China Daheng Group, Inc. Beijing Image Vision Technology Branch

DirectShow

User Manual

Version: V1.0.0

Date: 2018-08-01

DAHENG IMAGING 大恒图像

Notice

© 2018China Daheng Group, Inc. Beijing Image Vision Technology Branch

Web: http://www.daheng-imaging.com

Sales Email: isales@daheng-imaging.com

Sales Tel: +86 10 8282 8878

Support Email: isupport@daheng-imaging.com

Contents

1. Install the DirectShow Program

1.1. Install the DirectShow Program

In the Daheng Imaging Camera Software Suite, when using **DX_Setup_en.exe** to install the DirectShow program, the registration interface appears. In the **Enter the number of DX registrations:** edit box, enter the number of cameras you want to register (up to 32 cameras can be registered), click the **Register** button to register, as shown in Figure 1- 1:

Figure 1- 1

After registration is complete, **GxDirectShowRegister.exe** exits automatically. To verify that cameras are registered successfully, run **graphedt.exe**, select **Graph** in the menu bar, and click **Insert Filters** in the drop-down list.

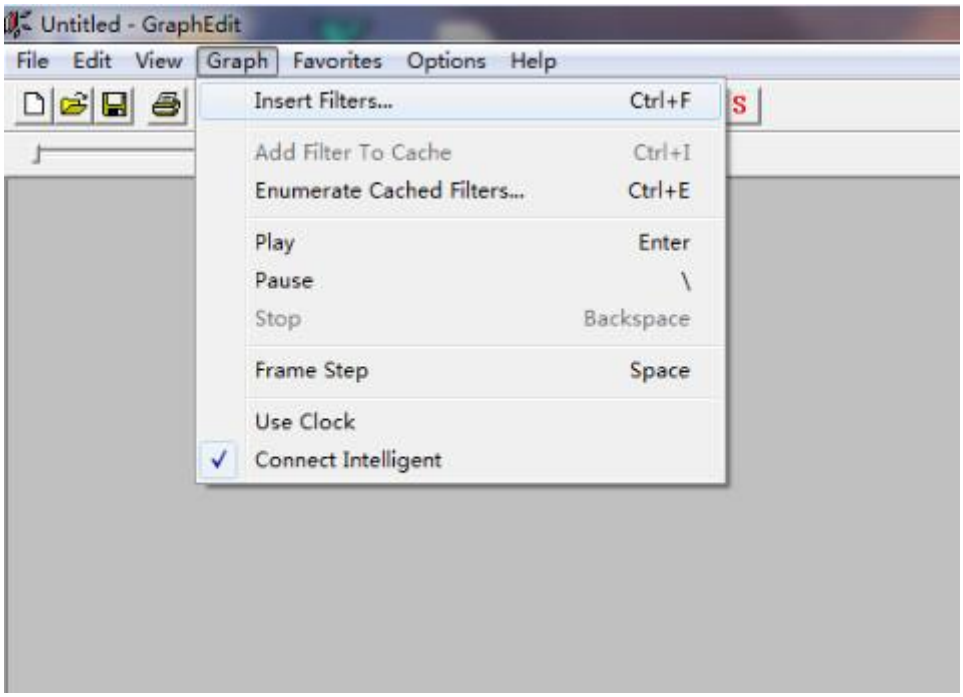

Figure 1- 2

In the pop-up window, select **Video Capture Sources** to check whether the number of registered cameras is correct.

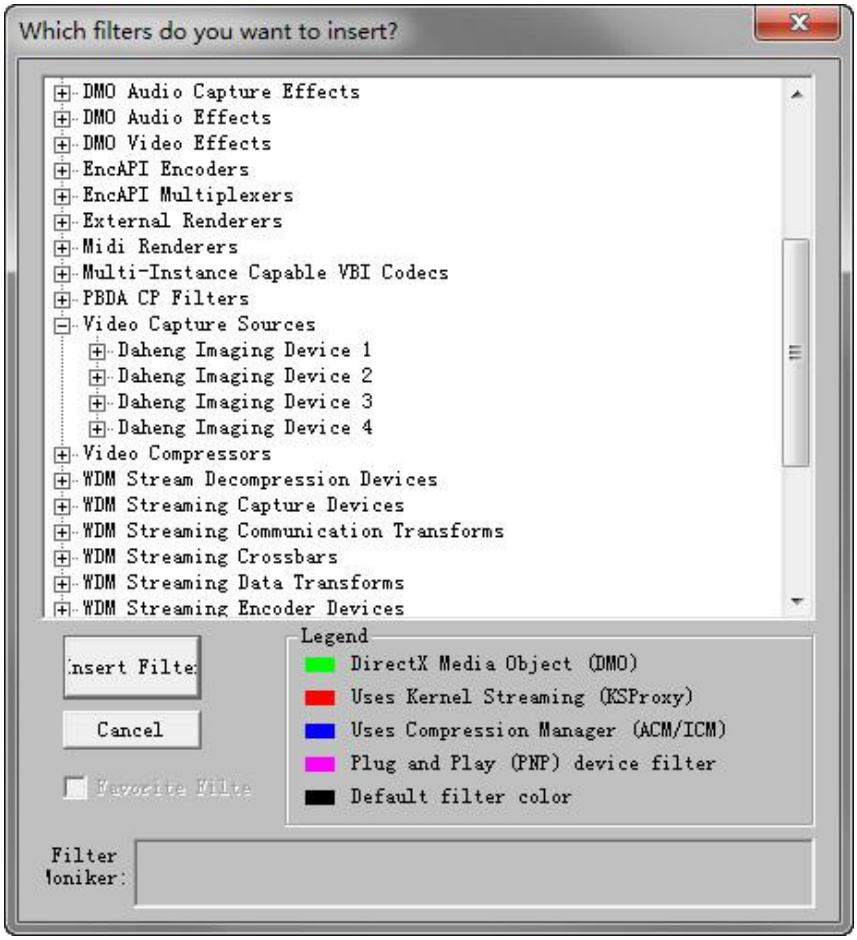

Figure 1- 3

1.2. Uninstall the DirectShow Program

Run **unins000.exe** to uninstall the **DirectShow** program. During the uninstall process, the unregistration interface appears, as shown in Figure 1- 4. In the **Enter the number of DX registrations:** edit box, enter the number of cameras you want to unregister (up to 32 cameras can be unregistered), and click the **Unregister** button to unregister:

Figure 1- 4

After the unregistration is completed, **GxDirectShowRegister.exe** automatically exits. Verify that cameras are unregistered successfully, when the number of cameras that are unregistered is less than the number

of cameras registered, the unregistration should be performed from the first camera. Run **graphedt.exe**, select **Graph** in the menu bar, and click **Insert Filters** in the drop-down list.

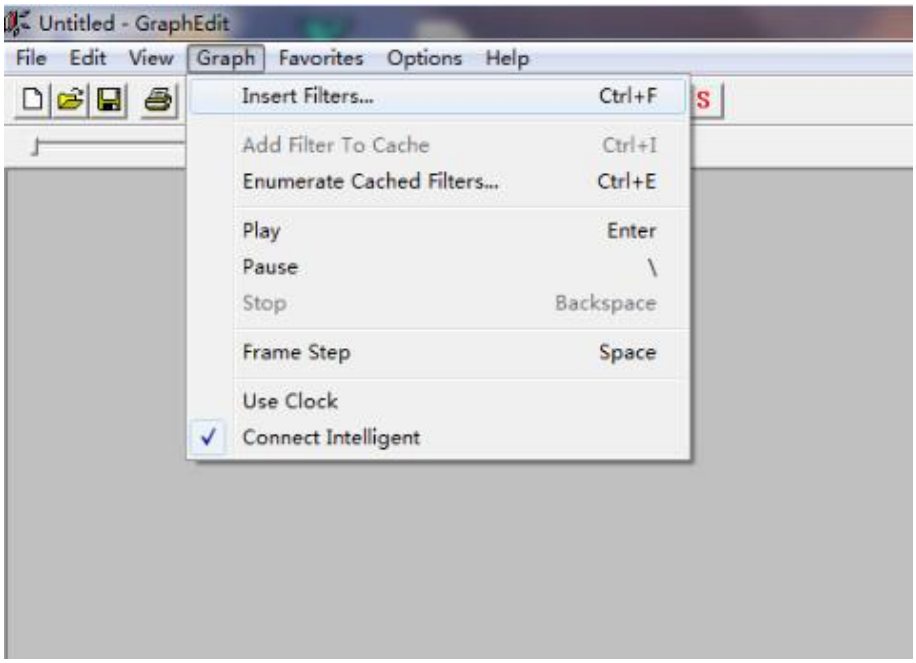

Figure 1- 5

In the pop-up window, select **Video Capture Sources** to check whether the number of cameras that are unregistered is correct.

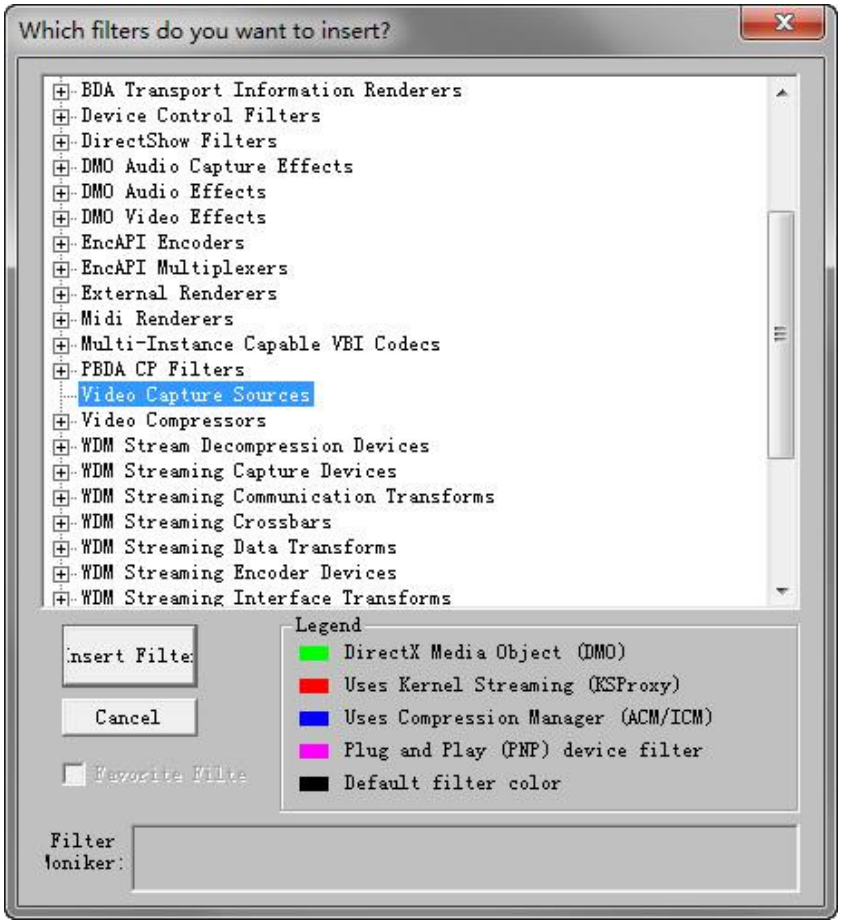

Figure 1- 6

2. Environmental Configuration

2.1. OpenCV3.0.0 Configuration

2.1.1. Install the OpenCV3.0.0

Download **OpenCV3.0.0** (Link: https://sourceforge.net/projects/opencvlibrary/files/opencv-win/3.0.0/)

Run **OpenCV-3.0.0.exe**, select the export directory, click the **Extract** button to export the **OpenCV** source code.

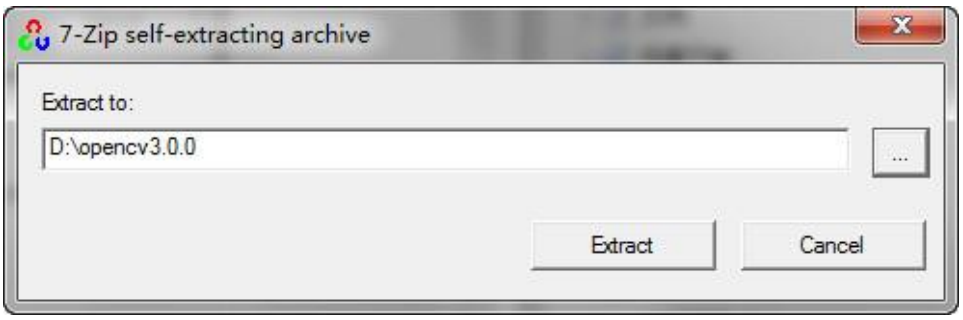

Figure 2- 1

2.1.2. Configuring OpenCV3.0.0 in Visual Studio project using the libraries that already generated by OpenCV

Take the configuration of the 32 bit **Debug** version program under **Visual Studio 2010** as an example, add the header file path:

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

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

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

As shown in Figure 2- 2:

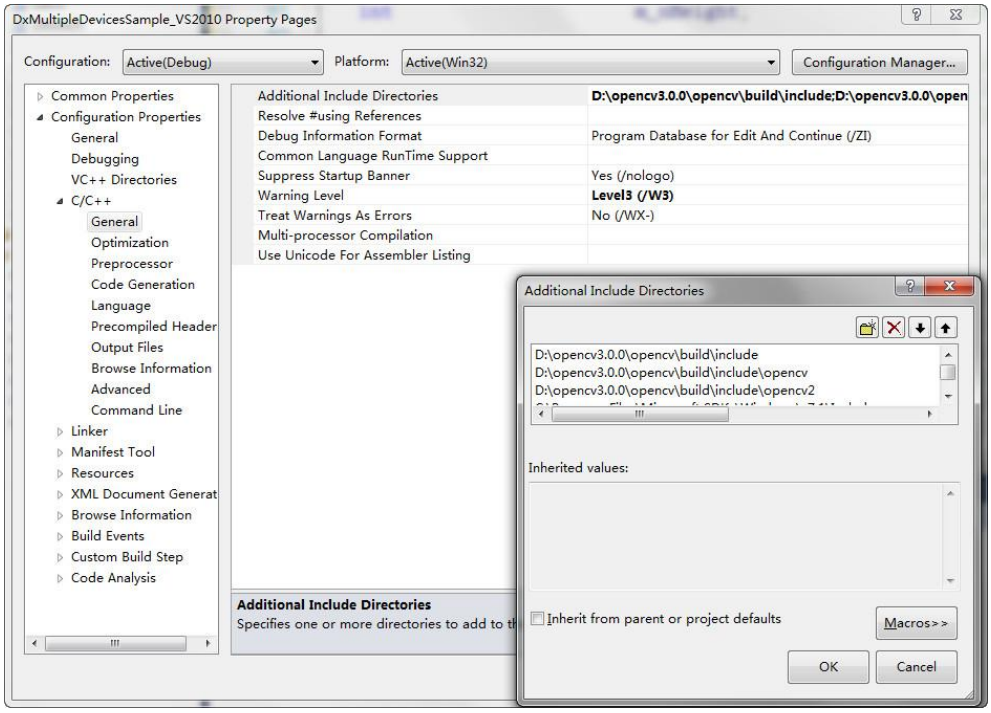

Figure 2- 2

Configure the **lib** file paths:

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

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

As shown in Figure 2- 3:

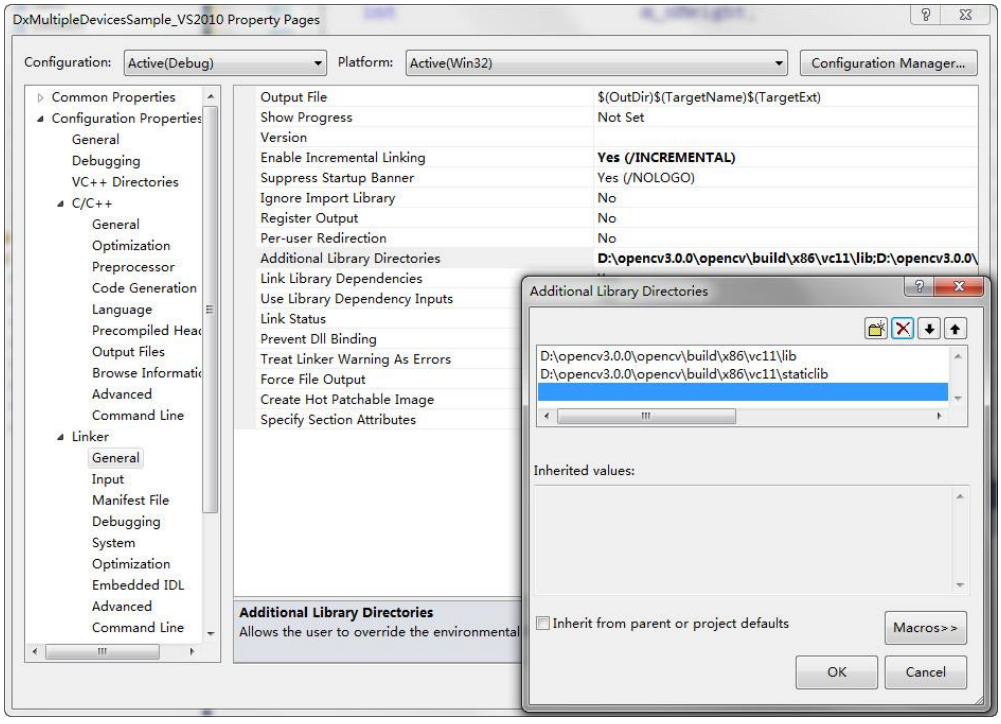

Figure 2- 3

Add a reference to the **lib** file under the path of **debug** mode, the specific files are:

opencv_ts300.lib,

opencv_world300.lib.

As shown in Figure 2- 4:

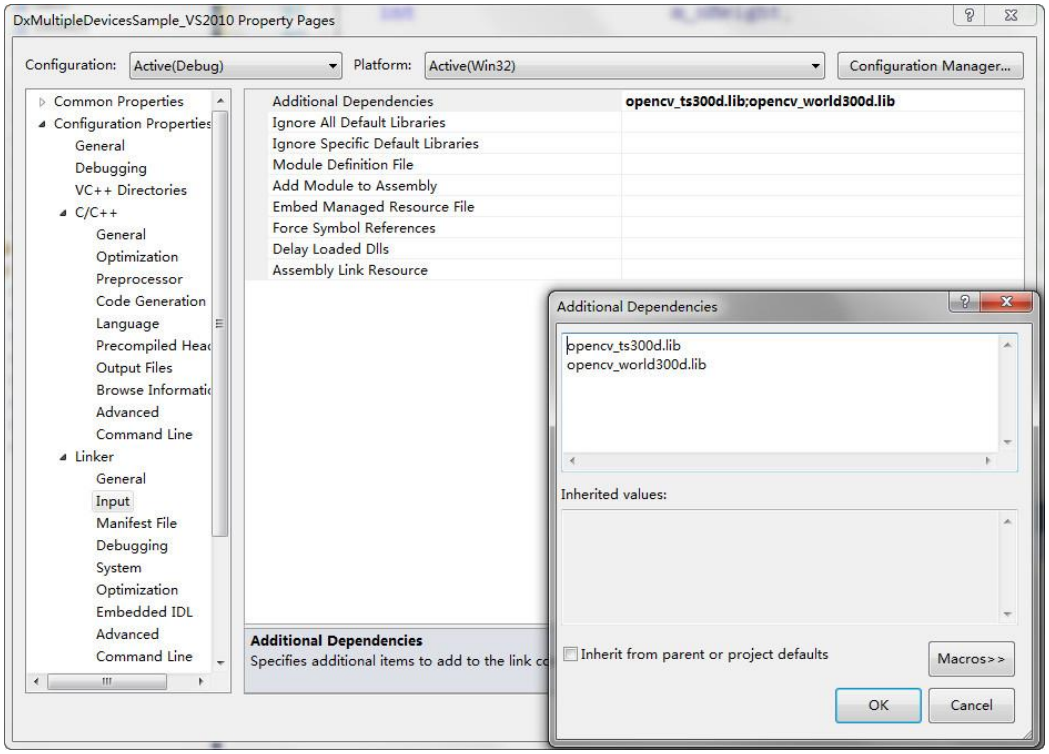

Figure 2- 4

2.1.3. Configuring OpenCV3.0.0 in Visual Studio project using CMake mode

When you can't configure **Visual Studio** with **vc11** or **vc12**, use **CMake** to compile **OpenCV** source code. In the unpacked **CMake** directory, find **\bin\cmake-gui.exe**, click **Run**, select **OpenCV** source directory, and choose to generate binary file storage directory. As shown in Figure 2- 5:

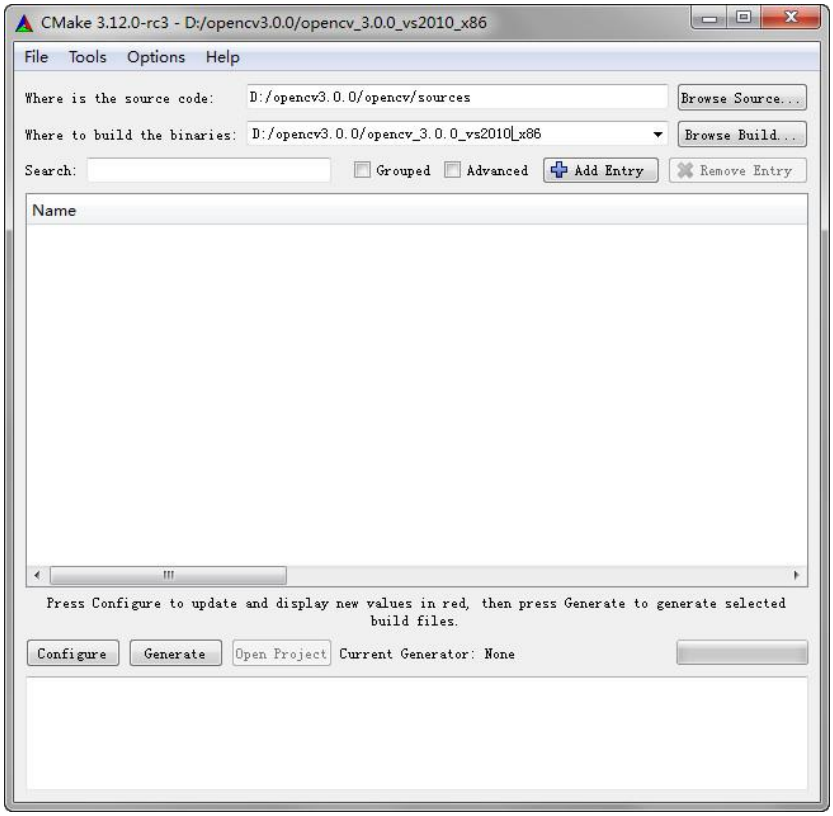

Figure 2- 5

Click the **Configure** button. Take the example of compiling 32 bit source code using **Visual Studio 2010**, as shown in Figure 2- 6:

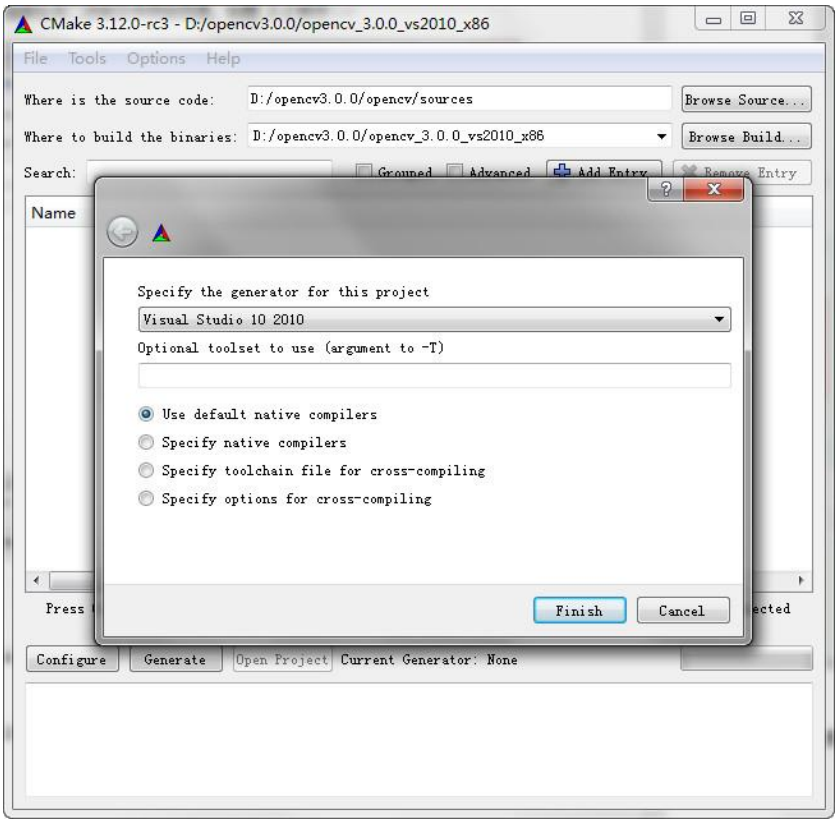

Figure 2- 6

You need to download resources from the network during the configuration process, so keep the network unblocked. As shown in Figure 2- 7, it is in the process of being configured:

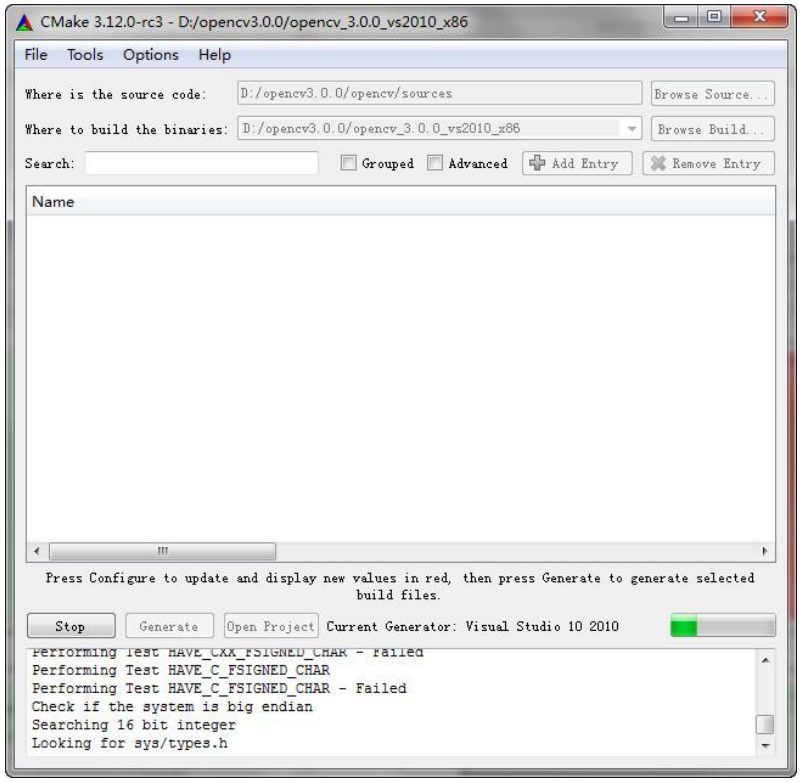

Figure 2- 7

When the progress bar execution is complete, and the words **Configuring done** appear. The first source configuration is complete, as shown in Figure 2- 8:

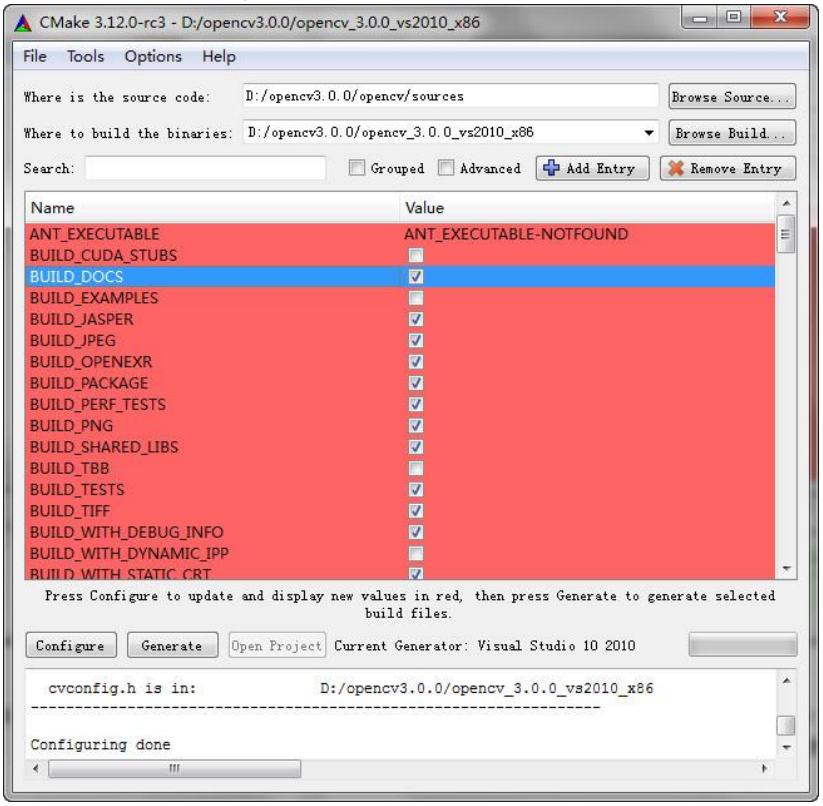

Figure 2- 8

After the first configuration is complete, select the build files in the reddish file shown in the Figure 2- 8, and then click the **Configure** button again for the second configuration. When the progress bar is completed, the words **Configure done** appear and the red marked part becomes normal. As shown in Figure 2- 9:

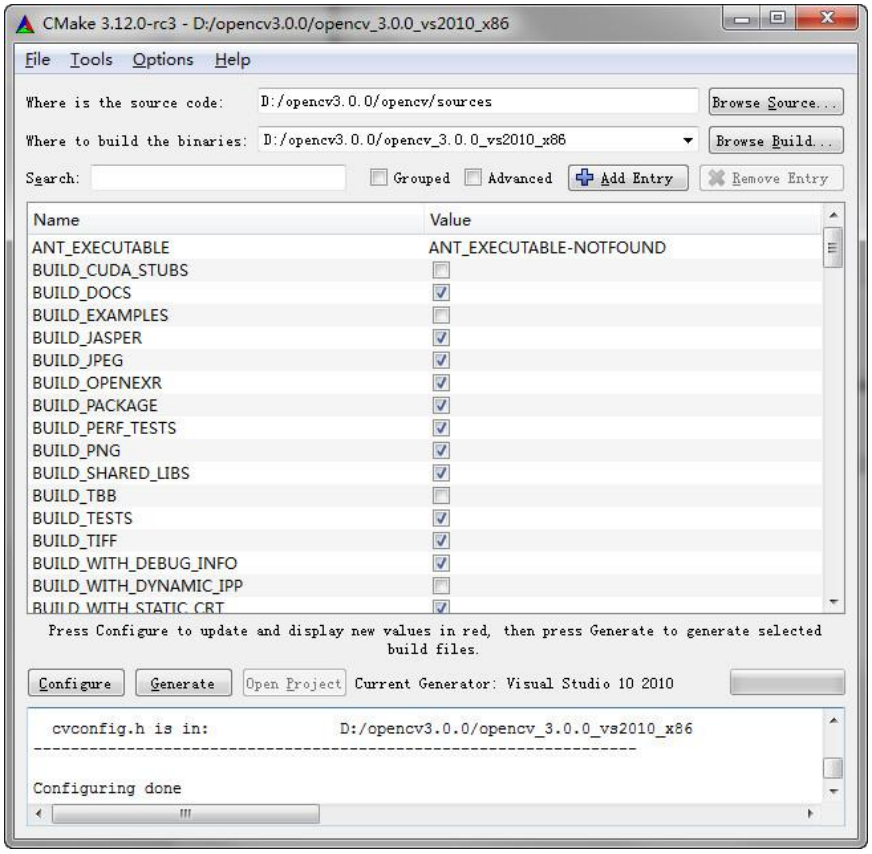

Figure 2- 9

Click the **Generate** button to create a solution for **Visual Studio 2010**, namely **OpenCV.sln**. As shown in Figure 2- 10:

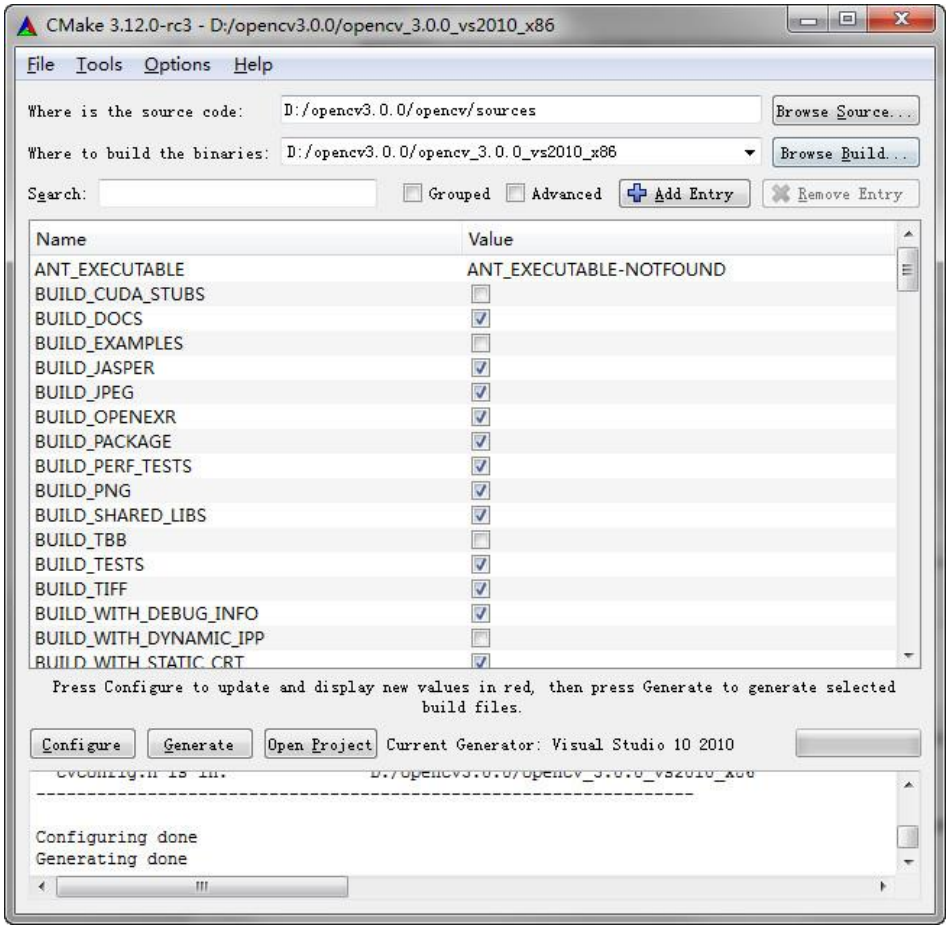

Figure 2- 10

Clicking on the **Open Project** button will open **OpenCV.sln** and generate a solution in **Visual Studio 2010**. After the compilation is successful, the **OpenCV3.0.0** dependent libraries are generated in the **\opencv_3.0.0_vs2010_x86\lib\Debug** directory, as shown in Figure 2- 11:

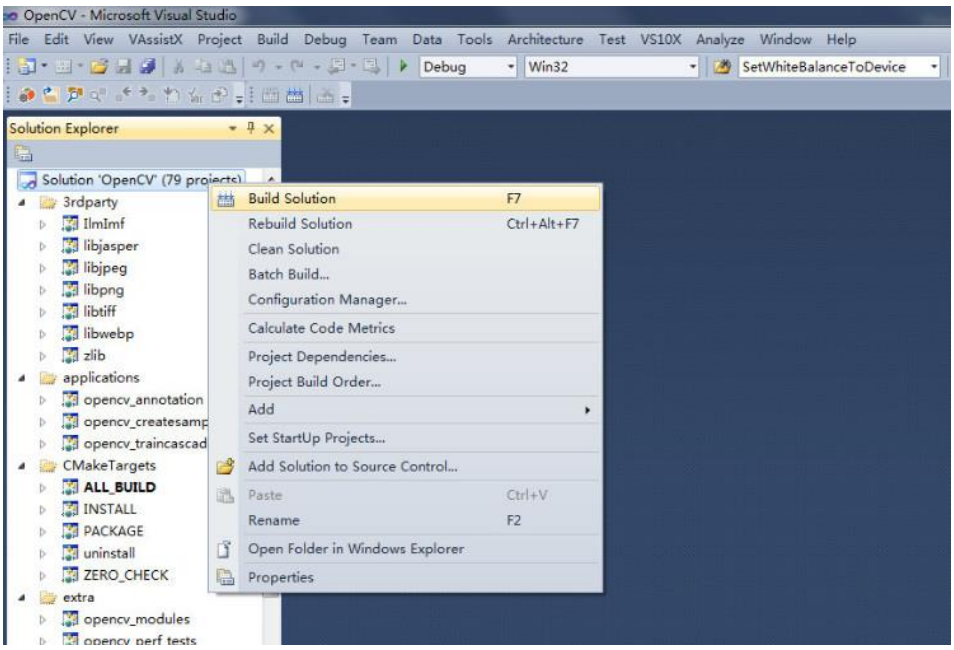

Figure 2- 11

The compilation is successful, as shown in Figure 2- 12:

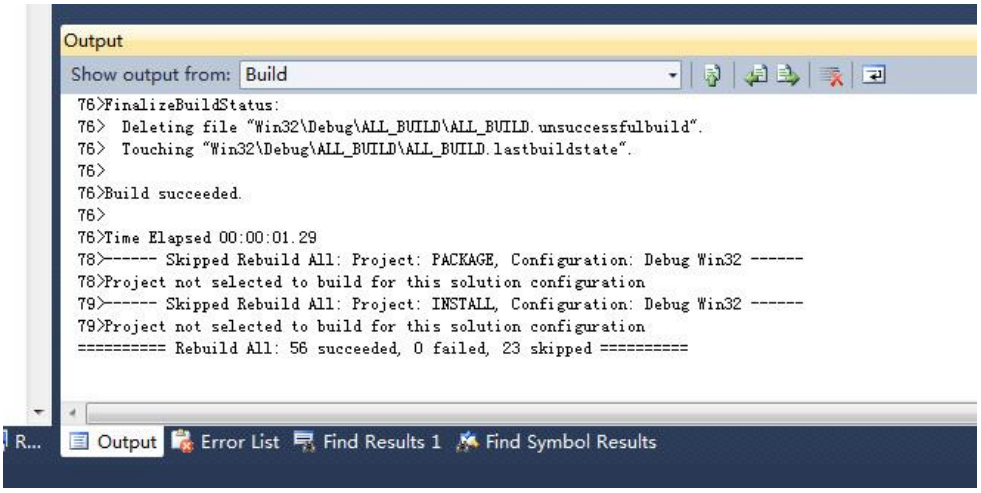

Figure 2- 12

Add references to the **OpenCV** header file in **Visual Studio 2010**:

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

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

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

As shown in Figure 2- 13:

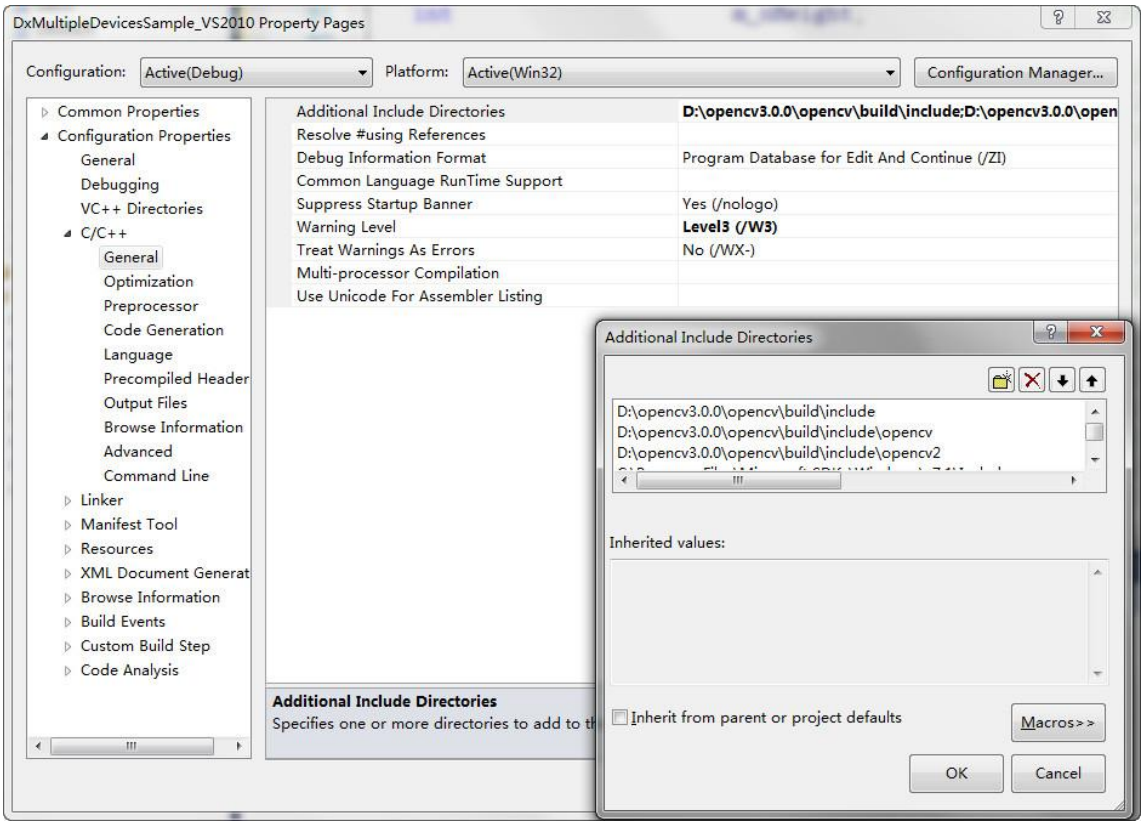

Figure 2- 13

Add **OpenCV**'s **lib** file path in **Visual Stdio 2010**: **D:\opencv3.0.0\opencv_3.0.0_vs2010_x86\lib\Debug**.

As shown in Figure 2- 14:

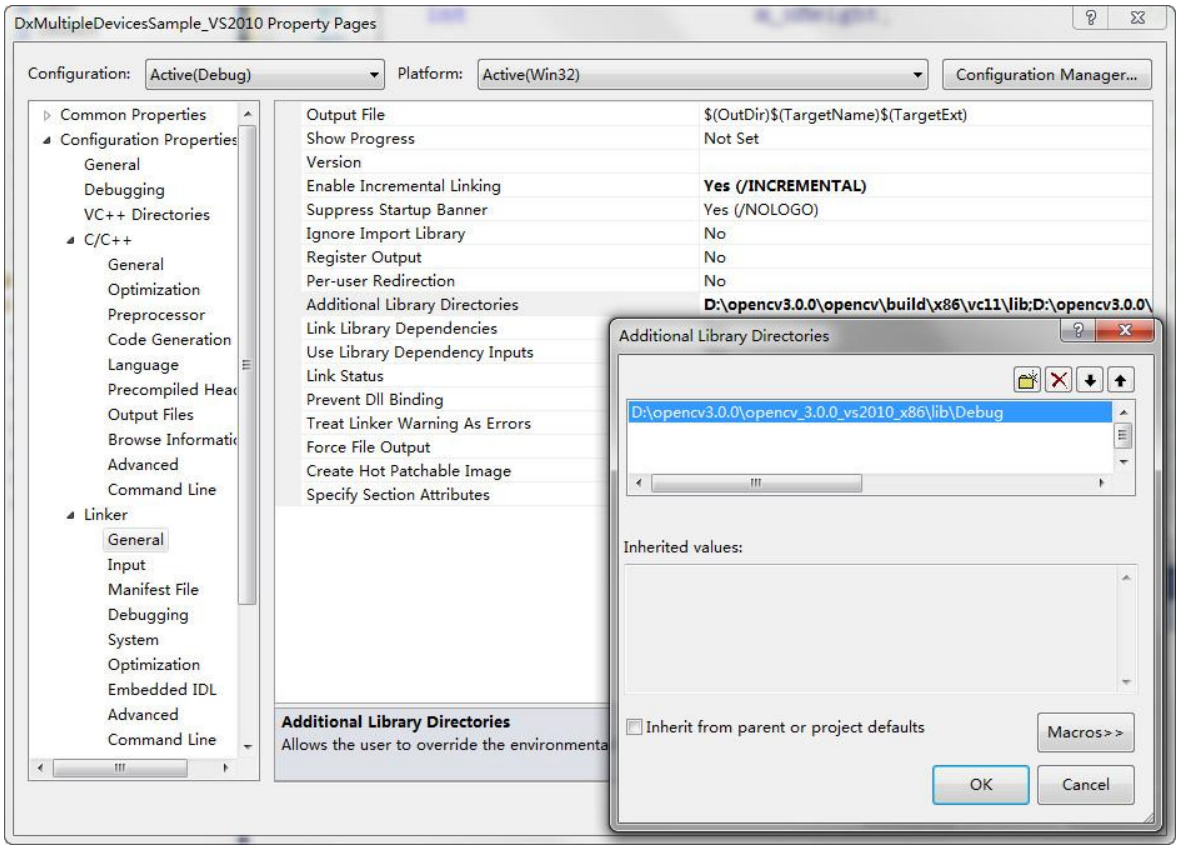

Figure 2- 14

Add all the **lib** files in **\opencv_3.0.0_vs2010_x86\lib\Debug** to the **Visual Studio 2010** project:

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

As shown in Figure 2- 15:

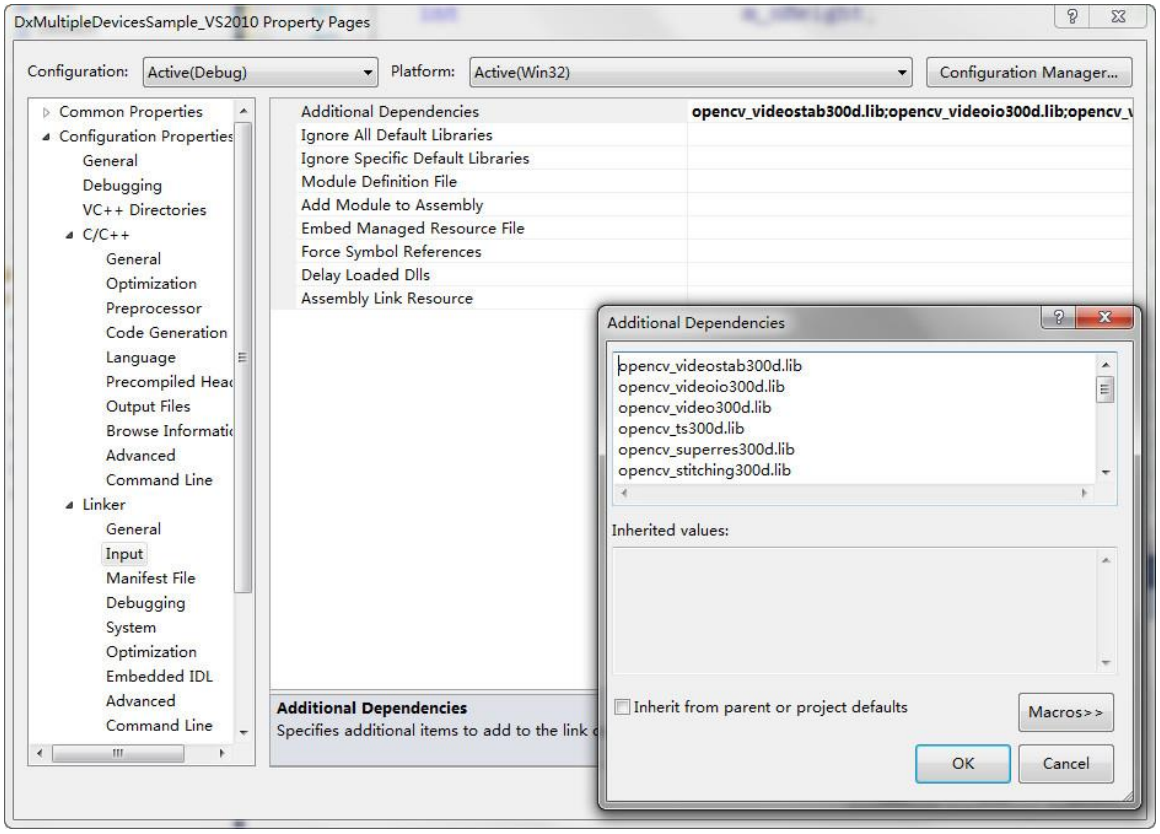

Figure 2- 15

2.1.4. Configuration of environment variables

When running the **exe** program which is generated by **Visual Studio**, the pop-up dialog prompts the lack of **OpenCV dll**, you need to configure the environment variables. Add the variable named **OPENCV** to the system variable, the corresponding variable value is **D:\opencv3.0.0\opencv\build\include**, the value of the variable to be added in the system variable name **Path** are:

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 Configuration

Need to generate and configure the **strmbase.lib** file, which is the base class library file used to code the **DirectShow** interface.

2.2.1. Install the SDK

Download the corresponding **windows SDK 7.1** according to the number of operating system bits. （Link: https://www.microsoft.com/en-us/download/details.aspx?id=8279 ）

Run **setup.exe**, the installation interface appears, click **Next >**.

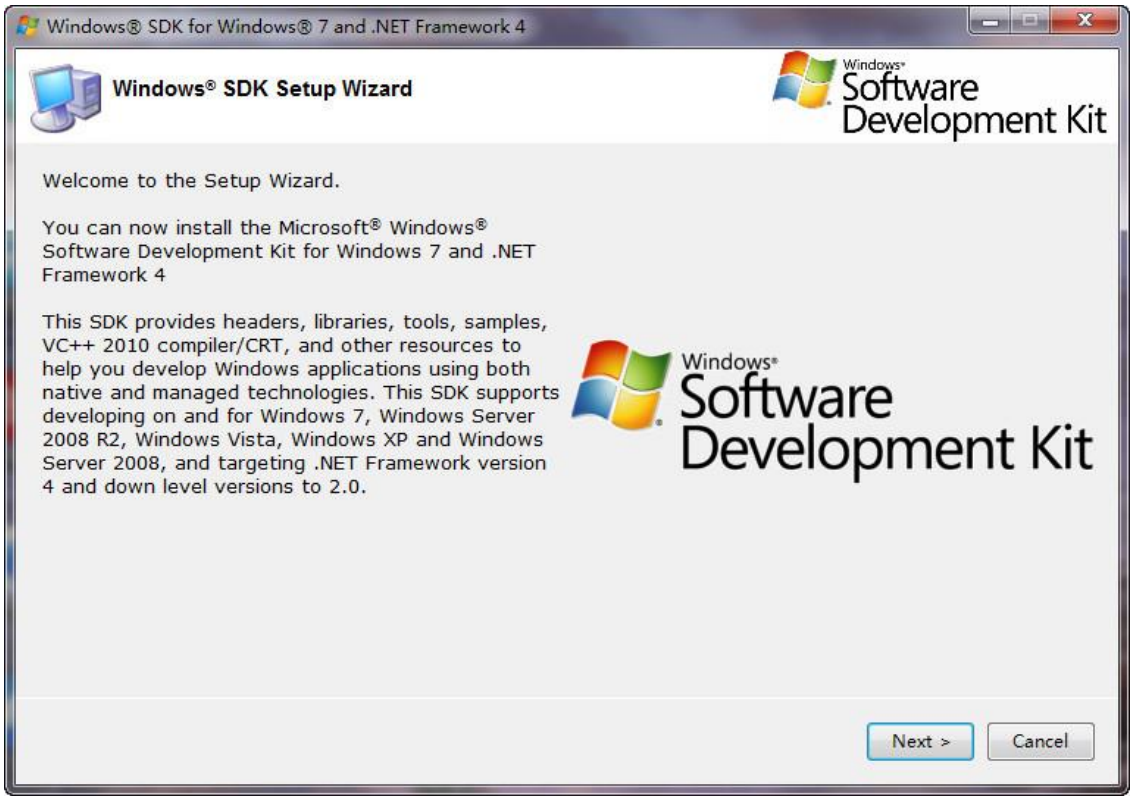

Figure 2- 16

Select **I Agree** to accept the terms in the license agreement and click **Next >** to proceed to the next step. Select **I Disagree** does not accept the terms of the license agreement and will not allow the next step.

Figure 2- 17

Select the installation directory of **Windows SDK 7.1**, click **Next >** to enter the next step:

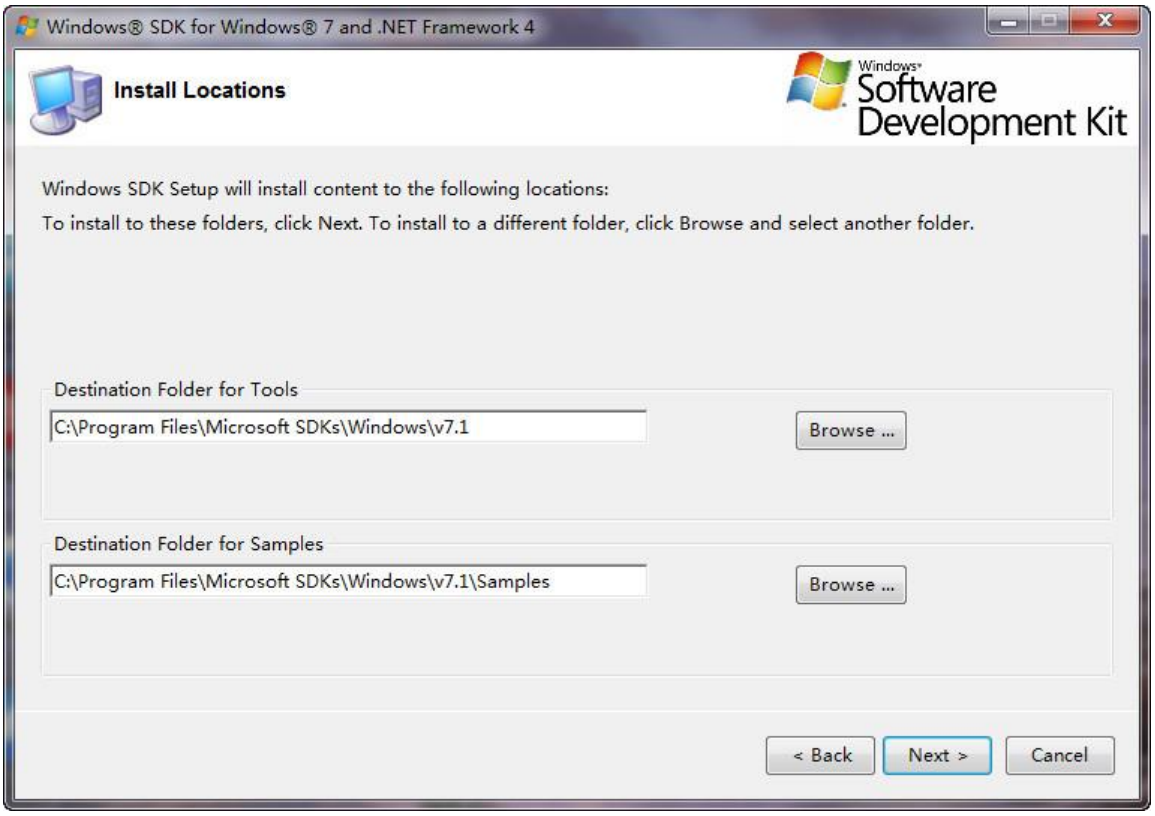

Figure 2- 18

Install the default selected content, click **Next >**, go to the next step, start the installation:

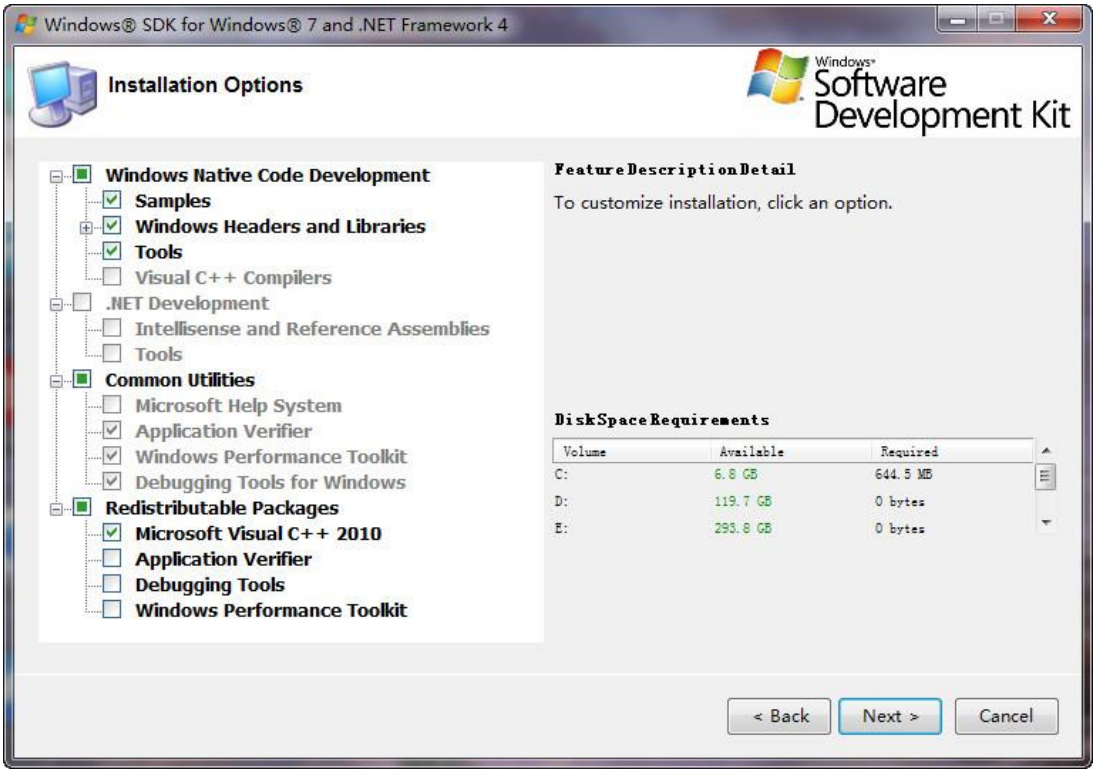

Figure 2- 19

The installation is complete.

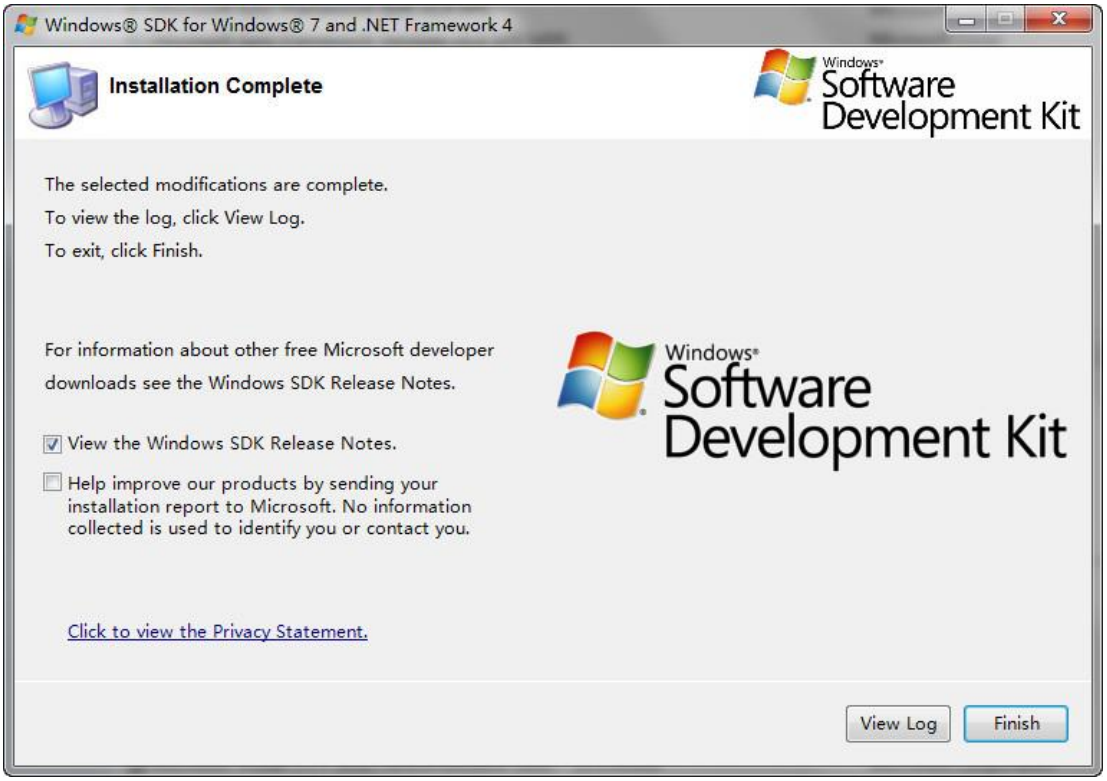

Figure 2- 20

2.2.2. Configuring the SDK in Visual Studio project

Take **Visual Studio 2010** as an example, configure the header file **C:\Program Files\Microsoft SDKs\Windows\v7.1\Include**:

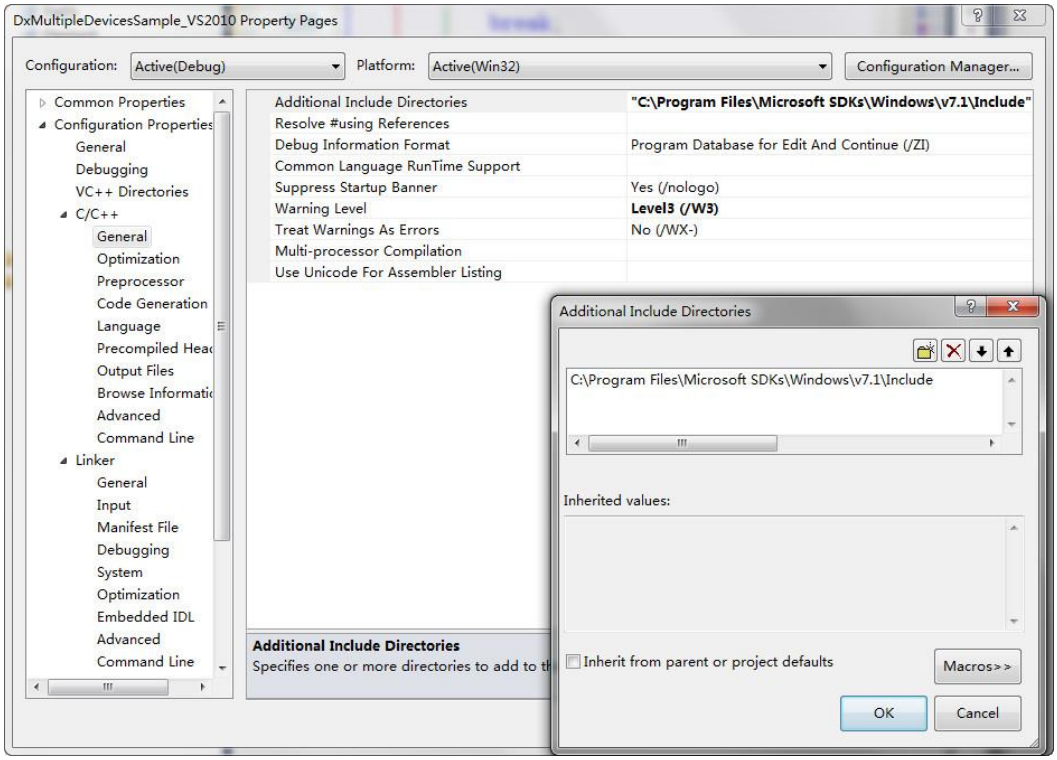

Figure 2- 21

Configuring the **lib** file path **C:\Program Files\Microsoft SDKs\Windows\v7.1\Lib**:

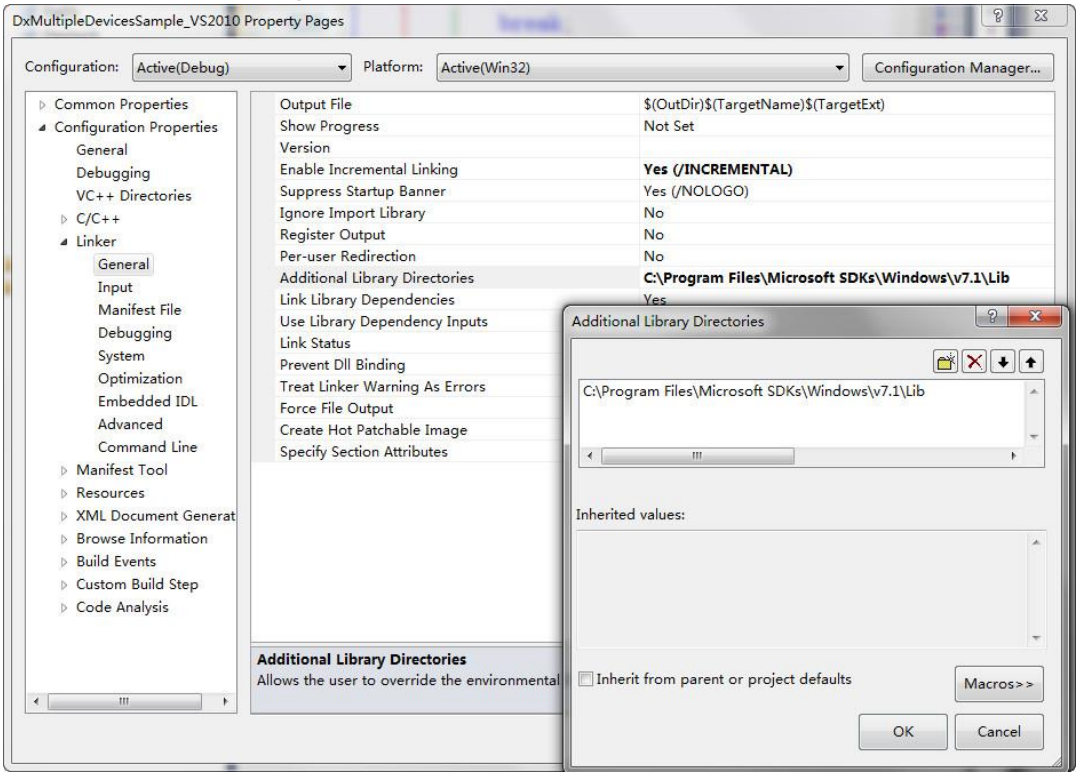

Figure 2- 22

Add a reference to **winmm.lib** under the **lib** file path:

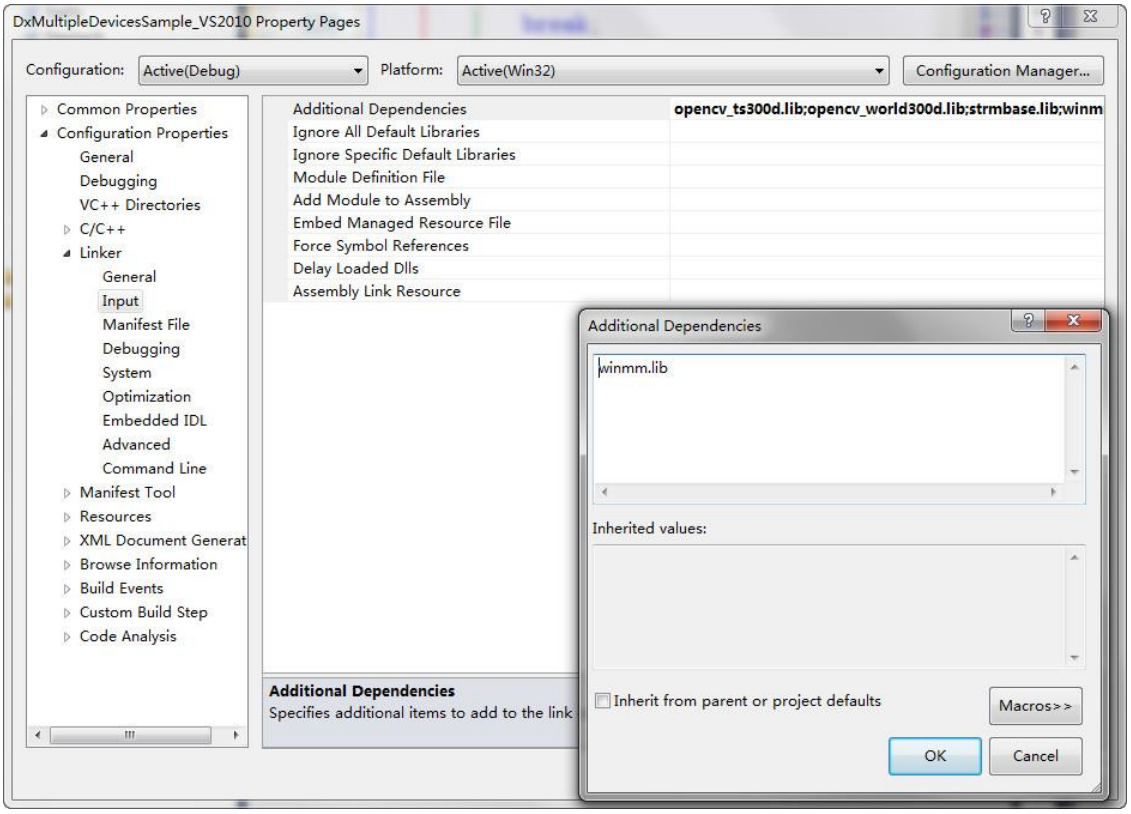

Figure 2- 23

In addition, configuring a reference to **strmbasd.lib** is needed. Use **Visual Studio** to open the **\Microsoft SDKs\Windows\v7.1\Samples\multimedia\directshow\baseclasses\baseclasses.sln** file, generating the corresponding version of **strmbasd.lib** as needed, and click the compile button:

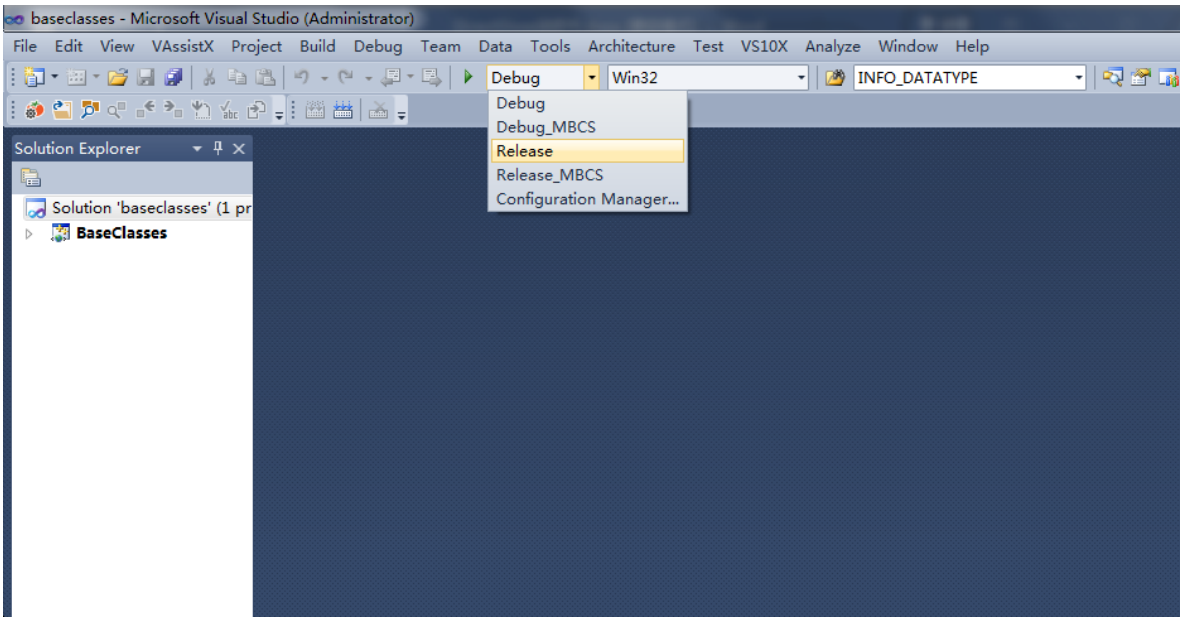

Figure 2- 24

Add the generated **strmbasd.lib** to your project and reference it:

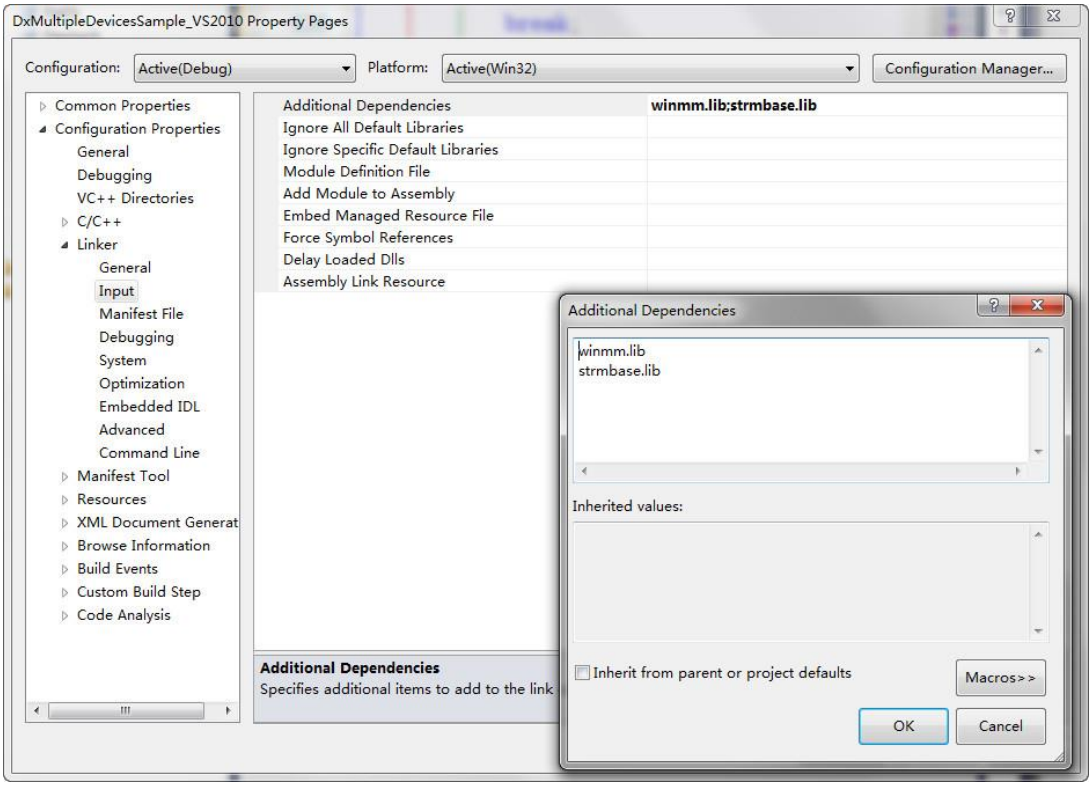

Figure 2- 25

At this point, the development environment configuration of **DirectShow** is completed.

3. Interface Process Introduction

The **IDHCamFilter** and **IDHCamPin** interfaces are declared in the **GXBase.h** header file.

3.1. IDHCamFilter Interface

3.1.1. IsColor

Statement:

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

Significance:
Get whether the current camera is a color camera.

Formal parameter:

```
[out]bIsColor    //Color camera flag bit
```

Return values:

```
S_OK              //Read flag bit successfully
E_NOTIMPL         //Interface not implemented
E_NOINTERFACE     //This interface is not supported
E_FAIL            //Unspecified failure
E_POINTER         //Invalid pointer
E_HANDLE          //Invalid handle
```

The error not covered above and uncommon error, please refer to **HRESULT**.

Sample code:

```
//----------------------------------------------------
/**
\brief                  //Get whether the current camera is a color
                         //camera from the device
\param  bColorFlag      //Output color flag bit
\return DEVICE_STATUS   //Output error code
*/
//----------------------------------------------------
DEVICE_STATUS CDevice::GetColorFromDevice(bool &bColorFlag)
{
      HRESULT                hResult     = S_OK;
      DEVICE_STATUS          emStatus    = DEVICE_GET_COLOR_FAIL;
      CComPtr<IDHCamFilter>  pCamFilter  = NULL;

      do
      {
         // Judge whether the current camera is open
         if (m_bOpenFlag == false)
         {
            break;
         }
```

```
            //Query IDHCamFilter interface in Filter
            hResult = m_pDeviceFilter->QueryInterface(IID_IDHCamFilter,
            (void **)&pCamFilter);
            VERIFY_HRESULT_PRINT_RETURN_STATUS(hResult);

            //Get the flag bit of color camera
            hResult = pCamFilter->IsColor(bColorFlag);
            VERIFY_HRESULT_PRINT_RETURN_STATUS(hResult);

            emStatus = DEVICE_SUCCESS;
        } while (0);

        //Release the resource of IDHCamFilter interface
        pCamFilter = NULL;

        return emStatus;
}
```

3.1.2. EnableColorCorrect

Statement:

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

Significance:

Enable or disable the color correction flag bit.

Formal parameter:

```
[in]bIsEnable    //Enable or disable the color correction flag bit
```

Return values:

```
S_OK             //Read flag bit successfully
E_NOTIMPL        //Interface not implemented
E_NOINTERFACE    //This interface is not supported
E_FAIL           //Unspecified failure
E_POINTER        //Invalid pointer
E_HANDLE         //Invalid handle
```

The error not covered above and uncommon error, please refer to **HRESULT**.

3.1.3. GetColorCorrectStatus

Statement:

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

Significance:

Get the enable state of color correction bit

Formal parameter:

```
[out]bIsEnable   //Get the enable state of color correction bit
```

Return values:

```
S_OK             //Read flag bit successfully
E_NOTIMPL        //Interface not implemented
E_NOINTERFACE    //This interface is not supported
E_FAIL           //Unspecified failure
E_POINTER        //Invalid pointer
E_HANDLE         //Invalid handle
```

The error not covered above and uncommon error, please refer to **HRESULT**.

3.1.4. SetSharpen

Statement:

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

Significance:
Set the current value of sharpening.

Formal parameters:

```
[in] bIsEnable   //Set the enable state of sharpening
[in] dValue      //Set the current value of sharpening
```

Return values:

```
S_OK             //Read flag bit successfully
E_NOTIMPL        //Interface not implemented
E_NOINTERFACE    //This interface is not supported
E_FAIL           //Unspecified failure
E_POINTER        //Invalid pointer
E_HANDLE         //Invalid handle
```

The error not covered above and uncommon error, please refer to **HRESULT**.

3.1.5. SetLightness

Statement:

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

Significance:
Set the current value of lightness.

Formal parameters:

```
[in] bIsEnabl    //Set the enable state of lightness
[in] dValue      //Set the current value of lightness
```

Return values:

```
S_OK              //Read flag bit successfully
E_NOTIMPL         //Interface not implemented
E_NOINTERFACE     //This interface is not supported
E_FAIL            //Unspecified failure
E_POINTER         //Invalid pointer
E_HANDLE          //Invalid handle
```

The error not covered above and uncommon error, please refer to **HRESULT**.

3.1.6. SetContrast

Statement:

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

Significance:
Set the current value of contrast.

Formal parameters:

```
[in] bIsEnable   //Set the enable state of contrast
[in] dValue      //Set the current value of contrast
```

Return values:

```
S_OK              //Read flag bit successfully
E_NOTIMPL         //Interface not implemented
E_NOINTERFACE     //This interface is not supported
E_FAIL            //Unspecified failure
E_POINTER         //Invalid pointer
E_HANDLE          //Invalid handle
```

The error not covered above and uncommon error, please refer to **HRESULT**.

3.1.7. SetSaturation

Statement:

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

Significance:
Set the current value of saturation.

Formal parameters:

```
[in] bIsEnable   //Set the enable state of saturation
[in] dValue      // Set the current value of saturation
```

Return values:

```
S_OK              //Read flag bit successfully
E_NOTIMPL         //Interface not implemented
E_NOINTERFACE     //This interface is not supported
E_FAIL            //Unspecified failure
```

```
E_POINTER          //Invalid pointer
E_HANDLE           //Invalid handle
```

The error not covered above and uncommon error, please refer to **HRESULT**.

3.1.8. SetGamma

Statement:

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

Significance:
Set the current value of Gamma.

Formal parameters:

```
[in] bIsEnable    //Set the enable state of Gamma
[in] dValue       //Set the current value of Gamma
```

Return values:

```
S_OK               //Read flag bit successfully
E_NOTIMPL          //Interface not implemented
E_NOINTERFACE      //This interface is not supported
E_FAIL             //Unspecified failure
E_POINTER          //Invalid pointer
E_HANDLE           //Invalid handle
```

The error not covered above and uncommon error, please refer to **HRESULT**.

3.1.9. GetSharpen

Statement:

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

significance:
Get the current value of sharpening.

Formal parameters:

```
[out] bIsEnable  //Get the enable flag bit of sharpening
[out] dValue     //Get the current value of sharpening
```

Return values:

```
S_OK               //Read flag bit successfully
E_NOTIMPL          //Interface not implemented
E_NOINTERFACE      //This interface is not supported
E_FAIL             //Unspecified failure
E_POINTER          //Invalid pointer
E_HANDLE           //Invalid handle
```

The error not covered above and uncommon error, please refer to **HRESULT**.

3.1.10. GetLightness

Statement:

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

significance:
Get the current value of lightness.

Formal parameters:

```
[out] bIsEnable  //Get the enable flag bit of lightness
[out] dValue     //Get the current value of lightness
```

Return values:

```
S_OK             //Read flag bit successfully
E_NOTIMPL        //Interface not implemented
E_NOINTERFACE    //This interface is not supported
E_FAIL           //Unspecified failure
E_POINTER        //Invalid pointer
E_HANDLE         //Invalid handle
```

The error not covered above and uncommon error, please refer to **HRESULT**.

3.1.11. GetContrast

Statement:

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

Significance:
Get the current value of contrast.

Formal parameters:

```
[out] bIsEnable  //Get the enable flag bit of contrast
[out] dValue     //Get the current value of contrast
```

Return values:

```
S_OK             //Read flag bit successfully
E_NOTIMPL        //Interface not implemented
E_NOINTERFACE    //This interface is not supported
E_FAIL           //Unspecified failure
E_POINTER        //Invalid pointer
E_HANDLE         //Invalid handle
```

The error not covered above and uncommon error, please refer to **HRESULT**.

3.1.12. GetSaturation

Statement:

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

Significance:
Get the current value of saturation.

Formal parameters:

```
[out] bIsEnable  //Get the enable flag bit of saturation
[out] dValue     //Get the current value of saturation
```

Return values:

```
S_OK             //Read flag bit successfully
E_NOTIMPL        //Interface not implemented
E_NOINTERFACE    //This interface is not supported
E_FAIL           //Unspecified failure
E_POINTER        //Invalid pointer
E_HANDLE         //Invalid handle
```

The error not covered above and uncommon error, please refer to **HRESULT**.

3.1.13. GetGamma

Statement:

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

Significance:
Get the current value of Gamma.

Formal parameters:

```
[out] bIsEnable  //Get the enable flag bit of Gamma
[out] dValue     //Get the current value of Gamma
```

Return values:

```
S_OK             //Read flag bit successfully
E_NOTIMPL        //Interface not implemented
E_NOINTERFACE    //This interface is not supported
E_FAIL           //Unspecified failure
E_POINTER        //Invalid pointer
E_HANDLE         //Invalid handle
```

The error not covered above and uncommon error, please refer to **HRESULT**.

3.1.14. GetPixelSize

Statement:

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

Significance:
Get the image pixel size of current device.

Formal parameter:

```
[out] emPixelSize//Get the current value of image pixel size
```

Return values:

```
S_OK              //Read flag bit successfully
E_NOTIMPL         //Interface not implemented
E_NOINTERFACE     //This interface is not supported
E_FAIL            //Unspecified failure
E_POINTER         //Invalid pointer
E_HANDLE          //Invalid handle
```

The error not covered above and uncommon error, please refer to **HRESULT**.

3.1.15. GetDevicePointer

Statement:

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

Significance:
Get the pointer of current device.

Formal parameter:

```
[out] pDevice    //Get the pointer of current device
```

Return values:

```
S_OK              //Read flag bit successfully
E_NOTIMPL         //Interface not implemented
E_NOINTERFACE     //This interface is not supported
E_FAIL            //Unspecified failure
E_POINTER         //Invalid pointer
E_HANDLE          //Invalid handle
```

The error not covered above and uncommon error, please refer to **HRESULT**.

3.1.16. IsConnect

Statement:

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

Significance:
Judge whether the current Pin is connected

Formal parameter:

```
[out] bIsConnected//Get the connection flag bit of current Pin
```

Return values:

```
S_OK               //Read flag bit successfully
E_NOTIMPL          //Interface not implemented
E_NOINTERFACE      //This interface is not supported
E_FAIL             //Unspecified failure
```

```
E_POINTER          //Invalid pointer
E_HANDLE           //Invalid handle
```

The error not covered above and uncommon error, please refer to **HRESULT**.

3.2. IDHCamPin Interface

3.2.1. GetCurrentDeviceIndex

Statement:

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

Significance:
Get the index value of current device.

Formal parameter:

```
[out] plDevice   //Get the index value of current device
```

Return values:

```
S_OK               //Read flag bit successfully
E_NOTIMPL          //Interface not implemented
E_NOINTERFACE      //This interface is not supported
E_FAIL             //Unspecified failure
E_POINTER          //Invalid pointer
E_HANDLE           //Invalid handle
```

The error not covered above and uncommon error, please refer to **HRESULT**.

Sample code:

```
//-----------------------------------------------------
/**
\brief                  //Get the index value of current device
\param   plDevice       //Output the index value of current device
\return  DEVICE_STATUS  //Output error code
*/
//-----------------------------------------------------
DEVICE_STATUS CDevice::GetDeviceIndexFromDevice(long &plDevice)
{
      HRESULT              hResult = S_OK;
      DEVICE_STATUS emStatus=DEVICE_GET_CURRENT_DEVICE_INDEX_FAIL;
      CComPtr<IDHCamPin>  pCamPin = NULL;

      do
      {
       // Query IDHCamPin interface in Filter
        hResult   =   m_pDeviceFilter->QueryInterface(IID_IDHCamPin,
        (void **)&pCamPin);
         VERIFY_HRESULT_PRINT_RETURN_STATUS(hResult);
```

```
        //Get the index value of the current device in the device list
            hResult = pCamPin->GetCurrentDeviceIndex(plDevice);
            VERIFY_HRESULT_PRINT_RETURN_STATUS(hResult);

            emStatus = DEVICE_SUCCESS;
        } while (0);

        //Release the resource of IDHCamPin interface
        pCamPin = NULL;

        return emStatus;
}
```

3.2.2. SetCurrentDeviceIndex

Statement:

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

Significance:
Set the index value of current device.

Formal parameter:

```
[in] lDevice      //Set the index value of current device
```

Return values:

```
S_OK              //Read flag bit successfully
E_NOTIMPL         //Interface not implemented
E_NOINTERFACE     //This interface is not supported
E_FAIL            //Unspecified failure
E_POINTER         //Invalid pointer
E_HANDLE          //Invalid handle
```

The error not covered above and uncommon error, please refer to **HRESULT**.

3.2.3. GetDeviceList

Statement:

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

Significance:
Get device information in the current device list.

Formal parameters:

```
[out] pEnumDescription //Output the information structure of device,
                       //including device serial number, device name
[out]pBufferSizet     //Output the number of device
```

Return values:

```
S_OK               //Read flag bit successfully
E_NOTIMPL          //Interface not implemented
E_NOINTERFACE      //This interface is not supported
E_FAIL             //Unspecified failure
E_POINTER          //Invalid pointer
E_HANDLE           //Invalid handle
```

The error not covered above and uncommon error, please refer to **HRESULT**.

4. FAQ

No.	General Question	Answer
1	Note to **HALCON**: when the number of registered Daheng devices is larger than the number of connected devices, a message box will be displayed indicating that the device cannot be found.	1) Make the number of registered Daheng cameras consistent with the number of connected devices.
2	Note to **MATLAB**: use **MATLAB** to open the connected device, prompt: **No Image Acquisition adaptors found. Image acquisition adaptors may be available as downloadable support packages. Open Support Package Installer to install additional vendors.**	1) Click on **Support Package Installer**, **MATLAB** will provide about 13 packages, then choose **OS Generic Video Interface** to download and install.

5. Revision History

No.	Version	Changes	Data
1	V1.0.0	Initial release	2018-08-01