

MFC (VS2010) + Vimba API 编程实例

说明：本文档适合于初次使用 Vimba C++API 做 AVT 相机二次开发的工程人员。通过本文档的介绍，可以轻松建立一个调用 Vimba C++API 的 MFC 程序，或者将官方例程移植到您既有的项目中

1. 准备工作

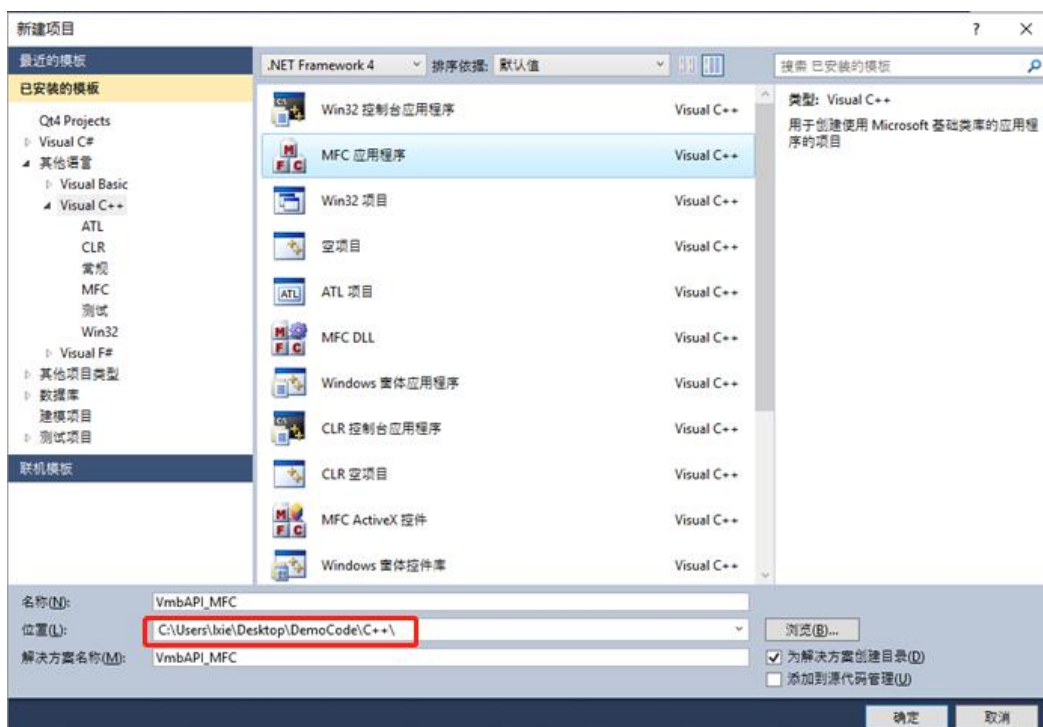
1.1 首先，确认已完全安装 Vimba SDK



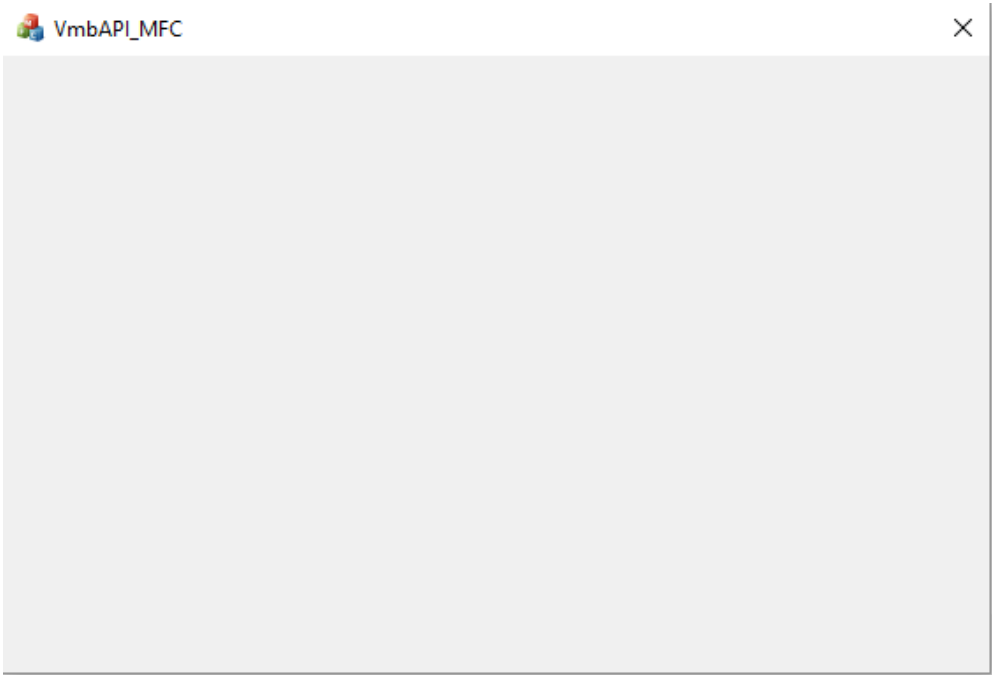
安装可以参考：

<https://github.com/avtcn/notes/blob/master/skills/AVT%E9%87%87%E5%9B%BE%E8%BD%AF%E4%BB%B6%E5%85%A5%E9%97%A8%E6%8C%87%E5%8D%97.pdf>。请留意“Target Folder”以及“Examples Target Folder”的位置

1.2 新建一个 MFC 对话框工程，命名为“VmbAPI_MFC”



1.3 删掉对话框界面不必要的的控件，并启动程序

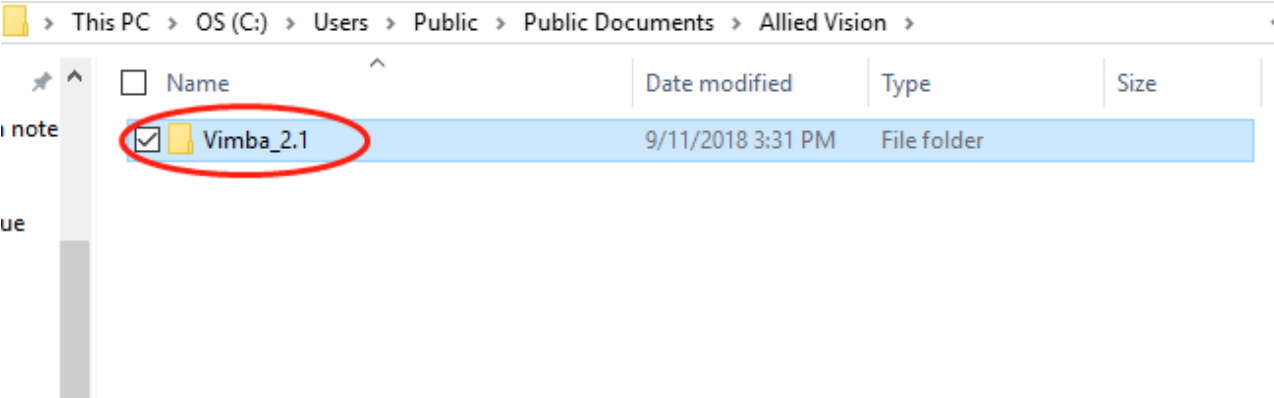


启动后再停止，进行下一步

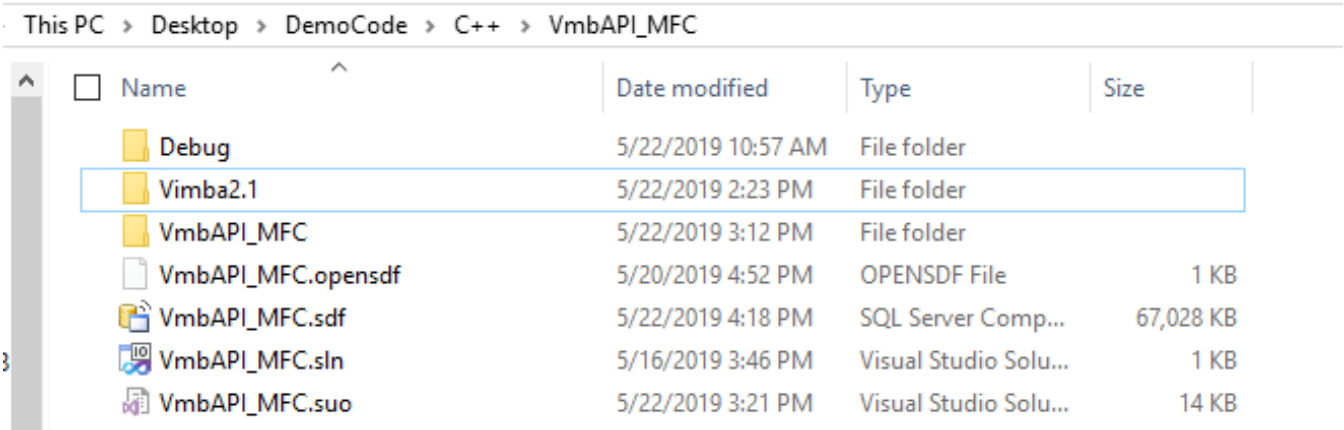
2. 调用 VmbAPI 相关的库

2.1 将 VmbAPI 相关的文件拷到工程文件夹（这里主要是为了后面采用相对引用，方便程序的移植）

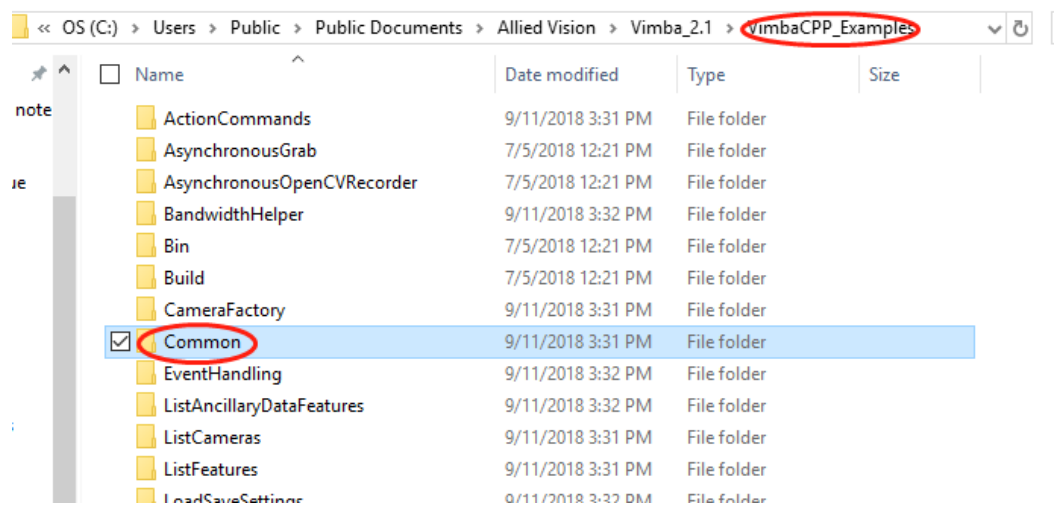
2.1.1 复制“Target Folder”、即“Vimba_2.1”文件夹并粘贴到当前工程文件夹里



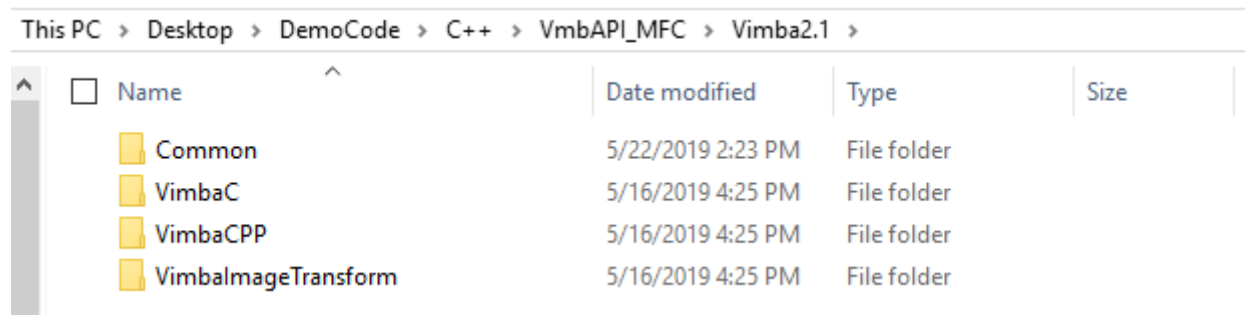
并且命名为“Vimba2.1”（个人习惯）



2.1.2 再将“Examples Target Folder”里的“Common”文件夹复制到已拷贝进来的“Vimba2.1”文件夹

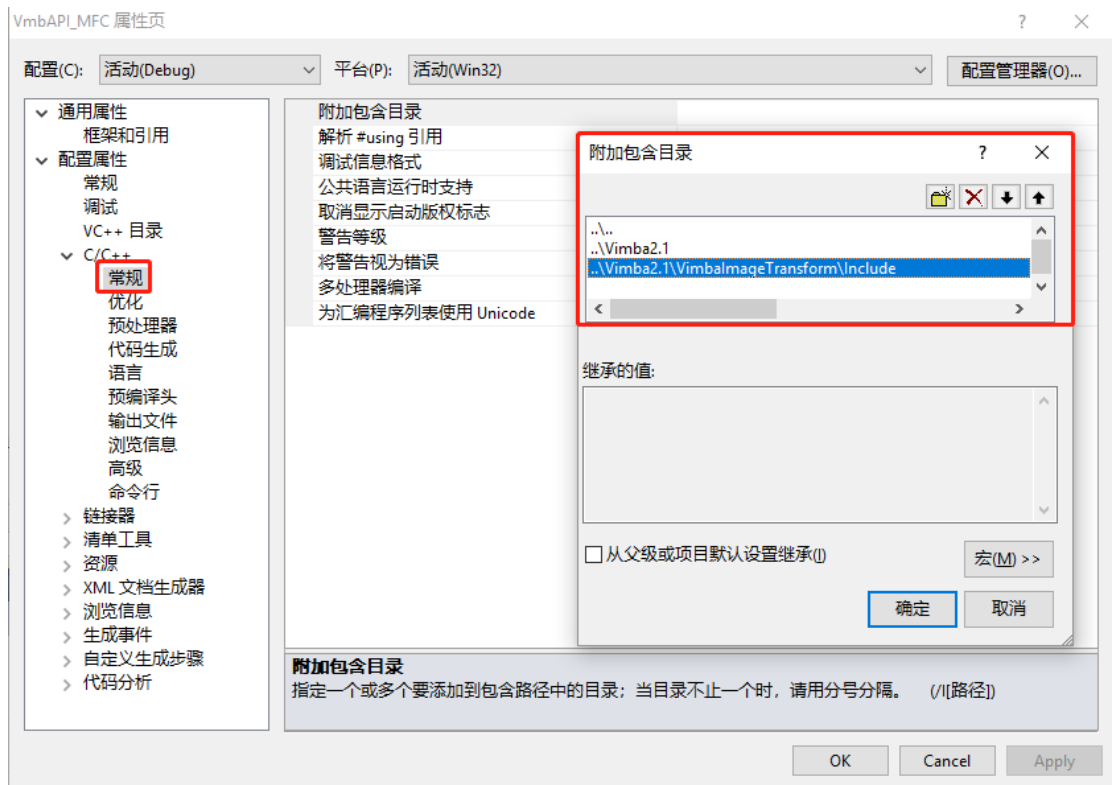


2.1.3 精简拷贝进来的“Vimba2.1”文件夹，将里面不必要的文件删除，仅保留如下四个文件夹：

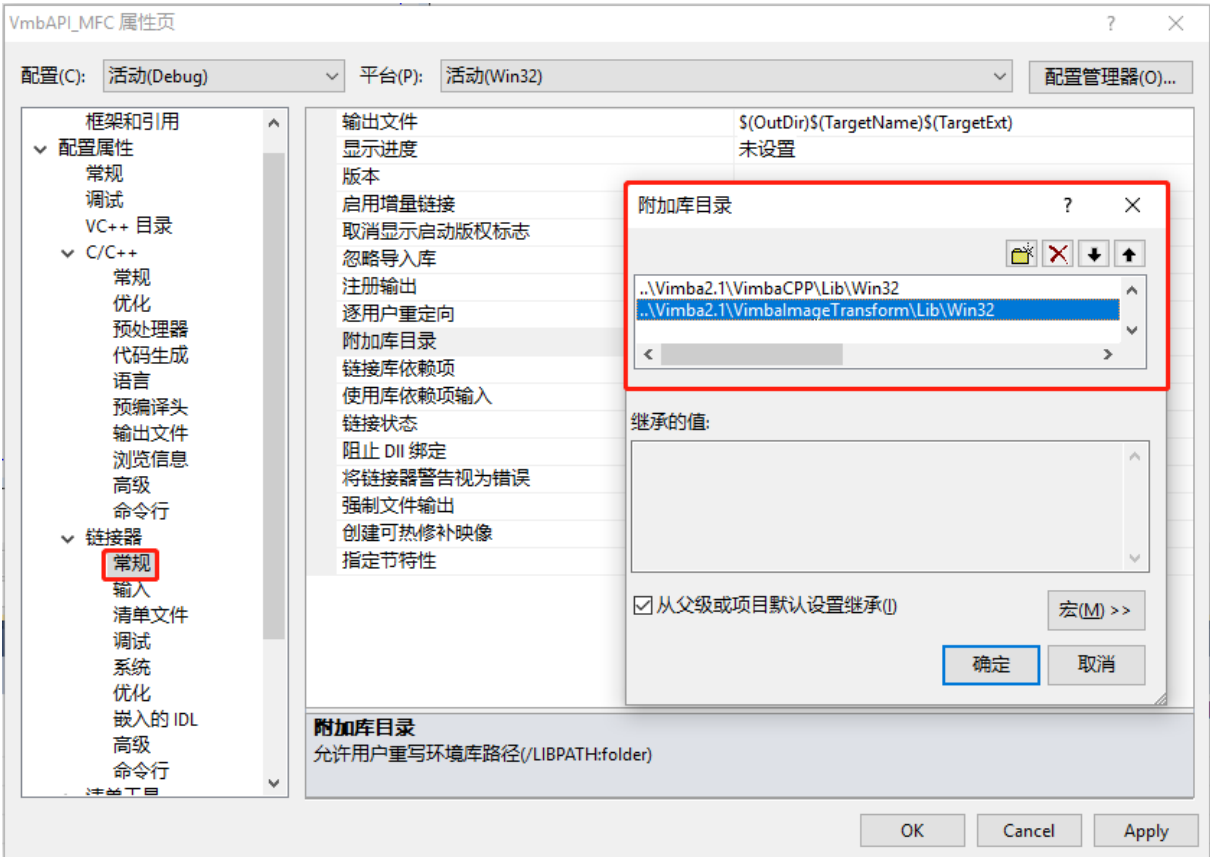


2.2 链接 VmbAPI 库文件

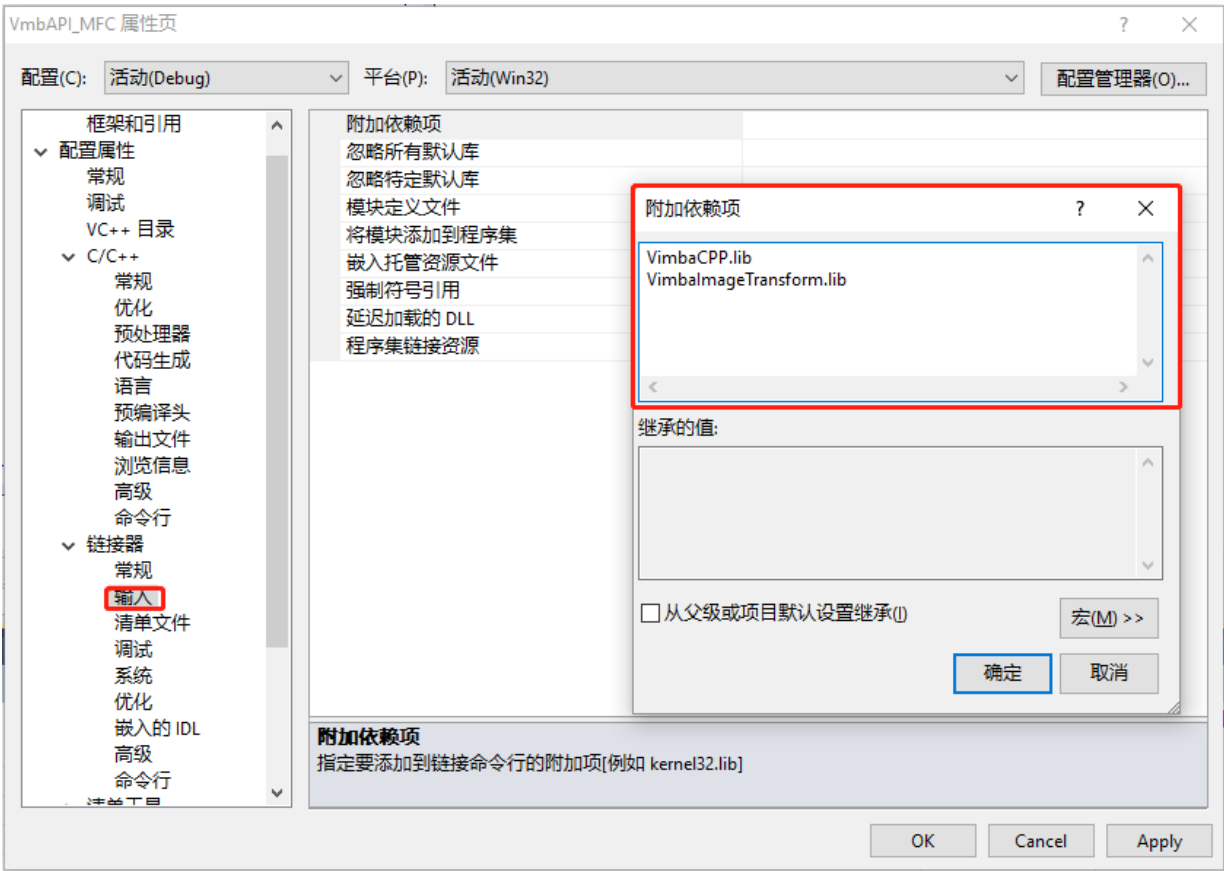
2.2.1 设置“附加包含目录”



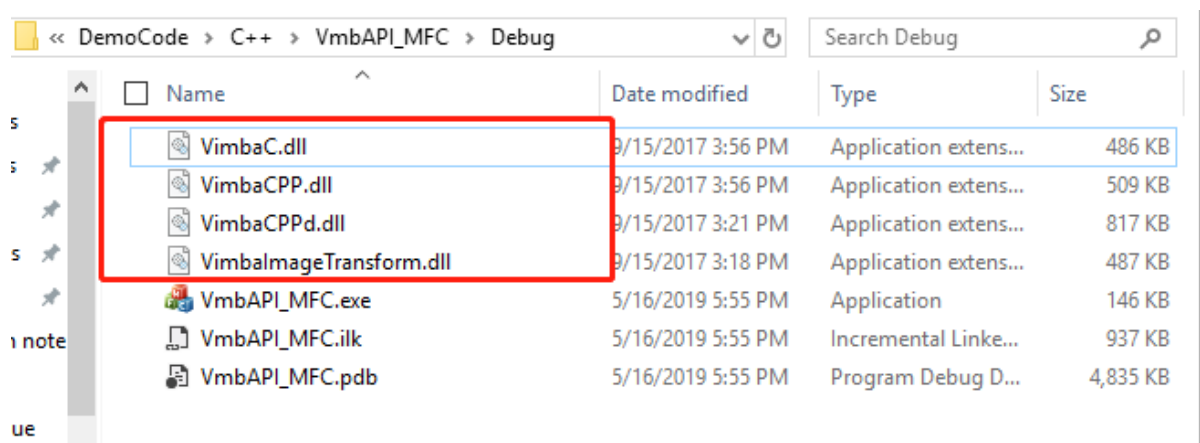
2.2.2 设置“附加库目录” (这里以 Win32 程序为例，Win32 程序在 X64 平台上也能运行，如果纯 X64 程序，则引用库目录改成 Win64 就可以了)



2.2.3 设置“附加依赖项”

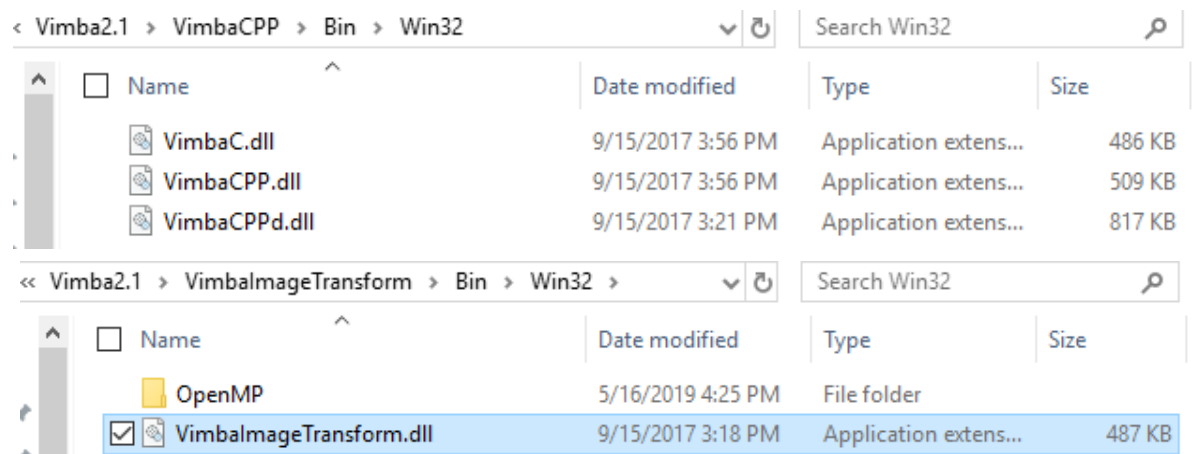


2.2.4 将必须的 4 个动态链接库（dll 文件）拷贝到"exe"所在的文件夹内



Name	Date modified	Type	Size
VimbaC.dll	9/15/2017 3:56 PM	Application extens...	486 KB
VimbaCPP.dll	9/15/2017 3:56 PM	Application extens...	509 KB
VimbaCPPd.dll	9/15/2017 3:21 PM	Application extens...	817 KB
VimbalmageTransform.dll	9/15/2017 3:18 PM	Application extens...	487 KB
VmbAPI_MFC.exe	5/16/2019 5:55 PM	Application	146 KB
VmbAPI_MFC.ilk	5/16/2019 5:55 PM	Incremental Linke...	937 KB
VmbAPI_MFC.pdb	5/16/2019 5:55 PM	Program Debug D...	4,835 KB

这四个文件可以从 Vimba2.1 文件夹对应的 Win32 平台文件夹里找到，对应的 x64 程序也有对应的文件夹



First screenshot: < Vimba2.1 > VimbaCPP > Bin > Win32

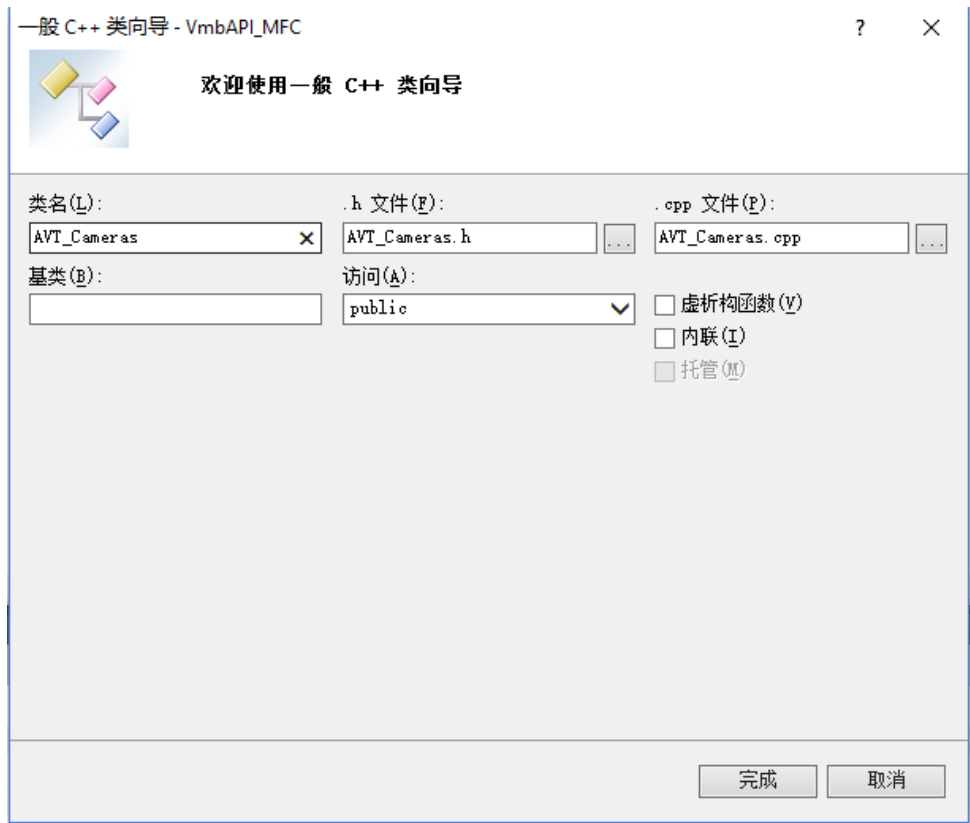
Name	Date modified	Type	Size
VimbaC.dll	9/15/2017 3:56 PM	Application extens...	486 KB
VimbaCPP.dll	9/15/2017 3:56 PM	Application extens...	509 KB
VimbaCPPd.dll	9/15/2017 3:21 PM	Application extens...	817 KB

Second screenshot: << Vimba2.1 > VimbalmageTransform > Bin > Win32 >

Name	Date modified	Type	Size
OpenMP	5/16/2019 4:25 PM	File folder	
VimbalmageTransform.dll	9/15/2017 3:18 PM	Application extens...	487 KB

3. 启动程序调用 VmbAPI，并输出当前版本号

3.1 新建一个相机控制类，命名为“AVT_Cameras”



一般 C++ 类向导 - VmbAPI_MFC

欢迎使用一般 C++ 类向导

类名(L): AVT_Cameras

.h 文件(F): AVT_Cameras.h

.cpp 文件(P): AVT_Cameras.cpp

基类(B):

访问(A): public

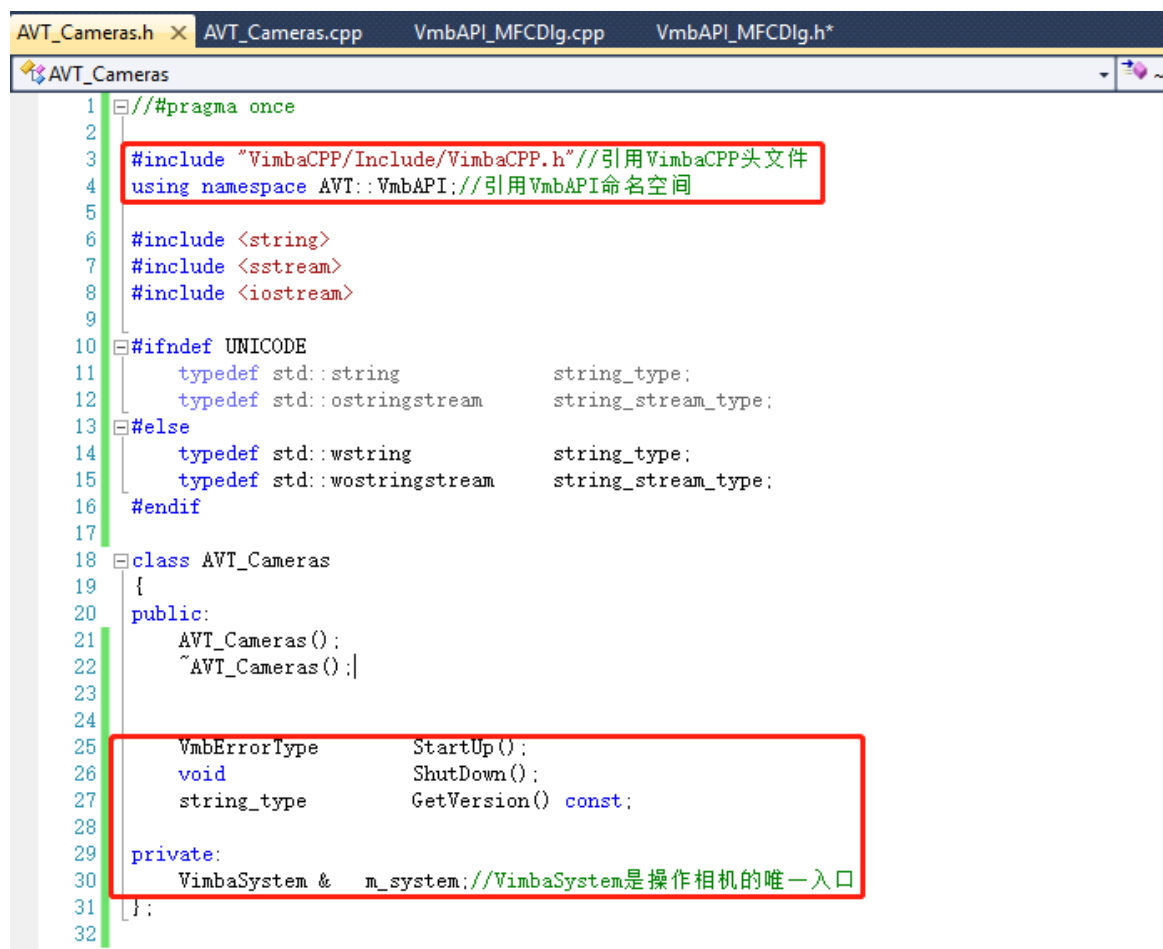
☐ 虚析构函数(V)

☐ 内联(I)

☐ 托管(M)

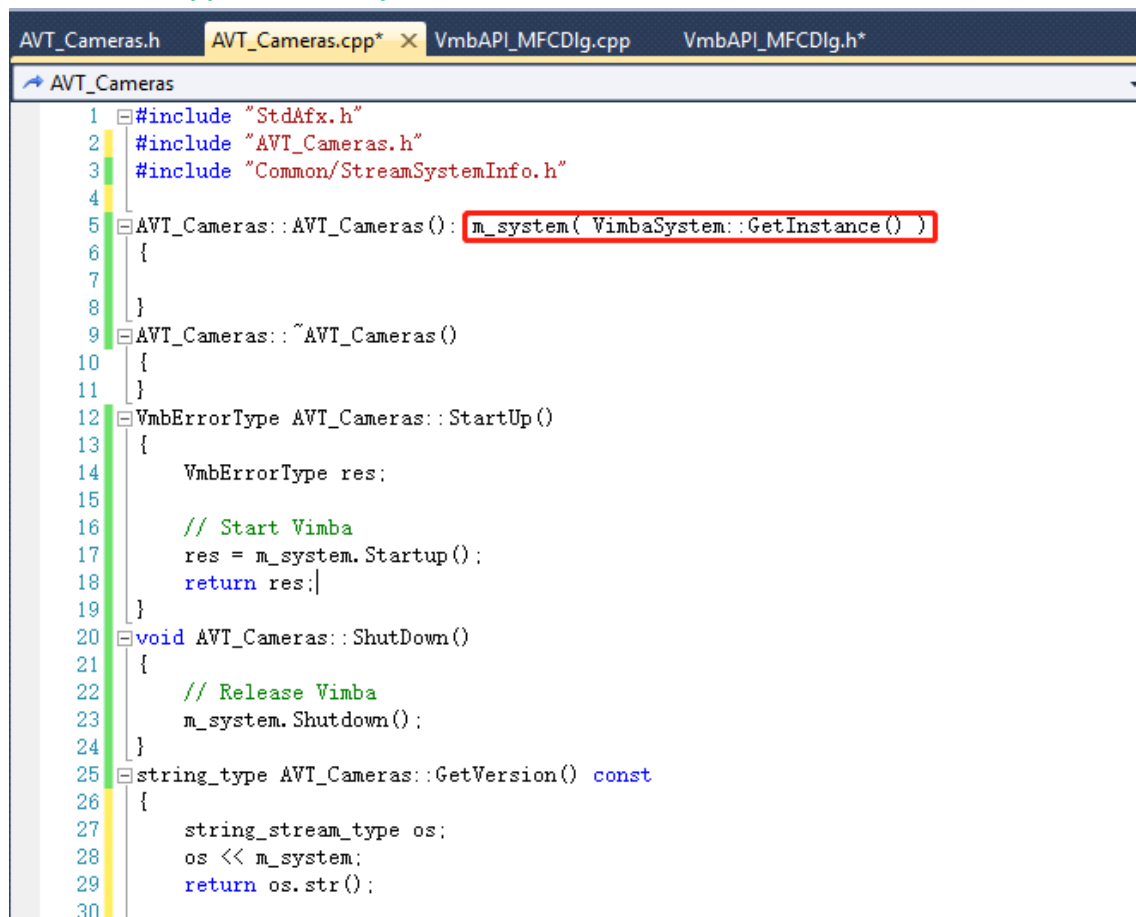
完成 取消

3.1.1 头文件“AVT_Cameras.h”做如下改动



```
1 // #pragma once
2
3 #include "VimbaCPP/Include/VimbaCPP.h" // 引用 VimbaCPP 头文件
4 using namespace AVT::VmbAPI; // 引用 VmbAPI 命名空间
5
6 #include <string>
7 #include <sstream>
8 #include <iostream>
9
10 #ifndef UNICODE
11     typedef std::string string_type;
12     typedef std::ostringstream string_stream_type;
13 #else
14     typedef std::wstring string_type;
15     typedef std::wostringstream string_stream_type;
16 #endif
17
18 class AVT_Cameras
19 {
20 public:
21     AVT_Cameras();
22     ~AVT_Cameras();
23
24
25     VmbErrorType StartUp();
26     void ShutDown();
27     string_type GetVersion() const;
28
29 private:
30     VimbaSystem & m_system; // VimbaSystem 是操作相机的唯一入口
31 };
32
```

3.1.2 “AVT_Cameras.cpp”实例化“m_system”,并实现对应的方法



```
1 #include "StdAfx.h"
2 #include "AVT_Cameras.h"
3 #include "Common/StreamSystemInfo.h"
4
5 AVT_Cameras::AVT_Cameras(): m_system( VimbaSystem::GetInstance() )
6 {
7 }
8
9 AVT_Cameras::~AVT_Cameras()
10 {
11 }
12
13 VmbErrorType AVT_Cameras::StartUp()
14 {
15     VmbErrorType res;
16
17     // Start Vimba
18     res = m_system.Startup();
19     return res;
20 }
21
22 void AVT_Cameras::ShutDown()
23 {
24     // Release Vimba
25     m_system.Shutdown();
26 }
27
28 string_type AVT_Cameras::GetVersion() const
29 {
30     string_stream_type os;
31     os << m_system;
32     return os.str();
33 }
```

3.2 调用相机控制类

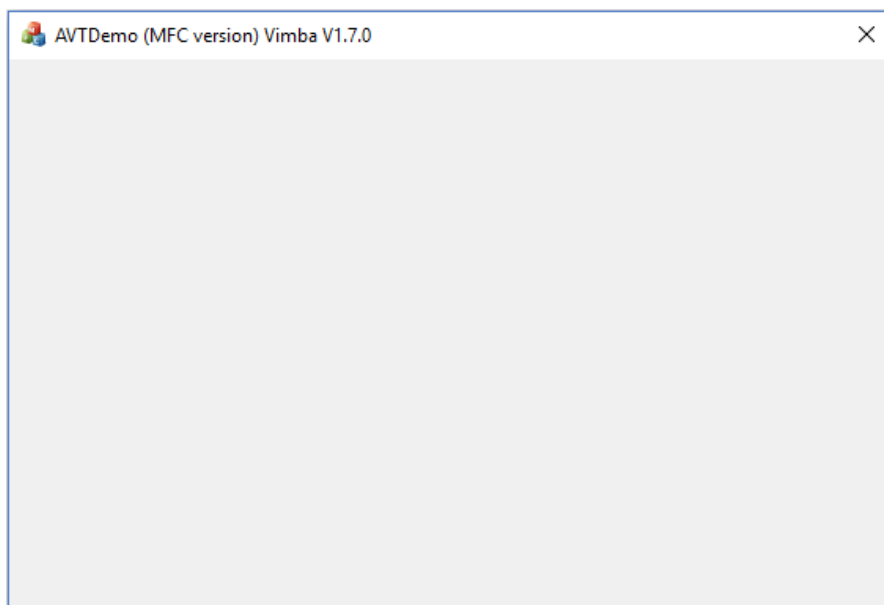
3.2.1 回到主界面的头文件“VmbAPI_MFCDlg.h”，并注意如下改动：

```
AVT_Cameras.h  AVT_Cameras.cpp*  VmbAPI_MFCDlg.cpp  VmbAPI_MFCDlg.h* X
CVmbAPI_MFCDlg
4
5 #pragma once
6
7 #include <string>
8 #include <iostream>
9 #include "AVT_Cameras.h"//引用AVT相机控制类的头文件
10
11 // CVmbAPI_MFCDlg 对话框
12
13 class CVmbAPI_MFCDlg : public CDialogEx
14 {
15 // 构造
16 public:
17     CVmbAPI_MFCDlg(CWnd* pParent = NULL);    // 标准构造函数
18
19 // 对话框数据
20     enum { IDD = IDD_VMBAPI_MFC_DIALOG };
21
22     protected:
23     virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV 支持
24
25 private:
26
27     AVT_Cameras m_AVT_Cameras;//实例化一个AVT相机控制类
28
29 // 实现
30 protected:
31     HICON m_hIcon;
32
33     // 生成的消息映射函数
34     virtual BOOL OnInitDialog();
35     afx_msg void OnPaint();
36     afx_msg void OnSysCommand(UINT nID, LPARAM lParam);//关闭时释放Vimba对象
37     afx_msg HCURSOR OnQueryDragIcon();
38     DECLARE_MESSAGE_MAP()
39 }
```

3.2.2 主界面“VmbAPI_MFCDlg.cpp”实现“启动 vimba”->“显示版本”->“退出”

```
AVT_Cameras.h  AVT_Cameras.cpp  VmbAPI_MFCDlg.cpp* X  VmbAPI_MFCDlg.h
CVmbAPI_MFCDlg
35
36 BEGIN_MESSAGE_MAP(CVmbAPI_MFCDlg, CDialogEx)
37     ON_WM_PAINT()
38     ON_WM_SYSCOMMAND()//设置关闭时的动作
39     ON_WM_QUERYDRAGICON()
40 END_MESSAGE_MAP()
41
42
43 // CVmbAPI_MFCDlg 消息处理程序
44
45 BOOL CVmbAPI_MFCDlg::OnInitDialog()
46 {
47     CDialogEx::OnInitDialog();
48
49     // 设置此对话框的图标。当应用程序主窗口不是对话框时，框架将自动
50     // 执行此操作
51     SetIcon(m_hIcon, TRUE);        // 设置大图标
52     SetIcon(m_hIcon, FALSE);     // 设置小图标
53
54     // TODO: 在此添加额外的初始化代码
55     VmbErrorType err = m_AVT_Cameras.StartUp();
56     string_type DialogTitle( _TEXT( "AVIDemo (MFC version) Vimba V" ) );
57     SetWindowText( ( DialogTitle+m_AVT_Cameras.GetVersion() ).c_str() );
58
59     return TRUE;    // 除非将焦点设置到控件，否则返回 TRUE
60 }
61
62 声明
63 void CVmbAPI_MFCDlg::OnSysCommand(UINT nID, LPARAM lParam)
64 {
65     if ( SC_CLOSE == nID )
66     {
67         // Before we close the application we stop Vimba
68         m_AVT_Cameras.ShutDown();//关闭Vimba对象
69     }
70
71     CDialog::OnSysCommand(nID, lParam);
72 }
```

运行结果如下：



以上代码可以通过如下地址下载：

https://github.com/avtcn/notes/blob/master/vimbasdk/VmbCPP/VmbAPI_MFC_Simple.zip

-----分--割--线-----

接下来就可以再做一些相机操作相关的细化流程了，下面我们决定在当前程序的基础上、直接移植官方例子程序的代码

4. 官方例程代码移植

4.1 准备工作（如果是自己的既有工程，请先参考“2.调用 VmbAPI 相关的库”）

4.1.1 将官方异步采图例程的“Source”文件夹拷贝到当前程序文件夹（“Vimba2.1”同一目录）

This PC > OS (C:) > Users > Public > Public Documents > Allied Vision > Vimba_2.1 > VimbaCPP_Examples > AsynchronousGrab > MFC

<input type="checkbox"/>	Name	Date modified	Type	Size
<input type="checkbox"/>	Build	1/22/2018 2:45 PM	File folder	
<input checked="" type="checkbox"/>	Source	3/11/2019 5:32 PM	File folder	
<input type="checkbox"/>	Screenshot.png	9/15/2017 3:11 PM	PNG image	583 KB

This PC > Desktop > DemoCode > C++ > VmbAPI_MFC

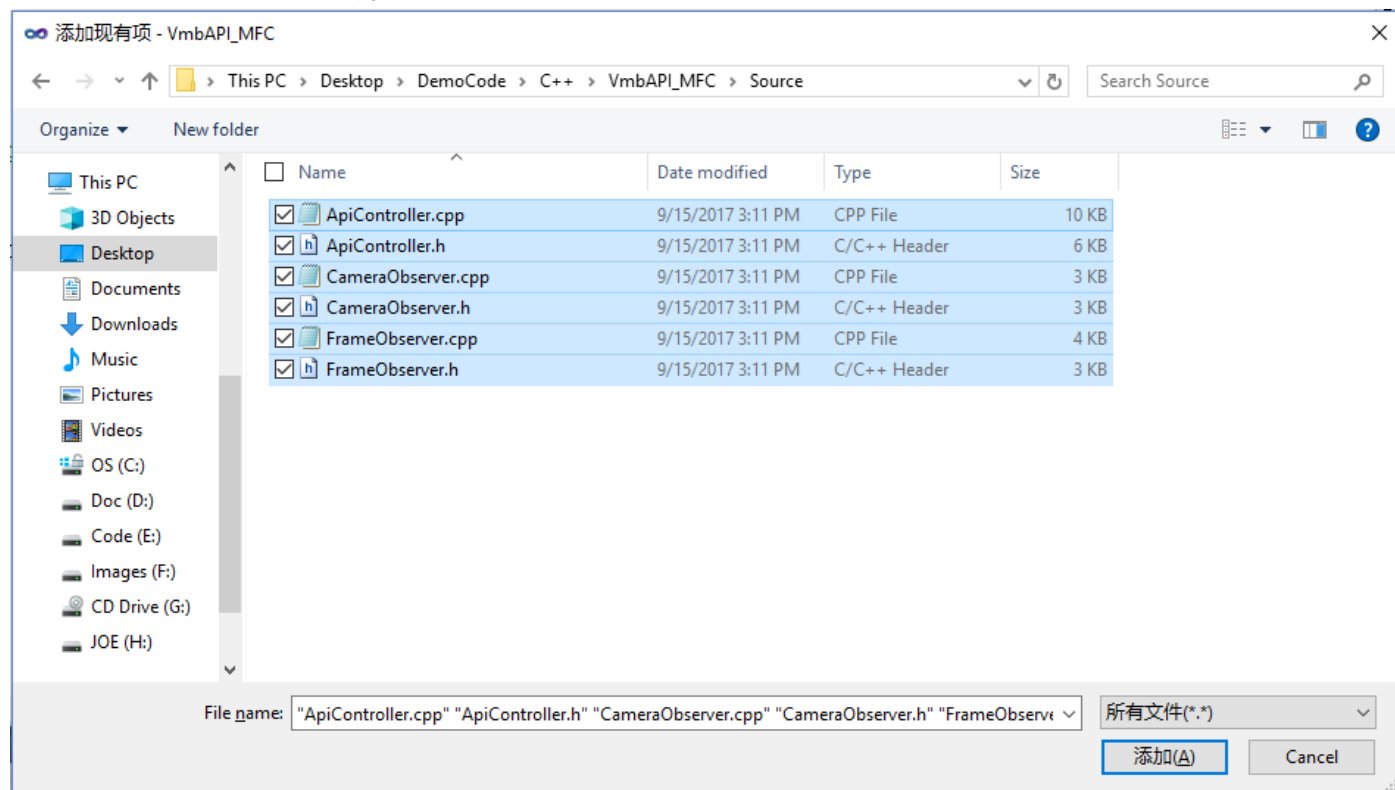
<input type="checkbox"/>	Name	Date modified	Type	Size
<input type="checkbox"/>	Debug	5/22/2019 10:57 AM	File folder	
<input checked="" type="checkbox"/>	Source	5/22/2019 5:40 PM	File folder	
<input type="checkbox"/>	Vimba2.1	5/22/2019 2:23 PM	File folder	
<input type="checkbox"/>	VmbAPI_MFC	5/22/2019 3:12 PM	File folder	
<input type="checkbox"/>	VmbAPI_MFC.sln	5/16/2019 3:46 PM	Visual Studio Solu...	1 KB
<input type="checkbox"/>	VmbAPI_MFC.suo	5/22/2019 5:21 PM	Visual Studio Solu...	19 KB

4.1.2 精简拷贝进来的“Source”文件夹，保留如下文件：

This PC > Desktop > DemoCode > C++ > VmbAPI_MFC > Source

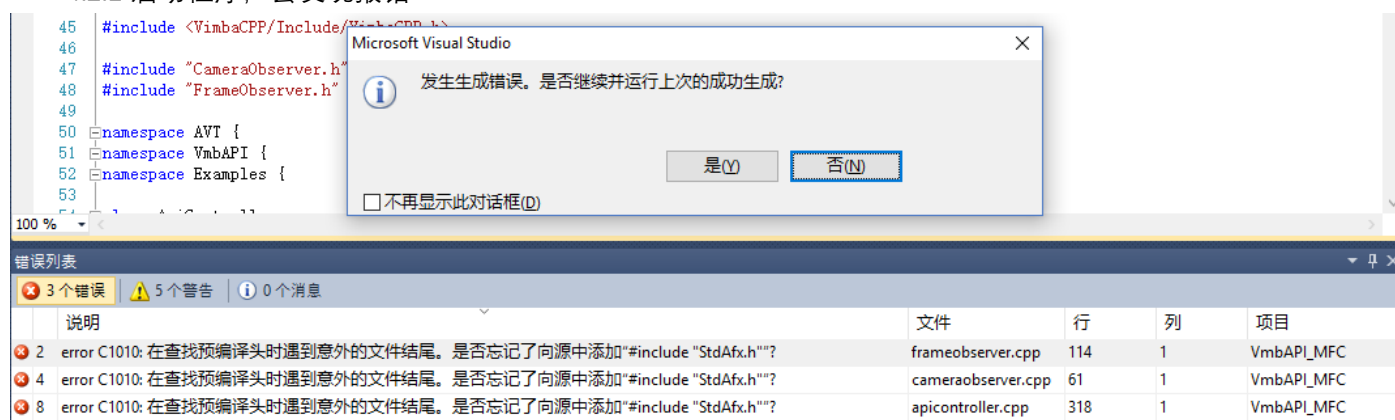
<input type="checkbox"/>	Name	Date modified	Type	Size
<input type="checkbox"/>	ApiController.cpp	9/15/2017 3:11 PM	CPP File	10 KB
<input type="checkbox"/>	ApiController.h	9/15/2017 3:11 PM	C/C++ Header	6 KB
<input type="checkbox"/>	CameraObserver.cpp	9/15/2017 3:11 PM	CPP File	3 KB
<input type="checkbox"/>	CameraObserver.h	9/15/2017 3:11 PM	C/C++ Header	3 KB
<input type="checkbox"/>	FrameObserver.cpp	9/15/2017 3:11 PM	CPP File	4 KB
<input type="checkbox"/>	FrameObserver.h	9/15/2017 3:11 PM	C/C++ Header	3 KB

4.1.3 再将上述文件添加到当前工程

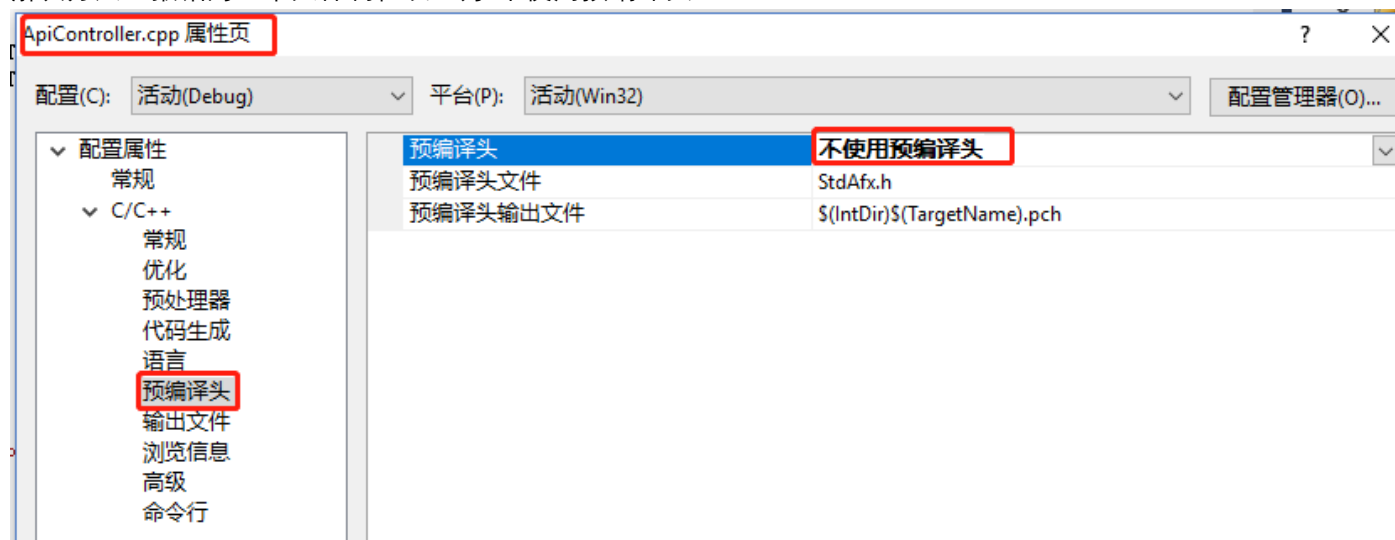


4.2 运行程序，检查移植是否成功

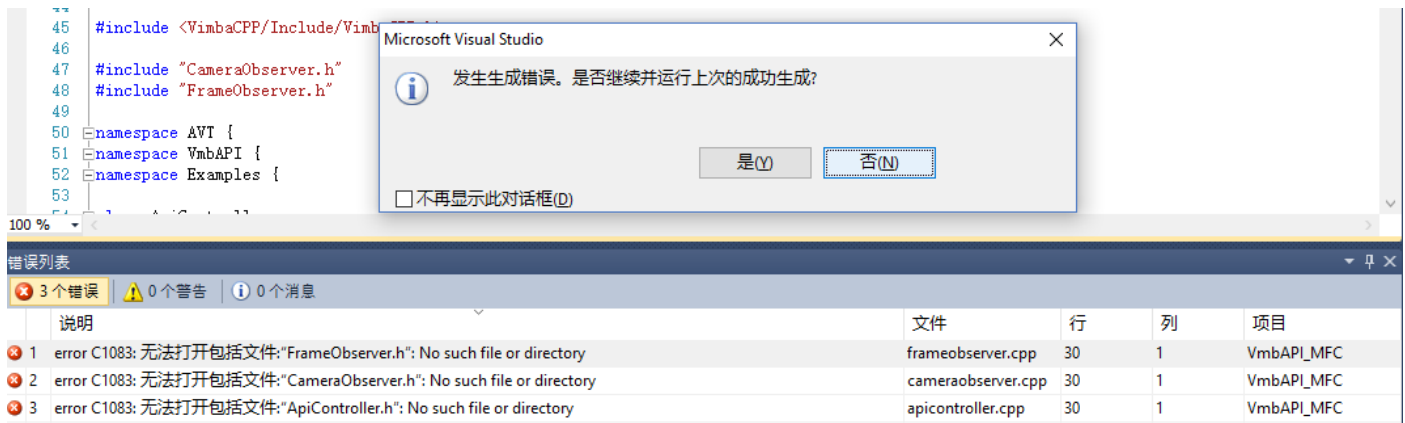
4.2.1 启动程序，会发现报错：



解决办法：报错的三个文件属性设置为“不使用预编译头”

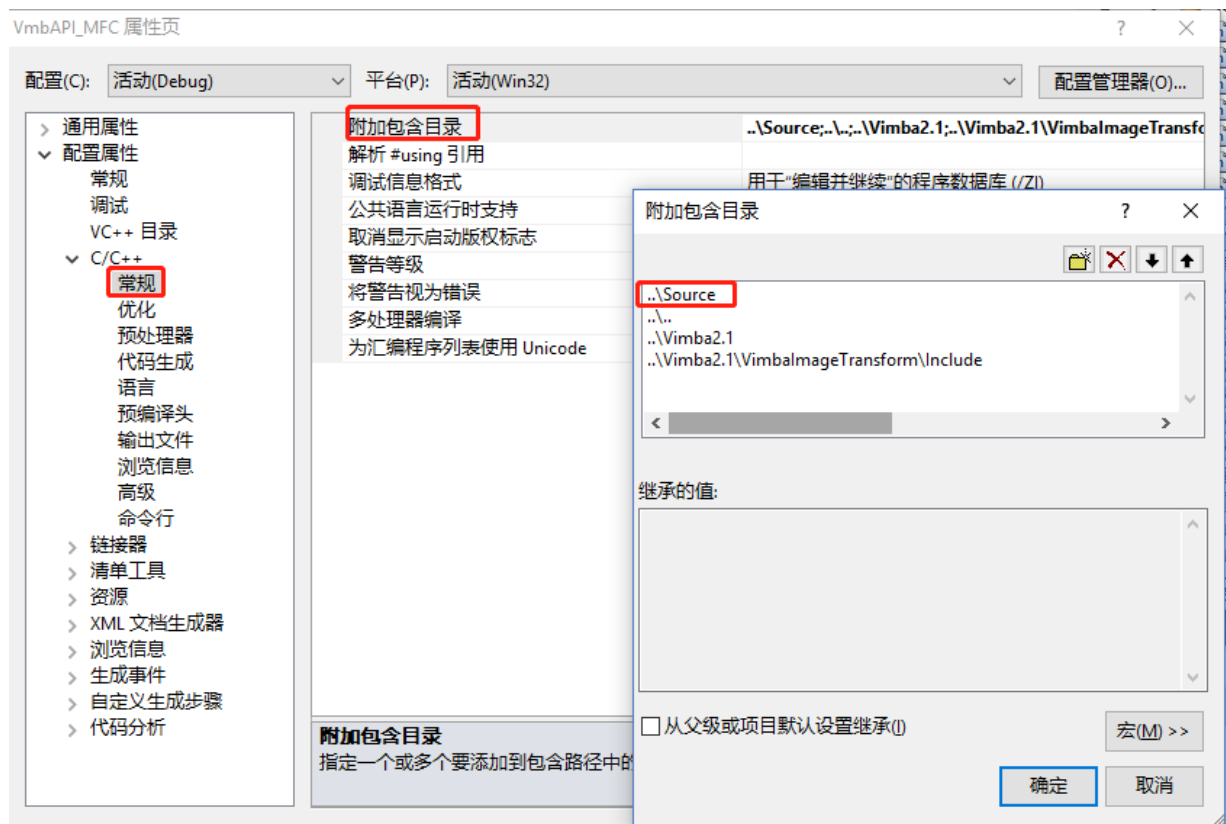


4.2.2 再次启动程序，会报另一个错误



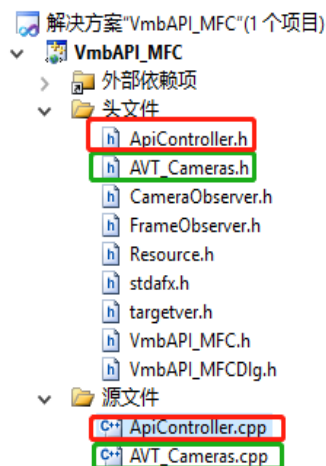
解决办法：附加包含目录增加拷贝进来的“Source”文件夹

(这里说明一下，如果步骤 4.1.1 直接把 4.1.3 需要的 6 个文件拷到与“VmbAPI_MFC.h”同一目录下，则不会报相关错误)



4.3 调用移植过来的代码

4.3.1 简单说明：例程移植过来的“ApiController”类的定位和原有的“AVT_Cameras”类是一样的



4.3.2 删除“AVT_Cameras”类，并引用“ApiController”

```

VmbAPI_MFCDlg.h  ApiController.cpp  ApiController.h  VmbAPI_MFCDlg.cpp
CVmbAPI_MFCDlg
4
5 #pragma once
6
7 #include <string>
8 #include <iostream>
9
10 // #include "AVT_Cameras.h" // 引用AVT相机控制类的头文件
11 #include "ApiController.h"
12 using AVT::VmbAPI::Examples::ApiController;
13
14 // CVmbAPI_MFCDlg 对话框
15
16 class CVmbAPI_MFCDlg : public CDialogEx
17 {
18 // 构造
19 public:
20     CVmbAPI_MFCDlg(CWnd* pParent = NULL);    // 标准构造函数
21
22 // 对话框数据
23     enum { IDD = IDD_VMBAPI_MFC_DIALOG };
24
25     protected:
26     virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV 支持
27
28 private:
29
30     // AVT_Cameras m_AVT_Cameras; // 实例化一个AVT相机控制类
31     ApiController m_AVT_Cameras;
32
33 // 实现

```

4.4 接下来就是主界面添加一些控件、以及调用”ApiController”类的一些方法，这里不再详细描述最终实现的效果如下图：



代码下载地址:

https://github.com/avtcn/notes/blob/master/vimbasdk/VmbCPP/VmbAPI_MFC.zip