INOVANCE

中型PLC编程软件使用手册

AM400/AM600/AP700/AC800

综合手册

B00

资料编码 19010334

目录

版本变更记录

修订日期	发布版本	变更内容
2018-04	A00	第一版发行
2020-10	B00	1. 增加 "3.9 EtherNET 通讯”； 2. 增加“附录 I 同步工程信息”； 3. 增加“附录 J PLC 运行异常处理方法指导”； 4. 对上一版本进行细节勘误。

第 1 章 产品简介

1. 产品简介

1.1 概述

1.1.1 产品简介

汇川中型可编程逻辑控制器（以下简称中型 PLC），涵盖 AM400/AM600/AC800/AP700 等系列，为用户提供智能自动化解决方案。中型 PLC 采用 IEC61131-3 编程语言体系，支持 PLCopen 标准编程语言。AM400/AM600 采用机架式布局，每个机架支持本地扩展 16 个扩展模块，并可通过 Profibus-DP、EtherCAT、CANopen 等多种工业现场总线远程扩展机架。AM600 本地扩展模块通过内部总线协议进行 IO 扩展，支持数字输入 / 输出模块、模拟输入 / 输出模块、温度模块等多种功能模块。通过 EtherCAT 总线可实现高性能运动控制功能；具有单轴加减速控制功能、电子齿轮功能、电子凸轮功能，还可通过高速 I/O 实现单轴基本定位功能，且最高频率可达 200kHz；同时支持 RS485、以太网、USB 等通信功能。AC800 系列采用书本式结构，运控性能卓越，最多可支持 256 轴运动控制，使用灵活，满足用户多样化的应用需求。

中型可编程控制器具有以下功能特点：

(1) 多种运动控制功能：总线运动控制、脉冲运动控制；

(2) 支持更多的 I/O 点数：最多可达上万点；

(3) 更大的程序容量和数据存储区；

(4) 更快的指令执行速度；

(5) 支持更多的高端现场总线（EtherCAT、CANopen、EtherNet/IP、Profibus-DP）；

(6) 更易用的软件，满足用户不同应用需求；

(7) 支持在线侦错模式；

(8) 支持在线编辑模式。

下面列出了中型 PLC 涵盖的 CPU 模块对象以及对应的差异点。

表 1-1 CPU 模块软件功能特性

产品 型号【注 1】	本地扩展模块数	程序存储空间	数据存储空间	掉电数据保存大小	运动控制轴数	高速 I/O 功能	软元件特性	输出类型
AM401-CPU1608TP	8	10M	20M	480K	伺服轴 4 个，4PME 4 个	16 入 8 出高速 I/O	√	源型输出
AM402-CPU1608TP	8	10M	20M	480K	伺服轴 8 个，4PME 4 个	16 入 8 出高速 I/O	√	源型输出
AM401-CPU1608TN	8	10M	20M	480K	伺服轴 4 个，4PME 4 个	16 入 8 出高速 I/O	√	漏型输出
AM402-CPU1608TN	8	10M	20M	480K	伺服轴 8 个，4PME 4 个	16 入 8 出高速 I/O	√	漏型输出
AM403-CPU1608TN	16	10M	20M	480K	伺服轴 16 个，4PME 4 个	16 入 8 出高速 I/O	√	漏型输出
AM600-CPU1608TP	16	10M	20M	480K	32	16 入 8 出高速 I/O	√	源型输出
AM600-CPU1608TN	16	10M	20M	480K	32	16 入 8 出高速 I/O	√	漏型输出
AM610-CPU1608TP	16	10M	20M	480K	×	16 入 8 出高速 I/O	√	源型输出
AC801-0221-U0R0	×	128M	128M	5M（需外接 UPS）	48	×	×	×
AC802-0222-U0R0	×	128M	128M	5M（需外接 UPS）	128	×	×	×
AC810-0122-U0R0	×	128M	128M	5M（需外接 UPS）	256	×	×	×

产品 型号【注 1】	本地扩展模块数	程序存储空间	数据存储空间	掉电数据保存大小	运动控制轴数	高速 I/O 功能	软元件特性	输出类型
AC812-0322-U0R0	×	128M	128M	5M（需外接 UPS）	256	×	×	×
AP702-0221-U0R0	×	128M	128M	5M（需外接 UPS）	48	×	×	×
AP703-0221-U0R0	×	128M	128M	5M（需外接 UPS）	48	×	×	×
AP705-LM【注 2】	×	128M	128M	5M（需外接 UPS）	48	×	×	×

表 1-2 CPU 模块通讯特性

产品型号【注 1】	通讯					
	EtherCAT	Profibus-DP	CANopen/CANlink	Modbus TCP	Modbus（串口）	EtherNET/IP
AM401-CPU1608TP	1 路 (最多 128 从站)	×	1 路 (最多 63 从站)	1 路 (最多 63 从站)	1 路 (最多 31 从站)	1 路 (最多 64 从站)
AM402-CPU1608TP	1 路 (最多 128 从站)	×	1 路 (最多 63 从站)	1 路 (最多 63 从站)	1 路 (最多 31 从站)	1 路 (最多 64 从站)
AM401-CPU1608TN	1 路 (最多 128 从站)	×	1 路 (最多 63 从站)	1 路 (最多 63 从站)	1 路 (最多 31 从站)	1 路 (最多 64 从站)
AM402-CPU1608TN	1 路 (最多 128 从站)	×	1 路 (最多 63 从站)	1 路 (最多 63 从站)	1 路 (最多 31 从站)	1 路 (最多 64 从站)
AM403-CPU1608TN	1 路 (最多 128 从站)	×	1 路 (最多 63 从站)	1 路 (最多 63 从站)	2 路 (每路最多 31 从站)	1 路 (最多 64 从站)
AM600-CPU1608TP	1 路 (最多 128 从站)	×	1 路 (最多 63 从站)	1 路 (最多 63 从站)	2 路 (每路最多 31 从站)	1 路 (最多 64 从站)
AM600-CPU1608TN	1 路 (最多 128 从站)	×	1 路 (最多 63 从站)	1 路 (最多 63 从站)	2 路 (每路最多 31 从站)	1 路 (最多 64 从站)
AM610-CPU1608TP	×	1 路 (最多 124 从站)	×	1 路 (最多 63 从站)	2 路 (每路最多 31 从站)	1 路 (最多 64 从站)
AC801-0221-U0R0	1 路 (每路最多 128 从站)	×	×	2 路 (最多 128 从站)	2 路 (每路最多 31 从站)	1 路 (最多 64 从站)
AC802-0222-U0R0	2 路 (每路最多 128 从站)	×	×	2 路 (最多 128 从站	2 路 (每路最多 31 从站)	1 路 (最多 64 从站)
AC810-0122-U0R0	2 路 (每路最多 128 从站)	×	×	2 路 (最多 128 从站)	2 路 (每路最多 31 从站)	1 路 (最多 64 从站)
AC812-0322-U0R0	2 路 (每路最多 128 从站)	×	×	2 路 (最多 128 从站)	2 路 (每路最多 31 从站)	1 路 (最多 64 从站)
AP702-0221-U0R0	1 路 (每路最多 128 从站)	×	×	2 路 (最多 128 从站	2 路 (每路最多 31 从站)	1 路 (最多 64 从站)
AP703-0221-U0R0	1 路 (每路最多 128 从站)	×	×	2 路 (最多 128 从站)	2 路 (每路最多 31 从站)	1 路 (最多 64 从站)
AP705-LM【注 2】	1 路 (每路最多 128 从站)	×	×	2 路 (最多 128 从站)	2 路 (每路最多 31 从站)	1 路 (最多 64 从站)

【注 1】: 1) 上述信息不包括电源模块及尾板 ;2)AM610-CPU1608TP 产品已经停止维护。

【注 2】: AP705-LM 为泛机床行业专机。

1.1.2 产品配置及模块说明

中型可编程控制器产品丰富，用户可根据需要选择适用于具体应用的产品配置。以 AM600 系列 PLC 为例，中型 PLC 包含 AM600-CPU1608TP 架构（EtherCAT+CANopen 总线型）和 AM610-CPU1608TP 架构（Profibus-DP 总线型），典型集成示意图如下图所示：

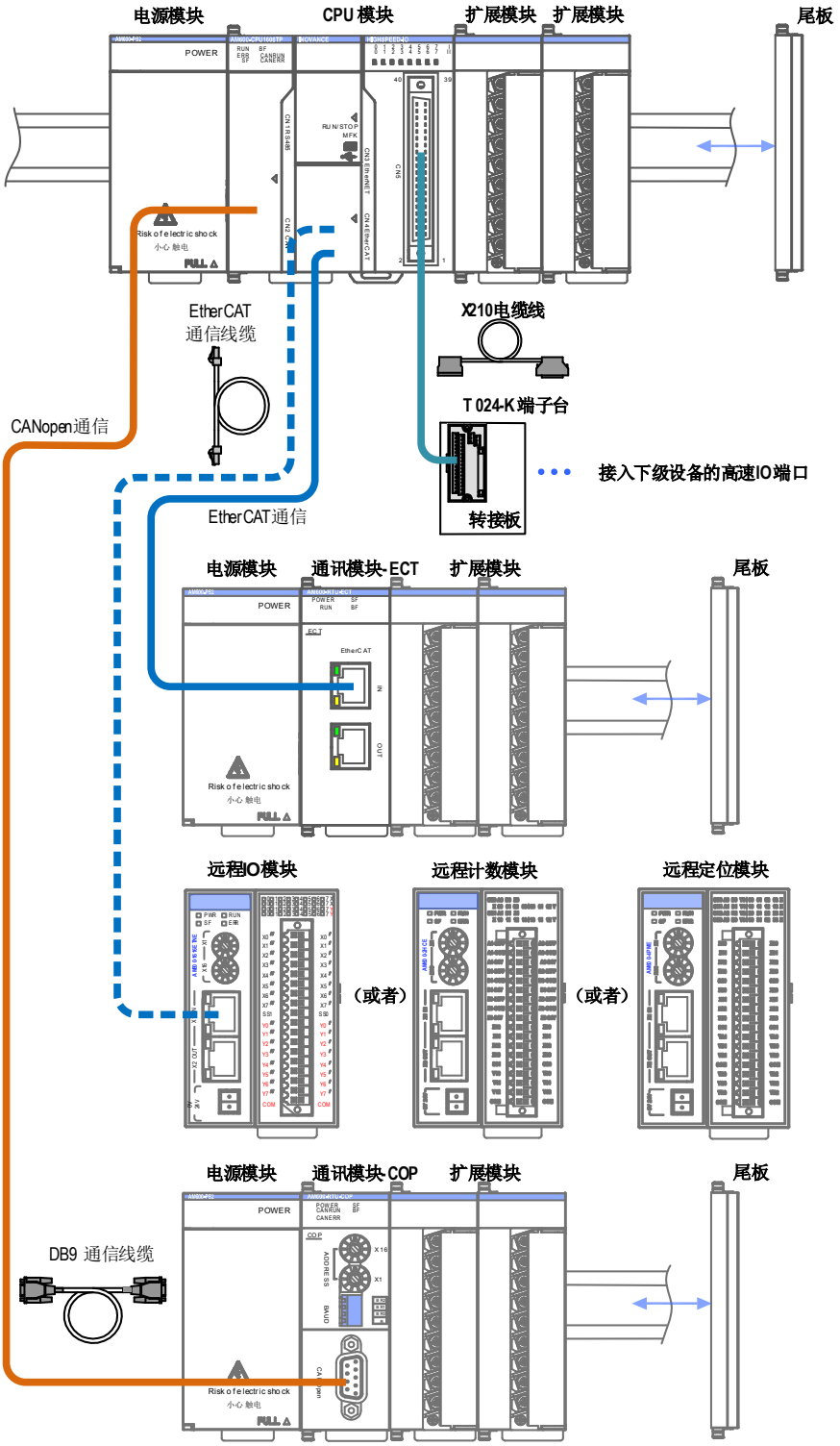


图 1-1 AM600-CPU1608TP 系统集成示意图

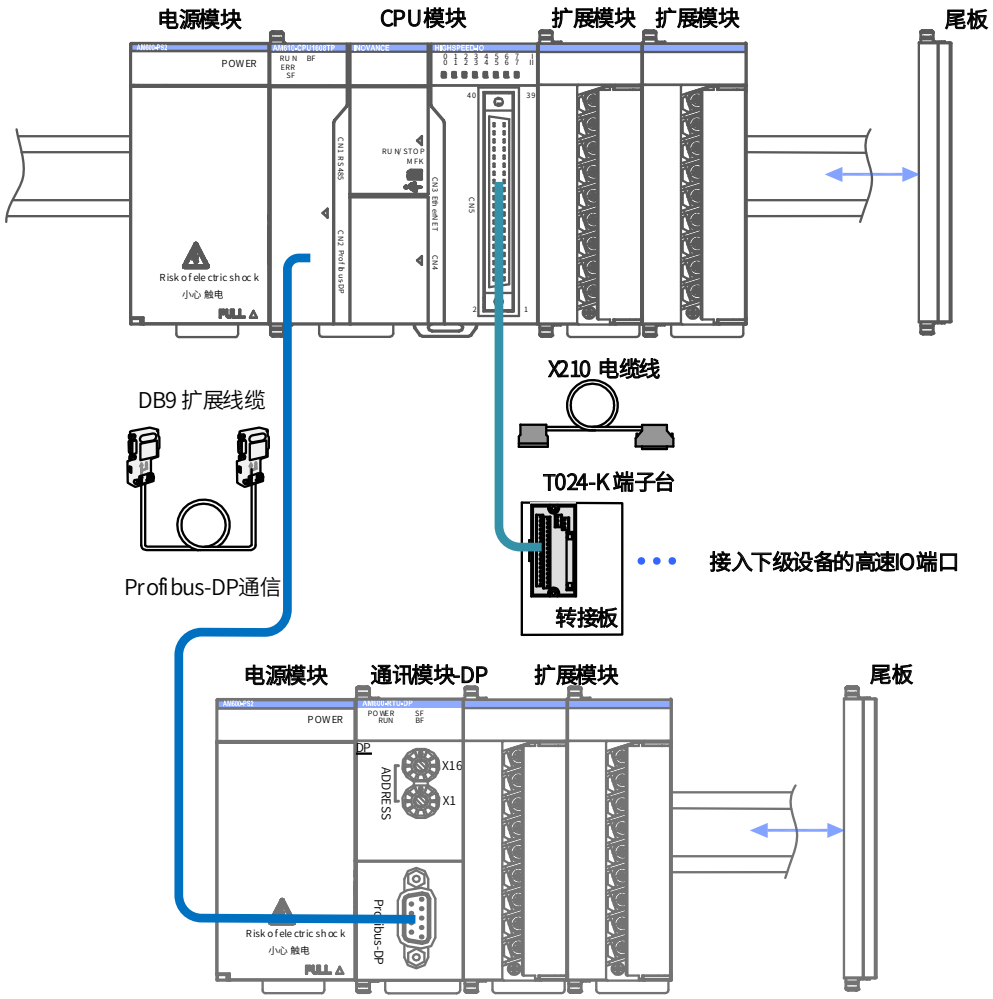


图 1-2 AM610-CPU1608TP 系统集成示意图

NOTE

◆ AM610-CPU1608TP 产品和 AM600-RTU-DP 模块产品已经停止维护。

AM600 系列中型 PLC 按功能可分为电源模块、CPU 模块、远程通信模块及扩展模块等四大类，具体包含：

订货编码	型号【注】	分类	描述
01440010	AM600-PS2	电源模块	220V 电压输入，24V/2A 输出
01440028	AM401-CPU1608TP	CPU 模块	1 路 RS485，1 路 CANopen/CANlink，1 路 LAN 支持 4 轴运动控制，支持 EtherCAT 内置 16 入 8 出高速 I/O 源型输出
01440029	AM402-CPU1608TP	CPU 模块	1 路 RS485，1 路 CANopen/CANlink，1 路 LAN 支持 8 轴运动控制，支持 EtherCAT 内置 16 入 8 出高速 I/O 源型输出
01440032	AM401-CPU1608TN	CPU 模块	1 路 RS485，1 路 CANopen/CANlink、1 路 LAN 支持 4 轴运动控制，支持 EtherCAT 内置 16 入 8 出高速 I/O，漏型输出

订货编码	型号【注】	分类	描述
01440031	AM402-CPU1608TN	CPU 模块	1 路 RS485，1 路 CANopen/CANlink，1 路 LAN 支持 8 轴运动控制，支持 EtherCAT 内置 16 入 8 出高速 I/O 漏型输出
01440126	AM403-CPU1608TN		2 路 RS485，1 路 CANopen/CANlink，1 路 LAN 支持 16 轴运动控制，支持 EtherCAT 内置 16 入 8 出高速 I/O 漏型输出
01440014	AM600-CPU1608TP		2 路 RS485，1 路 CANopen/CANlink，1 路 LAN 支持基本运动控制功能，支持 EtherCAT 内置 16 入 8 出高速 I/O 源型输出
01440016	AM610-CPU1608TP		2 路 RS485，1 路 LAN 支持基本运动控制功能，支持 Profibus-DP 内置 16 入 8 出高速 I/O 源型输出
01440143	AC812-0322-U0R0	书本式控制器	2 路 USB2.0，2 路 USB3.0 1 路 RS485/RS232，4 路 LAN 支持多达 256 轴运动控制功能，支持 EtherCAT 多功能扩展槽，内部 Mini-PCIE 扩展槽
01440038	AC810-0122-U0R0		
01440101	AC802-0222-U0R0		2 路 USB2.0，2 路 USB3.0 1 路 RS485/RS232，4 路 LAN 支持多达 128 轴运动控制功能，支持 EtherCAT 多功能扩展槽，内部 Mini-PCIE 扩展槽
01440103	AC801-0221-U0R0		2 路 USB2.0，2 路 USB3.0 1 路 RS485/RS232，支持以太网 支持多达 48 轴运动控制功能，支持 EtherCAT 内部 Mini-PCIE 扩展槽
01440066	GL10-1600END	数字输入模块	16 点 DI 模块，直流 24V 输入（源 / 漏型）
01440081	GL10-0016ER	数字输出模块	16 点 DO 模块，继电器输出
01440069	GL10-0016ETP		16 点 DO 模块，晶体管输出（源型）
01440067	GL10-0016ETN		16 点 DO 模块，晶体管输出（漏型）
01440080	GL10-4AD	模拟输入模块	4 通道 AD 模块，支持电压 / 电流模拟量输入
01440071	GL10-4DA	模拟输出模块	4 通道 DA 模块，支持电压 / 电流模拟量输出
01440082	GL10-0032ETN	数字量输出模块	32 点 DO 模块，晶体管输出（漏型）
01440066	GL10-3200END	数字量输入模块	32 点 DI 模块，直流 24V 输入（源 / 漏型）
01440121	GL10-2PH	差分输出脉冲模块	2 路高速差分输出脉冲定位模块，输出频率 1MHz，8 路普通输入
01440129	GL10-4PM	本地脉冲定位模块	4 通道脉冲定位输出
01440075	GL10-4PT	温度模块	4 通道热电阻温度采集，支持多种热电阻类型
01440078	GL10-4TC	温度模块	4 通道热电偶温度采集，支持多种热电偶类型
01440070	GL10-8TC	温度模块	8 通道热电偶温度采集，支持多种热电偶类型
01440074	GL10-PS2	电源模块	220V 电压输入，24V/2A 输出
01440077	GR10-0808ETNE	EtherCAT 从站 IO 模块	8 点数字量输出，晶体管输出（漏型）；8 点数字量输入，支持源漏型
01440255	GR10-1616ETNE	EtherCAT 从站 IO 模块	16 点数字量输出，晶体管输出（漏型）；16 点数字量输入，支持源漏型
01440252	GR10-4PME	EtherCAT 从站定位模块	EtherCAT 从站 4 通道定位模块
01440279	GR10-2HCE	EtherCAT 从站计数模块	EtherCAT 从站 2 通道计数模块
01440058	GR10-4ADE	模拟量输入模块	EtherCAT 从站 4 通道模拟量输入模块
01440060	GR10-4DAE	模拟量输出模块	EtherCAT 从站 4 通道模拟量输出模块
01440123	GR10-4TCS-PID	温度控制模块	4 通道 TC 热电偶温度检测模块
01440059	GR10-8TCE	温度检测模块	8 通道热电偶温度采集，支持多种热电偶类型
01440127	GR10-8PBE	EtherCAT 从站探针模块	GR10 系列 8 通道 EtherCAT 探针模块
01440256	GR10-2PHE	差分输出脉冲模块	支持 2 通道高速差分输出脉冲，可配置 8 个输入
01440135	GR10-EC-6SW	6 路 EtherCAT 分支模块	1 路 EtherCAT 输入，5 路 EtherCAT 输出
01440177	GR10-1616ERE-BD	扩展板卡	GR10 系列可编程控制器 16 点输入 16 点输出 EtherCAT 从站板卡

订货编码	型号【注】	分类	描述
01440012	AM600-RTU-DP	DP 通讯模块	Profibus-DP 协议通讯接口模块
01440073	GL10-RTU-ECTA	EtherCAT 通讯模块	EtherCAT 协议通讯接口模块
01440013	AM600-RTU-ECTA	EtherCAT 通讯模块	EtherCAT 协议通讯接口模块
01440083	GL10-RTU-COP	CANopen 通讯模块	CANopen 协议通讯接口模块

【注】原 AM600 系列的扩展模块已升级更新为 GL10/GR10 系列，升级前后产品兼容；AM610-CPU1608TP 产品和 AM600-RTU-DP 模块产品已经停止维护。

1.1.3 系统应用流程

中型可编程逻辑控制器的基本应用流程如下图所示，模块的安装及配线操作指导请参见《AM600 系列可编程逻辑控制器硬件手册》和《AC810 系列可编程逻辑控制器硬件手册》。

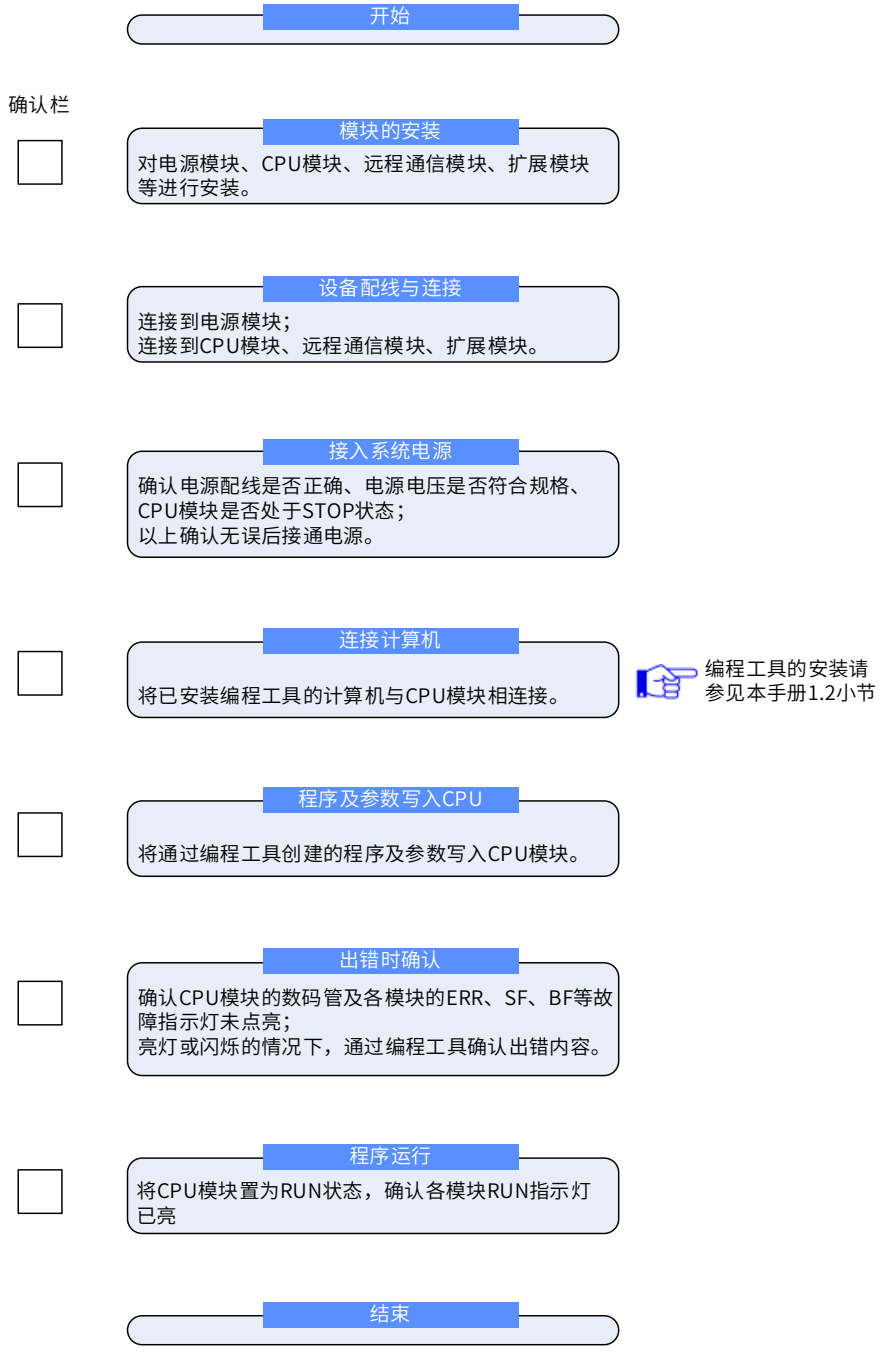


图 1-3 中型可编程控制器应用基本流程图

1.2 InoProShop 概述

1.2.1 InoProShop 简介

InoProShop 是面向中型可编程控制器产品的编程组态软件。InoProShop 基于 CoDeSys V3（简称 CoDeSys）平台，为中型可编程控制器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。

通过 InoProShop 可完成对工程和设备的管理，为中型 PLC 产品提供以下配置方案：

1） CPU 配置；
2） I/O 模块配置；
3） EtherCAT 总线；
4） Profibus-DP 总线；
5） CANopen/CANlink 总线；
6） Modbus/ModbusTCP 总线；
7） EtherNet/IP 总线；
8） 高速 I/O。

支持程序的编写、下载和调试等功能，并为编程者提供如下便利：

■ 标准化编程（符合 IEC 61131-3 标准）

支持多种编程语言：结构化文本（ST）、梯形图（LD)、顺序功能图（SFC）和 IEC61131-3 扩展编程语言连续功能图（CFC）。

■ 灵活的功能块库

全面的功能块库并支持用户自定义库。

■ 离线仿真功能

不需要连接 PLC 硬件，完成程序调试仿真。

■ 智能的调试查错功能

预编译及编译查错，快速定位编程错误，诊断及日志。

■ 采样跟踪

过程变量的时序图建立。

1.2.2 InoProShop 与硬件的连接

编程设备可以通过以太网（可经过集线器、交换机等）或 USB 线缆与中型 PLC 相连，使用 InoProShop 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。

图 1-4 InoProShop 软件与硬件连接示意图

1.2.3 软件获取与安装

1 软件的获取

汇川中型可编程控制器的用户编程软件 InoProShop 为免费软件，如需安装文件及中型 PLC 系列产品的参考资料等，可通过以下途径获取：

- 从汇川的各级经销商处获得软件安装光盘；
- 在汇川技术官网（www.inovance.com）的“服务与支持” “资料下载”页面免费下载软件安装包；

由于汇川公司在不断完善产品和资料，建议用户在需要时及时更新软件版本，查阅最新发布的参考资料，有利于用户的应用设计。

2 软件安装环境要求

具备以下条件的台式 PC 或便携式 PC 机：

- Windows 7/ 或 Windows 10 操作系统；推荐 64 位操作系统；
- 内存：4GB 或更高配置；
- 空间：可用硬盘空间 5GB 以上。

PC 与中型 PLC 控制器按以下方式完成连接：

连接方式	所需电缆	备注
采用 LAN 网络电缆连接（推荐）	需要本地网络中有 1 个空闲的 LAN 网口、1 根网络电缆。	支持 PC 与中型 PLC 之间较远距离连接，如在办公室对车间里的中型 PLC 进行编程等应用环境，而且交互通讯速率更快
采用 USB 电缆连接	需要 1 根 USB 电缆，其电缆的连接中型 PLC 控制器一端需为 Mini USB 插头。	目前支持 AM400/AM600 系列 PLC

1.2.4 安装步骤

1 安装前准备

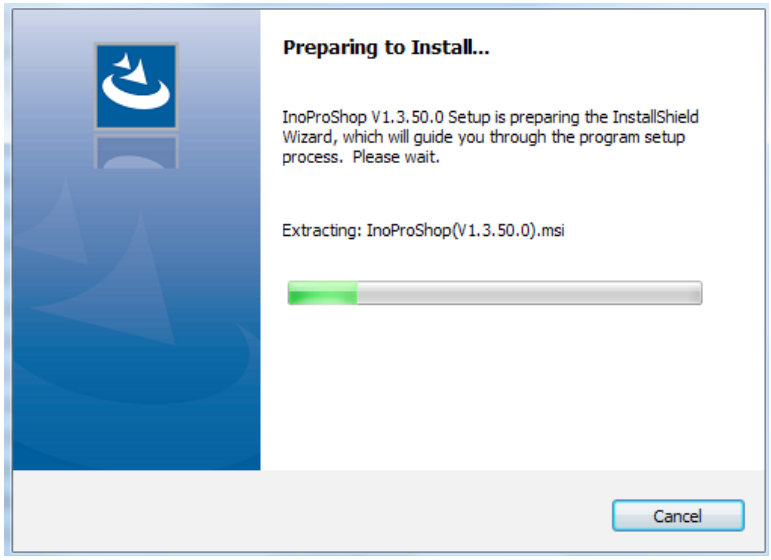
首次安装 InoProShop 时，请检查电脑硬盘的剩余空间情况，确认所要安装的目标盘剩余空间有 5GB 以上，直接安装即可。

如果是升级安装 InoProShop，请先备份已有的工作文件，然后卸载旧版本 InoProShop；重新启动电脑后，再开始安装新版本软件。

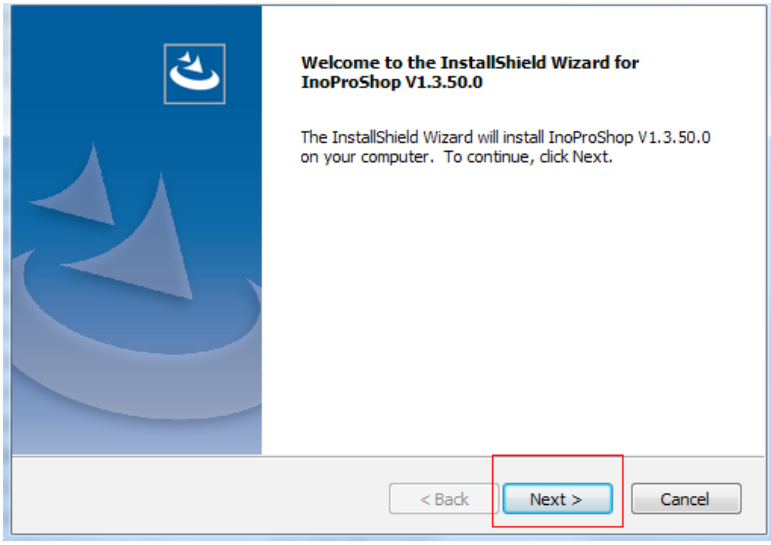
2 开始安装

通过 Windows 的资源管理器，在安装文件所在目录，双击打开 InoProShop（V*.*.*.*）.exe 文件（V*.*.*.* 为 InoProShop 的软件版本，请确保您安装的版本为最新版本）。

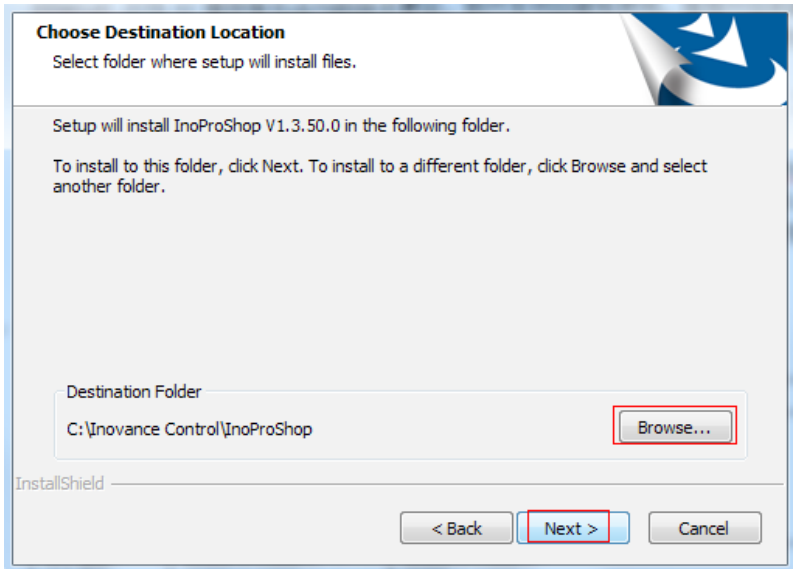
启动安装后，可以看到如下界面，表示进入安装准备阶段：



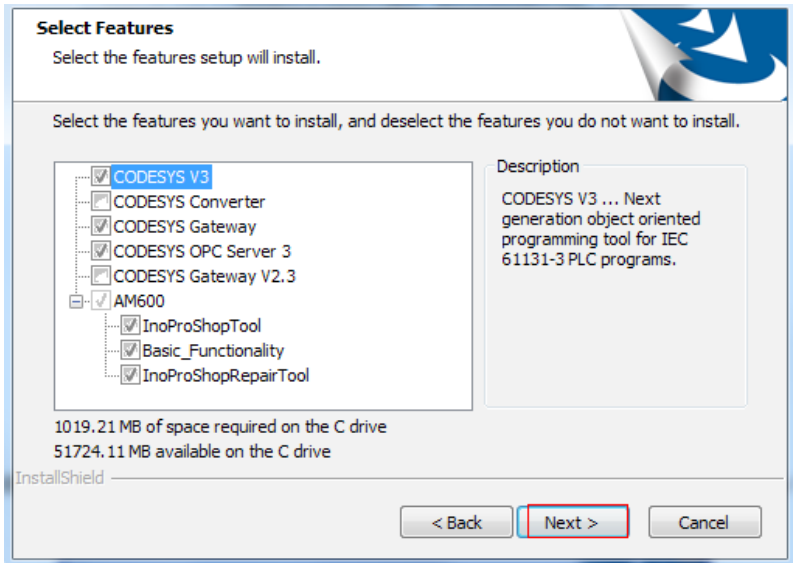
出现如下提示界面后，点击“Next”，开始安装：



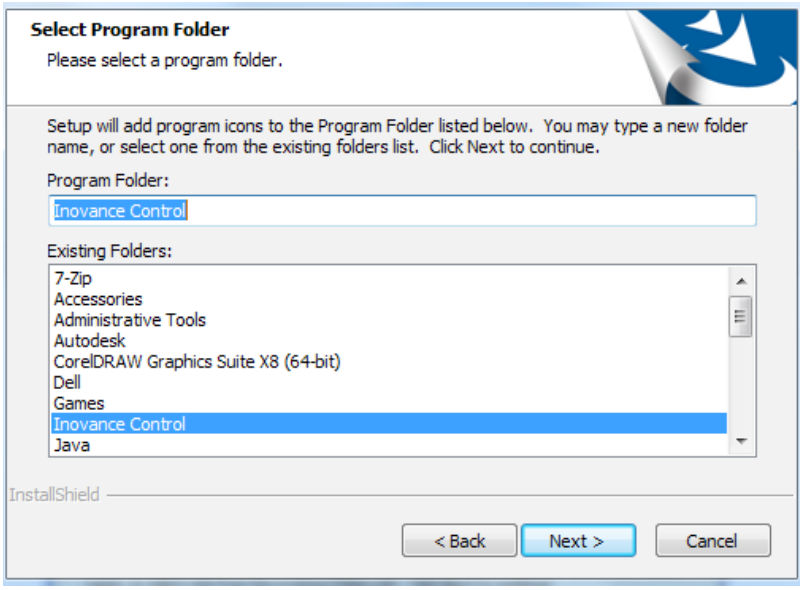
设置好软件安装路径后，点击“Next”，进入下一步：



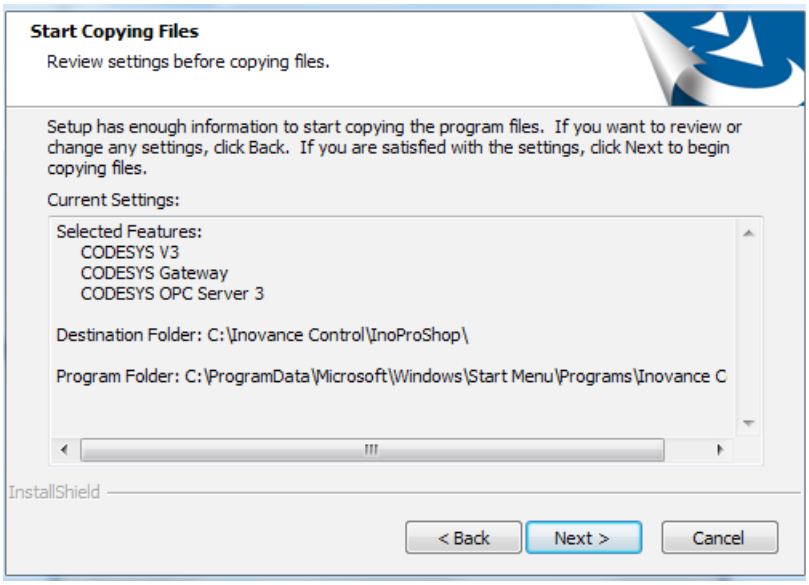
进入安装组件选择界面，可个性化进行勾选，如无特殊需求，按默认勾选即可，点击“Next”：



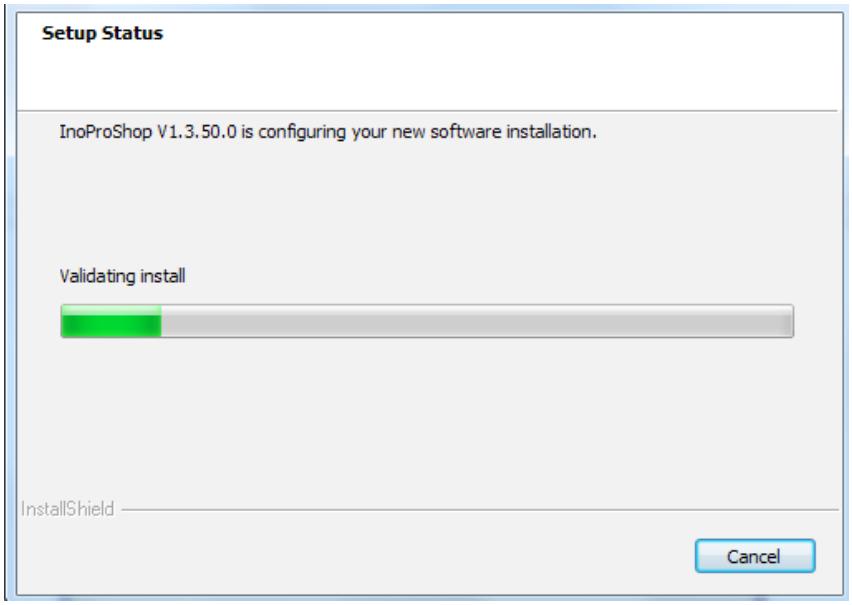
出现如下提示界面，点击“Next”：

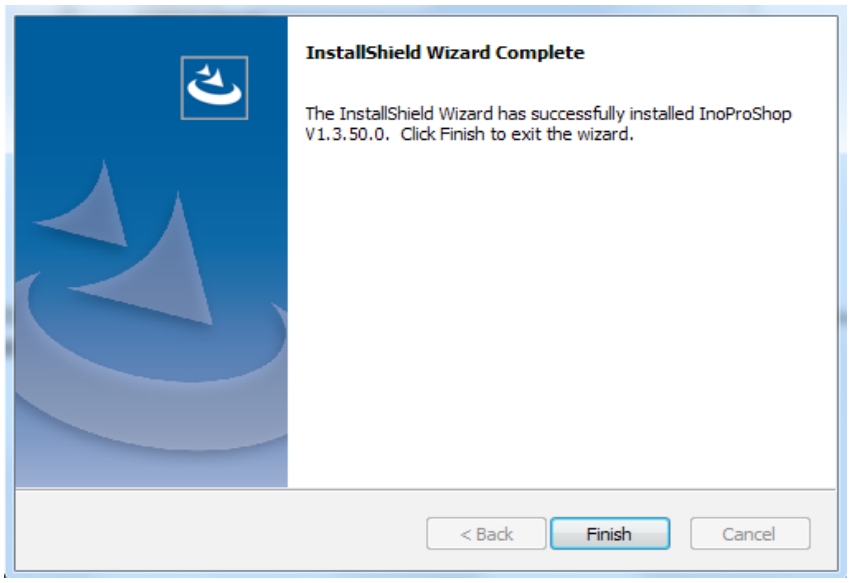


出现如下提示界面，点击“Next”：



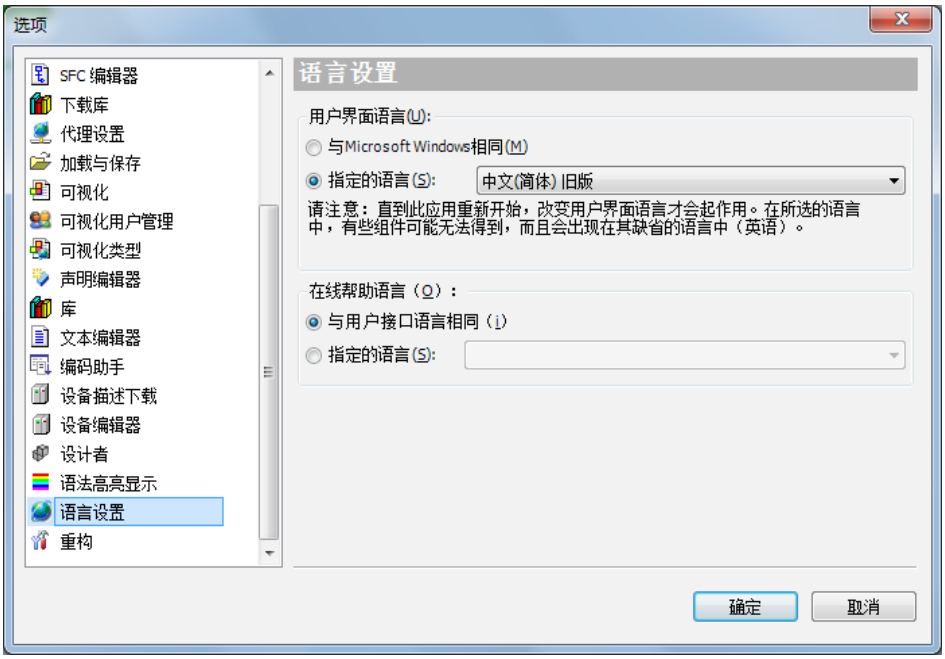
显示如下界面后，等待安装进度条，直到出现下图所示界面提示，点击“Finish”完成 InoProShop 的安装。





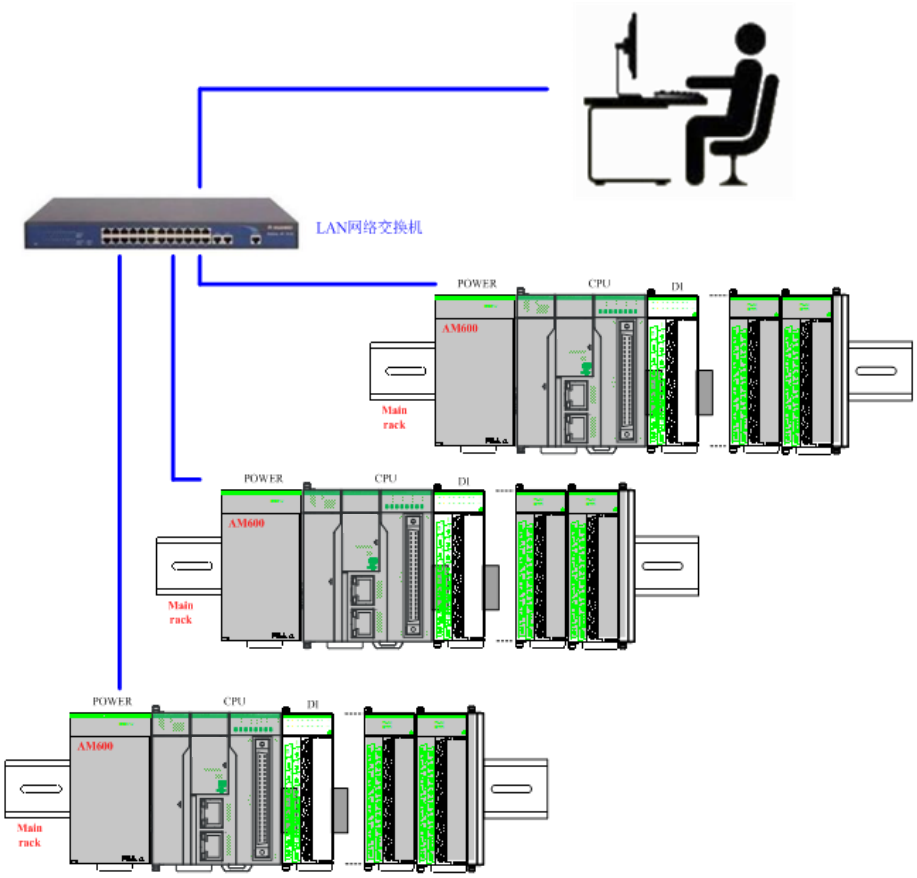
3 语言设置

完成安装后，InoProShop 界面的操作语言默认为简体中文，若需要切换为其他语言，可点击软件主界面的“选项” “语言设置”，进行语言选择设置。



4 确认所选控制器是否正确

有时同一个局域网内有多个 AM600，当选择登录了某台控制器后，可能需要试验确认所选控制器是否正确：



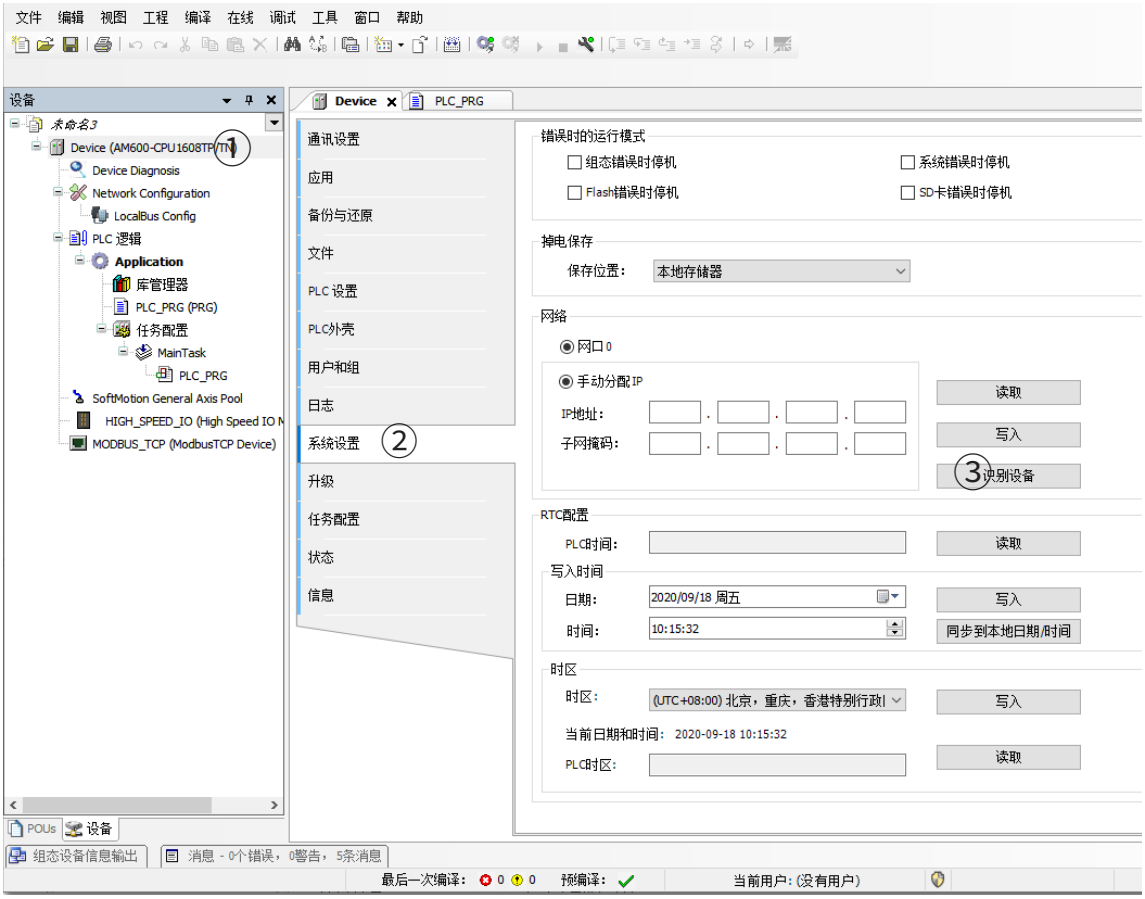
试验方法是在 InoProShop 的“Device”设备页面，选择“系统设置”标签，点击“识别设备”按钮。

① 双击 Device 设备。

② 打开“系统设置”界面。

③ 点击“识别设备”，“通讯设置”界面所选的 PLC 数码管会交替闪烁。

如下图：



1.2.5 卸载 InoProShop

使用标准 Windows 系统卸载软件方法卸载 InoProShop 即可，具体步骤如下：

- 退出 InoProShop 软件，确认 Gateway 已关闭。
- 如果操作系统任务栏存在 CoDeSys 图标，可在该图标上点击鼠标右键，选择“退出”（Exit）关闭“Gateway”。
- 选择“开始 -> 设置 -> 控制面板”（Start -> Settings -> Control Panel）。
- 双击“添加 / 删除程序”（Add or Remove Programs）。
- 选择需要卸载的软件项，找到“InoProShop”。
- 右键单击软件，选择“删除”按钮，并确认删除。

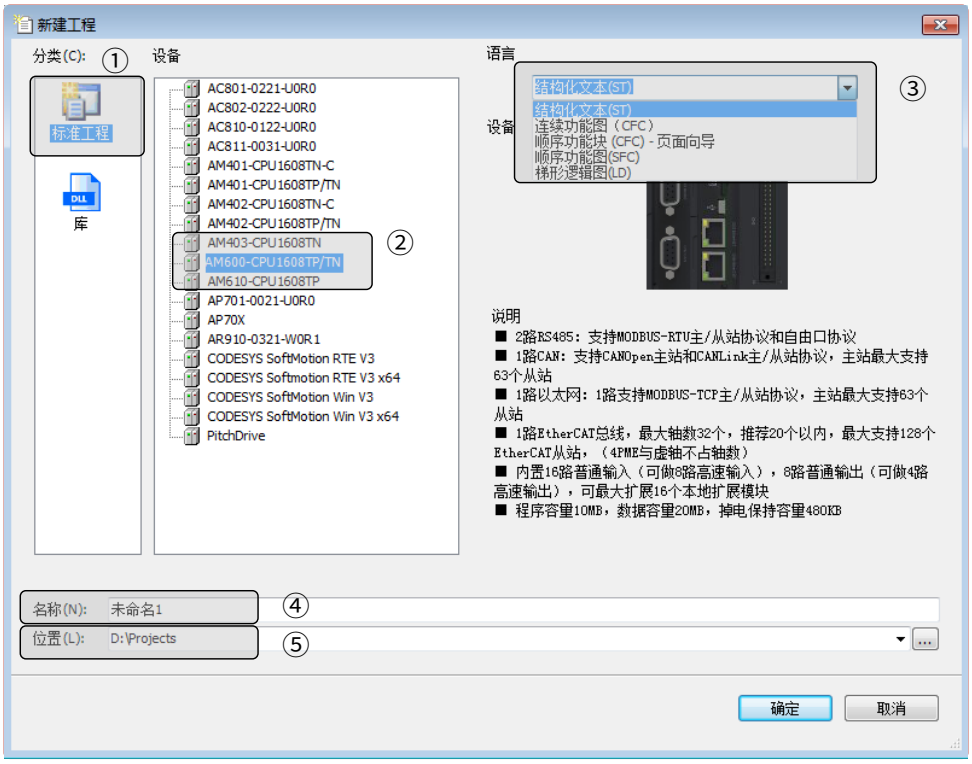
第 2 章 快速入门

2 快速入门

2.1 启动编程环境

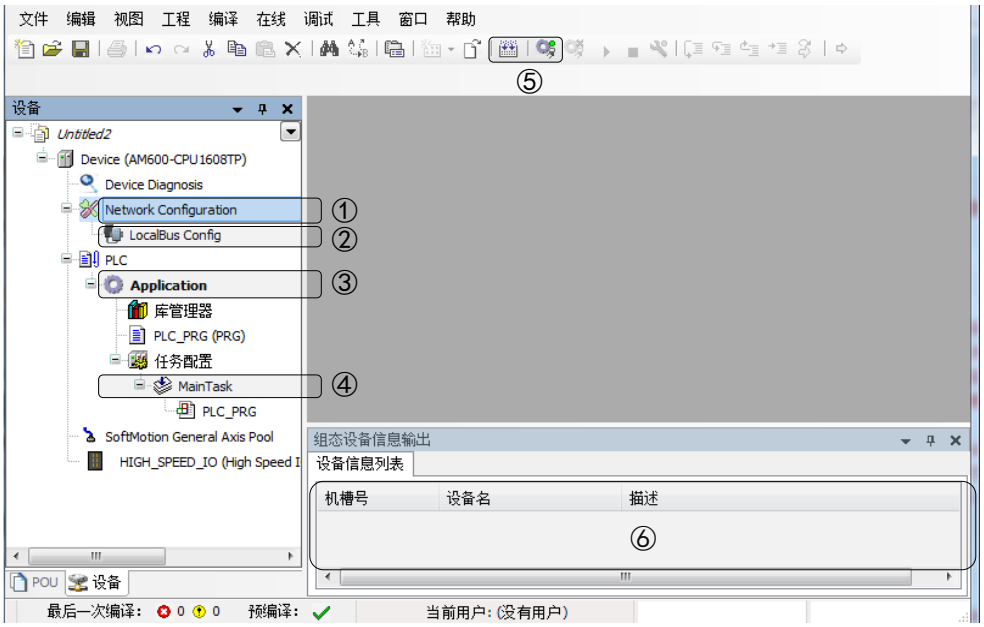
1） 双击桌面编程软件图标  即可启动 InoProShop 编程环境，起始页显示画面如下：

2） 点击菜单栏左上角新建工程或者选择“文件”-“新建工程”，选择工程类型“标准工程”选择设备类型和编程语言，并指定工程文件名及保存路径，如下图所示：



①: 选择工程类型为“标准工程”；②: 选择主模块机型；③选择熟悉的编程语言；④填写工程名；⑤选择存放路径。

3） 点击“确定”后，进入系统组态配置与编程界面，常用的按钮与窗口分布如下图：



① 网络配置

② 本地总线配置

③ 用户程序管理单元

④ 配置任务执行方式及周期

⑤ 编译、登录及调试

⑥ 设备信息窗口

2.2 编写用户程序的典型步骤

初次使用汇川中型 PLC 的用户需要注意，编写调试一个完整的用户程序需要 5 个步骤。

1） 基于中型 PLC 应用系统的硬件连接架构进行硬件系统配置。

■ 若只用了 CPU 主模块和 IO 扩展模块，只需进行硬件配置：即根据实际选用的模块类型和型号、安装顺序，在 InoProShop 的硬件配置页面把这些“元件”放进“主机架”；

■ 若用到了扩展机架，需要先配置网络总线，再根据扩展机架数量，增置对应数量的网络扩展模块，然后给每个机架放置扩展模块硬件；

2） 根据应用系统的控制工艺编写用户程序。用户程序编程基于数据的存储宽度、使用范围来自由定义变量，可以与硬件配置无关；

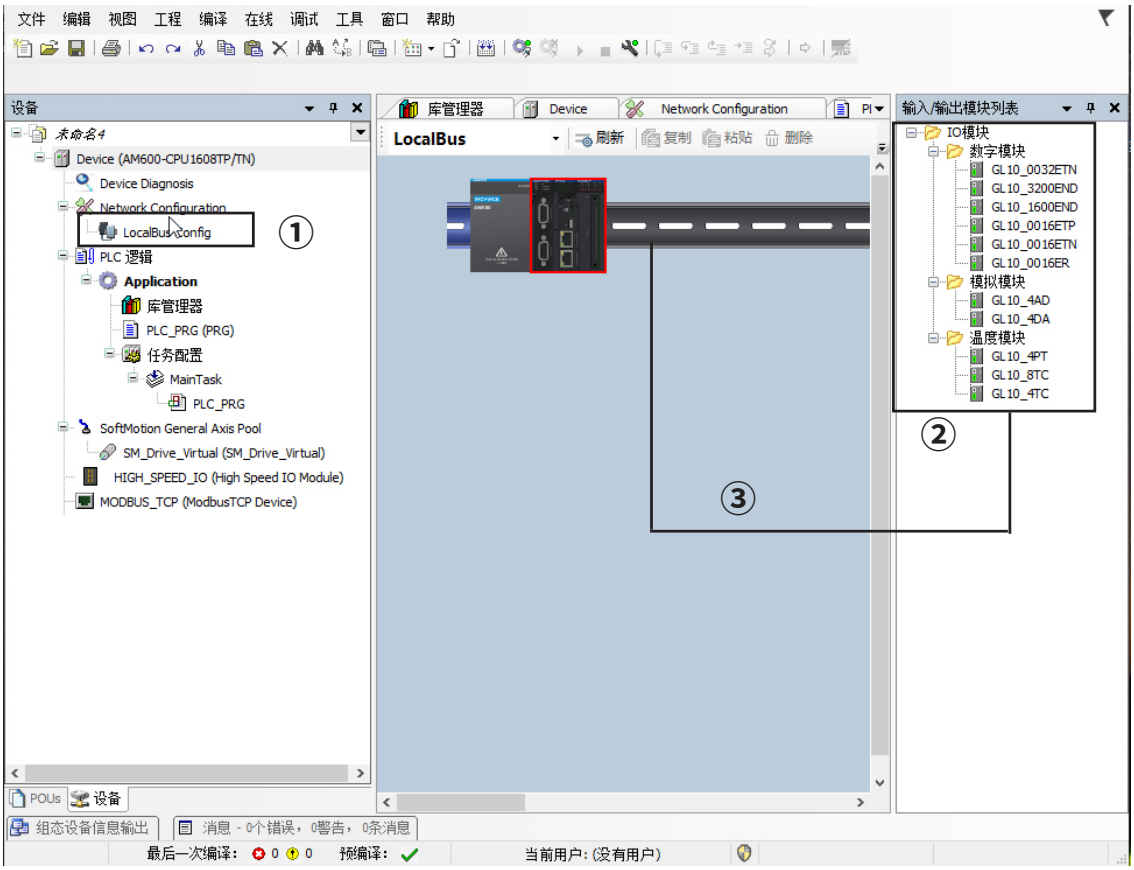
3） 将系统架构中的各硬件端口对应的输入端口变量（I）、输出状态（Q）或数值（M）与用户程序中的变量进行关联；

4） 配置网络通讯的同步周期（如 EtherCAT 总线）。根据各任务的实时性要求，配置用户程序单元的执行周期；

5） 在 InoProShop 编程环境下登录中型 PLC，下载用户程序，仿真调试、排错，直到正确无误地运行。

2.2.1 用户系统的配置操作

在 InoProShop 的主画面，双击左侧设备树中的“LocalBus Config”项，进入 PLC 主机架的硬件配置画面：



①双击，进入本地扩展模块配置界面；

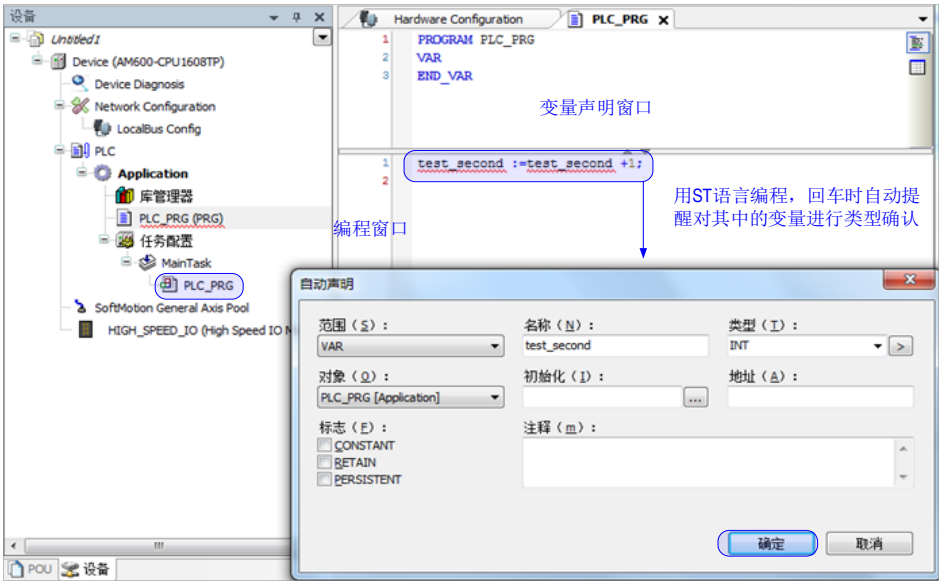
②扩展模块元件库；

③安装槽上选中 CPU 单元右侧位置，在扩展模块元件库，双击选中所需的 IO 模块，依次摆放。根据实际应用系统使用的模块型号、安装顺序，从右侧的扩展模块库中依次双击选中模块，将其拖放到“安装机架”上；若要删除某个模块，选中该模块后按 Del 键可以删除。

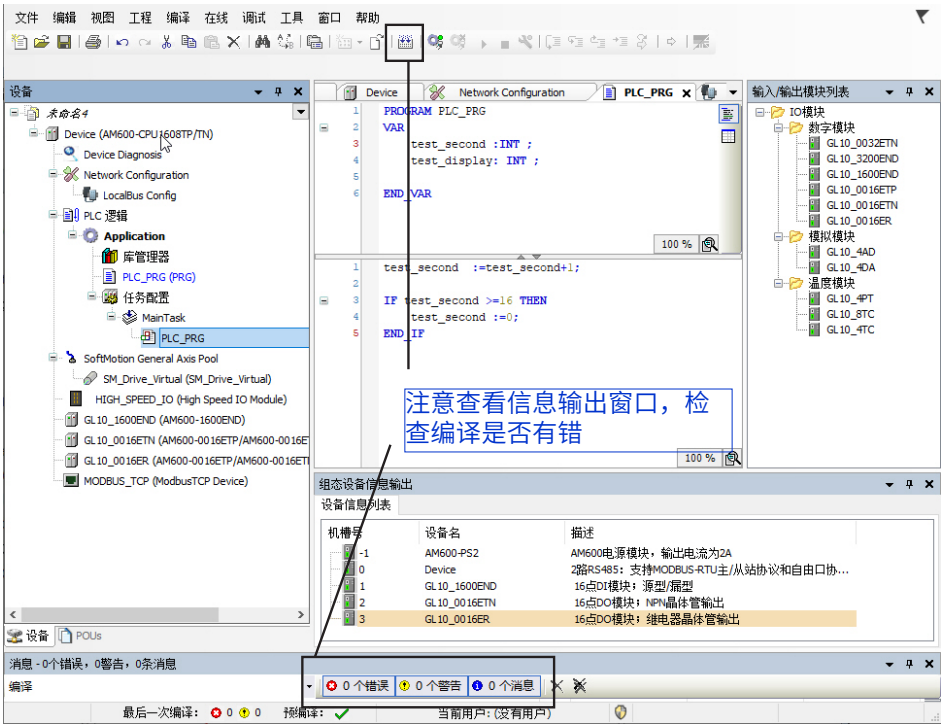
以 AM600 为例，主机架上最多可以接入 16 个扩展模块，其中模拟量模块可以接入 8 个。

2.2.2 用户程序的编写操作

双击左侧设备树窗口中的“PLC_PRG(PRG)”项，即可打开用户编程界面，编程语言为 ST（新建工程时选择），如下图所示。与 C 语言编程相似，每个变量需要声明后才能使用；如果先直接编写程序语句，回车时编程环境会自动弹出声明框; 经用户填写并点击“确定”后，变量声明窗口会自动增加该变量的声明语句，这样简化了编程:

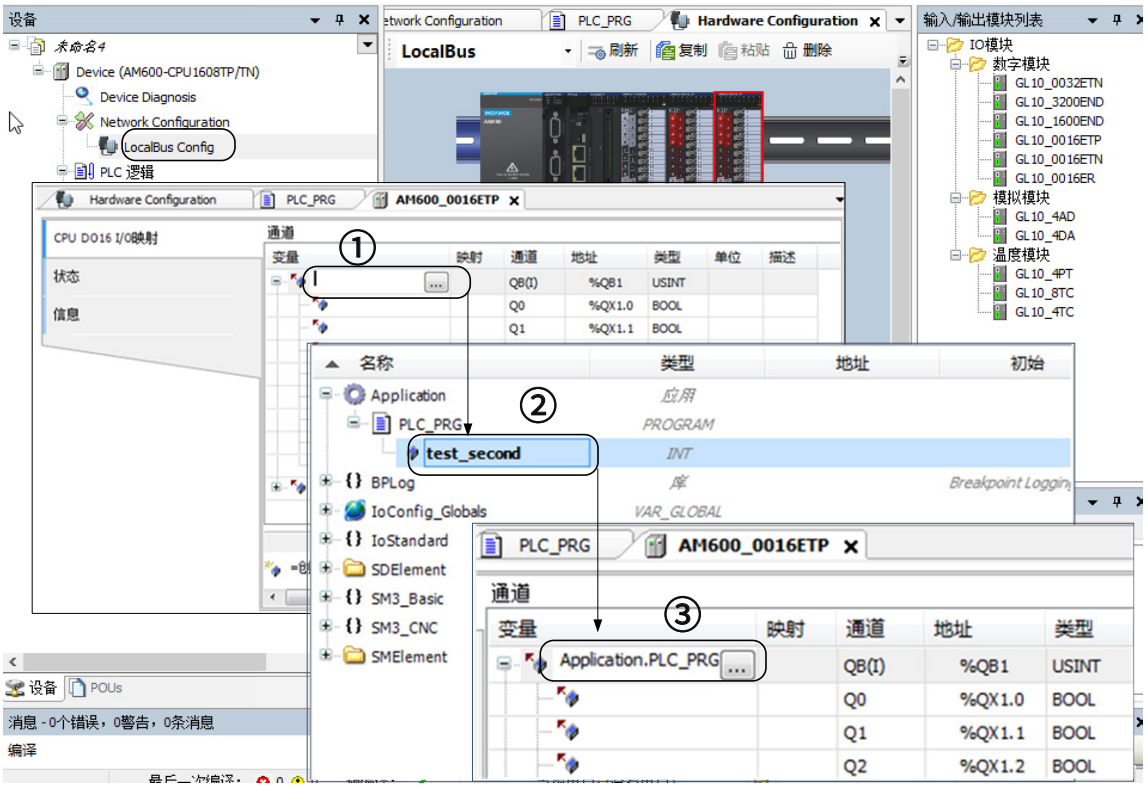


编写一个简单示例：将第二个变量赋值给第一个变量，然后递增：



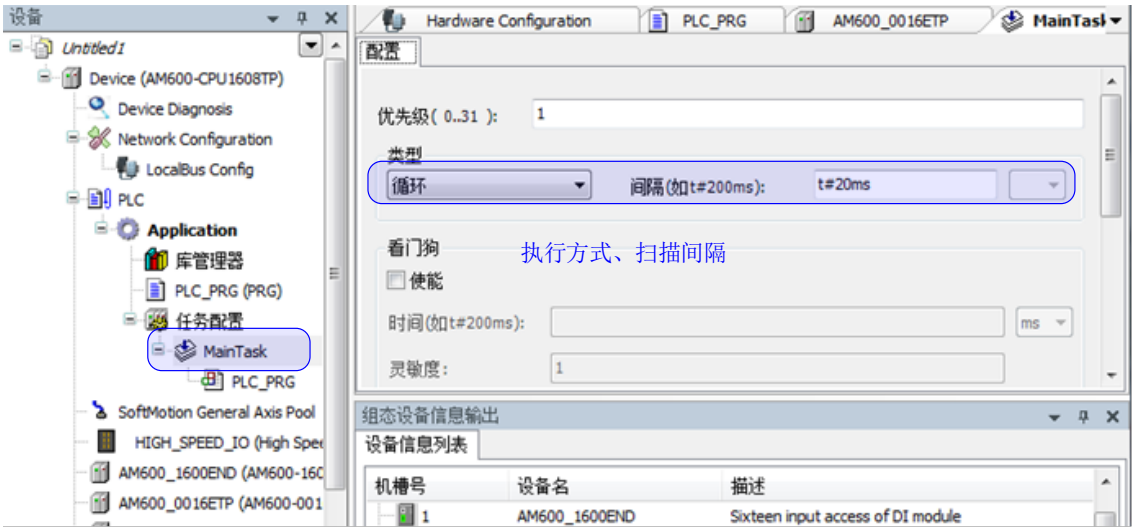
2.2.3 用户程序变量与端口的关联配置

在本地总线配置页面，将需要关联的硬件端口与用户程序中的变量进行关联。如下图所示，将“test_display”的变量值，关联到第一个 DO 模块的输出端口，配置步骤如下：



2.2.4 配置用户程序的执行方式和运行周期

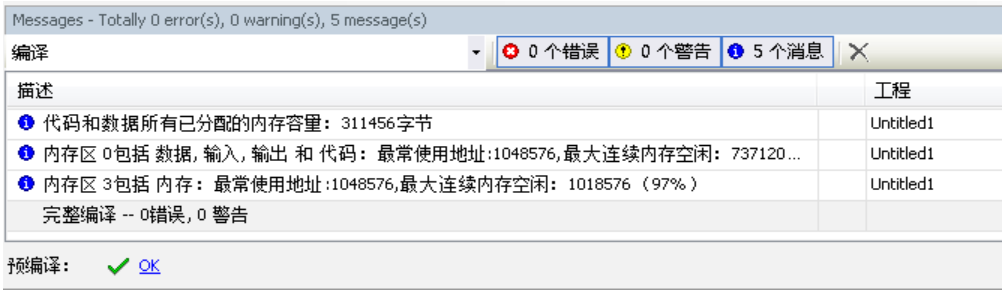
上文示例中编写的子程序默认为 20ms 执行一次，如要改为其他的执行方式，如反复执行、定时执行、执行周期等等，可以按下图所示分别设置：



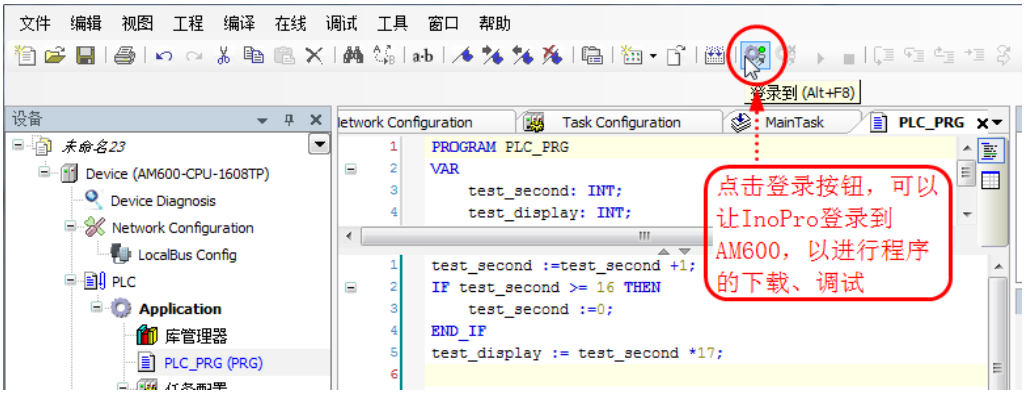
2.2.5 用户程序的编译、登录下载

完成上文的程序编写后编译程序，查看是否有错；若有错，点击错误信息行可定位到用户程序的报错点，方便修改，直到错误全部排除。

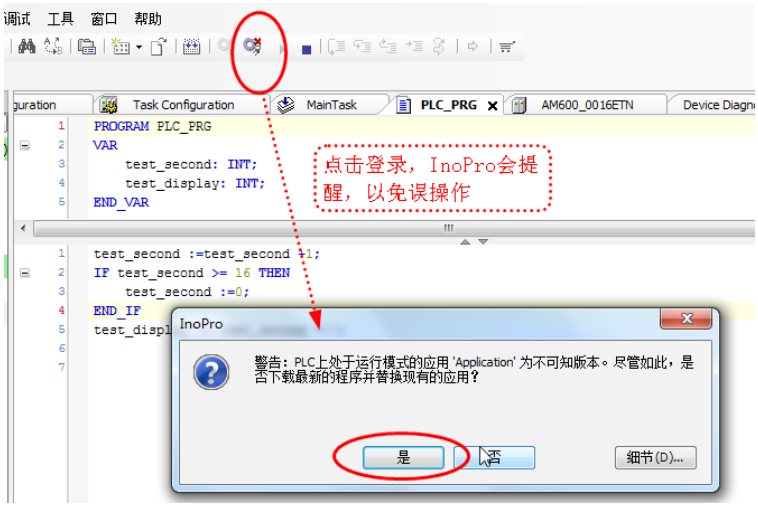
相关编译信息会显示在如下编译信息框中：



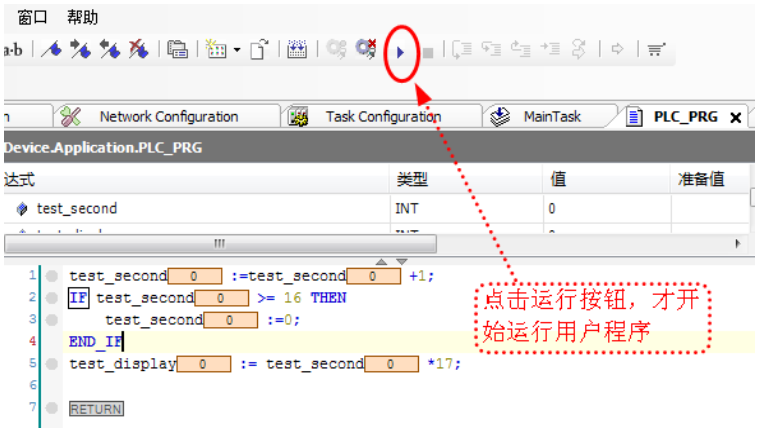
编译无误后，点击“在线”-“登录到”，如下图所示：



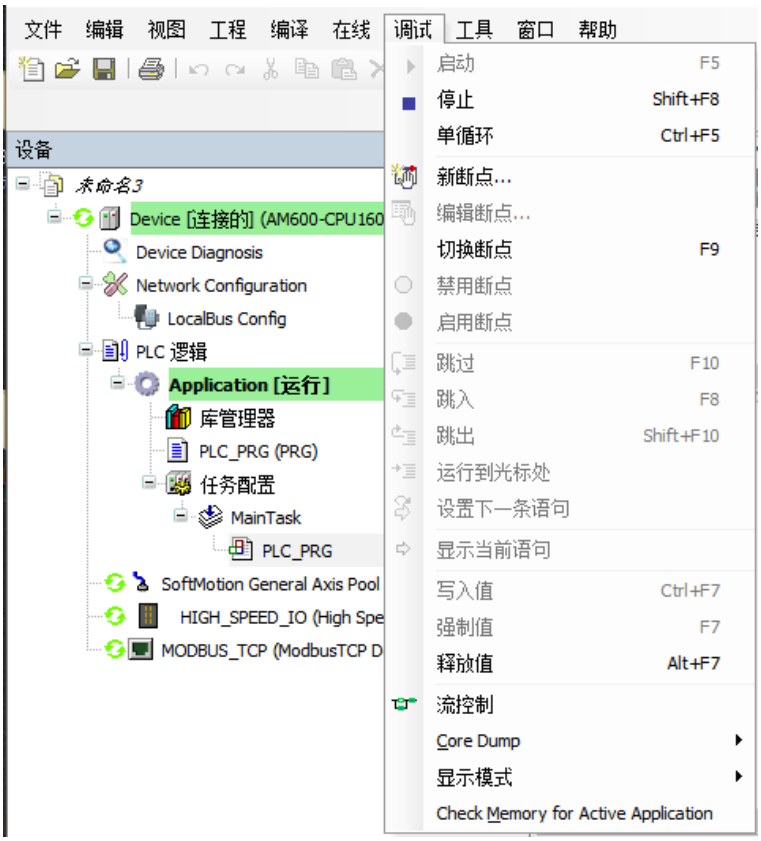
然后弹出如下对话框，选择是否创建程序并继续下载：



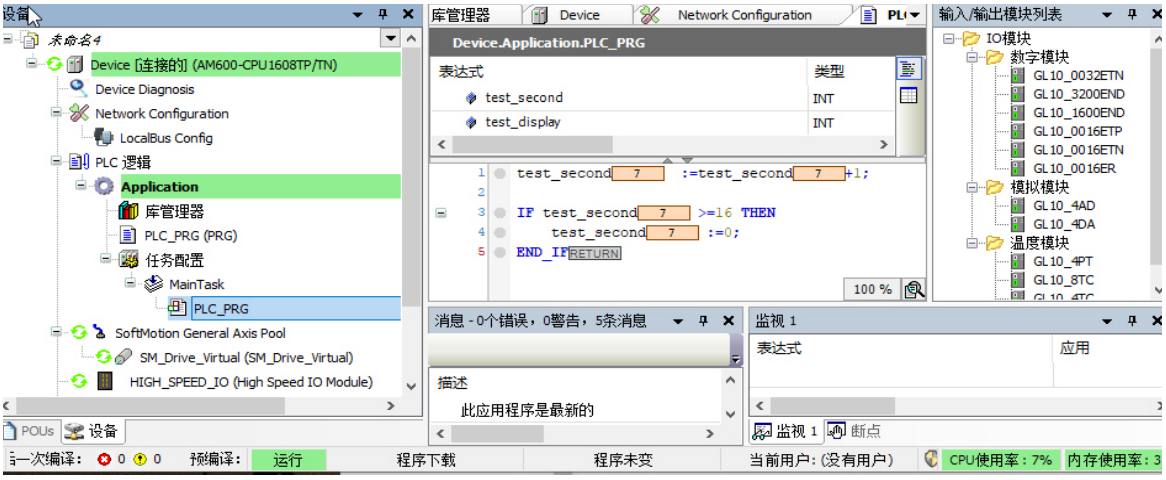
选择“是”，上位机与设备建立连接并保持，初始状态为“停止”，如下图：



点击“调试”-“启动”，设备进入运行状态，并开始执行用户程序。



下图为正在运行的用户程序监控画面：



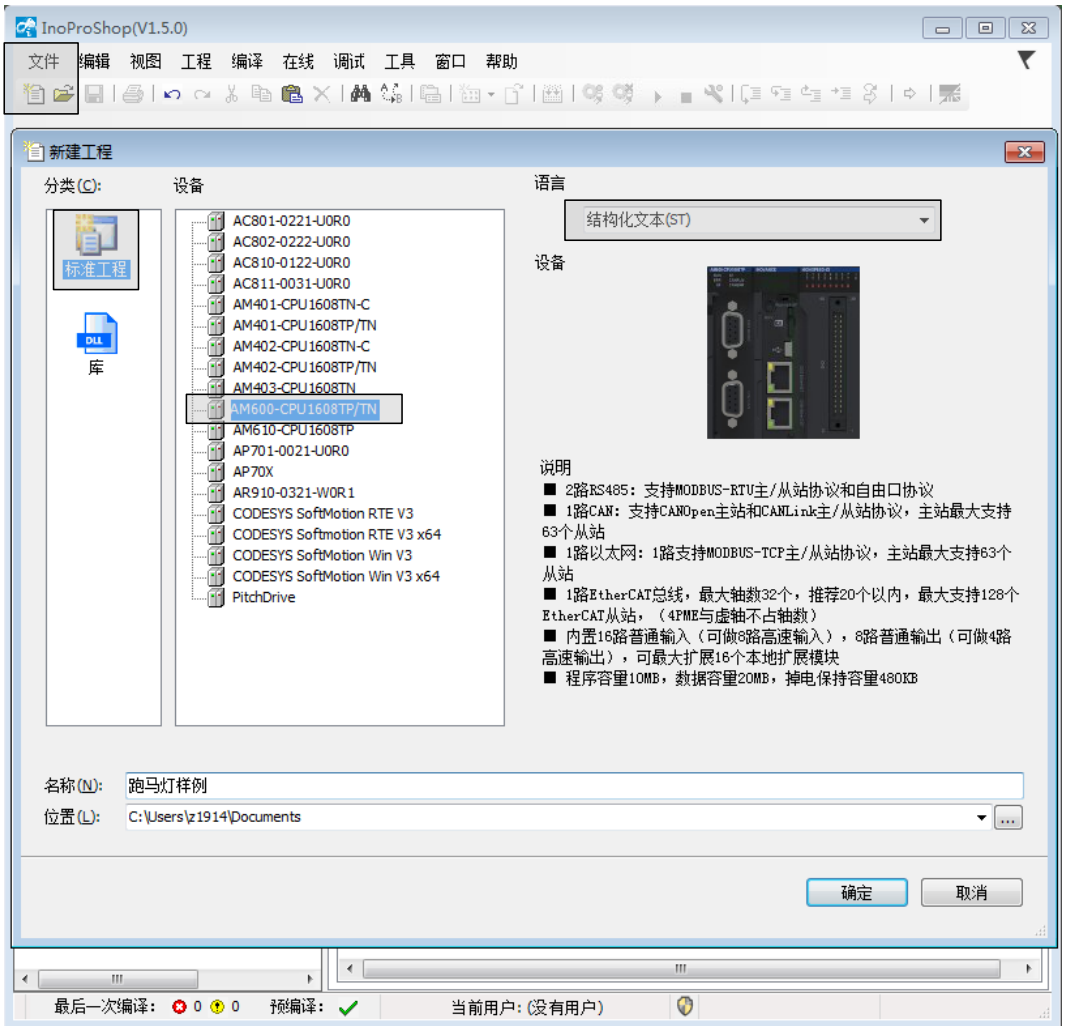
此时查看 AM600 后面的第一个 DO 模块，可以看到其输出状态指示灯以二进制计数方式循环计数。

2.3 用 InoProShop 编写一个跑马灯样例工程

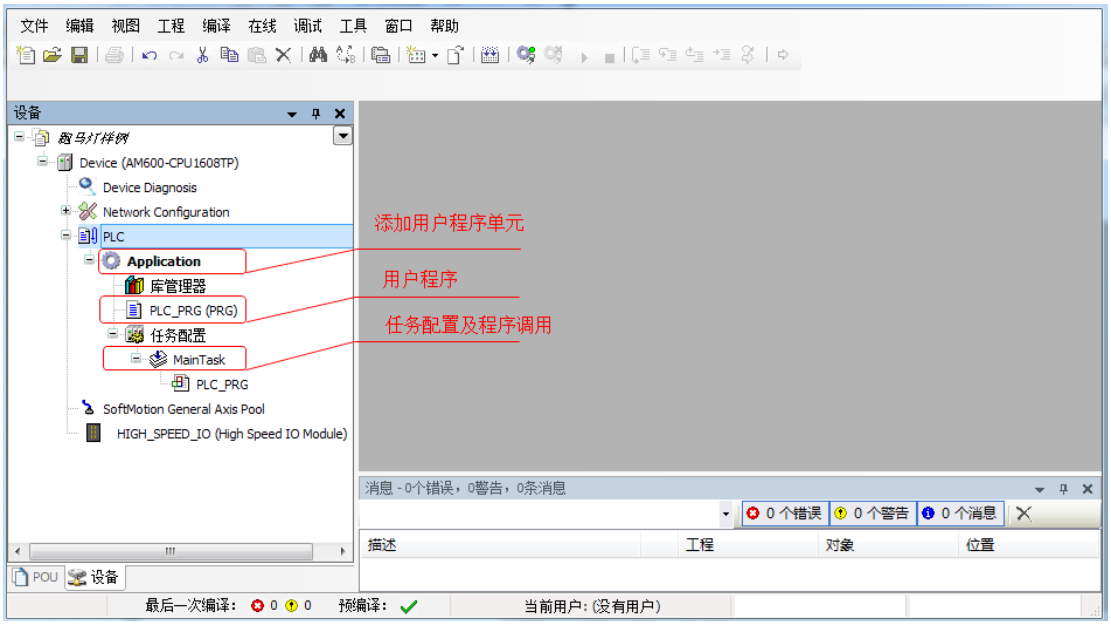
1 启动 InoPro 编程环境

新建工程：

点击菜单栏左上角新建工程或者“文件”-“新建工程”，选择工程类型为“标准工程”，选择设备类型（选择主模块的机型）和编程语言，指定工程文件名及保存路径，如下图所示：

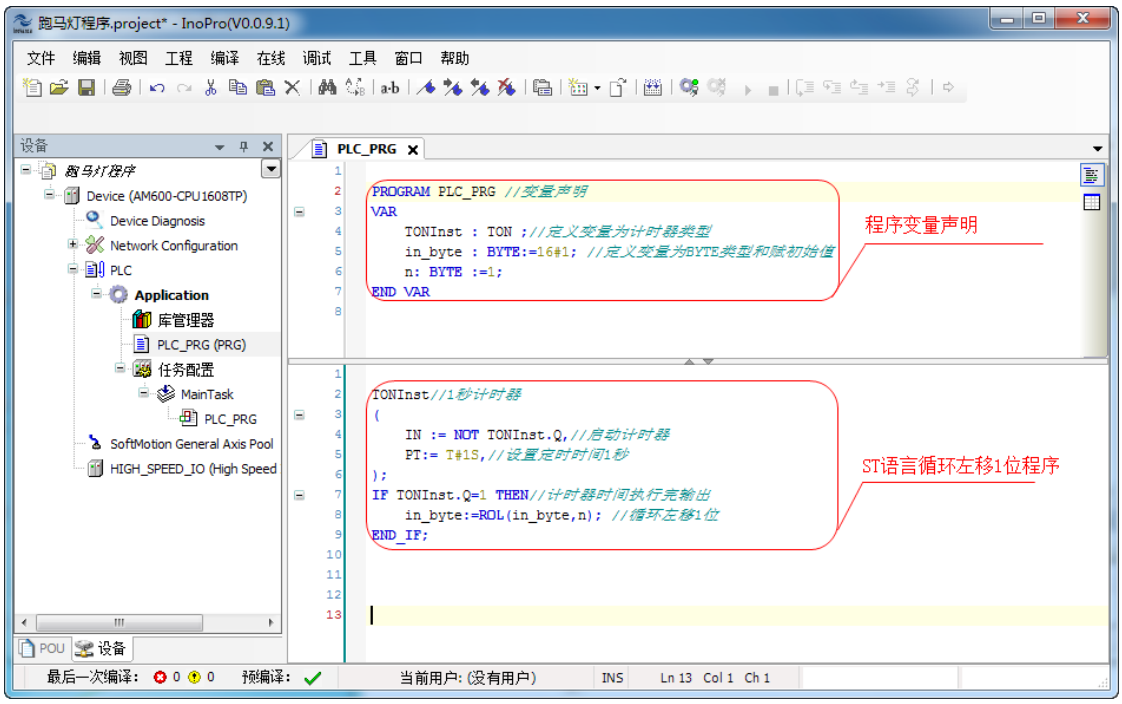


2 系统组态配置与编程界面



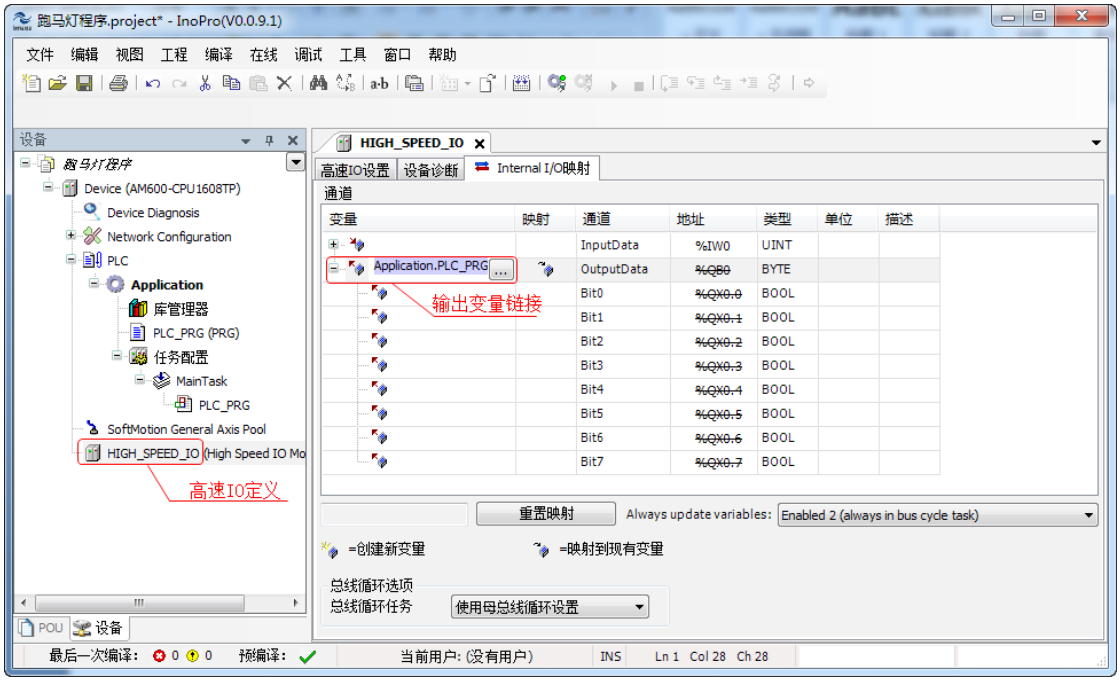
3 ST 语言编写“跑马灯样例程序”

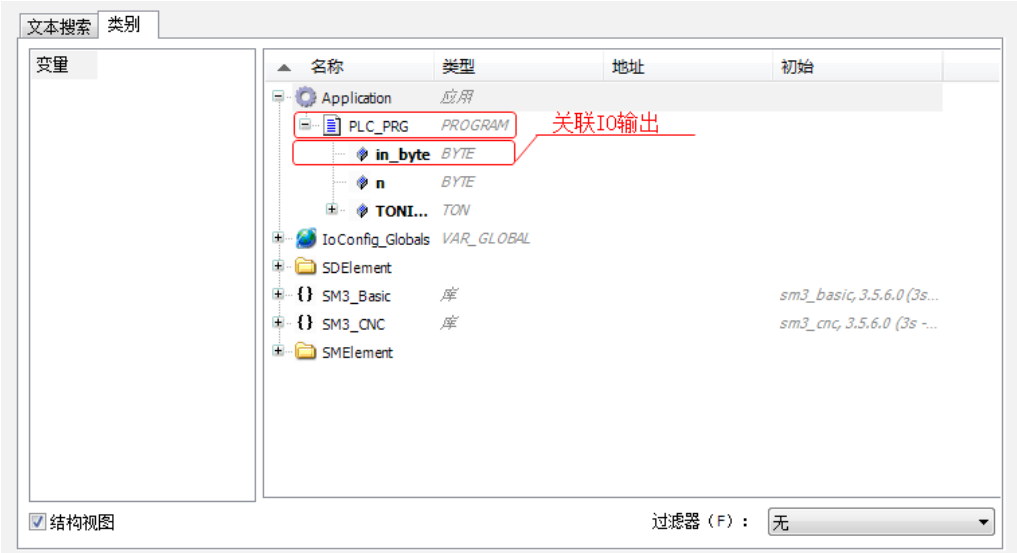
双击打开“PLC_PRG”程序组织单元



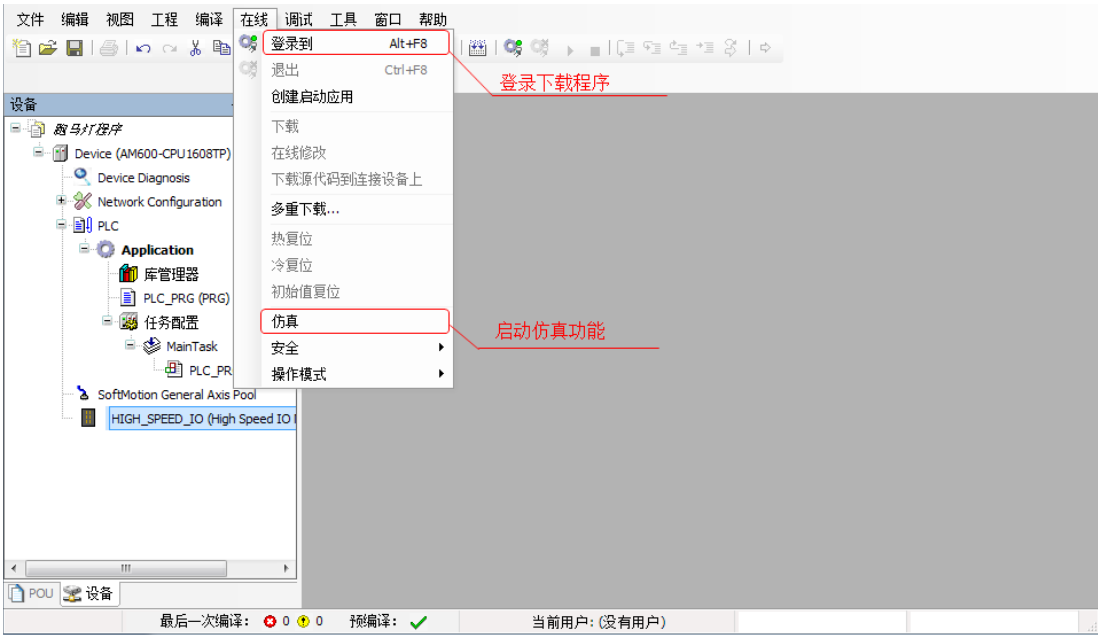
4 关联 PLC 输出 IO

循环左移 “in_byte” 变量与 PLC 自带的 8 路输出端口链接（Bit0-Bit7），观察输出灯的变化





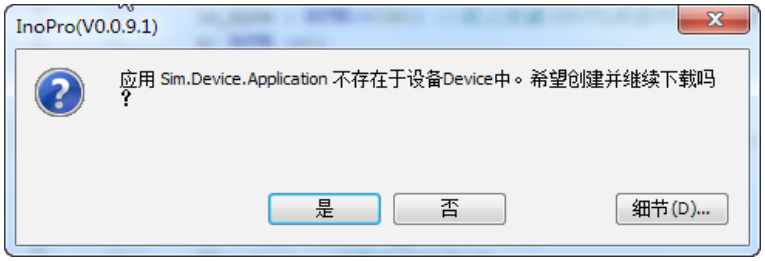
5 仿真调试



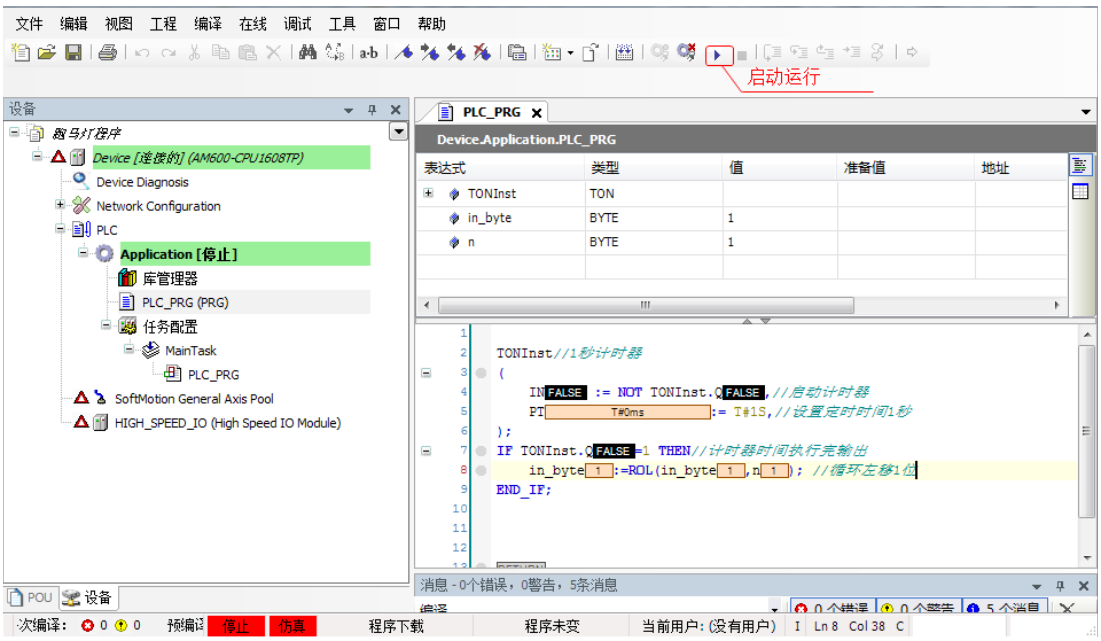
点击“仿真”进入仿真功能，这时不需要链接 PLC 亦可观察 IO 移位状态；

6 在仿真模式下下载程序

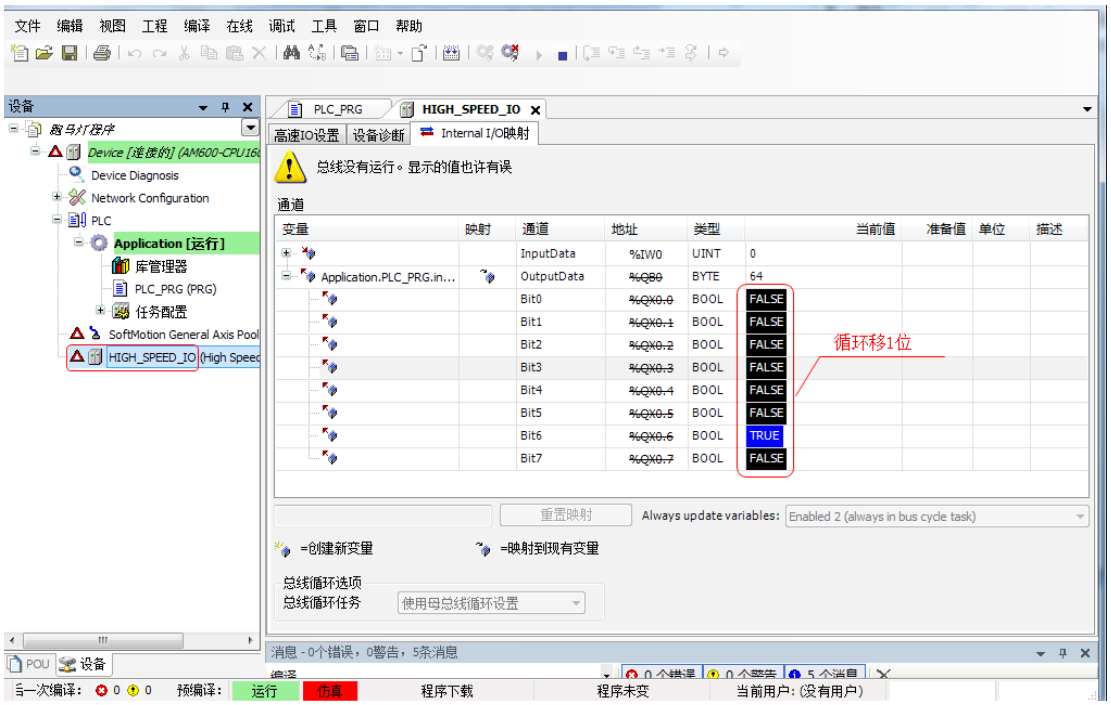
点击“登录”，在仿真模式下下载程序



7 下载完成后，启动运行 PLC



8 监控 IO 变化



2.4 如何登录主模块

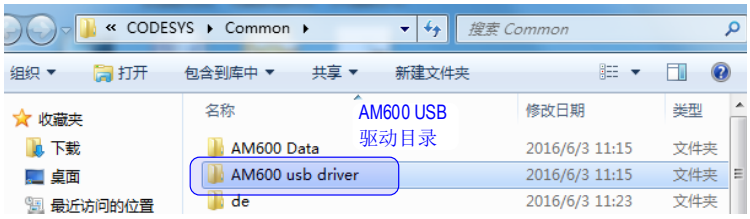
2.4.1 登录主模块的必备条件与操作简介

“登录主模块”是指在 PC 上运行的 InoProShop 与中型 PLC 主模块建立通讯，从而进行用户程序的运行、下载、启停和监控，以及程序参数查看或修改操作等等。

- 目前有两种方法登录中型 PLC：通过 LAN 局域网；通过 USB 连接。
- PC 与中型 PLC 之间可以通过网线进行 1 对 1 直连；也可以通过路由器、集线器连接 PLC，这种情况下，一台 PC 可以与多台中型 PLC 联机，也可以多台 PC 访问同一个中型 PLC；
- PC 与中型 PLC 两者的 IP 地址必需在同一个网段才能登录，否则 InoProShop 将无法扫描到中型 PLC。比如，AM600 的出厂默认 IP 地址为 192.168.1.88，若 PC 的 IP 地址为 192.168.1.xxx（这里 xxx 范围为 1~254，但不得与 AM600 的 IP 相同），那么 InoProShop 就可以扫描到 AM600 并与之交互数据，进行用户程序下载、运行监控等。若 AM600 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，此时需要将 AM600 的 IP 地址恢复为出厂地址 192.168.1.88，再将 PC 本机的地址修改为 192.168.1.xxx，二者建立 1 对 1 联机后，将 AM600 的地址修改为所需的 IP 网段地址。
- 如果通过用户想通过 USB 登录 PLC，插上 USB 连接线（MiniUSB 口），等待 20s-60s 就可以扫描到设备。

USB 连接注意事项：

1） USB 驱动在安装软件时会自动安装，如果没有自动安装，可以在安装目录中的 Common 文件夹下找到，如下图。



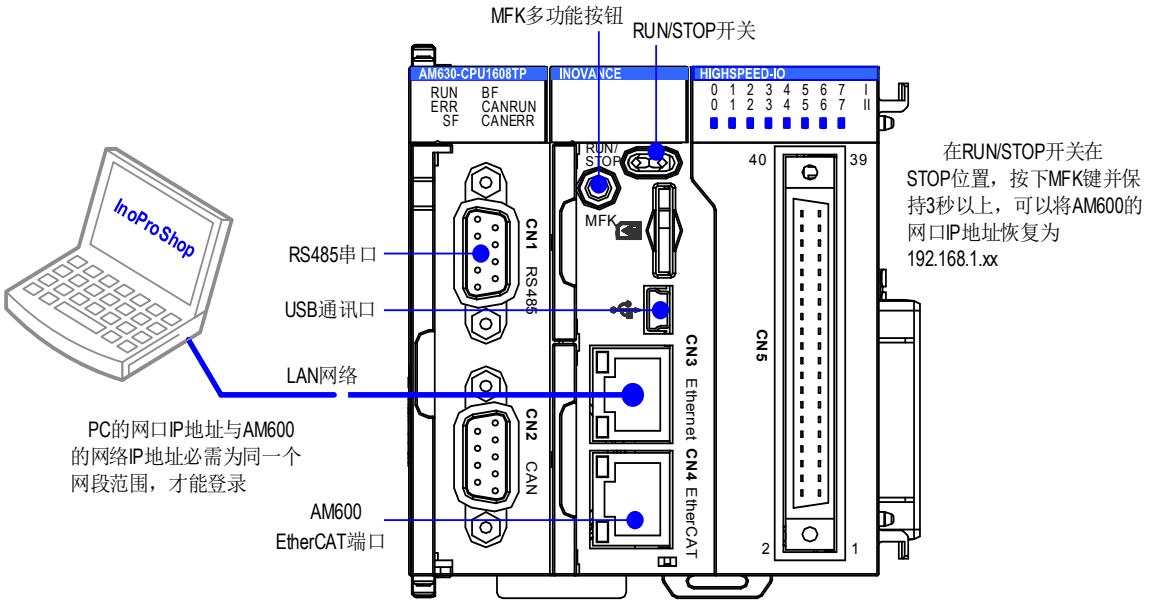
然后通过 Windows 设备管理器更新驱动，从安装目录中安装驱动，USB 连接成功后，Windows 设备管理器

显示如下图。

2） 若 USB 连接和网络连接同时存在，默认使用网络连接（网络扫描速度更快）。

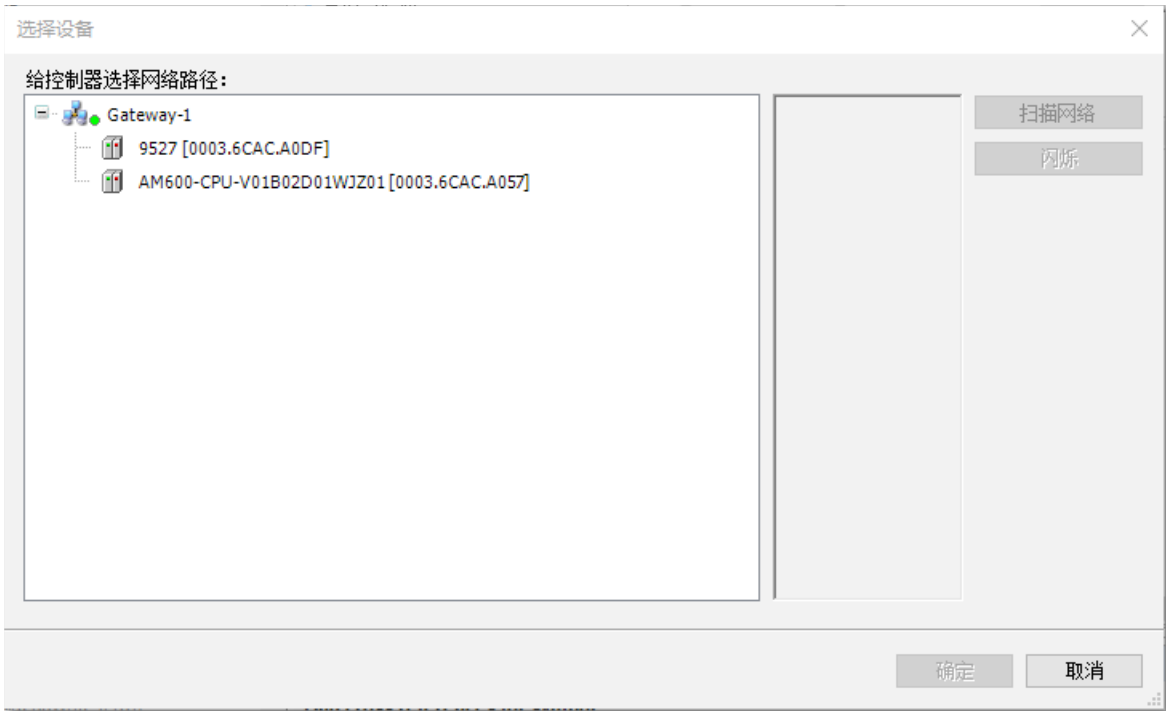
2.4.2 在 InoProShop 中扫描中型 PLC 网络设备

PC 可通过 LAN 网络登录中型 PLC，以 AM600 为例，连接方式如下：



在 InoProShop 中，双击 Device(AM600-CPU-1608TP/TN)，弹出如下界面：

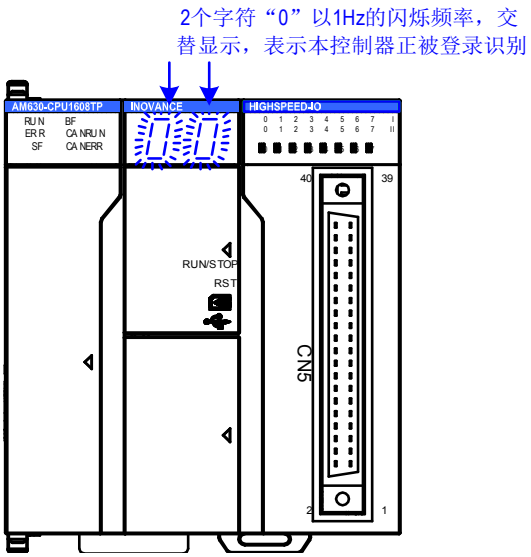
在该界面点击“扫描网络”标签，弹出如下界面，在窗口左侧点击其中的 AM600-CPU 控制器，即可在窗口右侧可以看到其简介信息：



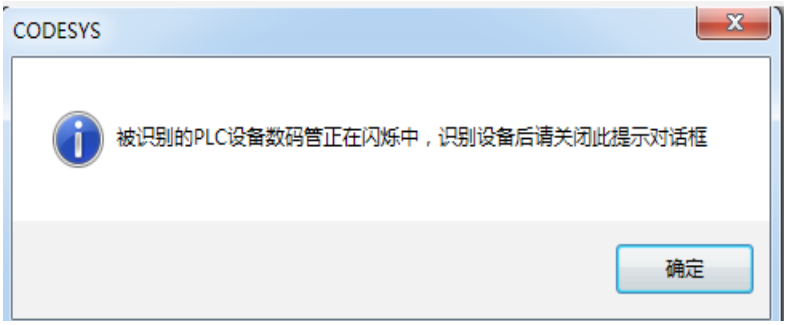
上例图中找到了 2 台控制器，分两行显示：

- 第 1 台 9527[0003.6CAC.A0DF]：网段内的设备，名称 9527，括号中数字的最后 2 位“DF”为该 AM600 的 IP 地址第 4 个段位，为 16 进制显示，转换为十进制为 223。
- 第 2 台 AM600CPU-V01B02D01WJZ01[0003.6CAC.A057]：网段内的另一个设备，其设备名为 AM600CPU-V01B02D01WJZ01。用户登录后可以根据需要自行修改设备名，这样在有多台控制器的应用场合方便识别。

此时，登录的 AM600 或 AM610 上的两位数码管，将交替显示字符“0”，如下图：

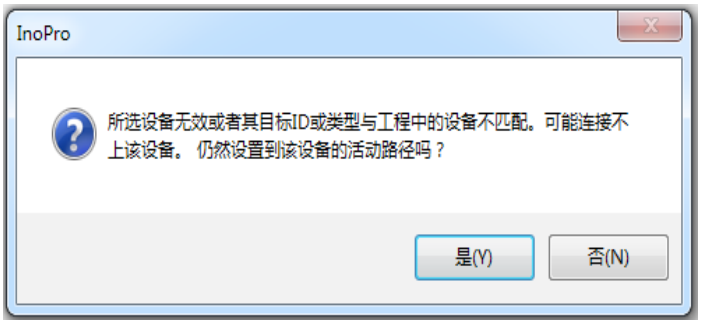


直到用户点击 InoProShop 中弹出窗口的确认按钮后，才停止闪烁，恢复原有的显示信息：



双击选中的设备，或者选中设备后再点击“确认”即可激活上位机与当前设备的连接。

若当前的工程中记录的控制器标识号与所选择的控制器不符，可能提示以下信息，若要联机，点击“是”确认即可。

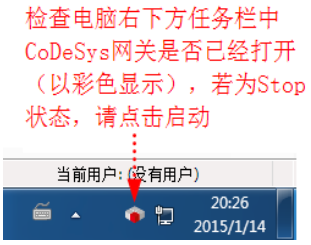


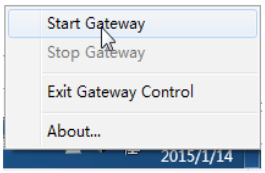
2.4.3 扫描不到设备的处理对策

如果在 InoProShop 中扫描不到 AM600 设备，可能原因和对策如下：

1） CoDeSys 网关没有启动。

解决方法：重新启动网关后，再进行扫描。

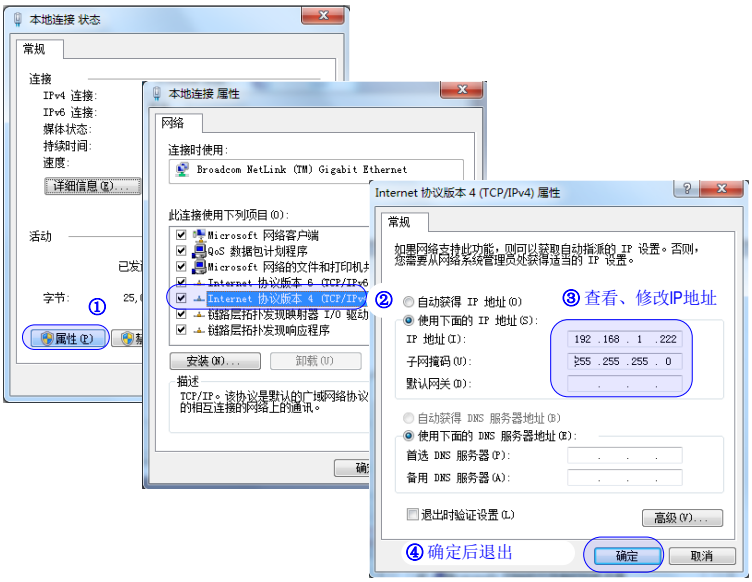




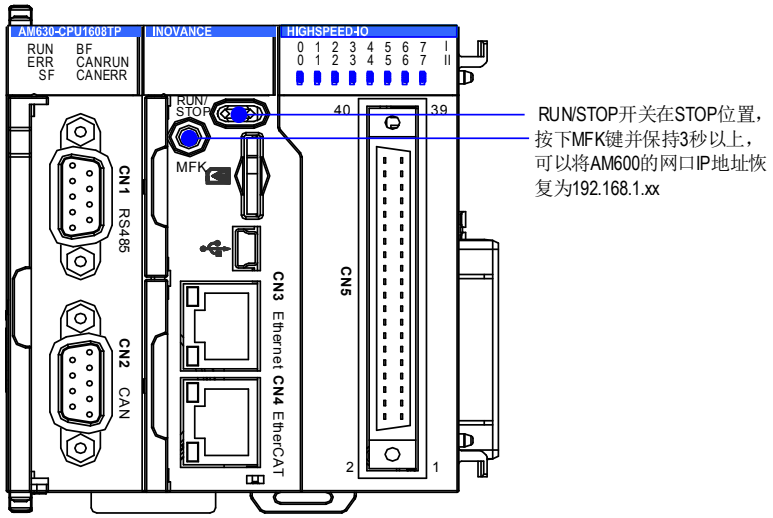
2） PC 的 IP 地址与 AM600 的 IP 地址不在同一个网段。

解决方法：将 PC 的 IP 地址设置到 AM600 IP 地址的同一网段。如果忘记了 AM600 的 IP 地址，建议将其恢复为默认 IP，再将 PC 的 IP 地址设置为 192.168.1.xxx 网段，即可正常扫描设备。

■ 首先在 PC 的资源管理器中，点击网络的本地连接，按下图进行 IP 地址的检查和修改：



■ 其次恢复 AM600 的出厂默认 IP 地址（192.168.1.88）。AM600 上电启动后，将 RUN/STOP 拨到 STOP 位置，按下 MFK 多功能按键保持 3 秒以上，AM600 会将网口 IP 地址恢复为默认值 192.168.1.88，恢复前会有“I.P.”、“10”、“9”、、、倒计时提醒，若不希望恢复 IP 地址，可以在倒计时为 0 之前，再按 MFK 键放弃。

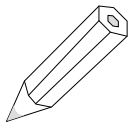


无论是复位 AM600 的默认 IP，还是通过 InoProShop 后台软件修改 AM600 的 IP 地址，都会使修改后的 IP 地址立即生效。

一旦扫描联机成功，在设备扫描画面可看到如下网络状态信息：

Memo NO. _____

Date / /



第 3 章 网络配置

3 网络配置

3.1 设备组态

设备组态是用户进行 PLC 编程的第一步，具体包括“网络组态”和“硬件组态”两个功能，用户可以通过这两个功能来对设备进行布局。

■ 网络组态

从总线型的网络拓扑视角设计，是设备组态的入口。

■ 硬件组态

添加中型 PLC 扩展 IO 模块。

3.1.1 网络组态

新建一个 InoProShop 工程后，双击软件左侧设备树中的”Network Configuration”的节点， 如图所示：

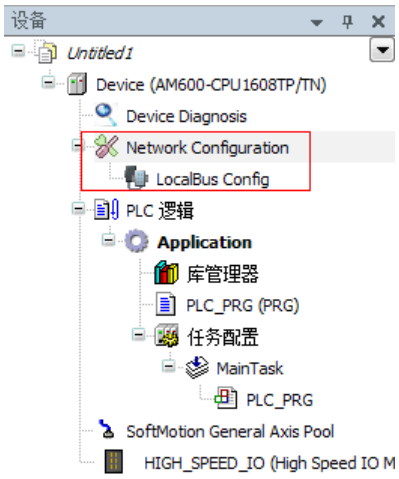


图 3-1 网络组态节点

双击节点后，打开网络组态“Network Configuration”的界面以及“网络设备列表”（图 3-1），网络组态界面显示用户工程当前所使用的 PLC 设备，而”网络设备列表”显示当前 PLC 支持的所有设备。

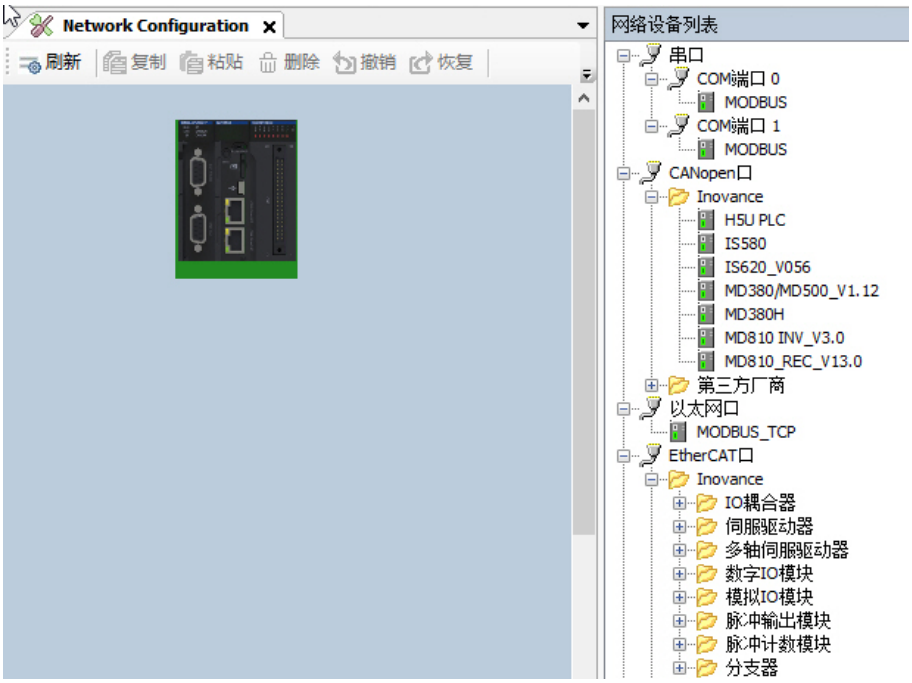
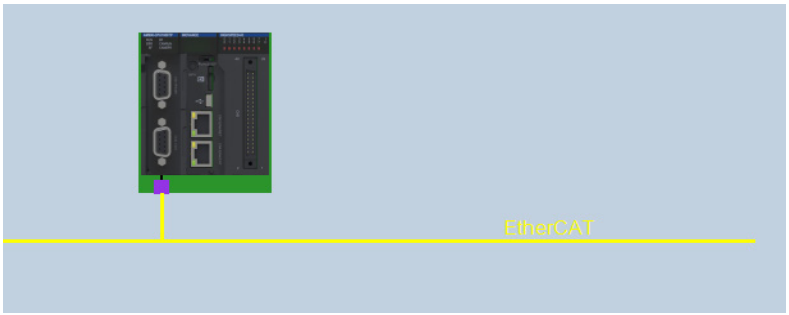


图 3-2 网络组态“Network Configuration”

1 设置 PLC 作为主站或从站设备

单击网络组态中的 PLC 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。

除 CANlink 主站外，使能 CPU 的其他特定主站功能时，都会显示总线型的拓扑界面。例如下图所示为使能 EtherCAT 主站：



■ 添加从站设备

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，然后从图 3-1 所示的网络设备列表中的 "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。

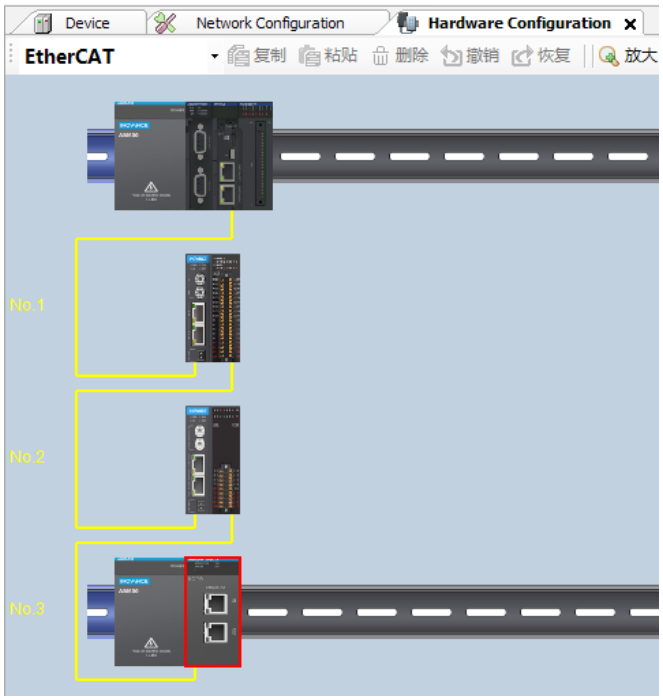
2） 先使能 EtherCAT 主站功能，然后从图 3-1 所示的网络设备列表中的 "EtherCAT 口 " 节点下双击一个从站

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。

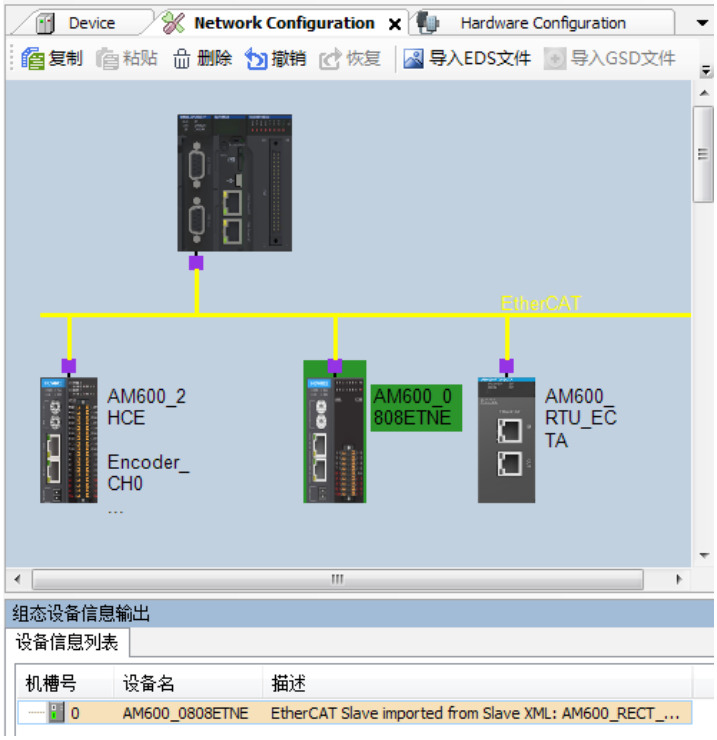
4） 如果添加 EtherCAT 分支器设备，请参见章节 3.3.9.2。

添加完从站，如果想配置从站后的 IO 模块 (针对 AM600 系列通讯从站)，则双击设备即可进入硬件组态配置”Hardware Configuration”界面进行配置。



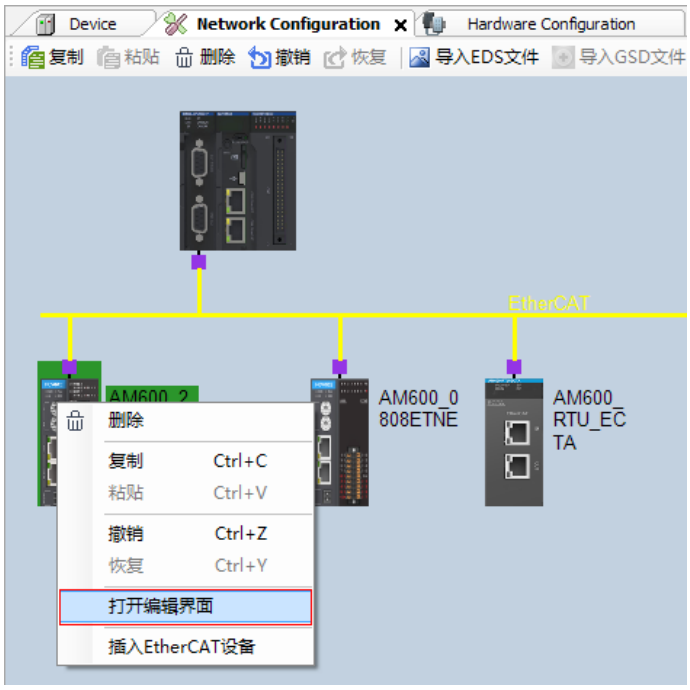
■ 查看设备基本信息

选中网络组态界面中的设备后，可在“组态设备信息输出”-“设备信息列表”中找到对应的设备基本信息。



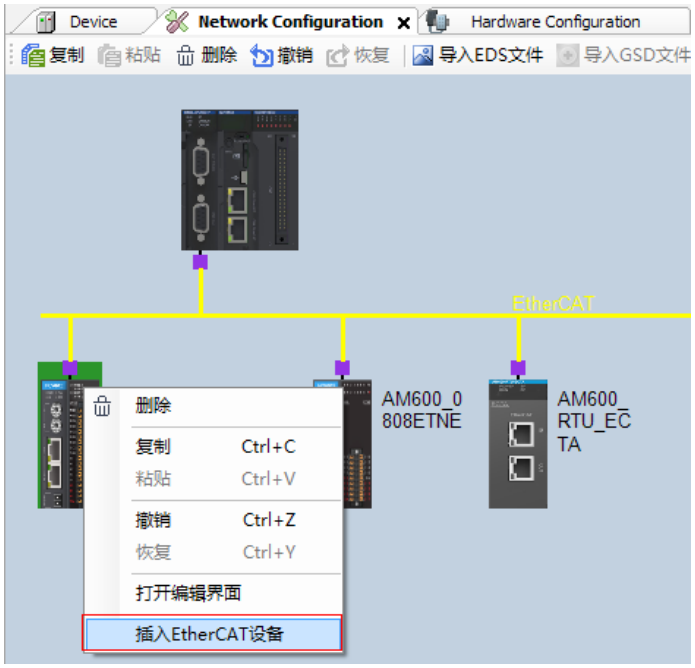
■ 打开设备配置界面

右键单击网络组态中的 EtherCAT 从站，通过 " 打开编辑界面 " 选项进入设备的配置界面。如图所示：



■ 插入 EtherCAT 从站

右键单击网络组态中的 EtherCAT 从站，通过 " 插入 EtherCAT 设备 " 选项插入一个 EtherCAT 从站设备，如图所示：



■ 组态设备可进行复制、删除、添加等操作，详情请参见组态设备的常用操作。

■ 编译组态

当网络组态中有两个或多个 Modbus 从站设备站号相同，或 ModubsTCP 从站设备的 IP 地址相同，编译工程时会显示在编译信息输出框中，详情请参见组态编译错误定位。

2 设备信息列表

设备信息列表通过菜单项 " 视图 "->" 组态设备信息视图 " 打开，用于显示组态设备的基本信息，包括机槽号、设备名称、描述。如果默认最小化在软件底部 (消息 - 0个错误，0警告，0条消息 | 组态设备信息输出)，需要手动点击打开列表。

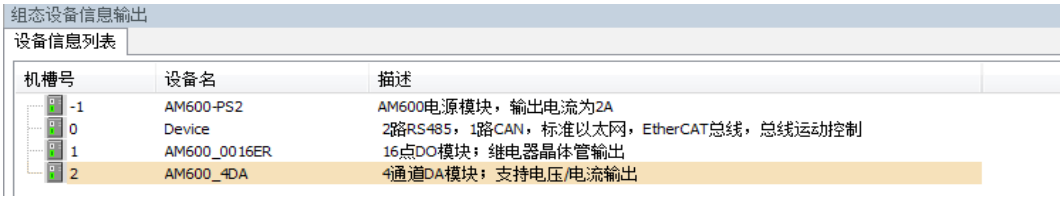


图 3-3 设备信息列表

■ 机槽号

对应硬件组态中的设备插槽，无论是主机架上的模块或是通讯从站后的模块，其槽号都是从 1 开始。-1 号机槽对应 AM600 电源模块，0 号机槽对应 CPU 模块。

■ 设备名

与软件左侧设备视图中所示的设备名称一致。

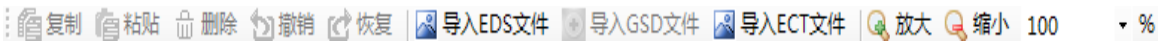
■ 描述

设备的基本描述，包括设备基本工作指标和功能。

单击设备列表中的某行可以定位组态界面中的对应设备，双击某行也可打开对应的设备组态界面。

3 组态设备的常用操作

组态设备基本操作包括设备的复制、粘贴、撤销、恢复、删除、导入 EDS、GSD、ECT 文件、放大、缩小功能。



NOTE

1) 设备复制、粘贴、删除、撤销、恢复操作在硬件组态界面只针对 IO 模块；在网络组态界面只针对从站；
2) 如果对网络组态中的从站进行复制、粘贴、删除，则后面的模块也会被相应操作；
3) AM600-CPU1608TP 型号 CPU 支持 EDS 和 ECT 文件导入，不支持 GSD 文件导入；AM610-CPU1608TP 型号 CPU 支持 GSD 文件导入，不支持 EDS 和 ECT 文件导入。

■ 导入 EDS: 网络设备列表中默认带部分 CANopen 设备和 EtherNet/IP 设备，如果要加入其他 CANopen 设备或 EtherNet/IP 设备，需要导入其对应的标准 EDS 文件。导入成功后，该设备会添加到网络设备列表中，如果是汇川设备，会显示在 Inovance 节点下，否则会显示在 " 第三方厂商 " 节点下。

■ 导入 GSD：网络设备列表中默认带部分 DP 设备，如果要加入其他 DP 设备，需要导入其对应的标准 GSD 文件。导入成功后，设备会添加到网络设备列表中的“DP 口”节点下，如果是汇川设备，会显示在 Inovance 节点下，否则会显示在 " 第三方厂商 " 节点下。

■ 导入 ECT：网络设备列表中默认带部分 EtherCAT 设备，如果要加入其他 EtherCAT 设备，需要导入其对应的标准 EtherCAT xml(*.xml) 文件。导入成功后，设备会添加到网络设备列表中的 "EtherCAT 口 " 节点下，如果是汇川设备，会显示在 Inovance 节点下，否则会显示在 " 第三方厂商 " 节点下。

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。硬件组态主要面向中型 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 模块配置；如果只是配置本地 IO 模块，则直接打开硬件组态操作即可。硬件组态同时支持多总线 IO 配置，具体取决于所选用的 CPU 型号。

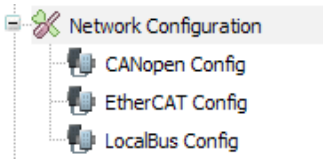
1 进入硬件组态界面

除 Modbus 设备和 ModbusTCP 设备外，其他总线类型设备都有对应的硬件组态界面。

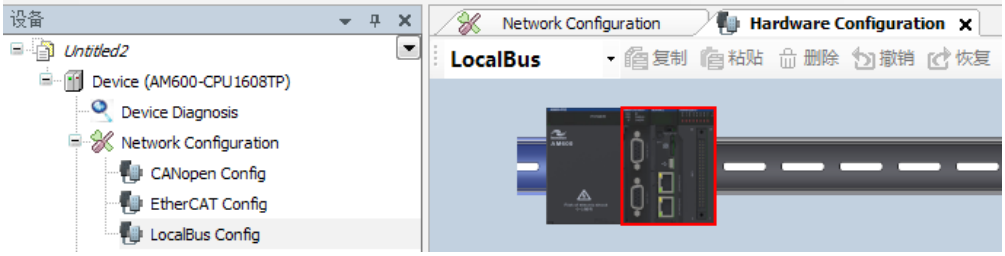
有两种方式可进入硬件组态界面：

1） 在网络组态界面双击一个设备

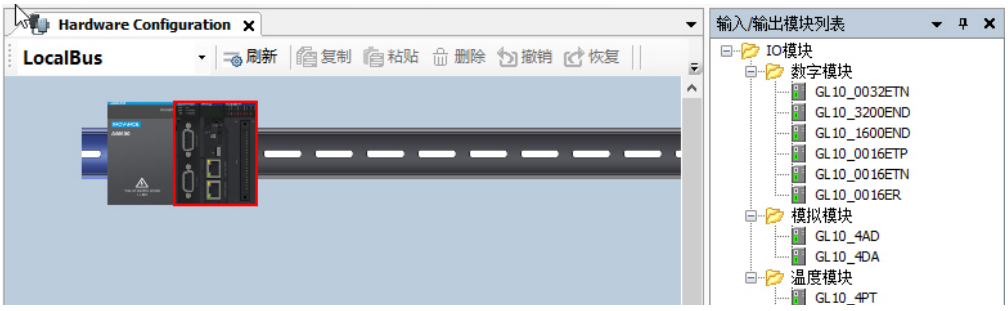
2） 双击软件左侧设备树 "Network Configuration" 节点下的某个总线节点（如下图所示）。



默认有 "LocalBus Config" 即本地总线配置节点，双击即可进入本地模块的配置：



同时软件右侧会显示 " 输入 \ 输出模块列表 ":

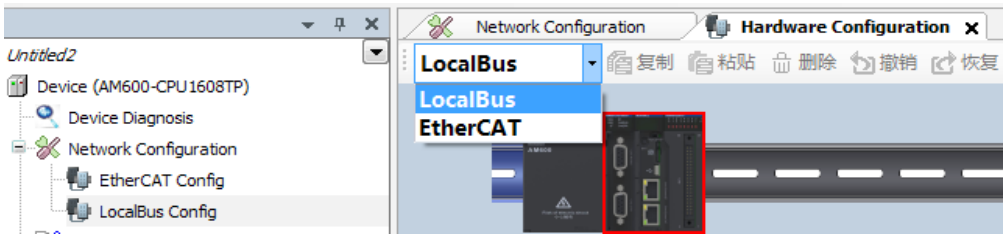


2 总线切换

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 "Network Configuration" 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其他总线类型，如下图所示：



3 添加模块

有 3 种方式用来添加 IO 模块：

1） 双击打开机架上的一个空机槽，在弹出的模块列表中双击特定模块即可添加。

2） 在右侧视图的模块列表中，选中一个设备节点，按住鼠标左键将其拖放到空机槽上。

3） 选中一个机架 (点击图中的蓝色部位) 或设备，然后双击视图右侧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。

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，拖动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，但不支持主机架和扩展机架之间模块拖拽操作。

3.1.3 设备树操作

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：

各功能操作符合基本标准操作。

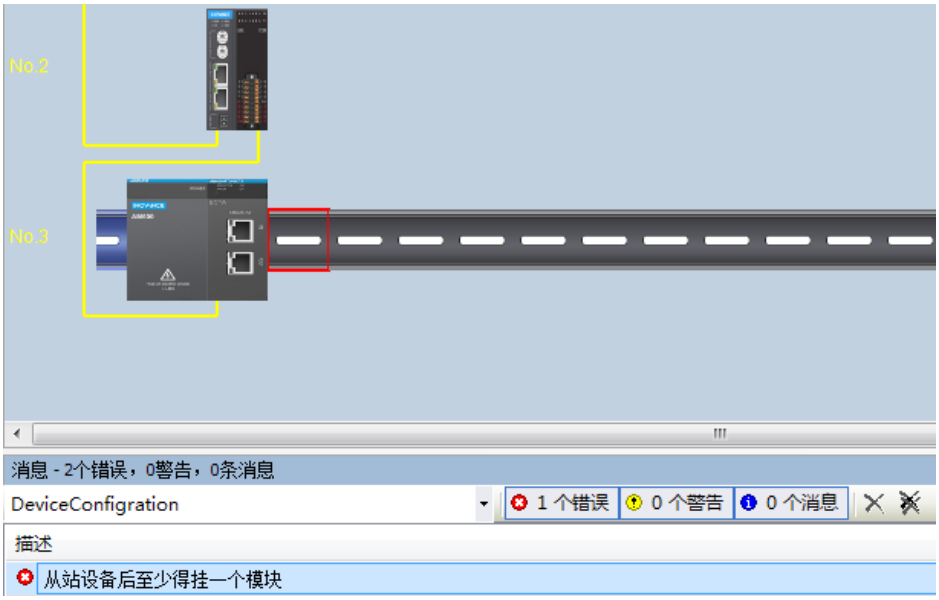
NOTE

复制和剪贴功能仅适用于本地主站设备、本地从站设备以及单独轴设备。

3.1.4 组态编译错误定位

组态设备制定了一些配置规则以及错误检测机制：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，InoProShop 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，并会闪烁 3 次红色的矩形框。如图所示：



3.2 CPU 配置

CPU 模块即中型 PLC 主模块，CPU 配置基于 PLC 硬件控制系统的要求，来完成对 PLC 及其控制系统的配置。中型 PLC 支持 EtherCAT 总线、Profibus DP、Modbus RTU、CAN 总线、ModbusTCP，另外还提供高速 I/O 功能。因此，若要完成整个 CPU 的配置，则需要根据 PLC 硬件网络配置相应的总线参数。

以 AM600 CPU 模块为例，其自带高速 I/O，后面可扩展本地 IO 模块。另外，在 CPU 模块配置界面还可以配置 CPU 系统参数、CPU 固件升级等功能。

3.2.1 CPU 配置的一般过程

1） 设计 CPU 整个硬件网络结构。

2） 根据设计的硬件架构在网络组态中激活相应总线，并添加总线对应的从站。目前 CPU 支持 EtherCAT 总线、DP 总线、CANopen、CANlink、ModbusRTU、ModbusTCP。

3） 如果是 EtherCAT AM600 从站、CANopen AM600 从站或者 DP AM600 从站，需要在硬件组态中添加 I/O 模块。

4） 配置总线对应的主站、从站和模块配置参数。

AM600 CPU 模块默认带高速 I/O 功能，每个 CPU 最多可扩展 16 个本地 I/O 模块，另外还需按具体需求对 CPU 本身系统参数、PLC I/O 刷新设置、PLC 总线任务、PLC 用户管理、日志、升级、任务配置等进行设置。

如需了解总线及对应从站的参数配置，请参见各总线对应的章节。PLC I/O 刷新设置、PLC 总线任务、PLC 用户管理、日志、任务配置等是 Codesys 自带功能，详见 Codesys 软件帮助链接。本文中的 CPU 配置主要讲述中型 PLC 的功能：CPU 参数配置、I/O 模块配置和高速 I/O 配置。

3.2.2 CPU 参数配置

1 系统设置

系统设置用于配置 CPU 故障停机、掉电保存位置、网络地址和系统时间，如下图所示。

图 3-4 系统设置对话框

错误时的运行模式

- 组态错误时停机：出现组态不一致时 CPU 是否停止运行，例如，CPU 后挂接的组态 IO 和实际硬件连接 IO 不匹配是否停机。
- 系统错误时停机：出现系统错误时 CPU 是否停止运行，如中断错误、堆栈溢出等。
- Flash 错误时停机：出现 Flash 错误时 CPU 是否停止运行，暂不支持。
- SD 卡错误时停机：出现 SD 卡错误时 CPU 是否停止运行，如 SD 卡内存满，SD 卡丢失等，暂时不支持。

掉电保存

- 保存位置：设置掉电保存位置：本地存储器和 SD 扩展卡。

NOTE

当选择 SD 扩展卡时，首先要确保 SD 扩展卡是否存在，否则掉电保存数据将丢失。

网络设置

- 网口：PLC 使用的 EtherNET 通讯网口名，AM600 和 AM400 系列只有一个 EtherNET 网口，AC800 系列有两个 EtherNET 通讯网口。不同 PLC 的网络名称不同，用户可以对不同网口设置不同的网络信息。
- 使用下面的 IP：PLC 的 IP 地址既可以通过手动修改，也可以自动获取，此配置用于手动修改 PLC 网络信息。
- 自动获取 IP：PLC 的 IP 地址由路由器或者交换机自动分配。注意：此功能当前只有 AC800 支持。
- IP 地址：PLC 的 IP 地址。
- 子网掩码：PLC 的子网掩码。

- 网关：设置 PLC 的网关。注意：此功能当前只有 AC800 支持。
- 读取：读取 PLC 的 IP 地址和子网掩码，显示在 IP 地址和子网掩码编辑框中。
- 写入：将编辑框中的 IP 地址和子网掩码写入 PLC。如果在登录状态并且使用网络连接，系统将会退出登录，此时用户需重连设备；如果使用 USB 连接，则不需要重连设备。

NOTE

◆ AC800 系列 PLC 的两个网口禁止设置相同段 IP，否则影响连接。
◆ 读、写 PLC 的 IP 地址时，需要先在通信设置选项卡选择要读、写的 PLC 设备，另外，写入的 IP 地址和子网掩码要符合 IP 地址和子网掩码规范。

- 识别设备：识别要连接的 PLC。当在通信设置选项卡扫描 PLC 时，有可能扫描到多台 PLC 设备，当选中其中一台 PLC 后，点击此按钮，此时 PLC 设备的显示面板上的两位数码管，将交替显示字符“0”，如图 3-5 所示：

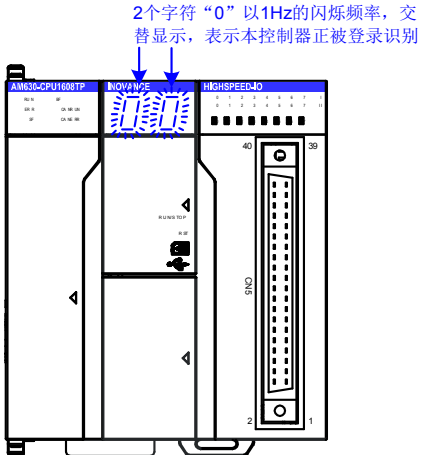


图 3-5 PLC 数码管处于识别状态

同时后台软件 InoProShop 显示图 3-6 的正在识别对话框，当用户关闭此对话框时表示识别完成，数码管恢复正常。

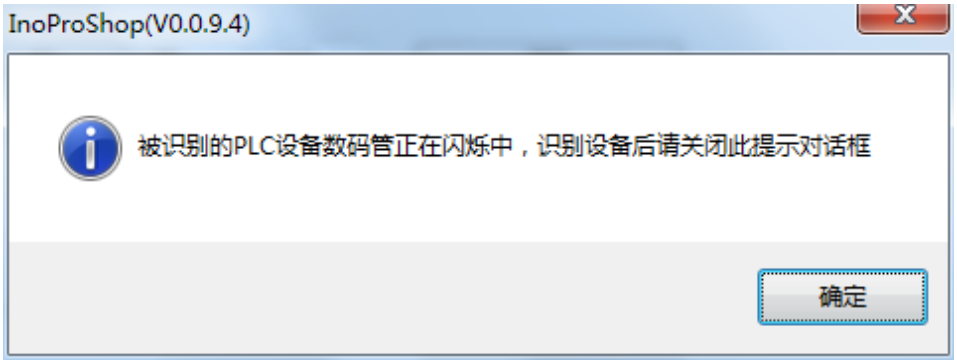


图 3-6 正在识别对话框

RTC 配置

- PLC 时间：显示读取的当前 PLC 时间
- 读取：读取 PLC 时间
- 写入：写入 PLC 当前设置的日期和时间，当前设置的日期和时间在左侧的日期和时间编辑框内。
- 同步到本地日期 / 时间：写入当前电脑对应的时间到 PLC。

NOTE

在写入 PLC 时间或者同步本地日期 / 时间时，有可能会对 PLC 造成影响，如影响总线同步功能，因此在写入 PLC 时间前，请先确认 PLC 是否处于停止状态，建议在写入时间后热复位 PLC。

时区

- PLC 时区：读取的 PLC 时区，如北京时区（东八区）为 UTC+8。
- 读取：读取 PLC 对应的时区
- 写入：写入 PLC 当前选择的时区

2 升级

升级页面用于 PLC 的固件升级，如图 3-7 所示。PLC 固件升级包提供要升级的软件数据，其中可能包含 UBOOT、Device Tree、内核、系统程序中的一种或者几种，一般升级包只包含系统程序。

NOTE

此处的升级功能不再支持，使用 InoProShop 工具进行升级。

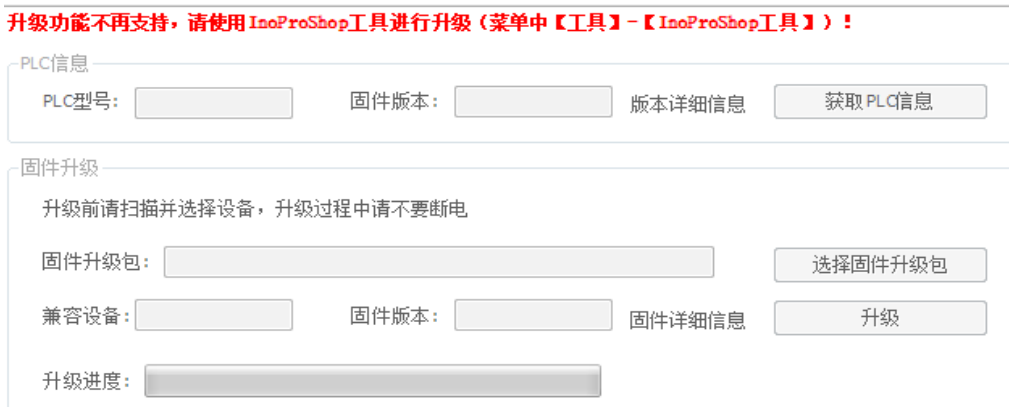


图 3-7 升级对话框

PLC 信息

- PLC 型号：当前 PLC 型号，如 AM600、AM610
- 固件版本：当前 PLC 的固件版本，如 1.2.3.0
- 版本详细信息：显示当前 PLC 详细的版本信息，详细版本信息可能包含 UBOOT、Device Tree、内核、系统程序版本中一种或者几种，如图 3-8 所示。如果 PLC 没有固件升级过，可能不包含版本信息。
- 获取 PLC 信息：获取 PLC 型号信息和固件版本信息，如果 PLC 没有进行固件升级，可能获取不到 PLC 信息。

固件升级

- 固件升级包：设置固件升级包，固件升级包以 .upgrade 为扩展名。
- 兼容设备：显示固件升级包的兼容设备，只有和当前 PLC 型号兼容的设备才能升级
- 固件版本：显示固件升级包的固件版本
- 固件详细信息：获取固件升级包的详细信息，详细信息如下图

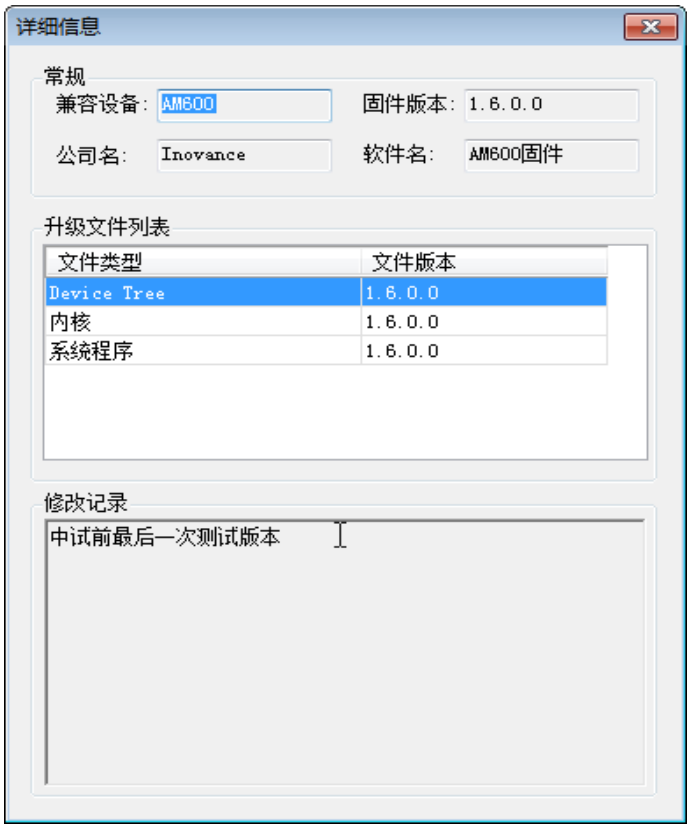


图 3-8 固件详细信息对话框

■ 升级：开始固件升级。升级时会检查设备类型和升级固件文件版本，如果升级的固件版本比 PLC 的固件版本高则直接升级，版本相同不需要升级，如果升级固件版本旧于 PLC 设备固件版本需要确认后才能升级。

NOTE

◆ 升级之前必需先在“通信设置”窗口扫描设备并选择要升级的 PLC；
◆ 升级过程中不能断电，否则可能造成系统不可恢复的故障；
◆ 升级大概需要 2 分钟，完成后会自动重启设备；
◆ 重启后（升级完成后）数码管会显示 00 或者动态变化的数字；
◆ 另外升级完成后，PLC 设备名称可能会发生变化，需要重新扫描设备。

升级完成后，获取 PLC 信息及版本详细信息，核对 PLC 信息及版本详细信息是否与固件版本及详细信息一致。如图 3-9 所示：

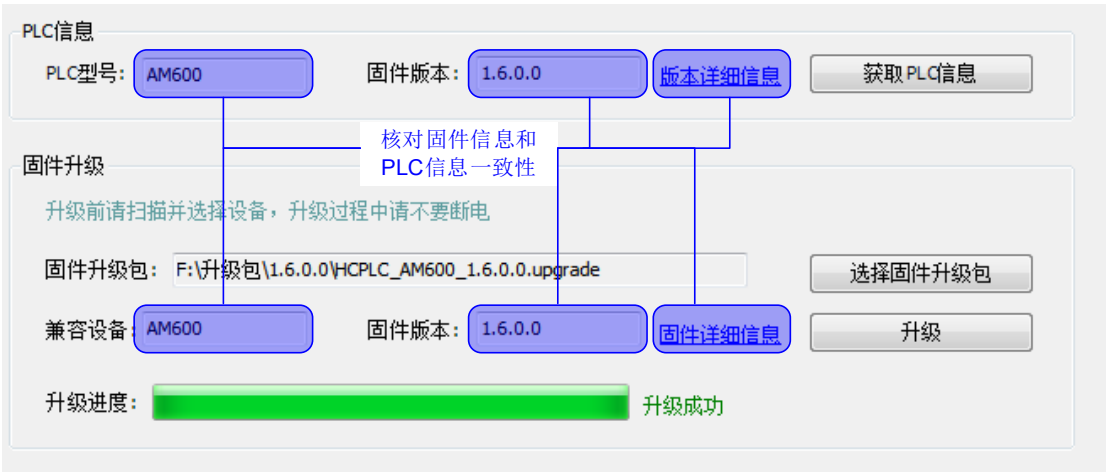


图 3-9 核对升级是否成功

3 信息

显示 PLC 设备基本信息：名称、供应商、组、类型、序号、版本，模块序号、描述、图像，如图 3-9 所示。

另外，登陆后信息页面会显示 CPU 的单板软件版本和逻辑软件版本信息。单板软件版本为 CPU 系统程序版本，逻辑软件版本为 CPU 内 FPGA 软件版本。

图 3-10 CPU 信息页面

3.2.3 I/O 模块配置

I/O 模块分 5 种类型：

数字量输入（DI）、数字量输出（DO）、模拟量输入（AD）和模拟量输出（DA）和温度模块。

数字量输出分 3 种类型：继电器输出（ERN 型）、NPN 输出（ETN 型）和 PNP 输出（ETP 型）。

数字量输入和输出包含 16 点和 32 点。

温度模块包含 4TC（4 通道温度检测模块，支持热电偶）、8TC（8 通道温度检测模块，支持热电偶）和 4PT（4 通道温度检测模块，支持热电阻）。

1 数字量输入模块

数字量输入模块无模块参数配置，只有 I/O 映射、状态和信息页面，一般情况下只需要在 I/O 映射界面映射 I/O 变量以获取数字量输入值。在此以 16 点数字量输入模块为例。

1） DI16 I/O 映射

DI16 为 16 位数字输入模块，如图 3-11 所示，可通过在 I/O 映射界面对每位或者每 8 位映射到一个变量，获取输入值，具体请参照 I/O 映射链接。

图 3-11 DI16 I/O 映射对话框

2） 信息

显示 DI16 模块设备基本信息：名称、供应商、组、类型、序号、版本，订购号、描述、图像，如图 3-12 所示。另外，登陆后信息页面会显示 DI 模块逻辑软件版本，逻辑软件版本为 DI 模块内 FPGA 软件版本。

图 3-12 DI 信息页面

2 数字量输出模块

数字量输出模块，无模块参数配置，只有 I/O 映射、状态和信息页面，一般情况下只需要在 I/O 映射界面映射 I/O 变量，然后把映射变量值输出到数字量输出模块。数字量输出包含 16 点和 32 点，两者的 IO 映射界面相似。在此以 16 点数字量输出模块为例。

1） DO16 I/O 映射

DO16 为 16 位输出模块，如图 3-13 所示，可通过在 I/O 映射界面对每位或者每 8 位映射到一个变量，以输出变量值，具体请参照 I/O 映射链接。

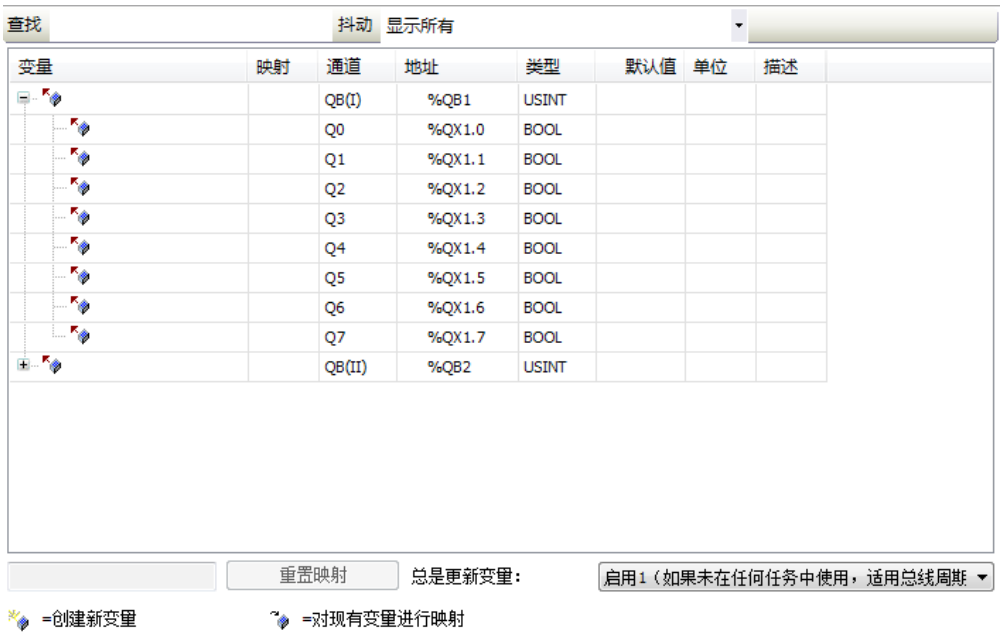


图 3-13 DO16 I/O 映射对话框

2） 信息

显示 DO16 模块设备基本信息：名称、供应商、组、类型、序号、版本，模块号、描述、订购号、图像，如图 3-14 所示。

另外，登陆后信息页面会显示 DO 模块逻辑软件版本，逻辑软件版本为 DO 模块内 FPGA 软件版本。

图 3-14 DO 信息页面

3 模拟量输入模块

1） 一般配置

模拟量输入模块包括 4 个通道，每个通道都有参数配置和 I/O 映射寄存器（16 位）设置，下面仅对每个模块的一个通道进行说明。

图 3-15 模拟量输入一般配置对话框

- 模块诊断上报：模块出现故障时，是否上报给父设备（如 CPU、远程模块从站），如果上报，则若父设备设置了故障停机，则父设备会停止本设备的运行。
- 使能通道：是否激活此通道，只有激活了通道，此通道才能使用。
- 通道诊断上报：模块对应的通道出现故障时，是否上报给父设备（如 CPU、远程模块从站），如果上报。若父设备设置了故障停机，则父设备会停止本设备的运行。
- 转换模式：设置模拟量输入转换类型，此设置决定了通道输入转换类型和转换值的范围，转换类型及对应的数字量值关系、范围如下表：

表 3-1 模拟量输入模拟值和数字量值相对应关系表

	输入额定范围	额定对应数字量	输入极限范围	极限对应数字量
模拟电压输入	-10V~10V	-20000~20000	-11V~11V	-22000~22000
	0V~10V	0~20000	-0.5V~10.5V	-1000~21000
	-5V~5V	-20000~20000	-5.5V~5.5V	-22000~22000
	0V~5V	0~20000	-0.25V~5.25V	-1000~21000
	1V~5V	0~20000	0.8V~5.2V	-1000~21000
模拟电流输入	-20mA~20mA	-20000~20000	-22mA~22mA	-22000~22000
	0mA~20mA	0~20000	-1 mA~21mA	-1000~21000
	4mA~20mA	0~20000	3.2mA~20.8mA	-1000~21000

- 滤波参数：模拟量输入通道滤波时间，范围 1ms-255ms。
- 断线标志：设置模拟量输入通道是否检测断线，因不能区分模拟量输入 0 值和断线，所以所有转换模式范围中，包含 0 值输入的，不能激活断线标志。
- 超限标志：设置模拟量输入通道是否检测超限。

■ 峰值保持功能：设置模拟量输入通道是否保持峰值输入。

2） AI4 I/O 映射

AI4 为 4 通道模拟量输入，每个通道模拟输入对应一个 16 位的整数值，模拟量值和数字量值关系见模拟量输入一般配置。可以在此界面对每个 16 位整数值映射一个变量，以获取一个输入通道模拟量对应的数字量值。具体请参照 I/O 映射链接。

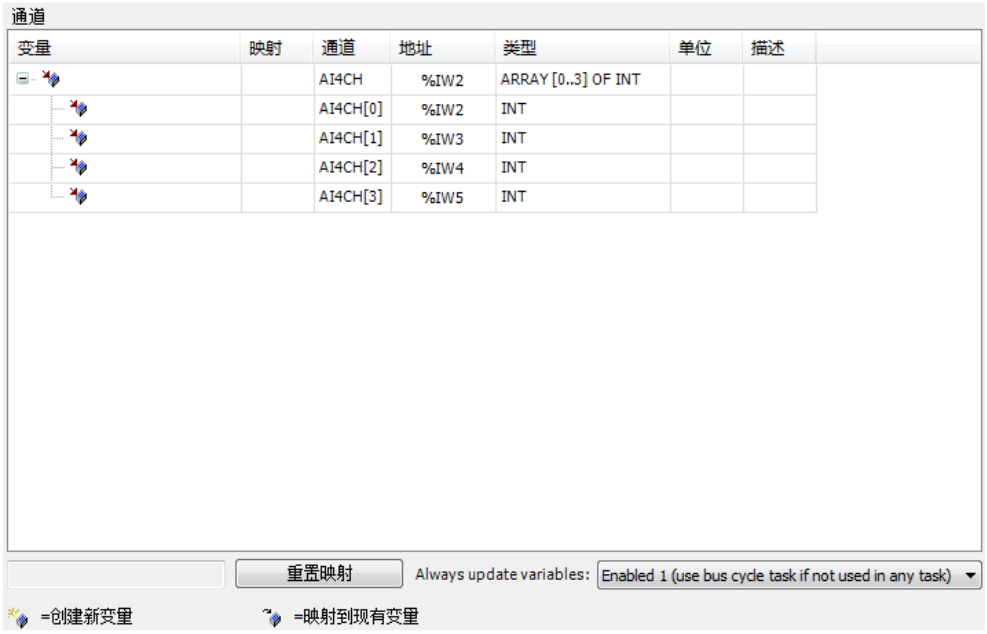


图 3-16 AI4 I/O 映射对话框

3） 信息

显示 AI4 模块设备基本信息：名称、供应商、组、类型、序号、版本，模块号、描述、订购号、图像。

另外，登陆后信息页面会显示 AI4 模块单板软件版本和逻辑软件版本，单板软件版本为 AI4 嵌入式软件版本，逻辑软件版本为 AI4 模块内 FPGA 软件版本。

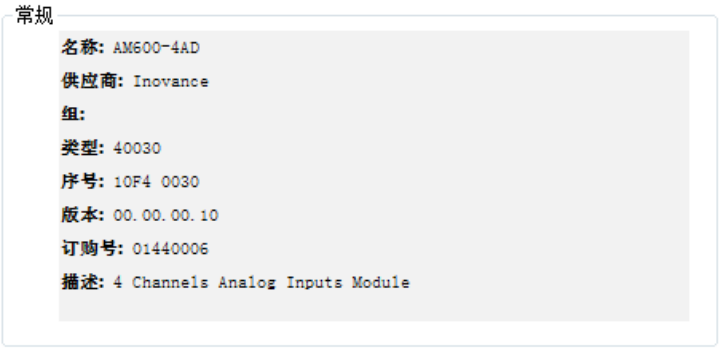


图 3-17 AI4 信息页面

4 模拟量输出模块

1） 一般配置

模拟量输出模块包括 4 个通道，每个通道都包括参数配置和 I/O 映射寄存器设置（16 位），下面仅对每个模块的一个通道进行说明。

图 3-18 模拟量输出一般配置对话框

- 模块诊断上报：模块出现故障时，是否上报给父设备（如 CPU、远程模块从站）。如果上报，则若父设备设置了故障停机，则父设备会停止本设备的运行。
- 使能通道：是否激活此通道，只有激活了通道，此通道才能使用。
- 通道诊断上报：模块对应的通道出现故障时，是否上报给父设备（如 CPU、远程模块从站），如果上报，则若父设备设置了故障停机，则父设备会停止本设备的运行。

■ 转换模式：设置模拟量输出转换类型，此设置决定了此通道输出转换类型和转换的值范围，转换类型及对应的数字量值关系、范围如下表：

表 3-2 模拟量输出模拟值和数字量值相对应关系

	输出额定范围	额定对应数字量	输出极限范围	极限对应数字量
模拟电压输出	-10V~10V	-20000~20000	-11V~11V	-22000~22000
	0V~10V	0~20000	-0.5V~10.5V	-1000~21000
	-5V~5V	-20000~20000	-5.5V~5.5V	-22000~22000
	0V~5V	0~20000	-0.25V~5.25V	-1000~21000
	1V~5V	0~20000	0.8V~5.2V	-1000~21000
模拟电流输出	0mA~20mA	0~20000	0 mA~21mA	0~21000
	4mA~20mA	0~20000	3.2mA~20.8mA	-1000~21000

■ 停止后输出状态：模块运行停止后，设置输出保持值。

■ 输出清零：模块运行停止后，输出一直为 0。

■ 输出保持：模块运行停止后，输出一直保持上次输出值。

■ 输出预设值：模块运行停止后，输出一直为预定值。 预设值可以设置模拟量值，也可以设置数字量值，模拟量值和数字量值一一对应，修改其中一个，另外一个自动变化。预设值范围和当前转换模式有关，具体请参见上述转换模式。

2） AO4 I/O 映射

AO4 为 4 通道模拟量输出，每个通道模拟输出对应一个 16 位的整数值，模拟量值和数字量值关系见模拟量输出一般配置，如图 3-19。可以在此界面对每个 16 位整数值映射一个变量，以把此变量值输出到当前通道，然后经模拟量输出模块转换为模拟量值输出。具体请参照 I/O 映射链接。

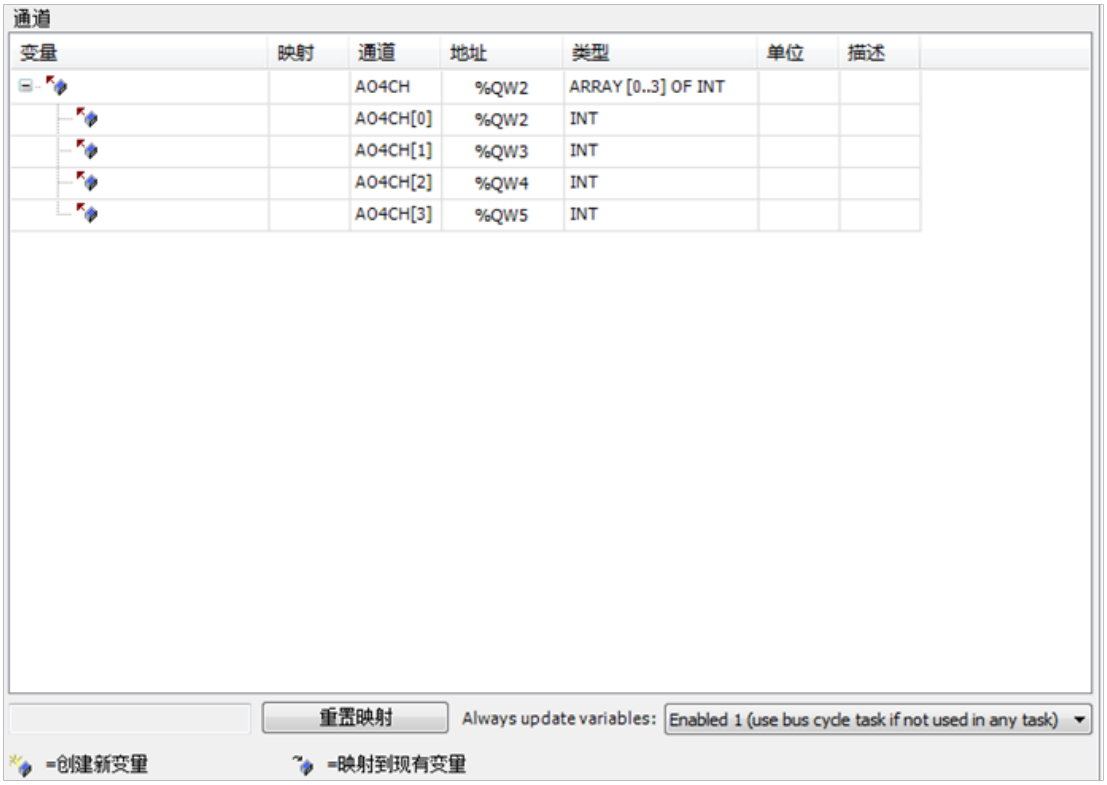


图 3-19 AO4 I/O 映射对话框

3） 信息

显示 AO4 模块设备基本信息：名称、供应商、组、类型、序号、版本、模块号、描述、订购号、图像。

另外，登陆后信息页面会读取 AO4 模块单板软件版本和逻辑软件版本并显示出来，单板软件版本为 AO4 嵌入式软件版本，逻辑软件版本为 AO4 模块内 FPGA 软件版本。

图 3-20 AO4 信息页面

5 温度模块

温度模块包含 4TC（4 通道温度检测模块，支持热电偶）、8TC（8 通道温度检测模块，支持热电偶）和 4PT（4 通道温度检测模块，支持热电阻），都有对应的一般配置和通道配置。

一般配置用于配置温度模块的单位类型和采样周期，通道配置分别配置每个通道传感器类型、滤波时间、超限、温度偏移等参数。

1） 一般配置

不同类型温度模块配置有稍微差异，4TC 和 8TC 有冷端补偿功能，4PT 不支持。另外 8TC 支持外部冷端补偿，而 4TC 不支持。下图以 8TC 配置界面为例。

图 3-21 8TC 一般配置对话框

- 模块诊断上报：模块出现故障时，是否上报给父设备（如 CPU、远程模块从站），如果上报，则若父设备设置了故障停机，则父设备会停止本设备的运行。
- 冷端补偿：选择冷端补偿方式。只有 8TC 才支持外部冷端补偿，并且 8TC 使用通道 7（最后一个通道）来进行外部冷端补偿输入。
- 温度单位：温度模块输入使用的单位，支持摄氏度或者华氏度。
- 采样周期：设置温度模块采样时间，支持 250ms、500ms 和 1000ms。

2） 通道配置

不同类型模块支持不同的通道。4TC 和 4PT 支持 4 通道；8TC 支持 8 通道；由于每个通道的配置参数基本相同，仅对一个通道进行说明。8TC 一个通道配置如下图。

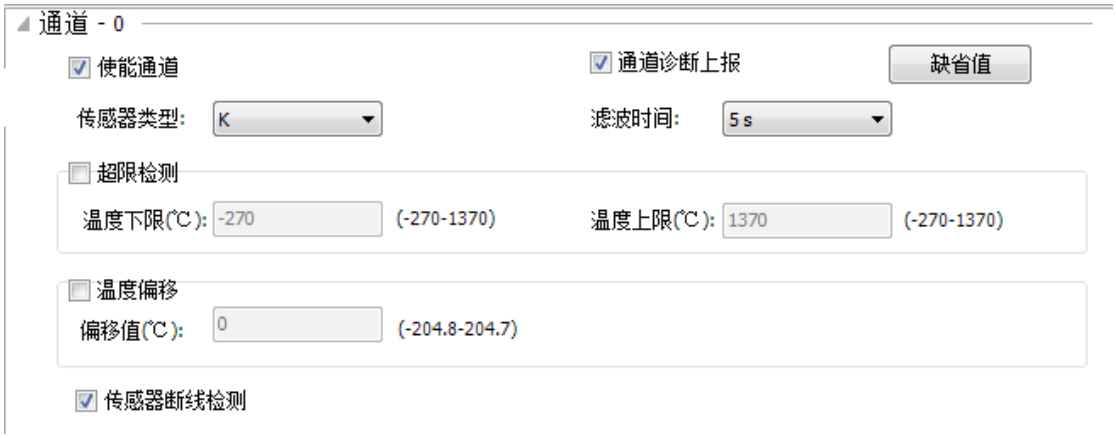


图 3-22 温度模块通道配置对话框

- 使能通道：是否激活此通道，只有激活了通道，此通道才能使用。
- 通道诊断上报：模块对应的通道出现故障时，是否上报给父设备（如 CPU、远程模块从站），如果上报，则若父设备设置了故障停机，则父设备会停止本设备的运行。
- 缺省值：复位此通道值为默认值。
- 传感器类型：8TC 和 4TC 传感器类型、传感器规格如下表。默认值使用 K 传感器。

表 3-3 8TC 和 4TC 传感器类型、传感器规格

项目	传感器名称	摄氏温度范围（°C）	华氏温度范围（°F）
热电偶类型	B	250°C ~1800°C	482° F~3272° F
	E	-270° C~1000° C	-454° F~1832° F
	N	-200° C~1300° C	-328° F~2372° F
	J	-210° C~1200° C	-346° F~2192° F
	K	-270° C~1372° C	-454° F~2502° F
	R	-50° C~1768° C	-58° F~3214° F
	S	-50° C~1768° C	-58° F~3214° F
	T	-270° C~400° C	-454° F~752° F

表 3-4 4PT 传感器规格

项目	传感器名称	摄氏温度范围（°C）	华氏温度范围（°F）
热电阻类型	Pt100	-200°C ~850°C	-328 °F ~1562 °F
	Pt500	-200°C ~850°C	-328 °F ~1562 °F
	Pt1000	-200°C ~850°C	-328 °F ~1562 °F
	Cu100	-50°C ~150°C	-58 °F ~302 °F

- 滤波参数：温度模块此通道使用的滤波时间，范围 0ms-100ms。默认 5ms。
- 超限检测：使用此通道超限检测功能，如果不在温度上下限之间，会报超限故障。范围见表 3-3。
- 温度偏移：设置温度模块偏移补偿值，范围 -204.8-204.7。
- 传感器断线检测：使能传感器断线报警功能。

3） I/O 映射

不同类型的温度模块包含不同个数的通道，相应的，也包含不同个数的 IO 映射。下图为 4PT 的 I/O 映射界面，每个通道参数值为温度值。具体请参*照 I/O 映射*链接。

图 3-23 温度 I/O 映射对话框

4） 信息

显示温度模块设备基本信息：名称、供应商、组、类型、序号、版本，模块号、描述、订购号、图像。

另外，登陆后信息页面会显示温度模块单板软件版本和逻辑软件版本，单板软件版本为温度模块嵌入式软件版本，逻辑软件版本为温度模块内 FPGA 软件版本。

图 3-24 温度模块信息页面

3.2.4 高速 I/O 配置

点击设备窗口中的 HIGH_SPEED_IO 选项，会弹出 HSIO 的参数配置界面。在该界面中可以对高速 I/O 功能及相关参数进行配置，具体包括：

（1）高速计数功能；

（2）高速输出功能；

（3）高速输入沿中断的功能。

NOTE

◆ 有关高速 I/O 的硬件端口接线指导以及高速 I/O 配置兼容性，请分别参见本文附录 E 和附录 F。

◆ 使能成高速 IO 的端口无法作普通端口使用，即无法通过 IO 映射改变 IO 状态。

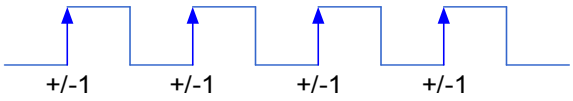
1 高速计数器功能

1） 高速计数功能的参数配置：包括计数模式选择、比较一致输出以及外部触发输入；其中计数模式包括单相计数，AB 相计数，CW/CCW 计数，内部时钟计数。以【计数器 0】为例（计数器号支持 0~7），配置步骤如下：

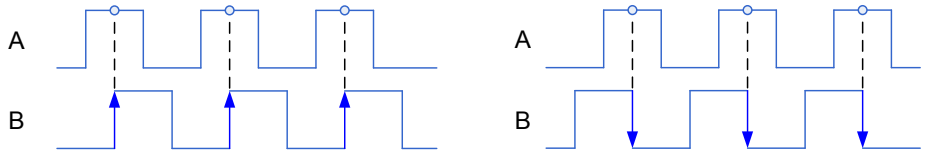
■ 勾选使用的【计数器 0】；

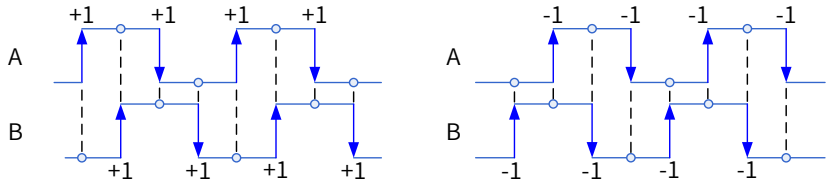
■ 配置高速计数的模式：

单相计数：接收外部单相编码器脉冲信号，只占用硬件 X0 端口；

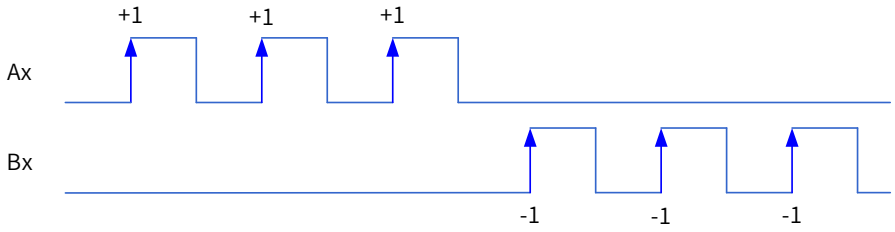


AB 相计数：接收外部 AB 相编码器脉冲信号，AB 亦可配置成 4 倍频，占用硬件 X0、X1 端口；





CW/CCW 计数：接收外部 CW/CCW 编码器脉冲信号，占用硬件 X0、X1 端口；



内部时钟：高速计数器 0 的脉冲信号来源于软件内部定时产生脉冲信号，支持 1us/10us/100us/1ms；

■ 比较一致输出：当高速 I/O 的计数值达到设定值的时候，对应的硬件输出端口会输出相应的电平信号。使用该功能时候需调用中断使能功能块 HC_EnableInterrupt 和计数值设定功能块 HC_SetCompare/HC_SetCompareM；

■ 外部触发输入：启用外部触发输入引脚可以实现高速计数器值的锁存和脉冲宽度测量的功能，计数锁存功能需要调用 HC_TouchProbe 功能块；脉冲宽度测量功能需要调用 HC_MeasurePulseWidth 功能块。

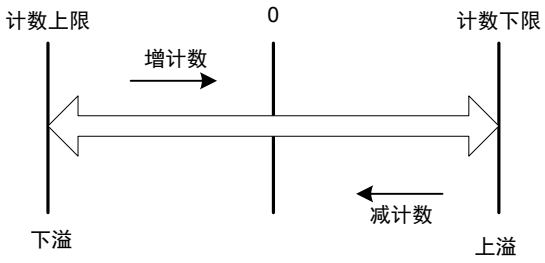
■ 滤波参数：调整高速计数端口的滤波，默认 2us；

2） 实例化高速计数器，数据类型 COUNTER_REF；以【计数器 0】为例配置：

【计数器 0】实例化默认名称：HS_Counter0；

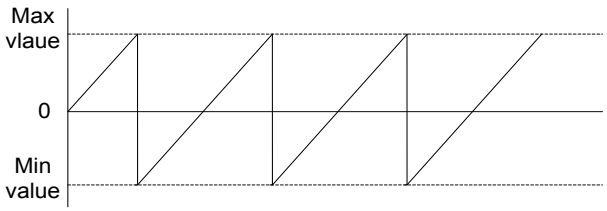
计数方式：

■ 线性计数：线性计数器在最大值和最小值之间计数，当正向计数达到最大值或反向计数达到最小值时停止计数，溢出标志有效。



■ 环形计数：需要配合【HC_SetRing】使用

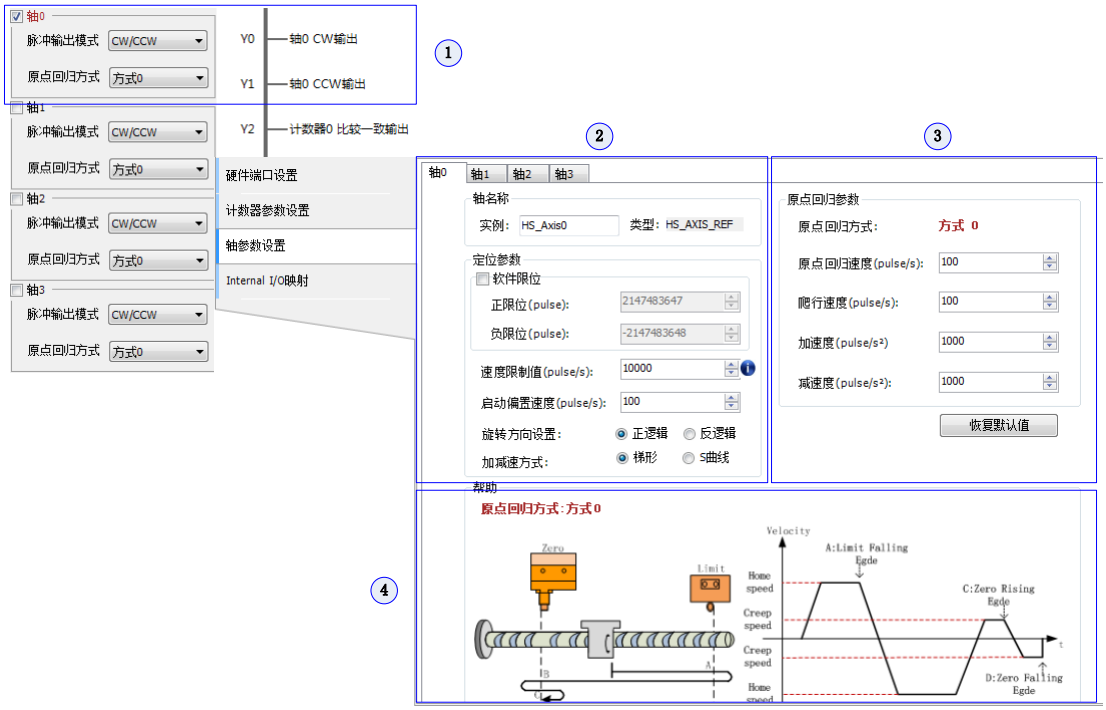
环形计数器在最大值和最小值之间计数，当正向计数超过最大值之后跳转到最小值，当反向计数时小于最小值则跳转到最大值。



3） 外部触发功能选择：

当 X8 输入的电平有效时，可以对计数器的功能进行设定，例如 X8 可配置作为禁用【计数器 0】计数的信号；禁用【计数器 0】预制功能；计数锁存的功能。

2 高速脉冲输出功能

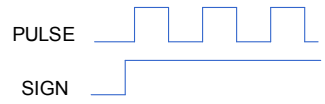
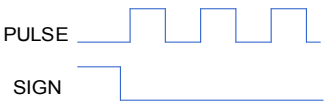
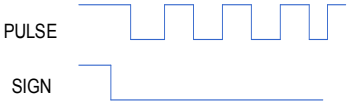
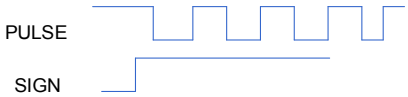
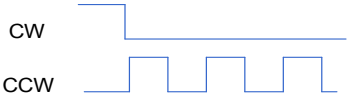
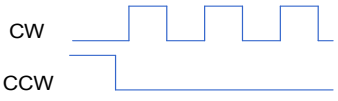
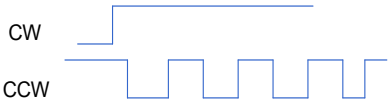
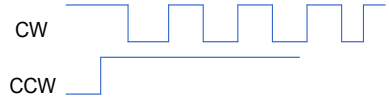


1） 配置高速脉冲输出功能，包括输出脉冲方式（脉冲 + 方向、CW/CCW）、原点回归方式；

以【轴 0】为例配置（轴号支持 0~3）：

■ 勾选使用的【轴 0】；

■ 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
正逻辑		
负逻辑		
脉冲指令形态	CW/CCW	
	正转	反转
正逻辑		
负逻辑		

■ 回零方式

支持 0~3，详见回零示意图；

2） 轴的实例化，数据类型 HS_AXIS_REF；以【轴 0】为例配置：

■ 【轴 0】实例化默认名称：HS_Axis0；

■ 正负限位：软限位；

■ 速度限制：最大速度限制；

■ 启动偏置速度：脉冲启动时基底速度；

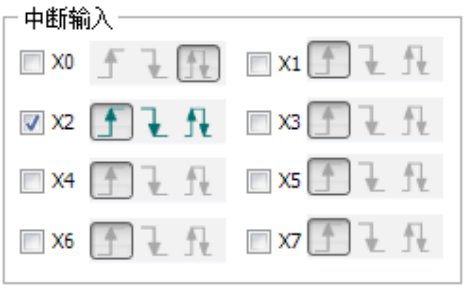
■ 加速方式：梯形加速和 S 曲线加速。

3） 原点回归参数配置，包括回零速度、爬行速度等；

需要配合【MC_Home_P】功能块使用；

4） 不同的原点回归方式的示意图；

3 高速输入中断

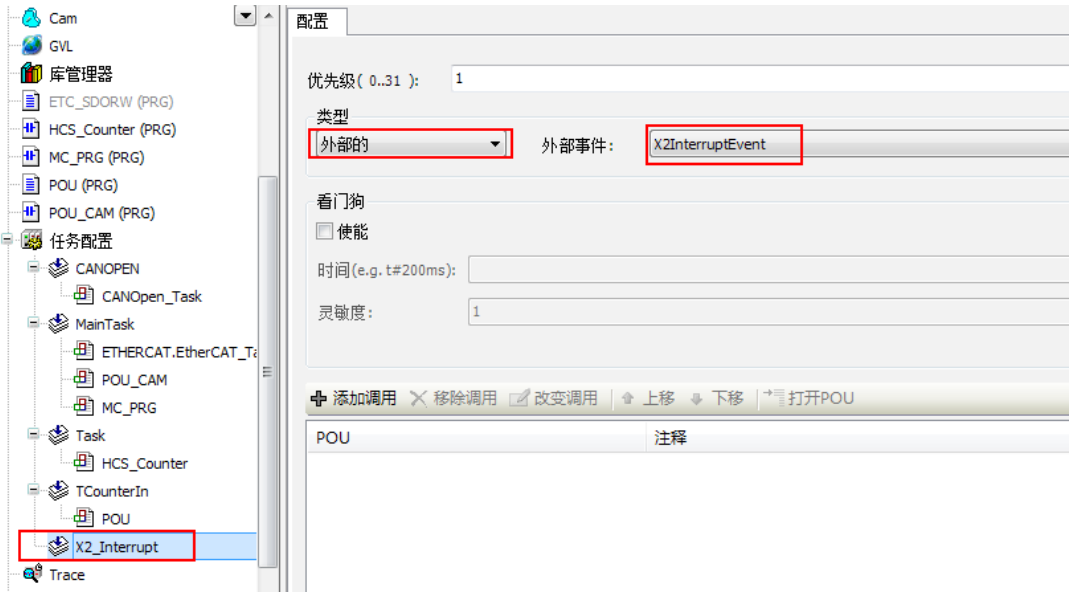


例如

1） 勾选 X2 高速输入中断；

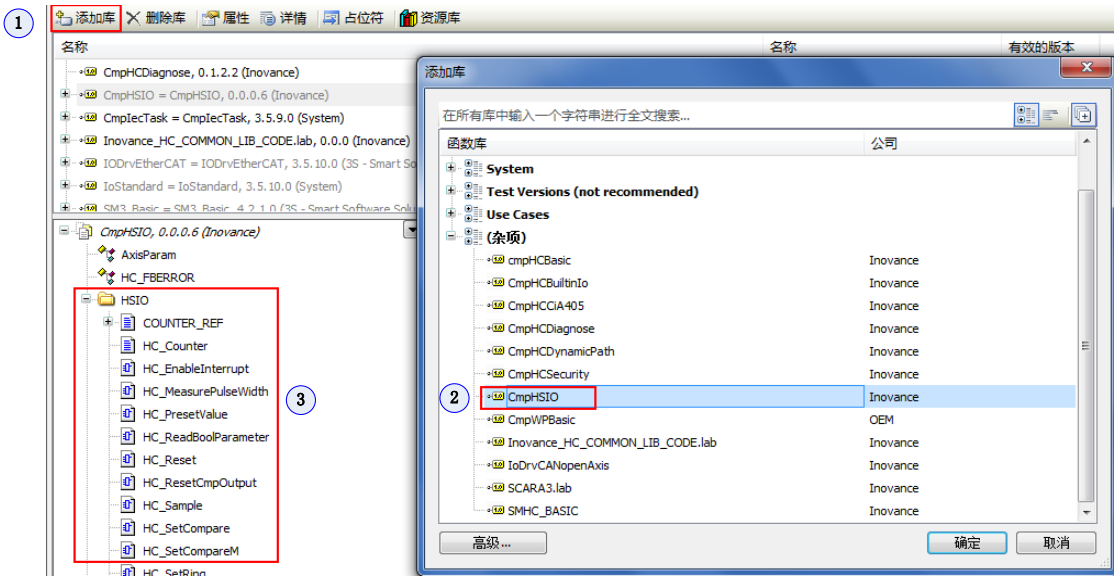
选择配置 X2 的沿中断：上升沿有效 / 下降沿有效 / 上升沿和下降沿都有效；

2） 中断任务的配置：



打开外部中断功能块【HC_EnableInterrupt】，增加 X2 的中断任务，当 X2 上升沿信号有效时，触发中断任务中的程序执行；

4 高速 I/O 库



1） 【库管理器】添加库；

2） 添加新高速 I/O 库；

3） 新高速 I/O 库的功能块。

3.3 EtherCAT 配置

3.3.1 概述

EtherCAT 是一种基于以太网的开放式工业现场技术，具有通信刷新周期短、同步抖动小、硬件成本低等特点，支持线型、树型、星型以及混合网络拓扑结构。EtherCAT 从站必须使用专用的通信控制芯片（ESC），EtherCAT 主站可以使用标准的以太网控制器。

如需了解 EtherCAT 原理和相关技术，请参考书籍《工业以太网现场总线 EtherCAT 驱动程序设计和应用》，或者登录 EtherCAT 技术委员会官网，网址：https://www.EtherCAT.org.cn。

3.3.2 常用功能

1 设备安装

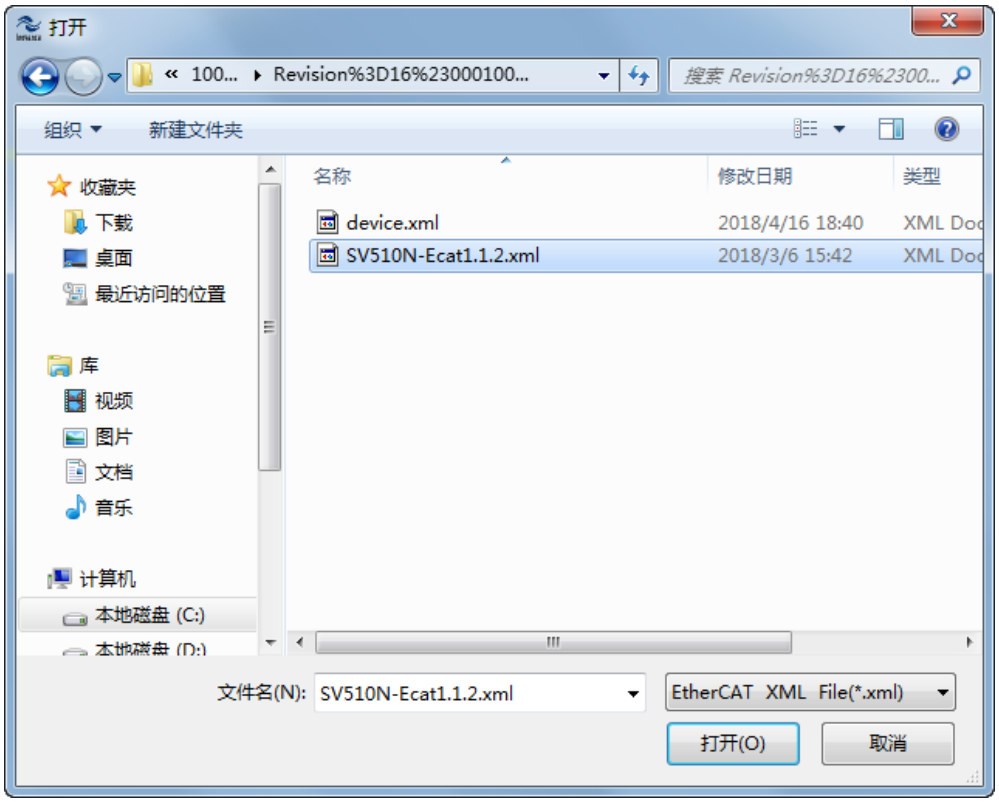
EtherCAT 设备安装是指将符合 ETG（EtherCAT 技术委员会）标准的设备描述文件（后缀名为“.XML”）导入到编程软件 InoProShop，经过软件的文件解析处理后生成可供用户添加、删除等操作的 EtherCAT 组态设备。编程软件 InoProShop 内部集成了汇川所有 EtherCAT 从站设备，无需单独安装。如需使用第三方 EtherCAT 设备，必须先安装第三方厂商提供的设备描述文件。

安装方法有两种，分别在网络组态界面安装和通过菜单工具安装，具体操作步骤如下。

■ 在网络组态界面安装

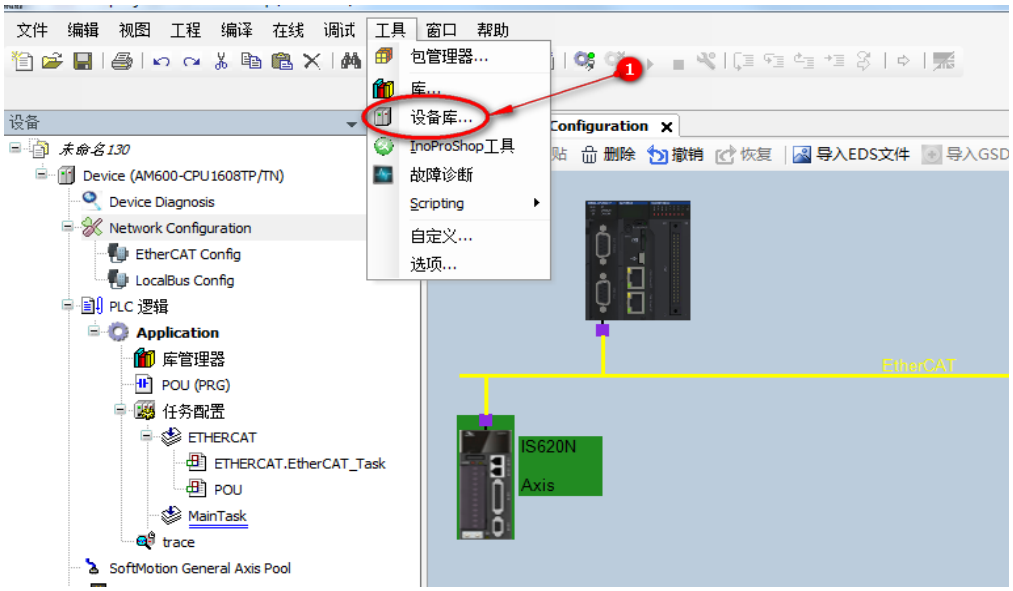
1） 点击“导入 ECT 文件”，弹出如下对话框：

2） 选择相应设备的 XML 文件后点击“打开”即可。

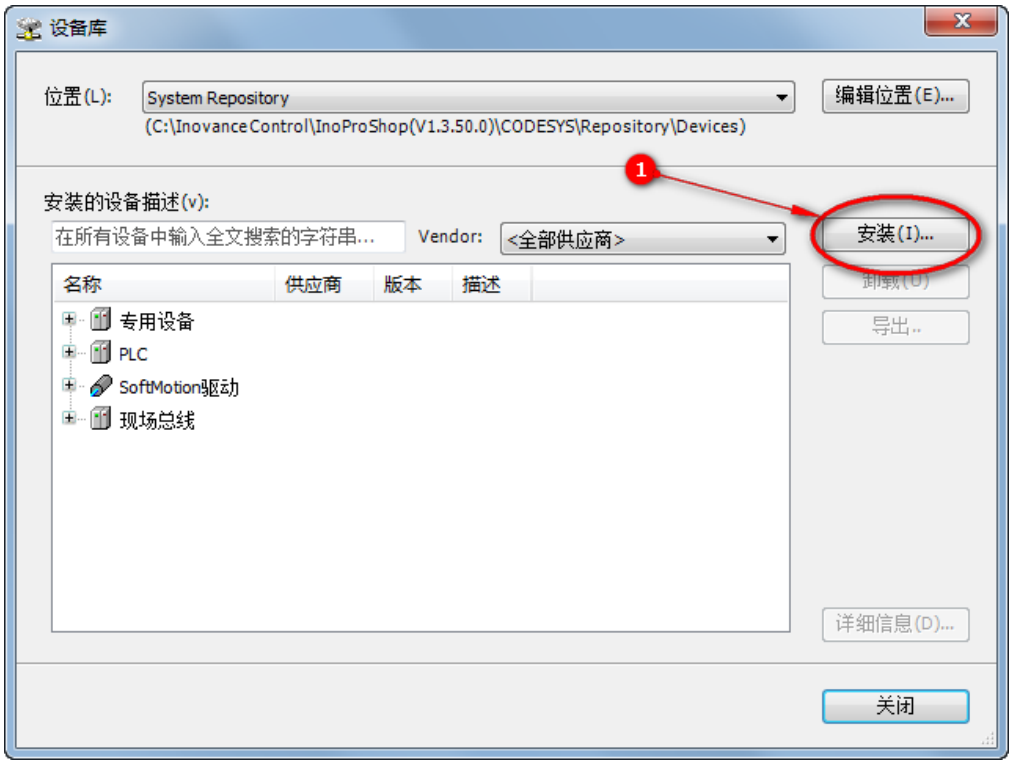


■ 通过菜单工具安装

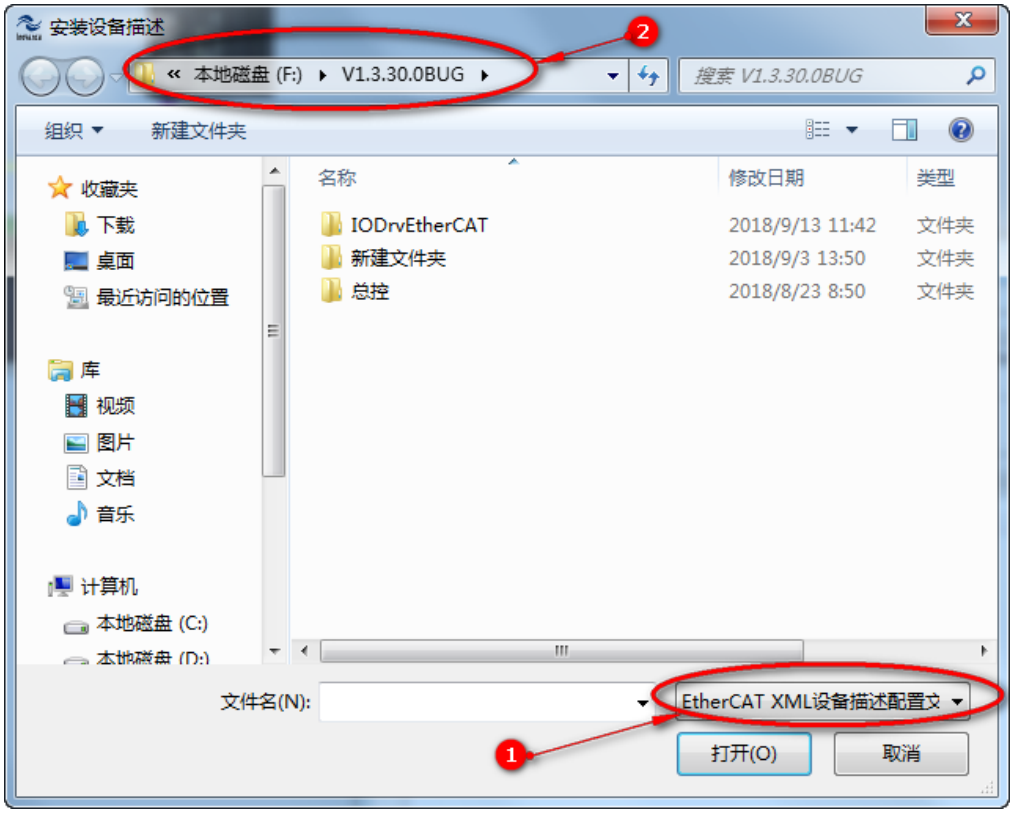
1） 打开菜单栏中“工具”选项卡下的“设备库”。



2） 在弹出的对话框中选择“安装 (I)”。



3） 在弹出的“安装设备描述”对话框选择“EtherCAT XML 设备描述配置文件”项，选中本机路径中保存的从站设备描述文件，打开相应的 XML 文件即可。



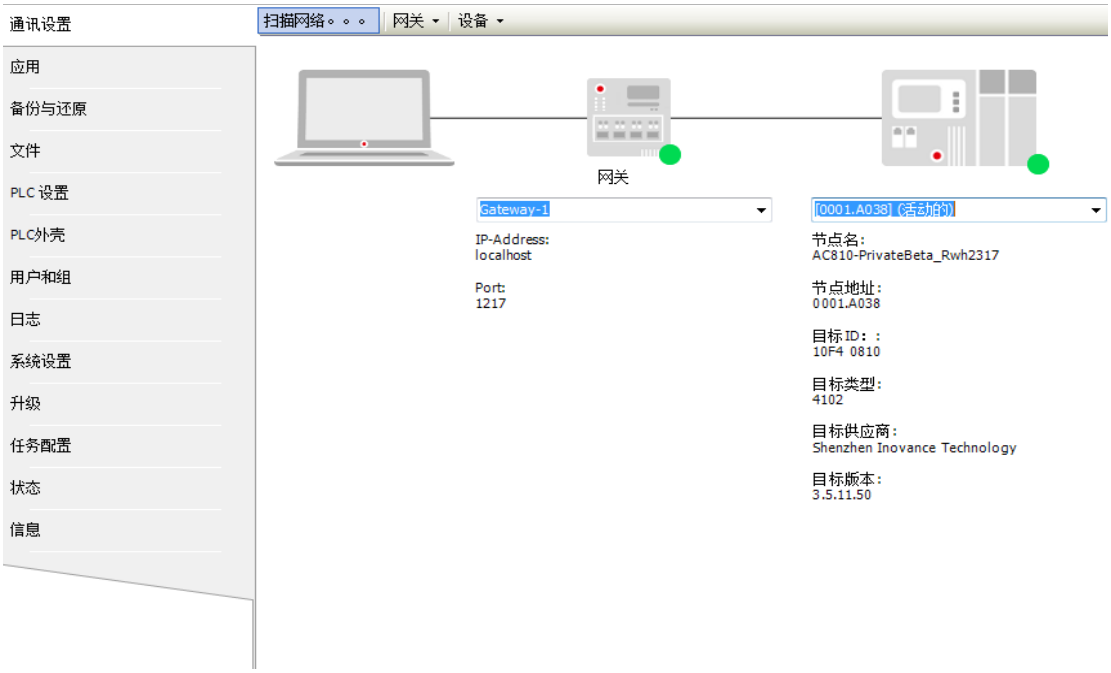
2 扫描设备

推荐使用扫描功能，按照【热复位】->【退出登录】->【扫描设备】流程操作。

■ 准备条件

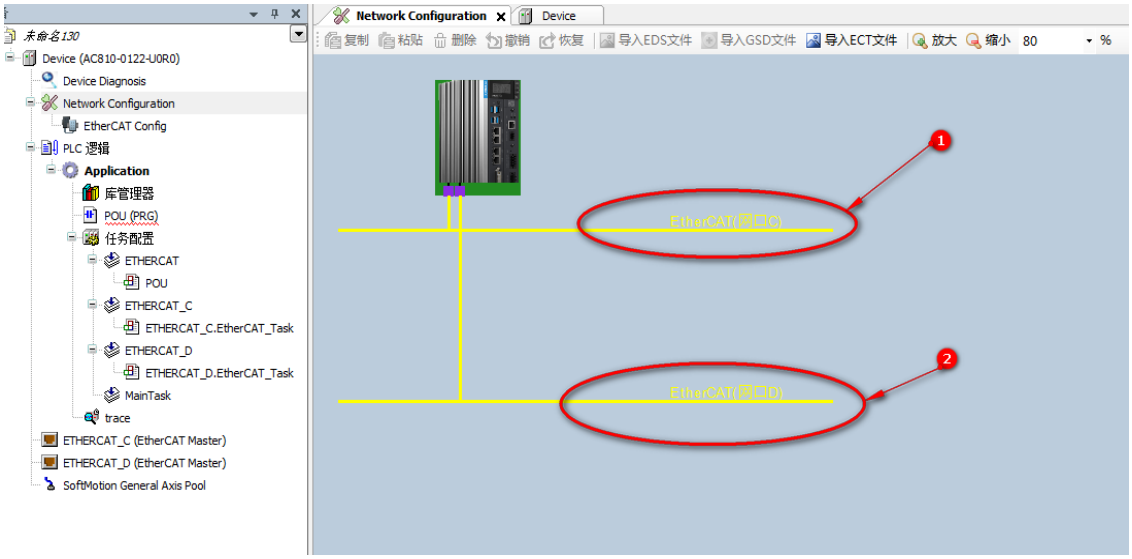
扫描设备命令的使用前提条件如下：

1） PC 与 PLC 通过网关正确连接，如下图：



2） PLC 与从站组网正常。

3） 后台组态端口配置信息与实际 PLC 连接端口一致，如下图：

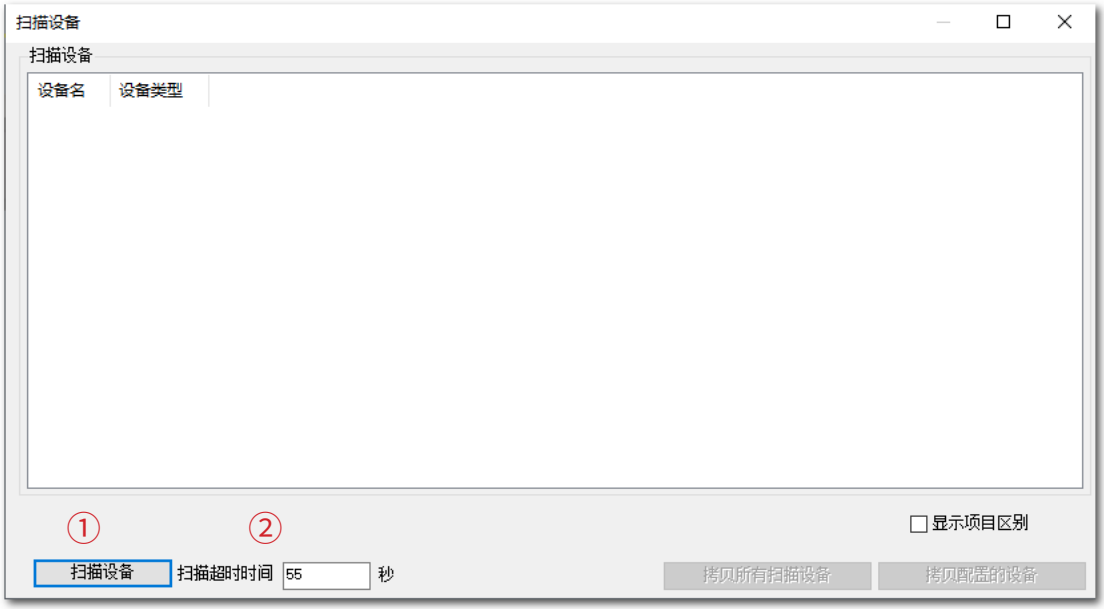


NOTE

- ◆ 后台组态配置 EtherCAT 有 EtherCAT_C 与 EtherCAT_D，PLC 端也要有 EtherCAT_C 与 EtherCAT_D 端口信息，为保持一致，建议在使用扫描命令前先下载一次端口配置信息。
- ◆ PLC 处于停止状态下才可正常扫描设备。

■ 扫描操作

1） 正常情况下，点击扫描设备会弹出下图所示的扫描设备框：



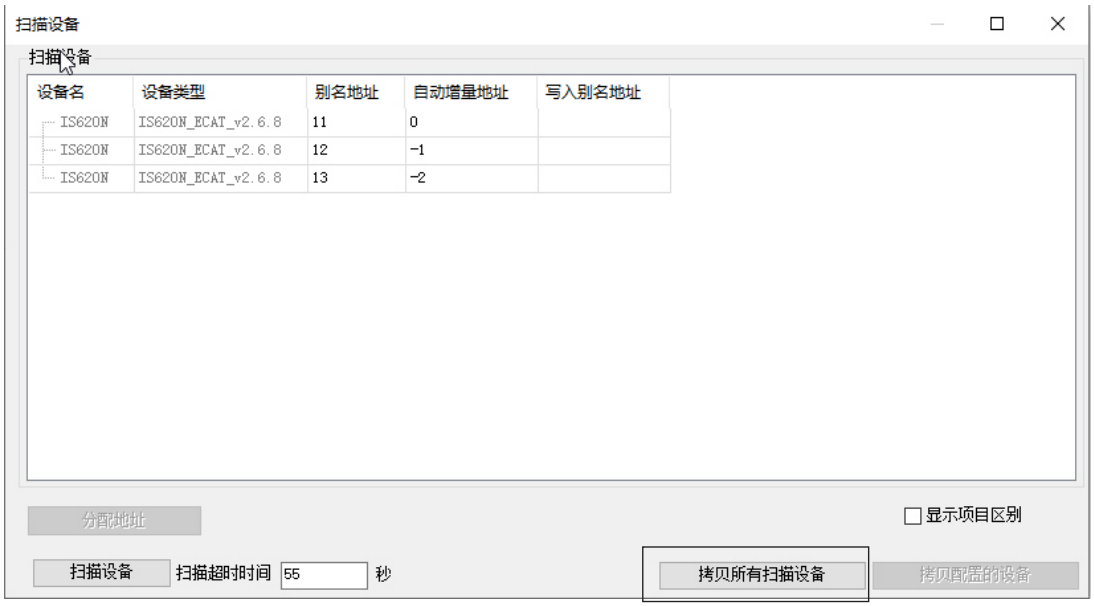
2） 图中的选项定义如下：

编号	选项	定义
1	扫描设备	执行扫描设备操作。
2	扫描超时时间	表示执行一次扫描操作所需要的最大超时时间时隔，扫描不到结果时，可以适当延长扫描超时时间，最小默认值是 20 秒。

■ 对扫描结果的操作

正常情况下，扫描结果如下图所示：

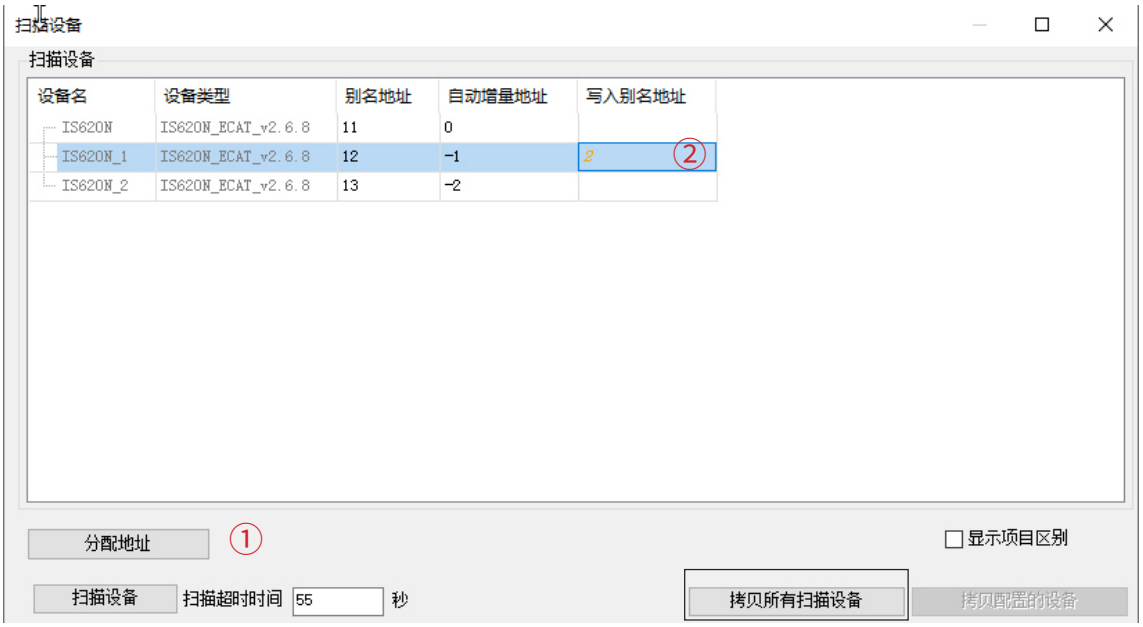
1） 复制所有设备



点击“拷贝所有扫描设备”即可完成扫描结果添加到设备树和组态。

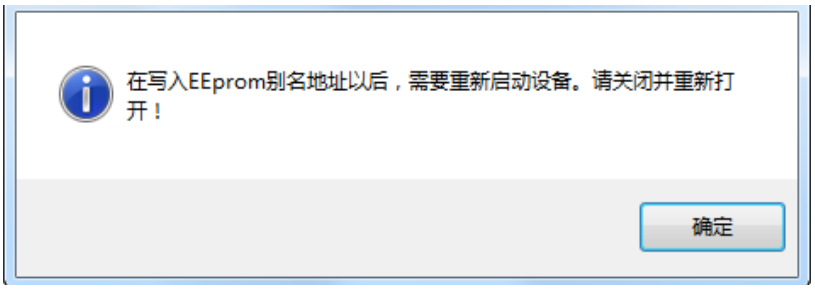
2） 分配别名地址

在“写入别名地址”栏①中，可通过双击编辑别名地址。别名地址修改完后，字体颜色会有所区分，如下图所示：

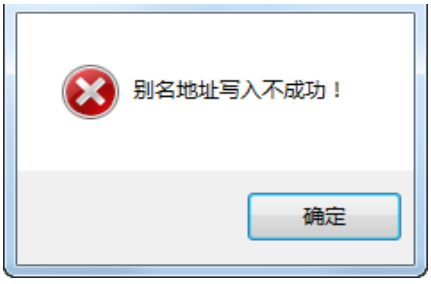


修改完别名地址，需要点击②所示的按键“分配地址”使别名地址生效。

如写入成功，系统会弹出如下提示：



如写入失败，则会弹出如下提示：



3） 显示项目区别

显示现有组态设备与扫描设备之间的区别，如下图所示：

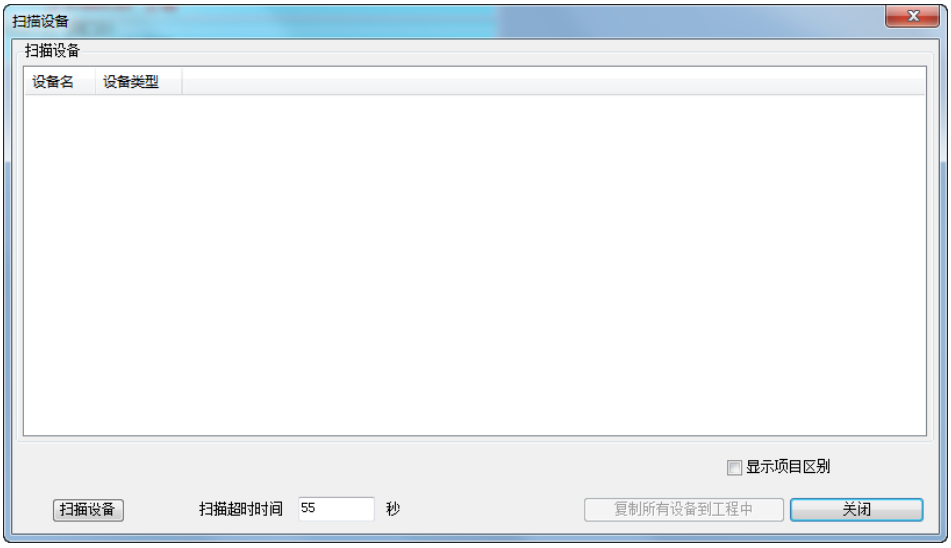
图中的编号定义如下表所示：

编号	名称	描述
1	复制之前	选择左侧的扫描设备和右侧现有设备后，选择①，则扫描设备会插入到右侧选择设备之前（模块之间也可以使用此命令）
2	复制之后	选择左侧的扫描设备和右侧现有设备后，选择②，则扫描设备会插入到右侧选择设备之后。（模块之间也可以使用此命令）
3	变为	选择左侧的扫描设备和右侧现有设备后（左侧设备必须和右侧设备为同一类设备），选择③，则右侧设备会变为扫描的设备。
4	全部复制	右侧设备清空，左侧扫描设备全部复制到右侧。
5	删除	选择右侧设备，删除单个设备。（模块也可以使用此命令）
6	拷贝所有扫描设备	拷贝所有扫描设备。点击后，拷贝配置的设备到工程。

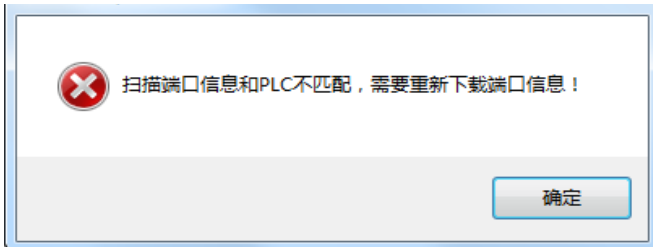
■ 扫描异常说明

1） PC 未连接 PLC，弹出如下提示：

2） PC 连接 PLC，但 PLC 没有连接从站：扫描不到从站时显示如下：



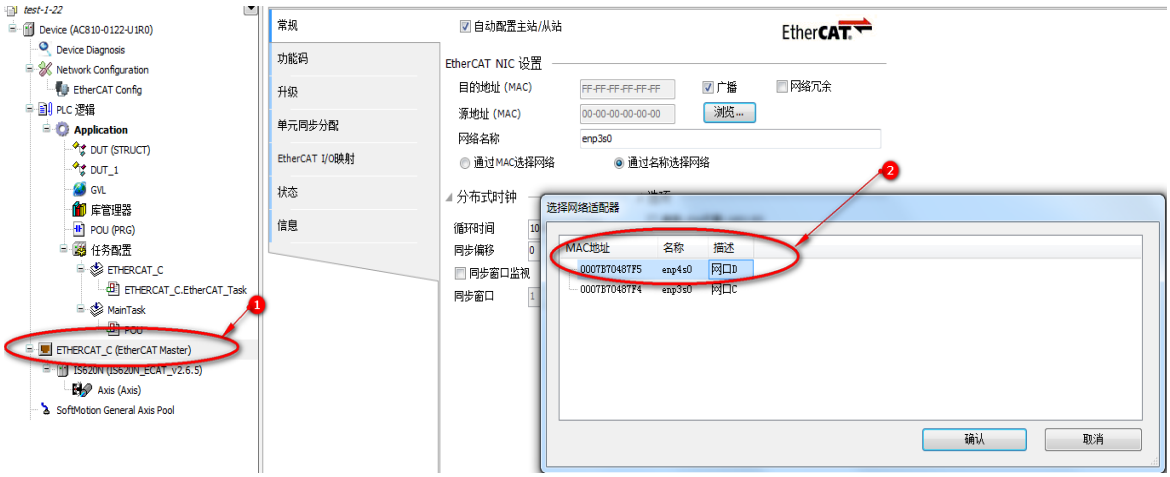
3） 后台组态端口信息和 PLC 信息不一致，弹出如下提示：



此时需要重新下载后台端口信息。

NOTE

如果如下图所示，组态选择的 EtherCAT_C, 但实际 MAC 变成了 EtherCAT_D，则可能引起扫描 C 口设备时，扫描结果为 D 口的设备。

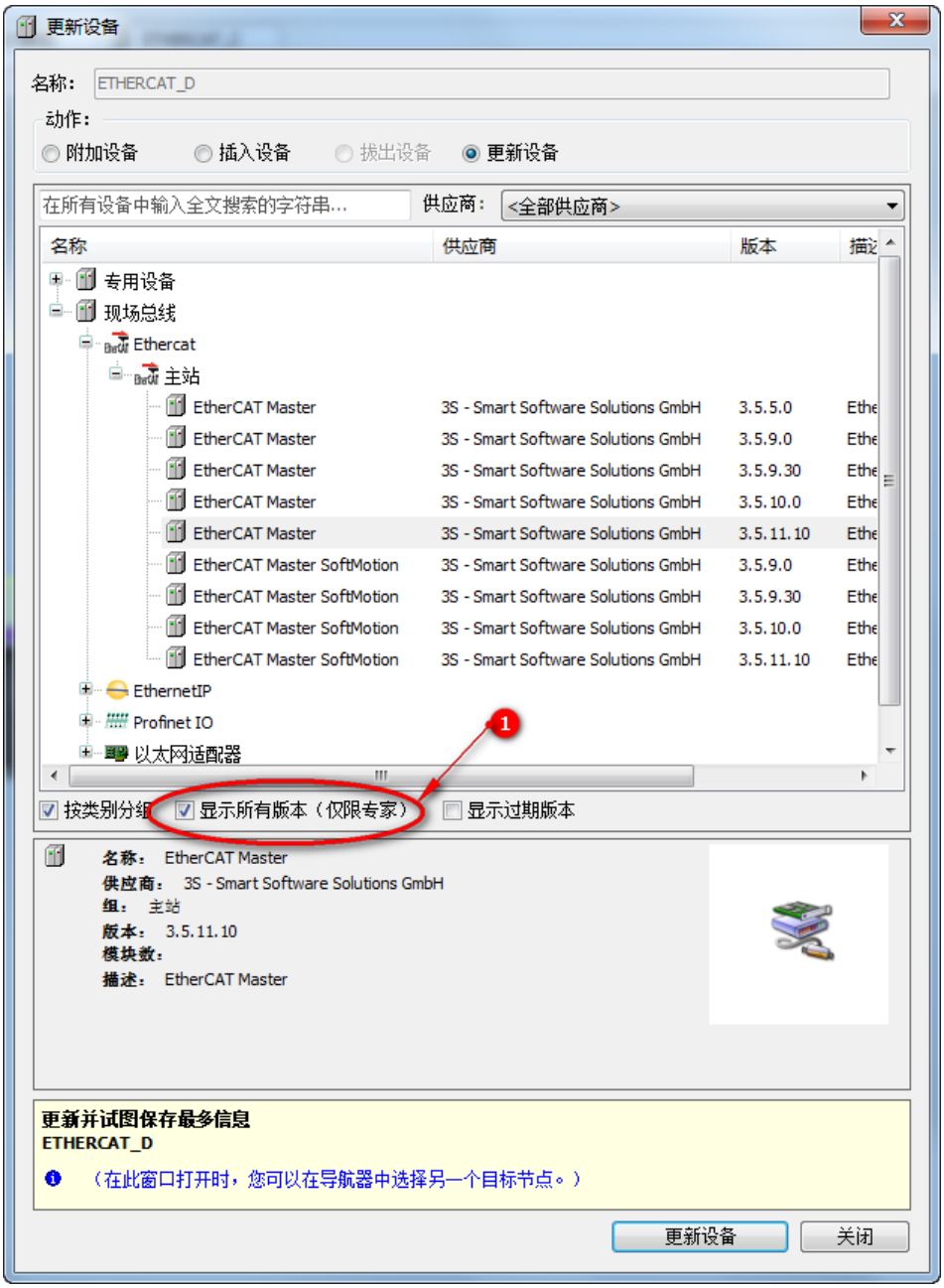


EtherCAT 总线正常启动后，若用户退出登录，在线执行“扫描设备”时扫描结果可能会失败或者不准确。原因是“扫描设备”过程中以 SDO 通讯方式读取从站信息（例如对象字典 0xF050）可能超时，导致影响扫描结果 (SDO 通讯属于异步通信，CPU 负载、总线循环周期、用户程序执行时间都会影响 SDO 通讯)。

3 更新设备

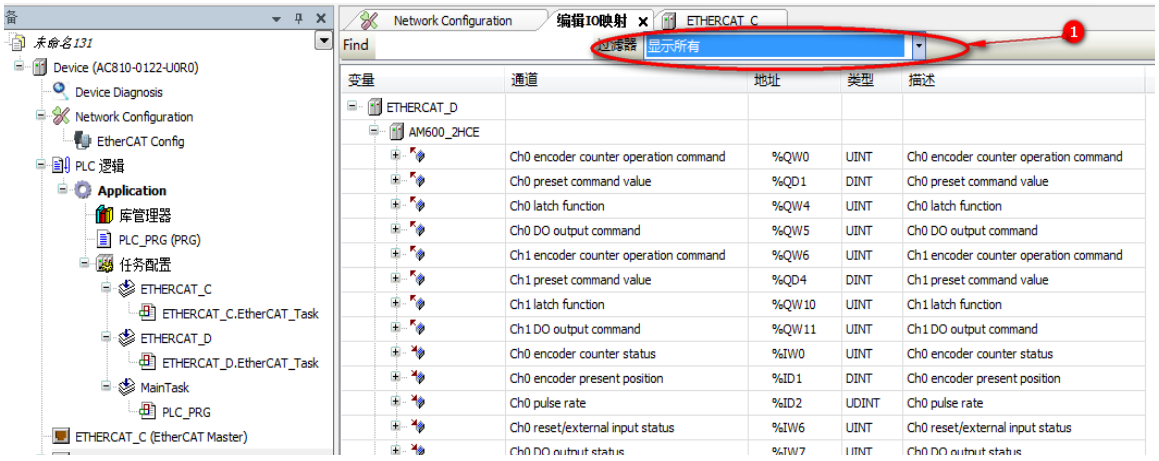
如果主站的版本不匹配或者需要升级，可以使用更新设备命令。

操作步骤：右键单击 EtherCAT 主站，在弹出的选项栏中选择“更新设备”。选中后会弹出如下对话框，勾选“显示所有版本（仅限专家）”可显示所有的版本，选择对应的版本后点击 “更新设备”：



4 编辑 IO 映射

选择“编辑 IO 映射”命令会弹出如下界面：



在“过滤器”中选择合适的过滤器，可以过滤掉不需要的显示选择项。

5 总线任务

PLC 所有 IEC 总线任务严格按照相同的逻辑顺序循环扫描执行，逻辑顺序可分为四个步骤：刷新输入（1）、执行 IEC 任务（2）、刷新输出（3）、执行总线循环（4），如下图所示：

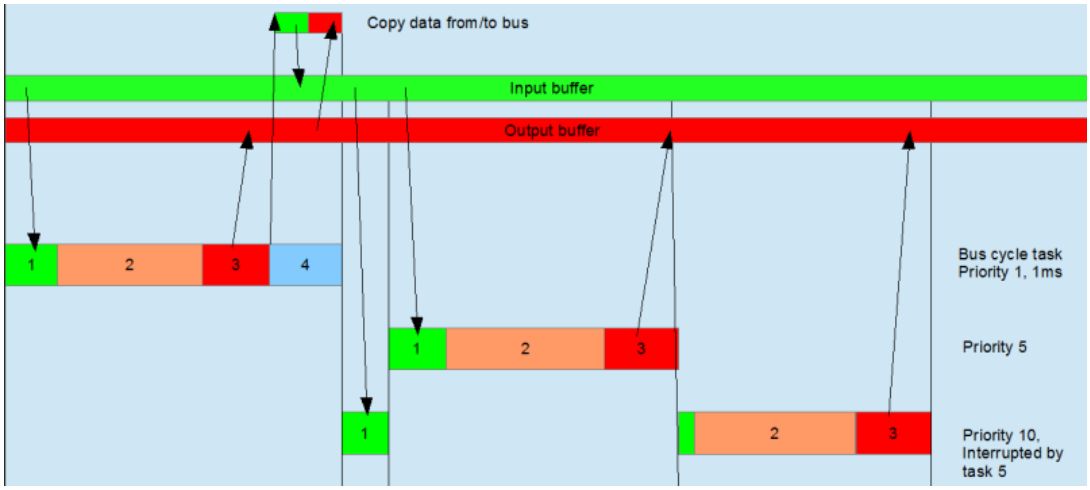


图 3-25 总线循环任务

各个步骤的详细说明如下表所示：

编号	步骤	图例	说明
1	刷新输入	绿	IEC 任务开始之前，从总线输入缓冲区读取数据，并将数据拷贝到与任务相关联的输入变量中。
2	执行 IEC 任务	橙	扫描执行总线任务下的 POU。
3	刷新输出	红	IEC 任务结束之前，将任务中与总线相关的输出变量拷贝到总线输出数据缓冲区。
4	总线循环	蓝	执行总线通讯程序，主要由底层 I/O 驱动实现，包含两个功能：将总线输出缓冲区数据传输到远端从站的数据接收缓冲区，以及将远端从站发送的缓冲区数据读取到总线输入缓冲区。

NOTE

◆ 警告！

1) 如果一个输出变量被多个任务使用，变量的值为不确定（任务此输出变量可能被其他任务修改、覆盖）；
2) 当一个任务被高优先级的任务打断时，高优先级任务从输入缓冲区读取数据，并将数据同步到当前任务中的输入变量，可能会造成同一个扫描周期内输入变量值不一致的问题。可以通过任务开始的时候复制输入变量值、任务调用复制的输入变量的方式进行避免。

EtherCAT 特殊的总线循环动作

上一个周期的总线数据会在 IEC 输入之前被拷贝。

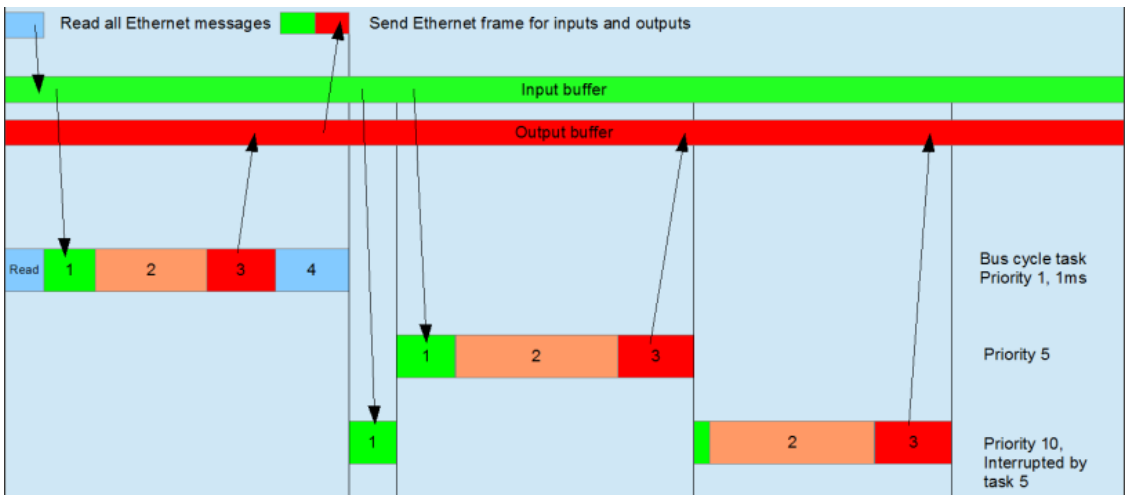
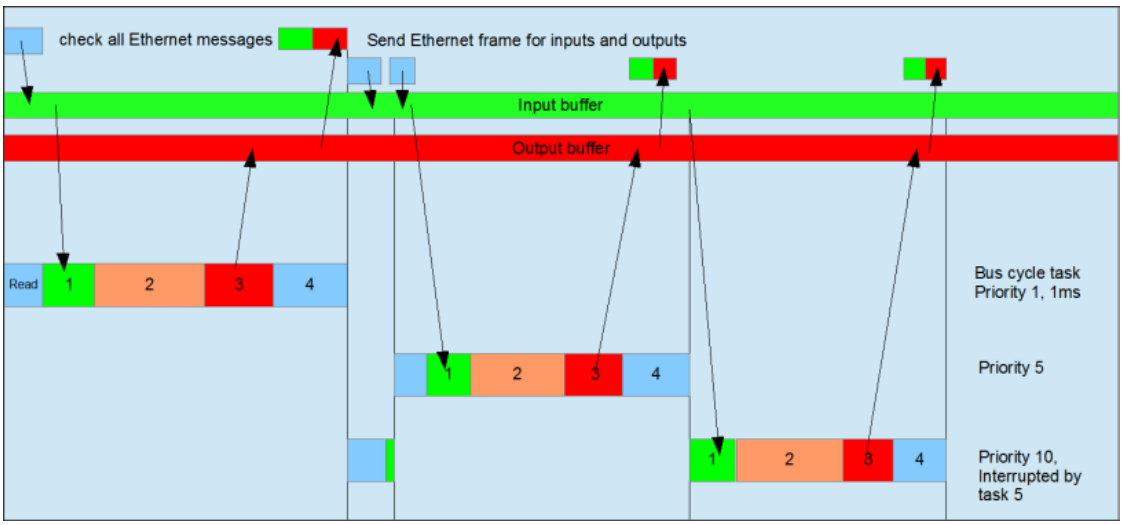


图 3-26 EtherCAT 总线周期表格

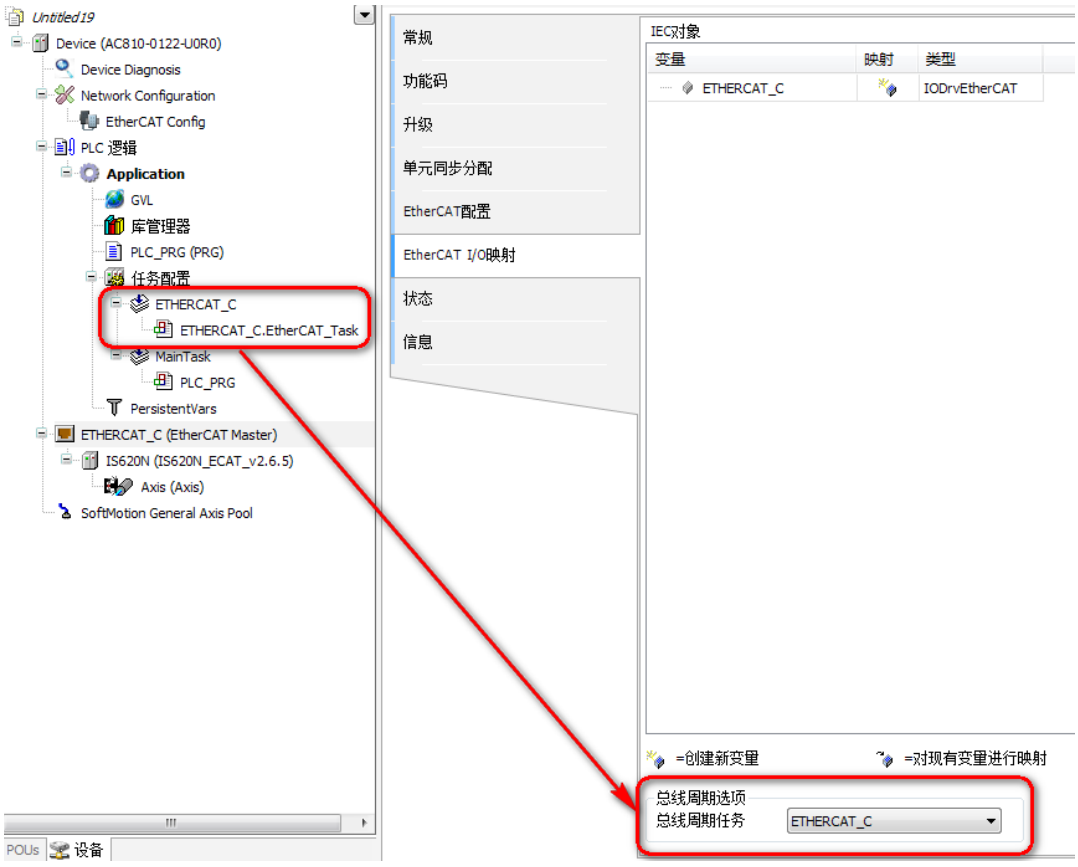
EtherCAT 主站设置 中的选项“激活每个任务上的消息”可以被激活。这种情况下每个任务的附加信息将会被发送到设备。这样总线通讯即可在多个任务下执行，并且这样还能够减少总线上的负载。

使用“激活每个任务消息”选项后的 EtherCAT 总线周期表格如下图所示：



NOTE

- EtherCAT 主站设备自动插入后，“EtherCAT_***”任务也自动插入到当前应用的任务配置中；
- EtherCAT 主站的总线周期任务必须与 EtherCAT_***.EtherCAT_Task 在同一任务中执行；
- EtherCAT 主站的输入输出与 EtherCAT_***.EtherCAT_Task 在同一任务中执行；EtherCAT 主站 I/O 映射中总线周期任务需设置对应的 EtherCAT 任务。因此，设备的控制程序（例如 PLCOpen 的轴控指令）推荐在该任务下执行。



3.3.3 EtherCAT 主站

1 常规

EtherCAT 主站配置对话框提供了“主站”的主要设置。

自动配置主站 / 从站

如果勾选该选项，主站和从站的主要配置将会自动完成。此时，所有从站设备编辑器将不显示 FMMU/Sync 设置选项卡。

NOTE

◆ 自动配置是默认选择，强烈推荐用于标准应用；
◆ 如果取消选中该选项，所有主站和从站设置都必须手动完成，这要求配置人员具备充分的专业知识。

EtherCAT NIC 设置

■ “目标地址 (MAC)”

接收报文的 EtherCAT 网络成员的 MAC 地址。如果选中“广播”选项，则使用广播地址（FF FF FF FF FF FF）。

■ “网络冗余”

如果使用环型拓扑，需要冗余，则激活此选项。使用此功能，如果网络连接出现单点故障，EtherCAT 网络功能依然可以保持正常工作。激活此选项，还需要定义用于冗余功能的第二个 EtherCAT NIC 设置。

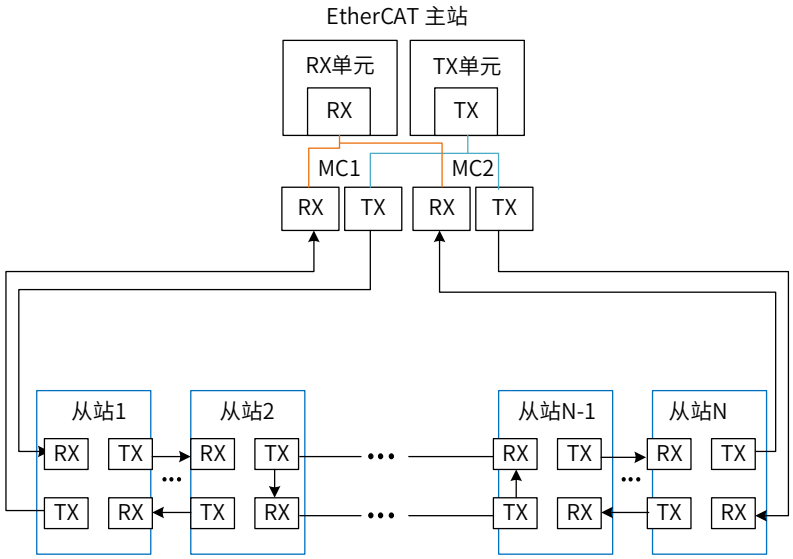


图 3-27 EtherCAT 环拓扑（冗余）

■ “源地址 (MAC)”

PLC 的 MAC 地址

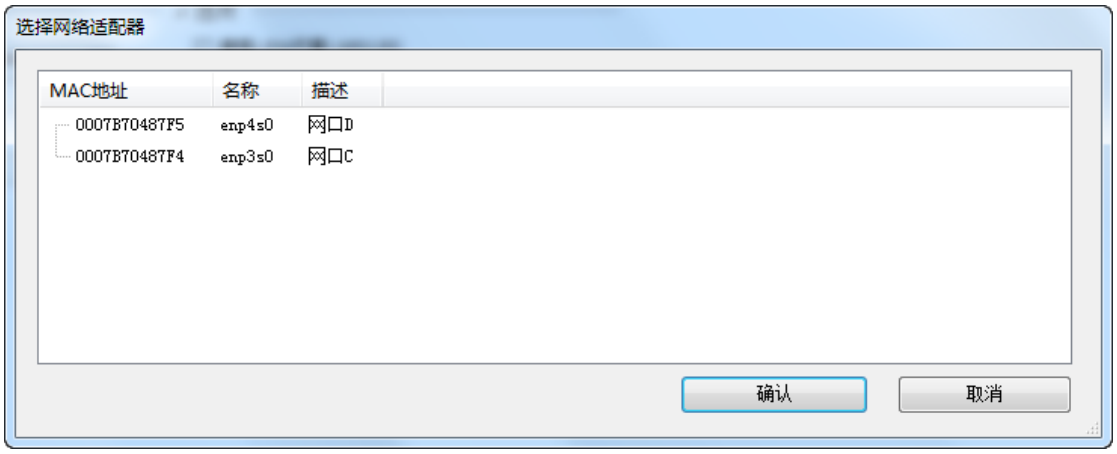
■ “网络名称”

网卡名称，可选择以下选项之一：

■ “通过 MAC 选择网络” / “通过名称选择网络”

每个 EtherCAT NIC 有唯一的 MAC 地址。因此，如果使用“通过 MAC 选择网络”，将不能在其他设备上使用此工程。

如果想使工程独立于设备，最好使用“通过名称选择网络”。在每个选项中，都可以使用“浏览…”选项得到当前可用的目标设备的 MAC 地址和名称，如下图所示：



■ 冗余 EtherCAT NIC 设置

如果“网络冗余”选项被激活，则会显示此设置。用户可根据‘冗余 EtherCAT NIC 设置’来定义相关的设置项。

分布式时钟

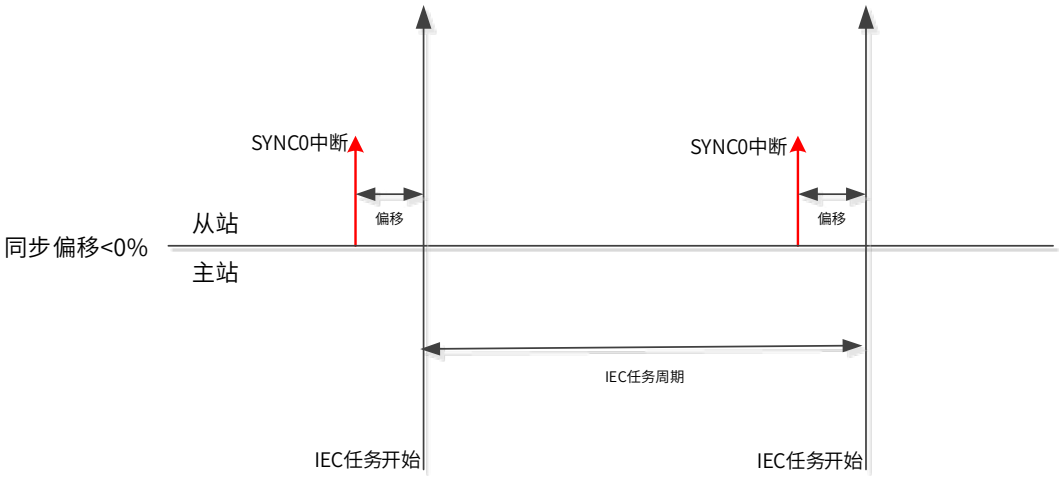
■ “循环时间 [µs]”

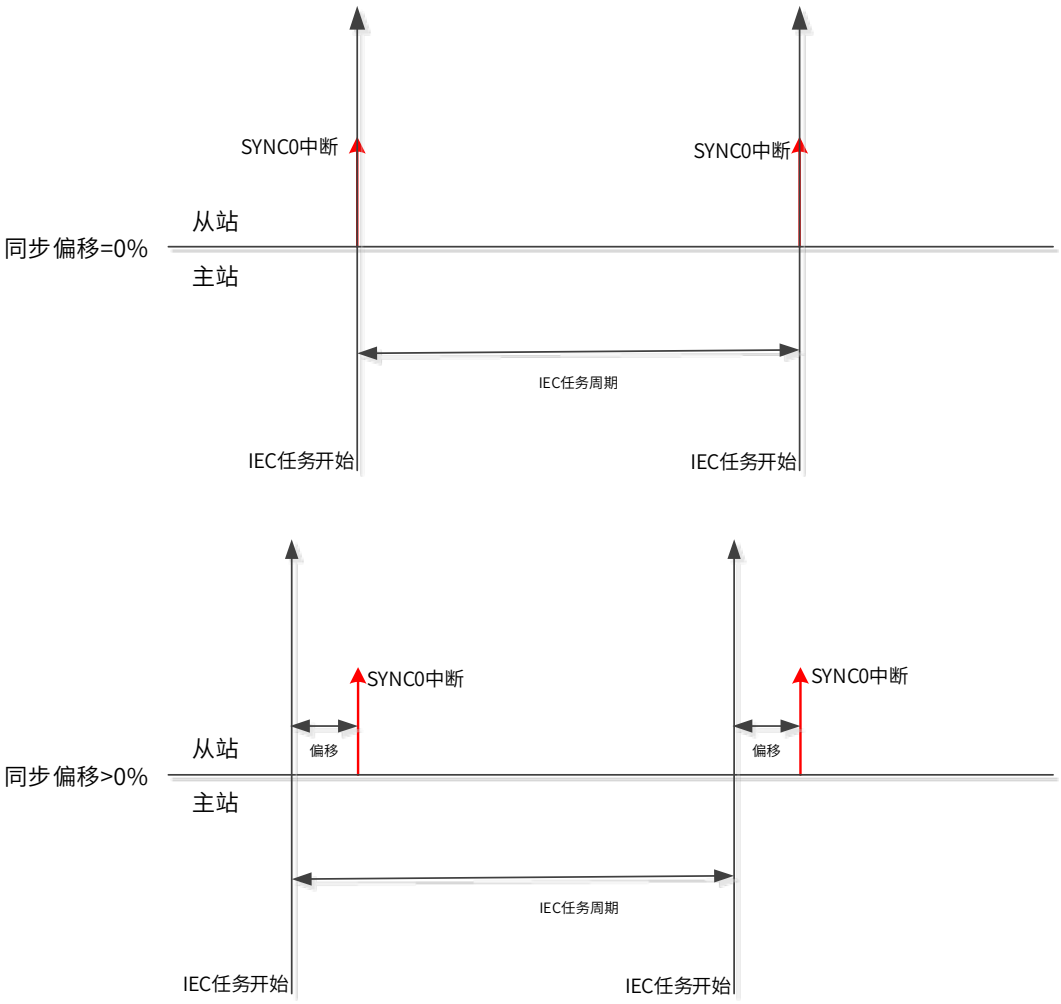
执行 EtherCAT 主站功能的周期时间。它必须与 EtherCAT 主站绑定的 IEC 任务的循环时间相同。如果从站“分布时钟”功能被激活，则该“循环时间”将同步刷新从站设备编辑器中“分布时钟”的设置值。

■ “同步偏移 [%]”

EtherCAT 主站 IEC 任务（PLC 任务）的循环时间相对于参考分布时钟（一般为 SYNC0 同步中断）偏移的比例，范围为 -50%~+50%，默认值：0%。

同步偏移 [%] = （SYNC0 中断时间 - PLC 任务循环开始时间）/ PLC 任务循环时间





NOTE

- 默认情况下，PLC 任务循环时间与从站的分布时钟的循环时间相等；
- 实际设置时，需考虑控制器的主站时钟抖动（系统实时性）、PLC 任务执行时间、PLC 任务循环时间的长短以及从站数等的影响。

■ “同步窗口监控”

激活此选项会允许监控从站的同步状况。

■ “同步窗口”

同步窗口监控的时间。如果所有从站同步信息都在这个时间窗口，则变量 xSyncInWindow (IoDrvEtherCAT) 被置成 TRUE, 否则置成 FALSE。

■ “诊断信息”

在线模式下，用于显示 EtherCAT 主站启动、运行等相关诊断信息。

选项

■ “使用 LRW 代替 LWR/LRD”

勾选此选项将使能从站 - 从站的通讯。EhterCAT 主站逻辑寻址将使用组合读 / 写命令（LRW）代替单独读（LRD）和单独写（LWR）命令。

■ “使能每个任务的信息”

勾选此选项，处理输入输出信息的读、写命令将由不同任务完成。

■ “自动重启从站”

勾选此选项，主站会在通讯异常后立即尝试重启从站。

主站设置

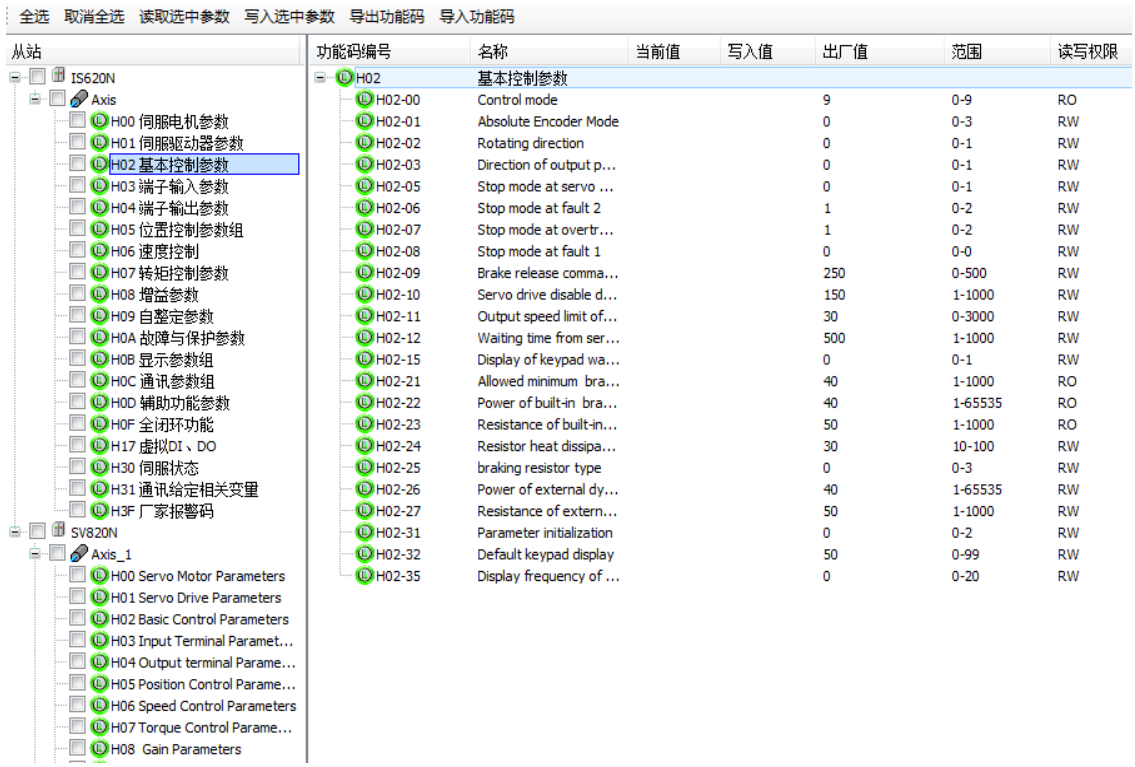
只有在自动模式无效的情况下，才可以进行这些设置（见下文），否则这些设置会自动完成并在此对话框中隐藏。

■ “输入映射地址”：用于输入数据的第一个从站的第一个逻辑地址。

■ “输出映射地址”：用于输出数据的第一个从站的第一个逻辑地址。

2 功能码

功能码指的是汇川（Inovance）伺服类产品定义的厂家参数。在主站选项卡下，可以批量读写、导入和导出多个产品的厂家参数，便于调试和维护。



■ 全选：选择所有的从站及轴，以及轴下边的伺服功能码。

■ 取消全选：取消所有的选择项。

■ 读取选中参数：在伺服运行状态下，读取具有 RO 或者 RW 属性的伺服功能码。

■ 写入选中参数：在伺服运行状态下，写入具有 WO 或者 RW 属性的伺服功能码。

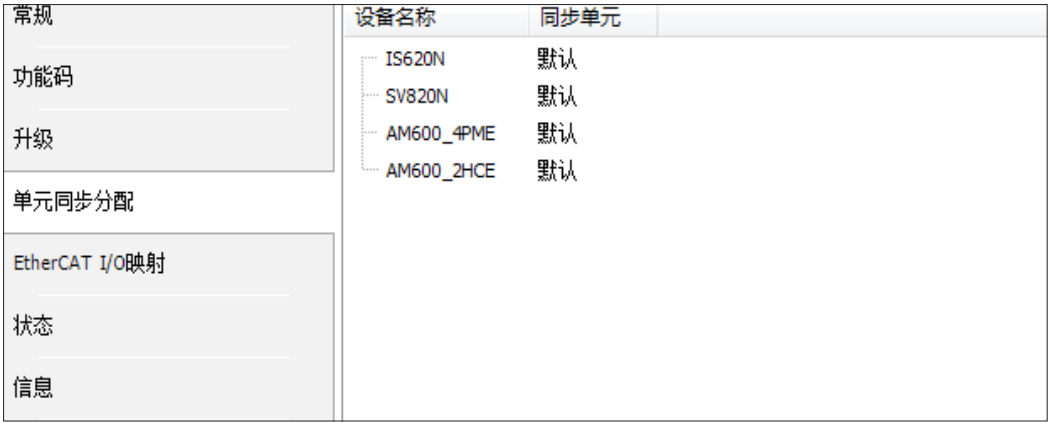
■ 导出功能码：导出所有从站的伺服功能码，格式为 Excel 文件格式。

■ 导入功能码：从外部 Excel 文件中，导入所有从站的功能码，如果组态与文件不一致，则报错。

3 升级

- 全选：选择所有的从站。
- 取消全选：取消所有选择的从站。
- 下载 EtherCAT XML 文件：将 InoProShop 中的 EtherCAT 从站 XML 文件下载到从站的 E2PROM 中。通过勾选多个从站可以实现批量下载。

4 单元同步分配



- 同步单元：可将从站分组，组中任一从站丢失，整个组都会丢失，但其他组不受影响。
- 增加：增加组别，之后可以在从站中选择相应的组。

5 EtherCAT I/O 映射

EtherCAT 主站配置编辑器的选项卡，其中为 EtherCAT I/O 指定了 IODrvEtherCAT 类型的实例（变量），这样与 EtherCAT 主站连接的 PLC 可以由用户程序控制。有关如何映射的描述，请参见本文档章节“I/O 映射”。

自动创建的主站实例显示在对话框“IEC 对象”的底部，可以用于应用中。

NOTE

备注：映射的变量和类型要一致。

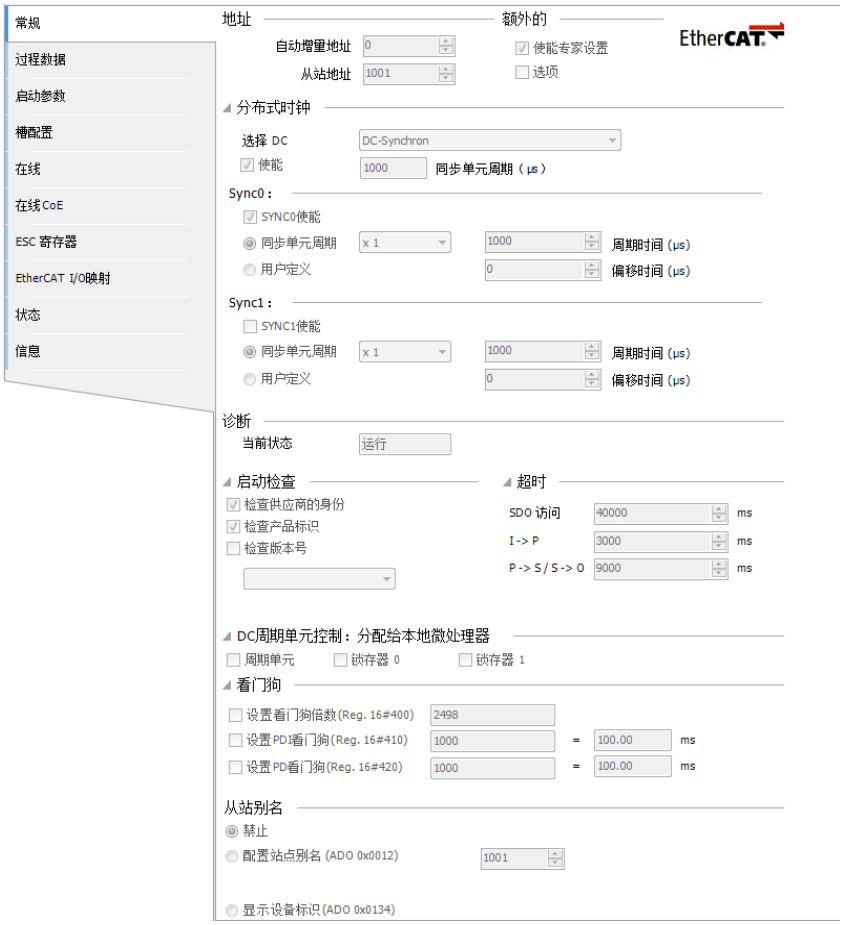
6 信息

此对话框显示当前模块的下列信息：名称、供应商、组、类型、ID、版本、等。

3.3.4 EtherCAT 从站

1 常规

EtherCAT 从站“常规”界面如下，其中提供了从站基本配置信息：



地址

如果主站设备编辑器中 “自动配置主站 / 从站” 没有勾选，则以下选项可设：

■ “自动增量地址”：自动增量地址（16 位），由网络中从站的物理拓扑位置确定。此地址只在 EtherCAT 主站启动时使用，通过顺序寻址的方式，将 EtherCAT 从站地址分配给相应物理拓扑位置的从站。

顺序寻址时，EtherCAT 协议标准，从站的自动增量地址由其在物理拓扑网络的连接位置确定，用一个负数来表示。顺序寻址会发送包含的子报文数据帧，数据帧经过每个从站设备时，子报文中的自动增量地址数据将自动加 1；物理从站在接收数据帧时，通过查找数据帧子报文中自动增量地址是否为 0 的方式，来寻址到自己的报文。这种数据帧经过从站更新报文内部地址变量的机制，被称为“顺序寻址”或者“自动增量寻址”。

■ “从站地址”：从站的配置地址（正名地址），由主站在启动时分配。此地址独立于网络中的实际位置，从站的地址与其在网段内的连接顺序无关。

■ “使能专家设置”：如果勾选此选项，将可以配置分布式时钟设置、启动时检查、超时设置、周期单元控制、看门狗等选项。

分布式时钟

■ “使能”如果勾选，表示使用“分布式时钟”功能，

■ 同步单元周期 (µs): 如果“分布式时钟”功能被使能，同步单元周期值将与 EtherCAT 主站的循环时间置相同。

■ “选择 DC”：该下拉菜单提供设备描述文件中所有关于分布式时钟的设置。一般包含自由运行、SM 事件同步、DC 同步。各项设置的功能如下表所述：

编号	设置	功能
1	自由运行模式	在该模式下，本地控制周期由一个本地定时器中断产生，周期时间可以由主站设定，这是从站的可选功能。
2	SM 事件同步	本地周期在发生数据输入或者输出事件的时候触发，主站可以将过程数据帧的发送周期写给从站，从站可以检查是否支持这个周期时间或对周期时间进行本地优化。从站可以选择支持这个功能 。通常同步于数据输出事件，如果从站只有输入数据，则同步于数据输入事件。
3	DC 同步	本地周期由 SYNC 事件触发，主站必须在 SYNC 事件之前完成数据帧的发送，为此要求主站时钟也要同步于参考时钟。
4	SYNC0: SYNC1:	从站同步信号 0/1。从站的分布时钟控制单元（EtherCAT 专用的通讯芯片的内部功能）可以产生两个同步信号，分别为 SYNC0 和 SYNC1，这两个信号给从站的应用层程序提供中断标志，或者直接触发输出数据更新
5	SYNC 0 使能: SYNC 1 使能:	勾选后将启动 SYNC0/SYNC1 的同步信号。 “同步单元周期”：如果选择这个选项，主站周期时间乘以选择的系数即为从站的同步周期时间。 “周期时间 (µs)”域显示当前设置的从站周期时间。

■ “用户定义”：如果选择此选项，可以在“循环时间 (µs)”字段输入微秒级的用户自定义同步周期时间。

诊断

此部分仅在在线模式下可见：

■ “当前状态”：显示从站的当前通讯状态机的状态，可能值为：初始化、预运行、安全运行、运行、引导（汇川的伺服暂时不支持）。如果当前状态是‘运行’，则表示从站配置已正确完成。

选项

“选项”：如果一个从站设备被定义为‘可选的’，则不会创建错误消息，以防止该设备不存在于总线系统中。若要激活此选项，必须在从站设备中存储一个站地址，因此必须在 E2PROM 中定义并写入“站点别名”地址。且只有在勾选 EtherCAT 主站设置中的“自动配置主站 / 从站”选项并且 EtherCAT 从站支持此功能的前提下，此选项才有效。

启动检查

默认情况下，系统启动时会根据当前配置自动检查供应商 ID 和产品 ID。如果检测到不匹配，总线会停止运行且不执行下一步操作。此设定是为了避免下载错误的配置。

超时

默认情况下，以下操作不定义为超时。不过，如要观察这些操作是否超过了特定时间，可以在这里设置时间：

■ “SDO 访问”：EtherCAT 主站协议中 SDO（服务数据对象）访问从站的超时时间。

■ “I -> P”：从站通讯状态机由‘初始化’转换到‘预运行’。

■ “P -> S / S-> O”: 从站通讯状态机由‘预运行’转换到‘安全运行’，或由‘安全运行’转换到‘运行’。

- DC 周期单元控制：分配给本地微处理器来选择分布时钟功能定义的选项。该控制功能已经在 EtherCAT 从站的寄存器 0x980 中完成，可能的设置为：周期单元、锁存器单元 0、锁存器单元 1。

看门狗

- “设置倍数”：设置看门狗定时器的倍频数，以确定其计时的最小增量单位。默认值：2498，看门狗定时器的最小计时增量单位为 100us。
- “设置 PDI 看门狗”：如果激活 PDI 看门狗，当 EtherCAT 从站的 PDI(物理设备接口) 通讯时间超过设置值，则会触发看门狗。
- “设置 SM 看门狗”：如果激活 SM (同步管理) 看门狗，当 EtherCAT 从站的 PD（过程数据）通讯时间超过设置值，则会触发看门狗。

从站别名

这些设置只有在勾选“选项”且从站设备支持别名地址（由设备描述文件定义）的情况下才有效。如果已配置从站相应的别名地址，调整其在物理拓扑网络的位置时，无需更改用户程序中的配置，从站仍可以正常运行。

注意，更改从站的别名地址后，必须重新下载用户程序才能使别名生效；另外，某些从站必须断电重启才能使别名生效，具体请参考相关从站的使用手册。

- “禁止”：如果配置此项，则表示该从站不会检测别名地址。
- “配置站点别名（ADO 0x0012）”: 在从站设备支持的情况下，且“选项”勾选的情况下，才可以在在线运行的状态下写入别名地址。
- “写入 E2PROM”：此命令只在在线模式下可见。它允许将定义的地址写入从站的 E2PROM。如果从站不支持，则此命令无效且从站不能以站点别名形式工作。
- “实际地址”：此栏目只在在线模式下可见，显示从站的实际地址。它可以用来检查‘写 E2PROM’的命令是否成功。
- “显示设备标识 ADO(0X0134)”：保留。
- “数据字”（2 字节）：保留。

2 FMMU/Sync

当主站的自动配置模式配置为无效时，此对话框仅由 EtherCAT 从站配置编辑器的选项卡提供。它显示由设备描述文件定义的从站 FMMUs（现场总线内存管理单元）和 Sync（同步管理器），这些设置可以修改，例如配置从站 - 从站通讯。


NOTE

这些都是高级设置，对标准应用而言一般不需要。

FMMU

此选项卡显示了用于处理过程数据的从站现场总线内存管理单元，其中定义了每个物理地址 (“Ph. GlobStartAddr”) 上的逻辑地址 (“Ph. Start Address”) 映射。通过编辑 FMMU 对话框中的“添加 ...”以及“编辑 ...”按键可以添加新单元。

同步管理器（Sync Manager）

此选项卡显示了从站的同步管理器。其中可定义每个可用同步管理器的类型 (Mailbox In, Mailbox Out, Inputs, Outputs)、物理起始地址、访问类型、缓冲区以及中断需要访问的物理地址。在同步管理器编辑器中，通过“添加”或“编辑”按键可以新增或修改同步管理。

FMMU

添加 编辑 删除

逻辑起始地址	长度(Byte)	起始位	结束位	物理起始地址	物理起始位	标志
16#02000000	12	0	7	16#1800	0	WE
16#01000000	28	0	7	16#1C00	0	RE

Sync Manager

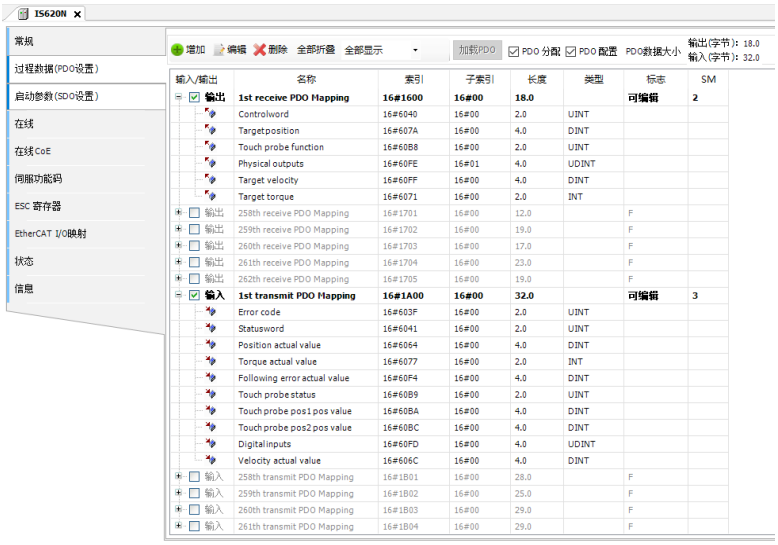
添加 编辑 删除

物理起始地址	长度(Byte)	标志	类型
16#1000	128	16#00010026 (1WPE)	邮箱输出
16#1400	128	16#00010022 (1RPE)	邮箱输入
16#1800	12	16#00010064 (3WPE)	输出
16#1C00	28	16#00010020 (3RPE)	输入

3 过程数据

实际自动化控制系统中，应用程序之间通常有两种数据交换形式：时间关键（time-critical）和非时间关键（non-time-critical）。时间关键表示特定动作必须在指定时间窗口内完成，如果不能在指定时间窗口内完成通信，则有可能导致控制失效。周期性发送时间关键数据的过程称为周期性过程数据通信（PDO），非时间关键数据可以非周期性发送，在 EtherCAT 中采用 MailBox 数据通信（SDO）。

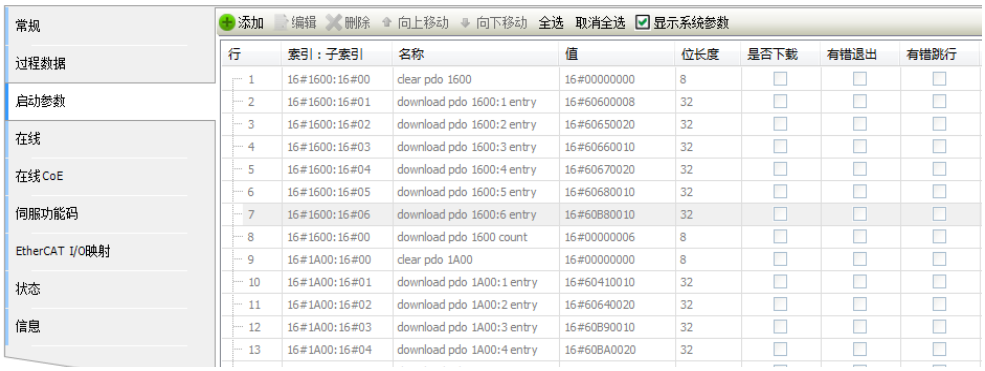
过程数据第一行为 PDO 编辑功能键及显示 PDO 信息，如下图所示：



- “增加” 根据当前 PDO 组的属性（仅限可编辑属性）增加 PDO 选择项。用户可以选择增加一组数据或多组数据，使用“Ctrl+”,” Shfit+” 鼠标左键可同时增加多组数据，请注意选择增加项的对象字典索引是否正确。注：PDO 增加的个数不能超过伺服手册中限制的个数。
- “编辑”根据当前 PDO 组的属性（仅限可编辑属性）编辑当前的 PDO 选择项。
- “删除”根据当前 PDO 组的属性（仅限可编辑属性）删除当前的 PDO 选择项。用户可以删除多个或者单个项，使用“Ctrl+”,” Shfit+”鼠标左键可同时选中多项，使用快捷键“Delete”或者鼠标右键删除。
- “全部折叠”： 折叠当前所有展开的 PDO 组。
- “过滤功能”：包含，全部显示，显示输出 PDO，显示输入 PDO，全部显示，显示输出和输入的 PDO，显示输出 PDO，只显示输出 PDO，显示输入 PDO，只显示输入 PDO 组。

- “加载 PDO”：只有在从站运行状态下，才可以将当前从站运行中的 PDO 组数据上传到后台。
- “PDO 分配”：勾选 PDO 分配，同时勾选“启动参数”界面的“显示系统参数”选择项，则启动参数组会增加当前输入和输出 PDO 组的分配信息，如下图所示：

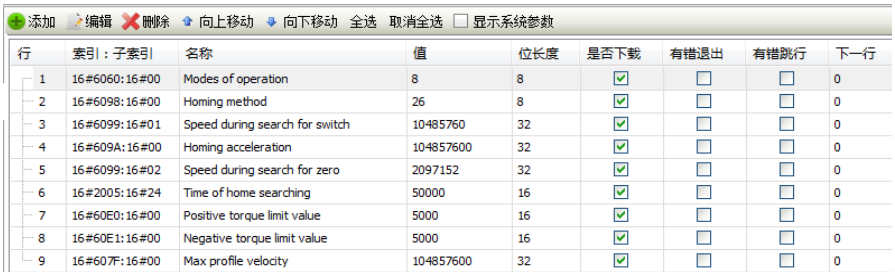
- “PDO 配置”：勾选 PDO 配置，同时勾选“启动参数”界面的“显示系统参数”选择项，则启动参数组会增加当前输入和输出 PDO 组的配置信息，如下图所示：



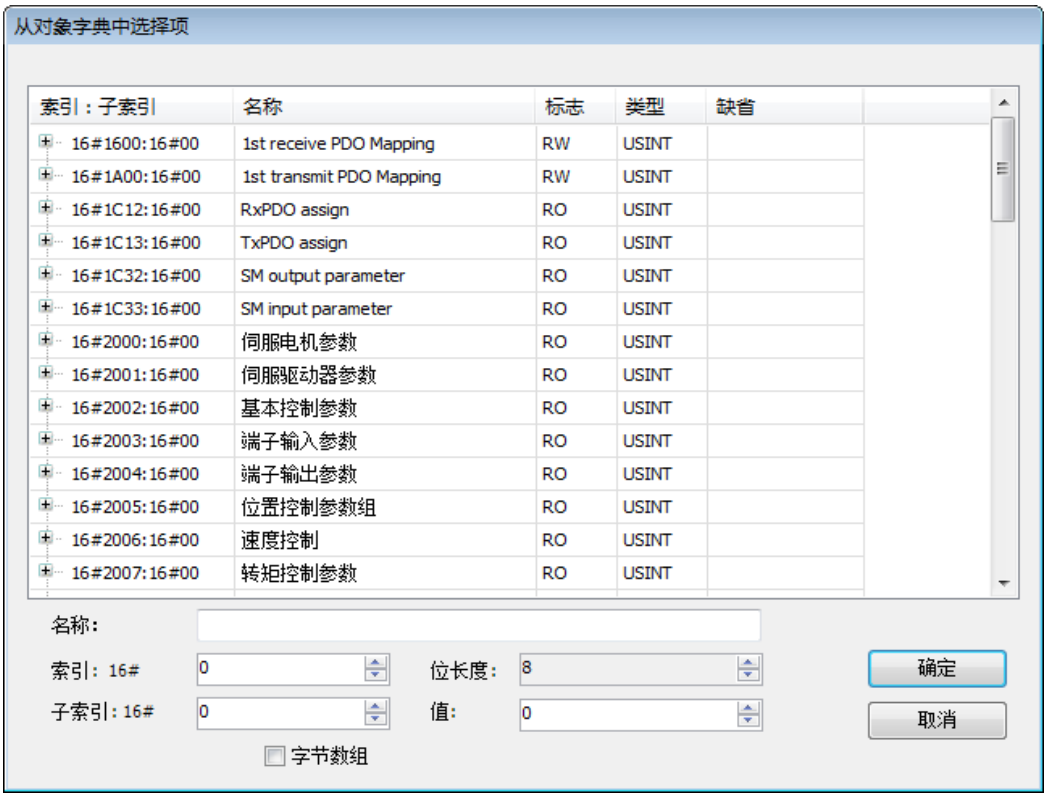
- “PDO 数据大小”显示当前所有输出和输入 PDO 总长度。

4 启动参数

启动参数在系统启动时可由 SDO（服务数据对象）传送给从站，启动参数包含了从站启动时所需的一些基本配置参数，常见界面如下：



- “添加”添加一个 SDO 项目到启动参数列表中，弹出的对象字典选择框如下所示：



添加 SDO 之前，可以通过编辑栏下方的栏目修改其参数，定义其索引、子索引、位长度及值，从而形成一个新的启动参数。可以使用组合按键“Ctrl+”,” Shfit+” 鼠标左键同时添加多组启动参数。

- “编辑”可以编辑当前选择项的参数，只读参数是无法编辑的，例如系统参数。
- “删除”可以删除当前选择项，使用组合按键“Ctrl+”,” Shfit+” 鼠标左键同时选中多组启动参数，然后使用快捷键 DEL 或者删除按键删除。
- 向上移动，向下移动

SDO 列表的顺序（由上至下）代表了启动参数被传输到模块的顺序。通过“向上移动”和“向下移动”按钮可以改变其传输到模块的先后顺序。

- 全选，取消全选

SDO 下载后，可以不用重复下载。使用取消全选（系统参数无法取消）可取消下载属性，或者勾选部分需要下载的属性；使用全选可全部设置为下载属性。

- 显示系统参数

显示 PDO 分配和 PDO 配置勾选后，SDO 中增加的参数，只做显示对比使用。

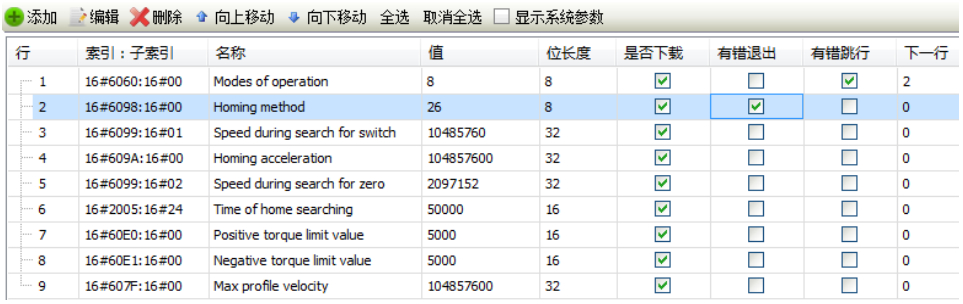
- SDO 部分的“值”和“注释”栏，可以通过按 [space] 或点击鼠标对其进行直接编辑。

NOTE

SDO 传输过程出现错误时，可以通过如下步骤解决：

1) “有错退出”：如果检测到错误，则退出 SDO 传输；

2) “有错跳行”以及“下一行”：如果检测到错误，则 SDO 传输会跳转到“下一行”所指的行号。（行号，行号显示于行栏目中）中输入的行中继续进行。



行	索引：子索引	名称	值	位长度	是否下载	有错退出	有错跳行	下一行
1	16#6060:16#00	Modes of operation	8	8	☑	☐	☑	2
2	16#6098:16#00	Homing method	26	8	☑	☑	☐	0
3	16#6099:16#01	Speed during search for switch	10485760	32	☑	☐	☐	0
4	16#609A:16#00	Homing acceleration	104857600	32	☑	☐	☐	0
5	16#6099:16#02	Speed during search for zero	2097152	32	☑	☐	☐	0
6	16#2005:16#24	Time of home searching	50000	16	☑	☐	☐	0
7	16#60E0:16#00	Positive torque limit value	5000	16	☑	☐	☐	0
8	16#60E1:16#00	Negative torque limit value	5000	16	☑	☐	☐	0
9	16#607F:16#00	Max profile velocity	104857600	32	☑	☐	☐	0

图 3-28 错误处理配置示例

5 槽配置

■ “槽配置”用于支持 ETG5001.1 协议的从站，是配置其模块或功能的选项卡。

如下图所示，当前的模式为 CSP_CSV：

用户可选以下模式：

编号	模式	定义
1	CSP	表示同步位置
2	PP	轮廓位置
3	CSV	同步速度
4	PV	轮廓速度
5	CST	同步力矩
6	PT	轮廓力矩

如果位置速度模式不满足实际需求，可切换成其他模式，如 CST“同步力矩模式”。

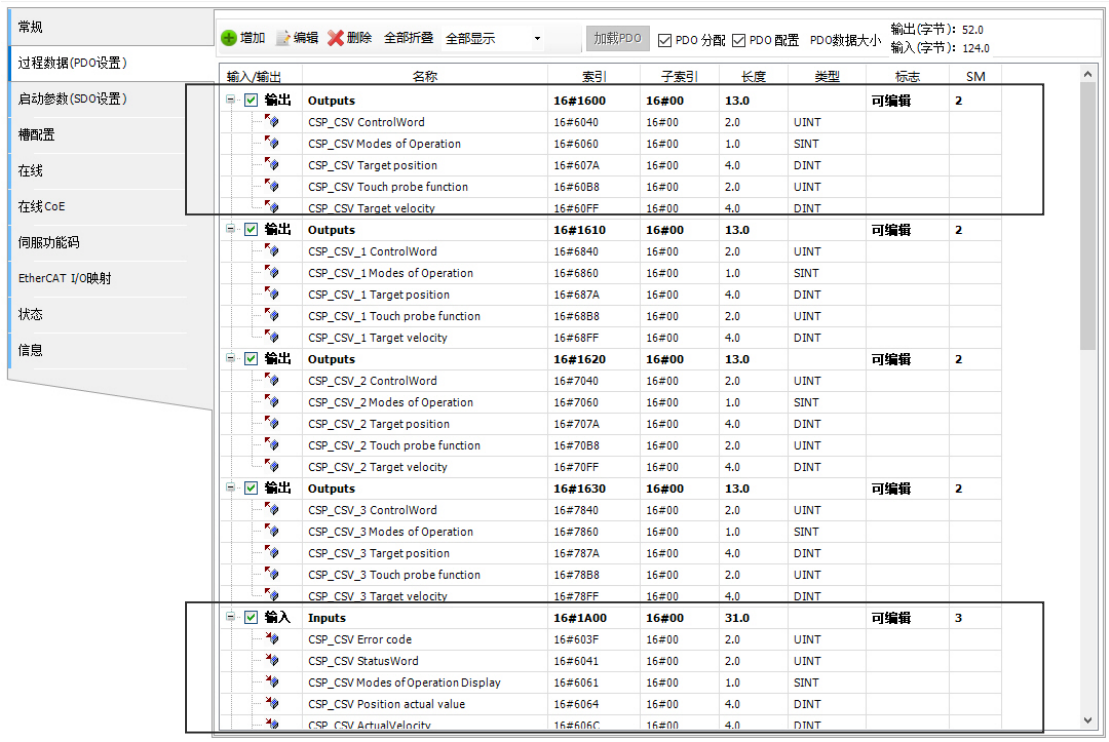
默认的“CSP/CSV”满足大部分应用场合，如果现场驱动轴要同时使用同步位置和同步力矩功能，请选择“CSP/CST”项。

NOTE

◆ “变更”用于切换不同的模式；
◆ “删除”可以删除当前的模式；
◆ 如需变更新的模式，需要先删除当前槽模式。

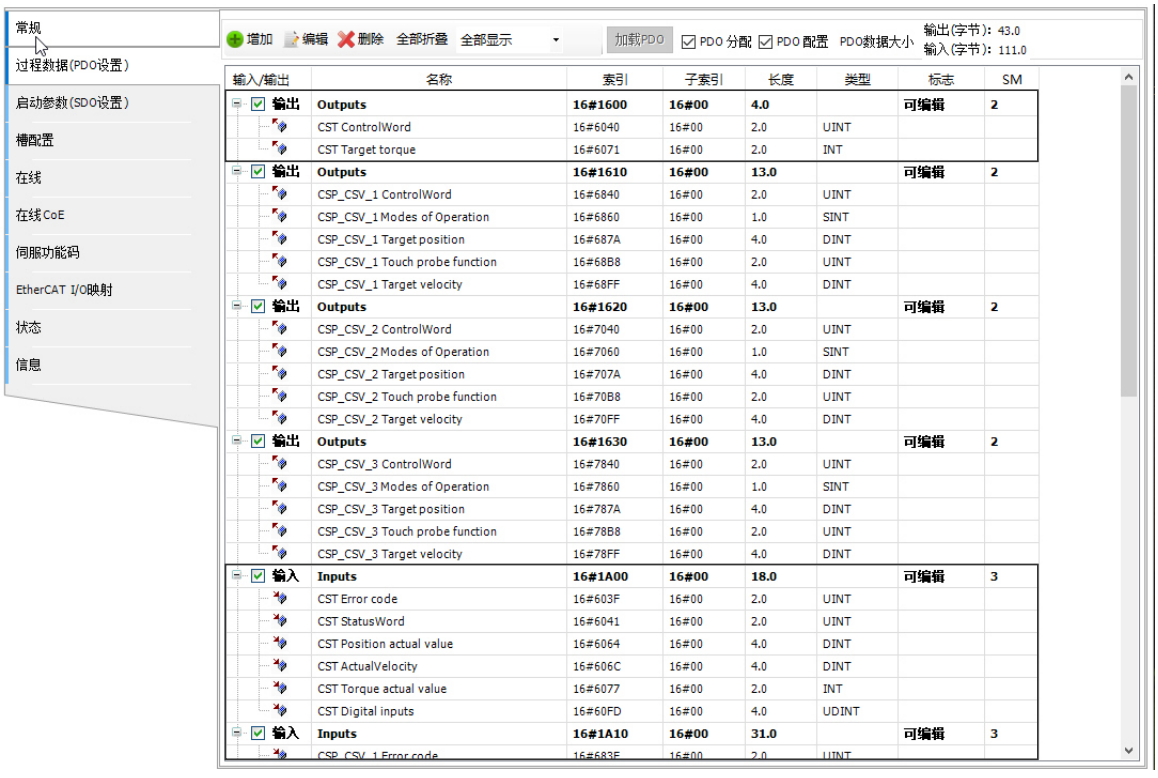
由 CSP_CSV 变更为 CST 模式后，过程数据和 I/O 映射都会有相应变化：

变更为 CST 前：



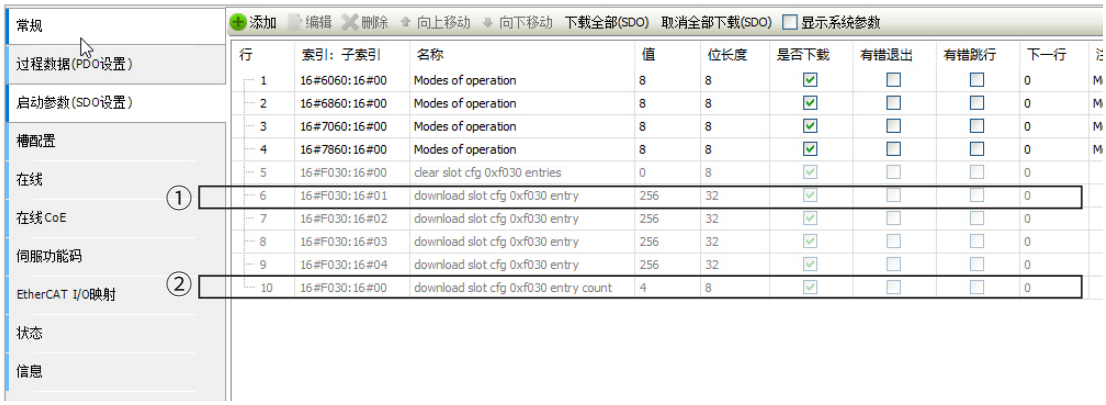
输入/输出	名称	索引	子索引	长度	类型	标志	SM
输出	**Outputs**	**16#1600**	**16#00**	**13.0**		**可编辑**	**2**
	CSP_CSV ControlWord	16#6040	16#00	2.0	UINT		
	CSP_CSV Modes of Operation	16#6060	16#00	1.0	SINT		
	CSP_CSV Target position	16#607A	16#00	4.0	DINT		
	CSP_CSV Touch probe function	16#60B8	16#00	2.0	UINT		
	CSP_CSV Target velocity	16#60FF	16#00	4.0	DINT		
输出	**Outputs**	**16#1610**	**16#00**	**13.0**		**可编辑**	**2**
	CSP_CSV_1 ControlWord	16#6840	16#00	2.0	UINT		
	CSP_CSV_1 Modes of Operation	16#6860	16#00	1.0	SINT		
	CSP_CSV_1 Target position	16#687A	16#00	4.0	DINT		
	CSP_CSV_1 Touch probe function	16#68B8	16#00	2.0	UINT		
	CSP_CSV_1 Target velocity	16#68FF	16#00	4.0	DINT		
输出	**Outputs**	**16#1620**	**16#00**	**13.0**		**可编辑**	**2**
	CSP_CSV_2 ControlWord	16#7040	16#00	2.0	UINT		
	CSP_CSV_2 Modes of Operation	16#7060	16#00	1.0	SINT		
	CSP_CSV_2 Target position	16#707A	16#00	4.0	DINT		
	CSP_CSV_2 Touch probe function	16#70B8	16#00	2.0	UINT		
	CSP_CSV_2 Target velocity	16#70FF	16#00	4.0	DINT		
输出	**Outputs**	**16#1630**	**16#00**	**13.0**		**可编辑**	**2**
	CSP_CSV_3 ControlWord	16#7840	16#00	2.0	UINT		
	CSP_CSV_3 Modes of Operation	16#7860	16#00	1.0	SINT		
	CSP_CSV_3 Target position	16#787A	16#00	4.0	DINT		
	CSP_CSV_3 Touch probe function	16#78B8	16#00	2.0	UINT		
	CSP_CSV_3 Target velocity	16#78FF	16#00	4.0	DINT		
输入	**Inputs**	**16#1A00**	**16#00**	**31.0**		**可编辑**	**3**
	CSP_CSV Error code	16#603F	16#00	2.0	UINT		
	CSP_CSV StatusWord	16#6041	16#00	2.0	UINT		
	CSP_CSV Modes of Operation Display	16#6061	16#00	1.0	SINT		
	CSP_CSV Position actual value	16#6064	16#00	4.0	DINT		
	CSP_CSV ActualVelocity	16#606C	16#00	4.0	DINT		

变更为 CST 后：



输入/输出	名称	索引	子索引	长度	类型	标志	SM
输出	**Outputs**	**16#1600**	**16#00**	**4.0**		**可编辑**	**2**
	CST ControlWord	16#6040	16#00	2.0	UINT		
	CST Target torque	16#6071	16#00	2.0	INT		
输出	**Outputs**	**16#1610**	**16#00**	**13.0**		**可编辑**	**2**
	CSP_CSV_1 ControlWord	16#6840	16#00	2.0	UINT		
	CSP_CSV_1 Modes of Operation	16#6860	16#00	1.0	SINT		
	CSP_CSV_1 Target position	16#687A	16#00	4.0	DINT		
	CSP_CSV_1 Touch probe function	16#68B8	16#00	2.0	UINT		
	CSP_CSV_1 Target velocity	16#68FF	16#00	4.0	DINT		
输出	**Outputs**	**16#1620**	**16#00**	**13.0**		**可编辑**	**2**
	CSP_CSV_2 ControlWord	16#7040	16#00	2.0	UINT		
	CSP_CSV_2 Modes of Operation	16#7060	16#00	1.0	SINT		
	CSP_CSV_2 Target position	16#707A	16#00	4.0	DINT		
	CSP_CSV_2 Touch probe function	16#70B8	16#00	2.0	UINT		
	CSP_CSV_2 Target velocity	16#70FF	16#00	4.0	DINT		
输出	**Outputs**	**16#1630**	**16#00**	**13.0**		**可编辑**	**2**
	CSP_CSV_3 ControlWord	16#7840	16#00	2.0	UINT		
	CSP_CSV_3 Modes of Operation	16#7860	16#00	1.0	SINT		
	CSP_CSV_3 Target position	16#787A	16#00	4.0	DINT		
	CSP_CSV_3 Touch probe function	16#78B8	16#00	2.0	UINT		
	CSP_CSV_3 Target velocity	16#78FF	16#00	4.0	DINT		
输入	**Inputs**	**16#1A00**	**16#00**	**18.0**		**可编辑**	**3**
	CST Error code	16#603F	16#00	2.0	UINT		
	CST StatusWord	16#6041	16#00	2.0	UINT		
	CST Position actual value	16#6064	16#00	4.0	DINT		
	CST ActualVelocity	16#606C	16#00	4.0	DINT		
	CST Torque actual value	16#6077	16#00	2.0	INT		
	CST Digital inputs	16#60FD	16#00	4.0	UDINT		
输入	**Inputs**	**16#1A10**	**16#00**	**31.0**		**可编辑**	**3**
	CSP_CSV_1 Error code	16#683F	16#00	2.0	UINT		

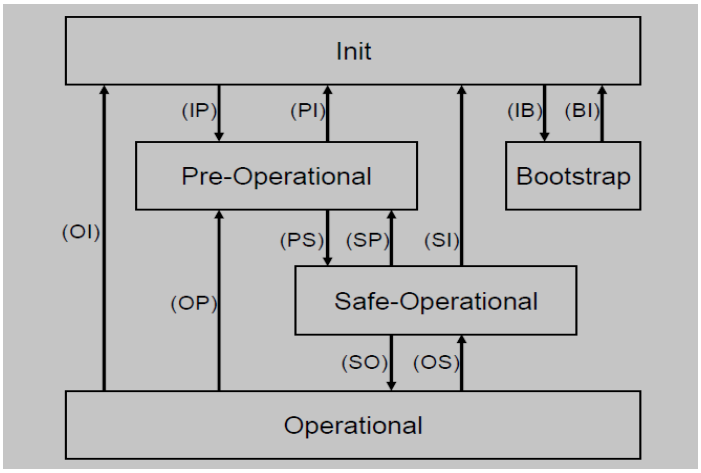
■ “下载槽配置”：配置完当前模块的后，需要把当前配置的槽信息下载到设备中。勾选此选择项后会在启动参数中增加如下参数：



6 在线

登陆到设备后才可使用从站在线配置编辑器。这个界面主要用于手动切换从站的状态机、读写从站的 E2PROM、以及 FoE 协议的上传和下载和从站固件升级。

EtherCAT 状态机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，如下图所示：



EtherCAT 运行转换顺序：初始化 -> 预运行 -> 安全运行 -> 运行。

状态过程详细描述：

状态转换	本地管理服务
IP	开始邮箱通信（Start Mailbox Communication）
PI	停止邮箱通信（Stop Mailbox Communication）
PS	开始输入更新（Start Input Update）
SP	停止输入更新（Stop Input Update）
SO	开始输出更新（Start Output Update）
OS	停止输出更新（Stop Output Update）
OP	停止输出更新，停止输入更新（Stop Output Update, Stop Input Update）
SI	停止输出更新，停止邮箱通信（Stop Input Update, Stop Mailbox Communication）
OI	停止输出更新，停止输入更新，停止邮箱通信（Stop Output Update, Stop Input Update, Stop Mailbox Communication）
IB	开始引导模式（Start Bootstrap Mode）
BI	重启设备（Restart Device）

“初始化”，“预运行”，“安全运行”，“运行”以及“清除错误”按键可以用来进行调试。

EtherCAT 文件访问

如果想要向从站传输固件文件，进行固件升级，可以点击“引导状态”按钮来将从站转换为“引导模式”。

使用相应按钮可以完成对固件文件的“下载”和“上传”。此时，会打开一个保存或打开固件文件的对话框，需要使用字符串和密码来执行文件的传输。此信息由从站设备提供，并记录在从站的数据表中，升级固件过程中，不建议断电或者进行状态切换，升级完成后，再进行此类操作。

E2PROM 访问

从站配置可以由 E2PROM 读取，也可写入 E2PROM。同固件文件传输一样，该操作也会弹出一个打开或保存文件的对话框。

用户可以利用“写 E2PROM XML”的命令直接将从站配置从 XML 文件写入到设备中。只有在 XML 文件中存在配置数据时，此命令才有效 (< 配置数据 > 部分)。

清除错误

当前状态有错误时，可以使用此按键，清除当前的错误状态。

7 在线 COE

只在总线正常运行并且登录 PLC 后才可以读取在线 COE 的值，如下图所示：

Read this page | Auto Update | Offline from ESI file | Online from device

索引：子索引	名称	标志	类型	值
16#1000:16#00	Device type	RO	UDINT	---
16#1001:16#00	Error Register	RO	USINT	---
16#1008:16#00	Device name	RO	STRING(31)	
16#1009:16#00	Hardware version	RO	STRING(4)	
16#100A:16#00	Software version	RO	STRING(4)	
16#1018:16#00	Identity	RO	USINT	
16#1600:16#00	1st receive PDO Mapping	RW	USINT	
16#1701:16#00	258th receive PDO Mapping	RO	USINT	
16#1702:16#00	259th receive PDO Mapping	RO	USINT	
16#1703:16#00	260th receive PDO Mapping	RO	USINT	
16#1704:16#00	261th receive PDO Mapping	RO	USINT	
16#1705:16#00	262th receive PDO Mapping	RO	USINT	
16#1A00:16#00	1st transmit PDO Mapping	RW	USINT	
16#1B01:16#00	258th transmit PDO Mapping	RO	USINT	
16#1B02:16#00	259th transmit PDO Mapping	RO	USINT	
16#1B03:16#00	260th transmit PDO Mapping	RO	USINT	
16#1B04:16#00	261th transmit PDO Mapping	RO	USINT	
16#1C00:16#00	Sync manager type	RO	USINT	
16#1C12:16#00	RxPDO assign	RO	USINT	
16#1C13:16#00	TxPDO assign	RO	USINT	
16#1C32:16#00	SM output parameter	RO	USINT	
16#1C33:16#00	SM input parameter	RO	USINT	
16#2000:16#00	伺服电机参数	RO	USINT	
16#2001:16#00	伺服驱动器参数	RO	USINT	
16#2002:16#00	基本控制参数	RO	USINT	
16#2003:16#00	端子输入参数	RO	USINT	
16#2004:16#00	端子输出参数	RO	USINT	
16#2005:16#00	位置控制参数组	RO	USINT	
16#2006:16#00	速度控制	RO	USINT	

8 EOE 设置

EtherNET over EtherCAT 允许将任意 EtherNET（以太网）设备通过转换端子连接到 EtherCAT，同时不影响 EtherCAT 的实时性。类似于众所周知的互联网协议（例如 TCP/IP、VPN、PPPoE(DSL)），以太网帧通过 EtherCAT 协议传输。该设置允许将标准网络设备通过交换机连接到终端设备，如打印机或 PC 机。

对于支持 EtherNET over EtherCAT（EOE）的特殊从站，可以进行通讯设置，且只有在设备支持 EtherNET over EtherCAT 时，才会提供以下的对话框。

- “虚拟以太网端口”：此选项使能从站的 EOE 功能。如果激活此选项，则必须定义一个特殊的“虚拟 MAC 地址”。
- “交换机端口”：此设备作为 IP 端口，必须设置以太网通讯参数。
- “IP 站”：此设备作为 IP 端口，必须设置以太网通讯参数。

以太网通讯参数必须基于虚拟以太网适配器的参数设置。“IP 地址”，“子网掩码”以及“默认网关”中的条目各分配 4 字节以识别网络中的从站。将光标定位在相应的编辑区域后，可以对默认设置进行修改。

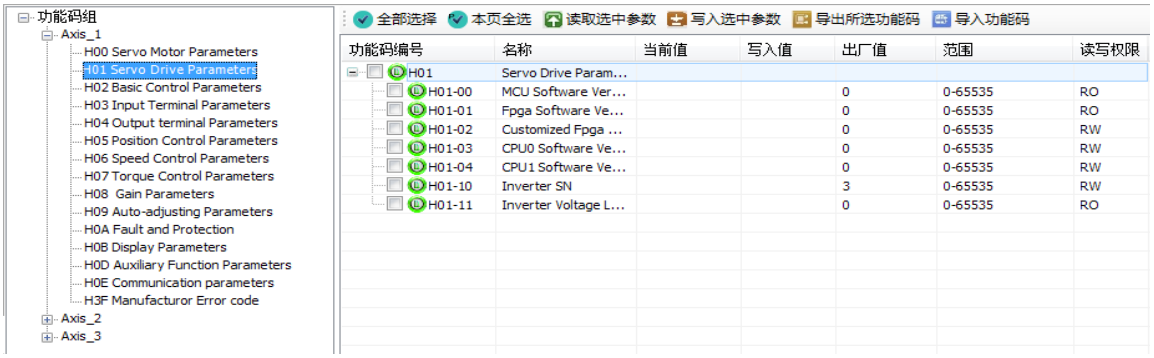
NOTE

IP 端口必须与虚拟以太网适配器处于同一网段。例如，如果以太网适配器的地址是 192.168.1.1，子网掩码是 255.255.255.0，则 IP 端口必须在 192.168.1.2 到 192.168.1.254 的范围之内。

- “DNS 服务器”:DNS 服务器的 IP 地址
- “DNS 名称”:DNS 服务器的名称

9 伺服功能码

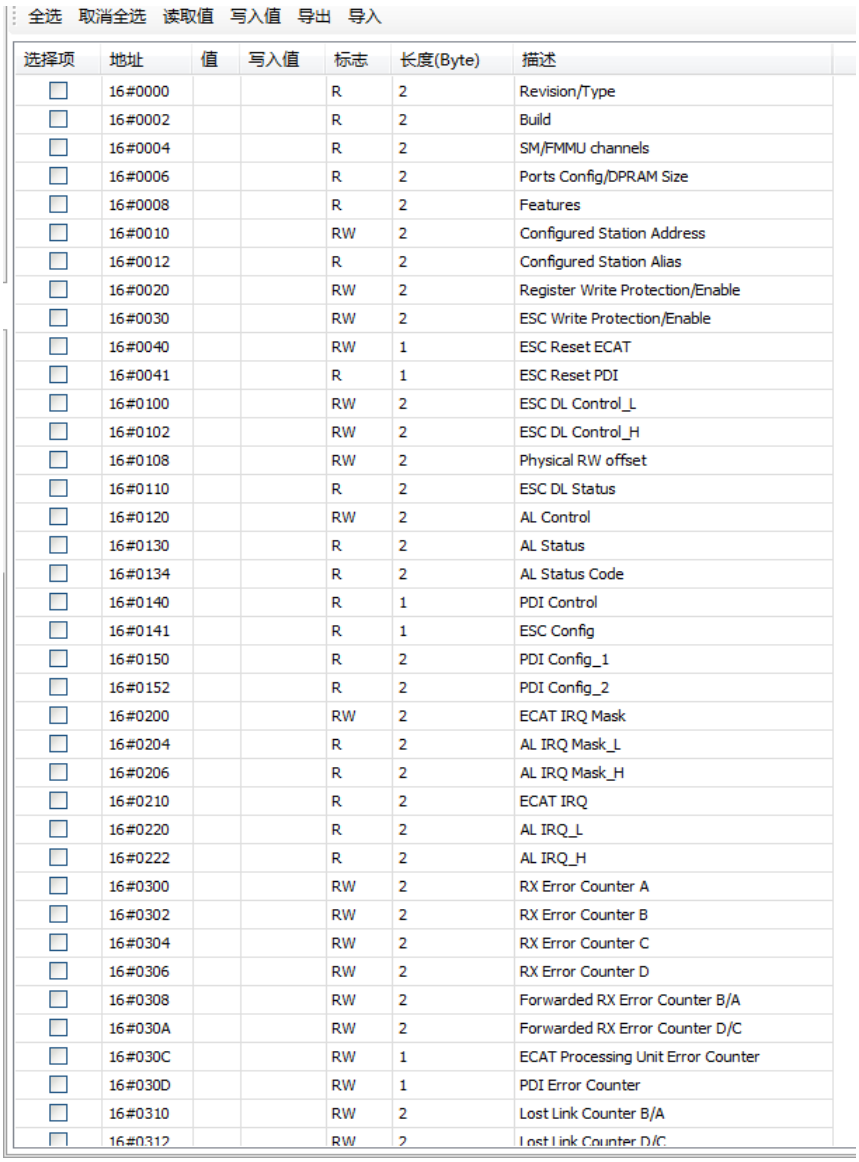
功能码指的是汇川（Inovance）的伺服类产品定义的厂家参数。在从站的“功能码”选项卡下，针对伺服从站读写、导入和导出厂家参数，便于调试和维护。如下图所示：



- “全部选择”：可以全部选择伺服功能码和“取消”全部选择项。
- “本页全选”：只针对当前页有效，全选和取消全选。
- “读取选中参数”：只有从站在运行状态时，且项目属性为读属性时，才可以读取。
- “写入选中参数”：只有从站在运行状态时，且项目属性为写属性时，才可以写入。
- “导出所选功能码”：导出选择的功能码。
- “导入功能码”：从外部导入功能码。

10 ESC 寄存器

“ESC 寄存器”在勾选“专家模式”后才会显示，用于在高级调试中读取 ESC 芯片寄存器地址的值，如下图所示:



选择项	地址	值	写入值	标志	长度(Byte)	描述
☐	16#0000			R	2	Revision/Type
☐	16#0002			R	2	Build
☐	16#0004			R	2	SM/FMMU channels
☐	16#0006			R	2	Ports Config/DPRAM Size
☐	16#0008			R	2	Features
☐	16#0010			RW	2	Configured Station Address
☐	16#0012			R	2	Configured Station Alias
☐	16#0020			RW	2	Register Write Protection/Enable
☐	16#0030			RW	2	ESC Write Protection/Enable
☐	16#0040			RW	1	ESC Reset ECAT
☐	16#0041			R	1	ESC Reset PDI
☐	16#0100			RW	2	ESC DL Control_L
☐	16#0102			RW	2	ESC DL Control_H
☐	16#0108			RW	2	Physical RW offset
☐	16#0110			R	2	ESC DL Status
☐	16#0120			RW	2	AL Control
☐	16#0130			R	2	AL Status
☐	16#0134			R	2	AL Status Code
☐	16#0140			R	1	PDI Control
☐	16#0141			R	1	ESC Config
☐	16#0150			R	2	PDI Config_1
☐	16#0152			R	2	PDI Config_2
☐	16#0200			RW	2	ECAT IRQ Mask
☐	16#0204			R	2	AL IRQ Mask_L
☐	16#0206			R	2	AL IRQ Mask_H
☐	16#0210			R	2	ECAT IRQ
☐	16#0220			R	2	AL IRQ_L
☐	16#0222			R	2	AL IRQ_H
☐	16#0300			RW	2	RX Error Counter A
☐	16#0302			RW	2	RX Error Counter B
☐	16#0304			RW	2	RX Error Counter C
☐	16#0306			RW	2	RX Error Counter D
☐	16#0308			RW	2	Forwarded RX Error Counter B/A
☐	16#030A			RW	2	Forwarded RX Error Counter D/C
☐	16#030C			RW	1	ECAT Processing Unit Error Counter
☐	16#030D			RW	1	PDI Error Counter
☐	16#0310			RW	2	Lost Link Counter B/A
☐	16#0312			RW	2	Lost Link Counter D/C

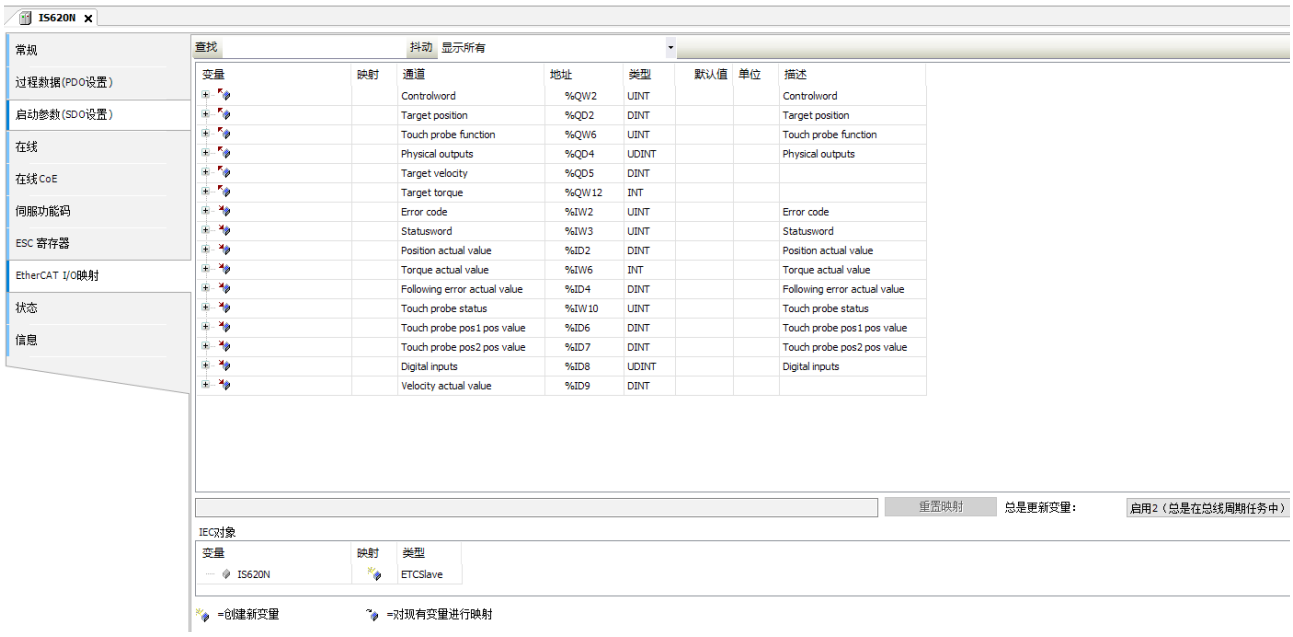
- “全选”：全部选择。
- “取消全选” 取消全部的选择项。
- “读取值”：在运行状态下，可以读取选择项的值。
- “写入值”：在运行状态下，可以写入具有写入属性的值。
- “导出”：以 XML 格式导出选择的项。
- “导入”：导入外部符合格式要求的 XML 文件，只显示导出的 XML 项。
- 右键：可以进行十六进制，十进制，二进制切换。

11 EtherCAT I/O 映射

EtherCAT 从站配置编辑器中的选项卡，其中为 EtherCAT I/O 指定了 ETCSlave 类型的实例（变量）和从站定义的 IO 变量，因此连接到 PLC 的 EtherCAT 从站可以由用户程序控制。

有关如何映射的描述，请参见“I/O 映射”。

自动创建的的从站实例显示在对话框的下部（IEC 对象）主站实例，可以用于应用中



NOTE

映射的变量和类型要一致。

12 状态

这个配置编辑器表格用于 EtherCAT 从站配置，提供关于网卡和内部总线系统的状态信息（例如‘启动’‘停止’）以及特定设备的诊断信息。

13 信息

此对话框由 EtherCAT 主站或从站配置对话框中提供。如果配置当前模块，将显示下列信息：名称、开发商、类型、版本号、分类、订单号、描述、图像等。

3.3.5 CiA402 轴

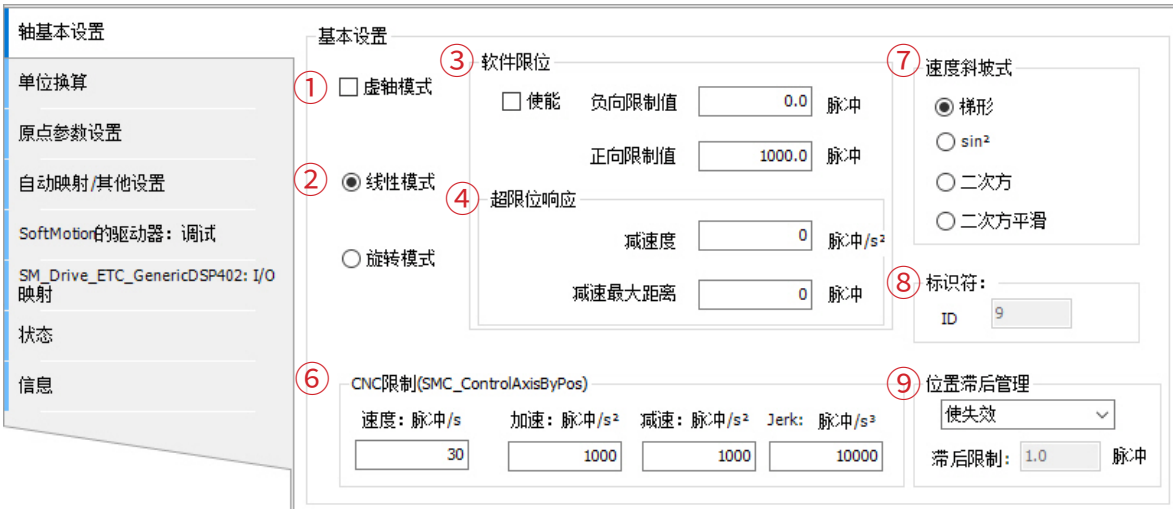
添加伺服从站后，双击“轴”会出现轴配置界面，下面按照由上至下的顺序依次对轴配置界面选项卡功能进行介绍。

1 轴基本设置

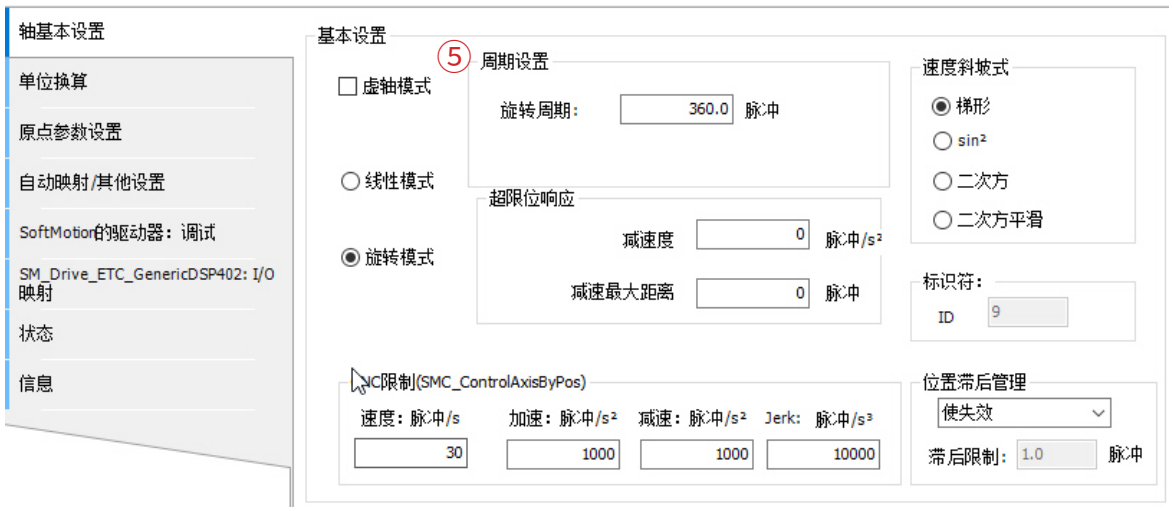
轴基本设置中包括虚轴 / 实轴模式两种工作模式，其定义如下表所述：

类别	功能描述
虚轴模式	虚轴模式，即可以不带物理伺服及电机运行的模式，可以模拟运行，得出自己需要的参数。不受外界环境的干扰。
实轴模式	实轴模式，必须带伺服电机运行，一些参数的获取必须在实轴模式才可以得到，如在线 COE，实轴模式会有外界的干扰，如在 TRACE 监控过程中，会影响到显示效果。

虚轴模式和实轴模式设置



线性模式和旋转模式设置



图中的主要选项功能定义如下表所述：

编号	选项名称	功能说明
1	虚轴模式	勾选，则为虚轴模式；不勾选，则为实轴模式。
2	轴类型	线性模式：轴位置以线性方式增加或者减少； 旋转模式：轴位置在固定范围增加或者减少。
3	软件限位	勾选使能后，对轴负向和正向的位置限制，用于线性模式下。

编号	选项名称	功能说明
4	软件限位出错响应	与软件限位关联，使能软件限位的状态下才具有实际意义：勾选使能后，若轴位置参数超出软件限位设置，软件会对出错做出的响应，即发生错误后，驱动器在减速最大距离内停止。
5	超限位响应	用于旋转模式下，对旋转周期进行限制。该参数与单位换算界面中电机旋转一圈的指令脉冲数、原点参数设置界面原点返回参数以及自动映射 / 其他设置界面最大速度相关联，设置时注意关联参数的设置，当关联参数值不匹配时，不匹配参数处会有警告提示该参数对应的匹配范围值。
6	CNC 限制	主要用于有 CNC 功能轴的限制设置。
7	速度斜坡式	主要用于轴的速度变化轨迹。
8	标识符	轴对于外部 ID。
9	位置滞后管理	位置滞后之后，管理轴的运行的方式。

下图为伺服启动后轴在线性模式下的界面，其中实时显示位置、速度、加速度、转矩以及状态通讯。如果有错误，则会显示错误信息。

NOTE

如果出现编码器位置信息丢失，请确认编码器电池是否接触可靠、电量充足。电池存储期间请按规定的环境温度进行存储。

2 单位换算

该界面主要功能是通过设置界面相关参数来计算脉冲数。

用户单位

① ◉ 脉冲 ○ 毫米 ○ 微米 ○ 纳米 ○ 度 ○ 英寸

行程距离

② ☐ 反向

③ 电机旋转一圈的指令脉冲数 16#100000 指令脉冲/转

○ 不使用变速装置

④ 工作台旋转一圈的工作行程 1 脉冲/转

参考：单位换算公式

$$\text{脉冲数(pulse)} = \frac{\text{电机旋转一圈的指令脉冲数[DINT]}}{\text{工作台旋转一圈的工作行程[LREAL]}} * \text{移动距离[用户单位]}$$

◉ 使用变速装置

⑤ 工作台旋转一圈的工作行程 1 脉冲/转

(如果轴类型是旋转模式，请参考轴基本设置界面的旋转周期值)

齿轮比分子（下图中（5）的齿数） 1

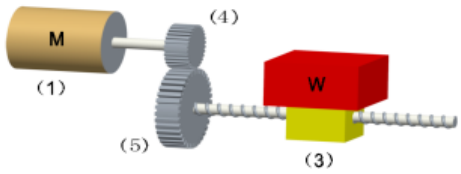
齿轮比分母（下图中（4）的齿数） 1

轴类型为线性模式

参考：单位换算公式

$$\text{脉冲数(pulse)} = \frac{\text{电机旋转一圈的指令脉冲数[DINT]}}{\text{工作台旋转一圈的工作行程[LREAL]}} * \frac{\text{齿轮比分母[DINT]}}{\text{齿轮比分子[DINT]}} * \text{移动距离[用户单位]}$$

M:电机，W:工作台

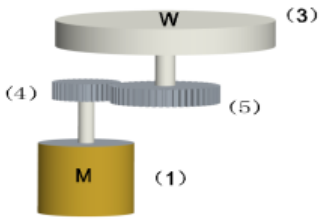


轴类型为旋转模式

参考：单位换算公式

$$\text{脉冲数(pulse)} = \frac{\text{电机旋转一圈的指令脉冲数[DINT]}}{\text{工作台旋转一圈的工作行程[LREAL]}} * \frac{\text{齿轮比分母[DINT]}}{\text{齿轮比分子[DINT]}} * \text{移动距离[用户单位]}$$

M:电机，W:工作台



上图中的主要选项及其功能如下表所述：

编号	选项名称	功能说明
1	用户单位	根据需要选择对应的长度单位，当单位选定后，轴基本设置、单位换算、原点参数设置以及自动映射 / 其他设置界面中应用的用户单位处均会作出相应改变。
2	反向	勾选后，轴以相反的方向旋转。
3	电机旋转一圈的指令脉冲数	电机的编码器分辨率，电机旋转一周需要的脉冲数量，默认值为 1048576（汇川 20 位编码器）。
4	不使用变速装置	勾选后，可对工作台旋转一圈的工作行程根据设备情况进行设置。工作台旋转一圈的工作行程：工作台（如皮带轮、机械齿轮、减速机等）旋转一圈机械末端所移动的距离（用户单位）。可根据参考的单位换算公式计算脉冲数。
5	使用变速装置	勾选后，可对工作台旋转一圈的工作行程、齿轮比分子以及齿轮比分母根据设备情况进行设置。工作台旋转一圈的工作行程：工作台旋转一圈机械末端所移动的距离（用户单位）；齿轮比分子：工作台齿轮的齿数；齿轮比分母：电机齿轮的齿数。可根据轴基本设置界面的勾选轴类型以及对应的参考单位换算公式计算脉冲数。注：齿轮比分子和分母可进行等比例的缩放。

应用示例

1） 电机直接驱动丝杆移动，电机旋转一圈丝杆移动 10mm，电机选择汇川 IS620N 增量式电机（编码器分辨率为 20 位），配置如下面示：

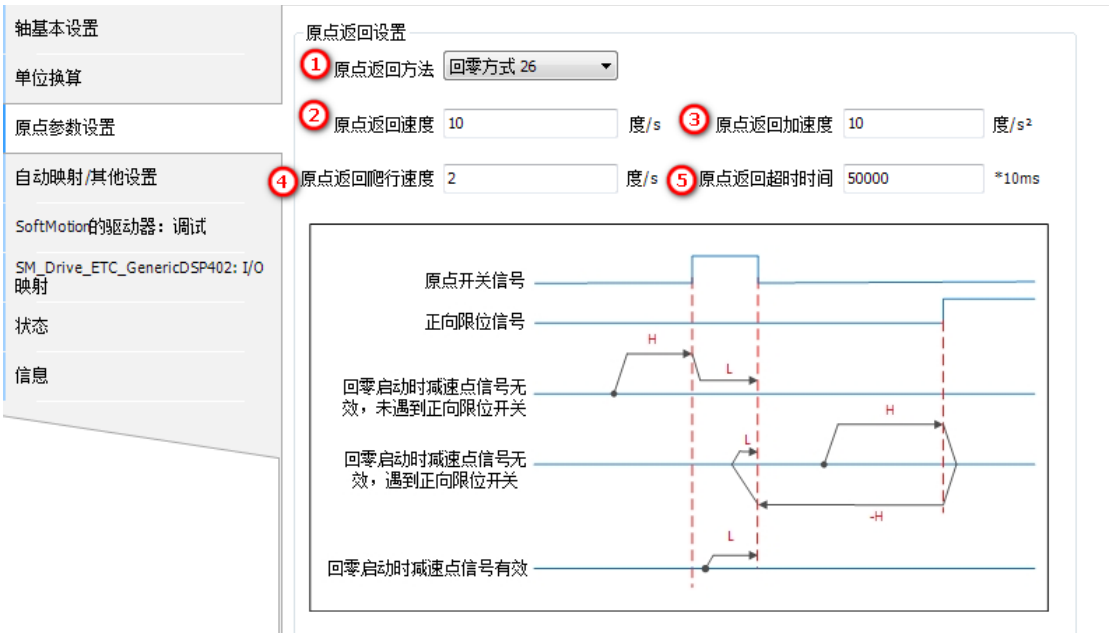
2） 电机间接驱动转盘，电机与转盘之间存在减速比为 30：1 变速装置（若电机齿轮齿数为 1，工作台齿轮齿数为 30，即工作台齿轮旋转 1 圈，电机齿轮旋转 30 圈），转盘的行程为 0-360 度，电机选择汇川 IS620N 绝对值电机（编码器分辨率为 23 位），配置如下所示：

NOTE

修改齿轮比分子、齿轮比分母及工作台旋转一圈的工作行程后，会影响到轴其他界面参数，请调整其他界面参数。

3 原点参数设置

原点参数设置，主要用于轴回零图形化参数配置（如下述界面）。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模式，方便用户更加直观、便捷地完成参数配置过程，如下图中①所示。

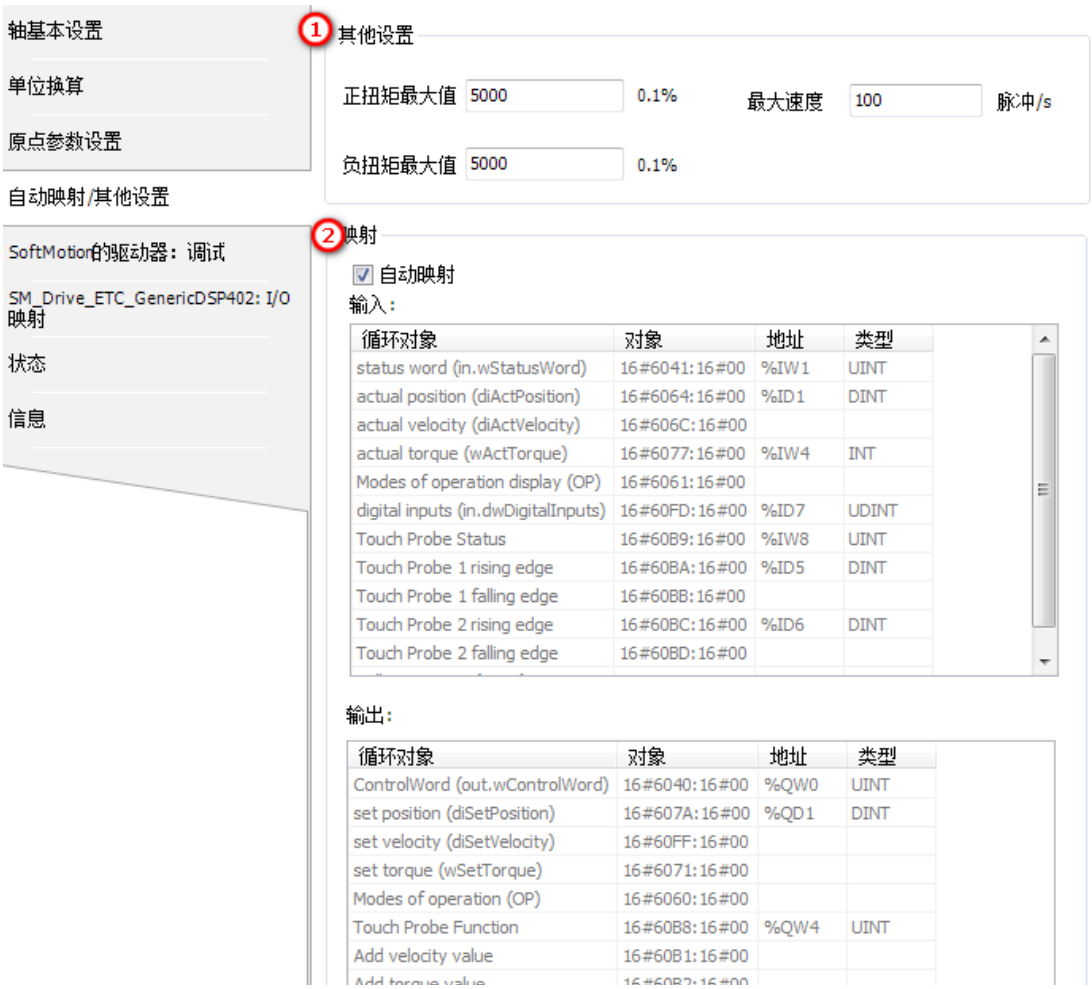


图中的主要选项及其功能如下表所述：

编号	选项名称	说明
1	原点返回方法	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，根据需要选择不同的回零方式。
2	原点返回速度	回零时，驱动轴搜索减速度点信号的高速速度值，如图中回零时的 H。

编号	选项名称	说明
3	原点返回加速度	回零时，驱动轴搜索速度变化的加速度。
4	原点返回爬行速度	回零时，驱动轴搜索原点时的低速速度值，如图中回零时的 L。
5	原点返回超时时间	限定原点回零总时间，超时则发生警告，驱动轴执行回零流程最大允许时间，如果回零超时，驱动轴回零失败。

4 自动映射 / 其他设置



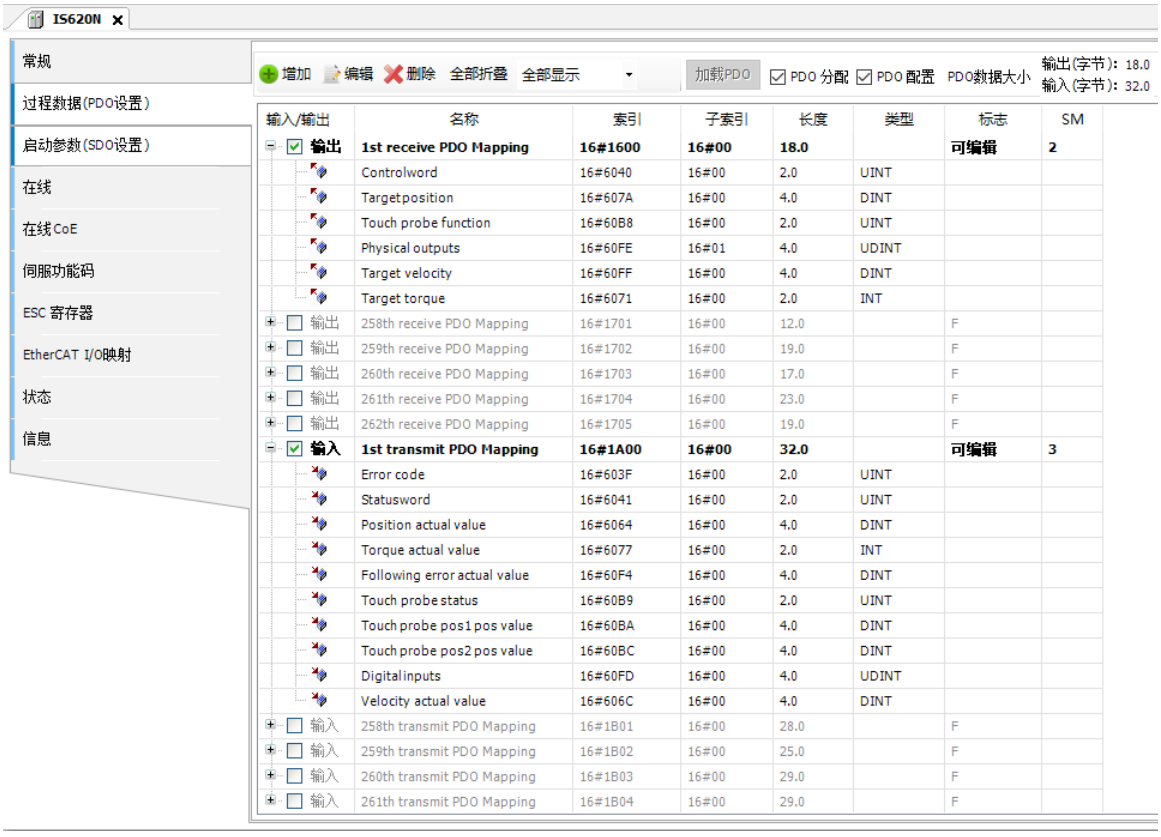
图中的主要选项及其功能如下表所述：

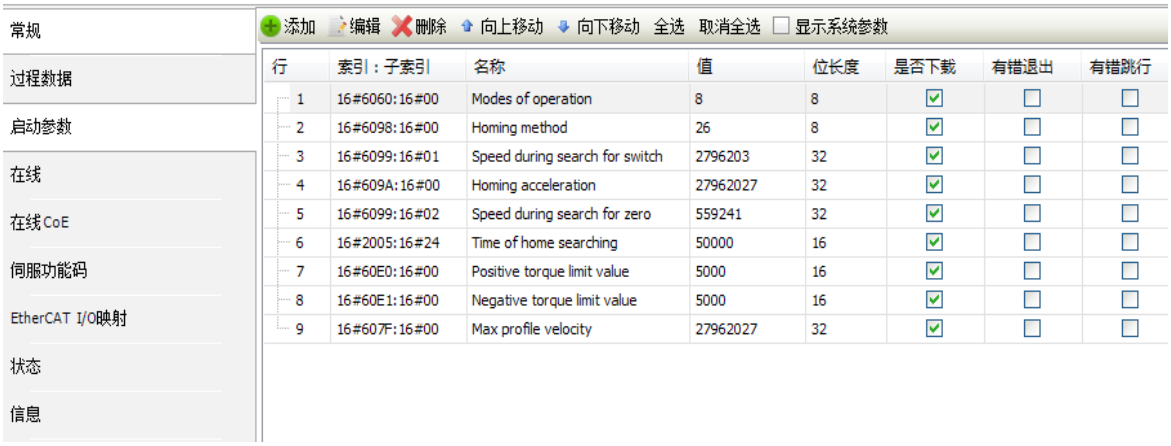
编号	选项名称	说明
1	其他设置	可对正 / 负扭矩最大值以及最大速度进行设置。 正 / 负扭矩最大值：为保护驱动器而设定的转矩指令限制值，当驱动器转矩指令大于转矩指令限制值，则实际驱动器的转矩指令被限幅等于转矩指令限制值。 最大速度：防止给定转矩指令过大，大于机械负载转矩，导致电机持续加速，可发生超速现象，损坏机械设备；设定速度限制后，实际转速将限制在速度限制值以内。 注：最大速度一般不可以设置为 0，如果为 0，轴运行可能出错。
2	映射	当自动映射勾选时，从站与轴之间存在关联，从站数据直接映射给轴；不勾选时，可以手动对轴映射数据中的地址进行修改。其中： ◆ 输入格式为 %I+ 类型对应的字母 + 阿拉伯数字； ◆ 输出格式为 %Q+ 类型对应的字母 + 阿拉伯数字。 类型对应的字母（类型所占用字节）SINT-B、UINT-W、DINT-D、UDINT-D。 当手动输入地址导致编译报错时，需要删除输入地址，重新输入正确地址。 注：当输入地址时加单引号，编译报错但显示结果正常，此时应删除显示结果，重新输入正确格式方可解决编译报错问题。

5 信息

该界面显示轴的基本信息，包括名称、供应商、组、类型、ID、版本、模块号、说明。

若轴参数没有改动的话，PDO 和 SDO 界面参数保持不变。若轴参数有改动，则 PDO 和 SDO 界面参数也对应自动变化。

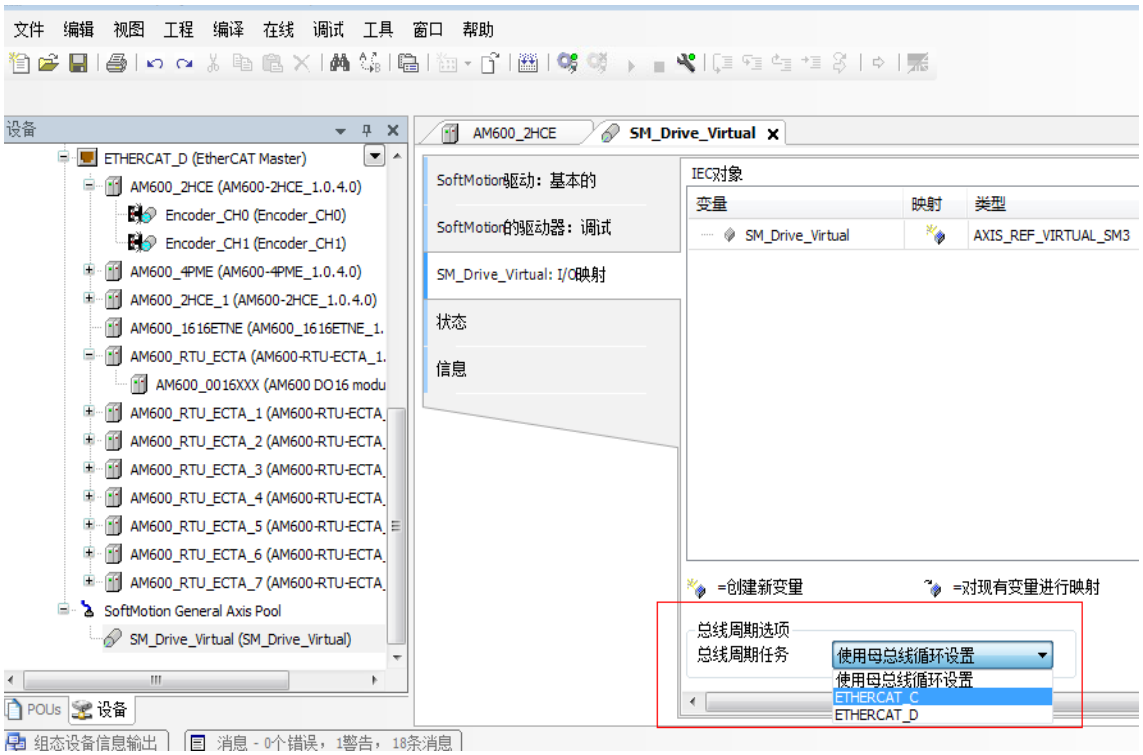




3.3.6 虚轴

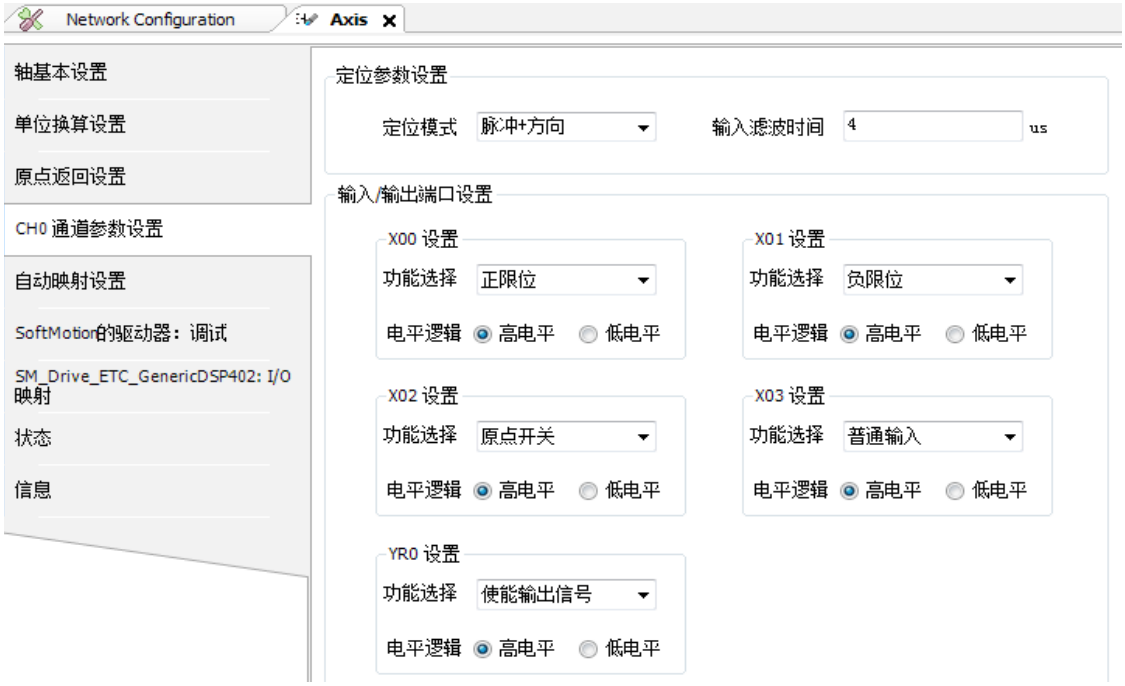
虚轴的界面与 EtherCAT 配置界面内容大致相同，需要注意的是，在同时启用多个 EtherCAT 主站并且使用虚轴时（从【轴池】-【虚轴】添加），虚轴的“总线周期任务”必须绑定到被调用的 EtherCAT 任务下（默认“使用母总线循环设置”，只适用单个主站情况），不允许同一个虚轴在多个 EtherCAT 任务下调用，否则会出现运行异常。

下图红色框内是虚轴的“总线周期任务”配置。



3.3.7 GR10-4PME 定位模块

定位模块界面除了增加如下界面，其他界面与 CiA402 轴设置界面相同。



GR10-4PME 模块是一款具有 4 个高速输出通道的脉冲定位模块，可实现脉冲型伺服、步进等以脉冲为给定信号的驱动器的速度和位置控制。该界面用于设置 GR10-4PME 模块的一些配置参数，以 GR10-4PME 模块第一通道为例，在该界面可配置如下功能：

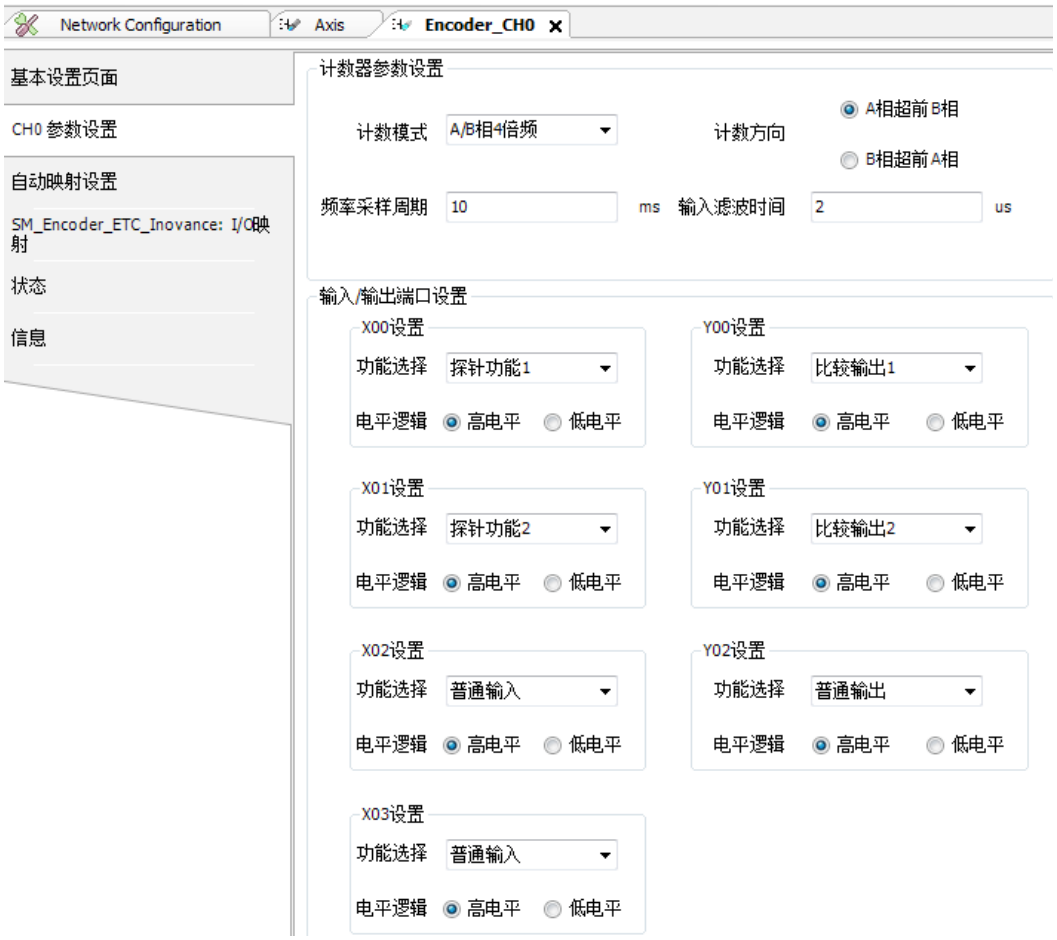
名称		描述	默认设置
定位模式		选择设置高速脉冲输出口输出的脉冲类型 AB 相 1 倍频 脉冲 + 方向 CW/CCW	脉冲 + 方向
输入滤波时间		数字量输入端子的脉冲输入滤波时间	4us
X00 设置	功能选择	X00 数字量输入端子功能选择 普通输入 急停开关 正限位	正限位
	电平逻辑	X00 数字量输入有效电平逻辑选择 高电平 - 输入高电平为有效电平 低电平 - 输入低电平为有效电平	高电平
X01 设置 电平逻辑	功能选择	X01 数字量输入端子功能选择 普通输入 急停开关 负限位	负限位
	X01 数字量输入有效电平逻辑选择 高电平 - 输入高电平为有效电平 低电平 - 输入低电平为有效电平	高电平	
X02 设置 电平逻辑	功能选择	X02 数字量输入端子功能选择 普通输入 急停开关 原点开关	原点开关
	X02 数字量输入有效电平逻辑选择 高电平 - 输入高电平为有效电平 低电平 - 输入低电平为有效电平	高电平	

名称		描述	默认设置
X03 设置 电平逻辑	功能选择	X03 数字量输入端子功能选择 普通输入 急停开关	普通输入
	X03 数字量输入有效电平逻辑选择 高电平 - 输入高电平为有效电平 低电平 - 输入低电平为有效电平	高电平	
YR0 设置 电平逻辑	功能选择	数字量输出端子 YR0 功能设置 普通数字量输出端子 伺服使能（输出信号）	伺服使能 （输出信号）
	数字量输出端子 YR0 导通关断设置 高电平 - 控制给定为 1 时导通 低电平 - 控制给定为 0 时导通	高电平	

有关定位模块的具体应用，请参见《EtherCAT 远程通信应用手册》。

3.3.8 GR10-2HCE 计数模块

计数模块除了增加如下界面，其他界面与 CiA402 轴设置界面相同。



GR10-2HCE 模块是一款具有 2 个高速输入通道的脉冲计数模块，可实现 AB 相脉冲、脉冲 + 方向、CW/CCW 形式的脉冲计数与测频。该界面用于设置 GR10-2HCE 模块的一些配置参数，以 GR10-2HCE 模块第一通道为例，该界面可配置如下功能：

名称		描述	默认设置
计数模式		通道输入脉冲输入方式选择 AB 相 1 倍频 AB 相 2 倍频 AB 相 4 倍频 脉冲 + 方向 CW/CCW	AB 相 4 倍频
频率采样周期		输入滤波频率计算采样周期	10ms
输入滤波时间		脉冲输入通道和数字量输入通道采样滤波	2us
计数方向	A 相超前	AB 相 当 A 相超前 B 相时计数器增加 脉冲 + 方向 B 相输入高电平时计数器增加 CW/CCW A 相有计数则计数器增加	A 相超前
	B 相超前	AB 相 当 B 相超前 A 相时计数器增加 脉冲 + 方向 B 相输入低电平时计数器增加 CW/CCW B 相有计数则计数器增加	
X00 设置	功能选择	X00 数字量输入端子功能选择 普通输入 探针 1 计数器清 0 计数器预置 门控	探针 1
	电平逻辑	X00 数字量输入有效电平逻辑选择 高电平 - 输入高电平为有效电平 低电平 - 输入低电平为有效电平	高电平
X01 设置 电平逻辑	功能选择	X01 数字量输入端子功能选择 普通输入 探针 2 计数器清 0 计数器预置 门控	探针 2
	X01 数字量输入有效电平逻辑选择 高电平 - 输入高电平为有效电平 低电平 - 输入低电平为有效电平	高电平	
X02 设置 电平逻辑	功能选择	X02 数字量输入端子功能选择 普通输入 计数器清 0 计数器预置 门控	普通输入
	X02 数字量输入有效电平逻辑选择 高电平 - 输入高电平为有效电平 低电平 - 输入低电平为有效电平	高电平	

名称		描述	默认设置
X03 设置 电平逻辑	功能选择	X03 数字量输入端子功能选择 普通输入 计数器清 0 计数器预置 门控	普通输入
	X03 数字量输入有效电平逻辑选择 高电平 - 输入高电平为有效电平 低电平 - 输入低电平为有效电平	高电平	
Y00 设置 电平逻辑	功能选择	数字量输出端子 Y00 功能设置 普通输出 比较输出 1	比较输出 1
	数字量输出端子 Y00 导通关断设置 高电平 - 控制给定为 1 时导通 低电平 - 控制给定为 0 时导通	高电平	
Y01 设置 电平逻辑	功能选择	数字量输出端子 Y01 功能设置 普通输出 比较输出 2	比较输出 2
	数字量输出端子 Y01 导通关断设置 高电平 - 控制给定为 1 时导通 低电平 - 控制给定为 0 时导通	高电平	
Y02 设置 电平逻辑	功能选择	数字量输出端子 Y02 功能设置 普通输出	普通输出
	数字量输出端子 Y02 导通关断设置 高电平 - 控制给定为 1 时导通 低电平 - 控制给定为 0 时导通	高电平	

有关定位模块的具体应用，请参见《EtherCAT 远程通信应用手册》。

3.3.9 分支器

1 分支器简介

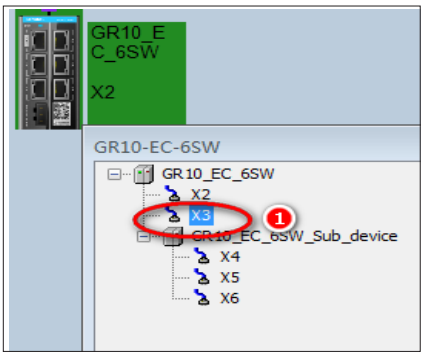
分支器模块用于扩展 EtherCAT 端口，如下图所示：

接口	选项名称	功能说明
IN1	分支器入口	分支器入口连接设备 EtherCAT 出口。
X2~X6	分支器出口	X2~X6 都为分支器出口，各个出口之间互不影响，每路出口可以连接一路 EtherCAT 从站设备。

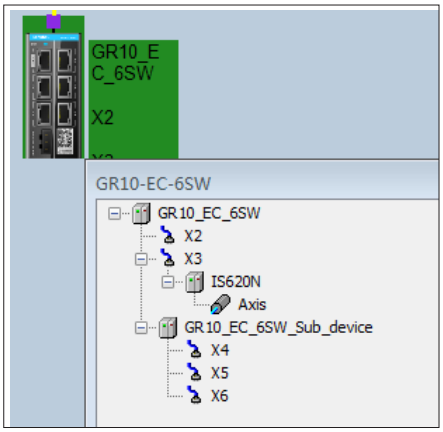
2 添加分支器及其下从站设备

分支器设备的添加，删除，复制，粘贴和普通从站设备一样（见 3.1.1.1 及 3.1.1.3 章节），添加完分支器设备后，添加分支器下从站需按如下步骤操作：

1） 双击组态分支器设备，弹出如下图所示框：



2） 先选择与实际物理组态一致的合适的节点，再选择网络设备列表设备，即可完成如下图所示分支器从站组态：



也可以使用扫描设备命令（请参见章节 3.3.2.2 扫描设备），扫描分支器及其下从站设备，复制到工程中，自动完成设备的组态。

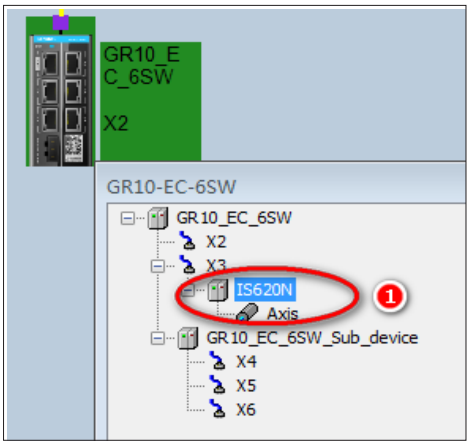
3 删除分支器及其下从站设备

删除分支器设备和从站设备删除相同（请参见章节 3.1.1.3）。

删除分支器下从站设备的操作如下：

1：打开分支器配置界面。

2：选择需要删除的从站，如下图编号 1 所示：



按“删除”（Delete）按键，即可删除从站设备。

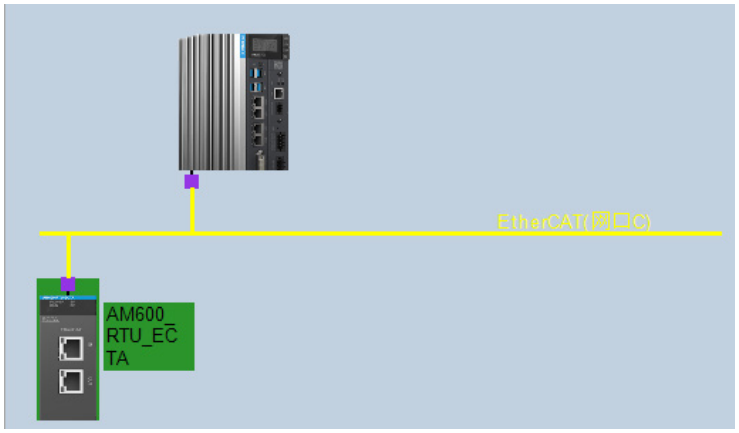
NOTE

删除后，不支持撤销和恢复操作，需要重新增加设备组态。

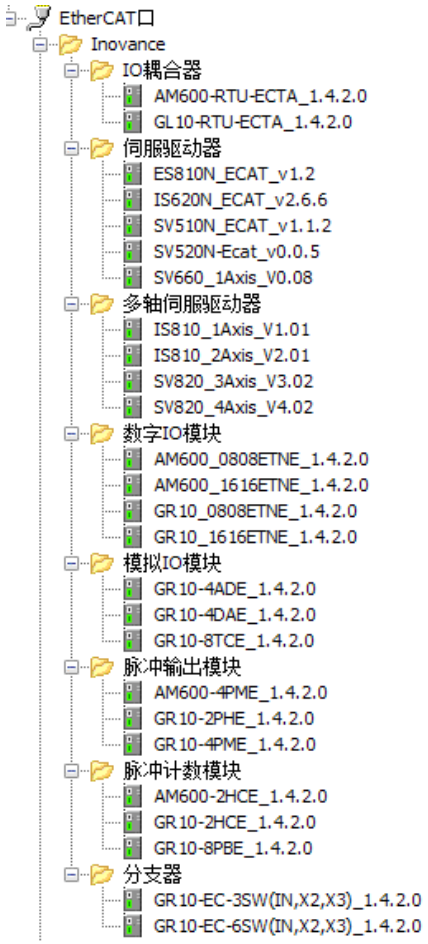
3.3.10 I/O 模块

添加 Inovance AM600-RTU-ECTA 模块的步骤如下：

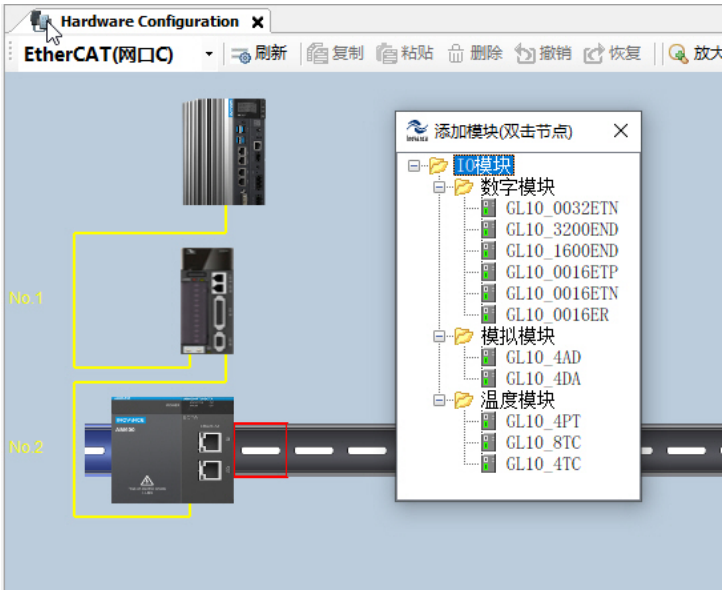
1） 鼠标双击【Network Configuration】，在弹出的图形组态界面，鼠标单击选中相应的 EtherCAT 总线；



2） 在【网络设备列表】内，鼠标双击添加 AM600-RTU-ECTA 从站（注：不推荐使用 AM600_EtherCAT_Slave）；



3） 双击 AM600-RTU-ECTA 从站，在如下的红框中，添加所需的 IO 模块；



4） 有关各个模块的详细功能，请参见《EtherCAT 远程通信应用手册》。

3.3.10 库（隐含变量）

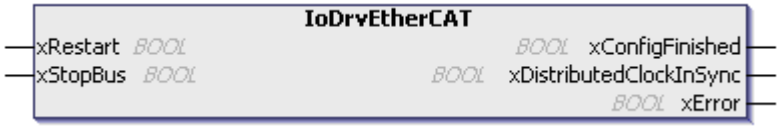
1 主站的隐式实例

EtherCAT 主站只要被插入到设备列表中，将建立一个‘IoDrvEtherCAT’类型的隐式实例。实例名称与设备列表中使用的设备名称完全相同。

NOTE

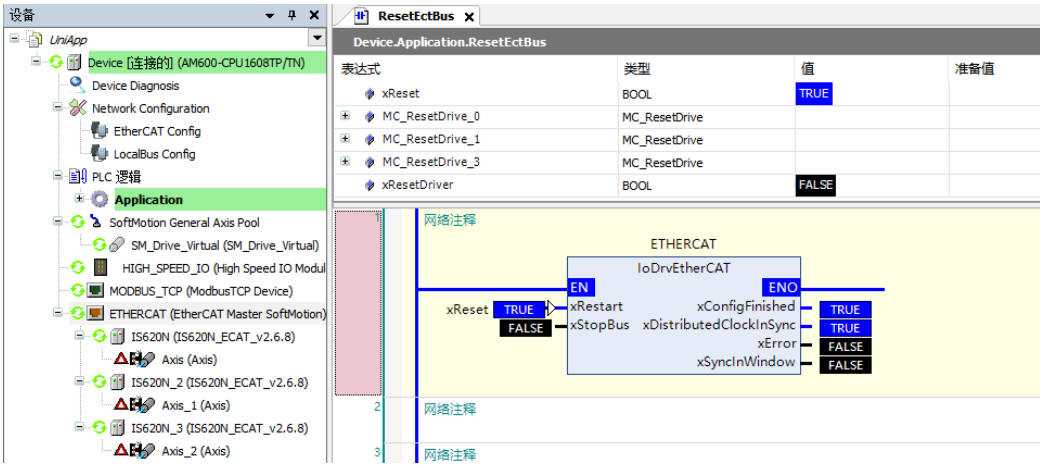
隐式实例是由系统自动生成，不可在程序中定义，否则会导致 PLC 运行异常。

IoDrvEtherCAT 类型的隐式实例定义如下表所述：

IoDrvEtherCAT 隐式实例	
	
输入参数	定义
xRestart	总线重启：上升沿时，当前主站会重启，所有配置参数会重新加载。
xStopBus	总线停止：电平触发，当输入值为“TRUE”时，EtherCAT 总线通信将会停止，通讯进入错误状态，停止后若要继续使用，必须通过“xRestart”重启 EtherCAT 通信。
输出参数	定义
xConfigFinished	如果这个参数为“TRUE”，所有配置参数的传送已经正确完成。通讯正在运行。
xDistributedClockInSync	如果 EtherCAT 从站配置了 DC(分布时钟)，总线启动时首先会配置 EtherCAT 从站参数，待参数配置完毕（“xConfigFinished”为“TRUE”），开始调整从站时钟以及主站的时钟。当主站和从站时钟同步成功后，输出“TRUE”；正常运行后总线出现故障丢失同步，则输出“FALSE”。DC 模式下，必须等到这个参数为“TRUE”再启动运动控制功能块，否则，运动轴的位置可能会产生跳跃。
xError	如果 EtherCAT 主站启动时检测到错误，或者从站通讯状态机到达操作模式后通讯被中断，则该输出为 TRUE，因为 EtherCAT 主站再收不到任何消息（比如由于连线中断）。此时，主站诊断或者日志信息定位错误原因。

示例

示例程序如下：该功能块的名称为 IoDrvEtherCAT，主站名 ETHERCAT（AM600 默认）即为其实例化名称，不需要再在程序里面实例化声明。AC800 系列主站名默认为 ETHERCAT_C 或者 ETHERCAT_D。



如图 xRestart 输入变量由 FALSE 变化 TRUE 上升沿触发主站重启，xStopBus 为 True 电平触发总线通讯停止，通过输出参数状态判断总线通讯信息。

主站属性

属性	定义
AutoSetOperational	如果此属性被设置为 TRUE，一旦通讯中止，主站总是会尝试重启从站。 默认值： FALSE
ConfigRead	若此属性返回 TRUE，表明已完成配置的读取，用户可以编辑设置了。如为了实现自定义 SDOs 的添加。
DCInSyncWindow	XDistributedClockInSync 置 TURE 的时间窗口条件。主站的同步抖动必须在此窗口范围内，XDistributedClockInSync 的值才为 TRUE。 默认值：50 微秒
DCIntegralDivider	分布时钟控制回路的积分因子。 默认值： 20
DCPropFactor	分布时钟控制回路的比例因。 默认值： 25
DCSyncToMaster	分布时钟与主站的同步性。如果设为 TRUE，所有从站都会与主站同步，而不是第一个从站与 PLC 同步。 默认值：FALSE
DCSyncToMasterWithSysTime	与主站的分布时钟同步。如果设置为 TRUE 所有从站都将会与主站系统时间同步。通过 SysTimeRtcHighResGet 读取的时间也可以用于 PLC 到所有 EtherCAT 从站直接的同步。 默认值 : FALSE
EnableTaskOutputMessage	EtherCAT 消息通常都是由总线周期任务发送，额外也会由每个使用从站输出的任务发送。在总线周期任务中，会写入所有的输出，并读取所有的输入。在其他任务中，输出会多传输一次，以使他们立刻被写入各自的从站。因此要尽可能的缩短截止时间直到写的足够快。在有分布时钟协同时，这可能导致某些从站出现问题，例如伺服控制器与 同步中断不同步，却对内部同步器使用写的时间点。此情况下，一个循环内可能出现多个写访问。如果 EnableTaskOutputMessage 设为 FALSE，则只会使用总线周期任务，额外任务不会再影响消息。 默认值： TRUE
FirstSlave	主站下第一个从站的指针。
FrameAtTaskStart	如果 FrameAtTaskStart 被设为 TRUE，则给从站的帧内容将在任务开始被发送，以保证最小的抖动。此命令用来得到伺服驱动器的平滑运动。如果此标志位被设为 TRUE，则输出缓冲帧被写入下一个循环中。 默认值： FALSE
LastInstance	关联主站列表的指针 -> 上一个主站。
LastMessage	此属性与 EtherCAT 栈最新消息一起返回一个字符串，如果成功完成启动，则应该返回‘完成所有从站’。字符串的作用，同在线模式下 EtherCAT 主站设备编辑器中显示的诊断信息一样。
NextInstance	关联主站列表的指针 -> 下一个主站。
NumberActiveSlaves	此属性返回实际连接从站的数量。如果 StartConfigWithLessDevice 设为 TRUE, 则可以确定实际设备的数量。
OpenTimeout	打开网管超时时间，默认为 4 秒。
StartConfigWithLessDevice	此属性可以用来影响堆栈启动行为。例如组态中配置了五个伺服控制器，但只连接了三个的情况，则堆栈会马上停止，总线配置失败，PLC 报错。但是如果在第一个周期中 StartConfigWithLessDevice 设为 TRUE，则堆栈会尝试启动，总线正常配置运行，PLC 不报错。因此，如果建立了一个 10 个伺服控制器的通用配置，但连接控制器的实际数量可变，并会逐个检查每一个从站的供应商 ID 和产品 ID，如果发现一个不匹配，堆栈启动都会停止。

另外，只要设备支持，由库 IIoDrvBusControl.library 提供的接口可以用来实现从应用外部对 EtherCAT 装置进行访问。

2 从站的隐式实例

对于每个 EtherCAT 从站，只要设备被插入到设备列表中，则会建立一个”ETCSlave”类型的隐式实例。实例名称与设备列表中使用的设备名称完全相同。

使用实例化对象的输入输出参数实现特殊的用法，如在应用运行时，使用从站实例获得、切换和检查从站状态。

ETCSlave 类型的隐式实例定义如下表所述：

ETCSlave 隐式实例	
	
输入参数	定义
xSetOperational	在上升沿时，尝试将从站通讯状态机设置为 ETC_SLAVE_ OPERATIONAL 模式。
输出参数	定义
wState	返回从站的当前状态，可能取值为： 0: ETC_SLAVE_BOOT 1: ETC_SLAVE_INIT 2: ETC_SLAVE_PREOPERATIONAL 4: ETC_SLAVE_SAVEOPERATIONAL 8: ETC_SLAVE_OPERATIONAL

NOTE

状态 ETC_SLAVE_OPERATIONAL 表示配置已成功完成。如果配置时发生错误，有可能设备会回到之前的状态。以下以 IS620N 从站为示例，实现显示从站状态功能。

ETCSlave 示例

以 620N 为例添加 620N 实例名称，定义：nSlaveState: ETC_SLAVE_STATE;

编程	说明
IS620N(xSetOperational:= , wState=> nSlaveState);	从站 IS620N 隐式实例调用，将从站状态输出到 nSlaveState 变量中。

从站属性

属性	定义
VendorID	在 EtherCAT 主站启动后，此属性返回设备中读取的供应商 ID。
ConfigVendorID	此属性从配置中读取供应商 ID。
ProductID	在 EtherCAT 主站启动后，该属性返回从站中读取的产品 ID。
ConfigProductID	此属性从配置中读取产品 ID。
SerialID	在 EtherCAT 堆栈启动后，该属性包含了设备序列号。
LastEmergency	如果收到消息，则将此信息存储于从站内部。基于这一属性可在应用中读取信息。另外还添加了一个日志消息。

另外，只要设备支持，由 IIoDrvBusControl.library 库提供的接口可以实现从应用外访问 EtherCAT 设备。如果在高级设置中激活了供应商和产品 ID，只要检测到供应商 ID 与配置供应商 ID 或者产品 ID 与配置产品 ID 不匹配，则主站的启动将被停止。

3 检查所有从站的链表

每个 EtherCAT 主站和 EtherCAT 从站都会隐式创建一个功能块实例，它可以用来监视各个从站的状态。为此，

此实例必须在应用程序中调用，通过 wState 读取从站状态。为了简化编程，所有主站和从站都可以由链表找到，因此所有从站都可以由一个简单的‘WHILE’循环检查。

主站和从站分别有对应的属性 NextInstance 和 LastInstance，可以返回指向下一个和最后一个从站的指针。另外，主站的 FirstSlave 属性有效，其提供了一个指向第一个从站的指针。

链表功能示例

检查所有从站状态，定义：pSlave: POINTER TO ETCSlave;

编程	说明

pSlave := EtherCAT_Master.FirstSlave;
WHILE pSlave <> 0 DO
pSlave^();
//TODO：用户添加代码，例如可以做从站 Op 状态 计数
pSlave := pSlave^.NextInstance;
END_WHILE
	首先通过 EtherCAT_Master.FirstSlave 找到主站的第一个从站。
	在‘WHILE’循环中调用各个实例，由此确定 wState，然后检查状态。
	通过 pSlave^.NextInstance 找到指向下一个从站的指针。
	在列表结尾出指针为空，循环结束。

4 用于 CoE 的 IODrvEtherCAT 函数库功能块

CoE 功能块：CANOPEN over EtherCAT 功能块

EtherCAT 库 IODrvEtherCAT.library 使能 EtherCAT 配置后会被自动添加入工程中，其中包含了读写参数的功能块，因此在在线模式下，能够检查与修改特殊参数。在使用 CANOPEN over EtherCAT 功能块时可以同时调用多个功能模块，内部请求会以队列方式处理。

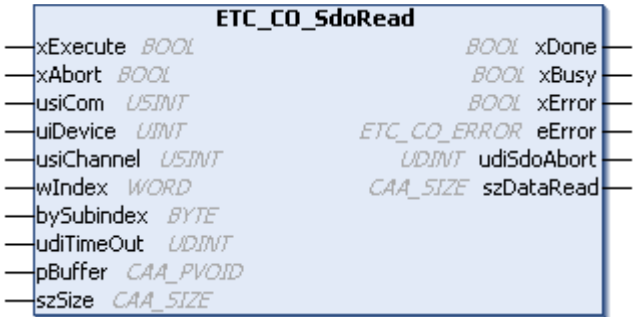
CANOPEN over EtherCAT 功能块包含以下功能块：

- “ETC_CO_SdoRead”（取出参数，长度可能超过 4 字节）
- “ETC_CO_SdoRead4”（读取参数，长度不大于 4 字节）
- “ETC_CO_SdoReadDword”（读取参数，并将值存到一个 DWORD 中）
- “ETC_CO_SdoRead_Access”（读取所有记录）
- “ETC_CO_SdoRead_Channel”（读取从站参数）
- “ETC_CO_SdoWrite”（写参数，长度可能超过 4 字节）
- “ETC_CO_SdoWrite4”（写参数，长度不大于 4 字节）
- “ETC_CO_SdoWriteDWord”（直接在 DWORD 中写参数值）E
- “ETC_CO_SdoWriteAccess”（写从站参数）

ETC_CO_SdoRead

此功能模块由 IODrvEtherCAT.library 库文件提供，用于读取 EtherCAT 从站参数。与 ETC_CO_SdoRead4 不同，该模块支持大于 4 字节的参数。读取的参数由对象字典索引和子索引指定。

ETC_CO_SdoRead 功能块



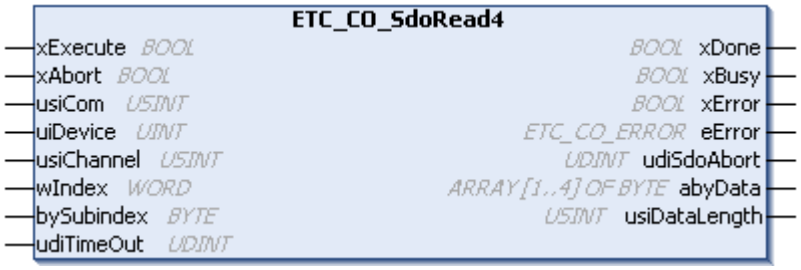
ETC_CO_SdoRead 功能块	
输入参数	定义
xExecute	在这个输入的上升沿，启动读取从站参数。为了获得内部通道存储单元分配，这个实例必须通过‘xExecute:= FALSE’调用至少一次。
xAbort	如果此参数为‘TRUE’，当前读取过程将被终止。
usiCom	EtherCAT 主站个数：如果仅使用一个 EtherCAT 主站，usiCom 为‘1’。使用多个主站时，第一个主站为‘1’，第二个为‘2’，依次类推。
uiDevice	从站的物理地址。 如果主站的 自动配置模式被取消，从站可以设置特有地址。这个任意选择的地址必须输入到这里。 如果激活自动配置模式，第一个从站将得到地址‘1001’. 当前从站地址总能在 设备编辑器的从站配置对话框中的 ‘EtherCAT 地址’ 栏进行查核。
usiChannel	保留用于扩展。
wIndex	对象字典中的参数索引。
bySubIndex	对象字典中的参数子索引。
udiTimeout	在这里你可以设定以毫秒为单位的超时时间。如果在这时间内 , 读取参数没有被执行，将提示一个错误信息。
pBuffer	数据缓冲区的指针，数据缓冲区即参数成功传递后数据的存储区域。
szSize	数据缓冲区（见上：pBuffer）的大小，以字节表示。
输出参数	定义
xDone	只要读取参数已经成功结束，此值输出为‘TRUE’。
xBusy	只要读取参数尚未结束时，此值输出为‘TRUE’。
xError	如果出错，此值输出为‘TRUE’。eError 参数将显示错误原因。
eError	此输出（类型 ETC_CO_ERROR）显示 xError 所指示的出错原因。例如‘ETC_CO_TIMEOUT’表示超时错误。
udiSdoAbort	当设备检查出错时，此输出将提供更多有关错误的信息。
szDataRead	读取字节的数目；最大的 szSize（参见输入参数）。

ENUM ETC_CO_ERROR

错误	代码	说明
ETC_CO_NO_ERROR	0	没有错误
ETC_CO_FIRST_ERROR	5750	错误原因被存储于 udiSdoAbort 的输出
ETC_CO_OTHER_ERROR	5751	没找到主站
ETC_CO_DATA_OVERFLOW	5752	ETC_CO_Expedited 并且大小 > 4
ETC_CO_TIME_OUT	5753	超过时限
ETC_CO_FIRST_MF	5770	没有使用
ETC_CO_LAST_ERROR	5799	没有使用

ETC_CO_SdoRead4

此功能块由库 IODrvEtherCAT.library 提供，用于读取 EtherCAT 从站参数。不同于 ETC_CO_SdoRead 功能块只能读取不大于 4 字节的参数。读取的参数特别由对象字典里的索引和子索引指定。

ETC_CO_SdoRead4 功能块


输入参数	定义
xExecute	在这个输入的上升沿，启动读取从站参数。为了获得内部通道的存储单元分配，这个实例必须通过‘xExecute:= FALSE’调用至少一次。
xAbort	如果此输入为‘TRUE’，当前读取过程将被终止。
usiCom	EtherCAT 主站个数: 如果仅使用一个 EtherCAT 主站，usiCom 为‘1’。使用多个主站时，第一个主站为‘1’，第二个为‘2’，依次类推。
uiDevice	从站的物理地址。 如果主站的 自动配置模式被取消，从站可以设置特有地址。这个任意选择的地址必须输入到这里。 如果激活自动配置方式，第一个从站将得到地址”1001”. 当前从站地址总能在 设备编辑器的从站配置对话框中的‘EtherCAT 地址’栏检查。
usiChannel	为将来扩展保留的，当前未使用。
wIndex	对象字典中的参数索引。
bySubIndex	对象字典中的参数子索引。
udiTimeout	在这里可以设定以毫秒为单位的超时时间。如果在此时间内，读取参数没有被执行，将生成一个错误信息。
输出参数	定义
wState	返回从站的当前状态，可能取值为： 0: ETC_SLAVE_BOOT 1: ETC_SLAVE_INIT 2: ETC_SLAVE_PREOPERATIONAL 4: ETC_SLAVE_SAVEOPERATIONAL 8: ETC_SLAVE_OPERATIONAL
xDone	只要读取参数已经成功结束，此输出为‘TRUE’。
xBusy	只要读取参数尚未结束，此输出为‘TRUE’。
xError	如果出错，此值输出为‘TRUE’。eError 参数将显示错误原因。
eError	此输出（类型 ETC_CO_ERROR）表示 xError 所指示的出错原因。例如‘ETC_CO_TIMEOUT’表示超时错误。
abyData	读到的参数数据会被复制到这个 4 字节的数组中。 如果第一个字节已经被读取 , 将保存在数组的第一个索引中。如果是 2 个或者 4 个字节数据，按照 Intel 字节顺序拷贝到这个数组中。
usiDataLength	读取字节的个数 (1,2,4)。

ENUM ETC_CO_ERROR

错误	代码	说明
ETC_CO_NO_ERROR	0	没有错误
ETC_CO_FIRST_ERROR	5750	错误原因被存储于 udiSdoAbort 的输出
ETC_CO_OTHER_ERROR	5751	没找到主站
ETC_CO_DATA_OVERFLOW	5752	ETC_CO_Expedited 且大小 > 4

错误	代码	说明
ETC_CO_TIME_OUT	5753	超过时限
ETC_CO_FIRST_MF	5770	没有使用
ETC_CO_LAST_ERROR	5799	没有使用

ETC_CO_SdoReadDword

此功能块由库 IODrvEtherCAT.library 提供，如功能块 ETC_CO_SdoRead4 一样用来读取 EtherCAT 从站参数。不过读取的数据不拷贝到数组中而是到一个 DWORD 中（dwData）。如果需要字节交换，则可自动完成。因此读到的数据可直接被后面的过程所用。

ETC_CO_SdoRead_Access

此功能块由库 IODrvEtherCAT.library 提供，如功能块 ETC_CO_SdoRead 一样用来读取 EtherCAT 从站参数。不过通过附加输入 xCompleteAccess (BOOL)，允许读入所有记录的完整索引。为此，xCompleteAccess 必须要设为 ‘TRUE’ 且 bySubIndex 必须为 ‘0’ 。

ETC_CO_SdoRead_Channel

EtherCAT 配置编辑器 > EtherCAT 编程接口 > 用于 “CAN over EtherCAT” 的 IODrvEtherCAT 函数库功能块 > ETC_CO_SdoRead_Channel

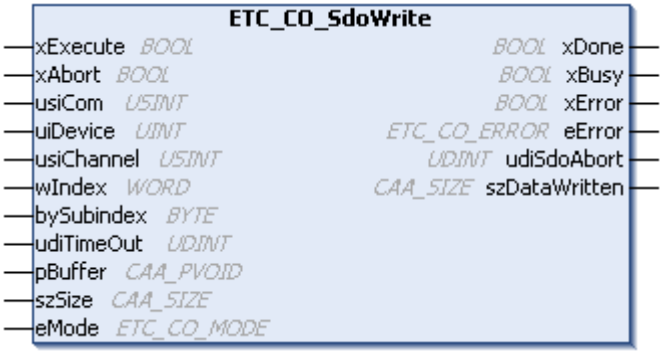
此功能块由库 IODrvEtherCAT.library 提供，如功能块 ETC_CO_SdoRead_Access 一样用来读取所有 EtherCAT 从站参数。

但是它有一个附加输入 byChannelPriority (BYTE) 用来指定 CoE 邮箱消息中的通道和优先权。此字节的前 6 位指定通道，后 2 位指定优先级。

ETC_CO_SdoWrite

此功能块由库 IODrvEtherCAT.library 提供，用于写 EtherCAT 从站参数。不同于 ETC_CO_SdoWrite4 功能块可以用于读取大于 4 字节的参数。写入的参数特别由对象字典里的索引和子索引指定。

ETC_CO_SdoWrite 功能块



输入参数	定义
xExecute	在这个输入的上升沿，启动写入从站参数。为了获得内部通道的存储单元分配，这个实例必须通过”xExecute:= FALSE”调用至少一次。
xAbort	如果此输入参数为“TRUE”，当前写入过程将被终止。
usiCom	EtherCAT 主站个数：如果仅使用一个 EtherCAT 主站，usiCom 为 ‘1’ 。使用多个主站时，第一个主站为 ‘1’ ，第二个为 ‘2’ ，依次类推。
uiDevice	从站的物理地址。 如果取消主站 自动配置模式，从站可以设置特有地址。这个任意选择的地址必须输入到这里。 如果激活自动配置方式，第一个从站将得到地址 ‘1001’ . 当前从站地址总能在 从站配置对话框中的 ‘EtherCAT 地址’ 栏检查。
usiChannel	为将来扩展保留的，当前未使用。
wIndex	对象字典中的参数索引。

ETC_CO_SdoWrite 功能块	
bySubIndex	对象字典中的参数子索引。
udiTimeout	在这里可以设定以毫秒为单位的超时时间。如果在此时间内 , 写入参数没有被执行，将生成一个错误信息。
pBuffer	数据缓冲区的指针，数据缓冲区即参数成功传递后数据的存储区域。
szSize	数据缓冲区（见上：pBuffer）的大小，以字节表示。
eMode	这个输入（枚举 ETC_CO_MODE）定义了写入字节的个数: 可能的值包括: ETC_CO_AUTO（自动）,ETC_CO_EXPEDITED（加速）或 ETC_CO_SEGMENTED（分段）。通常使用 ETC_CO_AUTO 模式，因为在这个模式下，自动匹配数据长度。
输出参数	定义
xDone	只要写入参数已经成功结束，此输出为“TRUE”。
xBusy	只要写入参数尚未结束，此输出为“TRUE”。
xError	如果出错，此输出为“TRUE”。eError 参数将显示错误原因。
eError	此输出（类型 ETC_CO_ERROR）表示 xError 所指示的出错原因。例如”ETC_CO_TIMEOUT”表示超时错误。
udiSdoAbort	当设备出错时，此输出将提供更多的有关错误的信息。
szDataWritten	写入的字节个数；成功写入后，将设置到 szSize。

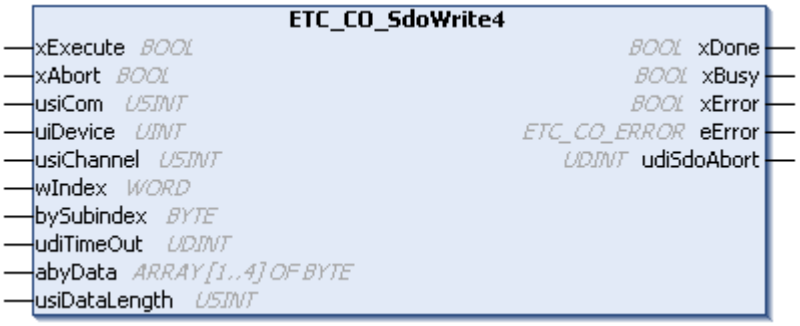
ENUM ETC_CO_MODE

模式	编号	说明
AUTO	0	客户选择自动模式
EXPEDITED	1	客户使用加速协议
SEGMENTED	2	客户使用分段传输协议

ETC_CO_SdoWrite4

此功能块由库 IODrvEtherCAT.library 提供，用于写 EtherCAT 从站参数。不同于 ETC_CO_SdoWrite 功能块此功能块只支持写小于 4 字节的参数。写入的参数特别由对象字典里的索引和子索引指定。

ETC_CO_SdoWrite4 功能块



输入参数	定义
xExecute	在这个输入的上升沿，启动写入从站参数。为了获得内部通道的存储单元分配，这个实例必须通过‘xExecute:= FALSE’调用至少一次。
xAbort	如果此输入参数为‘TRUE’，当前写入过程将被终止。
usiCom	EtherCAT 主站个数：如果仅使用一个 EtherCAT 主站，usiCom 为‘1’。使用多个主站时，第一个主站为‘1’，第二个为‘2’，依次类推。

ETC_CO_SdoWrite4 功能块	
uiDevice	从站的物理地址。 如果取消主站的 自动配置模式，从站可以设置特有地址。这个任意选择的地址必须输入到这里。 如果激活自动配置方式，第一个从站将得到地址 ‘1001’. 当前从站地址总能在 从站配置对话框中的 “EtherCAT 地址”检查。
usiChannel	为将来扩展保留的，当前未使用。
wIndex	对象字典中的参数索引。
bySubIndex	对象字典中的参数子索引。
udiTimeout	在这里可以设定以毫秒为单位的超时时间。如果在此时间内 , 写入参数没有被执行，将生成一个错误信息。
abyData	这个数组包含了 4 个写入的数据。数据必须按照 Intel 字节顺序存储。
usiDataLength	写入字节的个数（1，2，4）
输出参数	定义
xDone	一旦写参数完成这个输出将会被置位 TRUE。
xBusy	如果写没有完成此输出将会为 TRUE 。
xError	如果一个错误产生此输出将会被置为 TRUE， eError 将显示错误原因。
eError	这个输出 (类型 ETC_CO_ERROR) 显示当前错误的原因，标识为 xError 。例如 “ETC_CO_TIMEOUT” 表示超时。
udiSdoAbort	如果设备中发生一个错误，这个输出将提供更多的错误信息。

ENUM ETC_CO_MODE

模式	编号	说明
AUTO	0	客户选择自动模式
EXPEDITED	1	客户使用加速协议
SEGMENTED	2	客户使用分段传输协议

ETC_CO_SdoWriteDWord

此功能块由库 IODrvEtherCAT.library 提供，如功能块 ETC_CO_SdoWrite4 一样用来写 EtherCAT 从站数据。但是要写入的数据不以数组而以一个 DWORD（dwData）的形式给出。若需要字节交换，则可以自动完成。因此要被写入的值可直接被指定。

ETC_CO_SdoWriteAccess

此功能块由库 IODrvEtherCAT.library 提供，如功能块 ETC_CO_SdoWrite 一样用来写 EtherCAT 从站参数。

但是通过附加的输入 xCompleteAccess (BOOL)，所有记录的完整索引都可被写入。为此，xCompleteAccess 必须要设为 ‘TRUE’ 且 bySubIndex 必须为 ‘0’ 。另一个输入 byChannelPriority (BYTE) 可被用来指定 CoE 邮箱消息中的各个数据元素（通道和优先级）。

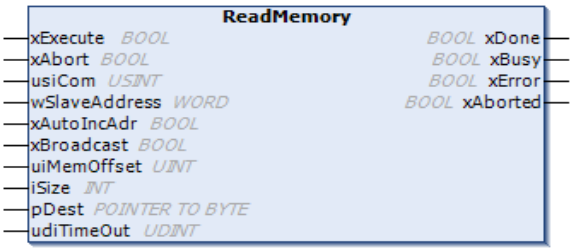
5 直接访问 EtherCAT 从站内存

EtherCAT 配置编辑器 > EtherCAT 编程接口 > 直接访问 EtherCAT 从站内存

直接访问 EtherCAT 从站内存：ReadMemory 与 WriteMemory 。

■ ReadMemory

这个功能块位于函数库 IODrvEtherCAT.library 用于读取 EtherCAT 从站的内存数据。

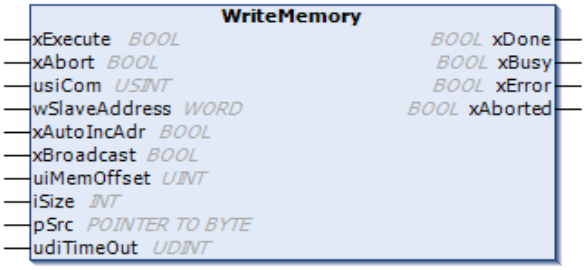
ReadMemory 功能块		
		
输入参数	类型	定义
xExecute	BOOL	上升沿：动作启动 下降沿：复位输出 如果在功能块完成动作之前有一个下降沿，输出操作会以通常方式完成并且在动作完成或者中断 (xAbort) 时会被复位。在这种情况下，相关的输出值 (xDone, xError, iError) 正好位于一个周期内输出。
xAbort	BOOL	如果这个输入为 TRUE，动作会立即停止并且所有输出被复位为初始值。
usiCom	USINT	主站索引 1：第一个 EtherCAT 主站
wSlaveAddress	WORD	自动创建的地址或者设备的物理地址
xAutoIncAdr	BOOL	确定要使用地址方式的标志
xBroadcast	BOOL	如果要使用板卡的标志，如果为 true 那么 wSlaveAddress 和 bAutoIncAdr 不会被使用
uiMemOffset	UINT	内存地址偏移
iSize	INT	读取字节
pDest	POINTER TO BYTE	存取数据缓冲区
udiTimeOut	UDINT	操作超时时间（ms）
输出参数	类型	定义
xDone	BOOL	动作成功完成
xBusy	BOOL	功能块激活
xError	BOOL	TRUE：产生错误，功能块中止动作 FALSE：没有错误
xAborted	BOOL	用于中止动作

示例：寄存器 0x130 读取 (当前状态)

```
PROGRAM PLC_PRG
VAR
    etcreadmemory : ReadMemory;
    wStatus : WORD;
    xRead : BOOL;
END_VAR

etcreadmemory(xExecute := xRead, usiCom:=1, wSlaveAddress := 1002,
            xAutoIncAdr := FALSE, xBroadcast := FALSE, uiMemOffset := 16#130,
            iSize := 2, pDest := ADR(wStatus), udiTimeout := 500);
```

■ WriteMemory

这个功能块位于函数库 IODrvEtherCAT.library 用于往 EtherCAT 从站内存中写数据。

WriteMemory 功能块		
		
输入参数	类型	定义
xExecute	BOOL	上升沿：动作启动 下降沿：复位输出 如果在功能块完成动作之前有一个下降沿，输出操作会以通常方式完成并且在动作完成或者中断 (xAbort) 时会被复位。在这种情况下，相关的输出值 (xDone, xError, iError) 正好位于一个周期内输出。
xAbort	BOOL	如果这个输入为 TRUE，动作会立即停止并且所有输出被复位为初始值。
usiCom	USINT	主站索引 1：第一个 EtherCAT 主站
wSlaveAddress	WORD	自动创建的地址或者设备的物理地址
xAutoIncAdr	BOOL	确定要使用地址方式的标志
xBroadcast	BOOL	如果要使用板卡的标志，如果为 true 那么 wSlaveAddress 和 bAutoIncAdr 不会被使用
uiMemOffset	UINT	内存地址偏移
iSize	INT	写入字节
pDest	POINTER TO BYTE	存取写入数据缓冲区
udiTimeOut	UDINT	操作超时时间（ms）
输出参数	类型	定义
xDone	BOOL	动作成功完成
xBusy	BOOL	功能块激活
xError	BOOL	TRUE：产生错误，功能块中止动作 FALSE：没有错误
xAborted	BOOL	用于中止动作

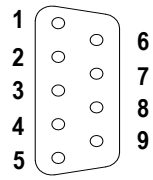
示例：写入寄存器 0x120 (控制寄存器)

```
PLC_PRG
VAR
    etcwritememory : WriteMemory;
    xWrite : BOOL;
    wCommand : WORD := 4; // set to safe operational
END_VAR
etcwritememory(xExecute := xWrite, usiCom:=1, wSlaveAddress := 1002,
            xAutoIncAdr := FALSE, xBroadcast := FALSE, uiMemOffset := 16#120,
            iSize := 2, pSrc := ADR(wCommand), udiTimeout := 500);
```

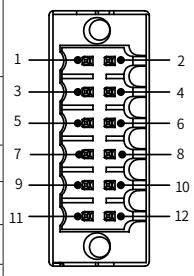
3.4 Modbus 设备编辑器

3.4.1 串行硬件端口

AM600 支持两路 485 串口通信，分别是串口 0 和串口 1，均支持自由协议；

端口	通道	引脚	定义
	COM0（RS485）	1	RS485-
		2	RS485+
		5	GND
	COM1（RS485）	6	RS485-
		9	RS485+
		3	GND

AC800 系列 PLC 支持一路 485 通信和一路 232 通信，通过 I/O 通信接口接线，其中，485 通信引脚为 8/9/11，232 通信引脚为 8/10/12，具体引脚定义如下表所示。

描述	功能	信号名	编号	IO/ 通信接口	编号	信号名	功能	描述
输入高电平宽度为 500ms 的脉冲时，模拟外部开机按键按下一次，启动 PLC	开机信号（配合UPS使用）		1		2	P_STATUS	上电点灯信号	控制器上电启动后输出
ON-OFF 变化时启动掉电保存功能	掉电检测信号	P_OK	3		4	P_STATUS	运行状态信号	控制器上电启动后输出
OFF 时 RUN；ON 时 STOP	RUN/STOP	RUN	5		6	0V	输出公共端	--
--	输入公共端	0V	7		8	GND	通信参考地	--
--	RS485+	485+	9		10	232R	RS232 接收	--
--	RS485-	485-	11		12	232T	RS232 发送	--

AC800 系列，在后台软件中 COM 端口 0 对应 RS232 通信接口，COM 端口 1 对应 RS485 通信接口，在配置使用时须区分。

3.4.2 网络组态

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“网络设备列表”中单击“MODBUS”将从站添加到网络中。

图 3-29 Modbus 组态示例

此时，在界面左侧视图中将出现 Modbus 组态对应设备树。如下：

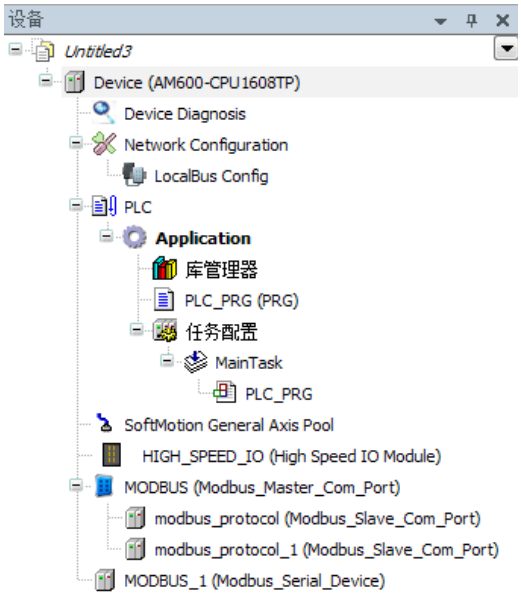


图 3-30 Modbus 组态配置对应设备树示例

AM600 支持两路 Modbus 串口通信，分别是串口 0 和串口 1，均支持标准的 ModbusRTU 协议，可独立配置为主站或从站，支持 4800, 9600, 19200, 38400, 57600, 115200 这 6 种波特率。

PLC 做 Modbus 从站设备时，可以被主站设备访问的地址范围定义如下：

1） 所有的位变量操作 (功能码 0x01、0x02、0x05、0x0F) 可以读写 %QX0.0-%QX8191.7 共 65535 个位变量；

2） 所有的寄存器变量操作 (功能码 0x03、0x04、0x06、0x10) 可以读写 MW0-MW65535 共 65536 个寄存器变量；

3） 汇川 HMI 可以访问 AM600 的系统变量 SM0-SM7999，寄存器变量 SD0-SD7999。

3.4.3 Modbus 主站配置

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，需要注意，Modbus 主从站通信参数一致时，才能正常通信。

图 3-31 Modbus 主站配置

主站配置参数

配置项	功能
COM 端口	该主站物理连接选择串口 0 或串口 1
波特率	通信时的速率
奇偶校验	通信帧的校验方式
数据位	通信帧包含的实际数据位
停止位	通信时标识单个包的最后位
传输模式	RTU
帧间隔	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

示例：

配置项	配置值
COM 端口	0
波特率	115200
奇偶校验	偶校验
数据位	8
停止位	1
传输模式	RTU
帧间隔	2ms

3.4.4 Modbus 主站通信配置

PLC 做 Modbus 主站时，双击设备树中的从站设备打开 Modbus 从站设置窗口，如下图所示：

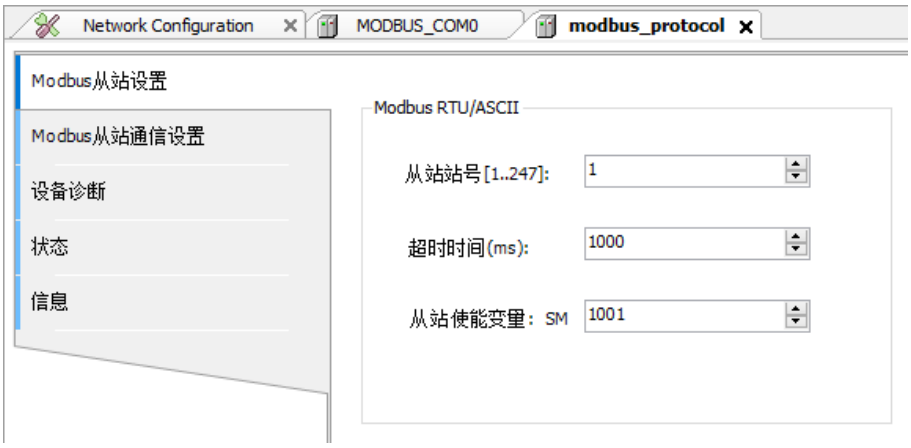


图 3-32 端口作为主站时的 Modbus 从站配置

从站配置参数：

配置项	功能
从站站号	标识从站号，范围 1~247
超时时间	主站发帧后，超过该时间从站未响应，主站报接收超时
从站使能变量	编程使能该变量后，主站开始向该从站发送通信帧

示例：

配置项	配置值
从站站号	11
超时时间	1000ms
从站使能变量	1001

切换页面到 Modbus 从站通信设置窗口，添加 Modbus 主从站通信配置，配置表最多支持 60 条配置，如下图所示：

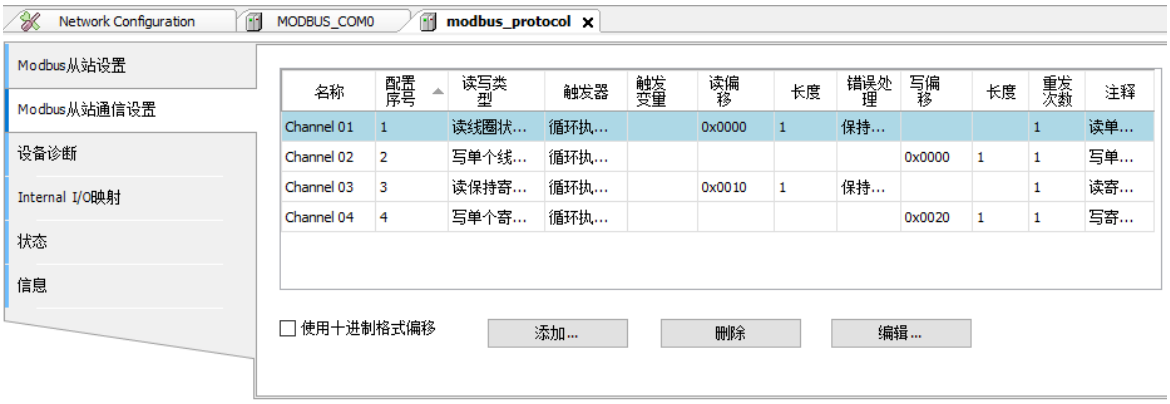


图 3-33 端口做主站时的 Modbus 从站通信设置

图 3-33 中，每个通道代表一个独立的 Modbus 请求，其中，第 4 行定义了一个循环执行写单个寄存器（功能码 0x06）的请求，向偏移量为 0x0020 的 1 个寄存器写一个字的数据。

点击“添加…”后会出现一个为 Modbus 从站添加新通道的对话框。点击“确定”按钮可新建一个通道

在“Modbus 从站通道”列表中选择一个通道，单击“编辑…”按钮将会出现一个对话框，通过修改其中的参数可改变通道的配置，点击“确定”按钮可更新通道设置。当添加或者编辑通道时，对话框中有以下参数可供使用：

图 3-34 端口做主站时连接 Modbus 从站通信设置对话框

Modbus 通信设置配置

配置项	功能
名称	通道命名的字符串

配置项	功能	
存取类型	读线圈状态（功能码 01） 读输入状态（功能码 02） 读保持寄存器（功能码 03） 读输入寄存器（功能码 04） 写单个线圈（功能码 05） 写单个寄存器（功能码 06） 写多个线圈（功能码 15） 写多个寄存器（功能码 16）	
触发器	循环执行：周期触发的请求	循环时间：设置时间再次执行
	电平触发：编程进行改变时触发	触发变量（SM）：设置触发 SM 元件，触发成功后，自动复位该元件
重发次数	本次发生通信故障未获得从站返回帧，则按重发次数进行重新发送。	
注释	可以对数据进行描述的简短文本区域	
读寄存器		
起始地址	读取的寄存器开始位置	
长度	读取的寄存器个数	
错误处理	保持最后的值：使数据保持最后一次的有效值 设置为 0：使所有值归零	
写寄存器		
起始地址	写寄存器开始位置	
长度	写寄存器长度	

“长度”参数的有效范围取决于以下功能码：

功能码	类型访问	寄存器数
01	读线圈状态	1~2000
02	读输入状态	1~2000
03	读保持寄存器	1~125
04	读输入寄存器	1~125
05	写单个线圈	1
06	写单个寄存器	1
15	写多个线圈	1~1968
16	写多个寄存器	1~123

Modbus 从站 Internal I/O 映射

图 3-35 端口做主站时的 Modbus 从站 Internal I/O 映射

在 Modbus 从站通信设置中添加主从站通信配置后，Internal I/O 映射中会自动分配每条配置的映射地址，如上图第一行的 %IW1 表示将读取的一个线圈数值映射到 %IW1 这个地址。另外，还可以通过输入助手或者直接输入示例变量路径，将程序中的自定义变量映射到 I/O 地址。

3.4.5 Modbus 主站广播配置

当 Modbus 主站连接多台 Modbus 从站，所有 Modbus 从站都接收写操作时，需要 Modbus 主站进行广播。

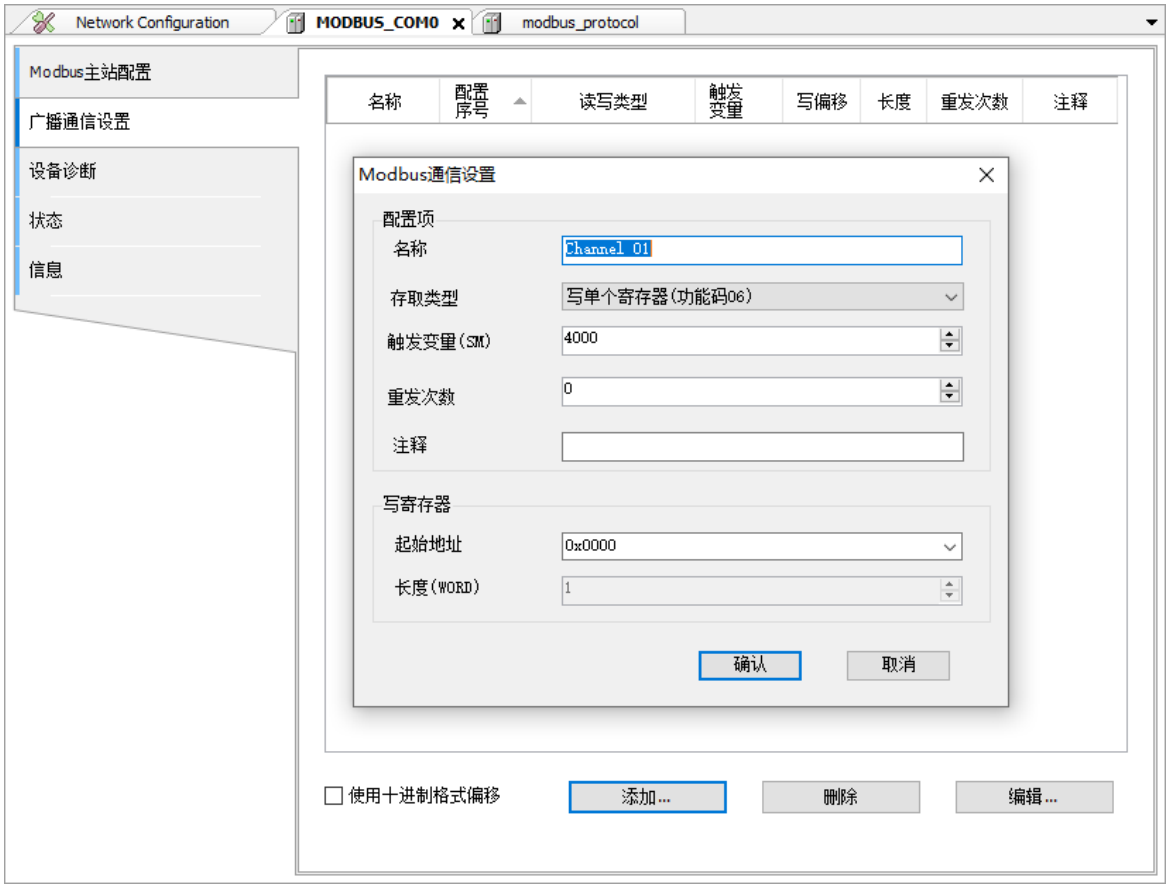


图 3-36 端口作为主站时广播配置

广播通信设置选项卡中通过点击“添加”按钮，会显示 “Modbus 通信设置”对话框，其中的相关配置项有：名称、存取类型、触发变量（SM）、重发次数、注释，写寄存器包括起始地址和长度。

存取类型包括多个功能码，有写单个线圈（功能码 05）、写单个寄存器（功能码 06）、写多个线圈（功能码 15）、写多个寄存器（功能码 16）。

- 触发变量：Modbus 主站进行通信的触发条件，只有触发变量为 true 时，Modbus 主站才进行广播通信，需要在编程时对触发变量进行置位。
- 重发次数：本次发生通信故障未获得从站返回帧，则按重发次数进行重新发送。

3.4.6 Modbus 从站配置

PLC 做 Modbus 从站时，双击设备树中的从站设备，打开 Modbus 从站配置窗口，如下图所示：

图 3-37 端口作为从站时的配置

Modbus 从站配置参数中，串口配置部分与 Modbus 主站串口配置含义相同，Modbus 从站配置站号是指本设备站号；帧间隔为收到主站发送的通信帧后，延时回复主站的具体时间段。

注意：Modbus 主从站通信参数配置一致时，才能正常通信。

3.4.7 Modbus 设备诊断

Modbus 主站设备诊断可显示发生故障的从站和故障的通信配置项。

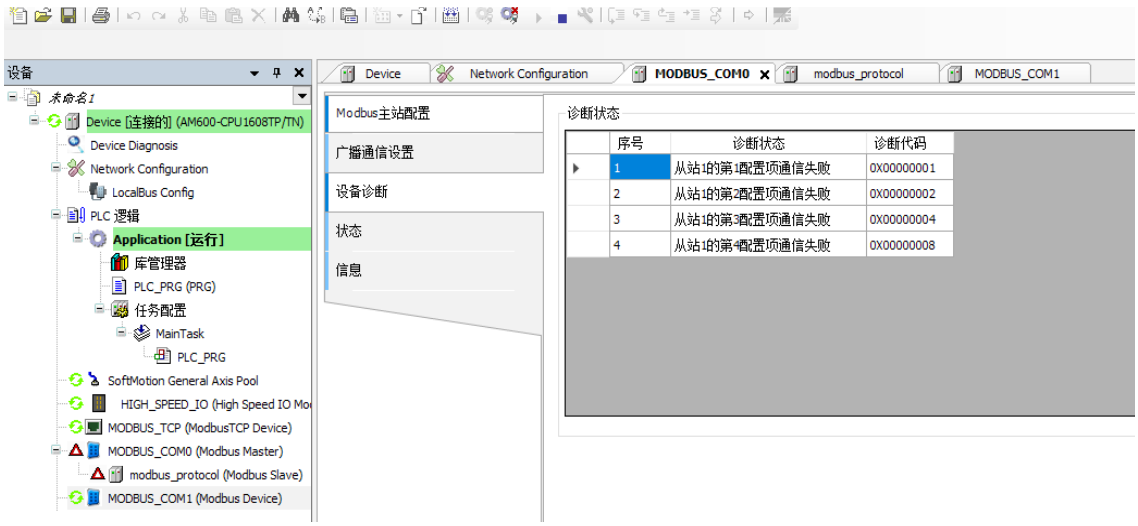


图 3-38 Modbus 主站设备诊断

3.4.8 Modbus 常见故障

Modbus 主站连接从站时发生的主要故障如下：

■ Modbus 主站与 Modbus 从站配置不一致，导致主站与从站通信无法建立。

- Modbus 主站访问 Modbus 从站非法地址，返回错误应答。
- Modbus 主站操作 Modbus 从站写寄存器，但是 Modbus 从站该寄存器只支持读不支持写操作，Modbus 主站会收到 Modbus 从站返回的出错应答。

错误响应帧格式及含义

错误响应： 从机地址 +（命令码 +0x80）+ 错误码 +CRC 校验。

本错误帧适合所有的操作命令帧。

序号	数据 (字节) 意义	字节数量	说明
1	从机地址	1 个字节	取值 1~247
2	命令码 +0x80	1 个字节	错误命令码
3	错误码	1 个字节	1~4

3.4.9 Modbus 变量编址

线圈：位变量，只有两种状态 0 和 1。本 PLC 中包含 Q 区及 SM 区等变量。

变量名称	命令码	起始地址	线圈数量	说明
QW0-QW511	0X01,0x05,0x0f	0	8192	通用标准 Modbus 协议都可以访问
SM0-SM7999	0x31,0x35,0x3f	0	8000	与汇川 HMI 的专用协议，使用不同的功能码

寄存器：16 位（字）变量，本 PLC 中包含 M 区及 SD 区等变量

变量名称	命令码	起始地址	寄存器数量	说明
MW0-MW65535	0x03,0x06,0x10	0	65536	通用标准 Modbus 协议都可以访问
SD0-SD7999	0x33,0x36,0x40	0	8000	与汇川 HMI 的专用协议，使用不同的功能码

说明：

汇川 HMI 的专用协议使用不同功能码：在访问 SM 时，使用 0x31 ,0x35，0x3f (在访问位变量的命令的基础上加了 0x30)；在访问 SD 时，使用 0x33,0x36,0x40 (在访问寄存器变量的命令的基础上加了 0x30)。

AM600 软元件有 Q 区，I 区，M 区这三种，均可以按位，按字节，按字和按双字进行访问，
如：%QX、%QB、%QW、%QD，转换如下：

QB0=（QX0.0~QX0.7）

QW0=（QB0~QB1）=（（QX0.0~QX0.7）+ （QX1.0~QX1.7））;

QD0=（QW0~QW1）=（QB0~QB4）=（（QX0.0~QX0.7）+（QX1.0~QX1.7）+（QX2.0~QX2.7）+（QX3.0~QX3.7））

寄存器地址索引规则

按 bit 寻址	按 Byte 寻址	按 Word 寻址	按 Dword 寻址		按 bit 寻址	按 Byte 寻址	按 Word 寻址	按 Dword 寻址
QX0.0	QB0	QW0	QD0		MX0.0	MB0	MW0	MD0
QX0.1					MX0.1			
QX0.2					MX0.2			
QX0.3					MX0.3			
QX0.4					MX0.4			
QX0.5					MX0.5			
QX0.6					MX0.6			
QX0.7					MX0.7			
QX1.0	QB1				MX1.0	MB1		
QX1.1					MX1.1			
QX1.2					MX1.2			
QX1.3					MX1.3			
QX1.4					MX1.4			
QX1.5					MX1.5			
QX1.6					MX1.6			
QX1.7					MX1.7			
QX2.0	QB2	QW1			MX2.0	MB2	MW1	
QX2.1					MX2.1			
QX2.2					MX2.2			
QX2.3					MX2.3			
QX2.4					MX2.4			
QX2.5					MX2.5			
QX2.6					MX2.6			
QX2.7					MX2.7			
QX3.0	QB3				MX3.0	MB3		
QX3.1					MX3.1			
QX3.2					MX3.2			
QX3.3					MX3.3			
QX3.4					MX3.4			
QX3.5					MX3.5			
QX3.6					MX3.6			
QX3.7					MX3.7			
QX4.0	QB4	QW2	QD1		MX4.0	MB4	MW2	MD1
QX4.1					MX4.1			

AM600 的 Word 型寄存器的起始地址为偶数 Byte 地址；DWord 型寄存器的起始地址为偶数 Word 地址对齐，其索引号呈 2 倍关系，这样方便地址的计算。

3.4.10 Modbus 通信帧格式说明

■ 读线圈

采用 0x01 命令码，可以读取 Q 变量；

采用 0x31 命令码，可以读取 SM 变量。

请求帧格式：从机地址 +0x01+ 线圈起始地址 + 线圈数量 +CRC 检验

序号	数据 (字节) 意义	字节数量	说明
1	从机地址	1 个字节	取值 1~247
2	0x01/0x31（命令码）	1 个字节	读线圈
3	线圈起始地址	2 个字节	高位在前，低位在后，见线圈编址
4	线圈数量 N	2 个字节	高位在前，低位在后
5	CRC 校验	2 个字节	高位在前，低位在后

响应帧格式：从机地址 +0x01+ 字节数 + 线圈状态 +CRC 检验

序号	数据 (字节) 意义	字节数量	说明
1	从机地址	1 个字节	取值 1~247
2	0x01/0x31 (命令码)	1 个字节	读线圈
3	字节数	1 个字节	值：[（N+7）/8]
4	线圈状态	[（N+7）/8] 个字节	每 8 个线圈合为一个字节，最后一个若不足 8 位，未定义部分填 0。前 8 个线圈在第一个字节，最地址最小的线圈在最低位。依次类推
5	CRC 校验	2 个字节	高位在前，低位在后

■ 读寄存器

采用 0x03 命令码，可以读取 M 变量；

采用 0x33 命令码，可以读取 SD 变量。

请求帧格式：从机地址 +0x03+ 寄存器起始地址 + 寄存器数量 +CRC 检验

序号	数据 (字节) 意义	字节数量	说明
1	从机地址	1 个字节	取值 1~247
2	0x03/0x33（命令码）	1 个字节	读寄存器
3	寄存器起始地址	2 个字节	高位在前，低位在后，见寄存器编址
4	寄存器数量	2 个字节	高位在前，低位在后，N
5	CRC 校验	2 个字节	高位在前，低位在后

响应帧格式：从机地址 +0x03+ 字节数 + 寄存器值 +CRC 检验

序号	数据 (字节) 意义	字节数量	说明
1	从机地址	1 个字节	取值 1~247
2	0x03/0x33 (命令码)	1 个字节	读寄存器
3	字节数	1 个字节	值：N*2
4	寄存器值	N*2 个字节	每两字节表示一个寄存器值，高位在前低位在后。寄存器地址小的排在前面
5	CRC 校验	2 个字节	高位在前，低位在后

■ 写单个线圈

采用 0x05 命令码，可以写 Q 变量。

采用 0x35 命令码，可以写 SM 变量。

请求帧格式：从机地址 +0x05+ 线圈地址 + 线圈状态 +CRC 检验

序号	数据 (字节) 意义	字节数量	说明
1	从机地址	1 个字节	取值 1~247
2	0x05/0x35（命令码）	1 个字节	写单线圈
3	线圈地址	2 个字节	高位在前，低位在后，见线圈编址
4	线圈状态	2 个字节	低位在前，高位在后。非 0 即为有效
5	CRC 校验	2 个字节	高位在前，低位在后

响应帧格式：从机地址 +0x05+ 线圈地址 + 线圈状态 +CRC 检验

序号	数据 (字节) 意义	字节数量	说明
1	从机地址	1 个字节	取值 1~247
2	0x05/0x35（命令码）	1 个字节	写单线圈
3	线圈地址	2 个字节	高位在前，低位在后，见线圈编址
4	线圈状态	2 个字节	低位在前，高位在后。非 0 即为有效
5	CRC 校验	2 个字节	高位在前，低位在后

■ 写单个寄存器

采用 0x06 命令码，可以写 M 变量。

采用 0x36 命令码，可以写 SD 变量。

请求帧格式：从机地址 +0x06+ 寄存器地址 + 寄存器值 +CRC 检验

序号	数据 (字节) 意义	字节数量	说明
1	从机地址	1 个字节	取值 1~247
2	0x06/0x36（命令码）	1 个字节	写单寄存器
3	寄存器地址	2 个字节	高位在前，低位在后，见寄存器值编址
4	寄存器值	2 个字节	高位在前，低位在后。非 0 即为有效
5	CRC 校验	2 个字节	高位在前，低位在后

响应帧格式：从机地址 +0x06+ 寄存器地址 + 寄存器值 +CRC 检验。

序号	数据 (字节) 意义	字节数量	说明
1	从机地址	1 个字节	取值 1~247
2	0x06/0x36（命令码）	1 个字节	写单寄存器
3	寄存器地址	2 个字节	高位在前，低位在后，见寄存器编址
4	寄存器值	2 个字节	高位在前，低位在后。非 0 即为有效
5	CRC 校验	2 个字节	高位在前，低位在后

■ 写多个线圈

采用 0x0f 命令码，可以写连续的多个 Q 变量。

采用 0x3f 命令码，可以写连续的多个 SM 变量。

请求帧格式：从机地址 +0x0f+ 线圈起始地址 + 线圈数量 + 字节数 + 线圈状态 +CRC 检验。

序号	数据 (字节) 意义	字节数量	说明
1	从机地址	1 个字节	取值 1~247
2	0x0f/0x3f（命令码）	1 个字节	写多个单线圈
3	线圈起始地址	2 个字节	高位在前，低位在后，见线圈编址
4	线圈数量 N	2 个字节	高位在前，低位在后。最大为 1968
5	字节数	1 个字节	值：值：[（N+7）/8]
6	线圈状态	[（N+7）/8] 个字节	每 8 个线圈合为一个字节，最后一个若不足 8 位，未定义部分填 0。前 8 个线圈在第一个字节，最地址最小的线圈在最低位。依次类推
7	CRC 校验	2 个字节	高位在前，低位在后

响应帧格式：从机地址 +0x05+ 线圈起始地址 + 线圈数量 +CRC 检验

序号	数据 (字节) 意义	字节数量	说明
1	从机地址	1 个字节	取值 1~247
2	0x0f/0x3f（命令码）	1 个字节	写多个单线圈
3	线圈起始地址	2 个字节	高位在前，低位在后，见线圈编址
4	线圈数量	2 个字节	高位在前，低位在后。
5	CRC 校验	2 个字节	高位在前，低位在后

■ 写多个寄存器

采用 0x10 命令码，可以写连续的多个 M 变量。

采用 0x40 命令码，可以写连续的多个 SD 变量。

请求帧格式：从机地址 +0x10+ 寄存器起始地址 + 寄存器数量 + 字节数 + 寄存器值 +CRC 检验。

序号	数据 (字节) 意义	字节数量	说明
1	从机地址	1 个字节	取值 1~247
2	0x10/0x40（命令码）	1 个字节	写多个寄存器
3	寄存器起始地址	2 个字节	高位在前，低位在后，见寄存器编址
4	寄存器数量	2 个字节	高位在前，低位在后，数量为 N
5	字节数	1 个字节	值：N*2
6	寄存器值	N*2（N*4）	
7	CRC 校验	2 个字节	高位在前，低位在后

响应帧格式：从机地址 +0x05+ 线圈起始地址 + 线圈数量 +CRC 检验。

序号	数据 (字节) 意义	字节数量	说明
1	从机地址	1 个字节	取值 1~247
2	0x10/0x40（命令码）	1 个字节	写多个寄存器
3	寄存器起始地址	2 个字节	高位在前，低位在后，见寄存器编址
4	寄存器数量	2 个字节	高位在前，低位在后，N
5	CRC 校验	2 个字节	高位在前，低位在后

3.5 串口自由协议使用

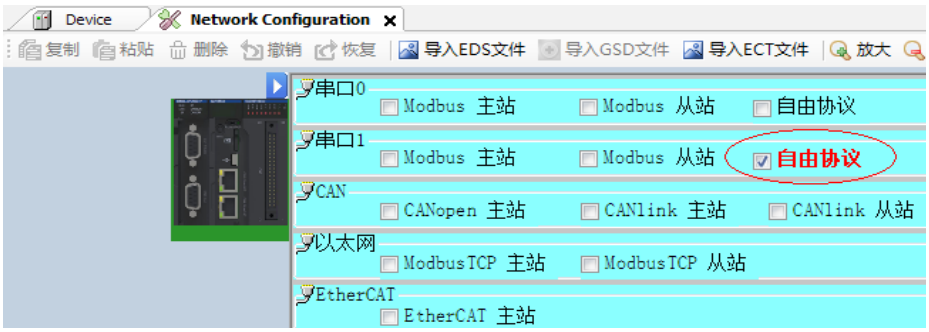
3.5.1 概要

本节针对中型 PLC 的串口自由协议通讯进行说明，适用的版本要求如下：

设备名称	版本要求
InoProshop	1.2.0 及以上
PLC 固件	1.19.70 以上

3.5.2 串口配置

以 PLC 的 COM1 为例：将【串口 1】配置成【自由协议】



3.5.3 通讯配置

配置参数如下表所示：

配置项	功能
COM 端口号	该主站物理连接选择串口 0 或串口 1
波特率	通信时的速率
奇偶校验	通信帧的校验方式
数据位	通信帧包含的实际数据位
停止位	通信时标识单个包的最后位

配置【自由协议】串口的通讯格式，需要和链接的从站设备通讯格式一致，比如：9600 8 E 1。

3.5.4 数据发送与接收寄存器

端口作为自由协议的配置如下图：

图 3-39 自由协议配置

配置项	功能
寄存器类型	可以选择 %MW 或 SD
接收字节数寄存器	当有数据接收时此寄存器将显示接收字节数
接收数据起始地址	接收到数据缓存的起始寄存器
最大接收长度	最大缓存字节数
发送字节数寄存器	此寄存器不为 0 时，将触发数据发送，发送完数据后自动复位
发送数据起始地址	将要发送的数据缓存起始寄存器
最大发送长度	一次最多发送的数据字节数

上图配置了“%MW0”，则“%MW0”的数值就是记录接收到了外部设备发送的数据帧长度；MW0 需要用户清零，如果不清零将会一直累加，直到最大接收后清零，并且接收缓存会从开始覆盖。

比如，接收数长度“%MW0”=10，那么接收数据的缓存是：“%MW1”~“%MW5”；（1 个 %MW 占用了 2 个字节）。

比如，发送数据长度“%MW300”=8，那么发送数据的缓存是：“%MW301”~“%MW304”；当“%MW300”不为 0 时，会自动触发发送，发送完成后“%MW300”会自动清零。

应用示例：

如果上图设置端口后，操作过程如下：

1） 当有数据接收时，%MW0 会显示接收到的数据字节数，接收到的数据将存放在 %MW1 开始的寄存器里面。

2） 每次接收到数据后，用户需要手动把 %MW0 里面的数据清零，以便数据可以重新以 %MW1 开始重新缓存。如果不清除 %MW0 那么数据将依次缓存。

3） 当接收到的数据超过设置的最大接收长度 256 个字节时，%MW0 会重新计数，接收到的数据也将从 %MW1 开始重新缓存。

4） 当有数据要发送时，首先把需要发送数据写入以 %MW301 起始的寄存器里面。

5） 在 %MW300 中写入要发送的数据字节数后，便开始发送以 %MW301 开始的数据。

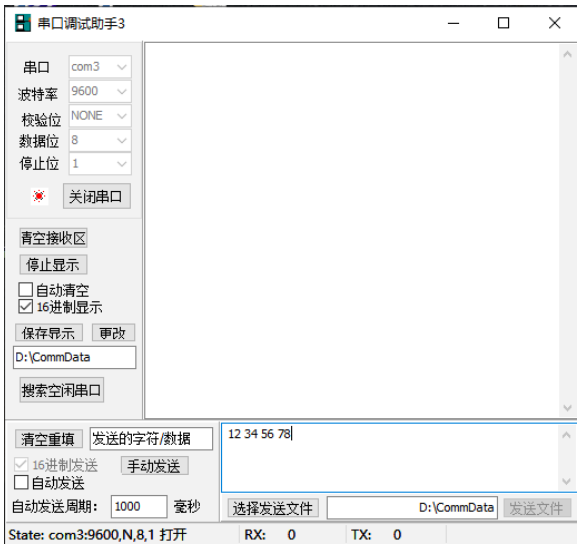
6） 数据发送完后，%MW300 将会自动清零。

7） 当写入 %MW300 的数据个数大于设置的最大发送长度 256 时，数据将按照最大长度发送。

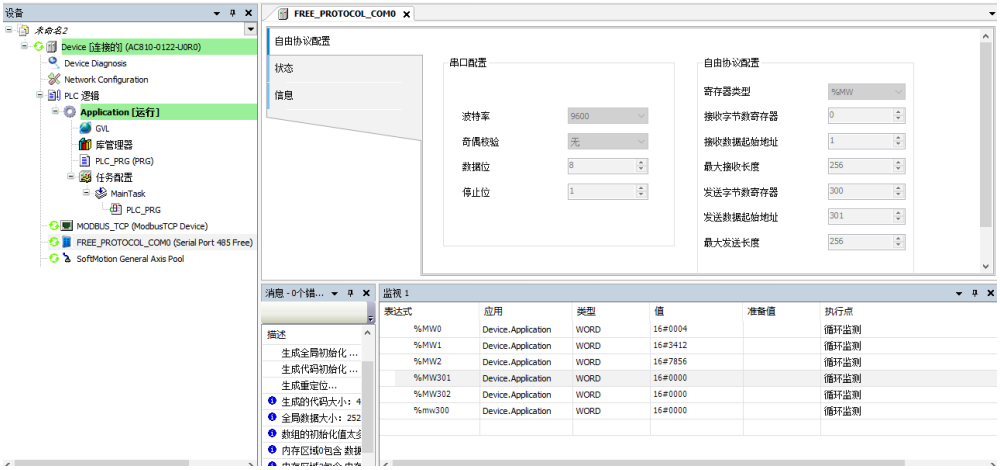
3.5.5 串口调试助手模拟通讯

1） PLC 接收数据

首先打开【串口调试助手】，配置串口通讯参数。要求与后台串口自由协议参数配置保持一致，后台参数波特率 =9600、校验方式 = 无、数据位 =8、停止位 =1。则对应串口参数配置如下图所示。



使用串口助手发送 16 进制“12 34 56 78” 4 个字节长度的数据，在监视 PLC 对应缓存区变量时可以看到 PLC 接收到数据长度“%MW0”=4 个字节，接收到的数据缓存为：“%MW1~%MW2”。 如果通过串口助手再次发送数据，PLC 端下次接收前需要手动清零“%MW0”，否则接收缓存不会更新。



2） PLC 发送数据。

如果 PLC 端要发送 4 个字节长度的数据，则对应发送数据缓存区 %MW301~%MW302 写入需发送的数据值，将 PLC 发送数据长度缓存区 %MW300 写入 4，即可自动发送数据，发送成功的后“%MW300”会自动清零，此时串口调试助手接收区域即能接收到 PLC 发送的对应数据。

3.6 Modbus TCP 设备编辑器

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“网络设备列表”中单击“MODBUS_TCP”将从站添加到网络中。

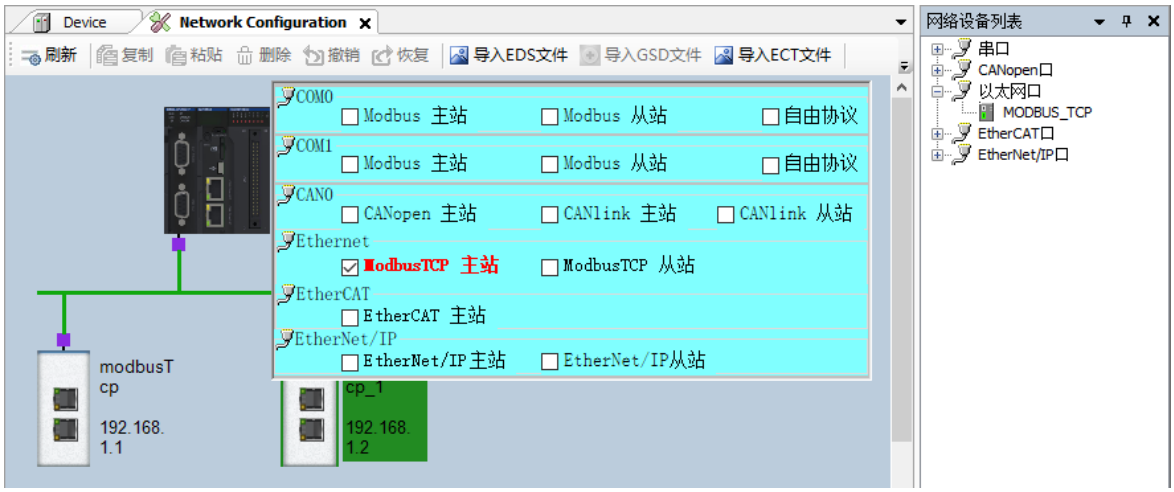


图 3-40 Modbus TCP 组态配置示例

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：

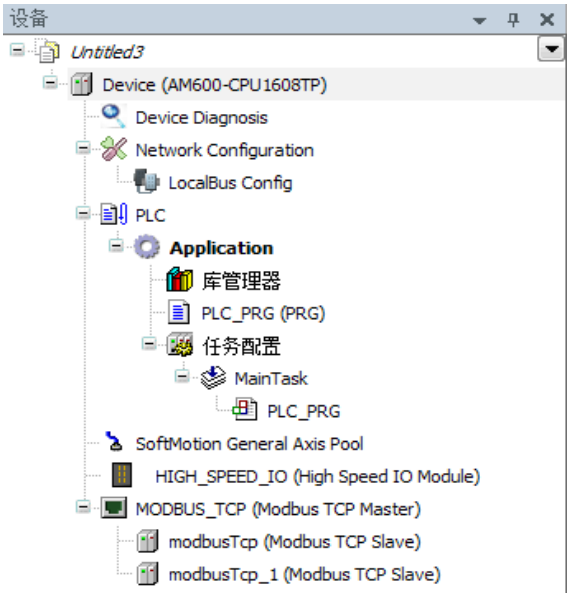


图 3-41 Modbus TCP 组态配置对应设备树示例

AM400/AM600 系列 PLC 支持 1 路 Modbus TCP 通信，可以同时做 Modbus TCP 主站和从站，做主站时，最多支持 63 个从站；

AC800 系列 PLC 支持 1 路 Modbus TCP 通信，可以同时做 Modbus TCP 主站和从站，做主站时，最多支持 128 个从站。

3.6.1 Modbus TCP 主站配置

PLC 做 Modbus TCP 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。

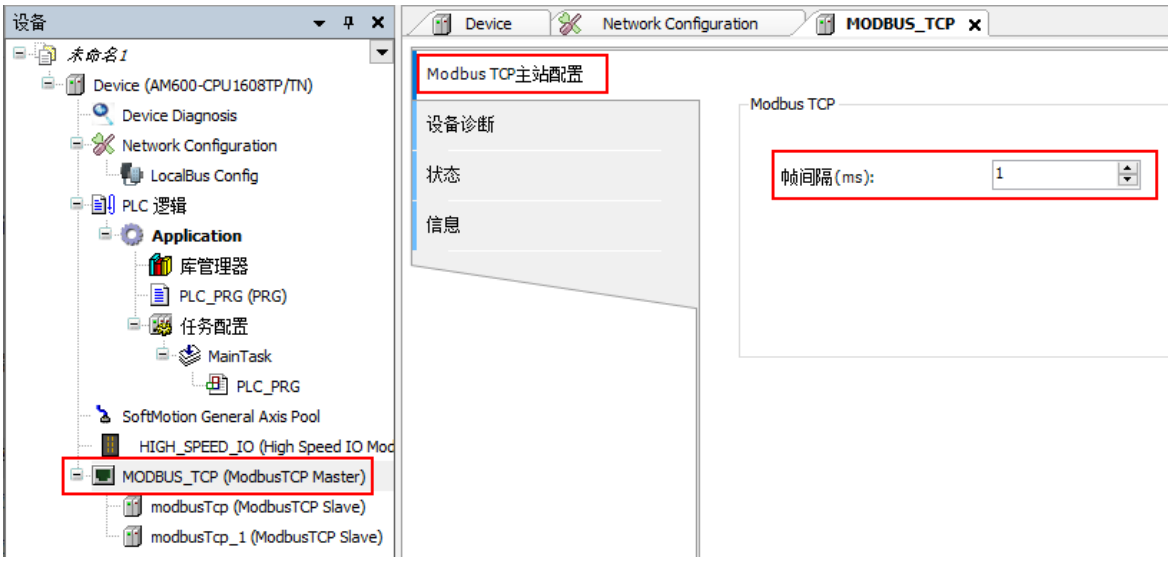


图 3-42 Modbus TCP 主站配置

帧间隔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。

3.6.2 Modbus TCP 主站通信配置

Modbus TCP 从站设置

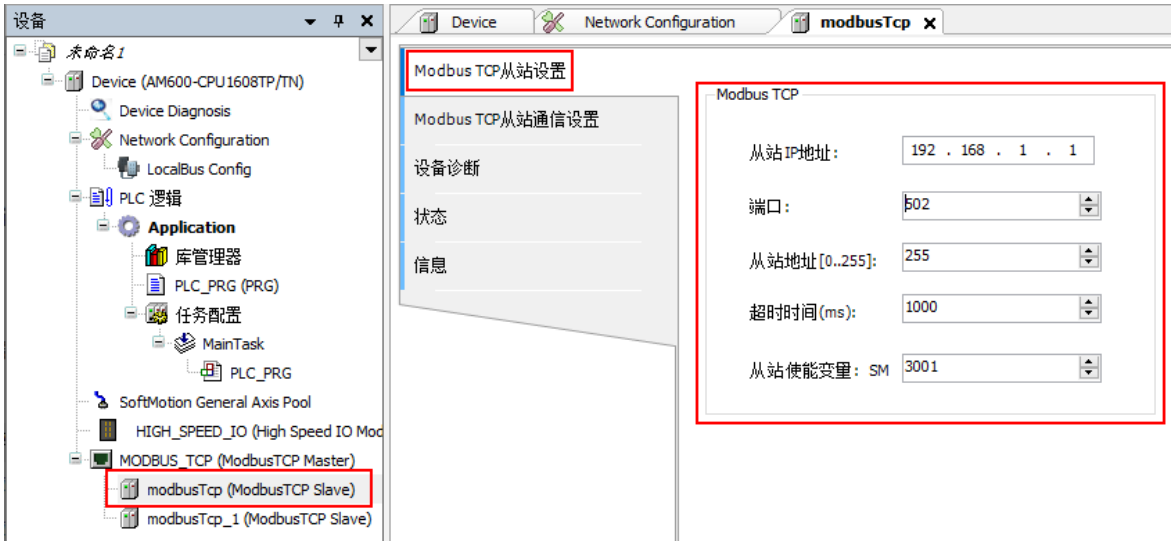


图 3-43 Modbus TCP 主站连接从站配置

配置参数：

配置项	功能
从站 IP 地址	主站连接 Modbus TCP 从站的 IP 地址
端口	主站连接 Modbus TCP 从站的 TCP 端口号
从站地址	主站连接 Modbus TCP 从站的协议站地址

配置项	功能
超时时间	主站发帧后，超过该时间从站未响应，主站报接收超时
从站使能变量	编程使能该变量后，主站开始向该从站发送通信帧

示例：

配置项	配置值
从站 IP 地址	192.168.10.16
端口	502
从站地址	05
超时时间	1000
从站使能变量	3001

Modbus TCP 主站连接 Modbus TCP 从站通信设置

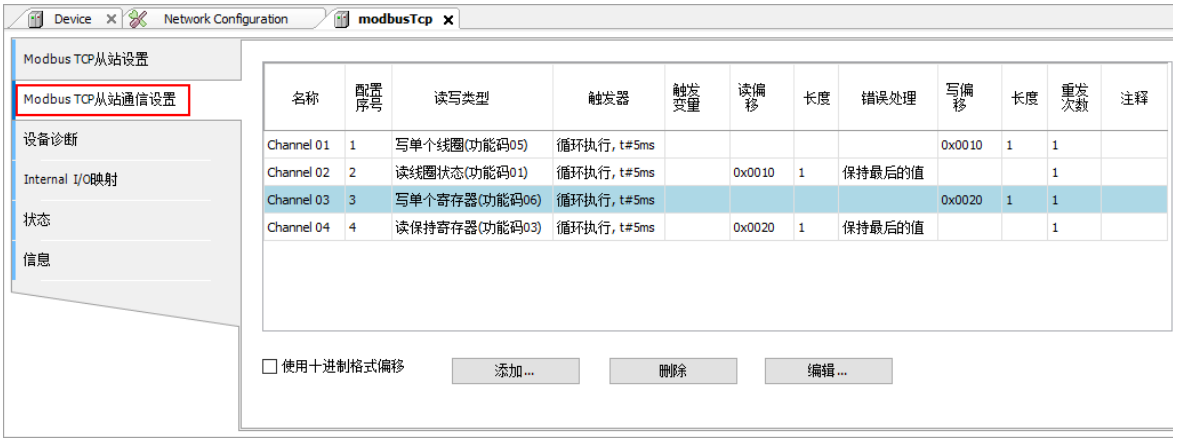


图 3-44 Modbus TCP 主站连接 Modbus TCP 从站通信设置

上图中，每一个通道代表一个独立的 Modbus TCP 请求，第三行定义了一个以 5ms 周期循环写单个寄存器（功能码 06）的操作，向偏移地址为 0x0020 的寄存器写长度为 2 字节的数据（1 个寄存器占 2 个字节）。

点击“添加…”后会出现一个为 Modbus TCP 从站添加新通道的对话框，点击“确定”按钮可新建一个通道

在“Modbus TCP 从站通道”列表中选择一个通道，单击“编辑...”按钮将会出现一个对话框，通过修改其中的参数可改变通道的配置，点击“确定”按钮可更新通道设置。

当添加或者编辑通道时，对话框中有以下参数可供使用：

图 3-45 Modbus TCP 主站连接 Modbus TCP 从站通信设置对话框

Modbus 通信设置参数

配置项	功能	
名称	通道命名的字符串	
存取类型	读线圈状态（功能码 01） 读输入状态（功能码 02） 读保持寄存器（功能码 03） 读输入寄存器（功能码 04） 写单个线圈（功能码 05） 写单个寄存器（功能码 06） 写多个线圈（功能码 15） 写多个寄存器（功能码 16）	
触发器	循环执行：周期触发的请求	循环时间：设置时间再次执行
	电平触发：编程进行改变时触发	触发变量（SM）：设置触发 SM 元件
重发次数	本次发生通信故障未获得从站返回帧，则按重发次数进行重新发送。	
注释	可以对数据进行描述的简短文本区域	
读寄存器		
起始地址	读取的寄存器开始位置	
长度	读取的寄存器个数	
错误处理	保持最后的值：使数据保持最后一次的有效值 设置为 0：使所有值归零	
写寄存器		
起始地址	写寄存器开始位置	
长度	写寄存器长度	

“长度”参数的有效范围取决于以下功能码：

功能码	类型访问	寄存器数
01	读线圈状态	1~2000
02	读输入状态	1~2000
03	读保持寄存器	1~125
04	读输入寄存器	1~125

功能码	类型访问	寄存器数
05	写单个线圈	1
06	写单个寄存器	1
15	写多个线圈	1~1968
16	写多个寄存器	1~123

Modbus TCP 从站 Internal I/O 映射

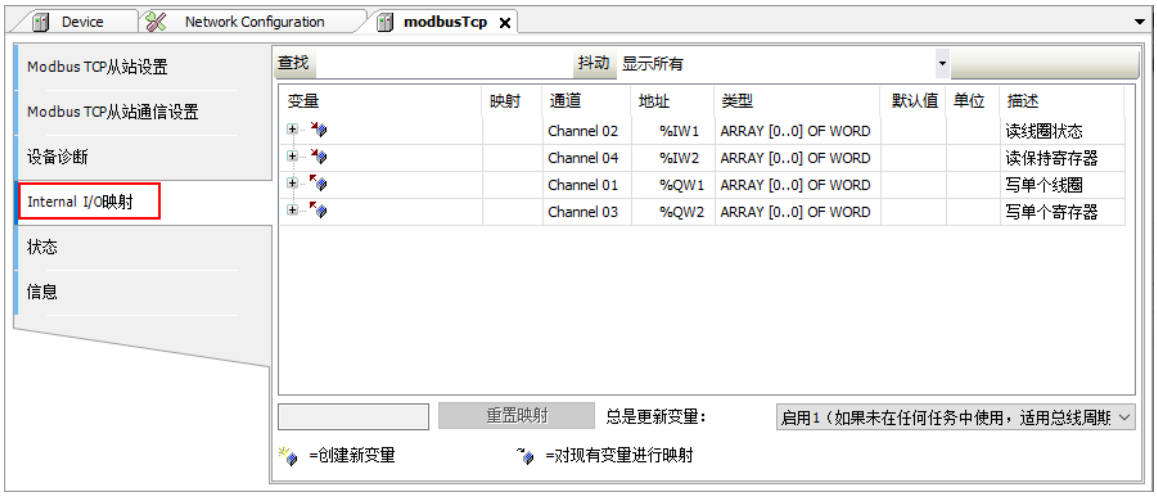


图 3-46 Modbus TCP 主站连接 Modbus TCP 从站 Internal I/O 映射

在 Modbus TCP 从站通信设置中添加主从站通信配置后，Internal I/O 映射中会自动分配每条配置的映射地址，如上图第一行的 %IW1 表示将读取的一个线圈数值映射到 %IW1 这个地址。另外，还可以通过输入助手或者直接输入示例变量路径，将程序中的自定义变量映射到 I/O 地址。

3.6.3 Modbus TCP 从站配置

PLC 做 Modbus TCP 从站时，双击设备树中的从站设备，打开 Modbus TCP 从站配置窗口，如下图所示：

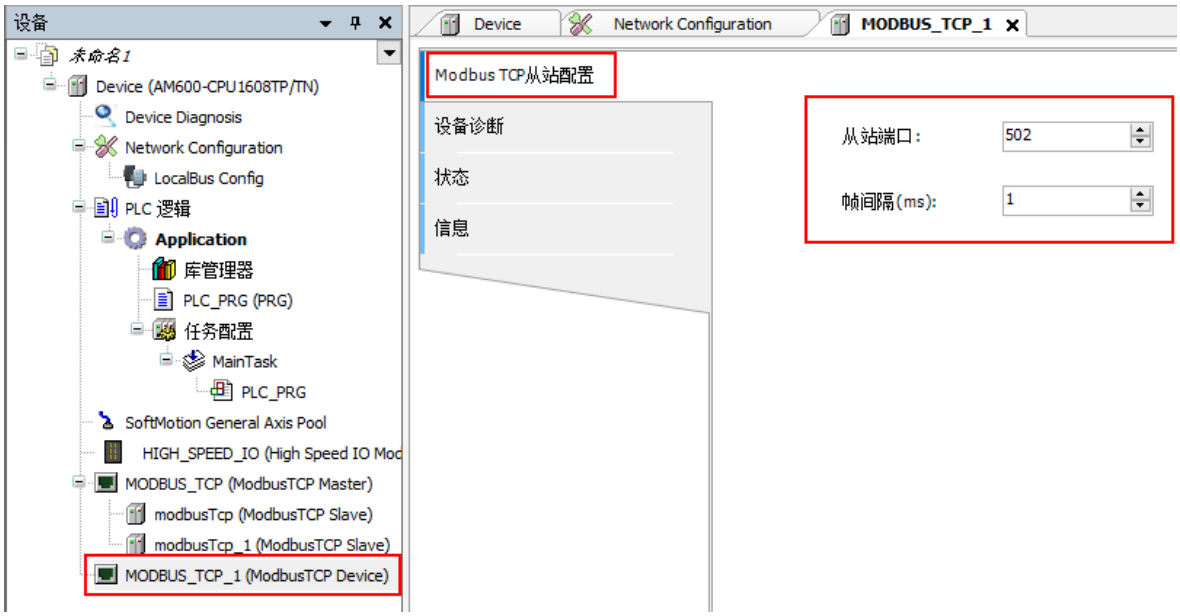


图 3-47 Modbus TCP 从站配置

配置参数：

配置项	功能
端口	Modbus TCP 从站的 TCP 端口号

配置项	功能
帧间隔	Modbus TCP 从站接收通信帧后延迟时间发送回复帧

示例：

配置项	配置值
端口	502
帧间隔	1

3.6.4 Modbus TCP 设备诊断

Modbus TCP 主站设备诊断

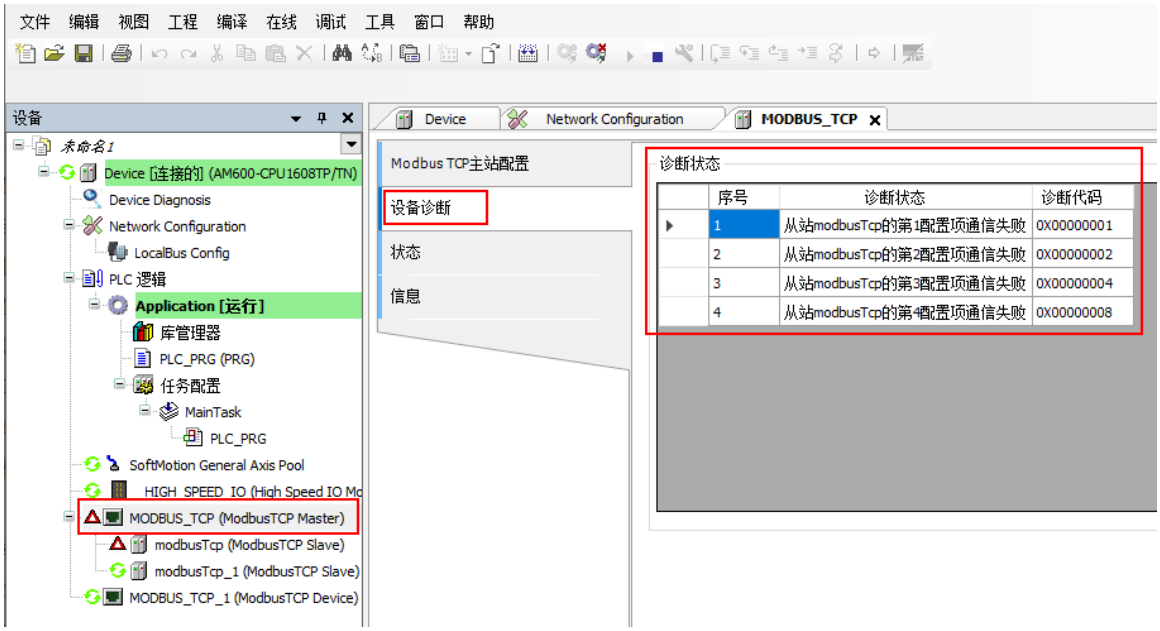


图 3-48 Modbus TCP 主站诊断

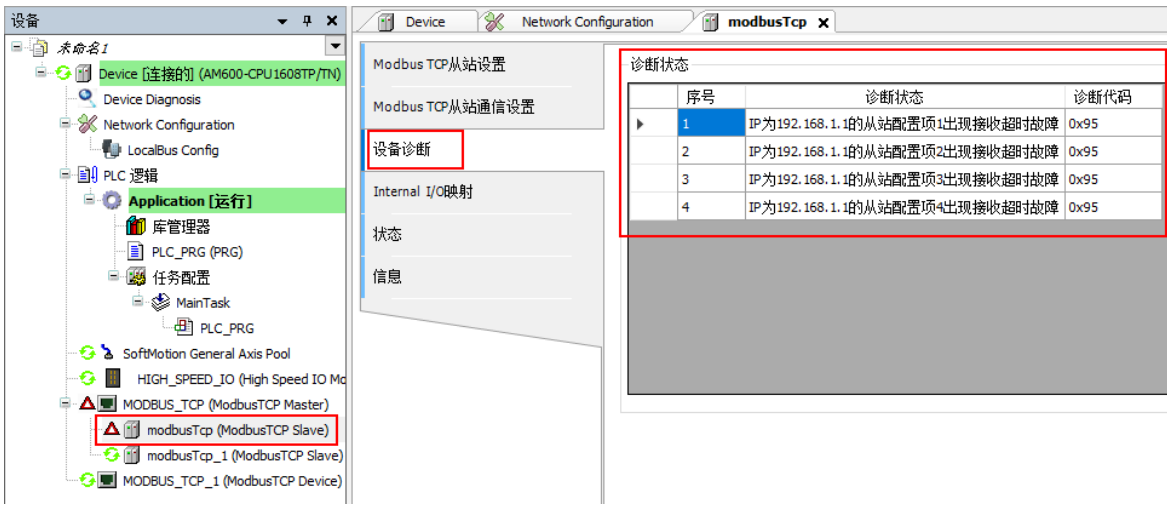


图 3-49 Modbus TCP 主站连接 Modbus TCP 从站诊断

Modbus TCP 从站诊断

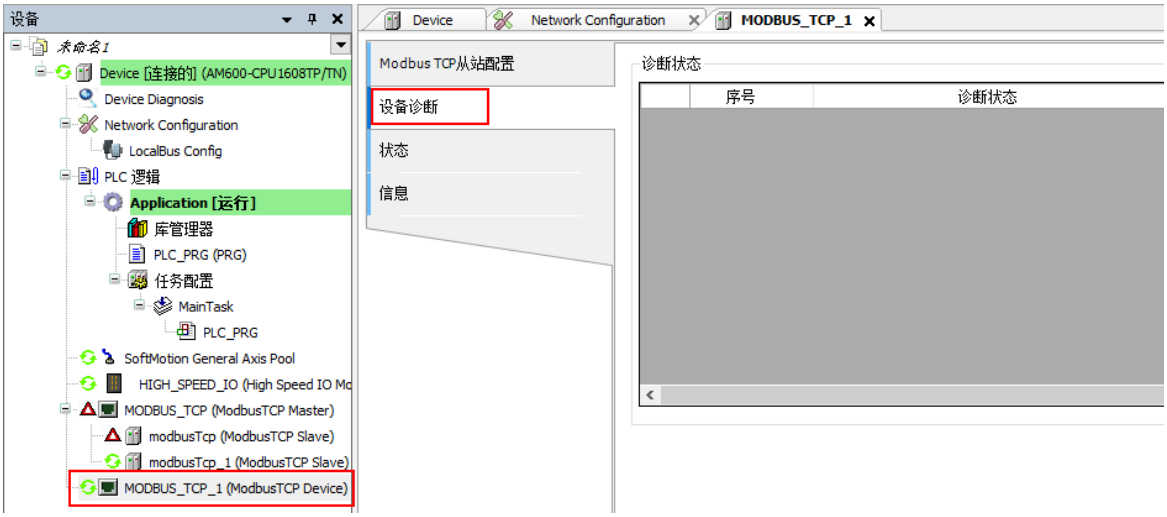


图 3-50 Modbus TCP 从站诊断

3.6.5 Modbus TCP 常见故障

Modbus TCP 主站连接 Modbus TCP 从站时发生的主要故障如下：

1：Modbus TCP 主站连接 Modbus TCP 从站配置 IP 不正确，导致主站与从站通信无法建立。

2：Modbus TCP 主站访问 Modbus TCP 从站非法地址，返回错误应答。

3：Modbus TCP 主站操作 Modbus TCP 从站写寄存器，但是 Modbus TCP 从站该寄存器只支持读不支持写操作，Modbus TCP 主站会收到 Modbus TCP 从站返回的出错应答。

错误帧格式：事务元标识符 + 协议标识符 + 长度 + 从机地址 +（命令码 +0x80）+ 错误码 +CRC 校验。

本错误帧适合所有的操作命令帧。

序号	数据 (字节) 意义	字节数量	说明
1	事务元标识符	2 个字节	MODBUS 请求 / 响应事务处理的识别码
2	协议标识符	2 个字节	0=MODBUS 协议
3	长度	2 个字节	以下字节的数量
4	从机地址	1 个字节	取值 1~247
5	命令码 +0x80	1 个字节	错误命令码
6	错误码	1 个字节	1~4

3.6.6 Modbus TCP 通信帧格式说明

■ 读线圈

采用 0x01 命令码，可以读取 Q 变量；

采用 0x31 命令码，可以读取 SM 变量。

请求帧格式：事务元标识符 + 协议标识符 + 长度 + 从机地址 +0x01+ 线圈起始地址 + 线圈数

序号	数据 (字节) 意义	字节数量	说明
1	事务元标识符	2 个字节	MODBUS 请求 / 响应事务处理的识别码
2	协议标识符	2 个字节	0=MODBUS 协议

序号	数据 (字节) 意义	字节数量	说明
3	长度	2 个字节	以下字节的数量
4	从机地址	1 个字节	取值 1~247
5	0x01/0x31（命令码）	1 个字节	读线圈
6	线圈起始地址	2 个字节	高位在前，低位在后，见线圈编址
7	线圈数量 N	2 个字节	高位在前，低位在后

响应帧格式：事务元标识符 + 协议标识符 + 长度从机地址 +0x01+ 字节数 + 线圈状态

序号	数据 (字节) 意义	字节数量	说明
1	事务元标识符	2 个字节	MODBUS 请求 / 响应事务处理的识别码
2	协议标识符	2 个字节	0=MODBUS 协议
3	长度	2 个字节	以下字节的数量
4	从机地址	1 个字节	取值 1~247
5	0x01/0x31（命令码）	1 个字节	读线圈
6	字节数	1 个字节	值：[（N+7）/8]
7	线圈状态	[（N+7）/8] 个字节	每 8 个线圈合为一个字节，最后一个若不足 8 位，未定义部分填 0。前 8 个线圈在第一个字节，最地址最小的线圈在最低位。依次类推

■ 读寄存器

采用 0x03 命令码，可以读取 M 变量；

采用 0x33 命令码，可以读取 SD 变量。

请求帧格式：事务元标识符 + 协议标识符 + 长度 + 从机地址 +0x03+ 寄存器起始地址 + 寄存器数量

序号	数据 (字节) 意义	字节数量	说明
1	事务元标识符	2 个字节	MODBUS 请求 / 响应事务处理的识别码
2	协议标识符	2 个字节	0=MODBUS 协议
3	长度	2 个字节	以下字节的数量
4	从机地址	1 个字节	取值 1~247
5	0x03/0x33（命令码）	1 个字节	读寄存器
6	寄存器起始地址	2 个字节	高位在前，低位在后，见寄存器编址
7	寄存器数量 N	2 个字节	高位在前，低位在后

响应帧格式：事务元标识符 + 协议标识符 + 长度 + 从机地址 +0x03+ 字节数 + 寄存器值

序号	数据 (字节) 意义	字节数量	说明
1	事务元标识符	2 个字节	MODBUS 请求 / 响应事务处理的识别码
2	协议标识符	2 个字节	0=MODBUS 协议
3	长度	2 个字节	以下字节的数量
4	从机地址	1 个字节	取值 1~247
5	0x03/0x33（命令码）	1 个字节	读寄存器
6	字节数	1 个字节	值：N*2
7	寄存器值	N*2 个字节	每两字节表示一个寄存器值，高位在前低位在后。寄存器地址小的排在前面

■ 写单个线圈

采用 0x05 命令码，可以写 Q 变量。

采用 0x35 命令码，可以写 SM 变量。

请求帧格式：事务元标识符 + 协议标识符 + 长度 + 从机地址 +0x05+ 线圈地址 + 线圈状态

序号	数据 (字节) 意义	字节数量	说明
1	事务元标识符	2 个字节	MODBUS 请求 / 响应事务处理的识别码
2	协议标识符	2 个字节	0=MODBUS 协议
3	长度	2 个字节	以下字节的数量
4	从机地址	1 个字节	取值 1~247
5	0x05/0x35（命令码）	1 个字节	写单线圈
6	线圈地址	2 个字节	高位在前，低位在后，见线圈编址
7	线圈状态	2 个字节	低位在前，高位在后，非 0 即为有效

响应帧格式：事务元标识符 + 协议标识符 + 长度 + 从机地址 +0x05+ 线圈地址 + 线圈状态

序号	数据 (字节) 意义	字节数量	说明
1	事务元标识符	2 个字节	MODBUS 请求 / 响应事务处理的识别码
2	协议标识符	2 个字节	0=MODBUS 协议
3	长度	2 个字节	以下字节的数量
4	从机地址	1 个字节	取值 1~247
5	0x05/0x35（命令码）	1 个字节	写单线圈
6	线圈地址	2 个字节	高位在前，低位在后，见线圈编址
7	线圈状态	2 个字节	低位在前，高位在后，非 0 即为有效

■ 写单个寄存器

采用 0x06 命令码，可以写 M 变量。

采用 0x36 命令码，可以写 SD 变量。

请求帧格式：事务元标识符 + 协议标识符 + 长度 + 从机地址 +0x06+ 寄存器地址 + 寄存器值

序号	数据 (字节) 意义	字节数量	说明
1	事务元标识符	2 个字节	MODBUS 请求 / 响应事务处理的识别码
2	协议标识符	2 个字节	0=MODBUS 协议
3	长度	2 个字节	以下字节的数量
4	从机地址	1 个字节	取值 1~247
5	0x06/0x36（命令码）	1 个字节	写单寄存器
6	寄存器地址	2 个字节	高位在前，低位在后，见寄存器值编址
7	寄存器值	2 个字节	高位在前，低位在后。非 0 即为有效

响应帧格式：事务元标识符 + 协议标识符 + 长度 + 从机地址 +0x06+ 寄存器地址 + 寄存器值

序号	数据 (字节) 意义	字节数量	说明
1	事务元标识符	2 个字节	MODBUS 请求 / 响应事务处理的识别码
2	协议标识符	2 个字节	0=MODBUS 协议
3	长度	2 个字节	以下字节的数量
4	从机地址	1 个字节	取值 1~247
5	0x06/0x36（命令码）	1 个字节	写单寄存器
6	寄存器地址	2 个字节	高位在前，低位在后，见寄存器编址
7	寄存器值	2 个字节	高位在前，低位在后。非 0 即为有效

■ 写多个线圈

采用 0x0f 命令码，可以写连续的多个 Q 变量。

采用 0x3f 命令码，可以写连续的多个 SM 变量。

请求帧格式: 事务元标识符 + 协议标识符 + 长度 + 从机地址 +0x0f+ 线圈起始地址 + 线圈数量 + 字节数 + 线圈状态。

序号	数据 (字节) 意义	字节数量	说明
1	事务元标识符	2 个字节	MODBUS 请求 / 响应事务处理的识别码
2	协议标识符	2 个字节	0=MODBUS 协议
3	长度	2 个字节	以下字节的数量
4	从机地址	1 个字节	取值 1~247
5	0x0f/0x3f（命令码）	1 个字节	写多个单线圈
6	线圈起始地址	2 个字节	高位在前，低位在后，见线圈编址
7	线圈数量 N	2 个字节	高位在前，低位在后。最大为 1968
8	字节数	1 个字节	值：值：[（N+7）/8]
9	线圈状态	[（N+7）/8] 个字节	每 8 个线圈合为一个字节，最后一个若不足 8 位，未定义部分填 0。前 8 个线圈在第一个字节，最地址最小的线圈在最低位。依次类推

响应帧格式：事务元标识符 + 协议标识符 + 长度 + 从机地址 +0x05+ 线圈起始地址 + 线圈数

序号	数据 (字节) 意义	字节数量	说明
1	事务元标识符	2 个字节	MODBUS 请求 / 响应事务处理的识别码
2	协议标识符	2 个字节	0=MODBUS 协议
3	长度	2 个字节	以下字节的数量
4	从机地址	1 个字节	取值 1~247
5	0x0f/0x3f（命令码）	1 个字节	写多个单线圈
6	线圈起始地址	2 个字节	高位在前，低位在后，见线圈编址
7	线圈数量	2 个字节	高位在前，低位在后。

■ 写多个寄存器

采用 0x10 命令码，可以写连续的多个 M 变量。

采用 0x40 命令码，可以写连续的多个 SD 变量。

请求帧格式：事务元标识符 + 协议标识符 + 长度 + 从机地址 +0x10+ 寄存器起始地址 + 寄存器数量 + 字节数 + 寄存器值

序号	数据 (字节) 意义	字节数量	说明
1	事务元标识符	2 个字节	MODBUS 请求 / 响应事务处理的识别码
2	协议标识符	2 个字节	0=MODBUS 协议
3	长度	2 个字节	以下字节的数量
4	从机地址	1 个字节	取值 1~247
5	0x10/0x40（命令码）	1 个字节	写多个寄存器
6	寄存器起始地址	2 个字节	高位在前，低位在后，见寄存器编址
7	寄存器数量	2 个字节	高位在前，低位在后，N
8	字节数	1 个字节	值：N*2
9	寄存器值	N*2（N*4）	

响应帧格式：事务元标识符 + 协议标识符 + 长度 + 从机地址 +0x05+ 线圈起始地址 + 线圈数量。

序号	数据 (字节) 意义	字节数量	说明
1	事务元标识符	2 个字节	MODBUS 请求 / 响应事务处理的识别码
2	协议标识符	2 个字节	0=MODBUS 协议

序号	数据 (字节) 意义	字节数量	说明
3	长度	2 个字节	以下字节的数量
4	从机地址	1 个字节	取值 1~247
5	0x10/0x40（命令码）	1 个字节	写多个寄存器
6	寄存器起始地址	2 个字节	高位在前，低位在后，见寄存器编址
7	寄存器数量	2 个字节	高位在前，低位在后，N

3.7 CANopen 网络

3.7.1 CANopen 通信简介

1 CANopen 总线简介

CANopen 是基于 CAN 总线的分布式自动化控制设备的工业通信协议族。它由制造商与用户协会 CiA（CAN in Automation）推广发展，并于 2002 年底标准化，标准号为 CENELEC EN 50325-4。CANopen 定义了应用层协议、通信层协议和多种应用协议。CANopen 总体结构如下：

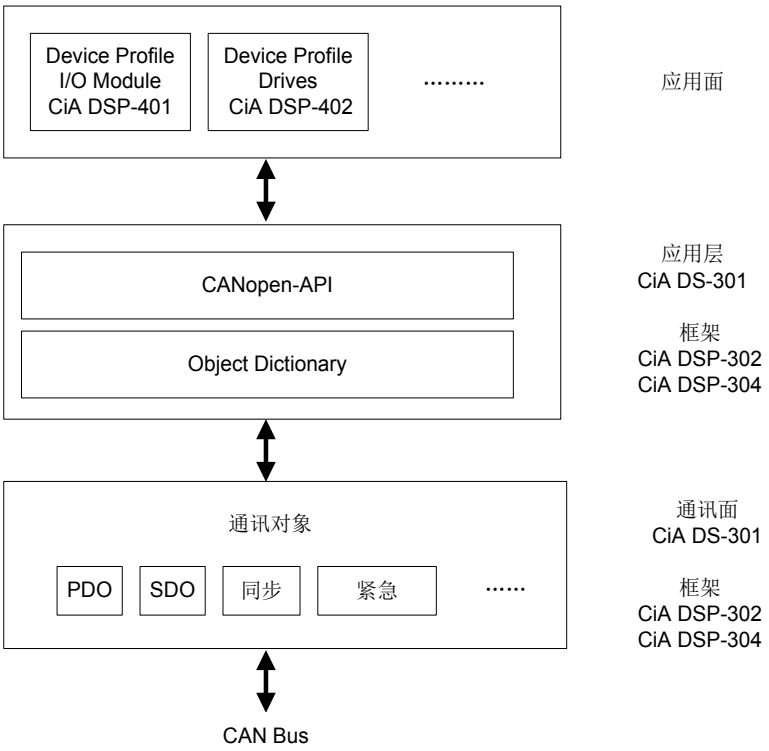


图 3-51 CANopen 总体结构

应用层为应用提供确认和不带确认的服务，并定义了通信对象。这些服务可用于从一个服务端请求数据。

通信对象用于数据交互。它们用于交换过程数据和服务数据，进程管理或系统时钟同步，错误状态管理和节点状态的控制和监测。这些对象由结构体，传输类型和 CAN 标识定义。通信对象的典型参数包括：用于数据传输的 CAN 标识，消息的传输类型，禁止时间或事件时间，通信协议对它们进行了定义。

每一个 CANopen 设备都以对象字典作为主要的数据结构。对象字典是应用层和 CAN 总线通信之间的首要数据交互介质。可以通过特定消息从应用层和 CAN 总线两边访问对象字典的入口。这些对象字典的入口，可以被看做是变量或者程序员定义的区域。

对象字典的每一个入口都具有一个索引和子索引。使用索引结构，可以准确的定位对象字典的入口。CANopen 协议栈提供标准的 API 函数来定义对象字典的入口，包括这些入口的读写属性。通过通信对象，还可以通过

CAN 总线访问对象字典。

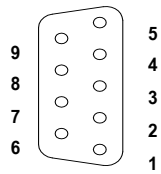
在对象字典中，必须定义每一个入口的属性，包括数据类型、访问权限、过程数据对象的数据传输及变量范围等。

- PDO（Process data objects），过程数据对象，用于过程实时数据的传输。
- SDO（Service Data Objects），服务数据对象，用于访问一个设备的对象字典，对参数进行配置或非实时性数据的传输。
- SYNC 消息不包含数据，由生产者向 CAN 网络周期性的发送，触发节点 PDO 数据的发送。
- Emergency 消息由设备内部致命错误触发，由应急错误代码，错误寄存器和制造商特定错误构成。

CiA DS-301 规范详细定义了应用层和通信协议，CiA DSP-302 规范详细定义了可编程 CANopen 设备的框架，CiA DSP-304 规范详细定义了安全冗余数据传输的框架，而特定设备的数据描述定义在由各自的设备协议构成的应用协议中，如 CiA DSP-401 规范定义了 I/O 模块的数据格式，CiA DSP-402 规范定义了驱动控制的数据格式。

2 硬件端口

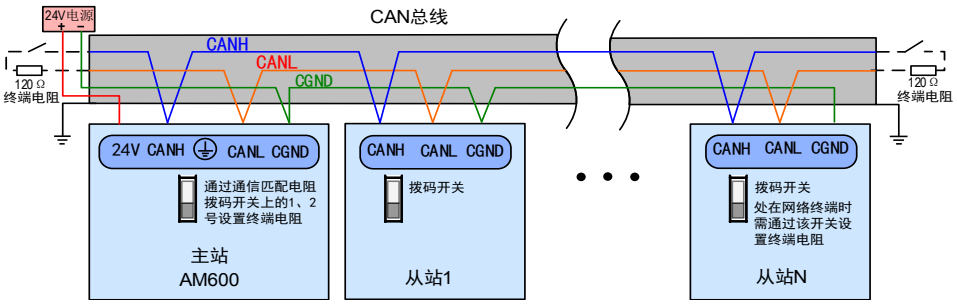
CANopen 采用 DB9 接头进行数据传输，DB9 引脚定义如下：

示意图	引脚	信号定义
	PIN2	CANL
	PIN7	CANH
	PIN3	CGND

CAN 总线推荐使用带屏蔽双绞线连接，总线两端分别连接两个 120Ω 终端匹配电阻防止信号反射，屏蔽层一般使用单点可靠接地，固定线缆时不要和交流电源线、高压线缆等捆扎在一起，避免通信信号受干扰影响。

组网示意图

CAN 总线连接拓扑结构如下所示，CAN 总线推荐使用带屏蔽双绞线连接，总线两端分别连接两个 120Ω 终端匹配电阻防止信号反射。屏蔽层一般使用单点可靠接地。



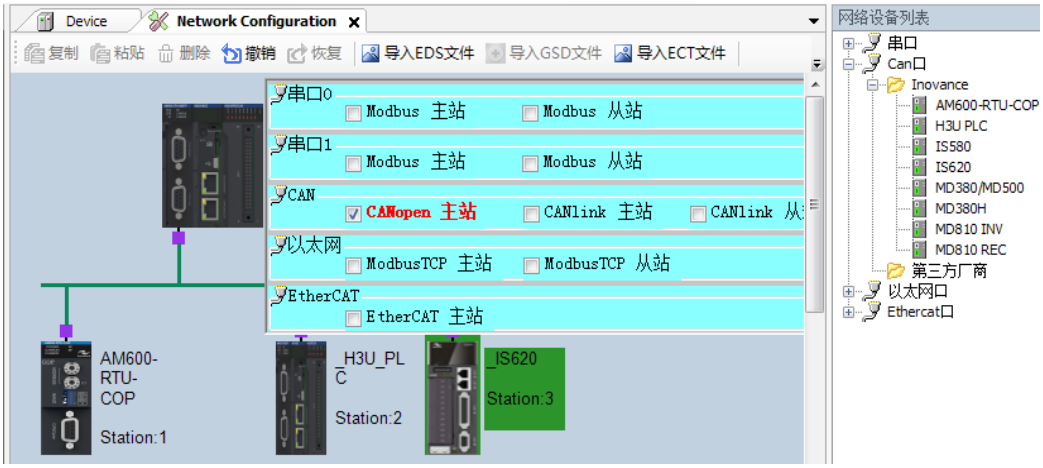
3 CANopen 使用的一般过程

CANopen 一般使用流程如下：

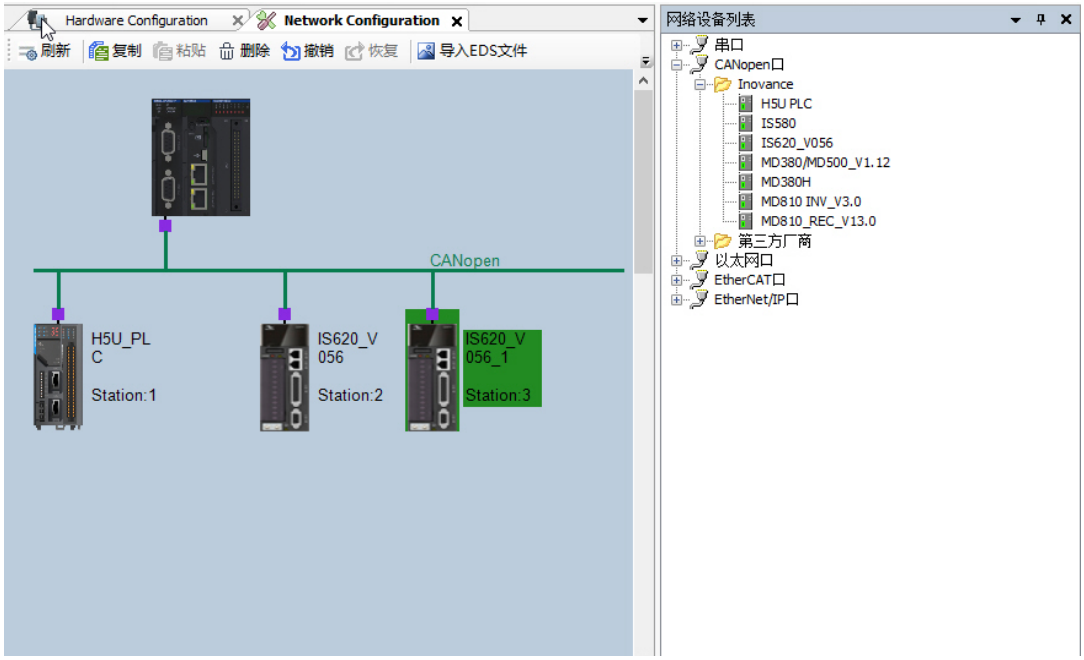
1） 设计 CANopen 硬件网络结构。

2） 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，并且添加 CANopen 总线任务“CANopen”，默认 CANopen 总线使用此任务进行 I/O 刷新。

3） 根据硬件结构在网络组态中添加 CANopen 从站及模块。如果是第三方从站，可以在网络组态中通过导入 EDS 文件导入第三方从站，然后添加第三方从站。



4） 如果是 AM600 从站，需要在硬件组态中添加 I/O 模块（如下图，双击右侧列表中的模块即可完成添加）。CANopen 从站指 CANopen 远程设备。



5） 设置合适的主站配置参数、从站配置参数及模块配置参数。一般情况下，从站节点 ID 会自动生成，PDO 及映射根据 EDS 文件会自动生成，一些特殊的配置需要手动修改。

配置主站参数及从站参数时，主站波特率与从站的节点 ID 需要和从站的波特率及从站节点 ID 拨码开关匹配。

图 3-52 主站参数配置

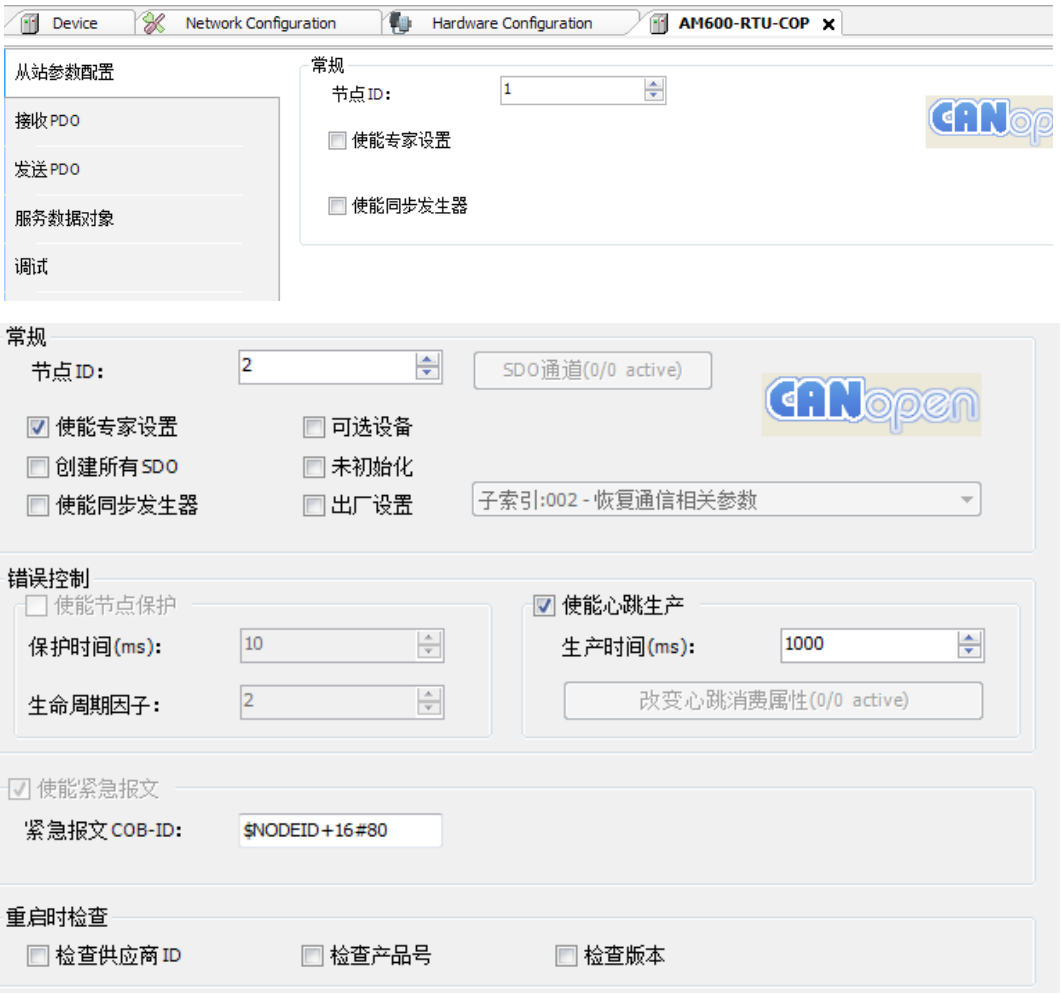


图 3-53 从站参数配置

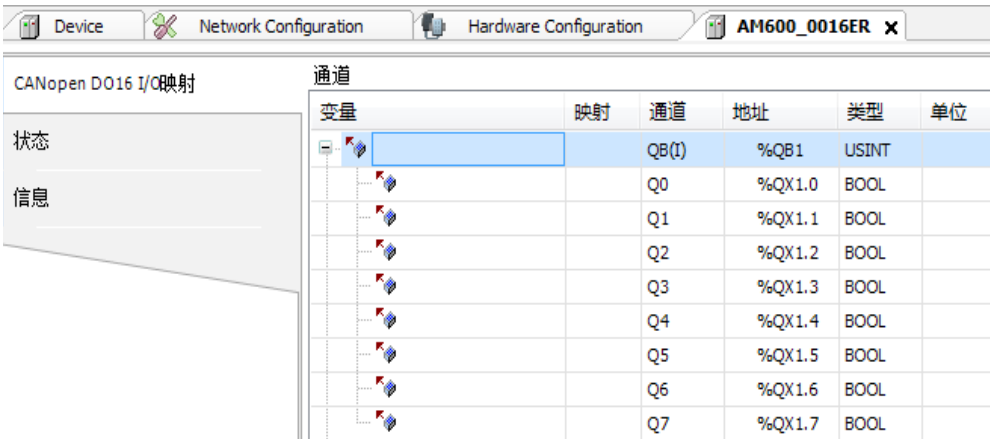


图 3-54 模块参数配置

另外软件还提供了软元件用于获取 CANopen 从站状态及 CiA-DSP405 库用于从站管理和操作。

3.7.2 CANopen 主站配置

1 主站配置

如果主站设备是 AC800 系列 PLC，则需要对转换模块进行配置，如下图所示。

图 3-55 CANopen 主站配置界面

转换模块有两种：内置 CAN 卡和网络转 CAN。使用内置 CAN 卡时只需配置模块类型为内置 CAN 卡；使用网络转 CAN 时需要配置模块类型、IP 地址、子网掩码，网络转 CAN 目前支持设置的网段应与 AC800 的网口 A 或 B 网段保持一致，否则 AC800 扫描不到网络转 CAN。

NOTE

网络转 CAN 模块的 IP 地址和子网掩码发生修改后，登录时配置信息会生效，网络转 CAN 模块会重启并重连，请在网络转 CAN 模块重启完成后、SYS 指示灯由红灯变为绿色并闪烁 10 秒后，再启动应用程序。

网络管理

- 节点 ID：主站在 CANopen 网络唯一标示号，默认 127，范围 1-127，必须是十进制进入数。
- 检查并修正配置：请参见 " 检查并修正配置”。
- 故障停机设置：请参见 " 故障停机设置”。
- 波特率：总线上用于传输的波特率。单位 Kbit/s, 可以设置以下的波特率：10, 20, 50, 100, 125, 250, 500, 800 及 1000。默认值 500。

NOTE

如果 CPU 模块处于网络的首端或末端，CANopen 端口的匹配电阻需要拨到 ON 的位置。通讯距离与波特率有关，要设置合理的波特率。

- 程序运行过程中禁止 SDO,NMT 访问：如果这个复选框被激活，在应用程序运行的时候，用户不能通过 SDO 和 NMT 访问从站，例如用户不能在用户程序中或者在从站调试页面通过 SDO 和 NMT 访问从站。
- 网络负载：总线运行过程中 CANopen 网络实时负载。在登陆 PLC 后才能显示网络负载。

同步

- 使能同步生产：如果启用这个选项（默认：禁用）， 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。如果从站启用了同步生产，使用的也是此 COB-ID。
- 同步周期 (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 2000us-4294967000us，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。值为 0 不能修改。

心跳

心跳是另外一种节点保护机制：不同于节点守护功能，此功能可以由主站或者从站触发。通常情况下主站发送心跳到从站设备，从站设置消费的主站节点 ID，实现从站对主站的监护。

- 使能心跳生产：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。
- 生产时间 (ms)：心跳信息发送的时间间隔，单位为毫秒，范围为 2ms-32767ms，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。值为 0 不能修改。

如果主站是 AC800，则需要额外对外置转换模块进行配置，如下图所示。

图 3-56 图 CANopen 外置模块配置界面

2 检查并修正配置

当多个从站被添加到 CANopen 系统中，由于不同从站 EDS 文件可能默认配置了 COB-ID 或者修改了从站的节点 ID，可能导致从站或者主站的节点 ID 重复或者 COB-ID 冲突。在 CANopen 主站配置界面点击“检查并修正配置”，再点击“对所有都应用此建议”，可以解决重复的节点 ID 或者冲突的 COB-ID。

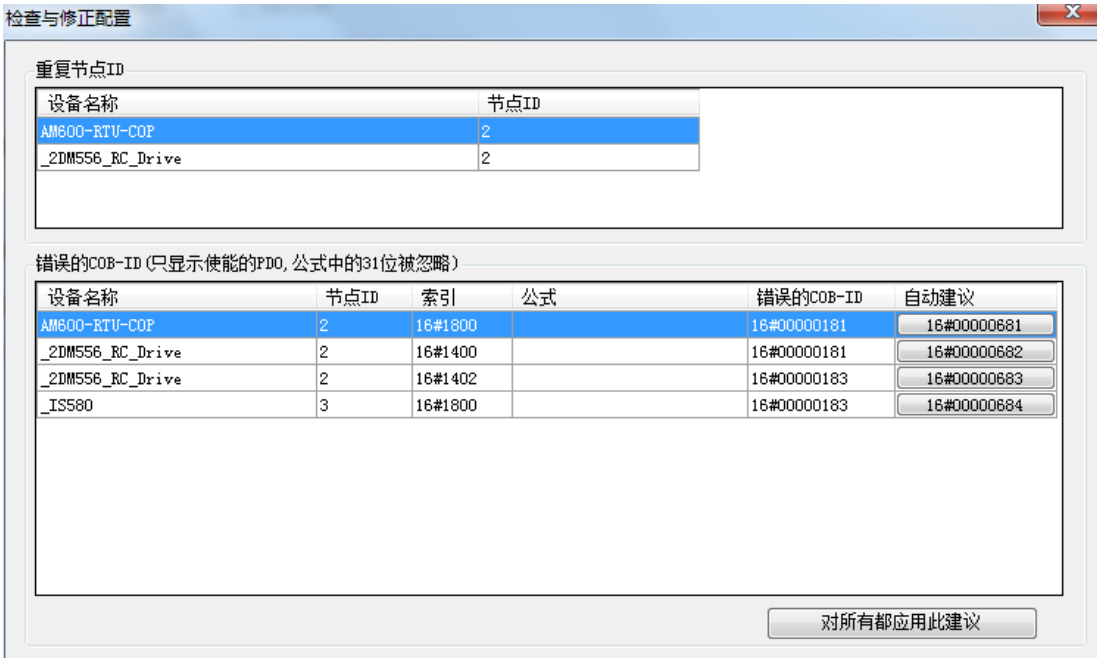


图 3-57 检查与修正配置界面

重复节点 ID

该框中包含具有相同节点 ID 的所有从站设备的列表。用户可以通过编辑“节点 ID”列来重新分配新的节点 ID，修改完成后重复节点 ID 自动取消。

错误的 COB-ID

该框中显示所有冲突的和非法 COB-ID 的所有从站设备的列表。用户 3 种修改 COB-ID 方式：

■ 通过编辑“错误的 COB-ID”列，手动修改当前从站索引对应的 COB-ID；

■ 点击“自动建议”列对应的按钮，按照按钮显示值修改当前从站索引对应的 COB-ID；

■ 点击“对所有都应用此建议”按钮，按照所有“自动建议”按钮显示值修改所有错误的 COB-ID

修改完成后，无错误的 COB-ID 从站自动取消。

3 故障停机设置

故障停机功能，主要是当从站或者模块出现故障或者组态不一致时，从站是否停止运行。此设置为故障停机开关，此功能只支持 AM600 CANopen 从站。

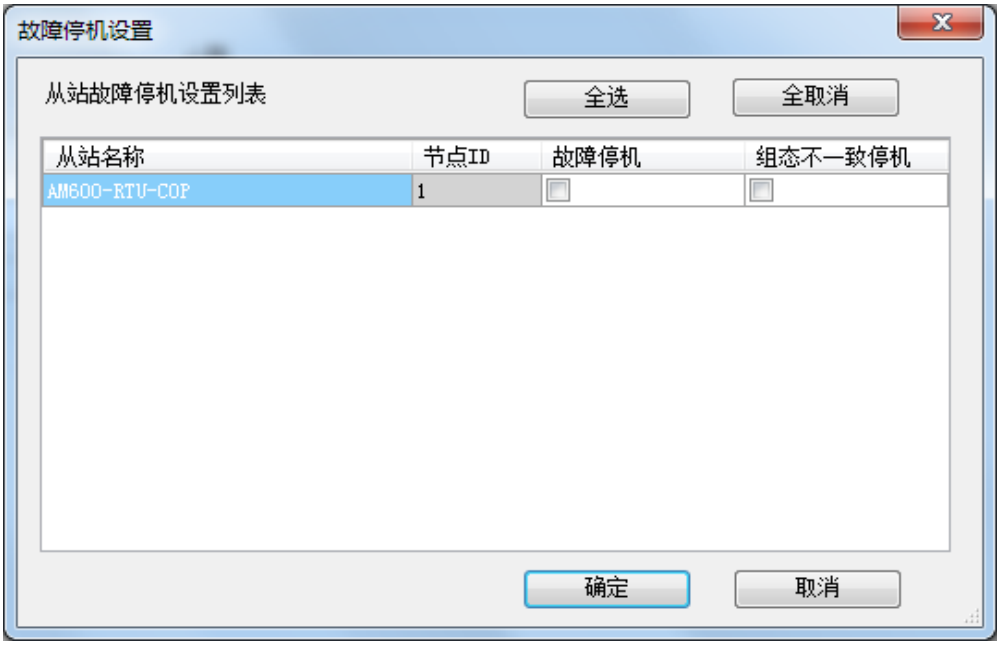


图 3-58 故障停机设置

- 从站故障停机设置列表 : 显示并设置从站故障或者组态不一致时是否停机。
- “故障停机”列控制当从站或者模块出现故障时从站是否停机。如果选中了“故障停机”功能，对于从站，如果从站本身出现故障，从站停止运行；对于模块，如果 I/O 模块本身的诊断上报功能激活并且模块出现故障，从站停止运行；
- “组态不一致”列控制从站下 I/O 模块组态不一致从站是否停机，如果选中了“组态不一致停机”，当从站下 I/O 类型不匹配、模块少于实际模块数、多于实际模块数时，从站停止运行。
- 全选 / 全取消 : 激活 / 不激活故障停机列表所有从站设置。
- 确定 / 取消 : 保存 / 不保存故障停机设置。

4 CANopenMaster I/O 映射

I/O 映射的总体描述和本对话框的使用请参照 I/O 映射（链接有误）链接。

5 状态

CANopen 总线设备或模块的状态配置编辑器可显示状态信息（如: “运行”，“停止”）和内部总线系统的状态。

6 信息

如果当前设备可用，将会显示下列基本信息：名称、供应商、类型、版本号，模块号、描述。

3.7.3 CANopen 从站配置

CANopen 从站配置主要是配置从站基本参数、PDO 配置、SDO 配置及调试功能

1 从站参数配置

图 3-59 从站参数配置

常规

- 节点 ID: 从站在 CANopen 网络唯一标示号范围 1-127（十进制），需要和从站本身标示（如拨码开关）一致。
- SDO 通道：暂不支持。
- 使能专家配置：激活此选项，用户可以配置专家参数，如从站节点保护、心跳生产、应急报文、重启检查、PDO 映射操作、系统 SDO 显示、SDO 异常跳转。
- 可选设备：暂不支持。
- 创建所有 SDO：创建对象字典中具有可写属性的 SDO 对象，如对象访问属性为 rw，wo，rwr，rww。创建的 SDO 在服务数据对象界面显示。
- 未初始化：暂不支持。
- 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。
- 出厂设置：如果激活此选项，在下载配置或者配置从站之前，从站参数将被复位。复位参数类型取决于复位类型列表的选择。定义规则如下：

1） “子索引 :001”： 所有参数都将会被重新复位。

2） “子索引 :002”： 相关通讯参数 (索引 1000h - 1FFFh 制造商指定的通讯参数) 将会被重新复位。

3） “子索引 :003”： 相关应用参数 (索引 6000h - 9FFFh 制造商指定应用参数) 将会被重新复位。

4） “子索引 :004” - “子索引 :127”: 制造商自定义的参数将被复位。

5） “子索引 :128” - “Sub254”: 保留。

复位参数类型列表内容基于当前的对象字典（EDS 文件），来自于 EDS 文件索引 1011 的解析，子索引和上述

定义规则一一对应。

错误控制

错误控制主要用于检测节点的在线状态，包括节点保护和心跳。

节点保护用于主站检测从站的在线状态，主站定时发送节点守护信息，从站响应此信息，如果在节点守护时间（保护时间 x 生命周期因子）内，从站没有响应，主站认为从站丢失。

心跳可以由从站生产，也可以由主站生产，生产者把心跳报文广播到 CAN 总线上，心跳消费者消费心跳，如果节点设置了心跳消费，在设定的心跳消费的时间内，没有检测到节点 ID 对应的心跳生产，则认为此节点丢失。一般从站消费主站的心跳，用于检测主站的在线状态。

- 使能节点保护: 激活节点保护功能，节点保护和心跳生产是互斥的。主站在保护时间内定时发送节点保护阵，如果从站没有在节点守护时间（保护时间 x 生命周期因子）内给出包含特定防护 COB-ID（通信对象标识）的响应，则从站认为掉线状态。
- 保护时间：主站定时发送节点保护帧间隔，范围为 10ms-65535ms，并且为总线任务周期的整数倍。
- 生命周期因子: 和保护时间共同使用，如果在节点守护时间（保护时间 x 生命周期因子）内，从站没有响应，主站认为从站丢失。范围为 1-255。
- 使能心跳生产：激活从站心跳生产，从站以生产时间间隔定时发送心跳生产帧，和节点保护互斥。
- 生产时间：从站发送心跳生产帧间隔，范围为 10ms-32767ms，并且为总线任务周期的整数倍。
- 改变心跳消费属性：打开一个对话框，设置从站消费的心跳生产者。通过设置心跳消费，此从站可以检查对应的心跳生产从站在线状态。一般从站消费主站的心跳生产。

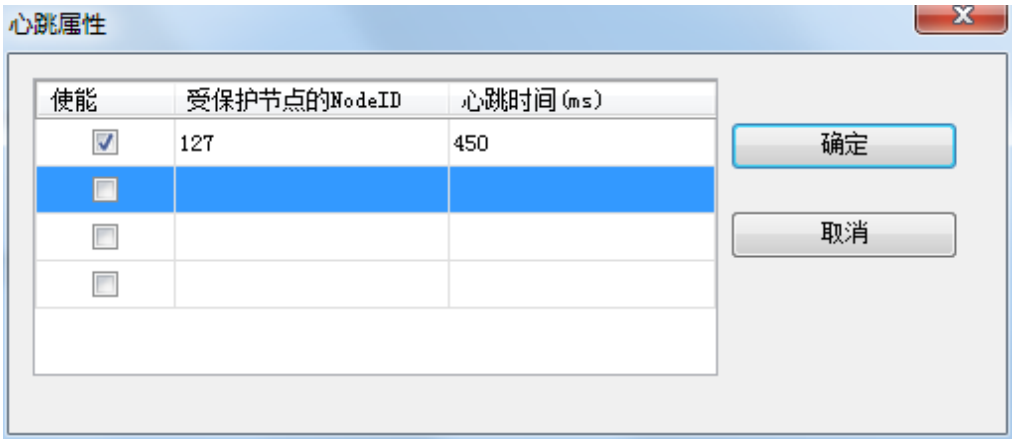


图 3-60 心跳消费界面

心跳消费配置列表用于配置被消费的心跳生产者。只有使能后才能配置。

使能后，“受保护的节点的 NodeID” 默认为主站心跳生产节点 ID，如果主站没有使能心跳生产，此 NodeID 为 0，范围 1-127。心跳时间默认为主站心跳生产时间 x 1.5，范围 1-65535。

紧急报文

- 使能紧急报文: 激活从站紧急报文功能，如果这个选项被激活，从站将通过紧急报文 COB-ID 发送紧急消息。这些紧急信息可以通过 CiA405 library (RECV_EMCY_DEF, RECV_EMCY) 函数库提供的函数获取紧急消息。
- 紧急报文 COB-ID：从站发送紧急报文的 COB-ID，默认为 $NODEID+16#80，NodeID 为此从站的节点 ID。此 COB-ID 格式为 $NODEID+16#+16 进制数字、16#+16 进制数字或者 10 进制数字。（示例说明）

重启时检查

- 检查供应商 ID：激活供应商 ID 检查功能，如果这个选项被激活，从站将检查对象字典中供应商 ID（索引 1018，子索引 01）和从站本身的供应商 ID 是否匹配，如果不匹配，从站不能正常运行。

- 检查产品号：激活产品号检查功能，如果这个选项被激活，从站将检查对象字典中产品号（索引 1018，子索引 02）和从站本身的产品号是否匹配，如果不匹配，从站不能正常运行。
- 检查版本：激活版本检查功能，如果这个选项被激活，从站将检查对象字典中版本（索引 1018，子索引 03）和从站本身的版本是否匹配，如果不匹配，从站不能正常运行。

2 接收 PDO

PDO（过程数据对象）用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。

PDO 包含通信参数和映射参数。通信参数包括通信唯一标示 COB-ID、传输类型、传输控制等。PDO 映射参数表示此 PDO 传输数据来源于对象字典的索引和子索引。

接收 PDO 来自于对象字典中从索引 1400 开始到索引 1600 结束的对象，每个 PDO 对应的通信参数默认值分别来自于其对应的子索引，接收 PDO 可映射的对象来自于对象字典中可写访问权限的对象，如访问权限为 RW、RWW、WO。

NOTE

1) 在非专家模式下只能修改 PDO 通信参数，不能增加、删除 PDO 及 PDO 映射；
2) AM600 从站只能修改 PDO 通信参数，不能增加、删除 PDO 及 PDO 映射，PDO 映射随着 AM600 I/O 的添加而增加。

接收 PDO 映射 AM600 输出模块，每个模块和固定的索引对应，对应关系如下表：

模块类型	索引（16 进制）	子索引（16 进制）	数据类型
DO16	6300	01-10	unsigned short int
DA4	6411	01-10	short int

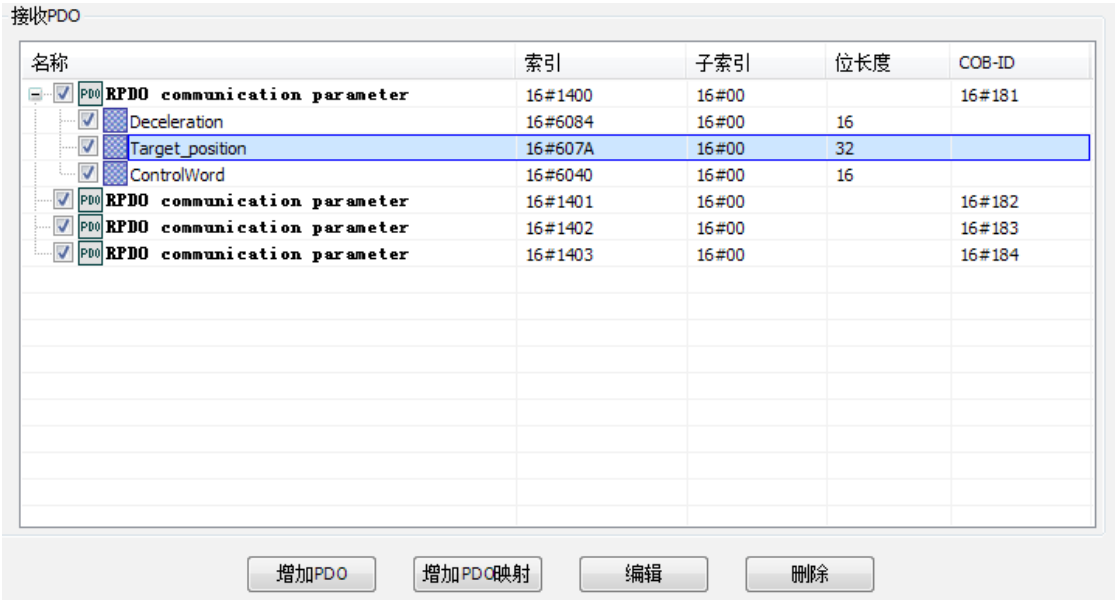


图 3-61 接收 PDO 界面

- 增加 PDO：增加一个 PDO，增加的 PDO 添加到最后面。从站中最大接收 PDO 个数由对象字典索引 1400-1600 个数决定的，超过最大个数不能添加。添加的 PDO 名称和索引从对象字典根据使用顺序自动获取，不能修改。

  添加 PDO 会弹出一个 PDO 属性设置对话框，详见 PDO 属性。

- 增加 PDO 映射：在当前所选的 PDO 中增加一个 PDO 映射，此 PDO 映射增加到当前 PDO 的后面。PDO 映射最大 64 位，超过 64 位不能添加。PDO 映射来自于对象字典，接收 PDO 可映射的对象来自于对象字典中可写访问权限的对象，如访问权限为 RW、RWW、WO。对于非 AM600 从站，添加一个接收 PDO 映射，会在 CANopen 添加 PDO 映射弹出添加对象对话框，详见添加对象。
- 编辑：编辑当前所选的 PDO 通信参数或者 PDO 映射参数。如果所选为 PDO，编辑 PDO 通信参数，如果

所选为 PDO 映射，则编辑 PDO 映射。如果是 AM600 从站，只能编辑通信参数。

- 删除：删除当前所选的 PDO 或者 PDO 映射。如果所选为 PDO，则删除 PDO，如果所选为 PDO 映射，则编辑 PDO 映射。如果是 AM600 从站，不能执行删除操作。

3 发送 PDO

PDO（过程数据对象）用于主站和从站之间的实时数据传输，发送 PDO 为从站向从主站发送的实时数据。

发送 PDO 来自于对象字典中从索引 1800 开始到索引 1A00 结束的对象，每个 PDO 对应的通信参数默认值分别来自于其对应的子索引，发送 PDO 可映射的对象来自于对象字典中可读访问权限的对象，如访问权限为 RW、RWR、RO、CONST。

NOTE

1) 在非专家模式下只能修改 PDO 通信参数，不能增加、删除 PDO 及 PDO 映射；
2) AM600 从站只能修改 PDO 通信参数，不能增加、删除 PDO 及 PDO 映射，PDO 映射随着 AM600 I/O 的添加而增加。

发送 PDO 映射 AM600 输入模块，每个模块和固定的索引对应，对应关系如下表：

模块类型	索引（16 进制）	子索引（16 进制）	数据类型
DI16	6100	01-10	unsigned short int
AD4	6401	01-10	short int

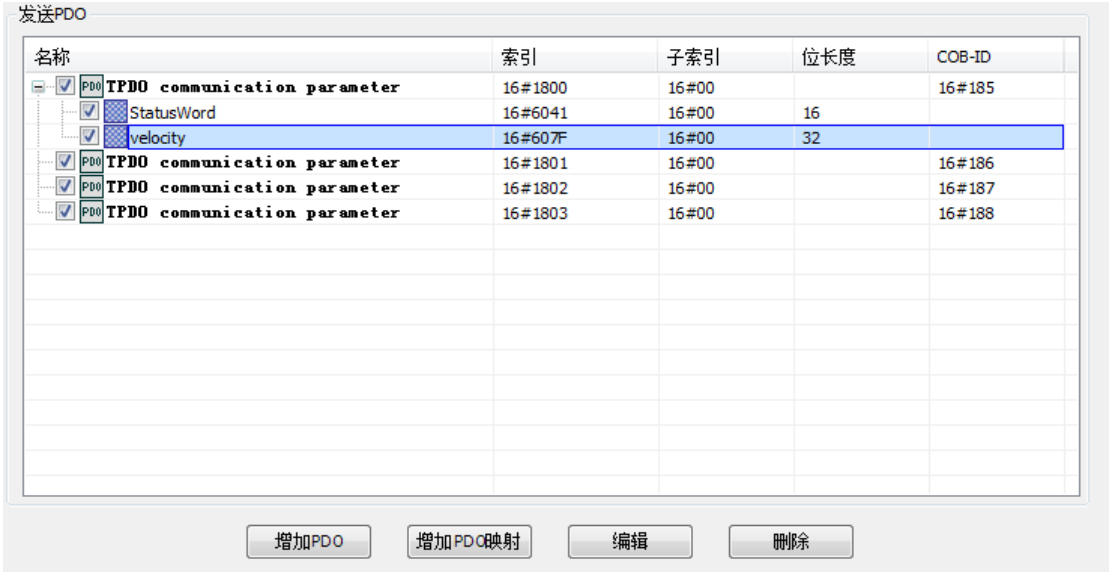


图 3-62 发送 PDO 界面

- 增加 PDO：增加一个 PDO，增加的 PDO 添加到最后面。从站中最大发送 PDO 个数由对象字典索引 1800-1A00 个数决定的，超过最大个数不能添加。添加的 PDO 名称和索引从对象字典根据使用顺序自动获取，不能修改。

  添加 PDO 会弹出一个 PDO 属性设置对话框，详见 PDO 属性。

- 增加 PDO 映射：在当前所选的 PDO 中增加一个 PDO 映射，此 PDO 映射增加到当前 PDO 的后面。PDO 映射最大 64 位，超过 64 位不能添加。PDO 映射来自于对象字典，发送 PDO 可映射的对象来自于对象字典中可读访问权限的对象，如访问权限为 RW、RWR、RO、CONST。

  添加 PDO 映射弹出添加对象对话框，详见添加对象。

- 编辑：编辑当前所选的 PDO 通信参数或者 PDO 映射参数。如果所选为 PDO，编辑 PDO 通信参数，如果所选为 PDO 映射，则编辑 PDO 映射。如果是 AM600 从站，只能编辑通信参数。

- 删除：删除当前所选的 PDO 或者 PDO 映射。如果所选为 PDO，则删除 PDO，如果所选为 PDO 映射，则编辑 PDO 映射。如果是 AM600 从站，不能执行删除操作。

4 服务数据对象

服务数据对象（SDO）可以在从站初始化过程中进行数据传输，也可以在从站运行过程中进行数据传输，本页面所有配置用于在从站初始化时写入从站。

服务数据对象配置界面可配置选中的 SDO，修改 SDO 的传输顺序，并定义 SDO 在传输过程中出现错误时的处理方式。

行号	索引:子索引	名称	值	位长度	有错则退出	有错则跳转到行	下一行
32	16#1800:16#01	Disable PDO	16#80000181	32	☐	☐	0
33	16#1801:16#01	Disable PDO	16#80000281	32	☐	☐	0
34	16#1802:16#01	Disable PDO	16#80000381	32	☐	☐	0
35	16#1803:16#01	Disable PDO	16#80000481	32	☐	☐	0
36	16#1804:16#01	Disable PDO	16#80000000	32	☐	☐	0
37	16#1805:16#01	Disable PDO	16#80000000	32	☐	☐	0
38	16#1806:16#01	Disable PDO	16#80000000	32	☐	☐	0
39	16#1807:16#01	Disable PDO	16#80000000	32	☐	☐	0
40	16#1808:16#01	Disable PDO	16#80000000	32	☐	☐	0
41	16#1809:16#01	Disable PDO	16#80000000	32	☐	☐	0
42	16#180A:16#01	Disable PDO	16#80000000	32	☐	☐	0
43	16#180B:16#01	Disable PDO	16#80000000	32	☐	☐	0
44	16#180C:16#01	Disable PDO	16#80000000	32	☐	☐	0
45	16#180D:16#01	Disable PDO	16#80000000	32	☐	☐	0
46	16#180E:16#01	Disable PDO	16#80000000	32	☐	☐	0
47	16#180F:16#01	Disable PDO	16#80000000	32	☐	☐	0
48	16#6411:16#07	Analog_out_CH 7	0	16	☐	☐	0
49	16#6320:16#02	Digital32_out 2	0	32	☐	☐	0
50	16#1010:16#03	Save Application Parameters	0	32	☐	☐	0

图 3-63 服务数据对象列表界面

服务数据对象列表：此列表显示从站初始化时写入的所有 SDO，其中灰色的 SDO 为系统自动添加的 SDO，位于列表的最上面，最先配置，此 SDO 为从站配置界面参数自动产生，如心跳、节点守护、应急信息、PDO 及 PDO 映射等配置自动产生的。也可以通过下面的“增加”按钮自定义添加 SDO。自定义添加的 SDO 可以修改和上下移动。

可以双击自定义添加的 SDO“值”单元格，修改 SDO 值。

在配置 SDO 出现错误时可以设置出错处理方式。可以激活“有错则退出”，当配置此 SDO 出现错误时直接退出配置，下面的 SDO 不再配置；也可以激活“有错则跳转到行”，当配置此 SDO 出现错误时，直接跳转到指定的行开始向下执行。如果没有激活这两个选项，则按默认处理方式，直接执行下一个。

NOTE

1) 只有在专家模式系统 SDO 和错误处理才能显示。
2) SDO 配置错误处理时，需要警惕跳转行处理，如果向上跳转时可能出现死循环配置 SDO。

- 向上：把选择的 SDO 上移一行，只有自定义添加的 SDO 才可以上移。
- 向下：把选择的 SDO 下移一行，只有自定义添加的 SDO 才可以下移
- 增加：在所选择的 SDO 上方增加一个 SDO。此按钮会弹出“添加对象”对话框。虽然“添加对象”界面和添加 PDO 界面相似，但添加 SDO 对象界面会显示 SDO 值编辑框和注释，可以在此编辑框中编辑 SDO 值，修改 SDO 注释。
- 编辑：编辑所选择的 SDO，此按钮会弹出“添加对象”对话框，通过此对话框修改 SDO 信息。只有自定义添加的 SDO 才可以编辑。
- 删除：删除所选择的 SDO，只有自定义添加的 SDO 才可以删除。
- SDO 超时：设置配置一条 SDO 的超时时间，默认 1000ms，范围 0ms-4294967ms。

5 调试

调试界面用于从站 NMT 控制、SDO 读写和诊断信息获取。

图 3-64 调试页面

NMT

NMT 用于提供网络管理（如初始化、启动和停止节点，侦测失效节点）服务。这种服务是采用主从通讯模式（所以只有一个 NMT 主节点，主站）来实现。

从站在启动过程中其状态可以用下面的节点状态转换图表示。

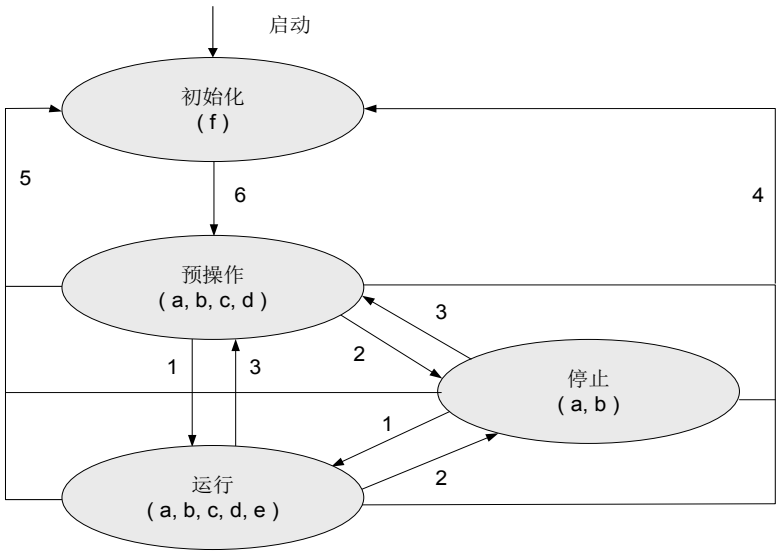


图 3-65 从站节点转换图

说明：

a. NMT， b. Node Guard， c. SDO， d. Emergency， e. PDO， f. Boot-up

状态转移（1 — 5 由 NMT 服务发起）序列，NMT 命令字（在括号中）：

1：Start_Remote_node (0x01，启动节点)

2：Stop_Remote_Node (0x02，停止节点)

3：Enter_Pre-Operational_State (0x80，进入试运行)

4：Reset_Node (0x81，复位节点)

5：Reset_Communication (0x82，复位通信)

6：设备初始化结束，自动进入 Pre_Operational 状态，发送 Boot-up 消息

初始化包括应用数据初始化和通信初始化，复位节点复位从站节点所有数据，而复位通信只复位通信数据。

在任何时候 NMT 服务都可使所有或者部分节点进入不同的工作状态。

- 启动节点：运行从站节点，在运行状态下才能进行 PDO 通信。当从站处于预运行状态或者停止状态时，启动节点会把从站置于运行状态（图 66 状态 1）。
- 停止节点：停止从站节点运行，在此状态下会停止除节点守护和心跳之外的所有通信。当从站处于预运行状态或者运行状态时，停止节点会把从站置于停止状态（图 66 状态 2）。
- 进入试运行：从站进入试运行状态，在此状态下可以进行 SDO 通信不能进行 PDO 通信。从站初始化完成后会自动进入试运行状态。当从站处于运行或者停止状态时，进入试运行会把从站置于试运行状（图 66 状态 3）。
- 复位节点：重置从站配置数据。首先复位应用配置，然后复位通信配置数据，复位完成后自动进入预运行状态（图 66 状态 4）。
- 复位通信: 重置从站通信配置数据。只复位通信配置数据，复位完成后自动进入预运行状态 (图 66 状态 5)。

服务数据对象（SDO）

SDO 用来在设备之间传输大的低优先级数据，典型的是用来配置 CANopen 网络上的设备，本页面主要用于从站节点运行过程中，读取或者写入 SDO 对象值。读取或者写入一个 SDO 对象需要确定 SDO 对象的索引、子索引、位长度，写入时还需要要写入的值。

- 索引：SDO 读 / 写索引，范围 16#0-16#FFFF。
- 子索引：SDO 读 / 写子索引，范围 16#0-16#FF。
- 位长度：SDO 读 / 写位长度，范围 8,16,24,32。
- 数据：SDO 读 / 写的值，前一个编辑框为 16 进制值，“=”后为 10 进制值。在写入 SDO 时，值范围和位长度有关，最小值 0，最大值为位长度所能表示的最大值。
- 结果：SDO 读 / 写结果，如果读 / 写异常，为异常信息。

诊断

显示从站节点的运行状态和应急信息。

- 在线状态：显示从站在线状态，从站在线显示绿色背景，内容为“在线”，从站离线显示红色背景，内容为“离线”。
- 运行状态: 显示从站运行状态，前面图标显示从站状态，图标后为状态文字。状态对应的图标和文字如下表:

状态	图标	文字信息
运行		运行
停止		停止
试运行		预操作
初始化		已初始化
未连接		断开连接
正在连接		正在连接 ...

状态	图标	文字信息
正在准备		正在准备 ...
复位节点		复位节点 ...
复位通信		复位通信 ...
扫描节点		扫描从站 ...
配置节点		配置从站 ...
启动节点		启动从站 ...
未知状态		未知状态

- 诊断字符串：显示从站当前诊断信息。
- 最近的应急信息 : 显示当前未确认的第一条应急信息。当由应急信息过来时，如果上一条没有确认，此应急信息不显示，直到上一条确认后，才显示当前应急信息。

一个应急报文由 8 字节组成，格式如下：

sender receiver(s) COB-ID	Byte 0-1	Byte 2	Byte 3-7
0x080+Node_ID	应急错误代码	错误寄存器 (对象 0x1001)	制造商特定的错误区域

16 进制的应急错误代码如下表所示。应急错误代码中‘xx’部分由相应的设备子协议定义。

应急错误代码（16 进制）	代码功能描述
00xx	Error Reset 或 No Error
10xx	Generic Error
20xx	Current
21xx	Current，device input side
22xx	Current，inside the device
23xx	Current，device output side
30xx	Voltage
31xx	Mains voltage
32xx	Voltage inside the device
33xx	Output voltage
40xx	Temperature
41xx	Ambient temperature
42xx	Device tempearture
50xx	Device hardware
60xx	Device software
61xx	Internal software
62xx	User software
63xx	Data set
70xx	Additional modules
80xx	Monitoring
81xx	communication
8110	CAN overrun
8120	Error Passive
8130	Life Guard Error 或 Heartbeat Error
8140	Recovered from Bus-Off
82xx	Protocol Error
8210	PDO no processed Due to length error
8220	Length exceedd
90xx	External error
F0xx	Additional functions
FFxx	Device specific

应急信息包含时间、错误码、错误寄存器和制造商指定代码。

- 时间：软件获取应急信息的时间，不是故障发生时间。
- 错误码：应急错误码，鼠标悬浮上，会提示应急错误码对应的应急信息。
- 错误寄存器：应急信息错误寄存器。定义如下表：

位错误寄存器位定义 Bit	错误类型
0	Generic
1	Current
2	Voltage
3	Temperature
4	Communication
5	Device profile specific
6	Reserved(=0)
7	Manufacturer

- 制造商指定代码：应急信息制造商指定代码
- 确认：确认应急信息。应急信息只保留一条，只有确认后，后来的应急信息才接收显示。

6 PDO 属性

PDO 属性用于设置 PDO 通信的通信参数，包括 COB-ID（通信对象标示符）、通信传输方式、抑制时间和事件时间。在编辑或者添加 PDO 时弹出此对话框。

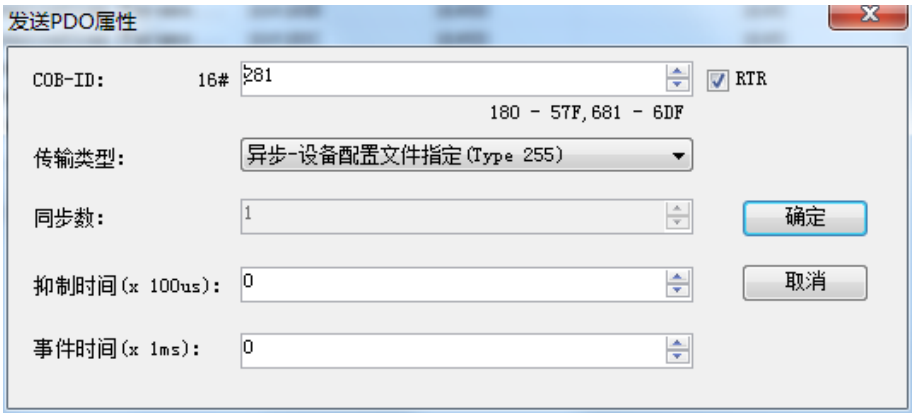


图 3-66 PDO 属性对话框

- COB-ID：PDO 通信对象标示符。一个 CANopen 总线内是唯一的，不能和其它通信对象标示符重复（如其它 PDO COB-ID、应急 COB-ID、心跳 COB-ID）。PDO COB-ID 范围为 16#180-57F，681-6DF，如果不合法可以手动修改或者通过主站检查并修正配置修正。每个 PDO 默认 COB-ID 来自于对象字典对应 PDO 子索引 01 的对象，如果对象字典格式为 $NodeID+ 值，则此 COB-ID 随着从站节点 ID 的更改自动更改，当手动更改此 COB-ID 后，自动随节点 ID 更改特性消失。如果 COB-ID 对应的对象字典访问权限无写权限，则此 COB-ID 不能更改。
- RTR：接收远程帧，接收到远程帧后触发 PDO 发送，只有发送 PDO 才显示。
- 传输类型：PDO 通信传输方式。

PDO 可以有多种传送方式：

1） 同步（通过接收 SYNC 对象实现同步）

非周期：由远程帧触发传送，或者由设备子协议中规定的对象特定事件触发传送。

周期：传送在每 1 到 240 个 SYNC 消息后触发。

2） 异步

由远程帧触发传送； 由设备子协议中规定的对象特定事件触发传送。

下表给出来了由传输类型定义的不同 PDO 传输模式，传输类型为 PDO 通讯参数对象的一部分，由 8 位无符号整数定义。

传输类型	触发 PDO 通信条件 (B = both needed O = one or both)			PDO 传输
	SYNC	RTR	Event	
0	B	-	B	同步，非循环
1-240	O	-	-	同步，循环
241-251	-	-	-	Reserved
252	B	B	-	同步，在 RTR 之后
253	-	O	-	异步，在 RTR 之后
254	-	O	O	异步，制造商特定事件
255	-	O	O	异步，设备子协议特定事件
说明： SYNC – 接收到 SYNC-object。 RTR －接收到远程帧。 Event – 例如数值改变或者定时器中断。 传输类型为：1 到 240 时，该数字代表两个 PDO 之间的 SYNC 对象的数目）。				

每个 PDO 默认传输类型来自于对象字典对应 PDO 子索引 02 的对象，如果其对应的对象字典访问权限无写权限，则传输类型不能更改。

- 同步数：和传输类型有关，只有传输类型选择 1-240 时可以修改，表示从站接收到同步数个同步帧后开始处理 PDO 数据传输。
- 抑制时间 : 同一个 PDO 传输两条信息之间的最小间隔时间与 100μs 的乘积，该值域才可修改。该设置可避免当一个值改变的时候 PDO 发送太频繁。默认值为 0，范围 0-65535。只有发送 PDO 并且传输方式为 254 或者 255 才能设置。每个 PDO 默认抑制时间来自于对象字典对应 PDO 子索引 03 的对象。如果抑制时间对应的对象字典访问权限无写权限，则此抑制时间不能更改。
- 事件时间 : 同一个 PDO 传输两条信息之间的间隔时间，单位 ms，默认值为 0，范围 0-65535。只有发送 PDO 并且传输方式为 254 或者 255 才能设置。每个 PDO 默认事件时间来自于对象字典对应 PDO 子索引 05 的对象。如果事件时间对应的对象字典访问权限无写权限，则此事件时间不能更改。

7 添加对象

添加对象对话框用于添加、编辑接收 PDO 映射、发送 SDO 映射或者 SDO。在对 SDO 操作时，此对话框会增加 SDO 值编辑框和 SDO 注释输入框。

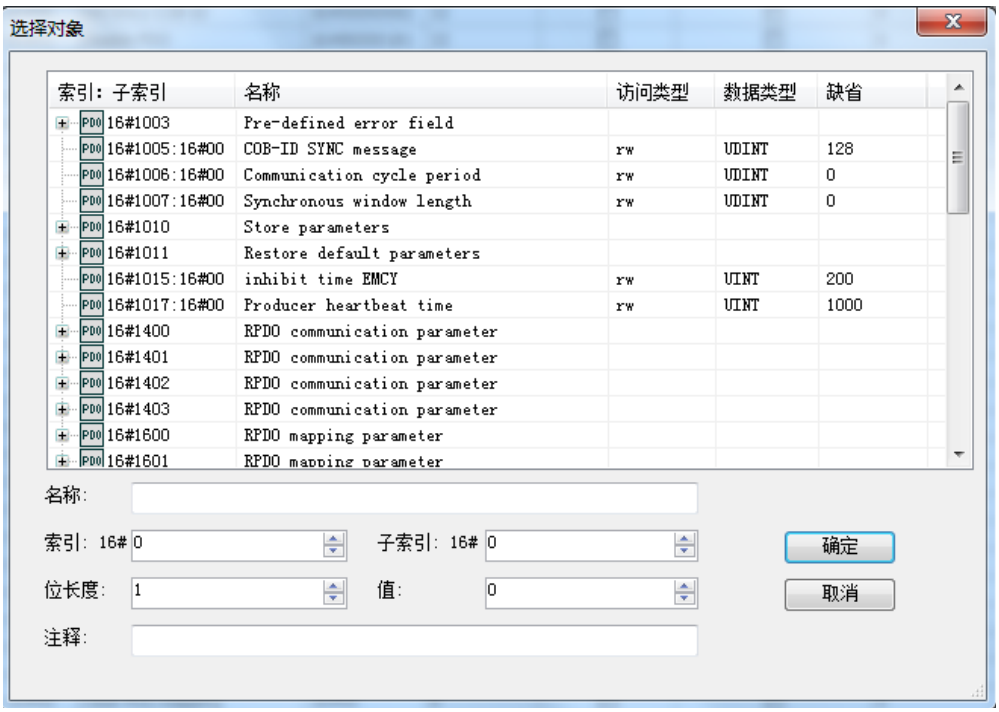


图 3-67 添加对象对话框

对象列表：此列表来自于 EDS 文件中的对象。在接收 PDO 映射编辑时，只有访问权限为 RW、RWW、WO 并且索引大于 16#2000 的对象才能显示；在发送 PDO 映射编辑时，只有访问权限为 RW、RWR、RO、CONST 并且索引大于 16#2000 的对象才能显示；在 SDO 编辑时，只有访问权限为 RW、RWW、RWR、WO 的对象才能显示。

对于 AM600 从站 SDO 编辑时，索引为 16#2000-16#40df 之间的对象不能显示（这些参数用于模块配置，而模块配置参数在模块本身已经配置）。

- 索引：对象索引，范围 16#0-16#FFFF，当选中对象列表对象时，自动显示选中对象索引。
- 子索引：对象子索引，范围 16#0-16#FF，当选中对象列表对象时，自动显示选中对象子索引。
- 位长度：对象位长度，范围 0-32，当选中对象列表对象时，自动显示选中对象位长度。
- 值：要配置的 SDO 值，只有编辑 SDO 时才显示，范围和选中对象数据类型有关，当选中对象列表对象时，自动显示选中对象值。
- 注释：SDO 注释，只有编辑 SDO 时才显示，最大 50 个字符。

8 CANopen 从站 I/O 映射

此页面只有在非 AM600 从站时显示，AM600 从站对应的 I/O 映射和 AM600 I/O 模块对应，不再此显示。I/O 映射的总体描述和本对话框的使用请参照 I/O 映射链接。

9 状态

CAN 总线设备或模块的状态配置编辑器可显示状态信息（如：“运行”，“停止”）和内部总线系统的状态。

10 信息

如果当前设备可用，将会显示下列基本信息：名称、供应商、类型、版本号，模块号、描述、图像。

3.7.4 CANopen 模块

1 模块化设备和非模块化设备

在 CANopen 从站配置 中，CANopen 从站节点可接入两种类型的模块设备：

模块化设备：接入 CANopen 从站节点下，模块自带 I/O 映射列表，不需要使用 CANopen 从站 I/O 映射对话框，从站节点 PDO 映射随着模块添加自动添加。目前 AM600 I/O 模块为此类型。

非模块化设备：从站节点对话框包含 I/O 映射对话框，PDO 映射不能自动配置。

2 AM600 CANopen I/O 模块

AM600 CANopen I/O 模块在硬件组态中添加，添加一个 I/O，自动添加一个 PDO 映射，和 PDO 映射关系见接收 PDO 和发送 PDO。添加 I/O 后，可以配置 I/O 参数，也可以给 I/O 添加 I/O 映射，实现数据刷新，具体见 CPU 下 I/O 模块。

3.7.5 编程接口

CiA-405 库，请参见本手册“6.3.1 CiA-405 库”介绍。

3.8 CANlink 3.0 配置编辑器

CANlink 协议是汇川技术股份有限公司基于 CAN2.0 总线协议制定的 CAN 实时总线应用层协议。主要用于汇川技术产品 PLC，变频器，伺服控制器和远程扩展模块等产品之间进行高速实时数据交互。在使用 AM600 系列 PLC 的 CANlink 功能前，请仔细阅读本章节内容。

CANlink3.0 采用主 / 从模式，在一个网络中必须具备并只有一个主站，从站数量为 1~62 个，所有主从站点号范围为 1~63，且站号必须唯一。

1） 支持心跳监控主 / 从站运行状态；

2） 支持总线占有率预警和实时总线占有率监控；

3） 支持掉线重连功能；

4） 支持热接入方式；

5） 主站支持发送配置（包括时间触发、事件触发、同步触发）发送数据共 256 条；

6） 单个从站支持发送配置（包括时间触发、事件触发、同步触发）16 条，从站总计最多支持 256 条配置；

7） 每个站点支持接收其它 8 个站点发送的点对多数据；

8） 支持主 / 从式数据交互和从 / 从式数据交互；

9） 主站支持同步写最多 128 条。

3.8.1 CANlink3.0 网络组成

1 硬件端口

有关 CANlink 通信端口定义请参见第 147 页上的“2 硬件端口”。

2 通信距离

一个 CANlink3.0 网络由一台主站、以及若干从站组成，最多支持的从站数目为 62 个（与波特率有关）。

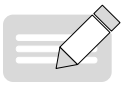
波特率	最大通信距离	通信电缆线径	可接入站点数
1000Kbps	20m	⩾ 0.3 mm^2	18
500 Kbps	80m	⩾ 0.3 mm^2	32
250 Kbps	150m	⩾ 0.3 mm^2	63
125 Kbps	300m	⩾ 0.5 mm^2	63
100 Kbps	500m	⩾ 0.5 mm^2	63
50 Kbps	1000m	⩾ 0.7 mm^2	63

以上数据是在使用标准屏蔽双绞线前提下，可接入站点数是当前波特率下网络中允许的最大节点数（主站和从站总数）。

3 支持 CANlink3.0 的设备

在一个 CANlink3.0 网络中，必须有且只有一个主站，这个主站可能是 AM400、AM600 或 AC800 等系列 PLC。在该网络中，可以存在 1~62 个从站，这些从站包括 AM400、AM600 或 AC800（查看 D8280，为 300 表示支持）、远程扩展模块 (51210 或 52210 以上版本)、214 非标 IS500 伺服（功能码 H00-02=214.xx）、IS620P（H01-00=6.0 或以上）、IS700（H01-00=301.05）、MD310（F7-11=u37.18 或以上）、MD380（F7-11=4.71.06 或以上）等。部分产品需配置专用的 CANlink 通讯扩展卡才能使用 CANlink 功能，具体请参照各产品的用户手册。

4 AM600 支持 CANlink3.0 的特殊元件说明


NOTE

AM600 的 CANLINK 功能使用的软元件名称是 SD 和 SM，与小型 PLC 的 D 和 M 元件类似，但它们之间没有对应关系。

SD 区间分配	功能	SM 区间分配	功能
0 - 7000	用户通用的字软元件区（CANLINK 配置表中能使用到的范围）	0 - 3071	用户通用的位软元件区（CANLINK 配置表中能使用到的范围）
7000-7999	用户通用的字软元件区	3072-7999	用户通用的位软元件区
8000 - 8999	系统定义的特殊寄存器元件区	8000 - 8999	系统定义的特殊位元件区：目前只有 CANLINK 使用
9000 - 9999	系统定义的特殊寄存器元件区：目前只有 高速 I/O 使用	9000 - 9999	系统定义的特殊寄存器元件区：目前只有 高速 I/O 使用

AM600 CANLINK 功能涉及的特殊软元件定义如下（具体定义请参见 CANLINK3.0 标准）：

特殊软元件	属性
SD8100 - SD8163	SD8100 本机节点的当前状态；SD81xx 表示站点号为 xx 的节点的当前状态，例如 SD8101 代表站点号为 1 的节点状态。寄存器值含义： 0：未配置；1：有配置；2：在线；5：离线
SD8164 - SD8239	保留
SD8240	总线负载率 （后台监控用）
SD8241	CAN3.0 从站状态
SD8242	CAN3.0 从站状态
SD8243	CAN3.0 从站状态
SD8244	CAN3.0 从站状态
SD8245	接收帧数
SD8246	接收错误计数
SD8247	发送错误计数
SD8287	厂商代码
SD8288	产品系列
SD8289	产品型号
SD8290	固件版本
SD8307	CAN3.0 报错（命令异常码）
SD8308	CAN3.0 报错 (配置异常码)

3.8.2 CANlink 一般使用流程

CANlink 一般使用流程如下：

1） 设计 Canlink 硬件网络结构。

2） 在网络组态中激活 CANlink 总线。AM600 CPU 既可以作为 CANlink 主站也可以作为 CANlink 本地从站。激活总线后，会自动添加总线设备。

3） 如果 AM600 CPU 作为 CANlink 主站，可以在 CANlink 网络配置向导中，配置主站参数和添加、删除从站（远程从站，后面单独说从站指远程从站）。如果 AM600 CPU 作为 CANlink 本地从站，也可以配置其参数。

4） 设置合适的发送配置参数、接收配置参数及同步配置参数。

5） 在程序中通过操作软元件控制 CANlink 传输数据。

6） 在网络管理页面控制主从站启停、监控主从站运行状态。

3.8.3 CANlink 网络配置

在配置 CANlink 网络之前，需要在网络组态激活 CANlink 总线。如果激活 CANlink 主站，在设备树中添加 CANlink 主站节点，第一个配置 CANlink 网络时，双击此节点显示网络配置向导；如果激活 CANlink 从站，在设备树中添加 CANlink 从站节点，双击此节点显示本地从站配置界面。

1 CANlink3.0 网络配置向导

第一次配置 CANLink 总线或者在网络管理界面，单击“站点管理”，弹出网络配置向导。网络配置向导配置主站参数和从站参数。

2 主站配置

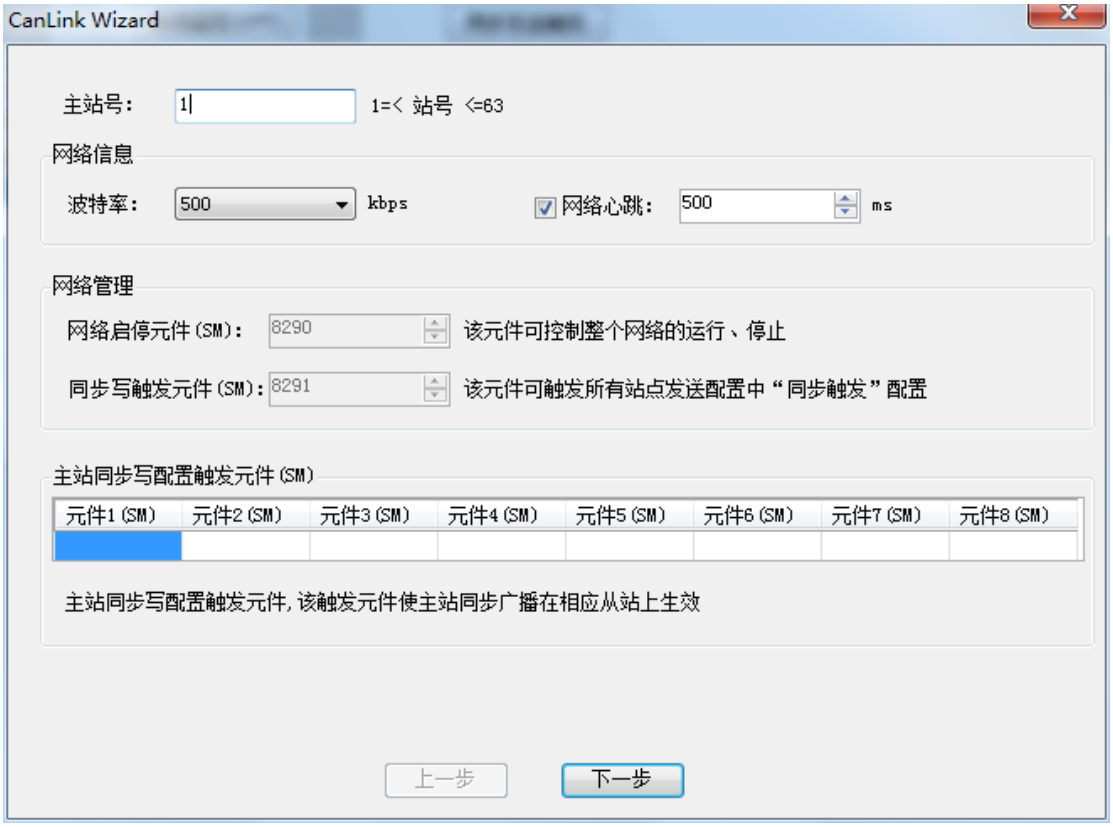


图 3-68 主站参数配置界面

- 主站号：主站号作为网络中主站的标示，表明该站号的站点为主站，由主站对整个 CANlink3.0 网络进行管理和监控。注意，主站号必须与实际下载配置的 PLC 一致。

网络信息

- 波特率：选择网络使用的波特率，要求所有站点的波特率均与此处选择的一致，即使是没有进行配置，但线路接入网络中的站点，其波特率也需一致。
- 网络心跳：网络心跳设置范围为 10~20000ms，主从站按设定值定时向网络中发送心跳，主从站通过心跳监控各自通信情况，当出现通信异常时告警并作出相应的处理。网络心跳设定值越小，监控越灵敏，但同时对网络占用率也更高，当心跳对网络占用率超过 10% 时将自动告警。

如果将网络心跳前的勾去掉，则网络心跳功能将取消，将无法对网络进行监控。

网络管理

- 网络启停元件：控制 CANlink 网络启动和停止，使用软元件 SM8290，当 SM8290 值为 TRUE 时表示启动 CANlink，否则停止 CANlink。
- 同步写触发元件：控制所有站点发送配置的同步写功能，使用软元件 SM8291，详见同步触发配置。

主站同步写配置触发元件

主站同步写触发元件用于主站同步配置。最多可以设置 8 个同步触发元件。当网络中不需主站同步配置时，此处可不填写，具体参见“主站同步写”。

3 增加和删除从站

此界面用于添加、修改和删除 CANlink 从站。

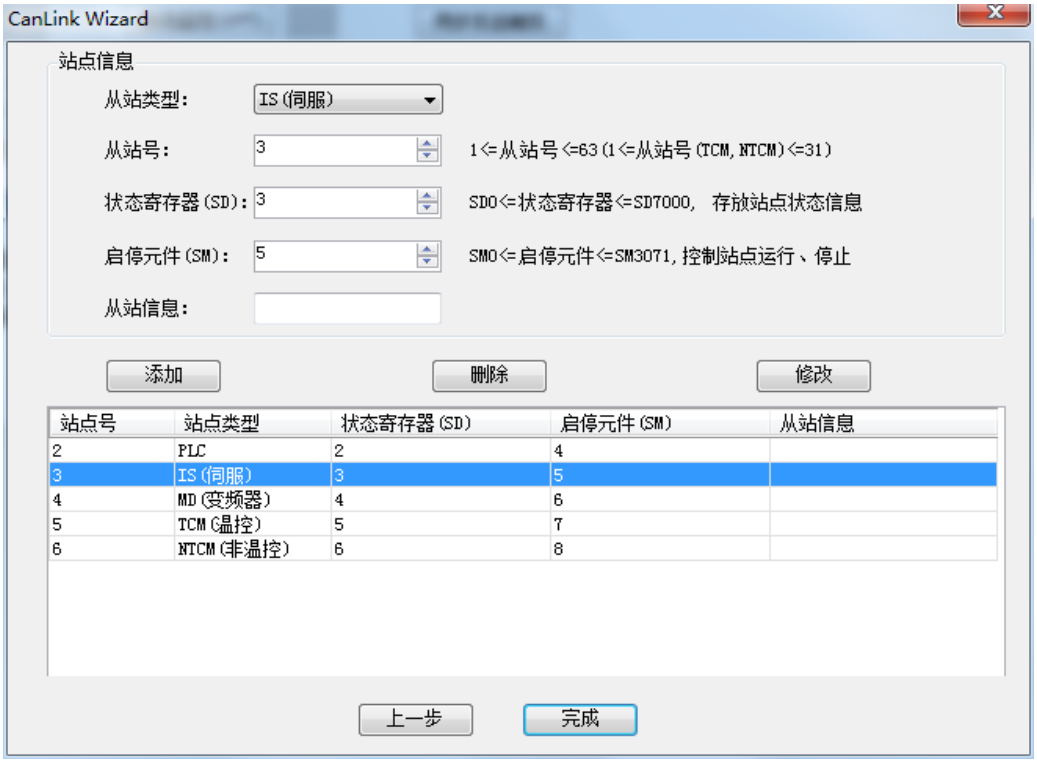


图 3-69 从站添加和删除界面

站点信息

- 从站类型：支持的从站类型，当前支持 PLC、伺服、变频器、温控模块、非温控模块从站。
- 从站号：作为网络中从站的标示，不能和主站或者其它从站号重复。
- 状态寄存器：表示相应从站运行状态的 SD 元件，这些元件不能和其它从站状态寄存器元件重复，这些状态主要包括从站的运行、是否有故障（不包括掉线）等。
- 启停元件：启停元件是主站控制从站启停的 SM 元件，网络运行时可通过改变这些 SM 元件来控制各从站通信的启停，这些元件不能和其它从站启停元件、同步写触发元件重复。
- 从站信息：表示从站信息，给从站增加注释信息，最大 32 字符。
- 添加：添加站点信息对应的从站到从站列表，添加从站时会检查状态寄存器、启停元件的唯一性。
- 删除：删除从站列表所选从站。
- 修改：修改从站列表所选从站点信息，这些信息从站点信息设置中获取。修改从站时会检查状态寄存器、启停元件的唯一性。

完成所有从站的设定后，点击“完成”即进入网络管理界面。

3.8.4 网络管理

网络管理界面可以管理网络的启停、同步发送触发、启停监控、启停从站，也可以进入站点配置向导，同样可以修改站点网络信息，另外可以清空所有配置。

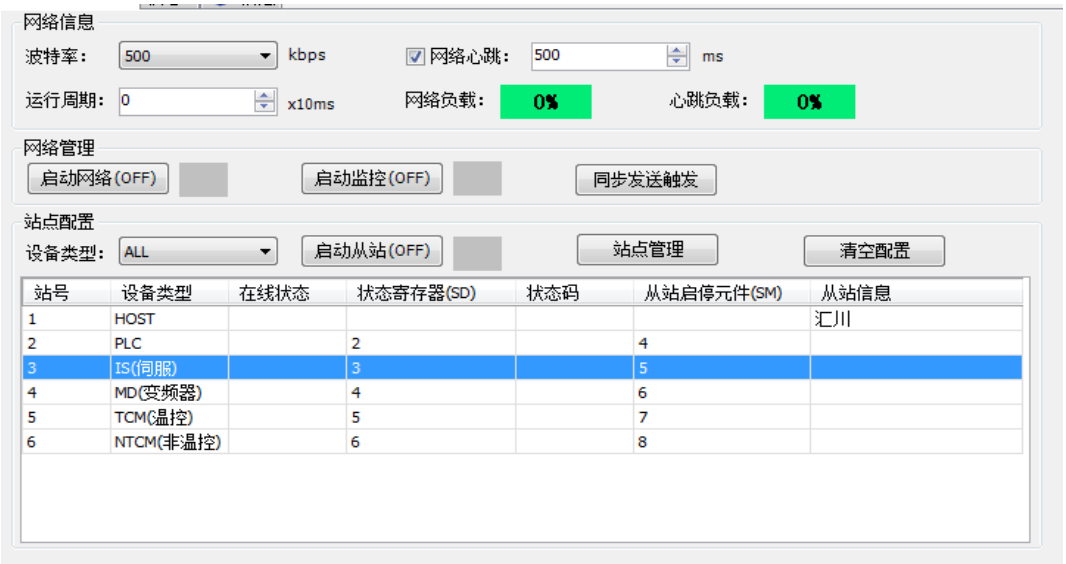


图 3-70 网络管理界面

网络信息

- 波特率：选择网络使用的波特率，要求所有站点的波特率均与此处选择的一致，即使是没有进行配置，但线路接入网络中的站点，其波特率也需一致。
- 网络心跳：网络心跳设置范围为 10~20000ms，主从站按设定值定时向网络中发送心跳，主从站通过心跳监控各自通信情况，当出现通信异常时告警并作出相应的处理。网络心跳设定值越小，监控越灵敏，但同时对网络占用率也更高，当心跳对网络占用率超过 10% 时将自动告警。

NOTE

如果将网络心跳前的勾去掉，则网络心跳功能将取消，将无法对网络进行监控。

- 运行周期：用于预估 CANlink 负载率。
- 网络负载：CANlink 网络负载，包括 CANlink 接收、发送、心跳所有负载，监控条件下获取 CANLink 总线实时负载（获取 SD8240 寄存器值），非监控条件下预估网络负载。
- 网络负载背景颜色：

颜色	范围（%）	负载文字
绿色	0-50	负载百分比，如 10%
黄色	51-75	负载百分比，如 55%
红色	76-90	负载百分比，如 78%
红色	大于 90	ERR

- 心跳负载：CANlink 心跳负载，心跳负载通过预估计算得到（登陆后获取的也是预估值）。
- 心跳负载背景颜色：

颜色	范围（%）	负载文字
绿色	0-10	负载百分比，如 10%
红色	大于 10	ERR

网络管理

网络管理功能必须在登陆 PLC 后才能操作。

- 启动网络：启停 CANlink 网络，通过控制网络启停元件 SM8290 来启动或者停止网络。
- 启动监控：开始 CANlink 网络监控，定时获取 CANlink 主从站运行状态。在站点列表中会显示站点在线状态更新。
- 同步发送触发：触发一次同步写功能。

站点配置

- 设备类型：选择站点列表要显示的设备类型。
- 启动从站：把从站置于运行状态，通过更改从站启停元件的值为 TRUE，来使从站运行。
- 站点管理：弹出网络配置向导对话框，配置 CANLink 网络。
- 清空配置：清空 CANlink 所有配置信息，把主站配置信息置为默认值。
- 站点列表：显示 CANlink 站点信息和站点状态。包含站号、设备类型、在线状态、状态寄存器、状态码、从站启停元件和从站信息。双击列表每行对应的从站，弹出站点发送配置、接收配置和主站同步写界面。
- 站号：站点唯一标示符。
- 设备类型：站点设备类型，包括 PLC、伺服、变频器、温控模块、非温控模块。
- 在线状态：登陆 PLC 后，点击“启动监控”，显示从站状态。从站在线状态通过 SD8240、SD8241、SD8242、SD8243 在线状态寄存器获取从站是否在线和离线，如果从站在线、通过获取从站启停元件值，判断从站是否运行或者停止。在线状态寄存器为只读，不允许对其进行修改。在线状态寄存器每位和站号对应关系如下表。

软元件	位	从站号
SD8243	Bit15-bit0	32-47
SD8242	Bit15-bit0	48-63
SD8241	Bit14-bit0	1-16
SD8240	Bit15-bit0	17-31

- 状态寄存器：保存从站运行状态的运行状态寄存器。运行状态寄存器为只读，不允许对其进行修改。

运行状态寄存器定义见下表：

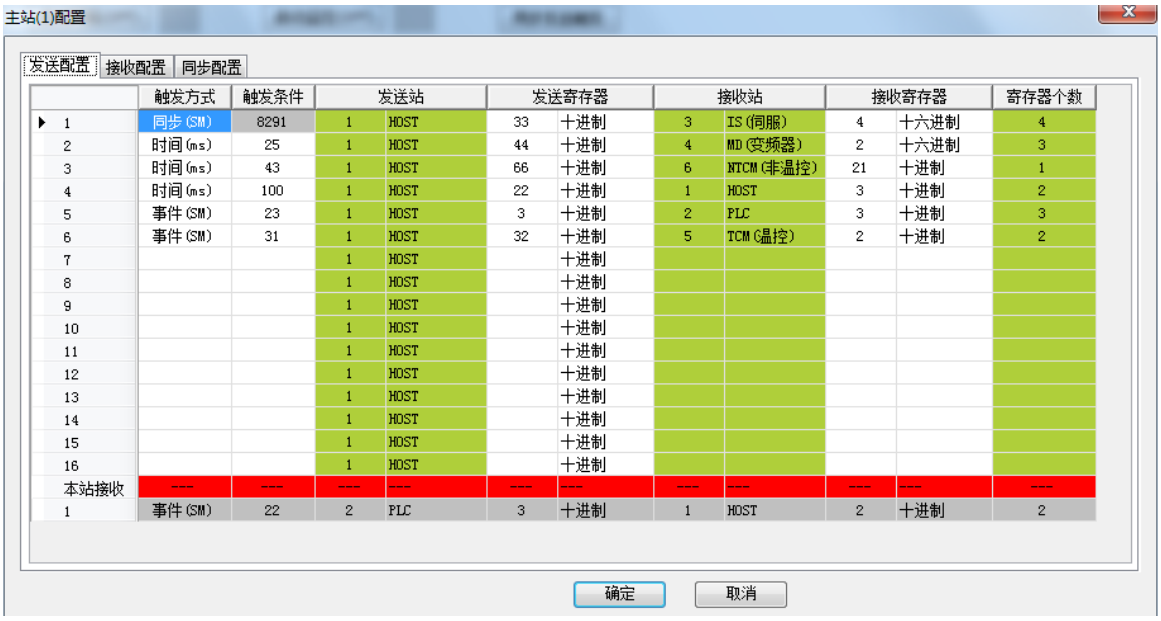
位域	说明
bit0	故障标示，“1”表示节点设备故障、“0”表示无故障
bit1	运行标示，“1”表示运行、“0”表示停机
bit2	设备就绪，“1”表示就绪、“0”表示未就绪 (伺服专用)
...	保留
bit15	保留

- 状态码：从站状态寄存器值，表示从站定义的状态码。
- 从站启停元件：控制从站的运行和停止。
- 从站信息：从站注释，可以定义从站说明信息。

3.8.5 发送配置

CANLink 发送配置分时间触发配置、事件触发配置、同步触发配置三种，其中事件触发配置又有 PLC 事件触发和非 PLC 事件触发两种。

1 发送配置编辑器



	触发方式	触发条件	发送站		发送寄存器		接收站		接收寄存器		寄存器个数
1	同步(SM)	8291	1	HOST	33	十进制	3	IS(伺服)	4	十六进制	4
2	时间(ms)	25	1	HOST	44	十进制	4	MD(变频器)	2	十六进制	3
3	时间(ms)	43	1	HOST	66	十进制	6	NTCM(非温控)	21	十进制	1
4	时间(ms)	100	1	HOST	22	十进制	1	HOST	3	十进制	2
5	事件(SM)	23	1	HOST	3	十进制	2	PLC	3	十进制	3
6	事件(SM)	31	1	HOST	32	十进制	5	TCM(温控)	2	十进制	2
7			1	HOST		十进制					
8			1	HOST		十进制					
9			1	HOST		十进制					
10			1	HOST		十进制					
11			1	HOST		十进制					
12			1	HOST		十进制					
13			1	HOST		十进制					
14			1	HOST		十进制					
15			1	HOST		十进制					
16			1	HOST		十进制					
本站接收	---	---	---	---	---	---	---	---	---	---	---
1	事件(SM)	22	2	PLC	3	十进制	1	HOST	2	十进制	2

图 3-71 发送配置编辑器

发送配备编辑界面用于配置站点发送设置。发送列表分两部分，上半部分显示本站发送给别的站点配置，后半部分显示，别的站点发送给本站点的配置。

对于主站能配置的发送条数最大为 256 条，对于从站最大发送条数为 16 条。

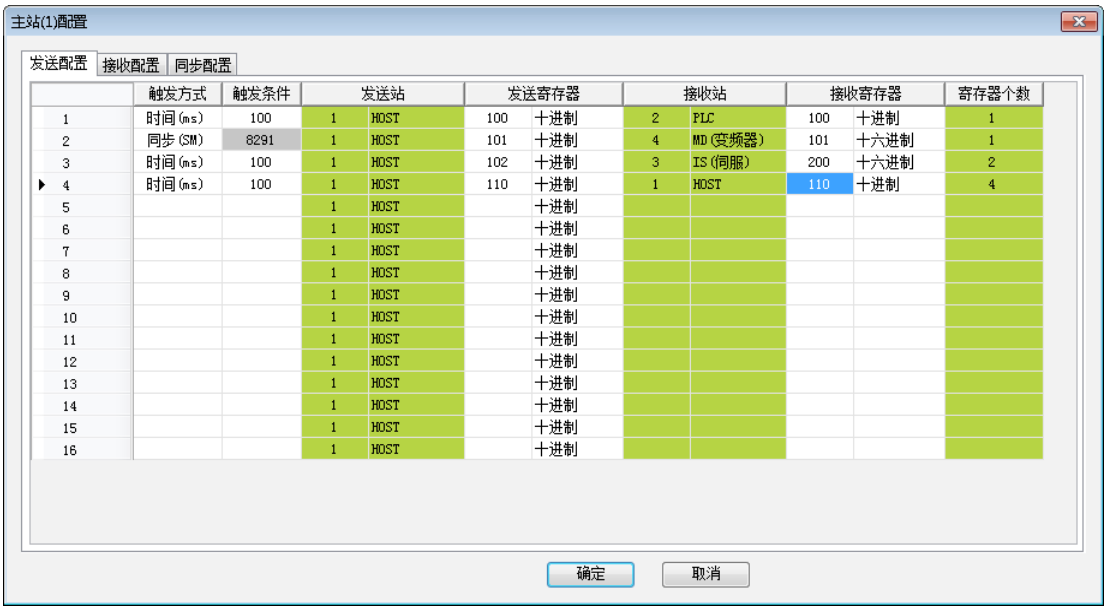
发送列表包含触发方式、触发条件、发送站、发送寄存器、接收站、接收寄存器和寄存器个数。

- 触发方式：站点发送数据触发类型，包括时间触发、事件触发和同步触发。
- 触发条件：站点发送数据触发条件，满足条件时开始发送配置数据。当触发方式为时间触发时，范围为 0-30000，站点按照触发条件值定时发送配置数据；当触发方式为事件触发时，如果站点为 Host 或者 PLC 类型，触发条件为 SM 软元件，范围为 0-3071，当触发条件为 TRUE 时站点发送配置数据，如果为伺服、变频器、温控模块、非温控模块时，范围为 0-30000，表示定时发送配置数据；当触发方式为同步触发时，当软元件 SM8291 值为 TRUE，执行发送动作。
- 发送站：发送配置对应的站点，不能更改。
- 发送寄存器: 发送配置站对应的寄存器，如果发送站为 PLC 类型，范围（此范围表示寄存器值加寄存器个数，本小节的范围都是此意义）为 0-7000；如果发送站为温控模块，范围为 0-499,700-722；如果发送站为非温控模块，范围为 0-63,700-722；如果发送站为伺服或者变频器，范围为 0-65535。
- 接收站：接收发送配置的站点，只能为已经添加的站点。
- 接收寄存器: 接收发送配置的站点中的寄存器。如果是点对多通信（发送站和接收站相同），范围为 0-65535; 如果接收站为 PLC 类型，范围为 0-7000；如果接收站为温控模块，范围为 0-499,721-722；如果接收站为非温控模块，范围为 0-63,721-722；如果接收寄存器个数站为伺服或者变频器，范围为 0-65535。
- 寄存器个数: 发送或者接收寄存器个数，范围 1-4，发送寄存器或者接收寄存器总长度不能超过范围限制值。

2 时间触发配置

时间触发配置是一种常用配置，发送站按配置的时间循环将源地址数据写入目标站的目标地址中（即相当于目标站循环读取发送站的源地址数据，并存放于目标地址中）。

若发送站和接收站相同，则该条是点对多配置，网络中所有配置接收该站点对多数据的站点均可接收该类数据（请参见“接收配置”）。



	触发方式	触发条件	发送站		发送寄存器		接收站		接收寄存器		寄存器个数
1	时间(ms)	100	1	HOST	100	十进制	2	PLC	100	十进制	1
2	同步(SM)	8291	1	HOST	101	十进制	4	MD(变频器)	101	十六进制	1
3	时间(ms)	100	1	HOST	102	十进制	3	IS(伺服)	200	十六进制	2
4	时间(ms)	100	1	HOST	110	十进制	1	HOST	110	十进制	4
5			1	HOST		十进制					
6			1	HOST		十进制					
7			1	HOST		十进制					
8			1	HOST		十进制					
9			1	HOST		十进制					
10			1	HOST		十进制					
11			1	HOST		十进制					
12			1	HOST		十进制					
13			1	HOST		十进制					
14			1	HOST		十进制					
15			1	HOST		十进制					
16			1	HOST		十进制					

图 3-72 时间触发配置界面

时间触发方式中，时间设定范围为 1~30000ms，循环时间设定数值越小，数据更新速度越快，网络负载也会越高。

如上图中，第 3 行配置表示 1 号站以 100ms 的循环周期将 SD102、SD103 两个寄存器的值写入到 3 号站伺服的 H0200、H0201 寄存器中。第 4 行配置中，发送站和接收站均为 1 号站，则表示该配置为点对多配置，1 号站以 100ms 的周期将 SD110-SD113 四个寄存器的值以点对多的形式发送到网络中，所有配置接收 1 号站点对多数据的站点都可以接受到该帧。

3 事件触发配置

事件触发是一种实时性的触发配置，当满足条件时即时发送，若条件不满足无论距离上次发送多久也不会触发。

根据产品的特点，PLC 的事件触发与其他产品的事件触发略有不同。

PLC 事件触发配置

当作为触发条件的 SM 被置 ON 时，触发相应的事件。触发事件的范围为 SM0~3071， 超出后会自动提示。

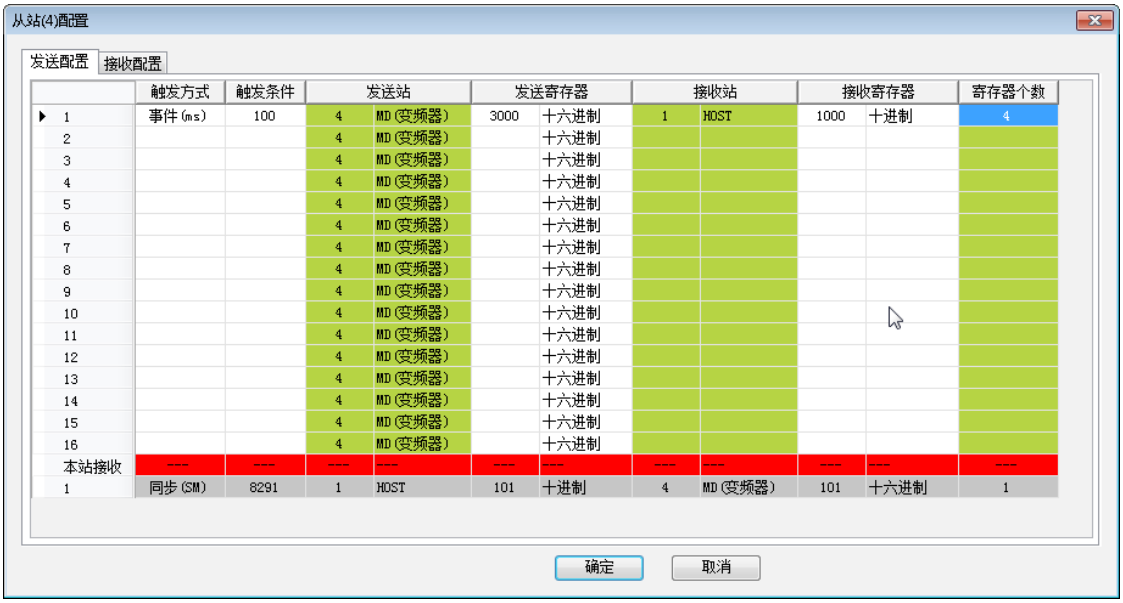
	触发方式	触发条件	发送站		发送寄存器		接收站		接收寄存器		寄存器个数
1	事件(SM)	100	1	HOST	100	十进制	2	PLC	100	十进制	1

图 3-73 PLC 事件触发配置

如上图中，当 SM100=1 时，1 号站将 SD100 的值发送到 2 号站 PLC 的 SD100 中，发送后 SM100 会自动复位。

非 PLC 设备事件触发配置

除了 PLC 外，变频器、伺服、扩展模块等的事件触发均是采取寄存器值改变，且距离上次发送满足最小间隔时间时触发的方式。



	触发方式	触发条件	发送站		发送寄存器		接收站		接收寄存器		寄存器个数
1	事件(ms)	100	4	MD(变频器)	3000	十六进制	1	HOST	1000	十进制	4
2			4	MD(变频器)		十六进制					
3			4	MD(变频器)		十六进制					
4			4	MD(变频器)		十六进制					
5			4	MD(变频器)		十六进制					
6			4	MD(变频器)		十六进制					
7			4	MD(变频器)		十六进制					
8			4	MD(变频器)		十六进制					
9			4	MD(变频器)		十六进制					
10			4	MD(变频器)		十六进制					
11			4	MD(变频器)		十六进制					
12			4	MD(变频器)		十六进制					
13			4	MD(变频器)		十六进制					
14			4	MD(变频器)		十六进制					
15			4	MD(变频器)		十六进制					
16			4	MD(变频器)		十六进制					
本站接收	---	---	---	---	---	---	---	---	---	---	---
1	同步(SM)	8291	1	HOST	101	十进制	4	MD(变频器)	101	十六进制	1

图 3-74 非 PLC 事件触发配置

如上图，4 号站变频器的 H3000 发生变化时，如果距离上次该配置发送的时间达到 100ms，则立即发送，如距离上次发送不足 100ms 则等待直到时间足够才发送，触发条件中的间隔时间越小，实时性越强，同时也对网络影响越大。

最小间隔时间的设置范围为 1~30000ms，超出范围将自动提示。

4 同步触发配置

同步触发配置是主站检测到同步发送触发元件 SM8291 被置位时，向网络中广播命令，要求网络中所有站点的同步触发配置（触发条件为 SM8291 的配置）全部依次触发的一种配置。主站发送命令帧后将自动复位 SM8291。如下图，1 号主站、2 号 PLC 从站与 4 号 MD380 从站均有同步触发配置，若主站 SM8291 被置位，这三个站的同步触发都将发送。

	触发方式	触发条件	发送站		发送寄存器		接收站		接收寄存器		寄存器个数
1	同步(SM)	8291	1	HOST	101	十进制	4	MD(变频器)	101	十六进制	1
2	时间(ms)	100	1	HOST	100	十进制	2	PLC	100	十进制	1
3	时间(ms)	100	1	HOST	102	十进制	3	IS(伺服)	200	十六进制	2
4	时间(ms)	100	1	HOST	110	十进制	1	HOST	110	十进制	4
5			1	HOST		十进制					
6			1	HOST		十进制					
7			1	HOST		十进制					
8			1	HOST		十进制					
9			1	HOST		十进制					
10			1	HOST		十进制					
11			1	HOST		十进制					
12			1	HOST		十进制					
13			1	HOST		十进制					
14			1	HOST		十进制					
15			1	HOST		十进制					
16			1	HOST		十进制					
本站接收	---	---	---	---	---	---	---	---	---	---	---
1	同步(SM)	8291	2	PLC	100	十进制	1	HOST	100	十进制	4
2	同步(SM)	8291	4	MD(变频器)	8000	十六进制	4	MD(变频器)	0	十六进制	1

图 3-75 同步触发配置

SM8291 的使用方法与 PLC 事件触发的 SM 元件一致，同时也可以在网络运行时在 CANlink3.0 主界面点击“同

步发送触发”执行相应操作。

只有主站的 SM8291 才有意义，对从站 PLC 的 SM8291 执行操作不会产生任何作用。

同步触发实际上就是网络中所有站点共用主站 SM8291 为事件的事件触发。

5 时间、事件、同步触发区别

项目	时间触发	事件触发	同步触发
发送方式	定时循环发送	事件发生或数据改变时发送，发送完成后自动清除事件	主站使能发送，完成后自动清除事件
实时性	与设定时间有关，时间越短实时性越高	事件发生即触发，实时性好	主站 M8291 置位即触发，实时性略好于事件触发
执行次数	定时发送	事件发生时触发一次	事件发生时触发一次
PLC 中编程	定时发送，无需程序设计	需在程序中设计相应的时序逻辑	需在程序中设计相应的时序逻辑
网络占用率	高	低	低
适用场合	无特殊的时序要求，可以持续写入的数据	有时序要求，只需一次写入或持续写入会造成故障的场合	主站在特定场合时需多个从站集体返回的数据

3.8.6 接收配置

接收配置是配置各站接收网络中点对多数据的配置。

图 3-76 接收配置对话框

如上图，2 号从站接收配置中有 1 和 3 号站，表示 2 号从站可以接收到网络中所有 1 号站和 3 号站发出的点对多数据帧，而其他站发出的点对多数据则不会接收。如接收到的点对多数据的目的寄存器地址符合 2 号站的定义，则接收的数据生效，如不符合则即使接收也会将其丢弃。

每个站点最多允许接收其他 8 个站点发出的所有点对多数据。每个站点发出的点对多数据没有限制可以接收的站点数目。

通过点对多可以实现一个站点对多个站点同一功能码地址的修改，并可以实现同步的功能。

3.8.7 主站同步写

主站同步配置是主站特有的一个配置，主要用于主站对一台或多台从站的多个寄存器或功能码同时进行写入操作。譬如，主站可以通过同步写多台伺服的功能码 H31-00 来控制它们同时启动或停止。

1 触发元件

如在“配置向导”中的“主站同步写触发元件 (M)”已填写相应的触发元件，则可以在主界面双击主站打开主站配置，在“同步配置”中“触发条件”下拉切换不同的配置条件进行配置。如在前面配置向导中没有填写，此处可打开但无法填写或下拉切换条件，可以点击主界面的“站点管理”重新进入向导增加 / 删除触发元件。

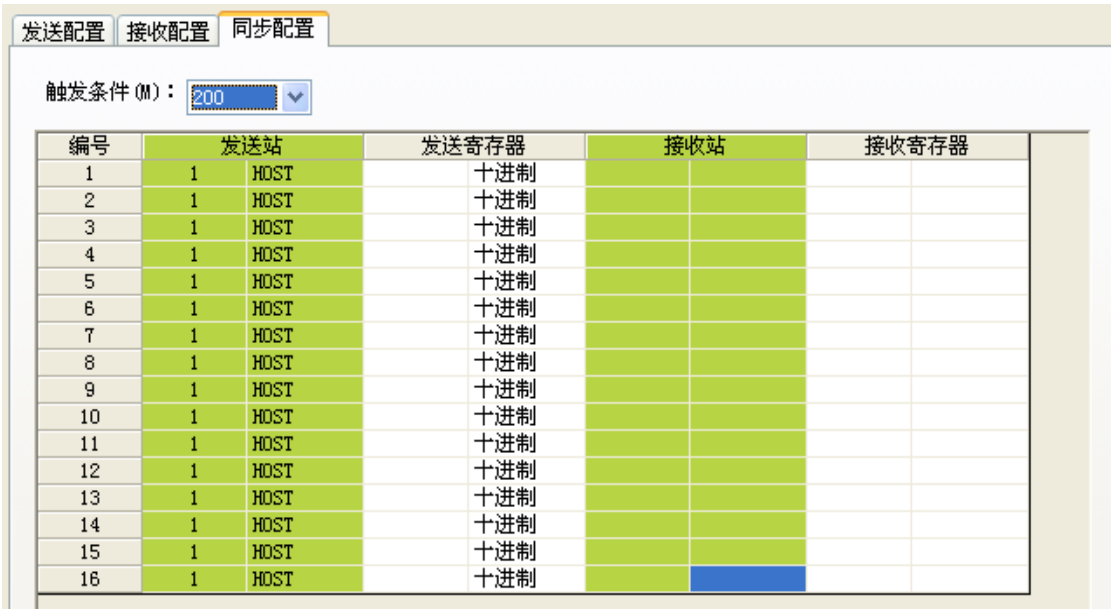


图 3-77 主站同步写对话框

当主站的某个同步写触发元件置 ON 时，该触发元件下的所有配置将依次全部发送出去，从站接到这些数据后，会存放于缓冲区。主站检测到触发条件下的同步写已全部成功发送时将自动广播发送一个生效指令，使所有接收到数据的从站将数据从缓冲区中取出并生效。

主站同步写最多允许 8 个不同的触发条件，每个触发条件最多可以配置 16 条同步写，同时单个从站最多接收 8 条同步写配置（超出后后台会自动提示）。

2 对伺服 32 位寄存器的操作

CANlink3.0 中是对 16 位寄存器和功能码进行操作，如需对 32 位寄存器或功能码进行操作，有如下方法：

主站同步写 32 位寄存器或功能码的高低 16 位地址。

例如，假设在伺服中的功能码 H11-12（第 1 段移动位移），如下操作可将主站的 SD1000 作为低 16 位，SD1001 作为高 16 位写入到 3 号伺服的 H11-12 中。

图 3-78 两个 16 位寄存器合并 32 值

建议用 M 元件的上下沿导通触发元件的 SET 语句。操作 32 位地址的功能码时，不允许只对低 / 高 16 位地址进行操作，也不允许把 32 位功能码的高低 16 位地址拆分到不同的触发条件下操作。

另外，也可以用普通发送配置将连续的两个 SD 元件的值一次性写入到伺服 32 位功能码的低地址位中，如下图:

编号	触发方式	触发条件	发送站		发送寄存器		接收站		接收寄存器		寄存器个数
1	时间(ms)	100	1	HOST	1000	十进制	3	IS(伺服)	110C	十六进制	2
2			1	HOST		十进制					
3			1	HOST		十进制					
4			1	HOST		十进制					

图 3-79 一个 32 为值写入到两个 16 位寄存器

3.8.8 本地从站配置

本地从站指 PLC 作为 CANlink 从站，远程从站指主站下挂接的从站，如伺服、变频器、温控模块，甚至 PLC。一台 PLC 可以作为主站或者作为本地从站，只能选择其一。

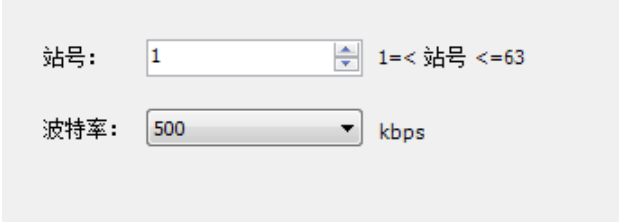


图 3-80 本地从站配置对话框

站号：本地从站站点标示符，不能和 CANlink 网络中其它站点重复，范围 1-63。

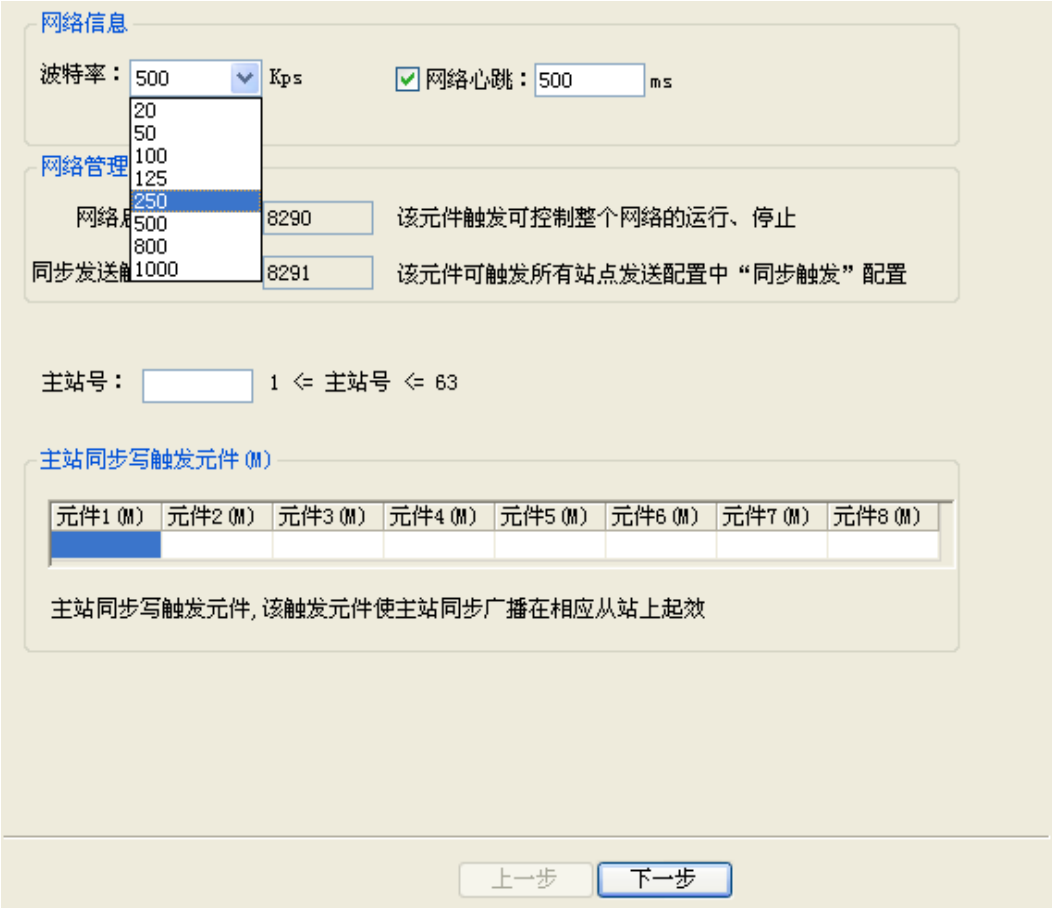
波特率：本地从站数据通信波特率，必须和主站波特率设置一致。

3.8.9 设备接入 CANlink3.0 网络

1 AM600 系列 PLC 接入 CANlink3.0 网络

AM600 的站号和波特率只支持软件设置，设置完成生效需要重启机器或者重新下载运行程序。

在同一个网络中，站点号需唯一，如出现重复站号，后接入的站将会自动关闭通信功能。同时不允许网络中存在 0 号站。

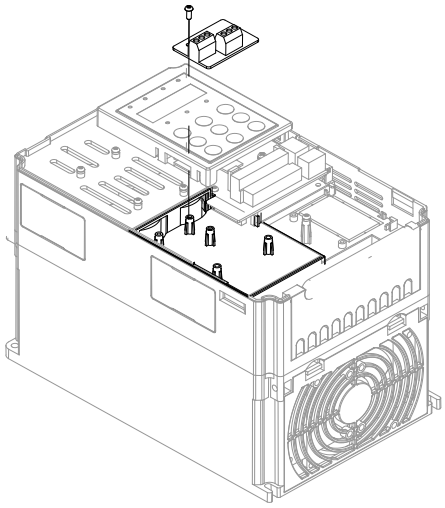


2 MD380/500 变频器接入 CANlink3.0 网络

变频器 MD38CAN1 扩展卡安装

将 MD38CAN1 卡嵌入汇川技术的变频器中，MD38CAN1 扩展卡（CANlink）不允许带电拆装，安装前请关断变频器供电电源，10 分钟后等变频器充电指示灯彻底熄灭后才能进行安装。请参考下图的安装示意进行安装。

将 MD38CAN1 卡插入变频器后，固定相应的螺钉。



变频器 MD380/500 相关的设定

如需设定变频器的启停由 CANlink 网络控制，则需设定变频器的命令源为通信命令通道，即，将 F0-02 设为 2。如不需要，按实际需要设定即可。

站号设置

设置功能码 Fd-02 可设定站号，设定范围 1~63，其它值 CANlink3.0 将不能访问，修改后即时生效。网络中站号需唯一，如有重复，后接入的重复站点将自动关闭通信功能。

波特率设定

设置功能码 Fd-00 可设定波特率，其中千位对应 CANlink 的波特率，对应关系如下：

功能码地址	名称	设定范围	默认值
HFd-00	波特率	个位：Modbus 十位：Profibus-DP 百位：保留 千位：CANlink 0：20Kbps 1：50Kbps 2：100Kbps 3：125Kbps 4：250Kbps 5：500Kbps 6：1000Kbps	5005

CANlink3.0 网络中所有站点的波特率必须一致，否则不能正常通信。

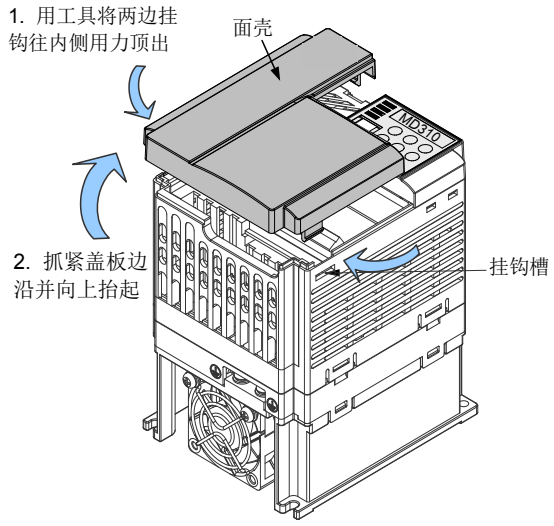
NOTE

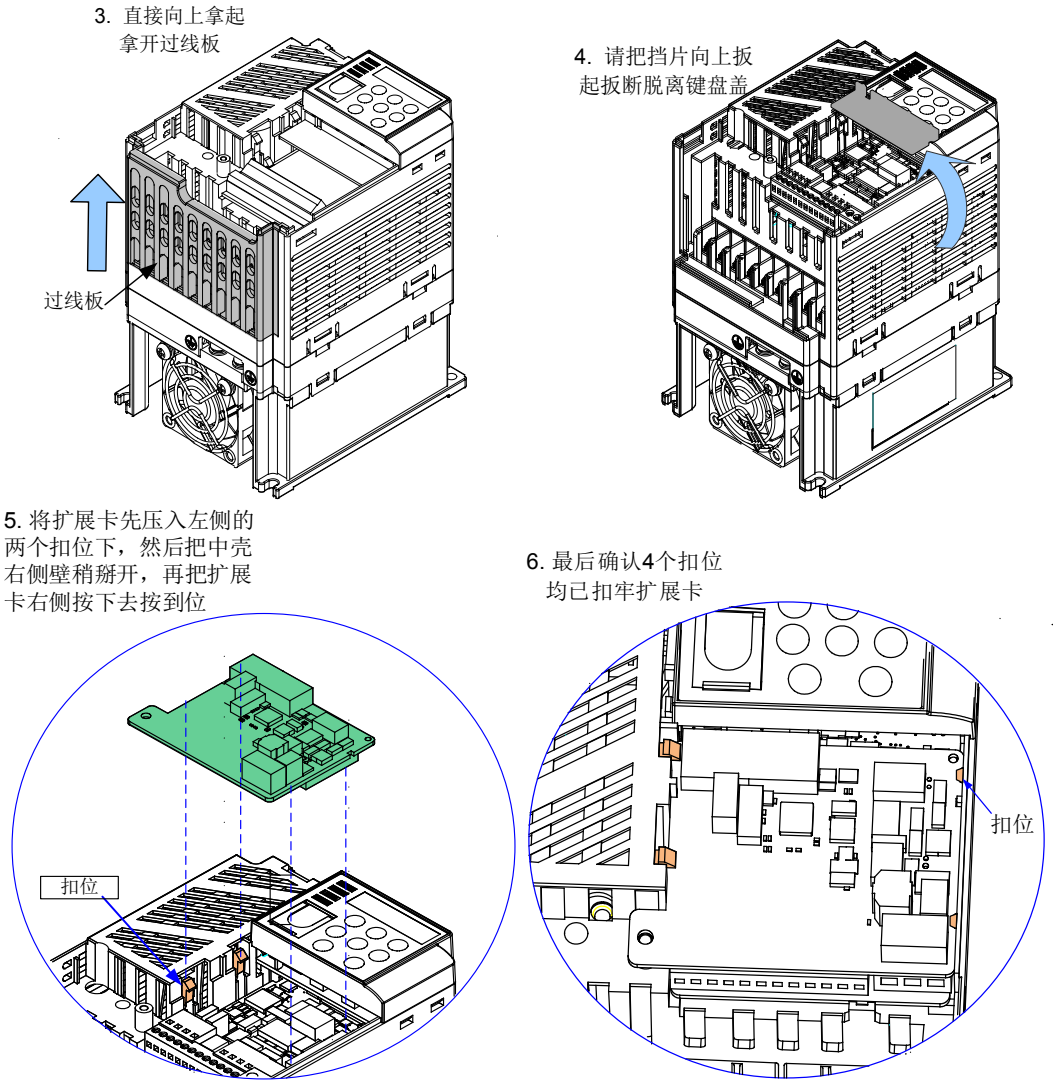
MD380/500 不支持 800Kbps 波特率。

3 MD310 变频器接入 CANlink3.0 网络

MD310-CANL 卡安装

将 MD310-CANL 卡嵌入汇川技术的变频器中，安装前请关断变频器供电电源，10 分钟后等变频器充电指示灯彻底熄灭后才能进行安装。请参考下图的安装示意进行安装。





MD310 相关设定

MD310 在使用 CANlink3.0 时，站号及波特率均由 CAN 扩展卡上的拨码开关进行设定，MD310-CANL 的拨码开关 S1、S2 组成 10 位拨码开关，用于设置 CAN 总线通讯波特率与通讯设备地址。拨码开关编号如下图所示，其中 Bd1、2、3 用于设置波特率，Adr1~7 用于设置 CANlink 地址。拨码打到“ON”表示“1”，打到下面表示“0”。波特率及站号的修改会立即生效。

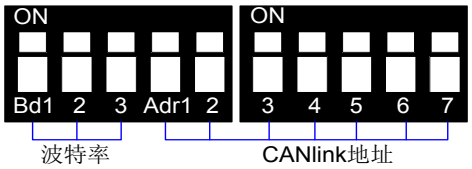


图 3-81 MD310-CANL 拨码开关

波特率设置

拨码与波特率的对应关系如下表所示，可设置 8 种波特率。

MD310-CANL 波特率

拨码号 Bd			波特率
1	2	3	
0	0	0	20Kbps
0	0	1	50Kbps

拨码号 Bd			波特率
1	2	3	
0	1	0	100Kbps
0	1	1	125Kbps
1	0	0	250Kbps
1	0	1	500Kbps
1	1	0	800Kbps
1	1	1	1Mbps

CANlink 设备地址

MD310-CANL 提供 7 位拨码开关用于 CANlink 通讯地址设置，拨码 “Adr1” 表示最高位，拨码 “Adr7” 表示最低位。拨码 Adr1--7 对应一个地址站号的 b6--b0 位。拨码开关有效地址设置范围是 1~63，如下表所示，0 地址以及 64~127 为保留地址，不允许使用，设置为保留地址时 MD310-CANL 卡将不工作。

MD310-CANL 拨码地址

拨码号 Adr							地址
1	2	3	4	5	6	7	
0	0	0	0	0	0	0	保留
0	0	0	0	0	0	1	1
0	0	0	0	0	1	0	2
0	0	0	0	0	1	1	3
……							…
0	1	1	1	1	1	1	63
1	x	x	x	x	x	x	保留

关于 MD310-CANL 扩展卡的其它使用注意事项，请参见《MD310-CANL 通讯扩展卡说明书》。

4 伺服接入 CANlink3.0 网络

站号设置

修改功能码 H0C-00 的值可修改伺服的站号，设定范围为 1~63，其它值 CANlink3.0 将不能访问，修改后即时生效。网络中的站号需唯一，如有重复，后接入的重复站点将自动关闭通信。

波特率设定

修改功能码 H0C-08 的值可修改伺服的波特率，对应关系为：

功能码地址	设定范围	默认值
H0C-08	0：20Kbps 1：50Kbps 2：100Kbps 3：125Kbps 4：250Kbps 5：500Kbps 6：800Kbps[1] 7：1000Kbps	5

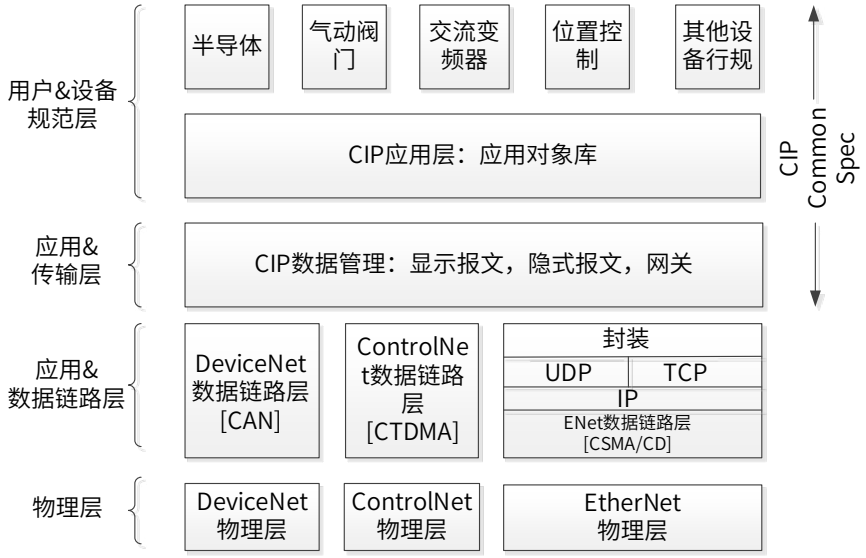
[1] IS620P 不支持 800Kbps 波特率，选择 6 时是 1000Kbps。

设定后即时生效，网络中所有站点的波特率必须一致，否则不能正常通信。

3.9 EtherNet/IP 通讯

3.9.1 协议概述

EtherNet/IP 通讯协议是适合工业环境应用的协议体系（IP 是“Industrial Protocol”的简称），基于传统的以太网协议和标准的 TCP/IP 协议，可以实现工业设备之间应用信息的高效交换。 EtherNet/IP 应用层协议使用标准的面向对象的 CIP 协议【注】。各层结构如图所示。



【注】协议详情请参考 EIP-CIP-V1-1.0、EIP-CIP-V2-1.0 官方标准文档。

EtherNet/IP 协议的技术特点：

1） 标准化。EtherNet/IP 建立在标准的 TCP/UDP 协议之上，完全符合标准的 IEEE802.3U 标准，所有有标准以太网节点的设备均可加入此网络。

2） 实时性【注】。EtherNet/IP 数据传输分为隐式报文通讯（Implicit Messaging）和显式报文通讯（Explicit Messaging）。隐式通讯用于传输实时性数据，采用周期循环方式；显式通讯用于传输非实时性数据（如配置信息等），采用请求 - 应答的方式。

3） 通讯效率高。EtherNet/IP 采用生产者 / 消费者技术，允许网络上的节点同时存取同一个源的数据。在生产 / 消费模式中，对每个数据均分配一个唯一的标识，每个数据源将数据一次性的发送到网络上，其他节点选择性的读取这些数据，从而极大的提高系统的通信效率。

4） EtherNet/IP 设备应用广泛。EtherNet/IP 通讯设备包括简单的 I/O 设备、传感器（扫码枪 / 相机）、执行器以及复杂的控制设备（机器人 /PLC/ 焊机等）。

中型 PLC 后台 InoProShop1.5.0 及以上版本支持 EtherNet/IP 功能，其通讯规格如下：

1） AM400/AM600 系列 PLC 支持一路 EtherNet/IP 主站或从站功能。

2） AC800 系列 PLC 支持一路 EtherNet/IP 主站功能。

3） 最小循环通讯周期（RPI）为 5ms。

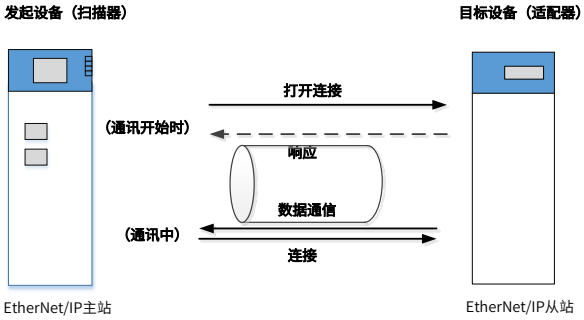
4） 单个通信连接最大支持 500 个字节数据读写。

5） 支持最多从站连接个数为 64 个。

【注】：在同时有 EtherCAT 和 EtherNet/IP 网络的组网工程中，由于默认 EtherCAT 优先级为最高 , 故会降低 EtherNet/IP 网络的通讯实时性。

3.9.2 PLC 作为 EtherNet/IP 主站的配置

通讯开始时，打开连接的一端称为发起设备，也叫扫描器（Scanner），为通常意义的 EtherNet/IP 主站。被打开的一端称为目标设备，也叫适配器（Adapter），也就是通常意义上的 EtherNet/IP 从站。

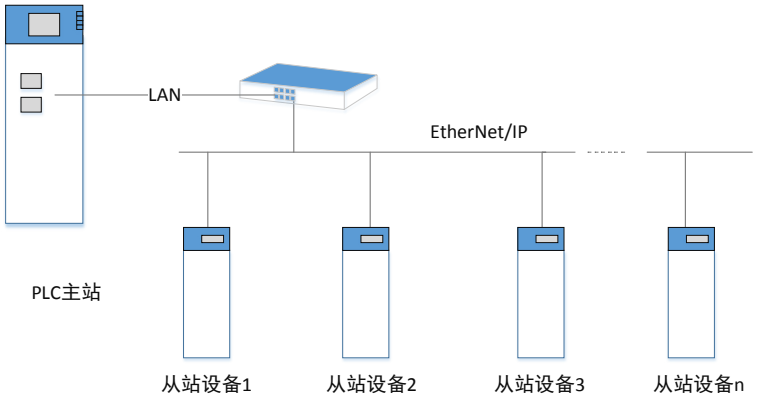


本章节将详细介绍在 InoProShop 后台以汇川技术中型 PLC 为 EtherNet/IP 主站的配置方法。

1 EtherNet/IP 设备 IP 设置

EtherNet/IP 支持的拓扑结构有总线型结构、星型结构、混合结构和环形结构。

对于星型结构，所有的节点都连接在网络集线器上，网络材料价格低廉，搭接容易，市面上可以找到很多合适的设备，且增减节点和维修都很方便，是目前经常采用的网络结构。在星型结构中，同一个 EtherNet/IP 网络中的所有设备 IP 地址都需要设置在同一个网段，且保证所有设备的 IP 地址不重复。



例如，网络的中的设备 IP 可以设置为如下的形式：

EtherNet/IP 主站的 IP 地址：192.168.1.100

EtherNet/IP 从站 1 的 IP 地址：192.168.1.101

EtherNet/IP 从站 2 的 IP 地址：192.168.1.102

……

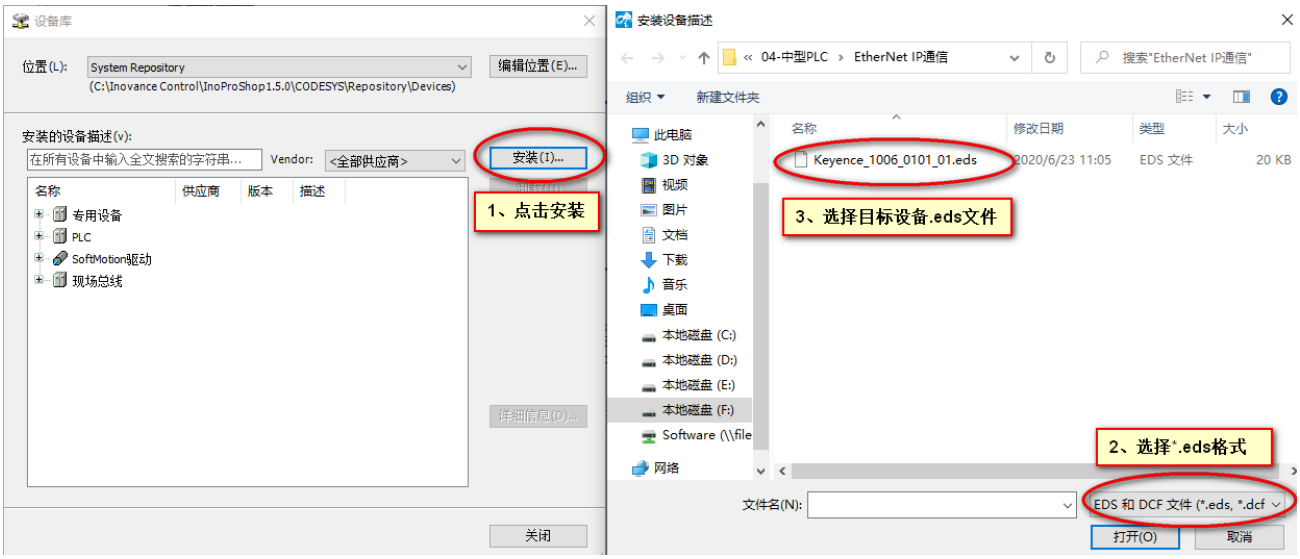
EtherNet/IP 从站 n 的 IP 地址：192.168.1.XXX

2 添加 EtherNet/IP 从站

1） 导入 eds 从站设备描述文件

InoProShop 后台提供两种方法导入 EtherNet/IP 从站设备的 eds 文件。一种是通过设备库添加，点击菜单栏 -> 工具 -> 设备库，点击“安装”导入；另外一种是直接在“Network Configuration”网络组态界面，选择“导入 eds 文件”导入。

以基恩士 N-L20 EtherNet/IP 从站模块设备为例，按照第一种方法，参考下图三个步骤，可以导入第三方设备的 eds 描述文件。



2） 勾选“EtherNet/IP”主站

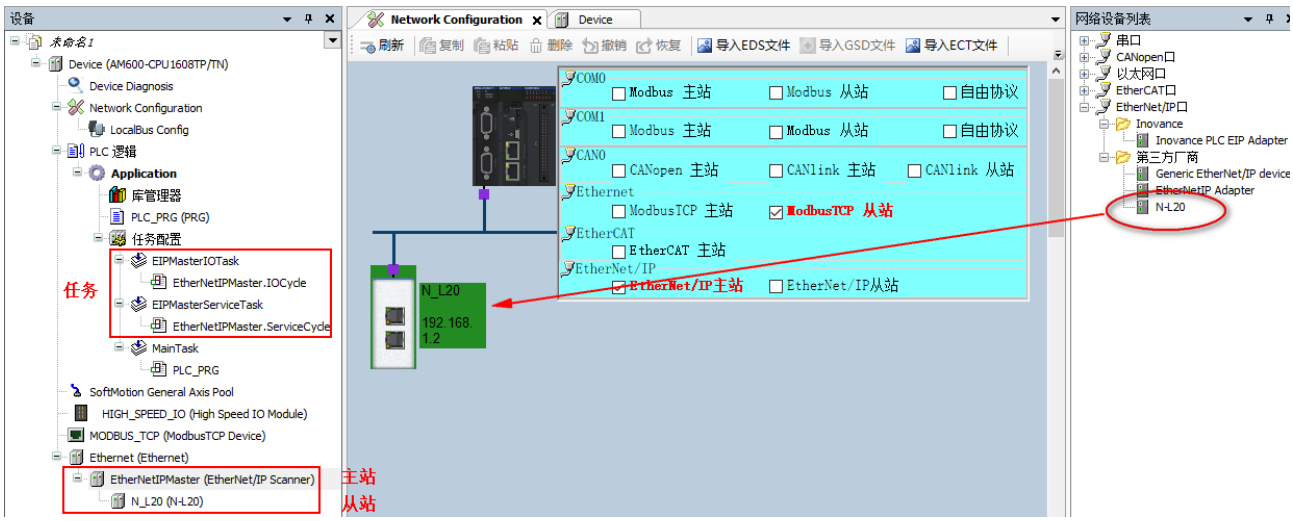
鼠标单击主站设备，在弹出的界面中，勾选“EtherNet/IP 主站”。

3） 添加 EtherNet/IP 从站

■ 方式一：通过“网络设备列表”界面来添加

在右侧“网络设备列表”界面中，在“EtherNet/IP 口列表”下，找到第一步导入的第三方厂商设备，鼠标双击，将该从站设备加入网络组态。

如下图所示，右边的设备树中，主站下生成了 N_L20 从站设备，如果有多个 EtherNet/IP 从站设备，可以依次添加。



在加入 EtherNet/IP 设备后的任务配置下，系统自动生成了 EIPMaster.IOTask 和 EIPMaster.ServiceTask 两个任务，用于更新 EtherNet/IP 的循环通讯数据和服务数据。EIPMaster.IOTask 任务优先级默认为 0，用户可根

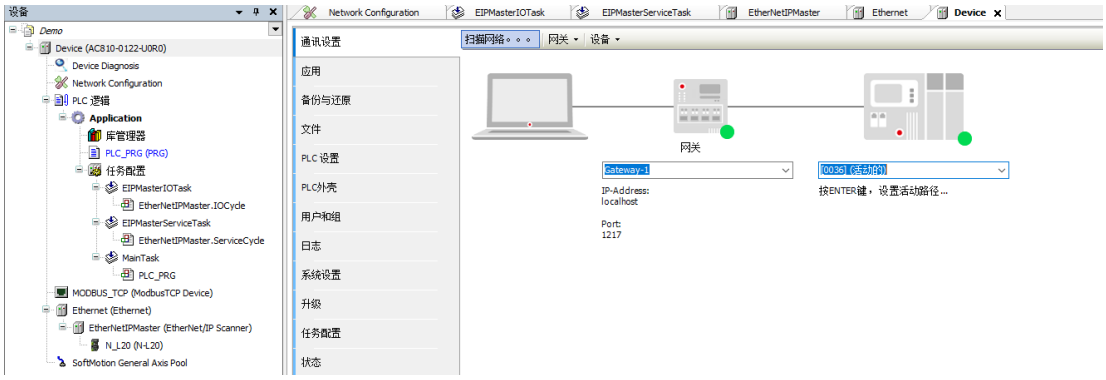
据实际情况进行调整，比如工程中另外有 EtherCAT 任务，其优先级必须指定为 0（最高），那么 EIPMaster.IOTask 的优先级可以修改为 1。

■ 方式二：通过 EtherCAT 扫描功能来添加

在添加从站 eds 文件之后，可登录 PLC，选择设备树“EtherNetIPMaster”，右键选择“扫描设备”功能，扫描当前网络下的 EtherNet/IP 设备，再通过“拷贝所有扫描设备”功能添加从站，详细操作请参考 EtherCAT 扫描功能。

3 EtherNet 通用设置

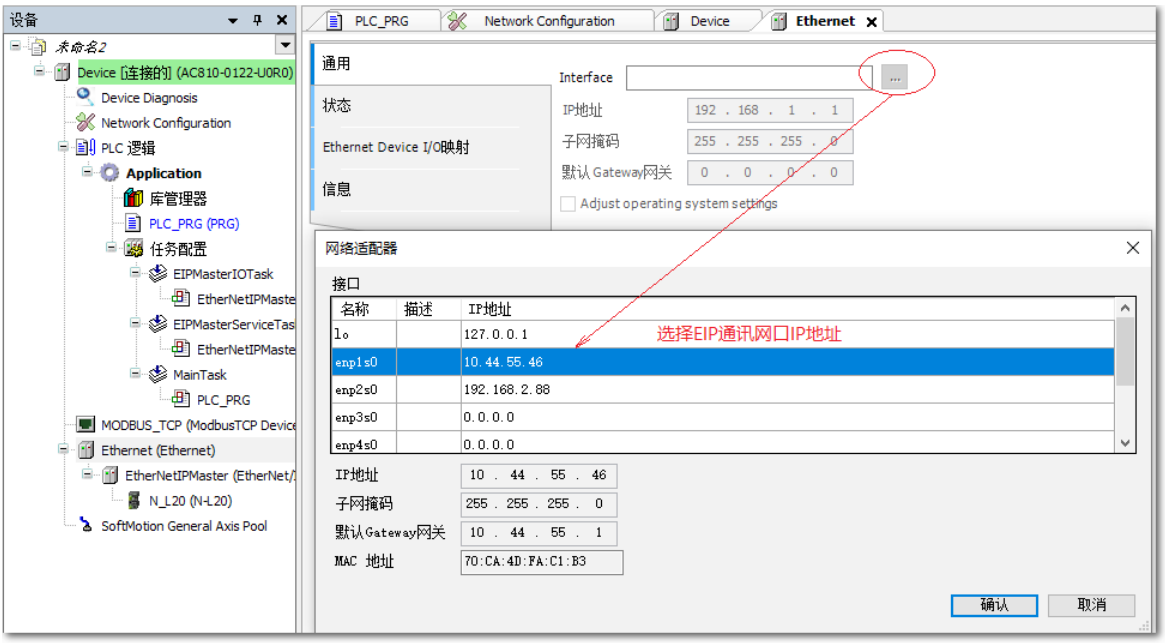
在 EtherNet 通用设置之前，首先配置 PLC 网关，可实现通过后台登录连接 PLC。



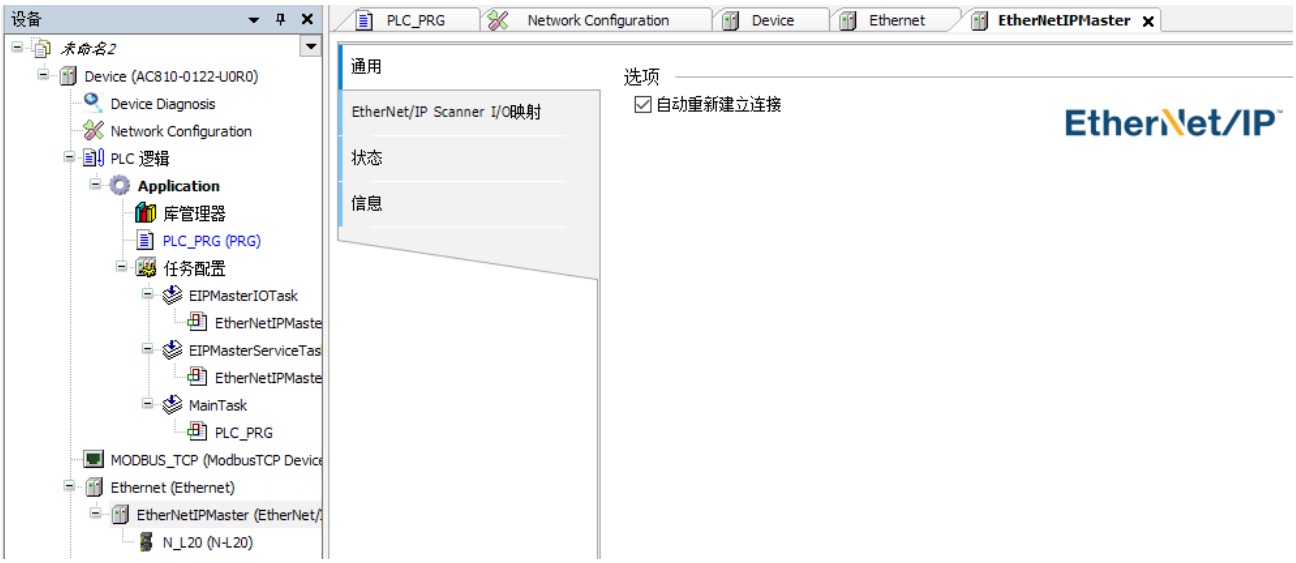
双击 Ethernet 设备，进入“通用”设置界面，如下图，打开“网络适配器”界面。

选择对应网络接口的主站 IP 地址。

注意：该 IP 地址是指 PLC 作为主站与外部设备连入 EtherNet/IP 网络的 IP 地址。对于 AC800 系列，具备双网口，请注意区分连接设备的 IP 和网口对应是否一致；对于 AM600 系列，只有一个以太网口，请确保其 IP 地址和登陆 PLC 的 IP 地址相一致。



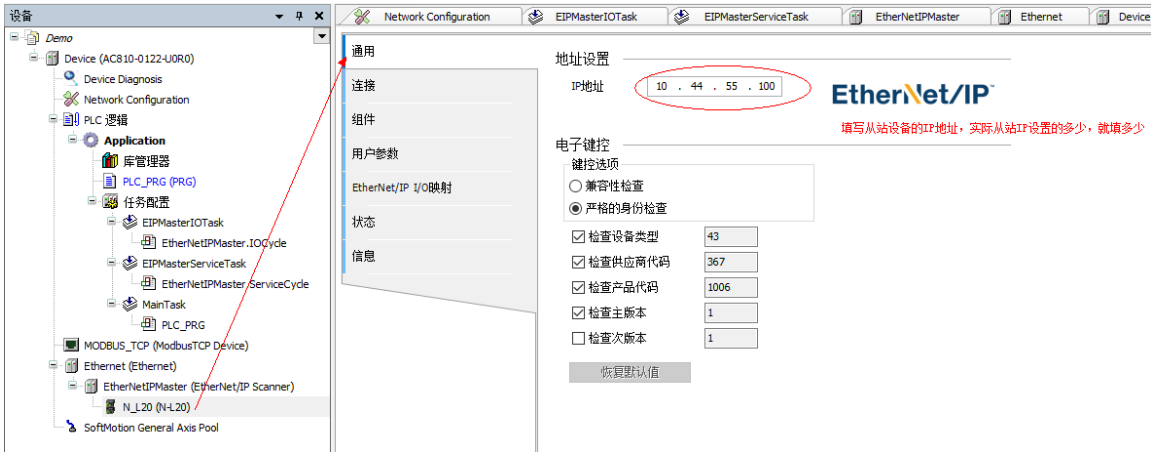
另外，主站通用设置界面，默认勾选自动重新建立连接，在从站链路丢失恢复时可以自动重连，如图所示。



4 EtherNet/IP 从站设置

■ 通用设置

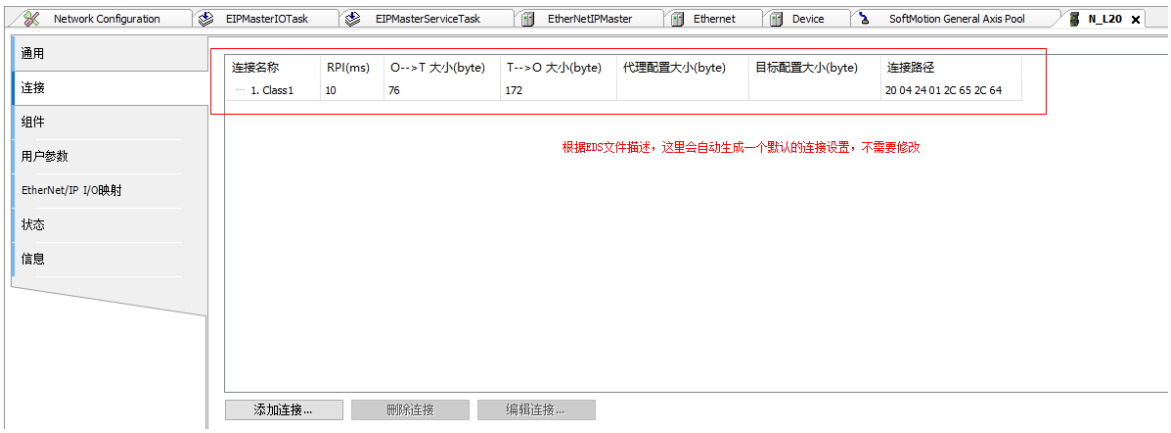
双击需要设置的 EtherNet/IP 从站“N-L20”，进入设置界面。需要手动设置从站 IP 地址，使其与从站设备实际的 IP 地址一致。



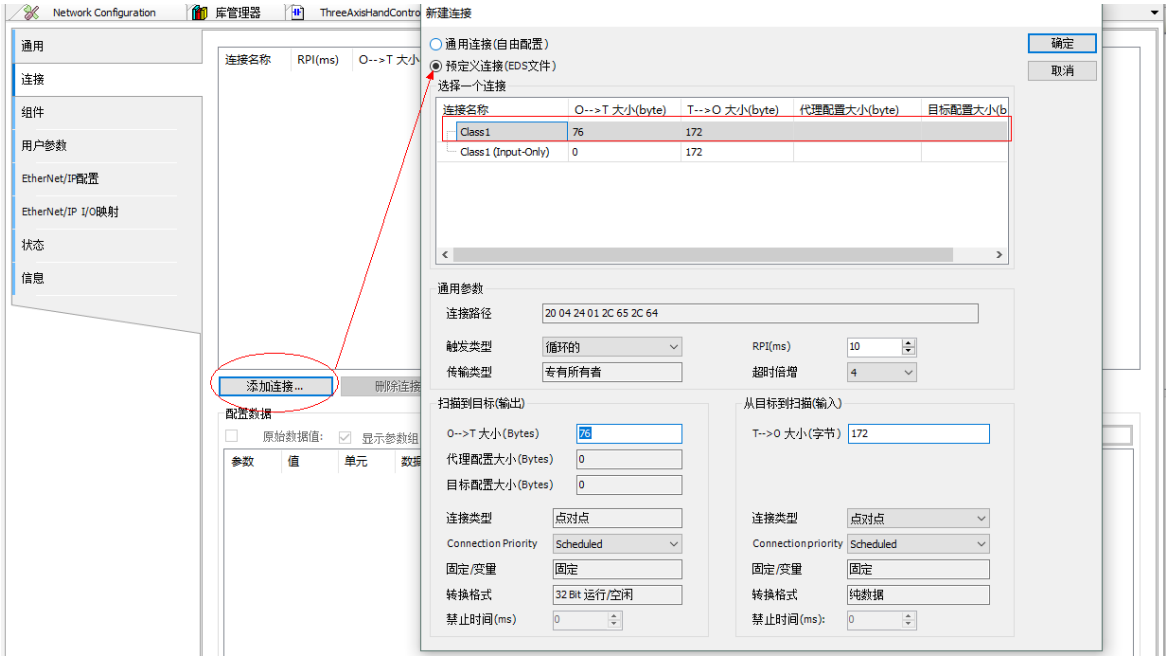
5 连接设置

1） 添加连接

EtherNet/IP 从站 eds 描述文件均包含一个默认的连接路径，添加 EtherNet/IP 网络组态后，后台连接界面会加载该默认的连接路径，如下所示。



也可通过“添加连接”选择设置 eds 文件中默认预定义的其他连接路径。如下图所示。



在上图中单击“编辑连接”，可进入连接设置界面，一般情况下，默认路径除了 RPI（通讯周期）需要根据具体应用做修改以外，其他参数均直接使用默认值。

2） 通用参数

■ 连接路径：该参数规定了一帧字节流的格式和连接实例。例如连接路径：20 04 2C C6 2C 68 （详情参考 EIP-CIP-V1-1.0，Appendix C: Data Management）。

20：Logical Segment、ClassID、8bit logical address

04：表示 Assembly Object（04H）

2C：Logical Segment、Connection Point、8bit logical address

C6：Assembly Object 的实例的 ID-C6H

2C：Logical Segment、Connection Point、8bit logical address

68：Assembly Object 的实例的 ID-68H

注意：连接路径因厂家而异，请根据具体的从站手册来配置。

■ RPI（ms）：Requested Packet Interval 的简称，以 ms 为单位的通讯传输间隔周期，各个节点的 RPI 可单独设置，互不影响。

注意：主站 RPI 周期必须为任务周期的整数倍。

3） 扫描到目标

■ 传输字节大小

O->T Size（Bytes）：从生产（发起设备）到消费（目标设备）传输的数据量，以 byte 为单位。

T->O Size（Bytes）：从消费（目标设备）到生产（发起设备）传输的数据量，以 byte 为单位。

■ 传输类型（Transport Type）

专有所有者（Exclusive Owner）：可同时设定“从发起设备到目标设备的数据发送”和“从目标设备到发起设备的数据接收”。

冗余所有者（Redundant Owner）：允许多个发起设备对同一个目标设备建立相对独立的、相同的连接。

只输入（Input Only）：此连接只能设定“从目标设备到发起设备的数据接收”。

只监听（Listen Only）：应用此连接类型监听组播数据，而不提供配置或调度信息的 EtherNet/IP 设备。

■ 触发类型（Trigger Type）

循环的（Cyclic）：定期触发数据传输。

状态改变（Change-Of-State）：检测到应用对象状态发生改变时传输数据。

应用程序（Application Object）：应用对象触发时传输数据。

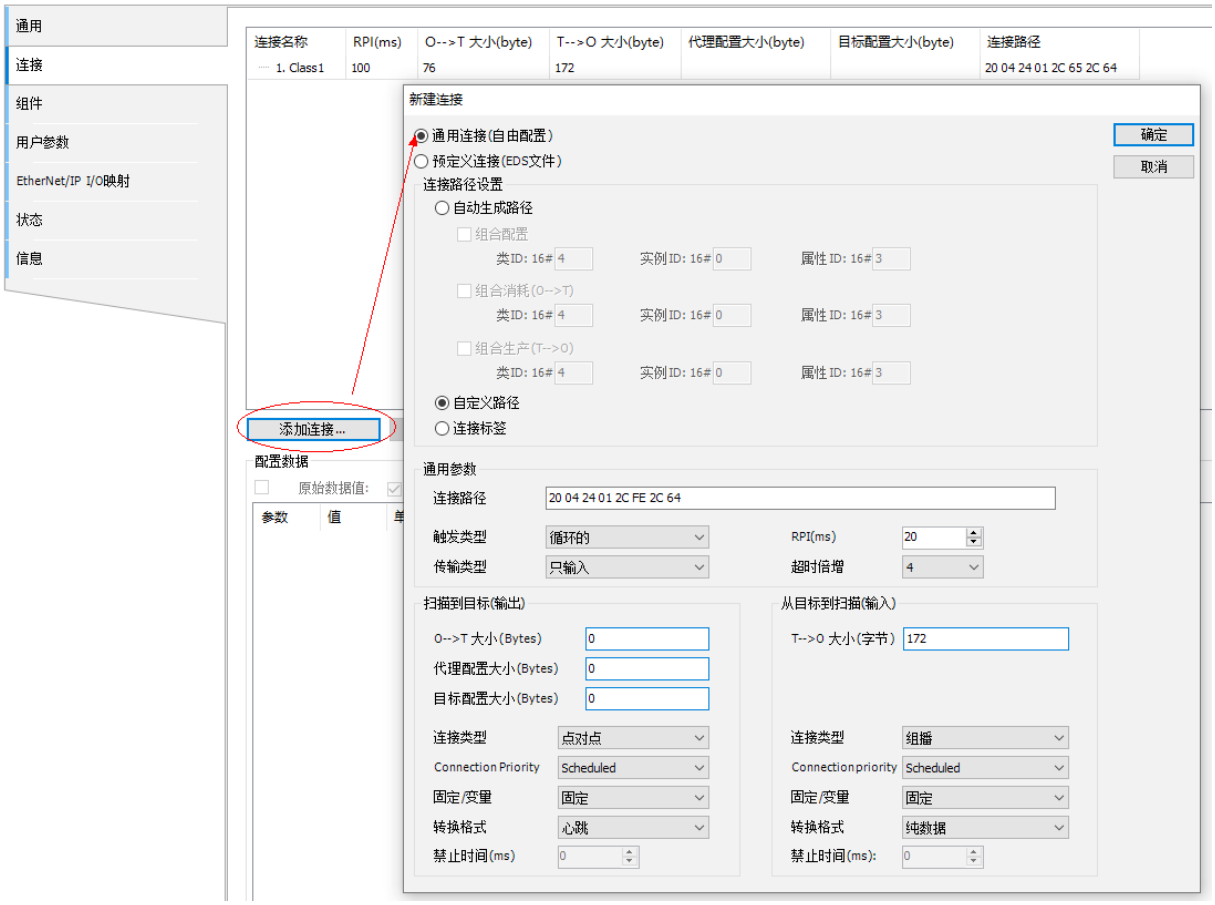
■ 模式（Connection Type）

组播（Multicast）：多台扫描器同时接收一台目标设备的数据。

点对点（Point-to-Point）：扫描器以一对一方式接收目标设备的数据。

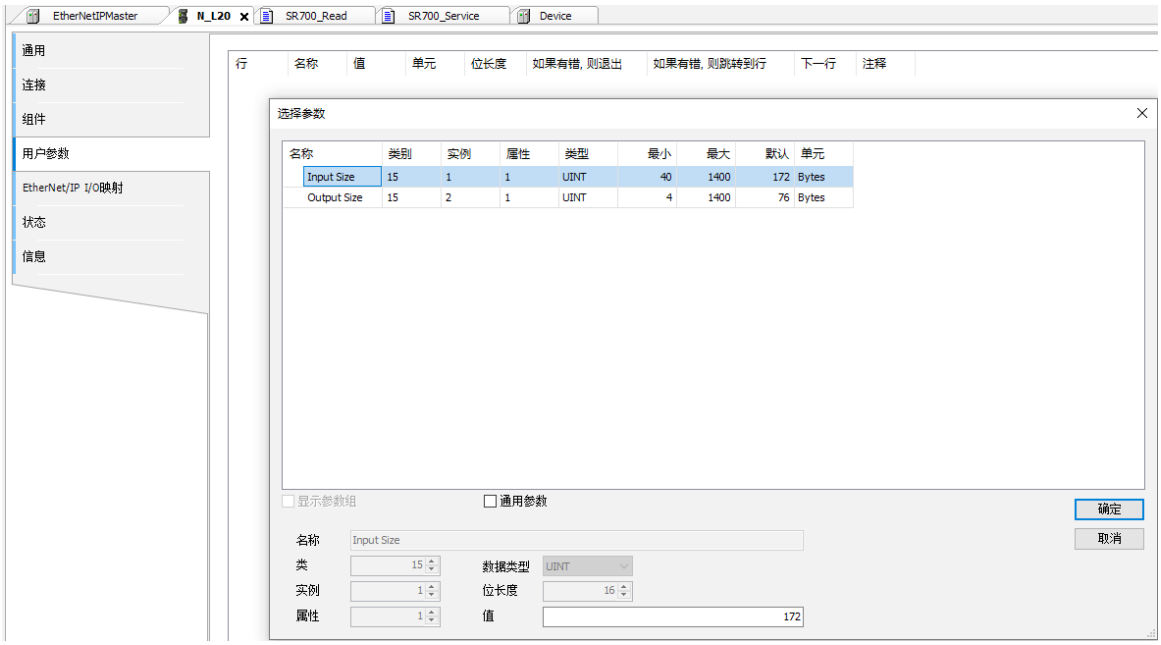
NOTE

在“连接”页面单击“添加连接”，打开连接设置界面，选择“通用连接”，建议使用默认的连接配置（可根据需要自定义一个连接，但需要使用者有一定的 CIP 协议基础）。

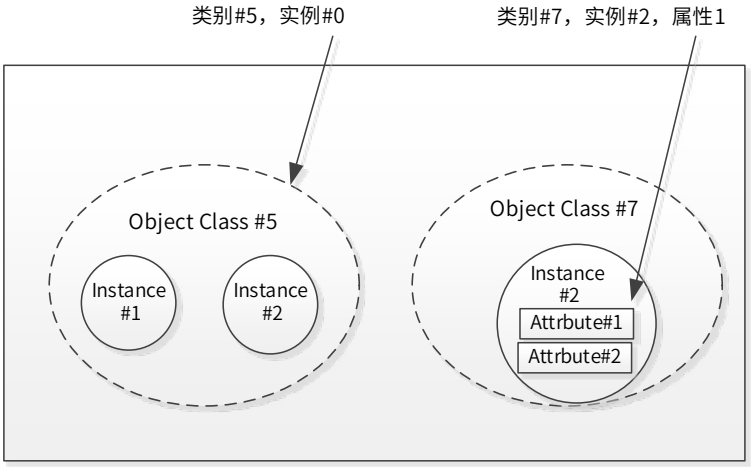


6 设置用户参数

如需根据从站设备要求额外配置一些 EtherNet/IP 总线通信参数，可通过此选项进行配置（此操作需要一定程度的 CIP 协议基础），大多情况下无需配置。配置完成后，从站设备每次通讯启动或者重启时，都会把这些配置的用户参数向主站发送一次。

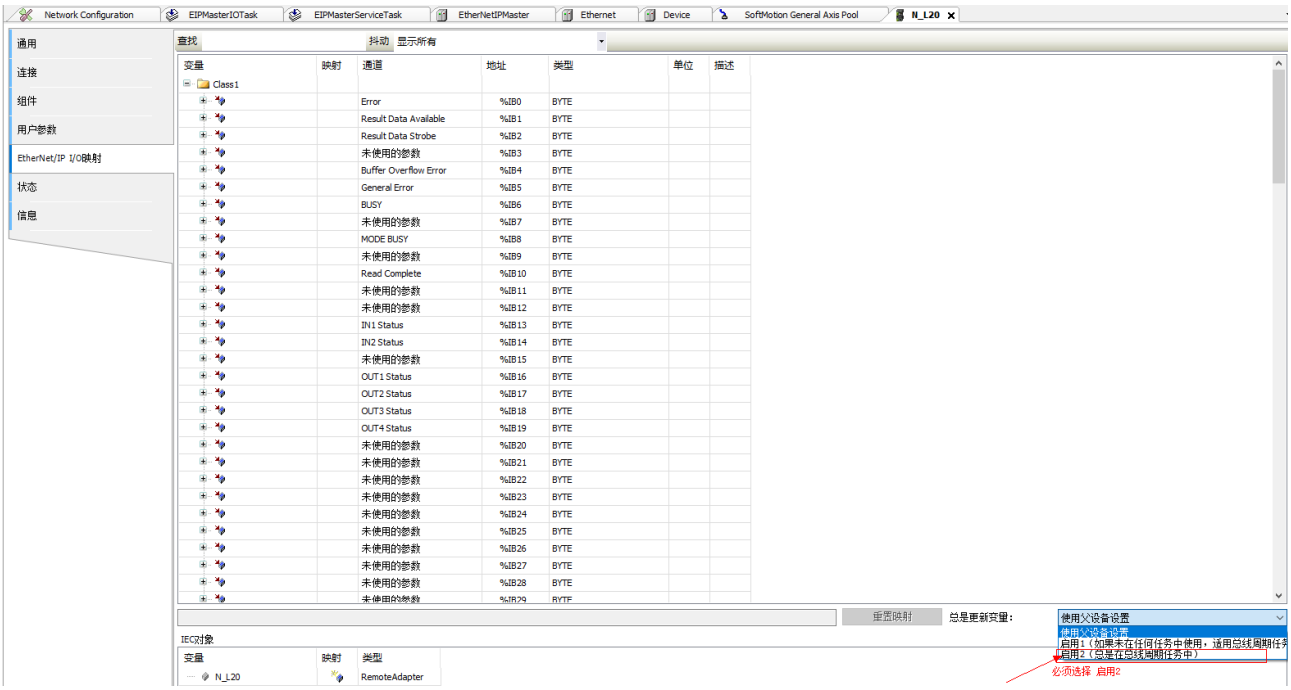


- 名称：参数的名称。
- 类别（Class）：网络中所有可访问的对象类别都有一个唯一的整数值标识号。如图，类别 #5，#7。
- 实例（Instance）：一个物体的具体的和真实的（物理的）出现。例如：新建新西兰是对象类国家的一个实例，如图 Instance#1 和 Instance#2。（CIP 实例 ID 值零为 0，用来表示与类内的特定实例的引用。）
- 属性（Attrbute）：对物体的外部可见特征或特征的描述。通常属性提供状态信息或控制对象。如图 Attrbute#1 和 Attrbute#2。



7 在程序中 IO 变量映射

变量的 EtherNet/IP I/O 映射界面如下图所示，通过地址设置方式可以设置输入、输出变量。直接选中地址选项进行编辑，可修改系统自动分配的地址，从而实现与程序中变量的映射。注意：将右下角的更新变量设置为：启动 2（总是在总线任务周期中）。



3.9.3 PLC 作为 EtherNet/IP 主站的例程

1 EtherNet/IP 主站工程一般配置步骤

1） 新建工程，在“Network Configuration ”组态配置中勾选主站。

2） 导入第三方 eds 文件，将 EtherNet/IP 从站加入组网工程。

3） 设置 EtherNet 通用设置的 IP 地址，以及从站通用界面的 IP 地址，确保均在一个局域网内。

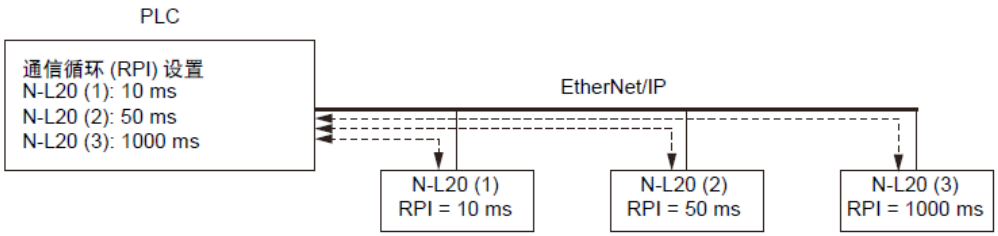
4） 添加从站默认连接，根据需要修改 RPI 通讯周期和任务周期时间。

5） 在从站 EtherNet/IP I/O 映射中进行参数映射。

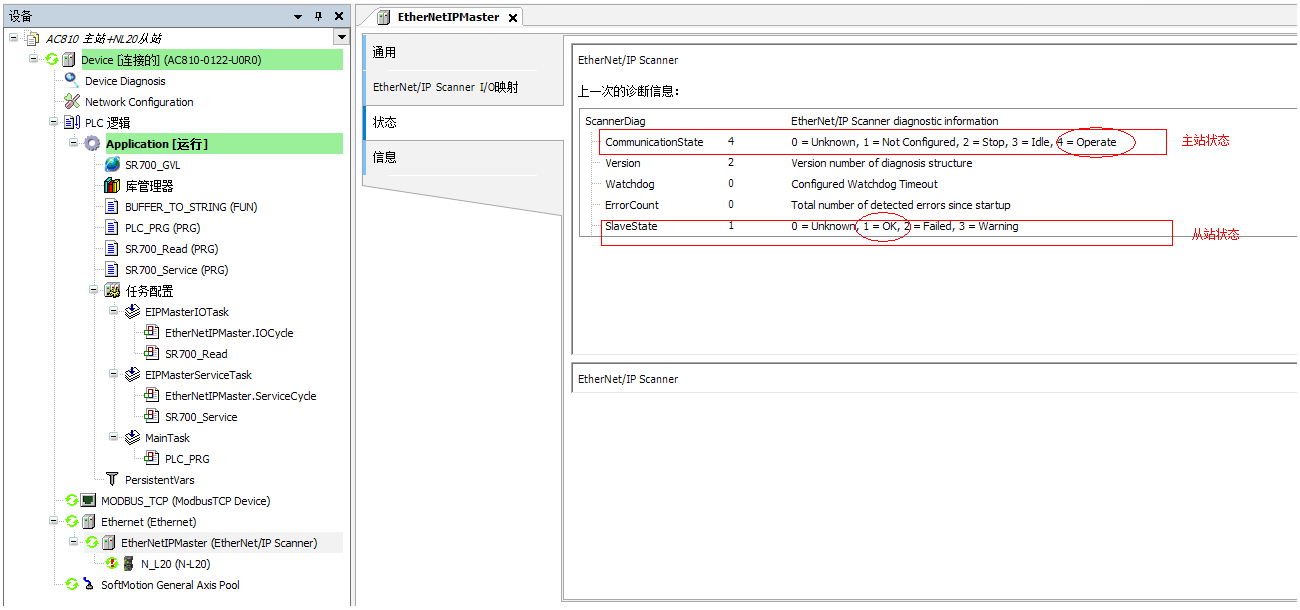
6） 根据需求使用编写用户 POU 程序。

2 EtherNet/IP 主站循环数据通讯工程实例

在循环通信 , 可根据发送和接收数据的优先级设置 RPI（通信周期），以发送和接收整体通信负载调整后的数据，如下示意图所示。



本实例工程以 AC810 为 EtherNet/IP 主站，基恩士 NL-20+SR700 扫码器为 EtherNet/IP 从站。按照上述设置步骤，配置完成后进行默认连接，启动运行程序。设备树状态全部显示为绿色时，表示启动通讯连接成功。同时，在主站状态中可以查看 CommunicationState=4，为 OP 状态，从站 SlaveState=1，为 OK 状态，ErrorCount =0。如下图所示。

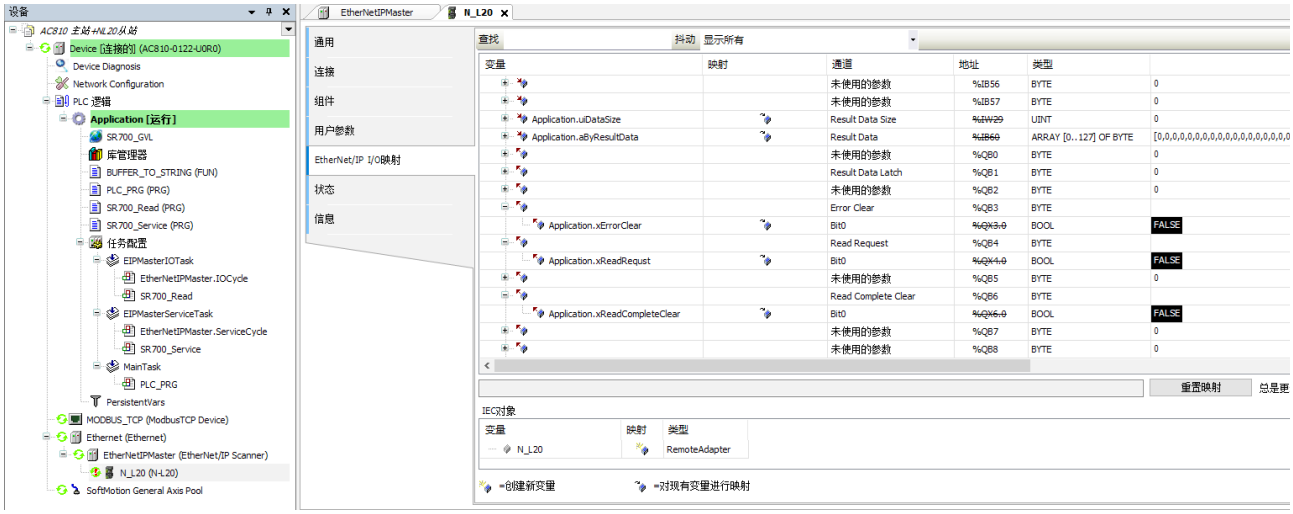


NOTE

在用户程序中判断主从站是否正常通讯，可通过 EtherNetIPMaster.eState =6（RUNNING）进行状态判断，从站通过具体实例化的从站名称 N_L20 .eState = 8（RUNNING）进行状态判断。

■ 基恩士 NL20 和扫码模块循环通讯实例测试步骤如下：

1） 映射从站模块的操作变量，如下图，定义全局变量与 EtherNet/IP I/O 映射相关联。



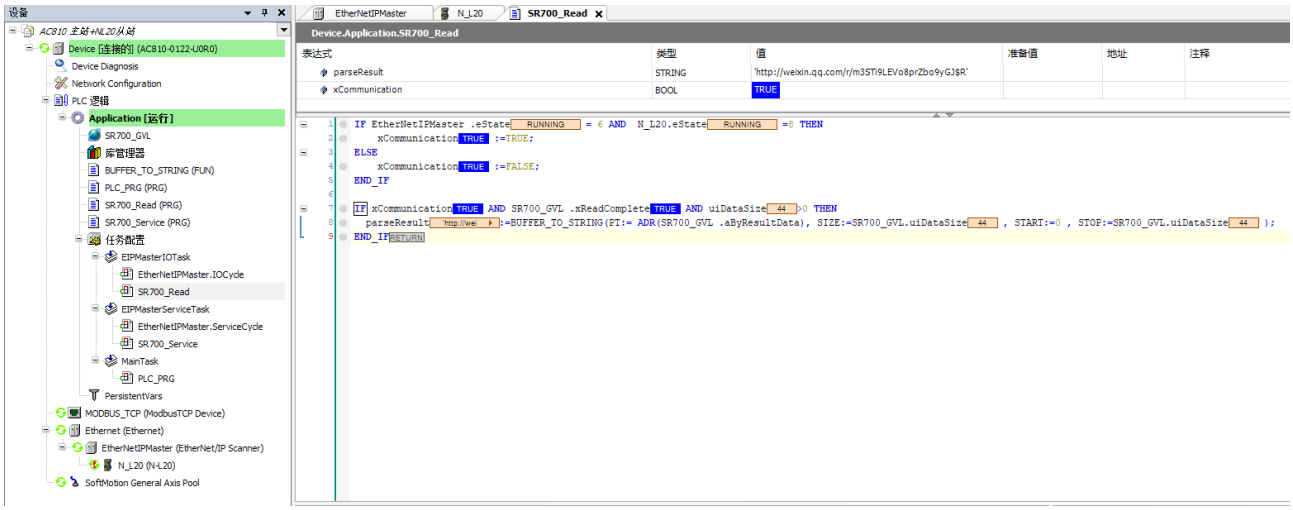
2） 读取条码时，将在“读取数据”中写入数据。此时，“Read Complete”变为 TRUE(1)。

3） 确认“Read Complete”为 TRUE（1）后，将“Read Comlplete Clear”设为 TRUE（1）。

4） 将“Read Comlplete Clear”设为 TRUE（1）时，“Read Complete”变为 FALSE（0）。

5） 确认“Read Complete”为 FALSE（0）后，将“Read Comlplete Clear”设为 FALSE（0）

用户程序中在每个循环周期判断通讯状态，当通讯连接状态正常，传感器扫码完成后，xReadComplete 标志置 TRUE，返回扫描数据不为 0，通过 BUFFER_TO_STRING 对数据结果进行解析，使用示例程序和结果如图所示：



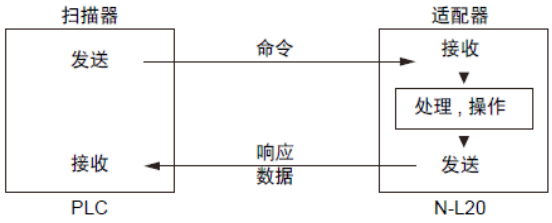
扫码器扫出的二维码字符串结果如图。

Device.Application.SR700_Read

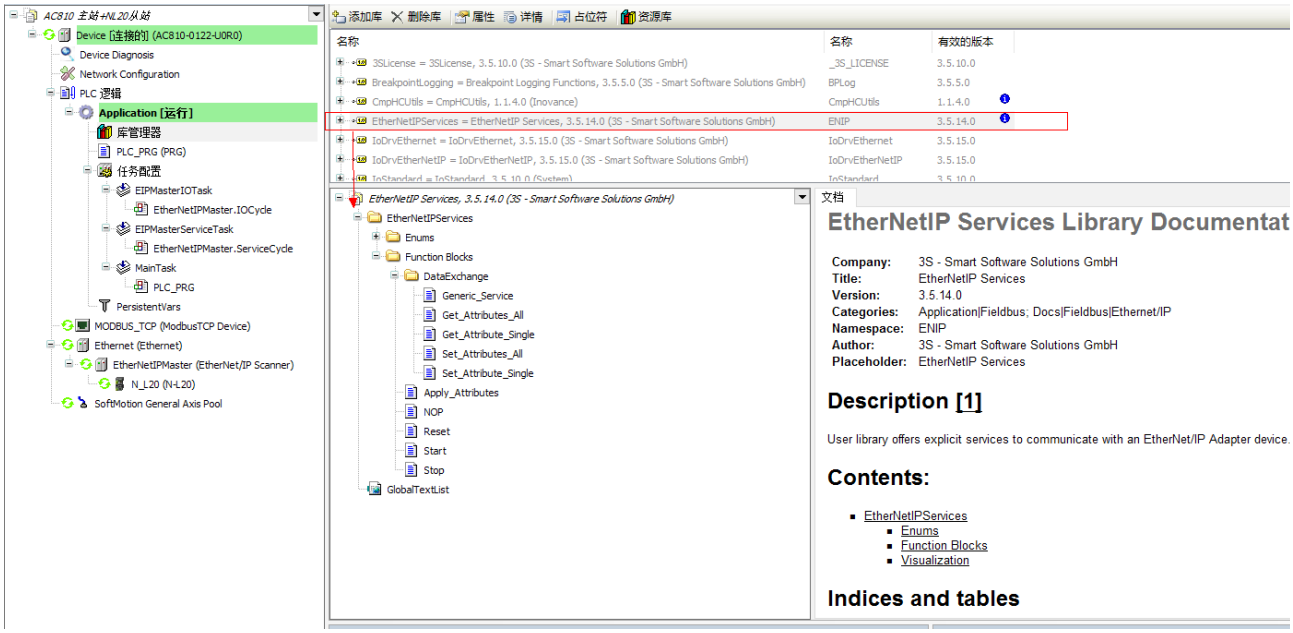
表达式	类型	值	准备值	地址	注释
parseResult	STRING	'http://weixin.qq.com/r/m3STi9LEVo8prZbo9yGJ$R'			
xCommunication	BOOL	TRUE			

3 EtherNet/IP 主站服务数据通讯工程实例

在服务信息通信中 , 通过命令 / 响应控制时序，如下示意图所示。



根据服务数据通讯发送命名格式，通常需要指定 Service code、Class ID，Instance、Attribute ID、Service data。在应用程序中可以通过 EtherNet/IP 服务功能块库 EtherNetIPService 来实现，如下图库管理器中所示：



该功能块库提供了 CIP 通讯协议大部分公用的服务（CIP Common Services），服务 ID 和名称如下表所示。

表 3-5 表 CIP 公用服务 ID 和名称

ID	名称
00	Reserved for future use
01	Get_Attributes_All
02	Set_Attributes_All Request
03	Get_Attribute_List
04	Set_Attribute_List
05	Reset
06	Start
07	Stop
08	Create
09	Delete
0A	Multiple Service Packet
0B-0C	Reserved for future use
0D	Apply_Attributes
0E	Get_Attribute_Single
0F	Reserved for future use
10	Set_Attribute_Single
11	Find_Next_Object_Instance
12-13	Reserved for future use
14	Error Response (used by DeviceNet only)

通过 Get_Attribute_Single 和 Generic_Service 两个功能块，可以获取用户程序中的属性和服务数据。

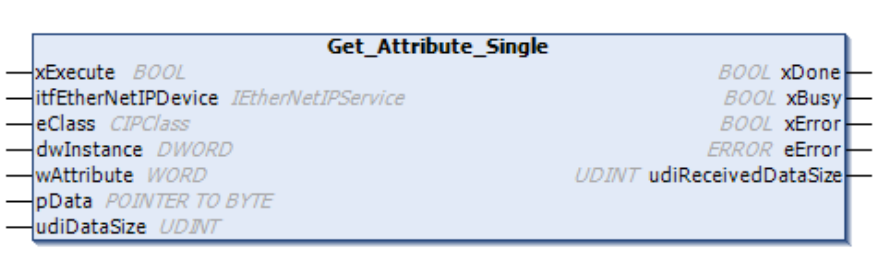
■ Get_Attribute_Single

基恩士 NL-20 模块手册上提供了特殊的属性 ID，来获取当前读码器的状态，如下表所示：

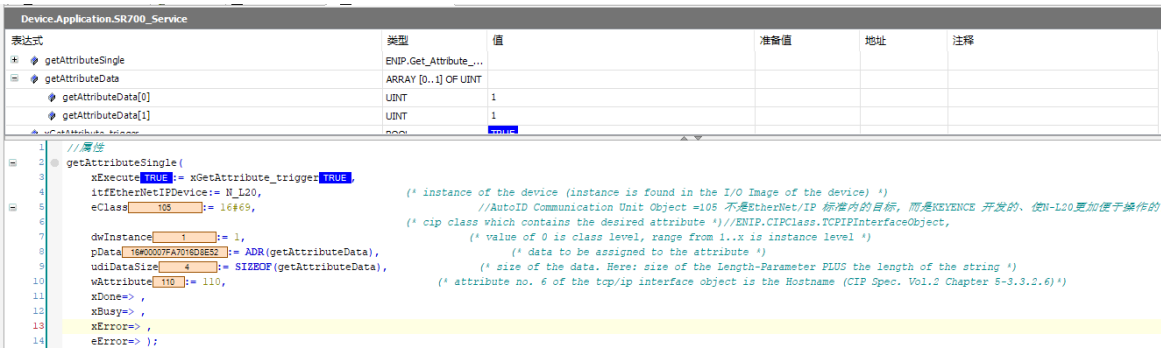
实例 ID	属性 ID	名称	相应参数	
			数量	说明
1 (0x01)	100 (0x64)	Read Status	UINT	bit0 : Error bit1 : Result Data Available bit2 : Result Data Strobe bit3 至 5 : Reserved bit6 : Buffer Overflow Error bit7 : General Error bit8 : BUSY bit9 至 10 : Reserved bit11 : MODE BUSY bit12 : ERR BUSY bit14 至 15: Reserved
			UINT	bit0: Read Complete
			UINT	Reserved
			UINT	Reserved

实例 ID	属性 ID	名称	相应参数	
			数量	说明
1 (0x01)	108 (0x6C)	IN/OUT Status	UINT	bit0 : IN1 bit1 : IN2 bit2 至 3 : Reserved bit4 : OUT1 bit5 : OUT2 bit6 : OUT3 bit7 : OUT4 bit8 至 15 : Reserved
	110 (0x6E)	Result Data Count	UINT	Result Data Ready Count
			UINT	General Error Code
	111 (0x6F)	General Error Code	UINT	General Error Code
	128 (0x80)	Result Data Ready Count	UINT	Result Data Ready Count
	129 (0x81)	Result Data Update Count	UINT	Result Data Update Count

利用功能块 Get_Attribute_Single（功能块如下图），可以实现上表中的相应参数数据。



通过实例化 Get_Attribute_Single 功能块，获取返回结果次数 Result Data Count （属性为 0x6E）。实例代码如下。其中 AutoID Communication Unit Object =16#69，不是 EtherNet/IP 标准内的目标 , 而是 KEYENCE 开发的使 N-L20 更加便于操作的目标。

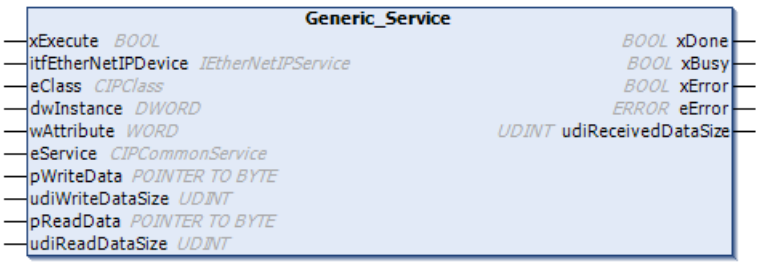


■ Generic_Service

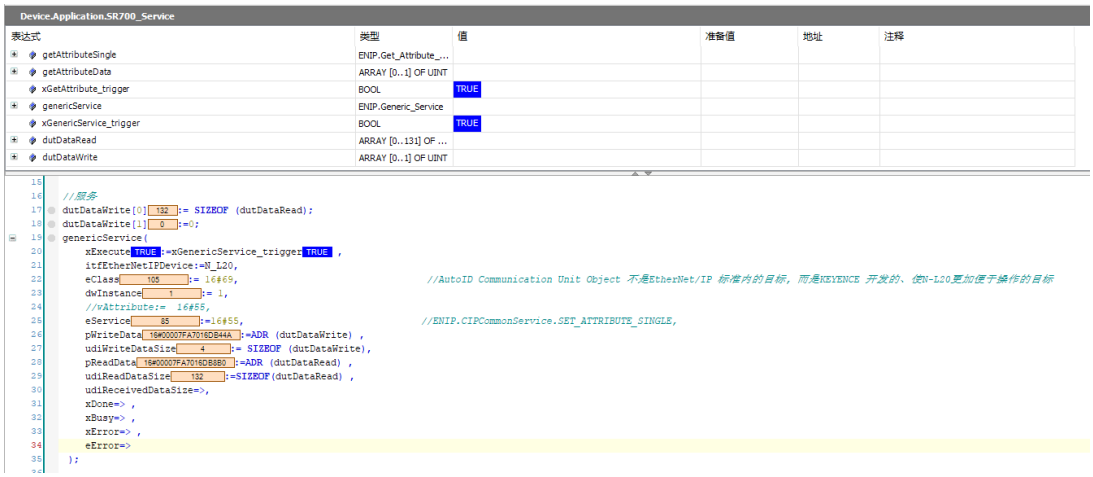
除了公用的服务，还有一些 EtherNet/IP 设备厂商提供的特殊服务类型，例如基恩士 NL-20 模块，提供了如下服务代码用于获取设备特殊的交互数据。

实例 ID	服务代码	服务数据 数据类型：数据	名称	说明
1 (0x01)	14 (0x0E)	-	Get_Attribute _Single	获取属性的一个项目。
	16 (0x10)	-		获取属性的一个项目。
	75 (0x4B)	UINT: Bank Number	Read Start	开始读取。
	76 (0x4C)	-	Read Stop	停止读取。
	83 (0x53)	-	Error Clear	清除错误。
	85 (0x55)	UINT: Result Data Size UINT: Offset	Get Result Data	获取读取数据。 响应数据 UINT : 结果数据大小 UINT : 缓存到 N-L20 的剩余数据大小 BYTE[] : 结果数据
	86 (0x56)	-	Sequence Reset	清除以下信息： Result Data Ready Count Result Data Update Count Main unit statistical information Buffering data Sequence bit
	90 (0x5A)	-	Read Status Clear	读取完成通知和读取失败通知的清除

可通过功能块 Generic_Service（功能块如下图），获取以上服务数据。通过提供详细的 ClassID、实例 ID、服务代码等，获取指定厂商提供的接口数据。



通过服务码 85（0x55）来读取扫码器的数据，示例代码如下。



3.9.4 PLC 作为 EtherNet/IP 从站的配置

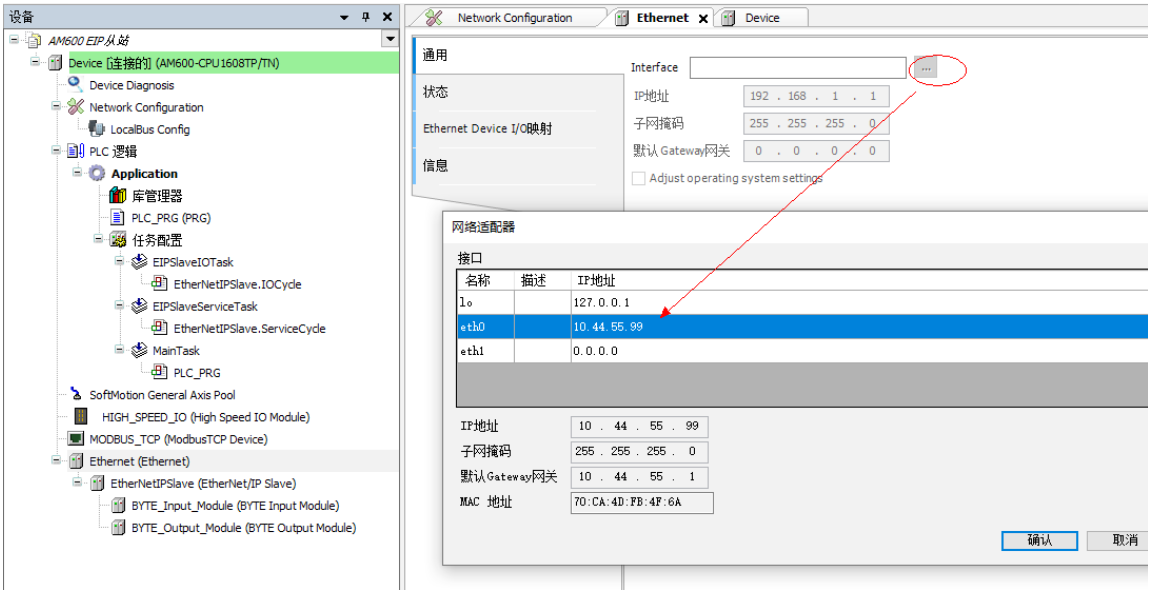
汇川 AM600 系列中型 PLC 支持 EtherNet/IP 从站功能，相关通讯配置步骤说明如下。

1 勾选 EtherNet/IP 从站

鼠标单击主站设备，在弹出的界面中，勾选“EtherNet/IP 从站”。

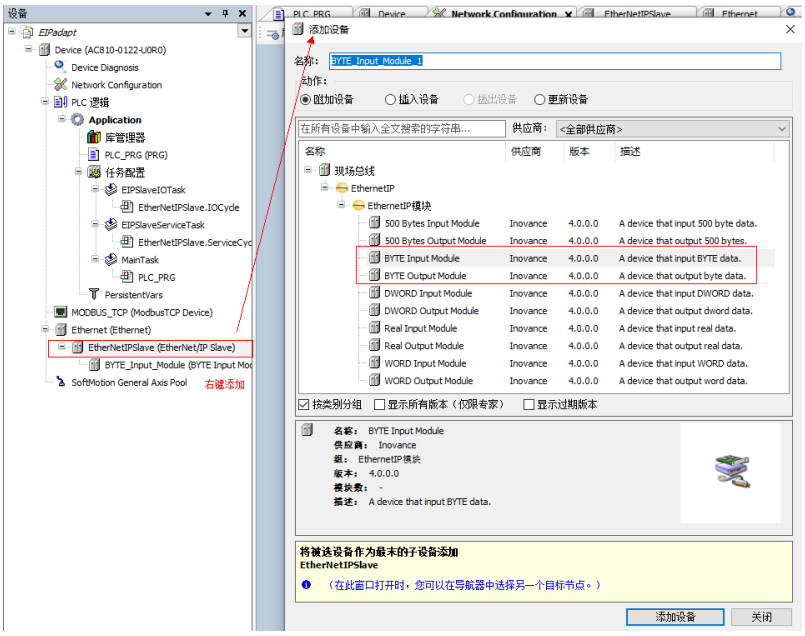
2 设置从站 IP 地址

后台登录 PLC 网关，选择 Ethernet 模块，双击进入通用界面，手动设置从站的 IP 地址。该 IP 地址设置是实际连接到 EtherNet/IP 主站设备的 IP 地址。

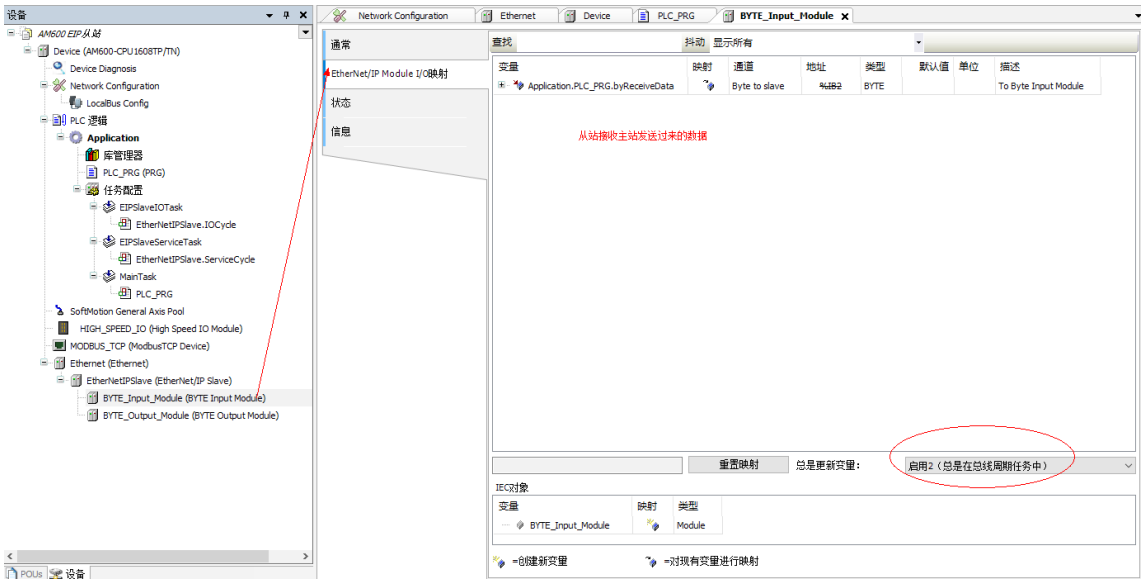


3 添加输入输出模块

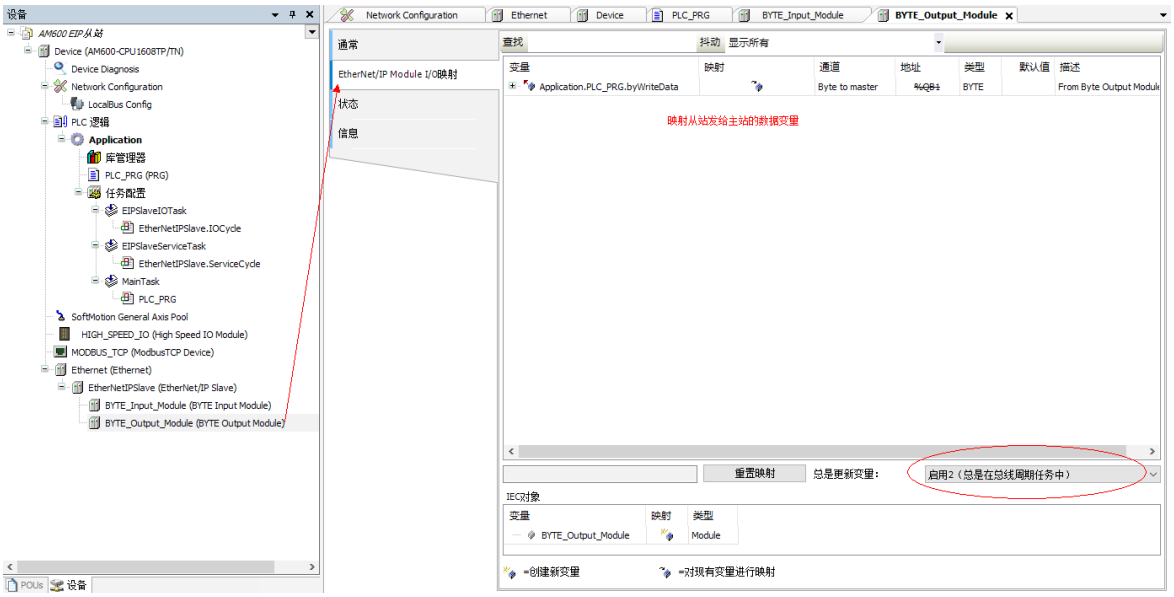
选择 EtherNetIPSalve，右键添加设备，添加从站输入输出模块。用户可以根据数据参数传输需要自由组合模块。本文以 BYTE Input Module 和 BYTE Output Module 输入输出模块为例。



添加之后，PLC 自动分配数据类型为 BYTE 的输入输出映射，当 PLC 的 EtherNet/IP 从站收到数据，则在 InputData 中映射变量刷新，如下图 byReceiveData，数据类型为 BYTE。

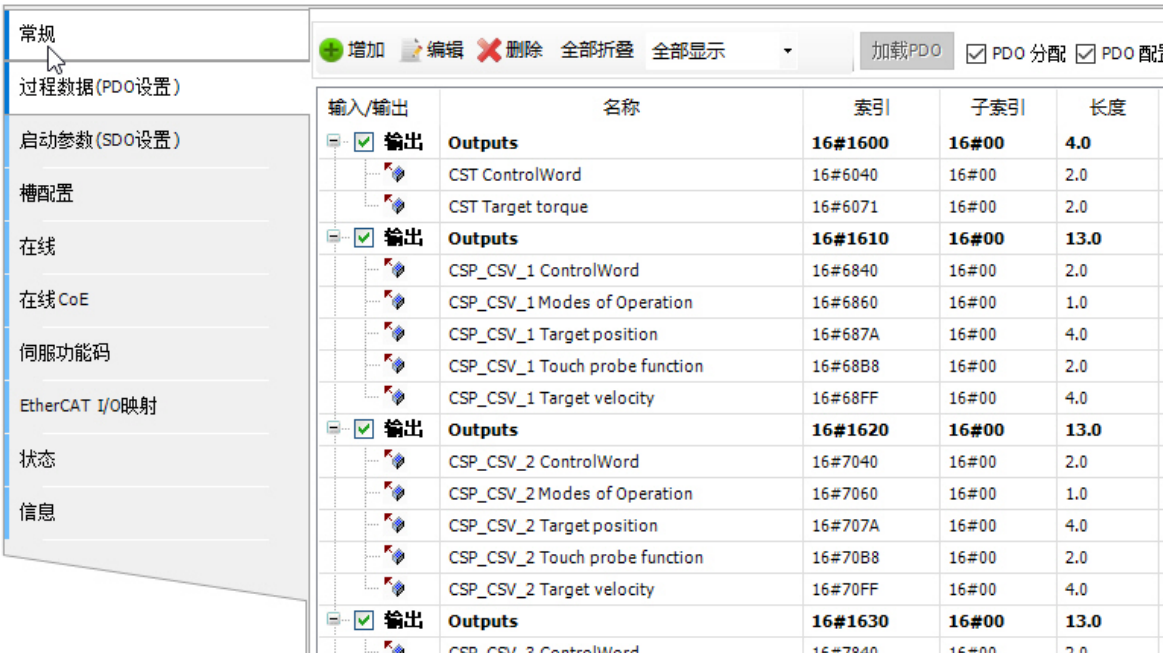


当 PLC 的 EtherNet/IP 从站发送数据，则在 OutputData 中映射变量将数据写入，如下图 byWriteData，数据类型为 BYTE。



NOTE

勾选从站配置后，后台会自动生成任务配置，建议设置从站工程中的 EIPMasterIOTask 任务周期与主站的 RPI 时间一致，以保证数据传输同步。



输入/输出	名称	索引	子索引	长度
输出	**Outputs**	**16#1600**	**16#00**	**4.0**
	CST ControlWord	16#6040	16#00	2.0
	CST Target torque	16#6071	16#00	2.0
输出	**Outputs**	**16#1610**	**16#00**	**13.0**
	CSP_CSV_1 ControlWord	16#6840	16#00	2.0
	CSP_CSV_1 Modes of Operation	16#6860	16#00	1.0
	CSP_CSV_1 Target position	16#687A	16#00	4.0
	CSP_CSV_1 Touch probe function	16#68B8	16#00	2.0
	CSP_CSV_1 Target velocity	16#68FF	16#00	4.0
输出	**Outputs**	**16#1620**	**16#00**	**13.0**
	CSP_CSV_2 ControlWord	16#7040	16#00	2.0
	CSP_CSV_2 Modes of Operation	16#7060	16#00	1.0
	CSP_CSV_2 Target position	16#707A	16#00	4.0
	CSP_CSV_2 Touch probe function	16#70B8	16#00	2.0
	CSP_CSV_2 Target velocity	16#70FF	16#00	4.0
输出	**Outputs**	**16#1630**	**16#00**	**13.0**

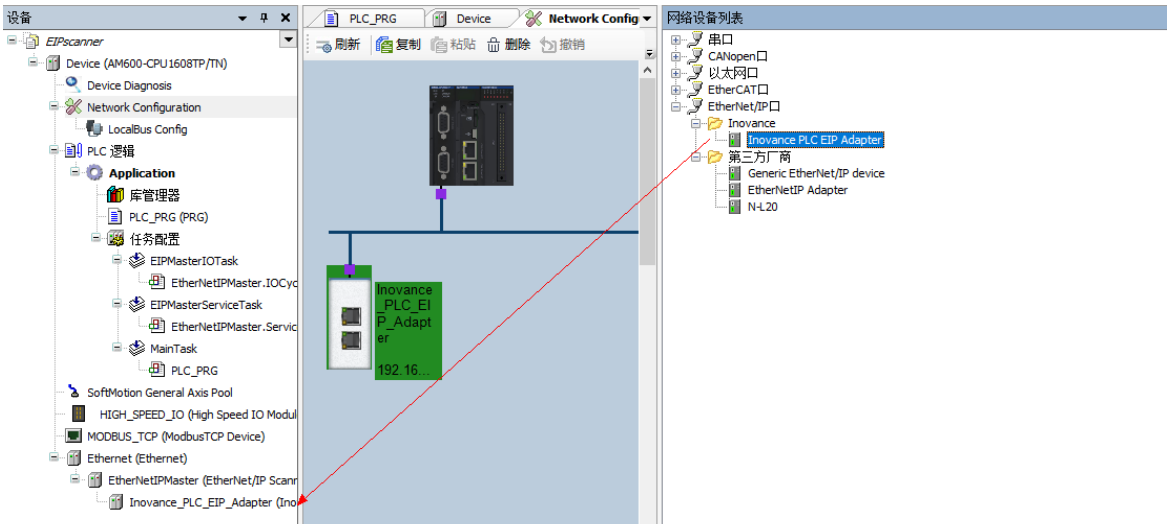
3.9.5 PLC 作为 EtherNet/IP 从站的配置例程

按照上述小节的 EtherNet/IP 从站配置步骤创建从站例程，分别同汇川 AM600 EtherNet/IP 主站工程和欧姆龙 NJ501 EtherNet/IP 主站工程进行通讯连接测试。

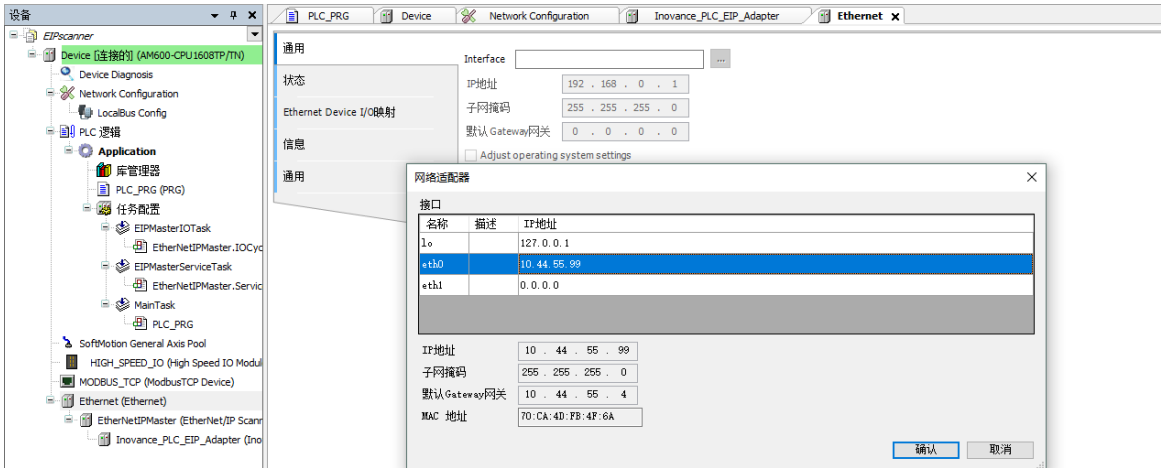
1 从站工程与 AM600 主站通讯

■ PLC 为主站工程配置

1） 本工程以 AM600 为主站，进行 EtherNet/IP 网络组态搭建。可以直接在网络设备列表中选择汇川 PLC 从站的 EtherNet/IP 适配器。

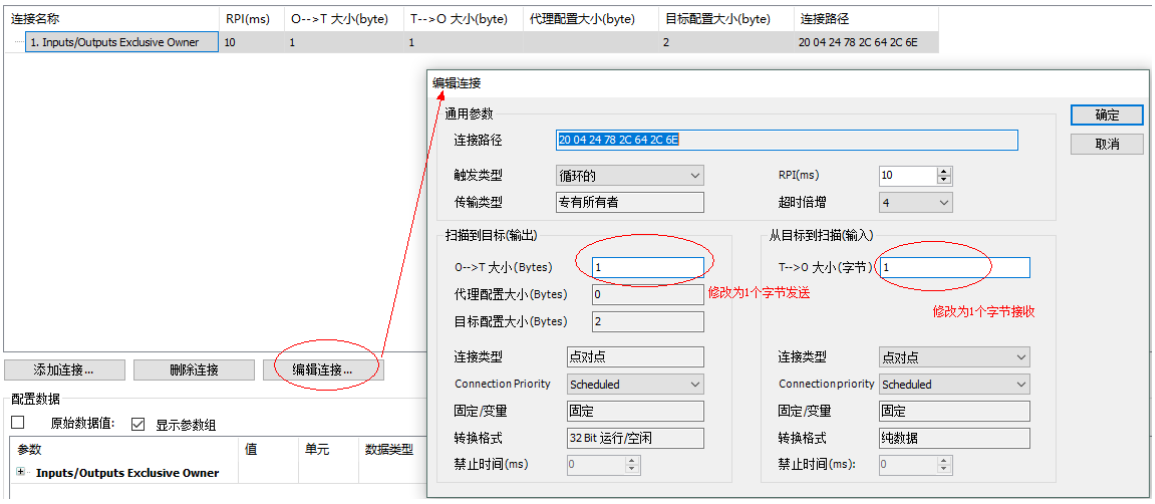


2） 设置主站的 EtherNet/IP 通讯 IP 地址，以及需连接的从站模块 IP。

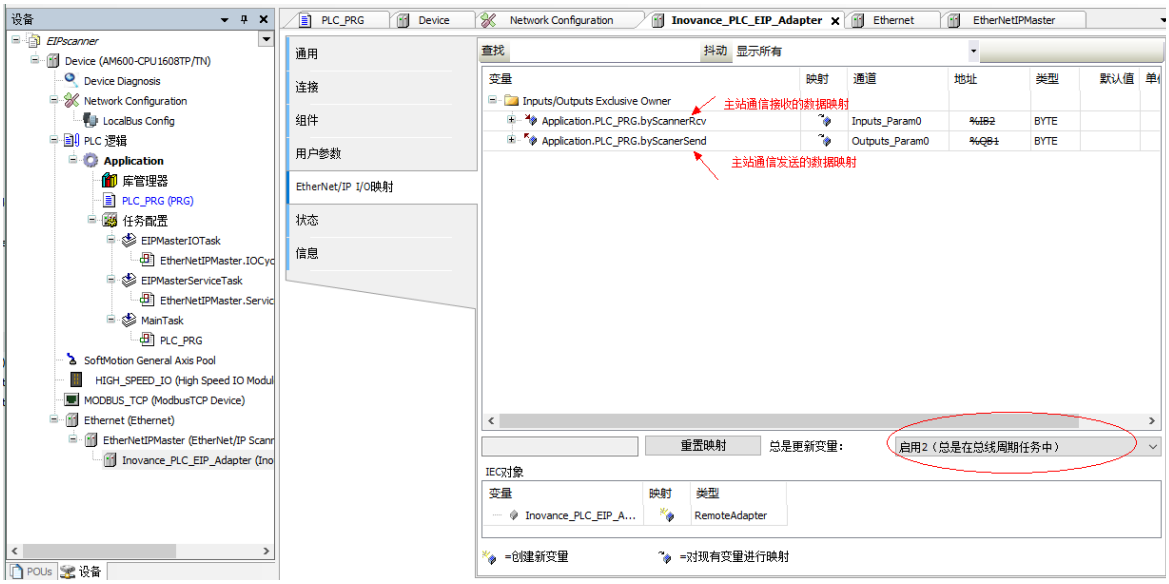


3） 设置从站 IP 地址。

4） 修改默认连接。从站模块中默认一个连接类型为发送和接收都为 4 个字节，从站的模块实际上为 1 个字节接收和 1 个字节输入。



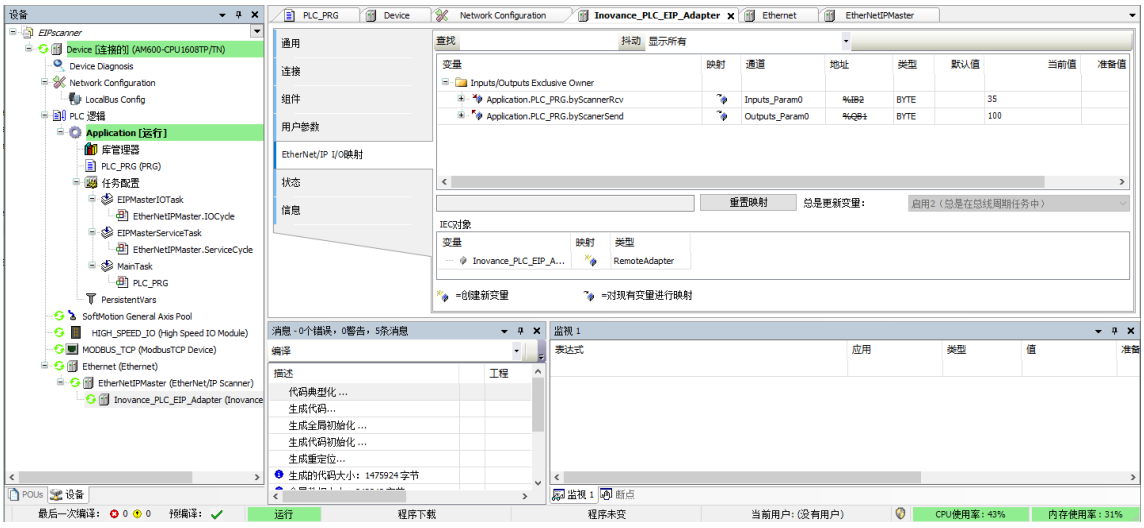
添加映射变量。在 EtherNet/IP 界面分别添加主站通讯接收和发送的映射变量，类型为 BYTE。



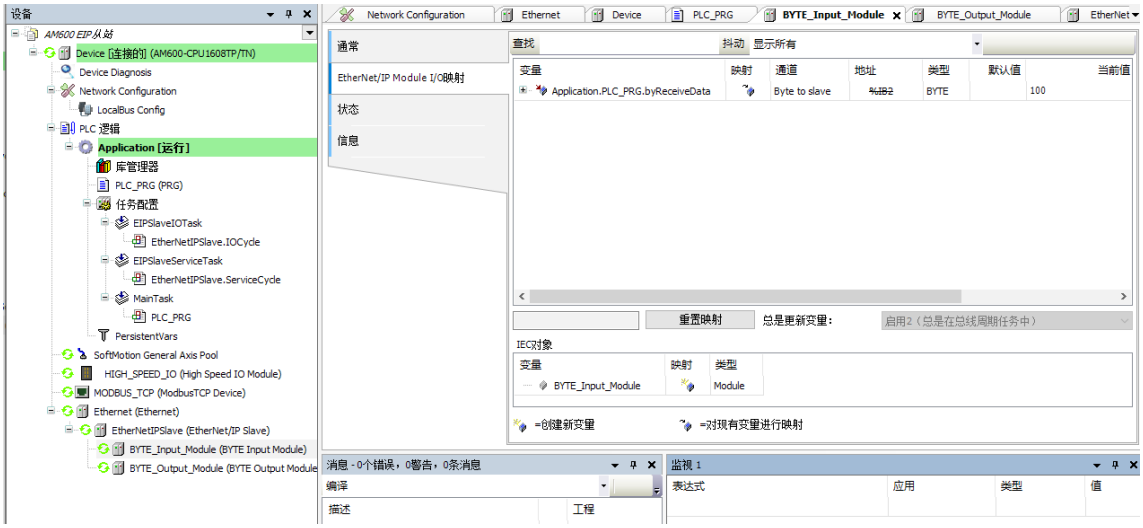
2 EtherNet/IP 主从通讯功能测试

■ 分别登录下载，启动主站工程和从站工程。

如下图所示，主站工程和从站工程分别登录启动，通讯连接正常。在主站工程从站发送的变量 byWriteData 初始值为 35，在主站中接收数据映射为 byScannerRcv 初始值默认为 0，现在已更新为 35，说明从站的数据成功发送到主站。



如下图所示，创建从站工程，登录启动，确认通讯正常连接。主站发送的变量 byScanerSend 默认为 100，在从站输入模块的映射变量 byReceiveData，初始默认为 0，已经更新为 100，说明从站成功接收到主站发送的数据。



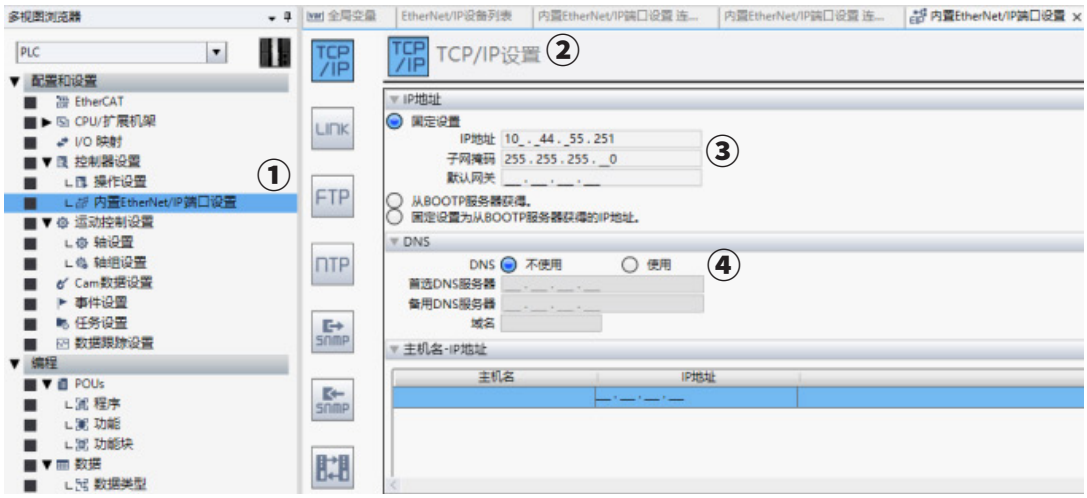
3 从站工程与 NJ501 主站通讯

测试欧姆龙 PLC EtherNet/IP 为主站和汇川 AM600 从站工程进行通讯。

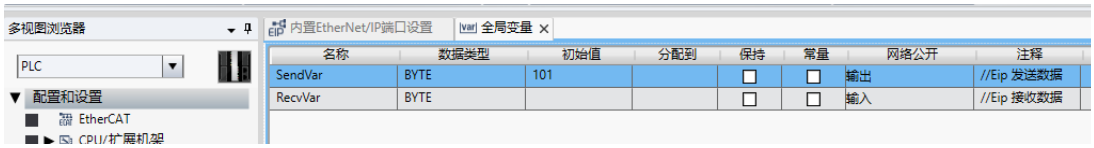
■ NJ501 EtherNet/IP 主站工程配置

1） 新建欧姆龙 NJ501 标准工程。

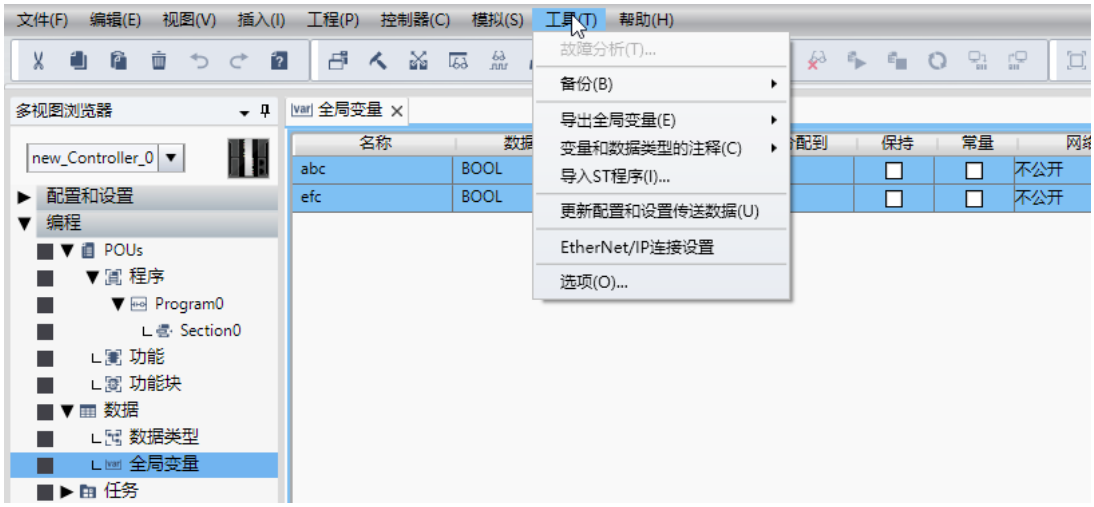
2） 配置和设置中，选择内置 EtherNet/IP 端口设置，TCP/IP 设置中设置主站 EtherNet/IP 通讯端口 IP 地址，确保该 IP 地址和汇川 AM600 从站属于同一网段。



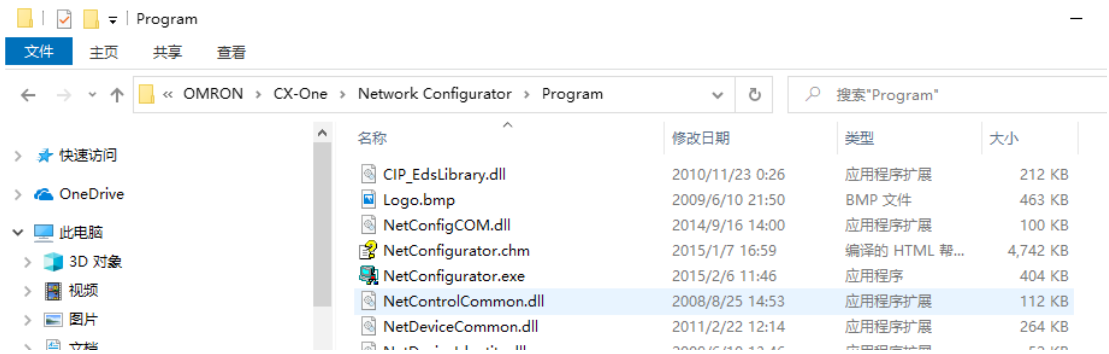
3） 新建全局变量。



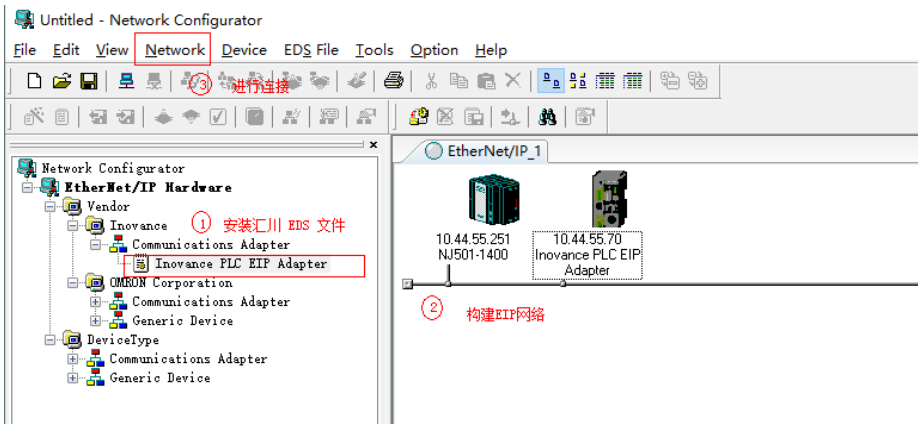
4） 导出全局变量，注册标签。先选中如图列表新建的所有全局变量，然后在工具 -> 导出全局变量中，选择 NetworkConfigurator 工具，导出 csv 格式数据。



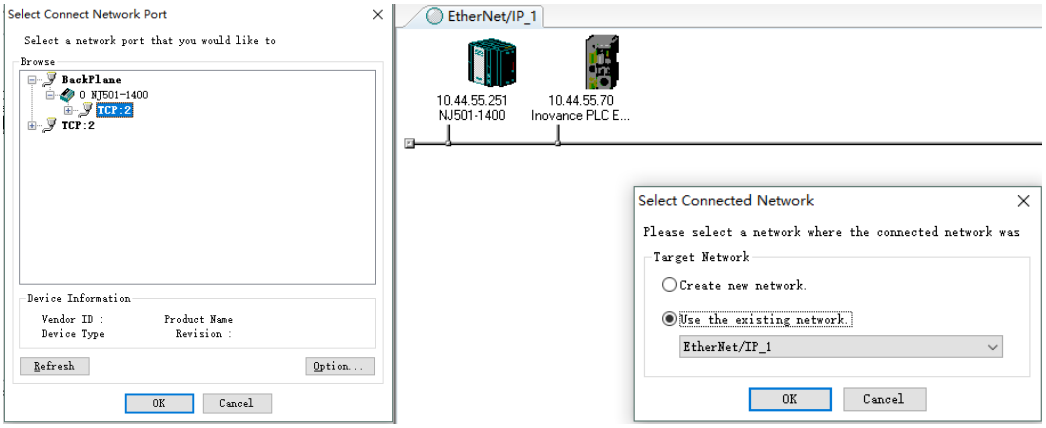
5） 通过 NetworkConfigurator 工具对标签数据进行下载注册。在 sysmac studio 的安装路径打开该网络配置软件。



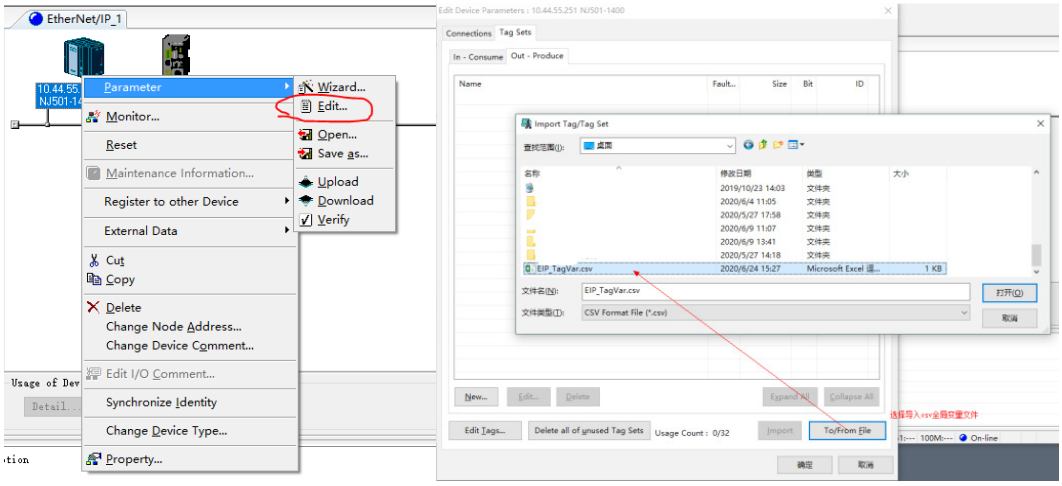
6） 按照如下步骤设置，导入汇川 eds 文件，然后构建 EtherNet/IP 网络，分别修改默认 IP 地址，和 PLC 中设置的相一致。



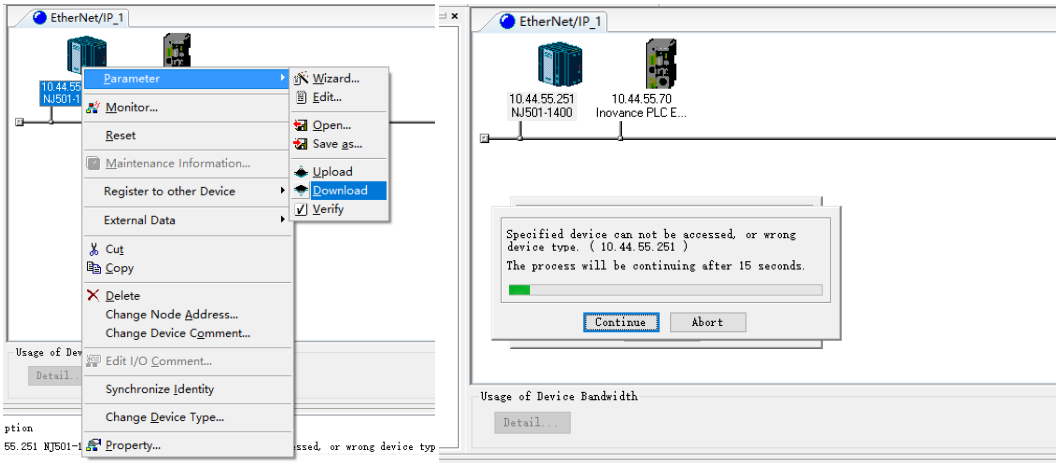
7） 连接到 PLC 设备。



8） 导入之前保存的 csv 格式的全局变量文件。

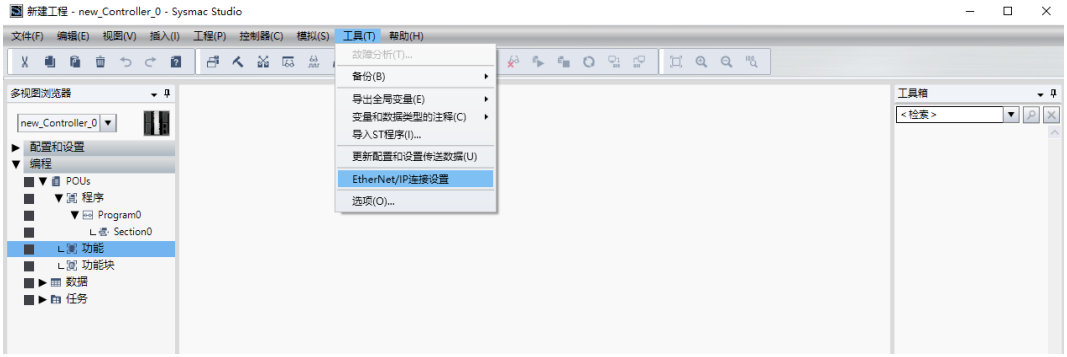


9） 下载完成即可。

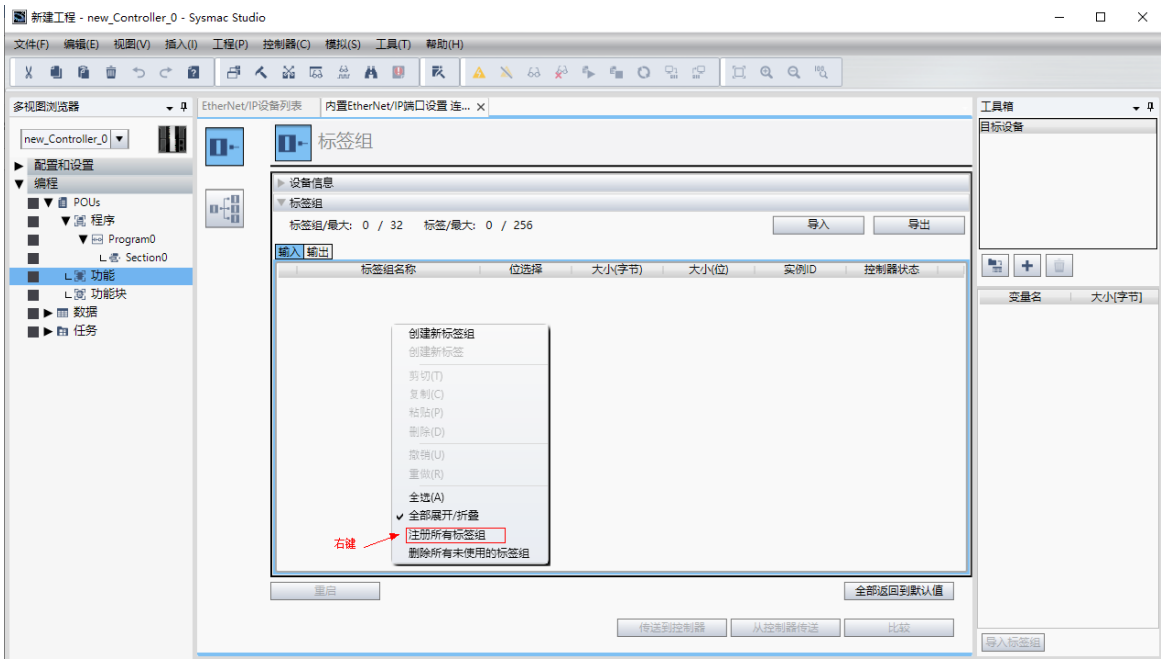


10） 设置标签通讯。

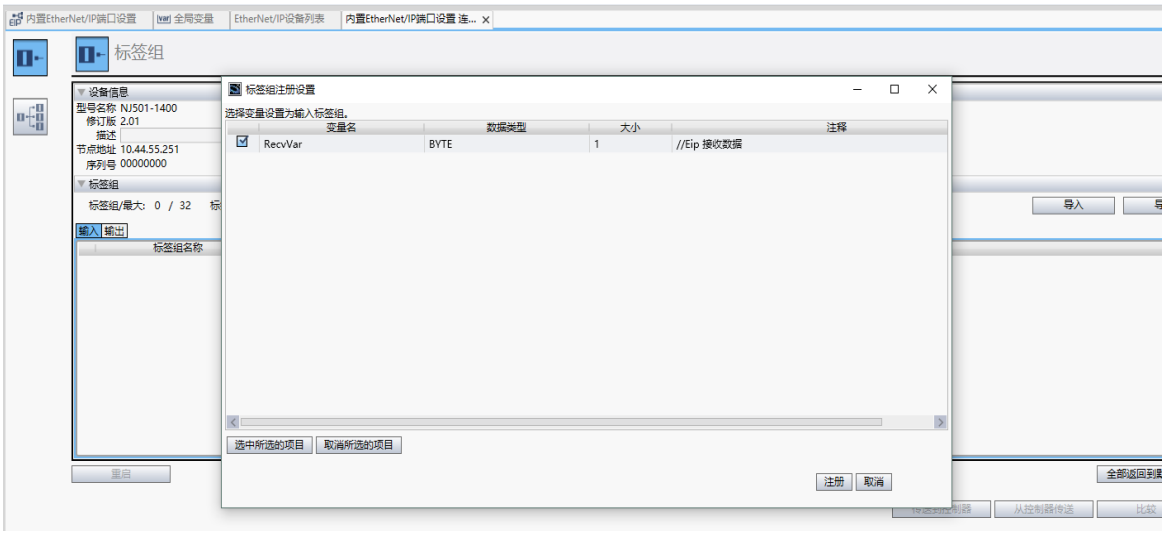
首先打开“工具 ->EtherNet/IP 连接设置” 。



然后进入 EtherNet/IP 端口设置界面，最后在如下标签组的列表中选择右键 -> 注册所有标签组。



在标签组注册界面，此时经过 NetworkConfigurator 下载的标签变量已经注册自动加载了。选中所选的项目，点击“注册”即可。

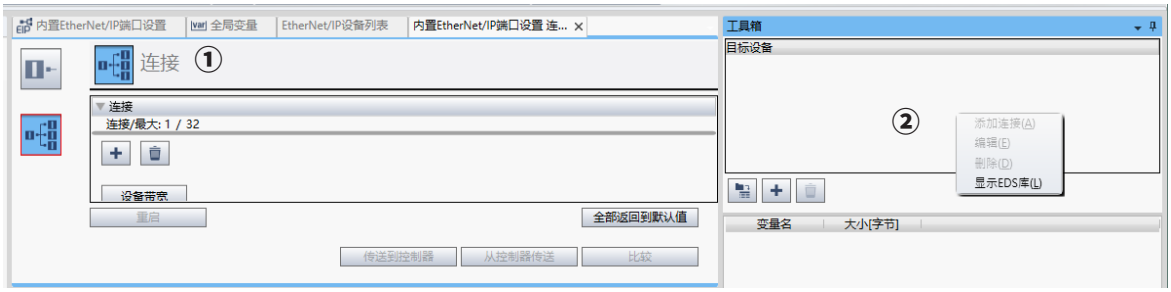


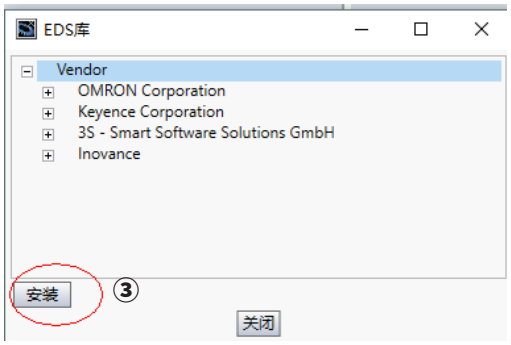
注册完后，输入输出列表显示的注册变量如下图所示。

11） 设置连接路径。

a. 导入汇川 eds 文件。

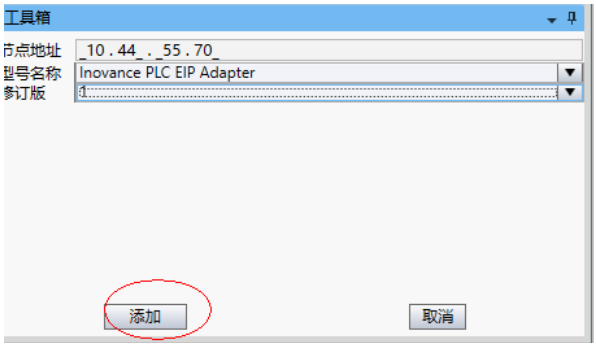
在连接界面工具箱中，右键 -->“显示 EDS 库（L）”，在弹出的 EDS 库目录列表中点击“安装”，选择存放汇川 eds 文件的路径进行安装。





b. 添加从站设备。

安装完成后，在工具箱列表下方选择“添加目标设备 "，在型号名称下拉列表中即可找到刚才导入 EDS 库中的汇川 EtherNet/IP 从站设备名，选中即可。再设置连接到汇川 EtherNet/IP 从站设备的节点 IP 地址和修订版本（该 IP 地址应和欧姆龙 PLC 主站在同一网段）， 最后点击“添加”即可。

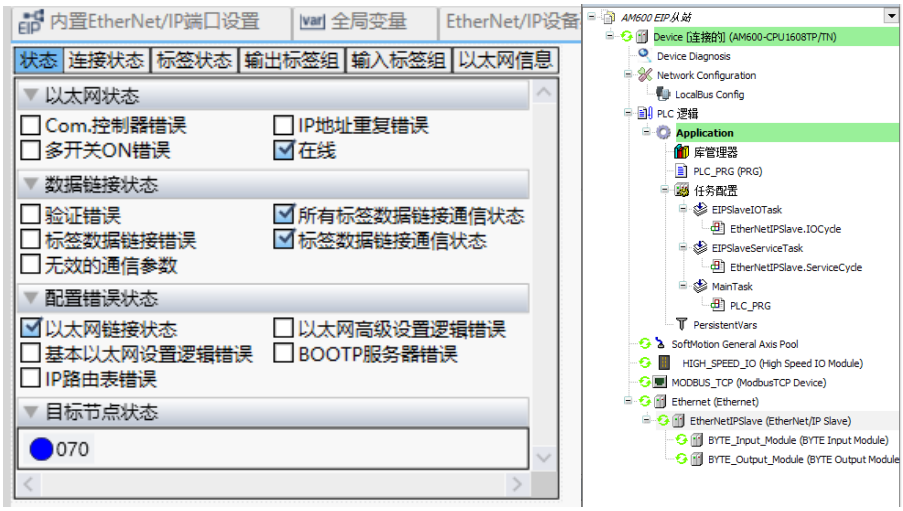


c. 设置连接。

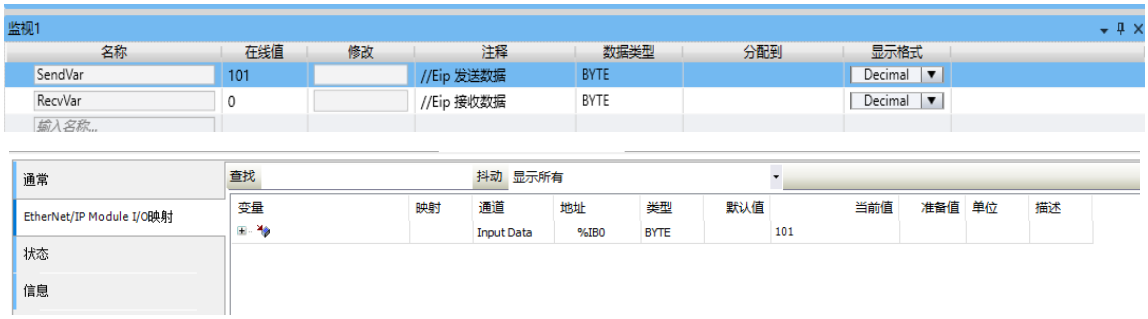
如下图所示，设置目标设备、连接类型和输入输出变量等参数。

4 EtherNet/IP 主从通讯功能测试

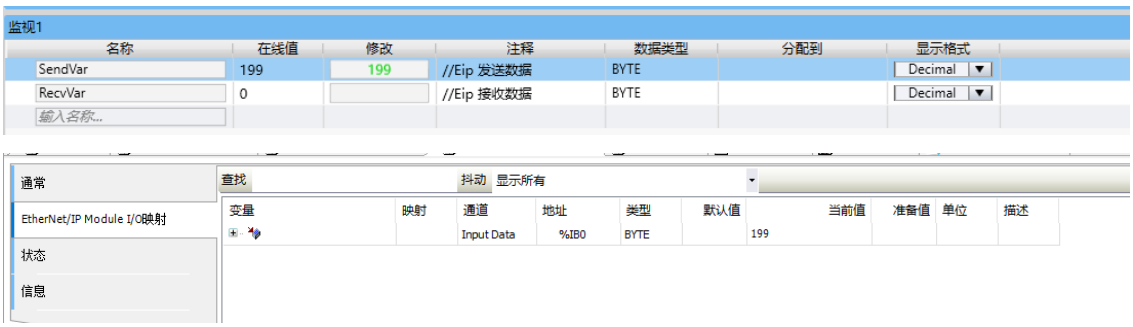
通过以上设置后，将欧姆龙 PLC 工程进行编译，同步后运行，分别查看状态，确认主站和从站通讯正常。如下图所示。



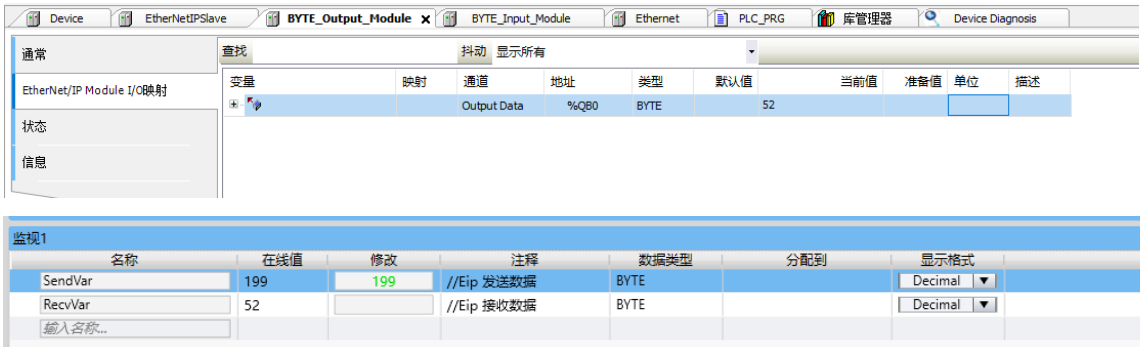
修改 NJ501 EtherNet/IP 标签通讯 SendVar 的值，查看 AM600 EtherNet/IP 从站接收变量的变化，如果能跟随修改值变化，表示通讯测试正常。如下图。



SendVar 在线修改为 199，对应 AM600 映射变量也更新为 199。



修改 AM600 EtherNet/IP 从站 Output 变量发送值为 52，NJ501 EtherNet/IP 通讯 RecvVar 值也更新为 52。



3.10 Profibus-DP 总线

总线简介

PROFIBUS 是一种国际化，开放式，不依赖于设备生产商的现场总线标准。广泛适用于制造业自动化，流程工业自动化和楼宇、交通、电力等其它领域自动化，1996 年成为欧洲工业标准，1999 年成为国际标准，2001 年被批准成为中华人民共和国工业自动化领域行业标准中唯一的现场总线标准。

PROFIBUS 利用了现有国际标准，它的协议结构以国际标准 OSI 系统互连模型为基础，如图 3-82 所示，因此符合了开放性和标准化的要求。在这个模型中，每个传输精确处理所定义的任务。第一层物理层定义了物理的传输特性，第二层数据链路层定义了总线的存取协议，第七层应用层定义了应用功能。

PROFIBUS-DP 使用了第一层、二层和用户接口，第三层到第七层未加描述，这种流体结构确保了数据传输的快速和有效、直接数据链路映像 DDLM（direct data link mapper）提供了易于进入第二层的服务用户接口，该用户接口规定了用户与系统以及不同设备可调用的应用功能，并详细说明了各种不同 PROFIBUS-DP 设备的设备行为，还提供了 RS-485 传输技术或光纤。

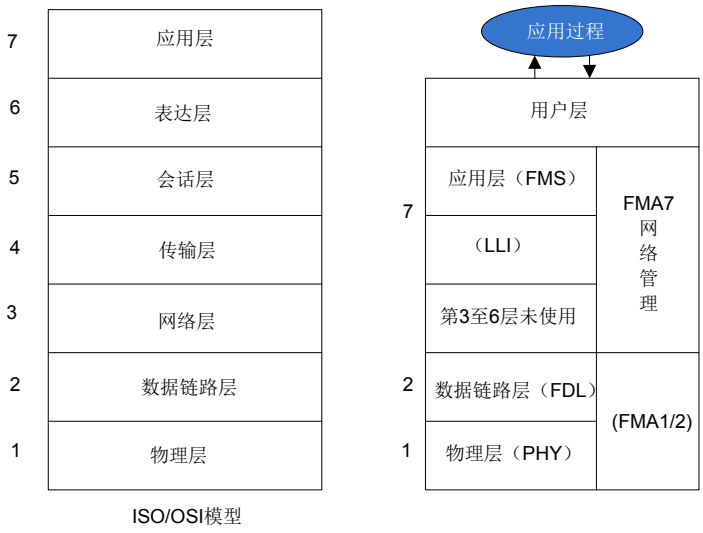


图 3-82 PROFIBUS-DP 现场总线模型

PROFIBUS-DP 用于现场层的高速数据传输，主站周期地读取从站的输入信息并周期地向从站发送输出信息。除周期性用户数据传输外，PROFIBUS-DP 还提供智能化现场设备所需的非周期性通讯以进行组态、诊断和报警处理。

3.10.1 Profibus-DP 使用的一般过程

Profibus-DP 一般使用流程如下：

1） 设计 Profibus-DP 硬件网络结构。

2） 在网络组态中激活 Profibus-DP 总线。激活总线后，会自动添加 Profibus-DP 主站。

3） 根据硬件结构在网络组态中添加 Profibus DP 从站及模块。如果是第三方从站，可以在网络组态中通过导入 GSD 文件导入第三方从站，然后添加第三方从站；如果是 AM600 从站需要在硬件组态中添加 I/O 模块。Profibus-DP 从站指 DP 远程设备。

4） 设置合适的主站配置参数、从站配置参数及模块配置参数。一般情况下，从站节点 ID 自动生成，IO 映射根据 GSD 文件会自动生成，一些特殊的配置需要手动修改。

配置主站参数及从站参数时，主站波特率与从站的波特率是自适应的，所配置的从站的节点 ID 需要和实际的从站节点 ID 拨码开关匹配。

3.10.2 Profibus-DP 主站配置

1 主站参数配置

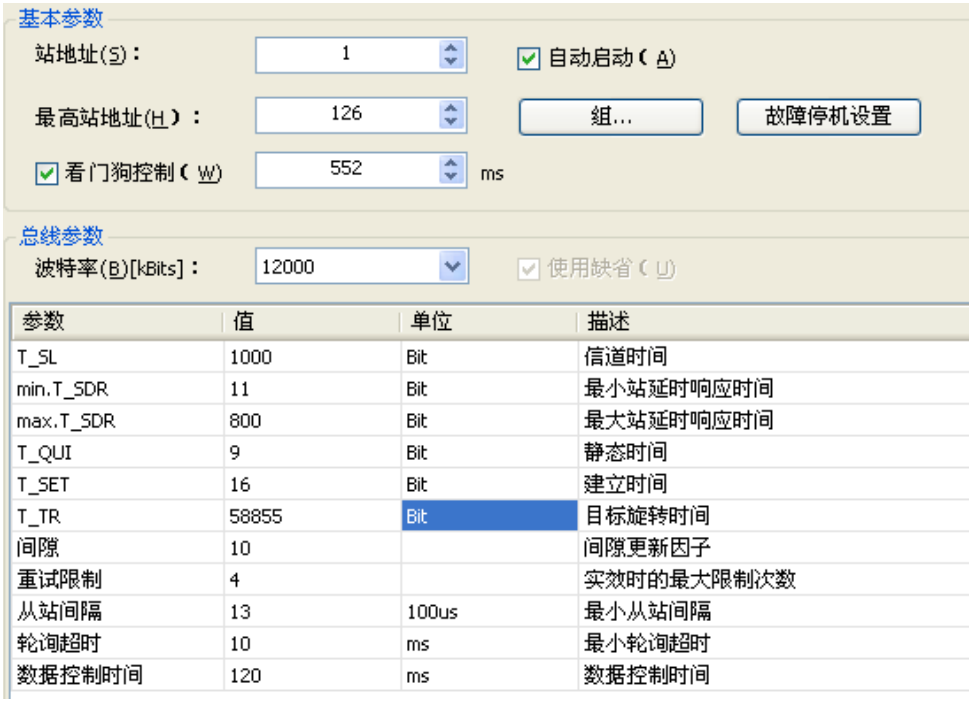


图 3-83 Profibus-DP 主站配置界面

基本参数

- 节点 ID：主站在 Profibus-DP 网络唯一标示号，默认 1，范围 1-126，必须是十进制数。
- 最高站地址：令牌帧传递的最大站地址，默认 126。
- 看门狗控制：传递给从站的看门狗时间，用来确认从站与主站之间的连接。

总线参数

- 波特率：总线上用于传输的波特率。单位 Kbit/s, 可以设置以下的波特率：9.6, 19.2, 45.45, 93.75, 187.5, 500, 1500, 3000,6000 及 12000。默认值 1500。

NOTE

根据不同的通讯距离及通讯的站点个数，要设置合理的波特率，只有在总线配置文件参数相互匹配时，PROFIBUS 子网才能正常工作。因此，只有在熟悉 PROFIBUS 总线配置文件的参数分配时，才能更改缺省值，建议用户使用默认的总线参数。

2 故障停机设置

故障停机功能是指当从站或者模块出现故障或者组态不一致时，从站是否停止运行。此设置为故障停机

开关，此功能只支持 AM600 Profibus-DP 从站。

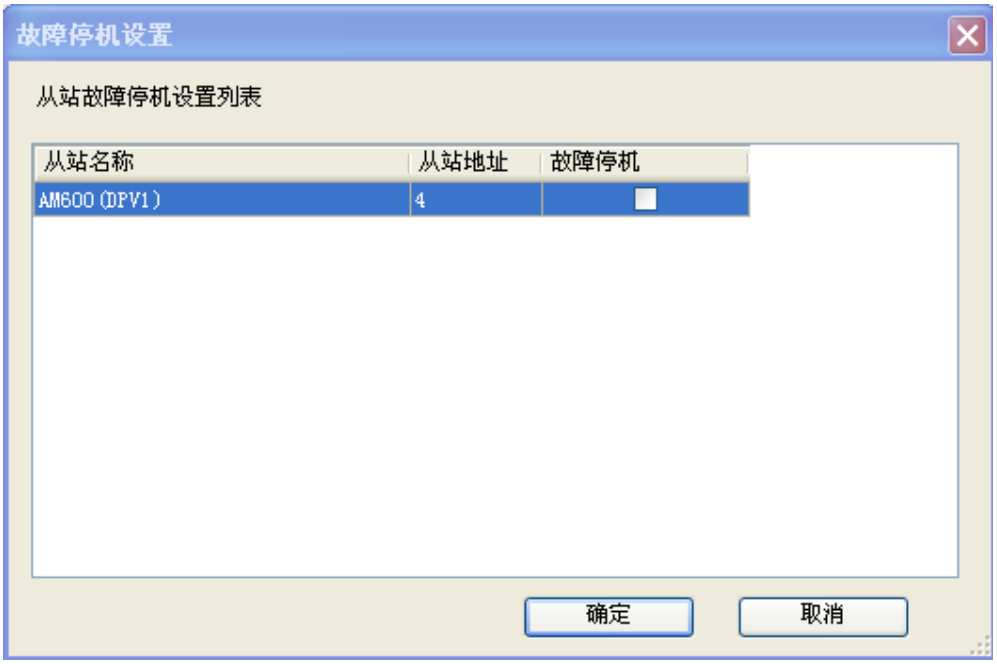


图 3-84 故障停机设置

■ 从站故障停机设置列表：显示并设置从站故障或者组态不一致时是否停机。

“故障停机”列控制当从站或者模块出现故障时从站是否停机。如果激活了故障停机功能，如果从站本身出现故障，从站停止运行；如果 I/O 模块本身的诊断上报功能激活并且模块出现故障，从站停止运行；

■ 确定 / 取消：保存 / 取消保存故障停机设置。

3 Profibus-DP 主站 I/O 映射

I/O 映射的总体描述和相关对话框的使用请参照 I/O 映射链接。

4 状态

DP 总线设备或模块的状态配置编辑器可显示状态信息（如：“运行”，“停止”）和内部总线系统的状态。

5 信息

如果当前设备可用，将会显示下列基本信息：名称、供应商、类型、版本号，模块号、描述。

3.10.3 Profibus-DP 从站配置

Profibus-DP 从站配置主要是配置从站基本参数及 GSD 中声明的从站参数。

1 从站参数配置

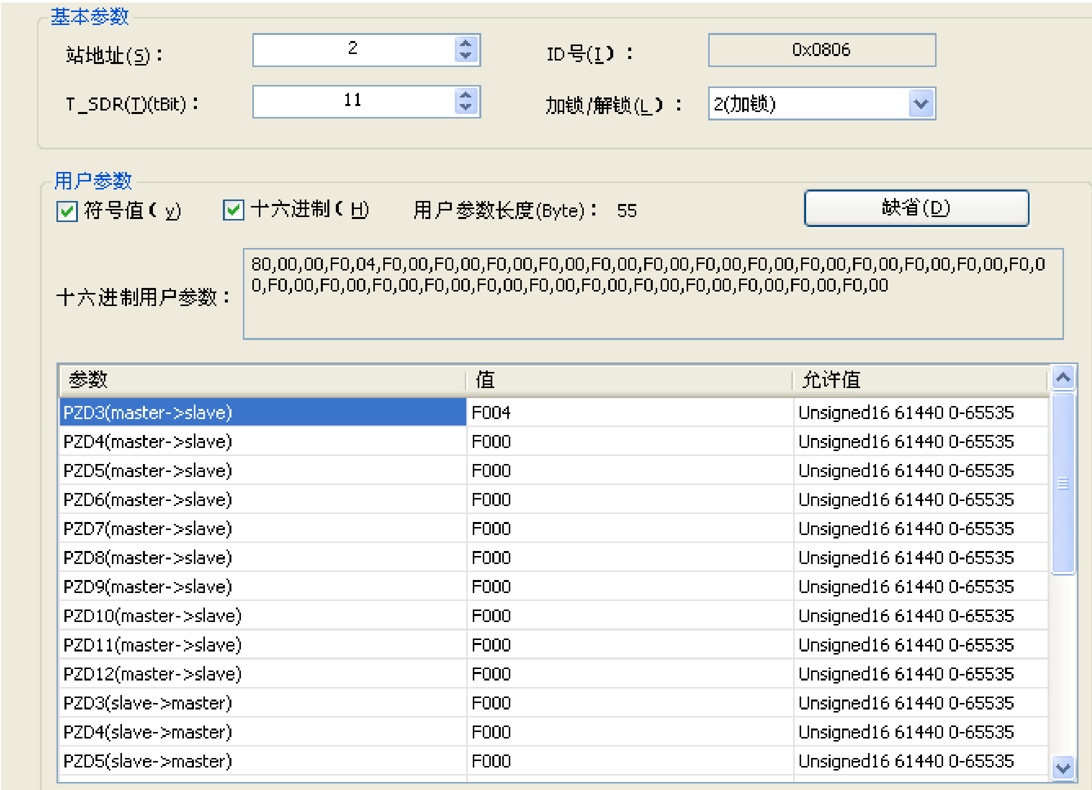


图 3-85 Profibus-DP 从站参数配置

基本参数

- 节点 ID：从站在 DP 网络唯一标示号，范围 1-125，默认 2，，必须是十进制数，要求与从站本身标示号（如拨码开关）一致。
- ID 号：是从站设备唯一的识别符，由 GSD 文件决定。
- T_SDR：从站的最小响应间隔，指示从站在收到主站数据后的最小回复间隔。
- 加锁 / 解锁：指示从站当前状态。

用户参数

用户参数是由 GSD 文件定义的，可以选择 10 进制或者 16 进制显示，具体含义请参考设备商提供的手册。

2 从站诊断

诊断指示了从站节点的运行状态

诊断的解析：

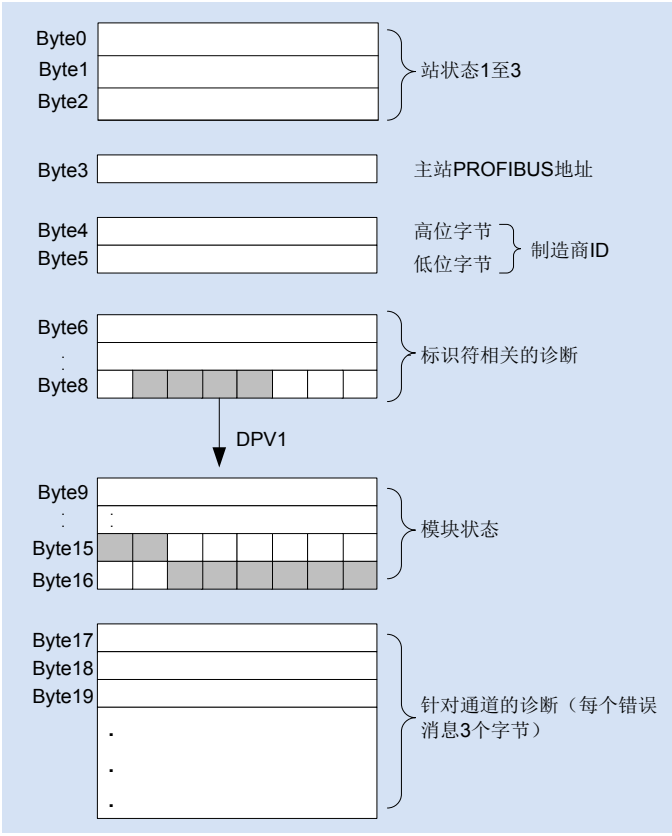


图 3-86 诊断报文结构

表 3-6 站状态 1 的含义

位	含义		原因 / 解决方法
0	0:	DP 主站无法寻址 DP 从站，DP 从站中该位始终为 0。	DP 从站上设置的 PROFIBUS 地址是否正确? 总线连接器与 FOC 是否已连接? DP 从站的电压是多少? RS485 中继器的设置是否正确? 是否复位了 DP 从站（打开 / 关闭）?

位	含义		原因 / 解决方法
1	1:	DP 从站尚未准备好进行交换数据。	请等待 DP 从站完成启动。
2	1:	DP 主站发送到 DP 从站的组态数据与 DP 从站的实际组态不匹配。	在组态软件中输入的站类型或 DP 从站的组态是否正确?
3	1:	外部诊断可用。	评估标识符相关的诊断、模块状态和 / 或通道相关的诊断。纠正了所有错误后，位 3 即会复位。 在上面所示的诊断字节中，出现新的诊断消息时该位复位。
4	1:	DP 从站不支持请求的功能。	检查组态。
5	1:	DP 主站无法解释 DP 从站的响应。	检查总线组态。
6	1:	DP 从站类型与软件组态不相符。	站类型的组态软件设置是否正确?
7	1:	其他 DP 主站（不是当前访问 DP 从站的 DP 主站）已组态 DP 从站。	在通过编程设备或其他 DP 主站访问 DP 从站的情况下，该位始终为 1. 组态 DP 从站的 DP 主站的 PROFIBUS 地址位于“主站 PROFIBUS 地址”诊断字节中。

表 3-7 站状态 2 的含义

位	含义	
0	1:	必须重新组态 DP 从站。
1	1:	从站处于启动阶段。
2	1:	DP 从站中该位始终为“1”。
3	1:	已为此 DP 从站启用响应监视。
4	1:	DP 从站已接收到“FREEZE”控制命令。
5	1:	DP 从站已接收到“SYNC”控制命令。
6	0:	该位始终为 0。
7	1:	DP 从站被取消激活，即已将其从当前处理中删除。

表 3-8 站状态 3 的含义

位	含义	
0~6	0:	该位始终为“0”。
7	1:	特定于通道的诊断消息数量超过诊断帧中可以容纳表示的消息。

- 主站 Profibus 地址：标明该 Profibus-DP 主站已组态 DP 从站并且对该 DP 从站有读写访问权限。值为 FF，则 DP 主站尚未组态 DP 从站。
- 制造商 ID：描述 DP 从站类型，由设备商声明，在 GSD 中体现。

3.10.4 Profibus-DP 模块

1 模块化设备和非模块化设备

在 Profibus-DP 从站配置中，DP 从站节点可接入两种类型的模块设备：

模块化设备：接入 DP 从站节点下，模块本身自带 I/O 映射列表，不需要使用 Profibus-DP 从站 I/O 映射对话框，从站节点数据随模块自动添加。目前 AM600 I/O 模块为此类型。

非模块化设备：从站节点对话框包含 I/O 映射对话框，数据不能自动配置。

2 AM600 Profibus-DP I/O 模块

AM600 Profibus-DP I/O 模块在硬件组态中添加，添加模块后可以配置相关 I/O 参数，也可以添加 I/O 映射实现数据刷新，详情请参见 CPU 下的 I/O 模块。

3.11 与 HMI 通信配置

3.11.1 通信配置

本驱动构件通过 InoTouch Editor 软件组态，基于 Modbus TCP/IP 协议读写汇川中型系列 PLC 设备的各种寄存器的数据；

HMI 支持 01、31、03、33、05、35、06、36、0F、3F、10、40 功能码，有关功能码的详细信息，请参见相应 HMI 产品用户手册。

配置项目	说明
通讯协议	采用汇川 AM600 系列 PLC 内置 MODBUS TCP IP 协议
通讯方式	一主一从、一主多从方式。驱动构件为主，设备为从
通讯设备	以太网子设备，须挂接在“通用 TCP IP 父设备”下才能工作

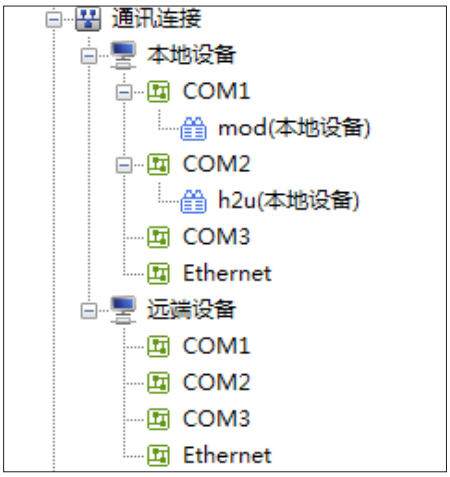
1 硬件连接

InoTouch Editor 软件与设备通讯之前，必须保证硬件通讯连接正确。

连接方式：采用 RJ-45 网线直连 HMI 与 PLC（直通网线或使用 HUB 交换）。

2 设备通讯参数

InoTouchEditor 客户端通讯参数设置如下：



通讯连接中，选择 Ethernet, 增加设备如下：

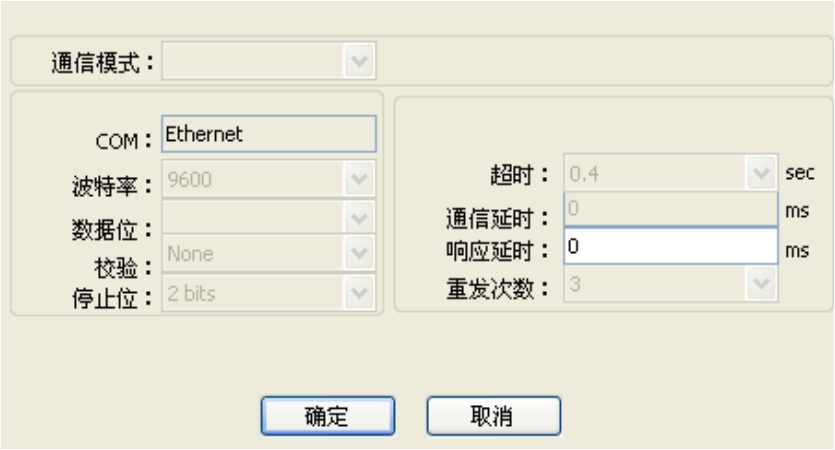
■ PLC IP 地址：本栏需按 PLC 设定实际情况正确填写 PLC IP 地址。

■ 端口号：上位机与下位机数据帧发送 / 接收端口，默认情况下为 502，建议用户不要修改默认设置值。

其他条目采用默认设置。

3 通信设置

“汇川 AM600-ModbusTCP”子设备参数设置如下：



■ 响应延时：发送帧发送后到启动接收的时间间隔，默认为 0ms。

4 采集通道

通讯状态

通讯状态值	意义
通讯正常	表示当前通讯正常
命令失败	表示设备命令读写操作失败错误
校验失败	表示采集数据校验错误
通讯超时	表示采集无数据返回错误

5 内部属性

用户可通过内部属性添加通道，本驱动构件可支持汇川 AM600 系列 PLC 内置 ModbusTCP 寄存器类型及对应功能码如下：

寄存器	数据类型	读取功能码	写入功能码	操作方式	通道示例
[SM 区] 输入线圈	BT	31	35、3F	读写	“SM 只读 0000”表示 SM 区地址 0
[Q 区] 输出线圈	BT	01	05、0F	读写	“Q 读写 0001” 表示 Q 区地址 1
[SD 区] 输入寄存器	16-bit-BCD 32-bit-BCD 16-bit-unsigned 16-bit-signed 32-bit-unsigned 32-bit-unsigned 32-bit-float	33	36、40	读写	“SD 只读 0002” 表示 SD 区地址 2

寄存器	数据类型	读取功能码	写入功能码	操作方式	通道示例
[M 区] 输出寄存器	16-bit-BCD 32-bit-BCD 16-bit-unsigned 16-bit-signed 32-bit-unsigned 32-bit-unsigned 32-bit-float	03	06、10	读写	“M 读写 0003” 表示 M 区地址 3

功能码：[SD 区] 在双字 (32 位) 数据写操作或批量写入多个数据时，使用 40 功能码；

[M 区] 在双字 (32 位) 数据写操作或批量写入多个数据时，使用 10 功能码。

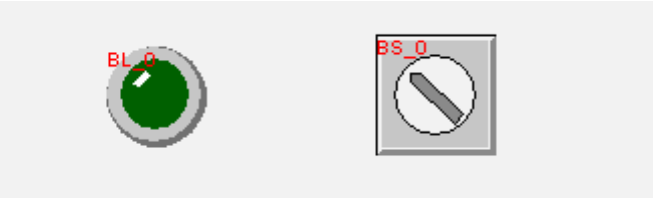
NOTE

在内部属性中添加通道时，起始地址均为 0，这是遵从汇川 AM600 内置 ModbusTCP 协议的，即所谓的“协议地址”。

3.11.2 通信示例

1 读写位变量

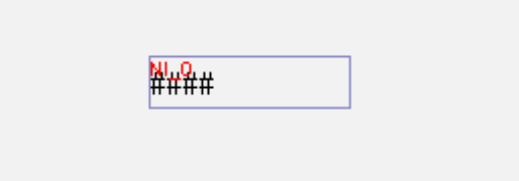
分别位状态指示灯和位状态切换开关各一个控件，如下所示：



修改位状态指示灯控件一般属性，设置地址为 Q_bit(0:0)。

2 读写字变量

选择数值输入控件放入页面，如下图所示：



修改数值输入控件一般属性，设置地址为 MW0。

3.11.3 常见故障分析

故障现象	分析	处理建议
通讯超时	采集初始化错误 或采集无数据返回 （通讯硬件连接、参数设置问题）	1、检查网络设备参数设置是否正确
		2、检查串口是否被其他程序占用
		3、检查通讯电缆是否正确连接
		4、读取数据地址超范围

故障现象	分析	处理建议
命令失败	读写操作失败	1，读取数据地址超范围
		2、通讯电缆太长，做短距离测试
		3、现场干扰太大，避免周围环境干扰

1 驱动构件支持的寄存器及功能码

寄存器	读取功能码	写入功能码	功能码说明
[SM 区] 输入线圈	31	35 3F	31：读取输入线圈状态 35：强制单个输入线圈 3F：强制多个输入线圈
[Q 区] 输出线圈	01	05 0F	01：读取输出线圈状态 05：强制单个输出线圈 0F：强制多个输出线圈
[SD 区] 输入寄存器	33	36 40	33：读输入寄存器 36：预置单个寄存器 40：预置多个寄存器
[M 区] 输出寄存器	03	06 10	03：读保持寄存器 06：预置单个寄存器 10：预置多个寄存器

说明：

1） 本驱动构件支持 01、31、03、33、05、35、06、36、0F、3F、10、40 等常用功能码，对于其它非数据通讯用功能码暂不支持。

2） 以上功能码均以 16 进制标注。功能码 0x0F 和 0x10 分别对应 10 进制的 15 和 16。

3） “[SM 区] 输入线圈”在批量写入多个继电器时，使用 3F 功能码。

4） “[Q 区] 输出线圈”在批量写入多个继电器时，使用 0F 功能码。

5） “[SD 区] 输出寄存器”在双字 (32 位) 数据写操作或批量写入多个数据时，使用 40 功能码。

6） “[M 区] 输出寄存器”在双字 (32 位) 数据写操作或批量写入多个数据时，使用 10 功能码。

NOTE

添加寄存器通道时，起始地址均为 0，这是遵从汇川 AM600 内置 Modbus 协议的，即所谓的“协议地址”。

2 数据类型表

BTdd	位（dd 范围：00 — 15）
16-bit-BCD	16 位 BCD
32-bit-BCD	32 位 BCD
16-bit-unsigned	16 位 无符号
16-bit-signed	16 位 有符号
32-bit-unsigned	32 位 无符号
32-bit-unsigned	32 位 有符号
32-bit-float	32 位 浮点数

3 32 位数在寄存器中的存储

7） PLC 工程中变量名称与寄存器地址对应关系（寄存器 16 位）

- 变量 %MW5--- 对应寄存器地址 5。
- 变量 %MD5--- 对应寄存器地址是 10。

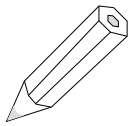
变量名称前缀：W --- 字 (16 Bits)； D--- 双字 (32 Bits）。

8） 驱动设备命令以及通道采集寄存器地址都是从 0 开始编址，0 就代表寄存器地址为 0。

9） 当读取或者写入 32 位数据时，都是从对应寄存器地址开始占两个寄存器（即 32 位）。

Memo NO. _____

Date / /



第 4 章 编程基础

4 编程基础

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

与其他高级语言类似，CoDeSys 也有常量和变量的概念。所谓常量就是数值不变的数；变量是由用户定义的标识符。变量的存储位置可由用户指定为 %I 区、%Q 区、%M 区的特定地址，亦可不指定地址，由系统自行分配，用户不需要关注这些变量的存储位置。

4.1 直接地址

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

4.1.1 定义语法

语法：%< 存储器区前缀 >< 大小前缀 >< 数字 >|.< 数字 >

■ < 存储器区前缀 >：编程系统支持以下存储区前缀

1） I：输入，物理输入，“传感器”

2） Q：输出，物理输出，“执行器”

3） M：存储位置

■ < 大小前缀 >：编程系统支持以下大小前缀

1） X：Bit，一位

2） B：Byte，一个字节

3） W：Word，一个字

4） D：Double Word，四个字节（双字）

■ < 数字 >|.< 数字 >

第一个数字是存储区前缀的偏移地址，如果定义 BOOL 类型变量，需使用 < 数字 >.< 数字 > 格式，“.”后的数字是偏移地址的偏移位数。

示例：

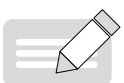
%QX7.5 输出区域偏移 7 个字节的第六位（bit5）

%QB17 输出区域偏移 17 个字节

%IW215 输入区域偏移 215 个字

%MD48 内存区域偏移 48 个双字

iVar AT %IW10: WORD;//iVar 变量是字类型，映射到输入区域偏移 10 字的位置



NOTE

◆ 大小前缀为 X 类型变量代表的数据类型为 BOOL 型，偏移地址应具体到位；

◆ 大小前缀和数据类型是匹配的，大小前缀为 B 类型的变量应声明为一个字节的数据类型，如 BYTE，SINT，USINT；大小前缀为 W 类型的变量应声明为一个字的数据类型，如 WORD，INT，UINT；大小前缀为 D 类型的变量应声明为一个双字的数据类型，如 DWORD，DINT，UDINT。

4.1.2 PLC 直接地址存储区域

不同 PLC 提供的直接存储区域不同。对于 PLC 数据，%I、%Q 区地址不能掉电保存，对于 %M 区可以掉电保存。

AM600、AM610、AM401、AM402 编程系统提供 128KB（Byte）的输入区域（I 区），128KB（Byte）输出区域（Q 区）和 512KB 存储区域（M 区），其中存储区域中的前 480KB 用户可以直接使用，后 32K 为系统使用区域（主要用作软元件），用户不要直接使用。编程时，用户可以直接访问地址，也可以定义变量后把变量映射到地址间接访问。存储区域定义及使用的地址范围如下表。

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）512KB	用户使用区域	240KWords	%MW0-%MW245759
	SD 元件	10000Words	%MW245760-%MW255759
	SM 元件	10000Bytes	%MB511520-%MB521519
	保留	2768Bytes	%MB521520-%MB524287

AC800 系列编程系统提供 128KB（Byte）的输入区域（I 区），128KB（Byte）输出区域（Q 区）和 5MB 存储区域（M 区）。AC800 系列不支持 SD 和 SM 软元件，%M 区地址可以随意使用。存储区域定义及使用的地址范围如下表。

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）5MB	用户使用区域	2.5MWords	%MW0-%MW2321439

4.2 变量

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

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

4.2.1 变量定义

变量可以在声明编辑器中定义。声明编辑器有两种显示形式：文本视图和表格视图。POU 的文本视图声明编辑器如下图：

```
PROGRAM PLC_PRG
VAR
    a AT %MB1:INT;
    a_b_C:INT;
    fb1: FB1;
    // test for comment
    c: INT;
    e: INT;
    f: INT;
    h AT %M*:INT;
    arr :ARRAY[0..2] OF INT;
    b1 AT %MX0.1:BOOL;
    b2 AT %MX0.2:BOOL;
    b AT %MB0:BYTE:=2#1; (* 测试注释 variable with initial value 1*)
    g AT %MX2.12:BOOL;//测试注册
    t1:TIME;
    t2:TOD;
    t3:TIME_OF_DAY;
    t4:DATE;
    t5:DATE_AND_TIME;
    t6:DT;
    f1:REAL:=1.64E+7;
    str1:STRING:='$30b';
    ref_int: REFERENCE TO INT;
```


图 4-1 文本视图声明编辑器

表格视图声明编辑器如下图：

图 4-2 表格视图声明编辑器

定义语法：< 标识 > {AT < 地址 >}:< 数据类型 > {:=< 初始值 >};

位于大括号 {} 中的是可选部分。

1 标识

标识即变量的名称。变量命名应注意以下事项：

1） 不能包含空格或者特殊字符

2） 不能是预定义的关键字

3） 名称不区分大小写

4） 名称长度没有限制

5） 名称不能重复定义

定义的本地变量名称可以和全局变量重名，默认使用本地变量，该变量可以用来表示全局变量，也可以通过全路径变量名来指定具体的变量。例如 : 本地变量 iVar:=1; 全局变量 .iVar:=2; 全路径变量 globlist1.iVar：=3;

命名时需考虑部分命名建议：如变量名应准备表达其意义及数据类型，变量最好采用匈牙利命名法 (变量名 = 属性 + 类型 + 对象描述)。

2 AT 地址

AT 地址为直接地址，具体见请参见章节 4.1。

3 数据类型

数据类型分为标准数据类型和用户自定义数据类型。

1） 标准数据类型

标准数据类型分为布尔类型，整形，浮点型，字符串，时间类型。

类型	关键字	范围	占用内存
布尔类型	BOOL	TRUE，FALSE，0，1	8Bit
BIT 类型	BIT	TRUE，FALSE，0，1，只能在结构体或者功能块中使用	1Bit

类型	关键字	范围	占用内存
整数	BYTE	0-255	8Bit
	WORD	0-65535	16Bit
	DWORD	0-4294967295	32Bit
	LWORD	0 - 2^{64}-1	64Bit
	SINT	-128-127	8Bit
	USINT	0-255	8Bit
	INT	-32768-32767	16Bit
	UINT	0-65535	16Bit
	DINT	-2147483648-2147483647	32Bit
	UDINT	0-4294967295	32Bit
	LINT	-2^{63} - 2^{63}-1	64Bit
	ULINT	0 - 2^{64}-1	64Bit
浮点类型	REAL	1.401e-45 - 3.403e+38	32Bit
	LREAL	2.2250738585072014e-308 - 1.7976931348623158e+308	64Bit
字符串	STRING	只支持 ASCII 码字符（不支持中文字符）。默认最大长度 80 字符，超过最大长度会被去掉。可以声明字符最大长度，如 str:STRING(35):=’This is a String’;字符串函数最大支持 255 字符	ASCII 形式存储字符串，用一个字节存储结束符
	WSTRING	只支持 UNICODE 字符（支持中文字符）。默认最大长度 80 字符，超过最大长度会被去掉。可以声明字符最大长度，如 wstr:WSTRING(35):=”This is a WString”;	UNICODE 形式存储字符串，用两个字节存储结束符
时间	TIME	时间常量，日，时，分，秒，毫秒	32Bit，内部按 Dword 处理
	TIME_OF_DAY(TOD)	一天的时间常量	32Bit，内部按 Dword 处理
	DATE	日期常量，从 1970 年 1 月 1 日开始	32Bit，内部按 Dword 处理
	DATE_ADN_TIME(DT)	日期和时间常量，从 1970 年 1 月 1 日开始	32Bit，内部按 Dword 处理

2） 用户自定义数据类型

用户自定义数据类型包括数组 / 结构体 / 枚举 / 联合 / 别名 / 子集 / 引用 / 指针。在中型 PLC 编程软件 InoProShop 中，可以通过右键应用 - 添加对象 -DUT 来添加结构体、枚举、联合、别名 4 种自定义数据类型。

■ 数组

语法：<Array_Name>:ARRAY [<ll1>..<ul1>,<ll2>..<ul2>,<ll3>..<ul3>] OF <elem. Type>

ll1, ll2, ll3 定义区域的下限，ul1, ul2 ul3 定义上限，数值必须为整数，elem. Type 为每个数组元素数据类型

初始化及示例

```
Card_game: ARRAY [1..13, 1..4] OF INT;

arr1 : ARRAY [1..5] OF INT := [1,2,3,4,5];

arr2 : ARRAY [1..2,3..4] OF INT := [1,3(7)]; (* 数组值 1,7,7,7*)

arr3 : ARRAY [1..2,2..3,3..4] OF INT := [2(0),4(4),2,3];    (* 数组值
0,0,4,4,4,4,2,3*)

arr1 : ARRAY [1..10] OF INT := [1,2]; (* 数组部分初始化，没有初始化元素为默认值 0*)
```

数组结构初始化示例

结构定义：

```
TYPE STRUCT1
STRUCT
 p1:int;
 p2:int;
 p3:dword;
END_STRUCT
END_TYPE
```

数组结构初始化：

```
arr1:ARRAY[1..3] OF STRUCT1:= [(p1:=1,p2:=10,p3:=4723),(p1:=2,p2:=0,p3:=299),(p1
:=14,p2:=5,p3:=112)];
```

访问联合元素语法：

```
<Array-Name>[Index1,Index2].
```

例子：

Card_game [9,2]

■ 结构

语法：

```
TYPE <structurename> | EXTENDS DUTTYPE:
STRUCT
    <declaration of variables 1>
    ...
    <declaration of variables n>
END_STRUCT
END_TYPE
```

<structurename> 是一个类型，可作为数据类型使用的。EXTENDS DUTTYPE 是可选项，表示继承了 DUTTYPE 中的成员，可以通过 structurename 类型变量访问 DUTTYPE 的成员。这里的 DUTTYPE 可以为结构类型，联合类型或者别名。

初始化及示例

Polygonline 类型结构定义：

```
TYPE Polygonline:
STRUCT
    Start:ARRAY [1..2] OF INT;
    Point1:ARRAY [1..2] OF INT;
    Point2:ARRAY [1..2] OF INT;
    Point3:ARRAY [1..2] OF INT;
    Point4:ARRAY [1..2] OF INT;
    End:ARRAY [1..2] OF INT;
```

```
END_STRUCT
END_TYPE
```

初始化：

```
Poly_1:polygonline := ( Start:=[3,3], Point1:=[5,2], Point2:=[7,3],
Point3:=[8,5], Point4:=[5,7], End:= [3,5]);
```

访问结构元素语法：

```
<structurename>.<variable>
```

例子：

Poly_1.Start

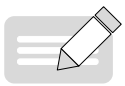
■ 枚举

枚举类型是由若干串常量组成的，这些常量被称为枚举类型值。

语法：

```
TYPE <identifier>:(<enum_0> ,<enum_1>, ...,<enum_n>) |<base data type>;
END_TYPE
```

identifier: 自定义的枚举类型；enum_n：枚举类型对应的常量值，每个常量可以声明其对应值，如果不声明使用默认值；base data type 枚举常量对应数据类型，可以不用声明，默认为整数。



NOTE

◆ 同枚举变量一在多个库中同时存在时，需添加库名前缀，否则编译报错。

■ 联合

语法：

```
TYPE <unionname>:UNION
    <declaration of variables 1>
    ...
    <declaration of variables n>
END_UNION
END_TYPE
```

< unionname > 是一个类型，并且可作为数据类型使用的。联合中的所有变量具有相同的存储位置，联合类型变量分配的空间大小为其中占用最大空间的变量分配的大小

示例

```
TYPE union1: UNION
a : LREAL;
b : LINT;
END_UNION
END_TYPE
```

访问数组元素语法：

```
< unionname >.<variable>
```

示例

```
union1.a
```

■ 别名

把一种数据类型用一种别名来表示。

语法：

```
TYPE <aliasname>:basetype END_TYPE
```

aliasname 为别名类型名，用作数据类型。basetype 可以是标准数据类型，也可以是用户自定义数据类型。

示例

```
TYPE alias1 : ARRAY[0..200] of byte; END_TYPE
```

初始化及访问方式和其对应的基本类型一致。

■ 子集

子集数据类型是其定义的基本数据类型的子集，可以通过增加 DUT 来增加一个子集类型，也可以直接声明一个变量为子集类型。

DUT 对象语法：

```
TYPE <name> : <Inttype> (<ug>..<og>) END_TYPE;
```

name：有效的 IEC 标示符

Inttype：是数据类型 SINT, USINT, INT, UINT, DINT, UDINT, BYTE, WORD, DWORD (LINT, ULINT, LWORD) 中的一种。

ug：是一个常数，必须符合基本类型对应的下边界范围。下边界本身包含在此范围内。

og：是一个常数，必须符合基本类型对应的上边界范围。上边界本身包含在此范围内。

DUT 对象声明示例

```
TYPE
SubInt : INT (-4095..4095);
END_TYPE
```

变量直接声明示例

```
VAR
    i : INT (-4095..4095);
    ui : UINT (0..10000);
END_VAR
```

■ 引用

引用是一个对象的别名，操作引用就如同操作对象。

语法：

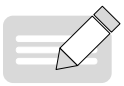
```
<identifier> : REFERENCE TO <data type>
```

identifier：引用标示符。data type：引用对象的数据类型。

示例及初始化

```
ref_int : REFERENCE TO INT;
```

```
a : INT;
b : INT;
ref_int REF= a;    (* ref_int 引用 a *)
ref_int := 12;     (* a 值为 12 *)
b := ref_int * 2; (* b 值为 24 *)
ref_int REF= b;    (* ref_int 引用 b *)
ref_int := a / 2; (* b 值为 6 *)
```



NOTE

不能引用 BIT 类型，即不允许定义 ref1:REFERENCE TO BIT。

■ 指针

指针保存的是一个对象的地址，指针可以指向任何数据类型（BIT 类型除外）

语法：

```
<identifier>: POINTER TO <data type>;
```

identifier：指针标示符。data type：指针指向的数据类型。

通过地址操作符对指针进行操作。地址操作符包括 ADR（获取变量地址）和 ^（变量地址对应的值）

示例及初始化

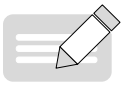
```
VAR
    pt:POINTER TO INT;  (* 声明指向 INT 类型的指针 pt*)
    var_int1:INT := 5;
    var_int2:INT;
END_VAR

pt := ADR(var_int1); (* 变量 varint1 的地址分配给指针 pt *)
var_int2:= pt^;      (* 通过 ^ 地址操作符获取指针对应的值 *)
pt^:=33;   (* 给指针对应的变量 var_int1 赋值 *)
```

4 初始值

变量初始化默认值为 0，用户可以在变量声明时通过赋值运算符“:=”来增加自定义初始化值。初始化值为有效的 ST 表达式。ST 表达式由操作符，操作数，赋值表达式组成，操作符主要是加（+），减（-），乘（*），除（/）等组成；操作数主要是指常量，变量和函数；赋值表达式指 ST 表达式中的赋值运算符“:=”。因此，初始化可以为常量，变量或者函数，只是要确保使用的变量已经被初始化。

示例：

```
VAR
var1:INT := 12;  (* 整形变量初始值 12*)
x : INT := 13 + 8;  (* 常量表达式定义初始值 *)
y : INT := x + fun(4);  (* 初始值包含函数调用 *)
z : POINTER TO INT := ADR(y);  (* 指针通过地址函数 ADR 来初始化 *)
END_VAR
```


NOTE

◆ 全局变量表（GVL）一般在 POU 本地变量定义之前已经初始化；
◆ 定义时初始化指针时，如果是在线修改默认，指针将不会被初始化（指针还是指向在线修改之前的变量）。

4.2.2 变量类型

主要的变量类型包括：本地变量（VAR）、输入变量（VAR_INPUT）、输出变量（VAR_OUTPUT）、

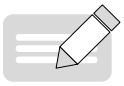
输入输出变量（VAR_IN_OUT）、全局变量（VAR_GLOBAL）、临时变量（VAR_TEMP）、静态变量（VAR_STAT）以及配置变量（VAR_CONFIG）。

变量类型声明语法：`<type_key> |atribute_key`

```
variable1;
variable2;
...
END_VAR
```

type_key：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）。

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


NOTE

1) 有关 RETAIN，、PERSISTEN 变量的详细信息，请参见下文章节见 4.4.2 章节；
2) 有关 CONSTANT 的详细信息，请参见下文章节见 4.3。

1 本地变量（VAR）

在 POU 内部 VAR 和 END_VAR 之间的变量都为本地变量，不能被外部访问。

赋值格式：

本地变量 := 值

示例

```
VAR
    iLoc1:INT; (* 本地变量 *)
END_VAR
```

2 输入变量（VAR_INPUT）

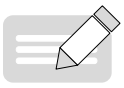
在 POU 内部 VAR_INPUT 和 END_VAR 之间的变量都为输入变量，可以在调用位置给输入变量赋值。

POU 调用格式：

本地变量 := 调用者输入值

示例

```
VAR_INPUT
    iIn1:INT; (* 输入变量 *)
END_VAR
```



NOTE

输入变量在 POU 内部也是可以修改的，即使增加了 CONSTANT 属性。

3 输出变量（VAR_OUTPUT）

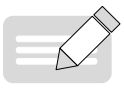
在 POU 内部 VAR_OUTPUT 和 END_VAR 之间的变量都为输出变量。输出变量可以在调用时返回给调用者，调用者可以做进一步处理。

POU 调用格式：

输出变量 => 调用者匹配类型变量

示例

```
VAR_OUTPUT
    iOut1:INT; (* 输出变量 *)
END_VAR
```



NOTE

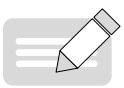
1) 对于功能（FUNCTION）和方法（METHOD）除了返回值外，可以有另外的输出变量，但是必须在调用时分配调用者接收变量。例如 fun(iIn1 := 1, iIn2 := 2, iOut1 => iLoc1, iOut2 => iLoc2);
2) 对于功能块，可以在调用后把功能块输出变量赋值给调用者。

4 输入输出变量（VAR_IN_OUT）

在 POU 内部 VAR_IN_OUT 和 END_VAR 之间的变量都为输入输出变量。输入输出变量不仅可以传入被调用的 POU 内，并且可以在被调用的 POU 内部修改。实际上传递给被调用 POU 内的变量是调用者变量的引用。

示例

```
VAR_IN_OUT
    iInOut1:INT; (* 输入输出变量 *)
END_VAR
```



NOTE

1) 因传递给被调用 POU 内的变量是调用者变量的引用，所以不能直接访问功能块实例中的输入输出变量，也就是不能直接使用 <FBinstance>.<InOutVariable>，因输入变量已经是调用者变量的引用，已经发生了变化；
2) 输入输出变量不能是常量和 Bit 类型直接变量（如 xBit0 AT %I2.0:BOOL）。如果需要声明输入输出常量，可以加 CONSTANT 属性（VAR_IN_OUT CONSTANT）。如果需要 Bit 类型直接变量，需要增加一个中间变量作为输入输出变量，然后把中间变量值给 Bit 类型直接变量。

Bit 类型直接变量示例：

```
VAR_GLOBAL
    xBit0 AT %MX0.1 : BOOL;(* 声明 Bit 类型直接变量 *)
    xTemp : BOOL; (* 中间变量 *)
```

```
END_VAR
// 带有输入输出变量 (xInOut) 的功能块
FUNCTION_BLOCK FB_Test
VAR_INPUT
    xIn    : BOOL;
END_VAR
VAR_IN_OUT
    xInOut : BOOL;
END_VAR
IF xIn THEN
    xInOut := TRUE;
END_IF
 // 程序中调用功能块
PROGRAM Main
VAR
    xIn : BOOL;
    I1  : FB_Test;
    I2  : FB_Test;
END_VAR
 // 使用 Bit 类型的直接地址变量，编译报错
//I1(xIn:=xIn, xInOut:=xBit0);
// 通过中间变量 xTemp 把 xBit0 的值传递给功能块，然后把中间变量赋值给 xBit0
xTemp := xBit0;
I2(xIn:=xIn, xInOut:=xTemp);
xBit0 := xTemp;
```

输入输出常量 (VAR_IN_OUT CONSTANT) 只能读不能写，而输入变量在当前版本是能够修改的，即使增加了常量属性，所以可以用输入输出常量来把变量属性变为不可修改。

输入输出常量示例：

```
PROGRAM PLC_PRG
VAR
    sVarFits : STRING(16);
    sValFits : STRING(16) := '1234567890123456';
    iVar: DWORD;
END_VAR
POU(sReadWrite:='1234567890123456', scReadOnly:='1234567890123456',
iVarReadWrite:=iVar);
//POU(sReadWrite:=sVarFits, scReadOnly:=sVarFits, iVarReadWrite:=iVar);
//POU(sReadWrite:=sValFits, scReadOnly:=sValFits, iVarReadWrite:=iVar);
```

```
//POU(sReadWrite:=sVarFits, scReadOnly:='23', iVarReadWrite:=iVar);


FUNCTION POU : BOOL
VAR_IN_OUT
    sReadWrite  : STRING(16);     (* 在此 POU 内此字符串可读可写 *)
    iVarReadWrite : DWORD;  (* 在此 POU 内此字此变量可读可写 *)
END_VAR
VAR_IN_OUT CONSTANT
    scReadOnly : STRING(16);  (* 在此 POU 内此字符串只读的 *)
END_VAR
sReadWrite := 'string_from_POU';
iVarInPOU := STRING_TO_DWORD(scReadOnly);
```

5 全局变量（VAR_GLOBAL）

定义在 VAR_GLOBAL 和 END_VAR 之间的变量为全局变量。一般变量，常量，保留变量都可以声明为全局变量。在 AM600 编程软件 InoPro 中可以通过右键应用 - 添加对象 - 添加全局变量表来添加全局变量表，然后在全局变量表中添加全局变量。

示例

```
VAR_GLOBAL
    iGlobVar1:INT; (* 全局变量 *)
END_VAR
```

NOTE

1) 本地变量如果和全局变量具有相同的名称，直接对此变量名进行操作时表示操作的为本地变量，可以在变量名前加全局范围操作符（.）来操作全局变量，如 .iGlobVar1；
2) 全局变量总是在局部变量之前初始化。

6 临时变量（VAR_TEMP）

定义在 VAR_TEMP 和 END_VAR 之间的变量为临时变量，临时变量在每次调用时会进行初始化。

示例

```
VAR_TEMP
    iTemp1:INT; (* 临时变量 *)
END_VAR
```

NOTE

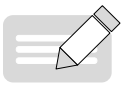
1) 临时变量只能在程序和功能块中声明；
2) 临时变量只能声明的程序或者功能块中使用。

7 静态变量（VAR_STAT）

定义在 VAR_STAT 和 END_VAR 之间的变量都为静态变量。静态变量在第一次调用时被初始化，在每次调用完此 POU 后，变量值依然保持下来。

示例

```
VAR_STAT
    iStat1:INT; (* 静态变量 *)
END_VAR
```


NOTE

1) 静态变量只能在功能块，功能和方法中声明，不能在程序中声明；
2) 静态变量只能声明的 POU 中使用。

8 配置变量（VAR_CONFIG）

定义在 VAR_CONFIG 和 END_VAR 之间的变量都为配置变量。配置变量是直接变量，一般是映射到功能块定义的不确定地址直接变量。在功能块内可以定义一个不确定地址的变量，此变量的地址通过“*”来表示不确定的地址（任意的地址），然后添加一个配置变量表（通过添加全局变量表方式），把所有功能块实例中不确定地址变量添加到配置变量表中，在此变量表中把所有的不确定地址给明确下来，这样就可以集中管理所有功能块中不确定地址变量。

功能块不确定地址变量定义语法：

< 标识符 > AT %<I|Q|M>* : < 数据类型 >

地址的最终确定是在全局变量列表的“变量配置”中完成：

■ 示例

```
FUNCTION_BLOCK locio

VAR

    xLocIn AT %I*: BOOL := TRUE;

    xLocOut AT %Q*: BOOL;

END_VAR
```

这里定义了两个 I/O- 变量，一个本地输入变量 (%I*) 和一个本地输出变量 (%Q*)。

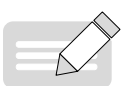
然后添加“全局变量列表”对象（GVL）。在关键词 VAR_CONFIG 和 END_VAR 之间输入实例变量声明的具体地址，这里的实例变量指包含 POU 完整的实例路径，具体地址对应于功能块中不确定指定的地址（% I*，% Q*），另外数据类型必须与功能块的声明一致。

配置变量定义语法：

< 实例变量路径 > AT %<I|Q|M>< 位置 > : < 数据类型 >;.

■ 示例

```
PROGRAM PLC_PRG

VAR

locioVar1: locio;

locioVar2: locio;

END_VAR

VAR_CONFIG (* 正确的变量配置表 *)

PLC_PRG.locioVar1.xLocIn AT %IX1.0 : BOOL;

PLC_PRG.locioVar1.xLocOut AT %QX0.0 : BOOL;

PLC_PRG.locioVar2.xLocIn AT %IX1.0 : BOOL;

PLC_PRG.locioVar2.xLocOut AT %QX0.3 : BOOL;

END_VAR
```


NOTE

1) 一般是不需要配置变量的，因为对于 I/Q 地址输入 / 输出可以在对应模块的 I/O 映射界面通过输入助手（或者直接输入实例变量路径）把变量映射到 I/Q 地址；
2) 配置变量一般映射到功能块中不确定地址变量，它也可以映射程序中的不确定地址变量；
3) 只存在不确定地址变量或者只存在配置变量都会编译报错，两者是配合使用。

4.3 常量

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

常量声明语法：

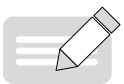
```
VAR CONSTANT
    <identifier>:<type> := <initialization>;
END_VAR
```

示例

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

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

类型	描述	示例
布尔类型	有两个值 TRUE 和 FALSE（也可以用 1 和 0），1 等于 TRUE，0 表示 FALSE	TRUE，FALSE，1
BIT 类型	和布尔类型相似，只能在结构体（占用位数）或者功能块（映射 BOOL 类型的直接地址）中使用	TRUE，FALSE，0
整数	整数常量的数值可以是二进制、十进制、八进制和十六进制。如果整数值不是十进制值，可以用“进制”加符号“#”放在整数值前面来表示。十进制的 10 至 15 在十六进制中表示为 A 至 F	十进制：66 二进制：2#101 八进制：8#72 十六进制：16#3A 类型常数： INT#22 BYTE#204
浮点类型	浮点常量用十进制小数和指数来表示，遵循标准的科学计数法格式	7.4 2.3e+9 REAL#3.12
ASCII 字符串	ASCII 字符串常量在两个单引号之间，可以包含空格和特殊字符。一个字符用一个字节表示，只支持 ASCII 码字符（不支持中文字符）。默认最大长度 80 字符，超过最大长度会被去掉。可以声明字符最大长度，如 str:STRING(35):=’ This is a String’; 字符串函数最大支持 255 字符。	$ 做转义字符例： ‘$30’：0，字符 0，16 进制 30 对应的 ASCII 字符 $$：$，美元字符 $’：’，单引号
UNICODE 字符串	UNICODE 字符串常量在两个双引号之间，一个字符站两个字节，只支持 UNICODE 字符（支持中文字符）。默认最大长度 80 字符，超过最大长度会被去掉。可以声明字符最大长度，如 wstr:WSTRING(35):=”This is a WString”；	“Unicode string”
时间	时间常量一般用来操作时间，由“T#”（或“t#”）加上“时间值”构成，时间值的单位包括天（d）、小时（h）、分（m）、秒（s）和毫秒（ms）。	t#12h34m15s;
时刻	一天的时间范围，语法：TOD# 时间值。	TOD#15:36:30.123
日期	从 1970 年 1 月 1 日开始，语法：d# 日期。	d#2015-02-12
日期时刻	日期常量和时刻常量合并起来称为日期时刻常量，由 从 1970 年 1 月 1 日开始，语法：dt# 日期。	dt#2004-03-29-11:00:00



NOTE

除 BOOL、BIT 和字符串类型类型外，其他类型都可以用关键字 # 常量值来表示某一类型常数。

4.4 掉电保持变量

4.4.1 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。

掉电保持特性主要通过属性关键字 PERSISTENT RETAIN 来声明。

■ 示例

```
VAR_GLOBAL PERSISTENT RETAIN
    iVarPers1 AT %MW100: DINT;
    bVarPers AT %MX1.1 : BOOL;
END_VAR
```

下表列出了执行复位、掉电等动作时，不同掉电保持变量的响应动作

动作	VAR	VAR PERSISTENT RETAIN 或者 VAR RETAIN PERSISTENT	VAR RETAIN
掉电	初始化	保持原值	保持原值
热复位	初始化	保持原值	保持原值
冷复位	初始化	保持原值	初始化
初始值复位	初始化	初始化	初始化
程序下载	初始化	保持原值	初始化
在线修改	保持原值	保持原值	保持原值

NOTE

1.RETAIN 变量和 PERSISTENT RETAIN 变量都属于保持变量，但保持特性不同；

2. 映射到 %M 地址的直接变量可以声明为保持变量，而映射到 %I 和 %Q 的直接变量不能声明为保持变量。

4.4.2 掉电保持变量表

如果工程中定义了掉电保持变量，则必须生成一个掉电保持变量表，否则定义的变量不具有掉电保持功能。掉电保持变量表可通过以下两种方式生成：

■ 手动添加：通过右键“应用”-“添加对象”-“掉电保持变量”来添加掉电保持变量表；

■ 自动添加：当声明了掉电保持变量，编译时会自动创建掉电保持变量表。

掉电保持变量表包含两种模式：传统模式和标准模式（推荐使用）。传统模式即旧模式，具体使用与原来保持不变。标准模式下掉电保持变量表如下图：

刷新 掉电存储区 清零存储数据

名称	地址	类型	初始值	注释	配方1	配方2
PLC_PRG.a	%MB491518	INT	INT#1		1	2
PLC_PRG.b	%MB491517	BOOL			TRUE	TRUE
PLC_PRG.c	%MB491436	STRING	'wo'		hao	dui
PLC_PRG.d	%MB491274	WSTRING	"ni"		huai	cuo
PLC_PRG.e	%MB491270	REAL	REAL#1.23		3.21	4.56
PLC_PRG.f	%MB491250	ARRAY [1..10] OF INT				
PLC_PRG.g	%MB491248	ENUM1			enum_1	enum_2
PLC_PRG.h	%MB491244	TIME	TIME#120ms		1	2

其中，工具栏选项的功能如下：

选项	功能	说明
刷新	将外部工程的持久性变量添加到持久性变量表中，并对未进行地址分配的持久性变量进行地址分配； 刷新变量配方数据结构； 检测所有掉电保持变量地址合法性。	-
掉电存储区	实现模式切换和标准模式存储区域地址范围设置	区域地址范围设置请参见本文章节 4.4.4.2
清零存储数据	在线状态下，用于清零设备内存中掉电保持变量	-

列表条目说明如下：

条目	功能	属性
名称	显示掉电保持变量的来源和变量名	不可编辑
地址	显示掉电保持变量的地址	可编辑
类型	显示掉电保持变量的类型	不可编辑
初始值	显示掉电保持变量声明时的初始值 选择初始值列，执行菜单命令“当前值 -> 初始值”命令时，掉电保持变量在线值会被写入初始值列，并同步到工程中的掉电保持变量声明中。	不可编辑
注释	显示掉电保持变量的注释信息	不可编辑
配方	显示并保存掉电保持变量配方值 用户可按需添加	可编辑

4.4.3 掉电保持模式

1 模式对比

为了保持工程与软件版本的兼容性，工程中保留了掉电保持变量旧编辑模式，即传统模式。

当打开的旧工程含有“持久变量”时，系统仍可按传统模式来显示和编辑工程；如果原工程不存在“掉电保持变量”，则新建的掉电保持变量将遵循“标准模式”。

目前掉电保持变量包含标准模式和传统模式（旧模式），二者的差异对比如下表所示：

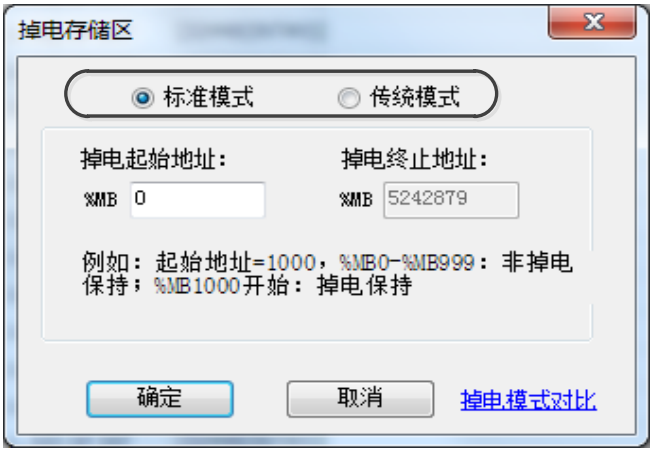
选项	标准模式	传统模式（旧模式）
数据来源	全部来源于掉电保持变量表外部带有 Persistent Retain 特性或分配到 M 区掉电保存区范围的变量	掉电保持变量表内部定义变量和外部带 Persistent Retain 特性变量
地址映射	所有的变量均需强关联到 M 区掉电保存区	不需强关联
中间插入或删除变量	由于变量是通过地址映射，因此插入或删除变量对其它变量保存值无影响	意外操作可能引起数据丢失（例: 清除全部、PLC 设备更新，编译选项修改等）
配方功能	内部融合配方功能	只能使用外部独立配方功能

NOTE

仿真模式下，如果掉电保持变量为标准模式，执行“冷复位”命令，变量值将会被初始化。

2 传统模式切换标准模式

掉电保持变量表可以通过工具栏“掉电存储区”选项进行模式切换。界面如下图所示：



“传统模式”切换为“标准模式”时，传统模式中自定义的掉电保持变量会保存到新生成的 GVL 程序中。重新编译或点击刷新时，系统会将 GVL 中的掉电保持变量添加到标准模式数据表中，并为其分配地址。

NOTE

如果用户重新切换回“传统模式”，原数据保持不变。但新生成的 GVL 程序不会被删除。因此，在切回“传统模式”时，需手动删除新生成的 GVL 程序。

3 模式兼容性处理

对于掉电保持模式，PLC 固件和工程版本的兼容性存在以下两种情况：

1） 如果 PLC 支持掉电保持模式切换，下载工程后，PLC 自动切换为与工程设置相同的保持模式。

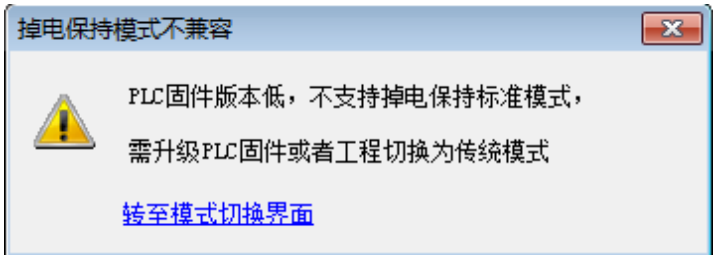
NOTE

若工程版本和 PLC 模式不一致，第一次下载工程时 PLC 会自动切换掉电保持模式，可能导致数据不符合预期。

支持掉电保持模式的 PLC 固件版本列表如下：

PLC 型号	版本限制
AM600	1.24.20.0（含）以上版本
AM401	21.24.20.0（含）以上版本
AM402	41.24.20.0（含）以上版本
AM403	81.24.20.0（含）以上版本
AP700	1.13.30.0（含）以上版本
AC810/AC801/AC802	1.13.30.0（含）以上版本

2） 如果 PLC 不支持掉电保持模式切换，则需要升级 PLC 固件或者将工程掉电保持模式切换为传统模式。



转至模式切换界面：进入掉电保持变量表，并打开模式切换界面。

4.4.4 地址分配

若用户程序中定义了掉电保持变量，但并未给变量分配地址，在标准模式下，只需点击工具栏中的“编译”或掉电保持界面的“刷新”，系统会自动为其分配地址。

1 地址分配的基本流程

系统为掉电保持变量分配地址的基本流程：

1） 系统获取当前掉电保存区的范围。例如：Am600 型号 PLC 的默认掉电保存区域为 %MB0~%MB491519。

2） 将已被占用的地址从可用地址中去除。

3） 从可用地址中部开始，根据变量所需占用的空间大小，对变量进行地址分配。

【注意】考虑用户 Modbus 地址使用范围，在初次分配地址时会避开其常规使用区域（%MB0-%MB131071），以 %MB131072 开始，向后分配；只有当尾部地址使用完或者无法再为变量分配地址，再从 %MB0-%MB131071 范围从头向后分配。

例如，当前可用最大地址为 %MB50000，要为 Real 型变量 Var 进行地址分配：

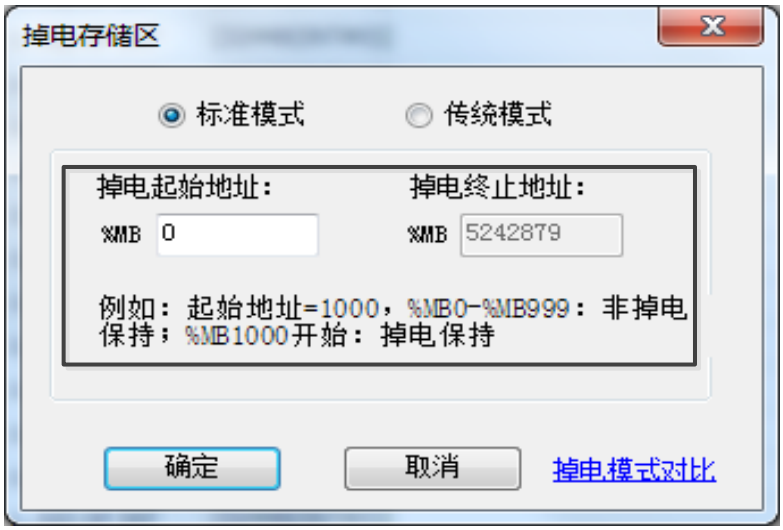
由于 Real 类型变量空间大小为 4Byte，起始地址计算方法为 : 首先在“%MB131072-%MB50000”范围开始选择地址，首先选取“%MB131072”为起始地址，再与现有分配的地址进行冲突检测，如果发生地址冲突，则抛弃该地址，从下一个可用地址再次重新计算地址，再次检测，直至地址不发生冲突为止；如果在“%MB131072-%MB50000”范围内找不到一块完整的区域保存该变量，则开始在 “%MB0-%MB1310721”范围内开始查找，选取“%MB0”为起始地址，同样按照前边方式进行冲突检测，直至找到合法地址为止。

NOTE

- 针对 FB、Struct 类型的变量或是基类型为 FB/Struct 的数组变量，分配地址时，会根据 4 字节对齐原则为其分配能被 4 整除的起始地址。
- 例如：%MB1000 分配给 FB 类型变量，但 %MB1001 就无法分配给该变量。

2 地址区域设置

地址分配区域为 PLC 的 M 区，具体的地址范围可以通过工具栏中“掉电存储区”选项查看和设置。如下图所示：



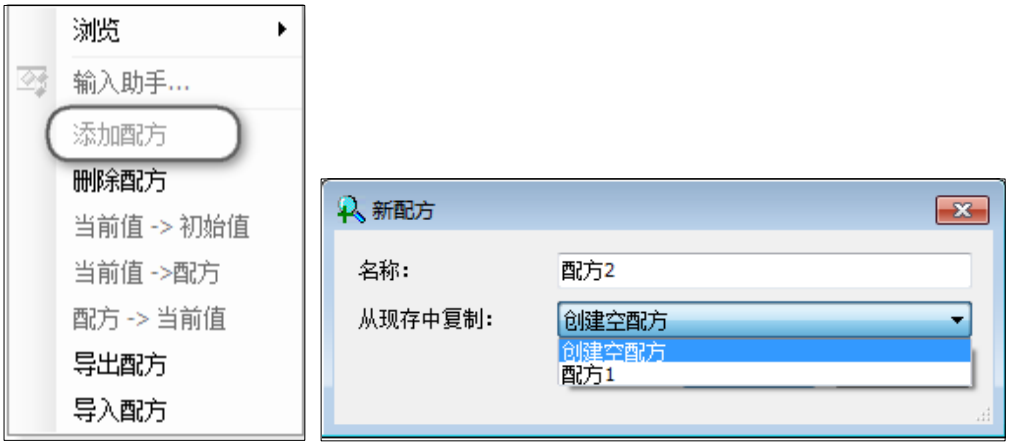
4.4.5 配方操作

配方可以保存一组变量的数据，作为掉电保持变量写入值。

在掉电保持标准模式下，用户可以在掉电保持变量表中右键单击“配方”条目来执行“添加配方”、“删除配方”、“当前值 -> 配方”、“配方 -> 当前值”、“导出配方”、“导入配方”操作。

1 添加配方

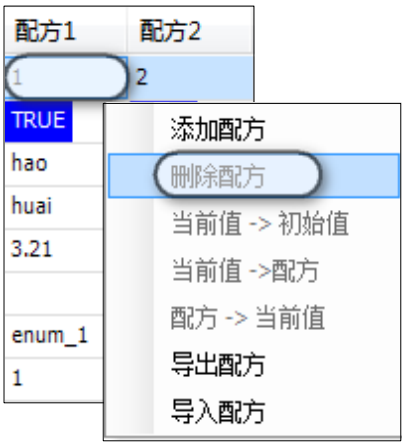
在掉电保持变量表中右键选择 “添加配方”：



- 功能：此操作在掉电保持变量表最后一列后添加新的配方列。在弹出的“新配方”设置对话框中设置新配方列的名称，也可选择从已有的配方列中复制。
- 使能条件：存在标准模式的掉电保持变量表。
- 注意事项：首次添加配方时，需要先执行“刷新”，否则会弹框提示进行刷新。

2 删除配方

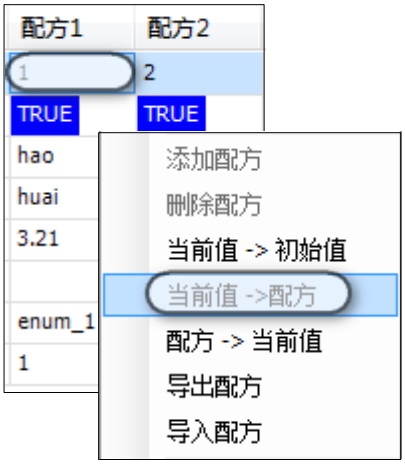
在掉电保持变量表中选中配方列，右键单击并选择 “删除配方”：



- 功能：此操作将删除鼠标所选中的配方列。
- 使能条件：存在标准模式的掉电保持变量表，且鼠标选中要删除的配方列。

3 当前值 -> 配方

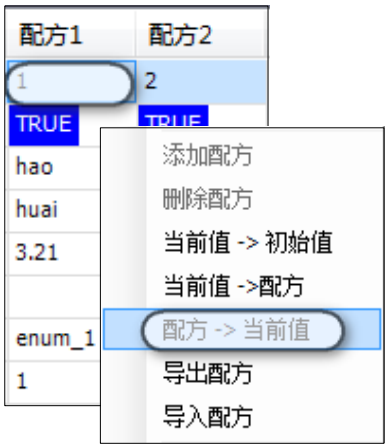
在掉电保持变量表中选中配方列，右键单击并选择 “当前值 -> 配方”：



- 功能：此操作将掉电保持变量的在线值保存到鼠标所选的配方列。
- 使能条件：登录工程，存在标准模式的掉电保持变量表，且鼠标选中对应的配方列。
- 注意事项：初始值列和配方列的数据结构要一致，才可以正常执行该命令，否则会弹出对话框提示刷新。

4 配方 -> 当前值

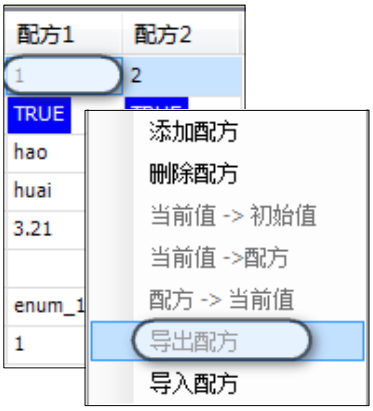
在掉电保持变量表中选中配方列，右键单击并选择 “配方 -> 当前值”：



- 功能：此操作将鼠标所选配方列中的掉电保持变量配方值写入在线值列。
- 使能条件：登录工程，存在标准模式的掉电保持变量表，且鼠标选中对应的配方列。
- 注意事项：初始值列和配方列的数据结构要一致，才可以正常执行该命令，否则会弹出对话框提示刷新。

5 导出配方

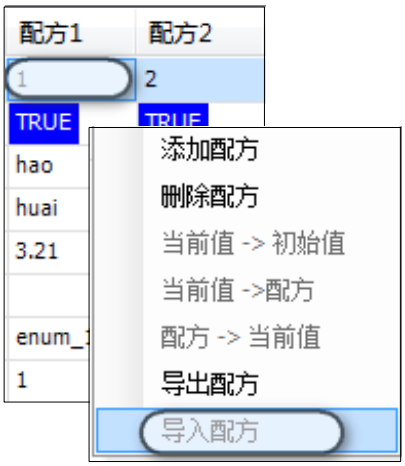
在掉电保持变量表中选中配方列，右键单击并选择 “导出配方”：



- 功能：此操作将鼠标所选配方列的值导出到后缀为“.txtrecipe”的新文件中，并将该文件保存到指定位置。
- 使能条件：存在标准模式的掉电保持变量表，且鼠标选中对应的配方列。

6 导入配方

在掉电保持变量表中选中配方列，右键单击并选择 “导入配方”：



- 功能：此操作将指定位置中所选文件（后缀为“.txtrecipe”）的配方值导入到鼠标选中的配方列。
- 使能条件：存在标准模式的掉电保持变量表，且鼠标选中对应的配方列。
- 注意事项：导入配方中的变量需包含在掉电保持变量表中，否则系统会提示不匹配。

第 5 章 编程语言

5 编程语言

5.1 InoProShop 支持的编程语言简介

编程软件支持下列 PLC 编程语言：

·Ladder diagram(LD) 梯形图

·Function block diagram(FBD) 功能块图

·Structured text (ST) 结构化文本

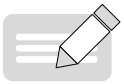
·Sequential function chart (SFC) 顺序功能图

·Continuous function chart (CFC) 连续功能图

其中，LD、FBD、ST、SFC 是基于 IEC 61131-3 标准，CFC 是 IEC 61131-3 标准的一个扩展。

不论用户选用哪种语言，编程界面中的基本编辑方法是通用的，为编程带来很大方便。

- 标准的编辑器功能，如支持“复制”（Ctrl+C）、“粘贴”（Ctrl+V）和“删除”（Del）等快捷键；
- 标准的 <Ctrl>、<Shift> 按键多项选择；
- 支持功能键 <F2> 启动输入助手，系统根据具体环境提供相对应的输入提示或选择。



NOTE

关于 SFC、CFC 内容详情，请参见帮助文档。

5.2 结构化文本语言 (ST)

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

例如：

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

5.2.1 表达式

表达式是一种结构，对它求值后，这个值可以在指令中使用。

表达式由操作符和操作数组成。一个操作数可以是一个常量，变量，功能调用或其他表达式。例如：

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

表达式的求值以特定的操作符优先权定义的顺序，按操作符对操作数进行求值。表达式中具有最高优先权的操

作符应首先进行求值，接着是下一个较低优先权的操作符等，从高到低依次求值完成。优先权相等的操作符应按表达式中书写的从左到右的顺序进行。

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

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

下表记录了 ST 语言的操作符 :

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

5.2.2 ST 指令

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

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

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

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

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

WHILE	WHILE 循环	J:=1; WHILE J<= 100 AND ARR[J] <> 70 DO J:=J+2; END_WHILE;
REPEAT	REPEAT 循环	J:=-1; REPEAT J:=J+2; UNTIL J= 101 OR ARR[J] = 70 END_REPEAT;
EXIT	退出循环	EXIT;
CONTINUE	继续循环下次执行	CONTINUE;
JMP	跳转	label: i:=i+1; JMP label;
;	空语句	;

1 赋值指令

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

- “:=”为一般赋值，右值直接赋给左值，左值和右值相等。

  例如：Var1 := Var2 * 10;

  完成执行后，Var1 值为 Var2 的 10 倍。

- “S=”为置位赋值，表示如果右值为 TRUE，左值变量变为 TRUE（置位），直到调用 R= 命令来初始化。
- “R=”为复位赋值，表示如果右值为 TRUE，左值变量变为 FALSE（复位）。用于复位 S= 指令置位的变量。

例如：a S= b;

一旦 b 为 TRUE 后，a 会保持 TRUE, 即使 b 变为 FALSE 后。

2 功能块的调用

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

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

注：TON 功能块通过 "TMR:TON" 实例化。

实例化语法：`<FB instance name> :<FB variable >;`

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

3 RETURN 指令

RETURN 指令表示当前置条件为 TRUE 时，离开此 POU。

语法：

```
RETURN;
```

示例

```
IF b=TRUE THEN
  RETURN;
END_IF;
  a:=a+1;
```

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

4 IF 指令

通过 IF 关键字，可以判断执行条件，根据执行条件，执行相应的指令。

语法：

```
IF < 布尔表达式 1> THEN
  <IF_ 指令 >
  {ELSIF < 布尔表达式 2> THEN
  <ELSIF_ 指令 1>

  ELSIF < 布尔表达式 n> THEN
  <ELSIF_ 指令 -1>
  ELSE
  <ELSE_ 指令 >}
END_IF;
```

{} 内部分是可选的

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

示例

```
IF temp<17
  THEN heating_on := TRUE;
  ELSE heating_on := FALSE;
END_IF;
```

这里，当温度低于 17 度时加热打开，否则它保持关闭。

5 CASE 指令

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

语法 :

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

CASE 指令根据以下流程处理：

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

示例

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

6 FOR 循环

通过 FOR 循环，可以编写重复处理逻辑。

语法：

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

{} 内的部分是可选的。

INT_Var 是计数器，是整数类型，只要计数器 <INT_Var> 不大于 <END_VALUE>，<Instructions> 会被执行。在执行 <Instructions> 之前首先要检查该条件，如果 <INIT_VALUE> 大于 <END_VALUE>，<instructions> 不会被执行。

当 <Instructions> 执行一次后 , <INT_Var> 自动增加 <Step size>。<Step size> 可以是任意整数值，如果不写此参数，默认值为 1。当 <INT_Var> 大于 <END_VALUE> 时，循环停止。

示例

```
FOR Counter:=1 TO 5 BY 1 DO
  Var1:=Var1*2;
END_FOR;
Erg:=Var1;
```

假设 Var1 默认值是 2, 经过 FOR 循环后，它的值是 32。

7 WHILE 循环

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

语法 :

```
WHILE <boolean expression> DO
  <instructions>
END_WHILE;
```

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

示例:

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

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

8 REPEAT 循环

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

语法 :

```
REPEAT
  <instructions>
  UNTIL <Boolean expression>
END_REPEAT;
```

执行逻辑：

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

示例:

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

9 CONTINUE 语句

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

示例：

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

10 EXIT 语句

EXIT 指令用于退出 FOR, WHILE, 或 REPEAT 循环。

11 JMP 语句

JMP 指令可用于无条件的跳转到指定标签处的代码行。

语法：

```
<label>:
JMP <label>;
```

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

示例：

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

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

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

12 注释

在结构化文本中有两种写注释的方法。

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

注释可以在 ST 编辑器声明或实现部分的任意地方。

注释的嵌套：注释可以放置在其他注释中

示例：

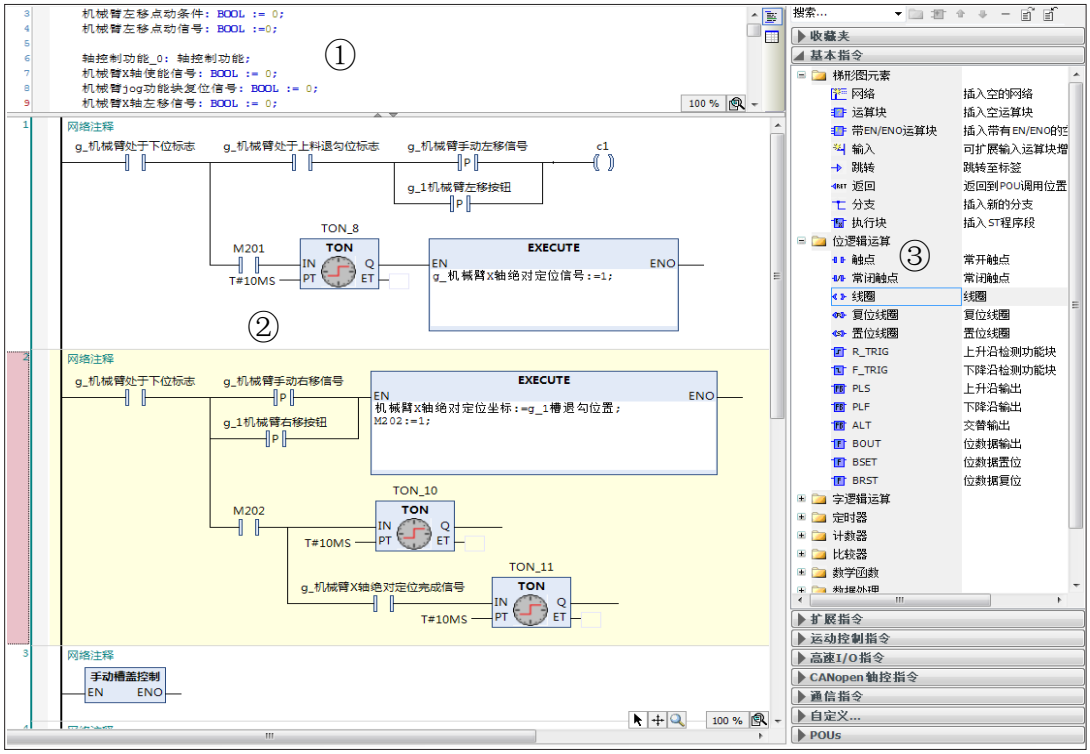
```
(*
  a:=inst.out; (*to be checked*)
  b:=b+1;
*)
```

5.3 梯形图 (LD)

5.3.1 概述

梯形图是图形化的编程语言，跟电路图的结构相近。梯形图包括一系列网络（也叫节，下文统一以“网络”代替），每个网络由左侧的竖线（电源轨，能流线）开始。一个网络由触点，线圈，运算块（函数、功能块、程序、执行块、动作、方法）、跳转、标签和连接线等构成。

网络左侧母线为能流线，其状态永远为 TRUE，母线后会连接触点、运算块、线圈等元素。每个触点均分配布尔变量。如果变量值为 TRUE，相当于开关闭合，条件会从沿着连接线从左向右传递，否则开关断开。在网络右侧的线圈，接收到从左侧传来的“开”或“关”信号，相应的 TRUE 或 FALSE 会被写到与线圈关联的布尔变量中。梯形图编辑界面如下图。

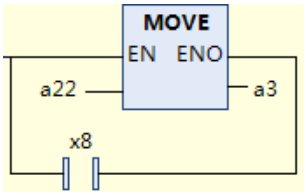


NOTE

◆ 1 – 变量定义区；2 – 梯形图编程区；3 – 工具箱。
◆ 梯形图语言中支持嵌入 EXECUTE 执行块，用于插入 ST 语言程序段。

梯形图主要元素包括触点、线圈、运算块、分支、注释等。通过插入、拖拽、划线、复制粘贴操作在网络中添加这些元素，形成梯形图执行逻辑。对于梯形图界面字体、操作数及注释显示可以通过【工具】-【选项】-【FBD/LD 编辑器器】设置。

梯形图支持监控、写入值、强制值、断点等在线调试功能。



5.3.2 梯形图元素

梯形图元素包括网络、触点、线圈、运算块、执行块、分支、跳转、标号、返回。

触点、线圈、运算块输入输出都和操作数关联，操作数可以是变量、常量 (TRUE、FALSE、1,2 等)、地址，具体见变量定义。

LD 元素位于工具箱（菜单命令【视图】-【工具箱】）中，如下图所示。在工具箱中除了常规下元素、梯形图元素和 IEC 标准操作符（如布尔操作符、数学操作符）外，还包括功能块，当前程序中定义的 POU。

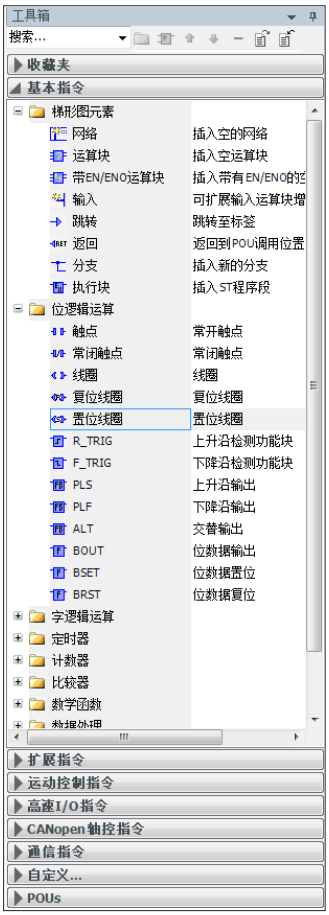


图 5-1 LD 工具箱

1 网络

图标 -

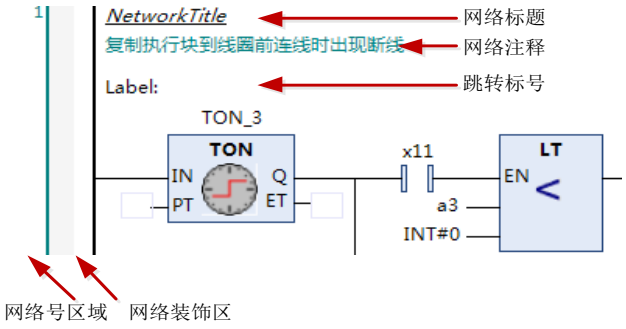
梯形图由一系列网络组成，其它所有的梯形图元素都位于网络内。每个网络都由左边的网络序号指示。

网络中可插入标题（网络总结性说明）和网络注释（网络比较详细的说明）。网络标题、网络注释是否显示通过选项配置 (【工具】-【选项】-【FBD/LD 编辑器】-【常规】) 来控制。

网络中可插入标签，标签位于网络标题和网络注释下面，作为跳转目标。

网络也可以处于注释状态，通过菜单命令【切换网络注释状态】来使能或者禁用网络。

网络序号和网络内容之间有个区域叫网络装饰区，用来显示断点标志和书签位置。



2 触点

图标 -

触点分常开触点和常闭触点。触点传递 ON（TRUE）和 OFF（FALSE）值，触点为 BOOL 型变量，如果变量值为 TRUE，常开触点向右传递 ON（TRUE），否则传递 OFF（FALSE），常闭触点传递值相反。

触点可以增加延信号功能，选中触点右键菜单命令【边沿检测】，可以把触点变为上升沿触发触点（触点变量值由 FALSE 变为 TRUE 时触点向右传递 ON）或者下降沿触发触点（变量值由 TRUE 变为 FALSE 时触点向右传递 ON）。

3 线圈

图标 -

线圈位于网络的末尾。左侧逻辑运算结果，赋值给线圈变量。线圈变量只能为 BOOL 类型，TRUE 表示（ON），FALSE 表示（OFF）。线圈仅支持向上或者向下插入并联线圈。

线圈分为线圈、取反线圈、置位线圈、复位线圈。通过右键菜单命令或者快捷键可以进行 4 种线圈类型之间的切换；

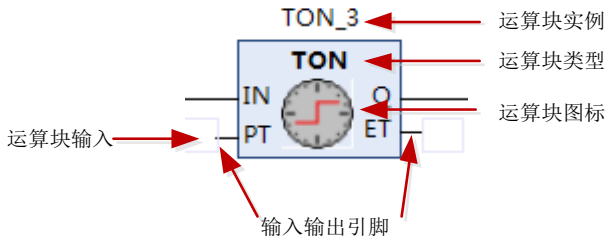
- 线圈：左侧逻辑运算结果直接赋值给线圈变量。
- 取反线圈：把左侧逻辑运算结果取反赋值给线圈变量。
- 置位线圈：如果线圈左侧的状态值为 ON（TRUE），则把线圈变量的值设置为 ON（TRUE），并且一直保持这个状态，直到下次该变量被复位线圈重置为 OFF（False）。
- 复位线圈：对置位线圈进行复位。

4 运算块

图标 -

运算块可以是操作符、函数、功能块、程序、动作、方法。如果为功能块类型，则运算块框上面会增加编辑框来显示功能块实例。

一个运算块至少包含一个输入和一个输出。运算块主要有由下图组成：



运算块分普通运算块和 En/Eno 运算块。

- EN/ENO 类型运算块：除了包含运算块本身所带的输入和输出外，还增加 EN 输入和 ENO 输出。EN/ENO 运算块执行逻辑为：当 EN 为 TRUE 时执行运算块逻辑，执行完成后 ENO 为 TRUE，如果 EN 为 FALSE，

不执行运算块，ENO 为 FALSE。注意：EN/ENO 运算块输入连线只能连接在 EN 引脚，输出连线只能连接在 ENO 引脚。

- 运算块输入输出引脚：BOOL 型输入引脚可以增加取反、上升沿、下降沿信号。运算块输出连引脚可以增加取反信号。
- 多输入连线运算块：有多个输入且多个输入均连接到能流线上，如下图为两个输入连线的多输入连线运算块。由于多输入连线运算块有多个连线和能流线相连，所以多输入连线运算块不能再并联分支中，并且只能在第一个分支中。

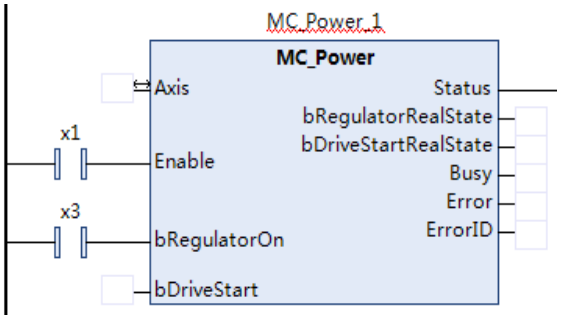


图 5-2 多输入连线运算块

5 执行块

图标 -

执行块是一个可以插入内嵌 ST 的块，在块中可以编辑 ST 语句。执行块可以放大缩小，最大为 1000*400。

6 分支

图标 -

分支形成一个非闭合的并联逻辑。

7 标签

图标 -

标签标明跳转位置，位于网络头部，通过跳转元素跳转到标签位置。跳转标签是字符串，需符合标识符命名规则。

8 跳转

图标 -

当跳转元素左侧的输入为 TRUE 时，跳转到指定的标签位置执行。跳转元素位于网络的最右侧 .

9 返回

图标 -

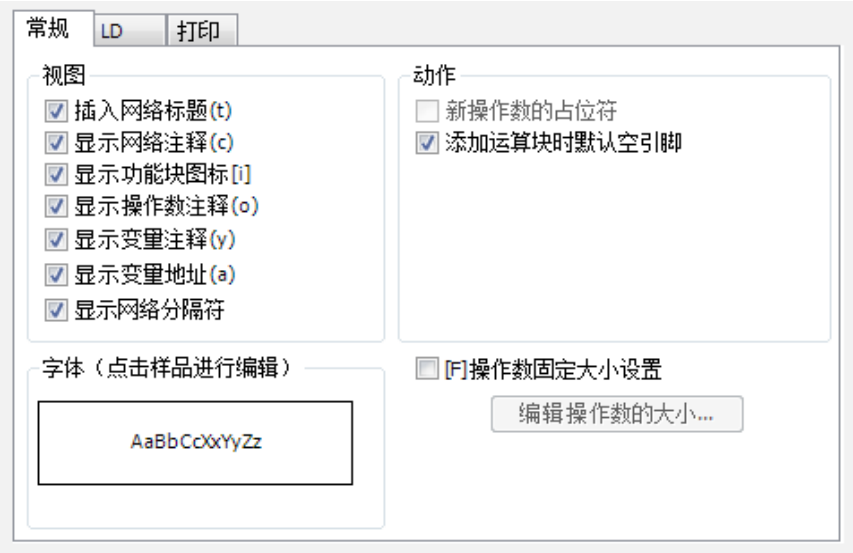
当返回元素左侧的输入为 TRUE 时，当前程序立即退出执行。返回元素位于网络的最右侧。

5.3.3 LD 编辑器选项

LD 编辑器选项用来控制 LD 界面的显示、单键命令设置、打印时显示方式。LD 编辑器选项通过菜单命令【工具】-【选项】-【FBD/LD】打开。LD 编辑器选项包含 3 个选项卡：常规、LD 和打印。

1 常规设置

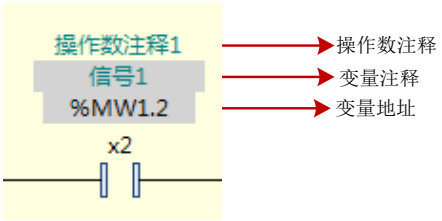
常规选项卡设置如下图。



LD 编辑器常规选项卡

视图

- 插入网络标题：如果激活此选项，梯形图每个网络可以插入标题、编辑标题。如果已经插入了标题会在当前网络最上面增加一行，用来显示标题，如果没有标题，标题行不会显示。插入标题通过菜单命令实现。
- 显示网络注释：如果激活此选项，梯形图每个网络可以编辑网络注释。如果增加了网络注释，在网络标题下增加一行显示网络注释，如果网络注释不存在，不会产生空白行显示网络注释。编辑网络注释通菜单命令实现。
- 显示功能块图标 - 如果激活此选项，如果运算块定义了图标，运算块中间会显示图标。标准操作符（如 ADD、SUB）和功能块（如 TON、TOF）都定义了图标，用户自定义的函数、功能块或者程序可以通过【右键对象】-【属性】-【位图】-【点击此处选择与工程有关的位图】来添加图片，形成运算块图标。
- 显示操作数注释：如果激活此选项，梯形图界面每个操作数上都可以编辑和显示操作数注释。操作数是一个编程概念，如变量、常量、地址都是操作数。由于梯形图中不一定都使用变量，如常量或者地址，这时可以通过操作数注释对它们进行注释。编辑操作数注释通过选择操作数字符串，然后右键菜单实现编辑。
- 显示变量注释: 如果激活此选项，梯形图界面变量上，会显示变量声明时的注释。变量注释来自于变量声明，不能编辑。



字体

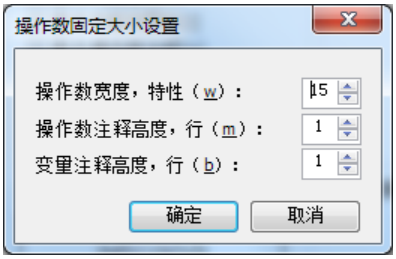
点击样本文字，弹出编辑器字体选择框，设置梯形图文字字体。默认字体为微软雅黑，使用小五号字体。字体范围主要包括操作数字符、注释。执行块字体使用文本编辑器字体（ST 文本、变量声明文本）。

动作

- 新操作符的占位符：未实现
- 添加运算块时默认空引脚：如果激活此选项，新增加运算块时，运算块输入和输出引脚用空字符，如果未激活此选项，运算块输入和输出引脚用”???”字符。

操作数固定大小设置

如果激活此选项，可以设置操作数固定宽带、操作费注释高度和变量注释高度。如下图。

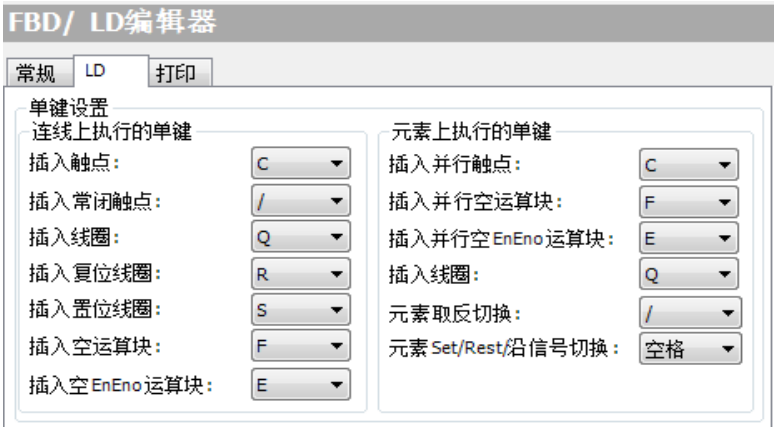


操作数固定大小设置

- 操作数宽度：设置操作数固定字符个数，默认 15 个。
- 操作数注释高度：设置操作数注释固定高度行数，默认 1 行。
- 变量注释高度：设置变量注释固定高度行数，默认 1 行。

2 LD 选项设置

LD 选项卡设置如下图。



单键设置

单键功能，通过单个按键进行编辑操作，包括连线上执行的单键和元素上执行的单键。连线上执行单键命令插入串行元素，元素上执行单键命令插入并行元素。

另外选择元素时可以对元素功能进行切换，如取反切换、延信号切换、置位 / 复位切换。取反切换作用对象为触点、线圈，使用 “/” 按键；延信号切换作用对象为触点，使用空格按键；置位 / 复位切换作用对象为线圈使用空格按键。

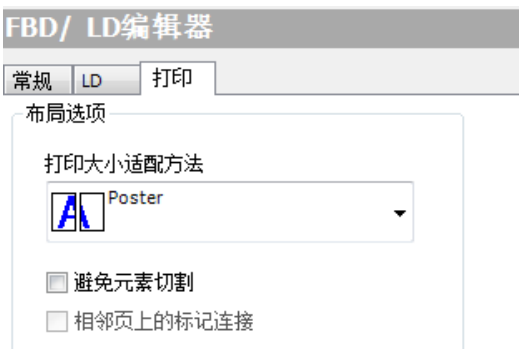
单键设置，设置单键功能按键字符。每种功能都有默认单键字符，但是可以根据个人喜好自行设置。


NOTE

连线上单键功能对应字符或者元素上单键功能对应字符不能相同，但是连线上单键字符和元素上单键字符可以相同。

3 打印

打印选项卡设置如下图。



布局选项

打印大小适配方式：

- Poster，正常比例打印，打印时如果当前页面高度显示不了此网络使用，使用下一个页面打印，如果页面宽度显示了整个网络，使用下一个页面打印剩余的。
- Shrink to fit the widest network，表示打印时压缩显示内容使所有网络都一个页面宽度内显示，并且如果一个页面高度显示不完整个网络，显示部分网络，剩下的网络在下一个页面显示。
- 避免元素切割：如果激活了此选项，表示当一个元素在两个页面之间都有显示，则把当前元素在下一个页面显示，只能在 Poster 打印方式设置。
- 相邻页上的标记连接：表示设置了避免元素切割后，为了表示前后连接关系，增加连接标记。

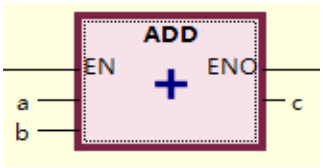
5.3.4 元素选择

选择是编辑的基础。选择对象可以是元件或者连线，选择可以单选也可以按下 Ctrl、Shift 多选，可以连续选择，也可以非连续选择。

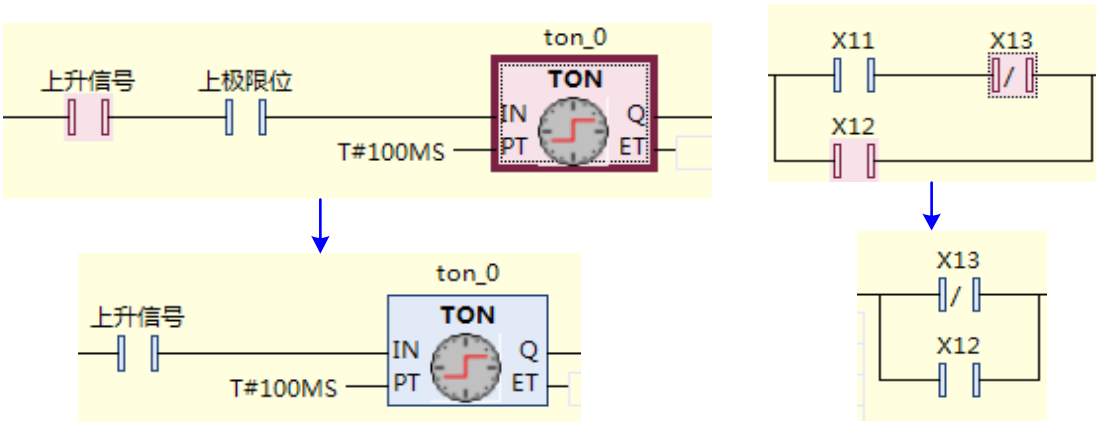
元素选择后元件会处于高亮状态，如触点被选中时显示形式为 ，其中外框的虚线表示触点处于焦点状态。选择的元素可以和其它元素粘贴并联，拖拽串并联。由于多选元素可能不连续，这就需要说明选择元素形成的结果逻辑。选择结果逻辑的原则是保证原来逻辑的一致性。支持鼠标框选、Ctrl 和 Shift 多选、全选。

1 元素选择形成的结果逻辑

- 单选时：选择触点、线圈，只包含选择的触点、线圈；选择运算块包括运算块本身、无连线输入操作数和输出操作数。如下图，选择 ADD 的运算块，粘贴时会包含 ADD 运算块本身、输入 a、输入 b、输出 c。

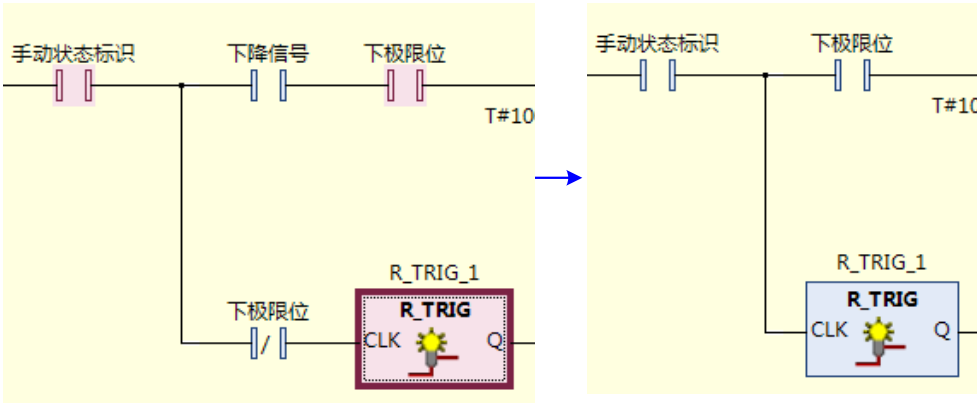


- 多选时：选择一条线上的串联（包含非连续选择）；选择并行线上元素（包含非连续选择），形成并行结果。

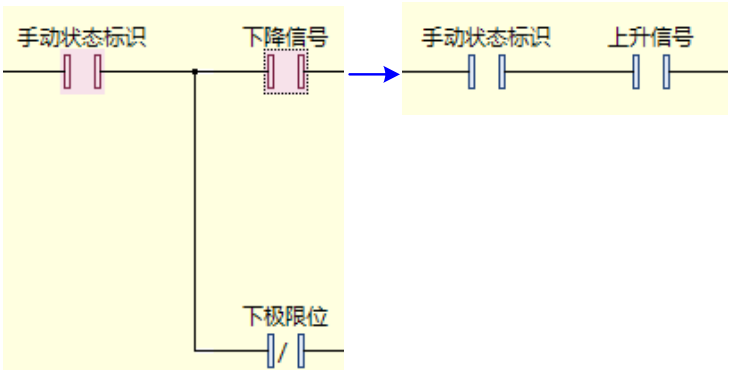


形成串并联逻辑

- 多选时：如果选择元素跨多分支，形成结果也跨多分支，不改变原来逻辑，但是如果只选择两个分支的，结果会把两个分支元素串联。

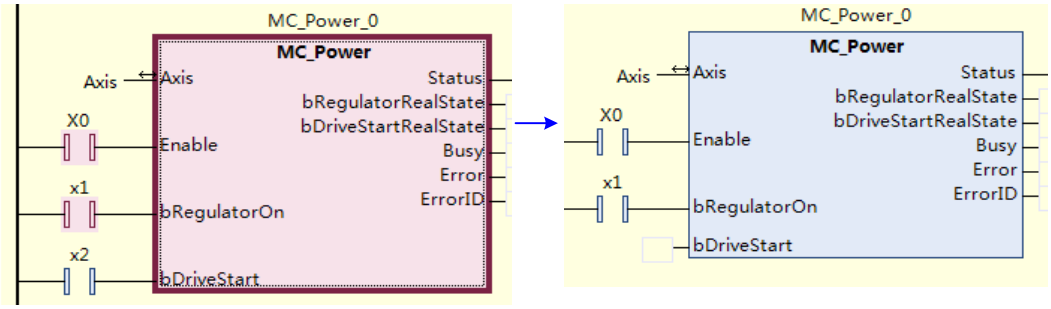


超过两个分支结果也多个分支

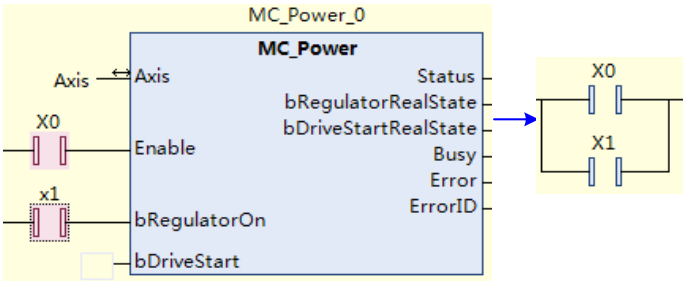


选择两个分支，两个分支结果串联

2 多选时：如果同时选择多输入连线运算块本身和多个输入连线上的元素，结果和选择一致；如果只选择多输入连线上元素，没有选择运算块，则把多个输入连线元素形成打开并联。



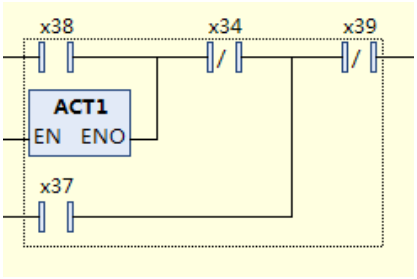
运算块和输入元素都选择，结果和选择一致



只选择输入元素，不选择运算块，形成打开并联逻辑

3 框选

从鼠标按下位置到鼠标松开位置之间矩形内的元素被选择。如下图：



NOTE

框选只能单网络，不能跨网络；鼠标按下的空白处网络开始。

4 Ctrl 和 Shift 多选

Ctrl 和 Shift 多选符合标准多选方式：

- Ctrl 多选：按下 Ctrl 时，如果当前元素没有选择则把当前元素增加到选择列表中；如果已经选择，则从选择列表中去除
- Shift 多选：从上次选择元素到本次选择元素矩形内的元素进行选择。

5 全选

使用 Ctrl+A 快捷键进行全选功能。全选会把所有网络选择。

5.3.5 标准编辑命令

梯形图标准编辑主要是我们常规编辑操作，如复制、粘贴、删除、剪切、撤销、恢复。使用标准的编辑快捷键

1 复制

对选择的元素进行复制。复制的结果也就是选择的元素。详见元素选择章节。

梯形图中可以复制的元素包括网络、触点、线圈、运算块、字符串、分支连线。

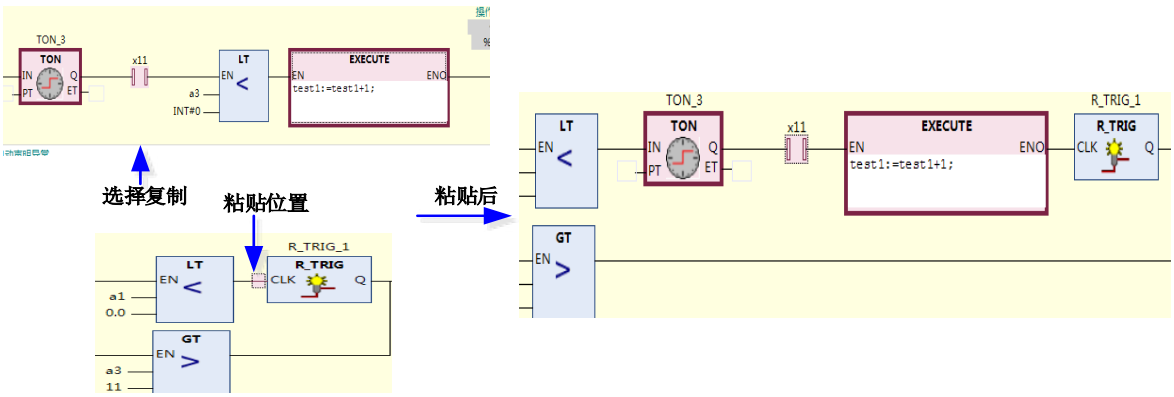
复制可以复制连续选择的元素，也可以复制非连续的，和选择有关。但是复制的元素只能单个网络内，如果跨网络选择，只能复制焦点元素网络选择的数据。

2 粘贴

粘贴是对复制的元素进行粘贴。粘贴规则如下：

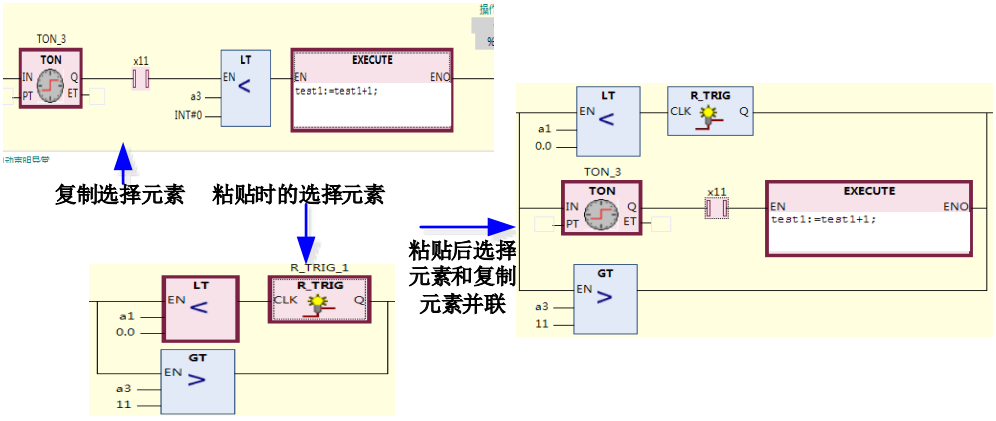
- 连线上粘贴。

在选择连线上粘贴时，连线位置插入复制元素，形成串联关系；



■ 元素上粘贴。

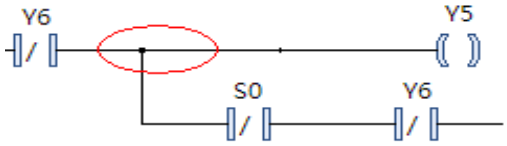
在选择元件上粘贴时，和所选择的批量元件生成并联关系。在形成并联关系时，被选择元件需要满足可并联条件，才能粘贴，即选择的元件必现满足：在一条线上、必须连续、不能跨越分支、开始元素和结束元素不能再并联分支内部和外部。



■ 线圈位置粘贴

由于线圈右侧不能存在元素，所以粘贴时有些特殊规则。跳转元素、返回元素规则和线圈相同，下面仅说明线圈。

1） 如果选择为线圈（选择元素并联），粘贴的元素位于一个新分支上，此分支位于线圈下面，如下图（选中 Y5 粘贴 S0 和 Y6，S0 和 Y6 在一个新分支上，分支位于 Y5 线圈下面）

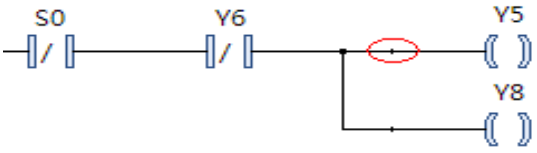


2） 如果选择为线圈之前的连线（选择连线串联），粘贴时，根据复制的元素内容，分为复制内容仅包含一个分支和复制内容包含多个分支

复制内容仅包含一个分支

1） 如果复制内容不包含线圈，复制元素直接和线圈串联；

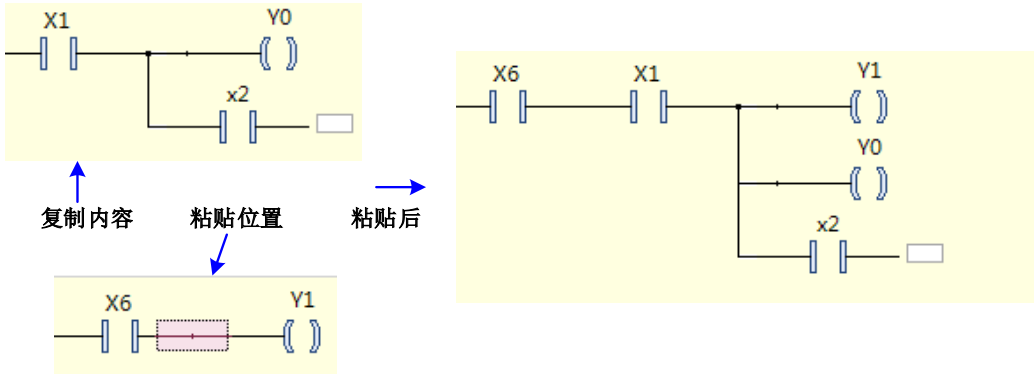
2） 如果复制元素包含线圈，则复制元素线圈之前的数据插入到连线选择位置，复制元素中的线圈和选择连线之后的线圈形成一个非闭合并联，如下图（选择 Y5 之前连线粘贴 S0、Y6、Y8，粘贴后，Y8 之前 S0、Y6 串联在 Y5 之前连线上，Y8 和 Y5 通过分支并联）。



复制内容包含多分支

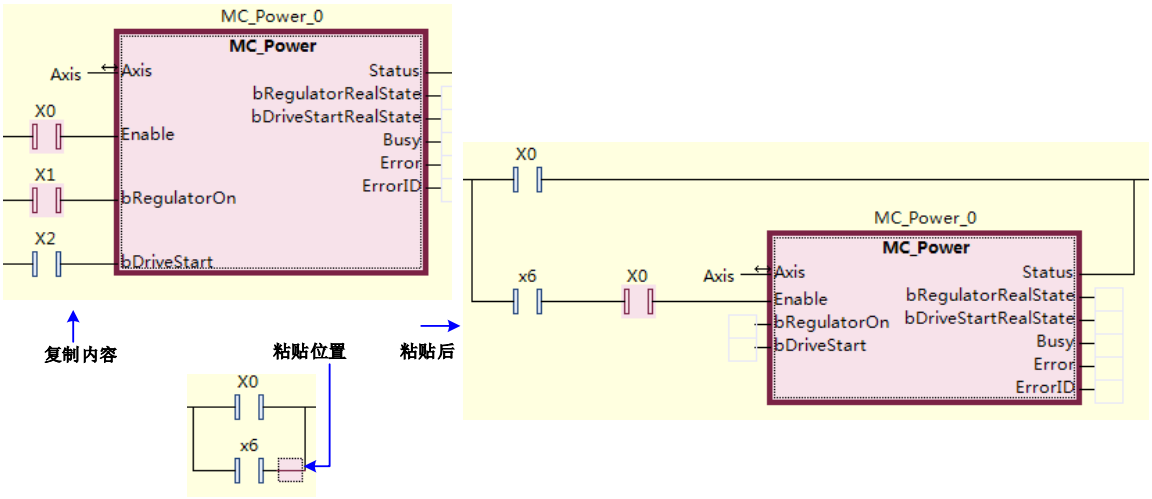
1） 如果复制内容水平最后一个分支不包含线圈，在选择连线位置插入复制内容；

2） 如果复制内容水平最后一个分支包含线圈，则把复制内容水平最后一个分支的线圈和选择连线之后的线圈并联，其它数据串并联不变，如下图：复制内容为 X1、Y0、X2 粘贴在 Y1 线圈之前的连线位置，粘贴后，Y0 位于 Y1 线圈下，X1 位于 Y1 之前的连线上。



■ 多输入连线运算块粘贴

多输入连线运算块只能在第一个分支非并联位置，所以在粘贴位置，如果不能包含多连线运算块，会删除复制运算块从第二个输入连线开始的连线元素；如下图，复制的运算块输入 X0、X1 触点，粘贴时 X1 被删除。

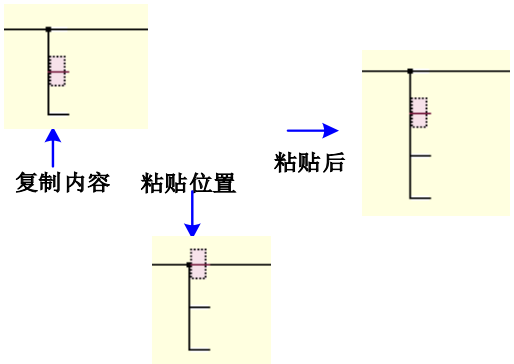


■ 网络粘贴

可以单选或者多选网络，进行复制粘贴，只有选择了网络才能粘贴复制的网络。

■ 单分支连线粘贴

单分支连线粘贴，用于增加单分支，和向上 / 下插入分支功能相同。复制单分支连线，可以在连线位置粘贴，粘贴在选择连线的下面，如下图：



3 删除

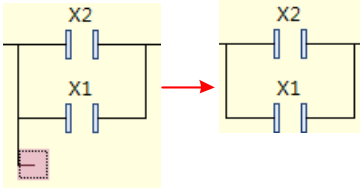
删除是对选择的元素进行删除。元素删除后，会选择下一个元素，以保证操作的连贯性。

删除可分为元素删除和连线删除：

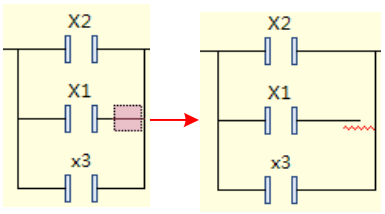
■ 元素删除 - 删除本身；

■ 连线删除 - 删除分支连线、垂直连线及垂直连线相连的左侧连线。

删除分支连线 - 空分支连线删除后，此分支也同时会被删除。



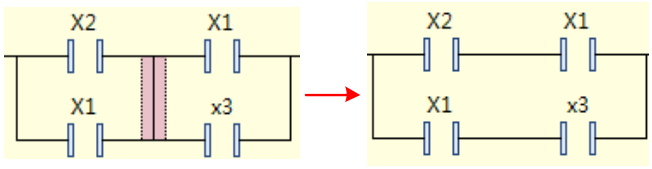
删除垂直连线相连的左侧连线 - 删除垂直连线左侧相连的连线，会把连线断开，分支打开。



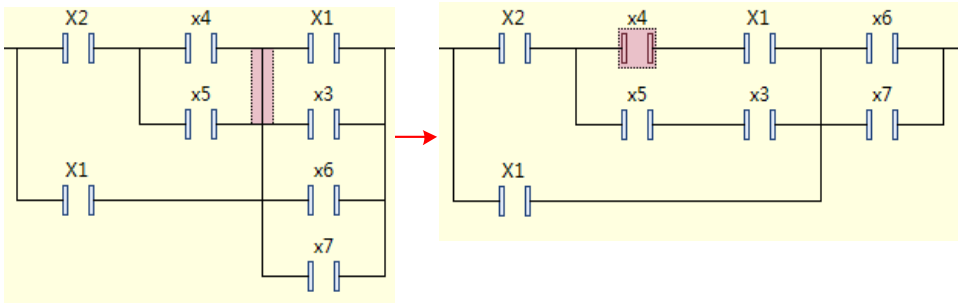
删除垂直连线 - 原则上，删除垂直连线后，此垂直连线不再存在。垂直连线删除后，会产生三个结果：合并分支、打开分支、分支左移。

1） 合并分支

如果选择的垂直连线左右两侧都在分支内，删除垂直连线后，两侧的分支会合并。

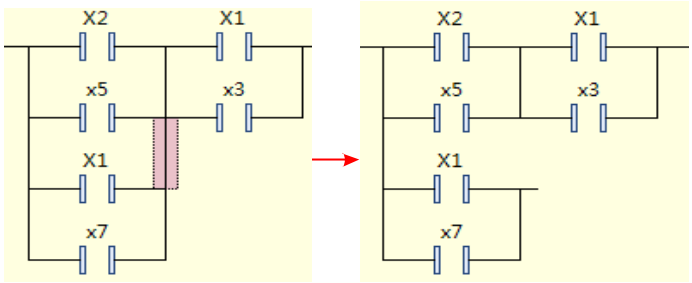


注意：如果删除垂直连线后产生桥式电路，这时会把右侧上下分支左移合并，右侧其它分支下移。如下图，删除后 x1、x3 左移合并，x6、x7 上移。



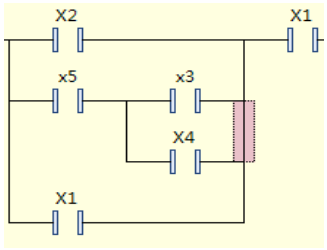
2） 打开分支

如果选择的垂直连线左侧在分支内，右上存在分支，右下不存在，删除垂直连线后会把左侧选择之下的分支断开。



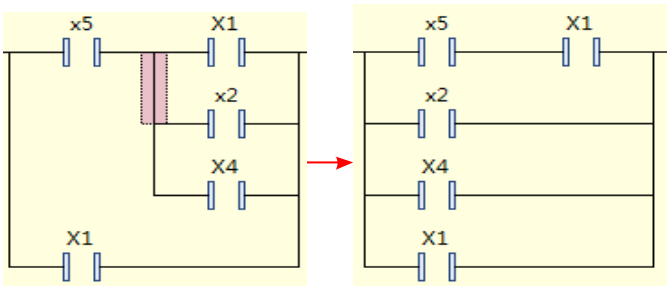
NOTE

如果删除垂直连线后产生桥式电路，则不能将其删除。



3） 分支左移

如果垂直连线右侧在分支内，左上存在分支，左下不存在，删除垂直连线后会把右侧选择之下的分支左移到上一个垂直连线上。



4 剪切

剪切功能是把选择的元素复制，以备粘贴，然后删除。

5 撤销 / 恢复

撤销：撤退到上一步编辑状态，并且恢复上一步选择元素。

恢复：恢复到下一步编辑状态，并且恢复下一步的选择元素。

5.3.6 LD 菜单命令

菜单命令可以通过右键菜单或者 LD 工具栏菜单执行的梯形图命令。

1 插入网络

包含插入网络命令和插入网络 (在下方) 两个菜单命令，可以通过拖动工具箱中“网络”来插入网络。

命令执行条件：先选择网络，在选择的网络上方 / 下方插入新网络。

插入网络：图标 - 快捷键：Ctrl + I，表示在选择网络上面插入一个空网络。

插入网络（在下方）：图标 - 快捷键：Ctrl + T，表示在选择网络下方插入一个空网络。

2 切换网络注释状态

图标 - 快捷键：Ctrl + O，把一个网络在注释状态和非注释状态进行切换。

在注释状态，整个网络的代码无效，代码不会执行，执行块也不能编辑。

命令执行条件：选择网络

3 插入网络头信息

网络头主要包含网络标题、网络注释和标号。

插入网络头相关命令包含插入标号、编辑网络标题和编辑网络注释。

- 编辑网络标题：图标 - ，编辑选择网络的标题。
- 命令执行条件：选择网络，并且选项中“显示网络标题”使能
- 编辑网络注释：图标 - ，编辑选择网络的注释。
- 命令执行条件：选择网络，并且选项中“显示网络注释”使能。
- 插入标号：图标 - ，给选择网络插入跳转标签，作为跳转元素跳转位置。
- 命令执行条件：选择网络。

4 插入运算块

包含插入运算块、插入空运算块、插入带 EN/ENO 的运算块、插入带有 EN/ENO 的空功能块和插入并行运算块（在下方）5 个菜单命令，用来插入操作符、功能、功能块和程序，也可以通过拖动工具箱中“运算块”或者“带有 EN/ENO 的运算块”来插入运算块。

前四个命令用来插入串联运算块，最后一个命令用于插入和选择元素并联的运算块。

插入位置：

1） 选择水平连线，在水平连线处插入一个运算块。

2） 选择元素，在选择元素左侧插入运算块。

- 插入运算块：图标 - 快捷键：Ctrl + B，弹出输入助手选择一个要插入的运算块。
- 插入空运算块：图标 - 快捷键：Ctrl + Shift + B，插入一个空的运算块，不弹出输入助手。可以在运算块类型处输入运算块类型。
- 插入带 EN/ENO 运算块：图标 - 快捷键：Ctrl + Shift + E，弹出输入助手选择一个要插入的运算块，此运算块带有 EN/ENO 输入和输出。带 EN/ENO 的运算块，当 EN 为 TRUE 时运算块才执行，为 FALSE 时不执行，ENO 和 EN 结果相同。
- 插入带有 EN/ENO 的功能块: 插入一个空的运算块，不会弹出输入助手，此运算块带有 EN/ENO 输入和输出。
- 插入并行运算块（在下方）：在选择的元素下方插入一个空的运算块。选择元素可以是触点、运算块。

5 插入执行块

插入执行块：图标 - ，在当前选择位置插入串联执行块，也可以通过拖动工具箱中“执行块”来插入。

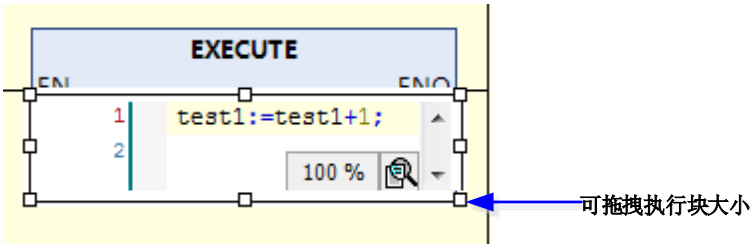
插入位置：

1） 选择水平连线，在水平连线处插入一个执行块；

2） 选择元素，在选择元素左侧插入运算块。

执行块是一个可以编辑 ST 语句的块，单键文本区域进行编辑；执行块只有 EnEno 输入和输出。

执行块大小拖拽：在编辑状态可以拖拽执行块边框，实现执行块大小控制，如下图：



6 插入输入

插入输入：图标 - 快捷键：Ctrl + Q，给可变输入运算块增加输入。

可变输入运算块：ADD、+、MUL、*、SEL、AND、&、OR、|、XOR、MAX、MIN、MUX。

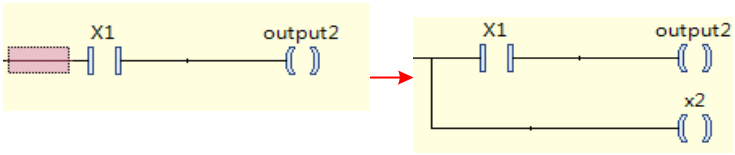
插入位置：当选择输入引脚时，在当前输入引脚之前加入一个输入；选择运算块，增加的输入引脚位于最后位置。

7 插入线圈

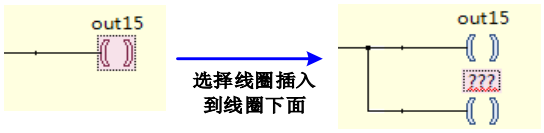
包含插入线圈、插入置位线圈和插入复位线圈三个菜单命令，也可以通过拖动工具箱中“线圈”、“置位线圈”和“复位线圈”元素来插入线圈。

- 命令执行条件：选择位置不能位于并联分支中，也不能位于多输入连线运算块输入位置。
- 插入线圈：图标 快捷键：Ctrl + Shift + A，在当前位置输出一个线圈。
- 插入位置：

1） 选择水平连线或者元素，在水平连线处或者元素左侧插入一个线圈，此线圈和连线通过非闭合分支处理。



2） 如果选择线圈、返回或者跳转，则在当前选择元素下面插入新线圈。



插入的线圈缺省变量为“???”，需要输入所需的变量或常量，可以使用输入助手（功能键 <F2>），直接从变量列表中选择输入。

- 插入置位线圈：图标 - ，表示在当前位置插入一个置位线圈。操作方式和上述“插入线圈”相同。
- 插入复位线圈：图标 - ，表示在当前位置插入一个复位线圈。操作方式和上述“插入线圈”相同。

8 插入触点

包含插入触点、插入常闭触点、插入并联下触点和插入并联上触点四个菜单命令，也可以通过拖动工具箱中“触点”、“常闭触点”元素来插入触点。

- 插入触点：图标 - 快捷键：Ctrl + K，表示在当前位置前串联插入一个常开触点。

插入位置：

1） 选择水平连线，在水平连线处插入一个触点；

2） 选择一个网络，那么新触点插入到最后；

3） 选择元素，新触点插入到元素左侧。

触点缺省变量名为“???”。 点击文本输入所需的变量或常量，可以使用输入助手（功能键 <F2>），直接从变量列表中选择输入。

- 插入常闭触点：图标 - ，表示在当前位置串联插入一个常闭触点。操作方式和上述“插入触点”相同。
- 插入并联下触点：图标 - ，快捷键：Ctrl + R，表示在选择元素下并联插入一个常开触点。选择元素可以是触点或者运算块。
- 插入并联上触点：图标 - ，快捷键：Ctrl + P，表示在选择元素上面并联插入一个常开触点。操作方式和上述“插入并联下触点”相同。

9 插入分支

包含插入分支、在上面插入分支两个菜单命令，也可以通过拖动工具箱中“分支”元素来插入分支。插入的分

支是一个非闭合的线。

■ 插入分支：图标 - ，快捷键：Ctrl + Shift + V，表示在所选连线位置插入一个分支。

插入位置：

1） 选择连线，在连线下方插入一条分支。

2） 选择触点或者线圈，在选择元素之前插入。

如下图，每个选择位置表示一个分支。

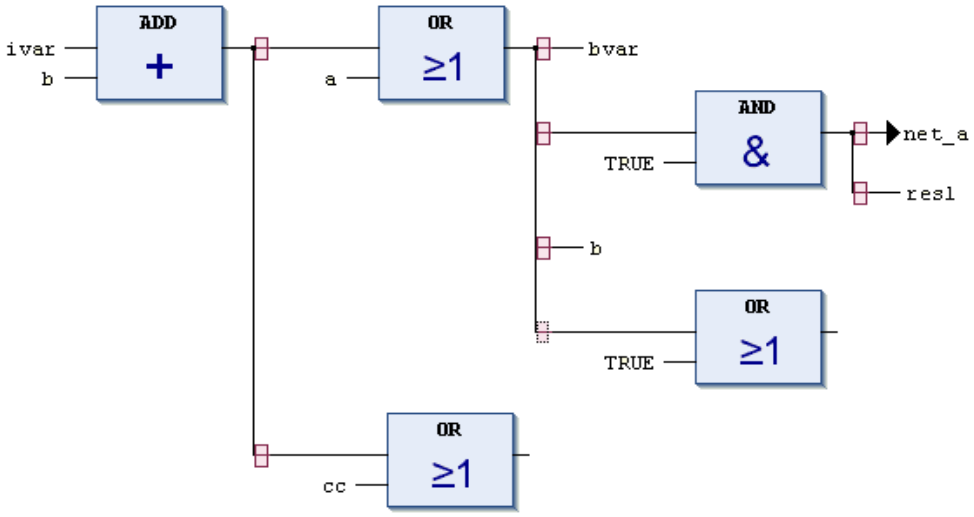


图 5-3 分支标记

■ 在上面插入分支：图标 - ，表示在所选分支上面增加一个分支。选择分支连线时，才能执行。

10 跳转和返回

包含插入跳转命令和插入返回两个菜单命令。跳转和返回属于程序执行顺序控制命令，正常程序按照网络顺序从上到下，从左到右依次执行。可以通过拖动工具箱中“跳转”来插入跳转元素或者工具箱中的“返回”来插入返回元素。

跳转、返回和线圈一样，必须在最右侧，所以插入跳转和插入返回的规则和插入线圈一样，详见插入线圈命令。

■ 插入跳转：图标 - 快捷键：Ctrl + L，表示插入一个跳转元素，跳转到指定标号位置。

跳转位置为网络中的标号，也就是说可以从一个网络跳转到另一个网络。当跳转前的输入条件满足时才能执行跳转。

■ 插入返回：图标 - ，表示插入一个返回元素。当输入条件满足时，当前 POU 执行返回，返回到调用它的 POU。

11 取反

图标 - 快捷键：Ctrl + N，对运算块输入、运算块输出、跳转条件、返回条件、触点值或者线圈取反。

取反命令可以在两种位置执行：

1） 元素取反：主要是触点和线圈。取反后触点和线圈内增加一个斜线（/）。

2） 连线取反：主要包含运算块输入连线、运算块输出连线、线圈输入连线、跳转输入连线、返回输入连线，取反后连线处增加一个圆圈。

可取反位置如下图：

图 5-4 可“取反”位置

再次执行取反命令，返回到初始状态。

12 边沿检测

图标 - 快捷键：Ctrl + E，表示对触点、运算块输入连线、线圈输入连线、跳转元素输入连线、返回元素输入连线增加边沿触发功能。

上升沿检测相当于 R_TRIG 功能块，下降沿相当于 F_TRIG 功能块。

边沿检测命令可以在两种位置执行：

1） 触点边沿检测：选中触点执行边沿检测命令，触点增加边沿检测功能，表示上升沿；表示下降沿。

2） 连线增加边沿检测：运算块输入连线、线圈输入连线、跳转元素输入连线、返回元素输入连线，执行块边沿检测命令，连线上增加延信号符号，上升沿检测符号为；下降沿检测符号为。只有 BOOL 型输入连线才能添加边沿检测功能。

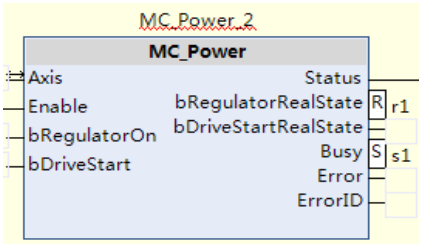
13 置位 / 复位

图标 - 快捷键：Ctrl + M，该命令用于增加置位或者复位输出功能。置位输出显示为“S”，复位输出显示为“R”。可以多次执行此命令，在置位、复位和正常输出之间切换。

置位 / 复位命令可以在两种位置执行：

1） 选择线圈，执行此命令变为置位、复位线圈。置位线圈：。复位线圈：

2） 选择运算块 BOOL 型输出连线（非主输出），设置置位复位功能，如下图：



14 设置输出连接

图标 - 快捷键：Ctrl + W，当运算块有多个输出时，可以修改主输出的引脚（一个运算块只有一个主输出，主输出和后继的元素相连）。如下图：

选择要修改的输出引脚，执行此命令，修改输出连接。

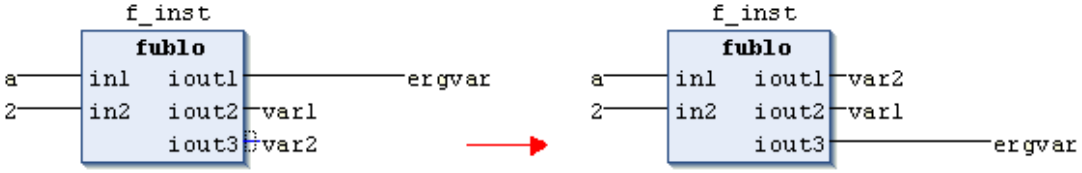


图 5-5 更改输出连接

15 更改输入、输出引脚显示

包含更新参数和删除未使用的 FB 调用参数两个菜单命令。

更新参数：图标 - 快捷键：Ctrl + U，表示更新所选的运算块的输入和输出参数。如果运算块的输入或者输出参数发生变化时，通过执行“更新参数”命令，更新运算块的输入和输出参数。

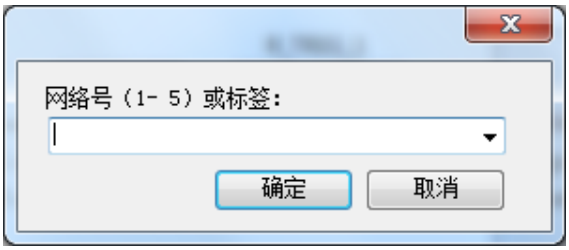
删除未使用的 FB 调用参数：图标 - ，删除未使用的运算块输入引脚和输出引脚，也就是一个运算块中输入或者输出为“???”或者空时，这些输入和输出将不再显示。

16 转换为 LD 语言

显示为梯形图逻辑：快捷键：Ctrl + 2。把 FBD/IL 转换为 LD 语言；由于 FBD、IL 暂时不再支持，旧工程可以通过此命令把 FBD/IL 转换为 LD 语言显示。

17 跳转网络

转到...：跳转到指定的网络。弹出跳转网络输入框，指定跳转的网络编号。



18 编辑操作数注释

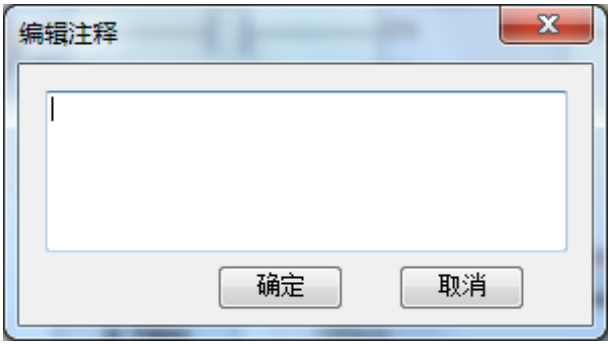
编辑操作数注释：编辑选择的操作数的注释。

命令执行条件：

- 在 FBD/LD 选项中，激活选项“显示操作数注释”。
- 需要选择操作数字符串。

操作数是逻辑概念，输入变量、常量、地址都是操作数，如运算块输入变量、触点关联变量、线圈关联变量、运算块实例等。

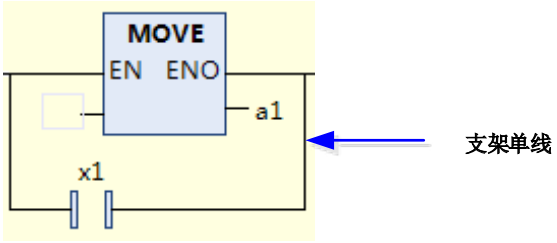
选择操作数字符串，执行此命令显示编辑操作数注释对话框，对操作数注释进行编辑，如下图。



19 并联模式切换

Toggle Parallel Mode：切换并联分支并联模式。并联模式分为顺序型并联分支和短路型并联分支。

- 顺序型并联的并联支架使用单线，分支输出结果为单个分支输出取或操作，如下图，通过 OR 来形成分支结果。



■ 短路型并联的并联支架使用双线，分支输出结果需要考虑每个分支是否包含非运算块。

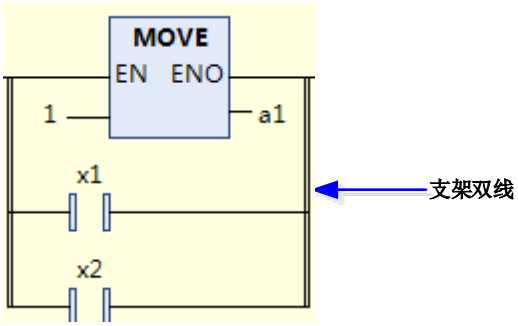
非运算块的分支作为条件，如果非运算块的分支有一个结果为 True，都不去执行有运算块的分支（可以理解为触点短路运算块）。如下图，只有 X1 分支和 X2 分支结果都不是 TRUE，才去执行第一个 Move 分支指令。

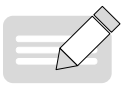
非运算块分支需要同时满足以下条件：

1） 此分支只包含触点或者操作符运算块。

2） 触点不能具有延信号。

3） 操作符运算块不能是 EnEno 类型，操作符运算块输入连线不能包含取反、或者延信号。




NOTE

考虑到短路型分支的复杂性，不建议使用短路型分支。

20 设置分支起点 / 终点

设置分支起点 / 终点：图标 -  ，快捷键：Ctrl + D。

设置分支启动 / 终点命令用于把启动和终止连接起来，功能同划线功能。

连接两个点，首先执行此命令，设置起点，这时起点位置显示  ，表示连接开始点，然后再选择终止点，执行此命令，这时会把启动和终点连接起来，具体连接逻辑见划线功能。

5.3.7 单键命令

单键命令通过单个字符快捷键进行快速编辑。可以在连线上执行单键命令，也可以在元素上执行单键命令。连线上执行单键插入串行元素；元素上单键命令用于插入并行元素或者元素功能切换。

每个命令对应的字符，可以在【选项】-【FBD/LD】-【LD】页面设置，具体见选项设置。

连线上执行的单键

■ 插入触点：默认单键为“C”。

■ 插入常闭触点：默认单键为“/”。

■ 插入线圈：默认单键为“Q”。

■ 插入复位线圈：默认单键为“R”。

■ 插入置位线圈：默认单键为“S”。

- 插入空运算块：默认单键为“F”。
- 插入空 EnEno 运算块：默认单键为“E”。
- Set/Reset/ 延信号切换：默认单键为“空格”。用于运算块 BOOL 型输入、输出连线切换；当选择运算块 BOOL 型输入连线，执行延信号切换；选择运算块非主输出 BOOL 型连线，执行 Set/Reset 切换。

元素上执行的单键

- 插入并行触点：默认单键为“C”。选择的元素可以是触点、运算块。
- 插入并行空运算块：默认单键为“F”。选择的元素可以是触点、运算块。
- 插入并行空 EnEno 运算块：默认单键为“E”。 选择的元素可以是触点、运算块。
- 插入线圈：默认单键为“Q”。 选择的元素可以是线圈、返回、跳转元素。
- 元素取反切换：默认单键为“/”。选择触点进行常开和常闭触点切换；选择线圈对线圈进行取反切换。
- 元素 Set/Reset/ 延信号切换：默认单键为“空格”。选择触点时，进行上升沿、下降沿和正常信号切换。选择线圈进行 Set、Reset 和正常线圈切换。

5.3.8 划线功能

划线功能主要是把划线起点和终点两个位置连接起来。划线功能首先要满足以下条件：

- 起点或者终点位置必须是连线，并且是可以选择的位置（能流线除外）。
- 运算块输入、输出引脚可以拖拽互换位置，所以靠近运算块引脚连线区域（大概 11 个像素）是拖拽引脚区域，不能划线。

从划线结果来看，划线功能分为三类：划线并联（增加并联分支），闭合分支、拆分分支。

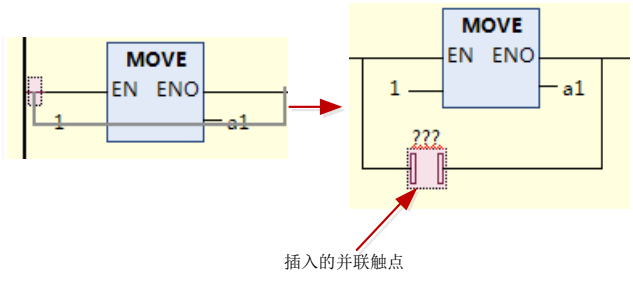
1 划线并联

当划线起点和终点在一个分支上时，自动增加一个触点和起点、终止并联，如下图：

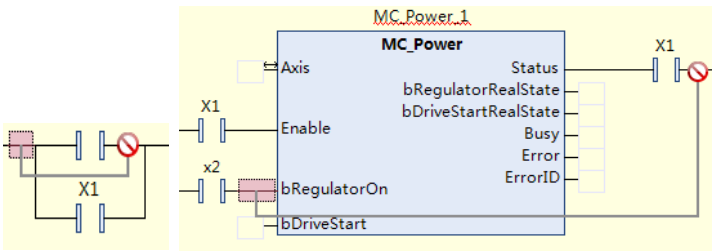
1） 如果在一个分支内划线，在开始和结束位置之间自动并联一个触点；

2） 如果从打开分支终止连线开始划线到其它分支，表示把打开分支闭合；

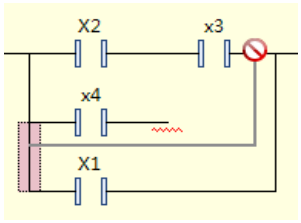
3） 如果相邻两个上下划线，表示把上下两个分支拆分。



- 划线并联起点和结束点必须满足可并联条件，不能跨越并联分支内外、不能从多连线运算块输入到输出。

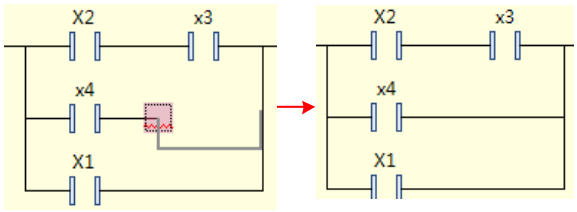


- 不能跨越打开分支。

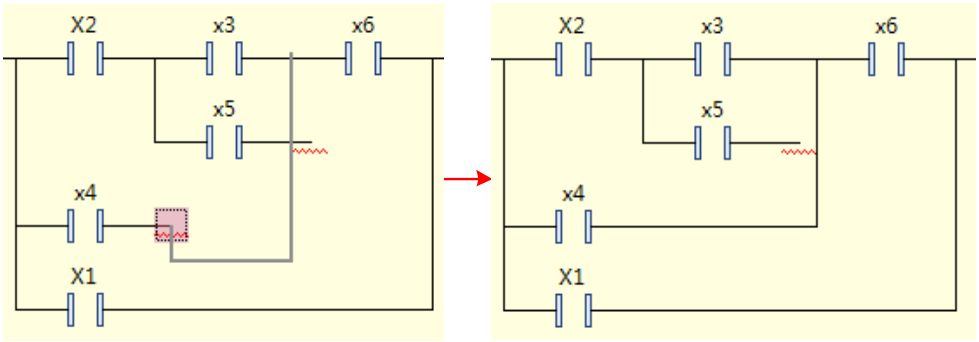


2 闭合分支

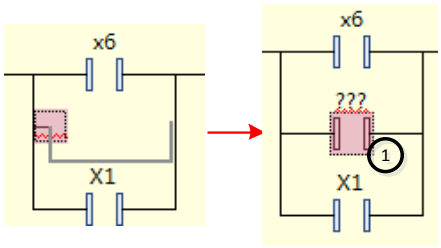
当划线起点为打开分支终止连线时，划线到另外一个可以闭合的连线上会把当前打开分支闭合，如下图：



■ 闭合分支功能可以跨越打开的分支。

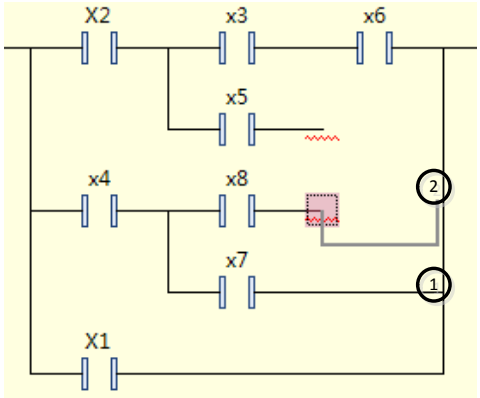


■ 如果打开分支只有一个终止连线，闭合时，自动添加空触点。



1：自动添加空触点

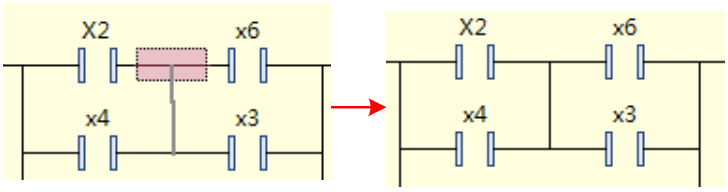
■ 如果打开分支对应的层级分支是闭合的，此打开分支只能和右侧垂直连线划线闭合。



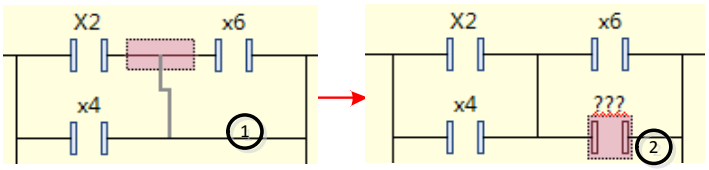
1：打开分支对应的层级分支是闭合的 2：只能向右侧垂直连线划线闭合

3 拆分分支

当划线起点和终点为并联两相邻分支时，划线时会把起点、终点连线两侧拆分为两个并联。

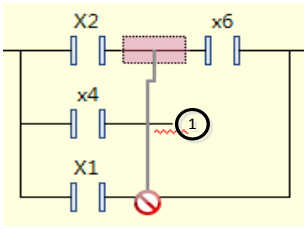


■ 如果起点或者终点左侧或者右侧没有元素，自动在没有元素的一侧增加空触点。



1：右侧没有元素 2：增加的空触点。

■ 拆分分支不支持跨越打开分支。



1：打开的分支。

5.3.9 拖拽操作

梯形图可以进行元素拖拽，主要包括工具箱拖拽元素到网络、梯形图界面内元素拖拽及跨界面拖拽。

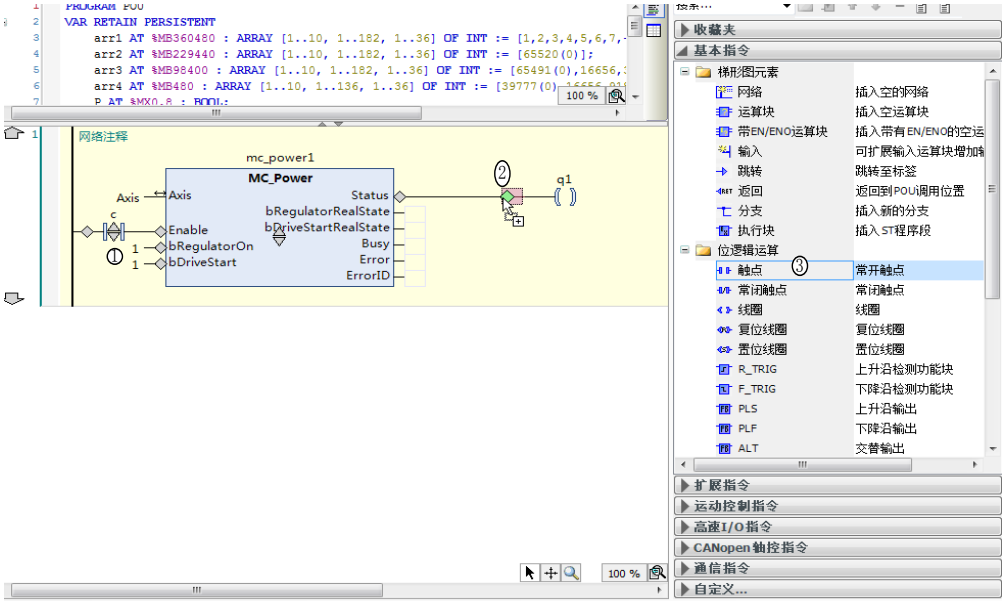
当拖拽元素时，梯形图界面会显示可拖拽位置。可以拖拽位置有三种显示形式：

■ 菱形显示：使用菱形，表示可拖拽到当前位置串联插入。

■ 上下三角显示：使用上下三角箭头，表示在当前元素上方或者下方插入并联元素。

■ 上下箭头显示：使用上下箭头，表示向上或者向下增加一个网络，并拖拽到新添加的网络。

当拖拽元素到插入位置时，每种图形内部会变成绿色，如，表示要插入到此位置。拖拽显示如下图所示。



① 插入并联 ②拖拽上去后绿色，表示可以插入 ③插入触点

1 工具箱元素拖拽

工具箱中的元素可以拖拽到梯形图编辑器中。

工具箱元素主要包含基本指令、扩展指令、运动控制、高速 I/O，CANOpen 轴控指令、通讯指令、POUs；另外用户可以自定义类别并添加指令，还可以把指令添加到工具箱中。

梯形图元素在基本指令中。

POUS 主要包含当前工程中定义的程序、功能块、函数、方法、动作。最大显示不能超过 200 个，如果工程中超过 200 个，为防止显示混乱，不再显示 POUS 的内容。

拖拽时，每个元素有限定的拖拽位置，可拖拽规则如下：

- 触点可以拖拽到触点、运算块（包含执行块）上并联，拖拽到连线上串联。
- 运算块可以拖拽到触点、运算块（包含执行块）上并联，拖拽到连线上串联。
- 线圈可以拖拽到非闭合并联、非多连线运算块输入连线上串联，可以拖拽到线圈、返回、跳转上面或者下面。

2 编辑界面元素拖拽

在梯形图界面，可以拖拽选择的元素从一个位置到另外一个位置。拖拽元素可以在本编辑界面内，也可以拖拽到其它梯形图编辑界面。

拖拽的元素是选择的元素（见元素选择章节），可以多选、单选。

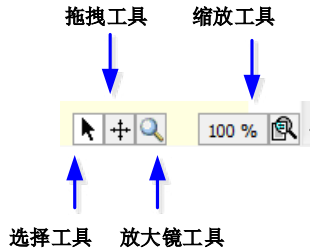
拖拽包含正常拖拽和复制式拖拽（按下 Ctrl，拖拽）。正常拖拽，选择的元素拖拽过去后，会把原来选择的元素删除；复制式拖拽，选择的元素拖拽过去后，选项的元素保留。

拖拽功能都是按照标准操作方式实现。

于多选或者单选的拖拽规则，和标准编辑命令（粘贴）一致。

5.3.10 图形显示工具

梯形图图形显示工具，用来控制梯形图显示模式，包含选择工具、拖拽工具、放大镜工具和缩放工具，默认梯形图是选择工具模式。图形显示工具位于梯形图界面右下侧，如下图：



■ 选择工具

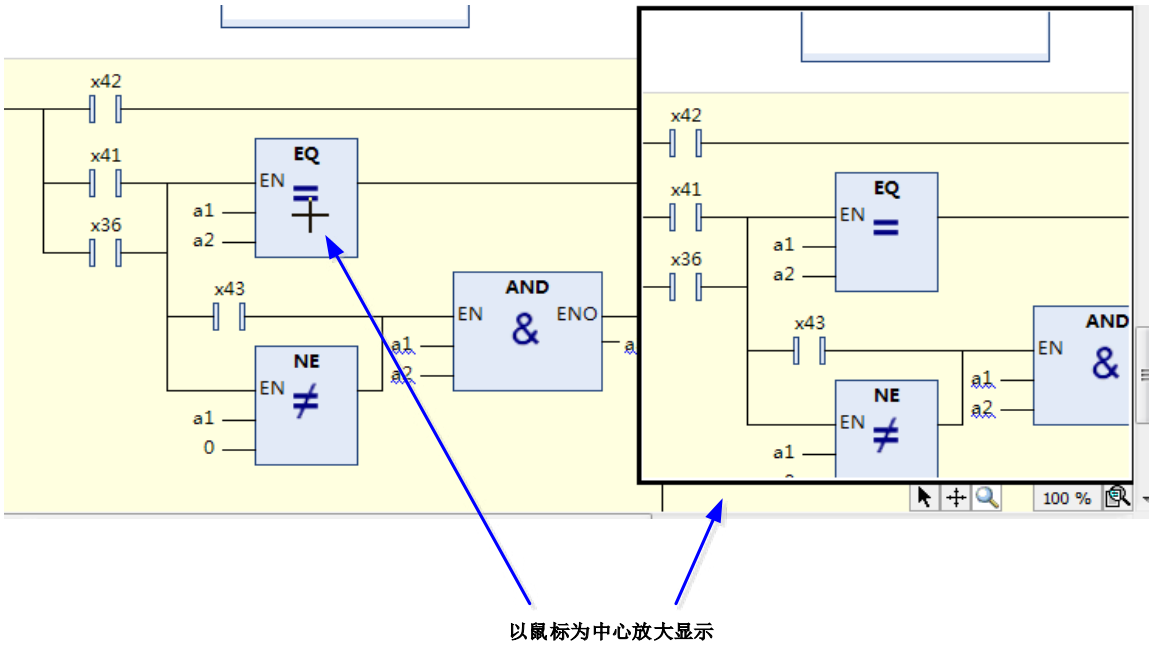
选择工具是默认显示工具，在选择工具模式下，鼠标样式为，可以进行选择元素，从而进行编辑操作。

■ 拖拽工具

拖拽工具模式下，鼠标样式为，可以对区域进行拖拽显示操作。

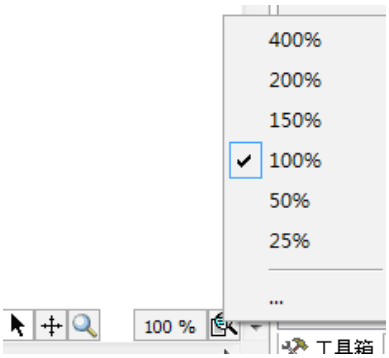
■ 放大镜工具

放大镜模式下，鼠标样式为，以鼠标为中心，进行放大显示，有放大镜作用。如下图：



■ 缩放工具

缩放工具可以显示当前界面缩放比例，也可以设置缩放比例，如下图：



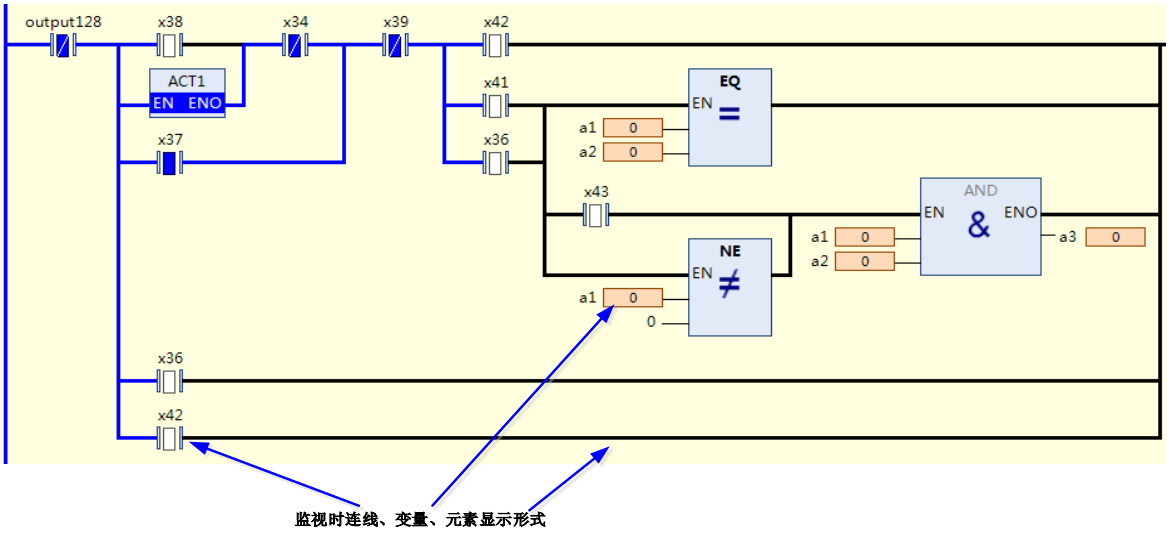
另外点击“…”，弹出缩放比例设置对话框，输入希望的缩放比例，如下图：

5.3.11 LD 调试

梯形图提供了强大的调试功能，除了已有的监控表监控，梯形图还提供了在线模式下的监视、操作数写入、强制值写入、断点和单步调试功能。

1 监视

在线模式下，梯形图界面中的连线、元素、操作数变量等通过特定的形式来表达执行结果。如下图。



■ 监控连线

1） 对于 BOOL 型值连线，当导通时（值为 TRUE）时，显示蓝色粗线，没有导通时显示黑色粗线。

2） 非 BOOL 型值连线（运算块输入、输出中的整形变量、时间类型变量、浮点数变量等），使用细线，并且值为零时使用黑色细线；不为零使用蓝色细线。

■ 监控元素

1） 触点导通时，常开触点显示 或者常闭触点显示；触点不导通时，常开触点显示或者常闭触点显示。

2） 线圈导通时，正常线圈显示或者取反线圈显示显示；线圈不导通时，正常线圈显示或者取反线圈显示显示。

3） 对于 EnEno 运算块，由于 EnEno 运算块只有 En 为 True，才执行运算块本身逻辑，为了使 EnEno 运算块本身能够一目了然了解此运算块是否执行（运算块是否使能），对运算块类型文本显示做了区分，如果此运算块执行了（En 输入为 True），运算块类型显示黑色字体，没有执行显示灰色字体（运算块禁用），如下图：

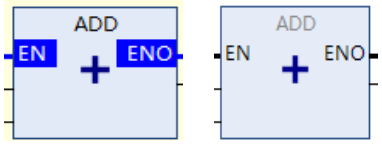


图 5-6 运算块执行显示 运算块未执行显示

■ 监控变量

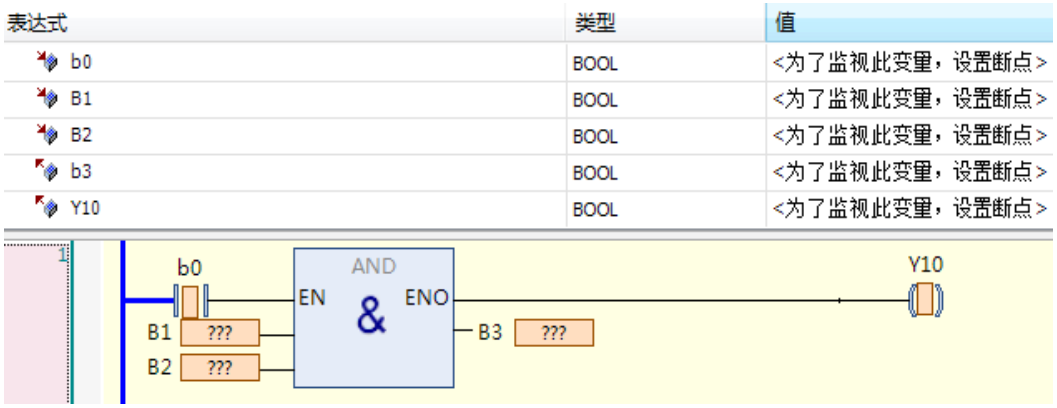
1） 监控变量根据类型不同显示宽度不同，以减少占用空间，对于不定长的，如字符串、枚举类型（显示枚举名），默认长度为 12 个字符，如果显示不完，使用…替换，通过信息提示来完整显示；对于定长的，如整数、浮点数等根据最大长度显示。

2） 可以拖拽监控变量到监控变量列表。

3） 可以更改变量显示模式，执行菜单命令：菜单【调试】-【显示模式】。

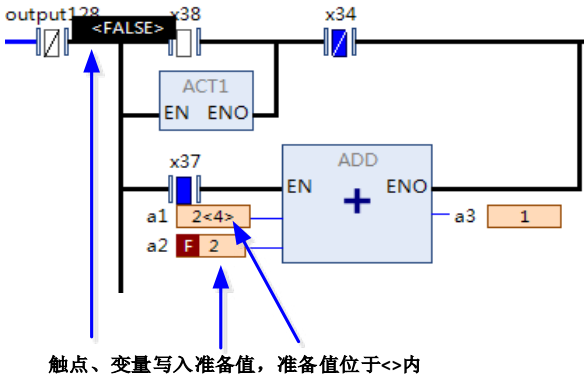
NOTE

由于函数和方法是即时执行，只有临时数据，所以登录后，方法和函数不能直接监控。如果需要监控函数和方法，需要在函数和方法中添加断点，中断执行，才能监视，如下图。



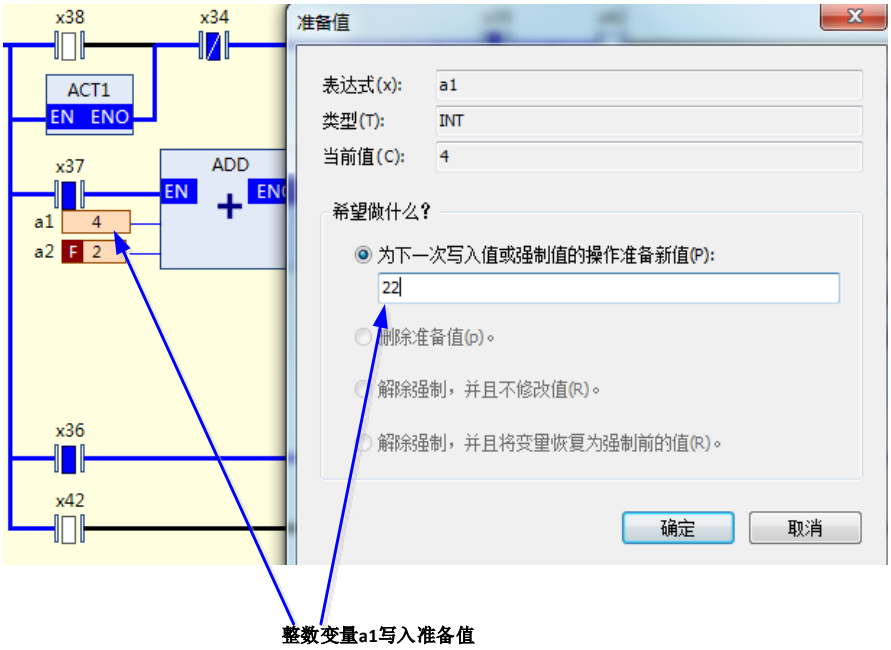
2 写入和强制

梯形图触点、线圈和变量可以写入准备值，然后执行调试菜单下的“写入值”命令或者“强制值”命令，给变量写入值或者强制值。在写值或者强制值之前，需写入准备值，如下图：



■ 对于触点、线圈和 BOOL 型变量，通过双击元素位置或者变量值位置，进行 TRUE、FALSE 准备值切换。如双击触点或者线圈中间位置，准备值进行切换。

■ 对于非 BOOL 型变量，双击变量值位置，弹出准备值对话框，输入准备值，如下图：

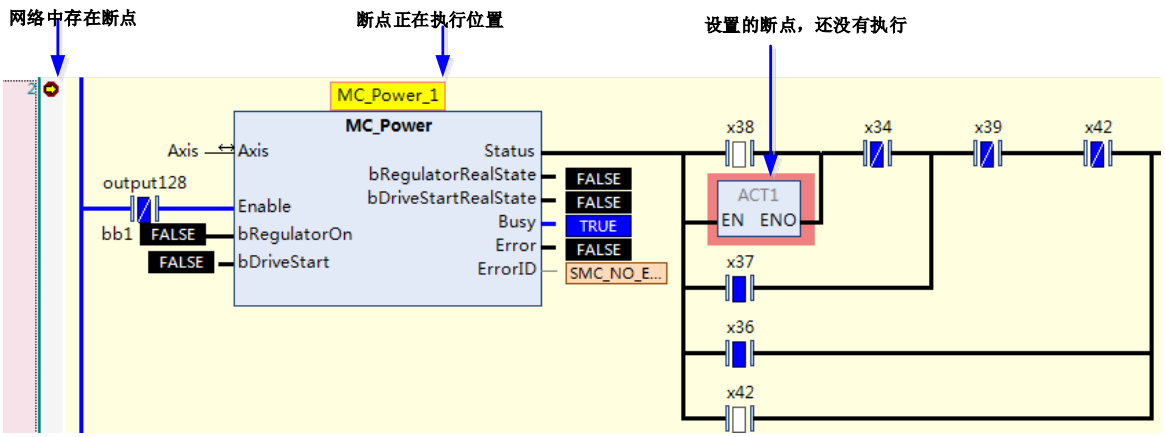


- 写入强制值后，值的最前方，增加 F 标识，标识此值是强制值。
- 把强制值释放，通过菜单命令【在线】-【释放值】

3 断点

LD 支持断点功能，如果添加断点后，程序执行到断点位置自动中断，可以进行程序调试，支持跳入、跳过、跳出、运行到光标位置等操作。

添加断点后，添加断点的位置（元素）用一个浅红色的矩形框表示，当执行到断点位置，正在执行的断点位置用一个黄色的矩形框表示；如果网络中存在了断点，则网络装饰区域的圆点，如下图。



由于梯形图是图形化的，而断点在有逻辑语句的地方才可以添加，梯形图为了优化性能，并不是所有地方都会有逻辑语句，也就是，不是所有地方都能添加断点，例如触点位置、非 EnEno 操作符运算块位置不能添加断点。

断点一般在变量值可能发生变化的地方、程序的分支处或者另外一个 POU 调用的地方，如 POU，输出变量赋值等地方。可以打开断点对话框（菜单【视图】-【断点】）查看所有可能的断点位置。

断点主要在可以添加到以下位置：

- 网络开始位置，表示网络中第一个可能的断点位置，给网络增加断点时，自动增加到第一个断点位置。
- 不包括非 EnEno 操作符的运算块，如 FB、动作、程序调用、执行块等。
- 线圈、返回、跳转元素位置。

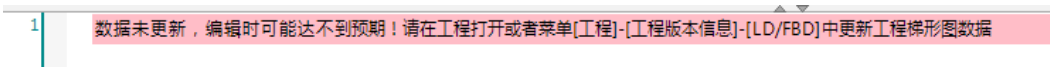
5.3.12 梯形图数据更新

对于 InoProShop(V1.4.0 及之前的版本)，如 InoProShop(V0.0.9.10)，InoProShop(V1.1.0)，InoProShop(V1.2.0) InoProShop(V1.2.60.0)， InoProShop(V1.2.70.1)，打开的梯形图数据需要升级，才能编辑使用梯形图优化版本的功能。

梯形图数据升级有两种方式：

- 在工程打开时，弹出的工程版本信息对话框 -【LD/FBD】选项卡，选择下图中所有更新标志，然后点击“确定”按钮；
- 通过菜单命令【工程】-【工程版本信息…】，弹出工程版本信息对话框，切换到【LD/FBD】选项卡，选择下图中所有更新标志，然后点击“确定”按钮。

如果梯形图数据没有更新，在第一个网络，显示更新说明信息，如下图：



NOTE

使用前梯形图必须更新。

第 6 章 汇川指令库

6 汇川指令库

6.1 指令速查表

6.1.1 指令简介

可编程控制系统中，使 CPU 完成某种操作或实现某种功能的命令及多个命令的组合称为指令，指令的集合称为指令系统。

6.1.2 指令分类

CoDeSys 指令安照实现方式的不同分为功能和功能块两类。以功能方式实现的指令，在使用的时候无需声明（实例化）。以功能块方式实现的指令，在使用的时候需声明实例名。因 AM600 在沿用 3S 基本指令（参见附录）的基础上，提供的指令非常丰富，支持如高速 I/O、CANopen、EtherCAT 远程计数等各大功能，本章节仅介绍运动控制相关指令。3S 基本指令的概览说明请参见附录。

6.1.3 运动控制指令速查表

运动控制指令又分为 SM3_Basic、SM3_CNC 和 SM3_ETC、CmpHCBasic、IoDrvCANopenAxis、CmpHSIO 指令库。由于 SM3_Basic、SM3_CNC 和 SM3_ETC 由 3S 提供，其详细功能说明可参考软件帮助说明资料，故下文仅罗列出指令一览表，不再赘述。对于由汇川提供的 CmpHCBasic、IoDrvCANopenAxis、CmpHSIO 等指令库，本章节将提供详细说明。

1 SM3_BASIC

分为基本驱动指令、文件操作指令和 PLCOpen 指令。

基本驱动指令

指令类型	指令说明	指令名	指令分类
制动控制	制动状态设置	SMC3_BrakeControl	功能块
	获取制动状态	SMC3_BrakeStatus	功能块
配置指令	保存多圈编码器轴位置（重启后）	SMC3_PersistPosition	功能块
	保存单圈编码器轴位置（重启后）	SMC3_PersistPositionSingleturn	功能块
	保持逻辑轴位置（重启后）	SMC3_PersistPositionLogical	功能块
	设置控制模式	SMC_SetControllerMode	功能块
诊断指令	检查轴通信状态	SMC_CheckAxisCommunication	功能块
	获取轴最大设置加减速度	SMC_GetMaxSetAccDec	功能块
	获取轴最大设置速度	SMC_GetMaxSetVelocity	功能块
	获取滞后错误	SMC_GetTrackingError	功能块
	轴位置监测	SMC_InPosition	功能块
	测量旋转轴转动距离	SMC_MeasureDistance	功能块

指令类型	指令说明	指令名	指令分类
轴直接控制指令	轴位置超限监测	SMC_CheckLimits	功能块
	无检测写入轴位置	SMC_FollowPosition	功能块
	无检测写入轴位置和速度	SMC_FollowPositionVelocity	功能块
	无检测写入轴参数	SMC_FollowSetValues	功能块
	无检测写入轴速度	SMC_FollowVelocity	功能块
错误处理指令	删除之前产生的功能块错误	MC_ClearFBError	功能块
	读取上次功能块错误	SMC_ReadFBError	功能块
	错误格式化字符串	SMC_ErrorString	功能块
回零指令	回零	SMC_Homing	功能块
缩放指令	改变轴缩放比和轴类型	SMC_ChangeGearingRatio	功能块

2 文件操作指令

指令类型	指令说明	指令名	指令分类
CAM 指令	从文件加载凸轮表数据	SMC_ReadCAM	功能块
	写凸轮表数据到文件	SMC_WriteCAM	功能块
诊断指令	轴诊断日志记录	SMC_AxisDiagnosticLog	功能块

3 PLCOpen 指令

指令类型	指令说明	指令名	指令分类
附加指令	间隙补偿	SMC_BacklashCompensation	功能块
	读取轴当前设置位置	SMC_ReadSetPosition	功能块
	设置转矩	SMC_SetTorque	功能块
凸轮表附加指令	根据凸轮表主轴参数计算从轴位置、速度、加减速最大和最小值	SMC_CAMBounds	功能块
	根据凸轮表主轴参数计算从轴位置最大和最小值	SMC_CAMBounds_Pos	功能块
	凸轮表注册并获取挺杆信息	SMC_CamRegister	功能块
	在可视化部分显示凸轮表	SMC_CamEditor	功能块
	根据凸轮表和主轴信息获取从轴位置、速度和加速度	SMC_GetCamSlaveSetPosition	功能块
	获取当前挺杆状态	SMC_GetTappetValue	功能块
主 / 从控制指令	从轴开始以凸轮控制执行	MC_CamIn	功能块
	从站停止以凸轮控制执行	MC_CamOut	功能块
	建立主从轴凸轮控制关系	MC_CamTableSelect	功能块
	设置主从轴的速度齿轮比	MC_GearIn	功能块
	设置主从轴位置齿轮比	MC_GearInPos	功能块
	取消主从轴齿轮比设置	MC_GearOut	功能块
	设置主从轴相位偏移	MC_Phasing	功能块

指令类型	指令说明	指令名	指令分类
单轴控制指令	设置轴在不同的时间对应的加速度	MC_AccelerationProfile	功能块
	停止轴的运动（可以被打断）	MC_Halt	功能块
	轴开始回零	MC_Home	功能块
	控制轴运动到指定的绝对位置	MC_MoveAbsolute	功能块
	使轴在上次位置的基础上增加指定的移动距离	MC_MoveAdditive	功能块
	使轴开始以相对位置运行	MC_MoveRelative	功能块
	轴叠加一个运动	MC_MoveSuperImposed	功能块
	设置轴以恒定的速度运行	MC_MoveVelocity	功能块
	设置轴位置在不同的时间对应的位置	MC_PositionProfile	功能块
	使能轴	MC_Power	功能块
	读取轴当前的位置	MC_ReadActualPosition	功能块
	读取当前轴错误状态	MC_ReadAxisError	功能块
	读取运动控制 Bool 类型的参数值	MC_ReadBoolParameter	功能块
单轴控制指令	读取轴的当前状态	MC_ReadStatus	功能块
	读取运动控制参数值	MC_ReadParameter	功能块
	复位轴错误	MC_Reset	功能块
	停止轴的运动（不可以打断）	MC_Stop	功能块
	设置轴速度在不同的时间对应的速度	MC_VelocityProfile	功能块
	写入运动控制 Bool 类型的参数值	MC_WriteBoolParameter	功能块
	写入运动控制参数值	MC_WriteParameter	功能块
	终止功能块和事件的联系	MC_AbortTrigger	功能块
	数字凸轮开关控制	MC_DigitalCamSwitch	功能块
	读取当前的转矩	MC_ReadActualTorque	功能块
	读取当前的速度	MC_ReadActualVelocity	功能块
	设置轴的位置	MC_SetPosition	功能块
	事件触发记录一个轴位置	MC_TouchProbe	功能块
	控制轴运动到指定的绝对位置后并以一定的速度继续运动	SMC_MoveContinuousAbsolute	功能块
	使轴开始以相对位置运行，运动到指定的位置后以一定的速度继续运动	SMC_MoveContinuousRelative	功能块
	点动控制	MC_Jog	功能块
	单步运动控制，可以控制运动距离	SMC_Inch	功能块

4 SM3_CNC

分为文件操作指令、CNC 运动控制指令、CNC 变换指令。

文件操作指令

指令类型	指令说明	指令名	指令分类
文件操作指令	读取 NC 文件	SMC_ReadNCFile	功能块
	读取 OutQueue 队列文件	SMC_ReadNCQueue	功能块

5 CNC 运动控制指令

指令类型	指令说明	指令名	指令分类
坐标转换指令	以长方体方式实现坐标转换	SMC_DetermineCuboidBearing	功能块
	3D 坐标转换	SMC_CoordinateTransformation3D	功能块
	逆 3D 坐标转换	SMC_InvCoordinateTransformation3D	功能块
	计算新坐标系统中的坐标向量	SMC_TeachCoordinateSystem	功能块
	计算新坐标系统中单位矢量的 RPY- 角度 (对应旧坐标系统)	SMC_UnitVectorToRPY	功能块
	围绕 Z 轴旋转运动路径数据	SMC_RotateQueue2D	功能块
	运动路径数据按比例缩放	SMC_ScaleQueue3D	功能块
	运动路径数据根据矢量进行转换	SMC_TranslateQueue3D	功能块
轴直接控制指令	以速度方式控制轴	SMC_ControlAxisByVel	功能块
	以位置方式控制轴	SMC_ControlAxisByPos	功能块
	以位置速度方式控制轴	SMC_ControlAxisByPosVel	功能块
G 代码查看指令	转换 G 代码数据位字符串	SMC_GCodeViewer	功能块
OutQueue 队列指令	给 OutQueue 增加 GEOINFO 数据	SMC_AppendObj	功能块
	删除 OutQueue 中 GEOINFO 数据	SMC_DeleteObj	功能块
	获取 OutQueue 中 GEOINFO 数据个数	SMC_GetCount	功能块
	获取 OutQueue 中 GEOINFO 数据	SMC_GetObj	功能块
	从 OutQueue 尾部开始查找获取 GEOINFO 数据	SMC_GetObjFromEnd	功能块
	初始化 OutQueue	SMC_OutQueueInit	功能块
	设置 OutQueue 数据容量	SMC_SetQueueCapacity	功能块
运动控制指令	去除循环路径	SMC_AvoidLoop	功能块
	检查路径是否脱离一个矩形范围并输出在范围内的路径	SMC_CheckForLimits	功能块
	检查路径速度并输出在范围内的路径	SMC_CheckVelocities	功能块
	修改路径对应的速度和加减速值使速度值在预定的范围内。	SMC_ExtendedVelocityChecks	功能块
	插补	SMC_Interpolator	功能块
	插补（支持反向插补）	SMC_Interpolator2Dir	功能块
	支持低优先级的反向插补	SMC_Interpolator2Dir_SlowTask	功能块
	动态调整速度和加减速以使其值范围不超过最大值	SMC_LimitDynamics	功能块
	限制圆弧速度	SMC_LimitCircularVelocity	功能块
	G 代码解析	SMC_NCDecoder	功能块
	曲线分解器	SMC_ObjectSplitter	功能块
	重新计算 A、B、C 轴斜率	SMC_RecomputeABCSlopes	功能块
	路径圆滑处理	SMC_RoundPath	功能块
	路径段分析器	SMC_SegmentAnalyzer	功能块
	附加轴路径圆滑处理	SMC_SmoothAddAxes	功能块
	路径圆滑处理	SMC_SmoothPath	功能块
	路径偏移处理	SMC_ToolCorr	功能块
	支持 CAM 混合的插补	SMC_XInterpolator	功能块
	获取 M 功能的参数	SMC_GetMParameters	功能块
	插补到达 M 功能时提前读取参数	SMC_PreAcknowledgeMFunction	功能块

指令类型	指令说明	指令名	指令分类
BlockSearch 指令	缩短路径	SMC_BlockSearch	功能块
	保存当前路径位置	SMC_BlockSearchSavePos	功能块

6 CNC 变换指令

指令类型	指令说明	指令名	指令分类
辅助指令	通过 3D 向量坐标计算方向	SMC_CalcDirectionFromVector	功能块
龙门系统指令	通过轴位置计算工具中心点	SMC_TRAFOF_5Axes	功能块
	二维坐标转换为 G 代码坐标	SMC_TRAFOF_Gantry2	功能块
	线性工具方式的二维坐标转换为 G 代码坐标	SMC_TRAFOF_Gantry2Tool1	功能块
	矩形工具方式的二维坐标转换为 G 代码坐标	SMC_TRAFOF_Gantry2Tool2	功能块
	三维坐标转换为 G 代码坐标	SMC_TRAFOF_Gantry3	功能块
	带有旋转轴的二维坐标转换为 G 代码坐标	SMC_TRAFOF_GantryCutter2	功能块
	带有旋转轴的三维坐标转换为 G 代码坐标	SMC_TRAFOF_GantryCutter3	功能块
	T 型二维坐标转换为 G 代码坐标	SMC_TRAFOF_GantryT2	功能块
	H 型二维坐标转换为 G 代码坐标	SMC_TRAFOF_GantryH2	功能块
	G 代码坐标转换为二维反向坐标	SMC_TRAFOV_Gantry2	功能块
	G 代码坐标转换为三维反向坐标	SMC_TRAFOV_Gantry3	功能块
	G 代码坐标转换为带旋转轴二维反向坐标	SMC_TRAFOV_GantryCutter2	功能块
	G 代码坐标转换为带旋转轴三维反向坐标	SMC_TRAFOV_GantryCutter3	功能块
	G 代码坐标转换为 H 型二维反向坐标	SMC_TRAFOV_GantryH2	功能块
	计算工具中心点对于的轴位置	SMC_TRAFO_5Axes	功能块
	G 代码坐标转换为二维坐标	SMC_TRAFO_Gantry2	功能块
	线性工具方式的 G 代码坐标转换为二维坐标	SMC_TRAFO_Gantry2Tool1	功能块
	矩形工具方式的 G 代码坐标转换为二维坐标	SMC_TRAFO_Gantry2Tool2	功能块
	G 代码坐标转换为三维坐标	SMC_TRAFO_Gantry3	功能块
	带旋转轴的 G 代码坐标转换为二维坐标	SMC_TRAFO_GantryCutter2	功能块
	带旋转轴的 G 代码坐标转换为三维坐标	SMC_TRAFO_GantryCutter3	功能块
	G 代码坐标转换为 H 型二维坐标	SMC_TRAFO_GantryH2	功能块
	G 代码坐标转换为 T 型二维坐标	SMC_TRAFO_GantryT2	功能块
并联机器人系统指令	两脚臂正向转换	SMC_TRAFOF_Bipod_Arm	功能块
	三脚架正向转换	SMC_TRAFOF_Tripod	功能块
	三脚臂正向转换	SMC_TRAFOF_Tripod_Arm	功能块
	二角臂反向转换	SMC_TRAFO_Bipod_Arm	功能块
	三脚架反向转换	SMC_TRAFO_Tripod	功能块
	三角臂反向转换	SMC_TRAFO_Tripod_Arm	功能块
机器人运动学变换指令	有多关节的六自由度机器人世界坐标与六轴位置坐标变换的运动学变换	SMC_TrafoF_ArticulatedRobot_6DOF	功能块
	六自由度机器人的运动学反变换功能	SMC_Trafo_ArticulatedRobot_6DOF	功能块

指令类型	指令说明	指令名	指令分类
关节机器人系统指令	Polar 正向转换	SMC_TRAFOF_Polar	功能块
	两关节 Scara 正向转换	SMC_TRAFOF_Scara2	功能块
	三关节 Scara 正向转换	SMC_TRAFOF_Scara3	功能块
	Polar 反向转换	SMC_TRAFO_Polar	功能块
	两关节 Scara 反向转换	SMC_TRAFO_Scara2	功能块
	三关节 Scara 反向转换	SMC_TRAFO_Scara3	功能块

7 ETC 指令

指令类型	指令说明	指令名	指令分类
EtherCAT 参数读写指令	读取从站索引、子索引对应的值	SMC3_ETC_ReadParameter_CoE	功能块
	写入从站索引、子索引对应的值	SMC3_ETC_WriteParameter_CoE	功能块

8 CmpHCBasic

指令类型	指令说明	指令名	指令分类
单轴控制	增强型点动控制	MC_Jog_HC	功能块
EtherCAT 驱动器复位	增强型复位功能，单个驱动器通讯、轴复位	MC_ResetDrive	功能块
EtherCAT 从站复位	单个从站通讯复位	MC_ResetRemoteModule	功能块
位置保持	带绝对值电机驱动器上下电，位置保持功能	MC_PersistPosition	功能块

9 IoDrvCANopenAxis

指令类型	指令说明	指令名	指令分类
单轴控制	电机使能	MC_Power_CO	功能块
	绝对定位	MC_MoveAbsolute_CO	功能块
	相对定位	MC_MoveRelative_CO	功能块
	速度模式	MC_MoveVelocity_CO	功能块
	回原点	MC_Home_CO	功能块
	停止，不能被其他指令打断	MC_Stop_CO	功能块
	停止，可以被其他指令打断	MC_Halt_CO	功能块
	轴错误复位	MC_Reset_CO	功能块
	点动	MC_Jog_CO	功能块
	读取当前状态机	MC_ReadStatus_CO	功能块
参数读写	读从站对象字典值	MC_ReadParameter_CO	功能块
	写从站对象字典值	MC_WriteParameter_CO	功能块

10 CmpHSIO

指令类型	指令说明	指令名	指令分类
高速计数	计数器使能	HC_Counter	功能块
	设置比较一致输出	HC_SetCompare	功能块
	预置值写入	HC_PresetValue	功能块
	打开外部中断和比较一致中断	HC_EnableInterrupt	功能块
	读取锁存值	HC_TouchProbe	功能块
	脉冲宽度测量值读取	HC_MeasurePulseWidth	功能块
	计数器采样，统计一段时间中的计数个数	HC_Sample	功能块
	参数读取	HC_ReadBoolParameter	功能块
	参数写入	HC_WriteBoolParameter	功能块
	比较一致输出：可以设置多达 100 个比较值，Done 信号是输出，Value 是当前输出索引号	HC_SetCompareM	功能块
	设置环形计数	HC_SetRing	功能块
	复位比较一致输出的端口（后台设置的端口），硬件直接输出端口（HC_SetCompare 的 ImRefresh 和 ImRefreshCycle）无效	HC_ResetCmpOutput	功能块
	参数写入	HC_WriteInterruptParameter	功能块
高速轴	点动	MC_Jog_P	功能块
	回零运动	MC_Home_P	功能块
	绝对运动	MC_MoveAbsolute_P	功能块
	相对运动	MC_MoveRelative_P	功能块
	速度模式运行	MC_MoveVelocity_P	功能块
	停止	MC_Stop_P	功能块
	错误复位	MC_Reset_P	功能块
	轴使能控制	MC_Power_P	功能块
	参数读取	MC_ReadParameter_P	功能块
	参数写入	MC_WriteParameter_P	功能块
	逻辑地址设置	MC_SetPosition_P	功能块
	轴状态读取	MC_ReadStatus_P	功能块

6.2 高速 I/O

硬件特性：16 路输入端口，支持 8 路 AB 相输入，前 8 路可作为中断端口。输入最高 200kHz。8 路输出端口，支持 4 路脉冲 + 方向输出或者 cw/ccw 输出。前两路输出支持原点回归和硬件限位功能，以及速度位置切换功能。

功能概述：本地的高速输入、输出端口，包含计数、采样、测频、比较输出、探针、中断、脉冲轴控等功能。采样频率、输出频率最大可支持 200kHz；高速中断最大可支持 3kHz。

对应库名：CmpHSIO(适用版本 V0.0.0.6 和以上)

6.2.1 高速计数

1 功能块

表 6-1 输入资源分配表

功能	输入资源 X0-Xf
单相	0 1 2 3 4 5 6 7
AB 相	0 1 2 3 4 5 6 7 8 9 a b c d e f
测频率、旋转速度	0 1 2 3
脉宽	8 9 a b
探针	8 9 a b
采样	8 9 a b

■ HC_Counter

计数器使能。

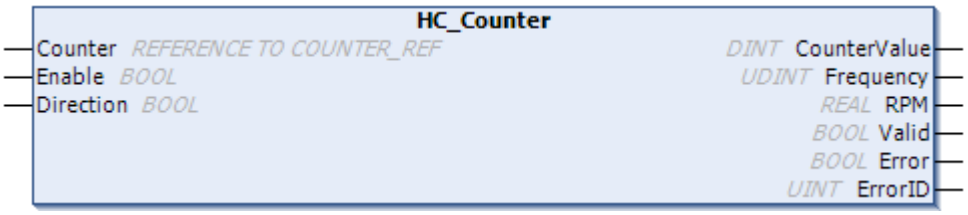


图 6-1 HC_Counter 功能块示例图

表 6-2 HC_Counter 功能块输入输出参数表

参数名称	参数类型	输入输出类型	参数作用
Counter	COUNTER_REF	In	指定使用的计数器号码 [0..7]
Enable	Bool	In	功能块的使能位，1：使能计数器，开始计数，0：停止计数。
Direction	Bool	In	0 加法计数， 1 减法计数 仅对单相输入或内部时钟模式有效； 能流变化才会有效。
CounterValue	Dint	Out	计数器的当前值。
Frequency	UDINT	Out	脉冲频率测量值 ,Hz
RPM	REAL	Out	每分钟旋转速度 (revolution per minute), 单位是 r/min
Valid	Bool	Out	1 有效， 0 无效
Error	Bool	Out	功能块出错标志
ErrorID	Uint	Out	错误码

表 6-3 计数器及计数端口资源分配关系表

计数方式	端口 计数器	X0	X1	X2	X3	X4	X5	X6	X7	X8	X9	Xa	Xb	Xc	Xd	Xe	Xf
AB AB 相计数	Counter0	A	B							T							
	Counter1			A	B						T						
	Counter2 (△)					A	B					T					
	Counter3 (△)							A	B				T				
	Counter4 (△)									A	B						
	Counter5 (△)											A	B				
	Counter6 (△)													A	B		
	Counter7 (△)															A	B
A 相 计数	Counter0	A															
	Counter1		A														
	Counter2			A													
	Counter3				A												
	Counter4 (△)					A											
	Counter5 (△)						A										
	Counter6 (△)							A									
	Counter7 (△)								A								

1） （△）：代表 FrequencyValue 和 RotationRateValue 使用软件计算。

2） Counter0/ Counter1 选择 AB 相计数时，只显示 A 相频率。

3） T 为外部触发信号。

4） 前 4 路计数器和后 4 路有点区别：

正向计数：当计数值达到 2147483647 时，前四路停止计数，而后四路则继续从 -2147483648 开始计数。

负向计数：当计数值达到 -2147483648 时，前四路停止计数，而后四路则继续从 2147483647 开始计数。

5） 线性计数和环形计数的区别：

线性计数：（DownLimitValue，UpLimitValue）

环形计数：[iRingDownValue, iRingUpValue]

计数器 0-3：线性计数范围是 (-2147483648, 2147483647)，不包含 -2147483648 和 2147483647。

环形计数最大范围是 [-2147483648, 2147483647]，包含 -2147483648 和 2147483647。

计数器 4-7：线性计数范围是 (-2147483648, 2147483647)，不包含 -2147483648 和 2147483647。

6） 计数器 4-7：如果计数值到边界值附近时，因为扫描周期影响，计数器溢出可能不报错。

7） 程序下载时，计数器值不清零。

8） HC_Counter 及 HC_MeasurePulseWidth 功能块不能同时使用，推荐程序中只选择一个功能块使用。

【注】：HC_Sample 功能块在 HC_Counter 使能后才能运行。

■ HC_SetCompare

设置比较一致输出，需要调用 HC_EnableInterrupt 打开中断。

图 6-2 HC_SetCompare 功能块示例图

表 6-4 HC_SetCompare 功能块输入输出参数表

参数名称	参数类型	输入输出类型	参数作用
Counter	COUNTER_REF	In	指定使用的计数器号码 [0..7]
Execute	Bool	In	功能块的使能位，上升沿触发
CompareValue	Dint	In	设置的计数器的比较值。 (-2147483648, 2147483647)，不包含 -2147483648 和 2147483647
ImRefresh	Bool		0：后台配置输出端口，是软件输出。 1：硬件直接输出： 计数器 0 输出 Y0， 计数器 1 输出 Y1， ……， 计数器 7 输出 Y7
ImRefreshCycle	Uint		输出时间 [0-30000]，单位是 100us，最大输出时间是 3000ms。比如：设置 10000，就是指 1000ms
Done	Bool	Out	功能块初始化成功标志
Error	Bool	Out	功能块出错标志
ErrorID	Uint	Out	错误码

【注】：对 Y0-Y7 的控制，分为 ImRefresh=0 和 ImRefresh=1 配置两种方式。

1） ImRefresh=0：后台配置，输出软控制，每个计数器可以自由选择 Y0-Y7, 输出存在一定延时；

输出时间为 100us-3000ms。

如果 ImRefreshCycle 为 0，则调用 HC_ResetCmpOutput 拉低输出。

2） ImRefresh=1：硬件立即输出，Y0-Y7 不能自由选择，不存在延时，输出时间通过 ImRefreshCycle 设定。

输出时间为 0-3000ms。

3） 需要提前调用 HC_EnableInterrupt 打开比较一致中断。

4） 热复位和冷复位后，保持上一次的比较值。

■ HC_PresetValue

预置值写入。

图 6-3 HC_PresetValue 功能块示例图

表 6-5 HC_PresetValue 功能块输入输出参数表

参数名称	参数类型	输入输出类型	参数作用
Counter	COUNTER_REF	In	指定使用的计数器号码 [0..7]， 外部触发写入预置值时，仅支持 [0..3]
TriggerType	BYTE	In	0：上升沿触发写入。 1: 外部输入触发 外部信号： couter0->x8 couter1->x9 couter2->xa couter3->xb 2: 比较一致输出时预置，上升沿触发。
Execute	Bool	In	功能块的使能位，上升沿触发
PresetValue	Dint	In	设置的计数器的预置值。 (-2147483648, 2147483647)，不包含 -2147483648 和 2147483647
Done	Bool	Out	功能块执行完成
Error	Bool	Out	功能块出错标志
ErrorID	Uint	Out	错误码

■ HC_EnableInterrupt

打开外部中断和比较一致中断。

图 6-4 HC_EnableInterrupt 功能块示例图

表 6-6 HC_EnableInterrupt 功能块输入输出参数表

参数名称	参数类型	输入输出类型	参数作用
Enable	Bool	In	使能相应中断 1：使能，0：无效
External	Uint	In	打开外部输入中断，比如 3，二进制为 2#11，则端口 0 和 1 位被打开
Compare	Uint	In	打开比较一致中断，比如 3，二进制为 2#11，则端口 0 和 1 位被打开
Valid	Bool	Out	中断生效
Error	Bool	Out	功能块出错标志
ErrorID	Uint	Out	错误码

【注】：

1） HC_WriteInterruptParameter 早于 HC_EnableInterrupt 执行，则 HC_WriteInterruptParameter 参数有效。

2） HC_EnableInterrupt 早于 HC_WriteInterruptParameter 执行，后台中断参数有效；

HC_WriteInterruptParameter 再执行，则 HC_WriteInterruptParameter 参数有效。

■ HC_TouchProbe

当外部中断产生时，读取计数器 [0..3] 的当前值并锁存。

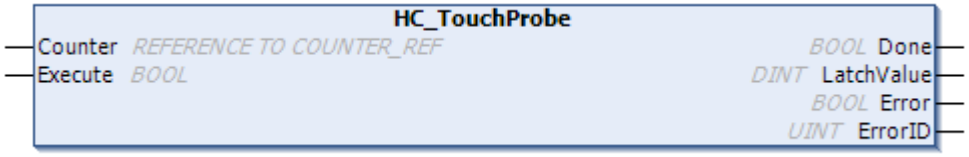


图 6-5 HC_TouchProbe 功能块示例图

表 6-7 HC_TouchProbe 功能块输入输出参数表

参数名称	参数类型	输入输出类型	参数作用
Counter	COUNTER_REF	In	指定使用的计数器号码 [0..3]
Execute	Bool	In	上升沿触发，一次触发只能产生一次中断，下次中断需要再次触发。
Done	Bool	Out	0：未完成 1：完成
Busy	Bool	Out	执行过程中
Value	Dint	Out	锁存值
Error	Bool	Out	功能块出错标志
ErrorID	Uint	Out	错误码

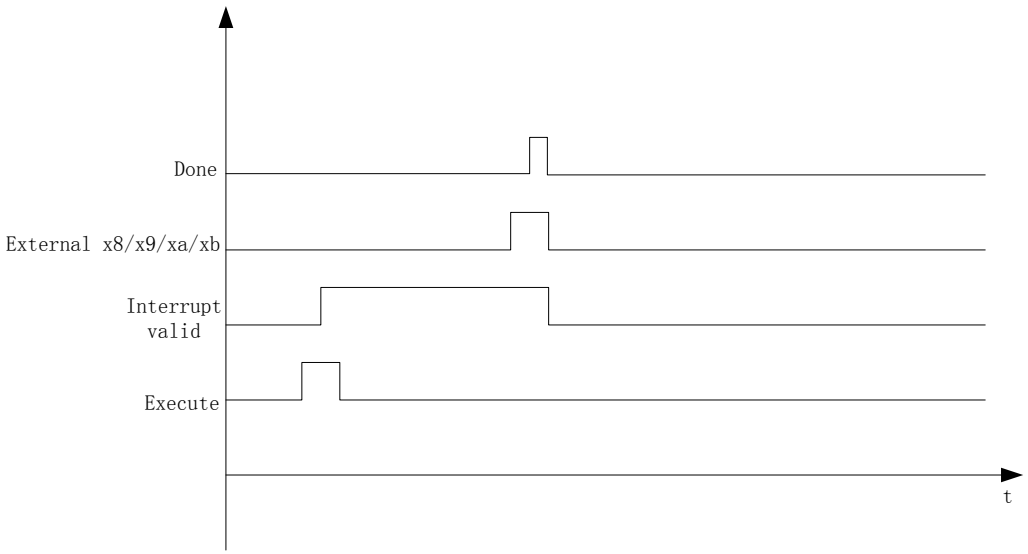


图 6-6 HC_TouchProbe 功能块时序图

■ HC_MeasurePulseWidth

外部触发有效时，读取 x8/x9/xa/xb 的脉冲宽度测量值。

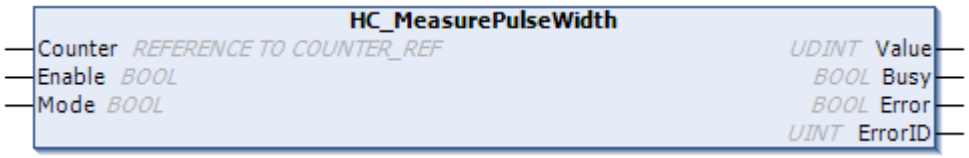


图 6-7 HC_MeasurePulseWidth 功能块示例图

表 6-8 HC_MeasurePulseWidth 功能块输入输出参数表

参数名称	参数类型	输入输出类型	参数作用
Counter	COUNTER_REF	In	指定使用的计数器号码 [0..3]
Enable	Bool	In	功能块的使能位，电平触发
Value	Udint	Out	读取的脉宽测量值，单位 us，范围是 [0,4000000]
Mode	Bool	In	0：外部信号高电平 (测量高电平脉宽)，1：外部信号低电平（测量低电平脉宽）
Busy	Bool	Out	1 正在采样 0 没有采样
Error	Bool	Out	功能块出错标志
ErrorID	Uint	Out	错误码

【注】：HC_Counter 及 HC_MeasurePulseWidth 功能块不能同时使用，推荐程序中只选择一个功能块使用。

■ HC_Sample

计数器采样，统计一段时间中的计数个数。

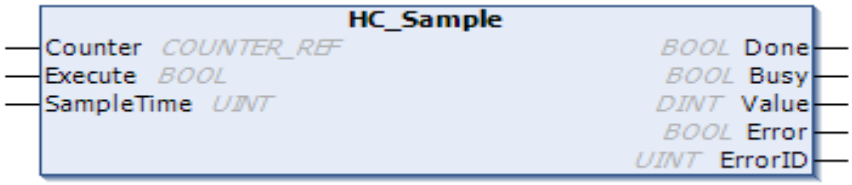


图 6-8 HC_Sample 功能块示例图

表 6-9 HC_Sample 功能块输入输出参数表

参数名称	参数类型	输入输出类型	参数作用
Counter	COUNTER_REF	In	指定使用的计数器号码 [0..3]
Execute	Bool	In	功能块的使能位，上升沿触发。 等待外部输入 (x8/x9/xa/xb) 触发
SampleTime	Uint	In	10~65535(10ms~65535ms)
Value	Dint	Out	采样值
Done	Bool	Out	1: 采样完成，0：采样未完成
Busy	Bool	Out	1：正在采样，0：停止采样 只对计数器 0-3 有效
Error	Bool	Out	功能块出错标志
ErrorID	Uint	Out	错误码

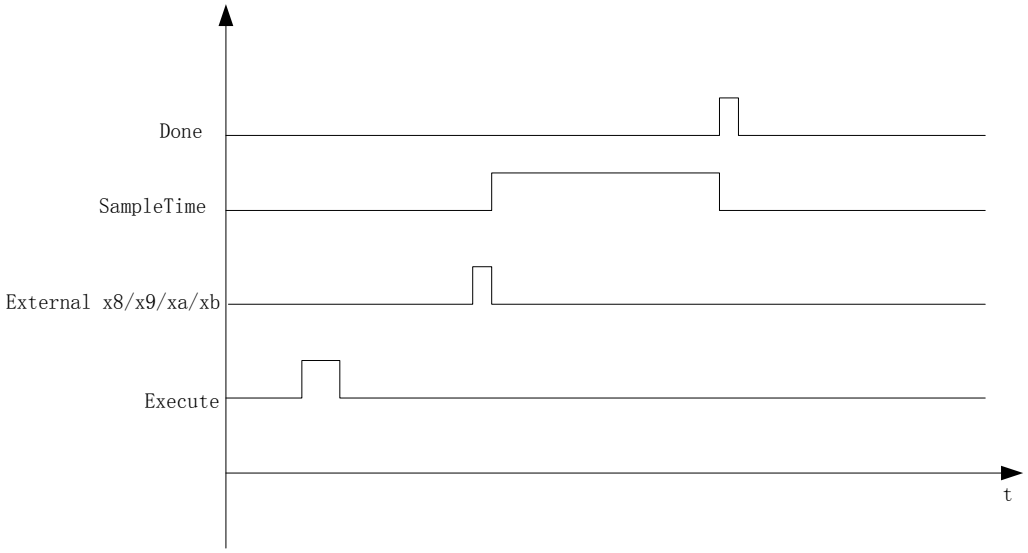


图 6-9 HC_Sample 功能块时序图

【注】：HC_Sample 在 HC_Counter 使能后才能运行。

HC_ReadBoolParameter

参数读取。

图 6-10 HC_ReadBoolParameter 功能块示例图

表 6-10 HC_ReadBoolParameter 功能块输入输出参数表

参数名称	参数类型	输入输出类型	参数作用
Counter	COUNTER_REF	In	指定使用的计数器号码 [0..7]
Enable	Bool	In	功能块的使能位
ParameterNumber	Dint	In	参数号
Valid	Bool	Out	1：有效，0：无效
Value	Bool	Out	参数值
Error	Bool	Out	功能块出错标志
ErrorID	Uint	Out	错误码

表 6-11 PN 参数映射表

PN	Name	DataByte	R/W	Comments
10001	bDisableCmpEvent	Bool	R/W	置用户任务外部比较事件：0 有效，1 比较事件无效。
10002	bDisableOutput	Bool	R/W	比较一致对应的输出口是否输出：0：不输出 1：输出。对 ImRefresh 无效。
10008	Direction	Bool	R/W	计数器方向
10009	bDisableFrequency	Bool	R/W	HC_Counter 中使 FrequencyValue 无效
10010	bDisableRotationRate	Bool	R/W	HC_Counter 中使 RotationRateValue 无效

■ HC_WriteBoolParameter

参数写入。

图 6-11 HC_WriteBoolParameter 功能块示例图

表 6-12 HC_WriteBoolParameter 功能块输入输出参数表

参数名称	参数类型	输入输出类型	参数作用
Counter	COUNTER_REF	In	指定使用的计数器号码 [0..7]
Execute	Bool	In	功能块的使能位，上升沿触发
ParameterNumber	DINT	In	参数号
Value	BOOL	In	参数值
Done	Bool	Out	1：执行完成，0：执行未完成
Error	Bool	Out	功能块出错标志
ErrorID	Uint	Out	错误码

表 6-13 PN 参数映射表

PN	Name	DataByte	R/W	Comments
10001	bDisableCmpEvent	Bool	R/W	置用户任务外部比较事件：1 比较事件无效，0 有效。
10002	bDisableOutput	Bool	R/W	比较一致对应的输出口是否输出：0：输出 1：不输出。对 ImRefresh=true 时无效。
10008	Direction	Bool	R/W	计数器方向，会立即生效。
10009	bDisableFrequency	Bool	R/W	HC_Counter 中使 FrequencyValue 无效
10010	bDisableRotationRate	Bool	R/W	HC_Counter 中使 RotationRateValue 无效

■ HC_SetCompareM

比较一致输出：可以设置多达 100 个比较值，Done 信号是输出。

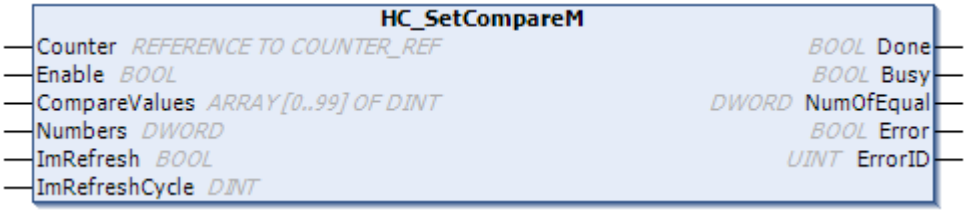


图 6-12 HC_SetCompareM 功能块示例图

表 6-14 HC_SetCompareM 功能块输入输出参数表

参数名称	参数类型	输入输出类型	参数作用
Counter	COUNTER_REF	In	指定使用的计数器号码 [0..7]
Enable	Bool	In	功能块的使能位，上升沿触发
CompareValues	ARRAY [0..99] OF DINT	In	设置的计数器的比较值一维数组，最大支持 100 个。每个值范围是 (-2147483648, 2147483647)，不包含 -2147483648 和 2147483647

参数名称	参数类型	输入输出类型	参数作用
Numbers	Dword	In	CompareValues 一维数组的实际个数，最大值是 100
ImRefresh	Bool	In	硬件直接输出
ImRefreshCycle	Uint	In	硬件直接输出时间，单位是 100us，最大输出时间是 3000ms。比如：设置 10000，就是指 1000ms
Done	Bool	Out	所有执行完成标志
NumOfEqual	Dword	Out	相等个数
Error	Bool	Out	功能块出错标志
ErrorID	Uint	Out	错误码

【注】：

1） 当执行 Enable=TRUE，计数方向不允许改变。

2） 数组元素必须是正向逐渐增大或负向逐渐减小，中间不能出现相等。

3） 正向：计数器起始值必须小于第一个比较值，否则，第一个比较值将没时间执行。

反向：计数器起始值必须大于第一个比较值，否则，第一个比较值将没时间执行。

4） 需要提前调用 HC_EnableInterrupt 打开比较一致中断。

5） 正向运行时，按照如下格式定义：

arr1 :ARRAY [0..99] OF DINT := [2000,4000,6000,8000,10000,12000,14000,16000,18000,20000];

反向运行时，按照如下格式定义：

arr2 :ARRAY [0..99] OF DINT := [20000,18000,16000,14000,12000,10000,8000,6000,5000,3000];

6） 本功能块依赖于 HC_Counter。

7） 对 Y0-Y7 的控制，分为 ImRefresh=0 和 ImRefresh=1 配置两种方式：

■ ImRefresh=0：后台配置，输出软控制，每个计数器可以自由选择 Y0-Y7, 输出存在一定延时；

输出时间为 100us-3000ms。

如果 ImRefreshCycle 为 0，则调用 HC_ResetCmpOutput 拉低输出。

■ ImRefresh=1：硬件立即输出，Y0-Y7 不能自由选择，不存在延时，输出时间通过 ImRefreshCycle 设定。

输出时间为 0-3000ms。

■ 需要提前调用 HC_EnableInterrupt 打开比较一致中断。

8） 热复位和冷复位后，保持上一次的比较值。

■ HC_SetRing

设置环形计数。

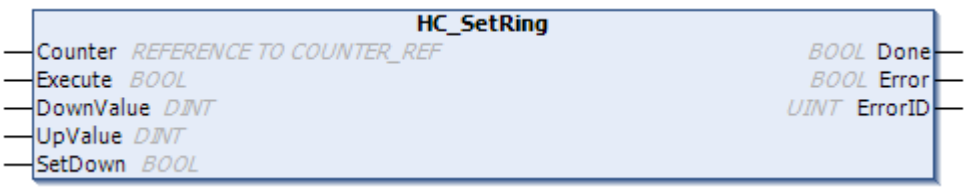


图 6-13 HC_SetRing 功能块示例图

表 6-15 HC_SetRing 功能块输入输出参数表

参数名称	参数类型	输入输出类型	参数作用
Counter	COUNTER_REF	In	指定使用的计数器号码 [0..3]
Execute	Bool	In	功能块的使能位，上升沿触发
DownValue	Dint	In	下限值，范围是 (-2147483648, 2147483647)，不包含 -2147483648 和 2147483647
UpValue	Dint	In	上限值，范围是 (-2147483648, 2147483647)，不包含 -2147483648 和 2147483647
SetDown	Bool	In	强制从下限值开始计数
Done	Bool	Out	设置成功
Error	Bool	Out	功能块出错标志
ErrorID	Uint	Out	错误码

■ HC_ResetCmpOutput

复位比较一致输出的端口（后台设置的端口），硬件直接输出端口（HC_SetCompare 的 ImRefresh 和 ImRefreshCycle）无效。

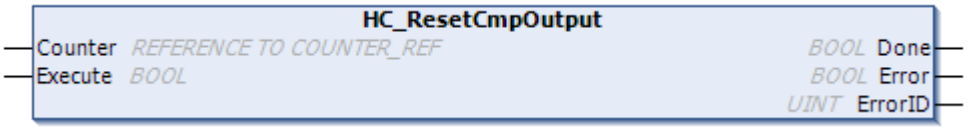


图 6-14 HC_ResetCmpOutput 功能块示例图

表 6-16 HC_ResetCmpOutput 功能块输入输出参数表

参数名称	参数类型	输入输出类型	参数作用
Counter	COUNTER_REF	In	指定使用的计数器号码 [0..7]
Execute	Bool	In	功能块的使能位，上升沿触发
Done	Bool	Out	设置成功标志
Error	Bool	Out	功能块出错标志
ErrorID	Uint	Out	错误码

■ HC_WriteInterruptParameter

中断参数写入。

图 6-15 HC_WriteInterruptParameter 功能块示例图

表 6-17 HC_WriteInterruptParameter 功能块输入输出参数表

参数名称	参数类型	输入输出类型	参数作用
Execute	Bool	In	功能块的使能位，上升沿触发
ParameterNumber	DINT	In	参数号
Value	BOOL	In	参数值

参数名称	参数类型	输入输出类型	参数作用
Done	Bool	Out	1：执行完成，0：执行未完成
Error	Bool	Out	功能块出错标志
ErrorID	Uint	Out	错误码

表 6-18 PN 参数映射表

PN	Name	DataByte	R/W	Comments
10080	X0InterruptEvent	Byte	W	0: 上升沿，1 为下降沿，2 为上升沿 + 下降沿
10081	X1InterruptEvent	Byte	W	0: 上升沿，1 为下降沿，2 为上升沿 + 下降沿
10082	X2InterruptEvent	Byte	W	0: 上升沿，1 为下降沿，2 为上升沿 + 下降沿
10083	X3InterruptEvent	Byte	W	0: 上升沿，1 为下降沿，2 为上升沿 + 下降沿
10084	X4InterruptEvent	Byte	W	0: 上升沿，1 为下降沿，2 为上升沿 + 下降沿
10085	X5InterruptEvent	Byte	W	0: 上升沿，1 为下降沿，2 为上升沿 + 下降沿
10086	X6InterruptEvent	Byte	W	0: 上升沿，1 为下降沿，2 为上升沿 + 下降沿
10087	X7InterruptEvent	Byte	W	0: 上升沿，1 为下降沿，2 为上升沿 + 下降沿
10088	X8InterruptEvent	Byte	W	Counter0 HC_TouchProbe 中断：0: 上升沿，1 为下降沿
10089	X9InterruptEvent	Byte	W	Counter1 HC_TouchProbe 中断：0: 上升沿，1 为下降沿
10090	XaInterruptEvent	Byte	W	Counter2 HC_TouchProbe 中断：0: 上升沿，1 为下降沿
10091	XbInterruptEvent	Byte	W	Counter3 HC_TouchProbe 中断：0: 上升沿，1 为下降沿

■ HC_Reset

清除错误。

图 6-16 HC_Reset 功能块示例图

表 6-19 HC_Reset 功能块输入输出参数表

参数名称	参数类型	输入输出类型	参数作用
Counter	COUNTER_REF	In	指定使用的计数器号码 [0..7]
Execute	Bool	In	功能块的使能位，上升沿触发
Done	Bool	Out	1：执行完成，0：执行未完成
Error	Bool	Out	功能块出错标志
ErrorID	Uint	Out	错误码

2 ErrorID

基本格式如下：

库 + 功能块 + 错误码

3 3 位

库：高速 I/O 默认是 0

功能块号码：从 01 开始，功能块列表详见 6.2.4. 功能块列表

错误码：从 01 开始，错误码详见表“计数器错误码列表”。错误码小于 500 是严重错误，大于 500 是功能块错误。

比如：14506，14 指的是 HC_WriteParameter， 506 指设置参数错误。

表 6-20 计数器错误码列表：

错误码	定义	说明
001	ERR_COUNTERID_INVALID	输入通道号无效，有效范围是 [0,7]
003	ERR_CNT_OVERFLOW	计数器上溢 / 下溢出错
004	ERR_COUNTER_NOT_CHOSEN	高速功能未选中，在后台配置中选择
007	ERR_COUNTER_NOT_ENABLED	计数器 HC_Counter 未使能
101	ERR_WRITEINTERRUPTPARAMETER_UNVALIAD	写中断参数无效
102	ERR_INTERRUPT_NOT_CHOSE	未在后台选择“中断输入”
501	ERR_SETCOMPARE_IMREFRESHCYCLE_OVERFLOW	比较值 ImRefreshCycle 超过 30000，有效范围是 [0,30000]
502	ERR_SETCOMPAREM_NUMBERS_OVERFLOW	HC_SetCompareM 的 Number 范围是 [1,100]
503	ERR_PREWR_VALUE_OVERFLOW	预置值超限
504	ERR_AVERAGE_PARA_UNVALIAD	频率和旋转速度设置平均参数无效
505	ERR_ROTATION_PULSES_UNIT_UNVALIAD	转速测量的每转脉冲数设置无效
506	ERR_WRITEBOOlPARAMETER_UNVALIAD	设置参数无效，HC_WriteBoolParameter
507	ERR_READBOOLPARAMETER_UNVALIAD	获取参数无效 HC_ReadBoolParameter
508	ERR_MEASURE_WIDTH_OVERFLOW	测量宽度无效
509	ERR_SETCOMPAREM_IMREFRESHCYCLE_OVERFLOW	比较值 ImRefreshCycle 超过 30000，有效范围是 [0,30000]
510	ERR_PRESET_TRIGGERTYPE_OVERFLOW	预置参数无效
511	ERR_WRITEPARAMETER_UNVALIAD	设置参数无效，HC_WriteParameter
513	ERR_FUNC_COUNTERID_INVALID	特殊功能通道号无效 , 有效范围是 [0,3]
514	ERR_COUNTER_NOT_CHOSE_EXETERNAL_X	计数器后台没有选择“外部触发输入”
515	ERR_CNT_FORMAT_NOT_RING	环形计数类型不对，在后台配置中选择
516	ERR_RING_DOWNVAL_BEYOND_UPVAL	环形计数下限值≥上限值
517	ERR_SAMPLE_VALUE_LESS	采样时间太小，采样时间范围是 10~65535(10ms~65535ms)
518	ERR_RING_VALUE_OVERFLOW	环形计数超限

6.2.2 高速轴

高速轴输出最大支持 200kHz。

1 功能块

■ MC_Jog_P

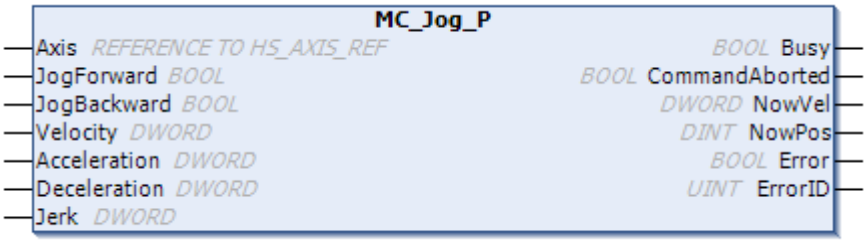


图 6-17 MC_Jog_P 功能块示例图

表 6-21 MC_Jog_P 功能块输入输出参数表

参数名称	参数类型	输入输出类型	参数作用
Axis	HS_AXIS_REF	In	指令轴号码 [0..3]
JogForward	Bool	In	正向运动，电平触发
JogBackward	Bool	In	负向运动，电平触发
Velocity	Dword	In	JOG 运动的速度，pulse/s, 最大 200K
Acceleration	Dword	In	加速度，pulse/s^2, 默认最大值为 20000000，可以通过 dwMaxAcceleration 修改。 参考备注
Deceleration	Dword	In	减速度 , pulse/s^2，默认最大值为 20000000，可以通过 dwMaxDeceleration 修改。 参考备注
Jerk	Dword	In	加加速度 , pulse/s^3
Busy	Bool	Out	JOG 运动标志，运动中时，Busy 为 TRUE。停止运动，Busy 为 FALSE
CommandAborted	Bool	Out	被其它命令打断
Error	Bool	Out	轴出错标志
ErrorID	Uint	Out	错误码

1） Acceleration 和 Deceleration 的取值是 125* 倍率的整数倍。

$$Rate = \begin{cases} 1 & Vmax \leqq 8000 \\ \frac{Vmax}{8000} & Vmax > 8000 \end{cases}$$

$$Vel = \left[\frac{Vel}{Rate}\right] * Rate$$

$$实际加速度 = \begin{cases} 125 * Rate & Acc \leqq 125 * Rate \\ \left[\frac{Acc}{125*Rate}\right] * 125 * Rate & Acc > 125 * Rate \end{cases}$$

[ ]为四舍五入取整数

2） 加减速过程中不允许改变速度。

3） S 型曲线，Jerk 是内部计算，所有涉及 Jerk 的参数全部都是内部计算。

4） S 型曲线，不支持运动过程中改变速度、加速度和减速度。

5） S 型曲线，减速度≥ 5*Vel 时，不会存在拖尾和截断现象。

■ MC_Home_P

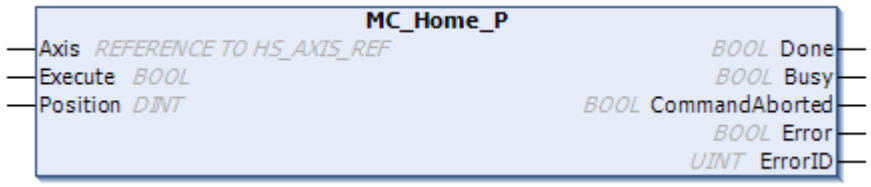


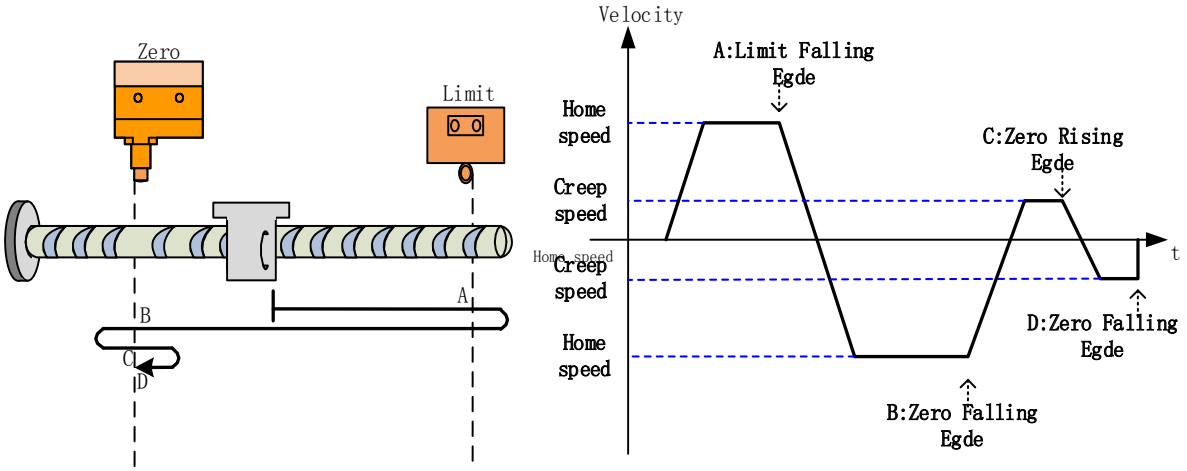
图 6-18 MC_Home_P 功能块示例图

表 6-22 MC_Home_P 功能块输入输出参数表

参数名称	参数类型	输入输出类型	参数作用
Axis	HS_AXIS_REF	In	指令轴号码 [0..3]
Execute	Bool	In	功能块的使能位，上升沿触发
Position	Dint	In	原点回归位置
Done	Bool	Out	原点回归完成标志
Busy	Bool	Out	运动中标志，运动中时，Busy 为 TRUE。停止运动，Busy 为 FALSE
CommandAborted	Bool	Out	被其它命令打断
Error	Bool	Out	轴出错标志
ErrorID	Uint	Out	错误码

mode0

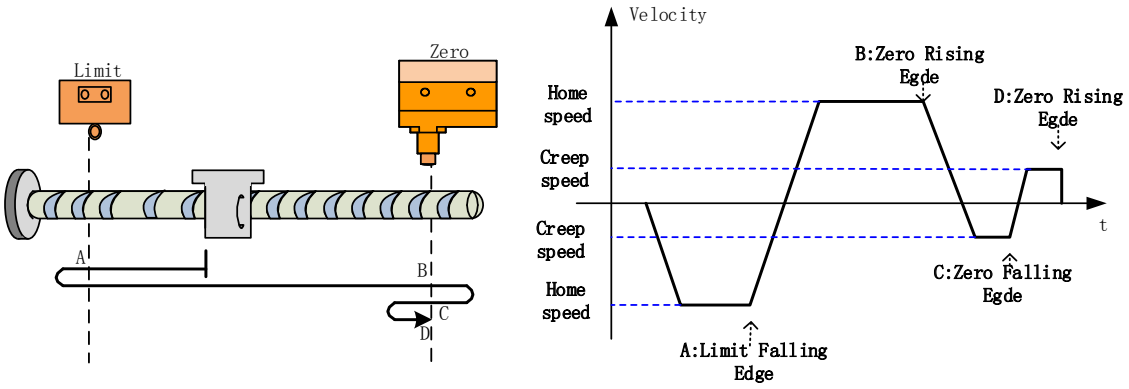
正向回零，减速点、原点为原点开关；并需要设置回零右限位。



【注】：回原启动时当前位置处于右极限位置右侧无效。

mode1

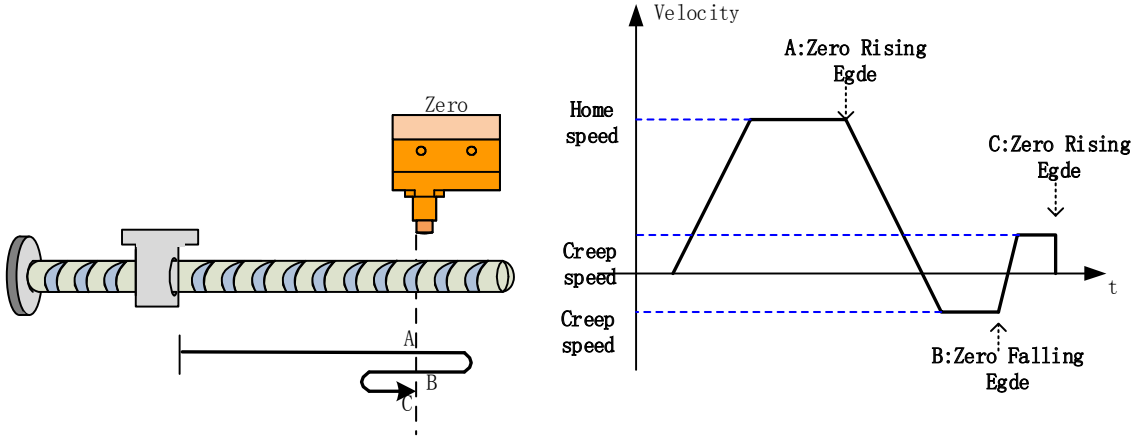
反向回零，减速点、原点为原点开关；并需要设置回零左限位。



【注】：回原启动时当前位置处于左极限位置左侧无效。

mode2

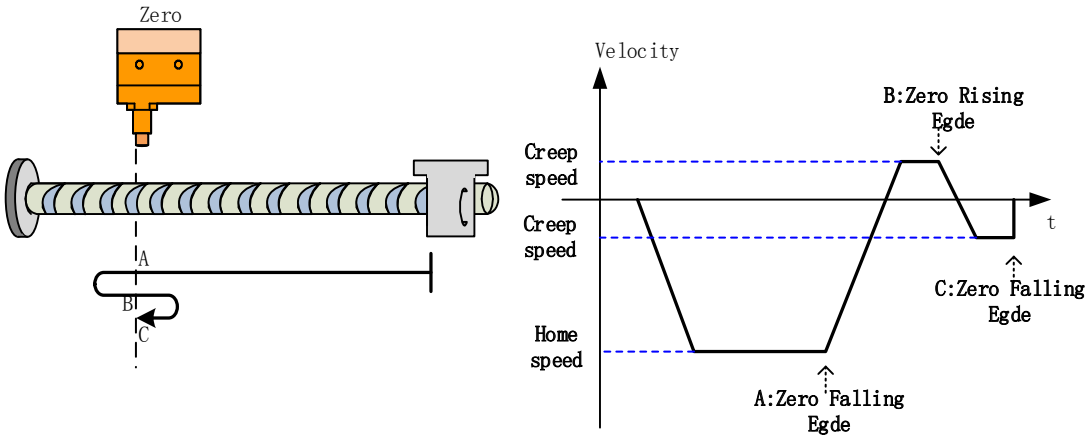
正向回零，减速点、原点为原点开关



【注】：回原启动时当前位置处于零点位置中间或右侧无效。

mode3

反向回零，减速点、原点为原点开关；



【注】：回原启动时当前位置处于零点位置中间或左侧无效。

■ MC_MoveAbsolute_P

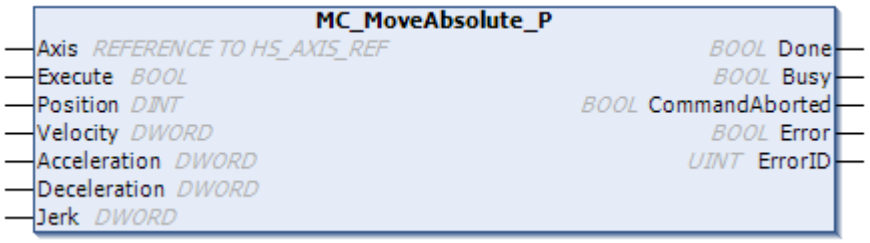


图 6-19 MC_MoveAbsolute_P 功能块示例图

表 6-23 MC_MoveAbsolute_P 功能块输入输出参数表

参数名称	参数类型	输入输出类型	参数作用
Axis	HS_AXIS_REF	In	指令轴号码 [0..3]
Execute	Bool	In	功能块的使能位，上升沿触发
Position	DINT	In	位置
Velocity	Dword	In	速度，最大 200K ,pulse/s。
Acceleration	Dword	In	加速度，pulse/s^2, 默认最大值为 20000000，可以通过 dwMaxAcceleration 修改。 参考备注
Deceleration	Dword	In	减速度 , pulse/s^2，默认最大值为 20000000，可以通过 dwMaxDeceleration 修改。 参考备注
Jerk	Dword	In	加加速度 , pulse/s^3
Done	Bool	Out	完成标志
Busy	Bool	Out	运动中标志，运动中时，Busy 为 TRUE。停止运动，Busy 为 FALSE
CommandAborted	Bool	Out	被其它命令打断
Error	Bool	Out	轴出错标志
ErrorID	Uint	Out	错误码

【备注】：

1） Acceleration 和 Deceleration 的取值是 125* 倍率的整数倍。

$$Rate = \begin{cases} 1 & Vmax \leqq 8000 \\ \frac{Vmax}{8000} & Vmax > 8000 \end{cases}$$

$$Vel = \left[\frac{Vel}{Rate}\right] * Rate$$

$$实际加速度 = \begin{cases} 125 * Rate & Acc \leqq 125 * Rate \\ \left[\frac{Acc}{125*Rate}\right] * 125 * Rate & Acc > 125 * Rate \end{cases}$$

[ ]为四舍五入取整数

2） 加减速过程中不允许改变速度。

3） 定位距离不要超过 2147483647，当定位距离超过 2147483647 时，会反向运行。

4） S 型曲线，Jerk 是内部计算，所有涉及 Jerk 的参数全部都是内部计算。

5） S 型曲线，不支持运动过程中改变速度、加速度和减速度。

6） 6 S 型曲线，减速度≥ 5*Vel 时，不会存在拖尾和截断现象。

■ MC_MoveRelative_P

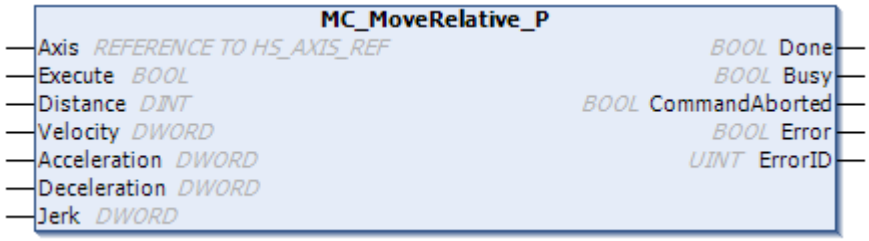


图 6-20 MC_MoveRelative_P 功能块示例图

表 6-24 MC_MoveRelative_P 功能块输入输出参数表

参数名称	参数类型	输入输出类型	参数作用
Axis	HS_AXIS_REF	In	指令轴号码 [0..3]
Execute	Bool	In	功能块的使能位，上升沿触发
Distance	DINT	In	相对距离
Velocity	Dword	In	速度，最大 200K ,pulse/s
Acceleration	Dword	In	加速度，pulse/s^2, 默认最大值为 20000000，可以通过 dwMaxAcceleration 修改 参考备注
Deceleration	Dword	In	减速度 , pulse/s^2，默认最大值为 20000000，可以通过 dwMaxDeceleration 修改 参考备注
Jerk	Dword	In	加加速度 , pulse/s^3
Done	Bool	Out	完成标志
Busy	Bool	Out	运动中标志，运动中时，Busy 为 TRUE。停止运动，Busy 为 FALSE
CommandAborted	Bool	Out	被其它命令打断
Error	Bool	Out	轴出错标志
ErrorID	Uint	Out	错误码

【注】：

1） Acceleration 和 Deceleration 的取值是 125* 倍率的整数倍。

$$\text{Rate} = \begin{cases} 1 & \text{Vmax} \leqq 8000 \\ \frac{\text{Vmax}}{8000} & \text{Vmax} > 8000 \end{cases}$$

$$\text{Vel} = \left[\frac{\text{Vel}}{\text{Rate}}\right] * \text{Rate}$$

$$\text{实际加速度} = \begin{cases} 125 * \text{Rate} & \text{Acc} \leqq 125 * \text{Rate} \\ \left[\frac{\text{Acc}}{125*\text{Rate}}\right] * 125 * \text{Rate} & \text{Acc} > 125 * \text{Rate} \end{cases}$$

[ ]为四舍五入取整数

2） Distance 为负数的话，是从停止位置再运行 Distance。

对于负数，脉冲要先停止，然后再反向运行 Distance。

3） 加减速过程中不允许改变速度。

4） 定位距离不要超过 2147483647，当定位距离超过 2147483647 时，会反向运行。

5） S 型曲线，Jerk 是内部计算，所有涉及 Jerk 的参数全部都是内部计算。

6） S 型曲线，不支持运动过程中改变速度、加速度和减速度。

7） S 型曲线，减速度≥ 5*Vel 时，不会存在拖尾和截断现象。

■ MC_MoveVelocity_P

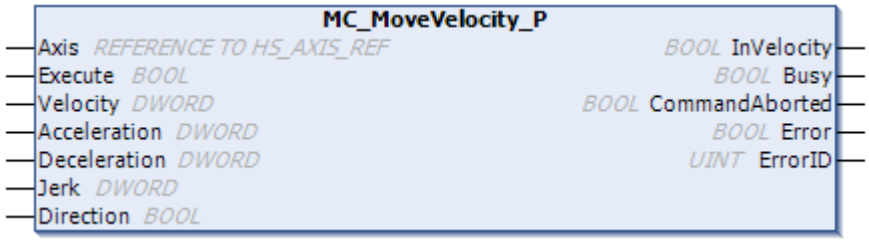


图 6-21 MC_MoveVelocity_P 功能块示例图

表 6-25 MC_MoveVelocity_P 功能块输入输出参数表

参数名称	参数类型	输入输出类型	参数作用
Axis	HS_AXIS_REF	In	指令轴号码 [0..3]
Execute	Bool	In	功能块的使能位，上升沿触发
Velocity	Dword	In	速度，最大 200K ,pulse/s
Acceleration	Dword	In	加速度，pulse/s^2, 默认最大值为 20000000，可以通过 dwMaxAcceleration 修改。 参考备注
Deceleration	Dword	In	减速度 , pulse/s^2，默认最大值为 20000000，可以通过 dwMaxDeceleration 修改。 参考备注
Jerk	Dword	In	加加速度 , pulse/s^3
Direction	Bool	In	方向
InVelocity	Bool	Out	速度到达设置的速度 Velocity
Busy	Bool	Out	运动中标志，运动中时，Busy 为 TRUE。停止运动，Busy 为 FALSE
CommandAborted	Bool	Out	被其它命令打断
Error	Bool	Out	轴出错标志
ErrorID	Uint	Out	错误码

【注】：

1） Acceleration 和 Deceleration 的取值是 125* 倍率的整数倍。

$$Rate = \begin{cases} 1 & Vmax \leqq 8000 \\ \frac{Vmax}{8000} & Vmax > 8000 \end{cases}$$

$$Vel = \left[\frac{Vel}{Rate}\right] * Rate$$

$$实际加速度 = \begin{cases} 125 * Rate & Acc \leqq 125 * Rate \\ \left[\frac{Acc}{125*Rate}\right] * 125 * Rate & Acc > 125 * Rate \end{cases}$$

[ ]为四舍五入取整数

2） 加减速过程中不允许改变速度。

3） 定位距离不要超过 2147483647，当定位距离超过 2147483647 时，会反向运行。

4） S 型曲线，Jerk 是内部计算，所有涉及 Jerk 的参数全部都是内部计算。

5） S 型曲线，不支持运动过程中改变速度、加速度和减速度。

6） S 型曲线，减速度≥ 5*Vel 时，不会存在拖尾和截断现象。

■ MC_Stop_P

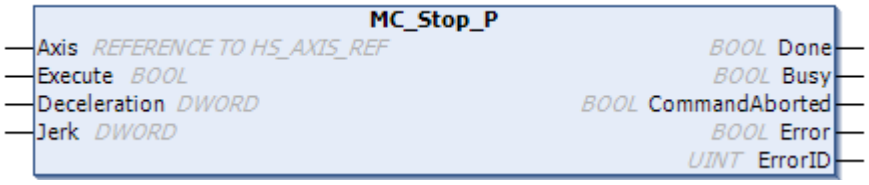


图 6-22 MC_Stop_P 功能块示例图

表 6-26 MC_Stop_P 功能块输入输出参数表

参数名称	参数类型	输入输出类型	参数作用
Axis	HS_AXIS_REF	In	指令轴号码 [0..3]
Execute	Bool	In	功能块的使能位，上升沿触发
Deceleration	DWORD	In	减速度 , pulse/s^2，默认最大值为 20000000，可以通过 dwMaxDeceleration 修改。 【注】： 1 参考备注 2 减速度为 0 时，是立即停止。
Jerk	DWORD	In	加加速度 , pulse/s^3
Done	Bool	Out	执行完成标志
Busy	Bool	Out	执行状态标志
CommandAborted	Bool	Out	被其它命令打断
Error	Bool	Out	轴出错标志
ErrorID	Uint	Out	错误码

【注】：

1） Acceleration 和 Deceleration 的取值是 125* 倍率的整数倍。

$$\text{Rate} = \begin{cases} 1 & \text{Vmax} \leqq 8000 \\ \frac{\text{Vmax}}{8000} & \text{Vmax} > 8000 \end{cases}$$

$$\text{Vel} = \left[\frac{\text{Vel}}{\text{Rate}}\right] * \text{Rate}$$

$$\text{实际加速度} = \begin{cases} 125 * \text{Rate} & \text{Acc} \leqq 125 * \text{Rate} \\ \left[\frac{\text{Acc}}{125*\text{Rate}}\right] * 125 * \text{Rate} & \text{Acc} > 125 * \text{Rate} \end{cases}$$

[ ]为四舍五入取整数

2） S 型曲线，减速度≥ 5*Vel 时，不会存在拖尾和截断现象。

■ MC_Reset_P

图 6-23 MC_Reset_P 功能块示例图

表 6-27 MC_Reset_P 输入输出参数表

参数名称	参数类型	输入输出类型	参数作用
Axis	HS_AXIS_REF	In	指令轴号码 [0..3]
Execute	Bool	In	功能块的使能位，上升沿触发
Done	Bool	Out	执行完成标志
Busy	Bool	Out	执行状态标志
Error	Bool	Out	轴出错标志
ErrorID	Uint	Out	错误码

■ MC_Power_P

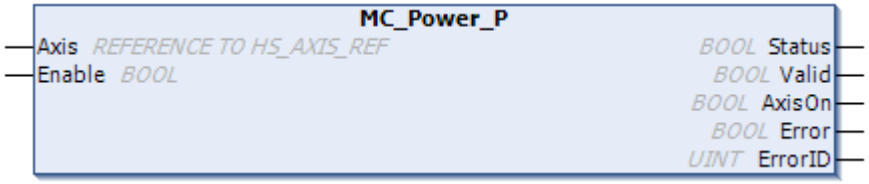


图 6-24 MC_Power_P 功能块示例图

表 6-28 MC_Power_P 功能块输入输出参数表

参数名称	参数类型	输入输出类型	参数作用
Axis	HS_AXIS_REF	In	指令轴号码 [0..3]
Enable	Bool	In	功能块的使能位，电平触发
Status	Bool	Out	执行状态标志
Valid	Bool	Out	执行结果有效标志
AxisOn	Bool	Out	使能轴
Error	Bool	Out	轴出错标志
ErrorID	Uint	Out	错误码

■ MC_ReadParameter_P

图 6-25 MC_ReadParameter_P 功能块示例图

表 6-29 MC_ReadParameter_P 功能块输入输出参数表

参数名称	参数类型	输入输出类型	参数作用
Axis	HS_AXIS_REF	In	指令轴号码 [0..3]
Enable	Bool	In	功能块的使能
ParameterNumber	Dint	In	参数号
Valid	Bool	Out	执行有效标志
Busy	Bool	Out	执行状态标志
Value	Dint	Out	参数返回值
Error	Bool	Out	轴出错标志
ErrorID	Uint	Out	错误码

表 6-30 PN 参数映射表

PN	Name	DataByte	R/W	Comments
10030	diSWLimitPos	dint	R/W	软限位最大值
10031	diSWLimitNeg	dint	R/W	软限位最小值
10032	bEnableLimitPos	Bool	R/W	使能软限位最小值，0 为 FALSE
10033	bEnableLimitNeg	Bool	R/W	使能软限位最大值，0 为 FALSE
10034	Velocity	dint	R	实际速度
10035	Position	dint	R	实际位置
10036	dwMaxAcceleration	dword	R/W	最大加速度
10037	dwMinAcceleration	dword	R/W	最小加速度
10038	dwMaxDeceleration	dword	R/W	最大减速度
10039	dwMinDeceleration	dword	R/W	最小减速度
10040	dwMaxVelocity	Dint	R/W	最大速度
10041	dwHomeVelocity	dword	R/W	回原速度
10042	dwHomeCreepVelocity	dword	R/W	回原蠕动速度
10043	dwHomeAcceleration	dword	R/W	回原加速度
10044	dwHomeDeceleration	dword	R/W	回原减速度
10045	bHomeMode	Byte	R/W	回原模式，[0, 3]
10046	bErrorStopMode	Bool	R/W	轴出错时的停止模式，0 为 FALSE
10046	bSoftLimitStopMode	Bool	R/W	软限位时的停止模式，0 为 FALSE

■ MC_WriteParameter_P

图 6-26 MC_WriteParameter_P 功能块示例图

表 6-31 MC_WriteParameter_P 功能块输入输出参数表

参数名称	参数类型	输入输出类型	参数作用
Axis	HS_AXIS_REF	In	指令轴号码 [0..3]
Execute	Bool	In	功能块的使能位，上升沿触发

参数名称	参数类型	输入输出类型	参数作用
ParameterNumber	Dint	In	参数号
Value	Dint	In	参数值
Done	Bool	Out	执行完成标志
Busy	Bool	Out	执行状态
Error	Bool	Out	轴出错标志
ErrorID	Uint	Out	错误码

表 6-32 PN 参数映射表

PN	Name	DataByte	R/W	Comments
10030	diSWLimitPos	dint	R/W	软限位最大值， 注：限位实际值，到此位置会报错
10031	diSWLimitNeg	dint	R/W	软限位最小值， 注：限位实际值，到此位置会报错
10032	bEnableLimitPos	Bool	R/W	使能软限位最大值，0 为 FALSE
10033	bEnableLimitNeg	Bool	R/W	使能软限位最小值，0 为 FALSE
10036	dwMaxAcceleration	dword	R/W	最大加速度
10037	dwMinAcceleration	dword	R/W	最小加速度
10038	dwMaxDeceleration	dword	R/W	最大减速度
10039	dwMinDeceleration	dword	R/W	最小减速度
10041	dwHomeVelocity	dword	R/W	回原速度
10042	dwHomeCreepVelocity	dword	R/W	回原蠕动速度
10043	dwHomeAcceleration	dword	R/W	回原加速度
10044	dwHomeDeceleration	dword	R/W	回原减速度
10045	bHomeMode	Byte	R/W	回原模式，[0, 3]
10046	bErrorStopMode	Bool	R/W	轴出错时的停止模式，0：减速停止， 1：立即停止
10047	bSoftLimitStopMode	Bool	R/W	软限位时的停止模式，0：减速停止， 1：立即停止

■ MC_SetPosition_P

逻辑地址设置。

图 6-27 MC_SetPosition_P 功能块示例图

表 6-33 MC_SetPosition_P 功能块输入输出参数表

参数名称	参数类型	输入输出类型	参数作用
Axis	HS_AXIS_REF	In	指令轴号码 [0..3]
Execute	Bool	In	功能块的使能位，上升沿触发
Position	Dint	In	地址
Relative	Bool	In	0：逻辑地址的绝对位置， 1：逻辑地址的相对位置。
Done	Bool	Out	执行完成标志

参数名称	参数类型	输入输出类型	参数作用
Busy	Bool	Out	执行状态标志
Error	Bool	Out	轴出错标志
ErrorID	Uint	Out	错误码

【注】：1、程序下载时，当前位置值不清零。

■ MC_ReadStatus_P

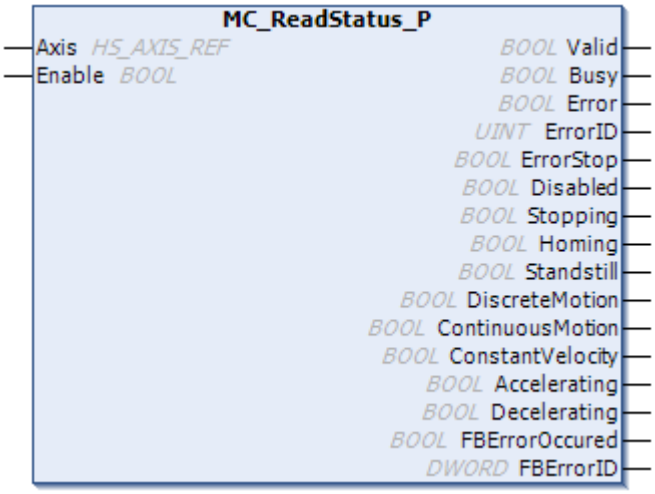
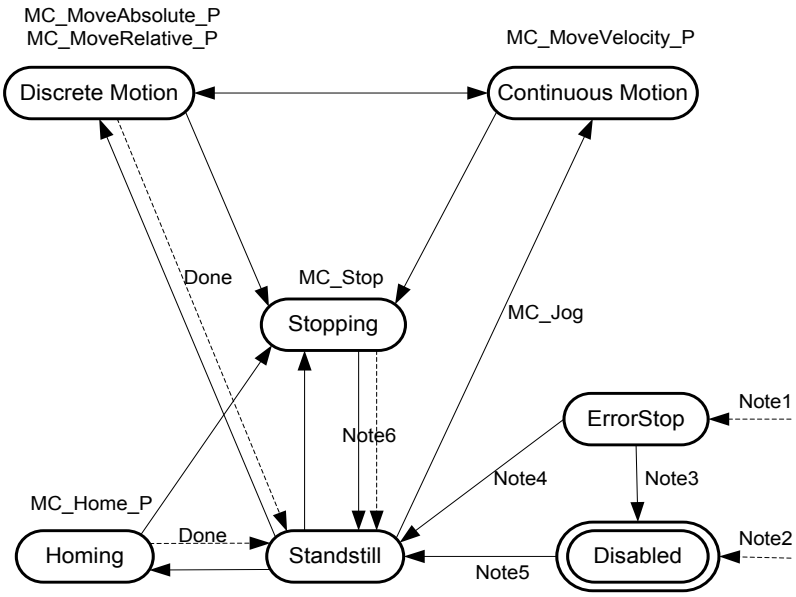


图 6-28 MC_ReadStatus_P 功能块示例图

表 6-34 MC_ReadStatus_P 功能块输入输出参数表

参数名称	参数类型	输入输出类型	参数作用
Axis	HS_AXIS_REF	In	指令轴号码 [0..3]
Enable	Bool	In	功能块的使能位，电平触发
Valid	Bool	Out	执行有效标志
Busy	Bool	Out	执行状态标志
Error	Bool	Out	轴出错标志
ErrorID	Uint	Out	错误码
ErrorStop	Bool	Out	错误停止模式
Disabled	Bool	Out	无效模式
Stopping	Bool	Out	正在停止模式
Homing	Bool	Out	回原模式
Standstill	Bool	Out	静止模式
DiscreteMotion	Bool	Out	定位模式
ContinuousMotion	Bool	Out	速度模式
ConstantVelocity	Bool	Out	匀速阶段
Accelerating	Bool	Out	加速阶段
Decelerating	Bool	Out	减速阶段
FBErrorOccured	Bool	Out	功能块出错
FBErrorID	Dword	Out	功能块错误码

2 状态机



Note 1: From any state. An error in the axis occurred. Reference 2.3

Note 2: From any state. MC_Power.Enable = FALSE and there is no error in the axis.

Note 3: MC_Reset AND MC_Power.Status = FALSE

Note 4: MC_Reset AND MC_Power.Status = TRUE AND MC_Power.Enable = TRUE

Note 5: MC_Power.Enable = TRUE AND MC_Power.Status = TRUE

Note 6: MC_Stop.Done = TRUE AND MC_Stop.Execute = FALSE

Note7: MC_Jog only join Continuous Motion.

```
#define PMC_AS_Disabled          0
#define PMC_AS_ErrorStop         1
#define PMC_AS_Stopping          2
#define PMC_AS_Standstill        3
#define PMC_AS_DiscreteMotion    4
#define PMC_AS_ContinuousMotion  5
#define PMC_AS_Homing            7
```

3 ErrorID

基本格式如下：

库 + 功能块 + 错误码

3 3 位

库：高速 I/O 默认是 0

功能块号码：从 01 开始，功能块列表详见 6.2.4 功能块列表

错误码：从 01 开始，错误码详见表“计数器错误码列表”。错误码小于 500 是严重错误，大于 500 是功能块错误。

比如：31520，31 指的是 MC_WriteParameter_P， 520 指设置参数错误。

表 6-35 高速轴错误码列表：

错误码	定义	说明
001	ERR_NOT_POWER	MC_Power 没使能
002	ERR_UP_SOFTWARE_LIMIT	当前位置超出软件行程限制 +
003	ERR_DOWN_SOFTWARE_LIMIT	当前位置超出软件行程限制 -
004	ERR_AXIS_FUNC_UNUSED	高速轴功能未使能，在后台使能
005	ERR_INPUT_CHANNAL_NUM_INVALID	轴号无效，有效范围是 [0,3]
006	ERR_DEST_POS_OVER_SOFT_UP_LIMIT	目标位置超出软件上限
007	ERR_DEST_POS_OVER_SOFT_DOWN_LIMIT	目标位置超出软件下限
010	ERR_POS_DECPOINT_OVERLOW	减速点无效：位置模式下，重新定位时，减速长度大于实际距离
011	ERR_VEL_DECPOINT_OVERLOW	减速点无效：速度模式下，切换为定位时，减速长度大于实际距离
012	ERR_POS_PLSNUM_OVERLOW	超过 PLSNUM 最大定位长度 2147483647
013	ERR_POS_DECPOINT2_OVERLOW	2 次求减速点出错。
501	ERR_ACC_SET_OVERFLOW	加速度超出允许范围，超出 MC_WriteParameter_P 设定的最大加速度
502	ERR_ACC_SET_LOW	加速度设置太小，低于 MC_WriteParameter_P 设定的最小加速度
503	ERR_DEC_SET_OVERFLOW	减速度超出允许范围，超出 MC_WriteParameter_P 设定的最大减速度
504	ERR_DEC_SET_LOW	减速度设置太小，低于 MC_WriteParameter_P 设定的最小减速度
505	ERR_VEL_SET_OVERFLOW	速度设置超出允许范围，后台或者 MC_WriteParameter_P 设置
506	ERR_VEL_SET_LOW	速度设置太小
508	ERR_VEL_LESS_THAN_STARTVEL	速度小于启动偏置速度，后台设置启动偏置速度
509	ERR_STARTVEL_SET_LOW	起始速度太小
510	ERR_FBD_MOVEMODE_INVALIAD	功能块的运动模式无效，状态不对
511	ERR_WASNT_STANDSTILL	当前状态不是 STANDSTILL
512	ERR_WASNT_DISABLED	当前状态不是 DISABLE
513	ERR_IN_ERRORSTOP	当前状态是 ERRORSTOP
514	ERR_NOT_READY_FOR_MOTION	轴没准备好，不能运行
515	ERR_INVLALID_VELOCITY_MODE	无效的速度模式

错误码	定义	说明
516	ERR_INVLALID_POSTION_MODE	无效的位置模式
520	ERR_AXIS_WRITEPARAMETER_UNVALIAD	MC_WriteParameter_P 参数无效
521	ERR_AXIS_READPARAMETER_UNVALIAD	MC_ReadParameter_P 参数无效
522	ERR_HOME_MODE_UNVALIAD	回原模式无效，后台选择
523	ERR_AXIS_WRITEPARAMETER_HOME_MODE_UNVALIAD	回原模式设置无效

错误分为：轴错误和功能块错误

置为 ErrorStop 状态的条件为：

1） 发生轴错误；

2） 2 在 DiscreteMotion、ContinuousMotion 和 Homing 状态时发生功能块错误。

4 变速时机

模式	梯型加减速模式			S 型加减速模式		
	加速中	恒定速度	减速中	加速中	恒定速度	减速中
速度模式	---	允许改变速度	---	---	---	---
位置模式	---	允许改变速度	---	---	---	---
速度 -> 位置	---	允许改变速度	---	---	---	---
位置 -> 速度	---	允许改变速度	---	---	---	---
回原模式	---	---	---	---	---	---
JOG	---	---	---	---	---	---

注意：“---”符号表示不支持。

6.2.3 外部中断

1） 建议通常选用 1kHz 以下，极限输入频率 3kHz。

2） 如果外部输入中断频率过高，可能会导致系统无响应。

6.2.4 功能块列表

项目	功能块名称	编号
计数	HC_Counter	1
	HC_SetCompare	2
	HC_PresetValue	3
	HC_ControlInterrupt	4
	HC_TouchProbe	5
	HC_MeasurePulseWidth	6
	HC_Sample	7
	HC_ReadBoolParameter	8
	HC_WriteBoolParameter	9
	HC_SetCompareM	10
	HC_SetRing	11
	HC_ResetCmpOutput	12
	HC_ReadParameter	13
	HC_WriteParameter	14
	HC_WriteInterruptParameter	15
	HC_Reset	16
轴	Axis	20
	MC_Power_P	21
	MC_MoveAbsolute_P	22
	MC_MoveRelative_P	23
	MC_MoveVelocity_P	24
	MC_Stop_P	25
	MC_Home_P	27
	MC_Jog_P	28
	MC_Reset_P	29
	MC_ReadParameter_P	30
	MC_WriteParameter_P	31
	MC_SetPosition_P	32
	MC_ReadStatus_P	33

6.2.5 数码管显示

高速 I/O 数码管显示	含义
60	高速输入错误
62	高速输出错误
64	外部中断和比较一致中断出现错误

6.3 CANopen

6.3.1 CiA405

功能概述：提供符合 CANopen 通讯、CiA405 标准，并且支持 IEC61131-3 标准的功能块，功能块包含 SDO 读写访问、NMT 状态机、紧急时间报文、设备节点 ID、从站的通讯状态机、负载率显示。

对应库名：CmpHCCiA405(适用版本 V0.1.2.1 和以上)。

1 CiA405 简介

IEC 61131-3 可编程控制器 (CiA405) 的 CANopen 接口和设备配置文件介绍了两种从控制器访问 CANopen 网络的方法：网络变量和功能块。

网络变量通常会映射到要接收或要传输的 PDO 中。在对象字典中，可在定义的索引范围内访问 IEC 61131-3 变量。

配置文件还定义了一些特定于 CANopen 的功能块，例如 SDO、NMT 和紧急通讯服务。

CAA CiA 405 库提供了一组满足 CiA405 要求的功能块，用于从控制器（CANopen 主站）的应用程序（IEC61131-3 程序）访问 CANopen 网络。将 CANopen 管理器添加到控制器 CAN 总线接口后，控制器库管理器会自动声明该库。

在该库中，按如下方式组织功能块：

■ 网络管理 功能块：

CIA405 .NMT： 控制 CANopen 设备 NMT 状态

CIA405 .RECV_EMCY： 扫描所有设备的 EMCY 存储

CIA405 .RECV_EMCY_DEV： 获取指定设备的最后一条存储的 EMCY 消息

■ 自有节点 id 功能块：

CIA405 .GET_LOCAL_NODE_ID： 获取控制器 CANopen 管理器节点 ID

查询状态 功能块：

CIA405 .GET_CANOPEN_KERNEL_STATE： 获取 CANopen 内核当前状态

CIA405 .GET_STATE： 获取指定设备的当前状态

■ SDO 访问 功能块：

CIA405 .SDO_READ： 读取指定设备的任意大小的对象

CIA405 .SDO_READ4： 读取指定设备的最多 4 个字节的对象

CIA405 .SDO_WRITE： 写入指定设备的任意大小的对象

CIA405 .SDO_WRITE4： 写入指定设备的最多 4 个字节的对象

■ .CANopen 计数 功能块：

CIA405 .CANOPEN_COUNT： 显示收发帧数、网络负载率

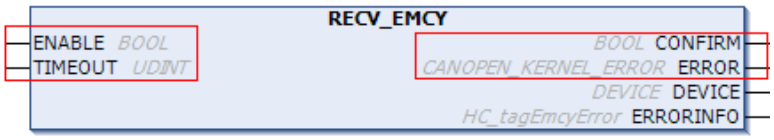
CIA405 .GET_MST_STATISTICS： CANOPEN_COUNT 的优化，显示收发帧数、网络负载率

2 功能块描述

功能块通用 I/O 和行为

本主题以 CIA405 .RECV_EMCY 功能块为示例，介绍 CAA CiA 405 库功能块的常规管理和执行。下文介绍了通用于所有功能块（CANopen 计数功能块之外）的 I/O。这些功能块继承自 CiA405Base 内部隐藏功能块。

下图中突出显示了 CAA CiA 405 库中所有功能块（CANopen 计数功能块之外）共有的参数：



下表介绍了 CAA CiA 405 库的所有功能块共有的输入参数和输出变量。

参数名称	参数类型	初始值	参数作用
VAR_IN			
ENABLE	BOOL	FALSE	启用功能块的执行。 在上升沿：执行开始。 在下降沿：如果执行未完成，则取消执行。否则，输出数据会复位为 0。
TIMEOUT	UDINT	0	最大执行时间（以毫秒为单位）。如果在收到响应之前达到超时，则执行会因超时错误而中止。 0 （出厂设置） = 禁用超时。 1...65535 = 超时值（以毫秒为单位）。
VAR_OUT			
CONFIRM	BOOL	FALSE	成功完成执行时为 TRUE。
ERROR	CANOPEN_ KERNEL_ERROR (USINT)	0	包含 CANopen 内核返回的检测到错误代码的执行。 00 （十六进制） = 未检测到任何执行错误。 01 （十六进制） = 检测到错误，但是代码在另一个功能块输出上提供（而不是来自 CANopen 内核）。 02..FF （十六进制） = CANopen 内核检测到的错误代码。

CANopen 内核检测到的错误代码

下面提供了错误代码及关联全局变量和说明。

检测到的错误代码	说明
CANOPEN_KERNEL_NO_ERROR = 00 （十六进制）	CANopen 内核未检测到错误。
CANOPEN_KERNEL_OTHER_ERROR = 01（十六进制）	如果 ERROR 输出 = 01 （十六进制），则已检测到错误；如果在功能块上存在其他错误代码 输出，则该输出包含更加具体的信息。 示例： CIA405 .SDO_READ 和 CIA405.SDO_WRITE 功能块：ERRORINFO 输出包含 SDO 中止消息的内容。 CIA405 .RECV_EMCY 和 CIA405.RECV_EMCY_DEV 功能块：ERRORINFO 输出包含接收到的 EMCY 消息的内容。如果 ERRORINFO 输出中提供了紧急消息，则 ERROR 输出 = 1 且 CONFIRM 输出 = 0。 CIA405 .GET_CANOPEN_KERNEL_STATE 功能块：绝不会提供十六进制 01 值，因为不存在任何其他错误代码输出。
CANOPEN_KERNEL_DATA_OVERFLOW = 02 （十六进制）	CANopen 对象的发送缓冲区或接收缓冲区溢出。
CANOPEN_KERNEL_TIMEOUT = 03 （十六进制）	发生功能块执行超时。
CANOPEN_KERNEL_CANBUS_OFF = 10（十六进制）	CANopen 节点从 CAN 总线断开连接。
CANOPEN_KERNEL_CAN_ERROR_PASSIVE = 11 （十六进制）	CANopen 节点处于错误被动状态：节点能够通讯，但是不允许在检测到错误时发送活动错误标志。

检测到的错误代码	说明
CANOPEN_INTERNAL_FB_ERROR = 21 （十六进制）	特定制造商错误代码。 注意： 21 （十六进制）到 FF （十六进制）为设备制造商专用值。

功能块执行图

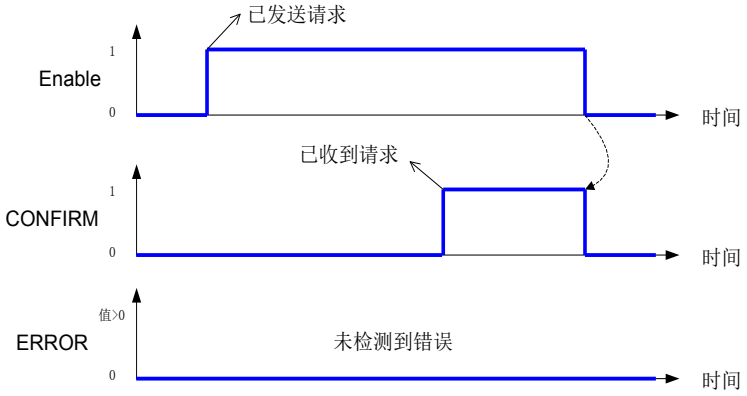
控制信号的行为

下面介绍了控制信号 ENABLE 、CONFIRM 和 ERROR 的三个典型行为：

- 执行在没有检测到错误的情况下结束
- 执行被应用程序取消
- 执行在检测到错误时中止或结束

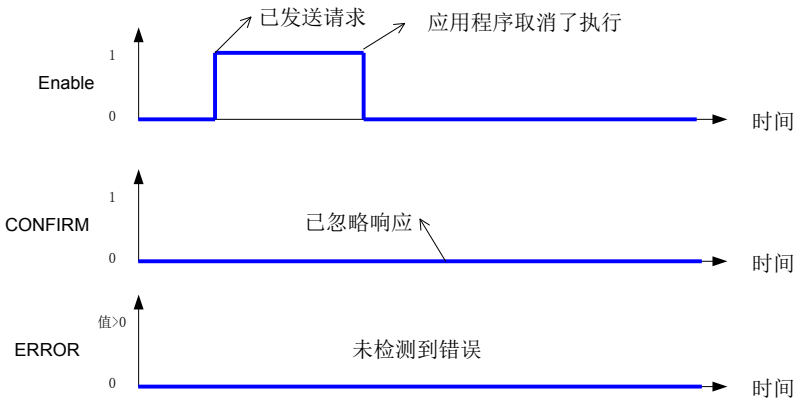
执行在没有检测到错误的情况下结束

一旦对当前请求做出响应，并且没有检测到任何错误，输出 CONFIRM 便会立即设置为 TRUE，并保留 TRUE 值（前提是在输入 ENABLE 为 TRUE 的情况下调用了该功能块）。如果在 ENABLE 被复位为 FALSE 的情况下调用了该功能块，则 CONFIRM 会复位为 FALSE，并且功能块可以开始新执行。



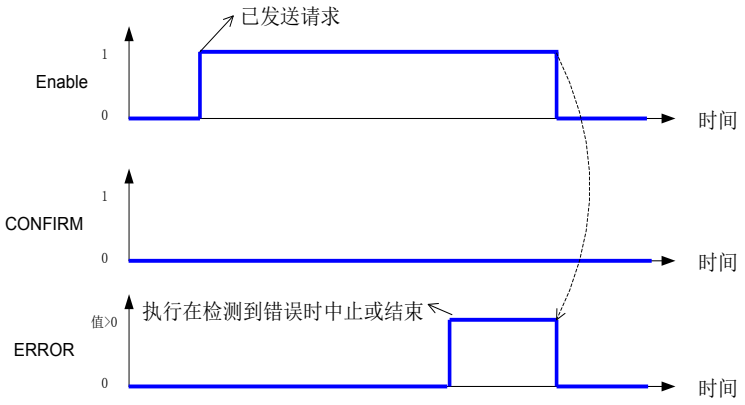
执行被应用程序取消

如果在当前执行结束之前，在 ENABLE 复位为 FALSE 的情况下调用该功能块，则会取消功能块执行。可能目前已对取消的请求做出回应，或该响应稍后将到达，但这些回应此时会被忽略。



执行在检测到错误时中止或结束

一旦当前执行由于收到 SDO 中止消息而中止，或由于检测到错误而结束，输出 ERROR 就会被设置为除 0 以外的值（有关检测到的错误代码的更多信息，请参阅 CANopen 内核检测到的错误代码 (参见第 28 页)）。如果在 ENABLE 复位为 FALSE 的情况下调用该功能块，则输出 ERROR 会复位为 0，并且功能块可以开始新执行。



3 网络管理功能块

1） 设备 NMT 状态管理

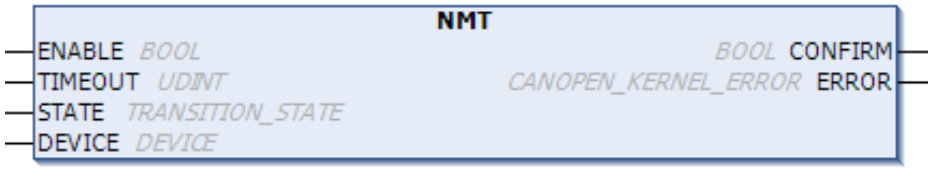
功能块描述

NMT	状态管理
使用 CIA405 .NMT 功能块可以从控制器应用程序控制 CANopen 设备的 NMT 状态。该功能块通过执行对 CANopen 目标设备的 NMT 服务请求，来执行请求的 NMT 状态转换。	

特定输入参数和输出变量描述

参数名称	参数类型	初始值	参数作用
VAR_IN			
DEVICE	DEVICE(USINT)	0	CANopen 目标设备节点 ID。 0 （缺省值） = 所有 NMT 从站设备 1...127 = 目标设备节点 ID
STATE	TRANSITION_STATE	FALSE	请求的 NMT 状态转换。
VAR_OUT			
无	无	无	无

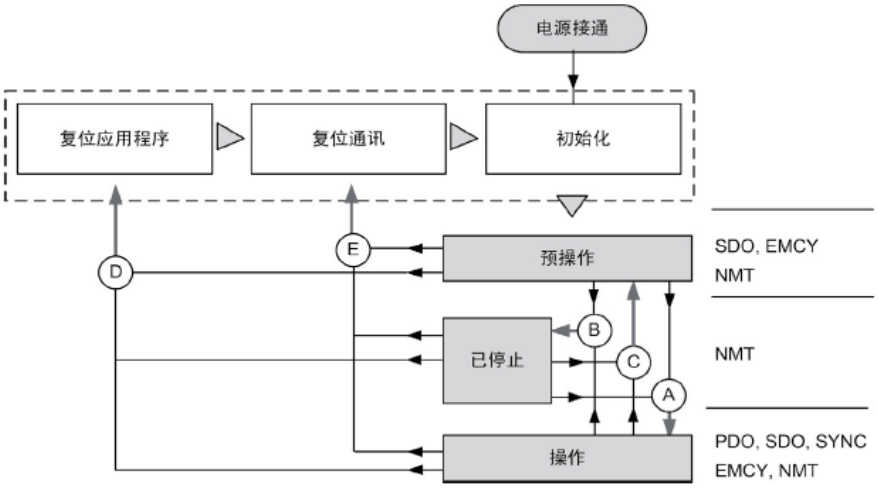
图形表示形式



CIA405 .TRANSITION_STATE ENUM

NMT 状态机描述主要操作中的 NMT 从站的初始化和状态。

下图显示的是 NMT 状态、关联的可用通讯对象（PDO、SDO、SYNC、EMCY 和 NMT）和 5 种状态转换（A 到 E）。

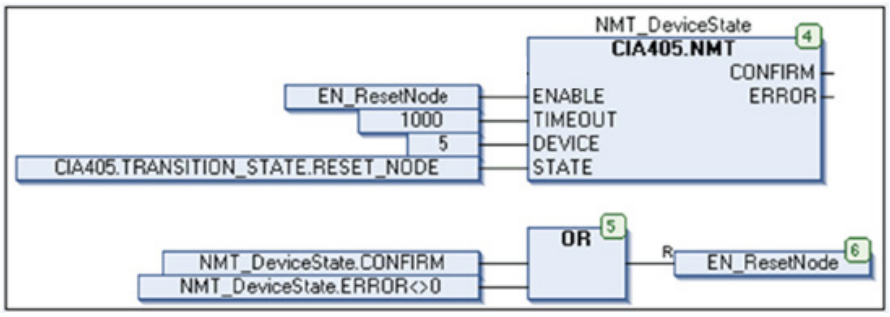


CIA405 .TRANSITION_STATE 枚举类型包含下表中介绍的 NMT 状态转换命令。

枚举器	值（十六进制）	说明
STOP_REMOTE_NODE	0004	切换到停止状态。（转换 B）
START_REMOTE_NODE	0005	切换到正常操作状态。（转换 A）
RESET_NODE	0006	切换到复位应用程序状态。加载设备配置文件的已保存数据，并自动从复位通讯切换到预操作状态。（转换 D）
RESET_COMMUNICATION	0007	切换到复位通讯状态。加载通讯配置文件的已存储数据，并自动切换到预操作状态。（转换 E）
ENTER_PRE_OPERATIONAL	007F	切换到预操作状态。（转换 C）
ALL_EXCEPT_NMT_AND_SENDER	0800	未实现（无效参数）

示例

以下示例显示了如何在超时 1 秒（1000 毫秒）的情况下，将“复位节点”命令发送到连接到第一个 CAN 总线接口的 CANopen 节点 5。该命令在布尔变量 EN_ResetNode 设置为 TRUE （由用户在线设置或应用程序设置）时发送。当执行成功结束（输出 CONFIRM = TRUE）或检测到错误（输出 ERROR <> 0）时，EN_ResetNode 命令会复位为 FALSE。



2） EMCY 消息扫描

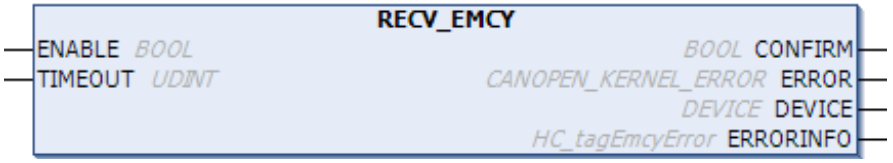
功能块描述

RECV_EMCY	消息扫描
CIA405.RECV_EMCY 功能块会在一个循环中对所有现有 CANopen 设备的紧急 (EMCY) 消息存储进行扫描，并返回找到的 EMCY 消息。	

特定输入参数和输出变量描述

参数名称	参数类型	初始值	参数作用
VAR_IN			
无	无	无	无
VAR_OUT			
DEVICE	DEVICE(USINT)	0	与返回的 EMCY 消息关联的 CANopen 设备的节点 ID。 0 = 未找到任何 EMCY 消息 1...127 = 设备节点 ID
ERRORINFO	CS.EMCY_ERROR	0	从节点 ID 为 DEVICE 的 CANopen 设备接收的最后一个 EMCY 消息。

图形表示形式



NOTE

- 启用该功能块后， EMCY 存储扫描将从上一个扫描停止点开始启动：
- 如果找到一条无错误之外的 EMCY 消息，则扫描会在此点结束，功能块返回该 EMCY 消息及关联设备节点 ID。
- 如果找到一条无错误 EMCY 消息，但该消息以前不是无错误消息，则扫描会在此点结束，功能块返回值为 0 的 EMCY 消息及关联设备节点 ID。
- 如果没有无错误之外的 EMCY 消息，且在一个完整扫描过程中未找到新的无错误 EMCY 消息，则功能块会返回值为 0 的 EMCY 消息和值为 0 的设备节点 ID。

3） 获取设备 EMCY 消息

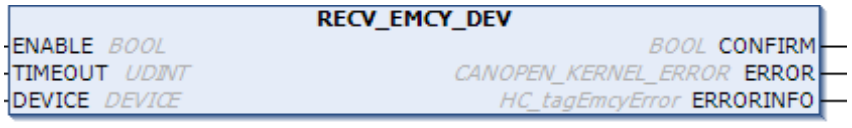
功能块描述

RECV_EMCY_DEV	获取设备 EMCY 消息
CIA405 .RECV_EMCY_DEV 功能块返回从指定 CANopen 设备接收的最后一条存储的紧急 (EMCY) 消息。	

特定输入参数和输出变量描述

参数名称	参数类型	初始值	参数作用
VAR_IN			
DEVICE	DEVICE(USINT)	0	要检查的 CANopen 设备的节点 ID。 0 （缺省值） = 本地设备（控制器） 1...127 = 设备节点 ID
VAR_OUT			
ERRORINFO	CS.EMCY_ERROR	0	从节点 ID 为 DEVICE 的 CANopen 设备接收的最后一个 EMCY 消息。

图形表示形式



CS.EMCY_ERROR 结构

CS.EMCY_ERROR 是与 EMCY 消息内容关联的结构，CS.EMCY_ERROR 包含以下元素。

元素	类型	说明
EMCY_ERROR_CODE	字	EMCY 消息的错误代码
ERROR_REGISTER	字节	EMCY 消息的错误寄存器（位字段）
ERROR_FIELD	ARRAY[1...5] OF BYTE	EMCY 消息的特定设备制造商错误字段

NOTE

此结构在 CAA CANopen 栈库（命名空间 = CS ）中声明。因此要使用结构全名（< 命名空间 >.< 数据类型 >）。CAA CiA 405 库命名空间为 CIA405 。

4 自有节点 ID 功能块

获取控制器 CANopen 节点 ID

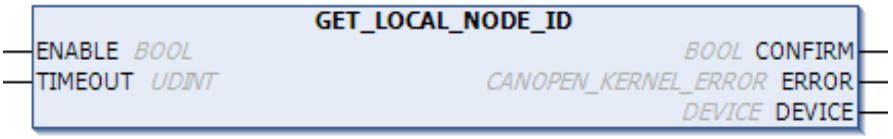
功能块描述

GET_LOCAL_NODE_ID	获取控制器节点 ID
CIA405 .GET_LOCAL_NODE_ID 功能块返回指定 CAN 总线接口上的控制器 CANopen 节点 ID。 图形表示形式	

特定输入参数和输出变量描述

参数名称	参数类型	初始值	参数作用
VAR_IN			
无	无	无	无
VAR_OUT			
DEVICE	DEVICE(USINT)	0	控制器的 CANopen 节点 ID。 0 = 无效 1...127 = 控制器节点 ID

图形表示形式



5 查询状态功能块

1） 获取 CANopen 内核状态

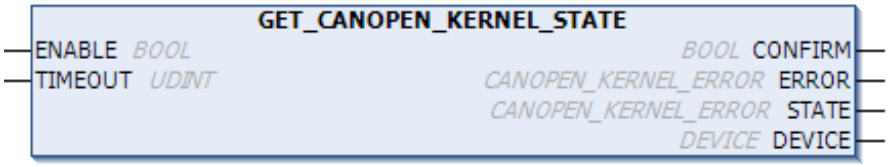
功能块描述

GET_CANOPEN_KERNEL_STATE	获取内核状态
CIA405 .GET_CANOPEN_KERNEL_STATE 功能块返回控制器 CANopen 内核的当前状态。	

特定输入参数和输出变量描述

参数名称	参数类型	初始值	参数作用
VAR_IN			
无	无	无	无
VAR_OUT			
STATE	CANOPEN_KERNEL_ERROR		控制器 CANopen 内核的当前状态
DEVICE	DEVICE(USINT)	0	控制器的 CANopen 节点 ID。 0 = 无效 1...127 = 控制器节点 ID

图形表示形式



2） 获取 CANopen 设备状态

功能块描述

GET _STATE	获取设备状态
CIA405 .GET_STATE 功能块在心跳或节点保护处于活动状态时，返回指定 CANopen 设备的当前 NMT 状态。	

特定输入参数和输出变量描述

参数名称	参数类型	初始值	参数作用
VAR_IN			
DEVICE	DEVICE(USINT)	0	要检查的 CANopen 设备的节点 ID。 0 （缺省值） = 本地设备（控制器） 1...127 = 设备节点 ID
VAR_OUT			
STATE	DEVICE_STATE	0	CANopen 设备的 NMT 状态。

图形表示形式

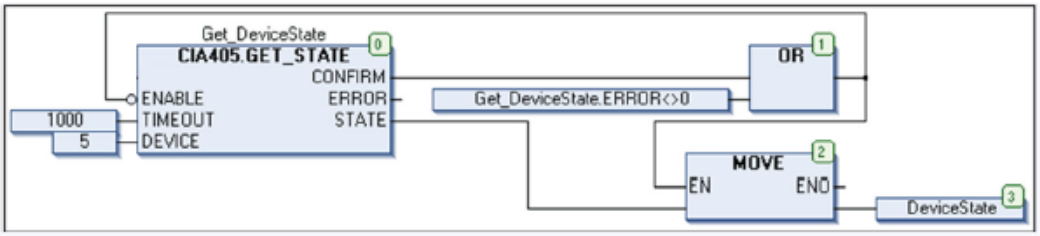
CIA405.DEVICE_STATE ENUM

CIA405 .DEVICE_STATE 枚举类型包含 CANopen 设备 NMT 状态的列表。

枚举器	值	说明
INIT	0	初始化状态
RESET_COMM	1	复位通讯状态
RESET_APP	2	复位应用程序状态
PRE_OPERATIONAL	3	预操作状态
STOPPED	4	停止状态
OPERATIONAL	5	正常操作状态
UNKNOWN	6	未知 NMT 状态。 所选设备的节点保护或心跳不处于活动状态，或者控制器不是心跳消费者。
NOT_AVAIL	7	NMT 状态不可用。 所选设备的节点保护或心跳处于活动状态，但是该设备在超时之前未正确报告其 NMT 状态。

示例

以下示例显示了如何在超时 1 秒（1000 毫秒）的情况下，获取连接到第一个 CAN 总线接口的 CANopen 节点 5 的状态。CIA405.GET_STATE 功能会自动执行，以进行连续的状态读取。设备 NMT 状态会复制到 CIA405. DEVICE_STATE 类型的 DeviceState 变量中。



6 SDO 访问功能块

1） 读取任意大小的 CANopen 对象

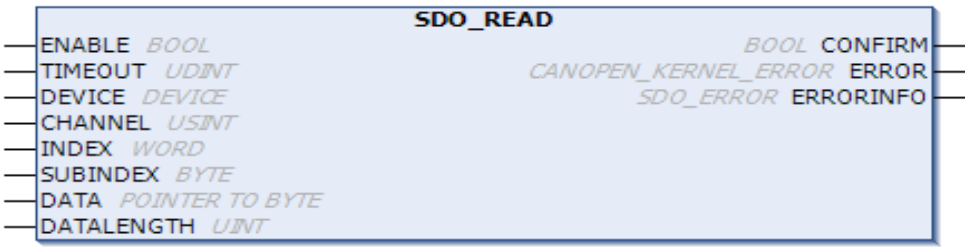
功能块描述

SDO_READ	读取任意大小的 CANopen 对象
CIA405 .SDO_READ 功能块用于通过 SDO 消息读取指定设备的任意大小的 CANopen 对象。	

特定输入参数和输出变量描述

参数名称	参数类型	初始值	参数作用
VAR_IN			
DEVICE	DEVICE (USINT)	0	CANopen 设备的节点 ID。 0 （缺省值） = 本地设备（控制器） 1...127 = 设备节点 ID
CHANNEL	USINT	1	SDO 通道编号。 缺省值 = 1
INDEX	WORD	0	对象索引。 范围：0000 （十六进制）...FFFF （十六进制）
SUBINDEX	BYTE	0	对象子索引。 范围：00 （十六进制）… FF （十六进制）
DATA	POINTER TO BYTE	NULL	接收从设备对象读取的数据的数据缓冲区地址。 必须使用 ADR 标准功能定义关联指针。
VAR_OUT			
ERRORINFO	SDO_ERROR (UDINT)	0	当 ERROR 输出 = 1 时，返回 SDO 中止消息内容（大小为 4 个字节）。
VAR_IN/ VAR_OUT			
DATALENGTH	UINT	0	作为输入：数据缓冲区的大小（以字节为单位）。 注意： 要确保 DATALENGTH 输入正确初始化（当功能块执行在 ENABLE 输入上升沿上开始时）为数据缓冲区大小，请使用 SIZEOF 标准功能。 作为输出：读取的对象的大小（以字节为单位）。

图形表示形式



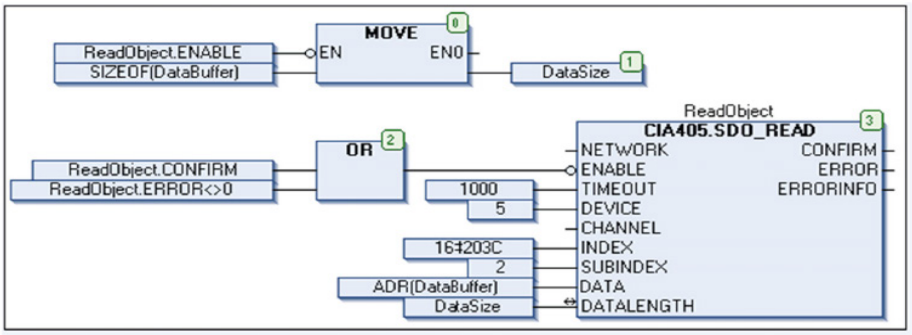
示例

以下示例显示了如何在超时 1 秒（1000 毫秒）的情况下，读取连接到第一个 CAN 总线接口的 CANopen 节点 5 的对象索引 203C （十六进制） / 子索引 02 （十六进制）。CIA405.SDO_READ 功能块实例 (ReadObject) 会自动执行，以进行连续读取。

变量 DataSize （UINT 类型）：

初始化为数据缓冲区大小（DataBuffer：N 字节的数组）（当功能块的 ENABLE 输入为 FALSE 时，且在启动下一个执行之前）。

包含在功能块输出 CONFIRM 的上升沿上读取的数据的大小（以字节为单位）（该示例不演示如何从数据缓冲区提取值，也不演示如何管理错误检测）。



NOTE

必须向该功能块传递以下这些特定参数：

1) 设备节点 ID
2) SDO 客户端 / 服务器通道（缺省情况下，仅定义一个通道）
3) CANopen 对象索引 / 子索引
4) 用来存储对象值的数据缓冲区的指针
5) 数据缓冲区大小

如果读取成功结束，则该功能块会返回读取的对象大小。数据在数据缓冲区中提供。

如果要读取的对象的大小小于或等于 4 个字节，最好使用 CIA405.SDO_READ4 功能块。

6） 读取最多 4 个字节的 CANopen 对象

功能块描述

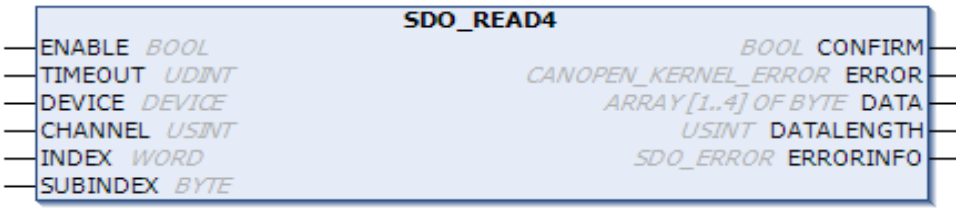
SDO_READ4	读取最多 4 个字节的 CANopen 对象
CIA405 .SDO_READ4 功能块用于通过 SDO 消息读取指定设备的最多 4 个字节的 CANopen 对象。	

特定输入参数和输出变量描述

参数名称	参数类型	初始值	参数作用
VAR_IN			
DEVICE	DEVICE (USINT)	0	CANopen 设备的节点 ID。 0 （缺省值） = 本地设备（控制器） 1...127 = 设备节点 ID
CHANNEL	USINT	1	SDO 通道编号。 缺省值 = 1
INDEX	WORD	0	对象索引。 范围：0000 （十六进制）...FFFF （十六进制）
SUBINDEX	BYTE	0	对象子索引。 范围：00 （十六进制）… FF （十六进制）

参数名称	参数类型	初始值	参数作用
VAR_OUT			
DATA	ARRAY[1...4] OF BYTE	0	接收从设备对象读取的数据的数据数组。
DATALENGTH	USINT	0	读取的对象的大小（以字节为单位）。
ERRORINFO	SDO_ERROR (UDINT)	0	当 ERROR 输出 = 1 时，返回 SDO 中止消息内容（大小为 4 个字节）。

图形表示形式



下表显示对象大小及相应的 DATA 数组内容。

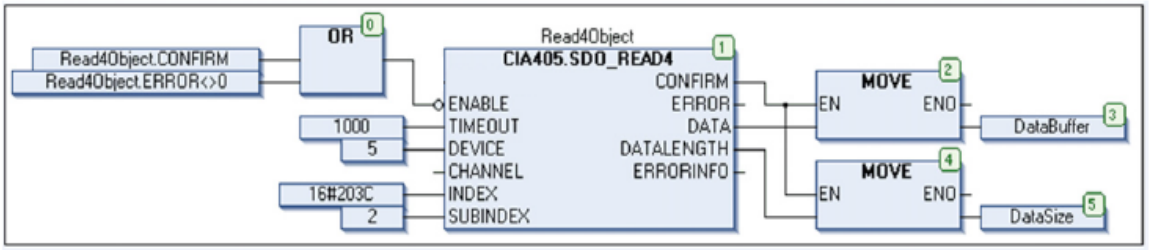
对象大小 示例	DATALENGTH	DATA(1)	DATA(2)	DATA(3)	DATA(4)
1 字节 01 （十六进制）	1	01 （十六进制）	无效	无效	无效
2 个字节 01 23 （十六进制）	2	LSB 23 （十六进制）	MSB 01 （十六进制）	无效	无效
3 个字节 01 23 45 （十六进制）	3	LSB 45 （十六进制）	23 （十六进制）	MSB 01 （十六进制）	无效
4 个字节 01 23 45 67 （十六进制）	4	LSB 67 （十六进制）	45 （十六进制）	23 （十六进制）	MSB 01 （十六进制）

LSB = 低有效字节

MSB = 最高有效字节

示例

以下示例显示了如何在超时 1 秒（1000 毫秒）的情况下，读取连接到第一个 CAN 总线接口的 CANopen 节点 5 的对象索引 203C （十六进制） / 子索引 02 （十六进制）。CIA405.SDO_READ4 功能块实例 (Read4Object) 会自动执行，以进行连续读取。变量 DataBuffer（4 字节数组）包含最后读取的数据的值。变量 DataSize（USINT 类型）包含最后读取的数据的大小（最大 4 字节）。该示例不演示如何管理错误检测。



NOTE

必须向该功能块传递以下这些特定参数：

1）设备节点 ID

2）SDO 客户端 / 服务器通道（缺省情况下，仅定义一个通道）

3）CANopen 对象索引 / 子索引

如果读取成功完成，则该功能块会返回读取的对象大小。数据在一个 4 字节数组中提供。

4） 写入任意大小的 CANopen 对象

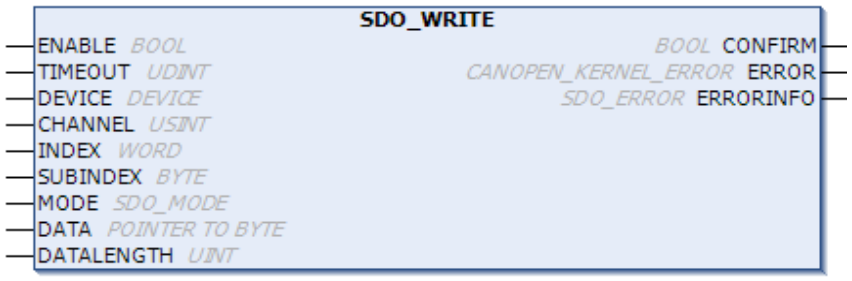
功能块描述

SDO_WRITE	写入任意大小的 CANopen 对象
CIA405 .SDO_WRITE 功能块用于通过 SDO 消息写入指定设备的任意大小的 CANopen 对象。	

特定输入参数和输出变量描述

参数名称	参数类型	初始值	参数作用
VAR_IN			
DEVICE	DEVICE (USINT)	0	CANopen 设备的节点 ID。 0 （缺省值） = 本地设备（控制器） 1...127 = 设备节点 ID
CHANNEL	USINT	1	SDO 通道编号。 缺省值 = 1
INDEX	WORD	0	对象索引。 范围：0000 （十六进制）...FFFF （十六进制）
SUBINDEX	BYTE	0	对象子索引。 范围：00 （十六进制）… FF （十六进制）
MODE	SDO_MODE	0	数据传输模式。 0 （缺省值） = AUTO （表示自动模式选择）
DATA	POINTER TO BYTE	NULL	用来存储要写入的对象值的数据缓冲区的地址。 必须使用 ADR 标准功能定义关联指针。
DATALENGTH	UINT	0	要写入的对象的大小（以字节为单位）。
VAR_OUT			
ERRORINFO	SDO_ERROR (UDINT)	0	当 ERROR 输出 = 1 时，返回 SDO 中止消息内容（大小为 4 个字节）。

图形表示形式

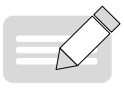


CIA405 .SDO_MODE ENUM

CIA405 .SDO_MODE 枚举类型包含 SDO 传输模式的列表。

枚举器	值	说明
AUTO	0	自动模式选择。

枚举器	值	说明
EXPEDITED	1	用于最多 4 个字节的数据的 SDO 加速模式。 数据在一个 SDO 请求中发送。
SEGMENTED	2	用于超过 4 个字节的数据的 SDO 分段模式。 数据被分割为 7 字节的段，通过连续的 SDO 确认请求发送。
BLOCK	3	用于超过 4 个字节的数据的 SDO 块模式。 通过连续帧发送的数据被分割为 7 字节的数据块，这些块不会进行确认。接收到所有块后，接收方会发送确认。 注意： 这是较快的传输模式，但是您的设备可能不支持，因为此模式是近期的 CANopen 配置附加项。


NOTE

必须向该功能块传递以下这些特定参数：
1) 设备节点 ID
2) SDO 客户端 / 服务器通道（缺省情况下，仅定义一个通道）
3) CANopen 对象索引 / 子索引
4) SDO 模式
5) 用来存储要写入的对象值的数据缓冲区的指针
6) 要写入的字节数
如果要写入的对象的大小小于或等于 4 个字节，则使用 CIA405.SDO_WRITE4 功能块。

7） 写入最多 4 个字节的 CANopen 对象

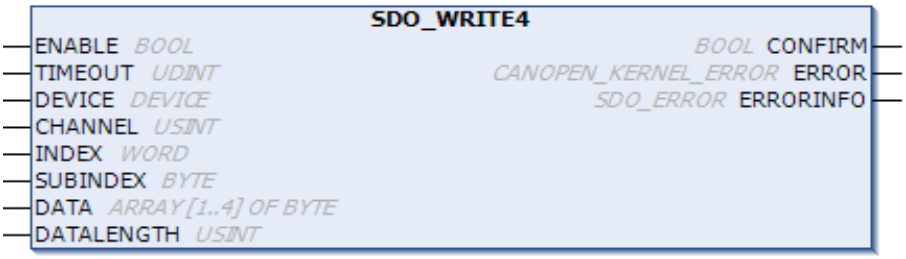
功能块描述

SDO_WRITE4	写入最多 4 个字节的 CANopen 对象
CIA405 .SDO_WRITE4 功能块用于通过 SDO 消息写入指定设备的最多 4 个字节的 CANopen 对象。	

特定输入参数和输出变量描述

参数名称	参数类型	初始值	参数作用
VAR_IN			
DEVICE	DEVICE (USINT)	0	CANopen 设备的节点 ID。 0 （缺省值） = 本地设备（控制器） 1...127 = 设备节点 ID
CHANNEL	USINT	1	SDO 通道编号。 缺省值 = 1
INDEX	WORD	0	对象索引。 范围：0000 （十六进制）...FFFF （十六进制）
SUBINDEX	BYTE	0	对象子索引。 范围：00 （十六进制）… FF （十六进制）
DATA	ARRAY[1...4] OF BYTE	0	用来存储要写入的对象值的数据数组。
DATALENGTH	USINT	0	要写入的对象的大小（以字节为单位）。
VAR_OUT			
ERRORINFO	SDO_ERROR (UDINT)	0	当 ERROR 输出 = 1 时，返回 SDO 中止消息内容（大小为 4 个字节）。

图形表示形式



下表显示对象大小及相应的 DATA 数组内容。

对象大小 示例	DATALENGTH	DATA(1)	DATA(2)	DATA(3)	DATA(4)
1 字节 01 （十六进制）	1	01 （十六进制）	无效	无效	无效
2 个字节 01 23 （十六进制）	2	LSB 23 （十六进制）	MSB 01 （十六进制）	无效	无效
3 个字节 01 23 45 （十六进制）	3	LSB 45 （十六进制）	23 （十六进制）	MSB 01（十六进制）	无效
4 个字节 01 23 45 67 （十六进制）	4	LSB 67 （十六进制）	45 （十六进制）	23（十六进制）	MSB 01 （十六进制）

LSB = 低有效字节

MSB = 最高有效字节

NOTE

必须向该功能块传递以下这些特定参数：

1) 设备节点 ID
2) SDO 客户端 / 服务器通道（缺省情况下，仅定义一个通道）
3) CANopen 对象索引 / 子索引
4) 要写入的值
5) 要写入的字节数（对象大小）

7 CANopen 计数功能块

■ 获取 CANopen 内核状态

功能块描述

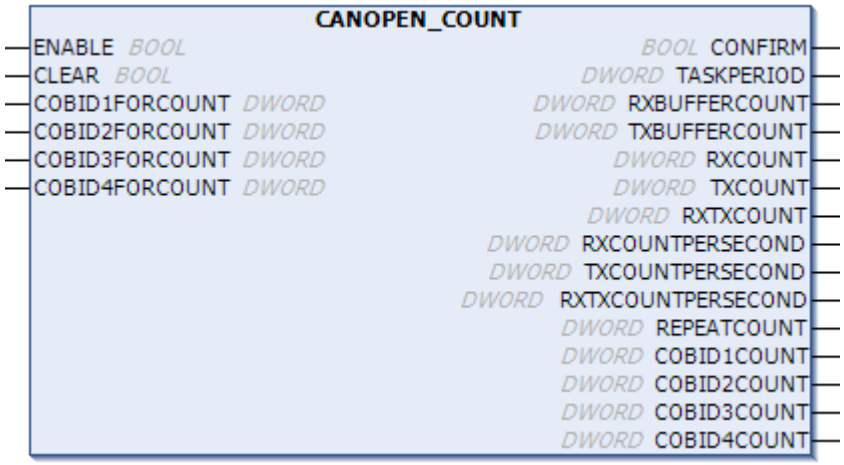
CANOPEN_COUNT	获取内核状态
CIA405 . CANOPEN_COUNT 功能块显示收发帧、网络负载率的显示。	

特定输入参数和输出变量描述

参数名称	参数类型	初始值	参数作用
VAR_IN			
ENABLE	BOOL	FALSE	启用功能块的执行。 在上升沿：执行开始。 在下降沿：如果执行未完成，则取消执行。否则，输出数据会复位为 0。
CLEAR	BOOL	FALSE	清除当前输出参数的值
COBID1FORCOUNT	DWORD	0	帧 ID 为 1 的数据帧，目前不支持使用
COBID2FORCOUNT	DWORD	0	帧 ID 为 2 的数据帧，目前不支持使用
COBID3FORCOUNT	DWORD	0	帧 ID 为 3 的数据帧，目前不支持使用
COBID4FORCOUNT	DWORD	0	帧 ID 为 4 的数据帧，目前不支持使用
VAR_OUT			

参数名称	参数类型	初始值	参数作用
CONFIRM	BOOL	FALSE	成功完成执行时为 TRUE。
TASKPERIOD	DWORD	0	网络负载率
RXBUFFERCOUNT	DWORD	0	CAN 控制器错误
TXBUFFERCOUNT	DWORD	0	目前不支持使用
RXCOUNT	DWORD	0	发送帧计数
TXCOUNT	DWORD	0	接收帧计数
RXTXCOUNT	DWORD	0	收发帧计数
RXCOUNTPERSECOND	DWORD	0	接收帧速率，目前不支持使用
TXCOUNTPERSECOND	DWORD	0	发送帧速率，目前不支持使用
RXTXCOUNTPERSECOND	DWORD	0	收发帧速率，目前不支持使用
REPEATCOUNT	DWORD	0	重发帧的数目，目前不支持使用
COBID1COUNT	DWORD	0	帧 ID 为 1 的数据帧数目，目前不支持使用
COBID2COUNT	DWORD	0	帧 ID 为 2 的数据帧数目，目前不支持使用
COBID3COUNT	DWORD	0	帧 ID 为 3 的数据帧数目，目前不支持使用
COBID4COUNT	DWORD	0	帧 ID 为 4 的数据帧数目，目前不支持使用

图形表示形式



■ 获取 CANopen 内核状态

功能块描述

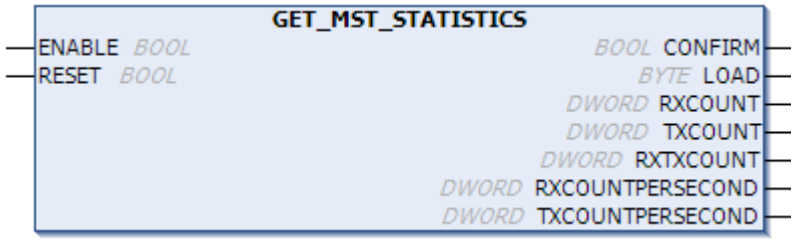
GET_MST_STATISTICS	获取内核状态
CIA405 . GET_MST_STATISTICS 功能块显示收发帧、网络负载率的显示。	

特定输入参数和输出变量描述

参数名称	参数类型	初始值	参数作用
VAR_IN			
ENABLE	BOOL	FALSE	启用功能块的执行。 在上升沿：执行开始。 在下降沿：如果执行未完成，则取消执行。否则，输出数据会复位为 0。
RESET	BOOL	FALSE	清除当前输出参数的值
VAR_OUT			
CONFIRM	BOOL	FALSE	成功完成执行时为 TRUE。
LOAD	BYTE	0	网络负载率
RXCOUNT	DWORD	0	发送帧计数
TXCOUNT	DWORD	0	接收帧计数
RXTXCOUNT	DWORD	0	收发帧计数
RXCOUNTPERSECOND	DWORD	0	接收帧速率，目前不支持使用

参数名称	参数类型	初始值	参数作用
TXCOUNTPERSECOND	DWORD	0	发送帧速率，目前不支持使用

图形表示形式

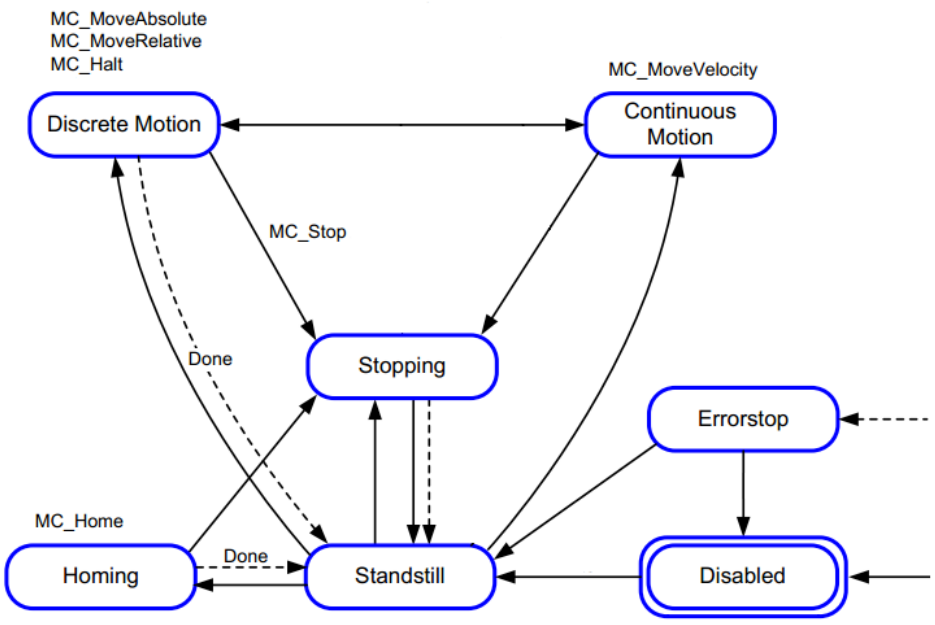


8 术语

名称	定义
CAN	控制器区域网络 (CAN) 是一种串行总线系统，其目的是实现嵌入式控制。
CANopen	CANopen 是用于嵌入式控制系统的符合国际标准 (EN 50325-4) 的基于 CAN 的高层次协议。
CiA	CAN in Automation (CiA) 是用户和制造商组成的国际组织，致力于开发和支持 CANopen 及其他基于 CAN 的高层次协议。
CiA405	IEC 61131-3 可编程控制器的 CANopen 接口和设备配置文件
COB	CANopen 协议通讯对象
EMCY	紧急
NMT	CANopen 网络管理
OD	协议对象字典
PDO	协议过程数据对象
SDO	协议服务数据对象

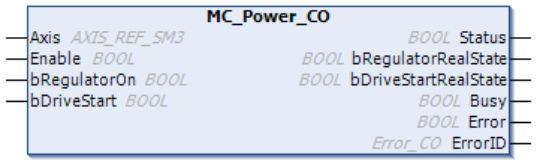
6.3.2 CANopen 402

功能块设计以 PLCopen、CIA402 标准为参考，主要支持 CIA402 的轮廓速度模式、轮廓位置模式、回零模式等线性控制模式。不支持旋转模式、轴软限位、硬件限位。线性模式控制下位置计算范围为 -2147483648 到 2147483647。轴状态切换以 PLCopen 状态机图为参考设计，具体转换图如下所示。



1 MC_Power_CO

1） 指令格式

图形表现		
		
指令	名称	ST 表现
MC_Power_CO	电机使能	MC_Power_CO(Axis:= , Enable:= , bRegulatorOn:= , bDriveStart:= , Status=> , bRegulatorRealState=> , bDriveStartRealState=> , Busy=> , Error=> , ErrorID=>);

2） 相关变量

■ 输入输出变量

输入输出变量	名称	数据类型	有效范围	初始值	描述
Axis	轴	AXIS_REF_HC_CO	—	—	CANopen 轴

■ 输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Enable	使能	BOOL	[FALSE,TRUE]	FALSE	电平触发，TRUE：使能功能块。
bRegulatorOn	电机使能	BOOL	[FALSE,TRUE]	FALSE	使能电机，配合 DriveStart 使用
bDriveStart	快速急停	BOOL	[FALSE,TRUE]	FALSE	不支持快速急停

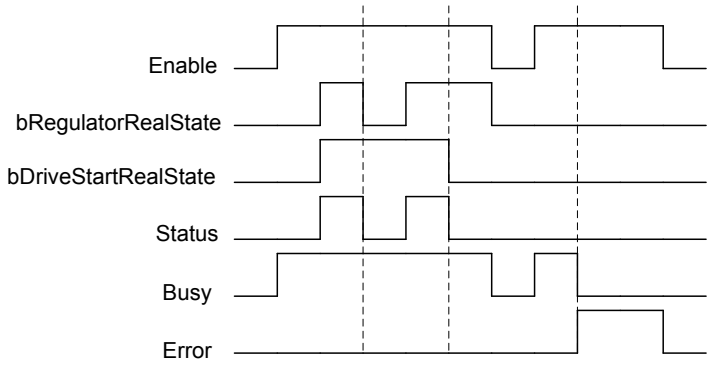
■ 输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Status	轴准备运动状态	BOOL	[FALSE,TRUE]	FALSE	轴运动准备状态。
bRegulatorRealState	使能有效状态	BOOL	[FALSE,TRUE]	FALSE	驱动器使能状态，TRUE 表示已经使能。
bDriveStartRealState	快速停止机制的有效状态	BOOL	[FALSE,TRUE]	FALSE	紧急停止，暂不支持，
Busy	执行标志	BOOL	[FALSE,TRUE]	FALSE	TRUE: 功能块正在执行
Error	错误标志	BOOL	[FALSE,TRUE]	FALSE	TRUE: 功能块内部发生错误
ErrorID	错误 ID	ERROR_CO	-	NO_ERROR	错误 ID，具体参考 ERROR_CO

3） 功能说明

- 本功能块主要实现轴的使能控制、快速急停控制，快速急停功能要根据驱动器是否支持来使用。
- RegulatorRealState=TRUE 表示使能上，但驱动器不一定可以控制。只有 Status=TRUE 的条件下，才可以进行运动控制，
- 快速停机功能的停机方式由对象字典 16#605A 的值决定，如果需要更改停机方式，在启动参数配置 16#605A 值，视驱动器对该功能的支持情况决定。

4） 时序图



5） 错误说明

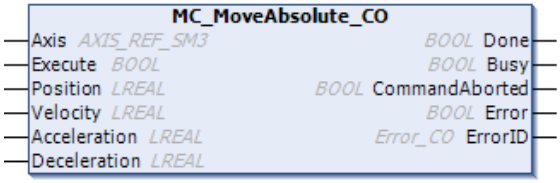
当出现错误时，通过帮助文档找到 COUNTER_ERROR 中 ErrorID 所对应的错误，根据错误定位到错误原因。

错误 ID	枚举值	描述
0	NO_ERROR	没有错误
1	DI_GENERAL_COMMUNICATION_ERROR	轴通讯错误
2	DI_AXIS_ERROR	驱动轴报错
21	WRONG_OPMODE	错误的操作模式
33	AXIS_IN_ERRORSTOP	轴在 errorstop 状态
34	AXIS_NOT_READY_FOR_MOTION	轴当前状态错误，不能进行运动
35	MA_MR_MODULO_ACT_POS_NOT_MAPPED	保留
36	MV_INVALID_VELACCDEC_VALUES	不合理的速度或者加减速度
80	RAG_ERROR_DURING_STARTUP	保留
81	RAG_ERROR_WRITING_COMSTATE	保留
82	RAG_ERROR_READING_COMSTATE	保留
90	CGR_ZERO_VALUES	保留
91	CGR_AXIS_POWERED	保留
93	CGR_MODULOPERIOD_NOT_INTEGRAL	保留
94	CGR_MOVEMENTTYPE_INVALID	保留
95	CGR_MODULOPERIOD_NON_POSITIVE	保留
96	CGR_MODULOPERIOD_TOO_SMALL	保留
97	CGR_MODULOPERIOD_TOO_LARGE	保留
120	R_NO_ERROR_TO_RESET	没有错误需要复位
121	R_DRIVE_DOESNT_ANSWER	轴没有应答
122	R_ERROR_NOT_RESETTABLE	错误不能被复位
123	R_DRIVE_DOESNT_ANSWER_IN_TIME	保留
130	RP_PARAM_UNKNOWN	读参数错误
131	RP_REQUESTING_ERROR	通讯请求错误

错误 ID	枚举值	描述
132	RP_RCV_PARAM_CONVERSION_ERROR	保留
133	RP_LOCAL_PARAM_NOT_DONE_IMMEDIATELY	保留
134	RP_CANNOT_SEND_MSG	不能发送消息
140	WP_PARAM_INVALID	写参数错误
141	WP_SENDING_ERROR	通讯发送错误
142	WP_TMT_PARAM_CONVERSION_ERROR	保留
143	WP_LOCAL_PARAM_NOT_DONE_IMMEDIATELY	保留
144	WP_CANNOT_SEND_MSG	不能发送消息
170	H_AXIS_WASNT_STANDSTILL	回零需要轴在 standstill 状态
183	MS_AXIS_IN_ERRORSTOP	mc_stop 功能块不能在 errorstop
184	MS_AXIS_IN_STOPPING	mc_stop 功能块不能在 stopping
10000	TIMEOUT_CHANGING_OPMODE	切换模式超时
10001	INTERNAL_UNKNOWN_CMD	保留
10002	CANNOT_START_MOVEMENT	保留
10003	CANNOT_START_HOMING	保留
10004	STOP_ALREADY_ACTIVE	保留
10005	POWER_ALREADY_ACTIVE	保留
10006	SMC_DI_VOLTAGE_DISABLED	轴运动过程断开使能

2 MC_MoveAbsolute_CO

1） 指令格式

<table>
<tr><th colspan="3">图形表现</th></tr>
<tr><td colspan="3"> 
</td></tr>
<tr><th>指令</th><th>名称</th><th>ST 表现</th></tr>
<tr><td>MC_
MoveAbsolute_CO</td><td>绝对定位</td><td>MC_MoveAbsolute_CO(
Axis:= ,
Execute:= ,
Position:= ,
Velocity:= ,
Acceleration:= ,
Deceleration:= ,
Done=> ,
Busy=> ,
CommandAborted=> ,
Error=> ,
ErrorID=>);</td></tr>
</table>

2） 相关变量

■ 输入输出变量

输入输出变量	名称	数据类型	有效范围	初始值	描述
Axis	轴	AXIS_REF_HC_CO	—	—	CANopen 轴

■ 输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Execute	执行输入	BOOL	[FALSE,TRUE]	FALSE	上升沿触发，由 FALSE->TRUE：开始执行
Position	目标位置	LREAL	(0, 1.7E 308)	0.0	单位：unit，仅支持线性模式。
Velocity	目标速度	LREAL	(0, 1.7E 308)	0.0	单位：unit/s，仅支持线性模式。
Acceleration	目标加速度	LREAL	(0, 1.7E 308)	0.0	单位：unit/s^2，仅支持线性模式。
Deceleration	目标加速度	LREAL	(0, 1.7E 308)	0.0	单位：unit/s^2，仅支持线性模式。

■ 输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Done	完成标志	BOOL	[FALSE,TRUE]	FALSE	TRUE：运动执行完成，定位成功。
Busy	执行标志	BOOL	[FALSE,TRUE]	FALSE	TRUE：运动正在执行。
CommandAborted	中止标志	BOOL	[FALSE,TRUE]	FALSE	TRUE：运动被另外的运动中断。
Error	错误标志	BOOL	[FALSE,TRUE]	FALSE	TRUE：运动发生错误。
ErrorID	错误 ID	ERROR_CO	-	NO_ERROR	错误 ID，具体参考 ERROR_CO。

3） 功能说明

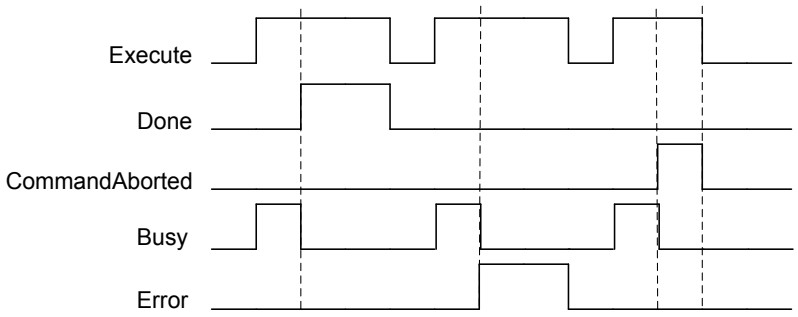
■ 绝对位置定位功能，将受控轴运动到指定绝对位置。绝对置位的脉冲计数范围（-2147483648 到 2147483647），如果计数脉冲大于此范围，位置将计数错误！运动功能块不能正常运行。

■ Velocity、Acceleration、Deceleration 必须大于 0。

■ 如果定位长度超出（-2147483648 到 2147483647），缩小伺服电子齿轮比系数，或者降低步进驱动器。

■ 当定位完成后，Done 置 TRUE，并且受控轴 .nAxisState 为 StandSitll。

4） 时序图

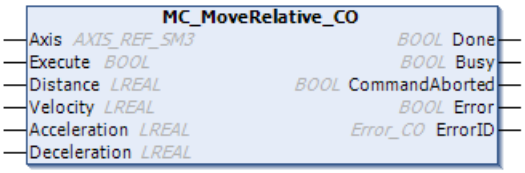


5） 错误说明

当出现错误时，通过帮助文档找到 COUNTER_ERROR 中 ErrorID 所对应的错误，根据错误内容定位到错误原因。

3 MC_MoveRelative_CO

1） 指令格式

图形表现


指令	名称	ST 表现
MC_MoveRelative_CO	相对定位	MC_MoveRelative_CO(Axis:= , Execute:= , Distance:= , Velocity:= , Acceleration:= , Deceleration:= , Done=> , Busy=> , CommandAborted=> , Error=> , ErrorID=>);

2） 相关变量

■ 输入输出变量

输入输出变量	名称	数据类型	有效范围	初始值	描述
Axis	轴	AXIS_REF_HC_CO	—	—	CANopen 轴

■ 输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Execute	执行输入	BOOL	[FALSE,TRUE]	FALSE	上升沿触发，由 FALSE->TRUE：开始执行
Distance	相对距离	LREAL	(0, 1.7E 308)	0.0	单位：unit，仅支持线性模式。
Velocity	目标速度	LREAL	(0, 1.7E 308)	0.0	单位：unit/s，仅支持线性模式。
Acceleration	目标加速度	LREAL	(0, 1.7E 308)	0.0	单位：unit/s^2，仅支持线性模式。
Deceleration	目标加速度	LREAL	(0, 1.7E 308)	0.0	单位：unit/s^2，仅支持线性模式。

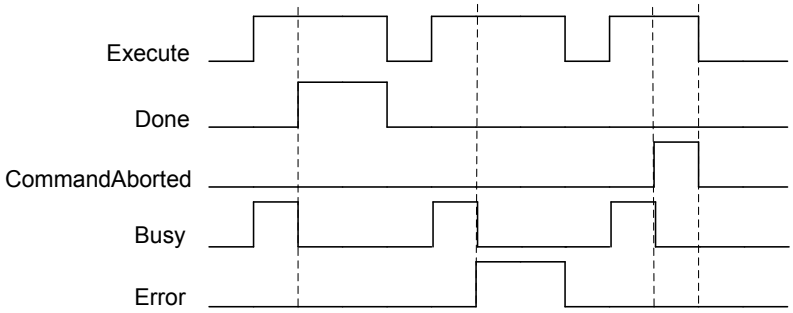
■ 输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Done	完成标志	BOOL	[FALSE,TRUE]	FALSE	TRUE：运动执行完成，定位成功。
Busy	执行标志	BOOL	[FALSE,TRUE]	FALSE	TRUE：运动正在执行。
CommandAborted	中止标志	BOOL	[FALSE,TRUE]	FALSE	TRUE：运动被另外的运动中断。
Error	错误标志	BOOL	[FALSE,TRUE]	FALSE	TRUE：运动发生错误。
ErrorID	错误 ID	ERROR_CO	-	NO_ERROR	错误 ID，具体参考 ERROR_CO。

3） 功能说明

- 相对定位功能块，将受控轴运动一段距离，运动目标位置 = 功能块执行时刻静止位置 + 相对距离。 如果轴在绝对定位指令运动过程中，启动相对定位指令，运动目标位置 = 绝对定义的目标位置 + 相对距离。
- 如果定位长度超出（-2147483648 到 2147483647），缩小伺服电子齿轮比系数，或者降低步进驱动器
- 配合汇川 IS620_CO 使用时，使用相对运动命令打断正在执行的相对定位命令或者绝对定位命令，本次正要执行的相对命令计算的的绝对的目标位置 = 正在执行的相对运动或者绝对命令应该到达的位置 + 本次运动指令的相对距离。
- 当定位完成后，Done 置 TRUE，并且受控轴 .nAxisState 为 StandSitll。

4） 时序图

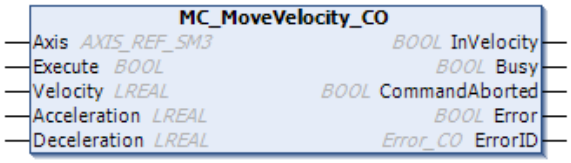


5） 错误说明

当出现错误时，通过帮助文档找到 COUNTER_ERROR 中 ErrorID 所对应的错误，根据错误内容定位到错误原因。

4 MC_MoveVelocity_CO

1） 指令格式

图形表现		
		
指令	名称	ST 表现
MC_MoveVelocity_CO	速度模式运动	MC_MoveVelocity_CO(Axis:= , Execute:= , Velocity:= , Acceleration:= , Deceleration:= , InVelocity=> , Busy=> , CommandAborted=> , Error=> , ErrorID=>);

2） 相关变量

■ 输入输出变量

输入输出变量	名称	数据类型	有效范围	初始值	描述
Axis	轴	AXIS_REF_HC_CO	—	—	CANopen 轴

■ 输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Execute	执行输入	BOOL	[FALSE,TRUE]	FALSE	上升沿触发，由 FALSE->TRUE：开始执行
Velocity	目标速度	LREAL	(-1.7E 308, 1.7E 308)	0.0	单位：unit/s，仅支持线性模式。
Acceleration	目标加速度	LREAL	(0, 1.7E 308)	0.0	单位：unit/s^2，仅支持线性模式。
Deceleration	目标加速度	LREAL	(0, 1.7E 308)	0.0	单位：unit/s^2，仅支持线性模式。

■ 输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
InVelocity	速度到达标志	BOOL	[FALSE,TRUE]	FALSE	TRUE：轴速度达到设置速度值。
Busy	执行标志	BOOL	[FALSE,TRUE]	FALSE	TRUE：运动正在执行。
CommandAborted	中止标志	BOOL	[FALSE,TRUE]	FALSE	TRUE：运动被另外的运动中断。
Error	错误标志	BOOL	[FALSE,TRUE]	FALSE	TRUE：运动发生错误。
ErrorID	错误 ID	ERROR_CO	-	NO_ERROR	错误 ID，具体参考 ERROR_CO。

3） 功能说明

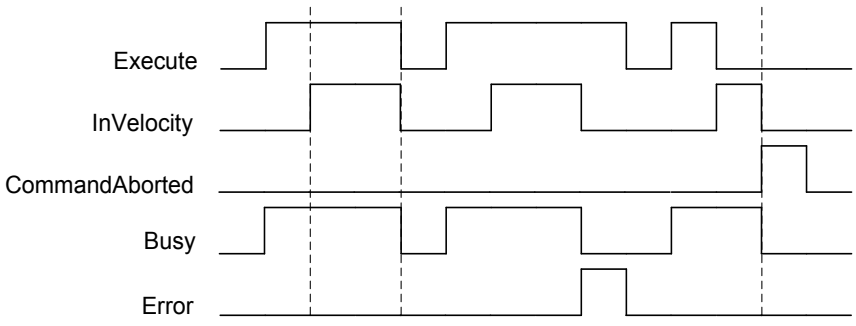
■ 速度运动功能块，受控轴到达设置速度后，将以该速度一直运行。如果要停止轴必须以另一个命令打断中止当前功能块，被另一个命令打断时输出参数 InVelocity 状态必复位。

■ 位置将在（-2147483648 到 2147483647）内变化。

■ 配合 IS620N_CO 使用，如果在运动模式下位置产生一次溢出（脉冲位置由 2147483647 到 -2147483648，或者 -2147483648 到 2147483647），切换到位置模式，必须先回零，然后使用位置运动命令，因为位置模式不记忆溢出次数（与同步周期模式不同）。

■ 正常运动时，受控轴 .nAxisState 为 ContinuesMoiton。

4） 时序图

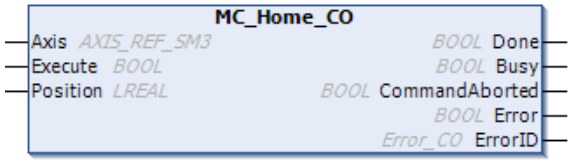


5） 错误说明

当出现错误时，通过帮助文档找到 COUNTER_ERROR 中 ErrorID 所对应的错误，根据错误内容定位到错误原因。

5 MC_Home_CO

1） 指令格式

图形表现		
		
指令	名称	ST 表现
MC_Home_CO	回零运动	MC_Home_CO(Axis:= , Execute:= , Position:= , Done=> , Busy=> , CommandAborted=> , Error=> , ErrorID=>);

2） 相关变量

■ 输入输出变量

输入输出变量	名称	数据类型	有效范围	初始值	描述
Axis	轴	AXIS_REF_HC_CO	—	—	CANopen 轴

■ 输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Execute	执行输入	BOOL	[FALSE,TRUE]	FALSE	上升沿触发，由 FALSE->TRUE：开始执行
position	原点偏移	LREAL	(0, 1.7E 308)	0.0	单位：unit/s，仅支持线性模式。

■ 输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Done	完成标志	BOOL	[FALSE,TRUE]	FALSE	TRUE：回零完成。
Busy	执行标志	BOOL	[FALSE,TRUE]	FALSE	TRUE：运动正在执行。
CommandAborted	中止标志	BOOL	[FALSE,TRUE]	FALSE	TRUE：运动被另外的运动中断。
Error	错误标志	BOOL	[FALSE,TRUE]	FALSE	TRUE：运动发生错误。
ErrorID	错误 ID	ERROR_CO	-	NO_ERROR	错误 ID，具体参考 ERROR_CO。

3） 功能说明

■ 回零功能块，受控轴以设置的模式执行回零运动，当轴检测到参考信号停止，当前的绝对位置将更新为输入参数“position”的值。

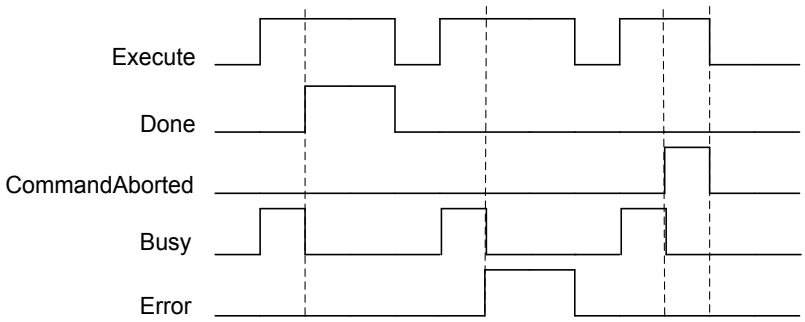
■ 回零不能被定位命令、速度命令打断。

■ 正常运动时，受控轴 .nAxisState 状态为 Homing。

■ Position 回零偏移，相对零点的绝对位置偏移距离，单位：unit。当回零检测到硬件型号，并停止后，当

前绝位置设置为 Position 值，并且受控轴 .nAxisState 为 StandSitll。

4） 时序图



5） 错误说明

当出现错误时，通过帮助文档找到 COUNTER_ERROR 中 ErrorID 所对应的错误，根据错误内容定位到错误原因。

6 MC_Stop_CO

1） 指令格式

图形表现		
		
指令	名称	ST 表现
MC_Stop_CO	运动停止	MC_Stop_CO(Axis:= , Execute:= , Done=> , Busy=> , Error=> , ErrorID=>);

2） 相关变量

■ 输入输出变量

输入输出变量	名称	数据类型	有效范围	初始值	描述
Axis	轴	AXIS_REF_HC_CO	—	—	CANopen 轴

■ 输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Execute	执行输入	BOOL	[FALSE,TRUE]	FALSE	上升沿触发，由 FALSE->TRUE：开始执行

■ 输出变量

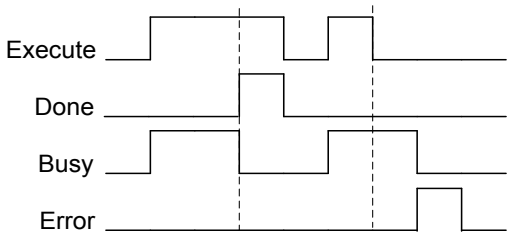
输出变量	名称	数据类型	有效范围	初始值	描述
Done	完成标志	BOOL	[FALSE,TRUE]	FALSE	TRUE：停止。
Busy	执行标志	BOOL	[FALSE,TRUE]	FALSE	TRUE：正在执行停止命令。

输出变量	名称	数据类型	有效范围	初始值	描述
Error	错误标志	BOOL	[FALSE,TRUE]	FALSE	TRUE：停止命令发生错误。
ErrorID	错误 ID	ERROR_CO	-	NO_ERROR	错误 ID，具体参考 ERROR_CO。

3） 功能说明

- 运动停止功能块，上升沿触发执行后受控轴运动停止。只要 Execute=TRUE 或停止未完成，轴不能有其他运动命令，轴状态为“Stopping”。 停止完成并且 Execute 被复位后，轴状态被设置为“Standstill”。
- 由于很多厂商不支持控制字 bit8 执行停止，停止功能实际上是将模式切换到速度模式，并将目标速度设置为 0 执行速度命令方式停止。
- 运动停止后，必须复位 Execute 状态，否则不能执行其他运动命令。

4） 时序图

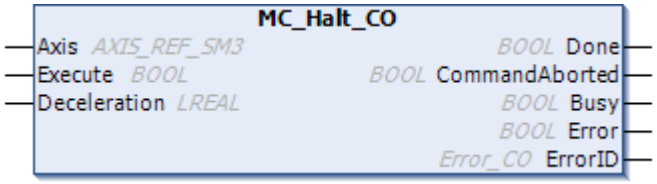


5） 错误说明

当出现错误时，通过帮助文档找到 COUNTER_ERROR 中 ErrorID 所对应的错误，根据错误内容定位到错误原因。

7 MC_Halt_CO

1） 指令格式

图形表现		
		
指令	名称	ST 表现
MC_Halt_CO	运动暂停	MC_Halt_CO(Axis:= , Execute:= , Deceleration:= , Done=> , CommandAborted=> , Busy=> , Error=> , ErrorID=>);

2） 相关变量

- 输入输出变量

输入输出变量	名称	数据类型	有效范围	初始值	描述
Axis	轴	AXIS_REF_HC_CO	—	—	CANopen 轴

- 输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Execute	功能块执行端口	BOOL	[FALSE,TRUE]	FALSE	上升沿触发，由 FALSE->TRUE：开始执行
Deceleration	减速度	LREAL	(0, 1.7E 308)	0.0	单位：unit/s^2，仅支持线性模式。

■ 输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Done	完成标志	BOOL	[FALSE,TRUE]	FALSE	TRUE：停止。
Busy	执行标志	BOOL	[FALSE,TRUE]	FALSE	TRUE：正在执行停止命令。
CommandAborted	中止标志	BOOL	[FALSE,TRUE]	FALSE	TRUE：运动被另外的运动中断。
Error	错误标志	BOOL	[FALSE,TRUE]	FALSE	TRUE：停止命令发生错误。
ErrorID	错误 ID	ERROR_CO	-	NO_ERROR	错误 ID，具体参考 ERROR_CO。

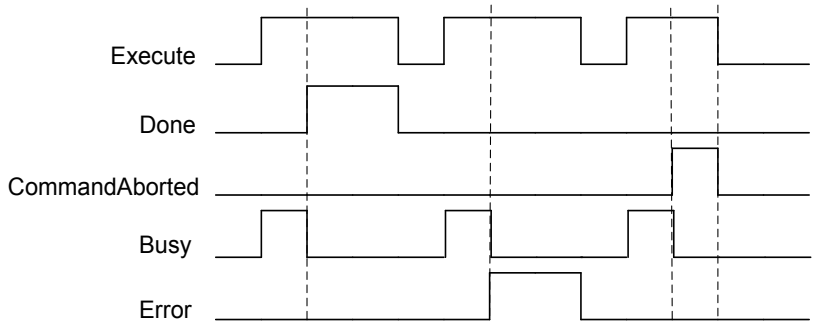
3） 功能说明

■ 命令控制运动停止，轴被切换到状态“DiscreteMotion”，直到速度为零。 当输出 Done=TRUE 时，状态切换为“Standstill”。

■ MC_Halt_CO 执行停止过程中可以被其他运动指令打断。

■ MC_Halt_CO 执行完成后，不需要复位 Execute 端口状态可以继续使用其他运动命令，与 MC_Stop_CO 不同。

4） 时序图

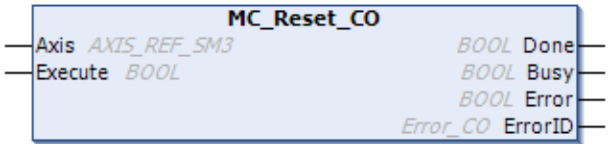


5） 错误说明

当出现错误时，通过帮助文档找到 COUNTER_ERROR 中 ErrorID 所对应的错误，根据错误内容定位到错误原因。

8 MC_Reset_CO

1） 指令格式

图形表现		
		
指令	名称	ST 表现
MC_Reset_CO	错误复位	MC_Reset_CO(Axis:= , Execute:= , Done=> , Busy=> , Error=> , ErrorID=>);

2） 相关变量

■ 输入输出变量

输入输出变量	名称	数据类型	有效范围	初始值	描述
Axis	轴	AXIS_REF_HC_CO	—	—	CANopen 轴

■ 输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Execute	执行输入	BOOL	[FALSE,TRUE]	FALSE	上升沿触发，由 FALSE->TRUE：开始执行

■ 输出变量

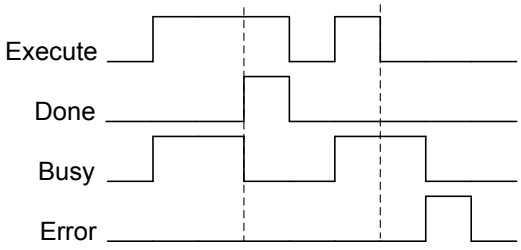
输出变量	名称	数据类型	有效范围	初始值	描述
Done	完成标志	BOOL	[FALSE,TRUE]	FALSE	TRUE：复位完成。
Busy	执行标志	BOOL	[FALSE,TRUE]	FALSE	TRUE：正在执行复位命令。
Error	错误标志	BOOL	[FALSE,TRUE]	FALSE	TRUE：复位命令发生错误。
ErrorID	错误 ID	ERROR_CO	-	NO_ERROR	错误 ID，具体参考 ERROR_CO。

3） 功能说明

■ 复位所有内部轴相关的错误，强制从状态“ErrorStop”到“Standstill”或“PowerOff”的转换。如果受控轴不处于“ErrorStop”状态，执行复位动作，功能块将发出错误。

■ 复位内容包含运动错误、伺服支持复位的错误。

4） 时序图



5） 错误说明

当出现错误时，通过帮助文档找到 COUNTER_ERROR 中 ErrorID 所对应的错误，根据错误内容定位到错误原因。

9 MC_WriteParameter_CO

1） 指令格式

图形表现		
		
指令	名称	ST 表现
MC_WriteParameter_CO	写远端设备参数	MC_WriteParameter_CO(Axis:= , Execute:= , ParameterNumber:= , Value:= , Done=> , Busy=> , Error=> , ErrorID=>);

2） 相关变量

■ 输入输出变量

输入输出变量	名称	数据类型	有效范围	初始值	描述
Axis	轴	AXIS_REF_HC_CO	—	—	CANopen 轴

■ 输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Execute	执行输入	BOOL	[FALSE,TRUE]	FALSE	上升沿触发，由 FALSE->TRUE：开始执行
ParameterNumber	参数	DINT	[0,2^32)	0	参数序号，需要组合运算。
Value	数值	LREAL	(-1.7E 308, 1.7E 308)	0.0	写入值。

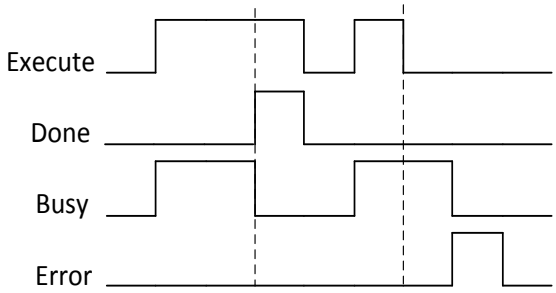
■ 输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Done	完成标志	BOOL	[FALSE,TRUE]	FALSE	TRUE：复位完成。
Busy	执行标志	BOOL	[FALSE,TRUE]	FALSE	TRUE：正在执行命令。
Error	错误标志	BOOL	[FALSE,TRUE]	FALSE	TRUE：功能块发生错误。
ErrorID	错误 ID	ERROR_CO	-	NO_ERROR	错误 ID，具体参考 ERROR_CO。

3） 功能说明。

■ 写对象字典数据，基于 CANopen 的参数，轴特定参数号由对象字典长度 16 # UV（1 字节），对象字典索引 16 # ABCD（2 字节）和子索引 16 # EF（1 字节），共同组合成 -16 # UVABCDEF 模式。示例：写对象索引值 16 # 607C，子索引值 16#00，数据长度值 16#04，则 ParameterNumber: = -16 # 04607C00。

4） 时序图



5） 错误说明

当出现错误时，通过帮助文档找到 COUNTER_ERROR 中 ErrorID 所对应的错误，根据错误内容定位到错误原因。

10 MC_ReadParameter_CO

1） 1 指令格式

图形表现		
		
指令	名称	ST 表现
MC_ReadParameter_CO	读远端设备参数	MC_ReadParameter_CO(Axis:= , Enable:= , ParameterNumber:= , Valid=> , Busy=> , Error=> , ErrorID=> , Value=>);

2） 相关变量

■ 输入输出变量

输入输出变量	名称	数据类型	有效范围	初始值	描述
Axis	轴	AXIS_REF_HC_CO	—	—	CANopen 轴

■ 输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Execute	执行输入	BOOL	[FALSE,TRUE]	FALSE	上升沿触发，由 FALSE->TRUE：开始执行
ParameterNumber	参数	DINT	[0,2^32)	0	参数序号，需要组合运算。

■ 输出变量

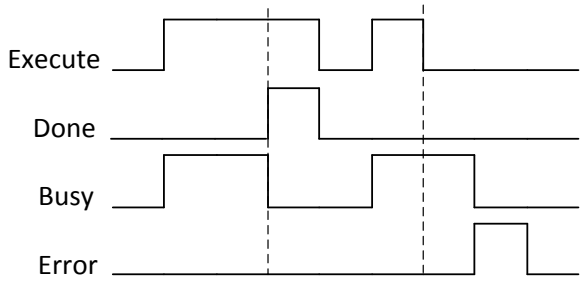
输出变量	名称	数据类型	有效范围	初始值	描述
Done	完成标志	BOOL	[FALSE,TRUE]	FALSE	TRUE：复位完成。
Busy	执行标志	BOOL	[FALSE,TRUE]	FALSE	TRUE：正在执行命令。
Error	错误标志	BOOL	[FALSE,TRUE]	FALSE	TRUE：功能块发生错误。
ErrorID	错误 ID	ERROR_CO	-	NO_ERROR	错误 ID，具体参考 ERROR_CO。

输出变量	名称	数据类型	有效范围	初始值	描述
Value	数值	LREAL	(-1.7E 308, 1.7E 308)	0.0	读取值。

3） 功能说明。

■ 读对象字典数据，基于 CANopen 的参数，轴特定参数号由对象字典长度 16 # UV（1 字节），对象字典索引 16 # ABCD（2 字节）和子索引 16 # EF（1 字节），共同组合成 -16 # UVABCDEF 模式。示例：读对象索引值 16 # 607C，子索引值 16#00，数据长度值 16#04，则 ParameterNumber: = -16 # 04607C00。

4） 时序图

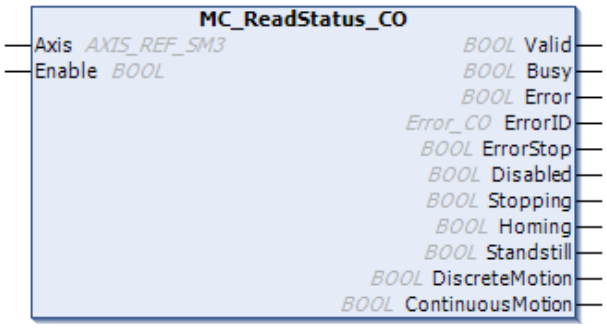


5） 错误说明

当出现错误时，通过帮助文档找到 COUNTER_ERROR 中 ErrorID 所对应的错误，根据错误内容定位到错误原因。

11 MC_ReadStatus_CO

1） 指令格式

图形表现


指令	名称	ST 表现
MC_ReadStatus_CO	当前轴的运动状态	MC_ReadStatus_CO(Axis:= , Enable:= , Valid=> , Busy=> , Error=> , ErrorID=> , ErrorStop=> , Disabled=> , Stopping=> , Homing=> , Standstill=> , DiscreteMotion=> , ContinuousMotion=>);

2） 相关变量

■ 输入输出变量

输入输出变量	名称	数据类型	有效范围	初始值	描述
Axis	轴	AXIS_REF_HC_CO	—	—	CANopen 轴

■ 输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Enable	执行输入	BOOL	[FALSE,TRUE]	FALSE	电平沿触发，TRUE：读取状态。

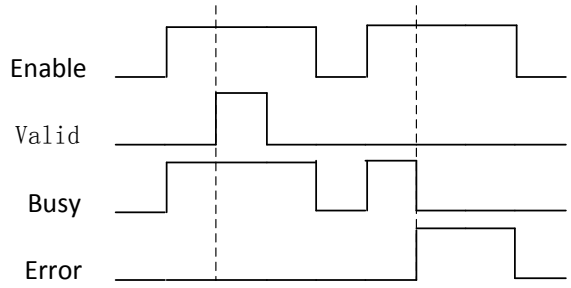
■ 输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Valid	读取状态是否有效标志	BOOL	[FALSE,TRUE]	FALSE	TRUE：复位完成。
Busy	执行标志	BOOL	[FALSE,TRUE]	FALSE	TRUE：正在执行命令。
Error	错误标志	BOOL	[FALSE,TRUE]	FALSE	TRUE：功能块发生错误。
ErrorID	错误 ID	ERROR_CO	-	NO_ERROR	错误 ID，具体参考 ERROR_CO。
ErrorStop	错误停止	BOOL	[FALSE,TRUE]	FALSE	TRUE：轴发生故障。
Disabled	未使能	BOOL	[FALSE,TRUE]	FALSE	TRUE：轴未使能。
Stopping	停止中	BOOL	[FALSE,TRUE]	FALSE	TRUE：轴正在执行停止命令。
Homing	回零中	BOOL	[FALSE,TRUE]	FALSE	TRUE：轴正在执行回零。
Standstill	电机使能	BOOL	[FALSE,TRUE]	FALSE	TRUE：轴已使能。
DiscreteMotion	离散运动中	BOOL	[FALSE,TRUE]	FALSE	TRUE：轴正在执行点位运动。
ContinuousMotion	连续运动中	BOOL	[FALSE,TRUE]	FALSE	TRUE：轴正在执行速度命令。

3） 功能说明。

■ 根据 PLCopen 状态机标准，将受控轴的运动状态以布尔类型的变量显示。

4） 时序图

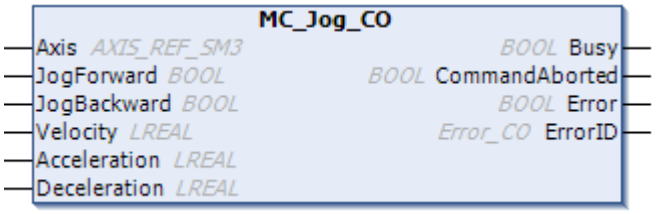


5） 错误说明

当出现错误时，通过帮助文档找到 COUNTER_ERROR 中 ErrorID 所对应的错误，根据错误内容定位到错误原因。

12 MC_Jog_CO

1） 指令格式

图形表现


指令	名称	ST 表现
MC_Jog_CO	读远端设备参数	MC_Jog_CO(Axis:= , JogForward:= , JogBackward:= , Velocity:= , Acceleration:= , Deceleration:= , Busy=> , CommandAborted=> , Error=> , ErrorID=>);

2） 相关变量

■ 输入输出变量

输入输出变量	名称	数据类型	有效范围	初始值	描述
Axis	轴	AXIS_REF_HC_CO	—	—	CANopen 轴

■ 输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
JogForward	正向点动	BOOL	[FALSE,TRUE]	FALSE	点动正向运动。
JogBackword	反向点动	BOOL	[FALSE,TRUE]	FALSE	点动负向运动。
Velocity	速度	LREAL	(0, 1.7E 308)	0.0	点动速度。
Acceleration	加速度	LREAL	(0, 1.7E 308)	0.0	加速度。
Deceleration	减速度	LREAL	(0, 1.7E 308)	0.0	减速度。

■ 输出变量

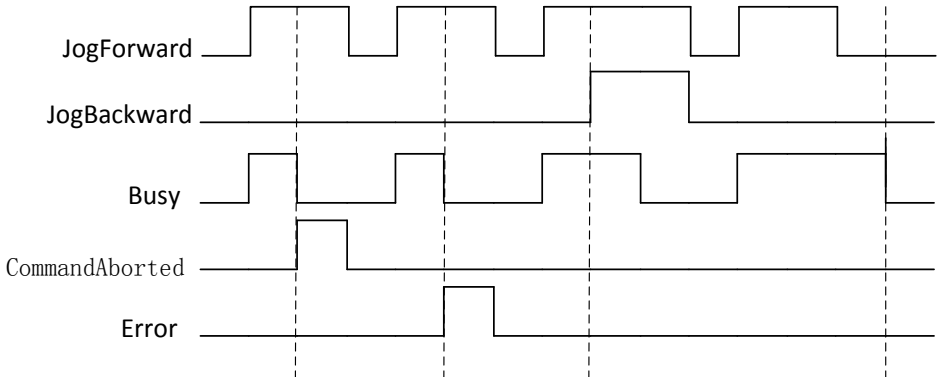
输出变量	名称	数据类型	有效范围	初始值	描述
Busy	执行标志	BOOL	[FALSE,TRUE]	FALSE	TRUE：正在执行命令。
CommandAborted	中止标志	BOOL	[FALSE,TRUE]	FALSE	TRUE：运动被中止。
Error	错误标志	BOOL	[FALSE,TRUE]	FALSE	TRUE：功能块发生错误。
ErrorID	错误 ID	ERROR_CO	-	NO_ERROR	错误 ID，具体参考 ERROR_CO。

3） 功能说明。

■ 本功能块主要实现轴的点动控制，实现点动正向、点动负向运动。

■ JogBackword、JogForward 同时置 TRUE, 运动将会停止，Velocity、Acceleration、Deceleration 设置值必须都大于 0。

4） 时序图



5） 错误说明

当出现错误时，通过帮助文档找到 COUNTER_ERROR 中 ErrorID 所对应的错误，根据错误内容定位到错误原因。

6.3.3 CANopen 402 参数设置

组态是汇川 Inoproshop 编程软件中重要一部分，Inoproshop 目前仅支持 IS620P-CO 型号，如需使用其他厂商的驱动器，必须先需要导入其他厂商 EDS 文件。并确认其他厂商驱动器程序是否严格按照 CANopen 通讯标志和 CIA402 标准的设计。

调用 CANopen402 功能块之前，必须完成正确的参数设置。详细如下：

1 主站配置

主站主要配置有通讯波特率、同步模式、同步时间、心跳、心跳间隔时间。

- 通讯波特率：波特率越高，通讯效率越高，但是波特率越高，通讯距离越短。一般情况下配置波特率为 500k。
- 同步模式：必须勾选周期同步功能，否则功能块无法工作。
- 同步周期时间：在只有 CANopen 轴的功能中，推荐配置是 4ms：3 个从站 , 从站 PDO 配置接收、发送都少于 8 个字节。这个时间建议与程序中 CANopen 任务扫描周期时间相同。
- 心跳：心跳是主站每个心跳时间发送心跳帧，用于从站监控主站是否掉线。此功能必须配合从站同时使用，有些从站未设计心跳检查功能，默认不需要心跳。
- 心跳时间：主站每隔此设置时间长度，给从站发送心跳帧，默认 300ms，在勾选心跳条件下生效。

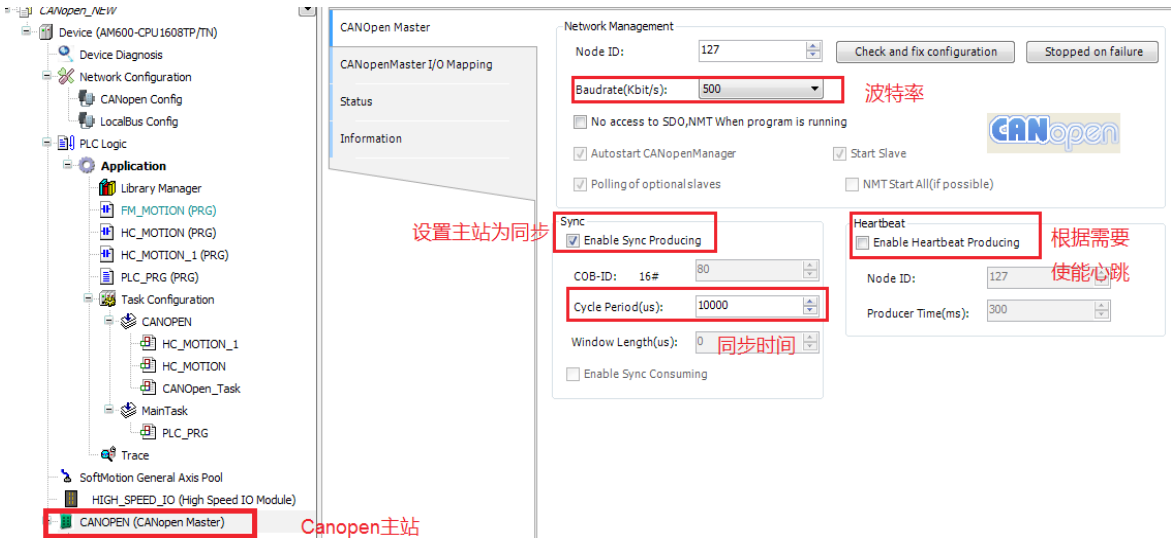


图 6-29 CANopen 主站配置

2 从站配置

从站主要配置节点 ID、心跳、心跳时间、PDO 同步方式、同步对象字典配置。具体配置参考图 6-30、图 6-31、图 6-32。

- 节点 ID：又称为站号，从站通讯的基本参数，站号与实际的物理站号保持一致。
- 心跳：从站向指定站号发送心跳帧，主要用于其他站点监控本站点的通讯状态，默认勾选。
- 心跳时间：从站每隔此设置时间长度，给指定站号发送心跳帧，默认 1000ms，在勾选心跳下生效。
- PDO 同步方式：默认为异步方式，需要更改为同步周期模式。
- PDO 对象字典配置：PDO 对象字典保证从站与主站数据每个总线周期交互一次，数量越多，主站与从站状态交互越高效，但是 PDO 越多会导致总线负载越大，可能导致总线数据传输滞后、严重情况下会导致掉线。对于轴控设备，根据应用经验，发送和接收 PDO 必须配置项有 6040，6041，6060，6061，607A，6064（6063），60FF，606C，6081。选配项为 6083，6084，607C，6098。

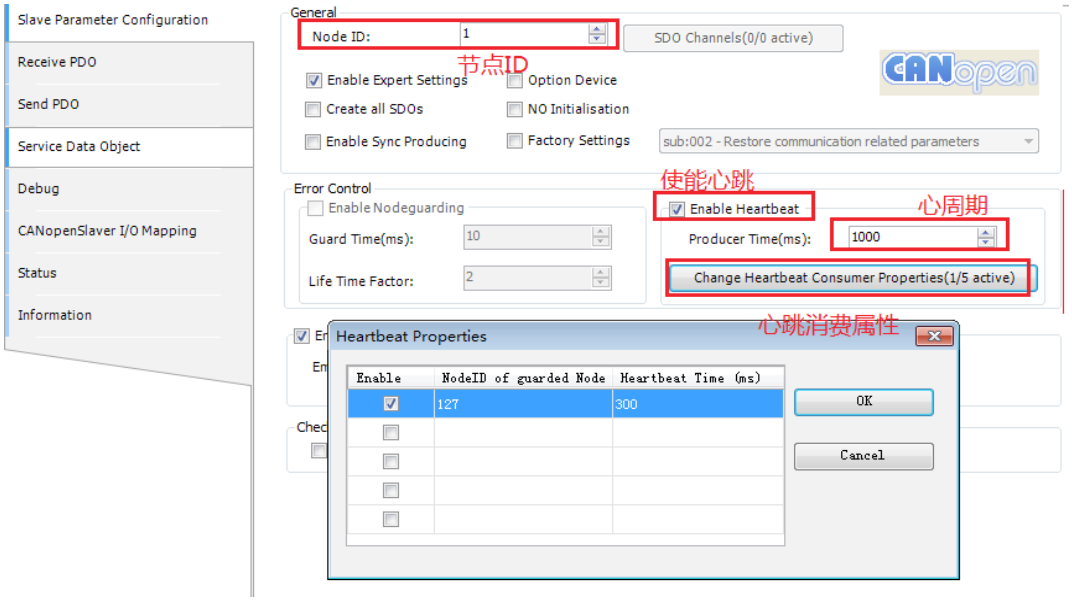


图 6-30 CANopen 从站参数配置

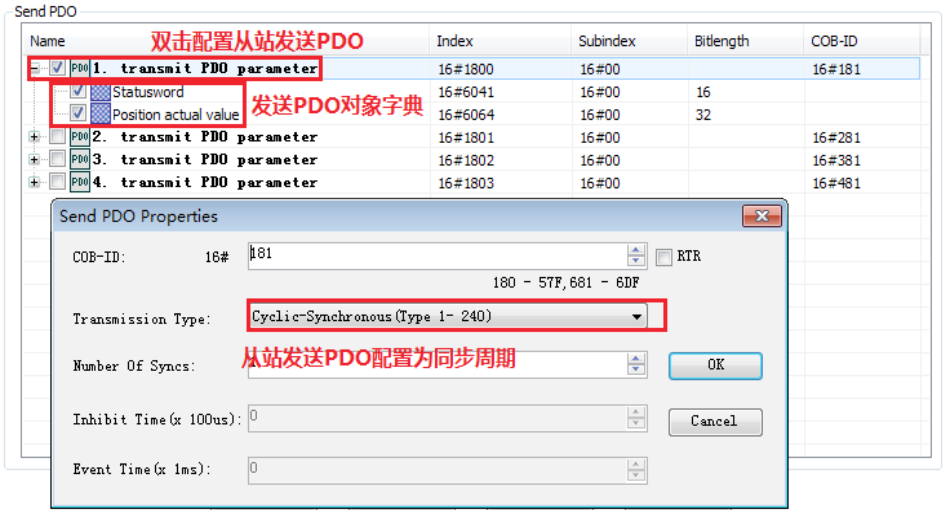


图 6-31 CANopen 从站发送 PDO 配置

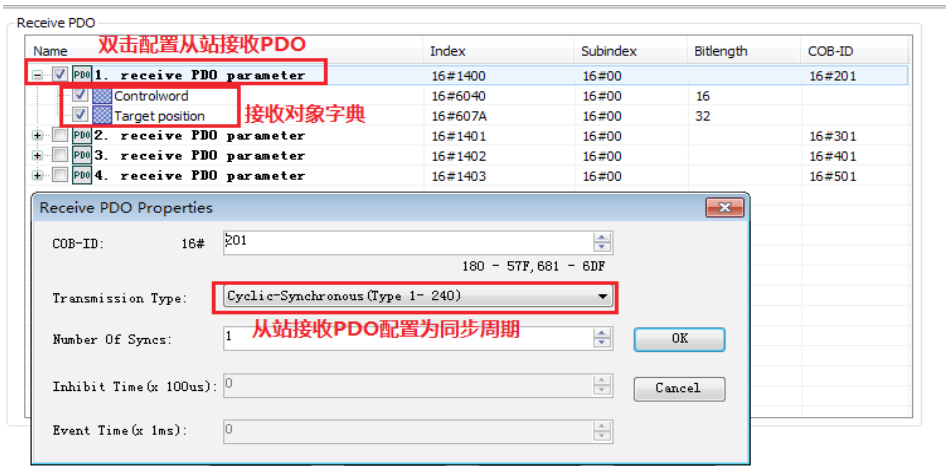


图 6-32 CANopen 从站接收 PDO 配置

3 轴配置

图 6-33 CANopen 轴配置

6.3.4 CANopen 402 错误诊断

在线诊断

显示功能块错误或者系统错误最近一次错误，能快速诊断错误类型。

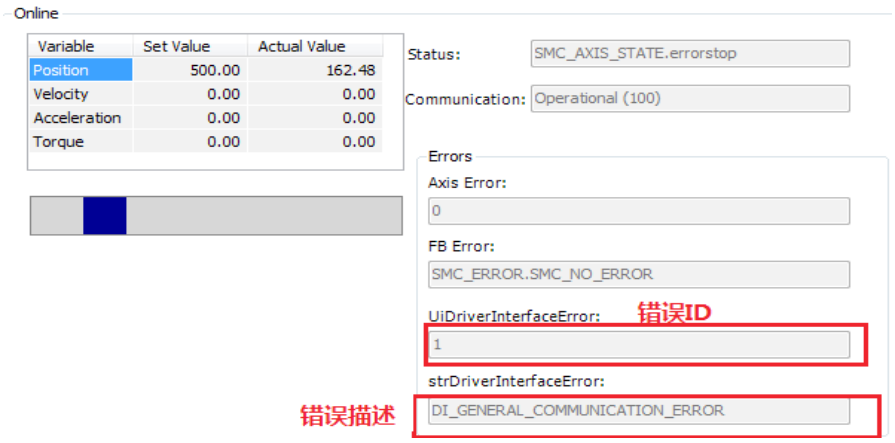


图 6-34 CANopen 轴在线诊断

错误序号、描述对照表如下：

错误 ID	枚举值	描述
0	NO_ERROR	没有错误
1	DI_GENERAL_COMMUNICATION_ERROR	轴通讯错误
2	DI_AXIS_ERROR	驱动轴报错
21	WRONG_OPMODE	错误的操作模式
33	AXIS_IN_ERRORSTOP	轴在 errorstop 状态
34	AXIS_NOT_READY_FOR_MOTION	轴当前状态错误，不能进行运动
35	MA_MR_MODULO_ACT_POS_NOT_MAPPED	保留
36	MV_INVALID_VELACCDEC_VALUES	不合理的速度或者加减速度
80	RAG_ERROR_DURING_STARTUP	保留
81	RAG_ERROR_WRITING_COMSTATE	保留
82	RAG_ERROR_READING_COMSTATE	保留
90	CGR_ZERO_VALUES	保留
91	CGR_AXIS_POWERED	保留
93	CGR_MODULOPERIOD_NOT_INTEGRAL	保留
94	CGR_MOVEMENTTYPE_INVALID	保留
95	CGR_MODULOPERIOD_NON_POSITIVE	保留
96	CGR_MODULOPERIOD_TOO_SMALL	保留
97	CGR_MODULOPERIOD_TOO_LARGE	保留
120	R_NO_ERROR_TO_RESET	没有错误需要复位
121	R_DRIVE_DOESNT_ANSWER	轴没有应答
122	R_ERROR_NOT_RESETTABLE	错误不能被复位
123	R_DRIVE_DOESNT_ANSWER_IN_TIME	保留
130	RP_PARAM_UNKNOWN	读参数错误
131	RP_REQUESTING_ERROR	通讯请求错误
132	RP_RCV_PARAM_CONVERSION_ERROR	保留
133	RP_LOCAL_PARAM_NOT_DONE_IMMEDIATELY	保留

错误 ID	枚举值	描述
134	RP_CANNOT_SEND_MSG	不能发送消息
140	WP_PARAM_INVALID	写参数错误
141	WP_SENDING_ERROR	通讯发送错误
142	WP_TMT_PARAM_CONVERSION_ERROR	保留
143	WP_LOCAL_PARAM_NOT_DONE_IMMEDIATELY	保留
144	WP_CANNOT_SEND_MSG	不能发送消息
170	H_AXIS_WASNT_STANDSTILL	回零需要轴在 standstill 状态
183	MS_AXIS_IN_ERRORSTOP	mc_stop 功能块不能在 errorstop
184	MS_AXIS_IN_STOPPING	mc_stop 功能块不能在 stopping
10000	TIMEOUT_CHANGING_OPMODE	切换模式超时
10001	INTERNAL_UNKNOWN_CMD	保留
10002	CANNOT_START_MOVEMENT	保留
10003	CANNOT_START_HOMING	保留
10004	STOP_ALREADY_ACTIVE	保留
10005	POWER_ALREADY_ACTIVE	保留
10006	SMC_DI_VOLTAGE_DISABLED	轴运动过程断开使能

日志诊断

日志能存储一段时间内系统程序运行的错误信息，用户可根据日志定位程序发生错误的过程，日志记录时间发生的日期、时间、驱动轴、发生错误的功能块、错误 id。

下图所示: 错误发生信息记录发生日期: 2017.9.20，时间: 09:09:24，驱动器站号: 4，功能块: Axis_CO，错误信息: “DI_GENERAL_COMMUNICATION_ERROR”。通过查询上面的错误表，轴错误为通讯错误。

通过记录可以定位轴先出现通讯故障，然后绝对定位功能块报错。

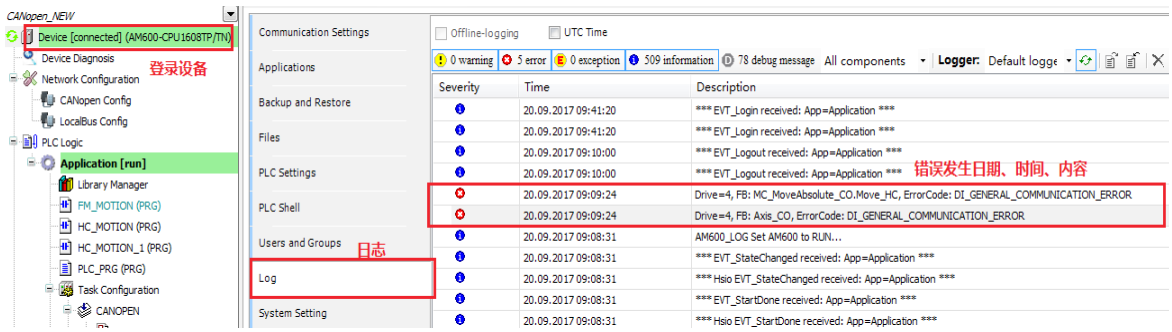


图 6-35 CANopen 轴日志诊断

NOTE

◆ 中型 PLC 配合第三方 CANopen 设备的应用情况，请联系厂家技术支持。

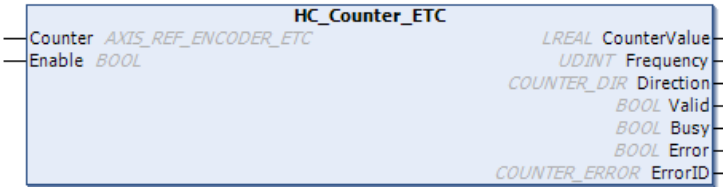
6.4 EtherCAT 远程计数

功能概述：带有高速输入口的 EtherCAT 从站模块，包含计数、采样、比较输出、探针等功能。采样频率、输出频率最大可支持 200K Hz；高速中断最大可支持 3K Hz。

对应库名：IoDrvEtherCATEncoder(适用版本 V1.0.5.0 和以上)。

6.4.1 HC_Counter_ETC

1） 指令格式

图形表现		
		
指令	名称	ST 表现
HC_Counter_ETC	计数器使能	HC_Counter_ETC(Counter:= , Enable:= , CounterValue=> , Frequency=> , Direction=> , Valid=> , Busy=> , Error=> , ErrorID=>);

2） 相关变量

■ 输入输出变量

输入输出变量	名称	数据类型	有效范围	初始值	描述
Counter	远程计数器	AXIS_REF_ENCODER_ETC	—	—	AXIS_REF_ENCODER_ETC 计数器类型变量

■ 输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Enable	使能	BOOL	[FALSE,TRUE]	FALSE	电平触发，TRUE：使能计数器。

■ 输出变量

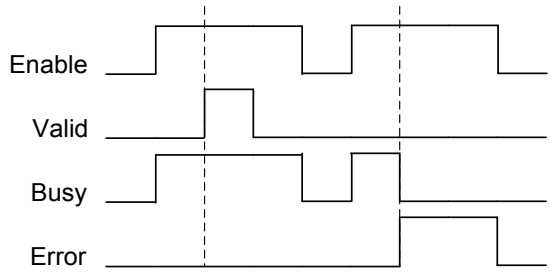
输出变量	名称	数据类型	有效范围	初始值	描述
CounterValue	计数值	LREAL	(-1.7E 308, 1.7E 308)	0.0	当前计数值，由计数脉冲换算而来。
Frequency	频率	UDINT	[0,2^32)	0	采集的脉冲的脉冲频率。
Direction	方向	COUNTER_DIR	[-1,1]	-1	计数的方向。
Valid	使能状态	BOOL	[FALSE,TRUE]	FALSE	计数使能的状态。
Busy	执行标志	BOOL	[FALSE,TRUE]	FALSE	执行计数使能命令
Error	错误标志	BOOL	[FALSE,TRUE]	FALSE	TRUE: 功能块内部发生错误

输出变量	名称	数据类型	有效范围	初始值	描述
ErrorID	错误 ID	COUNTER_ERROR	-	COUNTER_NO_ERROR	错误 ID，具体参考 COUNTER_ERROR

3） 功能说明

■ 本功能块主要实现远程计数器使能、计数、测频、计数方向输出功能。

4） 时序图



5） 错误说明

当出现错误时，通过帮助文档找到 COUNTER_ERROR 中 ErrorID 所对应的错误，根据错误定位到错误原因。

错误 ID	枚举值	描述
0x00	COUNTER_NO_ERROR	无错误
0x01	COUNTER_ERROR_ECT_COMMUNICATION	ETC 通讯故障
0x02	COUNTER_ERROR_PV_PRESETVALUE_EXCEED	预置值超过限制
0x03	COUNTER_ERROR_CP_COMPARE_VAULE_INVALID	比较值设置不合理
0x04	COUNTER_ERROR_FILT_PARAM_EXCEED	滤波系数超范围
0x10	COUNTER_ERROR_PV_DI_CONFIGURE_ERROR	DI 端子未配置预置功能
0x11		计数器清零，暂时未用
0x12	COUNTER_ERROR_TP_DI_CONFIGURE_ERROR	DI 端子未配置探针功能
0x30	COUNTER_ERROR_CP_DO_CONFIGURE_ERROR	DO 输出端子未配置比较一致输出功能
0x50	COUNTER_ERROR_INPUT_PUSLE_FREQUENCY_TOO_HIGH	输入频率大于 202k
0x51	COUNTER_ERROR_OVERFLOW	计数值上溢
0x52	COUNTER_ERROR_UNDERFLOW	计数值下溢
0x1001	COUNTER_ERROR_PV_TYPE_INVALID	预置类型超出范围
0x1002	COUNTER_ERROR_DISABLE	计数器未使能
0x1003	COUNTER_ERROR_TP_PARAM_INVALID	探针参数不合理
0x1004	COUNTER_ERROR_CP_PARAM_INVALID	比较通道参数错误
0x1005	COUNTER_ERROR_R_NO_ERROR_CLEAR	未有错误清除
0x1006	COUNTER_ERROR_R_ERROR_CANNOT_RESET	错误不能清除
0x1100	COUNTER_ERROR_PARAM_NO_PDO_MAPPING	指令未配置相应 PDO

6.4.2 HC_SetCompare_ETC

1） 指令格式

图形表现


指令	名称	ST 表现
HC_SetCompare_ETC	远程计数器比较一致输出功能	HC_SetCompare_ETC(Counter:= , Execute:= , Abort:= , Channel:= , CompareValue:= , ImRefreshCycle:= , Done=> , Busy=> , CommandAborted=> , Error=> , ErrorID=>);

2） 相关变量

■ 输入输出变量

输入输出变量	名称	数据类型	有效范围	初始值	描述
Counter	计数器	AXIS_REF_ENCODER_ETC	—	—	AXIS_REF_ENCODER_ETC 计数器类型变量。

■ 输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Execute	使能	BOOL	[FALSE,TRUE]	FALSE	边沿触发，执行比较一致功能的开关。
Abort	中止	BOOL	[FALSE,TRUE]	FALSE	电平触发，TRUE：中止功能块执行。
Channel	比较通道	BYTE	[1,2]	1	计数器比较通道选择，共有 2 个通道。
CompareValue	比较值	LREAL	(-1.7E 308, 1.7E 308)	0.0	比较值，浮点类型，单位：unit。
ImRefreshCycle	输出保持时间	UINT	(0,2^32)	1000	打开输出口保持时间，最小单位为 100us。 默认值：1000，1000*100us=100ms。

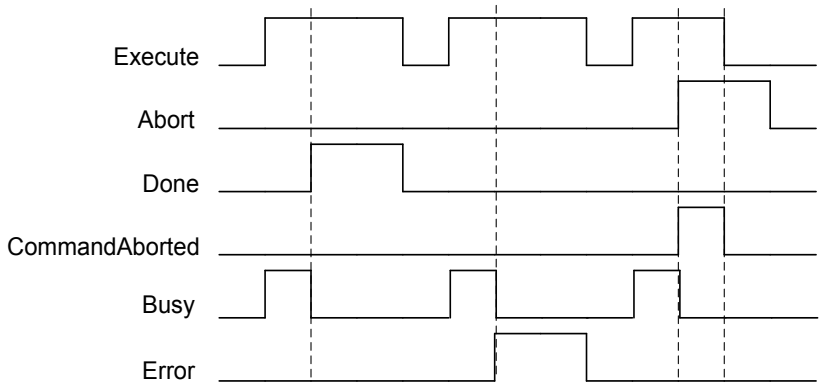
■ 输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Done	比较一致完成状态	BOOL	[FALSE,TRUE]	FALSE	计数脉冲到达设置比较值，并且输出保持时间结束。
CommandAborted	中止状态	BOOL	[FALSE,TRUE]	FALSE	中止状态，在 Abort=TRUE 置位。
Busy	执行标志	BOOL	[FALSE,TRUE]	FALSE	执行比较一致功能。
Error	错误标志	BOOL	[FALSE,TRUE]	FALSE	TRUE: 功能块内部发生错误。
ErrorID	错误 ID	COUNTER_ERROR	-	COUNTER_NO_ERROR	错误 ID，具体参考 COUNTER_ERROR。

3） 功能说明

- 本功能块实现远程计数器当前计数值与设置比较值相等时，触发一个硬件高速输出口，并保持一段时间。
- 比较一致输出功能属于单次触发，当 Done=TRUE 后，比较一致功能完成，如果需要继续使用比较功能，需要再次触发 Execute。
- 确认开发软件中配置计数器比较一致功能的输出点属性。

4） 时序图



5） 错误说明

当出现错误时，通过帮助文档找到 COUNTER_ERROR 中 ErrorID 所对应的错误，根据错误内容定位到错误原因。

6.4.3 HC_Presetvalue_ETC

1） 指令格式

图形表现		
		
指令	名称	ST 表现
HC_PresetValue_ETC	计数器预置	HC_PresetValue_ETC(Counter:= , Execute:= , Abort:= , TriggerType:= , PresetValue:= , Done=> , Busy=> , CommandAborted=> , Error=> , ErrorID=>);

2） 相关变量

- 输入输出变量

输入输出变量	名称	数据类型	有效范围	初始值	描述
Counter	远程计数器	AXIS_REF_ENCODER_ETC	—	—	AXIS_REF_ENCODER_ETC 计数器类型变量。

■ 输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Execute	使能	BOOL	[FALSE,TRUE]	FALSE	边沿触发，执行预置功能的开关。
Abort	中止	BOOL	[FALSE,TRUE]	FALSE	电平触发，TRUE：中止功能块执行。
TriggerType	触发类型	BYTE	[1,7]	1	内部触发：0x01； DI 触发：0x02； 内部、DI 触发：0x03； Z 相触发：0x04； 内部、Z 相触发：0x05； DI、Z 相触发：0x06； 内部、DI、Z 相触发：0x07。
PresetValue	预置值	LREAL	(-1.7E 308, 1.7E 308)	0.0	计数器的预置值。

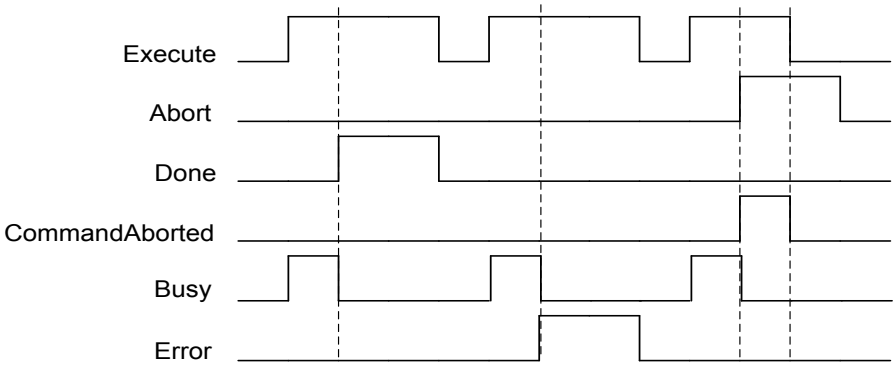
■ 输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Done	预置成功状态	BOOL	[FALSE,TRUE]	FALSE	计数脉冲到达设置比较值，并且输出保持时间结束。
CommandAborted	中止状态	BOOL	[FALSE,TRUE]	FALSE	中止状态，在 Abort=TRUE 置位。
Busy	执行标志	BOOL	[FALSE,TRUE]	FALSE	执行计预置命令。
Error	错误标志	BOOL	[FALSE,TRUE]	FALSE	TRUE: 功能块内部发生错误。
ErrorID	错误 ID	COUNTER_ERROR	-	COUNTER_NO_ERROR	错误 ID，具体参考 COUNTER_ERROR。

3） 功能说明

■ 本功能块主要实现远程计数器的预置功能，预置方法总共 7 种，当选择两种组合以上的触发类型，当组合中任意一种条件满足，计数器都会执行预置，预置完成后输出 Done 信号。预置功能块的执行属于上升沿单次触发有效。

4） 时序图

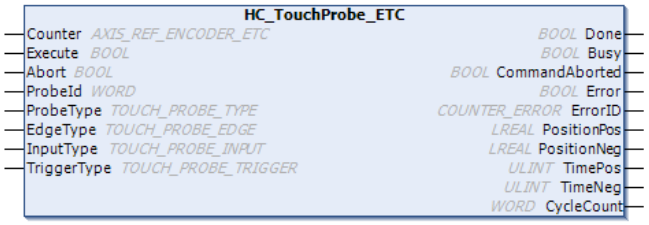


5） 错误说明

当出现错误时，通过帮助文档找到 COUNTER_ERROR 中 ErrorID 所对应的错误，根据错误定位到错误原因。

6.4.4 HC_TouchProbe_ETC

1） 指令格式

图形表现


指令	名称	ST 表现
HC_TouchProbe_ETC	计数器探针	HC_TouchProbe_ETC(Counter:= , Execute:= , Abort:= , ProbeId:= , ProbeType:= , EdgeType:= , InputType:= , TriggerType:= , Done=> , Busy=> , CommandAborted=> , Error=> , ErrorID=> , PositionPos=> , PositionNeg=> , TimePos=> , TimeNeg=> , CycleCount=>);

2） 相关变量

■ 输入输出变量

输入输出变量	名称	数据类型	有效范围	初始值	描述
Counter	远程计数器	AXIS_REF_ENCODER_ETC	—	—	AXIS_REF_ENCODER_ETC 计数器类型变量。

■ 输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Execute	使能	BOOL	[FALSE,TRUE]	FALSE	边沿触发，执行预置功能的开关。
Abort	中止	BOOL	[FALSE,TRUE]	FALSE	电平触发，TRUE：中止功能块执行。
ProbeId	探针 id	WORD	[1,2]	1	探针 ID, 仅有两路探针
ProbeType	探针类型	TOUCH_PROBE_TYPE	-		探针类型：时间、位置

输入变量	名称	数据类型	有效范围	初始值	描述
EdgeType	边沿类型	TOUCH_PROBE_EDGE	[1，3]	RisingEdge	1： 上升沿 2： 下降沿 3：上升和下降沿
InputType	硬件触发类型	TOUCH_PROBE_INPUT	[0，1]	DigtalInput	0：探针输入端子为外部输入端子 1：探针输入端子为 Z 相
TriggerType	触发方式	TOUCH_PROBE_TRIGGER	[0，1]	TrigSingle	0：探针单次触发 1：探针连续触发

■ 输出变量

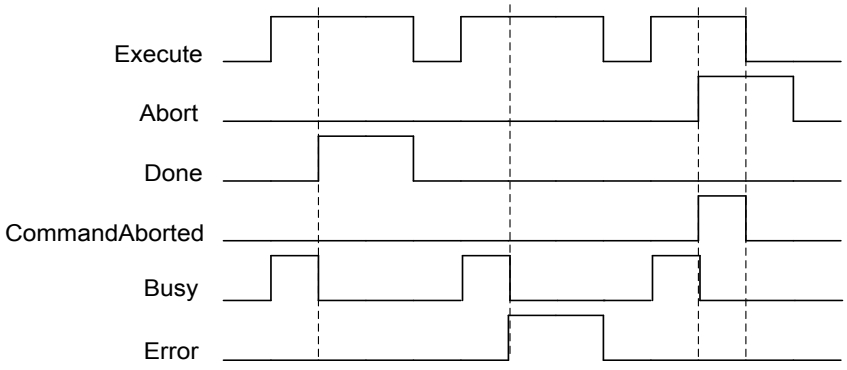
输出变量	名称	数据类型	有效范围	初始值	描述
Done	探针执行状态	BOOL	[FALSE,TRUE]	FALSE	探针功能执行完毕。
CommandAborted	中止状态	BOOL	[FALSE,TRUE]	FALSE	中止状态，在 Abort=TRUE 置位。
Busy	执行标志	BOOL	[FALSE,TRUE]	FALSE	执行计数探针功能。
Error	错误标志	BOOL	[FALSE,TRUE]	FALSE	TRUE: 功能块内部发生错误。
ErrorID	错误 ID	COUNTER_ERROR	-	COUNTER_NO_ERROR	错误 ID，具体参考 COUNTER_ERROR。
PositionPos	上升沿锁存位置	LREAL	(-1.7E 308, 1.7E 308)	0.0	上升沿锁存位置，单位 :unit
PositionNeg	下降沿锁存位置	LREAL	(-1.7E 308, 1.7E 308)	0.0	下降沿锁存位置，单位 :unit
TimePos	上升沿锁存时间	LINT	[0,2^63)	0	上升沿锁存时间，单位 ns
TimeNeg	下降沿锁存时间	LINT	[0,2^63)	0	下降沿锁存时间，单位 ns
CycleCount	探针检查计数值	WORD	[0,2]	0	连续触发模式下，探针有效计数值 , 探针触发一次 CycleCount 累加一次，值 0-2 循环变化。

3） 功能说明

■ 本功能块主要实现远程计数器探针功能，支持两路探针，都支持单次触发、连续触发。

■ 位置探针和时间探针可以同时使用，在使用探针功能时注意计数器的探针属性配置和 PDO 映射配置，如果配置不合理，触发功能块 Execute 时，功能块 Error 置 TRUE，ErrorID 查询 COUNTER_ERROR 错误表。

4） 时序图

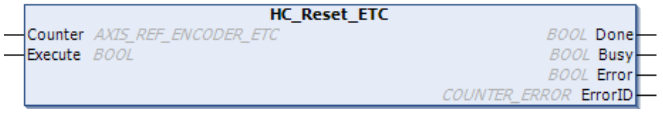


5） 错误说明

当出现错误时，通过帮助文档找到 COUNTER_ERROR 中 ErrorID 所对应的错误，根据错误定位到错误原因。

6.4.5 HC_Reset_ETC

1） 指令格式

图形表现		
		
指令	名称	ST 表现
HC_Reset_ETC	计数器错误复位	HC_Reset_ETC(Counter:= , Execute:= , Done=> , Busy=> , Error=> , ErrorID=>);;

2） 相关变量

■ 输入输出变量

输入输出变量	名称	数据类型	有效范围	初始值	描述
Counter	远程计数器	AXIS_REF_ENCODER_ETC	—	—	AXIS_REF_ENCODER_ETC 计数器类型变量。

■ 输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Execute	使能	BOOL	[FALSE,TRUE]	FALSE	边沿触发，计数器复位功能开关。

■ 输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Done	复位完成状态	BOOL	[FALSE,TRUE]	FALSE	复位功能执行完毕。
Busy	执行标志	BOOL	[FALSE,TRUE]	FALSE	TRUE= 正在执行复位。
Error	错误标志	BOOL	[FALSE,TRUE]	FALSE	TRUE: 功能块内部发生错误。
ErrorID	错误 ID	COUNTER_ERROR	-	COUNTER_NO_ERROR	错误 ID，具体参考 COUNTER_ERROR。

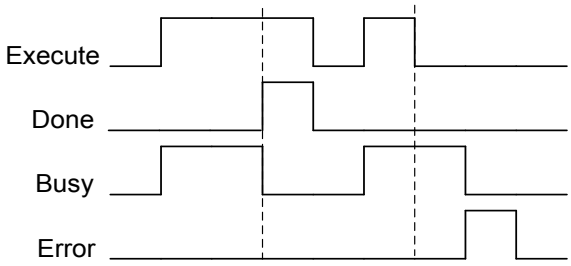
3） 功能说明

■ 本功能块主要实现远程计数器故障、错误复位功能。

■ 计数器发生故障，必须用 HC_Reset_ETC 清除，如果功能块无法清除，需要断电重启计数模块。

■ 功能块发生错误，只要将 Execute 由 TRUE 变成 FALSE，就能清除功能块错误标志。

4） 时序图



5） 错误说明

当出现错误时，通过帮助文档找到 COUNTER_ERROR 中 ErrorID 所对应的错误，根据错误定位到错误原因。

6.5 工艺库

功能概述：基于 PLCopen 功能块的基础上开发的集成特殊工艺的功能块，包含 Modbus 浮点重组、异性双边抛光、描点电子凸轮、智能温度控制、自整定 PID、轮切、飞剪等功能。

如需了解更多信息，请联系汇川技术或访问官网：http://www.inovance.com/。

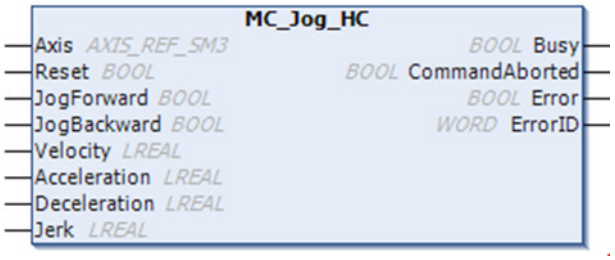
6.6 其他

基于产品优化考虑，汇川为用户另外提供了较为实用的功能块，如总线复位、位置保存等。本章节将逐一介绍。

注意：在库管理器中，将 CmpHCBasic.libaray 添加到工程，即可使用本小节描述的所有功能块。

6.6.1 MC_Jog_HC

1） 指令格式

<table>
<tr><th colspan="3">图形表现</th></tr>
<tr><td colspan="3"> 
</td></tr>
<tr><th>指令</th><th>名称</th><th>ST 表现</th></tr>
<tr><td>MC_Jog_HC</td><td>JOG 功能块</td><td>MC_Jog_HC(
Axis:= ,
Reset:= ,
JogForward:= ,
JogBackward:= ,
Velocity:= ,
Acceleration:= ,
Deceleration:= ,
Jerk:= ,
Busy=> ,
CommandAborted=> ,
Error=> ,
ErrorID=>);</td></tr>
</table>

2） 相关变量

■ 输入输出变量

输入输出变量	名称	数据类型	有效范围	初始值	描述
Axis	轴	AXIS_REF_SM3	—	—	AXIS_REF_SM3 类型变量

■ 输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Reset	复位	BOOL	[FALSE,TRUE]	FALSE	边沿触发，当运动功能不能正常使用情况下置 TRUE, 功能块可以继续使用。
JogForward	JOG+	BOOL	[FALSE,TRUE]	FALSE	电平触发，Jog 正向运动
JogBackward	JOG-	BOOL	[FALSE,TRUE]	FALSE	电平触发，Jog 负向运动
Velocity	速度	LREAL	(0, 1.7E 308)	0.0	速度，单位 :unit/s
Acceleration	加速度	LREAL	(0, 1.7E 308)	0.0	加速度，单位 :unit/s^2
Deceleration	减速度	LREAL	(0, 1.7E 308)	0.0	减速度，单位 :unit/s^2
Jerk	加加速度	LREAL	(0, 1.7E 308)	0.0	加加速度，单位 :unit/s^3

■ 输出变量

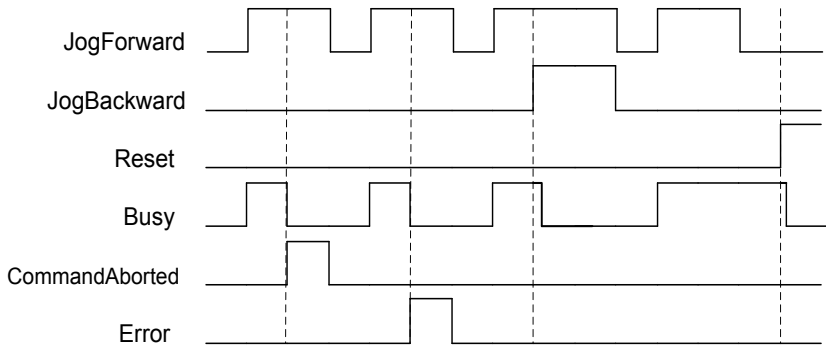
输出变量	名称	数据类型	有效范围	初始值	描述
Busy	执行标志	BOOL	[FALSE,TRUE]	FALSE	在 JogForward 或者 JogBackward 置 TRUE 时 Busy 为 TRUE。反之，轴的速度将按照减速度要求缓慢停止，并且当轴状态机到达 standstill 状态时，Busy 将会被设置为 FALSE
CommandAborted	中止标志	BOOL	[FALSE,TRUE]	FALSE	TRUE: 功能块被另一个功能块终止
Error	错误标志	BOOL	[FALSE,TRUE]	FALSE	TRUE: 功能块内部发生错误
ErrorID	错误 ID	SMC_ERROR	—	NO_ ERROR	错误 ID，具体参考 SMC_ERROR

3） 功能说明

■ 本功能块主要实现控制轴点动正向运动或者负向运动，与 MC_Jog 功能类似。

■ 当减速停止过程遇到硬件限位，轴不能点动情况下，触发功能块输入参数 Reset，轴可以恢复点动。

4） 时序图

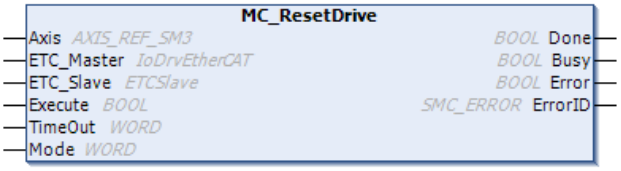


5） 错误说明

当出现错误时，通过帮助文档找到 SMC_ERROR 中 ErrorID 所对应的错误，根据错误定位到错误原因。

6.6.2 MC_ResetDrive

1） 指令格式

图形表现


指令	名称	ST 表现
MC_ResetDrive	单个伺服从站复位	MC_ResetDrive(Axis:= , ETC_Master:= , ETC_Slave:= , Execute:= , TimeOut:= , Mode:= , Done=> , Busy=> , Error=> , ErrorID=>);

2） 相关变量

■ 输入输出变量

输入输出变量	名称	数据类型	有效范围	初始值	描述
Axis	轴	AXIS_REF_SM3	—	—	AXIS_REF_SM3 类型变量
ECT_Master	EtherCAT 主站	IoDrvEtherCAT	—	—	IoDrvEtherCAT 类型变量，EtherCAT 主站名称
ETC_Slave	EtherCAT 从站	ETCSlave	—	—	ETCSlave 类型变量，EtherCAT 从站名称

■ 输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Execute	复位	BOOL	[FALSE,TRUE]	FALSE	边沿触发，上升沿有效。
TimeOut	超时时间	WORD	[0,2^32-1]	10000	通讯状态机切换超时时间，默认 1000 个扫描周期 如果状态机切换超时，状态机会自动回到 Init 状态，重新开始 Init->Pre-op->SafeOP->OP 流程
Mode	复位模式	WORD	[0,1]	1	复位模式，0：快速复位，1：正常复位，复位时间比较长，防止复位过程中出现 Er.15 错误

■ 输出变量

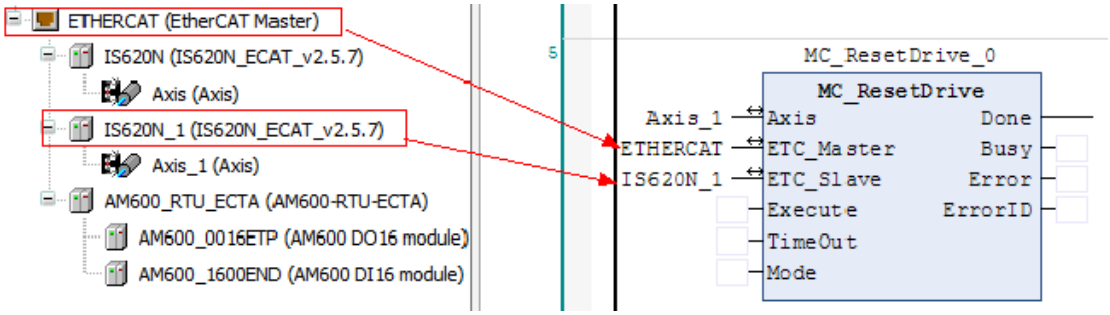
输出变量	名称	数据类型	有效范围	初始值	描述
Done	完成标志	BOOL	[FALSE,TRUE]	FALSE	复位完成标志，TRUE：从站和轴复位成功

输出变量	名称	数据类型	有效范围	初始值	描述
Busy	执行标志	BOOL	[FALSE,TRUE]	FALSE	在 Execute 触发后，Busy 为 TRUE。当这个命令输入为 FALSE，仍会保持 TRUE, 功能块执行错误，或者功能块执行成功
Error	错误标志	BOOL	[FALSE,TRUE]	FALSE	TRUE: 功能块内部发生错误
ErrorID	错误 ID	SMC_ERROR	—	NO_ERROR	错误 ID，具体参考 SMC_ERROR

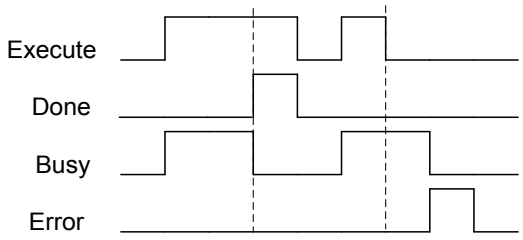
3） 功能说明

- 本功能块主要实现单个伺服从站通讯掉线后能够继续接入网络并正常使用功能，第一个从站除外。
- TimeOut 为扫描周期个数，正在的超时时间应该等于 TimeOut* 每个扫描周期时间 ms。
- Mode = 1：正常复位，复位过程中轴切换到安全模式等待 20s，避开调整伺服的时钟抖动。Mode = 0：快速复位，复位过程中轴切换到安全模式不等待，直接进入 402 状态机复位。
- 如果第一个从站掉线了，此功能块不适用，需要使用执行全局功能块 ETHERCAT，并上升沿触发 ETHERCAT.xReStart，重启整个网络。

示例：复位 Axis_1 所在的从站 IS620N_1, 主站 ETHERCAT。



4） 时序图

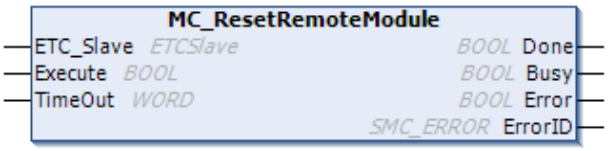


5） 错误说明

当出现错误时，通过帮助文档找到 SMC_ERROR 中 ErrorID 所对应的错误，根据错误定位到错误原因。

6.6.3 MC_ResetRemoteModule

1） 指令格式

图形表现


指令	名称	ST 表现
MC_ResetRemoteModule	单个 ECT 从站复位	MC_ResetRemoteModule(ETC_Slave:= , Execute:= , TimeOut:= , Done=> , Busy=> , Error=> , ErrorID=>);

2） 相关变量

■ 输入输出变量

输入输出变量	名称	数据类型	有效范围	初始值	描述
ETC_Slave	EtherCAT 从站	ETCSlave	—	—	ETCSlave 类型变量，EtherCAT 从站名称

■ 输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Execute	复位	BOOL	[FALSE,TRUE]	FALSE	边沿触发，上升沿有效
TimeOut	超时时间	WORD	[0,2^32-1]	1000	通讯状态机切换超时时间，默认 1000 个扫描周期。 如果状态机切换超时，状态机会自动回到 Init 状态，重新开始 Init->Pre-op->SafeOP->OP 流程

■ 输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Done	完成标志	BOOL	[FALSE,TRUE]	FALSE	复位完成标志，TRUE：从站复位成功
Busy	执行标志	BOOL	[FALSE,TRUE]	FALSE	在 Execute 触发后，Busy 为 TRUE。当 Execute 命令输入为 FALSE，Busy 仍会保持 TRUE, 直到功能块执行错误，或者功能块执行成功时才置 FALSE。
Error	错误标志	BOOL	[FALSE,TRUE]	FALSE	TRUE: 功能块内部发生错误
ErrorID	错误 ID	SMC_ERROR	—	NO_ERROR	错误 ID，具体参考 SMC_ERROR

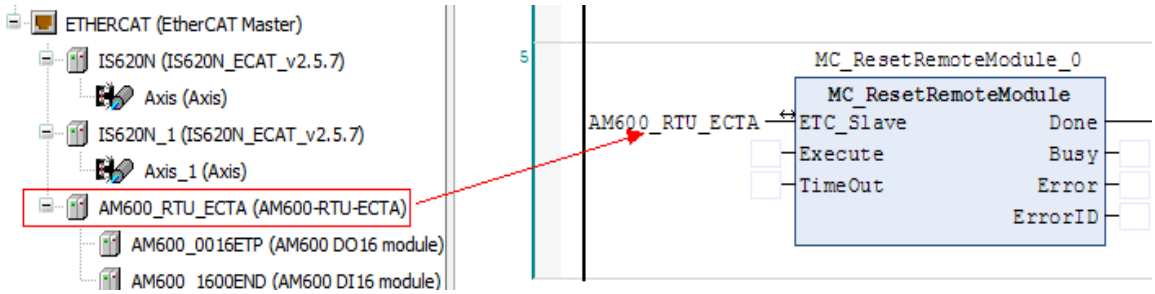
3） 功能说明

■ 本功能块主要实现单个 ECT 从站通讯掉线后能够继续接入网络并正常使用的功能，第一个从站除外。

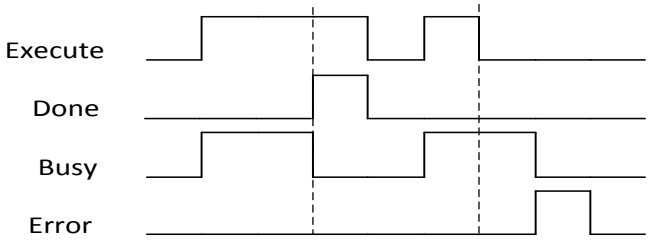
■ TimeOut 为扫描周期个数，正在的超时时间应该等于 TimeOut* 每个扫描周期时间 ms。

■ 如果第一个从站掉线了，此功能块不适用，需要使用执行全局功能块 ETHERCAT，并上升沿触发 ETHERCAT.xReStart，重启整个网络。

示例：复位 ECT 从站 AM600_RTU_ECTA 。



4） 时序图



5） 错误说明

当出现错误时，通过帮助文档找到 SMC_ERROR 中 ErrorID 所对应的错误，根据错误定位到错误原因。

6.6.4 MC_PersistPosition

1） 1 指令格式

图形表现


指令	名称	ST 表现
MC_PersistPosition	掉电保持功能块	MC_PersistPosition(Axis:= , PersistPositionSingleturn_Data:= , bEnable:= , Thresthold:= , bPositionRestored=> , bPositionStored=> , bBoundary=> , bBusy=> , bError=> , eErrorID=>);

2） 相关变量

■ 输入输出变量

输入输出变量	名称	数据类型	有效范围	初始值	描述
Axis	轴	AXIS_REF_SM3	—	—	AXIS_REF_SM3 类型变量
PersistPositionSingleturn_Data	结构体变量	SMC3_PersistPositionSingleturn_Data	—	—	用于存储轴位置数据的结构体变量

■ 输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
bEnable	使能	BOOL	[FALSE,TRUE]	FALSE	电平触发，高电平有效
Thresthold	上下电编码器脉冲允许的门槛值	UDINT	$[0,2^{32}-1]$	16#800000	轴下电后存在位置移动，当移动的距离大于门槛值时，功能块会出现恢复失败报错

■ 输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
bPositionRestored	恢复完成标志	BOOL	[FALSE,TRUE]	FALSE	TRUE：恢复成功
bPositionStored	正在保存标志	BOOL	[FALSE,TRUE]	FALSE	TRUE：数据正在保存
bBoundary	边界标志	BOOL	[FALSE,TRUE]	FALSE	TRUE：轴位置正处于边界位置
Busy	执行标志	BOOL	[FALSE,TRUE]	FALSE	在 bEnable 触发后，Busy 为 TRUE
Error	错误标志	BOOL	[FALSE,TRUE]	FALSE	TRUE: 功能块内部发生错误
eErrorID	错误 ID	MC_ERROR	—	NO_ERROR	错误 ID，具体参考 MC_ERROR

3） 功能说明

■ 本功能块主要实现单个绝对值伺服轴位置上、下位置保存功能。

■ SMC3_PersistPositionSingleturn_Data 为系统的数据结构。

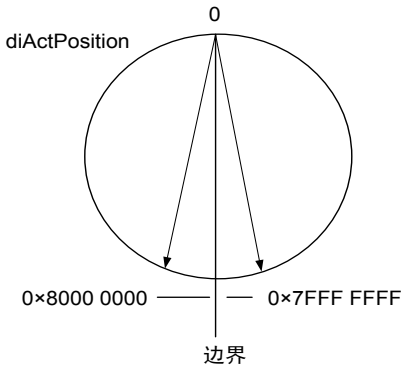
■ Thresthold 值非常重要，PLC 下电时会将对象字典 16#6064（伺服反馈给 PLC 的 32bit 的实际位置）存储到 Flash 中。在 bEnable=TRUE 的条件，系统启动时会从 Flash 读取实际位置，当系统总线通信能进行 PDO 通讯，系统再读取当前伺服轴的实际位置。如果（读取的实际位置 - 当前实际位置）的绝对值 > Thresthold, 系统会认为恢复失败，并将 bError 置 TRUE（即使恢复的数据完全正确）。如果系统下电后会可能移动，可以将 Thresthold 值放大，但最大值不能超过 2^31-1，否则不能判断下电后电机是否经过边界位置。

■ 只有 bPositionRestored 置 TRUE，伺服轴位置数据恢复成功标志，可以通过上升沿检测，绝对值伺服运动之前必须检测用户 bPositionRestored=TRUE, 否则可能引起撞机的风险。

■ 功能块每个周期会检测轴是否停在边界位置。bBoundary=TRUE 表示在边界附近（16#7F000000 < 轴 .dwActPosition< 16#81000000），如果在边界附近掉电，重启 PLC 可能引起位置数据恢复失败，最好避开此范围断电 .。

■ 配合汇川的 IS62N 使用时，伺服需要调整两个参数：伺服模式设置绝对值线性模式，并且屏蔽绝对值位置溢出报警。

■ MC_PersistPosition 必须放在程序扫描最开始的位置，并且 bEnable 初始默认值要设置为 TRUE。保证系统上电后就可以扫描此功能块。

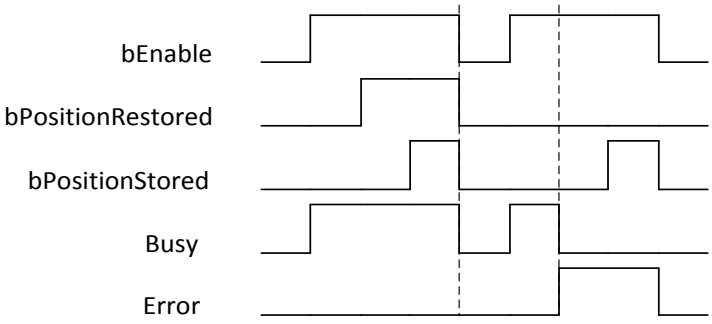


SMC3_PersistPositionSingleturn_Data 具体描述如下：

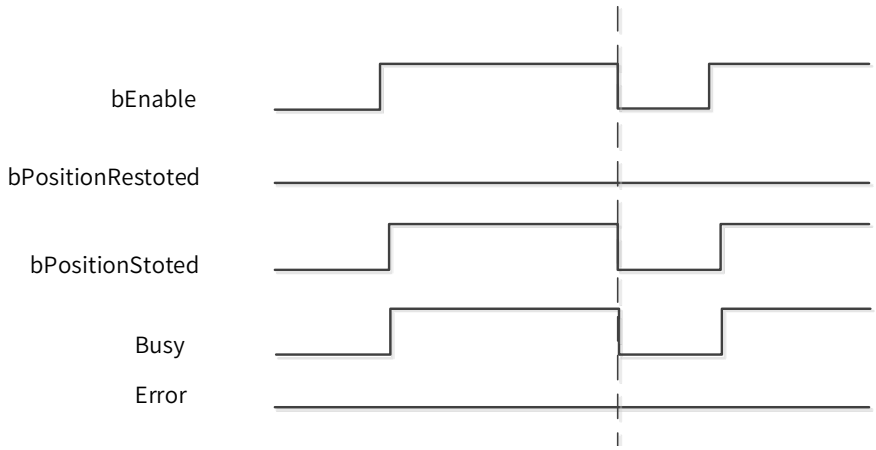
成员	类型	初始值	描述
fOffsetPosition	LREAL	0	位置偏移
dwPosOffsetForResiduals	DWORD	0	位置偏移剩余误差
dwActPosition	DWORD	0	当前的编码器位置
iIncrementsCompensated	INT	0	--
iTurn	INT	0	PLC 计算的溢出次数
wCheckSum	WORD	0	校验值

4） 时序图

■ 上电过程时序状态图



■ 下电过程时序状态图



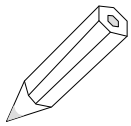
5） 错误说明

MC_ERROR 中 eErrorID 所对应的错误，如下图所示。

成员	类型	枚举值	描述
MC_NO_ERROR	INT	0	无错误
MC_PP_CRC_ERROR	INT	20000	数据恢复异常，CRC 校验错误
MC_PP_THRESTHOLD_EXCEEDED	INT	20001	当前的位置与保存的位置差值大于门槛值。

Memo NO. _____

Date / /



第 7 章 诊断

7 诊断

7.1 诊断简介

诊断是为了快速定位 PLC 运行过程出现的错误，通过错误信息和状态找出应对的解决方案。InoProShop 上的诊断界面只有登录 PLC 后才能获取和显示。

InoProShop 编程系统支持各种通信设备的诊断，可以根据各通信设备实际运行状态生成故障、离线等信息。

故障诊断涉及的模块类型主要包括：CPU 模块、Modbus、ModbusTCP、EtherCAT、CANopen、CANlink、Profibus-DP 等。

InoProShop 编程系统主要提供了四种获取诊断方式：组态诊断，诊断信息列表、设备自身诊断信息列表和诊断编程接口。

所有诊断都是通过诊断码解析获取，诊断码和诊断编程接口相对应。

7.2 组态诊断

组态分为网络组态和硬件组态，相应的诊断也分为网络组态诊断和硬件组态诊断。组态中，每个通信模块诊断状态通过不同的图标来呈现不同状态：运行状态、停止状态、离线状态和故障状态，即：

 ：运行状态，设备无故障。

 ：停止状态，设备没有运行，处于停止状态。

 ：离线状态，设备没有连接或者设备不存在。

 ：故障状态，设备处于故障状态，不能正常运行。

在组态中能直接看到设备的运行状态。

7.2.1 网络组态诊断

网络组态可以配置一个 PLC 总线系统，激活总线并添加从站。在登录状态下打开网络组态，可以看到其中各个通信设备的诊断状态，如下图所示：

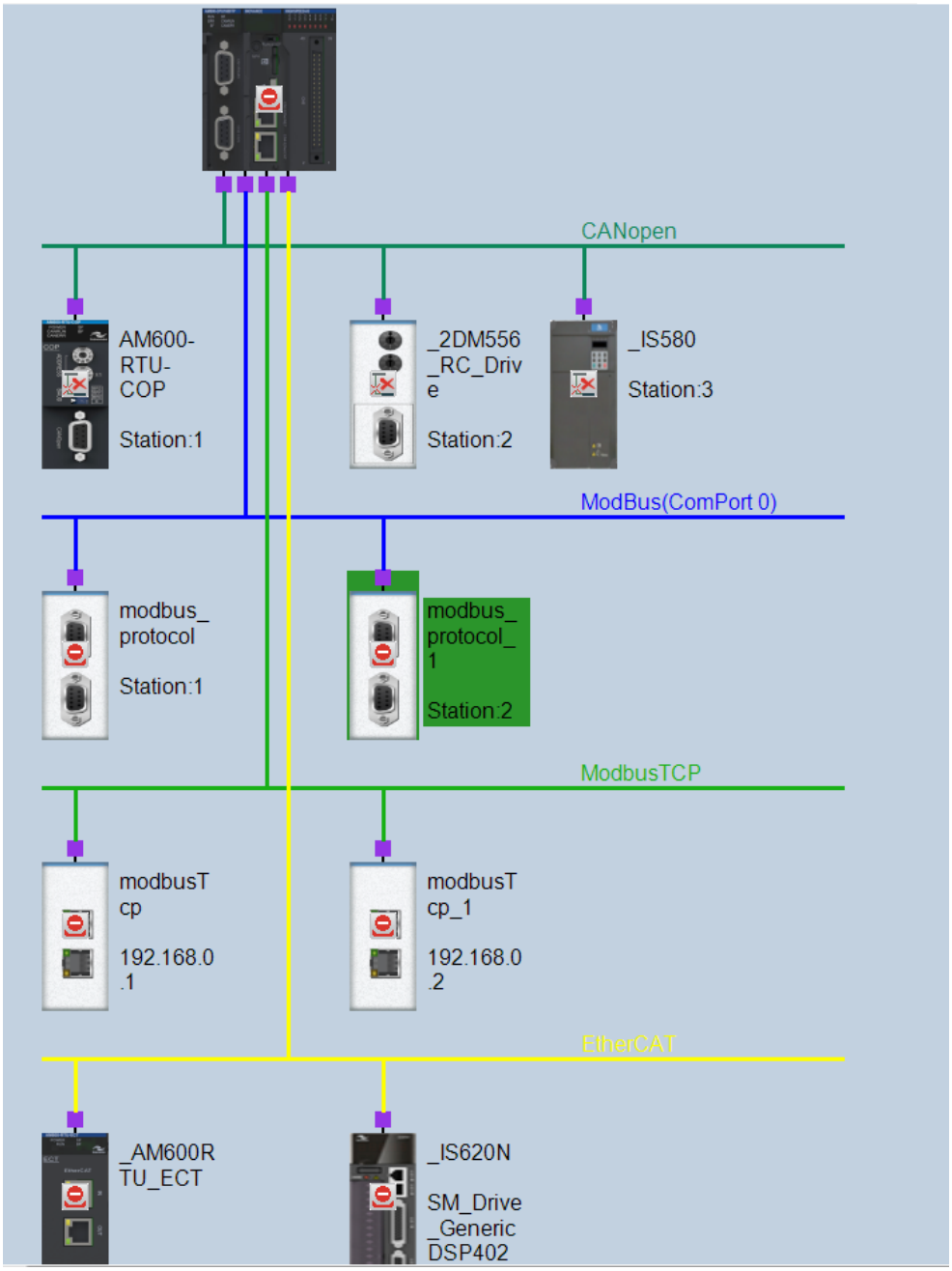


图 7-1 网络组态诊断

登录后，网络组态中的每个从站或者 CPU 会显示运行、故障或者离线状态，如需了解网络组态操作，请参见硬件组态。

7.2.2 硬件组态诊断

硬件组态主要用于添加总线对应的扩展模块，包括本地 IO 硬件组态、EtherCAT 硬件组态、CANopen 硬件组态；而 CANlink、Modbus 和 ModbusTCP 只在网络组态中显示。双击网络组态子节点或者网络组态中的从站模块可以打开硬件组态，也可以在一个硬件组态中选择另外一个硬件组态。硬件组态诊断基本相似，CANopen 硬件组态下图所示：

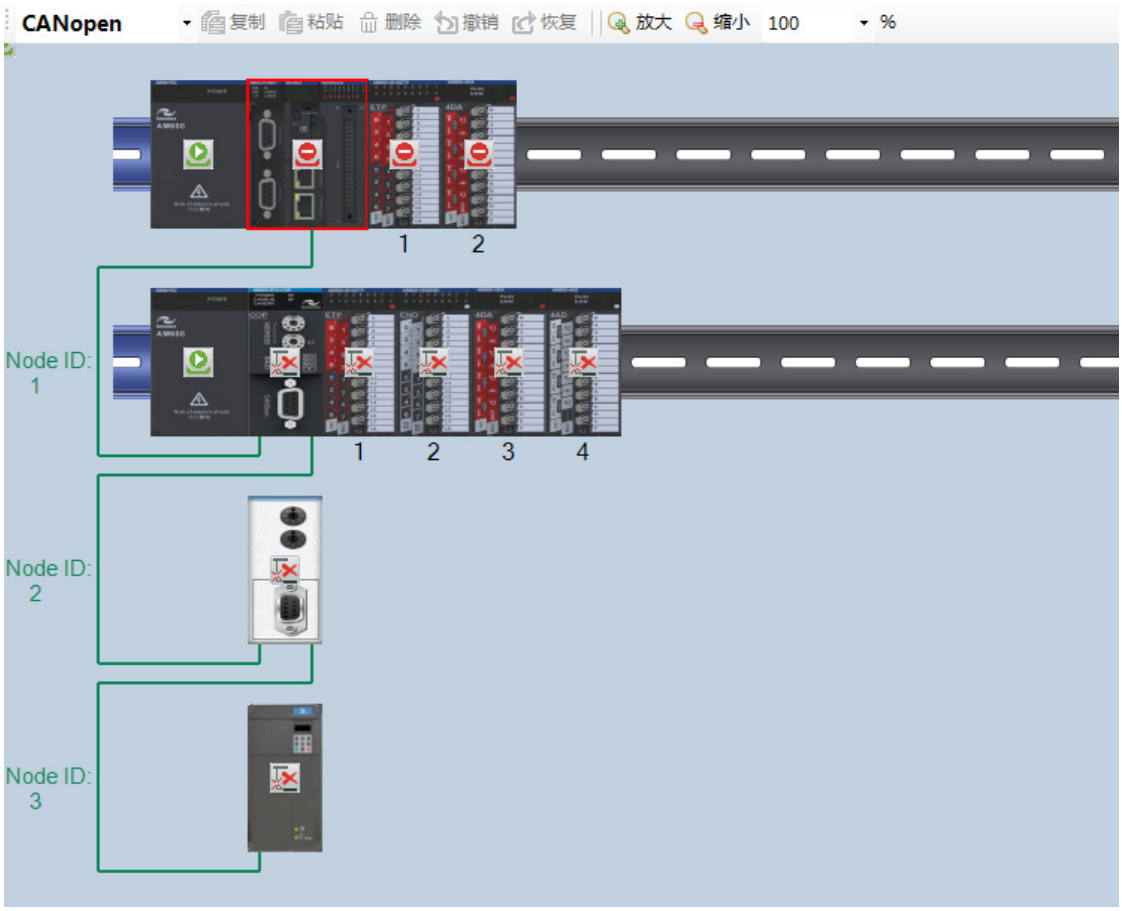


图 7-2 CANopen 硬件组态诊断

7.3 故障诊断

故障诊断用于显示所有设备出现的故障信息，并提供相关故障信息的详细说明、原因排查方法；同时，针对特殊情况还可以提供更详细的诊断信息。

与设备创建连接后，可以通过“工具”->“故障诊断”来打开故障诊断界面。故障诊断的界面如下图所示：

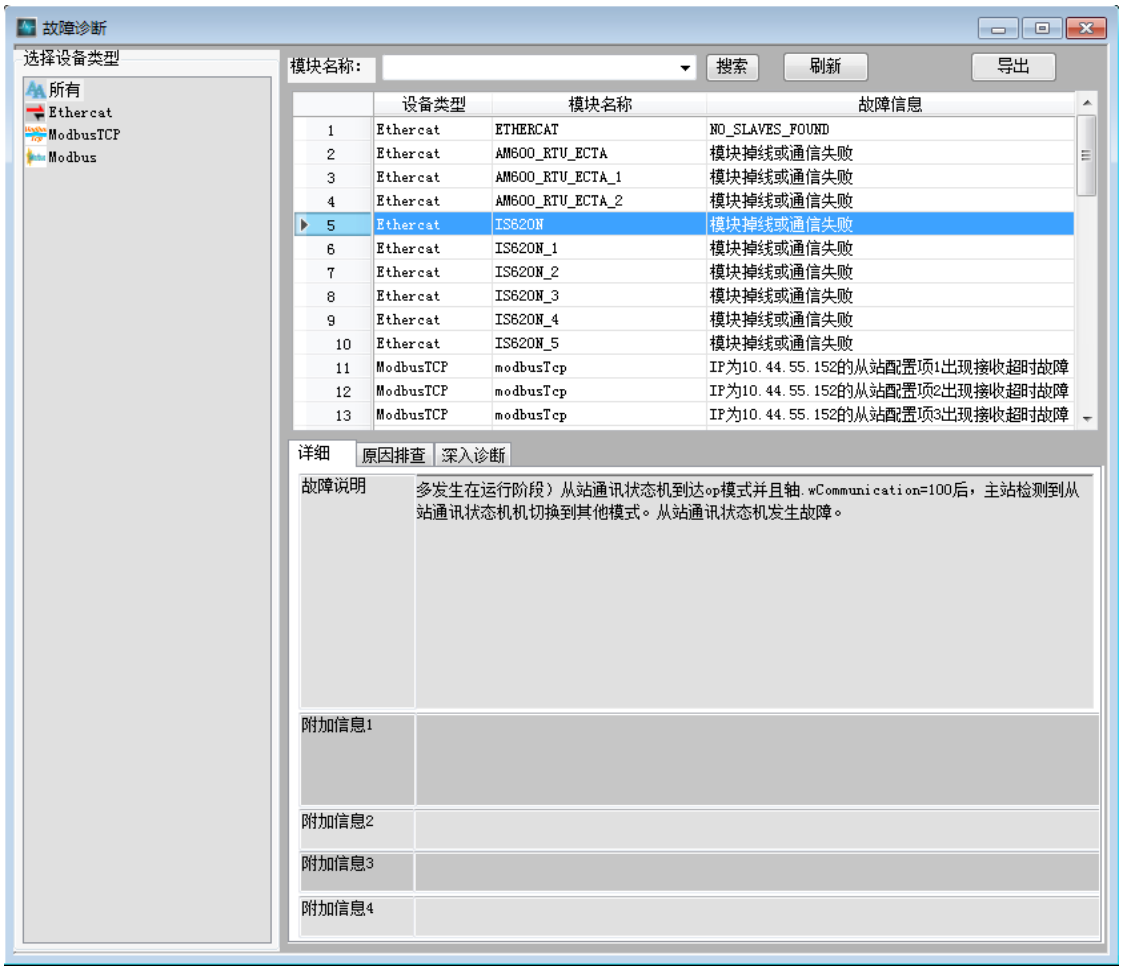


图 7-3 故障诊断界面图

诊断界面功能描述：

■ 设备类型窗口：用于显示当前出现故障的类型，同时提供故障显示过滤功能，可以针对特定设备类型进行故障信息显示。设备类型包括 CPU 模块、Modbus 模块、ModbusTCP 模块、本地模块、EtherCAT 模块 CANopen。选择不同的设备类型，诊断显示列表显示对应类型的诊断，默认为所有，显示所有设备诊断。

■ 按钮说明：

a.模块名称: 包含各个设备类型下的模块名称,默认包括全部设备下的设备名称,与搜索按钮组合实现搜索功能;

b．搜索按钮：用于针对某个特定模块或模块名称进行搜索过滤，支持模糊搜索；

c．刷新按钮：用于重新刷新故障信息；

d．导出按钮：用于将故障信息列表中的数据进行导出，导出诊断信息为 CSV 文件。

■ 故障信息列表：主要用于显示具体模块故障信息，包括设备类型，模块名称，故障信息。

■ 信息详细说明窗口：当在故障信息列表中选中了某一条故障信息，在信息详细说明窗口会显示该故障的详细信息。该窗口包括：详细、原因排查和深入诊断三个选项：

a、详细窗口：第一栏为故障可能原因说明项，后边增加了四个附加信息，主要用于对故障提供更多的信息说明。

b、原因排查窗口：用于提供原因排查的具体操作方法方式。

c、深入诊断窗口：针对某些复杂错误，需要获取更多细节信息定位的。[注]：该选项不是所有故障信息都有，目前有的主要是 EtherCAT 部分相关故障（主站报错、从站报错、ECTA 模块）。

7.4 设备自身诊断信息列表

除了设备诊断信息总表外，设备模块界面还包含一个模块自身诊断。模块自身诊断界面中的诊断列表包含诊断状态和诊断代码。下图为 ModbusRTU 从站诊断对话框示例：

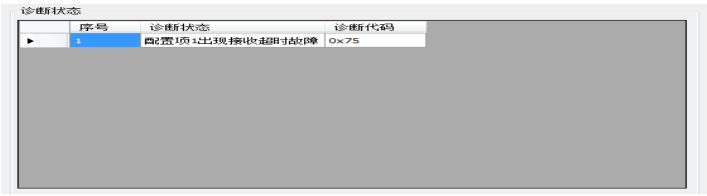


图 7-4 ModbusRTU 从站模块自身诊断列表对话框

诊断状态

- 序号：诊断列表诊断编号。
- 诊断状态：当前设备诊断信息。
- 诊断代码: 和诊断信息对应。有些诊断信息从设备状态或者设备标志位解析，没有诊断码，如 EtherCAT 的“设备出现故障”诊断，ModbusRTU 及 ModbusTCP 主站对从站的标志位诊断，只显示诊断信息，没有诊断码。

具有自身诊断界面的设备类型包括：EtherCAT、CANopen、Profibus-DP、ModbusRTU、ModbusTCP、高速 I/O 和 I/O 模块，CPU 诊断和 CANlink 诊断本身没有专门的诊断界面，可以在诊断信息列表中查看诊断信息。

7.4.1 CPU 诊断

CPU 本身不存在诊断界面，可以在诊断信息列表中查看诊断信息。

CPU 诊断码及诊断信息见 CPU 诊断码。

关于自身诊断界面的描述，请参考设备自身诊断信息列表简介。

7.4.2 EtherCAT 诊断

EtherCAT 诊断用于记录和描述总线错误信息，其中包括主站诊断、从站诊断、从站模块诊断、从站伺服驱动器诊断。EtherCAT 诊断仅对汇川技术的从站设备错误内容进行解析，诊断方法请参见本文章节 7.3 “故障诊断”，诊断的错误 ID 具体内容请参见本文附录。

有些应用场合需要通过触摸屏显示错误 ID，只需把 EtherCAT 主站下的变量 m_LastError 赋值给一个关联 HMI 地址的变量即可。如下图所示，,HMI_LastError 就是关联 HMI 地址的一个 WORD 类型变量。触摸屏上即可显示 EtherCAT 诊断的 ID。

AM600 EtherCAT 从站诊断

EtherCAT AM600 从站对应的 CANopen Emergency 帧格式如下：

应急错误代码		错误寄存器	制造商特定的错误区域				
BYTE0	BYTE1	BYTE2	BYTE3	BYTE4	BYTE5	BYTE6	BYTE7

应急错误代码		错误寄存器	制造商特定的错误区域		
BaseInfo	SlaveError	0x80	InterCommError	ConformenceError	IOModulePosError

NOTE

错误寄存器为 0x80，表示从站 Emergency 帧。

其中 BaseInfo 暂时未使用，其余诊断码及诊断信息见 CANopen 诊断码。如果诊断格式符合此格式则解析为对应的诊断信息，否则解析为“设备出现故障”。

关于自身诊断界面的描述，请参见设备自身诊断信息列表简介。

7.4.3 I/O 诊断

I/O 可以添加在 CPU、CANopen AM600 从站、Profibus-DP AM600 从站和 EtherCAT AM600 从站下。其诊断信息基本相同，具体诊断码及诊断信息请参见 I/O 模块诊断码。

关于自身诊断界面的描述，请参见设备自身诊断信息列表简介。

7.4.4 CANOpen 诊断

CANopen 诊断通过 Emergency 帧获取。在每个从站的“调试页面”，可以查看从站在线状态、诊断字符串和应急信息。

AM600 从站存在专门的设备诊断页面，对 AM600 从站进行特殊诊断，其诊断码及诊断信息参见 CANopen 诊断码，其自身诊断界面的描述，请参考设备自身诊断信息列表简介。

此外，通过 CANopen 特殊软元件也可以获取 CANopen 从站状态。

7.4.5 Profibus-DP 诊断

Profibus-DP 诊断主要是指 Profibus-DP 从站诊断，数据在诊断数组中。每个从站都包含一个“从站诊断”界面，如下图所示，其中显示从站诊断信息。对于从站下的非 AM600 模块，诊断信息显示在此诊断界面中，对于 AM600 Profibus-DP 从站 I/O 模块，诊断信息显示在 I/O 模块本身界面中，详情请参见 I/O 诊断。

对于 Profibus-DP 通道诊断，包括已定义的通道诊断和 GSD 文件定义的通道诊断，详情请参见 Profibus-DP 诊断简介中的通道诊断。

图 7-5 Profibus-DP 从站诊断

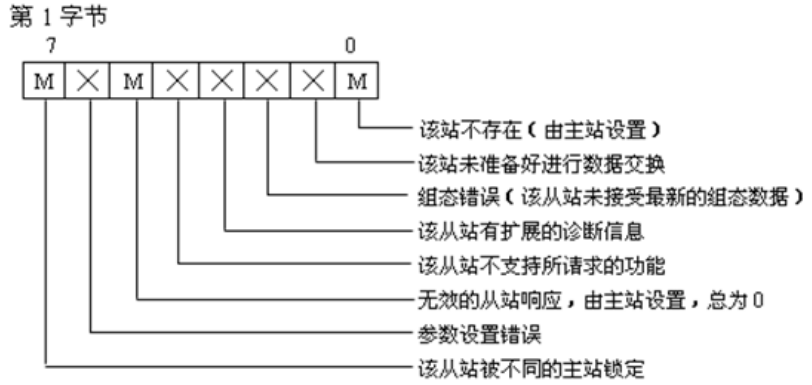
- 主站地址：主站站地址，诊断数组中的第 4 个字节。
- 制造商 ID：从站自定义的 ID，诊断数组中的第 5 和第 6 个字节。GSD 文件也定义了此 ID。
- 16 进制格式诊断：诊断数组数据 16 进制显示。
- 从站标准诊断：从站基本诊断信息和扩展诊断信息中状态诊断和标示符诊断，详见 Profibus-DP 诊断简介。
- 指定通道诊断：从站扩展诊断中的通道诊断，包括已定义的通道诊断和 GSD 文件定义的通道诊断，详见 DP 诊断简介中的通道诊断。

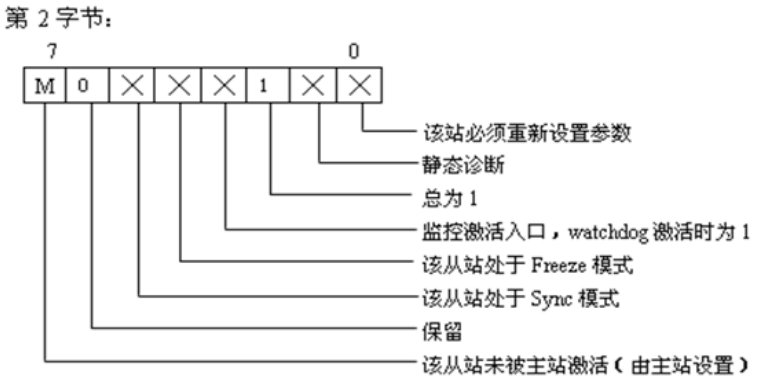
诊断数组结构如下表：

表 7-1 诊断数组结构

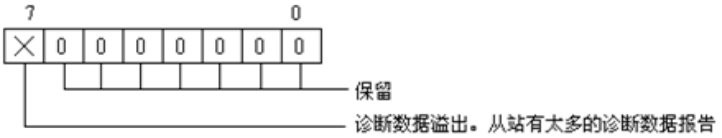
前 6 字节为基本诊断信息（必选）	报警或状态信息块（4-63 字节）（可选）	标志模块诊断信息块（可选）	通道诊断信息块（每个通道 3 个字节）（可选）
1----------6 基本诊断信息部分	7----------244 扩展诊断信息部分		
DU 单元最少 6 字节，最多可有 244 字节			

1） 基本诊断信息





第 3 字节：



第 4 字节：为主站地址。范围为 0~7Dh（0~125）。当其值为 FFh（255）时，表明该从站未被任何从站控制或未进行参数设置。

第 5 字节：该从站设备的 PROFIBUS ID 号高字节。范围为 0~FFh（0~255）。

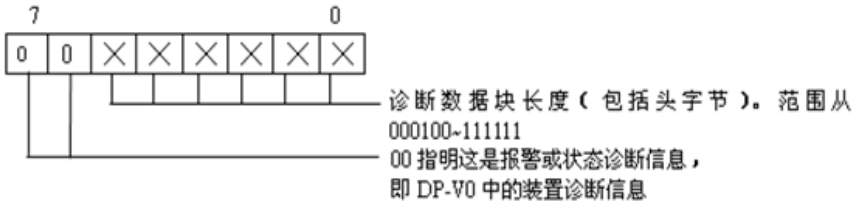
第 6 字节：该从站设备的 PROFIBUS ID 号低字节。范围为 0~FFh（0~255）。

2） 扩展诊断信息

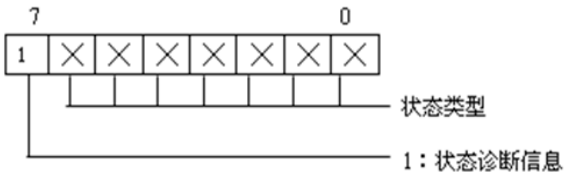
扩展信息诊断包括状态诊断、标示符诊断和通道诊断。

■ 状态诊断

第 7 个字节



第 8 个字节



当位 7 为 1 时，指明的是状态诊断信息，这时位 0~ 位 6 所指定的状态信息类型：

0：保留。

1：表示在状态详细特点信息字节后是状态信息。

2：表示在状态详细特点信息字节后是模块状态信息（影响第 9 字节后的字节内容）。

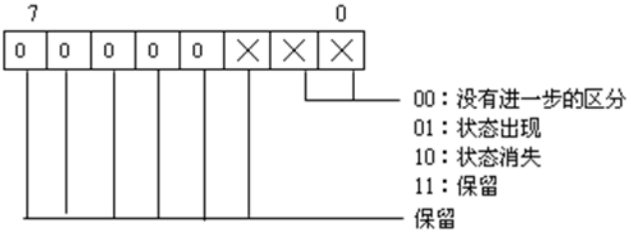
3~31：保留。

32~126：表示在状态详细特点信息字节后是制造商特殊数据。

127：保留。

第 9 个字节：用来指明报告状态异常的从站设备的槽号，范围：0~254。

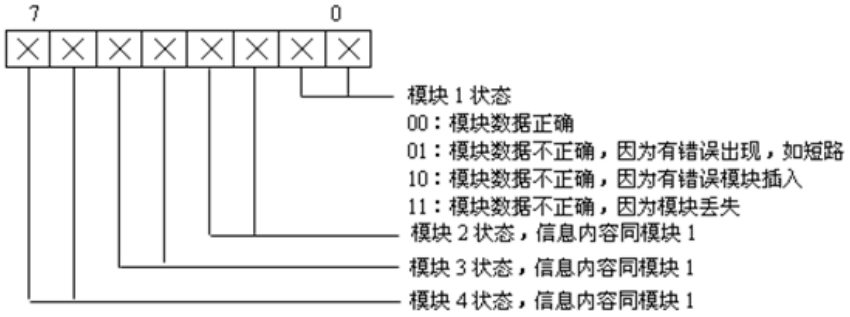
第 10 个字节：用来指定状态的详细特点。



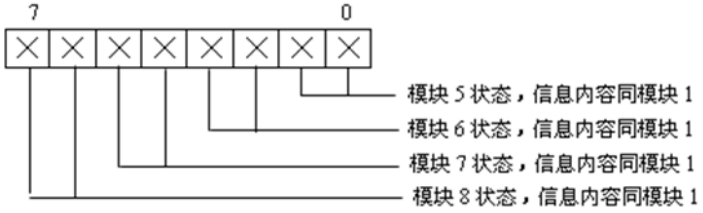
第 11 个字节以后：为用户数据字节。

若第 8 个字节中的状态类型为 2，即模块状态信息，则第 9 字节为 0，即从站槽号为 0。从而第 11 字节以后就

不再是用户数据字节了，其具体结构和含义如下：



第 12 字节：描述模块 5~ 模块 8 的状态

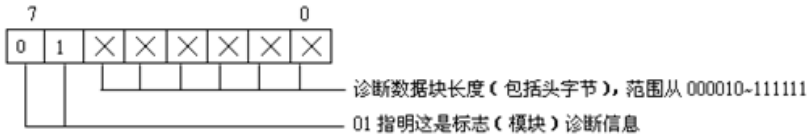


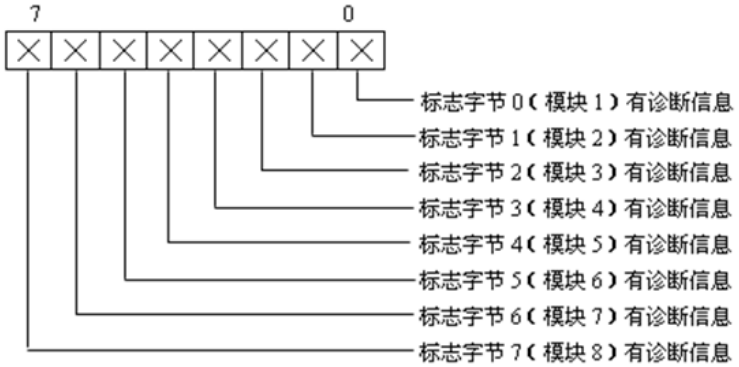
以后的字节可以依照上述的排列规则继续下去，直至将所有模块的信息写完。

■ 标示符诊断

头字节	和标志（模块）有关的诊断数据字节
01××××××	1-62 字节

和上面的头字节类似，头字节指明“标志诊断信息”类型和长度（即有几个字节的诊断信息），该长度包括头字节。头字节结构及含义如下：





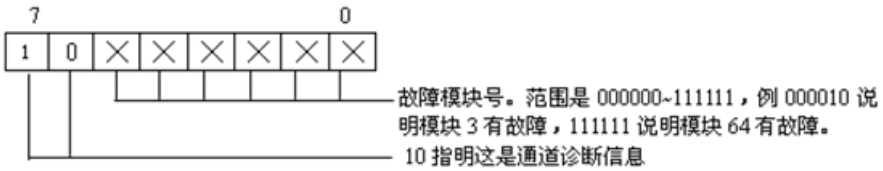
如果该设备的模块数多于 8 个，则可以继续使用接下去的字节指明标志字节号（或模块号）

■ 通道诊断

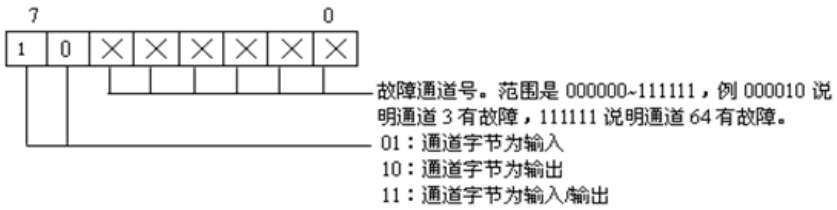
每个通道诊断信息由 3 个字节组成。通道诊断包含多个通道诊断信息，一个通道诊断信息的结构如下：

头字节	和通道有关的诊断数据字节
10××××××	2 字节（包括头字节有 3 个字节）

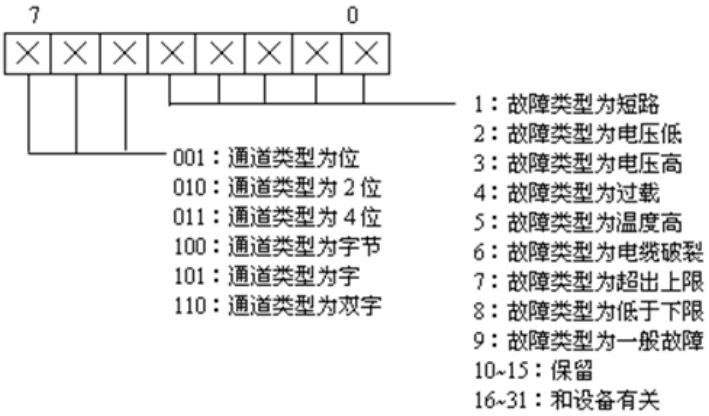
头字节指明“通道诊断信息”类型和发生故障的模块号。头字节的具体结构和含义如下：



第 2 个字节：说明通道字节类型，具体结构和含义如下：



第 3 个字节说明通道字节长度和故障类型，具体结构和含义如下：



7.4.6 ModbusRTU 诊断

ModbusRTU 支持 Modbus 串口 0 和串口 1 两个串口总线。Modbus 串口 0 或者串口 1 可以作为 Modbus 主站或者 Modbus 从站。

Modbus 串口作为主站时，可以为主站添加从站（远程从站）。在主站和从站的配置界面中，都有一个“设备诊断”界面。主站诊断信息主要是用于标示从站配置项出现故障，不包含具体故障原因，因此“设备诊断”界面没有故障码；在从站“设备诊断”页面中详细说明了哪个配置项出现了具体的故障信息。

Modbus 串口作为从站时，也有一个“设备诊断”界面，显示此从站和主站通信出现的故障，此界面显示内容详见设备自身诊断信息列表。

Modbus 串口作为主站或者从站时其诊断码及诊断信息都是一致的，请参见 Modbus 诊断码。

7.4.7 ModbusTCP 诊断

AM600 PLC 可以作为 ModbusTCP 主站，也可以作为 ModbusTCP 从站。

ModbusTCP 作为主站时，可以为主站添加从站（远程从站）。在主站和从站的配置界面中，都有一个“设备诊断”界面。主站诊断信息主要是用于标示从站配置项出现故障，不包含具体故障原因，因此“设备诊断”界面没有故障码；在从站“设备诊断”页面中详细说明了哪个配置项出现了具体的故障信息。

ModbusTCP 作为从站时，也有一个“设备诊断”界面，显示此从站和主站通信出现的故障，此界面显示内容详见设备自身诊断信息列表。

ModbusTCP 作为主站或者从站时其诊断码及诊断信息都是一致的，见 Modbus 诊断码。

7.4.8 CANlink 诊断

CANlink 本身没有“设备诊断”界面，但是在 CANlink 的网络管理功能中启动监控后，可以查看从站的在线和运行状态，详情请参见 CANlink 网络管理。

通过软元件可以获取 CANlink 站点状态，详情请参见 CANlink 软元件。

登陆 PLC 后，可以在诊断信息列表中查看 CANlink 的具体诊断信息。CANlink 诊断码及诊断信息请参见 CANlink 诊断码。

关于自身诊断界面的描述，请参见设备自身诊断信息列表简介。

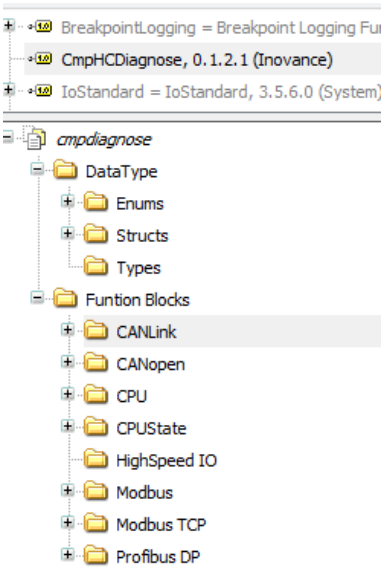
7.5 诊断编程接口

【注】只有 AM400 和 AM600 系列 PLC 支持该功能。

7.5.1 诊断编程接口简介

诊断编程接口提供了在用户程序中获取诊断的解决方案：可以在用户程序中判断各个设备模块的诊断信息，从而作出相关的处理。

诊断编程接口以库的形式存在，可以在“库管理器”添加，添加后如下图所示。



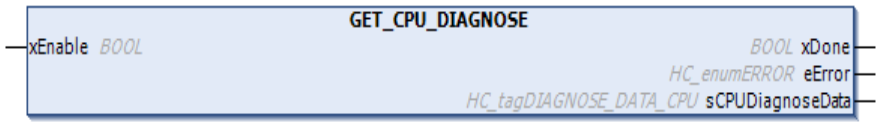
编程接口中提供了 CANlink，CANopen、CPU、Modbus、ModbusTCP、Profibus-DP 对应的诊断编程接口，每种诊断对应一组功能块，用于获取对应的诊断码。

在获取诊断数据时，自定义的诊断结果和诊断状态在“DataType”中定义。获取每个诊断时一般都有一个 HC_enumERROR 类型表示诊断是否获取成功，具体的定义见下表。

枚举器	值（十进制）	说明
NO_ERROR	0	无错误。
WRONG_PARAMETER	1	参数错误。
UNKNOWN_DEVICEID	2	无此设备 ID。
INVAILD_DEVICEID	3	设备 ID 无效。
INVAILD_IO_POS	4	无效 IO 位置。
UNSUPPORT_DIAGNOSE	5	不支持诊断。
TIME_OUT	6	超时。
INTERNAL_FB_ERROR	7	内部功能块错误。
UNKNOWN_ERROR	8	未知错误。
INVAILD_IP	9	无效 IP。

7.5.2 CPU 诊断编程接口

1 CPU 自身诊断编程接口

获取 CPU 诊断数据 GET_CPU_DIAGNOSE


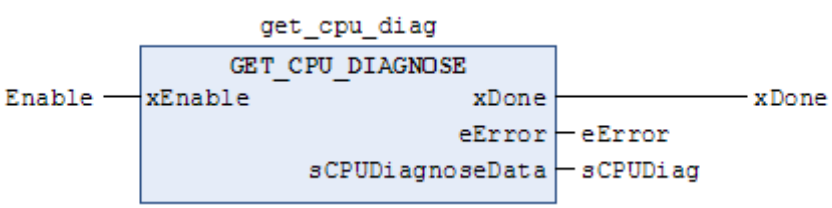
参数名称	参数类型	初始值	参数作用
输入参数			
xEnable	BOOl	FALSe	功能块的使能位，电平触发
输出参数			
xDone	Bool	FALSE	获取诊断结果是否完成
eError	HC_enumERROR	NO_ERROR	获取诊断结果是否成功
sCPUDiagnoseData	HC_tagDIAGNOSE_DATA_CPU		CPU 诊断数据

HC_tagDIAGNOSE_DATA_CPU 为结构体数据类型，如下表，每个数据诊断码和诊断信息关系，见诊断章节 CPU 诊断码

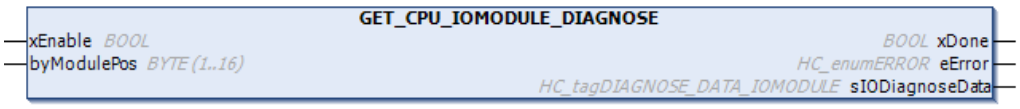
名称	类型
SDCardError	BYTE
FlashError	BYTE
SystemError	BYTE
InterCommError	BYTE
ConformenceError	WORD
IOModulePosError	WORD
FunctionErrorCode	WORD

示例

```
PROGRAM POU
VAR
     get_cpu_diag: GET_CPU_DIAGNOSE;
    Enable: BOOL;
     eError: HC_enumERROR;
     xDone: BOOL;
     sCPUDiag: HC_tagDIAGNOSE_DATA_CPU;
END_VAR
```



2 CPU 本地 IO 扩展模块

获取 CPU 下 IO 诊断 GET_CPU_IOMODULE_DIAGNOSE


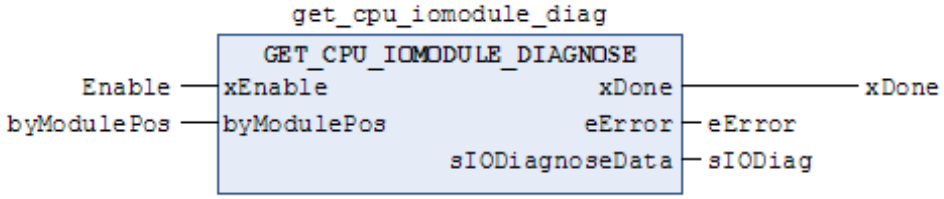
参数名称	参数类型	初始值	参数作用
输入参数			
xEnable	BOOl	FALSe	功能块的使能位，电平触发
byModulePos	BYTE (1..16)	0	获取的 IO 位置
输出参数			
xDone	Bool	FALSE	获取诊断结果是否完成
eError	HC_enumERROR	NO_ERROR	获取诊断结果是否成功
sIODiagnoseData	HC_tagDIAGNOSE_DATA_IOMODULE		IO 诊断数据

HC_tagDIAGNOSE_DATA_IOMODULE 为结构体数据类型，如下表，每个数据诊断码和诊断信息关系，见诊断章节 I/O 诊断码

结构成员	类型	说明
ModuleError	BYTE	模块错误。
ChannelError	ARRAY[0..3] OF BYTE	通道错误。

示例

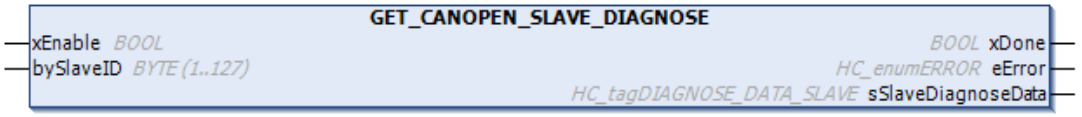
```
PROGRAM POU
VAR
     get_cpu_iomodule_diag: GET_CPU_IOMODULE_DIAGNOSE;
    Enable: BOOL;
     eError: HC_enumERROR;
     xDone: BOOL;
     byModulePos: BYTE (1..16);
     sIODiag: HC_tagDIAGNOSE_DATA_IOMODULE;
END_VAR
```



7.5.3 CANopen 诊断编程接口

1 CANopen 从站诊断编程接口

获取 CANopen 从站诊断数据 GET_CANOPEN_SLAVE_DIAGNOSE



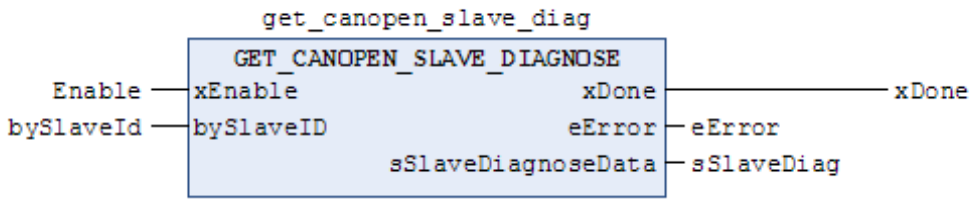
参数名称	参数类型	初始值	参数作用
输入参数			
xEnable	BOOl	FALSe	功能块的使能位，电平触发
bySlaveID	BYTE (1..127)	0	获取的从站节点 ID，范围 1-127
输出参数			
xDone	Bool	FALSE	获取诊断结果是否完成
eError	HC_enumERROR	NO_ERROR	获取诊断结果是否成功
sSlaveDiagnoseData	HC_tagDIAGNOSE_DATA_SLAVE		CANopen 从站诊断数据

HC_tagDIAGNOSE_DATA_SLAVE 为结构体数据类型，如下表，每个数据诊断码和诊断信息关系，见诊断章节 CANopen 诊断码

结构成员	类型	说明
SlaveError	BYTE	从站错误。
InterCommError	BYTE	内部通信错误。
ConformenceError	WORD	一致性错误。
IOModulePosError	WORD	IO 模块位置错误。

示例

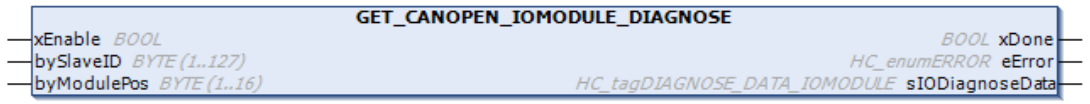
```
PROGRAM POU
VAR
     get_CANOpen_slave_diag: GET_CANOPEN_SLAVE_DIAGNOSE;
    Enable: BOOL;
     eError: HC_enumERROR;
     xDone: BOOL;
     bySlaveId: BYTE (1..127);
     sSlaveDiag: HC_tagDIAGNOSE_DATA_SLAVE;
END_VAR
```



NOTE

该诊断接口仅适用于 InoProShop1.2 及之前版本。使用 InoProShop1.3 及之后的版本时，如需诊断从站通讯状态，请在库管理器中添加 CmpHCCiA405 库 , 调用 GET_STATE 接口进行获取。

2 CANOpen 从站下 IO 诊断编程接口

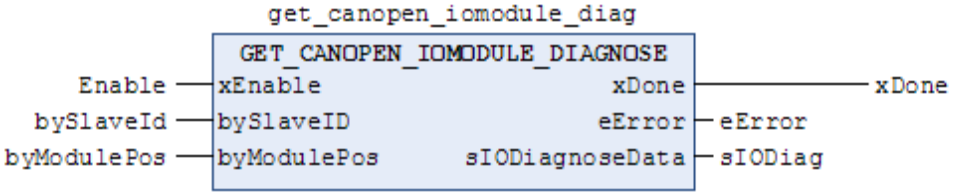
获取 CANopen 从站 IO 诊断 GET_CANOPEN_IOMODULE_DIAGNOSE			
			
参数名称	参数类型	初始值	参数作用
输入参数			
xEnable	BOOl	FALSE	功能块的使能位，电平触发
bySlaveID	BYTE (1..127)	0	从站节点 ID，范围 1-127
byModulePos	BYTE (1..16)	0	获取诊断的 IO 位置
输出参数			
xDone	Bool	FALSE	获取诊断结果是否完成
eError	HC_enumERROR	NO_ERROR	获取诊断结果是否成功
sIODiagnoseData	HC_tagDIAGNOSE_DATA_IOMODULE		IO 诊断数据

HC_tagDIAGNOSE_DATA_IOMODULE 为结构体数据类型，如下表，每个数据诊断码和诊断信息关系，见诊断章节 I/O 诊断码

结构成员	类型	说明
ModuleError	BYTE	模块错误。
ChannelError	ARRAY[0..3] OF BYTE	通道错误。

示例

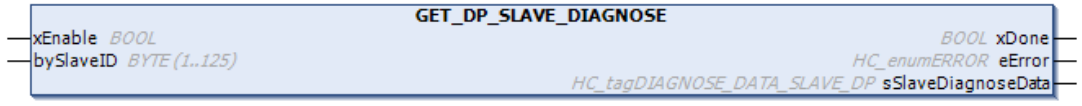
```
PROGRAM POU
VAR
    get_CANOpen_iomodule_diag: GET_CANOPEN_IOMODULE_DIAGNOSE;
 Enable: BOOL;
    eError: HC_enumERROR;
    xDone: BOOL;
    bySlaveId: BYTE (1..127);
    byModulePos: BYTE (1..16);
    sIODiag: HC_tagDIAGNOSE_DATA_IOMODULE;
END_VAR
```



7.5.4 Profibus-DP 诊断编程接口

1 Profibus-DP 从站诊断编程接口

获取 DP 从站诊断数据 GET_DP_SLAVE_DIAGNOSE



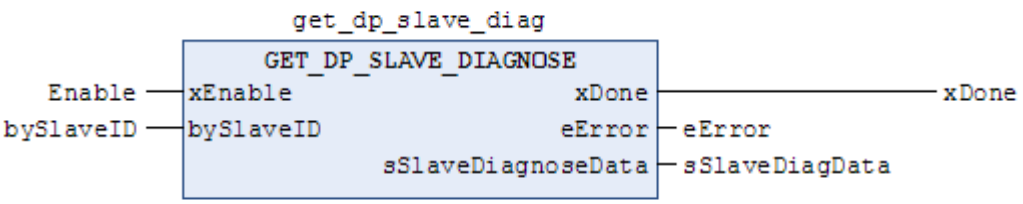
参数名称	参数类型	初始值	参数作用
输入参数			
xEnable	BOOl	FALSe	功能块的使能位，电平触发
bySlaveID	BYTE (1..125)	0	获取诊断从站的站地址，范围 1-125
输出参数			
xDone	Bool	FALSE	获取诊断结果是否完成
eError	HC_enumERROR	NO_ERROR	获取诊断结果是否成功
sSlaveDiagnoseData	HC_tagDIAGNOSE_DATA_SLAVE_DP		DP 从站诊断数据

HC_tagDIAGNOSE_DATA_SLAVE_DP 为结构体数据类型，如下表，每个数据诊断码和诊断信息关系，见诊断章节 DP 诊断码

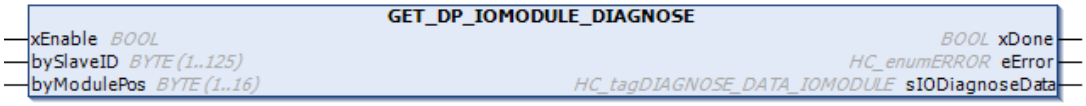
结构成员	类型	说明
Length	BYTE	诊断数据长度。
ExtDiagData	ARRAY[0..243]OF BYTE	诊断数据。

示例

```
PROGRAM POU
VAR
     get_dp_slave_diag: GET_DP_SLAVE_DIAGNOSE;
    Enable: BOOL;
     eError: HC_enumERROR;
     xDone: BOOL;
     bySlaveID: BYTE (1..125);
     sSlaveDiagData: HC_tagDIAGNOSE_DATA_SLAVE_DP;
END_VAR
```



2 Profibus-DP 从站下 IO 诊断编程接口

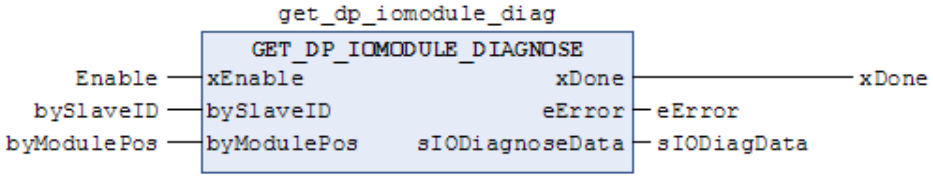
获取 DP 从站 IO 诊断 GET_DP_IOMODULE_DIAGNOSE			
			
参数名称	参数类型	初始值	参数作用
输入参数			
xEnable	BOOl	FALSE	功能块的使能位，电平触发
bySlaveID	BYTE (1..125)	0	从站的站地址，范围 1-125
byModulePos	BYTE (1..16)	0	获取诊断的 IO 位置
输出参数			
xDone	Bool	FALSE	获取诊断结果是否完成
eError	HC_enumERROR	NO_ERROR	获取诊断结果是否成功
sIODiagnoseData	HC_tagDIAGNOSE_DATA_IOMODULE		IO 诊断数据

HC_tagDIAGNOSE_DATA_IOMODULE 为结构体数据类型，如下表，每个数据诊断码和诊断信息关系，见诊断章节 I/O 诊断码。

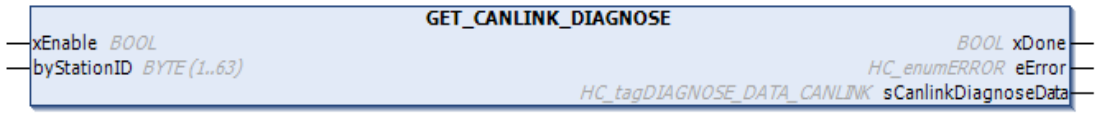
ModuleError	BYTE
ChannelError	ARRAY [0..3] OF BYTE

示例

```
PROGRAM POU
VAR
     get_dp_iomodule_diag: GET_DP_IOMODULE_DIAGNOSE;
    Enable: BOOL;
     eError: HC_enumERROR;
     xDone: BOOL;
     bySlaveID: BYTE (1..125);
     byModulePos: BYTE (1..16);
     sIODiagData: HC_tagDIAGNOSE_DATA_IOMODULE;
END_VAR
```



7.5.5 CANlink 诊断编程接口

获取 CANlink 诊断数据 GET_CANLINK_DIAGNOSE


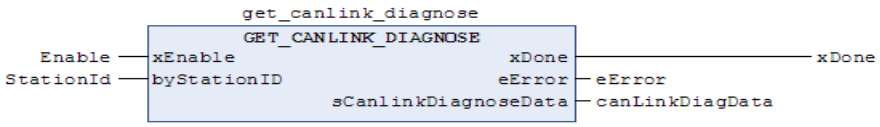
参数名称	参数类型	初始值	参数作用
输入参数			
xEnable	BOOl	FALSe	功能块的使能位，电平触发
byStationID	BYTE (1..63)	0	获取的站节点 ID，范围 1-63
输出参数			
xDone	Bool	FALSE	获取诊断结果是否完成
eError	HC_enumERROR	NO_ERROR	获取诊断结果是否成功
sCanlinkDiagnoseData	HC_tagDIAGNOSE_DATA_CANLINK		CANlink 站点诊断数据

HC_tagDIAGNOSE_DATA_CANLINK 为结构体数据类型，如下表，每个数据诊断码和诊断信息关系，见诊断章节 CANlink 诊断码。

结构成员	类型	说明
IsUsed	BOOL	是否使用。
IsMaster	BOOL	是否是主站。
StationStatus	WORD	CANlink 站状态。
CfgFrameError	WORD	配置帧错误。
CmdFrameError	WORD	命令帧错误。

示例

```
PROGRAM POU
VAR
     get_canlink_diagnose:GET_CANLINK_DIAGNOSE;
 Enable: BOOL;
     StationId: BYTE (1..63);
     eError: HC_enumERROR;
     canLinkDiagData: HC_tagDIAGNOSE_DATA_CANLINK;
     xDone: BOOL;
END_VAR
```



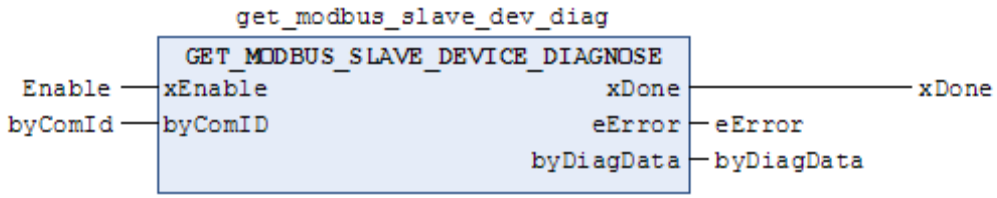
7.5.6 MODBUS 诊断编程接口

1 MODBUS 本地从站诊断编程接口

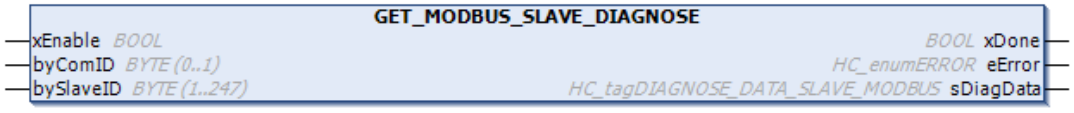
获取 MODBUS 本地从站诊断数据 GET_MODBUS_SLAVE_DEVICE_DIAGNOSE			
			
参数名称	参数类型	初始值	参数作用
输入参数			
xEnable	BOOl	FALSe	功能块的使能位，电平触发
byComID	BYTE (0..1)	0	本地从站使用的串口号，范围 0-1
输出参数			
xDone	Bool	FALSE	获取诊断结果是否完成
eError	HC_enumERROR	NO_ERROR	获取诊断结果是否成功
byDiagData	Byte		诊断码，每个数据诊断码和诊断信息关系，见诊断章节 Modbus 诊断码

示例

```
PROGRAM POU
VAR
     get_modbus_slave_dev_diag: GET_MODBUS_SLAVE_DEVICE_DIAGNOSE;
    Enable: BOOL;
     eError: HC_enumERROR;
     xDone: BOOL;
     byComId: BYTE (0..1);
     byDiagData: BYTE;
END_VAR
```



2 MODBUS 远程从站诊断编程接口

获取 MODBUS 远程从站诊断 GET_MODBUS_SLAVE_DIAGNOSE			
			
参数名称	参数类型	初始值	参数作用
输入参数			

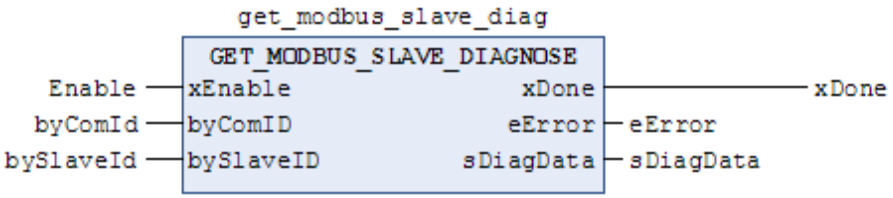
xEnable	BOOl	FALSE	功能块的使能位，电平触发
byComID	BYTE (0..1)	0	主站使用的串口号，范围 0-1
bySlaveID	BYTE (1..247)	0	从站的站地址，范围 1-247
输出参数			
xDone	Bool	FALSE	获取诊断结果是否完成
eError	HC_enumERROR	NO_ERROR	获取诊断结果是否成功
sDiagData	HC_tagDIAGNOSE_DATA_SLAVE_MODBUS		从站诊断数据

HC_tagDIAGNOSE_DATA_SLAVE_MODBUS 为结构体数据类型，如下表，每个数据诊断码和诊断信息关系，见诊断章节 MODBUS 诊断码。

名称	类型
ChannelNum	BYTE
DiagData	BYTE

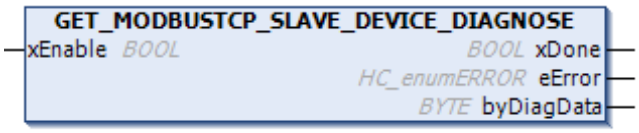
示例

```
PROGRAM POU
VAR
     get_modbus_slave_diag: GET_MODBUS_SLAVE_DIAGNOSE;
    Enable: BOOL;
     eError: HC_enumERROR;
     xDone: BOOL;
     byComId: BYTE (0..1);
     bySlaveId: BYTE (1..247);
     sDiagData: HC_tagDIAGNOSE_DATA_SLAVE_MODBUS;
END_VAR
```



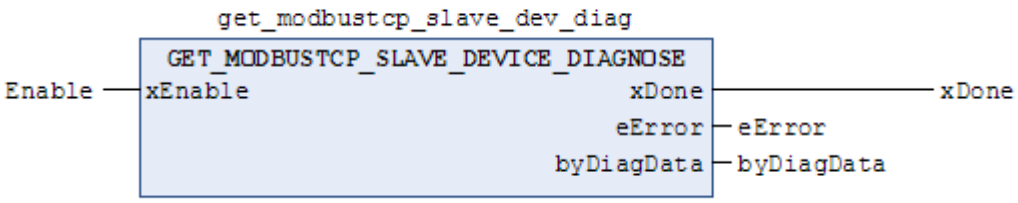
7.5.7 MODBUSTCP 诊断编程接口

MODBUSTCP 本地从站诊断编程接口

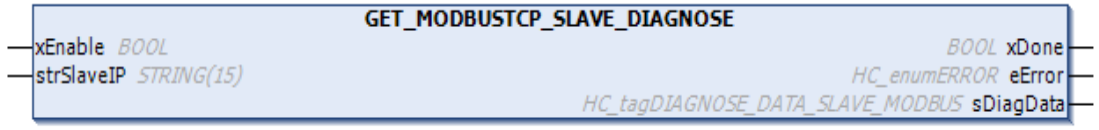
获取 MODBUSTCP 本地从站诊断数据 GET_MODBUSTCP_SLAVE_DEVICE_DIAGNOSE			
			
参数名称	参数类型	初始值	参数作用
输入参数			
xEnable	BOOl	FALSe	功能块的使能位，电平触发
输出参数			
xDone	Bool	FALSE	获取诊断结果是否完成
eError	HC_enumERROR	NO_ERROR	获取诊断结果是否成功
byDiagData	Byte		诊断码，每个数据诊断码和诊断信息关系，见诊断章节 Modbus 诊断码

示例

```
PROGRAM POU
VAR
     get_modbustcp_slave_dev_diag: GET_MODBUSTCP_SLAVE_DEVICE_DIAGNOSE;
    Enable: BOOL;
     eError: HC_enumERROR;
     xDone: BOOL;
     byDiagData: BYTE;
END_VAR
```



MODBUSTCP 远程从站诊断编程接口

获取 MODBUSTCP 远程从站诊断 GET_MODBUSTCP_SLAVE_DIAGNOSE			
			
参数名称	参数类型	初始值	参数作用
输入参数			
xEnable	BOOl	FALSE	功能块的使能位，电平触发

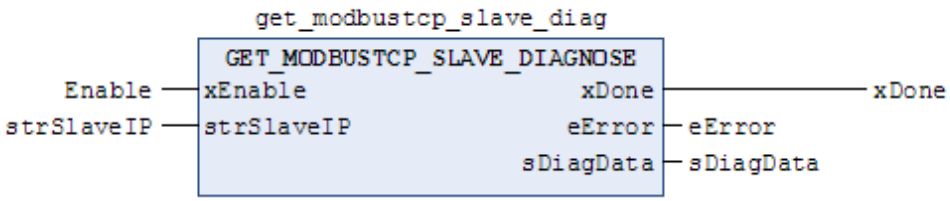
strSlaveIP	STRING(15)	‘’	远程从站 IP 地址
输出参数			
xDone	Bool	FALSE	获取诊断结果是否完成
eError	HC_enumERROR	NO_ERROR	获取诊断结果是否成功
sDiagData	HC_tagDIAGNOSE_DATA_SLAVE_MODBUS		从站诊断数据

HC_tagDIAGNOSE_DATA_SLAVE_MODBUS 为结构体数据类型，如下表，每个数据诊断码和诊断信息关系，见诊断章节 MODBUS 诊断码

名称	类型
ChannelNum	BYTE
DiagData	BYTE

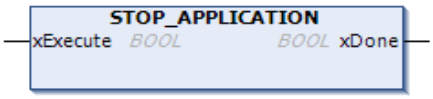
示例

```
PROGRAM POU
VAR
     get_modbustcp_slave_diag: GET_MODBUSTCP_SLAVE_DIAGNOSE;
    Enable: BOOL;
     eError: HC_enumERROR;
     xDone: BOOL;
     sDiagData: HC_tagDIAGNOSE_DATA_SLAVE_MODBUS;
     strSlaveIP: STRING(15);
END_VAR
```



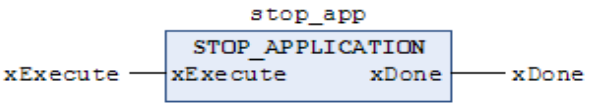
7.5.8 CPU 停止控制

功能块描述

停止应用程序 STOP_APPLICATION			
			
参数名称	参数类型	初始值	参数作用
输入参数			
xExecute	BOOL	FALSE	功能块的使能，上升沿触发
输出参数			
xDone	BOOL	FALSE	执行完成输出

示例

```
PROGRAM POU
VAR
    stop_app: STOP_APPLICATION;
    xExecute: BOOL;
END_VAR
```



7.5.9 EtherCAT 诊断

EtherCAT 诊断用于记录和描述总线错误信息，其中包括主站诊断、从站诊断、从站模块诊断、从站伺服驱动器诊断。EtherCAT 诊断仅对汇川技术的从站设备错误内容进行解析，诊断方法请参见本文章节 7.3 “故障诊断”，诊断的错误 ID 具体内容请参见本文附录。

有些应用场合需要通过触摸屏显示错误 ID，只需把 EtherCAT 主站下的变量 m_LastError 赋值给一个关联 HMI 地址的变量即可。如下图所示，,HMI_LastError 就是关联 HMI 地址的一个 WORD 类型变量。触摸屏上即可显示 EtherCAT 诊断的 ID。

附录

附录

附录 A 各通信端口的通信协议简介

中型 PLC 系列在出厂时就提供了 Mini-USB 口、串行通信口、Ethernet 网口、EtherCAT 网口、Mini-SD 卡插槽、CAN 通信口、Profibus-DP 通信口、高速 I/O 接口，本地总线扩展接口等。现将各端口的协议设置方法说明如下:

A.1 Mini-USB 端口及其内置通信协议

Mini-USB 口的主要用途是下载 PLC 用户程序、进行监控调试，因此该端口的通信协议是固定的，用户无需选择通信协议，只要 PC 安装了相应的 USB 驱动程序，PC 在 InoProshop 中就可以随时与中型 PLC 进行用户程序下载或监控。

Mini-USB 端口内置的下载协议是汇川公司的专有协议，不支持第三方编程软件对中型 PLC 的程序下载。

首次安装 InoProshop 编程软件后，USB 的驱动程序会自动安装。同一 PC 安装不同的版本的 InoProshop，必须使用不同的安装目录。

A.2 COM0/COM1 通信端口及其内置协议

COM0、COM1 口为 PLC 对外通信的基本端口 , 两个通信端口集成在同一个 DB9 物理端口上，其主要用途是 RS485 或者 Modbus 通信。

COM0、COM1 通信端口支持的协议与设置单元定义如下表:

<table>
<tr><th>COM0/COM1 协议</th><th>半双工 / 全双工模式</th><th>通信格式</th><th>波特率</th><th>数据位</th><th>停止位</th><th>校验</th></tr>
<tr><td>MODBUS-RTU 主站</td><td>半双工</td><td>固定</td><td rowspan="3">4800Bits/s
9600Bits/s
19200Bits/s
38400Bits/s
57600Bits/s
115200Bits/s</td><td>7bit
8bit</td><td rowspan="3">1bit
2bit</td><td rowspan="3">NONE
ODD
EVEN</td></tr>
<tr><td>MODBUS-RTU 从站</td><td>半双工</td><td>固定</td><td>8bit</td></tr>
<tr><td>RS485 自由协议</td><td>半双工</td><td>不固定</td><td>8bit</td></tr>
<tr><td>MODBUS-ASCII 主站</td><td>不支持</td><td>-</td><td>-</td><td>-</td><td>-</td><td>-</td></tr>
<tr><td>MODBUS-ASCII 从站</td><td>不支持</td><td>-</td><td>-</td><td>-</td><td>-</td><td>-</td></tr>
</table>

这几种协议简介如下:

■ Modbus 主站协议

PLC 作为控制主机，常用该协议与变频器、伺服等下位机进行通信，或读取智能仪表、传感器的信息等。PLC 之间采用 Modbus 通信，也会带来通信的灵活性。

■ Modbus 从站协议

当有上位机需要读取 PLC 的内部数据时，常采用 Modbus 协议，PLC 作为通信从站。当 PLC 将端口设置为 Modbus 从站后，PLC 根据上位机的通信命令，自动进行响应处理并给与应答。

■ 自由通信协议

PLC 内置通信协议以外的协议，都称为“自由通信协议”，要以自由协议进行通信，编程人员必须完整理解该协议的帧结构定义。编程人员根据从站通信协议，以及要求的通信操作，在用户程序中事先准备好寄存器区中待发送的数据串，系统将指定寄存器区域的数据，自动向串口依次发送；然后串口进入接收状态，将串口接收到的数据存放到指定区域，当接收到指定长度的数据后，通过系统标志，通知用户程序，以便用户程序对接收的数据，按照协议规定，去解析得到所要要求的数据。

AM600 系列自由通信协议中操作寄存器，相当于用户程序直接访问通信缓冲区，借助用户程序对通信收发缓冲区的处理，实现自定义协议的通信。实际编程时，需要作一些串行通信的配置和准备，如设定串口的收发模式、波特率、位数、校验位、软件协议的设定、超时判断条件、收发缓冲区的数据准备、收发标志处理等，才能按预期的要求进行通信。

A.3 CANopen 通信协议

CANopen 通信采用功能块读写 SDO/PDO 方式实现通讯。将需要的通信变量（协议中的对象字典数据）赋值给功能块对应的输入参数，触发执行条件即可访问从站设备数据，AM600 仅支持 CANopen 主站。

A.4 CANlink 通信协议

CANlink 通信可采用配置表格法，将需要的通信变量、希望的通信频度、触发条件等，采用填表方式，事先进行设定。作为网络 CANlink 主站时，可连接汇川各种远程扩展模块、MD380/500 系列变频器、IS620 系列伺服等从站设备；同时也可作为 CANlink 网络从站接入其他设备。

CANlink3.0 通信协议中，提供了如下通信帧：

- 定时触发和条件触发的通信帧，用于普通从站的通信数据交互；
- 同步触发，用于多个具有同步控制的高实时性设备控制，例如多个伺服的位置同步控制；
- 心跳帧，用于监视 CANlink 网络各从站的通信状态，便于对控制系统的异常状态作及时响应理，避免造成更大损失。

A.5 EtherNET 端口及通信协议

EtherNET 端口主要有两个用途 : 第一个用途与 Mini-USB 口一样，用于下载 PLC 用户程序、进行监控调试；第二个用途是以太网通信，包括标准的 TCP/IP Modbus 通信和自由通信。Modbus 协议通过后台直接配置相关通讯功能码和相应的地址映寄存器，用户程序中访问寄存器值即可与远端的 Modbus 设备进行数据交互。自由通信协议只能通过操作标准的 socket 功能块实现数据交互。

A.6 EtherCAT 端口及通信协议

EtherCAT 口主要用途用户标准的 EtherCAT 协议通讯，线性拓扑结构，全双工通讯，波特率 1Mbit/s ，从站节点之间通讯距离最大支持 100 米。主站支持同步事件、DC 模式，AM600 机型任务最大抖动在 120us（典型值）。

A.7 高速 I/O 接口

高速 I/O 接口包含高速脉冲控制、高速脉冲计数功能。

- 高速脉冲控制用于控制脉冲式伺服驱动器、步进驱动器控制等设备；
- 高速脉冲计数用于 AB 相、单相、CW/CCW 等脉冲信号频率和计数采集。

A.8 Mini-SD 卡插槽

Mini-SD 插槽主要用于 PLC 底层固件升级，不对外开放。

A.9 本地总线扩展接口

本地总线扩展接口实现 PLC 直接连接 IO 模块功能。PLC 通过内部总线周期刷新 IO 模块的数据在 PLC 的映射地址。

用户仅需访问映射地址就可以对 IO 模块进行操作。

A.10 Profibus-DP 端口

Profibus-DP 端口和 CAN 端口集成在同一个 DB9 硬件端口上，目前 DP 功能仅在 AM610 中使用，在其他产品保留未使用。

附录 B 软元件概述

软元件是 AM600 编程系统预定义全局变量，用户编程时可以直接使用，不需要定义。软元件是直接变量，映射到 M 区（%M），并且具有 RETAIN 特性（掉电保存特性）。AM600 编程系统包含两种软元件：SD 软元件和 SM 软元件。SD 软元件是 INT 类型全局直接变量，SM 为 BOOL 类型全局直接变量。

M 区（%M）共 512KB, 前 480KB 为用户使用区域，后 32KB 时系统使用区域，用户不要直接使用地址。最后 32KB 中的前 30000Bytes 为 SD 和 SM 软元件使用区域，用于实现 CANlink、CANopen、高速 I/O 指令、MODBUS 等特殊功能，具体如下表，用户可以访问这些软元件。

SD 区间分配	功能	SM 区间分配	功能
0 - 7999	用户使用寄存器元件：0-7000 为 CANlink 使用（CANlink 配置，兼容小型 PLC）	0-7999	用户使用的位元件： 0 - 3071，8000-8511 为 CANlink 使用（CANlink 配置，兼容小型 PLC） 0-7999 为 MODBUS/MODBUSTCP 触发变量，从站使能变量使用。
8000 - 8999	系统使用的寄存器元件： CANlink/CANopen	8000 - 8999	系统使用的位元件：CANlink/CANopen
9000 - 9999	系统使用的寄存器元件：目前只有高速 I/O 使用	9000 - 9999	系统使用的位元件：目前只有高速 I/O 使用

NOTE

Modbus 触发变量在置位后，系统会自动复位，编程时请注意。

系统软元件，用户只可以读取不能写入，否则系统可能会出现异常。

关于软元件具体使用参见 CANlink 软元件和 MODBUS 及 MODBUSTCP 软元件说明。

附录 C 基本指令速查表

指令类型	指令说明	指令名	指令分类
算术运算指令	加法指令	ADD	函数
	乘法指令	MUL	函数
	减法指令	SUB	函数
	除法指令	DIV	函数
	取余指令	MOD	函数
数据处理指令	赋值指令	MOV	函数
	数据批量传送	BMOV	函数
	数据一对多传输	FMOV	函数
	获取数据指定位的状态	BON	函数
	ON 位总数	SUM	函数
	字节单位的数据结合	BTOW	函数
	字节单位的数据分类	WTOB	函数
	高低字节交换	SWAP	函数
	数据交换	XCH	函数
字逻辑指令	与指令	AND	函数
	或指令	OR	函数
	异或指令	XOR	函数
	取非指令	NOT	函数
位逻辑运算（CmpHCUtils）	上升沿输出	PLS	功能块
	下降沿输出	PLF	功能块
	交替输出	ALT	功能块
	位数据输出	BOUT	函数
	位数据置位	BSET	函数
	位数据复位	BRST	函数
移位指令	左移指令	SHL	函数
	右移指令	SHR	函数
	循环左移指令	ROL	函数
	循环右移指令	ROR	函数
	带进位的循环右移	RCR	函数
	带进位的循环左移	RCL	函数
	位数据向左拷贝	SFTL	函数
	位数据向右拷贝	SFTR	函数
	字数据向左拷贝	WSFL	函数
	字数据向右拷贝	WSFR	函数
	先进先出的数据读出	SFRD	函数
	先进先出的数据写入	SFWR	函数

指令类型	指令说明	指令名	指令分类
选择指令	二选一指令	SEL	函数
	取最大值指令	MAX	函数
	取最小值指令	MIN	函数
	极限值指令	LIMIT	函数
	多选一指令	MUX	函数
比较指令	大于指令	GT	函数
	小于指令	LT	函数
	大于等于指令	GE	函数
	小于等于指令	LE	函数
	等于指令	EQ	函数
	不等于指令	NE	函数
数学基本运算指令	绝对值指令	ABS	函数
	平方根指令	SQRT	函数
	自然对数指令	LN	函数
	常用对数指令	LOG	函数
	指数指令	EXP	函数
	正弦指令	SIN	函数
	余弦指令	COS	函数
	正切指令	TAN	函数
	反正弦指令	ASIN	函数
	反余弦指令	ACOS	函数
	反正切指令	ATAN	函数
	幂指令	EXPT	函数
	角度值转换成弧度值	RAD	函数
数学辅助运算指令（Util 库）	微分指令	DERIVATIVE	功能块
	积分指令	INTEGRAL	功能块
	整形统计指令	STATISTICS_INT	功能块
	实型统计指令	STATISTICS_REAL	功能块
	平方偏差	VARIANCE	功能块

指令类型	指令说明	指令名	指令分类
类型转换指令 【注】<TYPE> 数据类型有：BYTE、WORD、DWORD、LWORD、SINT、USINT、INT、UINT、DINT、UDINT、LINT、ULINT、REAL、LREAL、STRING、WSTRING、TIME、TIME_OF_DAY(TOD)、DATE、DATE_ADN_TIME(DT)	布尔类型转换指令	BOOL_TO_<TYPE>	函数
	字节类型转换指令	BYTE_TO_<TYPE>	函数
	日期类型转换指令	DATE_TO_<TYPE>	函数
	长整型转换指令	DINT_TO_<TYPE>	函数
	日期时间类型转换指令	DT_TO_<TYPE>	函数
	双字类型转换指令	DWORD_TO_<TYPE>	函数
	整数类型转换指令	INT_TO_<TYPE>	函数
	字类型转换指令	WORD_TO_<TYPE>	函数
	实数类型转换指令	REAL_TO_<TYPE>	函数
	短整型转换指令	SINT_TO_<TYPE>	函数
	字符类型转换指令	STRING_TO_<TYPE>	函数
	时钟类型转换指令	TIME_TO_<TYPE>	函数
	时间类型转换指令	TOD_TO_<TYPE>	函数
	无符号长整型转换指令	UDINT_TO_<TYPE>	函数
地址运算指令	取地址指令	ADR	函数
	取地址内容指令	^	函数
	位地址指令	BITADR	函数
	索引指令	INDEXOF	函数
	数据类型大小指令	SIZEOF	函数
调用指令	调用指令	CAL	函数
初始化操作指令	初始化操作指令	INI	函数
字符串处理指令（Standard 库）	取字符串长度指令	LEN	函数
	左边取字符串指令	LEFT	函数
	右边取字符串指令	RIGHT	函数
	中间取字符串指令	MID	函数
	合并字符串指令	CONCAT	函数
	插入字符串指令	INSERT	函数
	删除字符指令	DELETE	函数
	替换字符串指令	REPLACE	函数
	查找字符串指令	FIND	函数
双稳态指令（Standard 库）	置位优先双稳态器	SR	功能块
	复位优先双稳态器	RS	功能块
触发器指令（Standard 库）	上升沿检测触发器	R_TRIG	功能块
	下降沿检测触发器	F_TRIG	功能块
计数器（Standard 库）	递增计数器	CTU	功能块
	递减计数器	CTD	功能块
	递增递减计数器	CTUD	功能块

指令类型	指令说明	指令名	指令分类
定时器（Standard 库）	普通定时器	TP	功能块
	通电延时定时器	TON	功能块
	断电延时定时器	TOF	功能块
	实时时钟	RTC	功能块
BCD 转换指令（Util 库）	BCD 码转整形指令	BCD_TO_INT	函数
	整形转 BCD 码指令	INT_TO_BCD	函数
位 / 字节操作指令（Util 库）	位提取指令	EXTRACT	函数
	位整合指令	PACK	函数
	位拆分指令	UNPACK	功能块
	位赋值指令	PUTBIT	函数
控制器指令（Util 库）	比例微分控制器指令	PD	功能块
	比例积分微分控制器指令	PID	功能块
	比例积分微分控制器指令	PID_FIXCYCLE	功能块
信号发生器指令（Util 库）	脉冲信号发生器	BLINK	功能块
	周期性信号发生器	GEN	功能块
机器人操作指令（Util 库）	特征曲线指令	CHARCURVE	功能块
	整形限速指令	RAMP_INT	功能块
	实型限速指令	RAMP_REAL	功能块
模拟量处理指令（Util 库）	滞后指令	HYSTERESIS	功能块
	上下限报警指令	LIMITALARM	功能块
PLC 系统信息指令 (SysHCPlcInfo 库，具体指令使用详见帮助)	获取系统硬件相关信息	SysHC_HWInfo	功能块
	获取系统软件相关信息	SysHC_SWInfo	功能块
	获取 CPU 相关信息	SysHC_CPUInfo	功能块
	获取 CPU 相关故障诊断信息	SysHC_CPUDiagnose	功能块
	获取 ModbusRTU 从站设备故障诊断	SysHC_ModbusRtuDeviceDiagnose	功能块
	获取 ModbusRTU 主站访问从站故障诊断	SysHC_ModbusRtuSlaveDiagnose	功能块
	获取 ModbusTCP 从站设备故障诊断	SysHC_ModbusTcpDeviceDiagnose	功能块
	获取 ModbusTCP 主站访问从站故障诊断	SysHC_ModbusTcpSlaveDiagnose	功能块
	设置网络信息	SysHC_NetworkConfig	功能块
	获取网络信息	SysHC_NetworkInfo	功能块
	获取 U 盘路径信息	SysHC_UDiskPath	功能块
	获取 Boot 版本号	GetBootVersion	函数
	获取 PLC 版本号	GetPLCVersion	函数
	获取设备名	GetProductName	函数
	获取 Runtime 版本号	GetRuntimeVersion	函数
	获取 SN 号	GetSerialNumber	函数
	保存掉电保存信息	SysHC_SaveAllRetains	函数

指令类型	指令说明	指令名	指令分类
时间和日期（CmpHCUtils）	设置当前系统时钟	SetSystemDate	功能块
	获取当前系统时钟、时区	GetSystemDate	功能块
	获取系统运行时间，计时单位分别为毫秒、微妙、纳秒	GetSystemTime	功能块
表格和区间（CmpHCUtils）	死区控制指令	BZAND_TAB	函数
	数据平均值计算	MEAN_TAB	函数
	区域控制指令	ZONE_TAB	函数
	全部数据复位	ZRST_TAB	函数
	表格坐标获取	SCL_TAB	函数
	表格数据排序	SORT_TAB	函数
	斜坡指令	RAMP_TAB	功能块
	数据总和计算	WSUM_TAB	函数
通讯指令（CmpHCUtils）	创建 TCP 服务器端通信服务	TCP_Server	功能块
	创建 TCP 客户端通信服务	TCP_Client	功能块
	创建 TCP 连接，并连接到服务器	TCP_Connect	功能块
	TCP 通信数据接收	TCP_Recieve	功能块
	TCP 通信数据发送	TCP_Send	功能块
	创建 UDP 通信连接	UDP_Peer	功能块
	UDP 通信数据接	UDP_Receive	功能块
	UDP 通信数据发送	UDP_Send	功能块
滤波指令（CmpHCUtils）	限幅滤波	LimitingFilter	功能块
	中位值滤波	MedianFilter	功能块
	算术平均滤波	ArithmeticAverageFilter	功能块
	递推平均滤波	RecursiveAverageFilter	功能块
	中位值平均滤波	MedianAverageFilter	功能块
	限幅平均滤波	LimitingAverageFilter	功能块
	一阶滞后滤波	FirstOrderLagFilter	功能块
	加权递推平均滤波	WeightRecursiveAverageFilter	功能块
	消抖滤波	DebounceFilter	功能块
	限幅消抖滤波	LimitingDebounceFilter	功能块
	获取系统运行时间，计时单位分别为毫秒、微妙、纳秒	GetSystemTime	功能块
队列（CmpHCUtils）	先入先出队列	FIFO	功能块

附录 D PLC 编程软件升级指导

D.1 版本说明

1） 1 编程软件 InoProshop V1.1.0 及之前版本在持久性变量、硬盘分区、高速 I/O 功能、EtherCAT 总线 IO 模块等方面无法与新版本兼容，建议用户升级到最新版本。此外，本文描述中未涉及的版本（V1.3.2 以下）不推荐使用，如有特殊要求请联系本地供应商。

2） 2 从站设备文件，如 EtherCAT 描述文件（.xml）、CANopen 描述文件（.eds）、Profibus-DP 描述文件（.gds），应与从站设备固件版本相匹配，如有疑问，请联系本地供应商。V1.3.2 版本中默认未安装的从站设备，可通过安装相应的设备文件获得支持。

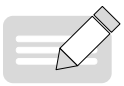
3） 3 如需了解 AM400/AM600/AC800 系列产品具体使用方法，请参见相关硬件手册或与供应商联系。

D.2 升级方法

■ 应用软件安装

电脑配置要求 Windows7 或 Windows10，内存不小于 4G，强烈推荐使用 64 位中英文操作系统（32 位系统不建议使用）。

请按照安装导向进行安装，或者根据需求在安装过程中对安装路径进行设置，默认安装路径为：C:\Inovance Control\InoProShop。


NOTE

不可与其他版本安装在同一个文件夹中。

■ 用户工程

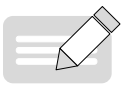
如果打开旧版本工程，会弹出“工程版本信息”窗口。若不希望更新工程，选择“不更新”即可直接编辑使用，但是梯形图必须更新。

显示“工程版本信息”窗口有两种方式：一种是旧工程打开时自动弹出、另一种是使用菜单命令【工程】-【工程版本信息…】打开。

有两种更新工程形式：全部更新和部分更新。

1） 全部更新，在弹出“工程版本信息”窗口时，选择“全部设置成最新版本”按钮，然后点击“确定”按钮。

2） 部分更新，在弹出“工程版本信息”窗口时，选择需要更新的类型对应的选项卡，然后点击“确定”按钮。

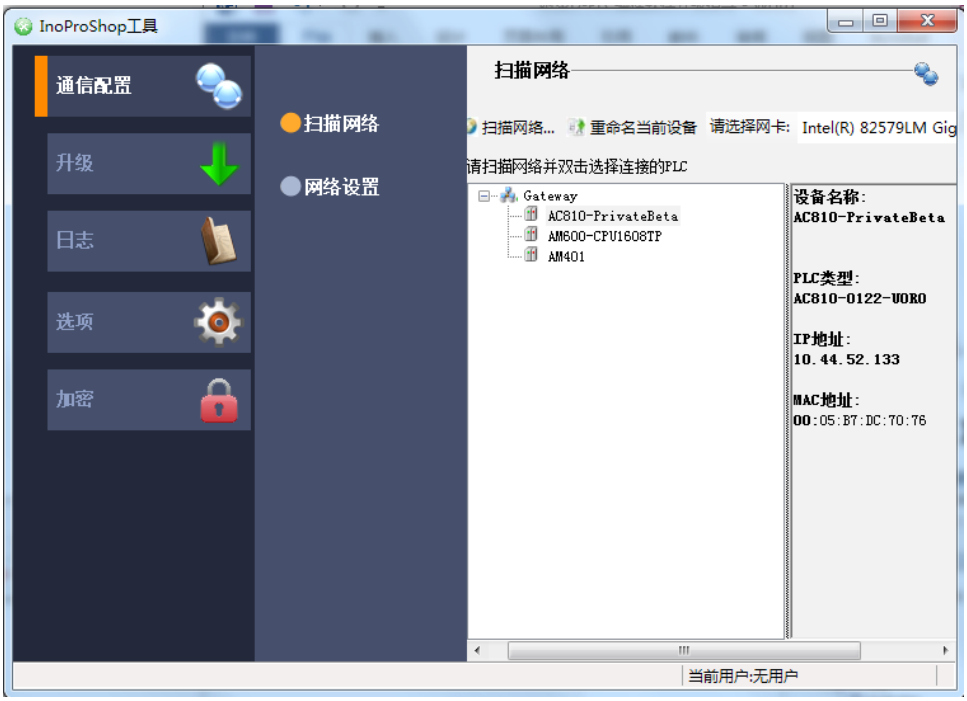

NOTE

梯形图必须更新。

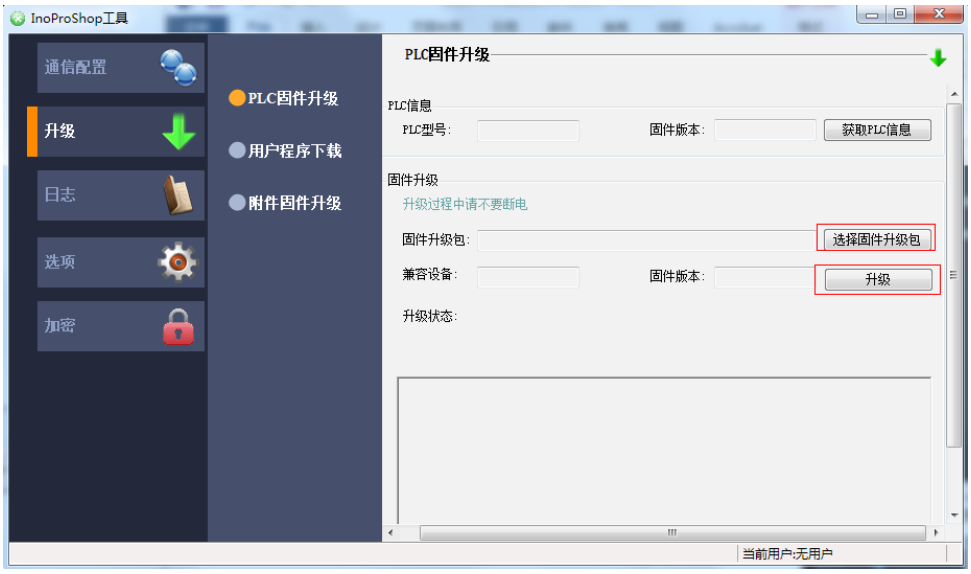
■ 在线固件升级

PLC 设备（CPU 模块）升级

步骤 1 【工具】-【InoProshop 工具】-【扫描网络】- 选中相应设备



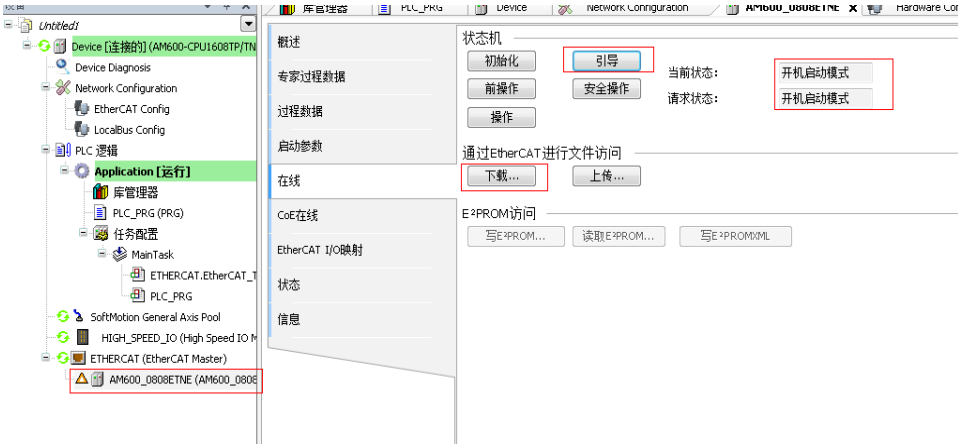
步骤 2 【升级】-【选择固件升级包】-【升级】



■ EtherCAT 模块升级

步骤 1：点击【设备】-【概述】，勾选“启动专家设置”，然后点击【登录下载】-【运行】。

步骤 2：点击【设备】-【在线】，在【状态机】中选择“引导”。进入引导状态后，选择【通过 EtherCAT 进行文件访问】-【下载】。在弹出的框中，找到要升级的后缀名为“.bin”固件文件路径，选中固件文件启动升级。



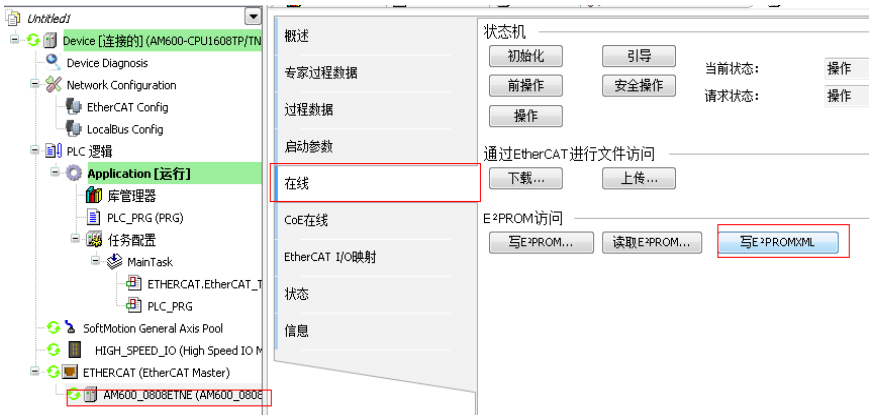
■ 库升级

参考下文“常见问题”中的“如何在工程添加编译库”。

■ EtherCAT 设备文件升级

步骤 1：点击【设备】-【概述】，勾选“启动专家设置”，选择【登录下载】-【运行】。

步骤 2：点击【设备】-【在线】-【写 EEPROMXML】，在弹出的框中，找到要升级的 XML 文件的路径，选中该文件后启动升级。



D.3 常见问题

1如何查看版本

点击【设备】-【升级】-【获取 PLC 信息】

2目标系统与连接设备不匹配

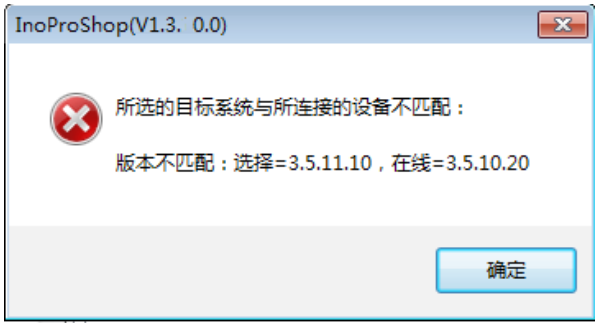


图 D-1 目标系统与连接设备不匹配

原因: 出现此错误的原因是 InoProShop 的 PLC 设备版本为“3.5.11.10”，而实际 PLC 设备版本为“3.5.10.20”。InoProShop 上的设备版本不能大于实际版本!

■ 解决方案 1：更新 PLC 固件，升级到与设备版本 (3.5.11.10) 匹配的固件版本。

第 1 步：右键单击“设备”，选择“更新设备”，在弹出的窗口勾选“显示所有版本”，找到更新设备的对应版本。如果在设备列表中找不到与实际硬件相同的版本，选择前三个数字一样的版本即可。

如下图所示，列表中不存在与硬件设备相同的“3.5.10.20”，可以选择“3.5.10.40”（前面的三个数字版本相同）并更新设备。

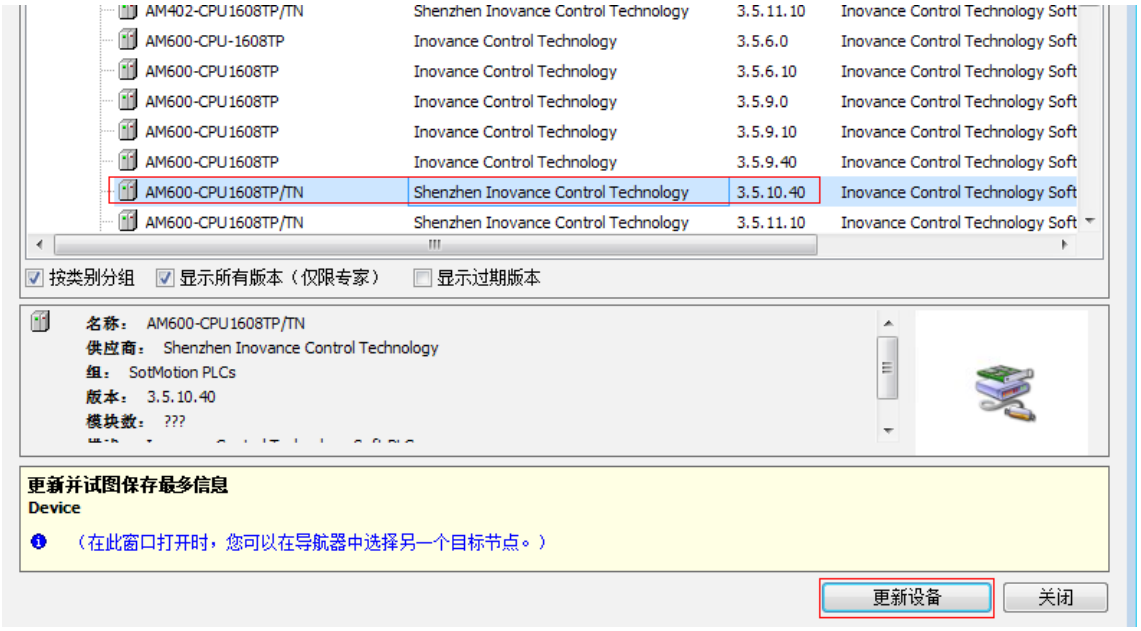


图 D-2 更新 PLC 设备版本

第 2 步：重新扫描并选中相应设备，系统将不会弹出错误窗口。如下图：

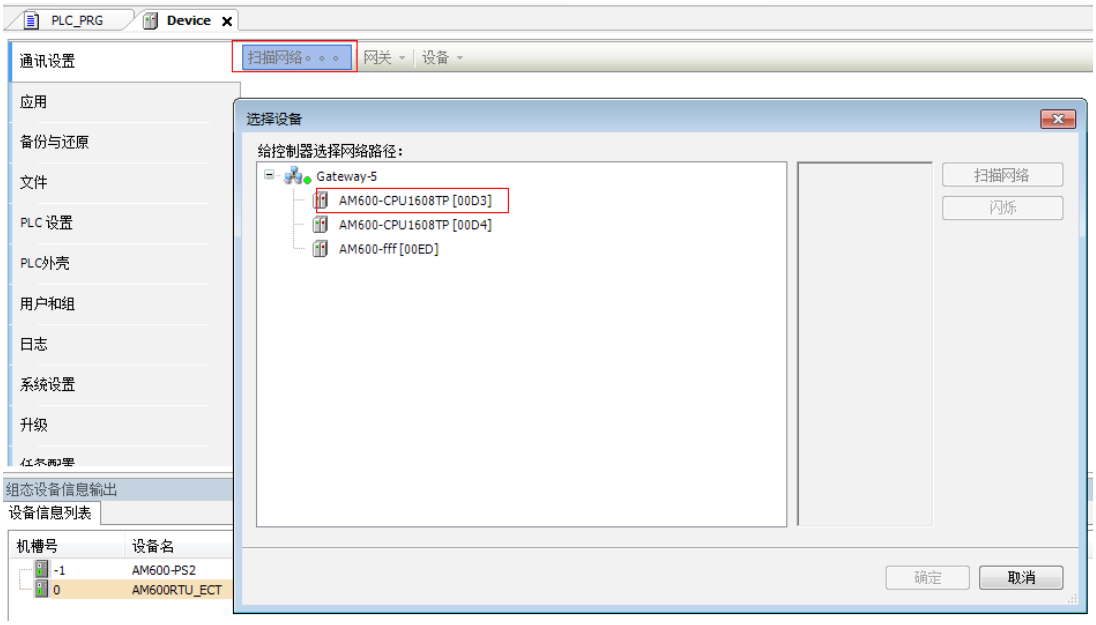


图 D-3 连接 PLC 设备

第 3 步：选择“设备”，找到“升级选项”，在固件升级界面在线升级。

图 D-4 在线升级固件

第 4 步：固件升级完毕后，再按照步骤 1 将 PLC 设备版本更新到版本“3.5.11.10”，即可使用最新版本的 PLC 设备和 PLC 固件。

■ **解决方案 2**：更新设备版本，降低到与固件匹配的设备版本。

执行解决方案 1 中的第 1 步即可达到目的，但低版本的 PLC 设备文件只能跟该版本配套 IEC 库使用。

工程中添加 IEC 库时，由于 IEC 库添加的规则是默认最新版本，编译程序时可能出现编译报错，原因是库版本与 PLC 设备版本匹配不一致，可以通过手动更改 IEC 库版本解决。

3 新编程软件版本中添加库时编译报错

V1.3.2 打开 V1.3.0 以前版本上的工程（以 V1.2.0 为例），添加 IEC 库“CmpBasic”，并使用库里面的“MC_ResetDrive”功能块时，结果编译报错。

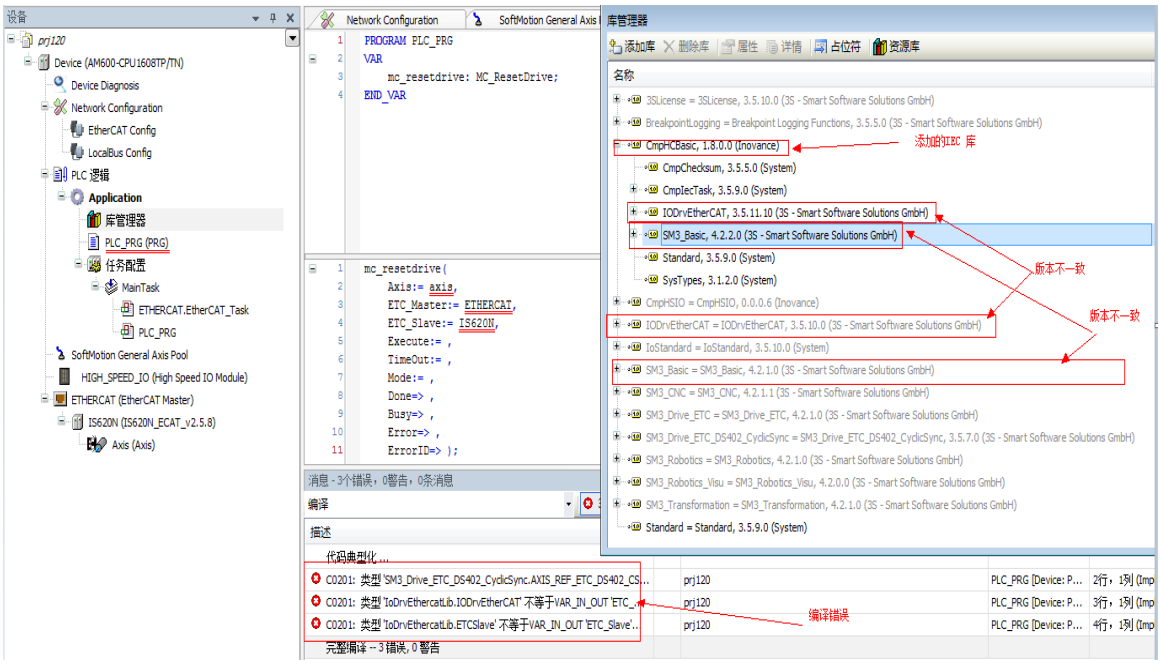


图 D-5 新添加库编译报错

原因：CmpBasic 库内依赖的特定库 (SM3_Basic、IODrvEtherCAT) 与工程库管理中依赖的特定库版本不统一。CmpBasic(V1.8.0.0) 版本依赖的“IODrvEtherCAT”版本为“V3.5.11.10”，而工程引用的版本是“V3.5.10.0”；CmpBasic(V1.8.0.0) 版本依赖的“SM3_Basic”版本为“V4.2.2.0”，而工程引用的版本是“V4.2.1.0”。

解决方案：

第 1 步：双击“库管理器”打开“库管理器”界面，在库列表中选“CmpBasis”库，此时版本为“1.8.0.0”。

第 2 步：在“库管理器”中选择“属性”选项，在弹出的窗口内找到“版本”选项卡，在下拉菜单中选择与 PLC 设备对应的 IEC 库版本为 1.6.0.0(V1.2.0 新建的工程)，然后点击“确定”。

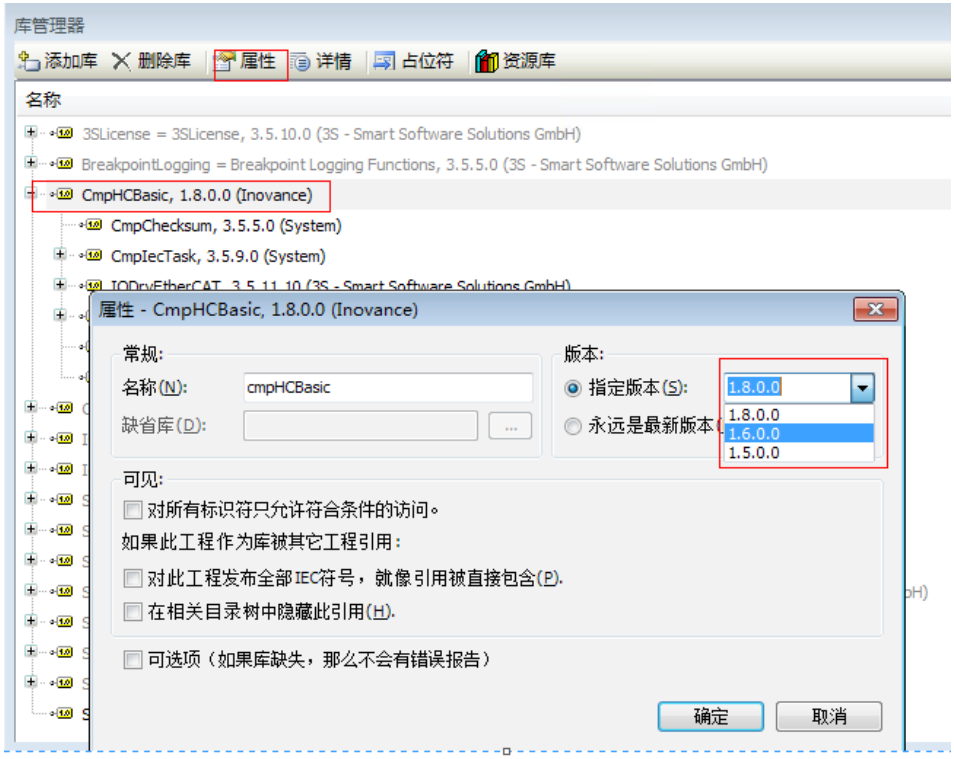


图 D-6 手动更新 IEC 库

第 3 步：编译工程，如下图所示：

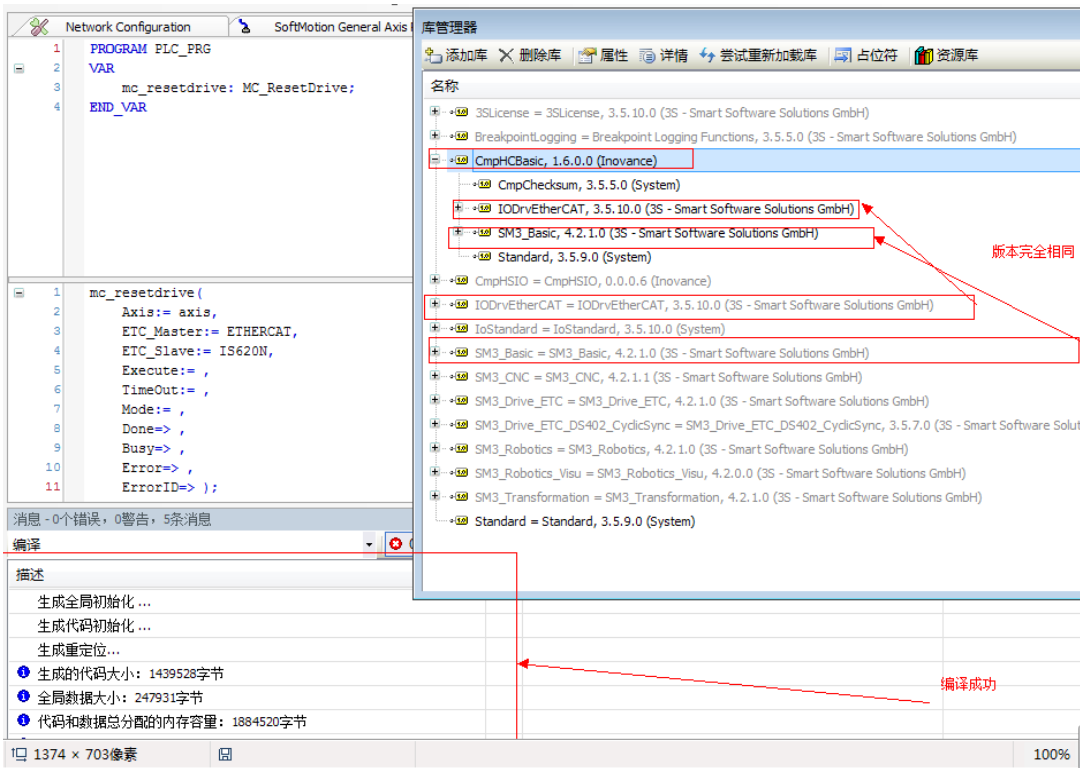


图 D-7 更新后的 IEC 库、编译信息

NOTE

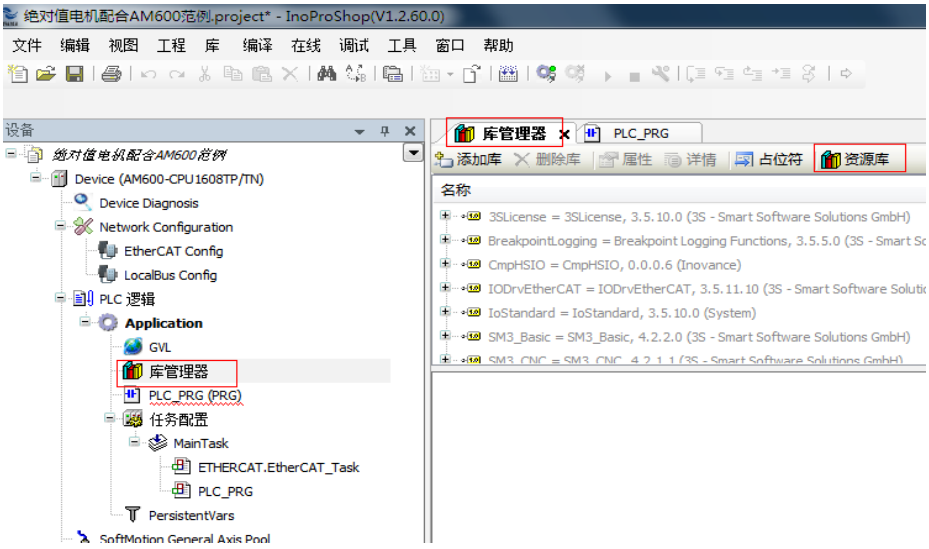
库不兼容的主要原因是使用库内依赖系统库与前工程中包含的系统库版本不一致。常见库有 IODrvEtherCAT 库、SM3_Basic 库。

4 如何在工程添加编译库

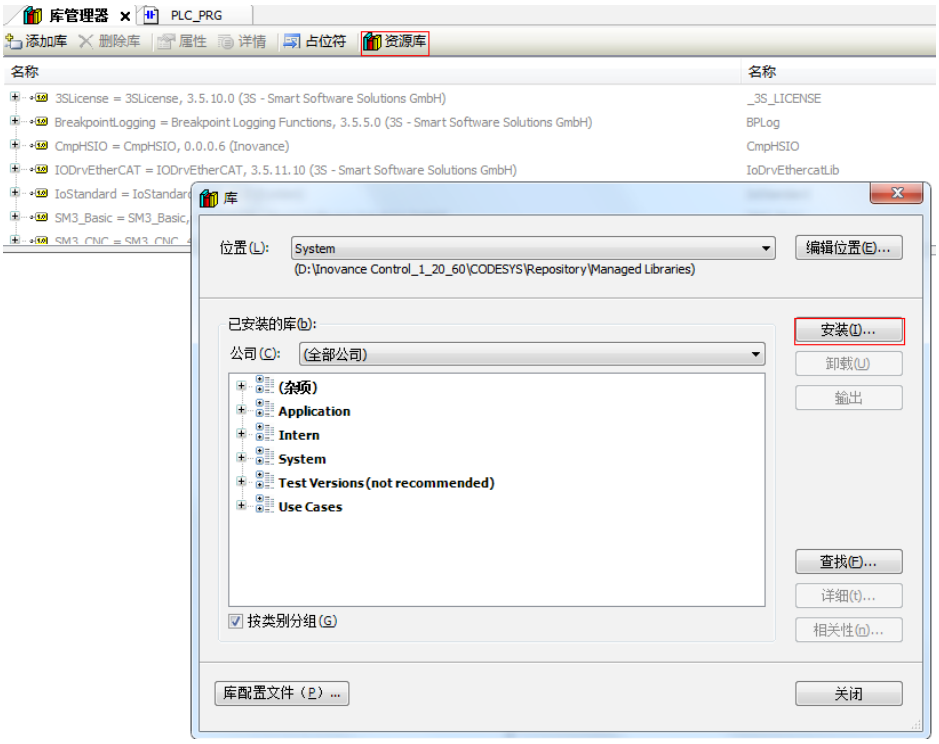
安装库以“CmpBasic. compiled-library”版本 V1.11.0.0 为例，后台软件版本以 V1.2.60 为例。其他后台软件版本操作步骤与范例相同。

第 1 步：安装编译库

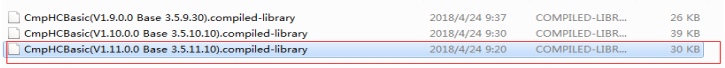
打开工程，点击“库管理器”。



在“库管理器”界面，点击“资源库”，点击“安装”。

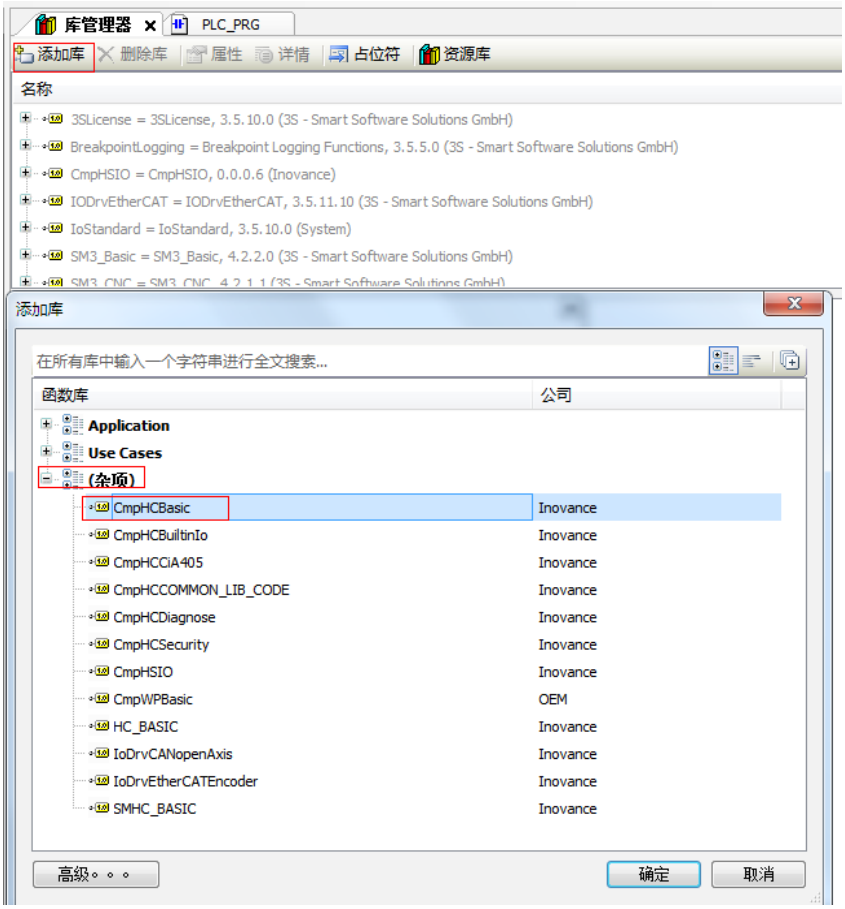


找到需要安装的编译库存储路径（CmpBasic.compiled-library V1.11.0.0），选中并“打开”。

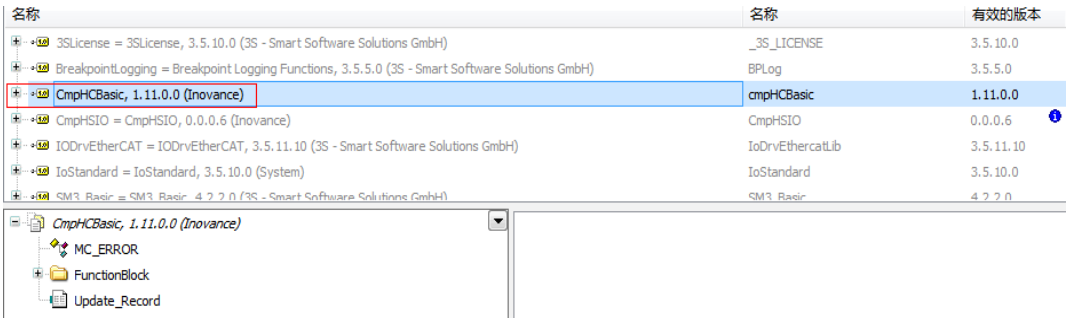


第 2 步：工程添加库

在“库管理器”界面，点击“添加库”，在“库”弹出界面选择“杂项”，点开“+”。

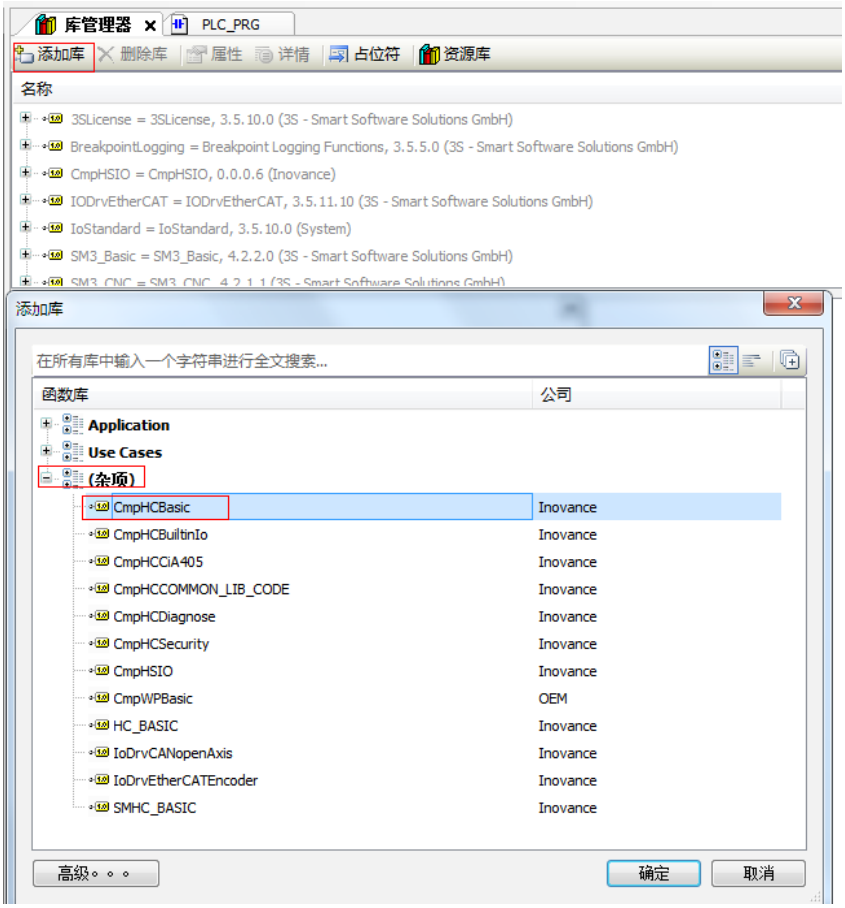


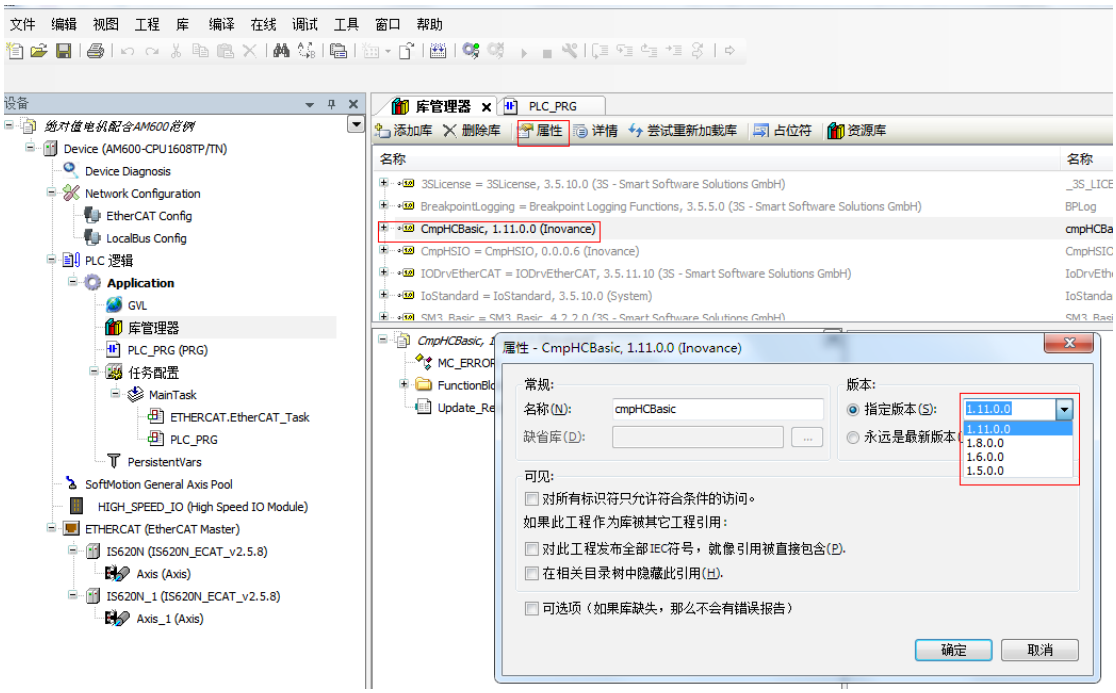
选择对应的库“CmpHCBasic”并“确定”。默认将最高的版本添加到工程。



第 3 步：手动选择库版本

在“库管理器”界面选中需要更新库“CmpBasic”。点击“属性”，弹出“CmpBasic”库的属性窗口。在版本选项的下拉选择框，可以指定任意版本（注意库版本与后台的匹配关系，不匹配的情况使用库里面的功能会出现编译报错）。





NOTE

1) 工程中添加或更新库时，必须在菜单栏选择【编译】-【清除全部】再编译工程。
2) 工程中添加或更新库，编译无错误后，登录必须重新下载（在线下载可能导致 PLC 异常）。

附录 E AM400/600 高速 I/O 接线指导

AM400/600/610 CPU 模块支持高速 I/O 数据处理功能，自带一个高密度端口，具有 16 路高速输入，其中前 6 路支持 24V 单端输入或差分输入，后 10 路支持 24V 单端输入。本文档对高速 I/O 信号接口及转接使用进行说明指导。

高密度端口（端口丝印：CN5）示意图如下：

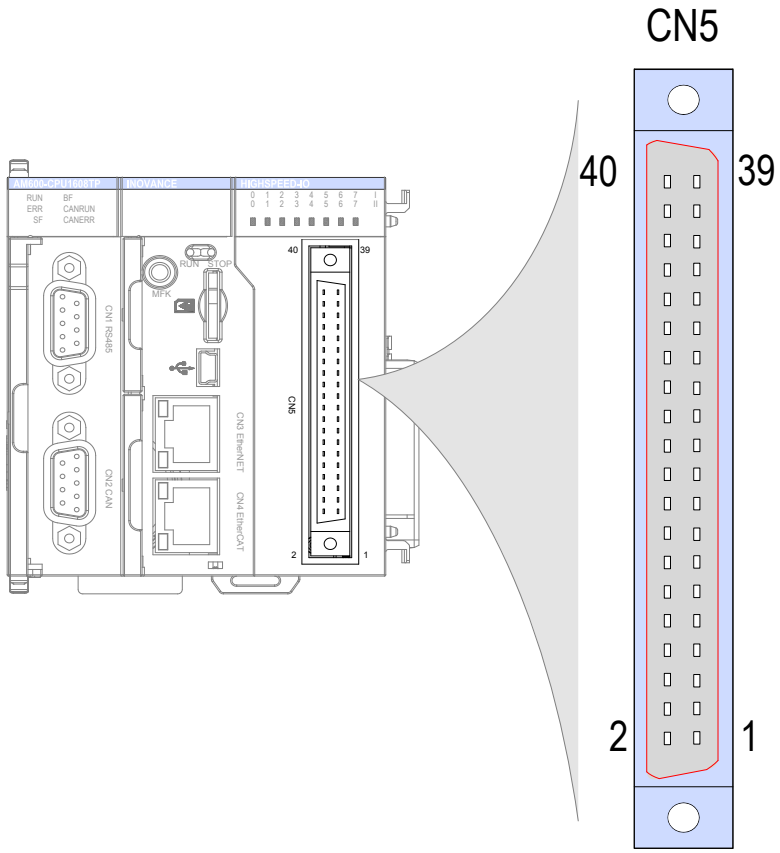
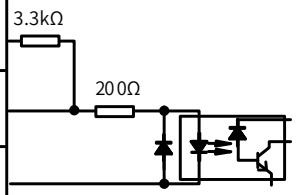
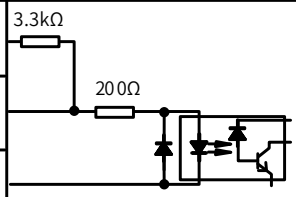
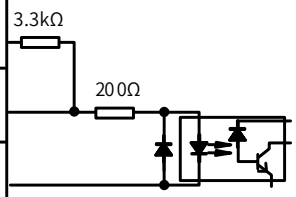
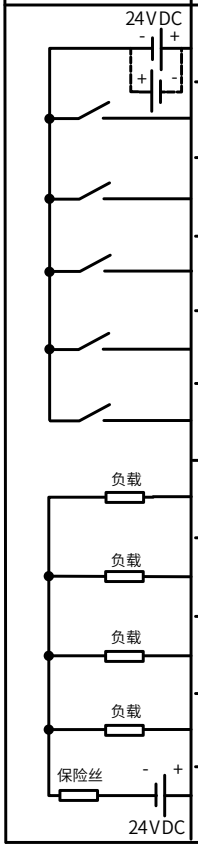
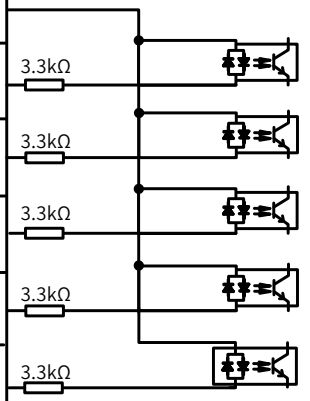
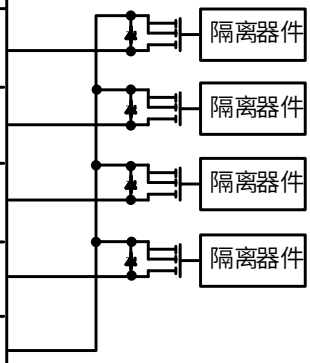


图 1 AM600/610 CPU 模块高密度端口示意图

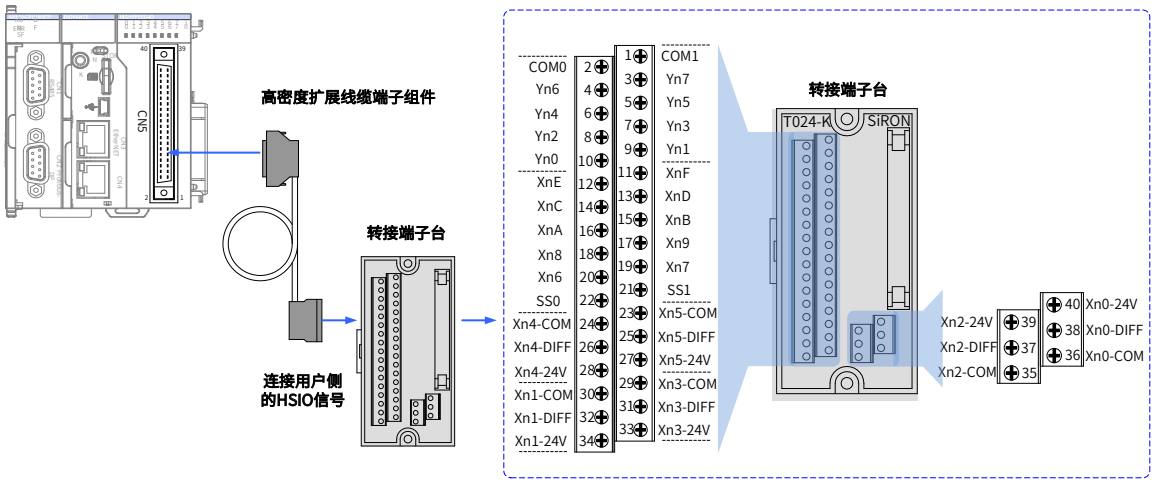
端口的内部电路及外部配线要求如下，请根据实际需求完成相应配线：

外部配线	信号名称 B列	CN5针编号		信号名称 A列	内部电路
参照应用举例	高速24V输入（Xn0-24V）	40	39	高速24V输入（Xn2-24V）	
	高速差动输入（Xn0-DIFF）	38	37	高速差动输入（Xn2-DIFF）	
	高速输入公共端（Xn0-COM）	36	35	高速输入公共端（Xn2-COM）	
参照应用举例	高速24V输入（Xn1-24V）	34	33	高速24V输入（Xn3-24V）	
	高速差动输入（Xn1-DIFF）	32	31	高速差动输入（Xn3-DIFF）	
	高速输入公共端（Xn1-COM）	30	29	高速输入公共端（Xn3-COM）	
参照应用举例	高速24V输入（Xn4-24V）	28	27	高速24V输入（Xn5-24V）	
	高速差动输入（Xn4-DIFF）	26	25	高速差动输入（Xn5-DIFF）	
	高速输入公共端（Xn4-COM）	24	23	高速输入公共端（Xn5-COM）	
	输入公共端（SS0）	22	21	输入公共端（SS1）	
	标准输入（Xn6）	20	19	标准输入（Xn7）	
	标准输入（Xn8）	18	17	标准输入（Xn9）	
	标准输入（XnA）	16	15	标准输入（XnB）	
	标准输入（XnC）	14	13	标准输入（XnD）	
	标准输入（XnE）	12	11	标准输入（XnF）	
	输出（Yn0）	10	9	输出（Yn1）	
	输出（Yn2）	8	7	输出（Yn3）	
	输出（Yn4）	6	5	输出（Yn5）	
	输出（Yn6）	4	3	输出（Yn7）	
	输出公共端（COM0）	2	1	输出公共端（COM1）	

NOTE

前 6 路高速 DI 接线使用请参见下文使用案例进行接线使用，避免出现错误接线。

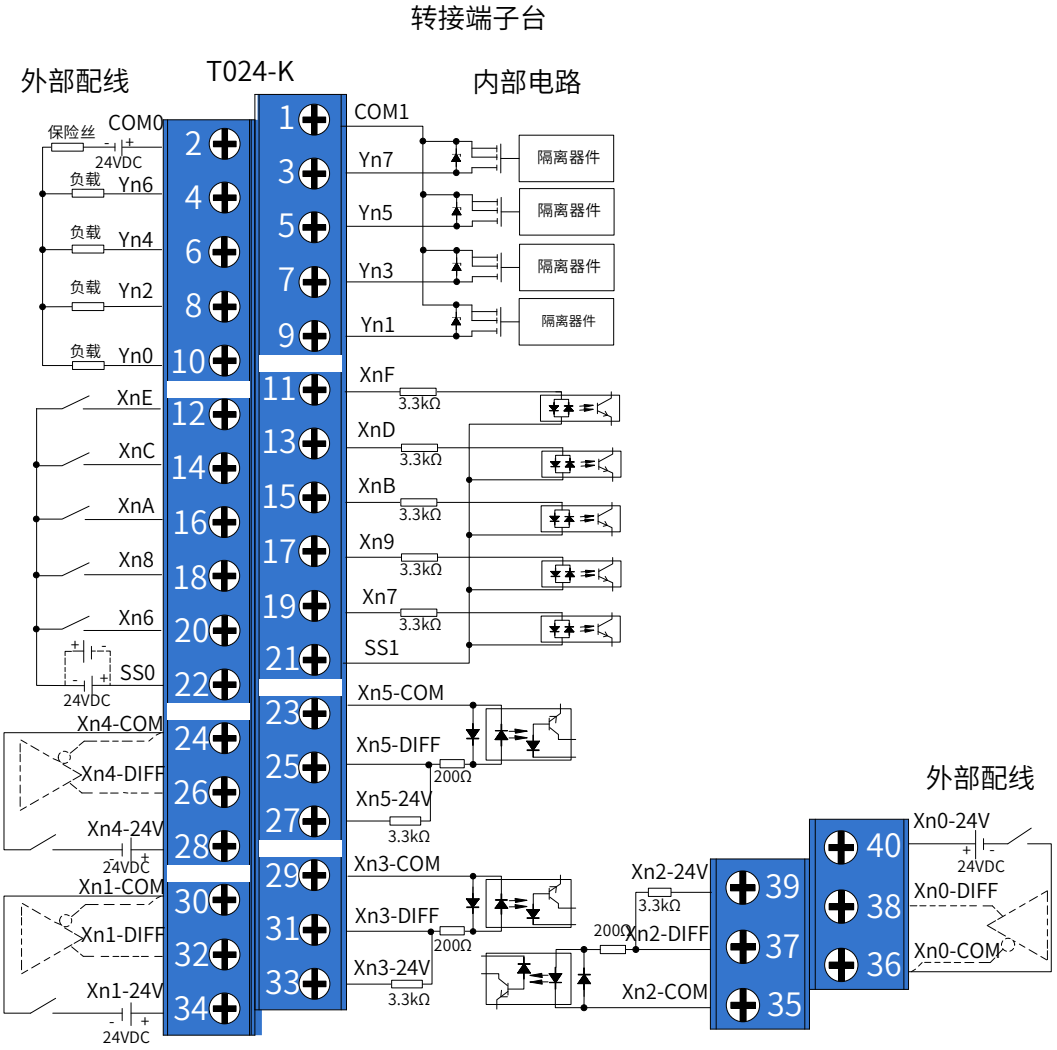
如果采用 SIRON T024-K 转接端子进行转接接线，端子序号与模块 CN5 针编号对应如下图：



其中如上图所示，汇川公司提供“①高密度扩展线缆、②连接插头（自制线缆时需要）、③转接端子台”可供选购，订货信息如下：

序号	订货编码	描述
①	15300119	40PIN FCN 转 MIL 高密度扩展线缆（500mm，含两个 40PIN FCN 连接插头）
②	15050180	40PIN FCN 连接插头（如不选购上述线缆，用户可选购此插头后自制线缆）
③	15020452	40PIN MIL 转螺钉接线端子台

SIRON T024-K 转接端子台配线示意如下：

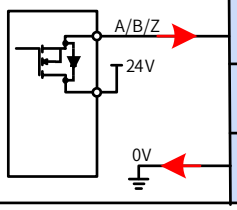
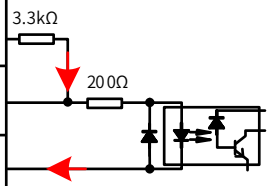


NOTE

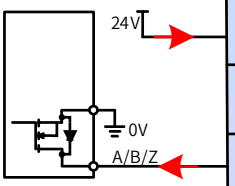
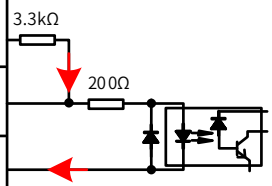
以上为 CPU 模块高密度端口针脚定义及接线说明，请详细阅读后再进行配线操作。

■ 应用举例

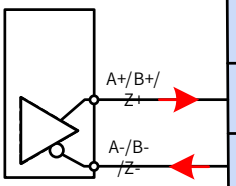
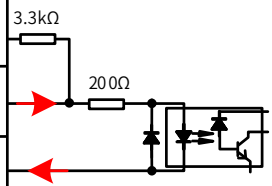
高速 I/O 前 4 路 DI 支持高速单端及差分信号，使用时需注意接线正确，以 Xn0 为例进行应用举例说明如下。（1）当输入为 PNP 型，24V 电平输入时；

输入类型	外部配线	编号	信号名称	内部电路
PNP集电极型（24V电平）		40	高速24V输入（Xn0-24V）	
		38	高速差 动输入（Xn0-DIFF）	
		36	高速输 入公共端（Xn0-COM）	

（2）当输入为 NPN 型，24V 电平输入时；

输入类型	外部配线	编号	信号名称	内部电路
NPN集电极型（24V电平）		40	高速24V输入（Xn0-24V）	
		38	高速差 动输入（Xn0-DIFF）	
		36	高速输 入公共端（Xn0-COM）	

（3）当输入为差分信号，5V 电平输入时；

输入类型	外部配线	编号	信号名称	内部电路
差分信号（5V电平）		40	高速24V输入（Xn0-24V）	
		38	高速差 动输入（Xn0-DIFF）	
		36	高速输 入公共端（Xn0-COM）	

附录 F 高速 I/O 兼容性

F.1 高速 I/O 新旧界面介绍

CmpHCBuiltinIo：旧高速 I/O 功能块库

CmpHSIO：新高速 I/O 功能块库

InProShop 版本 V1.2.0（临时版本中 1.1.60.0）、AM600 固件版本 V1.19.70.0、FPGA 版本 A624 以上支持高速 I/O 功能块的新库。下图为新旧高速 I/O 设备对应的软件界面：

高速 I/O 旧版本界面

高速 I/O 新版本界面

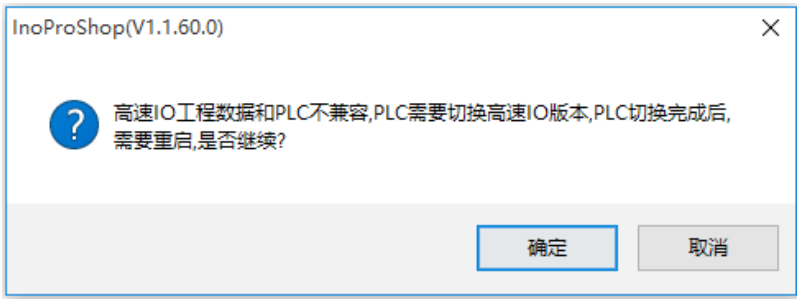
高速 I/O 新版本新界面

说明：

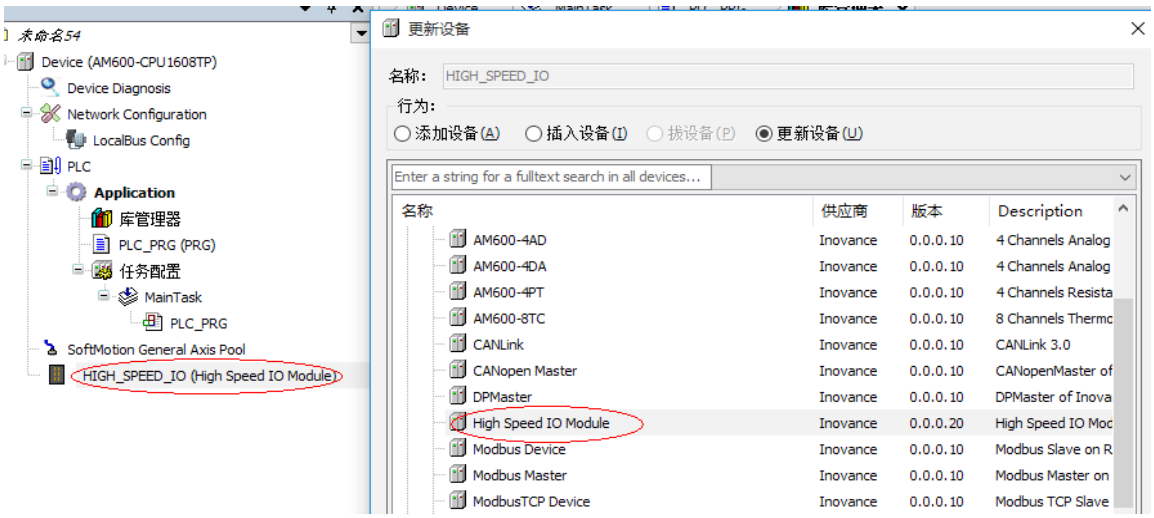
1） 新高速 I/O 设备和新高速 I/O 库匹配、旧高速 I/O 设备和旧高速库匹配，两者不能混用

2） 新高速 I/O 功能需要新版本 PLC 固件和 FPGA 同时升级。

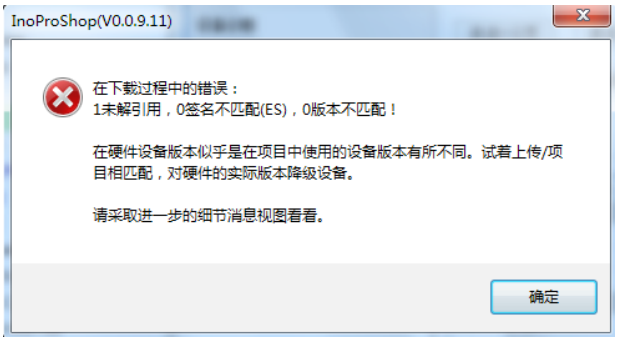
3） 新高速和旧高速 I/O 功能可以相互切换（回原功能不兼容），如果新高速 I/O 和旧高速 I/O 混用（新高速 I/O 工程和旧 PLC 高速 I/O、旧高速 I/O 工程和新 PLC 高速 I/O），会提示切换 PLC，登录时提示如下图所示，需 PLC 切换高速 I/O 版本，重新上电才生效；



4） 如果不想切换 PLC，可以切换工程的高速 I/O 版本，使用【更新设备】命令，如下图：旧高速 I/O 设备版本为 0.0.0.10，新高速 I/O 设备对应版本为 0.0.0.20.



5） 在旧版本后台（如 1.1.0 或者 0.0.9.10 版本）中，将旧高速 I/O 工程下载到新固件版本的 PLC（1.19.70.0 以上版本）中时会提示如下错误：



解决策施：安装最新后台，使用新后台打开旧工程（未使用高速 I/O 功能）后下载到新固件版本的 PLC 中时，不会有提示，可以正常下载使用。

F.2 高速 I/O 诊断

1 新版本高速 I/O

基本格式：

库 + 功能块 + 错误码

3 3 位

- 库：高速 I/O 默认是 0
- 功能块号码：从 01 开始，功能块列表详见 6.2.4. 功能块列表
- 错误码：从 01 开始，错误码详见表“计数器错误码列表”。错误码小于 500 是严重错误，大于 500 是功能块错误。比如：14506，14 指的是 HC_WriteParameter，506 指设置参数错误；31520，31 指的是 MC_WriteParameter_P，520 指设置参数错误。

表 E-2 计数器错误码列表：

错误码	定义	说明
001	ERR_COUNTERID_INVALID	输入通道号无效，有效范围是 [0,7]
003	ERR_CNT_OVERFLOW	计数器上溢 / 下溢出错
004	ERR_COUNTER_NOT_CHOSEN	高速功能未选中，在后台配置中选择
007	ERR_COUNTER_NOT_ENABLED	计数器 HC_Counter 未使能
101	ERR_WRITEINTERRUPTPARAMETER_UNVALIAD	写中断参数无效
102	ERR_INTERRUPT_NOT_CHOSE	未在后台选择“中断输入”
501	ERR_SETCOMPARE_IMREFRESHCYCLE_OVERFLOW	比较值 ImRefreshCycle 超过 30000，有效范围是 [0,30000]
502	ERR_SETCOMPAREM_NUMBERS_OVERFLOW	HC_SetCompareM 的 Number 范围是 [1,100]
503	ERR_PREWR_VALUE_OVERFLOW	预置值超限
504	ERR_AVERAGE_PARA_UNVALIAD	频率和旋转速度设置平均参数无效
505	ERR_ROTATION_PULSES_UNIT_UNVALIAD	转速测量的每转脉冲数设置无效
506	ERR_WRITEBOOlPARAMETER_UNVALIAD	设置参数无效，HC_WriteBoolParameter
507	ERR_READBOOLPARAMETER_UNVALIAD	获取参数无效 HC_ReadBoolParameter
508	ERR_MEASURE_WIDTH_OVERFLOW	测量宽度无效
509	ERR_SETCOMPAREM_IMREFRESHCYCLE_OVERFLOW	比较值 ImRefreshCycle 超过 30000，有效范围是 [0,30000]
510	ERR_PRESET_TRIGGERTYPE_OVERFLOW	预置参数无效
511	ERR_WRITEPARAMETER_UNVALIAD	设置参数无效，HC_WriteParameter
513	ERR_FUNC_COUNTERID_INVALID	特殊功能通道号无效 , 有效范围是 [0,3]
514	ERR_COUNTER_NOT_CHOSE_EXETERNAL_X	计数器后台没有选择“外部触发输入”
515	ERR_CNT_FORMAT_NOT_RING	环形计数类型不对，在后台配置中选择
516	ERR_RING_DOWNVAL_BEYOND_UPVAL	环形计数下限值≥上限值
517	ERR_SAMPLE_VALUE_LESS	采样时间太小，采样时间范围是 10~65535(10ms~65535ms)
518	ERR_RING_VALUE_OVERFLOW	环形计数超限

表 E-3 高速轴错误码列表：

错误码	定义	说明
001	ERR_NOT_POWER	MC_Power 没使能
002	ERR_UP_SOFTWARE_LIMIT	当前位置超出软件行程限制 +
003	ERR_DOWN_SOFTWARE_LIMIT	当前位置超出软件行程限制 -
004	ERR_AXIS_FUNC_UNUSED	高速轴功能未使能，在后台使能
005	ERR_INPUT_CHANNAL_NUM_INVALID	轴号无效，有效范围是 [0,3]
006	ERR_DEST_POS_OVER_SOFT_UP_LIMIT	目标位置超出软件上限

错误码	定义	说明
007	ERR_DEST_POS_OVER_SOFT_DOWN_LIMIT	目标位置超出软件下限
010	ERR_POS_DECPOINT_OVERLOW	减速点无效：位置模式下，重新定位时，减速长度大于实际距离
011	ERR_VEL_DECPOINT_OVERLOW	减速点无效：速度模式下，切换为定位时，减速长度大于实际距离
012	ERR_POS_PLSNUM_OVERLOW	超过 PLSNUM 最大定位长度 2147483647
013	ERR_POS_DECPOINT2_OVERLOW	2 次求减速点出错。
501	ERR_ACC_SET_OVERFLOW	加速度超出允许范围，超出 MC_WriteParameter_P 设定的最大加速度
502	ERR_ACC_SET_LOW	加速度设置太小，低于 MC_WriteParameter_P 设定的最小加速度
503	ERR_DEC_SET_OVERFLOW	减速度超出允许范围，超出 MC_WriteParameter_P 设定的最大减速度
504	ERR_DEC_SET_LOW	减速度设置太小，低于 MC_WriteParameter_P 设定的最小减速度
505	ERR_VEL_SET_OVERFLOW	速度设置超出允许范围，后台或者 MC_WriteParameter_P 设置
506	ERR_VEL_SET_LOW	速度设置太小
508	ERR_VEL_LESS_THAN_STARTVEL	速度小于启动偏置速度，后台设置启动偏置速度
509	ERR_STARTVEL_SET_LOW	起始速度太小
510	ERR_FBD_MOVEMODE_INVALIAD	功能块的运动模式无效，状态不对
511	ERR_WASNT_STANDSTILL	当前状态不是 STANDSTILL
512	ERR_WASNT_DISABLED	当前状态不是 DISABLE
513	ERR_IN_ERRORSTOP	当前状态是 ERRORSTOP
514	ERR_NOT_READY_FOR_MOTION	轴没准备好，不能运行
515	ERR_INVLALID_VELOCITY_MODE	无效的速度模式
516	ERR_INVLALID_POSTION_MODE	无效的位置模式
520	ERR_AXIS_WRITEPARAMETER_UNVALIAD	MC_WriteParameter_P 参数无效
521	ERR_AXIS_READPARAMETER_UNVALIAD	MC_ReadParameter_P 参数无效
522	ERR_HOME_MODE_UNVALIAD	回原模式无效，后台选择
523	ERR_AXIS_WRITEPARAMETER_HOME_MODE_UNVALIAD	回原模式设置无效

错误分为：轴错误和功能块错误

置为 ErrorStop 状态的条件为：

1） 发生轴错误；

2） 在 DiscreteMotion、ContinuousMotion 和 Homing 状态时发生功能块错误。

（2）旧版本高速 I/O

高速 I/O 自身诊断界面显示高速 I/O 诊断信息，关于自身诊断界面的描述，请参考设备自身诊断信息列表简介。

高速 I/O 诊断主要通过获取高速 I/O 诊断软元件获得的。高速 I/O 诊断包括通道出错、通道报警、轴出错、轴报警和其它故障五类。通道出错、通道报警、轴出错、轴报警使用软元件表示诊断状态和诊断码。诊断状态表示此是否出现诊断信息，诊断码表示出现错误的代码。每种类型对应的软元件表、诊断码诊断信息表如下：

■ 通道出错

通道号和错误标志软元件、错误诊断码软元件关系如下表：

通道号	出错标志软元件	出错诊断码软元件
0	SM9030	SD9007
1	SM9080	SD9017
2	SM9130	SD9027

通道号	出错标志软元件	出错诊断码软元件
3	SM9180	SD9037
4	SM9230	SD9047
5	SM9380	SD9057
6	SM9330	SD9067
7	SM9380	SD9077

诊断码和诊断信息对应关系如下表：

诊断码	诊断信息
1001	通道类型不匹配
1002	计数器溢出
1003	脉冲宽度测量溢出
1011	环形计数器下限值超过上限值
1012	计数器类型不匹配
1013	高速计数功能未使用
1014	高速计数器功能选择不匹配
1015	预置值超限
1016	平均参数无效
1017	转速测量的每转脉冲数设置无效

■ 通道报警

通道号和报警标志软元件、报警诊断码软元件关系如下表：

通道号	报警标志软元件	报警诊断码软元件
0	SM9031	SD9008
1	SM9081	SD9018
2	SM9131	SD9028
3	SM9181	SD9038
4	SM9231	SD9048
5	SM9381	SD9058
6	SM9331	SD9068
7	SM9381	SD9078

诊断码和诊断信息对应关系如下表：

诊断码	诊断信息
1501	采样值上溢

■ 轴出错

轴号和错误标志软元件、错误诊断码软元件关系如下表：

轴号	出错标志软元件	出错诊断码软元件
0	SM9405	SD9105
1	SM9425	SD9125
2	SM9445	SD9145
3	SM9465	SD9165

诊断码和诊断信息对应关系如下表：

诊断码	诊断信息
2001	正方向硬件限位
2002	负方向硬件限位
2003	始动时停止命令 ON
2004	正向软件限位
2005	负向软件限位
2006	运行中 cpu 模块切换为 stop 状态
2007	驱动模块就绪 OFF
2008	零点信号 ON
2009	没有实施机械原点回归
2010	重试出错
2011	ABS 传送超时
2012	ABS 传送 sum
2013	速度 0 出错
2014	加减速时间设置超时
2015	减速停止时间设置超时
2016	速度位置切换控制中移动量设置超出了允许范围
2017	禁止速度位置切换启动
2018	非轴停止状态更改当前值
2019	加减速时间设置为 0
2020	始动时轴未停止
2021	启动时遇轴停止命令

■ 轴报警

轴号和报警标志软元件、报警诊断码软元件关系如下表：

轴号	报警标志软元件	报警诊断码软元件
0	SM9406	SD9106
1	SM9426	SD9126
2	SM9446	SD9146
3	SM9466	SD9166

诊断码和诊断信息对应关系如下表：

诊断码	诊断信息
2501	超出速度范围
2502	禁止目标位置更改
2503	禁止速度更改

■ 其它故障

其它故障当前定义了高速 I/O 功能块输入参数无效诊断。此诊断不能通过软元件获取，可以在高速 I/O 自身诊断界面和诊断信息列表界面查看。其诊断代码和诊断信息如下表：

诊断码	诊断信息
1018	高速输入功能块的通道号配置无效
1019	高速输入功能块的输入参数配置无效
2022	高速输出功能块的通道号配置无效
2023	高速输出功能块的输入参数配置无效

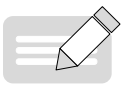
附录 G 诊断码和诊断信息

每个诊断码都有个名称，与诊断编程接口对应类型名称匹配，详见诊断编程接口。

G.1 CPU 诊断码

名称	诊断码	诊断信息
SDCardError	1	SD 卡错误
FlashError	1	Flash 错误
SystemError	0 x 40	高速 I/O 接口板连接错误
InterCommError	0x11	无 IO 扩展模块 (板间通信错误 : 读校验失败)
	0x12	无 IO 扩展模块 (板间通信错误 : 写校验失败)
	0x13	无 IO 扩展模块 (板间通信错误 :ACK 为高电平)
	0x14	无 IO 扩展模块 (板间通信错误 : ACK 低电平)
	0x21	实际 IO 扩展模块数少于组态配置 (板间通信错误 : 读校验失败)
	0x22	实际 IO 扩展模块数少于组态配置 (板间通信错误 : 写校验失败)
	0x23	实际 IO 扩展模块数少于组态配置 (板间通信错误 :ACK 为高电平)
	0x24	实际 IO 扩展模块数少于组态配置 (板间通信错误 : ACK 低电平)
	0x31	实际 IO 扩展模块数多于组态配置 (板间通信错误 : 读校验失败)
	0x32	实际 IO 扩展模块数多于组态配置 (板间通信错误 : 写校验失败)
	0x33	实际 IO 扩展模块数多于组态配置 (板间通信错误 :ACK 为高电平)
	0x34	实际 IO 扩展模块数多于组态配置 (板间通信错误 : ACK 低电平)
	0x41	IO 扩展模块类型错误 (板间通信错误 : 读校验失败)
	0x42	IO 扩展模块类型错误 (板间通信错误 : 写校验失败)
	0x43	IO 扩展模块类型错误 (板间通信错误 :ACK 为高电平)
	0x44	IO 扩展模块类型错误 (板间通信错误 : ACK 低电平)
ConformenceError (每位表示一个模块故障)	1	插槽 1 对应的 IO 模块和实际 IO 模块组态不一致
	2	插槽 2 对应的 IO 模块和实际 IO 模块组态不一致
	4	插槽 3 对应的 IO 模块和实际 IO 模块组态不一致
	8	插槽 4 对应的 IO 模块和实际 IO 模块组态不一致
	16	插槽 5 对应的 IO 模块和实际 IO 模块组态不一致
	32	插槽 6 对应的 IO 模块和实际 IO 模块组态不一致
	64	插槽 7 对应的 IO 模块和实际 IO 模块组态不一致
	128	插槽 8 对应的 IO 模块和实际 IO 模块组态不一致
	256	插槽 9 对应的 IO 模块和实际 IO 模块组态不一致
	512	插槽 10 对应的 IO 模块和实际 IO 模块组态不一致
	1024	插槽 11 对应的 IO 模块和实际 IO 模块组态不一致
	2048	插槽 12 对应的 IO 模块和实际 IO 模块组态不一致
	4096	插槽 13 对应的 IO 模块和实际 IO 模块组态不一致
	8192	插槽 14 对应的 IO 模块和实际 IO 模块组态不一致
	16384	插槽 15 对应的 IO 模块和实际 IO 模块组态不一致
	32768	插槽 16 对应的 IO 模块和实际 IO 模块组态不一致

名称	诊断码	诊断信息
IOModulePosError (每位表示一个模块故障，因具体故障在 IO 模块已经显示，所以此诊断信息不显示，只作标志状态)	1	插槽 1 对应的 IO 模块出现故障
	2	插槽 2 对应的 IO 模块出现故障
	4	插槽 3 对应的 IO 模块出现故障
	8	插槽 4 对应的 IO 模块出现故障
	16	插槽 5 对应的 IO 模块出现故障
	32	插槽 6 对应的 IO 模块出现故障
	64	插槽 7 对应的 IO 模块出现故障
	128	插槽 8 对应的 IO 模块出现故障
	256	插槽 9 对应的 IO 模块出现故障
	512	插槽 10 对应的 IO 模块出现故障
	1024	插槽 11 对应的 IO 模块出现故障
	2048	插槽 12 对应的 IO 模块出现故障
	4096	插槽 13 对应的 IO 模块出现故障
	8192	插槽 14 对应的 IO 模块出现故障
	16384	插槽 15 对应的 IO 模块出现故障
	32768	插槽 16 对应的 IO 模块出现故障
FunctionErrorCode (每位表示一个总线出现故障，只作标志状态)	0x01	DP 总线出现故障
	0x02	EtherCAT 总线出现故障
	0x04	CANopen 总线出现故障
	0x08	CANlink 总线出现故障
	0x10	ModbusTCP 出现故障
	0x20	Modbus 串口 0 出现故障
	0x40	Modbus 串口 1 出现故障
	0x80	高速 I/O 出现故障



NOTE

因 EtherCAT 为 Codesys 实现，PLC 暂时无法直接获取 EtherCAT 诊断，EtherCAT 总线标志暂时没用。

G.2 I/O 模块诊断码

名称	模块类型	诊断码	诊断信息
BaseInfo	所有模块	64	出现故障，系统停机
ModuleError (每位表示一种模块故障)	AI	2	无外部负载电压
		4	模拟芯片连接错误
	AO	2	无外部负载电压
		4	模拟芯片连接错误
		8	模拟芯片过热

名称	模块类型	诊断码	诊断信息
ChannelError[i] (数组每个元素表示每个通道诊断代码，每位表示一种故障)	AI	2	上溢
		4	下溢
		8	超上限
		16	超下限
		32	断线
	AO	2	上溢
		4	下溢
		8	电流断线
		16	电压短路
		32	DAC 通道硬件故障

G.3 CANopen 诊断码

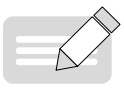
名称	诊断码	诊断信息
SlaveError	1	从站离线
InterCommError	0x11	无 IO 扩展模块 (板间通信错误 : 读校验失败)
	0x12	无 IO 扩展模块 (板间通信错误 : 写校验失败)
	0x13	无 IO 扩展模块 (板间通信错误 :ACK 为高电平)
	0x14	无 IO 扩展模块 (板间通信错误 : ACK 低电平)
	0x21	实际 IO 扩展模块数少于组态配置 (板间通信错误 : 读校验失败)
	0x22	实际 IO 扩展模块数少于组态配置 (板间通信错误 : 写校验失败)
	0x23	实际 IO 扩展模块数少于组态配置 (板间通信错误 :ACK 为高电平)
	0x24	实际 IO 扩展模块数少于组态配置 (板间通信错误 : ACK 低电平)
	0x31	实际 IO 扩展模块数多于组态配置 (板间通信错误 : 读校验失败)
	0x32	实际 IO 扩展模块数多于组态配置 (板间通信错误 : 写校验失败)
	0x33	实际 IO 扩展模块数多于组态配置 (板间通信错误 :ACK 为高电平)
	0x34	实际 IO 扩展模块数多于组态配置 (板间通信错误 : ACK 低电平)
	0x41	IO 扩展模块类型错误 (板间通信错误 : 读校验失败)
	0x42	IO 扩展模块类型错误 (板间通信错误 : 写校验失败)
	0x43	IO 扩展模块类型错误 (板间通信错误 :ACK 为高电平)
	0x44	IO 扩展模块类型错误 (板间通信错误 : ACK 低电平)

名称	诊断码	诊断信息
ConformenceError (每位表示一个模块故障)	1	插槽 1 对应的 IO 模块和实际 IO 模块组态不一致
	2	插槽 2 对应的 IO 模块和实际 IO 模块组态不一致
	4	插槽 3 对应的 IO 模块和实际 IO 模块组态不一致
	8	插槽 4 对应的 IO 模块和实际 IO 模块组态不一致
	16	插槽 5 对应的 IO 模块和实际 IO 模块组态不一致
	32	插槽 6 对应的 IO 模块和实际 IO 模块组态不一致
	64	插槽 7 对应的 IO 模块和实际 IO 模块组态不一致
	128	插槽 8 对应的 IO 模块和实际 IO 模块组态不一致
	256	插槽 9 对应的 IO 模块和实际 IO 模块组态不一致
	512	插槽 10 对应的 IO 模块和实际 IO 模块组态不一致
	1024	插槽 11 对应的 IO 模块和实际 IO 模块组态不一致
	2048	插槽 12 对应的 IO 模块和实际 IO 模块组态不一致
	4096	插槽 13 对应的 IO 模块和实际 IO 模块组态不一致
	8192	插槽 14 对应的 IO 模块和实际 IO 模块组态不一致
	16384	插槽 15 对应的 IO 模块和实际 IO 模块组态不一致
	32768	插槽 16 对应的 IO 模块和实际 IO 模块组态不一致
IOModulePosError (每位表示一个模块故障，因具体故障在 IO 模块已经显示，所以此诊断信息不显示，只作标志状态)	1	插槽 1 对应的 IO 模块出现故障
	2	插槽 2 对应的 IO 模块出现故障
	4	插槽 3 对应的 IO 模块出现故障
	8	插槽 4 对应的 IO 模块出现故障
	16	插槽 5 对应的 IO 模块出现故障
	32	插槽 6 对应的 IO 模块出现故障
	64	插槽 7 对应的 IO 模块出现故障
	128	插槽 8 对应的 IO 模块出现故障
	256	插槽 9 对应的 IO 模块出现故障
	512	插槽 10 对应的 IO 模块出现故障
	1024	插槽 11 对应的 IO 模块出现故障
	2048	插槽 12 对应的 IO 模块出现故障
	4096	插槽 13 对应的 IO 模块出现故障
	8192	插槽 14 对应的 IO 模块出现故障
	16384	插槽 15 对应的 IO 模块出现故障
	32768	插槽 16 对应的 IO 模块出现故障

G.4 DP 诊断码

名称	诊断码	诊断信息
ExtDiagData[0] (每位表示一种故障)	0x02	未准备好进行数据交换
	0x04	组态错误
	0x08	该从站有扩展的诊断信息
	0x10	该从站不支持所请求的功能
	0x20	无效的从站响应
	0x40	参数设置错误
	0x80	被不同的主站锁定

名称	诊断码	诊断信息
ExtDiagData[1] (每位表示一种故障)	0x01	必须重新设置参数
	0x02	静态诊断
	0x08	看门狗监控被激活
	0x10	该从站处理锁存模式
	0x20	该从站处理同步模式
	0x80	该从站未被主站激活
ExtDiagData[2] (每位表示一种故障)	0x80	诊断数据溢出
ExtDiagData[3] (主站地址)		
ExtDiagData[4] 和 ExtDiagData[5] (从站 ID)		



NOTE

诊断前 6 个字节为基本诊断，后面的诊断数据位扩展诊断，具体解析详见 DP 诊断。

G.5 CANlink 诊断码

名称	诊断码	诊断信息
CmdFrameError (诊断码为除以 100 的余数)	1	命令码非法
	2	命令码地址异常
	3	数据值不在允许范围内
	4	命令码操作无效
	5	命令码数据长度无效
	6	命令码超时
CfgFrameError (诊断码为除以 100 的余数)	1	配置编码出错
	2	配置索引出错
	3	配置信息出错
	5	配置数据长度出错
	6	配置帧未响应

G.6 Modbus 诊断码

名称	诊断码	诊断信息
DiagData	0x70	Modbus 从站地址设置错误
	0x71	数据帧长度错误，串口 0（COM0)
	0x72	非法数据地址
	0x73	CRC 校验错误
	0x74	不支持的命令码

名称	诊断码	诊断信息
DiagData	0x75	接收超时
	0x76	非法数据值
	0x77	缓冲区溢出
	0x78	帧错误
	0x79	串口协议错误
	0x7C	地址错误
	0x7D	未收到数据
	0x7E	从站返回错误数据
	0x80	Modbus 从站地址设置错误
	0x81	数据帧长度错误，串口 1（COM1）
	0x82	非法数据地址
	0x83	CRC 校验错误
	0x84	不支持的命令码
	0x85	接收超时
	0x86	非法数据值
	0x87	缓冲区溢出
	0x88	帧错误
	0x89	串口协议错误
	0x8C	地址错误
	0x8D	未收到数据
	0x8E	从站返回错误数据
	0x90	Modbus 从站地址设置错误
	0x91	数据帧长度错误，以太网（MODBUS TCP）
	0x92	非法数据地址
	0x93	CRC 校验错误
	0x94	不支持的命令码
	0x95	接收超时
	0x96	非法数据值
	0x97	缓冲区溢出
	0x98	帧错误
	0x99	串口协议错误
	0x9A	从站无连接
	0x9B	协议标示符不正确
	0x9C	地址错误
	0x9D	未收到数据
	0x9E	从站返回错误数据
	0x9F	客户端连接数超限
	0xA0	非法数据值

G.7 EtherCAT 诊断码

名称	诊断码	诊断信息
m_LastError	0	无错误
	1	PLC 通讯断开，检查网线。
m_LastError	2	同步单元的工作计数器错误。
	3	总线分布时钟错误
	4	打开第一个网络适配器失败
	5	打开第二个网络适配器失败
	6	第二个网络适配器与第一个网络适配的网卡地址相同
	7	初始化所有从站失败
	8	工作计数器错误
	9	读取的从站供应商 ID 与配置的不一致
	10	读取的产品 ID 与配置的不一致
	11	SDO 写错误
	12	SDO 传输超时
	13	从站紧急报文
	14	SOE 写错误
	15	SOE 传输超时
	16	看门狗超时
	101	未知错误
	102	邮箱申请内存失败
	106	固件版本与 EEPROM 内容不一致
	107	固件更新失败
	117	无效的请求状态变化
	118	不确定的状态请求
	119	不支持引导模式
	120	固件程序无效
	121	Boot 模式下的无效邮箱配置
	122	Pre-op 模式下的无效邮箱配置
	123	无效的 SM 通道配置
	124	无效的输入数据
	125	无效的输出数据
	126	同步错误
	127	SM 看门狗错误
	128	无效的 SM 类型
	129	无效的输出配置
	130	无效的输入配置
	131	无效的看门狗配置
	132	从站需要冷启动
	133	从站需要初始化状态
	134	从站需要预操作状态
	135	从站需要安全操作状态
	136	无效的输入映射

名称	诊断码	诊断信息
m_LastError	137	无效的输出映射
	138	不一致的设置
	139	从站不支持自由运行模式
	140	从站不支持同步运行模式
	141	从站自由运行模式需要工作 3 个缓冲模式
	142	看门狗
	143	无效输入和输出
	144	致命的同步错误
	145	从站没有同步中断信号错误
	146	同步周期时间太小
	147	
	148	无效的 DC 同步配置
	149	无效的 DC 锁存配置
	150	从站 PLL 错误
	151	无效的 DC 输入输出错误
	152	无效的 DC 超时错误
	153	无效的同步循环时间
	154	Sync0 的 DC 周期设置不合理
	155	Sync1 的 DC 周期设置不合理
	165	MBX_AOE 错误
	166	MBX_EOE 错误
	167	MBX_COE 错误
	168	MBX_FOE 错误
	169	MBX_SOE 错误
	179	MBX_VOE 错误
	180	EEPROM 地址不能访问
	181	EEPROM 错误
	182	外部硬件未准备好
	212	从站监测到的配置与后台组态配置不一致。

附录 H CPU 占有率过高分析

H.1 CPU 占有率定义

0%-89%：PLC 运行比较稳定。逻辑执行、总线同步、IO 刷新、数据同步、数据保存都有时间保证。

90%-100%：PLC 运行稳定性降低。主要影响：

- EtherCAT 运行稳定性难以保证，可能出现 EtherCAT 从站掉线、同步丢失；
- 严重的可能使 PLC 处于“假死”状态，不能扫描登录 PLC
- 掉电存储数据不能保存。
- CANopen、CANlink、Modbus/ModbusTCP 存在数据刷新、断线风险
- 在线修改或者下载 PLC 程序可能变慢，并且有可能失败
- 监视的 PLC 变量值，存在刷新缓慢或者无法刷新的风险。

NOTE

CPU 占有率问题，目前适用于 AM600、AM400 系列 PLC，对 AC800、AP700 系列暂不定义。

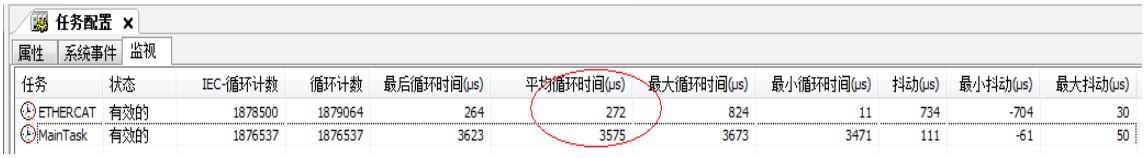
H.2 分析步骤

步骤一：查看 PLC CPU 占有率。

登录 PLC，通过后台状态条能查看 CPU 占有率；如下图。

步骤二：查看任务执行时间，并计算执行时间在任务中占比。

登录 PLC 后，打开【任务配置】-【监视】界面，查看任务的平均循环时间，如下图。



任务	状态	IEC-循环计数	循环计数	最后循环时间(μs)	平均循环时间(μs)	最大循环时间(μs)	最小循环时间(μs)	抖动(μs)	最小抖动(μs)	最大抖动(μs)
ETHERCAT	有效的	1878500	1879064	264	272	824	11	734	-704	30
MainTask	有效的	1876537	1876537	3623	3575	3673	3471	111	-61	50

NOTE

如果登录之前已经打开，登录后再次打开时，需要右键任务，执行菜单命令【复位】，恢复初始计算状态。

上图中，EtherCAT 任务和 MainTask 任务循环周期都是 4ms，而 MainTask 任务占比约 3575/4000 约等于 89%，也就是说，MainTask 占有过多的执行逻辑。

步骤三：优化任务中程序。

优化程序首先找到执行时间过长的程序，然后再找到程序中执行比较久的代码段。

找到占用过多 CPU 时间的程序，一般通过删减任务下的程序来判断。如果删除任务下的程序后，任务执行时间明显减少，表示此程序可能需要优化。

找到程序后，需要找到此执行比较久的代码，也是通过删除程序中的代码来判断。

H.3 常见优化方式

- 增大任务扫描周期

任务扫描周期增加后，任务中程序执行次数减少，相应的占用 CPU 时间会减少。

- 批量数据处理代码优化

一般程序是循环执行的，对于批量数据的处理，可以考虑多个周期处理。例如初始化代码、对时效性要求不是很高的逻辑，都可以多周期执行。

- 增加 IF 条件

程序中功能块和函数，如果不增加条件，每个周期都会一直执行的。实际情况下，可能需要某个条件才需要执行，可以增加 IF 条件，满足条件才执行。在 ST 中可以考虑增加 IF 条件，在 LD 中，把运算块变为 EnEno 类型。

- 更换更高性能 PLC

附录 I 同步工程信息

I.1 概述

同步工程信息，主要是将工程数据与下载数据同步，保证准确登录 PLC。

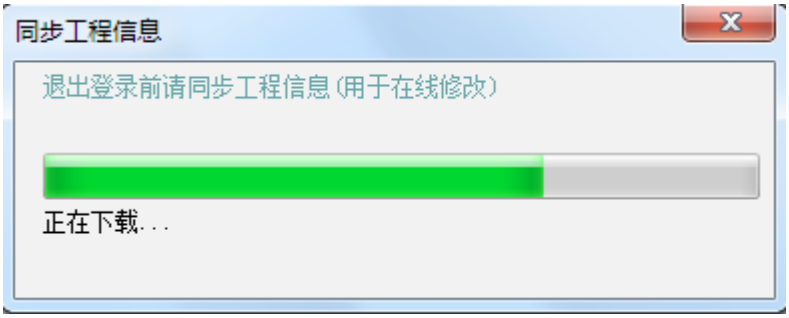
同步工程信息主要解决非预期的下载问题：

- 只拷贝工程文件
- 一个工程对多个 PLC 调试
- 多人同步编程调试。
- 工程“清除全部”，需要下载

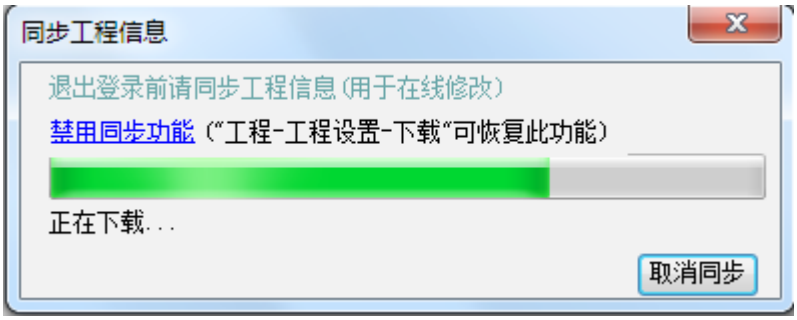
I.2 自动下载同步工程信息

如果勾选菜单【工程】-【工程设置...】-【下载】-【下载工程信息】，当登录下载完用户程序后，会自动启动工程信息同步，在同步过程中，状态栏【同步工程信息...】橙色闪烁，如下图：

如果双击状态栏【同步工程信息...】，则显示当前同步过程，如下图：



如果同步工程信息没有完成，退出登录时，会弹出同步工程信息状态框，如下图：

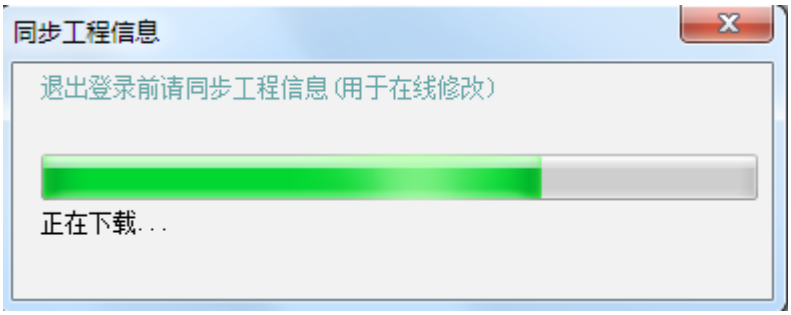


- 同步完成后，同步工程信息界面自动关闭，退出登录
- 如果点击“X”，关闭界面，取消退出登录
- 如果选择【取消同步】，同步工程过程中断，状态界面关闭，退出登录
- 如果点击【禁用同步功能】，则当前工程的“同步工程信息”功能禁用，状态界面关闭，退出登录

I.3 手动下载同步工程信息

通过菜单命令【在线】-【同步工程信息...】，可以同步工程信息。此命令在登录 PLC 后才能执行。

执行此命令后，显示同步状态，如下图：



手动下载工程信息，会强制下载，不管工程是否禁用了“同步工程信息”功能。

I.4 同步工程信息的特殊说明

- 如果 CPU 正常运行负载比较高（超过 85%），可能“同步工程信息”过程相对较慢
- 如果需要对单 PLC 进行多次调试，可先禁用同步功能，调试完手动操作“同步工程信息。

附录 J PLC 运行异常处理方法指导

汇川技术中型 PLC（AM 系列、AC 系列）开发语言是基于 IEC61131-3 国际标准而设计的一种编译型语言。

编译型语言不同于解析型语言（小型 PLC 一般使用），编译型语言的程序在执行之前需要一个专门的编译过程，把程序编译成为机器语言的文件，运行时不需要重新翻译，直接使用编译的结果。编译型语言程序编写方式灵活、执行效率高，对编程人员能力要求相对较高（有 C/C++ 基础比较合适）。

用户在编写程序时需要注意指针非法访问、除数为 0、数组越界、类型隐式转换、死循环、全局变量保护等规避事项，否则用户程序将大概率造成 PLC 运行异常 , 甚至死机。

本文档重点介绍 PLC 运行用户程序异常 (下载异常、死机) 的主要原因、定位步骤、解决方法。

J.1 异常现象

NOTE

◆ 问题定位后，建议手动删除所有隐含检查函数，因为隐含检查函数会消耗一部分 CPU 资源。
◆ AM 系列 PLC 的固件版本必须在 V1.22.0.0 及以上。编程软件 InoProShop 版本必须在 V1.3.2 及以上。

- AM 系列 LED 数码管显示停止刷新（正常情况显示“00”），AC 系列液晶显示屏显示内容“Runtime crash”。
- 编程软件无法扫描到相应的 PLC 设备，PLC 重新上电后正常，运行一段时间又无法扫描到 PLC。
- 下载程序后或者 PLC 运行一段时间，登录 PLC 后信息显示栏，提示程序“停止”，错误内容“程序下载 - 异常”，如下图所示。

- 下载程序后或者运行一段时间，登录 PLC 后弹出“没有源代码使用于这个项目…”提示框 , 如下图所示。

J.2 异常原因

1 指针非法访问

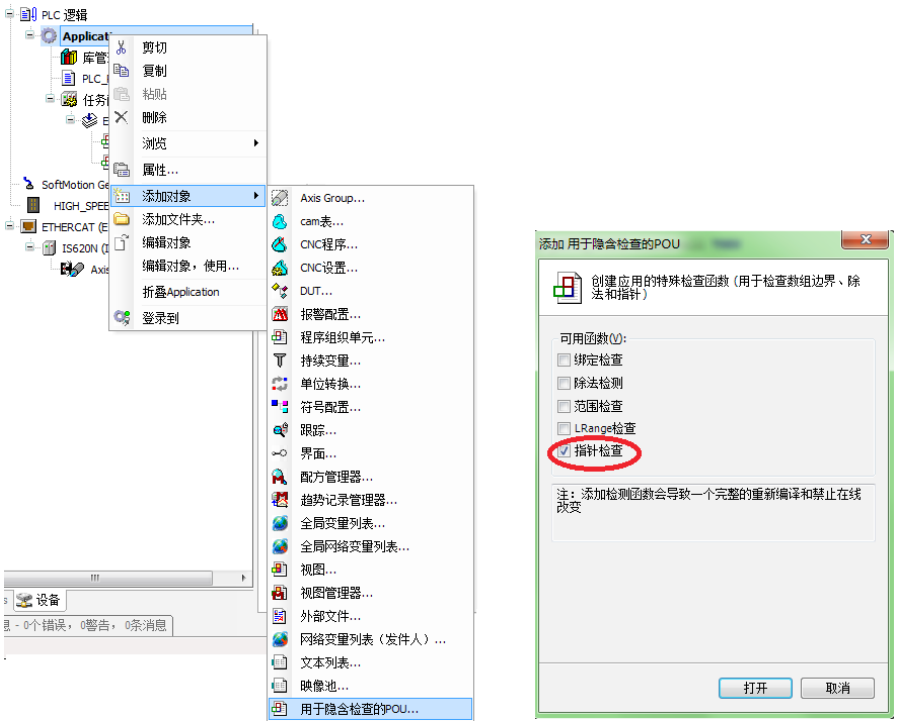
分为两种情形：空指针（指针指向地址值为 0x00000000）、指针指向不合法区域（指针指向的地址与操作系统内部地址冲突）。

PLC 操作系统不能执行空指针（地址 0x00000000 是一些单片机系统启动地址，会导致单片机软重启），空指针比较容易定位。

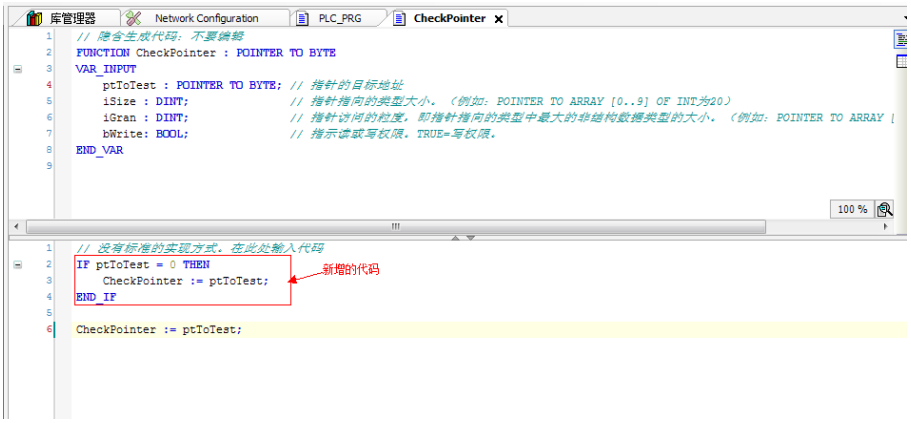
指针指向不合法区域会与系统其他正在运行的程序冲突，执行异常，这一类指针比较难查找原因，建议初级用户能用数组尽量用数组，不要使用指针。

- 定位步骤

用户程序设备树“应用”添加“用于隐含检查的 POU”，并勾选“指针检查”。如下图所示：



添加“指针检查”后，编程软件会在“应用”设备树下默认添加“CheckPointer”函数，在函数中手动添加调试代码，如下图所示：



用户程序每执行一次指针调用，系统将默认执行 “CheckPointer”隐含检查函数一次。通过在函数内部添加程序断点的调试方法，可以定位空指针在用户程序中使用的具体位置（断点调试方法请参考附录）。

登录 PLC，在新增代码处“CheckPointer := ptToTest;”右键添加断点并激活，手动启动运行 PLC(AM 系列调试阶段需要将 PLC 的运行开关拨到“STOP”，如果是 AC 系列，需要在小屏幕设置为“程序开机运行”)，目前只能定位指针为 0 的情况，如下图。

■ 解决方法

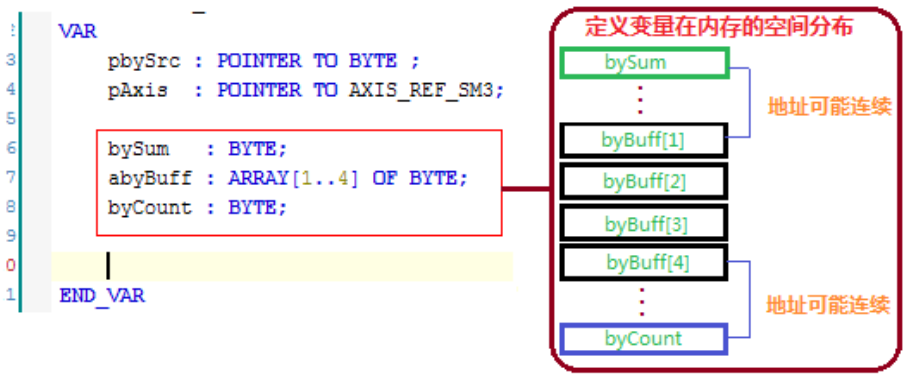
用户程序指针调用处增加指针非 0 判断条件，如下图：

```
IF ( pAxis <> 0) THEN
    IF pAxis^.nAxisState = SMC_AXIS_STATE.standstill THEN
        //用户执行代码
    END_IF
END_IF
```

2 数组越界

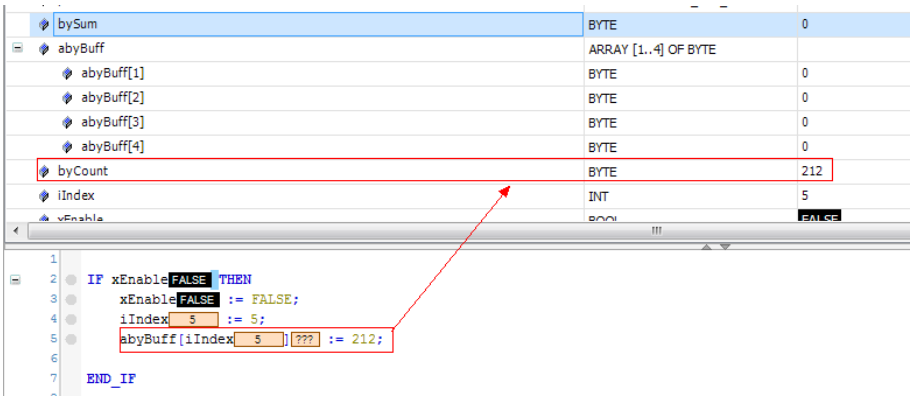
数组越界，分为上越界、下越界。程序中调用越界数组，可能导致数组相邻的变量值被调用的越界数组值覆盖。

程序中 POU 变量定义和数组在系统的内存分布大致如下图中 bySum、abyBuff、byCount 之间的关系。



变量定义区，byBuff[1] 和变量 bySum 相邻，byBuff[4] 与 byCount 相邻。如果用户程序对 byBuff[0]、by-Buff[5] 进行写操作，那么 bySum、byCount 值可能分别被数组的 byBuff[0]、byBuff[5] 值覆盖，共用相同内存地址（编译时，系统已经所有变量分配内存地址。由于内存地址分配有优化算法，因此变量之间地址可能连续，可能不连续。一般情况下，数组上界与变量内存地址连续的可能性较大）。

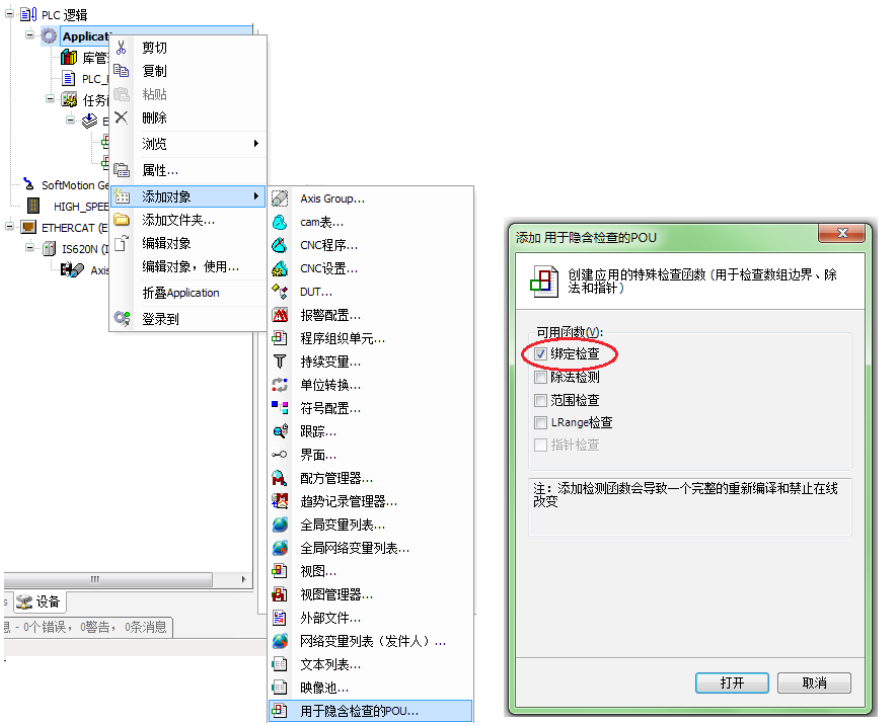
举例：iIndex 作为数组变量，当 iIndex 等于 5 时，byBuff[iIndex] 赋值“212”，则 byCount 等于 212（如下图），因为 byCount 指向的地址与 byBuff[5] 地址连续导致。



当 iIndex 等于 0 时，byBuff[iIndex] 赋值“212”，则 bySum 等于 0，因为 bySum 指向的地址与 byBuff[0] 地址不连续导致。

■ 定位步骤

与指针异常步骤类似，在用户程序设备树“应用”添加“用于隐含检查的 POU”，并勾选“绑定检查（纯音译结果，正确翻译应该是检查界限 / 边界）”，系统将在“应用”设备树下默认添加“CheckBounds”函数，如下图所示：



登录 PLC，在代码处“CheckBounds := lower; 和“CheckBounds := upper;”，增加断点并激活。通过单步调试（F10）定位到发生异常的用户程序位置，对比当前数组变量是否在数组定义的范围，如下图所示，abyBuff[5] 不在定义 byBuff[1..4] 范围内。

```
1  // 隐含生成代码：这里只是对代码实现的建议
2  IF  index  5  < lower  1  THEN
3      CheckBounds  0  := lower  1  ;
4  ELSIF  index  5  > upper  4  THEN
5      CheckBounds  0  := upper  4  ;
6  ELSE
7      CheckBounds  0  := index  5  ;
8  END_IF
```

单步调试（F10）直到进入用户程序

```
1
2  IF xEnable FALSE  THEN
3      xEnable FALSE  := FALSE;
4      iIndex  5  := 5;
5      abyBuff[iIndex  5 ] ???  := 212;
6
7  END_IF
```

■ 解决方法

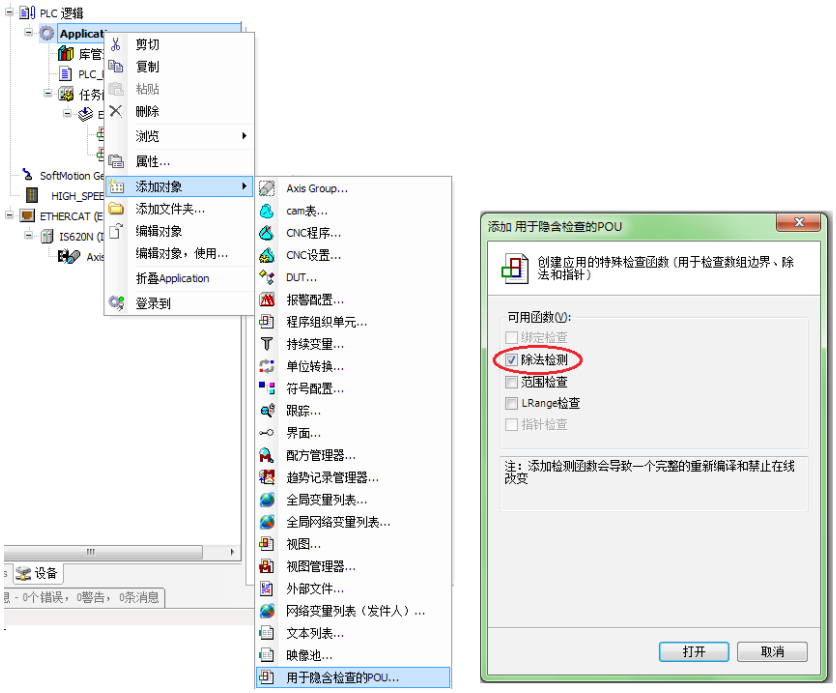
无，必须在代码中解决。

3 除数为 0

变量未初始化、用户程序变量初始化在调用之后、全局变量在多个任务或者多个 POU 被赋值。

■ 定位步骤

在用户程序设备树“应用”添加“用于隐含检查的 POU”，并勾选“除法检查”。如下图所示：



系统在“应用”设备树下默认添加“CheckDivDInt”、“CheckDivLInt”、“CheckDivLReal”、“CheckDivReal”四个函数。

登录 PLC，在四个函数代码“CheckDivDInt:=1;”、“CheckDivLInt:=1;”、“CheckDivLReal:=1;”、“CheckDivReal:=1;” 位置增加断点并激活。通过单步调试（F10）定位到用户程序位置（具体操作参考前面指针异常调试步骤）。

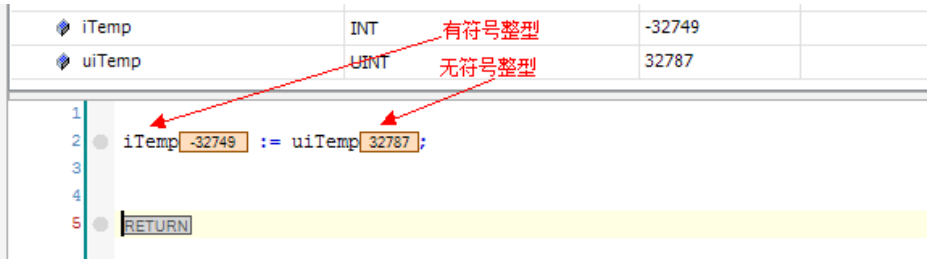
■ 解决方法

参与运算的代码处增加“除数等于 0”的条件判断，32bit 变量通常用 10E-6(0.000001) 作为判断门槛值，64bit 变量根据需要调整精度，最小≥ 10E-15，如下图。

```
IF ABS(fDiv) >= 0.000001 THEN
    fRet := (fSum / fDiv);
END_IF
```

4 数据类型隐式转换

具有相同数据宽度的有符号和无符号变量之间赋值运算，强制赋值可能导致变量的值不在预期范围，变量在其他程序段被引用，可能会导致程序执行异常，如下图所示。



■ 定位步骤

无

■ 解决方法

用户程序在编译过程中会生成警告信息，重视编译警告信息的具体内容，保证参与赋值运算左右两边的数据类型相同。

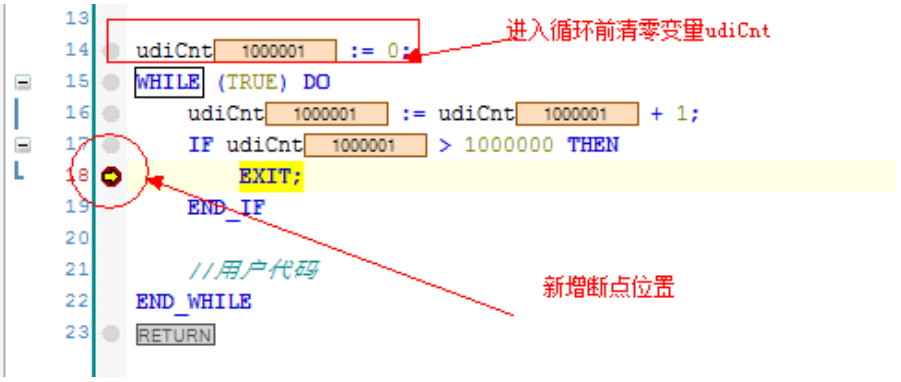
5 死循环

程序中 for、while、repeat 循环条件使用不当，系统会一直在执行循环体中程序代码段，无法执行循环体以外代码，比如：PLC 的 LED 数码管显示刷新、PLC 与编程软件之间的通信，而 EtherCAT 任务下的死循环程序会导致掉电保持数据功能无法正常工作，丢失掉电保持数据。

■ 定位步骤

for、while、repeat 循环中引入计数变量，当计数变量到达一定值，循环跳出。登录 PLC，在循环跳出位置“EXIT”增加断点，一旦系统循环计数超出预期，程序就会运行到断点处，断点使用方法参考帮助手册，

如下图所示，udiCnt 循环执行次数检测变量，当循环体 while 下的代码执行 100001 次之后，程序运行到“EXIT”处。



■ 解决方法

按照定位步骤解决，或者在循环体内增加定时器，当定时时间到达，程序跳出循环体。

6 功能块实例在多个任务调用

功能块实例的内部变量在扫描周期内执行一次后，第二次执行时的变量仍保持上一次的值（类似 C/C++ 语言的静态变量）。两个任务 A、B 同时调用相同功能块实例（包括方法、动作、属性、转移）时，会出现 A 任务中功能块实例未执行完毕，被优先级高的 B 任务抢占执行，等待 B 任务执行完毕，A 任务再执行时候，功能块实例的内部变量的数值与被抢占前的数值可能不完全一致，最后影响功能块在 A、B 任务的正常执行。

■ 定位步骤

无。

■ 解决方法

无，必须在前期代码设计时规避。

7 全局变量在多个任务调用

两个任务 A、B 中都对同一变量进行写访问操作，会出现 A 任务正在写全局变量，被优先级高的 B 任务抢占 CPU 资源，并写全局变量的值，等到 B 任务执行完毕，A 任务再执行时候，全局变量的值可能与被抢占之前的值不一致，最后影响功能块正常执行。

例如： wTemp 是一个具备 2 个字节的全局变量，wTempH、wTempL 分别为 wTemp 高、低位字节。当 A 任务执行写完 wTempL 还未写 wTempH 时 , 被高优先的 B 任务抢占 CPU 资源，B 任务写完 wTempL 和 wTempH 后，释放 CPU 资源后，A 任务接着执行写 wTempH。由于 wTempL 被任务改写，wTemp 在 A 任务的值不在预期范围内。

■ 定位步骤

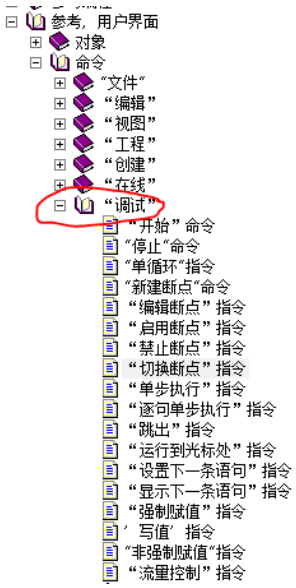
无

■ 解决方法

全局变量只能在一个任务执行写访问，必须在代码设计时候解决。（汇川技术中型 PLC 可以通过互斥信号量解决次类问题，但这仅仅适合对操作系统有深入理解的高级开发人员，普通用户不要轻易尝试，使用不当会造成 PLC 死机）。

8 其他

用户程序断点、单步等调试方法请参考帮助手册 , 相关内容在帮助手册路径如下图。



创变·精彩

官方微信

数字图书馆

深圳市汇川技术股份有限公司
Shenzhen Inovance Technology Co., Ltd.
地址：深圳市宝安区宝城70区留仙二路鸿威工业区E栋
总机：(0755)2979 9595
传真：(0755)2961 9897
http://www.inovance.com

苏州汇川技术有限公司
Suzhou Inovance Technology Co., Ltd.
地址：苏州市吴中区越溪友翔路16号
总机：(0512)6637 6666
传真：(0512)6285 6720
http://www.inovance.com

销售服务联络地址

19010334B00