Matlab M file与PLC进行ADS通讯接口MEX文件的编写方法（2014.01.08）

Matlab 中的M文件需要直接和PLC进行ADS通讯时，需要编写一个MEX接口文件，通过此接口文件调用Beckhoff提供的TcADSDLL.dll文件，然后才能调用Beckhoff在cADSDLL.dll文件中的功能函数，实现与PLC进行ADS通讯。

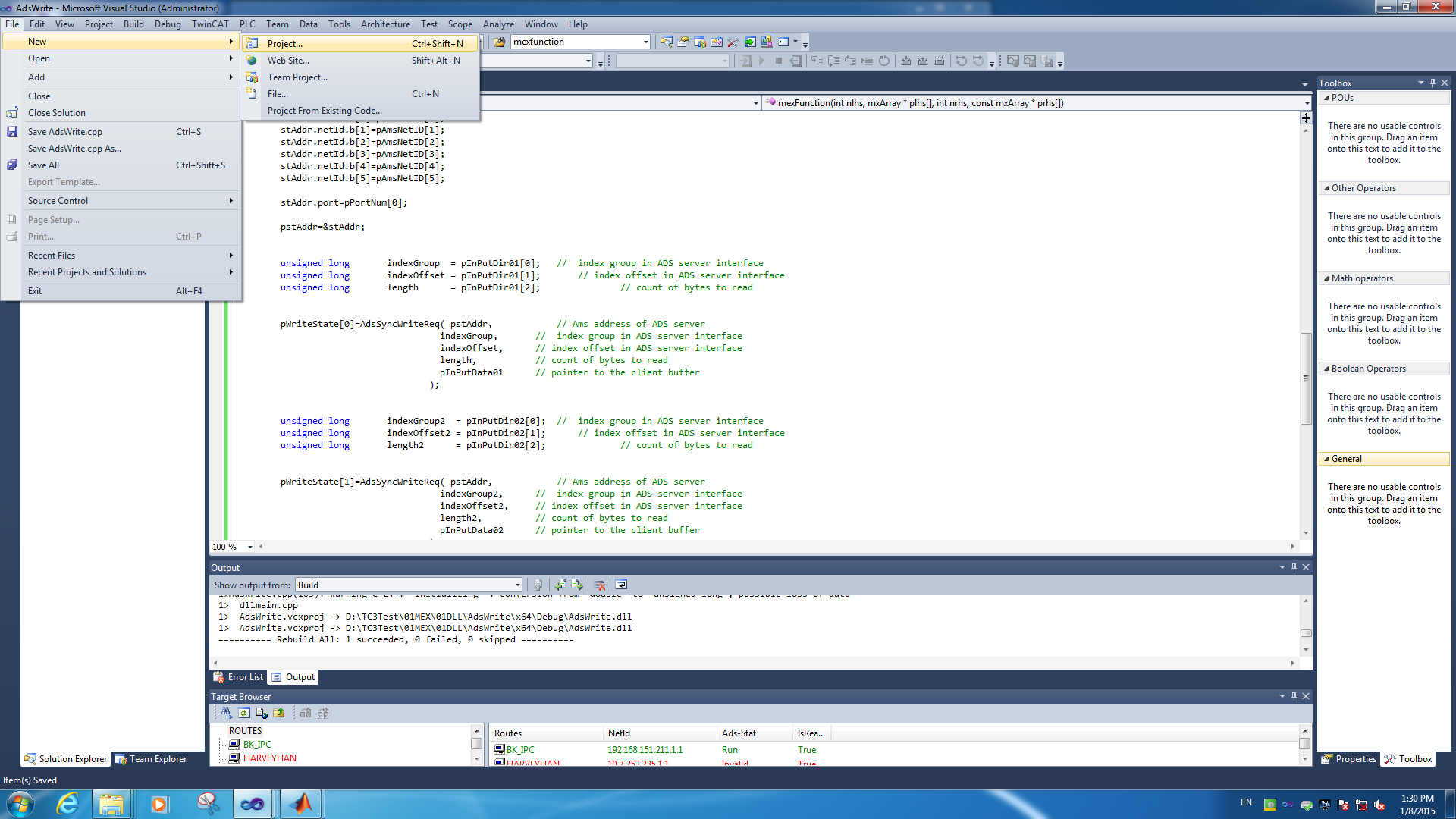
在Windows系统下，MEX文件是经过Matlab对VS下开发的源文件（C++开发的文件后缀为cpp）编译后，最终形成一个mexw32或mexw64文件，M file可以通过文件名直接调用此文件。

具体实现方法示例如下：

1. 实现平台

* VS2010 Ultimate：C++程序开发
* Matlab2012b(8.0.0.783) 64bits(win64)：M文件编写
* TwinCAT3 v3.1.4016.12：PLC程序编写
* Windows7 Ultimate 64bits：64位操作系统

1. MEX文件编写
2. 打开VS2010，新建一个Project

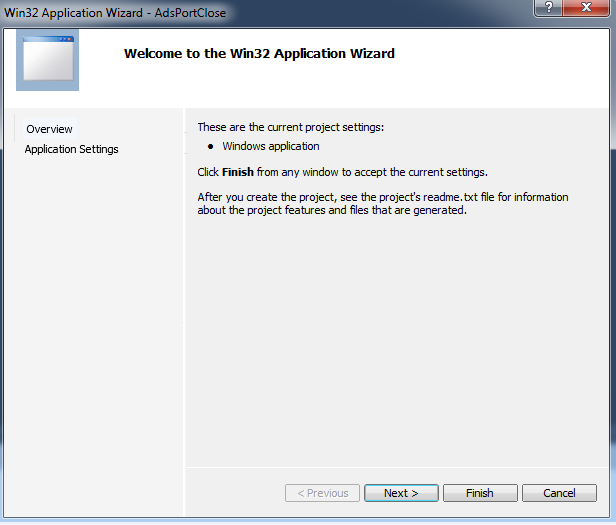


1. 向导窗口中，选择Win32 Project，并给项目起个名字，这里Solution name 可以随意，但项目的名称Name，将成为随后形成的cpp文件的名字，最终形成能被M file调用功能块的名字，所以最好与直接功能相关，如AdsPorClose,AdsRead等。

全部设置完成后，点击OK。



1. 出现信息窗口，点击Next



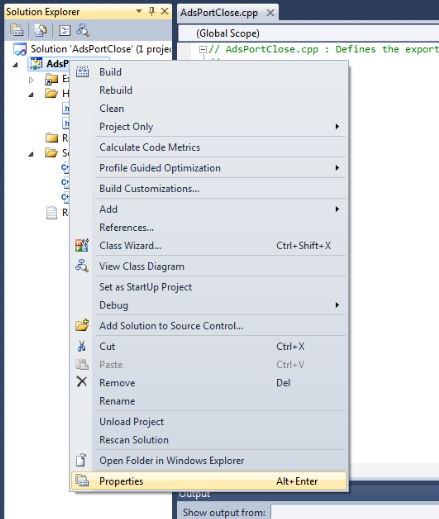
1. Application Type 选择 DLL，其他保持默认，点击Finish。



1. 项目建立完成，在左侧会出现VS2010为项目AdsPortClose建立的相关文件，其中AdsPortClose.cpp就是进行程序编写的文件。但是在编写程序之前，需要对环境进行配置。



1. 右击Solution的名字，出现下拉菜单中，选择Properties。



1. 弹出的属性设置窗口中，选择C/C++项目下的General，在右侧的Additional Include Directories 中需要添加随后编写的C++程序中所需引用的.H 文件的所在位置。

单击Additional Include Directories右侧的空白处，将出现下拉菜单，点击其中的Edit…选型，将出现设置窗口。



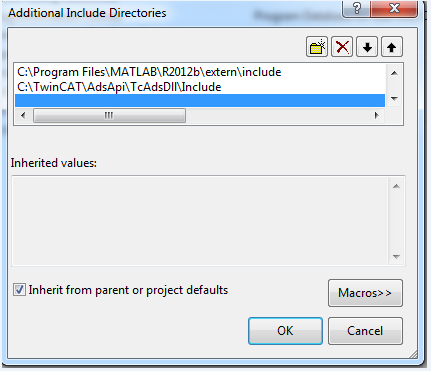
1. 在弹出的设置窗口中，点击右上角的“新建”快捷图标，添加下列两个路径

Matlab安装目录下的路径“extern\include”

TwinCAT安装目录下的路径“TwinCAT\AdsApi\TcAdsDll\Include”

添加完毕后，如下图所示。

然后，点击OK。



1. 回到Solution的属性设置窗口中，选择Linker项目下的General，在右侧的Additional Library Directories 中需要添加随后编写的C++程序中所需的.lib 文件的所在位置。

单击Additional Library Directories右侧的空白处，将出现下拉菜单，点击其中的Edit…选型，将出现设置窗口。



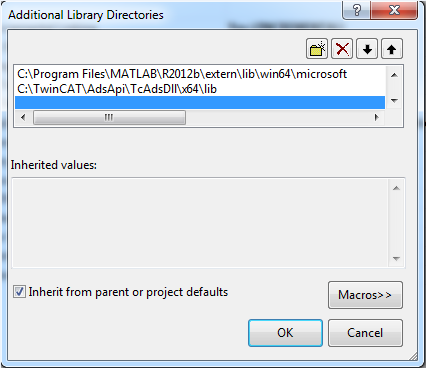
1. 在弹出的设置窗口中，点击右上角的“新建”快捷图标，添加下列两个路径

Matlab安装目录下的路径“extern\lib\win64\microsoft”

TwinCAT安装目录下的路径“\TwinCAT\AdsApi\TcAdsDll\x64\lib”

添加完毕后，如下图所示。

然后，点击OK。



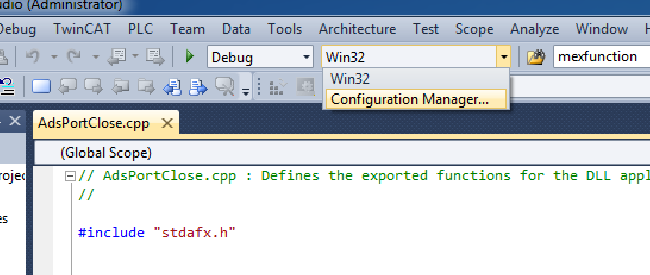
1. 回到Solution的属性设置窗口中，选择Linker项目下的Input，在右侧的Additional Dependencies 中需要添加下述几个lib文件：

libmx.lib、libmex.lib、libmat.lib、libeng.lib、TcAdsDll.lib

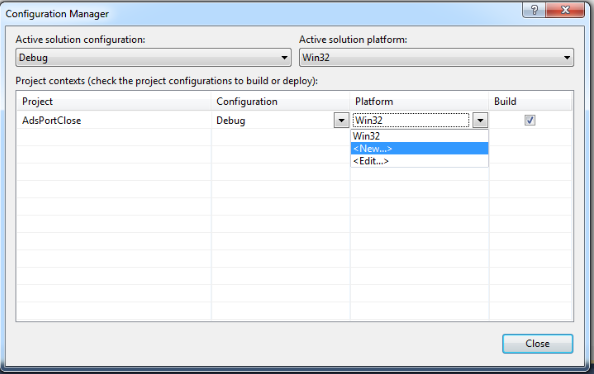
添加完毕后，点击OK。



1. 完成上述设置后，对于64位系统，还需设置编译器，选择VS2010工具栏中编译器选择位置如下图，在下拉菜单中选择Configuration Manager…

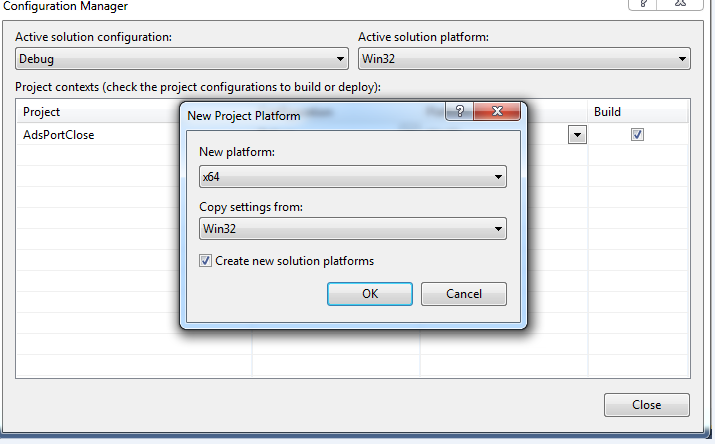


1. 在弹出的设置菜单中，platform的位置，点击下拉菜单，选择New…

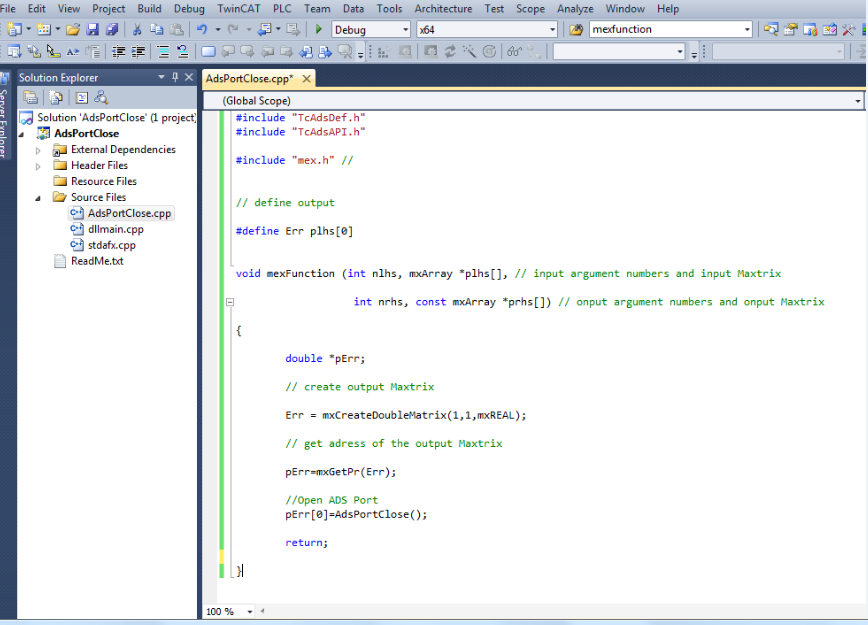


1. 弹出的设置菜单中，New Platform位置选择x64，如下图，然后点击OK。

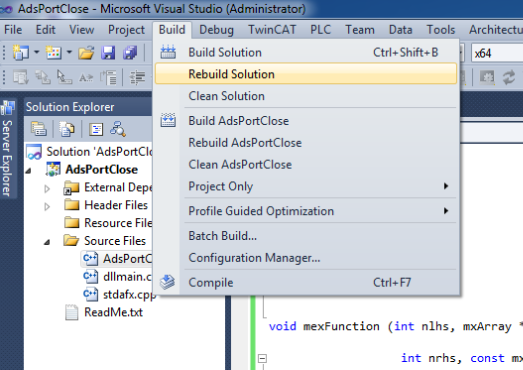
至此，环境设置完毕。



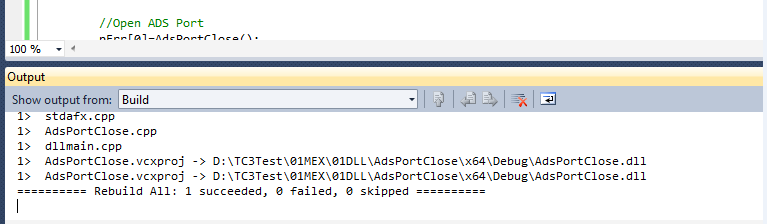
1. 点击AdsPortClose.cpp，在右侧的空白位置编写所需程序，详见例程AdsPortOpen.sln、AdsPortClose.sln、AdsRead.sln、AdsWrite.sln及下文程序编写注意事项。



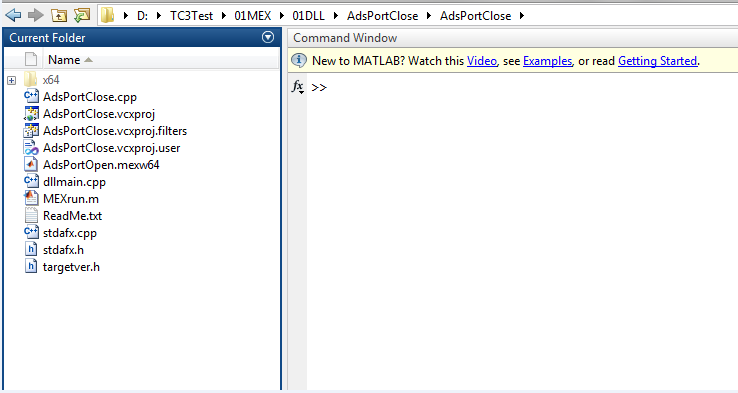
1. 程序编写完毕后，点击菜单栏的Rebuild Solution，对程序进行编译



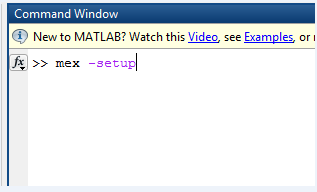
1. 一切正常的话，将在output窗口出现下述信息，表示编译通过。



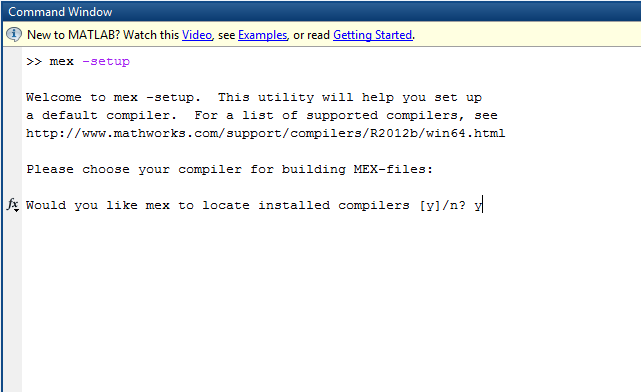
1. 打开Matlab，并将当前目录设置为文件AdsPortClose.cpp所在目录，如下图



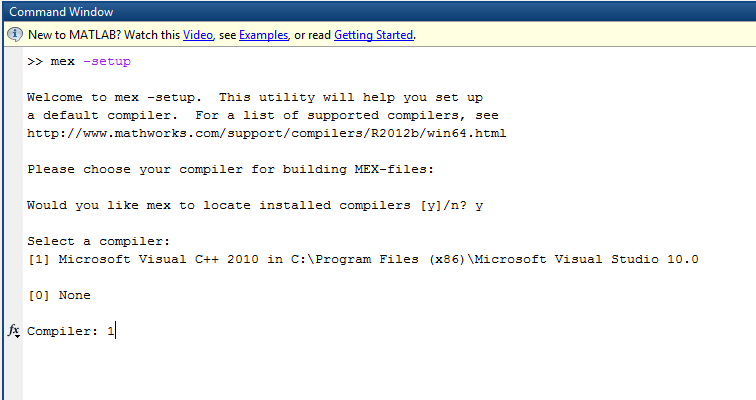
1. 命令行输入 mex –setup



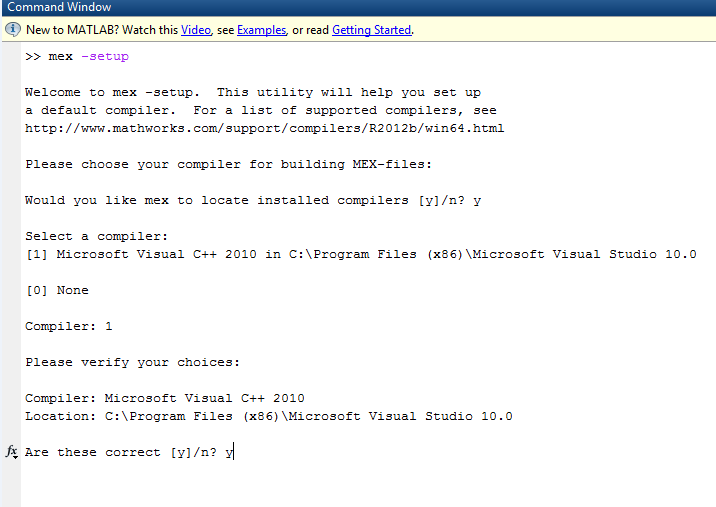
1. 提示信息后，输入y



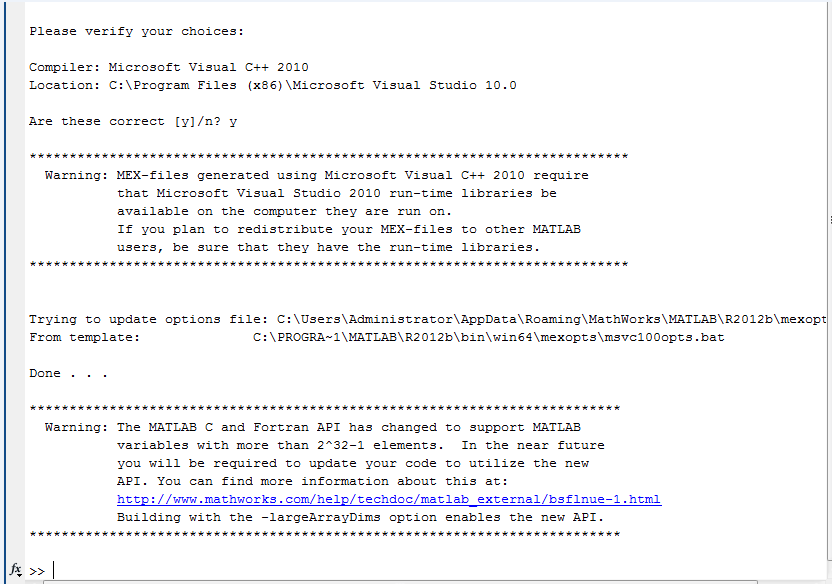
1. 选择，Microsoft Visual C++ 2010 的选项，本例子中选择，1



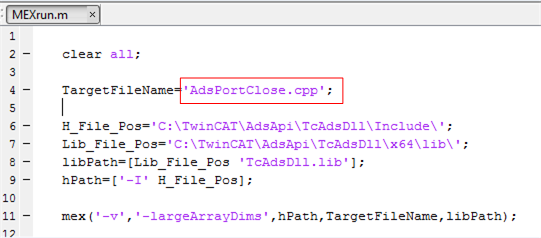
1. 弹出信息后，选择y



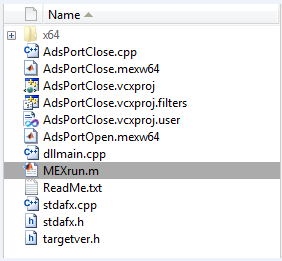
1. 弹出下述信息和提示框，表示设置完成。



1. 将MEXrun.m 文件拷贝到AdsPortClose.cpp所在的目录，然后将其中的TargetFileName更改为目标文件名称，此处为AdsPortClose.cpp。



1. 运行MEXrun.m文件，MATLAB开始对AdsPortClose.cpp进行编译，编译完成后，会在相同文件夹下出现AdsPortClose.mexw64文件，表示编译完成。



1. 至此，mexw64接口文件的制作完成，将AdsPortClose.mexw64拷贝到需要对其进行调用的M文件所在的文件夹，在M文件中直接输入文件名，即可进行调用。

在本例中，可以直接在输入[pErr]=AdsPortClose()；将完成对ADS通讯口的关闭。如有错误，返回值将写入pErr矩阵。

**MEX程序编写说明及注意事项：**

MEX是一种MATLAB调用DLL文件的接口文件，成功的MEX文件编写，需要注意一些问题，下面以AdsRead.mexw64文件为例，说明程序各部分的注意事项。

1. MATLAB在对MEX文件进行调用时，实际上是通过其中的mexFunction函数进行的，其4个参量的意义分别为：

nlhs：输入参量的个数

\*plhs[]：指向输入参量地址的指针矩阵

nrhs：输出参量个数

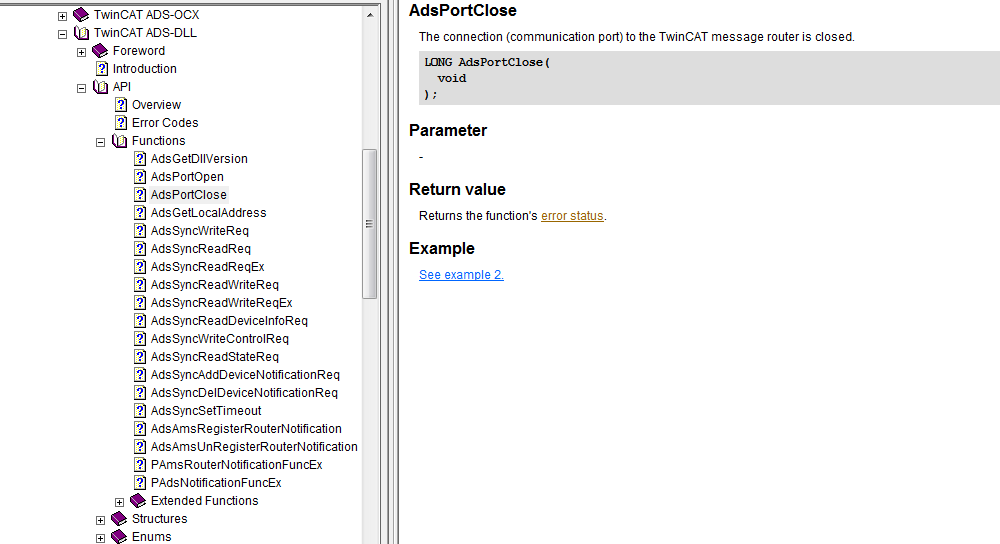
\*prhs[]：指向输出参量地址的指针矩阵

其中，将需要输入的矩阵或变量都依次定义为 prhs[0]、prhs[1]、prhs[2]…..，将需要输出的矩阵或变量都依次定义为plhs[0]、plhs [1]、plhs [2]…..

当Matlab进行调用时，将按顺序将输入矩阵依次传输给prhs[0]、prhs[1]、prhs[2]…..

运算完成后，又依次将plhs[0]、plhs [1]、plhs [2]…..中的变量传输给Matlab。

1. mexFunction函数内部可以直接编写需要的运算程序，对于ADS功能块的调用，可以直接写功能块的名称如 AdsSyncReadReq。可以调用的ADS功能块可以参考，Beckhoff帮助文件，TwinCAT ADS-DLL 位置查找详细说明。



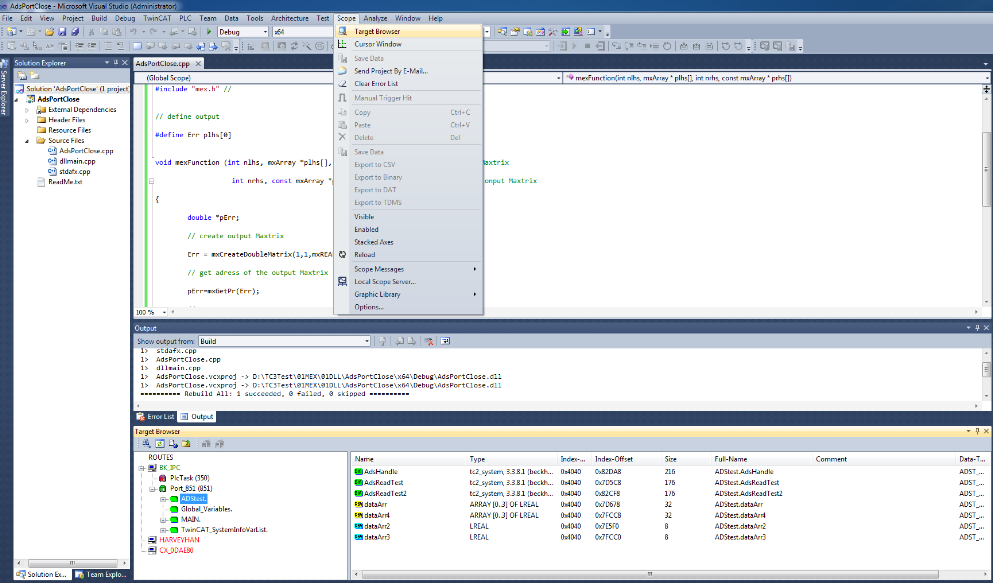
1. 由于Matlab的默认数据类型试double型，所以Matlab与PLC进行数据交换时，尽量以double型进行交换，在MEX文件编写中，涉及到接口的位置都使用double型，在PLC内需要与Matlab进行交换的数据都定义为LREAL型。
2. 由于32位操作系统和64位操作系统中，相同数据类型长度是不同的，所以两种系统上运行的文件是不一样的，后缀分别为mexw32和mexw64，两者不能通用，可以在相应系统上，用matlab对源文件进行重新编译，即可形成所需的mex文件。
3. 在M文件中进行调用时，可以直接输入mex文件的名称，引用格式为

[output1,output2]=AdsRead(input1,input2);

其中，output1，output2 根据其引用时的输入位置，依次对应mex文件中的plhs[0]、plhs [1]；

input1,input2根据其引用时的输入位置，依次对应mex文件中的prhs[0]、prhs [1]；

1. 本例程以变量地址的方式，进行ADS寻址，对于需要访问变量的地址可以通过TwinCAT界面的Scope 功能进行查找。当PLC程序运行时，在下图位置可查看相应变量的地址和长度。





AdsRead.cpp 源程序

// AdsRead.cpp : Defines the exported functions for the DLL application.

//

#include "stdafx.h"

#include "TcAdsDef.h"

#include "TcAdsAPI.h"

#include "mex.h" //

// define input

#define AmsNetID prhs[0]

#define PortNum prhs[1]

#define InPut01 prhs[2]

#define InPut02 prhs[3]

// define output

#define ReadState plhs[0]

#define OutPut01 plhs[1]

#define OutPut02 plhs[2]

void mexFunction (int nlhs, mxArray \*plhs[], // input argument numbers and input Maxtrix

int nrhs, const mxArray \*prhs[]) // output argument numbers and onput Maxtrix

{

int r1,c1,r2,c2;

double \*pAmsNetID,\*pPortNum,\*pInPut01,\*pInPut02;

double \*pReadState,\*pOutPut01,\*pOutPut02;

AmsAddr stAddr;

AmsAddr \*pstAddr;

// get address of the input Maxtrix

pAmsNetID=mxGetPr(AmsNetID);

pPortNum=mxGetPr(PortNum);

pInPut01=mxGetPr(InPut01);

pInPut02=mxGetPr(InPut02);

// get row and culomn numbers of the Output Maxtrix deserved

r1 = pInPut01[3];

c1 = pInPut01[4];

r2 = pInPut02[3];

c2 = pInPut02[4];

// create output Maxtrix

ReadState = mxCreateDoubleMatrix(1,2,mxREAL);

OutPut01 = mxCreateDoubleMatrix(r1,c1,mxREAL);

OutPut02 = mxCreateDoubleMatrix(r2,c2,mxREAL);

// get adress of the output Maxtrix

pReadState=mxGetPr(ReadState);

pOutPut01=mxGetPr(OutPut01);

pOutPut02=mxGetPr(OutPut02);

//Read ADS Data

stAddr.netId.b[0]=pAmsNetID[0];

stAddr.netId.b[1]=pAmsNetID[1];

stAddr.netId.b[2]=pAmsNetID[2];

stAddr.netId.b[3]=pAmsNetID[3];

stAddr.netId.b[4]=pAmsNetID[4];

stAddr.netId.b[5]=pAmsNetID[5];

stAddr.port=pPortNum[0];

pstAddr=&stAddr;

unsigned long indexGroup = pInPut01[0]; // index group in ADS server interface

unsigned long indexOffset = pInPut01[1]; // index offset in ADS server interface

unsigned long length = pInPut01[2]; // count of bytes to read

pReadState[0]=AdsSyncReadReq( pstAddr, // Ams address of ADS server

indexGroup, // index group in ADS server interface

indexOffset, // index offset in ADS server interface

length, // count of bytes to read

pOutPut01 // pointer to the client buffer

);

unsigned long indexGroup2 = pInPut02[0]; // index group in ADS server interface

unsigned long indexOffset2 = pInPut02[1]; // index offset in ADS server interface

unsigned long length2 = pInPut02[2]; // count of bytes to read

pReadState[1]=AdsSyncReadReq( pstAddr, // Ams address of ADS server

indexGroup2, // index group in ADS server interface

indexOffset2, // index offset in ADS server interface

length2, // count of bytes to read

pOutPut02 // pointer to the client buffer

);

return;

}