



**XS 系列 PLCopen 标准控制器  
用户手册【软件篇】(XS Studio)**

XS 系列 PLCopen 标准控制器  
用户手册[软件篇] (XS Studio)

目录

---

产品简介 1

---

快速入门 2

---

网络配置 3

---

EtherCAT 配置 4

---

编程基础 5

---

编程语言 6

---

特殊功能 7

---

附录 8

---

手册更新日志

---

## 基本说明

- ◆ 感谢您购买了信捷 XS 系列可编程控制器。
- ◆ 本手册主要介绍 XS 系列可编程控制器的软件的相关使用内容。
- ◆ 在使用产品之前，请仔细阅读本手册，并在充分理解手册内容的前提下，进行编程。
- ◆ 请将本手册交付给最终用户。

## 用户须知

- ◆ 只有具备一定的电气知识的操作人员才可以对产品进行接线等其他操作，如有使用不明的地方，请咨询本公司的技术部门。
- ◆ 手册等其他技术资料中所列举的示例仅供用户理解、参考用，不保证一定动作。
- ◆ 将该产品与其他产品组合使用的时候，请确认是否符合有关规格、原则等。
- ◆ 使用该产品时，请自行确认是否符合要求以及安全。
- ◆ 请自行设置后备及安全功能，以避免因本产品故障而可能引发的机器故障或损失。

## 责任申明

- ◆ 手册中的内容虽然已经过仔细的核对，但差错难免，我们不能保证完全一致。
- ◆ 我们会经常检查手册中的内容，并在后续版本中进行更正，欢迎提出宝贵意见。
- ◆ 手册中所叙述的内容如有变动，恕不另行通知。

## 关联手册

关于 XS 系列 PLC 的硬件相关、高级运动控制指令应用等内容，请查询以下手册。

手册下载网址：[www.xinje.com](http://www.xinje.com)。

- ◆ 《XS 系列可编程控制器用户手册【硬件篇】》
- ◆ 《XS 系列可编程控制器用户手册【指令篇】》

WUXI XINJE ELECTRIC CO., LTD. 版权所有

未经明确的书面许可，不得复制、传翻或使用本资料及其中的内容，违者要对造成的损失承担责任。  
保留包括实用模块或设计的专利许可及注册中提供的所有权力。

二〇二三年 一月

# 目 录

1. 产品简介	1
1-1. 概述	2
1-1-1. 产品简介	2
1-1-2. 系统构成	4
1-2. XS Studio 概述	5
1-2-1. XS Studio 简介	5
1-2-2. XS Studio 与硬件的连接	5
1-2-3. 软件获取与安装	5
1-2-4. 软件安装步骤	5
1-2-5. 软件卸载	10
2. 快速入门	12
2-1. 软件启动	13
2-2. XS Studio 编程举例	13
2-2-1. 基本编程操作	14
2-2-2. 任务配置	19
2-2-3. 程序下载/读取	22
2-2-4. 程序调试	25
2-2-5. 仿真	27
2-2-6. PLC 脚本功能	27
2-3. XS Studio 编写流水灯程序样例	28
2-4. 如何登录设备	32
2-4-1. 登录设备的必备条件和操作简介	32
2-4-2. 扫描不到设备或者连接不上设备的解决方法	32
3. 网络配置	34
3-1. 设备组态	35
3-1-1. 网络组态	35
3-1-2. 硬件组态	40
3-1-3. 设备树操作	41
3-1-4. 组态编辑错误定位	42
3-2. MODBUS 通讯	43
3-2-1. MODBUS 主站配置	43
3-2-2. MODBUS 从站配置	45
3-2-3. MODBUS 通讯帧格式说明	46
3-3. 串口自由协议通讯	49
3-3-1. 参数配置	49
3-3-2. 应用举例	50
3-4. ModbusTCP 通讯	51
3-4-1. MODBUS TCP 主站配置	51
3-4-2. MODBUS TCP 从站配置	52
3-4-3. MODBUS TCP 常见故障	53
3-4-4. MODBUS TCP 通讯帧格式说明	53
3-5. CANopen 网络	55
3-5-1. 参数配置	56
3-5-2. 应用举例	63
3-6. EtherNet/IP 通讯	65
3-6-1. EtherNet/IP 作为从站的样例	66

3-6-2. EtherNet/IP 作为主站的样例	68
3-7. OPC UA 通讯	72
3-7-1. 通讯概述	72
3-7-2. 参数配置	72
3-7-3. OPC UA 样例	74
3-8. 信捷 Modbus TCP 协议	83
3-8-1. 参数配置	83
3-8-2. 触摸屏设置	84
4. EtherCAT 配置	86
4-1. EtherCAT 概要	87
4-1-1. 概述	87
4-1-2. 系统构成	87
4-1-3. 通讯规格	87
4-1-4. EtherCAT 通讯连接说明	88
4-2. EtherCAT 通讯规格	89
4-2-1. EtherCAT 帧结构	89
4-2-2. 状态机 ESM	89
4-2-3. 从站控制器 ESC	90
4-2-4. SII 区域	93
4-2-5. SDO	93
4-2-6. PDO	95
4-2-7. 通信同步模式	97
4-3. EtherCAT 参数配置	99
4-3-1. EtherCAT 主站	99
4-3-2. EtherCAT 从站	100
4-3-3. 轴配置	104
4-3-4. EtherCAT 控制工程	107
5. 编程基础	109
5-1. 直接地址	110
5-1-1. 定义语法	110
5-1-2. PLC 直接地址存储区域	110
5-2. 变量	111
5-2-1. 变量定义	111
5-2-2. 变量类型	112
5-3. 掉电保持变量	115
5-4. 配方操作	118
5-4-1. 应用举例	118
6. 编程语言	123
6-1. XS Studio 支持的编程语言	124
6-2. 结构化文本语言 (ST)	124
6-2-1. 简介	124
6-2-2. ST 程序执行顺序	124
6-2-3. 语句	126
6-3. 梯形图	134
6-3-1. 简介	134
6-3-2. LD 程序执行顺序	134
6-3-3. 组成元素	135
7. 特殊功能	139

7-1. 外部中断	140
7-1-1. 固件 1.1.0 版本以下外部中断应用举例	140
7-1-2. XS 系列固件 1.1.0 版本应用举例	140
7-2. 高速计数	142
7-3. 高速 IO 配置	143
7-4. 系统设置	147
7-5. PLC 指令	148
7-5-1. 应用举例	148
7-6. 时钟	155
7-6-1. 功能概述	155
7-6-2. 应用举例	155
8. 附录：常见问题及解决方法	157
8-1. Package	158
8-1-1. Package 命名规则	158
8-1-2. Package 的获取	158
8-1-3. Package 的安装	158
8-2. XS 系列 PLC 固件升级	159
8-2-1. 固件命名规则	159
8-2-2. 固件获取	159
8-2-3. 固件安装及其注意事项	159
8-3. XS 系列的本地扩展模块	160
8-4. XS 系列的远程扩展模块	162
8-5. 拨码	164
8-6. 用户新安装 XS Studio 上位机，打开后进行编译，会出现很多报错的原因	164
8-7. 网关显示红点的原因	164
8-8. 添加多个 EtherCAT 从站后，有 warning 提示的原因	164
8-9. EtherCAT 轴一运行会出现通讯断掉的原因	164
8-10. 用户怎么取消密码登录	164
8-11. 为什么无法连接到 PLC 设备	165
8-12. IP 地址修改不成功问题	165
8-13. 提示“没有可用于此对象的源码，是否浏览原始库以显示源代码”问题	166
8-14. setposition 清完位置之后断电上电绝对值伺服位置发生变化的问题	166
8-15. PLC 死机	166
8-16. 在线下载程序丢失	166
8-17. 不同电脑在同一局域网有时会连接到其他设备	166
8-18. 添加隐含检查功能	166
8-19. 实现掉电保持的注意点	169
8-20. 打开工程报错，工程保存必须以存档的方式保存	169
8-21. 设置可以添加行注释和节点注释	170
手册更新日志	171

# 1. 产品简介

本章就 PLCopen 标准控制器产品概述、上位机软件 XS Studio 安装和卸载两个部分进行展开介绍。

---

1. 产品简介 .....	1
1-1. 概述 .....	2
1-1-1. 产品简介 .....	2
1-1-2. 系统构成 .....	4
1-2. XS Studio 概述 .....	5
1-2-1. XS Studio 简介 .....	5
1-2-2. XS Studio 与硬件的连接 .....	5
1-2-3. 软件获取与安装 .....	5
1-2-4. 软件安装步骤 .....	5
1-2-5. 软件卸载 .....	10

## 1-1. 概述

### 1-1-1. 产品简介

XS Studio 涵盖 XSDH、XS3、XSLH、XSA330-W 等系列，为用户提供智能自动化解决方案。采用国际标准 IEC61131-3 架构，支持梯形图 LD、结构化文本 ST、功能块图 FBD、顺序功能流程图 SFC、控制流程框图 CFC 等多种编程语言。支持的总线包括：EtherCAT、Modbus/ModbusTCP、EtherNet/IP、OPC UA(Server)

支持的扩展模块：

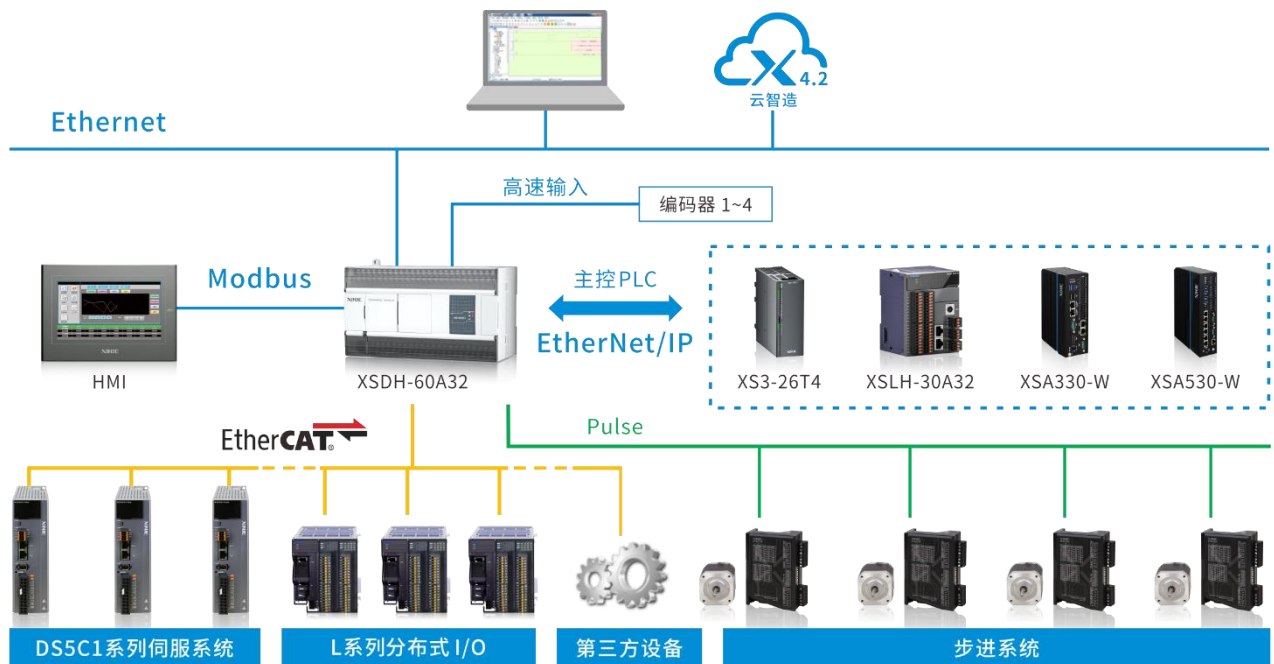
型号	功能
<b>XSDH 系列</b>	
XD-EnXmY	n 点输入，m 点输出，PNP/NPN 型输入，继电器/晶体管输出
XD-E4AD	14Bit, 4 通道模拟量输入（电流电压可选），一阶系数可调，每个通道可单独使能；
XD-E2DA	12Bit, 2 通道模拟量输出模块（电流电压可选）；
XD-E4DA	12Bit, 4 通道模拟量输出模块（电流电压可选）；
XD-E4DA-H	12Bit, 4 通道模拟量输出模块（电流电压可选）；通道间隔离处理，抗干扰性能更优；
XD-E8AD	14Bit, 8 通道模拟量输入模块；前 4 路为电压（0~5V、0~10V、-5~5V、-10~10V）输入，后 4 路为电流（0~20mA、4~20mA、-20~20mA）输入，一阶系数可调，每个通道可单独使能；（注：硬件版本需 H2.2 及以上支持双极性）
XD-E8AD-A	14Bit, 8 通道模拟量电流（0~20mA、4~20mA、-20~20mA）输入，一阶系数可调，每个通道可单独使能；（注：硬件版本需 H2.2 及以上支持双极性）
XD-E8AD-V	14Bit, 8 通道模拟量电压（0~5V、0~10V、-5~5V、-10~10V）输入，一阶系数可调，每个通道可单独使能；（注：硬件版本需 H2.2 及以上支持双极性）
XD-E12AD-V	14Bit, 12 通道模拟量电压（0~5V、0~10V、-5~5V、-10~10V）输入，一阶系数可调，每个通道可单独使能；
XD-E4AD2DA	14Bit, 4 通道模拟量输入（电流电压可选），电流 0~20mA、4~20mA、-20~20mA 可选，电压 0~5V、0~10V、-5~5V、-10~10V 可选；12Bit 2 通道模拟量输出模块（电流电压可选），电压 0~5V、0~10V、-5~5V、-10~10V 可选，电流 0~20mA、4~20mA 可选，电流一阶系数可调，每个通道可单独使能；（注：V6 及以下版本的 XD-E4AD2DA 模块不支持-5~5V、-10~10V、-20~20mA 范围）
XD-E2AD2PT2DA	2 通道 PT100 温度采集(分辨率为 0.1℃)；16Bit, 2 通道模拟量输入(电流、电压可选)；10Bit, 2 通道模拟量输出(电压电流可选)；每个通道可单独使能；
XD-E3AD4PT2DA	4 通道 PT100 温度采集(分辨率为 0.1℃)；14Bit, 3 通道模拟量输入(0~20mA, 4~20mA 可选)；10Bit, 2 通道模拟量输出(0~5V, 0~10V 可选)；每个通道可单独使能；
XD-E2TC-P	2 通道热电偶，支持多种类型热电偶温度传感器用模拟输入，分辨率 0.1℃，2 通道独立输出 PID 参数；
XD-E6TC-P	6 通道热电偶，支持多种类型热电偶温度传感器用模拟输入，分辨率 0.1℃，6 通道独立输出 PID 参数；
XD-E6TC-P-H	6 通道热电偶，支持多种类型热电偶温度传感器用模拟输入，通道间隔离，分辨率 0.1℃，6 通道晶体管输出，6 组独立的 PID 参数，支持自整定功能，内置冷端补偿；
XD-E6PT-P	-100~500℃，6 通道 PT100 温度采集模块，分辨率 0.1℃，PID 输出；
XD-E4PT3-P	-100~500℃，4 通道 PT100（三线制）温度采集模块，分辨率 0.1℃，4 通道独立 PID 输出；
XD-E1WT-D	可采集一路压力传感器的模拟量电压信号(-20~20mV)，22 位的高精度 A/D 转换，采用 $\Delta$ - $\Sigma$ ADC 的 A/D 转换方式，更高更快的 CPU 处理速度，更加优化的算法，抗共振性能更好，供电电源 DC24V；



型号	功能
<b>XSDH 系列</b>	
XD-E2WT-D	可采集二路压力传感器的模拟量电压信号(-20~20mV), 22 位的高精度 A/D 转换, 采用 $\Delta$ - $\Sigma$ ADC 的 A/D 转换方式, 更高更快的 CPU 处理速度, 更加优化的算法, 抗共振性能更好, 供电电源 DC24V;
XD-E4WT-D	可采集四路传感器模拟电压信号(-20~20mV), 22 位高精度 AD 转换, 采用 $\Delta$ - $\Sigma$ ADC 的 A/D 转换方式, 更高更快的 CPU 处理速度, 更加优化的算法, 抗共振性能好, 供电电源 DC24V;
XD-E4SSI	XD 系列连接 SSI 信号编码器专用扩展模块, 一个模块同时最多可以连接 4 路, 通讯速度可达 400us/通道;
XD-NES-ED	XD 系列 PLC 扩展 ED 模块, 可以扩展 1 个 RS232 或者 RS485 通讯口; (注: 只能使用其一)
XD-NS-BD	XD 系列 PLC 扩展 BD, RS-232 通讯功能;
XD-NE-BD	XD 系列 PLC 扩展 BD, 总线通讯功能, X-NET 标准线接口, 此 BD 板还可用作 RS485 通讯扩展板;
<b>XSLH 系列</b>	
XL-EnXmY	n 点输入, m 点输出, PNP/NPN 型输入, 输入滤波时间可调, 继电器/晶体管输出(注: -A 类型的扩展模块为牛角端子, 需配合端子排以及专用扩展电缆同时使用)
XL-E4AD	14Bit 4 通道模拟量输入(电压可选 0~10V、0~5V、-5~5V、-10~10V; 电流可选 0~20mA、4~20mA、-20~20mA), 一阶系数可调, 每个通道可单独使能, 供电电源 DC24V;
XL-E4AD2DA	14Bit 4 通道模拟量输入(电压可选 0~10V、0~5V、-5~5V、-10~10V; 电流可选 0~20mA、4~20mA、-20~20mA); 12Bit 2 通道模拟量输出模块(电压电流可选 0~10V、0~5V、-5~5V、-10~10V、0~20mA、4~20mA), 一阶系数可调, 每个通道可单独使能, 供电电源 DC24V;
XL-E4DA	12Bit 4 通道模拟量输出模块(电压可选 0~10V、0~5V、-5~5V、-10~10V; 电流可选 0~20mA、4~20mA), 一阶系数可调, 每个通道可单独使能, 供电电源 DC24V;
XL-E8AD-A	14Bit, 8 通道模拟量输入(电流可选 0~20mA、4~20mA、-20~20mA), 供电电源 DC24V;
XL-E8AD-A-S	16Bit, 8 通道模拟量输入(电流可选 0~20mA、4~20mA、-20~20mA), 供电电源 DC24V;
XL-E8AD-V	14Bit, 8 通道模拟量输入(电压可选 0~10V、0~5V、-10~10V、-5~5V), 供电电源 DC24V;
XL-E8AD-V-S	16Bit, 8 通道模拟量输入(电压可选 0~5V、0~10V、-5~5V、-10~10V), 供电电源 DC24V;
XL-E4TC-P	4 通道热电偶, 支持多种类型热电偶温度传感器用模拟输入, 分辨率 0.1°C, 4 通道独立输出 PID 参数, 供电电源 DC24V;
XL-E4PT3-P	-100~500°C, 4 路 PT100(三线制)温度采集, 分辨率 0.1°C, 模块自带 PID 控制输出功能, 供电电源 DC24V;
XL-E1WT-D	可采集一路压力传感器的模拟量电压信号(-20~20mV), 24 位的高精度 A/D 转换, 采用 $\Delta$ - $\Sigma$ ADC 的 A/D 转换方式, 更高更快的 CPU 处理速度, 更加优化的算法, 抗共振性能更好, 供电电源 DC24V;
XL-E2WT-D	可采集二路压力传感器的模拟量电压信号(-20~20mV), 24 位的高精度 A/D 转换, 采用 $\Delta$ - $\Sigma$ ADC 的 A/D 转换方式, 更高更快的 CPU 处理速度, 更加优化的算法, 抗共振性能更好, 供电电源 DC24V;
XL-E4WT-D	可采集四路压力传感器的模拟电压信号(-20~20mV), 24 位高精度 AD 转换, 采用 $\Delta$ - $\Sigma$ ADC 的 A/D 转换方式, 更高更快的 CPU 处理速度, 更加优化的算法, 抗共振性能更好, 供电电源 DC24V;
XL-ETR	当 XL 系列扩展模块数量超过 5 个及以上时增加此终端电阻模块;
XL-P50-E	XL 系列电源模块, AC220V 输入, DC24V 输出, 输出功率 50W;

型号	功能
<b>XSLH 系列</b>	
XL-NES-ED	XL 系列 PLC 扩展 ED 模块，可以扩展 1 个 RS232 或者 RS485 通讯口；（注：只能使用其一）
<b>XS3 系列</b>	
XG-EnXmY	n 点输入, m 点输出, 正负逻辑可设, 输入滤波时间可调; 模块无需供电电源, NPN&PNP 输入兼容; （注：64 点模块需配备专用延长线和端子台使用）
XG-E8TC-P	8 路热电偶 TC 温度采集, 分辨率 0.1℃, 支持多种类型热电偶温度传感器用模拟量输入, 模块自带 PID 控制输出功能, 供电电源 DC24V;
XG-E8PT3-P	-100~500℃, 8 路 PT100（三线制）温度采集, 分辨率 0.1℃, 模块自带 PID 控制输出功能, 供电电源 DC24V;

1-1-2. 系统构成



## 1-2. XS Studio 概述

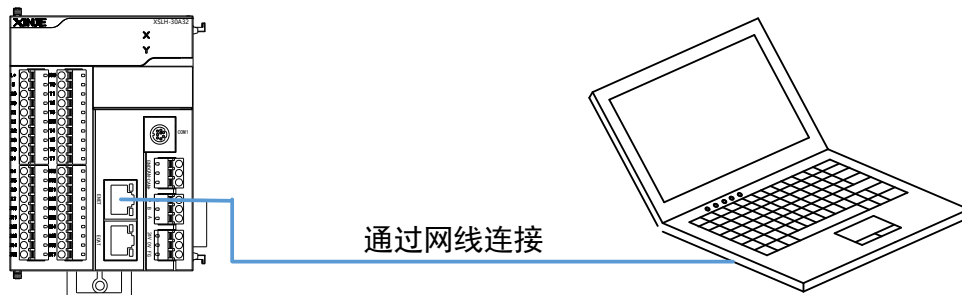
### 1-2-1. XS Studio 简介

XS Studio 是基于 CODESYS 开发，面向 XS 系列的编程组态软件。集成了 PLC 编程、可视化 HMI、安全 PLC、控制器实时核、现场总线及运动控制，能够提供一套完整的配置、编程、调试、监控环境，可以灵活自由地处理功能强大的 IEC 语言。

- ◆ 强大的软件仿真、在线调试及程序检查能力，不需要连接 PLC 硬件，即可完成程序调试仿真。
- ◆ 具备方便的产品配置功能，可以轻松快速的实现包括 CPU 配置、IO 模块配置和高速 IO 等。
- ◆ 提供智能的调试查错功能。当用户输入了错误的应用程序代码时，立刻会接收到编译器发出的语法错误警告及错误信息，让编程人员可以迅速做出相应纠正。
- ◆ 强大的运动控制模块。基于 PLCopen 的工具包可以实现单轴、多轴运动，电子凸轮传动，电子齿轮传动，复杂多轴 CNC 控制等。

### 1-2-2. XS Studio 与硬件的连接

编程设备可以通过网线与 PLC 相连，使用 XS Studio 软件编写用户程序，将程序下载到 PLC 后进行程序监控并控制 PLC。



### 1-2-3. 软件获取与安装

#### 1、系统配置要求

硬件及软件的要求：

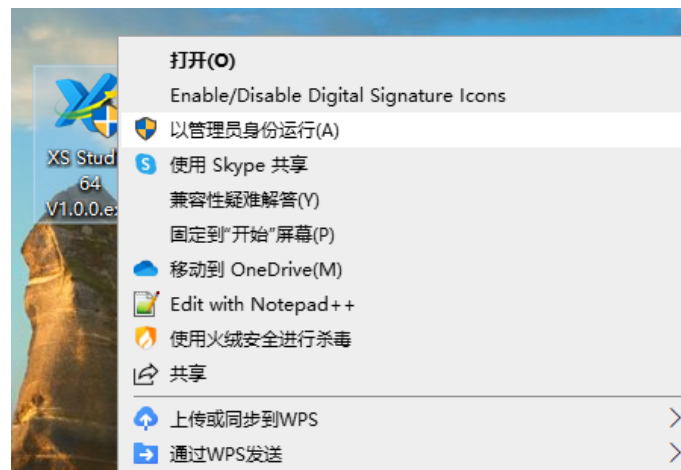
- ◆ windows 7、windows 8 或 windows 10 操作系统，推荐 64 位操作系统；
- ◆ 内存在 4GB 及以上；
- ◆ 硬盘空间 12GB 以上。

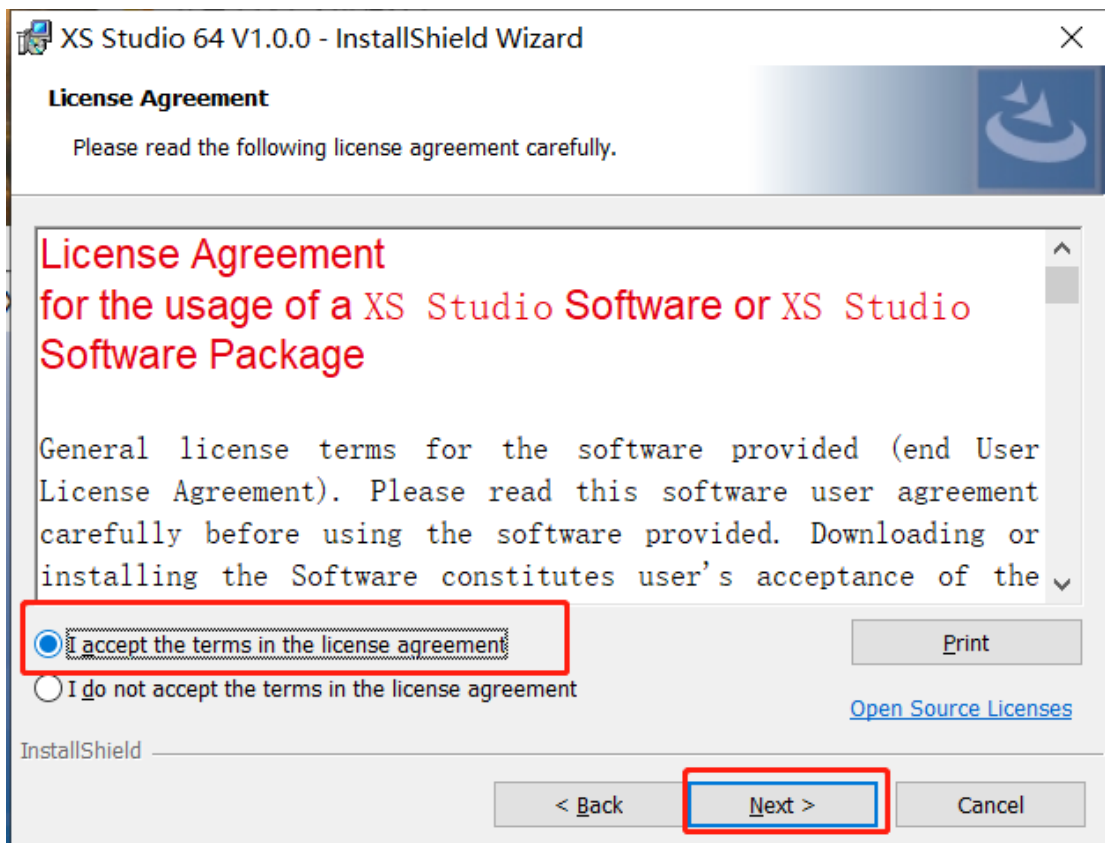
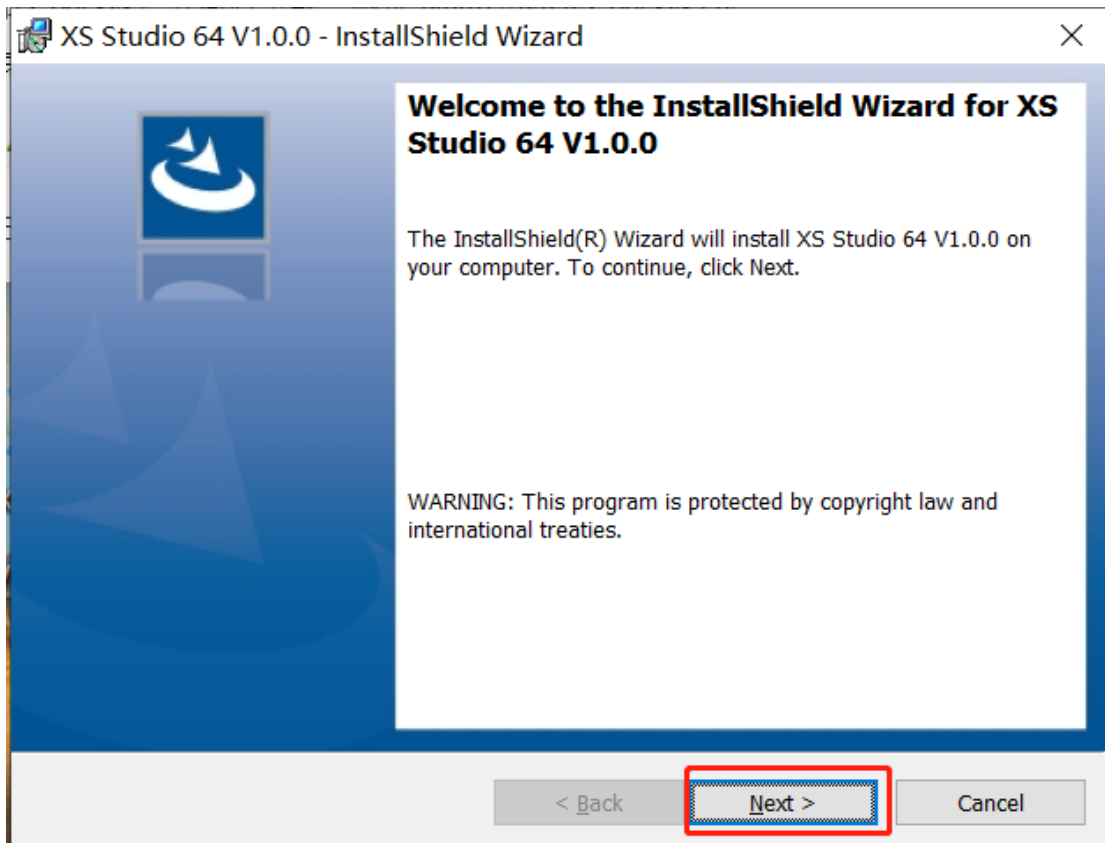
#### 2、软件获取

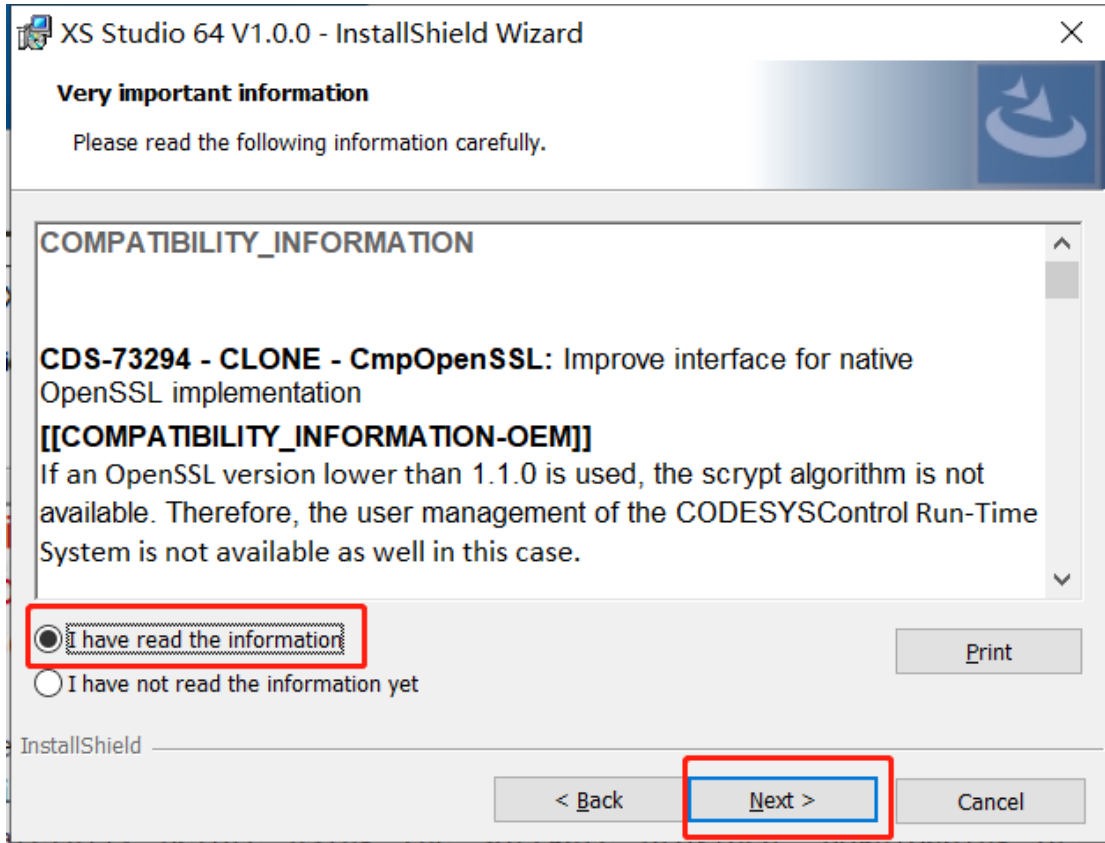
信捷官网服务于与支持-下载中心，下载网址：[www.xinje.com](http://www.xinje.com)。

### 1-2-4. 软件安装步骤

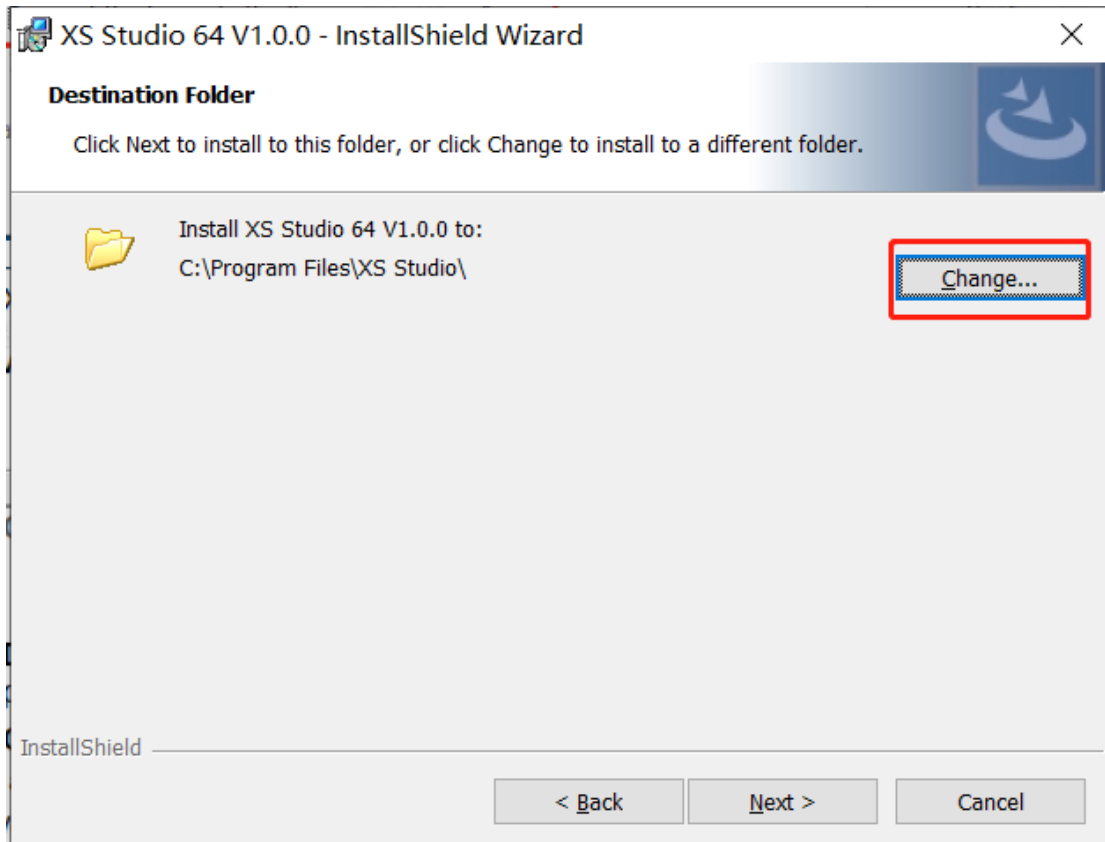
- 1、右键以管理员身份运行；

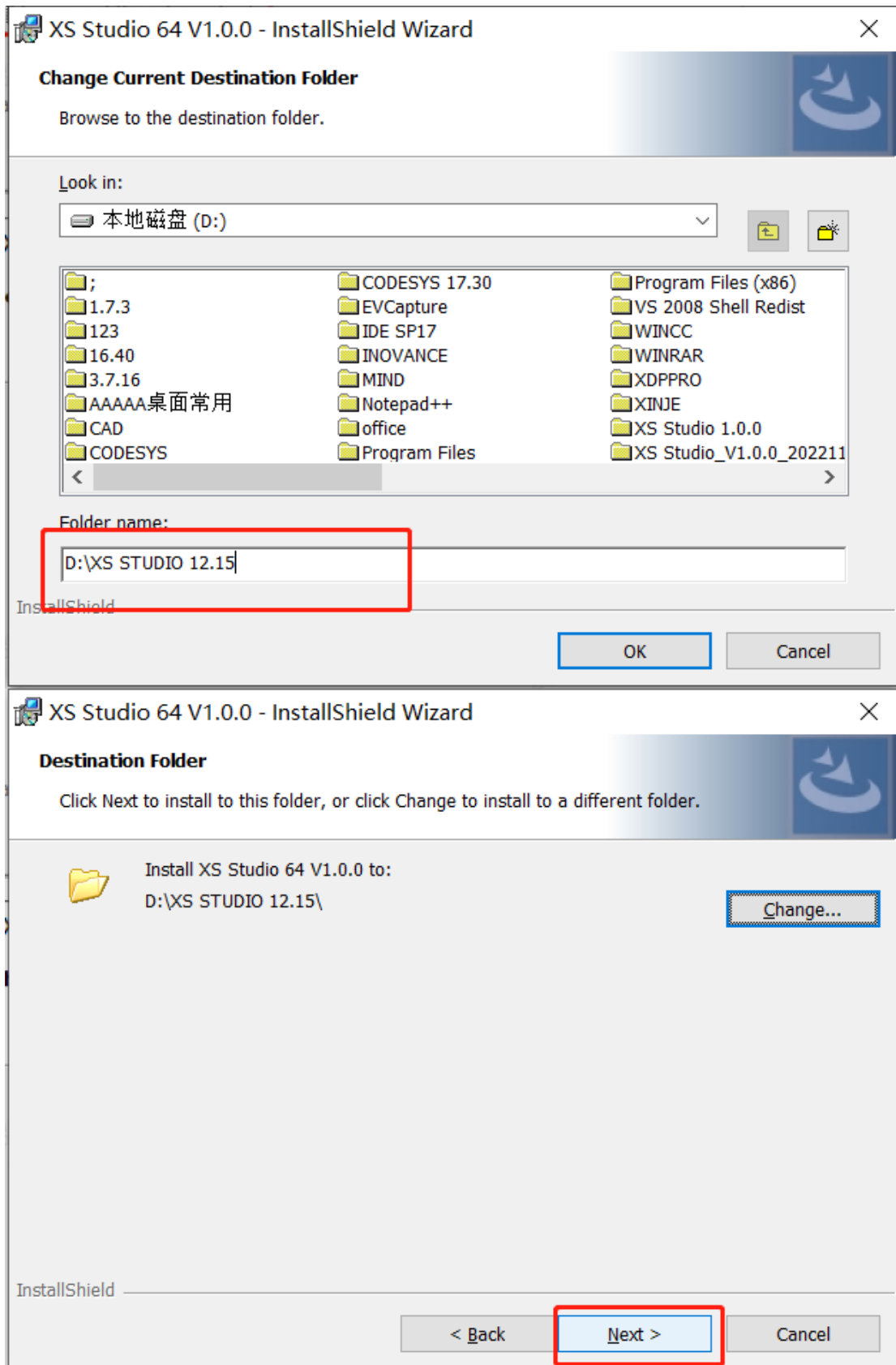




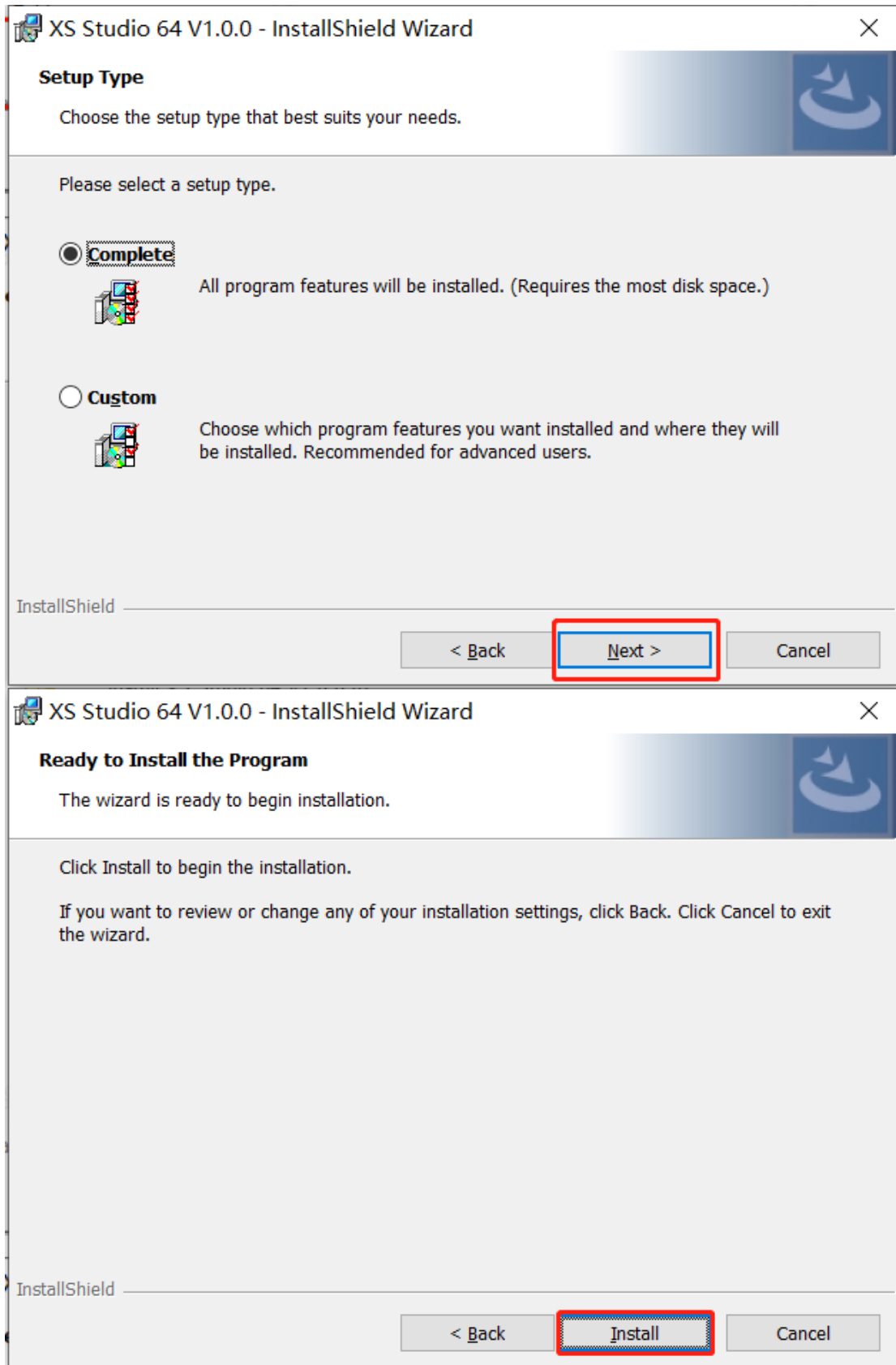


2、软件建议安装在系统盘以外的盘；

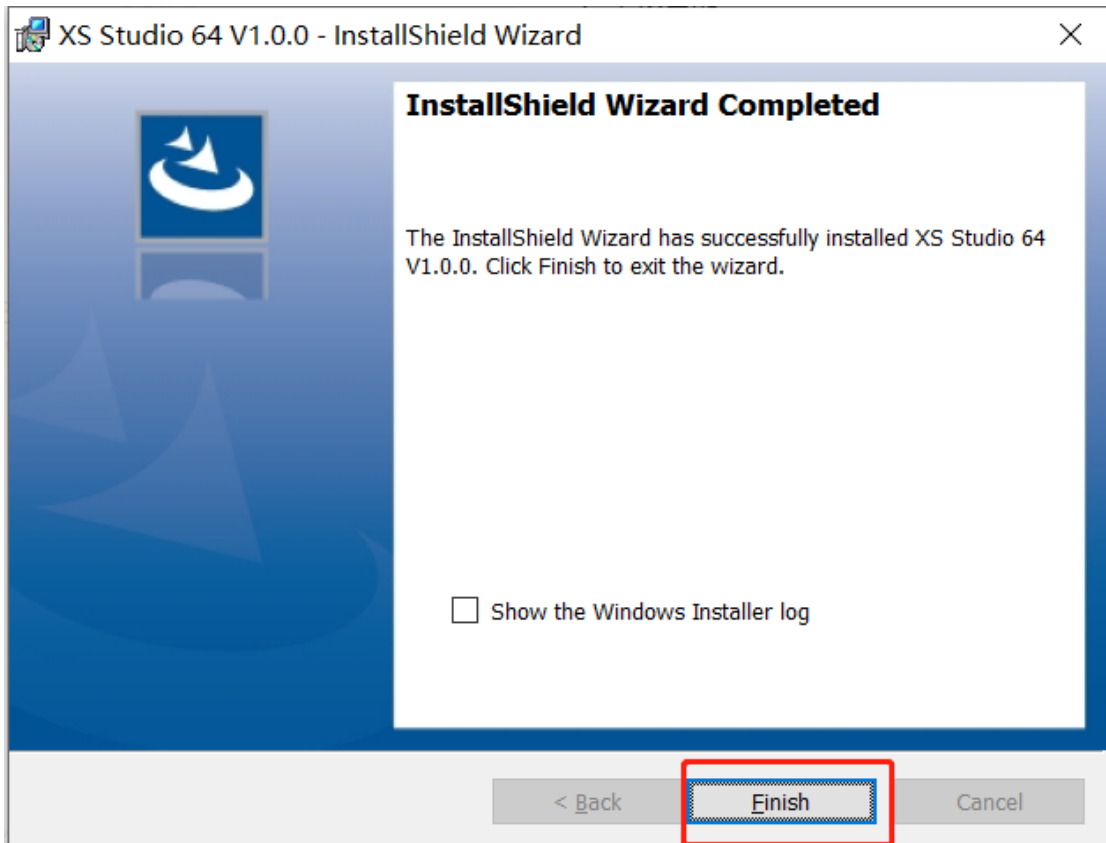




3、完全安装:

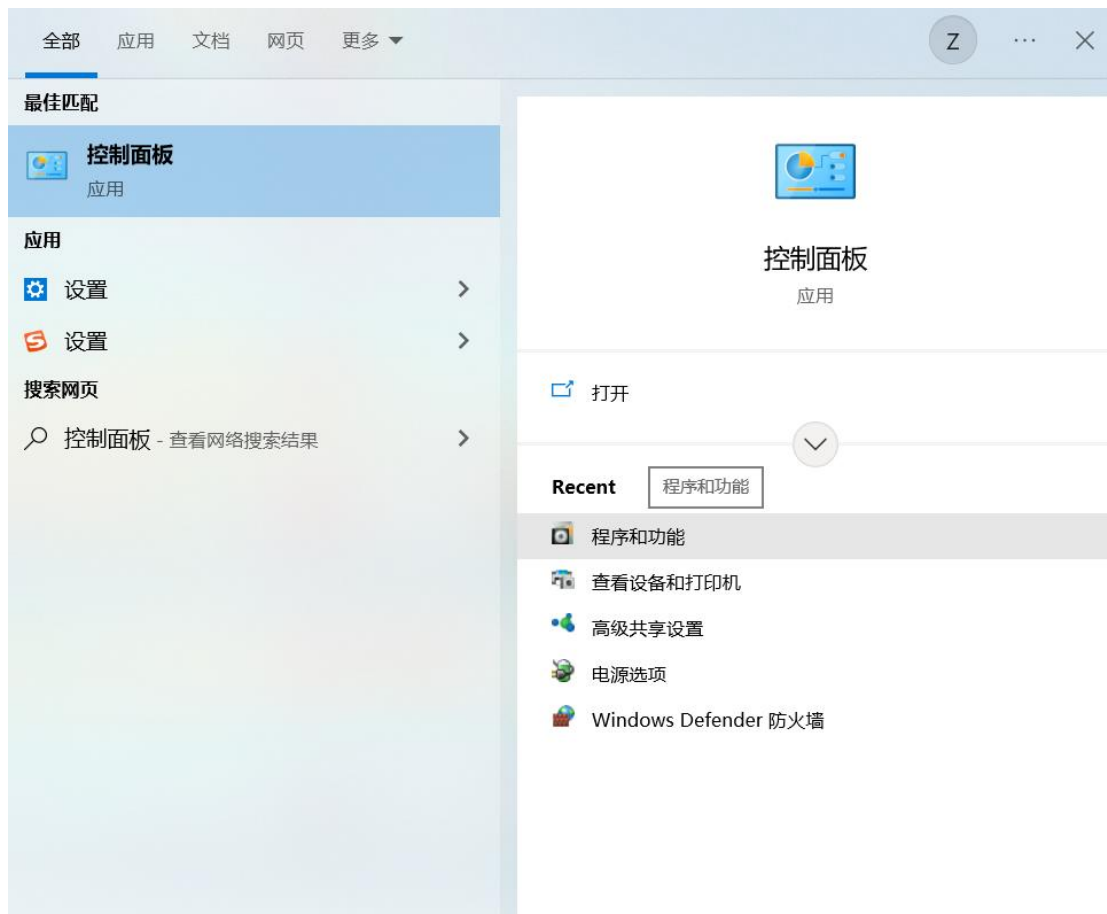


4、安装完成。

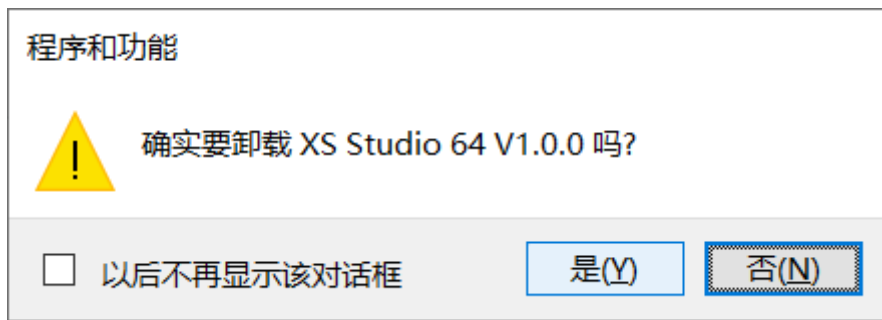


### 1-2-5. 软件卸载

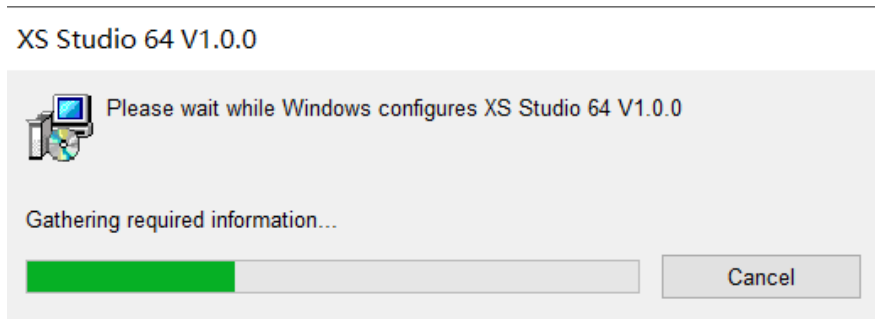
1、控制面板-程序-程序功能；







2、等待卸载完成。



## 2. 快速入门

本章举例介绍 XS Studio 的编程方法，并说明如何登陆设备。

---

2. 快速入门.....	12
2-1. 软件启动.....	13
2-2. XS Studio 编程举例 .....	13
2-2-1. 基本编程操作.....	14
2-2-2. 任务配置.....	19
2-2-3. 程序下载/读取.....	22
2-2-4. 程序调试.....	25
2-2-5. 仿真.....	27
2-2-6. PLC 脚本功能 .....	27
2-3. XS Studio 编写流水灯程序样例 .....	28
2-4. 如何登录设备.....	32
2-4-1. 登录设备的必备条件和操作简介.....	32
2-4-2. 扫描不到设备或者连接不上设备的解决方法.....	32

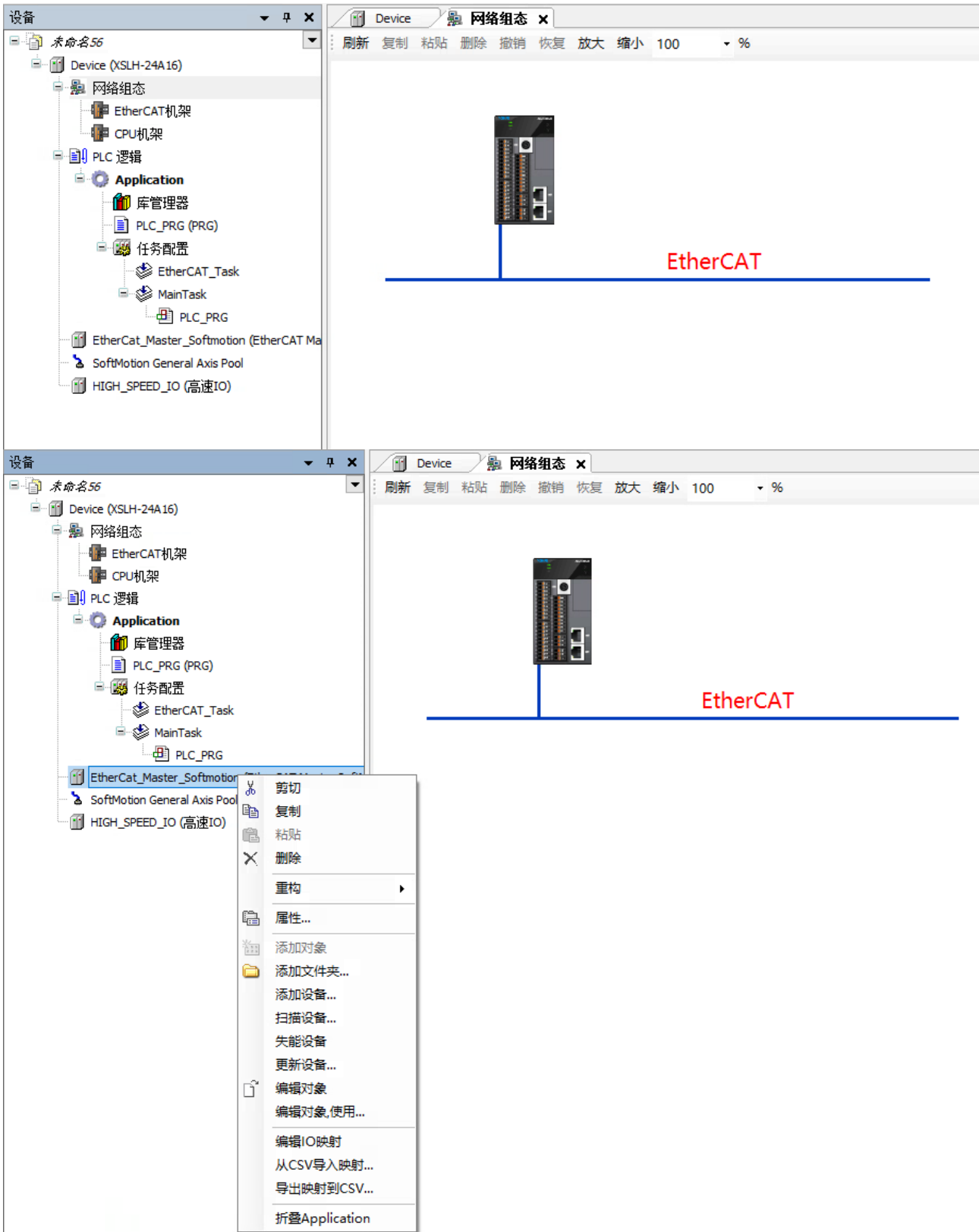
## 2-1. 软件启动

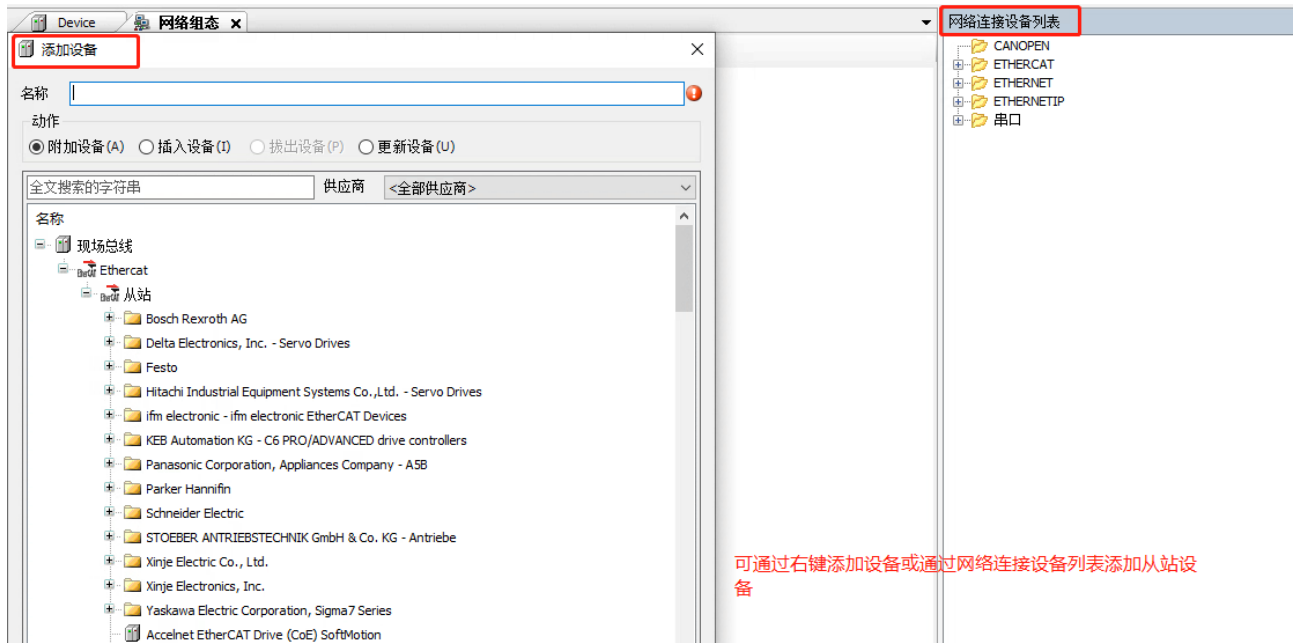


双击 XS Studio 图标，启动 XS Studio 软件。

## 2-2. XS Studio 编程举例

根据实际拓扑配置对应设备。



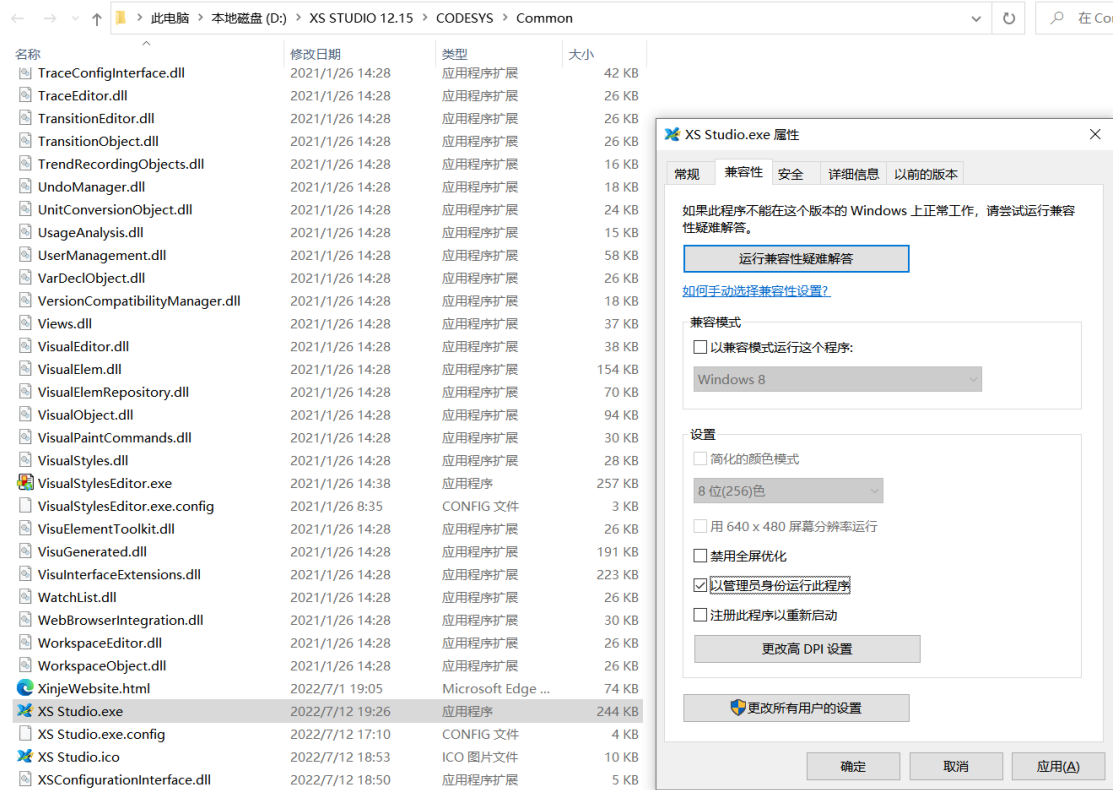


## 2-2-1. 基本编程操作

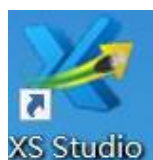
### 1、启动 XS Studio

#### 1) 设置管理员权限

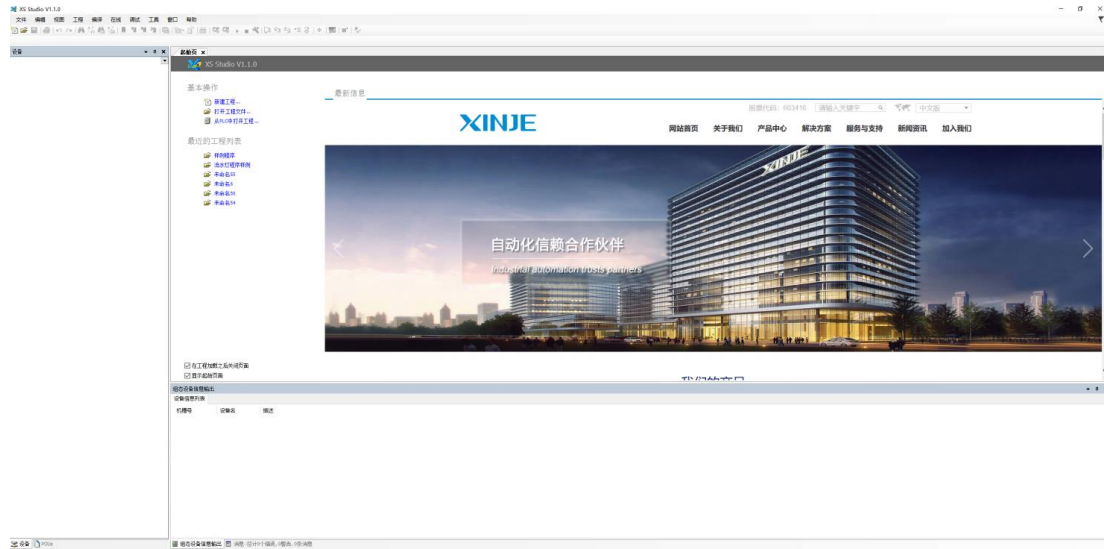
在 Win7 系统下需要以管理员权限打开软件, 在 XS Studio 的默认安装路径下找到 XS Studio.EXE 文件, 选中该文件后点击鼠标右键, 选择属性, 将“Run This Program as an administrator”或者是“以管理员身份运行此程序”的勾选上, 点击“OK”确认, 如图所示, 确认后每次运行 XS Studio 系统则会默认以管理员权限自动进入 XS Studio。




#### 2) 启动 XS Studio



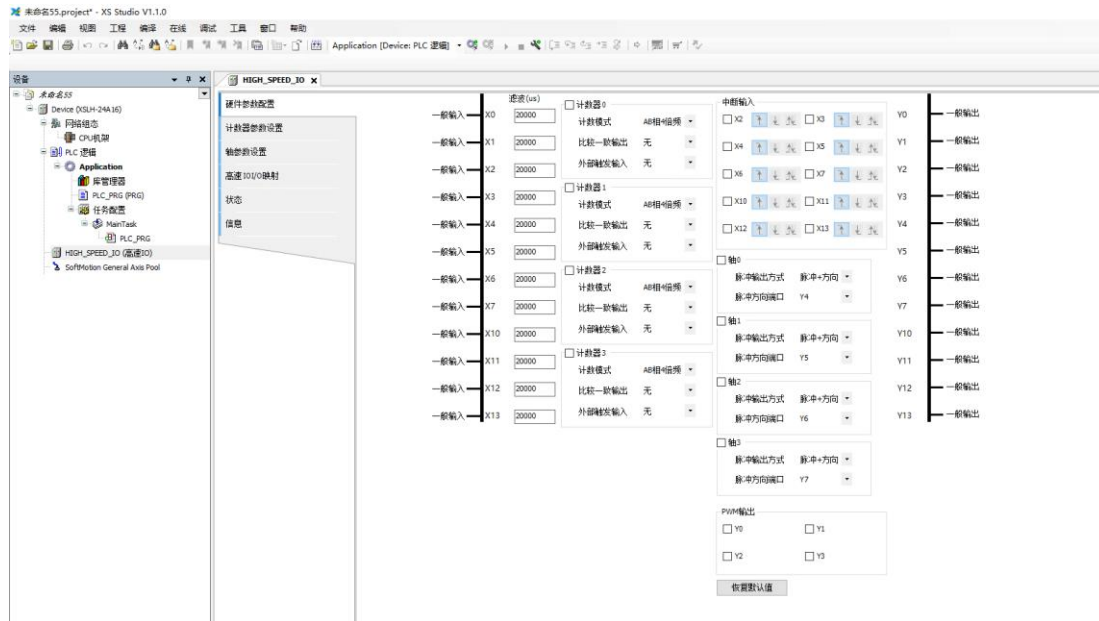
从开始菜单选择>XS Studio > XS Studio 或直接双击桌面上的图标启动 XS Studio。



### 3) 创建工程项目

(1) 点击  图标，新建工程。选择“标准工程”，选择对应机型，选择熟悉的编程语言，填写工程名称，选择文件保存位置。





## 2、PLC 程序文件的建立

PLC 程序文件的建立，是运行结构的运行顺序的建立，也是编程模式的建立，甚至包括了数据区域的分割。在建立程序文件之前，应当对运行结构进行详细的划分，确定连续型、周期型和事件触发型任务，并对周期型和事件触发型任务安排优先级别。新建一个 XS Studio 项目后，自动生成一个默认连续型任务，任务下有一个默认的程序及 PLC\_PRG。

### 1) 创建任务

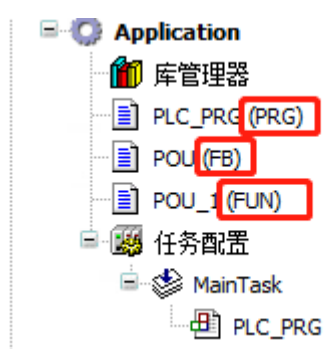
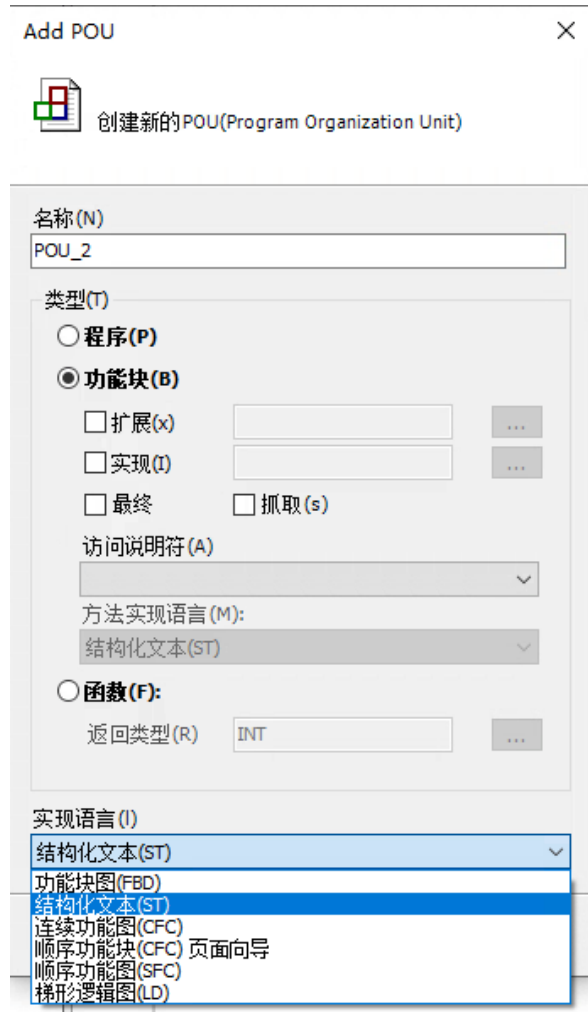
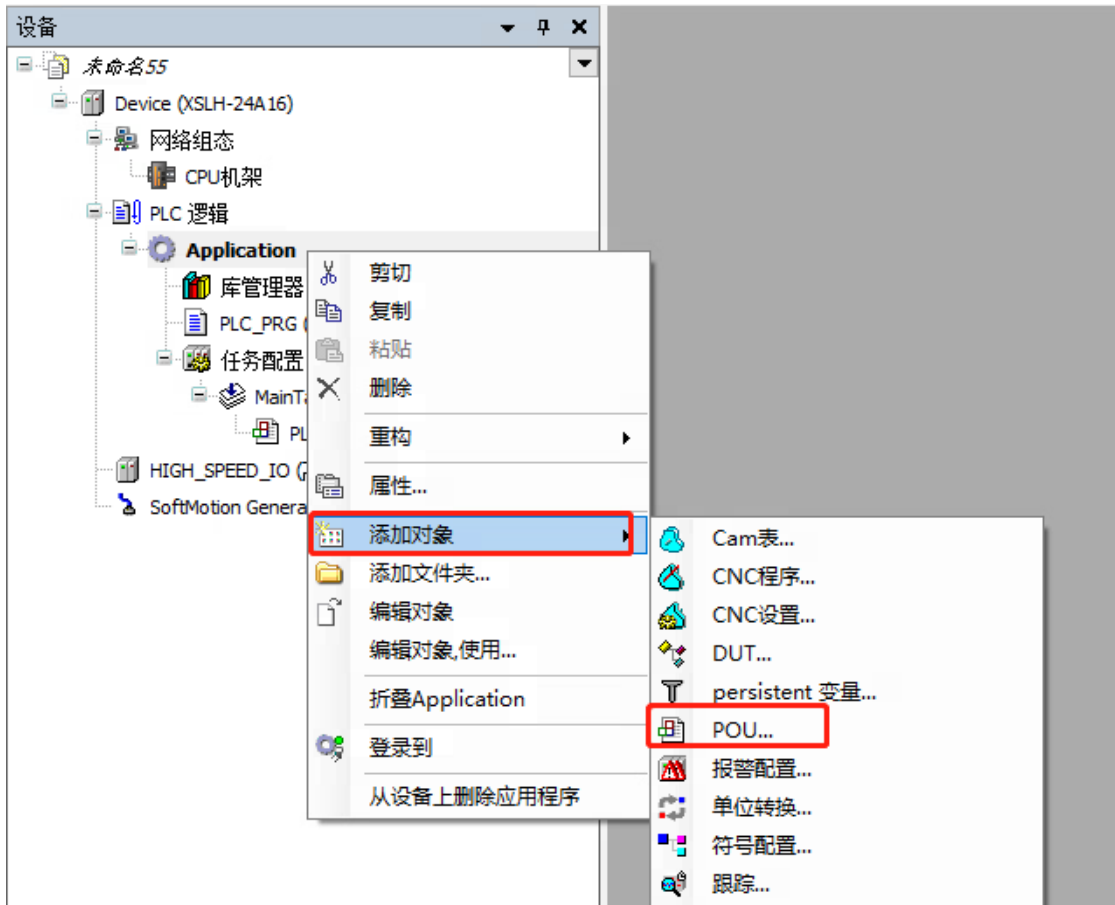
首先，在“任务配置”中管理任务，通常的项目应用中，可以分为主逻辑任务、通讯任务，通讯由于要更新数据源，会将它放在比较高的任务优先等级及较短的循环时间。此外，如项目中涉及到运动控制，也会将其独立出一个任务，并将其放在最高的任务优先等级，如图所示：



### 2) 添加 POU

#### (1) 自定义程序/功能块/函数

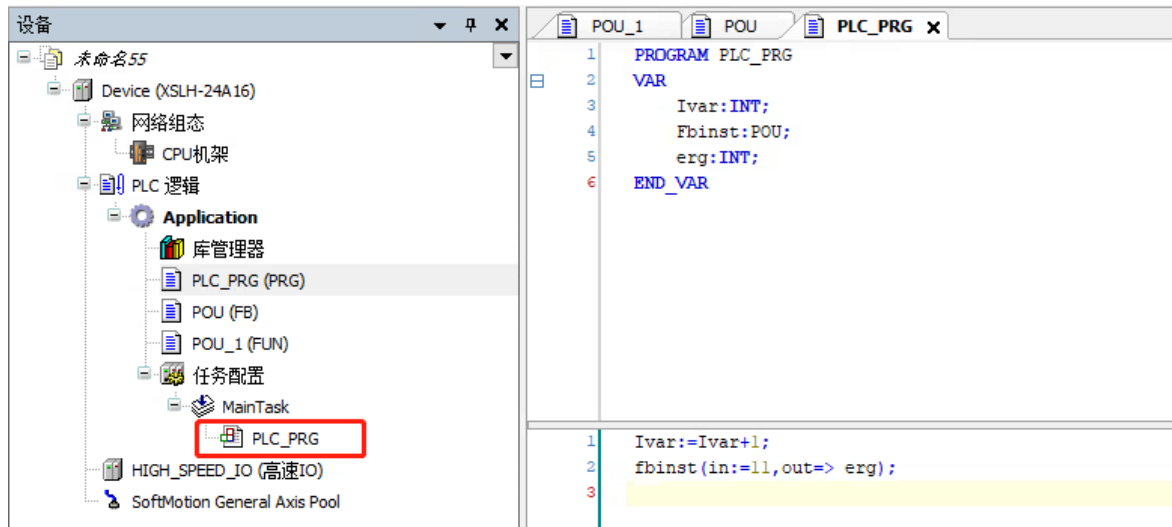
用户可以在项目中使用右键菜单的命令“添加对象”，选择“POU”程序组织单元，会弹出如下图所示的对话框，用户可以选择添加程序，功能块或函数，下拉菜单中可以选择对应的编程语言。添加后，可以在左边的项目设备树中查看程序组织单元括号内对应的属性，FB 为功能块，FUN 为函数，PRG 为程序。



## (2) 声明变量

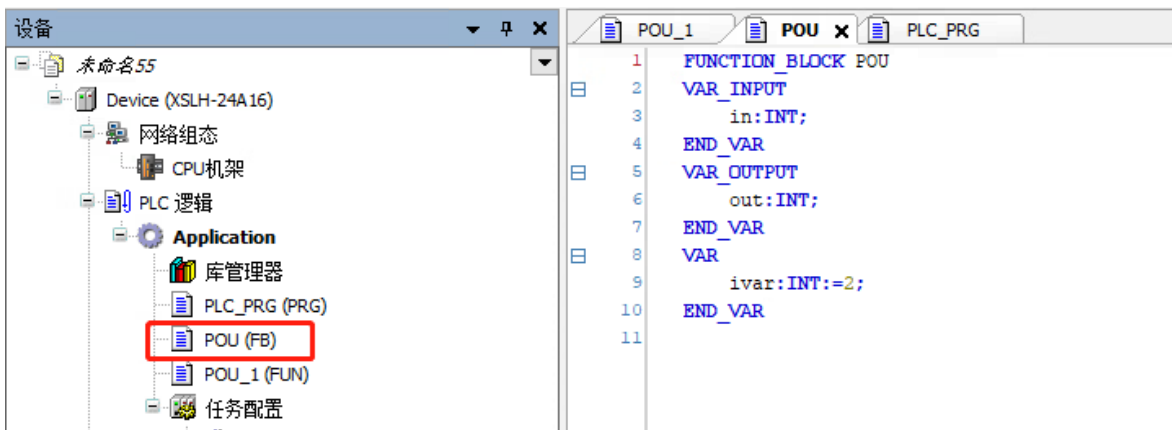
## ◆ 在“PLC\_PRG”程序内声明变量

双击设备树中的“PLC\_PRG”，自动在 XS Studio 用户界面的 ST 语言编辑器中打开。语言编辑器包括声明部分（上部），实现部分（下部），由一个可调的分割线分开。声明部分包括：显示在左侧边框中的行号、POU 类型和名称（如“PROGRAM PLC\_PRG”），以及在关键字“VAR”和“END\_VAR”之间的变量声明。如下图所示：



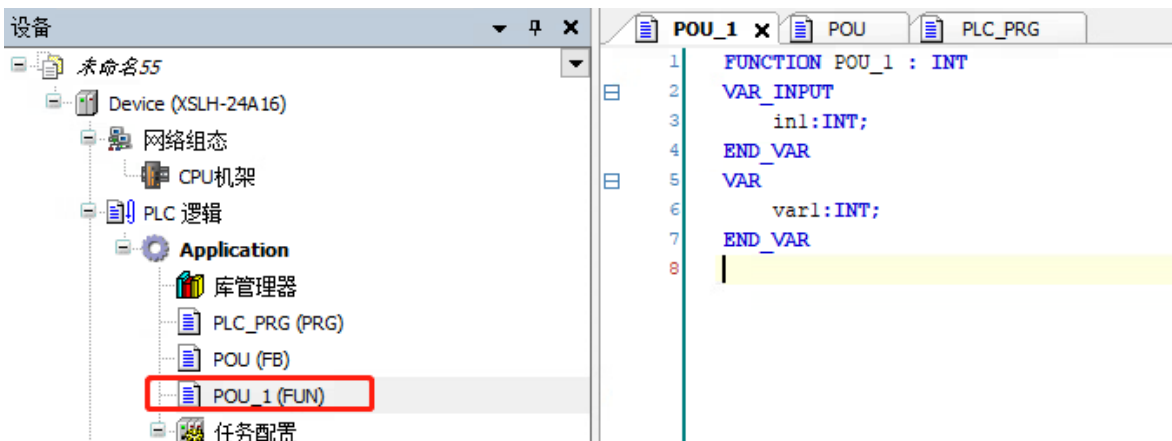
## ◆ 在功能块 FB 内声明变量

功能块语言编辑器界面与程序的编辑器界面类似，也包括声明部分与代码部分。用户声明的所有变量最终是给程序组织单元所用，变量声明中可声明接口变量、静态变量和本地变量，具体如下图所示：



## ◆ 在函数 FUN 内声明变量

函数是有至少一个输入变量、无私有数据、仅有一个返回值的基本算法单元。函数是没有静态变量的程序组织单元。即用相同的输入参数调用某一函数时，该函数总能生成相同的结果作为函数值（返回值）。函数的一个重要特性是它们不能使用内部变量存储数值，这点与功能块完全不同。具体如下图所示：



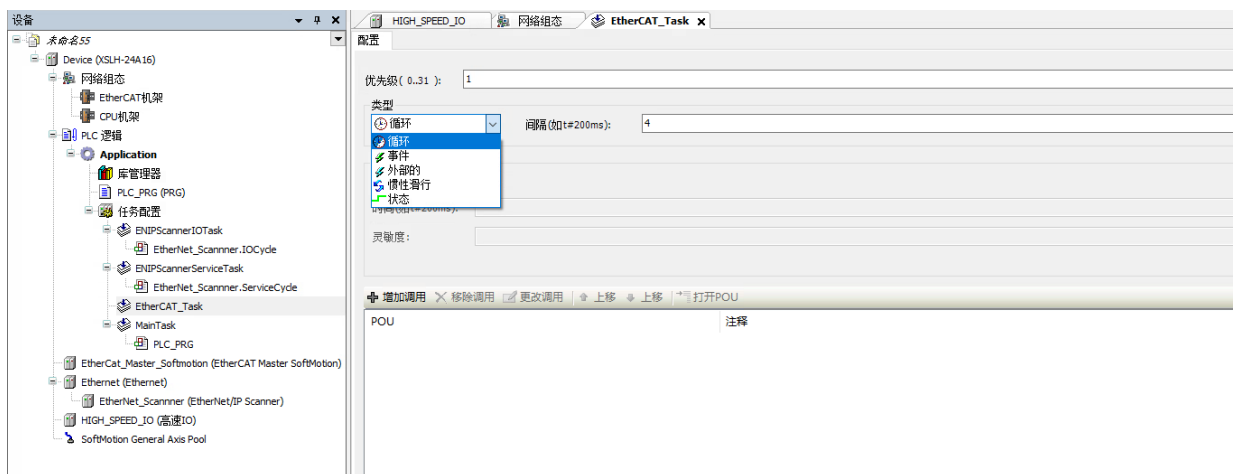


## 2-2-2. 任务配置

### 1、概述

一个程序可以用不同的编程语言来编写。典型的程序由许多互连的功能块组成，各功能块之间可互相交换数据。在一个程序中不同部分的执行通过“任务”来控制。“任务”被配置以后，可以使一系列程序或功能块周期性地执行或由一个特定的事件触发开始执行程序。在设备树中有“任务管理器”选项卡，使用它除了声明特定的 PLC\_PRG 程序外，还可以控制工程内其他子程序的执行处理。任务是用于规定程序组织单元在运行时的属性，它是一个执行控制元素，具有调用的能力。在一个任务配置中可以建立多个任务，而一个任务中，可以调用多个程序组织单元，一旦任务被设置，它就可以控制程序周期执行或者通过特定的事件触发开始执行。

在任务配置中，用名称、优先级和任务的启动类型来定义它。这启动类型可以通过时间（周期的，随机的）或通过内部或外部的触发任务时间来定义，例如使用一个布尔型全局变量的上升沿或系统中的某一特定事件。对每个任务，可以设定一串由任务启动的程序。如果在当前周期内执行此任务，那么这些程序会在一个周期的长度内被处理。优先权和条件的结合将决定任务执行的时序，任务设置界面如下图所示：



在任务配置时有如下的属性，编程者需遵循如下规则：

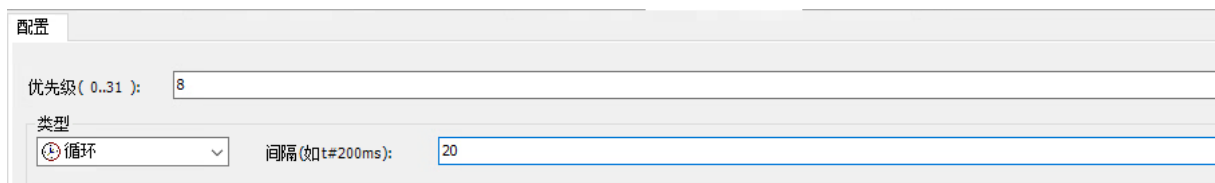
- ◆ 循环任务的最大数为 100。
- ◆ 自由运行任务的最大数为 100。
- ◆ 事件触发任务的最大数为 100。
- ◆ 根据目标系统，PLC\_PRG 可能会在任何情况下作为一个自由程序执行，而不用插入任务配置中。
- ◆ 处理和调用程序是根据任务编辑器内自上而下的顺序所执行的。

#### 1) 任务优先级 Priority

XS Studio 中的可以对任务的优先级进行设置，一共可以设 32 个级别（0~31 之间的一个数字，0 是最高优先级，31 是最低优先级）。当一个程序在执行时，优先级高的任务优先于优先级任务低的任务，高优先级任务 0 能中断同一资源中较低优先级的程序执行，使较低优先级程序执行被放缓。

**注意：**在任务优先级等级分配时，请勿分配具有相同优先级的任务。如果还存在其他任务试图先于具有相同优先级的任务，则结果可能不确定且不可预知。

如果任务的类型为“循环”，则按照“间隔”中的时间循环执行，具体设置如下图所示：

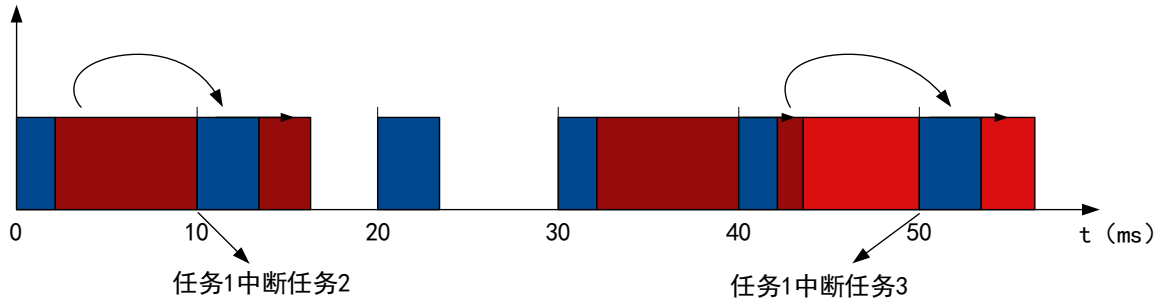


示例：

假设有 3 个不同的任务，分别对应三种不同的优先等级，具体分配如下：

- ◆ 任务 1 具有优先级 0 和循环时间 10ms；
- ◆ 任务 2 具有优先级 1 和循环时间 30ms；
- ◆ 任务 3 具有优先级 2 和循环时间 40ms。

在控制器内部，各任务的时序关系如图所示，具体说明如下：  
 0~10ms：先执行任务 1（优先级最高），如在本周期内已将程序执行完，剩余时间执行任务 2 程序。但是如果此时任务 2 没有被完全执行完，但时间已经到了第 10ms 时，由于任务 1 是每 10ms 执行一次的且优先级更高，此时将会打断任务 2 的执行。  
 10~20ms：先将任务 1 的程序执行完毕，如有剩余时间，再执行上个周期为完成的任务 2。  
 20~30ms：由于任务 2 是每 30ms 执行一次的，在 10~20ms 之间任务 2 已经全部执行完毕，此时不需要再执行任务 2，只需将优先级最高的任务 1 执行一次即可。  
 30~40ms：与之前类似。  
 40~50ms：此时出现了任务 3，任务 3 的优先级更低，所以只有在确保任务 2 彻底执行完后，才能执行任务 3。



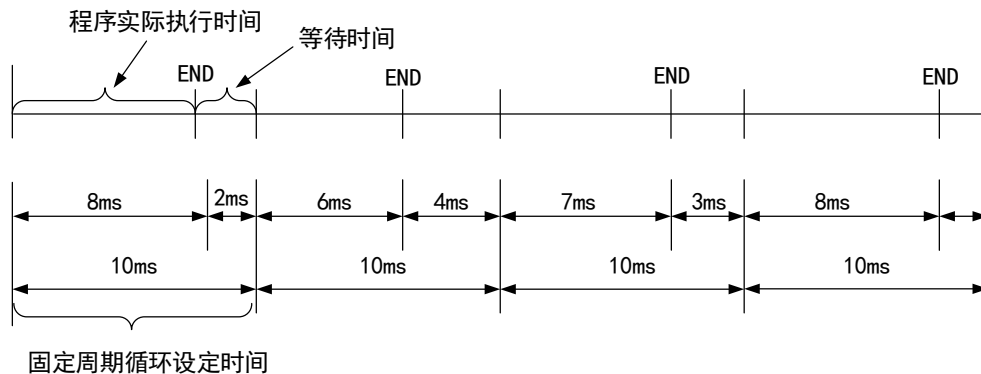
## 2) 任务的执行类型

针对每个独立的任务可以对其进行执行的类型编辑及配置。包括固定周期循环、事件触发、外部触发、自由运行和状态触发 5 种类型。

### (1) 固定周期循环

根据程序中所使用的指令执行与否，程序的处理时间会有所不同，所以实际执行时间在每个扫描周期都发生不同的变化，执行时间有长有短。通过使用固定周期循环方式，能保持一定的循环时间反复执行程序。即使程序的执行时间发生变化，也可以保持一定的刷新闻隔时间。在这里，也推荐大家优先选择固定周期循环任务启动方式。

例如，假设将程序对应的任务设定为固定周期循环方式，间隔时间设定为 10ms 时，实际程序执行的时序图如下图所示。

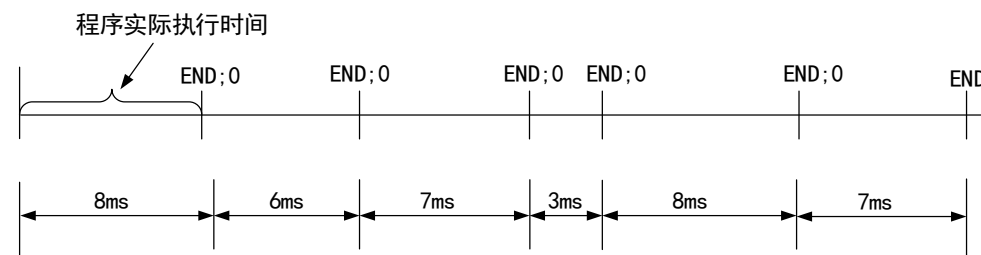


如果程序实际执行时间在规定的固定周期循环设定时间内执行完，则空余时间用作等待。如应用中还有优先级较低的任务未被执行，则剩下的等待时间用来执行低优先级的任务。

### (2) 自由运行

程序一开始运行任务就被处理，一个运行周期结束后任务将在下一个循环中被自动重新启动。

不受程序扫描周期（间隔时间）的影响。即确保每次执行完程序的最后一条指令后才进入下一个循环周期。否则不会结束该程序周期。



该执行方式因为没有固定的任务时间，所以每次执行的时间可能都不一样。故不能保证程序的实时性，在实际的应用中选用此方式的场合较少。

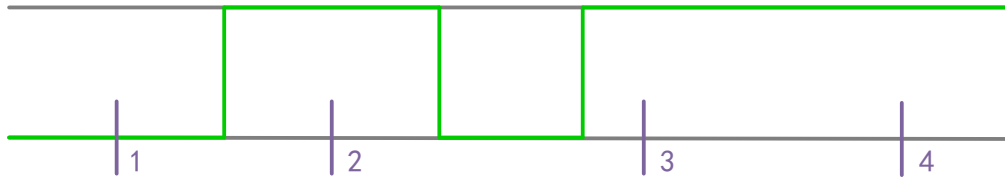
(3) 事件触发

如果事件区域的变量得到一个上升沿，任务开始。

(4) 状态触发

如果事件区域的变量为 TRUE，任务开始。

下图中针对事件触发和状态触发分别进行了比较，绿色实线为两种触发方式选择的布尔变量状态，下表为比较的结果。



任务输入触发信号

状态触发方式与事件触发功能类似，区别在于状态触发的触发变量只要为 TRUE 程序就执行，为 FALSE 则不执行。而事件触发只采集触发变量的上升沿有效信号。

在采样点 1-4（紫色）不同类型的任务展示了不同的反应。这个具体的事件为 TRUE 完成了状态驱动任务的条件，然而一个事件驱动任务需要事件从 FALSE 变为 TRUE。如果任务计划的采样频率过低，事件的上升沿可能检测不到。

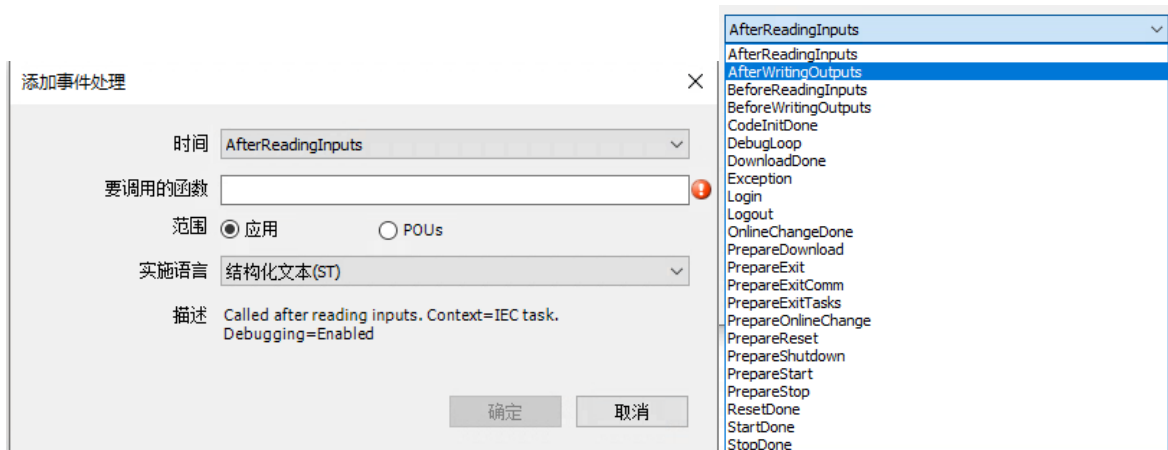
执行点	1	2	3	4
事件触发 (Event)	不执行	执行	执行	执行
状态触发 (Status)	不执行	执行	不执行	不执行

(5) 系统事件

用户可选择的系统事件是根据实际的硬件目标系统而定的，由目标系统对应的库文件提供相应的系统事件，所以不同的目标硬件设备对应的系统事件可能会不同。但通常来说，通用的系统事件有：停止，开始，登入，改变等。在任务配置中，可以对任务配置中的系统事件进行设置。



用户可以通过鼠标选择“任务配置”->“系统事件”进入添加事件处理界面，单击“添加事件处理”按钮即可进行添加系统事件。支持用户可下拉选择时间，分别如下图所示：



### (6) 外部中断

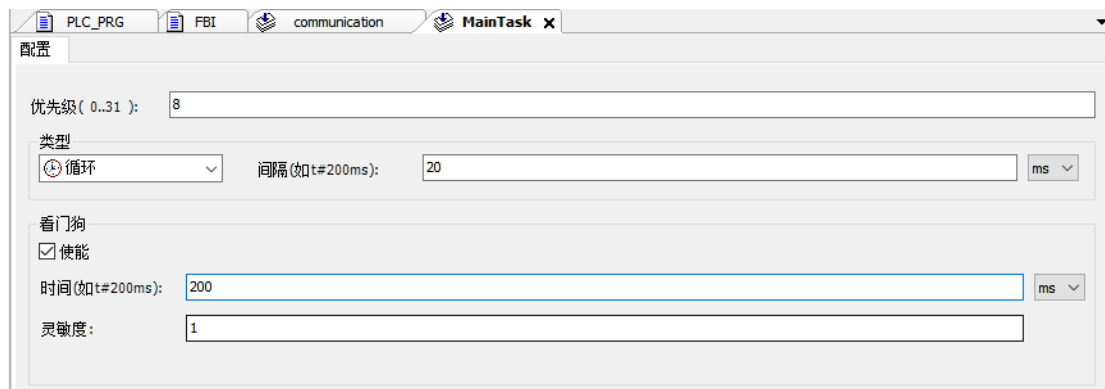
如果事件区域的变量得到一个外部中断信号 X 的上升沿或者下降沿，任务开始。

输入端子 X 可以作为外部中断的输入用，每一输入端对应于一个外部中断，也可指定上升沿或下降沿或上升下降沿作为触发条件。

### (7) 看门狗

看门狗是一种控制器硬件式的计时设备，XS Studio 内可以通过“任务配置”对其进行使能，默认不使用看门狗功能。

看门狗的主要功能是监控程序执行时出现的异常或内部时钟发生的故障。如当系统出现死机或当程序进入死循环时，看门狗计时器就会对系统发出重置信号或停止 PLC 当前运行的程序。我们可以形象的将它理解为一只有主人定时的去给它喂食，如果超过规定的时间没有喂，则他马上就会饿。要配置看门狗，必须定义两个参数，时间和灵敏度，看门狗的配置如图所示：



#### ① 时间

XS Studio 针对每个任务可以配置独立的看门狗。如果目标硬件支持长看门狗时间设置，则可以设置上限和下限。默认的看门狗时间单位为毫秒 (ms)。如果程序执行周期超过看门狗触发时间，将激活看门狗功能，并将中止当前任务。

#### ② 灵敏度

“灵敏度”用于定义必须在控制器检测到应用程序错误之前发生的任务看门狗例外数。默认为 1。

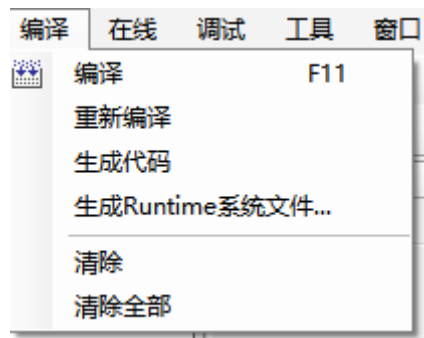
最终的看门狗触发时间=时间×灵敏度。如果程序实际执行时间超过看门狗触发时间，则激活看门狗。例如，时间为 10ms，灵敏度设为 5，则看门狗触发时间为 50ms，一旦任务的执行时间超过 50ms，则立即激活看门狗并将任务中止。

## 2-2-3. 程序下载/读取

### 2-2-3-1. 编译

程序编写完成后，在下载之前需要对程序进行编译。编译命令对编写的程序进行语法检查，并且只编译添加到任务中的程序。如果创建的 POU 没有添加到任务中，编译命令不对该 POU 进行语法检查。

编译指令不生成任何代码，只针对 POU 的语法进行检查。直接执行设备登录命令，系统也将默认执行编译指令（等同于先手动执行编译命令），在编译检查没有语法错误后执行连接登录指令。同样，在编译中不对没有添加到任务中的 POU 进行语法检查。执行登录命令同时会生成代码。



1) 编译：对当前的应用进行编译。

2) 重新编译：如果需要对已经编译过的应用再次编译，可以通过重新编译进行。

- 3) 生成代码：执行此命令后生成当前应用的机器代码，执行登录命令时，生成代码默认执行。
- 4) 清除：删除当前应用的编译信息，如果再次登录设备时需要重新生成编译信息。
- 5) 清除全部：删除工程中所有的编译信息。

执行编译命令后可以看到，添加到任务里面的“PLC\_PRG”显示为蓝色，没有添加到任务里面的则显示为灰色。编译指令不会对灰色的 POU 进行语法检查，因为该程序单元没有处于活动状态，编译指令只针对于处于活动状态的 POU 进行语法检查。如果在编译的过程中发现需要运行的程序单元显示为灰色，可以检查该程序单元是否已经被成功的添加到了所需要运行的任务当中。

编译命令执行完成之后可以在消息栏看到编译生成的信息，其中可以看到编译的程序是否有错误或者警告，以及错误和警告的数量。如果有错误和警告产生可以通过消息窗口进行查看和查找，根据提示信息对程序进行修改。



## 2-2-3-2. 登录下载

### 1、登录

登陆使应用程序与目标设备建立起连接，并进入在线状态。能正确登陆的前提条件是要正确配置设备的通讯设置并且应用程序必须是无编译错误的。

对于以当前活动应用登录，生成的代码必须没有错误和设备通信设置必须配置正确。登入后，系统会自动选择程序下载。

### 2、下载

下载命令，在在线模式下有效。它包括对当前应用程序的编译和生成目标代码两部分。除了语法检查（编译处理）外，还生成应用目标代码并装载到 PLC。



#### 1) 登录-在线修改 (Login with online change)

用户选择此选项后，项目的更改部分被装载到控制器中。使用“登录-在线修改”操作，可以防止控制器进入 STOP 状态。建议同时勾选“更新自动启动程序”，防止前面修改程序的内存导致数据程序等丢失。

**注意：**

- ① 用户之前至少执行过一次完整的下载。
- ② 指针数据会更新最近一个周期的数值，如果改变了原变量的数据类型，无法确保数据的准确性，此时，需要重新分配指针数据。

#### 2) 登录并下载 (Login with download)

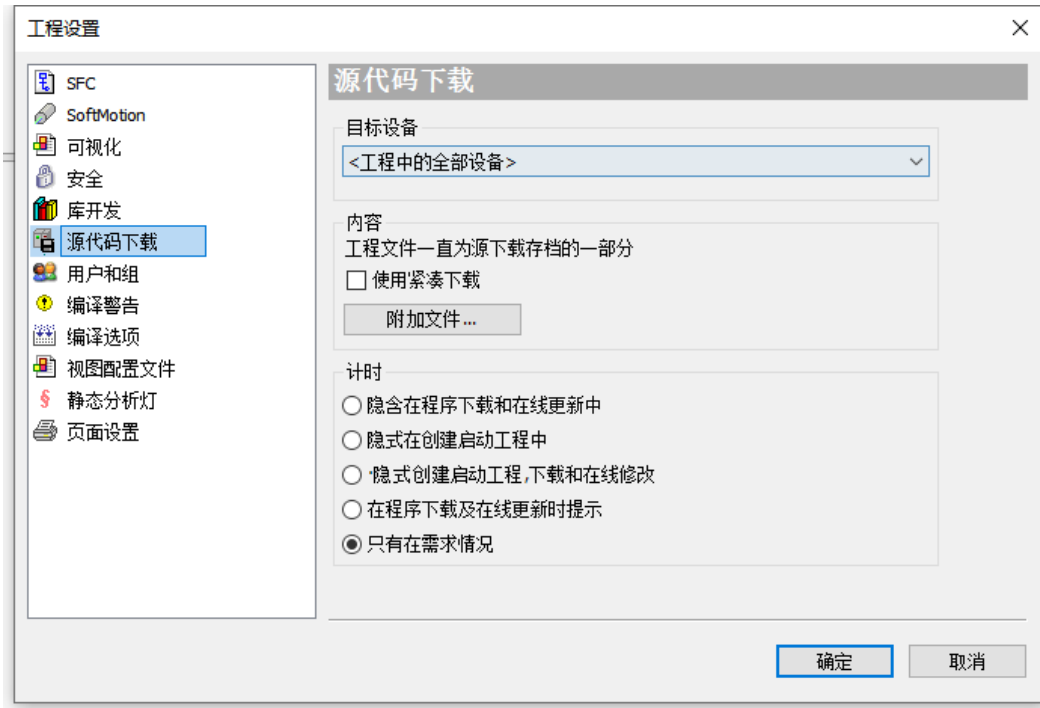
选择“登录并下载”后，将整个项目重新装载到控制器中。与“登录-在线修改”最大的区别是当完成下载后，控制器会停留在 STOP 模式，等待用户发送 RUN 指令，或重启控制器程序才会运行。

## 3) 登录-不做任何修改

登录时，不更改上次装载到控制器中的程序。

## 2-2-3-3. 源代码下载

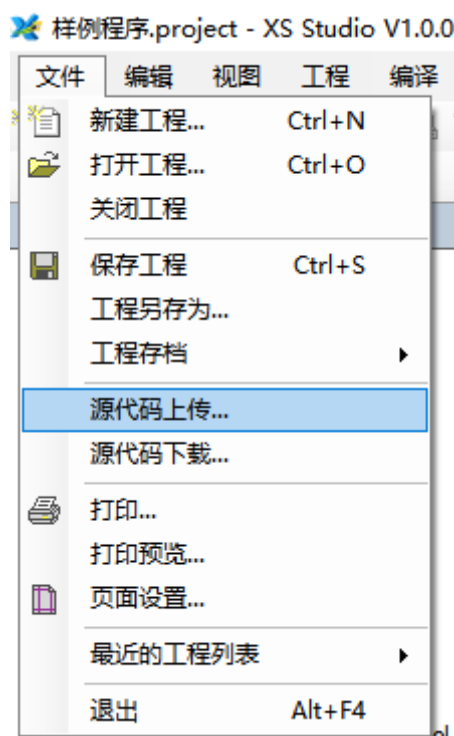
出于对程序员源代码的保护，下载时默认不自动下载源代码，如需下载源代码则需要手动设置，点击“在线”→“下载源代码到连接设备上”。用户也可以在“工程”→“工程设置”→“源代码下载”→“计时”选项中对该属性进行设置。



## 2-2-3-4. 读取程序

在“文件”>“源代码上传”命令打开一个设备选择对话框，用户选择连接 PLC 的网络路径，点击“确定”按钮。如果用户选择的路径下已经存在存档文件，给出是否覆盖提示。

这里需要注意的是，在读取程序之前需要确保在前一次的下载过程中已经做过“下载源代码到连接设备上”。否则不能读出控制器中的数据。



## 2-2-4. 程序调试

### 2-2-4-1. 复位

XS Studio 程序复位有如下三种方式，在“在线”菜单中进行选择。



#### 1、热复位

热复位后，除了保持型变量（PERSISTENT 和 RETAIN 变量）外，其它当前的应用的变量都被重新初始化。如果设置了初始值的变量，热复位后这些变量值还原为设定的初始值，否则变量都会被置为标准初始值 0。

#### 2、冷复位

与“热复位”不同的是，冷复位命令不但将普通变量的值设置为当前活动应用程序的初始值，而且将保持型变量（RETAIN 变量）的值也设置为初始值 0。保持型变量（PERSISTENT）保持不变。

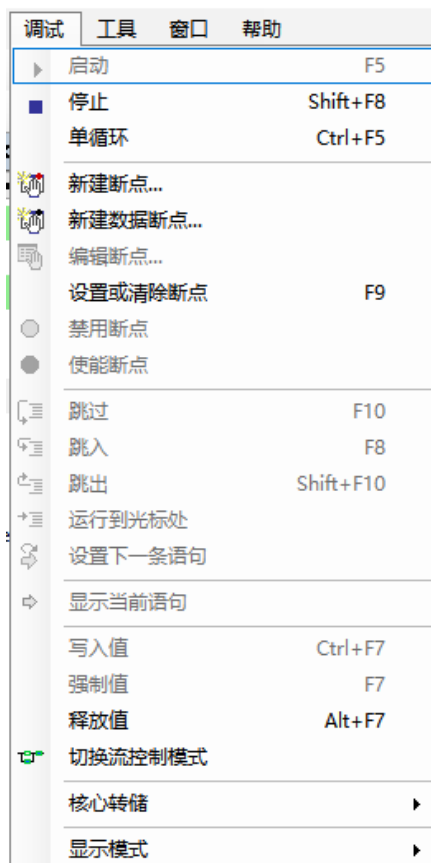
#### 3、初始复位

当在设备树中选择一个可编程设备时，无论是在离线还是在在线状态下都可以使用此命令。使用此命令将使设备复位到初始状态，即设备中的任何应用、引导工程和剩余变量都将被清除。

由于所有工程信息被清除，重新登入后，需要重新“下载”程序，并“启动”运行。

## 2-2-4-2. 程序调试

在 XS Studio 中“调试”菜单的视图如图所示。主要的操作涉及断点设置及单循环。



### 1、断点

断点是程序内处理停止的功能，当程序停止后，程序研发人员可以借此观察程序到断点位置时其变量及 I/O 等相关变量的内容，有助于深入了解程序运行的机制，发现及排除程序故障。

在 XS Studio 中所有的编程语言都可以设置断点。在文本编辑器 ST 语言中，断点设置在行上；在 FBD 和 LD 编辑器中设置在网络号上；而在 SFC 中，设置在步上。

### 2、单步执行

设置断点后，可以进行单步执行程序，该功能可以让程序一步一步的运行，方便编程人员进行调试，以便检查程序中的逻辑错误。

#### 1) 跳过

该命令会执行程序中当前的一条指令，执行完停止。在不调用 POU 时，跳过和跳入命令效果是一样的。但如果是调用 POU，那么跳过不会进入这个 POU，而是把这个 POU 调用当作完整的一步，一次执行完；跳入则会进入该 POU。如果用的是 SFC 语言，那么跳过会把一个动作当作完整的一步，一次执行完。如果想要实现跳到被调用的 POU 中单步调试，就必须用跳入。

#### 2) 跳入

执行时，当前的指令位置由一个黄色箭头表示。如果当前指令没有调用 POU，那么使用该命令和使用跳过命令的效果一样。

#### 3) 跳出

当正在一个 POU 中单步调试时，用跳出会把这个 POU 剩下的指令一次执行完，然后返回到这个 POU 被调用处的下一条指令。所以，如果是一层一层向下调用 POU，那么跳出就会一层一层向上返回，每次返回一层。如果程序中不包含任何 POU 调用，那么跳出无法向上层返回，就会返回到程序的开头处。

### 3、单循环

在“调试”一>中选择“单循环”，这样程序进行单步运行。即按一次运行，程序执行一个周期即停止并等待下一次运行指令。



## 2-2-5. 仿真

### ■ 离线仿真

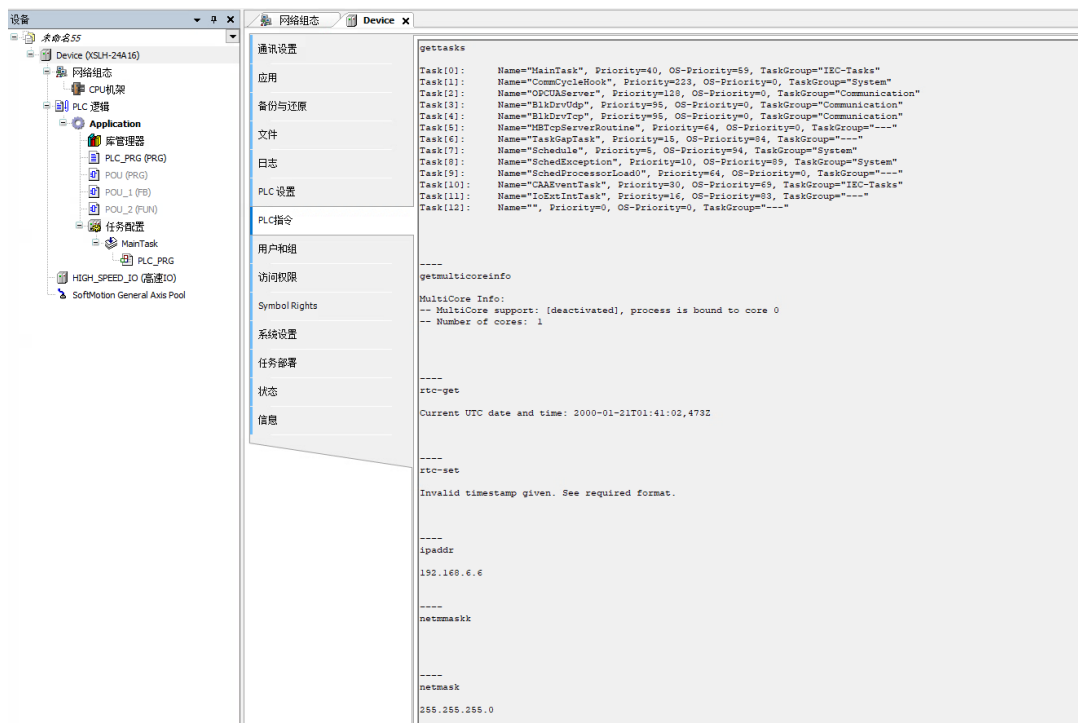
在菜单“在线”中选择“仿真”，便进入了仿真模式下的程序运行过程。确认选项“仿真”前已打上“√”后，编译程序，没有错误后，即可进入仿真模式。



## 2-2-6. PLC 脚本功能

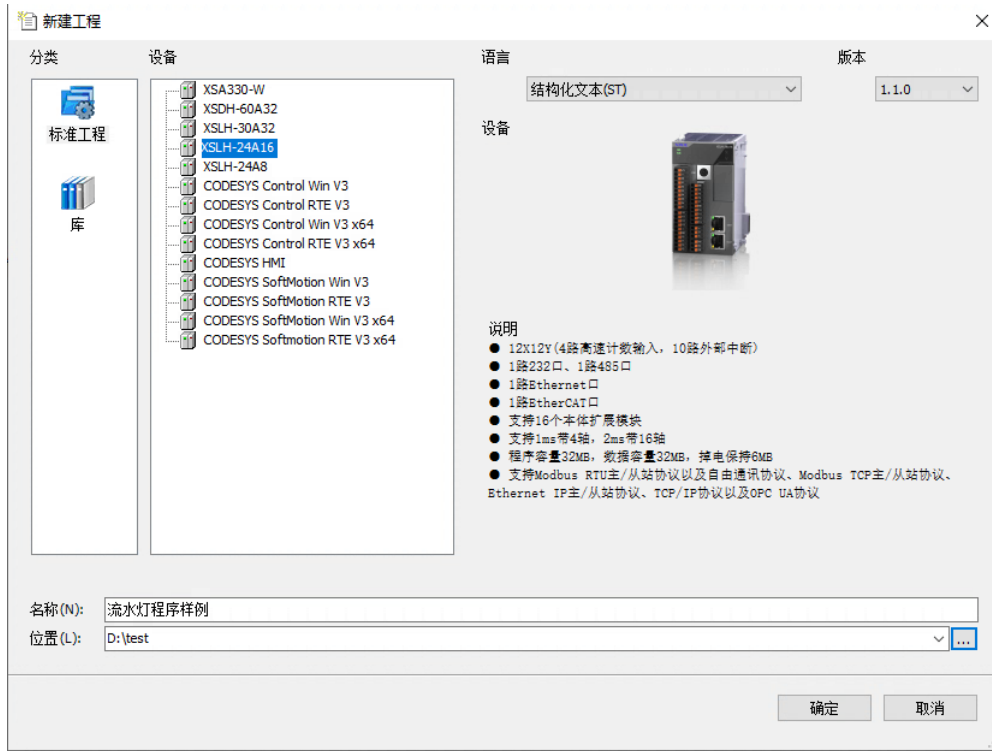
该 PLC 脚本是一个基于文本的控制监视（终端）。该功能目前仅支持 ARM 机型从控制器中得到的具有特定信息的命令以一个输入行进行输入并且作为一个字符串发送到控制器，返回相关的字符串在浏览窗口中的结果显示，这个功能用于诊断和调试目的。

鼠标双击选中“Device”，在右边视图中找到“PLC 指令”，在下方的命令输入框中输入相应指令即可。输入“？”，按回车键即可显示所有该控制器支持的命令。

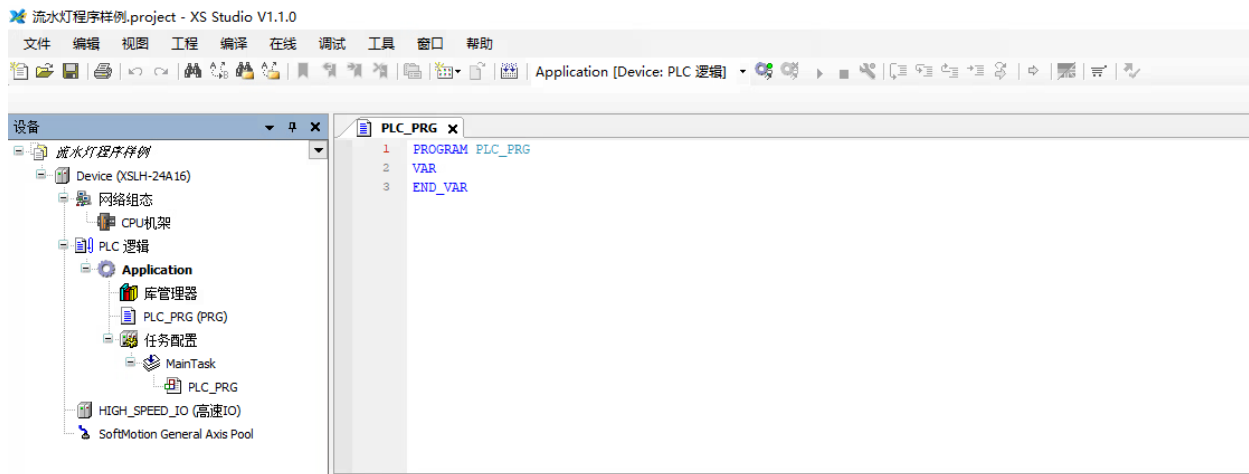


## 2-3. XS Studio 编写流水灯程序样例

### 1、新建工程：



### 2、编写样例程序：

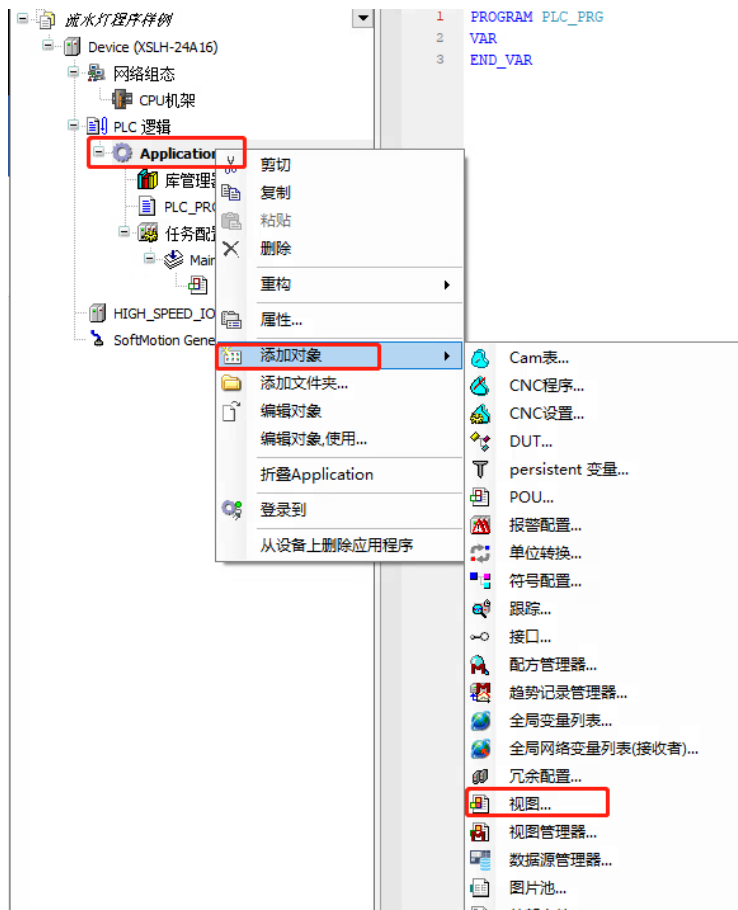


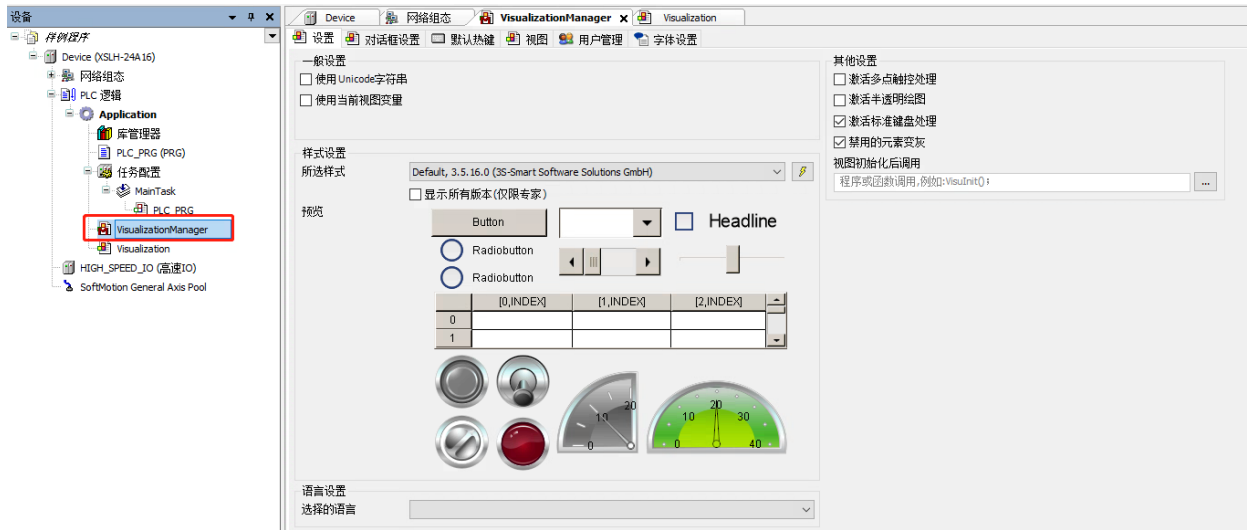
```

Device | PLC_PRG x
1  PROGRAM PLC_PRG
2  VAR
3      switch:BOOL;
4      redlight:BOOL;
5      greenlight:BOOL;
6      yellowlight:BOOL;
7      nored:BOOL;
8      noyellow:BOOL;
9      nogreen:BOOL;
10     TON1:TON;
11     TON2:TON;

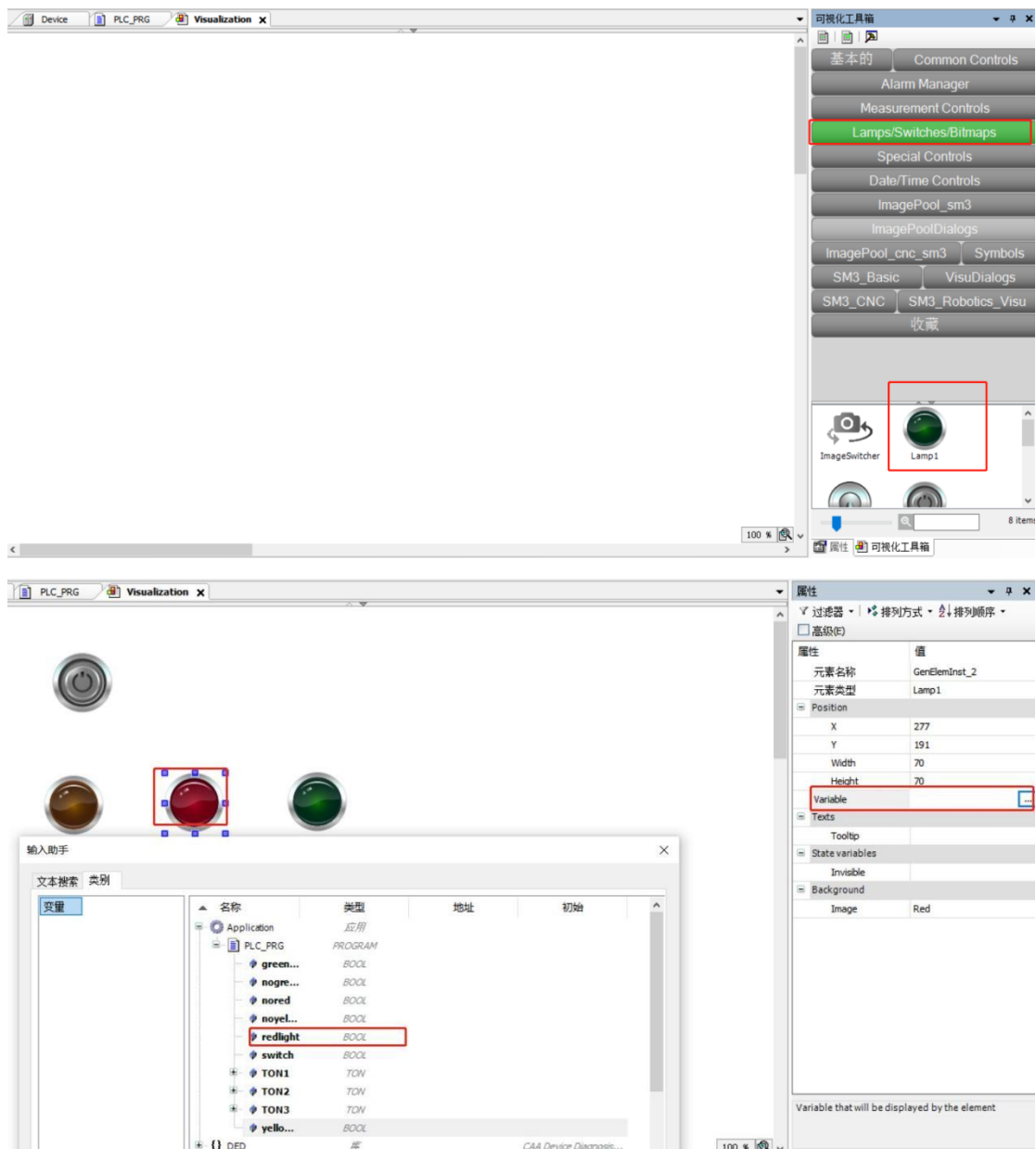
1  IF switch THEN
2      greenlight:=1;
3      switch:=0;
4  END_IF
5  TON1(IN:=greenlight , PT:=T#1S , Q=> nogreen, ET=> );
6  IF nogreen THEN
7      greenlight:=0;
8      redlight:=1;
9  END_IF
10 TON2(IN:=redlight , PT:=T#1S , Q=>nored , ET=> );
11 IF nored THEN
12     redlight:=0;
13     yellowlight:=1;
14 END_IF
15 TON3(IN:=yellowlight , PT:=T#1S , Q=>noyellow , ET=> );
16 IF noyellow THEN
17     switch:=1;
18     yellowlight:=0;
19 END_IF
    
```

3、“application” – “添加对象” – “视图”；

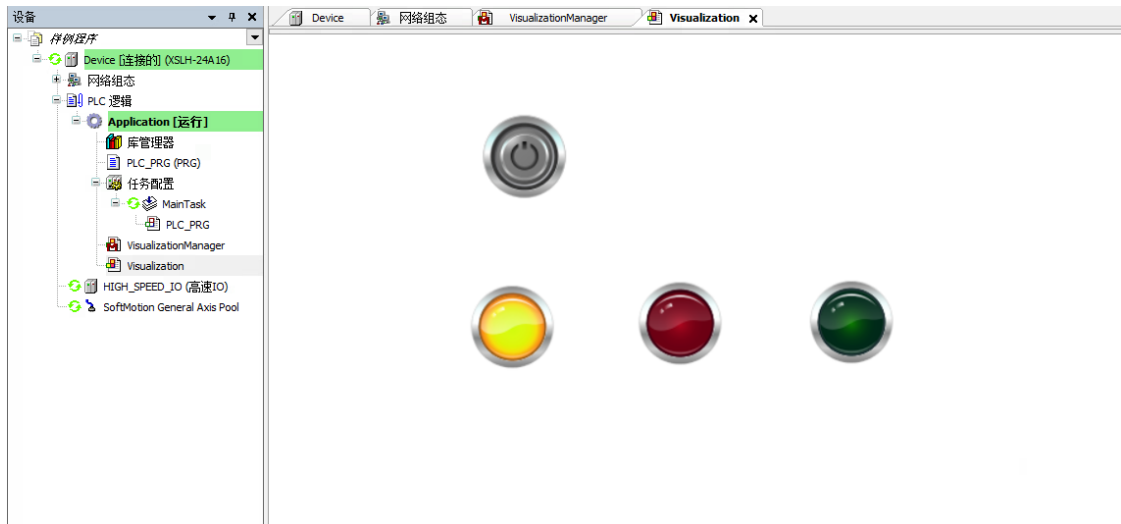




4、添加可视化对象，映射变量；



5、登录设备，启动。



## 2-4. 如何登录设备

### 2-4-1. 登录设备的必备条件和操作简介

“登录设备”是指在 PC 上运行的 XS Studio 与 XS 系列控制器建立通讯，从而进行用户程序的下载、监控调试等操作。

PC 与 XS 系列控制器之间可以通过网线直连；也可以通过路由器、集线器连接 PLC，此时，一台 PC 可以与多台 XS 系列控制器联机，也可以多台 PC 访问同一个 XS 系列控制器。

PC 与 XS 系列控制器两者的 IP 地址必须在同一个网段且网关正常工作才能登录，否则 XS Studio 将无法扫描到 XS 系列控制器。比如，XSDH 的出厂默认 IP 地址为 192.168.6.6，若 PC 的 IP 地址为 192.168.6.xxx（这里 xxx 范围为 1~254，但不得与 XSDH 的 IP 相同），那么 XS Studio 就可以扫描到 XSDH 并与其进行连接，进行用户程序下载、监控调试等。若 XSDH 的 IP 被人为修改过，其地址与 PC 不在同一 IP 网段，二者将无法建立通讯，若客户知道其 IP 地址可以将 PC 的 IP 修改为与 XSDH 同一网段再进行连接。若用户此时不知道该设备的 IP 地址，此时需要将 XSDH 的 IP 地址恢复为出厂地址 192.168.6.6，再将 PC 本机的地址修改为 192.168.6.xxx 进行连接。

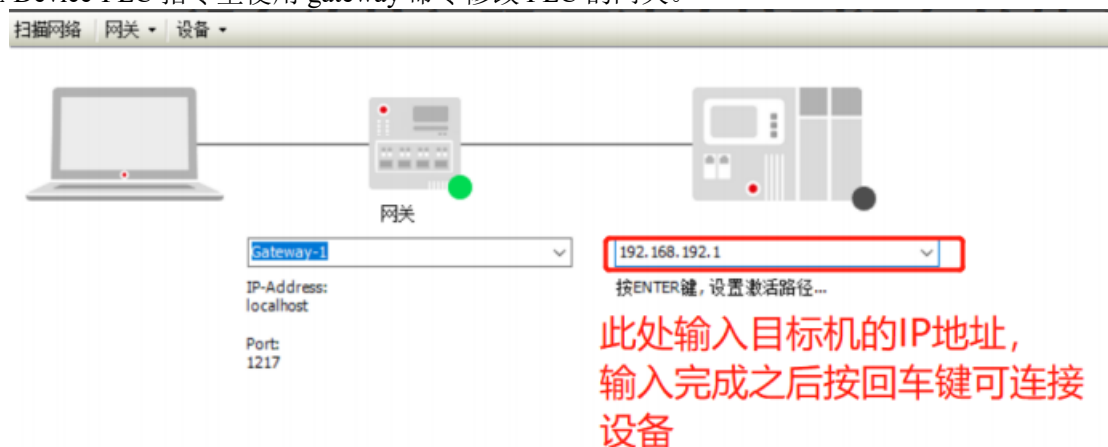
### 2-4-2. 扫描不到设备或者连接不上设备的解决方法

#### ■ XS 系列

(1) 不记得 IP: PLC 断电，只把拨码 1 置 ON，再重新上电恢复默认 IP 为 192.168.6.6 再联机

(2) 若 IP 确认无误还是无法连接设备，可能是 PLC 程序死机(程序里有死循环或超出 PLC 的负载能力)，此时可将拨码 2 置 ON(上电不加载用户程序)，再次扫描连接设备；若能扫描连接，此时下载一个空程序，进行抹除异常的程序之后，再恢复拨码状态，同时检查异常程序(是否有超长循环或任务周期时间过小)。

(3) 修改了 IP 网段，但 PLC 的网关没有同时设置也会连不上，可以如下图直接输入 IP 地址联机，然后在 Device-PLC 指令里使用 gateway 命令修改 PLC 的网关。



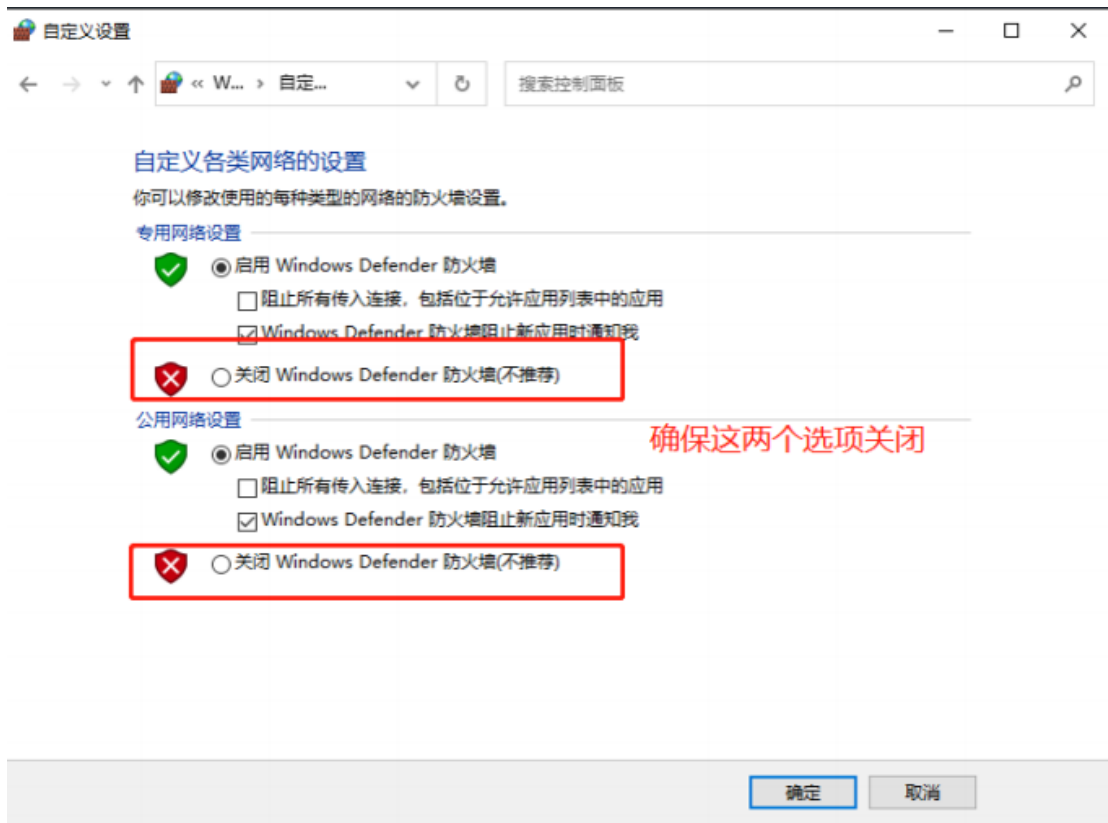
(4) 确认上位机工程设备型号与目标设备一致，否则也会无法扫描到设备。

(5) 若以上步骤还是无法连接设备，请反馈技术支持，并同时提供出现此问题之前做了什么操作，RUN 和 ERR 灯的状态。

#### ■ XSA 系列

(1) 确认上位机工程设备与目标机设备一致。

(2) 连接显示器确认 IP、子网掩码、网关无误，确认 PLC 与电脑双方 IP 是否为同一网段，能否 ping 通；若 IP 无误，但是无法 ping 通，则可能是防火墙的问题，需关闭工控机防火墙之后再行连接



若能 ping 通，则直接输入 IP 地址进行连接，排除子网掩码问题。

(3) 若能 ping 通且输入 IP 地址也无法连接设备，此时有以下两种可能：

① PLC 程序死机，删除 D:\CODESYS\Application 文件夹(删除用户程序)，然后重启工控机。

② 设备信息丢失，可通过 RTE 配置界面查看目标 ID，该系列产品的高 16 位 ID 为 1707。可联系技术支持进行恢复处理。



注：此步骤查询，需外接显示器，否则无法处理。

## 3. 网络配置

XS 系列 PLC 标准控制器拥有强大的通讯功能，支持 Modbus、自由格式、Modbus TCP、CANopen、Ethernet/IP、OPC UA、信捷 Modbus TCP 协议。本章就上述 7 种通讯协议进行展开介绍。

3. 网络配置	34
3-1. 设备组态	35
3-1-1. 网络组态	35
3-1-2. 硬件组态	40
3-1-3. 设备树操作	41
3-1-4. 组态编辑错误定位	42
3-2. MODBUS 通讯	43
3-2-1. MODBUS 主站配置	43
3-2-2. MODBUS 从站配置	45
3-2-3. MODBUS 通讯帧格式说明	46
3-3. 串口自由协议通讯	49
3-3-1. 参数配置	49
3-3-2. 应用举例	50
3-4. ModbusTCP 通讯	51
3-4-1. MODBUS TCP 主站配置	51
3-4-2. MODBUS TCP 从站配置	52
3-4-3. MODBUS TCP 常见故障	53
3-4-4. MODBUS TCP 通讯帧格式说明	53
3-5. CANopen 网络	55
3-5-1. 参数配置	56
3-5-2. 应用举例	63
3-6. EtherNet/IP 通讯	65
3-6-1. EtherNet/IP 作为从站的样例	66
3-6-2. EtherNet/IP 作为主站的样例	68
3-7. OPC UA 通讯	72
3-7-1. 通讯概述	72
3-7-2. 参数配置	72
3-7-3. OPC UA 样例	74
3-8. 信捷 Modbus TCP 协议	83
3-8-1. 参数配置	83
3-8-2. 触摸屏设置	84



## 3-1. 设备组态

组态是用户进行 PLC 编程的第一步，可通过“网络组态”和“硬件组态”界面来添加 PLC 可支持的功能。

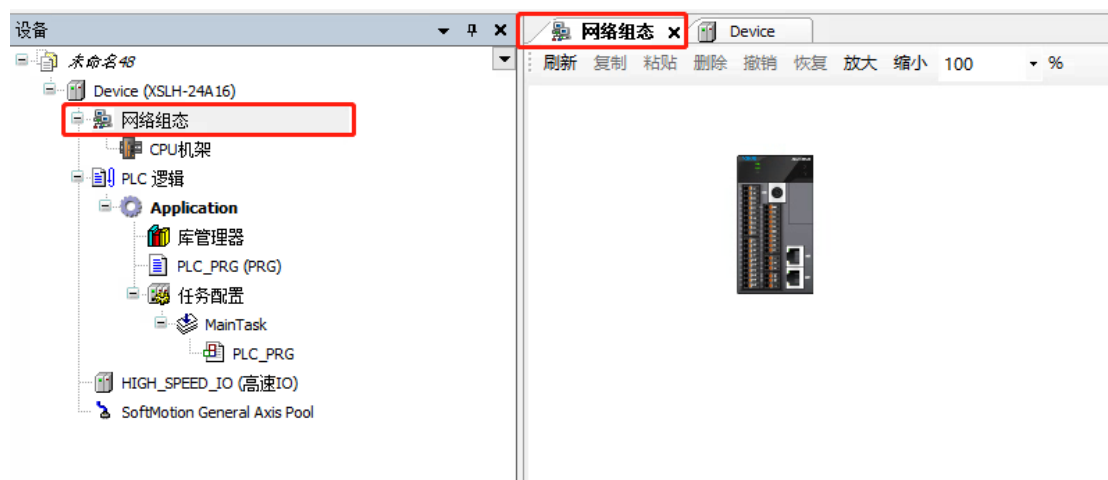
网络组态：是组态设备的入口，可通过使能窗口及右侧网络设备连接列表对主从站设备进行布局，并以总线型的网络拓扑视角在界面内显示。

硬件组态：可添加中型 PLC 的扩展 IO 模块。

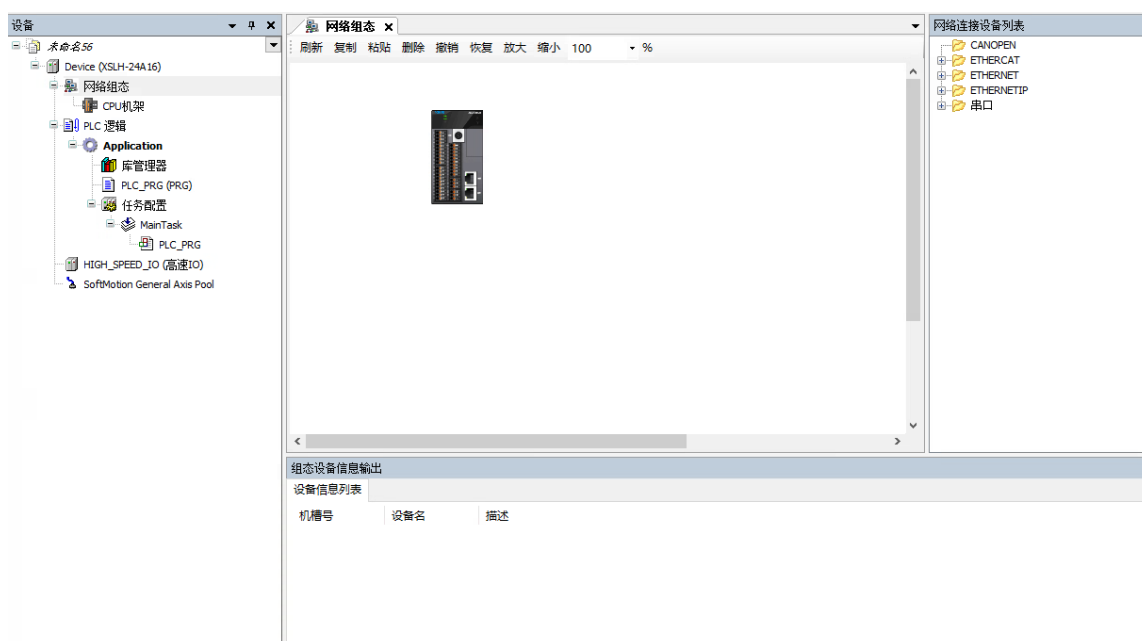
### 3-1-1. 网络组态

#### 1、打开组态界面

新建 XS Studio 工程后，可通过双击软件左侧设备树内的“网络组态”节点打开组态界面。



双击“网络组态”节点后可打开网络组态界面、右侧网络连接设备列表以及组态设备信息输出界面。网络组态界面显示用户工程当前所使用的 PLC 设备，而网络连接设备列表显示当前 PLC 所支持的所有设备，设备信息输出界面则显示当前组态界面设备的名称及相关描述信息。



**注：**

① 网络连接设备列表默认为折叠状态；

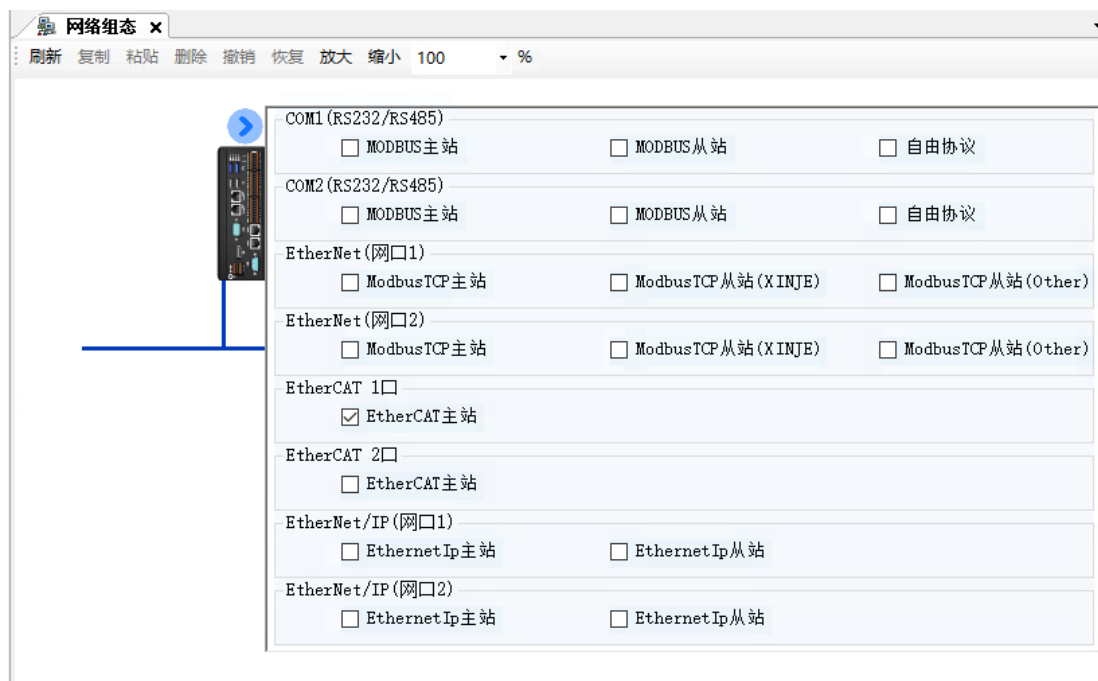
② 默认情况下，设备信息列表为空。选中网络组态界面内的设备时会动态显示当前选中设备的相关信息描述。

#### 2、设置 PLC 作为主站或从站设备

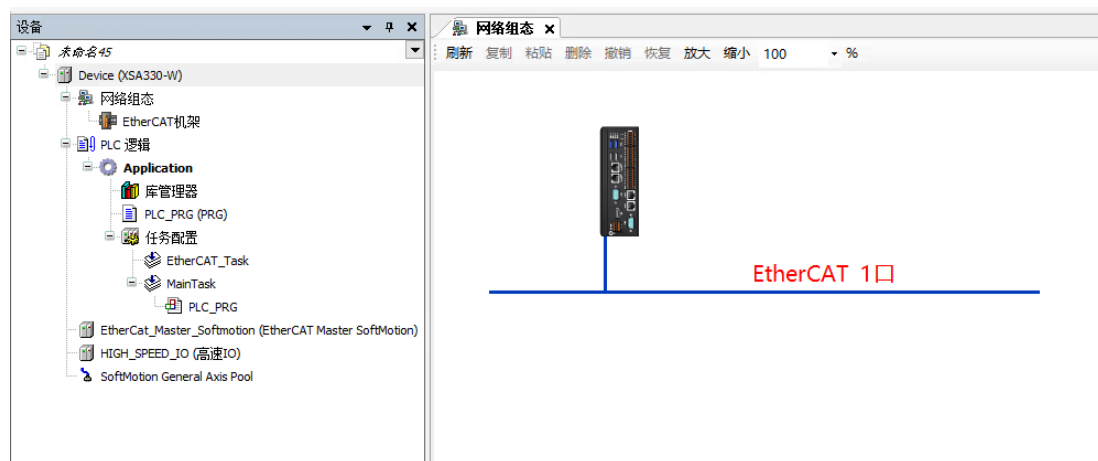
##### (1) 使能主站

单击网络组态中的 PLC 本体设备，会显示 PLC 支持的主/从站使能窗口。如下图所示，根据应用需要

选中窗口中的复选框按钮即可使能 CPU 所支持的主/从站功能。以 XSA330-W 机型为例：



使能 CPU 的主站功能时，都会显示总线型的拓扑界面，并在左侧生成对应的设备节点。下图所示为使能 EtherCAT 主站：



#### ■ 失能设备

在原来勾选的基础上再次点击，则会有弹框询问是否确定移除当前设备，用户可选择确定或取消当前失能操作。

#### ■ 互斥规则

- ◆ COM 口：在对已有硬件接口上面做协议修改时，会有弹窗提示，点击确定则替换为新添加的设备，点击取消则取消当前操作；
- ◆ EtherNet: ModbusTCP 信捷从站与官方从站互斥，同时点击勾选会有弹框询问；
- ◆ EtherCAT: 无互斥；
- ◆ CANopen: 无互斥；
- ◆ EtherNet/IP: EtherNet/IP 主站/从站，可以同时使用，无互斥。

#### (2) 添加从站

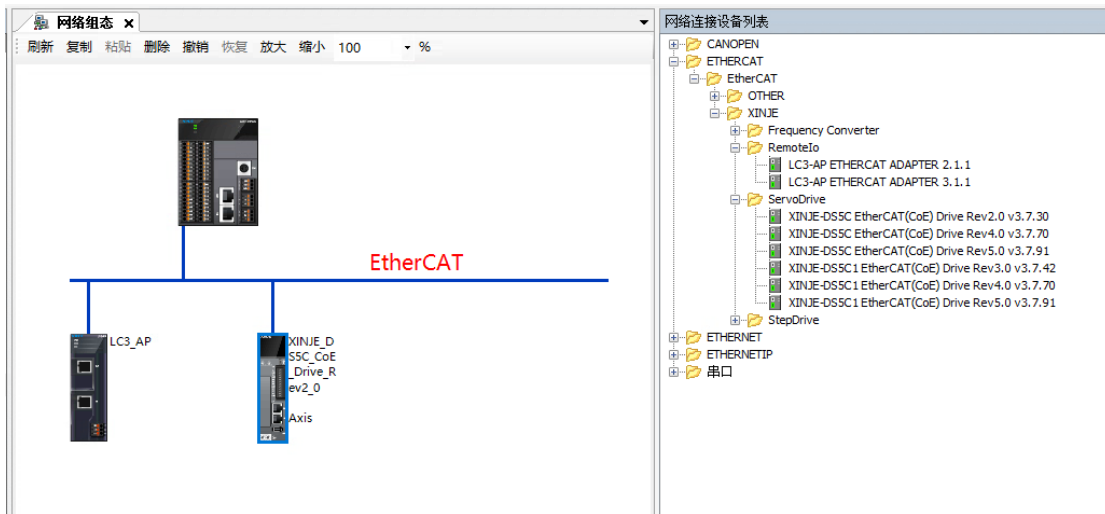
使能 CPU 中的某个特定的主站后，即可添加其相对应的总线下的从站设备，添加从站设备有三种方式（以 EtherCAT 总线为例）：

① 先使能 EtherCAT 主站功能，然后从网络连接设备列表中的“EtherCAT 口”节点下选中一个从站设备节点，按住鼠标左键不松拖拽到网络组态界面中。

② 先使能 EtherCAT 主站功能，然后网络连接设备列表中的“EtherCAT 口”节点下双击一个从站设备节点即可。

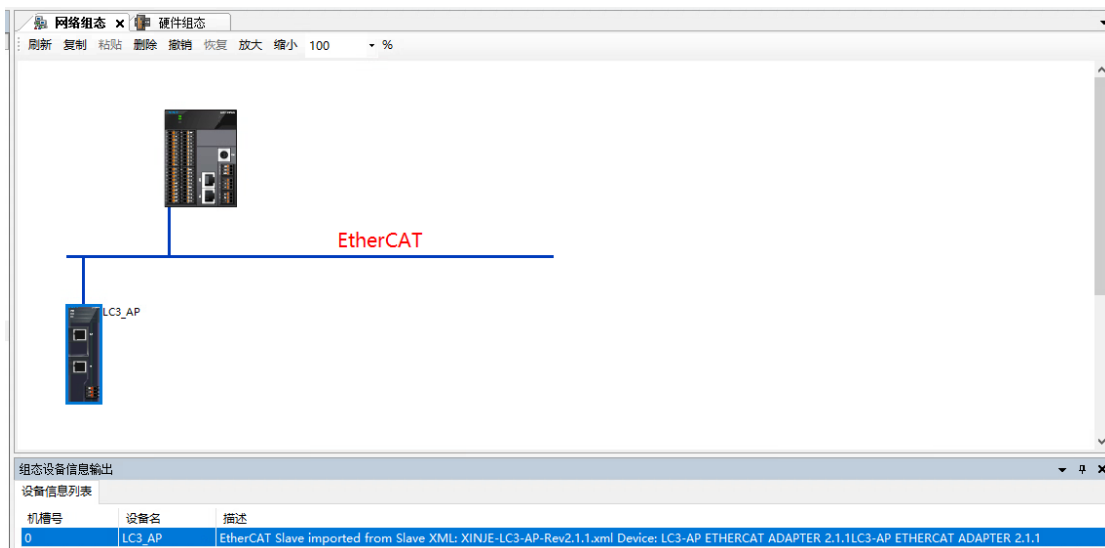
③ 直接在网络设备列表中的“EtherCAT 口”节点下双击一个从站设备节点添加，此种方式会默认先使能 CPU 内部的特定主站功能。

若添加的从站为 EtherCAT 远程 IO 设备，需要配置从站后的 IO 模块，可双击设备进入“EtherCAT 机架”界面进行配置。添加从站后的网络组态界面如下所示：



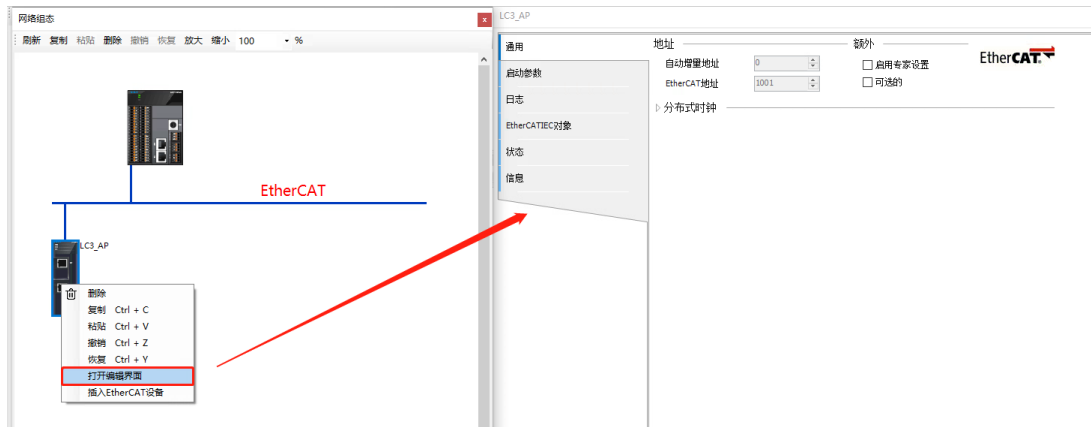
(3) 查看设备基本信息

选中网络组态界面内的设备后，可在“组态设备信息输出”框内的设备信息列表内查看设备对应的基本信息。



(4) 打开设备配置界面

右键单击网络组态界面内的从站设备，通过“打开编辑界面”菜单项进入设备的参数配置界面。以 EtherCAT 为例，如图所示：

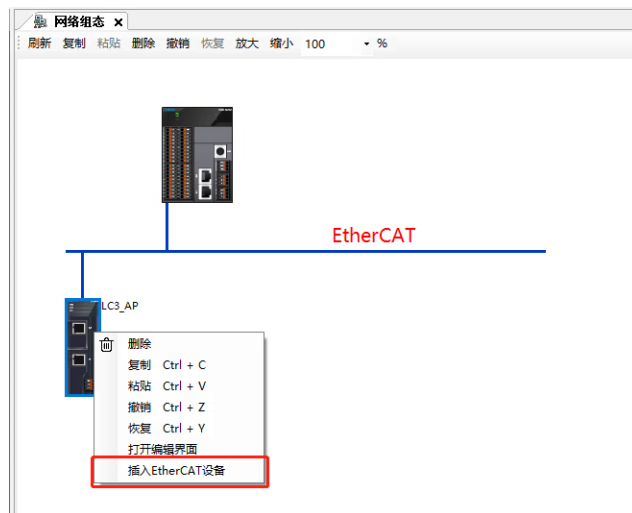


**注:**

- ① 网络组态界面内双击 EtherCAT 或 CANopen 设备图标可跳转到该设备模块对应的硬件组态界面，点击其他的设备图标则跳转到模块参数配置界面；
- ② 双击扩展模块或从站后的 IO 模块，可打开模块配置界面。

(5) 插入设备

右键单击网络组态内的从站设备，可通过“插入 XX 设备”菜单项打开插入设备弹框添加从站设备。以插入 EtherCAT 设备为例，如下图所示：



组态设备可进行复制、粘贴、删除等操作，具体情况请参见组态的基础操作说明。

(6) 设备信息列表

设备信息列表通过软件菜单栏“视图”下的“组态设备信息列表视图”打开。设备信息列表显示组态设备的基本信息，主要包括机槽号、设备名及对应的信息描述。若设备信息列表处于隐藏状态，则需要手动点击 组态设备信息输出 打开列表界面。

机槽号	设备名	描述
0	LC3_AP	EtherCAT Slave imported from Slave XML: XINJE-LC3-AP-Rev2.1.1.xml Device: LC3-AP ETHERCAT ADAPTER 2.1.LC3-AP ETHERCAT ADAPTER 2.1.1
1	XL_E16X_V3	扩展16点直流输入,供电电源DC24V,NPN输入
2	XL_E32X_V3	扩展32点直流输入,供电电源DC24V,NPN 输入
3	XL_E32X_V4	扩展32点直流输入,供电电源DC24V,NPN 输入
4	XL_E16V_V2	扩展16点继电器输出,此模块无需供电电源
5	XL_E16V_V3	扩展16点继电器输出,此模块无需供电电源

◆ 机槽号

对应硬件组态中的设备插槽，无论是主机架 CPU 上的模块还是通讯从站后的模块，其槽号都是从 1 开始。其中，在硬件组态界面内的通讯从站本体槽号默认为 0。主机架 CPU 内第一个 1 号机槽对应左扩扩展模块，第二个 1 号机槽对应中扩扩展模块，第三个 1 号机槽对应第一个右扩扩展模块。如下图所示：



- ◆ 设备名  
与软件左侧设备树中所示的设备名称一致。
- ◆ 描述  
设备的基本描述，包括设备基本工作指标及功能。

■ 组态基础操作说明

组态设备基本操作包括设备的刷新、复制、粘贴、删除、撤销、恢复、放大、缩小功能：



◆ 刷新

点击刷新按钮，硬件组态界面或 CPU 机架界面内两个右扩模块之间有空槽的，刷新之后可将空槽右边的模块左移将空槽替换。

新增 XML、EDS、DCF 等描述文件的设备，刷新之后在网络连接设备列表中更新。

◆ 复制

点击新添加的设备后，鼠标点击对应设备图标，设备高亮显示，点击“复制”即可对相应设备进行复制。未点击对应设备图标时，复制按钮不可用。支持快捷键 Ctrl + C 进行复制操作。

◆ 粘贴

对所添加的设备点击复制后可进行粘贴。未复制的情况下，粘贴按钮不可用。支持快捷键 Ctrl + V 进行粘贴操作。

◆ 删除

选中设备后可点击“删除”来删除相应的设备。未选中状态下，删除按钮不可用。支持快捷键 Del 进行删除操作。

◆ 撤销

可撤销针对界面组态上一步的操作步骤，可连续撤销多次。支持快捷键 Ctrl + Z 进行撤销操作。

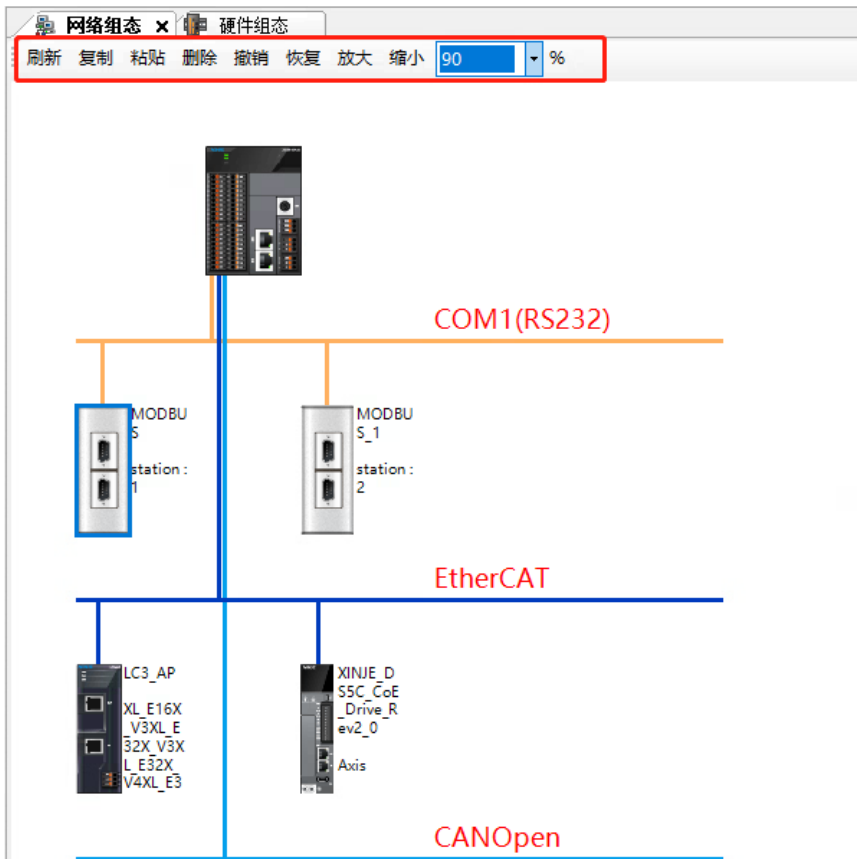
◆ 恢复

针对已撤销的内容，点击“恢复”后可恢复到上一次撤销前的界面。无撤销操作时，恢复按钮不可用。支持快捷键 Ctrl + Y 进行恢复操作。

◆ 放大/缩小

可通过放大/缩小下拉框设置界面缩放比例，也支持快捷键 Ctrl+鼠标放大或缩小当前界面。

如下图所示：



**注:**

- ① 设备复制、粘贴、删除、撤销、恢复操作在硬件组态 EtherCAT 机架与 CPU 机架界面内只针对 IO 模块的操作，CANOpen 机架界面内禁用，而网络组态界面仅针对从站设备；
- ② 如果在网络组态界面中对从站设备进行复制、粘贴、删除，则后面的模块也会被相应操作；
- ③ 导入设备文件：支持通过软件菜单栏的“工具”、“设备存储库”菜单项导入需要的设备文件，可导入 XML、EDS、DCF 等类型的设备描述文件。

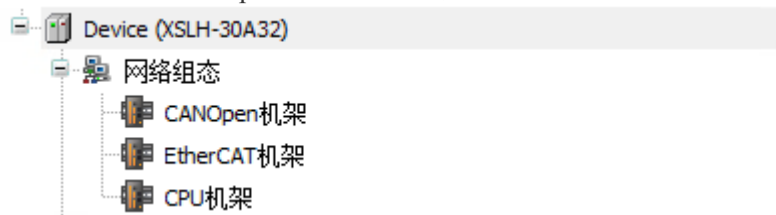
**3-1-2. 硬件组态**

硬件组态引入了实际设备组态中的机架和机槽概念，用来模拟现场设备的模块化配置。硬件组态主要面向 PLC 系列产品的 IO 模块。

从组态流程来讲，如果是添加远程 IO 模块，应该在网络组态中完成通讯模块组态之后，再在硬件组态中进行 IO 模块的配置。

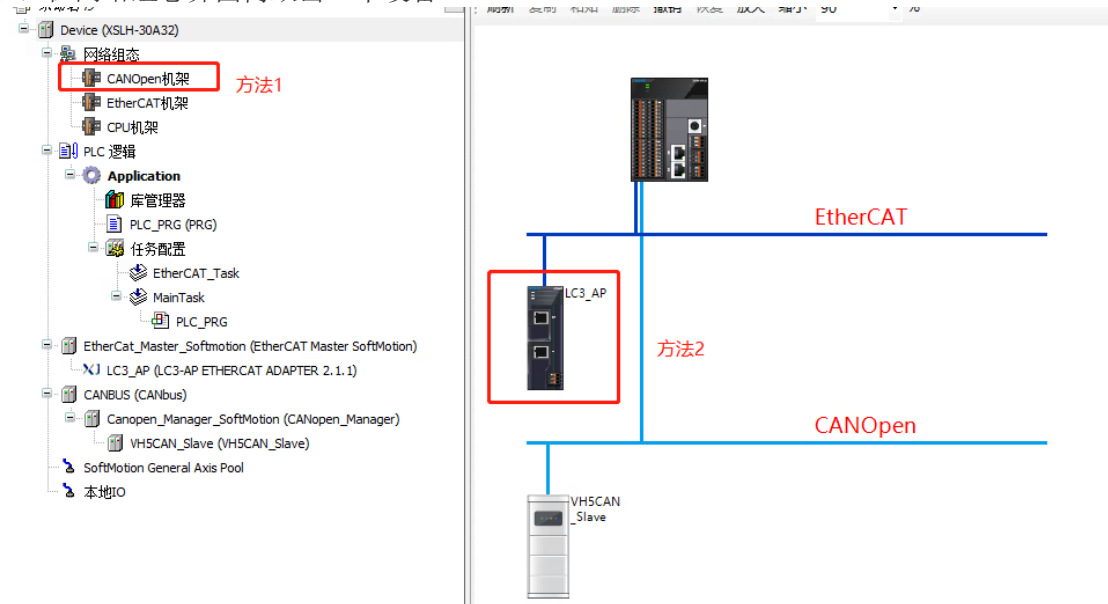
**1、硬件组态界面入口**

目前总线型设备 EtherCAT 与 CANopen 有对应的硬件组态界面。

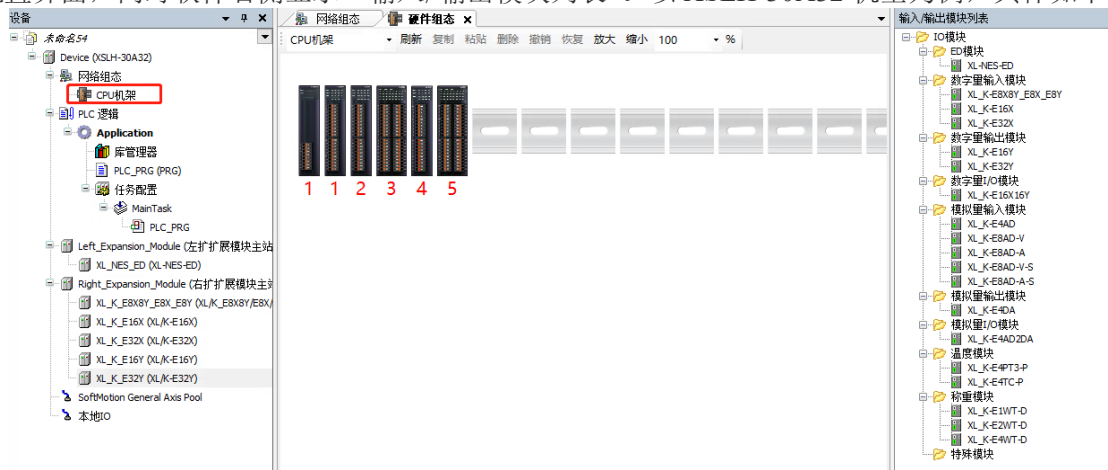


有两种方式可进入硬件组态界面：

- (1) 双击设备树网络组态节点下的某个总线节点；
- (2) 在网络组态界面内双击一个设备。



ARM 系列控制器默认有“CPU 机架”总线配置节点，双击左侧设备树“CPU 机架”即可进入本地模块的配置界面，同时软件右侧显示“输入/输出模块列表”。以 XSLH-30A32 机型为例，具体如下图所示：



注:

- ① XSDH 系列支持添加 1 个 ED 和 1 个 BD, 16 个右扩扩展模块;
- ② XSLH 系列支持添加 1 个 ED 和 16 个右扩扩展模块。

### 2、总线切换

有两种方式可实现硬件组态总线之间的切换。

- (1) 双击软件左侧设备树“网络组态”节点下的某个总线节点, 进入相应的配置界面;
- (2) 在当前硬件组态界面选中下拉框内的其他总线类型进行切换。

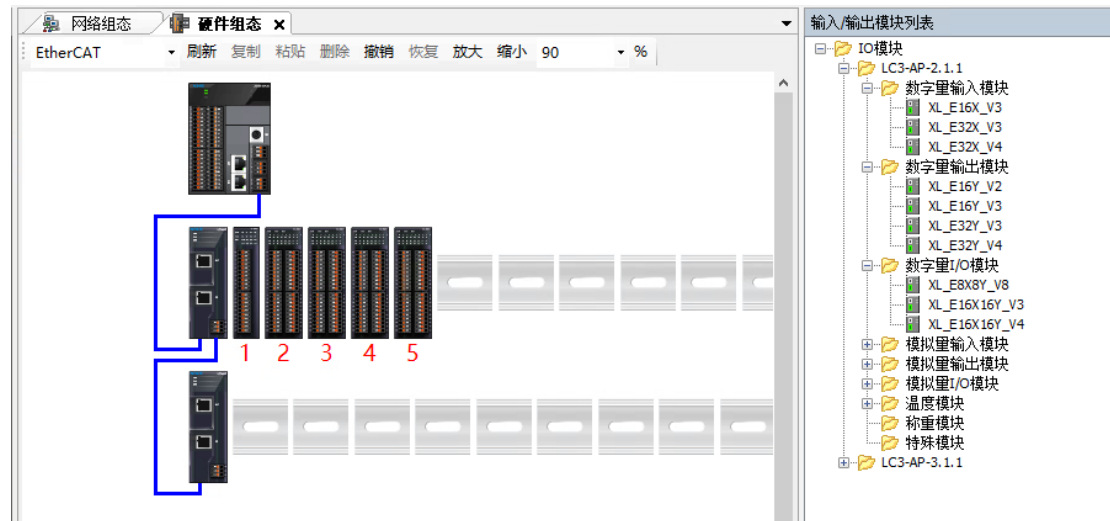


### 3、添加模块

有三种方式来添加 IO 模块。

- (1) 双击打开机架上的一个空机槽, 在弹出的“插入远程 IO 模块”中双击特定模块即可添加;
- (2) 在右侧视图的“输入/输出模块列表”中, 选中一个设备节点, 按住鼠标左键将其拖放到空机槽上;
- (3) 从站后添加 IO 模块则需选中一个设备, 双击视图右侧“输入/输出模块列表”中的某个设备, 即可将设备按顺序自动添加到机架上的空槽中。而如果选中某个空槽, 则会添加到该空槽中。CPU 机架界面内添加 IO 模块则不需要在添加之前先选中本体, 直接双击即可。

如下图所示:



### 4、拖拽模块

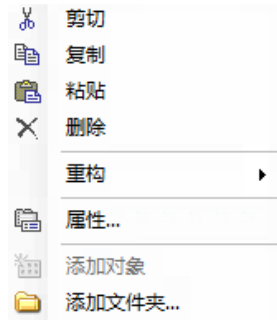
通过选中一个模块按住鼠标左键进行模块的拖拽操作, 拖拽到目标槽位置松开鼠标即可。拖拽操作包括两个模块之间的位置交换, 或将一个模块拖动到空槽中, 但不支持两个扩展机架之间模块的拖拽操作。

## 3-1-3. 设备树操作

### 1、设备树右键菜单

添加总线设备后, 用户可以在设备树选中某一设备, 通过右键菜单或快捷键对设备进行复制、粘贴、删除、剪切和拖动等相关操作。

菜单项如下图所示:

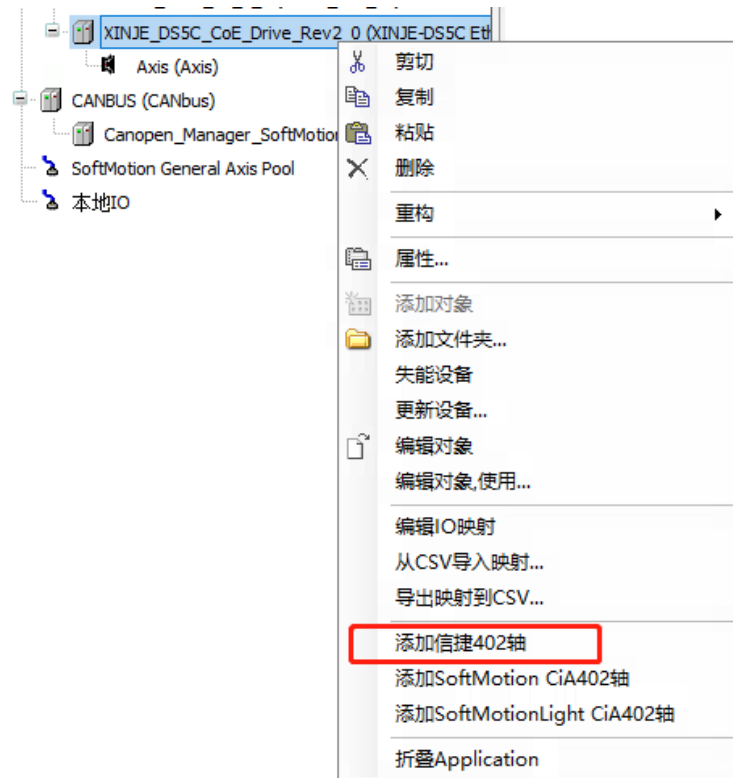


各个功能操作符合基本标准操作。

**注：**复制和剪切功能仅适用于本地主站设备、本地从站设备以及单独的轴设备。

### 2、信捷 402 轴

EtherCAT 伺服支持右键添加“信捷 402 轴”；



信捷 402 轴支持 Home 回原界面配置。

### 3-1-4. 组态编辑错误定位

组态设备制定了一些配置规则及错误检测机制。比如网络组态中两个 MODBUS 设备的站号相同，或是 TCP 设备的 IP 地址相同；硬件组态设备中扩展机架上的从站设备后没有接 IO 模块；EtherCAT 挂载的非失能轴数超出可支持的范围等都会导致组态编译报错。

在编译工程时，如果出现组态错误，XS Studio 消息输出框中会显示，双击其中的错误列表可以自动定位到对应的组态界面。





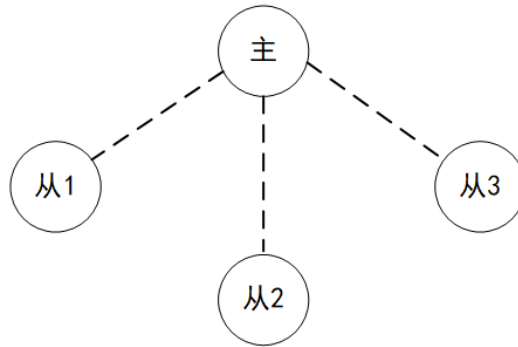
### 3-2. MODBUS 通讯

XS Studio 支持 Modbus 协议通讯主、从机形式。

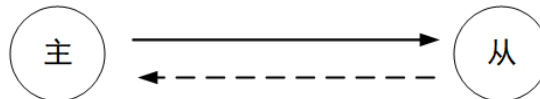
主站形式：可编程控制器作为主站设备时，可与其它使用 Modbus 协议的从机设备通讯；与其他设备进行数据交换。例：信捷 XS 系列 PLC，可以通过通讯来控制变频器。

从站形式：可编程控制器作为从站设备时，只能对其它主站的要求作出响应。

主从的概念：在 RS485 网络中，某一时刻，可以有一主多从（如下图），其中主站可以对其中任意从站进行读写操作，从站之间不可直接进行数据交换，主站需编写通讯程序，对其中的某个从站进行读写，从站无需编写通讯程序，只需对主站的读写进行响应即可。（接线方式：所有的 485+ 连在一起，所有的 485- 连在一起）



在 RS232 网络中（如下图），只能一对一通讯，某一时刻只有一主一从。

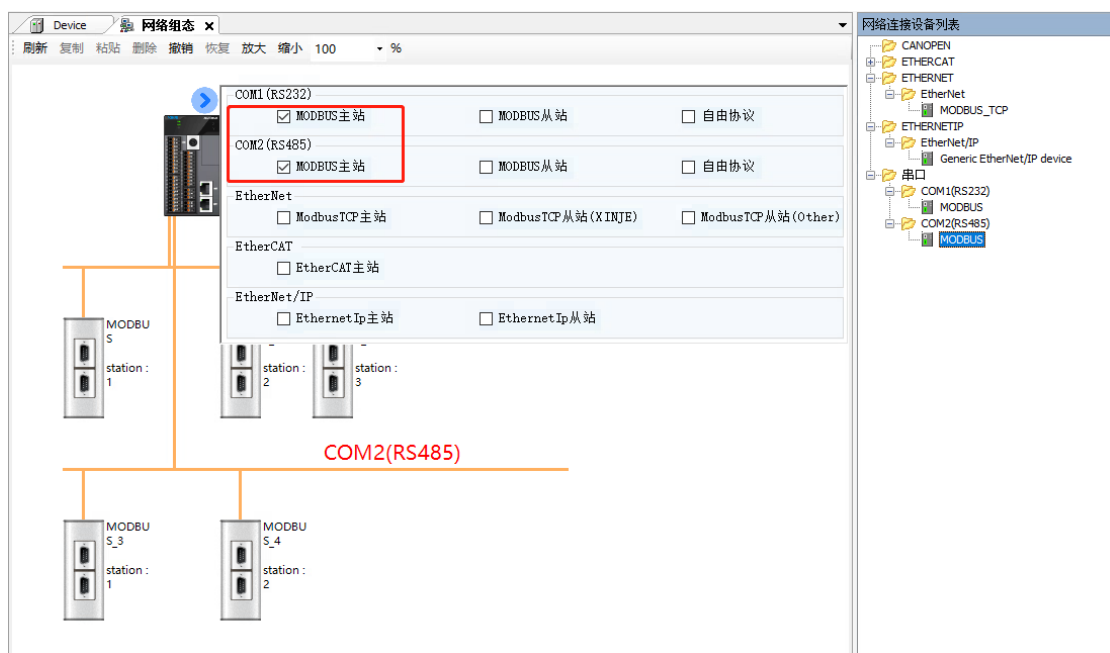


之所以图中有虚线箭头（包括 RS485 网络中），是因为理论上在两个网络中，只要各个 PLC 不发数据，网络中任意 PLC 都可以用来作为主站，其它 PLC 作为从站；但是由于多个 PLC 之间没有一个统一的时钟基准，容易出现在同一时刻有多个 PLC 发送数据，会导致通讯冲突失败，因此不建议这样使用。

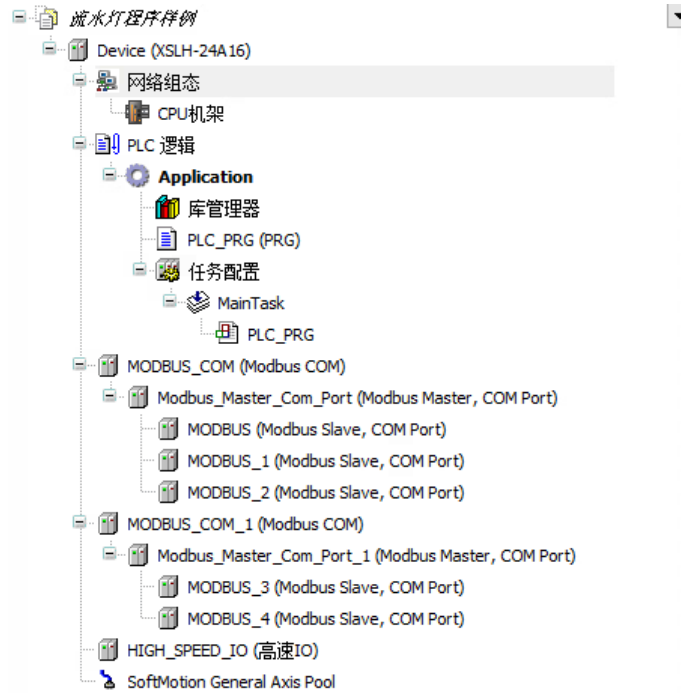
#### 3-2-1. MODBUS 主站配置

##### 1、使能主站、添加主站

单击网络组态中的 PLC 设备，会显示 PLC 内部所支持的主/从站的使能窗口。如下图所示：单击窗口中的复选框按钮来使能 CPU 所支持的主/从站功能，再从视图右侧的“网络连接设备列表”中单击“MODBUS”将从站添加到网络图中。



此时，在界面左侧视图中将出现 Modbus 组态配置对应设备树，如下图所示：



2、主站通信配置

PLC 做 Modbus 主站时，双击设备树中的主站设备，打开 Modbus 主站配置窗口，如下图所示，Modbus 串口通信主站配置界面如图：



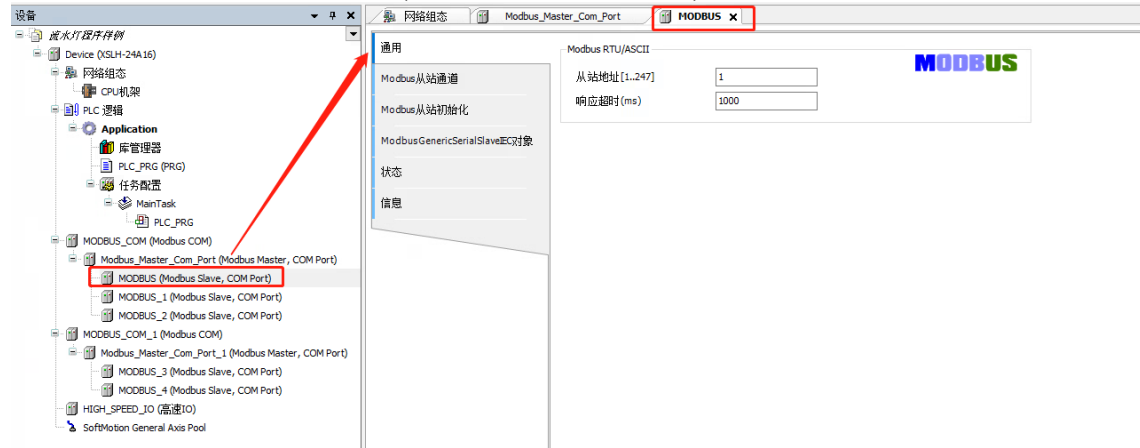
传输模式：选择 RTU 或 ASCII 码。

响应超时 (ms)：指主站等待从站响应的时间间隔。如果在这段时间中从站没有发出响应，主站将会请求下一个从站。此时输入的值会认为是每个从站的缺省值。在从站配置页面，每个从站可单独设置合适的时间间隔。

帧之间的时间 (ms)：指主站接收上一个响应数据帧到下一个请求数据帧之间等待的时间间隔。这个参数可用于调节数据交换率。

至此，主站的配置结束，接下来，需要对主站连接的从站做相应的配置。

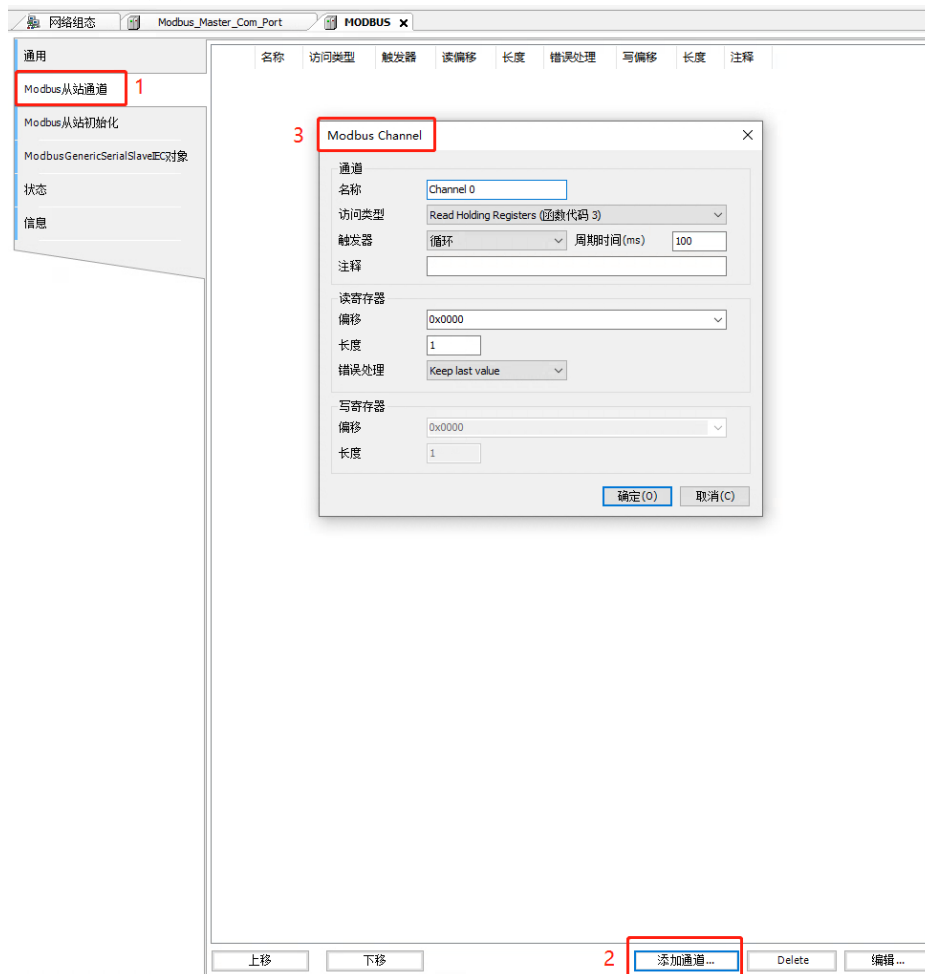
主站配置完成后，双击 MODBUS(Modbus Slave,COM Port)节点打开从站配置界面，如图所示：



从站地址：设置从站的站地址，1~247 有效。

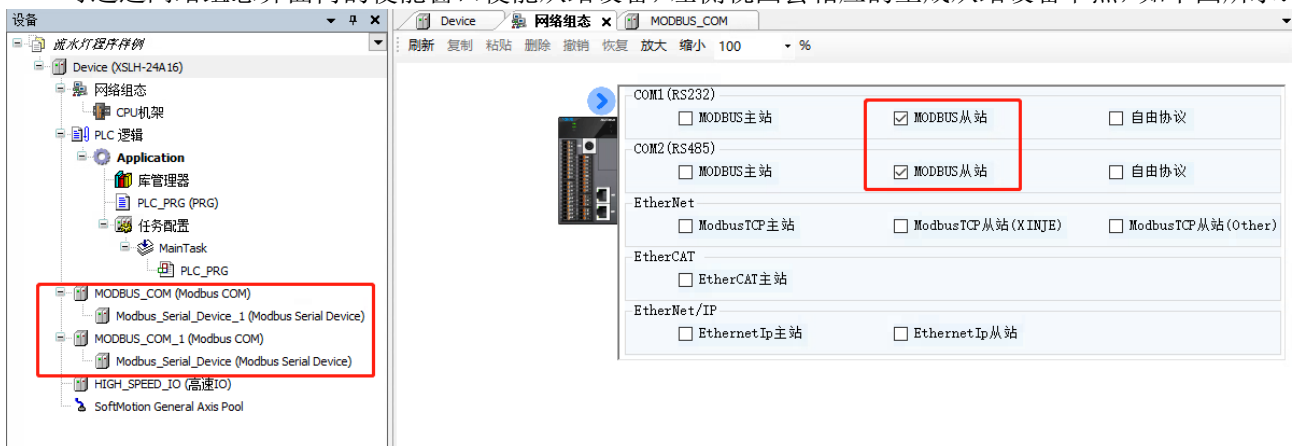
响应超时：设置从站的响应超时时间，如果超过该时间从站还没有响应主站，则主站认为该从站有通讯故障。

设置从站的通讯通道如图所示，在该设置选项中，用户可以自定义从站的 Modbus 通讯通道，但必须与实际的从站硬件相匹配，点击“添加通道”后，系统会自动弹出 Modbus Channel 对话框，用户可以直接选择访问类型、地址偏移、数据长度及通讯周期时间等。

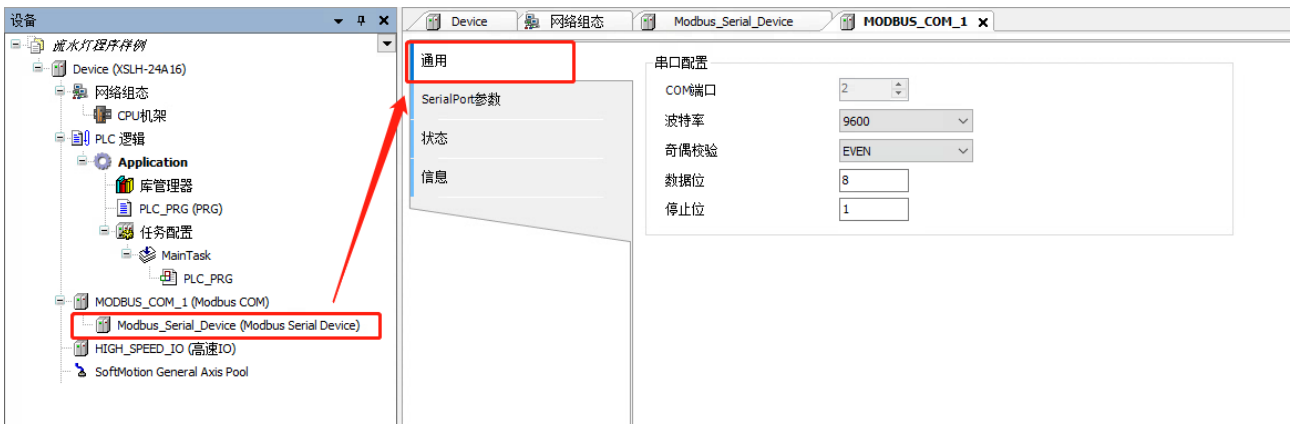


### 3-2-2. MODBUS 从站配置

可通过网络组态界面内的使能窗口使能从站设备，左侧视图会相应的生成从站设备节点，如下图所示：

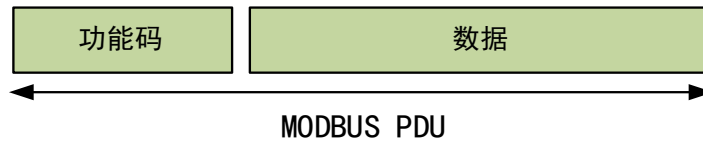


添加从站设备后，双击 MODBUS(Modbus Slave,COM Port)节点打开配置界面，可切换至 Modbus 从站配置界面。具体如下图：



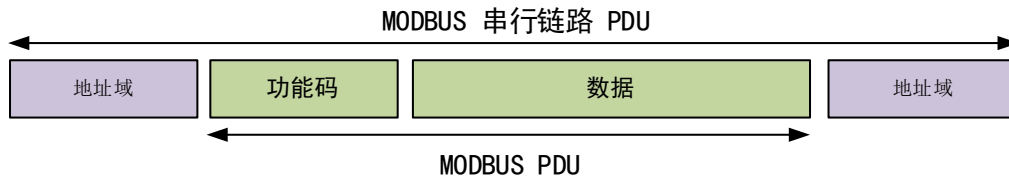
### 3-2-3. MODBUS 通讯帧格式说明

《Modbus 应用协议》定义了简单的协议数据单元 (PDU - Protocol Data Unit) :



Modbus 协议数据单元

发起 Modbus 事务处理的客户端构造 Modbus PDU, 然后添加附加的域以构造通信 PDU。



串行链路上的Modbus数据帧

1、在 Modbus 串行链路, 地址域只含有子节点地址。

如前文所述, 合法的子节点地址为十进制 0~247, 每个子设备被赋予 1~247 范围中的地址, 主节点通过将子节点的地址放到报文的地址域对子节点寻址。当子节点返回应答时, 它将自己的地址放到应答报文的地址域以让主节点知道哪个子节点在回答。

2、功能码指明服务器要执行的动作。功能码后面可跟有表示含有请求和响应参数的数据域。

3、错误检验域是对报文内容执行 "冗余校验" 的计算结果。

根据不同的传输模式 (RTU or ASCII) 使用两种不同的计算方法。

有两种串行传输模式被定义: RTU 模式和 ASCII 模式。所有设备必须实现 RTU 模式, ASCII 传输模式是选项。Modbus RTU 一般采用串口 RS232C 或 RS485/422, 而 Modbus TCP 一般采用以太网口。

#### ■ RTU 传输模式

当设备使用 RTU (Remote Terminal Unit) 模式在 Modbus 串行链路通信, 报文中每个 8 位字节含有两个 4 位十六进制字符。这种模式的主要优点是较高的数据密度, 在相同的波特率下比 ASCII 模式有更高的吞吐率。每个报文必须以连续的字符流传送。

**RTU 模式每个字节 (11 位) 的格式为:**

编码系统: 8 位二进制, 报文中每个 8 位字节含有两个 4 位十六进制字符 (0~9, A~F)

**每字节的 bit 流:**

- ◆ 1 起始位
- ◆ 8 数据位, 首先发送最低有效位
- ◆ 1 位作为奇偶校验
- ◆ 1 停止位

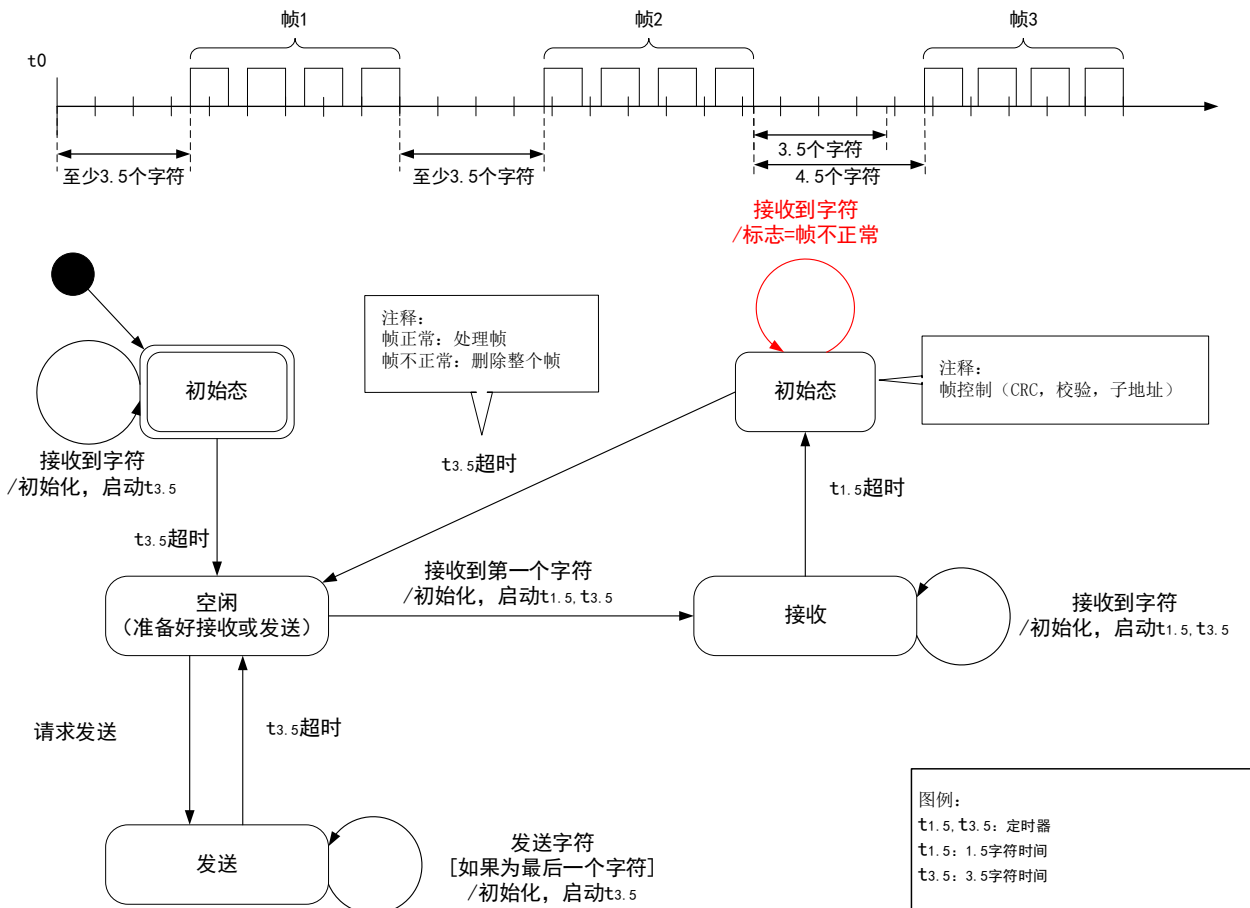
偶校验是要求的, 其它模式 (奇校验, 无校验) 也可以使用。为了保证与其它产品的最大兼容性, 同时

支持无校验模式是建议的。默认校验模式模式必须为偶校验。

注：使用无校验要求 2 个停止位。

地址码	功能码	数据	校验码
1 字节	1 字节	n 节	2 字节 (CRC)

在 RTU 模式，报文帧由时长至少为 3.5 个字符时间的空闲间隔区分。在后续的部分，这个时间区间被称作 t3.5。



- ◆ 从“初始”态到“空闲”态转换需要 t3.5 定时超时：这保证帧间延迟；
- ◆ “空闲”态是没有发送和接收报文要处理的正常状态；
- ◆ 在 RTU 模式，当没有活动的传输的时间间隔达 3.5 个字符长时，通信链路被认为在“空闲”态。
- ◆ 当链路空闲时，在链路上检测到的任何传输的字符被识别为帧起始。链路变为“活动”状态。然后，当链路上没有字符传输的时间间隔达到 t3.5 后，被识别为帧结束
- ◆ 检测到帧结束后，完成 CRC 计算和检验。然后，分析地址域以确定帧是否发往此设备，如果不是，则丢弃此帧。为了减少接收处理时间，地址域可以在一接到就分析，而不需要等到整个帧结束。这样，CRC 计算只需要在帧寻址到该节点(包括广播帧)时进行。

■ ASCII 传输模式

当 Modbus 串行链路的设备被配置为使用 ASCII 模式通信时，报文中的每个 8 位字节以两个 ASCII 字符发送。当通信链路或者设备无法符合 RTU 模式的定时管理时使用该模式。

注：由于一个字节需要两个字符，此模式比 RTU 效率低。

ASCII 模式每个字节（11 位）的格式为：

编码系统：十六进制，ASCII 字符 0~9, A~F。报文中每个 ASCII 字符含有 1 个十六进制字符

每字节的 bit 流：

- ◆ 1 起始位
- ◆ 8 数据位，首先发送最低有效位
- ◆ 1 位作为奇偶校验

◆ 1 停止位

偶校验是要求的，其它模式（奇校验、无校验）也可以使用。为了保证与其它产品的最大兼容性，同时支持无校验模式是建议的。默认校验模式必须为偶校验。

注：使用无校验要求 2 个停止位。

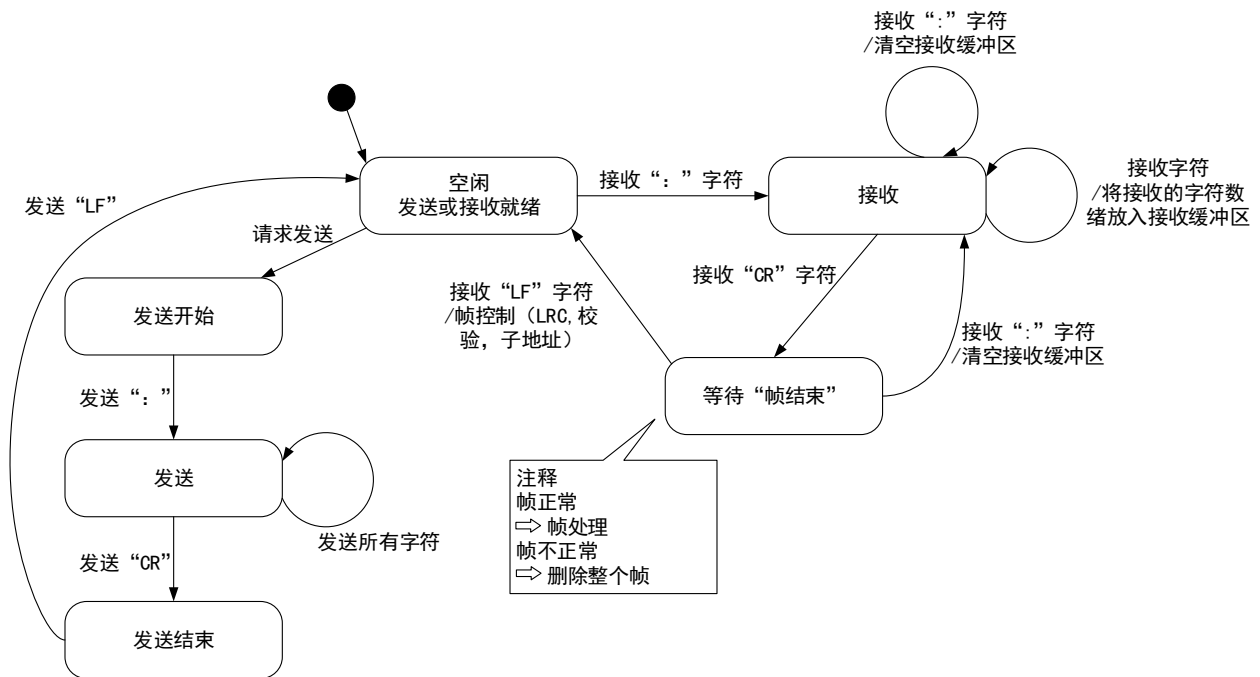
起始	地址码	功能码	数据	校验码	回车换行
字符“:”(冒号)	2 字节	2 字节	0 到 2*252 字节	2 字节 (LRC 校验)	2 字节 (CR, LF)

报文帧的地址域含有两个字符。

在 ASCII 模式，报文用特殊的字符区分帧起始和帧结束。一个报文必须以一个“冒号”(ASCII 十六进制 3A)起始，以“回车-换行”对应(ASCII 十六进制 0D 和 0A)结束。

对于所有的域，允许传送的字符为十六进制 0~9, A~F(ASCII 编码)。设备连续的监视总线上的“冒号”字符。当收到这个字符后，每个设备解码后续的字符一直到帧结束。

报文中字符间的时间间隔可以达一秒。如果有更大的间隔，则接受设备认为发生了错误。



ASCII 传输模式状态图

◆ “空闲”态是没有发送和接收报文要处理的正常状态。

◆ 每次接收到“:”字符表示新的报文的开始。如果在一个报文的接收过程中收到该字符，则当前地报文被认为不完整并被丢弃。而一个新的接收缓冲区被重新分配。

◆ 检测到帧结束后，完成 LRC 计算和检验。然后，分析地址域以确定帧是否发往此设备，如果不是，则丢弃此帧。为了减少接收处理时间，地址域可以在一接到就分析，而不需要等到整个帧结束。

### 3-3. 串口自由协议通讯

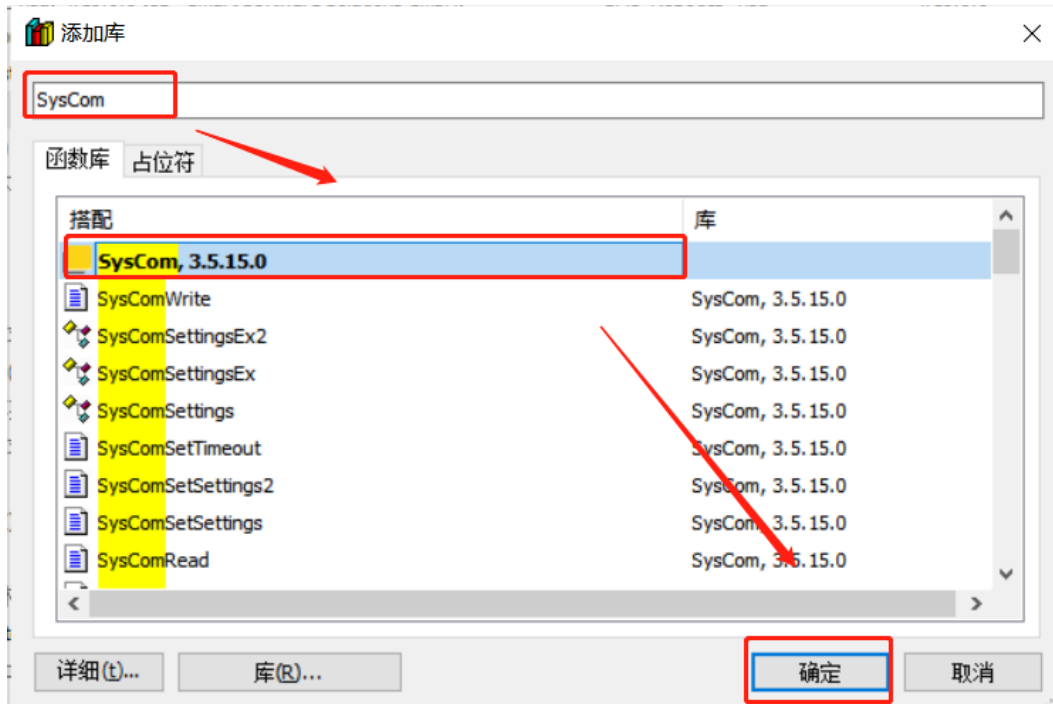
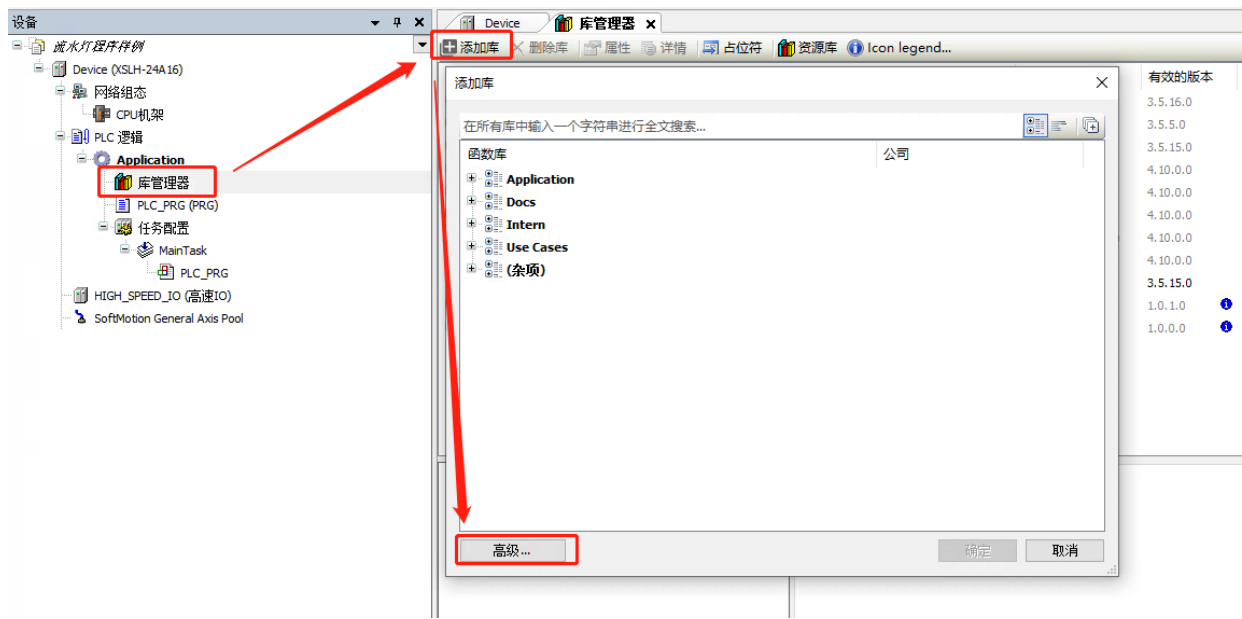
信捷 PLC 在与其它设备通讯的时候, 如果作为下位机, 则上位机必须按照 MODBUS RTU 的数据格式与其进行数据交换; 如果信捷 PLC 作为上位机, 当下位机也支持 MODBUS RTU 协议的时候, 可直接使用相关通讯指令进行通讯, 使程序编写更简单, 效率更高, 如果下位机不直接支持 MODBUS RTU 协议, 则可使用自由格式通讯。

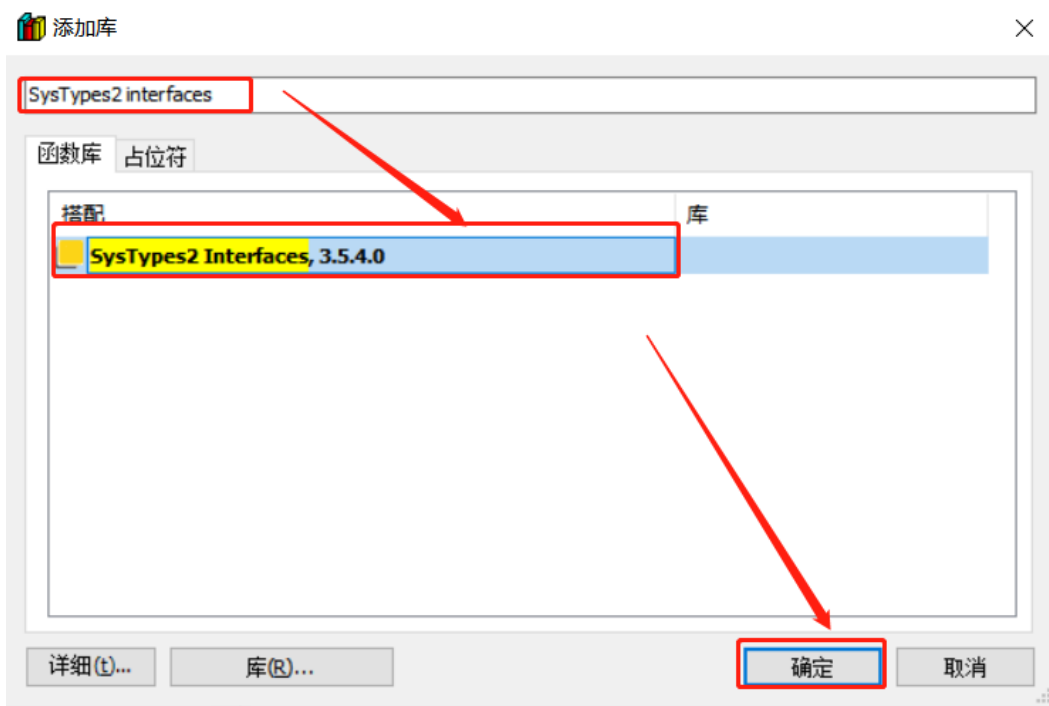
所谓自由格式, 就是当下位机通讯协议与 PLC 协议不匹配时候, PLC 内部自定义数据格式来发送数据, 这样可以和很多下位机进行通讯。

自由格式通讯是以数据块的形式进行数据传送, 每块最大可传送 256 字节, 同时每块可设置一个起始符和终止符, 也可以不设。

#### 3-3-1. 参数配置

在“库管理器”中添加两个库--SysCom 和 SysTypes2 interfaces, 添加与上位机版本对应的库版本。





### 3-3-2. 应用举例

**例一：**通过下面程序，实现两台 PLC 之间进行自由格式通讯，进行数据发送/接收。

**注意：**自由格式通讯进行数据接收时，要么常 ON 导通时需将对应的 TASK 的周期拉大，要么接收使用上升沿接收。

程序操作：

- (1) 按照 3-2-1 节中的步骤安装需要使用的库；
- (2) 编写自由格式程序。

```

1  PROGRAM POU
2  VAR
3      SysCom2Settings:SysCom.SysComSettings;
4      SysCom2SettingsEx:SysCom.SysComSettingsEx;
5      StartSetting:BOOL:=1;
6      SendData:ARRAY[0..19] OF BYTE;
7      result:(*POINTER TO *)SysTypes.RTS_IEC_RESULT;
8      Send_start:BOOL;
9      Ton0:Standard.TON;
10     i:INT;
11     RCVData:ARRAY[0..19] OF BYTE;
12     RCV_start:BOOL;
13     hCom:SysTypes.RTS_IEC_HANDLE:=SysTypes.RTS_INVALID_HANDLE;
14     size : UDINT;
15     // close_en:BOOL;
16 END_VAR

17 IF StartSetting THEN
18     hCom:=SysCom.SysComOpen(sPort:=SysCom.SYS_COMPORT2,pResult:=ADR(result));
19     IF hCom<>RTS_INVALID_HANDLE THEN
20         result := SysComGetSettings(hCom:= hCom, pSettings:= ADR(SysCom2Settings), pSettingsEx:= ADR(SysCom2SettingsEx)); //调用接口
21         SysCom2Settings.sPort:=SysCom.SYS_COMPORT2; //端口
22         SysCom2Settings.ulBaudrate:=SysCom.SYS_BR_19200; //波特率
23         SysCom2Settings.byStopBits:=SysCom.SYS_ONESTOPBIT; //停止位
24         SysCom2Settings.byParity:=SysCom.SYS_EVENPARITY; //校验
25         SysCom2Settings.ulTimeout:=SYS_NOWAIT;
26         //SysCom2Settings.ulBufferSize:=8;
27         SysCom2SettingsEx.byByteSize:=8; //数据位
28         SysCom2SettingsEx.bOutX := FALSE; //流控制
29         SysComSetSettings(hCom:= hCom, pSettings:= ADR(SysCom2Settings), pSettingsEx:= ADR(SysCom2SettingsEx)); //将设置的机构体里的参数通过接口设置
30         SysComPurge(hCom:= hCom);
31     END IF
32     StartSetting:=0; //端口设置不能为常ON
33 END IF
34 Ton0(IN:=NOT Ton0.Q ,PT:=T#1S,Q=>,EI=>);
35 FOR i:=0 TO 19 BY 1 DO
36     IF Ton0.Q THEN
37         SendData[i]:=SendData[i]+1;
38     END_IF
39 END_FOR
40 //开始发送数据
41 IF Send_start AND Ton0.Q THEN
42     SysCom.SysComWrite(hCom:=hCom,pbyBuffer:=ADR(SendData),ulSize:=SIZEOF(SendData),ulTimeout:=SysCom.SYS_NOWAIT,pResult:=ADR(result));
43 END_IF
44 //进行接收数据 (注意: 要么TASK的周期拉大, 要么接收使用上升沿)。
45 IF RCV_start THEN
46     size := SysCom.SysComRead(hCom:=hCom,pbyBuffer:=ADR(RCVData),ulSize:=SIZEOF(RCVData),ulTimeout:=SysCom.SYS_NOWAIT,pResult:=ADR(result));
47 END_IF

```



### 3-4. ModbusTCP 通讯

Modbus TCP 使用 TCP/IP 在站点间传送 Modbus 报文, Modbus TCP 结合了 TCP/IP 协议以及以 Modbus 协议作为应用协议标准的数据表示方法。Modbus TCP 通信报文被封装于以太网 TCP/IP 数据包中。与传统的串口方式, Modbus TCP 插入一个标准的 MODBUS 报文到 TCP 报文中, 不再带有数据校验和地址。

XS 系列可编程控制器本体支持 Modbus TCP 协议通讯主、从机形式。

主站形式: 可编程控制器作为主站设备时, 可与其它使用 Modbus TCP 协议的从机设备通讯。一个主站最多可以连接 64 个从站。

从站形式: 可编程控制器作为从站设备时, 只能对其它主站的要求作出响应。

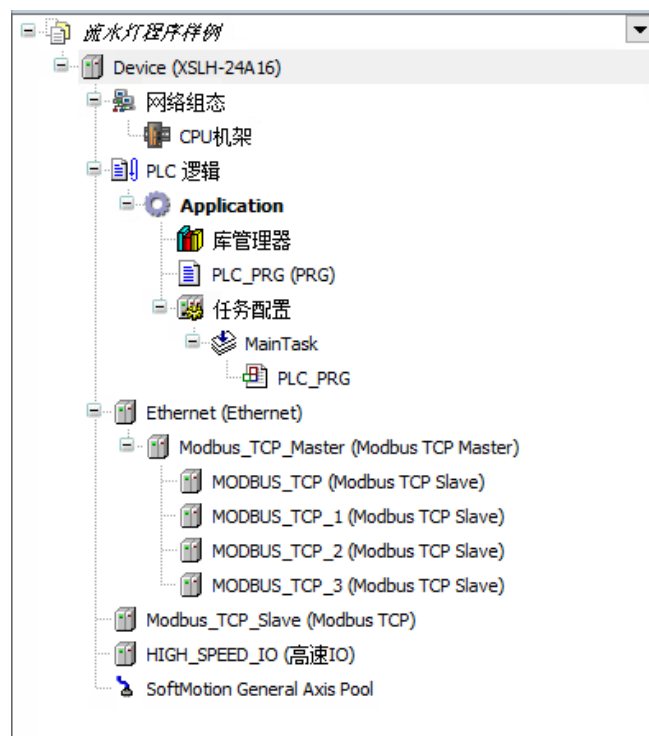
#### 3-4-1. MODBUS TCP 主站配置

##### 1、使能主站、添加从站

单击网络组态中的 PLC 设备, 会显示 PLC 内部所支持的主/从站的使能窗口。如下图所示: 单击窗口中的复选框按钮来使能 CPU 所支持的主/从站功能, 再从视图右侧的“网络连接设备列表”中单击“MODBUS\_TCP”将从站添加到网络中。

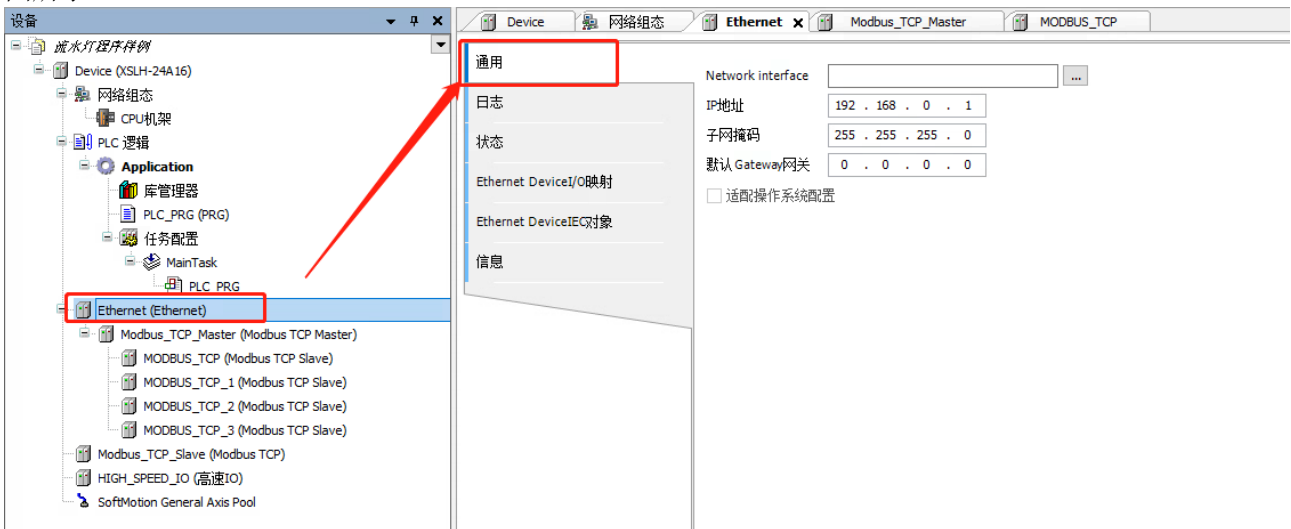


此时, 在界面左侧视图中将出现 ModbusTCP 组态配置对应设备树, 如下图所示:

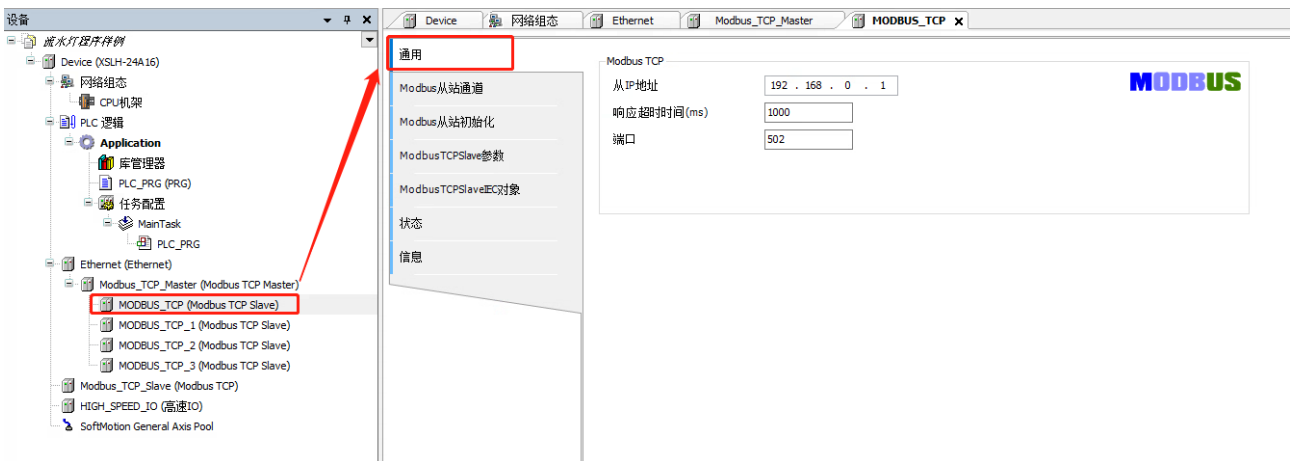


### 2、主站通讯配置

PLC 做 Modbus TCP 主站时，双击设备树中的 Ethernet，打开 Modbus 主站配置窗口并进行配置，如下图所示：

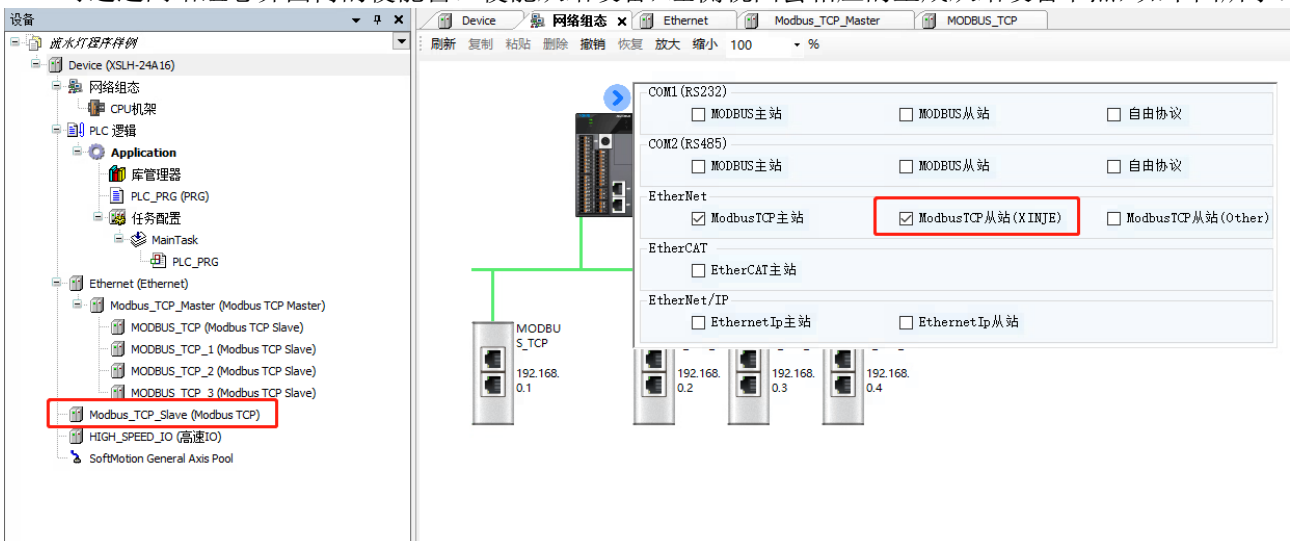


双击 MODBUS\_TCP(MODBUS\_TCP\_Slave)节点打开配置界面配置从站信息，如图所示：

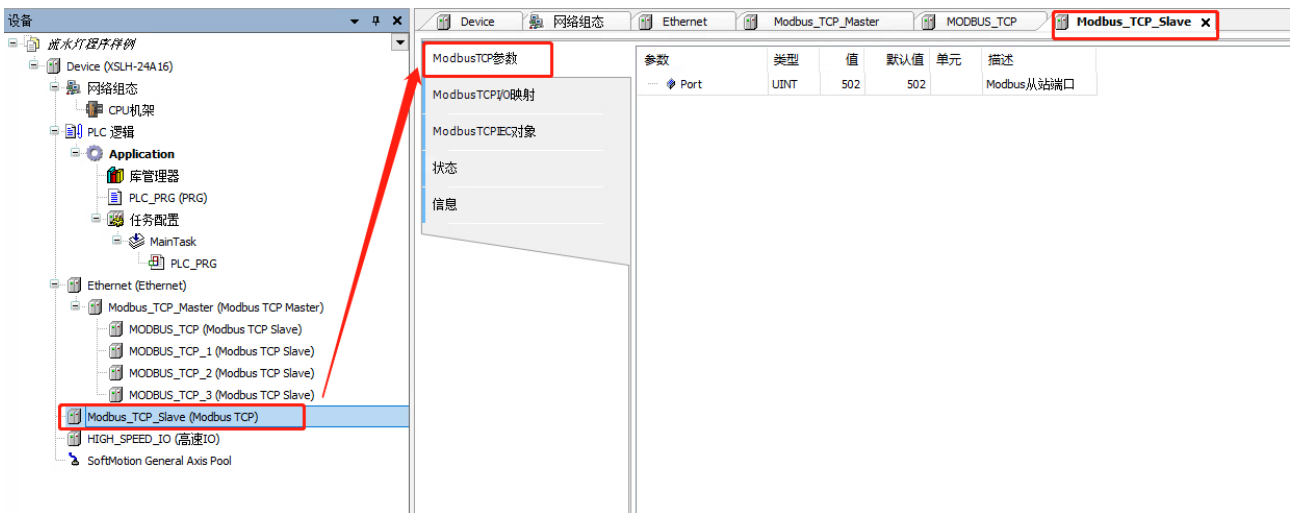


### 3-4-2. MODBUS TCP 从站配置

可通过网络组态界面内的的使能窗口使能从站设备，左侧视图会相应的生成从站设备节点，如下图所示：



添加从站设备后，在设备树节点下双击 Modbus\_TCP\_Slave 节点打开配置界面，可切换至 Modbus TCP 从站配置界面。具体如下图：



### 3-4-3. MODBUS TCP 常见故障

1、主站无法对作为从站的信捷 XS 控制器进行读写处理：主站方配置从站参数时，站号要填写 0。

2、信捷 XS 控制器作为主站，与从站通讯不上处理：

(1) 为客户端去访问这个地址，首先得保证服务器里有这个地址，不然客户端无法访问不存在的地址，会报错；

(2) 查看从站是否有这些访问的数据类型、首地址及通讯个数；

(3) 需注意一下功能码，功能码对不上，也不能通讯上。

### 3-4-4. MODBUS TCP 通讯帧格式说明

Modbus 设备可分为主站 (poll) 和从站 (slave)。主站只有一个，从站有多个，主站向各从站发送请求帧，从站给予响应。在使用 TCP 通信时，主站为 client 端，主动建立连接；从站为 server 端，等待连接。

- ◆ 主站请求：功能码+数据；

- ◆ 从站正常响应：请求功能码+响应数据；

- ◆ 从站异常响应：异常功能码+异常码，其中异常功能码即将请求功能码的最高有效位置 1，异常码指示差错类型；

- ◆ 注意：需要超时管理机制，避免无期限的等待可能不出现的应答。

IANA (Internet Assigned Numbers Authority, 互联网编号分配管理机构) 给 Modbus 协议赋予 TCP 端口号为 502，这是目前在仪表与自动化行业中唯一分配到的端口号。

ModbusTCP 的数据帧可分为两部分：MBAP+PDU。

#### ■ 报文头 MBAP

MBAP 为报文头，长度为 7 字节，组成如下：

事务处理标识	协议标识	长度	单元标识符
2 字节	2 字节	2 字节	1 字节

**事务处理标识：**可以理解为报文的序列号，一般每次通信之后就要加 1 以区别不同的通信数据报文。

**协议标识符：**00 00 表示 ModbusTCP 协议。

**长度：**表示接下来的数据长度，单位为字节。

**单元标识符：**可以理解为设备地址。

#### ■ 帧结构 PDU

PDU 由功能码+数据组成。功能码为 1 字节，数据长度不定，由具体功能决定。

**通信过程：**

- 1、connect 建立 TCP 连接；
- 2、准备 Modbus 报文；
- 3、使用 send 命令发送报文；

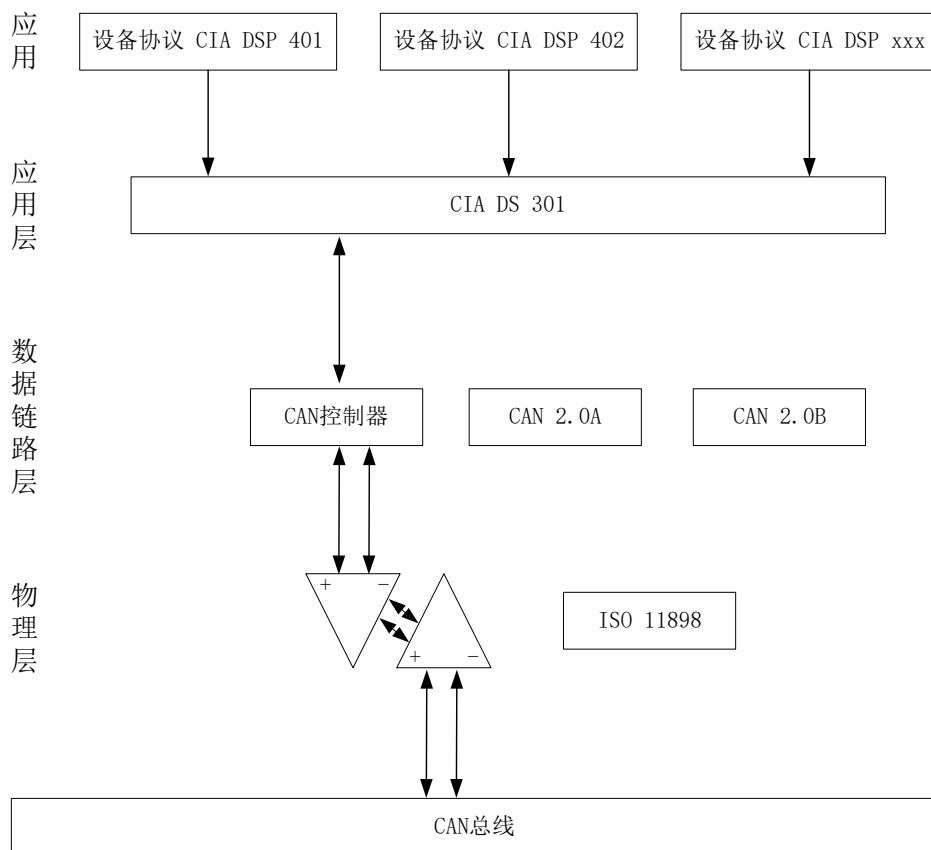
- 4、在同一连接下等待应答；
- 5、使用 `recv` 命令读取报文，完成一次数据交换；
- 6、通信任务结束时，关闭 TCP 连接。

### 3-5. CANopen 网络

CANopen 协议是在 20 世纪 90 年代末，由总部位于德国纽伦堡的 CiA 组织（CAN-in-Automation）在 CAL（CAN Application Layer）的基础上发展而来。

CANopen 是一个基于 CAN 串行总线的网络传输系统的应用层协议，遵循 ISO/OSI 标准模型。其中基础协议是 CANopen 应用层协议（CANopen Application Layer and Communication Profile）(DS 301)，规定了 CANopen 协议层及通信结构描述。在基础协议之上，各个行业具备设备子协议。所谓子协议，就是针对不同行业的应用对象，对 CANopen 内部的数据含义进行重新定义或添加新的控制逻辑。

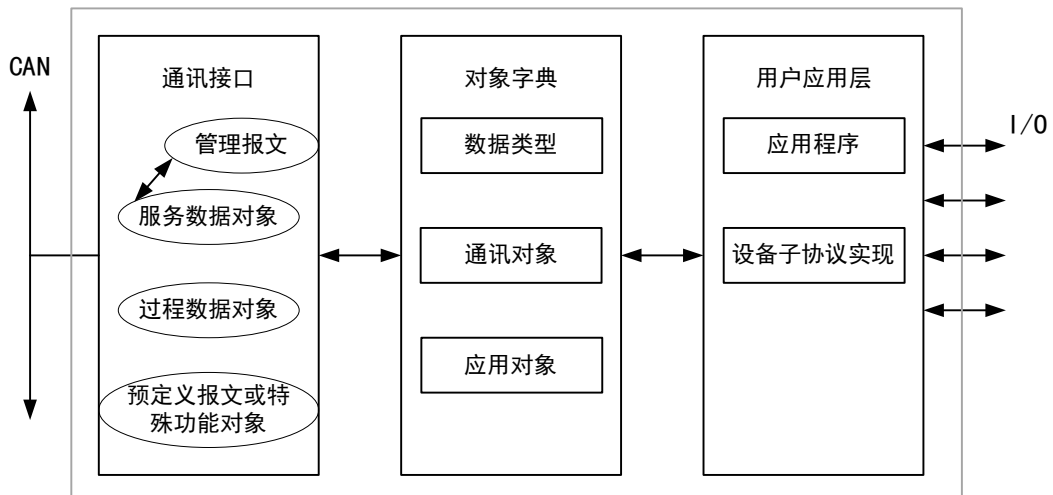
在 OSI 模型中，CAN 标准、CANopen 协议之间的关系如下图所示：



OSI 模型是一个概念模型，用于标准化各种通信技术之间的通信功能。较低的层描述基本的通信（例如原始比特流），而较高的层描述诸如长消息的分段之类的东西以及诸如消息的发起、指示、响应和确认之类的服务。

CANopen 协议通常分为用户应用层、对象字典以及通信三个部分。其中最核心的是对象字典，CANopen 通讯通过对象字典（OD）能够访问驱动器的所有参数。

CANopen 设备结构如下图所示：



## ■ 通信对象

CANopen 协议中常用的通信对象包含如下几点：

### 1、网络管理对象(NMT)

网络管理对象包括 Boot-up 消息, Heartbeat 协议及 NMT 消息, 基于主从通信模式, NMT 用于管理和监控网络中的各个节点, 主要实现三种功能: 节点状态控制、错误控制和节点启动。

### 2、服务数据对象(SDO)

服务数据对象主要用于主节点对从节点的参数配置。服务确认是 SDO 最大的特点, 为每个消息都生成一个应答, 以确保数据传输的准确性。在一个 CANopen 系统中, 通常 CANopen 从节点作为 SDO 服务器, CANopen 主节点作为客户端。客户端通过索引和子索引能够访问数据服务器上的对象字典, 所以 CANopen 主节点可以访问从节点的任意对象字典项的参数, 并且 SDO 可以传输任何长度的数据(当数据长度超过 4 字节时, 拆分成多个报文来传输)。

### 3、过程数据对象(PDO)

用来传输实时数据, 数据从一个创建者传到一个或多个接收者。数据传送限制在 1~8 个字节。每个 CANopen 设备包含 8 个缺省的 PDO 通道, 4 个发送 PDO 通道和 4 个接收 PDO 通道。PDO 包含同步和异步两种传输方式, 由该 PDO 对应的通信参数决定。

### 4、同步对象(SYNC)

同步对象是由 CANopen 主站周期性地广播到 CAN 总线的报文, 用来实现基本的网络时钟信号, 每个设备可以根据自己的配置, 决定是否使用该事件来跟其它网络设备进行同步通信。

### 5、紧急报文(EMCY)

设备内部通信故障或者应用故障错误时发送的报文。

## ■ 对象字典

CANopen 对象字典(Object Dictionary, OD)是 CANopen 协议最为核心的概念。所谓的“对象字典”, 就是一个有序的对象组; 每个对象采用一个 16 位的索引值来寻址。为了访问数据结构中的元素, 同时定义了一个 8 位的子索引。

CANopen 网络中每个节点都有一个对象字典。对象字典包含了描述这个设备和它的网络行为的所有参数。

CANopen 对象字典中的项由一系列子协议来描述。子协议描述对象字典中每个对象的功能、名字、索引、子索引、数据类型、读/写属性, 以及这个对象是否必需, 从而保证不同厂商的同类型设备兼容。

CANopen 协议的核心描述子协议是 DS301, 包括 CANopen 协议应用层及通信结构描述, 其他子协议都是对 DS301 协议描述文本的补充与扩展。

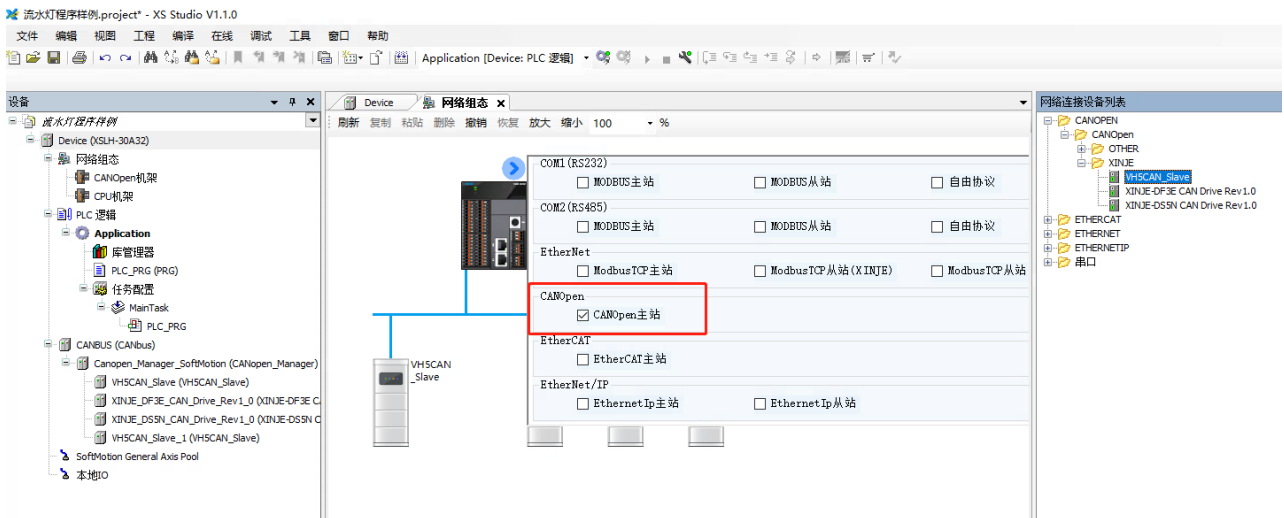
## 3-5-1. 参数配置

### 1、硬件端口

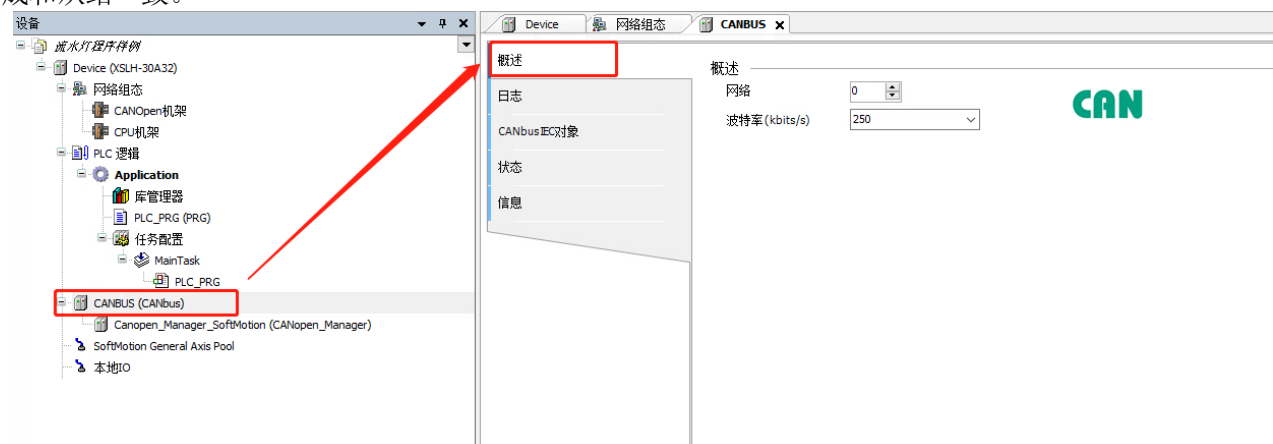
设备接入 CAN 总线时, 需要将 CAN+连接 CAN+, CAN-连接 CAN-, 若从站为伺服, 需要将网线一端的第一根(TX+)和第二根(TX-)线分别接到 CAN+和 CAN-上, 另外一端插入到伺服的网口。同时, PLC 上的拨码 3 和 4 为自带的终端电阻, 需要将其置 ON, 为了增强 CAN 通讯的可靠性, 消除 CAN 总线终端信号反射干扰, CAN 总线网络最远的两个端点通常要加入终端电阻, 若其它 CANopen 设备未自带终端电阻, 需用户自行安装。

### 2、软件配置

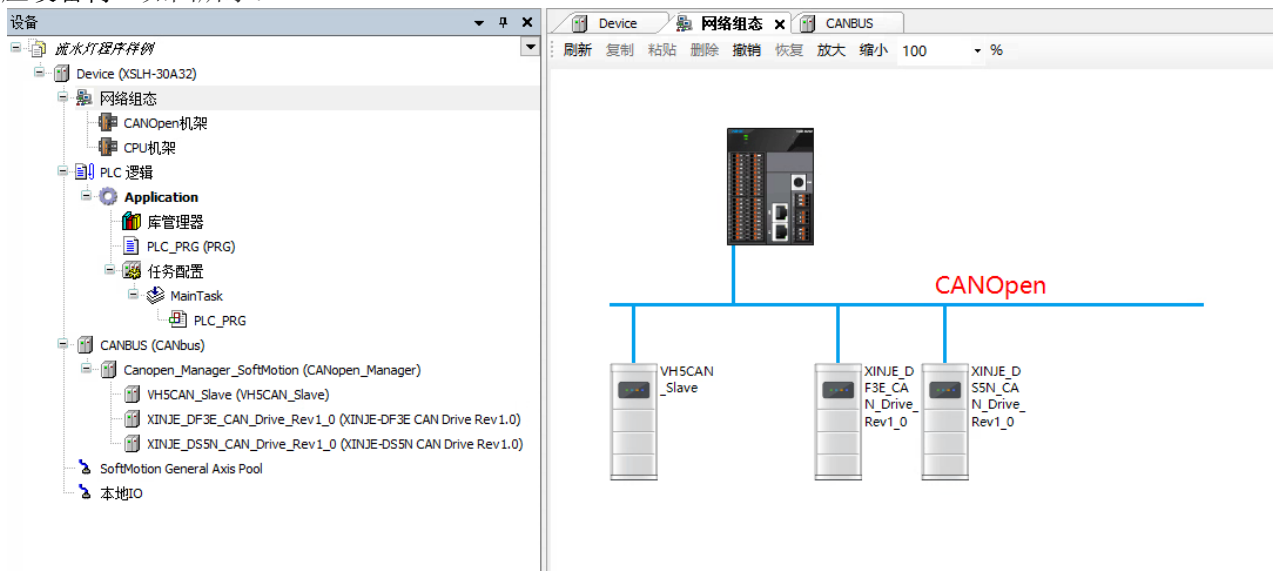
(1) 在网络组态中激活 CANopen 总线。激活 CANopen 总线后, 会自动添加 CANopen 主站。



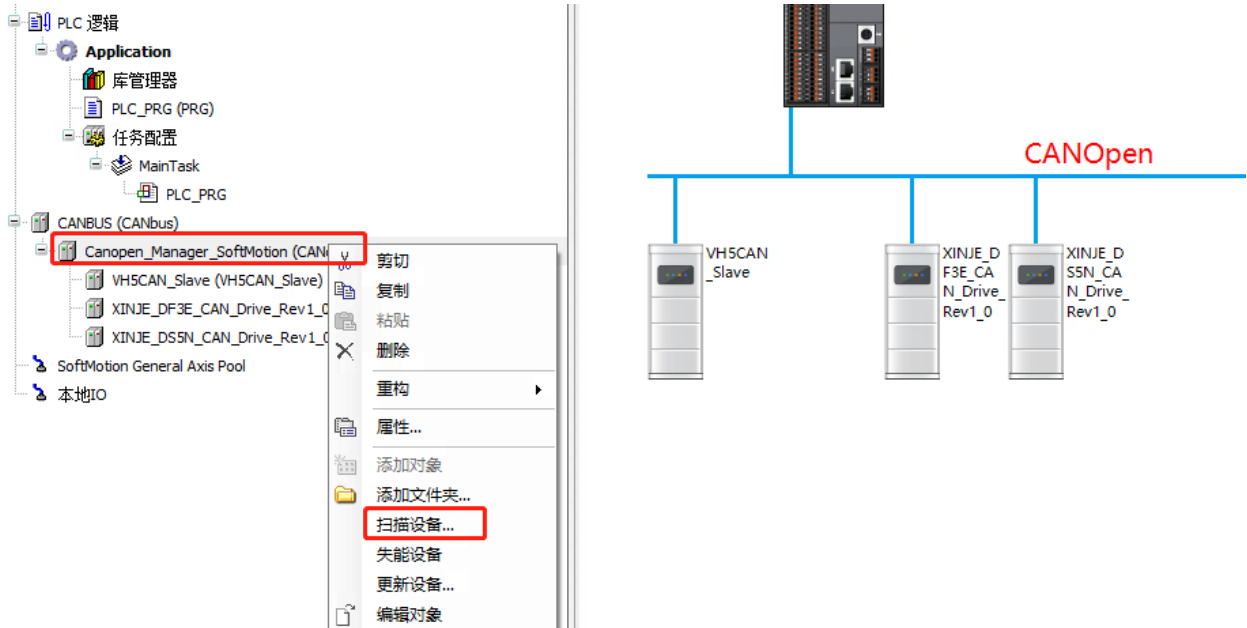
(2) 添加成功后可看到设备栏下的“CANBus”，双击“CANBus”，在界面“概述”的波特率设置成和从站一致。



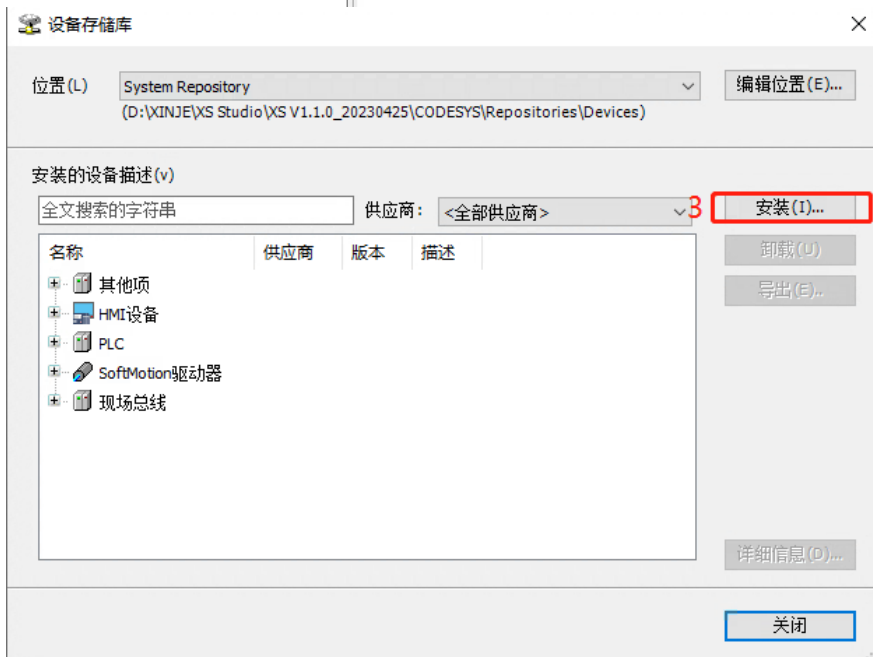
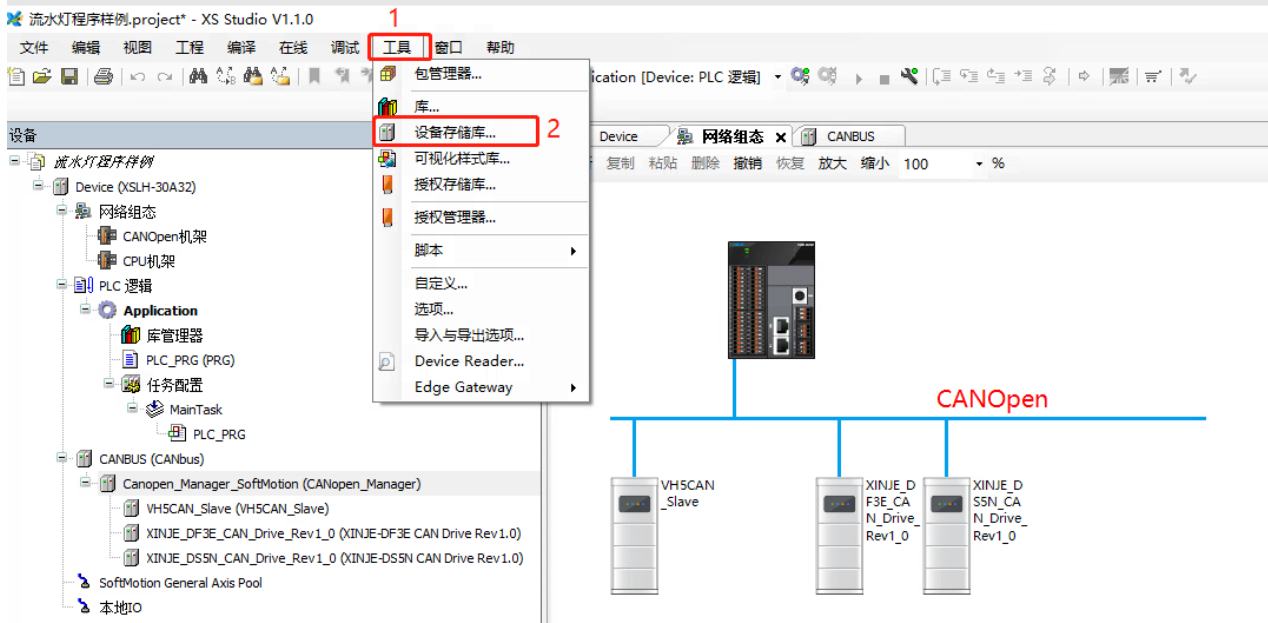
通过右侧“网络设备连接列表”可添加 CANopen 从站模块，并在界面左侧视图中将出现组态配置对应设备树。如图所示：



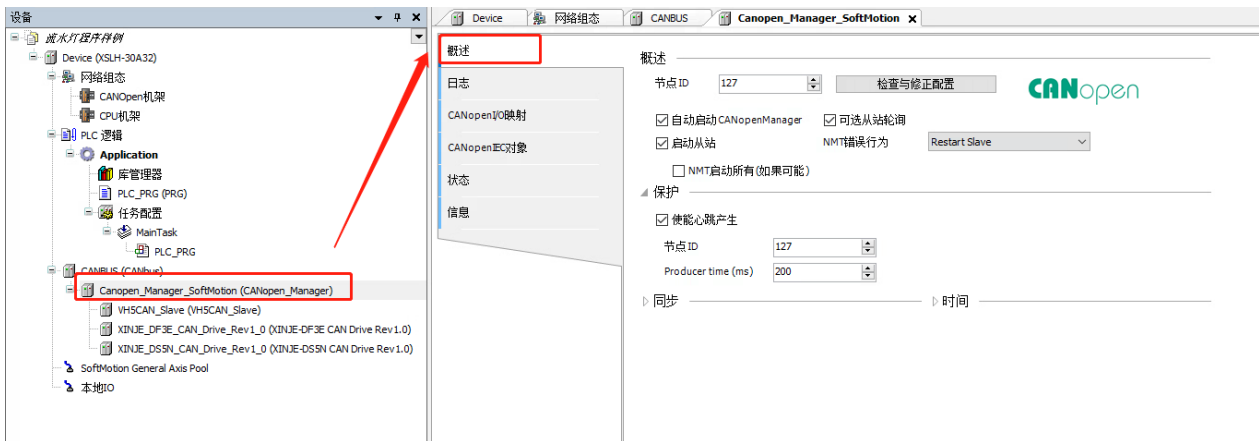
(3) 添加“CANopen\_Manager\_SoftMotion”后，需要先下载程序，下载后右击“CANopen\_Manager\_SoftMotion”扫描设备，扫描成功后复制到工程中。



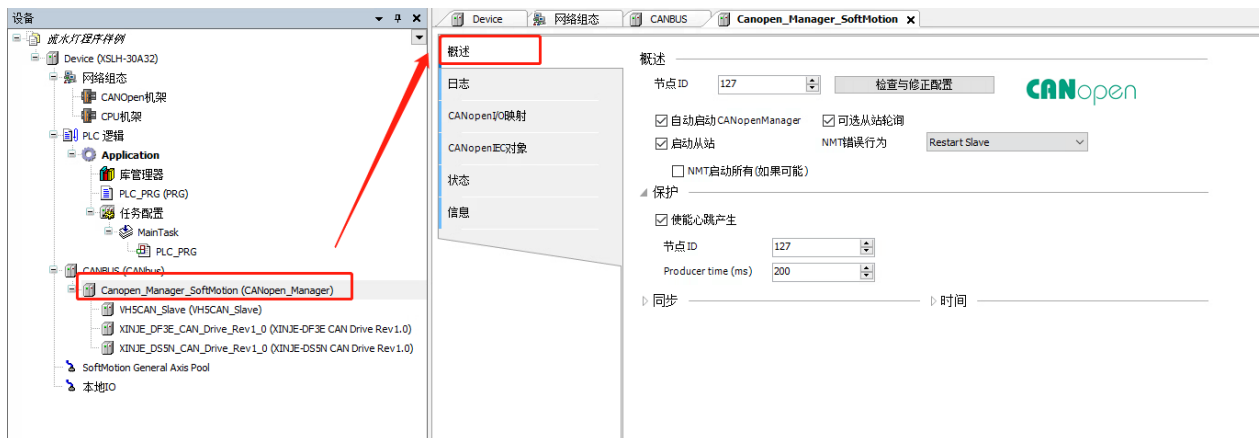
若没有扫描成功，可检查是否导入 EDS 文件，在工具-设备存储器-安装，将从站的 EDS 文件导入进来。扫描进来后，从站的节点 ID 会自动识别。若是手动添加从站设备，需要手动修改从站 ID。







(4) 在“CANopen\_Manager\_SoftMotion”中，需要对 CANopen 主站进行设置。

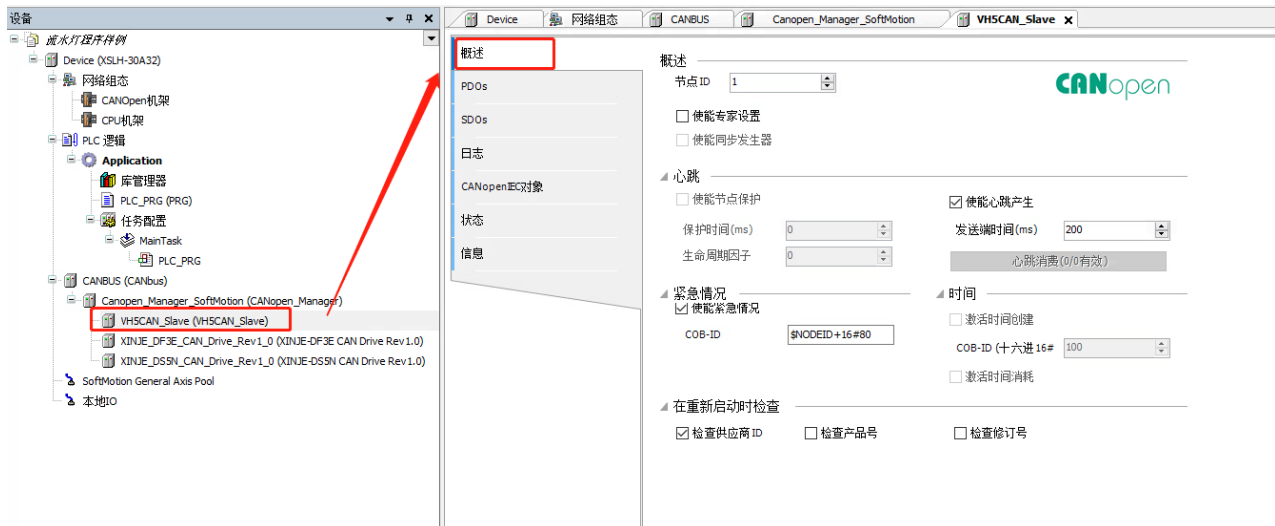


- ◆ 节点 ID: 主站在 CANopen 网络唯一标示号，默认 127，范围 1~127。
- ◆ 检查与修正配置: 点击进入后，若有错误可点击“自动修复”。
- ◆ 自动启动 CANopenManager (默认: 开启): 勾选后在所有从站准备就绪后 CANopen 管理器自动重启。
  - ◆ 可选从站轮廓 (默认: 开启): 当从站在引导序列中没有响应时，CANopen 管理器每秒询问一次它，直到它做出响应。不断轮询从站会增加总线周期时间，这会干扰应用 (尤其是运动应用)。您可以停用轮询以避免此行为。如果轮询被停用，则在发送启动消息时会再次检测到从属服务器。
    - ◆ 启动从站 (默认: 开启): CANopen 管理器负责启动从站。
    - ◆ NMT 启动所有 (如果可能): 如果激活了启动从站选项 (默认: 禁用)，则 CANopen 管理器使用“NMT 全部启动”命令启动所有从站。只要可选从站尚未准备好启动，就不会执行“NMT 全部启动”命令。在这种情况下，CANopen 管理器单独启动每个从站。“NMT 全部启动”命令只能在没有可选从属设备的项目中得到保证。
      - ◆ NMT 错误行为: Restart Slave-如果在从机监控期间发生错误 (NMT 错误事件)，则堆栈会自动重新启动从站 (NMT 复位+ SDO 配置+ NMT 启动); Stop Slave-如果在从站监控期间发生错误 (NMT 错误事件)，则从站将停止。然后，您必须使用 CiA405 NMT 功能块从应用程序重置从机。
- 保护
  - ◆ 使能心跳产生: 如果启用这个选项 (默认: 禁用)，主站将发送心跳信息。
  - ◆ 节点 ID: 发送心跳信息的唯一标识符，默认为主站节点 ID，范围 1~127 (十进制)。
  - ◆ 生产时间(ms): 心跳信息发送的时间间隔，单位为毫秒，范围为 1ms~65535ms，并且是总线任务时间的整数倍。
- 同步:
  - ◆ 启动同步生成: 如果启用这个选项 (默认: 开启)，主站将发送同步信息。一个 CANopen 总线系统只能有一个站启用同步生产。同步类型 PDO 在同步信息发送后根据设置类型发送信息。
  - ◆ COB-ID: 通信对象标识，此设置用于标识同步消息 ID。值不能修改，为 16#80。
  - ◆ 循环周期 (us): 同步信息以同步周期定义的时间间隔发送，同步周期的单位为微秒，范围为 100-4294967295us，并且是总线任务时间的整数倍。
  - ◆ 窗口长度 (us): 用于同步 PDO，以微秒为单位的时间窗长度。
  - ◆ 启用同步消费: 如果启用这个选项 (默认: 禁用)，另一个设备必须生成由 CANopen 管理器接收

的 SYNC 报文。

- 时间

- ◆ 启动时间生成：如果启用这个选项（默认：禁用），CANopen 管理器发送 TIME 消息。
  - ◆ COB-ID:（通信对象标识符）：标识消息的时间戳。默认值：[0...2047]，预设 16#100。
  - ◆ Producer time (ms)：发送时间戳时的时间间隔，此值必须是任务周期时间的倍数，范围[0..65535]。
- (5) 双击从站设备，在 CANopen 从站里配置从站基本参数、PDO 配置、SDO 配置。



- 基本参数：

- ◆ 节点 ID：从站在 CANopen 网络唯一标示号范围 1~127（十进制），需要和从站本身一致。
- ◆ 使能专家设置：如果启用这个选项（默认：禁用），显示由设备的设备描述（EDS 文件）预定义的所有设置。
  - ◆ 使能同步发生器：如果启用这个选项（默认：禁用），此从站将发送同步信息。一个 CANopen 总线系统只能有一个启用同步生产。同步发送参数使用主站的同步配置参数。
  - ◆ 使能同步发生器：仅在“CANopen\_Manager\_SoftMotion”中选择“启用同步生成”选项时可用。如果启用这个选项（默认：禁用），I/O 传输在总线上同步。从站充当同步生产者。同步间隔的参数在“CANopen\_Manager\_SoftMotion”的设置中定义。

- 心跳

- ◆ 使能节点保护：如果启用这个选项（默认：禁用），CANopen 管理器在保护时间（ms）间隔内向从站发送消息。如果从站没有使用给定的防护 COB-ID（通信对象标识符）进行响应，则 CANopen 管理器会按照生存期因子中定义的次数重新发送此消息，或者直到从站响应为止。如果从站没有响应，则将其标记为“不可用”。
  - ◆ 保护时间（ms）：发送消息的间隔（默认：200 毫秒）。
  - ◆ 生命周期因子：当从站没有响应时，将根据生命周期因子时间乘以保护时间建立节点保护错误。
  - ◆ 使能心跳产生：模块按照生产者时间（ms）中给出的时间间隔发送检测信号。
  - ◆ 发送端时间（ms）：看设备描述文件里设置的时间。

- 紧急情况

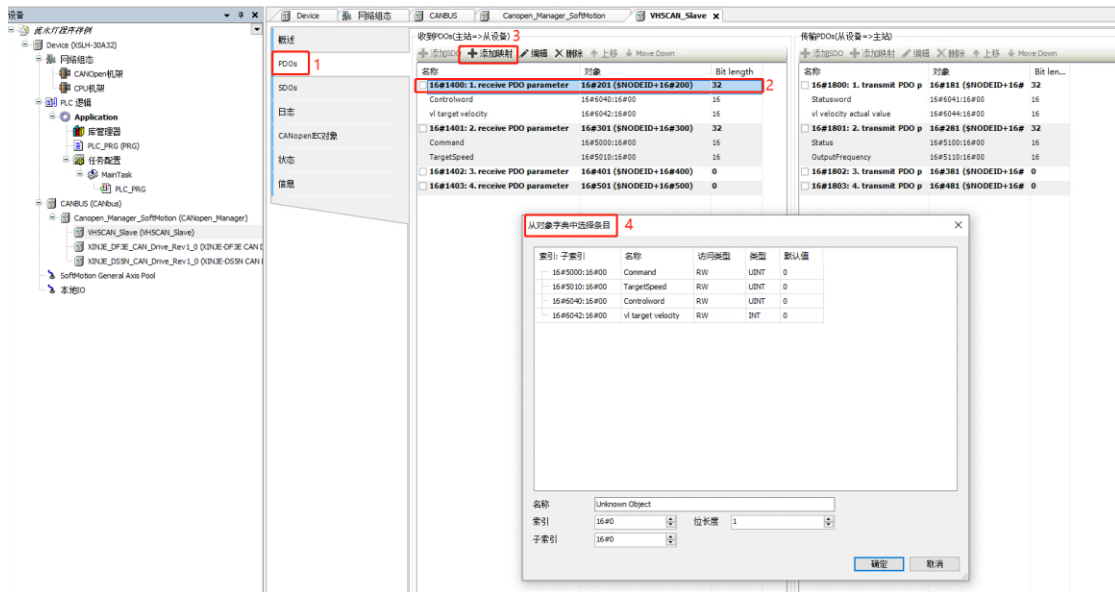
- ◆ 使能紧急情况：当发生内部错误时，从站发送具有唯一 COB-ID 的紧急消息。
- ◆ COB-ID：从站发送紧急报文的 COB-ID，默认为 \$NODEID+16#80
- ◆ 时间--此功能的可用性取决于设备描述
- ◆ 激活时间创建：设备发送时间消息。
- ◆ COB-ID（十六进制）：（通信对象标识符）：标识消息的时间戳。
- ◆ 激活时间消耗：设备处理时间消息。

- 在重新启动时检查

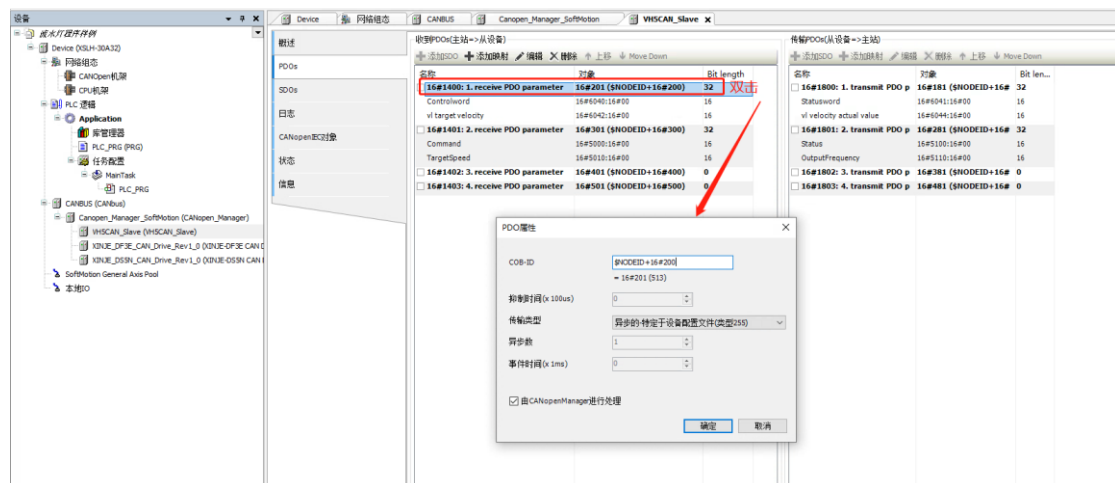
从 CANopen 从站（0x1018 身份对象）的固件中读取相应的信息，并与 EDS 文件中的信息进行比较。如果出现差异，则停止配置，不启动从站。

- ◆ 检查供应商 ID：在启动时检查供应商 ID。
- ◆ 检查产品号：在启动时检查产品编号。
- ◆ 检查修订号：启动时检查修订版本号。

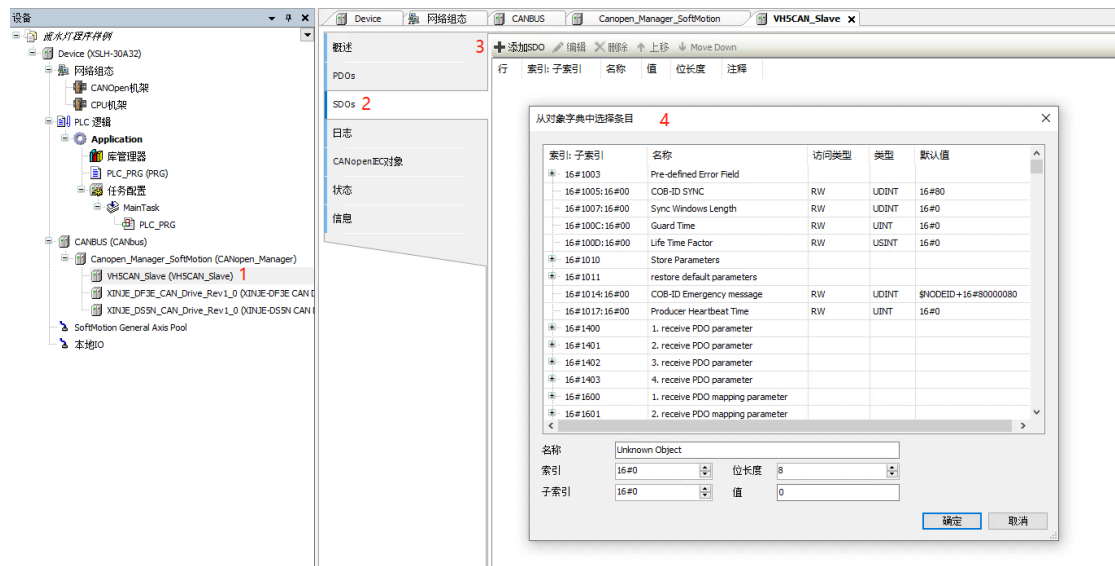
- ◆ PDOs:
- ◆ PDO（过程数据对象）对于主站和从站之间的实时数据传输，接收 PDO 为主站向从站发送的实时数据。
- ◆ 通过在 PDOs 界面，接收 PDO 来自于对象字典中从索引 1400~1403，发送 PDO 来自于对象字典中从索引 1800~1803，点击索引添加需要的通信参数，选择索引和子索引，点击“确定”。如用户需要添加/删除/修改映射地址，则需要在“接收 PDO”和“发送 PDO 映射”中设置。



双击加粗字体-索引，可以对具体 PDO 进行设置，可以设置其 COB-ID，传输类型的方式。

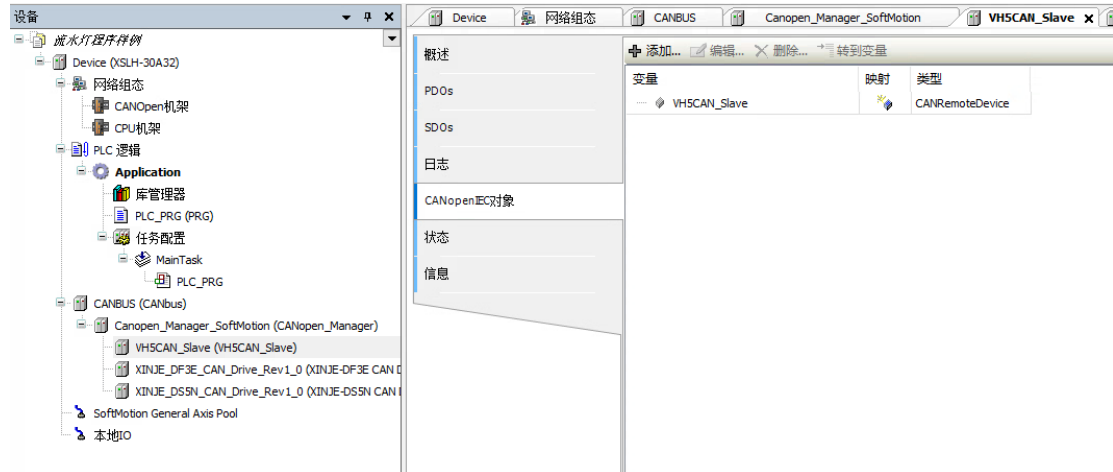


SDOs:



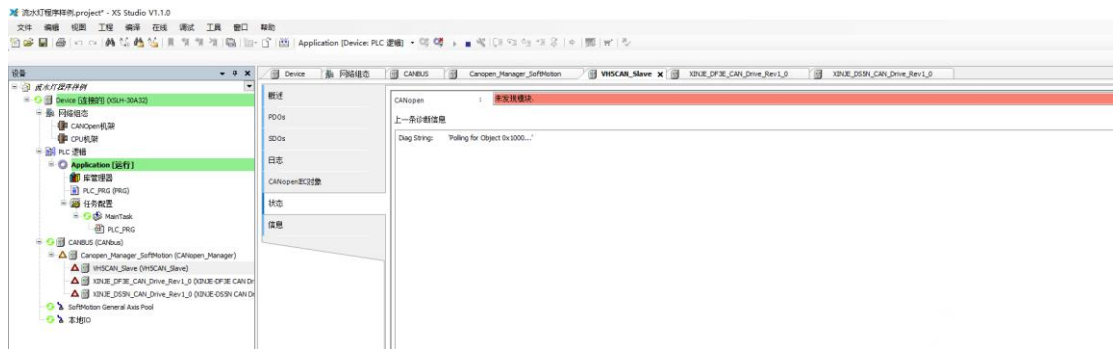
CANopen/IO 映射:

可以查看 CANopen/IO 映射关系、功能描述、实际地址以及映射变量的类型。



状态:

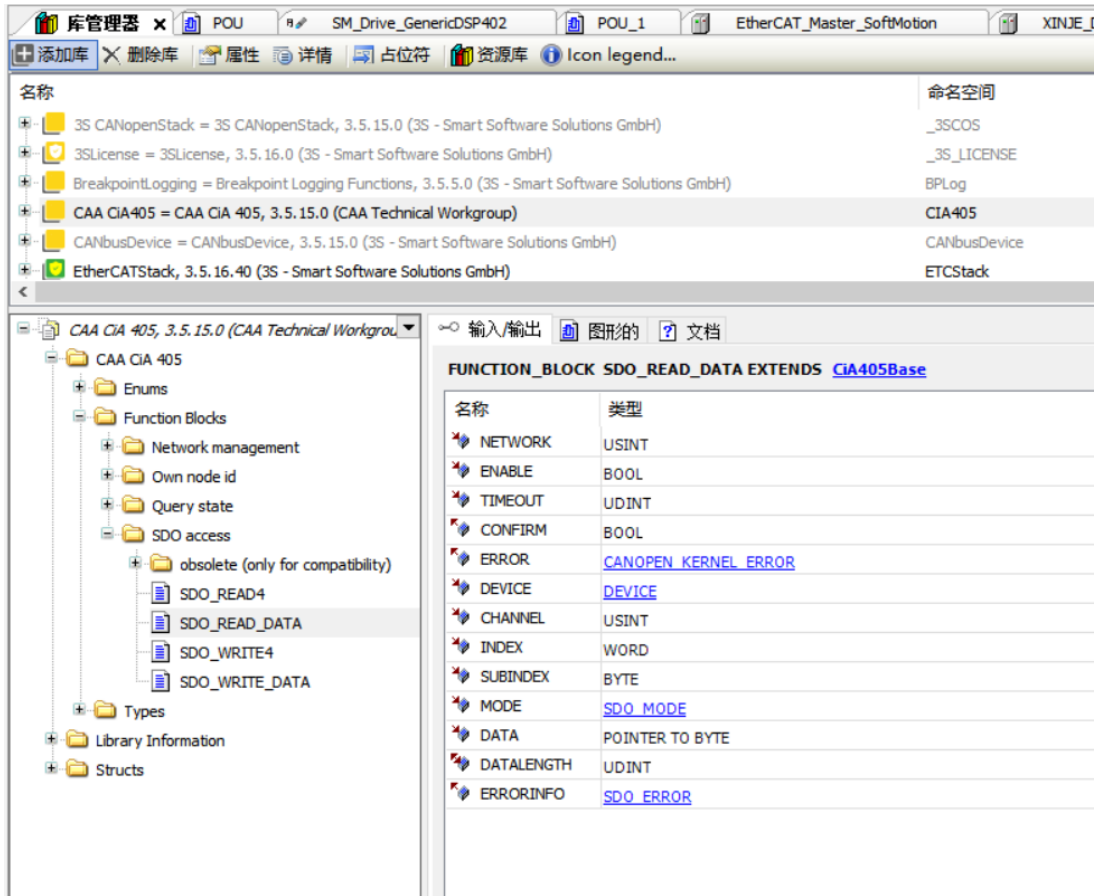
可以给用户提供设备状态（如“运行”，“未运行”）以及设备的诊断信息。



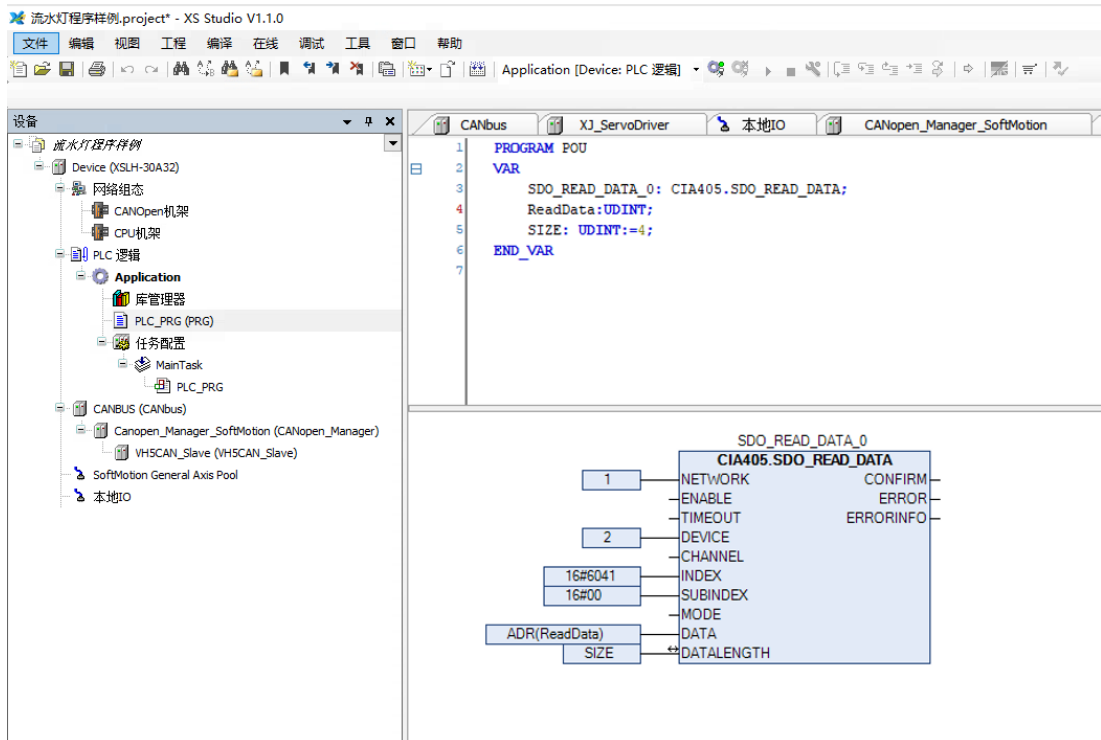
### (6) SDO 通信功能块

采用 PDO 方式做数据交换，简单且直接。但受到数量限制，而且这些数据都会占用总线，会造成连接在总线上的设备不能太多。SDO 通信主要是用于主节点对从节点的参数配置，用来设备之间传输的低优先级数据。

- 若使用 SDO 通信方式，需要添加库“CAA CIA405”，添加后即可在该库文件中看到“SDO access”文件夹。

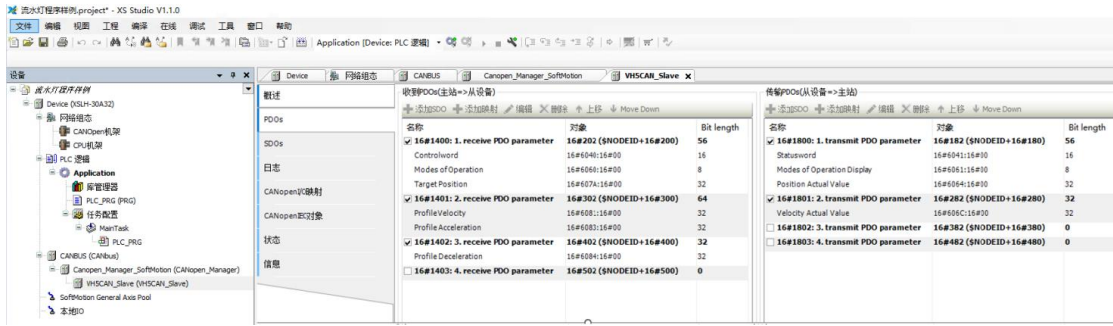


比如：添加功能块“CIA405.SDO\_READ\_DATA”，可以通过程序功能块并读取该参数。

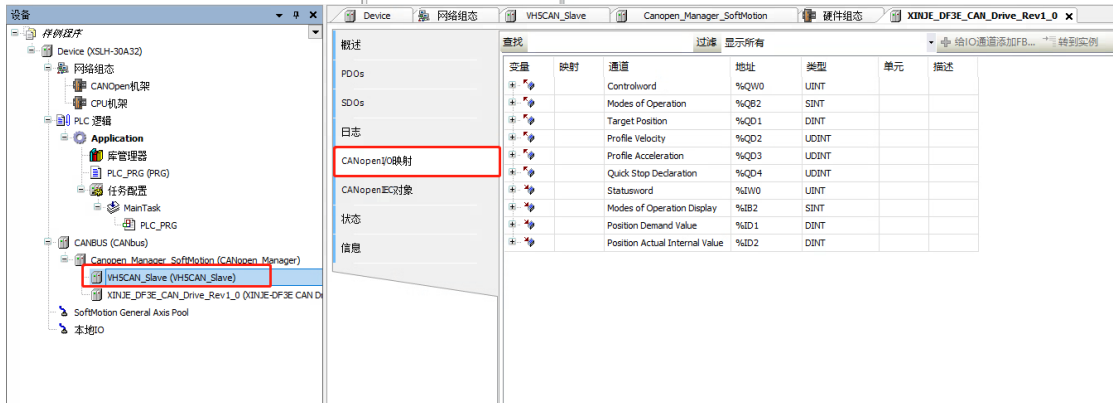
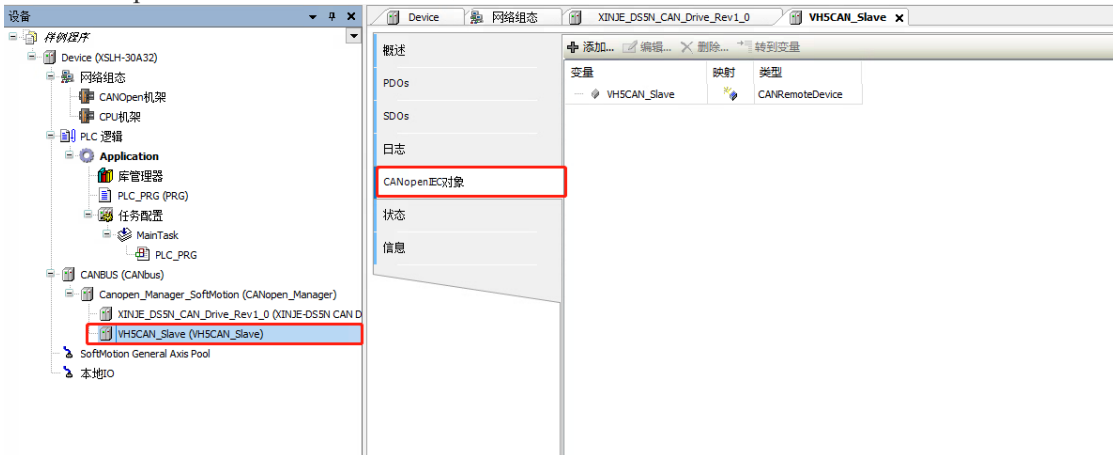


### 3-5-2. 应用举例

例一：以信捷 DS5N1 伺服作为从站，设置为 PP 模式为列，在“VH5CAN\_Slave”里的 PDOs 界面里配置 TxPDO 和 RxPDO 的对象绑定，这里绑定 PP 模式的几个常用的对象，如有其它需要可自行添加，完成绑定后需启用配置的 PDO。具体配置如下图所示：



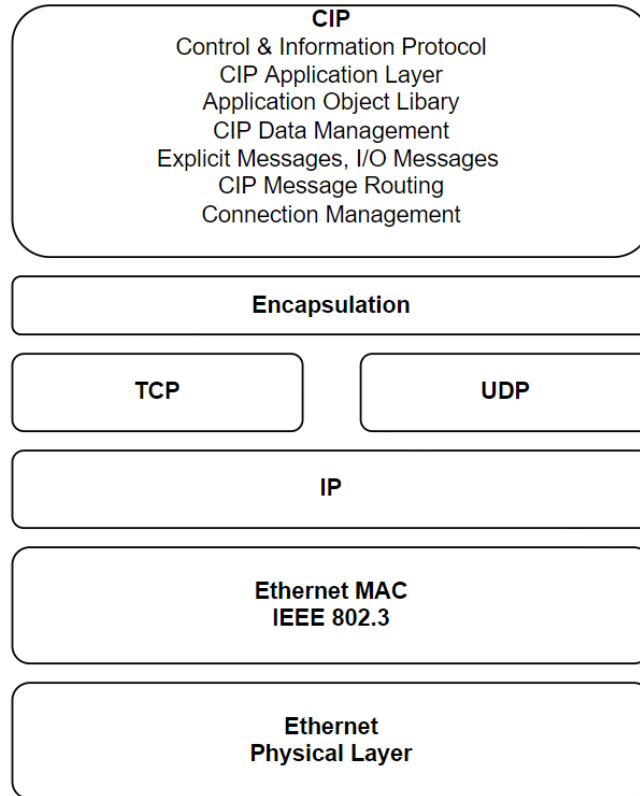
可在“CANopenI/O 映射”界面看到参数的映射地址。可根据需要设置“一直更新变量”。



可在此界面地址里的%QB2 设置为 1 (PP 模式)，修改%QW0 (控制字为 6040h) 为 0X6-->0X7-->0XF/0X4F，可以令从站使能，通过设置给定位置、速度、加减速等参数后修改控制字 0XF-->0X1F 实现绝对位置运动，0X4F-->0X5F 实现相对位置运动。其它监控参数由%IW0 开始的地址监控。

### 3-6. EtherNet/IP 通讯

Ethernet/IP 是一个面向工业自动化应用的工业应用层协议。是 2000 年 3 月由 Control Net International 和 ODVA (Open DeviceNet Vendors Association) 共同开发的工业以太网标准。它建立在标准 UDP/IP 与 TCP/IP 协议之上, 利用固定的以太网硬件和软件, 为配置、访问和控制工业自动化设备定义了一个应用层协议。各层结构如图所示:



Ethernet/IP 实现实时性的方法是在 TCP/IP 层之上增加了用于实时数据交换和运行实时应用的 CIP 协议 (Common Industrial Protocol)。

EtherNet/IP 协议的技术特点:

- Ethernet/IP 实现实时性的方法是在 TCP/IP 层之上增加了用于实时数据交换和运行实时应用的 CIP 协议 (Common Industrial Protocol)。

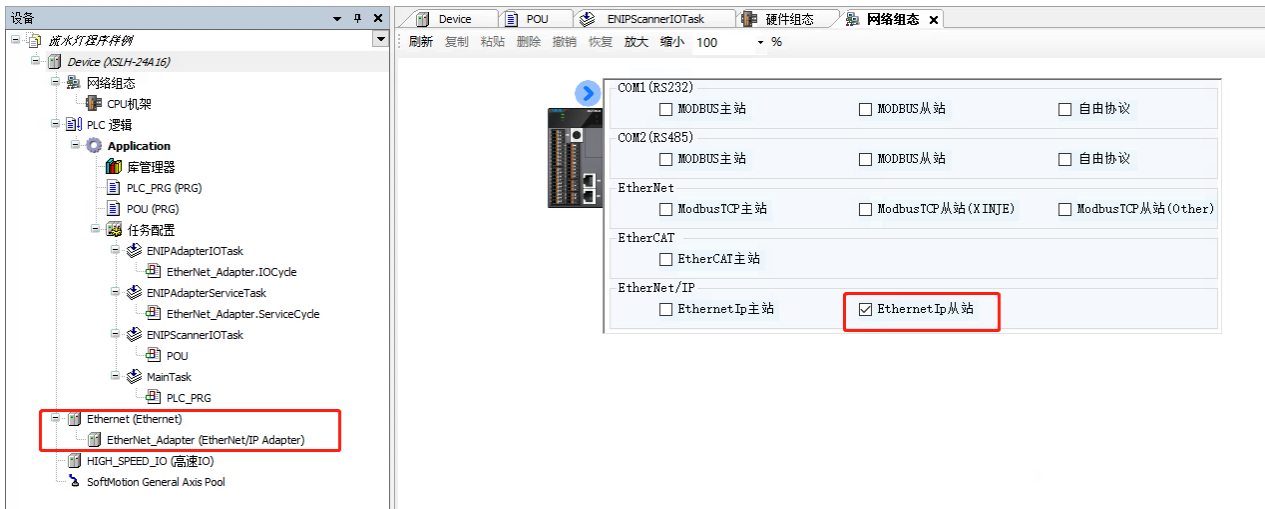
- Ethernet/IP 在物理层和数据链路层采用标准的以太网技术, 在网络层和传输层使用 IP 协议和 TCP、UDP 协议来传输数据。UDP 是一种非面向连接的协议, 它能够工作在单播和多播的方式, 只提供设备间发送数据报的能力。对于实时性很高的 I/O 数据、运动控制数据和功能行安全数据, 使用 UDP/IP 协议来发送。而 TCP 是一种可靠的、面向连接的协议。对于实时性要求不是很高的数据 (如参数设置、组态和诊断等) 采用 TCP/IP 协议来发送。

- Ethernet/IP 采用生产者/消费者数据交换模式。生产者向网络中发送有唯一标识符的数据包。消费者根据需要通过标识符从网络中接收需要的数据。这样数据源只需一次性地把数据传到网上, 其它节点有选择地接收数据, 这样提高了通信的效率。

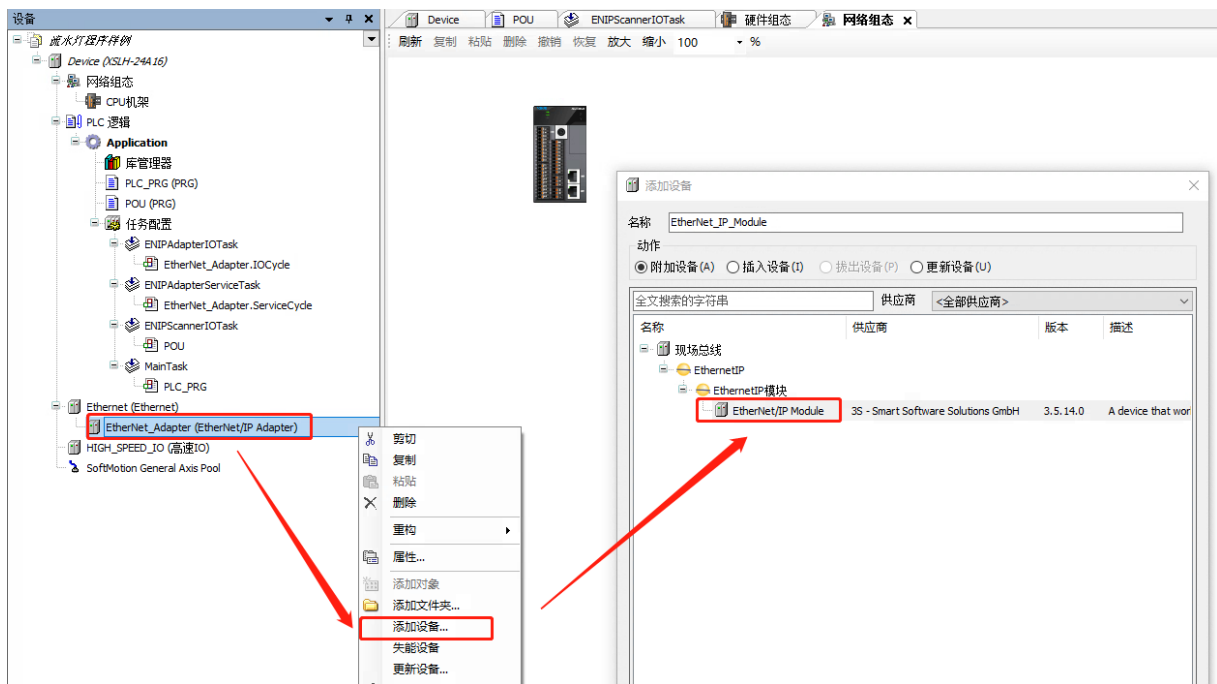
- Ethernet/IP 是在 CIP 这个协议的控制下实现非实时数据和实时数据的传输。CIP 是一个提供工业设备端到端的面向对象的协议, 且独立于物理层及数据链路层, 这使得不同供应商提供的设备能够很好的交互。另外, 为了获得更好的时钟同步性能, 2003 年 ODVA 将 IEEE 1588 引入 Ethernet/IP, 并制定了 CIPsync 标准以提高 Ethernet/IP 的时钟同步精度。

### 3-6-1. EtherNet/IP 作为从站的样例

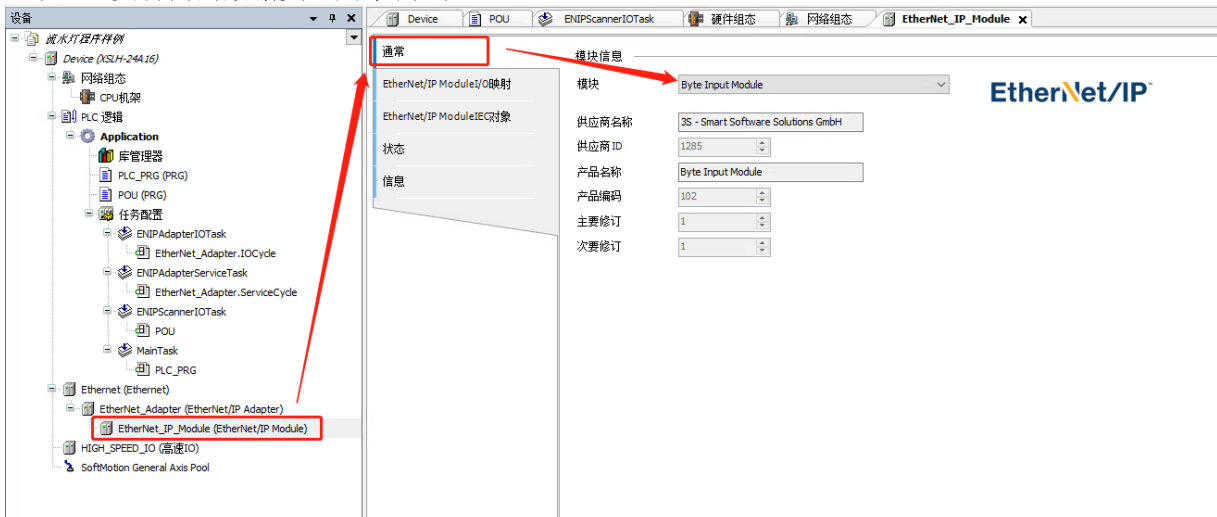
1、鼠标单击网络组态内的使能窗口，勾选“EtherNet IP 从站”。左侧设备树节点下会自动添加“EtherNet/IP Adapter”



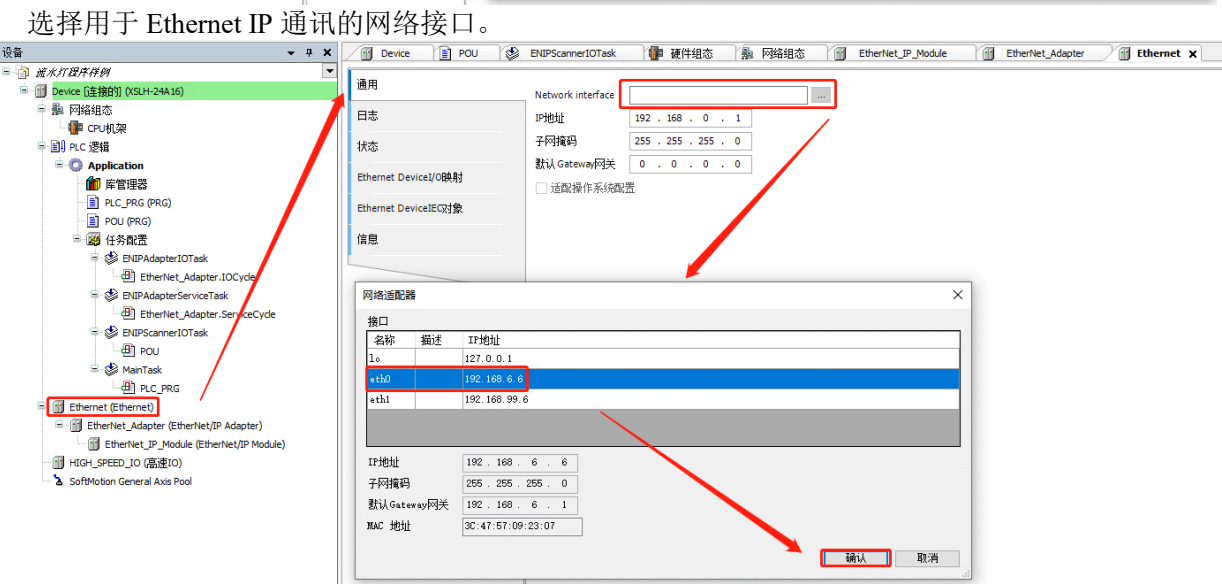
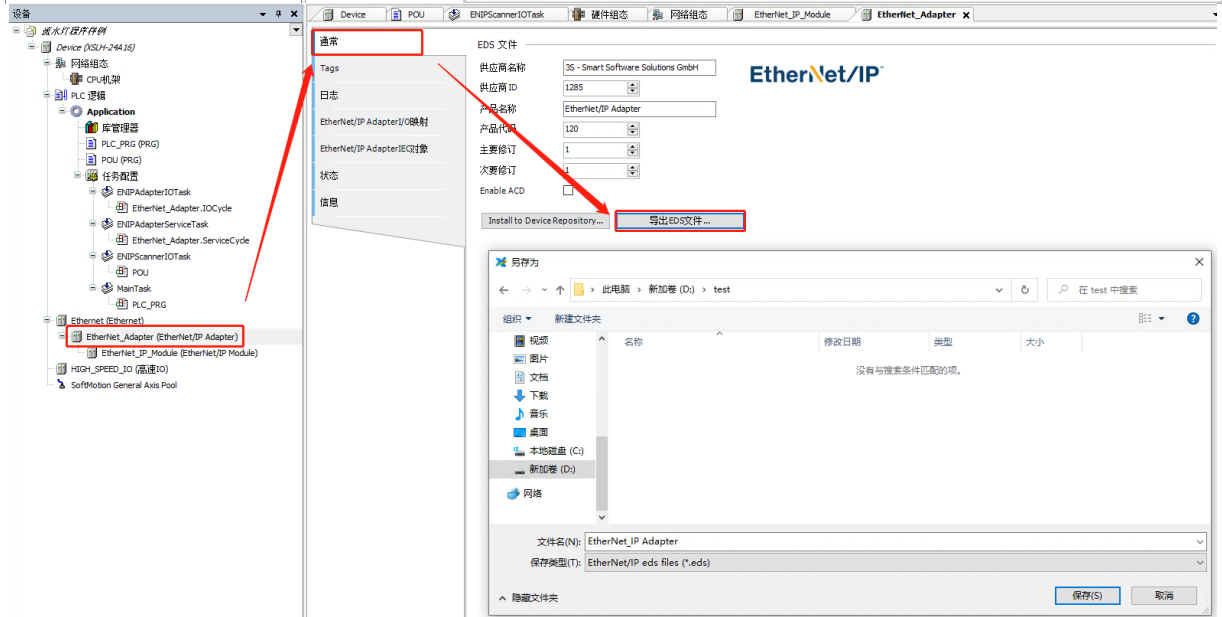
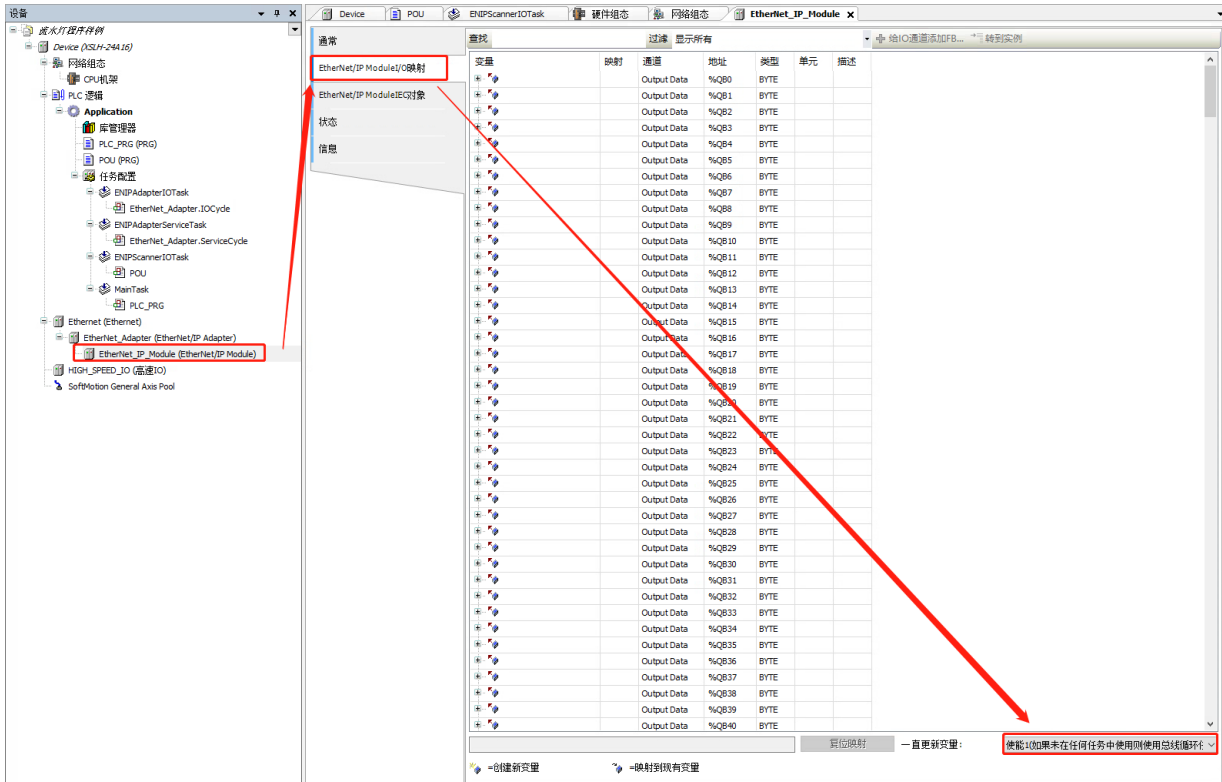
2、右击“EtherNet/IP Adapter” → 添加“EtherNet/IP Module”。



添加想要访问的数据类型用来测试。





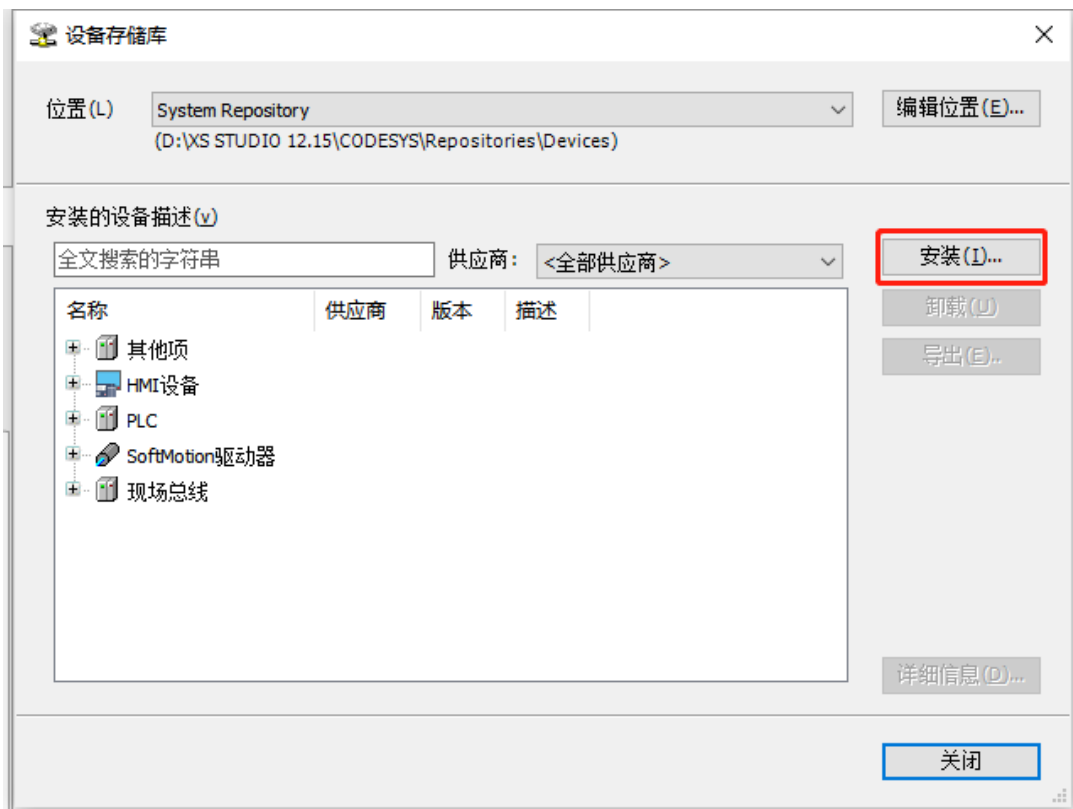
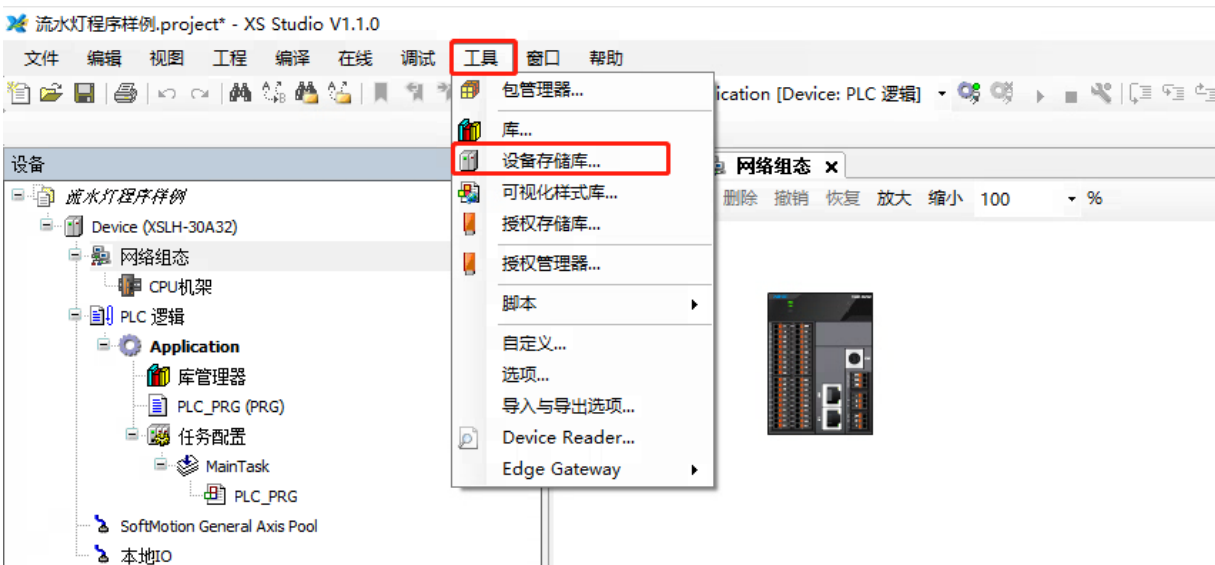


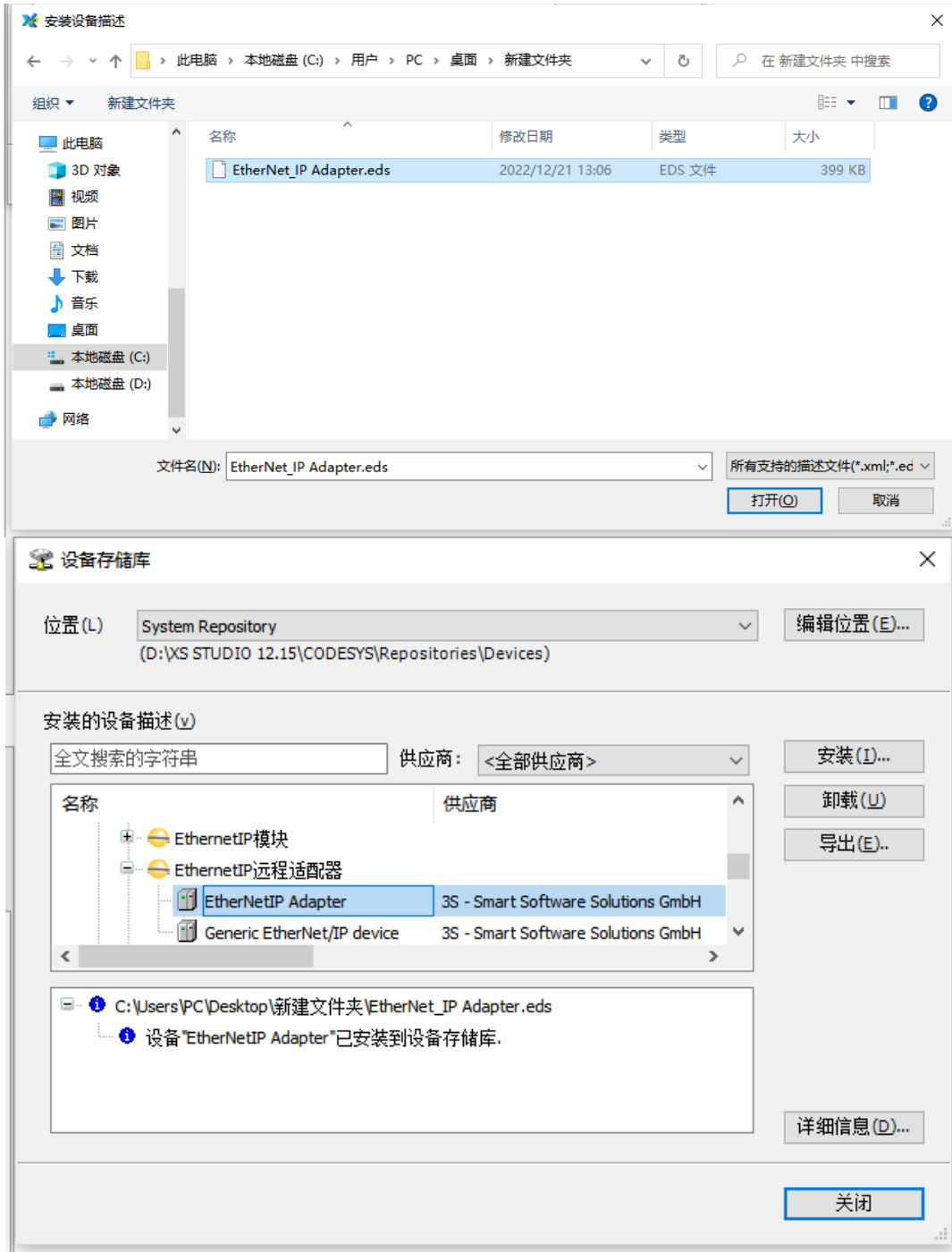
总线循环设置改为“ENIPAdapterIOTask”:



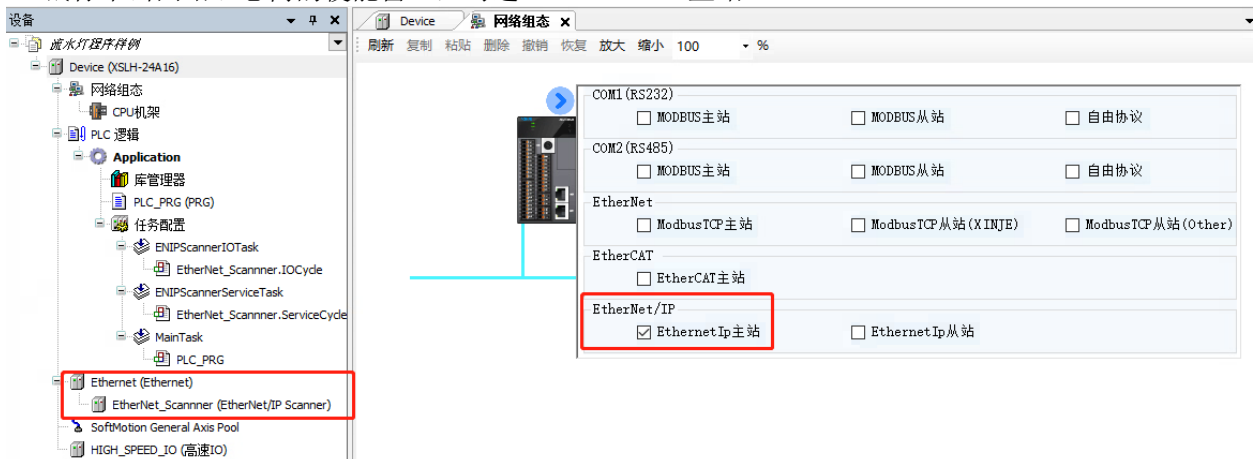
### 3-6-2. EtherNet/IP 作为主站的样例

1、工具→设备存储库→安装→打开刚才导出的 EDS 文件→如图，添加完成。

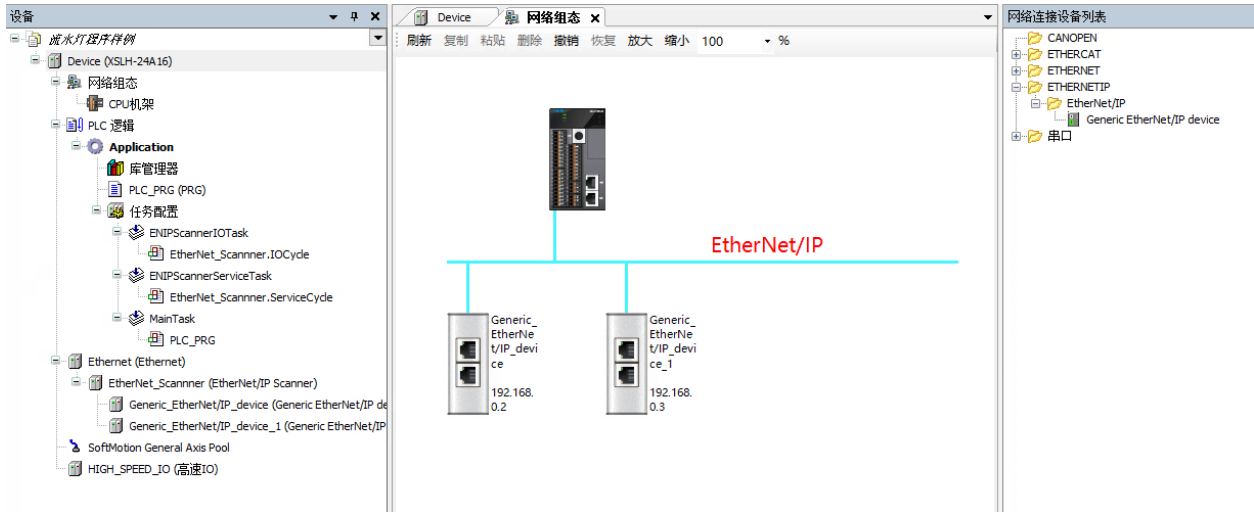




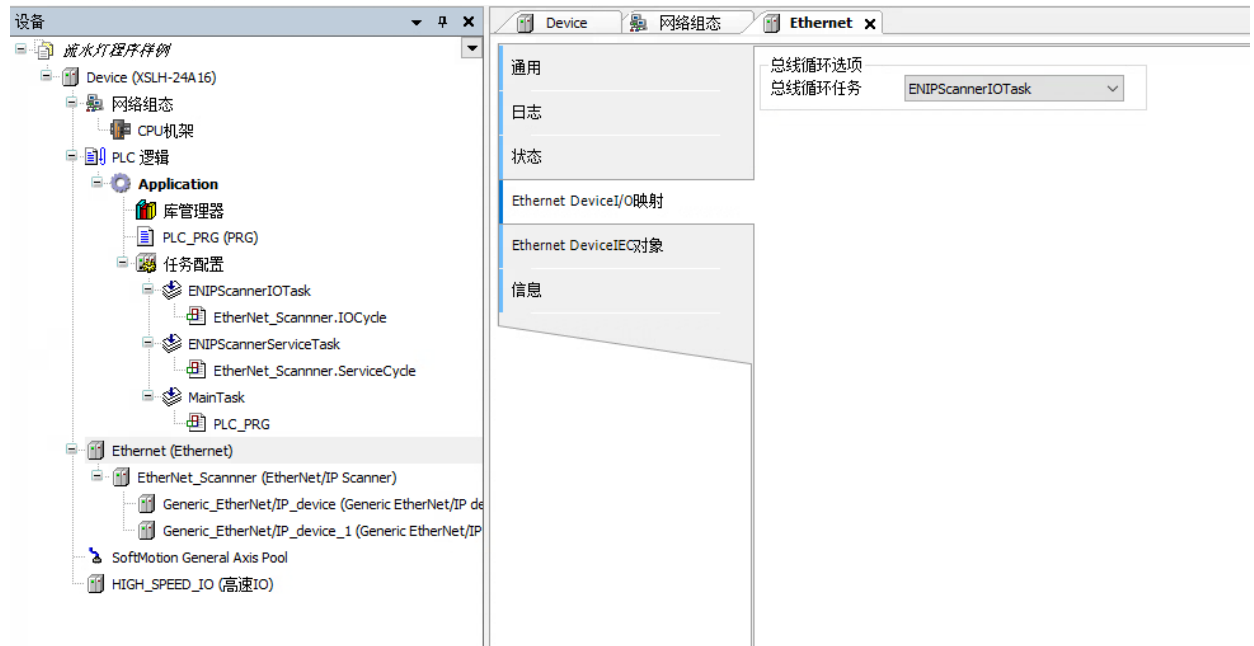
鼠标单击网络组态内的使能窗口，勾选“EtherNet IP 主站”。



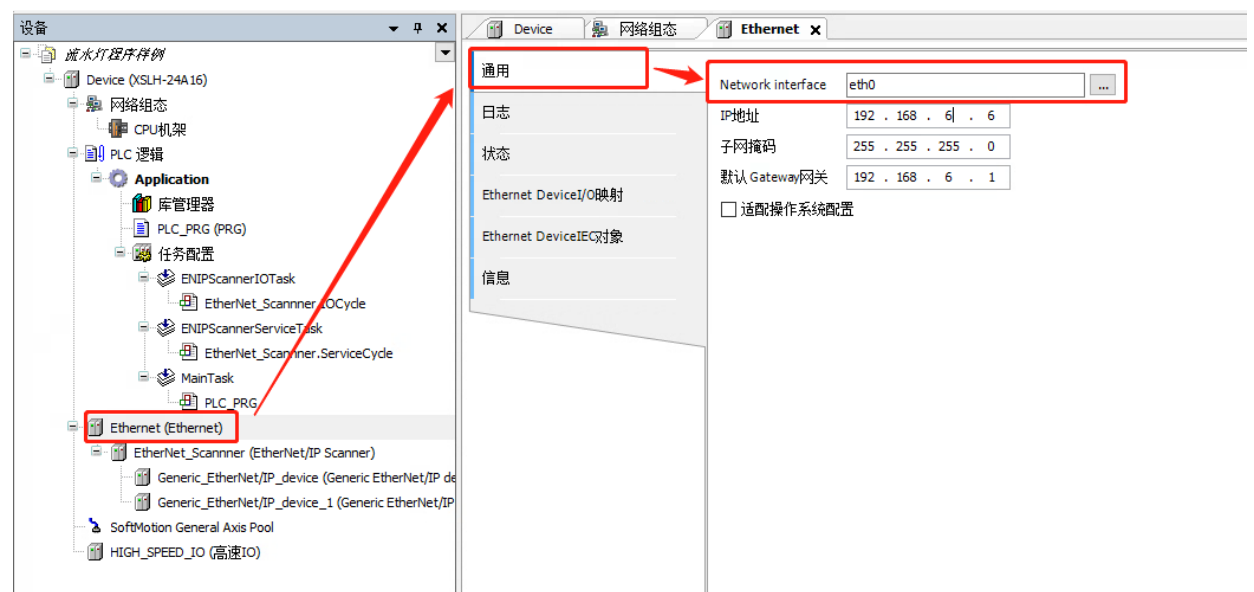
从右侧“网络连接设备列表”添加从站设备，如图所示：



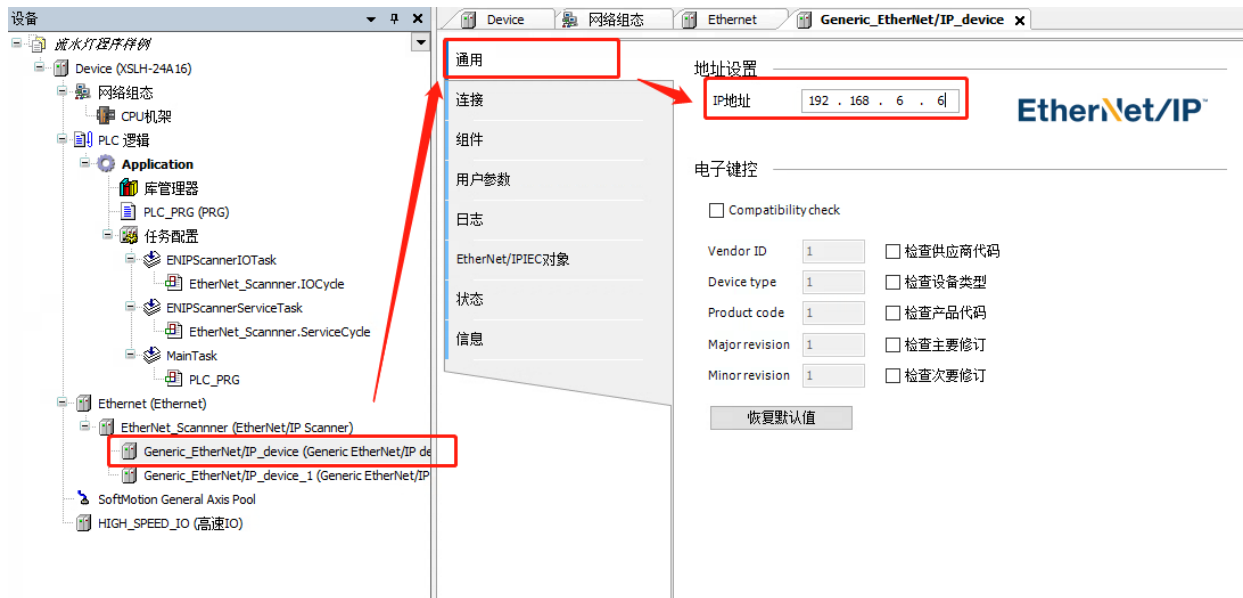
将总线循环任务设置为统一的配置：



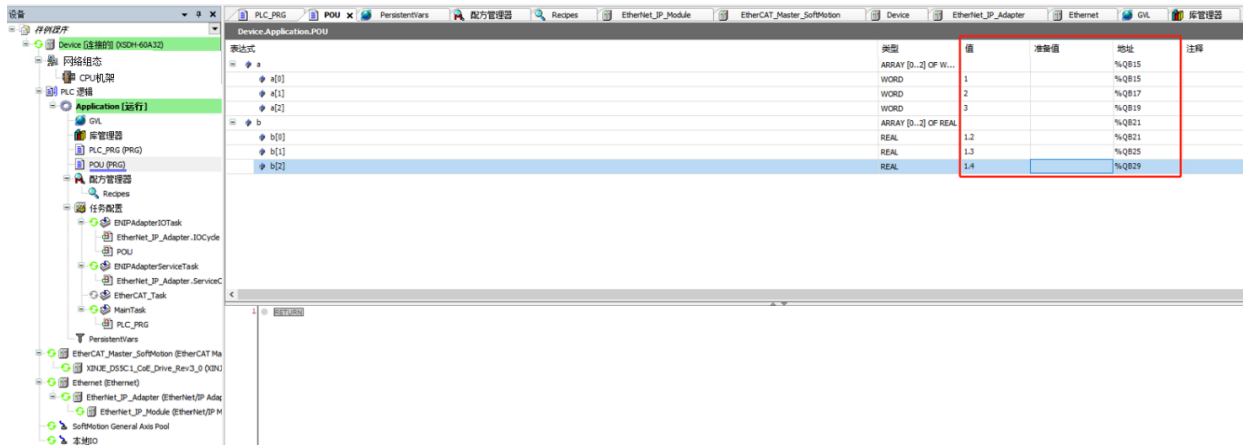
选择主站网口：



设置从站 IP：



通讯测试：  
定义并关联变量，将程序下载进 PLC 中。



通讯成功。

## 3-7. OPC UA 通讯

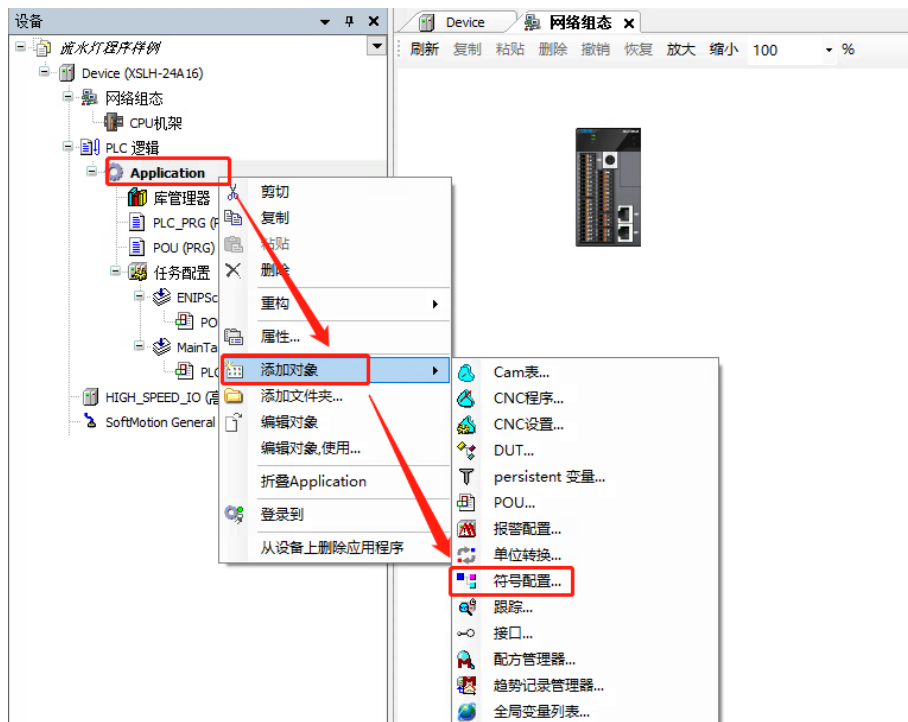
### 3-7-1. 通讯概述

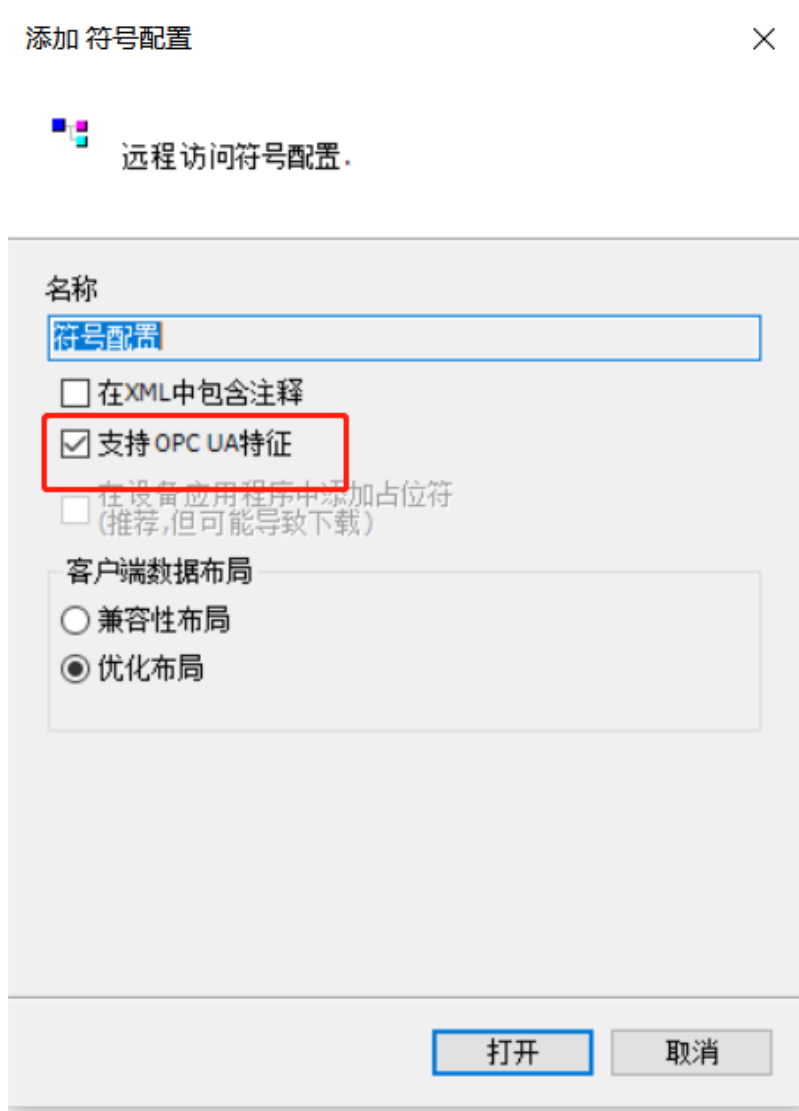
OPC UA (OPC Unified Architecture, 基于 OPC 统一架构的时间敏感网络技术), 建立支持网络间互操作的时间敏感机制, 突破性实现信息技术 (IT) 与操作技术 (OT) 在物理层、数据链层、网络层、传输层、会话层、表达层和应用层全面融合的技术。该技术基于国际电工委员会 (IEC) 和电气和电子工程师协会 (IEEE) 国际标准搭建, 可为工业互联网网络体系构建提供标准化模块, 是建立从传感器到云端大带宽、高同步、广兼容通讯的关键技术。

OPC UA 实质上是一种抽象的框架, 是一个多层架构, 其中的每一层完全是从其相邻层抽象而来。这些层定义了线路上的各种通信协议, 以及能否安全地编码/解码包含有数据、数据类型定义等内容的讯息。利用这一核心服务和数据类型框架, 人们可以在其基础上 (继承) 轻松添加更多功能。

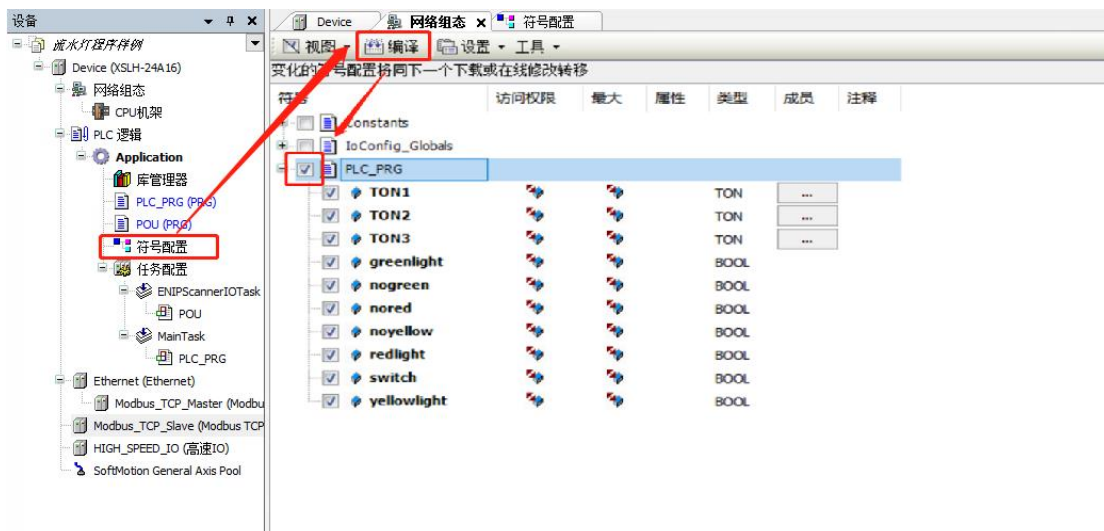
### 3-7-2. 参数配置

① 在应用的工程中, 右击 “Application”, 选择 “Add Object” - “Symbol Configuration..”, 在弹出的对话框中勾选 “Support OPC UA feature” 进行添加, 则开启 OPC UA 的功能。





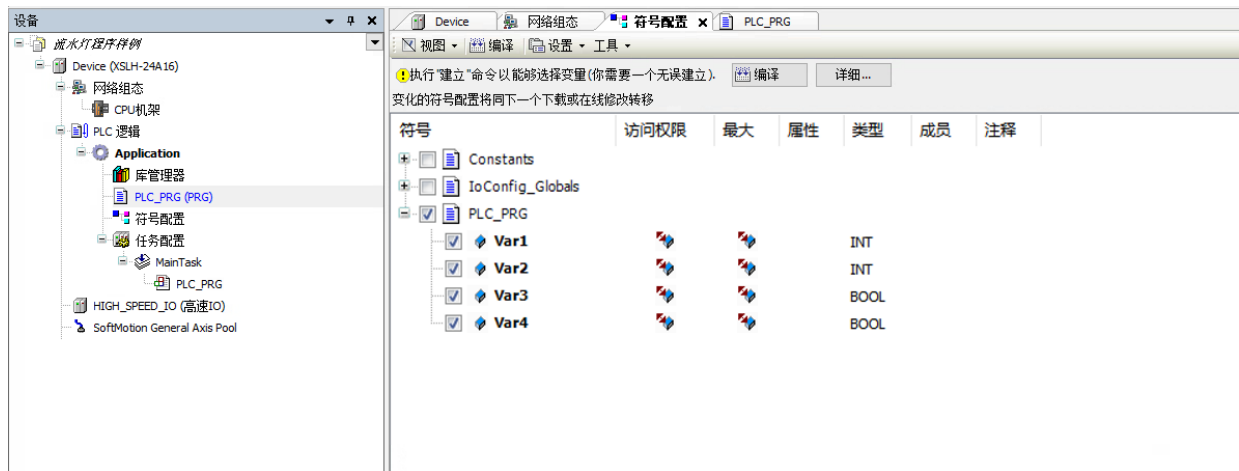
② 双击“符号配置”，在弹出的界面里点击“编译”后，勾选需要监控的参数。



## 3-7-3. OPC UA 样例

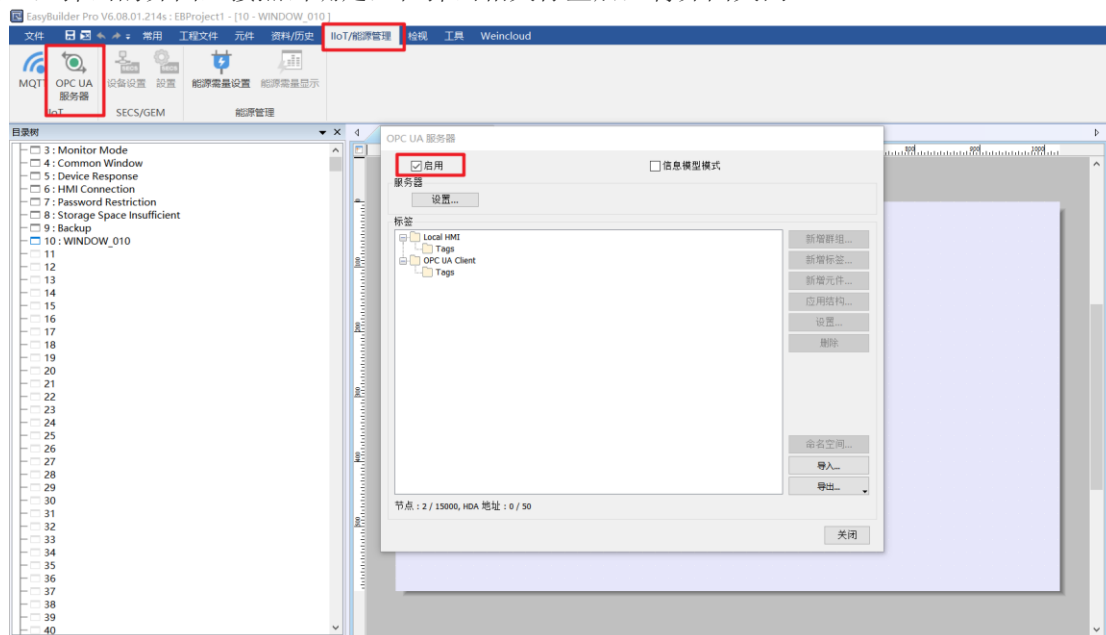
例一：使用信捷 XSLH-24A16 和威纶通（型号为 CMT3105X）的触摸屏进行 OPC UA 通讯。  
程序编写：

(1) 在 XSLH-24A16 中写了几个参数，并在 OPC UA 界面中勾选登录下载。



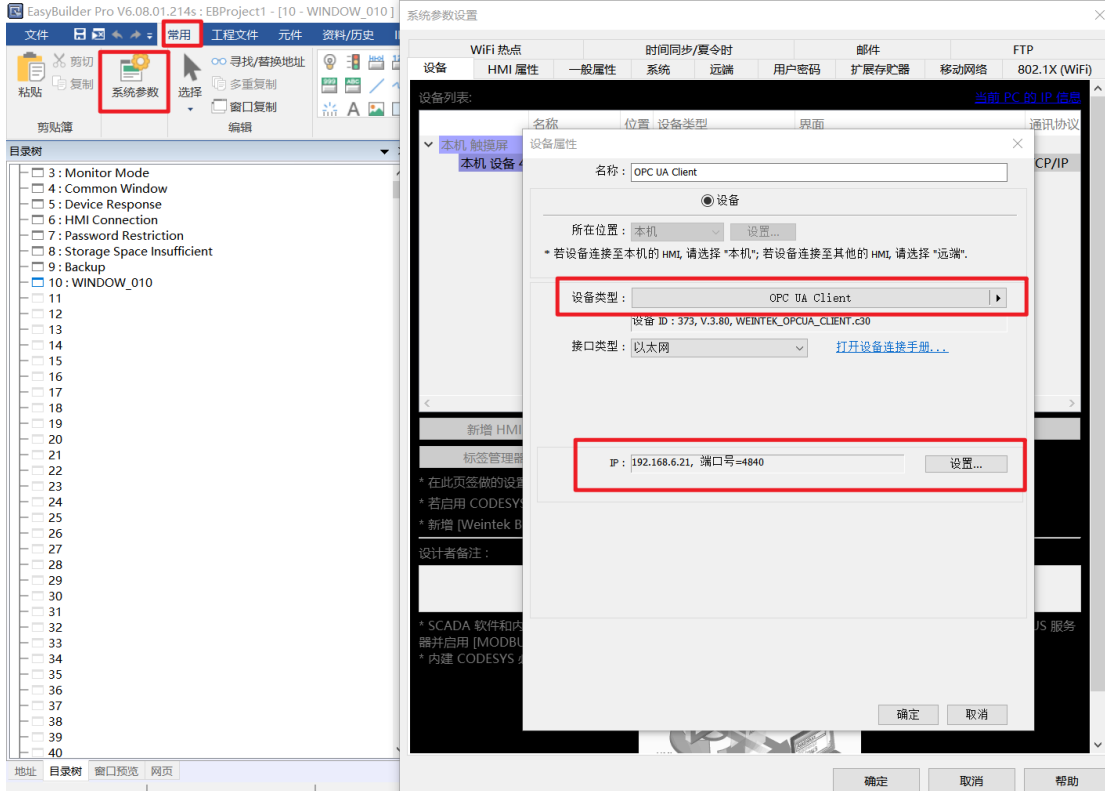
(2) 威纶通触摸屏配置：

① 在“IIOT 能源管理”界面中选择“OPCUA 服务器”，在点开的“OPCUA 服务器”界面中勾选“启用”，弹出的界面直接点击确定，在弹出相关标签后，将界面关闭。

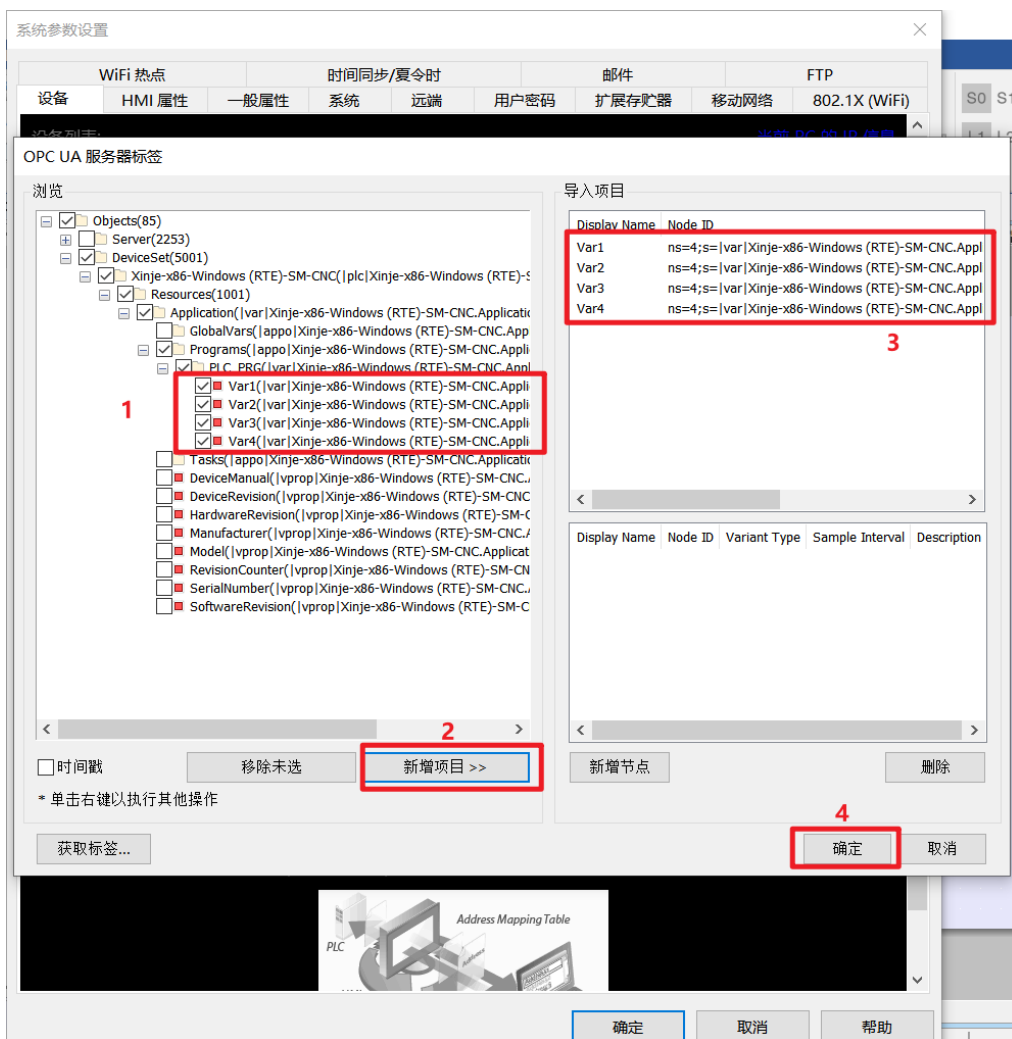




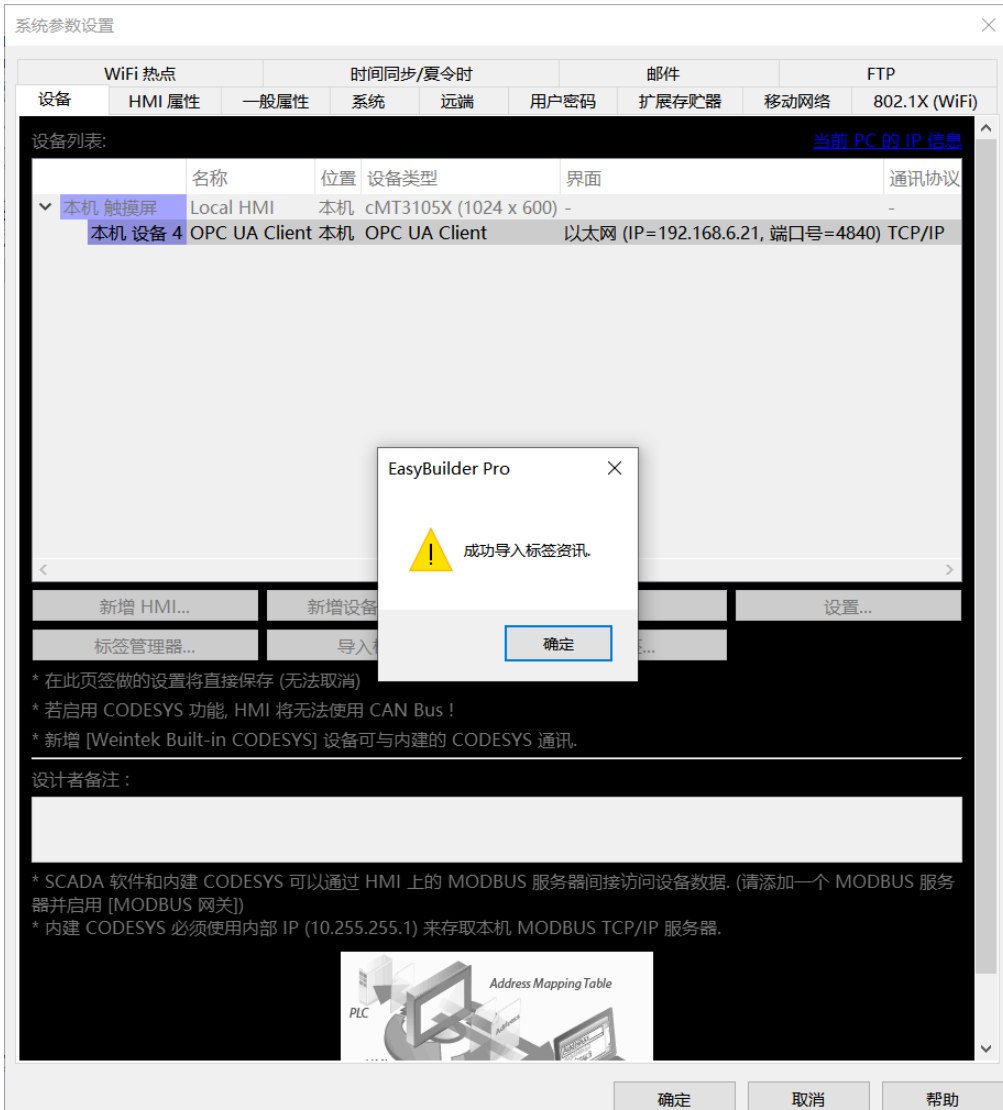
② 在“系统参数设置”界面中，点击“新增设备/服务器”，在弹出来的“设备属性”界面中选择设备类型为“OPC UA Client”，并设置 IP 为 XSLH-24A16 的 IP 地址，设置好后点击确定。



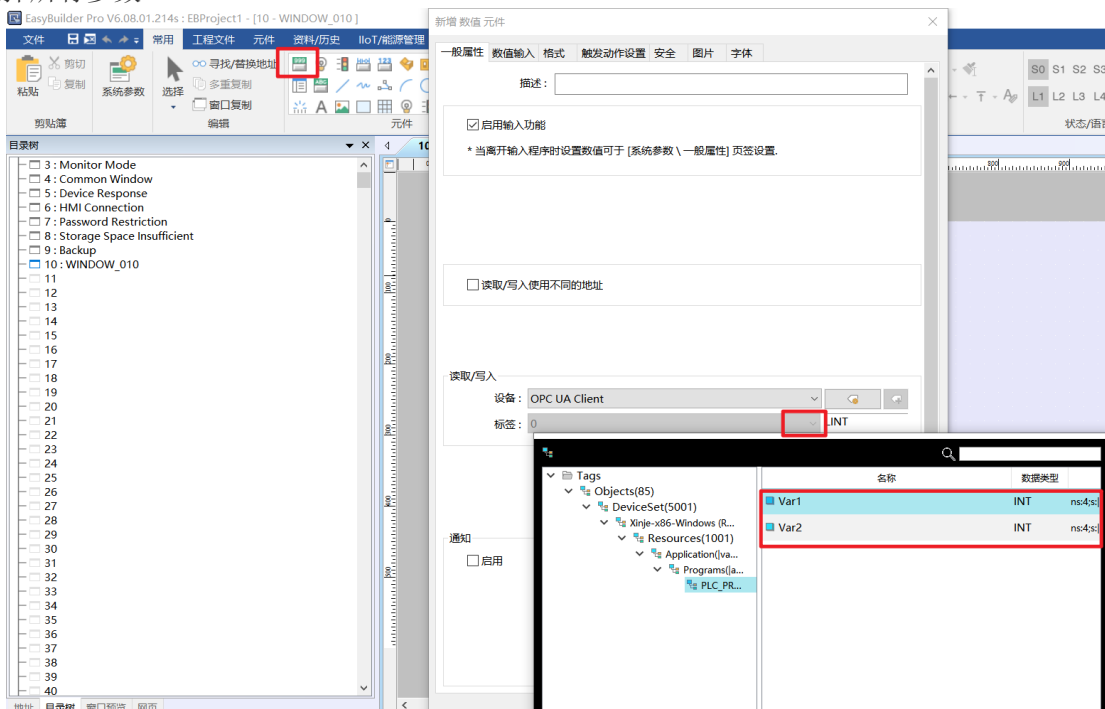
③ 点击“导入标签”，在弹出来的界面中选择确定后，会出现“OPC UA”服务器标签界面，可在此界面中选择 PLC 的相关数据。

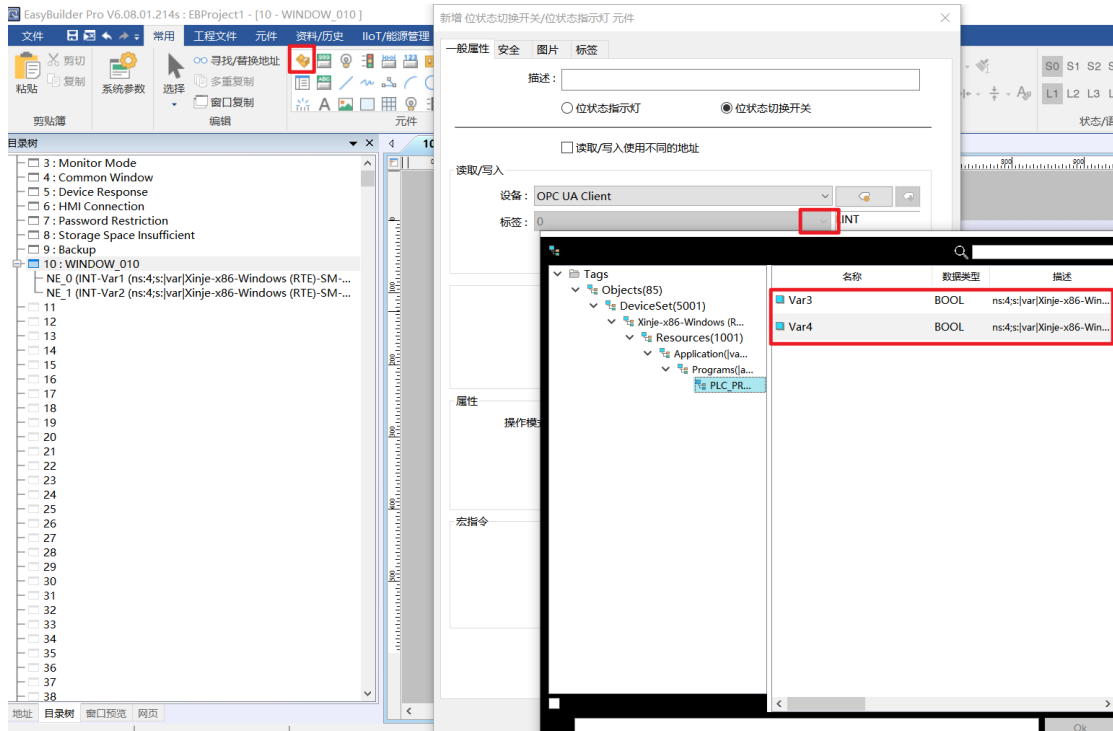


④ 点击确定，出现“成功导入标签通讯”。

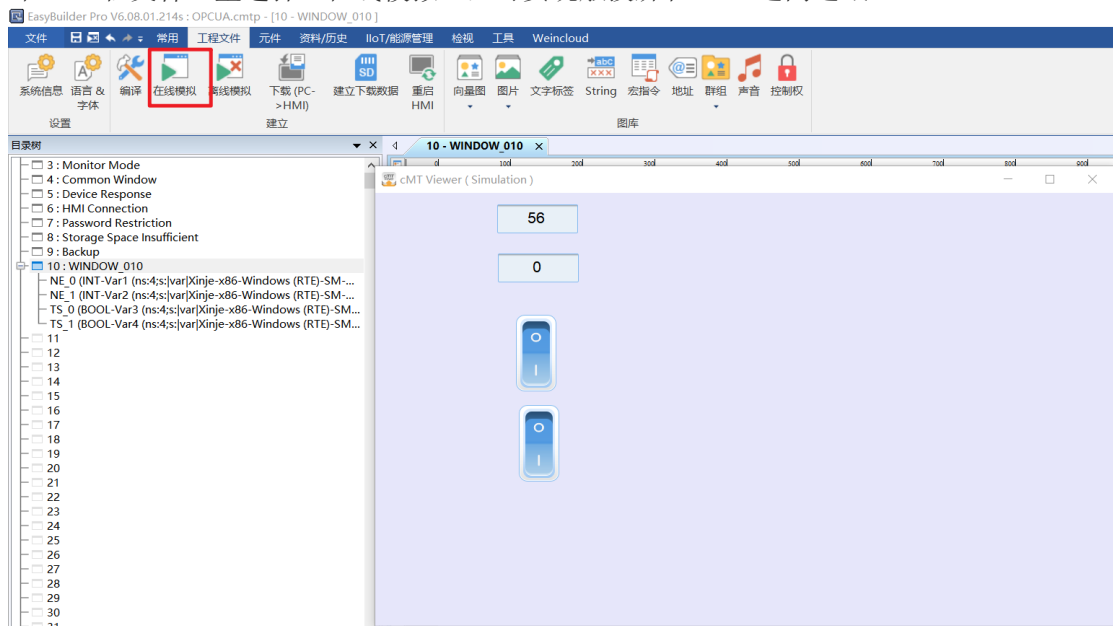


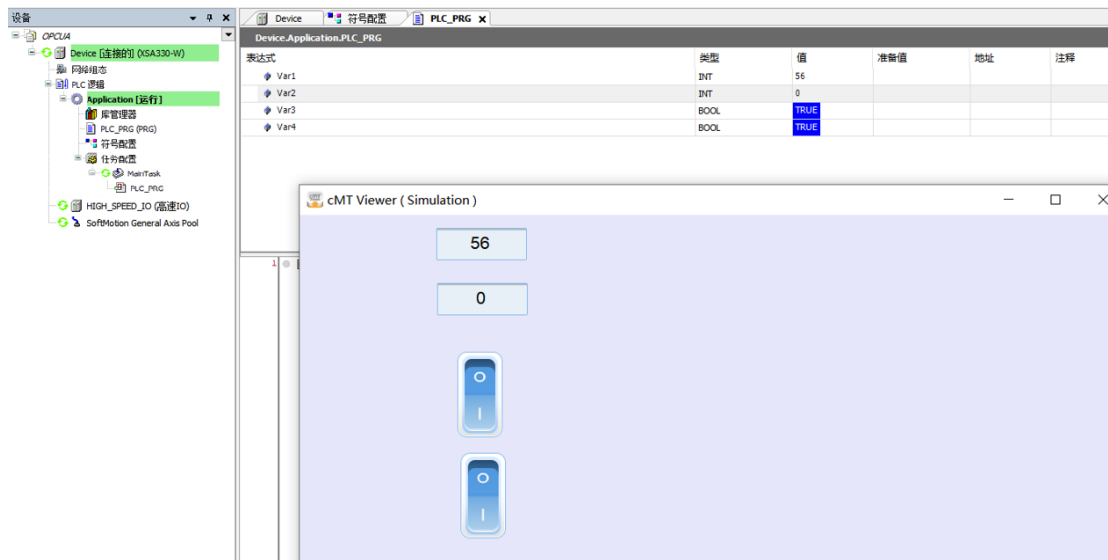
⑤ 在“元件”里选择相关类型的元件，在弹出的界面里的“标签”点击下拉图标，出现相关的参数。依次选择所有参数。





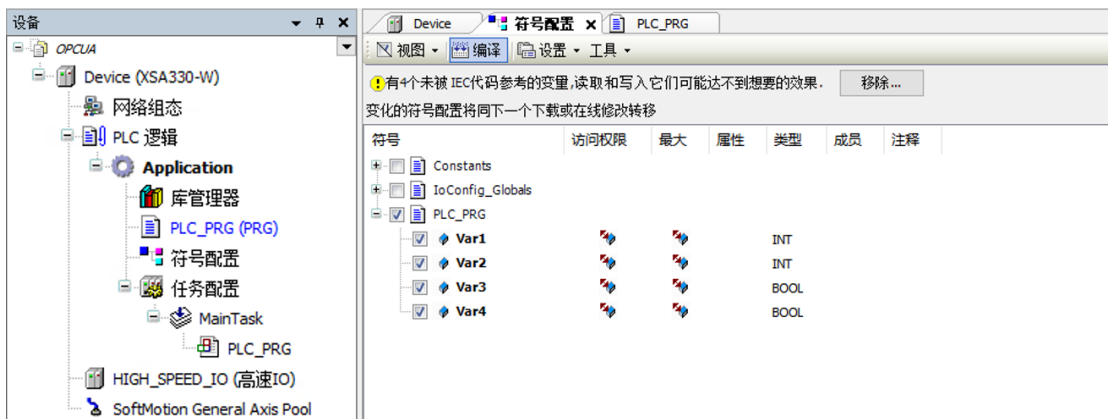
⑥ 在“工程文件”里选择“在线模拟”，可实现触摸屏和 PLC 之间通讯。





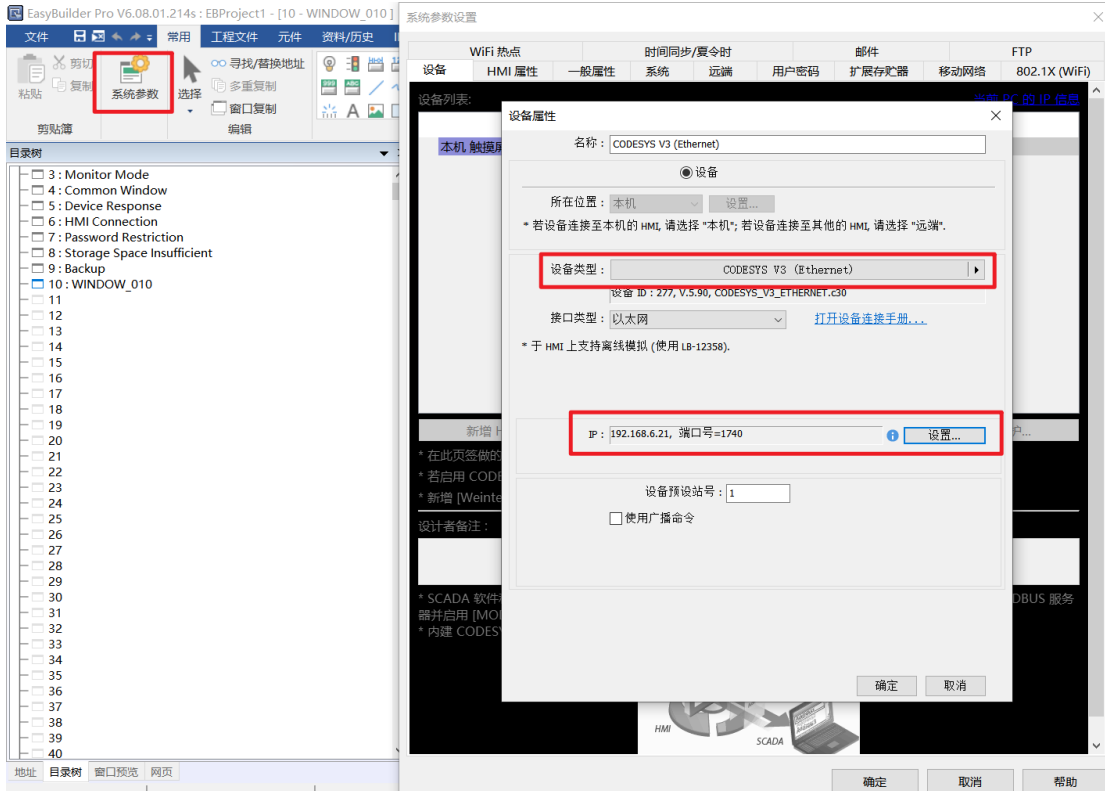
例二：使用信捷 XSA330-W 和威纶通（型号为 CMT3105X）的触摸屏进行“codesys v3”通讯。程序编写：

(1) 在 XSA330-W 中写了几个参数，并在 OPC UA 界面中勾选登录下载。

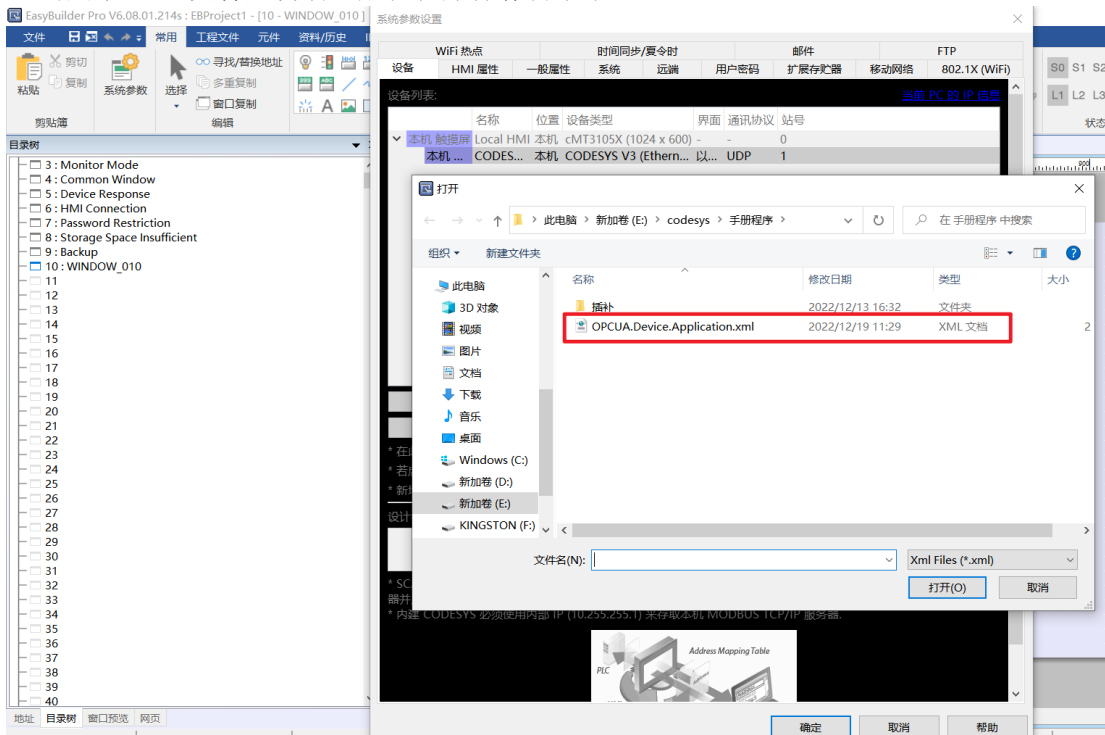


(2) 威纶通触摸屏配置：

①在“系统参数设置”界面中，点击“新增设备/服务器”，在弹出来的“设备属性”界面中选择设备类型为“CODESYS Automation Alliance”中的“CODESYS V3”，并设置 IP 为 XSA330-W 的 IP 地址，设置好后点击确定。

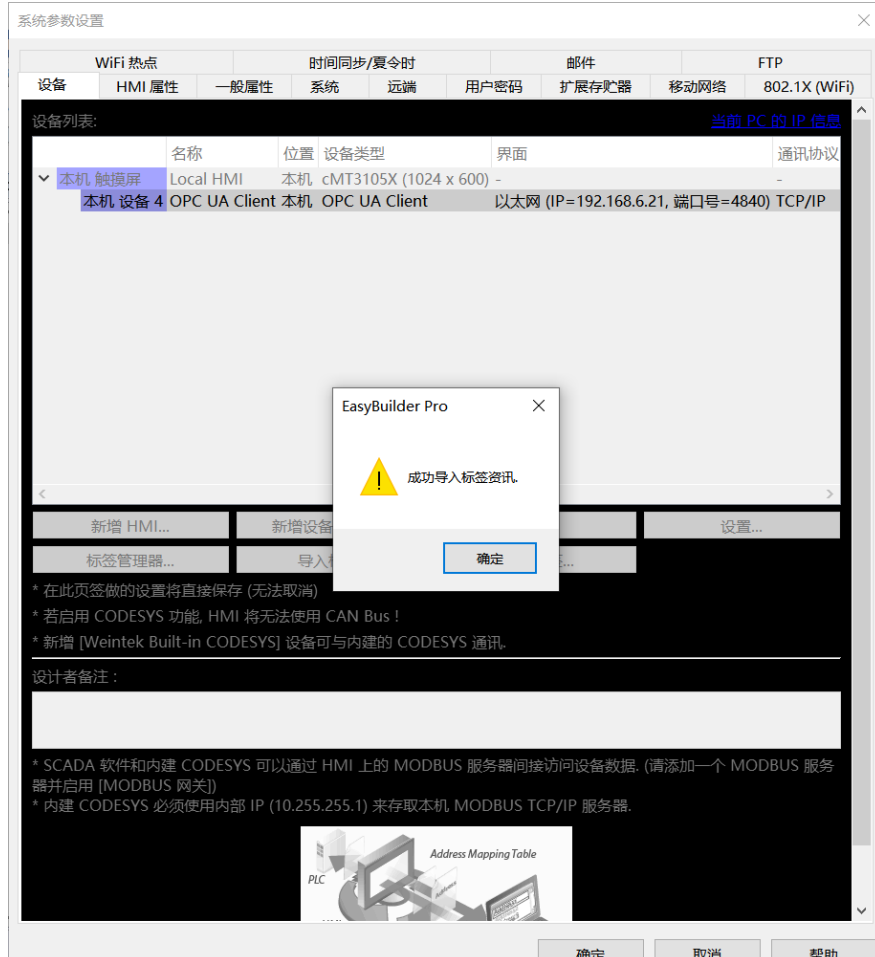


②点击“导入标签”，导入生成的 XML 文件，导入成功后会出现“成功导入标签资讯”。  
注：生成的 XML 文件是自动生成在程序保存目录中。

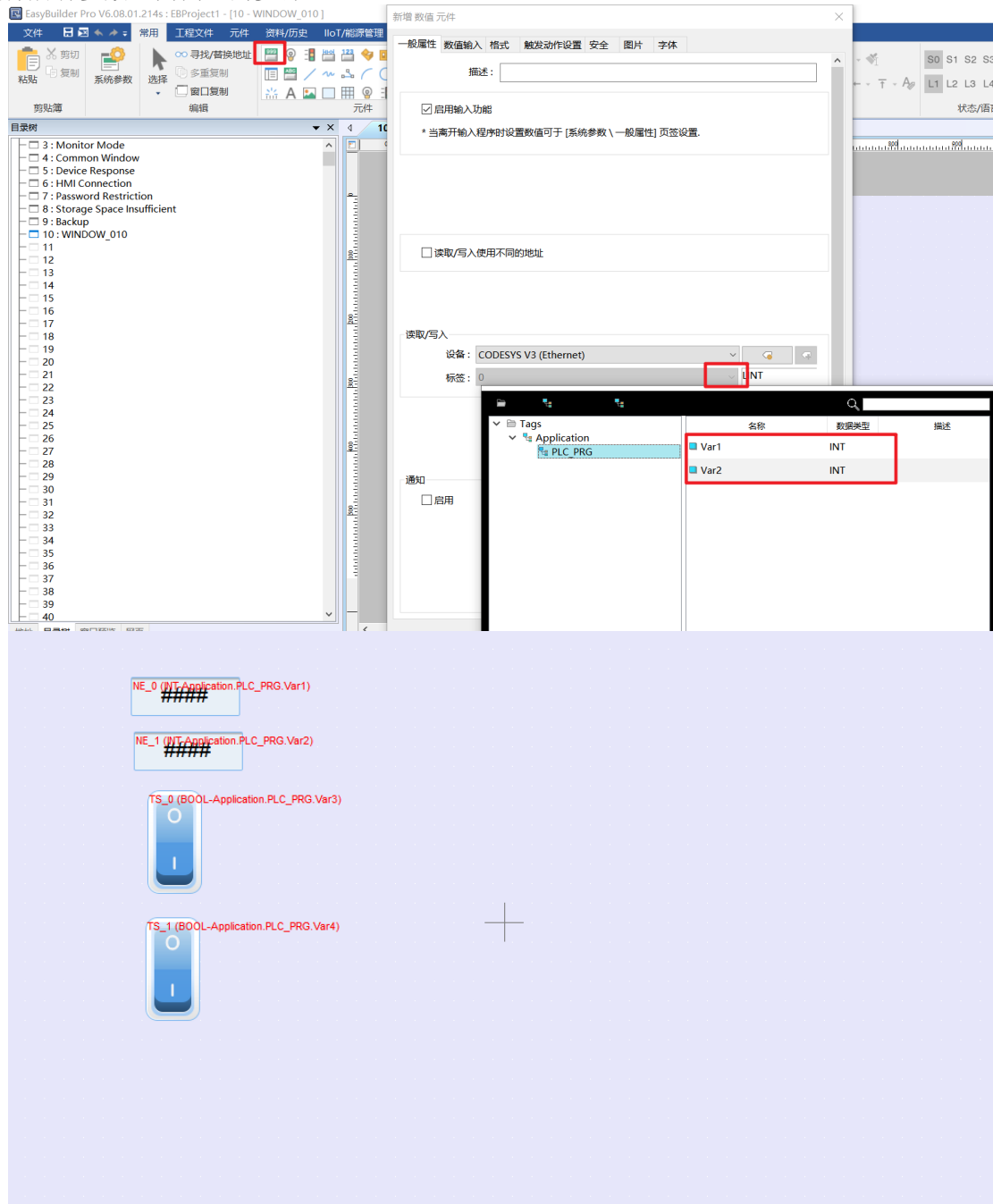




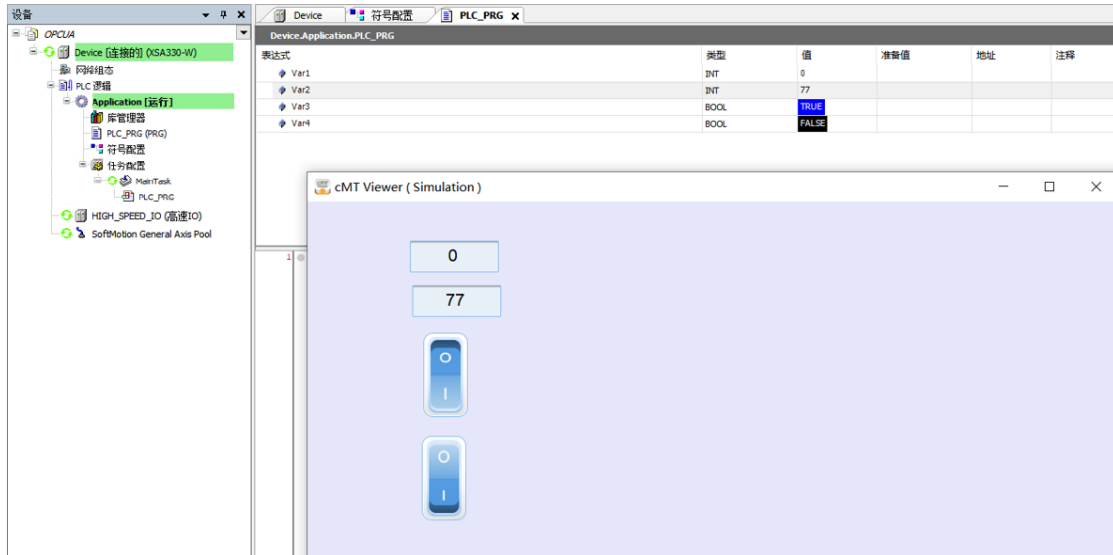
③ 点击确定，出现“成功导入标签资讯”。



④在“元件”里选择相关类型的元件，在弹出的界面里的“标签”点击下拉图标，出现相关的参数。依次选择所有参数。同例一的步骤。



⑤在“工程文件”里选择“在线模拟”，可实现触摸屏和 PLC 之间通讯。



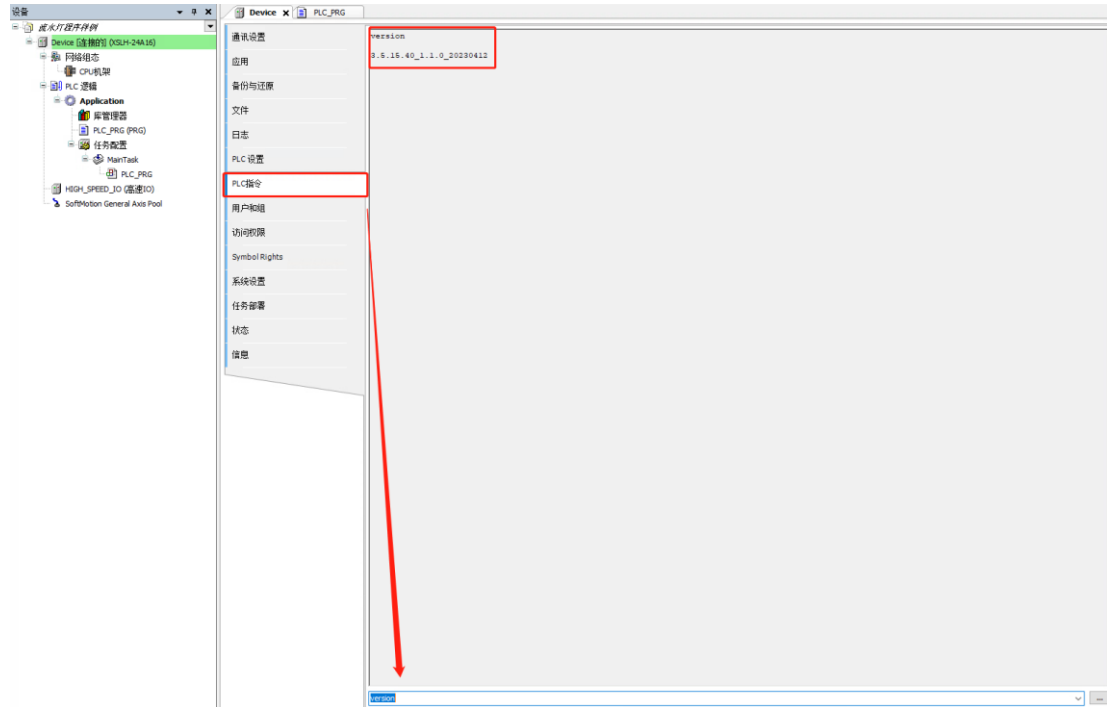


### 3-8. 信捷 Modbus TCP 协议

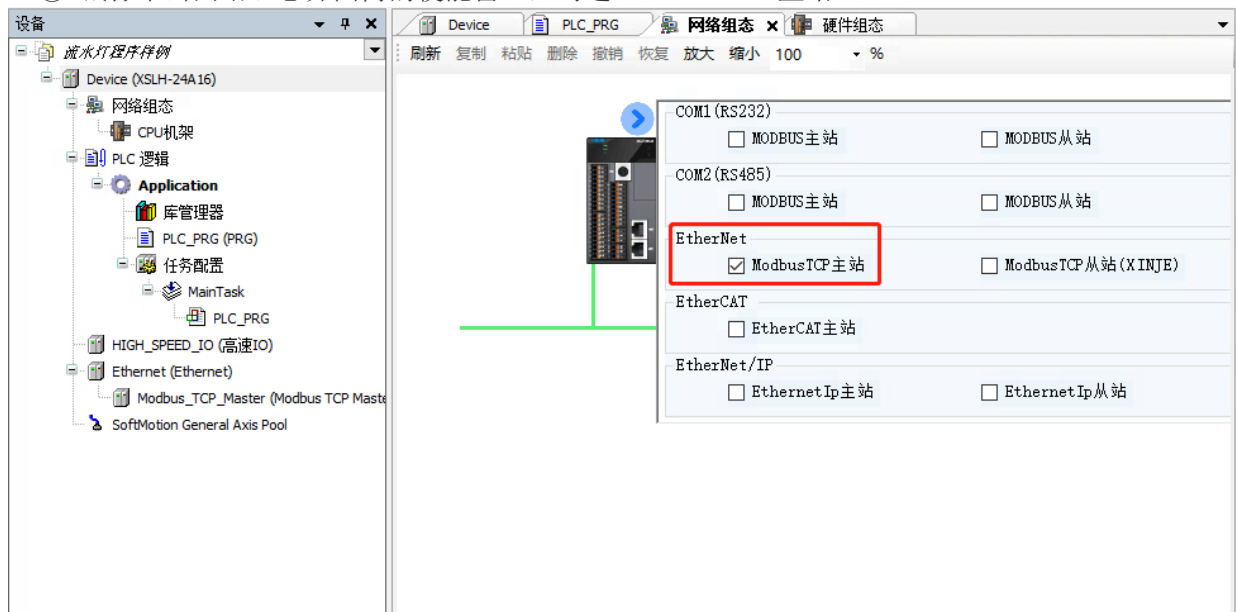
**注意：**信捷自己开发的 Modbus TCP 暂时只支持 ARM 系列的机型，后续会支持所有 PLC open 标准控制器。

#### 3-8-1. 参数配置

① 在使用此通讯功能时，请先检查 PLC 的固件版本是否为 3.5.15.40\_1.1.0\_0020412 及其以上，如果不是此版本，请先升级固件。



② 鼠标单击网络组态界面内的使能窗口，勾选 Modbus TCP 主站。



③ 关联变量。

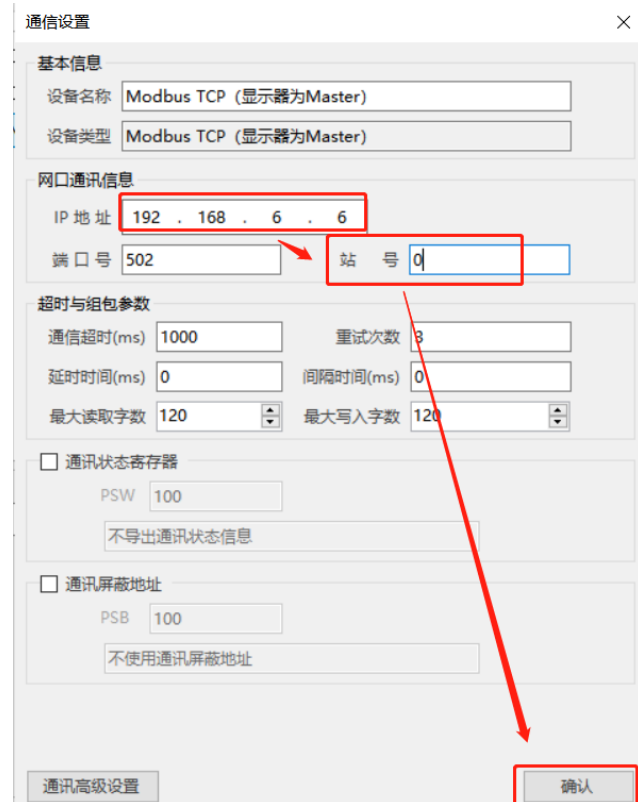
```
PROGRAM PLC
VAR
    M AT %MB0: ARRAY[0..9] OF BOOL; //触摸屏上的0X和1X是一样的, MB0等于触摸屏上的0X0和1X0
    D AT %MW40000: ARRAY[0..9] OF WORD; //单寄存器, 触摸屏上的3X和4X是一样的, MW40000等于触摸屏上的3X0和4X0
    DD AT %MW40010: ARRAY[0..9] OF REAL; //双寄存器, 触摸屏上的3X和4X是一样的, 3X10和4X10会占用MW40010+MW40011
END_VAR
```

## 3-8-2. 触摸屏设置

- ① 设置触摸屏和需要连接设备的 IP 地址（需要新建设备）。



- ② 添加所需要的触摸屏元素，设备选择 Modbus\_通用，站点号必须设为 0！



按钮或者指示灯的对象类型选择 0X（可读可写）或者 1X（只读），0X0 和 1X0 都对应 MB0，然后依次类推。

数据输入或者数据显示的对象类型选择 3X(只读)或者 4X(可读可写), 3X0 和 4X0 都对应 MW40000，然后依次类推。

如果数据类型输入或者显示的是 DWORD，那么 3X0 和 4X0 占用 MW40000、MW40001 这两个寄存器,然后依次类推。



## 4. EtherCAT 配置

XS 系列 PLC 标准控制器支持 EtherCAT 总线功能，本章从 EtherCAT 总线的基本信息、通讯规格、参数配置三方面出发，详细介绍 EtherCAT 总线配置。

---

4. EtherCAT 配置 .....	86
4-1. EtherCAT 概要 .....	87
4-1-1. 概述 .....	87
4-1-2. 系统构成 .....	87
4-1-3. 通讯规格 .....	87
4-1-4. EtherCAT 通讯连接说明 .....	88
4-2. EtherCAT 通讯规格 .....	89
4-2-1. EtherCAT 帧结构 .....	89
4-2-2. 状态机 ESM .....	89
4-2-3. 从站控制器 ESC .....	90
4-2-4. SII 区域 .....	93
4-2-5. SD0 .....	93
4-2-6. PDO .....	95
4-2-7. 通信同步模式 .....	97
4-3. EtherCAT 参数配置 .....	99
4-3-1. EtherCAT 主站 .....	99
4-3-2. EtherCAT 从站 .....	100
4-3-3. 轴配置 .....	104
4-3-4. EtherCAT 控制工程 .....	107

## 4-1. EtherCAT 概要

### 4-1-1. 概述

EtherCAT 是 Ethernet for Control Automation Technology 的简称。是 Beckhoff Automation GmbH 开发的实时以太网用的主站和从站间的开放网络通信，由 ETG (EtherCAT Technology Group) 进行管理。

### 4-1-2. 系统构成

EtherCAT 的连接形态是，线型连接主站（FA 控制器）和多个从站的网络系统。

从站可连接的节点数取决于主站处理或者通信周期、传送字节数等。

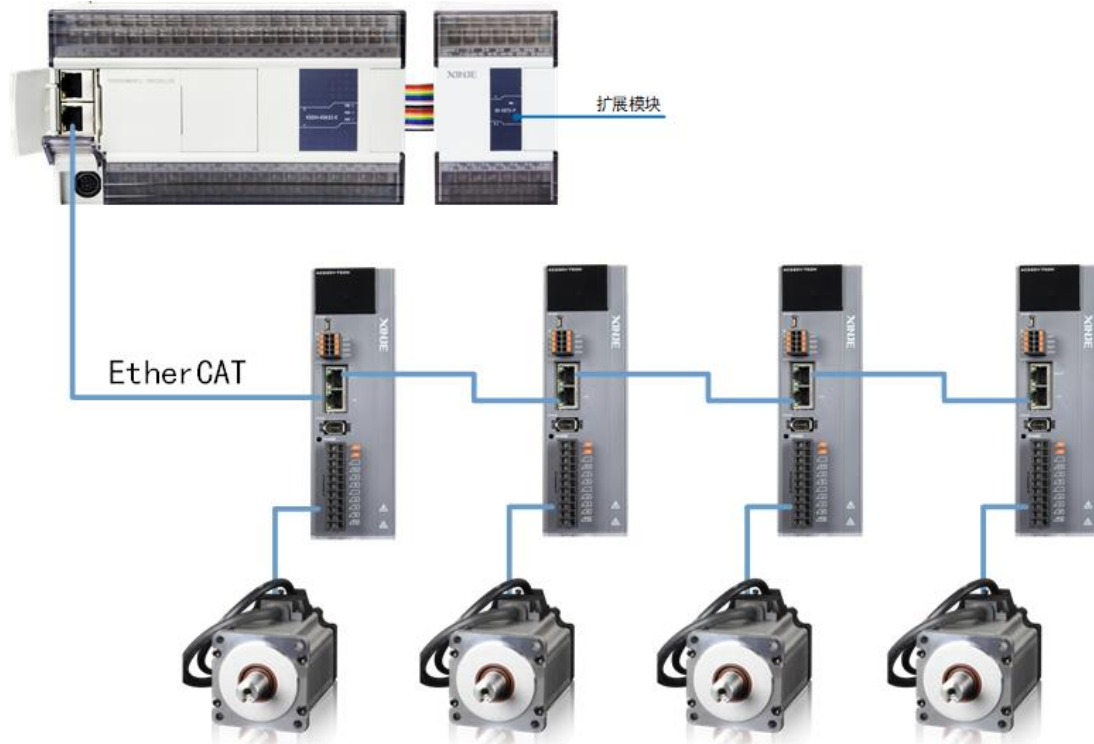
### 4-1-3. 通讯规格

项目	规格																			
物理层	100BASE-TX (IEEE802.3)																			
波特率	100[Mbps] (full duplex)																			
拓扑	Line																			
连接电缆	JC-CA 双绞线 (屏蔽双绞线)																			
电缆长	节点间最长 100m																			
通信口	2 Port (RJ45)																			
EtherCAT Indicators (LED)	[Run] RUN Indicator [L/A IN] Port0 Link/Activity Indicator (Green) [L/A OUT] Port1 Link/Activity Indicator (Green)																			
Station Alias (ID)	设定范围: 0~65535 设定地址: 2700h																			
Explicit Device ID	不支持																			
邮箱协议	COE (CANopen Over EtherCAT)																			
SyncManager	4																			
FMMU	3																			
Modes of operation 控制模式	<table border="1"> <thead> <tr> <th colspan="2">Modes of operation</th> </tr> </thead> <tbody> <tr> <td rowspan="3">位置</td> <td>csp</td> <td>Cyclic synchronous position mode (Cyclic 位置控制模式)</td> </tr> <tr> <td>PP</td> <td>Profile position mode (Profile位置控制模式)</td> </tr> <tr> <td>hm</td> <td>Homing mode (原点复位位置控制模式)</td> </tr> <tr> <td rowspan="2">速度</td> <td>csv</td> <td>Cyclic synchronous velocity mode (Cyclic 速度控制模式)</td> </tr> <tr> <td>pv</td> <td>Profile velocity mode (Profile速度控制模式)</td> </tr> <tr> <td rowspan="2">转矩</td> <td>cst</td> <td>Cyclic synchronous torque mode (Cyclic 转矩控制模式)</td> </tr> <tr> <td>tq</td> <td>Torque profile mode (Profile 转矩控制模式)</td> </tr> </tbody> </table>	Modes of operation		位置	csp	Cyclic synchronous position mode (Cyclic 位置控制模式)	PP	Profile position mode (Profile位置控制模式)	hm	Homing mode (原点复位位置控制模式)	速度	csv	Cyclic synchronous velocity mode (Cyclic 速度控制模式)	pv	Profile velocity mode (Profile速度控制模式)	转矩	cst	Cyclic synchronous torque mode (Cyclic 转矩控制模式)	tq	Torque profile mode (Profile 转矩控制模式)
Modes of operation																				
位置	csp	Cyclic synchronous position mode (Cyclic 位置控制模式)																		
	PP	Profile position mode (Profile位置控制模式)																		
	hm	Homing mode (原点复位位置控制模式)																		
速度	csv	Cyclic synchronous velocity mode (Cyclic 速度控制模式)																		
	pv	Profile velocity mode (Profile速度控制模式)																		
转矩	cst	Cyclic synchronous torque mode (Cyclic 转矩控制模式)																		
	tq	Torque profile mode (Profile 转矩控制模式)																		
Touch Probe	2 路																			
同期模式	DC (SYNCO 事件同期) SM (SM事件同步)																			
Cyclic time (DC 通信周期)	500, 1000, 2000, 4000[μs]																			
通信对象	SDO[服务数据对象], PDO[过程数据对象]																			
单站 PDO 最大分配数	TxPDO: 4 [个]      RxPDO: 4 [个]																			
单站 PDO 最大字节数	TxPDO: 24[byte]      RxPDO: 24[byte]																			
PreOP 模式下邮箱通讯间隔	1ms																			
电子邮箱	SDO 请求和 SDO 信息																			

#### 4-1-4. EtherCAT 通讯连接说明

EtherCAT 运动控制系统的接线十分简单，得益于 EtherCAT，Ethernet 的星型拓扑结构可以被简单的线型结构所替代。以信捷 DS5C 系列伺服为例，由于 EtherCAT 无需集线器和交换机，DS5C 系列伺服均自带 EtherCAT 通讯网口，因而电缆、桥架的用量大大减少，连线设计与接头校对的工作量也大大减少，便于节省安装费用。

EtherCAT 总线接线建议使用线型接法。XSDH 系列接线方式如下图所示：

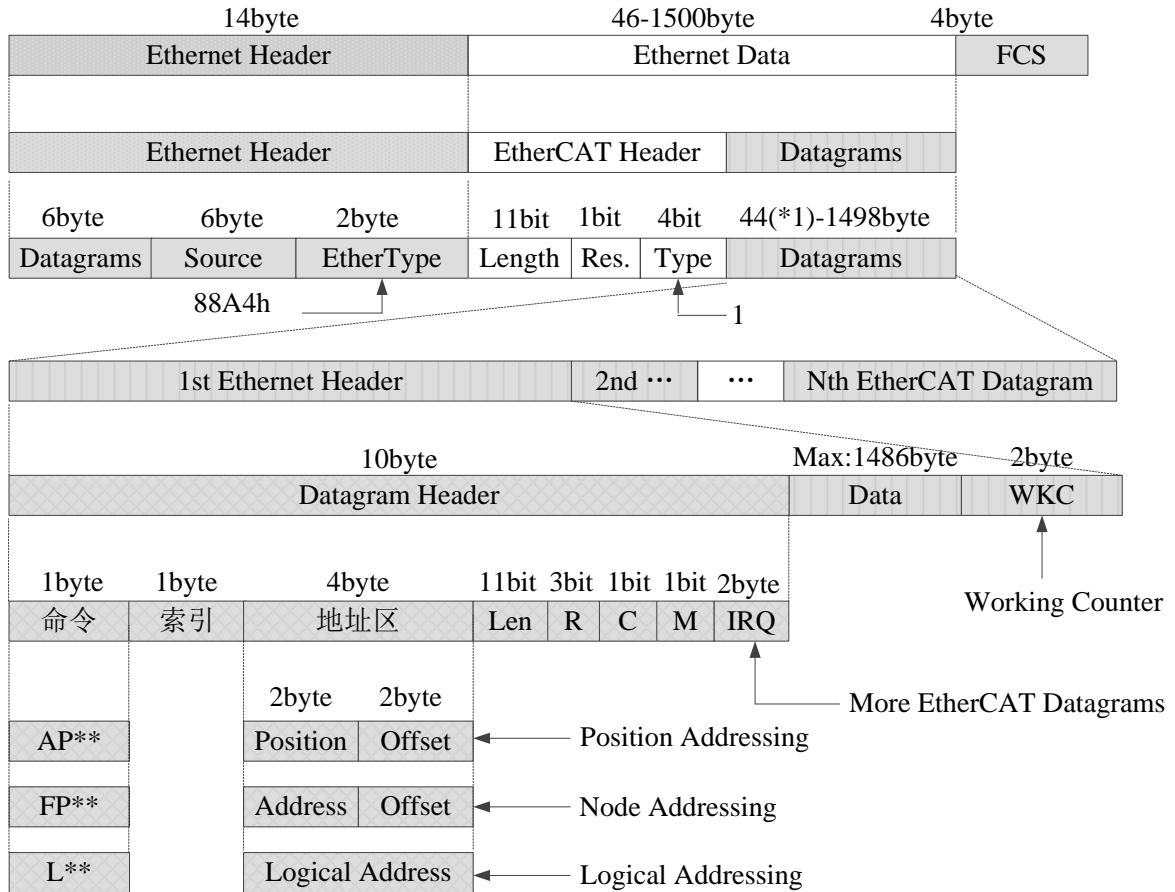


整个总线网络为线型结构，其中 XSDH 系列控制器为主站，信捷 DS5C1 系列总线控制型伺服为从站。XS3 系列 PLC 带有上下两个网口，上面的网口为 Ethernet/IP，用于连接 XS Studio 上位机；下面的网口为 EtherCAT 接口，用于连接信捷 DS5C1 系列伺服实现 EtherCAT 通讯。信捷 DS5C1 系列伺服驱动器的两个通讯网口则需遵循“下进上出”的原则。

## 4-2. EtherCAT 通讯规格

### 4-2-1. EtherCAT 帧结构

EtherCAT 是基于 Ethernet 可实时控制的工业用通信协议，只是对 IEEE 802.3Ethernet 规格进行扩充，并未对基本结构进行任何变更，所以可以转送标准的 Ethernet 帧内的数据。



\*1: Ethernet 帧比 64byte 短时，追加 1~32byte。

(Ethernet Header + Ethernet Data + FCS)

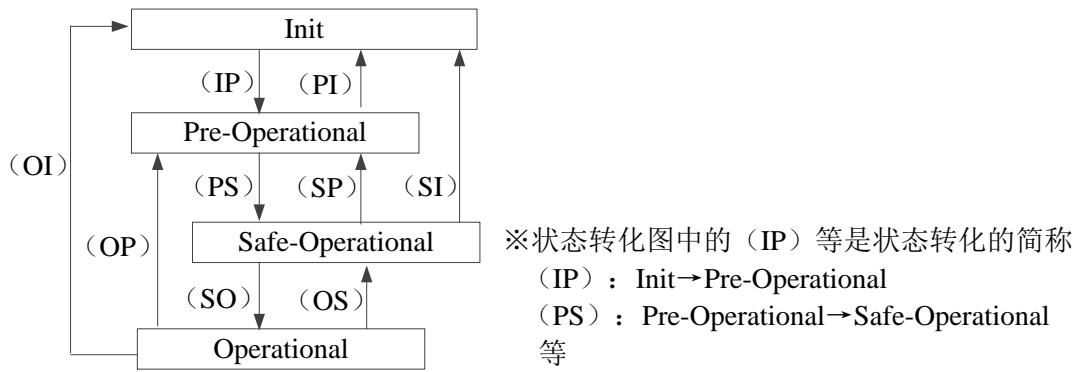
因为 Ethernet Header 的 EtherType 为「88A4h」，所以将之后的 Ethernet Data 作为 EtherCAT 帧来处理。“EtherCAT 帧”是以某种协议来进行定义和解析，只要主站和从站都遵守这个协议，就可以实现数据通信了。一般采用的协议有 CANopen Over EtherCAT (CoE)，还有 Sercos Over EtherCAT (SoE) 等。

### 4-2-2. 状态机 ESM

EtherCAT 状态机 (ESM) 负责协调主站和从站应用程序在初始化和运行时的状态关系。

状态改变请求由主站执行，主站向应用层服务提出控制请求，后者在从站中产生应用层控制事件，从站在状态改变请求成功或失败后通过本地的应用层状态写服务来响应应用层控制服务。如状态改变失败，从站保持状态并置出错误标志。

下图为 ESM 的状态转化图：



- Init: 初始化状态;
- Pre-Operational: 预运行状态;
- Safe-Operational: 安全运行状态;
- Operational: 运行状态;

从站状态	各状态下的动作	通讯动作		
		SDO (邮箱) 收发信	PDO 发信	PDO 收信
Init	通信初始化, SDO、PDO 无法收发信的状态	-	-	-
Pre-Operational (简称 PreOP)	仅 SDO 收发信的状态	Yes	-	-
Safe-Operational (简称 SafeOP)	仅 SDO 收发信, PDO 发信的状态	Yes	Yes	-
Operational (简称 OP)	SDO 收发信, PDO 收发信全部可行的状态	Yes	Yes	Yes

注：从主站到 ESC 寄存器的访问与上表无关，随时都可以。

PDO (Process Data Object) 过程数据对象用来传输周期性通讯数据。

SDO (Service Data Object) 服务数据对象用来传输非周期性通讯数据。

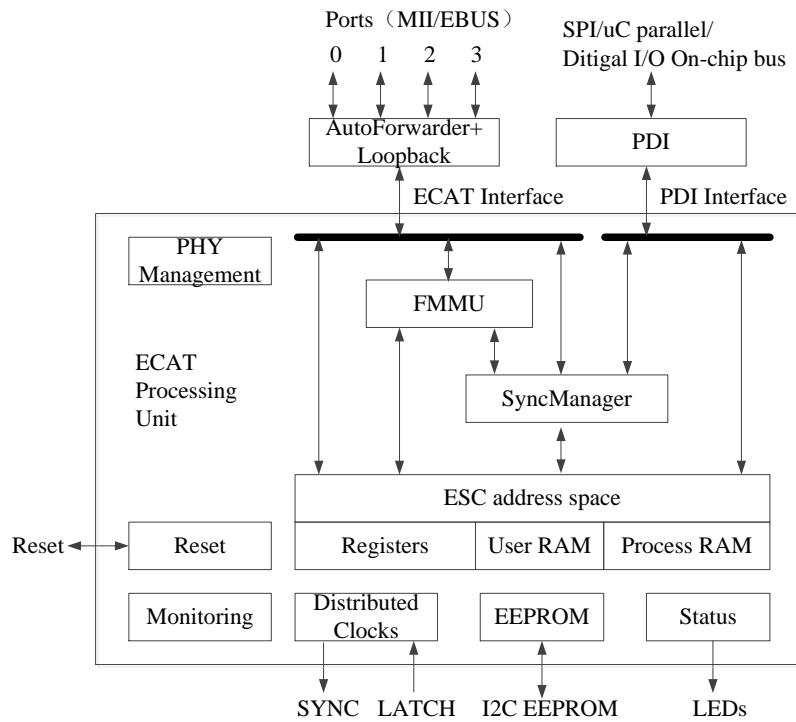
ESM 状态切换时进行指令或界面操作可能造成通信异常报错。

### 4-2-3. 从站控制器 ESC

#### 4-2-3-1. 原理概述

ESC 指的是 EtherCAT 从站控制器 (EtherCAT Salve Controller)。通信过程完全由 ESC 处理，它具有四个数据收发端口，每个端口具有一个 TX 和 RX。每个端口都可以收发以太网数据帧，ESC 中的数据流向是固定的：端口 0——>端口 3——>端口 1——>端口 2——>端口 0 的顺序依次传输。如果 ESC 检测到某个端口没有外接 PHY，则自动闭合这个端口，通过内部回环自动转发到下一个端口。





4-2-3-2. 地址空间

DS5C1 系列持有 8Kbyte 的物理地址空间。

最初的 4Kbyte (0000h~0FFFh) 是作为寄存器空间使用，另外 4Kbyte (1000h~1FFFh) 是过程数据 PDO 作为 RAM 领域使用。寄存器的详细内容请参考 IP (ET1810/ET1811/ET1812) 的数据表。

ESC 寄存器字节地址	长度 (Byte)	说明	初始值
ESC Information (从站控制器信息)			
0000h	1	Type	04h
0001h	1	Revision	02h
0002h~0003h	2	Build	0040h
0004h	1	FMMUs supported	03h
0005h	1	SyncManagers supported	04h
0006h	1	RAM Size	08h
0007h	1	Port Descriptor	0Fh
0008h~0009h	2	ESC Features supported	0184h
Station Address			
0010h~0011h	2	Configured Station Address	-
0012h~0013h	2	Configured Station Alias	-
...			
Data Link Layer			
...			
0100h~0103h	4	ESC DL Control	-
...			
0110h~0111h	2	ESC DL Status	-
Application Layer			
0120h~0121h	2	AL Control	-

ESC 寄存器字节地址	长度 (Byte)	说明	初始值
0130h~0131h	2	AL Status	-
0134h~0135h	2	AL Status Code	-
...			
<b>PDI</b>			
0140h	1	PDI Control	08h
0141h	1	ESC Configuration	0Ch
0150h	1	PDI Configuration	-
0151h	1	SYNC/LATCH PDI Configuration	66h
0152h~153h	2	Extend PDI Configuration	-
...			
<b>Watchdogs</b>			
0400h~0401h	2	Watchdog Divider	-
0410h~0411h	2	Watchdog Time PDI	-
0420h~0421h	2	Watchdog Time Process Data	-
0440h~0441h	2	Watchdog Status Process Data	-
0442h	1	Watchdog Counter Process Data	-
0443h	1	Watchdog Counter PDI	-
...			
<b>FMMU</b>			
0600h~062Fh	3x16	FMMUs[2:0]	-
+0h~3h	4	Logical Start Address	-
+4h~5h	2	Length	-
+6h	1	Logical Start bit	-
+7h	1	Logical Stop bit	-
+8h~9h	2	Physical Start Address	-
+Ah	1	Physical Start bit	-
+Bh	1	Type	-
+Ch	1	Activate	-
+Dh~Fh	3	Reserved	-
...			
<b>Distributed Clocks (DC) -SYNC Out Unit</b>			
0981h	1	Activation	-
...			
0984h	1	Activation Status	-
098Eh	1	SYNCO Status	-
...			
0990h~0993h	4	Start Time Cyclic Operation/Next SYNCO Pulse	-
...			
09A0h~09A3h	4	SYNCO Cycle Time	-
...			

### 4-2-4. SII 区域

ESC 配置区域（EEPROM 字地址 0000h~0007h）内，Configured Station Alias 在驱动器电源启动后，根据 ESC 自动读取，写入 ESC 寄存器。将 SII EEPROM 变更后的值反映到 ESC 寄存器时，需要再次启动电源。除此之外 IP 核（ET1810/ET1811/ET1812）的初始值被设定。详细内容请参照 IP 核（ET1810/ET1811/ET1812）的数据表。

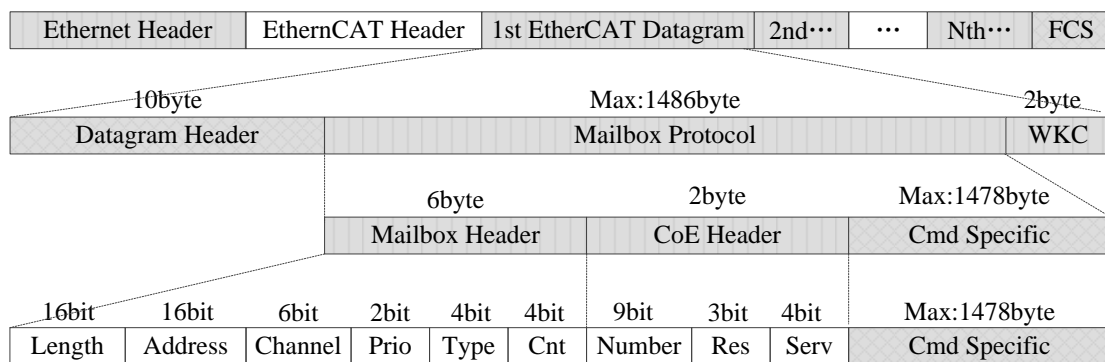
### 4-2-5. SDO

DS5C1系列支持SDO（Service Data Object）。SDO的数据交换使用Mailbox通信，所以SDO的数据刷新时间变得不稳定。

主站侧在对象字典内的记录中读写数据，可进行对象设定以及从站的各种状态的监测。到SDO的读写动作的响应需要花费时间。用PDO刷新的对象请不要用SDO来刷新，用PDO的值覆盖。

#### 4-2-5-1. Mailbox 帧结构

Mailbox/SDO 的帧结构如下所示。详细请参照 ETG 规格书（ETG1000-5 及 ETG1000-6）。



帧部	数据区域	数据类型	功能
MailBox Header	Length	WORD	Mailbox的数据长度
	Address	WORD	发信源的站地址
	Channel	Unsigned6	(Reserved)
	Priority	Unsigned2	优先度
	Type	Unsigned4	Mailbox型 00h: 错误 01h: (Reserved) 02h: EoE (未对应) 03h: CoE 04h: FoE (未对应) 05h: SoE (未对应) 06h-0Eh: (Reserved) 0Fh: VoE (未对应)
	Cnt	Unsigned3	Mailbox计数器
	Reserved	Unsigned1	(Reserved)
	CoE Header	Number	Unsigned9
Reserved		Unsigned3	Reserved
Service		Unsigned4	信息型
Cmd specific	Size Indicator	Unsigned1	Data Set Size使用许可
	Transfer Type	Unsigned1	Normal转送/Expedited转送择
	Data Set Size	Unsigned2	指定数据大小
	Complete Access	Unsigned1	对象的访问方法的选择 (未对应)

帧部	数据区域	数据类型	功能
	Command Specfier	Unsigned3	上传/下载 要求/响应等的选择
	Index	WORD	对象的Index
	Subindex	BYTE	对象的Subindex
			对象的数据或者Abort message等

4-2-5-2. Mailbox 超时

本伺服驱动器在Mailbox通信中进行下述超时设定。

Mailbox请求的超时时间：100ms

主站向从站(驱动器)发出请求,请求帧的发信数据的WKC如果被更新,从站则被认为正常接收请求。直到WKC被更新为止,反复重试,然而直到此设定时间WKC仍未被更新则主站侧超时。

Mailbox响应的超时时间：10s

主站接收来自从站(驱动器)请求的响应,如果此WKC被更新则认为是正常接收响应。直到此设定时间为止,如果无法接收WKC被更新的响应,则主站侧超时。

从站(驱动器)的响应完成所需的最大时间。

4-2-5-3. 异常报警时信息

1) Error code

Error code返回和603Fh (Error code) 相同的值。

0000h~FEFFh根据IEC61800-7-201进行定义。

FF00h~FFFFh由制造商定义的,下述内容所示:

Index	Sub-Index	Name/Description	Range	Date Type	Access	PDO	Op-mode
603Fh	00h	Error code	0-65535	U16	ro	TxPDO	All
现在伺服驱动器发生的报警(只有主编号)。 报警未发生时,显示0000h。 报警发生时,显示报警。 <b>FF**h</b> 报警(主)编号(00h~FFh) (例) FF03h ... 03h=3d E-030(过压保护)发生 FF55h ... 55h=85d E-850(TxPDO配置异常保护)、E-851(RxPDO配置异常保护), 其中任意一个发生 作为例外,E-817(SyncManager2/3设定异常)的情况下,显示A000h。							

2) Error register

Error register返回和1001h (Error register) 相同的值。

Index	Sub-Index	Name/Description	Range	Date Type	Access	PDO	Op-mode
1001h	00h	Error register	0-65535	U16	ro	TxPDO	All
显示伺服驱动器正发生的报警种类(状态)。 报警未发生时,显示0000h。 不显示警告。							
		Bit	内容				
		0	不支持				
		1					
		2					
		3					
		4	AL status code定义的报警 生*1				

		5	不支持
		6	保留
		7	AL status code未定义的报警发生*2

\*1: 所谓“AL status code定义的报警”,指EtherCAT通信关联异常E-800~7、E-810~7、E-850~7。  
 \*2: 所谓“AL status code未定义的报警”,指EtherCAT通信关联异常E-880~7和EtherCAT通信关联以外的异常。

#### 4-2-6. PDO

DS5C1系列支持PDO (Process Data Object)。

基于EtherCAT的实时数据转送通过PDO (Process Data Object) 的数据交换进行。

PDO有从主站到从站转送的RxPDO和从从站到主站转送的TxPDO。

	发信侧	受信侧
RxPDO	主站	从站
TxPDO	从站	主站

##### 4-2-6-1. PDO 映像对象

PDO映射是指,从对象字典到PDO的应用对象的映射。

DS5C系列PDO映射用的表,可以使用RxPDO用1600h~1603h、TxPDO用1A00h~1A03h的映射对象。

一个映射对象可以映射的应用对象的最大数如下所示:

RxPDO: 24 [byte], TxPDO: 24 [byte]

以下表示的是PDO映射的设定示例。

< 设定示例 >

分配应用对象6040h, 6060h, 607Ah, 60B8h到映射对象1600h (Receive PDO mapping 1: RxPDO\_1) 的情况。

Index	Sub	Object contents	
1600h	00h	04h	
	01h	6040 00 10 h	
	02h	6060 00 08 h	
	03h	607A 00 20 h	
	04h	60B8 00 10 h	
	05h	0000 00 00 h	
	...		
	18h	0000 00 00 h	
6040h	00h	Controlword	U16
6060h	00h	Mode of operation	I8
607Ah	00h	Target Position	I32
60B8h	00h	Touch probe function	U16

##### 4-2-6-2. PDO 分配对象

为了PDO数据交换,必须分配PDO映射用的表到SyncManager。PDO映射用的表和SyncManager的关系记述到PDO分配对象。DS5C系列,作为PDO分配对象,可以使用RxPDO (SyncManager2)用1C12h、TxPDO (SyncManager3)用1C13h。

一个映射对象可以映射的应用对象的最大数如下所示:

RxPDO: 4 [Table] (1600h~1603h)。

RxPDO: 4 [Table] (1A00h~1A03h)。

通常、因为映射对象1个就足够了,所以默认的不需要变更。

PDO分配对象的设定示例:

分配映射对象1600h到分配对象1C12h (Sync manager channel 2) 的情况。

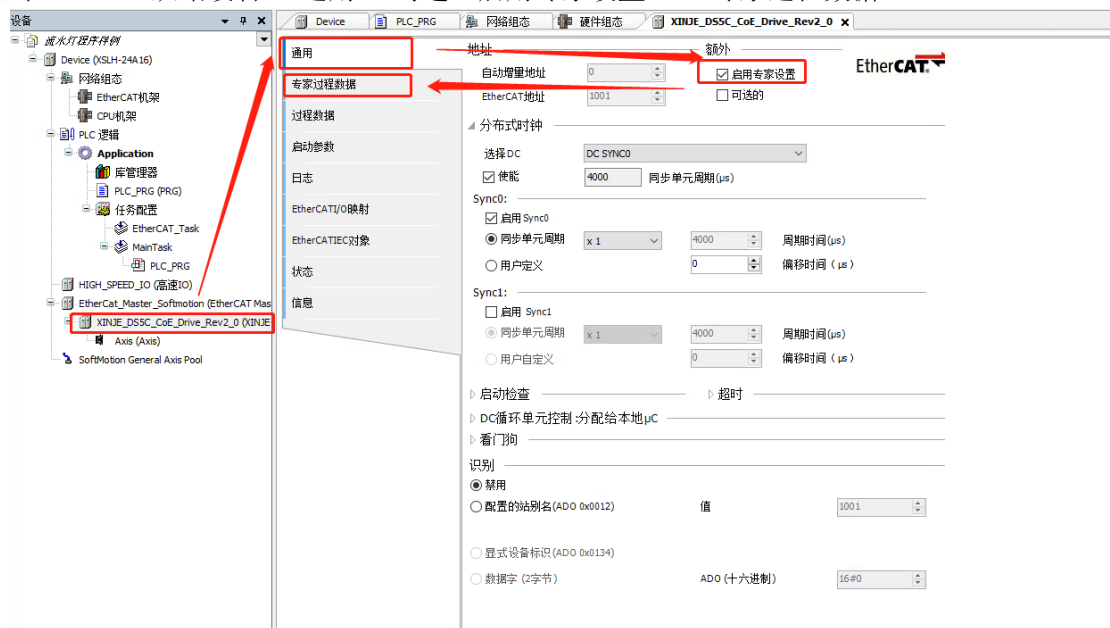
Index	Sub	Object contents
1C12h	00h	01h
	01h	1600h
	02h	0000h
	03h	0000h
	04h	0000h

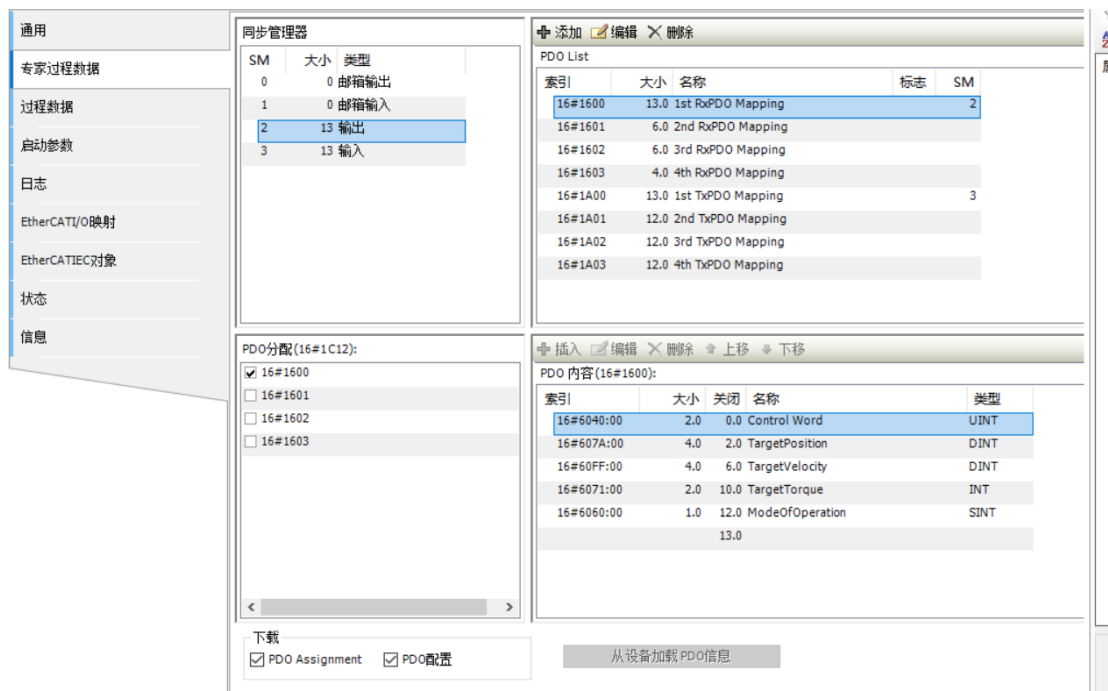
分配映射对象1600h到分配对象1C13h (Sync manager channel 3) 的情况。

Index	Sub	Object contents
1C13h	00h	01h
	01h	1A00h
	02h	0000h
	03h	0000h
	04h	0000h

### 4-2-6-3. PDO 配置

双击EtherCAT从站设备-“通用”-勾选“启用专家设置”-“专家过程数据”





### 4-2-7. 通信同步模式

DS5C系列可以选择以下的同步模式。

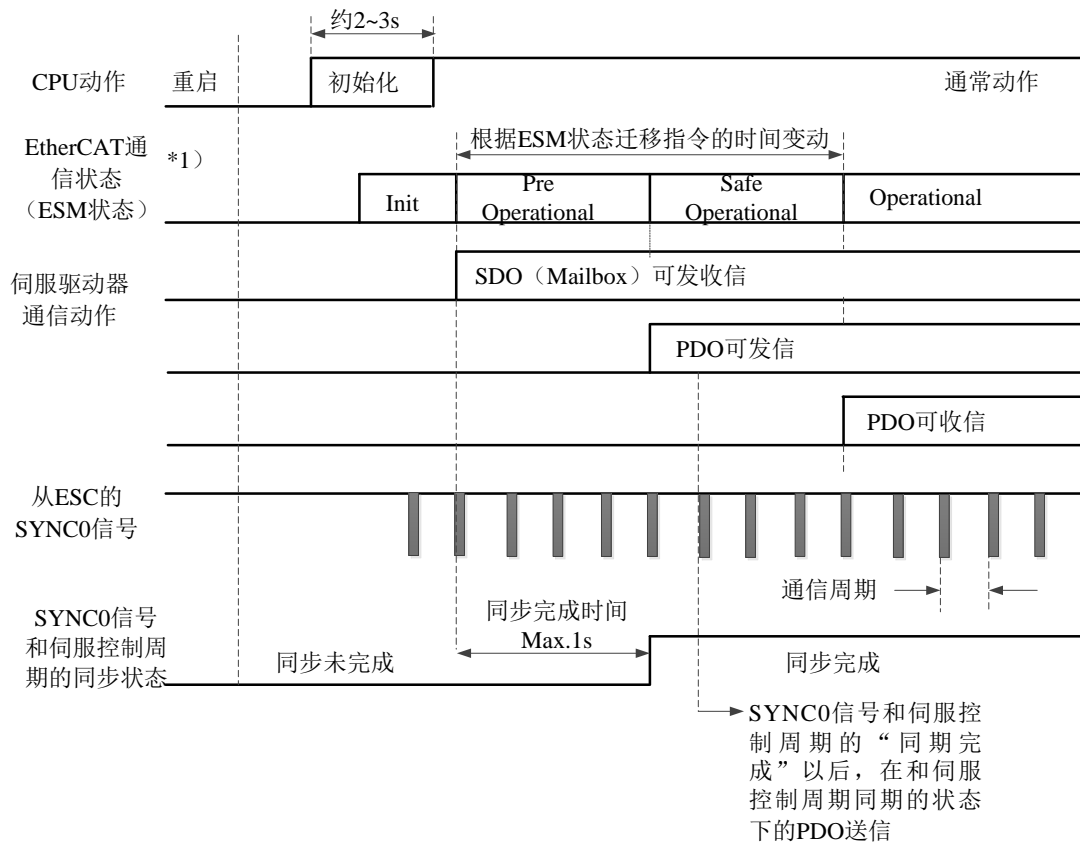
同步模式	内容	同步方法	特征
DC	SYNC0事件同步	以第1轴的时间为基准同步 其他从站的时刻信息	高精度 需要在主站侧进行补偿处理
SM2	SM2事件同步	根据RxPDO的收信时间进行同步	无传送延迟补偿，精度差 需要在控制器侧保持传送时间（专用硬件等）
FreeRun	非同步	非同步	处理简单 实时性差

#### 4-2-7-1. DC (SYNC0 事件同步)

DS5C系列有64bit的DC (Distributed Clock)。

EtherCAT通信的同步是基于此DC进行的。依据DC从站通过共有相同基准的时钟 (System Time) 实现同步。从站的本地周期开始于SYNC0事件。因为从站的处理 (伺服处理) 是开始于SYNC0事件周期，所以总是与SYNC0事件同步。

主站在通信初始化时需要进行传输延时补偿 (偏移量补偿)，还有定期的偏差补偿。下图表示从控制电源投入到SYNC0事件和从站的处理 (伺服处理) 的同步完成的过程。



#### 4-2-7-2. SM2 (SM2 事件同步)

从站的本地周期开始于SM2事件。

因为从站的处理开始于SM2事件周期，所以总是与SM2事件同步。

因为SM2事件发生在PDO的收信完成时，所以一定要确保上位（主站）侧定时送信。如果送信时间的波动（偏差）太大，同步无法完成，或者发生报警。

如果发生上述问题，请使用DC（SYNC0事件同步）。

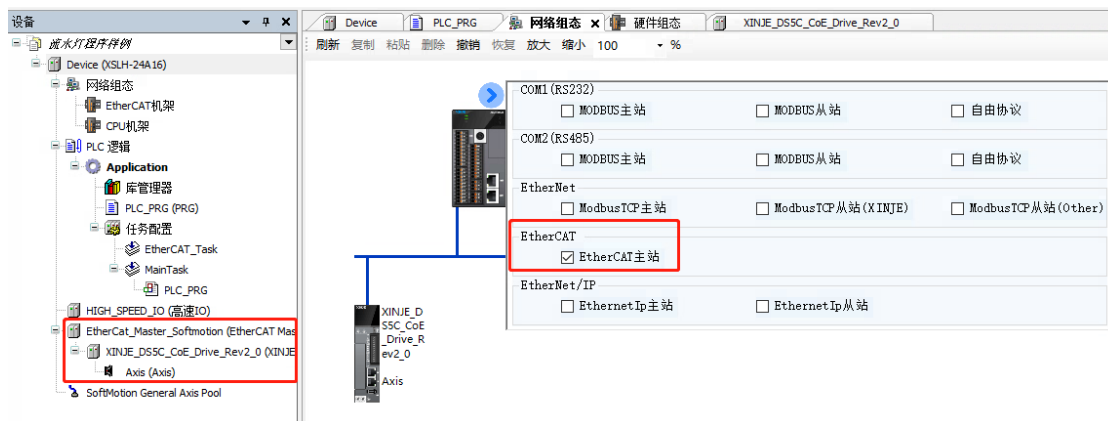


## 4-3. EtherCAT 参数配置

### 4-3-1. EtherCAT 主站

#### 4-3-1-1. 添加主站

单击网络组态界面内的使能窗口，通过勾选“EtherCAT 主站”添加主站设备，具体如图所示：



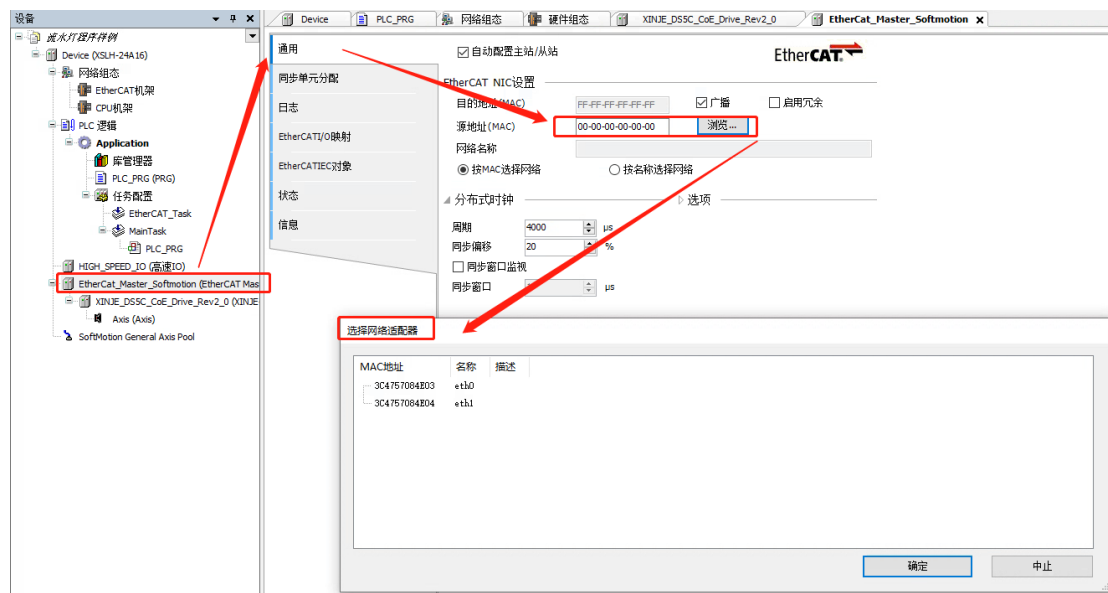
#### 4-3-1-2. 通用

##### 1) EtherCAT NIC 设置

目的地址Destination address (MAC)：为接受EtherCAT报文的目标地址，如果“广播”选项被激活，则不需要输入目标地址，系统会自动进行通过广播搜索目标地址。

激活冗余 (Redundancy)：该选项使能后，则正式开启EtherCAT冗余模式，其支持环形拓扑结构。

源地址Source address (MAC)：PLC网络接口的的MAC地址，可以选择“根据MAC选择网络”或者选择“根据名称选择网络”。用户可以选择“浏览 (Browse)” 鼠标选择想要设置的源地址。



##### 2) 分布式时钟

周期时间 (Cycle time)：如果分布式时钟功能被激活，主站发送将会根据该循环时间向从站发送相应数据报文。因此数据交换可以实现精确同步，在分布式过程中要求同步动作时(例如几个伺服轴执行同时联动任务)，此功能尤为重要的。可以在网络范围内提供信号抖动小于1微秒的主时钟。

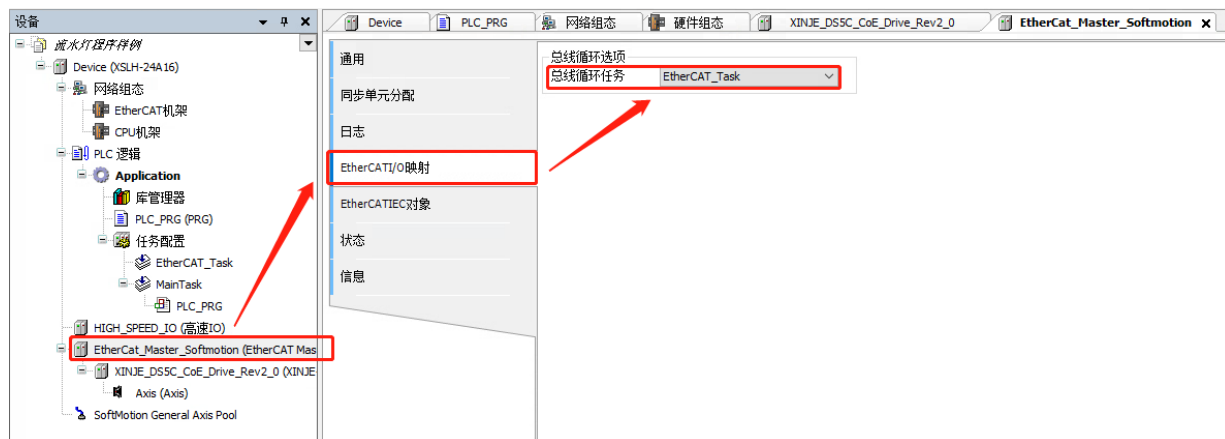
同步偏移 (Sync offset)：通常当PLC任务开始20%后，同步报文开始影响从站，这就意味着PLC的任务周期可以有80%的延迟，在此延迟内不会有数据会丢失。

同步窗口监控 (Sync window monitoring)：如果该选项打开，可以监控从站的同步状态。

同步窗口 (Sync window)：用于监控同步窗口的时间。如果所有的从站在同步窗口时间内，则变量 xSyncInWindow (IoDrvEtherCAT)将会被置为TRUE，否则为FALSE。

### 4-3-1-3. EtherCAT I/O 映射

在建立EtherCAT主站时，会自动建立EtherCAT\_Task任务，在EtherCAT I/O映射中设置总线循环任务，默认为EtherCAT\_Task。

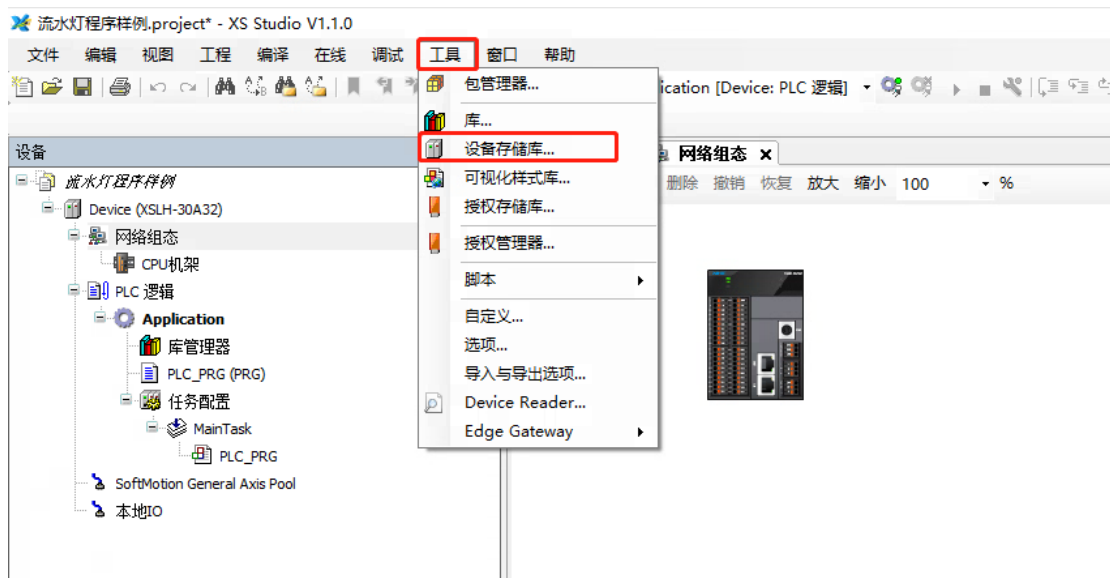


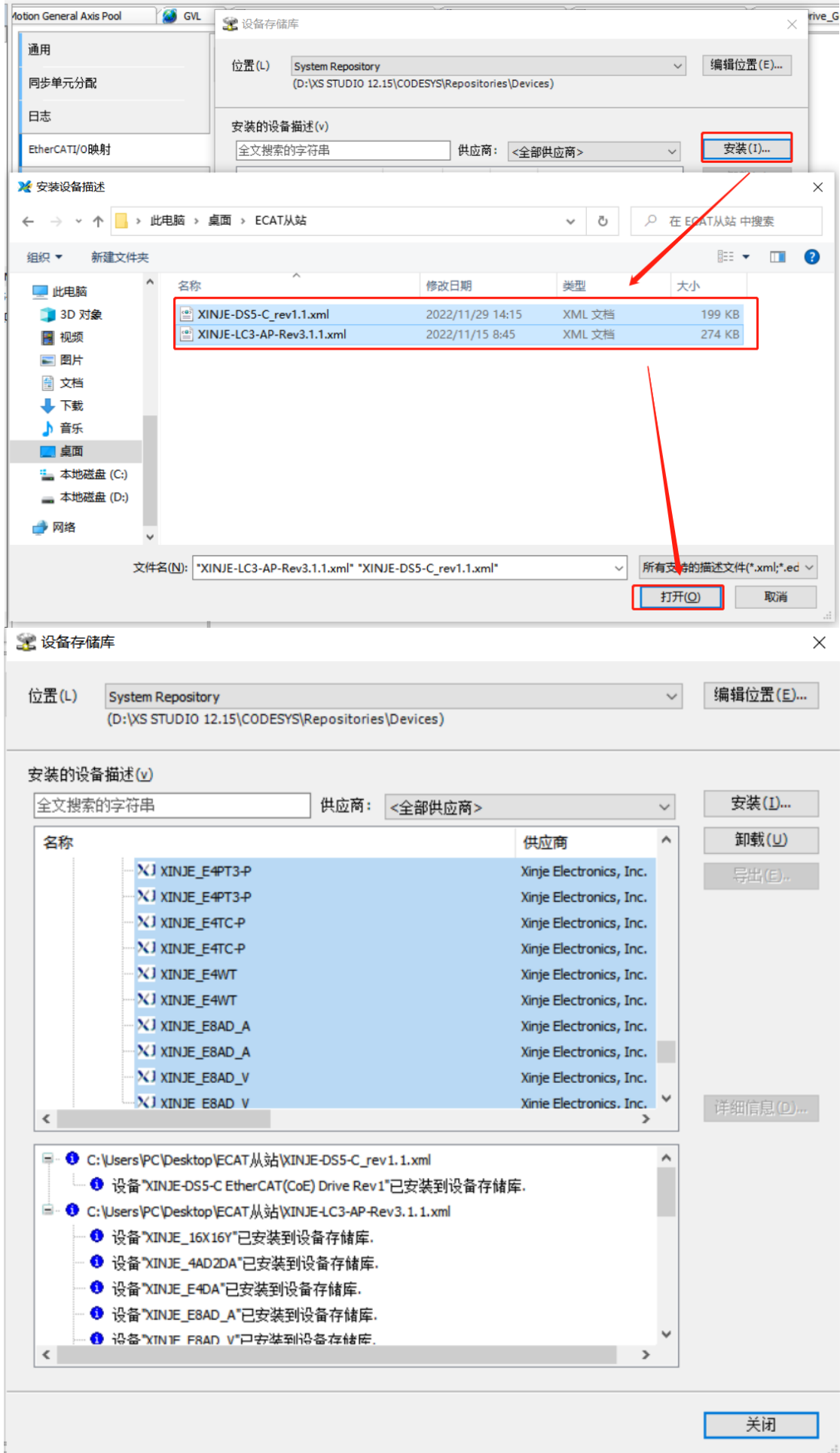
## 4-3-2. EtherCAT 从站

### 4-3-2-1. 添加从站

#### 1) 添加xml文件

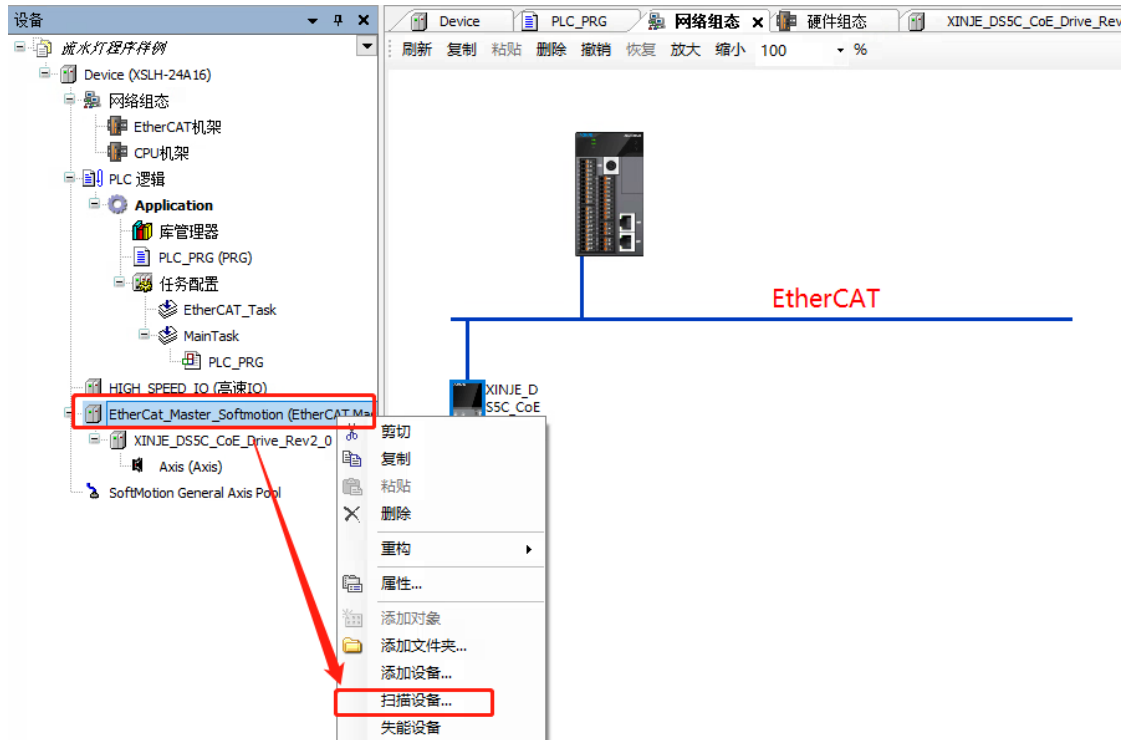
打开工具设备库，添加从站设备的XML文件。添加从站设备的XML文件。依次点击“工具”--“设备存储库...”，在弹出的对话框中点击“安装”，选择XML文件所在的路径找到XML文件，选中后点击打开。



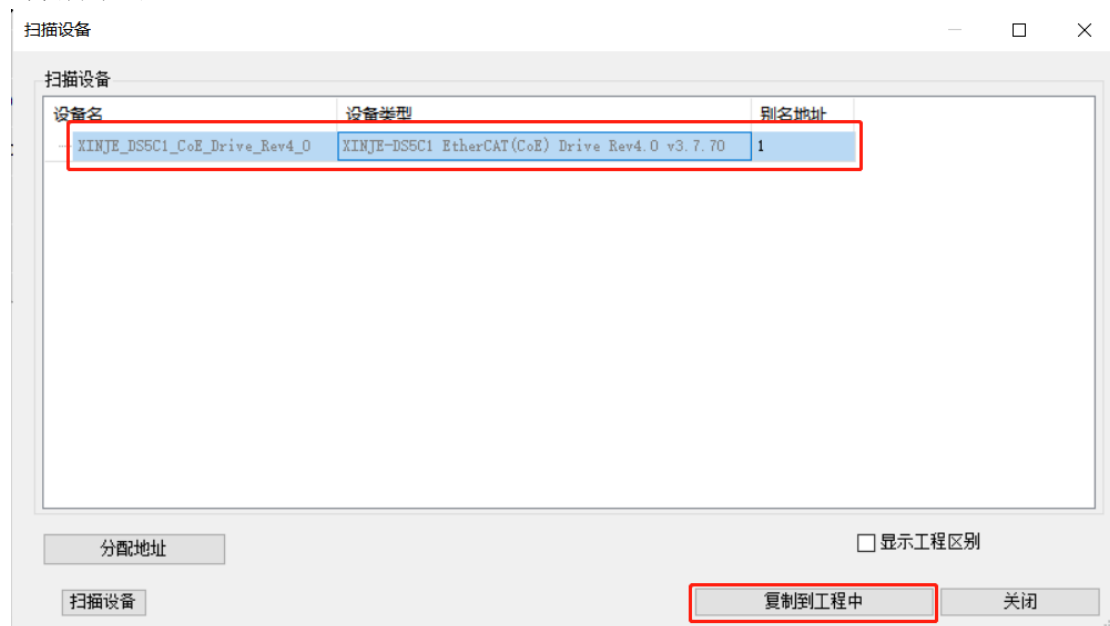


## 2) 扫描从站

在Device工程栏中，右击EtherCAT\_Master\_SoftMotion，点击“扫描设备”扫描EtherCAT从站设备，或者右击EtherCAT\_Master\_SoftMotion，点击“添加设备”，手动添加设备。

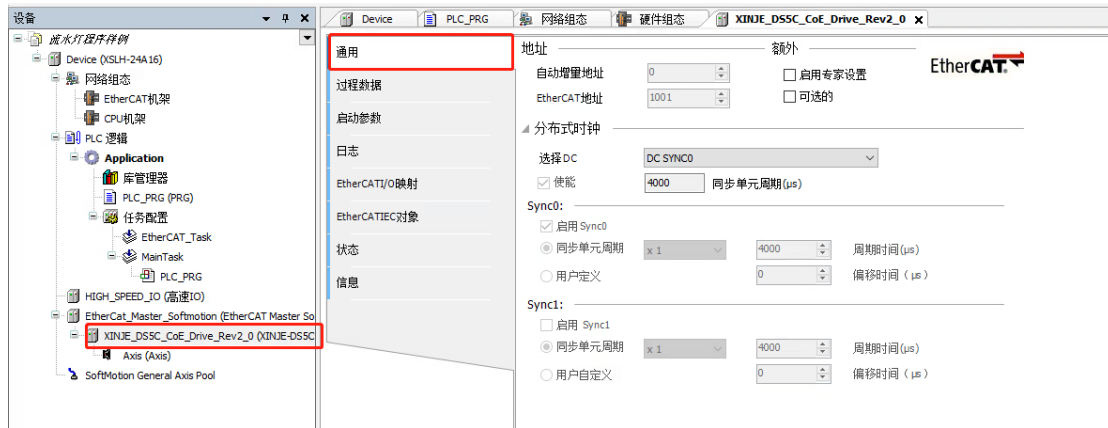


本例中连接了1台DS5C1系列的伺服，扫描结果如下图所示，单击Copy All to Project将扫描到的所有从站添加到项目中去。



**注：**使用“扫描设备”功能之前，必须确保从站的 EtherCAT 设备描述文件已经安装在调试 PC 的 XS Studio 内，否则无法使用此功能。

4-3-2-2. 通用



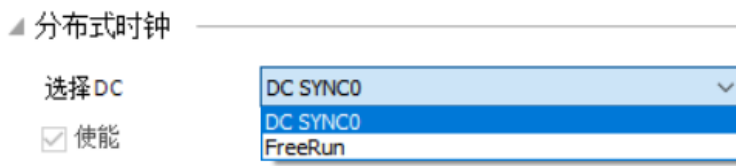
1) 地址

自动配置地址：由从站在网络中的位置确定。这个地址仅仅在启动时使用，主站需要向从站分配 EtherCAT 地址。当用于此目的的首个报文通过从站时，每一个经过的从站将自身的自动增量地址加1。

EtherCAT 地址：从站的最终地址，由主站在启动时分配。

2) 分布式时钟

选择DC：下拉菜单提供了由设备描述文件提供的所有关于分布时钟的设置，可以选择同步或者freerun 非同步模式。



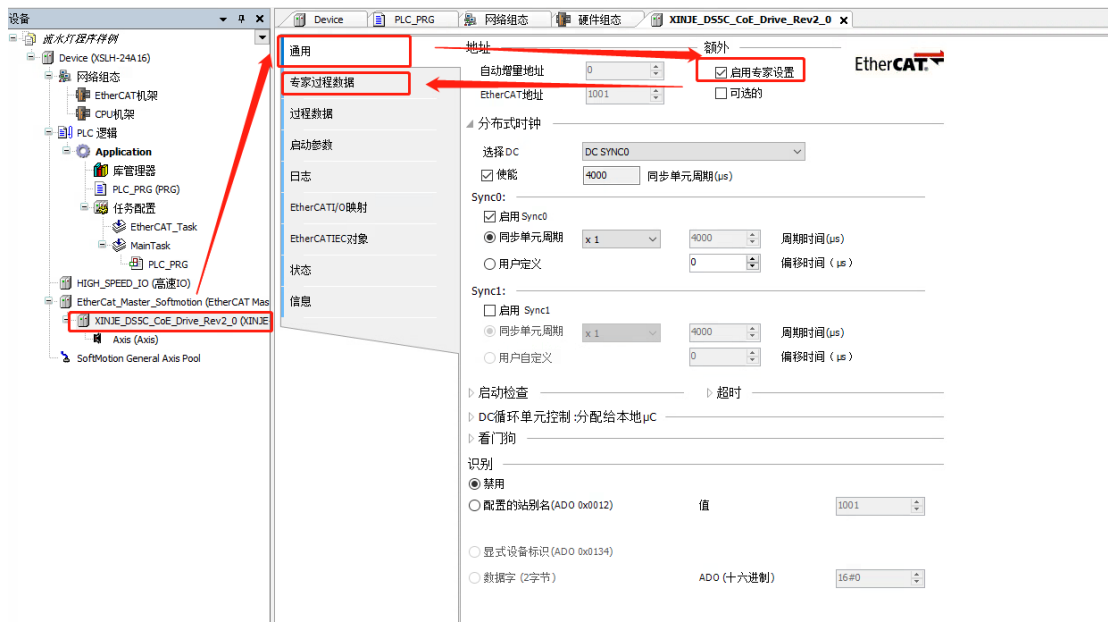
3) 同步 0/1

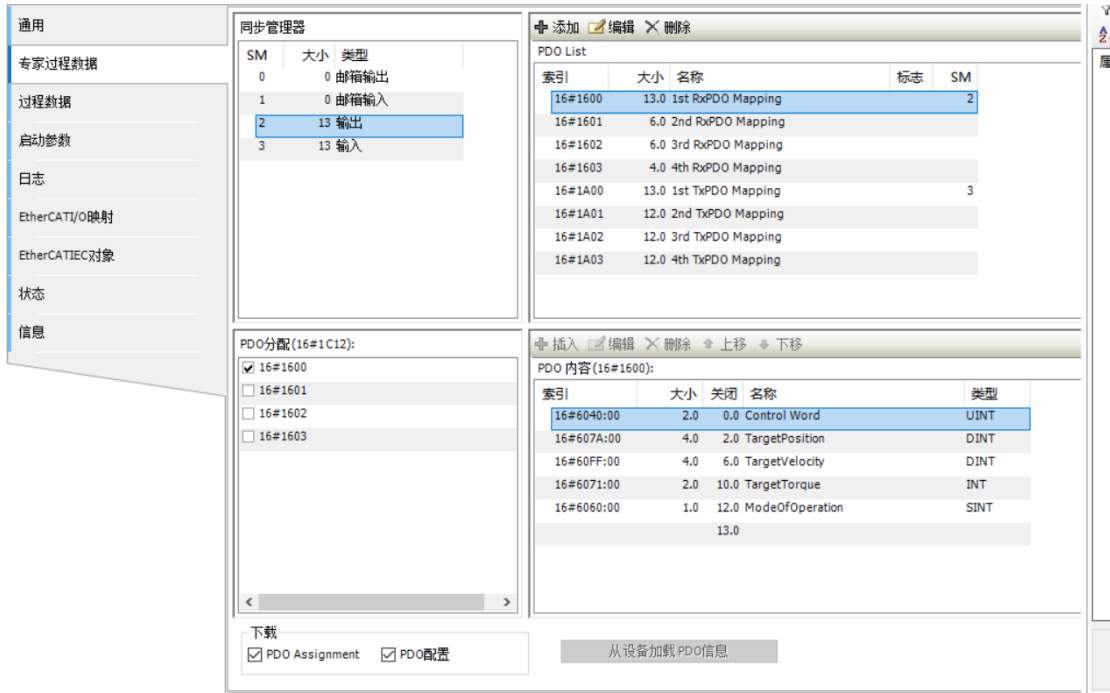
sun同步0/1使能：如果该选项被选中，使用“sync0/1”同步单元。一个同步单元描述了一套同步交换的过程数据。

同步单元循环：主站周期的时间乘以所选择的系数，将被用作从站的同步周期时间。循环时间（us）栏显示当前设置的周期时间。

4-3-2-3. 专家过程数据

在通用界面，勾选专家设置，则会出现专家过程数据的配置界面。



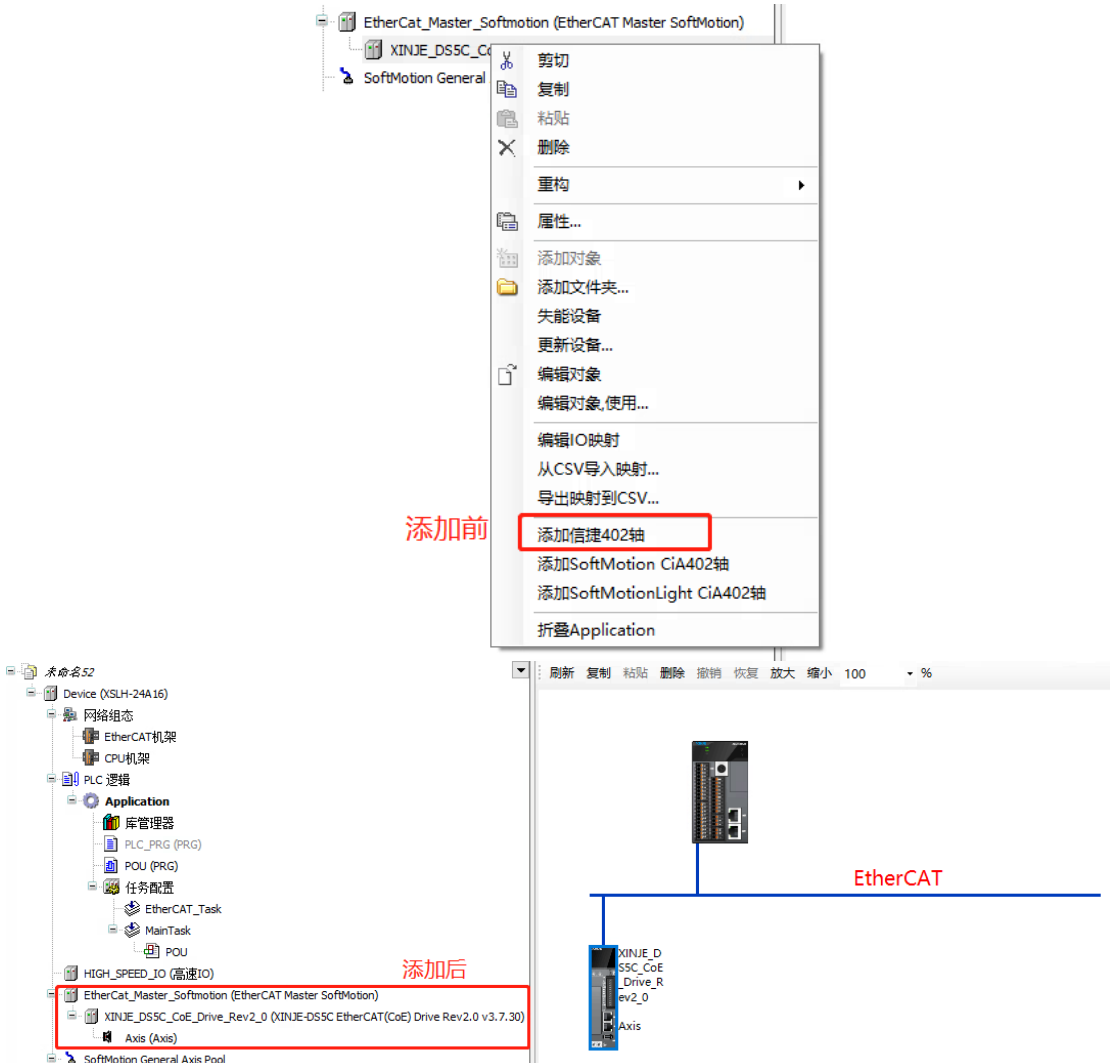


### 4-3-3. 轴配置

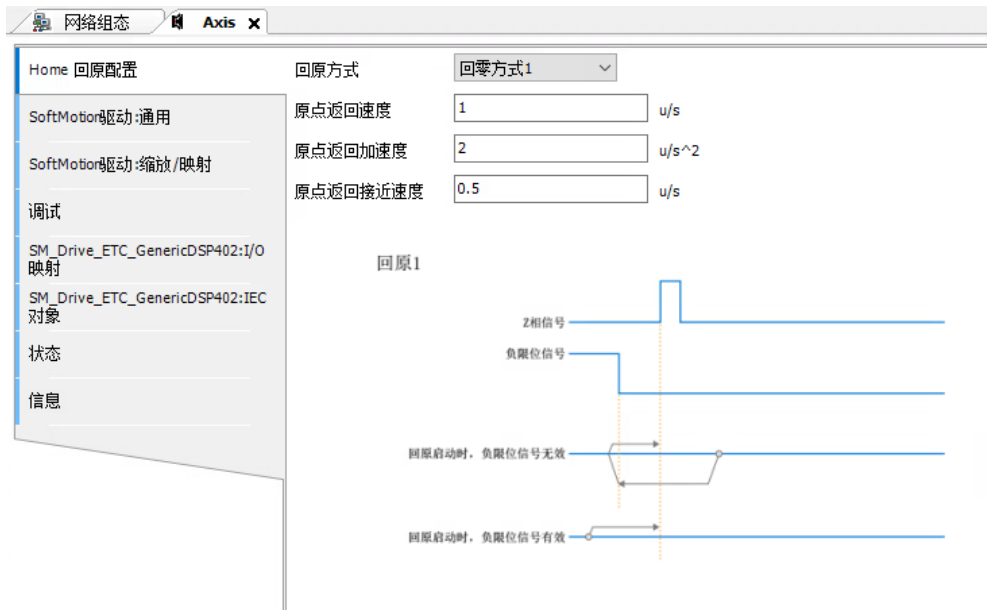
#### 4-3-3-1. 信捷 402 轴

##### 1、添加“信捷 402 轴”

添加伺服从站后，右键菜单可添加“信捷 402 轴”。如下图所示：

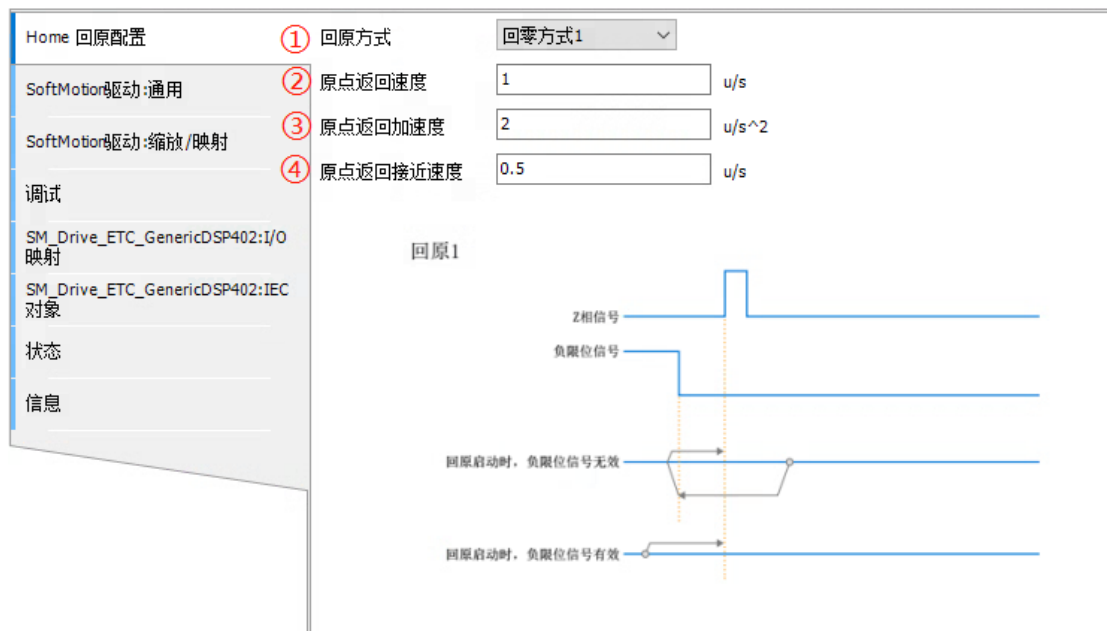


双击“轴”可打开对应轴的配置界面，如下图所示：



## 2、HOME 回原配置

HOME 回原参数设置，主要用于轴回零图形化参数配置。提供了图形化配置指导，无需另外查阅伺服手册便可通过配置界面中的下拉菜单直接选择所需的回零方式，方便用户更加直观、便捷地完成参数配置过程。



图中的主要选项及其功能如下所述：

① 回原方式（#6098h: Home method）

配置驱动器回原点的方式，总共支持 35 种选项（实际的支持回零方式由驱动器决定）。每种不同的回零方式，下边的示例图会有所不同（示例图参考 DS5C 系列的伺服回原方式），根据需要选择不同的回零方式。

② 原点返回速度（#6099h 子索引：01h）

设定到 Switch 信号检出的动作的速度。

③ 原点返回加速度（#609Ah）

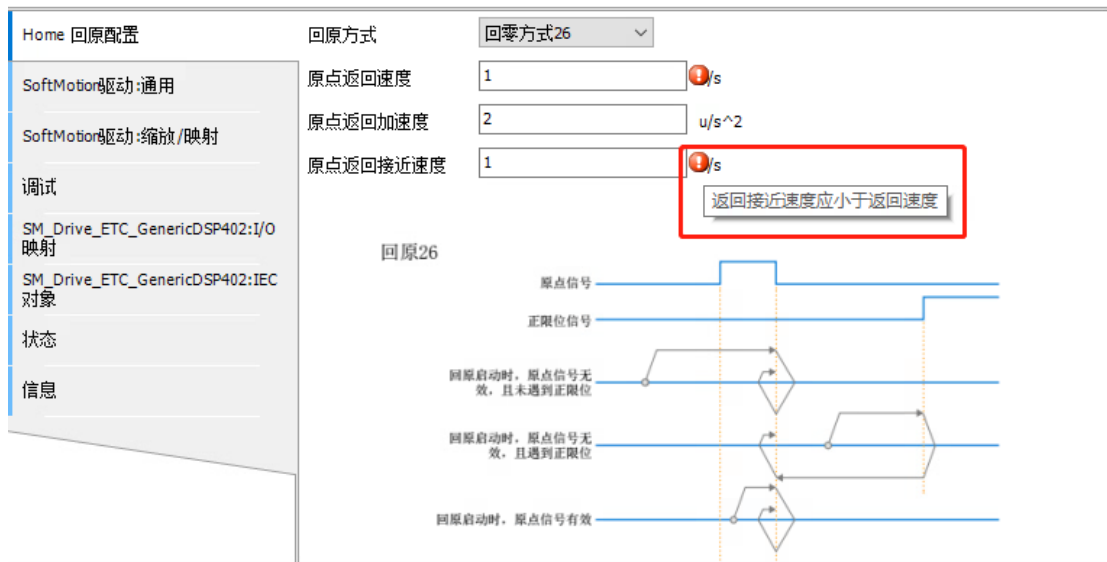
设定原点复位时的加速度以及减速度。

④ 原点返回接近速度（#6099h 子索引：02h）

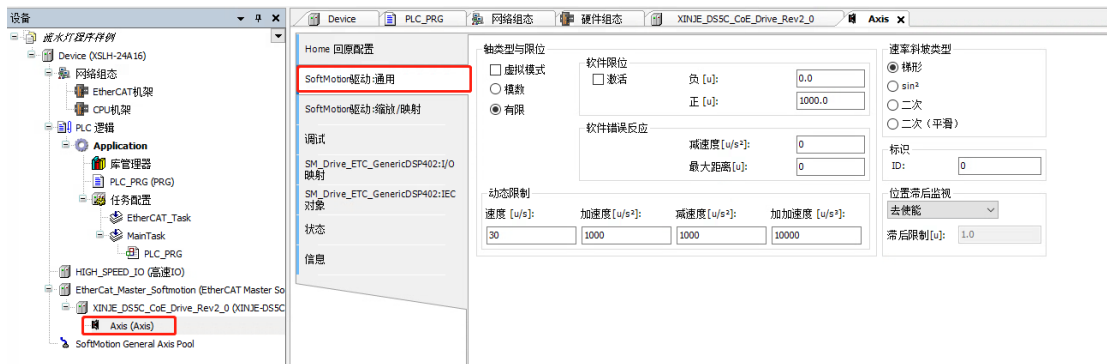
设定到原点检出的动作速度。

**注：**若原点返回速度 ≤ 原点返回接近速度则，在两个输入框的右边进行感叹号报警及信息提示。如下

图所示:



### 4-3-3-2. SoftMotion 驱动：通用



#### 1) 轴类型

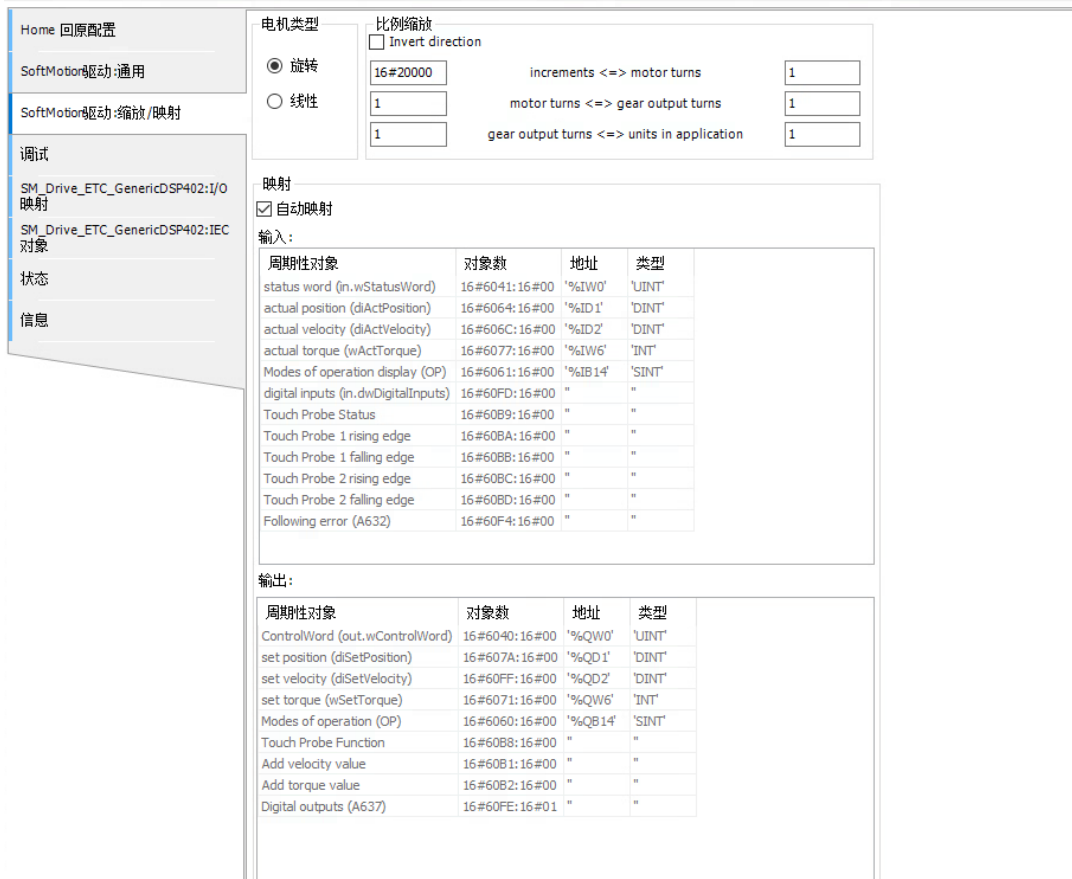
为了精确地控制运动位置，控制器必需准确计算伺服电机的位置，根据应用系统的运转特性、行程特点，选择“轴类型与限制”，以便控制器内部对读电机编码器反馈信息进行计算，得到准确位置，避免编码器脉冲数累积溢出造成的错误。

没有接入实际伺服电机的场合，则选择“虚轴模式”；对于丝杆类型的往复运行机构，其行程是有限的，我们往往需要知道其在丝杆行程范围内的绝对位置，此时选择“线性模式”比较好；若是单方向运转类型的转轴，采用线性模式容易出现位置计数溢出，导致位置计算错误，则选择“周期模式”比较好。

### 4-3-3-3. SoftMotion 驱动：缩放/映射

电机的编码器参数（如分辨率），应用系统的机械减速比可能各不相同，在编程时也需要根据实际情况进行设定。





#### 4-3-3-4. SM\_Drive\_ETC\_GenericDSP402: I/O 映射



### 4-3-4. EtherCAT 控制工程

#### 4-3-4-1. 运动工程控制

在一个工程中，程序中所用到的指令，都需要来自文件库的支持。而每一个 POU，如果不在某个任务中调用的话，便不会执行。用户可以选择直接配置到某一个任务之中去执行，或者选择在另外一个已经在任务中的 POU 对该为配置任务的 POU 进行调用。POU 中执行的程序，如果需要和外部的 IO 或总线进行交互，则需要在程序中另外配置相应的高速 IO 模块或 EtherCAT 总线及从站设备。

#### 4-3-4-2. 多个 POU 用法

在编写应用程序时，不同执行周期的程序功能，应放到不同的 POU 中进行编写，并配置到不同优先级和周期时间的任务之中，以便于后续程序的查看和优化。

- 合理分配 CPU 资源，按每个功能需要的周期时间去进行周期的分配；

- 程序结构清晰，各功能区分清楚，相较于将所有程序堆叠在一起，多个 POU 用法可通过不同的命名区分功能，体现在工程栏上，程序的逻辑结构一目了然；
- 调试方便，在调试时，可以很方便地对某部分需要屏蔽的功能进行屏蔽；
- 可以实现在不同工程之间，对 POU 进行直接的引用，直接将 POU 从工程 1 复制到工程 2；
- 将程序规划清楚后，可以分给多个人进行编程开发，提高编程的效率；
- 可以在不同的 POU 使用不同的编程语言，只要保证接口清晰，对 POU 内的编程语言没有统一要求。

#### 4-3-4-3. 运动功能的调用方式

在一个工程中，为了更合理地分配 CPU 的资源，编程时会将不同周期的程序放到不同的 POU 和任务之中。

运动功能需要最高优先级的任务，而逻辑功能一般不需要那么高优先级的任务配置，因此，在实际工程中，通常这两块是放在两个不同的 POU 与任务之中的。那么，要怎么实现即使运动功能与逻辑功能分开，依然能够在逻辑功能之中去控制一个运动功能的执行？一般都是在运动功能中定义一些输入变量和输出变量，给其它功能进行调用，如在逻辑 POU 中，需要调用运动功能，就往运动 POU 的输入变量写入控制数据，运动 POU 将运动状态放到输出变量中，给到逻辑 POU，用以判断运动状态，判断程序逻辑的执行。

## 5. 编程基础

XS Studio 也有与其他高级编程语言相似的常量、变量概念。常量具有和变量一样的概念和性质，不同的是，常量不可修改，是固定的，而变量可以多次修改。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，或不指定地址，由系统自行分配，用户可以不关注这些变量的存储位置。

---

5. 编程基础.....	109
5-1. 直接地址.....	110
5-1-1. 定义语法.....	110
5-1-2. PLC 直接地址存储区域.....	110
5-2. 变量.....	111
5-2-1. 变量定义.....	111
5-2-2. 变量类型.....	112
5-3. 掉电保持变量.....	115
5-4. 配方操作.....	118
5-4-1. 应用举例.....	118

## 5-1. 直接地址

### 5-1-1. 定义语法

在 XS Studio 应用中，当需要和可编程逻辑控制器的 I/O 模块进行变量映射或和外部设备进行网络通讯时，需要采用此声明方法。

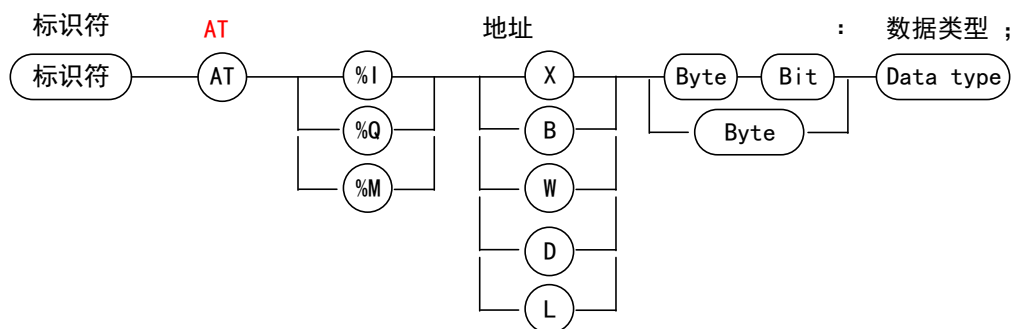
使用关键字 AT 把变量直接联结到确定地址，直接变量须符合以下规则：

AT<地址>：

<标识符>AT<地址>：<数据类型>{：=<初始化值>}；

{ } 中为可选部分。

使用“%”开始，随后是位置前缀符号和大小前缀符号，如果有分级，则用整数表示分级，并用小数点符号“.”表示，如%IX0.0，%QW0。直接变量声明的具体格式如下图所示：



位置前缀的定义：

- I：表示输入单元；
- Q：表示输出单元；
- M：表示存储区单元。

大小前缀的定义如下表所示：

前缀符号	定义	约定数据类型
X	位 (bit)	BOOL
B	字节 (byte)	BYTE
W	字 (WORD)	WORD
D	双字 (DWORD)	DWORD
L	长字 (LWORD)	LWORD
*	未特定位置的内部变量，系统自动分配	

示例：

%IX3.2 输入区域偏移 3 个字节的第三位 (bit2)

%QW10 输出区域偏移 10 个字

%MB20 内存区域偏移 20 个字节

Var1 AT%ID48:DWORD; //Var1 变量是双字类型，映射到输入区域偏移 48 双字位置

### 5-1-2. PLC 直接地址存储区域

区域	用途	大小	地址范围
I 区 (%I) 128KB	用户使用区域	64KWords	%IW0-%IW65535
Q 区 (%Q) 128KB	用户使用区域	64KWords	%QW0-%QW65535
M 区 (%M) 256KB	用户使用区域	128KWords	%MW0-%MW131070

## 5-2. 变量

### 5-2-1. 变量定义

#### ■ 文本声明

```

1  {attribute 'qualified only'}
2  VAR_GLOBAL PERSISTENT RETAIN
3      M AT %MB0: BOOL:=TRUE;
4      D AT %MW40000:ARRAY [0..5] OF WORD;
5      HD AT %MW40006:ARRAY [0..1] OF LREAL;
6  END_VAR
    
```

#### ■ 表格声明

类别	名称	地址	数据类型	初值	注释	特性
VAR_GLOBAL RETAIN PERSISTENT	M	%MB0	BOOL	TRUE		
VAR_GLOBAL RETAIN PERSISTENT	D	%MW40000	ARRAY [0..5] OF WORD			
VAR_GLOBAL RETAIN PERSISTENT	HD	%MW40006	ARRAY [0..1] OF LREAL			

表格声明	描述
类别	变量的类别（如本地变量(VAR)，输入变量(VAR_INPUT)，输出变量(VAR_OUTPUT)等）
名称	变量的名称
地址	变量映射的地址
数据类型	变量的数据类型（如 BOOL, INT 等）
初值	变量的初始值
注释	变量的注释
特性	变量的特性

#### ■ 注释

可通过表格也可通过文本进行注释编辑

类别	名称	地址	数据类型	初值	注释	特性
VAR_GLOBAL RETAIN PERSISTENT	M	%MB0	BOOL	TRUE	开关	
VAR_GLOBAL RETAIN PERSISTENT	D	%MW40000	WORD		长度	
VAR_GLOBAL RETAIN PERSISTENT	HD	%MW40006	LREAL		距离	

表格声明

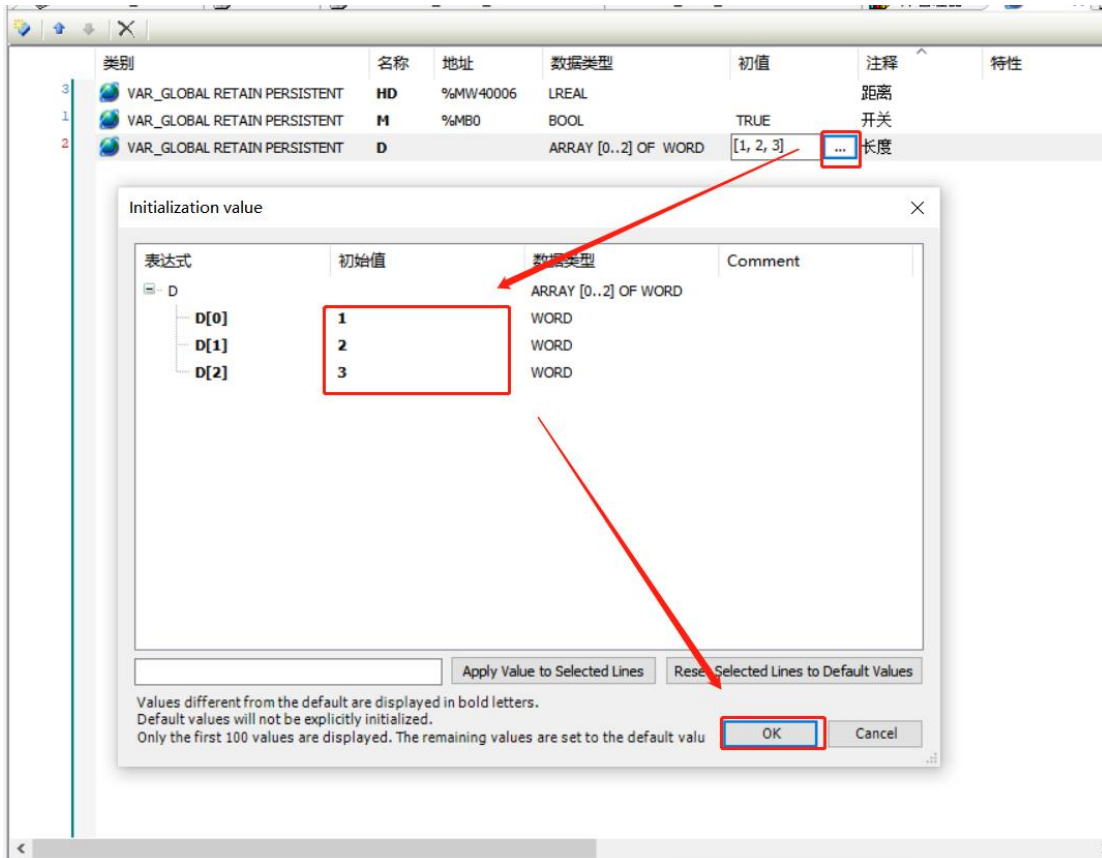
```

EtherCAT_Task  Device  EtherCAT_Master_SoftMotion  SM_Drive_GenericDSP402  库管理器  GVL x
1  {attribute 'qualified only'}
2  VAR_GLOBAL PERSISTENT RETAIN
3      // 开关
4      M AT %MB0: BOOL:=TRUE;
5      // 长度
6      D AT %MW40000: WORD;
7      // 距离
8      HD AT %MW40006:LREAL;
9  END_VAR
    
```

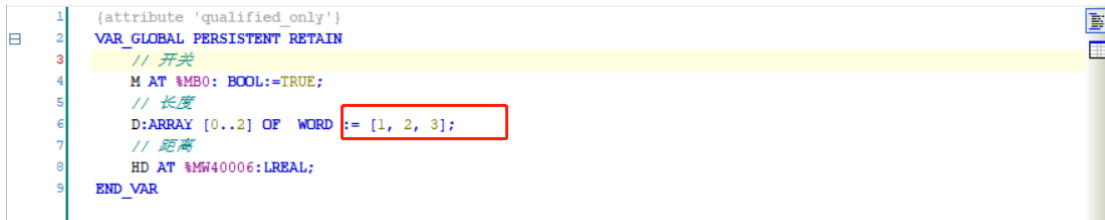
文本声明

■ 初始值设置

1、在变量声明的初始值一列中双击打开设置窗口



2、也可以用文本形式直接声明



5-2-2. 变量类型

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)。

变量类型声明语法: <TYPE> | Attribute

variable1;

variable2;

...

END\_VAR

TYPE: 类型关键字, 包括 VAR (本地变量), VAR\_INPUT (输入变量), VAR\_OUTPUT (输出变量), VAR\_IN\_OUT (输入输出变量), VAR\_GLOBAL (全局变量), VAR\_TEMP (临时变量), VAR\_STAT (静态变量), VAR\_CONFIG (配置变量)。

Attribute: 属性关键字, 包括 RETAIN, PERSISTENT, CONSTANT, 用于明确变量的范围。

■ 变量类型

变量类型关键字	变量属性	外部读写	内部读写
VAR	局部变量	-	R/W
VAR_INPUT	输入变量, 由外部提供	R/W	R

变量类型关键字	变量属性	外部读写	内部读写
VAR_OUTPUT	输出变量, 有内部变量提供给外部	W	R/W
VAR_IN_OUT	输入-输出变量	R/W	R/W
VAR_GLOBAL	全局变量, 能在所有配置、资源内使用	R/W	R/W
VAR_TEMP	临时变量, 程序和功能块内部存储使用的变量	-	R
VAR_STAT	静态变量		
VAR_EXTERNAL	外部变量, 能在程序内部修改, 但需由全局变量提供	R/W	R/W

VAR、VAR\_INPUT、VAR\_OUTPUT 和 VAR\_IN\_OUT 是在程序组织单元 (POU) 中被应用的最多的几种变量类型。

VAR\_GLOBAL 全局变量在实际的工程项目中也需要被大量的使用。

#### ■ 变量属性

变量附加属性关键字	变量附加属性
RETAIN	保持型变量, 用于掉电保持。
PERSISTENT	保持型变量
VAR RETAIN PERSISTENT VAR PERSISTENT RETAIN	两者功能一样, 皆为保持性变量, 用于掉电保持
CONSTANT	常量

#### ● RETAIN

以关键字 RETAIN 声明类型变量。RETAIN 型变量在控制器正常关闭、打开 (或收到在线命令“热复位”), 甚至意外关闭之后这类变量仍然能保持原来的值。随着程序重新开始运行, 存储的值能继续发挥作用。

RETAIN 类型变量声明格式如下:

```
VAR RETAIN
<标识符>:<数据类型>;
END_VAR
```

但 RETAIN 变量在“初始值位”、“冷复位”和程序下载之后将会重新初始化

内存存储位置: RETAIN 型变量仅仅被存储在一个单独的内存区中。

在实际的工程应用中, 如生产线上的计件器便是一个典型的例子: 电源被切断之后, 它仍然可以在再次启动时继续计数。而其它所有变量此时都将被重新初始化, 变为指定初始值或标准初始化的值。

#### ● PERSISTENT

目前只有少数 PLC 还保留独立的内存区域用于存放 PERSISTENT 类型数据, 在 XS Studio 中, 取消了其原掉电保持的功能, 取而代之的是通过 VAR RETAIN PERSISTENT 或 VAR PERSISTENT RETAIN 来实现, 两者从功能上完全一样。

PERSISTENT 类型变量声明格式如下:

```
VAR GLOBAL PERSISTENT RETAIN
<标识符>:<数据类型>;
END_VAR
```

内存存储位置: 与 RETAIN 变量一样, RETAIN PERSISTENT 和 PERSISTENT RETAIN 变量也存储在一个独立的内存区中。

#### ● CONSTANT

常量, 在程序运行过程中, 只能对其读取数据而不能进行修改的量称之为常量, 关键字为 CONSTANT。可以将常量声明为局部常量, 也可以为全局常量。

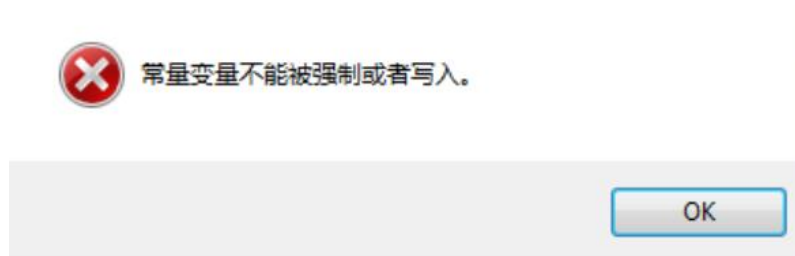
CONSTANT 常量声明格式如下。

```
VAR CONSTANT
<标识符>:<数据类型> := <初始化值>;
END_VAR
```

在实际应用中，通常可以将一些重要参数或系数设为常量，这样可以有效的避免其他变量对其修改最终影响系统整体稳定性及安全性。举例如下。

```
VAR CONSTANT  
pi:REAL:= 3.1415926;  
END_VAR
```

程序一旦开始运行，通过 CONSTANT 声明的变量在程序运行过程中，是不允许被修改的，如强制修改系统会出现如图 4.x 所示的系统错误。





### 5-3. 掉电保持变量

#### ■ 特性

掉电保持变量在 PLC 掉电、程序下载后继续保留原来的值，常用来定义工程中重要的参数，防止 PLC 突发掉电或者程序下载而导致的重要参数丢失。

掉电保持主要通过属性关键字 PERSISTENT RETAIN 来声明。

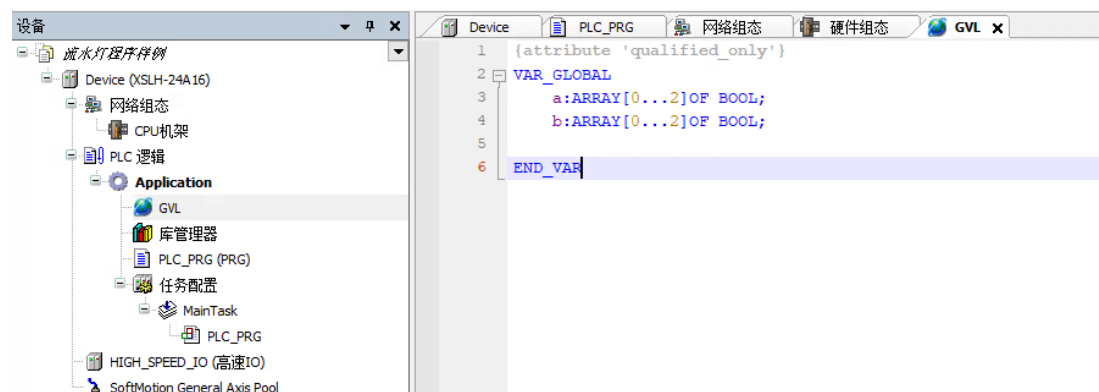
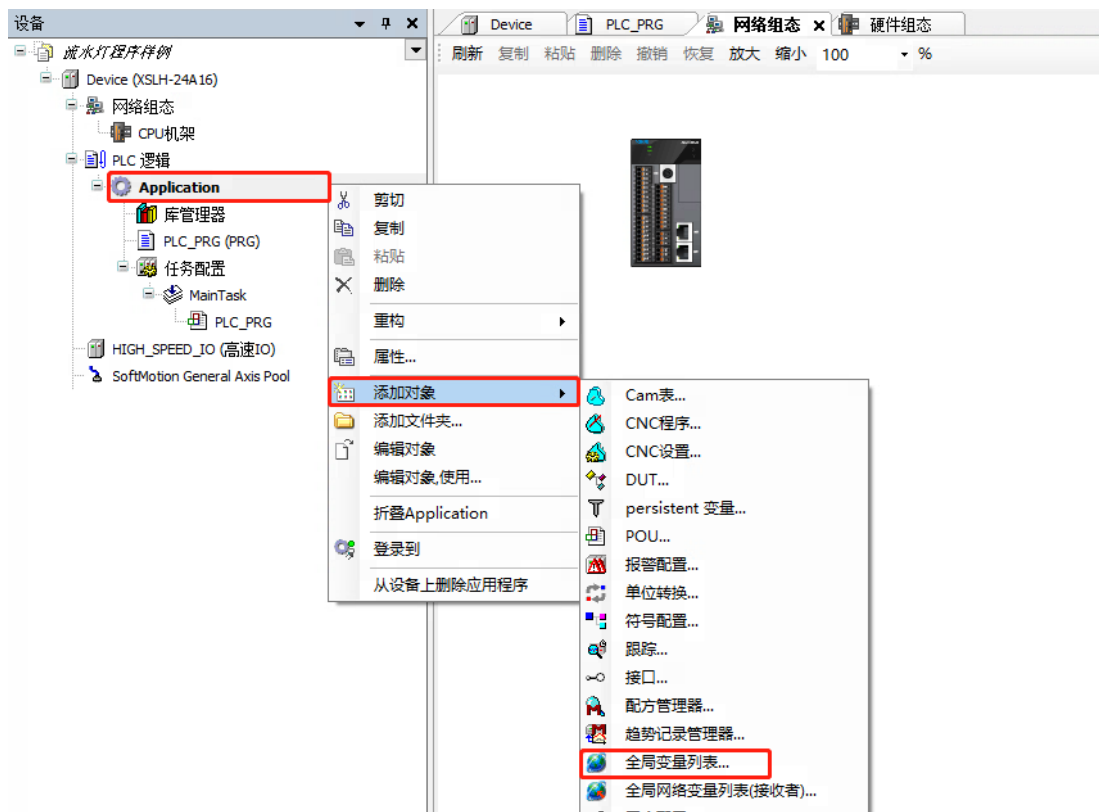
保持变量在线命令行为表如下：

在线命令	VAR	VAR RETAIN	VAR PERSISTENT RETAIN
热复位	初始值	保留值	保留值
冷复位	初始值	初始值	保留值
原始复位	初始值	初始值	初始值
下载	初始值	初始值	保留值
在线修改	保留值	保留值	保留值
重新下载	初始值	保留值	保留值

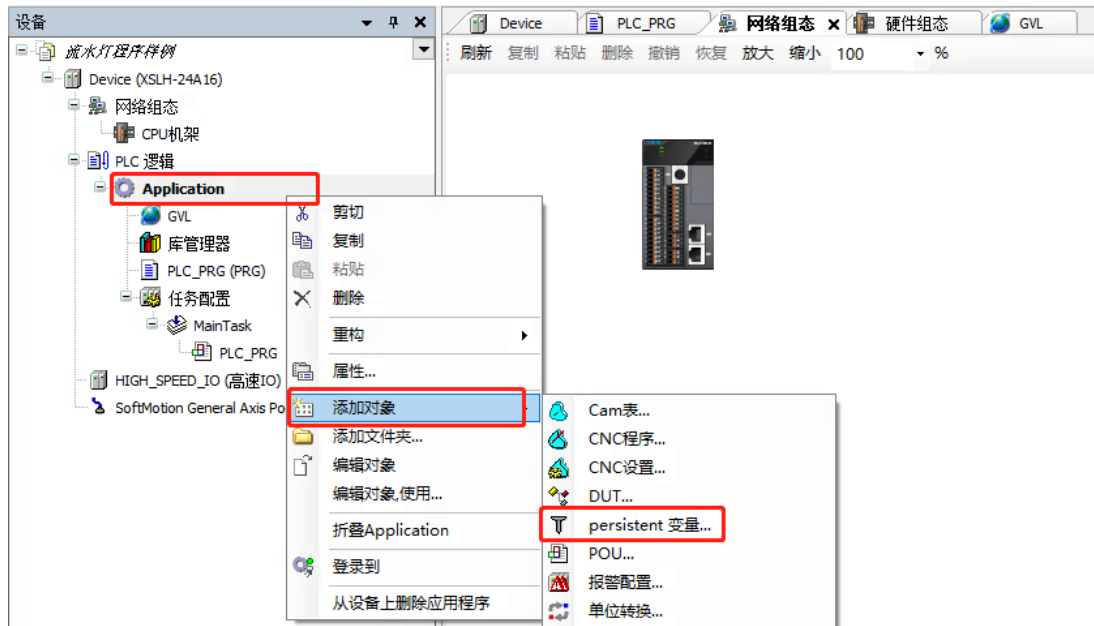
#### ■ 配置

PersistentVars 变量掉电保持

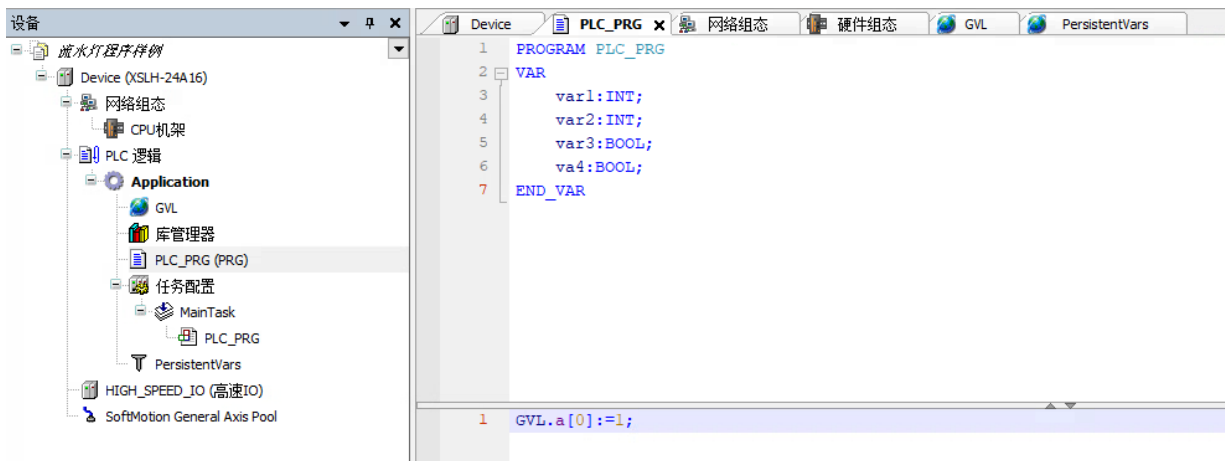
1、建立全局变量，类型为 PERSISTENT RETAIN;



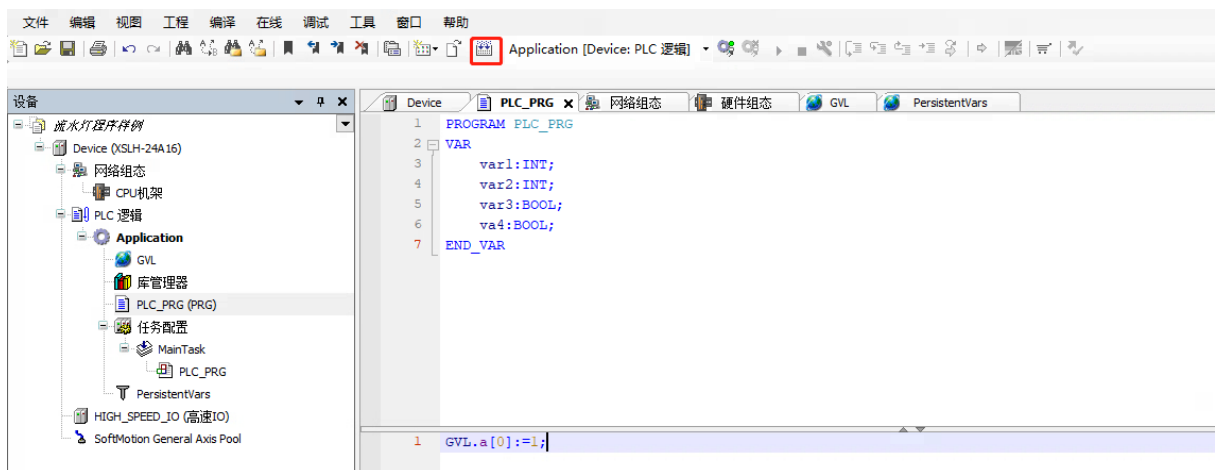
### 2、建立 persistent 变量



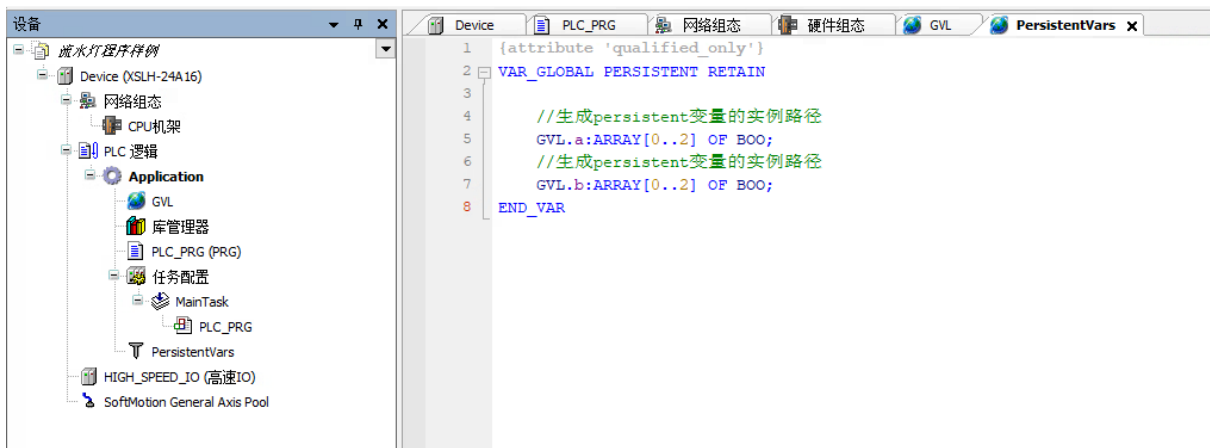
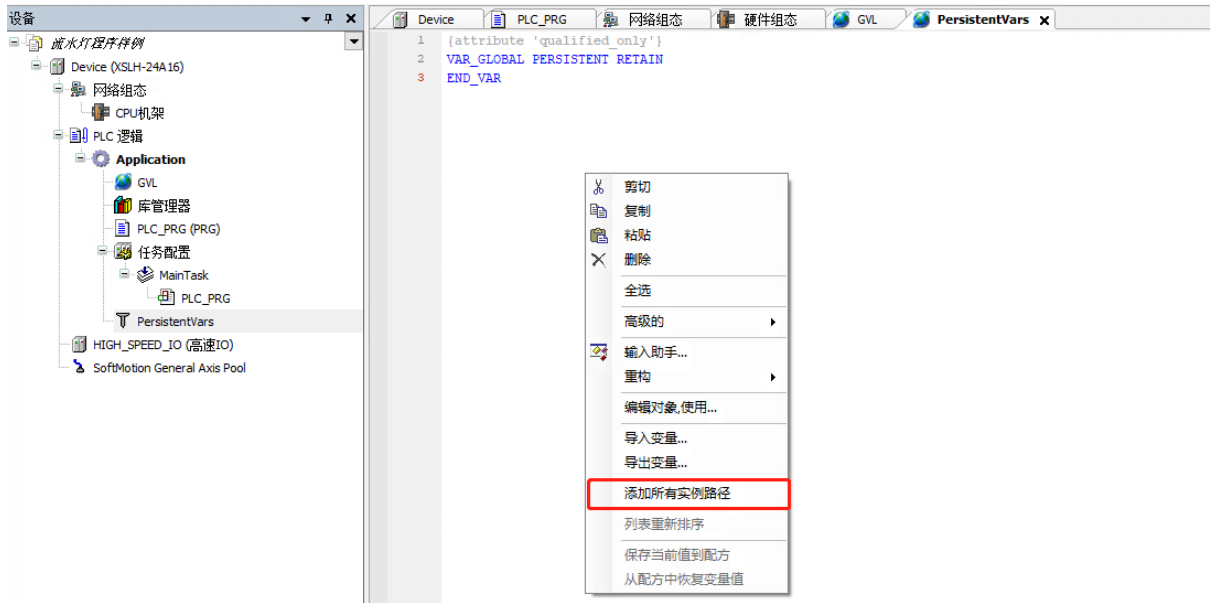
### 3、在 POU 中调用变量



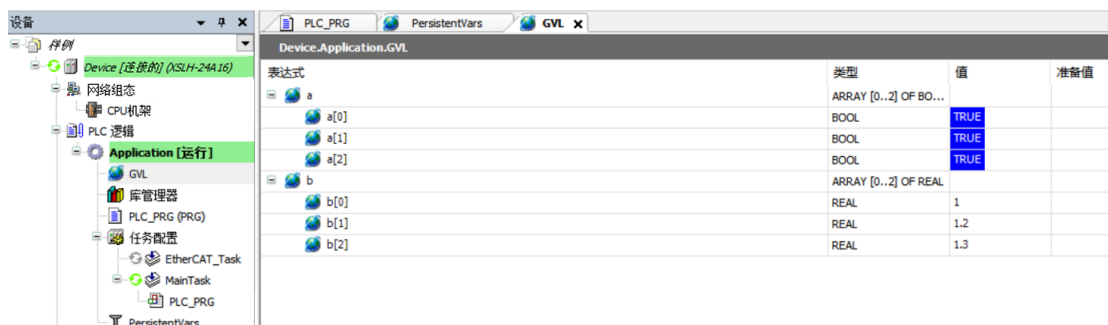
### 4、编译



5、编译，右键添加实例路径



6、登录下载程序，给变量赋值



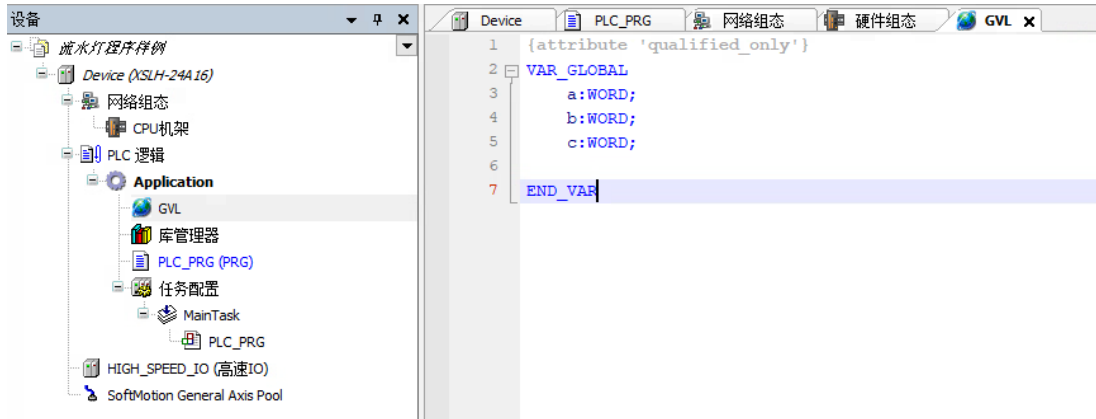
重新上电数据没有变化，掉电保持变量成功运行。

## 5-4. 配方操作

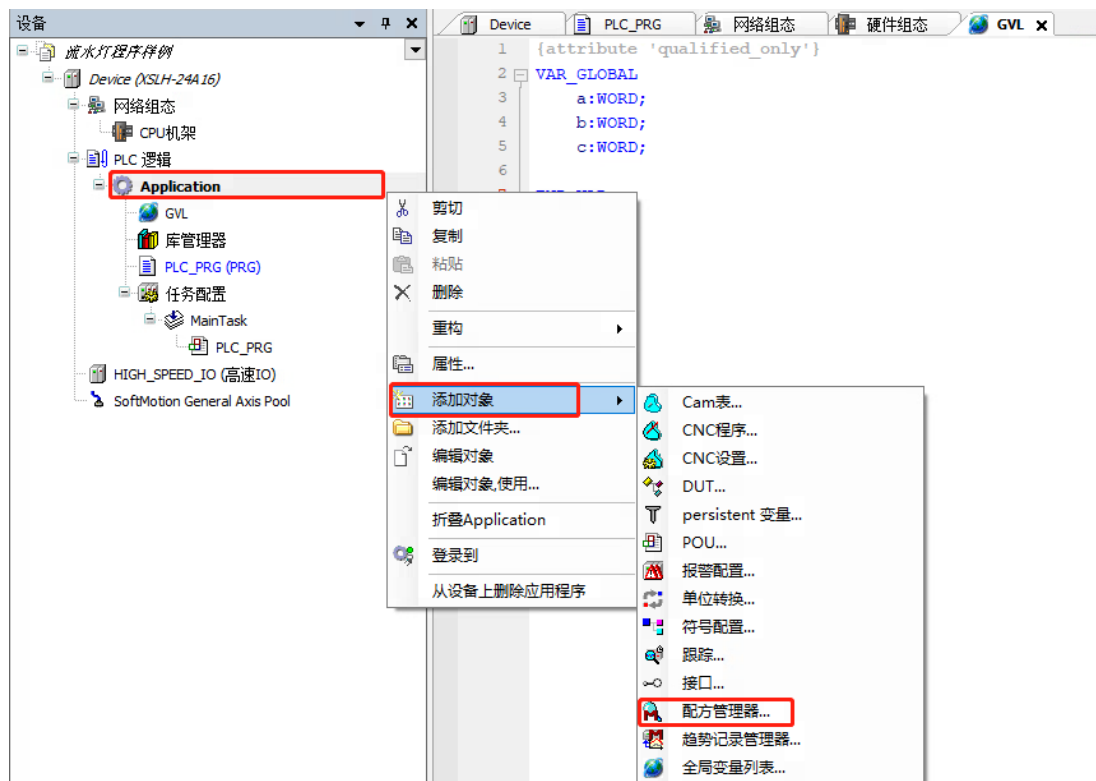
配方管理器的功能是提供用于维护用户定义的变量列表（配方定义）。用户通过配方管理器可以对存储位置、存储方式、存储类别进行配置，如下图所示。配方管理器配置成功后，用户即可实现对配方定义上传与下载。

### 5-4-1. 应用举例

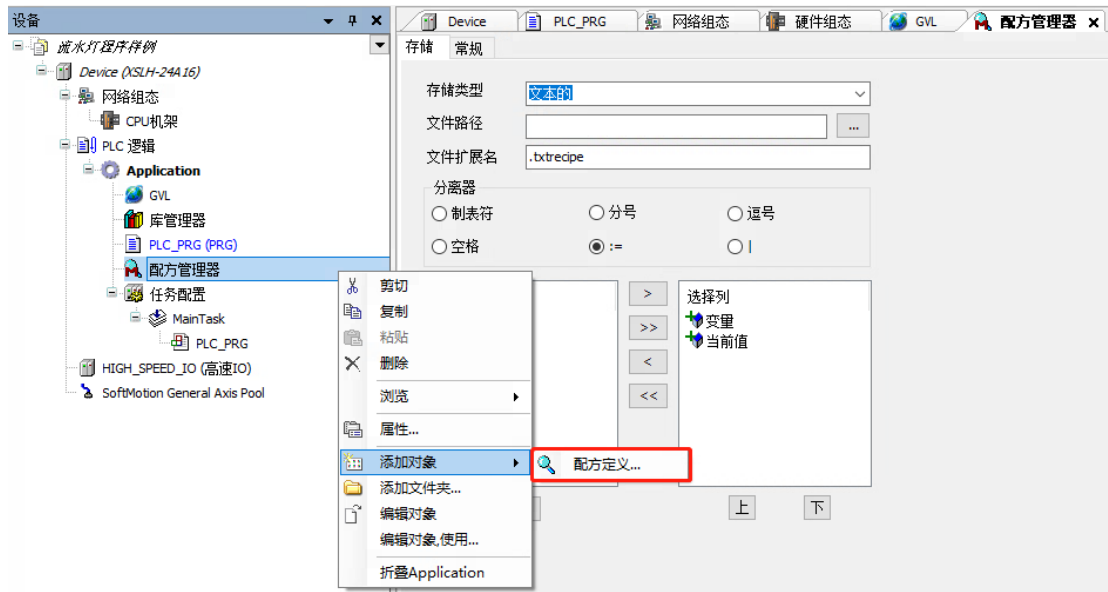
#### 1、创建变量



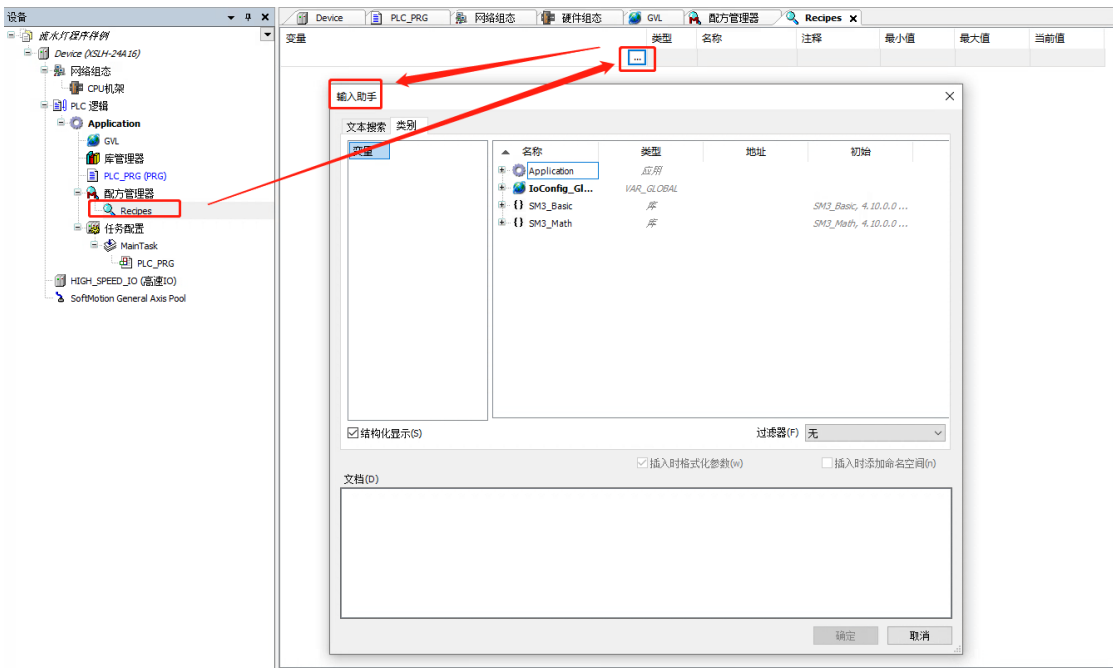
#### 2、建立配方管理器



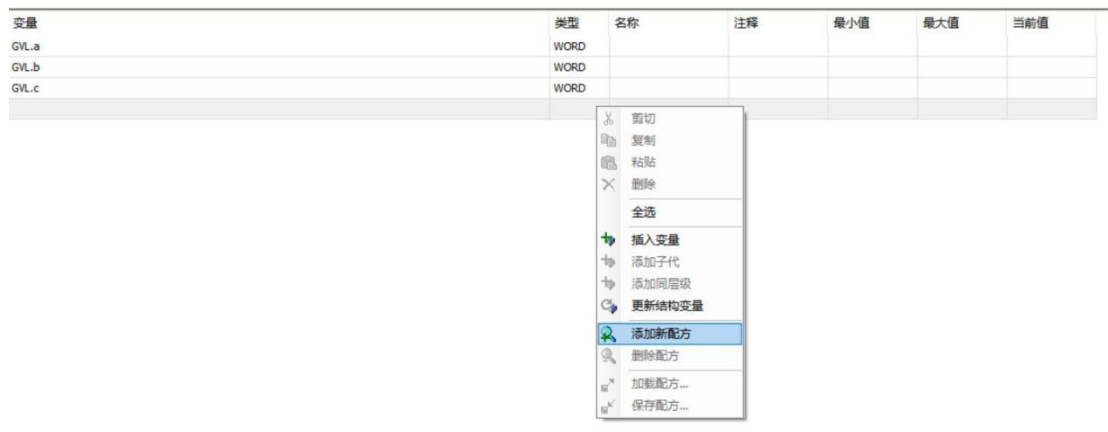
### 3、创建配方定义

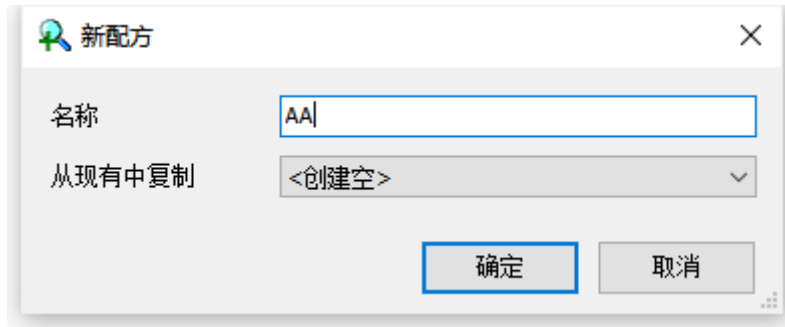


### 4、选择需要使用配方功能的变量

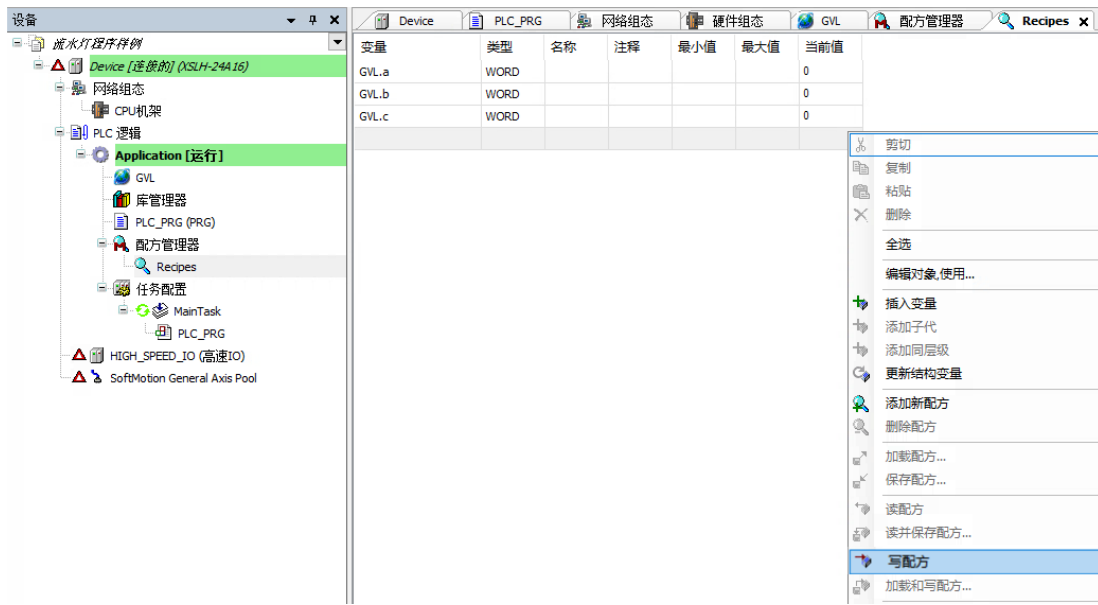


### 5、添加配方





6、登录设备在对应配方最下方表格右击可对配方进行写入读取等操作。



7、也可使用指令进行配方创建、读取、写入等操作。

① 编写配方创建、读取写入的程序。

例：

VAR

```
RecipeManCommands_0:Recipe_Management.RecipeManCommands;
READ:BOOL;
WRITE:BOOL;
CREAT:BOOL;
```

END\_VAR

IF READ THEN

```
RecipeManCommands_0.ReadRecipe(RecipeDefinitionName:= 'Recipes', RecipeName:= 'CC');//配方  
读取
```

END\_IF

IF WRITE THEN

```
RecipeManCommands_0.WriteRecipe(RecipeDefinitionName:= 'Recipes', RecipeName:= 'CC');//配方  
值写入
```

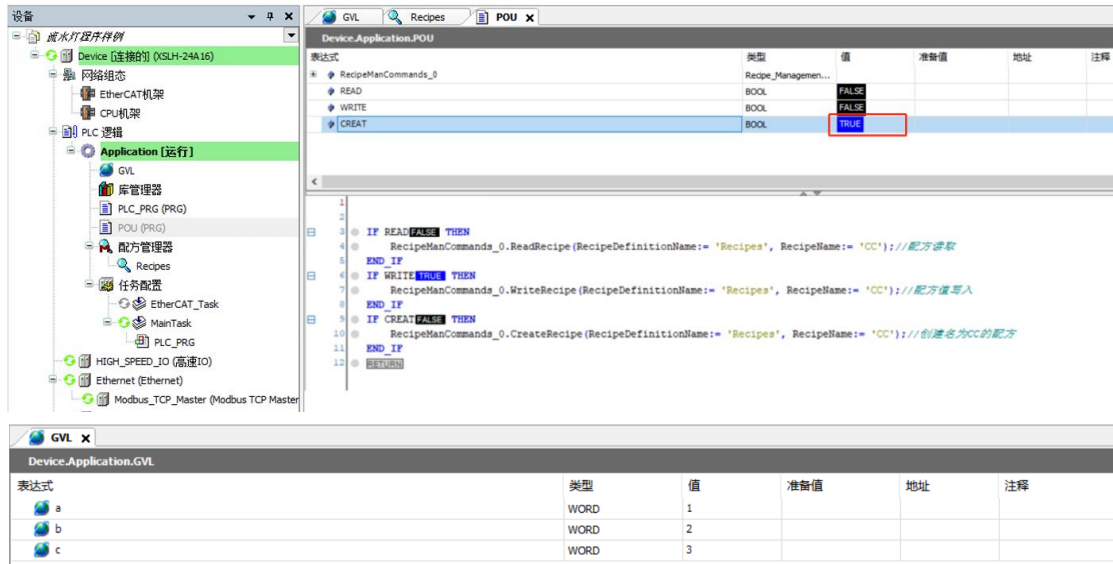
END\_IF

IF CREAT THEN

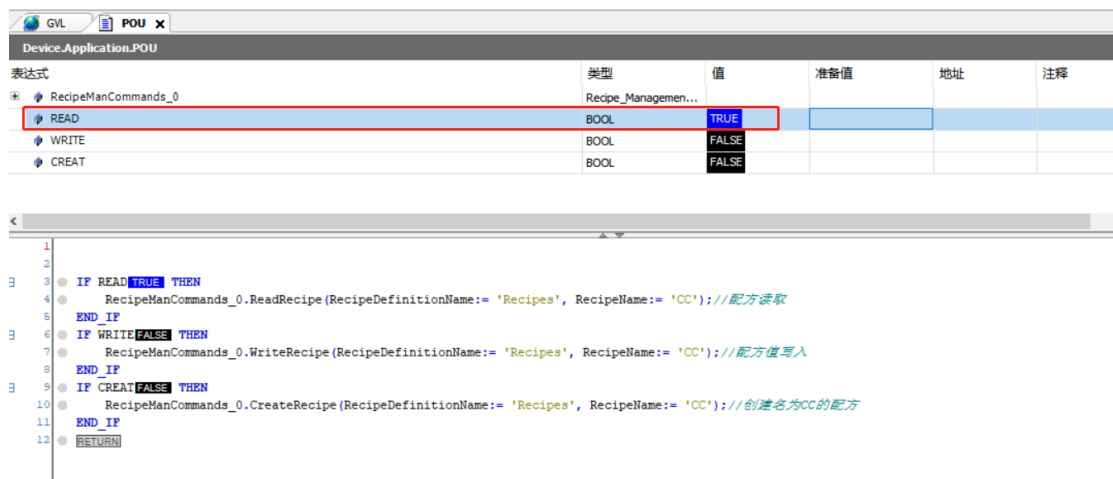
```
RecipeManCommands_0.CreateRecipe(RecipeDefinitionName:= 'Recipes', RecipeName:= 'CC');//创建  
名为 CC 的配方
```

END\_IF

登录创建配方



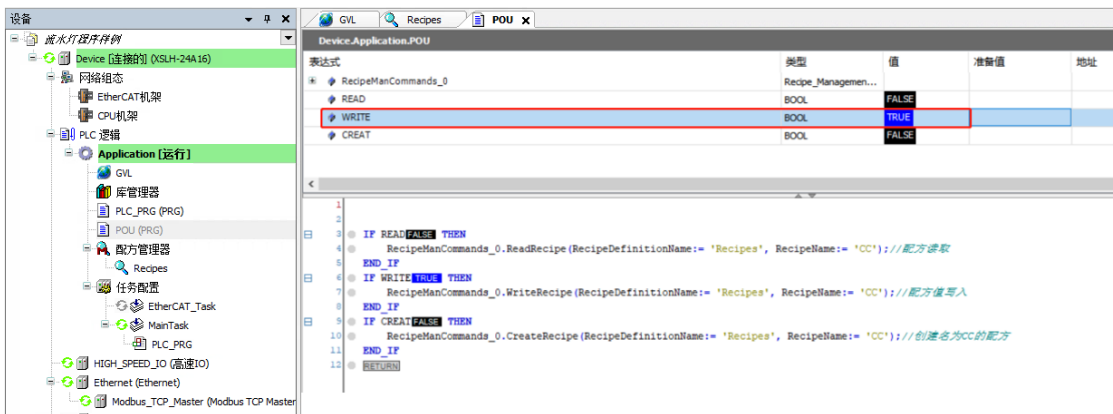
② 读取值到配方



③ 将当前值修改为其他值

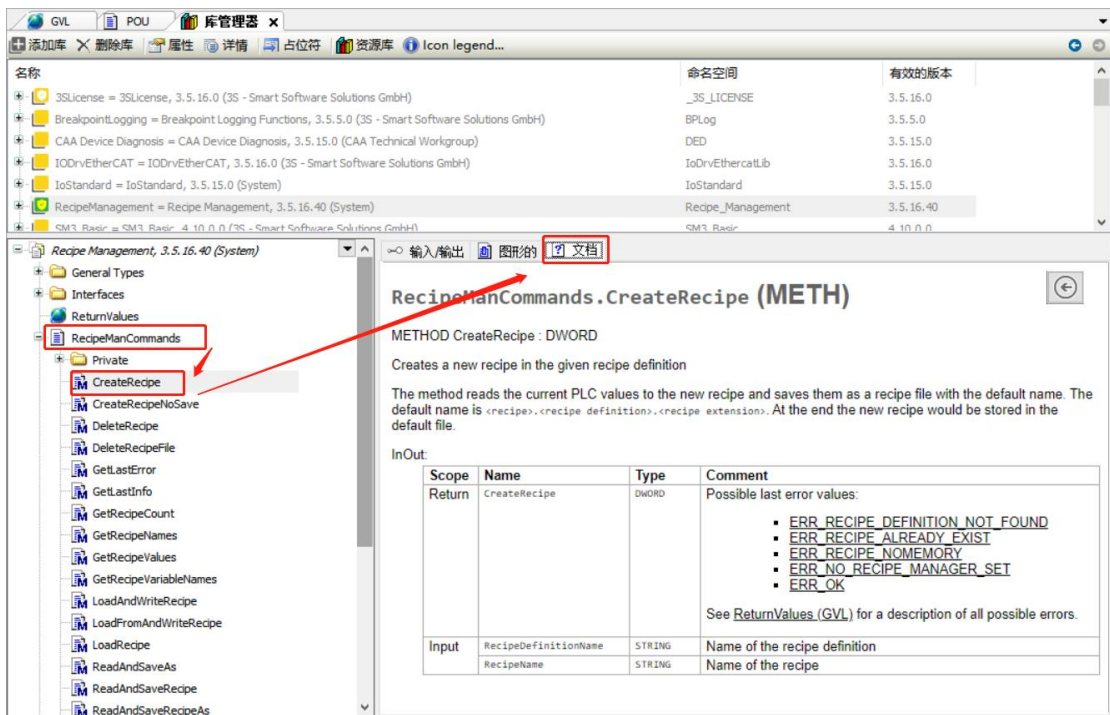
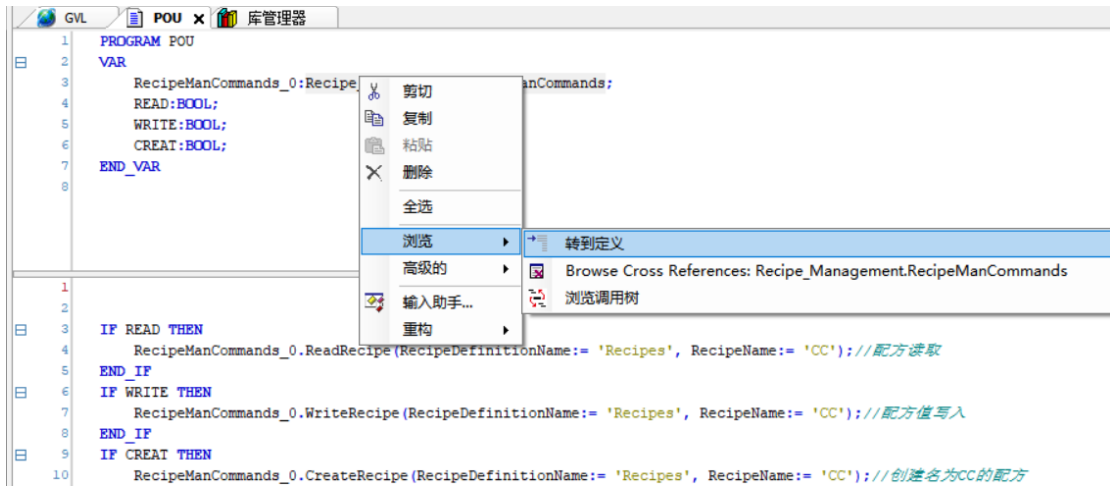


④ 写入配方值



表达式	类型	值	准备值	地址
a	WORD	1		
b	WORD	2		
c	WORD	3		

注意：配方功能其他功能可以在定义中查找使用





## 6. 编程语言

XS Studio 支持多种编程语言，本章主要介绍结构化文本语言和梯形图两种编程语言，详细说明了两种程序的执行顺序、使用方法。

---

6. 编程语言 .....	123
6-1. XS Studio 支持的编程语言 .....	124
6-2. 结构化文本语言 (ST) .....	124
6-2-1. 简介 .....	124
6-2-2. ST 程序执行顺序 .....	124
6-2-3. 语句 .....	126
6-3. 梯形图 .....	134
6-3-1. 简介 .....	134
6-3-2. LD 程序执行顺序 .....	134
6-3-3. 组成元素 .....	135

## 6-1. XS Studio 支持的编程语言

XS Studio 编程软件支持的 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

以上所有语言,在编辑器界面均支持标准的 Ctrl、Shift 编辑器功能快捷键。如复制(Ctrl+C)、粘贴(Ctrl+V)和“撤销”(Ctrl+Z)等快捷键;同时支持快捷键<F2>启动输入助手,系统会根据当前编程环境提供相对应的输入提示或选择。

## 6-2. 结构化文本语言 (ST)

### 6-2-1. 简介

结构化文本(ST)是一种高级的文本语言,可以用来描述功能,功能块和程序的行为,还可以在顺序功能流程图中描述步、动作和转变的行为。结构化文本编程语言是一种高级语言,类似于 Pascal,是一种特别为工业控制应用而开发的一种语言,也是在 XS Studio 中最常用的一种语言,对于熟悉计算机高级语言开发的人员来说,结构化文本语言更是易学易用,它可以实现选择、迭代、跳转语句等功能。此外,结构化文本语言还易读易理解,特别是当用有实际意义的标识符、批注来注释时,更是这样。在复杂控制系统中,结构化文本可以大大减少其代码量,使复杂系统问题变得简单,缺点是调试不直观,编译速度相对较慢。

例如:

```
FOR a:=0 TO 0 BY 1 DO
    D_温度显示值[a] :=TO_REAL(D_温度实际值[a]) / 10;
    D_温度最终值[a] := D_温度显示值[a] + D_温度补偿值[a];
END_FOR
```

```
IF M_自整定开关 THEN
    M_温控模式[0]:= 1;
END_IF
```

### 6-2-2. ST 程序执行顺序

#### 1、程序执行顺序

使用结构化文本的程序执行顺序根据“行号”依次从上至下开始顺序执行,如下图所示:

```

1  SMC3_ETC_WriteParameter_CoE_0(
2      xExecute:= WRITE, //上升沿触发
3      xAbort:= ,
4      uiIndex:=16#607D , //对象索引例如16#6060
5      usiSubIndex:=1 , //对象的子索引,例如0
6      usiDataLength:=4 , //写入数据的长度,以字节为单位(1 ~ 4)
7      dwValue:= 16#8000, //写入值 DWORD
8      Axis:=SM_Drive_GenericDSP402 , //”SoftMotion轴。
9      xDone=> ,
10     xBusy=> ,
11     xError=> ,
12     dwErrorCode=> ,
13     eError=> );
14 SMC3_ETC_WriteParameter_CoE_1(
15     xExecute:= WRITE,
16     xAbort:= ,
17     uiIndex:=16#607D ,
18     usiSubIndex:=2 ,
19     usiDataLength:=4 ,
20     dwValue:= 16#8000,
21     Axis:=SM_Drive_GenericDSP402 ,
22     xDone=> ,
23     xBusy=> ,
24     xError=> ,
25     dwErrorCode=> ,
26     eError=> );
    
```

行号

## 2、表达式执行顺序


表达式中包括操作符和操作数，操作数按照操作符指定的规则进行运算，得到结果并返回。操作数可以为变量、常量、寄存器地址、函数等。

a+b+c;

3.14\*R\*R;

ABS(-10)+var1;

如果在表达式中有若干个操作符，则操作符会按照约定的优先级顺序执行：先执行优先级高的操作符运算，再顺序执行优先级低的操作符运算。如果在表达式中具有优先级相同的操作符，则这些操作符按照书写顺序从左至右执行。操作符的优先级如下表所示：

操作符	符号	优先级
小括号	()	最高
函数调用	Function name (Parameter list)	
求幂	EXPT	
取反	NOT	
乘法	*	
除法	/	
取模	MOD	
加法	+	
减法	-	
比较	<, >, <=, >=	
等于	=	
不等于	<>	
逻辑与	AND	
逻辑异或	XOR	
逻辑或	OR	

## 6-2-3. 语句

结构化文本语句如下表所示：

指令类型	指令语句	举例
赋值语句	:=	bFan:= TRUE;
功能块/函数调用	功能块/函数调名();	
选择语句	IF	IF <布尔表达式> THEN <语句内容>; END_IF
	CASE	
迭代语句	FOR	
	WHILE	
	REPEAT	
跳转语句	EXIT	
	CONTINUE	
	JMP	
返回语句	RETURN	
NULL 语句	;	

## 1、赋值语句

是结构化文本中最常用的语句之一，作用是将其右侧的表达式产生的值赋给左侧的操作数（变量或地址），使用“:=”表示。

< 变量>:=< 表达式>;

例子：分别给两个布尔型变量赋值，bFan 置 TRUE，bHeater 置 FALSE。

VAR

bFan: BOOL;

bHeater:BOOL;

END\_VAR

bFan:= TRUE;

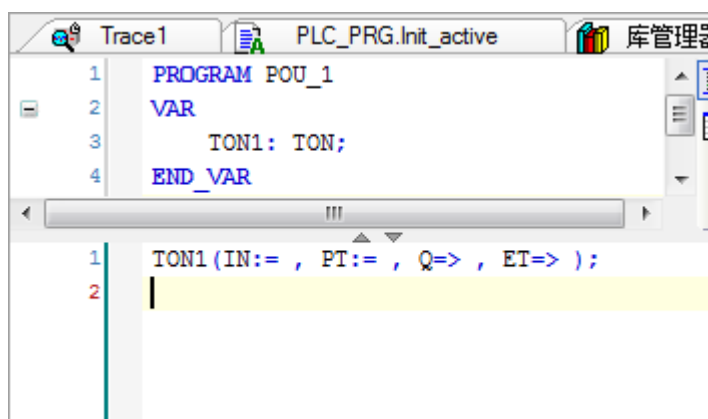
bHeater:= TRUE;

## 2、函数及功能块调用

功能块调用采用将功能块名进行实例化实现调用，如 Timer 为 TON 功能块的实例名，具体格式如下  
功能块实例名:(功能块参数);

如果需要在 ST 中调用功能块，可直接输入功能块的实例名称，并在随后的括号中给功能块的各参数分配数值或变量，参数之间以逗号隔开；功能块调用以分号结束。

例如，在结构化文本中调用功能块 TON 定时器，假设其实例名为 TON1，具体实现如图所示：



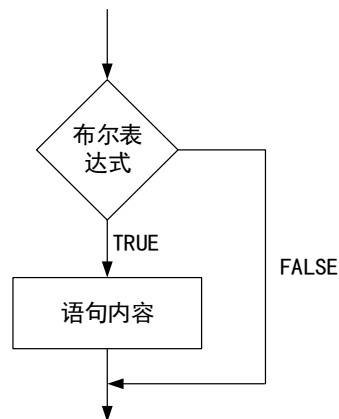
### 3、选择语句

#### 1) IF语句

用 IF 语句实现单分支选择结构，基本格式如下：

```
IF <布尔表达式> THEN
<语句内容>;
END_IF
```

如果使用上述格式，只有当<布尔表达式>为 TRUE 时，才执行语句内容，否则不执行 IF 语句的<语句内容>。语句内容可以为一条语句或者可以为空语句，也可以并列多条语句，该语句表达式执行流程图如图所示：

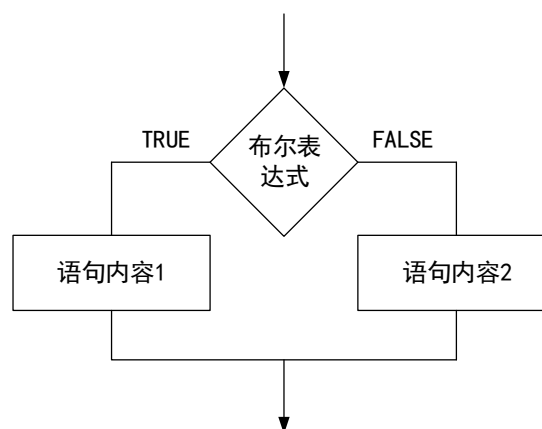


#### 2) IF...ELSE语句

用 IF 语句实现双分支选择机构，基本格式如下：

```
IF <布尔表达式> THEN
<语句内容 1>;
ELSE
<语句内容 2>;
END_IF
```

如上表达式先判断<布尔表达式>内的值，如果为 TRUE，则执行<语句内容 1>，如为 FALSE，则执行<语句内容 2>，程序执行流程图如图所示：



当程序的条件判断式不止一个时，此时，需要再一个嵌套的 IF...ELSE 语句，即多分支选择结构，基本格式如下。

```
IF <布尔表达式 1> THEN
  IF <布尔表达式 2> THEN
```

```

    <语句内容 1>;
ELSE
    <语句内容 2>;
END_IF
ELSE
    <语句内容 3>;
END_IF

```

如上，在 IF...ELSE 中有放入了一个 IF...ELSE 语句，实现嵌套，下面通过一个例子说明嵌套的使用。如上表达式先判断<布尔表达式 1>内的值，如果为 TRUE，则继续判断<布尔表达式 2>的值，如果<布尔表达式 1>的值为 FALSE，则执行<语句内容 3>，回到<布尔表达式 2>判断，如果<布尔表达式 2>为 TRUE，则执行<语句内容 1>，反之则执行<语句内容 2>。

### 3) IF..ELSIF..ELSE 语句

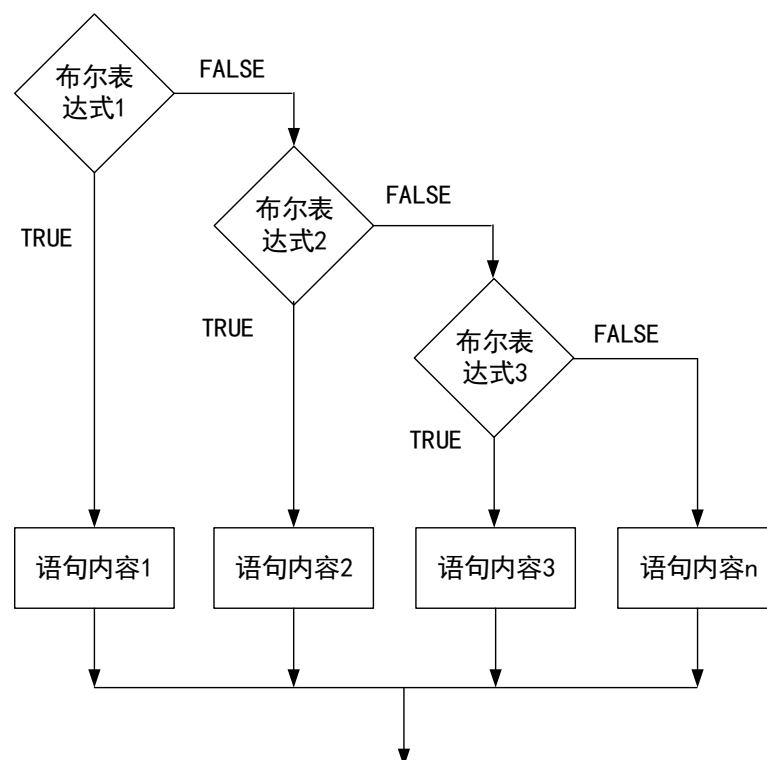
此外，多分支选择结构还能通过如下方式来呈现。具体格式如下

```

IF < 布尔表达式 1> THEN
< 语句内容 1>;
ELSIF < 布尔表达式 2> THEN
< 语句内容 2>;
ELSIF < 布尔表达式 3> THEN
< 语句内容 3>;
...
...
ELSE
< 语句内容 n>;
END_IF

```

如果表达式<布尔表达式 1>为 TRUE，那么只执行指令<语句内容 1>，不执行其它指令。否则，从表达式<布尔表达式 2>开始进行判断，直到其中的一个布尔表达式为 TRUE，然后执行与此布尔表达式对应语句内容。如果布尔表达式的值都不为 TRUE，则只执行指令<语句内容 n>，程序执行流程图如图所示。：



#### 4) CASE语句

CASE 语句是多分支选择语句，他根据表达式的值来使程序从多个分支中选择一个用于执行的分支，基本格式如下：

```
CASE < 条件 变量> OF
< 数值 1>:< 语句内容 1>;
< 数值 2>:< 语句内容 2>;
< 数值 3, 数值 4, 数值 5>:< 语句内容 3>;
< 数值 6.. 数值 10>:< 语句内容 4>;
```

...

```
< 数值 n>:< 语句内容 n>;
```

```
ELSE
```

```
<ELSE 语句内容>;
```

```
END_CASE;
```

CASE 语句按照下面的模式进行执行：

- ◆ 如果<条件变量>的值为<数值 i>，则执行指令<语句内容 i>。
- ◆ 如果<条件变量>没有任何指定的值，则执行指令< ELSE 语句内容>。
- ◆ 如果条件变量的几个值都需要执行相同的指令，那么可以把几个值相继写在一起，并且用逗号分开。这样，共同的指令被执行，如上程序第四行。
- ◆ 如果需要条件变量在一定的范围内执行相同的指令，可以通过写入初、终值，以两个点分开。这样，共同的指令被执行，如上程序第五行。

#### 4、迭代语句

迭代语句主要用于重复执行的程序，在 XS Studio 中，常见的迭代语句有 FOR, REPEAT 及 WHILE 语句，下面对这种语句做详细解释：

##### 1) FOR循环

FOR 循环语句用于计算一个初始化序列，当某个条件为 TRUE 时，重复执行嵌套语句并计算一个迭代表达式序列，如果为 FALSE，则终止循环，具体格式如下。

```
FOR < 变量> := < 初始值> TO < 目标值> {BY < 步长>} DO
< 语句内容>
END_FOR;
```

FOR 循环的执行顺序如下：

- ◆ 计算<变量>是否在<初始值>与<目标值>的范围内。
- ◆ 当<变量>小于<目标值>，执行<语句内容>。
- ◆ 当<变量>大于<目标值>，则不会执行<语句内容>。
- ◆ 当每次执行<语句内容>时，<变量>总是按照指定的步长增加其值。步长可以是任意的整数值。

如果不指定步长，则其缺省值是 1。当<变量>大于<目标值>时，退出循环。

从某种意义上理解，FOR 循环的原理就像复印机，在复印机上先预设要复印的份数，在此即循环的条件，当条件满足时，即复印的张数等于设置的张数，停止复印。

FOR 循环是循环语句中最常用的一种，FOR 循环体现了一种规定次数、逐次反复的功能，但由于代码编写方式不同，也可以实现其他循环功能，下面通过一个实例，演示如何使用 FOR 循环。

例：使用 FOR 循环实现 2 的五次方计算。

```
VAR
Counter: BYTE; (*循环计数器*)
Var1: WORD; (*输出结果*)
END_VAR
```

```
FOR Counter:=1 TO 5 BY 1 DO
```

```
Var1:=Var1*2;
END_FOR;
```

假设 Var1 的初始值是 1，那么循环结束后，Var1 的值为 32。

**注意：**如果<目标值>等于<变量>的极限值，则会进入死循环。假设【例 5.X】中的计数变量 Counter 的类型为 SINT (-128 至 127)，将<目标值>设定为 127 时，控制器会进入死循环。故不能对<目标值>设极限值。

## 2) WHILE 循环

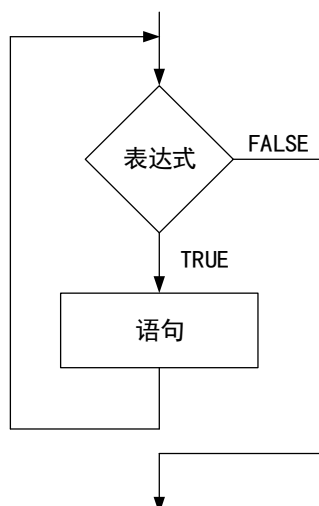
WHILE 循环与 FOR 循环使用方法类似。二者的不同之处是，WHILE 循环的结束条件可以是任意的逻辑表达式。即可以指定一个条件，当满足该条件时，执行循环，具体格式如下。

```
WHILE <布尔表达式>
<语句内容>;
END_WHILE;
```

WHILE 循环的执行顺序如下：

- ◆ 计算<布尔表达式>的返回值。
- ◆ 当<布尔表达式>的值为 TRUE 时，重复执行<语句内容>。
- ◆ 当<布尔表达式>初始值为 FALSE，那么指令<语句内容>不会被执行，跳转至 WHILE 语句的结尾。其流程图见图所示：

其流程图见图所示：



**注意：**如果<布尔表达式>的值始终为 TRUE，那么将会产生死循环，应当避免死循环的产生。可以通过改变循环指令的条件来避免死循环的产生。例如：利用可增减的计数器避免死循环的产生。

WHILE 语句在像工程中控制一台电机，当按下“启动”按钮时（布尔表达式为 TRUE 时），电机不停的旋转，当按下“停止”按钮后（布尔表达式为 FALSE 时），电机也立即停止。下面通过一个实例，演示如何使用 WHILE 循环。

例：只要计数器不为零，则始终执行循环体内的程序。

```
VAR
Counter: BYTE; (*计数器*)
Var1: WORD;
END_VAR
```

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

在一定的意义上，WHILE 循环比 FOR 循环的功能更加强大，这是因为在执行循环之前，WHILE 循环不需要知道循环的次数。因此，在有些情况下，只使用这两种循环就可以了。然而，如果清楚地知道了循环的次数，那么 FOR 循环更好，因为 FOR 循环可以避免产生死循环。



### 3) REPEAT 循环

REPEAT 循环与 WHILE 循环不同，因为只有在指令执行以后，REPEAT 循环才检查结束条件。这就意味着无论结束条件怎样，循环至少执行一次。

```
REPEAT
< 语句内容>
UNTIL
< 布尔表达式>
END_REPEAT;
```

REPEAT 循环的执行顺序如下：

- ◆ 当<布尔表达式>的值为 FALSE 时，执行<语句内容>。
- ◆ 当<布尔表达式>的值为 TRUE 时，停止执行<语句内容>。
- ◆ 在第一次执行<语句内容>后，如果<布尔表达式>的值为 TRUE，那么<语句内容>只被执行一次。

**注意：**如果<布尔表达式>的值始终为 TRUE，那么将会产生死循环，应当避免死循环的产生。可以通过改变循环指令部分的条件来避免死循环的产生。例如：利用可增减的计数器避免死循环的产生。

例：REPEAT 循环举例，当计数器为 0 时，则停止循环。

```
VAR
Counter: BYTE;
END_VAR

REPEAT
Counter := Counter+1;
UNTIL
Counter=0
END_REPEAT;
```

此例的结果为，每个程序周期都进入该 REPEAT 循环，Counter 为 BYTE (0-255)，即每个周期内都进行了 256 次自加一计算。

因为之前提到的“这就意味着无论结束条件怎样，循环至少执行一次”，所以每当进入该 REPEAT 语句时，Counter 先为 1，每个周期内都执行了 256 遍的 Counter := Counter+1 指令，直到将 Counter 变量累加至溢出为 0，跳出循环。再被加到溢出，如此往复。

## 5、跳转语句

### 1) EXIT语句

如果 FOR、WHILE 和 REPEAT 三种循环中使用了 EXIT 指令，那么无论结束条件如何，内循环立即停止，具体格式如下：

```
EXIT;
例子：使用 EXIT 指令避免当使用迭代语句时中出现除零。
FOR Counter:=1 TO 5 BY 1 DO
INT1:= INT1/2;
IF INT1=0 THEN
EXIT; (* 避免程序除零*)
END_IF
Var1:=Var1/INT1;
END_FOR
当 INT1 等于 0 时，则 FOR 循环结束。
```

### 2) CONTINUE语句

该指令为 IEC 61131-3 标准的扩展指令，CONTINUE 指令可以在 FOR、WHILE 和 REPEAT 三种循环中使用。

CONTINUE 语句中断本次循环，忽略位于它后面的代码而直接开始一次新的循环。当多个循环嵌套时，CONTINUE 语句只能使直接包含它的循环语句开始一次新的循环，具体格式如下：

```
CONTINUE;
例：使用 CONTINUE 指令避免当使用迭代语句时中出现除零。
VAR
Counter: BYTE; (*循环计数器*)
```

```

INT1,Var1: INT; (*中间变量*)
Erg: INT; (*输出结果*)
END_VAR

FOR Counter:=1 TO 5 BY 1 DO
INT1:= INT1/2;
IF INT1=0 THEN
CONTINUE; (* 为了避免除零 *)
END_IF
Var1:=Var1/INT1;(*只有当 INT1 不等于 0 的情况下才执行*)
END_FOR;
Erg:=Var1;

```

### 3) JMP 语句

跳转语句，跳转指令可以用于无条件跳转到使用跳转好标记的代码行，具体格式如下：

< 标识符>:

```
JMP < 标识符>;
```

<标识符>可以是任意的标识符，它被放置在程序行的开始。JMP 指令后面为跳转目的地，即一个预先定义的标识符。当执行到 JMP 指令时，将跳转到标识符所对应的程序行。

**注意：**必须避免制造死循环，可以配合使用 IF 条件控制跳转指令。

例：使用 JMP 语句实现计数器在 0..10 范围内循环。

```

VAR
nCounter: BYTE;
END_VAR

Label1:nCounter:=0;
Label2:nCounter:=nCounter+1;
IF nCounter<10 THEN
JMP Label2;
ELSE
JMP Label1;
END_IF

```

上例中的 Label1 和 Label2 属于标签，不属于变量，故在程序中不需要进行变量声明。

通过 IF 语句判断计数器是否在 0-10 的范围内，如果在范围内，则执行语句 JMP Label2，程序会在下一个周期跳转到至 Label2，执行程序 nCounter:=nCounter+1，将计数器进行自加 1，反之，则会跳转至 Label1，执行 nCounter:=0，将计数器清零。

此例子中的功能同样可以通过使用 FOR，WHILE 或者 REPEAT 循环来实现。通常情况下，应该避免使用 JMP 跳转指令，因为这样降低了代码的可读性和可靠性。

### 4) RETURN 指令

RETURN 指令是返回指令，用于退出程序组织单元 (POU)，具体格式如下：

```
RETURN;
```

例：使用 IF 语句作为判断，当条件满足时，立即终止执行本程序。

```

VAR
nCounter: BYTE;
bSwitch: BOOL; (*开关信号*)
END_VAR

```

```

IF bSwitch=TRUE THEN
RETURN;
END_IF;
nCounter:= nCounter +1;

```

当 bSwitch 为 FALSE 时，nCounter 始终执行自加 1，如 bSwitch 为 TRUE 时，nCounter 保持上一周期的数值，立刻退出此程序组织单元 (POU)。

## 6、空语句

即什么内容都不执行。

具体格式如下。

;

## 7、注释

注释是程序中非常重要的一部分，它使程序更加具有可读性，同时不会影响程序的执行。在 ST 编辑器的声明部分或执行部分的任何地方，都可以添加注释。

在 ST 语言中，有两种注释方法：

1) 多行注释以 (\*) 开始，以 \*) 结束。这种注释方法允许多行注释，如下图所示：

```
(*  
    bOperationActive:=FALSE;  
    bOrderActive:=FALSE;  
    bRecipeActive:=FALSE;  
    bInfoActive:=FALSE;  
    bServiceActive:=FALSE;  
    bSimulationActive:=FALSE;  
*)  
IF iMainAreaIndex = 0 THEN  
    bOperationActive:=TRUE;  
ELSIF iMainAreaIndex = 1 THEN  
    bOrderActive:=TRUE;
```

单行注释以 “//” 开始，一直到本行结束。这是单行注释的方法，如下图所示：

```
// gesture handling:  
// only when mouseup was done  
IF xRight AND bDragCanStart = FALSE THEN  
    xRight := FALSE;  
    IF iMainAreaIndex < MAX_MODULES-1 THEN  
        iMainAreaIndex := iMainAreaIndex + 1;  
        bIndexChanged := TRUE;  
    END_IF
```

## 6-3. 梯形图

### 6-3-1. 简介

梯形图来源于美国，它基于图形表示的继电器逻辑，是 PLC 编程中被最广泛使用一种图形化语言。梯形图程序的左、右两侧有两垂直的电力轨线，左侧的电力轨线名义上为功率流从左向右沿着水平梯级通过各个触点、功能、功能块、线圈等提供能量，功率流的终点是右侧的电力轨线。每一个触点代表了一个布尔变量的状态，每一个线圈代表了一个实际设备的状态，功能或功能块与 IEC 1131-3 中的标准库或用户创建的功能或功能块相对应。

梯形图是国内应用最为广泛的编程语言，它也是 IEC 1131-3 的三种图形化编程语言种一种，梯形图是传统 PLC 使用得最多的图形编程语言，也被称为 PLC 的第一编程语言。根据梯形图中各触点的状态和逻辑关系，求出与图中各线圈对应的编程元件的状态，称为梯形图的逻辑解算。

梯形图中的某些编程元件沿用了继电器这一名称，如线圈、触点等，但是它们不是真实的物理继电器，而是一些存储单元（软继电器），每一软继电器与 PLC 存储器中映像寄存器的一个存储单元相对应。该存储单元如果为“TRUE”状态，则表示梯形图中对应软继电器的线圈“通电”，其常开触点接通，常闭触点断开，称这种状态是该软继电器的“TRUE”或“ON”状态。如果该存储单元为“FALSE”状态，对应软继电器的线圈和触点的状态与上述的相反，称该软继电器为“FALSE”或“OFF”状态。使用中也常将这些“软继电器”称为编程元件。

### 6-3-2. LD 程序执行顺序

梯形图的执行过程是按照从左至右，从上到下的顺序进行执行，如图所示：



#### 1、执行过程

##### 1) 母线

梯形图采用网络结构，一个梯形图的网络以左母线为界。在分析梯形图的逻辑关系时，为了借用继电器电路图的分析方法，可以想象左右两侧母线（左母线和右母线）之间有一个左正右负的直流电源电压，母线之间有“能流”从左向右流动。右母线不显示。

##### 2) 节

节是梯形图网络结构中最小单位，从输入条件开始，到一个线圈的有关逻辑的网络称为一个节。在编辑器中，节垂直排列。在 XS Studio 中，每个节通过左侧的一系列节号标示，包含输入指令和输出指令，由逻辑式，算术表达式，程序，功能或功能块调用指令，跳转或返回指令所构成。

要插入一个节，可以使用命令插入节或从工具箱拖动它。一个节所包含的元素都可以通过在编辑器中拖放来进行复制或移动。

梯形图执行时，从标号最小的节开始执行，从左到右确定各元素的状态，并确定其右侧连接元素的状态，逐个向右执行，操作执行的结果由执行控制元素输出。然后进行下一节的执行过程。上图显示了梯形图的执行过程。

##### 3) 能流

如上图所示，蓝色的加粗线即为能流，可以理解为一个假想的“概念电流”或“能流”

(PowerFlow)从左向右流动，这一方向与执行用户程序时的逻辑运算的顺序是一致的。能流只能从左向右流动。利用能流这一概念，可以帮助我们更好地理解和分析梯形图。

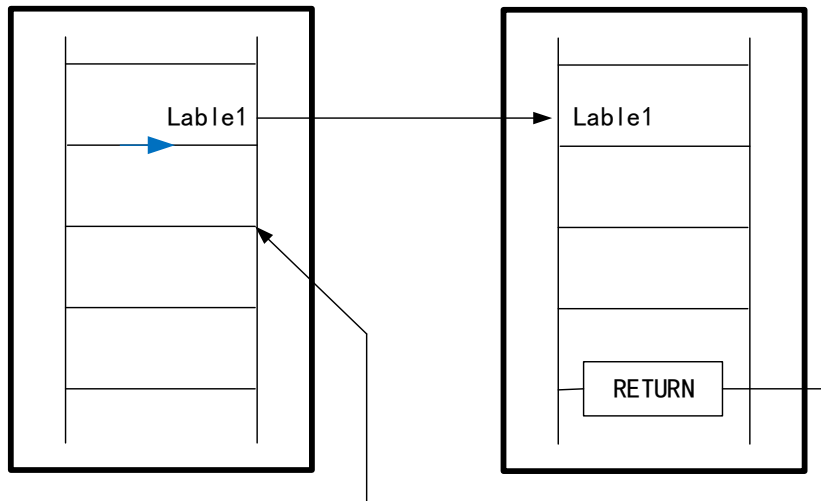
4) 分支

当梯形图中有分支出现时，同样依据从上到下，从左至右的执行顺序分析各图形元素的状态，对垂直连接元素根据上述有关规定确定其右侧连接元素的状态，从而逐个从左向右，从上向下执行求值过程。在梯形图中，没有反馈路径的求值不是很明确。其所有外部输入值与这些有关的触点必须在每个梯级以前被求值。

2、执行控制

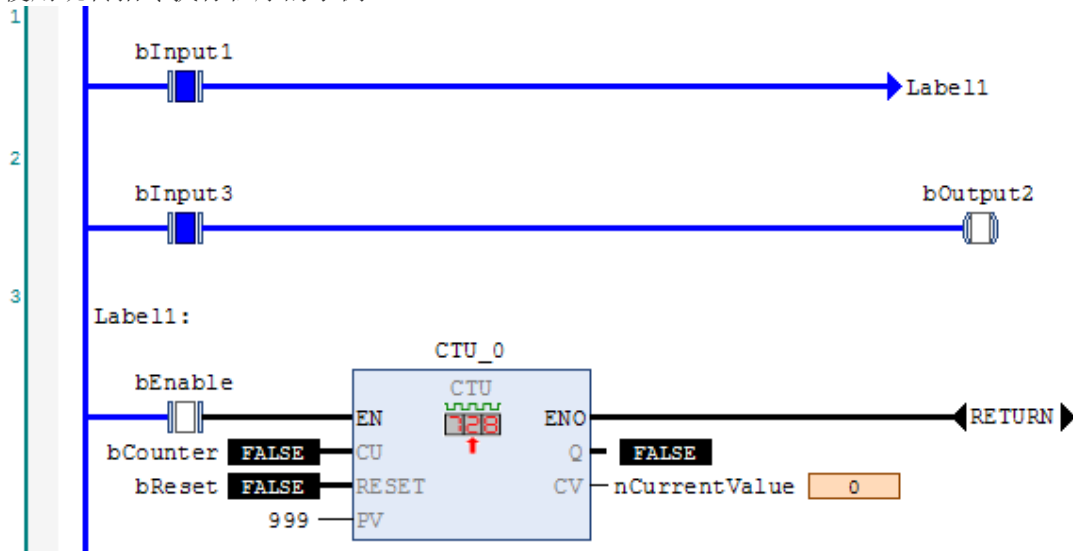
1) 跳转及返回

当满足跳转条件时，程序跳转到 Label 标号的节开始执行，直至该部分程序执行到 RETURN 时，返回原来的节并继续执行。其结构图如下图所示：



XS Studio 中使用梯形图的跳转指令和返回指令如下，如例 5.X 所示：

例：使用跳转指令执行程序的示例：



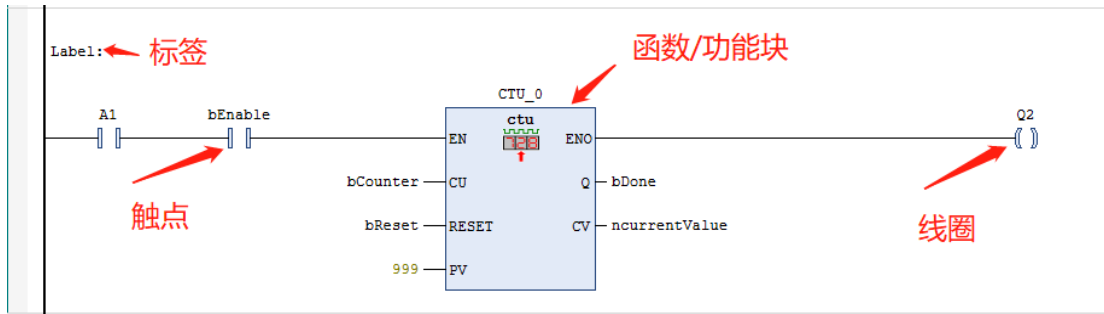
如图所示，bInput1 被置为 True，故执行跳转语句，根据标签 Label1，程序跳转至第 3 节的 Label，所以尽管当第 2 节的 bInput3 被置为 ON，bOutput2 始终不会被置为 True，因为程序直接跳过了该语句。只有当 B1 为 False 时，且 bInput3 为 True 时，bOutput2 才会 True。

6-3-3. 组成元素

IEC 1131-3 中的梯形图语言是对各 PLC 厂家的梯形图语言合理地吸收、借鉴，语言中的各图形符号与各 PLC 厂家的基本一致，图 5.x 为梯形图编辑器视图。IEC 61131-3 的主要的图形符号包括：

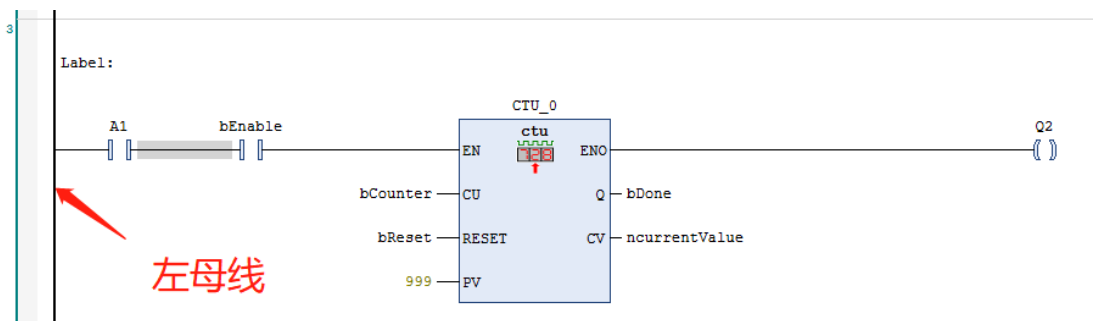
- 基本连接类：电源柜先、连接元素。

- 触点类：常开触点、常闭触点、正转换读出触点、负转换触点。
- 线圈类：一般线圈、取反线圈、置位（锁存）线圈、复位去锁线圈、保持线圈、置位保持线圈、复位保持线圈、正转换读出线圈、负转换读出线圈。
- 功能和功能块：包括标准的功能和功能块以及用户自己定义的功能块。



### 1、电源轨线

梯形图电源轨线（Power Rail）的图形元素亦称为母线。其图形表示是位于梯形图左侧，也可称其为左电源母线。



### 2、连接元素

在梯形图中，各图形符号用连接元素连接，连接元素的图形符号有水平线和垂直线，它是构成梯形图的最基本元素。下图是水平和垂直连接元素的图形表示：



### 3、标签

标签是一个可选的标识符且当定义 跳转 时可以确定其地址。它可以包含任何字符。

### 4、触点

触点是梯形图的图形元素。梯形图的触点（Contact）沿用了电气逻辑图的触点术语，用于表示布尔型变量的状态变化。触点是向其右侧水平连接元素传递一个状态的梯形图元素。

触点可以分为常开触点（Normally Open Contact, NO）和常闭触点（Normally Closed Contact, NC）。常开触点指在正常工况下，触点断开，其状态为 FALSE。常闭触点指在正常工况下，触点闭合，其状态为 True。表 2-6-2-1 列出了梯形图中常用的触点图形符号及说明。

类型	图形符号	说明
常开触点		如果该触点对应当前布尔变量值为 True 时，则该触点吸合，如触点左侧连接元素的状态为 True 时，则状态 True 被传递至该触点右侧，使右侧连接元素的状态为 True。反之，当布尔变量值为 False 时，右侧连接元素状态为 False。
常闭触点		如果该触点对应当前布尔变量值为 False 时，则该常闭触点处于吸合状态，如触点左侧连接元素的状态为 True 时，则状态 True 被传递至该触点右侧，使右侧连接元素的状态为 True。反之，当布尔变量值为 True 时，触点断开，则右

类型	图形符号	说明
		侧连接元素状态为 False。
插入右触点		可以进行多个触点的串联，在右侧插入触点。多个串联的触点都为吸合状态时，最后一个触点才能传输 True。
插入并联下常开触点		可以进行多个触点的并联，在触点下侧并联插入常开触点。两个并联触点中只需一个触点为 True，则平行线传输 True。
插入并联下常闭触点		可以进行多个触点的并联，在触点下侧并联插入常闭触点。常闭触点默认为吸合状态，如该触点对应当前布尔变量值为 False 时，左侧连接元素的状态为 True 时，则该并联触点右侧传输 True。
插入并联上常开触点		可以进行多个触点的并联，在触点上侧并联插入常开触点。两个并联触点中只需一个触点为 True，则平行线传输 True。

## 5、线圈

线圈是梯形图的图形元素。梯形图中的线圈沿用了电气逻辑图的线圈术语，用于表示布尔型变量的状态变化。根据线圈的不同特性，可以分为瞬时线圈和锁存线圈，锁存线圈分为置位线圈和复位线圈。下表列出了梯形图中常用的线圈图形符号及说明。

类型	图形符号	说明
线圈		左侧连接元素的状态被传递到有关的布尔变量和右侧连接元素，如果线圈左侧连接元素的状态为 TRUE，则线圈的布尔变量为 TRUE，反之线圈为 FALSE。
置为线圈		线圈中有一个 S。当左侧连接元素的状态为 TRUE 时，该线圈的布尔变量被置位并且保持，直到由 Reset（复位）线圈的复位。
复位线圈		线圈中有一个 R。当左侧连接元素的状态为 TRUE 时，该线圈的布尔变量被复位并且保持，直到由 Set（置位）线圈的置位。

## 6、辅助

可以针对线圈及触点进行边沿检测、取反及置位/复位操作。



类型	图形符号	说明
取反		将信号取反。
边沿检测		有 P, N 两种模式，可以通过单击该工具进行切换。P 为采集信号上升沿触发，N 为采集信号下降沿触发。
置位/复位		有 R, S 两种模式，可以通过单击该工具进行切换。S 为置位，R 为复位。

例如，LD 中的网络右侧，可以有任意数量的线圈，它用括号“()”表示。它们只能并联。一个线圈将连接的值从左传输到右，并将它复制到一个相应的布尔变量。在进入线处，可以出现值 ON（相当于布尔变量 TRUE），或值 OFF（相当于布尔变量 FALSE）。也可对接点和线圈求反（示例中，接点 SWITCH1 和线圈 %QX3.0 是求反的）。若一个线圈是求反的（可通过线圈符号中的斜线“/”识别），则它将求反值复制到相应的布尔变量内。若一个接点是求反的，则它仅当相应的布尔值为 FALSE 时才接通。

## 7、函数及功能块调用

与接点和线圈一起，你也可以插入功能块和程序。在网络中，它们必须有带布尔值的一个输入和一个输出，并可在相同位置上像接点那样使用，也就是说在 LD 网络的左侧。

类型	图形符号	说明
插入运算块		插入函数或功能块，根据弹出的对话框通过鼠标选择想要使用的函数及功能块。适用于对函数和功能块不太熟悉者使用。
插入空运算块		直接插入矩形块，在“???”处直接输入函数或功能块名，适用于对函数及功能块较为熟悉的用户。

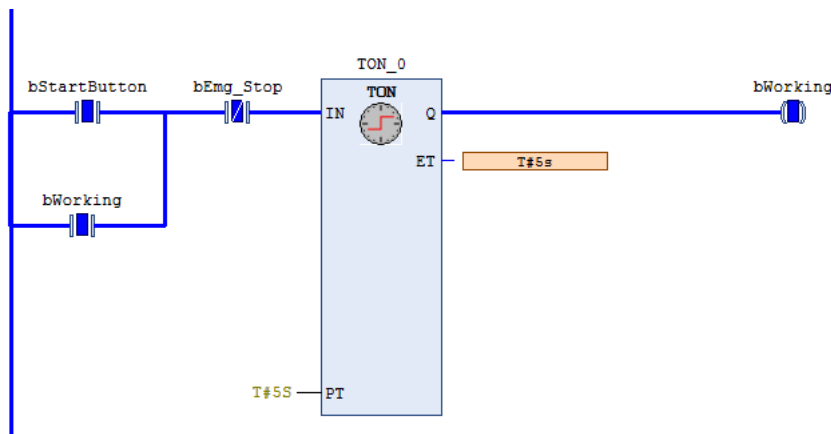
类型	图形符号	说明
插入带 EN/ENO 的运算块		只有当 EN 为 True 时，才执行函数或功能块并允许将状态传递至下游。适用于对函数和功能块不太熟悉者使用。
插入带 EN/ENO 的空运算块		插入带 EN/ENO 矩形块，在“???”处直接输入函数或功能块名，只有当 EN 为 True 时，才执行函数或功能块并允许将状态传递至下游。适用于对函数及功能块较为熟悉的用户。

梯形图编程语言支持函数和功能块的调用。在函数和功能块调用时，需要注意如下事项：

- 1) 梯形图中，函数和功能块用一个矩形框表示。函数可以有多个输入参数但只能有一个返回参数。功能块可以有多个输入参数和多个输出参数。
- 2) 输入列于矩形框的左侧，输出列于矩形框的右侧。
- 3) 函数和功能块的名称显示在框内的上中部，功能块需要将其实例化，实例名列于框外的上中部。用功能块的实例名作为其在项目中的唯一标识。
- 4) 为了保证能流可以通过函数或功能块，每个被调用的函数或功能块至少应有一个输入和输出参数。为了使被连接的功能块执行，至少应有一个布尔输入经水平梯级连接到垂直的左电源轨线。
- 5) 功能块调用时，可以直接将实际参数值填写在该内部形参变量名的功能块外部连接线处。

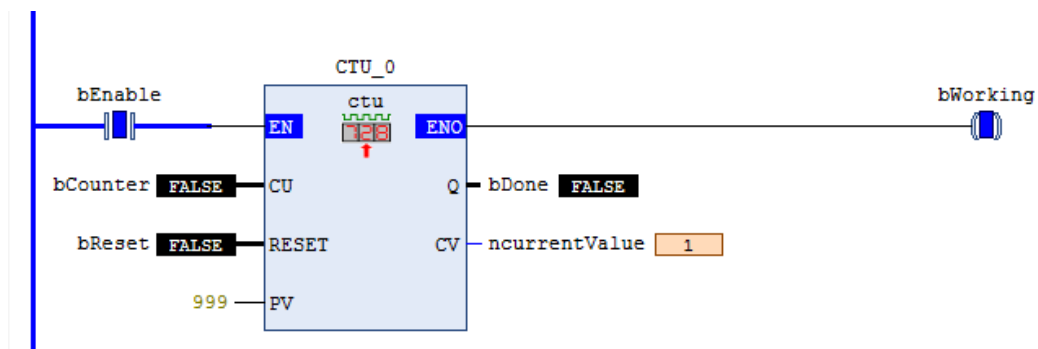
**例：**功能块调用实参的设置。

调用 TON 延时 ON 功能块，TON\_1 为功能块 TON 实例化后的实例名。功能块的输入形参 PT 设置为 t#5s。输出形参 Q 及 ET，当不需要输出形参如示例中的 ET 时可以不连接变量。



可以看到，功能块 TON 的输出 Q 连接到了线圈 bWorking。表示当触点 bStartButton 为 True 且 bEmg\_Stop 为 False 持续时间超过 5s 后，bWorking 为 True。当 bEmg\_Stop 断开即为 True 时，bWorking 为 False。

- 6) 如果没有 EN 和 ENO 的专用输入输出参数，则函数和功能块会被自动执行，且将状态传递至下游。



可以看出，当 bCounter 有上升沿触发信号时，则形参输出变量 CV 进行加 1 计算。

- ◆ 当 EN 为 False 时，功能块本体定义的操作不被执行，ENO 的值也相应为 False。
- ◆ 当 ENO 的值为 True，则说明该功能块的正在被执行。



## 7. 特殊功能

本章主要介绍外部中断、高速计数、PLC SHELL 等功能的应用方法。

---

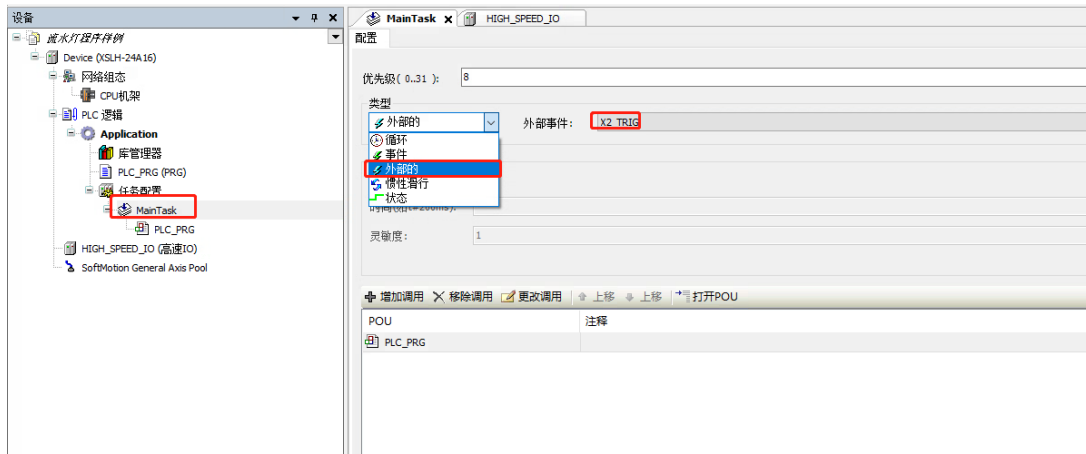
7. 特殊功能	139
7-1. 外部中断	140
7-1-1. 固件 1.1.0 版本以下外部中断应用举例	140
7-1-2. XS 系列固件 1.1.0 版本应用举例	140
7-2. 高速计数	142
7-3. 高速 IO 配置	143
7-4. 系统设置	147
7-5. PLC 指令	148
7-5-1. 应用举例	148
7-6. 时钟	155
7-6-1. 功能概述	155
7-6-2. 应用举例	155

## 7-1. 外部中断

### 7-1-1. 固件 1.1.0 版本以下外部中断应用举例

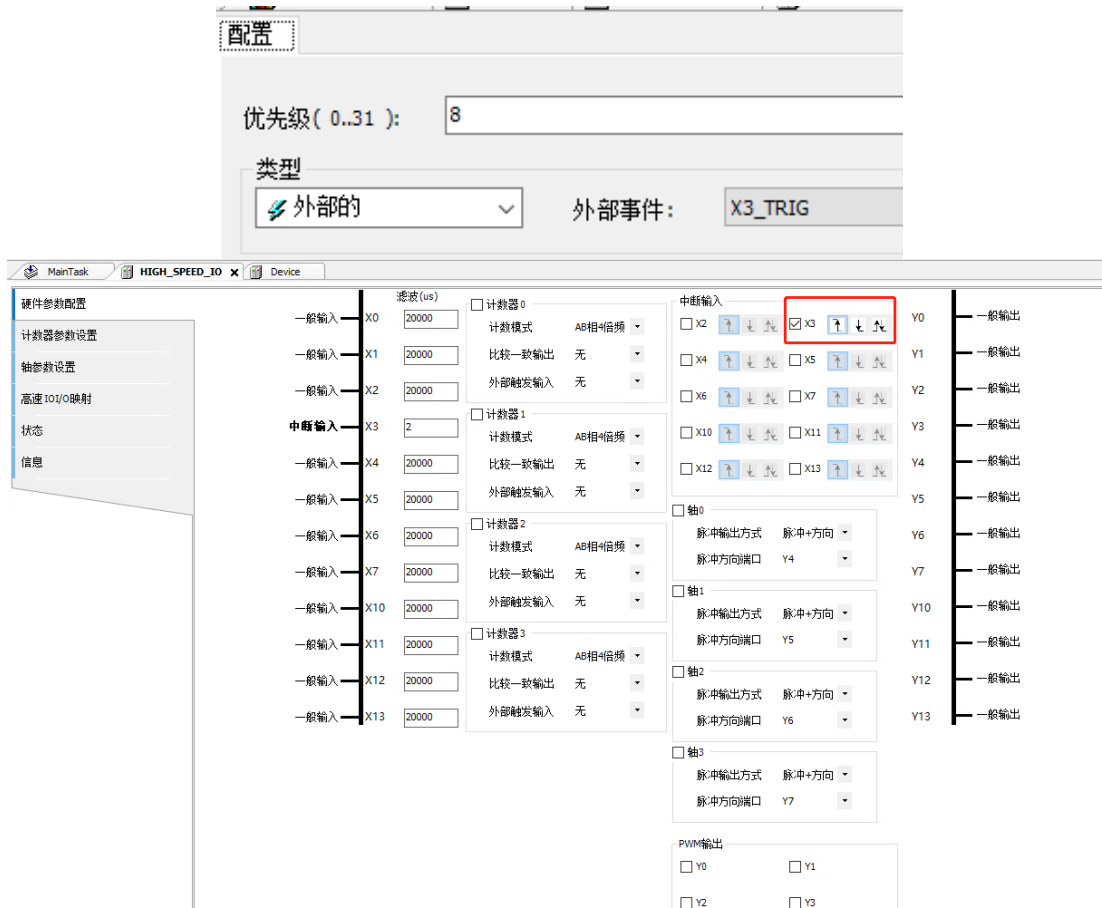
XS 系列 PLC 支持 X 端子中断，同一个端子支持上升沿和下降沿中断，在 XS Studio 中通过任务类型中的外部事件形式来使用中断。如 X2R\_TRIG 代表 X2 上升沿中断，X2F\_TRIG 代表下降沿中断，各机型支持的中断数量和类型见外部事件“外部的”选项。

双击“MainTask”，在弹出界面中设置为外部事件“外部的”-外部中断使用端子 X，也可设置外部中断事件的优先级。



### 7-1-2. XS 系列固件 1.1.0 版本应用举例

需要使用【XJ\_Interrupt】和【XJ\_WriteInterruptParameter】指令和界面（功能详见指令篇手册）。设置 X3 为外部中断输入，取其双边沿信号，可以在硬件参数配置界面配置，也可以使用 XJ\_WriteInterruptParameter 指令配置，给一次 X3 的边沿信号，就执行一次另一个 task（配置为外部的、X3\_TRIG）下的 POU 程序里的自加 1 指令，参数配置和指令如下图：



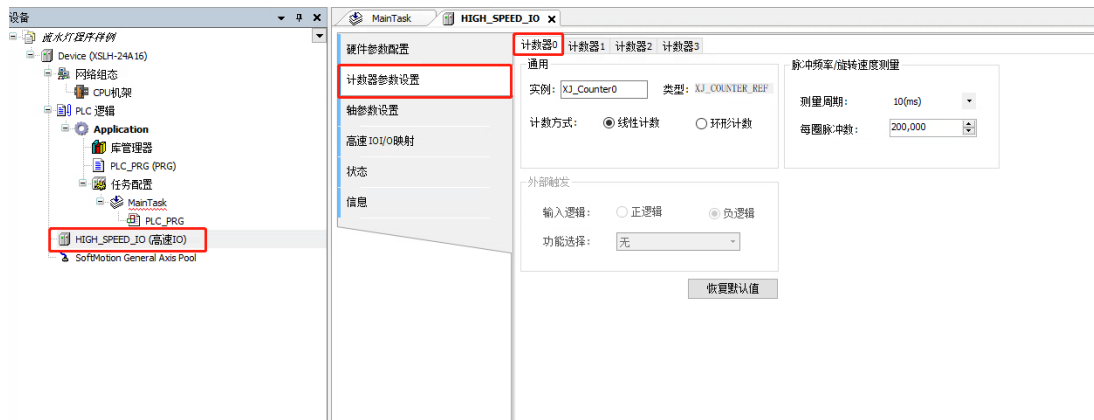
Device.Application.PLC_PRG						
表达式	类型	值	准备值	地址	注释	
[-] XJ_EnableInterrupt_0	XJ_EnableInterrupt	TRUE				
[-] xEnable	BOOL	8			使能	
[-] udiExternal	UINT	0			打开外部输入中断，例如	
[-] uiCompare	UINT				打开比较一致中断，例如	
[-] xValid	BOOL	TRUE			中断生效	
[-] xBusy	BOOL	FALSE			正在运行	
[-] xError	BOOL	FALSE			错误标志	
[-] eErrorID	HSIO_ERROR	ERR_OK			错误代码	
[-] XJ_WriteInterruptParameter_0	XJ_WriteInterruptParameter					
[-] Port	UINT	8			端口号，例如（外部输入中	
[-] xExecute	BOOL	TRUE			触发	
[-] byValue	BYTE	2			值(0: 上升沿, 1 为下降沿,	
[-] xDone	BOOL	TRUE			完成标志	
[-] xBusy	BOOL	FALSE			正在运行	
[-] xError	BOOL	FALSE			错误标志	
[-] eErrorID	HSIO_ERROR	ERR_OK			错误标志	

## 7-2. 高速计数

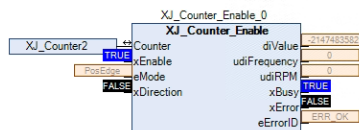
### ■ 固件 1.1.0 版本应用举例

**注意：**固件 1.1.0 版本以下机型不支持高速 IO 界面，高速计数指令详见指令篇

例一：使用【XJ\_Counter\_Enable】指令，测量外部的高速信号输入，配置如下图



表达式	类型	值	准值	地址	注释
XJ_Counter_Enable_0	XJ_Counter_Enable				
Counter	REFERENCE TO XJ_COUNTER...				数据类型XJ_COUNTER_REF
xEnable	BOOL	TRUE			使能
eMode	HSC_EDGE_MODE	PosEdge			PosEdge: 上升沿计数; NegEdge: 下降沿计数; BothEdge: 双边沿计数
xDirection	BOOL	FALSE			false=加计数; true: 减计数;
diValue	DINT	-2147483582			高速计数值
udiFrequency	UDINT	0			脉冲频率测量值 (单位: Hz), 若为低频可通过界面测量周期配合使用
udiRPM	UDINT	0			每分钟旋转速度 (单位: r/min), 与界面配置中的每圈脉冲数配合使用
xBusy	BOOL	TRUE			忙碌中
xError	BOOL	FALSE			错误标志
eErrorID	HSIO_ERROR	ERR_OK			错误代码



### 7-3. 高速 IO 配置

双击设备树中的 HIGH-SPEED-IO(高速 IO)选项,可打开高速 IO 的硬件参数配置界面。在该界面可对高速脉冲输出功能及 PWM 输出功能进行配置(XS Studio 1.1.0 版本仅 XSLH-24A16、XSLH-24A8 支持)。默认参数配置界面如下图所示:



#### 1、高速脉冲输出功能

##### ■ 脉冲指令

指令名	功能说明
MC_Power	使轴开始可运行状态
MC_Reset	复位轴内部相关错误
MC_Jog	点动
MC_Stop	停止控制器运动
MC_MoveAbsolute	实现一个控制轴达到指定绝对运动位置
MC_MoveRelative	从当前轴的位置将轴移动一个相对位置
MC_MoveVelocity	轴以一个指定速度持续运行下去
MC_SetPosition	设置轴的当前位置
MC_ReadStatus	读取轴状态
MC_ReadSetPosition	读取当前轴的设置位置
MC_ReadActualPosition	读取当前相关轴的当前位置

以上指令均为 SM3\_Basic 库内的指令。具体指令说明见《XS 系列 PLCopen 标准控制器用户手册【指令篇】》。

##### ■ 配置高速脉冲输出功能

- ◆ 主要包括脉冲输出方式及脉冲方向端口(以轴 0 配置为例,轴号支持 0-3)。
- ◆ 勾选使用的轴 0; 勾选后配置情况如图所示:

◆ 配置高速脉冲输出的工作方式：

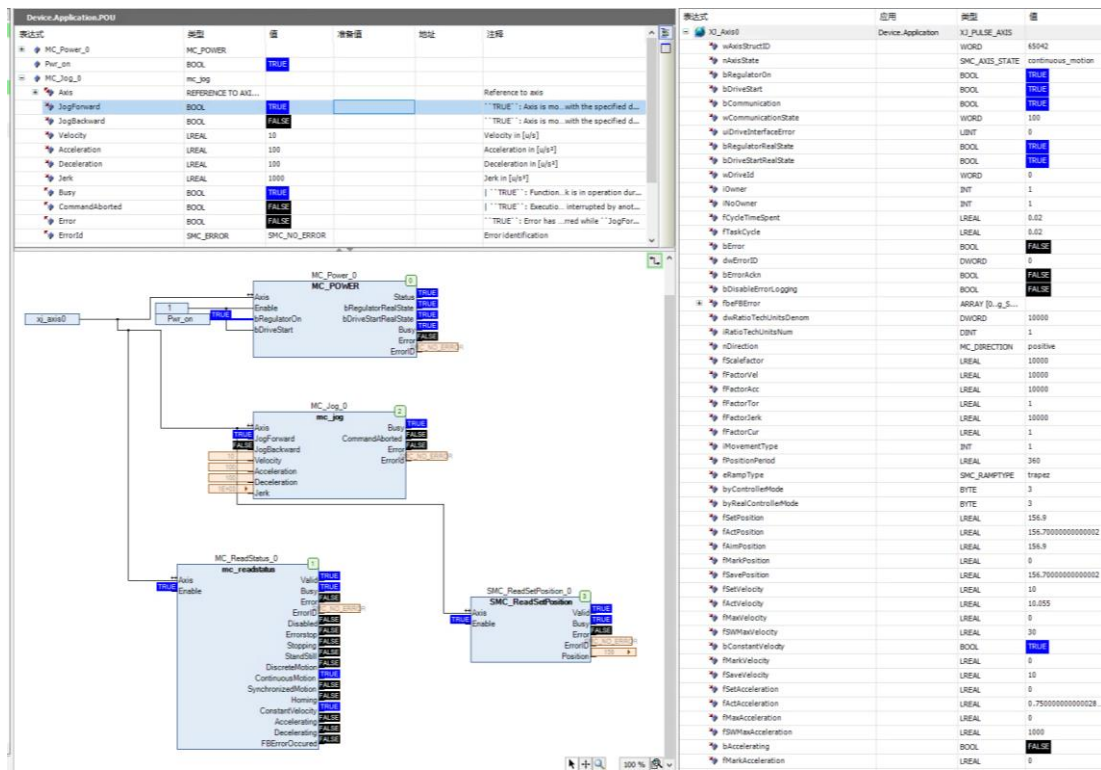
脉冲指令形态	脉冲+方向	
	正转	反转
正方向	PULSE SIGN	PULSE SIGN
负方向	PULSE SIGN	PULSE SIGN

### ■ 脉冲轴参数配置

在轴参数设置界面可对已配置的轴进行实例化等操作。以轴 0 为例，配置界面如下图所示：

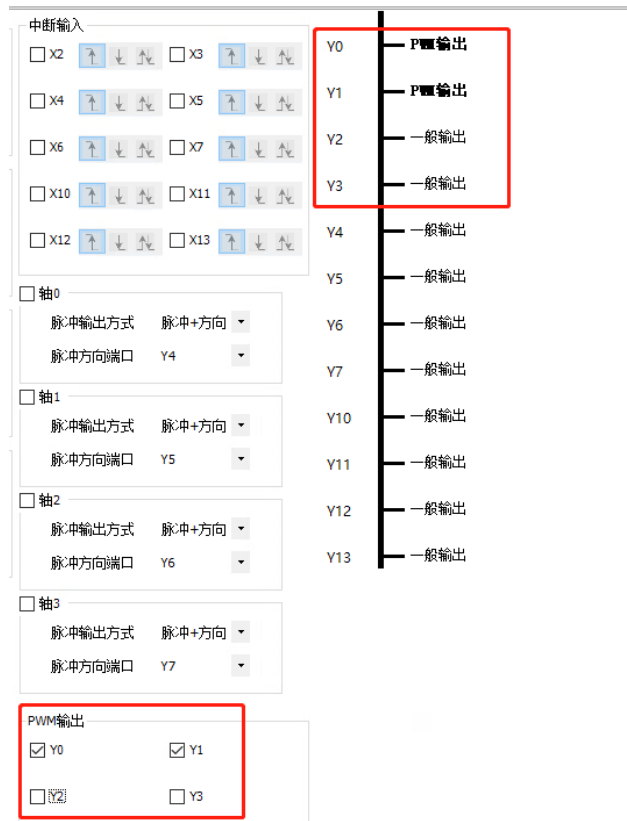
- ◆ 轴 0 实例化名称默认为：XJ\_Axis0；支持用户手动修改。
- 指令使用示例

使用【MC\_POWER】、【MC\_JOG】等指令实现脉冲轴点动、位置以及轴状态获取功能，配置如下图：



## 2、PWM 输出功能

在硬件参数配置界面可对 PWM 输出进行配置。结合【XJ\_PWM】指令使用时需先在硬件参数界面勾选对应的端口。配置如图所示：



可结合脉宽调制【XJ\_PWM】指令使用，以 Y0 端子为例进行 PWM 脉宽调制输出，配置如下图所示：

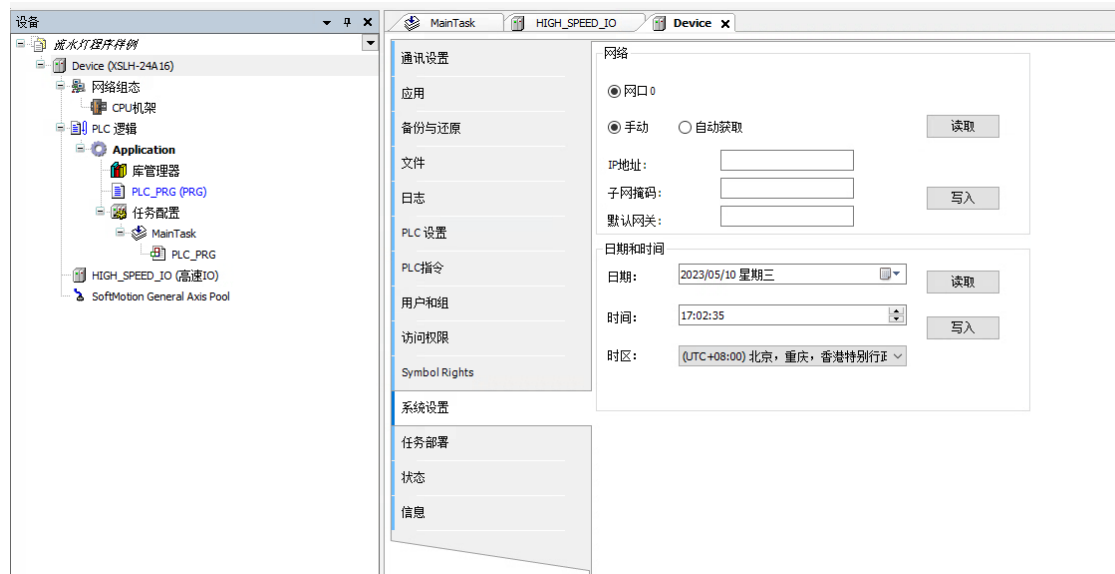
Device.Application.POU						
表达式	类型	值	准备值	地址	注释	
XJ_PWM_0	XJ_PWM					
xEnable	BOOL	TRUE			TRUE: 输出PWM波形, FALSE: 停止输出	
ePort	Y_PORT	Y0			输出端口	
byDuty	WORD	32767			占空比, 范围为 1~65535	
udiFrequency	UDINT	100			输出频率, 单位为0.1Hz, 范围为1~200KHz	
xValid	BOOL	TRUE			是否有效	
xBusy	BOOL	TRUE			是否正在执行	
xError	BOOL	FALSE			错误标记	
eErrorID	HSTIO_ERROR	ERR_OK			错误代码	
cycleUs	REAL	100000				



## 7-4. 系统设置

### ■ 固件 1.1.0 版本系统设置应用举例

在应用的工程中, 双击设备“Device”, 找到“系统设置”, 在系统设置的界面中可读取/设置网口 IP, 系统时间。



- 注意:**
- ① 读取 IP--若该网口没有插网线, 则无法获取该网口的全部 IP 信息;
  - ② 写入/读取日期和时间--同时读取/写入日期、时间、时区信息。

## 7-5. PLC 指令

**注意：**该功能仅支持 XSDH、XSLH、XS3 系列

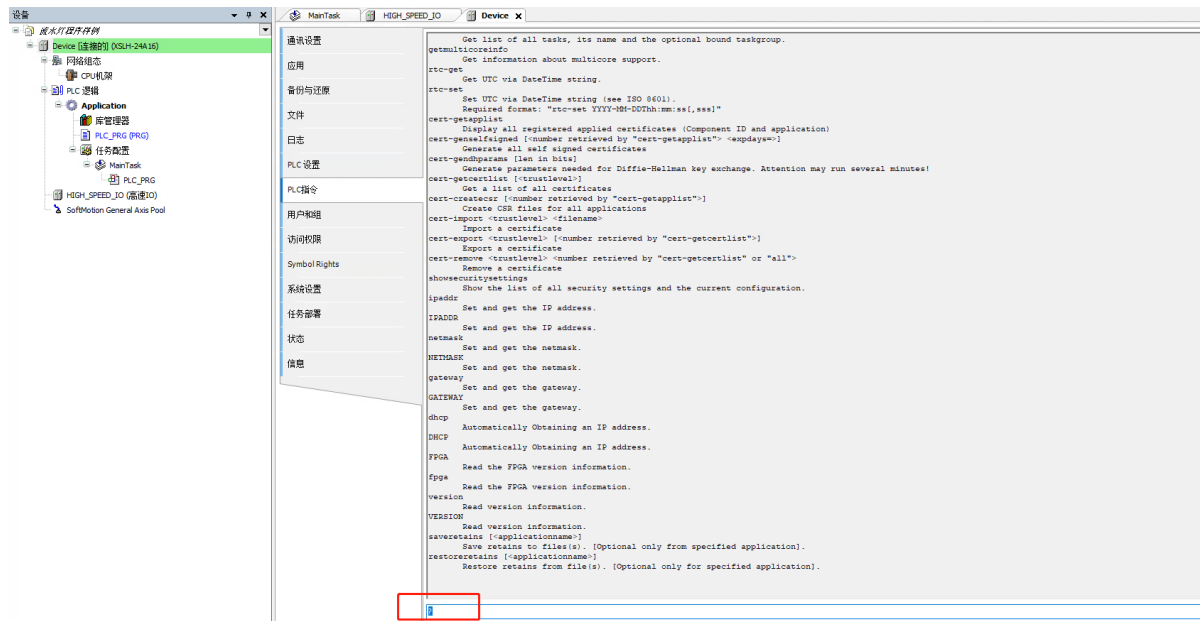
PLC 指令功能是一个基于文本的控制监视器，可以用于查询控制器的特定信息，在输入窗口中输入指定命令，结果窗口中接收来自控制器的响应。

### ■ 指令列表

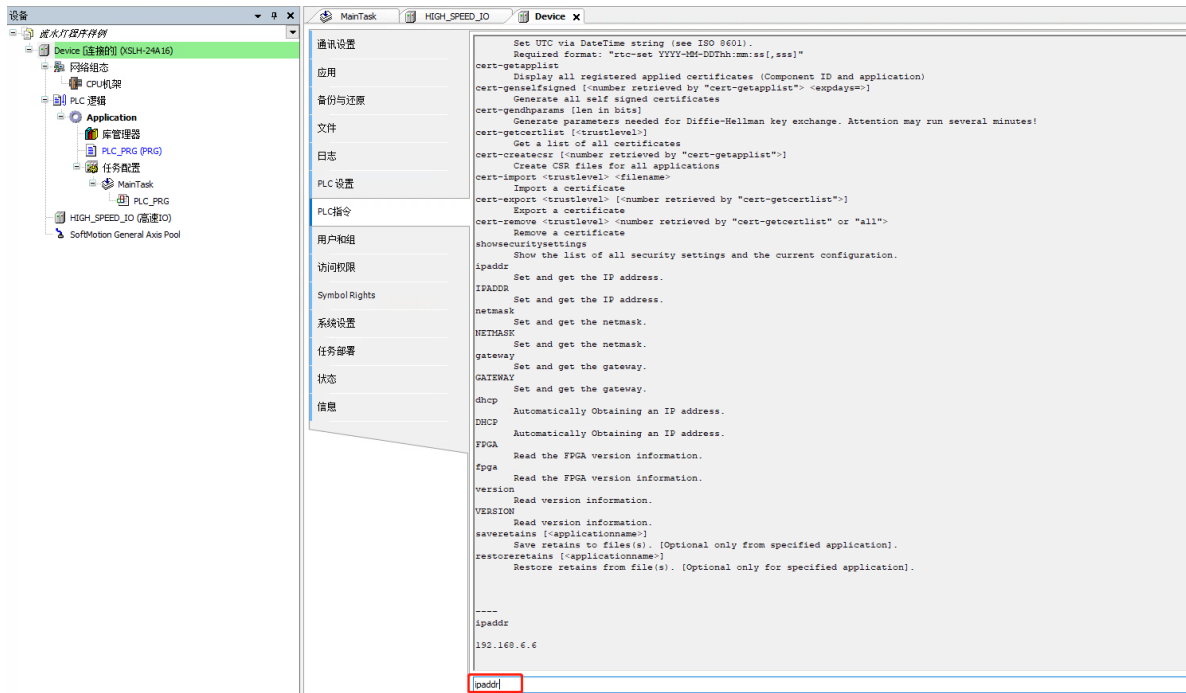
指令名称	功能介绍
ipaddr / IPADDR	获取/设置 PLC 的 IP 地址
netmask / NETMASK	获取/设置 PLC 的子网掩码
gateway / GATEWAY	获取/设置 PLC 的网关
dhcp / DHCP	设置 IP 为自动获取
fpga / FPGA	获取 PLC 的 fpga 版本
version / VERSION	获取 PLC 的固件版本
rtc-get / RTC-GET	获取当前 UTC 时间
rtc-set / RTC-SET	设置 UTC 时间

### 7-5-1. 应用举例

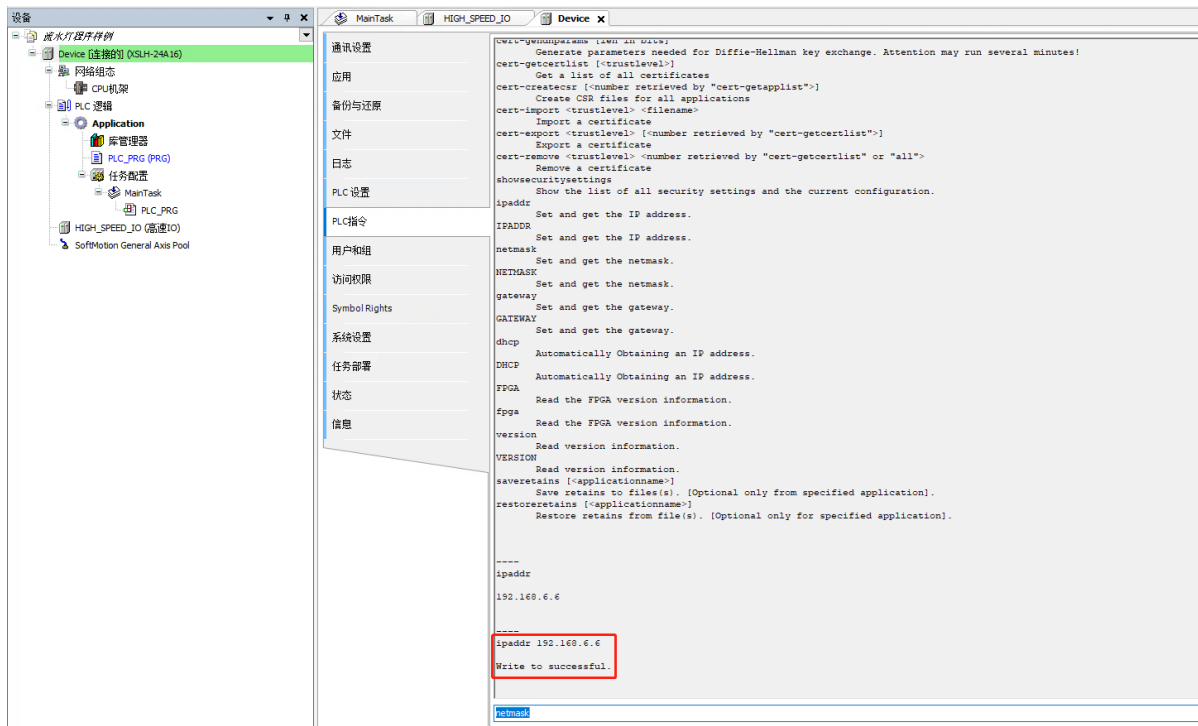
双击“Device”，在“PLC 指令”里输入“？”，可以出来全部功能。可以在这里修改 IP，获取固件版本，设置/读取时钟信息等等。



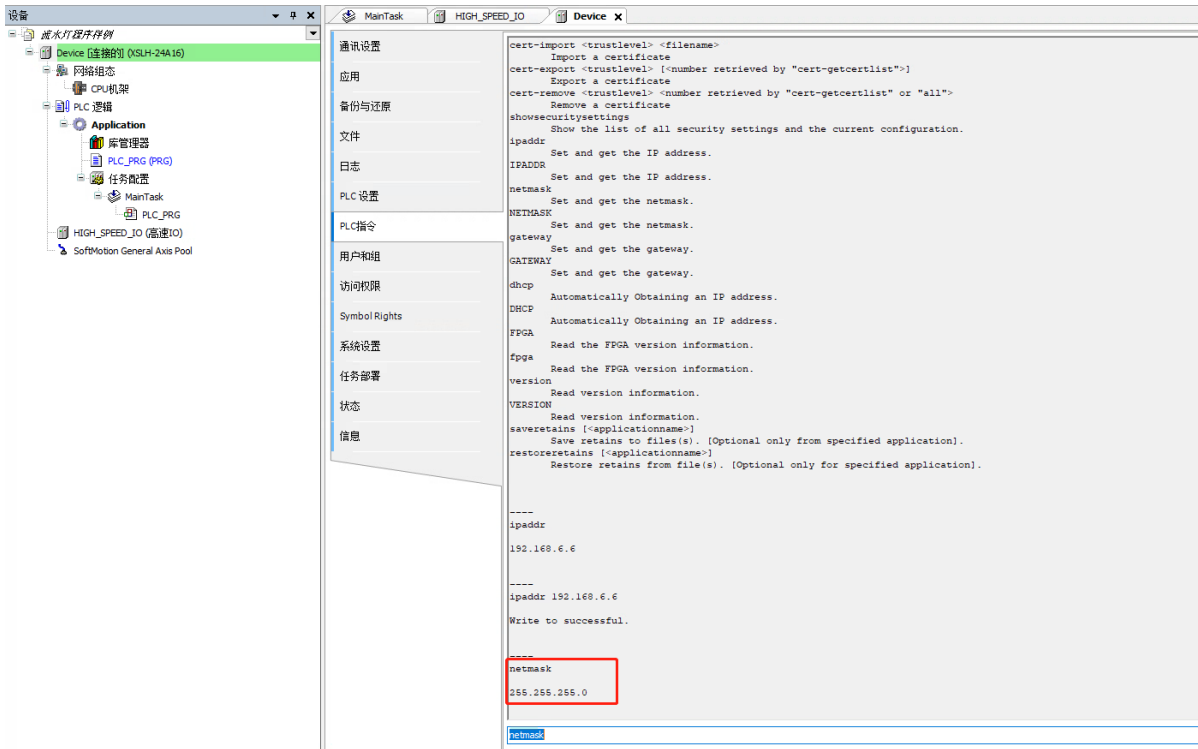
比如：输入“ipaddr”就可以获取到 PLC 的当前的 IP 地址。



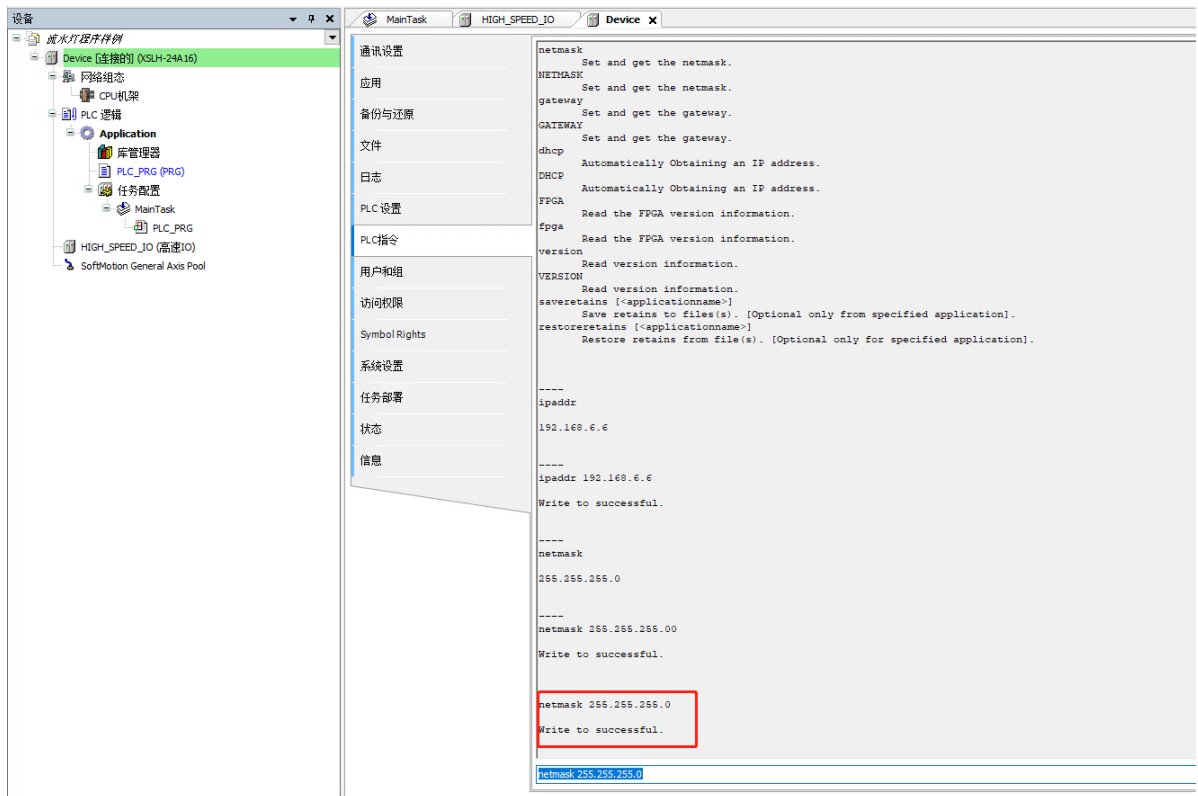
输入“ipaddr 192.168.6.10”，设置 PLC 的 IP 地址，显示“Write to successful”则写入成功，重新上电生效。



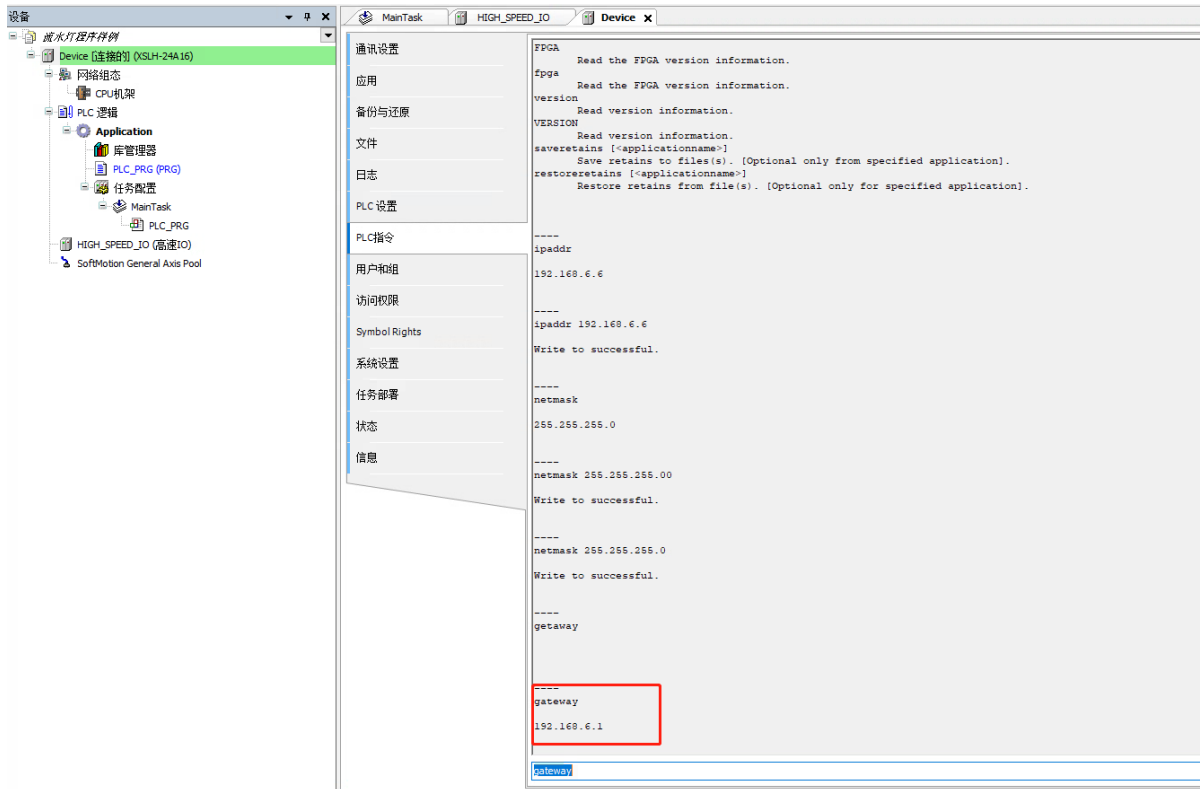
输入“netmask”，就可以获取到 PLC 的当前的子网掩码。



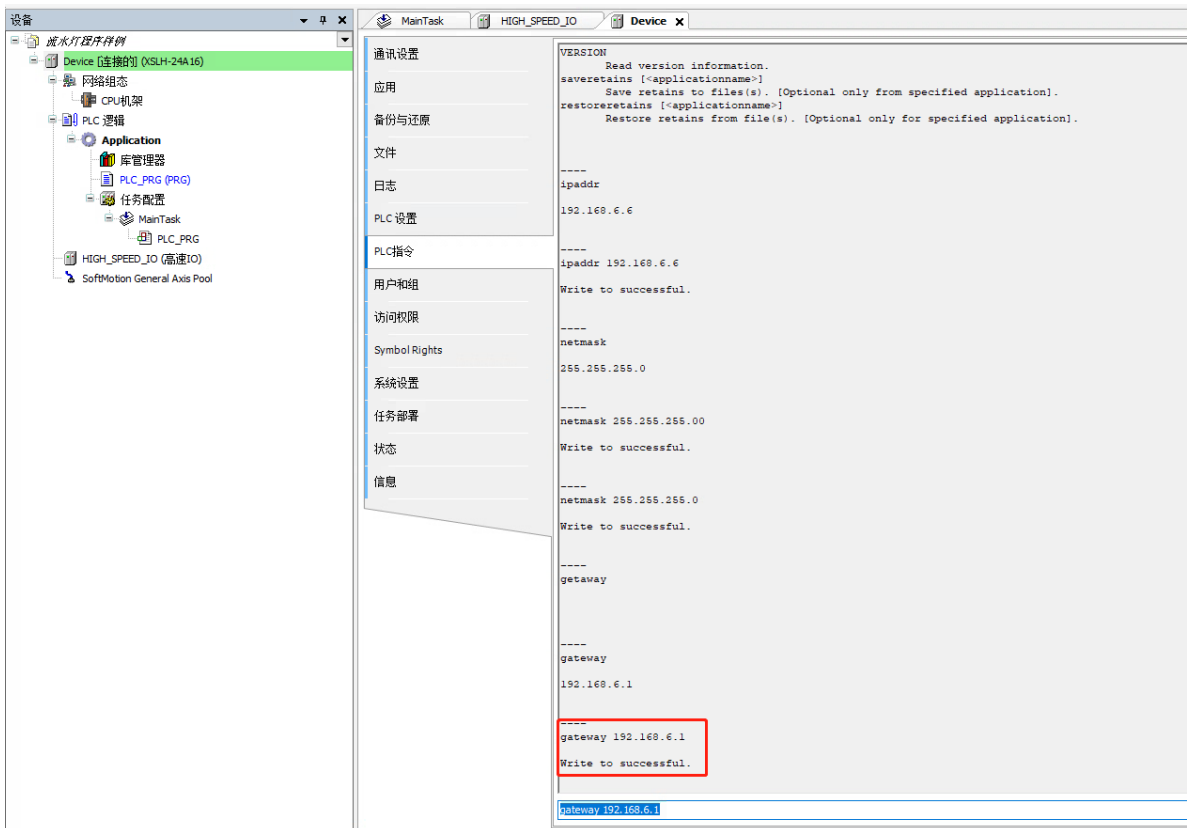
输入“netmask 255.255.254.0”，设置 PLC 的子网掩码，显示“Write to successful”则写入成功。



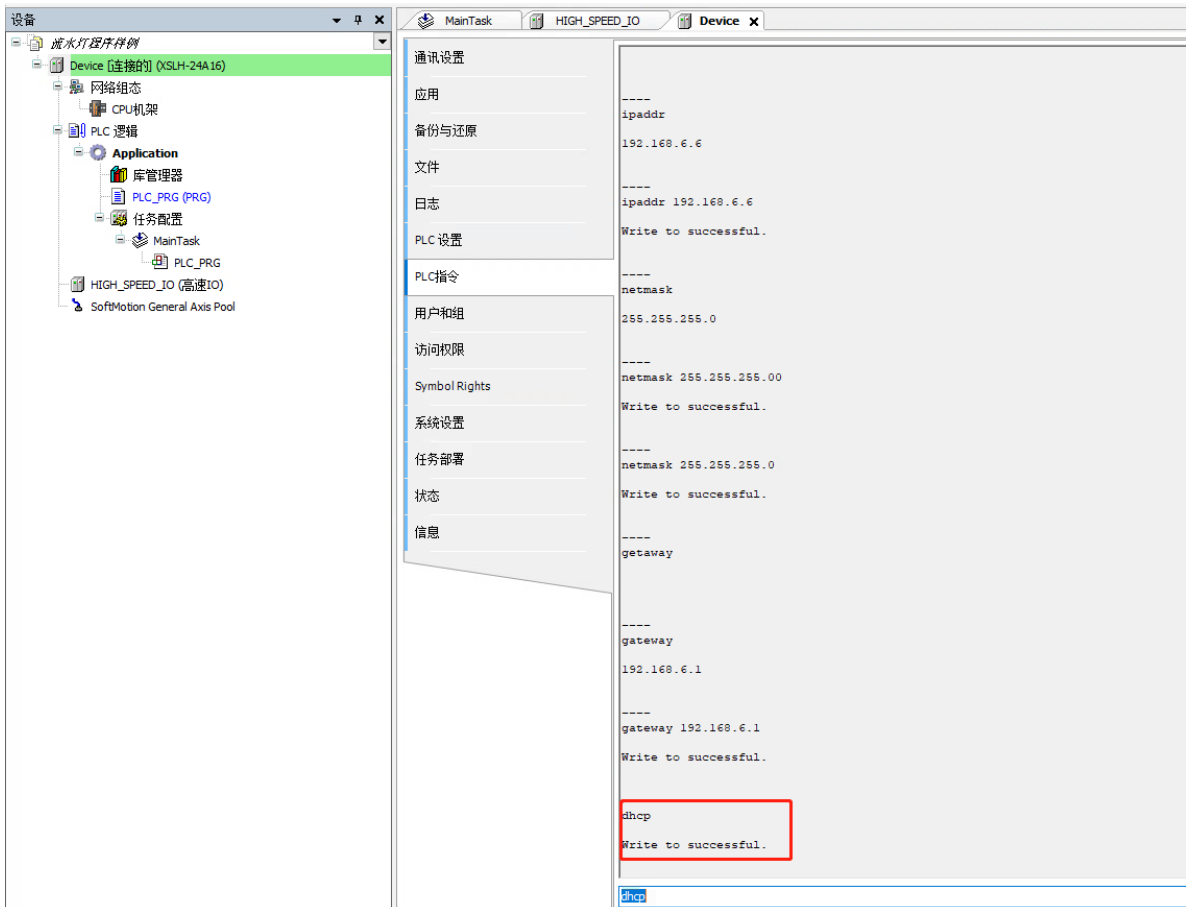
输入“gateway”，就可以获取到 PLC 的当前的默认网关。



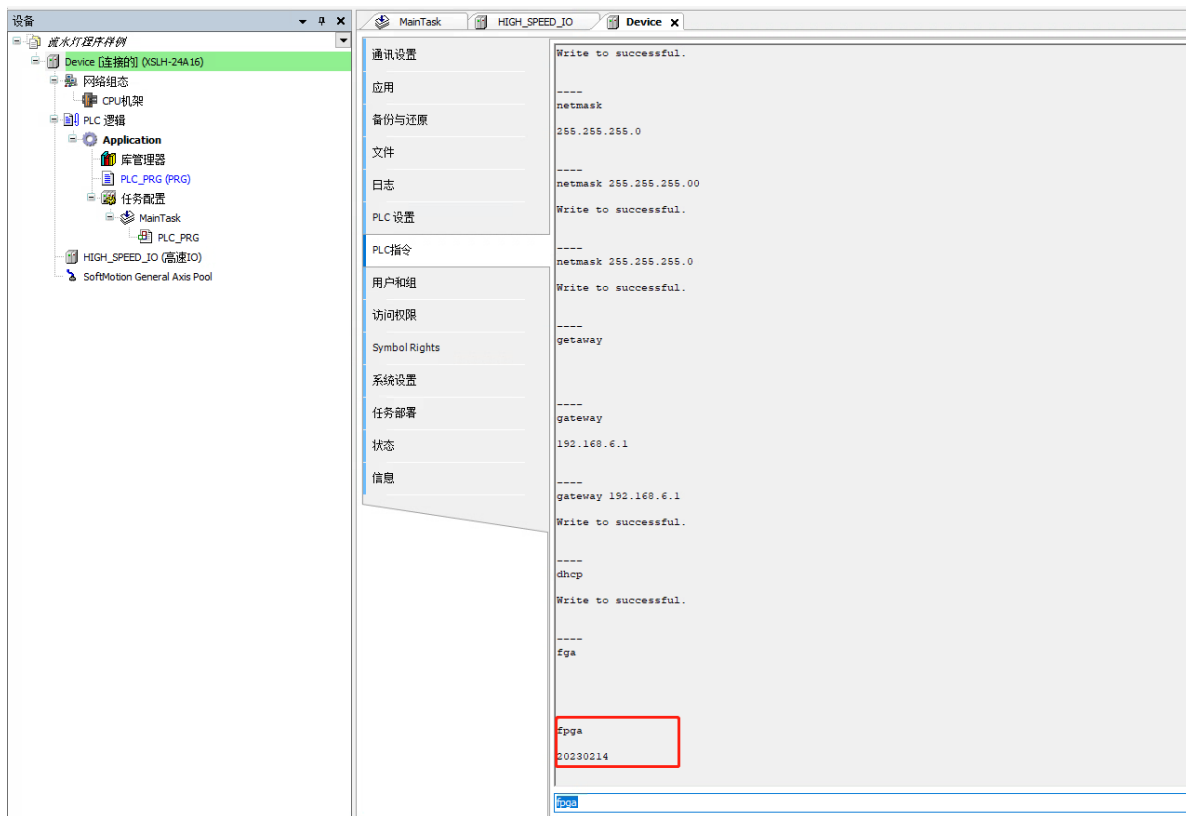
输入“gateway 192.168.6.1”，设置 PLC 的网关，显示“Write to successful”则写入成功。



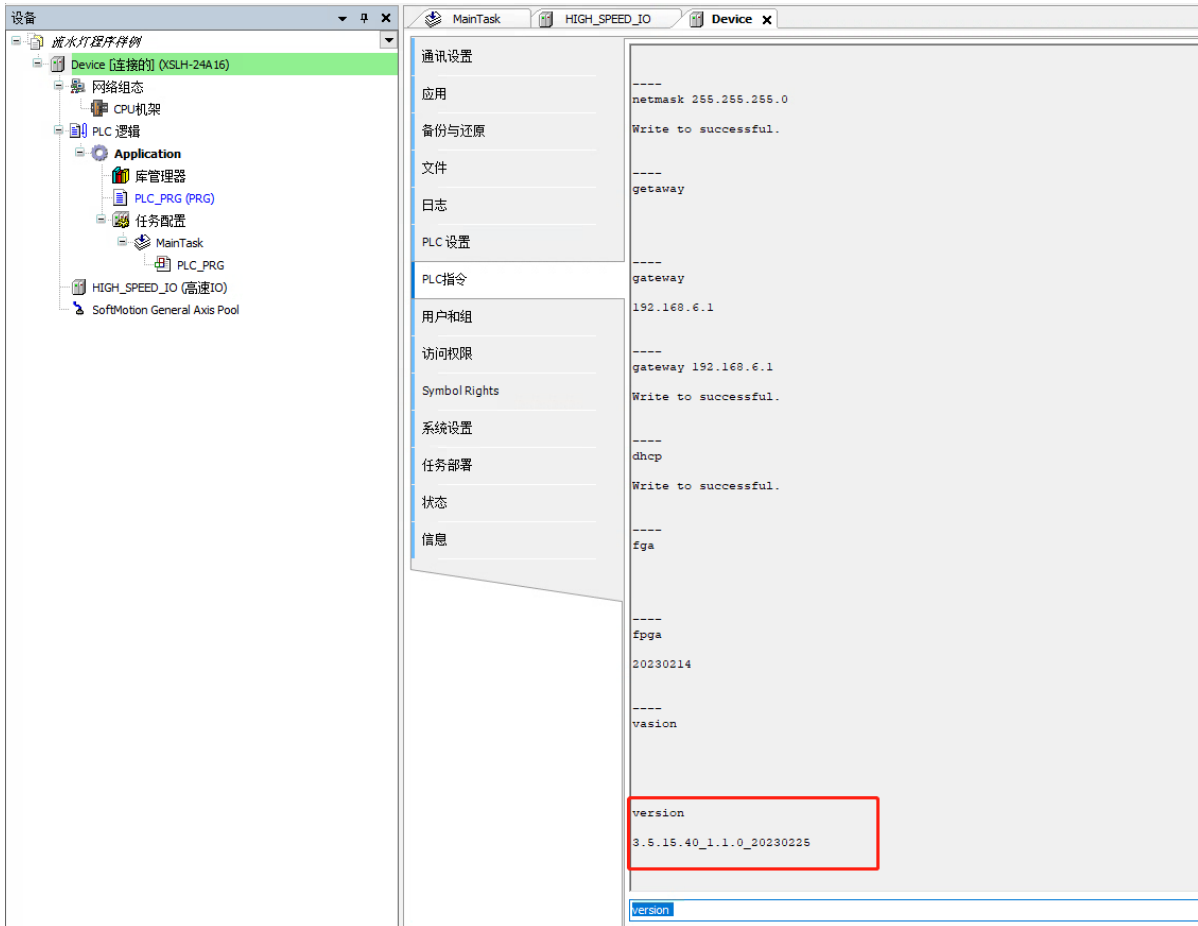
输入“dhcp”，设置 PLC 的 IP 获取方式为自动获取，显示“Write to successful”则写入成功。当 IP 的获取方式为自动获取时，需要保证网络环境良好。



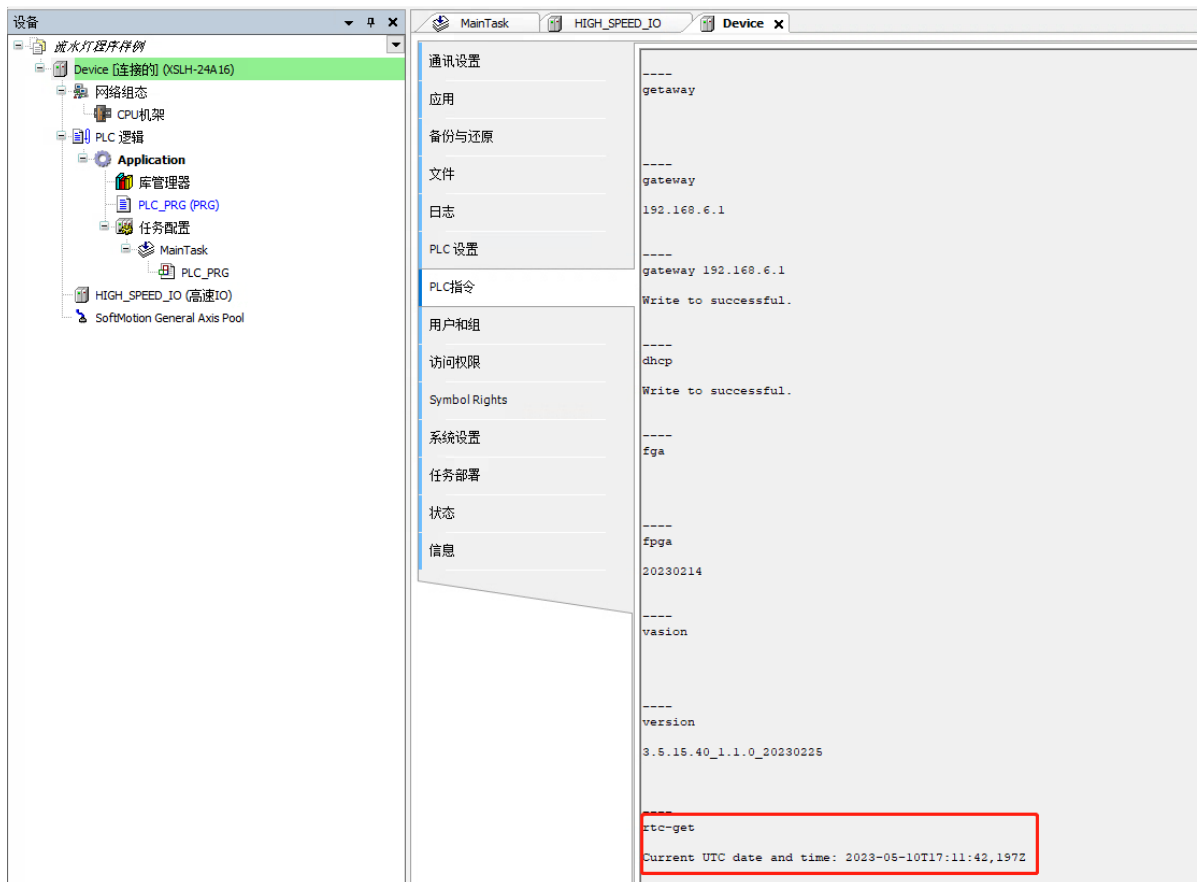
输入“fpga”，就可以获取到 PLC 当前的 FPGA 版本。



输入“version”，就可以获取到 PLC 当前的固件版本。

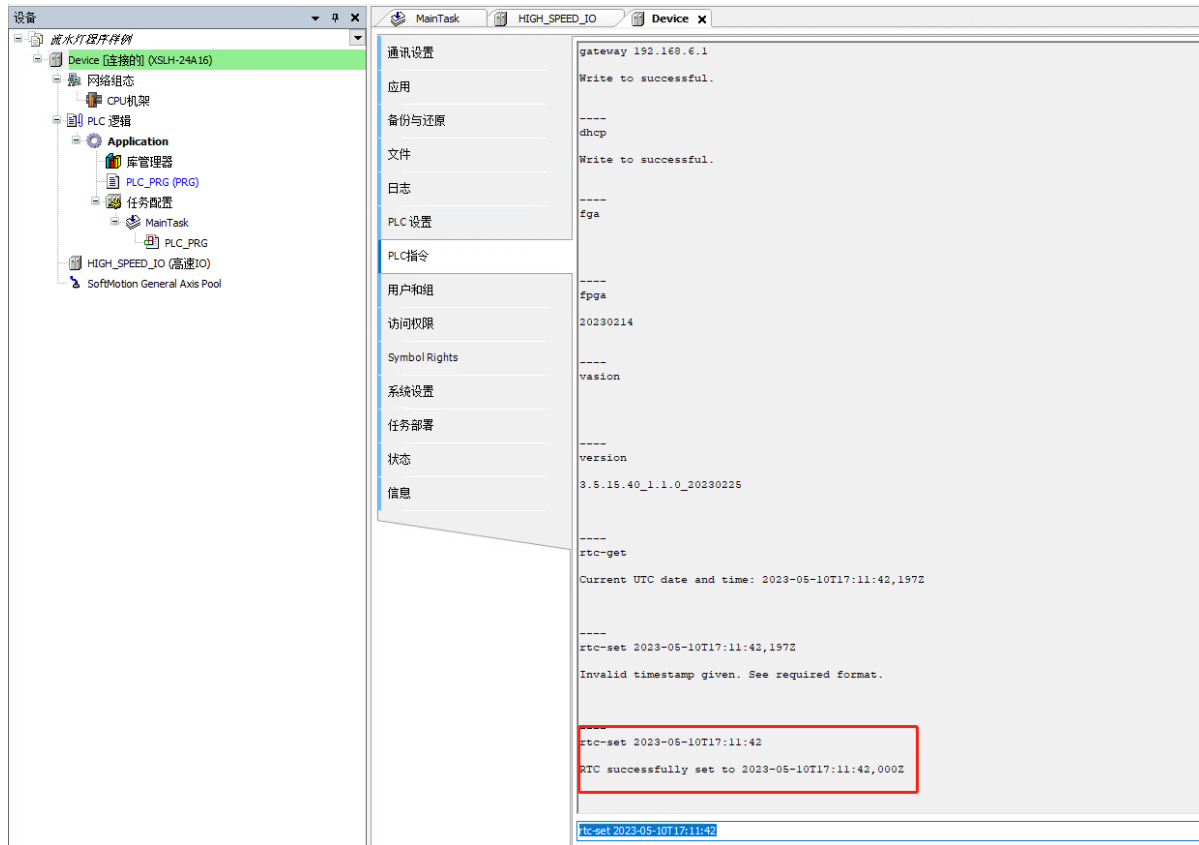


输入“rtc-get”，就可以获取当前的 UTC 时间。



输入“rtc-set 2021-10-25T18:24:30”，可设置 UTC 时间。显示“RTC successfully set to

2021-10-25T18:24:30,000Z” 则写入成功。其中“000Z”显示不定。





## 7-6. 时钟

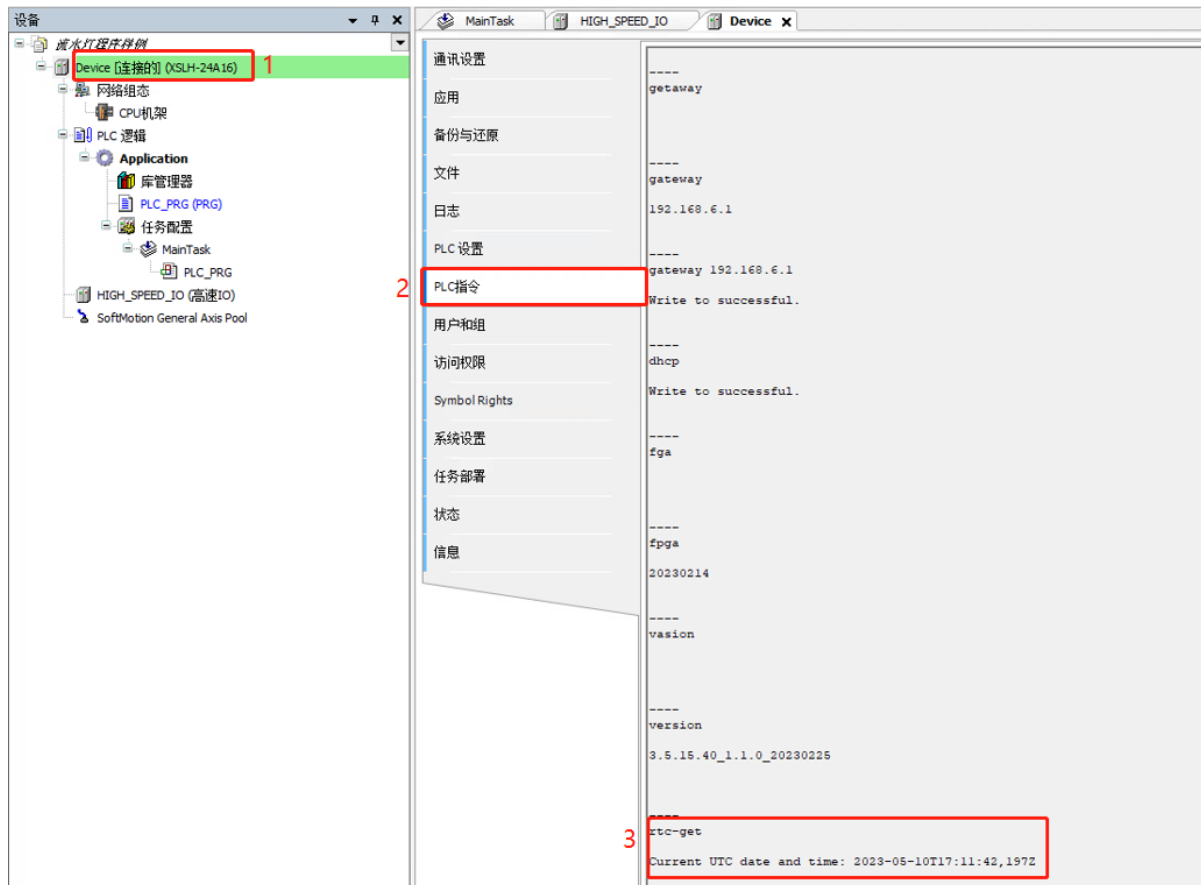
### 7-6-1. 功能概述

XS 系列 PLC 集成了 RTC，用于记录当前的系统时间，该时钟由电池供电，可保证时间的精确性，同时也支持用户手动修改 RTC 时间。

### 7-6-2. 应用举例

获取事件的方式：

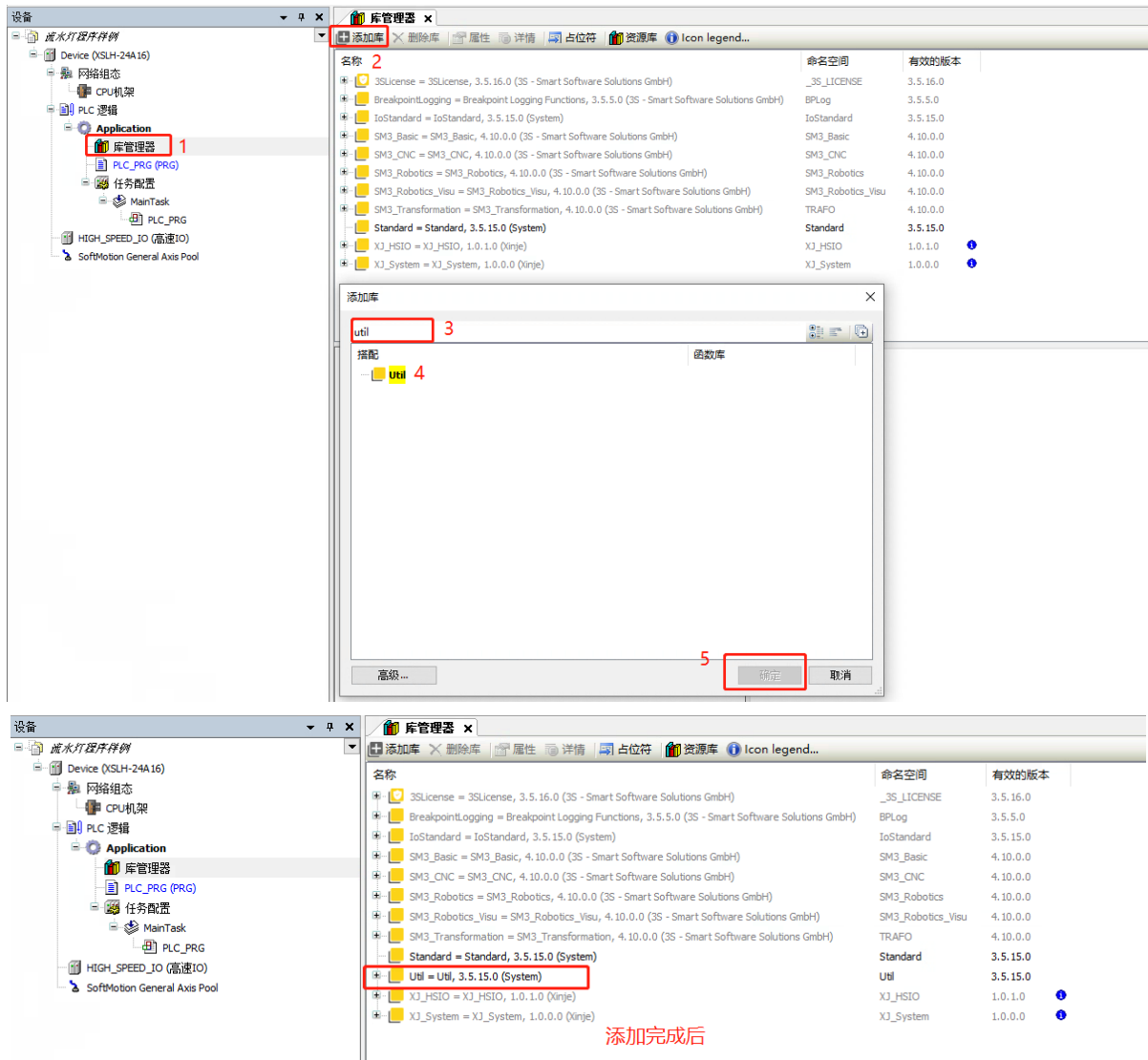
- 1、双击“Device”，在“PLC 指令”里输入“rtc-get”，获取当前时间。



- 2、通过时钟指令

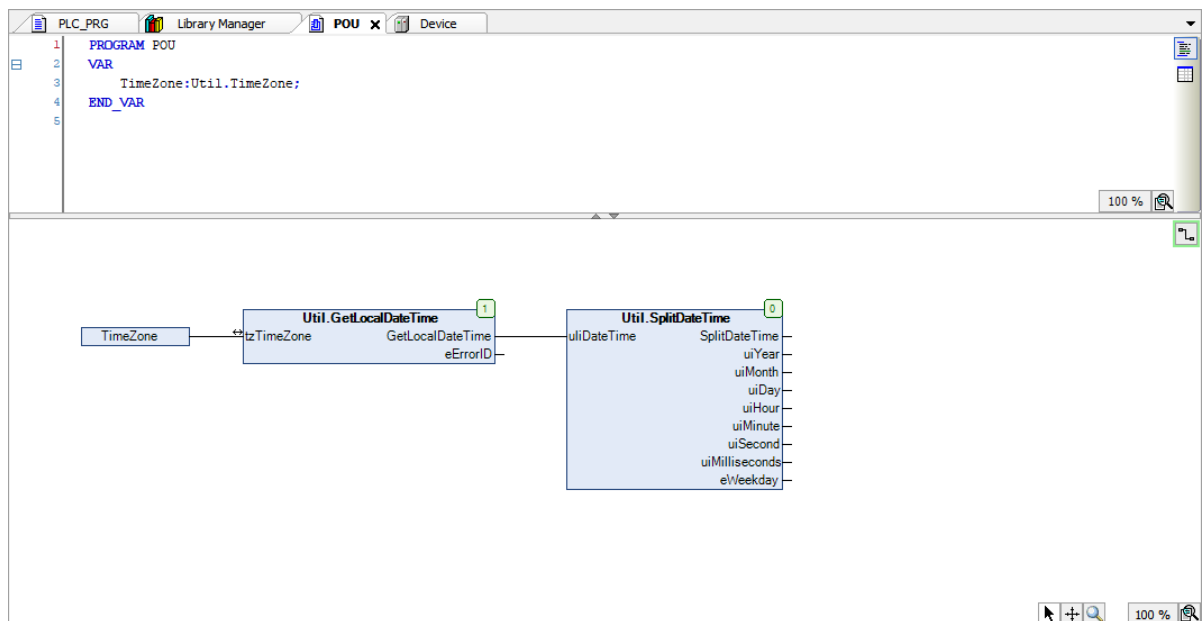
(1) 添加相应的库文件

在“库管理器”里添加“Util”，添加完成可使用时钟功能。



(2) 编写时钟程序

可使用功能块“Util.GetLocalDateTime”、“Util.SplitDateTime”获取当前时间。在此库中还有其它关于时钟的功能块，可在库“Util”中查看。



## 8. 附录：常见问题及解决方法

本章主要介绍 XS Studio 编程中遇到的常见问题及解决方法。

---

8. 附录：常见问题及解决方法	157
8-1. Package	158
8-1-1. Package 命名规则	158
8-1-2. Package 的获取	158
8-1-3. Package 的安装	158
8-2. XS 系列 PLC 固件升级	159
8-2-1. 固件命名规则	159
8-2-2. 固件获取	159
8-2-3. 固件安装及其注意事项	159
8-3. XS 系列的本地扩展模块	160
8-4. XS 系列的远程扩展模块	162
8-5. 拨码	164
8-6. 用户新安装 XS Studio 上位机，打开后进行编译，会出现很多报错的原因	164
8-7. 网关显示红点的原因	164
8-8. 添加多个 EtherCAT 从站后，有 warning 提示的原因	164
8-9. EtherCAT 轴一运行会出现通讯断掉的原因	164
8-10. 用户怎么取消密码登录	164
8-11. 为什么无法连接到 PLC 设备	165
8-12. IP 地址修改不成功问题	165
8-13. 提示“没有可用于此对象的源码，是否浏览原始库以显示源代码”问题	166
8-14. setposition 清完位置之后断电上电绝对值伺服位置发生变化的问题	166
8-15. PLC 死机	166
8-16. 在线下载程序丢失	166
8-17. 不同电脑在同一局域网有时会连接到其他设备	166
8-18. 添加隐含检查功能	166
8-19. 实现掉电保持的注意点	169
8-20. 打开工程报错，工程保存必须以存档的方式保存	169
8-21. 设置可以添加行注释和节点注释	170

## 8-1. Package

### 8-1-1. Package 命名规则

命名格式: XSDH-60A32\_3.5.15.40\_1.0.0\_P1\_20211027

①                      ②                      ③                      ④                      ⑤

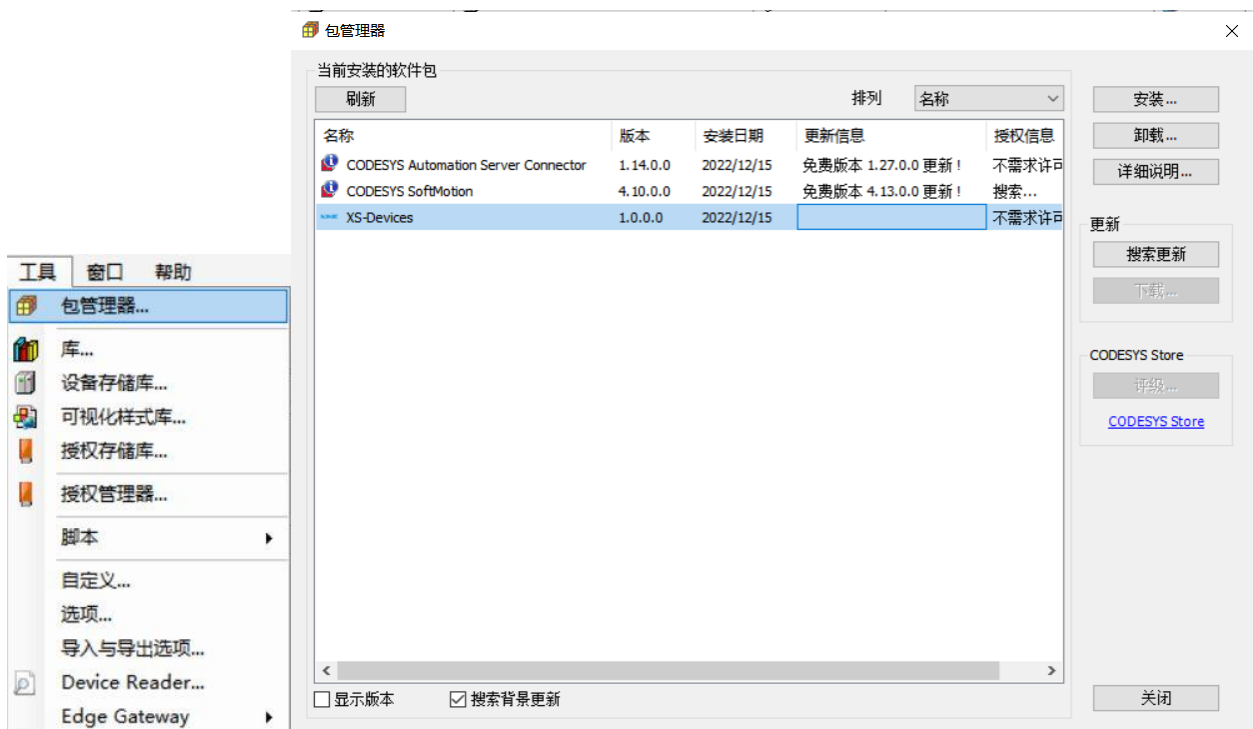
序号	命名	备注
①	XSDH-60A32	PLC 型号
②	3.5.15.40	Runtime 版本
③	1.0.0	Package 投产版本
④	P1	投产后第 1 次在线升级包
⑤	20211027	Package 包升级日期

### 8-1-2. Package 的获取

请在官网获取或联系技术支持，官网网址: [www.xinje.com](http://www.xinje.com) ; 技术服务热线: 400-885-0136。

### 8-1-3. Package 的安装

选择“工具”—“包管理器”，在弹出的界面里安装 Package 包，选择“Install”，找到 Package 所在位置进行安。例如：想要安装 XSLH-24A16 的 Package，最好先将之前的 Package 进行卸载后再安装新的。



## 8-2. XS 系列 PLC 固件升级

### 8-2-1. 固件命名规则

命名格式: XSDH-60A32\_3.5.15.40\_1.0.0\_P1\_20211027

①                    ②                    ③                    ④                    ⑤

序号	命名	备注
①	XSDH-60A32	PLC 型号
②	3.5.15.40	Runtime 版本
③	1.0.0	固件投产版本
④	P1	投产后第 1 次在线固件升级
⑤	20211027	固件升级日期

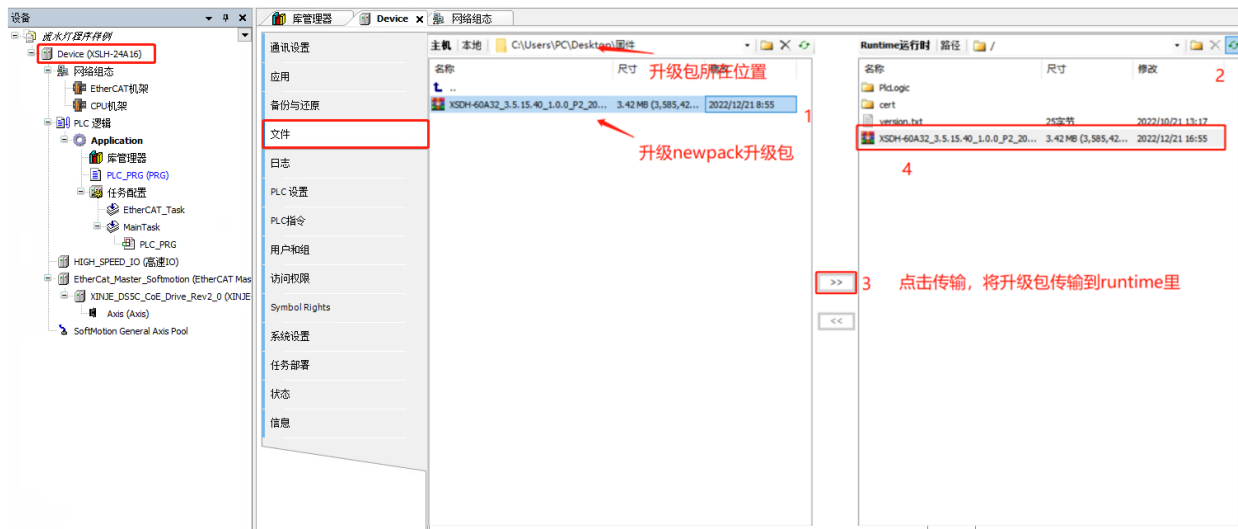
### 8-2-2. 固件获取

联系技术支持, 电话: 400-885-0136。

### 8-2-3. 固件安装及其注意事项

通过 newpack 包升级固件:

创建设备标准工程, 连接设备, 在主设备目录下选择“文件”选项, 点击右上角的刷新后, 将 newpack 升级包传输到 runtime 中, 等待传输完成, 重启设备, 设备在升级时 ERR 灯会常亮, 更新结束后 ERR 会熄灭, 此时就可以扫到设备。

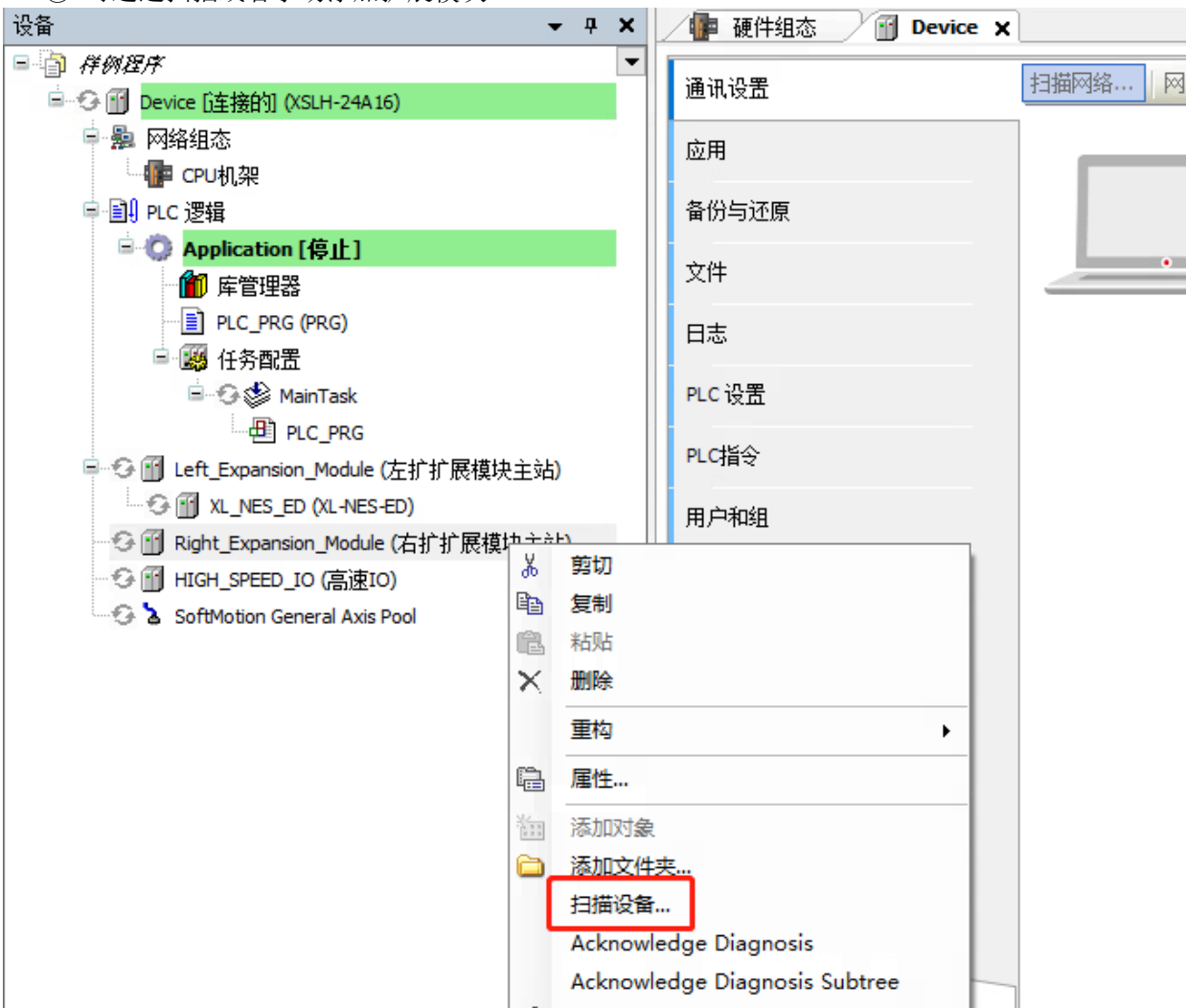


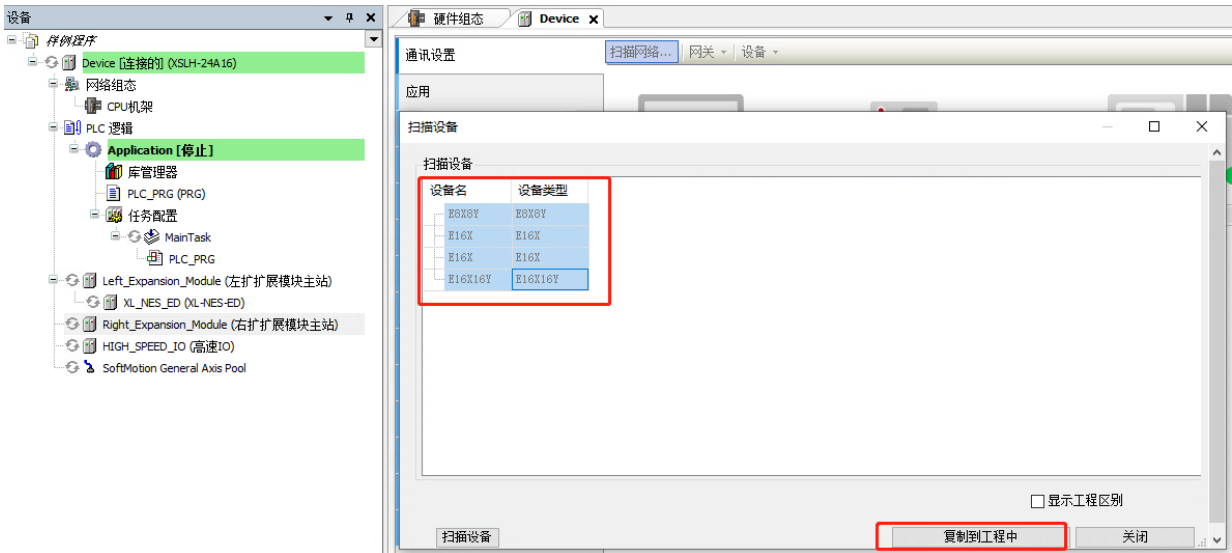
### 8-3. XS 系列的本地扩展模块

① 双击网络组态节点下的 CPU 机架总线节点，可打开本地硬件组态配置界面与右侧“输入/输出模块列表”界面。可通过“输入/输出模块列表”添加本地 IO 模块。如图所示：

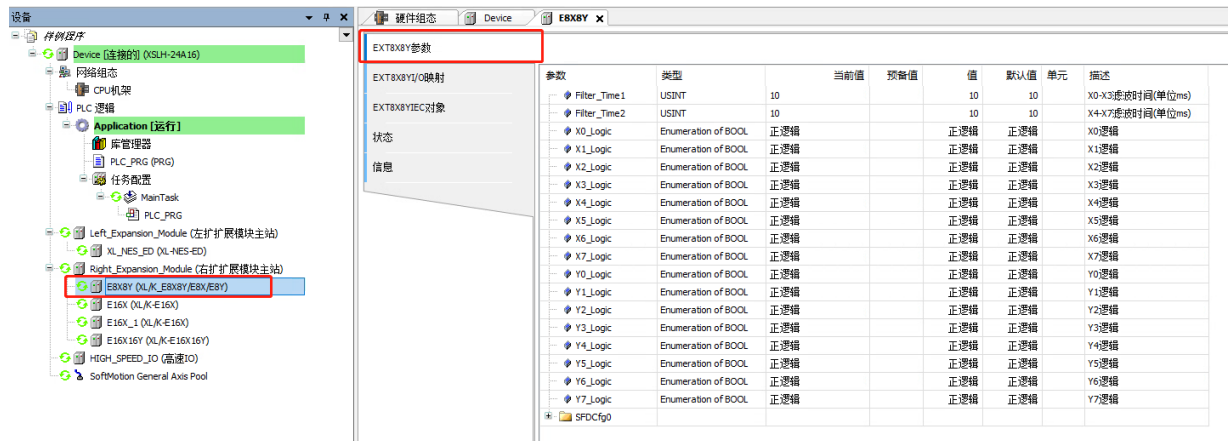


② 可通过扫描或者手动添加扩展模块。



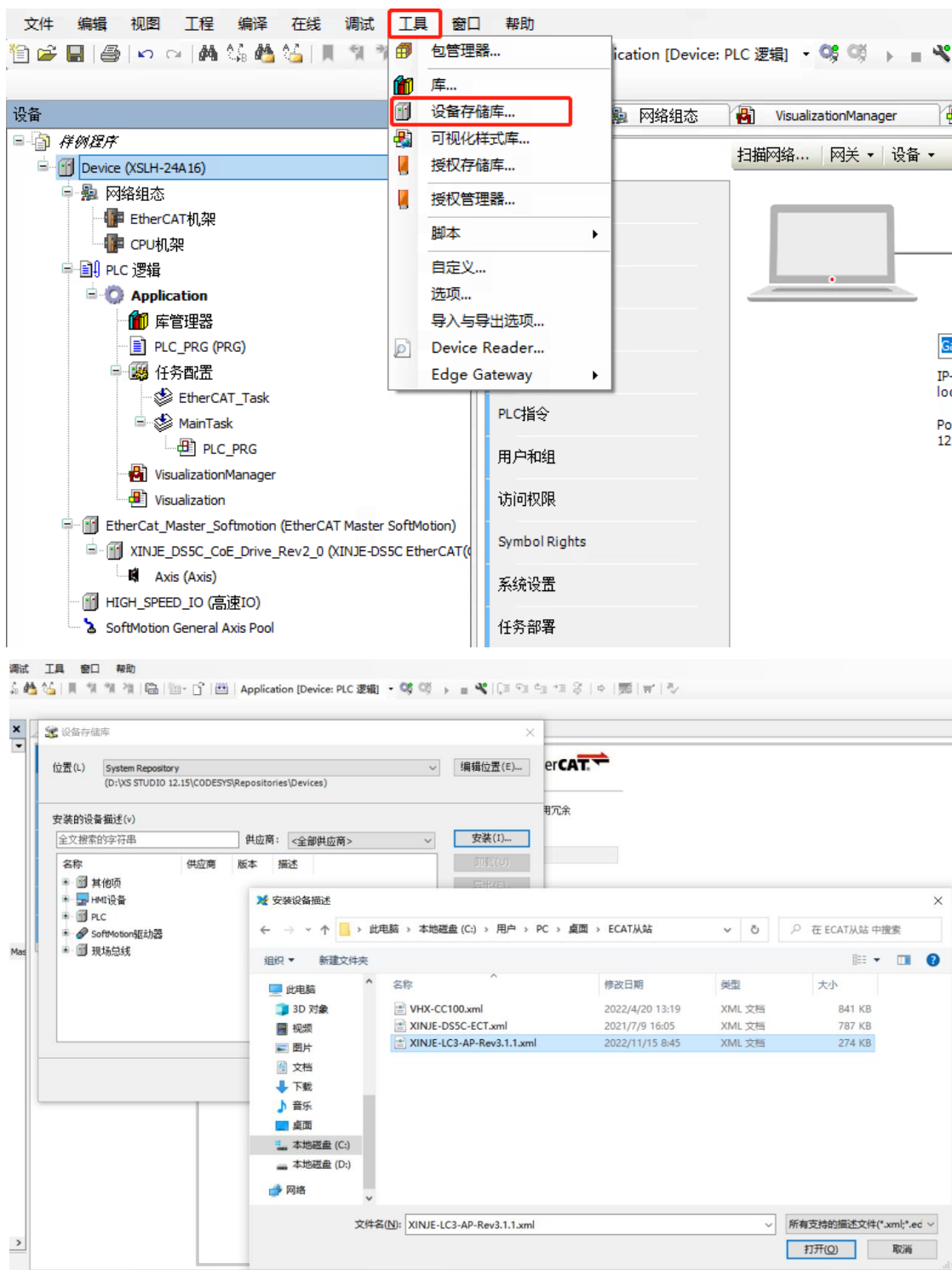


③ 测试结果。



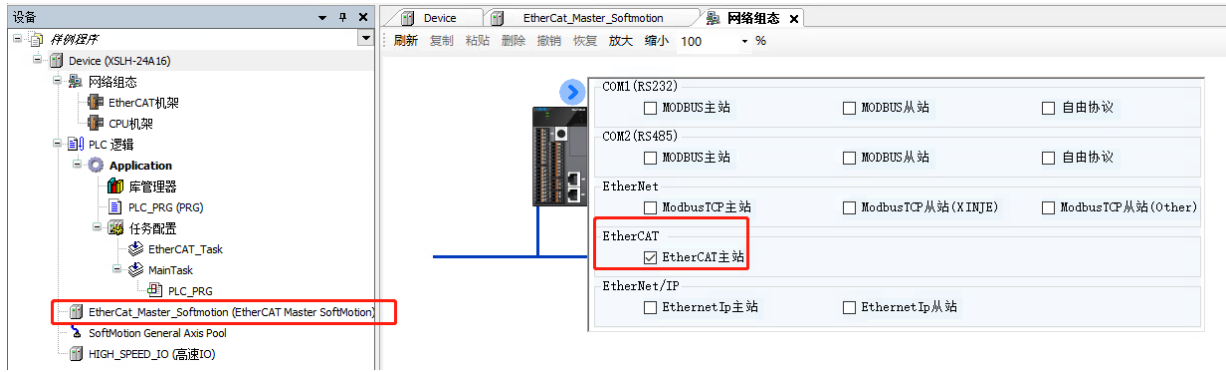
### 8-4. XS 系列的远程扩展模块

- ① 将 LC3-AP 远程模块接上 24V 电源供电。
- ② XS Studio 上位机添加 LC3-AP 描述文件。

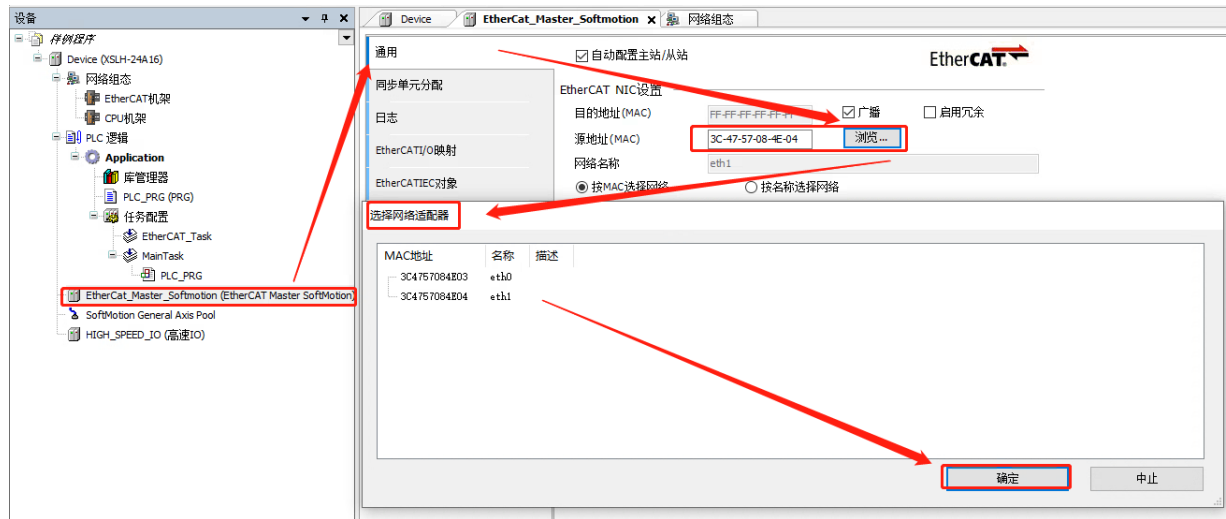


- ③ 添加 EtherCAT 主站。





④ 选择用于通讯的网口。



⑤ 扫描添加 LC3-AP 远程模块。



## ⑥ 复制所有设备到工程中。



## 8-5. 拨码

XSDH-60A32-E 支持拨码功能，其具体功能如下所示：

- 00: 正常启动，无特殊处理，加载用户程序；
- 10: 初始化 IP；
- 01: 上电不加载用户程序。

## 8-6. 用户新安装 XS Studio 上位机，打开后进行编译，会出现很多报错的原因

一般来说是由于缺失库导致的，在工程栏双击打开库管理器-点击下载缺失的库-等待下载缺失库。

## 8-7. 网关显示红点的原因

可能是网关服务被关闭。可以去任务管理器中打开服务“Codesys Gateway V3”或者重启电脑。

## 8-8. 添加多个 EtherCAT 从站后，有 warning 提示的原因

是由于伺服站号重复，不会影响使用，若想清除 warning 和双蓝色下划线，重新扫描一下伺服，然后修改重复站号即可。

## 8-9. EtherCAT 轴一运行会出现通讯断掉的原因

EtherCAT 相关 POU 必须要放在 EtherCAT 任务下，它是有位置同步周期的。

## 8-10. 用户怎么取消密码登录

分为两步：

(1) 在扫描设备的界面的“设备”里--点击 change communication Policy--在弹出来的界面里的 Device User Management 里的 New Policy，改成 Optional User Management。

(2) 在 Devices 界面里选择“Device”--右击选择“初始复位设备[Device]”。这样操作后，就不需要每次登录都需要密码。

若客户想登录时输入密码，将在扫描设备的界面的“设备”里--点击 change communication Policy--在弹出来的界面里的 Device User Management 里的 New Policy，改成“Enforced User Management”。

注：XS3 出厂默认

用户名：Administrator

默认密码：xinje

## 8-11. 为什么无法连接到 PLC 设备

无法连接 PLC 一般归纳为以下几点：

1. 确认为 XS 系列产品(出现过很多将 XD, XG 系列产品当成 XS 系列)。
2. 确认上位机工程设备与目标设备一致，否则也会无法扫描到设备。
3. 确认双方 IP 是否为同一网段，能否 ping 通；若无法确认 IP 地址，可尝试将拨码 1 置 ON 之后重启设备(上电初始化 IP 为 192.168.6.6)，再次进行扫描连接；若网段相同但子网掩码不同，会无法扫描到设备，但可直接输入 IP 地址进行连接设备



[可以保护您的设备,了解更多信息...](#)

4. 若 IP 确认无误还是无法连接设备，可能是 PLC 程序死机(程序里有死循环或超出 PLC 的负载能力)，此时可将拨码 2 置 ON(上电不加载用户程序)，再次扫描连接设备；若能扫描连接，此时下载一个空程序，进行抹除异常的程序之后，再恢复拨码状态，同时检查异常程序(是否有超长循环或任务周期时间过小)。

5. 若以上步骤还是无法连接设备，请反馈技术支持。

## 8-12. IP 地址修改不成功问题

修改 IP 之后若网段不同，同时需要修改网关。修改成功之后重新上电生效。

### 8-13. 提示“没有可用于此对象的源码，是否浏览原始库以显示源代码”问题



- ①指针非法访问：空指针、指针指向不合法区域（指针指向的地址与操作系统内部地址冲突）
- ②数组越界
- ③除数为 0
- ④有符号和无符号变量之间赋值运算
- ⑤for、while、repeat 循环条件使用不当

### 8-14. setposition 清完位置之后断电上电绝对值伺服位置发生变化的问题

- ①中转。
- ②信捷伺服固件 3792 版本，使用 MC\_Home，35 模式。

### 8-15. PLC 死机

- ①ARM 机型：拨码 10N 断电上电，不加载程序。重新下载程序，拨码拨回原位。
- ②X86 机型：D 盘--CODESYS 文件夹--Plclogic--Application 删除。

### 8-16. 在线下载程序丢失

在线下载勾选如下图：



### 8-17. 不同电脑在同一局域网有时会连接到其他设备

解决方法：关闭网络再连接 PLC 或在连接时使用闪烁判断扫描到的设备是否为实际连接设备。

### 8-18. 添加隐含检查功能

在程序的编写过程中，可能会发生如下的几种情况：

- 除法运算的被除数在某些情况下会为零；

- 指针在赋值的过程中可能不小心指向空地址;
- 调用数组时数组边界溢出了。

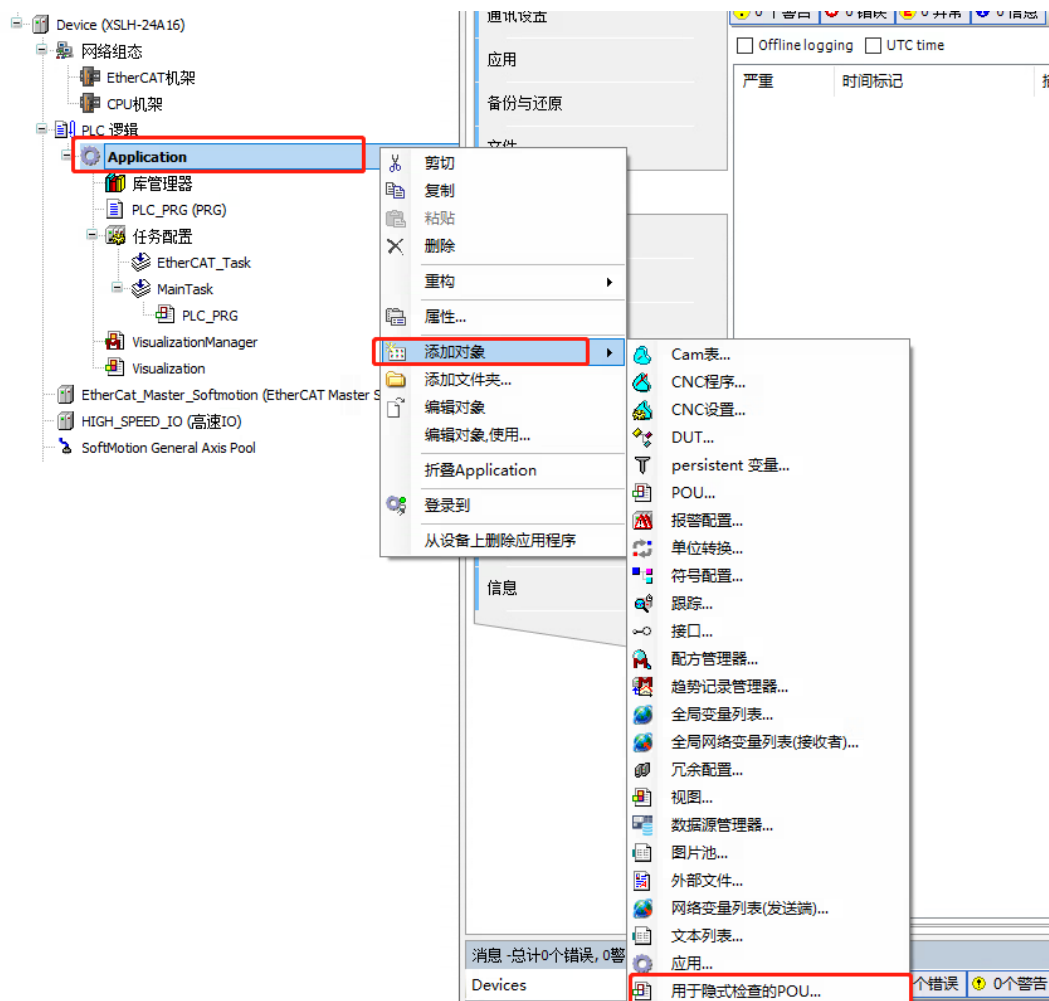
针对上述情况, XS Studio 有专门的解决方案, 在一个应用下可以添加特殊的 POU, 但是这个 POU 程序必须存在在应用下, 如果隐式检查能够检查函数的阵列和界限, 以及除数为零和运行系统中的指针。

注意: 如果设备的校验功能是由一个特殊的库提供的, 那么这个功能可以被停用。

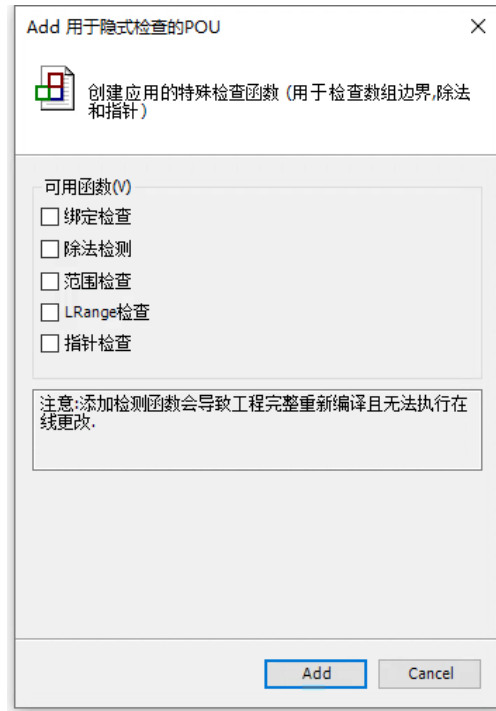
因为这个原因“添加对象“菜单中的 POU”的隐式检查”属性中提供了以下函数功能:

- CheckBounds
- CheckDivInt
- CheckDivLInt
- CheckDivReal
- CheckDivLreal
- CheckRange
- CheckRangeUnsigned
- CheckPointer

当在 POU 中添加一个检查之后, 将会按照选定的编程语言方式打开。默认的编程环境是 ST 语言编辑器。用户可以在应用下鼠标右键选择“添加对象”, 选择“用于隐含检查的 POU”, 随后, 系统则会弹出对话框, 如图所示:



如下, 会对这几种常用的函数做介绍。



隐含检查的 POU 添加选项

1) CheckBounds 此函数检查是否对数组的边界有侵犯（例如，通过检测到的错误标志设置或更改索引）。一个可变的数组类型被分配这个函数被称作隐藏的函数。

当调用该函数参照以下输入参数：

- 索引：字段元素的索引；
- 下限：字段部分的下限；
- 上限：字段部分的下限。

只要索引的范围内，则返回值是索引本身。否则将被返回对应违反上限或下限范围内的字段。

例如，在程序的数组中“a”超出上限，其程序如下，

```
PROGRAM PLC_PRG
VAR a: ARRAY[0..7] OF BOOL;
b: INT:=10;
END_VAR
```

```
a[b]:=TRUE;
```

在程序之初，数组 a 只有 0~7 八个成员，而在实际的程序中，a 数组的第 b 个成员为 TRUE，而 b 在程序定义时却为 10，实际已经超出了数组 a 的定义范围。

当使用了该 CheckBound 函数后，将影响的分配前的影响，被改变的索引值从“10”到上限“7”。因此值 TRUE 将会分配给数组元素 a[7]。

2) Check+数据类型

为了检查除数的值，避免除数为零，可以使用检查函数 CheckDivInt, CheckDivLint, CheckDivReal 以及 CheckDivLReal。在将它们包含在应用中之后每个相关代码发生的除法过程都将产生一个此函数调用的预处理。

例如，使用除法指令，具体程序如下：

```
PROGRAM PLC_PRG
VAR
erg:REAL;
v1:REAL:=799;
d:REAL;
END_VAR
```

```
erg:= v1 / d;
```

在上例中，erg 等于 v1 除以 d，而 d 在变量定义之初并没有赋予初值，故其初值为 0，在程序中如

果直接将数除以 0，系统则会出错。但是，如果在指令中经过了指向除法运算检查的 CheckDivReal 函数之后除数“d”的值在初始化的时候变为“1”。因此除法最终结果为 799，能够有效的避免控制器异常出错。

### 3) CheckRange(Un)Signed

为了在运行过程中检查域的限制，可以使用函数 CheckRangeSigned 或 CheckRangeUnsigned。此检查功能的目的是恰当的子集违规处理（如设置一个检测到的错误旗帜或改变值）。当一个变量的子集类型被确认后这个功能将被隐藏访问。

当访问这个功能时得到以下输入参数：

- 值：域类型被分配的值
- 低：域的下限
- 高：域的上限

如果被分配的值是在有效的域内，他将作为功能中的返回值被使用。否则，对应超过范围的数值，要么上限或下边界的范围将被返回。

例如，分配  $i:=10*y$  将被隐性替代为

```
i := CheckRangeSigned(10*y, -4095, 4095);
```

如果 y 的值为 1000，变量 i 将不会分配到原始执行提供的  $10*1000=10000$ ，取而代之的是 4095，因为函数设定的上限值最大为 4095。

例如，死循环的例子：

```
VAR
ui : UINT (0..10000);
END_VARFOR ui:=0 TO 10000 DO
...
END_FOR
```

FOR 循环将永远不会离开，因为检查功能制止了将 ui 的超过 10000。

**注意：**使用 CheckRangeSigned 指令和 CheckRangeUnsigned 的功能，可能会导致在一个无限循环，例如：如果一个子界类型被用作循环不匹配子范围的增量。

### 4) CheckPointer

函数 CheckPointer 检查地址的指针引用是否都在有效的内存范围内。在运行过程中，用户可能在每个指针操作都可以使用 CheckPointer 来进行指针访问的检查。

## 8-19. 实现掉电保持的注意点

- 1、掉电保持新增或删减需要将程序登录并下载或在线修改时勾选更新自动启动程序。
- 2、修改了掉电保持区域，内存分配重新排列了。数据会被清 0。
- 3、ARM 机型（XSLH、XSDH、XS3）和 X86 机型（XSA330-W）内部有 UPS 可以实现掉电保持，其他 X86 机型（原 M210）需要确定是否配有 UPS。
- 4、确定程序中是否有其他地方进行赋值。

## 8-20. 打开工程报错，工程保存必须以存档的方式保存

project 格式工程是没有包含全部信息的，当打开他人的工程或以不同的版本打开时，会丢失信息，要以打包格式存储，这样才不会丢失信息



8-21. 设置可以添加行注释和节点注释





## 手册更新日志

本手册的资料编号记载在手册封面的右下角，关于手册改版的信息汇总如下：

序号	资料编号	章节	更新内容
1	PS06 20230109 1.0	-	1、第一版手册发布
2	PS06 20230504 1.1	-	1、增加设备组态章节； 2、增加信捷 402 轴，替换原本 CIA402 轴； 3、新增特殊功能中的高速 IO 配置； 4、更新手册相应配图。



微信扫一扫，关注我们

**XINJE** 无锡信捷电气股份有限公司  
WUXI XINJE ELECTRIC CO., LTD.

---

地址：江苏省无锡市滨湖区建筑西路 816 号

总机：0510-85134136

传真：0510-85111290

网址：[www.xinje.com](http://www.xinje.com)

邮箱：[xinje@xinje.com](mailto:xinje@xinje.com)

全国技术服务热线：400-885-0136