds uint64_t sec; // Nanoseconds uint64_t nsec; };

返回值

返回设置QoS策略后的TopicOptions对象。

使用说明

需要先配置DDS通信方式，该接口才会生效。

注意事项

- 本接口不可重入，同步接口。
- 若用户不调用该接口，则使用QoS系统默认值。

4.7 OpenHiva::Publisher 接口

4.7.1 OpenHiva::Publisher::Publisher

函数功能

发布端的默认构造函数。

函数原型

OpenHiva::Publisher::Publisher()

参数说明

无

返回值

无

使用说明

不需要显示创建，只需通过OpenHiva::Node调用CreatePublisher创建。

注意事项

本接口不可重入，同步接口。

4.7.2 OpenHiva::Publisher::Publish

函数功能

消息发布函数。

函数原型

- 支持用户输入序列化函数
uint32_t OpenHiva::Publisher::Publish(const std::function<bool (uint8_t *, uint32_t)> &serializeFunc, const uint32_t bufferSize)
- 支持HivaMessage消息发布
uint32_t OpenHiva::Publisher::Publish(const HivaMessage &message)

参数说明

- 支持用户输入序列化函数

参数名	输入/输出	说明
serializeFunc	输入	序列化回调函数。
bufferLength	输入	消息序列化的长度。

- 支持HivaMessage消息发布

参数名	输入/输出	说明
message	输入	用户要发布的消息，详见 4.3.12 OpenHiva::HivaMessage类型定义 。

返回值

返回uint32_t类型，成功返回HIVA_SUCCESS，该值为0；其他值为失败。

使用说明

通过Publisher对象调用，在Publish之前，必须保证Topic已经通过CreatePublisher接口声明。

注意事项

- 本接口不可重入，同步接口。
- 不允许多个线程同时进行publish。
- SHM通信模式下，消息大小不能超过30000000（ bytes ）。
- 使用uint32_t OpenHiva::Publisher::Publish(const HivaMessage &message)接口时，消息类型必须继承HivaMessage，必须实现GetMessageType，GetMd5Sum等static接口。

4.7.3 OpenHiva::Publisher::Ready

函数功能

判断Publisher是否处于工作状态。

函数原型

bool OpenHiva::Publisher::Ready()

参数说明

无

返回值

返回bool类型，若为true，表示发布者状态正常；否则异常。

使用说明

判断Publisher是否有效，若无效则不再发布消息。

注意事项

- 可重入，同步接口。
- 当Publisher未被初始化或Publisher处于退出状态时，该接口返回false。

4.8 OpenHiva::Subscriber 接口

4.8.1 OpenHiva::Subscriber::Subscriber

函数功能

订阅端的默认构造函数。

函数原型

OpenHiva::Subscriber::Subscriber()

参数说明

无

返回值

无

使用说明

不需要显示创建，只需通过OpenHiva::Node调用CreateSubscriber创建。

注意事项

可重入，同步接口。

4.8.2 OpenHiva::Subscriber::Ready

函数功能

判断Subscriber是否处于工作状态。

函数原型

bool OpenHiva::Subscriber::Ready()

参数说明

无

返回值

返回bool类型，若为true，表示订阅者状态正常；否则异常。

使用说明

判断Subscriber是否有效，若无效则不再订阅消息。

注意事项

- 可重入，同步接口。
- 当Subscriber未被初始化或Subscriber处于退出状态时，该接口返回false。

4.8.3 OpenHiva::Subscriber::StartReadMessage

函数功能

启动接收消息，用户调用OpenHiva::Node::CreateSubscriber接口成功后，默认是启动状态。

函数原型

`uint32_t OpenHiva::Subscriber::StartReadMessage()`

参数说明

无

返回值

返回uint32_t类型，成功返回HIVA_SUCCESS，该值为0；其他值为失败。

使用说明

当接口停止调用后，可以调用StartReadMessage接口恢复启动状态。

注意事项

- 可重入，同步接口。
- 当Subscriber初始化失败时，StartReadMessage接口必然返回失败。

4.8.4 OpenHiva::Subscriber::StopReadMessage

函数功能

停止接收消息。停止后不再调用用户回调，直接将消息丢弃。

函数原型

`uint32_t OpenHiva::Subscriber::StopReadMessage()`

参数说明

无

返回值

返回uint32_t类型，成功返回HIVA_SUCCESS，该值为0；其他值为失败。

使用说明

- 在正常订阅消息时，可通过StopReadMessage接口来停止，停止后不再调用用户回调函数。
- 调用StopReadMessage后，可通过[4.8.3 OpenHiva::Subscriber::StartReadMessage](#)恢复。

注意事项

- 可重入，同步接口。
- 调用CreateSubscriber接口失败时，StopReadMessage接口必然返回失败。

4.9 OpenHiva::ServiceServer 接口

4.9.1 OpenHiva::ServiceServer::Enable

函数功能

使能服务端。

函数原型

`uint32_t OpenHiva::ServiceServer::Enable()`

参数说明

无

返回值

返回uint32_t类型，成功返回HIVA_SUCCESS，该值为0；其他值为失败。

使用说明

只有调用Enable成功后，服务端才能正常工作。

注意事项

- 当服务未使能或者使能失败时，客户端的[4.10.1 OpenHiva::ServiceClient::InvokeService](#)接口调用会超时失败。
- 不可多次调用。

4.9.2 OpenHiva::ServiceServer::Disable

函数功能

去使能服务端。

函数原型

```
uint32_t OpenHiva::ServiceServer::Disable()
```

参数说明

无

返回值

返回uint32_t类型，成功返回HIVA_SUCCESS，该值为0；其他值为失败。

使用说明

调用该接口后，服务端停止接收客户端请求。

注意事项

无

4.9.3 OpenHiva::ServiceServer::Destroy

函数功能

释放[4.5.4 OpenHiva::Node::CreateServer](#)申请的资源。

函数原型

```
void OpenHiva::ServiceServer::Destroy()
```

参数说明

无

返回值

返回uint32_t类型，成功返回HIVA_SUCCESS，该值为0；其他值为失败。

使用说明

调用该接口后，服务端所有资源将被释放。

注意事项

无

4.10 OpenHiva::ServiceClient 接口

4.10.1 OpenHiva::ServiceClient::InvokeService

函数功能

客户端向服务端发送请求。

函数原型

```
template <typename ReqType, typename ResType> uint32_t
OpenHiva::ServiceClient::InvokeService(ReqType req, ResType &res, const
uint32_t timeout = 5000U)
```

参数说明

参数名	输入/输出	说明
req	输入	请求消息类型，必须是HivaMessage的子类。
timeout	输入	时间限制，用户可自行设置，默认是5000ms。
res	输出	用户从服务端获取的响应消息类型，必须是HivaMessage的子类。

返回值

返回uint32_t类型，成功返回HIVA_SUCCESS，该值为0；其他值为失败。

使用说明

只有调用OpenHiva::Node::CreateClient返回的ServiceClient才能正常工作。

注意事项

要与调用[4.5.5 OpenHiva::Node::CreateClient](#)注册的请求和响应的消息类型相匹配。

4.10.2 OpenHiva::ServiceClient::Destroy

函数功能

释放[4.5.5 OpenHiva::Node::CreateClient](#)申请的资源。

函数原型

```
void OpenHiva::ServiceClient::Destroy()
```

参数说明

无

返回值

返回uint32_t类型，成功返回HIVA_SUCCESS，该值为0；其他值为失败。

使用说明

调用该接口后，客户端所有资源将被释放，不再工作。

注意事项

无

4.11 NnEngine 接口

4.11.1 Hiva::CreateNnEngine

函数功能

创建NN引擎，加载指定的模型文件，对inTopicInfo发布的数据执行推理，并将推理的结果以outTopicInfo的形式发布出来。

函数原型

```
uint32_t Hiva::CreateNnEngine(const std::string& engineName, const EngineAttr& engineAttr, const NnTopicInfo& inTopicInfo, const NnTopicInfo& outTopicInfo)
```

参数说明

参数名	输入/输出	说明
engineName	输入	指推理引擎的名字，必须唯一、不能为空，不能超过128字节，不能包含 '\、*'。

参数名	输入/输出	说明
engineAttr	输入	指推理引擎的属性，包括推理模型的路径、所在分组、模型优先级等。 typedef struct EngineAttr_ { // 定义了当前推理模型属性 std::string modelPath; // 模型在文件系统的路径 uint32_t groupId; // 模型所在分组，示例参见 模型分组 ，默认范围0-1(范围可改，最大范围0-3) uint32_t pri; // 模型的优先级，数字越小优先级越高，取值为[0-7] };
inTopicInfo	输入	模型输入数据的Topic名称（vector结构）及queue size，所有输入数据拥有相同的队列长度，开启队列流控和写覆盖。topicVec中的topic name首字母必须为'/'。
outTopicInfo	输出	模型输出数据的topic名称（vector结构）及queue size，所有输出数据拥有相同的队列长度，关闭队列流控和写覆盖。topicVec中的topic name首字母必须为'/'。

返回值

返回uint32_t类型，成功返回HIVA_SUCCESS，该值为0；其他值为失败。

使用说明

通过调用此接口，APP可以将inTopicInfo发布的数据传递给NN进行异步推理，并将推理的结果以outTopicInfo的形式发布出来。

- 在调用此接口之前，APP需要先进行inTopicInfo注册，避免在调用此接口之前执行inTopicInfo的Publish操作。
- 在调用此接口之后，APP需要对outTopicInfo执行Subscribe操作。

```
typedef struct NnTopicInfo_{  
    const std::vector<std::string> topicVec;  
    const uint32_t queueSize; // queueSize的范围1-127  
}NnTopicInfo;
```

注意事项

- 本接口不可重入，同步接口。
- 对于同一个inTopicInfo的数据，可通过此接口创建多个不同的NNEngine执行推理，但需要确保使用不同的模型和outTopicInfo。
- inTopicInfo发布的数据需要采用buffer管理的接口申请共享内存，并采用Publish接口发布。
- outTopicInfo需要使用Subscriber接口订阅。
- NN推理完成后，会对inTopicInfo发布的数据执行释放操作；同一个数据多个推理时，会在最后一个推理完成后释放。
- NN推理完成后，会调用订阅outTopicInfo的回调，回调中传入的是Tensor类型的共享内存指针，此共享内存会在回调完成后由框架释放。

- 本接口非NodeHandle成员函数。

模型分组

模型分组是通过配置文件（yaml文件）来设置各组的属性，配置文件示例如下。如需修改配置文件，需通过CfgMgr接口操作（可参见[4.13 配置管理相关接口](#)）。

```
AlGroups:
  group0:
    aicore_number: 8
    aivector_number: 4
    sdma_number: 255    # 代表使用默认值
    aicpu_number: 255   # 代表使用默认值
    active_sq_number: 1
  group1:
    aicore_number: 2
    aivector_number: 4
    sdma_number: 255    # 代表使用默认值
    aicpu_number: 255   # 代表使用默认值
    active_sq_number: 1
  group2:
    aicore_number: 0    # 该值为0，则其他值无作用
    aivector_number: 0
    sdma_number: 255    # 代表使用默认值
    aicpu_number: 255   # 代表使用默认值
    active_sq_number: 1
  group3:
    aicore_number: 0    # 该值为0，则其他值无作用
    aivector_number: 0
    sdma_number: 255    # 代表使用默认值
    aicpu_number: 255   # 代表使用默认值
    active_sq_number: 1
```

4.11.2 Hiva::DestroyNnEngine

函数功能

销毁NN引擎。

函数原型

`uint32_t Hiva::DestroyNnEngine(const std::string& engineName)`

参数说明

参数名	输入/输出	说明
engineName	输入	待销毁推理引擎的名字，需要与CreateNnEngine里名称保持一致。

返回值

返回uint32_t类型，成功返回HIVA_SUCCESS，该值为0；其他值为失败。

使用说明

- 当不再需要使用NN Engine，可通过此接口释放。

- 调用[4.11.1 Hiva::CreateNnEngine](#)接口后，进程退出时一定要调用DestroyNnEngine接口来主动释放资源，不然可能会coredump。

注意事项

- 本接口不可重入，同步接口。
- 本接口非NodeHandle成员函数。

4.12 定时器接口

4.12.1 Hiva::CreateTimer

函数功能

创建Timer定时器。

函数原型

Hiva::Timer Hiva::CreateTimer(const Hiva::HivaDuration &timerPeriod, const TimerCallback &timerCb, const std::string &groupName, const bool oneshot = false, const bool autostart = true)

参数说明

参数名	输入/输出	说明
timerPeriod	输入	触发回调函数的周期，为框架标准的Duration类型。
timerCb	输入	定时器到期后的回调函数，类型为TimerCallback，其中TimerEvent定义如下： using TimerCallback = std::function<void (const TimerEvent &)>; struct TimerEvent { Time last_expected; // 上次回调应该发生的时刻 Time last_real; // 上次回调实际发生的时刻 Time last_expired; // 上次定时器在timer Server处过期，并向APP侧发起通知的时刻 Time current_expected; // 本次回调应该发生的时刻 Time current_real; // 本次回调实际发生的时刻 Time current_expired; // 本次定时器在timer Server处过期，并向APP侧发起通知的时刻 struct{ Hiva::WallDuration last_duration; // 上次callback执行耗时 } profile; };
groupName	输入	调用OpenHiva::Init时创建的线程组。如果为确定性组，则使用确定性调度框架；否则使用操作系统的调度框架。

参数名	输入/输出	说明
oneshot	输入	定时器的模式标识。 • true代表采用一次性的oneshot模式，定时器只会被触发一次。 • false代表采用周期性的periodic模式，定时器会被周期性触发，直至进程资源被清理。
autostart	输入	定时器的启动标识。 • true代表自动启动，接口返回时定时器已经开启。 • false代表不自动启动，接口返回时定时器未开启，需通过4.12.2 Hiva::Timer::Start手动开启。

返回值

返回Hiva::Timer的handle，后续可基于该handle进行start、stop等操作。

使用说明

进程调用该接口，即创建并开始运行一个定时器。

- APP创建Timer过程中会订阅一个定时器的事件，定时器名称由APP名称+定时器+序号号生成，以保证唯一性。
- APP通过Server/Client的Topic通信方式将定时器name和相关信息发送给定时器Server处理，等时间到期后，定时器Server发布该话题，通知到原APP，执行用户的回调函数。
- 定时器Topic有单独的分组，不会与业务线程分组产生冲突，可以并发。

注意事项

- 该接口优先使用Hiva clock时钟源，如果没有Hiva clock，则使用系统的wall clock。
- 针对确定性定时器，入参的group将会被用于订阅定时器事件，之前该组订阅的事件将被取消掉。
- 该接口创建的定时器，将使用操作系统自带的定时唤醒机制，不会使用基于硬件定时器的高精度用户态定时器。如果不使用Hiva clock，且需要高精度时钟，请调用[4.12.6 Hiva::CreateWallTimer](#)和[4.12.11 Hiva::CreateSteadyTimer](#)接口。
- 建议在启动定时器时，立即确定是否使用simulation time（Hiva clock时钟源），避免中途切换到wall clock。因为时钟源的切换不会主动通知定时器，因此有可能时钟源已经切换，但定时器还在使用切换前的时钟源计时。
- 如果中途切换了simulation time，则应该重启节点才能正常按照新的时钟源工作。

4.12.2 Hiva::Timer::Start

函数功能

停止后再次启动Timer定时器。

函数原型

void Hiva::Timer::Start()

参数说明

无

返回值

无

使用说明

调用CreateTimer创建定时器，如果中途调用了[4.12.5 Hiva::Timer::Stop](#)，可调用该接口再次启动。

注意事项

该接口只用于创建的定时器被Stop后，再次运行时使用。

4.12.3 Hiva::Timer::IsValid

函数功能

判断Timer定时器是否有效。

函数原型

bool Hiva::Timer::IsValid()

参数说明

无

返回值

返回bool类型，true代表为有效定时器，false代表定时器的周期不合规（周期为0）。

使用说明

无

注意事项

无

4.12.4 Hiva::Timer::SetPeriod

函数功能

设置Timer定时器周期。

函数原型

```
void Hiva::Timer::SetPeriod(const Duration& period, bool reset=true)
```

参数说明

参数名	输入/输出	说明
period	输入	新设定的周期。
reset	输入	• true代表下次触发时机，将以当前时间加上period间隔为准。 • false代表下次触发时机保持不变，再往后的触发以period为间隔。 当前只支持true模式，为接口兼容，保留该参数。

返回值

无

使用说明

无

注意事项

无

4.12.5 Hiva::Timer::Stop

函数功能

停止Timer定时器。

函数原型

```
void Hiva::Timer::Stop()
```

参数说明

无

返回值

无

使用说明

CreateTimer创建定时器后，可以调用本接口停止该定时器。

注意事项

- 对于oneshot模式，如果时间未到，则直接停止该定时器。
- 对于oneshot模式，如果已经执行过回调函数，再调用该接口对定时器无任何影响。
- 当正在执行回调函数时，回调函数会被继续执行完毕。

4.12.6 Hiva::CreateWallTimer

函数功能

创建WallTimer定时器。

函数原型

```
Hiva::WallTimer Hiva::CreateWallTimer(const Hiva::WallDuration
&wallPeriod, const WallTimerCallback &wallCallback, const std::string
&wallGroupName, const bool oneshot = false, const bool autostart = true)
```

参数说明

参数名	输入/输出	说明
wallPeriod	输入	触发回调函数的周期，为框架标准的Duration类型。
wallCallback	输入	定时器到期后的回调函数，WallTimerCallback类型，其中WallTimerEvent为： using WallTimerCallback = std::function<void (const WallTimerEvent &)>; struct WallTimerEvent { Hiva::WallTime last_expected; // 上次回调应该发生的时刻 Hiva::WallTime last_real; // 上次回调实际发生的时刻 Hiva::WallTime last_expired; // 上次定时器在timer Server处过期，并向APP侧发起通知的时刻 Hiva::WallTime current_expected; // 本次回调应该发生的时刻 Hiva::WallTime current_real; // 本次回调实际发生的时刻 Hiva::WallTime current_expired; // 本次定时器在timer Server处过期，并向APP侧发起通知的时刻 struct{ Hiva::WallDuration last_duration; // 上次callback执行耗时 } profile; };
wallGroupName	输入	调用OpenHiva::Init时创建的线程组。如果为确定性组，则使用确定性调度框架；否则使用操作系统的调度框架。
oneshot	输入	定时器的模式标识。 • true代表采用一次性的oneshot模式，定时器只会被触发一次。 • false代表采用周期性的periodic模式，定时器会被周期性触发，直至进程资源被清理。

参数名	输入/输出	说明
autostart	输入	定时器的启动标识。 • true代表自动启动，接口返回时定时器已经开启。 • false代表不自动启动，接口返回时定时器未开启，需通过4.12.7 Hiva::WallTimer::Start手动开启。

返回值

返回Hiva::WallTimer的handle，后续可基于该handle进行start、stop等操作。

使用说明

进程调用该接口，即创建并开始运行一个定时器。

- APP创建WallTimer过程中会订阅一个定时器的事件，定时器名称由APP名称+定时器+序列号生成，以保证唯一性。
- APP通过Server/Client的Topic通信方式将定时器name和相关信息发送给定时器Server处理，等时间到期后，定时器Server发布该话题，通知到原APP，执行用户的回调函数。

注意事项

无

4.12.7 Hiva::WallTimer::Start

函数功能

停止后再次启动WallTimer定时器。

函数原型

void Hiva::WallTimer::Start()

参数说明

无

返回值

无

使用说明

调用CreateWallTimer创建定时器，如果中途调用了4.12.10 Hiva::WallTimer::Stop，可调用该接口再次启动。

注意事项

该接口只用于创建的定时器被Stop后，再次运行时使用。

4.12.8 Hiva::WallTimer::IsValid

函数功能

判断WallTimer定时器是否有效。

函数原型

bool Hiva::WallTimer::IsValid()

参数说明

无

返回值

返回bool类型，true代表为有效定时器，false代表定时器的周期不合规（周期为0）。

使用说明

无

注意事项

无

4.12.9 Hiva::WallTimer::SetPeriod

函数功能

设置WallTimer定时器周期。

函数原型

void Hiva::WallTimer::SetPeriod(const Hiva::WallDuration& period, bool reset=true)

参数说明

参数名	输入/输出	说明
period	输入	新设定的周期。

参数名	输入/输出	说明
reset	输入	• true代表下次触发时机，将以当前时间加上period间隔为准。 • false代表下次触发时机保持不变，再往后的触发以period为间隔。 当前只支持true模式，为接口兼容，保留该参数。

返回值

无

使用说明

无

注意事项

无

4.12.10 Hiva::WallTimer::Stop

函数功能

停止WallTimer定时器。

函数原型

void Hiva::WallTimer::Stop()

参数说明

无

返回值

无

使用说明

CreateWallTimer创建定时器后，可以调用本接口停止该定时器。

注意事项

- 对于oneshot模式，如果时间未到，则直接停止该定时器。
- 对于oneshot模式，如果已经执行过回调函数，再调用该接口对定时器无任何影响。
- 当正在执行回调函数时，回调函数会被继续执行完毕。

4.12.11 Hiva::CreateSteadyTimer

函数功能

创建SteadyTimer定时器。

函数原型

Hiva::SteadyTimer Hiva::CreateSteadyTimer(const Hiva::WallDuration &steadyPeriod, const SteadyTimerCallback &steadyCallback, const std::string &steadyGroupName, const bool oneshot = false, const bool autostart = true)

参数说明

参数名	输入/输出	说明
steadyPeriod	输入	触发回调函数的周期，为框架标准的Duration类型。
steadyCallba ck	输入	定时器到期后的回调函数，参数类型为 SteadyTimerCallback，其中WallTimerEvent为： using SteadyTimerCallback = std::function<void (const SteadyTimerEvent &)>; struct SteadyTimerEvent { Hiva::SteadyTime last_expected; // 上次回调应该发生的时刻 Hiva::SteadyTime last_real; // 上次回调实际发生的时刻 Hiva::SteadyTime last_expired; // 上次定时器在timer Server处过期，并向APP侧发起通知的时刻 Hiva::SteadyTime current_expected; // 本次回调应该发生的时刻 Hiva::SteadyTime current_real; // 本次回调实际发生的时刻 Hiva::SteadyTime current_expired; // 本次定时器在timer Server处过期，并向APP侧发起通知的时刻 struct{ Hiva::WallDuration last_duration; // 上次callback执行耗时 } profile; };
steadyGroup Name	输入	调用OpenHiva::Init时创建的线程组。如果为确定性组，则使用确定性调度框架；否则使用操作系统的调度框架。
oneshot	输入	定时器的模式标识。 • true代表采用一次性的oneshot模式，定时器只会被触发一次。 • false代表采用周期性的periodic模式，定时器会被周期性触发，直至进程资源被清理。
autostart	输入	定时器的启动标识。 • true代表自动启动，接口返回时定时器已经开启。 • false代表不自动启动，接口返回时定时器未开启，需通过4.12.12 Hiva::SteadyTimer::Start手动开启。

返回值

返回Hiva::SteadyTimer的handle，后续可基于该handle进行start、stop等操作。

使用说明

进程调用该接口，即创建并开始运行一个定时器。

- APP创建SteadyTimer过程中会订阅一个定时器的事件，定时器名称由APP名称+定时器+序列号生成，以保证唯一性。
- APP通过Server/Client的Topic通信方式将定时器name和相关信息发送给定时器Server处理，等时间到期后，定时器Server发布该话题，通知到原APP，执行用户的回调函数。

注意事项

无

4.12.12 Hiva::SteadyTimer::Start

函数功能

停止后再次启动SteadyTimer定时器。

函数原型

```
void Hiva::SteadyTimer::Start()
```

参数说明

无

返回值

无

使用说明

调用CreateSteadyTimer创建了定时器，如果中途调用了[4.12.15 Hiva::SteadyTimer::Stop](#)，可调用该接口再次启动。

注意事项

该接口只用于创建的定时器被Stop后，再次运行时使用。

4.12.13 Hiva::SteadyTimer::IsValid

函数功能

判断SteadyTimer定时器是否有效。

函数原型

bool Hiva::SteadyTimer::IsValid()

参数说明

无

返回值

返回bool类型，true代表为有效定时器，false代表定时器的周期不合规（周期为0）。

使用说明

无

注意事项

无

4.12.14 Hiva::SteadyTimer::SetPeriod

函数功能

设置SteadyTimer定时器周期。

函数原型

void Hiva::SteadyTimer::SetPeriod(const Hiva::WallDuration& period, bool reset=true)

参数说明

参数名	输入/输出	说明
period	输入	新设定的周期。
reset	输入	• true代表下次触发时机，将以当前时间加上period间隔为准。 • false代表下次触发时机保持不变，再往后的触发以period为间隔。 当前只支持true模式，为接口兼容，保留该参数。

返回值

无

使用说明

无

注意事项

无

4.12.15 Hiva::SteadyTimer::Stop

函数功能

停止SteadyTimer定时器。

函数原型

`void Hiva::SteadyTimer::Stop()`

参数说明

无

返回值

无

使用说明

CreateSteadyTimer创建定时器后，可以调用本接口停止该定时器。

注意事项

- 对于oneshot模式，如果时间未到，则直接停止该定时器。
- 对于oneshot模式，如果已经执行过回调函数，再调用该接口对定时器无任何影响。
- 当正在执行回调函数时，回调函数会被继续执行完毕。

4.13 配置管理相关接口

4.13.1 约束与限制

配置管理模块（ CfgMgr ）主要用于管理系统中的配置参数，统一以yaml格式的保存，再配合参数的设置、获取、快照、重载等接口，提高用户管理参数的效率。CfgMgr接口在使用时，需注意以下事项：

- 总参数数量不能超过50000个；
- 单个key应小于128字节；
- key需保证唯一，即便属于不同的参数组，key也需要唯一；加载或设置相同key的配置参数时会进行覆盖；
- 单个value应小于256字节；
- key、value不能包含 ':'、';' 等字符；
- 参数组名称（ group ）应小于64字节；

- 使用set接口时，如果value是个map，相当set多个参数，一次性set的参数不超过128个；
- 快照名称不超过32字节；
- 取值范围表达式不超过256字节；
- 类型表达式不超过32字节；
- 版本信息不超过32字节；

4.13.2 CfgMgr::BatchReset

函数功能

批量删除持久化数据。

函数原型

- 按yaml文件中指定的key批量删除持久化数据
`uint32_t CfgMgr::BatchReset(const std::string &yamlPath, const std::string &yamlNamespace = "", const bool isSync = false)`
- 根据WNP属性批量删除持久化数据
`uint32_t CfgMgr::BatchReset(const uint32_t paramType = WNP, const bool isSync = false)`

功能安全等级

ASIL-D

参数说明

- 按yaml文件中指定的key批量删除持久化数据

参数名	输入/输出	说明
yamlPath	输入	yaml文件路径，必须是全路径。
yamlNamespace	输入	yaml文件namespace，对加载yaml文件的namespace对应，默认为空。
isSync	输入	是否同步写持久化文件，默认异步。

- 根据WNP属性批量删除持久化数据

参数名	输入/输出	说明
paramType	输入	需要批量复位的参数类型，当前仅支持WNP，其他会返回错误。
isSync	输入	是否同步写持久化文件，默认异步

返回值

返回uint32_t类型，成功返回CFGMGR_SUCCESS，失败返回对应的错误码。

使用说明

开启selinux权限控制后，非特权进程需要配置对dp:/res/cfgmgr/mgmt的读写权限才能调用。

注意事项

按yaml文件中指定的key批量删除持久化数据时：

- 当基础yaml文件和yamlPath指定的yaml文件中存在相同key值，则采用默认的基础yaml文件中的值（不再持久化，会修改内存中的值）；否则删除该key值。
- 当本次reset后所有持久化数据都被删除了，相当于删除持久化文件。重启时如果没有持久化数据，会尝试按版本继承持久化数据，继承失败会有相应告警。
- 如果yamlPath指定的yaml文件中的key值在环境中不存在，也会返回成功。

4.13.3 CfgMgr::CancelMonitor

函数功能

取消检查。

函数原型

```
uint32_t CfgMgr::CancelMonitor(const std::string &kParam = "")
```

功能安全等级

ASIL-D

参数说明

参数名	输入/输出	说明
kParam	输入	key值，例如use_sim_time，默认为空，表示取消所有参数的检查。

返回值

返回uint32_t类型，成功返回CFGMGR_SUCCESS，失败返回对应的错误码。

使用说明

无

注意事项

- 取消检查时如果正在执行Monitor注册的回调函数，则会等待本次回调处理后再注销。
- 如果对该kParam调用了多次检查，取消时会全部注销。

- 调用4.13.14 CfgMgr::Monitor接口后，在进程退出时，一定要调用CancelMonitor接口释放线程，否则可能coredump。如果调用了OpenHiva::Shutdown，可不调用CancelMonitor接口，因为OpenHiva::Shutdown里默认会调用CancelMonitor接口。

4.13.4 CfgMgr::CleanStartConfig

函数功能

清理指定启动描述文件（版本号）的数据。

函数原型

uint32_t CfgMgr::CleanStartConfig(const std::string &startConfig)

功能安全等级

ASIL-D

参数说明

参数名	输入/输出	说明
startConfig	输入	启动描述文件（版本号）。

返回值

返回uint32_t类型，成功返回CFGMGR_SUCCESS，失败返回对应的错误码。

使用说明

开启selinux权限控制后，非特权进程需要配置对dp:/res/cfgmgr/mgmt的读写权限才能调用。

注意事项

不能清理当前正在使用的版本（清理时返回false）。

4.13.5 CfgMgr::CreateSnapshot

函数功能

配置参数快照，记录当前所有持久化参数，以便还原。

函数原型

uint32_t CfgMgr::CreateSnapshot(const std::string &snapshotName, const std::string &snapshotNamespace = "/", const bool isSync = true)

功能安全等级

ASIL-D

参数说明

参数名	输入/输出	说明
snapshotName	输入	快照文件名称，还原时需按这个名称还原。
snapshotNamespace	输入	快照的名称域，默认快照所有。
isSync	输入	是否同步写快照文件，默认同步。

返回值

返回uint32_t类型，成功返回CFGMGR_SUCCESS，失败返回对应的错误码。

使用说明

- 按同一个名称重复快照，会进行覆盖。
- 快照时只会将持久化的参数保存下来。
- 开启selinux权限控制后，非特权进程需要配置对dp:/res/cfgmgr/mgmt的读写权限才能调用。

注意事项

- 快照名取值不超过32字节，不可以为空，只允许字母、数字、下划线，否则返回失败。
- 快照名不能以 "_backup" 为后缀，避免与内部备份文件命名冲突，否则返回失败。
- 创建快照文件失败时，需自行判断是否需要上报故障或其他处理。

4.13.6 CfgMgr::CreateSnapshotToFile

函数功能

指定快照路径进行快照。

函数原型

uint32_t CfgMgr::CreateSnapshotToFile(const std::string &filePath, const std::string &snapshotNamespace = "/", const bool isSync = true)

功能安全等级

ASIL-D

参数说明

参数名	输入/输出	说明
filePath	输入	指定的快照路径，必须是可访问的路径，且需要配置在初始化白名单中。
snapshotNamespace	输入	需要快照的参数所属的命名空间。
isSync	输入	是否同步写文件，默认同步。

返回值

返回uint32_t类型，成功返回CFGMGR_SUCCESS，失败返回对应的错误码。

使用说明

开启selinux权限控制后，非特权进程需要配置对dp:/res/cfgmgr/mgmt的读写权限才能调用。

注意事项

- 指定的路径必须合法且配置在启动文件中。
- 快照时，如果快照内容无任何参数，会返回失败。

4.13.7 CfgMgr::DeleteSnapshot

函数功能

删除快照文件。

函数原型

```
uint32_t CfgMgr::DeleteSnapshot(const std::string &snapshotName, const std::string &startConfig = "")
```

功能安全等级

ASIL-D

参数说明

参数名	输入/输出	说明
snapshotName	输入	快照文件名称，按名称进行删除。
startConfig	输入	启动描述文件（版本号），默认当前正在使用的启动描述文件。

返回值

返回uint32_t类型，成功返回CFGMGR_SUCCESS，失败返回对应的错误码。

使用说明

开启selinux权限控制后，非特权进程需要配置对dp:/res/cfgmgr/mgmt的读写权限才能调用。

注意事项

需按创建快照[4.13.5 CfgMgr::CreateSnapshot](#)时的名称进行删除。

4.13.8 CfgMgr::Get

函数功能

读取配置参数，可按参数名或namespace来读取。

函数原型

```
uint32_t CfgMgr::Get(const std::string &kParam, T &value, const std::string &srv = "")
```

功能安全等级

ASIL-D

参数说明

参数名	输入/输出	说明
kParam	输入	key值，为参数名称或参数名称域（ namespace ），例如/ros/use_sim_time、/ros，key值开头加不加“/”都可以。
srv	输入	参数组，和资源文件对应，不填则默认到有权限的参数组中获取参数，效率会降低。
value	输出	用于接收参数的值。由用户指定参数类型，支持的类型参见本表“使用说明”。

返回值

返回uint32_t类型，表示本次获取参数成功与否。成功返回CFGMGR_SUCCESS，失败返回对应的错误码。

使用说明

- 非模板函数，针对各种类型进行重载，value值支持的类型有：std::string、double、float、int、bool、std::vector<std::string>、std::vector<double>、std::vector<float>、std::vector<int>、std::vector<bool>、std::map<std::string,

std::string>、std::map<std::string, double>、std::map<std::string, float>、std::map<std::string, int>、std::map<std::string, bool>。

- 按参数名称域来获取参数时，value值类型必须为map类型。如果该名称域下的参数类型不一致，value值需为std::map<std::string, std::string>类型，表示键值对。
- 参数组指yaml_path中填的group，需和虚拟资源文件对应，用于权限管理。

注意事项

- key值需小于128个字节、单个value值需小于256个字节。
- 获取参数后，建议根据业务逻辑判断参数有效性。
- 对于map类型参数，Get时不会清空入参value中的原有数据，只会修改或新增。
- 参数获取失败或判断无效时，需自行判断是否需要上报故障或其他处理。
- 接口返回失败，可能的原因有：没权限、不存在该参数、加载参数时未加载成功、数据传输失败（管理面）。

4.13.9 CfgMgr::GetAllStartConfig

函数功能

获取当前所有的启动描述文件（版本号）。

函数原型

uint32_t CfgMgr::GetAllStartConfig(std::vector<std::string> &allStartConfig)

功能安全等级

ASIL-D

参数说明

参数名	输入/输出	说明
allStartConfig	输出	所有启动描述文件（版本号）。

返回值

返回uint32_t类型，成功返回CFGMGR_SUCCESS，失败返回对应的错误码。

使用说明

开启selinux权限控制后，非特权进程需要配置对dp:/res/cfgmgr/mgmt的读写权限才能调用。

注意事项

无

4.13.10 CfgMgr::GetAllSnapshot

函数功能

获取当前所有的快照文件名称及对应启动描述文件（版本号）。

函数原型

```
uint32_t CfgMgr::GetAllSnapshot(std::vector<std::pair<std::string, string>>
&allSnapshot)
```

功能安全等级

ASIL-D

参数说明

参数名	输入/输出	说明
allSnapshot	输出	所有的快照文件名称及对应启动描述文件（版本号）。

返回值

返回uint32_t类型，成功返回CFGMGR_SUCCESS，失败返回对应的错误码。

使用说明

开启selinux权限控制后，非特权进程需要配置对dp:/res/cfgmgr/mgmt的读写权限才能调用。

注意事项

无

4.13.11 CfgMgr::GetStartConfig

函数功能

获取当前正在使用的启动描述文件（版本号）。

函数原型

```
uint32_t CfgMgr::GetStartConfig(std::string &startConfig)
```

功能安全等级

ASIL-D

参数说明

参数名	输入/输出	说明
startConfig	输入/输出	当前正在使用的启动描述文件（版本号）。

返回值

返回uint32_t类型，成功返回CFGMGR_SUCCESS，失败返回对应的错误码。

使用说明

开启selinux权限控制后，非特权进程需要配置对dp:/res/cfgmgr/mgmt的读写权限才能调用。

注意事项

无

4.13.12 CfgMgr::Has

函数功能

判断某参数是否存在。

函数原型

uint32_t CfgMgr::Has(const std::string &kParam)

功能安全等级

ASIL-D

参数说明

参数名	输入/输出	说明
kParam	输入	key值，为参数名称。

返回值

返回uint32_t类型，存在返回CFGMGR_SUCCESS，不存在返回对应的错误码。

使用说明

无

注意事项

key值需小于128个字节。

4.13.13 CfgMgr::Load

函数功能

按yaml文件加载参数并进行持久化。

函数原型

```
uint32_t CfgMgr::Load(const std::string &yamlPath, const std::string
&yamlNamespace = "", const bool isSync = false, const std::string &group =
"")
```

功能安全等级

ASIL-D

参数说明

参数名	输入/输出	说明
yamlPath	输入	yaml文件路径，必须是全路径。
yamlNamesp ace	输入	yaml文件namespace，namespace+yaml文件中的key组成完整的key。
isSync	输入	是否同步写持久化文件，默认异步。
group	输入	表示加到哪个参数组（影响后续Get/Set时的访问权限）。仅对新增的参数有效，已有参数保持原有的group不变，默认为“”，表示global组。

返回值

返回uint32_t类型，成功返回CFGMGR_SUCCESS，失败返回对应的错误码。

使用说明

开启selinux权限控制后，非特权进程需要配置对dp:/res/cfgmgr/mgmt的读写权限才能调用。

注意事项

- 加载的参数会进行持久化。
- yaml文件是被CfgMgr进程读取，必须确保CfgMgr进程能正常读取。

4.13.14 CfgMgr::Monitor

函数功能

检查指定参数，当该参数发生Set操作时会调用回调函数。

函数原型

```
uint32_t CfgMgr::Monitor(const std::string &kParam, std::function<void(T)>
monitorFunc, const bool changeNeeded = false, const std::string &srv = "")
```

功能安全等级

ASIL-D

参数说明

参数名	输入/输出	说明
kParam	输入	key值，为参数名称或参数名称域（ namespace ），例如/ros/use_sim_time、/ros，key值开头加不加“/”都可以。
monitorFunc	输入	回调函数，函数入参支持的类型参见本表“使用说明”。
changeNeed ed	输入	• false：默认值，表示只要有Set这个参数就执行该回调函数，无论Set的值和之前比是否发生变化。 • true：表示只有当Set的值和之前比发生了变化才执行该回调函数。
srv	输入	参数组，和资源文件对应，不填则默认到有权限的参数组中获取参数，效率会降低。

返回值

返回uint32_t类型，成功返回CFGMGR_SUCCESS，失败返回对应的错误码。

使用说明

非模板函数，针对各种类型进行重载，回调函数参数支持的类型有：
std::string、double、float、int、bool、std::vector<std::string>、
std::vector<double>、std::vector<float>、std::vector<int>、std::vector<bool>、
std::map<std::string, std::string>、std::map<std::string, double>、
std::map<std::string, float>、std::map<std::string, int>、std::map<std::string, bool>。

注意事项

- key值需小于128个字节、单个value值需小于256个字节。

- 检查多个参数时需多次调用该接口。
- 调用检查函数会启单独的线程，对于同一个组多个检查函数共用一个线程，不同组不同线程。
- 对于检查Set操作，只有Set的key和Monitor的key完全一致时才会被检查到。
- 按完整的key检查时，Load和Revert操作也会触发检查。
- 同一个进程对同一个key调用多次Monitor时，注册的多个回调函数都会生效，即当这个key有Set操作时，所有的回调会被依次调用。
- 如果调用了Monitor接口，在进程退出时，一定要调用4.13.3 CfgMgr::CancelMonitor接口释放线程，否则可能coredump。如果调用了OpenHiva::Shutdown，可不调用CancelMonitor接口，因为OpenHiva::Shutdown里默认会调用CancelMonitor接口。
- 检查参数失败时，需自行判断是否需要上报故障或其他处理。
- 接口返回失败，可能的原因有：没权限、不存在该参数、加载参数时未加载成功、数据传输失败（管理面）。

4.13.15 CfgMgr::Param

函数功能

获取参数，可按参数名或namespace来获取，当获取失败时使用默认参数。

函数原型

```
uint32_t CfgMgr::Param(const std::string &kParam, T &value, const T &defaultValue, const std::string &srv = "")
```

功能安全等级

ASIL-D

参数说明

参数名	输入/输出	说明
kParam	输入	key值，为参数名称或参数名称域（ namespace ），例如/ros/use_sim_time、/ros，key值开头加不加“/”都可以。
defaultValue	输入	默认值，如果获取失败则使用默认值。
srv	输入	参数组，和资源文件对应，不填则默认到有权限的参数组中获取参数，效率会降低。
value	输出	用于接收参数的值。由用户指定参数类型，支持的类型参见本表“使用说明”。

返回值

返回uint32_t类型，表示本次获取参数成功与否，成功返回CFGMGR_SUCCESS，失败返回对应的错误码。

使用说明

- value值支持的类型有：std::string、double、float、int、bool、std::vector<std::string>、std::vector<double>、std::vector<float>、std::vector<int>、std::vector<bool>、std::map<std::string, std::string>、std::map<std::string, double>、std::map<std::string, float>、std::map<std::string, int>、std::map<std::string, bool>。
- 按参数名称域来获取参数时，value值必须为map类型。如果该名称域下的参数类型不一致，value值需为std::map<std::string, std::string>类型，表示键值对。

注意事项

- key值需小于128个字节、单个value值需小于256个字节。
- 获取参数后，建议根据业务逻辑判断参数有效性。
- 对map类型的参数，Get时不会清空入参value中的原有数据，只会修改或新增。
- 参数获取失败或判断无效时，需自行判断是否需要上报故障或其他处理。
- 接口返回失败，可能的原因有：没权限、不存在该参数、加载参数时未加载成功、数据传输失败（管理面）。

4.13.16 CfgMgr::Reload

函数功能

重新加载yaml文件。当需要加载的yaml文件发生变更时可通过该接口重新加载yaml文件，重新生成初始配置文件。

函数原型

uint32_t CfgMgr::Reload(bool needClean = false)

功能安全等级

ASIL-D

参数说明

参数名	输入/输出	说明
needClean	输入	是否需要清理持久化数据，默认不清理。

返回值

返回uint32_t类型，成功返回CFGMGR_SUCCESS，失败返回对应的错误码。

使用说明

开启selinux权限控制后，非特权进程需要配置对dp:/res/cfgmgr/mgmt的读写权限才能调用。

注意事项

调用后立马生效。

4.13.17 CfgMgr::Reset

函数功能

根据参数删除持久化数据。

函数原型

- 按单个参数删除持久化数据
`uint32_t CfgMgr::Reset(const std::string &kParam,const bool isSync = false)`
- 按多个参数删除持久化数据
`uint32_t CfgMgr::Reset(const std::vector<std::string> &kParam,const bool isSync = false)`

功能安全等级

ASIL-D

参数说明

- 按单个参数删除持久化数据

参数名	输入/输出	说明
kParam	输入	参数key值，必须是完整的key。
isSync	输入	是否同步写持久化文件，默认异步。

- 按多个参数删除持久化数据

参数名	输入/输出	说明
kParam	输入	参数key值，必须是完整的key，可传入多个key。
isSync	输入	是否同步写持久化文件，默认异步。

返回值

返回uint32_t类型，成功返回CFGMGR_SUCCESS，失败返回对应的错误码。

使用说明

开启selinux权限控制后，非特权进程需要配置对dp:/res/cfgmgr/mgmt的读写权限才能调用。

注意事项

- 如果基础yaml文件存在该key值，则采用默认的基础yaml文件中的值（不再持久化，会修改内存中的值）；否则删除该key值。
- 如果本次reset后所有持久化数据都被删除了，相当于删除持久化文件。重启时如果没有持久化数据，会尝试按版本继承持久化数据，继承失败会有相应告警。
- 如果该key值在环境中不存在，也会返回成功。

4.13.18 CfgMgr::RevertSnapshot

函数功能

按名称还原快照文件。

函数原型

```
uint32_t CfgMgr::RevertSnapshot(const std::string &snapshotName = "", const bool isSync = true)
```

功能安全等级

ASIL-D

参数说明

参数名	输入/输出	说明
snapshotName	输入	快照文件名称，
isSync	输入	是否同步写持久化文件，默认同步。

返回值

返回uint32_t类型，成功返回CFGMGR_SUCCESS，失败返回对应的错误码。

使用说明

开启selinux权限控制后，非特权进程需要配置对dp:/res/cfgmgr/mgmt的读写权限才能调用。

注意事项

- 需按创建快照[4.13.5 CfgMgr::CreateSnapshot](#)时的名称进行还原。
- 快照还原是对持久化中的参数进行的，非持久化的参数不受还原的影响。
- 还原失败时，需自行判断是否需要上报故障或其他处理。
- 接口返回失败，可能的原因有：文件不存在或没权限。

4.13.19 CfgMgr::RevertSnapshotFromFile

函数功能

指定路径进行恢复快照。

函数原型

```
uint32_t CfgMgr::RevertSnapshotFromFile(const std::string &filePath, const
bool isSync = true)
```

功能安全等级

ASIL-D

参数说明

参数名	输入/输出	说明
filePath	输入	快照文件的路径。
isSync	输入	是否同步写持久化文件，默认同步。

返回值

返回uint32_t类型，成功返回CFGMGR_SUCCESS，失败返回对应的错误码。

使用说明

开启selinux权限控制后，非特权进程需要配置对dp:/res/cfgmgr/mgmt的读写权限才能调用。

注意事项

指定的路径必须合法且配置在启动文件中。

4.13.20 CfgMgr::Set

函数功能

设置参数。可按参数名设置单个参数，或按namespace来设置多个参数。

函数原型

```
uint32_t CfgMgr::Set(const std::string &kParam, const T &value, const bool
isSync = false, const bool isPersist = false, const std::string &srv = "")
```

功能安全等级

ASIL-D

参数说明

参数名	输入/输出	说明
kParam	输入	key值，为参数名称或参数名称域（ namespace ），例如/ros/use_sim_time、/ros，key值开头加不加“/”都可以。
value	输入	用于接收参数的值。由用户指定参数类型，支持的类型参见本表“使用说明”。
isSync	输入	是否需要同步调用（ 同步表示等待持久化文件落盘 ），默认异步（ false ）。
isPersist	输入	是否需要持久化，默认不进行持久化（ false ）。
srv	输入	参数组，和资源文件对应，对应yaml_path.yaml配置示例中的group。修改参数时可不填srv参数，需判断权限，新增参数时需要指定参数组，不填则在默认参数组中新增参数。

返回值

返回uint32_t类型，成功返回CFGMGR_SUCCESS，失败返回对应的错误码。

使用说明

- 当参数不存在时则新增参数，当参数已存在时则修改参数。
- 非模板函数，针对不同类型的value进行重载，value值支持的类型有：
std::string、double、float、int、bool、std::vector<std::string>、
std::vector<double>、std::vector<float>、std::vector<int>、std::vector<bool>、
std::map<std::string, std::string>、std::map<std::string, double>、
std::map<std::string, float>、std::map<std::string, int>、std::map<std::string, bool>、const char*（ 当成string处理 ）。
- 按参数名称域来设置参数时，value值必须为map类型，表示键值对。

注意事项

- key值需小于128个字节（ map类型时，入参kParam加first值才是实际的key，实际的key需小于128字节，需保证实际key下面的参数都属于同一个组 ）、单个value值需小于256个字节。
- 一次性Set的参数不能超过128个字节（ map类型时，map的size等于参数个数 ），超过可能会失败。
- 设置参数失败时，需自行判断是否需要上报故障或其他处理。
- 接口返回失败，可能的原因有：没权限、数据传输失败、key或value超过长度。

4.13.21 CfgMgr::SetNextStartConfig

函数功能

设置启动参考描述文件（ merge标签 ）。

函数原型

```
uint32_t CfgMgr::SetNextStartConfig(const std::string &nextStartConfig)
```

功能安全等级

ASIL-D

参数说明

参数名	输入/输出	说明
nextStartConfig	输入	启动参考描述文件（merge标签）。

返回值

返回uint32_t类型，成功返回CFGMGR_SUCCESS，失败返回对应的错误码。

使用说明

开启selinux权限控制后，非特权进程需要配置对dp:/res/cfgmgr/mgmt的读写权限才能调用。

注意事项

- 多次设置会覆盖。
- 一次性有效。
- 不可为空。

4.13.22 CfgMgr::SetUpgrade

函数功能

使能系统重启后重新加载yaml文件，而不是读取初始配置文件，主要用于安装升级场景。

函数原型

```
uint32_t CfgMgr::SetUpgrade()
```

功能安全等级

ASIL-D

参数说明

无

返回值

返回uint32_t类型，成功返回CFGMGR_SUCCESS，失败返回对应的错误码。

使用说明

开启selinux权限控制后，非特权进程需要配置对dp:/res/cfgmgr/mgmt的读写权限才能调用。

注意事项

- 调用该接口后，系统重启时生效。
- 保留持久化数据。