中控 · SUPCON

GCS M4

通用型控制器

MCU4003-S

使用手册

20230821

版权声明

中控技术股份有限公司

地址：杭州市滨江区六和路309号中控科技园（310053）

电话：0571-88851888

传真：0571-86667198

用户服务：400-887-6000

E-mail：custserv@supcon.com

网址：http://www.supcon.com

文档标志符定义

警告：
标示有可能导致人身伤亡或设备损坏的信息。

防电击：

电击危险：标示有可能产生电击危险的信息。

防静电：

防止静电：标示防止静电损坏设备的信息。

注意：

提醒需要特别注意的信息。

提示：

标记对用户的建议或提示。

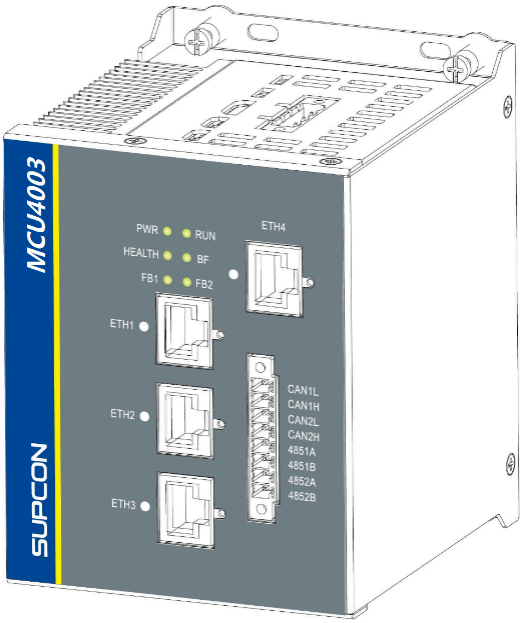
目 录

1 产品简介

通用型MCU4003-S为一款可编程控制器。控制器采用嵌入式硬件架构，全金属机身，提供丰富的网络接口，能够满足各种项目的扩展需求，具备强悍的控制性能。其内置高性能CPU，配有DDR内存和大容量SSD硬盘，具备卓越的运算能力，满足高负荷运算和存储的应用需求。

本产品采用MotionPro组态软件，支持标准IEC61131-3编程，开放式TCP/IP编程，串口/CAN自定义编程，各类功能库，减少用户学习与使用成本。

功能特点



- 64位4核CPU，主频1.9GHz
- 2GB DDR内存，64GB SSD硬盘
- 支持多种通信协议：
  - Modbus TCP
  - Modbus RTU
  - Profinet
  - OPC UA Sever
  - CANopen
  - EtherCAT
  - CAN/串口自定义
  - 开放式TCP/IP协议
- 支持标准IEC61131-3组态编程
- 支持WEB配置和管理
- 工作温度：（-30 ~ 65）℃
- 电磁兼容性EMC：3B
- G3防腐
- 1G抗振

2 技术指标

表 2-1 控制器技术指标

参数项	说明	
硬件		
处理器	4核64位 CPU，每核1.9GHz	
内存容量	2GB	
硬盘容量	64GB SSD	
以太网	2路，支持Modbus TCP，OPC UA，MQTT，开放式TCP/UDP	
Profinet通信	1路（每路最多128从站）	
EtherCAT通信	1路（每路最多128从站）	
RS485通信	2路，Modbus RTU主从（每路最多32从站），串口自定义协议	
CANopen通信	主站，2路（每路最多16从站），波特率1Mbps，兼容CAN2.0 A/B	
背板总线	中控ECI总线，速率128Mbps	
电源	支持端子和背板2种供电方式，均为双路冗余DC 24V（-15%~+20%），防反接	
功耗	15W	
技术参数		
编程方式	IEC 61131-3编程语言（LD、IL、ST、SFC、CFC），C语言	
布尔指令时间	0.007μs	
IO规模	32个机架（含本地机架在内，1个机架等视同1个Profinet从站）	
用户任务	是否支持多任务	支持循环、惯性、事件、状态4种任务类型
	任务个数	16个
	最小任务周期	500μs
组态及程序容量	用户程序容量	128M Byte
	用户数据容量	128M Byte
	掉电保持数据容量	512 K Byte
指示灯	1个电源灯，5个双色状态指示灯	
结构		

表 2-1 控制器技术指标（续）

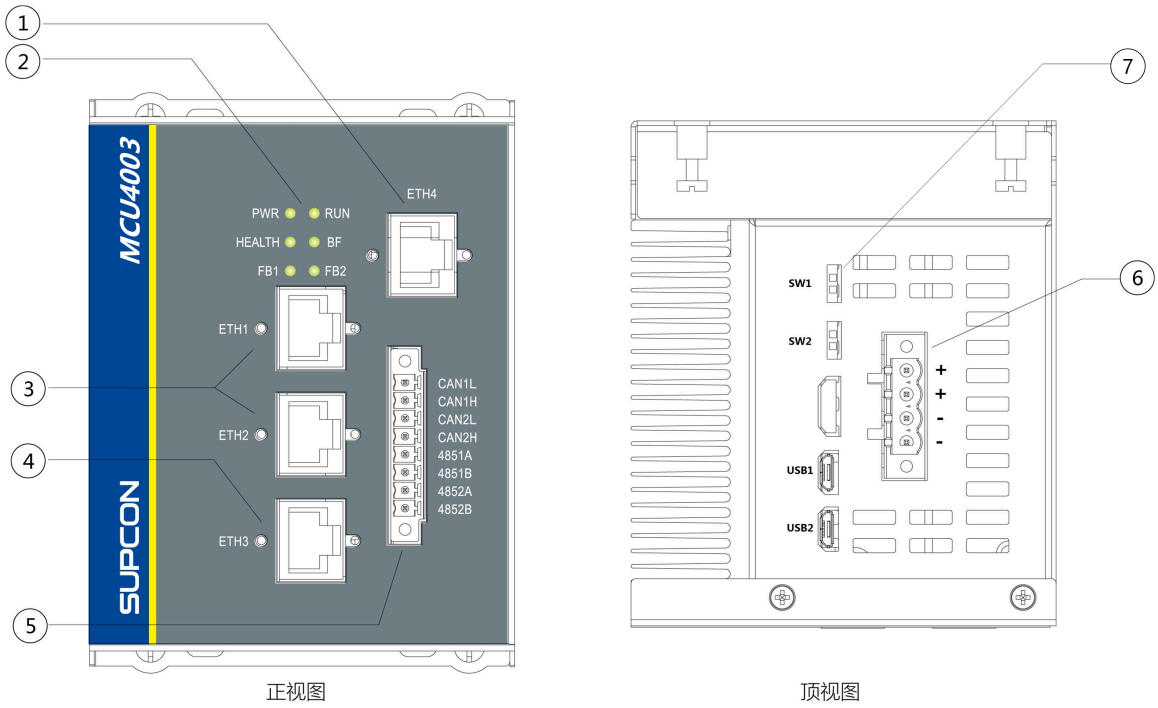
参数项	说明
结构 & 散热型材	钣金，无风扇散热
安装方法	机架/平板安装
尺寸（高×宽×深）	120mm×90mm×107mm
环境	
工作温度	（-30~65）℃
贮存温度	（-40~80）℃
防腐等级	G3（ANSI/ISA S71.04）
EMC指标	3B
相对湿度	5%RH ~95%RH 无冷凝
振动	频率：5~150Hz
	位移：3.5mm（<8.4Hz）
	加速度：1g（>=8.4Hz）
	方向：3轴向
冲击	15g，11ms，半正弦波，3轴向

3 外观和接线

本节介绍设备的外观结构，以及设备各接口接线要求。

3.1 接口图

控制器模块上的接口如下图所示。未指示的接口为保留接口，请勿使用。



正视图　　顶视图

1- Profinet接口，2-指示灯，3-百兆以太网接口，4-EtherCAT接口，5-CAN和RS485接口，6-24V DC电源接口，7-RS485终端电阻

图 3-1 控制器接口示意图

3.2 接口或端子定义

接口标识	功能描述
+ + - -	2路冗余电源接口： 24V DC（-15%~+20%）
SW1	RS485A的终端120欧姆电阻，ON为使能
SW2	RS485B的终端120欧姆电阻，ON为使能
CAN1L	第1路CAN通信接口，内置120欧姆终端电阻 支持CANOpen协议
CAN1H	

接口标识	功能描述	
CAN2L	第2路CAN通信接口，内置120欧姆终端电阻 支持CANOpen协议	
CAN2H		
4851A	+	第1路RS485通信接口：支持Modbus RTU协议
4851B	-	
4852A	+	第2路RS485通信接口：支持Modbus RTU协议
4852B	-	
ETH1	100Mbps，以太网口1： 1.系统程序调试 2.用户程序下载与调试 3.Modbus TCP协议 4.Socket（TCP，UDP） 5.默认IP地址：172.20.1.2	
ETH2	100Mbps，以太网口2： 1.系统程序调试 2.用户程序下载与调试 3.Modbus TCP协议 4.Socket（TCP，UDP） 5.默认IP地址：172.21.1.2	
ETH3	1000Mbps，EtherCAT接口： 1.EtherCAT协议 2.支持自动扫描 3.I/O从站 4.默认IP地址：172.23.1.2	
ETH4	100Mbps，Profinet接口： 1.Profinet协议 2.支持自动扫描 3.I/O从站 4.默认IP地址：172.22.1.2	

3.3 线缆和接线示例

控制器模块上所有接口连线如下图所示。

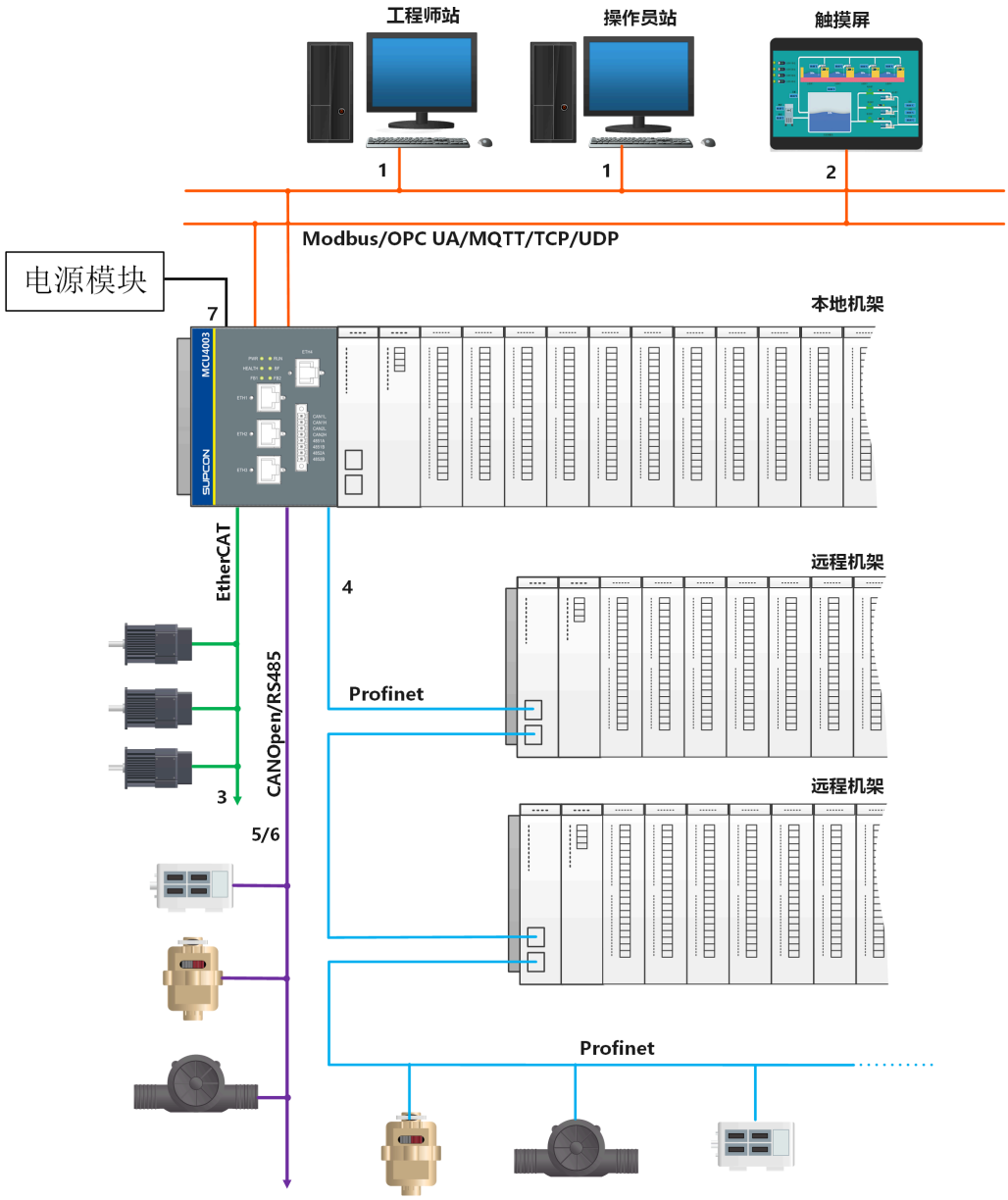


图 3-2 典型接线图

控制器模块接线和线缆说明如下表所示。

表 3-1 线缆和接线说明

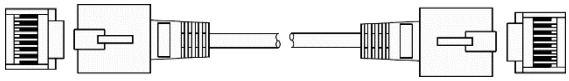
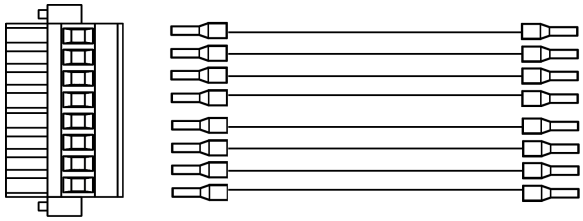
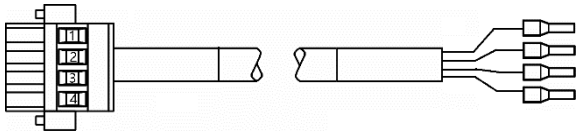
序号	线缆图	连接说明
1	超五类非屏蔽双绞线-直通型 	计算机通过网线连接到控制器的ETH1端口
2		触摸屏设备连接到控制器的ETH2端口
3		EtherCAT从设备连接到控制器的ETH3端口
4		远程IO设备/PN从站设备连接到控制器的ETH4端口

表 **3-1** 线缆和接线说明（续）

序号	线缆图	连接说明
5	线径：（0.2～1.5）mm^2 	CANopen从设备连接到控制器的CAN1或CAN2端口
6		Modbus RTU从设备连接到控制器的COM1或COM2端口
7	线径：（0.2～2.5）mm^2（str），（0.2～4）mm^2（sol） 	电源模块连接模块电源输入接口

注意：

可通过**2**种方式为控制器供电。

- 通过控制器顶部电源接线端子接入外部电源模块。
- 在控制器所在机架安装**PW3203AC-S**电源模块。

3.4 插拔式端子的安装和接线

控制器模块上的电源接口是一个4-PIN插拔式螺丝接线端子，CAN总线和485总线接口是一个8-PIN插拔式螺丝接线端子，接线端子安装和接线图如图 3-3 所示。

安装接线端子

插拔式接线端子插入模块上的底座后，用小号螺丝刀顺时针拧紧固定螺丝。

接线

1. 使用小号一字螺丝刀逆时针方向拧松螺丝，如下图①。
2. 将线缆插入螺丝旁的进线孔，如下图②。
3. 确认线缆已完全插入至底部后，使用小号一字螺丝刀顺时钟方向拧紧螺丝，如下图③。
4. 轻轻往外拽线缆，无法拽出即连接可靠，完成接线。

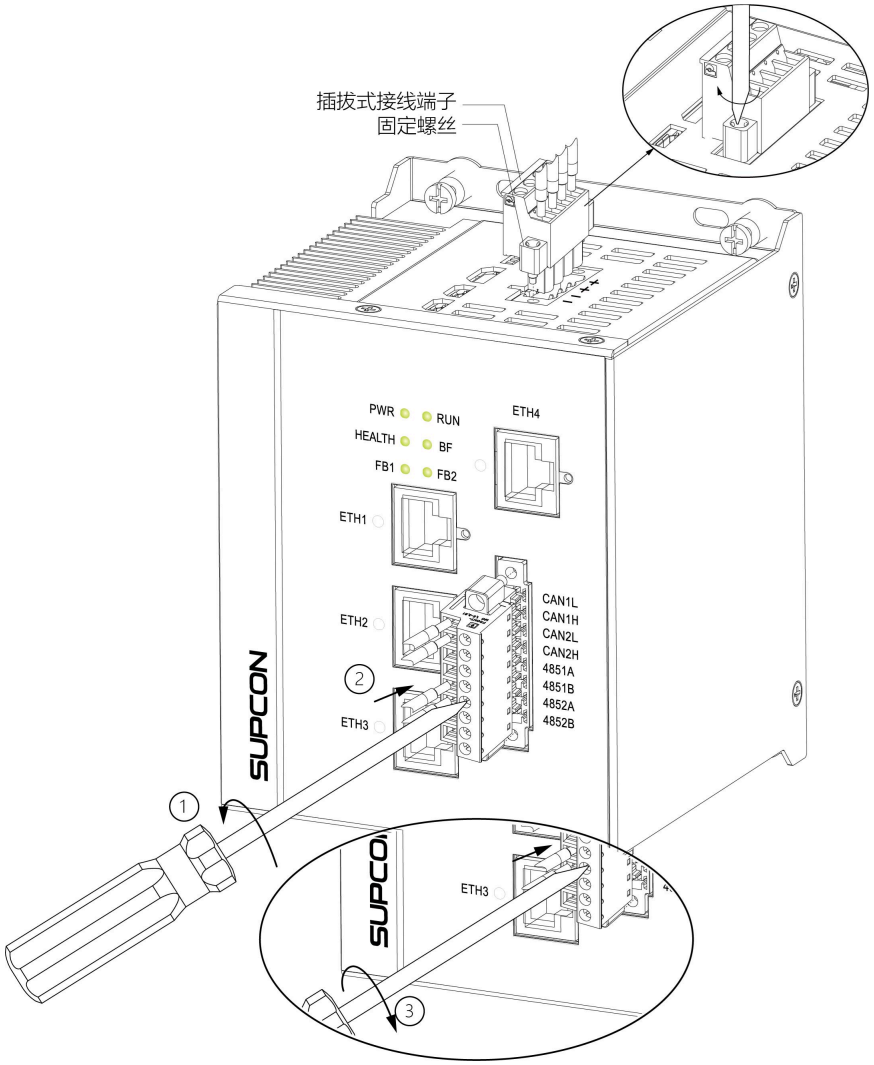


图 3-3 接线端子的安装和接线

4 尺寸和安装

本节介绍模块尺寸，及其安装步骤。

4.1 尺寸图

控制器的尺寸图如下图所示。

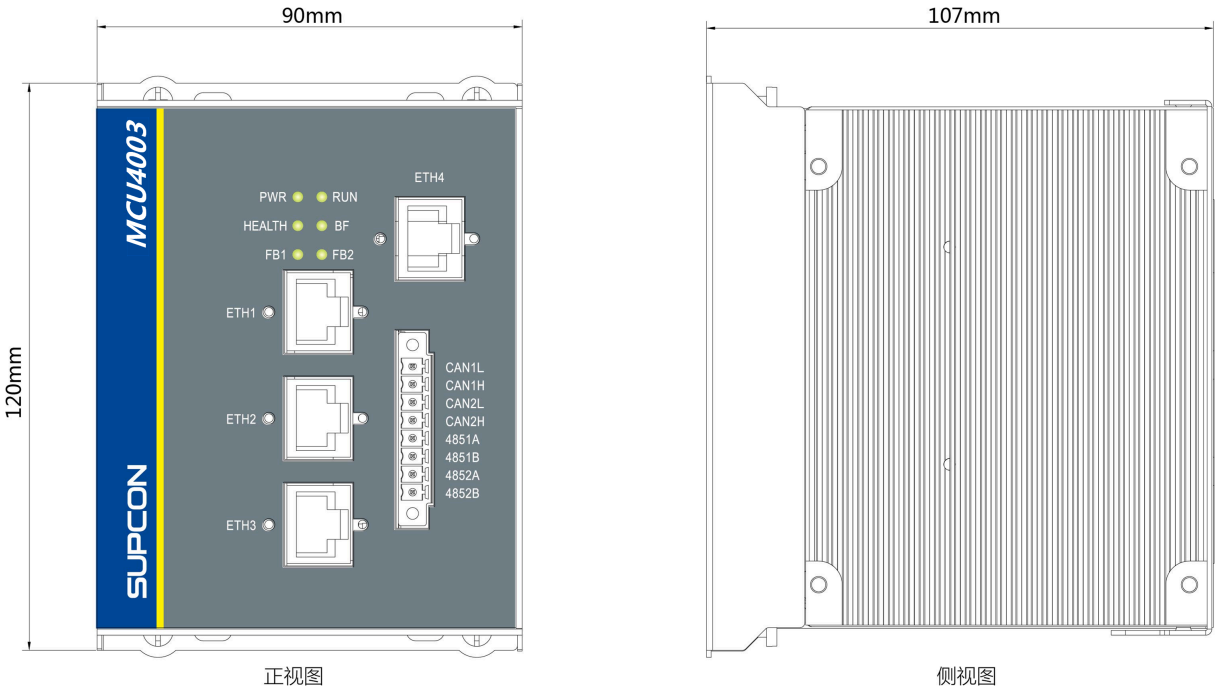


图 4-1 尺寸图

4.2 安装步骤

控制器模块安装于CN32系列机架的0~3槽位中，每个控制器模块占机架4个槽位。

安装步骤

1. 将模块垂直于机架，对齐模块背面和机架槽位上的连接器后，缓慢平稳的装入模块。
2. 用中号十字螺丝刀顺时针方向拧紧上下四颗固定螺丝，推荐扭矩为（0.5~0.6）Nm。

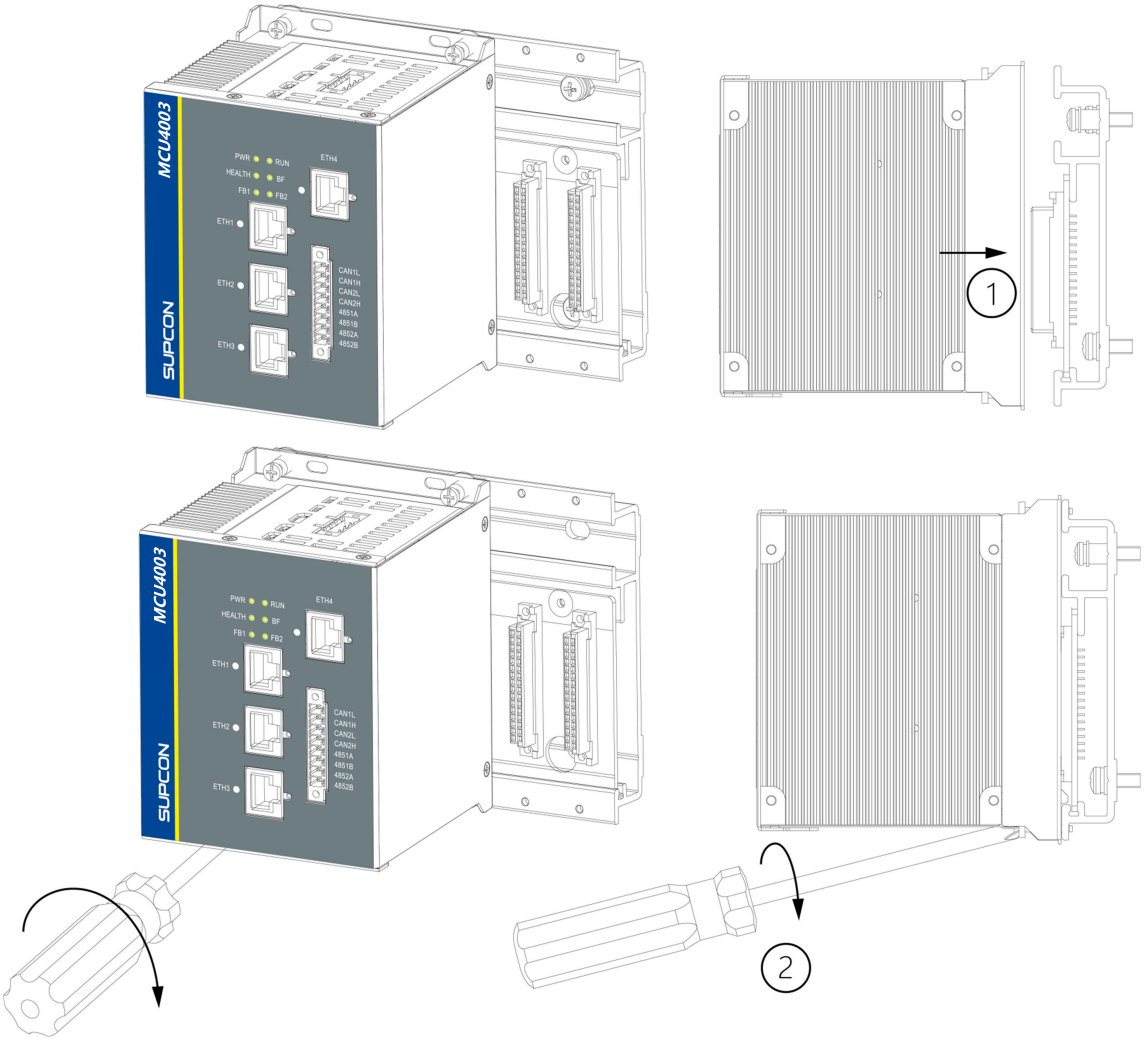

MCU4003
SUPCON
PWR
RUN
HEALTH
BF
FB1
FB2
ETH4
ETH1
ETH2
ETH3
CAN1L
CAN1H
CAN2L
CAN2H
485 1A
485 1B
485 2A
485 2B
1
2

图 **4-2** 安装图

5 快速入门

本章节介绍了最小系统的接线方式、编程环境、典型编程流程等，帮助您快速了解可控制器。

5.1 安装MotionPro组态软件

本节介绍安装MotionPro软件前的准备工作，以及如何安装/卸载软件。

5.1.1 操作系统版本

建议使用Windows 10 Enterprise LTSC或Windows 10 professional 1909。

5.1.2 操作系统最低配置要求

2.5 GHz处理器，8 GB以上可用内存，20GB以上C盘可用空间，64位操作系统。

5.1.3 安装前的系统设置

安装软件前，应在计算机中完成以下系统设置，以免安装异常。

1. 使用非Administrator账户登录计算机。
2. 在“控制面板 > 所有控制面板项 > 用户帐户 > 用户账户控制设置”中，将“选择何时通知你有关计算机更改的信息”设置为“从不通知”，点击确定。

图 **5-1** 设置用户账户控制设置

3. 在“Windows安全中心 > 防火墙和网络保护”中，暂时关闭Windows防火墙，如下图所示。

图 **5-2** 关闭**Windows**防火墙

4. 暂时关闭“病毒和威胁防护”中的实时保护。

图 **5-3** 关闭实时保护

5. 暂时关闭其余第三方电脑管家、杀毒软件、防火墙等。

5.1.4 软件安装

1. 右键点击MotionPro VX.Y.exe，选择“以管理员身份运行”，启动软件安装。

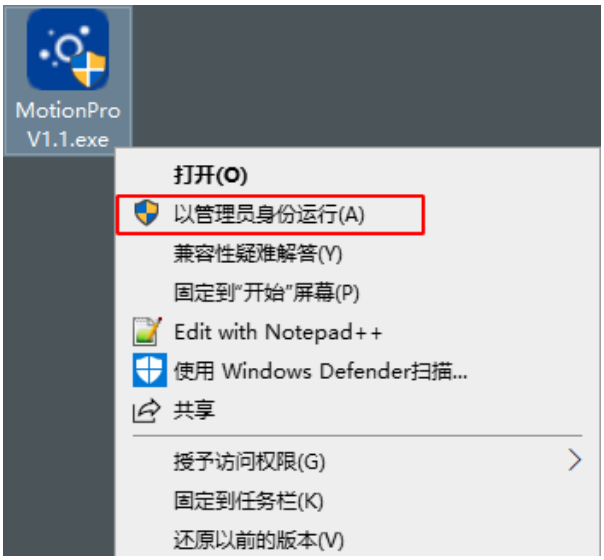


图 **5-4** 以管理员身份运行安装程序

2. 选择安装语言后，点击确定。

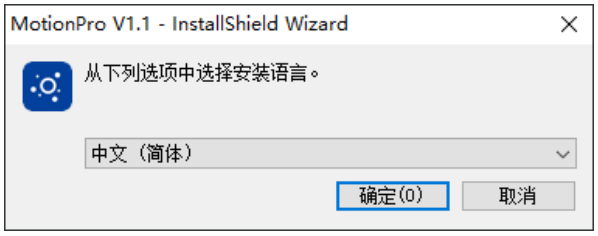


图 **5-5** 选择安装语言

3. 等待片刻后，进入安装向导。

图 **5-6** 进入安装向导

4. 单击下一步后，在许可证协议界面选择接受许可协议。

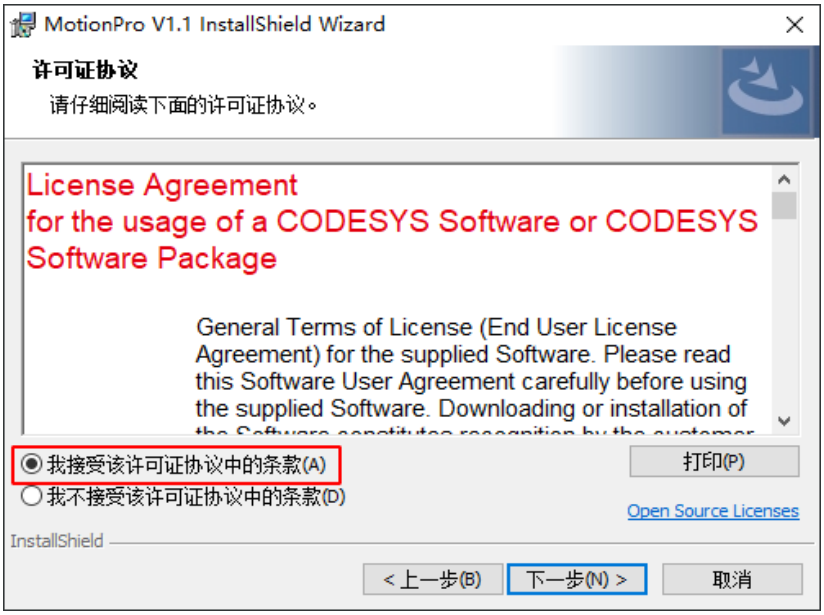


图 **5-7** 许可证协议

5. 选择安装路径，推荐保持默认。

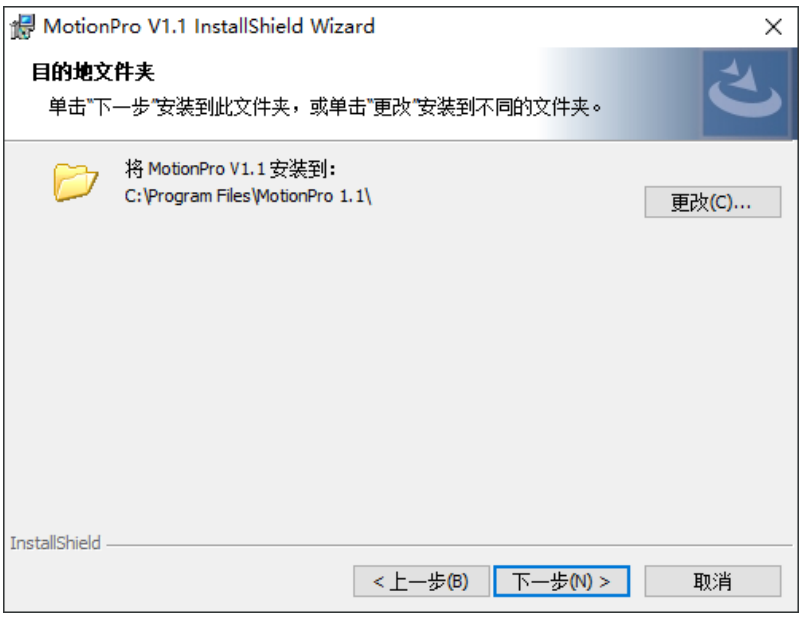


图 **5-8** 选择安装路径

6. 单击下一步后，选择安装类型，请选择“完整安装”。

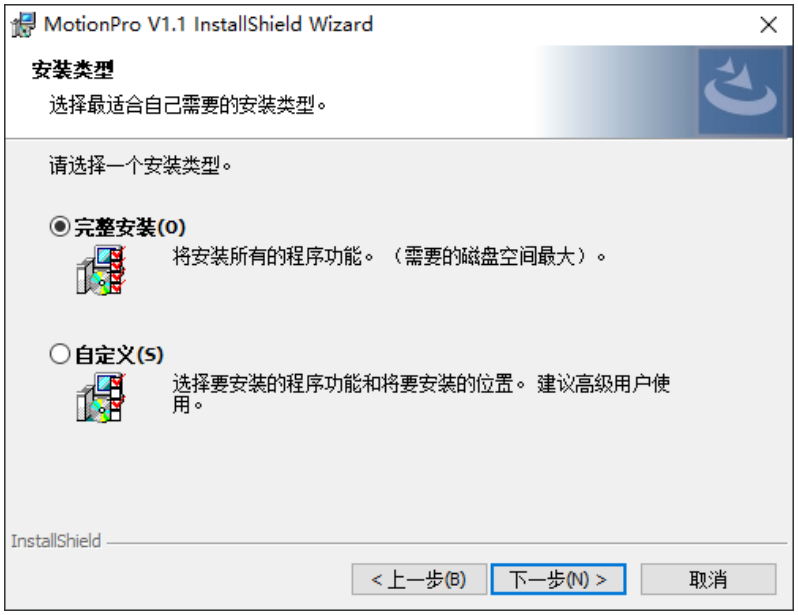


图 **5-9** 选择安装类型

7. 单击下一步后，等待安装完成。注意：根据电脑配置不同，该步骤需要耗时20~30分钟左右，请耐心等待。
8. 如果中途弹出CodeMeter控制中心界面，点击关闭即可。

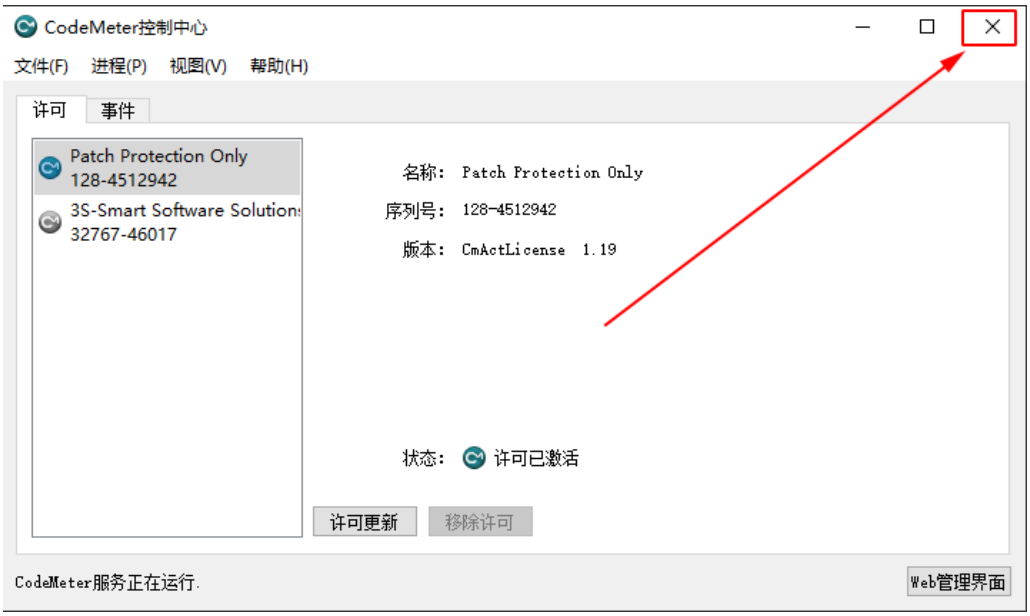


图 **5-10** 关闭**CodeMeter**控制中心界面

9. 安装完成后，点击安装向导“完成”退出。

5.1.5 软件卸载

1. 在“控制面板 > 卸载程序”，找到MotionPro，点击卸载即可。
2. 在“控制面板 > 卸载程序”，找到CodeMeter Runtime Kit，点击卸载。

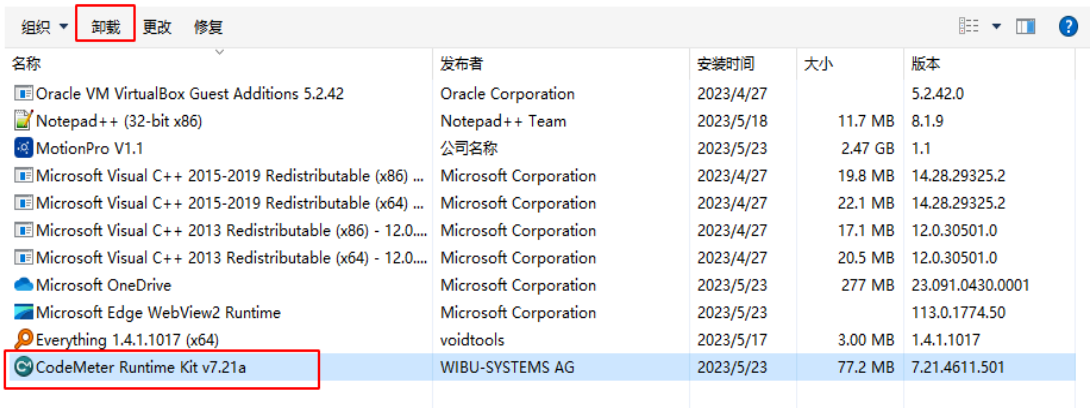


图 5-11 卸载CodeMeter Runtime Kit

3. 显示计算机中所有隐藏文件夹和文件后，删除以下文件夹和文件。

> 提示：
> 显示计算机中隐藏文件/文件夹方法：打开文件资源管理器，在“查看”中勾选“隐藏的项目”即可。

- C:\Program Files\MotionPro X.Y
- C:\ProgramData\CODESYS
- C:\ProgramData\MotionPro
- C:\ProgramData\CodeMeter
- C:\ProgramData\AP
- C:\Users\用户名\AppData\Roaming\CODESYS
- C:\Users\用户名\AppData\Roaming\MotionPro
- C:\Users\用户名\AppData\Roaming\OnlineHelp

> 提示：
> 或者打开清除文件“clean.txt”，将文件扩展名修改为**.bat**后，通过管理员权限运行“**clean.bat**”文件，完成清除后，按空格退出。

4. 重启电脑，至此完成全部卸载清除步骤。

5.2 MotionPro常用功能

本节介绍MotionPro软件中的常用功能，包括修改语言、加载与保存、源码上/下载等。

5.2.1 设置语言

在菜单栏，选择“工具 > 选项”，打开“语言设置”页。

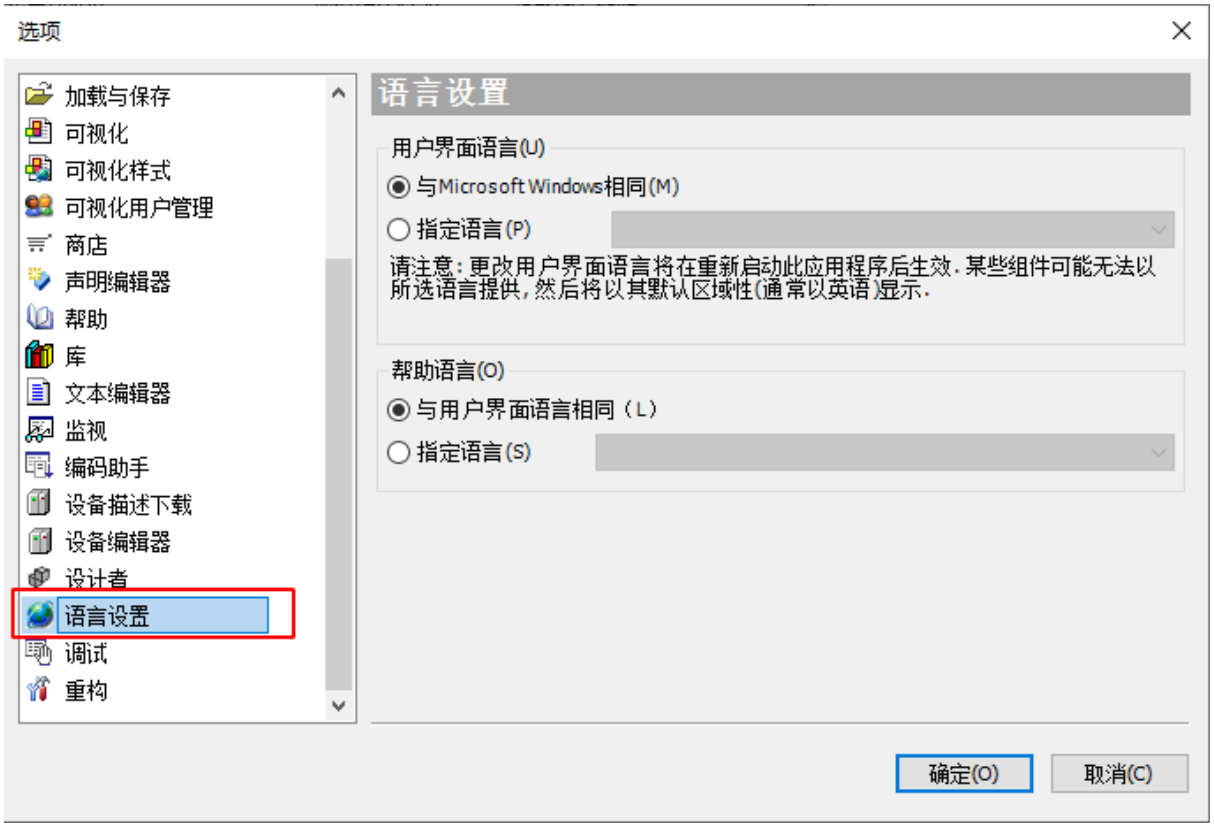


图 **5-12** 设置语言

5.2.2 加载与保存

在菜单栏，选择“工具 > 选项”，打开“加载与保存”页，可配置组态备份，周期自动保存，每次编译前保存等。

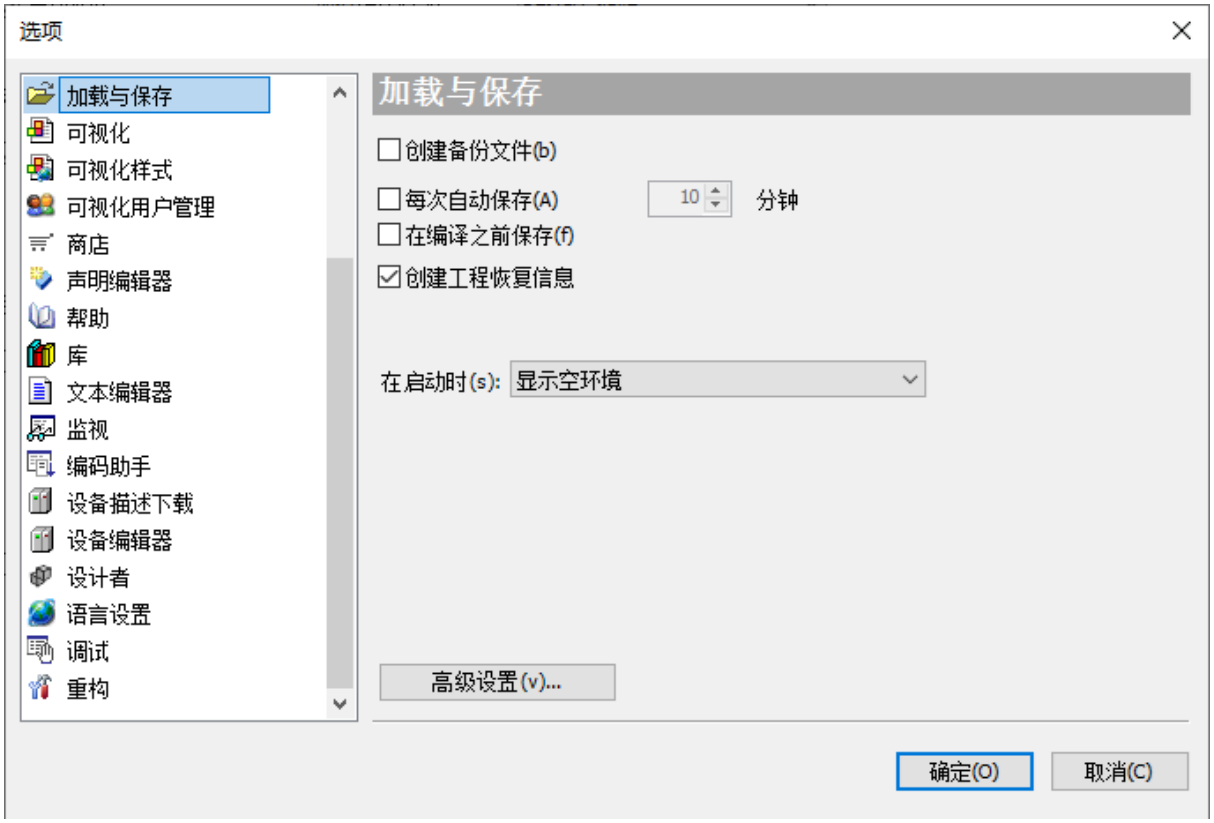


图 **5-13** 加载与保存

5.2.3 源码下载与上载

默认情况下，MotionPro只下载编译后的固件，不会下载组态源码到控制器内。如果组态源码丢失，则无法通过上载找回。

源码下载

如果希望能够上载源码，则需要在组态下载时将源码一并下载，方法为：在菜单栏“工程 > 工程设置”，选择“源代码下载”页，选择“隐式，在程序下载及在线修改时”后确定。此后MotionPro进行组态下载时会下载源码到控制器内。源码可以通过上载找回，但存在知识产权泄露风险，请使用者斟酌。

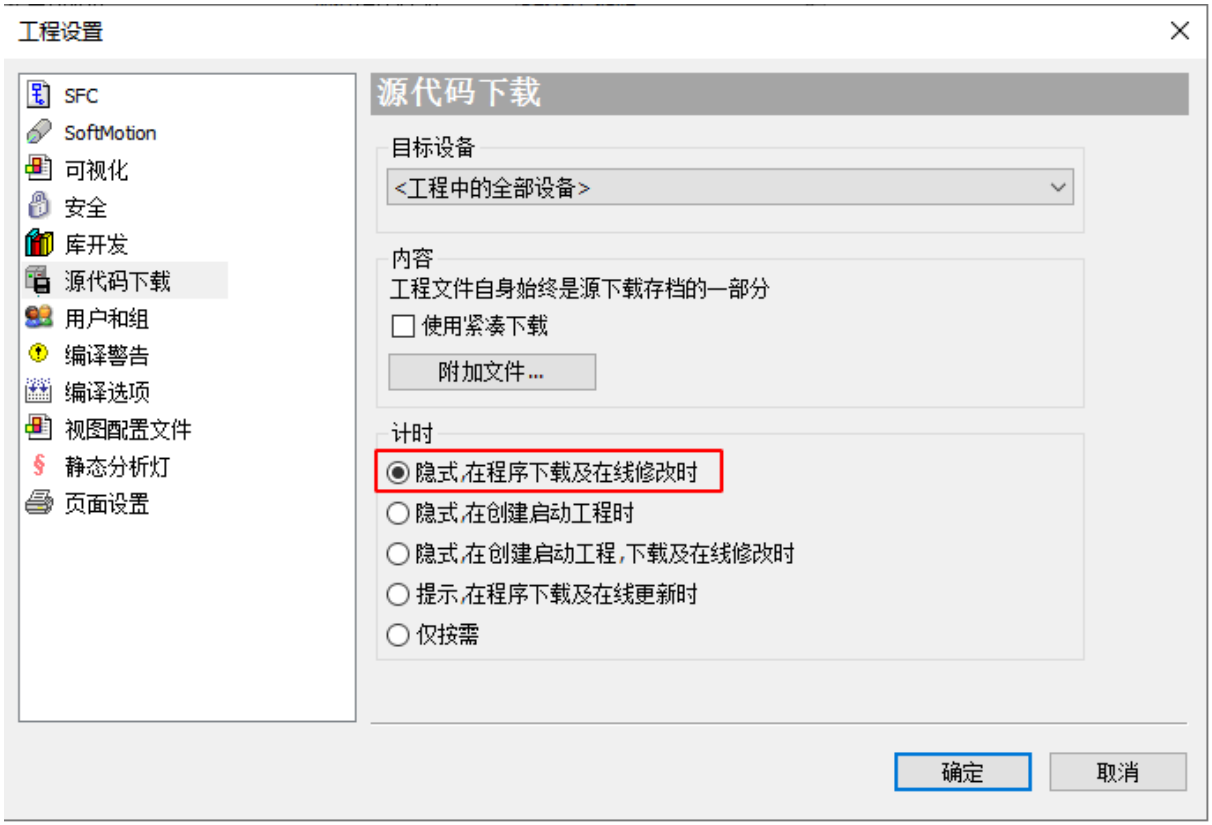


图 **5-14** 源码下载

源码上载

1. 在MotionPro菜单栏中点击“文件 > 源代码上传”，在选择设备界面选择相应控制器后，点击确定。

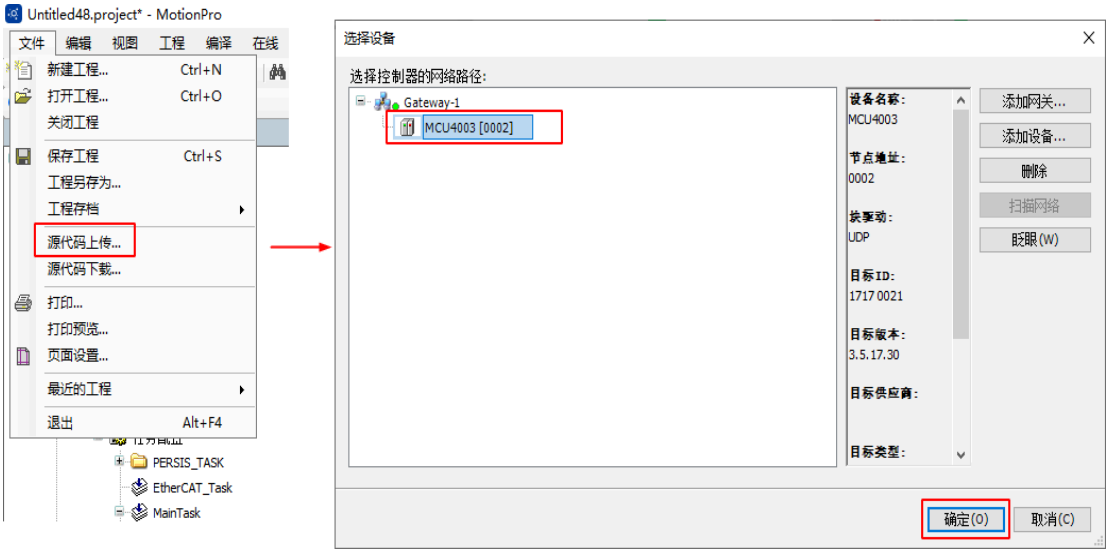


图 **5-15** 选择源代码上传

2. 在弹出的解压工程存档对话框，选择源码上传的目录。

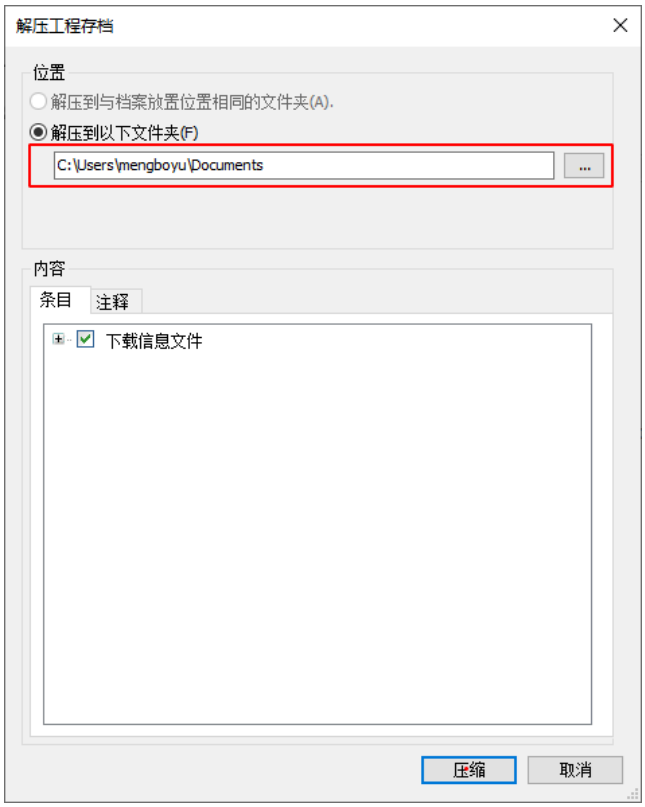


图 **5-16** 选择源码上传目录

5.2.4 监控位号实时值

联机调试状态下，在菜单栏中点击“视图 > 监视”，打开监视窗口。

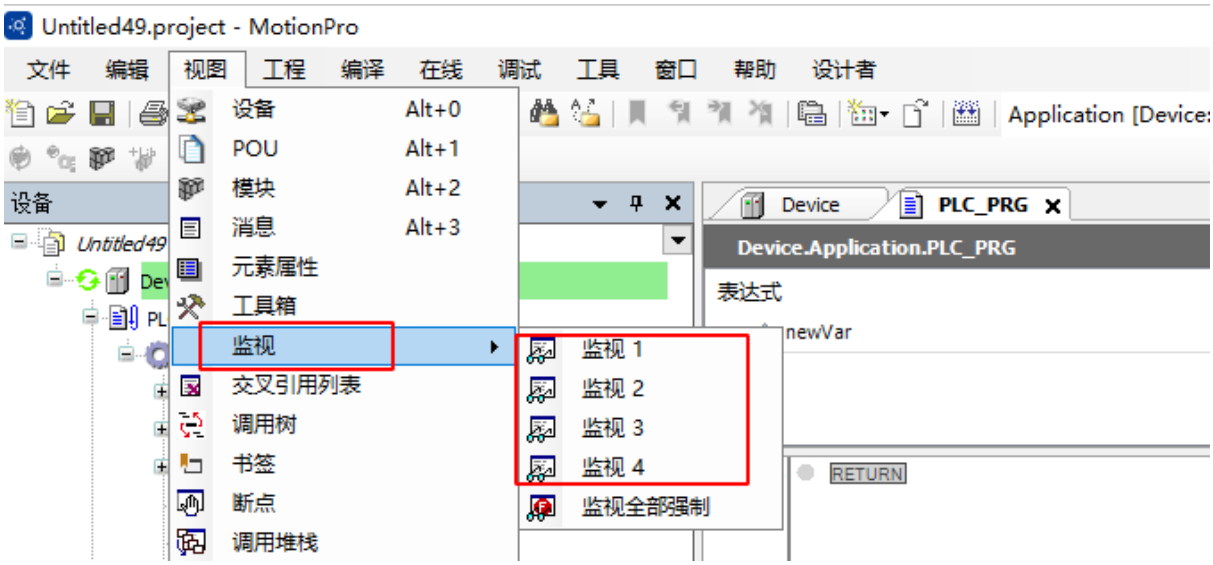


图 **5-17** 选择监视窗口

右键选中要监视的位号，选择“添加查看”，即可把位号添加到监视窗口（或选中位号，以拖放的方式把位号拖至监视窗口也可以完成添加）。

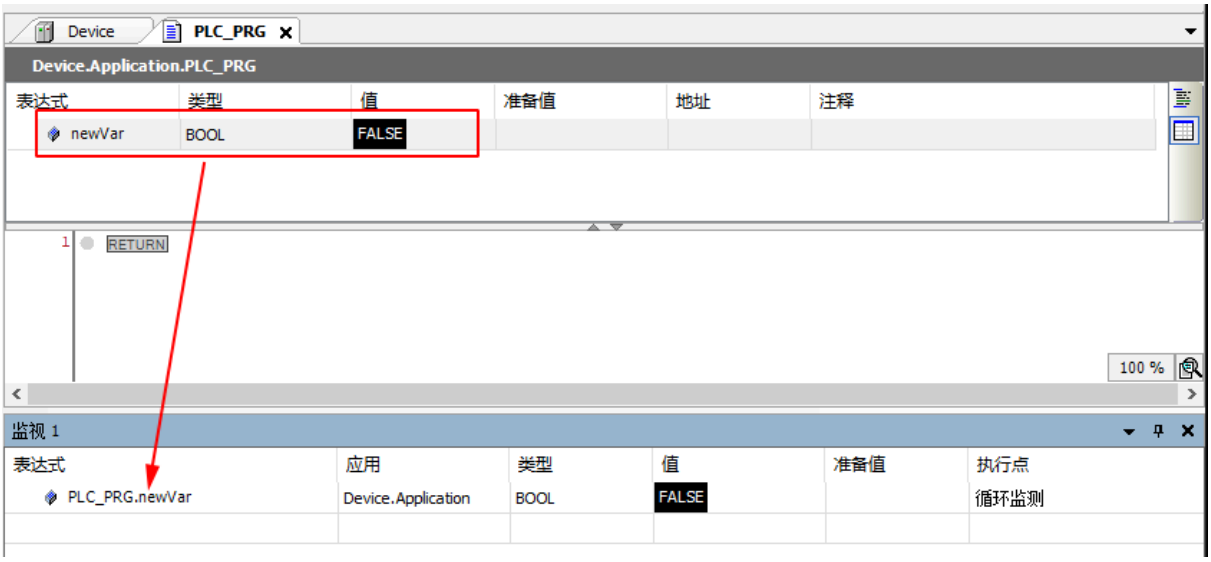


图 **5-18** 将位号拖拽至监视窗口

5.2.5 查看位号实时趋势

如果需要观测位号的趋势，可以使用跟踪功能。

1. 右键点击工程中的“Application > 添加对象 > 跟踪...”，选择用于跟踪的任务（一般选择MainTask即可），点击添加。

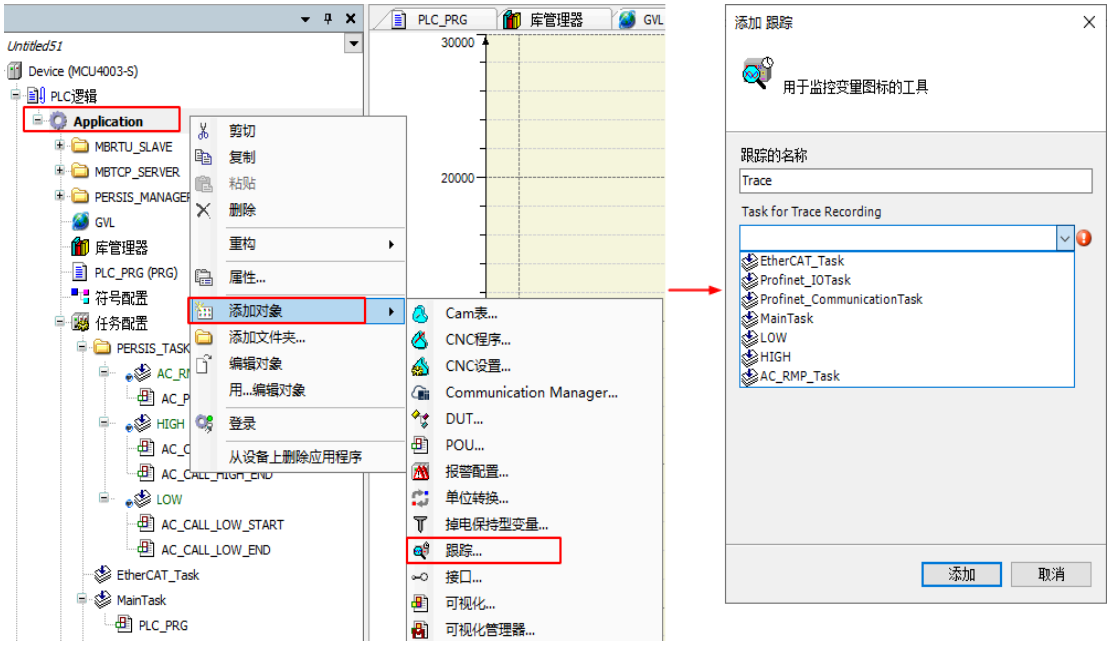


图 **5-19** 选择跟踪任务

2. 在Trace界面，点击“添加变量”，选择要监视的变量。

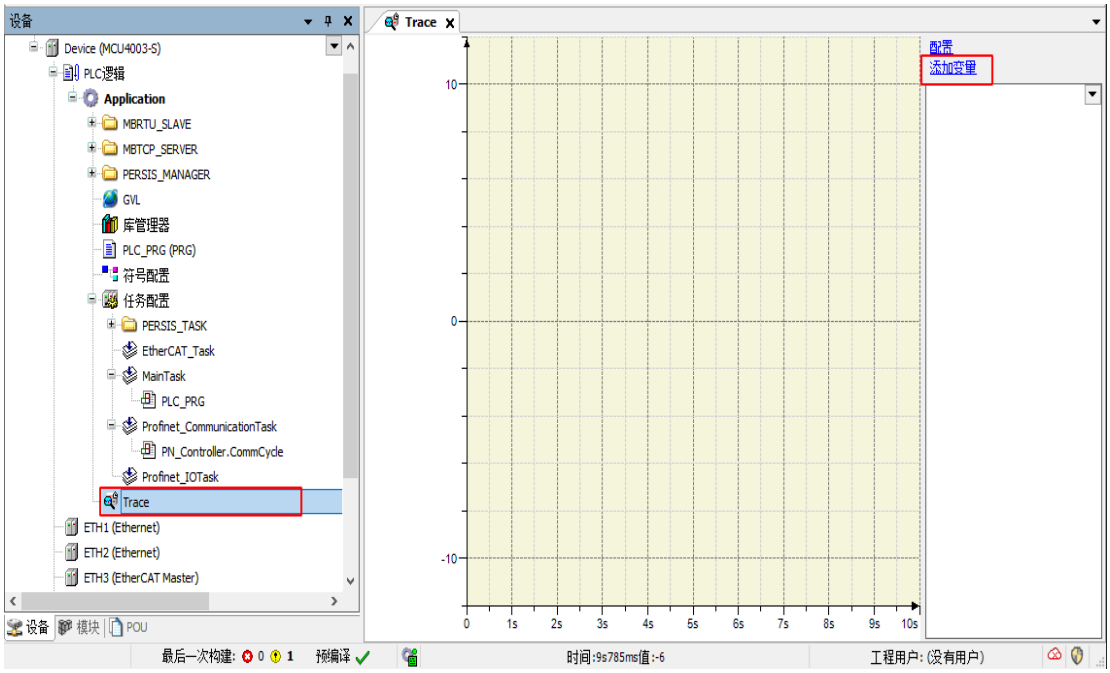


图 5-20 添加变量

3. 在跟踪配置界面，配置位号的显示模式、时间轴等参数，如下图所示。

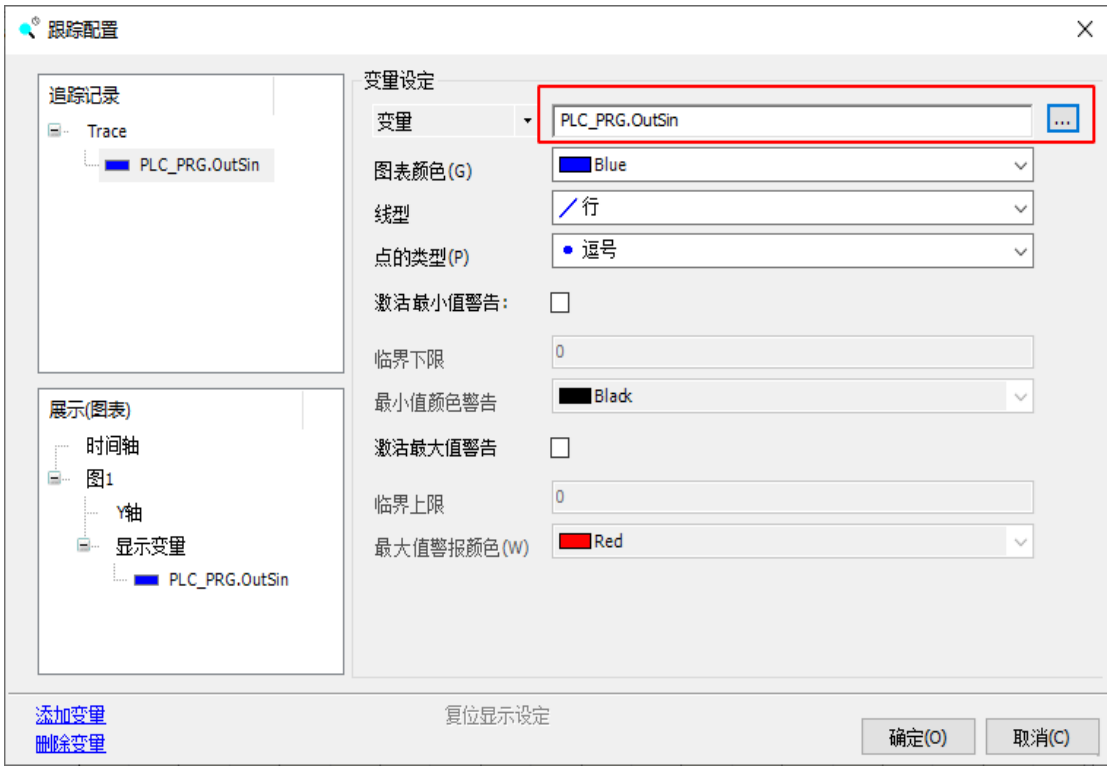


图 5-21 配置位号显示模式

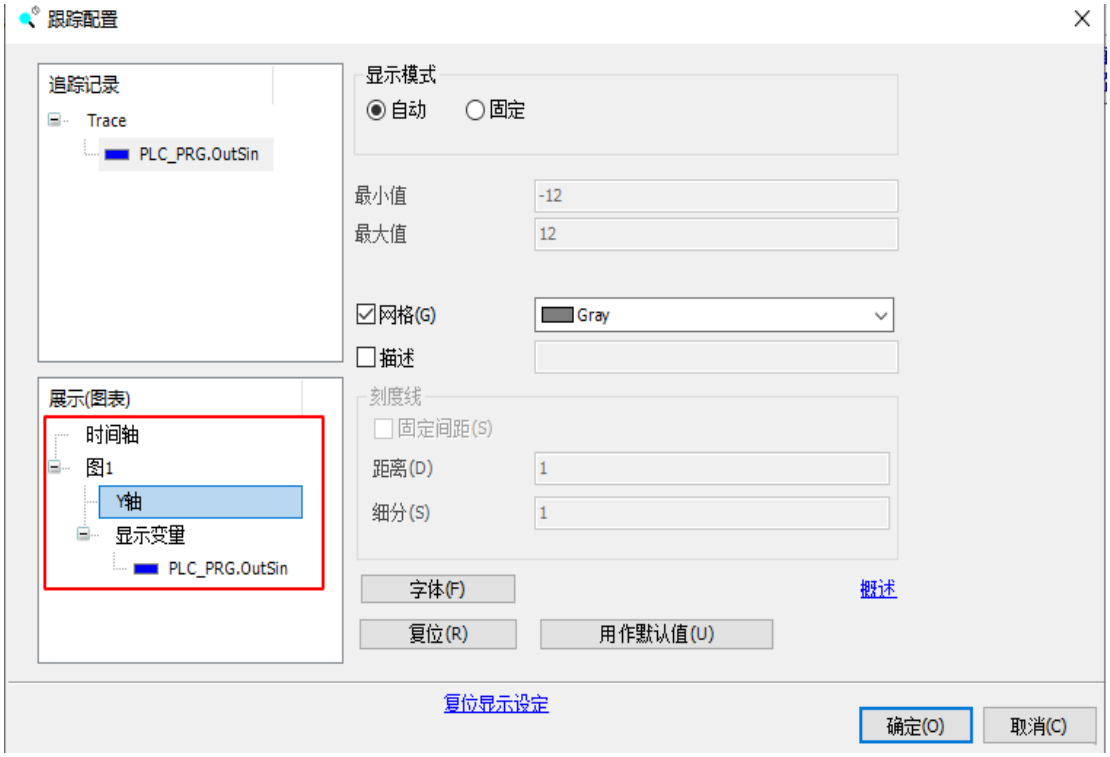


图 **5-22** 配置时间轴

4. 编译并下载组态到控制器后，进入联机调试状态。在Trace界面，右键选择“下载跟踪”。

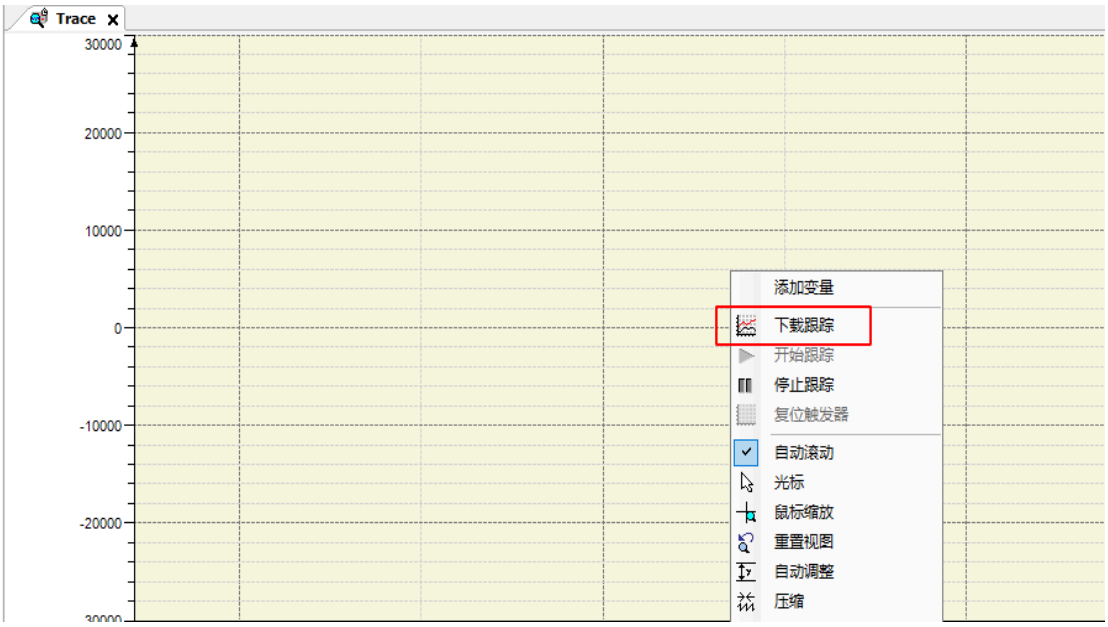


图 **5-23** 选择下载跟踪

5. 即可跟踪位号实时值，操作鼠标滚轮可以进行横坐标的缩放。

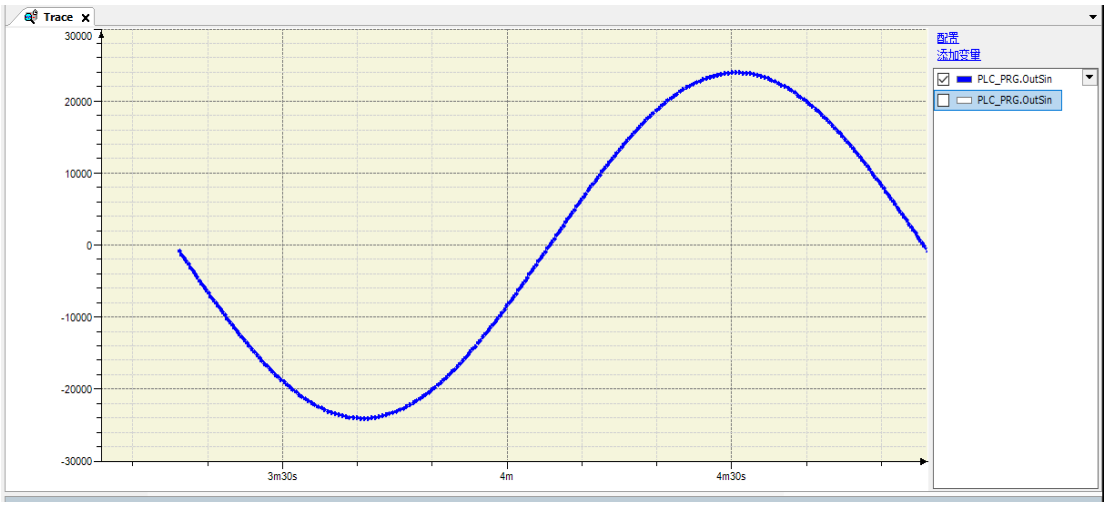


图 **5-24** 查看位号实时趋势

5.3 连接控制器

1. 将控制器安装于CN32系列机架的0、1、2、3槽位中。
2. 使用电源线缆连接控制器顶部的电源接口和电源模块。
3. 使用网线连接控制器网口ETH1和用于编程的计算机。

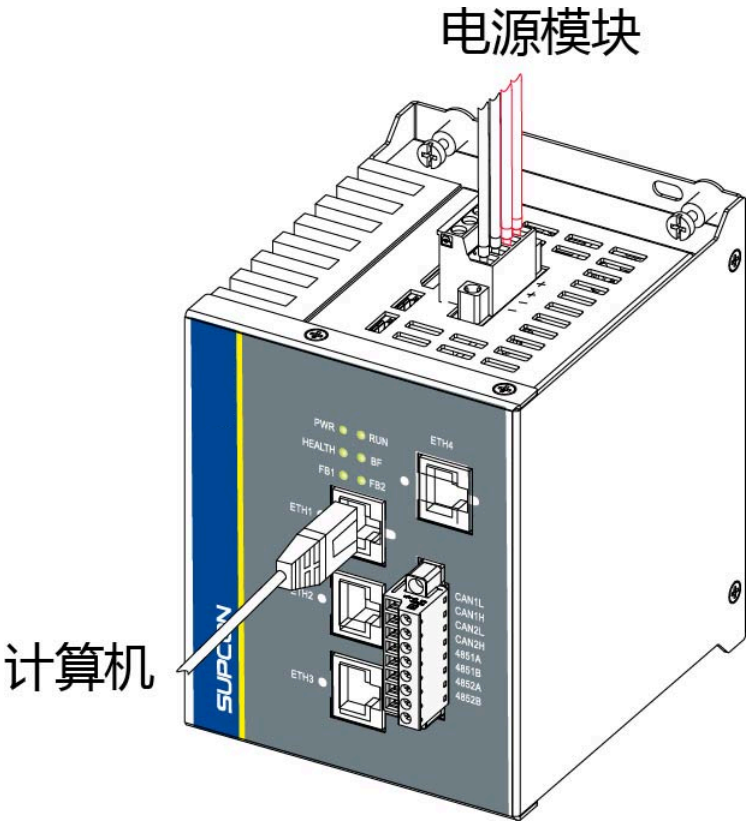


图 **5-25** 最小系统接线

5.4 启动编程环境

本节说明启动编程环境的准备工作及其操作步骤。

5.4.1 准备工作

- 执行启动前，请根据MotionPro软件的应用指南中的要求和操作指导，在计算机中正确安装软件。
- 获取功能扩展包：“GCS-M4_年月日.package”。详情请咨询中控运维或技术人员，

5.4.2 操作步骤

1. 配置计算机的IP地址，此地址应与控制器的ETH1网口地址（172.20.1.2/255.255.255.0）属于同一网段（172.20.1.3~254），掩码保持一致（255.255.255.0），示例如下图所示。

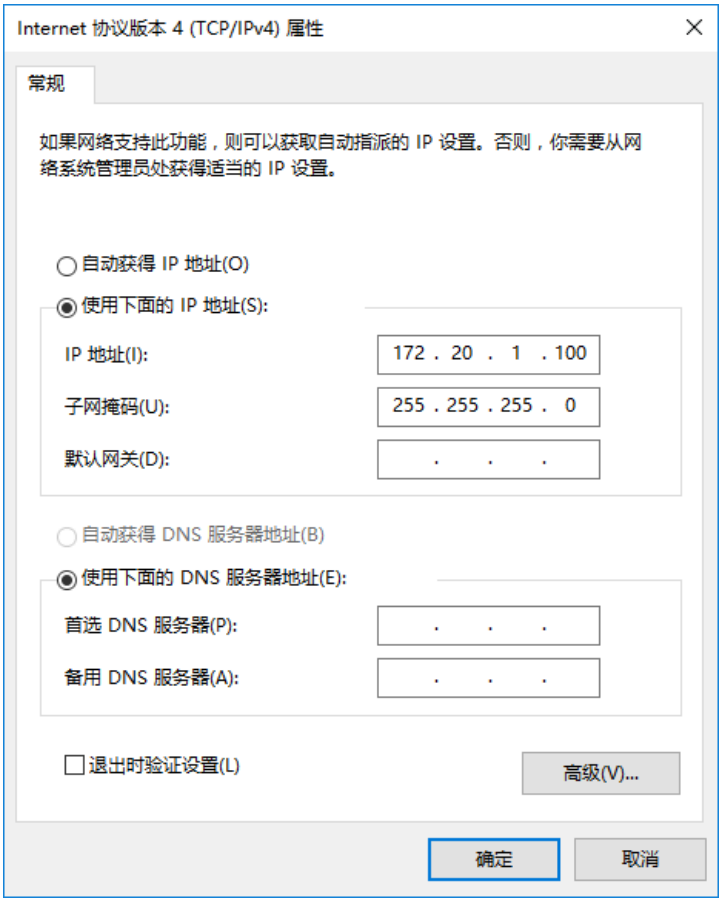


图 **5-26** 计算机**IP**地址配置示例

注意：

连接控制器的计算机网卡上不宜配置多个IP。

2. 右键管理员方式，打开计算机桌面上MotionPro图标 

，启动软件，初始页面如下图所示。

图 5-27 启动MotionPro软件

3. 单击“工具 > 包管理器”打开“包管理器”界面。单击“安装”，选中package包，单击“打开”，选择“完全安装”，等待界面提示“安装完成”。如下图所示，完成安装后包管理器界面出现安装完成的package信息，注意核对版本。

注意：

如果之前已经安装过某版本，请先卸载后再安装新版本package包。

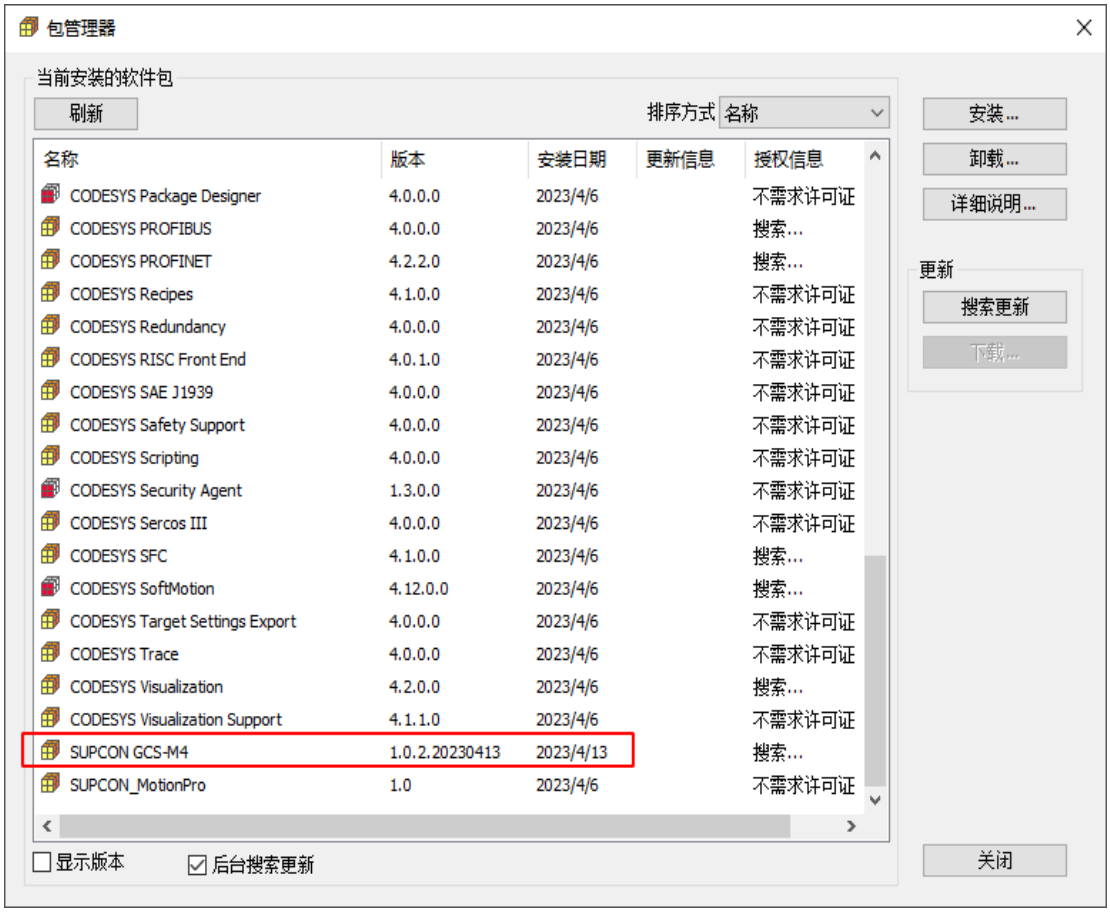


图 5-28 成功安装package后的界面

4. 单击MotionPro菜单栏左上角 新建工程，或者选择“文件 > 新建工程”，分类选择ZHEJIANG SUPCON，模版选择MCU4003 project，并指定工程文件名及保存路径，点击“确定”，如下图所示。

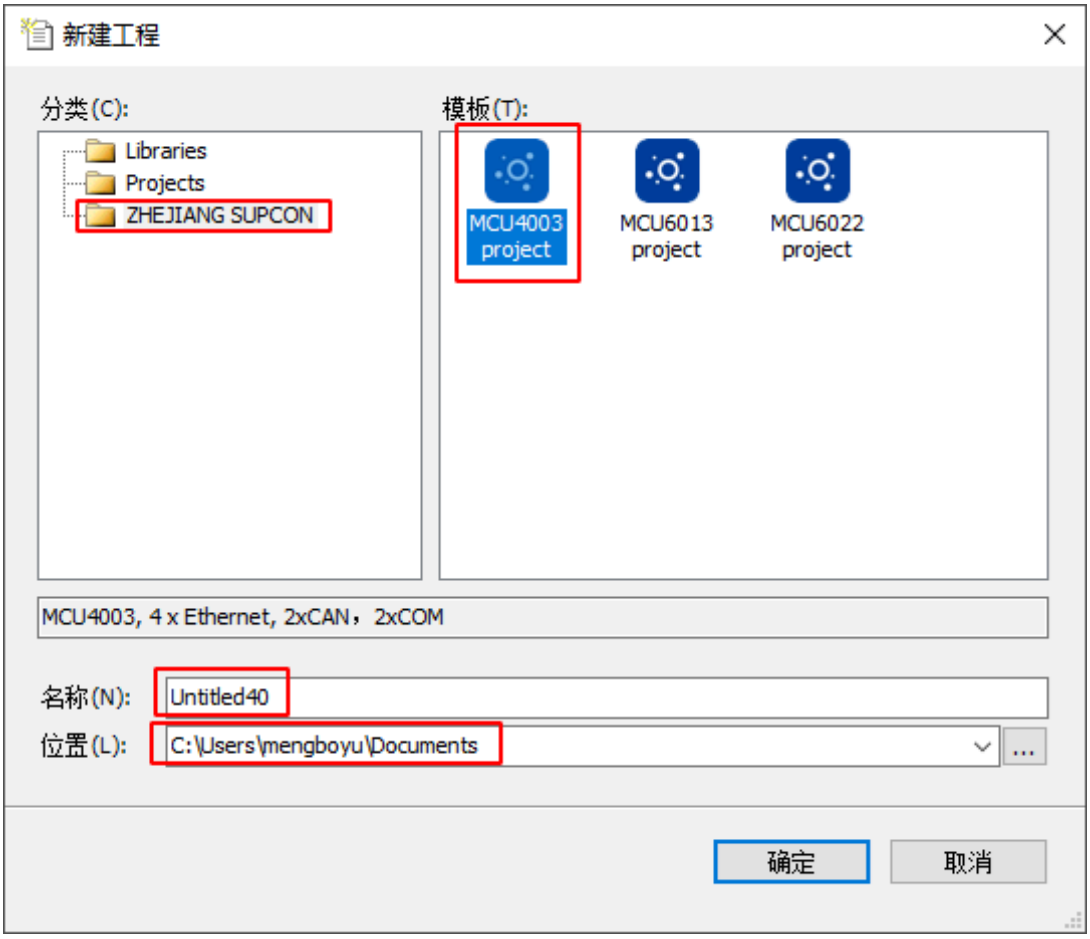


图 5-29 新建工程

5. 待工程新建完成后，进入系统组态配置与编程界面，常用的按钮与窗口分布如下图所示。

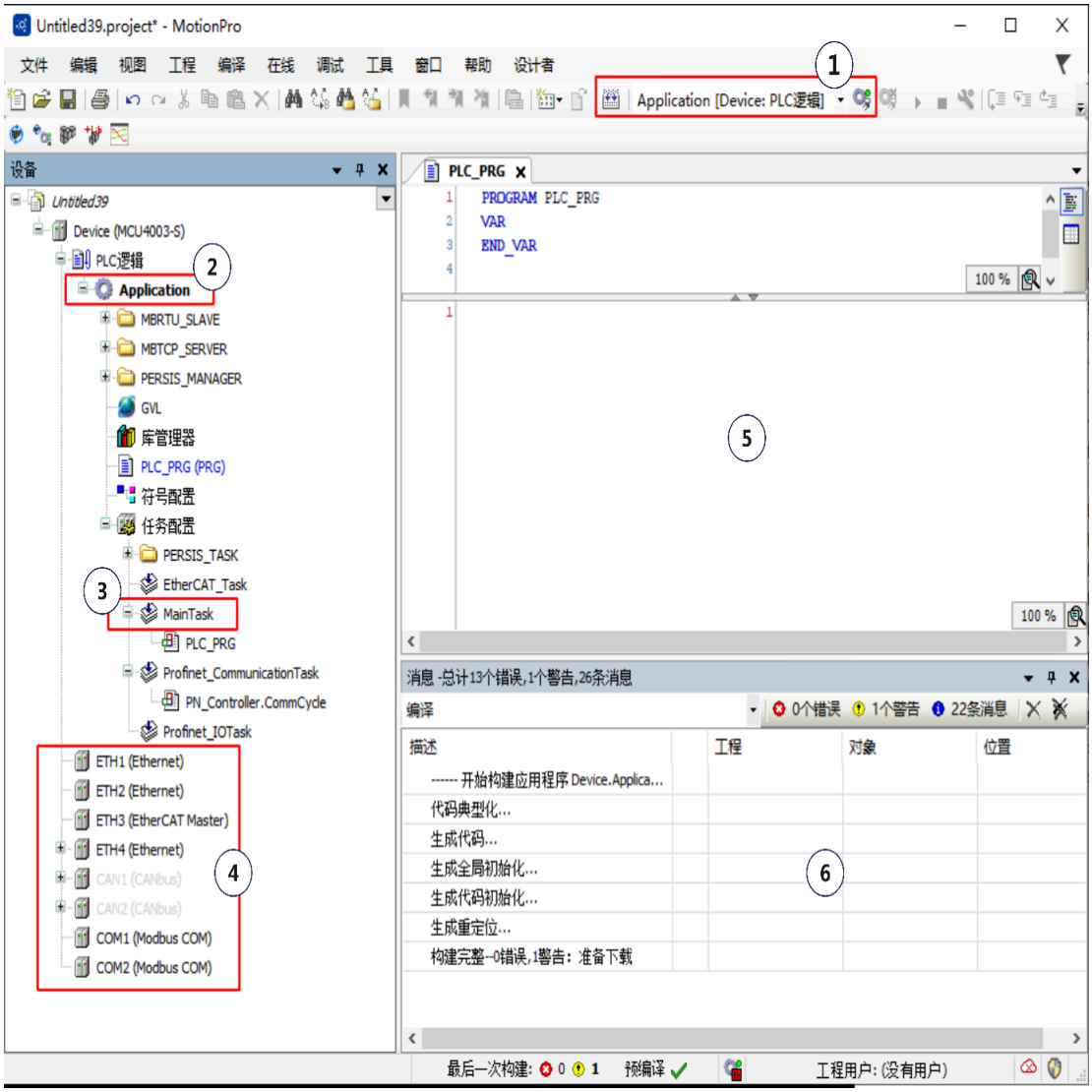


图 5-30 MotionPro主界面

①：编译，登录及调试
②：用户程序管理单元
③：配置任务执行方式及周期
④：网络及现场总线配置
⑤：用户程序编辑区
⑥：编译信息窗口

5.5 编写用户程序的典型步骤

本节说明常规情形下，编写用户程序的一般步骤。

5.5.1 编程的操作流程

初次使用控制器时，编写、调试用户程序的步骤如下所述。

1. 基于PLC应用系统的硬件链接架构配置硬件系统。
2. 根据应用系统的控制工艺编写用户程序。用户程序编程基于数据的存储宽度，使用范围自由定义变量，可以与硬件配置无关。
3. 将系统架构中的各硬件端口对应的输入端口变量（I）、输出状态（Q）关联用户程序中的变量。
4. 配置网络通讯的通讯周期（如EtherCAT/Profinet总线）等参数。根据各任务的实时性要求，配置用户程序单元的执行周期、优先级等。
5. 在MotionPro编程环境下登录PLC，下载用户程序，仿真调试、排错，确保系统正确无误地运行。

5.5.2 配置I/O系统

注意：

- **MCU4003-S系列PLC基于Profinet总线（即ETH4口下的PN_Controller）扩展本地及远程机架。您可通过自动扫描或手动添加的方式配置I/O系统。**
- **自动添加或手动添加的最终效果都相同。强烈推荐使用自动添加，可以较大程度避免配置错误。**

① 自动扫描方式

1. 在MotionPro的主界面，如下图所示，单击“ Device （ MCU4003-S ）”，在右侧的窗口点击“扫描网络”，输入 PLC 设备的用户名、密码，连接PLC系统。默认用户名supcon，密码root。

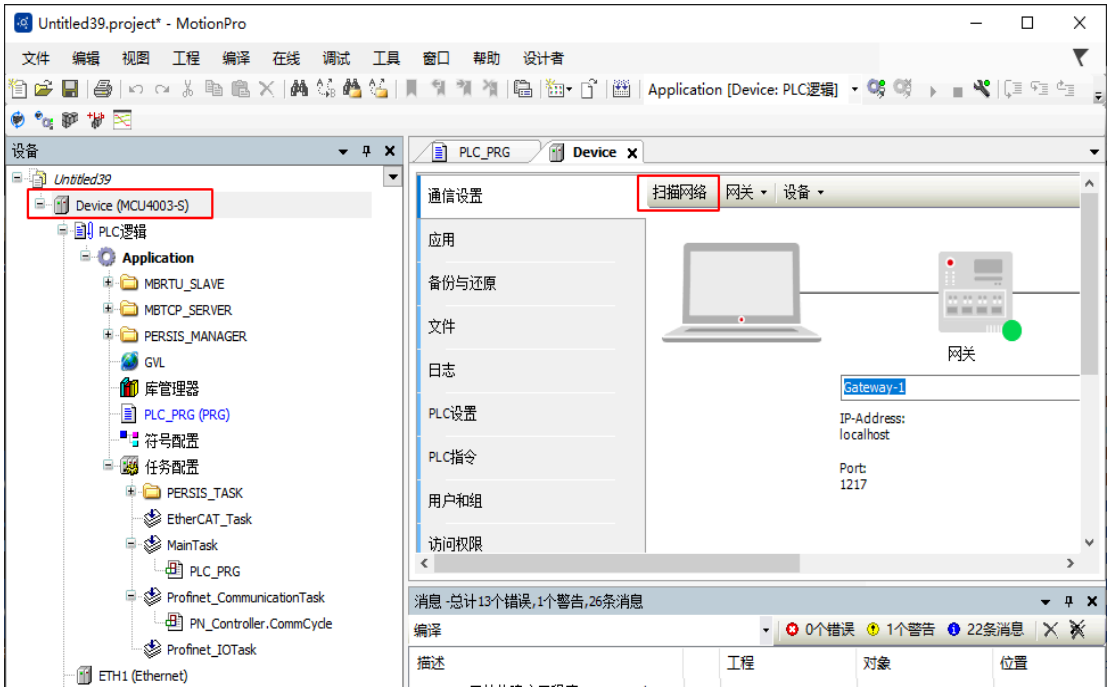


图 **5-31** 扫描PLC主控

2. 连接PLC后，在Device标签页，网关和控制器图标右下角出现绿色圆圈标识，且控制器图标下方出现“ MCU4003 （激活）”，表示已成功连接PLC设备，如下图所示。

图 **5-32 PLC已连接**

3. 成功连接PLC后，单击 编译工程，单击 下载PLC工程。完成下载后，点击 运行工程。如下图所示，MotionPro 的主界面显示工程正在运行。

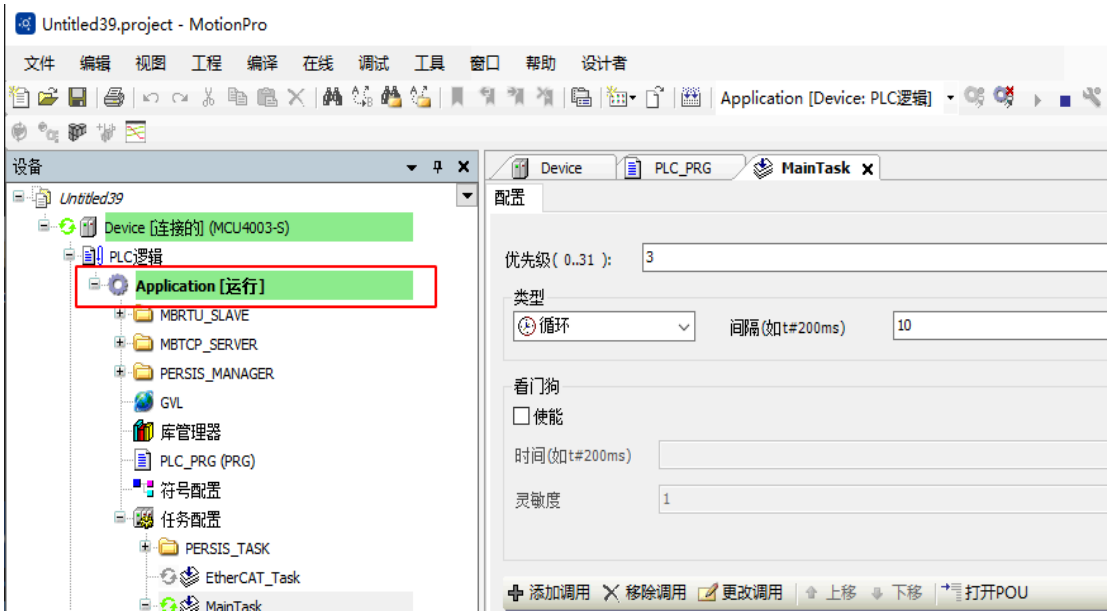


图 **5-33 PLC正在运行**

注意：

如果单击  编译时，出现编译错误的提示如下图中①，表示**PN_Controller**的**IP**配置与**ETH4**不匹配。此时，可在**PN_Controller**配置界面（下图中②）单击“调整”，消除错误。

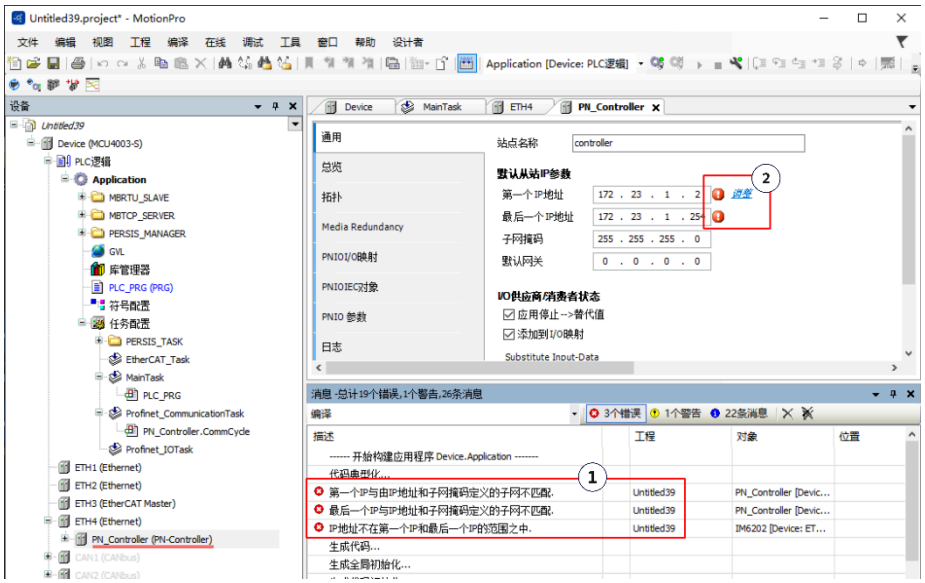


图 **5-34** 消除网段不一致错误

4. 在工程运行时，如下图所示，右键单击 PN_Controller，选择“扫描设备”。

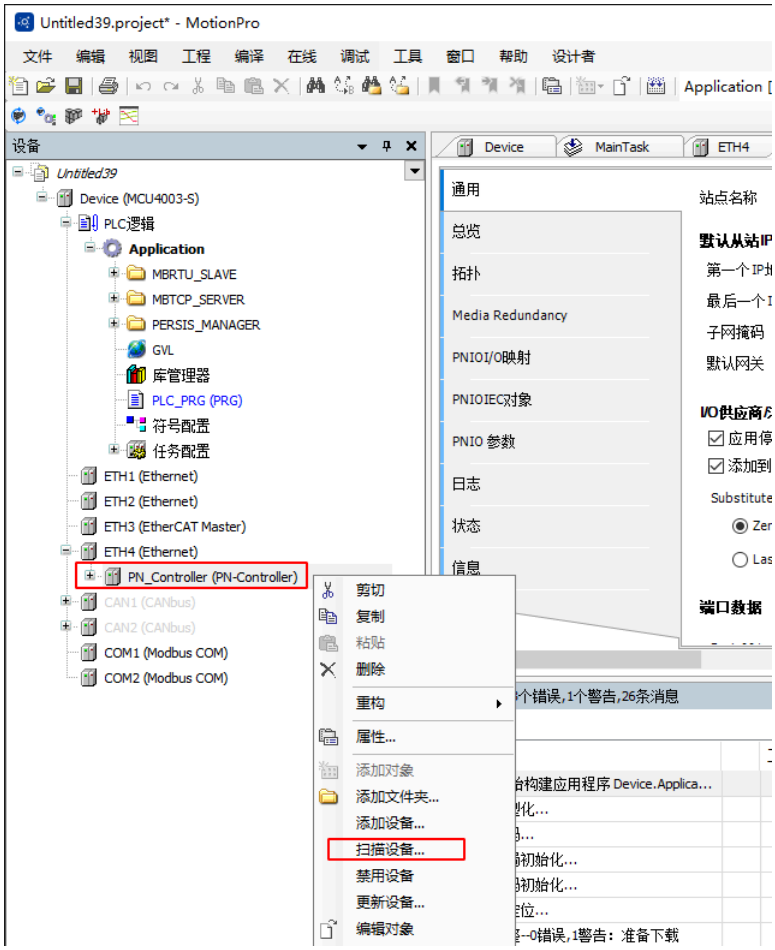


图 **5-35** 扫描**Profinet**设备

5. 完成扫描后，如下图所示，单击“复制所有设备到工程”，完成设备的添加。

注意：

如果出现不可识别的设备，原因可能是未使用最新版的**package**文件，请咨询运维或技术人员。

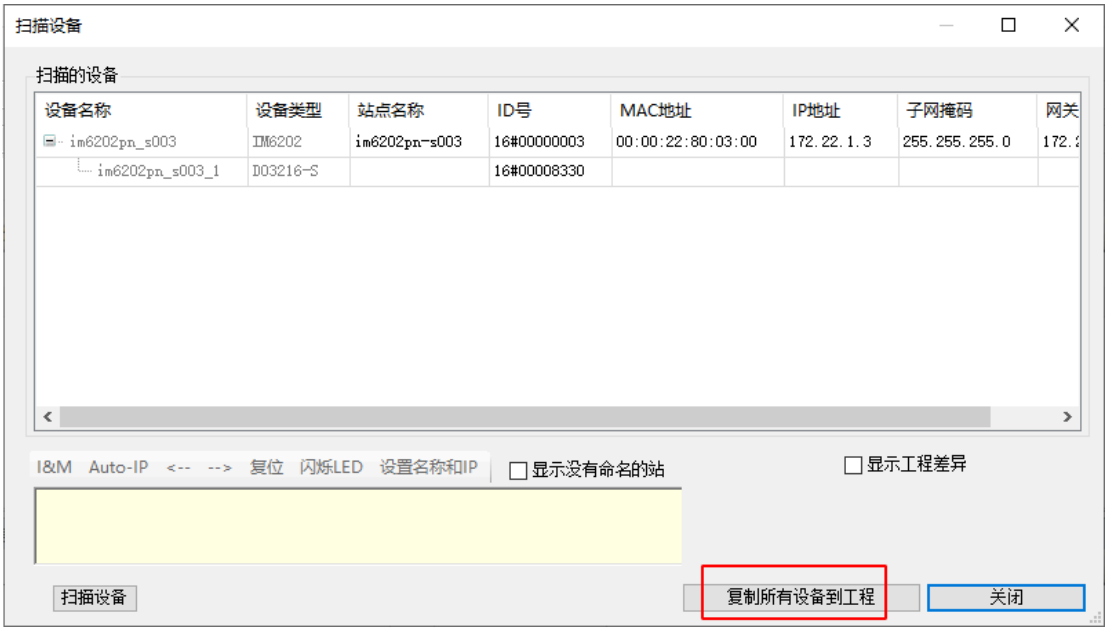

扫描设备
扫描的设备
设备名称
设备类型
站点名称
ID号
MAC地址
IP地址
子网掩码
网关
im6202pn_s003
IM6202
im6202pn-s003
16#00000003
00:00:22:80:03:00
172.22.1.3
255.255.255.0
im6202pn_s003_1
DO3216-S
16#00008330
I&M
Auto-IP
<--
-->
复位
闪烁LED
设置名称和IP
显示没有命名的站
显示工程差异
扫描设备
复制所有设备到工程
关闭

图 **5-36** 添加设备

注意：

a. 如下图所示，此处需要配置**IM6202PN**的“IP参数”中的“网关地址”为**ETH4**口地址（默认为**172.22.1.2**）。

b. 此外**IM6202PN**支持最快的**Profinet**总线周期**4ms**，所以在**IM6202PN**的配置界面，推荐设置：

- 发送时钟（**ms**）：**1**
- 减速比：**4**
- 看门狗（**ms**）：**1000**

c. 每次添加设备后都需要核对此处配置是否，确保不低于推荐配置。

图 **5-37** 设置**IP**参数

6. 完成自动添加。

② 手动添加方式

1. 新建工程后，右键单击“IM6202”下的<空>槽，单击“插入设备”，如下图所示。

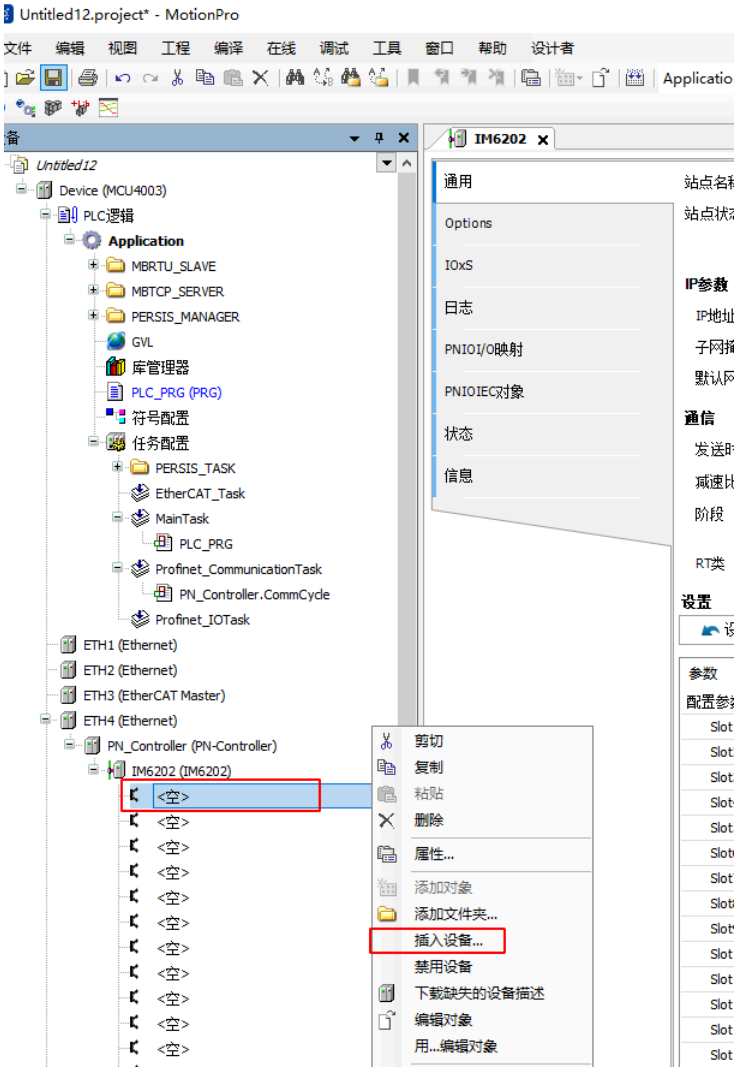


图 5-38 手动添加Profinet从站设备

2. 在弹出的“添加设备”窗口中，如下图所示，选择要添加的IO模块，点击“插入设备”。添加位置需要与实际物理位置一致。

图 **5-39** 手动添加设备

3. 依次按机架的实际配置，按槽位顺序添加模块。完成手工添加操作。

③配置实例

本小节介绍实际的模块分布与MotionPro中的最终配置效果。以1条控制器机架与1条远程机架为例，实际的模块分布如下表所示。

表 **5-1** 控制器机架与远程机架的模块分布表

机架槽位	本地机架（含控制器）	远程机架
1	MCU4003-S	IM3202PN
2		DO3216
3		DO3216
4		DO3216
5	DO3216	DO3216
6	DO3216	DO3216

表 **5-1** 控制器机架与远程机架的模块分布表（续）

机架槽位	本地机架（含控制器）	远程机架
7	DO3216	DO3216
8	DO3216	DI3216
9	DO3216	DI3216
10	DI3216	DI3216
11	DI3216	DI3216
12	DI3216	DI3216
13	DI3216	DI3216
14	DI3216	-
15	DI3216	-
16	AI3208	-
17	AI3208	-
18	AI3208	-
19	AO3208	-
20	AO3208	-

MotionPro配置界面如下图所示。

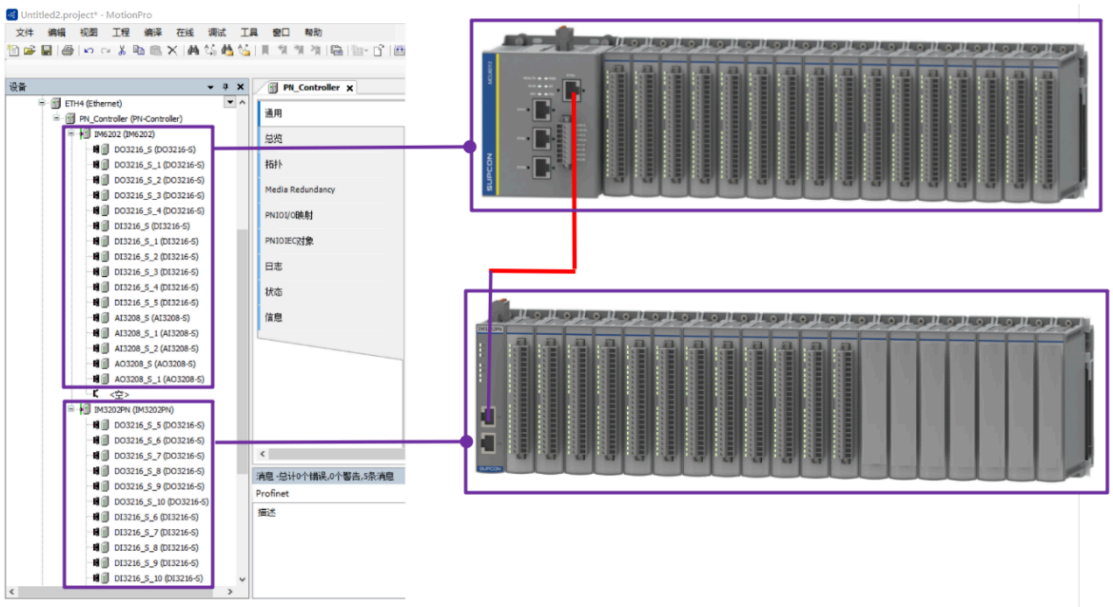


图 **5-40** 双机架配置实例

5.5.3 编写用户程序

1. 双击左侧设备树窗口中的“PLC_PRG(PRG)”项，即可打开用户编程界面。编程语言为ST（新建工程时选择），如图 5-41 所示。

提示：
ST编程与C语言相似，每个变量需要声明后才能使用。您可先编写程序语句，回车时编程环境会自动弹出声明框。此时填写声明框并点击“确定”后，变量声明窗口会自动增加该变量的声明语句，简化编程。

2. 完成程序编写后，点击  按钮编译工程，未报错则表示编译通过。

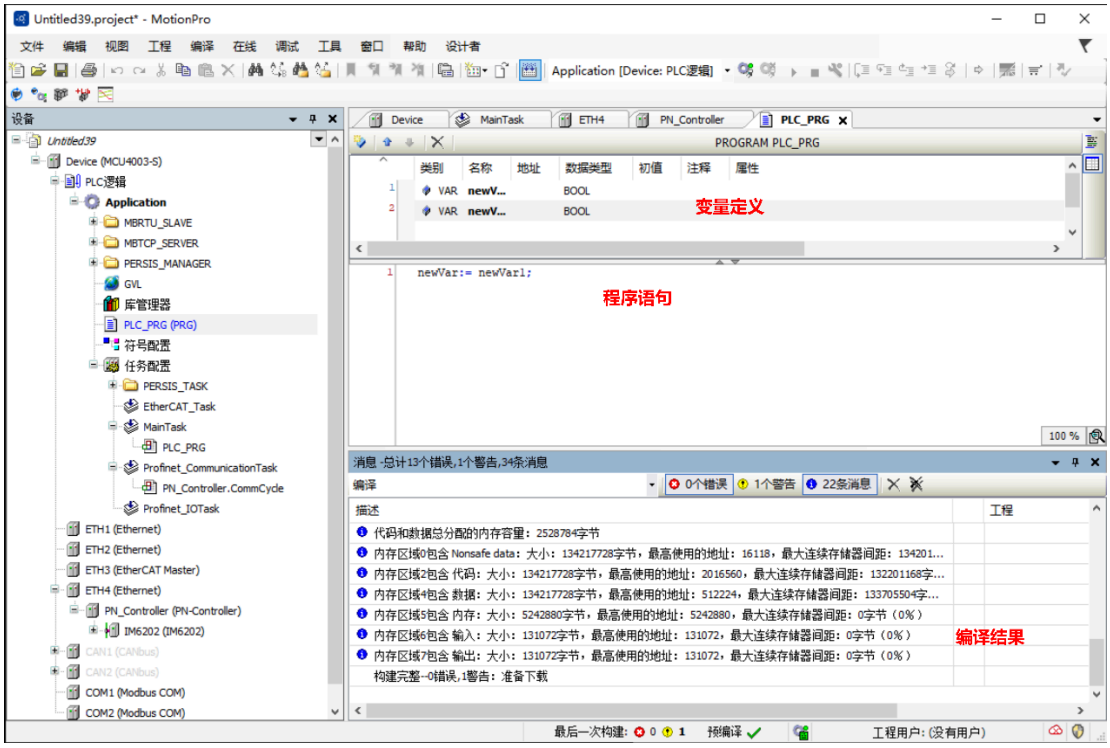


图 **5-41** 程序编写及编译

5.5.4 关联程序变量与硬件端口

在PN_Controler配置页面中，可关联硬件端口与用户程序中的变量。

如下图所示，将程序“PLC_PRG”定义的变量“newVar”，关联到控制器本地机架第5个槽位的DO3216模块的第1个通道，配置步骤顺序如图 5-42 所示。

完成关联后，对变量的操作相当于对硬件端口的操作。

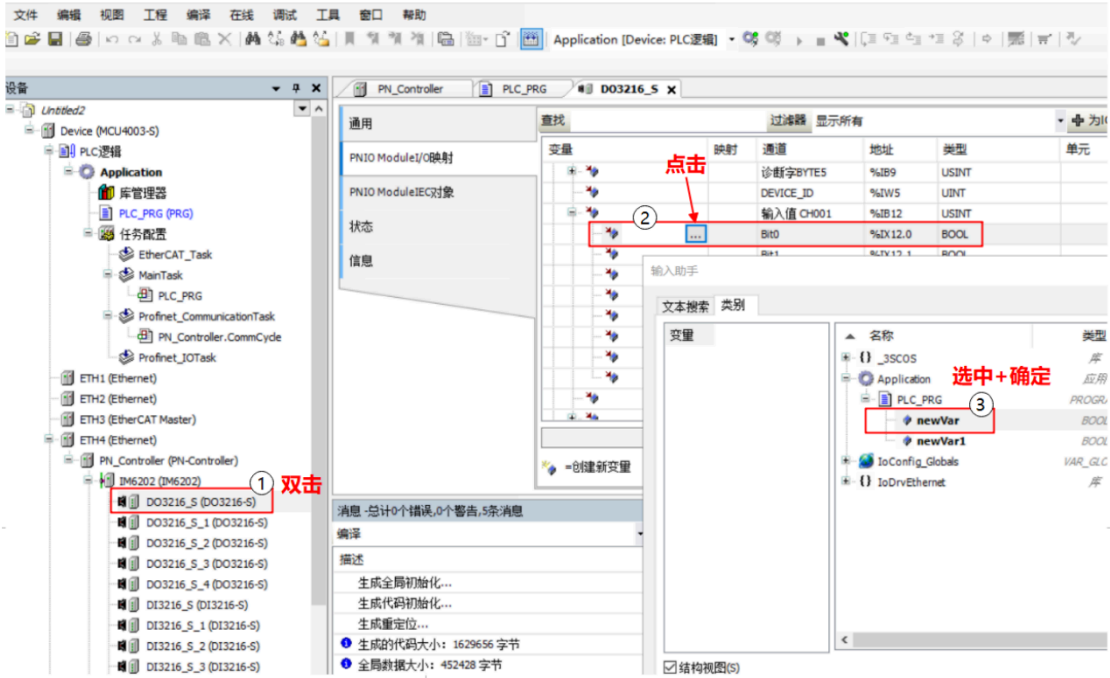


图 **5-42** 关联程序变量与硬件端口

5.5.5 配置用户程序的执行方式和运行周期

如图 5-43 ①，双击任务名，并在右侧的配置界面中进行如下操作：

1. 配置优先级，图中②，1最高，31最低。
2. 配置任务类型，图中③，支持“循环、事件、惯性（即无周期连续运行）、状态”。
3. 配置任务间隔，图中④，设置当任务类型为“循环”模式时的任务周期时间。
4. 配置任务看门狗，图中⑤，使能后如果任务运行时间超出看门狗时间，程序将停止运行。

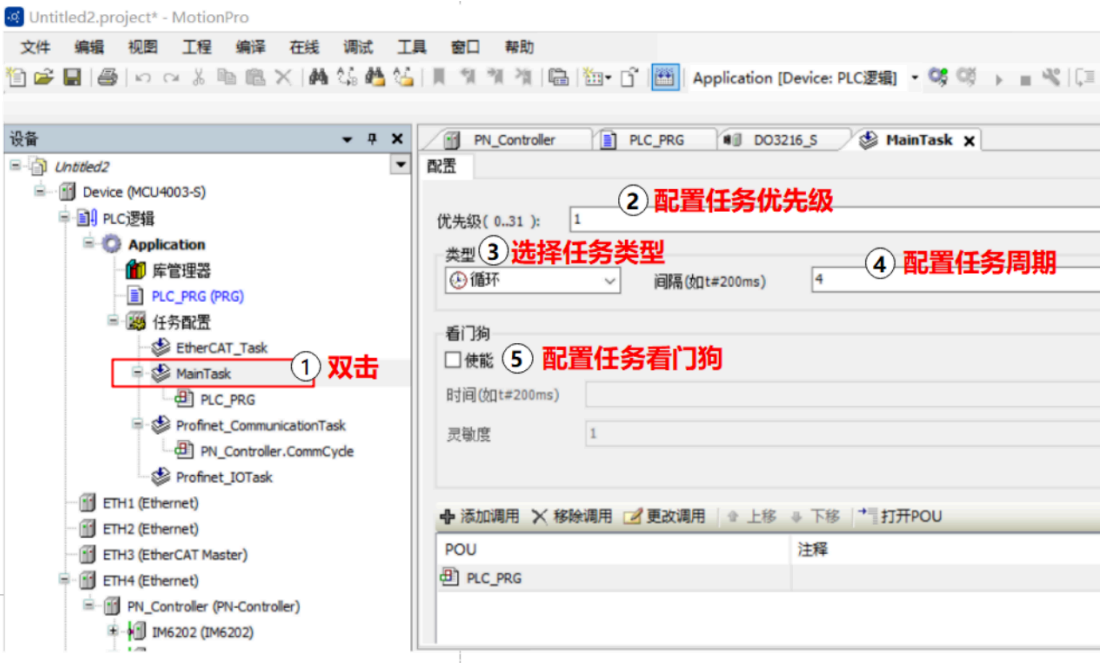


图 **5-43** 任务配置

5.5.6 用户程序的编译和下载

1. 完成编程后下载前，应编译程序，确认程序中没有错误。
   如用户程序的编译和下载所示，可在编译信息窗口查阅编译结果。如果程序中有错误，可单击错误信息行，软件将自动定位到用户程序的报错点，修改并排除所有错误，直至编译通过。

图 5-44 查看编译结果

2. 编译无误后，点击菜单栏“在线 > 登录”或工具栏  按钮。
3. 在弹出的对话框中，选择“是”，上位机与设备建立连接并保持，完成连接后，PLC程序默认进入“停止”状态。
4. 点击“调试 > 启动”，设备进入运行状态，并开始执行用户程序，此状态下，可调试系统。

6 数据掉电保持

本节介绍控制器的数据掉电保持功能。

6.1 功能描述

- 掉电保持容量512KB，可保持15000个标准类型变量，支持保持复合类型变量（数组，结构体等）。
- 保存周期为1秒。

保持逻辑如下：

表 6-1 保持逻辑

操作	保持型位号
热启动	保持
冷启动	保持
在线修改	保持
下载	保持
复位原点	恢复初值
位号类型修改	恢复初值

6.2 掉电保持位号配置方法

需要进行掉电保持的数据，在表格视图下，给位号的属性标记attribute 'ac_persist':='PersistenceChannel'即可，如下图所示。

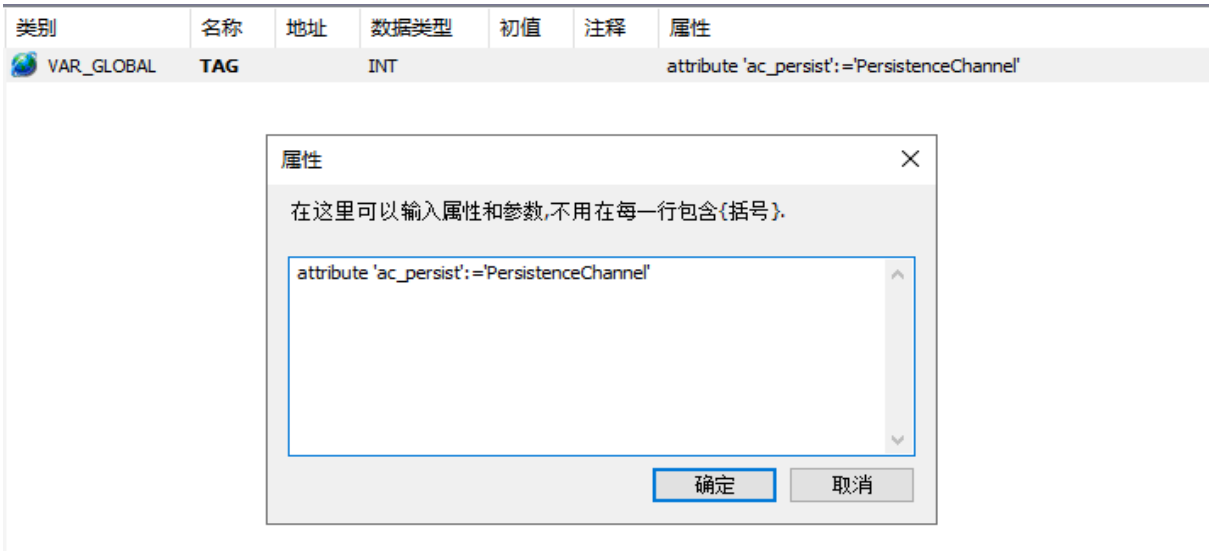


图 **6-1** 位号标记掉电保持属性（表格视图）

而在文本视图下，方法则为在位号上方添加（对比表格视图多了一对｛｝号）“{attribute 'ac_persist':='PersistenceChannel'}”。

```
{attribute 'qualified_only'}
VAR_GLOBAL
    {attribute 'ac_persist':='PersistenceChannel'}
    TAG: INT;
END_VAR
```

图 **6-2** 位号标记掉电保持属性（文本视图）

注意：

标记了掉电保持的位号，必须在程序中被引用才会生效。

7 诊断及显示

控制器模块的运行状态通过面板指示灯和WEB诊断页面显示。

7.1 模块状态指示灯

控制器模块面板上有一组模块运行状态指示灯，分布如下。

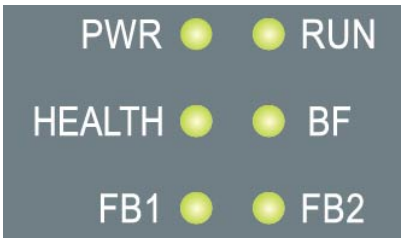


图 7-1 面板指示灯

控制器模块运行状态指示灯的现象和含义说明如下。

表 7-1 指示灯状态说明

指示灯标识	颜色	定义	故障排查
PWR （电源指示灯）	常灭	电源未接通	-
	常亮	电源已接通	-
RUN （运行状态指示灯）	绿灯常亮	用户程序正常运行	-
	红灯常亮	用户程序停止运行	-
	绿灯闪烁	没有组态	-
HEALTH （硬件状态指示灯）	绿色常亮	硬件正常	-
	红色常亮	硬件故障	1. 网口故障，尝试重新上电，若未恢复则需返厂 2. CPU温度过高，检测环境温度是否超过65℃，如超过需加强通风条件；若未超过则需返厂
BF （底板总线状态指示灯）	绿色常亮	背板通信正常	-
	绿色闪烁	背板通信异常/无背板通信	检查控制器是否牢固安装在背板
FB1 （现场总线1状态指示灯）	绿色常亮	EtherCAT主站/Modbus TCP主站/Profinet主站均正常	-

表 7-1 指示灯状态说明（续）

指示灯标识	颜色	定义	故障排查
	红色常亮	EtherCAT主站/Modbus TCP主站/Profinet主站出现异常	1. 检查网线连接是否正常 2. 检查从站是否正常工作 3. 检查是否有网络IP冲突
FB2 （现场总线2状态指示灯）	绿色常亮	CANopen主站/Modbus RTU主站均正常	-
	红色常亮	CANopen主站/Modbus RTU主站出现异常	1. 检查通信线连接是否正常，是否接反，线缆通信情况下终端电阻是否位于线缆两端，阻值是否正确 2. 检查从站是否正常工作

7.2 以太网通信指示灯

以太网口处的LED指示灯现象和含义说明如下。

表 7-2 以太网指示灯说明

指示灯位置	功能	状态	说明
	A：Link	灭	网线未连接
		闪烁	已连接且有数据收发
		常亮	已连接
	B：Speed	灭	未连接/10Mbps连接
		绿色	100Mbps连接
		橘色	1000Mbps连接

7.3 GCS-M4 Tool工具进行诊断

本节介绍如何使用GCS-M4 Tool进行控制器诊断。

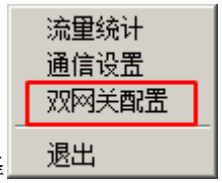
提示：

该工具请咨询中控技术人员获取。

7.3.1 连接方法

通过GCS-M4 Tool工具，可以从ETH1口查看控制器的设备信息和诊断信息。方法如下。

1. 设置电脑IP，使得电脑与控制器的ETH1口在同一网段，且掩码一致，能ping通。

2. 以管理员权限打开GCS-M4 Tool软件，在电脑右下角右键点击  图标，选择 

，在弹出的“双网关配置”界面，在“A网”处，选择相应电脑网口的IP，点击保存。如果一个网卡有多个IP，则选择与控制器 ETH1口网段一致的IP。

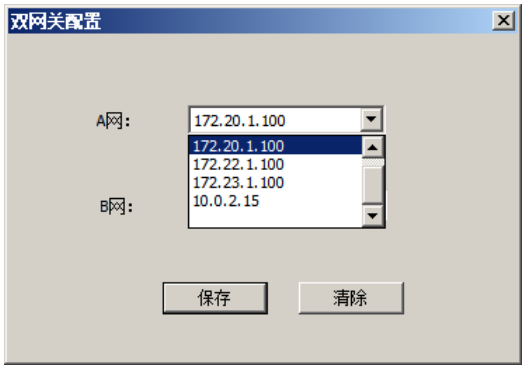


图 7-2 双网关配置

3. 在GCS-M4 Tool软件，点击菜单栏“扫描 > 添加节点”，在A网地址处填入控制器的ETH1口IP，勾选“IP自由设置”，设备类型选择“G3/G5”，点击确定，即可将控制器连接上来。

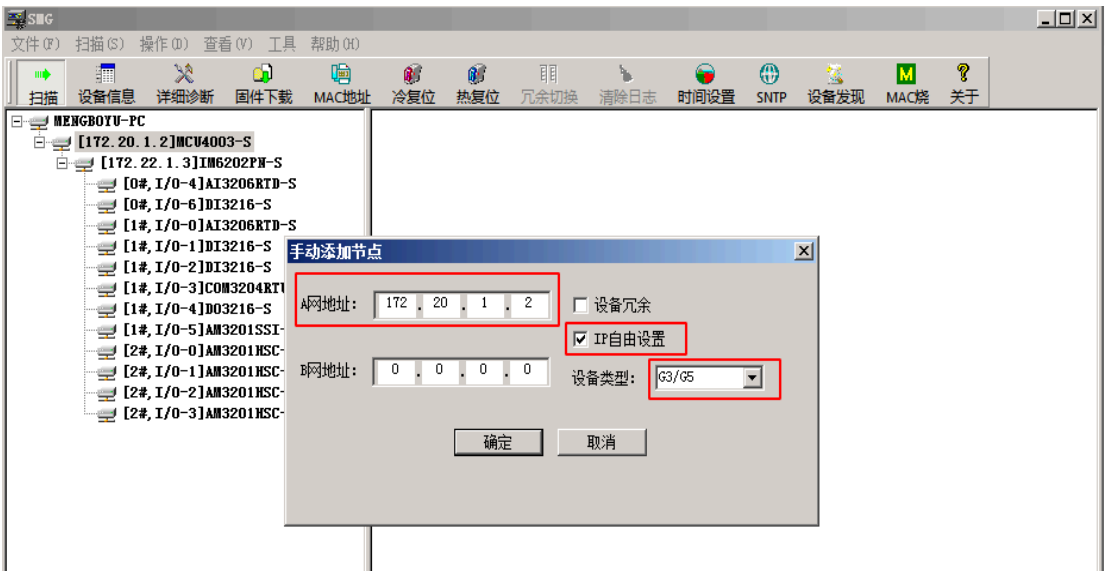


图 7-3 手动添加节点

7.3.2 诊断信息

点击选择 MCU4003-S （点中后灰色高亮），点击“详细诊断”，可以查看到控制器的诊断信息。

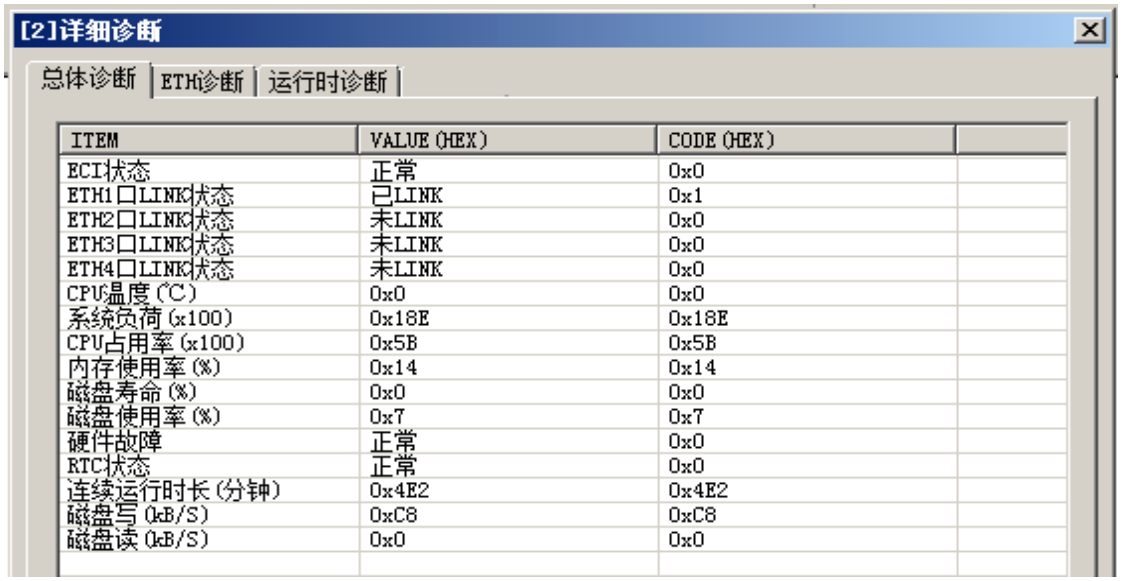


图 7-4 控制器详细诊断

控制器详细诊断信息如下表所示。

表 7-3 总体诊断

诊断项	说明
ECI状态	背板工作状态，值：正常/异常
ETH1口LINK状态	ETH口连接状态，值：已LINK/未LINK
ETH2口LINK状态	
ETH3口LINK状态	
ETH4口LINK状态	
CPU温度（℃）	单位℃
系统负荷（×100）	系统负荷
CPU占用率（×100）	CPU占用率
内存使用率（%）	百分比
磁盘寿命（%）	磁盘累计写入量占磁盘寿命百分比
磁盘使用率（%）	磁盘已使用空间百分比
硬件故障	正常/异常
RTC状态	RTC时钟工作状态，值：正常/异常

表 7-3 总体诊断（续）

诊断项	说明
连续运行时长（分钟）	从上电后连续运行的时间，单位分钟
磁盘写（kB/S）	当前磁盘每秒实时写入量
磁盘读（kB/S）	当前磁盘每秒实时读出量

表 7-4 ETH诊断

诊断项	说明
ETHX接收计数	网口接收包计数
ETHX接收错误	-
ETHX接收超限计数	-
ETHX接收帧错误计数	-
ETHX发送计数	网口发送包计数
ETHX发送错误计数	-
ETHX发送丢弃计数	-
ETHX发送carrier计数	-
ETHX发送冲突计数	-

表 7-5 运行时诊断

诊断项	说明
PLC APP状态	组态状态，值：无组态/停止/运行/故障

表 7-6 CAN诊断

诊断项	说明
CANX接收计数	-
CANX发送计数	-
CANX总线错误计数	-
CANX仲裁失败计数	-
CANX主动错误计数	-
CANX被动错误计数	-

表 **7-6 CAN**诊断（续）

诊断项	说明
CANX总线关闭计数	-

8 编程基础

本节说明使用MotionPro软件编程所需掌握的基本概念。

8.1 概述

操作数是用户程序中操作符、功能、功能块或者程序操作的对象，可以作为输入、输出、中间保存结果。在MotionPro中，常见的操作数包含直接地址、常量和变量。

与其他高级语言类似，MotionPro也有常量和变量的概念。

- 常量就是数值不变的数。
- 变量是由用户定义的标识符。变量的存储位置可由用户指定为%I区、%Q区、%M区的特定地址，亦可不指定地址，由系统自行分配，用户不需要关注这些变量的存储位置。

8.2 直接地址

此类型固定地址也叫直接变量，直接映射到PLC设备的具体地址。地址信息包含了变量在CPU的存储位置，存储大小及存储位置对应的偏移。

语法

%< 存储器区前缀>< 大小前缀>< 数字>|.< 数字>

- < 存储器区前缀 >：编程系统支持以下存储区前缀
  1. I：输入，物理输入，“传感器”
  2. Q：输出，物理输出，“执行器”
  3. M：存储位置
- < 大小前缀 >：编程系统支持以下大小前缀
  1. X：Bit，一位
  2. B：Byte，一个字节
  3. W：Word，一个字
  4. D：Double Word，四个字节（双字）
- < 数字 >|.< 数字 >
  第一个数字是存储区前缀的偏移地址，如果定义BOOL 类型变量，需使用< 数字>.< 数字> 格式，“. ”后的数字是偏移地址的偏移位数。
  示例：
  - %QX7.5 输出区域偏移7个字节的第六位（bit5）
  - %QB17 输出区域偏移17个字节
  - %IW215输入区域偏移215个字
  - %MD48内存区域偏移48个双字
  - iVar AT %IW10: WORD;//iVar变量是字类型，映射到输入区域偏移10字的位置

8.3 变量

本节介绍变量的概念。

8.3.1 概述

变量可以在POU的定义部分或者通过自动声明对话框定义，也可以在DUT 或者 GVL 编辑器定义。通过变量类型关键字来标识变量类型，例如通过VAR和END_VAR来标识它之间定义的变量为本地变量。

变量类型包括本地变量(VAR)，输入变量(VAR_INPUT)，输出变量(VAR_OUTPUT)，输入输出变量(VAR_IN_OUT)，全局变量(VAR_GLOBAL)，临时变量(VAR_TEMP)，静态变量(VAR_STAT)，配置变量(VAR_CONFIG)。

8.3.2 变量定义

变量可以在声明编辑器中定义。声明编辑器有文本视图和表格视图两种显示形式。此外，支持结构体和数组复杂数据类型，支持变量定义和数组元素注释，支持复杂数据类型地址递归显示。

文本声明如下图所示。

```
VAR_GLOBAL
    A AT%MW0:DUT := (A_0 := [2(FALSE), TRUE, 7(FALSE)]);
    B AT%MW100:DUT;
    C AT%MW400:DUT;
END_VAR
```

图 8-1 文本声明

表格声明如下图所示。

类别	名称	地址	数据类型	初值	注释	属性
VAR_GLOBAL	A	%MW0	DUT	(A_0 := [2(FALSE), TRUE, 7(FALSE)])		
VAR_GLOBAL	B	%MW100	DUT			
VAR_GLOBAL	C	%MW400	DUT			

图 8-2 表格声明

表格声明中，可以对变量的各类属性进行编辑设置。下表是对表格声明的具体说明。

表 8-1 表格声明

项目	描述
类别	变量的类别（如本地变量，输入变量，输出变量，临时变量等）
名称	变量的名称
地址	变量编译后的地址

表 8-1 表格声明（续）

项目	描述
数据类型	即变量的数据类型（如INT，BOOL等）
初值	变量的初始值
注释	变量的注释
属性	变量的属性

8.4 常量

在PLC编程的时候，会用到一些数值不变的参数，如定时器的时间、换算的比例等，这些数值不变的参数称为常量。

常量声明语法

- VAR CONSTANT
- <identifier>:<type> := <initialization>;
- END_VAR

示例

- VAR CONSTANT
- c_iCon1:INT:=12;
- END_VAR

MotionPro支持多种数据类型的常量，常见的常量包括布尔型、整形、时间型、字符串等。具体常量见下表。

表 8-2 常见常亮

类型	描述	示例
布尔类型	有两个值TRUE和FALSE（也可以用1和0）。1=TRUE，0=FALSE	TRUE，FALSE，1
BIT类型	和布尔类型相似，只能在结构体（占用位数）或者功能块（映射BOOL类型的直接地址）中使用	TRUE，FALSE，0

表 8-2 常见常亮（续）

类型	描述	示例
整数	整数常量的数值可以是二进制、十进制、八进制和十六进制。如果整数值不是十进制值，可以用“进制”，加符号“#”放在整数值前面来表示。十进制的10至15在十六进制中表示为A至F	十进制：66 二进制：2#101 八进制：8#72 十六进制：16#3A 类型常数：INT#22、BYTE#204
浮点类型	浮点常量用十进制小数和指数来表示，遵循标准的科学计数法格式	7.4 2.3e+9 REAL#3.12
ASCII字符串	ASCII字符串常量在两个单引号之间，可以包含空格和特殊字符。一个字符用一个字节表示，只支持ASCII码字符（不支持中文字符）。默认最大长度80字符，超过最大长度时将去除多余字符。可以声明字符最大长度，如str: STRING(35):=’This is a String’; 字符串函数最大支持255字符。	$做转义字符例： ‘$30’：0，字符0，16进制30对应的ASCII字符 $$：$，美元字符 $’：’，单引号
UNICODE字符串	UNICODE字符串常量在两个双引号之间，一个字符占两个字节，只支持UNICODE字符（支持中文字符），默认最大长度80字符，超过最大长度时将去除多余字符。可以声明字符最大长度，如wstr:WSTRING(35):=”This is a WString”;	“Unicode string”
时间	时间常量一般用来操作时间， 由“T#”（或“t#”）加上“时间值”构成，时间值的单位包括天（d）、小时（h）、分（m）、秒（s）和毫秒（ms）	t#12h34m15s
时刻	一天的时间范围。语法：TOD#时间值	TOD#15:36:30.123
日期	从1970年1月1日开始。语法：d#日期	d#2015-02-12
日期时刻	日期常量和时刻常量合并起来称为日期时刻常量，由从1970年1月1日开始。语法：dt#日期	dt#2004-03-29-11:00:00

9 编程语言

本节介绍MotionPro中支持的编程语言，以及各语言的详细用法。

9.1 MotionPro支持的编程语言

MotionPro编程软件支持下列PLC编程语言：

- Ladder diagram(LD) 梯形图
- Function block diagram(FBD)功能块图
- Structured text (ST) 结构化文本
- Sequential function chart (SFC) 顺序功能图
- Continuous function chart (CFC) 连续功能图
  其中，LD、FBD、ST、SFC是基于IEC 61131-3标准，CFC是IEC 61131-3标准的一个扩展。
  不论用户选用哪种语言，编程界面中的基本编辑方法是通用的，为编程提供极大便利。
- 标准的编辑器功能，如支持“复制”（Ctrl+C）、“粘贴”（Ctrl+V）和“删除”（Del）等快捷键
- 标准的<Ctrl>、<Shift>按键多项选择；
- 支持功能键<F2>启动输入助手，系统根据具体环境提供相对应的输入提示或选择

9.2 结构化文本语言（ST）

本节介绍结构化文本语言（ST）。

9.2.1 概述

结构化文本是一种文本化的高级语言，跟PASCAL或C类似。程序代码由指令组成，指令由关键字和表达式组。不同于IL语言，ST语句循环中可以包含众多的语句，因此允许开发复杂的结构。

例如

```
IF value < 7 THEN
WHILE value < 8 DO
value := value +1;
END_WHILE;
END_IF;
```

9.2.2 表达式

表达式是一种结构，对它求值后，这个值可以在指令中使用。表达式由操作符和操作数组成。一个操作数可以是一个常量、变量、功能调用或其他表达式。

例如：

- 常量，例如：20，t#20s，‘string’
- 变量，例如：iVar，Var1[2,3]
- 功能调用，值为调用返回值，例如：Fun1(1,2,4)
- 其它表达式：10+3，var1 OR var2，(x+y)/z，iVar1:=iVar2+22

表达式的求值以特定的操作符优先权定义的顺序，按操作符对操作数进行求值。表达式中具有最高优先权的操作符应首先进行求值，接着是下一个较低优先权的操作符等，从高到低依次求值完成。优先权相等的操作符应按表达式中书写的从左到右的顺序进行。

例子：若A、B、C和D属于类型INT，并分别具有值1、2、3、4，那么A+B-C*ABS(D)应等于-9，而(A+B-C) *ABS(D)应等于0。

当操作符具有两个操作数时，应首先对最左边的操作数求值。例如，在表达式SIN(A)*COS(B)中，应先对表达式SIN(A)求值，其次是对COS(B)求值，最后是积的求值。

下表记录了ST语言的操作符：

操作	符号	优先权
括号	(表达式)	高 ↓ 低 依次降低
函数调用	函数名(参数列表，由逗号分隔)	
求幂	EXPT	
求负值 求补	- NOT	
乘 除 取余	* / MOD	
加 减	+ -	
比较	<, >, <= , >=	
等于 不等于	= <>	
逻辑与	AND	
逻辑异或	XOR	
逻辑或	OR	

9.2.3 ST指令

整个ST程序由指令构成，指令由分号“；”分隔。这些指令由关键字和表达式组成，ST指令如下表。

表 9-1 ST指令一览表

指令	说明	示例
:=, S=, R=	赋值，置位，复位	A:=B; C S= cond0; b1 R=cond1;
功能块调用	功能块调用和输出	CMD_TMR :TON（CMD_TMR.Q为定时器输出状态） CMD_TMR(IN := %IX5, PT := 300); A:=CMD_TMR.Q
RETURN	返回（退出当前POU）	RETURN;

IF	选择	D:=B*B;
		IF D<0.0 THEN
		C:=A;
		ELSIF D=0.0 THEN
		C:=B;
		ELSE
		C:=D;
		END_IF

CASE	多重选择	CASE INT1 OF 1: BOOL1 := TRUE; 2: BOOL2 := TRUE; ELSE BOOL1 := FALSE; BOOL2 := FALSE; END_CASE;

FOR	FOR循环	J:=101;
		FOR I:=1 TO 100 BY 2 DO
		IF ARR[I] = 70 THEN
		J:=I;
		EXIT;
		END_IF;
		END_FOR;

WHILE	WHILE循环	J:=1; WHILE J<= 100 AND ARR[J] <> 70 DO J:=J+2; END_WHILE;
REPEAT	REPEAT循环	J:=-1; REPEAT J:=J+2; UNTIL J= 101 OR ARR[J] = 70 END_REPEAT;

表 9-1 ST指令一览表（续）

指令	说明	示例
EXIT	退出循环	EXIT;
CONTINUE	继续循环下次执行	CONTINUE;
JMP	跳转	label: i:=i+1; JMP label;
;	空语句	;

9.2.4 赋值指令

赋值指令用于变量赋值，也就是赋值关键字的左边是变量，右侧为要赋的值，通过赋值关键字进行赋值，赋值关键字包含三种：“:=”、“S=”、“R=”。

- “:=”为一般赋值，右值直接赋给左值，左值和右值相等。
  例如：Var1 := Var2 * 10;
  完成执行后，Var1值为Var2的10倍。
- “S=”为置位赋值，表示如果右值为TRUE，左值变量变为TRUE（置位），直到调用R=命令来初始化。
- “R=”为复位赋值，表示如果右值为TRUE，左值变量变为FALSE（复位）。用于复位S=指令置位的变量。
  例如：a S= b;
  一旦b为TRUE后，a会保持TRUE，即使b变为FALSE后。

9.2.5 功能块的调用

语法

<FB 实例名>(FB输入变量:=<值和地址>|, <更多FB 输入变量:=<值和地址>|...更多 FB 输入变量，FB输出变量 => <值和地址>|...更多 FB 输出变量);

调用语法

调用一个延时功能块(TON)的实例，分配输入参数IN和PT，功能执行后，可以把结果Q赋值到变量A。

注：TON功能块通过"TMR:TON"实例化。

实例化语法

```
<FB instance name> :<FB variable >;
```

```
TMR(IN := %IX5, PT:= T#300MS, Q=> q1, ET=>et1);
```

```
A:=TMR.Q;
```

9.2.6 RETURN指令

RETURN指令表示当前置条件为TRUE时，离开此POU。

调用语法

```
RETURN;
```

示例

```
IF b=TRUE THEN
```

```
RETURN;
```

```
END_IF;
```

```
a:=a+1;
```

```
// 如果b是TRUE，语句“a:=a+1;”不会被执行，POU会立即被返回。
```

9.2.7 IF指令

通过IF关键字，可以判断执行条件，根据执行条件，执行相应的指令。

语法

IF <布尔表达式1> THEN
<IF_指令
{ELSIF <布尔表达式2> THEN
<ELSIF_指令1>
ELSIF <布尔表达式n> THEN
<ELSIF_指令-1>
ELSE
<ELSE_指令}
END_IF;
{}内部分是可选的

如果 <布尔表达式1> 为TRUE, 那么只有 <IF_指令> 被执行，其它不被执行，否则，从 <布尔表达式2>开始，一个一个计算布尔条件表达式直到其中一个表达式值为TRUE，然后执行此表达式对应的指令，如果没有表达式值为TRUE，那么执行 <ELSE_指令>对应的指令。

示例

```
IF temp<17
THEN heating_on := TRUE;
ELSE heating_on := FALSE;
END_IF;
// 这里，当温度低于17度时加热打开，否则它保持关闭。
```

9.2.8 CASE指令

使用CASE指令，可以根据一个条件变量，根据其对应的多个值罗列处理对应的命令。条件变量只能是整数。

语法

CASE <Var1> OF
<value1>: <Instruction 1>
<value2>: <Instruction 2>
<value3, value4, value5>: <Instruction 3>
<value6 .. value10>: <Instruction4>
...
<value n>: <Instruction n>
ELSE <ELSE Instruction>
END_CASE;

CASE指令根据以下流程处理：

- 如果变量<Var1>的值为 <valueI>, 那么<Instruction I>会被执行。
- 如果 <Var1>没有匹配任何一个值，那么<ELSE Instruction>被执行。
- 如果同一个指令在几个变量值时执行，那么可以把这些值一个接一个的写出来，用逗号隔开，共同执行。
- 如果同一个指令会在一个变量范围内执行，可以写上初始值和结束值，中间用两个点隔开。

示例

```
CASE iVar1 OF
2:c1:=c1+1;
1,6:c1=c1-7;
7..20:c1:=c1+5;
END_CASE
```

9.2.9 FOR循环指令

通过FOR循环，可以编写重复处理逻辑。

语法

FOR <INT_Var> := <INIT_VALUE> TO <END_VALUE> {BY <Step >} DO
<instructions>
END_FOR;
其中：

- {} 内的部分是可选的。
- INT_Var是计数器，是整数类型，只要计数器<INT_Var>不大于<END_VALUE>，<Instructions>会被执行。在执行 <Instructions> 之前首先要检查该条件，如果 <INIT_VALUE>大于 <END_VALUE>，<instructions> 不会被执行。
- 当 <Instructions>执行一次后，<INT_Var>自动增加<Step size>。<Step size>可以是任意整数值，如果不写此参数，默认值为1。当<INT_Var>大于<END_VALUE>时，循环停止。

示例

```
FOR Counter:=1 TO 5 BY 1 DO
Var1:=Var1*2;
END_FOR;
// 假设Var1默认值是2，经过FOR循环后，它的值是32。
```

9.2.10 WHILE循环指令

WHILE循环和FOR循环一样可以作为循环处理使用，但和FOR循环不同是循环条件可以是任意布尔表达式。一旦循环条件满足，循环就执行，否则退出循环。

语法

WHILE <boolean expression> DO
<instructions>
END_WHILE;

执行逻辑

当<Boolean_expression>值为TRUE时，<Instructions>指令开始执行，直到<Boolean_expression>值为FALSE。如果<Boolean_expression>第一次值为FALSE，<Instructions>永不会被执行。如果<Boolean_expression> 永远为TRUE， <Instructions> 重复执行不停止，进入死循环状态，编程时一定确保不要出现死循环。

示例

```
WHILE Counter<>0 DO
Var1:= Var1*2;
Counter := Counter-1;
END_WHILE
```

在一定意义上来说，WHILE循环和REPEAT循环比FOR循环功能更强大，因为不需要在执行循环之前计算循环次数。因此，在有些情况下，用WHILE循环和REPEAT循环两种循环即可。然而，如果清楚地知道循环次数，那么选择FOR循环更佳。

9.2.11 REPEAT循环指令

REPEAT循环不同于WHILE循环，因为循环条件是在循环指令执行后才检查的，这意味着，循环至少执行一次，不管循环条件值如何。

语法

REPEAT
<instructions>
UNTIL <Boolean expression>
END_REPEAT;

执行逻辑

<Instructions>一直执行直到<Boolean expression>值为TRUE。如果<Boolean expression>在第一次值TRUE，那么 <Instructions> 只被执行一遍。如果 <Boolean_expression> 值永远是FALSE，那么<Instructions> 永远执行不停，导致死循环。

示例

```
REPEAT
Var1:=Var1*2;
Counter:=Counter-1;
UNTIL Counter=0;
END_REPEAT;
```

9.2.12 CONTINUE指令

CONTINUE指令在 FOR、WHILE和 REPEAT循环中使用，用于提前结束本轮循环，并重新开始下一轮循环。

示例

```
FOR Counter:=1 TO 5 BY DO
INT1:=INT1/2;
IF INT1=0 THEN
CONTINUE;
END_IF
Var:=Var1/UBT1L
END_FOR;
Erg:=Var1;
```

9.2.13 EXIT指令

EXIT指令用于退出FOR、WHILE、或REPEAT循环。

9.2.14 JMP语句

JMP指令可用于无条件的跳转到指定标签处的代码行。

语法

<label>:
JMP <label>;

执行逻辑

<label>标签名位于程序行的开始处，JMP指令必须有一个跳转目标，也就是预定义的标签。到达JMP指令后，程会跳转到指定的标签处开始执行。

示例

```
aaa :=0;
_label11:aaa:=aaa+1;
(*instructions*)
IF (aaa < 10) THEN
JMP _label1;
END_IF;
```

变量aaa初始为0，只要其小于10，程序就会跳转到label1处重新执行，因此它会影响JMP指令和标签之间的程序的重复执行。

这样的功能也可以通过WHILE 或 REPEAT循环来实现。一般应慎用跳转指令，因为它降低了代码的可读性。

9.2.15 注释

在结构化文本中有两种写注释的方法。

- 单行注释：用“//”开始，用“//”结束。例如：“// This is a comment.”
- 多行注释：用“(*”开始，用“*)”结束。例如：“(*This is a comment.*)”

注释可以在ST编辑器声明或实现部分的任意地方。

注释的嵌套：注释可以放置在其他注释中。

示例

```
(*
a:=inst.out; (*to be checked*)
b:=b+1;
*)
```

10 WEB使用说明

控制器提供WEB管理服务器，您可通过浏览器查看和配置控制器信息。

10.1 登录WEB

推荐使用Chrome浏览器访问WEB管理页面。默认的访问接口ETH1口，IP地址为：https://172.20.1.2，用户名：supcon，密码：supcon。

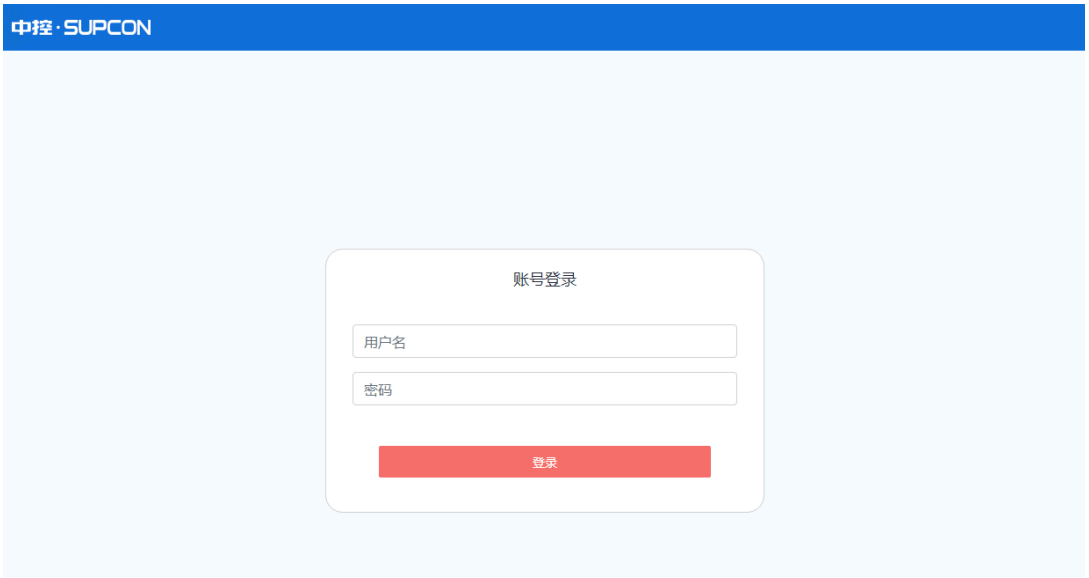


图 **10-1** 登录**WEB**

10.2 运行状态

登录WEB管理页面后，首页为运行状态页面，显示当前控制器系统信息，每三秒自动刷新一次。

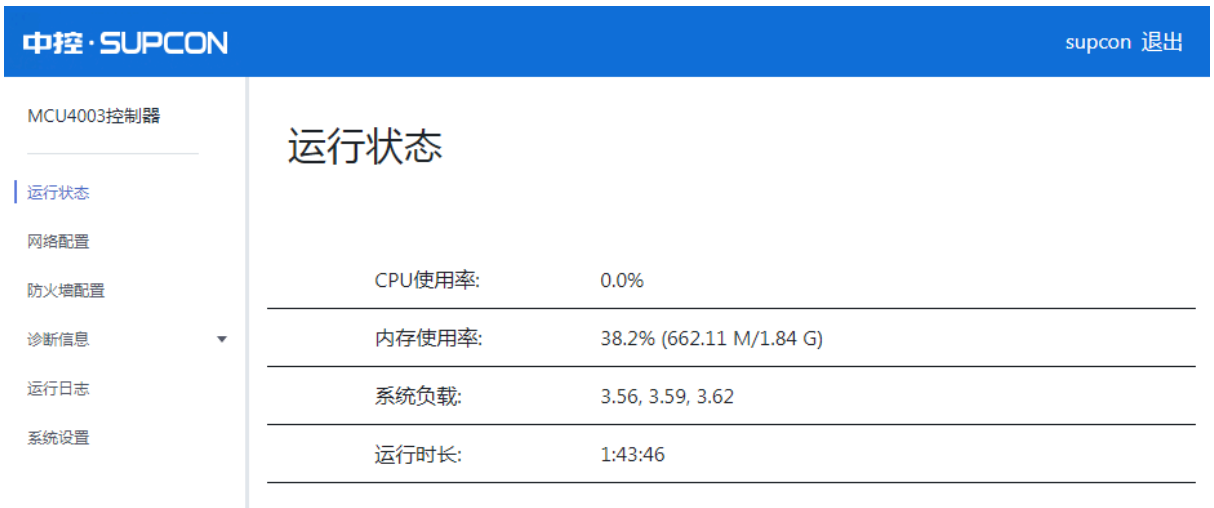


图 **10-2** 运行状态

- CPU使用率：系统所有程序运行占用CPU百分比。
- 内存使用率：系统所有程序内存使用百分比。
- 系统负载：系统CPU，磁盘，网络，内存等综合使用情况的度量值。
- 运行时间：控制器自上电启动之后运行的总时长。

10.3 网络配置

网络配置页中可以设置控制器ETH1~ETH4网口的IP地址及系统默认网关。只有当选择网卡ETH1时才可以配置系统默认网关。

配置方法：如下图所示，在对应参数项中修改参数后，单击“设置”即可。

注意：

修改网络配置后，控制器会自动重启！

图 10-3 网络配置页面

10.4 防火墙配置

防火墙配置可根据需要打开或关闭控制器端口，每个端口对应控制器的某项服务。

图 10-4 防火墙配置页面

端口号对应的服务如下：

- 21端口：FTP服务。
- 22端口：控制器ssh服务。
- 4840/1710/11740端口：控制器运行内核服务，关闭后MotionPro无法扫描及连接控制器。
- ICMP：控制器是否响应ping命令。

10.5 诊断信息

诊断信息页中可查看控制器当前的一些诊断相关信息，用于判断控制器是否存在故障。

目前支持查看控制器ETH1~ETH4网口的连接状态，如下图所示。

图 10-5 硬件诊断页面

10.6 运行日志

控制器运行日志页中可查看和导出控制器运行中的日志信息，便于发生故障时排查问题。

单击“导出”按钮，可将控制器运行日志导出为TXT格式文件。

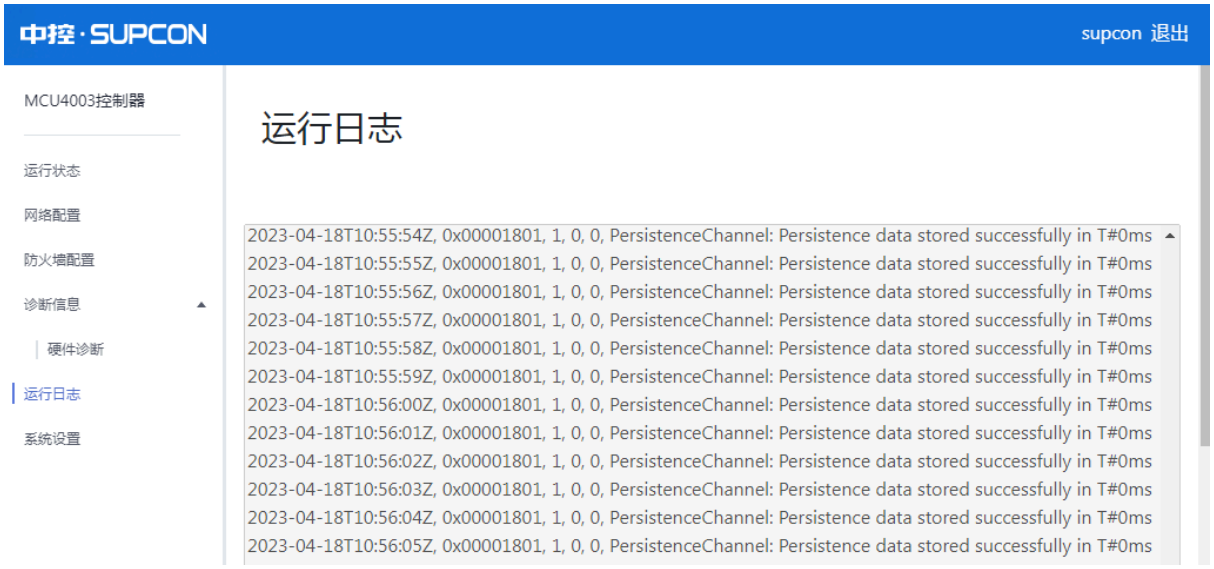


图 **10-6** 控制器日志页面

10.7 系统配置

系统配置页中可进行以下操作：查看固件版本信息、升级固件、设置日志级别、设置系统时间、修改WEB及FTP登录密码、配置NTP服务器地址、重启系统等。

图 **10-7** 系统配置页面

1. 日志级别

   配置控制器记录日志的等级，等级从高到低依次为error、warning、info、debug。当所发生事件的日志等级等于或高于配置的等级时，事件才会被记录到日志中。

2. 系统升级

   当控制器有新固件发布时，可使用该功能进行固件升级。

   注意：当上传固件完成后，控制器会自动重启。

3. 固件版本

   可查看当前控制器的固件版本。

4. 系统时间设置

   可手动填写时间后，点击“确定”下写，注意估计提前量。

   点击“同步本机时间”，会将本地电脑时间直接下写，推荐此操作。

   当配置的时间和系统时间相差太大时，WEB页面需要刷新后重新登录。

5. 修改WEB密码

   可修改当前登录用户的WEB密码，修改后需重新登录。

6. 修改FTP密码

   可修改FTP用户登录密码。

7. NTP服务器地址

   可配置NTP服务器的地址。控制器作为客户端，每隔10分钟主动向服务器同步自身时间。

8. 系统重启

   可手动重启控制器。

11 辅助服务功能

本节介绍控制器的辅助功能，包括支持FTP服务和NTP服务。

11.1 FTP服务

您可通过FTP客户端登录控制器，用于下载SOE日志、组态程序等。

默认FTP用户名：ftpuser，密码：123456。FTP用户权限为只读，即：只能下载，不能上传。

11.2 NTP服务

控制器默认开启NTP服务器，可作为NTP服务器向外提供时间同步服务。服务器地址即为控制器的IP地址，默认为：172.20.1.2。

12 Modbus RTU使用说明

本节介绍控制器作为Modbus RTU主/从站时的配置方法。

12.1 功能概述

控制器具有两路485端口：COM1和COM2，均支持Modbus RTU主站及从站协议。触摸屏等具有 Modbus RTU主站功能的设备，可通过控制器模块上的串行接口，读取控制器I 区、读写Q 区数据。

12.2 参数说明

- COM端口号：默认即可，无需修改。
- 波特率：可选择1200、2400、4800、9600、19200、38400、57600 和 115200。
- 数据位：8 位。
- 停止位：可选择 1 位和 2 位。
- 校验方式：可选择奇校验、偶校验、无校验。

12.3 通信参数配置

在MotionPro软件中，新建工程后，在设备树区域双击COM1或COM2设备，单击选择“通用”页，如下图所示。

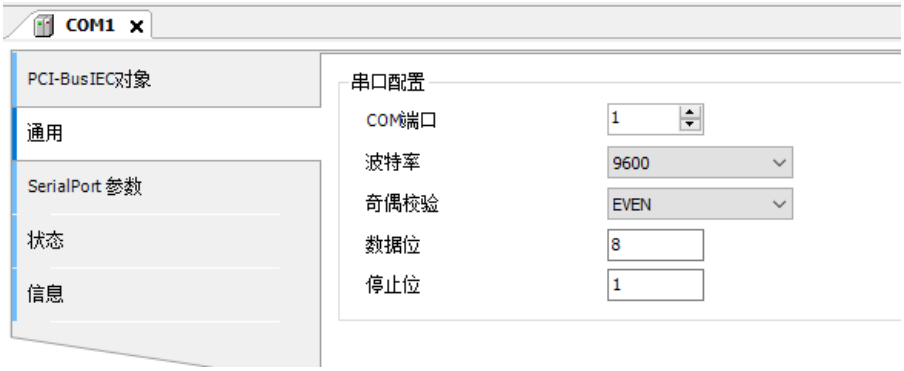


图 **12-1** 通信参数配置

12.4 Modbus RTU主站功能

1. 在MotionPro软件中，右键单击COM1/COM2端口，并在右键菜单中单击“添加设备”命令。

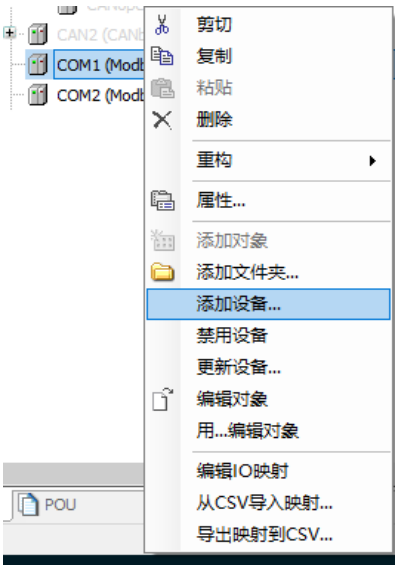


图 12-2 添加设备命令

2. 在弹出窗口中，依次单击展开“现场总线>Modbus>Modbus串行主站”，单击选择“Modbus_Master_COM_Port”后，单击“添加设备”，添加完成。

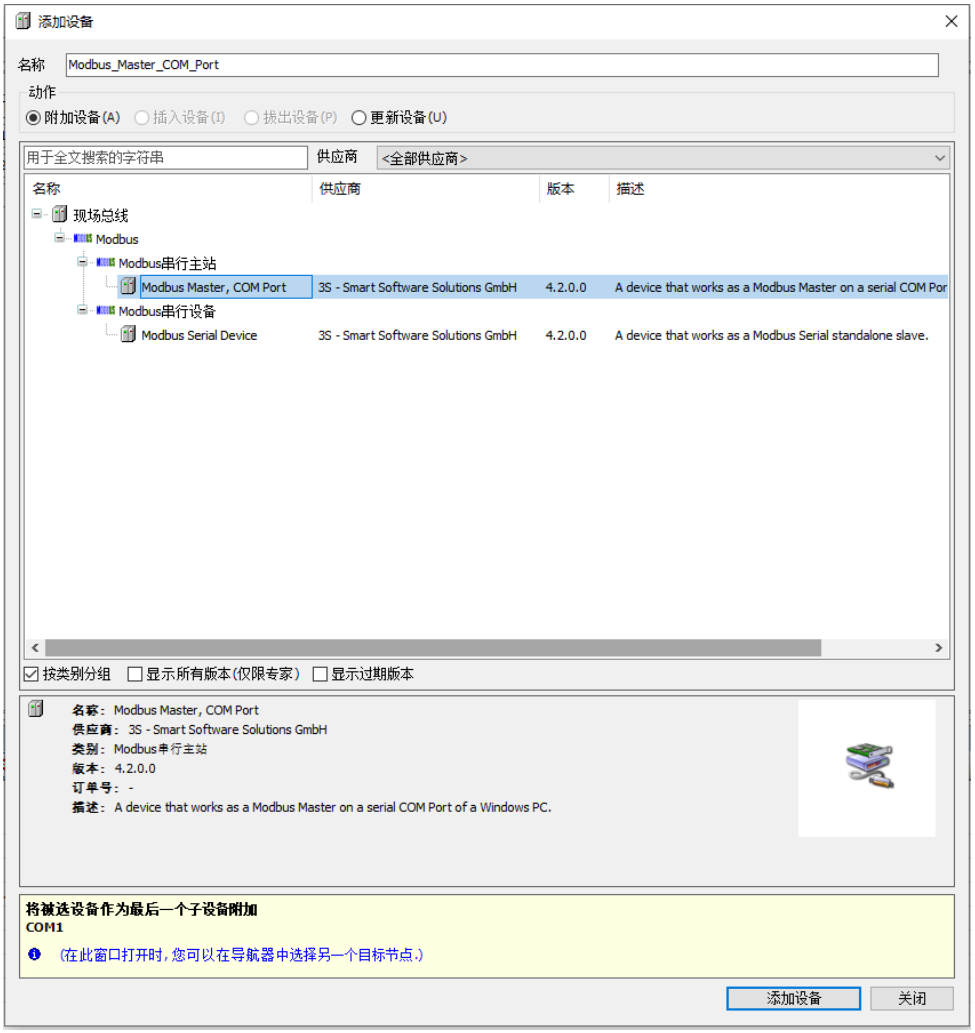


图 12-3 添加设备的操作界面

3. 配置Modbus RTU主站的命令超时时间、帧间隔时间及断线自动重连。

图 12-4 主站参数配置界面

4. 右键点击Modbus_Master_COM_Port，选择添加设备，添加Modbus RTU从站。

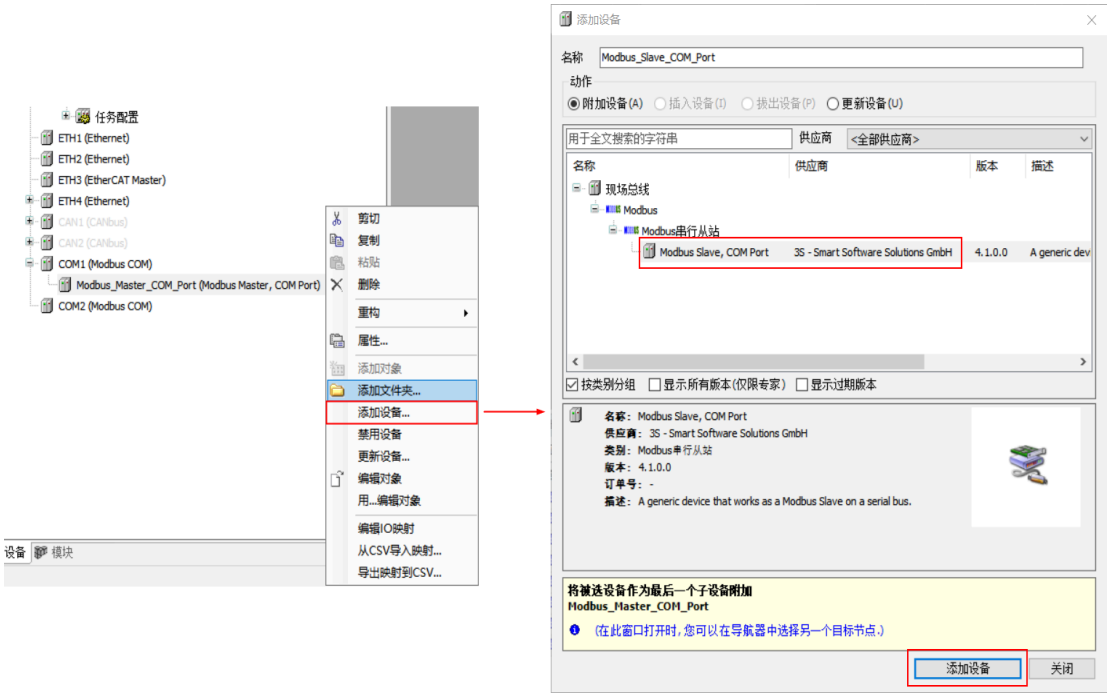


图 12-5 添加Modbus RTU从站

5. 添加从站后，配置从站的地址和响应超时时间。

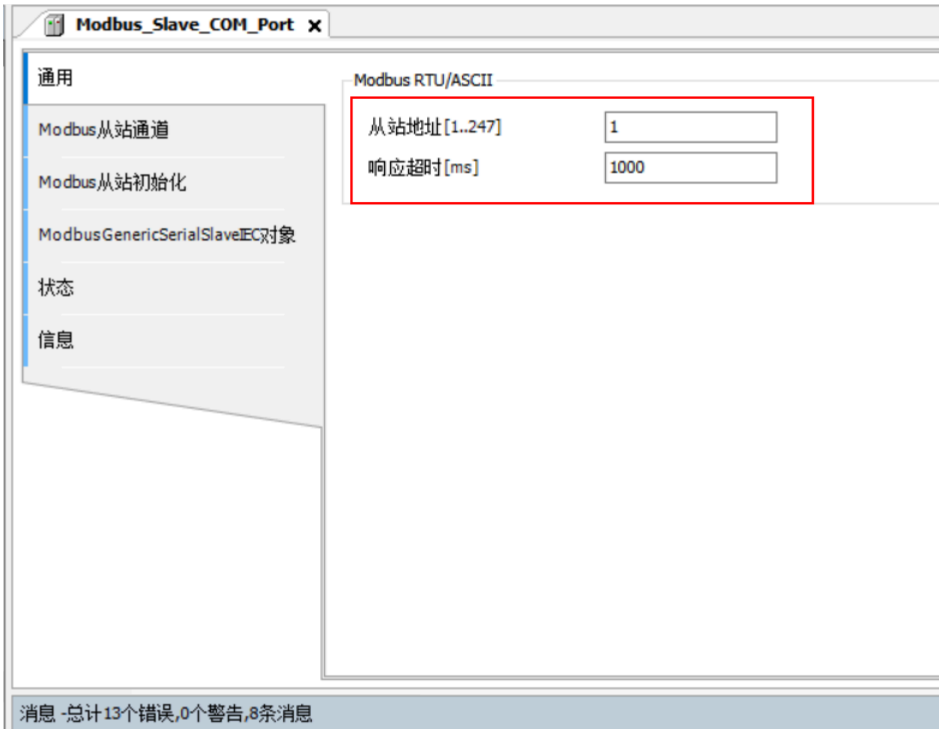


图 12-6 配置从站地址和响应超时时间

6. 点击“添加通道”，添加对从站的Modbus命令。

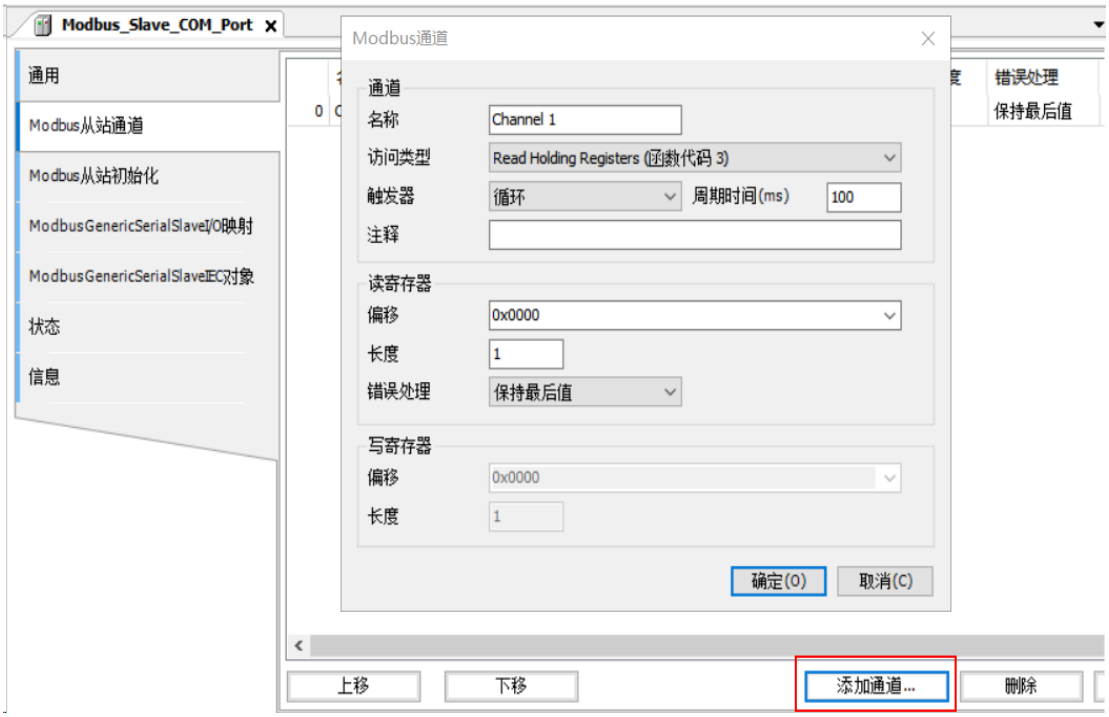


图 **12-7** 添加**Modbus**命令

7. 完成添加后，可以看到已添加命令。

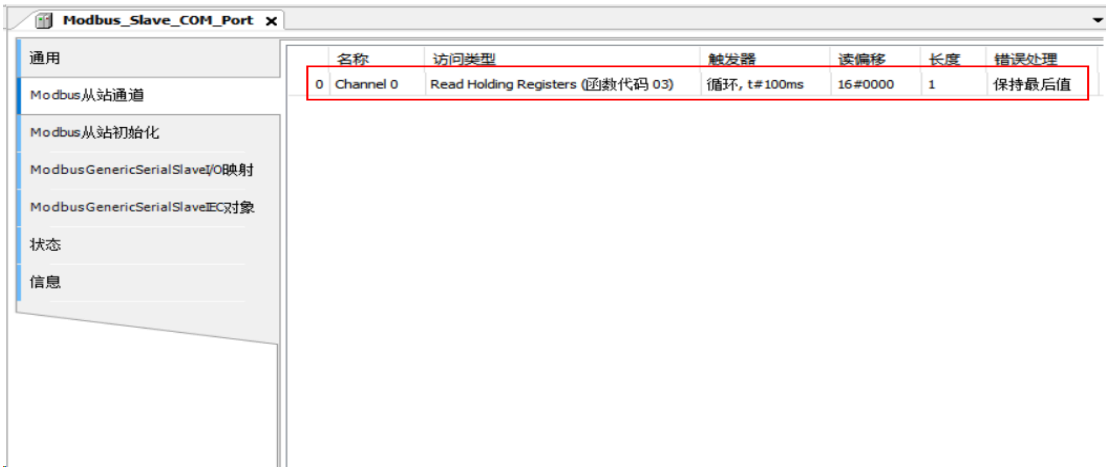


图 **12-8** 查看已添加的**Modbus**命令

8. 如下图所示，Modbus从站位号映射在I/Q数据区，可以直接访问或者通过变量绑定的方式访问。

图 **12-9** 设置位号映射

12.5 Modbus RTU从站功能

本节介绍控制器通过IEC程序实现Modbus RTU从站功能的方法。

12.5.1 通过IEC程序实现

通过IEC程序实现Modbus RTU从站，映射灵活，映射更加灵活，推荐通过本方式实现Modbus RTU从站功能。

1. 定义MBRTU_SLAVE实例及寄存器数组。

```
VAR_GLOBAL
MBRTU_SLAVE_0: MBRTU_SLAVE;
aDiscreteInputsMemory: ARRAY[0..65535] OF BOOL;
aCoilsMemory: ARRAY[0..65535] OF BOOL;
aInputRegistersMemory: ARRAY[0..65535] OF UINT;
aHoldingRegistersMemory: ARRAY[0..65535] OF UINT;
END_VAR
```

2. 调用MBRTU_SLAVE功能块实例：

```
GVL_1.MBRTU_SLAVE_0(
iPort:= SYS_COMPORT1,
dwBaudRate:= SysCom.SYS_BR_115200,
byDataBits:= 8,
eParity:=SysCom.SYS_NOPARITY ,
```

```
eStopBits:= SysCom.SYS_ONESTOPBIT,
eRtuAscii:= ModbusFB.RtuAscii.RTU,
uiUnitId:= 1,
aDiscreteInputsMemory:= ADR(GVL_1.aDiscreteInputsMemory),
aDiscreteInputsMemoryNum:= 65535,
aCoilsMemory:= ADR(GVL_1.aCoilsMemory),
aCoilsMemoryNum:= 65535,
aInputRegistersMemory:= ADR(GVL_1.aInputRegistersMemory),
aInputRegistersMemoryNum:= 65535,
aHoldingRegistersMemory:= ADR(GVL_1.aHoldingRegistersMemory),
aHoldingRegistersMemoryNum:= 65535);
```

1) 其中iPort参数表示要接入的串口序号，控制器支持2路串口，分别是SYS_COMPORT1，SYS_COMPORT2。
2) dwBaudRate表示波特率，支持：
   a. SysCom.SYS_BR_115200，波特率115200
   b. SysCom.SYS_BR_57600，波特率57600
   c. SysCom.SYS_BR_38400，波特率38400
   d. SysCom.SYS_BR_19200，波特率19200
   e. SysCom.SYS_BR_9600，波特率9600
   f. SysCom.SYS_BR_4800。波特率4800
3) eParity表示校验位，支持：
   a. SysCom.SYS_NOPARITY，无校验
   b. SysCom.SYS_ODDARITY，奇校验
   c. SysCom.SYS_EVENPARITY，偶校验
4) eStopBits表示停止位，支持：
   a. SysCom.SYS_ONESTOPBIT，1位停止位
   b. SysCom.SYS_ONE5STOPBITS，1.5位停止位
   c. SysCom.SYS_TWOSTOPBITS，2位停止位
5) eRtuAscii表示rtu或ascii模式，支持：
   a. ModbusFB.RtuAscii.RTU，rtu模式
   b. ModbusFB.RtuAscii.ASCII，ascii模式
6) uiUnitId表示本从站ID。
7) 注意传入的MemoryNum不要超过实际数组长度。

3. MBRTU_SERVER的线圈、寄存器的地址，与所定义的aDiscreteInputsMemory等数组的元素下表一一对应，如更新10号地址线圈的值：

```
GVL.aCoilsMemory[10] := TRUE;
```

4. 从站的响应性能和串口波特率，和其所在的任务周期有关，缩短任务周期可以提升响应速度，但为避免过多消耗CPU资源，建议任务周期不低于20ms。

13 Modbus TCP使用说明

本节介绍控制器作为Modbus TCP主/从站时的配置方法。

13.1 功能概述

控制器的ETH1和ETH2网口均支持Modbus TCP主站及从站协议。触摸屏等具有Modbus TCP主站功能的设备，可通过控制器模块上的网口，读取控制器I 区、读写Q 区数据。

13.2 Modbus-TCP主站功能

1. 在MotionPro软件中，右键单击ETH1/ETH2端口，并在右键菜单单击“添加设备”命令。

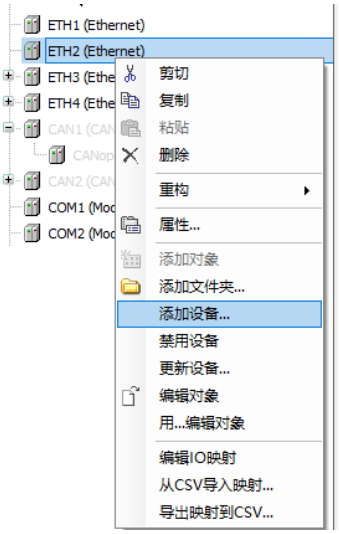


图 **13-1** 添加设备命令

2. 在弹出窗口中，依次单击展开“现场总线 > Modbus > ModbusTCP主站”，单击选择“Modbus_TCP_Master”，单击“添加设备”，添加完成。

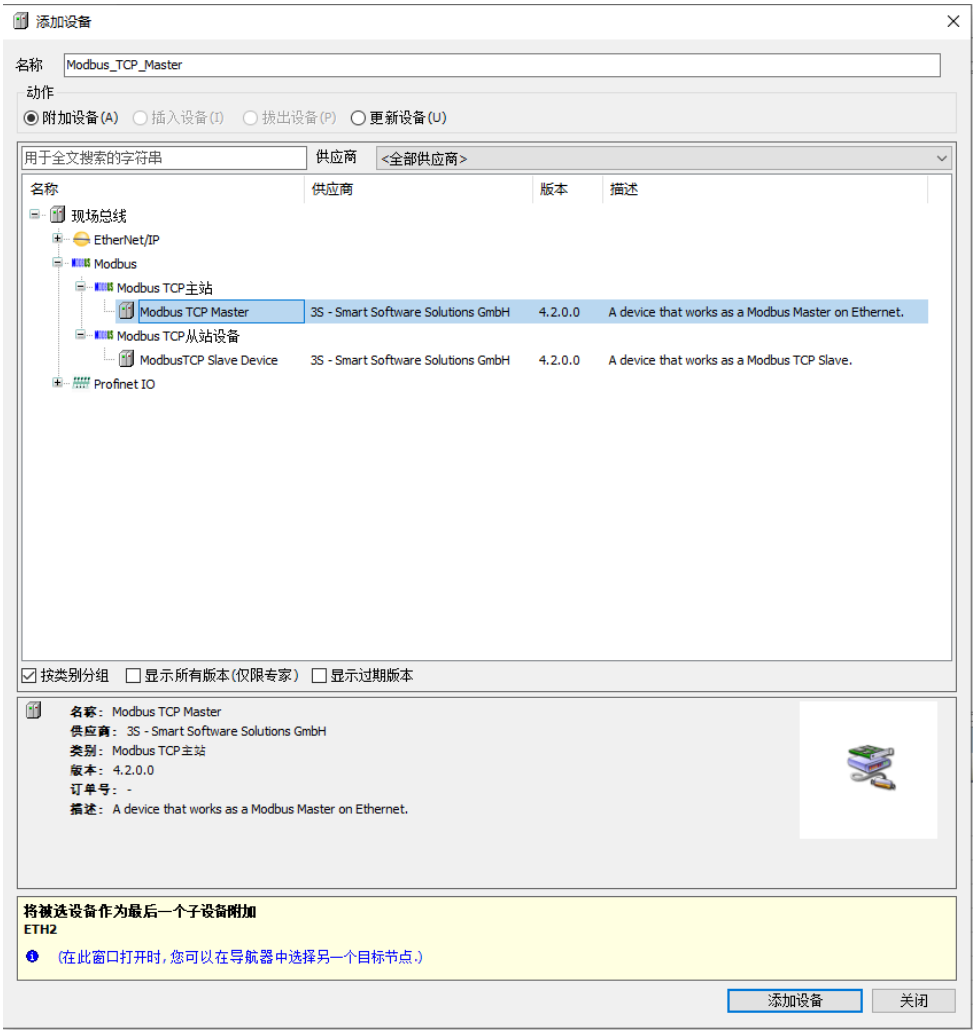


图 13-2 添加设备的操作界面

3. 配置Modbus TCP主站的命令响应时间，Socket超时及断线自动重连。

图 13-3 主站参数配置界面

4. 添加ModbusTCP从站。

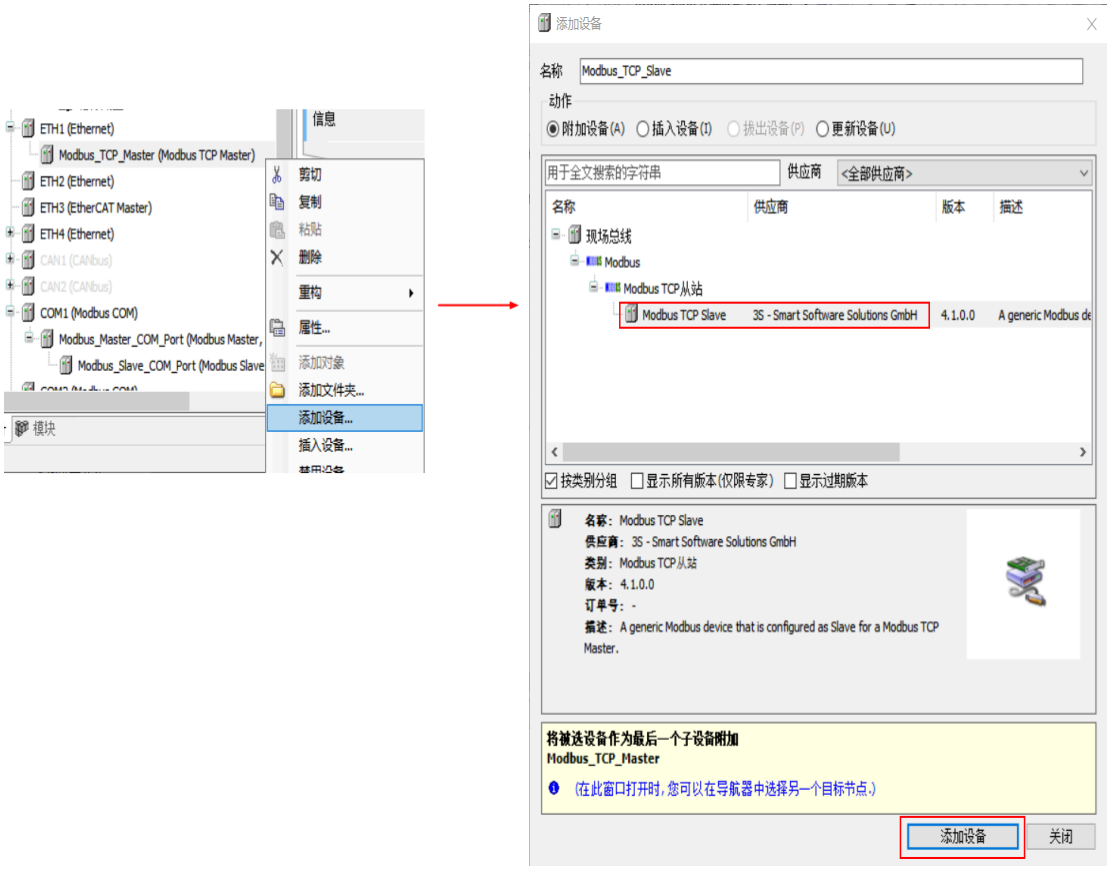


图 **13-4** 添加**Modbus TCP**从站

5. 配置从站的地址，端口和响应超时时间。

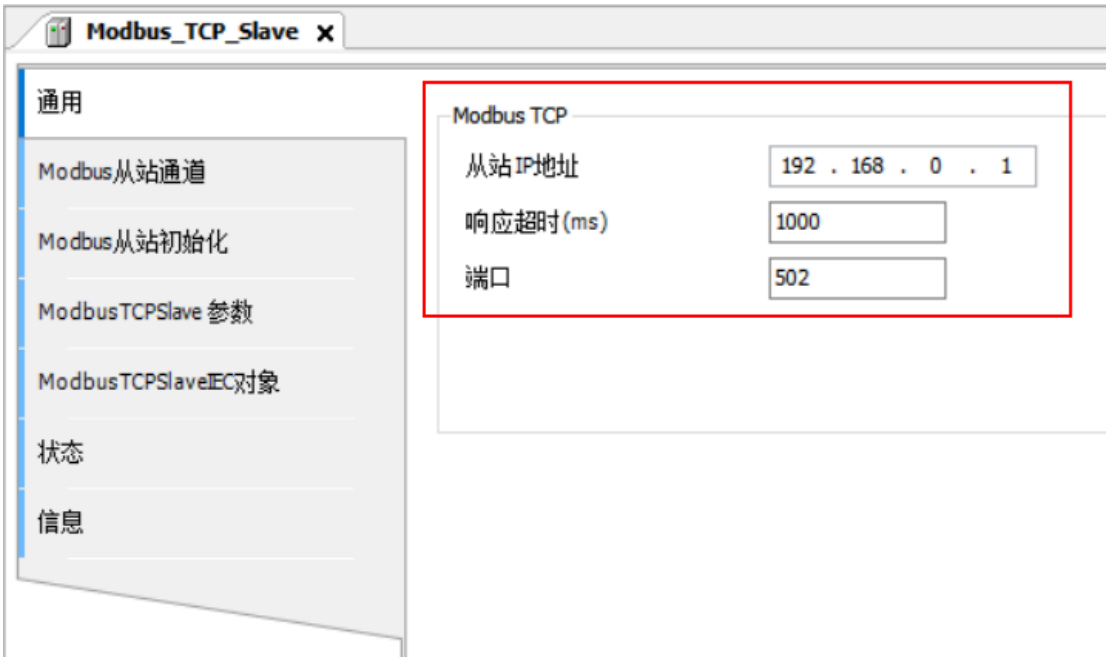


图 **13-5** 配置从站地址和响应超时时间

6. 点击“添加通道”，添加对从站的Modbus命令。

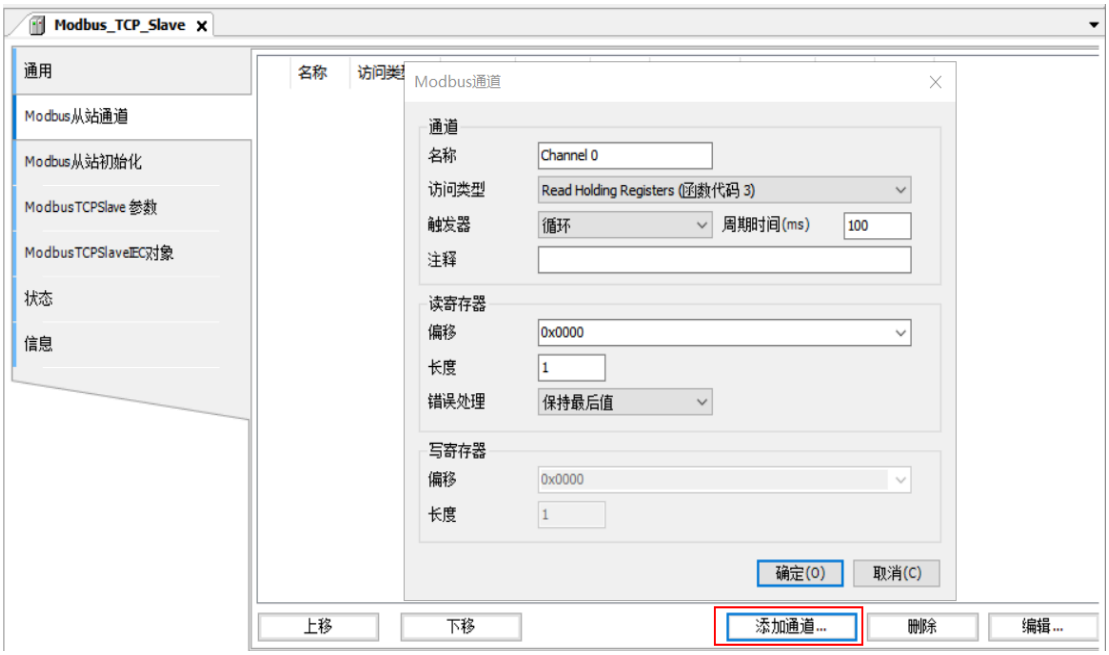


图 **13-6** 添加**Modbus**命令

7. 完成添加后，可以看到已添加命令。

图 **13-7** 查看已添加命令

8. Modbus从站位号映射在I/Q数据区，可以直接访问或者通过变量绑定的方式访问。

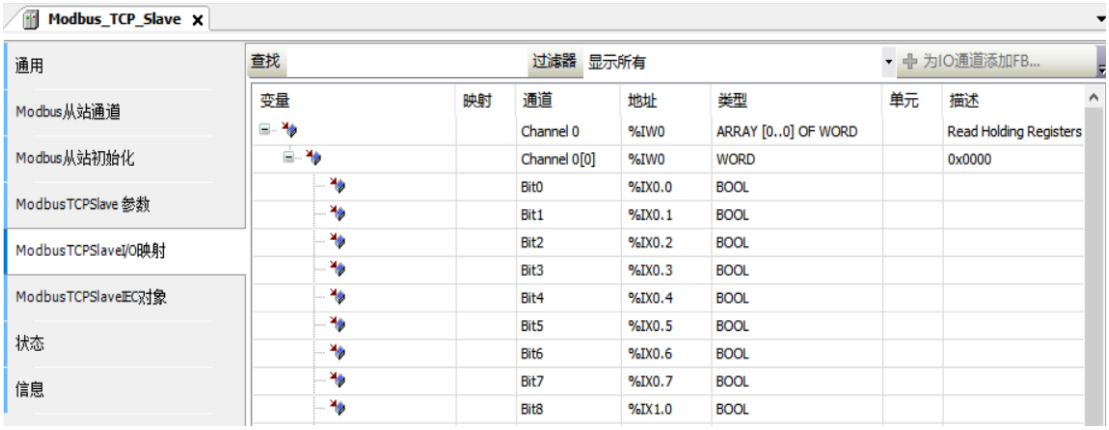


图 **13-8** 设置位号映射

13.3 Modbus-TCP从站功能

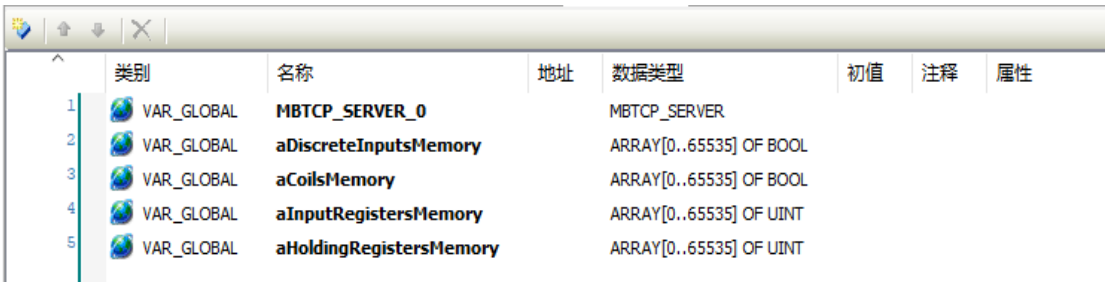
本节介绍控制器通过IEC程序实现Modbus TCP从站功能的方法。

13.3.1 通过IEC程序实现

通过IEC程序实现Modbus TCP从站，映射更加灵活，推荐通过本方式实现Modbus TCP从站功能。

1. 定义MBTCP_SERVER实例及寄存器数组。

```
VAR_GLOBAL
MBTCP_SERVER_0: MBTCP_SERVER;
aDiscreteInputsMemory: ARRAY[0..65535] OF BOOL;
aCoilsMemory: ARRAY[0..65535] OF BOOL;
aInputRegistersMemory: ARRAY[0..65535] OF UINT;
aHoldingRegistersMemory: ARRAY[0..65535] OF UINT;
END_VAR
```



	类别	名称	地址	数据类型	初值	注释	属性
1	VAR_GLOBAL	MBTCP_SERVER_0		MBTCP_SERVER			
2	VAR_GLOBAL	aDiscreteInputsMemory		ARRAY[0..65535] OF BOOL			
3	VAR_GLOBAL	aCoilsMemory		ARRAY[0..65535] OF BOOL			
4	VAR_GLOBAL	aInputRegistersMemory		ARRAY[0..65535] OF UINT			
5	VAR_GLOBAL	aHoldingRegistersMemory		ARRAY[0..65535] OF UINT			

图 13-9

2. 调用MBTCP_SERVER功能块实例：

```
GVL.MBTCP_SERVER_0(
iface:= "ETH1",
port:= 502,
aDiscreteInputsMemory:= ADR(GVL.aDiscreteInputsMemory),
aDiscreteInputsMemoryNum:= 65535,
aCoilsMemory:= ADR(GVL.aCoilsMemory),
aCoilsMemoryNum:= 65535,
aInputRegistersMemory:= ADR(GVL.aInputRegistersMemory),
aInputRegistersMemoryNum:= 65535,
aHoldingRegistersMemory:= ADR(GVL.aHoldingRegistersMemory),
```

```
aHoldingRegistersMemoryNum:= 65535);
```

1) 其中iface参数表示要接入的网口名，控制器支持“ETH1”，“ETH2”。
2) port表示监听的modbusTCP网络端口。
3) 注意传入的MemoryNum不要超过实际数组长度。

3. MBTCP_SERVER的线圈、寄存器的地址，与所定义的aDiscreteInputsMemory等数组的元素下表一一对应，如更新10号地址线圈的值：

```
GVL.aCoilsMemory[10] := TRUE;
```

4. 从站的响应性能与客户端数量，和其所在的任务周期C有关。假设所有客户端共有命令数量N，所有客户端希望在时间T内可以完成所有命令，则T、N、C应该满足以下算术关系：

C = T / N / 2 = 2T / N

且避免过多消耗CPU资源，建议C不小于20ms。

14 Profinet使用说明

节介绍控制器作为Profinet主站时的参数规格，以及组态步骤。

14.1 功能概述

控制器的ETH4口支持Profinet主站功能，可扩展中控远程I/O机架，或第三方Profinet从站。

14.2 规格说明

- 工作模式为Profinet控制器（Controller）。
- 符合2.42版本Profinet规范
- 一致性等级Class B
- 最大支持128从站
- 单从站最大输入数据1440字节，最大输出数据1440字节

14.3 添加从站设备描述文件

在MotionPro软件中，单击“工具 > 设备存储库”打开“设备存储库”界面。单击“安装”，选中配置文件（窗口右下角选择“Profinet IO配置文件(GSDML*.xml)”），单击“打开”，等待界面提示“设备已安装到设备存储库”，如下图所示。

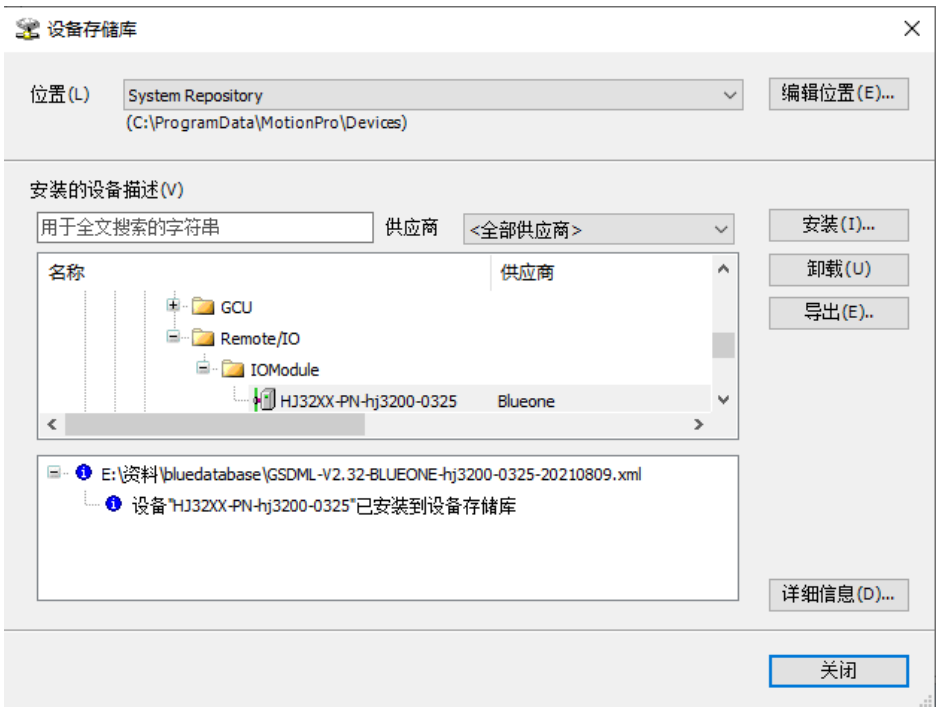


图 **14-1** 添加**Profinet**从站设备描述文件

14.4 添加从站模块

本节说明手动添加和扫描添加从站模块的方法。

14.4.1 手动添加

1. 在MotionPro软件中，新建工程后，在设备树区域，右键单击“PN_Controller”，并单击选择“添加设备”命令，打开如下图所示界面。

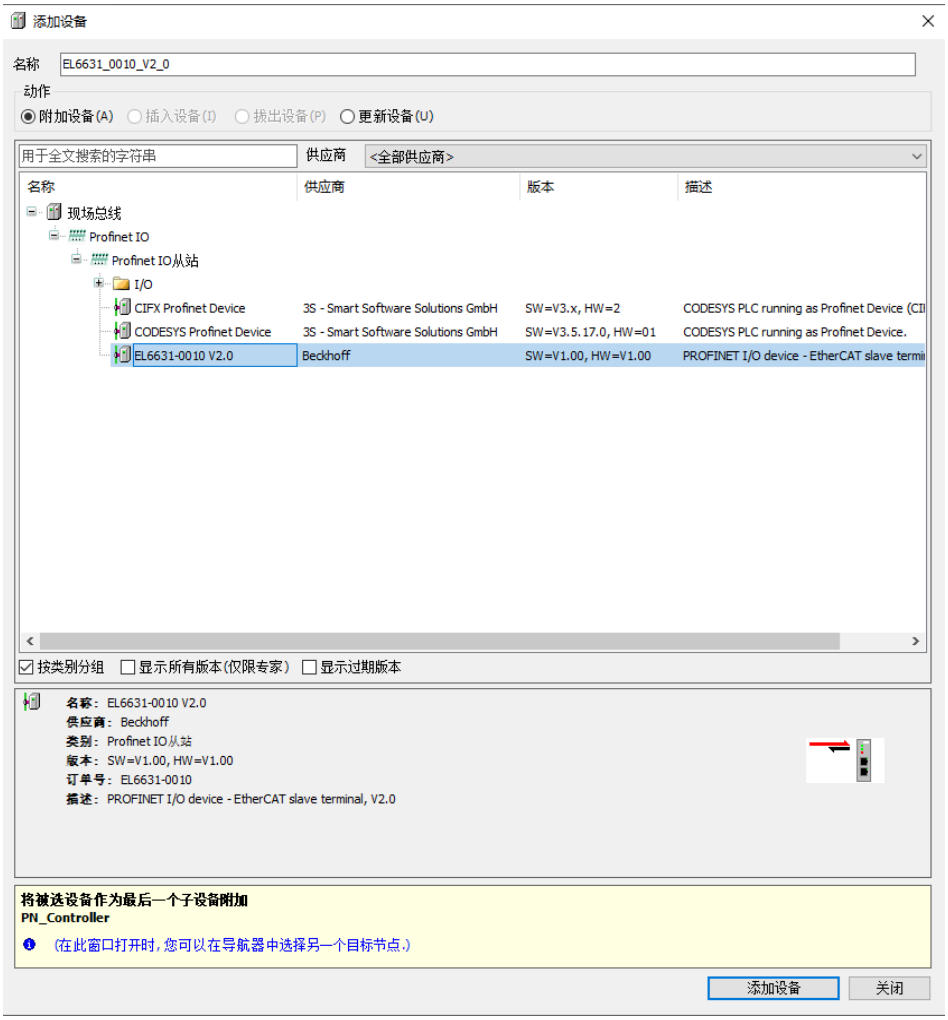


图 14-2 添加Profinet从站

2. 在上图所示界面中，选择对应的Profinet从站，单击“添加设备”即可。

14.4.2 扫描添加

如果从站设备已经连接上控制器，并且下载过一次组态，则可以采用自动扫描的方式添加从站设备。操作方法如下：

1. 在MotionPro软件中，右键单击“PN_Controller”节点，并在右键菜单中单击“扫描设备”命令。

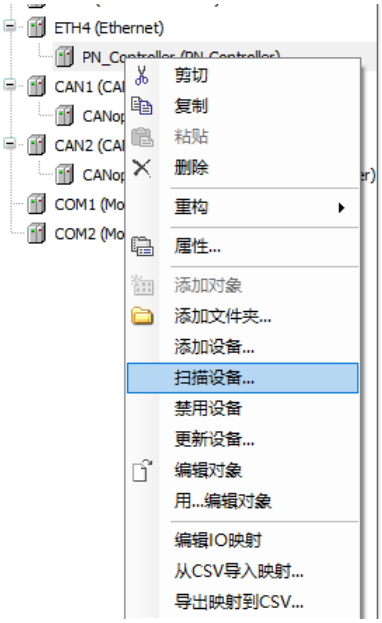


图 **14-3** 扫描设备命令

2. 单击选中扫描到的从站设备，单击“复制到工程”即可，如下图所示。

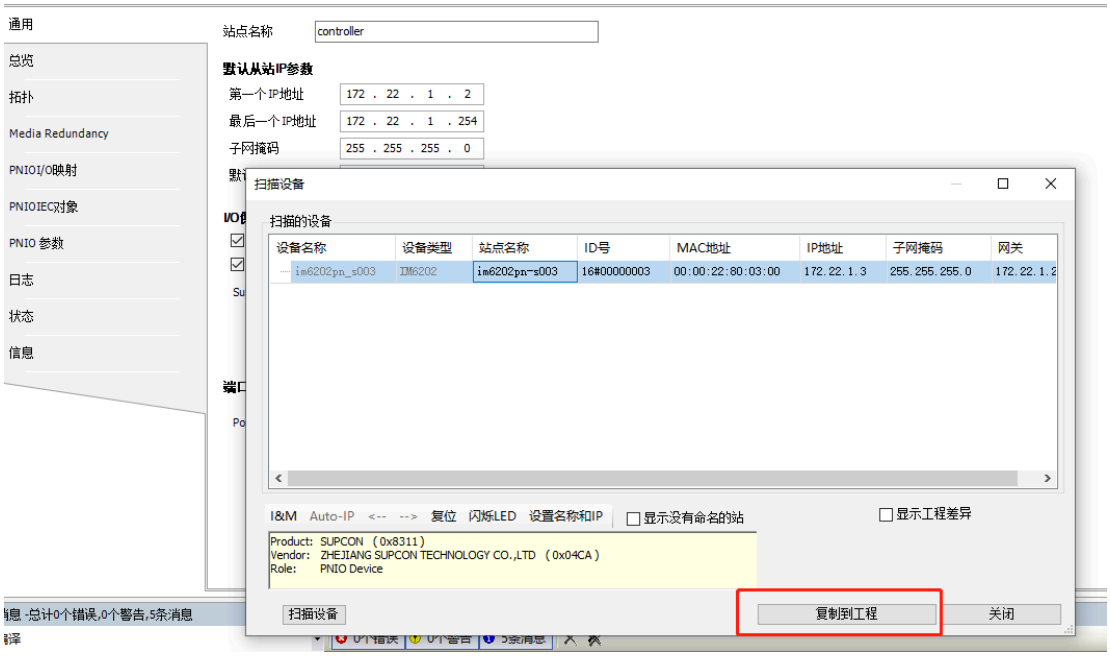


图 **14-4** 复制到工程

15 CANopen使用说明

本节介绍控制器作为CANopen主站时的通信配置说明。

15.1 功能概述

控制器有2路CAN接口，支持CANopen主站协议。

15.2 规格说明

1. 网络号：1~127可配置。
2. 每路支持16个CANopen从站。
3. 波特率支持10，20，50，100，125，250，500，800，1000kbit/s。
4. 状态机：符合CiA 306标准，支持NMT服务。
5. CAN ID支持11位长度。
6. 守护模式支持节点和心跳保护。
7. 支持SDO配置。
8. 同步模式支持生产者模式。
9. PDO支持512个发送和512个接收命令。
10. PDO变量可配置映射。
11. PDO传输类型。
    - PDO发送(从->主)：非循环/循环/仅RTR同步，特定制造商/特定设备配置文件/仅RTR异步。
    - PDO接收(主->从)：非循环/循环同步，特定制造商/特定设备配置文件异步。
    - PDO支持事件时间和抑制时间可配。

15.3 配置过程

本节详细介绍控制器CANopen主站的详细配置过程。

15.3.1 通信参数配置

在MotionPro软件中，新建工程后，在设备树区域，双击CAN1或CAN2节点，选择“通用”页，即可配置CAN端口参数，如下图所示。注意：CAN1的网络号对应为0，CAN2的网络号对应为1，且不可修改。

图 **15-1 CAN**端口参数配置界面

15.3.2 CANopen主站参数配置

在MotionPro软件中，双击“CANopen_Manager”，打开CANopen主站参数配置页面，如下图所示，选择“通用”页，配置主站参数。

图 **15-2** 主站参数配置页面

15.3.3 添加从站设备描述文件

在MotionPro软件中，单击“工具 > 设备存储库”打开“设备存储库”界面。单击“安装”，选中配置文件（窗口右下角选择“所有支持的描述文件”），单击“打开”，等待界面提示“设备已安装到设备存储库”，如下图所示。

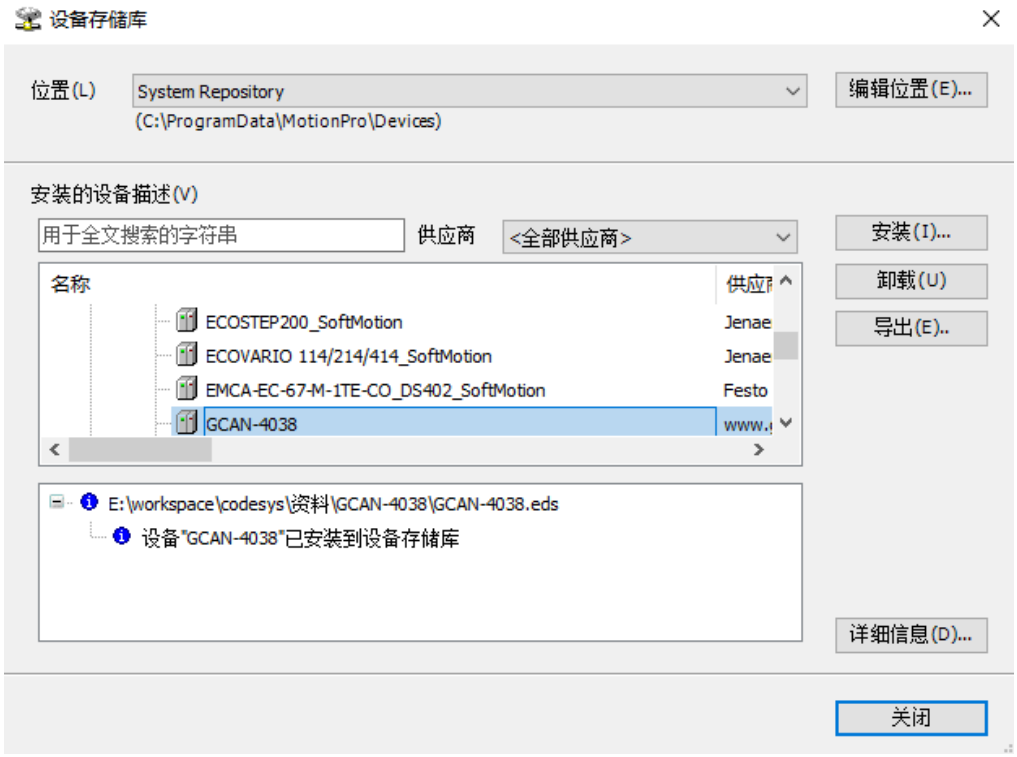


图 **15-3** 添加**CANopen**从站设备描述文件

15.3.4 添加CANopen从站模块

本节介绍添加CANopen从站模块的两种方法。

手动添加

在MotionPro软件中，新建工程后，在设备树区域，右键单击“CANopen_Manager”，并在右键菜单中单击“添加设备”命令，打开如下配置界面。

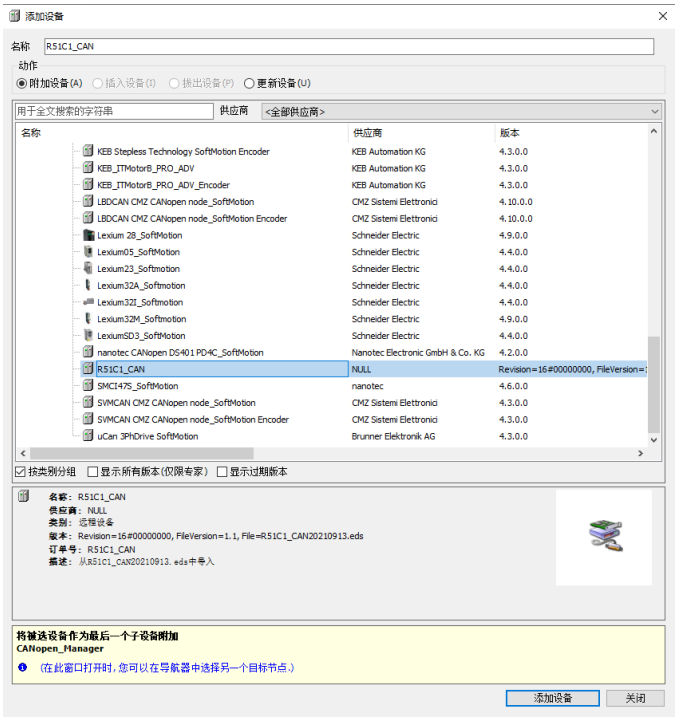


图 15-4 手动添加CANopen从设备

选择要添加的从站模块，单击“添加设备”即可。

扫描添加

如果从站设备已经连接了控制器，并且下载过一次组态，则可以采用自动扫描的方式添加从站模块。操作方式如下：

1. 在MotionPro软件中，右键单击“CANopen_Manager”节点，并在右键菜单中单击“扫描设备”命令。
2. 选择扫描到的从站设备，单击“复制到工程”即可。

15.4 CANopen配置项一览

本节说明MotionPro中CANopen相关配置项的配置方法。

15.4.1 CANopenManager

CANopen Manager作为CANopen通信协议组态配置的一部分，本节说明其在Motionpro软件中的配置方法。

通用配置页面

图 15-5 通用配置

配置区	参数	说明
通用	节点ID	CANopen Manager主站节点ID，取值范围1~127，不可与其他节点重复
	自动启动CANopen Manager	勾选：CANopen Manager自动启动，在所有从站就绪后，自动切换到OP状态。 不勾选：通过用户程序调用功能块CiA405.NMT启动。
	可选从站的轮询	勾选：在启动阶段，当从站不响应时，CANopen Manager主站将自动重复轮询从站，直至其响应。
	启动从站	勾选：CANopen Manager主站自动启动从站。 不勾选：通过用户程序调用功能块CiA405.NMT启动
	NMT启动所有（如果可能）	当“启动从站”勾选时，CANopen Manager主站使用用“NMT Start ALL（全网启动）”命令来启动从站。
	NMT错误行为	Restart Slave：当出现NMT错误时，自动复位从站，重新执行NMT Reset+SDO配置+NMT Start流程。 Stop Slave：当出现NMT错误时，停止从站，通过用户程序调用功能块CiA405.NMT恢复。
保护	使能心跳产生	勾选：主站主动发出心跳。
	节点ID	心跳包携带的唯一ID
	Producer time (ms)	产生心跳的周期时间
同步	使能同步产生	勾选：CANopen Manager发出SYNC同步帧，同步PDO对象会紧跟SYNC同步帧发送

配置区	参数	说明
	COB-ID(Hex)	SYNC同步帧的COB-ID，范围1~2047
	循环周期（us）	SYNC同步帧发送周期
	窗口长度（us）	同步PDO对象发出的窗口时间（距离SYNC同步帧发出的时间差）
	使能同步消耗	勾选：同步帧消费者，即接收总线上其他节点发出的同步帧，一般不用此模式
时间	使能TIME生产	勾选：CANopen Manager发送TIME（时间戳）信息
	COB-ID(Hex)	时间戳的COB-ID，范围1~2047
	Producer time (ms)	发送周期，范围0~65535

15.4.2 CANopen从站

本节说明CANopen从站通用配置页面和PDO配置页面参数的配置方法。

通用配置页面

图 15-6 通用配置页面

通用配置页面中，可按下表配置。

配置区	参数	说明
通用	节点ID	从站节点ID，取值范围1~127，不可与其他节点重复

配置区	参数	说明
	专家设置	显示所有配置
	可选设备	勾选：从站可选，主站不再依据此从站的状态启动网络
	使能同步生产	勾选：从站发出SYNC同步帧，与主站“使能同步产生”选项互斥，即总线上只能有1个节点产生同步帧
	未初始化	勾选：不对此从站进行初始化
保护	使能节点保护	勾选：主站通过对此从站周期（保护时间ms）发送守护帧的方式，检测从站是否在线，如果从站不应答，则进入重发，重发超过“生命因子”约定的次数后从站仍不应答，则视从站掉线
	保护时间（ms）	
	生命因子	
	使能心跳产生	勾选：从站主动发出心跳。
	生产者时间	从站产生心跳的周期时间
紧急情况	使能紧急情况（EMCY）	勾选：当从站出现内部错误时，发出紧急信息
	COB-ID	紧急信息的COB-ID，范围1~2047
时间	使能TIME生产	勾选：从站发送TIME（时间戳）信息
	COB-ID(Hex)	时间戳的COB-ID，范围1~2047
	使能TIME消费	发送周期，范围0~65535
启动时检查	检查供应商ID	主站启动时检查从站检查供应商ID，并与EDS文件对比，不一致则不启动该从站
	检查产品号	主站启动时检查从站检查产品号，并与EDS文件对比，不一致则不启动该从站
	检查修订号	主站启动时检查从站检查修订号，并与EDS文件对比，不一致则不启动该从站

PDO配置页面

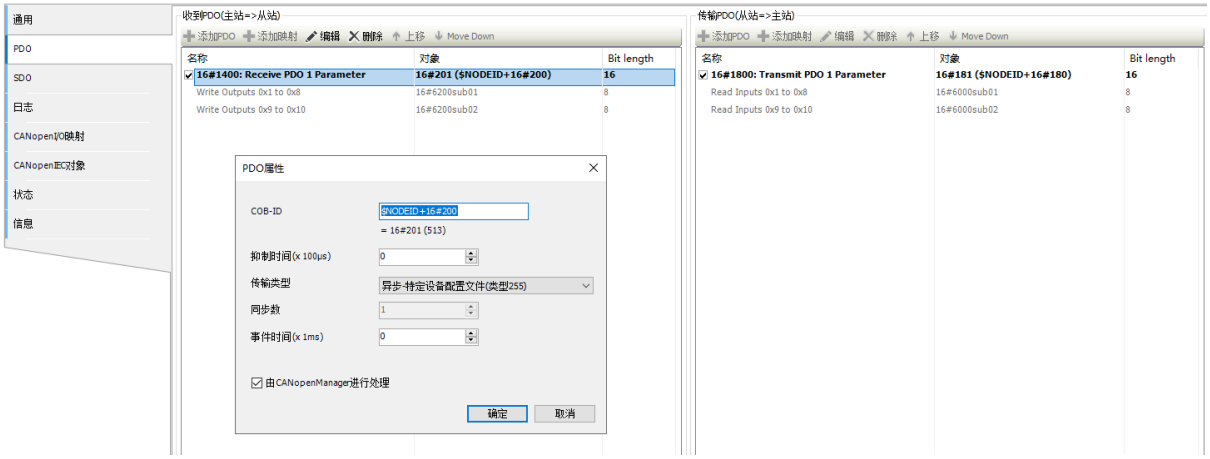


图 15-7 PDO配置页面

PDO页面中，可按下表配置。

配置区	参数	说明
收到PDO（主站=>从站） 传输PDO（从站=>主站）	添加PDO	进入PDO选择对话框，可选择添加TxPDO，RxPDO
	添加映射	编辑TxPDO，RxPDO的映射
	编辑	进入TxPDO，RxPDO的编辑界面
PDO属性	COB-ID	PDO的COB-ID，一般不需要修改
	抑制时间（x100us）	从站保持最短的静默（不发数据）时间，默认0，范围0~65535
	传输类型	从站传输类型： 非循环-同步（类型0）：变化时立即进行一次PDO传输。 循环-同步（类型1~240）：接收“同步数”个同步帧后，进行一次PDO传输。 异步-特定制造商（类型254）：厂商定义的特殊事件产生后进行一次PDO传输。 异步-特定设备配置文件（类型255）：按CiA规定的设备规范进行传输
	同步数	传输类型为“循环-同步（类型1~240）”时，接收同步帧的个数。
	事件时间（x 1ms）	传输类型为“异步-特定制造商（类型254）”时或“异步-特定设备配置文件（类型255）”时使用，起到周期传输产生PDO传输的作用。
	由CANopenManager进行处理	勾选：CANopenManager自动处理PDO传输。 不勾选：用户程序处理PDO传输。

16 OPC UA服务器使用说明

本节说明控制器的OPC UA功能、规格、及其配置方法。

16.1 功能概述

- 支持客户端浏览变量类型
- 支持变量变化通知
- 支持读，写，订阅服务
- 支持事件
- 支持结构体数据类型
- 支持签名&加密通信，支持Basic256Sha256，Aes128_Sha256_RsaOaep，Aes256_Sha256_RsaPss
- 支持匿名登录，用户名密码登录

16.2 规格说明

- 最大变量数量5000个（基本类型）
- 最大会话数100个
- 数据刷新率100ms
- 端口号4841

16.3 OPC UA与IEC变量类型对照表

OPC UA类型	IEC类型	说明
Boolean	BOOL	-
Byte	BYTE	
SByte	SINT	
Int16	INT	
UInt16	UINT	
INT32	DINT	
UInt32	UDINT	
Int64	LINT	

OPC UA类型	IEC类型	说明
UInt64	ULINT	
FLOAT	REAL	
Double	LREAL	
Duration	LREAL	
DateTime	LDATEANDTIME	
UtcTime	LTIME	
String	STRING	

16.4 配置方法

1. 编译OPC UA配置。
   双击打开“符号配置”页面，点击“编译”按钮，编译通过后，位号列表会刷新。

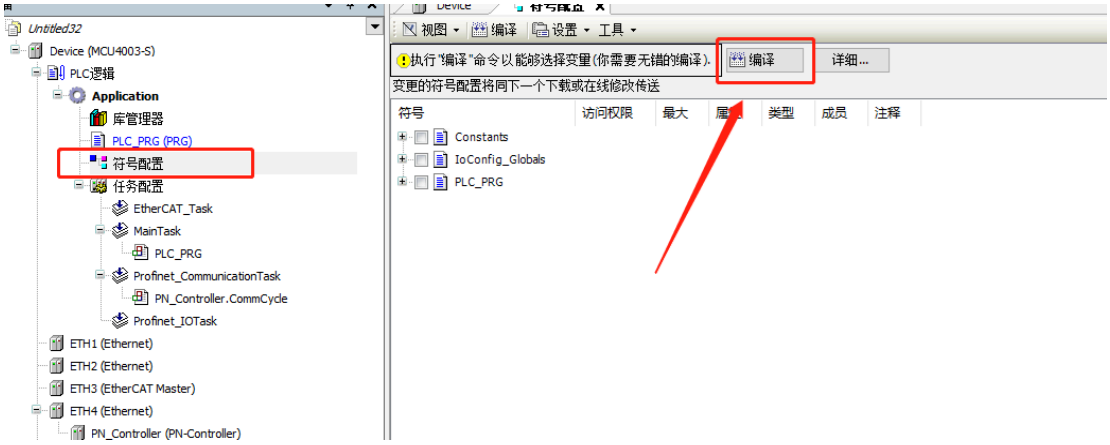


图 16-1 编译符号配置界面

2. 勾选需要开放给OPC UA客户端访问的及访问权限。

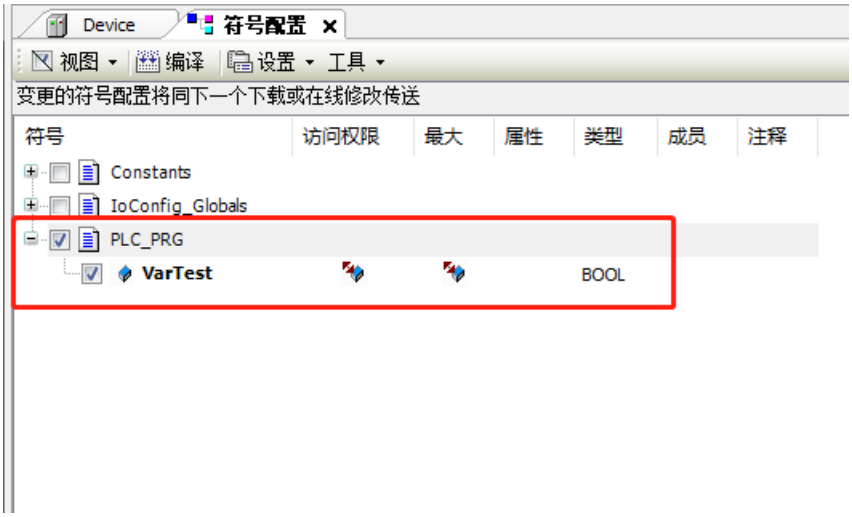


图 **16-2** 设置访问权限

- 访问权限：当前变量能够被客户端访问的权限。
- 最大：当前变量能够被配置的最大权限。

图标	功能
	可读可写
	只读
	只写

3. 依次编译OPC UA配置、编译工程，下载组态到控制器。

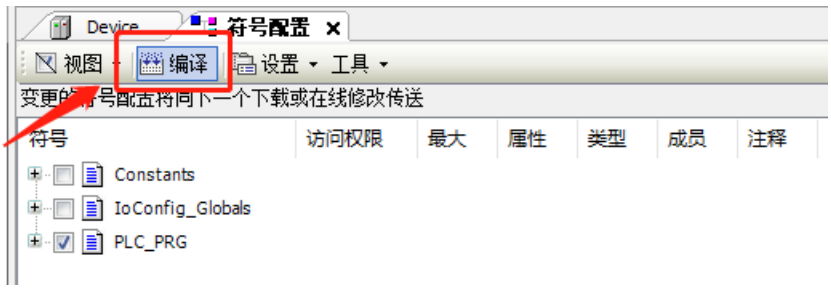


图 **16-3** 编译**OPC UA**配置

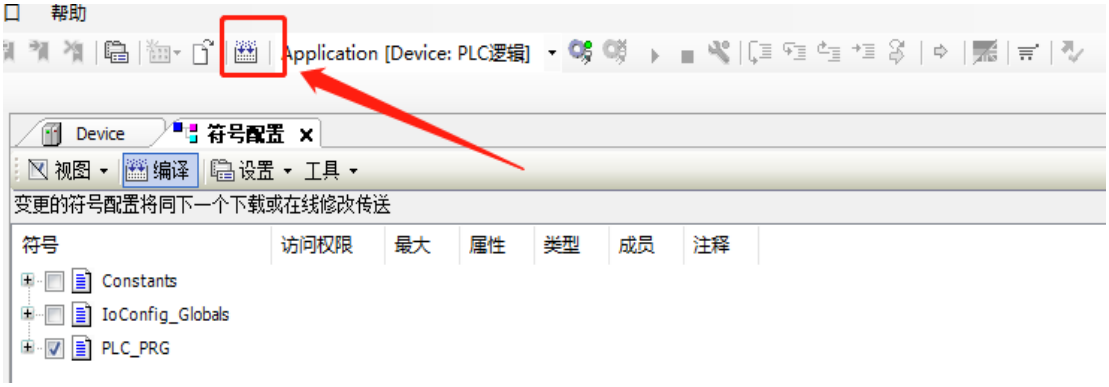


图 **16-4** 编译工程

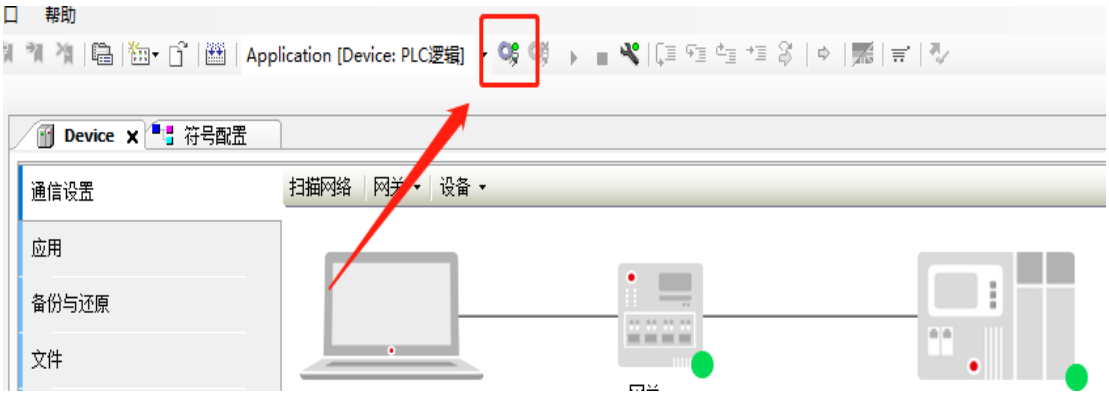


图 **16-5** 下载工程

16.5 OPC UA客户端连接示例

在上文配置好OPC UA服务器之后，本节说明使用1个客户端连接服务器的方法。

16.5.1 匿名登录方式

1. 点击“扫描网络”，登录到控制器。
2. 在“Device”页面，点击“设备 > 更改通信规则”。
3. 在弹出的对话框中勾选“Allow anonymous login”选项，点击确定。

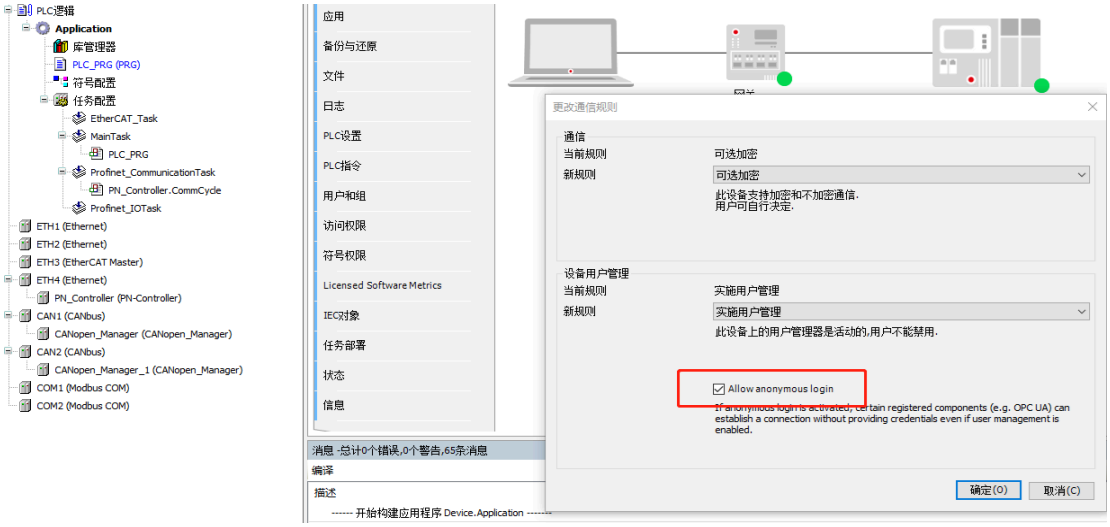


图 16-6 勾选“Allow anonymous login”

4. 打开UaExpert客户端，添加服务器。

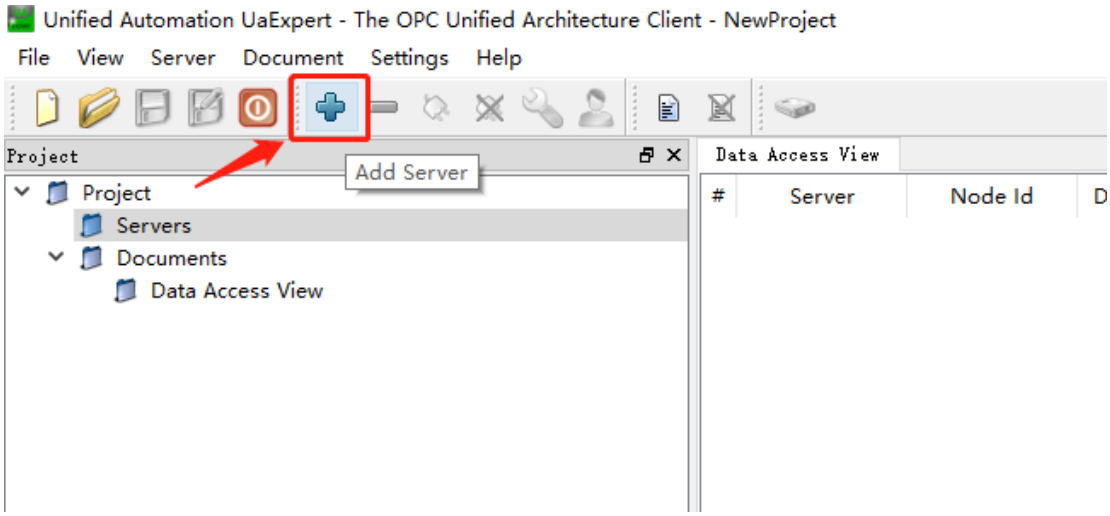


图 16-7 添加服务器

5. 输入控制器的IP地址，选择“Anonymous（匿名登录）”，点击OK。

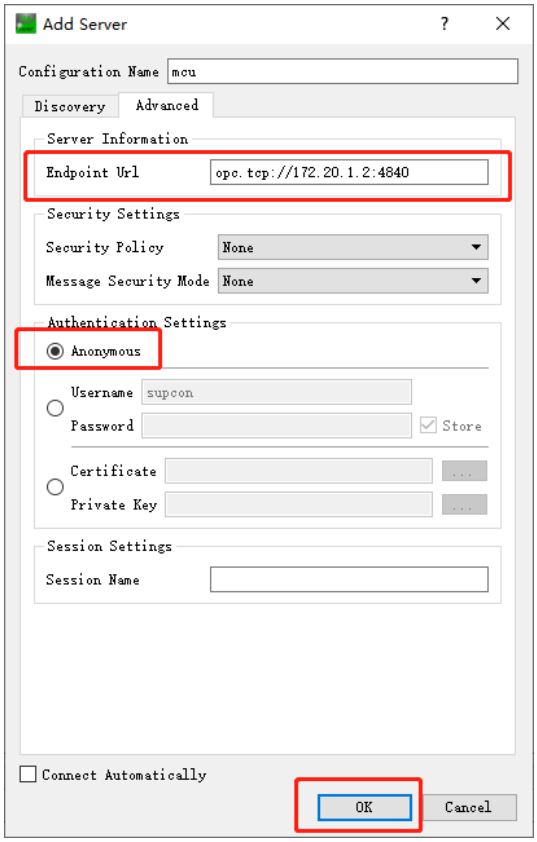


图 **16-8** 输入控制器IP

6. 按下图所示连接服务器。

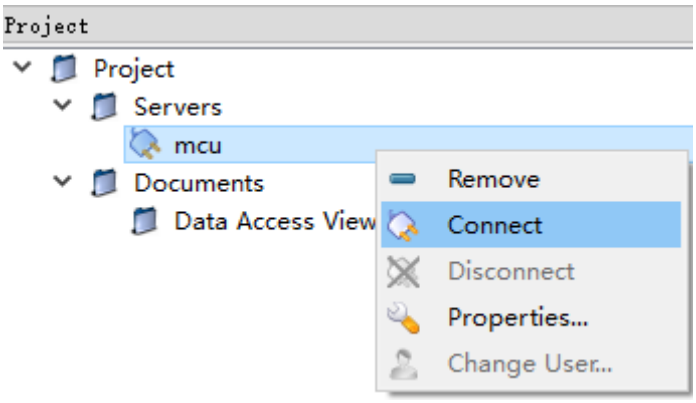


图 **16-9** 连接服务器

7. 连接成功，左侧显示访问到控制器的数据模型。

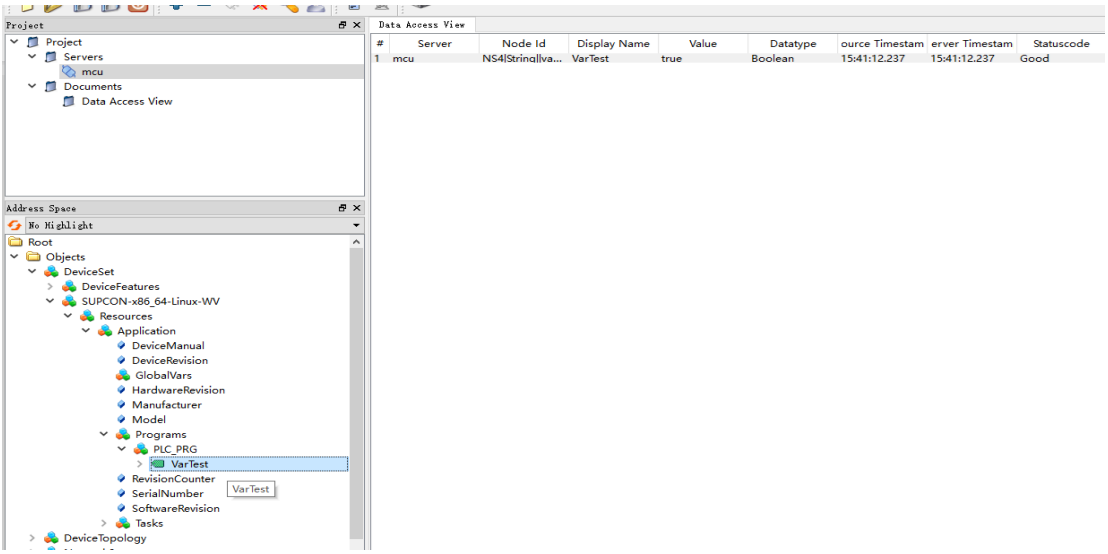


图 **16-10** 连接成功

16.5.2 用户名密码登录方式

1. 如果开启了匿名登录，则需要先**取消**匿名登录方式，后**重启控制器**。

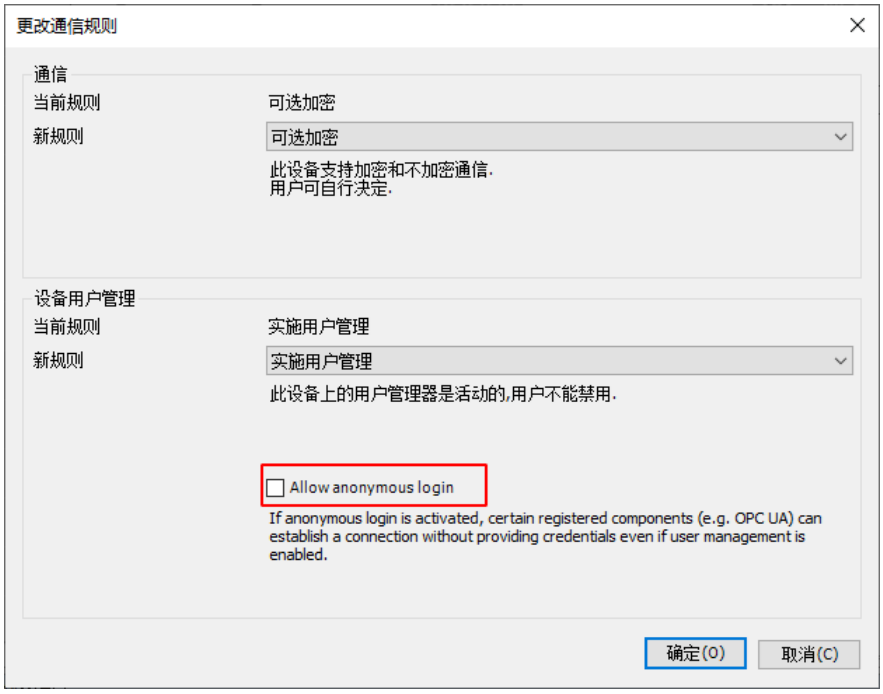


图 **16-11** 取消匿名登录

2. 打开UaExpert客户端，添加服务器配置连接参数，用户名和密码为MotionPro连接控制器时的用户名和密码。

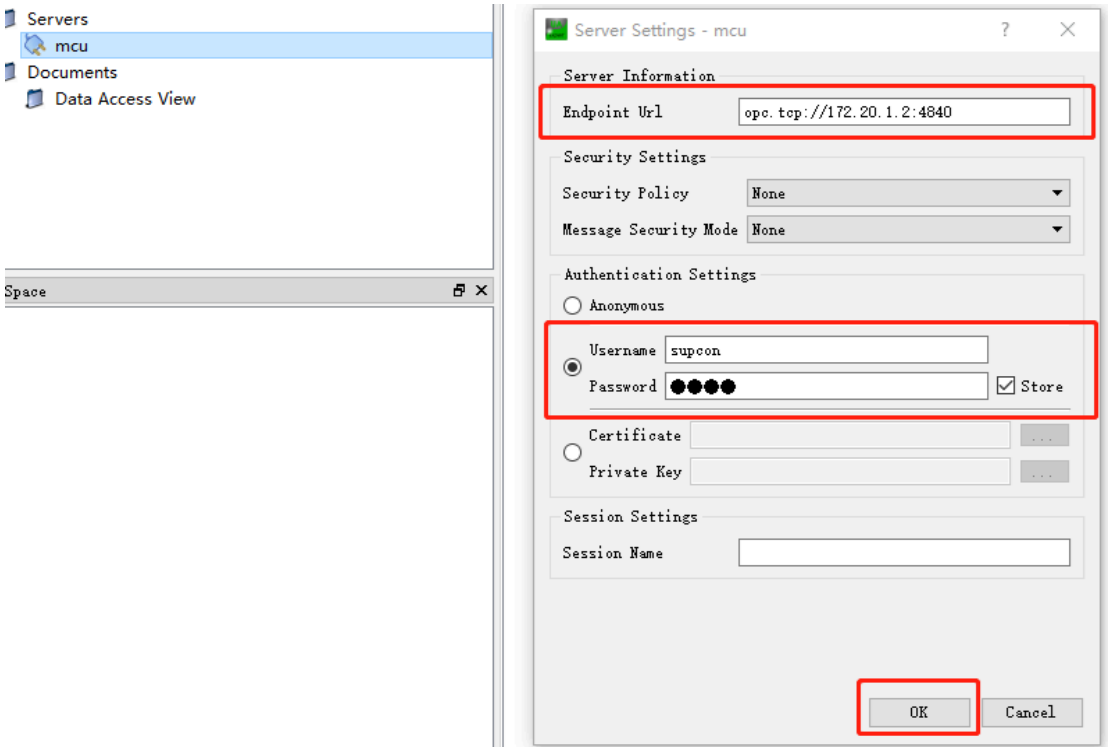


图 **16-12** 配置服务器参数

3. 连接服务器，会弹出警告窗口，提示证书不被信任。点击“Trust Server Certificate”后，点击“Continue”即可连接。

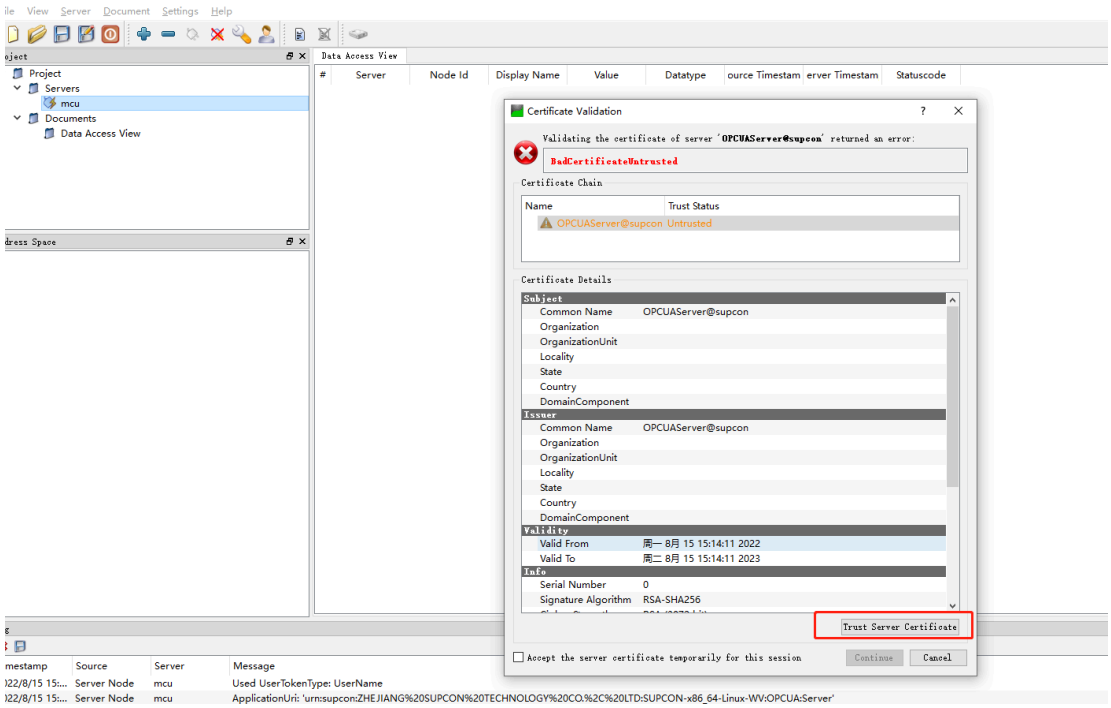


图 **16-13** 信任证书后连接

16.5.3 用户名密码+通信加密登录方式

1. 如果开启了匿名登录，则需要先**取消**匿名登录方式后**重启控制器**，如图 16-11 所示。
2. 打开UaExpert客户端，添加服务器配置连接参数，选择加密参数，用户名和密码为MotionPro连接控制器时的用户名和密码，如下图所示。

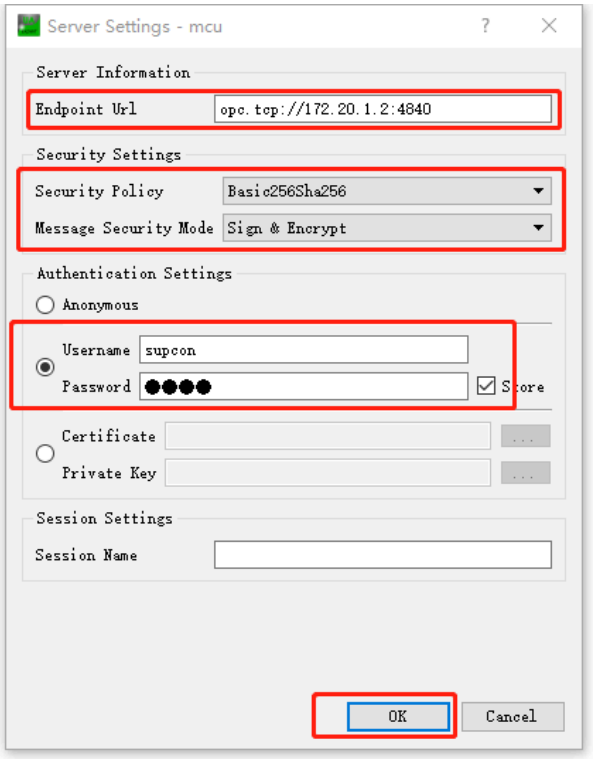


图 16-14 配置加密参数

3. 连接服务器，点击连接后会报错，如下图，这是因为控制器尚未添加客户端的证书。

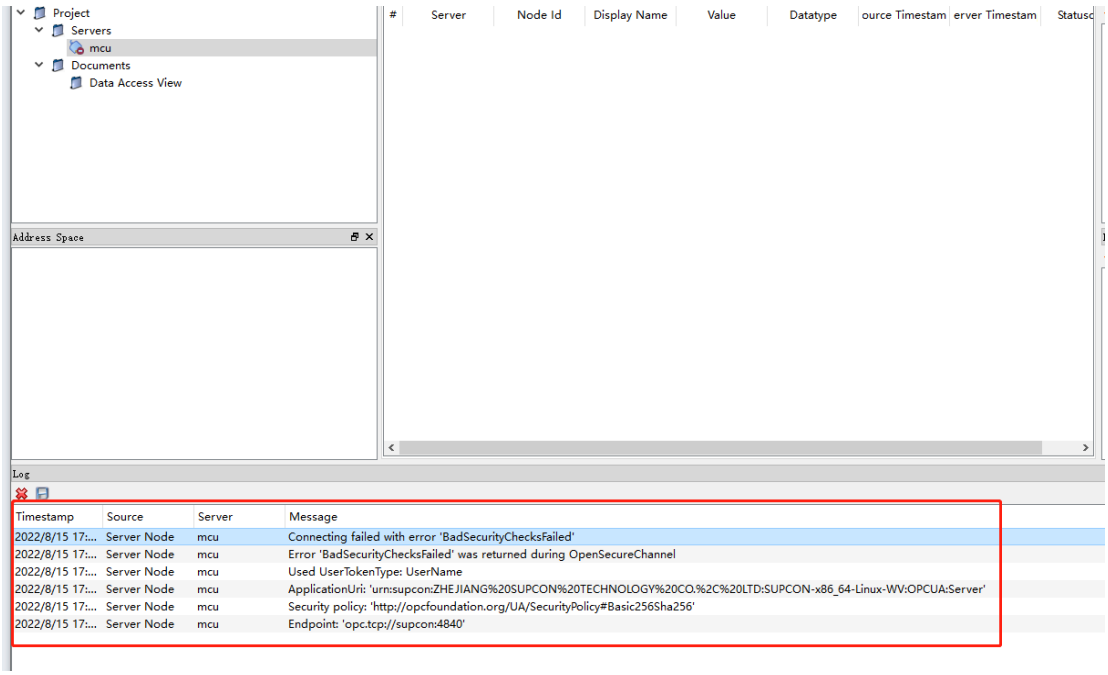


图 16-15 未添加证书将连接报错

4. 在控制器中添加信任证书。在“安全栅栏”页面点击刷新，会出现客户端的证书。

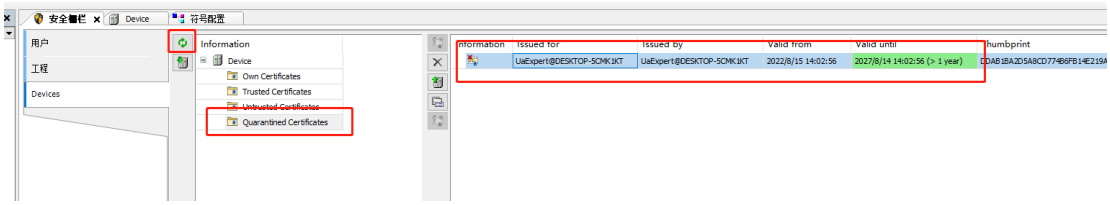


图 16-16 刷新证书

5. 点击左键将证书拖到“Trusted Certificates”中。

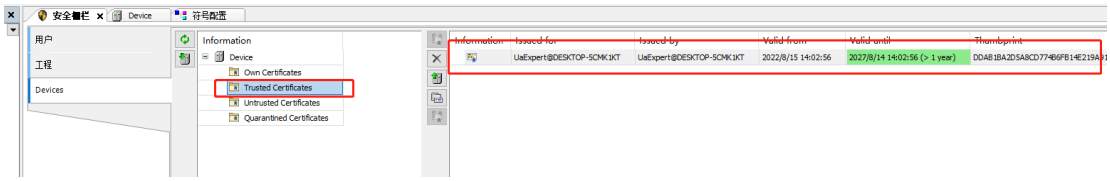


图 16-17 将证书添加到Trusted Certificates

6. 再次连接服务器，会弹出警告对话框，点击“Ignore”即可连接。

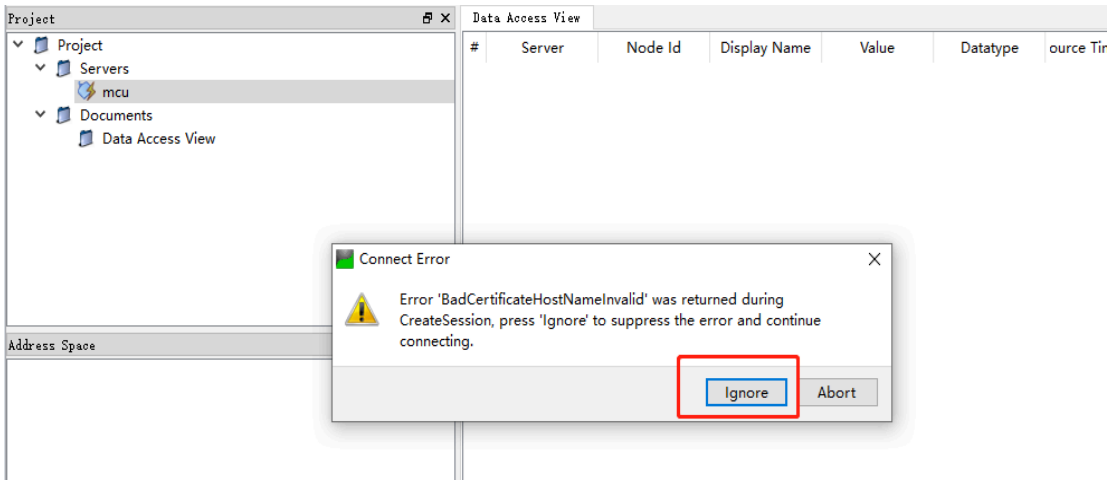


图 16-18 再次连接服务器

17 EtherCAT主站使用说明

本节介绍控制器作为EtherCAT主站时的应用说明。

17.1 功能概述

- 支持1ms分布式时钟
- 1路，每路最大支持128个从站
- 支持线性，星型网络
- 支持网线热插拔
- 支持在线调试和IEC程序两种方式进行总线诊断
- 支持从站扫描
- 支持CoE/SDO，EoE，SoE，FoE，VoE协议

17.2 配置方法

1. 准备好从站配置文件（*.xml文件）。在MotionPro的菜单栏，点击“工具 > 设备存储库 > 安装”，弹出“安装设备描述”界面，选中配置文件，点击“打开”，添加从站配置文件。

注意:

在“安装设备描述”界面，右下角文件类型点选“Ethercat ESI(*.xml)”，否则界面可能无法显示配置文件。

2. 完成添加后，以在设备存储库中能找到添加的从站为准。

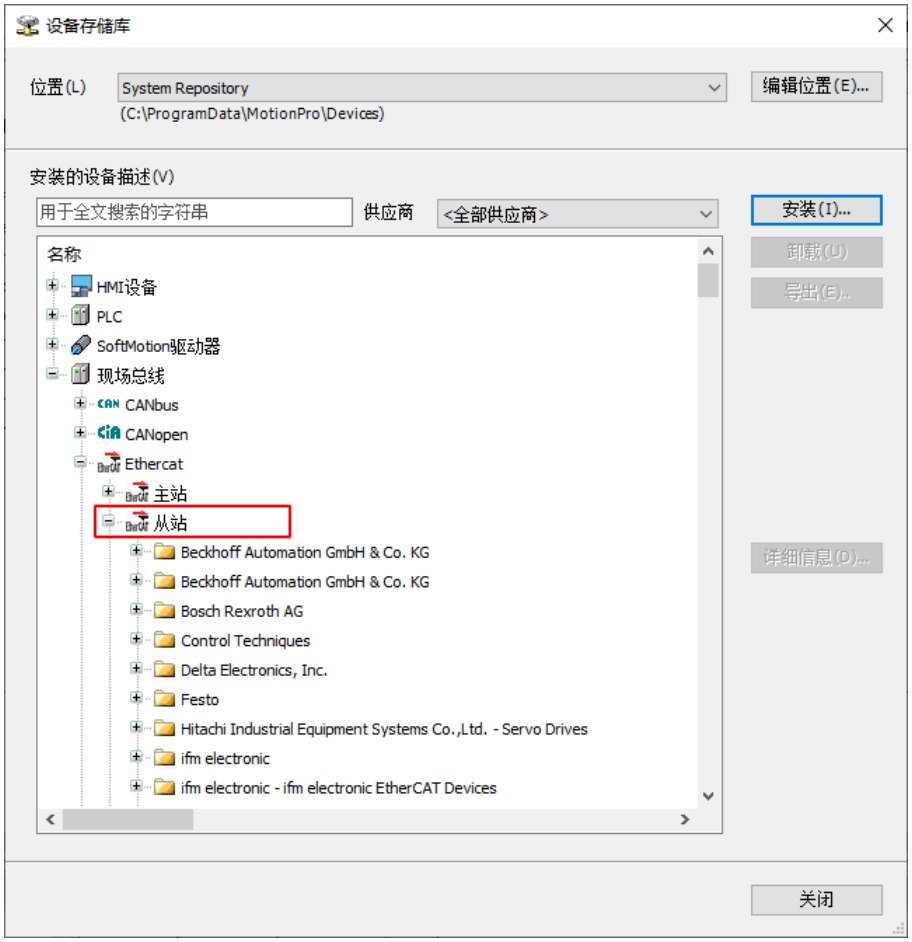


图 **17-1** 添加从站

3. 新建工程，将ETH3(EtherCAT Master)切换为使能，后编译，组态下载，进入联机调试状态。

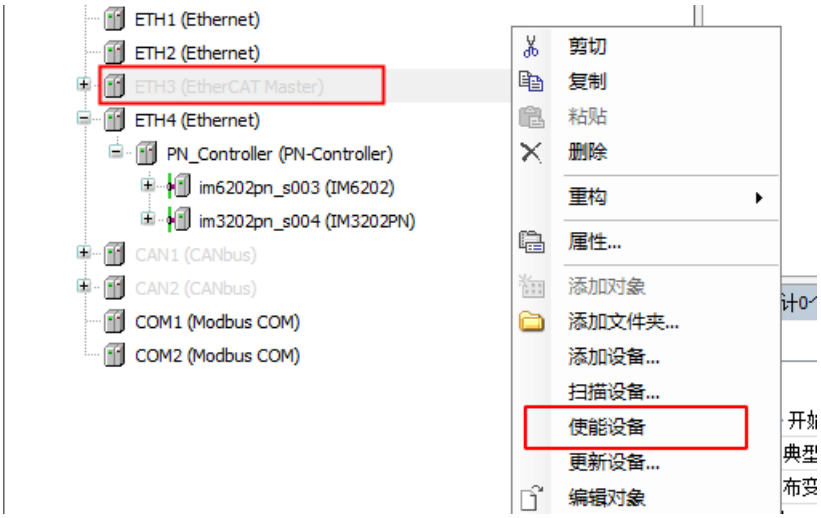


图 **17-2** 使能设备

4. 将ETH3口连接从站的IN口。右键点击ETH3(EtherCAT Master)，选择扫描设备。

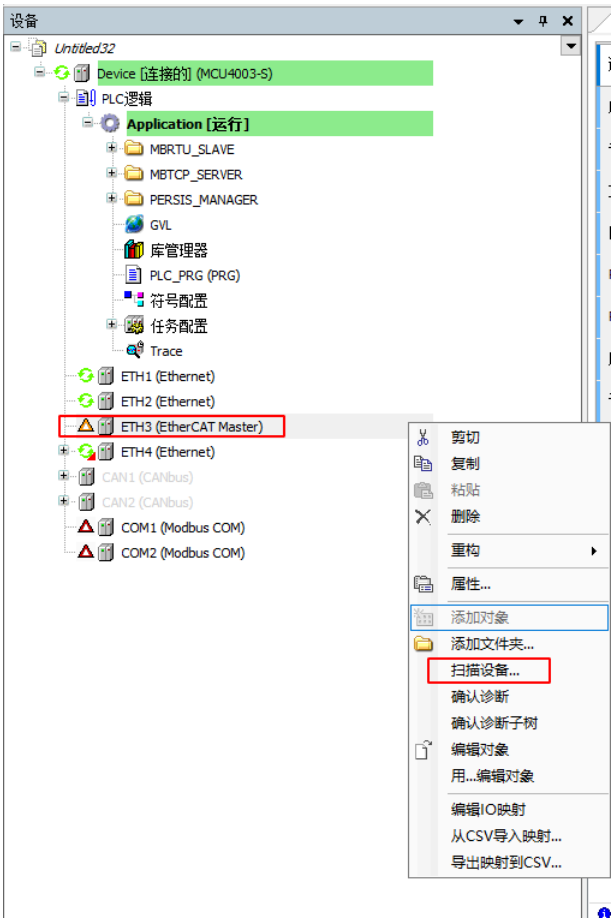


图 17-3 扫描设备

5. 如果设备连接无误且正常工作，则主站可以正常扫描到从站。点击“复制所有设备到工程”，完成从站添加。

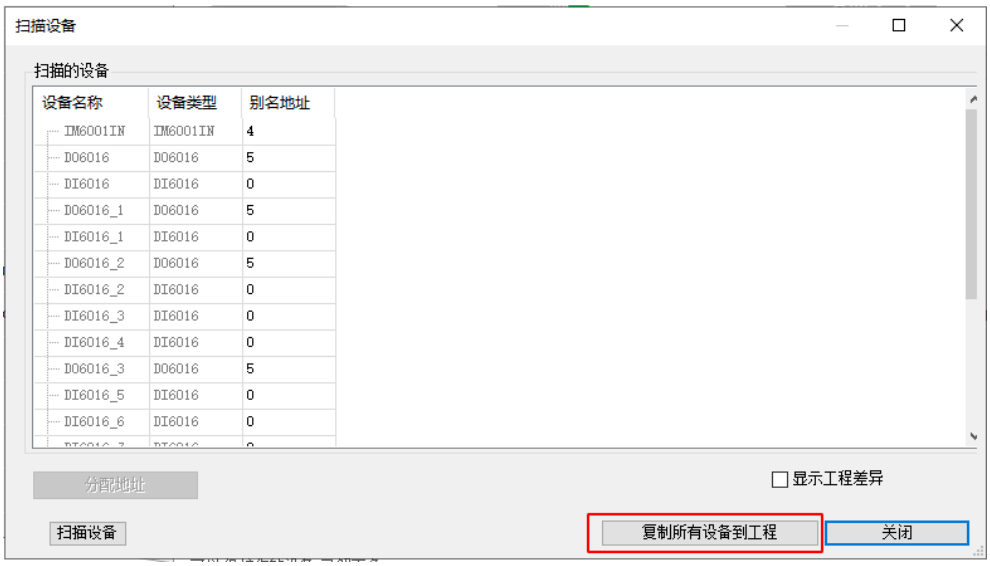


图 17-4 复制所有设备到工程

6. 退出联机调试状态，对EtherCAT主站进行参数配置。常用配置项为“主站分布时钟周期、自动重启从站”选项等。

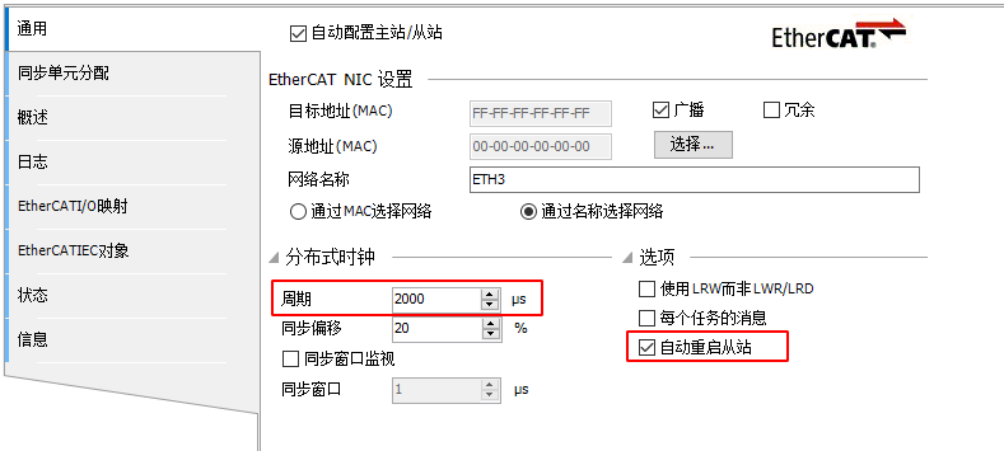


图 17-5 配置EtherCAT主站参数

7. 从站配置一般包括启动参数配置、I/O数据绑定等。

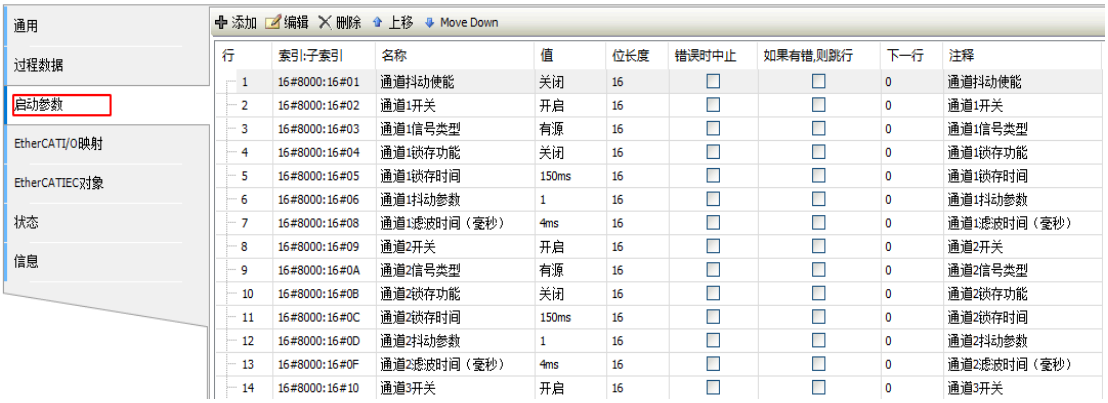


图 17-6 启动参数配置

图 17-7 EtherCAT I/O映射

8. 重新编译、下载、运行，各设备正常运行（显示绿色）后，即完成EtherCAT主站功能配置。

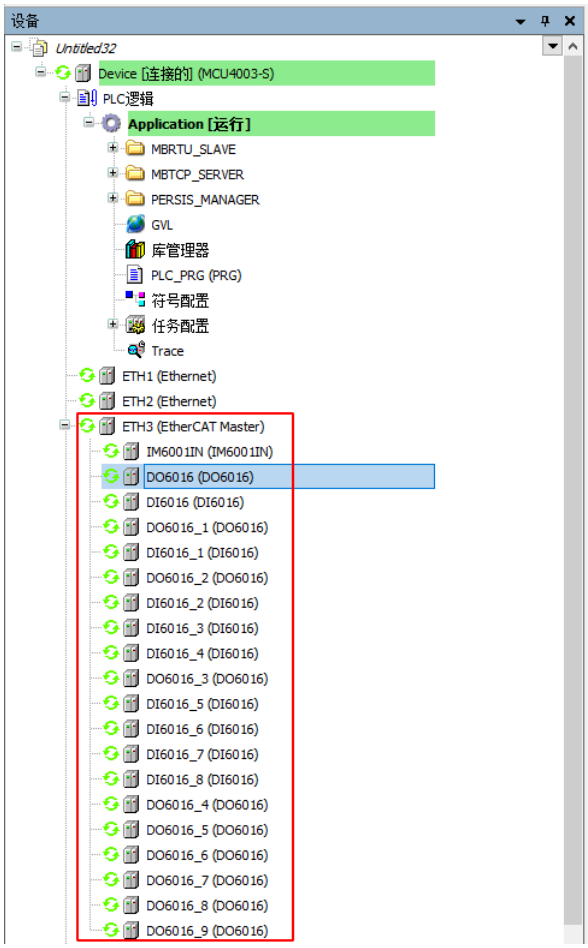


图 **17-8** 设备正常运行

9. 联机调试状态下，ETH3(EtherCAT Master)的状态页，可以查看EtherCAT主站的各项收发包统计。

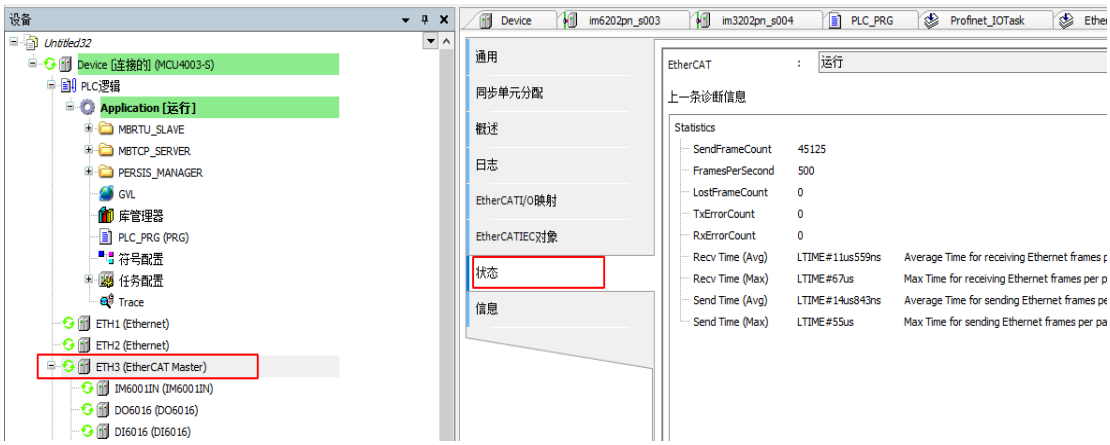


图 **17-9** 查看**EtherCAT Master**主站状态

10. 联机调试状态下，ETH3(EtherCAT Master)的日志页，可以查看EtherCAT主站的运行日志。

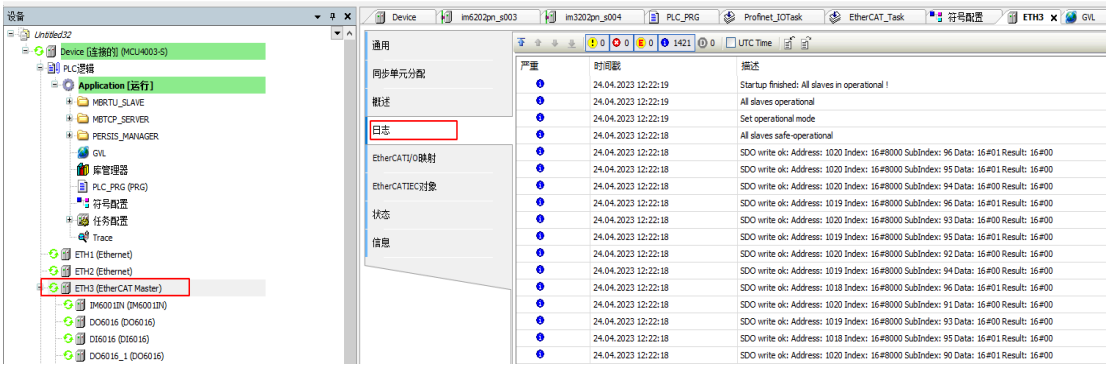


图 17-10 查看EtherCAT主站运行日志

18 C/C++高级语言编程

本章主要介绍如何通过C/C++语言，实现应用程序与IEC程序协同运行，实现控制器的功能扩展。

18.1 通过C/C++实现APP应用程序

本节主要介绍通过C/C++语言实现应用程序的方法。

18.1.1 原理简述

通过C/C++实现APP应用程序原理如下图所示。

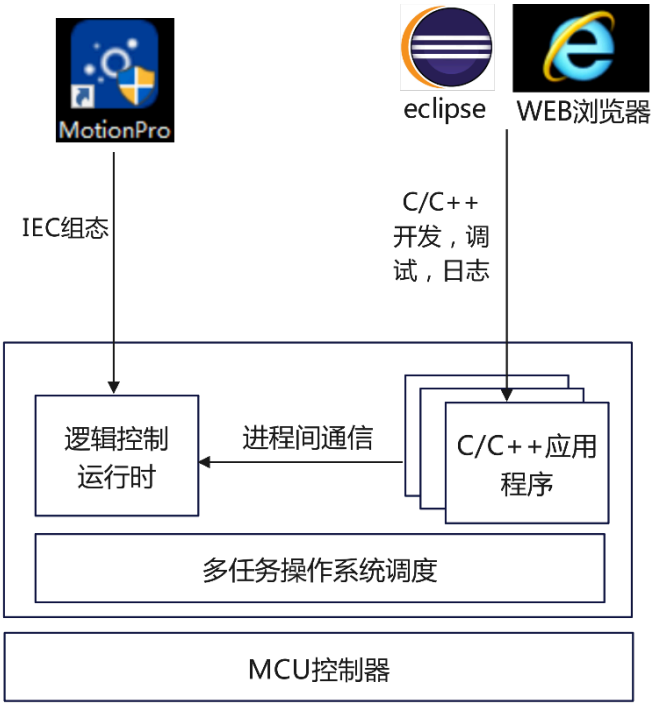


图 18-1 C/C++实现APP应用程序原理

18.1.2 功能指标

- 支持C编程语言，兼容LINUX C应用程序标准。
- 支持C++编程语言，支持STL标准模版库。
- Eclipse IDE环境编程，采用GCC编译器。
- 开放API，支持读写逻辑控制运行时侧的IEC变量，支持基本类型，数组类型。
- 支持日志打印API，持久化存储，可通过WEB浏览器查看。
- 进程间通信性能参考：读1KB数组变量耗时500us，写1KB数组耗时1ms。
- 最大支持10个C/C++应用程序同时部署运行。

18.1.3 使用说明

1. 在MotionPro环境中对控制器进行组态，在符号配置界面中，将需要开放给C/C++应用程序访问的位号勾选，点击编译，下载。

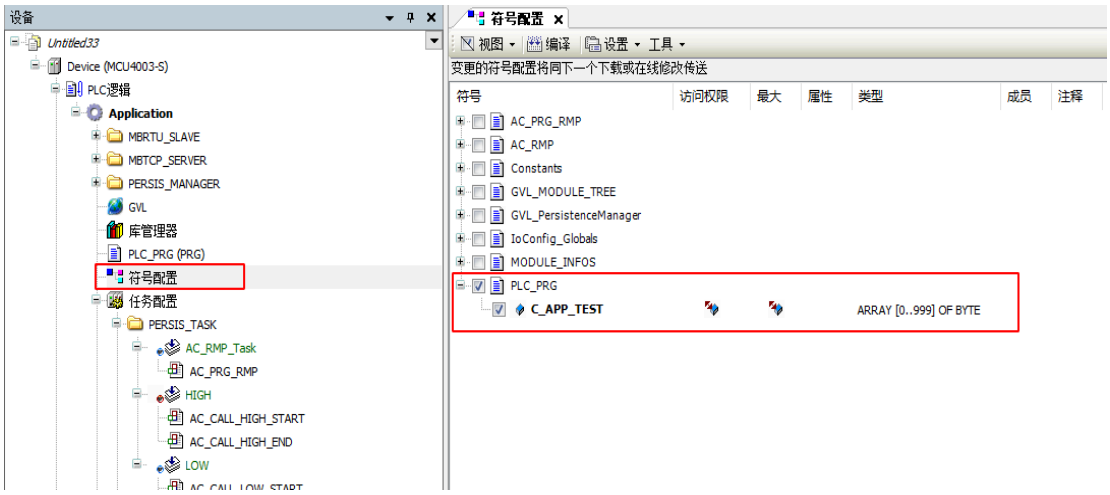


图 **18-2** 勾选位号

2. 在Eclipse环境中，新建C/C++工程（可向中控获取模版工程），编辑应用程序代码，至编译通过，生成*.capp文件。

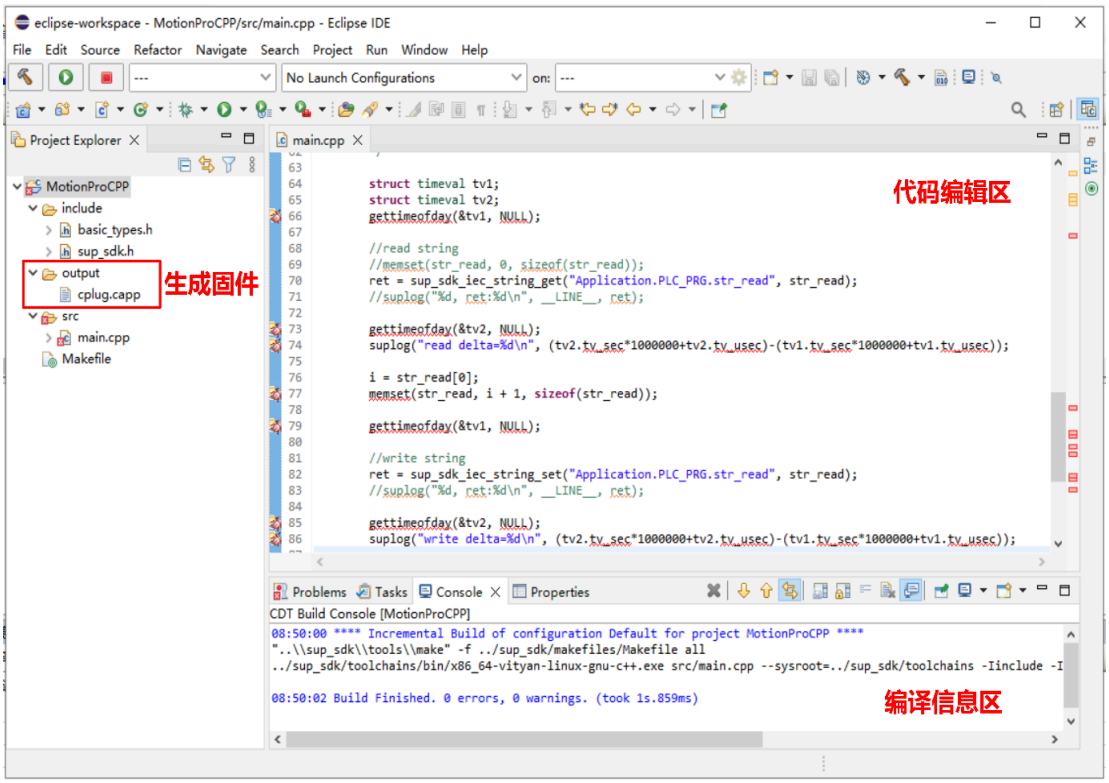


图 **18-3** 新建C/C++工程

3. 打开WEB浏览器，使用Admin账号登录，在C程序页面，点击“选择文件”，选中在eclipse IDE编译好的*.capp固件，点击导入。

图 18-4 选择文件导入

4. 控制器会请求重启。

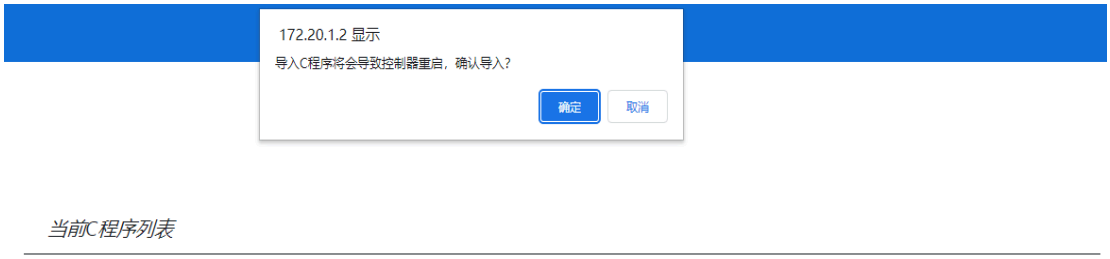


图 18-5 重启控制器

5. 完成重启后，重新登入WEB。在“C程序管理”页面可以查看到当前C程序列表，指示当前程序是否在运行状态。

图 18-6 查看程序运行状态

在C程序日志界面，可以查看到C程序通过syslog接口函数打印的日志。

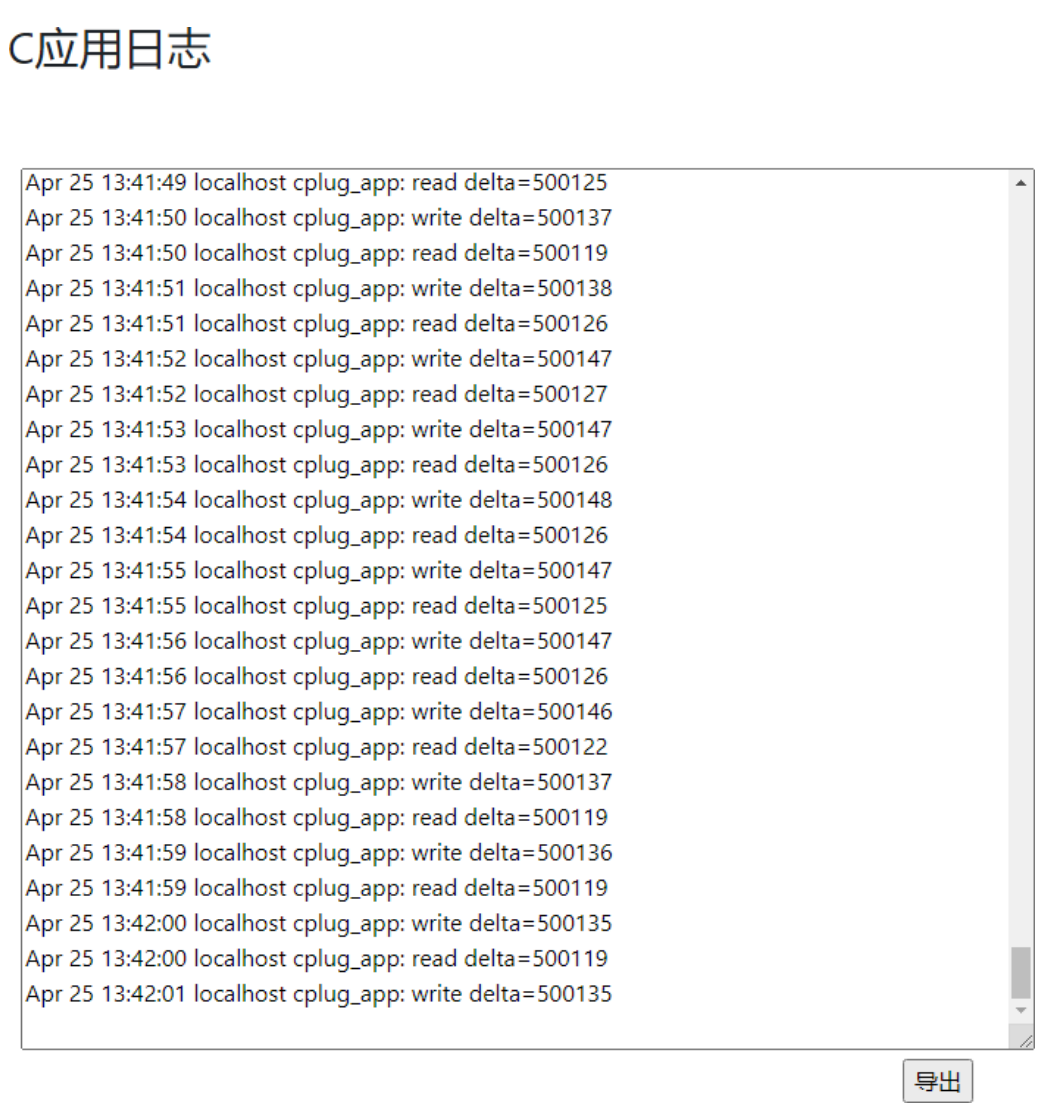


图 18-7 查看C应用日志

18.2 通过C/C++实现IEC功能块

本节主要介绍如何通过C/C++语言，实现算法功能块与IEC程序混合运行，实现控制器的功能扩展。

18.2.1 原理简述

通过C/C++实现IEC功能块原理如下图所示。

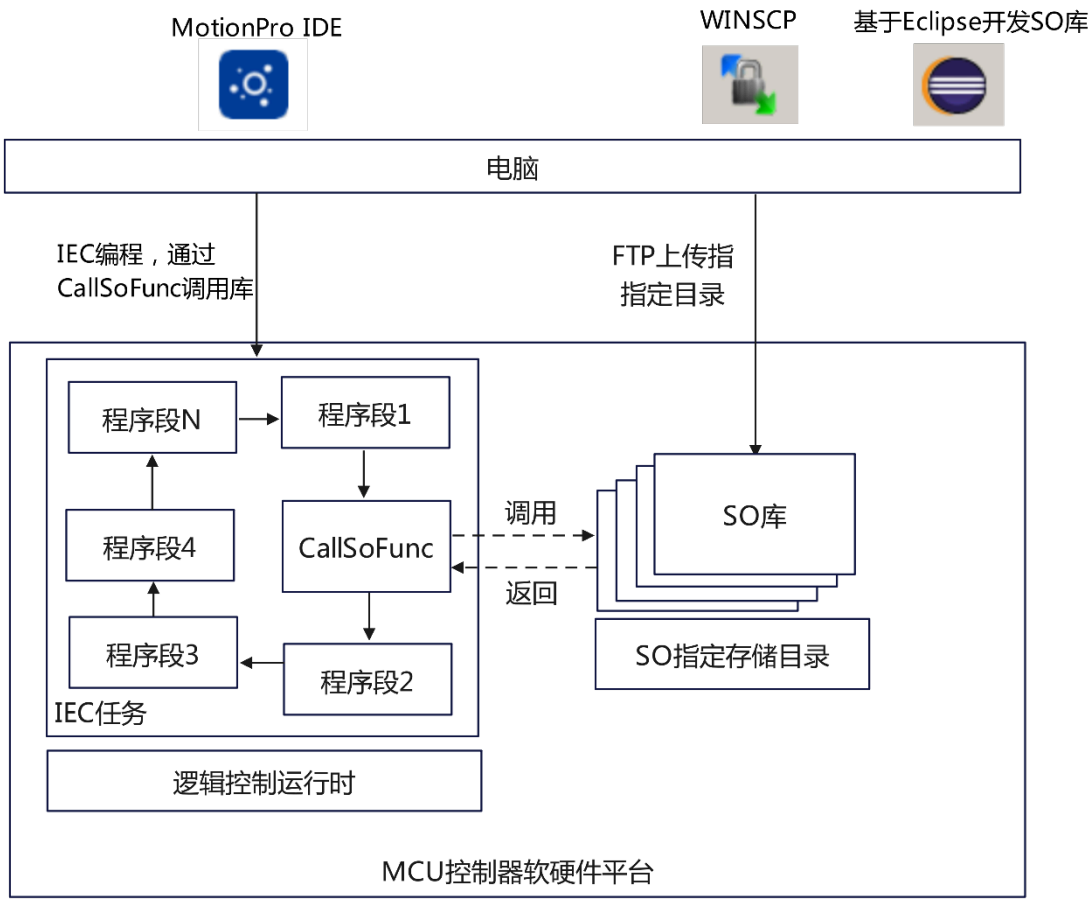


图 18-8 通过C/C++实现IEC功能块原理图

18.2.2 功能指标

- 支持C编程语言，兼容LINUX C应用程序标准。
- 支持C++编程语言，支持STL标准模版库。
- 支持静态库开发和动态库开发部署。
- Eclipse IDE环境编程，采用GCC编译器。
- 支持日志打印API，持久化存储，可通过WEB浏览器查看。
- 通过FTP上传SO库，通过CallSoFunction接口调用SO库。
- 单次SO调用时间开销：300us。
- 最大支持10个SO库部署运行。

18.2.3 使用说明

本节介绍开发一个静态库/动态库的开发方法。

工程导入

1. 打开Eclipse IDE，在菜单栏中点击“File > Import”。
2. 在Import界面选择Existing Project into Workspace后单击Next。

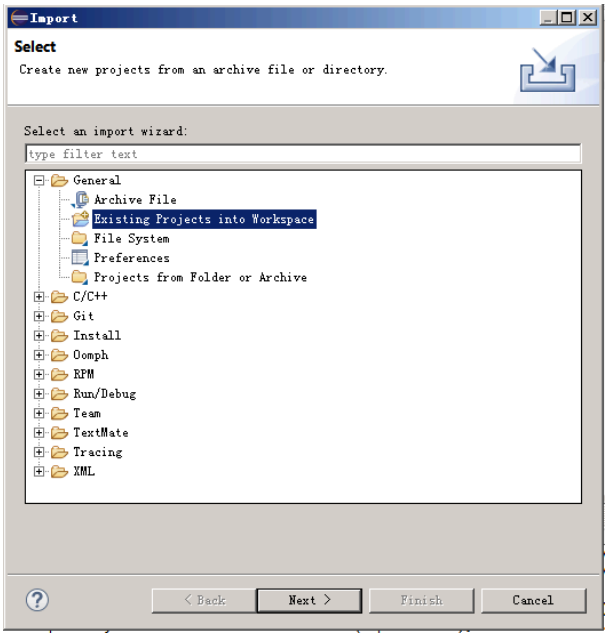


图 18-9 选择Existing Project into Workspace

3. 在Import界面点击Browse，在弹出的文件选择框中选择“CPP_静态库模版工程”文件夹后，点击“Finish”完成导入。

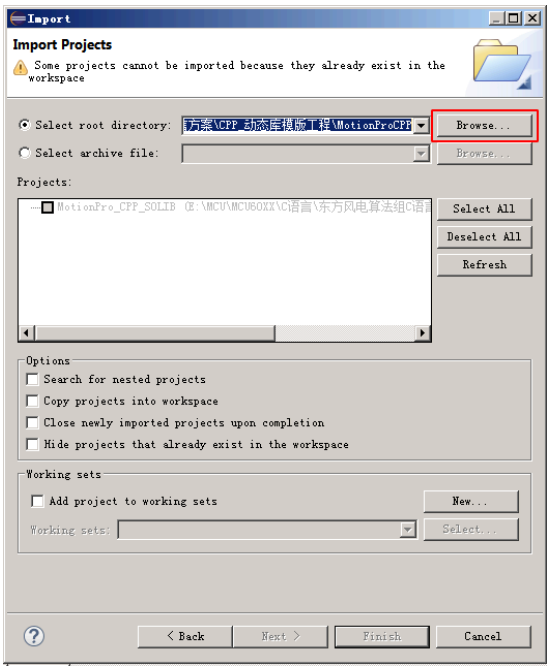


图 18-10 选择CPP_静态库模版工程

4. 完成导入后，Eclipse IDE界面如下，1是工程目录树，2是代码编辑区，3是信息输出区。

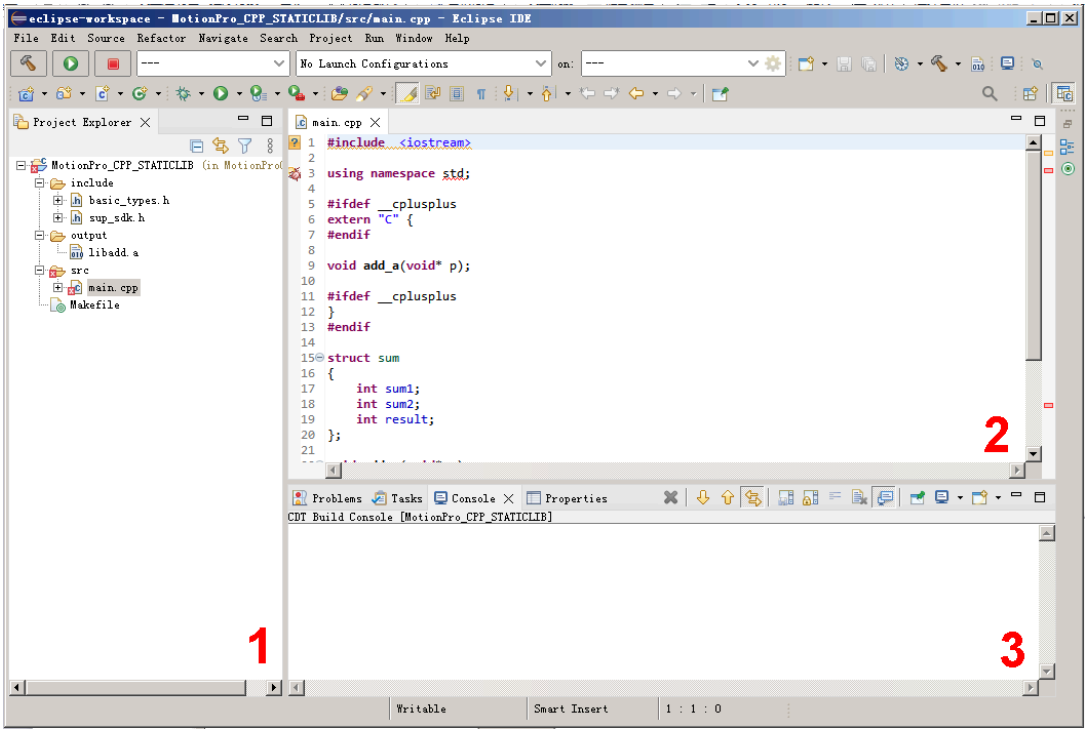


图 **18-11** 导入后**Eclipse**界面

5. 右键点击工程，弹出的菜单选择Build Project，即可编译工程。

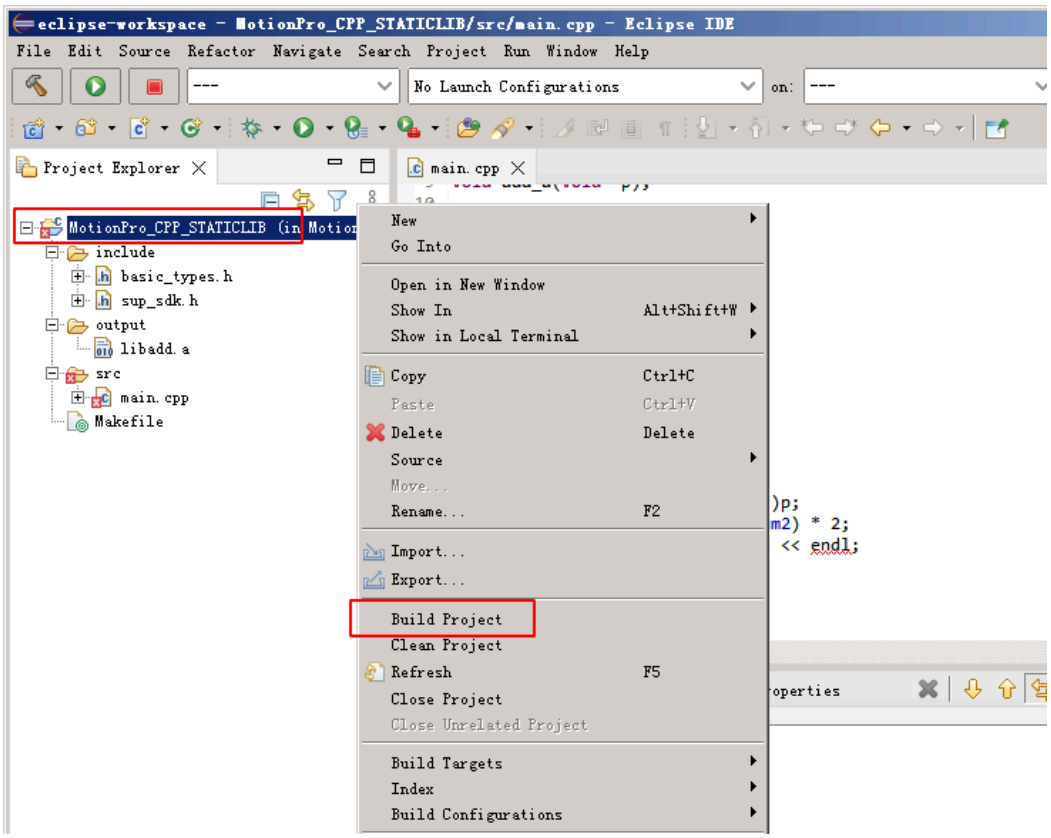


图 **18-12 Build Project**

静态库开发

静态库模版工程的目录结构如下图所示。

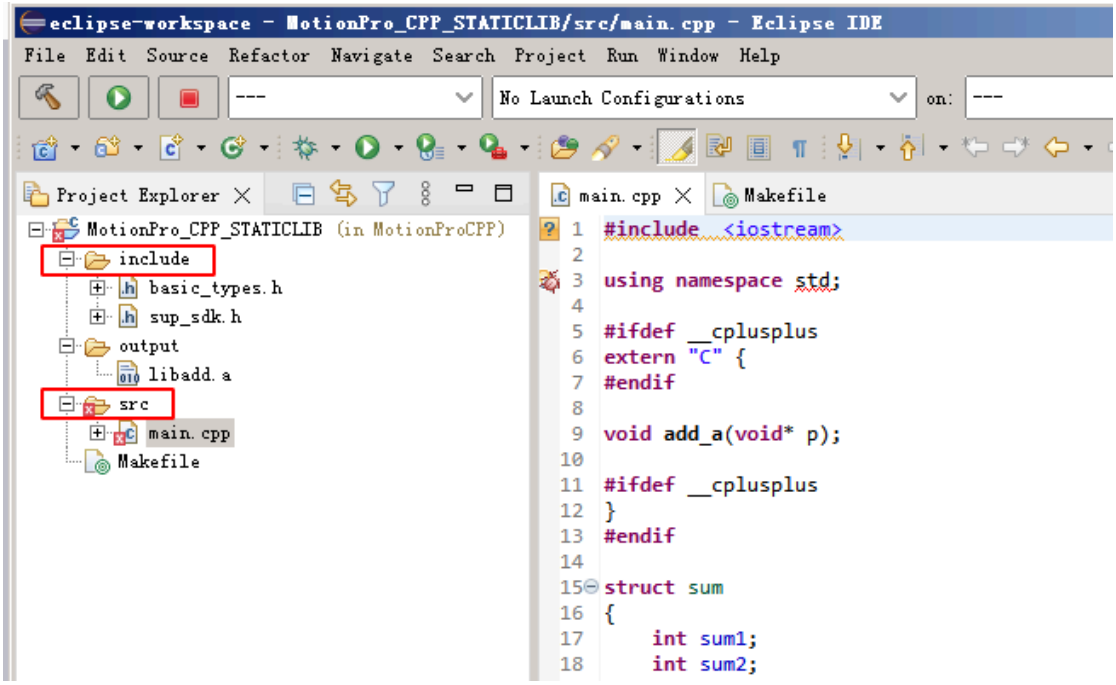


图 18-13 静态库模板工程目录结构

在静态库模板工程下，需完成以下操作。

1. 开发工程师需要将.h头文件拷贝到include目录，将.cpp源码文件拷贝到src目录。修改Makefile中的TARGET字段，指定输出的静态库名称。注意，静态库必须是lib+“名称”+.a的命名格式，如下图所示。

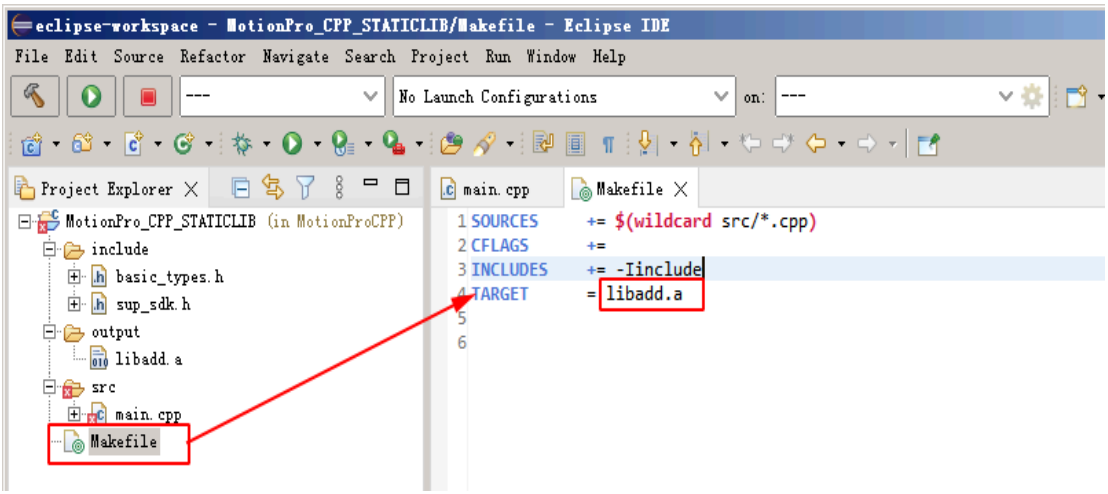


图 18-14 指定输出的静态库名称

2. 添加完毕后，执行Build Project即可在output目录得到静态库输出。
如果源码与头文件有多级目录，则需要修改Makefile文件，举例如下。
Include目录下新增subinc目录，内有.h文件，src目录下新增subsrc目录，内有.cpp文件，则Makefile需要相应新增：

```
SOURCES += $(wildcard src/subsrc/*.cpp) # 新增cpp源码路径psrc/subsrc
```

```
INCLUDES += -Iinclude/subinc # 新增头文件路径include/subinc
```

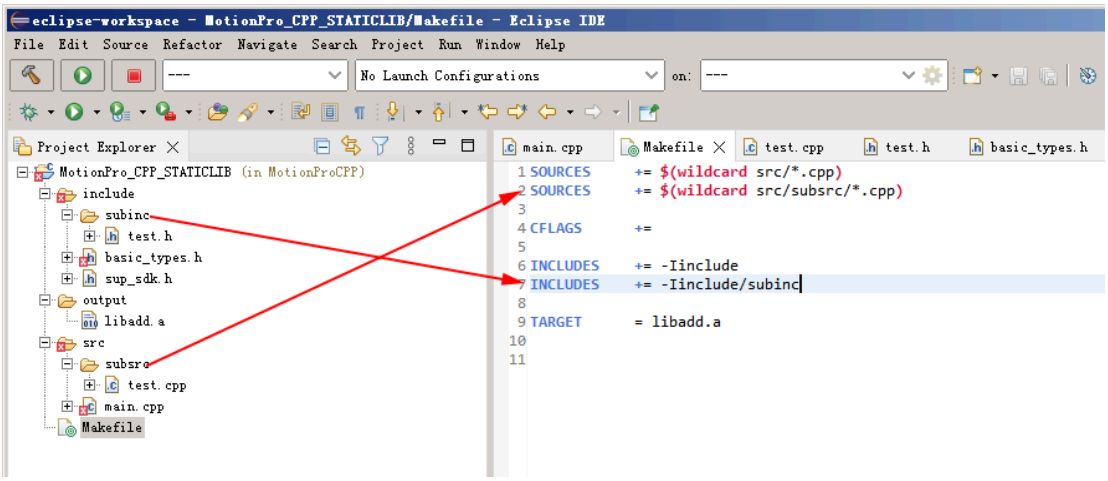


图 **18-15** 修改**Makefile**文件

动态库开发

动态库模版工程的目录结构如下图所示。

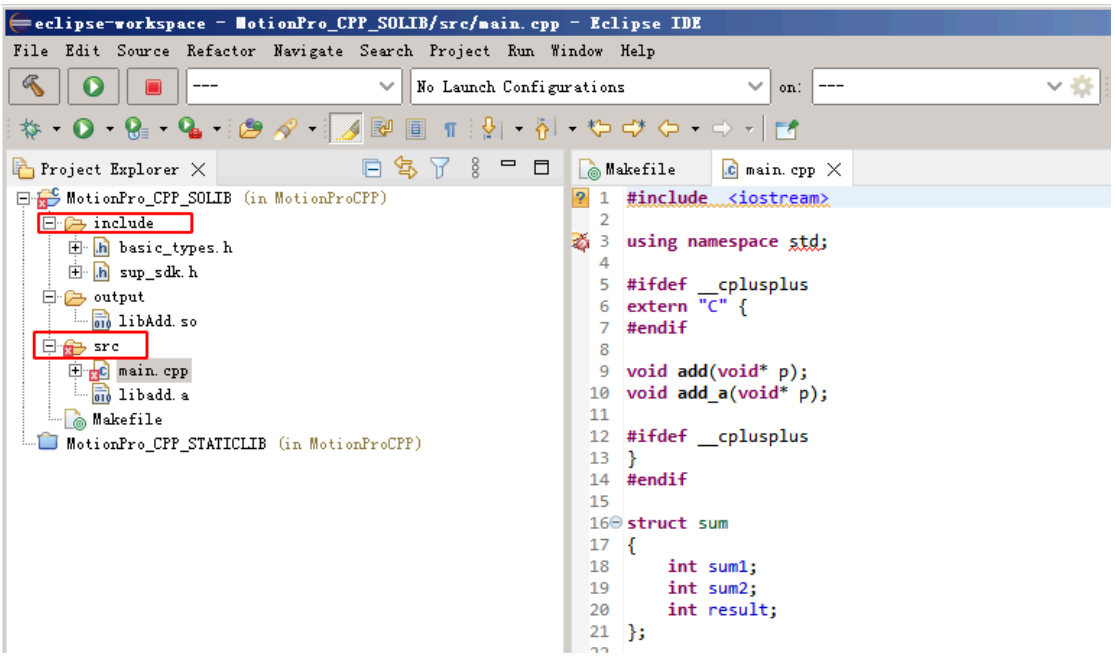


图 **18-16** 动态库模版工程目录结构

动态库开发与静态库开发大体类似，有以下几点区别。

1. 在修改Makefile的TARGET时，动态库必须是lib+“名称”+.so的命名格式。

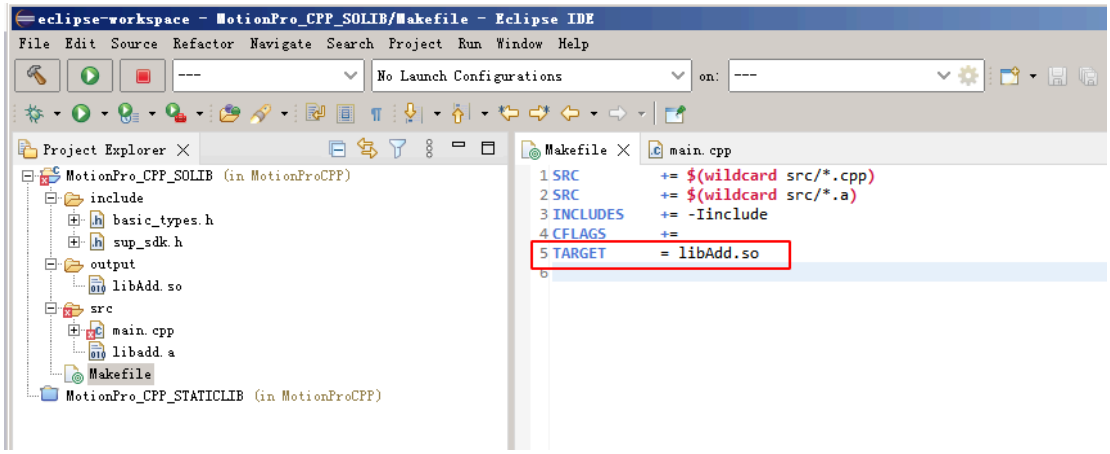


图 18-17 动态库命名格式

2. 如果动态库中需要引用静态库，则将静态库作为cpp源文件，直接拷贝到src目录即可。

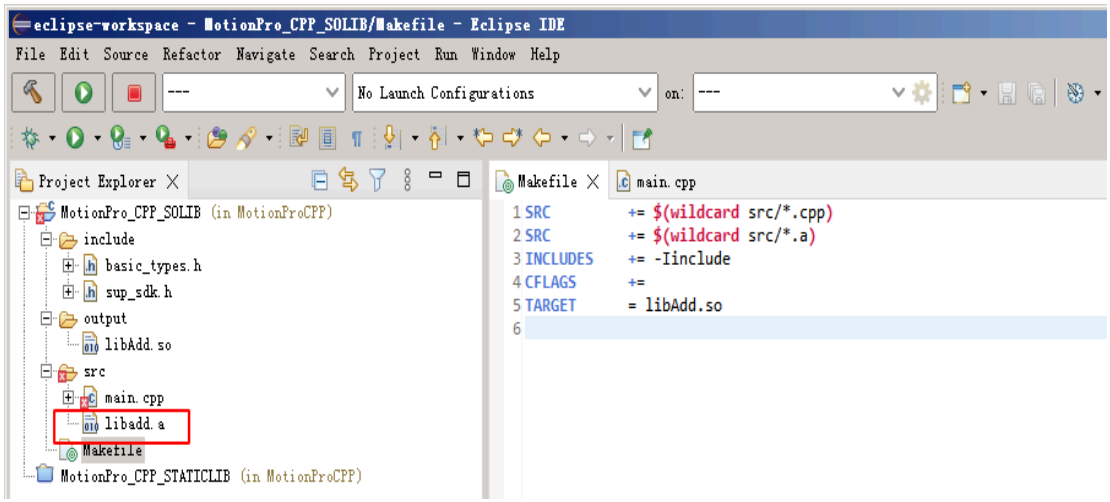


图 18-18 引用静态库

3. 次级目录处理方法同“静态库开发”。如果次级目录中有静态库引用，则Makefile中需要加上下述语句。

```
SOURCES += $(wildcard src/subdir/*.a) # 新增.a库路径src/subdir
```

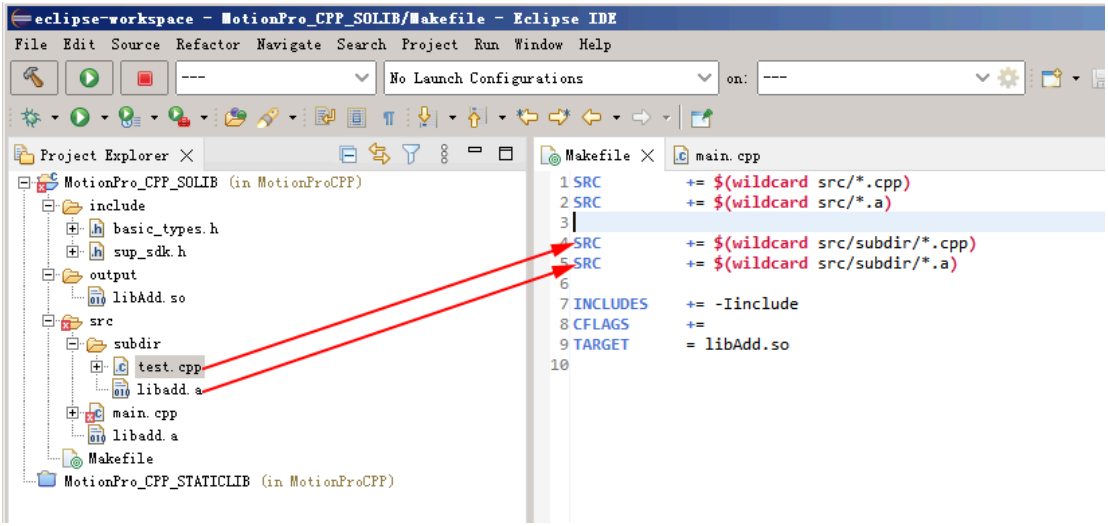


图 **18-19** 次级目录中包含静态引用

部署应用

1. 将开发好的算法块(.so)文件，通过Winscp上传到控制器的/opt/codesys/clibs目录下。

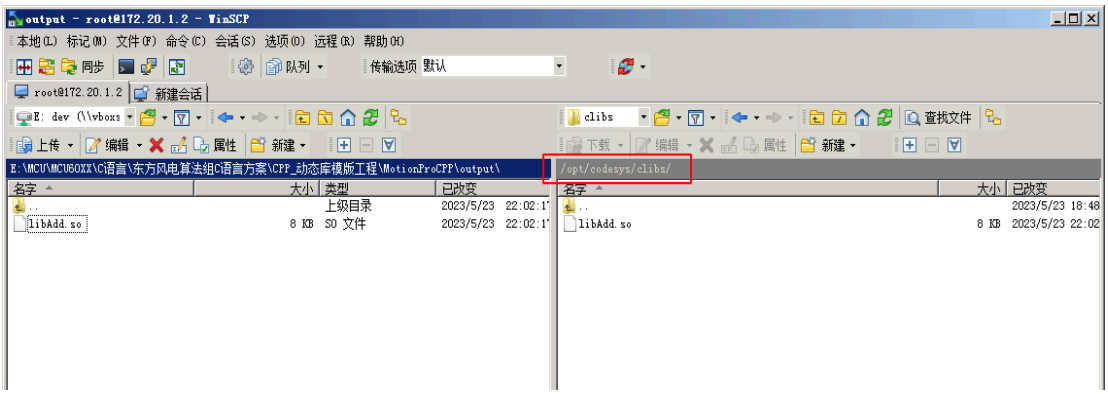


图 **18-20** 上传算法块**(.so)**文件

2. 在MotionPro的工程里，通过CallSOFunction函数访问so。

函数名	参数	说明
CallSOFunction	LibName	库名称，除去lib和.so后的名称。如so名称为libadd.so，则库名称为add
	FuncName	so库中的函数名
	Paras	函数参数地址。需使用ADR命令获取参数地址

函数名	参数	说明
	返回值	C_CALL_ERR： 0：调用成功 1：参数为空 2：没有找到库 3：没有找到函数 4：其他情况下调用失败

3. 通过CallSOFunction调用libadd.so中的add接口使用示例。

```
TYPE add_params :
STRUCT
a : DINT; b : DINT; c : DINT;
END_STRUCT
END_TYPE
PROGRAM PLC_PRG
VAR
dRet: C_CALL_ERR;
param: add_params := (a := 1, b := 2);
END_VAR
dRet := CallSOFunction('add', 'add', ADR(param));
```

19 常用编程库一览

本章介绍常用的编程库，及其下包含的指令用法。

19.1 Standard库

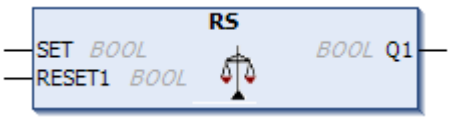
本节说明Standard库所包含的基本指令。

19.1.1 Bistable Function Blocks

本节介绍双稳态功能块指令。

RS

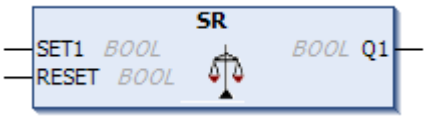
实现双稳态复位，支配锁存。




```
RS(SET:= ,RESET1:= ,Q1=> );
```

SR

实现双稳态集，支配锁存。



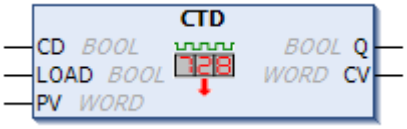

```
RS(SET1:= ,RESET:= ,Q1=> );
```

19.1.2 Counter

本节介绍计数器功能块指令。

CTD

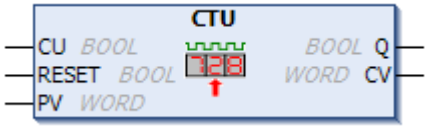
实现减计数。



```
CTD(CD:= ,  LOAD:= , PV:= ,  Q=> ,  CV=> );
```

CTU

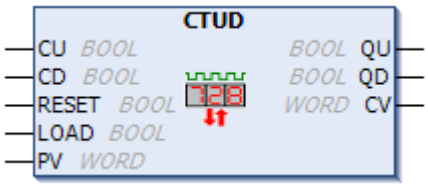
实现加计数。




```
CTD( CD:= ,  LOAD:= ,  PV:= ,  Q=> ,  CV=> );
```

CTUD

实现加减计数。




```
CTUD( CU:= ,  CD:= ,  RESET:= ,  LOAD:= ,  PV:= ,  QU=> ,  QD=> ,  CV=> );
```

19.1.3 Miscellaneous

RTC

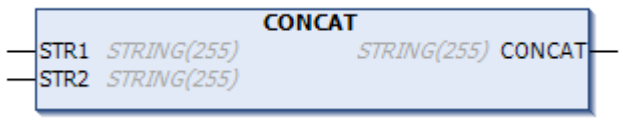
计算从给定开始时间经过的时间。


```
RTC(EN:= , PDT:= , Q=> , CDT=> );
```

19.1.4 String Functions

CONCAT

连接两个字符串。



```
CONCAT(STR1:= , STR2:= )
```

DELETE

从字符串中删除一定数量的字符。


```
DELETE(STR:= , LEN:= , POS:= )
```

FIND

在字符串中搜索部分字符串的位置。


```
FIND(STR1:= , STR2:= )
```

INSERT

将一个字符串插入到另一个字符串的特定位置。


```
INSERT(STR1:= , STR2:= , POS:= )
```

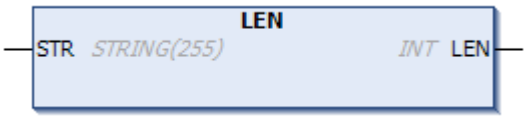
LEFT

返回字符串STR中从左边开始的第一个SIZE字符。


```
LEFT(STR:= , SIZE:= )
```

LEN

返回字符串的字符数。




```
LEN(STR:= )
```

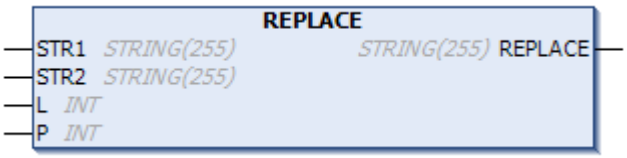
MID

从位置POS开始的STR字符串中检索LEN字符。


```
MID(STR:= , LEN:= , POS:= )
```

REPLACE

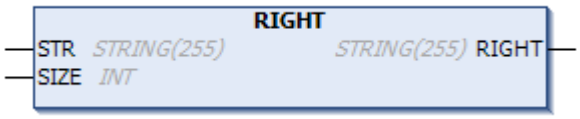
用另一个字符串替换一个字符串的特定字符数。




```
REPLACE(STR1:= , STR2:= , L:= , P:= )
```

RIGHT

返回字符串STR中从右开始的第一个SIZE字符。

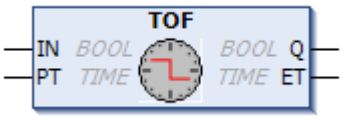



```
RIGHT(STR:= , SIZE:= )
```

19.1.5 Timer

TOF

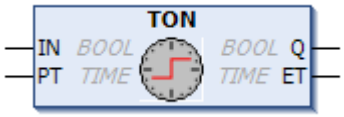
实现一个带有关闭延迟的定时器。




```
TOF(IN:= , PT:= , Q=> , ET=> );
```

TON

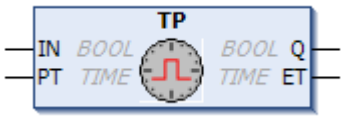
实现一个带有开启延迟的定时器。




```
TON(IN:= , PT:= , Q=> , ET=> );
```

TP

实现脉冲定时器。




```
TP(IN:= , PT:= , Q=> , ET=> );
```

19.1.6 Trigger

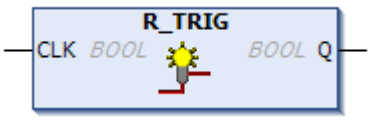
F_TRIG

检测一个下降沿的布尔信号。


```
F_TRIG(CLK:= , Q=> );
```

R_TRIG

检测一个上升沿的布尔信号。



```
R_TRIG(CLK:= , Q=> );
```

19.2 Util库

本节说明Util库所包含的基本指令。

19.2.1 Analog Monitors

HYSTERESIS

迟滞函数。


```
HYSTERESIS(IN:= , HIGH:= , LOW:= , OUT=> );
```

LIMITALARM

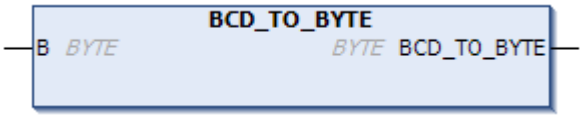
监视输入值是否在上限与下限之间。


```
LIMITALARM( IN:= ,  HIGH:= ,  LOW:= ,  O=> ,  U=> ,  IL=> );
```

19.2.2 BCD Conversions

BCD_TO_BYTE

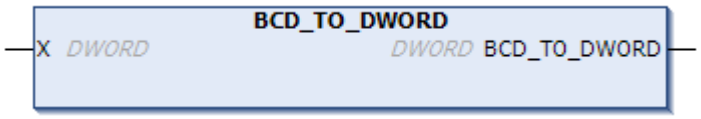
将一个BCD码字节转换为一个二进制码字节。




```
BCD_TO_BYTE(B:= )
```

BCD_TO_DWORD

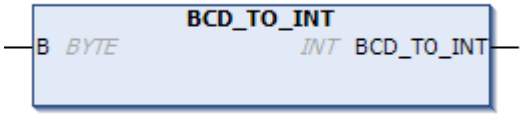
将一个BCD码的DWORD转换为二进制码的DWORD。




```
BCD_TO_DWORD(X:= )
```

BCD_TO_INT

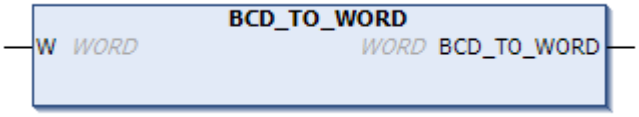
将一个BCD码字节转换为一个INT值。




```
BCD_TO_INT(B:= )
```

BCD_TO_WORD

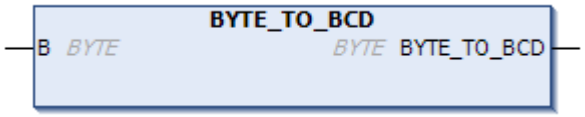
将一个BCD码的WORD转换为一个二进制码WORD。




```
BCD_TO_WORD(W:= )
```

BYTE_TO_BCD

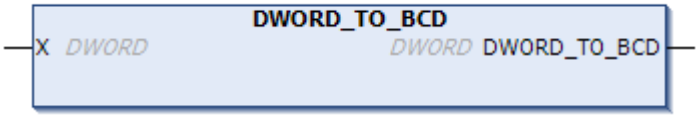
将一个二进制码字节转换为BCD码字节。




```
BYTE_TO_BCD(B:= )
```

DWORD_TO_BCD

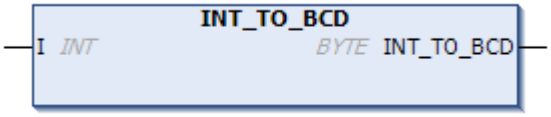
将一个二进制码的DWORD转换为BCD码的DWORD。




```
DWORD_TO_BCD(X:= )
```

INT_TO_BCD

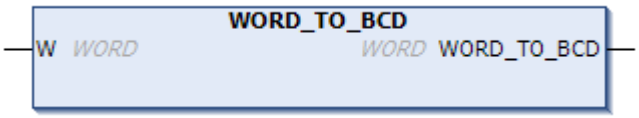
将一个INT值转换为一个BCD码字节。




```
INT_TO_BCD(I:= )
```

WORD_TO_BCD

将一个二进制码WORD转换为BCD码的WORD。

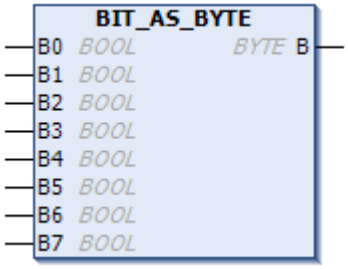



```
WORD_TO_BCD(W:= )
```

19.2.3 Bit/Byte Functions

BIT_AS_BYTE

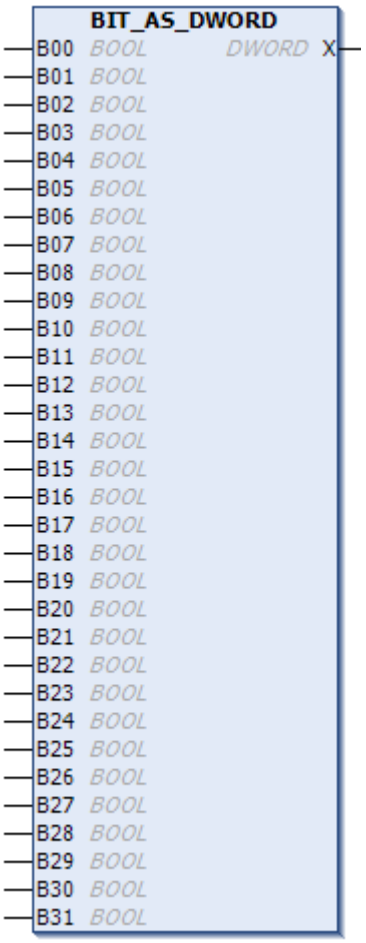
将8个BOOL类型输入值转换为BYTE类型的输出值。




```
BIT_AS_BYTE( B0:= ,  B1:= ,  B2:= ,  B3:= ,  B4:= ,  B5:= ,  B6:= ,  B7:= ,  B=> );
```

BIT_AS_DWORD

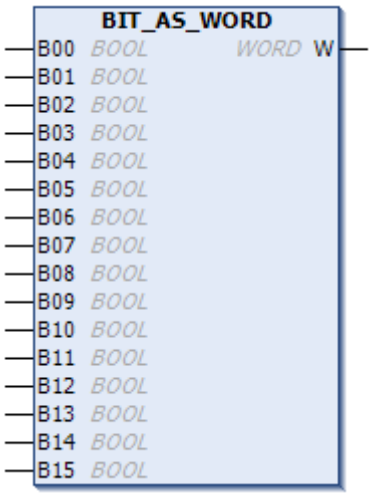
将32个BOOL类型的输入值转换为一个DWORD类型的输出值。




```
   BIT_AS_DWORD( B00:= ,  B01:= ,  B02:= ,  B03:= ,  B04:= ,  B05:= ,  B06:= ,  B07:= ,  B08:= ,  B09:= ,
B10:= ,  B11:= ,  B12:= ,  B13:= ,  B14:= ,  B15:= ,  B16:= ,  B17:= ,  B18:= ,  B19:= ,  B20:= ,  B21:= ,
B22:= ,  B23:= ,  B24:= ,  B25:= ,  B26:= ,  B27:= ,  B28:= ,  B29:= ,  B30:= ,  B31:= ,  X=> );
```

BIT_AS_WORD

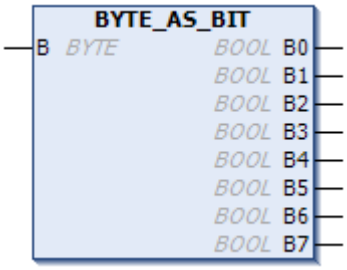
将16个BOOL类型的输入值转换为WORD类型的输出值。



```
    BIT_AS_WORD( B00:= ,  B01:= ,  B02:= ,  B03:= ,  B04:= ,  B05:= ,  B06:= ,  B07:= ,  B08:= ,  B09:= ,
B10:= ,  B11:= ,  B12:= ,  B13:= ,  B14:= ,  B15:= ,  W=> );
```

BYTE_AS_BIT

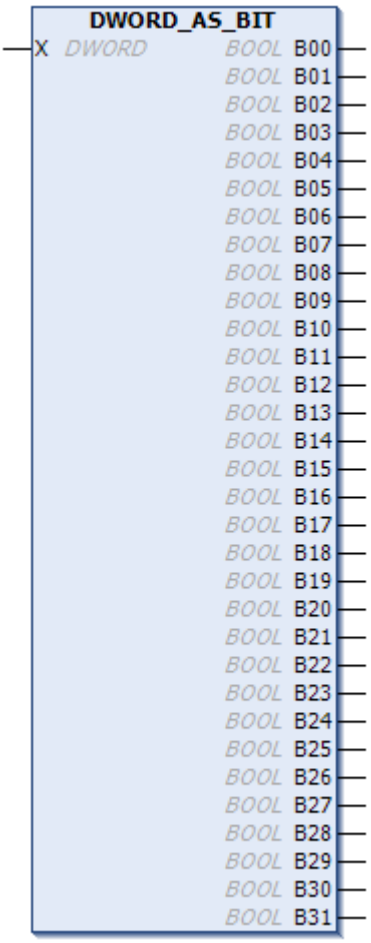
将一个BYTE类型的输入值转换为8个BOOL类型的输出值。




```
    BYTE_AS_BIT( B:= ,  B0:= ,  B1:= ,  B2:= ,  B3:= ,  B4:= ,  B5:= ,  B6:= ,  B7=> );
```

DWORD_AS_BIT

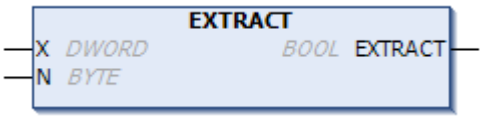
将一个DWORD类型的输入值转换为32个BOOL类型的输出值。



```
DWORD_AS_BIT( X:= , B00=> , B01=> , B02=> , B03=> , B04=> , B05=> , B06=> , B07=> , B08=> ,
B09=> , B10=> , B11=> , B12=> , B13=> , B14=> , B15=> , B16=> , B17=> , B18=> , B19=> , B20=> ,
B21=> , B22=> , B23=> , B24=> , B25=> , B26=> , B27=> , B28=> , B29=> , B30=> , B31=> );
```

EXTRACT

返回输入值X的BIT位数。




```
EXTRACT(X:= , N:= )
```

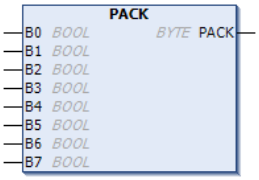
GETBIT

返回值X在位置N处的值。


```
GETBIT(X:= , N:= )
```

PACK

将8位BOOL值打包成一个BYTE字节。




```
PACK( B0:= , B1:= , B2:= , B3:= , B4:= , B5:= , B6:= , B7:= )
```

PUTBIT

设置DWORD值的一个位。


```
PUTBIT(X:= , N:= , B:= )
```

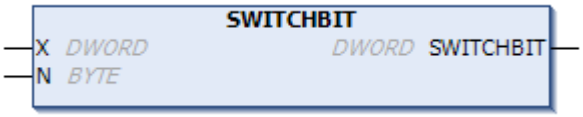
SETBIT

设置变量X的位置N位的值为B。


```
SETBIT(X:= , N:= , B:= )
```

SWITCHBIT

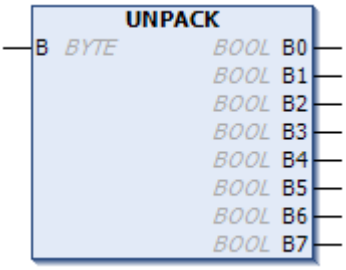
替换DWORD值在位置N处的一个位。




```
SWITCHBIT(X:= , N:= )
```

UNPACK

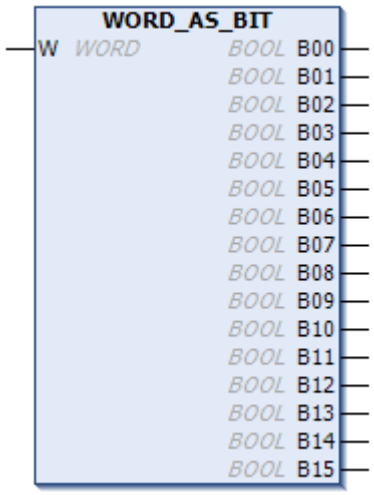
将一个字节转换为8个BIT位。




```
UNPACK(B:= , B0:= , B1:= , B2:= , B3:= , B4:= , B5:= , B6:= , B7:= )
```

WORD_AS_BIT

将一个WORD类型的输入值转换为16个BOOL类型的输出值。

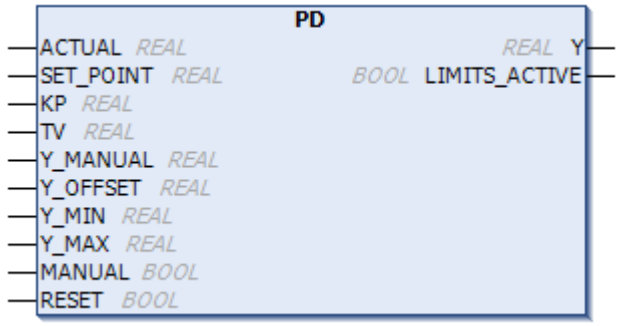



```
   WORD_AS_BIT( W:= ,  B00=> ,  B01=> ,  B02=> ,  B03=> ,  B04=> ,  B05=> ,  B06=> ,  B07=> ,  B08=> ,
B09=> ,  B10=> ,  B11=> ,  B12=> ,  B13=> ,  B14=> ,  B15=> );
```

19.2.4 Controller

PD

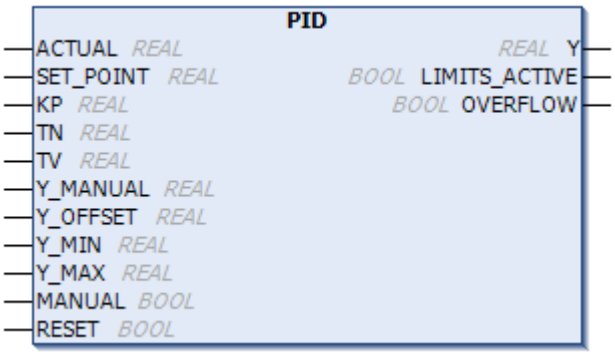
实现PD控制。




```
   PD( ACTUAL:= ,  SET_POINT:= ,  KP:= ,  TV:= ,  Y_MANUAL:= ,  Y_OFFSET:= ,  Y_MIN:= ,  Y_MAX:= ,
MANUAL:= ,  RESET:= ,  Y=> ,  LIMITS_ACTIVE=> );
```

PID

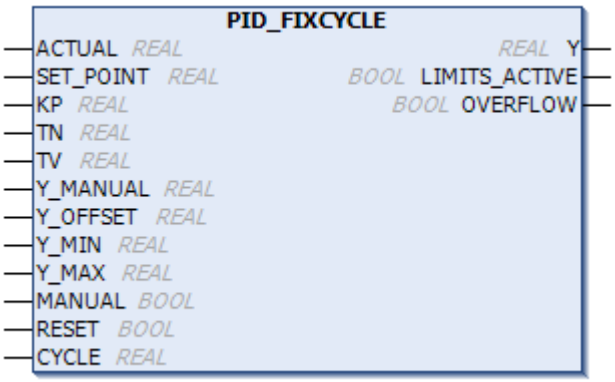
实现PID控制。




```
PID( ACTUAL:= , SET_POINT:= , KP:= , TN:= , TV:= , Y_MANUAL:= , Y_OFFSET:= , Y_MIN:= , Y_MAX:= ,
MANUAL:= , RESET:= , Y=> , LIMITS_ACTIVE=> , OVERFLOW=> );
```

PID_FIXCYCLE

可以手动设置周期时间的PID控制。

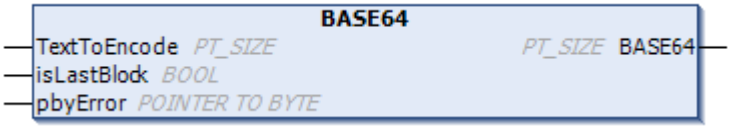



```
PID_FIXCYCLE( ACTUAL:= , SET_POINT:= , KP:= , TN:= , TV:= , Y_MANUAL:= , Y_OFFSET:= , Y_MIN:= ,
Y_MAX:= , MANUAL:= , RESET:= , CYCLE:= , Y=> , LIMITS_ACTIVE=> , OVERFLOW=> );
```

19.2.5 Encoding

BASE64

将8位二进制数据编码为ASCII数据。

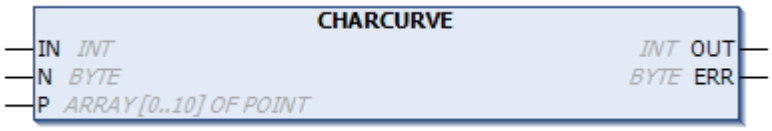



```
BASE64(TextToEncode:= , isLastBlock:= , pbyError:= )
```

19.2.6 Function Manipulators

CHARCURVE

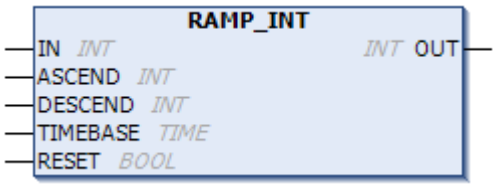
将输入信号映射到特征曲线上。




```
CHARCURVE( IN:= , N:= , P:= , OUT=> , ERR=> );
```

RAMP_INT

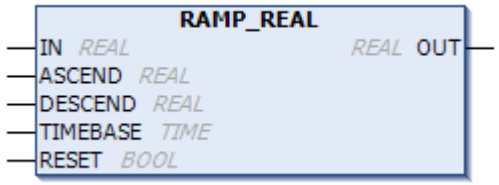
将某一个INT类型值的斜率限制到一个确定的值，以INT类型输出。




```
RAMP_INT( IN:= , ASCEND:= , DESCEND:= , TIMEBASE:= , RESET:= , OUT=> );
```

RAMP_REAL

将某一个REAL类型值的斜率限制到一个确定的值，以REAL类型输出。

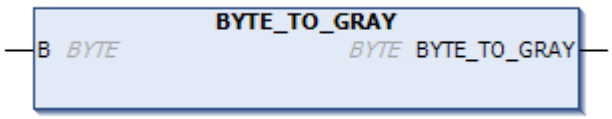



```
RAMP_REAL( IN:= , ASCEND:= , DESCEND:= , TIMEBASE:= , RESET:= , OUT=> );
```

19.2.7 Gray Conversions

BYTE_TO_GRAY

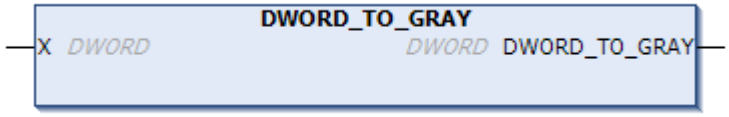
将一个二进制类型的字节数据转换为GRAY码值。




```
BYTE_TO_GRAY(B:= )
```

DWORD_TO_GRAY

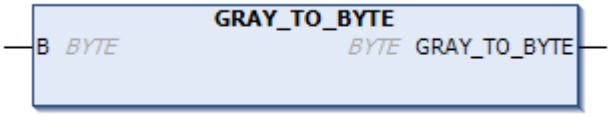
将一个二进制码的DWORD值转换为GRAY编码类型的DWORD值。




```
DWORD_TO_GRAY(X:= )
```

GRAY_TO_BYTE

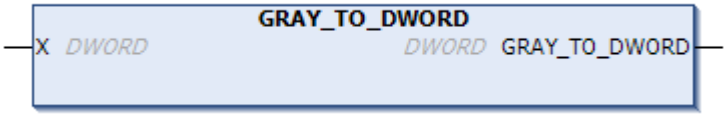
将一个GRAY码字节数据转换为一个二进制码字节数据。




```
GRAY_TO_BYTE(B:= )
```

GRAY_TO_DWORD

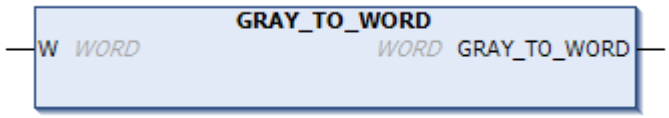
将一个GRAY码的DWORD数据转换为二进制码类型的DWORD。




```
GRAY_TO_DWORD(X:= )
```

GRAY_TO_WORD

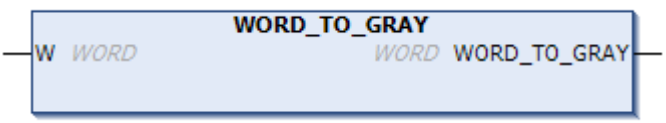
将一个GRAY码的WORD数据转换为二进制码类型的WORD。




```
GRAY_TO_WORD(W:= )
```

WORD_TO_GRAY

将一个二进制码的WORD值转换为GRAY编码类型的WORD值。

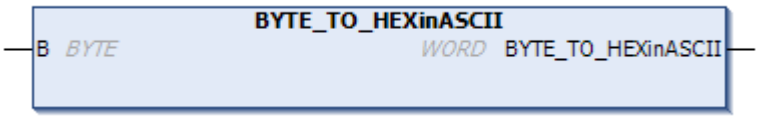



```
WORD_TO_GRAY(W:= )
```

19.2.8 HEX/ASCII Functions

BYTE_TO_HEXinASCII

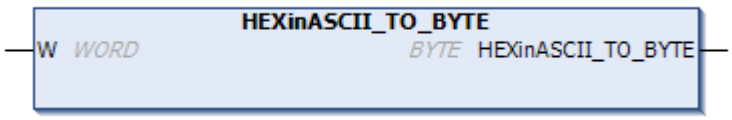
将二进制代码的一个字节转换为用HEX码表示的ACSII值。




```
BYTE_TO_HEXinASCII(B:=)
```

HEXinASCII_TO_BYTE

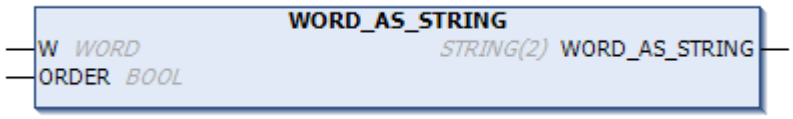
将一个用HEX码表示ACSII值的一个WORD值转换为用二进制码表示的一个字节。




```
HEXinASCII_TO_BYTE(W:= )
```

WORD_AS_STRING

将一个WORD值转换为ASCII字符串表示形式。




```
WORD_AS_STRING(W:= , ORDER:= )
```

19.2.9 Util库

本节说明Util库所包含的基本指令。

19.2.10 Library Information

GetLibVersion

用于获得库版本信息:VERSION。

```
GetLibVersion()
```

GetLibVersionNumber

用于获得库版本号。


```
GetLibVersionNumber()
```

IsLibReleased

判断当前库是否发布。


```
IsLibReleased()
```

19.2.11 Mathematical Functions

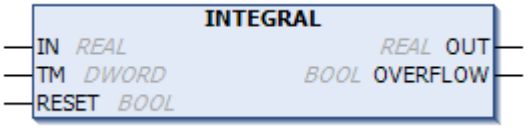
DERIVATIVE

导数。


```
DERIVATIVE(IN:= , TM:= , RESET:= , OUT=> );
```

INTEGRAL

积分。




```
INTEGRAL( IN:= ,  TM:= ,  RESET:= ,  OUT=> ,  OVERFLOW=> );
```

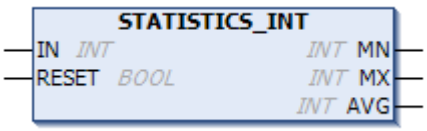
LIN_TRAFO

线性变换。


```
LIN_TRAFO( IN:= , IN_MIN:= , IN_MAX:= , OUT_MIN:= , OUT_MAX:= , OUT=> , ERROR=> );
```

STATISTICS_INT

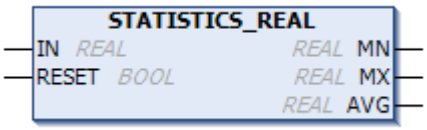
计算INT类型输入值的最小值，最大值和平均值。




```
STATISTICS_INT( IN:= , RESET:= , MN=> , MX=> , AVG=> );
```

STATISTICS_REAL

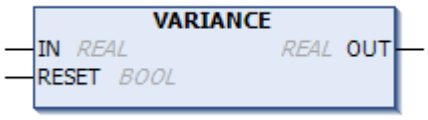
计算REAL类型输入值的最小值，最大值和平均值。




```
STATISTICS_REAL( IN:= , RESET:= , MN=> , MX=> , AVG=> );
```

VARIANCE

方差。




```
VARIANCE(IN:= , RESET:= , OUT=> );
```

19.2.12 Signals

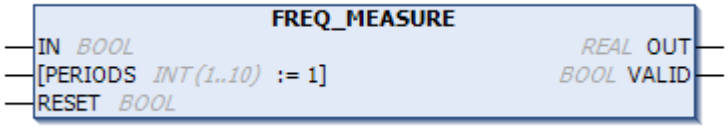
BLINK

模拟一个电平翻转信号。


```
BLINK(ENABLE:= , TIMELOW:= , TIMEHIGH:= , OUT=> );
```

FREQ_MEASURE

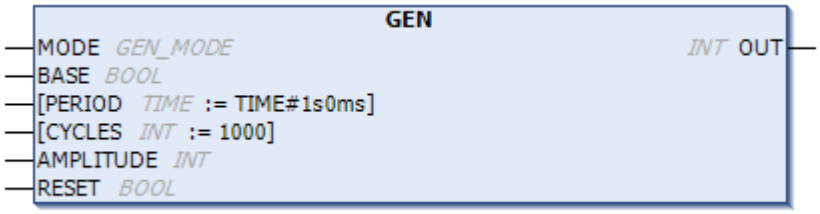
测量输入信号的频率。




```
FREQ_MEASURE( IN:= , PERIODS:= , RESET:= , OUT=> , VALID=> );
```

GEN

产生不同给定类型的周期函数信号。

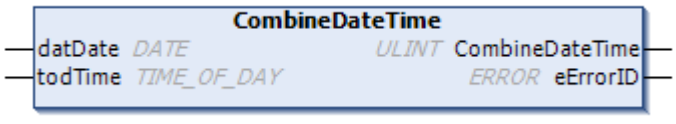



```
GEN( MODE:= , BASE:= , PERIOD:= , CYCLES:= , AMPLITUDE:= , RESET:= , OUT=> );
```

19.2.13 Util_TimerSwitch

CombineDateTime

将IEC数据类型部分组合成时间戳。




```
CombineDateTime(datDate:= , todTime:= , eErrorID=> )
```

DateTimeFromWeek

将ISO星期日期部分组合成为时间戳。


```
DateTimeFromWeek(uiYear:= , uiWeek:= , eWeekday:= , eErrorID=> )
```

DayOfWeek

计算与数据日期参数匹配的WEEKDAY枚举类型的近似值。


```
DayOfWeek(datDate:= , eErrorID=> )
```

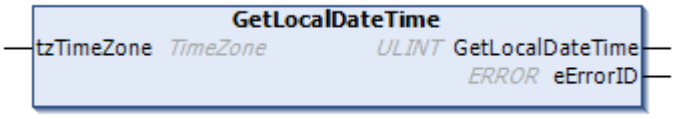
GetDateTime

获得以毫秒为单位的当前日期和时间。


```
GetDateTime(eErrorID=> )
```

GetLocalDateTime

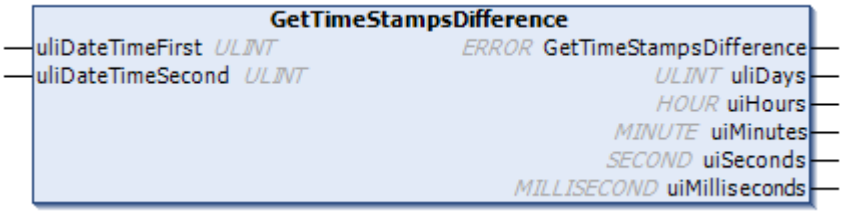
获得以毫秒为单位的当前日期和时间。




```
GetLocalDateTime(tzTimeZone:= , eErrorID=> )
```

GetTimeStampsDifference

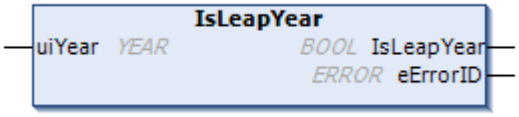
获取时间戳差异。




```
   GetTimeStampsDifference(uliDateTimeFirst:=, uliDateTimeSecond:= , uliDays=> , uiHours=> , uiMinutes=> ,
uiSeconds=> , uiMilliseconds=> )
```

IsLeapYear

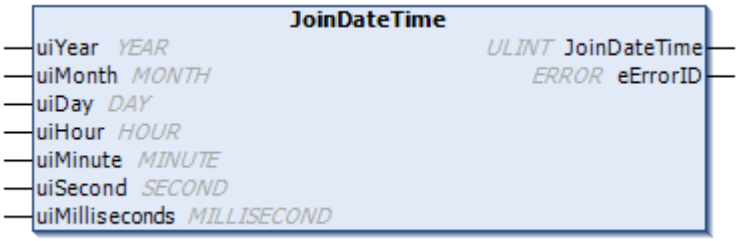
判断这个年是否是闰年。




```
    IsLeapYear(uiYear:= , eErrorID=> )
```

JoinDateTime

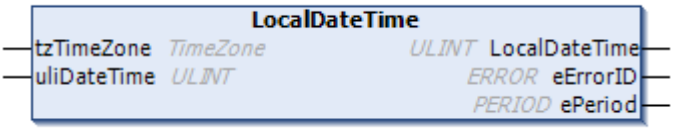
返回以毫秒为单位的时间戳。




```
   JoinDateTime( uiYear:= , uiMonth:= , uiDay:= , uiHour:= , uiMinute:= , uiSecond:= , uiMilliseconds:= ,
eErrorID=> )
```

LocalDateTime

根据输入的uliDataTime和tzTimeZone计算出相应的本地时间。




```
   LocalDateTime(tzTimeZone:= , uliDateTime:= , eErrorID=> , ePeriod=> )
```

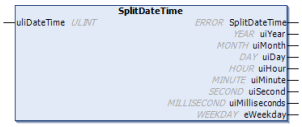
SeparateDateTime

在其IEC数据类型部分中分离时间戳。


```
SeparateDateTime(uliDateTime:= , eWeekDay=> , datDate=> , todTime=> )
```

SplitDateTime

将时间戳拆分为时间点。




```
SplitDateTime(uliDateTime:= , uiYear=> , uiMonth=> , uiDay=> , uiHour=> , uiMinute=> , uiSecond=> ,
uiMilliseconds=> , eWeekday=> )
```

WeekOfYear

计算与uliDateTime参数匹配的ISO星期日期部分的近似值。


```
WeekOfYear(uliDateTime:= , uiYear=> , uiWeek=>, eWeekday=> )
```

20 配套模块及模块参数一览

本节说明系统可配套的模块，以及各模块参数。

20.1 配套模块列表

表 20-1 系统配套模块一览表

序号	型号	名称	说明
1	DI3216-S	开关量信号输入模块	16 通道，支持有源或无源触点信号输入，统一隔离
2	DO3216-S	开关量信号输出模块	16 通道，支持晶体管输出，统一隔离
3	DO3216PNP-S	开关量信号输出模块	16通道，支持晶体管源型输出（PNP），统一隔离
4	DO3208RLY-S	开关量信号输出模块	8通道，支持常开干触点信号输出，点点隔离
5	AI3208-S	模拟量信号输入模块	8通道，支持（0～10）mA、（4～20）mA，支持配电/非配电信号类型，统一隔离
6	AO3208-S	模拟量信号输出模块	8通道，支持（0～10）mA、（4～20）mA、（0～20）mA，统一隔离
7	AI3208HS-S	高速模拟量信号输入模块	8通道，支持（0～10）mA、（4～20）mA，支持配电/非配电信号类型，统一隔离，3ms刷新周期
8	AI3204IV-S	模拟量输入模块（电压电流混合）	4通道，支持采集标准电流信号（4～20）mA、（0～20）mA，电压信号（-10～10）V、（0～5）V、（1～5）V
9	AO3204IV-S	模拟量输出模块（电压电流混合）	4通道，可输出量程为（0~5）V、（1~5）V、（-10~10）V，电压信号或（0~20）mA、（4~20）mA的电流信号
10	AI3206RTD-S	热电阻信号输入模块	6通道，支持（1~400）Ω、（2~1000）Ω、Pt100、Pt1000、Cu50信号类型，支持2线制/3线制接线方式，点点隔离
11	PI3204-S	高速计数器模块	4通道，支持100KHz脉冲信号测量，统一隔离
12	AM3201HSC-S	增量编码器采集模块	1通道增量编码器输入接口，最高200kHz，支持5V差分，24V单端
13	AM3201SSI-S	绝对值编码器采集模块	1通道绝对值编码器输入，SSI接口
14	AM3200-S	空模块	起防尘作用
15	COM3204RTU-S	串行通讯模块	4 路RS485接口，支持Modbus RTU主从站协议

表 **20-1** 系统配套模块一览表（续）

序号	型号	名称	说明
18	PW3203AC-S	AC-DC电源模块	220V AC输入，24V DC输出，支持24V/3A输出功率时3秒UPS能力，机架安装
19	CN3220/14/10/06-S	机架（20/14/10/6槽）	最多可安装20/14/10/6个模块，可作为本地机架或远程机架

20.2 配套模块参数说明

本节对系统配套模块的参数进行说明。

20.2.1 IM3202PN-S模块

配置参数

配置参数	参数	说明
slotn	数据下发周期	10ms~60000ms
	数据变化下发使能	状态改变立即下发数据

位号说明

变量	偏移地址	说明
Inputs PS	0x00	PN网络侧通信正常时为0x80，异常时为0x00
Inputs PS	0x01	预留
Inputs PS	0x02	预留

20.2.2 DI3216-S模块

配置参数

配置参数	参数	说明
所有通道	抖动使能	开启：抖动使能开启后，当每秒抖动次数大于“抖动参数”设置值时，通道质量码的状态为数据可疑（0x40） 关闭：关闭抖动功能
实时参数配置	数据上报周期	10ms~60000ms

配置参数	参数	说明
	数据上报模式	周期发布：按设置的数据上报周期，周期上报数据 周期发布+事件发布：除了周期上报数据外，输入状态改变立即上报
Channel_n	通道开关	开始：通道开启 关闭：通道关闭
	锁存功能	开启：锁存功能开启后，当通道输入状态变化后，保持当前通道状态，锁存时间到之后，通道状态才随着输入的状态而改变 关闭：关闭锁存功能
	锁存时间	可设置为150ms、300ms、600ms、1200ms
	抖动参数（每秒抖动次数）	1~255，每秒抖动次数
	滤波时间（单位：ms）	滤波时间，可设置为关闭、4ms、8ms、16ms、32ms滤波

位号说明

变量	参数	说明
Inputs	诊断字BYTE0	Bit2：轻故障标志位，轻故障时该位置为TRUE Bit3：重故障标志位，重故障时该位置为TRUE
	诊断字BYTE1	预留
	诊断字BYTE2	Bit1：组态校验故障，故障时该位置为TRUE
	诊断字BYTE3	Bit2：内部电路故障，故障时该位置为TRUE
	诊断字BYTE4	预留
	诊断字BYTE5	Bit0：奇偶槽位，奇数槽位为TRUE
	DEVICE_ID	模块ID
	输入值CH00n	通道n的输入值
	质量码CH00n_D	通道n的质量码。 0x00：通道开启 0x9F：通道关闭 0x40：数据可疑
Inputs PS	模块背板通信诊断	通信正常时为0x80，异常时为0x00

20.2.3 DO3216-S/DO3216PNP-S模块

配置参数

配置参数	参数	说明
实时参数配置	数据上报周期	10ms~60000ms
	数据上报模式	周期发布：按设置的数据上报周期，周期上报数据 周期发布+事件发布：除了周期上报数据外，输入状态改变立即上报
	接收超时时间	1000ms~60000ms，超时没有收到Q区数据，进入故障安全模式
Channel_n	通道开关	开始：通道开启 关闭：通道关闭
	故障安全模式	输出保持：进入故障安全模式后，保持当前输出值 按预设值输出：进入故障安全模式后，输出设置的故障输出值
	故障输出值	故障输出值，进入故障安全模式，且故障安全模式设置为按预设值输出时，按此参数设置输出
	输出类型	状态输出：通道输出ON/OFF状态 脉冲输出：按照设置的脉宽和脉冲输出个数，输出脉冲，脉宽输出通过通道输出值上升沿触发
	脉宽	1ms~600000ms，当输出类型为“脉冲输出”有效
	脉冲输出个数	1~255，当输出类型为“脉冲输出”有效

位号说明

变量	参数	说明
Inputs	诊断字BYTE0	Bit2：轻故障标志位，轻故障时该位置为TRUE Bit3：重故障标志位，重故障时该位置为TRUE
	诊断字BYTE1	预留
	诊断字BYTE2	Bit1：组态校验故障，故障时该位置为TRUE
	诊断字BYTE3	Bit2：内部电路故障，故障时该位置为TRUE Bit5：输出电压故障，故障时该位置为TRUE
	诊断字BYTE4	预留
	诊断字BYTE5	Bit0：奇偶槽位，奇数槽位为TRUE
	DEVICE_ID	模块ID
	输入值CH00n	模块收到通道输出值后，将通道n的输出值赋值给通道n的输入值

变量	参数	说明
	质量码CH00n_D	通道n的质量码 0x00：通道开启 0x9F：通道关闭 0x8C：输出电路故障
Inputs PS	模块背板通信诊断	通信正常时为0x80，异常时为0x00
Outputs	DEVICE ID	预留
	输出值CH00n	通道输出值
Outputs CS	输出数据诊断	当值为0x80时，输出数据有效

20.2.4 AI3208-S模块

配置参数

配置参数	参数	说明
所有通道	采样	正常采样：采样周期500ms，抗工频 快速采样：采样周期50ms 本参数对本模块所有通道有效
实时参数配置	数据上报周期	10ms~60000ms
Channel_n (n=1~8)	信号类型	表示通道n采集的信号类型，可选类型如下 电流信号(4~20)mA 电流信号(0~10)mA
	通道开关	开始：通道开启 关闭：通道关闭
	配电状况	配电：信号回路电源由内部提供 外部配电：信号回路电源由外部设备提供
	信号有效性检查	开启有效性检查：相邻两次采集的数据变化率超过一定的范围时，通道质量码报信号可疑 关闭：关闭有效性检查功能
	量程上限	被测信号的量程上限，小于20mA
	量程下限	被测信号的量程下限，大于0mA

位号说明

变量	参数	说明
Inputs	诊断字BYTE0	BIT2：轻故障，轻故障时该位置为TRUE BIT3：重故障，重故障时该位置为TRUE
	诊断字BYTE1	预留
	诊断字BYTE2	BIT0：标定数据故障, 故障时该位置为TRUE BIT1：组态校验故障, 故障时该位置为TRUE
	诊断字BYTE3	BIT2：内部电源故障，故障时该位置为TRUE BIT6：ADC故障，故障时该位置为TRUE BIT7：VREF故障，故障时该位置为TRUE
	诊断字BYTE4	预留
	诊断字BYTE5	Bit0：奇偶槽位，奇数槽位为TRUE
	DEVICE_ID	表示该模块ID
	输入值CH00n (n = 1~8)	表示通道n的输入值
	质量码CH00n_D (n = 1~8)	表示通道n的质量码。 0x9F：表示通道关闭 0x00：表示通道开启 0x40：表示通道信号可疑 0x92：表示通道超量程 0x88：表示通道断线
Inputs PS	模块背板通信诊断	通信正常时为0x80，异常时为0x00

20.2.5 AO3208-S模块

配置参数

配置参数	参数	说明
实时参数配置	数据上报周期	10ms~60000ms
	接收超时时间	1000ms~60000ms，超时没有收到Q区数据，进入故障安全模式
Channel_n	信号类型	电流信号（4～20）mA、（0～20）mA、（0~10）mA
	通道开关	开始：通道开启 关闭：通道关闭
	故障安全模式	输出保持：进入故障安全模式后，保持当前输出值 按预设值输出：进入故障安全模式后，输出设置的故障输出值
	量程上限	自由量程的上限值，设置值应在所选的量程内，单位mA

	量程下限	自由量程的下限值，设置值应在所选的量程内，单位mA
	故障输出值	故障输出值，进入故障安全模式，且故障安全模式设置为按预设值输出时，按此参数设置输出

位号说明

变量	参数	说明
Inputs	诊断字BYTE0	Bit2：轻故障标志位，轻故障时该位置为TRUE; Bit3：重故障标志位，重故障时该位置为TRUE。
	诊断字BYTE1	预留
	诊断字BYTE2	Bit0：标定数据故障，故障时该位置为TRUE Bit1：组态校验故障，故障时该位置为TRUE
	诊断字BYTE3	Bit2：内部电路故障，故障时该位置为TRUE Bit3：从CPU故障，故障时该位置为TRUE Bit6：ADC故障，故障时该位置为TRUE Bit7：VREF故障，故障时该位置为TRUE
	诊断字BYTE4	预留
	诊断字BYTE5	Bit0：奇偶槽位，奇数槽位为TRUE
	DEVICE_ID	模块ID
	输入值CH00n	模块收到通道输出值后，将通道n的输出值赋值给通道n的输入值
	质量码CH00n_D	通道n的质量码。 0x00：通道开启； 0x9F：通道关闭； 0x88：断线； 0x8C：输出电路故障； 0x92：超量程。
Inputs PS	模块背板通信诊断	通信正常时为0x80，异常时为0x00
Outputs	DEVICE ID	预留
	输出值CH00n	通道输出值
Outputs CS	输出数据诊断	当值为0x80时，输出数据有效

20.2.6 AI3208HS-S模块

配置参数

配置区	参数	说明
所有通道	采样周期	本参数对本模块所有通道有效。 正常采样：采样周期为20ms 快速采样：采样周期为3ms
实时参数配置	数据上报周期	10ms~60000ms
Channel_n (n = 1~8)	信号类型	表示通道n采集的信号类型，可选类型如下 ·电流信号(4~20)mA ·电流信号(0~10)mA
	通道开关	开始：通道开启 关闭：通道关闭
	配电状况	配电：信号回路电源由内部提供 外部配电：信号回路电源由外部设备提供
	信号有效性检查	开启有效性检查：相邻两次采集的数据变化率超过一定的范围时，报信号可疑 关闭：关闭有效性检查功能
	量程上限	被测信号的量程上限，小于20mA

位号说明

变量	参数	说明
Inputs	诊断字BYTE0	BIT2：轻故障，轻故障时该位置为TRUE BIT3：重故障，重故障时该位置为TRUE
	诊断字BYTE1	预留
	诊断字BYTE2	BIT0：标定数据故障, 故障时该位置为TRUE BIT1：组态校验故障, 故障时该位置为TRUE
	诊断字BYTE3	BIT2：内部电路故障，故障时该位置为TRUE BIT6：ADC故障，故障时该位置为TRUE BIT7：VREF故障，故障时该位置为TRUE
	诊断字BYTE4	预留
	诊断字BYTE5	Bit0：奇偶槽位，奇数槽位为TRUE
	DEVICE_ID	表示该模块ID
	输入值CH00n	表示通道n的输入值

变量	参数	说明
	质量码CH00n_D	表示通道n的质量码。 0x9F：表示通道关闭 0x00：表示通道正常 0x40：表示通道信号可疑 0x92：表示通道超量程 0x88：表示通道断线
Inputs PS	模块背板通信诊断	通信正常时为0x80，异常时为0x00

20.2.7 AI3206RTD-S模块

配置参数

配置参数	参数	说明
实时参数配置	数据上报周期	10ms~60000ms
Channel_n	信号类型	（1~400）Ω、（2~1000）Ω Pt100（-200℃～850℃）、Pt1000（-50℃～300℃）、Cu50（-50℃～150℃）
	通道开关	开始：通道开启 关闭：通道关闭
	量程上限	自由量程的上限值，设置值应在所选的量程内，单位和量程单位一致
	量程下限	自由量程的下限值，设置值应在所选的量程内，单位和量程单位一致

位号说明

变量	参数	说明
Inputs	诊断字BYTE0	Bit2：轻故障标志位，轻故障时该位置为TRUE Bit3：重故障标志位，重故障时该位置为TRUE
	诊断字BYTE1	预留
	诊断字BYTE2	Bit0：标定数据故障，故障时该位置为TRUE Bit1：组态校验故障，故障时该位置为TRUE
	诊断字BYTE3	Bit2：内部电路故障，故障时该位置为TRUE Bit4：R250故障，故障时该位置为TRUE Bit6：ADC故障，故障时该位置为TRUE Bit7：VREF故障，故障时该位置为TRUE
	诊断字BYTE4	预留

变量	参数	说明
	诊断字BYTE5	Bit0：奇偶槽位，奇数槽位为TRUE
	DEVICE_ID	模块ID
	输入值CH00n	通道输入值
	质量码CH00n_D	通道n的质量码。 0x00：通道开启 0x40：数据可疑 0x9F：通道关闭 0x88：断线 0x90：短路 0x92：超量程
Inputs PS	模块背板通信诊断	通信正常时为0x80，异常时为0x00

20.2.8 DO3208RLY-S模块

配置参数

配置参数	参数	说明
实时参数配置	数据上报周期	10ms~60000ms
	数据上报模式	周期发布：按设置的数据上报周期，周期上报数据 周期发布+事件发布：除了周期上报数据外，输入状态改变立即上报
	接收超时时间	1000ms~60000ms，超时没有收到Q区数据，进入故障安全模式
Channel_n (n=1~8)	通道开关	开始：通道开启 关闭：通道关闭
	故障安全模式	输出保持：进入故障安全模式后，保持当前输出值 按预设值输出：进入故障安全模式后，输出设置的故障输出值
	故障输出值	故障输出值，进入故障安全模式，且故障安全模式设置为按预设值输出时，按此参数设置输出
	输出类型	状态输出：通道输出ON/OFF状态

诊断位号

变量	参数	说明
Inputs	诊断字BYTE0	Bit2：轻故障标志位，轻故障时该位置为TRUE Bit3：重故障标志位，重故障时该位置为TRUE
	诊断字BYTE1	预留
	诊断字BYTE2	Bit1：组态校验故障，故障时该位置为TRUE
	诊断字BYTE3	Bit2：内部电路故障，故障时该位置为TRUE
	诊断字BYTE4	预留
	诊断字BYTE5	Bit0：奇偶槽位，奇数槽位为TRUE
	DEVICE_ID	模块ID
	输入值CH00n (n=1~8)	表示当前通道输出值
	质量码CH00n_D (n=1~8)	通道n的质量码。 0x00：通道开启 0x9F：通道关闭 0x8C：输出电路故障
Inputs PS	模块背板通信诊断	通信正常时为0x80，异常时为0x00
Outputs	DEVICE_ID	表示该模块ID
	输出值CH00n (n=1~8)	通道输出值
Outputs CS	输出数据诊断	当值为0x80时，输出数据有效

20.2.9 PI3204-S模块

配置参数

配置参数	参数	说明
实时参数配置	数据上报周期	10ms~60000ms
	采样	正常采样：50ms 快速采样：10ms
Channel_n	信号类型	（0~1000）Hz、（0~10000）Hz、（0~100000）Hz
	通道开关	开始：通道开启 关闭：通道关闭
	量程上限	自由量程的上限值，设置值应在所选的量程内，单位Hz

位号说明

变量	参数	说明
Inputs	诊断字BYTE0	Bit2：轻故障标志位，轻故障时该位置为TRUE Bit3：重故障标志位，重故障时该位置为TRUE
	诊断字BYTE1	预留
	诊断字BYTE2	Bit1：组态校验故障，故障时该位置为TRUE
	诊断字BYTE3	Bit2：内部电路故障，故障时该位置为TRUE
	诊断字BYTE4	预留
	诊断字BYTE5	Bit0：奇偶槽位，奇数槽位为TRUE
	DEVICE_ID	模块ID
	输入值CH00n(频率)	通道输入频率值
	质量码CH00n_D	通道n的质量码。 0x00：通道开启 0x9F：通道关闭 0x92：超量程
	输入值CH00n(脉冲累计)	通道输入脉冲累计值
Inputs PS	模块背板通信诊断	通信正常时为0x80，异常时为0x00

20.2.10 AM3201HSC-S模块

配置参数

配置区	参数	说明
实时数据参数	数据上报周期	10ms~60000ms
	接收超时时间	1000ms~60000ms，超时没有收到Q区数据，进入故障安全模式
channel_1	通道开关	开启：通道启用 关闭：通道禁用
	配电状况	配电：通道使用辅助电源供电 外部配电：通道使用外部电源供电
	计数模式	连续计数：在下溢值到上溢值之间计数 周期计数：当计数溢出时，重新开始从预设值

配置区	参数	说明
	输入脉冲类型	5V差分AB相正交脉冲
	计数值的置值方式	内部直接置值：在内部软件置值使能的上升沿时刻将计数值置为0
	计数使能方式	内部使能：通过软件进行开启、关闭计数器 默认使能（不通过软件开启）
	平滑选项（ms）	平滑窗口时间
channel_2	通道开关	开启：通道启用 关闭：通道禁用
	计数模式	连续计数：在下溢值到上溢值之间计数 周期计数：当计数溢出时，重新开始从预设值
	输入脉冲类型	1：24V端AB正交脉冲 2：24V方向脉冲 3：24V CW+CCW脉冲
	计数值的置值方式	内部直接置值：在内部软件置值使能的上升沿时刻将计数值置为0
	计数使能方式	内部使能：通过软件进行开启、关闭计数器 默认使能（不通过软件开启）
	平滑选项（ms）	平滑窗口时间

位号说明

变量	参数	说明
Inputs	诊断字BYTE0	bit2：轻故障 bit3：重故障
	诊断字BYTE1	-
	诊断字BYTE2	bit1：组态校验故障
	诊断字BYTE3	bit2：内部电路故障
	诊断字BYTE4	-
	诊断字BYTE5	bit0：奇偶槽位
	DEVICE_ID	模块ID
	通道1模式	0：5V差分AB相正交脉冲

变量	参数	说明
	通道1方向	0：正转 1：反转
	通道1 IO	bit0：DI1通道实时值 bit1：DI2通道实时值
	通道1正转计数	通道1正转计数，正转时递增，反转时递减
	通道1频率值	原始值x100后上送
	通道2模式	当前通道工作模式： 1：24V端AB正交脉冲 2：24V方向脉冲 3：24V CW+CCW脉冲
	通道2方向	0：正转 1：反转
	通道2 IO	bit0：DI3通道实时值 bit1：DI4通道实时值
	通道2正转计数	通道2正转计数，正转时递增，反转时递减
	通道2反转计数	仅在配置为“24V方向脉冲”和“24V CW+CCW脉冲”时有效
	通道2频率值	原始值x100后上送
Inputs PS	模块背板通信诊断	通信正常时为0x80，异常时为0x00
Outputs	DEVICE_ID	表示该模块ID
	通道1控制	bit0：通道1开启停止计数器，0停止，1开启 bit1：通道1边沿触发功能
	通道2控制	bit0：通道2开启停止计数器，0停止，1开启 bit1：通道2边沿触发功能

20.2.11 AM3201SSI-S模块

配置参数

配置区	参数	说明
实时数据参数	数据上报周期	10ms~60000ms
	接收超时时间	1000ms~60000ms，超时没有收到Q区数据，进入故障安全模式
	抖动使能	用于设置所有DI通道的抖动功能开关，默认开启

配置区	参数	说明
Channel_1	信号类型	SSI默认值
	通道开关	开始：通道开启 关闭：通道关闭
	时钟极性配置	用于设置时钟电平信号，默认为高电平
	传输速度	用于设置读取绝对值编码器数据的传输速度，可设置100KHz，125KHz，250KHz，500KHz，1MHz，默认为125KHz
	数据编码类型	用于设置绝对值编码器的数据编码类型，可分格雷码类型与自然二进制码类型，默认为格雷码
	单圈起始位	用于设置单圈数据在编码器数据帧结构中的哪一位开始（0~32）
	单圈位数	用于设置编码器的单圈位数（0~32）
	多圈起始位	用于设置多圈数据在编码器数据帧结构中的哪一位开始（0~32）
	多圈位数	用于设置编码器的多圈位数（0~32）
	ERR起始位	用于设置错误位数据在编码器数据帧结构中的哪一位开始（0~32）
	ERR位数	用于设置编码器的错误位数（0~32）
	间隔等待时间	用于设置读取编码器数据的时间间隔，默认为1ms
Channel_n (n=2~5)	通道开关	开始：通道开启 关闭：通道关闭
	锁存功能	开启：锁存功能开启后，当通道输入状态变化后，保持当前通道状态，锁存时间到之后，通道状态才随着输入的状态而改变 关闭：关闭锁存功能
	锁存时间	可设置为150ms、300ms、600ms、1200ms
	抖动参数（每秒抖动次数）	1~255，每秒抖动次数，当每秒抖动次数大于“抖动参数”设置值时，通道质量码的状态为数据可疑（0x40）
	滤波时间	滤波时间，可设置为关闭、4ms、8ms、16ms、32ms滤波
Channel_n (n=6~9)	通道开关	开始：通道开启 关闭：通道关闭
	故障安全模式	输出保持：进入故障安全模式后，保持当前输出值 按预设值输出：进入故障安全模式后，输出设置的故障输出值

配置区	参数	说明
	故障输出值	故障输出值，进入故障安全模式，且故障安全模式设置为按预设值输出时，按此参数设置输出
	输出类型	状态输出：通道输出ON/OFF状态 脉宽输出：按照设置的脉宽输出宽度和脉冲输出个数，输出脉冲，脉宽输出通过通道输出值上升沿触发
	脉宽	1ms~600000ms
	脉冲个数	1~255

位号说明

变量	参数	说明
Inputs	诊断字BYTE0	BIT2：轻故障，轻故障时该位置为TRUE BIT3：重故障，重故障时该位置为TRUE
	诊断字BYTE1	预留
	诊断字BYTE2	BIT1：组态校验故障，故障时该位置为TRUE
	诊断字BYTE3	BIT2：内部电路故障，故障时该位置为TRUE
	诊断字BYTE4	预留
	诊断字BYTE5	BIT0：奇偶槽位，奇数槽位为TRUE
	DEVICE_ID	表示该模块ID
	SSI通道：单圈值 CHSSI_SGvalue	表示SSI通道编码器的单圈数据输入值
	SSI通道：多圈值 CHSSI_MUvalue	表示SSI通道编码器的多圈数据输入值
	SSI通道：当前位置值 CHSSI_CurPos	表示SSI通道编码器的当前位置数据输入值
	DI输入值CH00n (n = 1~4)	表示DI通道n的输入值
	DI质量码CH00n_D (n = 1~4)	表示DI通道n的质量码 0x00：表示通道开启 0x01：表示通道关闭 0x40：表示通道不确定状态
	DO输出值CH00n (n = 1~4)	表示DO通道n的输出值
	DO质量码CH00n_D (n = 1~4)	表示DO通道n的质量码 0x9F：表示通道关闭 0x00：表示通道开启 0x8C：表示通道输出电路故障

变量	参数	说明
	SSI质量码	表示SSI通道的质量码 0x9F：表示通道关闭 0x00：表示通道开启
	SSI输入值	表示SSI通道的输入值，即绝对值编码器位置值
Inputs PS	模块通信诊断	通信正常时为0x80，异常时为0x00
Outputs	DEVICE_ID	表示该模块ID
	输出值CH00n (n=1~4)	DO通道输出值
Outputs CS	输出数据诊断	当值为0x80时，输出数据有效

20.2.12 COM3204RTU-S模块

模块配置参数

配置参数	参数	说明
实时参数配置	数据上报周期	10ms~60000ms
COMn	数据位	通信参数设置
	停止位	
	校验方式	
	波特率	
	命令响应时间	0~1000ms，等待命令响应的时间
	命令间隔时间	0~1000ms，两条命令发送的间隔时间

01、02、03、04读命令配置参数

配置参数	参数	说明
实时数据参数配置	数据上报周期	10ms~60000ms
	选择的串口号	COM1~COM4
	命令周期	0~1000ms，两条命令发送的间隔时间
	从站地址	Modbus从站地址
	起始地址	Modbus命令起始地址

配置参数	参数	说明
	数据转换	字节不转换：数据大小端不转换 字节转换：数据进行大小端的转换

05、06、15、16写命令配置参数

配置参数	参数	说明
实时参数配置	数据上报周期	10ms~60000ms
	接收超时时间	1000ms~60000ms，超时没有收到Q区数据，进入故障安全模式
	选择的串口号	COM1~COM4
	命令周期	0~1000ms，两条命令发送的间隔时间
	从站地址	Modbus从站地址
	起始地址	Modbus命令起始地址

模块位号说明

变量	参数	说明
Inputs	故障诊断	Bit2：轻故障标志位，轻故障时该位置为TRUE Bit3：重故障标志位，重故障时该位置为TRUE
	编译日期：20xx年	例：0x23表示2023年
	模块诊断字BYTE1	Bit1：组态校验故障，故障时该位置为TRUE
	模块诊断字BYTE2	Bit2：内部电路故障，故障时该位置为TRUE
	模块诊断字BYTE3	预留
	模块诊断字BYTE4	Bit0：奇偶槽位，奇数槽位为TRUE。 Bit1：COM1故障，故障时该位置为TRUE Bit2：COM2故障，故障时该位置为TRUE Bit3：COM3故障，故障时该位置为TRUE Bit4：COM4故障，故障时该位置为TRUE 注：COMn故障判断逻辑：当COMn下所有命令均没有回复时，认为COMn故障（如果COMn下没有添加命令，认为正常）
	编译日期：日 月	例：0x1201表示1月12日
Inputs PS	模块背板通信诊断	通信正常时为0x80，异常时为0x00

读、写命令位号

变量	参数	说明
Inputs	通信故障诊断	Bit4：通信故障，故障时该位置为TRUE
	诊断结构类型	预留
	模块诊断字BYTE1	预留
	模块诊断字BYTE2	预留
	模块诊断字BYTE3	预留
	模块诊断字BYTE4	预留
Inputs PS	-	-

20.2.13 PW3203AC-S模块

配置参数

配置区	参数	说明
实时数据参数	数据上报周期	10ms~60000ms

位号说明

变量	参数	说明
Inputs	诊断字BYTE0	预留
	诊断字BYTE1	预留
	诊断字BYTE2	Bit1：组态校验故障，故障时该位置为TRUE
	诊断字BYTE3	Bit2：内部电路故障，故障时该位置为TRUE
	诊断字BYTE4	预留
	诊断字BYTE5	Bit0：奇偶槽位，奇数槽位为TRUE
	DEVICE_ID	表示该模块ID
	ADC_I_检测	表示该模块的负载电流检测值(mA)
	法拉电容中端电容检测	表示该模块的法拉电容组中端电压检测值(mv)
	24V输入检测	表示该模块的24V输入电压检测值(mv)
	系统掉电后输出电压检测	表示该模块系统掉电后输出电压检测值(mv)

变量	参数	说明
	法拉电容组电压检测	表示该模块法拉电容组电压检测值(mv)
	3V3电压检测	表示该模块系统侧3V3电压检测值(mv)
	温度检测	表示该模块温度检测值(℃)
	PowFail掉电检测	BIT0：掉电检测状态，该模块市电掉电时该位置为TRUE
	放电状态检测	BIT0：放电检测状态，该模块正处于放电状态时该位置为TRUE
Inputs PS	模块背板通信诊断	通信正常时为0x80，异常时为0x00

21 资料版本说明

表 **21-1** 版本升级更改一览表

资料版本	适用产品型号	更改说明
V1.0（20230821）	MCU4003-S01 V10.10.00	第一版本编写