libsynexens4 SDK 多机使用说明 v1.7

修订历史版本				
日期	SDK 版本	文档版本	描述	作者
202212114	v4.0.0.0	v1.0	初始版本	YSY
20230424	v4.0.1.0	v1.1	release 版本	YSY
20230713	v4.0.2.0	v1.2	新增支持设备	YSY
20230815	v4.0.3.0	v1.3	新增接口	YSY
20230906	v4.1.0.0	v1.4	更新调用流程，滤波说明	YSY
20240206	v4.1.2.0	v1.5	新增接口	YSY
20240229	v4.1.3.0	v1.6	更新滤波	YSY
20240604	v4.2.1.0	v1.7	新增接口	YSY

更新内容：

1. 新增硬件滤波接口。
2. 调整滤波调用逻辑，取消默认滤波，SDK 完全控制滤波开关。
3. 修复已知 BUG

目录

1. 概述

支持设备：cs20 单频 cs20 双频 cs30 单频 cs30 双频 cs20-p cs40

支持系统：windows ubuntu20.04 armv7 armv8

2. 环境配置

2.1. Ubuntu 环境配置（以 Cmake 为例）

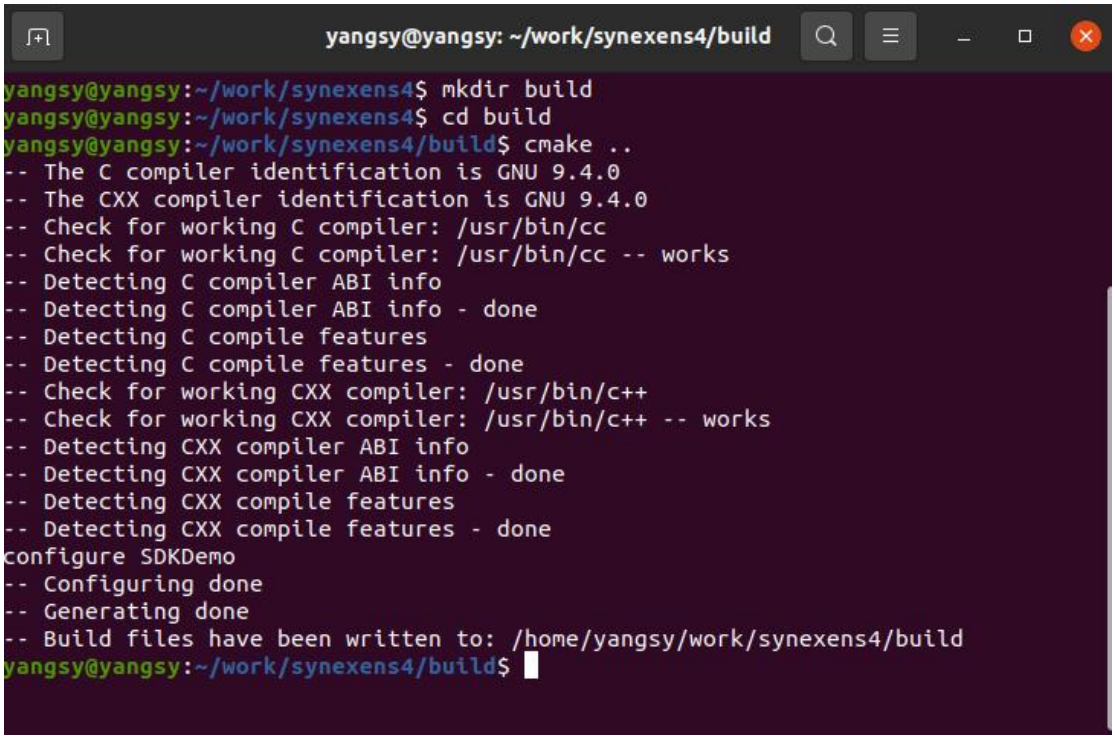
2.1.1. 安装依赖

```
sudo apt install libudev-dev

sudo apt install zlib1g-dev
```

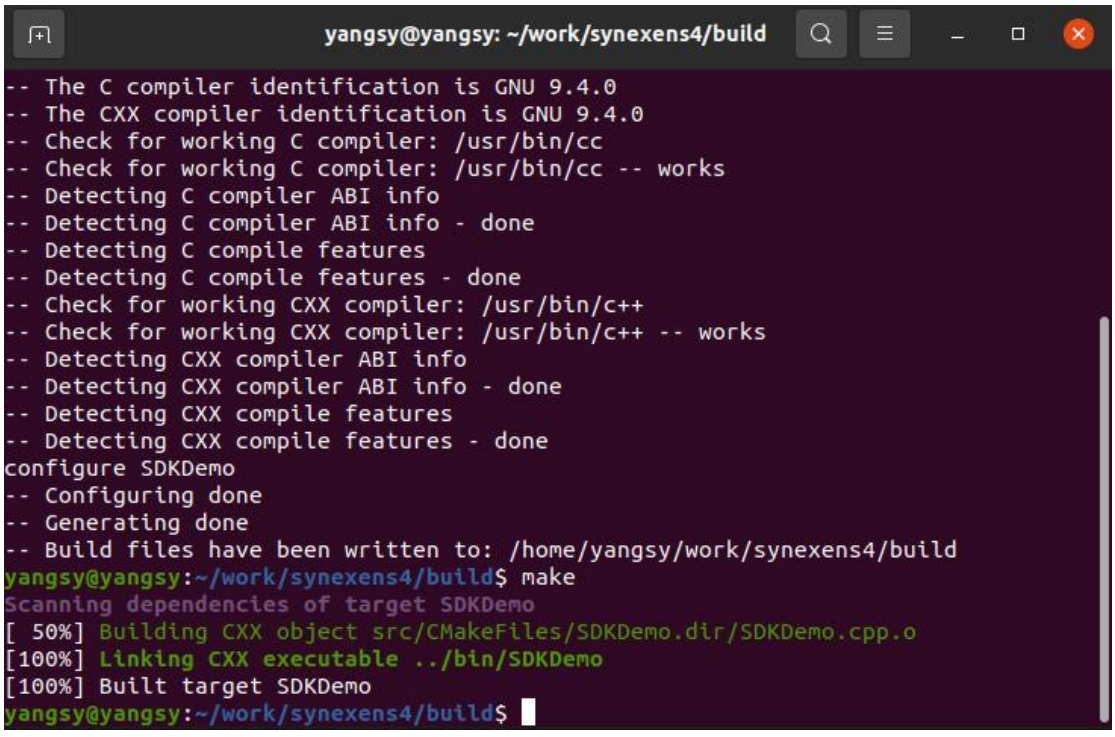
2.1.2. 编写 CmakeLists.txt 需要熟悉 CMake

```
set(TARGET_NAME SDKDemo)
message("configure ${TARGET_NAME}")

# +++++++ setting +++++++
set(CMAKE_CXX_FLAGS "${CMAKE_CXX_FLAGS} -std=c++11 -pthread")

# ###############
# ### opencv ####
# ###############
set(OpenCV440_INCLUDE_DIR "..//thirdpart/opencv/include")
set(OpenCV440_LIBS_DIR "../thirdpart/opencv/lib")
include_directories(${OpenCV440_INCLUDE_DIR})
link_directories(${OpenCV440_LIBS_DIR})

if(WIN32)
elseif(UNIX)
    set(OpenCV440_LIBS
        opencv_imgproc
        opencv_imgcodecs
        opencv_highgui
        opencv_core
        opencv_videoio
        opencv_calib3d
    )
endif()

# ############
# ### SDK ####
# ############
set(SDK_INCLUDE_DIR "../include")
set(SDK_LIB_DIR "../lib")
include_directories(${SDK_INCLUDE_DIR})
link_directories(${SDK_LIB_DIR})

if(WIN32)
    set(APP_PREFIX .exe)
    set(SDK_LIB SynexensSDK)
elseif(UNIX)
    set(APP_PREFIX)
    set(SDK_LIB SynexensSDK)
endif()

add_executable(${TARGET_NAME} SDKDemo.cpp)

target_link_libraries(${TARGET_NAME} ${OpenCV440_LIBS} ${SDK_LIB} udev dl z)
```

2.1.3. 创建项目编译文件



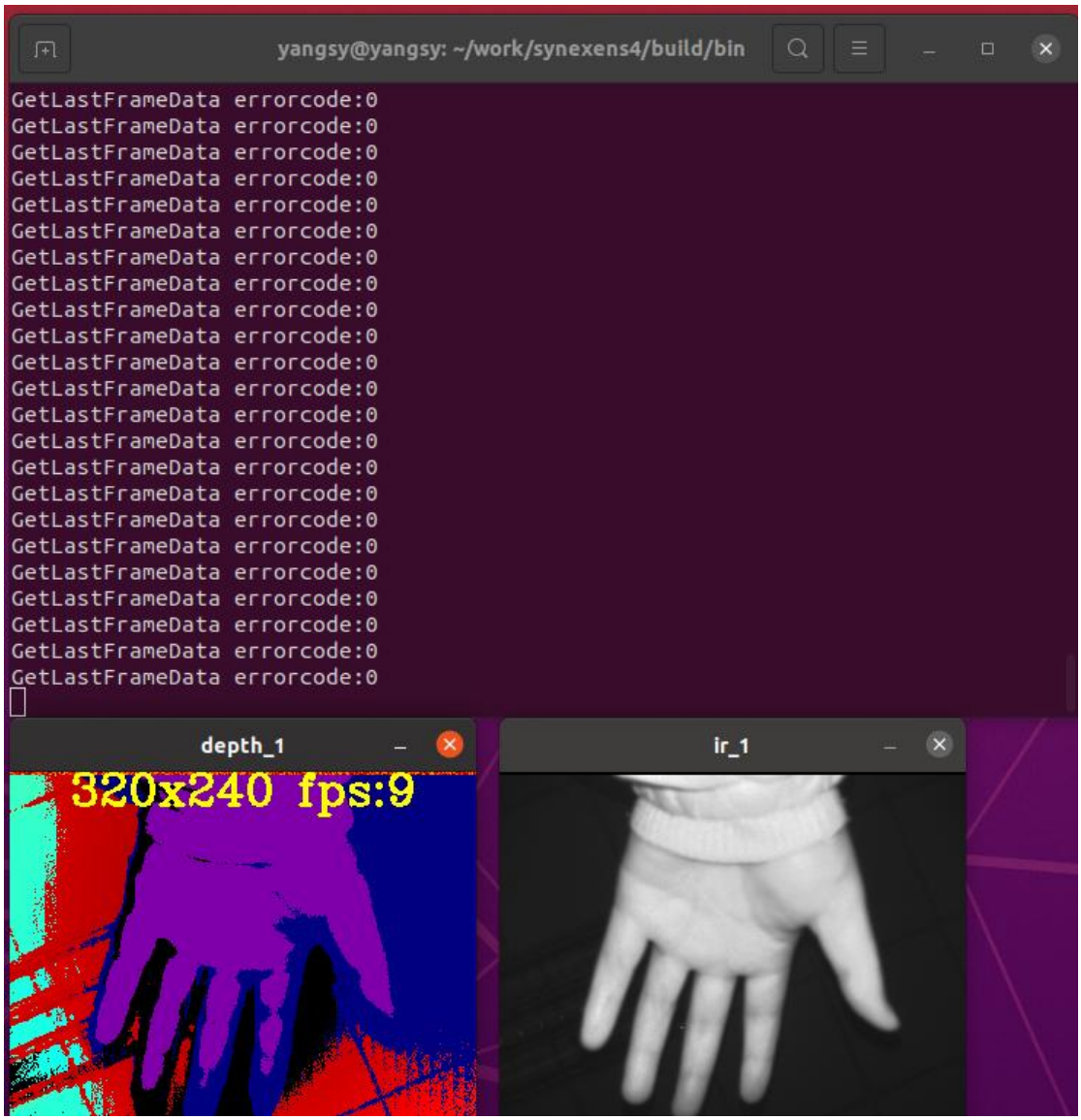
2.1.4. make 编译



2.1.5. 执行可执行文件测试效果

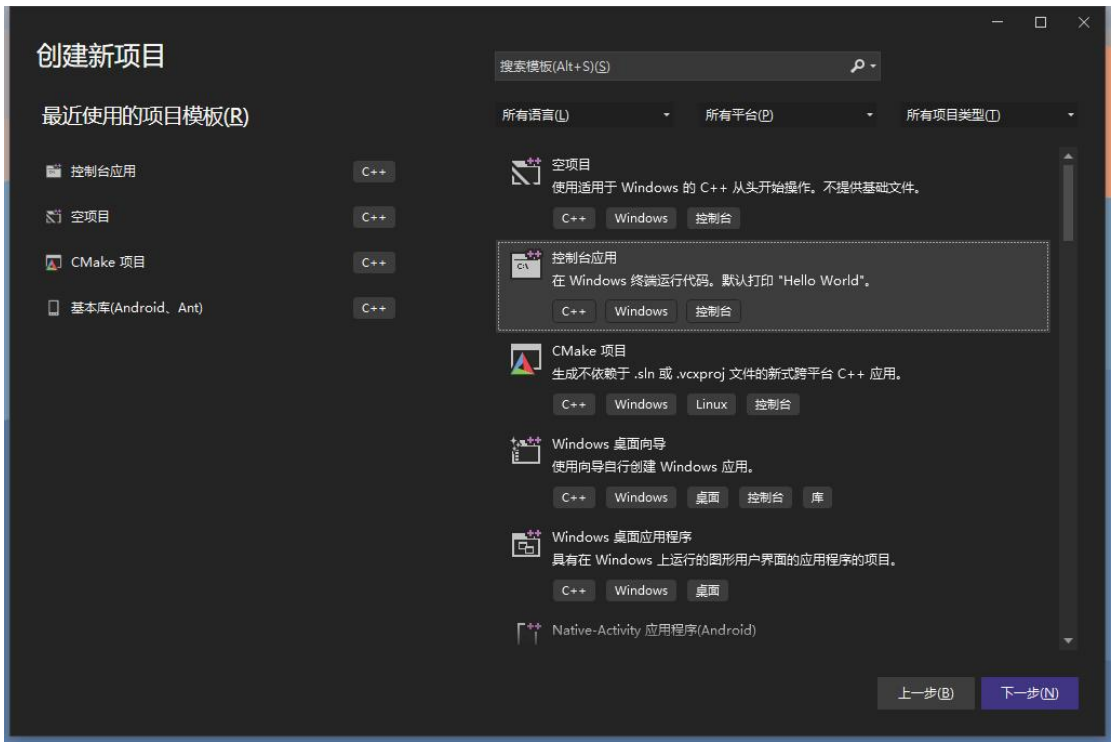
执行程序前需配置好 LD_LIBRARY_PATH，以便找到程序依赖的库文件，示例编写了 run.sh 脚本以方便执行程序。

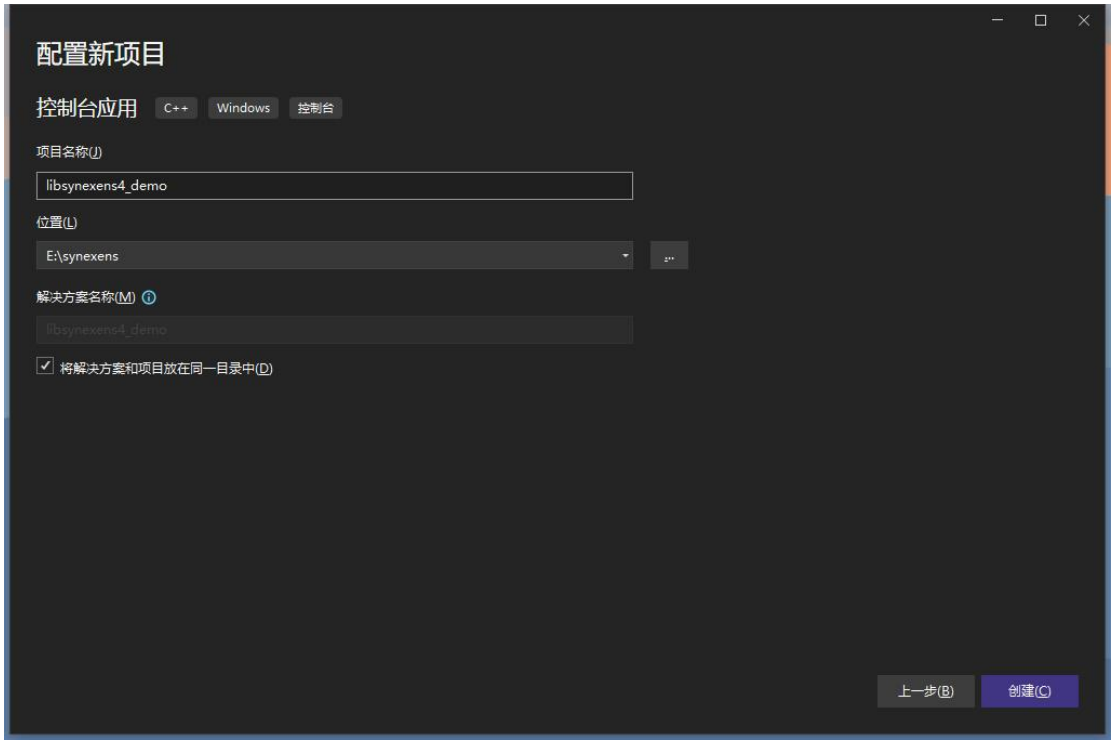
```
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:`pwd`
```



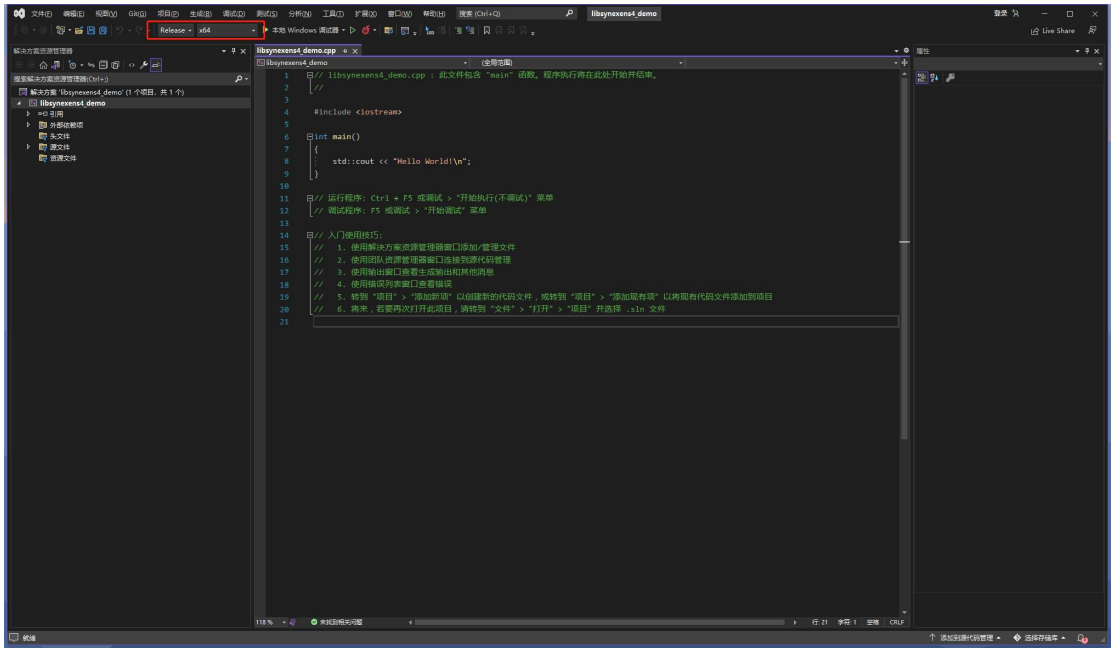
2.2. Windows 环境配置（以 vs2022 为例）

2.2.1. 创建 VS 工程

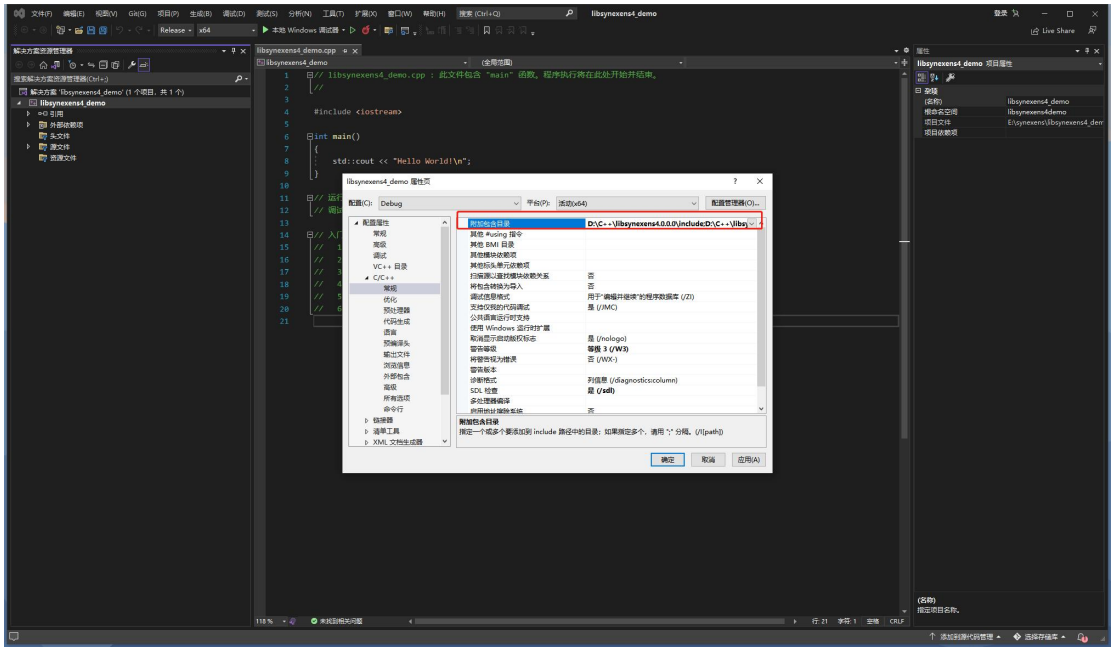


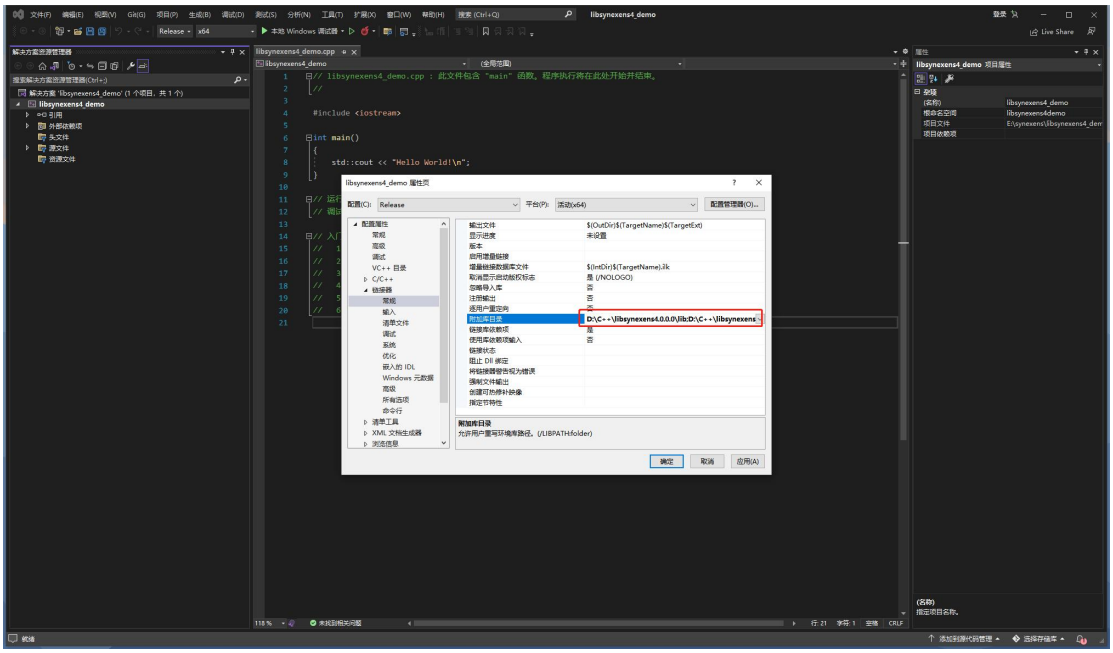


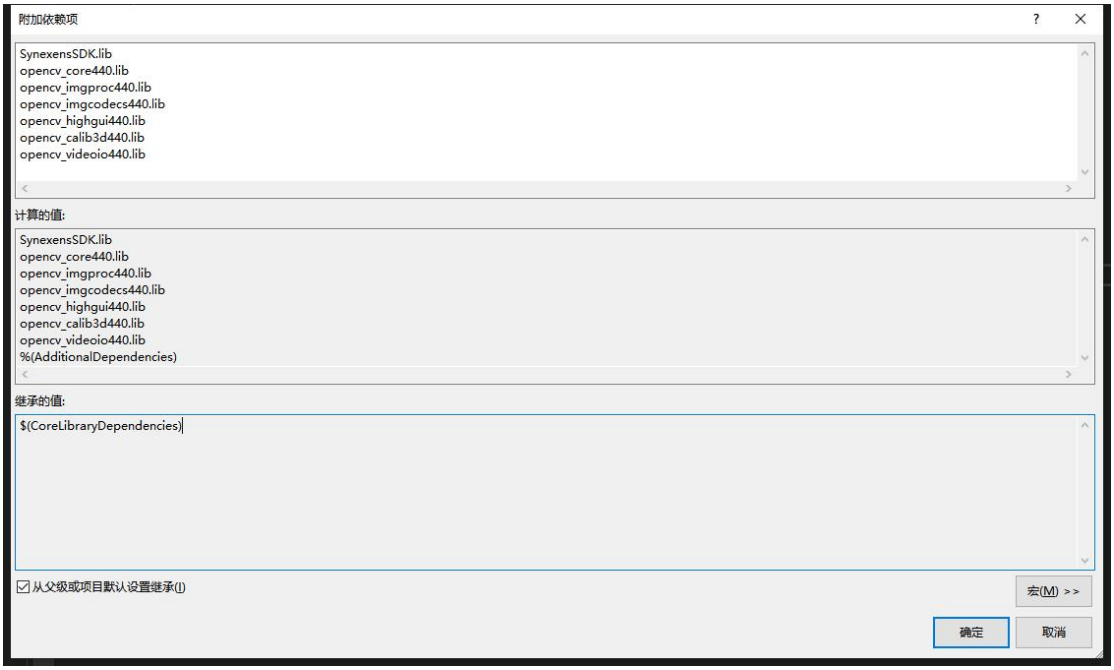
2.2.2. 选择与 SDK 对应的解决方案以及平台

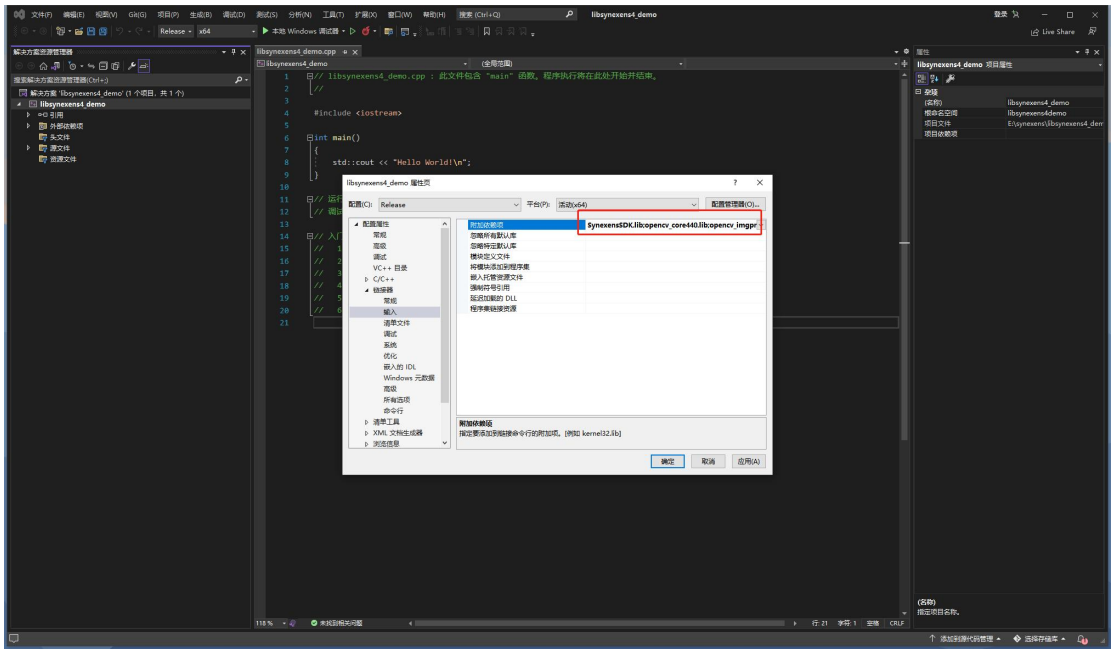


2.2.3. 在项目属性中配置 sdk 的头文件路径、库路径






附加依赖项
SynexensSDK.lib
opencv_core440.lib
opencv_imgproc440.lib
opencv_imgcodecs440.lib
opencv_highgui440.lib
opencv_calib3d440.lib
opencv_videoio440.lib
计算的值:
SynexensSDK.lib
opencv_core440.lib
opencv_imgproc440.lib
opencv_imgcodecs440.lib
opencv_highgui440.lib
opencv_calib3d440.lib
opencv_videoio440.lib
%(AdditionalDependencies)
继承的值:
$(CoreLibraryDependencies)
从父级或项目默认设置继承(I)
宏(M) >>
确定
取消

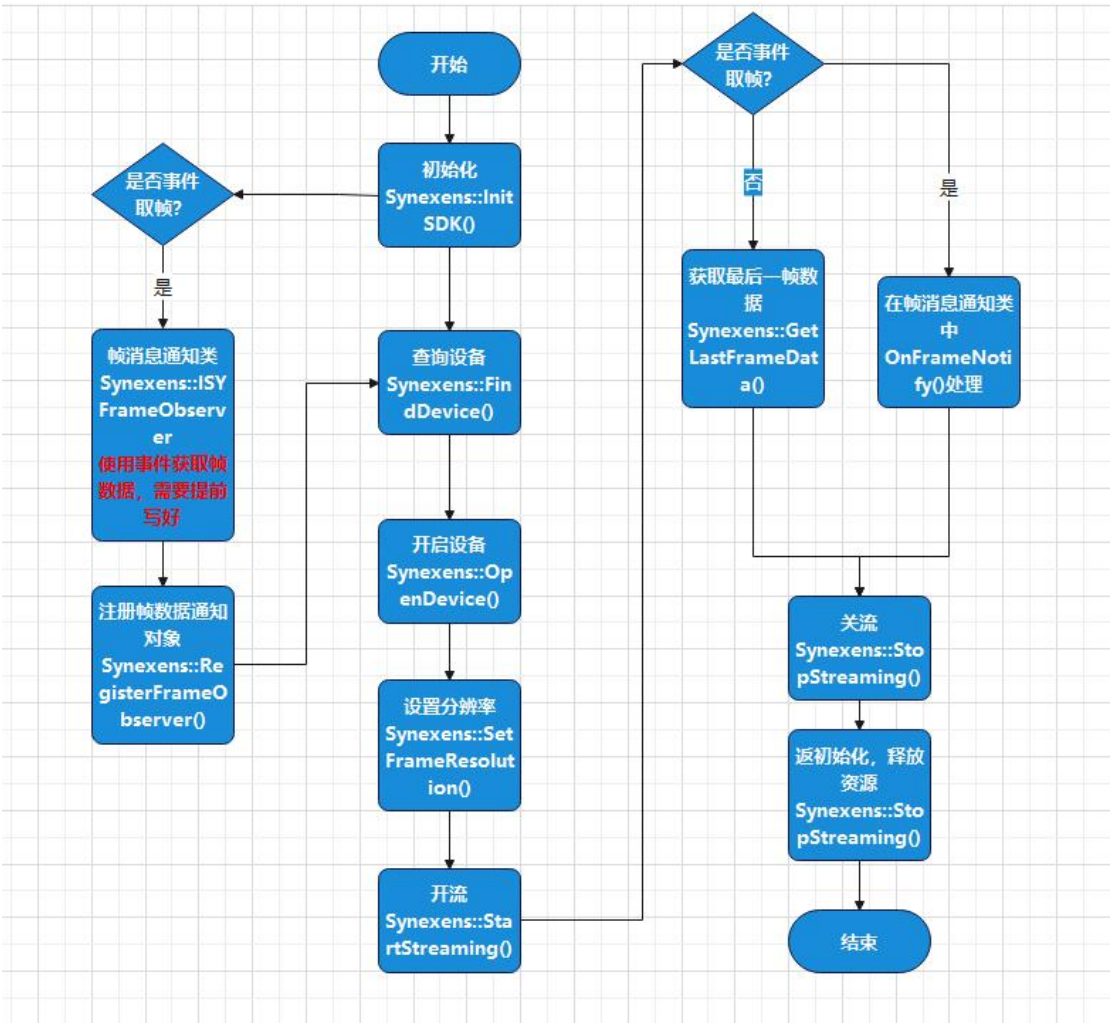


运行 demo 需要 opencv 依赖库，自行开发不需要依赖 opencv

2.2.4. 完成配置后可进入工程进行开发，如果需要运行 demo，只需将 demo 代码复制运行即可

注意：运行 demo 需要自行配置 include 路径，运行时缺少的 dll 文件需要自行拷贝到程序运行目录

2.3. SDK 必须调用流程



3. API 概述

3.1. 全局接口

3.1.1. GetSDKVersion

描述：获取 SDK 版本号

语法：

GetSDKVersion(int& nLength, char* pstrSDKVersion = nullptr);

参数：

参数名称	描述	in/out
nLenght	字符长度	in/out
pstrSDKVersion	SDK 版本号字符串指针	in/out

3.1.2. InitSDK

描述：初始化 SDK

语法：

InitSDK();

3.1.3. UnInitSDK

描述：反初始化 SDK，释放资源

语法：

UnInitSDK();

3.1.4. RegisterErrorObserver

描述：注册错误消息通知对象指针

语法：

RegisterErrorObserver(ISYErrorObserver* pObserver);

参数：

参数名称	描述	in/out
pObserver	错误消息通知对象指针	in

3.1.5. RegisterEventObserver

描述：注册事件通知对象指针

语法：

RegisterEventObserver(ISYEventObserver* pObserver);

参数：

参数名称	描述	in/out
pObserver	事件通知对象指针	in

3.1.6. RegisterFrameObserver

描述：注册数据帧通知对象指针

语法：

RegisterFrameObserver(ISYFrameObserver* pObserver);

参数：

参数名称	描述	in/out
pObserver	数据帧通知对象指针	in

3.1.7. UnRegisterErrorObserver

描述：注销错误消息通知对象指针

语法：

UnRegisterErrorObserver(ISYErrorObserver* pObserver);

参数：

参数名称	描述	in/out
pObserver	错误消息通知对象指针	in

3.1.8. UnRegisterEventObserver

描述：注销事件通知对象指针

语法：

UnRegisterEventObserver(ISYEventObserver* pObserver);

参数：

参数名称	描述	in/out
pObserver	事件通知对象指针	in

3.1.9. UnRegisterFrameObserver

描述：注销数据帧通知对象指针

语法：

UnRegisterFrameObserver(ISYFrameObserver* pObserver);

参数：

参数名称	描述	in/out
pObserver	数据帧通知对象指针	in

3.1.10. FindDevice

描述：查找设备

语法：

FindDevice(int& nCount, SYDeviceInfo* pDevice = nullptr);

参数：

参数名称	描述	in/out
nCount	设备数量	in/out
pDevice	设备信息，由外部分配内存，pDevice 传入 nullptr 时仅获取 nCount	in/out

3.1.11. OpenDevice

描述：打开设备

语法：

OpenDevice(const SYDeviceInfo& deviceInfo);

参数：

参数名称	描述	in/out
deviceInfo	设备信息	in

3.1.12. CloseDevice

描述：关闭设备

语法：

CloseDevice(unsigned int nDeviceID);

参数：

参数名称	描述	in/out
nDeviceID	设备 id	in

3.1.13. QueryDeviceSupportFrameType

描述：查询设备支持数据帧类型

语法：

QueryDeviceSupportFrameType(unsigned int nDeviceID, int& nCount, SYSupportType * pSupportType = nullptr);

参数：

参数名称	描述	in/out
nDeviceID	设备 id	in
nCount	支持的数据帧类型数量,pSupportType 为空时仅用作返回数量，否则用来校验 pSupportType 内存分配数量是否匹配	in/out
pSupportType	支持的数据帧类型，由外部分配内存，pFrameType 传入 nullptr 时仅获取 nCount	in/out

3.1.14. QueryDeviceSupportResolution

描述：查询设备支持的帧分辨率

语法：

QueryDeviceSupportResolution(unsigned int nDeviceID, SYSupportType supportType, int& nCount, SYResolution* pResolution = nullptr);

参数：

参数名称	描述	in/out

nDeviceID	设备 id	in
supportType	帧类型	in
nCount	支持的分辨率数量,pResolution 为空时仅用作返回数量，否则用来校验 pResolution 内存分配数量是否匹配	in/out
pResolution	支持的分辨率类型，由外部分配内存，pResolution 传入 nullptr 时仅获取 nCount	in/out

3.1.15. GetCurrentStreamType

描述：获取当前流类型

语法：

GetCurrentStreamType(unsigned int nDeviceID);

参数：

参数名称	描述	in/out
nDeviceID	设备 ID	in

3.1.16. StartStreaming

描述：启动数据流

语法：

StartStreaming(unsigned int nDeviceID, SYStreamType streamType);

参数：

参数名称	描述	in/out
nDeviceID	设备 id	in
streamType	数据流类型	in

3.1.17. StopStreaming

描述：关闭数据流

语法：

StopStreaming(unsigned int nDeviceID);

参数：

参数名称	描述	in/out
nDeviceID	设备 id	in

3.1.18. ChangeStreaming

描述：切换数据流

语法：

ChangeStreaming(unsigned int nDeviceID, SYStreamType streamType);

参数：

参数名称	描述	in/out

nDeviceID	设备 id	in
streamType	数据流类型	in

3.1.19. SetFrameResolution

描述：设置分辨率（如果已启动数据流，内部会执行关流->设置分辨率->重新开流的操作流程）

语法：

SetFrameResolution(unsigned int nDeviceID, SYFrameType frameType, SYResolution resolution);

参数：

参数名称	描述	in/out
nDeviceID	设备 id	in
frameType	帧类型	in
resolution	帧分辨率	in

3.1.20. GetFrameResolution

描述：获取设备帧分辨率

语法：

GetFrameResolution(unsigned int nDeviceID, SYFrameType frameType, SYResolution& resolution);

参数：

参数名称	描述	in/out
nDeviceID	设备 id	in
frameType	帧类型	in
resolution	帧分辨率	in

3.1.21. GetFilter

描述：滤波开启状态

语法：

GetFilter(unsigned int nDeviceID, bool& bFilter);

参数：

参数名称	描述	in/out
nDeviceID	设备 id	in
bFilter	滤波开启状态，true-已开启滤波， false-未开启滤波	out

3.1.22. SetFilter

描述：开启/关闭滤波

语法：

SetFilter(unsigned int nDeviceID, bool bFilter);

参数：

参数名称	描述	in/out
nDeviceID	设备 id	in
bFilter	滤波开启状态，true-已开启滤波，false-未开启滤波	in

3.1.23. GetFilterList

描述：获取滤波列表

语法：

GetFilterList(unsigned int nDeviceID, int& nCount, SYFilterType* pFilterType = nullptr);

参数：

参数名称	描述	in/out
nDeviceID	设备 id	in
nCount	滤波列表长度	in/out
pFilterType	滤波列表	in/out

3.1.24. SetDefaultFilter

描述：设置默认滤波

语法：

SetDefaultFilter(unsigned int nDeviceID);

参数：

参数名称	描述	in/out
nDeviceID	设备 id	in

3.1.25. AddFilter

描述：增加滤波

语法：

AddFilter(unsigned int nDeviceID, SYFilterType filterType);

参数：

参数名称	描述	in/out
nDeviceID	设备 id	in
filterType	滤波类型	in

3.1.26. DeleteFilter

描述：移除滤波

语法：

DeleteFilter(unsigned int nDeviceID, int nIndex);

参数：

参数名称	描述	in/out
nDeviceID	设备 id	in
nIndex	滤波列表中的索引	in

3.1.27. ClearFilter

描述：清除滤波

语法：

ClearFilter(unsigned int nDeviceID);

参数：

参数名称	描述	in/out
nDeviceID	设备 id	in

3.1.28. SetFilterParam

描述：设置滤波参数

语法：

SetFilterParam(unsigned int nDeviceID, SYFilterType filterType, int nParamCount, float* pFilterParam);

参数：

参数名称	描述	in/out
nDeviceID	设备 id	in

filterType	滤波类型	in
nParamCount	滤波参数个数	in/out
pFilterParam	滤波参数	in/out

3.1.29. GetFilterParam

描述：获取滤波参数

语法：

GetFilterParam(unsigned int nDeviceID, SYFilterType filterType, int& nParamCount, float* pFilterParam = nullptr);

参数：

参数名称	描述	in/out
nDeviceID	设备 id	in
filterType	滤波类型	in
nParamCount	滤波参数个数	in/out
pFilterParam	滤波参数	in/out

3.1.30. GetMirror

描述：获取水平镜像状态

语法：

GetMirror(unsigned int nDeviceID, bool& bMirror);

参数：

参数名称	描述	in/out
nDeviceID	设备 id	in
bMirror	水平镜像状态，true-已开启水平镜像，false-未开启水平镜像	out

3.1.31. SetMirror

描述：开启/关闭水平镜像

语法：

SetMirror(unsigned int nDeviceID, bool bMirror);

参数：

参数名称	描述	in/out
nDeviceID	设备 id	in
bMirror	水平镜像开关，true-开启水平镜像，false-关闭水平镜像	in

3.1.32. GetFlip

描述：获取垂直翻转状态

语法：

GetFlip(unsigned int nDeviceID, bool& bFlip);

参数：

参数名称	描述	in/out
nDeviceID	设备 id	in
bFlip	垂直翻转状态，true-已开启垂直翻转，false-未开启垂直翻转	out

3.1.33. SetFlip

描述：开启/关闭垂直翻转

语法：

SetMirror(unsigned int nDeviceID, bool bMirror);

参数：

参数名称	描述	in/out
nDeviceID	设备 id	in
bFlip	垂直翻转开关，true-开启垂直翻转，false-关闭垂直翻转	in

3.1.34. GetIntegralTime

描述：获取积分时间

语法：

GetIntegralTime(unsigned int nDeviceID, int& nIntegralTime);

参数：

参数名称	描述	in/out
nDeviceID	设备 id	in
nIntegralTime	积分时间	out

3.1.35. SetIntegralTime

描述：设置积分时间

语法：

SetIntegralTime(unsigned int nDeviceID, int nIntegralTime);

参数：

参数名称	描述	in/out
nDeviceID	设备 id	in
nIntegralTime	积分时间	in

3.1.36. GetIntegralTimeRange

描述：获取积分时间调节范围

语法：

GetIntegralTimeRange(unsigned int nDeviceID, SYResolution depthResolution, int& nMin, int& nMax);

参数：

参数名称	描述	in/out
nDeviceID	设备 id	in
depthResolution	depth 分辨率	in
nMin	积分时间最小值	out
nMax	积分时间最大值	out

3.1.37. GetDistanceMeasureRange

描述：获取测距量程

语法：

GetDistanceMeasureRange(unsigned int nDeviceID, int& nMin, int& nMax);

参数：

参数名称	描述	in/out
nDeviceID	设备 id	in
nMin	量程最小值	out
nMax	量程最大值	out

3.1.38. GetDistanceUserRange

描述：获取用户测距范围

语法：

GetDistanceUserRange(unsigned int nDeviceID, int& nMin, int& nMax);

参数：

参数名称	描述	in/out
nDeviceID	设备 id	in
nMin	测距范围最小值	out
nMax	测距范围最大值	out

3.1.39. SetDistanceUserRange

描述：设置用户测距范围

语法：

SetDistanceUserRange(unsigned int nDeviceID, int nMin, int nMax);

参数：

参数名称	描述	in/out
nDeviceID	设备 id	in
nMin	测距范围最小值	in
nMax	测距范围最大值	in

3.1.40. GetDeviceSN

描述：读取设备 sn 号

语法：

GetDeviceSN(unsigned int nDeviceID, int& nLength, char* pstrSN = nullptr);

参数：

参数名称	描述	in/out
nDeviceID	设备 id	in
nLength	字符长度	in/out
pstrSN	设备 sn 号字符串指针,由外部分配内存，pstrSN 传入 nullptr 时仅获取 nLength	in/out

3.1.41. SetDeviceSN

描述：写入设备 sn 号

语法：

SetDeviceSN(unsigned int nDeviceID, int nLength, const char* pstrSN);

参数：

参数名称	描述	in/out
nDeviceID	设备 id	in
nLength	字符长度	in
pstrSN	设备 sn 号字符串指针	in

3.1.42. GetDeviceHWVersion

描述：读取设备固件版本号

语法：

GetDeviceHWVersion(unsigned int nDeviceID, int& nLength, char* pstrHWVersion = nullptr);

参数：

参数名称	描述	in/out
nDeviceID	设备 id	in
nLength	字符长度	in/out
pstrHWVersion	固件版本号字符串指针,由外部分配内存，pstrHWVersion 传入 nullptr 时仅获取 nLength	in/out

3.1.43. GetDepthColor

描述：获取深度对应伪彩色

语法：

GetDepthColor(unsigned int nDeviceID, int nCount, const unsigned short* pDepth, unsigned char* pColor);

参数：

参数名称	描述	in/out

nDeviceID	设备 id	in
nCount	数据量(内存空间 pDepth 需要 nCount*2 字节，pColor 需要 nCount*3 字节)	in
pDepth	深度数据	in
pColor	深度对应伪彩色(24 位 RGB 格式)	in/out

3.1.44. GetDepthPointCloud

描述：获取深度对应点云数据

用法：

GetDepthPointCloud(unsigned int nDeviceID, int nWidth, int nHeight, const unsigned short* pDepth, SYPointCloudData* pPointCloud, bool bUndistort = false);

参数：

参数名称	描述	in/out
nDeviceID	设备 id	in
nWidth	宽度	in
nHeight	高度	in
pDepth	深度数据	in
pPointCloud	深度对应点云数据,由外部分配内存	in/out
bUndistort	裁剪标志，true-裁剪 false-不裁剪	in

3.1.45. GetRGBD

描述：获取 RGBD

语法：

GetRGBD(unsigned int nDeviceID, int nSourceDepthWidth, int nSourceDepthHeight, unsigned short* pSourceDepth, int nSourceRGBWidth, int nSourceRGBHeight, unsigned char* pSourceRGB, int nTargetWidth, int nTargetHeight, unsigned short* pTargetDepth, unsigned char* pTargetRGB);

参数：

参数名称	描述	in/out
nDeviceID	设备 id	in
nSourceDepthWidth	源深度数据宽度	in
nSourceDepthHeight	源深度数据高度	in
pSourceDepth	源深度数据	in
nSourceRGBWidth	源 RGB 数据宽度	in
nSourceRGBHeight	源 RGB 数据高度	in
pSourceRGB	源 RGB 数据	in
nTargetWidth	RGBD 数据宽度	in
nTargetHeight	RGBD 数据高度	in
pTargetDepth	RGBD 中的深度数据,由外部分配内存,数据长度与源 RGB 长度一致	in/out
pTargetRGB	RGBD 中的 RGB 数据,由外部分配内存,数据长度与源 RGB 长度一致	in/out

3.1.46. GetLastFrameData

描述：获取最新一帧数据

语法：

GetLastFrameData(unsigned int nDeviceID, SYFrameData*& pFrameData);

参数：

参数名称	描述	in/out
nDeviceID	设备 id	in
pFrameData	最后一帧数据	in/out

3.1.47. Undistort

描述：去畸变

语法：

Undistort(unsigned int nDeviceID, const unsigned short* pSource, int nWidth, int nHeight, bool bDepth, unsigned short* pTarget);

参数：

参数名称	描述	in/out
nDeviceID	设备 id	in
pSource	待去畸变数据指针	in
nWidth	图像宽度	in

nHeight	图像高度	in
bDepth	是否是深度数据/RGB 数据	in
pTarget	去畸变结果数据指针，由外部分配内存,数据长度与待去畸变数据指针长度一致	out

3.1.48. GetIntric

描述：获取相机参数

语法：

GetIntric(unsigned int nDeviceID, SYResolution resolution, SYIntrinsics& intrinsics);

参数：

参数名称	描述	in/out
nDeviceID	设备 id	in
resolution	设备分辨率	in
intrinsics	相机参数	in/out

3.1.49. GetTrailFilter

描述：获取拖影滤波开启状态

语法：

GetTrailFilter(unsigned int nDeviceID, bool& bFilter);

参数：

参数名称	描述	in/out
nDeviceID	设备 id	in
bFilter	拖影滤波开启状态	out

3.1.50. SetTrailFilter

描述：开启/关闭拖影滤波

语法：

SetTrailFilter(unsigned int nDeviceID, bool bFilter);

参数：

参数名称	描述	in/out
nDeviceID	设备 id	in
bFilter	拖影滤波开关	in

3.1.51. GetHardWareFilterMode

描述：获取硬件滤波模式开启状态

语法：

GetHardWareFilterMode(unsigned int nDeviceID, bool& bHardwareFilterMode);

参数：

参数名称	描述	in/out
nDeviceID	设备 id	in
bHardwareFilterMode	硬件滤波模式开启状态，true-已开启硬件滤波模式，false-未开启硬件滤波模式	in/out

3.1.52. SetHardWareFilterMode

描述：开启/关闭硬件滤波模式

语法：

SetHardWareFilterMode(unsigned int nDeviceID, bool bHardwareFilterMode);

参数：

参数名称	描述	in/out
nDeviceID	设备 id	in
bHardwareFilterMode	硬件滤波模式开启状态，true-已开启硬件滤波模式，false-未开启硬件滤波模式	in

3.1.53. HaveHardWareFilterMode

描述：硬件滤波模式是否可用

语法：

GetHardWareFilterMode(unsigned int nDeviceID, bool& bHardwareFilterMode);

参数：

参数名称	描述	in/out
nDeviceID	设备 id	in
bHardwareFilterMode	硬件滤波模式可用标志，true-可用，false-不可用	in/out

3.2. 返回参数说明

所有接口返回参数皆为 错误码。详情可看数据结构定义说明

4. 滤波设置说明

4.1. 滤波参数设置说明

幅值滤波 AMPLITITUD

例：

```
float threshold_value{ 0 };
threshold_value[0] = 10;// amplititud_threshold
int num = 1;
SetFilterParam(nDeviceID, filterType, num , threshold_value);
```

中值滤波 MEDIAN

例:

```
float threshold_value{ 0 };
threshold_value[0] = 3;// median_ksize
threshold_value[1] = 1;// median_iterations
int num = 2;
SetFilterParam(nDeviceID, filterType, num , threshold_value);
```

边界滤波 EDGE

例:

```
float threshold_value{ 0 };
threshold_value[0] = 50;//edge_threshold
int num = 1;
SetFilterParam(nDeviceID, filterType, num , threshold_value);
```

斑点滤波 SPECKLE

例:

```
float threshold_value{ 0 };
threshold_value[0] = 40;// speckle_size
threshold_value[1] = 100;// speckle_max_diff
int num = 2;
SetFilterParam(nDeviceID, filterType, num , threshold_value);
```

4.2. 滤波参数范围说明

滤波接口	参数 1-min	参数 1-max	参数 1 推荐值	参数 2-min	参数 2-max	参数 2 推荐值
AMPLITITUD	0	100	6			
MEDIAN	3	5	3	0	5	1
EDGE	20	200	50			
SPECKLE	24	200	40	40	200	

4.3. 滤波调用顺序说明

CS20:中值、边界、斑点、中值

CS30:前段内置了中值、边界、中值，后端可再加入 斑点、中值滤波。

4.4. 硬件滤波开启关闭说明

调用 SetHardWareFilterMode 开启硬件滤波后，滤波将会在模组内部 CPU 进行处理，关闭后将在上位机进行处理滤波逻辑，目前除 CS20 外产品均支持。

4.5. 部分滤波相关接口说明

4.5.1. SetFilter

调用该接口，会开启或者关闭滤波，使滤波生效/不生效。

4.5.2. ClearFilter

调用该接口只会使自己设置的滤波信息清除掉，并不会关闭滤波。会有默认滤波存在。

4.5.3. DeleteFilter

删除自己定义的滤波信息，全部删除后还是会存在默认滤波

4.5.4. AddFilter

添加滤波信息，重复添加只有一个会生效。

5. 数据结构定义说明

详情参考 include 头文件。

6. FQA

f：win 下运行出现 dll 找不到

a：需要将提示的 dll 文件拷贝到程序运行目录

f：Linux 运行时提示 uvc_open:-3

a：获取一下 script 压缩文件，执行一下里面的脚本文件

f：出现 select() timeout. 错误

a：设备打开超时，可能由供电不足，usb 带宽不足造成，建议外接 hub 供电，或者接入不同的 usb 接口

f：噪声点比较大

a：可以通过 GUI 设置滤波参数，获得想要的效果后记住滤波参数，加入到 SDK 中

f：xxx 库找不到

a：通过 run.sh 进行运行程序，保证 run.sh 导入的库路径正确，或者将依赖库安装到 usr/lib 下面

f：cs40 cs20-p 找不到设备

a：确定设备启动以及设备与工控机保持在同一个网段。确保能够 ping 通设备的 ip 地址

f：cs40 cs20p 连接多台设备时无只能找到找到一个或者都找不到

a：要确保设备通过同一个网口接入，建议通过交换机连接多台设备。

7. 关于设备连接

注意：SDK 中并没有限制设备连接上限，理论上是可以接入无限多设备。具体工控机上可以连接多少个设备，要看工控机硬件支持情况。目前经过测试有一下几条需要注意：

1. 一个外接 hub 哪怕有多个 usb 连接口也只能接一个设备。
2. 工控机上一个独立 usb 最多能接两个设备需要根据不同机型具体调整，实际上的独立 usb 接口（有的工控机可能存在多个 usb 口，但是这些 usb 可能用了同一个带宽，同一个供电）
3. 目前已经成功在工控机上接入两台 CS20，一台 CS30。
4. 经过我们优化，可以在同一个 hub 上同时连接两台 CS30 但是需要联系我们更新对应的固件。

免责声明

本出版物中所述的器件应用信息及其他类似内容仅为您提供便利，它们可能由更新之信息所替代。确保应用符合技术规范，是您自身应负的责任。本公司对这些信息不作任何明示或暗示、书面或口头、法定或其他形式的声明或担保，包括但不限于针对其使用情况、质量、性能、适销或特定用途的适用性的声明或担保。本公司对因这些信息及使用这些信息而引起的后果不承担任何责任。未经本公司书面批准，不得将该产品用作生命维持系统中的关键组件。