



AX IPCDemo 用户指南

文档版本: V1.5

发布日期: 2022/08/05

AXERA CONFIDENTIAL FOR Sipeed

前 言	4
目 录	
修订历史	5
1 概述	6
1.1 概要说明	7
1.2 模块框图	7
2 编译和执行	8
2.1 编译	9
2.2 执行	9
3 开发说明	10
3.1 主要模块时序图	11
3.2 实现类	12
3.2.1 CBaseSensor	12
3.2.2 CCamera	15
3.2.3 CIVPSStage	16
3.2.4 CVideoEncoder	18
3.2.5 CDetect	20
3.2.6 其他辅助类	20
3.3 EIS 实现	22
3.3.1 添加 EIS 使能参数配置	22
3.3.2 Sensor 使能状态初始化	25
3.3.3 配置 Sensor Chn 参数	26
3.3.4 EIS Load bin 并配置 EIS IQ 参数	27

3.3.5 设置 IVPS Engine	29
3.3.6 关闭 link 模式	29
3.4 LDC 实现.....	30
3.4.1 添加相机内参和畸变参数配置	30
3.4.2 GDC 参数配置	32
4 配置文件	34
5 新增 sensor	37
5.1 新增 sensor 流程图	37
5.2 新增 sensor 配置文件.....	38
5.3 在 COptionHelper 的接口 ParseArgs 指定 SensorID.....	39
5.4 新增 Sensor 操作实例类.....	39
5.5 在 CBaseSensor 中实例化 sensor 操作类	42
5.5.1 在 CBaseSensor 头文件中增加新的 sensor 枚举支持	42
5.5.2 在接口 CBaseSensor::NewInstance 中增加 sensor 实例化	43
5.5.3 在接口 CBaseSensor::GetNpuAttr 中设置 sensor 使用的 NPU 属性	44
6 常见问题	45
6.1 Sensor 配置不匹配.....	45
6.2 Web 切换码流失败	45

权利声明

爱芯元智半导体(上海)有限公司或其许可人保留一切权利。

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

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。

AXERA CONFIDENTIAL FOR Speed

适用产品



前言

爱芯 AX620A 和爱芯 AX620U

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/AX620A_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。

文档版本	发布时间	修订说明
V1.0	2021/08/26	文档初版
V1.1	2021/09/26	添加 ISP3 路输出，T 卡存储等相关说明
V1.2	2021/11/29	复杂 Pipeline 实现
V1.3	2022/01/28	精简 Pipeline 和增加 Detect 模块
V1.4	2022/05/30	增加 EIS 功能说明
V1.5	2022/08/05	增加 LDC 功能说明

本章节包含：

[1.1 概要说明](#)

[1.2 模块框图](#)

1 概述

AXERA CONFIDENTIAL FOR Sipeed

1.1 概要说明

IPCDemo 是 IPC 产品类的演示应用程序。涉及到 ISP, IVPS, VEnc/JEnc, AI-SDK 等多个模块。

1.2 模块框图

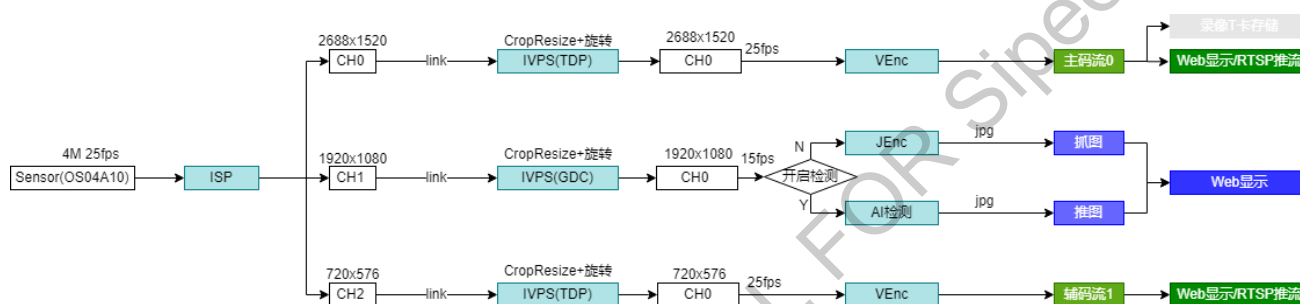


图1-1 模块框图

如上图所示，IPCDemo 的主要模块有：

- ISP：负责从 Sensor 获取图像 RAW 数据并转为 YUV，最终分 3 路通道输出以上信息。
- IVPS：图像视频处理模块。实现对视频图形进行一分多、Resize、Crop、旋转等功能。
- VENC / JENC：视频/JPEG 编码输出。
- Detect：支持人脸或结构化检测
- Web 显示：实现 H264 流的 Web 传输和提供 Web 方式查看实时视频。
- RTSP 推流：实现 H264 流的 RTSP 封装以及传输。
- 录像 T 卡存贮：封装 H264 流为 MP4 格式文件并保存至 T 卡或者 FLASH 空间。

本章节包含：

2 编译和执行

[2.1 编译](#)

[2.2 执行](#)

AXERA CONFIDENTIAL FOR Sipeed

2.1 编译

IPCDemo 源码随 AX SDK 发布，位于 `app/IPCDemo` 目录。

编译步骤

步骤1 执行 `cd app/IPCDemo` 命令

步骤2 执行 `make p=xxx clean` 命令。

步骤3 执行 `make p=xxx` 命令。

说明：

`p=xxx` 表示项目名称，比如 `p=AX620_demo`，可通过 `make plist` 命令查询 SDK 支持的项目列表。

2.2 执行

IPCDemo 默认安装在 `/opt/bin/IPCDemo` 目录。

操作步骤

步骤1 按需修改工具配置文件：`config/ipc_demo.conf`。通常无需进行任何修改。

步骤2 命令：`vi config/ipc_demo.conf`。

步骤3 可配置属性包括 `Detection` 开关/日志级别等。

步骤4 通过串口或者 SSH 进入 IPCDemo 目录，执行 `./run.sh` 命令。

工具当前会默认加载 `config/os08a20_config.json` 文件来对 `sensor` 属性进行相关配置。后续支持各种不同类型 `sensor` 以后，可以在此命令后手动加上 `json` 配置文件路径来区分加载何种 `sensor`。例如：`./run.sh config/os04a10_config.json`。

本章节包含：

[3.1 主要模块时序图](#)

[3.2 实现类](#)

3 开发说明

AXERA CONFIDENTIAL FOR Sipeed

3.1 主要模块时序图

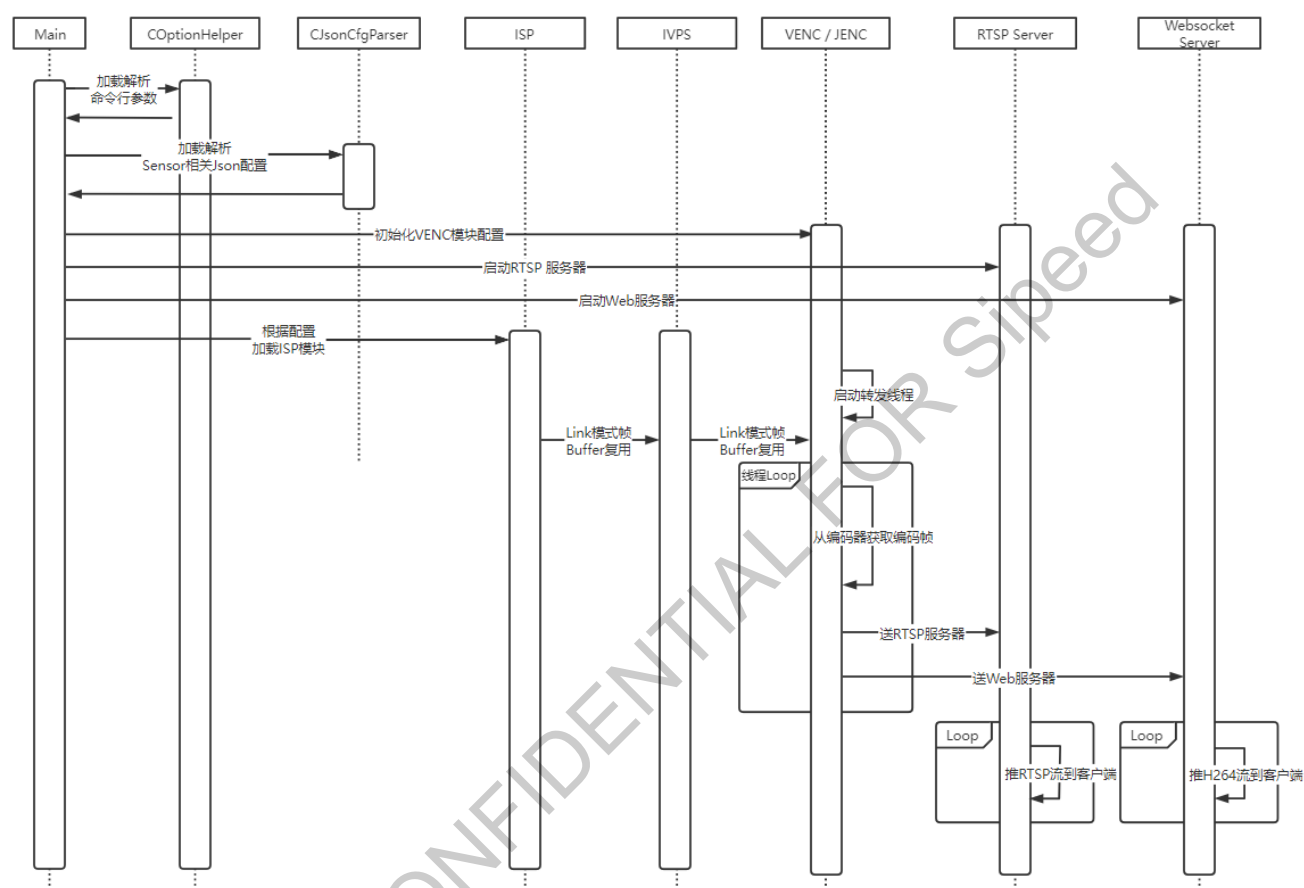


图3-1 主要模块时序图

说明

- 主程序（Main）通过加载解析命令行参数以及 Sensor 相关配置（XXXX_config.json），来初始化 ISP 模块、IVPS 模块以及 VENC 模块。
- ISP 和 IVPS 以及 VENC/JENC 通过配置 Link 方式来进行帧 buffer 复用以及传递。
- VENC/JENC 模块转发线程获取编码帧以后，VENC 转发编码帧到 RTSP Server 以及 Websocket Server，JENC 转发编码图片到 Websocket Server。
- RTSP Server 转换编码后的 H264 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 流到 websocket 客户端。
- Websocket 客户端通过 MSE（Media Source Extensions）解析 H264 流并显示在网页上。

3.2 实现类

IPCDemo 通过调用 AX SDK ISP 和 VENC 等模块的 API 接口实现应用层业务功能。本章节主要描述 IPCDemo 源码中比较关键的实现类功能和接口，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。

3.2.1 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 COS04a10。

Init

【功能说明】

Sensor 初始化流程所需属性的配置流程创建，通过不同的实现类实现不同属性的配置。

【接口定义】

```
virtual AX_BOOL Init(AX_U8 nSensorID, AX_U8 nDevID, AX_U8 nPipeID,  
AX_POOL_FLOORPLAN_T *stVbConf, AX_U8& nCount);
```

【接口参数】

nSensorID / nDeviceID / nPipeID 为 sensor 创建不同 pipeline（比如多 pipe 场景）过程中各 API 接口所需指定的参数。stVbConf 以及 nCount 为配置 ax_pool 所需的相关的参数。

【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 NPU/MIPI/Sensor/Device/Pipe 相关属性。

Open/Close

【功能说明】

Sensor 出流流程的创建与关闭。

【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。

NewInstance

【功能说明】

根据加载的 json 配置文件中配置的 sensor 类型，实例化不同的 sensor，并配置不同的 sensor 出流所需属性。

【接口定义】

```
static CBaseSensor* NewInstance(AX_U8 nSensorIndex);
```

【接口参数】

nSensorIndex 指定加载 json 配置文件中的哪个 sensor 配置。

【接口返回】

成功返回实例化后的 sensor 类指针，失败返回空指针。

【接口说明】

无

BuffPoolInit/BuffPoolExit

【功能说明】

根据 Init 接口收集到的 ax_pool 信息，进行 ax_pool 的配置与释放。

【接口定义】

```
static AX_BOOL BuffPoolInit(AX_POOL_FLOORPLAN_T *stVbConf, AX_U8 nCount)

static AX_BOOL BuffPoolExit();
```

【接口参数】

stVbConf、nCount 为 Init 接口收集到的配置参数信息。

【接口返回】

成功返回 AX_TRUE，失败返回 AX_FALSE。

【接口说明】

无

CalcBufSize

【功能说明】

计算 ax_pool 需要开辟的 buffer 的大小以及个数。

【接口定义】

```
AX_VOID CalcBufSize(AX_U8 nPipe, AX_POOL_FLOORPLAN_T *stVbConf, AX_U8&
nCount, AX_U32 nPreISPBlkCnt = 10, AX_U32 nNpuBlkCnt = 5);
```

【接口参数】

nPipe 为 pipe ID。

stVbConf、nCount 为 ax_pool 需要收集的信息。

nPreISPBlkCnt 为 PreISP 模块配置的默认 buffer 个数

nNpuBlkCnt 为给 NPU 配置的默认 buffer 个数

【接口返回】

无。

【接口说明】

无

3.2.2 CCamera

负责通过 CBaseSensor 启动 ISP 出流，并创建线程获取视频图像流。

Open/Close**【功能说明】**

通过 CBaseSensor 及其实现类创建出流以及断流流程。

【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。

Start/Stop**【功能说明】**

开启/停止 ITP 唤醒线程，以通知 ITP 出流。

【接口定义】

```
AX_BOOL Start()
```

```
AX_VOID Stop()
```

【接口参数】

无。

【接口返回】

Start: 成功返回 AX_TRUE, 失败返回 AX_FALSE。

Stop: 无。

【接口说明】

- Start 创建一个 RtpThreadFunc 线程来使能出流。
- Stop 用来停止 RtpThreadFunc 线程以停止出流。

3.2.3 CIVPSStage

负责将 ISP 输出的 YUV 帧按照 pipeline 要求做特定处理, 比如一分多, 旋转, Resize 等。

Init / Deinit**【功能说明】**

初始化/去初始化 IVPS 模块所需的参数信息。同时开启/关闭输出帧获取线程。

【接口定义】

```
virtual AX_BOOL Init()
```

```
virtual AX_VOID DeInit()
```

【接口参数】

无。

【接口返回】

Init: 成功返回 AX_TRUE, 失败返回 AX_FALSE。

DeInit: 无。

【接口说明】

无。

StartIVPS / StopIVPS**【功能说明】**

根据 Init 接口配置的参数信息, 初始化/去初始化 IVPS 模块。

【接口定义】

```
AX_BOOL StartIVPS()
```

```
AX_BOOL StopIVPS()
```

【接口参数】

无。

【接口返回】

Init: 成功返回 AX_TRUE, 失败返回 AX_FALSE。

DeInit: 无。

【接口说明】

无。

GrpGetThreadFunc**【功能说明】**

线程处理函数, 用于非 link 模式下获取 ivps 模块的输出帧, 并传递到后续模块。

【接口定义】

```
AX_VOID GrpGetThreadFunc (IVPS_GRP_THREAD_PARAM_PTR pThreadParam)
```

【接口参数】

线程函数入参，主要包含 IVPS 模块当前配置的 group 个数以及每个 group 对应的 channel 数。

【接口返回】

无。

【接口说明】

无。

3.2.4 CVideoEncoder

负责将 ISP 输出的 YUV 帧送入 VENC 模块进行编码。

Start/Stop**【功能说明】**

初始化/去初始化 VideoEncoder 模块，以及开启/关闭编码帧获取线程。

【接口定义】

```
virtual AX_BOOL Start (AX_BOOL bReload = AX_TRUE)
```

```
virtual AX_VOID Stop (AX_VOID)
```

【接口返回】

Open: 成功返回 AX_TRUE，失败返回 AX_FALSE。

Close: 无。

【接口说明】

YUV 帧获取线程由基类 CStage 中创建。

ProcessFrame

【功能说明】

线程处理函数，处理由 ISP 获取的每一帧 YUV 数据。

【接口定义】

```
AX_BOOL CVideoEncoder::ProcessFrame(CMediaFrame* pFrame)
```

【接口参数】

pFrame 为前一模块，即 IVPS 模块传入的带 YUV 数据的类指针。

【接口返回】

成功返回 AX_TRUE，失败返回 AX_FALSE。

【接口说明】

无。

GetThreadFunc

【功能说明】

线程处理函数，从编码模块获取编码后的帧。

【接口定义】

```
static AX_VOID *GetThreadFunc(AX_VOID *__this)
```

【接口参数】

__this 为类 CVideoEncoder 实例。

【接口返回】

无。

【接口说明】

无。

InitParams

【功能说明】

初始化 VENC 模块配置参数。

【接口定义】

```
AX_BOOL InitParams(VIDEO_CONFIG_T &config)
```

【接口参数】

VIDEO_CONFIG_T 封装了从配置文件获取的 VENC 相关属性。

【接口返回】

成功返回 AX_TRUE，失败返回 AX_FALSE。

【接口说明】

此接口通过 AX_VENC_CreateChn 配置相关参数。

3.2.5 CDetect

负责将 yuv 帧送到 AI SDK 做检测，将检测的结果保存到全局结构中。

具体 AI SDK 的使用可以参考 AX620 SDK 包中的 third-party/ai-sdk/sdk/sample。

3.2.6 其他辅助类

CConfigParser/CJsonCfgParser

配置文件解析基类以及 Json 配置文件解析实现类。

CAppLog

应用日志操作工具，支持多日志文件循环写入。

COptionHelper

配置参数管理封装。

CPrintHelper

终端信息打印类。

CAXRingBuffer

Ring Buffer 机制实现类，用于缓存编码后的图像帧。

CTimeUtils

生成用于 OSD 显示的时间信息。

CCommonUtils

通用工具类。

CStringUtils

字符串工具类。

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。

Stages/*

Pipeline 中各流程处理类，通过在 main 函数中指定各 stage 的先后关系，由 CStage 基类中的 EnqueueFrame 和 ProcessFrame 实现数据处理与传递。

3.3 EIS 实现

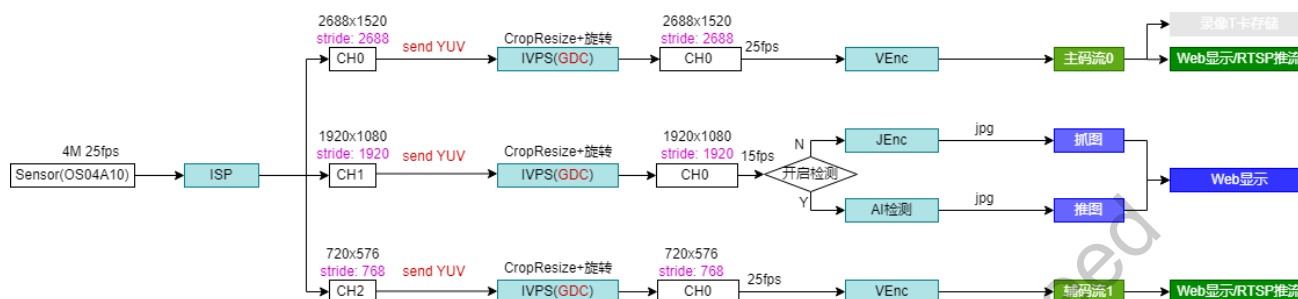


图3-2 模块框图（EIS 使能）

ISP 加载带 EIS 的模型(通过 loadbin 加载 tuning 参数)，YUV 输出包含 EIS 信息，通过 IVPS 的 GDC 引擎处理 EIS 信息（只有 GDC 引擎支持 EIS 信息处理），完成防抖 YUV 帧输出。

3.3.1 添加 EIS 使能参数配置

对应 Sensor json 配置文件添加 EIS 参数配置信息

```

"modules": [{
  "type": "camera",
  "instance": [{
    "id": 0,
    "frame_rate": 25,
    "sns_type": "0: OS04A10",
    "sns_type": 0,
    "sns_mode": "1: AX_SNS_LINEAR_MODE; 2: AX_SNS_HDR_2X_MODE",
    "sns_mode": 1,
    "run_mode": "0: NPU; 1: None NPU",
    "run_mode": 0,
    "normal_mode_bin": "/opt/etc/os04a10_sdr_wnr2d.bin",
    "hotbalance_mode_bin": "/opt/etc/os04a10_sdr_wnr3d.bin",
    "eis_enable": "0: Disable EIS; 1: Enable EIS",
    "eis enable": 0,
    "eis_sdr_bin": "/opt/etc/os04a10_sdr_eis_on.bin",
    "eis_hdr_bin": "/opt/etc/os04a10_hdr_eis_on.bin"
  ]
}]

```

EIS 功能使能

EIS load bin 路径

```
    ]
    },

```

说明:

*低内存方案不支持 EIS, **AX620U 不支持 EIS**

*EIS 功能需要模型支持, 需要确认对应 sensor 是否支持 EIS 功能

_SENSOR_CONFIG_T 增加对应成员变量

BaseSensor.h 文件中结构体 _SENSOR_CONFIG_T 增加 EIS 相关成员变量

```
typedef struct _SENSOR_CONFIG_T
{
    AX_U32          nFrameRate;
    AX_RUN_MODE_E   eRunMode;
    SNS_TYPE_E      eSensorType;
    AX_SNS_HDR_MODE_E eSensorMode;
    AX_CHAR          aNormalModeBin[SENSOR_BIN_PATH_LEN];
    AX_CHAR          aHotbalanceModeBin[SENSOR_BIN_PATH_LEN];
    AX_CHAR          aEISSdrBin[SENSOR_BIN_PATH_LEN];
    AX_CHAR          aEISHdrBin[SENSOR_BIN_PATH_LEN];
    AX_BOOL          bTuning;
    AX_U32           nTuningPort;
    AX_BOOL          bEnableEIS;
    CAMERA_CHAN_CFG_T arrChannels[MAX_ISP_CHANNEL_NUM];

    _SENSOR_CONFIG_T() {
        memset(this, 0, sizeof(_SENSOR_CONFIG_T));

        nFrameRate      = 25;
        eRunMode         = AX_ISP_PIPELINE_NORMAL;
        eSensorType      = E_SNS_TYPE_MAX;
        eSensorMode      = AX_SNS_LINEAR_MODE;
        bTuning          = AX_TRUE;
        nTuningPort      = 8082;
        bEnableEIS       = AX_FALSE;
    }
}
```



```
} SENSOR_CONFIG_T, *SENSOR_CONFIG_PTR;
```

Json 解析 EIS 参数

```
AX_BOOL CJsonCfgParser::GetCameraCfg(SENSOR_CONFIG_T &stOutCfg, SENSOR_ID_E
eSensorID)
{
    ...
    ...
    stOutCfg.eSensorType = (SNS_TYPE_E)objSensor["sns_type"].get<double>();
    stOutCfg.eSensorMode =
    (AX_SNS_HDR_MODE_E)objSensor["sns_mode"].get<double>();
    stOutCfg.nFrameRate = objSensor["frame_rate"].get<double>();
    stOutCfg.eRunMode = objSensor["run_mode"].get<double>() == 0 ?
AX_ISP_PIPELINE_NORMAL : AX_ISP_PIPELINE_NONE_NPU;
    if (objSensor.end() != objSensor.find("normal_mode_bin")) {
        strncpy(stOutCfg.aNormalModeBin,
objSensor["normal_mode_bin"].get<std::string>().c_str(), SENSOR_BIN_PATH_LEN
- 1);
    }
    if (objSensor.end() != objSensor.find("hotbalance_mode_bin")) {
        strncpy(stOutCfg.aHotbalanceModeBin,
objSensor["hotbalance_mode_bin"].get<std::string>().c_str(),
SENSOR_BIN_PATH_LEN - 1);
    }

    // EIS config
    if (objSensor.end() != objSensor.find("eis_enable")) {
        stOutCfg.bEnableEIS = objSensor["eis_enable"].get<double>() == 1 ?
AX_TRUE : AX_FALSE;
    }
    if (objSensor.end() != objSensor.find("eis_sdr_bin")) {
        strncpy(stOutCfg.aEISSdrBin,
objSensor["eis_sdr_bin"].get<std::string>().c_str(), SENSOR_BIN_PATH_LEN -
1);
    }
    if (objSensor.end() != objSensor.find("eis_hdr_bin")) {
        strncpy(stOutCfg.aEISHdrBin,
objSensor["eis_hdr_bin"].get<std::string>().c_str(), SENSOR_BIN_PATH_LEN -
1);
    }
}
```

获取 EIS 配置参数

```

}
...
}

```

3.3.2 Sensor 使能状态初始化

应用启动时（main.cpp），判断 EIS 是否使能，初始化 EIS 使能状态

```

int main(int argc, const char *argv[])
{
    ...
    ...
    AX_S32 nRet = GlobalApiInit();
    if (0 != nRet) {
        APP_LogClose();
        exit(1);
    }

    gPrintHelper.Start();

    AX_POOL_FLOORPLAN_T tVBConfig = {0};
    AX_U8 nPoolCount = 0;
    thread* pThreadCheckAutoSleep = nullptr;

    //Check whether support EIS
    #ifndef AX_SIMPLIFIED_MEM_VER
        EISSupportStateInit();
    #endif

    CMPEG4Encoder *pMpeg4Encoder = CMPEG4Encoder::GetInstance();
    RESULT_CHECK(pMpeg4Encoder);

    CMD::GetInstance()->SetWebServer(&g_webserver);
    COD::GetInstance()->SetWebServer(&g_webserver);
    ...
}
...
...
AX_VOID EISSupportStateInit()

```

支持 EIS 的条件：Sensor 对应的 json 文件中 eis_enable 配置为 1；eis_sdr_bin 和 eis_hdr_bin 配置路径均有效

```

{
#ifdef AX_SIMPLIFIED_MEM_VER
    SENSOR_CONFIG_T tSensorCfg;
    CConfigParser().GetInstance()->GetCameraCfg(tSensorCfg, E_SENSOR_ID_0);
    if (AX_FALSE == tSensorCfg.bEnableEIS) {
        LOG_M(MAIN, "EIS is unsupported!");
        gOptions.SetEISSupport(AX_FALSE);
        return;
    }

    if (access(tSensorCfg.aEISSdrBin, F_OK) != 0
        || access(tSensorCfg.aEISHdrBin, F_OK) != 0) {
        gOptions.SetEISSupport(AX_FALSE);
        LOG_M(MAIN, "EIS is unsupported for EIS sdr bin(%s) or hdr bin(%s) is
not exist.", tSensorCfg.aEISSdrBin, tSensorCfg.aEISHdrBin);
    } else {
        gOptions.SetEISSupport(AX_TRUE);
    }
}
#endif
}

```

3.3.3 配置 Sensor Chn 参数

```

AX_BOOL CBaseSensor::InitISP(AX_VOID)
{
    ...
    ...
    InitChnAttr();

    if (gOptions.IsEISSupport()) { //ALIGN_UP for GDC
        for (AX_U8 i = 0; i < MAX_ISP_CHANNEL_NUM; i++) {
            m_tChnAttr.tChnAttr[i].nWidthStride =
ALIGN_UP(m_tChnAttr.tChnAttr[i].nWidthStride, GDC_STRIDE_ALIGNMENT);
        }
    }
    ...
}

```

确保各 chn 对应的 stride 满足 GDC 对齐要求（64 位对齐）

3.3.4 EIS Load bin 并配置 EIS IQ 参数

```

AX_BOOL CBaseSensor::Open()
{
    LOG_M(SENSOR, "+++");
    ...
    nRet = AX_ISP_Open(m_nPipeID);
    if (0 != nRet) {
        LOG_M_E(SENSOR, "AX_ISP_Open failed");
        return AX_FALSE;
    }

    //EIS Load bin, call after AX_ISP_Open and before AX_VIN_Start

    if (gOptions.IsEISSupport()) {
        SENSOR_CONFIG_T tSensorCfg;
        CConfigParser().GetInstance()->GetCameraCfg(tSensorCfg,
        (SENSOR_ID_E)m_nSensorID);
        AX_CHAR *chEISBinPath = (AX_SNS_LINEAR_MODE == m_tSnsAttr.eSnsMode) ?
        tSensorCfg.aEISSdrBin : tSensorCfg.aEISHdrBin;

        if (access(chEISBinPath, F_OK) != 0) {
            LOG_M_E(SENSOR, "Sns[%d] EIS bin(%s) is not exist.", m_nSensorID,
            chEISBinPath);
            return AX_FALSE;
        }

        AX_S32 nRet = AX_ISP_LoadBinParams(m_nPipeID, (const AX_CHAR
        *)chEISBinPath);
        if (0 != nRet) {
            LOG_M_E(SENSOR, "AX_ISP_LoadBinParams (%s) failed, ret=0x%x.",
            chEISBinPath, nRet);
            return AX_FALSE;
        }
        else {
            LOG_M(SENSOR, "AX_ISP_LoadBinParams (%s) success.", chEISBinPath);
        }
    }

    nRet = AX_VIN_Start(m_nPipeID);

```

确保在 **AX_ISP_Open** 之后, **AX_VIN_Start** 之前
调用 **AX_ISP_LoadBinParams**

```

if (0 != nRet) {
    LOG_M_E(SENSOR, "AX_VIN_Start failed, ret=0x%x.", nRet);
    return AX_FALSE;
}

```

```

// Set EIS IQ Params

```

```

if (gOptions.IsEISSupport()) {
    if (!EnableEIS(gOptions.IsEnabledEIS())) {
        LOG_M_E(SENSOR, "Enable EIS failed!");
        return AX_FALSE;
    }
}

```

配置 EIS IQ 参数：开启
或关闭 EIS

可在运行过程中配置 IQ
参数实现动态开关 EIS

```

...

```

```

}

```

```

...

```

```

AX_BOOL CBaseSensor::EnableEIS(AX_BOOL bEnableEIS)

```

```

{

```

```

    AX_ISP_IQ_EIS_PARAM_T tEISParam;

```

```

    AX_S32 nRet = 0;

```

```

    nRet = AX_ISP_IQ_GetEisParam(m_nPipeID, &tEISParam);

```

```

    if (0 != nRet) {

```

```

        LOG_M_E(SENSOR, "AX_ISP_IQ_GetEisParam failed, ret=0x%x.", nRet);

```

```

        return AX_FALSE;

```

```

    }

```

```

    tEISParam.bEisEnable = bEnableEIS;

```

```

    tEISParam.nEisDelayNum = gOptions.GetEISDelayNum();

```

```

    tEISParam.nCropRatioW = gOptions.GetEISCropW();

```

```

    tEISParam.nCropRatioH = gOptions.GetEISCropH();

```

```

    nRet = AX_ISP_IQ_SetEisParam(m_nPipeID, &tEISParam);

```

```

    if (0 != nRet) {

```

```

        LOG_M_E(SENSOR, "AX_ISP_IQ_SetEisParam failed, ret=0x%x.", nRet);

```

```

        return AX_FALSE;

```

```

    }

```

```

    if (tEISParam.bEisEnable) {

```

```

        LOG_M(SENSOR, "EIS Model: %s", tEISParam.tEisNpuParam.szModelName);

```

```

        LOG_M(SENSOR, "Wbt Model: %s", tEISParam.tEisNpuParam.szWbtModelName);

```

EIS tuning 参数，详见“配置文件”章
节中 EIS 参数说明

```

}

return AX_TRUE;
}

```

3.3.5 设置 IVPS Engine

```

AX_BOOL CIVPSStage::Init()
{
    memset(&m_arrIvpsGrp[0], 0, sizeof(IVPS_GRP_T) * IVPS_GROUP_NUM);
    if (gOptions.IsEISSupport()) {
        // Make sure use AX_IVPS_ENGINE_GDC for EIS
        for (AX_U8 i = 0; i < IVPS_GROUP_NUM; i++) {
            g_tIvpsGroupConfig[i].arrChnEngineType[0] = AX_IVPS_ENGINE_GDC;
        }
        ...
    }

    for (AX_U32 i = 0; i < IVPS_GROUP_NUM; ++i) {
        for (AX_U8 nChn = 0; nChn < g_tIvpsGroupConfig[i].nGrpChnNum; ++nChn)
        {
            ...
            m_arrIvpsGrp[i].tPipelineAttr.tFilter[nChnFilter][0].eEngine
= g_tIvpsGroupConfig[i].arrChnEngineType[nChn];
            ...
        }
    }
    ...
}

```

IVPS eEngine 配置为 GDC

3.3.6 关闭 link 模式

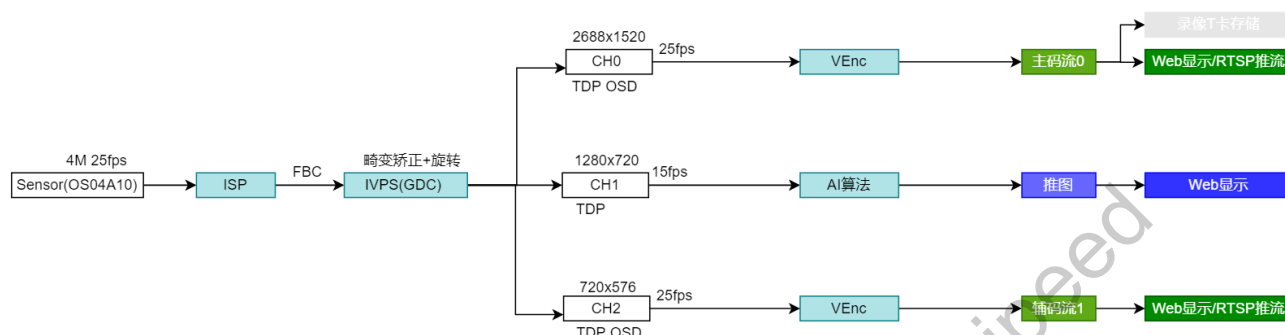
目前 EIS 不支持 link 模式

```

AX_BOOL COptionHelper::IsLinkMode(void) const
{
    return m_nEISSupport ? AX_FALSE : m_bLinkMode;
}

```

3.4 LDC 实现



ISP chn0 YUV 帧通过 IVPS GDC 做畸变矫正+旋转并分出三路流送到后续流程处理，详细功能实现可参考 **IPCLDCDemo**，以下是关键参数配置说明：

3.4.1 添加相机内参和畸变参数配置

对应 **Sensor json** 配置文件添加相机内参和畸变参数

```

"modules": [{
  "type": "camera",
  "instance": [{
    ...
    "distortion": [
      {
        "matrix": [2876663662, 0, 1401986697, 0, 2867276816,
789858593, 0, 0, 1000000],
        "coeff": [-465001, 297711, -1335, -1760, -191427, 0, 0, 0]
      }
    ]
  }
}]
  
```

matrix: 相机内参; coeff: 畸变参数

说明：

➤ 相机内参和畸变参数均为标定数据 **乘上 1000000 取整**

_SENSOR_CONFIG_T 增加对应成员变量

BaseSensor.h 文件中结构体 _SENSOR_CONFIG_T 增加 Dewarp 相关成员变量

```
typedef struct _SENSOR_CONFIG_T
```

```
{
    AX_U32          nFrameRate;
    AX_RUN_MODE_E   eRunMode;
    SNS_TYPE_E      eSensorType;
    AX_SNS_HDR_MODE_E eSensorMode;
    AX_CHAR          aNormalModeBin[SENSOR_BIN_PATH_LEN];
    AX_CHAR          aHotbalanceModeBin[SENSOR_BIN_PATH_LEN];
    AX_BOOL         bTuning;
    AX_U32          nTuningPort;
    CAMERA_CHAN_CFG_T arrChannels[MAX_ISP_CHANNEL_NUM];
    AX_IVPS_DEWARP_DISTORTION_FACTOR_S stDewarpFactors;
```

```
_SENSOR_CONFIG_T() {
```

```
...
```

```
}
```

```
} SENSOR_CONFIG_T, *SENSOR_CONFIG_PTR;
```

AX_IVPS_DEWARP_DISTORTION_FACTOR_S 定义

```
typedef struct
```

```
{
    AX_S64 nCameraMatrix[9];
    AX_S64 nDistortionCoeff[8];
} AX_IVPS_DEWARP_DISTORTION_FACTOR_S;
```

Json 解析相机内参和畸变参数

```
AX_BOOL CJsonCfgParser::GetCameraCfg(SENSOR_CONFIG_T &stOutCfg, SENSOR_ID_E
eSensorID)
```

```
{
```

```
...
```

```
...
```

获取相机内参和畸变参数配置

```
if (objSensor.end() != objSensor.find("distortion")) {
    picojson::array distortion =
objSensor["distortion"].get<picojson::array>();
    auto &it = distortion[0].get<picojson::object>();

    picojson::array &arrCMatrix = it["matrix"].get<picojson::array>();
    for (size_t j = 0; j < arrCMatrix.size(); j++) {
        stOutCfg.stDewarpFactors.nCameraMatrix[j] =
arrCMatrix[j].get<double>();
```



```

    }

    picojson::array &arrCoeff = it["coeff"].get<picojson::array>();
    for (size_t j = 0; j < arrCoeff.size(); j++) {
        stOutCfg.stDewarpFactors.nDistortionCoeff[j] =
arrCoeff[j].get<double>();
    }
}
...
}

```

3.4.2 GDC 参数配置

Distortion 模式

```

if (!CConfigParser().GetInstance()->GetCameraCfg(stSensorCfg,
E_SENSOR_ID_0)) {
    LOG_M_E(IVPS, "Load camera configure for sensor0 failed.");
    return AX_FALSE;
}

memset(&m_stGdcCfg, 0, sizeof(AX_IVPS_GDC_CFG_S));
m_stGdcCfg.bDewarpEnable = AX_TRUE;
m_stGdcCfg.bDewarpType = AX_IVPS_DEWARP_DISTORTION;
m_stGdcCfg.tDistortionFactor = stSensorCfg.stDewarpFactors;
m_stGdcCfg.bEnhanceMode = AX_TRUE;
m_stGdcCfg.eRotation = gOptions.GetRotation();
m_stGdcCfg.bMirror = gOptions.GetMirror();
m_stGdcCfg.bFlip = gOptions.GetFlip();

```

配置 GDC dewarp 类型和 Distortion 参数
(相机内参和畸变参数)

LDC 模式

```

if (!CConfigParser().GetInstance()->GetCameraCfg(stSensorCfg,
E_SENSOR_ID_0)) {
    LOG_M_E(IVPS, "Load camera configure for sensor0 failed.");
    return AX_FALSE;
}

AX_IVPS_DEWARP_LDC_FACTOR_S stLdcFactor;

```

```
stLdcFactor.bAspect = AX_FALSE;
```

```
stLdcFactor.nXRatio = 50; //水平缩放[0, 100]
```

```
stLdcFactor.nYRatio = 50; //垂直缩放[0, 100]
```

```
stLdcFactor.nDistortionRatio = 512; //畸变强度[0, 1023]
```

配置水平/垂直缩放比例及畸变强度

```
stLdcFactor.nCenterXOffset = m_stDewarpFactors.nCameraMatrix[2] / 1000000 - m_tChnAttr.tChnAttr[0].nWidth / 2;
```

```
stLdcFactor.nCenterYOffset = m_stDewarpFactors.nCameraMatrix[5] / 1000000 - m_tChnAttr.tChnAttr[0].nHeight / 2;
```

```
m_stGdcCfg.bDewarpEnable = AX_TRUE;
```

```
m_stGdcCfg.bDewarpType = AX_IVPS_DEWARP_LDC;
```

```
m_stGdcCfg.tDistortionFactor = stLdcFactor;
```

根据相机内参主点 X、Y 坐标计算主点偏移，固定值

```
m_stGdcCfg.bEnhanceMode = AX_TRUE;
```

```
m_stGdcCfg.eRotation = gOptions.GetRotation();
```

```
m_stGdcCfg.bMirror = gOptions.GetMirror();
```

```
m_stGdcCfg.bFlip = gOptions.GetFlip();
```

配置 GDC dewarp 类型和畸变参数

说明：

- **Distortion:** 依赖标定的相机内参和畸变参数，畸变矫正强度不可动态调整
- **LDC:** 依赖标定的光学中心偏移，畸变矫正强度可动态调整

IPCDemo 的配置文件位于 config/ipc_demo.conf，参数配置如下表所示：

4 配置文件:

表4-1 配置参数

参数名	参数范围	默认值	说明
LogTarget	[0 - 3]	1	应用日志目的地，默认输出到终端打印
LogLevel	[0 - 4]	2	应用日志级别
RTSPMaxFrmSize	-	2000000	单位：Byte。 RTSP Server 内部缓存大小
PrintFPS	0, 1	1	是否打印辅助信息，比如帧率等
EnableCoreDump	0, 1	1	是否生成 core dump 文件
RunInBackground	0, 1	0	是否在后台运行程序
Mp4Recorder	0, 1	0	是否开启视频流存储
Mp4Saved2SDCard	0, 1	0	Mp4 文件存储位置
Mp4SavedPath	-	/opt/mp4/	Mp4 文件存储路径

参数名	参数范围	默认值	说明
Mp4MaxFileNum	-	10	Mp4 文件最大个数
Mp4MaxFileSize	-	64	Mp4 单个文件最大 size
Rotation	[0 - 3]	0	配置旋转角度(0° /90° /180° /270°)
ActiveDetect	0, 1	1	打开检测功能
DetectModel	字符串	"/opt/etc/models"	检测模型的路径
DetectStreamType	"facehuman_video_all" 或者"hvcfp_video_all"	"hvcfp_video_all"	检测类型，只有两种可选，人脸或结构化
DetectConfigPath	字符串	"./config/hvcfp_config.json"	
EnableEIS	0, 1	0	应用启动是否开启 EIS
EISDelayNum	[1 - 4]	4	EIS 延迟帧数
EISCropW	[0 - 64]	8	宽方向裁剪比例 (单边)
EISCropH	[0 - 64]	8	高方向裁剪比例 (单边)
EISEffectComp	0, 1	0	是否开启 EIS 效果对比功能

🔍 说明：

1、当 RTSPMaxFrmSize 设置过小时，控制台会打印 “Exceeding max frame size: newFrameSize:XX > fMaxSize:XX” 警告信息，可以适当增加缓存大小避免图像数据丢失。

2、EIS 相关参数说明

1) EnableEIS: 开启生效的前提，对应 sensor 支持 EIS，同时 config 中对应 sensor 的 json 文件中 eis_enable 配置为 1，且 eis_sdr_bin/ eis_hdr_bin 均有效，即 EIS 使能

2) EISDelayNum: EIS 延迟帧数，帧数越大，内存消耗越大（需要缓存对应数量的 YUV 帧），延迟越大，理论上防抖效果越好

3) EISCropW: 原图宽方向裁剪比例（单边），左右分别裁剪 EISCropW/255

4) EISCropH: 原图高方向裁剪比例（单边），上下分别裁剪 EISCropH/255

5) EISEffectComp: 功能开启后，web 端开启两个 IPCDemo 窗口，两个窗口分别选择“主码流 0”（EIS），“子码流 1”（无 EIS）即可进行效果对比。生效前提，EIS 使能且 EnableEIS 置为 1

本章节包含：

5 新增 sensor

5.1 新增 sensor 流程图

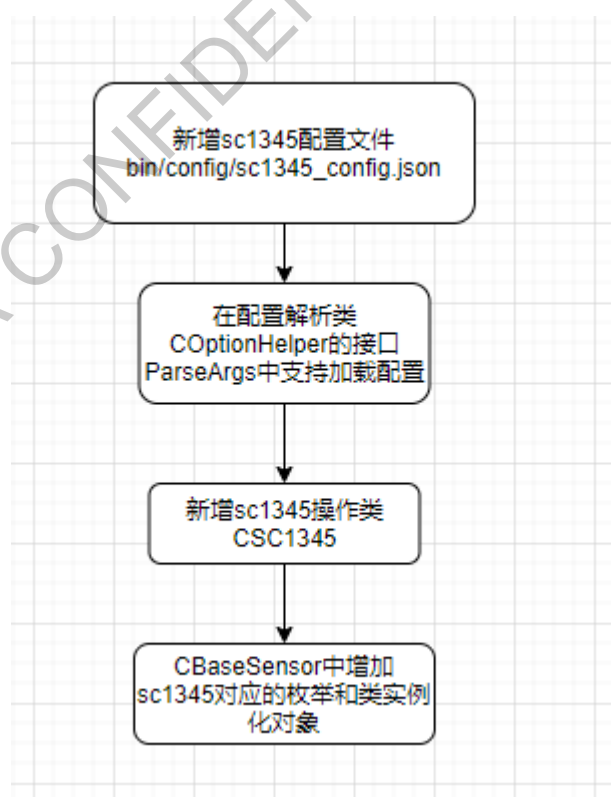
5.2 新增 sensor 配置文件

5.3 在 COptionHelper 的接口 ParseArgs 支持加载 sensor 配置

5.4 新增 Sensor 操作实例类

5.5 在 CBaseSensor 中实例化 sensor 操作类

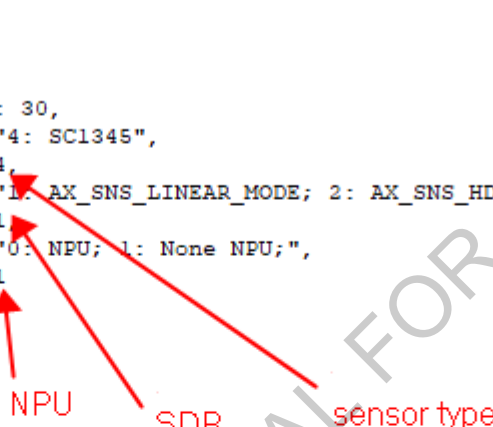
5.1 新增 sensor 流程图



5.2 新增 sensor 配置文件

在 bin/config 目录下新增 sensor 配置文件。此处以 sc1345 为例，新增 sc1345_config.json 文件，内容可以从 04a10 的 sensor 配置文件 os04a10_config.json 复制过来，修改要点：

```
"modules": [{
  "type": "camera",
  "instance": [{
    "id": 0,
    "frame_rate": 30,
    "sns_type": "4: SC1345",
    "sns_type": 4,
    "sns_mode": "1: AX_SNS_LINEAR_MODE; 2: AX_SNS_HDR_2X_MODE;",
    "sns_mode": 1,
    "run_mode": "0: NPU; 1: None NPU;",
    "run_mode": 1
  ]
},
{
  "type": "venc",
  "encoder": "h264",
  "instance": [{
    "id": 0,
    "rc": {
      "type": "CBR",
      "min_qp": 0,
      "max_qp": 51,
      "min_iqp": 0,
      "max_iqp": 51,
      "intra_qp_delta": -2
    },
    "bitrate": 2048,
    "fps": 30,
    "gop": 30
  ]
},
{
  "id": 1,
```



- Sensor type 要和 CBaseSensor 新增的 Sensor 枚举值一致，一般是顺序新增
- Sns Mode 一般使用 SDR 模式，当然可以根据需要定义为 HDR 模式
- Run Mode 一般都为 0，即启用 NPU 模式

5.3 在 COptionHelper 的接口 ParseArgs 指定 SensorID

```
if (!m_strJsonCfgFile.empty()) {
    if (string::npos != m_strJsonCfgFile.find("os04a10")) {
        m_nSensorID = 0;
    } else if (string::npos != m_strJsonCfgFile.find("imx334")) {
        m_nSensorID = 1;
    } else if (string::npos != m_strJsonCfgFile.find("gc4653")) {
        m_nSensorID = 2;
    } else if (string::npos != m_strJsonCfgFile.find("os08a20")) {
        m_nSensorID = 3;
    } else if (string::npos != m_strJsonCfgFile.find("sc1345")) {
        m_nSensorID = 4;
    } else {
        printf("[ERROR] Unkonown json configure file: %s!", m_strJsonCf
        return AX_FALSE;
    }
}

/* Log target */
m_nLogTarget = atoi((char *)argv[2]);
```

5.4 新增 Sensor 操作实例类

此处以 sc1345 为例。新增 CSC1345 类，从 CBaseSensor 继承。

```
class CSC1345 : public CBaseSensor
{
public:
    ...CSC1345(SENSOR_CONFIG_T tSensorConfig);
    ...virtual ~CSC1345(AX_VOID);

protected:
    ...virtual AX_VOID InitSnsAttr();
    ...virtual AX_VOID InitDevAttr();
    ...virtual AX_VOID InitPipeAttr();
    ...virtual AX_VOID InitMipiRxAttr();
    ...virtual AX_VOID InitChnAttr();
    ...virtual AX_S32 RegisterSensor(AX_U8 nPipe);
    ...virtual AX_S32 UnRegisterSensor(AX_U8 nPipe);
    ...virtual AX_S32 RegisterAeAlgLib(AX_U8 nPipe);
    ...virtual AX_S32 UnRegisterAeAlgLib(AX_U8 nPipe);
    ...virtual AX_S32 RegisterAwbAlgLib(AX_U8 nPipe);
    ...virtual AX_S32 UnRegisterAwbAlgLib(AX_U8 nPipe);
};
```


类实现参考 Demo，主要修改内容如下：

```
AX_VOID CSC1345::InitSnsAttr()
{
    ..../*Referenced-by-AX_VIN_SetSnsAttr-*/
    m_tSnsAttr.nWidth = 1280;
    m_tSnsAttr.nHeight = 720;
    m_tSnsAttr.nFrameRate = m_tSnsInfo.nFrameRate;
    m_tSnsAttr.eSnsMode = m_tSnsInfo.eSensorMode;
    if (AX_SNS_HDR_2X_MODE == m_tSnsAttr.eSnsMode) {
        m_tSnsAttr.eRawType = AX_RT_RAW10;
    } else {
        m_tSnsAttr.eRawType = AX_RT_RAW10;
    }

    m_tSnsAttr.eSnsHcgLcg = AX_LCG_NOTSUPPORT_MODE;
    m_tSnsAttr.eBayerPattern = AX_BP_RGGB;
    m_tSnsAttr.bTestPatternenable = AX_FALSE;
    m_tSnsAttr.eMasterSlaveSel = AX_SNS_MASTER;
    m_tSnsClkAttr.nSnsClkIdx = 0;
    m_tSnsClkAttr.eSnsClkRate = AX_SNS_CLK_24M;
}
```

```
..../*Referenced-by-AX_VIN_SetDevAttr-*/
m_tDevAttr.eSnsType = AX_SNS_TYPE_MIPI;
m_tDevAttr.tDevIngRgn.nStartX = 0;
m_tDevAttr.tDevIngRgn.nStartY = 0;
m_tDevAttr.tDevIngRgn.nWidth = 1280;
m_tDevAttr.tDevIngRgn.nHeight = 720;
m_tDevAttr.bDoISplit = AX_FALSE;
m_tDevAttr.bHwIrcon = AX_FALSE;
m_tDevAttr.eBayerPattern = AX_BP_RGGB;
m_tDevAttr.eSkipFrame = AX_SNS_SKIP_FRAME_NO_SKIP;
m_tDevAttr.eSnsGainMode = AX_SNS_GAIN_MODE_HCG;
m_tDevAttr.eSnsMode = m_tSnsInfo.eSensorMode;
if (AX_SNS_HDR_2X_MODE == m_tDevAttr.eSnsMode) {
    m_tDevAttr.ePixelFmt = AX_FORMAT_BAYER_RAW_10BPP;
} else {
    m_tDevAttr.ePixelFmt = AX_FORMAT_BAYER_RAW_10BPP;
}
..../*m_tDevAttr.eSnsOutputMode = AX_SNS_NORMAL;
```

```
AX_VOID CSC1345::InitPipeAttr()
{
    ..../*Referenced-by-AX_VIN_SetPipeAttr-*/
    m_tPipeAttr.nWidth = 1280;
    m_tPipeAttr.nHeight = 720;
    m_tPipeAttr.eBayerPattern = AX_BP_RGGB;
    m_tPipeAttr.eSnsMode = m_tSnsInfo.eSensorMode;
    if (AX_SNS_HDR_2X_MODE == m_tPipeAttr.eSnsMode) {
        m_tPipeAttr.ePixelFmt = AX_FORMAT_BAYER_RAW_10BPP;
    } else {
        m_tPipeAttr.ePixelFmt = AX_FORMAT_BAYER_RAW_10BPP;
    }
    if (AX_SNS_HDR_2X_MODE == m_tPipeAttr.eSnsMode) {
        m_tPipeAttr.ePipeDataSrc = AX_PIPE_SOURCE_DEV_OFFLINE;
    } else {
        m_tPipeAttr.ePipeDataSrc = AX_PIPE_SOURCE_DEV_ONLINE;
    }
    m_tPipeAttr.eDataFlowType = AX_DATAFLOW_TYPE_NORMAL;
    m_tPipeAttr.eDevSource = AX_DEV_SOURCE_SNS_ID;
    m_tPipeAttr.ePreOutput = AX_PRE_OUTPUT_FULL_MAIN;
}
```

```
AX_VOID CSC1345::InitChnAttr()
{
    ..../*Referenced-by-AX_VIN_SetChnAttr-*/
    m_tChnAttr.tChnAttr[0].nWidth = 1280;
    m_tChnAttr.tChnAttr[0].nHeight = 720;
    m_tChnAttr.tChnAttr[0].eImgFormat = AX_YUV420_SEMIPLANAR;
    m_tChnAttr.tChnAttr[0].bEnable = AX_TRUE;
    m_tChnAttr.tChnAttr[0].nWidthStride = 1280;
    m_tChnAttr.tChnAttr[0].nDepth = YUV_SC1345_DEPTH;

    m_tChnAttr.tChnAttr[1].nWidth = 640;
    m_tChnAttr.tChnAttr[1].nHeight = 320;
    m_tChnAttr.tChnAttr[1].eImgFormat = AX_YUV420_SEMIPLANAR;
    m_tChnAttr.tChnAttr[1].bEnable = AX_TRUE;
    m_tChnAttr.tChnAttr[1].nWidthStride = 640;
    m_tChnAttr.tChnAttr[1].nDepth = YUV_SC1345_DEPTH;

    m_tChnAttr.tChnAttr[2].nWidth = 320;
    m_tChnAttr.tChnAttr[2].nHeight = 240;
    m_tChnAttr.tChnAttr[2].eImgFormat = AX_YUV420_SEMIPLANAR;
    m_tChnAttr.tChnAttr[2].bEnable = AX_TRUE;
    m_tChnAttr.tChnAttr[2].nWidthStride = 320;
    m_tChnAttr.tChnAttr[2].nDepth = YUV_SC1345_DEPTH;
}
```

```

AX_S32 CSC1345::RegisterSensor(AX_U8 nPipe)
{
    ....AX_S32 nRet == 0;

    ....m_pSnsLib == dlopen("libsns_sc1345.so", RTLD_LAZY);
    ....if (NULL == m_pSnsLib) {
    ....    ....LOG_M_E(SC1345, "Load sensor lib failed!");
    ....    ....return -1;
    ....}

    ....m_pSnsObj == (AX_SENSOR_REGISTER_FUNC_T *)dlsym(m_pSnsLib, "gSnsSc1345Obj");
    ....if (NULL == m_pSnsObj) {
    ....    ....LOG_M_E(SC1345, "AX_VIN_Get_Sensor_Object_Failed!");
    ....    ....return -1;
    ....}

    ....nRet == AX_VIN_RegisterSensor(nPipe, m_pSnsObj); /* pipe 0 */
    ....if (nRet) {
    ....    ....LOG_M_E(SC1345, "AX_ISP_Register_Sensor_Failed, ret=0x%x.", nRet);
    ....    ....return nRet;
    ....}

    ....AX_SNS_COMMBUS_T tSnsBusInfo == {0};
    ....memset(&tSnsBusInfo, 0, sizeof(AX_SNS_COMMBUS_T));
    ....tSnsBusInfo.I2cDev == GetI2cDevNode(nPipe);
    ....if (NULL != m_pSnsObj->pfn_sensor_set_bus_info) {
    ....    ....nRet == m_pSnsObj->pfn_sensor_set_bus_info(nPipe, tSnsBusInfo); /* pipe 0 */
    ....    ....if (0 != nRet) {
    ....        ....LOG_M_E(SC1345, "Sensor set bus info failed, ret=0x%x.", nRet);
    ....    ....}
    ....}
}

```

值的选用 tuning 工具给出的值。

5.5 在 CBaseSensor 中实例化 sensor 操作类

5.5.1 在 CBaseSensor 头文件中增加新的 sensor 枚举支持

IPCDemo/source/sensor/BaseSensor.h File 4 of 8 Prev Up Next

SHOW BLAME Diff view: [icon] [icon] [icon]

File

@@ +10↑ - Show 16 common lines - +10↓

17 E_SENSOR_ID_0 = 0,
18 E_SENSOR_ID_1 = 1,
19 E_SENSOR_ID_MAX
20 } SENSOR_ID_E;
21
22 typedef enum {
23 E_SNS_TYPE_OS04A10 = 0,
24 E_SNS_TYPE_IMX334,
25 E_SNS_TYPE_GC4653,
26 E_SNS_TYPE_OS08A20,
27 E_SNS_TYPE_SC1345,
28 E_SNS_TYPE_MAX,
29 } SNS_TYPE_E;
30
31 enum COLOR_FORMAT_E
32 {
33 E_COLOR_FMT_YUV422I_UYVY = 0,
34 E_COLOR_FMT_YUV422I_YUYV,
35 E_COLOR_FMT_YUV420SP_NV12,
36 E_COLOR_FMT_YUV420SP_NV21,

5.5.2 在接口 CBaseSensor::NewInstance 中增加 sensor 实例化

```
CBaseSensor *CBaseSensor::NewInstance(AX_U8 nSensorIndex)
{
    if (nSensorIndex >= E_SENSOR_ID_MAX) {
        LOG_M_E(SENSOR, "Sensor index out of range.");
        return nullptr;
    }

    SENSOR_CONFIG_T tSensorCfg;
    if (!CConfigParser().GetInstance()->GetCameraCfg(tSensorCfg, (SENSOR_ID_E)nSensorIndex)) {
        LOG_M_W(SENSOR, "Load camera configure for sensor %d failed.", nSensorIndex);
        return nullptr;
    }

    switch (tSensorCfg.eSensorType) {
        case E_SNS_TYPE_OS04A10: {
            return new COS04a10(tSensorCfg);
        }
        case E_SNS_TYPE_IMX334: {
            return new CIMX334(tSensorCfg);
        }
        case E_SNS_TYPE_GC4653: {
            return new CGC4653(tSensorCfg);
        }
        case E_SNS_TYPE_OS08A20: {
            return new COS08a20(tSensorCfg);
        }
        case E_SNS_TYPE_SC1345: {
            return new CSC1345(tSensorCfg);
        }
    }
}
```

5.5.3 在接口 CBaseSensor::GetNpuAttr 中设置 sensor 使用的 NPU 属性

```
AX_NPU_SDK_EX_ATTR_T CBaseSensor::GetNpuAttr(SNS_TYPE_E eSnsType, AX_SNS_HDR_MODE_E eSnsHdrMode)
{
    AX_NPU_SDK_EX_ATTR_T npuAttr;

    switch (eSnsType) {
    case E_SNS_TYPE_OS04A10:
        if (eSnsHdrMode == AX_SNS_HDR_3X_MODE) {
            npuAttr.eHardMode = AX_NPU_VIRTUAL_DISABLE;
        } else {
            npuAttr.eHardMode = AX_NPU_VIRTUAL_1_1;
        }
        break;
    case E_SNS_TYPE_IMX334:
        npuAttr.eHardMode = AX_NPU_VIRTUAL_1_1;
        break;
    case E_SNS_TYPE_GC4653:
        npuAttr.eHardMode = AX_NPU_VIRTUAL_1_1;
        break;
    case E_SNS_TYPE_OS08A20:
        npuAttr.eHardMode = AX_NPU_VIRTUAL_1_1;
        break;
    case E_SNS_TYPE_SC1345:
        npuAttr.eHardMode = AX_NPU_VIRTUAL_DISABLE;
        break;
    default:
        npuAttr.eHardMode = AX_NPU_VIRTUAL_DISABLE;
        break;
    }
}
```

6 常见问题

6.1 Sensor 配置不匹配

```
i2c_read: Failed to read reg: Remote I/O error.!!  
i2c_write: Failed to write reg: Remote I/O error.!!  
i2c_read: Failed to read reg: Remote I/O error.!!  
i2c_read: Failed to read reg: Remote I/O error.!!
```

图6-1 Sensor 不匹配

【产生原因】

通常原因为执行 `./run.sh config/XX_config.json` 命令启动时，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 配置文件即可。

6.2 Web 切换码流失败



图6-2 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，IPCDemo 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 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。