LC 为 Client 端

1、运行前准备

1 台 LC3048，1 台电脑，NetAssist 调试软件

① LC3048 做 Client 端，电脑做 Server 端；

② Client IP：192.168.1.3；

③ Server IP：192.168.1.90。

2、LC 为 Client 编程

1）添加库

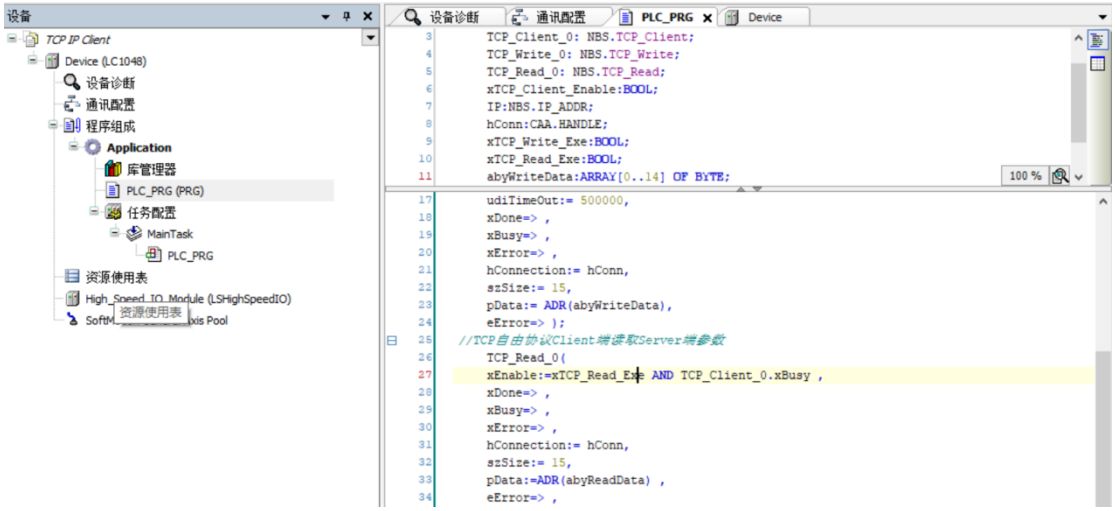
① 双击【库管理器】;

② 左键单击【添加库】;

③ 在添加库界面的文本框输入所需的库“Network”即可搜索完成；

④ 左键单击选择需要的库【Network】后左键单击右下角【确定】按钮即可 添加完成；

⑤ 【Netwok,3.5.7.0】即为添加的库。



2）创建变量 创建下图所示变量

```
PROGRAM PLC_PRG
VAR
    TCP_Client_0: NBS.TCP_Client;
    TCP_Write_0: NBS.TCP_Write;
    TCP_Read_0: NBS.TCP_Read;
    xTCP_Client_Enable:BOOL;
    IP:NBS.IP_ADDR;
    hConn:CAA.HANDLE;
    xTCP_Write_Exe:BOOL;
    xTCP_Read_Exe:BOOL;
    abyWriteData:ARRAY[0..14] OF BYTE;
    abyReadData:ARRAY[0..14] OF BYTE;
END_VAR
```

3）设置 Client 端参数及获取通信句柄

① 打开 NetAssist 网络调试助手软件，【协议类型】为 TCP Server，【本地 IP 地址】为 192.168.1.90，【本地端口】为 8080，设置好参数后左键单击【连接】



② 调用 TCP_Client 指令，ipAddr 为 NBS.IP_Client 结构体变量，需要对结构 体变量成员 sAddr 进行赋值（Server 端的 IP 地址，为字符串），即 IP.sAddr:='192.168.1.90';（Server 端 IP 地址为 192.168.1.90）

③ hConnection 为输出句柄，CAA.HANDLE 结构体变量；

④ TCP_Client 的 xEnable 为 TRUE 后即可连接成功。

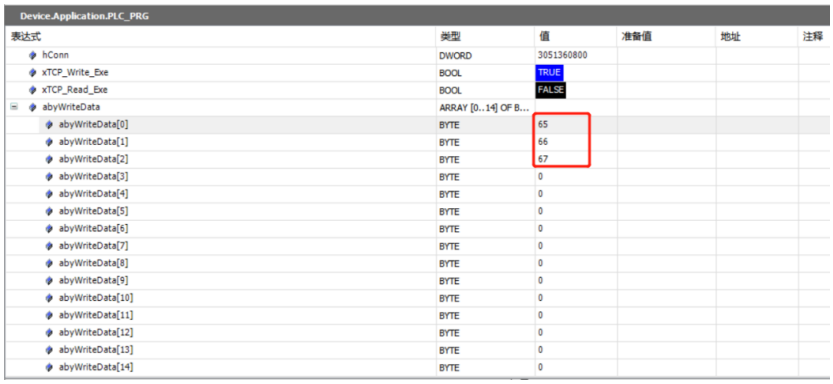


4）Client 端（LC 端）写数值，Server 端接收

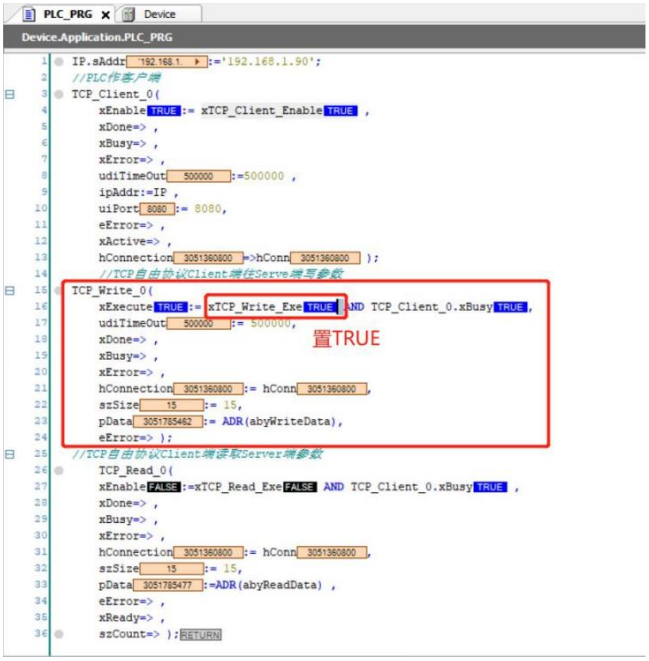
① 确保 TCP_Client 端与 TCP_Server 端连接成功；

② 调用 TCP_Write 指令。hConnection 为 TCP_Client 的输出句柄；pData 为 指针变量，是写入数据变量所指向的地址；szSize 为写入数据变量的数据长度；

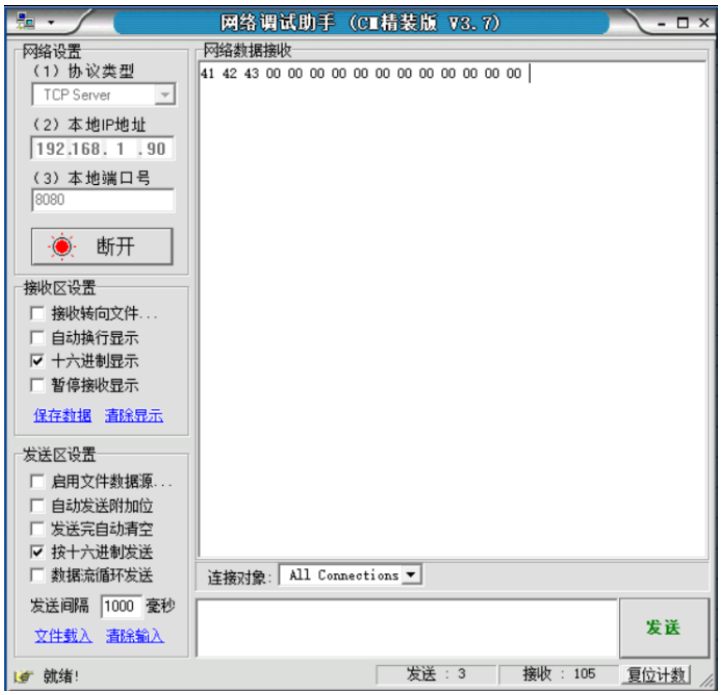
③ 将 abyWriteData[0]、abyWriteData[1]及 abyWriteData[2]赋值十进制数值 65、66 及 67；



④ 触发“xExecute”为 TRUE



⑤ Server 端可收到 16 进制数值 41、42 及 43（对应的十进制数值为 65、66 及 67）



5）Client 端（LC）读取 Server 端发送的数值

① 确保 TCP_Client 端的 xEnable 至 TRUE；

② 调用 TCP_Read 指令。hConnection 为 TCP_Client 的输出句柄；pData 为 指针变量，是读取数据变量所指向的地址；szSize 为读取数据变量的数据长度；

PLC_PRG | Device

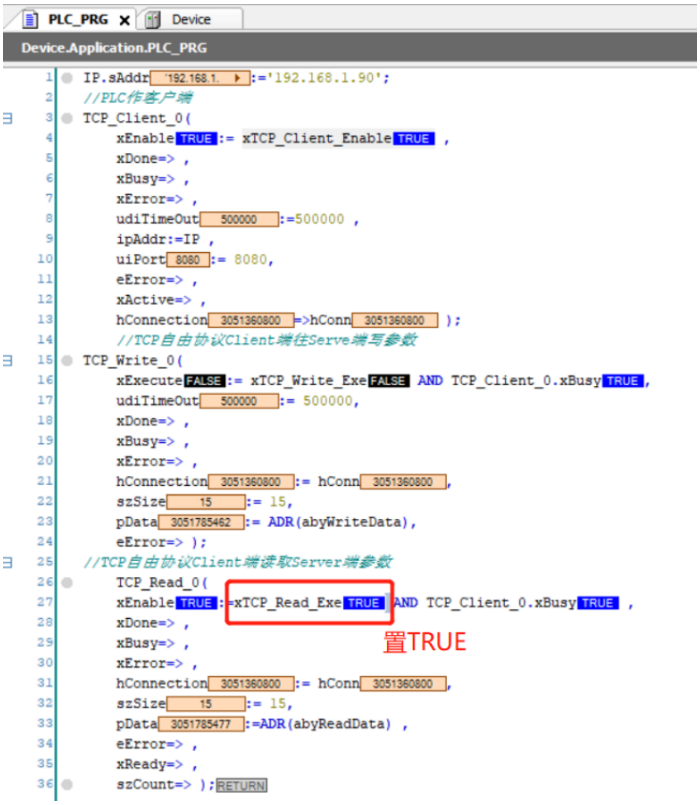
Device.Application.PLC_PRG

```
IP.sAddr[192.168.1.]:='192.168.1.90';
//PLC作客户端
TCP_Client_0(
    xEnable[TRUE]:= xTCP_Client_Enable[TRUE] ,
    xDone=> ,
    xBusy=> ,
    xError=> ,
    udiTimeOut[500000]:=500000 ,
    ipAddr:=IP ,
    uiPort[8080]:= 8080,
    eError=> ,
    xActive=> ,
    hConnection[3051360800]=>hConn[3051360800] );
    //TCP自由协议Client端往Serve端写参数
TCP_Write_0(
    xExecute[FALSE]:= xTCP_Write_Exe[FALSE] AND TCP_Client_0.xBusy[TRUE],
    udiTimeOut[500000]:= 500000,
    xDone=> ,
    xBusy=> ,
    xError=> ,
    hConnection[3051360800]:= hConn[3051360800],
    szSize[15]:= 15,
    pData[3051785462]:= ADR(abyWriteData),
    eError=> );
//TCP自由协议Client端读取Server端参数
    TCP_Read_0(
    xEnable[FALSE]:=xTCP_Read_Exe[FALSE] AND TCP_Client_0.xBusy[TRUE] ,
    xDone=> ,
    xBusy=> ,
    xError=> ,
    hConnection[3051360800]:= hConn[3051360800],
    szSize[15]:= 15,
    pData[3051785477]:=ADR(abyReadData) ,
    eError=> ,
    xReady=> ,
    szCount=> );RETURN
```

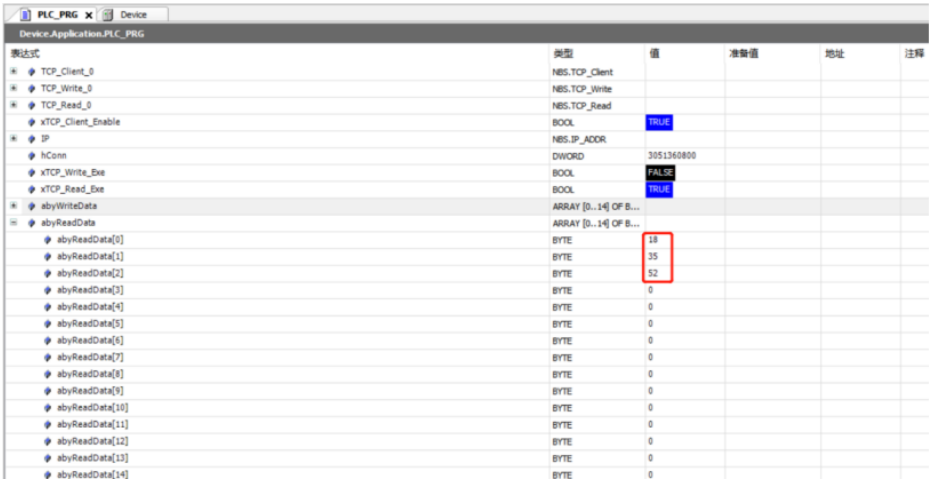
③ 使用 NetAssist 网络调试助手作为 Server 端发送 16 进制数值 12 23 34



④ 触发 TCP_Read 的 xExecute；



⑤ abyReadData[0]、abyReadData[1]及 abyReadData[2]的十进制数值分别为 18、35 及 52（16 进制为 16#12、16#23 及 16#34），则读取成功。

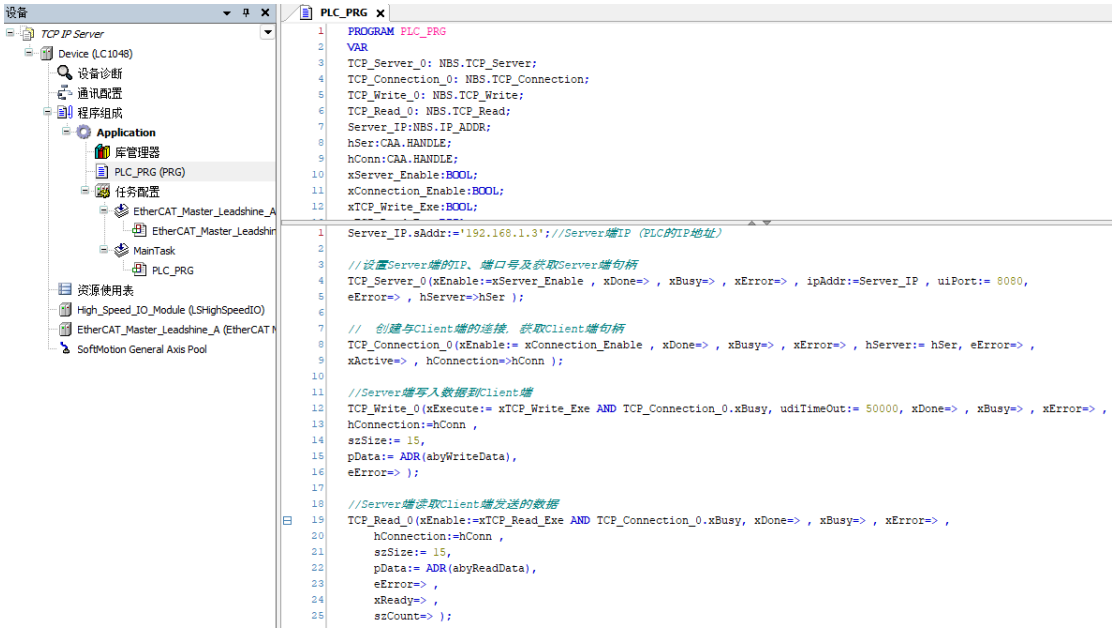


LC 为 Server 端

3、LC 为 Server 编程

1）添加库

添加的库 Netwok,3.5.7.0，与 LC 为 Client 端添加的库相同。



2）创建变量 创建下图所示变量

```
PROGRAM PLC_PRG
VAR
TCP_Server_0: NBS.TCP_Server;
TCP_Connection_0: NBS.TCP_Connection;
TCP_Write_0: NBS.TCP_Write;
TCP_Read_0: NBS.TCP_Read;
Server_IP:NBS.IP_ADDR;
hSer:CAA.HANDLE;
hConn:CAA.HANDLE;
xServer_Enable:BOOL;
xConnection_Enable:BOOL;
xTCP_Write_Exe:BOOL;
xTCP_Read_Exe:BOOL;
abyWriteData:ARRAY[0..14] OF BYTE;
abyReadData:ARRAY[0..14] OF BYTE;
END_VAR
```

3）设置 Server 端参数及获取通信句柄

① 调用 TCP_Server 指令；

② 设置相关参数，ipAddr 为 NBS.IP_Server 结构体变量，需要对结构体变量 成员 sAddr 进行赋值（Server 端的 IP 地址，为字符串），即 IP.sAddr:='192.168.1.3'; （Server 端 IP 地址为 192.168.1.3）；

③ uiPort 为 Server 端的端口号，UINT 数据类型，端口号为 8080（自由定义， 但不能和其余通信端口号重复）；

④ hServer 为通信句柄，CAA.HANDLE 结构体变量；

⑤ TCP_Server 的 xEnable 至 TRUE。

```
Server_IP.sAddr[192.168.1.]:='192.168.1.3';//Server端IP（PLC的IP地址）

//设置Server端的IP、端口号及获取Server端句柄
TCP_Server_0(xEnable[TRUE]:=xServer_Enable[TRUE], xDone=> , xBusy=> , xError=> , ipAddr:=Server_IP ,
uiPort[8080]:= 8080, eError=> , hServer[3042693752]=>hSer[3042693752] );

//  创建与Client端的连接，获取Client端句柄
TCP_Connection_0(xEnable[FALSE]:= xConnection_Enable[FALSE] , xDone=> , xBusy=> , xError=> ,
hServer[3042693752]:= hSer[3042693752], eError=> , xActive=> , hConnection[0]=>hConn[0] );

//Server端写入数据到Client端
TCP_Write_0(xExecute[FALSE]:= xTCP_Write_Exe[FALSE] AND TCP_Connection_0.xBusy[FALSE], udiTimeOut[50000]:= 50000,
xDone=> , xBusy=> , xError=> , hConnection[0]:=hConn[0] ,
szSize[15]:= 15,
pData[3042695000]:= ADR(abyWriteData),
eError=> );

//Server端读取Client端发送的数据
TCP_Read_0(xEnable[FALSE]:=xTCP_Read_Exe[FALSE] AND TCP_Connection_0.xBusy[FALSE], xDone=> , xBusy=> , xError=> ,
    hConnection[0]:=hConn[0] ,
    szSize[15]:= 15,
    pData[3042695015]:= ADR(abyReadData),
    eError=> ,
    xReady=> ,
    szCount=> );

RETURN
```

置TRUE

4）建立与 Client 端连接

① 调用 TCP_Connection;

② hServer 为 Server 端通信句柄，使用 TCP_Server 获取的通信句柄即可；

③ hConnection 为 Client 端输出句柄，CAA.HANDLE 结构体变量；

④ TCP_Connection 的 xEnable 至 TRUE。

```
Server_IP.sAddr[192.168.1.]:='192.168.1.3';//Server端IP（PLC的IP地址）

//设置Server端的IP、端口号及获取Server端句柄
TCP_Server_0(xEnable[TRUE]:=xServer_Enable[TRUE], xDone=> , xBusy=> , xError=> , ipAddr:=Server_IP ,
uiPort[8080]:= 8080, eError=> , hServer[3042693752]=>hSer[3042693752] );

//  创建与Client端的连接，获取Client端句柄
TCP_Connection_0(xEnable[TRUE]:= xConnection_Enable[TRUE] , xDone=> , xBusy=> , xError=> ,
hServer[3042693752]:= hSer[3042693752], eError=> , xActive=> , hConnection[0]=>hConn[0] );

//Server端写入数据到Client端
TCP_Write_0(xExecute[FALSE]:= xTCP_Write_Exe[FALSE] AND TCP_Connection_0.xBusy[TRUE], udiTimeOut[50000]:= 50000,
xDone=> , xBusy=> , xError=> , hConnection[0]:=hConn[0] ,
szSize[15]:= 15,
pData[3042695000]:= ADR(abyWriteData),
eError=> );

//Server端读取Client端发送的数据
TCP_Read_0(xEnable[FALSE]:=xTCP_Read_Exe[FALSE] AND TCP_Connection_0.xBusy[TRUE], xDone=> , xBusy=> , xError=> ,
    hConnection[0]:=hConn[0] ,
    szSize[15]:= 15,
    pData[3042695015]:= ADR(abyReadData),
    eError=> ,
    xReady=> ,
    szCount=> );

RETURN
```

5）Server 端 写数字，Client 端接收

① TCP_Server 及 TCP_Connection 的 xEnable 至 TRUE；

② 调用 TCP_Write 指令。hConnection 为 TCP_Client 的输出句柄；pData 为 指针变量，是写入数据变量所指向的地址；szSize 为写入数据变量的数据长度；

```
Server_IP.sAddr:='192.168.1.3';//Server端IP（PLC的IP地址）

//设置Server端的IP、端口号及获取Server端句柄
TCP_Server_0(xEnable:=xServer_Enable, xDone=> , xBusy=> , xError=> , ipAddr:=Server_IP ,
uiPort:= 8080, eError=> , hServer=>hSer );

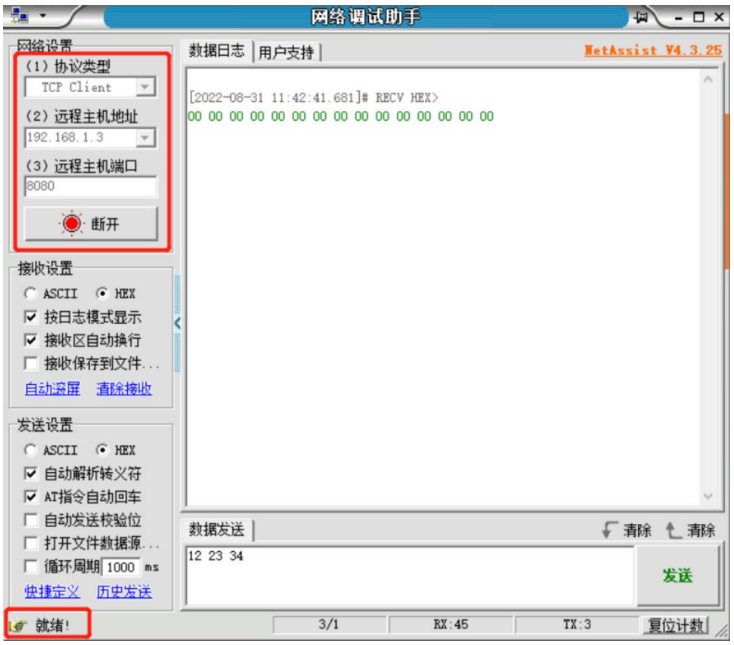
//  创建与Client端的连接，获取Client端句柄
TCP_Connection_0(xEnable:= xConnection_Enable , xDone=> , xBusy=> , xError=> ,
hServer:= hSer, eError=> , xActive=> , hConnection=>hConn );

//Server端写入数据到Client端
TCP_Write_0(xExecute:= xTCP_Write_Exe AND TCP_Connection_0.xBusy, udiTimeOut:= 50000,
xDone=> , xBusy=> , xError=> , hConnection:=hConn ,
szSize:= 15,
pData:= ADR(abyWriteData),
eError=> );

//Server端读取Client端发送的数据
TCP_Read_0(xEnable:=xTCP_Read_Exe AND TCP_Connection_0.xBusy, xDone=> , xBusy=> , xError=> ,
    hConnection:=hConn ,
    szSize:= 15,
    pData:= ADR(abyReadData),
    eError=> ,
    xReady=> ,
    szCount=> );

RETURN
```

③ 打开 NetAssist 网络调试助手软件，【协议类型】为 TCP Client，【本地 IP 地址】为 192.168.1.3，【本地端口】为 8080，设置好参数后左键单击【连接】，确保 TCP_Server 与 TCP_Client 端连接成功;



④ 将 byWriteData[0]、byWriteData[1]及 byWriteData[2]赋值十进制数值 55、 56 及 57；

表达式	类型	值	准备值	地址
TCP_Server_0	NBS.TCP_Server			
TCP_Connection_0	NBS.TCP_Connection			
TCP_Write_0	NBS.TCP_Write			
TCP_Read_0	NBS.TCP_Read			
Server_IP	NBS.IP_ADDR			
hSer	DWORD	3042693752		
hConn	DWORD	3042694440		
xServer_Enable	BOOL	TRUE		
xConnection_Enable	BOOL	TRUE		
xTCP_Write_Exe	BOOL	FALSE		
xTCP_Read_Exe	BOOL	FALSE		
abyWriteData	ARRAY [0..14] OF B...			
abyWriteData[0]	BYTE	55		
abyWriteData[1]	BYTE	56		
abyWriteData[2]	BYTE	57		
abyWriteData[3]	BYTE	0		
abyWriteData[4]	BYTE	0		
abyWriteData[5]	BYTE	0		
abyWriteData[6]	BYTE	0		
abyWriteData[7]	BYTE	0		
abyWriteData[8]	BYTE	0		
abyWriteData[9]	BYTE	0		
abyWriteData[10]	BYTE	0		
abyWriteData[11]	BYTE	0		
abyWriteData[12]	BYTE	0		
abyWriteData[13]	BYTE	0		
abyWriteData[14]	BYTE	0		
abyReadData	ARRAY [0..14] OF B...			

⑤ 触发 xExecute

```
Server_IP.sAddr := '192.168.1.3';//Server端IP（PLC的IP地址）

//设置Server端的IP、端口号及获取Server端句柄
TCP_Server_0(xEnable:=xServer_Enable , xDone=> , xBusy=> , xError=> , ipAddr:=Server_IP ,
uiPort:= 8080, eError=> , hServer=>hSer );

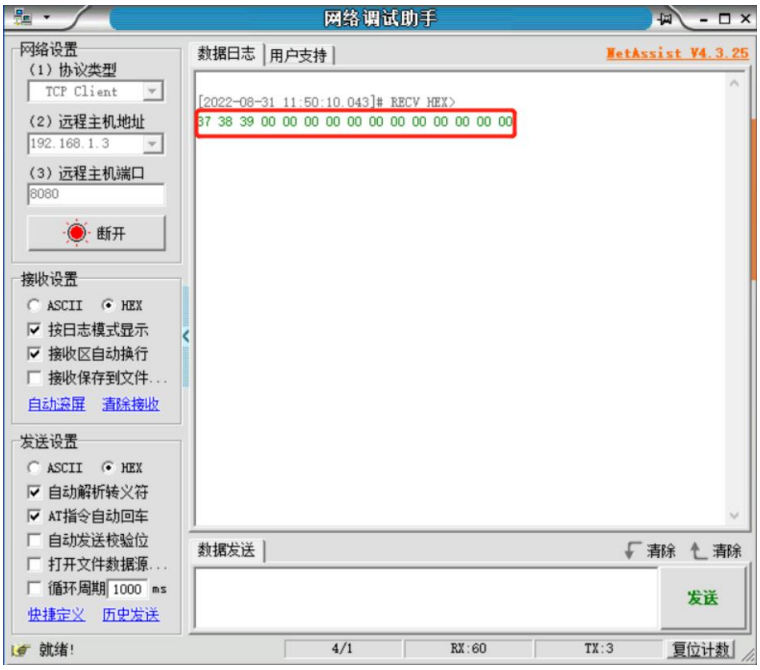
//  创建与Client端的连接，获取Client端句柄
TCP_Connection_0(xEnable:= xConnection_Enable , xDone=> , xBusy=> , xError=> ,
hServer:= hSer, eError=> , xActive=> , hConnection=>hConn );

//Server端写入数据到Client端
TCP_Write_0(xExecute:= xTCP_Write_Exe AND TCP_Connection_0.xBusy, udiTimeOut:= 50000,
xDone=> , xBusy=> , xError=> , hConnection:=hConn ,
szSize:= 15,
pData:= ADR(abyWriteData),
eError=> );

//Server端读取Client端发送的数据
TCP_Read_0(xEnable:=xTCP_Read_Exe AND TCP_Connection_0.xBusy, xDone=> , xBusy=> , xError=> ,
    hConnection:=hConn ,
    szSize:= 15,
    pData:= ADR(abyReadData),
    eError=> ,
    xReady=> ,
    szCount=> );

RETURN
```

⑥ Server 端可收到 16 进制数值 37、38 及 39（为十进制数值 55、56 及 57）



6）Server 端 读取 Client 端发送的数值

① TCP_Server 及 TCP_Connection 的 xEnable 设置为 TRUE；

② 调用 TCP_Read 指令。hConnection 为 TCP_Client 的输出句柄；pData 为 指针变量，是读取数据变量所指向的地址；szSize 为读取数据变量的数据长度；

```
Server_IP.sAddr:='192.168.1.3';//Server端IP（PLC的IP地址）

//设置Server端的IP、端口号及获取Server端句柄
TCP_Server_0(xEnable:=xServer_Enable , xDone=> , xBusy=> , xError=> , ipAddr:=Server_IP ,
uiPort:= 8080, eError=> , hServer=>hSer );

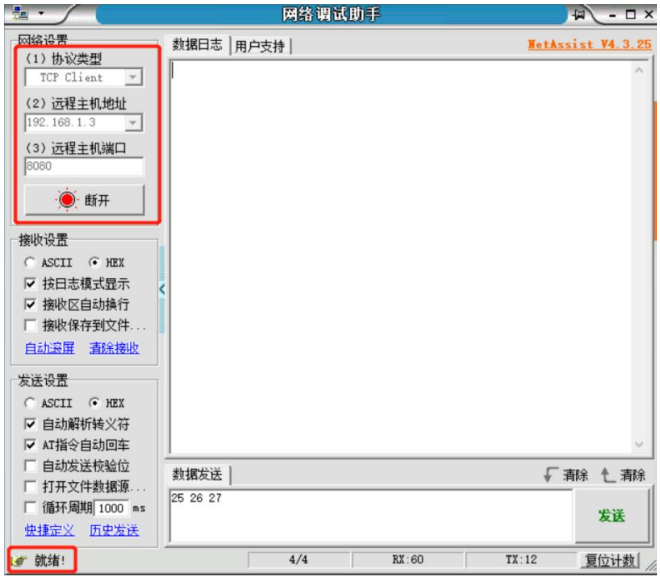
//  创建与Client端的连接，获取Client端句柄
TCP_Connection_0(xEnable:= xConnection_Enable , xDone=> , xBusy=> , xError=> ,
hServer:= hSer, eError=> , xActive=> , hConnection=>hConn );

//Server端写入数据到Client端
TCP_Write_0(xExecute:= xTCP_Write_Exe AND TCP_Connection_0.xBusy, udiTimeOut:= 50000,
xDone=> , xBusy=> , xError=> , hConnection:=hConn ,
szSize:= 15,
pData:= ADR(abyWriteData),
eError=> );

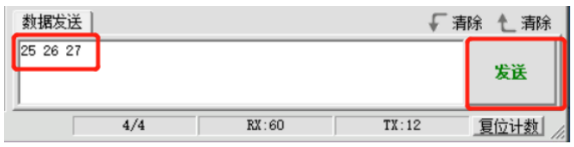
//Server端读取Client端发送的数据
TCP_Read_0(xEnable:=xTCP_Read_Exe AND TCP_Connection_0.xBusy, xDone=> , xBusy=> , xError=> ,
    hConnection:=hConn ,
    szSize:= 15,
    pData:= ADR(abyReadData),
    eError=> ,
    xReady=> ,
    szCount=> );

RETURN
```

③ 打开 NetAssist 网络调试助手软件，【协议类型】为 TCP Client，【本地 IP 地址】为 192.168.1.3，【本地端口】为 8080，设置好参数后左键单击【连接】， 确保 TCP_Server 与 TCP_Client 端连接成功



④ 使用 NetAssist 网络调试助手作为 Client 端发送 16 进制数值 25 26 27（十 进制为 37、38 及 39）；



⑤ 触发 TCP_Read 的 xExecute 设置为 TRUE；

```
Server_IP.sAddr:='192.168.1.3';//Server端IP（PLC的IP地址）

//设置Server端的IP、端口号及获取Server端句柄
TCP_Server_0(xEnable:=xServer_Enable , xDone=> , xBusy=> , xError=> , ipAddr:=Server_IP ,
uiPort:= 8080, eError=> , hServer=>hSer );

//  创建与Client端的连接，获取Client端句柄
TCP_Connection_0(xEnable:= xConnection_Enable , xDone=> , xBusy=> , xError=> ,
hServer:= hSer, eError=> , xActive=> , hConnection=>hConn );

//Server端写入数据到Client端
TCP_Write_0(xExecute:= xTCP_Write_Exe AND TCP_Connection_0.xBusy, udiTimeOut:= 50000,
xDone=> , xBusy=> , xError=> , hConnection:=hConn ,
szSize:= 15,
pData:= ADR(abyWriteData),
eError=> );
                                    置TRUE
//Server端读取Client端发送的数据
TCP_Read_0(xEnable:=xTCP_Read_Exe AND TCP_Connection_0.xBusy, xDone=> , xBusy=> , xError=> ,
    hConnection:=hConn ,
    szSize:= 15,
    pData:= ADR(abyReadData),
    eError=> ,
    xReady=> ,
    szCount=> );

RETURN
```

⑥ abyReadData[0]、abyReadData[1]及 abyReadData[2]分别的数值为 37、38 及 39，则读取成功

表达式	类型	值
TCP_Server_0	NBS.TCP_Server	
TCP_Connection_0	NBS.TCP_Connection	
TCP_Write_0	NBS.TCP_Write	
TCP_Read_0	NBS.TCP_Read	
Server_IP	NBS.IP_ADDR	
hSer	DWORD	3042693752
hConn	DWORD	3042694440
xServer_Enable	BOOL	TRUE
xConnection_Enable	BOOL	TRUE
xTCP_Write_Exe	BOOL	FALSE
xTCP_Read_Exe	BOOL	TRUE
abyWriteData	ARRAY [0..14] OF B...	
abyReadData	ARRAY [0..14] OF B...	
abyReadData[0]	BYTE	37
abyReadData[1]	BYTE	38
abyReadData[2]	BYTE	39
abyReadData[3]	BYTE	0
abyReadData[4]	BYTE	0
abyReadData[5]	BYTE	0
abyReadData[6]	BYTE	0
abyReadData[7]	BYTE	0
abyReadData[8]	BYTE	0
abyReadData[9]	BYTE	0
abyReadData[10]	BYTE	0
abyReadData[11]	BYTE	0
abyReadData[12]	BYTE	0
abyReadData[13]	BYTE	0
abyReadData[14]	BYTE	0