中国大恒（集团）有限公司北京图像视觉技术分公司

DirectShow
用户使用说明书

版本：V1.0.0
发布日期：2018-08-01

DAHENG IMAGING 大恒图像

网　　站：http://www.daheng-imaging.com

销售信箱：sales@daheng-imaging.com

销售热线：010-82828878 转 8068

支持信箱：support@daheng-imaging.com

支持热线：400-999-7595

目录

1. 安装 DirectShow 程序

1.1. 安装 DirectShow 程序

在大恒相机软件光盘中，使用 DX_Setup_en.exe 安装 DirectShow 程序时，在安装过程中会出现注册界面，在“Enter the number of DX registrations:”编辑框中输入所需注册的相机数量（最多可选择注册 32 台相机），点击 Register 按钮进行注册，如图 1-1：

图 1-1

注册完成后，GxDirectShowRegister.exe 自动退出。验证相机是否注册成功，运行 graphedt.exe，选择菜单栏中的 Graph，点击下拉列表中的 Insert Filters。

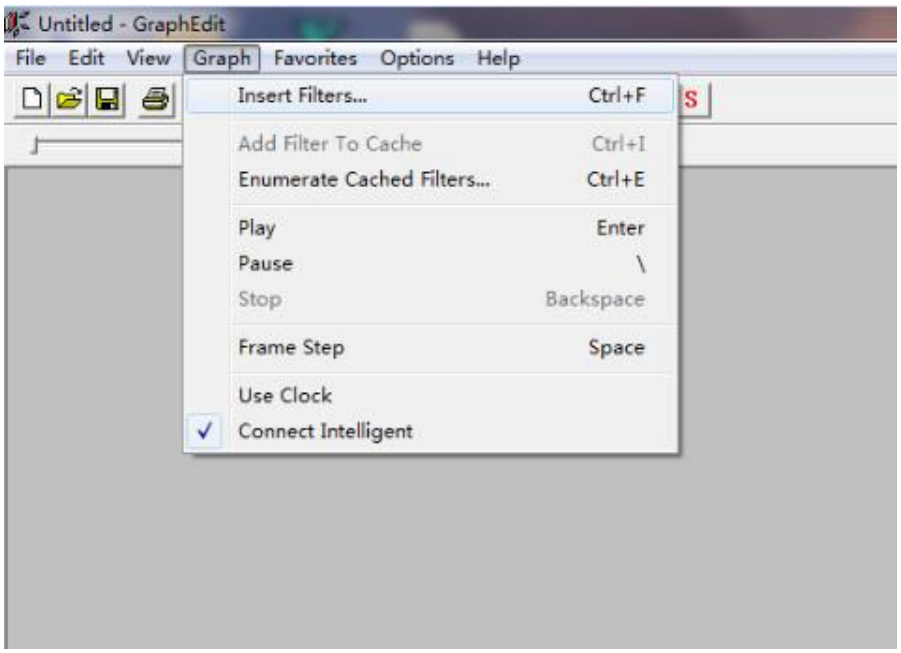

图 1-2

在弹出的窗口中选择 Video Capture Sources 查看注册的相机个数是否正确。

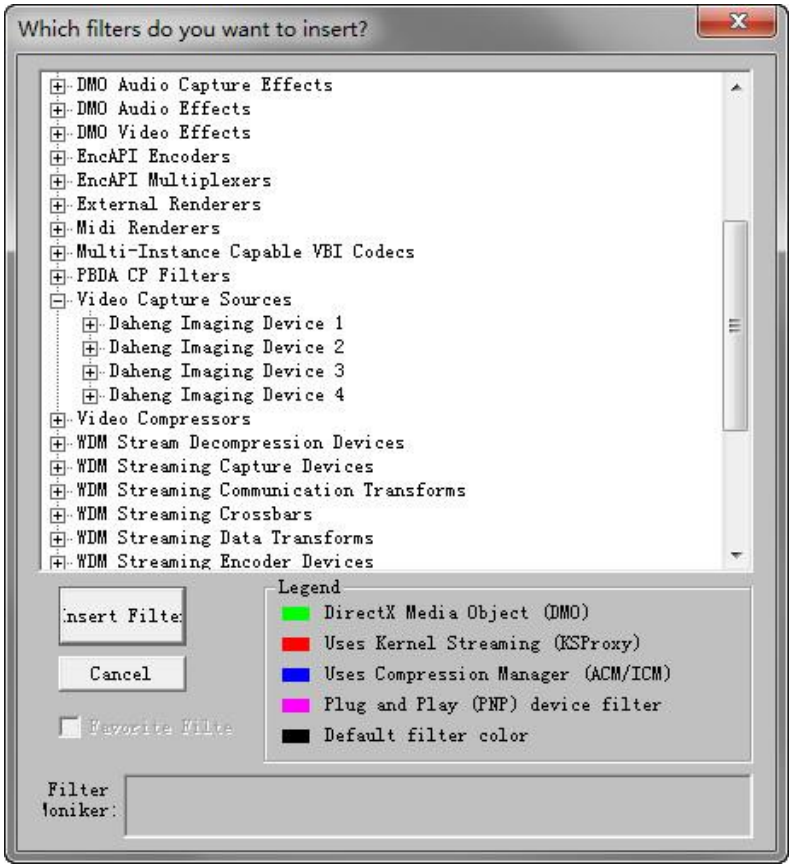

图 1-3

1.2. 卸载 DirectShow 程序

运行 unins000.exe 卸载 DirectShow 程序，在卸载过程中出现注销界面，如图 1-4 所示。在“Enter the number of DX registrations:”编辑框中输入要注销的相机数量（最多可选择注销 32 台相机），点击 Unregister 按钮进行注销：

图 1-4

注销完成后，GxDirectShowRegister.exe 自动退出。验证相机是否注销成功，当注销的相机数量小于已注册的相机数量时，从第一台相机开始执行注销。运行 graphedt.exe，选择菜单栏中的 Graph，点击下拉列表中的 Insert Filters。

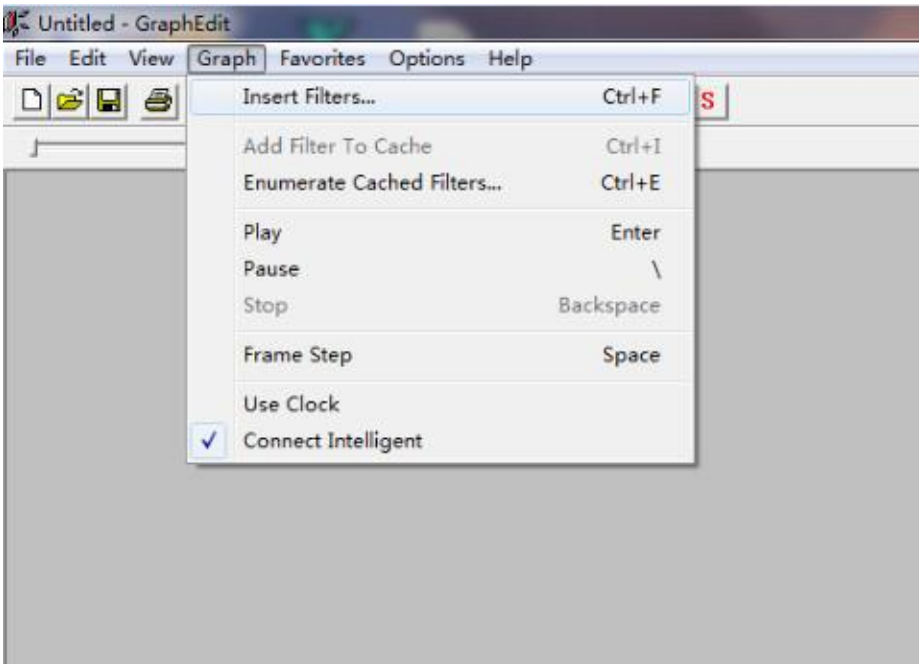

图 1-5

在弹出的窗口中选择 Video Capture Sources 查看注销的相机个数是否正确。

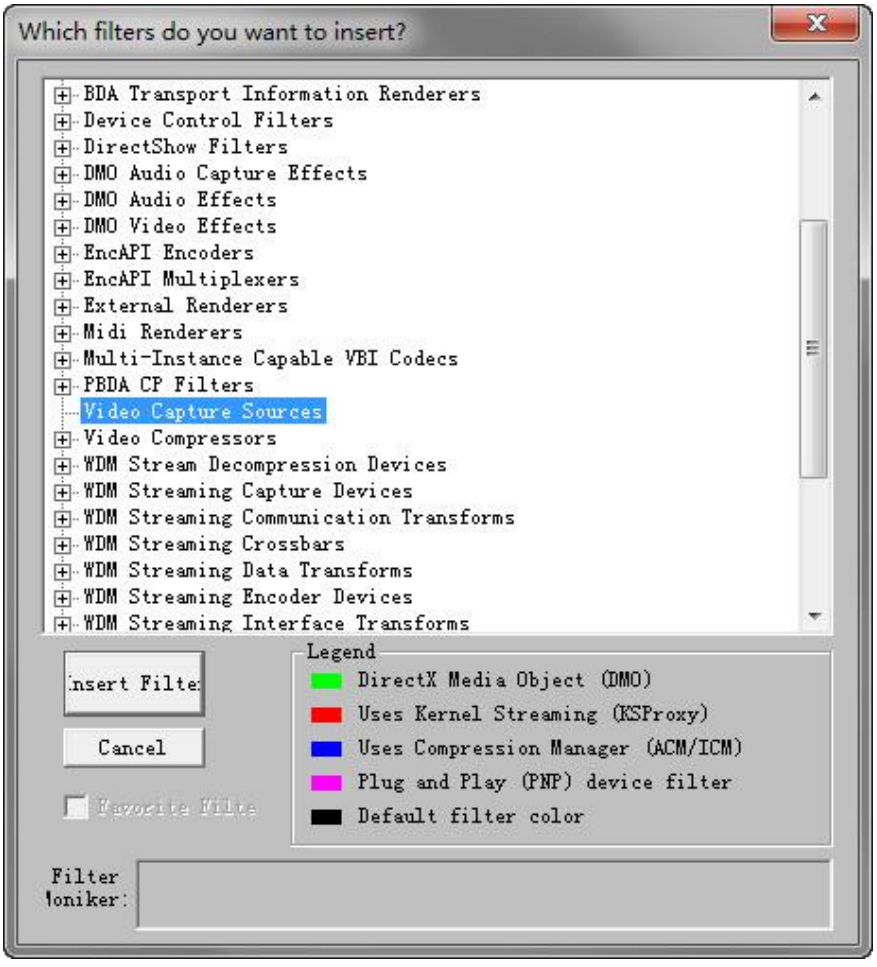

图 1-6

2. 环境配置

2.1. OpenCV3.0.0 配置

2.1.1. 安装 OpenCV3.0.0

下载 Opencv3.0.0（链接：

https://sourceforge.net/projects/opencvlibrary/files/opencv-win/3.0.0/ ）

运行 opencv-3.0.0.exe，选择导出目录，点击“Extract”按钮，导出 OpenCV 源码。

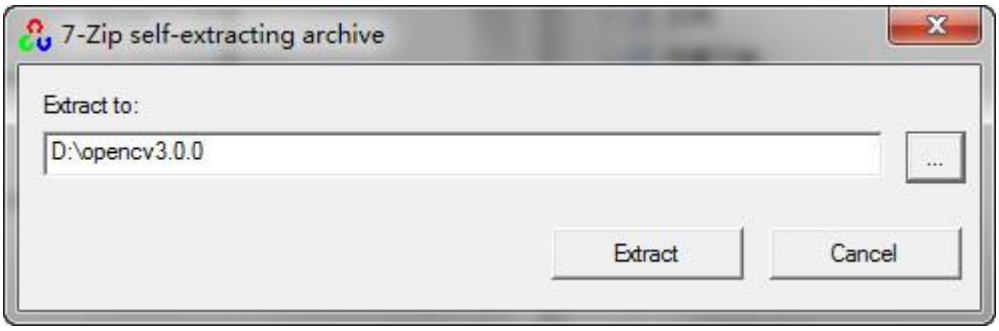

图 2-1

2.1.2. 使用 OpenCV 已经生成的库在 Visual Studio 工程中配置 OpenCV3.0.0

以在 Visual Studio 2010 下配置 32 位 Debug 版程序为例，添加头文件路径：

D:\opencv3.0.0\opencv\build\include;

D:\opencv3.0.0\opencv\build\include\opencv;

D:\opencv3.0.0\opencv\build\include\opencv2;

如图 2-2 所示：

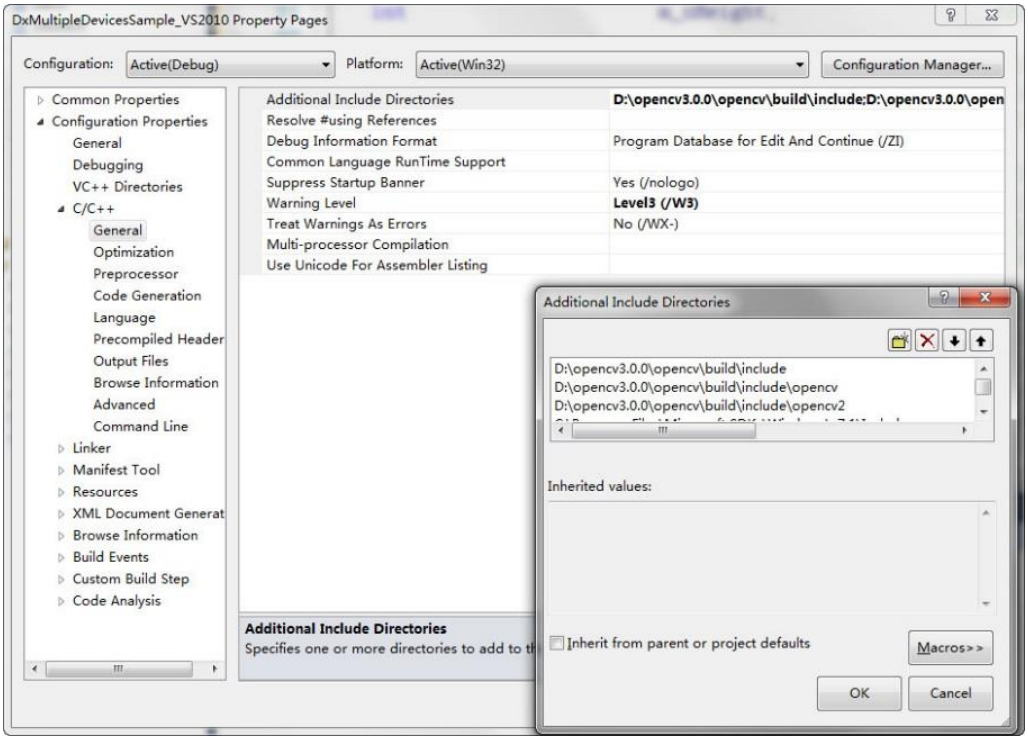

图 2-2

配置 lib 文件路径：

D:\opencv3.0.0\opencv\build\x86\vc11\lib;

D:\opencv3.0.0\opencv\build\x86\vc11\staticlib;

如图 2-3 所示：

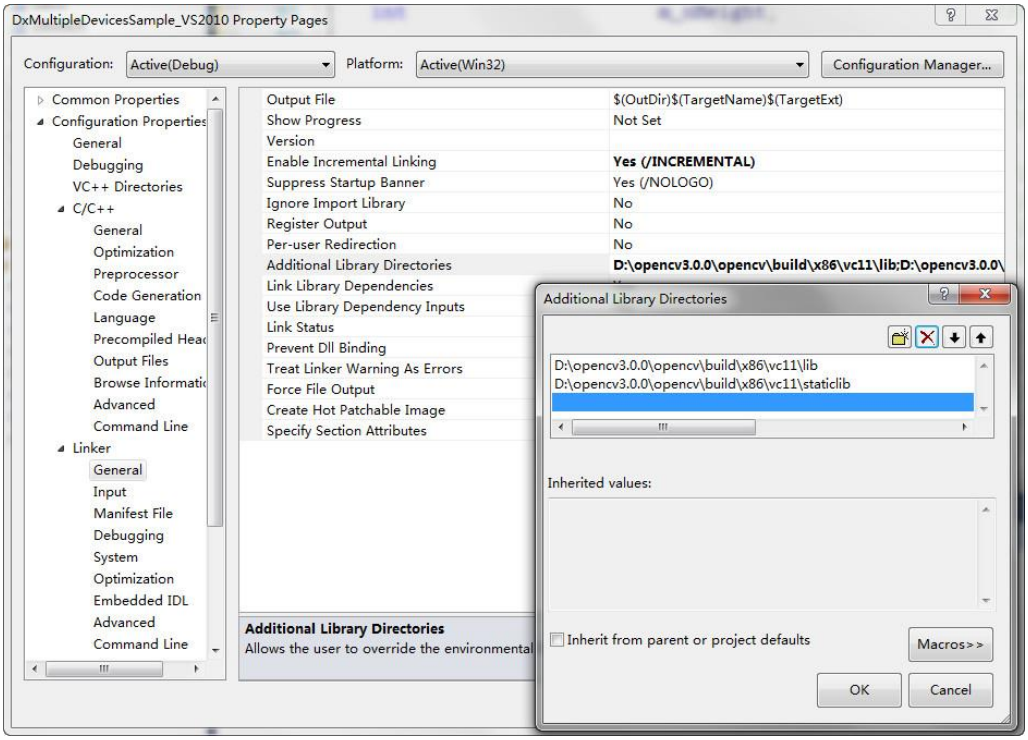

图 2-3

Debug 模式添加对路径下的 lib 文件的引用，具体文件为

opencv_ts300.lib;

opencv_world300.lib;

如图 2-4 所示：

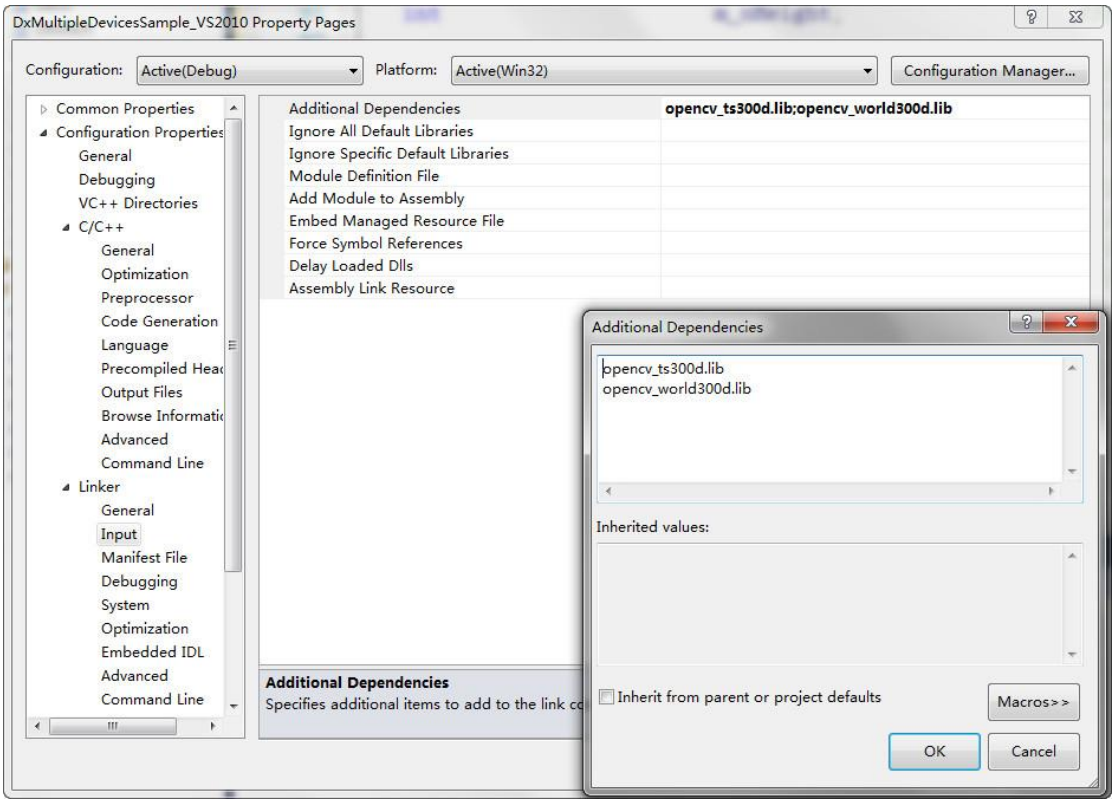

图 2-4

2.1.3. 使用 CMake 方式在 Visual Studio 工程中配置 OpenCV3.0.0

在无法使用 vc11 或 vc12 配置 Visual Studio 时，使用 CMake 编译 OpenCV 源码， 在解压后的 CMake 目录中，找到\bin\cmake-gui.exe，点击运行，选择 OpenCV 源码目录，选择生成二进制文件存放目录。如图 2-5 所示：

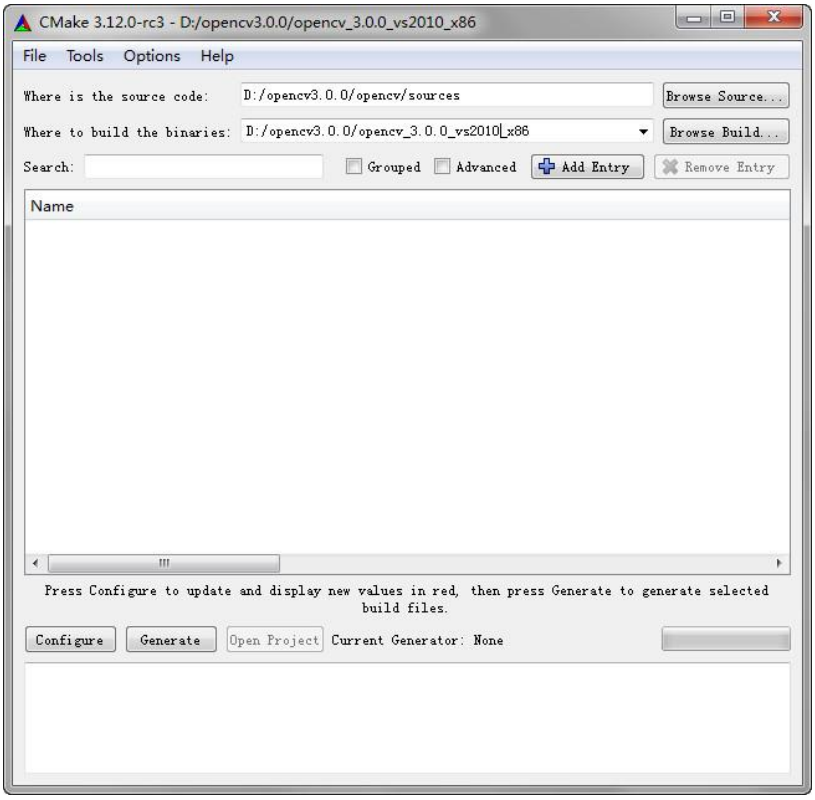

图 2-5

点击 Configure 按钮。以使用 Visual Studio2010 编译 32 位源码为例，如图 2-6 所示：

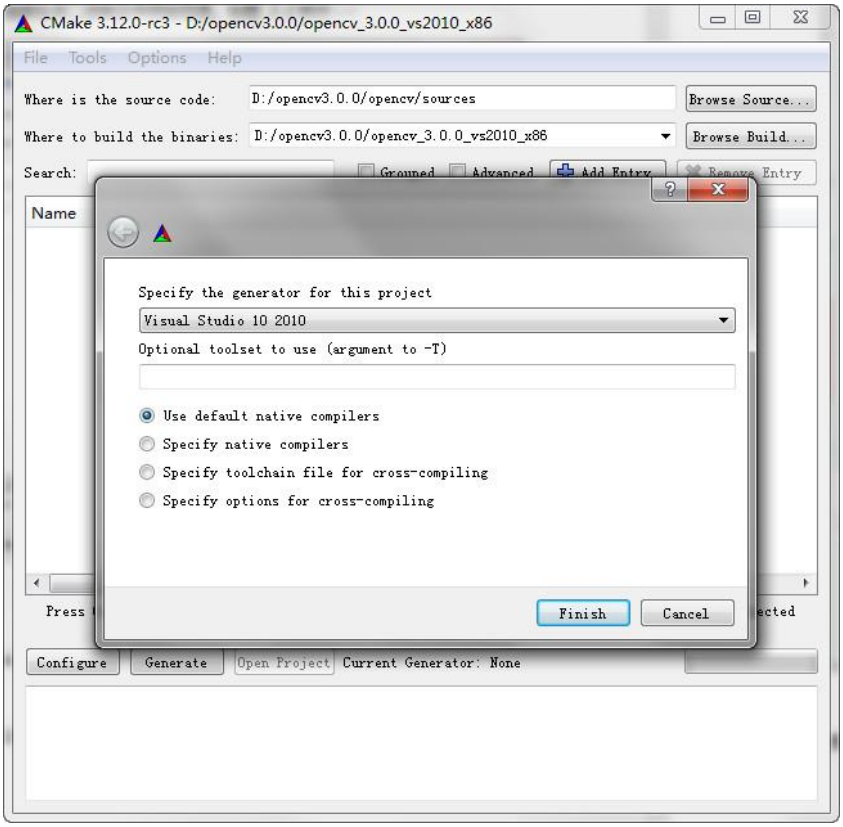

图 2-6

配置过程中需要从网络上下载资源，所以请保持网络通畅。如图 2-7 所示，为正在配置过程中：

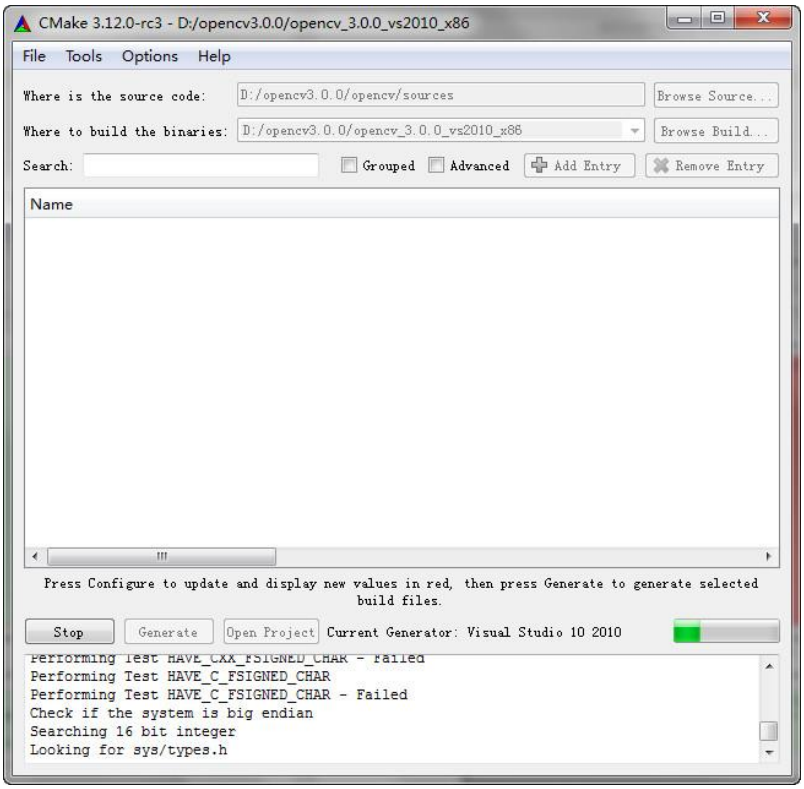

图 2-7

进度条执行完成，出现 Configuring done 字样，第一次源码配置完成，如图 2-8 所示：

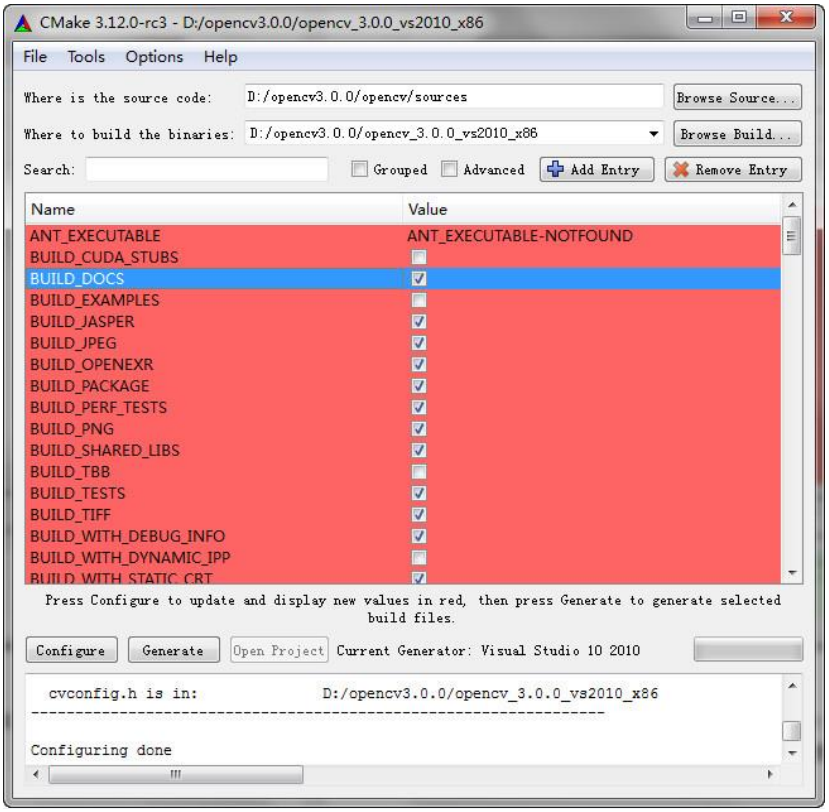

图 2-8

第一次配置完成后，在图中所示标红的文件中选择构建的文件，然后再次点击 Configure 按钮进行第二次配置。进度条执行完成后，出现 Configure done 字样，并且红色标记的文件变为正常。如图 2-9 所示：

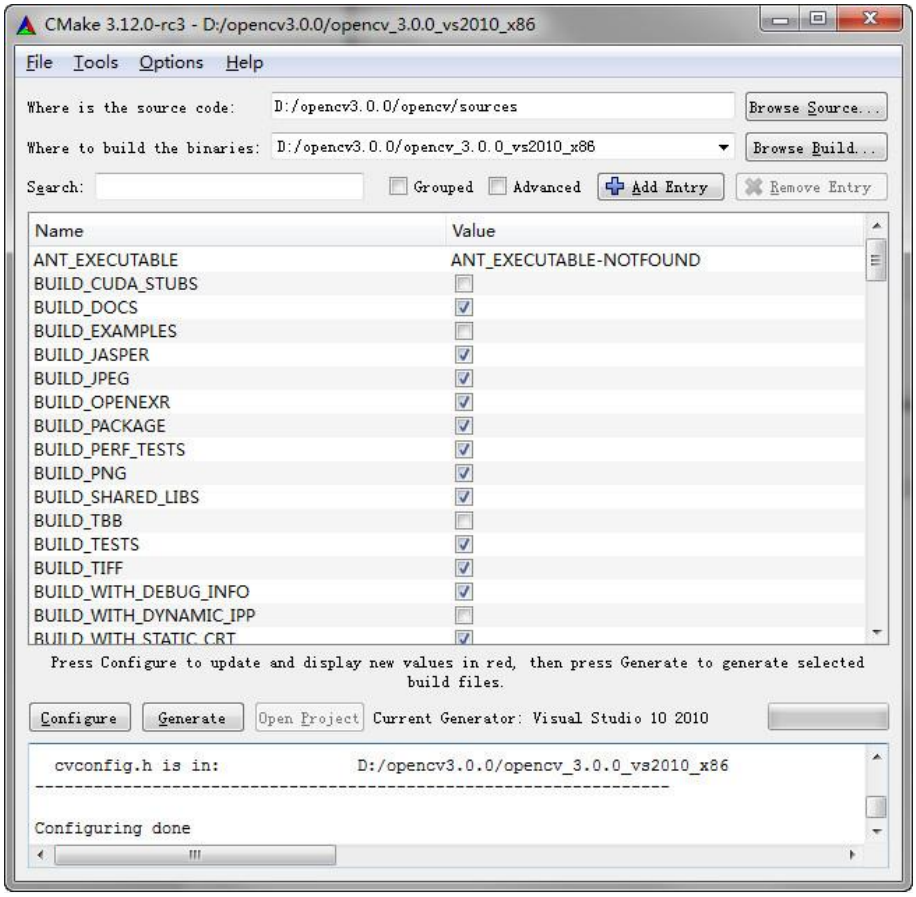

图 2-9

点击 Generate 按钮，生成可供 Visual Studio 2010 使用的解决方案，即 OpenCV.sln。如图 2-10 所示：

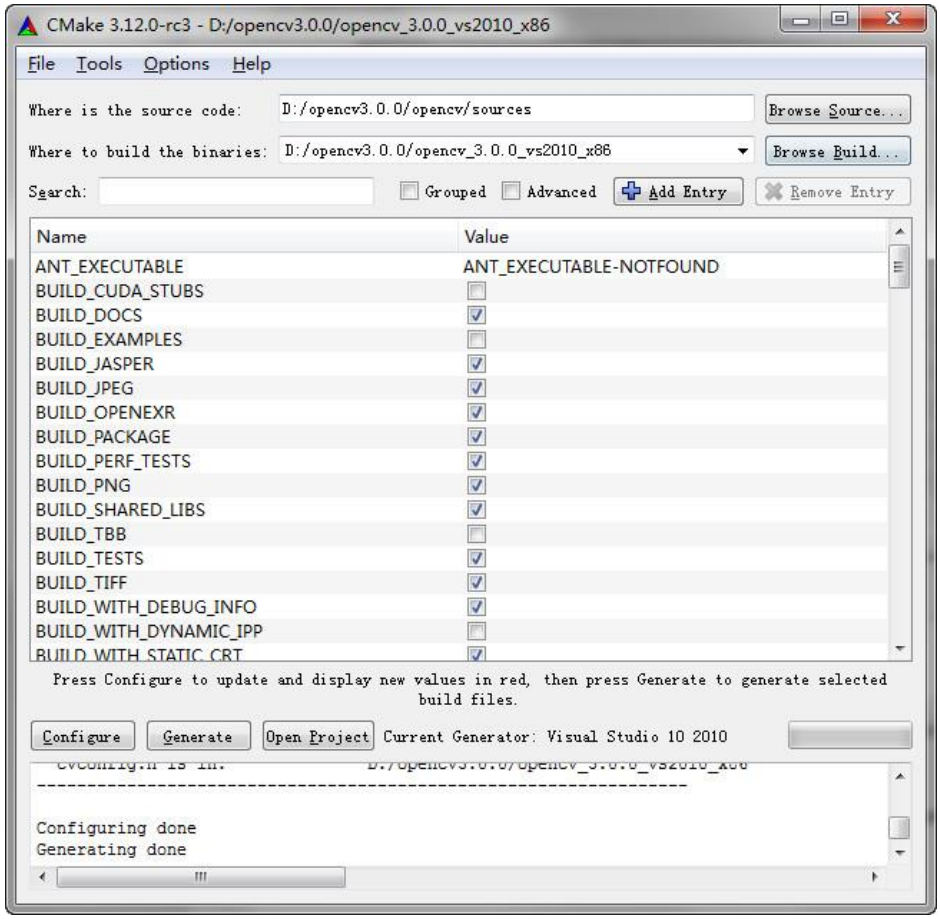

图 2-10

点击 Open Project 按钮，则会打开 OpenCV.sln，在 Visual Studio 2010 中生成解决方案。编译成功后，在\opencv_3.0.0_vs2010_x86\lib\Debug\目录下生成 OpenCV3.0.0 依赖库，如图 2-11 所示：

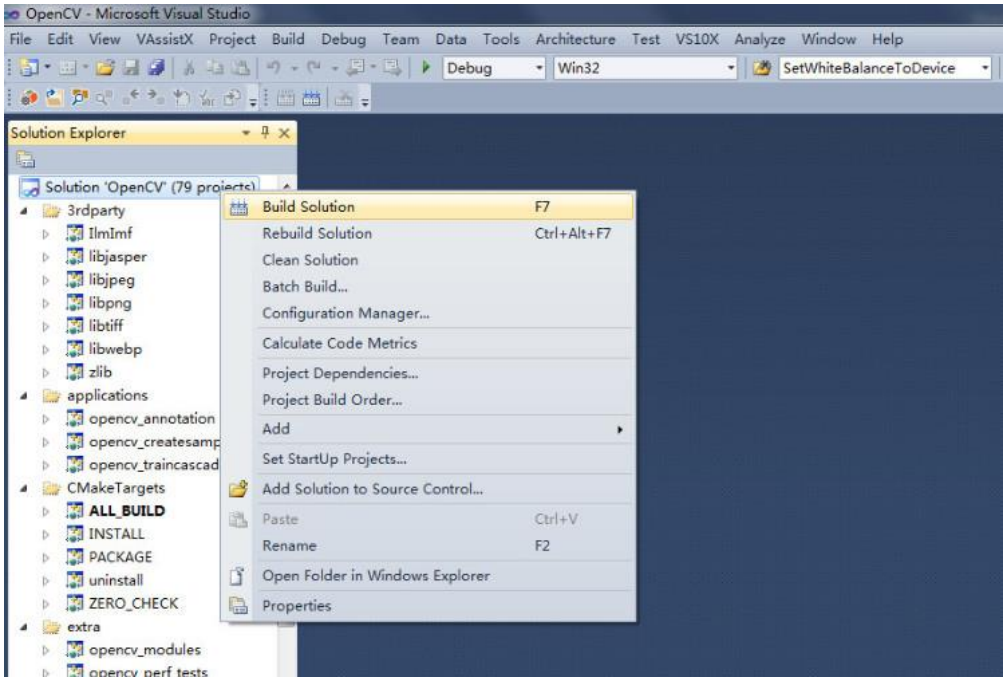

图 2-11

编译成功，如图 2-12 所示：

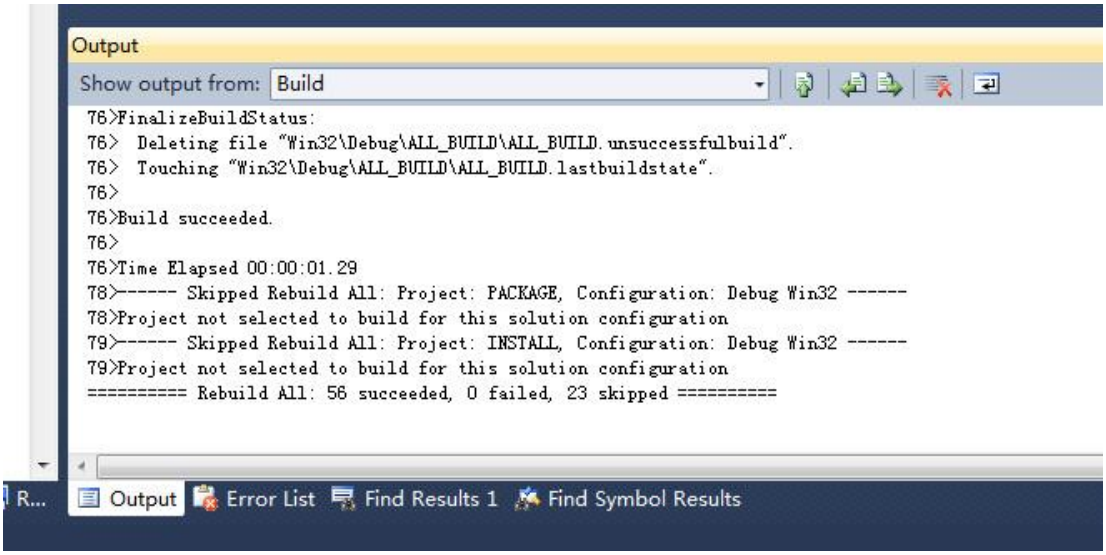

图 2-12

在 Visual Studio2010 中添加对 OpenCV 头文件的引用：

D:\opencv3.0.0\opencv\build\include;

D:\opencv3.0.0\opencv\build\include\opencv;

D:\opencv3.0.0\opencv\build\include\opencv2;

如图 2-13 所示：

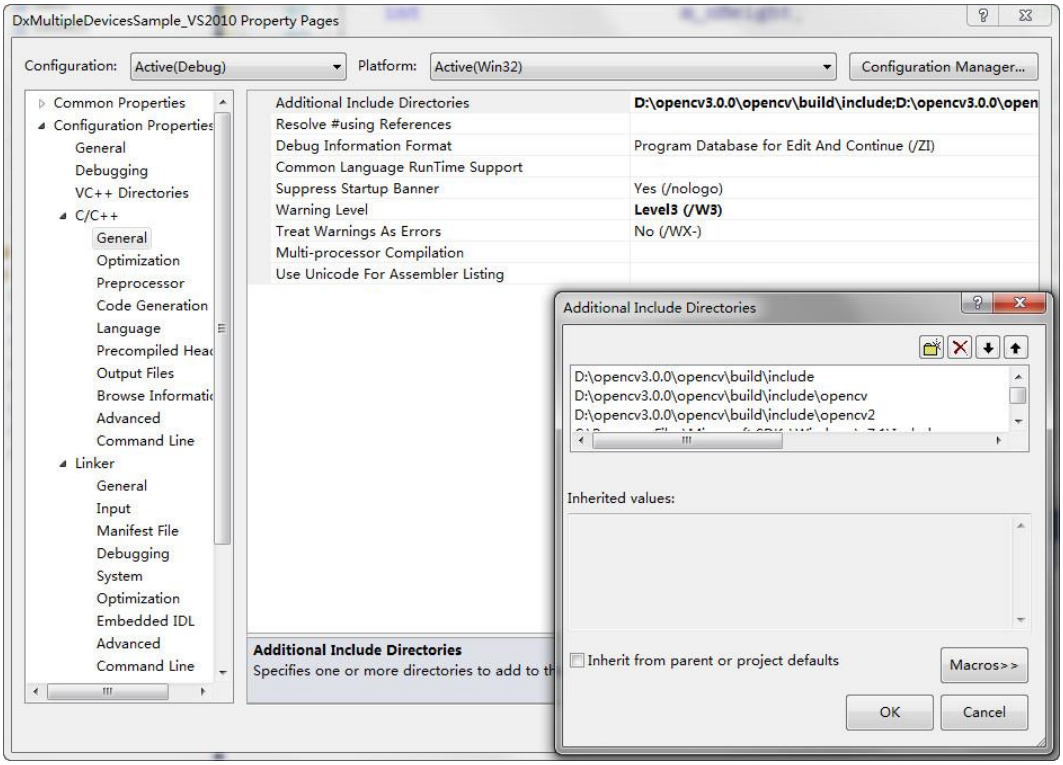

图 2-13

在 Visual Stdio2010 中添加 OpenCV 的 lib 文件路径 D:\opencv3.0.0\opencv_3.0.0_vs2010_x86\lib\Debug，如图 2-14 所示：

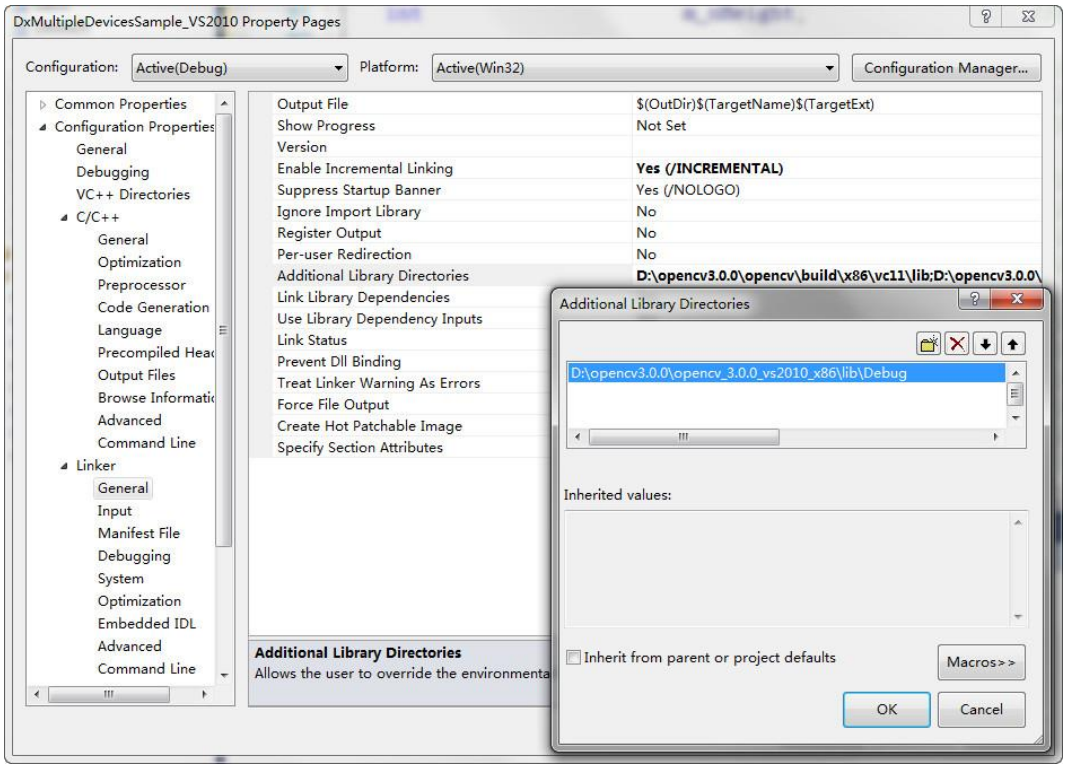

图 2-14

将\opencv_3.0.0_vs2010_x86\lib\Debug\中所有的 lib 文件添加到 Visual Studio2010 工程中:

opencv_videostab300d.lib

opencv_videoio300d.lib

opencv_video300d.lib

opencv_ts300d.lib

opencv_superres300d.lib

opencv_stitching300d.lib

opencv_shape300d.lib

opencv_photo300d.lib

opencv_objdetect300d.lib

opencv_ml300d.lib

opencv_imgproc300d.lib

opencv_imgcodecs300d.lib

opencv_highgui300d.lib

opencv_hal300d.lib

opencv_flann300d.lib

opencv_features2d300d.lib

opencv_core300d.lib

opencv_calib3d300d.lib

如图 2-15 所示：

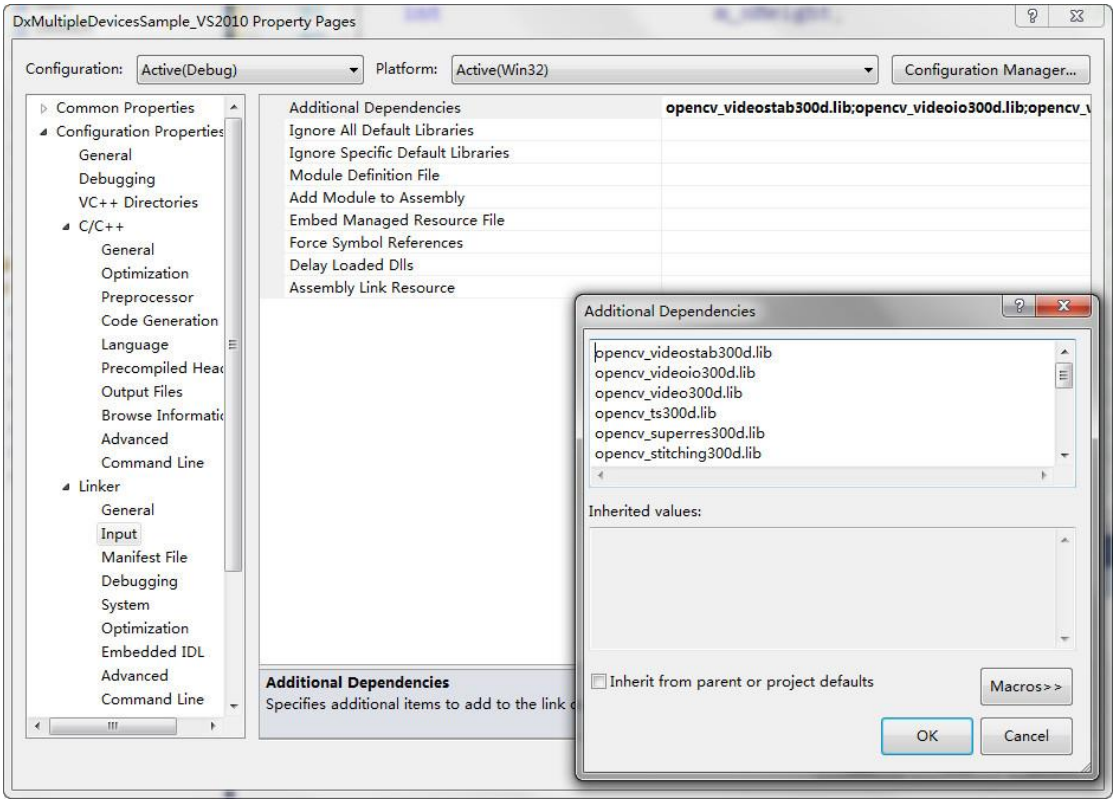

图 2-15

2.1.4. 环境变量的配置

在运行 Visual Studio 生成的 exe，弹框提示缺少 OpenCV 的 dll 时，需要配置环境变量。在系统变量中添加变量名为 OPENCV，对应的变量值为 D:\opencv3.0.0\opencv\build\include；在系统变量名 Path 中需要添加的变量值为：

D:\opencv3.0.0\opencv\build\x86\vc11\bin;

D:\opencv3.0.0\opencv\build\x86\vc12\bin;

D:\opencv3.0.0\opencv\build\x64\vc11\bin;

D:\opencv3.0.0\opencv\build\x64\vc12\bin;

2.2. SDK 7.1 配置

需要生成并配置 strmbase.lib 文件，该 lib 文件为用于编写 DirectShow 接口的基类库文件。

2.2.1. 安装 SDK

根据操作系统位数下载相应的 windows sdk 7.1。（链接：https://www.microsoft.com/en-us/download/details.aspx?id=8279）

运行 setup.exe，出现安装界面，点击“Next >”。

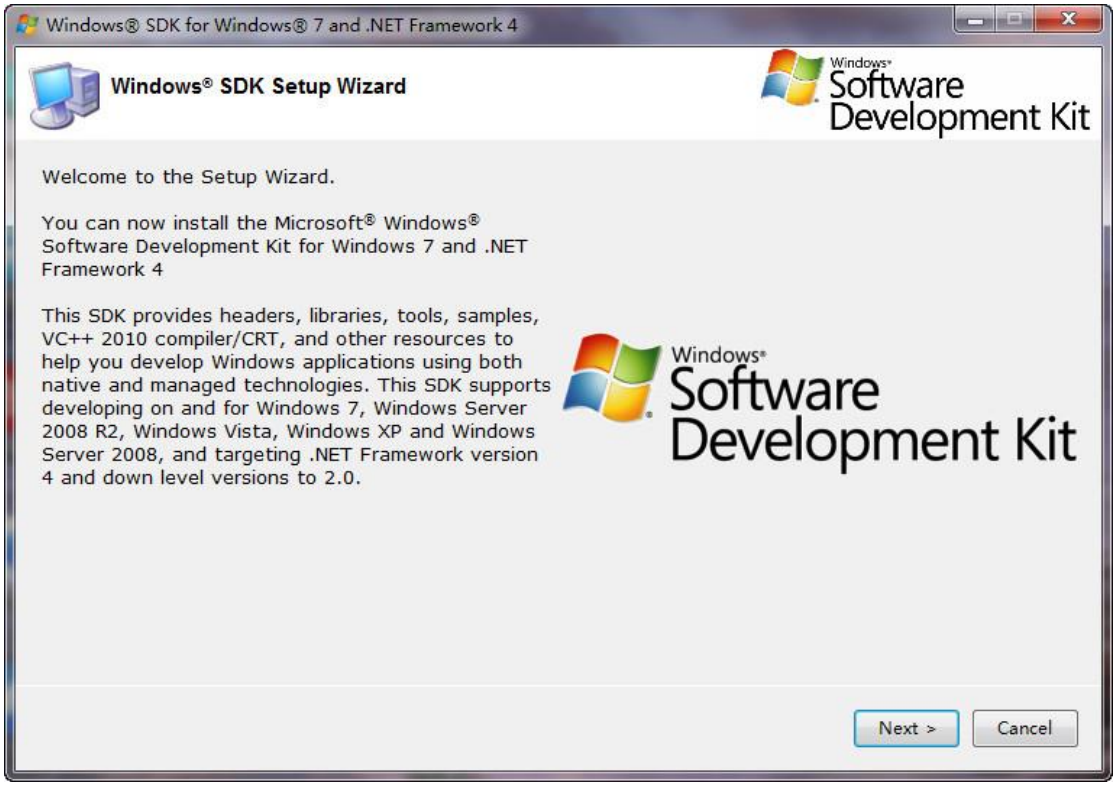

图 2-16

选择“I Agree”接受许可协议中的条款，点击“Next >”，进入下一步。选择“I Disagree”不接受许可协议中条款将不允许执行下一步操作。

图 2-17

选择 Windows SDK 7.1 的安装目录， 点击“Next >”，进入下一步：

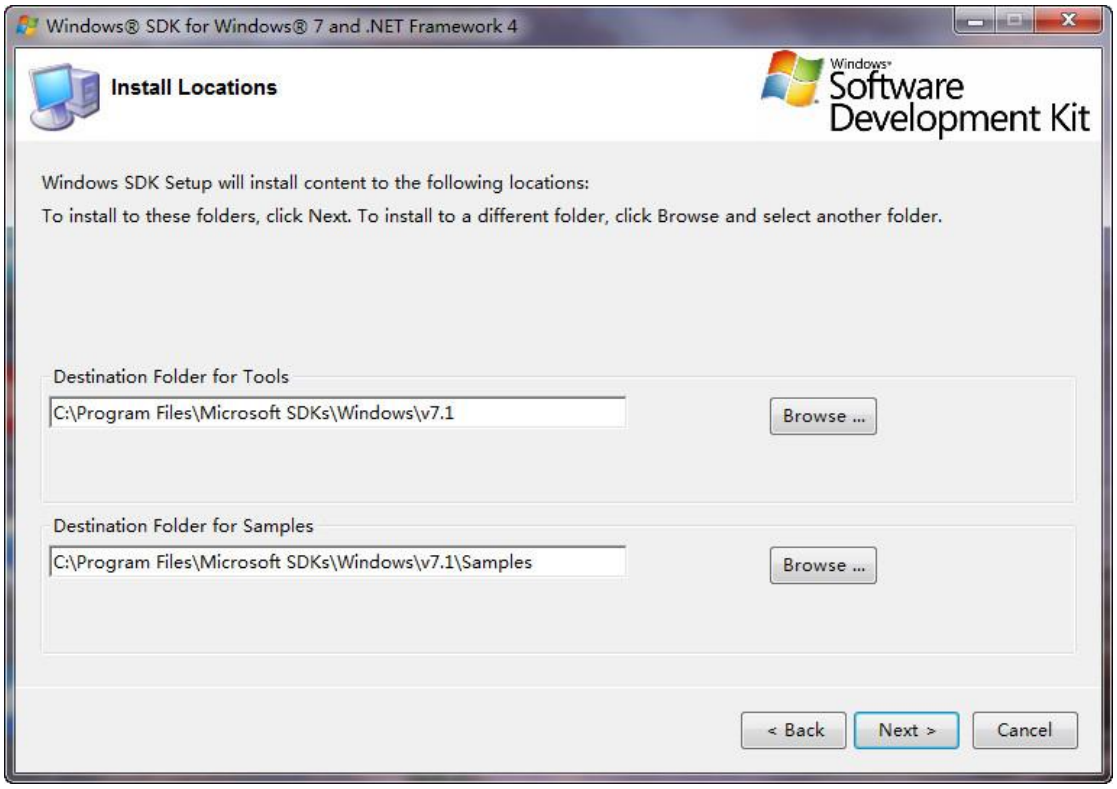

图 2-18

安装默认选择的内容即可，点击“Next >”，进入下一步，开始安装：

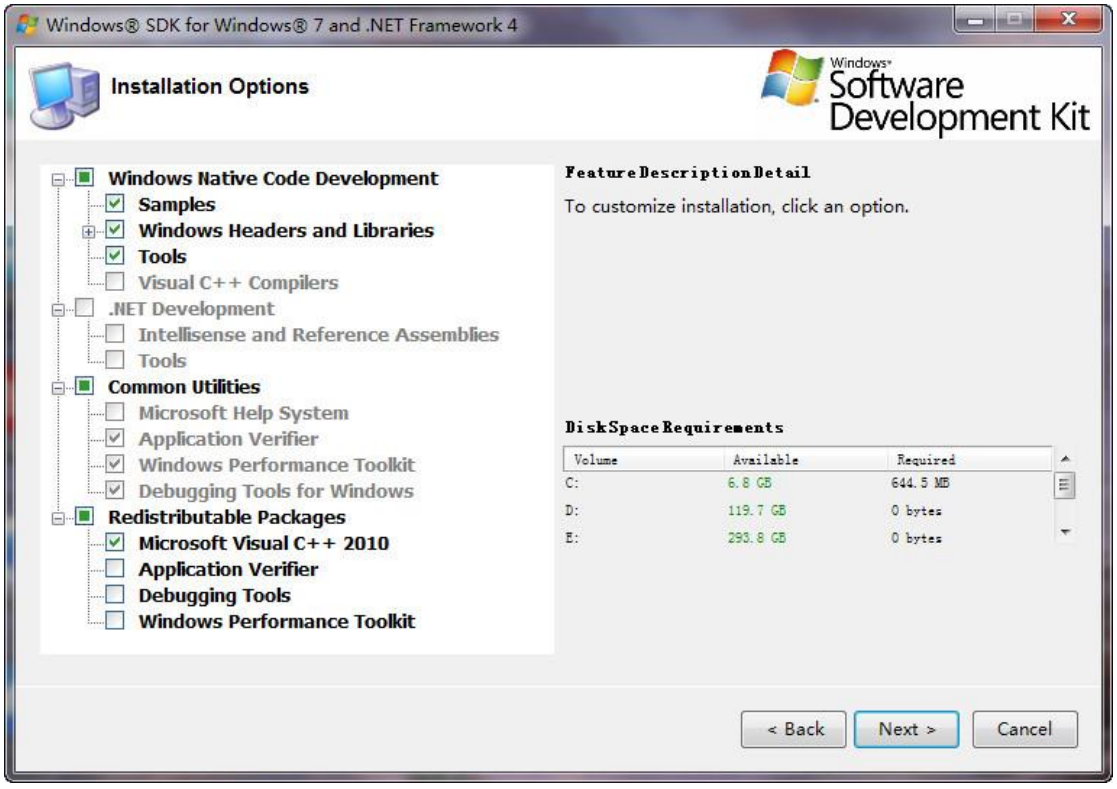

图 2-19

安装完成。

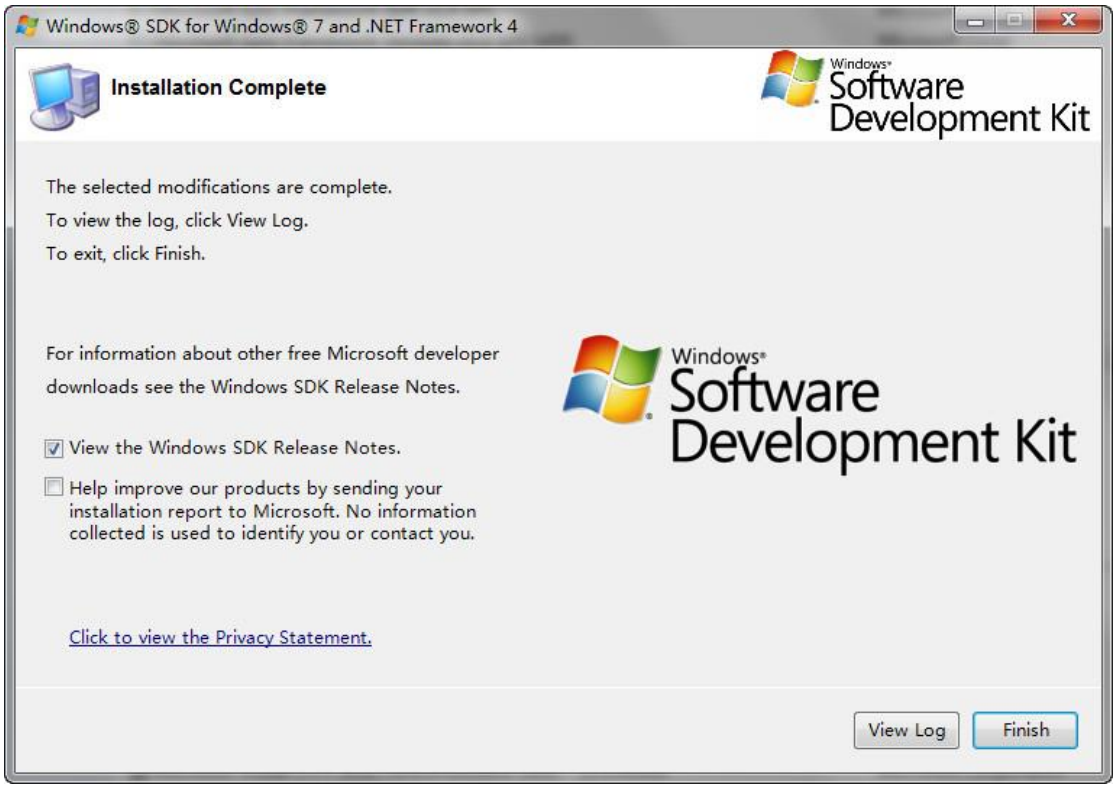

图 2-20

2.2.2. 在 Visual Studio 工程中配置 SDK

以 Visual Studio 2010 为例 配置头文件 C:\Program Files\Microsoft SDKs\Windows\v7.1\Include：

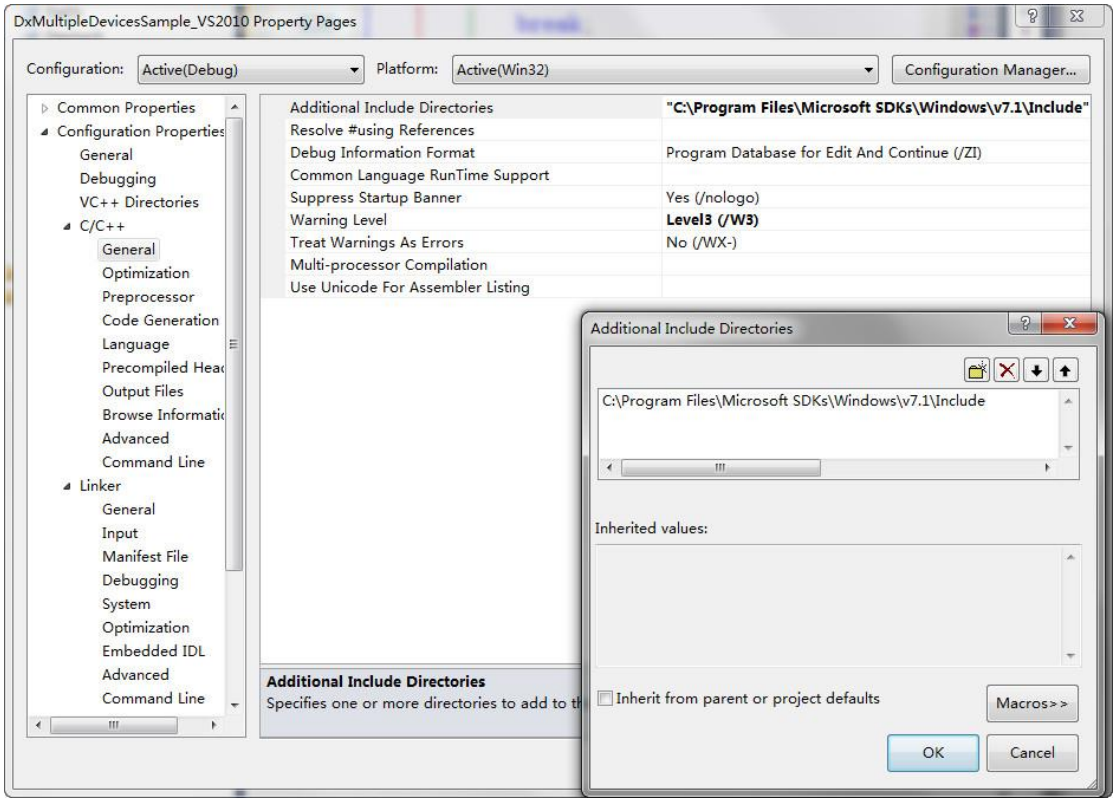

图 2-21

配置 lib 文件路径 C:\Program Files\Microsoft SDKs\Windows\v7.1\Lib：

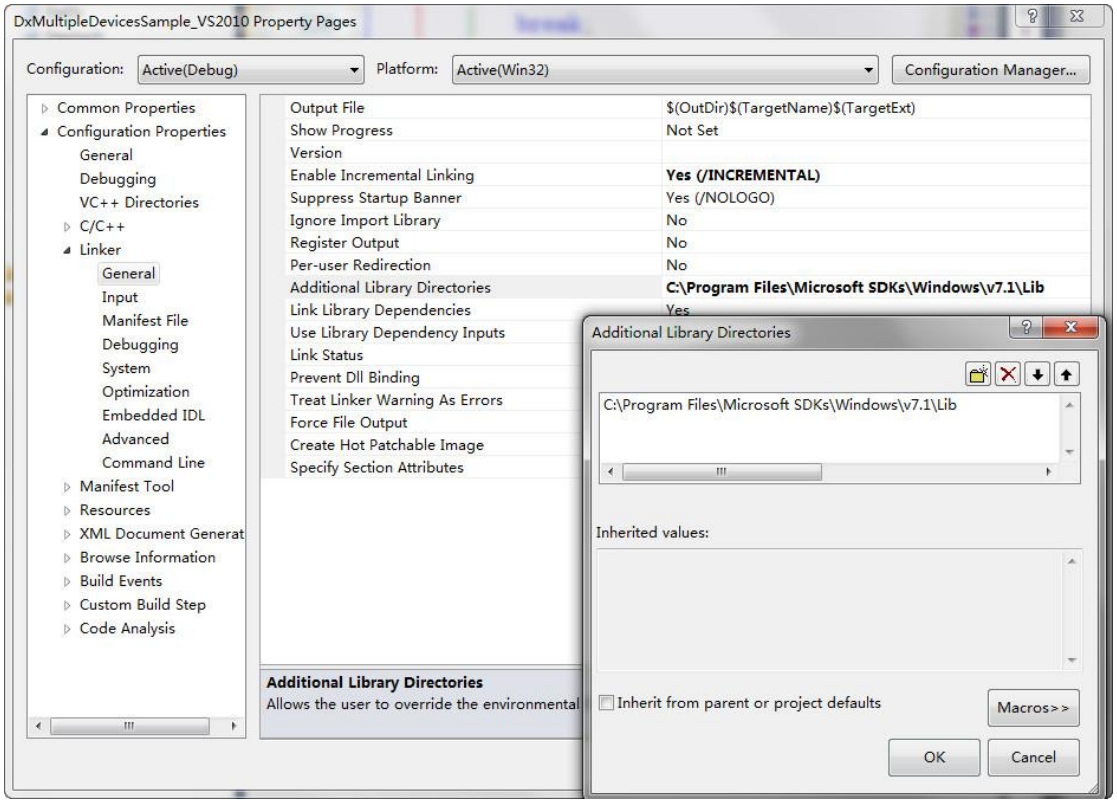

图 2-22

添加对 lib 文件路径下的 winmm.lib 引用：

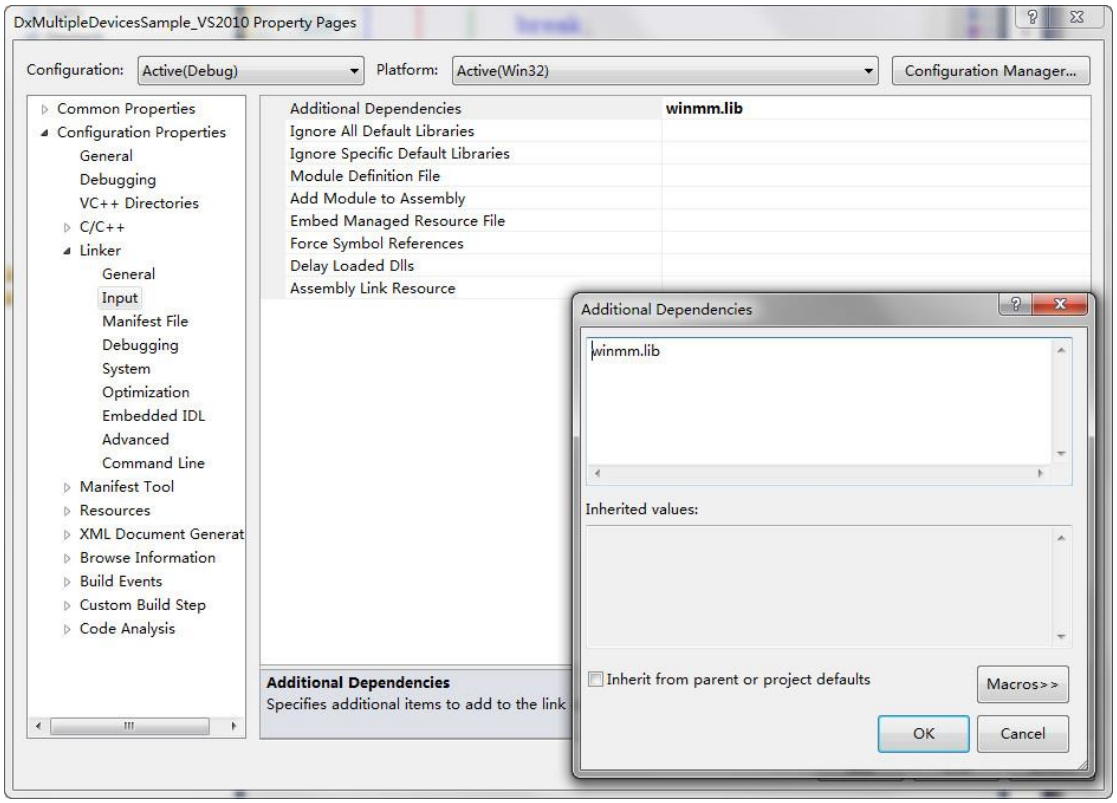

图 2-23

此外，还需要配置对 strmbase.lib 的引用。使用 Visual Studio 打开\Microsoft SDKs\Windows\v7.1\Samples\multimedia\directshow\baseclasses\baseclasses.sln 文件，根据需要生成对应版本的 strmbase.lib，点击编译按钮：

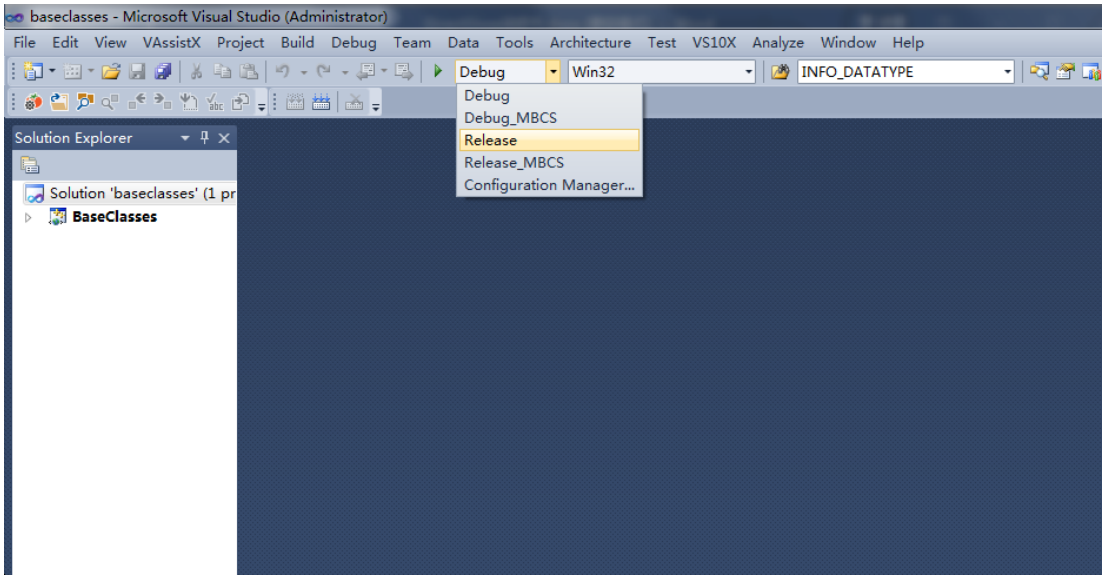

图 2-24

将生成的 strmbase.lib 添加到自己的工程中，并引用：

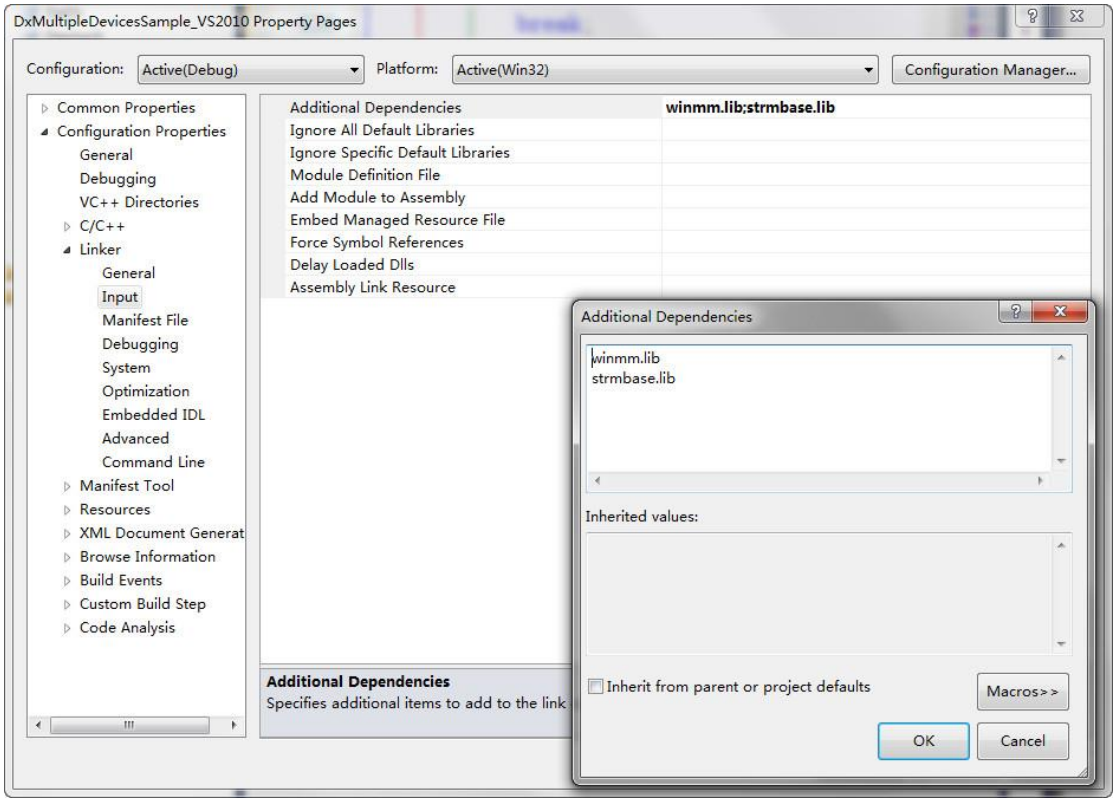

图 2-25

至此，DirectShow 的开发环境配置完成。

3. 接口流程介绍

IDHCamFilter 和 IDHCamPin 接口在 GXBase.h 头文件中声明。

3.1. IDHCamFilter 接口

3.1.1. IsColor

声明：

```
STDMETHOD(IsColor)(bool& bIsColor)
```

意义：

获取当前相机是否为彩色相机

形参：

```
[out]bIsColor //彩色相机标志位
```

返回值：

```
S_OK           //读取标志位成功
E_NOTIMPL      //接口未实现
E_NOINTERFACE //不支持此接口
E_FAIL         //未指定的失败
E_POINTER      //无效指针
E_HANDLE       //无效句柄
```

上面没有涵盖到的,不常见的错误情况请参考 HRESULT。

代码样例：

```
//-------------------------------------------------------
/**
\brief  从设备中获取是否为彩色相机标志位
\param  bColorFlag      输出彩色标志位
\return DEVICE_STATUS   输出错误码
*/
//-------------------------------------------------------
DEVICE_STATUS CDevice::GetColorFromDevice(bool &bColorFlag)
{
      HRESULT                 hResult     = S_OK;
      DEVICE_STATUS           emStatus    = DEVICE_GET_COLOR_FAIL;
      CComPtr<IDHCamFilter>   pCamFilter  = NULL;
```

```
        do
        {
            //判断当前相机是否已经打开
            if (m_bOpenFlag == false)
            {
                break;
            }

            //在 Filter 中查询 IDHCamFilter 接口
            hResult=m_pDeviceFilter->QueryInterface(IID_IDHCamFilte
            r, (void **)&pCamFilter);
            VERIFY_HRESULT_PRINT_RETURN_STATUS(hResult);

            //获取彩色相机标志位
            hResult = pCamFilter->IsColor(bColorFlag);
            VERIFY_HRESULT_PRINT_RETURN_STATUS(hResult);

            emStatus = DEVICE_SUCCESS;
        } while (0);

        //释放 IDHCamFilter 接口资源
        pCamFilter = NULL;

        return emStatus;
    }
```

3.1.2. EnableColorCorrect

声明：

```
STDMETHOD(EnableColorCorrect)(bool bIsEnable)
```

意义：

开启或关闭颜色校正标志位

形参：

```
[in]bIsEnable //开启或关闭颜色校正标志位
```

返回值：

```
S_OK            //读取标志位成功
E_NOTIMPL       //接口未实现
E_NOINTERFACE   //不支持此接口
E_FAIL          //未指定的失败
```

```
E_POINTER       //无效指针
E_HANDLE        //无效句柄
```

上面没有涵盖到的,不常见的错误情况请参考 HRESULT。

3.1.3. GetColorCorrectStatus

声明：

```
STDMETHOD(GetColorCorrectStatus)(bool& bIsEnable)
```

意义：

获取颜色校正位使能状态

形参：

```
[out]bIsEnable//颜色校正位使能状态
```

返回值：

```
S_OK            //读取标志位成功
E_NOTIMPL       //接口未实现
E_NOINTERFACE   //不支持此接口
E_FAIL          //未指定的失败
E_POINTER       //无效指针
E_HANDLE        //无效句柄
```

上面没有涵盖到的,不常见的错误情况请参考 HRESULT。

3.1.4. SetSharpen

声明:

```
STDMETHOD(SetSharpen)(bool bIsEnable, double dValue)
```

意义:

设置锐化当前值

形参:

```
[in] bIsEnable//设置锐化使能
[in] dValue   //设置锐化当前值
```

返回值：

```
S_OK            //读取标志位成功
E_NOTIMPL       //接口未实现
E_NOINTERFACE   //不支持此接口
E_FAIL          //未指定的失败
E_POINTER       //无效指针
E_HANDLE        //无效句柄
```

上面没有涵盖到的,不常见的错误情况请参考 HRESULT。

3.1.5. SetLightness

声明:

```
STDMETHOD(SetLightness)( bool bIsEnable, long nValue)
```

意义:

设置亮度当前值

形参:

```
[in] bIsEnable//设置亮度使能
[in] dValue   //设置亮度当前值
```

返回值：

```
S_OK            //读取标志位成功
E_NOTIMPL       //接口未实现
E_NOINTERFACE   //不支持此接口
E_FAIL          //未指定的失败
E_POINTER       //无效指针
E_HANDLE        //无效句柄
```

上面没有涵盖到的,不常见的错误情况请参考 HRESULT。

3.1.6. SetContrast

声明:

```
STDMETHOD(SetContrast)(bool bIsEnable, long nValue)
```

意义:

设置对比度当前值

形参:

```
[in] bIsEnable//设置对比度使能
[in] dValue   //设置对比度当前值
```

返回值：

```
S_OK           //读取标志位成功
E_NOTIMPL      //接口未实现
E_NOINTERFACE  //不支持此接口
E_FAIL         //未指定的失败
E_POINTER      //无效指针
E_HANDLE       //无效句柄
```

上面没有涵盖到的,不常见的错误情况请参考 HRESULT。

3.1.7. SetSaturation

声明:

```
STDMETHOD(SetSaturation)(bool bIsEnable, long nValue)
```

意义:

设置饱和度当前值

形参:

```
[in] bIsEnable//设置饱和度使能
[in] dValue   //设置饱和度当前值
```

返回值：

```
S_OK           //读取标志位成功
E_NOTIMPL      //接口未实现
E_NOINTERFACE  //不支持此接口
E_FAIL         //未指定的失败
E_POINTER      //无效指针
E_HANDLE       //无效句柄
```

上面没有涵盖到的,不常见的错误情况请参考 HRESULT。

3.1.8. SetGamma

声明:

```
STDMETHOD(SetGamma)(bool bIsEnable, double dValue)
```

意义:

设置 Gamma 当前值

形参:

```
[in] bIsEnable//设置 Gamma 使能
[in] dValue   //设置 Gamma 当前值
```

返回值：

```
S_OK           //读取标志位成功
E_NOTIMPL      //接口未实现
E_NOINTERFACE //不支持此接口
E_FAIL         //未指定的失败
E_POINTER      //无效指针
E_HANDLE       //无效句柄
```

上面没有涵盖到的,不常见的错误情况请参考 HRESULT。

3.1.9. GetSharpen

声明:

```
STDMETHOD(GetSharpen)( bool &bIsEnable, double& dValue)
```

意义:

获取锐化当前值

形参:

```
[out] bIsEnable//获取锐化使能标志位
[out] dValue   //获取锐化当前值
```

返回值：

```
S_OK           //读取标志位成功
E_NOTIMPL      //接口未实现
E_NOINTERFACE //不支持此接口
```

```
E_FAIL          //未指定的失败
E_POINTER       //无效指针
E_HANDLE        //无效句柄
```

上面没有涵盖到的,不常见的错误情况请参考 HRESULT。

3.1.10. GetLightness

声明:

```
STDMETHOD(GetLightness)( bool &bIsEnable, long& nValue)
```

意义:

获取亮度当前值

形参:

```
[out] bIsEnable//获取亮度使能标志位
[out] dValue    //获取亮度当前值
```

返回值：

```
S_OK            //读取标志位成功
E_NOTIMPL       //接口未实现
E_NOINTERFACE   //不支持此接口
E_FAIL          //未指定的失败
E_POINTER       //无效指针
E_HANDLE        //无效句柄
```

上面没有涵盖到的,不常见的错误情况请参考 HRESULT。

3.1.11. GetContrast

声明:

```
STDMETHOD(GetContrast)( bool &bIsEnable, long& nValue)
```

意义:

获取对比度当前值

形参:

```
[out] bIsEnable//获取对比度使能标志位
```

```
[out] dValue    //获取对比度当前值
```

返回值：

```
S_OK            //读取标志位成功
E_NOTIMPL       //接口未实现
E_NOINTERFACE   //不支持此接口
E_FAIL          //未指定的失败
E_POINTER       //无效指针
E_HANDLE        //无效句柄
```

上面没有涵盖到的,不常见的错误情况请参考 HRESULT。

3.1.12. GetSaturation

声明:

```
STDMETHOD(GetSaturation)( bool &bIsEnable, long& nValue)
```

意义:

获取饱和度当前值

形参:

```
[out] bIsEnable//获取饱和度使能标志位
[out] dValue    //获取饱和度当前值
```

返回值：

```
S_OK            //读取标志位成功
E_NOTIMPL       //接口未实现
E_NOINTERFACE   //不支持此接口
E_FAIL          //未指定的失败
E_POINTER       //无效指针
E_HANDLE        //无效句柄
```

上面没有涵盖到的,不常见的错误情况请参考 HRESULT。

3.1.13. GetGamma

声明:

```
STDMETHOD(GetGamma)( bool &bIsEnable, double& dValue)
```

意义:

获取 Gamma 当前值

形参:

```
[out] bIsEnable//获取 Gamma 使能标志位
[out] dValue   //获取 Gamma 当前值
```

返回值：

```
S_OK           //读取标志位成功
E_NOTIMPL      //接口未实现
E_NOINTERFACE  //不支持此接口
E_FAIL         //未指定的失败
E_POINTER      //无效指针
E_HANDLE       //无效句柄
```

上面没有涵盖到的,不常见的错误情况请参考 HRESULT。

3.1.14. GetPixelSize

声明:

```
STDMETHOD(GetPixelSize)(GX_PIXEL_SIZE_ENTRY& emPixelSize)
```

意义:

获取当前相机图像位深

形参:

```
[out] emPixelSize//获取图像位深当前值
```

返回值：

```
S_OK           //读取标志位成功
E_NOTIMPL      //接口未实现
E_NOINTERFACE  //不支持此接口
E_FAIL         //未指定的失败
E_POINTER      //无效指针
E_HANDLE       //无效句柄
```

上面没有涵盖到的,不常见的错误情况请参考 HRESULT。

3.1.15. GetDevicePointer

声明:

```
STDMETHOD(GetDevicePointer)(void** pDevice)
```

意义:

获取操作当前设备的指针

形参:

```
[out] pDevice //获取操作当前设备的指针
```

返回值：

```
S_OK           //读取标志位成功
E_NOTIMPL      //接口未实现
E_NOINTERFACE  //不支持此接口
E_FAIL         //未指定的失败
E_POINTER      //无效指针
E_HANDLE       //无效句柄
```

上面没有涵盖到的,不常见的错误情况请参考 HRESULT。

3.1.16. IsConnect

声明:

```
STDMETHOD(IsConnect)( bool& bIsConnected)
```

意义:

判断当前 Pin 是否连接

形参:

```
[out] bIsConnected//获取当前 Pin 连接标志位
```

返回值：

```
S_OK           //读取标志位成功
E_NOTIMPL      //接口未实现
E_NOINTERFACE  //不支持此接口
E_FAIL         //未指定的失败
```

```
E_POINTER      //无效指针
E_HANDLE       //无效句柄
```

上面没有涵盖到的,不常见的错误情况请参考 HRESULT。

3.2. IDHCamPin 接口

3.2.1. GetCurrentDeviceIndex

声明:

```
STDMETHOD(GetCurrentDeviceIndex)(long& plDevice)
```

意义:

获取当前设备索引值

形参:

```
[out] plDevice//获取当前设备索引值
```

返回值：

```
S_OK           //读取标志位成功
E_NOTIMPL      //接口未实现
E_NOINTERFACE  //不支持此接口
E_FAIL         //未指定的失败
E_POINTER      //无效指针
E_HANDLE       //无效句柄
```

上面没有涵盖到的,不常见的错误情况请参考 HRESULT。

代码样例

```
//-----------------------------------------------------
/**
\brief   获取当前设备索引值
\param   plDevice          输出当前设备索引值
\return DEVICE_STATUS      输出错误码
*/
//-----------------------------------------------------
DEVICE_STATUS CDevice::GetDeviceIndexFromDevice(long &plDevice)
{
     HRESULT               hResult = S_OK;
```

```
        DEVICE_STATUS
        emStatus=DEVICE_GET_CURRENT_DEVICE_INDEX_FAIL;
        CComPtr<IDHCamPin>  pCamPin = NULL;

        do
        {
            //在 Filter 中查询 IDHCamPin 接口
           hResult = m_pDeviceFilter->QueryInterface(IID_IDHCamPin,
            (void **)&pCamPin);
            VERIFY_HRESULT_PRINT_RETURN_STATUS(hResult);

            //获取当前设备在设备列表中的索引值
            hResult = pCamPin->GetCurrentDeviceIndex(plDevice);
            VERIFY_HRESULT_PRINT_RETURN_STATUS(hResult);

            emStatus = DEVICE_SUCCESS;
        } while (0);

            //释放 IDHCamPin 接口资源
        pCamPin = NULL;

        return emStatus;
    }
```

3.2.2. SetCurrentDeviceIndex

声明:

```
STDMETHOD(SetCurrentDeviceIndex)(long lDevice)
```

意义:

设置当前设备索引值

形参:

```
[in] lDevice  //设置当前设备索引值
```

返回值：

```
S_OK           //读取标志位成功
E_NOTIMPL      //接口未实现
E_NOINTERFACE //不支持此接口
E_FAIL         //未指定的失败
E_POINTER      //无效指针
E_HANDLE       //无效句柄
```

上面没有涵盖到的,不常见的错误情况请参考 HRESULT。

3.2.3. GetDeviceList

声明:

```
STDMETHOD(GetDeviceList)(GX_ENUM_DESCRIPTION
*pEnumDescription,long*pBufferSizet)
```

意义:

获取当前设备列表中的设备

形参:

```
[out] pEnumDescription//输出设备信息结构体包括设备序号，设备名称
[out]pBufferSizet     //输出设备数量
```

返回值：

```
S_OK           //读取标志位成功
E_NOTIMPL      //接口未实现
E_NOINTERFACE  //不支持此接口
E_FAIL         //未指定的失败
E_POINTER      //无效指针
E_HANDLE       //无效句柄
```

上面没有涵盖到的,不常见的错误情况请参考 HRESULT。

4. 常见问题处理

序号	常见问题	解决办法
1	Halcon 的注意事项：当注册的大恒设备个数大于连接的设备个数时，会弹框提示找不到设备的错误信息	1) 注册大恒相机个数与连接设备的个数一致。
2	MATLAB 的注意事项：使用 matlab 打开连接的设备，提示 No Image Acquisition adaptors found. Image acquisition adaptors may be available as downloadable support packages. Open Support Package Installer to install additional vendors	1) 点击 Support Package Installer， MATLAB 会提供大概 13 个软件包，这时候选择 OS Generic Video Interface 下载安装。

5. 版本说明

序号	修订版本号	所做改动	发布日期
1	V1.0.0	初始发布	2018-08-01