SynexensSDK4 Usage Description v1.4

Revision history version				
Date	SDK version	Documentation version	Description	Author
20220310	v0.7.3.0	v1.0	Single mode group connection	YSY
20230424	v4.0.1.0	v1.1	Update interface/add new module connections	YSY
20230713	v4.0.2.0	v1.2	Added support devices	YSY
20230815	v4.0.3.0	v1.3	New interface	YSY
20230906	v4.1.0.0	v1.4	Update call flow, filter instructions	YSY

Directory

1. Overview

Support equipment: cs20 single frequency cs20 dual frequency cs30 single frequency cs30 dual frequency CS20-P cs40

Supported system: windows ubuntu20.04 armv7 armv8

2. Environment Configuration

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

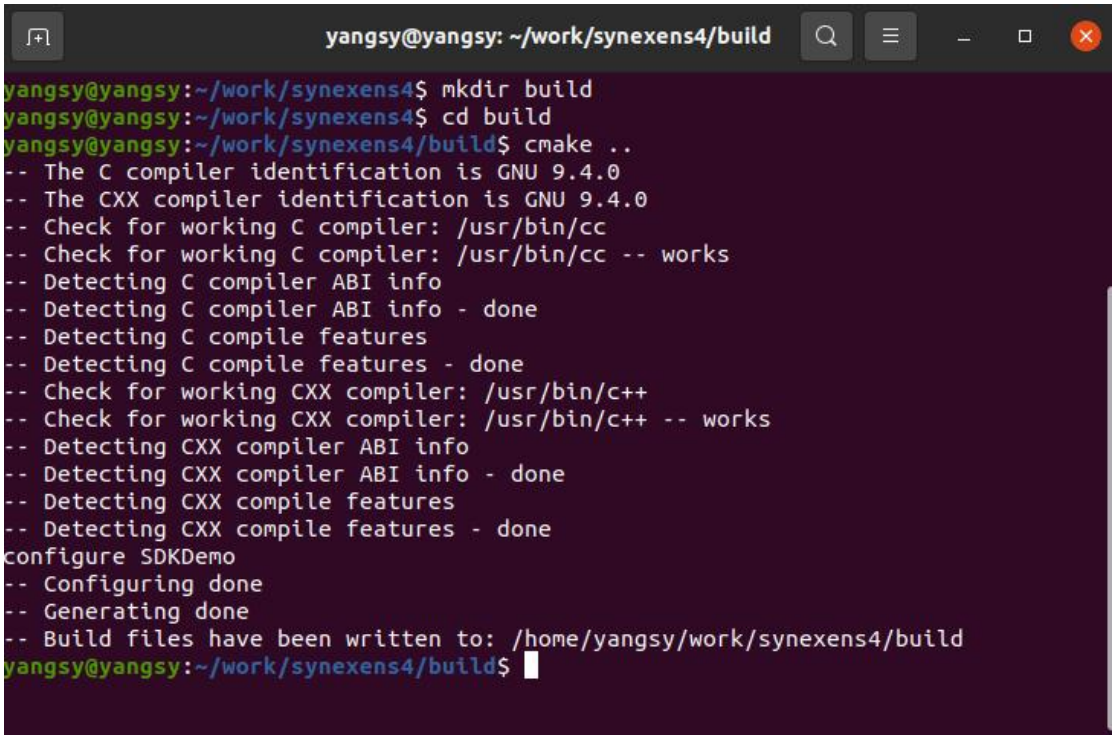
2.1.1. Install dependencies

```
sudo apt install libudev-dev

sudo apt install zlib1g-dev
```

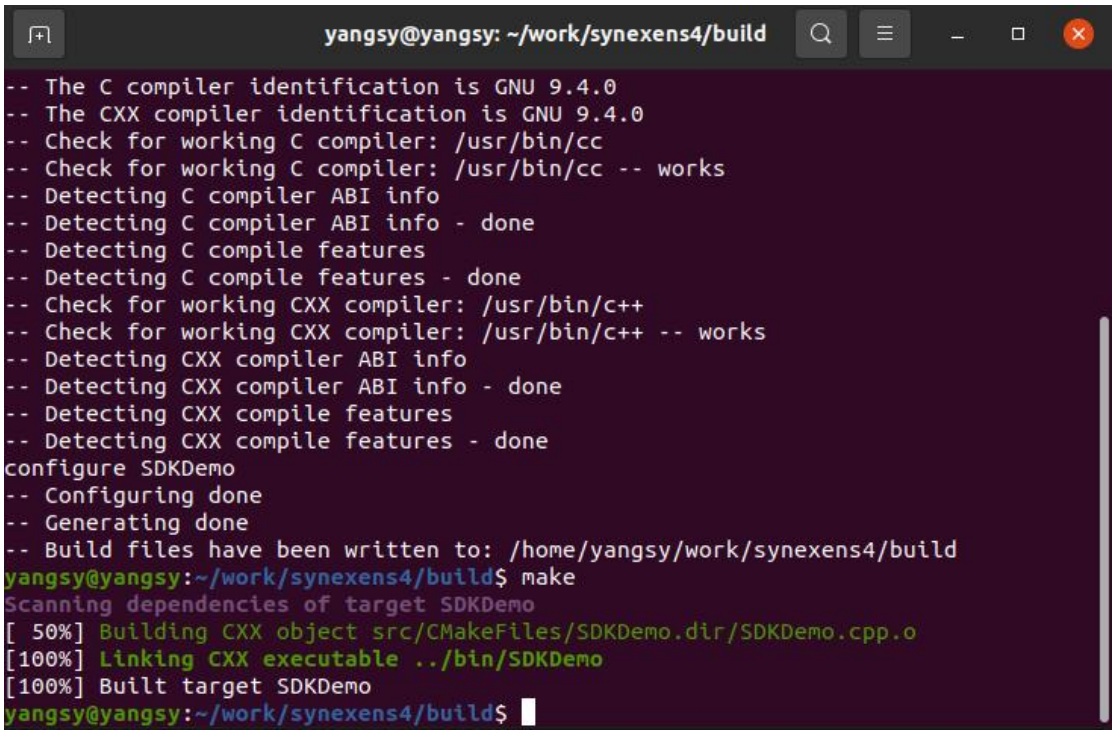
2.1.2. Writing CmakeLists.txt requires familiarity with CMake

```
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 the project compilation file



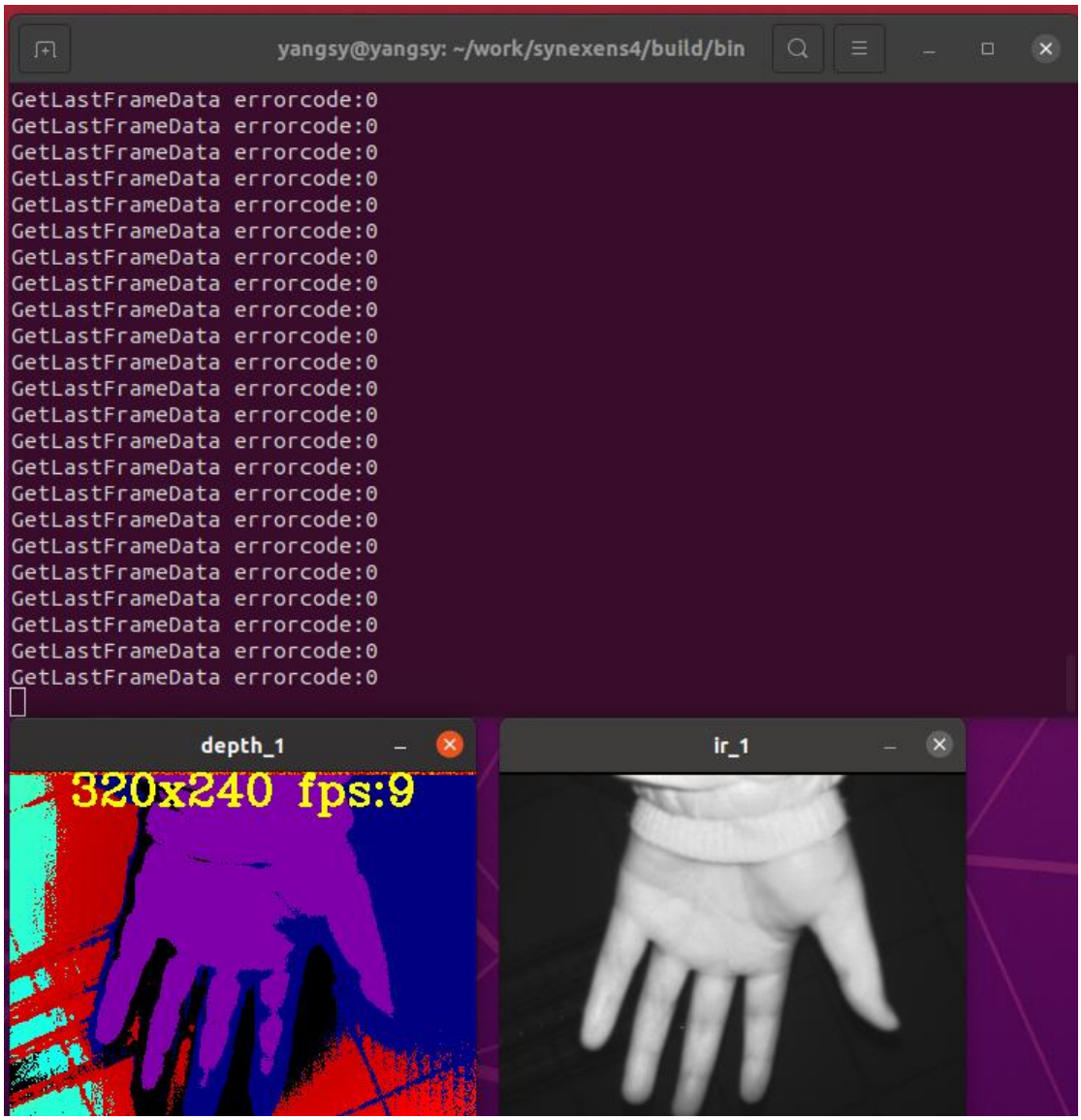
2.1.4. make compile



2.1.5. Execute the executable to test the effect

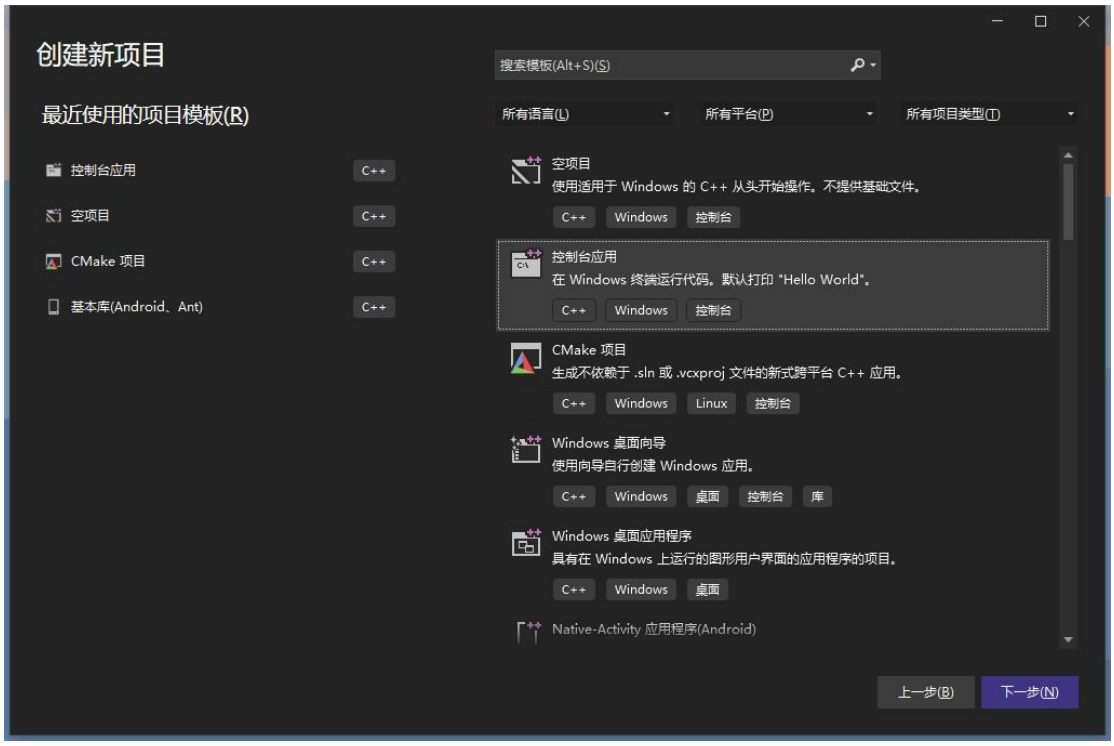
Before executing the program, LD_LIBRARY_PATH should be configured to find the library files that the program depends on. The example writes the run.sh script to facilitate the execution of the program.

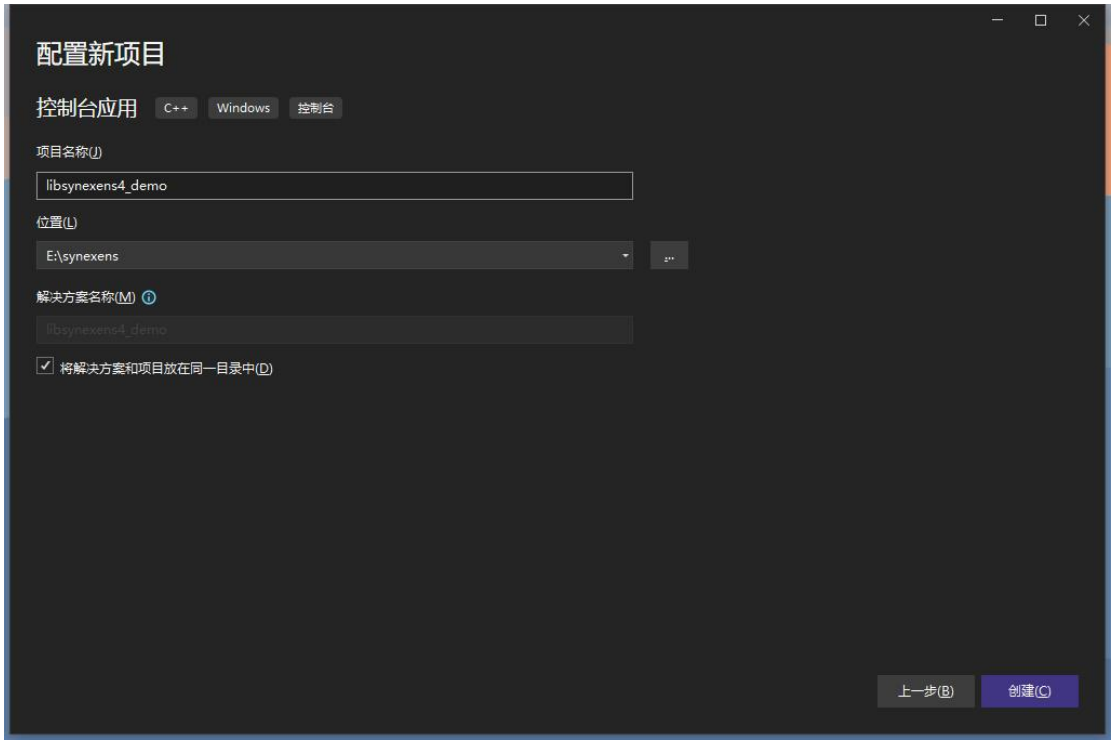
```
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:`pwd`
```



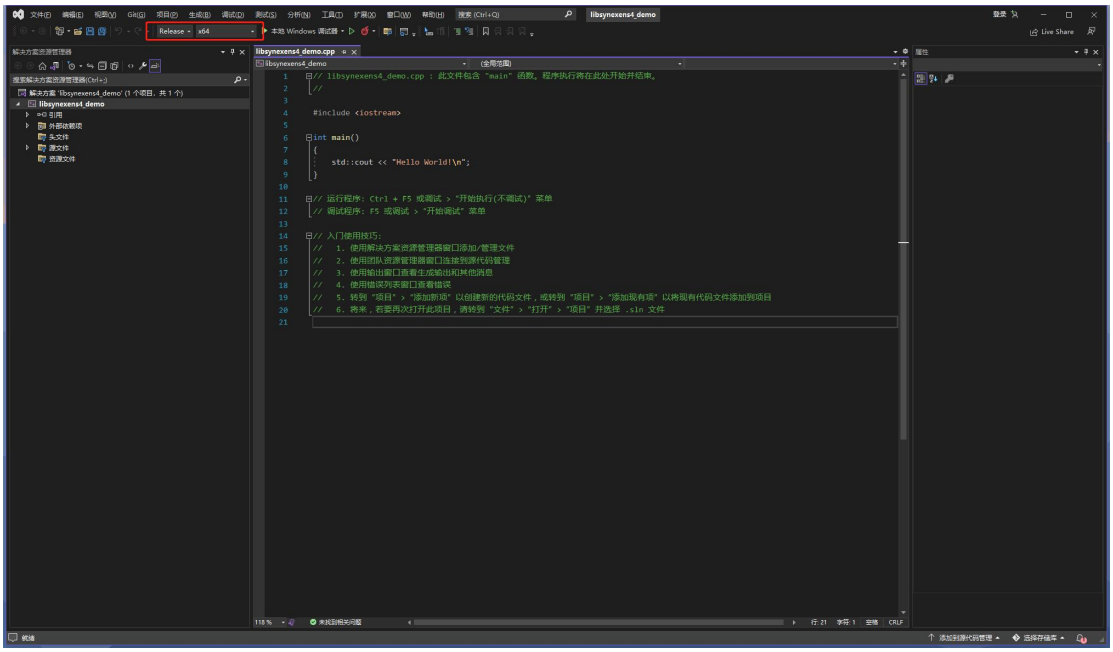
2.2. Windows environment configuration (vs2022 as an example)

2.2.1. Create a VS project

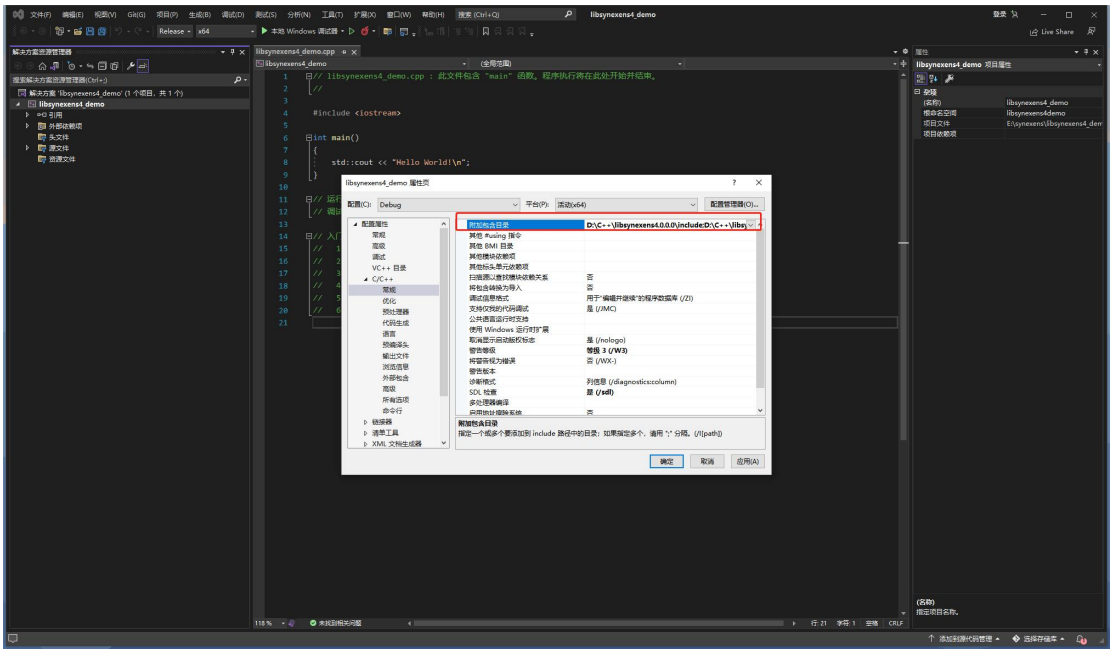


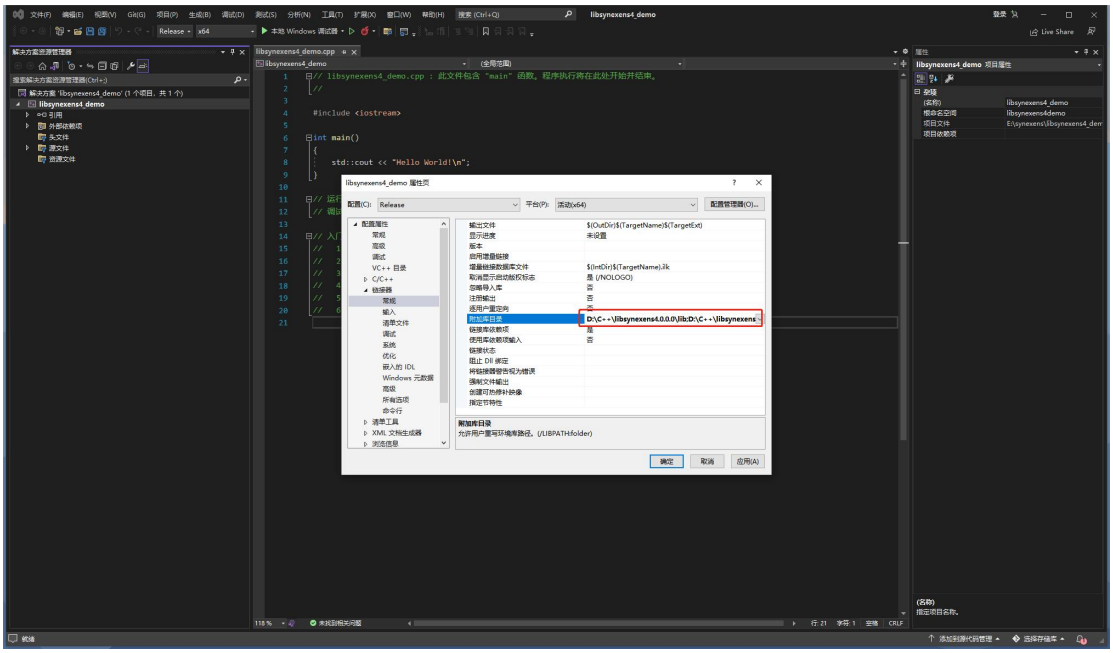


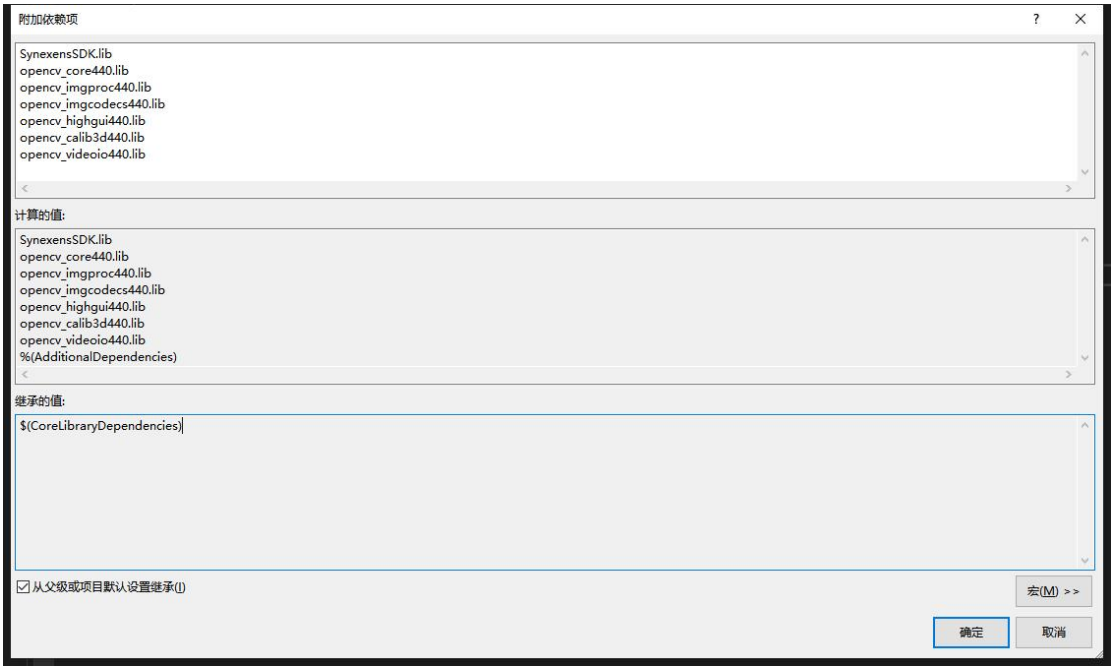
2.2.2. Select the solution that corresponds to the SDK as well as the platform

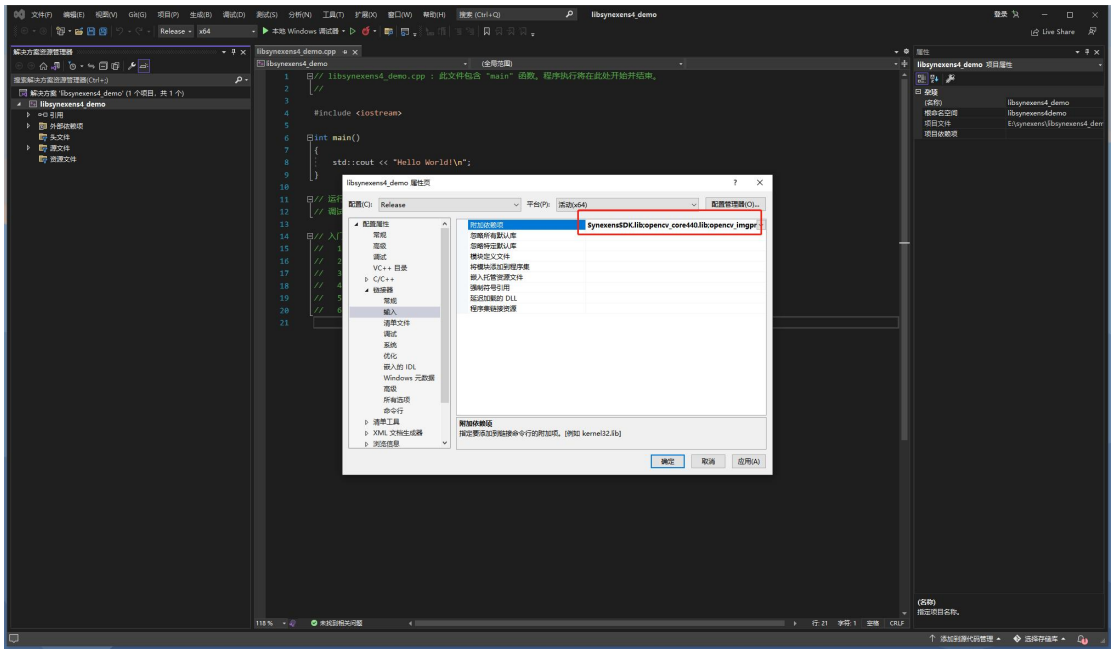


2.2.3. Configure the sdk header path, library path in the project properties






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

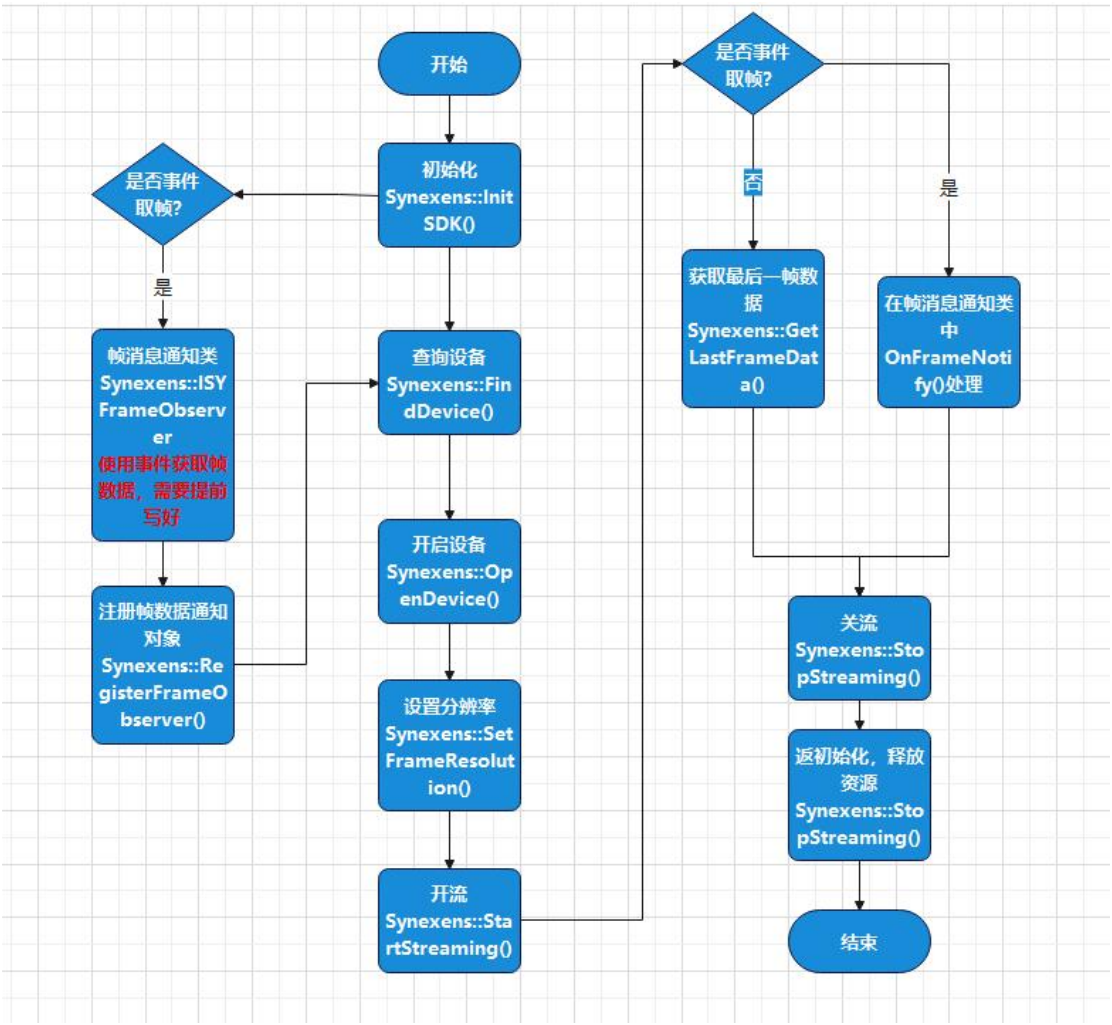


To run the demo, you need opencv dependency library. To develop by yourself, you do not need to rely on opencv

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

Note: to run demo, you need to configure the include path by yourself, and the dll file missing when running needs to be copied to the program running directory by yourself

2.3. SDK must call the process



3. API Overview

3.1. Global interface

3.1.1. GetSDKVersion

Description: Get the SDK version number

Syntax:

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: Uninitialize SDK and release resources

Syntax:

UnInitSDK();

3.1.4. RegisterErrorObserver

Description: Register error messages to notify object Pointers

Syntax:

RegisterErrorObserver(ISYErrorObserver* pObserver);

Parameters:

Parameter name	Description	in/out
pObserver	An error message notifies the object pointer	in

3.1.5. RegisterEventObserver

Description: Register event notification object Pointers

Syntax:

RegisterEventObserver(ISYEventObserver* pObserver);

Parameters:

Parameter name	Description	in/out
pObserver	Pobserver Event notifies object Pointers	in

3.1.6. RegisterFrameObserver

Description: Register data frames to notify object Pointers

Syntax:

RegisterFrameObserver(ISYFrameObserver* pObserver);

Parameters:

Parameter name	Description	in/out
pObserver	The dataframe informs the object pointer	in

3.1.7. UnRegisterErrorObserver

Unregistererrorobserver: A logout error message notifies object Pointers

Syntax:

UnRegisterErrorObserver(ISYErrorObserver* pObserver);

Parameters:

Parameter name	Description	in/out
pObserver	An error message notifies the object pointer	in

3.1.8. UnRegisterEventObserver

Description: Unregister event notification object pointer

Syntax:

UnRegisterEventObserver(ISYEventObserver* pObserver);

Parameters:

Parameter name	Description	in/out

pObserver	Pobserver Event notifies object Pointers	in

3.1.9. UnRegisterFrameObserver

Unregisterframeobserver: Unregister dataframe to notify object Pointers

Syntax:

UnRegisterFrameObserver(ISYFrameObserver* pObserver);

Parameters:

Parameter name	Description	in/out
pObserver	The dataframe informs the object pointer	in

3.1.10. FindDevice

Description: Find the device

Syntax:

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

Parameters:

Parameter name	Description	in/out
nCount	Number of devices	in/out
pDevice	Device information is	in/out

	allocated externally, and only nCount is obtained when pDevice is passed to nullptr	

3.1.11. OpenDevice

Description: Open the device

Syntax:

OpenDevice(const SYDeviceInfo& deviceInfo);

Parameters:

Parameter name	Description	in/out
deviceInfo	Device information	in

3.1.12. CloseDevice

Description: Closedevice

Syntax:

CloseDevice(unsigned int nDeviceID);

Parameters:

Parameter name	Description	in/out
nDeviceID	Device id	in

3.1.13. QueryDeviceSupportFrameType

Description: The query device supports the dataframe type

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 dataframe types, used only to return the number when pSupportType is empty, otherwise to check whether the number of pSupportType memory allocations matches	in/out
pSupportType	The supported dataframe type, which is externally allocated; pFrameType only gets nCount when passed to nullptr	in/out

3.1.14. QueryDeviceSupportResolution

Description: QueryDevicesupportreSolution

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 to return the number when pResolution is empty, otherwise to verify that the number of pResolution memory allocations matches	in/out
pResolution	Resolution type supported, externally allocated; pResolution only gets nCount when passed to nullptr	in/out

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 data streaming

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 data streaming

Syntax:

StopStreaming(unsigned int nDeviceID);

Parameters:

Parameter name	Description	in/out
nDeviceID	Device id	in

3.1.18. ChangeStreaming

Description: Switch data streaming

Syntax:

ChangeStreaming(unsigned int nDeviceID, SYStreamType streamType);

Parameters:

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

3.1.19. SetFrameResolution

Description: Set resolution (if data stream has been started, it will execute the operation process of closing stream -> Setting resolution -> re-opening stream)

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: Filter on state

Syntax:

GetFilter(unsigned int nDeviceID, bool& bFilter);

Parameters:

Parameter name	Description	in/out
nDeviceID	Device id	in
bFilter	Filter on state, true- filter turned on, false- filter not turned on	out

3.1.22. SetFilter

Description: Turn filter on/off

Syntax:

SetFilter(unsigned int nDeviceID, bool bFilter);

Parameter:

Parameter name	Description	in/out
nDeviceID	Device id	in
bFilter	Filter on state, true- filter turned on, false- filter not turned on	in

3.1.23. GetFilterList

Description: Get the filter 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	List of filters	in/out

3.1.24. SetDefaultFilter

Description: Sets the default filter

Syntax:

SetDefaultFilter(unsigned int nDeviceID);

Parameters:

Parameter name	Description	in/out
nDeviceID	Device id	in

3.1.25. AddFilter

Description: Add filter

Syntax:

AddFilter(unsigned int nDeviceID, SYFilterType filterType);

Parameters:

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

3.1.26. DeleteFilter

Description: Remove the filter

Syntax:

DeleteFilter(unsigned int nDeviceID, int nIndex);

Parameters:

Parameter name	Description	in/out
nDeviceID	Device id	in
nIndex	Index in the filter list	in

3.1.27. ClearFilter

Description: Clearfilter

Syntax:

ClearFilter(unsigned int nDeviceID);

Parameters:

Parameter name	Description	in/out
nDeviceID	Device id	in

3.1.28. SetFilterParam

Description: Set filter parameters

Syntax:

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: Get filter 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 filter parameters	in/out
pFilterParam	Filter parameters	in/out

3.1.30. GetMirror

Description: Get the horizontal mirror status

Syntax:

GetMirror(unsigned int nDeviceID, bool& bMirror);

Parameters:

Parameter name	Description	in/out
nDeviceID	Device id	in
bMirror	Horizontal mirror state, true- horizontal mirror is enabled, false- horizontal mirror is 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: Get the vertical flip status

Syntax:

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-	out

	vertical flip not enabled	

3.1.33. SetFlip

Description: Turn vertical flip on/off

Syntax:

SetMirror(unsigned int nDeviceID, bool bMirror);

Parameters:

Parameter name	Description	in/out
nDeviceID	Device id	in
bFlip	Vertical flip switch, true- turn on vertical flip, false- turn off vertical flip	in

3.1.34. GetIntegralTime

Description: Get the integraltime

Syntax:

GetIntegralTime(unsigned int nDeviceID, int& nIntegralTime);

Parameters:

Parameter name	Description	in/out
nDeviceID	Device id	in

nIntegralTime	Integration time	out

3.1.35. SetIntegralTime

Description: Set the integration time

Syntax:

SetIntegralTime(unsigned int nDeviceID, int nIntegralTime);

Parameters:

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

3.1.36. GetIntegralTimeRange

Description: Get the integraltimerange range

Syntax:

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	Max points in time	out

3.1.37. GetDistanceMeasureRange

Description: GetDistanceMeasurerange

Syntax:

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 value	out

3.1.38. GetDistanceUserRange

Description: Get the user ranging range

Syntax:

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

Parameters:

Parameter name	Description	in/out

nDeviceID	Device id	in
nMin	Minimum ranging range	out
nMax	The maximum value of the ranging range	out

3.1.39. SetDistanceUserRange

Description: Set the user ranging range

Syntax:

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

Parameters:

Parameter name	Description	in/out
nDeviceID	Device id	in
nMin	Minimum ranging range	in
nMax	The maximum value of the ranging range	in

3.1.40. GetDeviceSN

Description: Read the device sn number

Syntax:

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 only gets nLength when passed to nullptr	in/out

3.1.41. SetDeviceSN

Description: Write 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 number string pointer	in

3.1.42. GetDeviceHWVersion

Description: Reads the device firmware version number

Syntax:

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 string pointer, externally allocated memory, pstrHWVersion only gets nLength when passed to nullptr	in/out

3.1.43. GetDepthColor

Description: GetDepthColor corresponds to pseudo-color

Syntax:

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

Parameters:

Parameter name	Description	in/out
nDeviceID	Device id	in
nCount	Amount of data (nCount*2 bytes for memory space pDepth, nCount*3 bytes for pColor)	in
pDepth	Depth data	in
pColor	Depth corresponds to pseudo-color (24-bit RGB format)	in/out

3.1.44. GetDepthPointCloud

Getdepthpointcloud: GetDepthPointCloud Gets depthPointcloud data

Usage:

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

Parameters:

Parameter names	Description	in/out
nDeviceID	Device id	in
nWidth	width	in
nHeight	Height	in
pDepth	Depth data	in
pPointCloud	Depth corresponds to point	in/out

	cloud data, with externally allocated memory	
bUndistort	Crop logo, true- crop false- no crop	in

3.1.45. GetRGBD

Description: Get the RGBD

Syntax:

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);

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	The depth data in RGBD, memory allocated externally, data length is the same as the source RGB length	in/out
pTargetRGB	RGBD RGB data, externally allocated memory, data length is the same as the source RGB length	in/out

3.1.46. GetLastFrameData

Description: Get the latest frame of data

Syntax:

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

Parameters:

Parameter name	Description	in/out
nDeviceID	Device id	in
pFrameData	Last frame of data	in/out

3.1.47. Undistort

Dedistorting syntax

:

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	Psource to be dedistorted data pointer	in
nWidth	Image width	in
nHeight	Image height	in
bDepth	Whether depth data /RGB data	in
pTarget	The data pointer of dedistorted result is allocated externally, and the length of the data is the same as the length of the pointer of the data to be dedistorted	out

3.1.48. GetIntric

Description: Get camera parameters

Syntax:

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

3.2. Return parameter description

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

4. Filter setup instructions

4.1. Instructions for setting filter parameters

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);
```

Gaussian filter GAUSS

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);
```

Border filter EDGE

Example:

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

SPECKLE filter

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);
```

Sobel filter SOBEL

Example:

```
float threshold_value{ 0 };
threshold_value[0] = 150; // sobel_threshold
int num = 1;
SetFilterParam(nDeviceID, filterType, num , threshold_value);
```

Border filter 2 EDGE_MAD

Example:

```
float threshold_value{ 0 };
threshold_value[0] = 15; // EDGE_MAD_threshold
int num = 1;
SetFilterParam(nDeviceID, filterType, num , threshold_value);
```

Otsu filter OKADA

Example:

```
float threshold_value{ 0 };
threshold_value[0] = 15; // EDGE_MAD_threshold
int num = 1;
SetFilterParam(nDeviceID, filterType, num , threshold_value);
```

4.2. Description of the filter parameter range

Filter interface	Parameter 1-min	Parameter 1-max	Parameter 1 Recommended value	Parameter 2-min	Parameter 2-max	Parameter 2 Recommended value
AMPLITUD	0	100	6			

MEDIAN	3	5	3	0	5	1
EDGE	20	200	50			
SPECKLE	24	200	40	40	200	
GAUSS	3	5	3	0	5	1
EDGE_MAD	5	100	15			
SOBEL	20	300	150			
OKADA	10	100	10			

4.3. Description of the filtering call order

CS20: Median, Margin, blob, median

CS30: Median, border, and median are built into the front segment, and blob and median filters can be added to the back end.

5. Data structure definition description

5.1. Error code

```
enum SYErrorCode
    {
        // success
```

```
SYERRORCODE_SUCCESS = 0,

// fail

SYERRORCODE_FAILED = 1,

// device does not exist

SYERRORCODE_DEVICENOTEXIST = 2,

// device not open

SYERRORCODE_DEVICENOTOPENED = 3,

// unsupported resolution

SYERRORCODE_UNKOWNRESOLUTION = 4,

// device pointer handle is empty

SYERRORCODE_DEVICEHANDLEEMPTY = 5,

// Failed to format the device output data

SYERRORCODE_SETOUTPUTFORMATFAILED = 6,

// Failed to get video stream control pointer

SYERRORCODE_GETSTREAMCTRLFAILED = 7,

// Failed to start the video stream

SYERRORCODE_STARTSTREAMINGFAILED = 8,

// communication pointer is empty

SYERRORCODE_COMMUNICATEOBJECTEMPTY = 9,

// Invalid SN number

SYERRORCODE_UNKOWNSN = 10,

// String length overflow
```

```
SYERRORCODE_STRINGLENGTHOUTRANGE = 11,

// Invalid frame type

SYERRORCODE_UNKOWNFRAMETYPE = 12,

// Invalid device type

SYERRORCODE_UNKOWNDEVICETYPE = 13,

// device object pointer is empty

SYERRORCODE_DEVICEOBJECTEMPTY = 14,

// notify that the object pointer is empty

SYERRORCODE_OBSERVEREMPTY = 15,

// notify object not found

SYERRORCODE_OBSERVERNOTFOUND = 16,

// quantity overflow

SYERRORCODE_COUNTOUTRANGE = 17,

//UVC failed to initialize

SYERRORCODE_UVCINITFAILED = 18,

//UVC failed to find device

SYERRORCODE_UVCFINDDEVICEFAILED = 19,

// No data frame

SYERRORCODE_NOFRAME = 20,

// Failed to get program path

SYERRORCODE_GETAPPFOLDERPATHFAILED = 21,

// The video stream did not start
```

```
    SYERRORCODE_NOSTREAMING = 22,
    // algorithm pointer is empty
    SYERRORCODE_RECONSTRUCTIONEMPTY = 23,
};
```

5.2. Type of equipment

```
enum SYDeviceType
{
    // invalid
    SYDEVICETYPE_NULL = 0,
    //CS30 dual band
    SYDEVICETYPE_CS30_DUAL,
    //CS30 dual
    SYDEVICETYPE_CS30_SINGLE,
    //CS20 dual band
    SYDEVICETYPE_CS20_DUAL,
    //CS20 dual
    SYDEVICETYPE_CS20_SINGLE,
    //CS20_P
    SYDEVICETYPE_CS20_P,
    //CS40
    SYDEVICETYPE_CS40,
```

```
};
```

5.3. Data stream type

```
enum SYStreamType
    {
        // invalid
        SYSTREAMTYPE_NULL = 0,
        //RAW
        SYSTREAMTYPE_RAW,
        // depth
        SYSTREAMTYPE_DEPTH,
        //RGB
        SYSTREAMTYPE_RGB,
        // depth +IR
        SYSTREAMTYPE_DEPTHIR,
        // depth +RGB
        SYSTREAMTYPE_DEPTHRGB,
        // depth +IR+RGB
        SYSTREAMTYPE_DEPTHIRRGB,
        //RGBD(depth after mapping +RGB)
        SYSTREAMTYPE_RGBD,
        //RAW_RGB
```

```
        SYSTREAMTYPE_RAWRGB,
    };
```

5.4. Resolution enumeration

```
enum SYResolution
    {
        // invalid
        SYRESOLUTION_NULL = 0,
        / / 320 * 240
        SYRESOLUTION_320_240,
        / / 640 * 480
        SYRESOLUTION_640_480,
        / / 960 * 540
        SYRESOLUTION_960_540,
        / / 1920 * 1080
        SYRESOLUTION_1920_1080,
    };
```

5.5. Dataframe type

```
enum SYFrameType
    {
        // Invalid
```

```
    SYFRAMETYPE_NULL = 0,
    //RAW
    SYFRAMETYPE_RAW,
    // depth
    SYFRAMETYPE_DEPTH,
    //IR
    SYFRAMETYPE_IR,
    //RGB
    SYFRAMETYPE_RGB,
};
```

5.6. Support type

```
enum SYSupportType
{
    // invalid
    SYSUPPORTTYPE_NULL = 0,
    // depth
    SYSUPPORTTYPE_DEPTH,
    //RGB
    SYSUPPORTTYPE_RGB,
    //RGBD
    SYSUPPORTTYPE_RGBD,
```

```
};
```

5.7. Event type

```
enum SYEventType
    {
        // invalid
        SYEVENTTYPE_NULL = 0,
        // device connected
        SYEVENTTYPE_DEVICECONNECT,
        // device disconnects
        SYEVENTTYPE_DEVICEDISCONNECT,
    };
```

5.8. Filter type

```
enum SYFilterType
    {
        // invalid
        SYFILTERTYPE_NULL = 0,
        // median
        SYFILTERTYPE_MEDIAN,
        // amplitude
```

```
        SYFILTERTYPE_AMPLITUDE,
        // boundary
        SYFILTERTYPE_EDGE,
        // blob
        SYFILTERTYPE_SPECKLE,
        // large gold threshold
        SYFILTERTYPE_OKADA,
        // boundary 2
        SYFILTERTYPE_EDGE_MAD,
        // Gauss
        SYFILTERTYPE_GAUSS,
        // standby
        SYFILTERTYPE_EXTRA,
        // spare 2
        SYFILTERTYPE_EXTRA2,
    };
```

5.9. Equipment information

```
struct SYDeviceInfo
    {
        // Device unique ID
        unsigned int m_nDeviceID = 0;
```

```
        // device type
        SYDeviceType m_deviceType = SYDEVICETYPE_NULL;
    };
```

5.10. Event information

```
struct SYEventInfo
    {
        // Event type
        SYEventType m_eventType = SYEVENTTYPE_NULL;
        // event info data
        void* m_pEventInfo = nullptr;
        // data length
        int m_nLength = 0;
    };
```

5.11. Data frame information

```
struct SYFrameInfo
    {
        // Frame type
        SYFrameType m_frameType = SYFRAMETYPE_NULL;
        // Height (pixels)
        int m_nFrameHeight = 0;
```

```
        // width (pixels)
        int m_nFrameWidth = 0;
    };
```

5.12. Data frame

```
struct SYFrameData
    {
        // number of frames
        int m_nFrameCount = 0;
        // frame information
        SYFrameInfo* m_pFrameInfo = nullptr;
        // frame data
        void* m_pData = nullptr;
        // data length
        int m_nBuffferLength = 0;
    };
```

5.13. Point cloud data structure

```
struct SYPointCloudData
    {
        //X
        float m_fltX = 0.f;
```

```
        //Y
        float m_fltY = 0.f;
        //Z
        float m_fltZ = 0.f;
    };
```

5.14. Camera parameter struct

```
struct SYIntrinsics
    {
        // Lens view
        float m_fltFOV[2];
        // Distortion coefficient
        float m_fltCoeffs[5];
        // focal length in the x direction
        float m_fltFocalDistanceX;
        // focal length in y direction
        float m_fltFocalDistanceY;
        // The imaging center point in the x direction is cx
        float m_fltCenterPointX;
        // The imaging center point in the y direction is cy
        float m_fltCenterPointY;
        // width
```

```
        int m_nWidth;
        // height
        int m_nHeight;
    };
```

6. FQA

f: The dll cannot be found when running under win

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

f: Linux runtime prompt uvc_open:-3

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

f: A select() timeout. Error occurred

a: The device timeout may be caused by insufficient power supply and usb bandwidth. It is recommended to connect the external hub power supply or connect to a different usb interface

f: The noise point is relatively large

a: You can set the filter parameter through the GUI, remember the filter parameter after getting the desired effect, and add it to the SDK

f: xxx library could not be found

a: run the program through run.sh to make sure that the path to the library that run.sh imported is correct, or install the dependency library under usr/lib

f: cs40 cs20-p device not found

a: Make sure the device starts and the device stays on the same network segment as the IPC. Make sure you can ping the ip address of the device

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

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

7. About device connection

Note: There is no limit in the SDK to how many devices you can connect to. In theory, you can connect to an unlimited number of devices. The specific number of devices that can be connected to the IPC depends on the hardware support of the IPC. At present, after testing, there are a few things to pay attention to:

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

2. An independent usb on an industrial computer can connect to two devices at most, which needs to be adjusted according to different models. The actual independent usb interface (some industrial computers may have multiple usb ports, but these usb may use the same bandwidth and power supply)

3. At present, two CS20 and one CS30 have been successfully connected to the industrial personal computer.

Disclaimer

The device application information and other similar content described in this publication is for your convenience only and may be superseded by updated information. It is your own responsibility to ensure that the application complies with the technical specifications. We

make no representations or warranties, express or implied, written or oral, statutory or otherwise, including, but not limited to, representations or warranties regarding its use, quality, performance, merchantability or fitness for a particular purpose. The Company disclaims any liability for such information and for consequences arising out of its use. This product may not be used as a critical component in a life support system without written approval from the Company.