WPF应用程序调用C编译生成的DLL动态链接库

由于项目分为上位机（WPF C#）和嵌入式（C和Python）两大版本，同时项目成员的技术栈涉及到多种编程语言。如果每次一个项目成员用一种编程语言开发一个新模块，其他项目成员又需要用另一种编程语言进行移植，使得开发效率大大降低。

WPF可以通过引用DLL动态链接库文件来调用程序外部的功能。因此，如果实现在WPF上位机中直接调用嵌入式的功能模块，可以大大提升开发效率。

1、将C源码编译生成DLL动态链接库

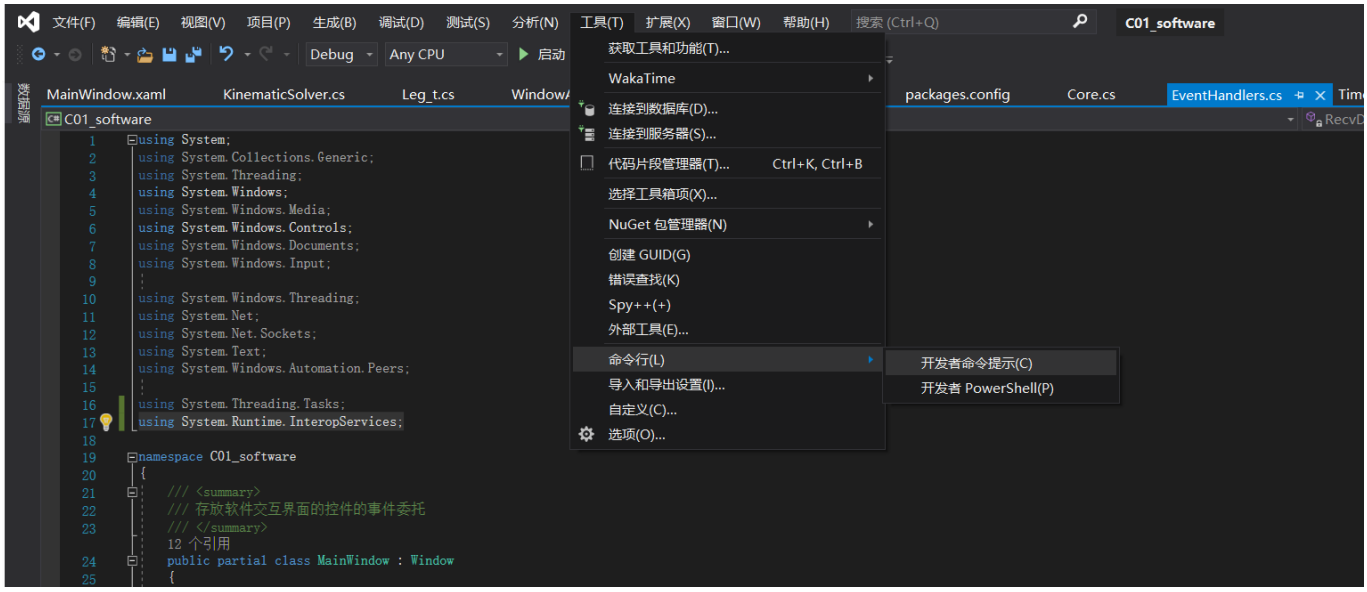
参考：https://blog.csdn.net/andrewniu/article/details/76443348

例如我想在上位机中调用一个如下C源文件（Test.c）中的函数功能，那么需要在被调用的函数前加“__declspec(dllexport)”来声明该函数是DLL的一个入口点。

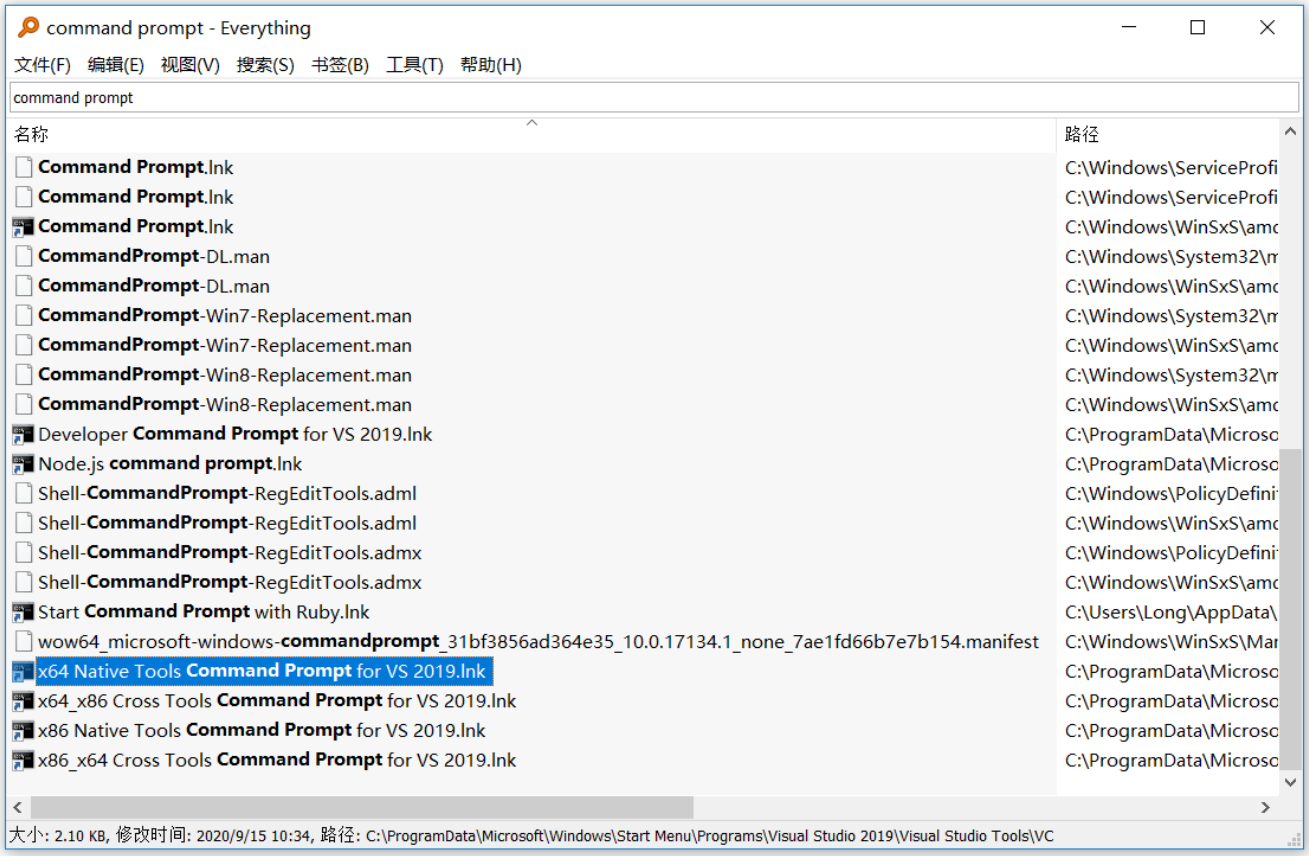
```
// Test.c
#include "stdio.h"

int tmp = 100;

__declspec(dllexport) int sum(int a, int b)
{
    return t1(a) + t2(b);
}

__declspec(dllexport) int sub(int a, int b)
{
    return t1(a) - t2(b);
}

void tmpHandler() {
    tmp = tmp + 100;
}

__declspec(dllexport) int test(int a, int b)
{
    tmpHandler();
    return tmp;
}

int t1(int a) {
    return a * 10;
}

int t2(int b) {
    return b * 100;
}
```

在该C源码中，设置了sum(), sub(), test()为入口点，可被C#调用。其中sum()和sub()会调用C源码中的t1()和t2()函数，这两个函数因为没有入口点声明，因此是不能直接被C#调用的，但是通过C#调用sum()和sub()可以间接地使用t1()和t2()的功能。另外C#调用test()，间接调用了tmpHandler()，使C源码中的tmp变量的值发生了改变。

C源码可以用Visual Studio的Prompt来编译生成DLL。有两种方法打开VS Prompt，一是在VS工具选项卡中直接打开，二是通过Everything搜索command prompt后选择VS的prompt。



打开VS Prompt的方法1：在VS工具中打开



打开VS Prompt的方法2：通过Everything搜索

在VS Prompt中输入以下命令来编程生成DLL：

```
cl /c Test.c

link /dll Test.obj
```

```
D:\Github\C01_Software\KinematicSolver\KinematicSolver>cl /c Test.c
用于 x86 的 Microsoft (R) C/C++ 优化编译器 19.27.29111 版
版权所有(C) Microsoft Corporation。保留所有权利。

Test.c

D:\Github\C01_Software\KinematicSolver\KinematicSolver>link /dll Test.obj
Microsoft (R) Incremental Linker Version 14.27.29111.0
Copyright (C) Microsoft Corporation.  All rights reserved.

  正在创建库 Test.lib 和对象 Test.exp
```

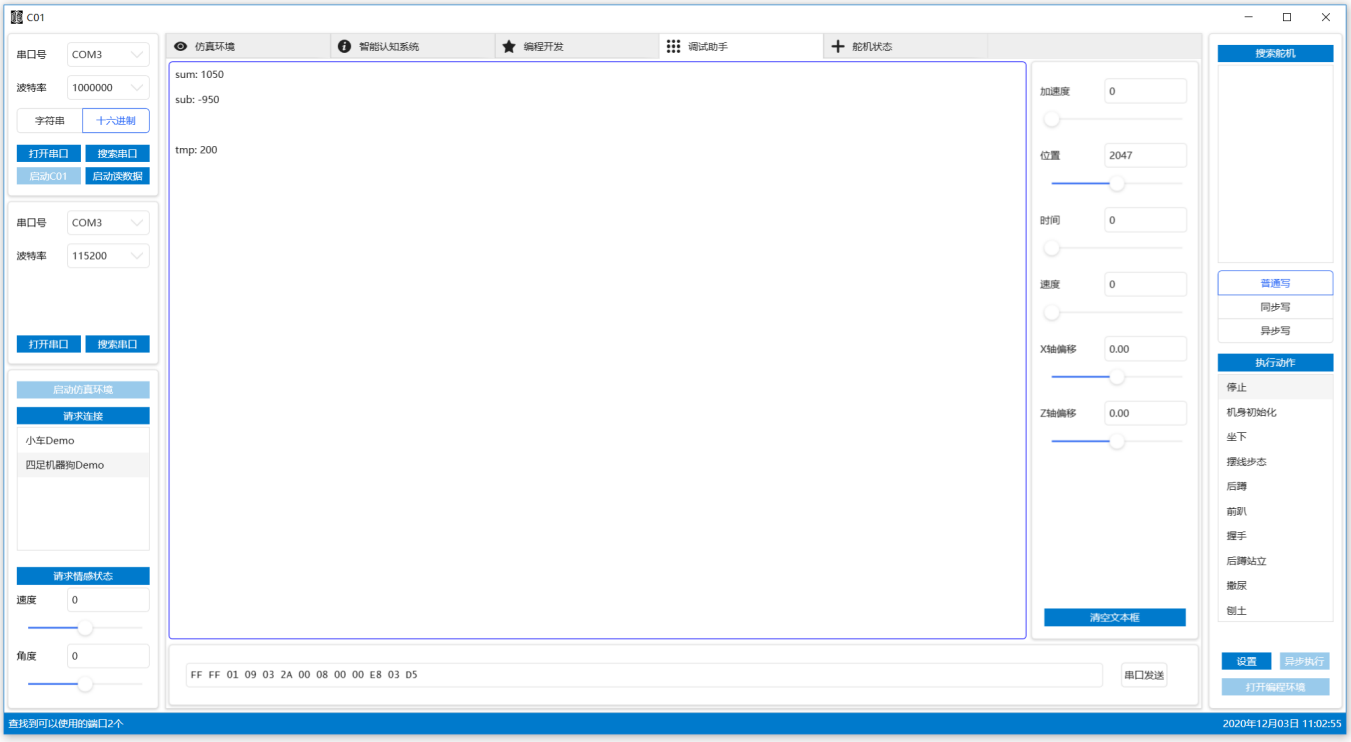
用VS Prompt编译生成DLL

2、在WPF中引用C代码生成的动态链接库DLL

由于由C编译生成的DLL是非托管DLL（非 .NET 环境下生成的DLL），所以无法直接通过在WPF项目结构树中添加引用。需要通过System.Runtime.InteropServices命名空间下的属性类DllImport来调用非托管DLL中的函数。

```
using System.Runtime.InteropServices;

[DllImport("Test.dll", EntryPoint = "sum", CallingConvention =
CallingConvention.Cdecl)]
public static extern int dllsum(int a, int b);
[DllImport("Test.dll", EntryPoint = "sub", CallingConvention =
CallingConvention.Cdecl)]
public static extern int dllsub(int a, int b);
[DllImport("Test.dll", EntryPoint = "test", CallingConvention =
CallingConvention.Cdecl)]
public static extern int dlltest();

private void clearRecvTextBlockButton_Click(object sender, RoutedEventArgs e)
{
    recvDataRichTextBox.AppendText("sum: " + dllsum(5, 10).ToString() + "\n");
    recvDataRichTextBox.AppendText("\n");
    recvDataRichTextBox.AppendText("sub: " + dllsub(5, 10).ToString() + "\n");
    recvDataRichTextBox.AppendText("\n");
    recvDataRichTextBox.AppendText("tmp: " + dlltest().ToString() + "\n");
}
```

上面的代码中，用DllImport分别对应入口点sum, sub, test，将它们的功能分别交给dllsum(), dllsub(), dlltest()。在按钮点击事件处理函数clearRecvTextBlockButton_Click()中依次引用了dllsum(), dllsub(), dlltest()，然后将它们返回的结果打印到文本框中。



WPF程序中点击按钮调用Test.dll中的函数并输出结果

如图所示，成功实现在WPF上位机中调用C的函数功能。