RERVISION USB Camera

Windows SDK

开发手册

2024-9-12 Ver.A

锐尔威视

www.rervision.cn

版本更新说明

版本号	修改日期	修改说明
A	2024-9-12	初稿建立

目 录

1. 概述

1.1 编写目的

此文档介绍了锐尔威视的 Windows SDK（以下简称 SDK）的 API 用法

开发人员可在 Windows 系统下使用 SDK 开发视频软件，不用关心 USB Camera 的底层实现，只需调用少量 API，即可实现图像解码预览、图像参数设置、拍照、获取视频数据流等功能，在此基础上可开发视频编码、网络推流、图像传输、图像算法等功能。

此 SDK 适用于锐尔威视的所有型号 USB Camera

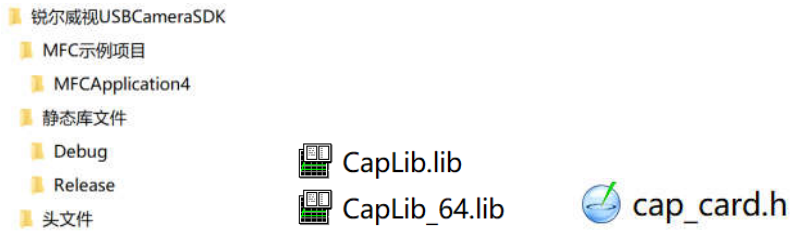
1.2 锐尔威视 SDK 简介

锐尔威视 SDK，基于 Microsoft 的 DirectShow 创建，底层调用系统的 DirectShow 接口，适用于 Win7、Win8、Win10、Win11 等系统。

DirectShow 是 DirectX 的组件之一，DirectX 是 Microsoft 提供的一套在 Windows 平台上开发高性能图形、声音、输入、输出等功能的编程接口。 这其中的 DirectShow 提供了应用程序从适当的硬件中捕捉和预览音、视频的能力。应用程序可以立刻显示捕捉的数据（预览），或是保存到文件中。

SDK 使用 MFC 开发，开发工具推荐用 Visual Studio 2022

SDK 内容包含静态库文件、头文件、MFC 示例项目源码



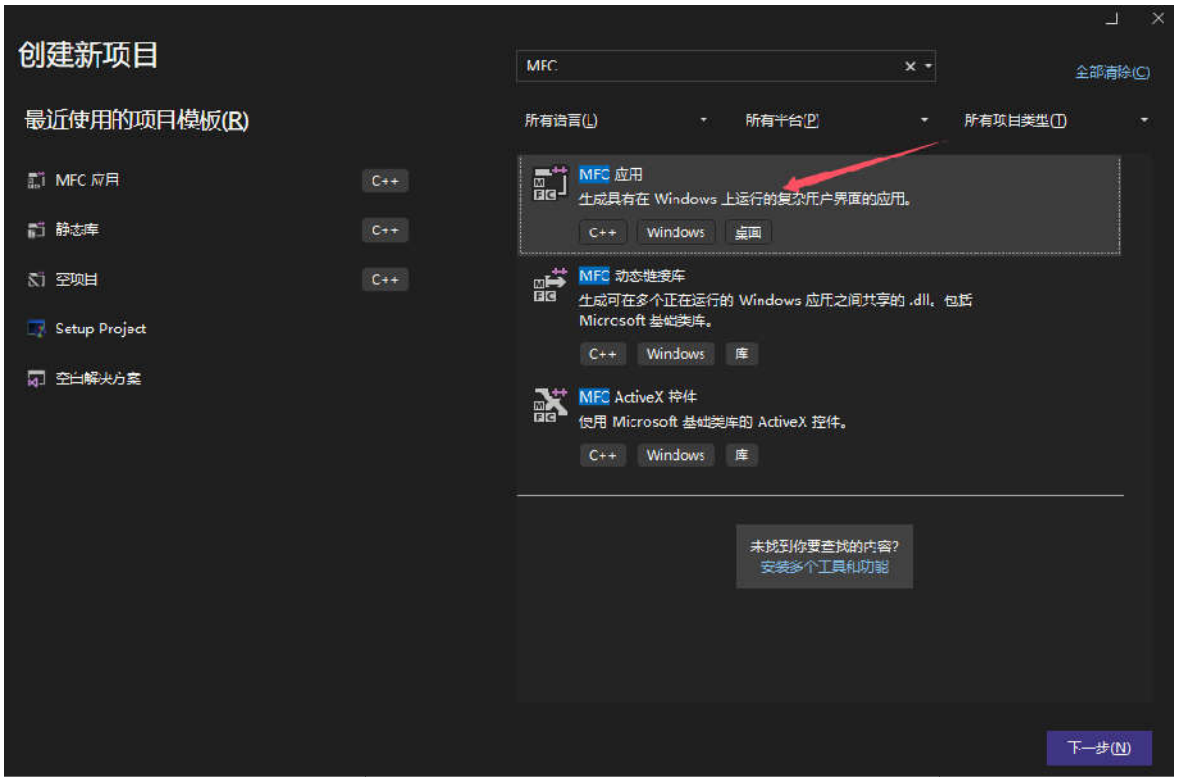
静态库文件分为 Debug 和 Release 版本，每个版本里包含 CapLib.lib(32 位)和 CapLib_64.lib（64 位），用户根据版本需求，将相应的 lib 替换到工程中

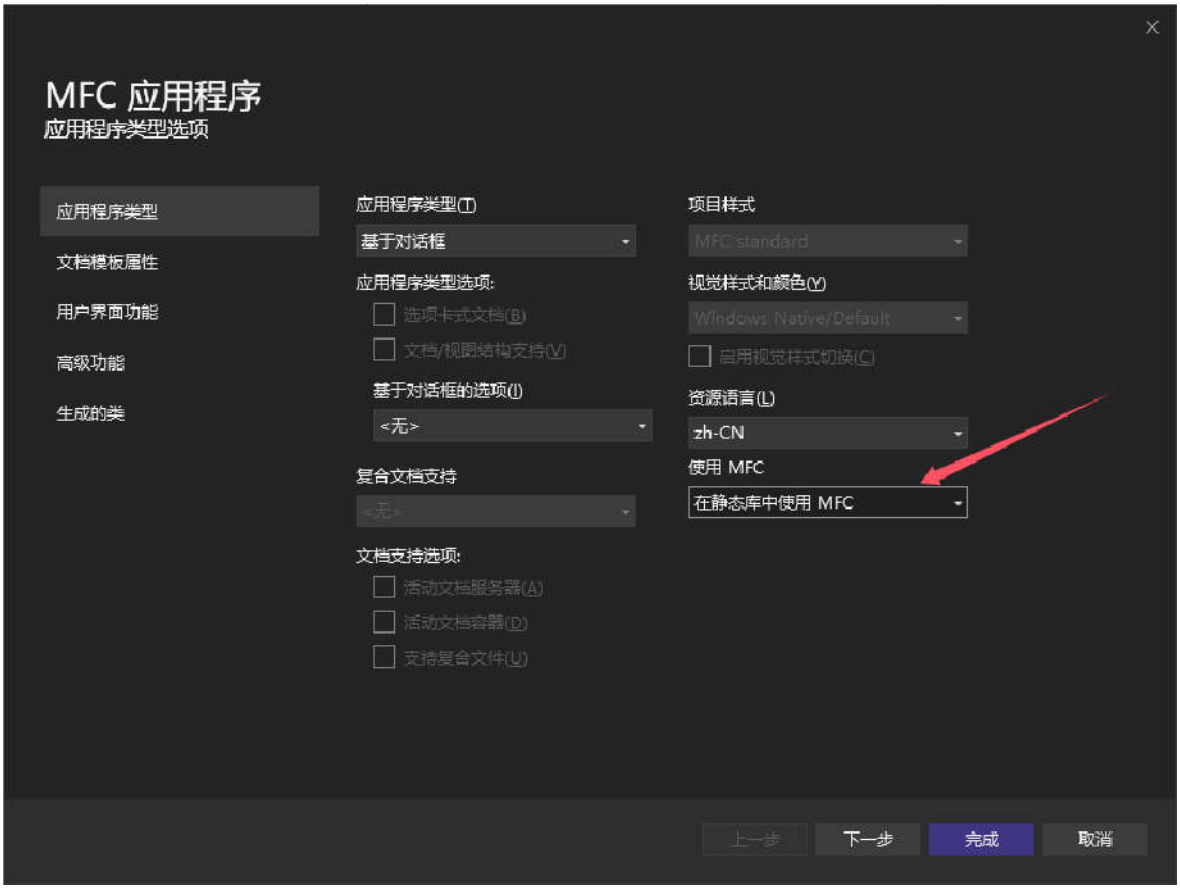
头文件是 cap_card.h，需要包含到工程中

MFC 示例项目是锐尔威视开发的 USBCamera 软件 Demo，包含此 SDK 文档中涉及的所有 API 用法的源码和图形界面源码

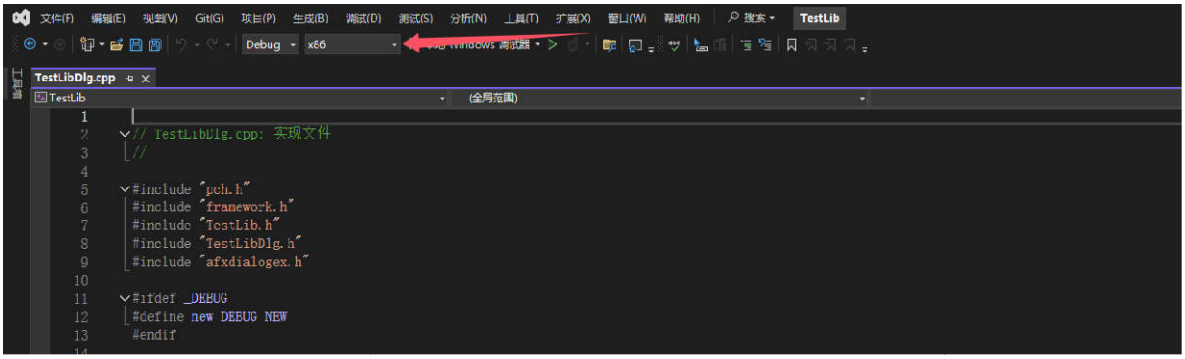
2. 搭建开发环境

创建 MFC 应用，并设置在静态库中使用 MFC

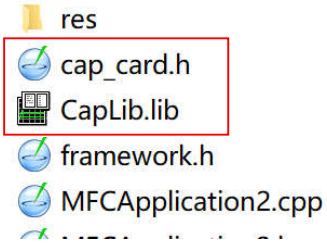




选择“x86”，即 32 位模式

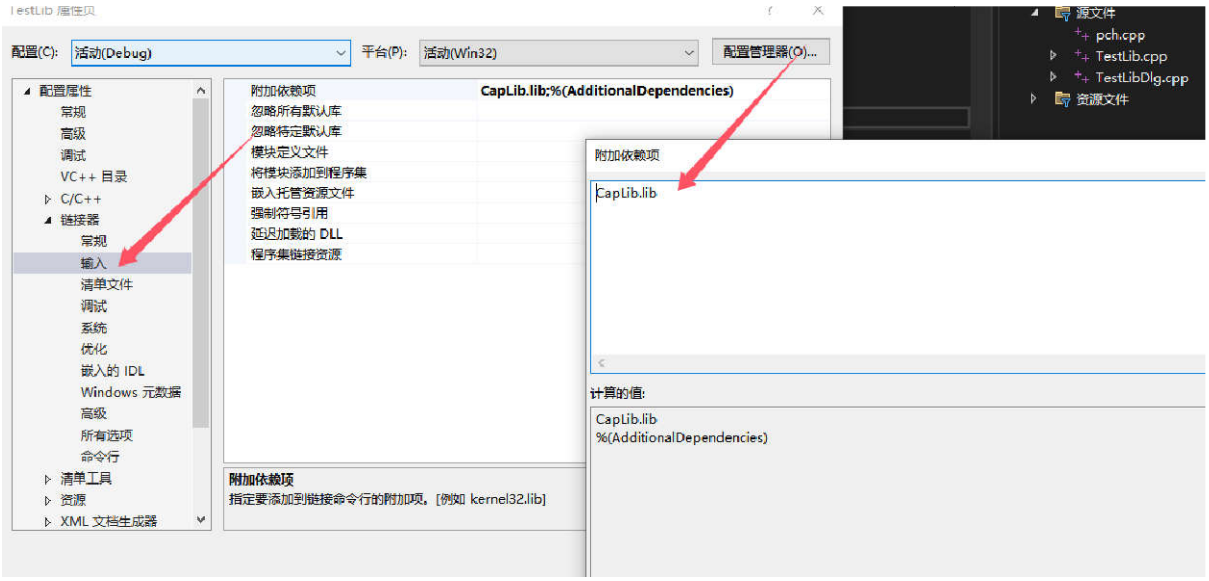


将 cap_card.h 和(Debug 中的)CapLib.lib 文件放入项目目录



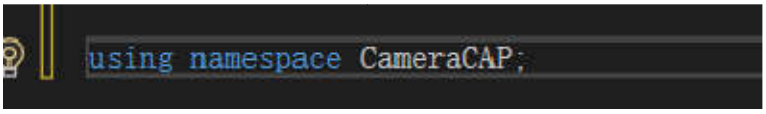
项目中包含 cap_card.h 头文件并添加依赖库 CapLib.lib

添加依赖库的第二种方式



设置运行库模式为 MT 模式

使用命名空间 CameraCAP



如果项目设置为其他模式，如 Release/64 位，应选择对应的静态库文件

3. 结构体说明

3.1 图像编码类型

```
typedef enum class VideoColorSpace_t
{
    Unknow,
    Mjpg,
    Yuy2
}vcVideoColorSpace;
```

相机的视频格式：MJPEG 和 YUV

3.2 图像信息

```
typedef struct vcVideoFormat_t
{
    vcVideoColorSpace vcs; // 图像编码类型
    uint16_t w, h; // 图像宽高
    float fps;  // 帧率
    GUID vcs_guid;
    vcVideoFormat_t() {
        vcs = vcVideoColorSpace::Unknow;
        w = 0;
        h = 0;
        fps = 0.0;
        memset(&vcs_guid, 0, sizeof(vcs_guid));
    }
}vcVideoFormat;
```

相机编码类型、分辨率、帧率

3.3 设备信息

```
typedef struct CaptureDevice_t
{
    char vidpid[9];
    wchar_t devName[256];
    CaptureDevice_t() {
        memset(vidpid, 0, 9);
        wmemset(devName, 0, 256);
    }
}CaptureDevice;
```

相机的设备名称、VID、PID

3.4 相机属性值 ID

```
//在头文件 strmif.h 中
typedef enum tagVideoProcAmpProperty
{
    VideoProcAmp_Brightness    = 0,
    VideoProcAmp_Contrast  = ( VideoProcAmp_Brightness + 1 ) ,
    VideoProcAmp_Hue    = ( VideoProcAmp_Contrast + 1 ) ,
    VideoProcAmp_Saturation    = ( VideoProcAmp_Hue + 1 ) ,
    VideoProcAmp_Sharpness = ( VideoProcAmp_Saturation + 1 ) ,
    VideoProcAmp_Gamma  = ( VideoProcAmp_Sharpness + 1 ) ,
    VideoProcAmp_ColorEnable   = ( VideoProcAmp_Gamma + 1 ) ,
    VideoProcAmp_WhiteBalance  = ( VideoProcAmp_ColorEnable + 1 ) ,
    VideoProcAmp_BacklightCompensation    = ( VideoProcAmp_WhiteBalance + 1 ) ,
    VideoProcAmp_Gain   = ( VideoProcAmp_BacklightCompensation + 1 )
} VideoProcAmpProperty;
```

相机的图像参数，包括：亮度、对比度、色调、饱和度、清晰度、伽马、白平衡、逆光对比、增益、电力线频率

3.5 相机控制属性值 ID

```
enum Property
{
    PowerLineFrequency = 13,
    Exposure = 14,
    Zoom = (Exposure + 1),
    Focus = (Zoom + 1),
    Pan = (Focus + 1),
    Tilt = (Pan + 1)
};
```

相机的控制参数，包括：曝光、焦点、缩放、全景、倾斜

属性名称	对应 ID
亮度	0
对比度	1
色调	2
饱和度	3
清晰度	4
伽马	5
白平衡	7
逆光对比	8
增益	9
电力线频率	13

曝光	14
缩放	15
焦点	16
全景	17
倾斜	18

3.6 相机属性值范围

```
class VideoPropertyRange1 {
public:
    long Min; // 最小值
    long Max; // 最大值
    long Step; // 调节幅度
    long Default; // 默认值
    long Flags; // 是否自动调节，1：自动， 2：手动

    VideoPropertyRange1() : Min(0), Max(0), Step(0), Default(0), Flags(0) {}

    //
    VideoPropertyRange1(long min, long max, long step, long def, long flags)
        : Min(min), Max(max), Step(step), Default(def), Flags(flags) {}
};
```

最小值、最大值、调节幅度、默认值

3.7 相机属性值

```
class VideoProperty1 {
public:
    long Value; // 属性值
    long Flags; // 是否自动调节，1：自动， 2：手动

    VideoProperty1() : Value(0), Flags(0) {}
    VideoProperty1(long value, long flags) : Value(value), Flags(flags) {}
};
```

对应每个相机属性 ID 的具体数值

4. 接口函数说明

4.1 InitLib

函数原型： bool InitLib();
功能：初始化 API 资源
参数：无
返回值：成功返回 true，失败返回 false
调用说明：调用其他 API 之前必须先调用该函数初始化

4.2 UnInitLib

函数原型： bool WinAPI UnInitLib();
功能：释放所有 API 资源
参数：无
返回值：成功返回 true，失败返回 false
调用说明：退出程序时调用

4.3 EnumCameras

函数原型： bool EnumCameras(CaptureDevice devs[], uint32_t arrSize_devs, uint32_t* devCount);
功能：枚举所识别的设备
参数：devs[]：所获取的设备数组 arrSize_devs：devs[]数组的长度 devCount：枚举得到的设备数量
返回值：枚举成功返回 true，枚举失败返回 false，无设备返回 false
调用说明：无

4.4 OpenCamera

函数原型： bool OpenCamera(uint32_t devIndex, HWND preWndHandle = 0, CapGrabber grab = 0, void* ptrClass = 0);
功能： 打开指定设备，即为指定设备创建视频流捕获器
参数： devIndex：设备序号 preWndHandle：将要预览的视频流 HWND 窗口，为 NULL 则不预览视频 grab：捕获图像帧数据的回调函数 ptrClass：传入对象指针，可从 grab 中访问该对象成员
返回值：成功返回 true，失败返回 false
调用说明：无

4.5 CloseCamera

函数原型： bool WinAPI CloseCamera(uint32_t devIndex);
功能： 关闭指定设备，销毁指定设备的视频流捕获器
参数： devIndex：设备序号
返回值：成功返回 true，失败返回 false
调用说明：无

4.6 RunCamera

函数原型： bool WinAPI RunCamera(uint32_t devIndex);
功能： 启动视频流捕获器，开始捕获视频
参数： devIndex：设备序号
返回值：成功返回 true，失败返回 false
调用说明：无

4.7 StopCamera

函数原型： bool WinAPI StopCamera(uint32_t devIndex);
功能： 停止捕获视频 参数： devIndex：设备序号 返回值：成功返回 true，失败返回 false 调用说明：无

4.8 GetFormats

函数原型： bool WinAPI GetFormats(uint32_t devIndex, vcVideoFormat formats[], uint16_t* formatNum, uint16_t arrSize_Formats);
功能： 获取设备所支持的格式，分辨率，帧率 参数： devIndex：指定的设备序号 formats： 指定设备所支持的格式数组 formatNum：指定设备所支持的格式的数量 arrSize_Formats：传入的 formats 长度 返回值：成功返回 true，失败返回 false 调用说明：无

4.9 CurFormat

函数原型： bool WinAPI CurFormat(uint32_t devIndex, uint16_t* formatIndex);
功能： 获取设备当前的格式 参数： devIndex：指定设备序号 formatIndex：指定设备当前的格式序号 返回值：成功返回 true，失败返回 false，如果设备未 Open 将返回 false 调用说明：无

4.10 SetFormat

函数原型： bool WinAPI SetFormat(uint32_t devIndex, uint16_t formatIndex, HWND preWndHandle = 0);
功能： 设置指定设备格式
参数： devIndex：指定设备序号
formatIndex：指定格式序号
preWndHandle：预览窗口，若没有可传入 0 或 NULL
返回值：成功返回 true，失败返回 false
调用说明：无

4.11 Getprops

函数原型： bool Getprops(uint32_t devIndex, long prop, VideoProperty1& b);
功能： 获取指定设备的指定属性值
参数： devIndex：指定设备序号
prop：指定属性值序号
b： 得到的属性值，参考 VideoProperty1 结构体
返回值：成功返回 true，失败返回 false
调用说明：无

4.12 Getrange

函数原型： bool Getrange(uint32_t devIndex,long prop, VideoPropertyRange1 &b);
功能： 获取指定属性值的范围
参数： devIndex：指定设备序号
prop：指定属性值序号
b： 得到的属性值范围，参考 VideoPropertyRange1 结构体
返回值：成功返回 true，失败返回 false
调用说明：无

4.13 Setprops

函数原型： bool Setprops(uint32_t devIndex, long props, int value,int flag);
功能： 设置相机的指定属性值
参数： devIndex：指定设备序号
prop：指定属性值序号
value： 将要设置的值，应该在该属性的范围内
flag：设置相机是否自动调节该属性值，只适用于曝光，白平衡，焦点
返回值：成功返回 true，失败返回 false
调用说明：无

4.14 SetFrameRate

函数原型： bool WinAPI SetFrameRate(uint32_t devIndex, uint16_t fps);
功能： 设置相机帧率
参数： devIndex：指定设备序号
fps：所设置的帧率
返回值：成功返回 true，失败返回 false
调用说明：超出相机所支持的最大帧率时无效

4.15 getFrameRate

函数原型： int getFrameRate(uint32_t devIndex);
功能： 获取设备实时帧率
参数： devIndex：正在启动的设备 ID
返回值：指定设备的单当前帧率
调用说明：需要设备处于正在启动状态

4.16 getBitrate

函数原型： int getBitrate(uint32_t devIndex);

功能： 获取设备实时码率

参数： devIndex：正在启动的设备 ID

返回值：指定设备的单当前码率

调用说明：需要设备处于正在启动状态

4.17 CapGrabber

函数原型： typedef int(WinAPI* CapGrabber)(double sampleTime, uint8_t* buf, long bufSize, void* ptrClass);

功能：用于捕获视频帧的回调函数，由用户实现，使用 OpenCamera 时传入

参数： sampleTime: 从捕获器开启到捕获到当前视频帧所经过的时间(单位秒)
buf： 捕获的视频帧数据地址
bufSize：视频帧的数据大小，单位字节
ptrClass：OpenCamera 中所传入的对象指针

返回值：无

调用说明：每捕获到一帧图像就被调用一次

4.18 SaveImage

函数原型： bool SaveImage(BYTE* imageBuf, long imgSize, vcVideoFormat format, wchar_t* photo_path);

功能： 将图像帧数据保存为 jpg 文件

参数： imageBuf：图像帧数据
imgSize： 图像帧数据的大小，单位为字节
format： 图像的格式，参考 vcVideoFormat 结构体
photo_path：照片保存路径

返回值：成功返回 true，失败返回 false

调用说明：需要拍照时，调用该接口

4.19 decompress_jpeg_to_yuy2

函数原型： bool decompress_jpeg_to_yuy2(uint8_t* jpeg_buffer, long jpeg_size, uint8_t* yuy2_buffer, long& size, int& width, int& height);

功能： MJPEG 图像数据解码成 yuy2 格式的数据

参数： jpeg_buffer：mjpeg 图像帧数据
jpeg_size： mjpeg 图像帧数据的大小
yuy2_buffer：用于保存解压后的 yuy2 数据，用户需提前申请好空间，大小为 width*height*2;
size： yuy2 数据大小，单位字节
width： 图像的宽
height： 图像的高

返回值：成功返回 true，失败返回 false

调用说明：当需要拿到解码后的数据时，调用该接口

4.20 UpdateVideoWind

函数原型： bool UpdateVideoWind(uint32_t devIndex, HWND preWndHandle);

功能： 更新视频窗口尺寸

参数： devIndex：指定设备序号
preWndHandle：更新后的视频窗口

返回值：成功返回 true，失败返回 false

调用说明：当需要更新视频窗口尺寸时，调用该接口

4.21 ClearVideoWind

函数原型： bool ClearVideoWind(uint32_t devIndex);

功能： 清除视频窗口的画面

参数： devIndex：指定设备序号

返回值：成功返回 true，失败返回 false

调用说明：在关闭相机或切换视频格式时，调用该接口