libsynexens4 SDK Instructions for using multiple machine v1.9

Revision History version				
Date	SDK version	Documentation version	Description	Author
202212114	v4.0.0.0	v1.0	Initial version	YSY
20230424	v4.0.1.0	v1.1	release version	YSY
20230713	v4.0.2.0	v1.2	New support devices added	YSY
20230815	v4.0.3.0	v1.3	New Interfaces	YSY
20230906	v4.1.0.0	v1.4	Update the call process, filter instructions	YSY
20240206	v4.1.2.0	v1.5	New Interfaces	YSY
20240229	v4.1.3.0	v1.6	Update filtering	YSY
20240604	v4.2.1.0	v1.7	Adding Interfaces	YSY

20241022	v4.2.3.0	v1.8	New interfaces added: CS40Pro device support, IP modification interface, IR deep filtering interface	YSY
20250427	v4.2.4.0	v1.9	Add Yaml, add interfaces	YSY

Contents

1. Overview

Supported devices: cs20 Single band cs20 Dual band cs30 Single band cs30 Dual Band CS20-P cs40 CS40 Pro

Supported systems: windows ubuntu20.04/ubuntu18.04(x86_64) armv7 armv8

1.1. Notes

1.1.1. Develop board connect device

When using the development board to connect to the cs20, it is recommended to use an external power supply device for additional power supply when connecting to the cs30. In some cases, connecting the development board to the cs20 without using an external power supply device may cause the cs20 to fail to work properly.

The independent hub is powered as shown in the following figure:

1.1.2. Network configuration

Network configuration is required for cs20p, cs40 and cs40Pro. The default IP of the device is 192.168.1.150. You need to configure your computer device IP to the same network segment, note not to 192.168.1.150, but to the same network segment such as 192.168.1.11

1.1.3. System architecture-related

The UbuntuSDK we provide is only for the Ubuntu x86_64 architecture. If you are using the Ubuntu system on your development board, please choose the corresponding SDK based on your system architecture.

1.1.4. About Graphics Display

1. We offer two executable programs: SDKTest has an executable program for graphics rendering, and SDkTest NotopencV does not have an executable program for graphics rendering. Linux can configure the program to be run at present in the run.sh script file.

2. Graphics rendering depends on the opencv library, and since the system graphics rendering library is different, opencv needs to be recompiled to run the SDKTest executable.

3. When using ssh for remote connection, graphic rendering cannot be used.

2. Environment configuration

2.1. Ubuntu environment configuration (taking Cmake as an example)

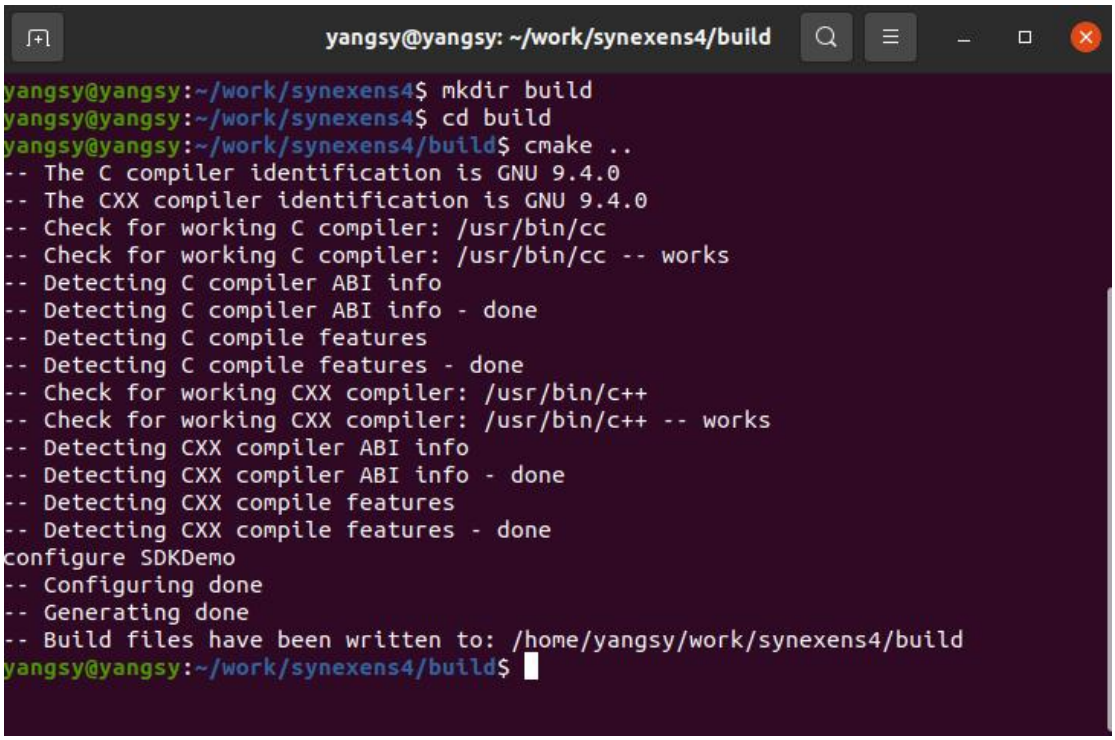
2.1.1. Install dependencies

```
sudo apt install libudev-dev

sudo apt install zlib1g-dev
```

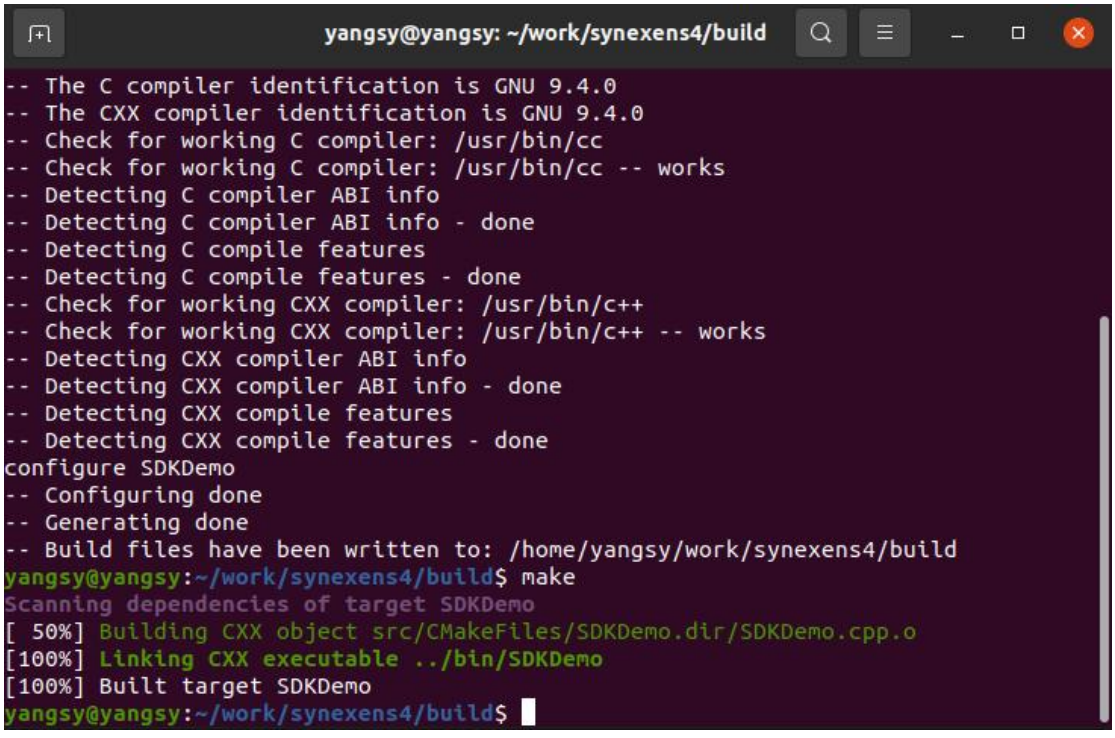
2.1.2. Writing CmakeLists.txt requires familiarity with CMake

```
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. Create a project compilation file



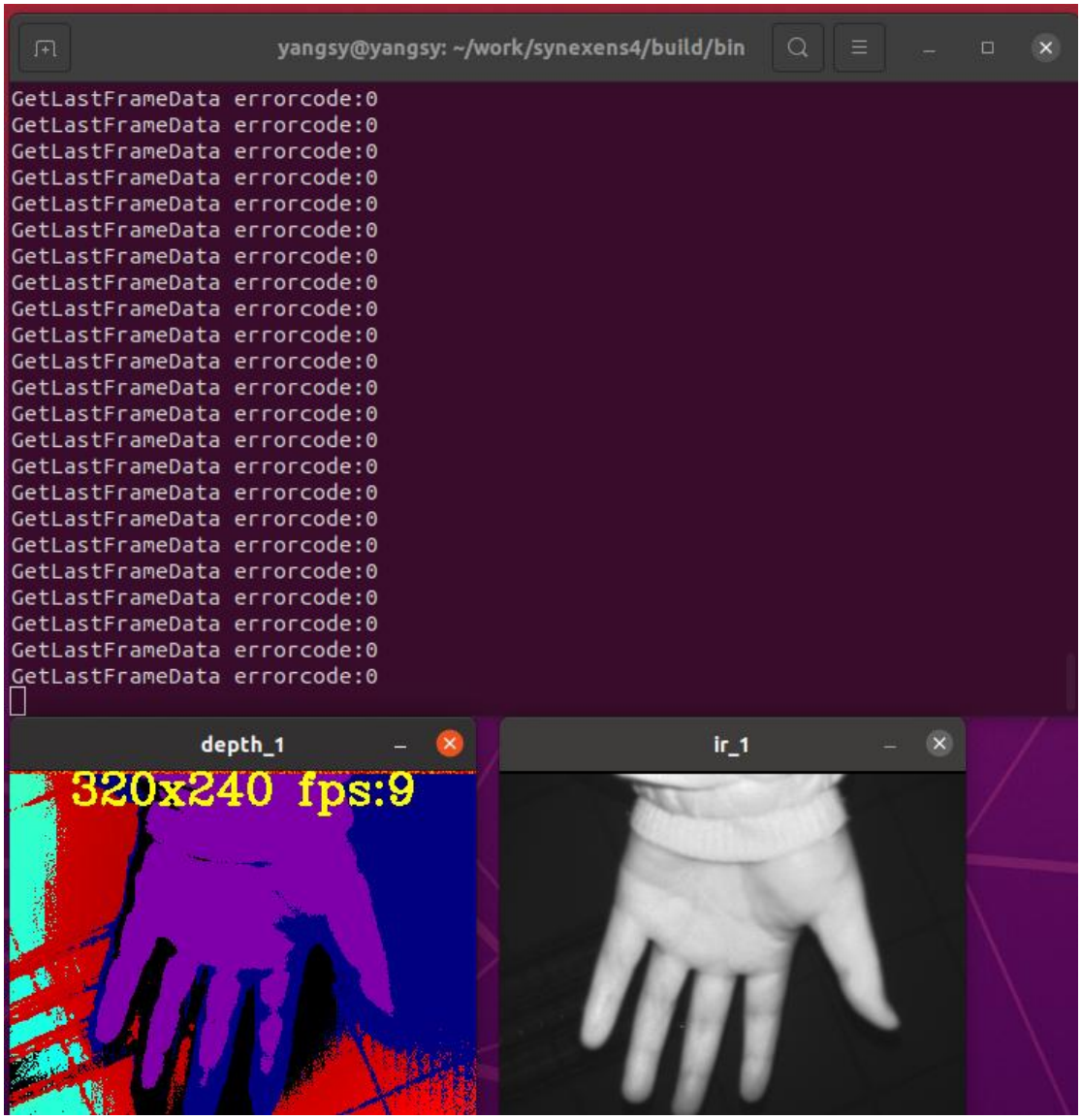
2.1.4. make Compile



2.1.5. Run the executable file to test the effect

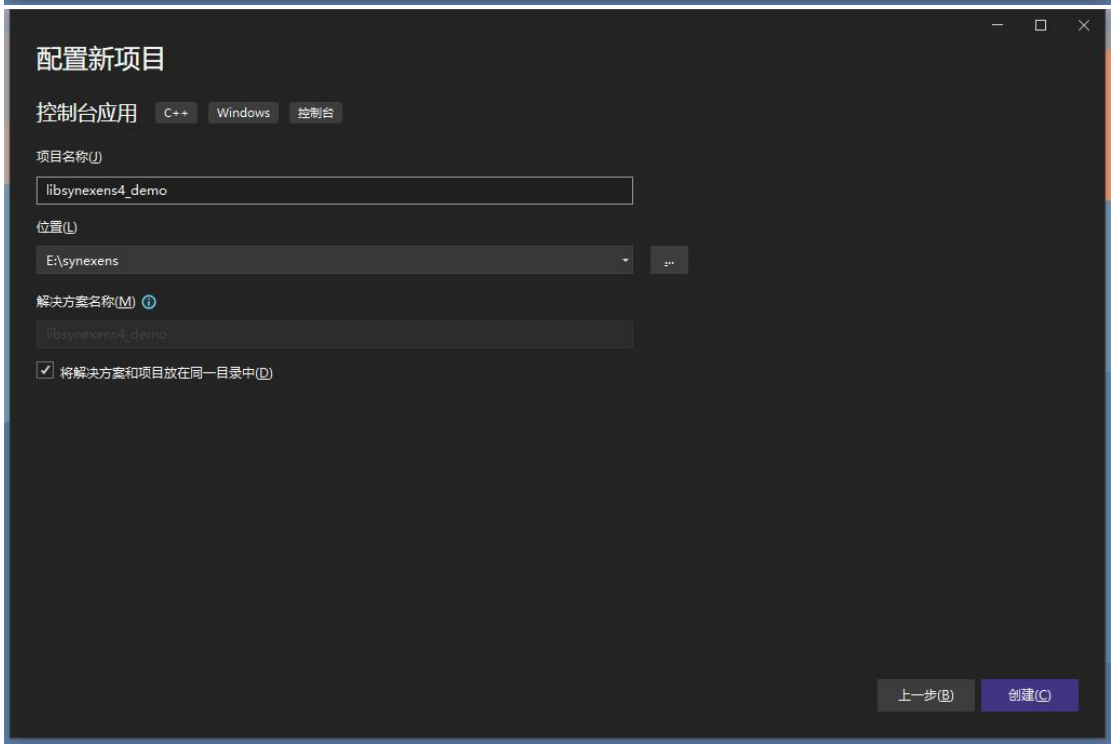
Configure LD_LIBRARY_PATH before running the program to find the library files the program depends on. The run.sh script is written in the example to facilitate the execution of the program.

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

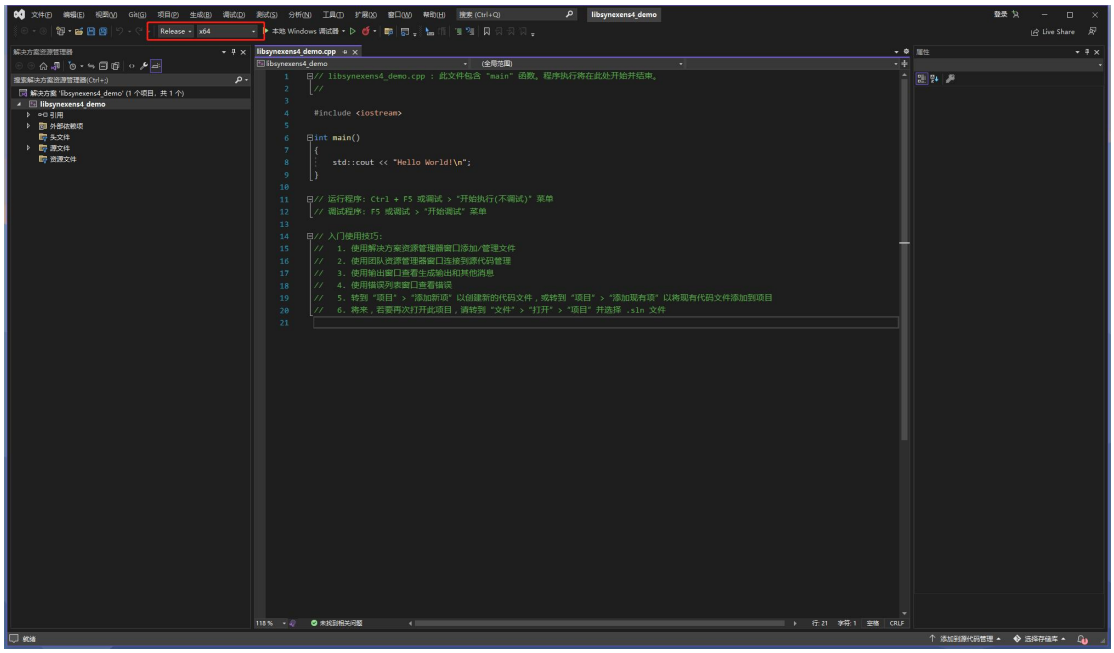


2.2. Windows Environment configuration (taking vs2022 as an example)

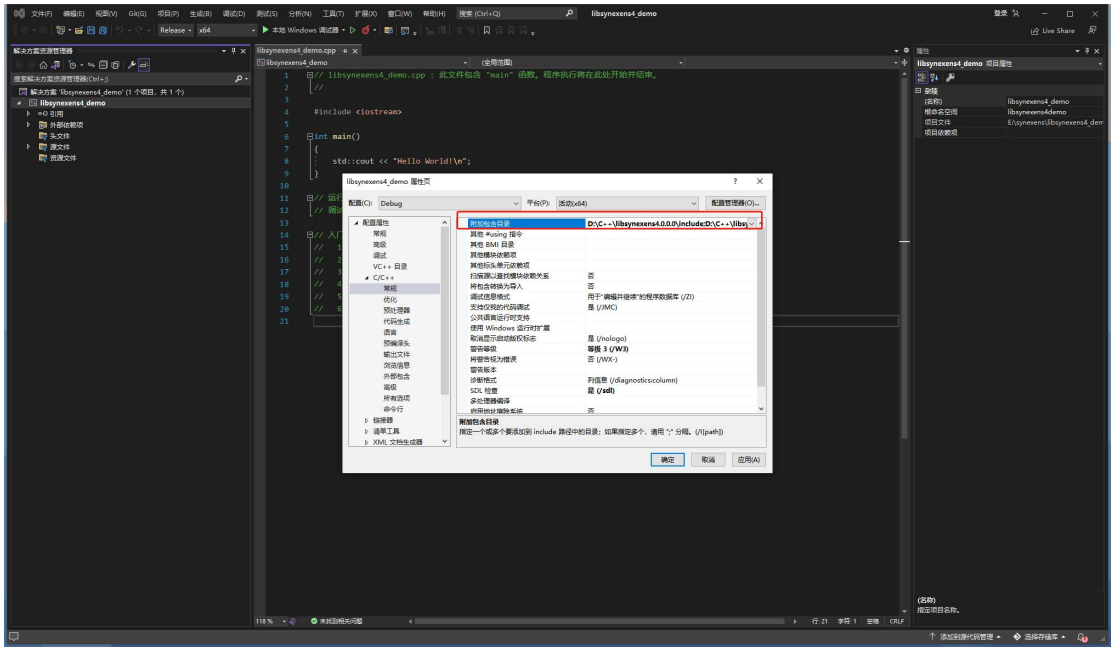
2.2.1. Create a VS project

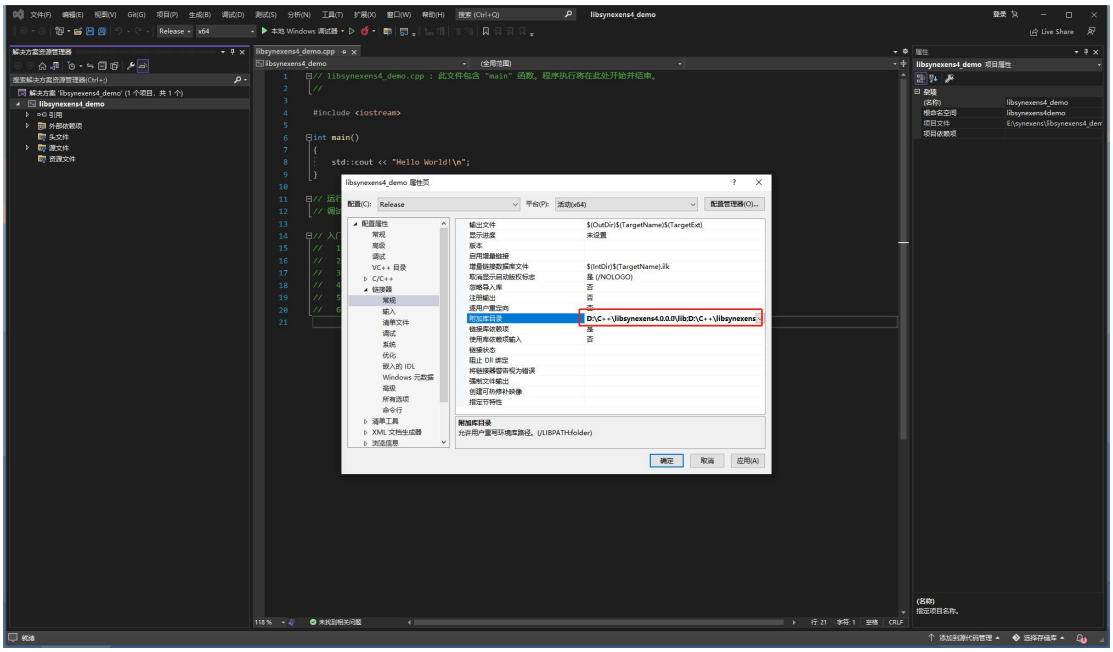


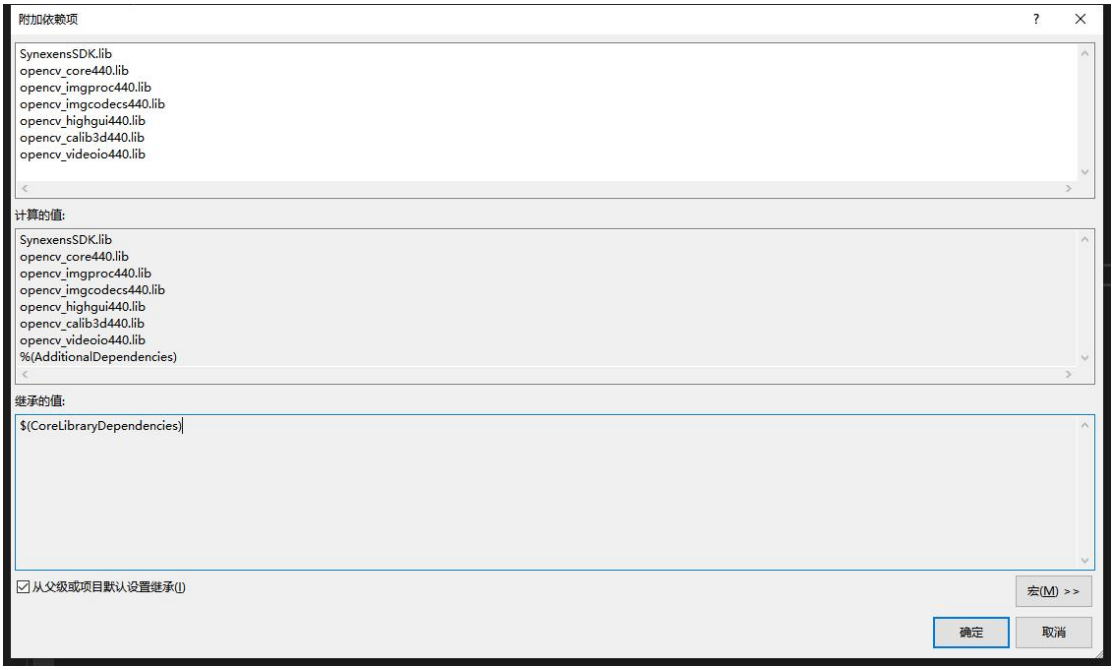
2.2.2. Select the solution and platform corresponding to the SDK

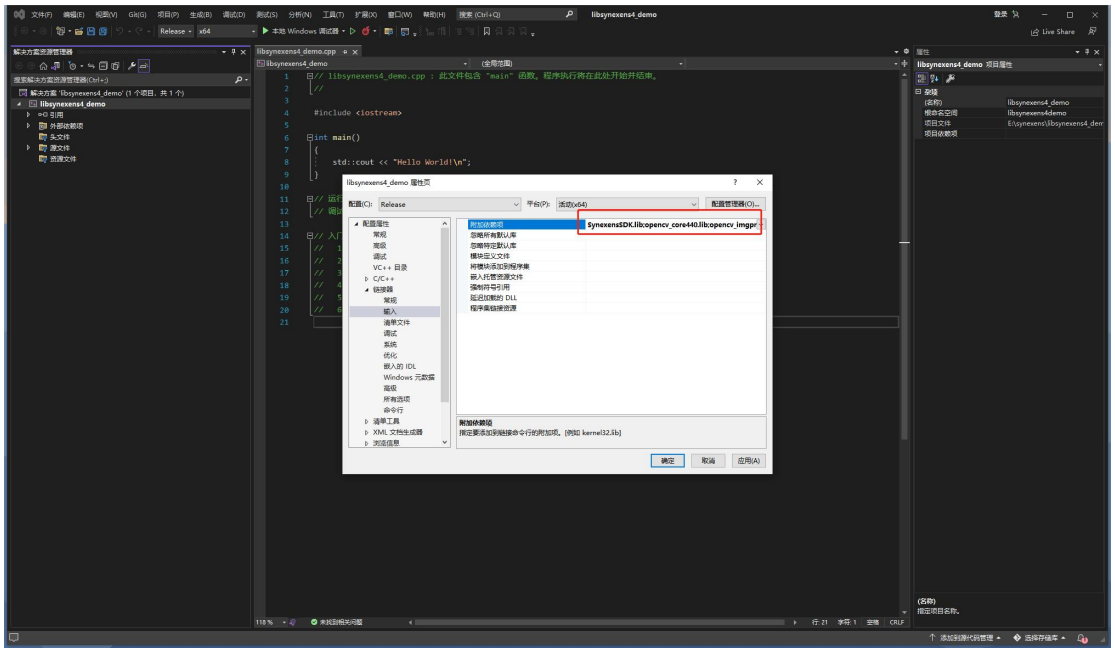


2.2.3. Configure the sdk header file path and library path in the project properties









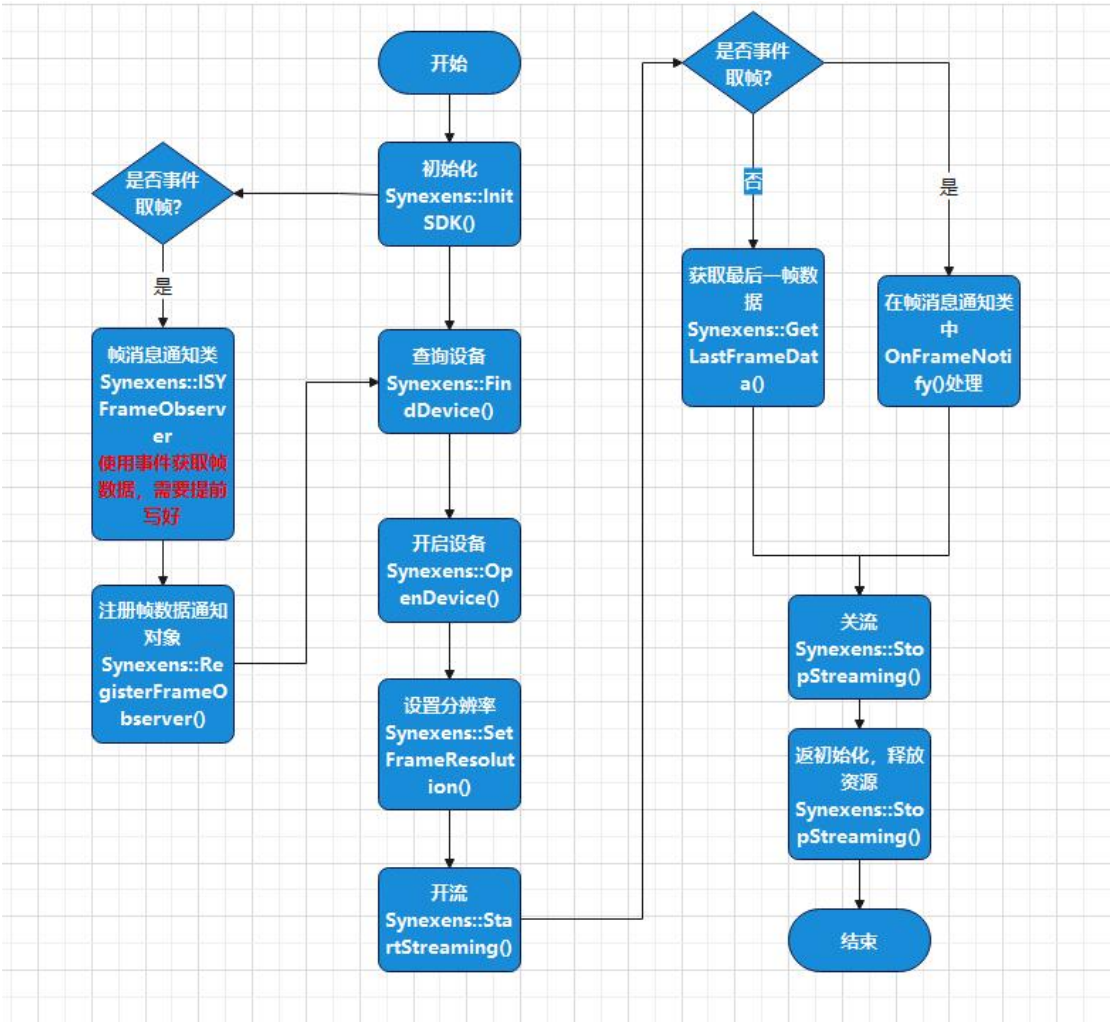
Running demo requires an opencv dependency library, but developing it yourself does not require an opencv dependency

2.2.4. After the configuration is completed, you can enter the project for development. If you need to run the demo, just copy and run the demo code

Note: You need to configure the include path yourself to run the demo, and you need to copy the missing dll files to the program running directory yourself

2.3. The SDK must call the process

2.3.1. Standard invocation process



2.3.2. Yaml call process

Set the resolution of the standard flow and open the stream. Replace parsing configuration file (ParseConfiguration) - using the configuration can be open flow (StartStreamingWithConfiguration)

3. API Overview

3.1. Global Interface

3.1.1. GetSDKVersion

Description: Get the SDK version number

Grammar:

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

Parameters:

Parameter name	Description	in/out
nLenght	Character length	in/out
pstrSDKVersion	SDK version number string pointer	in/out

3.1.2. InitSDK

Description: Initialize the SDK

Syntax:

InitSDK();

3.1.3. UnInitSDK

Description: Deinitialize the SDK and release resources

Syntax:

UnInitSDK();

3.1.4. RegisterErrorObserver

Description: Register error message notification object pointer

Syntax:

RegisterErrorObserver(ISYErrorObserver* pObserver);

Parameters:

Parameter name	Description	in/out
pObserver	Error message notifies object pointer	in

3.1.5. RegisterEventObserver

Description: Register event notification object pointer

Syntax:

RegisterEventObserver(ISYEventObserver* pObserver);

Parameters:

Parameter name	Description	in/out
pObserver	Event notification object pointer	in

3.1.6. RegisterFrameObserver

Description: Register data frame notification object pointer

Syntax:

RegisterFrameObserver(ISYFrameObserver* pObserver);

Parameters:

Parameter name	Description	in/out
pObserver	Data frame notifies object pointer	in

3.1.7. UnRegisterErrorObserver

Description: Cancel the error message notification object pointer

Syntax:

UnRegisterErrorObserver(ISYErrorObserver* pObserver);

Parameters:

Parameter name	Description	in/out
pObserver	Error message notifies object pointer	in

3.1.8. UnRegisterEventObserver

Description: Logout event notification object pointer

Syntax:

UnRegisterEventObserver(ISYEventObserver* pObserver);

Parameters:

Parameter name	Description	in/out
pObserver	Event notification object pointer	in

3.1.9. UnRegisterFrameObserver

Description: Unregister the data frame notification object pointer

Syntax:

UnRegisterFrameObserver(ISYFrameObserver* pObserver);

Parameters:

Parameter name	Description	in/out
pObserver	Data frame notifies object pointer	in

3.1.10. FindDevice

Description: Find the device

Grammar:

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

Parameters:

Parameter name	Description	in/out
nCount	Number of devices	in/out
pDevice	Device information, with externally allocated memory, pDevice gets only nCount when passing in nullptr	in/out

3.1.11. OpenDevice

Description: Turn on the device

Grammar:

OpenDevice(const SYDeviceInfo& deviceInfo);

Parameters:

Parameter name	Description	in/out
deviceInfo	Device information	in

3.1.12. CloseDevice

Description: Turn off the device

Grammar:

CloseDevice(unsigned int nDeviceID);

Parameters:

Parameter name	Description	in/out
nDeviceID	Device id	in

3.1.13. QueryDeviceSupportFrameType

Description: Query the data frame type supported by the device

Syntax:

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

Parameters:

Parameter name	Description	in/out
nDeviceID	Device id	in
nCount	The number of supported data frame types, used only as the return quantity when pSupportType is empty, otherwise used to verify whether the pSupportType memory allocation quantity matches	in/out
pSupportType	Supported data frame type,	in/out

	with externally allocated memory, pFrameType gets only nCount when passed to nullptr	

3.1.14. QueryDeviceSupportResolution

Description: Query the frame resolution supported by the device

Syntax:

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

Parameters:

Parameter name	Description	in/out
nDeviceID	Device id	in
supportType	Frame type	in
nCount	The number of supported resolutions, used only as the return quantity when pResolution is empty, otherwise used to verify whether the pResolution memory allocation quantity matches	in/out
pResolution	Supported resolution types, externally allocated memory,	in/out

	pResolution gets only nCount when passing in nullptr	

3.1.15. GetCurrentStreamType

Description: Get the current stream type

Syntax:

GetCurrentStreamType(unsigned int nDeviceID);

Parameters:

Parameter name	Description	in/out
nDeviceID	Device ID	in

3.1.16. StartStreaming

Description: Start the data stream

Syntax:

StartStreaming(unsigned int nDeviceID, SYStreamType streamType);

Parameters:

Parameter name	Description	in/out
nDeviceID	Device id	in
streamType	Data stream type	in

3.1.17. StopStreaming

Description: Shut down the data stream

Syntax:

StopStreaming(unsigned int nDeviceID);

Parameters:

Parameter name	Description	in/out
nDeviceID	Device id	in

3.1.18. ChangeStreaming

Description: Switching data streams

Syntax:

ChangeStreaming(unsigned int nDeviceID, SYStreamType streamType);

Parameter:

Parameter name	Description	in/out
nDeviceID	Device id	in
streamType	Data stream type	in

3.1.19. SetFrameResolution

Description: Set the resolution (if the data stream has been started, an internal process is executed to turn off the stream -> set the resolution -> turn the stream back on)

Syntax:

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

Parameters:

Parameter name	Description	in/out
nDeviceID	Device id	in
frameType	Frame type	in
resolution	Frame resolution	in

3.1.20. GetFrameResolution

Description: Get the device frame resolution

Syntax:

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

Parameters:

Parameter name	Description	in/out
nDeviceID	Device id	in
frameType	Frame type	in
resolution	Frame resolution	in

3.1.21. GetFilter

Description: The filter is on

Grammar:

GetFilter(unsigned int nDeviceID, bool& bFilter);

Parameter:

Parameter name	Description	in/out
nDeviceID	Device id	in
bFilter	Filter enabled, true- Filter enabled, false- Filter not enabled	out

3.1.22. SetFilter

Description: Turn filtering on/off

Grammar:

SetFilter(unsigned int nDeviceID, bool bFilter);

Parameters:

Parameter name	Description	in/out
nDeviceID	Device id	in
bFilter	Filter enabled, true- Filter enabled, false- Filter not enabled	in

3.1.23. GetFilterList

Description: Get the filtered list

Syntax:

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

Parameters:

Parameter name	Description	in/out
nDeviceID	Device id	in
nCount	Filter list length	in/out
pFilterType	Filter list	in/out

3.1.24. SetDefaultFilter

Description: Set the default filter

Syntax:

SetDefaultFilter(unsigned int nDeviceID);

Parameters:

Parameter name	Description	in/out
nDeviceID	Device id	in

3.1.25. AddFilter

Description: Add a filter

Grammar:

AddFilter(unsigned int nDeviceID, SYFilterType filterType);

Parameters:

Parameter name	Description	in/out
nDeviceID	Device id	in
filterType	Filtering type	in

3.1.26. DeleteFilter

Description: Remove the filter

Grammar:

DeleteFilter(unsigned int nDeviceID, int nIndex);

Parameters:

Parameter name	Description	in/out
nDeviceID	Device id	in
nIndex	Filter the index in the list	in

3.1.27. ClearFilter

Description: Clear the filter

Grammar:

ClearFilter(unsigned int nDeviceID);

Parameters:

Parameter name	Description	in/out
nDeviceID	Device id	in

3.1.28. SetFilterParam

Description: Set the filter parameters

Grammar:

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

Parameters:

Parameter name	Description	in/out
nDeviceID	Device id	in
filterType	Filter type	in
nParamCount	Number of filtering parameters	in/out
pFilterParam	Filter parameters	in/out

3.1.29. GetFilterParam

Description: Obtain the filtering parameters

Syntax:

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

Parameters:

Parameter name	Description	in/out
nDeviceID	Device id	in
filterType	Filter type	in
nParamCount	Number of filtering parameters	in/out
pFilterParam	Filter parameters	in/out

3.1.30. GetMirror

Description: Get the horizontal mirror state

Syntax:

GetMirror(unsigned int nDeviceID, bool& bMirror);

Parameters:

Parameter name	Description	in/out
nDeviceID	Device id	in
bMirror	Horizontal image status, true- Horizontal image enabled, false- Horizontal image not enabled	out

3.1.31. SetMirror

Description: Turn on/off the horizontal mirror

Syntax:

SetMirror(unsigned int nDeviceID, bool bMirror);

Parameters:

Parameter name	Description	in/out
nDeviceID	Device id	in
bMirror	Horizontal mirror switch, true- Turn on horizontal mirror, false- turn off horizontal mirror	in

3.1.32. GetFlip

Description: Obtain the vertical flip state

Grammar:

GetFlip(unsigned int nDeviceID, bool& bFlip);

Parameters:

Parameter name	Description	in/out
nDeviceID	Device id	in
bFlip	Vertical flip status, true- Vertical flip enabled, false- Vertical flip not enabled	out

3.1.33. SetFlip

Description: Turn on/off vertical flip

Grammar:

SetMirror(unsigned int nDeviceID, bool bMirror);

Parameters:

Parameter name	Description	in/out
nDeviceID	Device id	in
bFlip	Vertical flip switch, true-Enable vertical flip, false-disable vertical flip	in

3.1.34. GetIntegralTime

Description: Get integraltime

Grammar:

GetIntegralTime(unsigned int nDeviceID, int& nIntegralTime);

Parameters:

Parameter name	Description	in/out
nDeviceID	Device id	in
nIntegralTime	Integral time	out

3.1.35. SetIntegralTime

Description: Set the integral time

Grammar:

SetIntegralTime(unsigned int nDeviceID, int nIntegralTime);

Parameters:

Parameter name	Description	in/out
nDeviceID	Device id	in
nIntegralTime	Integral time	in

3.1.36. GetIntegralTimeRange

Description: Get the integration time adjustment range

Grammar:

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

Parameters:

Parameter name	Description	in/out
nDeviceID	Device id	in
depthResolution	depth resolution	in
nMin	Minimum integration time	out

nMax	Maximum integral time	out

3.1.37. GetDistanceMeasureRange

Description: Get the ranging range

Grammar:

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

Parameters:

Parameter name	Description	in/out
nDeviceID	Device id	in
nMin	Range minimum	out
nMax	Maximum range	out

3.1.38. GetDistanceUserRange

Description: Get the ranging range of the user

Grammar:

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

Parameter:

Parameter name	Description	in/out
nDeviceID	Device id	in
nMin	Minimum ranging range	out

nMax	Maximum ranging range	out

3.1.39. SetDistanceUserRange

Description: Set the ranging range for the user

Grammar:

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

Parameters:

Parameter name	Description	in/out
nDeviceID	Device id	in
nMin	Minimum ranging range	in
nMax	Maximum ranging range	in

3.1.40. GetDeviceSN

Description: Read the device sn number

Grammar:

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

Parameters:

Parameter name	Description	in/out
nDeviceID	Device id	in
nLength	Character length	in/out

pstrSN	Device sn string pointer, externally allocated memory, pstrSN gets only nLength when passing in nullptr	in/out

3.1.41. SetDeviceSN

Description: Write the device sn number

Syntax:

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

Parameters:

Parameter name	Description	in/out
nDeviceID	Device id	in
nLength	Character length	in
pstrSN	Device sn string pointer	in

3.1.42. GetDeviceHWVersion

Description: Read the device firmware version number

Grammar:

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

Parameters:

Parameter name	Description	in/out

nDeviceID	Device id	in
nLength	Character length	in/out
pstrHWVersion	Firmware version number string pointer, externally allocated memory, pstrHWVersion gets only nLength when passed to nullptr	in/out

3.1.43. GetDepthColor

Description: Obtain pseudo-color corresponding to depth

Grammar:

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

Parameters:

Parameter name	Description	in/out
nDeviceID	Device id	in
nCount	Data volume (Memory space pDepth requires nCount*2 bytes, pColor requires nCount*3 bytes)	in
pDepth	Depth data	in
pColor	Depth corresponds to pseudo-color (24-bit RGB format)	in/out

3.1.44. GetDepthPointCloud

Description: Obtain depth-corresponding point cloud data

Usage:

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

Parameters:

Parameter name	Description	in/out
nDeviceID	Device id	in
nWidth	Width	in
nHeight	Height	in
pDepth	Depth data	in
pPointCloud	Depth corresponds to point cloud data, with externally allocated memory	in/out
bUndistort	Trim the flag, true- trim false- Do not trim	in

3.1.45. GetRGBD

Description: Get the RGBD

Grammar:

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, bool bTOFtoRGB = true);

Parameters:

Parameter name	Description	in/out
nDeviceID	Device id	in
nSourceDepthWidth	Source depth data width	in
nSourceDepthHeight	Source depth data height	in
pSourceDepth	Source depth data	in
nSourceRGBWidth	Source RGB data width	in
nSourceRGBHeight	Source RGB data height	in
pSourceRGB	Source RGB data	in
nTargetWidth	RGBD data width	in
nTargetHeight	RGBD data height	in
pTargetDepth	Depth data in RGBD, with externally allocated memory, the data length is the same as the source RGB length	in/out

pTargetRGB	RGB data in the RGBD, with externally allocated memory, the data length is the same as the source RGB length	in/out
bTOFtoRGB	Alignment true- Align with RGB false- Align with TOF	in/out

3.1.46. GetLastFrameData

Description: Get the latest frame data

Syntax:

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

Parameters:

Parameter name	Description	in/out
nDeviceID	Device id	in
pFrameData	The last frame data	in/out

3.1.47. Undistort

Description: Remove distortion

Grammar:

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

Parameters:

Parameter name	Description	in/out
nDeviceID	Device id	in
pSource	To remove the distorted data pointer	in
nWidth	Image width	in
nHeight	Image height	in
bDepth	Whether it is depth data /RGB data	in
pTarget	De-distortion result data pointer, with externally allocated memory, data length is the same as the length of the data pointer to be de-distorted	out

3.1.48. GetIntric

Description: Get camera parameters

Grammar:

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

Parameters:

Parameter name	Description	in/out
nDeviceID	Device id	in
resolution	Device resolution	in
intrinsics	Camera parameters	in/out

3.1.49. GetTrailFilter

Description: Get the trailing filter on

Syntax:

GetTrailFilter(unsigned int nDeviceID, bool& bFilter);

Parameters:

Parameter name	Description	in/out
nDeviceID	Device id	in
bFilter	Trailing filter on	out

3.1.50. SetTrailFilter

Description: Turn on/off the trailing filter

Syntax:

SetTrailFilter(unsigned int nDeviceID, bool bFilter);

Parameters:

Parameter name	Description	in/out
nDeviceID	Device id	in
bFilter	Trailing filter switch	in

3.1.51. GetHardWareFilterMode

Description: Get the hardware filter mode on status

Grammar:

GetHardWareFilterMode(unsigned int nDeviceID, bool& bHardwareFilterMode);

Parameters:

Parameter name	Description	in/out
nDeviceID	Device id	in
bHardwareFilterMode	Hardware Filter mode enabled, true- Hardware Filter mode enabled, false- Hardware Filter mode not enabled	in/out

3.1.52. SetHardWareFilterMode

Description: Turn hardware filter mode on/off

Grammar:

SetHardWareFilterMode(unsigned int nDeviceID, bool bHardwareFilterMode);

Parameter:

Parameter name	Description	in/out
nDeviceID	Device id	in
bHardwareFilterMode	Hardware Filter mode enabled, true- Hardware Filter mode enabled, false- Hardware Filter mode not enabled	in

3.1.53. HaveHardWareFilterMode

Description: Is the hardware filter mode available

Syntax:

GetHardWareFilterMode(unsigned int nDeviceID, bool& bHardwareFilterMode);

Parameters:

Parameter name	Description	in/out
nDeviceID	Device id	in
bHardwareFilterMode	Hardware filter mode available flag, true- available, false- Not available	in/out

3.1.54. ChangeDeviceIP

Description: Modify the device IP

Syntax:

ChangeDeviceIP(unsigned int nDeviceID, unsigned int nIP);

Parameters:

Parameter name	Description	in/out
nDeviceID	Device id	in
nIP	32 is the IP address	in

3.1.55. IRFilter

Description: Filter the depth data using IR

Syntax:

IRFilter(unsigned int nDeviceID, unsigned short* pDepth, unsigned short* pIR, int nWidth, int nHeight, int nThreshold);

Parameters:

Parameter name	Description	in/out
nDeviceID	Device id	in
pDepth	Depth data pointer	in/out
pIR	IR data pointer	in
nWidth	Data width	in
nHeight	Data height	in
nThreshold	Filtering threshold	in

3.1.56. SetFrameRate

Description: Set the frame rate

Syntax:

SetFrameRate(unsigned int nDeviceID, unsigned int nFrameRate);

Parameters:

Parameter name	Description	in/out
nDeviceID	Device id	in
nFrameRate	Frame rate 10-30	in/out

3.1.57. GetFrameRate

Description: Get the frame rate

Grammar:

GetFrameRate(unsigned int nDeviceID, unsigned int& nFrameRate);

Parameters:

Parameter name	Description	in/out
nDeviceID	Device id	in
nFrameRate	Frame rate 10-30	in/out

3.1.58. GetTemperature

Description: Get the temperature

Grammar:

GetTemperature(unsigned int nDeviceID, float& fltTemperature);

Parameter:

Parameter name	Description	in/out
nDeviceID	Device id	in
nFrameRate	Temperature	in/out

3.1.59. IsFrameRateEnabled

Description: Query whether the frame rate is adjustable

Syntax:

IsFrameRateEnabled(unsigned int nDeviceID, bool& bEnabled);

Parameters:

Parameter name	Description	in/out
nDeviceID	Device id	in
bEnabled	Frame rate is adjustable marker	in/out

3.1.60. ParseConfiguration

Description: Parse the configuration file

Syntax:

ParseConfiguration(int nStringLength, const char* pFileName, SYConfiguration & configuration);

Parameters:

Parameter name	Description	in/out

nStringLength	Path string length	in
pFileName	Configure the full path of the file	in/out
configuration	Configuration data	in/out

3.1.61. StartStreamingWithConfiguration

Description: Stream using configuration

Syntax:

StartStreamingWithConfiguration(unsigned int nDeviceID, const SYConfiguration & configuration);

Parameters:

Parameter name	Description	in/out
nDeviceID	Device id	in
configuration	Configuration data	in

3.1.62. ExportConfiguration

Description: Export the configuration file

Syntax:

ExportConfiguration(unsigned int nDeviceID, int nStringLength, char* pFileName);

Parameters:

Parameter name	Description	in/out
nDeviceID	Device id	in
nStringLength	Path string length	in
pFileName	Configure the full path of the file	in

3.2. Return parameter description

All interface return parameters are error codes. See the data structure definition description for details

4. Filtering Settings Instructions

4.1. Filtering parameter setting instructions

Amplitude filter AMPLITITUD

Example:

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

MEDIAN filter median

Example:

```
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);
```

EDGE filtering

Example:

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

SPECKLE filtering

Example:

```
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);
```

Time-domain filtering SYFILTERTYPE_EXTRA

Example:

```
float threshold_value{ 0 };
threshold_value[0] = 3; // speckle_size
int num = 1;
```

SetFilterParam(nDeviceID, filterType, num , threshold_value);

4.2. Filter parameter range description

Filter interface	Parameters 1-min	Parameter 1-max	Parameter 1 Recommended value	Parameter: 2-min	Parameter 2-max	Parameter 2 Recommended values
AMPLITITUD	0	100	6			
MEDIAN	3	5	3	1	5	1
EDGE	20	200	50			
SPECKLE	24	200	40	24	200	100
IR	40	200	64			
SYFILTERTYPE_EXTRA	2	10	3			

4.3. Filter call order instructions

CS20: Median, boundary, speckle, median

CS30: Median, boundary, median are built-in in the front end, and spot, median filtering can be added in the back end.

4.4. Hardware filter on/off instructions

When hardware filtering is enabled by calling SetHardWareFilterMode, the filtering will be processed on the CPU inside the module. When it is disabled, the filtering logic will be processed on the host computer. Currently, all products except CS20 are supported.

4.5. Descriptions of some filtering-related interfaces

4.5.1. SetFilter

Calling this interface will enable or disable the filtering, making the filtering effective/ineffective.

4.5.2. ClearFilter

Calling this interface will only clear the filter information you set yourself, not turn off the filter. There will be a default filter.

4.5.3. DeleteFilter

Delete your own defined filter information, and the default filter will still exist after all of it is deleted

4.5.4. AddFilter

Add filter information. Only one repeated addition will take effect.

5. Data structure definition description

See the include header file for details.

6. Yaml configuration instructions

6.1. Parameter Description

Level 1 Configuration Name	Level 1 Description	Secondary Configuration Name	Secondary Description	Level 3 Configuration Name	Level Three Explanation
IntegralTime	Integral time				
FrameRate	Frame rate				
HorizontalMirror	Whether to enable horizontal mirror				
VerticalFlip	Whether to enable vertical flip				
StreamType	Data frame type				
ResolutionDepth	Depth data				
ResolutionRGB	RGB data resolution				
Filter	Filtering				
		UseFilter	Whether to enable filtering		
		FilterSize	There are several filters configured		
		Filter1	The first filter configuration		
				FilterType	Filter type
				FilterParam	Filter parameters (configurable multiple)
		Filter2	The second		

	Filter3...	filter configuration The... A filtering configuration

6.2. Parameter value description

Parameter name	Parameter values	Value meaning
IntegralTime	10-3000	Exposure, int
FrameRate	10-30	Frame rate, int
HorizontalMirror	true/false	
VerticalFlip	true/false	
StreamType	1-8	int
		1 RAW
		2 Depth
		3 RGB
		4 Depth+ir
		5 Depth+RGB
		6 Depth+ir+RGB
		7 RGBD
		8 raw+RGB
ResolutionDepth	1, 2,	int
		1 320*240
		2,640 *480
ResolutionRGB	2,3,4,5,6	int
		2 640*480
		3960 *540
		4 1920*1080
		5 1600*1200
		6 800*600
FilterType	1,2,3,4,8	int
		1 Median filtering
		2 Amplitude filtering
		3 Boundary filtering
		4 Speckle Filtering
		8 Time-domain filtering
FilterParam		For int Settings, read the filter Settings instructions

ps: For the range of filter parameter values, refer to the filter parameter range description

6.3. Device support parameter range description

Device Name	Support Settings	Value range
CS20 single frequency		
	IntegralTime	0-580
	HorizontalMirror	true/false
	VerticalFlip	true/false
	StreamType	1, 4-trichlorobenzene
	ResolutionDepth	1
CS20 dual-band		
	IntegralTime	0-1800
	HorizontalMirror	true/false
	VerticalFlip	true/false
	StreamType	1, 4-trichlorobenzene
	ResolutionDepth	1, 2,
CS30 Single & Dual		
	IntegralTime	10-2800
	HorizontalMirror	true/false
	VerticalFlip	true/false
	StreamType	1-8
	ResolutionDepth	1, 2,
	ResolutionRGB	3, 4
CS40Pro		
	IntegralTime	10-1951
	HorizontalMirror	true/false
	VerticalFlip	true/false
	StreamType	1-8
	ResolutionDepth	1, 2,
	ResolutionRGB	2 and 6
	FrameRate	10-30

ps: Filtering is supported by default across all devices. It can be configured according to the filter parameter Settings instructions

7. FQA

f: dll can't be found when running under win

a: You need to copy the prompted dll file to the program's running directory

f: The Linux runtime prompts uvc_open:-3

a: Get the script compressed file and execute the script file inside

f: select() timeout. Error appears

a: Device startup timeout may be caused by insufficient power supply or insufficient usb bandwidth. It is recommended to connect an external hub for power supply or connect to a different usb interface

f: The noise point is quite loud

a: You can set the filter parameters through the GUI, remember the filter parameters after achieving the desired effect, and add them to the SDK

f: The xxx library can't be found

a: run the program through run.sh to ensure that the paths of the libraries imported by run.sh are correct, or install the dependent libraries under usr/lib

f: cs40 cs20-p can't find the device

a: Make sure the device is started and the device is in the same network segment as the industrial control computer. Make sure the ip address of the device can be pinged

f: cs40 cs20p can only find one or none when connecting multiple devices

a: To ensure that devices are connected via the same network port, it is recommended to connect multiple devices via a switch.

8. About device connection

Note: There is no limit on device connection limits in the SDK. In theory, an unlimited number of devices can be connected. The exact number of devices that can be connected to the industrial computer depends on the hardware support of the industrial computer. Here are a few points to note after testing:

1. An external hub can only connect one device, even if it has multiple usb ports.

2. The maximum number of devices that can be connected to a separate usb on an industrial computer needs to be adjusted according to different models. The actual separate usb interface (some industrial computers may have multiple usb ports, but these usb ports may use the same bandwidth and the same power supply)

3. Two CS20s and one CS30 have been successfully connected to the industrial computer.

4. After our optimization, two CS30s can be connected simultaneously on the same hub, but you need to contact us to update the corresponding firmware.

Disclaimer

The device application information and other similar content described in this publication is provided for your convenience only and may be replaced by updated information. It is your own responsibility to ensure that the application complies with the technical specifications. The Company makes no express or implied, written or oral, statutory or otherwise representations or warranties regarding this information, including but not limited to representations or warranties regarding its use, quality, performance, merchantability or fitness for a particular purpose. The Company shall not be liable for any consequences arising

out of or in connection with the information and the use of such information. The product may not be used as a critical component of life support systems without the written approval of the Company.

