凌臣科技 LINGCHEN

资料编码：2304LC1500-2

LC1500系列
PLC快速应用手册

苏州市凌臣采集计算机技术有限公司
www.szpcbase.com

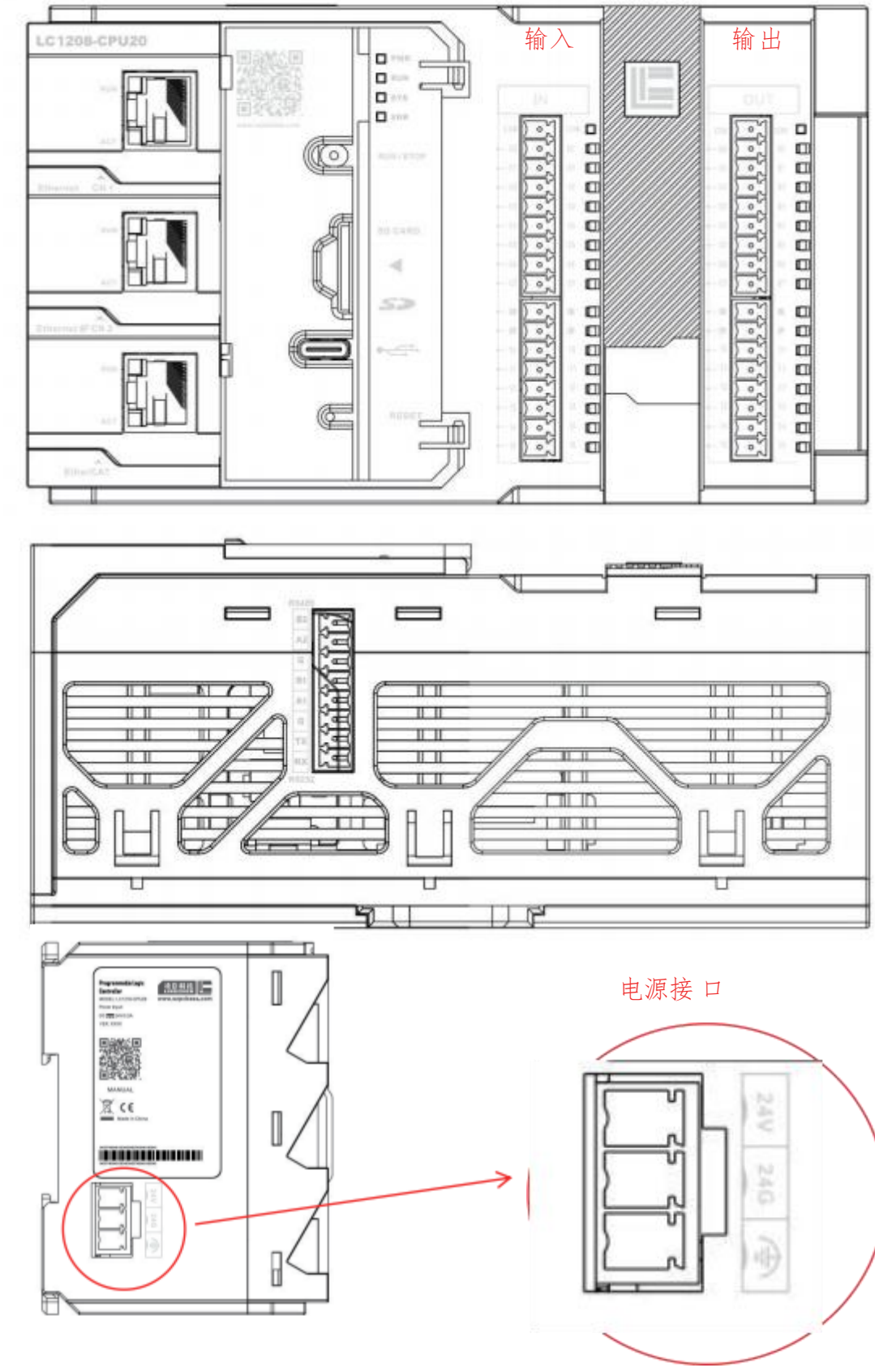
公司简介

苏州市凌臣采集计算机有限公司成立于2006年，是一家本着与客户共赢、为客户创造方案价值的经营理念的企业。为客户提供各种测试测量、运动控制、机器视觉、机器人等自动化设备的核心零部件和系统解决方案。 凌臣科技于2017年开始创立凌臣采集LCT品牌，研发了包括工控机、PLC控制器、EtherCAT步进驱动器、

PLC/PCIe运动控制卡、远程IO模块、气动阀岛等产品，为我司的客户提供了更具性价比的方案解决。

目 录

硬件接口展示



参数简介

Memory	DDR3L 1GB
EMMC	8GB
USB	1个USB插口（TYPE-C）
TF	1个SD卡插口
NET	2个EtherNET网口，1个EtherCAT网口
COM	2路RS485，1路RS232
INPUT	GPIO*16 输入高低电平检测
OUTPUT	GPIO*16 开漏控制输出
RESET	1个复位按键
POWER	24V 1A

指示灯说明

	常亮	常灭	闪烁	颜色
PWR	有供电	无供电		绿灯
RUN	run	stop		绿灯
SYS	启动中/死机	启动中/死机	PLC进程运行中	绿灯
ERR	运行错误	无错	总线错误	红灯

1. 设备存储库的使用

1.1 打开CODESYS V3.5 SP18软件，在界面最上方的菜单栏找到“工具”，点击“工具—设备存储库”；

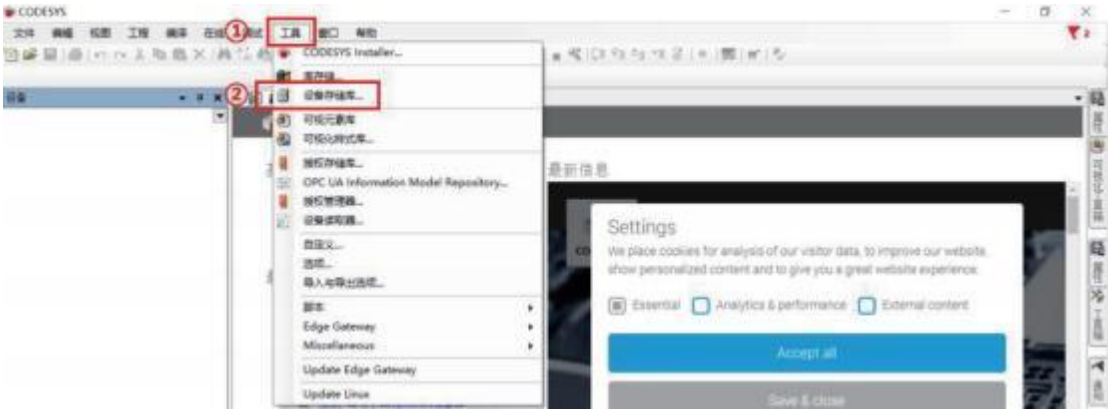
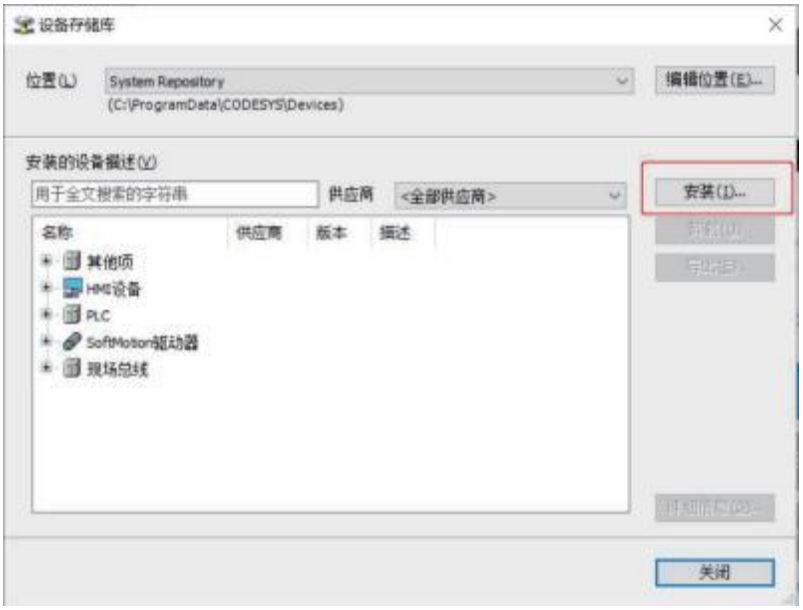


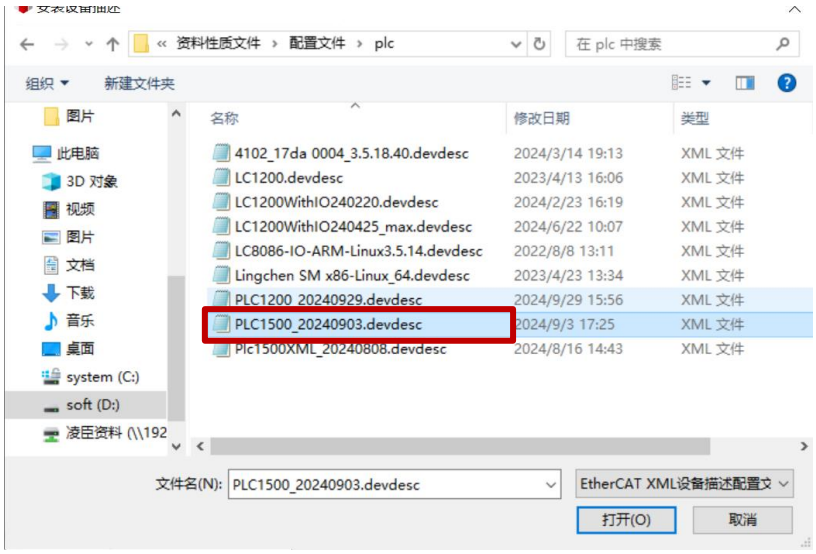
图_1.1

1.2 点击“安装”；



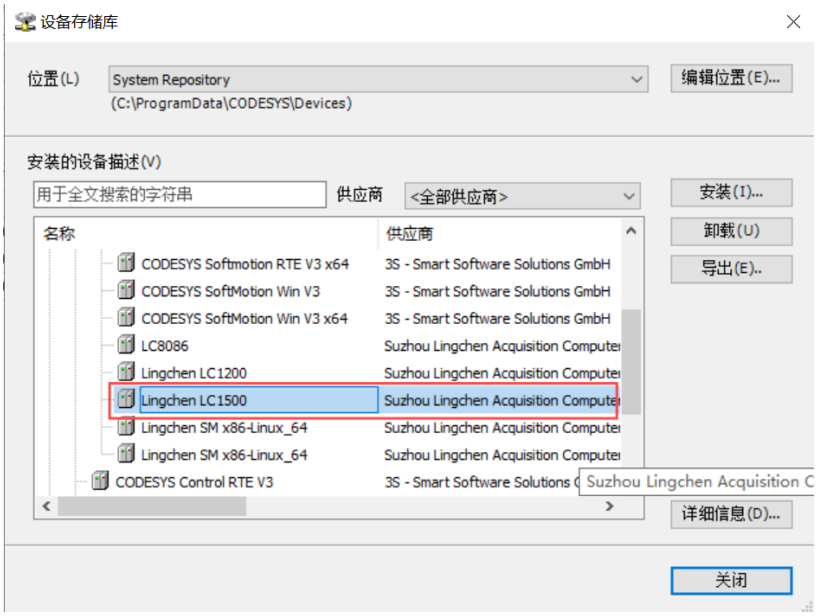
1.2

1.3 找到设备描述文件（*.xml）所存放的位置，选择相应设备描述文件，点击“确定”（或者直接双击相应设备描述文件）；



1.3

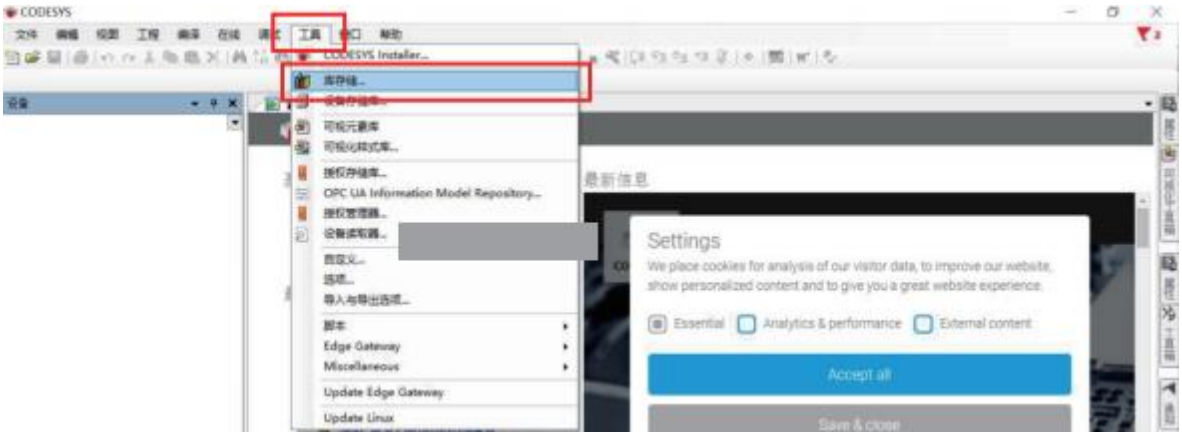
1.4 安装成功，点击“关闭”；



1.4

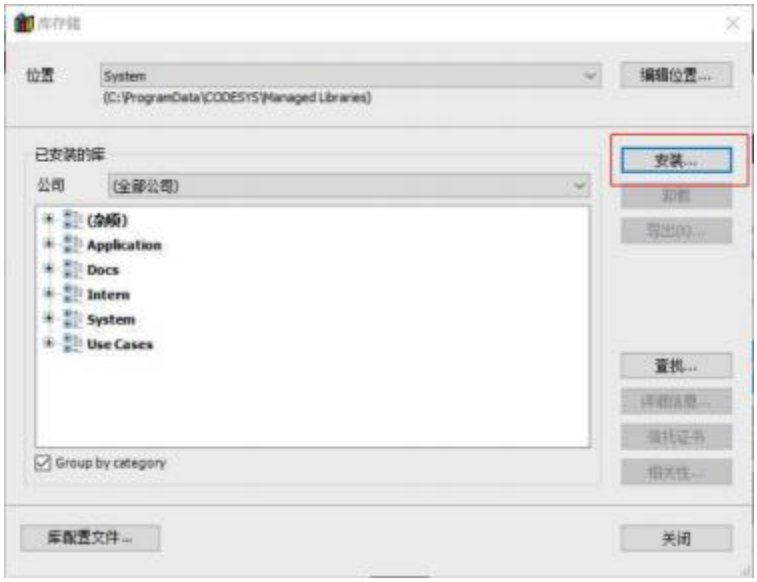
2. 库文件的使用

2.1 同样在菜单栏中找到“工具”，点击“工具—库存储”；



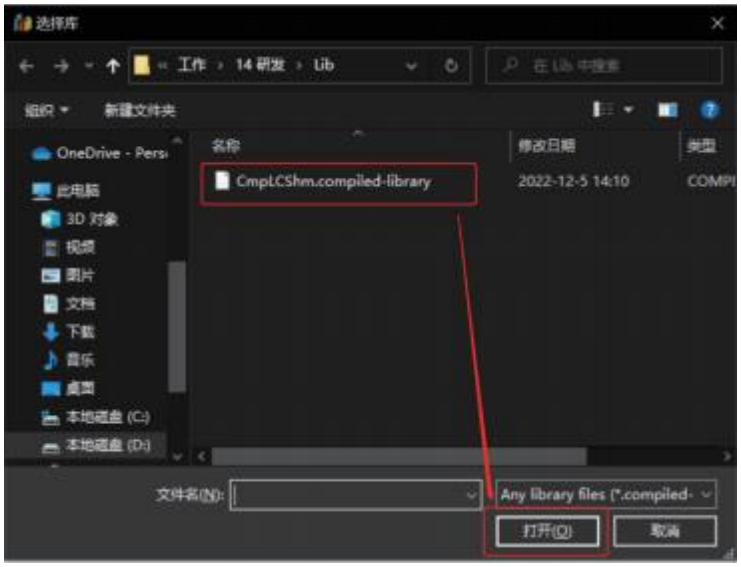
2.1

2.2 点击“安装”；



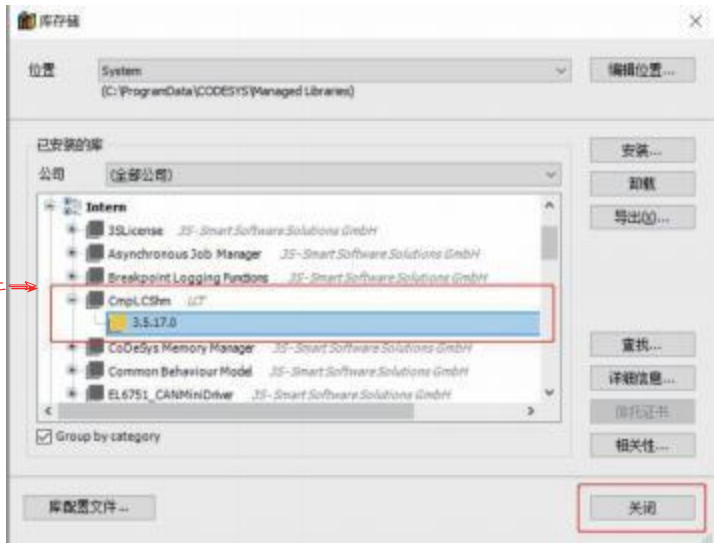
2.2

2.3 找到存放库文件的路径，选择要添加到存储的库文件，点击“打开”（或者直接双击要添加到存储的库文件）；



2.3

2.4 添加成功，点击“关闭”。



添加成功的库文件

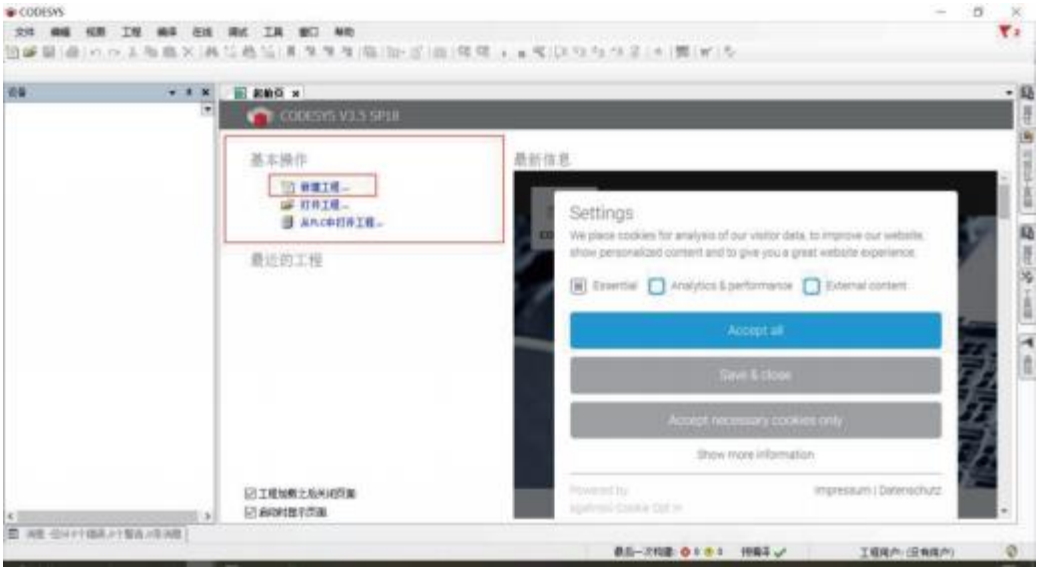
2.4

2.5 添加完成后，还需要安装到工程文件中，才能调用（详情见“3. 建立一个工程并下载调试”）。

3. 建立一个工程并下载调试

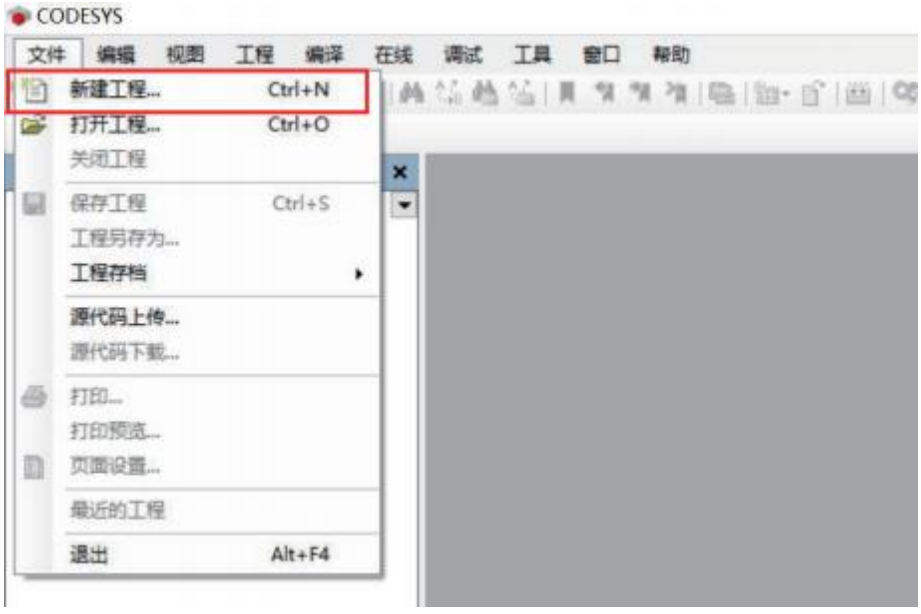
3.1 创建一个新的PLC程序

方法1：在“起始页”的“基本操作”区域，点击“新建工程...”；



3.1

方法2：若没有“起始页”，可以在菜单栏点击“文件—新建工程...”；



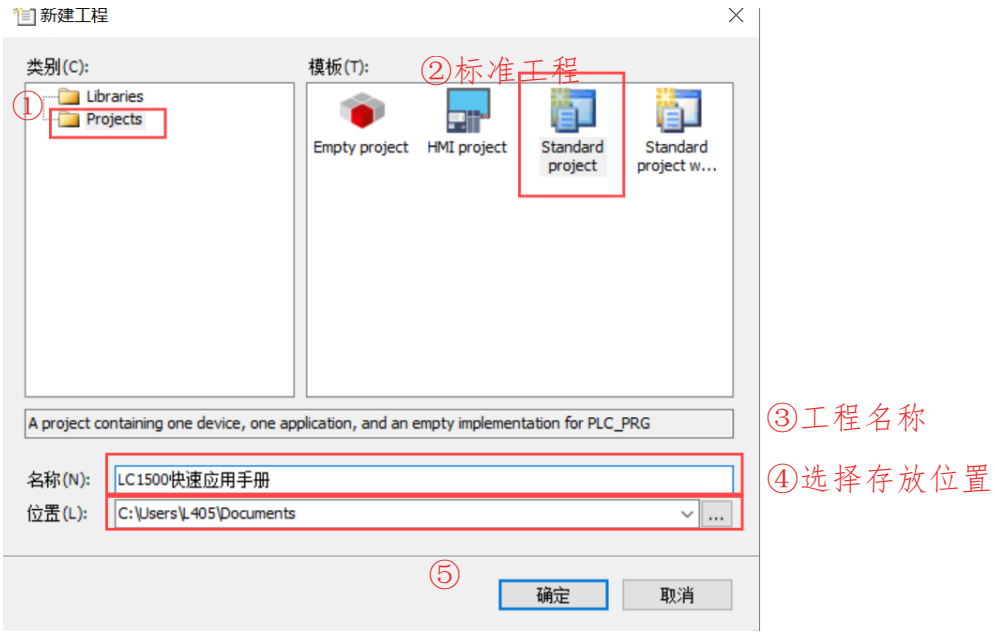
3.2

【注1：若想显示“起始页”，可以点击菜单栏“视图—起始页”。】

方法3：使用快捷键Ctrl＋N；

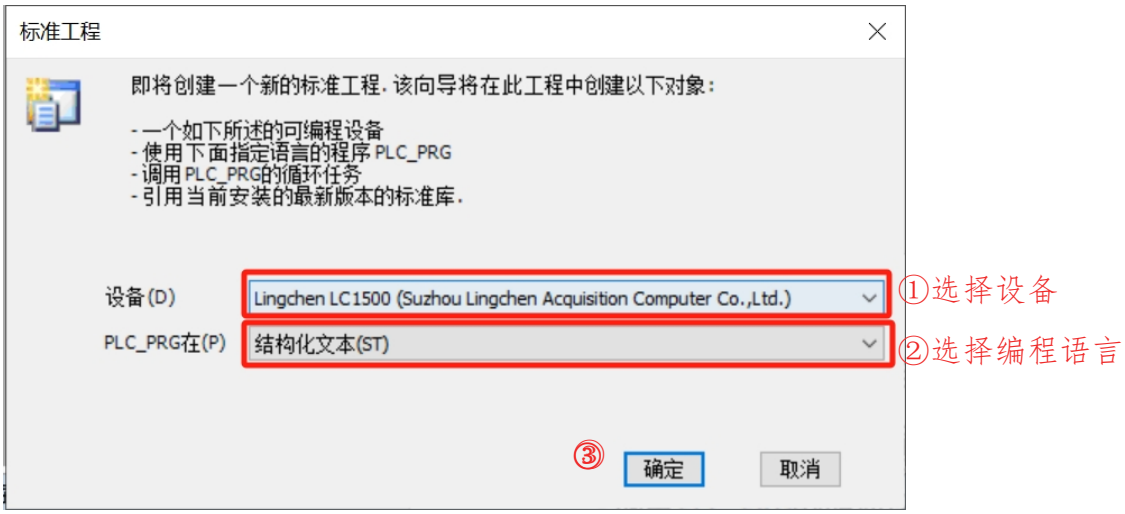
3.2 类别选择“工程”，模板选择“标准工程”，输入用户工程的名称，选择存放位置，点击“确定”；

LC1



3.3

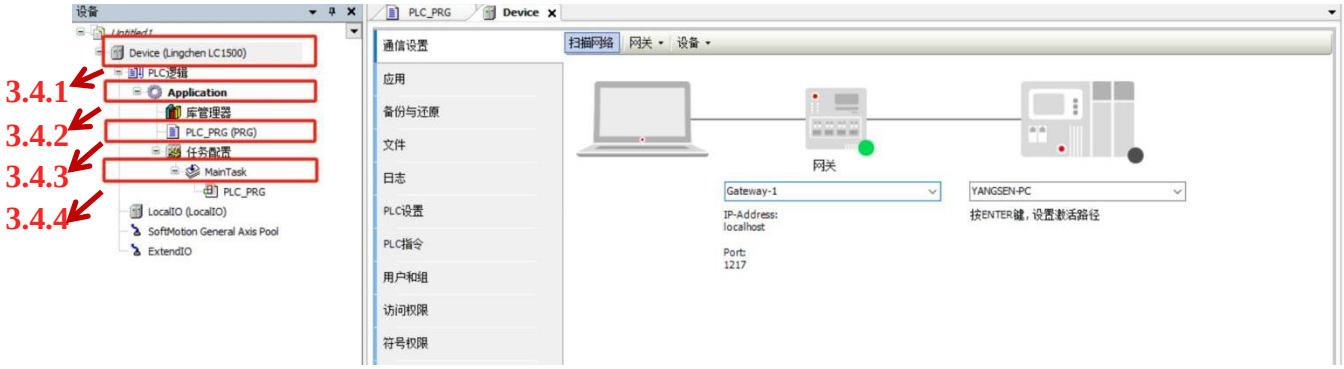
3.3 选择设备（Lingchen LC1216）与编程语言（后续新建POU可以更改编程语言）后，点击“确定”；



3.4

【注2：所有第一次使用的设备，都必须要在“设备存储库”中安装该设备（具体步骤参见本文档 1. 设备存储库 的使用）。】

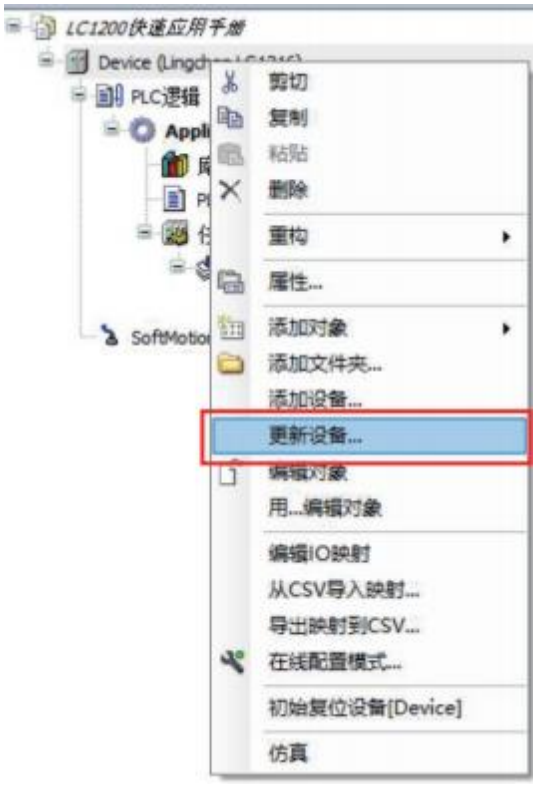
3.4 新建工程完成；



3.5

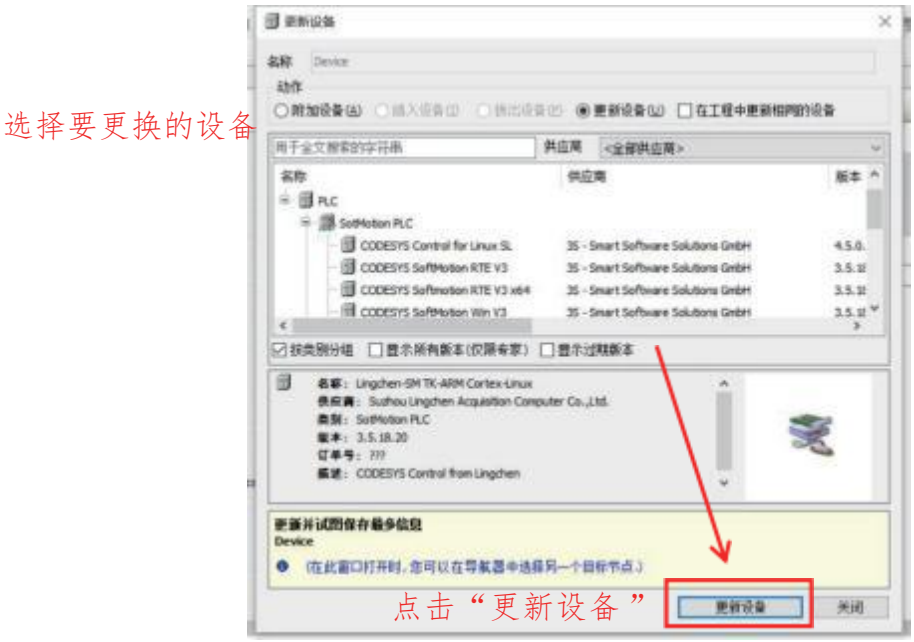
3.4.1 Device 设备。

此处是新建工程时所选择的设备，可以通过“右键（Device）—更新设备”来更换用户工程中的设备；



3.6

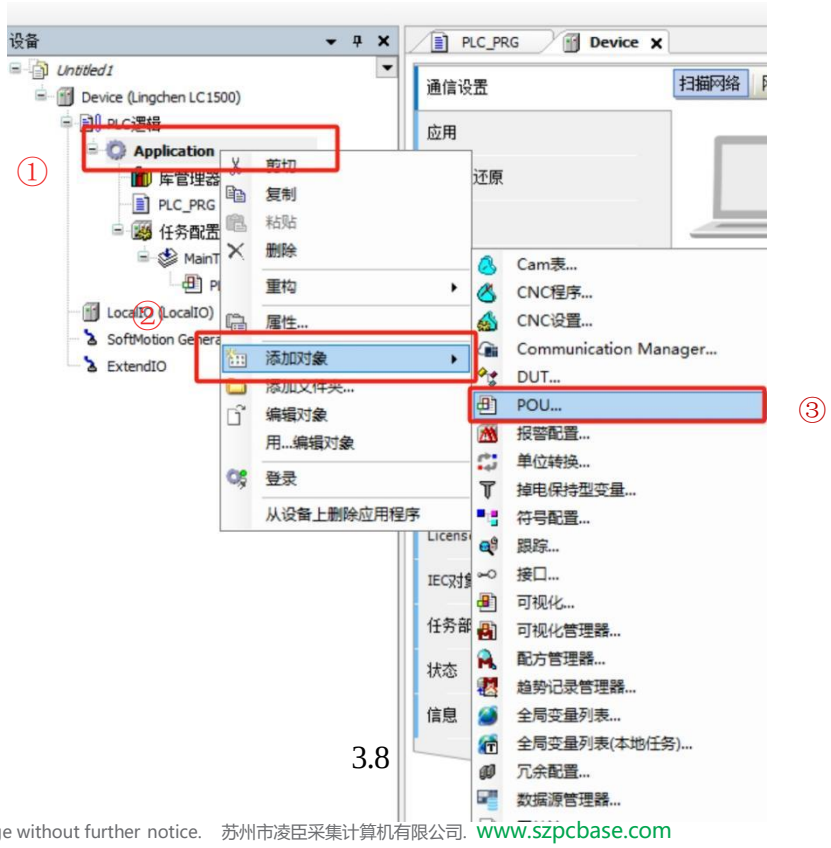
选择要更换的设备后，点击“更新设备”（或者直接双击要更换的设备）。



3.7

3.4.2 Application 应用。

在此处添加用户程序单元（POU）等。具体操作方法为：“右键Application -添加对象—POU”，输入POU名称，选择“程序”，选择该用户程序单元所要使用的语言。

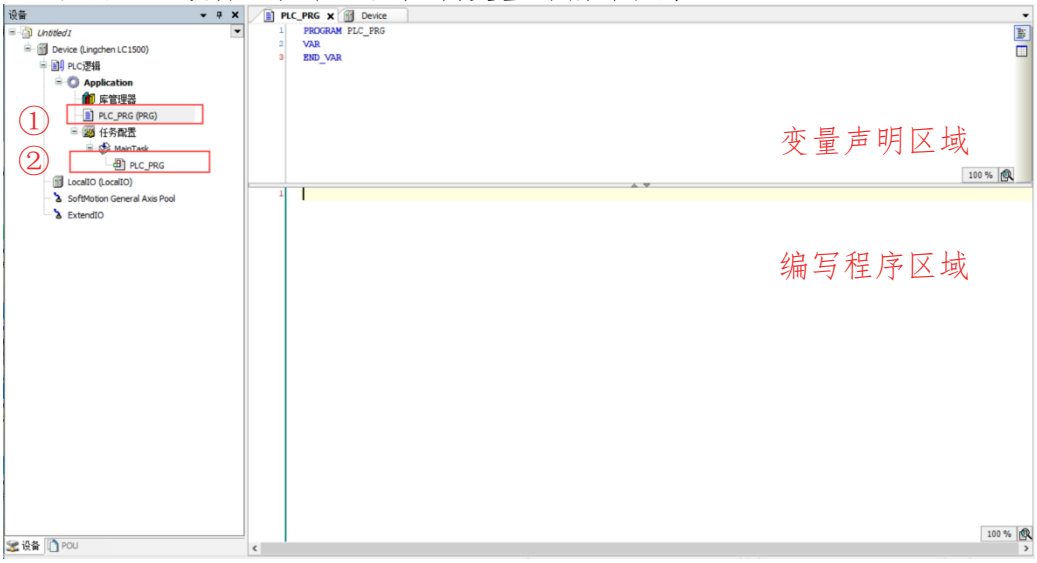


3.8

3.9

3.4.3 PLC_PRG(PRG) 用户程序。

双击进入该界面，在此处声明变量与编写程序。



3.10

【注3：要将写好的程序①拖入任务配置②下，方可编译调用。 】

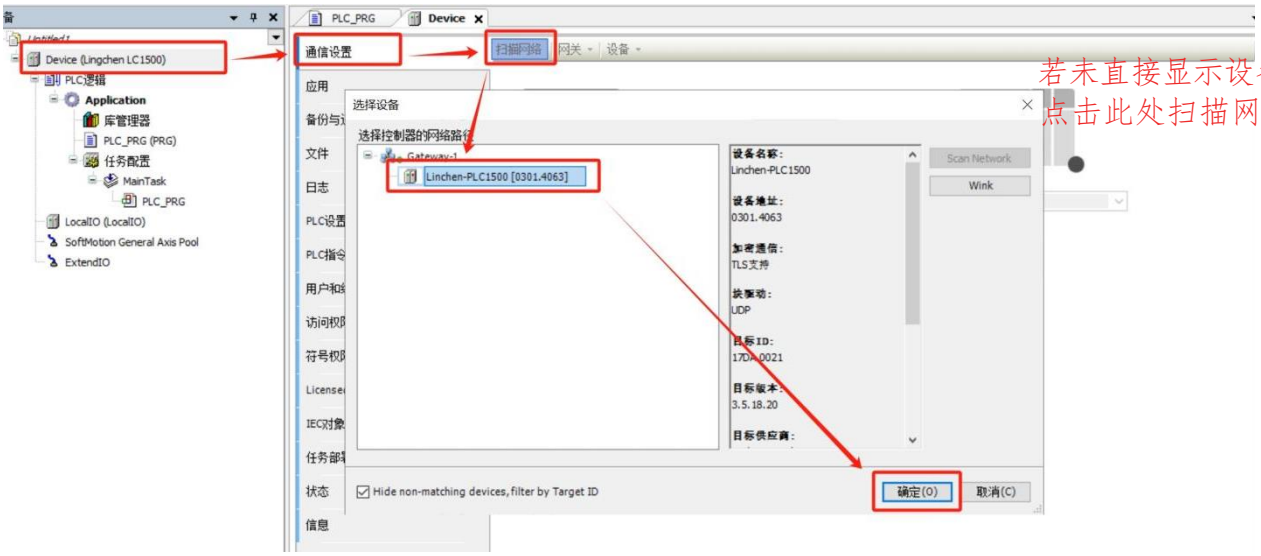
3.4.5 任务配置及程序调用选择。

3.5 编写程序；

3.6 系统配置及参数设定；

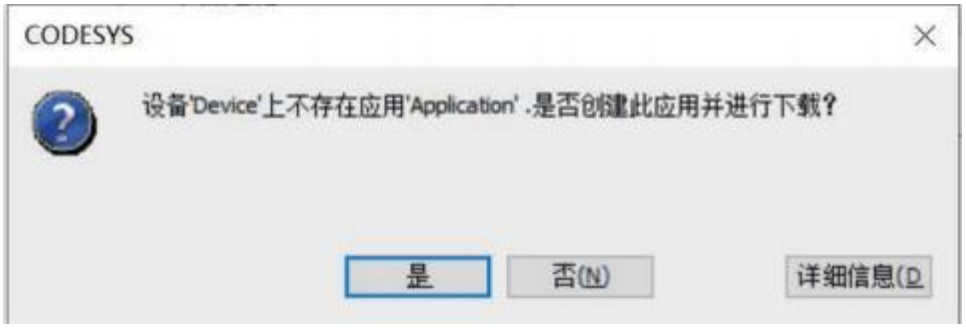
3.6.1连接设备

双击设备栏“Device”，在“通信设置”子页面内，点击“扫描网络”，选择设备“Lingchen-LC1500”后，点击“确定”。



3.11

初次连接，会有以下弹窗，点击“是”即可。



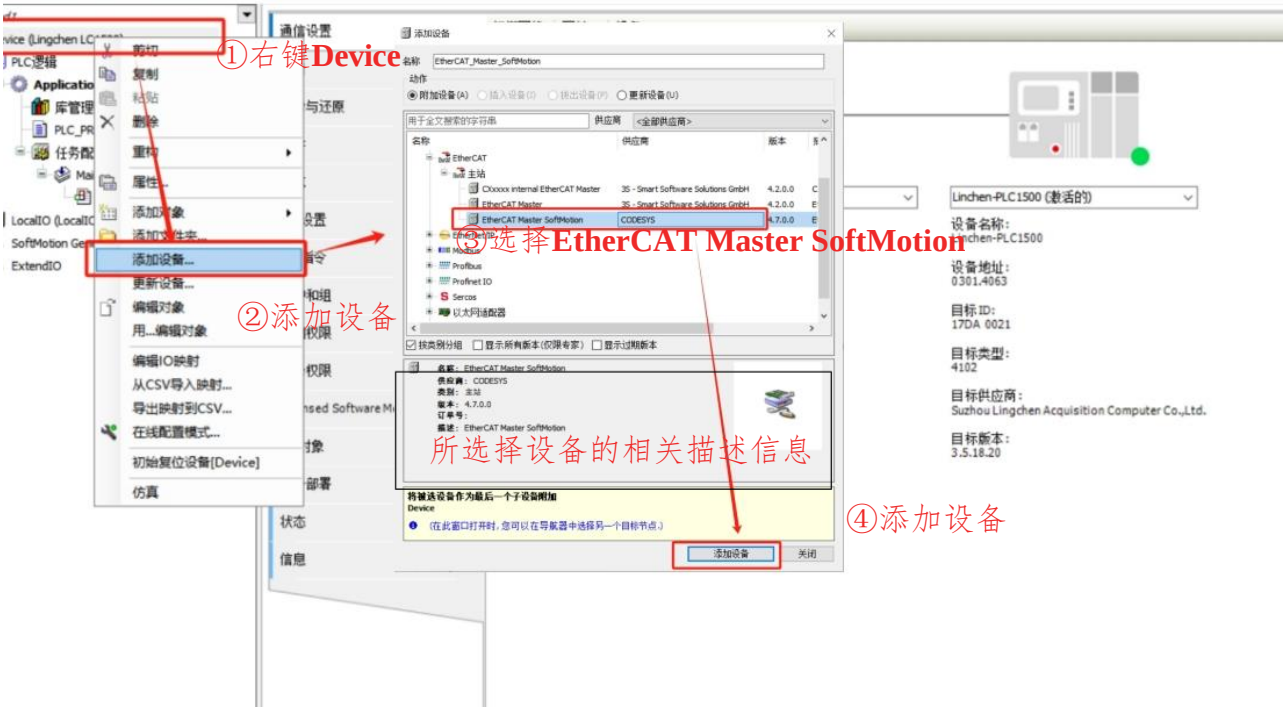
3.

连接成功，如3.12。

3.12

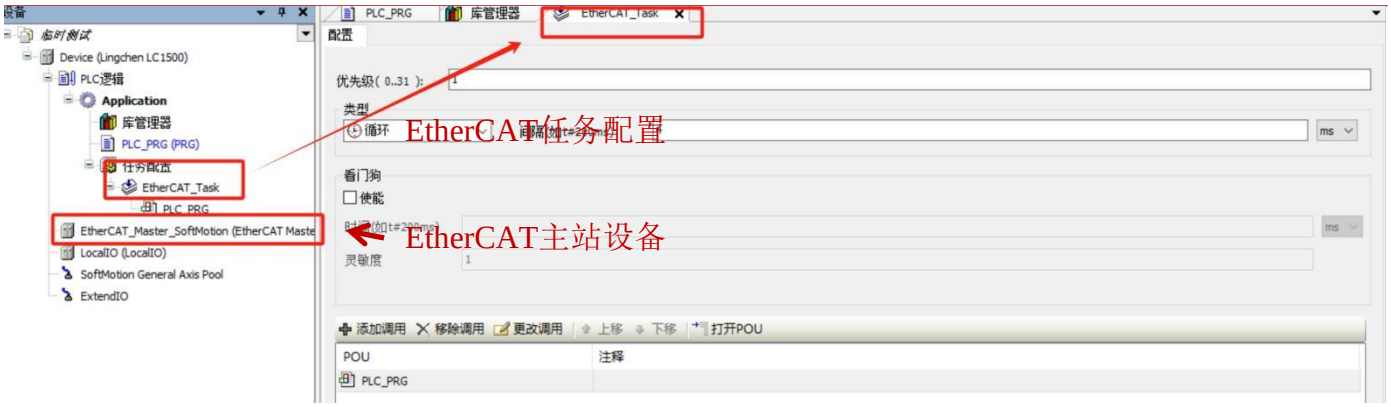
3.6.2 添加 EtherCAT_Master_SoftMotion

EtherCAT Master SoftMotion是带实时运动控制的EtherCAT主站模块。具体添加方法为：“右键Device－添加设备－现场总线－EtherCAT－主站－EtherCAT Master SoftMotion－添加设备”，添加 EtherCAT 主站。



3.13

添加后如3.14，同时系统会分配一个任务EtherCAT_Task，可配置EtherCAT任务相关参数。



3.14

在EtherCAT_Task配置界面，可以设定挂在该任务配置下的程序优先级、类型与扫描周期时间间隔等。

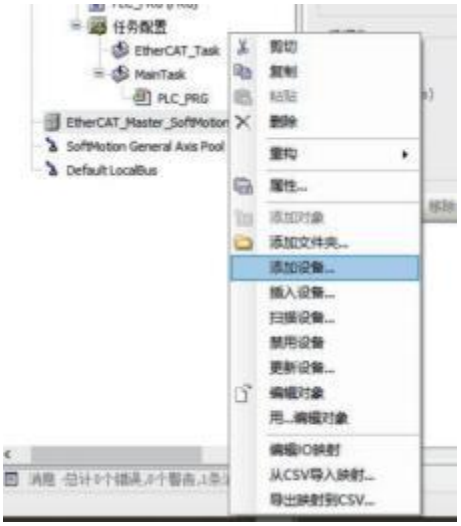
3.6.3 添加总线伺服

添加主站设备后，在该主站下方添加从站设备，此处添加设备为“LC_SDE_DC03A1（凌臣总线步进_单轴）”。添加前必须先安装设备描述文件（.XML），安装流程参考本文档1.设备存储库的使用。

具体添加方式有两种：

第一种：手动添加

未连接主站的离线状态下，通过“右键EtherCAT Master Softmotion —添加设备”，在“添加设备”弹窗中筛选对应供应商及设备型号“LC SDE-DC03A1”，选中该设备，点击“添加设备”（或直接双击要添加的设备）。

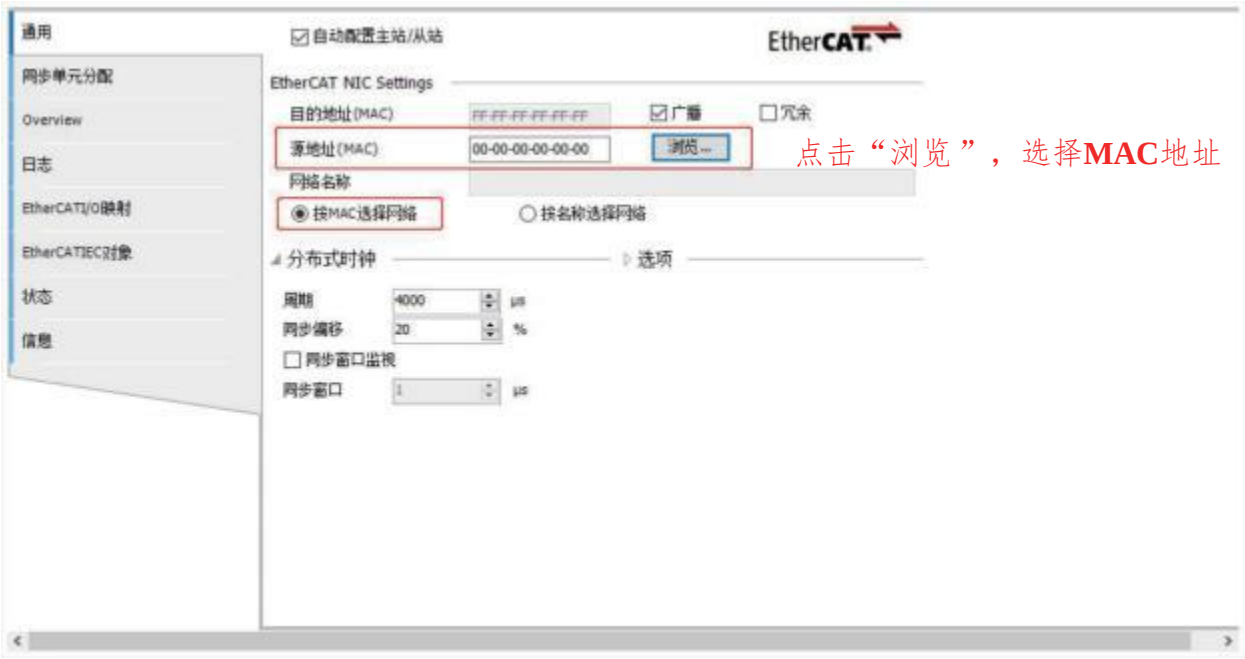


3.15

3.16

第二种：扫描添加

（1）“双击EtherCAT Master SoftMotion”打开主站设置界面，在“通用”子页面内，设置EtherCAT网口源地址，默认选项为“按MAC选择网络”时，可以通过点击“浏览”，选择EtherCAT网口MAC地址（3.17，3.18）；也可以更换为“按名称选择网络”选项，在“网络名称”输入框内，输入网口名称“eth1”（3.19）。


通用
同步单元分配
Overview
日志
EtherCAT I/O映射
EtherCAT IEC对象
状态
信息
自动配置主站/从站
EtherCAT
EtherCAT NIC Settings
目的地址(MAC)
源地址(MAC)
00-00-00-00-00-00
广播
冗余
浏览...
网络名称
按MAC选择网络
按名称选择网络
分布式时钟
选项
周期
4000
同步偏移
20
同步窗口监视
同步窗口
点击“浏览”，选择MAC地址

3.17


选择网络适配器
MAC地址
名称
描述
2C16BDD2A239
eth1
2C16BDD2A23A
eth2
2C16BDD2A238
eth0
选中eth1，点击“确定”
确定
中止

3.18

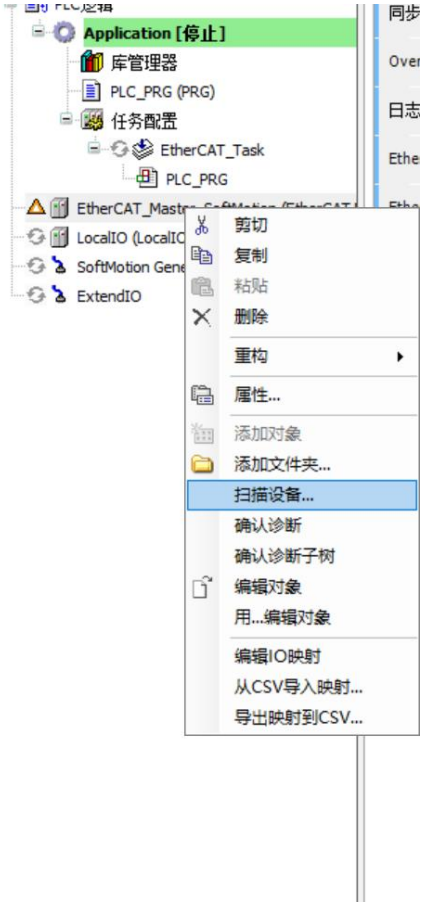

通用
同步单元分配
Overview
日志
EtherCAT I/O映射
EtherCAT IEC对象
状态
信息
自动配置主站/从站
EtherCAT
EtherCAT NIC Settings
目的地址(MAC)
源地址(MAC)
广播
冗余
浏览...
网络名称
eth1
按MAC选择网络
按名称选择网络
分布式时钟
选项
周期
4000
同步偏移
20
同步窗口监视
同步窗口
输入网络名称，按名称选择网络

3.19

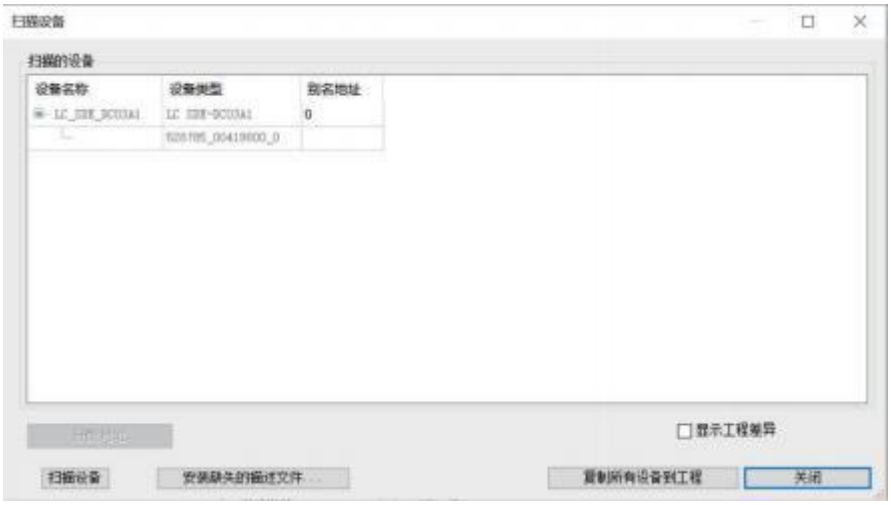
（2）选定EtherCAT网口后，点击“登录”或“在线配置模式”。

3.20

（3）登录成功后，“右键EtherCAT_Master_SoftMotion－扫描设备”。



3.21

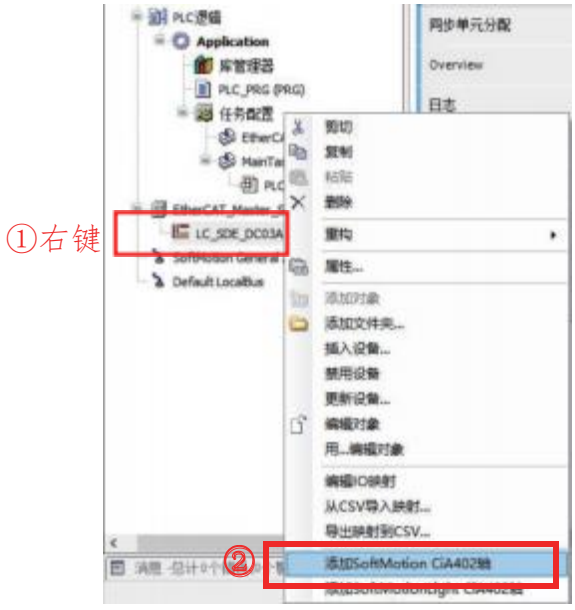


3.22

自动扫描到连接的设备后，点击“复制所有设备到工程”。

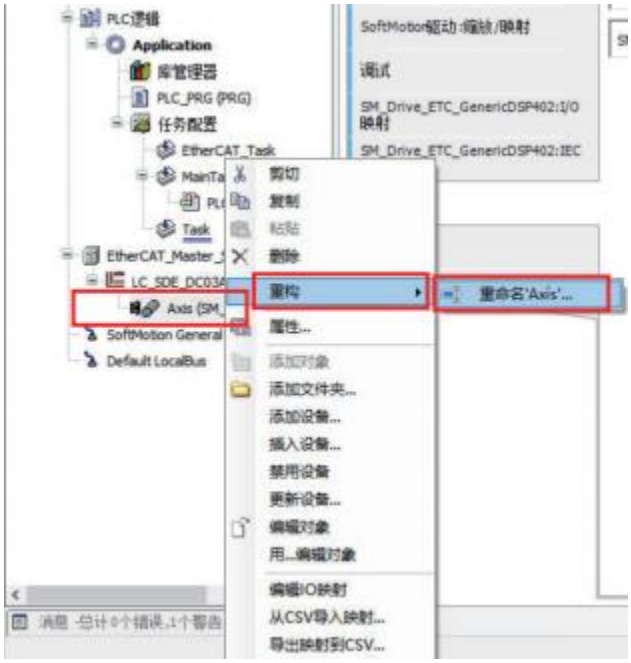
【注：设备的节点地址，默认是自动分配，即按照距主机距离从近到远来分配节点地址，本例不做修改，按默认设置。】

（4）完成之后，退出登录，在新添加的设备处，添加运动控制轴，具体操作步骤为“右键LC_SDE_DC03A1－添加SoftMotion CiA402轴”。



3.23

（5）添加完成，为方便编程，可以对新添加的轴重命名。



3.24

（6）双击新添加的SoftMotion CiA402轴，设置控制相关参数，在“SoftMotion驱动：缩放/映射”子页面内调整电机类型、比例缩放与映射等。

3.24

4. 单轴简单用户控制程序编写

此处编写一个单轴的简单程序，使控制器控制伺服电机执行绝对位置指令，做往返运动。

（1）首先建立对象（POU）（详情参考3.4.2）。

（2）打开编程界面POU_Abs（详情参照3.4.3，3.10），在变量声明区域添加变量，变量声明代码如下：

```
PROGRAM POU_Abs
VAR
    iStatus:INT;//执行步骤
    Power:MC_Power;  //使能模块
    ActPos:LREAL; //实际位置值
    MoveAbsolute:MC_MoveAbsolute; //绝对位移模块
    p:REAL:=30; //位移值
END_VAR
```

【注：声明变量时一般都是默认添加库文件“SM3_Basic”，如若未添加则需要手动通过双击“库管理器—添加库”，找到库“SM3_Basic”然后选择添加，也可以通过此方法添加更多的库。】

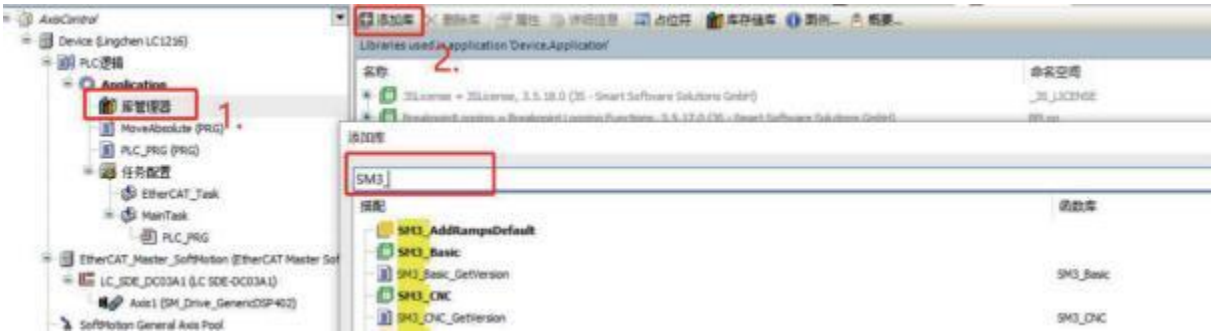


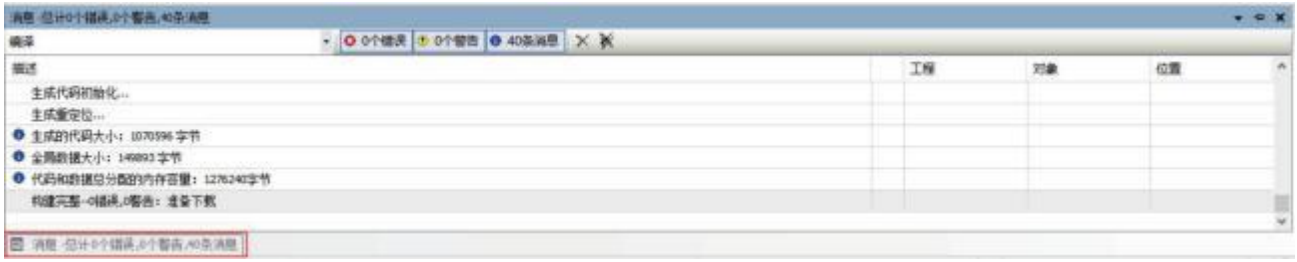
图3-

（3）在编程区域添加程序如下。（程序功能：程序执行时，立即使能伺服，等伺服使能成功后，控制电机在位置P与起点O之间做往返运动。）

```
CASE iStatus OF
0:  //启动后，轴Axis上使能
    Power(Axis:=Axis,      Enable:=TRUE      ,      bRegulatorOn:=TRUE,
bDriveStart:=TRUE );
```

```
    IF Power.Status  THEN
        iStatus:=iStatus+1;
    END_IF
1:  //走绝对位移，运行到P 处
    MoveAbsolute(Axis:=Axis, Execute:=TRUE, Position:= 0, Velocity:=100 ,
Acceleration:= 100, Deceleration:=100 );
    IF MoveAbsolute.Done THEN
        MoveAbsolute(Axis:=Axis, Execute:= FALSE); iStatus:=iStatus+1;
    END_IF
2:  //走绝对位移，运行回到O 处
    MoveAbsolute(Axis:=Axis, Execute:=TRUE, Position:= p, Velocity:=100 ,
Acceleration:= 100, Deceleration:=100 );
    IF MoveAbsolute.Done THEN
        MoveAbsolute(Axis:=Axis, Execute:= FALSE);
        iStatus:=1;
    END_IF
END_CASE
ActPos:= Axis.fActPosition;//获取轴实际位置值
```

（4）程序编写完成后，点击“编译”，确认编写没有错误。

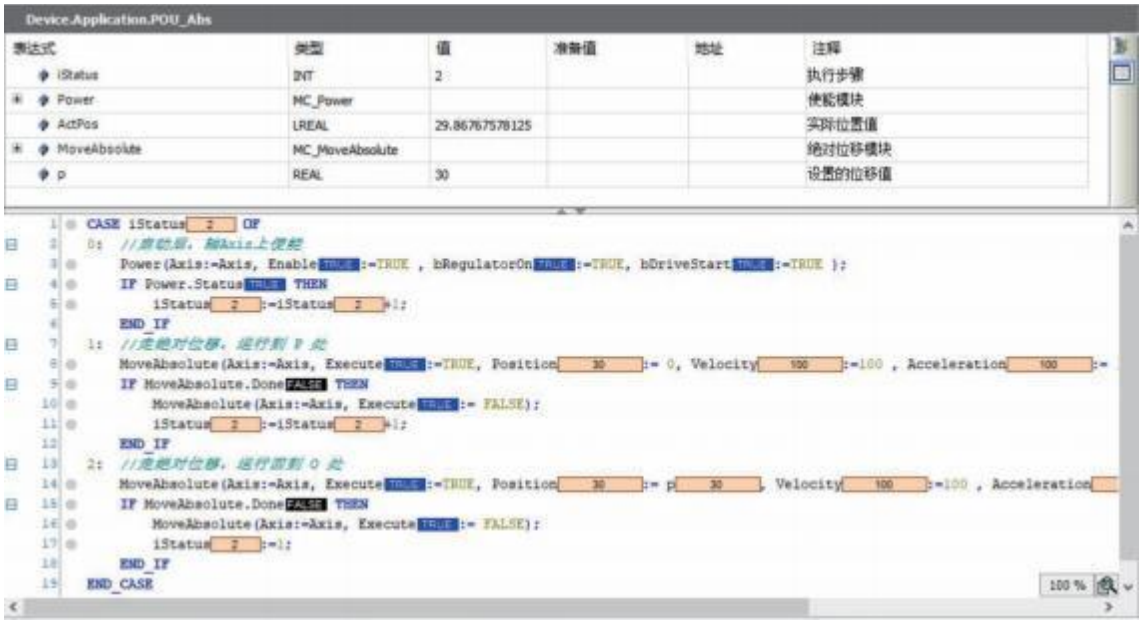


3.

在左下角的“消息”提示框内会显示编译过程描述，如果有错误，也会显示在该区域。

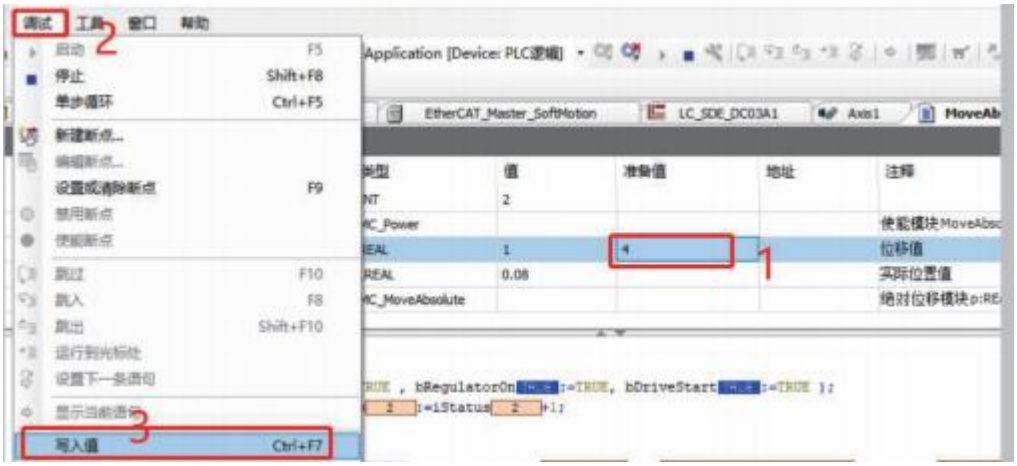
（5）将编写完成的对象（POU_Abs）拖进任务节点下。

（6）连接设备（参照3.6.1），无误后，点击“登录—启动”，打开“POU_Abs”，程序运行如下图所示，程序执行伺服使能后，可以看到电机在位置 p与起点位置O之间做往返运动。



3.

在线修改位置 P 的值：单击变量“P”的预设值“准备值”，使其为然后输入数值“10”，然后选择“调试 写入值”或者快捷键“Ctrl+F7”将值写入到“值”中，即可在线修改变量“P”的值。



3.

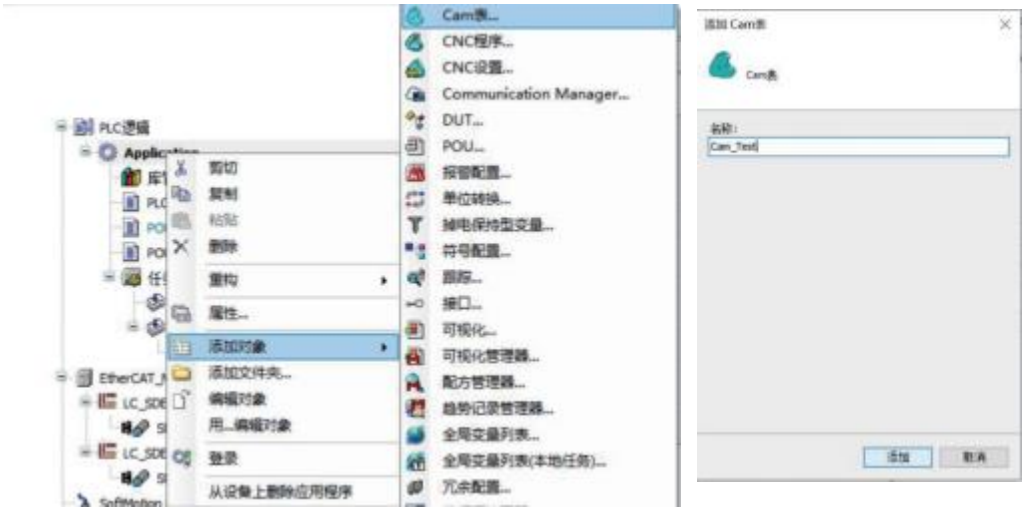
5. 基于 EtherCAT 通信的电子凸轮程序例程

参照上面的操作步骤，新建工程，添加EtherCAT Master SoftMotion主站，更改EtherCAT网口后，“登录—右键EtherCAT Master SoftMotion—扫描设备”，扫描到设备如下（本例程以两套“LC_SDE_DC03A1（凌臣总线步进_单轴）”为例），“复制所有设备到工程”，添加完毕并为每个步进添加与一个402轴，为方便编程示范分别重命名为x，y，如下图（3.25）所示（详情请参考3.6.4 单轴简单用户控制程序编写）。

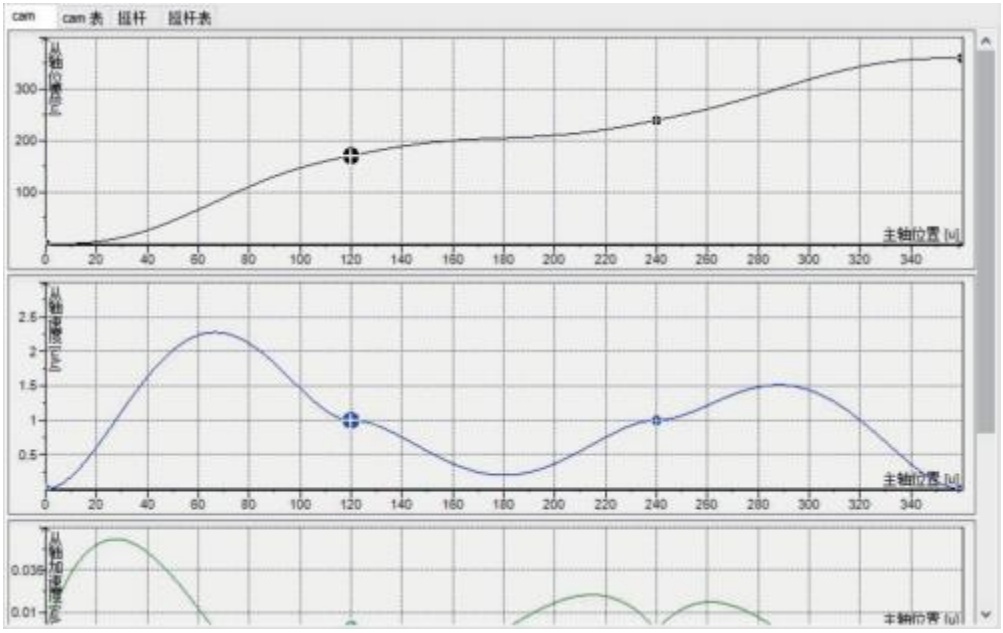
3.

3.

在Application下添加一个对象（POU_Cam），再添加一个对象（Cam表），具体操作步骤为“右键Application－添加对象－Cam表...”， 自定义Cam表的名称，



点击“添加”。添加完成后，Cam表界面如图所示。



可以对相关项进行自定义设置（本文档仅做示范作用，因此使用默认设置）。

打开编程界面POU_Cam，在变量声明区域添加变量，变量声明代码如下：

```
PROGRAM POU_Cam
VAR
    xstart          :BOOL;//启动开关
    power_x         :MC_Power;//轴x_使能模块
    power_y         :MC_Power;//轴y_使能模块
    mov         :MC_MoveRelative;//相对运动模块
    SEL1            :MC_CamTableSelect;//Cam表选择
    camin           :MC_CamIn;
    camout          :MC_CamOut;
    t               :TON;
    step            :INT;//执行步骤
END_VAR
```

在编程声明区域添加程序，程序如下：（程序功能：程序执行时，通过xstart开启运动，等伺服使能成功后，控制轴x与轴y进行电子凸轮运动。）

```
power_x( Axis:= x , Enable:= xstart , bRegulatorOn:= 1 , bDriveStart:= 1 );
power_y( Axis:= y , Enable:= xstart , bRegulatorOn:= 1 , bDriveStart:= 1 );
```

```
mov( Axis:= x , Execute:= , Distance:=10 , Velocity:= 10, Acceleration:= 100, Deceleration:=100 , Jerk:= 1000 );
SEL1( Master:= x , Slave:= y , CamTable:=Cam_Test , Execute:= , Periodic:= 1, MasterAbsolute:=0 , SlaveAbsolute:= 0 );
camin( Master:= x, Slave:= y, Execute:= , MasterScaling:=1 , SlaveScaling:= 1, StartMode:= absolute,
CamTableID:=SEL1.CamTableID , VelocityDiff:= , Acceleration:=100 , Deceleration:=100 , Jerk:=100 );
camout( Slave:= y );
t(IN:= , PT:= T#1S, Q=> , ET=> );//设定延时时间为1s
CASE step OF
0://等待轴使能
    IF power_x.Status THEN
        step:=100;
    END_IF
100://选择Cam表，开启Cam表：Cam_test
    SEL1.Execute:=1;
    step:=200;
200:
    camin.Execute:=1;//进入
    camout.Execute:=0;
    step:=500;
500:
    mov.Execute:=1;//开启相对运动
    camin.Execute:=0;
    step:=550;
550://运动到位时跳出
    IF mov.Done THEN
        camout.Execute:=1;
        step:=600;
```

```
        END_IF
600:
        mov.Execute:=0;
        t.IN:=1;
        IF t.Q THEN
            t.IN:=0;
            step:=200;
        END_IF
END_CASE
```

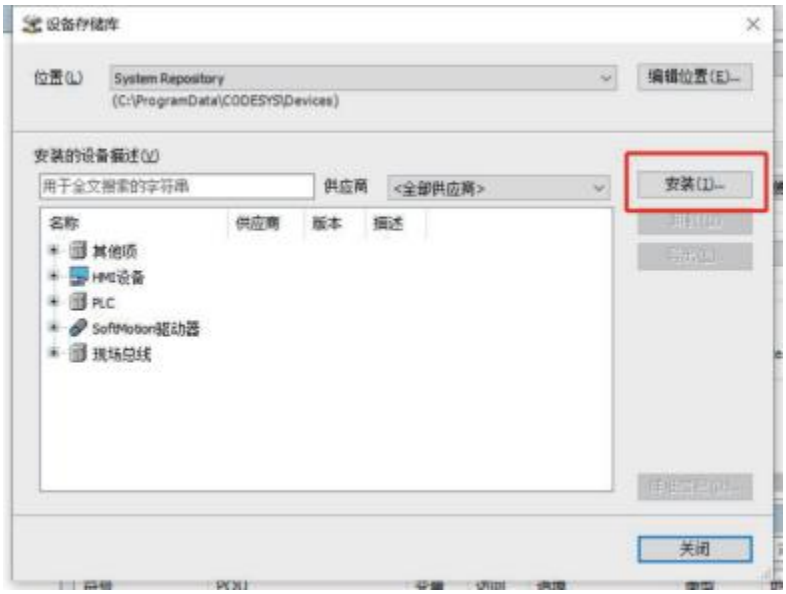
程序编写结束后，点击“编译”，确认无误，将POU_Cam放到“EtherCAT_Task”下。点击“登录”，登录成功后，点击“启动”，等驱动和轴前的图标变为绿色后，将“xstart”的值改为“true”，程序运行如下图所示。

3.

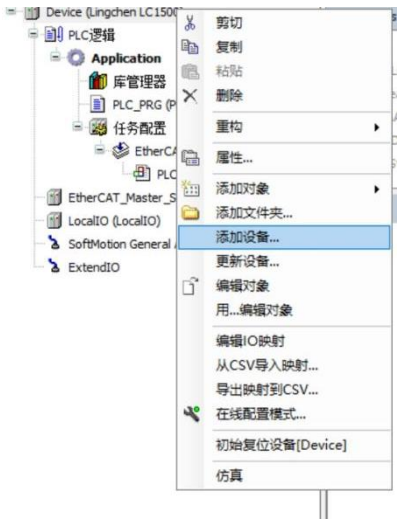
6. 添加本地IO

1. 创建一个标准工程，设备选择Lingchen LC1500；
2. 在菜单工具栏，点击“工具”，选择“设备存储库”，如下图所示；

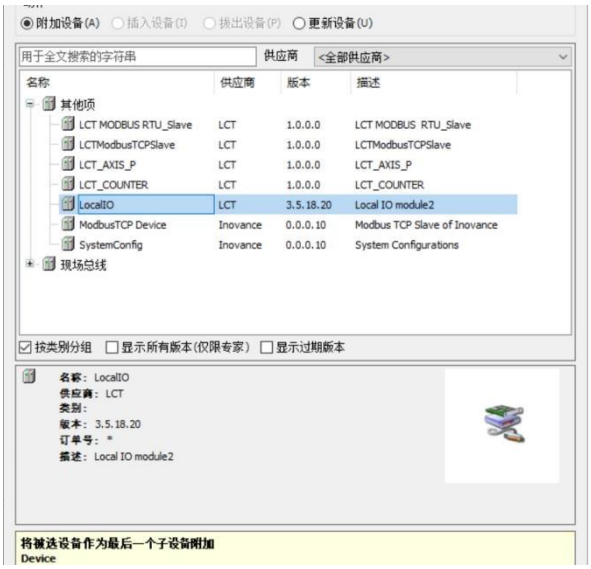
3. 在弹出的“设备存储库”对话框内，选择“安装”，选择已经存好的对应设备xml文件；



4. 安装完成后，在设备栏中右键“Device，选择“添加设备”，如下图所示：

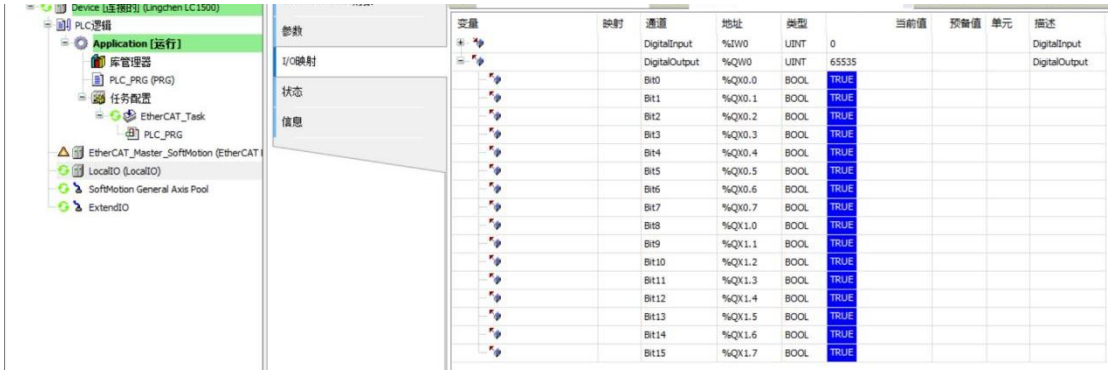


5. 在这里就可以看到前面已经安装完成的设备“LocalIO”，双击或选中后点右下角的“添加设备”，即可完成添加；



6. 添加完成后，如下图所示；

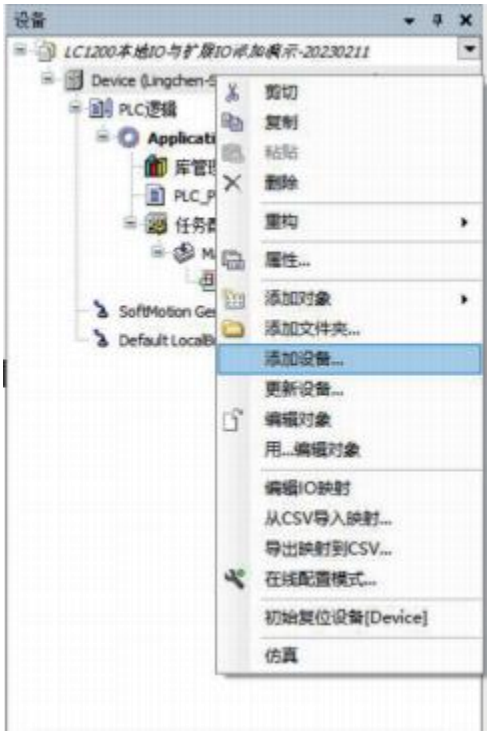
7. 登录后即可在LocalIO的“I/O映射”子界面进行查看。



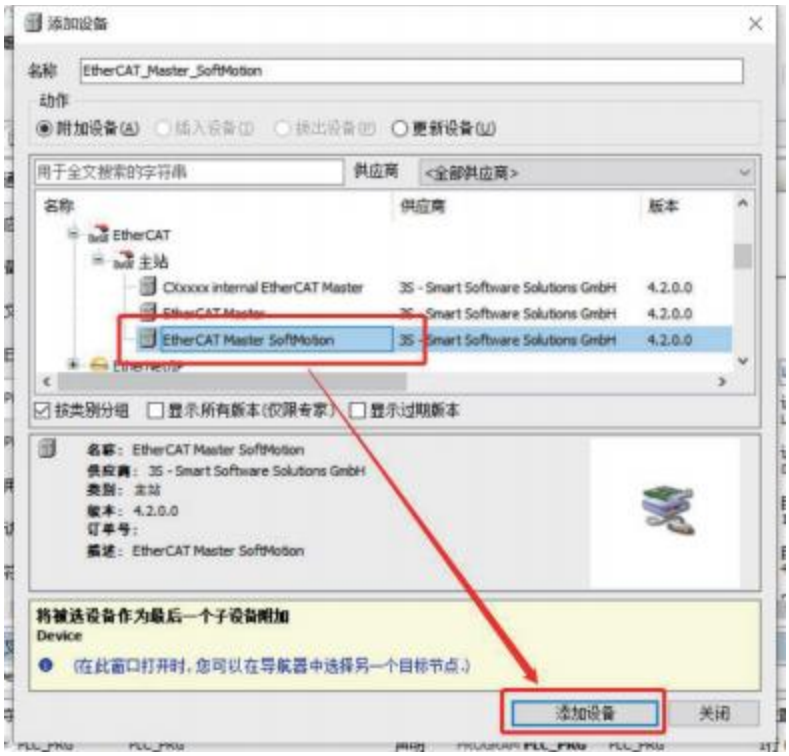
7. 添加拓展IO（以LC1100和1488、2488模块为例）

1. 同样创建标准工程后，在工具-设备存储库中导入设备文件（一次导入后，下次不用重复操作）；

2. 在设备栏，右键“Device”，选择“添加设备”；



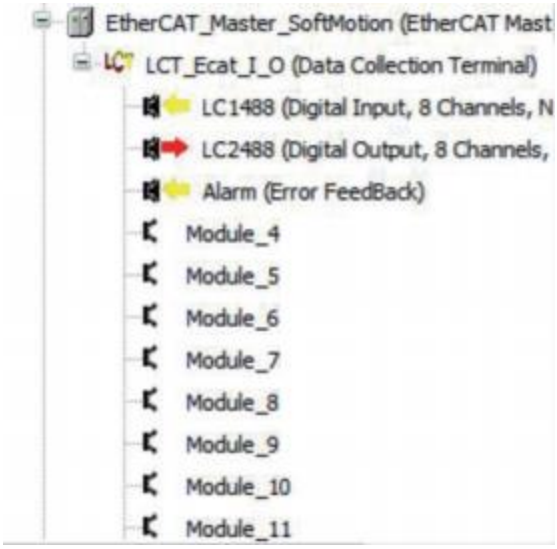
3. 添加主站；



4. 选中刚刚添加的设备，在“通用”界面选择“源地址”，选择准备连接的网口源地址；

5. 选好后，“登录”或“在线”，右键主站（EtherCAT_Master_SoftMotion），“扫描设备”，在弹出的对话框内确认设备是否全部正常后，“复制所有设备到工程”；

复制完成后，在设备栏展示如下：



6. “登录—启动”，在“设备栏”双击LC2488（输出模块）打开界面，点击“ModuleI/O映射”子界面，可以在此为输出赋值，同理，在LC1488（输入模块）界面的“ModuleI/O映射”子界面，监测输入当前值。


Device
EtherCAT_Master_SoftMotion
LCT_Ecat_1_O
LC2488
Module IO映射
Module IO对象
信息
查找
过滤器
显示所有
变量
映射
通道
地址
类型
当前值
预备值
单元
描述
%QX0.0
%QX0.1
%QX0.2
%QX0.3
%QX0.4
%QX0.5
%QX0.6
%QX0.7
BIT
FALSE
DOBN

8. Modbus

8.1 Modbus TCP Master

配合使用软件：Modbus Slave

具体操作步骤：

关闭电脑防火墙。

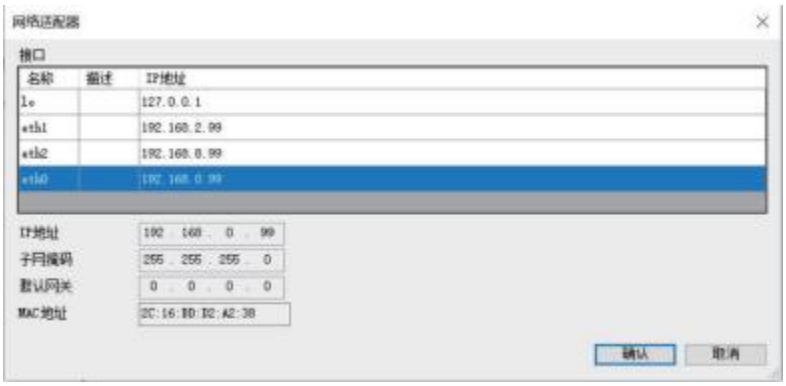
（1）新建标准工程文件；

（2）右键“Device”—添加设备；

（3）点开“以太网适配器”子选项，选中“Ethernet”，点击“添加设备”；

（4）添加完成后，双击打开Ethernet设备界面，在“通用”子界面，点击“Browse...”选择网络接口（选择网络接口前需要先连接设备Device）；

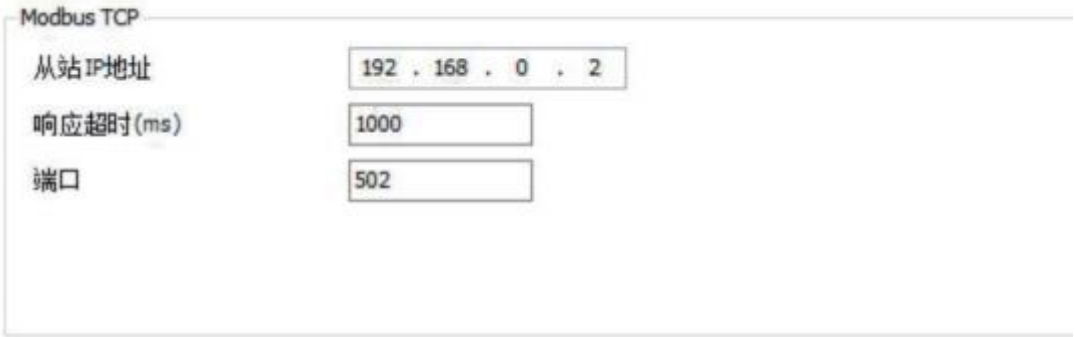
（5）选中“eth0”（也可以自定义），点击确认；



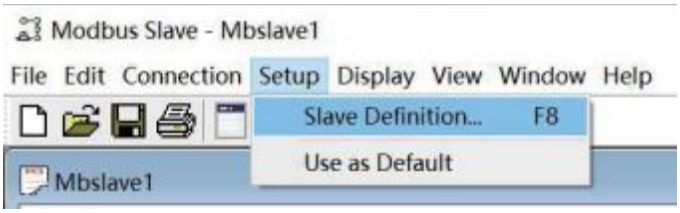
（6）“右键Ethernet—添加设备—按照图中所示位置，找到Modbus TCP Master选中—添加设备—关闭”；

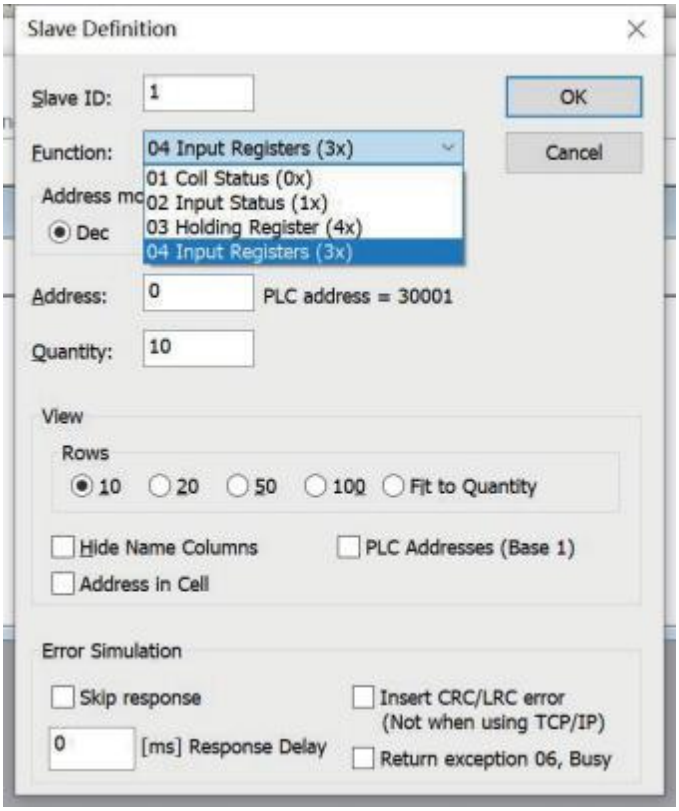
（7）“右键Modbus TCP Master—添加设备—按照图中所示位置，找到Modbus TCP slave选中—添加设备—关闭”；

（8）Master设备界面不需要做修改，双击打开slave设备界面，设置要连接的从站ip（也即测试笔记本使用的ip，这里设定的是192.168.0.2，因此如下图填入）；



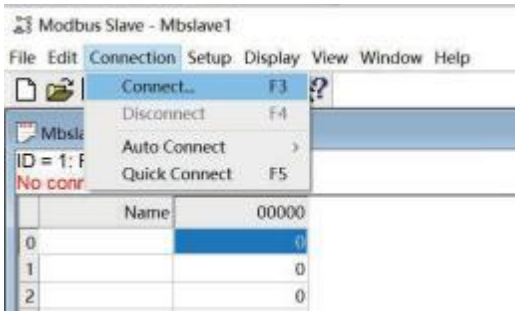
（9）设置完成后，打开Modbus Slave程序，点击“Setup －Slave Definition...”；



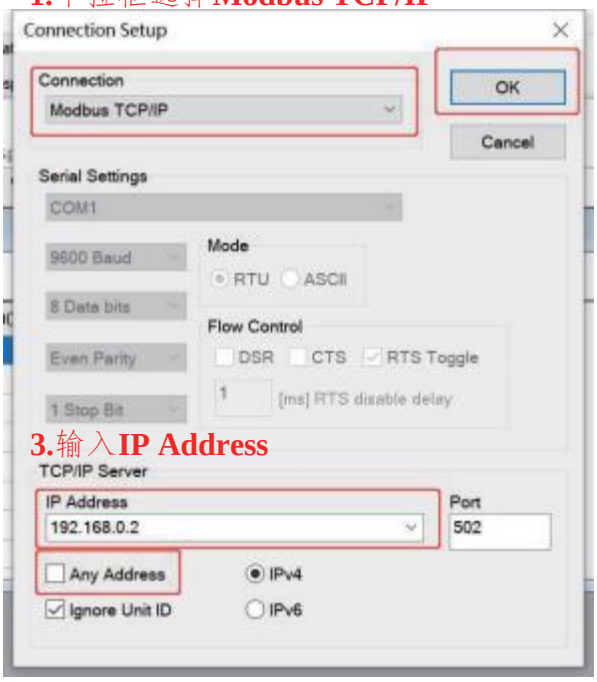


功能码选择04，可以改变长度（Quantity），点击“OK”。

（10）点击“Connection－Connect...”



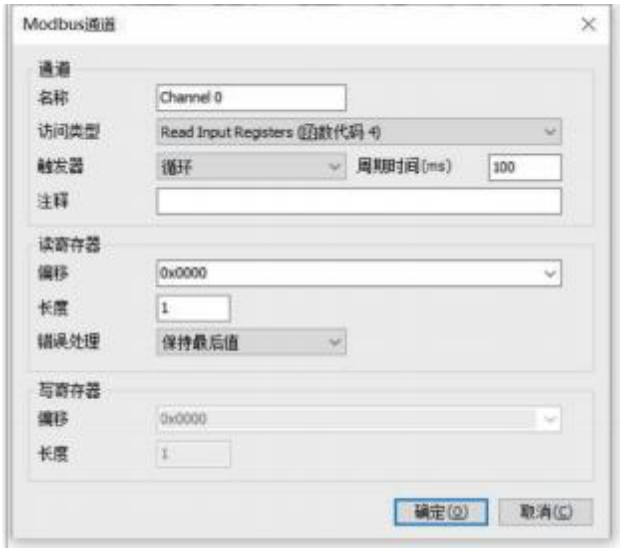
在弹出的界面内，先选择“connection ”为“Modbus TCP/IP ”，然后去掉“Any Address ”前的勾选，将测试电脑的ip地址（192.168.0.2）输入至“IP Address ”框内，点击“OK ”。



（11）此时已经连接上，返回codesys界面，在Modbus_TCP_Slave界面的“Modbus从站通道”子界面内，点击“添加通道”。

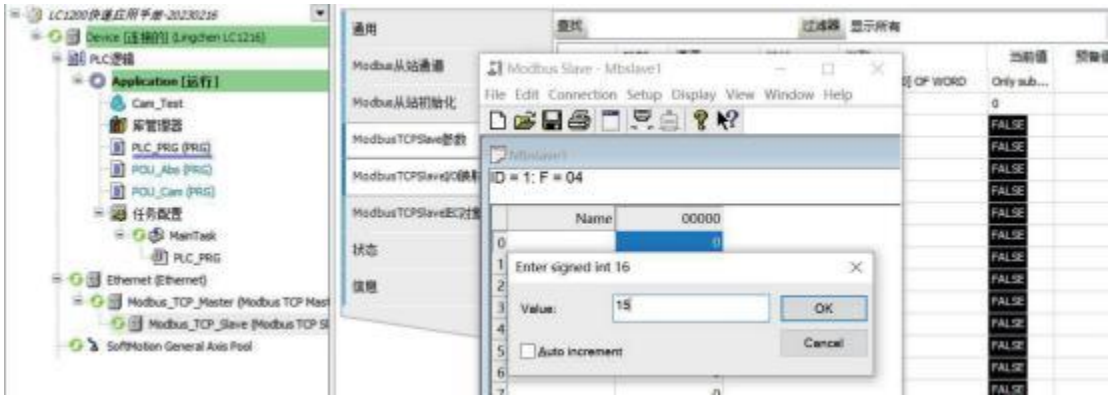
.

添加Read Input Register通道（访问类型选“Read Input Register通道（函数代码4）”），可对从站寄存器进行读操作；

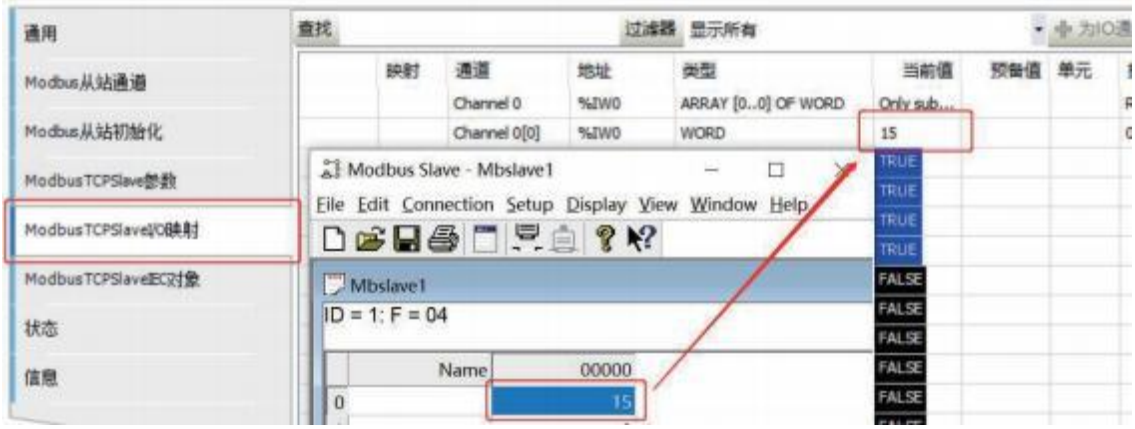


3	Channel 3	Read Input Registers (功能代码 04)	循环, t#100ms	16#0000	1	保持最后值

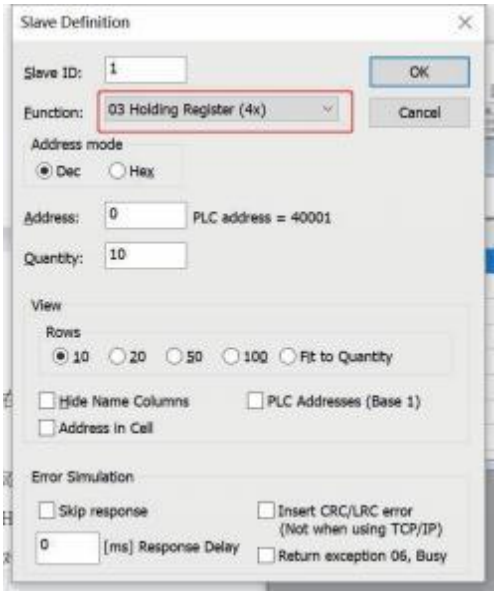
按照上图所示，读寄存器的地址偏移为0x0000，因此在Modbus Slave中双击地址为0的地方，在弹出的窗口内，输入数值（如15），点击“OK”写入。



在Modbus_Slave_TCP的“ModbusTCPSlaveI/O映射”子界面内可以看到数值已经同步过去。

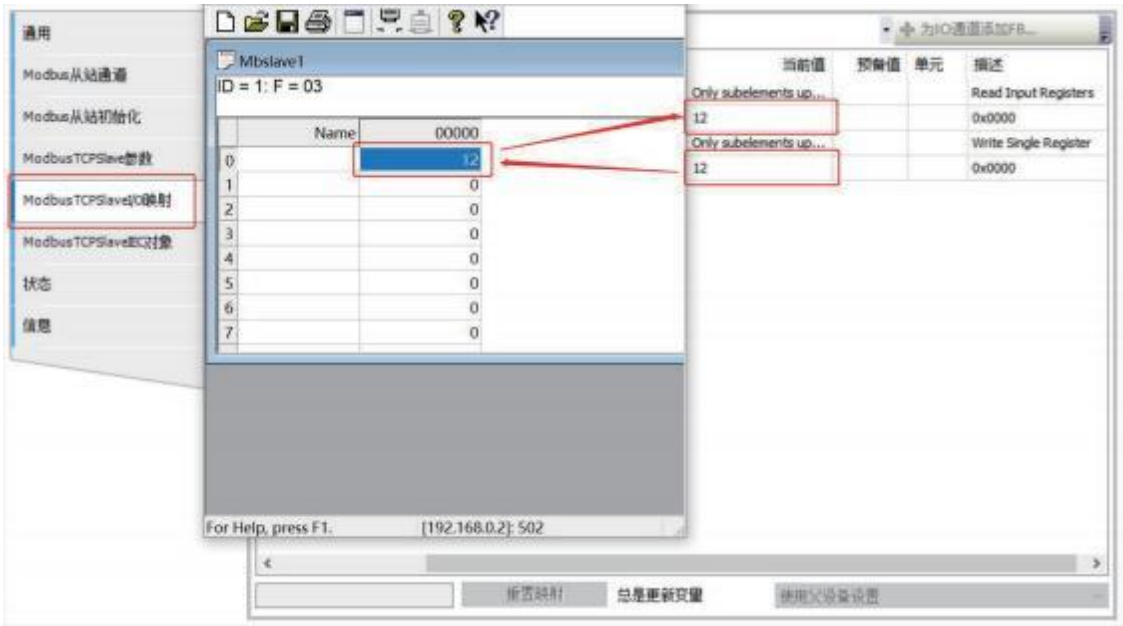


（12）在Modbus Slave程序中，选功能码03（03 Holding Register (4x)）



在添加Read Holding Register和Write Single Register通道（访问类型分别选“Read Holding Register（函数代码3）”和“Write Single Register（函数代码6）”），可对从站寄存器进行读写操作。

1	Channel 1	Write Single Register (函数代码 06)	上升沿				16#0000	1
2	Channel 2	Read Holding Registers (函数代码 03)	上升沿	16#0000	1	保持最后值		



如上图所示，读写地址都为0x0000，因此在“ModbusTCPSlaveI/O映射”子页面内，写入“12”时，ModbusSlave程序内和I/O映射界面的读输入寄存器位置都可以看到数值“12”。

8.2 Modbus TCP Slave

使用软件：Modbus Poll

具体操作步骤：

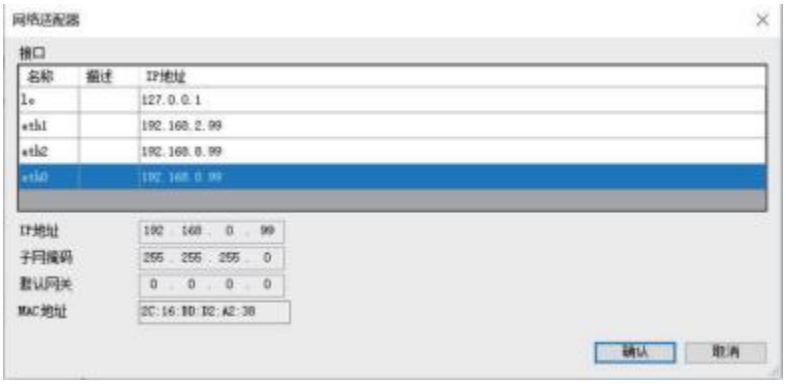
（1）新建标准工程文件；

（2）右键“Device”—添加设备；

（3）点开“以太网适配器”子选项，选中“Ethernet”，点击“添加设备”；

（4）添加完成后，双击打开Ethernet设备界面，在“通用”子界面，点击“Browse...”选择网络接口（选择网络接口前需要先连接设备Device）；

（5）选中“eth0”（也可以自定义），点击确认；

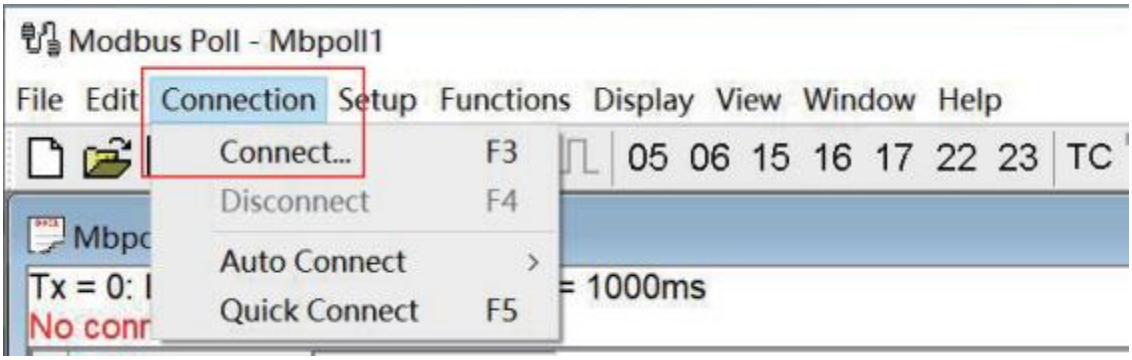


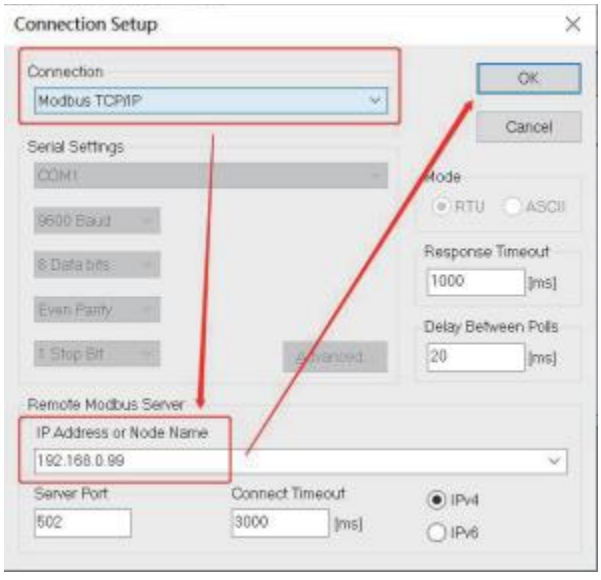
（6）“右键Ethernet—添加设备—按照图中所示位置，找到Modbus TCP Slave选中—添加设备—关闭”；

（7）双击打开刚添加的“Slave”设备—“通用”界面，可以设定保持寄存器和输入寄存器的长度及起始地址，若需要用到线圈和离散输入，请勾选“离散位区域”，并对其长度进行设置；

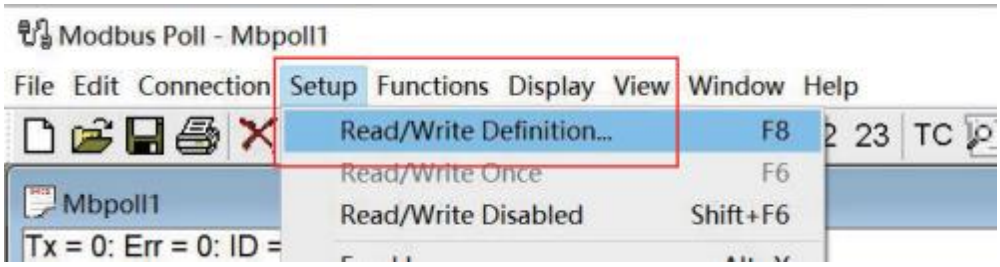
【注：功能码03，06，05对应的是保持寄存器，写线圈偏移0对应的是bit8。】

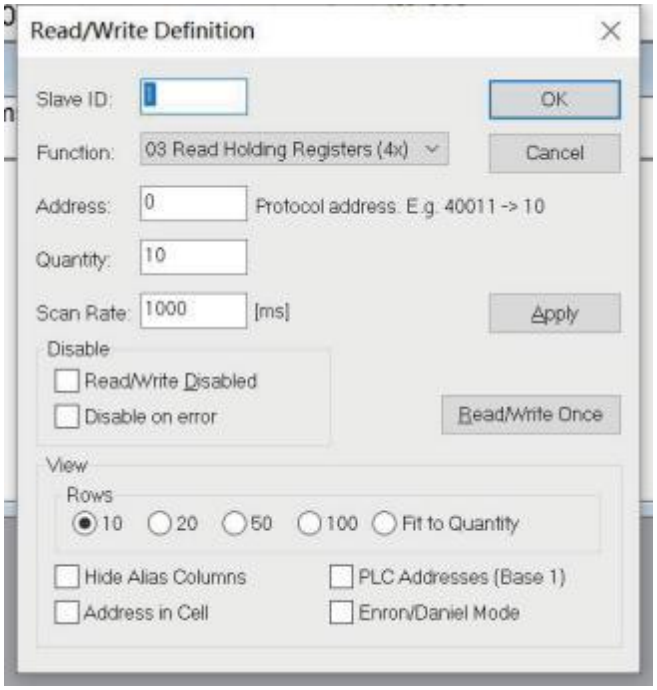
（8）打开Modbus Poll程序，点击“Connection－Connect...”，输入PLC的ip地址；



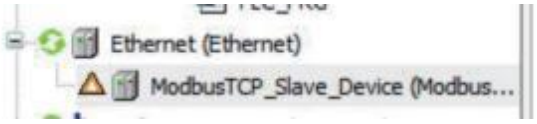


（9）在“Setup－Read/Write Definition...”中，可以设置读写功能码（本例针对保持寄存器或输入寄存器）、读写地址以及长度等；

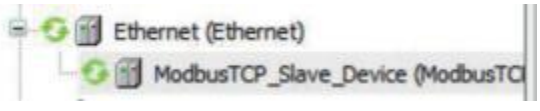


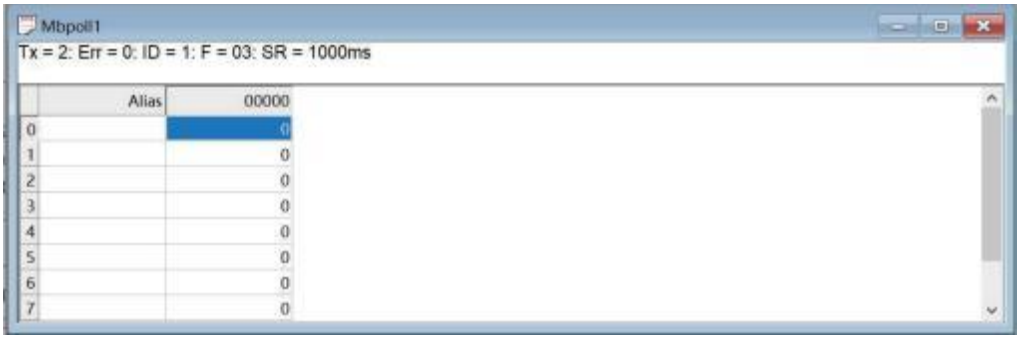


（10）对保持寄存器操作：在Codesys软件中，登录设备，启动，会发现如下状态，这是因为Modbus Poll软件未启动导致，在上图中的Function（功能）处选择功能码03，确定之后，点击“Connection－Connect...”；

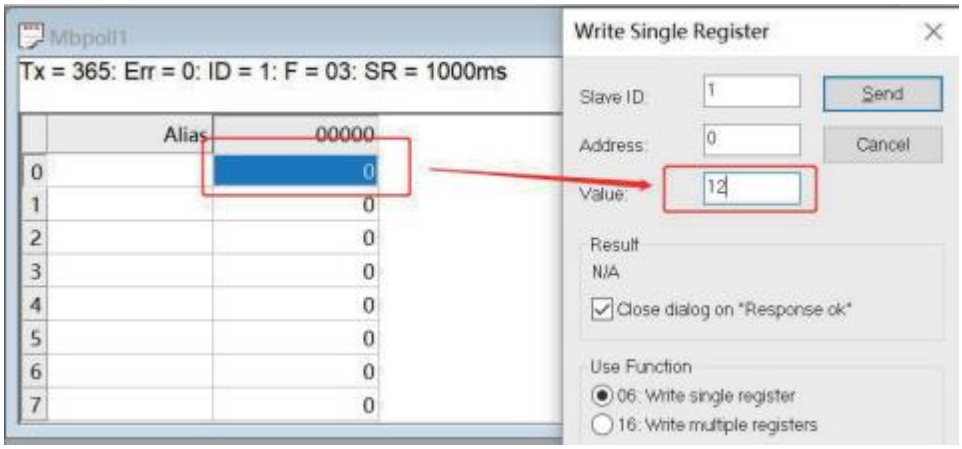


连接成功，如下图所示，





此时可以在Modbus Poll中可写对应寄存器值，

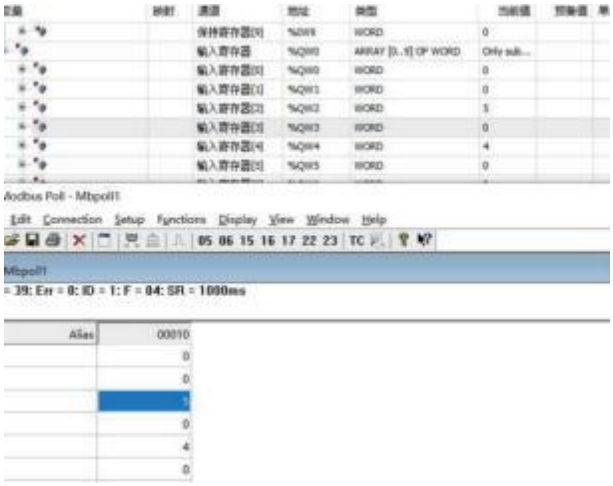


输入值12，点击“Send”发送成功后，可以在相应位置看到结果，点开“Modbus TCP Slave DeviceI/O”界面，发现保持寄存器[0]已经有数据收到。

（11）对输入寄存器操作：在Codesys软件中，登录设备，启动，在下图中的

Function（功能）处选择功能码04，地址改为10，确定之后，点击“Connection −Connect…”；

同样连接成功后，在PLC的“Modbus TCP Slave DeviceI/O”界面的输入寄存器写入值，在Modbus Poll中可读对应寄存器值。



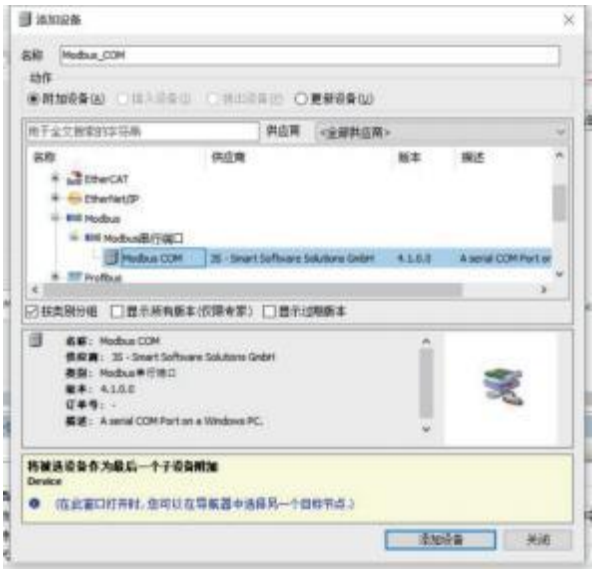
8.3 Modbus RTU

注意：请先关闭电脑防火墙。

LC1500只具有Modbus Serial Master功能，因此将其设为主站，操作步骤如下：

1. 右键“Device”—“添加设备”。

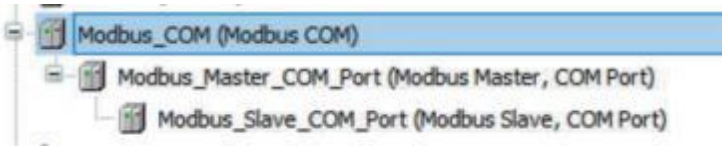
2. 依次展开“现场总线—Modbus—Modbus串行端口—Modbus COM”，选中后，点击“添加设备”。



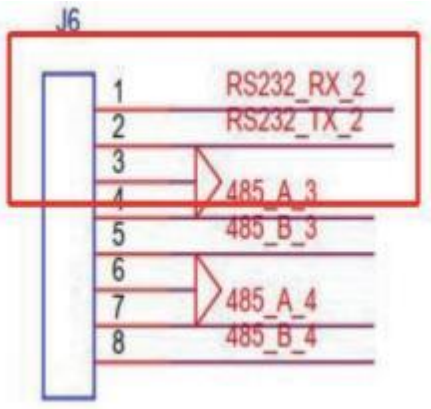
3. 添加完成后，可以不用关闭“添加设备”窗口，点击设备栏刚刚添加的“Modbus_COM”，“添加设备”的窗口会自行跳转，点击串行主站，选中添加。

添加完串行主站，同样不用关闭“添加设备”窗口，点击设备栏刚刚添加的Modbus_Master_Com_Port，“添加设备”窗口跳转，点击串行设备，选中添加。

4. 添加完成后，设备栏如下图所示，双击“Modbus_COM”打开界面，选择“通用”子界面，修改COM端口及其他串口配置参数。



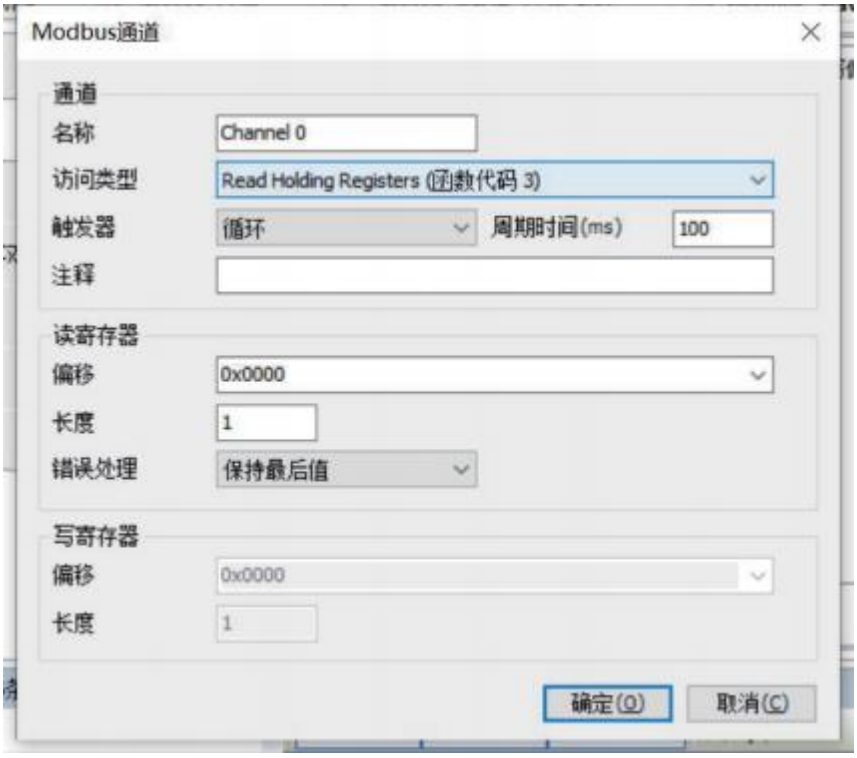
注意，端口号要和硬件连接的端口对应，如选232，则端口号为2，选485_3，端口号为3，选485_4,端口号为4。



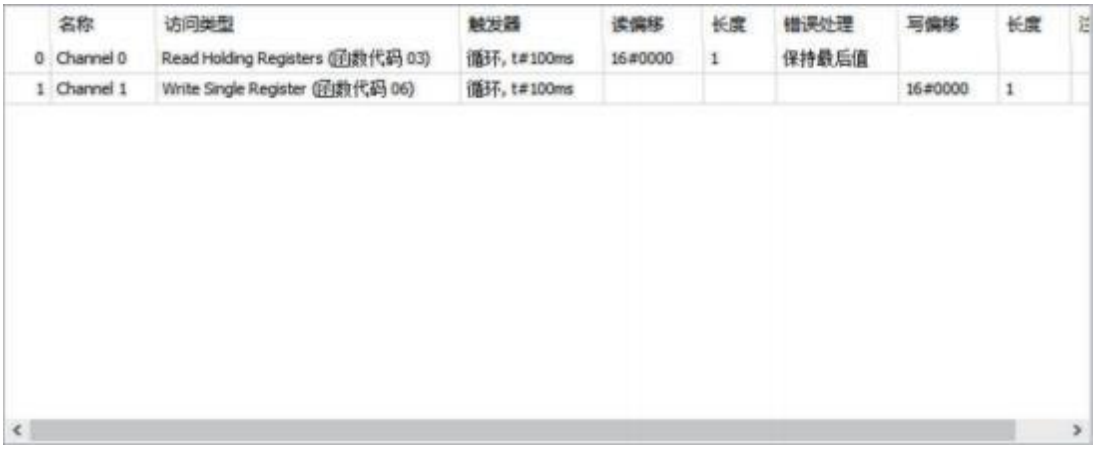
5. 双击“Modbus_Slave_Com_Port”打开界面，在“通用”界面可以选择从站地

址（1~247），在“Modbus从站通道”子界面点击“添加通道”。

6. 在弹出的界面选择访问类型及地址偏移。



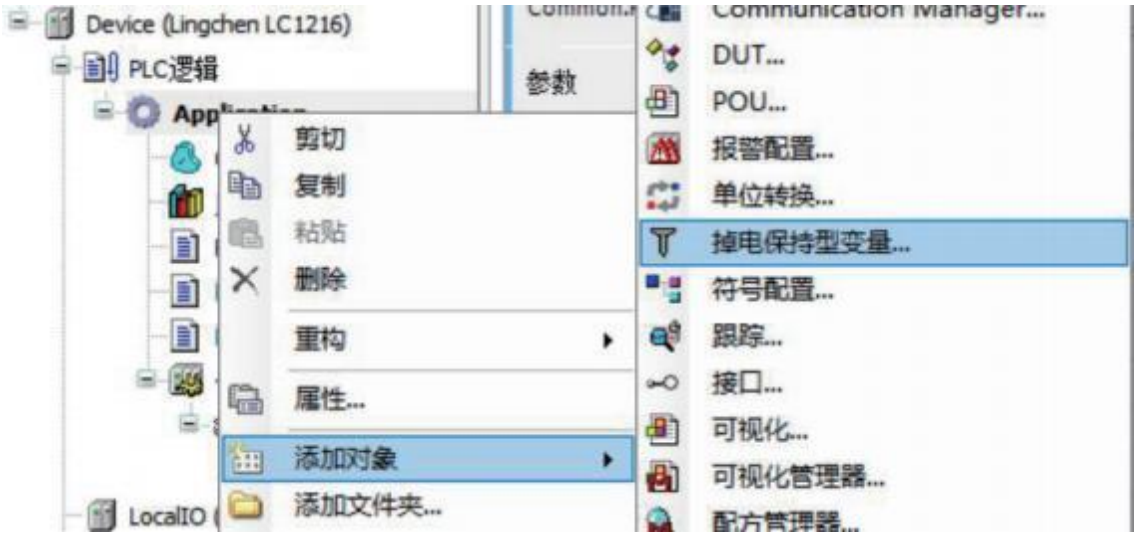
7. 添加完成后，如下图所示。可以配合其他串行从站进行使用。



	名称	访问类型	触发器	读偏移	长度	错误处理	写偏移	长度
0	Channel 0	Read Holding Registers (功能代码 03)	循环, t#100ms	16#0000	1	保持最后值		
1	Channel 1	Write Single Register (功能代码 06)	循环, t#100ms				16#0000	1

9. 断电保持功能的使用

1. 添加“掉电保持型变量”对象。右键“Application”—添加对象—掉电保持型变量。



设定列表名称（这里使用默认名称），点击“添加”。

2. 添加完成后，会自动跳转该页面，注释掉首行，声明一个数组变量。

```
1  //{attribute 'qualified_only'}
2  VAR_GLOBAL PERSISTENT RETAIN
3      test:ARRAY [1..20] OF USINT;
4  END_VAR
```

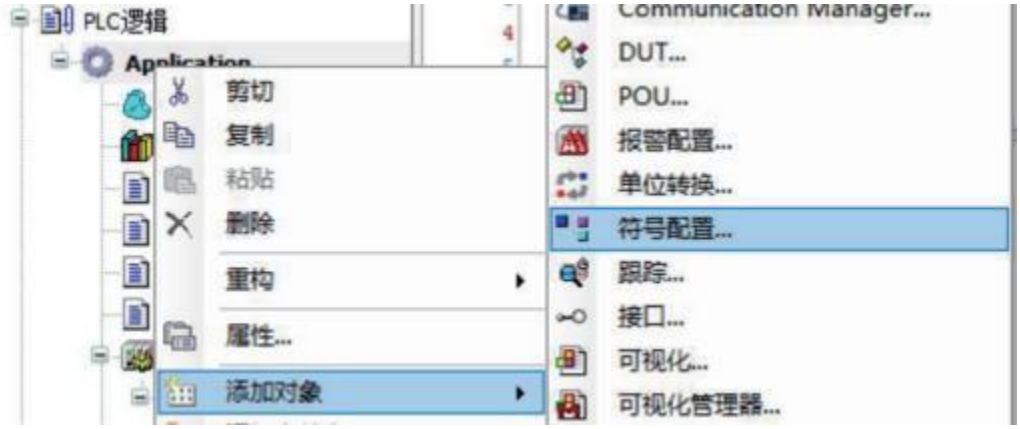
3. 登录—启动，打开掉电保持界面，展开变量，随意输入准备数字，“Ctrl+F7”写入。

4. 断掉LC1500电源，重新上电，待PLC重新运行起来后，登录查看。

因此，可将相关变量声明为掉电保持型变量，掉电后再次上电，依旧可以保存下来。

10. OPC UA例程

1. 编写好程序之后，右键“Application”—添加对象—符号配置...。



2. 在弹出的“添加 符号配置”界面，注意是否勾选支持OPC UA特征，点击“添加”。

3. 添加完成后，可以看到“符号配置界面”，注意，需要先执行“编译”命令，

才能够选择需要的变量，点击“编译”。在此之前，请确认需要的变量所在POU已拖入任务配置，分配Task。勾选变量。

4. 勾选完后，在菜单栏点击“编译——生成代码”。

5. 生成代码结束，可以在消息栏看到相关信息。

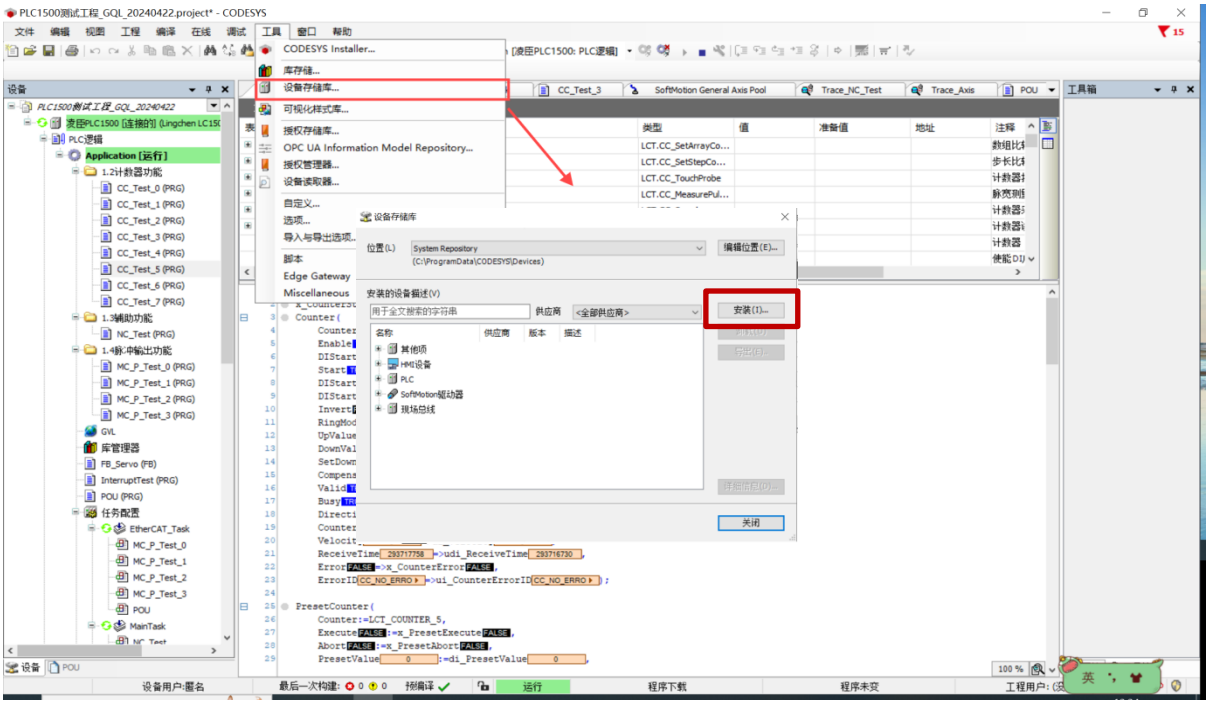
6. 此时，可以在该工程文件的存储目录下，找到生成的xx.Device.Application.xml文件。

7. 之后使用人机界面的组态程序，扫描标签即可。

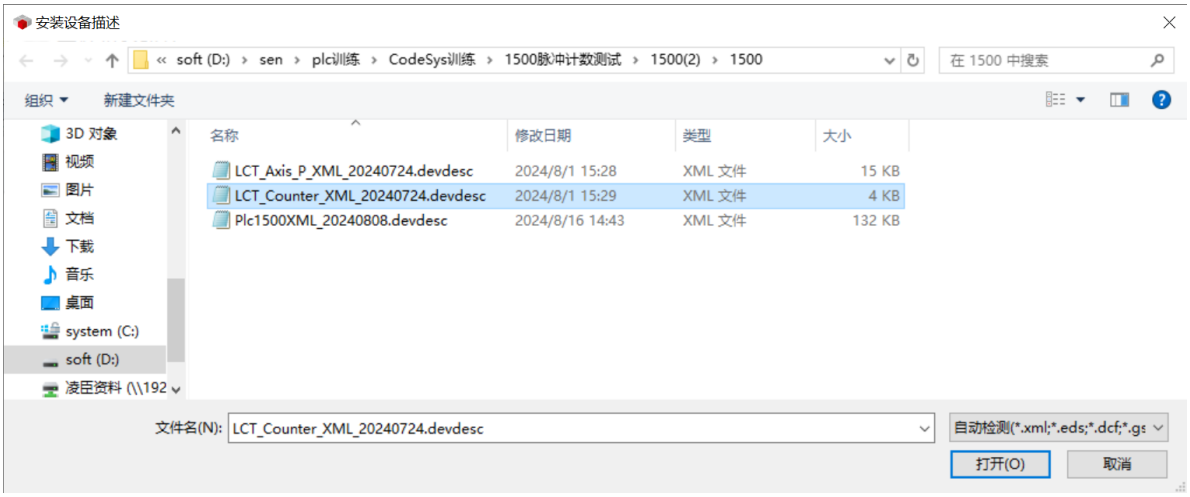
8.4 高速脉冲计数

1. 库文件安装及应用

点击工具栏“工具”选项，选择“设备存储库”，然后点击“安装”。

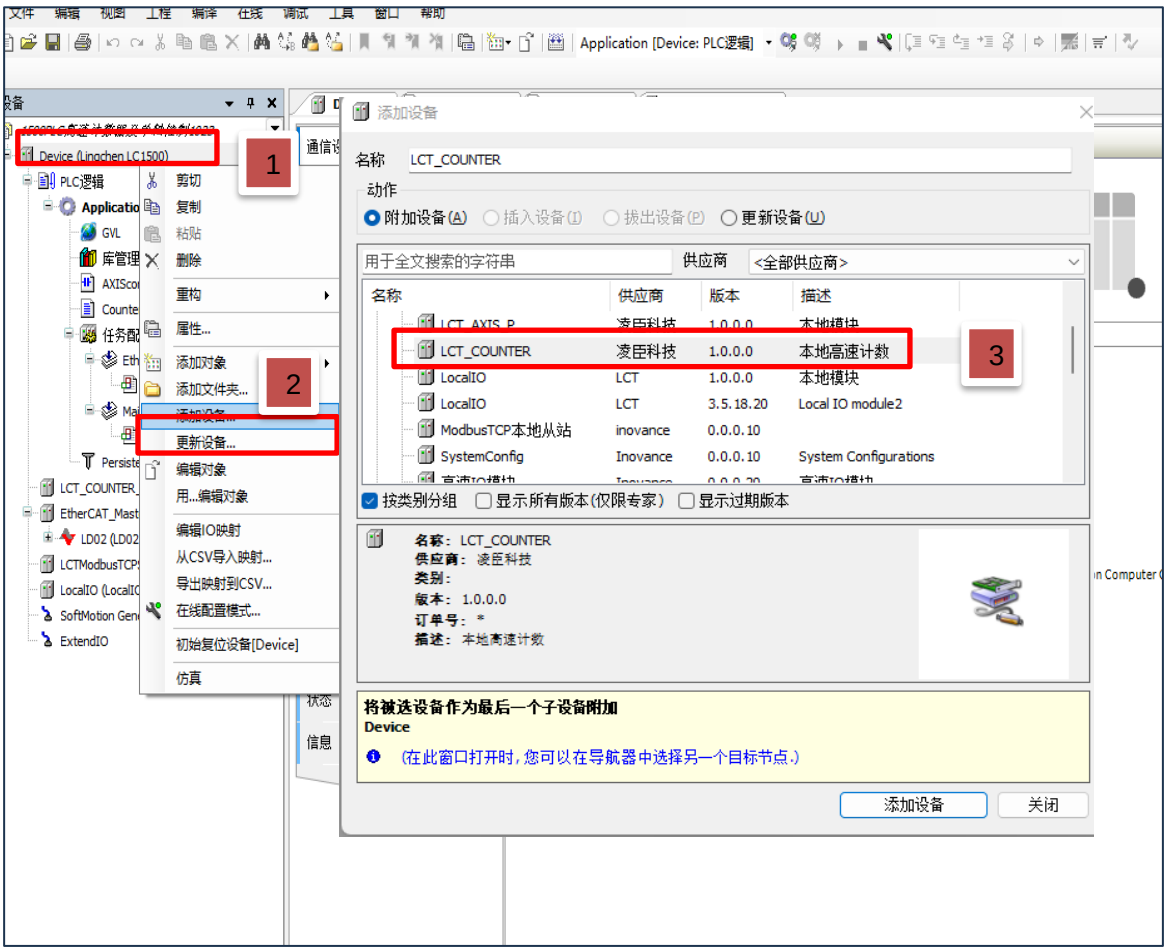


选择“LCT_Counter_XML”XML文件，点击打开进行安装。

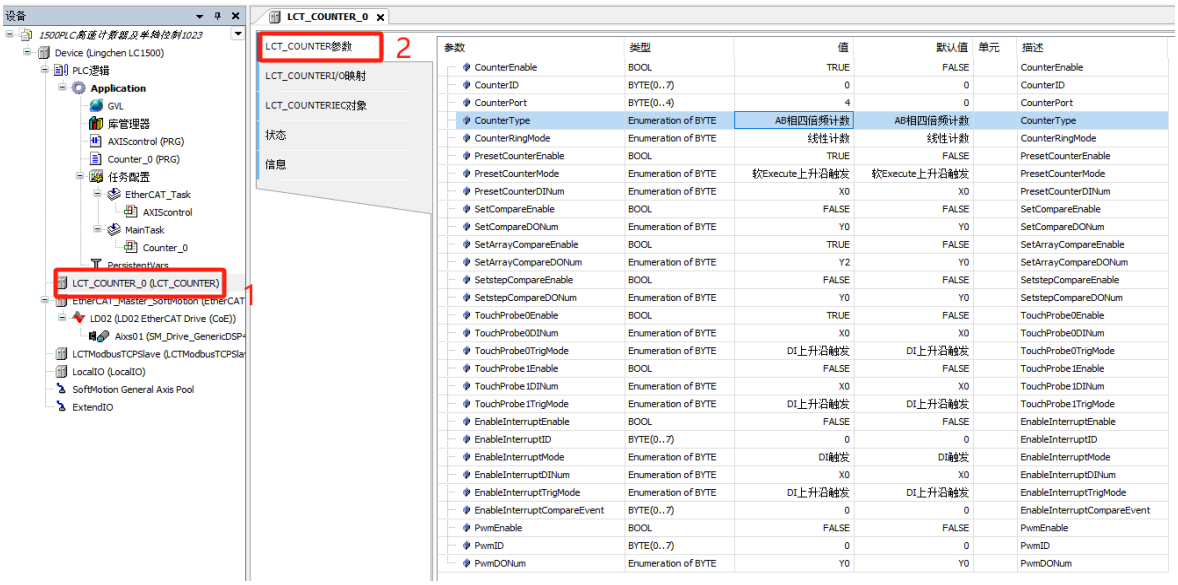


2.设备添加

右击“plc1500”，选择“添加设备”，选择“其他项”，选择“LCT_COUNTER”,点击确定。左图为添加后的计数器设备



点击计数器组件，即可进入计数器参数配置界面如下：



3.计数器参数配置

计数器ID

数值范围[0..7]，每一个计数器对应一个ID，每个计数器有且只有一个ID。

计数器输入硬件编号

数值范围[0..4]，编号1-3为常规脉冲单端计数器，编号4为编码器差分计数器；

常规脉冲单端计数器：

硬件编号0对应硬件输入X0-X1两个输入口，硬件编号1对应硬件输入X2-X3两个输入口，硬件编号2对应硬件输入X4-X5两个输入口，硬件编号3对应硬件输入X6-X7两个输入口。

编码器脉冲差分计数器：

硬件编号4，对应硬件输入X10-X17八个输入口；端口X10、X11分别为EA+、EA-；端口X12、X13分别为EB+、EB-；端口X14、X15分别为EZ+、EZ- ；端口X16、X17分别为编码器电源5V输出、编码器电源0V输出。

计数类型

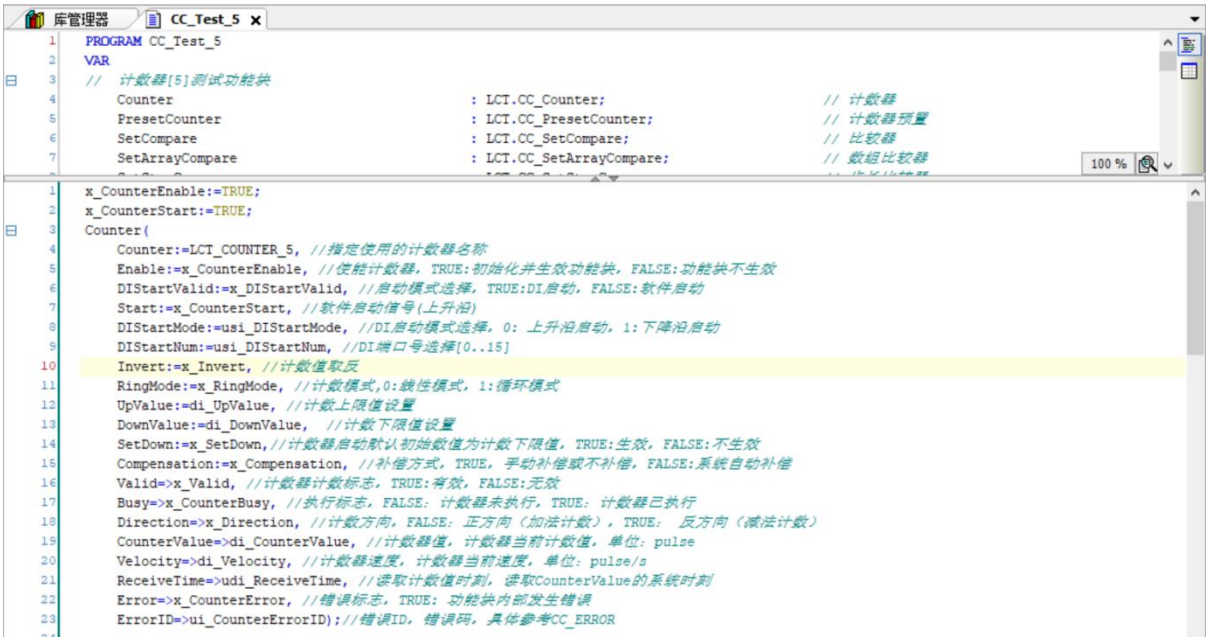
一输入单相计数：单相脉冲单输入口计数，无方向。

二输入单相计数：单相脉冲双输入口正反向计数，输入口一脉冲输入加计数，输入口二脉冲输入减计数。

AB相四倍频计数：A、B相双输入正交计数，若A相早于B相则加计数，若B相早于A相则减计数。

4.计数器指令块应用

声明计数器“Counter:Lct.CC_Counter;”实例化



功能块操作说明

```
Counter(
    Counter:=LCT_COUNTER_5, //指定使用的计数器名称
    Enable TRUE :=x_CounterEnable TRUE , //使能计数器，TRUE:初始化并生效功能块，FALSE:功能块不生效
    DIStartValid FALSE :=x_DIStartValid FALSE , //启动模式选择，TRUE:DI启动，FALSE:软件启动
    Start TRUE :=x_CounterStart TRUE , //软件启动信号(上升沿)
    DIStartMode 0 :=usi_DIStartMode 0 , //DI启动模式选择，0：上升沿启动，1:下降沿启动
    DIStartNum 0 :=usi_DIStartNum 0 , //DI端口号选择[0..15]
    Invert FALSE :=x_Invert FALSE , //计数值取反
    RingMode FALSE :=x_RingMode FALSE , //计数模式,0:线性模式，1:循环模式
    UpValue 1000 :=di_UpValue 1000 , //计数上限值设置
    DownValue -1000 :=di_DownValue -1000 , //计数下限值设置
    SetDown FALSE :=x_SetDown FALSE ,//计数器启动默认初始数值为计数下限值，TRUE:生效，FALSE:不生效
    Compensation FALSE :=x_Compensation FALSE , //补偿方式，TRUE，手动补偿或不补偿，FALSE:系统自动补偿
    Valid TRUE =>x_Valid TRUE , //计数器计数标志，TRUE:有效，FALSE:无效
    Busy TRUE =>x_CounterBusy TRUE , //执行标志，FALSE：计数器未执行，TRUE：计数器已执行
    Direction FALSE =>x_Direction FALSE , //计数方向，FALSE：正方向（加法计数），TRUE：反方向（减法计数）
    CounterValue 40000 =>di_CounterValue 40000 , //计数器值，计数器当前计数值，单位：pulse
    Velocity 0 =>di_Velocity 0 , //计数器速度，计数器当前速度，单位：pulse/s
    ReceiveTime 1469865941 =>udi_ReceiveTime 1469865941 , //读取计数值时刻，读取CounterValue的系统时刻
    Error FALSE =>x_CounterError FALSE , //错误标志，TRUE：功能块内部发生错误
    ErrorID CC_NO_ERRO =>ui_CounterErrorID CC_NO_ERRO );//错误ID，错误码，具体参考CC_ERROR
```

功能块操作说明

1，“Counter”引脚绑定添加的计数器名称，该功能块使用前必须使能计数器“Enable”引脚，否则计数器不生效。

2，“Start”，0->1跳变：启动计数器。

3，“Invert”引脚决定计数器计数方向，默认FALSE，随接收脉冲量累加；置TRUE，随接收脉冲量递减。

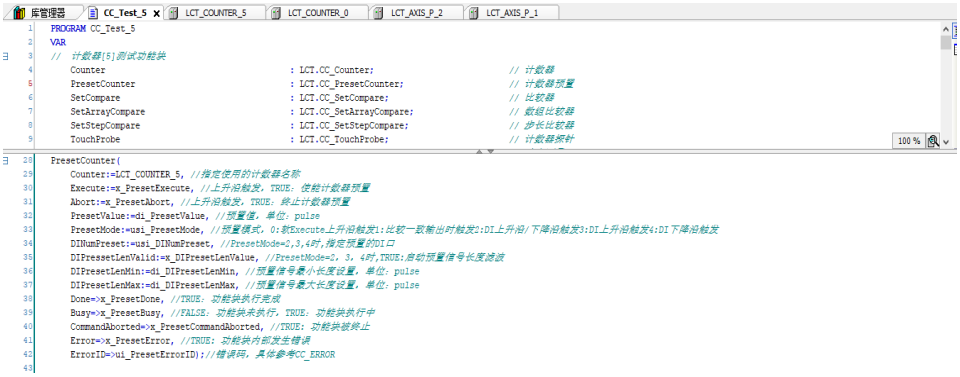
4，“RingMode”引脚决定计数器计数模式，默认FALSE，计数器数值可以逐渐增大，持续累加；置TRUE，计数器计数数值在引脚“UpValue”、“DownValue”之间循环计数。

5，“SetDown”引脚决定计数器初始值，默认FALSE，计数器计数初始值为0,；置TRUE，计数器初始值为设定的“DownValue”引脚值。

6，“DIStartValid”引脚置“TRUE”时，计数器则不受“Start”引脚控制，“DIStartNum”引脚选择的输入点信号置“TRUE”时，计数器启动，可通过“DIStartMode”引脚选择输入点信号上升沿或下降沿启动计数器。

7，“RingMode”引脚激活时，计数器会在“UpValue”上限引脚和“DownValue”下限引脚设定的范围内循环计数，当数值到达上限，计数会从下限重新计数；若“RingMode”引脚不激活则计数一直累加。

声明计数器预置“PresetCounter: LCT.CC_PresetCounter;”实例化



```
PROGRAM CC_Test_5
VAR
//  计数器[5]测试功能块
    Counter                         : LCT.CC_Counter;                   // 计数器
    PresetCounter                   : LCT.CC_PresetCounter;             // 计数器预置
    SetCompare                      : LCT.CC_SetCompare;                // 比较器
    SetArrayCompare                 : LCT.CC_SetArrayCompare;           // 数组比较器
    SetStepCompare                  : LCT.CC_SetStepCompare;            // 步长比较器
    TouchProbe                      : LCT.CC_TouchProbe;                // 计数器探针

PresetCounter(
    Counter:=LCT_COUNTER_5, //指定使用的计数器名称
    Execute:=x_PresetExecute, //上升沿触发，TRUE：使能计数器预置
    Abort:=x_PresetAbort, //上升沿触发，TRUE：终止计数器预置
    PresetValue:=di_PresetValue, //预置值，单位：pulse
    PresetMode:=usi_PresetMode, //预置模式，0:软Execute上升沿触发1:比较一致输出时触发2:DI上升沿/下降沿触发3:DI上升沿触发4:DI下降沿触发
    DINumPreset:=usi_DINumPreset, //PresetMode=2,3,4时,指定预置的DI口
    DIPressetLenValid:=x_DIPresetLenValue, //PresetMode=2，3，4时,TRUE:启动预置信号长度滤波
    DIPresetLenMin:=di_DIPresetLenMin, //预置信号最小长度设置，单位：pulse
    DIPresetLenMax:=di_DIPresetLenMax, //预置信号最大长度设置，单位：pulse
    Done=>x_PresetDone, //TRUE：功能块执行完成
    Busy=>x_PresetBusy, //FALSE：功能块未执行，TRUE：功能块执行中
    CommandAborted=>x_PresetCommandAborted, //TRUE：功能块被终止
    Error=>x_PresetError, //TRUE：功能块内部发生错误
    ErrorID=>ui_PresetErrorID);//错误码，具体参考CC_ERROR
```

功能块操作说明

```
PresetCounter(
    Counter:=LCT_COUNTER_5, //指定使用的计数器名称
    Execute TRUE :=x_PresetExecute TRUE , //上升沿触发，TRUE：使能计数器预置
    Abort FALSE :=x_PresetAbort FALSE , //上升沿触发，TRUE：终止计数器预置
    PresetValue  0  :=di_PresetValue  0 , //预置值，单位：pulse
    PresetMode  0  :=usi_PresetMode  0 , //预置模式，0:软Execute上升沿触发1:比较一致输出时触发2:DI上升沿/下降沿触发3:DI上升沿触发4:DI下降沿触发
    DINumPreset  0  :=usi_DINumPreset  0 , //PresetMode=2,3,4时,指定预置的DI口
    DIPressetLenValid FALSE :=x_DIPresetLenValue FALSE , //PresetMode=2，3，4时,TRUE:启动预置信号长度滤波
    DIPresetLenMin  0  :=di_DIPresetLenMin  0 , //预置信号最小长度设置，单位：pulse
    DIPresetLenMax  0  :=di_DIPresetLenMax  0 , //预置信号最大长度设置，单位：pulse
    Done TRUE =>x_PresetDone TRUE , //TRUE：功能块执行完成
    Busy FALSE =>x_PresetBusy FALSE , //FALSE：功能块未执行，TRUE：功能块执行中
    CommandAborted FALSE =>x_PresetCommandAborted FALSE , //TRUE：功能块被终止
    Error FALSE =>x_PresetError FALSE , //TRUE：功能块内部发生错误
    ErrorID CC_NO_ERRO  =>ui_PresetErrorID CC_NO_ERRO  );//错误码，具体参考CC_ERROR
```

1，“Counter”引脚绑定添加的计数器名称;计数器预置功能块能够实现设定计数器预置PresetCounter的“PresetValue”引脚的值赋值到计数器Counter的“CounterValue”引脚的计数值。

2，“Execute”引脚，默认FALSE，置TRUE上升沿，“PresetValue”引脚当前值赋值到计数器Counter“CounterValue”引脚的值；该功能需在“PresetMode”引脚值为0时实现。

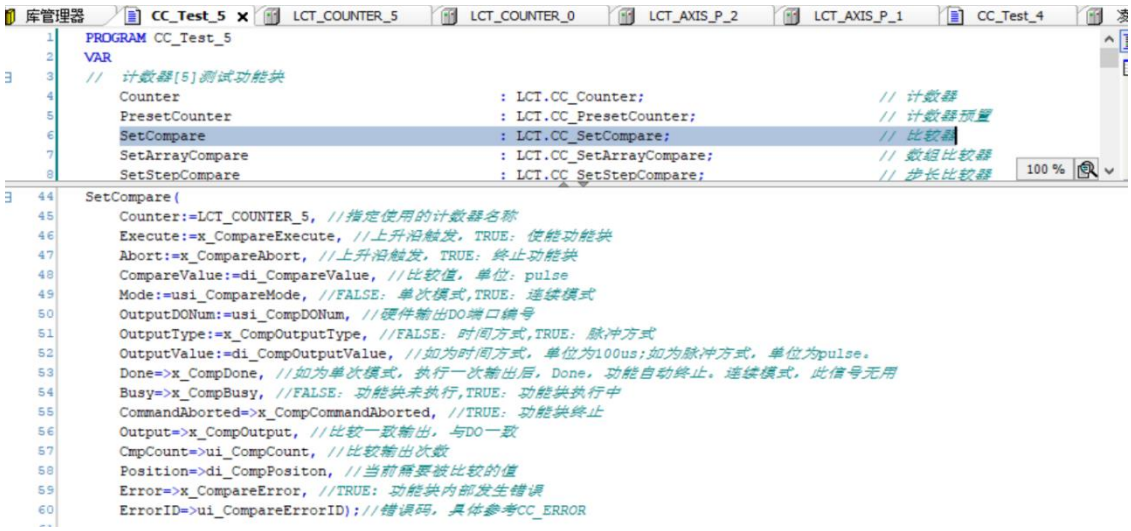
3，“PresetMode”引脚决定计数器预置的模式，0：“Execute”引脚触发，1：比较器PresetCounter比较完成后触发，2：“DINumPreset”引脚设定的端口信号置TRUE上升沿和下降沿各触发一次，3：“DINumPreset”引脚设定的端口信号置TRUE上升沿触发一次，4：“DINumPreset”引脚设定的端口信号置TRUE上升沿触发一次。

4，“DINumPreset”引脚决定预置模式选定DI触发的端口。

5，“DIPressetLenValid”引脚默认FALSE；置TRUE，功能块会过滤掉“DIPresetLenMin”引脚和“DIPresetLenMax”引脚分别设定的预置最大值和最小值外的长度不符高低电平。

6，若“Done”引脚输出表示功能块预置完成，“Execute”引脚需要重新触发上升沿，预置器重启。

声明比较器“SetCompare: LCT.CC_SetCompare;”实例化



```
PROGRAM CC_Test_5
VAR
//  计数器[5]测试功能块
    Counter                 : LCT.CC_Counter;           // 计数器
    PresetCounter           : LCT.CC_PresetCounter;     // 计数器预置
    SetCompare              : LCT.CC_SetCompare;        // 比较器
    SetArrayCompare         : LCT.CC_SetArrayCompare;   // 数组比较器
    SetStepCompare          : LCT.CC_SetStepCompare;    // 步长比较器
```

```
SetCompare(
    Counter:=LCT_COUNTER_5, //指定使用的计数器名称
    Execute:=x_CompareExecute, //上升沿触发，TRUE：使能功能块
    Abort:=x_CompareAbort, //上升沿触发，TRUE：终止功能块
    CompareValue:=di_CompareValue, //比较值，单位：pulse
    Mode:=usi_CompareMode, //FALSE：单次模式,TRUE：连续模式
    OutputDONum:=usi_CompDONum, //硬件输出DO端口编号
    OutputType:=x_CompOutputType, //FALSE：时间方式,TRUE：脉冲方式
    OutputValue:=di_CompOutputValue, //如为时间方式，单位为100us;如为脉冲方式，单位为pulse。
    Done=>x_CompDone, //如为单次模式，执行一次输出后，Done，功能自动终止。连续模式，此信号无用
    Busy=>x_CompBusy, //FALSE：功能块未执行,TRUE：功能块执行中
    CommandAborted=>x_CompCommandAborted, //TRUE：功能块终止
    Output=>x_CompOutput, //比较一致输出，与DO一致
    CmpCount=>ui_CompCount, //比较输出次数
    Position=>di_CompPositon, //当前需要被比较的值
    Error=>x_CompareError, //TRUE：功能块内部发生错误
    ErrorID=>ui_CompareErrorID);//错误码，具体参考CC_ERROR
```

功能块操作说明

```
SetCompare(
    Counter:=LCT_COUNTER_5, //指定使用的计数器名称
    Execute TRUE :=x_CompareExecute TRUE , //上升沿触发，TRUE：使能功能块
    Abort FALSE :=x_CompareAbort FALSE , //上升沿触发，TRUE：终止功能块
    CompareValue 10000 :=di_CompareValue 10000 , //比较值，单位：pulse
    Mode 1 :=usi_CompareMode 1 , //FALSE：单次模式,TRUE：连续模式
    OutputDONum 5 :=usi_CompDONum 5 , //硬件输出DO端口编号
    OutputType FALSE :=x_CompOutputType FALSE , //FALSE：时间方式,TRUE：脉冲方式
    OutputValue 8 :=di_CompOutputValue 8 , //如为时间方式，单位为100us;如为脉冲方式，单位为pulse。
    Done FALSE =>x_CompDone FALSE , //如为单次模式，执行一次输出后，Done，功能自动终止。连续模式，此信号无用
    Busy TRUE =>x_CompBusy TRUE , //FALSE：功能块未执行,TRUE：功能块执行中
    CommandAborted FALSE =>x_CompCommandAborted FALSE , //TRUE：功能块终止
    Output FALSE =>x_CompOutput FALSE , //比较一致输出，与DO一致
    CmpCount 0 =>ui_CompCount 0 , //比较输出次数
    Position 33564432 =>di_CompPositon 33564432 , //当前需要被比较的值
    Error FALSE =>x_CompareError FALSE , //TRUE：功能块内部发生错误
    ErrorID CC_NO_ERRO =>ui_CompareErrorID CC_NO_ERRO );//错误码，具体参考CC_ERROR
```

1，“Counter”引脚绑定添加的计数器名称;比较器功能块能够实现设定比较器SetCompar的“CompareValue”引脚的设定值与计数器Counter的“CounterValue”引脚的计数值比较，若相等则完成比较。

2，“Mode”引脚决定比较器比较模式；0：单次模式，比较器在引脚“Execute”上升沿触发后，执行一次比较，完成后功能块结束，“OutputDONum”引脚设定的端口和“Output”引脚输出，“Done”引脚置TRUE。1：循环模式，比较器比较器在引脚“Execute”上升沿触发后一直执行比较，完成后“OutputDONum”引脚设定的端口和“Output”引脚输出，“Done”不变，“Abort”引脚置TRUE后功能块比较停止。

3，“CmpCount”引脚显示比较完成次数。

4，“Position”引脚显示当前需要比较的设定值。

5，“OutputType”引脚可以设定输出。

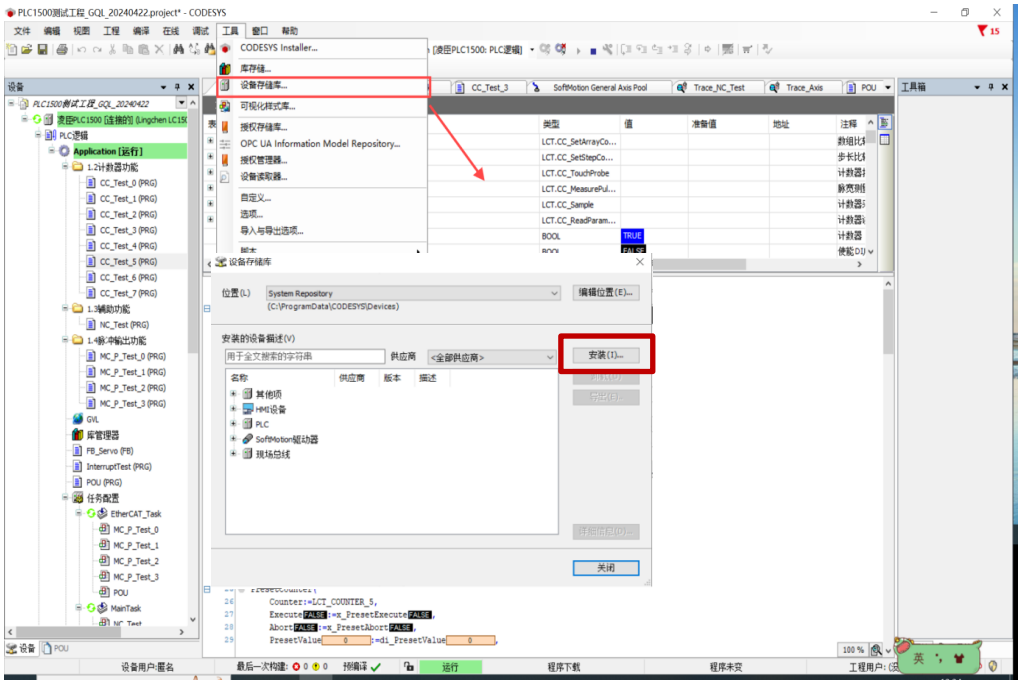
以上为个别功能块示例，其他计数器功能块还请查阅《高速IO硬件功能说明书》

8.5 高速脉冲输出

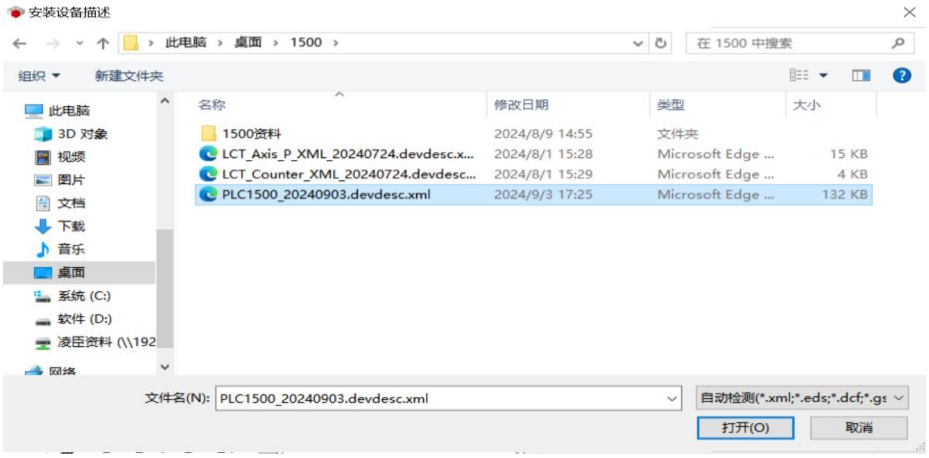
8.5.1 XML文件安装及应用

新建Codesys工程，设备选择Lingchen LC1500。

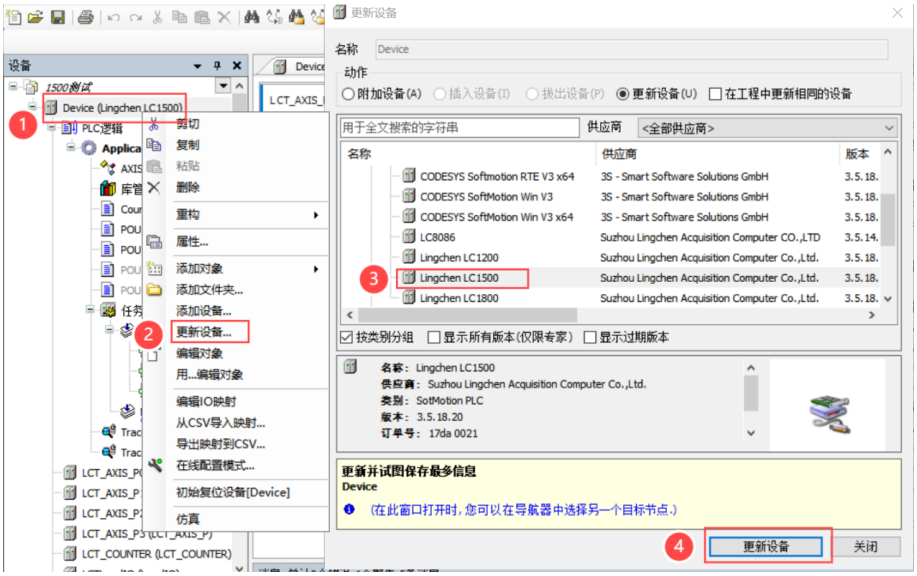
如果没有Lingchen LC1500选项，先安装Lingchen LC1500 XML文件，然后更新设备。



选择“PLC1500_20240903.devdesc”XML文件，点击打开进行安装。

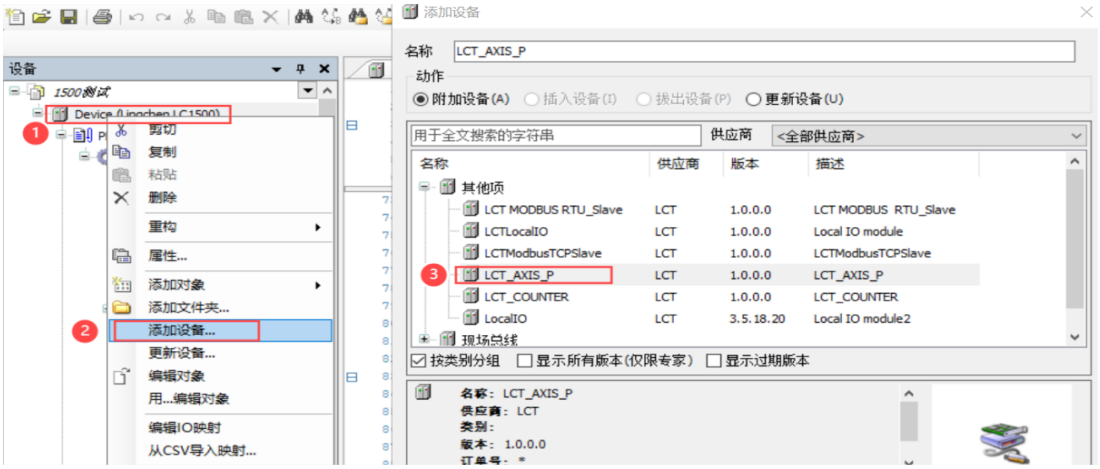


安装完成后，右击Device，点击更新设备，选择Lingchen LC1500，更新设备即可。



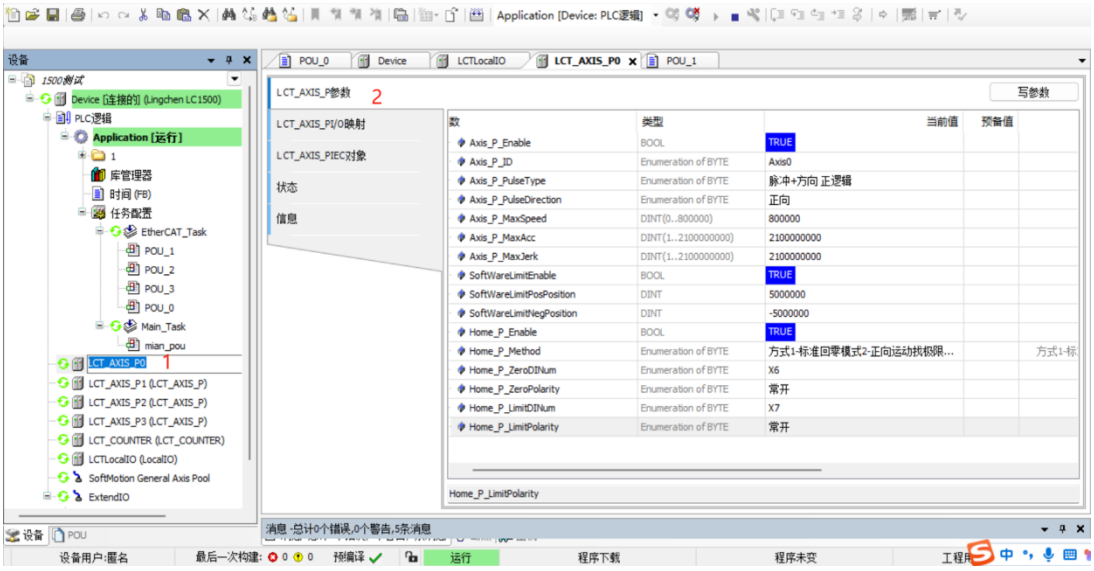
然后以同样的方式安装“LCT_Axis_P_XML”XML文件。

安装完成后，添加高速脉冲轴，右击Device，单击添加设备，在设备列表中选择LCT_AXIS_P。



添加完成后，双击已添加的轴进入参数设置

轴ID根据接线选择Axis0、Axis1、Axis2、Axis3，PLC设置脉冲类型选择与驱动器一致，这里选用脉冲+方向控制正逻辑，回零方式按需求选择，原点DI、极限DI按接线选择、软限位设置。



8.5.2 参数说明

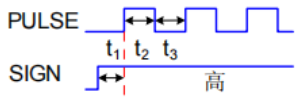
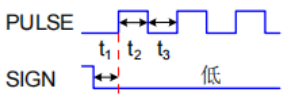
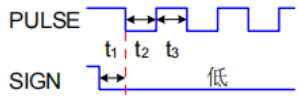
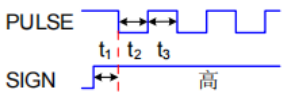
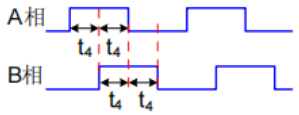
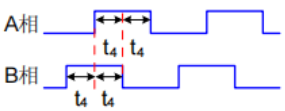
打开轴参数配置页面，可对轴参数进行配置。

参数	类型	值	默认值	单元	描述
Axis_P_Enable	BOOL	TRUE	FALSE		Axis_P_Enable
Axis_P_ID	Enumeration of BYTE	Axis0	Axis0		Axis_P_ID
Axis_P_PulseType	Enumeration of BYTE	脉冲+方向 正逻辑	AB相四倍频脉冲		Axis_P_PulseType
Axis_P_PulseDirection	Enumeration of BYTE	正向	正向		Axis_P_PulseDirection
Axis_P_MaxSpeed	DINT(0..800000)	1000000	1000000		Axis_P_MaxSpeed
Axis_P_MaxAcc	DINT(1..2100000000)	1000000	1000000		Axis_P_MaxAcc
Axis_P_MaxJerk	DINT(1..2100000000)	10000000	10000000		Axis_P_MaxJerk
SoftWareLimitEnable	BOOL	TRUE	FALSE		SoftWareLimitEnable
SoftWareLimitPosPosition	DINT	2147483647	2147483647		SoftWareLimitPosPosition
SoftWareLimitNegPosition	DINT	-2147483648	-2147483648		SoftWareLimitNegPosition
Home_P_Enable	BOOL	TRUE	FALSE		Home_P_Enable
Home_P_Method	Enumeration of BYTE	方式3-标准回零模式18-正向运动找极限	方式8-标准回...		Home_P_Method
Home_P_ZeroDINum	Enumeration of BYTE	X6	X0		Home_P_ZeroDINum
Home_P_ZeroPolarity	Enumeration of BYTE	常开	常开		Home_P_ZeroPolarity
Home_P_LimitDINum	Enumeration of BYTE	X7	X0		Home_P_LimitDINum
Home_P_LimitPolarity	Enumeration of BYTE	常开	常开		Home_P_LimitPolarity

1. 轴使能：Axis_P_Enable设为TRUE。

2. 轴ID：根据接线可选：AXIS0：DO0为脉冲信号，DO1为方向信号；AXIS1：DO2为脉冲信号，DO3为方向信号；AXIS2：DO4为脉冲信号，DO5为方向信号；AXIS3：DO6为脉冲信号，DO7为方向信号，脉冲＋方向控制支持正逻辑和负逻辑；AB相四倍频脉冲：AXIS0：DO0为A相，DO1为B相；AXIS1：DO2为A相，DO3为B相；AXIS2：DO4为A相，DO5为B相；AXIS3：DO6为A相，DO7为B相。

3. 脉冲类型：脉冲+方向（正逻辑或负逻辑）；A/B相正交脉冲4倍频。脉冲类型选择与驱动器设置一致。

脉冲形态	信号	正转脉冲示意图	反转脉冲示意图
脉冲＋方向 正逻辑	PULSE SIGN		
脉冲＋方向 负逻辑	PULSE SIGN		
A 相 +B 相 正交脉冲 4 倍频	PULSE(A 相) SIGN(B 相)	 A 相超前 B 相 90°	 B 相超前 A 相 90°

4. 方向：TRUE：方向取反，FALSE：正常方向。
5. 最大速度：速度允许设置的最大值，最大1000000，单位pulse。
6. 最大加速度：加速度允许设置的最大值，最大2100000000，单位pulse。
7. 最大加加速度：加加速度允许设置的最大值，最大2100000000，单位pulse。
8. 使能正向软限位：TRUE：启用软限位，FALSE：不启用软限位。
9. 使能负向软限位：TRUE：启用软限位，FALSE：不启用软限位。
10. 回零方式：见MC_Home_P功能块说明。
11. 原点DI：根据原点开关接线选择X0~X7。
12. 原点DI极性：根据原点开关极性选择常开常闭。
13. 回零极限DI：根据极限开关接线选择X0~X7。
14. 回零极限DI极性：根据极限开关极性选择常开常闭。

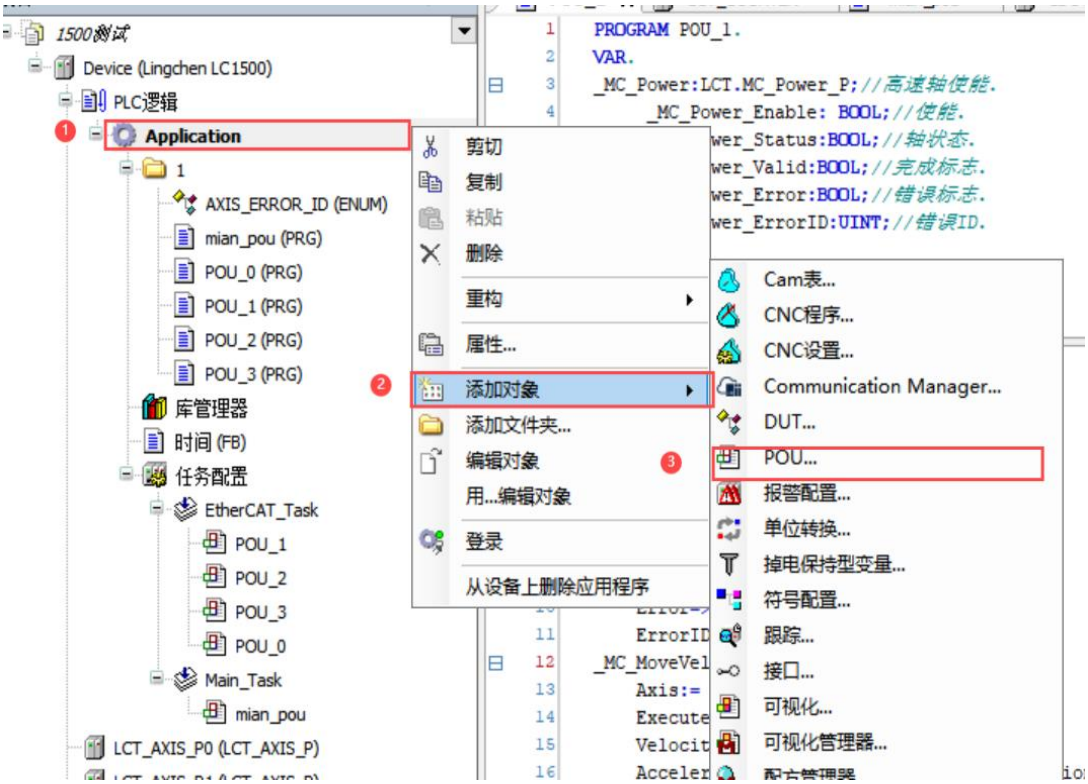
功能块指令列表：

序号	指令名称	FB/FC	功能
1.4.1	MC_Power_P	FB	高速轴使能
1.4.2	MC_Reset_P	FB	高速轴复位
1.4.3	MC_MoveAbsolute_P	FB	高速轴绝对定位控制
1.4.4	MC_MoveRelative_P	FB	高速轴相对定位控制
1.4.5	MC_MoveVelocity_P	FB	高速轴速度控制
1.4.6	MC_Home_P	FB	高速轴回零
1.4.7	MC_Stop_P	FB	高速轴停止
1.4.8	MC_Jog_P	FB	高速轴点动
1.4.9	MC_SetPosition_P	FB	高速轴位置设定
1.4.10	MC_ReadParameter_P	FB	高速轴读参数
1.4.11	MC_WriteParameter_P	FB	高速轴写参数
1.4.12	MC_ReadStatus_P	FB	高速轴读轴状态
1.4.13	MC_ReadMotionStatus_P	FB	高速轴读轴运动状态

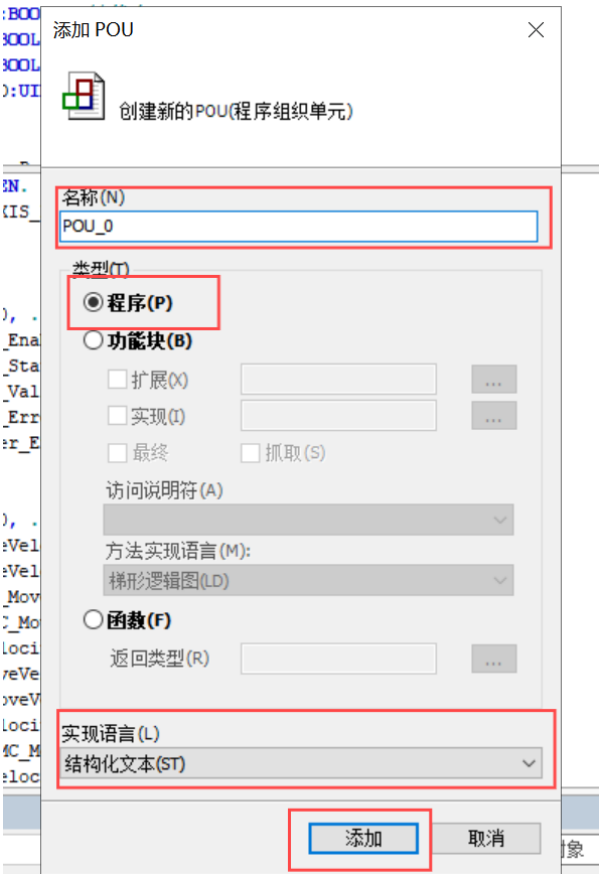
以上脉冲输出功能块是常用功能块，具体功能说明请参考“高速IO说明书”

8.5.3 创建POU程序

右击“Application”,点击“添加对象”，选择“POU”后会跳出添加页面。

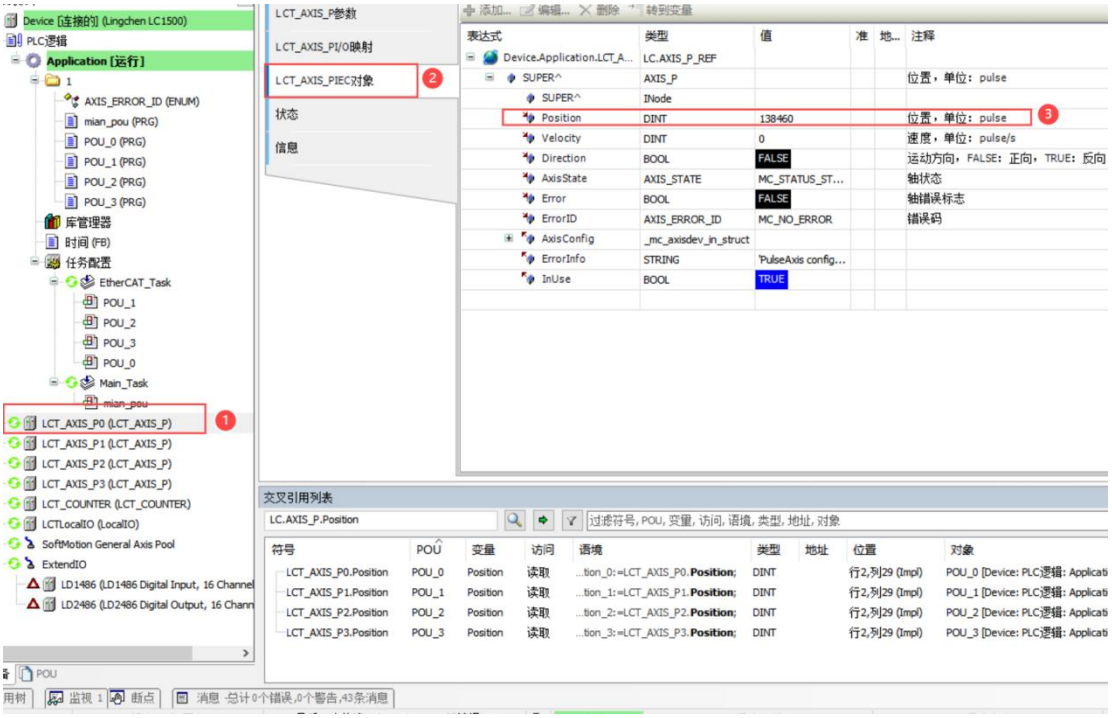


起定名称后选择“程序”，这里先选择“结构化文本”语言，最后点击“添加 即可。



8.5.4 获取脉冲轴位置

获取脉冲轴实际运行位置数据映射到程序中，方便本次测试观察，以轴“LCT_AXIS_P0”为例：如下图，程序登录进PLC，点击“LCT_AXIS_P0”脉冲轴，选择“LCT_AXIS_PIEC对象”，打开表达式列表，其中的“Position”便是此轴的实际轴位置，此外还有轴状态、轴错误码等轴信息，可根据需求应用到程序POU中。

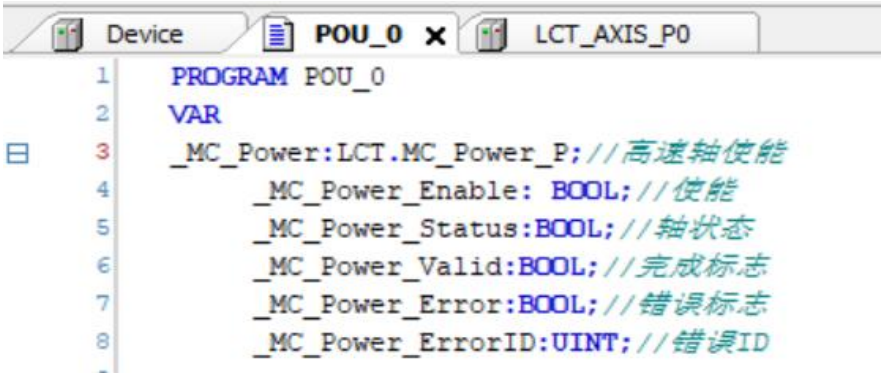


在刚建立的POU中定义个轴位置变量如下图“1”步，将上面所找到的轴位置赋值到定义的变量上，如下图“2”步，其中“LCT_AXIS_P0”是轴名称，“Position”是指次轴的位置。

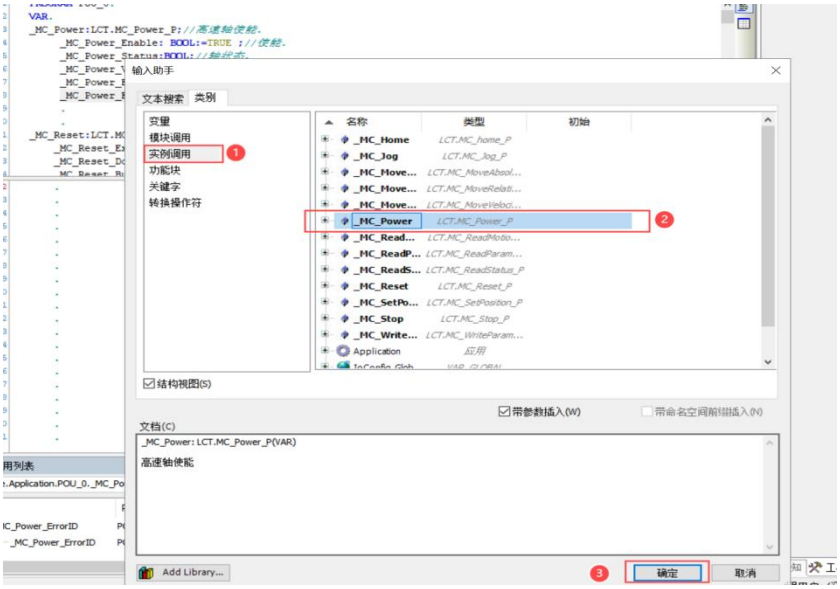
```
164     X8  AT %QX1.0:BOOL;//原点DI.
165     X9  AT %QX1.1:BOOL;//限位DI.
166     Axisposition_0:DINT;//轴位置.
167 END_VAR.

1   IF _MC_Power_Valid THEN.
2   Axisposition_0:=LCT_AXIS_P0.Position;.
3   END_IF.
4   //轴使能.
5   _MC_Power( .
6       Axis:= LCT_AXIS_P0, .
7       Enable:=_MC_Power_Enable, .
8       Status=>_MC_Power_Status , .
9       Valid=> _MC_Power_Valid , .
10      Error=> _MC_Power_Error , .
11      ErrorID=> _MC_Power_ErrorID);.
```

8.5.5 功能块的POU应用

先在POU中声明功能块使其实例化，以“MC_Power_P”为例，并声明出功能块所用引角的变量，如下图：



将实例化好的模块调用到程序中，在下面程序编辑中按“F2”按钮打开输入助手页面，点击实例调用，选择刚声明好的“_MC_Power_P”，点击“确定”即可。



调用好后如下，然后绑定上面定义好的各功能变量。功能块操作说明：1.Axis引脚绑定添加的轴名称；2.Enable使能轴功能块，高电平有效，Status和Valid置TRUE，功能块使能完成；3.此功能块为使能高速脉冲输出指令，如果此功能块轴使能没有完成，其他所以功能块都无法运行。

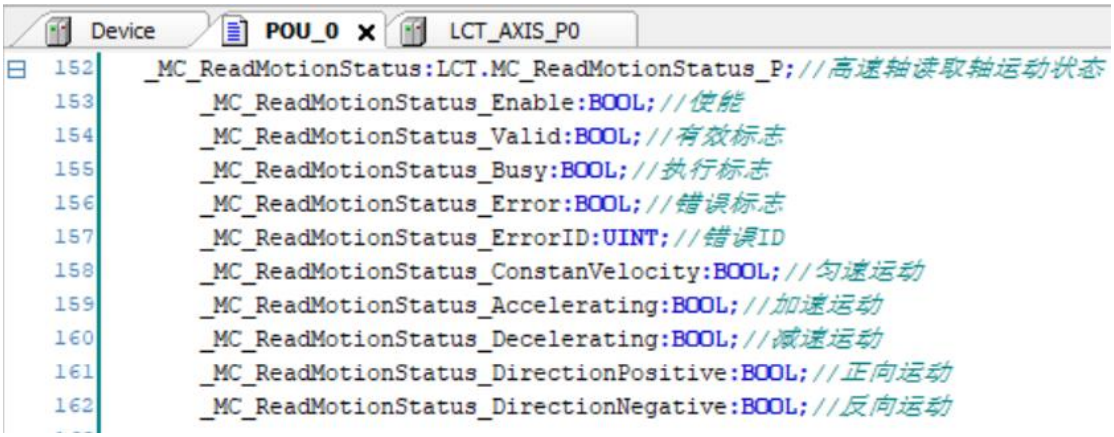
```
_MC_Power(
    Axis:= LCT_AXIS_P0, //轴名称
    Enable:=_MC_Power_Enable, //使能，电平触发，TRUE：使能功能块
    Status=>_MC_Power_Status , //轴状态，TRUE：高速轴使能
    Valid=> _MC_Power_Valid , //完成标志，TRUE：功能块有效
    Error=> _MC_Power_Error , //错误标志，TRUE：功能块内部发生错误
    ErrorID=> _MC_Power_ErrorID);//错误码，参考轴错误ID说明
```

按此方法可声明调用所需的其他功能块。

8.5.6 MC_ReadMotionStatus_P的应用

本功能块主要实现高速轴读取轴运动状态功能，便于测试观察轴的运动。

以轴“LCT_AXIS_P”为例，声明高速轴功能块“_MC_ReadMotionStatus:LCT.MC_ReadMotionStatus_P”实例化，并定义出用于功能引角的变量。

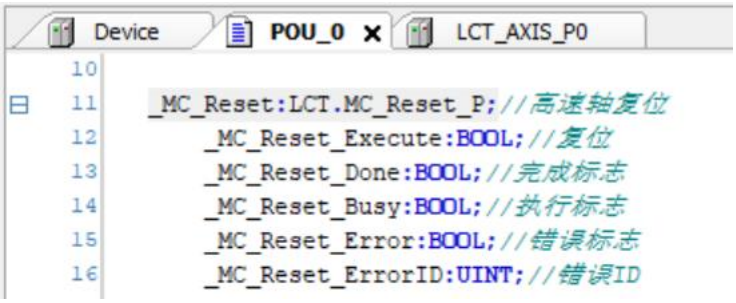


此功能块操作说明：1.Axis引脚绑定添加的轴名称；2.Enable使能轴功能块，高电平有效，Valid置TRUE，功能块使能完成；3.Busy引脚，执行标志，置TRUE，功能块正在执行；4.ConstanVelocity引脚，匀速运动，轴匀速运动时置TRUE；5.Accelerating引脚，加速运动，轴加速运动时置TRUE；6.Decelerating引脚，减速运动，轴减速运动时置TRUE；7.DirectionPositive引脚，正向运动，轴正向运动时置TRUE；8.DirectionNegative引脚，反向运动，轴反向运动时置TRUE。

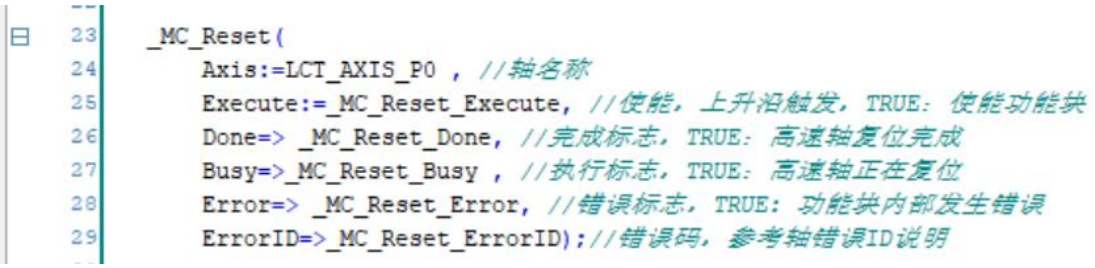
8.5.7 MC_Reset_P的应用

本功能块主要实现高速轴错误复位操作。

声明高速轴功能块“_MC_Reset:LCT.MC_Reset_P;”实例化并定义出用于功能引角的变量。



此模块功能块操作说明：1.Axis引脚绑定添加的轴名称；2.Execute引脚，默认FALSE，置TRUE上升沿使能功能块；3.Done引脚，完成标志，置TRUE，高速轴复位完成；4.Busy引脚，执行标志，置TRUE，高速轴正在复位。



8.5.8 MC_MoveAbsolute_P的应用

本功能块主要实现高速轴轴绝对定位控制，Position 数据为轴的绝对位置。加减速过程中不允许改变速度。

声明高速轴功能块“_MC_MoveAbsolute:LCT.MC_MoveAbsolute_P;”实例化，并定义出用于功能引角的变量。

Device | POU_0 | LCT_AXIS_P0

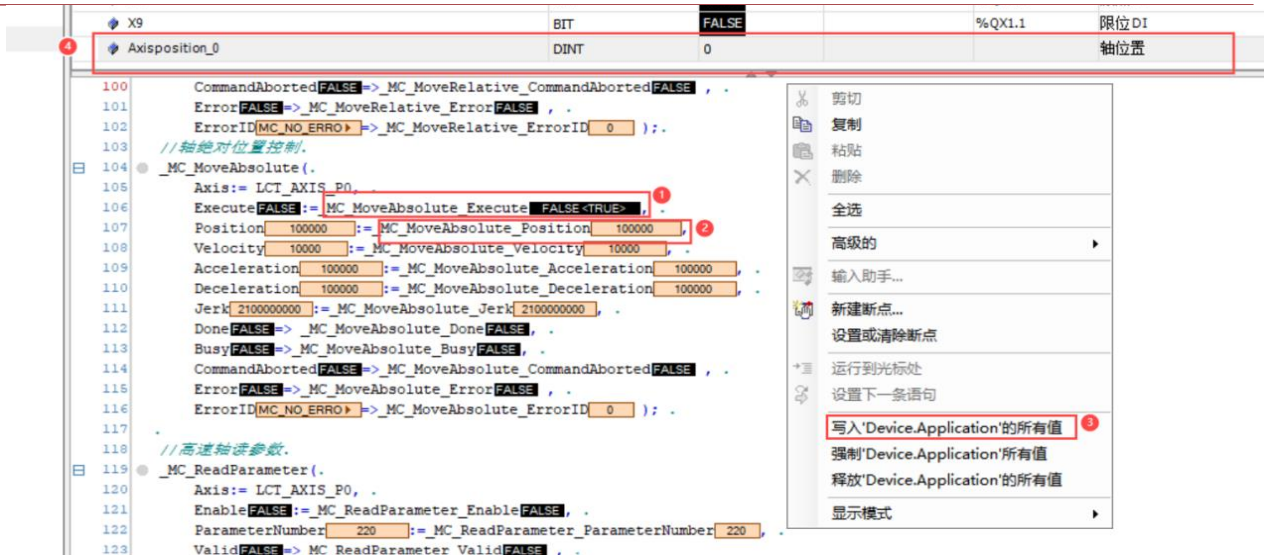
```
18
19    _MC_MoveAbsolute:LCT.MC_MoveAbsolute_P;//高速轴绝对位置控制
20        _MC_MoveAbsolute_Execute:BOOL;//使能
21        _MC_MoveAbsolute_Position:DINT:=100000;//绝对位置
22        _MC_MoveAbsolute_Velocity:UDINT:=10000;//速度
23        _MC_MoveAbsolute_Acceleration:UDINT:=100000;//加速度
24        _MC_MoveAbsolute_Deceleration:UDINT:=100000;//减速度
25        _MC_MoveAbsolute_Jerk:UDINT:=2100000000;//加加速度
26        _MC_MoveAbsolute_Done:BOOL;//完成标志
27        _MC_MoveAbsolute_Busy:BOOL;//执行标志
28        _MC_MoveAbsolute_CommandAborted:BOOL;//终止标志
29        _MC_MoveAbsolute_Error:BOOL;//错误标志
30        _MC_MoveAbsolute_ErrorID:UINT;//错误ID
```

此功能块操作说明：1.Axis引脚绑定添加的轴名称；2.Execute引脚，默认FALSE，置TRUE上升沿使能功能块；3.Position引脚，轴运动目标绝对位置；4.Done引脚，完成标志，置TRUE，功能块执行完成；5.Busy引脚，执行标志，置TRUE，功能块正在执行；6.CommandAborted引脚，中止标志，功能块执行过程中被打断时置TRUE。

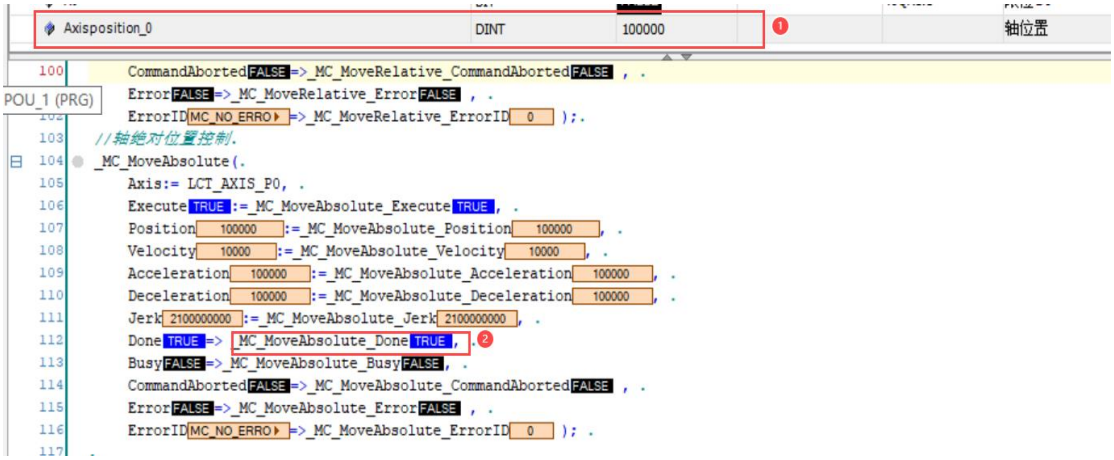
```
97    _MC_MoveAbsolute(
98        Axis:= LCT_AXIS_P0, //轴名称
99        Execute:=_MC_MoveAbsolute_Execute, //使能，上升沿触发，TRUE：使能功能块
100       Position:=_MC_MoveAbsolute_Position, //绝对位置，单位：pulse
101       Velocity:=_MC_MoveAbsolute_Velocity, //定位速度，单位：pulse/s
102       Acceleration:=_MC_MoveAbsolute_Acceleration, //加速度，单位：pulse/s²
103       Deceleration:=_MC_MoveAbsolute_Deceleration, //减速度，单位：pulse/s²
104       Jerk:=_MC_MoveAbsolute_Jerk, //加加速度，单位：pulse/s³
105       Done=> _MC_MoveAbsolute_Done, //完成标志，TRUE：功能块执行完成
106       Busy=>_MC_MoveAbsolute_Busy, //执行标志，TRUE：功能块正在执行
107       CommandAborted=>_MC_MoveAbsolute_CommandAborted , //终止标志，TRUE：功能块被终止
108       Error=>_MC_MoveAbsolute_Error , //错误标志，TRUE：功能块内部发生错误
109       ErrorID=>_MC_MoveAbsolute_ErrorID );//错误码，参考轴错误ID说明
```

【绝对定位调试实例】

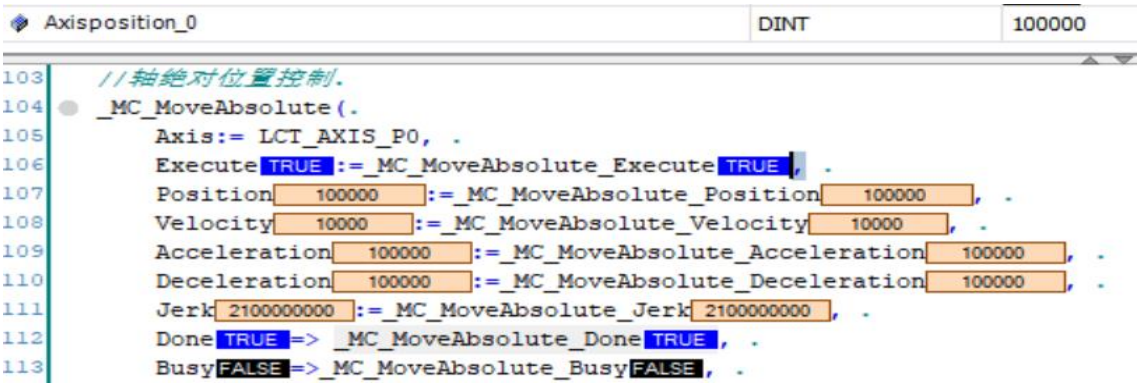
以轴“LCT_AXIS_P0”的为实例如下所示：当轴使能模块使能完成后，此时轴实际位置“Axisposition_0”为0，绝对位置“MC_MoveAbsolute_Posit”设为100000，设置好速度值后，点击绝对位置触发“_MC_MoveAbsolute_Execute”预值<TRUE>，后右击选择写入其所有值，步“3”或者按“Ctrl+F7”。



绝对位置触发后，电机以1000pul/s的速度开始运行至_MC_MoveAbsolute_Done为TRUE时停止，此时测试轴位置Axisposition_0为100000，正好等于设置值。



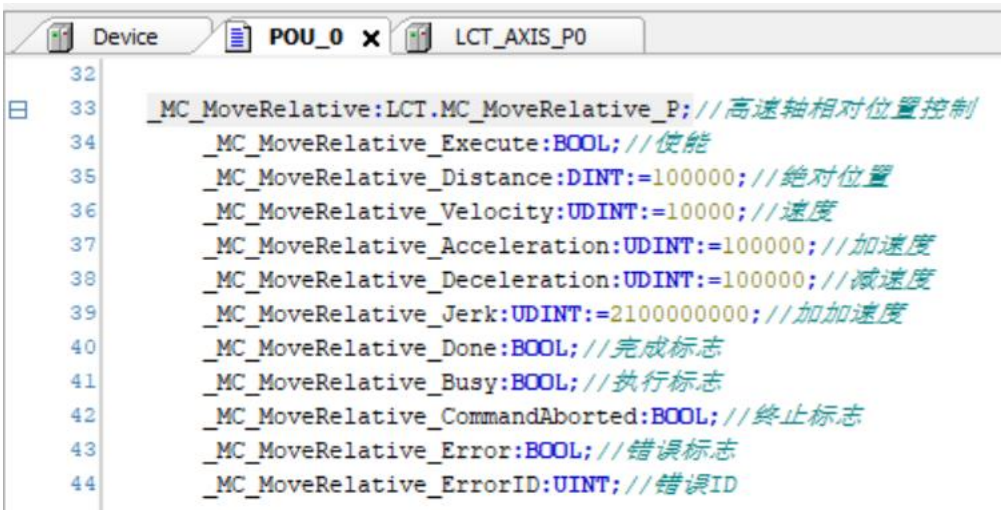
在设置值为100000不变情况下，再次触发“_MC_MoveAbsolute_Execute”，给其上值TRUE后，轴位置仍然为100000等于设置值，电机也处在当前位置不会运动。



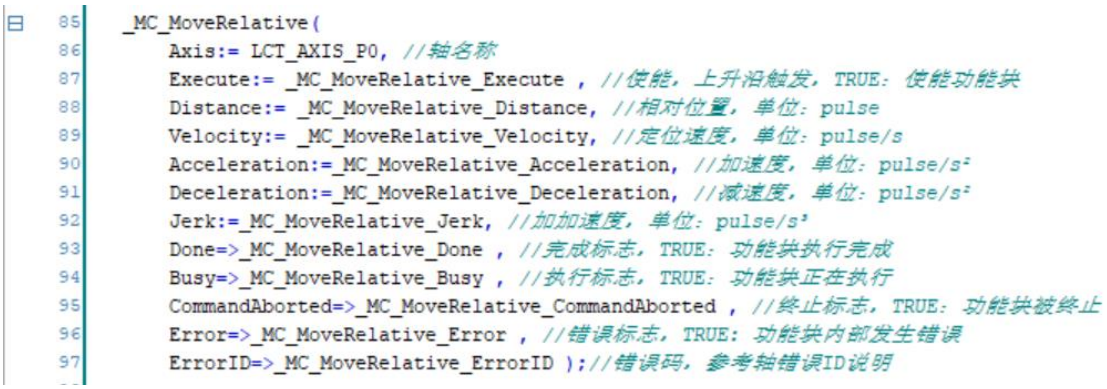
8.5.9 MC_MoveRelative_P 的应用

本功能块主要实现高速轴轴相对定位控制，Distance数据为轴的相对位置。加减速过程中不允许改变速度 。

声明高速轴功能块“_MC_MoveRelative:LCT.MC_MoveRelative_P;”实例化，并定义出用于功能引角的变量。

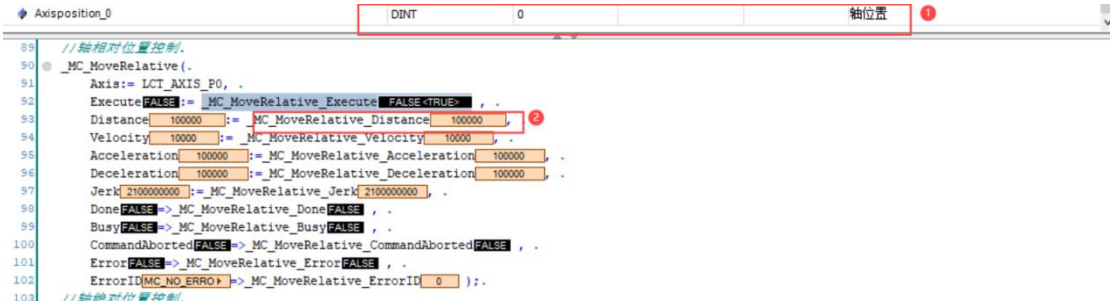


此功能块操作说明：1.Axis引脚绑定添加的轴名称；2.Execute引脚，默认FALSE，置TRUE上升沿使能功能块；3.Distance引脚，轴运动相对位置，即相对于当前位置轴运行位置；4.Done引脚，完成标志，置TRUE，功能块执行完成；5.Busy引脚，执行标志，置TRUE，功能块正在执行；6.CommandAborted引脚，中止标志，功能块执行过程中被打断时置TRUE。

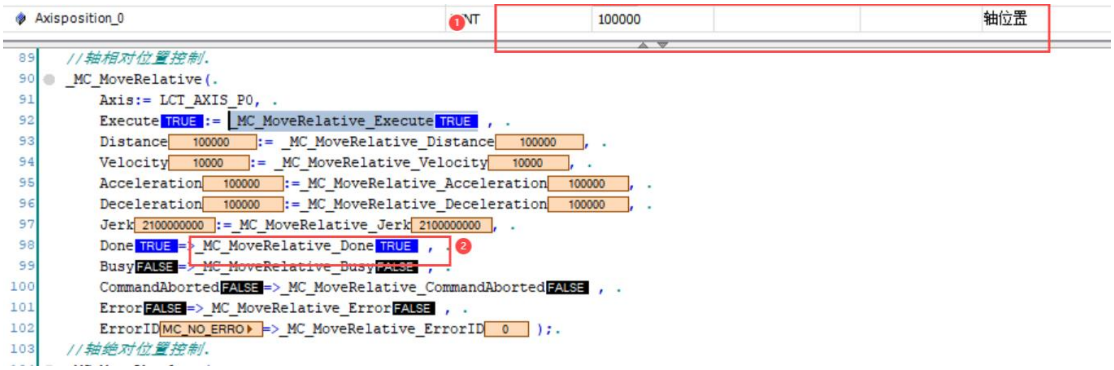


【相对定位调试实例】

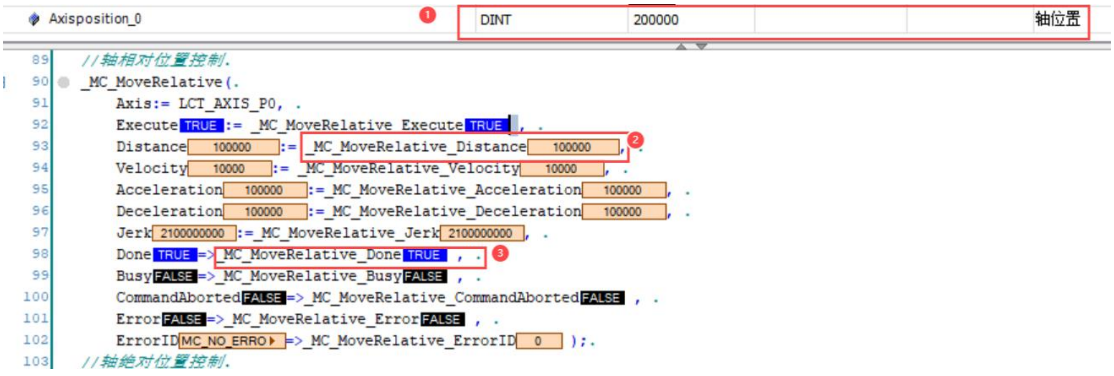
以轴“LCT_AXIS_P0”的为实例如下所示：当轴使能模块使能完成后，轴实际位置“Axisposition_0”为0时，相对位置值_MC_MoveRelative_Distance设为100000，触发_MC_MoveRelative_Execute启动模块运行，同时所接的电机开始运动。



_MC_MoveRelative_Done亮时相对运动结束，电机也停止运动，此时轴位置等于设置值100000。



保持_MC_MoveRelative_Distance值为100000不变，再次触发MC_MoveRelative_Execute,模块程序和电机再次运行至轴位置为200000时停止，可见此功能块不同与绝对运动，而是可以再当前位置上不断做相对运动。



8.5.10 MC_SetPosition_P 的应用

本功能块主要实现高速轴位置设定功能。当Relative=FALSE时设置的位置为绝对位置，当Relative=FALSE时设置的位置为绝对位置，即当前位置=Position；当Relative=TRUE时设置的位置为相对位置，即当前位置=当前位置+Position。

声明计数器“_MC_SetPosition:LCT.MC_SetPosition_P;”实例化，并定义出用于功能引角的变量 ：

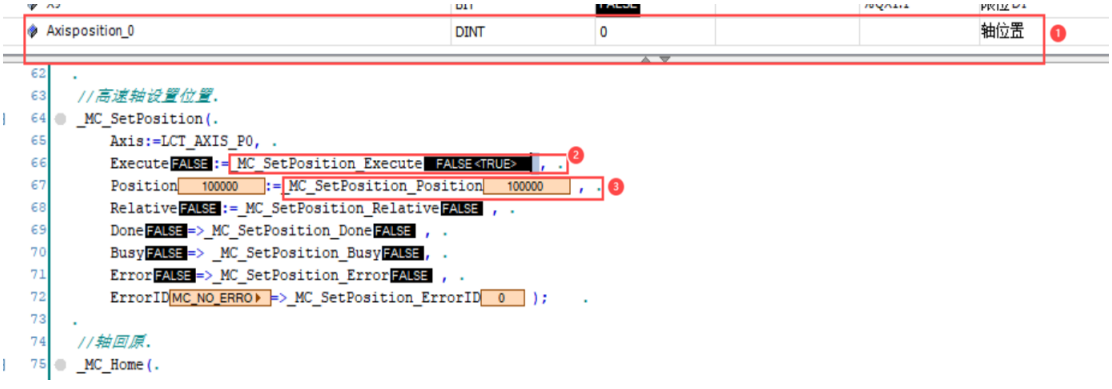
```
102  _MC_SetPosition:LCT.MC_SetPosition_P;//高速轴设置位置
103      _MC_SetPosition_Execute:BOOL;//使能
104      _MC_SetPosition_Position:DINT:=100000;//位置
105      _MC_SetPosition_Relative:BOOL;//相对位置
106      _MC_SetPosition_Done:BOOL;//完成标志
107      _MC_SetPosition_Busy:BOOL;//执行标志
108      _MC_SetPosition_Error:BOOL;//错误标志
109      _MC_SetPosition_ErrorID:UINT;//错误ID
```

此功能块操作说明：1.Axis引脚绑定添加的轴名称；2.Execute引脚，默认FALSE，置TRUE上升沿使能功能块；3.Position引脚，设定位置；4.Relative引脚，默认FALSE，绝对位置，触发时将轴位置设置为Position所设值，置TRUE，相对位置，触发时将轴位置设置为当前位置+Position所设值；5.Done引脚，完成标志，置TRUE，功能块执行完成；6.Busy引脚，执行标志，置TRUE，功能块正在执行。

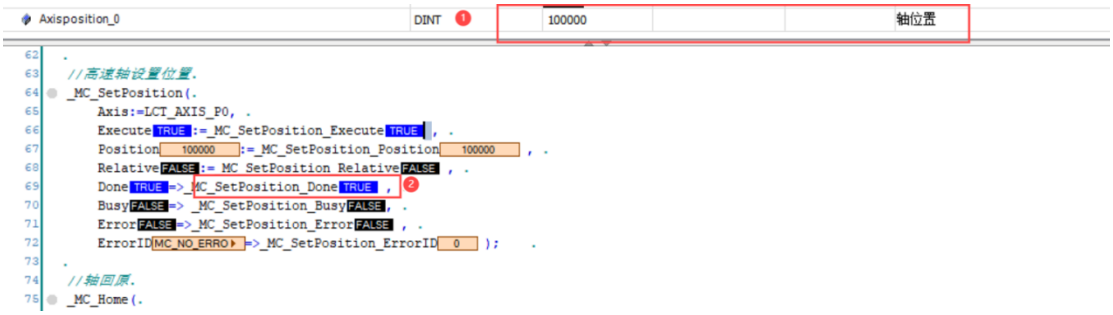
```
59  _MC_SetPosition(
60      Axis:=LCT_AXIS_P0, //轴名称
61      Execute:=_MC_SetPosition_Execute , //使能，上升沿触发，TRUE：使能功能块
62      Position:=_MC_SetPosition_Position , //位置，单位：pulse
63      Relative:=_MC_SetPosition_Relative , //相对位置，FALSE：绝对位置，TRUE：相对位置
64      Done=>_MC_SetPosition_Done , //完成标志，TRUE：功能块执行完成
65      Busy=> _MC_SetPosition_Busy, //执行标志，TRUE：功能块正在执行
66      Error=>_MC_SetPosition_Error , //错误标志，TRUE：功能块内部发生错误
67      ErrorID=>_MC_SetPosition_ErrorID ); //错误码，参考轴错误ID说明
```

【高速轴设置位置调试实例】

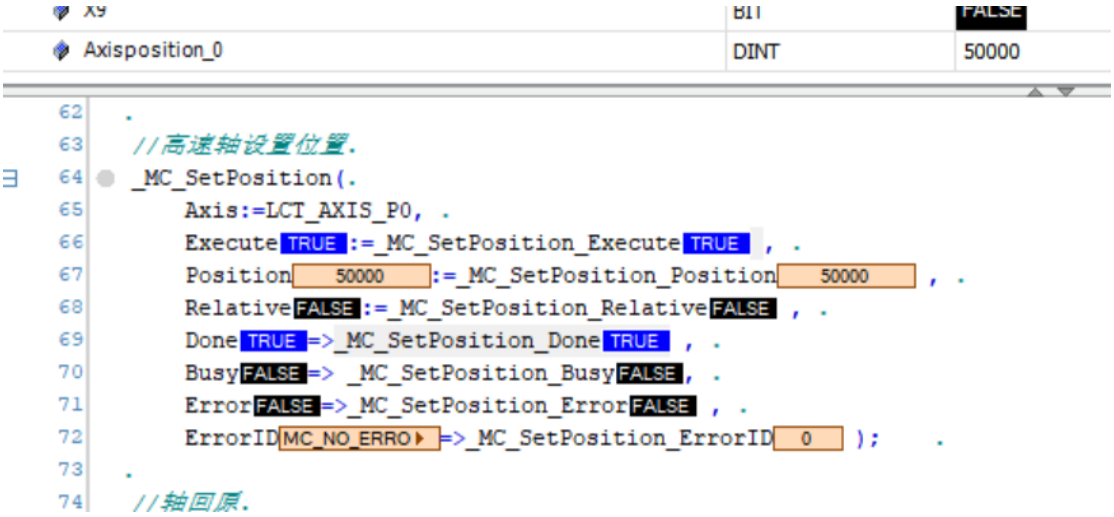
以轴“LCT_AXIS_P0”的为实例如下所示：当轴使能模块使能完成后，此时轴实际位置Axisposition_0为0，设置位值_MC_SetPosition_Position设为100000，触发_MC_SetPosition_Execute后功能模块运行，但电机不会运动。



_MC_SetPosition_Done完成标志亮起，模块运行结束，此时实际位置为100000。



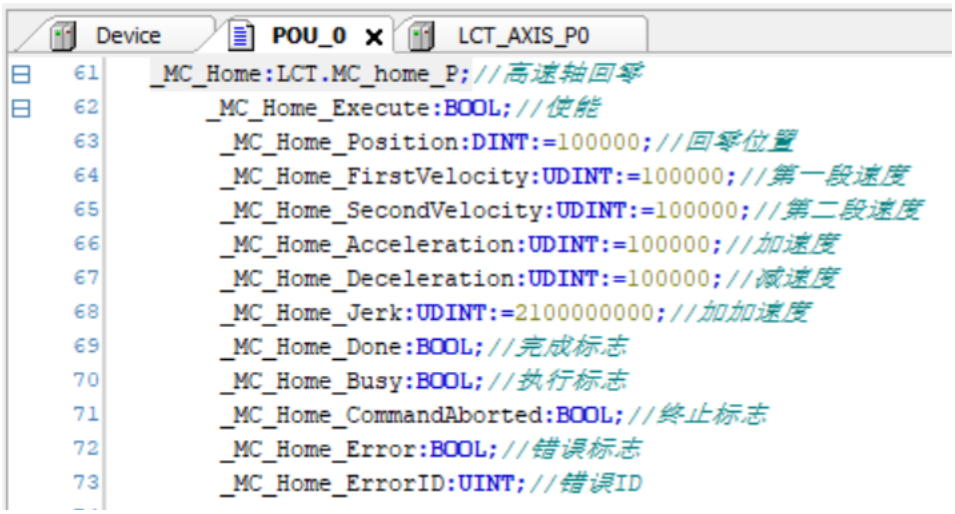
再把设置位置改为50000，重新触发_MC_SetPosition_Execute后，轴实际位置Axisposition_0变为设置值50000，由此可见，此功能可设置改变标定轴位置值，但不需要电机运动。



8.5.11 MC_Home_P的应用

本功能块为使轴回零点功能块，LC1500目前配有9种回零方式，具体功能描述请参考另一份“高速IO”说明书。

声明高速轴功能块“_MC_Home:LCT.MC_home_P;”实例化，并定义出用于功能引角的变量：



此功能块操作说明：1.Axis引脚绑定添加的轴名称；2.Execute引脚，默认FALSE，置TRUE上升沿使能功能块；3.Position引脚，轴回零位置；4.FirstVelocity引脚，第一段速度；5.SecondVelocity引脚，第二段速度；6.Done引脚，完成标志，置TRUE，功能块执行完成；Busy引脚，执行标志，置TRUE，功能块正在执行；7.CommandAborted引脚，中止标志，功能块执行过程中被打断时置TRUE。

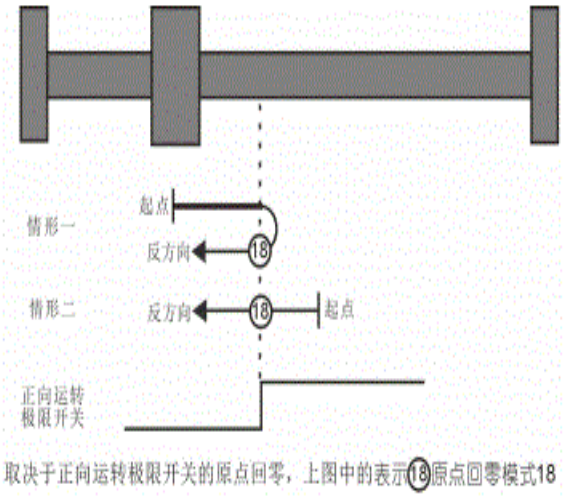
```
71  _MC_Home(
72      Axis:= LCT_AXIS_P0, //轴名称
73      Execute:= _MC_Home_Execute, //使能，上升沿触发，TRUE：使能功能块
74      Position:=_MC_Home_Position, //回零位置，单位：pulse
75      FirstVelocity:=_MC_Home_FirstVelocity, //第一段速度，单位：pulse/s
76      SecondVelocity:=_MC_Home_SecondVelocity , //第二段速度，单位：pulse/s
77      Acceleration:= _MC_Home_Acceleration, //加速度，单位：pulse/s²
78      Deceleration:=_MC_Home_Deceleration, //减速度，单位：pulse/s²
79      Jerk:= _MC_Home_Jerk, //加加速度，单位：pulse/s³
80      Done=>_MC_Home_Done , //完成标志，TRUE：功能块执行完成
81      Busy=>_MC_Home_Busy , //执行标志，TRUE：功能块正在执行
82      CommandAborted=>_MC_Home_CommandAborted , //终止标志，TRUE：功能块被终止
83      Error=>_MC_Home_Error , //错误标志，TRUE：功能块内部发生错误
84      ErrorID=>_MC_Home_ErrorID );//错误码，参考轴错误ID说明
```

【 高速轴轴回零调试实例】

以回零方式3（DS402协议标注回零方式18）：正向运动找极限，即正向运动找极限的下降沿为例子，此回原方式功能说明：

情形一：当执行回零时，若正向极限开关状态处于低电平时，那么轴开始以第一段速度正向运动，当遇正向极限开关处于高电平时，轴运动方向改变且以第二段速度开始运动；在正向极限开关状态处于低电平时的位置就是原点位置。

情形二：当执行回零时，若正向极限开关处于高电平时，那么轴直接会以第二段速度开始反向运动，在正向极限开关状态处于低电平时的位置就是原点位置。



取决于正向运转极限开关的原点回零，上图中的表示18原点回零模式18

应用调试操作：以轴“LCT_AXIS_P0”的为例如下所示：打开轴参数配置，回原使能设TRUE，选择回原方式18，选X7为极限位。

SoftwareLimitNegPosition	DINT	-5000000
Home_P_Enable	BOOL	TRUE
Home_P_Method	Enumeration of BYTE	方式3-标准回零模式18-正向运动找极限
Home_P_ZeroDINum	Enumeration of BYTE	X6
Home_P_ZeroPolarity	Enumeration of BYTE	常开
Home_P_LimitDINum	Enumeration of BYTE	X7
Home_P_LimitPolarity	Enumeration of BYTE	常开

情形一：在限位处于低电平“1”步、轴位置为0“2”步时，分别设回原第一速度“3”步和第二速度“4”步的值，再触发_MC_Home_Execute开启回原运动。

X6		BIT	FALSE	%QX1.0	原点DI
X7	1	BIT	FALSE	%QX1.1	限位DI
Axisposition_0	2	DINT	0		轴位置

```
62  //轴回原.
63  _MC_Home(.
64      Axis:= LCT_AXIS_P0, .
65      Execute FALSE := _MC_Home_Execute FALSE <TRUE> , .
66      Position 0 :=_MC_Home_Position 0 , .
67      FirstVelocity 100000 :=_MC_Home_FirstVelocity 100000 , 3
68      SecondVelocity 10000 :=_MC_Home_SecondVelocity 10000 ) 4.
69      Acceleration 100000 := _MC_Home_Acceleration 100000 , .
70      Deceleration 100000 :=_MC_Home_Deceleration 100000 , .
71      Jerk 2100000000 := _MC_Home_Jerk 2100000000 , .
72      Done FALSE =>_MC_Home_Done FALSE , .
73      Busy FALSE =>_MC_Home_Busy FALSE , .
74      CommandAborted FALSE =>_MC_Home_CommandAborted FALSE , .
75      Error FALSE =>_MC_Home_Error FALSE , .
76      ErrorID MC_NO_ERRO =>_MC_Home_ErrorID 0 );.
77  .
78  //高速轴设置位置.
79  _MC_SetPosition(.
```

如“1”和“2”步所示，电机以第一速度100000开始做正向回原运动。

_MC_ReadMotionStatus_Decelerating	BOOL	FALSE			减速运动
_MC_ReadMotionStatus_DirectionPositive ①	BOOL	TRUE			正向运动
_MC_ReadMotionStatus_DirectionNegative	BOOL	FALSE			反向运动
X6	BIT	FALSE		%QX1.0	原点DI
X7	BIT	FALSE		%QX1.1	限位DI
Axisposition_0 ②	DINT	781045			轴位置

```
62    //轴回原.
63    _MC_Home(.
64        Axis:= LCT_AXIS_P0, .
65        Execute TRUE := _MC_Home_Execute TRUE, .
66        Position 0 :=_MC_Home_Position 0, .
67        FirstVelocity 100000 :=_MC_Home_FirstVelocity 100000, .
68        SecondVelocity 10000 :=_MC_Home_SecondVelocity 10000 , .
69        Acceleration 100000 := _MC_Home_Acceleration 100000, .
70        Deceleration 100000 :=_MC_Home_Deceleration 100000, .
71        Jerk 2100000000 := _MC_Home_Jerk 2100000000, .
72        Done FALSE =>_MC_Home_Done FALSE , .
73        Busy TRUE =>_MC_Home_Busy TRUE ,③
74        CommandAborted FALSE =>_MC_Home_CommandAborted FALSE , .
75        Error FALSE =>_MC_Home_Error FALSE , .
76        ErrorID MC_NO_ERRO =>_MC_Home_ErrorID 0 );.
77    .
78    //高速轴设置位置.
```

如“2”步所示，手动触发变量X7为TRUE模拟正限位感应后，限位获得高电平且电机以第二速度开始做反向运动，“1”步所示 。

_MC_ReadMotionStatus_Decelerating	BOOL	FALSE			减速运动
_MC_ReadMotionStatus_DirectionPositive	BOOL	FALSE			正向运动
_MC_ReadMotionStatus_DirectionNegative	BOOL ①	TRUE			反向运动
X6	BIT	FALSE		%QX1.0	原点DI
X7	BIT ②	TRUE		%QX1.1	限位DI
Axisposition_0	DINT ③	3230755			轴位置

```
62    //轴回原.
63    _MC_Home(.
64        Axis:= LCT_AXIS_P0, .
65        Execute TRUE := _MC_Home_Execute TRUE, .
66        Position 0 :=_MC_Home_Position 0, .
67        FirstVelocity 100000 :=_MC_Home_FirstVelocity 100000, .
68        SecondVelocity 10000 :=_MC_Home_SecondVelocity 10000 , .
69        Acceleration 100000 := _MC_Home_Acceleration 100000, .
70        Deceleration 100000 :=_MC_Home_Deceleration 100000, .
71        Jerk 2100000000 := _MC_Home_Jerk 2100000000, .
72        Done FALSE =>_MC_Home_Done FALSE , .
73        Busy TRUE =>_MC_Home_Busy TRUE ,④
74        CommandAborted FALSE =>_MC_Home_CommandAborted FALSE , .
75        Error FALSE =>_MC_Home_Error FALSE , .
76        ErrorID MC_NO_ERRO =>_MC_Home_ErrorID 0 );.
77    .
78    //高速轴设置位置.
79    _MC_SetPosition(.
```

如“1”步，手动给限位FALSE模拟正限位下降沿后，回原完成标志亮起“3”步，轴位置为0，本次回原结束。

_MC_ReadMotionStatus_Decelerating	BOOL	FALSE			减速运动
_MC_ReadMotionStatus_DirectionPositive	BOOL	FALSE			正向运动
_MC_ReadMotionStatus_DirectionNegative	BOOL	FALSE			反向运动
X6	BIT	FALSE		%QX1.0	原点DI
X7 ①	BIT	FALSE		%QX1.1	限位DI
Axisposition_0 ②	DINT	0			轴位置

```
62  //轴回原.
63  _MC_Home(.
64      Axis:= LCT_AXIS_P0, .
65      Execute TRUE := _MC_Home_Execute TRUE, .
66      Position 0 :=_MC_Home_Position 0, .
67      FirstVelocity 100000 :=_MC_Home_FirstVelocity 100000, .
68      SecondVelocity 10000 :=_MC_Home_SecondVelocity 10000 , .
69      Acceleration 100000 := _MC_Home_Acceleration 100000, .
70      Deceleration 100000 :=_MC_Home_Deceleration 100000, .
71      Jerk 2100000000 := _MC_Home_Jerk 2100000000, .
72      Done TRUE =>_MC_Home_Done TRUE , ③
73      Busy FALSE =>_MC_Home_Busy FALSE , .
74      CommandAborted FALSE =>_MC_Home_CommandAborted FALSE , .
75      Error FALSE =>_MC_Home_Error FALSE , .
76      ErrorID MC_NO_ERRO =>_MC_Home_ErrorID 0 );.
77  .
78  //高速轴设置位置.
79  _MC_SetPosition(.
80      Axis:=LCT_AXIS_P0, .
```

情形二：在回原触发前就给限位TRUE信号，限位保持高电平，然后再触发回原“2”。

_MC_ReadMotionStatus_DirectionNegative	BOOL	FALSE			反向运动
X6	BIT	FALSE		%QX1.0	原点DI
X7 ①	BIT	TRUE		%QX1.1	限位DI
Axisposition_0	DINT	0			轴位置

```
62  //轴回原.
63  _MC_Home(.
64      Axis:= LCT_AXIS_P0, .
65      Execute FALSE := _MC_Home_Execute FALSE <TRUE>, . ②
66      Position 0 :=_MC_Home_Position 0, .
67      FirstVelocity 100000 :=_MC_Home_FirstVelocity 100000, .
68      SecondVelocity 10000 :=_MC_Home_SecondVelocity 10000 , .
69      Acceleration 100000 := _MC_Home_Acceleration 100000, .
70      Deceleration 100000 :=_MC_Home_Deceleration 100000, .
71      Jerk 2100000000 := _MC_Home_Jerk 2100000000, .
72      Done FALSE =>_MC_Home_Done FALSE , .
73      Busy FALSE =>_MC_Home_Busy FALSE , .
74      CommandAborted FALSE =>_MC_Home_CommandAborted FALSE , .
75      Error FALSE =>_MC_Home_Error FALSE , .
76      ErrorID MC_NO_ERRO =>_MC_Home_ErrorID 0 );.
77  .
```

如“1”、“2”所示，回原触发后电机就立即以第二速度做反向回原运动。

_MC_ReadMotionStatus_Decelerating	BOOL	FALSE			减速运动
_MC_ReadMotionStatus_DirectionPositive	BOOL	FALSE			正向运动
_MC_ReadMotionStatus_DirectionNegative ①	BOOL	TRUE			反向运动
X6	BIT	FALSE		%QX1.0	原点DI
X7	BIT	TRUE		%QX1.1	限位DI
Axisposition_0 ②	DINT	-47615			轴位置

```
62  //轴回原.
63  _MC_Home(.
64      Axis:= LCT_AXIS_P0, .
65      Execute TRUE :=  _MC_Home_Execute TRUE , ③
66      Position 0 :=_MC_Home_Position 0 , .
67      FirstVelocity 100000 :=_MC_Home_FirstVelocity 100000 , .
68      SecondVelocity 10000 :=_MC_Home_SecondVelocity 10000 , .
69      Acceleration 100000 := _MC_Home_Acceleration 100000 , .
70      Deceleration 100000 :=_MC_Home_Deceleration 100000 , .
71      Jerk 2100000000 := _MC_Home_Jerk 2100000000 , .
72      Done FALSE =>_MC_Home_Done FALSE , .
73      Busy TRUE =>_MC_Home_Busy TRUE , ④
74      CommandAborted FALSE =>_MC_Home_CommandAborted FALSE , .
75      Error FALSE =>_MC_Home_Error FALSE , .
76      ErrorID MC_NO_ERRO =>_MC_Home_ErrorID 0 );.
77  .
78  //高速轴设置位置.
```

手动给限位FALSE，限位获得下降沿的同时回原完成标志亮起“3”步，轴位置为0，回原完成。

_MC_ReadMotionStatus_Decelerating	BOOL	FALSE			减速运动
_MC_ReadMotionStatus_DirectionPositive	BOOL	FALSE			正向运动
_MC_ReadMotionStatus_DirectionNegative	BOOL	FALSE			反向运动
X6	BIT	FALSE		%QX1.0	原点DI
X7 ①	BIT	FALSE		%QX1.1	限位DI
Axisposition_0 ②	DINT	0			轴位置

```
62  //轴回原.
63  _MC_Home(.
64      Axis:= LCT_AXIS_P0, .
65      Execute TRUE := _MC_Home_Execute TRUE , .
66      Position 0 :=_MC_Home_Position 0 , .
67      FirstVelocity 100000 :=_MC_Home_FirstVelocity 100000 , .
68      SecondVelocity 10000 :=_MC_Home_SecondVelocity 10000 , .
69      Acceleration 100000 := _MC_Home_Acceleration 100000 , .
70      Deceleration 100000 :=_MC_Home_Deceleration 100000 , .
71      Jerk 2100000000 := _MC_Home_Jerk 2100000000 , .
72      Done TRUE =>_MC_Home_Done TRUE , ③
73      Busy FALSE =>_MC_Home_Busy FALSE , .
74      CommandAborted FALSE =>_MC_Home_CommandAborted FALSE , .
75      Error FALSE =>_MC_Home_Error FALSE , .
76      ErrorID MC_NO_ERRO =>_MC_Home_ErrorID 0 );.
77  .
78  //高速轴设置位置.
```