AX APP Demo 用户指南

文档版本：V1.4

发布日期：2024/01/24

目 录

权利声明

注意

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

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

前 言

适用产品

AX620E 系列产品（AX630C、AX620Q）

适读人群

- 软件开发工程师
- 技术支持工程师

符号与格式定义

符号/格式	说明
`xxx`	表示您可以执行的命令行。
斜体	表示变量。如，“*安装目录*/AX620E_SDK_Vx.x.x/build 目录”中的“安装目录”是一个变量，由您的实际环境决定。
☞ 说明/备注：	表示您在使用产品的过程中，我们向您说明的事项。
！ 注意：	表示您在使用产品的过程中，需要您特别注意的事项。

修订历史

文档版本	发布时间	修订说明
V1.0	2023/08/30	文档初版
V1.1	2023/10/07	修改格式
V1.2	2023/10/16	增加字库章节
V1.3	2023/11/22	增加 Pano 双摄拼接相关说明
V1.4	2024/01/24	更新编译的工程名

1 概述

1.1 概要说明

APP Demo 是本产品系列 SDK 使用演示应用程序集合，主要包括 FRTDemo。FRTDemo 是前端产品 Demo，实现了 IPC 和 PANO 等不同业务形态目录结构

APP Demo 源码随 AX SDK 发布，位于 app/demo 目录，目录结构如下：

```
component
├── 3rd
│   ├── inc
│   │   ├── appweb
│   │   └── freetype2
│   └── lib
├── audio
├── builder
├── capture
├── collector
├── config
├── config_parser
├── detector
├── encoder
├── header
├── ives
├── ivps
├── linkage
├── log
├── mp4enc
├── option_helper
├── osd
├── pool
├── resource
├── rtsp
├── sensor
├── stage
├── utils
├── webapp
└── webserver
demo
└── FRTDemo
    ├── bin
    │   ├── config
    │   │   ├── ipc
    │   │   │   └── sensor
    │   │   └── pano
    │   │       └── sensor
    │   └── run.sh
    └── src
        ├── ipc
        └── pano
```

1.2 编译说明

编译步骤

步骤1 执行 `cd app/demo` 命令

步骤2 执行 `make p=xxx clean` 命令。

步骤3 执行 `make p=xxx all install` 命令。

☞ **说明：**

p=xxx 表示项目名称，比如 p=AX630C_emmc_arm64_k419，可通过 `make plist` 命令查询 SDK 支持的项目列表。

2 公共类库

2.1 概要说明

本章节主要描述 APP Demo 源码中开放的各公共类或公共接口。

大致有以下几类：

- SDK 模块封装或管理类，比如 CBaseSensor，CIVPSGrpStage，CVideoEncoder 等。
- 配置文件解析类，比如 CPPLCfgParser，CSensorCfgParser 等。
- 辅助工具类，比如 CAppLogWarper，CIniWarper 等。

> 说明：
>
> SDK API 功能和参数说明不在本文赘述，若要详细了解 AX SDK API，请参阅 AX SDK 相应模块的 API 说明文档。

2.2 公共类库

2.2.1 CBaseSensor

负责 Sensor 初始化所需流程的创建。初始化所需的属性配置由继承其的不同类型 sensor 子类来实现，具体可参照类 COS04a10。

Init / DeInit

【功能说明】

Sensor 初始化/去初始化。通过加载配置文件配置属性，完成 sensor 开流断流所有相关接口所需的配置参数的设置。不同类型的 sensor 通过不同的实现类来实现不同属性的配置。

*双摄拼接需要配置 sensor **AX_SNS_ATTR_T** 的 **eMasterSlaveSel** 属性：*

master sensor：AX_SNS_SYNC_MASTER

slave sensor：AX_SNS_SYNC_SLAVE

【接口定义】

```
virtual AX_BOOL Init(AX_VOID);

virtual AX_BOOL DeInit(AX_VOID);
```

【接口参数】

无。

【接口返回】

成功返回 AX_TRUE，失败返回 AX_FALSE。

【接口说明】

通常需配置的有 Device/MIPI/Pipe/3A 相关属性。

Open / Close

【功能说明】

调用 SDK 接口实现 Sensor 出流/断流流程的搭建。

【接口定义】

```
virtual AX_BOOL Open();

virtual AX_BOOL Close()
```

【接口参数】

无。

【接口返回】

成功返回 AX_TRUE，失败返回 AX_FALSE。

【接口说明】

无。

IsSnsSync

【功能说明】

判断是否需要启动 3A 双摄同步。

【接口定义】

```
AX_BOOL IsSnsSync()
```

【接口参数】

无。

【接口返回】

需要启动 3A 双摄同步返回 AX_TRUE，不需要启动 3A 双摄同步返回 AX_FALSE。

【接口说明】

双摄拼接返回 AX_TRUE，非双摄拼接模式，返回 AX_FALSE。

SetMultiSnsSync

【功能说明】

设置双摄 3A 同步使能。

【接口定义】

```
AX_BOOL SetMultiSnsSync(AX_BOOL bSync)
```

【接口参数】

bSync：是否使能双摄 3A 同步

【接口返回】

成功返回 AX_TRUE，失败返回 AX_FALSE。

【接口说明】

双摄拼接模式下使用

2.2.2 CSensorMgr

Sensor 管理类，负责 CBaseSensor 封装类的实例化及管理。同时也负责所有针对 sensor 的启动/停止/切换属性等操作。

Init / DeInit

【功能说明】

根据 sensor 配置信息，创建以及销毁 CBaseSensor 类实例。

【接口定义】

```
virtual AX_BOOL Init()

virtual AX_BOOL DeInit()
```

【接口参数】

无。

【接口返回】

Init：成功返回 AX_TRUE，失败返回 AX_FALSE。

DeInit：成功返回 AX_TRUE，失败返回 AX_FALSE。

【接口说明】

无。

Start / Stop

【功能说明】

利用 CBaseSensor 实例，启动/停止 ISP 模块。

【接口定义】

```
AX_BOOL Start()

AX_BOOL Stop()
```

【接口参数】

无。

【接口返回】

Start：成功返回 AX_TRUE，失败返回 AX_FALSE。

Stop：成功返回 AX_TRUE，失败返回 AX_FALSE。

【接口说明】

启动内容包括 ISP 模块的启动，非 link 模式下的 YUV 获取线程的启动。

VideoFrameRelease

VideoFrameRelease

【功能说明】

非 link 模式下释放通过 ISP 模块 AX_VIN_GetYuvFrame 接口获取到的输出帧 Buffer。

【接口定义】

```
virtual AX_VOID VideoFrameRelease(CAXFrame* pFrame)
```

【接口参数】

pFrame：YUV 帧指针。

【接口返回】

无。

【接口说明】

通过 ISP 模块 AX_VIN_GetYuvFrame 接口获取到的帧需要在后级模块使用完以后才可进行释放，完全依赖于后级模块处理逻辑，此函数指针会作为帧信息一部分，一同传递到后级模块，由后级模块主动调用以释放帧 Buffer。

RegObserver / UnregObserver

【功能说明】

针对当前 ISP pipeline 上特定 pipe 特定 channel 的输出帧注册/注销观察者，以便 SensorMgr 获取输出帧以后传递帧数据给到所有已注册的观察者。

【接口定义】

```
AX_VOID RegObserver(AX_U32 nPipeID, AX_U32 nChannel, IObserver* pObserver)

AX_VOID UnregObserver(IObserver* pObserver)
```

【接口参数】

nPipeID：ISP pipe ID。

nChannel：ISP channel ID

pObserver：待注册/注销观察者实例指针。

【接口返回】

无。

【接口说明】

无。

UpdateAttrCB

【功能说明】

参数更新回调函数。在 CBaseSensor 实例加载配置文件配置参数后，以及利用配置文件配置参数设置 ISP 接口参数前，被唤醒来进行参数更新。

【接口定义】

```
static AX_BOOL UpdateAttrCB(ISensor* pInstance)
```

【接口参数】

pInstance：CBaseSensor 实例指针。

【接口返回】

成功返回 AX_TRUE，失败返回 AX_FALSE。

【接口说明】

此回调函数常用于在涉及 ISP 重新开流操作场景下，为保持当前设定，需要在每次 ISP 重新开流后，重新设置 ISP 属性。比如，当 Web 端设置日夜模式为 Night 模式后，由于每次 ISP 重新开流后，都会重置日夜模式，导致日夜模式状态无法保持，此种场景就需要在此回调中将日夜模式状态重新设置为当前 Web 端匹配状态。

SetAeSyncRatio

【功能说明】

设置 AE 同步参数

【接口定义】

```
AX_VOID SetAeSyncRatio(const AX_ISP_IQ_AE_SYNC_RATIO_T& tAeSyncRatio)
```

【接口参数】

tAeSyncRatio：AE 同步参数

AX_ISP_IQ_AE_SYNC_RATIO_T 定义参考《AX 3A 开发指南.docx》

【接口返回】

无。

【接口说明】

双摄拼接 AE 同步功能接口，参数通过标定生成

SetAwbSyncRatio

【功能说明】

设置 AWB 同步参数

【接口定义】

```
AX_VOID SetAwbSyncRatio(const AX_ISP_IQ_AWB_SYNC_RATIO_T& tAwbSyncRatio)
```

【接口参数】

tAwbSyncRatio：AWB 同步参数

AX_ISP_IQ_AWB_SYNC_RATIO_T 定义参考《AX 3A 开发指南.docx》

【接口返回】

无。

【接口说明】

双摄拼接 AWB 同步功能接口，参数通过标定生成

SetMuliSns3ASync

【功能说明】

使能 3A（AE/AWB）同步参数加载

【接口定义】

```
AX_BOOL SetMuliSns3ASync(AX_BOOL bSync)
```

【接口参数】

bSync：是否加载 3A 同步参数

【接口返回】

无。

【接口说明】

双摄拼接加载 3A 同步参数接口

2.2.3 CIVPSGrpStage

负责将一路 YUV 输入 IVPS 模块，按照 pipeline 要求做特定处理，比如一分多的分流，旋转，Crop，Resize，Dewarp 等。

Init / Deinit

【功能说明】

初始化/去初始化 IVPS 模块所需的参数信息。

【接口定义】

```
virtual AX_BOOL Init()

virtual AX_BOOL DeInit()
```

【接口参数】

无。

【接口返回】

Init：成功返回 AX_TRUE，失败返回 AX_FALSE。

DeInit：成功返回 AX_TRUE，失败返回 AX_FALSE。

【接口说明】

无。

Start / Stop

【功能说明】

根据 Init 接口配置的参数信息，启动/停止 IVPS 模块。

【接口定义】

```
AX_BOOL Start()

AX_BOOL Stop()
```

【接口参数】

无。

【接口返回】

Start：成功返回 AX_TRUE，失败返回 AX_FALSE。

Stop：成功返回 AX_TRUE，失败返回 AX_FALSE。

【接口说明】

无。

InitParams

【功能说明】

初始化 IVPS 模块相关参数。

【接口定义】

```
AX_BOOL InitParams()
```

【接口参数】

无。

【接口返回】

成功返回 AX_TRUE，失败返回 AX_FALSE。

【接口说明】

无。

RegObserver / UnregObserver

【功能说明】

针对当前 IVPS group 输出的某个 channel 注册/注销观察者，以便 IVPS 获取输出帧以后传递帧数据给到所有观察者。

【接口定义】

```
AX_VOID RegObserver(AX_S32 nChannel, IObserver* pObserver)

AX_VOID UnregObserver(AX_S32 nChannel, IObserver* pObserver)
```

【接口参数】

nChannel：输出帧 channel 号。

pObserver：待注册/注销观察者实例指针。

【接口返回】

无。

【接口说明】

无。

ProcessFrame

【功能说明】

非 link 模式下开启的线程函数，用于从队列中获取缓存帧后送入 IVPS 模块进行处理。

【接口定义】

```
virtual AX_BOOL ProcessFrame(CAXFrame* pFrame)
```

【接口参数】

pFrame：YUV 帧指针。

【接口返回】

成功返回 AX_TRUE，失败返回 AX_FALSE。

【接口说明】

此线程方法定义在基类 CAXStage 中，Start()方法中，基类根据传入的 bStartProcessingThread 标志决定是否开启此线程。

FrameGetThreadFunc

【功能说明】

通过调用 IVPS 模块 AX_IVPS_GetChnFrame 接口，循环获取 IVPS 输出帧，并传递帧数据给到所有观察者。

【接口定义】

```
AX_VOID FrameGetThreadFunc(IVPS_GET_THREAD_PARAM_PTR pThreadParam)
```

【接口参数】

pThreadParam：线程函数所需信息结构体指针。

【接口返回】

无。

【接口说明】

无。

VideoFrameRelease

【功能说明】

释放通过 IVPS 模块 AX_IVPS_GetChnFrame 接口获取到的输出帧 Buffer。

【接口定义】

```
virtual AX_VOID VideoFrameRelease(CAXFrame* pFrame)
```

【接口参数】

pFrame：YUV 帧指针。

【接口返回】

无。

【接口说明】

通过 IVPS 模块 AX_IVPS_GetChnFrame 接口获取到的帧需要在后级模块使用完毕后才可进行释放，完全依赖于后级模块处理逻辑，此函数指针会作为帧信息一部分，一同传递到后级模块，由后级模块主动调用以释放帧 Buffer。

2.2.4 CVideoEncoder

负责将前级模块输入的 YUV 帧送入 VENC 模块进行编码。

Init / Deinit

【功能说明】

初始化/去初始化 VENC 模块所需的参数信息。

【接口定义】

```
virtual AX_BOOL Init()

virtual AX_BOOL DeInit()
```

【接口参数】

无。

【接口返回】

Init：成功返回 AX_TRUE，失败返回 AX_FALSE。

DeInit：成功返回 AX_TRUE，失败返回 AX_FALSE。

【接口说明】

无。

Start / Stop

【功能说明】

根据 Init 接口配置的参数信息，启动/停止 VENC 模块。

【接口定义】

```
AX_BOOL Start()

AX_BOOL Stop()
```

【接口参数】

无。

【接口返回】

Start：成功返回 AX_TRUE，失败返回 AX_FALSE。

Stop：成功返回 AX_TRUE，失败返回 AX_FALSE。

【接口说明】

无。

InitParams

【功能说明】

初始化 VENC 模块相关参数。

【接口定义】

```
AX_BOOL InitParams()
```

【接口参数】

无。

【接口返回】

成功返回 AX_TRUE，失败返回 AX_FALSE。

【接口说明】

无。

RegObserver / UnregObserver

【功能说明】

针对当前类实例的输出通道注册/注销观察者，以便获取输出帧以后传递帧数据给观察者。

【接口定义】

```
AX_VOID RegObserver(IObserver* pObserver)

AX_VOID UnregObserver(IObserver* pObserver)
```

【接口参数】

pObserver：待注册/注销观察者实例指针。

【接口返回】

无。

【接口说明】

无。

ProcessFrame

【功能说明】

非 link 模式下开启的线程函数，用于从队列中获取缓存帧后送入 VENC 模块进行编码处理。

【接口定义】

```
virtual AX_BOOL ProcessFrame(CAXFrame* pFrame)
```

【接口参数】

pFrame：YUV 帧指针。

【接口返回】

成功返回 AX_TRUE，失败返回 AX_FALSE。

【接口说明】

此线程方法定义在基类 CAXStage 中，Start（）方法中，基类根据传入的 bStartProcessingThread 标志决定是否开启此线程。

FrameGetThreadFunc

【功能说明】

用于通过调用 VENC 模块 AX_VENC_GetStream 接口，循环获取一帧编码输出帧，并传递帧数据给观察者。

【接口定义】

```
AX_VOID FrameGetThreadFunc(CVideoEncoder* pCaller)
```

【接口参数】

pCaller：线程启动方实例指针，用于传递线程执行所需参数信息。

【接口返回】

无。

【接口说明】

无。

2.2.5 CJpegEncoder

负责将前级模块输入的 YUV 帧送入 JENC 模块进行编码。接口信息可参照 CVideoEncoder。

2.2.6 CPPLCfgParser

负责解析 ppl.json，获取各模块的前后级关联信息。

InitOnce

【功能说明】

单例模式下，仅被调用一次，用于进行文件的加载和解析。

【接口定义】

```
AX_BOOL InitOnce()
```

【接口参数】

无。

【接口返回】

成功返回 AX_TRUE，失败返回 AX_FALSE。

【接口说明】

此接口由基类 CAXSingleton 的 GetInstance 接口中进行调用。

ParseFile

【功能说明】

读入配置文件并调用解析文件接口进行解析。

【接口定义】

```
AX_BOOL ParseFile(const std::string& strPath,
        std::vector<IPC_MOD_RELATIONSHIP_T>& vecModRelations)
```

【接口参数】

strPath：配置文件路径。

vecModRelations: 上下级模块的关联表

【接口返回】

成功返回 AX_TRUE，失败返回 AX_FALSE。

【接口说明】

无。

ParseJson

【功能说明】

解析 json 格式配置文件，并将解析结果进行保存。

【接口定义】

```
AX_BOOL ParseJson(picojson::object& objJsonRoot,
```

```
std::vector<IPC_MOD_RELATIONSHIP_T>& vecModRelations)
```

【接口参数】

objJsonRoot：picojson 解析出的 json 结构根节点。

vecModRelations: 上下级模块的关联表

【接口返回】

成功返回 AX_TRUE，失败返回 AX_FALSE。

【接口说明】

无。

2.2.7 CSensorCfgParser

负责解析 sensor 相关 json 配置文件，以备 sensor 管理模块获取待加载的 sensor 配置信息。

同名接口均可参照上文 CPPLCfgParser 中对应同名接口描述。

GetConfig

【功能说明】

获取配置文件解析结果。

【接口定义】

```
AX_BOOL GetConfig(std::map<AX_U8, SENSOR_CONFIG_T>& mapSensorCfg, AX_U32&
nSensorCount)
```

【接口参数】

mapSensorCfg: <sensor_id, sensor 配置>

nSensorCount: sensor 数量

【接口返回】

成功返回 AX_TRUE，失败返回 AX_FALSE。

【接口说明】

无。

2.2.8 CIvpsCfgParser

负责解析 IVPS 模块相关 json 配置文件。

同名接口均可参照上文 CPPLCfgParser 中对应同名接口描述。

GetConfig

【功能说明】

获取配置文件解析结果。

【接口定义】

```
AX_BOOL GetConfig(std::map<AX_U8, IVPS_GROUP_CFG_T>& mapGrpSetting)
```

【接口参数】

mapGrpSetting：<ivps_group，ivps 配置>

【接口返回】

成功返回 AX_TRUE，失败返回 AX_FALSE。

【接口说明】

无。

2.2.9 CEncoderCfgParser

负责解析 Encoder 模块（包含 VENC/JENC/MJENC）相关 json 配置文件。

同名接口均可参照上文 CPPLCfgParser 中对应同名接口描述。

GetConfig

【功能说明】

获取配置文件解析结果中的编码相关配置信息。

【接口定义】

```
AX_BOOL GetConfig(std::map<AX_U8, VIDEO_CONFIG_T>& mapVencChnConfig,
std::map<AX_U8, JPEG_CONFIG_T>& mapJencChnConfig);
```

【接口参数】

mapVencChnConfig：<Venc_group，venc 配置>

mapJencChnConfig：<Jenc_group，venc 配置>

【接口返回】

成功返回 AX_TRUE，失败返回 AX_FALSE。

【接口说明】

无。

2.2.10 CBaseLinkage

封装系统 AX_SYS_Link/ AX_SYS_UnLink 方法，方便其他模块使用。

Link

【功能说明】

通过传入前后级模块信息，并使用 SDK 接口建立模块 link 关系。

【接口定义】

```
AX_S32 Link(const LINK_MOD_INFO_T& tLinkModInfo) const
```

【接口参数】

`tLinkModInfo`：待建立 link 关系的前后级模块信息。

【接口返回】

成功返回 0，失败返回 SDK 接口返回错误码。

【接口说明】

无。

UnLink

【功能说明】

通过传入前后级模块信息，并使用 SDK 接口撤销模块 link 关系。

【接口定义】

```
AX_S32 UnLink(const LINK_MOD_INFO_T& tLinkModInfo) const
```

【接口参数】

`tLinkModInfo`：待销毁 link 关系的前后级模块信息。

【接口返回】

成功返回 0，失败返回 SDK 接口返回错误码。

【接口说明】

无。

2.2.11 CAVS

负责 AVS 启动和板端标定功能的启动。

Init

【功能说明】

AVS 模块初始化。

【接口定义】

```
AX_BOOL Init(const AX_APP_AVS_ATTR_T& stAVSAttr)
```

【接口参数】

`stAVSAttr`：AVS 属性。

【接口返回】

成功返回 AX_TRUE，失败返回 AX_FALSE。

【接口说明】

标定参数不完整或破坏情况下，返回 AX_FALSE，但允许进入标定模式重新标定数据。

DeInit

【功能说明】

AVS 模块初始化。

【接口定义】

```
AX_BOOL DeInit()
```

【接口参数】

无。

【接口返回】

成功返回 AX_TRUE，失败返回 AX_FALSE。

【接口说明】

无。

Start/Stop

【功能说明】

AVS 模块初始化。

【接口定义】

```
AX_BOOL Start()

AX_BOOL Stop()
```

【接口参数】

无。

【接口返回】

成功返回 AX_TRUE，失败返回 AX_FALSE。

【接口说明】

无。

LoadParam

【功能说明】

加载 AVS 标定参数

【接口定义】

```
AX_BOOL LoadParam(string strCaliDataPath)
```

【接口参数】

strCaliDataPath：AVS 标定参数存储的路径

【接口返回】

成功返回 AX_TRUE，失败返回 AX_FALSE。

【接口说明】

无。

StartAVSCalibrate/ StopAVSCalibrate

【功能说明】

启动/停止板端标定

【接口定义】

```
AX_BOOL StartAVSCalibrate()

AX_VOID StopAVSCalibrate()
```

【接口参数】

无。

【接口返回】

成功返回 AX_TRUE，失败返回 AX_FALSE。

【接口说明】

无。

GetAvsResolution

【功能说明】

获取 AVS 拼接输出的分辨率

【接口定义】

```
AX_APP_AVS_RESOLUTION_T GetAvsResolution()
```

【接口参数】

无。

【接口返回】

标定输出的分辨率。

【接口说明】

无。

Get3ASyncRatio

【功能说明】

获取 3A 标定参数

【接口定义】

```
AX_AVSCALI_3A_SYNC_RATIO_T Get3ASyncRatio()
```

【接口参数】

无。

【接口返回】

3A 标定参数，AX_AVSCALI_3A_SYNC_RATIO_T 定义见《AX AVS Cali API 文档.docx》

【接口说明】

无。

2.2.12 其他辅助类

CAXRingBuffer

Ring Buffer 机制实现类，主要用于缓存编码后的图像帧。

CElapsedTimer

时间信息相关工具类。

CCommonUtils

通用工具类。

CAXStringHelper

字符串工具类。

CFramerateCtrlHelper

帧率控制工具类。

COptionHelper

运行参数配置文件解析类

log/*

log 相关实现代码

mp4enc/*

板端录像实现

webapp/*

Web 前端 VUE 工程代码。

webserver/*

搭建 Http Server/Websocket Server 的实现代码。

rtsp/*

搭建 RTSP Server 实现代码。

2.3 数据结构

2.3.1 AX_APP_AVS_ATTR_T

【说明】

定义 app avs 模块的属性

【定义】

```
typedef struct {
    AX_U8 u8PipeNum;
    AX_AVS_MODE_E enMode;
    AX_AVS_BLEND_MODE_E enBlendMode;
    AX_APP_AVS_PARAM_TYPE_E enParamType;
    AX_AVS_PROJECTION_MODE_E enProjectionType;
    AX_BOOL bSyncPipe;
    AX_BOOL bDynamicSeam;
    AX_FRAME_COMPRESS_INFO_T stAvsCompress;
    AX_U8 u8CaliEnable;
    string strCaliServerIP;
    AX_U16 u16CaliServerPort;
    AX_AVSCALI_INIT_PARAM_T tCaliInitParam;
} AX_APP_AVS_ATTR_T, *AX_APP_AVS_ATTR_PTR;
```

【成员】

成员名称	描述

u8PipeNum	AVS 拼接的 pipe 数
enMode	AVS 拼接模式，目前仅支持 AVS_MODE_BLEND。枚举 AX_AVS_MODE_E 定义见《AX AVS API 文档.docx》
enBlendMode	Blend 模式，默认 AVS_BLEND_PYRAMID。枚举 AX_AVS_BLEND_MODE_E 定义见《AX AVS API 文档.docx》
enParamType	AVS 拼接参数类型: E_AVS_PARAM_TYPE_NORMAL （加载参数） E_AVS_PARAM_TYPE_MESHTABLE（加载 mesh table）
enProjectionType	投影模式，输出视场角>=120°推荐柱面投影。 AX_AVS_PROJECTION_MODE_E 定义见《AX AVS API 文档.docx》
bSyncPipe	AVS 帧同步开关，默认开启。
bDynamicSeam	动态拼缝功能开启/关闭
stAvsCompress	AVS fbc 开关 [0, 0]：关闭 fbc 输出 [2, 4]：开启 fbc 输出
u8CaliEnable	是否使能板端标定功能
strCaliServerIP	板端标定连接服务端 ip 字符串
u16CaliServerPort	板端标定连接服务端端口号
tCaliInitParam	板端标定初始化参数，AX_AVSCALI_INIT_PARAM_T 定义见《AX AVS Cali API 文档.docx》

2.3.2 AX_APP_AVS_RESOLUTION_T

【说明】

定义 AVS 拼接输出分辨率

【定义】

```
typedef struct {
    AX_U32 u32Width;
    AX_U32 u32Height;
} AX_APP_AVS_RESOLUTION_T;
```

【成员】

成员名称	描述
u32Width	AVS 拼接输出图像宽
u32Height	AVS 拼接输出图像高

2.4 字库

为了节省内存，APP 默认使用的是内置点阵字库，支持中文 GB2312 中的字形和 ASCII 码字形。另外 APP 也支持 Freetype 的矢量字库，字形依赖于加载的字库文件。

2.4.1 Freetype 矢量字库

修改 Makefile 文件设置：

```
FONT_USE_FREETYPE=yes
```

字库文件在/app/components/resource/font 下，字库的名字可以在 demo 代码中搜索替换。

矢量字库使用的是开源的 libfreetype 库。代码参考：app/component/osd/OSDHandler.cpp

2.4.2 Bitmap 点阵字库

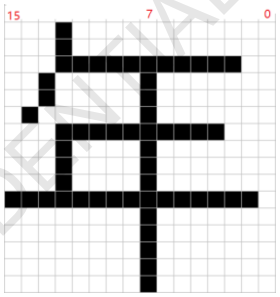
APP 默认使用的是内置点阵字库，参考代码：app/component/osd/font 和 app/component/osd/OSDHandler.cpp

FontEn16.h : ASCII 字库和索引文件（unicode 到字形索引），字形固定宽高：8x16

FontZh16.h：GB2312 字库和索引文件（unicode 到字形索引），字形固定宽高：16x16

字形点阵每个 bit 一个像素，0：表示透明，1：表示非透明。

如下下图中文字形“年”，bit0~15 的排列关系。



FontIndex.h/FontIndex.cpp 提供了两个函数：

GetFontBitmap：根据传入的 unicode，获取 bitmap。

OsdCalcStrSize：计算字符串的宽高，其中 fontsize 必须是 16 的倍数。

OSDHandler.cpp 主要提供字形的缩放并渲染到 ARGB1555 或 bitmap（单 bit）的 buffer 中。

☞ **说明：**

用户可以通过 libfreetype 从自己已有的矢量字库，导出单色的点阵 bitmap 数组，并填充到字形等宽的数组中，并做好 unicode 到字形的索引。

用到 libfreetype 的 api 如下，参考 OSDHandler.cpp 中代码：

FT_Load_Char(m_fontFace, u16CharCode, FT_LOAD_RENDER | FT_LOAD_MONOCHROME)。

3 FRTDemo

3.1 概要说明

3.1.1 FRTDemo 简介

FRTDemo 是前端产品 Demo，实现了 IPC 和 PANO 等不同业务形态

3.1.2 执行 FRTDemo

FRTDemo 可执行文件在板端/opt/bin/FRTDemo 目录，通过命令行参数**-p** 指定为 IPC 或者 PANO 业务形态，操作步骤：

步骤1 通过串口、SSH 或者 Telnet 进入 FRTDemo 目录（cd /opt/bin/FRTDemo）

步骤2 以 OS04A10 单摄为例，执行`./run.sh -p 0 -s 0 -n 1`命令以启动演示程序。具体命令行参数使用规则可通过 Help 信息获取，命令为：`./run.sh -h`

3.2 流程图

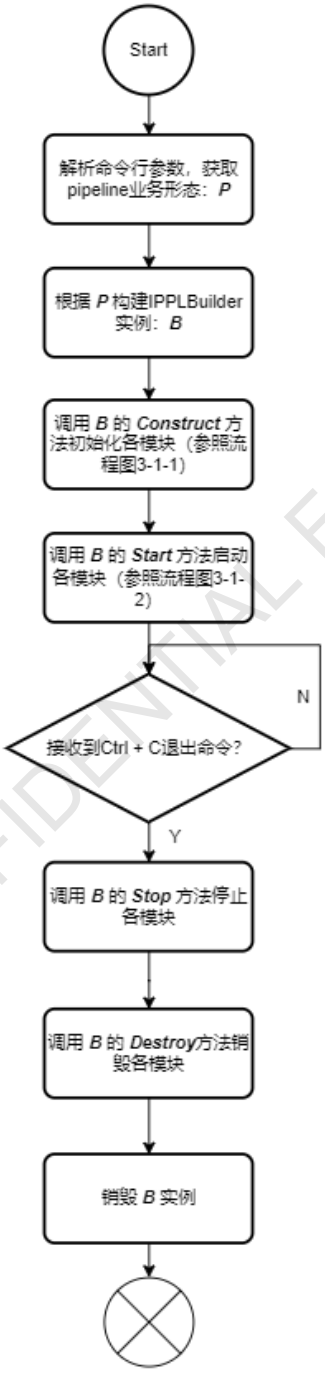


图3-1 FRTDemo 主流程图

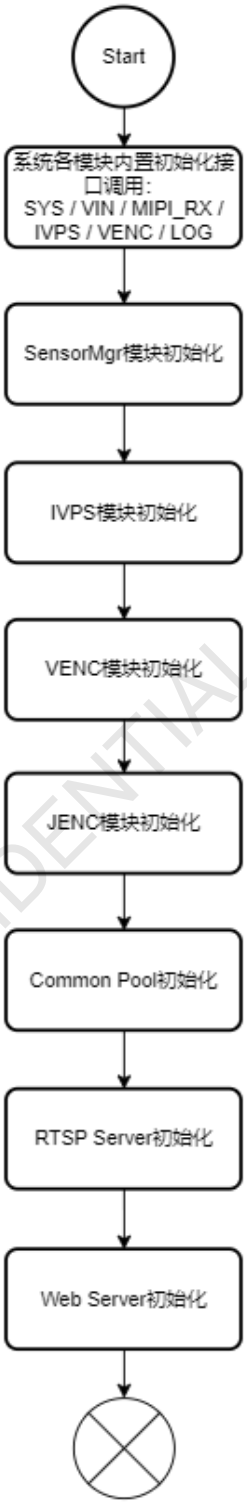


图 3-1-1 FRTDemo 模块初始化流程图

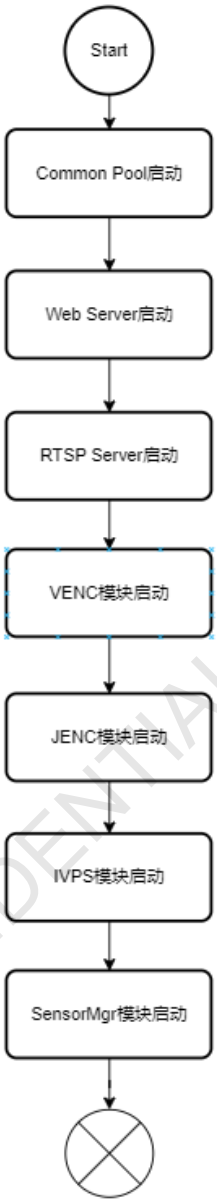


图 3-1-2 FRTDemo 模块启动流程图

说明

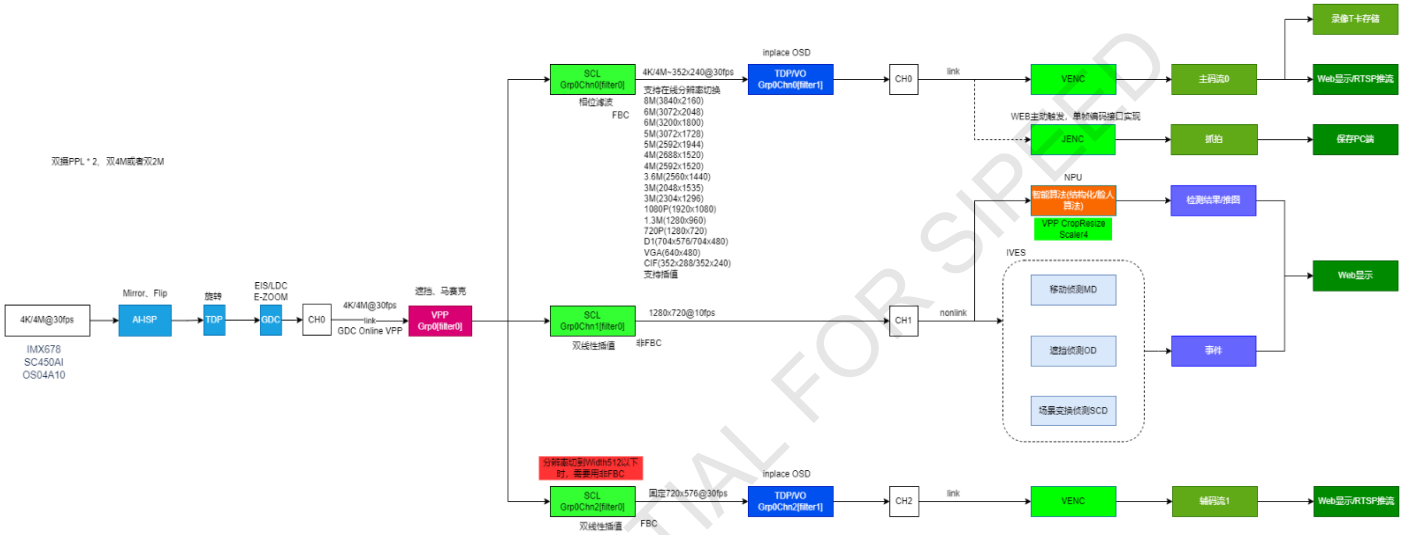
- 主程序（Main）通过解析命令行参数，识别当前执行的业务 pipeline 形态（即 FRTDemo-IPC 或者 FRTDemo-PANO 等），以及加载的 sensor 类型（OS04A10，SC450AI，IMX678 等），初始化对应的 IPPLBuilder 接口实现类实例（以下简称 PPL Builder）。再通过 PPL

Builder 的 Construct 和 Start 方法完成当前业务 PPL 所涉及各模块的初始化和启动。

- PPL Builder 作为整个 pipeline 的构建者，主要职责有以下几点：
  - 从配置文件读取各 SDK 模块配置参数，并按照一定顺序，初始化各模块。
  - 按照各模块配置参数，统计所需占用的公共 pool 的 size 列表以及各 size 对应的 block 数目，初始化 pool 模块。
  - 通过 ppl.json 配置文件配置的两两模块的上下级对应关系，建立 link table。
  - 按照一定顺序，启动各模块。
- ISP，IVPS 以及 VENC/JENC 等 SDK 模块可根据 ppl.json 中的“link”配置，或通过系统 link 方式，由模块内部自行进行帧 buffer 的传递，或通过 SDK 各模块提供的 GetFrame/SetFrame 接口，主动来进行帧 buffer 的传递。
- VENC/JENC/MJENC 模块转发线程通过 GetFrame 接口获取编码数据以后，送入 RTSP Server 或者 Websocket Server，或者两者都送以后，完成整个帧 Buffer 的流转。
- RTSP Server 转换编码后的 H264/H265 流为 RTSP 流以后，在有客户端请求并连接成功以后，推送 RTSP 流到 RTSP 客户端。
- Websocket Server 在有客户端连接请求并连接成功以后，推送编码后的 H264/H265/MJPEG 流或者单张 JPEG 图片到 websocket 客户端。
- Websocket 客户端通过 MSE（Media Source Extensions）解析 H264/H265/MJPEG/JPEG 并显示在网页上。

3.3 IPC 开发说明

3.3.1 Pipeline



3.3.2 关键类

3.3.2.1 CIPCBuilder

负责 FRTDemo-IPC 形态 pipeline 的搭建。包括各模块的初始化以及启动停止等。同时，CIPCBuilder 实例作为各模块实例拥有者，负责各模块协同运作。

Construct / Destroy

【功能说明】

Pipeline 运行所需各模块的创建/销毁以及初始化/反初始化。

【接口定义】

```
virtual AX_BOOL Construct(AX_VOID)

virtual AX_BOOL Destroy(AX_VOID)
```

【接口参数】

无。

【接口返回】

成功返回 AX_TRUE，失败返回 AX_FALSE。

【接口说明】

无。

Start / Stop

【功能说明】

Pipeline 运行所需各模块的启动与停止。

【接口定义】

```
virtual AX_BOOL Start(AX_VOID)

virtual AX_BOOL Stop(AX_VOID)
```

【接口参数】

无。

【接口返回】

成功返回 AX_TRUE，失败返回 AX_FALSE。

【接口说明】

启动/停止均需根据模块具体业务按照一定顺序进行。

ProcessWebOprs

【功能说明】

接收 Web 端操作请求，并转发到相应模块，进行具体切换操作。

【接口定义】

```
AX_BOOL ProcessWebOprs(WEB_REQUEST_TYPE_E eReqType, const AX_VOID* pJsonReq,
AX_VOID** pResult = nullptr)
```

【接口参数】

eReqType：web 操作请求类型。

pJsonReq：web 操作请求具体内容（MprJson 指针类型）。

pResult：web 操作请求返回值，比如 web 获取码流分辨率，码率信息等请求。

【接口返回】

成功返回 AX_TRUE，失败返回 AX_FALSE。

【接口说明】

无。

DispatchOpr

【功能说明】

转发 Web 操作请求到具体业务模块。

【接口定义】

```
AX_BOOL DispatchOpr(WEB_REQ_OPERATION_T& tOperation, AX_VOID** pResult =
nullptr)
```

【接口参数】

tOperation：web 操作请求结构体。

pResult：web 操作请求返回值。

【接口返回】

成功返回 AX_TRUE，失败返回 AX_FALSE。

【接口说明】

无。

3.4 Pano 开发说明

3.4.1 Pipeline

3.4.2 关键类

3.4.2.1 CPanoBuilder

负责 FRTDemo-Pano 形态 pipeline 的搭建。包括各模块的初始化以及启动停止等。同时，CPanoBuilder 实例作为各模块实例拥有者，负责各模块协同运作。

Construct / Destroy

【功能说明】

Pipeline 运行所需各模块的创建/销毁以及初始化/反初始化。

【接口定义】

```
virtual AX_BOOL Construct(AX_VOID)

virtual AX_BOOL Destroy(AX_VOID)
```

【接口参数】

无。

【接口返回】

成功返回 AX_TRUE，失败返回 AX_FALSE。

【接口说明】

Construct 中除基本模块初始化，还会完成 AVS 模块初始化，包括标定参数加载（**InitAvs**）。AVS 初始化工作需要在后级模块之前完成。

Destroy 中会完成 AVS 模块反初始化（**DeInit@ CAvs**）

Start / Stop

【功能说明】

Pipeline 运行所需各模块的启动与停止。

【接口定义】

```
virtual AX_BOOL Start(AX_VOID)

virtual AX_BOOL Stop(AX_VOID)
```

【接口参数】

无。

【接口返回】

成功返回 AX_TRUE，失败返回 AX_FALSE。

【接口说明】

启动/停止均需根据模块具体业务按照一定顺序进行。

Start 中除基本模块启动，还会完成 AVS 模块的启动（**Start@ CAvs**），如果板端标定功能开启，则包含 AVS 板端标定功能启动（**StartAVSCalibrate @ CAvs**）且在 sensor start 之后

Stop 中除基本模块停止，还会完成 AVS 模块的停止（**Stop@ CAvs**），如果板端标定功能开启，则包含 AVS 板端标定功能停止（**StopAVSCalibrate @ CAvs**）

ProcessWebOprs

【功能说明】

接收 Web 端操作请求，并转发到相应模块，进行具体切换操作。

【接口定义】

```
AX_BOOL ProcessWebOprs(WEB_REQUEST_TYPE_E eReqType, const AX_VOID* pJsonReq,
AX_VOID** pResult = nullptr)
```

【接口参数】

eReqType：web 操作请求类型。

pJsonReq：web 操作请求具体内容（MprJson 指针类型）。

pResult：web 操作请求返回值，比如 web 获取码流分辨率，码率信息等请求。

【接口返回】

成功返回 AX_TRUE，失败返回 AX_FALSE。

【接口说明】

无。

DispatchOpr

【功能说明】

转发 Web 操作请求到具体业务模块。

【接口定义】

```
AX_BOOL DispatchOpr(WEB_REQ_OPERATION_T& tOperation, AX_VOID** pResult =
nullptr)
```

【接口参数】

tOperation：web 操作请求结构体。

pResult：web 操作请求返回值。

【接口返回】

成功返回 AX_TRUE，失败返回 AX_FALSE。

【接口说明】

无。

APP_VIN_Stitch_Attr_Init

【功能说明】

完成 vin stitch 参数初始化

【接口定义】

```
AX_S32  APP_VIN_Stitch_Attr_Init()
```

【接口参数】

无。

【接口返回】

成功返回 0，失败返回非 0 值。

【接口说明】

完成 vin stitch 参数初始化，此接口调用需要放在 AX_VIN_Init 之后，AX_MIPI_RX_Init 之前。

CalibrateDone

【功能说明】

板端标定完成回调处理

【接口定义】

```
static AX_VOID CalibrateDone(AX_S32 result, AX_AVSCALI_AVS_PARAMS_T*
pAVSParams, AX_AVSCALI_3A_SYNC_RATIO_T* pSyncRatio, AX_VOID* pPrivData)
```

【接口参数】

result：标定结果返回，0：成功；非 0：失败

pAVSParams：几何标定参数，AX_AVSCALI_AVS_PARAMS_T 定义参考《AX AVS Cali API 文档》

pSyncRatio：3A 标定参数，AX_AVSCALI_3A_SYNC_RATIO_T 定义参考《AX AVS Cali API 文档》

pPrivData：用户私有数据指针，由调用 AX_AVSCALI_Init（接口定义参考《AX AVS Cali API 文档》）时传入

【接口返回】

无。

【接口说明】

回调函数中完成应用业务逻辑更新（比如 AVS 模块重启，后级模块分辨率更新...）

3.5 配置文件

FRTDemo 每个不同 pipeline 业务场景配置位于各 pipeline 独立配置文件夹下， IPC Pipeline 配置文件位于 bin/config/ipc 目录下，PANO Pipeline 配置文件位于 bin/config/pano 目录下。

3.5.1 Pipeline 配置

Pipeline 配置文件名为 ppl.json。具体参数配置如下表所示：

表3-1 Pipeline 配置参数

参数名	类型	示例	说明
scenario	array		scenario_1：单摄 scenario_0: 双摄
relations	array		前后模块串联关系集合
src_mod	string	"VIN" / "IVPS" / "VENC"	模块名称，对应 AX_MOD_ID_E，也可指定字

参数名	类型	示例	说明
		/ "JENC"	符串来定义自定义模块类型。主要定义非 link 场景用户自定义模块
src_grp	int		VIN 模块对应 pipe id；IVPS 模块对应 group；Encoder 模块无 group 概念，默认填 0；
src_chn	int		VIN 模块对应 channel id；IVPS 模块对应 channel id；Encoder 模块对应 channel id；
dst_mod	string	"VIN" / "IVPS" / "VENC" / "JENC"	同 src_mod
dst_grp	int		同 src_grp
dst_chn	int		同 src_chn
link	int	0 / 1	定义前后级模块是否通过 SDK 建立 link 关系

3.5.2 Sensor 配置

Sensor 配置文件可为多个。与命令行参数中<**-s**>选项一一匹配。具体参数配置如下表所示：

表3-2 Sensor 配置参数

参数名	类型	示例	说明
scenario	array		scenario_1：单摄 scenario_0：双摄 SensorMgr 初始化时解析所有场景配置，并按照当前配置的场景，加载对应场景配置

参数名	类型	示例	说明
sns_id	int	0	sensor id
dev_id	int	0	
pipe_id	int	0	
sns_framerate	int		sensor 输出帧率
pipe_framerate	int		pipe 输出帧率
sns_type	int		sensor 类型 0：OS04A10
sns_mode	int	1：AX_SNS_LINEAR_MODE 2：AX_SNS_HDR_2X_MODE	sensor 模式
dev_run_mode	int	0：Online；1：Offline	
sns_output_mode	int	0：AX_SNS_NORMAL	
yuv_depth	array<int>	2	每个 channel 的 depth
enable_channel	array<int>	0	每个 channel 的开关 0：OFF 1：ON
enable_aiisp	int	0：aiisp OFF 1：aiisp ON	aiisp 使能开关
tuning_ctrl	int	0：tuning ctrl OFF； 1：tuning ctrl ON；	
tuning_port	int		默认 8082
ife_compress	array<array<int>>		定义每个 pipe 的各 sensor 模式下的 ife compress 信息，包括 compress mode

参数名	类型	示例	说明
			和 compress level
ainr_compress	array<array<int>>		定义每个 pipe 的各 sensor 模式下的 ainr compress 信息，包括 compress mode 和 compress level
3dnr_compress	array<array<int>>		定义每个 pipe 的各 sensor 模式下的 3dnr compress 信息，包括 compress mode 和 compress level
chn_compress	array<array<int>>	[[2,4]]有损压缩，compress level 为 4	定义每个 pipe 中每个 channel 的 compress 信息，包括 compress mode 和 compress level
master_slave_select	int	0: AX_SNS_MASTER 1: AX_SNS_SLAVE 2: AX_SNS_SYNC_MASTER 3: AX_SNS_SYNC_SLAVE	默认配 0 双摄拼接模式： Master：2 Slave： 3

3.5.3 IVPS 配置

IVPS 配置文件名为 ivps.json。具体参数配置如下表所示：

表3-3 Sensor 配置参数

参数名	类型	说明
scenario	array	scenario_1：单摄 scenario_0: 双摄 IVPS 切换操作涉及大量 IVPS 通道以及通道属性变更场景，直接加载另一套配置。
|-- grp_id	int	
|-- grp_info	object	group 配置
|-- |-- grp_chn_num	int	group 包含的通道个数
|-- |-- engine_filter_0	AX_IVPS_ENGINE_E	filter_0 配置 engine，即 group filter 0x00
|-- |-- engine_filter_1	AX_IVPS_ENGINE_E	filter_1 配置 engine，即 group filter 0x01
|-- |-- engine_filter_2	array<AX_IVPS_ENGINE_E>	filter_2 配置 engine，即 channel filter 0x10 / 0x20 / 0x30 等
|-- |-- engine_filter_3	array<AX_IVPS_ENGINE_E>	filter_3 配置 engine，即 channel filter 0x11 / 0x21 / 0x31 等
|-- |-- grp_framerate	map<string, int>	group 输出帧率配置(仅作用于 group filter:0x00)
|-- |-- chn_framerate	array<map<string, int>>>	channel 输出帧率配置
|-- |-- grp_resolution	map<string, int>	group 输出分辨率配置(仅作用于 group filter:0x00)
|-- |-- chn_resolution	array<map<string, int>>>	channel 输出分辨率配置，stride 按打开 FBC 时 128 对齐，关闭 FBC 时 2 对齐

参数名	类型	说明
|-- |-- chn_link_flag	array<int>	对应输出 channel 是否为 link 模式，以此决定 get frame 线程是否往后级模块输送数据
|-- |-- chn_fbc	array<map<string, int>>	对应输出 channel 是否开启 FBC；可配置 compress mode 和 compress level
|-- |-- grp_inplac	array<int>	group inpace 模式配置(仅作用于 group filter:0x00，group filter:0x01)
|-- |-- chn_inplace	array<array<int, int>>>	channel inplace 模式配置(仅作用于 group filter:0x02，group filter:0x03)
|-- |-- chn_link_flag	array<int>	对应输出 channel 是否为 link 模式，以此决定 get frame 线程是否往后级模块输送数据
|-- |-- chn_scl	array<int>	channel scl 模式配置(仅作用于 group filter:0x02，group filter:0x03)
|-- |- - chn_voosd_flag	array<int>	对应输出 channel 是否使用 VO 绘制 OSD
|-- |- - spec_engine_config	object	特殊的 ivps(此 ivps 负责叠加隐私遮挡、马赛克)

3.5.4 Encoder 配置

Encoder 配置文件名为 encoder.json。具体参数配置如下表所示：

表3-4 Encoder 配置参数

参数名	类型	示例	说明
scenario	object		scenario_1： 单摄 scenario_0: 双摄
|-- instance	array		编码类型父结点，下面可以配置多个编码类型
|-- encoder_type	int	0	编码类型 0: H264, 1: MJPEG, 2:H265, 3: JENC
|-- channel	int	0	编码通道号
|-- pipe_src	int		数据源基于哪个 sensor
|-- bitrate	int		平均码率，单位：kbps
|-- in_fifo_depth	int	1	输入缓存数据块个数
|-- out_fifo_depth	int	1	输出缓存数据块个数
|-- mem_source	int	0：CMM； 1：Pool；	使用的内存类型
|-- gop	int		IDR 帧间隔
|-- enc_rc_info	array		
|--type	int	0: H264； 1: MJPEG； 2: H265； 3: JENC	需要生效的编码类型
|-- rc	array		
|-- |-- type	string	0	rc 类型 0: CBR, 1: VBR, 2:FIXQP

参数名	类型	示例	说明
|-- |-- min_qp	int	1	P 帧 qp 范围最小值 [0, 51]
|-- |-- max_qp	int	51	P 帧 qp 范围最大值 [0, 51]
|-- |-- min_iqp	int	1	I 帧 qp 范围最小值 [0, 51]
|-- |-- max_iqp	int	51	I 帧 qp 范围最大值 [0, 51]
|-- |-- min_i_prop	int	10	最小 I/P 帧码率的比值
|-- |-- max_i_prop	int		最大 I/P 帧码率的比值
|-- |-- intra_qp_delta	int		

3.5.5 Pano AVS 配置

AVS 配置文件名为 avs.json。具体参数配置如下表所示：

参数名	类型	归属	示例	说明
pipe_num	int	-	2	拼接 pipe 数
sync_pipe	bool	-	true	拼接帧同步处理开关
mode	int	-	0	0: AVS_MODE_BLEND
dynamic_seam	bool			动态拼接开关
blend_mode	int			0: AVS_BLEND_ALPHA; 1: AVS_BLEND_PYRAMID
param_type	int			0:

参数名	类型	归属	示例	说明
				AVS_CALIBRATION_PARAM_CAMERA; 1: AVS_CALIBRATION_PARAM_TRANSFORM
projection_type	int			0: AVS_PROJECTION_EQUIRECTANGULER; 1: AVS_PROJECTION_RECTLINEAR; 2: AVS_PROJECTION_CYLINDRICAL; 4:AVS_PROJECTION_CUBE_MAP
avs_compress	array<array<int>>		[2, 4]	[0, 0]: compress off; [2, 4]: compress on
cali_enable	int			0: disable 1: enable
cali_server_ip	string			"192.168.2.10"
cali_server_port	int			9999
param_file_path	string			"/param/avs/os04a10/"

3.6 常见问题

3.6.1 Sensor 配置不匹配

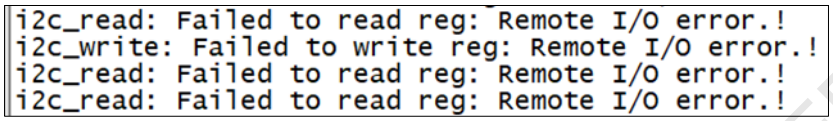


图3-2 Sensor 不匹配

【产生原因】

通常原因为命令行指定 sensor type(命令行参数：-s)启动时，加载的 Sensor 配置文件与实际外接设备不匹配。

【解决方法】

修改为正确的 sensor type 命令行参数即可。

3.6.2 Web 切换码流失败

图3-3 Web 切换码流失败

【产生原因】

通常原因为在 Web 页不关闭的场合，FRTDemo 重启以后，由于 Web 页的自动重连机制，Web 页可以继续获取 Websocket Server 推送的 H264/H265 流并显示视频图像，但由于处于非认证状态，导致切换码流失败。

【解决方法】

可以重新登录以后恢复正常。