**使用Microsoft Visual C++实现与TwinCAT 3 PLC变量的通讯**

|  |  |
| --- | --- |
|  | 修订人：吴昊天  职务：华北区 技术工程师  邮箱：ht.wu@beckhoff.com.cn  修订日期：2023-02-06 |
| **摘 要：**  使用Microsoft Visual C++ 软件，实现通过对TwinCAT 3 PLC中的变量（bool int float string array struct类型）进行读取和写入。  本次测试所用的TwinCAT3版本为4024.35，用到的Visual Studio版本为2017 Community。 | |
| **附 件：**   |  |  |  | | --- | --- | --- | | 序 号 | 文件名 | 备注 | |  | TcAdsDll.lib | 为了使用TcAdsDll.dll所要添加的库文件，需要在vs配置环境中去附加库目录和依赖项 | |  | TcAdsDef.h | 结构体和常量的声明 | |  | TcAdsApi.h | ADS 函数的声明 | |  |  | 上面两个是在头文件中要引用,安装了TwinCAT3自动就有,在c盘TwinCAT->AdsApi->TcAdsDll文件夹下 | |  |  |  | |  |  |  | | |
| **历史版本：**   |  |  |  | | --- | --- | --- | | 2018-07-10 | 王子鹏 | TC3的ADS通讯（C++） | | 2022-03-30 | 吴昊天 | TC3的ADS通讯（C++）V2.0 | |  |  |  | |  |  |  | |  |  |  | | |
| **免责声明：**  我们已对本文档描述的内容做测试。但是差错在所难免，无法保证绝对正确并完全满足您的使用需求。本文档的内容可能随时更新，如有改动，恕不事先通知，也欢迎您提出改进建议。 | |
| **参考信息：** | |

**目 录**

[1 软硬件版本 4](#_Toc99559428)

[1.1 倍福Beckhoff 4](#_Toc99559429)

[1.1.1 控制软件 4](#_Toc99559430)

[2 建立C++程序与目标控制器PLC程序间的通讯 4](#_Toc99559431)

[2.1 （建立通讯）编写PLC程序 4](#_Toc99559432)

[2.2 （建立通讯）编写C++程序 6](#_Toc99559433)

[3 通过C++程序读取/写入PLC程序中的布尔量 12](#_Toc99559434)

[3.1 （布尔量读取）编写PLC程序 12](#_Toc99559435)

[3.2 （布尔量读取）编写C++程序 13](#_Toc99559436)

[3.3 （布尔量写入）编写PLC程序 16](#_Toc99559437)

[3.4 （布尔量写入）编写C++程序 16](#_Toc99559438)

[4 通过C++程序读取/写入PLC程序中的整型量 17](#_Toc99559439)

[4.1 （整型量读取）编写PLC程序 17](#_Toc99559440)

[4.2 （整型量读取）编写C++程序 18](#_Toc99559441)

[4.3 （整型量写入）编写PLC程序 19](#_Toc99559442)

[4.4 （整型量写入）编写C++程序 19](#_Toc99559443)

[5 通过C++程序读取/写入PLC程序中的浮点数 21](#_Toc99559444)

[5.1 （浮点数读取）编写PLC程序 21](#_Toc99559445)

[5.2 （浮点数读取）编写C++程序 21](#_Toc99559446)

[5.3 （浮点数写入）编写PLC程序 23](#_Toc99559447)

[5.4 （浮点数写入）编写C++程序 23](#_Toc99559448)

[6 通过C++程序读取/写入PLC程序中的字符串 24](#_Toc99559449)

[6.1 （字符串读取）编写PLC程序 24](#_Toc99559450)

[6.2 （字符串读取）编写C++程序 25](#_Toc99559451)

[6.3 （字符串写入）编写PLC程序 26](#_Toc99559452)

[6.4 （字符串写入）编写C++程序 27](#_Toc99559453)

[7 通过C++程序读取/写入PLC程序中的数组 28](#_Toc99559454)

[7.1 （数组读取）编写PLC程序 28](#_Toc99559455)

[7.2 （数组读取）编写C++程序 29](#_Toc99559456)

[7.3 （数组写入）编写PLC程序 31](#_Toc99559457)

[7.4 （数组写入）编写C++程序 31](#_Toc99559458)

[8 通过C++程序读取/写入PLC程序中的结构体 32](#_Toc99559459)

[8.1 （结构体读取）编写PLC程序 32](#_Toc99559460)

[8.2 （结构体读取）编写C++程序 34](#_Toc99559461)

[8.3 （结构体写入）编写PLC程序 36](#_Toc99559462)

[8.4 （结构体写入）编写C++程序 36](#_Toc99559463)

[9 通过C++程序读取/写入PLC程序中结构体类型的数组 37](#_Toc99559464)

[9.1 （结构体类型的数组读取）编写PLC程序 37](#_Toc99559465)

[9.2 （结构体类型的数组读取）编写C++程序 38](#_Toc99559466)

[9.3 （结构体类型的数组写入）编写PLC程序 40](#_Toc99559467)

[9.4 （结构体类型的数组写入）编写C++程序 40](#_Toc99559468)

[10 通过定时方式来读取PLC程序中的变量 42](#_Toc99559469)

[10.1 编写PLC程序 42](#_Toc99559470)

[10.2 编写C++程序 43](#_Toc99559471)

[11 通过通知方式来读取PLC程序中的变量 46](#_Toc99559472)

[11.1 编写PLC程序 46](#_Toc99559473)

[11.2 编写C++程序 47](#_Toc99559474)

[12 通过C++程序在PLC变量被隐藏时读取所有变量信息 49](#_Toc99559475)

[12.1 编写PLC程序 50](#_Toc99559476)

[12.2 编写C++程序 50](#_Toc99559477)

# 软硬件版本

## 倍福Beckhoff

* + 1. 控制软件

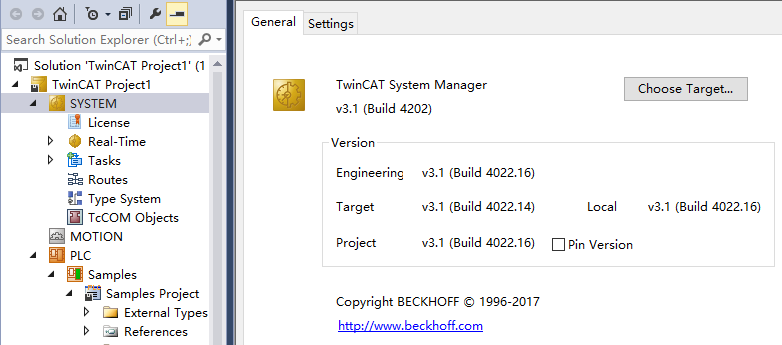
笔记本是基于TwinCAT 3.1 Build 4022.16版本

控制器是基于TwinCAT 3.1 Build 4022.14版本

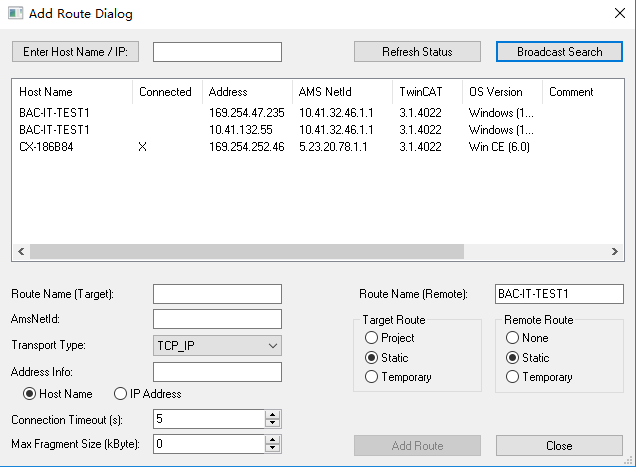
# 建立C++程序与目标控制器PLC程序间的通讯

## （建立通讯）编写PLC程序

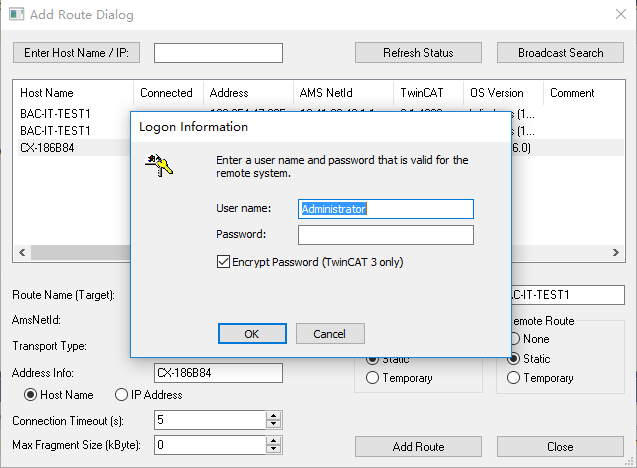
1. 打开TwinCAT3，新建一个项目。首先需要连接目标控制器（如果是本机模拟程序，则跳过添加路由这步）。
2. 双击左菜单中的SYSTEM，选择Choose Target System，如图所示：



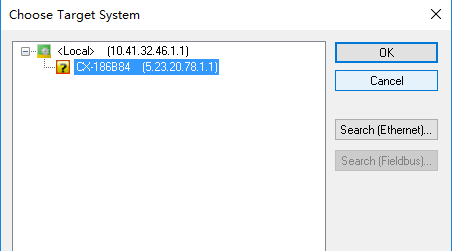
1. 在Choose Target System窗口，选择Search(Ethernet)。在Add Route Dialog窗口，选择Broadcast Search，在弹出窗口选择OK。效果如下图：



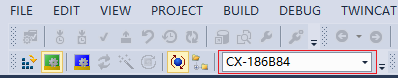
1. 选择的目标设备CX-186B84并在Address Info处选择IP Address，点击Add Route。效果如下：



1. CE设备密码为空白，直接选择OK，返回前一窗体。此前添加的目标控制器出现在这个列表中，选中，点击OK。



1. 当左上方图标栏中出现设备名称时，表示连接目标控制器成功，如图所示：

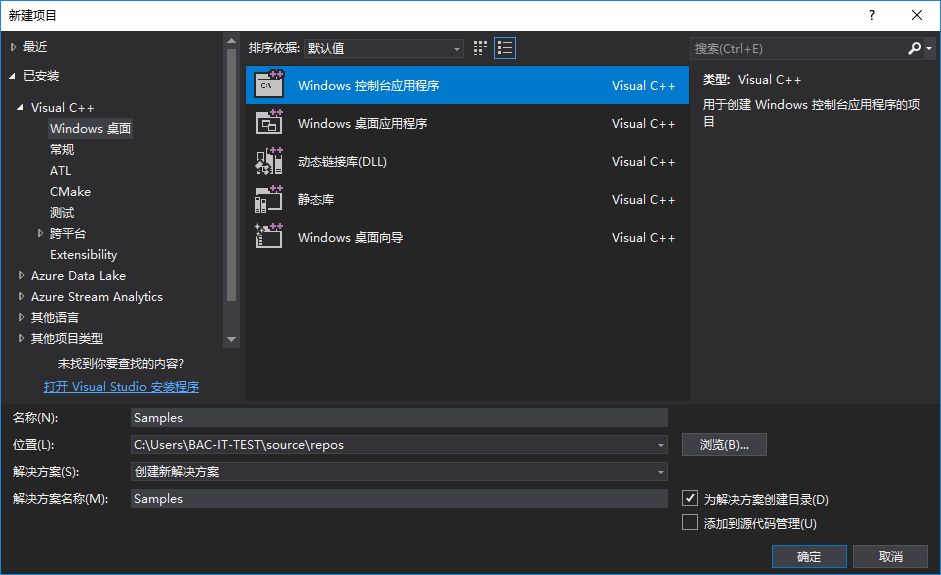


1. 创建并编写一个PLC程序，完成后点击左上角的 Activate Configuration图标，并LOGIN建立851端口连接。完成后效果如图所示：

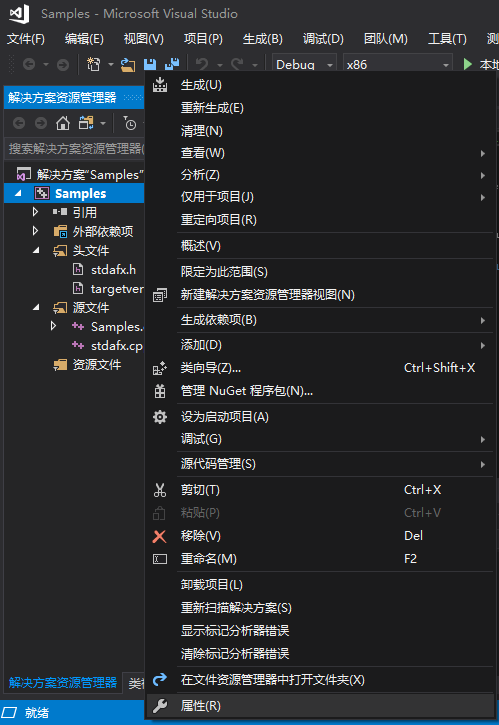


## （建立通讯）编写C++程序

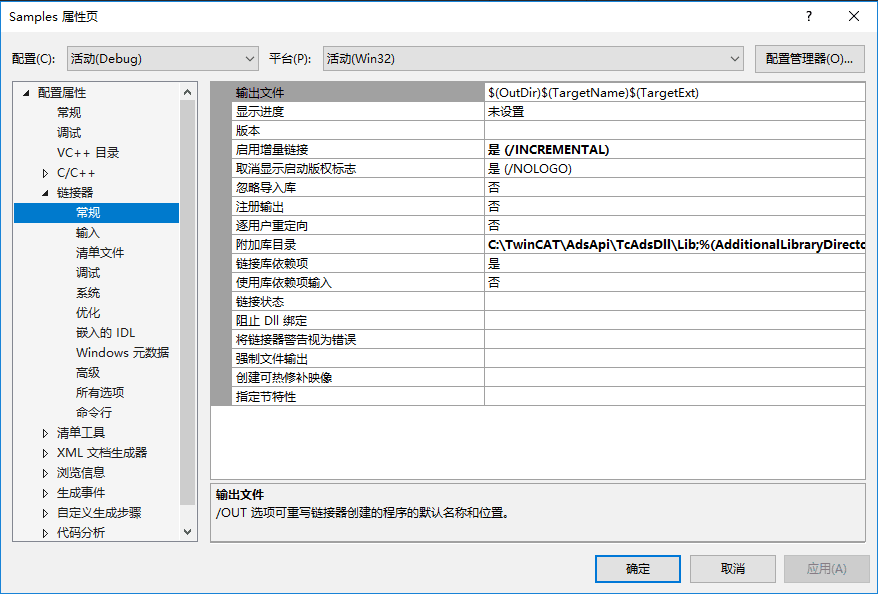
1. 首先启动Visual Studio 2017，新建一个项目。点击窗口左上方菜单栏的文件->新建->项目。出现窗口如下图所示。选择Visual C++中的Windows控制台应用程序选项。输入项目名称（这里起名为：Samples）并选择为解决方案创建目录。点击确定。



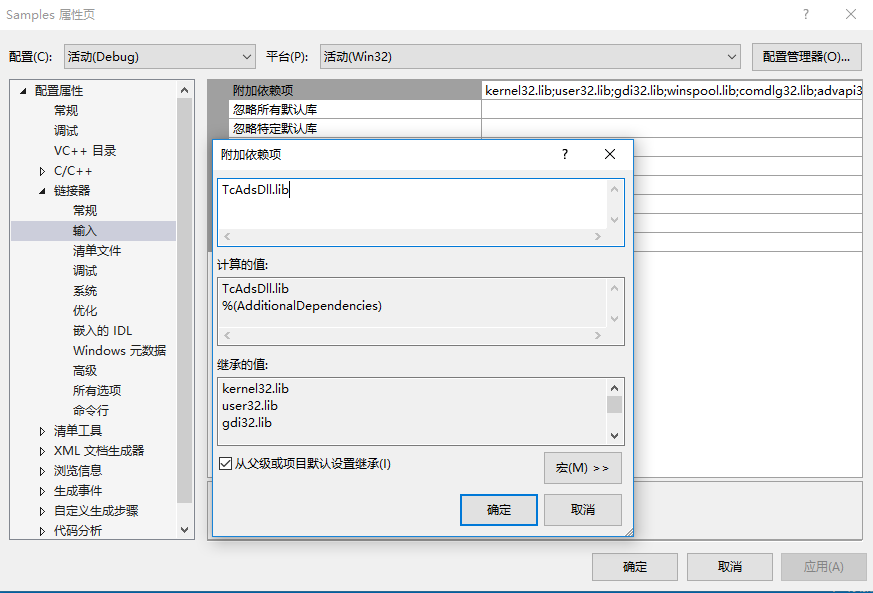
1. 此时需要添加链接库，以便实现通讯中用到的各种函数的直接调用。在左侧找到所建立的项目处，点击右键->属性。效果如下图。



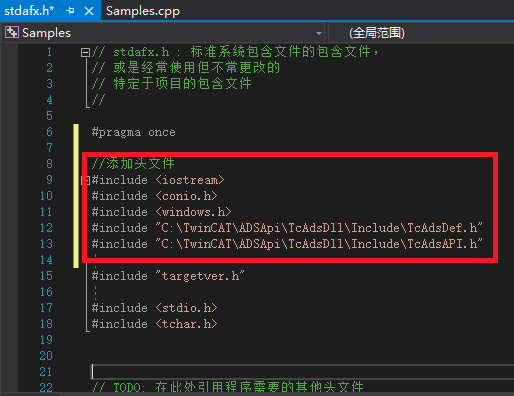
1. 选择配置属性->链接器->常规->附加库目录->编辑，进入附加包含目录的编辑界面。如下图所示，添加的程序的所在目录（C:\TwinCAT\AdsApi\TcAdsDll\Lib），点击确定。（本路径是以安装TwinCAT3时的默认路径配置的，若用户在安装时选择自定义路径，则请用户自行修改）



1. 再选择链接器－>输入－>附加依赖项，点击右边的按钮选择编辑进入附加依赖项界面。如下图所示添加的.lib文件的文件名（TcAdsDll.lib），点击确定。再次点击确定，链接库添加完成。



1. 此时还需要将头文件加入，点击解决方案下的Samples－> 头文件－> stdafx.h，双击打开。向其中添加头件，如下图所示（tcAdsDef.h 和TcAdsApi.h ）。因为头文件没有复制到程序目录，所以在前面加上了头文件所在地址。接着再次向程序添加头文件（iostream、windows.h、conio.h）。效果如图所示：



代码：

//添加头文件

#include <iostream> //标准的输入输出流头文件

#include <conio.h> //控制台输入输出头文件

#include <windows.h> //windowsAPI的函数声明和宏

#include "C:\TwinCAT\ADSApi\TcAdsDll\Include\TcAdsDef.h" //结构体和常量的声明

#include "C:\TwinCAT\ADSApi\TcAdsDll\Include\TcAdsAPI.h" // ADS函数的声明

1. 编写主程序

代码：

//添加头文件

#include "stdafx.h"

//指定名字空间

using namespace std;

void main()

{

long nErr, nPort; //定义端口变量

AmsAddr Addr; //定义AMS地址变量

PAmsAddr pAddr = &Addr;//定义端口地址变量

pAddr->port = 851; //TC3通讯使用的为851端口

nPort = AdsPortOpen(); //打开ADS通讯端口

pAddr->netId.b[0] = 5; //手动填写目标设备的AMSNETID

pAddr->netId.b[1] = 23; //在例子中设备通讯AMSNETID为5.23.20.78.1.1

pAddr->netId.b[2] = 20;

pAddr->netId.b[3] = 78;

pAddr->netId.b[4] = 1;

pAddr->netId.b[5] = 1;

1. 编译程序并运行，这样就建立了C++程序与目标控制器中的PLC程序的通讯。当目标控制器为本地设备时，便无需手动填写目标设备的AMSNETID，可以通过直接调用AdsGetLocalAddress（）自动填写本地设备的AMSNETID。

代码：

//添加头文件

#include "stdafx.h"

//指定名字空间

using namespace std;

void main()

{

long nErr, nPort; //定义端口变量

AmsAddr Addr; //定义AMS地址变量

PAmsAddr pAddr = &Addr; //定义端口地址变量

//建立接口通讯连接

nPort = AdsPortOpen(); //打开ADS通讯端口

nErr = AdsGetLocalAddress(pAddr); //自动获取本地地址

if (nErr) cerr << "Error: AdsGetLocalAddress: " << nErr << '\n';

pAddr->port = 851; //TC3的通讯端口为851

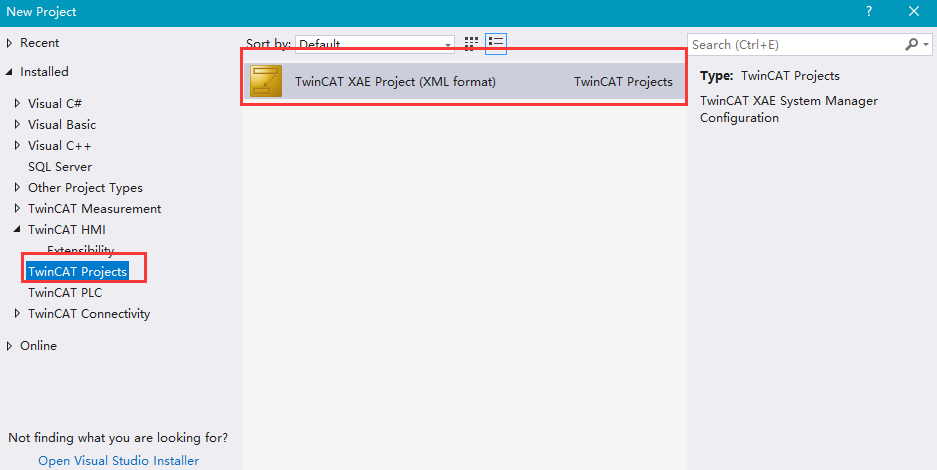
}

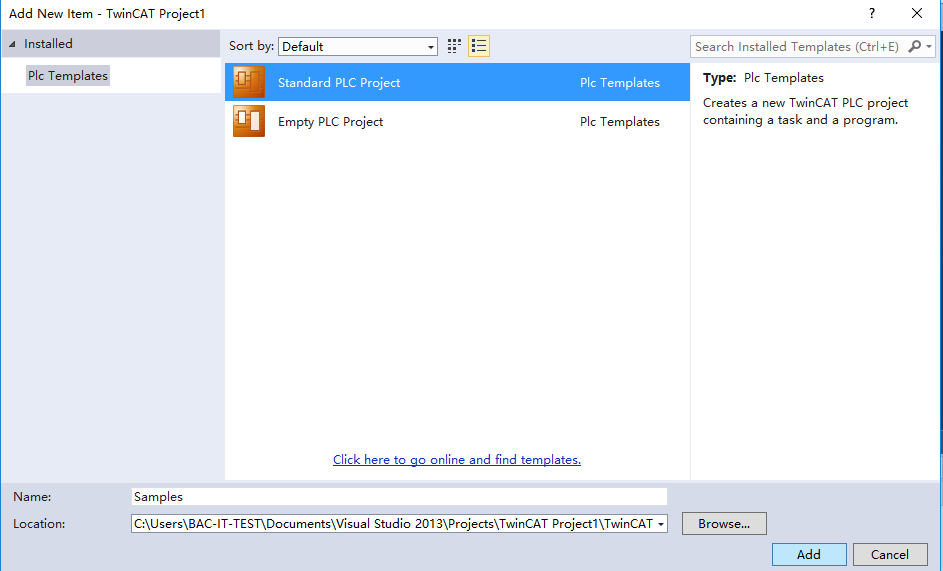
1. 编译程序并运行，这样就建立了C++程序与本地TC3中的PLC程序的通讯。

# 通过C++程序读取/写入PLC程序中的布尔量

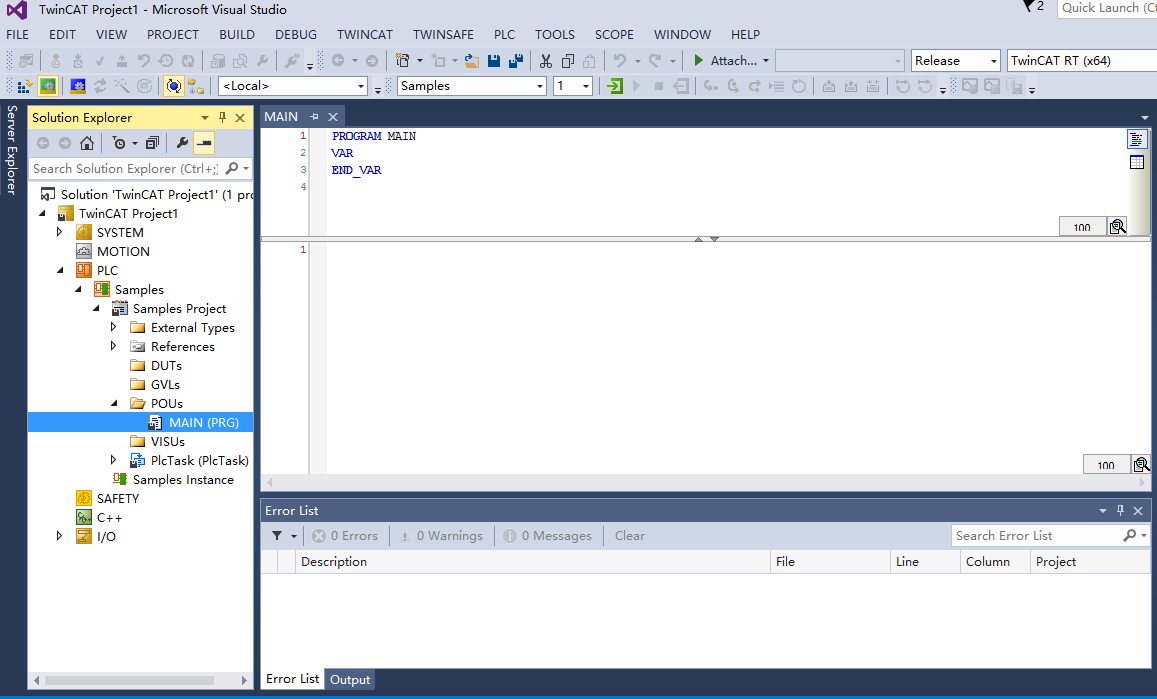
## （布尔量读取）编写PLC程序

1. 打开TwinCAT3，新建一个TwinCat Project，完成后鼠标右键点击左菜单栏中的PLC选项选择Add New Item，选择Standard PLC Project建立标准PLC项目。

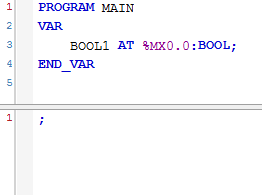




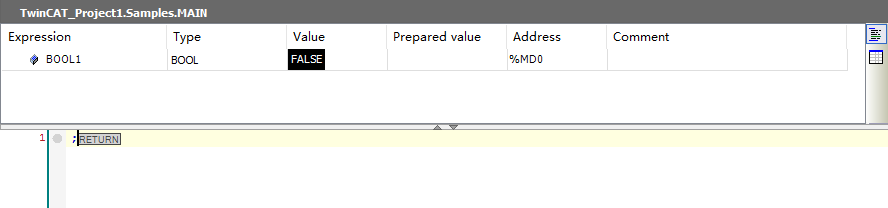
1. 展开左菜单栏中Project->POUs选项，双击选择编辑MAIN(PRG)，完成后效果如下：



1. 在变量声明区定义一个名称为BOOL1的BOOL型变量，并赋予其地址“AT%MX0.0”；

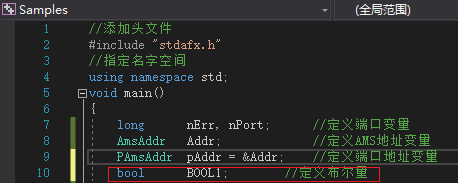


1. 编写完程序后，BUILD程序并激活，LOG IN建立851端口通讯，效果如下：



## （布尔量读取）编写C++程序

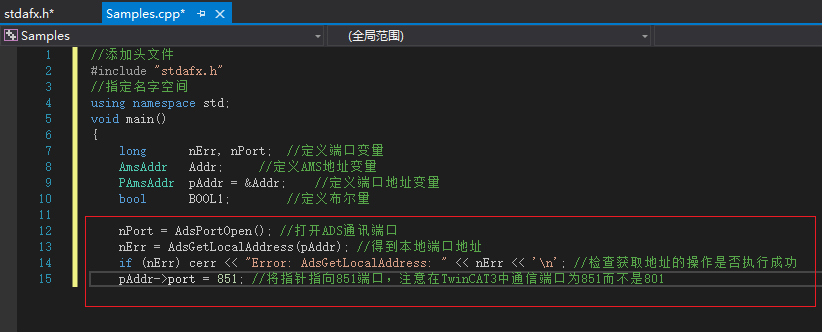
1. 首先打开Visual Studio 2017，创建一个控制台项目，并且为项目配置好所需链库。
2. 接下来开始编写主程序。首先需要定义变量，由于将要从PLC中读取bool型变量，所以程序中BOOL1的变量类型为bool。程序如下图所示：



代码：

bool BOOL1; //定义布尔量

1. 接着再编写“**打开ADS通讯端口**”的程序，如下图：



代码：

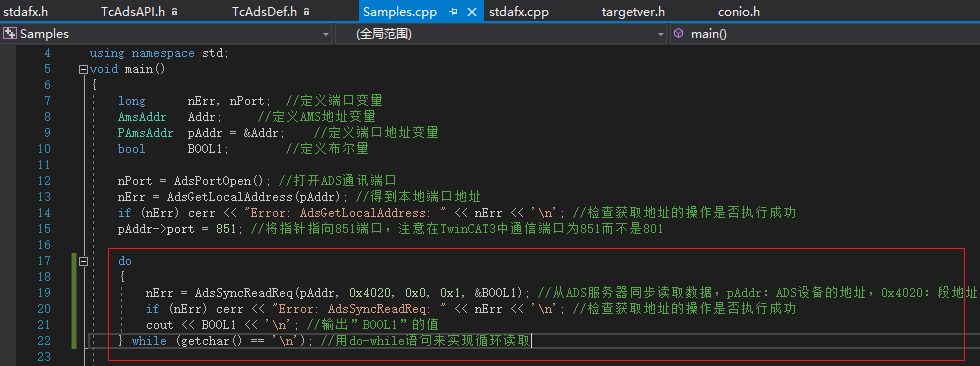
nPort = AdsPortOpen(); //打开ADS通讯端口

nErr = AdsGetLocalAddress(pAddr); //得到本地端口地址

if (nErr) cerr << "Error: AdsGetLocalAddress: " << nErr << '\n'; //检查获取地址的操作是否执行成功

pAddr->port = 851; //将指针指向851端口，注意在TwinCAT3中通信端口为851而不是801

1. 之后再编写“**从PLC中读取布尔量**”的程序，如下图所示。



代码：

do

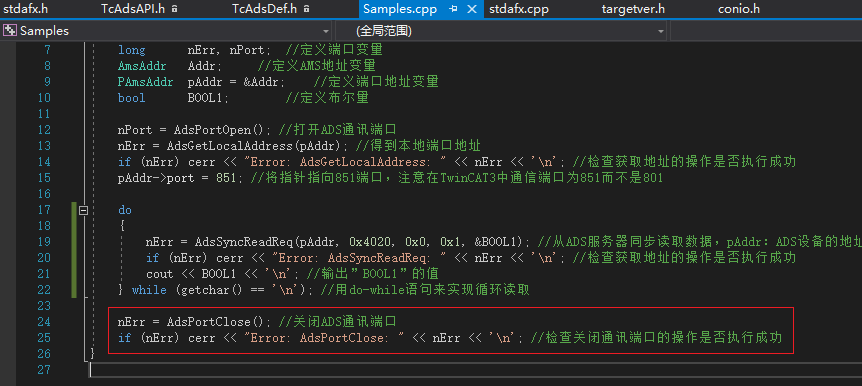
{ nErr = AdsSyncReadReq(pAddr, 0x4020, 0x0, 0x1, & BOOL1); //从ADS服务器同步读取数据，pAddr：ADS设备的地址，0x4020：段地址，0x0偏移地址，0x1：数据长度，&BOOL1:接收数据的缓存

if (nErr) cerr << "Error: AdsSyncReadReq: " << nErr << '\n'; //检查获取地址的操作是否执行成功

cout << BOOL1 << '\n'; //输出”BOOL1”的值

} while (getchar() == '\n'); //用do-while语句来实现循环读取

1. 最后再编写“**关闭通讯端口**”的程序，如图所示：



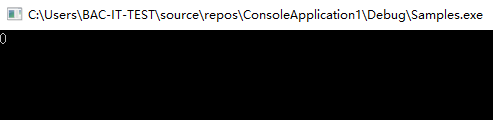
代码：

nErr = AdsPortClose(); //关闭ADS通讯端口

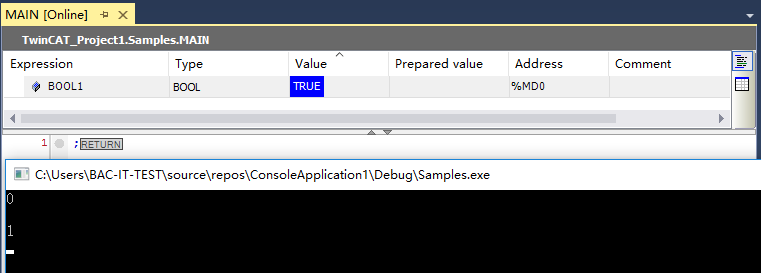
if (nErr) cerr << "Error: AdsPortClose: " << nErr << '\n'; //检查关闭通讯端口的操作是否执行成功

}

1. 程序编写完成后点击菜单 生成->生成解决方案，当编译没有错误后，点击菜单调试->开始调试，运行程序，出现如下图示：



1. 此时PLC中BOOL1的变量就会在的控制台上显示，为“0”。当将PLC中BOOL1的值改为“TRUE”时，在控制台上按下“回车”，值就变换为“1”了，如图所示：



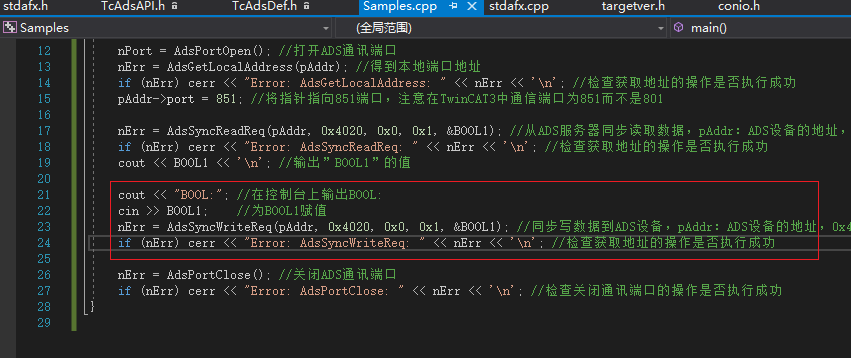
这样从PLC中读取BOOL量的值就完成了。

## （布尔量写入）编写PLC程序

1. 打开TwinCAT3
2. 编写程序并运行（与读取部分相同）

## （布尔量写入）编写C++程序

1. 打开Visual Studio 2017，创建一个控制台项目，并且为项目配置好所需链接库。
2. 编写程序，只需在已经编写好的“**布尔量读取**”程序中加入“**向PLC写入布尔量**”的程序即可，如图所示:



代码：

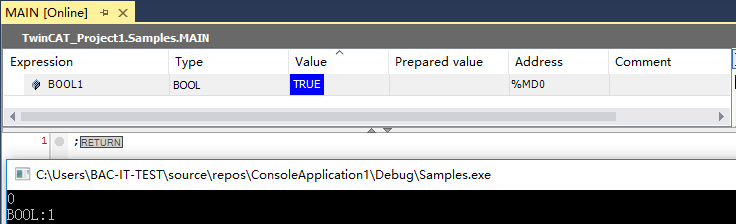
cout << "BOOL:"; //在控制台上输出BOOL:

cin >> BOOL1; //为BOOL1赋值

nErr = AdsSyncWriteReq( pAddr, 0x4020, 0x0, 0x1, &BOOL1 ); //同步写数据到ADS设备，pAddr：ADS设备的地址，0x4020：段地址，0x0偏移地址，0x1：数据长度，@BOOL1:接收数据的缓存

if (nErr) cerr << "Error: AdsSyncWriteReq: " << nErr << '\n'; //检查获取地址的操作是否执行成功

1. 编写程序后，编译，运行。在控制台“BOOL：”右侧输入“1”，并按“回车”，能看到TwinCAT3中的BOOL1变为“TRUE”了，效果如图所示：



这样从控制台向PLC写入布尔量的程序就完成了。

# 通过C++程序读取/写入PLC程序中的整型量

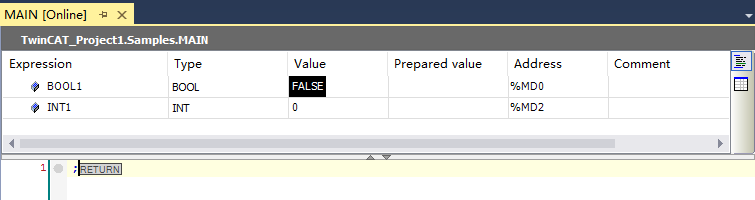
## （整型量读取）编写PLC程序

1. 打开TwinCAT3编写之前的PLC程序

2. 在变量声明区定义一个名称为INT1的INT型变量，并赋予其地址“AT %MD2”

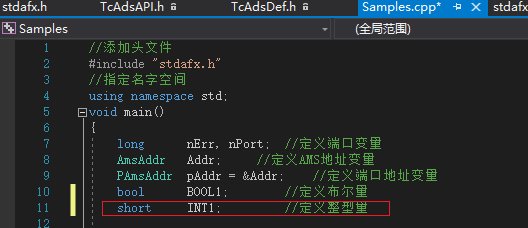


3. 编写完程序后，重新编译运行程序并LOG IN打开851端口连接，效果如下：



## （整型量读取）编写C++程序

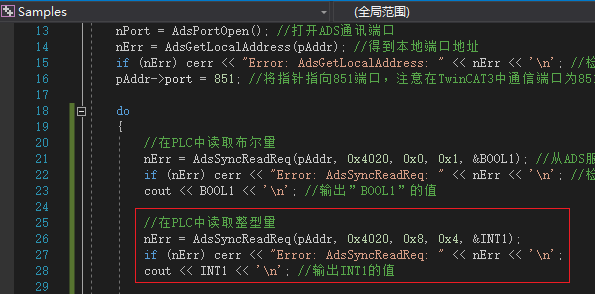
1. 打开Visual Studio 2017，创建一个Windows控制台项目，并且为项目配置好所需链接库。
2. 编写程序，在程序中定义一个”INT1”的整型量，如图所示：



代码：

short INT1; //定义整型量

1. 接下来在程序中编写“**从PLC中读取整型量**”的程序，如图所示:



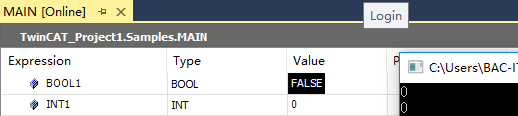
代码：

nErr = AdsSyncReadReq(pAddr, 0x4020, 0x8, 0x4, & INT1);

if (nErr) cerr << "Error: AdsSyncReadReq: " << nErr << '\n';

cout << INT1 << '\n'; //输出INT1的值

1. 程序编写完成后点击菜单Project->build solution,当编译没有错误后，点击菜单Debug->Start Debugging运行程序，出现如下图示：



可以看到，TwinCAT3 PLC中的整型量“INT1”已经读取到的控制台上，这样从PLC中读取整型量的值就完成了。

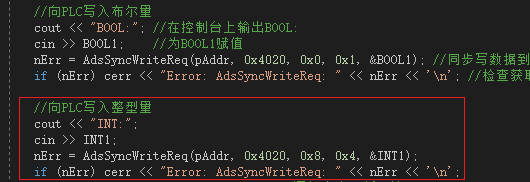
接下来教大家如何从控制台向PLC中写入整型量。

## （整型量写入）编写PLC程序

1. 打开TwinCAT3进入编写PLC界面
2. 编写程序并运行（与读取部分相同）

## （整型量写入）编写C++程序

1. 同样打开Visual Studio 2017，创建一个控制台项目，并且为项目配置好所需链库。
2. 编写程序，在已经编写好的“**整型量读取**”程序中输入“**向PLC写入整型量**” 的程序即可（注意TC2和TC3在寻址上有差异，故在此Dword偏移地址为0x8而不是0x2），如图所示：



代码：

//向PLC写入整型量

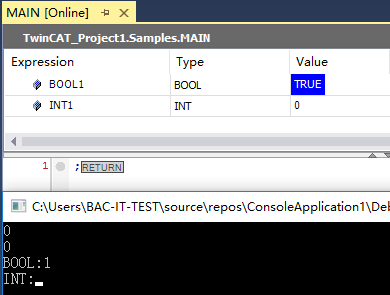
cout << "INT:";

cin >>INT1 ;

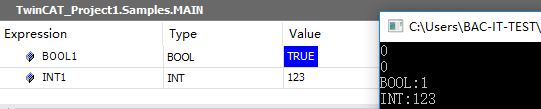
nErr = AdsSyncWriteReq( pAddr, 0x4020, 0x8, 0x4, &INT1 );

if (nErr) cerr << "Error: AdsSyncWriteReq: " << nErr << '\n';

1. 编写完成后，编译，运行，效果如图所示：



1. 在控制台“INT：”右侧输入“123”，并按“回车”，能看到TwinCAT3中的INT1变为“123”了，效果如图所示：

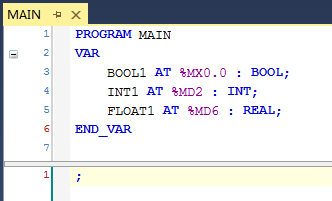


这样从控制台向PLC写入整形量的程序就完成了。

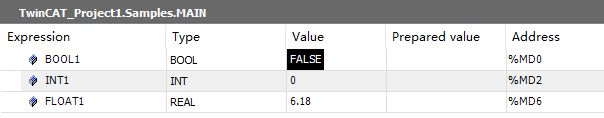
# 通过C++程序读取/写入PLC程序中的浮点数

## （浮点数读取）编写PLC程序

1. 同样打开TwinCAT3 PLC界面，在变量声明区定义一个实型变量“FLOAT1”，如图所示：

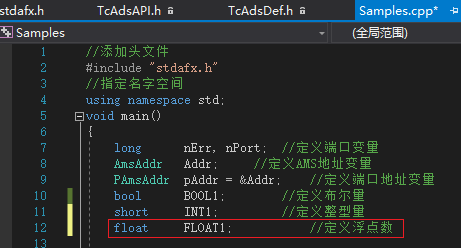


1. 最后编译运行程序并LOG IN打开851通讯端口，并赋予其初值：6.18，如图所示：



## （浮点数读取）编写C++程序

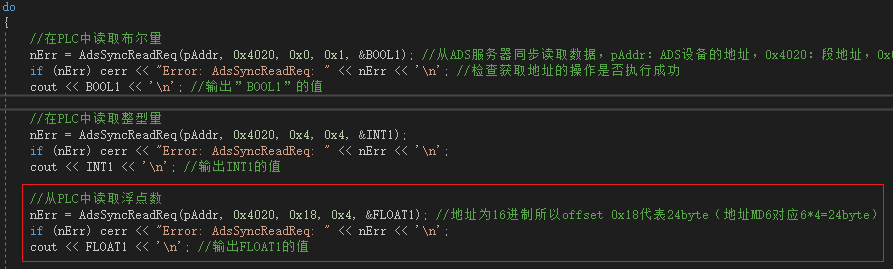
1. 打开Visual Studio 2017，创建一个Windows控制台项目，并且为项目配置好所需链接库。
2. 编写程序，在程序中定义一个”FLOAT1”的整型量，如图所示：



代码：

float FLOAT1; //定义浮点数

1. 接下来在程序中编写“**从PLC中读取浮点数**”的程序，如图所示:



代码：

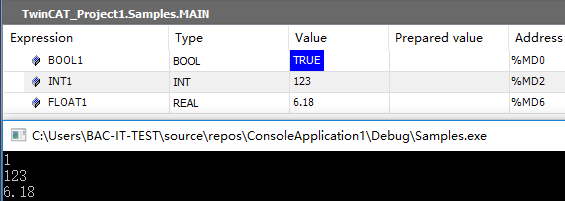
//从PLC中读取浮点数

nErr = AdsSyncReadReq(pAddr, 0x4020, 0x18, 0x4, & FLOAT1); //地址为16进制所以0x18代表24byte（地址MD6在TwinCAT3中对应6\*4=24byte的段地址）

if (nErr) cerr << "Error: AdsSyncReadReq: " << nErr << '\n';

cout << FLOAT1 << '\n';

6.编写完成后，编译，运行，效果如图所示：



能看到PLC中的浮点数：6.18已经显示在的控制台上了。

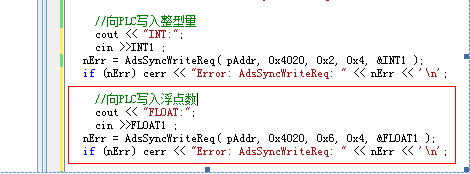
从PLC中读取浮点数就介绍到这里，接下来讲解如何向PLC写入浮点数。

## （浮点数写入）编写PLC程序

1. 打开TwinCAT3 PLC编写界面
2. 编写程序并运行（与读取部分相同）

## （浮点数写入）编写C++程序

1. 打开Visual Studio 2017，创建一个Windows控制台项目，并且为项目配置好所需链接库。
2. 编写程序，在已经编写好的“浮点数读取”程序中输入“向PLC写入浮点数”的程序即可，如图所示：



代码：

//向PLC写入浮点数

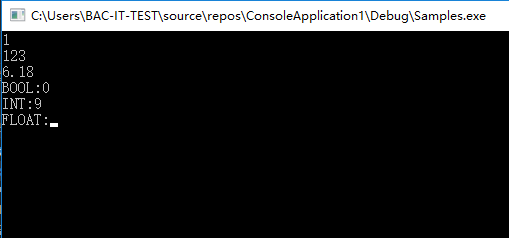
cout << "FLOAT:";

cin >>FLOAT1 ;

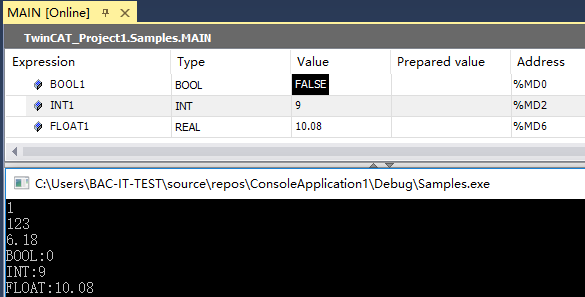
nErr = AdsSyncWriteReq( pAddr, 0x4020, 0x18, 0x4, &FLOAT1 );

if (nErr) cerr << "Error: AdsSyncWriteReq: " << nErr << '\n';

1. 编写完成后，编译，运行程序，效果如图所示：



1. 在“FLOAT：”的右侧输入10.08，单击“回车”，能从TwinCAT3 PLC中看到写入的浮点数。如下图所示：

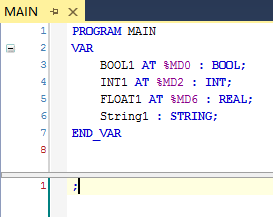


这样就完成了向PLC写入浮点数的操作了。

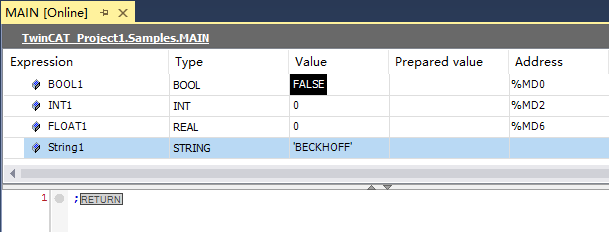
# 通过C++程序读取/写入PLC程序中的字符串

## （字符串读取）编写PLC程序

1. 打开TwinCAT3进入PLC编写界面
2. 在变量声明区定义一个字符串型变量“String1”，如图所示：

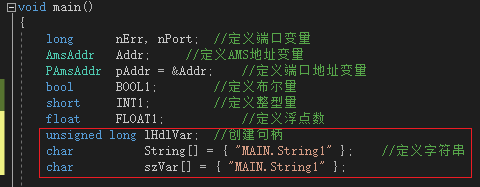


1. 编写完成后，运行程序，并且在String1中输入“BECKHOFF”效果如下：



## （字符串读取）编写C++程序

1. 打开Visual Studio 2017，创建一个Windows控制台项目，并且为项目配置好所需链接库。
2. 编写程序，在程序中定义一个“String”的字符串型变量，由于“**字符串读写**”的程序是通过句柄方式来编写的，所以还需定义句柄变量“lHdlVar”如图所示：



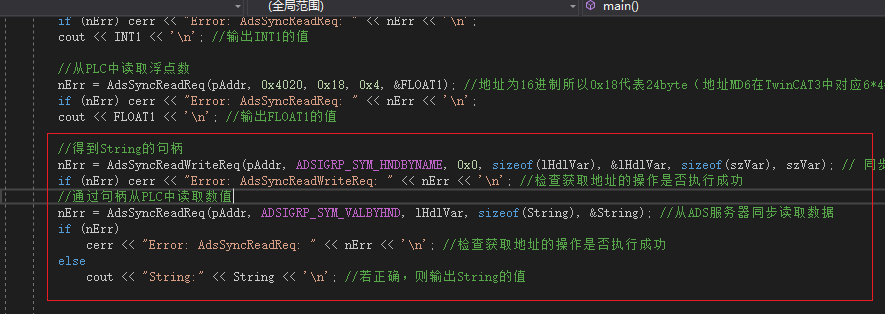
代码：

unsigned long lHdlVar; //创建句柄

char String[]={"MAIN.String1"}; //定义字符串

char szVar []={"MAIN.String1"};

1. 在程序中编写“得到String1的句柄”，“通过句柄从PLC中读取字符串”的程序，如图所示：



代码：

nErr = AdsSyncReadWriteReq(pAddr, ADSIGRP\_SYM\_HNDBYNAME, 0x0, sizeof(lHdlVar), &lHdlVar, sizeof(szVar), szVar); // 同步写数据到ADS服务器并从ADS设备接收数据，pAddr：ADS设备的地址 0x0：偏移地址 sizeof(lHdlVar)：由ADS设备返回的句柄大小 &lHdlVar：由ADS设备返回的数据缓存 sizeof(szVar)：写入ADS设备的数据大小 szVar：写入ADS设备的数据缓存

if (nErr) cerr << "Error: AdsSyncReadWriteReq: " << nErr << '\n'; //检查获取地址的操作是否执行成功

nErr = AdsSyncReadReq(pAddr, ADSIGRP\_SYM\_VALBYHND, lHdlVar, sizeof(String), &String); //从ADS服务器同步读取数据

if (nErr)

cerr << "Error: AdsSyncReadReq: " << nErr << '\n'; //检查获取地址的操作是否执行成功

else

cout << "String:"<< String << '\n'; //若正确，则输出String的值

1. 程序编写完成后运行程序，效果如下图示：



看到，TwinCAT3 PLC中的字符串“BECKHOFF”已经读取到的控制台上，这样从PLC中读取字符串的程序就完成了。

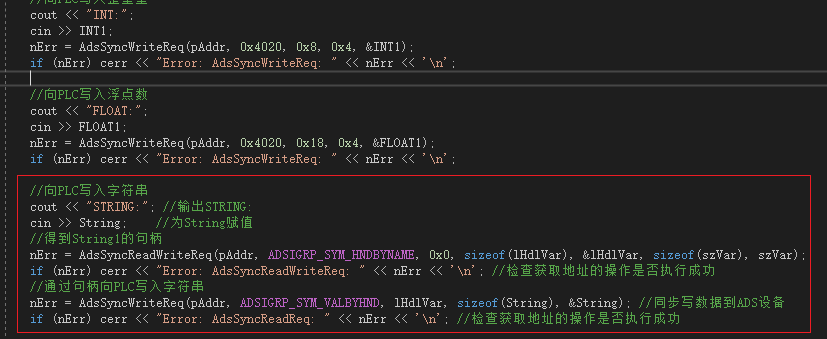
接下来教大家如何从控制台向PLC中写入字符串。

## （字符串写入）编写PLC程序

1. 打开TwinCAT3进入PLC编写界面。
2. 编写程序并运行（读取部分相同）。

## （字符串写入）编写C++程序

1. 同样打开Visual Studio 2017，创建一个Windows控制台项目，并且为项目配置好所需链接库。
2. 编写程序，在已经编写好的“字符串读取”程序中输入“向PLC写入字符串”，“得到String1的句柄”，“通过句柄向PLC写入字符串”的程序，如图所示：



代码：

//向PLC写入字符串

cout<<"STRING:"; //输出STRING:

cin>> String; //为String赋值

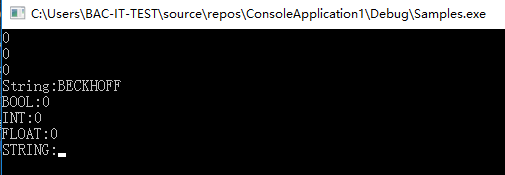
nErr = AdsSyncReadWriteReq(pAddr, ADSIGRP\_SYM\_HNDBYNAME, 0x0, sizeof(lHdlVar), &lHdlVar, sizeof(szVar), szVar);

if (nErr) cerr << "Error: AdsSyncReadWriteReq: " << nErr << '\n'; //检查获取地址的操作是否执行成功

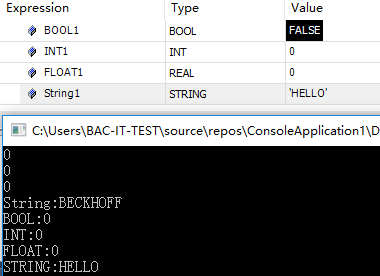
nErr = AdsSyncWriteReq(pAddr, ADSIGRP\_SYM\_VALBYHND, lHdlVar, sizeof(String), &String); //同步写数据到ADS设备

if (nErr) cerr << "Error: AdsSyncReadReq: " << nErr << '\n'; //检查获取地址的操作是否执行成功

1. 编写完成后，编译，运行，如图所示：



1. 在控制台“STRING：”右侧输入“HELLO”字符，能看到已将字符写入到PLC中了，效果如图所示：

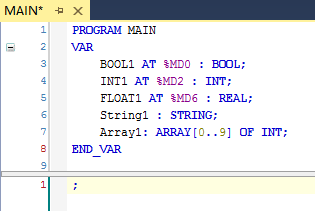


这样从控制台向PLC写入字符串的程序就完成了。

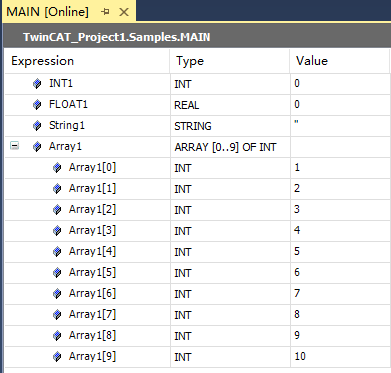
# 通过C++程序读取/写入PLC程序中的数组

## （数组读取）编写PLC程序

1. 打开TwinCAT3进入PLC界面
2. 在变量声明区定义一个含10个整型元素的数组“Array1”，如图所示：

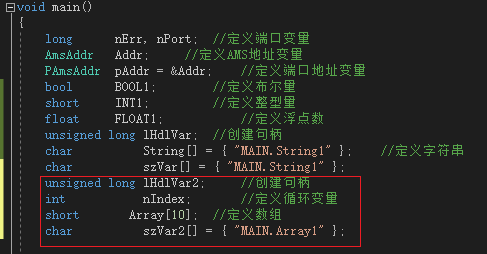


1. 编写完成后，运行程序，并且为数组赋初值，效果如下：



## （数组读取）编写C++程序

1. 打开Visual Studio 2017，创建一个Windows控制台项目，并且为项目配置好所需链接库。
2. 编写程序，在程序中定义一个“Array”的字符串型变量，同样“**数组读写**”的程序是通过句柄方式来编写的，所以还需定义句柄变量“lHdlVar2”如图所示：



代码：

unsigned long lHdlVar2; //创建句柄

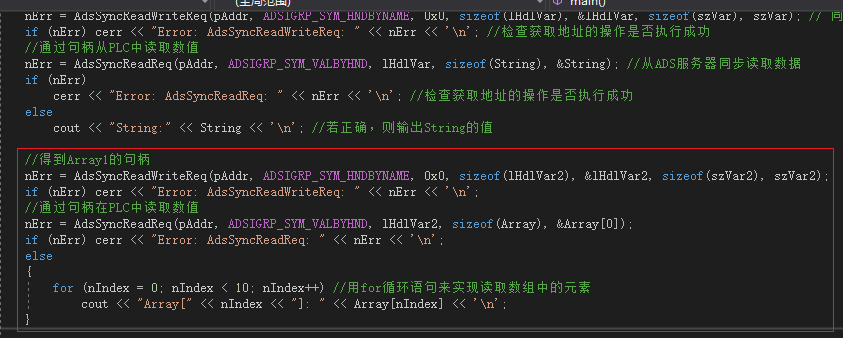
int nIndex; //定义循环变量

short Array[10]; //定义数组

char szVar2[]={"MAIN.Array1"};

1. 在程序中编写“得到Array1的句柄”，“通过句柄从PLC中读取数组”的程序，

如图所示：



代码：

nErr = AdsSyncReadWriteReq(pAddr, ADSIGRP\_SYM\_HNDBYNAME, 0x0, sizeof(lHdlVar2), &lHdlVar2, sizeof(szVar2), szVar2);

if (nErr) cerr << "Error: AdsSyncReadWriteReq: " << nErr << '\n';

nErr = AdsSyncReadReq(pAddr, ADSIGRP\_SYM\_VALBYHND, lHdlVar2, sizeof(Array), & Array [0]);

if (nErr) cerr << "Error: AdsSyncReadReq: " << nErr << '\n';

else

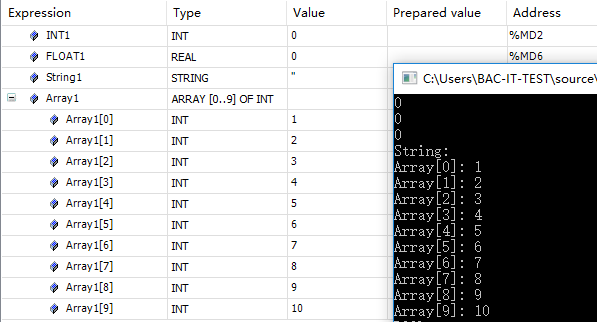
{

for (nIndex = 0; nIndex < 10; nIndex++) //用for循环语句来实现读取数组中的元素

cout << "Array[" << nIndex << "]: " << Array[nIndex] << '\n';

}

1. 程序编写完成后运行程序，效果如下图示：



看到，TwinCAT3 PLC中数组中的10个元素，有序地显示在的控制台上了，这样从PLC中读取数组的程序就完成了。

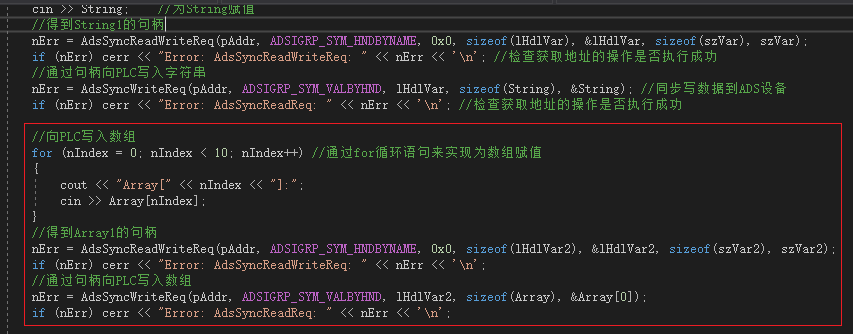
接下来教大家如何从控制台向PLC中写入数组。

## （数组写入）编写PLC程序

1. 打开TwinCAT3进入PLC编写界面
2. 编写程序并运行（与读取部分相同）

## （数组写入）编写C++程序

1. 同样打开Visual Studio 2017，创建一个Windows控制台项目，并且为项目配置好所需链接库。
2. 编写程序，在已经编写好的“**数组读取**”程序中输入“**向PLC写入数组**”，“**得到Array1的句柄**”，“**通过句柄向PLC写入数组**”的程序，如图所示：



代码：

for (nIndex = 0; nIndex < 10; nIndex++) //通过for循环语句来实现为数组赋值

{ cout << "Array[" << nIndex << "]:";

cin>> Array[nIndex];

}

//得到Array1的句柄

nErr = AdsSyncReadWriteReq(pAddr, ADSIGRP\_SYM\_HNDBYNAME, 0x0, sizeof(lHdlVar2), &lHdlVar2, sizeof(szVar2), szVar2);

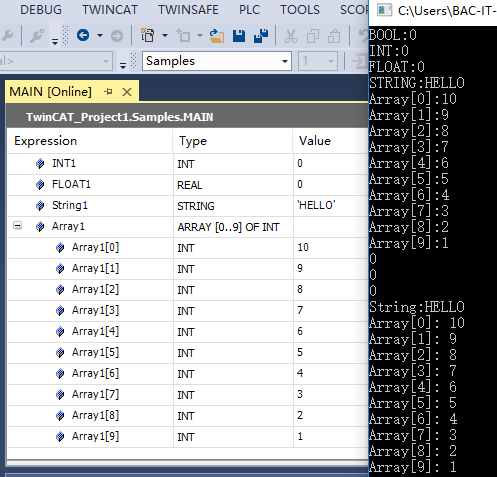
if (nErr) cerr << "Error: AdsSyncReadWriteReq: " << nErr << '\n';

//通过句柄向PLC写入数组

nErr = AdsSyncWriteReq(pAddr, ADSIGRP\_SYM\_VALBYHND, lHdlVar2, sizeof(Array), & Array[0]);

if (nErr) cerr << "Error: AdsSyncReadReq: " << nErr << '\n';

1. 编写完成后，编译，运行，并为数组赋值后，按“回车”，效果如图所示：



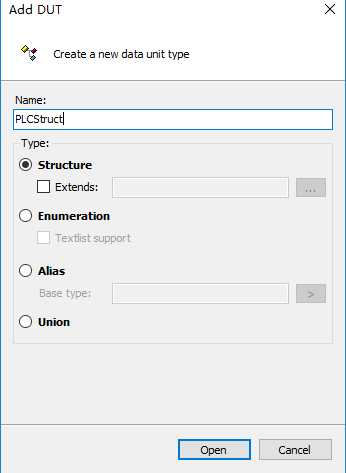
可以看到数组已写入到PLC中了。

这样从控制台向PLC写入数组的程序就完成了。

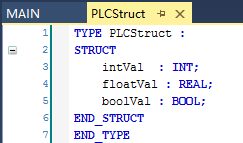
# 通过C++程序读取/写入PLC程序中的结构体

## （结构体读取）编写PLC程序

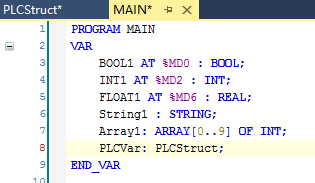
1. 打开TwinCAT3进入PLC编写界面。
2. 编写程序,由于这次编写的是结构体，所以先在DATA TYPES中编写一个名为“PLCStruct”的结构体。具体操作为在左菜单栏Project->References->DUTs，右键点击DUTs->Add->DUT并在ADD DUT窗口选择Structure type，点击Open。如下图所示：



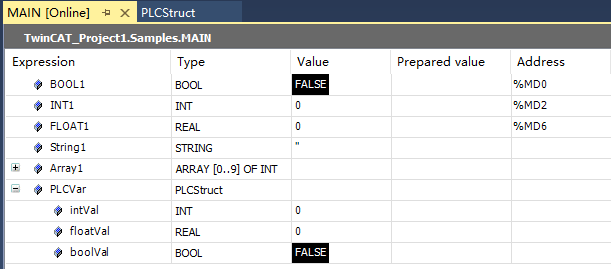
1. 在“PLCStruct”中定义3个不同类型的变量。如下图所示。



1. 随后再在“Main”程序中，定义一个名为“PLCVar”，类型为“PLCStruct”的变量。完成之后，以“；”结束。图示：

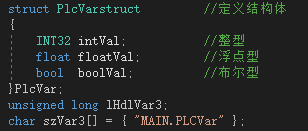


1. 运行程序，随后展开“PLCVar”中的元素。如图所示：



## （结构体读取）编写C++程序

1. 打开Visual Studio 2017，创建一个Windows控制台项目，并且为项目配置好所需链接库。
2. 接下来，开始编写主程序，结构体程序较为复杂，首先先定义变量“**定义结构体**”，同样在“结构体”里定义对应的三个不同类型的变量以及结构体的句柄，程序如图所示：



代码：

struct PlcVarstruct //定义结构体

{

INT32 intVal; //整型

float floatVal; //浮点型

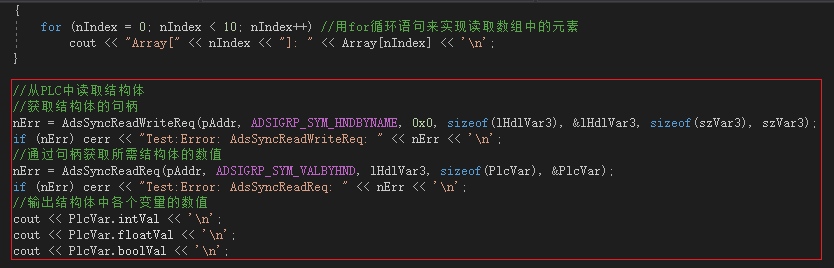
bool boolVal; //布尔型

}PlcVar;

unsigned long lHdlVar3;

char szVar3[] = { "MAIN.PLCVar" };

1. 在程序中编写“**从PLC中读取结构体**”的程序，如图所示：



代码：

//从PLC中读取结构体

//获取结构体的句柄

nErr = AdsSyncReadWriteReq(pAddr, ADSIGRP\_SYM\_HNDBYNAME, 0x0, sizeof(lHdlVar3), &lHdlVar3, sizeof(szVar3), szVar3);

if (nErr) cerr << "Test:Error: AdsSyncReadWriteReq: " << nErr << '\n';

//通过句柄获取所需结构体的数值

nErr = AdsSyncReadReq(pAddr, ADSIGRP\_SYM\_VALBYHND, lHdlVar3, sizeof(PlcVar), &PlcVar);

if (nErr) cerr << "Test:Error: AdsSyncReadReq: " << nErr << '\n';

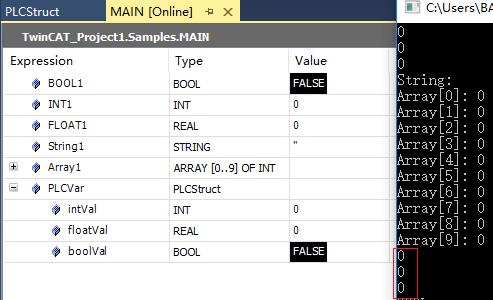
//输出结构体中各个变量的数值

cout << PlcVar.intVal << '\n';

cout << PlcVar.floatVal << '\n';

cout<<PlcVar.boolVal<<'\n';

1. 编写完成后，编译并运行程序，如图所示：



可以看到，TwinCAT3 PLC中“结构体”中的变量已显示到的控制台上了。

通过控制台从PLC中读取结构体就介绍到这里，接下来来讲解如何从控制台向PLC 中写入结构体。

## （结构体写入）编写PLC程序

1. 打开TwinCAT3进入PLC编写界面
2. 编写程序并运行（与读取部分相同）

## （结构体写入）编写C++程序

1. 同样打开Visual Studio 2017，创建一个Windows控制台项目，并且为项目配置好所需链接库。
2. 编写程序，在已经编写好的“**结构体读取**”程序中输入“**向PLC写入结构体**”的程序，如图所示：

电脑萤幕的截图

描述已自动生成

代码：

//向PLC写入结构体

//输入结构体的数值

cout << "PlcVar.intVal: ";

cin >> PlcVar.intVal;

cout << "PlcVar.floatVal: ";

cin >> PlcVar.floatVal;

cout << "PlcVar.boolVal: ";

cin >> PlcVar.boolVal;

cout << '\n';

//获取句柄

nErr = AdsSyncReadWriteReq(pAddr, ADSIGRP\_SYM\_HNDBYNAME, 0x0, sizeof(lHdlVar3), &lHdlVar3, sizeof(szVar3), szVar3);

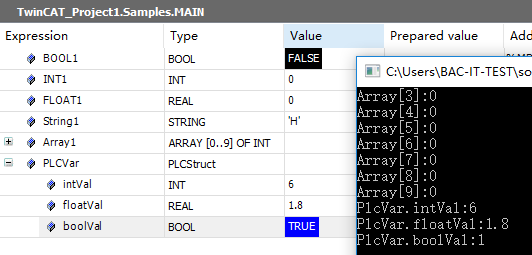
if (nErr) cerr << "Test: Error: AdsSyncReadWriteReq: " << nErr << '\n';

//通过之前获取的句柄向PLC写入结构体

nErr = AdsSyncWriteReq(pAddr, ADSIGRP\_SYM\_VALBYHND, lHdlVar3, sizeof(PlcVar), &PlcVar);

if (nErr) printf("Error: Ads: Write struct: %d\n", nErr);

1. 编写完成后，编译，运行，并为结构体赋值后，按“回车”，效果如图所示：



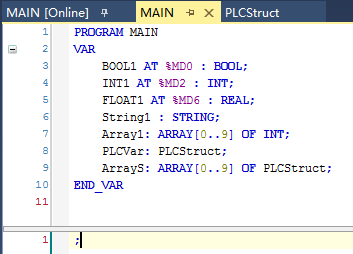
能看到结构体已写入到PLC中了。

这样从控制台向PLC写入结构体的程序就完成了。

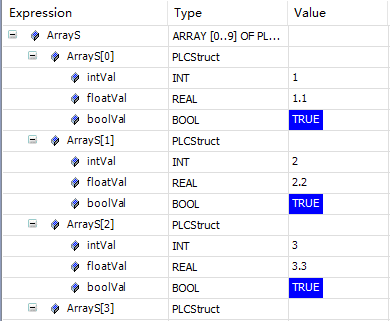
# 通过C++程序读取/写入PLC程序中结构体类型的数组

## （结构体类型的数组读取）编写PLC程序

1. 打开TwinCAT3进入PLC界面。
2. 在变量声明区定义一个含10个之前定义的PLCStruct结构体的数组“ArrayS”，如图所示：

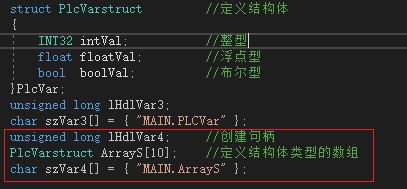


1. 编写完成后，运行程序，并且为结构体类型的数组赋初值，效果如下：



## （结构体类型的数组读取）编写C++程序

1. 打开Visual Studio 2017，创建一个Windows控制台项目，并且为项目配置好所需链接库。
2. 编写程序，在程序中定义一个“ArrayS”的结构体类型的数组，同样“**结构体类型的数组读写**”的程序是通过句柄方式来编写的，所以还需定义句柄变量“lHdlVar4”如图所示：



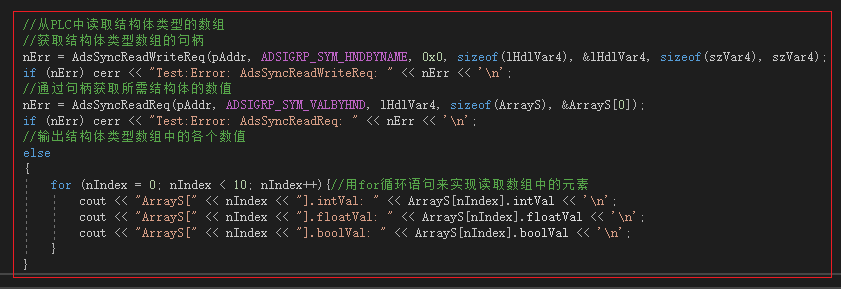
代码：

unsigned long lHdlVar4; //创建句柄

PlcVarstruct ArrayS[10]; //定义结构体类型的数组

char szVar4[] = { "MAIN.ArrayS" };

1. 在程序中编写“得到ArrayS的句柄”，“通过句柄从PLC中读取结构体类型的数组”的程序，如图所示：



代码：

//从PLC中读取结构体类型的数组

//获取结构体类型数组的句柄

nErr = AdsSyncReadWriteReq(pAddr, ADSIGRP\_SYM\_HNDBYNAME, 0x0, sizeof(lHdlVar4), &lHdlVar4, sizeof(szVar4), szVar4);

if (nErr) cerr << "Test:Error: AdsSyncReadWriteReq: " << nErr << '\n';

//通过句柄获取所需结构体的数值

nErr = AdsSyncReadReq(pAddr, ADSIGRP\_SYM\_VALBYHND, lHdlVar4, sizeof(ArrayS), &ArrayS[0]);

if (nErr) cerr << "Test:Error: AdsSyncReadReq: " << nErr << '\n';

//输出结构体类型数组中的各个数值

else

{

for (nIndex = 0; nIndex < 10; nIndex++){//用for循环语句来实现读取数组中的元素

cout << "ArrayS[" << nIndex << "].intVal: " << ArrayS[nIndex].intVal << '\n';

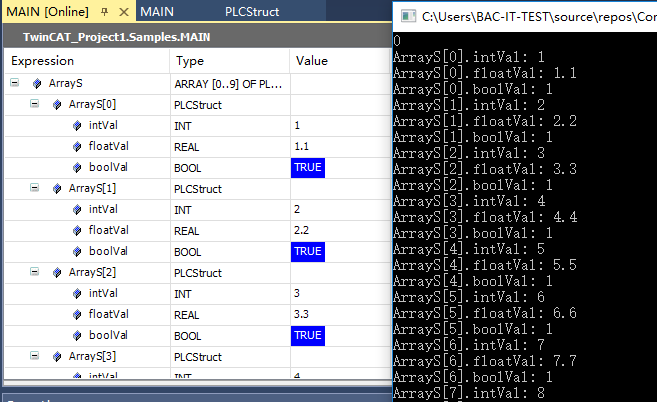
cout << "ArrayS[" << nIndex << "].floatVal: " << ArrayS[nIndex].floatVal << '\n';

cout << "ArrayS[" << nIndex << "].boolVal: " << ArrayS[nIndex].boolVal << '\n';

}

}

1. 程序编写完成后运行程序，效果如下图示：



看到，TwinCAT3 PLC中结构体类型的数组中的10个结构体元素，有序地显示在的控制台上了，这样从PLC中读取结构体类型的数组的程序就完成了。

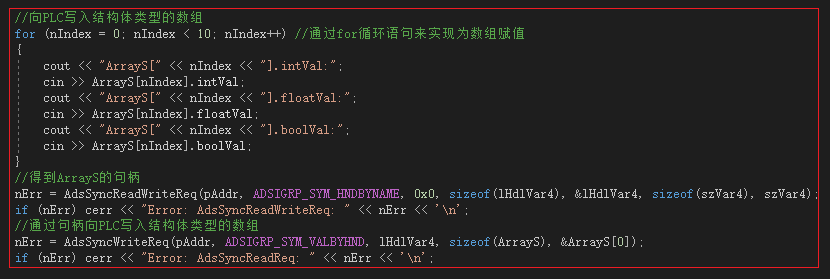
接下来教大家如何从控制台向PLC中写入结构体类型的数组。

## （结构体类型的数组写入）编写PLC程序

1. 打开TwinCAT3进入PLC编写界面
2. 编写程序并运行（与读取部分相同）

## （结构体类型的数组写入）编写C++程序

1. 同样打开Visual Studio 2017，创建一个Windows控制台项目，并且为项目配置好所需链接库。
2. 编写程序，在已经编写好的“结构体类型的数组读取”程序中输入“向PLC写入结构体类型的数组”，“得到ArrayS的句柄”，“通过句柄向PLC写入结构体类型的数组”的程序，如图所示：



代码：

//向PLC写入结构体类型的数组

for (nIndex = 0; nIndex < 10; nIndex++) //通过for循环语句来实现为数组赋值

{

cout << "ArrayS[" << nIndex << "].intVal:";

cin >> ArrayS[nIndex].intVal;

cout << "ArrayS[" << nIndex << "].floatVal:";

cin >> ArrayS[nIndex].floatVal;

cout << "ArrayS[" << nIndex << "].boolVal:";

cin >> ArrayS[nIndex].boolVal;

}

//得到ArrayS的句柄

nErr = AdsSyncReadWriteReq(pAddr, ADSIGRP\_SYM\_HNDBYNAME, 0x0, sizeof(lHdlVar4), &lHdlVar4, sizeof(szVar4), szVar4);

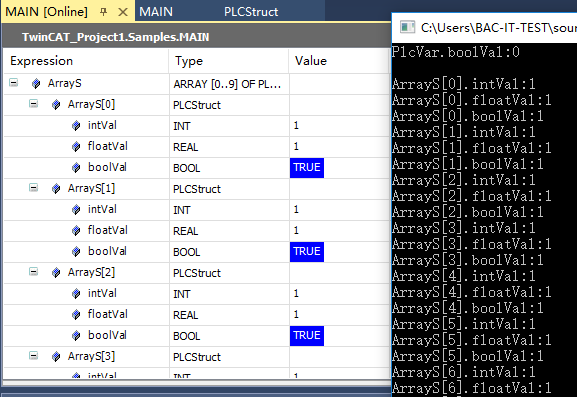
if (nErr) cerr << "Error: AdsSyncReadWriteReq: " << nErr << '\n';

//通过句柄向PLC写入结构体类型的数组

nErr = AdsSyncWriteReq(pAddr, ADSIGRP\_SYM\_VALBYHND, lHdlVar4, sizeof(ArrayS), &ArrayS[0]);

if (nErr) cerr << "Error: AdsSyncReadReq: " << nErr << '\n';

1. 编写完成后，编译，运行，并为结构体类型的数组赋值后，按“回车”，效果如图所示：



能看到结构体类型的数组已写入到PLC中了。

这样从控制台向PLC写入结构体类型的数组的程序就完成了。

# 通过定时方式来读取PLC程序中的变量

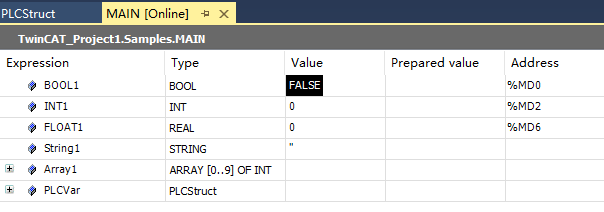
ADS 设备之间的通讯有多种方式，不同方式有不同的特点。以上编写的程序都是同步方式。

同步方式（Synchronous）：ADS 客户端向ADS 服务器发送ADS 请求，同时客户端继续自己的工作。ADS 服务器处理请求后，把响应以Call-back 函数方式发给客户端。除了上述方法外，还可以通过定时方式读取来自于PLC的数据。

定时方式：定时方式就是设定一个接收的间隔时间，程序每过一段时间自动地进行一次数据更新。

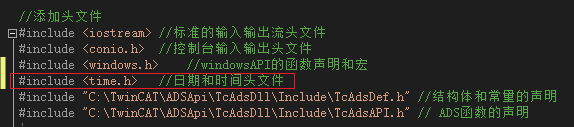
## 编写PLC程序

1. 打开TwinCat3进入PLC编写界面
2. 在变量声明区定义变量，编译并运行，效果如下：



## 编写C++程序

1. 打开Visual Studio 2017，创建一个Windows控制台项目，并且为项目配置好所需链接库。
2. 编写程序，用定时方式来读取PLC中的数据时，需要在“stdafx.h”添加头文件“time.h”，如图所示：



代码：

//添加头文件

#include "stdafx.h"

using namespace std;

void delay(int sec) //编写定时器

{

time\_t start\_time, cur\_time; //变量声明

time(&start\_time);

do

{

time(&cur\_time);

}while ((cur\_time - start\_time) < sec);

}

void main()

{

long nErr, nPort;

AmsAddr Addr;

PAmsAddr pAddr = &Addr;

bool BOOL1;

short INT1;

float FLOAT1;

unsigned long lHdlVar;

char String[] = { "MAIN.String1" };

char szVar[] = { "MAIN.String1" };

unsigned long lHdlVar2;

int nIndex;

short Array[10];

char szVar2[] = { "MAIN.Array1" };

struct PlcVarstruct

{

float floatVal;

INT32 intVal;

bool boolVal;

}PlcVar;

nPort = AdsPortOpen();

nErr = AdsGetLocalAddress(pAddr);

if (nErr) cerr << "Error: AdsGetLocalAddress: " << nErr << '\n';

pAddr->port = 851;

do

{

//在PLC中读取布尔量

nErr = AdsSyncReadReq(pAddr, 0x4020, 0x0, 0x1, &BOOL1);

if (nErr) cerr << "Error: AdsSyncReadReq: " << nErr << '\n';

cout << BOOL1 << '\n';

//在PLC中读取整型量

nErr = AdsSyncReadReq(pAddr, 0x4020, 0x4, 0x4, &INT1);

if (nErr) cerr << "Error: AdsSyncReadReq: " << nErr << '\n';

cout << INT1 << '\n';

//从PLC中读取浮点数

nErr = AdsSyncReadReq(pAddr, 0x4020, 0x18, 0x4, &FLOAT1);

if (nErr) cerr << "Error: AdsSyncReadReq: " << nErr << '\n';

cout << FLOAT1 << '\n';

//得到String的句柄

nErr = AdsSyncReadWriteReq(pAddr, ADSIGRP\_SYM\_HNDBYNAME, 0x0, sizeof(lHdlVar), &lHdlVar, sizeof(szVar), szVar);

if (nErr) cerr << "Error: AdsSyncReadWriteReq: " << nErr << '\n';

//通过句柄从PLC中读取数值

nErr = AdsSyncReadReq(pAddr, ADSIGRP\_SYM\_VALBYHND, lHdlVar, sizeof(String), &String);

if (nErr)

cerr << "Error: AdsSyncReadReq: " << nErr << '\n';

else

cout << "String:" << String << '\n';

//得到Array1的句柄

nErr = AdsSyncReadWriteReq(pAddr, ADSIGRP\_SYM\_HNDBYNAME, 0x0, sizeof(lHdlVar2), &lHdlVar2, sizeof(szVar2), szVar2);

if (nErr) cerr << "Error: AdsSyncReadWriteReq: " << nErr << '\n';

//通过句柄在PLC中读取数值

nErr = AdsSyncReadReq(pAddr, ADSIGRP\_SYM\_VALBYHND, lHdlVar2, sizeof(Array), &Array[0]);

if (nErr) cerr << "Error: AdsSyncReadReq: " << nErr << '\n';

else

{

for (nIndex = 0; nIndex < 10; nIndex++)

cout << "Array[" << nIndex << "]: " << Array[nIndex] << '\n';

}

//从PLC中读取结构体

nErr = AdsSyncReadReq(pAddr, 0x4020, 0x28, 0x4, &PlcVar.intVal);

if (nErr) cerr << "Error: AdsSyncReadReq: " << nErr << '\n';

cout << PlcVar.intVal << '\n';

nErr = AdsSyncReadReq(pAddr, 0x4020, 0x38, 0x4, &PlcVar.floatVal);

if (nErr) cerr << "Error: AdsSyncReadReq: " << nErr << '\n';

cout << PlcVar.floatVal << '\n';

nErr = AdsSyncReadReq(pAddr, 0x4020, 0x48, 0x1, &PlcVar.boolVal);

if (nErr) cerr << "Error: AdsSyncReadReq: " << nErr << '\n';

cout << PlcVar.boolVal << '\n';

cout << '\n';

delay(5); //定时5秒

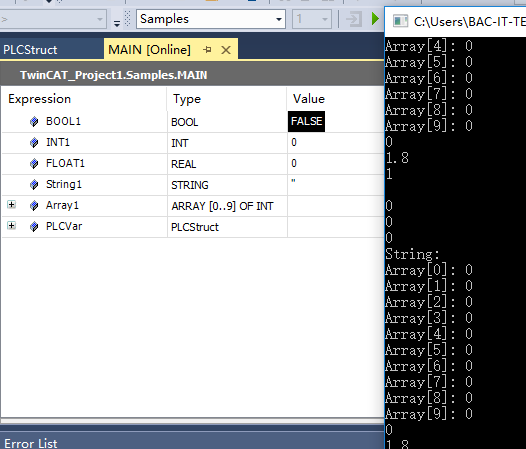
} while (1);

nErr = AdsPortClose(); //关闭ADS通讯端口

if (nErr) cerr << "Error: AdsPortClose: " << nErr << '\n';

}

1. 程序编写完成后，编译并运行，效果如图所示：



可以看到，控制台每隔5秒就从PLC中读取一次数据。

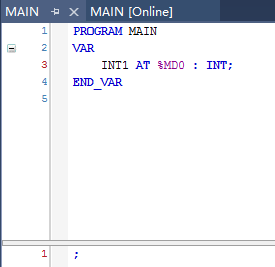
# 通过通知方式来读取PLC程序中的变量

在使用异步方式时，函数AdsSyncReadReq()连续读取来自于PLC的数据,则程序的运行效率会很低，因为该函数必须被循环调用。而定时方式在不改变变量的情况下会产生空的数据传输，浪费了数据的流量。这时候就需要另一种控制方式—通知方式。

通知方式（Notification）：ADS 客户端向ADS 服务器发送ADS 请求，ADS 服务器以Call-back 函数的方式不断向客户端发送响应，直到客户端取消该请求，其优点在于TwinCAT服务器仅仅在数据发生变化时才开始传输数据。

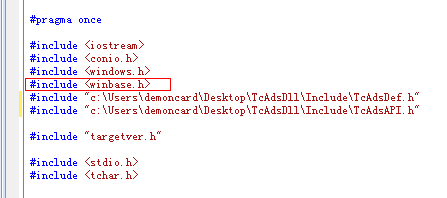
## 编写PLC程序

1. 打开TwinCAT3进入PLC编程界面。
2. 在变量声明区定义一个INT1的整型变量，并赋予其地址“%MD0”，编译并运行，效果如下：



## 编写C++程序

1. 打开Visual Studio 2017，创建一个Windows控制台项目，并且为项目配置好所需链接库。
2. 编写程序，用通知方式来读取PLC中的数据时，需要在“stdafx.h”添加头文件“winbase.h”，如图所示：



代码：

#include <winbase.h> //windows头文件之一

1. 接着，编写主程序

代码：

#include "stdafx.h"

using namespace std;

void \_stdcall Callback(AmsAddr\*, AdsNotificationHeader\*, unsigned long);

void main() //定义变量

{

long nErr, nPort;

AmsAddr Addr;

PAmsAddr pAddr = &Addr;

ULONG hNotification, hUser;

AdsNotificationAttrib adsNotificationAttrib; //为adsNotificationAttrib包含的所有属性定义一个通知

//打开ADS router的通讯端口

nPort = AdsPortOpen();

nErr = AdsGetLocalAddress(pAddr);

if (nErr) cerr << "Error: AdsGetLocalAddress: " << nErr << '\n';

pAddr->port = AMSPORT\_R0\_PLC\_TC3;

//设置notification的属性值

adsNotificationAttrib.cbLength = 4; //数据的长度，传递给回调函数

adsNotificationAttrib.nTransMode = ADSTRANS\_SERVERONCHA; //当变量值的变化后，通知的回调函数被调用

adsNotificationAttrib.nMaxDelay = 20000000; // 2sec,通知的回调函数被调用

adsNotificationAttrib.nCycleTime = 10000000; // 1sec,ADS服务器检查变量是否改变

// 32-Bit变量(和指针),这些变量(和指针)可以被传递给callback-function

hUser = 3474573467;

//开始PLC变量的传输

nErr = AdsSyncAddDeviceNotificationReq(pAddr, 0x4020, 0, &adsNotificationAttrib, Callback, hUser, &hNotification); //定义在ADS服务器的通知,当某个事件发生时,回调函数在ADS客户端被调用。pAddr：ADS设备地址 0x4020：段地址 0：偏移地址 Callback：回调函数的名称

if (nErr) cerr << "Error: AdsSyncAddDeviceNotificationReq: " << nErr << '\n';

cout << "Notification: " << hNotification << "\n\n";

cout.flush();

//等待用户操作

\_getch();

//完成PLC变量的传输

nErr = AdsSyncDelDeviceNotificationReq(pAddr, hNotification);

if (nErr) cerr << "Error: AdsSyncDelDeviceNotificationReq: " << nErr << '\n';

//关闭通讯端口

nErr = AdsPortClose();

if (nErr) cerr << "Error: AdsPortClose: " << nErr << '\n'; //判断返回值结果

}

//回调函数

void \_\_stdcall Callback(AmsAddr\* pAddr, AdsNotificationHeader\* pNotification, ULONG hUser)

{

int nIndex;

static ULONG nCount = 0;

SYSTEMTIME SystemTime, LocalTime;

FILETIME FileTime;

LARGE\_INTEGER LargeInteger;

TIME\_ZONE\_INFORMATION TimeZoneInformation;

cout << ++nCount << ". Call:\n";

//打印出变量的数值

cout << "INT1:" << \*(ULONG \*)pNotification->data << '\n';

cout << "Notification: " << pNotification->hNotification << '\n';

//将timestamp转变成SYSTEMTIME

LargeInteger.QuadPart = pNotification->nTimeStamp;

FileTime.dwLowDateTime = (DWORD)LargeInteger.LowPart;

FileTime.dwHighDateTime = (DWORD)LargeInteger.HighPart;

FileTimeToSystemTime(&FileTime, &SystemTime);

//将时间值Zeit转变为local time

GetTimeZoneInformation(&TimeZoneInformation);

SystemTimeToTzSpecificLocalTime(&TimeZoneInformation, &SystemTime, &LocalTime);

//打印出timestamp

cout << LocalTime.wHour << ":" << LocalTime.wMinute << ":" << LocalTime.wSecond << '.' << LocalTime.wMilliseconds <<

" den: " << LocalTime.wDay << '.' << LocalTime.wMonth << '.' << LocalTime.wYear << '\n';

// Size of buffer in bytes

cout << "SampleSize: " << pNotification->cbSampleSize << '\n';

// 32-Bit Variable (auch Zeiger), die beim AddNotification gesetzt wurde // (siehe main)

cout << "hUser: " << hUser << '\n';

//打印出发送方的ADS-address

cout << "ServerNetId: ";

for (nIndex = 0; nIndex < 6; nIndex++)

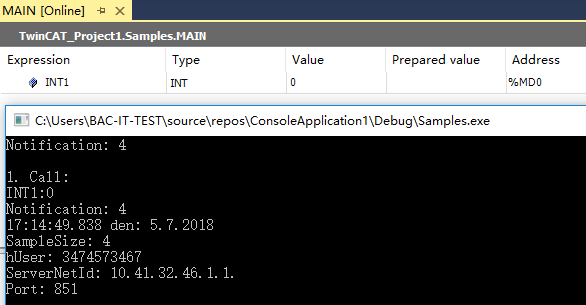
cout << (int)pAddr->netId.b[nIndex] << ".";

cout << "\nPort: " << pAddr->port << "\n\n";

cout.flush();

}

1. 程序编好后，编译并运行，效果如图所示：



当运行程序后，等待3秒，程序会自动地将PLC中的值读入到的控制台上，并且只有当PLC中的数据发生变化时，控制台才会再次读入新数据，这样就避免了数据流量的浪费。

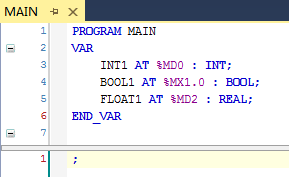
通过通知方式来读取PLC程序中的变量就介绍到这里。

# 通过C++程序在PLC变量被隐藏时读取所有变量信息

之前的读取/写入方式都需要在知道PLC变量名称以及地址的情况下通过C++中的特定功能块向PLC查询变量值，但是有时需要在PLC变量被隐藏的情况下高效查询变量信息。这时可以通过C++查询PLC变量表格，以数据流的方式输出PLC中所有变量的具体信息，以实现监测的目的。

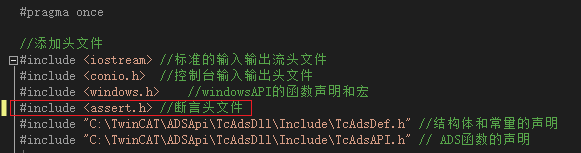
## 编写PLC程序

1. 打开TwinCAT3进入PLC编程界面。
2. 在变量声明区定义一个INT1的整型变量，并赋予其地址“%MD0”；定义一个BOOL1的布尔量，并赋予其地址“%MX1.0”；定义一个FLOAT1的浮点数，并赋予其地址“%MD2”。编译并运行，效果如下：



## 编写C++程序

1. 打开Visual Studio 2017，创建一个Windows控制台项目，并且为项目配置好所需链接库。
2. 编写程序，用通知方式来读取PLC中的数据时，需要在“stdafx.h”添加头文件“assert.h”，如图所示：



代码：

#include <assert.h> //断言头文件

5.接着，编写主程序

代码：

#include "stdafx.h"

using namespace std;

void \_stdcall Callback(AmsAddr\*, AdsNotificationHeader\*, unsigned long);

void main()

{

long nErr, nPort;

char \*pchSymbols = NULL;

UINT uiIndex;

AmsAddr Addr;

PAmsAddr pAddr = &Addr;

AdsSymbolUploadInfo tAdsSymbolUploadInfo;

PAdsSymbolEntry pAdsSymbolEntry;

// 建立端口通讯连接

nPort = AdsPortOpen();

nErr = AdsGetLocalAddress(pAddr);

if (nErr) cerr << "Error: AdsGetLocalAddress: " << nErr << '\n';

pAddr->port = 851;

// 向PLC读取所有变量声明数据流的长度

nErr = AdsSyncReadReq(pAddr, ADSIGRP\_SYM\_UPLOADINFO, 0x0, sizeof(tAdsSymbolUploadInfo), &tAdsSymbolUploadInfo);

if (nErr) cerr << "Error: AdsSyncReadReq: " << nErr << '\n';

pchSymbols = new char[tAdsSymbolUploadInfo.nSymSize];

assert(pchSymbols);

// 向PLC读取所有变量的信息

nErr = AdsSyncReadReq(pAddr, ADSIGRP\_SYM\_UPLOAD, 0, tAdsSymbolUploadInfo.nSymSize, pchSymbols);

if (nErr) cerr << "Error: AdsSyncReadReq: " << nErr << '\n';

// 输出PLC变量的信息

pAdsSymbolEntry = (PAdsSymbolEntry)pchSymbols;

for (uiIndex = 0; uiIndex < tAdsSymbolUploadInfo.nSymbols; uiIndex++)

{

cout << PADSSYMBOLNAME(pAdsSymbolEntry) << "\t\t" //输出变量名称

<< pAdsSymbolEntry->iGroup << '\t' //输出段地址

<< pAdsSymbolEntry->iOffs << '\t' //输出偏移地址

<< pAdsSymbolEntry->size << '\t' //输出变量大小

<< PADSSYMBOLTYPE(pAdsSymbolEntry) << '\t' //输出变量类型

<< PADSSYMBOLCOMMENT(pAdsSymbolEntry) << '\n'; //输出变量备注

pAdsSymbolEntry = PADSNEXTSYMBOLENTRY(pAdsSymbolEntry); cout.flush();

}

\_getch();

// 关闭通讯端口

nErr = AdsPortClose();

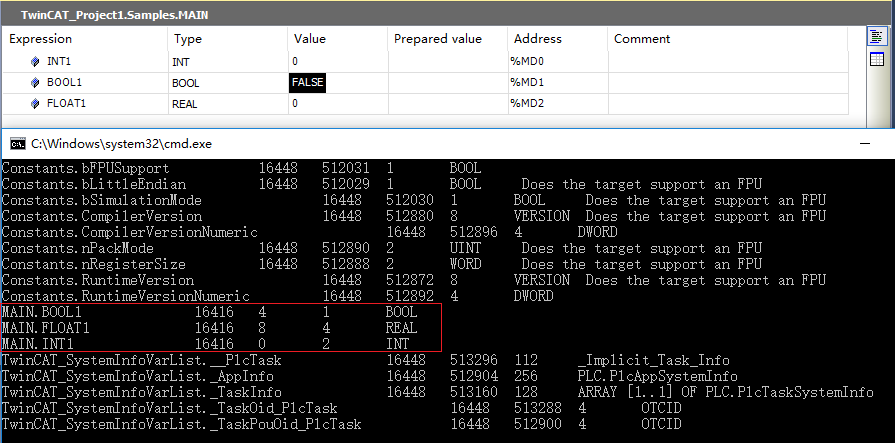
if (nErr) cerr << "Fehler: AdsPortClose: " << nErr << '\n';

// 释放内存

if (pchSymbols) delete(pchSymbols);

}

1. 程序编好后，编译并运行，效果如图所示：



从上图可以看到，通过调用C++的功能模块在不知道PLC变量信息的情况下对PLC变量表进行查询并获得了之前创建的变量信息，包括变量名称，段地址，偏移地址，变量大小，变量类型，变量注释。

**上海（ 中国区总部）**

中国上海市静安区汶水路 299 弄 9号（市北智汇园）

电话: 021-66312666

**北京分公司**

北京市西城区新街口北大街 3 号新街高和大厦 407 室

电话: 010-82200036 邮箱: beijing@beckhoff.com.cn

**广州分公司**

广州市天河区珠江新城珠江东路32号利通广场1303室

电话: 020-38010300/1/2 邮箱: guangzhou@beckhoff.com.cn

**成都分公司**

成都市锦江区东御街18号 百扬大厦2305 室

电话: 028-86202581 邮箱: chengdu@beckhoff.com.cn

|  |  |
| --- | --- |
| 请用微信扫描二维码  通过公众号与技术支持交流 | 倍福官方网站：  https://www.beckhoff.com.cn  在线帮助系统：  https://infosys.beckhoff.com/index\_en.htm |
| 倍福虚拟学院：  https://tr.beckhoff.com.cn/ |
| 招贤纳士：job@beckhoff.com.cn  技术支持：support@beckhoff.com.cn  产品维修：service@beckhoff.com.cn  方案咨询：sales@beckhoff.com.cn |
|  |