SC301 系列运动控制器

用户手册

使用产品前请仔细阅读本手册，并请妥善保管。

当前版本：V1.1

权利声明

本手册版权归高创传动科技开发（深圳）有限公司所有，未经本公司书面许可，任何单位及个人不得以任何方式对本手册内的任何部分进行复制、摘录、修改、翻译、将其全部或部分用于商业用途。

本手册依据现有信息编写，高创传动科技开发（深圳）有限公司保留对本手册的最终解释权和修改权，内容如有更改，恕不另行通知。

技术支持

如您在安装和使用本产品时需要帮助，请联系 Servotronix 技术支持部门。

电话：400-111-8669。

前言

感谢您购买高创传动科技开发（深圳）有限公司研发、生产的具有完全自主知识产权的 SC301 系列运动控制器产品！

SC301 系列运动控制器是高创传动面向中国 OEM 市场正式推出的自动化控制产品，具有功能性能强大、接口丰富、编程灵活、扩展能力强等特点，是您进行自动化系统设计的理想选择。

在使用 SC301 系列运动控制器前，请您仔细阅读本手册，以便更清楚地掌握产品的特性，更安全、充分地利用本产品丰富的功能。

本手册主要描述 SC301 系列运动控制器的规格、电气特性、安装与接线，以及与之配套的扩展 I/O 模块的选型，组态编程软件 CODESYS 使用方法等。

最新版本的资料，请以高创传动公司网站 www.servotronix.cn 发布的内容为准。

本手册中展示的图片仅为示例图，可能与您订购的实际产品略有差异，请以实际产品为准。

安全声明及注意事项

安全声明

在使用本产品之前，请仔细阅读本手册并正确理解【安全注意事项】等相关信息。如果不遵守安全事项中约定的事项，可能导致人员死亡、重伤，或者设备损坏。

为保障人身和设备安全，在设计、安装、操作和维护产品时，请遵循产品上标识及手册中说明的所有安全注意事项。

本手册中的“危险”、“警告”和“注意”事项，并不代表所应遵守的所有安全事项，只作为所有安全注意事项的补充。

本产品应在符合设计规格要求的环境下使用，否则可能造成故障。因未遵守相关规定引发的功能异常或部分损坏等不在产品质量保证范围之内。

因未遵守本手册的内容、违规操作产品引发的人身安全事故、财产损失等，我司不承担任何法律责任。

安全等级定义

危险 表示如果不按规定操作，则导致死亡或严重身体伤害。

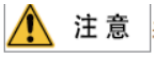
警告 表示如果不按规定操作，则可能导致死亡或严重身体伤害。

注意 表示如果不按规定操作，则可能导致轻微身体伤害或设备损坏。

安全注意事项

本说明书中的图解，有时为了展示产品的细节部分，产品为卸下外罩或安全遮盖物的状态，使用时候，请务必按规定装好外罩或者遮盖物，并按规定操作。

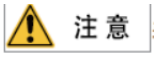
储存与运输



- 搬运产品时请务必轻抬轻放，随时注意脚下物体，防止绊倒或坠落，否则有导致受伤或产品损坏的风险！
- 徒手搬运产品时，请务必抓牢产品壳体，避免产品部件掉落，否则有导致受伤的风险！
- 请严格按照产品要求的储存与运输条件进行储存与运输，否则有导致产品损坏的风险！
- 避免在水溅雨淋、阳光直射、强电场、强磁场、强烈振动等场所储存与运输。
- 避免产品储存时间超过 3 个月，储存时间过长时，请进行更严密的防护和必要的检验。
- 请将产品进行严格包装后再进行车辆运输，长途运输时必须使用封闭的箱体！
- 严禁将本产品与可能对本产品构成影响或损害的设备或物品一起混装运输！

设计时

- 请务必设计安全电路，保证当外部电源掉电或可编程控制器故障时，控制系统依然能安全工作。
- 超过额定负载电流或者负载短路等导致长时间过电流时，模块可能冒烟或着火，应在外部设置保险丝或断路器等安全装置。



- 务必在运动控制器的外部电路中设置紧急制动电路、保护电路、正反转操作的互锁电路和防止机器损坏的位置上限、下限互锁开关。
- 为使设备安全运行，对于重大事故相关的输出信号，请设计外部保护电路和安全机构。

- 运动控制器检测到本身系统异常后可能会关闭所有输出；当控制器部分电路故障时，可能导致其输出不受控制，为保证正常运转，需设计合适的外部控制电路。
- 运动控制器的继电器、晶体管等输出单元损坏时，会使其输出无法控制为 ON 或 OFF 状态。
- 运动控制器设计应用于室内、过电压等级 II 级的电气环境，其电源系统级应有防雷保护装置，确保雷击过电压不施加于运动控制器的电源输入端或信号输入端、控制输出端等端口，避免损坏设备。

安装时

危险

- 只有受过电气设备相关培训，具有充分电气知识的专业人员才能操作。严禁非专业人员操作！

警告

- 在进行模块的拆装时，必须将系统使用的外部供应电源全部断开之后再执行操作。如果未全部断开电源，有可能导致触电或模块故障及误动作。
- 请勿在下列场所使用本产品：有灰尘、油烟、导电性尘埃、腐蚀性气体、可燃性气体的场所；强电场或强电磁波干扰的场合；暴露于高温、结露、风雨的场合；有振动、冲击的场合。电击、火灾、误操作也会导致产品损坏和恶化。
- 运动控制器为 Open type 设备，请安装在带门锁的控制柜内（控制柜外壳防护等级＞IP20），只有经电气设备相关培训、有充分电气知识的操作者才可以打开控制柜。
- 进行安装作业前，请确保安装位置的机械强度足以支撑设备重量，否则会导致机械危险。
- 进行安装作业时，请勿穿着宽松的衣服或佩戴饰品，否则可能会有触电的危险！
- 将产品安装到封闭环境（如机柜内或机箱内）中时，请用冷却装置（如冷却风扇或冷却空调）充分冷却以满足安装环境要求，否则可能导致产品过热或火灾。

- 本产品安装在柜体或终端设备中时，柜体或终端设备需要提供相应的防火外壳、电气防护外壳和机械防护外壳等防护装置，防护等级应符合相关 IEC 标准和当地法律法规要求。
- 请将产品安装在金属等阻燃物体上，勿使易燃物接触产品或将易燃物附着在产品上，否则会有引发火灾的危险。
- 在需要安装变压器等强电磁波干扰的设备时，请安装屏蔽保护装置，避免本产品出现误动作！

注意

- 进行安装作业时，请用布或纸等遮住产品顶部，以防止钻孔时的金属屑、油、水等异物进入产品内部，导致产品故障。作业结束后，请拿掉遮盖物，避免遮盖物堵住通风孔影响散热，导致产品异常发热。
- 安装时，应使其与各自的连接器紧密连接，将模块连接挂钩牢固锁定。如果模块安装不当，可能导致误动作、故障及脱落。
- 对于在干扰严重的场合，高频信号的输入或者输出电缆请使用屏蔽线缆，以提高系统的抗干扰能力。
- 请勿把控制线及通信电缆与主电路或动力电源线等捆扎在一起，走线应相距 100mm 以上，否则噪声可能导致误动作。

接线时

危险

- 严禁非专业人员进行本产品的接线！
- 接线前，请切断产品的电源，否则会有触电的危险。
- 请务必保证产品的良好接地，否则会有电击危险。

警告

- 严禁将输入电源连接到设备或产品的输出端，否则会引起设备损坏，甚至引发火灾。

- 接线时使用到的线缆必须符合相应的线径和屏蔽等要求，使用屏蔽线缆的屏蔽层需要单端可靠接地！
- 请按照手册中规定的紧固力矩进行端子螺丝紧固，紧固力矩不足或过大，可能导致连接部分过热、损坏，引发火灾危险。
- 接线完成后，请确保所有线缆接线正确，产品内部没有掉落的螺钉、垫片或裸露线缆，否则可能有触电危险或损坏产品。

注意

- 请遵守静电防止措施（ESD）规定的步骤，并佩戴静电手环进行接线等操作，避免损坏产品内部的电路。
- 对控制回路接线时，请使用双绞屏蔽线，将屏蔽层连接到产品的接地端子上进行接地，否则会导致产品动作异常。

上电时

危险

- 上电前，请确认产品安装完好，接线牢固。
- 上电前，请确认电源符合产品要求，避免造成产品损坏或引发火灾！

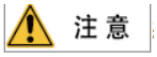
警告

- 接线作业和参数设定完成后，请进行机器试运行，确认机器能够安全动作，否则可能导致人员受伤或设备损坏。
- 通电前，请确保产品的额定电压与电源电压一致。如果电源电压使用有误，会有引发火灾的危险。
- 通电前，请确保产品、电机以及机械的周围没有人员，否则可能导致人员受伤或死亡。

运行时

- 严禁非专业人员进行产品运行，否则会有导致人员受伤或死亡危险！
- 严禁在运行状态下打开产品柜门或产品防护盖板、触摸产品的任何接线端子、拆卸产品的任何装置或零部件，否则有触电危险！

- 严禁触摸设备外壳、风扇或电阻等以试探温度，否则可能引起灼伤！
- 运行中，避免金属物体等掉入设备中，否则可能引起火灾或产品损坏！



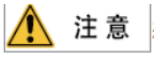
- 对于在线修改、强制输出、RUN（运行）、STOP（停止）等操作，须熟读用户手册，确认其安全性之后再进行相关操作。

维修与保养

- 严禁非专业人员进行设备保养维护、检查或部件更换！
- 严禁在通电状态下进行设备维修，否则有触电危险！

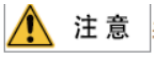
- 请按照产品保修协议进行设备保修。
- 产品出现故障或损坏时，务必由专业人员按照维修指导对产品进行故障排除和维修，并做好维修记录。
- 请勿继续使用已经损坏的机器，否则可能会造成人员伤亡或产品更大程度的损坏。

◆ 更换产品后，请务必重新进行设备接线检查与参数设置。



◆ 装卸产品前，请务必切断电源。

报废时



◆ 请按工业废弃物处理标准进行处理回收，避免污染环境。

◆ 废弃电池时，请根据各地区法令法规单独进行。

版本修订记录

修订后版本	发布日期	修订内容
V1.1	2026/09	修订部分错误描述； 新增控制器命名规则、安装与接线内容； 新增 ET、TL 系列扩展模块描述与选型内容； 新增随机配件内容； 新增 Modbus RTU/TCP 主从协议、自由协议通信内容； 新增固件更新、掉电保持、PLCHandler 通信、跨网段通信、配置网络及日期时间信息内容； 新增本体数字量输入和数字量输出接口接线原理图； 更新了本体数字量输入和数字量输出接口接线示意图； 更新了官网升级后的各种资料下载路径； 删除已退市型号 SC-E08-301-0103 相关内容。
V1.0	2023/02	初稿发布。

目　录

1. 产品概述

SC301 系列运动控制器是高创传动顺应工业 4.0 和物联网技术发展趋势，面向中国 OEM 市场正式推出的自动化控制产品，具有功能性能强大、接口丰富、编程灵活、扩展能力强等特点，是您进行小型自动化系统设计的理想选择。

高创传动具有丰富完善的工业自动化产品线，本产品通过搭配高创传动的伺服驱动器、伺服电机、扩展 I/O 模块及 HMI 产品可形成一站式整体解决方案，满足您多样化的需求。

1.1 SC301 运动控制器概述

高创传动 SC301 运动控制器，属于紧凑型运动控制器，致力于为用户提供智能化、自动化解决方案。

SC301 基于成熟稳定的 COSESYS V3 平台开发，24VDC 供电，内置 EtherCAT 标准主站，可实现高性能运动控制，最多可支持 4ms@16 个 EtherCAT 轴，具有点到点、位置同步、速度同步、电子凸轮、电子齿轮、插补等运动控制功能，并可通过 EtherCAT 扩展 I/O 设备；具有 Ethernet、CAN、RS-485、USB 等丰富的通信接口来连接其它设备；通过内部总线，最多支持本地扩展 16 个 ET 系列 I/O 模块。SC301 本体最多支持 16 路数字量输入（非高速）和 16 路数字量输出（非高速）。内置通过纽扣电池供电的 RTC。支持掉电保持，最大存储容量为 256KB。

SC301 通过 3S 公司的 CODESYS 软件进行编程、组态、调试及监控，采用符合 IEC61131-3 标准的编程语言体系，同时符合 PLCopen 规范，易于上手。

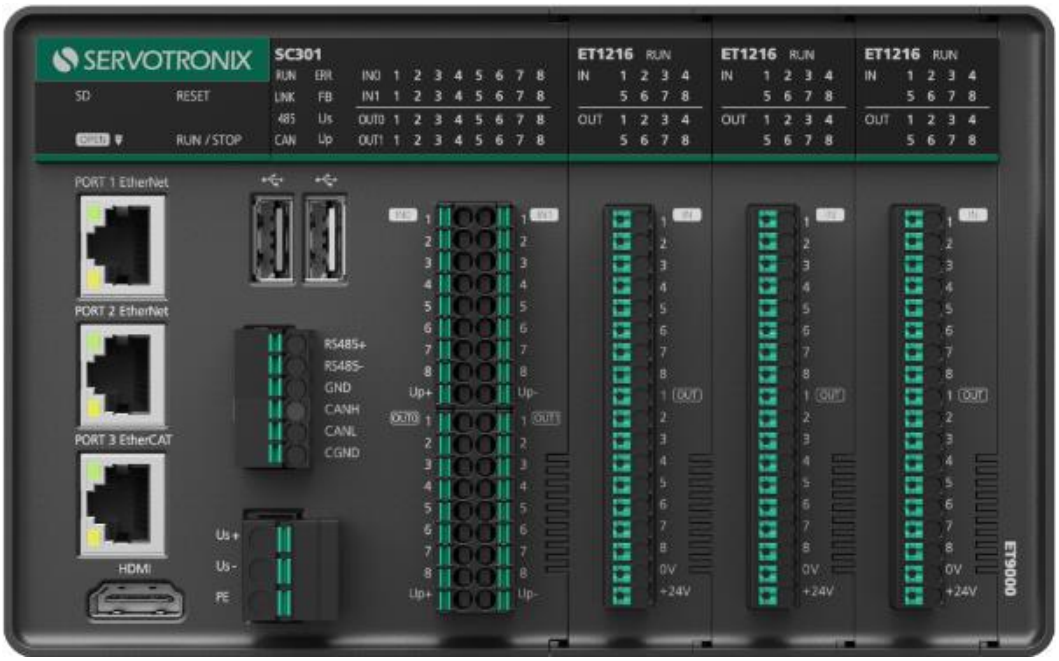


图 1-1 SC301 运动控制器搭配 ET 系列 I/O 模块组合图

SC301 控制器具有以下特点：

（1）支持点到点、位置同步、速度同步、插补、电子齿轮、电子凸轮等多种运动控制；

（2）庞大的程序和数据储存区、掉电保持区；

（3）快捷的指令执行速度；

（4）支持更多的高端协议（EtherCAT、CANopen、OPC-UA、PLCHandler）；

（5）可靠易用的软件，可满足不同的用户需求；

（6）支持在线编辑、在线侦错。

1.2 扩展 I/O 系统概述

当 SC301 控制器本体的 I/O 点数不足时，可以进行扩展，高创传动目前有 ET 和 TL 两个系列的扩展 I/O 系统可供选择，具体选型请参考本手册《附录 1》、《附录 2》。

其中 ET 系列支持本地集中扩展和通过 EtherCAT 耦合器远程集中扩展两种方式。

TL 系列仅支持通过 EtherCAT 耦合器远程集中扩展一种方式。

1.3 CODESYS 概述

CODESYS 是 3S 公司面向可编程控制器产品的编程组态软件，高创传动目前使用的版本是 CODESYS V3.5 SP17 Patch4，其提供了一套完整的配置、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。

CODESYS 可以提供如下配置功能：

(1)本体 I/O 及扩展 I/O 模块配置；

(2)EtherCAT 总线配置；

(3)CAN 总线配置；

(4)OPC-UA 配置；

(5)PLCHandler 配置；

(6)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：

①标准化编程（符合 IEC 61131-3 国际标准）：支持结构化文本（ST）、梯形图（LD）、功能块（FBD）、顺序功能图（SFC）和（CFC）多种编程语言；

②灵活的功能块库：全面的功能块库并支持用户自定义库；

③离线仿真功能：无需连接 PLC 硬件，即可完成程序调试及仿真；

④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，诊断及日志记录；

⑤采样跟踪：过程变量的时序图建立。

图 1-2 CODESYS 编程组态软件

2. 产品命名、型号与规格

2.1 运动控制器产品命名规则

高创运动控制器产品的命名规则如下表 2-1 所示。

SC - E 08 - 3 01 - 0 1 05

① ② ③ ④ ⑤ ⑥ ⑦ ⑧

表 2-1 运动控制器产品命名规则

序号	含义	说明
①	系列	MC：Motion Controller SC：SoftMC Compact Controller TC：Tiny Controller
②	现场总线	E：EtherCAT C：CANopen B：EtherCAT & CANopen
③	轴数	十进制 04：4 轴 06：6 轴 08：8 轴 16：16 轴 32：32 轴 64：64 轴
④	CPU 架构	3：ARM 7：X86（以色列） 8：X86（中国）
⑤	CPU 平台	01：ARM-单核 02：ARM-双核 03：X86-Celeron 04：X86-i3/ARM-4 核 05：X86-i5 07：X86-i7
⑥	操作系统	0：Linux 1：Windows

序号	含义	说明
⑦	软件平台	0：Control Studio 1：CODESYS 2：Control Studio+
⑧	功能许可	00：Control Studio/ Control Studio+ 01：PLC 02：PLC + WebHMI 03：SoftMotion 04：SoftMotion + WebHMI 05：SoftMotion + WebHMI + CNC + Robotics

2.2 SC301 系列选型

SC301 系列运动控制器现有型号如下表 2-2 所示，新型号陆续发布中。

SC-E08-301-0105 与 SC-E08-301-0104 相比，增加了 CNC、Robotics 功能，可以根据需要选择。

表 2-2 SC301 系列运动控制器型号及主要规格

型号	名称	主要规格
SC-E08-301-0104	DC 型 32 点 紧凑型运动控制器	①24VDC 供电，程序容量 32MB，数据容量 32MB，掉电保持容量 256KB； ②1 x EtherCAT 接口，标准 EtherCAT 主站功能，2ms@8 EtherCAT 轴，4ms@16 EtherCAT 轴； ③本体 16 通道数字量输入（NPN），16 通道数字量输出（NPN）； ④本地至多可扩展 16 个 ET 系列 I/O 模块； ⑤2 x 独立 IP 以太网接口、1 x 隔离 RS-485 接口，1 x CAN2.0A 接口、2 x USB 2.0 HOST 接口； ⑥支持 Modbus TCP 主从协议、UDP/TCP 自由协议，支持 Modbus RTU 主从协议及自由协议，支持 CANopen 协议，OPC-UA 协议，支持 PLCHandler 通信； ⑦内置 RTC（万年历）； ⑧SoftMotion+ WebHMI
SC-E08-301-0105	DC 型 32 点 紧凑型运动控制器	比 SC-E08-301-0104 增加了 CNC、Robotics 功能

2.3 SC301 规格

SC301 系列运动控制器的环境规格如下表 2-3 所示。

表 2-3 环境规格

项目	规格
使用环境	无腐蚀性、可燃性气体
工作温度	（-20～55）℃
存储温度	（-40～85）℃
工作/存储湿度	（5%～95%）RH，无凝露
工作海拔	≤2000 米
防护等级	IP20
抗振性	符合 IEC 61131-2、IEC 60068-2-6 标准。 扫描速率 1 octave/min，10 个扫描周期/轴向； （5～8.4）Hz，位移±3.5mm； （8.4～150）Hz，加速度 1g
抗冲击	符合 IEC 61131-2、IEC 60068-2-27 标准。 加速度 15g，持续时间 11ms，3 次/极性/轴向
EMC	符合 IEC 61131-2、IEC 61000-4 标准
自由跌落	带包装，1 米，5 次
接地	专用接地（接地电阻＜4Ω）或单点接地（接地电阻＜1Ω），不允许与强电系统共同接地

SC301 的主要性能规格见表 2-4。

表 2-4 性能规格

项目	规格
编程方式	IEC 61131-3 编程语言(LD、FBD、ST、SFC、CFC)
程序执行方式	编译执行
程序存储空间	32MB
数据存储空间	32MB
掉电保持容量	256KB
掉电保持方式	eMMC 保持(**如上电时间小于 30s 时发生掉电，由于充电未满，不做掉电保持**)
TF 卡容量	又称为 MicroSD 卡，容量可达 8GB，默认出厂不带

项目	规格					
各区域分配	标识	名称	容量	储存特性		
				默认	可更改属性	说明
	I	输入区	64K Words	不保存	否	X：1 位 B：8 位 W：16 位 D：32 位 L：64 位
	Q	输出区	64K Words	不保存	否	
	M	存储区	240K Words	保存	可	

3. 尺寸、接口、安装与接线

3.1 SC301 外形尺寸

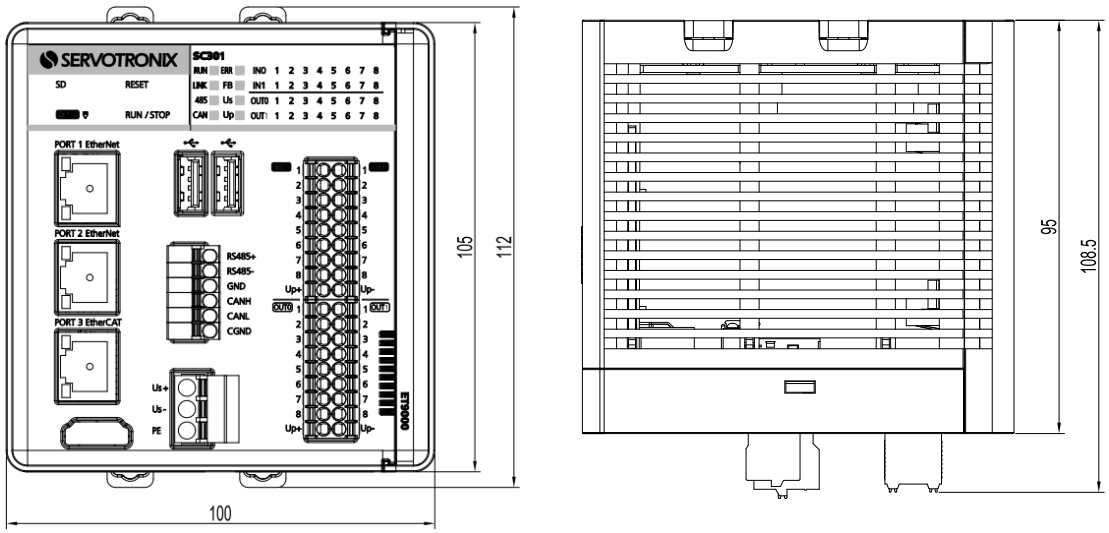


图 3-1 SC301 外形尺寸（单位:mm）

3.2 SC301 硬件接口

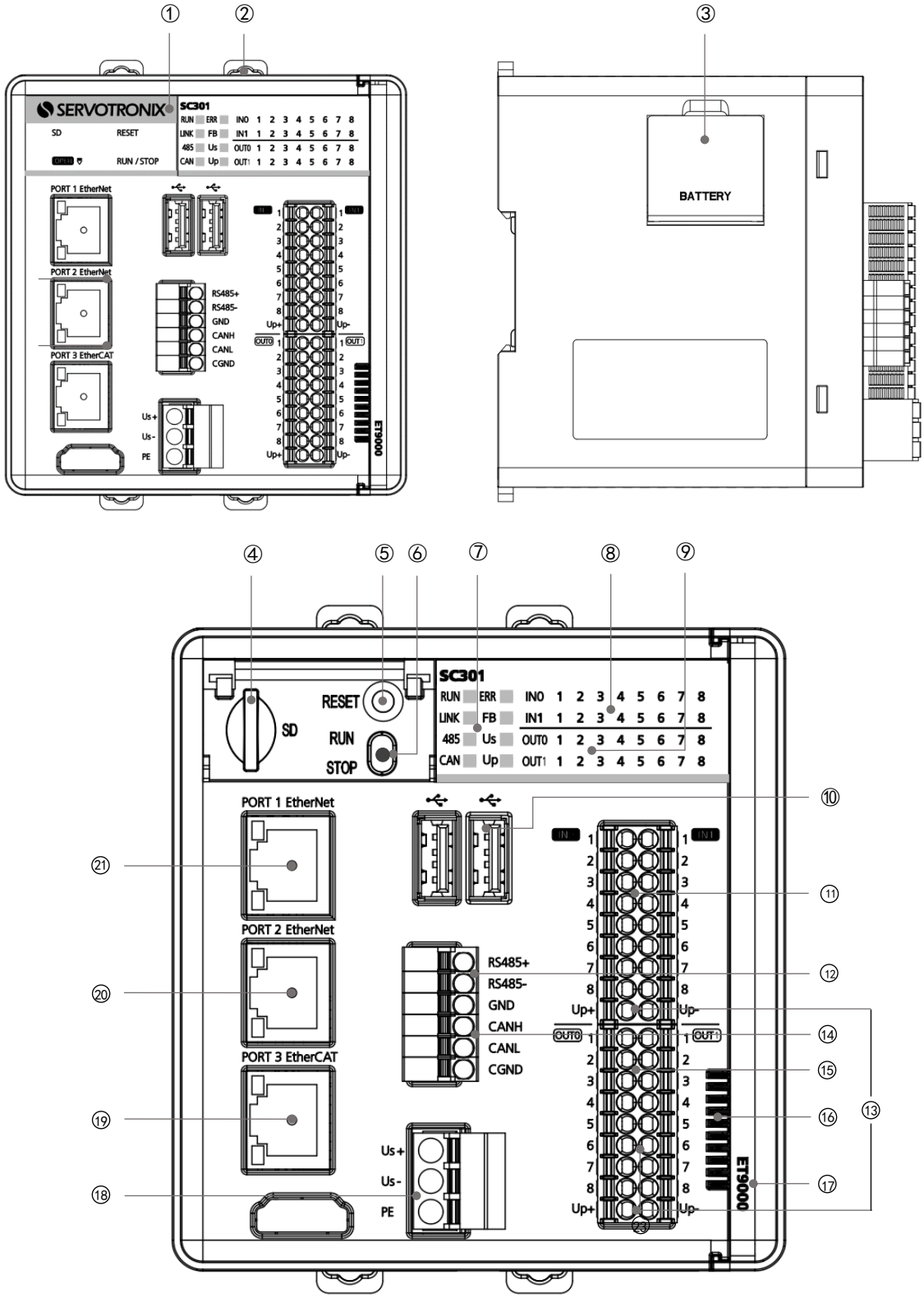


图 3-2 SC301 硬件接口

下表 3-1 给出了 SC301 硬件接口各部分的概要说明，详细说明请参考“**4. 硬**

件详细规格”章节。

表 3-1 SC301 硬件接口概要说明

序号	名称	定义
①	翻盖	用于保护内部各个接口
②	导轨安装卡扣	用于将产品固定于导轨上，2 对（共 4 个）
③	纽扣电池仓盖	用于更换纽扣电池，电池规格为 CR2032
④	MicroSD 卡接口	打开翻盖后操作，可用于固件更新 **【注意】：出厂默认不带 MicroSD 卡**
⑤	RESET 按键	打开翻盖后操作。 通过不同操作可复位默认 IP 地址/（强制）删除 CODESYS 应用程序
⑥	RUN/STOP 拨杆	打开翻盖后操作。 控制 CODESYS 应用程序在运行（RUN）/停止（STOP）两种状态切换； 配合 RESET 按键可强制删除 CODESYS 应用程序
⑦	状态指示灯	RUN：运行指示灯，红/绿双色指示； ERR：错误指示灯，红色； LINK：本地扩展 I/O 模块指示灯，绿色； FB：EtherCAT 总线指示灯，绿色； 485：RS-485 通信指示灯，绿色； Us：本体供电电源指示灯，绿色； CAN：CAN 通信指示灯，绿色； Up：I/O 供电电源指示灯，绿色
⑧	输入通道指示灯	绿色，共 2 组，ON 时亮，OFF 时灭；数字与输入通道编号逐一对应
⑨	输出通道指示灯	绿色，共 2 组，ON 时亮，OFF 时灭；数字与输出通道编号逐一对应
⑩	USB	USB2.0，共 2 个，Type-A，可接 U 盘、USB 转串口模块等
⑪	数字量输入接口	共 16 通道，NPN 型
⑫	RS-485	RS-485 接口，支持 ModbusRTU 主从协议和自由协议
⑬	I/O 供电电源	24VDC 输入：供本体 I/O 及扩展 I/O 模块使用
⑭	CAN	CAN2.0A 接口，支持 CANopen 协议
⑮	数字量输出接口	共 16 通道，NPN 型
⑯	扩展 I/O 接口	用于扩展本地 ET 系列 I/O 模块，最多 16 个
⑰	终端盖板	保护扩展 I/O 接口，并稳定通信
⑱	本体电源输入接口	24VDC 输入，供控制器本体使用
⑲	PORT3 EtherCAT	EtherCAT 主站接口，10/100/1000Mbps

序号	名称	定义
⑳	PORT2 EtherNet	标准以太网功能，10/100/1000Mbps，支持 Modbus TCP 主从协议，支持 UDP/TCP 自由协议，OPC-UA 协议，PLCHandler 通信，用户程序下载与调试(只支持 IPv4)
㉑	PORT1 EtherNet	标准以太网功能，10Mbps，支持 Modbus TCP 主从协议，支持 UDP/TCP 自由协议，(只支持 IPv4)

3.3 安装

操作时，请阅读本手册中的【安全声明】，遵守本手册中的【安全注意事项】相关内容，并请注意以下事项：

（1）在安装设备时，请把产生高电压和高电子噪声的设备与运动控制器产品分隔开。

（2）当安装于控制柜时，建议把运动控制器产品安排在控制柜中温度较低的区域内以延长其使用寿命。

（3）布线时，尽量避免把交流供电线、高能量、开关频率很高的直流信号线与低压信号线、通信电缆设计在同一个线槽中。

（4）在运动控制器产品的上、下、左、右方向都必须留有合适的空间以便正常散热。

3.3.1 安装方向

SC301 运动控制器及 ET 系列扩展 I/O 系统，仅支持水平方向安装，以保证良好的散热条件。

TL 系列远程 I/O 系统，支持水平和垂直两个方向安装，但更推荐水平安装，以保证良好的散热条件。

3.3.2 安装间距

产品安装时，请务必考虑散热情况，保证运动控制器、耦合器及扩展 I/O 模块的距离 D＞30mm，如下图 3-3 所示。

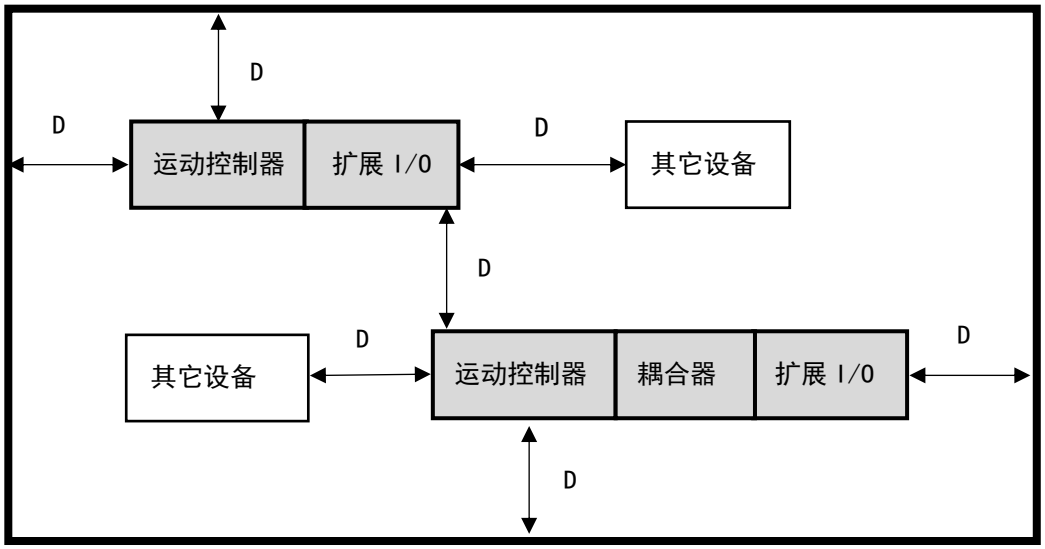


图 3-3 安装间距要求

3.3.3 安装方式

SC301 运动控制器及 ET 系列 I/O 系统仅持 DIN35 标准导轨安装一种方式，安装时请参考以下步骤进行操作：

（1）将 DIN35 导轨固定在控制柜内的安装面上；

（2）将控制器底部两侧的导轨卡扣向外侧拉出，如图 3-4；

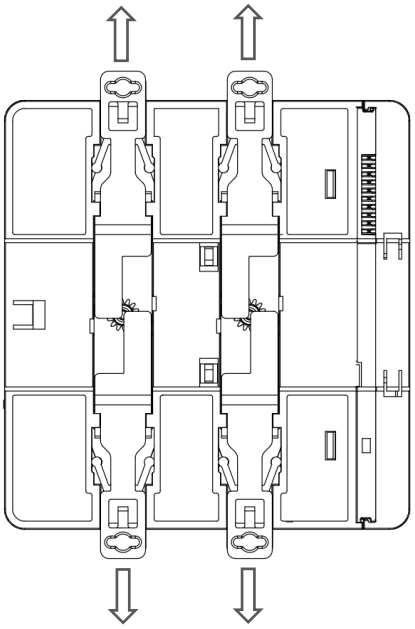


图 3-4 导轨卡扣向外侧拉伸

（3）将其沿着安装槽的上沿水平扣入导轨上沿，然后用力按压，确保产品紧嵌在 DIN35 导轨上，且没有歪斜现象；如图 3-5，然后将控制器底部两侧的导轨卡扣向内侧推动，使之与导轨紧密配合以确保锁紧，固定后效果如图 3-6；

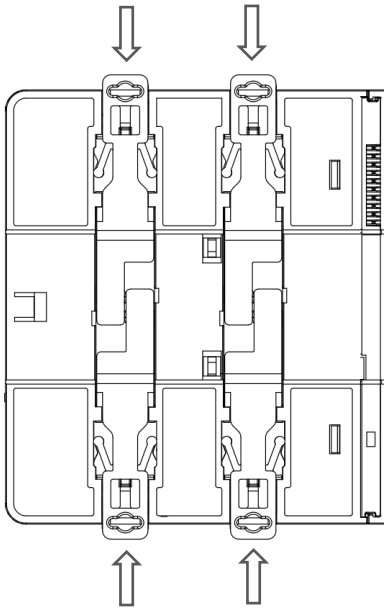


图 3-5 导轨卡扣向内侧推动

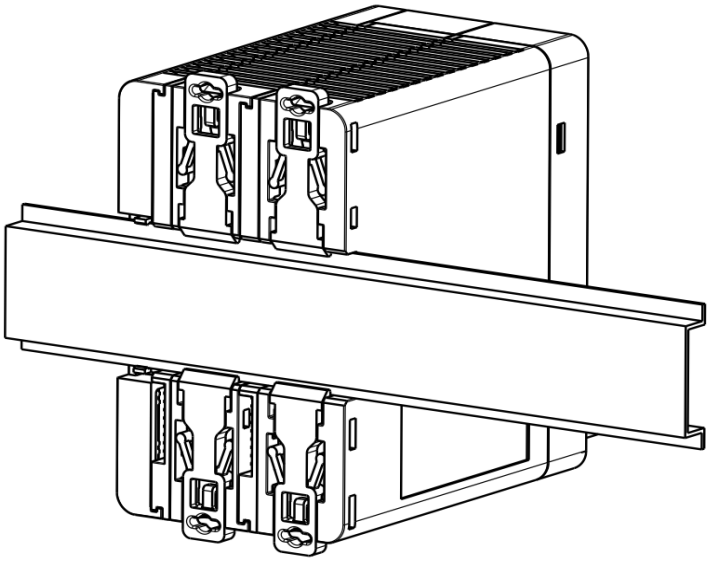


图 3-6 导轨卡扣固定后效果

当仅仅安装 SC301 运动控制器时，安装完毕后请确保控制器最右侧终端盖板 ET9000 也安装到位，以避免异物进入影响产品正常工作。

如下图 3-7，当安装的模块除了 SC301 运动控制器之外，还有 ET 系列扩展模块时，应先将控制器安装在导轨的最左侧，并取下右侧的 ET9000 终端盖板；然后将扩展模块逐个沿着导引槽垂直向下压入，最后将 ET9000 终端盖板安装在末端的扩展 I/O 模块上，然后将导轨卡扣锁紧。

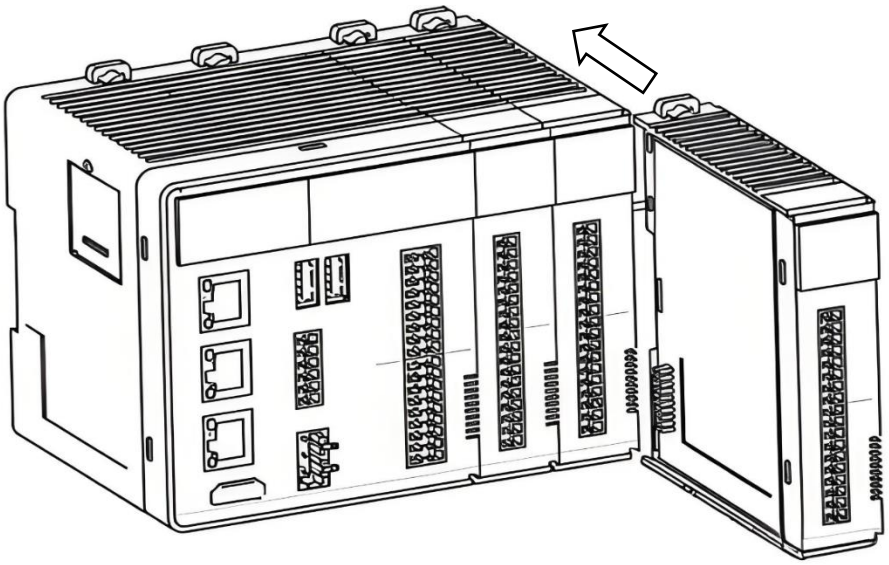


图 3-7 本地扩展模块安装示意

安装完毕后，建议将整个系统的两侧通过固定卡子进行锁附固定，避免在震动的情况下模块沿着导轨滑动或者导致连接不牢固影响产品正常工作。

如需搭配 TL 系列扩展 I/O 系统，其安装方式请参考《TL 系列远程 IO 系统用户手册》，该手册可以前往高创官网下载。

3.4 接线

运动控制器安装完毕以后，接下来按照本手册进行接线。在自动化应用现场，除了正确的接线之外，合理的接地能确保系统具备最优的操作特性，并为系统提供更好的噪声保护。

3.4.1 接线注意事项

（1）端子接线电缆布线时，避免与动力线（高电压、大电流）等产生强干扰信号的电缆捆在一起，应该分开走线并且避免平行走线；

（2）接线电缆建议选用屏蔽线缆以提高抗干扰能力；

（3）请对屏蔽线和焊封电缆的屏蔽做单点接地处理。

3.4.2 接地要求

运动控制器在安装时，请按照图 3-8 采用专用接地或者单点接地方式。

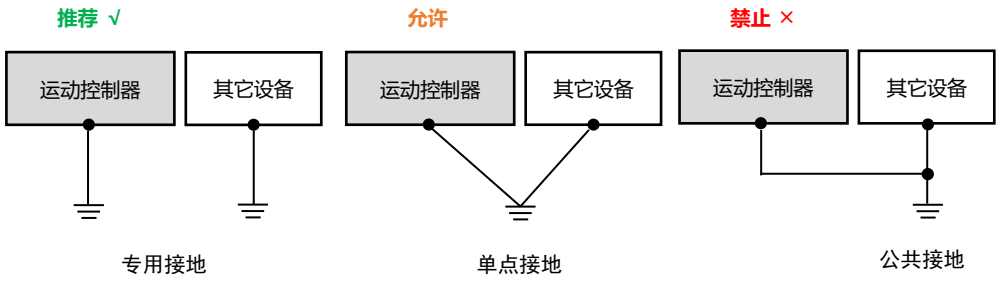


图 3-8 接地要求

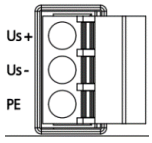
3.4.3 线缆选型与制作

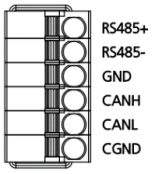
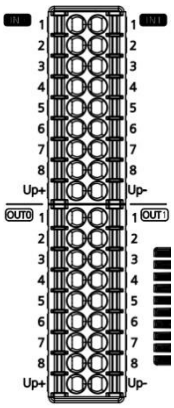
SC301 的采用了更加方便的免工具压接接线方式，且不同接口有不同的接线线缆规格，具体见下表 3-2，推荐的线径规格已经考虑了对应接口的过电流要求。

（1）对于单股硬导线，剥好对应长度的导线后，下压按钮同时将单线接入；

（2）对于多股柔性导线，剥好对应长度的导线后，将线捻成没有“线须”状态，配套使用对应标准规格的冷压端头，下压按钮同时将线接入。

表 3-2 接口线缆选型

接口	示意图	剥线长度	线径
本体供电电源		(9～10)mm	国标：(0.25～2.5) mm^2 美标：AWG23～AWG14

接口	示意图	剥线长度	线径
RS-485、CAN		(9～10)mm	国标：（0.2～1.5）mm^2 美标：AWG24～AWG16
I/O		(9～10)mm	国标：（0.5～1.5）mm^2 美标：AWG20～AWG16

4. 硬件详细规格

4.1 翻盖

如下图 4-1，翻盖位于产品正面左上方，用于保护内部的各个接口。翻盖内部有 MicroSD 卡接口、RESET 按键、RUN/STOP 拨杆。从翻盖上的 OPEN 字样处箭头所指位置向上撬动，即可打开翻盖进行操作。

图 4-1 翻盖

4.2 导轨卡扣

下图 4-2 为导轨卡扣，共有 4 个（2 对），卡扣采用齿轮式结构，用于将产品固定在 DIN35 导轨上。

图 4-2 导轨卡扣

4.3 纽扣电池仓盖

如下图 4-3 所示，SC301 产品侧面有纽扣电池仓盖，仓内装有纽扣电池，型号为 CR2032，用于给 RTC（实时时钟）供电，使用寿命为 2 年。当电池电量低时请及时更换电池。更换时首先将 SC301 断电，然后使用工具通过电池仓盖上方的

缺口撬开仓盖，拆除旧电池，更换新电池，盖上仓盖，再给 SC301 上电。

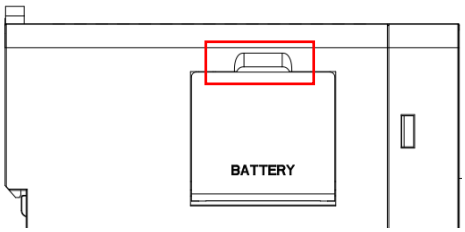


图 4-3 纽扣电池舱盖

【注意】：产品出厂时默认带有纽扣电池，第一次上电使用前，请先将纽扣电池底部的绝缘垫片抽出才能正常工作。

4.4 MicroSD 卡接口

MicroSD 卡接口位于控制器正面的翻盖仓内，打开翻盖即可操作，接口处印有“SD”标识，用于读写 MicroSD 卡。MicroSD 卡又称为 TF 卡，属于移动存储设备。

表 4-1 MicroSD 卡接口

序号	功能	操作描述	备注
1	数据存储	自行购买和使用 MicroSD 卡，可用于固件更新	出厂默认不配 MicroSD 卡

4.5 RESET 按键

RESET 按键位于控制器正面的翻盖仓内，打开翻盖即可操作，通过不同的操作可以实现多个不同的功能，见下表 4-2。

表 4-2 RESET 按键操作说明

序号	功能	操作描述	备注
1	恢复默认 IP	**概要说明**：系统处于正常运行状态后（**若新上电，则需要等待约 20S 使得系统完全启动**），长按 5S，恢复默认 IP 地址。控制器三个网口的 IP 地址同时会恢复。 **操作步骤**：正常运行状态下，长按 RESET 按键约 5S 不要松开，在 RUN 红灯以 500ms 亮/500ms 灭的频率闪烁期间（共 5 次）松开 RESET 键，将会恢复控制器所有网口默认的 IP 地址，需重新上电方可生效。 **【注意】：操作时，注意避免 RESET 按键抖动或者意外断电。**	如果修改了控制器默认的 IP 地址后又忘记，导致无法正常连接的，建议进行该操作
2	正常删除程序	**概要说明**：系统处于正常运行状态后（**若新上电，则需要等待约 20S 使得系统完全启动**），长按 10S，删除 CODESYS 应用程序。 **操作步骤**：正常运行状态下，长按 RESET 约 10S 不要松开，在 RUN 红灯以 1s 亮/1s 灭频率闪烁期间（共 5 次）松开 RESET 键，将会删除控制器中的 CODESYS 应用工程。 **【注意】：操作时，注意避免 RESET 按键抖动或者意外断电。**	该功能实现的前提是系统正常运行，如果应用程序导致系统崩溃，则该功能可能失效；若系统崩溃，建议强制删除程序
3	强制删除程序	**概要说明**：当 CODESYS 程序异常导致系统崩溃时，强制删除 CODESYS 程序。 **操作步骤**：先将控制器断电，再将 RUN/STOP 拨杆拨到 STOP 位置，然后给控制器上电，此后 Linux 系统开始正常启动。等到 RUN 指示灯变为黄色（红绿灯同时亮）期间（约持续 5S）立刻长按 RESET 按键 5S，当 RUN 绿灯熄灭且 RUN 红灯以 250ms 亮/250ms 灭开始闪烁期间（共 5 次），即可松开 RESET 按键，此时控制器中的 CODESYS 应用工程即被强制删除。 **【注意】：操作时，注意避免 RESET 按键抖动或者意外断电。**	如果遇到严重 CODESYS 程序错误导致系统崩溃，尝试该方法

4.6 RUN/STOP 拨杆

RUN/STOP 拨杆位于控制器正面的翻盖仓内，打开翻盖即可操作，主要用于手动控制应用程序在运行（RUN）和停止（STOP）两种状态下切换，以及配合 RESET

按键强制删除应用程序。

【注意】:

（1）如果在拨动过一次或者多次 RUN/STOP 拨杆之后，又登录 CODESYS 工程改变应用程序进行下载，则以 CODESYS 编程软件界面上的 Start/Stop 按钮为准。

（2）如果在线登录 CODESYS 工程，点击 CODESYS 编程软件界面中的 Start 之后，还是处于 STOP 状态，请检查硬件这个拨杆是否处于 STOP 状态。

表 4-3 RUN/STOP 拨动开关操作

序号	功能	操作描述	说明
1	程序运行	拨杆推到 RUN（上）位置	相当于单击 CODESYS 软件界面中(Start) ▶ 的功能，此时 RUN 运行指示灯变为绿色常亮； 若硬件一直处于 STOP 状态，则也可以在在线模式下，点击 Start 开始运行。
2	程序停止	拨杆推到 STOP（下）位置	相当于单击 CODESYS 软件界面中（Stop）■ 的功能，此时 RUN 指示灯变为红色常亮。 若硬件一直处于 RUN 状态,则也可以在在线模式下，点击 Stop 停止运行，或单步调试。
3	强制删除 CODESYS 应用程序	先将控制器断电，再将 RUN/STOP 拨杆拨到 STOP 位置,然后给控制器上电,此后 Linux 系统开始正常启动。等到 RUN 指示灯变为黄色（红绿灯同时亮）期间（约持续 5S)立刻长按 RESET 按键 5S，当 RUN 绿灯熄灭且 RUN 红灯以 250ms 亮/250ms 灭开始闪烁期间（共 5 次），即可松开 RESET 按键	**【注意】:需要与 RESET 按键配合操作才可以完成，**请参考“**4.5 RESET 按键**”章节。

4.7 系统状态指示灯

用于指示系统供电、运行、停止、故障、通信等状态，如下图 4-4 所示，共有 8 颗指示灯，其中 RUN 指示灯为红/绿双色，其余为单色。

图 4-4 系统状态指示灯

各指示灯的颜色、状态及含义定义见下表 4-4。

表 4-4 系统状态指示灯说明

指示灯	颜色	状态	含义描述
RUN（双色）	红色	常亮	①系统启动过程中（硬件默认控制，微亮） ②控制器处于停止状态 ③掉电保持数据时
		500ms 亮/500ms 灭频率闪烁 5 次	系统正常运行状态下，长按 RESET 按键约 5S 不要松开，在 RUN 红灯以 500ms 亮/500ms 灭的频率闪烁期间（共 5 次）松开 RESET 键，将会恢复控制器所有网口默认的 IP 地址
		1s 亮/1s 灭频率闪烁 5 次	系统正常运行状态下，长按 RESET 约 10S 不要松开，在 RUN 红灯以 1s 亮/1s 灭频率闪烁期间（共 5 次）松开 RESET 键，将会删除控制器中的 CODESYS 应用工程
		250ms 亮/250ms 灭频率闪烁 5 次	先将控制器断电，再将 RUN/STOP 拨杆拨到 STOP 位置，然后给控制器上电，此后 linux 系统开始正常启动。等到 RUN 指示灯变为黄色（红绿灯同时亮）期间（约持续 5S）立刻长按 RESET 按键 5S，当 RUN 绿灯熄灭且 RUN 红灯以 250ms 亮/250ms 灭开始闪烁期间（共 5 次），即可松开 RESET 按键，将会强制删除控制器中的 CODESYS 应用工程
		灭	控制器正常运行
	绿色	常亮	①系统启动过程中（硬件默认控制，微亮） ②控制器正常运行状态（CODESYS 应用程序正常运行）
		500ms 亮/500ms 灭频率持续闪烁	无 CODESYS 应用程序

指示灯	颜色	状态	含义描述
		灭	①Linux 系统未运行 ②CODESYS 启动异常
ERR	红色	常亮	控制器 CODESYS 故障
		灭	控制器正常运行
LINK	绿色	常亮	本地扩展 I/O 模块通信正常
		灭	①初始化异常 ②不存在本地扩展 I/O 模块 ③本地扩展 I/O 模块通信异常
FB	绿色	常亮	EtherCAT 总线通信正常
		1 秒/次频率闪烁	EtherCAT 总线通信异常
485	绿色	常亮	RS-485 收发数据正常
		1 秒/次频率闪烁	RS-485 仅接收或者仅发送数据
		灭	RS-485 无数据通信
Us	绿色	常亮	控制器本体供电电源（Us+/Us-）正常
		灭	控制器本体供电电源（Us+/Us-）异常
CAN	绿色	常亮	CAN 总线收发数据正常
		灭	CAN 总线收发数据异常
Up	绿色	常亮	I/O 供电电源（Up+/Up-）正常
		灭	I/O 供电电源（Up+/Up-）异常

4.8 数字量输入通道状态指示灯

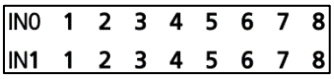


图 4-5 数字量输入通道状态指示灯

SC301 本体自带 16 通道数字量输入，非高速，分为 2 组，如上图 4-5 所示，输入通道状态指示灯用于指示输入通道的 ON/OFF 状态，具体说明见表 4-5。

表 4-5 数字量输入通道状态指示灯

序号	组别	描述
1	IN0	绿色，数字 1～8 指示灯分别对应 IN0 组数字量输入通道 1～8 的 ON/OFF 状态，ON 时亮，OFF 时灭
2	IN1	绿色，数字 1～8 指示灯分别对应 IN1 组数字量输入通道 1～8 的 ON/OFF 状态，ON 时亮，OFF 时灭

4.9 数字量输出通道状态指示灯

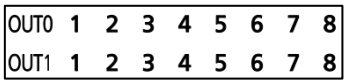


图 4-6 数字量输出通道状态指示灯

SC301 本体自带 16 通道数字量输出，非高速，分为 2 组，如上图 4-6 所示，输出通道状态指示灯用于指示输出通道的 ON/OFF 状态，具体说明见表 4-6。

表 4-6 数字量输出通道状态指示灯

序号	组别	描述
1	OUT0	绿色，数字 1～8 指示灯分别对应 OUT0 组数字量输出通道 1～8 的 ON/OFF 状态，ON 时亮，OFF 时灭
2	OUT1	绿色，数字 1～8 指示灯分别对应 OUT1 组数字量输出通道 1～8 的 ON/OFF 状态，ON 时亮，OFF 时灭

4.10 USB 接口

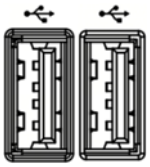


图 4-7 USB 接口

如上图 4-7，SC301 本体自带 2 个 USB Host 接口，形式为 Type-A，符合 USB 2.0 及 USB EHCI 标准，均支持即插即用和热插拔功能。通过这 2 个 USB 接口，可以实现下表 4-7 中的功能。

表 4-7 USB 接口功能描述

序号	功能	描述	备注
1	U 盘接入	当插入 U 盘时，系统会自动挂载 U 盘到/udisk 目录下	支持 NTFS/FAT32 格式，如果 U 盘容量在 32G 以上，先分区，并对小于 32G 的分区进行格式化

序号	功能	描述	备注
2	USB 转串口模块接入	插入 USB 转串口模块，会自动映射到 COM5、COM6 端口。第 1 个插入的模块端口映射为 COM5，第 2 个插入的模块端口映射为 COM6（编号与操作顺序有关，与位置无关）	重新插拔后，COM 端口号会发生变化，以实际为准

图 4-8 USB2.0 接口针脚编号

USB2.0 接口的针脚编号如上图 4-8 所示，各个针脚信号定义如下表 4-8 所示。

表 4-8 USB2.0 接口 Type-A 针脚信号定义

针脚	信号名称	描述
1	VBUS	电源（+5V）
2	D-（Data-）	USB2.0 差分数据对
3	D+（Data+）	
4	GND	接地（信号地）

4.11 数字量输入接口

SC301 本体自带 16 个通道的数字量输入，分为 2 组，如下图 4-9 所示。

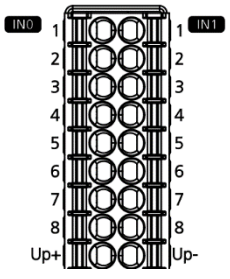


图 4-9 数字量输入接口

4.11.1 数字量输入接口规格

表 4-9 数字量输入接口规格

项目	规格
输入类型	NPN
输入电压等级	24VDC(最高 28.8VDC)
ON 电压	> 15VDC
OFF 电压	< 5VDC
隔离方式	数字隔离
输入状态指示	输入有效时点亮
输入通道	16
最高输入频率	1kHz
输入电流(典型)	5.3mA
输入滤波时间	不可配置
输入阻抗	4.3kΩ

4.11.2 数字量输入接口信号定义

表 4-10 数字量输入接口信号定义

组别	信号名	定义
IN0	1	IN0 组输入通道 1，NPN，普通
	2	IN0 组输入通道 2，NPN，普通
	3	IN0 组输入通道 3，NPN，普通
	4	IN0 组输入通道 4，NPN，普通
	5	IN0 组输入通道 5，NPN，普通
	6	IN0 组输入通道 6，NPN，普通
	7	IN0 组输入通道 7，NPN，普通
	8	IN0 组输入通道 8，NPN，普通

组别	信号名	定义
	Up+	本体 I/O 及扩展 I/O 模块 24VDC 供电电源（+），IN0 组与 IN1 组共用
IN1	1	IN1 组输入通道 1，NPN，普通
	2	IN1 组输入通道 2，NPN，普通
	3	IN1 组输入通道 3，NPN，普通
	4	IN1 组输入通道 4，NPN，普通
	5	IN1 组输入通道 5，NPN，普通
	6	IN1 组输入通道 6，NPN，普通
	7	IN1 组输入通道 7，NPN，普通
	8	IN1 组输入通道 8，NPN，普通
	Up-	本体 I/O 及扩展 I/O 模块 24VDC 供电电源（-），IN0 组与 IN1 组共用

4.11.3 数字量输入接口信号接线说明

数字量输入接口的接线示意图和接线原理图分别见图 4-10、图 4-11。

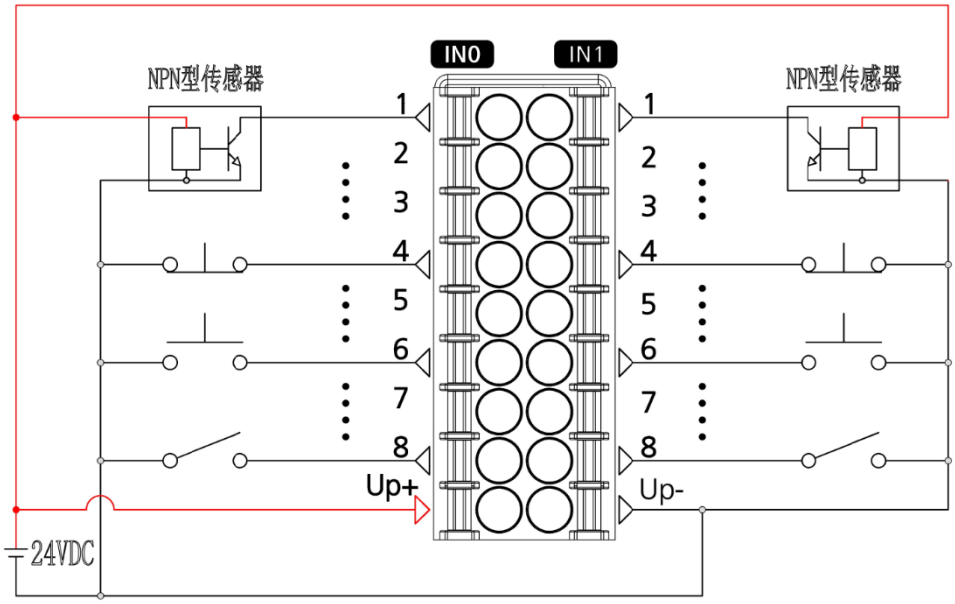


图 4-10 数字量输入接口接线示意图

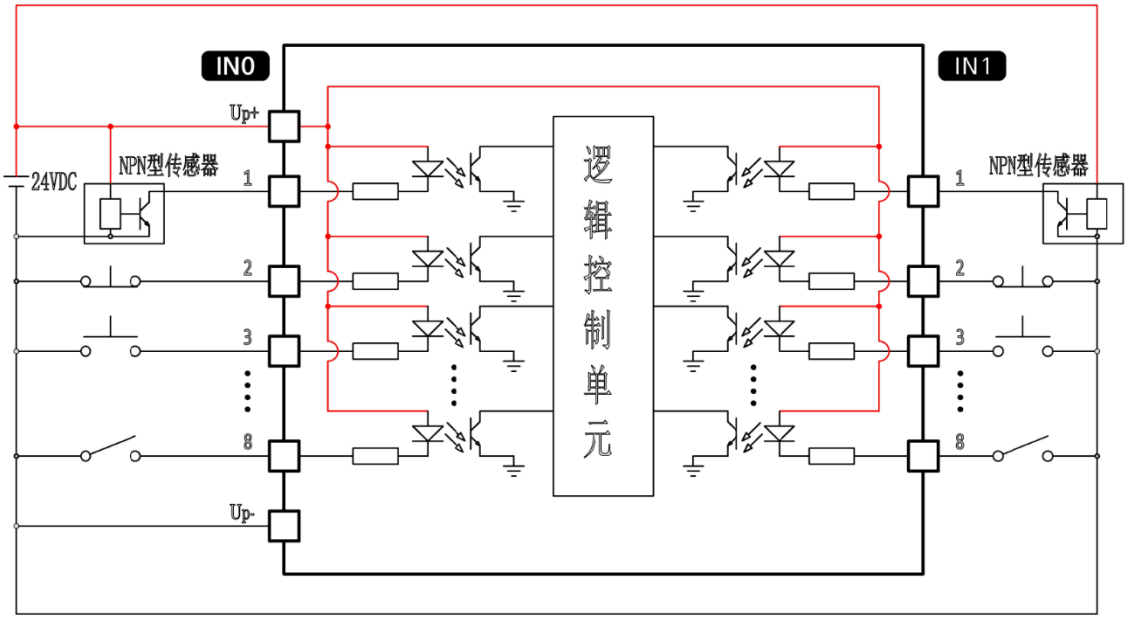


图 4-11 数字量输入接口接线原理图

4.12 RS-485 接口

SC301 集成了 1 个 RS-485 串行通信接口，可以方便地与第三方设备进行通信。RS-485 接口支持 Modbus RTU 主站、从站协议，以及自由协议。根据需要选择合适的协议进行通信，但同一时刻只能选择其中的一种协议。

4.12.1 RS-485 接口描述

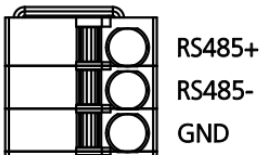


图 4-12 RS-485 接口

如上图 4-12，为 RS-485 串行通信接口，其接口信号定义见下表 4-11。

表 4-11 RS-485 接口信号定义

序号	丝印	信号
1	RS485+	差分信号的+端（常用 A 表示）
2	RS485-	差分信号的-端（常用 B 表示）
3	GND	信号地

4.12.2 RS-485 基本特性

表 4-12 RS-485 的基本特性

序号	特性	说明
1	COM 编号	4，编程时使用
2	组网方式	1:N，N:1，N:N，作为主站时至多支持 16 个从站
3	数据通信方式	半双工
4	通信速率 bps	共 6 种：4800、9600、19200、38400、57600、115200
5	通信距离	与通信速率成反比，在不加中继器的情况下，4800bps 至多可以传输 1219 米，但同时还与网络节点数、线缆线径、线缆直线电阻率、线缆的特性阻抗有关，不能一概而论
6	隔离方式	数字隔离
7	通信线缆	建议使用特性阻抗为 120Ω的屏蔽双绞线，屏蔽层单点接地
8	终端电阻	**SC301 内置了 120Ω的 RS-485 终端电阻**，适用于以下场景： ①当其作为主站，且处于总线最远端； ②当其作为从站，且处于总线最远端。

4.12.3 RS-485 拓扑结构

RS-485 总线上节点数量较多时，采用合适的拓扑结构将有助于提升通信的稳定性，推荐采用菊花链拓扑结构，可以采用“总线式”拓扑结构但应避免分支过长，不推荐采用“星型”拓扑结构。

(1)菊花链

菊花链（俗称“手拉手”）拓扑结构，如下图 4-13 所示，总线两侧的最远端接入 120Ω的终端电阻以防止信号反射。**【注意】：SC301 已经内置 120Ω电阻，无需重复接入。**

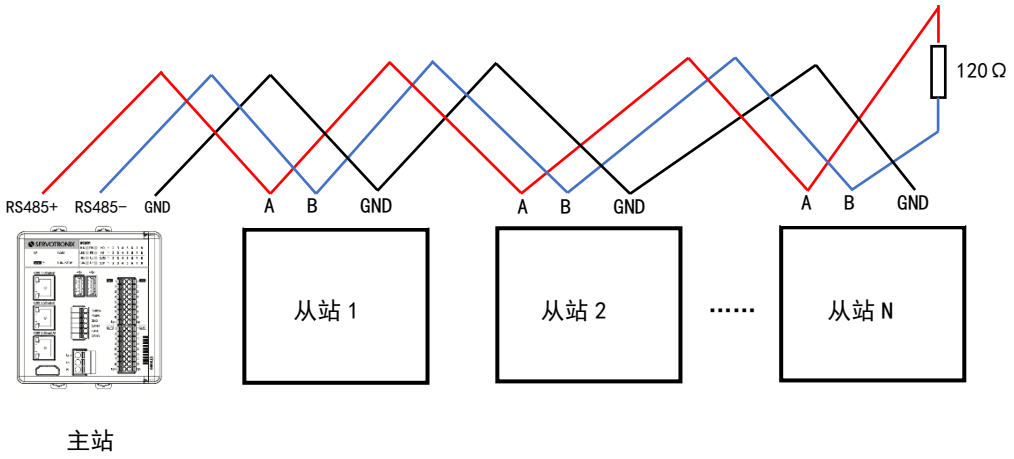


图 4-13 RS-485 菊花链拓扑结构示意

(2) 总线式

当采用“总线式”拓扑结构，且通过分支连接从站时，应确保分支不要超过 3m 长度，如下图 4-14 所示。**【注意】：SC301 已经内置 120Ω电阻，无需重复接入。**

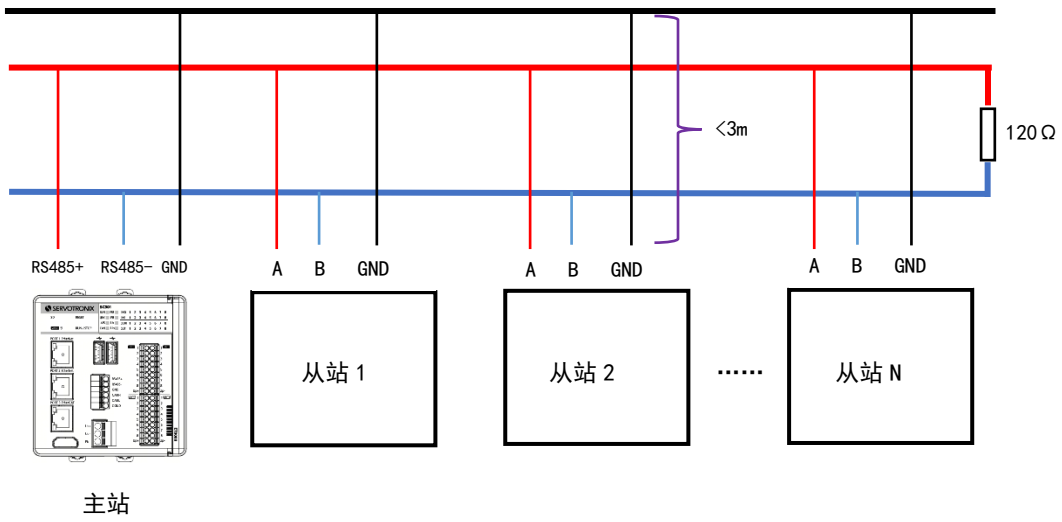


图 4-14 RS-485 总线式拓扑结构示意

(3)星型

星型拓扑结构对各分支的等长要求及终端电阻都有较为苛刻的要求，处理不当极易造成通信无法正常工作，因此不推荐采用星型拓扑结构。

4.12.4 接线注意事项

推荐采用特征阻抗为 120Ω的屏蔽双绞线作为 RS-485 的通信介质，并且屏蔽层单端接地。

由于 RS-485 为差分信号传输，总线通过检测 RS485+（A）与 RS485-（B）之间的电压差值，作为判定 0 或者 1 信号的标准。当从站 RS-485 接口具有信号地时，可以将其与 SC301 的 RS-485 接口的 GND 连接在一起，使得各个站点之间具有相同的参考信号地；当从站 RS-485 接口无信号地时，SC301 的 RS-485 接口的 GND 可以悬空不接。

4.13 I/O 供电电源

如图 4-15，SC301 提供了 2 组 I/O 供电电源输入接口，分别位于数字量输入端子和数字量输出端子，用于给本体 I/O 以及本体扩展 I/O 模块供电。**数字量输入侧的 Up+、Up-和数字量输出侧的 Up+、Up-已经在产品内部连接。**

图 4-15 I/O 供电电源

4.14 CAN 接口

SC301 集成了 1 个 CAN 通信接口，可以方便地与第三方 CAN 设备进行通信。CAN 接口支持标准 CANopen 主站协议和自由协议。

4.14.1 CAN 接口描述

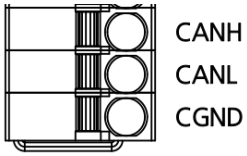


图 4-16 CAN 接口

如上图 4-16，为 CAN 通信接口，其接口信号定义见下表 4-13。

表 4-13 CAN 接口信号定义

序号	丝印	信号
1	CANH	CAN_H
2	CANL	CAN_L
3	CGND	CAN_GND（信号地）

4.14.2 CAN 的基本特性

表 4-14 CAN 的基本特性

<table>
<tr><th>序号</th><th>特性</th><th colspan="2">说明</th></tr>
<tr><td>1</td><td>组网方式</td><td colspan="2">1 主多从</td></tr>
<tr><td>2</td><td>数据通信方式</td><td colspan="2">半双工</td></tr>
<tr><td>3</td><td>通信速率 bps</td><td colspan="2">共 9 种：10k、20k、50k、100k、125k、250k、500k、800k、1000k</td></tr>
<tr><td rowspan="6">4</td><td rowspan="6">通信距离</td><td colspan="2">与通信速率成反比，但同时还与网络节点数、线缆线径、线缆直线电阻率、线缆的特性阻抗有关，不能一概而论，可以参考下值。</td></tr>
<tr><td>波特率（bps）</td><td>通信距离（m）</td></tr>
<tr><td>10k</td><td>5000</td></tr>
<tr><td>20k</td><td>2500</td></tr>
<tr><td>50k</td><td>1000</td></tr>
<tr><td>100k</td><td>650</td></tr>
<tr><td>125k</td><td>500</td></tr>
</table>

序号	特性	说明	
		250k	250
		500k	100
		800k	40
		1000k	30
5	隔离方式	数字隔离	
6	通信线缆	建议使用特性阻抗为 120Ω的屏蔽双绞线，屏蔽层单点接地	
7	终端电阻	**SC301 内置了 120Ω的终端电阻**，适用于以下场景： ①当其作为主站，且处于总线最远端； ②当其作为从站，且处于总线最远端。	

4.14.3 CAN 总线的拓扑结构

CAN 总线上节点数量较多时，采用合适的拓扑结构将有助于提升通信的稳定性，推荐采用菊花链拓扑结构，可以采用“总线式”拓扑结构但应避免分支过长，不推荐采用“星型”拓扑结构。

（1）菊花链

菊花链（俗称“手拉手”）拓扑结构，如下图 4-17 所示，总线两侧的最远端接入 120Ω的终端电阻以防止信号反射。**【注意】：SC301 已经内置 120Ω电阻，无需重复接入。**

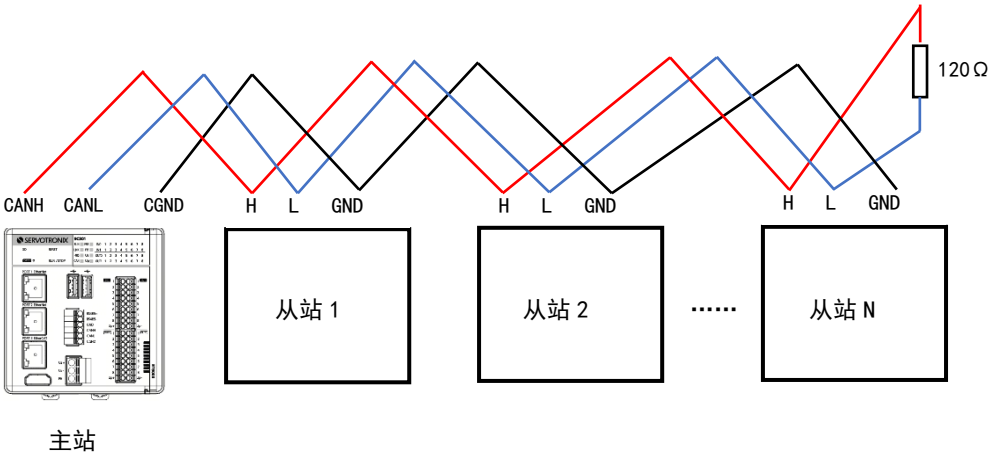


图 4-17 CAN 总线菊花链拓扑结构示意

(2) 总线式

当采用“总线式“拓扑结构，且通过分支连接从站时，应确保分支不要超过 0.3m 长度，如下图 4-18 所示。**【注意】：SC301 已经内置 120Ω 电阻，无需重复接入。**

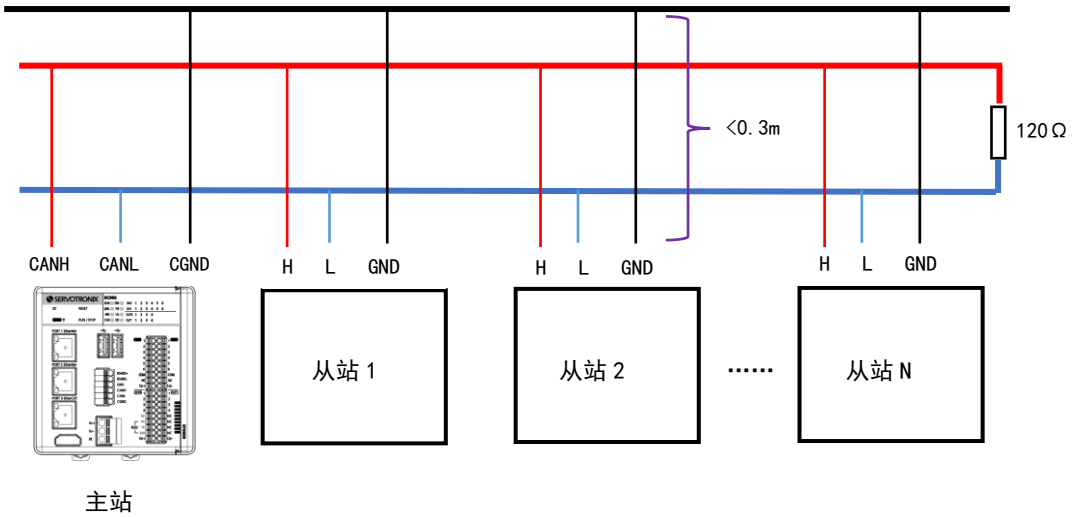


图 4-18 CAN 总线式拓扑结构示意

(3) 星型

星型拓扑结构对各分支的等长要求及终端电阻都有较为苛刻的要求，处理不当极易造成通信无法正常工作，因此不推荐采用星型拓扑结构。

4.14.4 接线注意事项

推荐采用特征阻抗为 120Ω 的屏蔽双绞线作为 CAN 的通信介质，并且屏蔽层单点接地。

由于 CAN 为差分信号传输，总线通过检测 CAN_H 与 CAN_L 之间的电压差值，作为判定 0 或者 1 信号的标准。当从站 CAN 接口具有信号地时，可以将其与 SC301 的 CAN 接口的 CGND 连接在一起，使得各个站点之间具有相同的参考信号地；当从站 CAN 接口无信号地时，SC301 的 CAN 接口的 CGND 可以悬空不接。

4.15 数字量输出接口

SC301 本体自带 16 个通道的数字量输出，分为 2 组，如下图 4-19 所示。

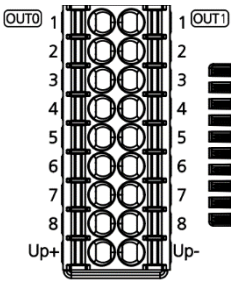


图 4-19 数字量输出接口

4.15.1 数字量输出接口规格

表 4-15 数字量输出接口规格

项目	规格	
输出通道	16 个，普通输出	
输出信号	晶体管，NPN（漏）型	
电源电压	24VDC （20.4VDC～28.8VDC）	
输出电压等级	24VDC （20.4VDC～28.8VDC）	
最大输出电流	500mA/通道	
OFF 时最大漏电流	<100uA	
最大负载	电阻负载	0.5A/点
	感性负载	7.2W/点
	电灯负载	5W/点
隔离方式	数字隔离	
输出状态指示	输出为驱动状态时亮	
短路保护	有	

4.15.2 数字量输出接口信号定义

表 4-16 数字量输出接口信号定义

组别	信号名	定义
OUT0	1	OUT0 组输出通道 1，NPN，普通
	2	OUT0 组输出通道 2，NPN，普通
	3	OUT0 组输出通道 3，NPN，普通
	4	OUT0 组输出通道 4，NPN，普通
	5	OUT0 组输出通道 5，NPN，普通
	6	OUT0 组输出通道 6，NPN，普通
	7	OUT0 组输出通道 7，NPN，普通
	8	OUT0 组输出通道 8，NPN，普通
	Up+	本体 I/O 及扩展 I/O 模块 24VDC 供电电源（+），OUT0 组与 OUT1 组共用
OUT1	1	OUT1 组输出通道 1，NPN，普通
	2	OUT2 组输出通道 2，NPN，普通
	3	OUT3 组输出通道 3，NPN，普通
	4	OUT4 组输出通道 4，NPN，普通
	5	OUT5 组输出通道 5，NPN，普通
	6	OUT6 组输出通道 6，NPN，普通
	7	OUT7 组输出通道 7，NPN，普通
	8	OUT8 组输出通道 8，NPN，普通
	Up-	本体 I/O 及扩展 I/O 模块 24VDC 供电电源（-），OUT0 组与 OUT1 组共用

4.15.3 数字量输出接口信号接线说明

数字量输出接口的接线示意图和接线原理图分别见图 4-20、图 4-21。

【注意】：接感性负载时要外接续流二极管，二极管可选用 1N4007 或类似参数的二极管，极性务必不能接反，否则将导致负载短路，烧毁输出通道。

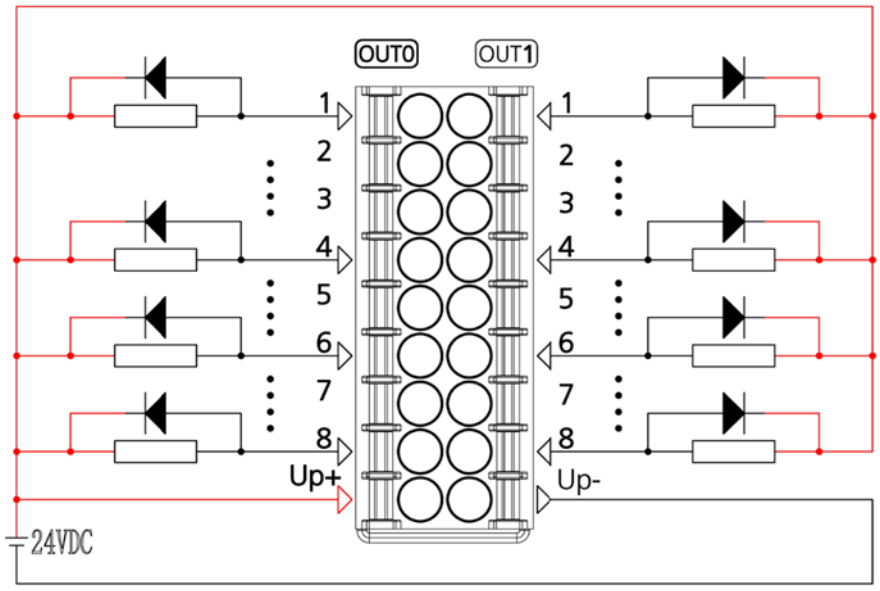


图 4-20 数字量输出接口接线示意图

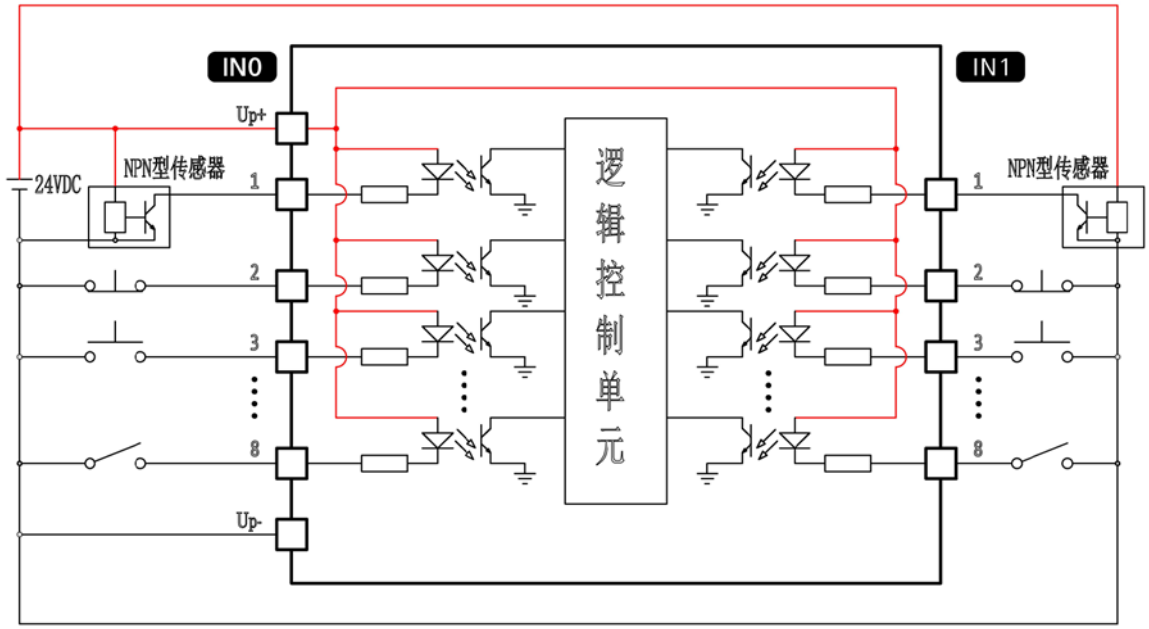


图 4-21 数字量输出接口接线原理图

4.16 扩展 I/O 接口

下图 4-22 是 SC301 的本体扩展 I/O 接口，当无扩展 I/O 时，需要接入 ET9000 终端盖板对接口进行保护。

图 4-22 本体 I/O 扩展接口

4.17 终端盖板

如下图 4-23，SC301 的最右侧为终端盖板，型号为 ET9000，用于保护扩展 I/O 接口并稳定通信。

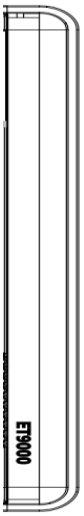


图 4-23 终端盖板

4.18 本体电源输入接口

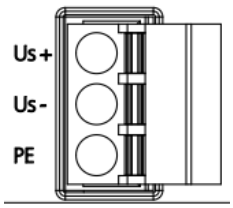


图 4-24 本体电源输入接口

如上图 4-24 所示，为本体 24VDC 电源输入接口，输入范围为（22～30）VDC，输入电流≥0.5A，其信号定义如下表 4-17 所示。

表 4-17 本体电源输入接口信号定义

序号	丝印	信号
1	Us+	24VDC 正
2	Us-	24VDC 负
3	PE	大地

4.19 PORT3 EtherCAT

下图 4-25 为 PORT3 网络接口，该接口物理层为 1000BASE-TX(千兆)，因此可以实现 10/100/1000Mbps 的通信速率，该接口将作为 EtherCAT 主站，专注于 EtherCAT 通信，与作为 EtherCAT 从站的伺服驱动器连接实现运动控制，与 EtherCAT 耦合器连接实现远程 I/O 扩展等。常见伺服驱动器的通信速率一般为 100Mbps。

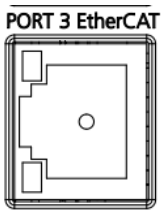


图 4-25 EtherCAT 接口

当不需要 EtherCAT 通信时，还可以当作普通 Ethernet 接口进行使用。为了突出其 EtherCAT 功能，该接口丝印选择了“EtherCAT”，但并不局限于 EtherCAT。

4.19.1 PORT3 EtherCAT 接口概述

表 4-18 PORT3 EtherCAT 接口概述

序号	项目	说明
1	网卡编号	eth2（**注意：与丝印编号不同**）
2	接口形式	RJ-45
3	默认 IP 地址	192.168.39.220
4	子网掩码	255.255.255.0
5	通信速率	10/100/1000Mbps
6	指示灯	**橙色指示灯：用于指示连接与活动（Link/Act）状态** ①线路连接失败：熄灭； ②线路连接成功：常亮； ③连接成功且有数据通信：闪烁。 **绿色/红色指示灯：用于指示通信速率(Speed)状态** ①10Mbps 通信：红绿灯均熄灭； ②100Mbps 通信：绿灯亮； ③1000Mbps 通信：红灯亮
7	协议支持	EtherCAT Master； TCP/UDP/IP； OPC-UA Server； WebVisu Server； Modbus TCP Master/Slave

4.19.2 通信速率及通信介质

由于该网口物理层具备 1000Mbps 的通信能力，因此在通信速率、通信介质的选择及制作时需要注意。

当进行 EtherCAT 通信时：

（1）推荐选择 100Mbps 通信速率以更好地发挥其性能；

（2）选择 1000Mbps 时需要通信的对方也具有 1000Mbps 的通信能力，且采用 EtherCAT G 技术。

表 4-19 通信速率选择与网线制作

通信速率	网线	接线线序	信号线
10/100Mbps	Cat.5(五类)及以上屏蔽双绞	推荐 T568B	使用 1、2、3、6 信号线，4、5、7、8 非必需但可以增强抗干扰能力；
1000Mbps	Cat.5e(超五类)及以上屏蔽双绞		使用 1～8 全部 8 根信号线

4.19.3 EtherCAT 接口规格

由于 PORT3 接口主要用作 EtherCAT 通信，因此重点介绍其作为 EtherCAT 接口的规格。

表 4-20 EtherCAT 接口规格

序号	项目	说明
1	通信协议	EtherCAT Master
2	支持服务	CoE (PDO/SDO)
3	最小同步周期	1ms@4 轴，2ms@8 轴，4ms@16 轴
4	同步方式	伺服采用 DC 时钟，I/O 采用输入、输出同步
5	物理层	1000BASE-TX
6	波特率	支持 1000Mbps，但 100Mbps 更常用

序号	项目	说明
7	双工方式	全双工
8	拓扑方式	线型、星型、树型、菊花链或者以上的混合
9	传输媒介	推荐采用 Cat.5 及以上的屏蔽网线
10	传输距离	两节点间<100M
11	从站数	65535
12	EtherCAT 帧长度	不含 EtherCAT 帧头后为（44 ～1498）字节； 但其总长度仍符合以太网（64～1518）字节帧长度范围
13	过程数据	单个帧中数据区最大为 1486 字节（去掉子报文头和 WKC 字节）
14	两个从站的同步抖动	<1us
15	刷新时间	1000 个开关量输入输出约 30us(32 个伺服约 100us)

4.20 PORT2 EtherNet

下图 4-26 为 PORT2 网络接口，该接口物理层为 1000BASE-TX(千兆)，因此可以实现 10/100/1000Mbps 的通信速率。

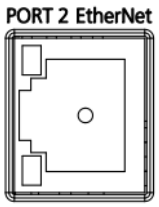


图 4-26 PORT2 EtherNet

推荐使用该网口用作 CODESYS 应用程序的下载调试接口，以获得更佳使用体验。

4.20.1 PORT2 EtherNet 接口概述

表 4-21 PORT2 EtherNet 接口概述

序号	项目	说明
1	网卡编号	eth0（**注意：与丝印编号不同**）
2	接口形式	RJ-45
3	默认 IP 地址	192.168.0.1
4	子网掩码	255.255.255.0
5	网关地址	192.168.0.171
6	通信速率	10/100/1000Mbps
7	指示灯	**橙色指示灯：用于指示连接与活动（Link/Act）状态** ①线路连接失败：熄灭； ②线路连接成功：常亮； ③连接成功且有数据通信：闪烁。 **绿色/红色指示灯：用于指示通信速率(Speed)状态** ①10Mbps 通信：红绿灯均熄灭； ②100Mbps 通信：绿灯亮； ③1000Mbps 通信：红灯亮
8	协议支持	TCP/UDP/IP; OPC-UA Server; WebVisu Server; Modbus TCP Master/Slave; PLCHandler

4.20.2 PORT2 通信速率及通信介质

由于该网口物理层具备 1000Mbps 的通信能力，因此在通信速率、通信介质的选择及制作时需要注意:

（1）推荐选择 1000Mbps 的通信速率以更好地发挥其性能；

（2）选择 1000Mbps 时需要通信的对方也具有 1000Mbps 的通信能力。

表 4-22 通信速率选择与网线制作

通信速率	网线	接线线序	信号线
10/100Mbps	Cat.5(五类)及以上，屏蔽更佳	推荐 T568B	使用 1、2、3、6 信号线，4、5、7、8 非必需但可以增强抗干扰能力；
1000Mbps	Cat.5e(超五类)及以上，屏蔽更佳		使用 1～8 全部 8 根信号线

4.21 PORT1 EtherNet

下图 4-27 为 PORT1 网络接口，该接口物理层为 100BASE-TX(百兆)，因此具备 10/100Mbps 的通信能力。

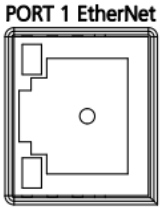


图 4-27 PORT1 EtherNet

【注意】：由于该接口在硬件上采用了 SPI 转以太网的实现方式，实测通信速率在 10Mbps 左右，故不推荐使用该网口做大数据量通信。

4.21.1 PORT1 EtherNet 接口概述

表 4-23 PORT1 EtherNet 接口概述

序号	项目	说明
1	网卡编号	eth1
2	接口形式	RJ-45
3	默认 IP 地址	90.0.0.1
4	子网掩码	255.255.255.0
5	通信速率	10Mbps
6	指示灯	**橙色指示灯：用于指示连接（Link）状态** ①物理连接失败，熄灭； ②物理连接成功：常亮。 **绿色指示灯：用于指示通信活动(Active)状态** ①无数据通信：熄灭； ②有数据通信：闪烁
7	协议支持	TCP/UDP/IP; Modbus TCP Master/Slave

4.21.2 PORT1 通信速率及通信介质

表 4-24 通信速率与网线制作

通信速率	网线	接线线序	信号线
10/100Mbps	Cat.5(五类)及以上，屏蔽更佳	推荐 T568B	使用 1、2、3、6 信号线，4、5、7、8 非必需但可以增强抗干扰能力；

5. CODESYS 软件获取与安装

5.1 软件获取

CODESYS 软件为免费软件，可在高创传动官网 www.servotronix.cn 首页，依次单击“资料下载中心”->“运动控制器&I/O 模块”->“控制器编程与工具软件”->“CODEESYS 编程软件”， 如下图 5-1 所示，在弹出的页面中点击“下载”按钮进行下载，并保存在合适位置。也可以联系高创销售或者技术支持人员进行获取。

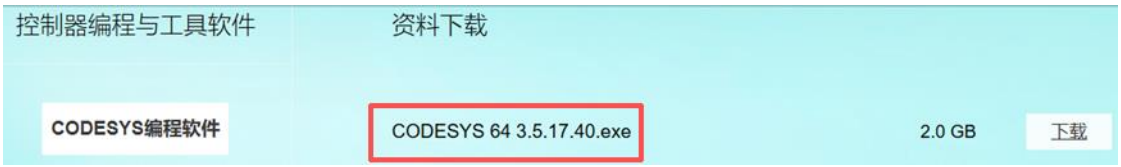


图 5-1 CODESYS 软件下载页面

5.2 软件安装步骤

5.2.1 安装前准备

在开始安装前，确保您已经获得了 CODESYS V3.5 SP17 Patch4 的软件安装包。

安装软件环境的台式计算机或者便携计算机要求具备以下条件：

（1）Window10 操作系统，推荐使用 64bit 操作系统；

（2）内存：8GB 或者更高配置；

（3）硬盘空间：可用空间 20G 以上；

（4）推荐电脑 CPU 主频在 2GHz 以上，否则会影响运行速度；

（5）建议关闭 Windows 的防火墙功能。

首次安装 CODESYS 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 20GB 以上之后，直接安装即可。

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

5.2.2 安装过程

安装时，建议将计算机联网进行。使用管理员权限点击安装包，选择好安装目录，点击 Next，直到安装完毕。

6. 安装设备描述文件并连接控制器

CODESYS 软件安装完以后，使用 CODESYS 软件编写应用程序，编译正确后，就可以通过以太网（可经过集线器、交换机等网络设备），将程序下载到控制器中运行了。

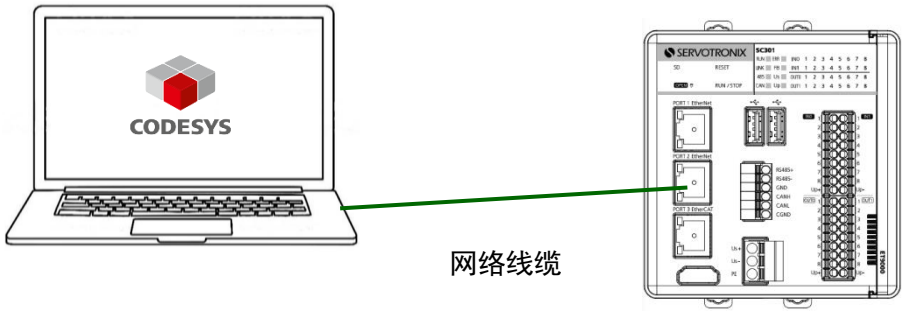


图 6-1 控制器通过网线连接计算机

6.1 下载和安装 SC301 的设备描述文件

在使用 SC301 之前需要先在 CODESYS 软件上安装其对应的设备描述文件，设备描述文件可在高创传动官网 www.servotronix.cn 首页，依次单击“资料下载中心”->“运动控制器&I/O 模块”->“紧凑型运动控制器”->“SC301”->“设备描述文件”，在弹出的页面选择进行下载，并保存在合适位置。

SC301 的设备描述文件名称为：Servotronix SC301 ARM SM WV dynamic Vx.x.x.devdesc.XML，其中 Vx.x.x 为版本号。后续版本可能会有更新，请留意。

如下图 6-2，点击菜单栏“工具”，在弹出的下拉列表中选择“设备存储库…”。

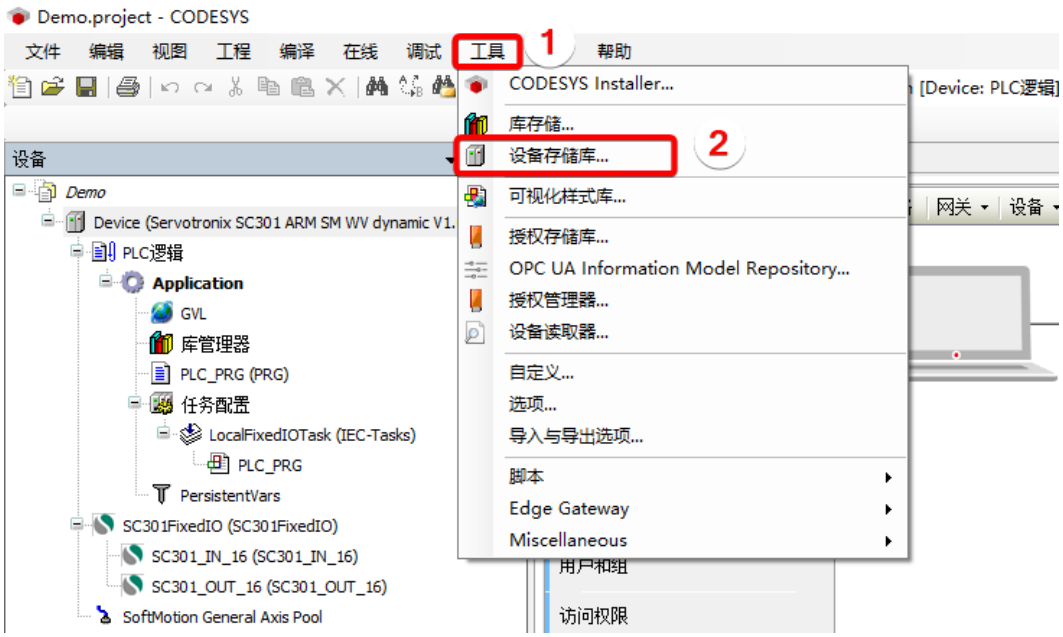


图 6-2 打开“设备存储库…”

如下图 6-3，在随后弹出的“设备存储库”窗口，点击“安装(I)…”按钮。

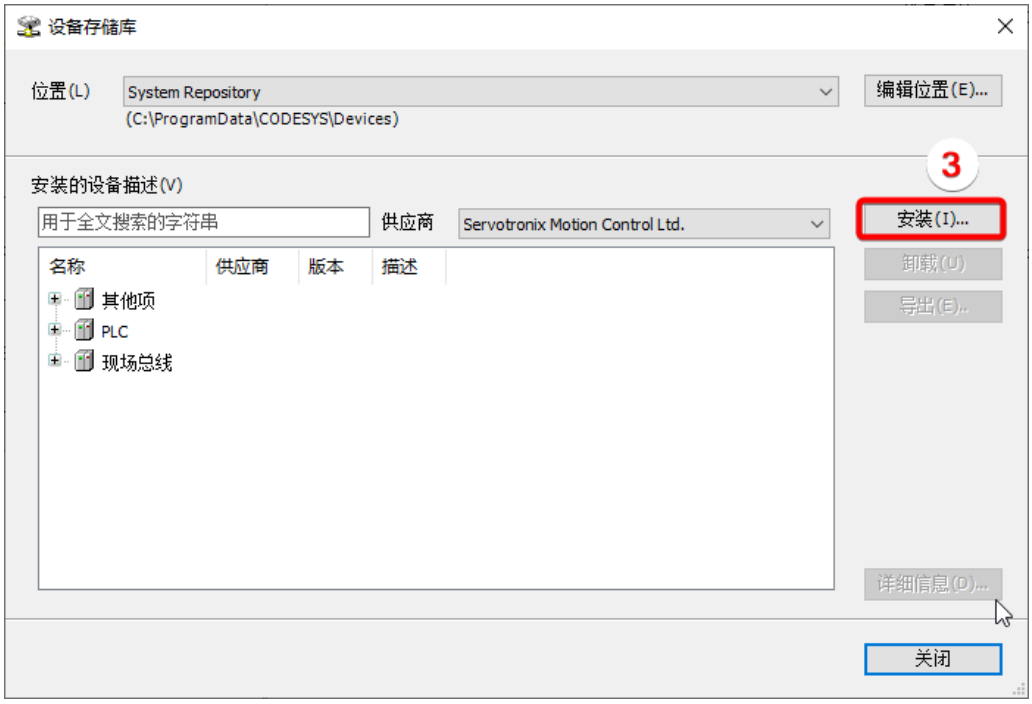


图 6-3 点击“安装（I）…”

如下图 6-4，在随后弹出的“安装设备描述”窗口，选中要安装的设备描述文件，然后点击下方“打开（O）”按钮，即可安装成功。

图 6-4 选择要安装的设备描述文件

如下图 6-5，随后可以在“设备存储库”页面中的“PLC”->“SoftMotion PLC”下即可看到“Servotronix SC301 ARM SM WV dynamic V1.0.1”。

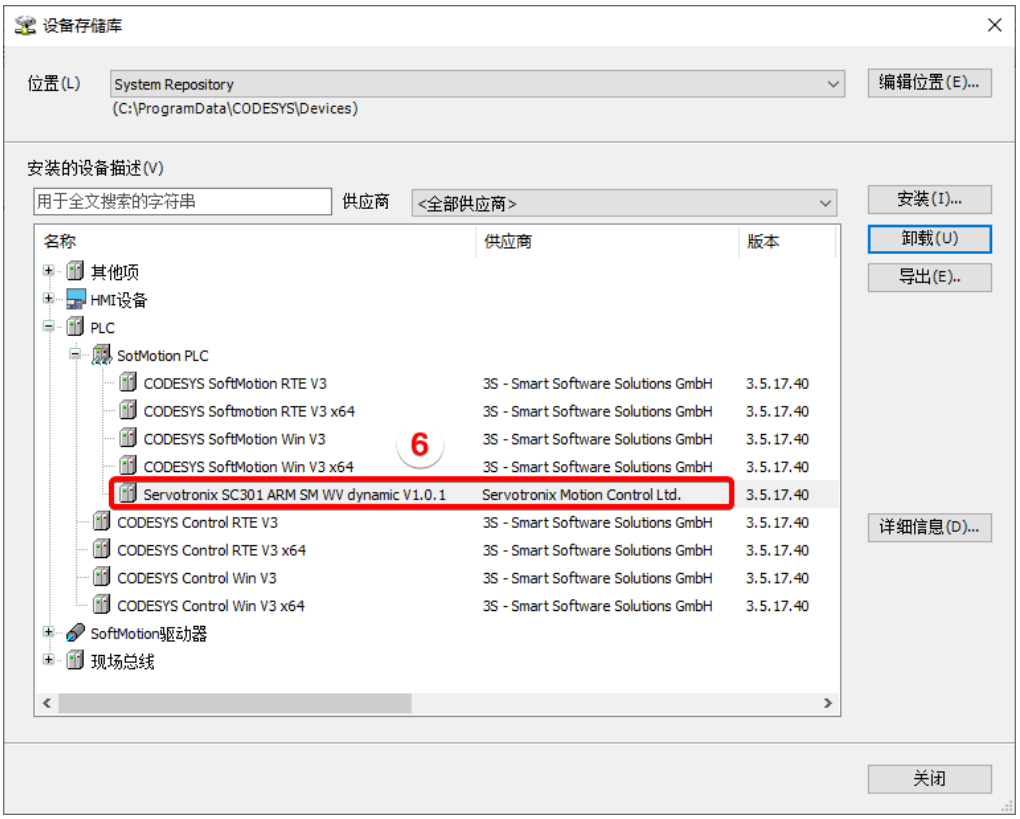


图 6-5 SC301 在设备存储库位置

6.2 设置正确的 IP 地址

SC301 具有两个普通以太网接口，PORT1 和 PORT2，均可以使用。由于 **PORT1 EtherNet 网口通信速率仅有 10Mbps，为了更好的使用体验，推荐使用 PORT2 EtherNet 网口下载 CODESYS 应用程序。**

【注意】：初次使用本产品，为了更轻松地先掌握扫描和连接控制器的方法，建议暂时不要修改控制器的默认 IP 地址。后续如有修改 IP 地址需求，请参考本手册相关章节操作。

如所使用的计算机网口的 IP 地址与所连接的控制器网口的 IP 地址不处于同一个网段，则在连接之前需先修改计算机网口 IP 地址，使之与所使用的控制器网口 IP 地址处于同一个网段。比如：控制器 PORT2 默认 IP 地址为 192.168.0.1，子网掩码是 255.255.255.0，则 PC 端的 IP 地址需要修改为 192.168.0.XX(XX 不能是 1，即不能是控制器的 IP 地址，XX 也不能用 0 和 255)。如下图 6-6 所示。

图 6-6 与控制器连接的计算机网络 IP 设置

6.3 扫描并连接控制器

打开 CODESYS IDE，然后按照下图 6-7 步骤就可以扫描控制器，进行应用程序的下载了。

（1）双击工程窗口的 Device，会弹出配置页面；

（2）选中配置页面的“通信设置”选项卡；

（3）点击页面左上方的“扫描网络”，然后弹出“选择设备”页面；

（4）在“选择设备”页面右上方的“扫描网络”按钮；

（5）选中正确识别的控制器，点击下方“OK”按钮。

至此，CODESYS 与控制器已经成功建立连接。

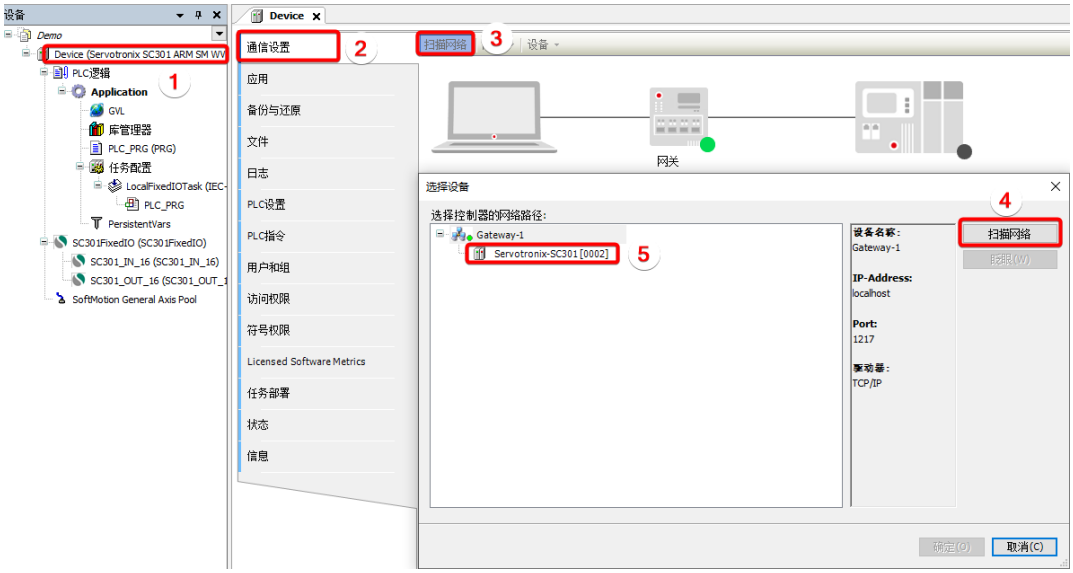


图 6-7 CODESYS 在线扫描控制器

需要说明的是，CODESYS 是个通用软件平台，所有使用这个软件平台的硬件产品以及这个硬件产品具有的一些功能模块，均需要通过添加 XML 设备描述文件来告知软件，从而可以识别、配置和使用。

例如：

（1）通过添加 SC301 的设备描述文件，COSESYS 可以识别 SC301 控制器设备，然后可以给其添加 EtherCAT Master SoftMotion 软件功能，然后再可以添加 SoftMotion CiA402 伺服轴设备。

（2）通过给 SC301 添加本体 I/O 设备描述文件，使其识别和使用 I/O 硬件。

7. 配置伺服驱动器

对于 SC301 而言，常见的应用场景之一是通过 EtherCAT 接口控制伺服驱动器，本章将介绍如何基于 CODESYS 软件配置 SC301 和 EtherCAT 伺服驱动器。

7.1 添加 EtherCAT Master SoftMotion

如下图 7-1，右键单击设备树中的“Device（Servotronix SC301…)”，在弹出的下拉列表中选择“添加设备…”。

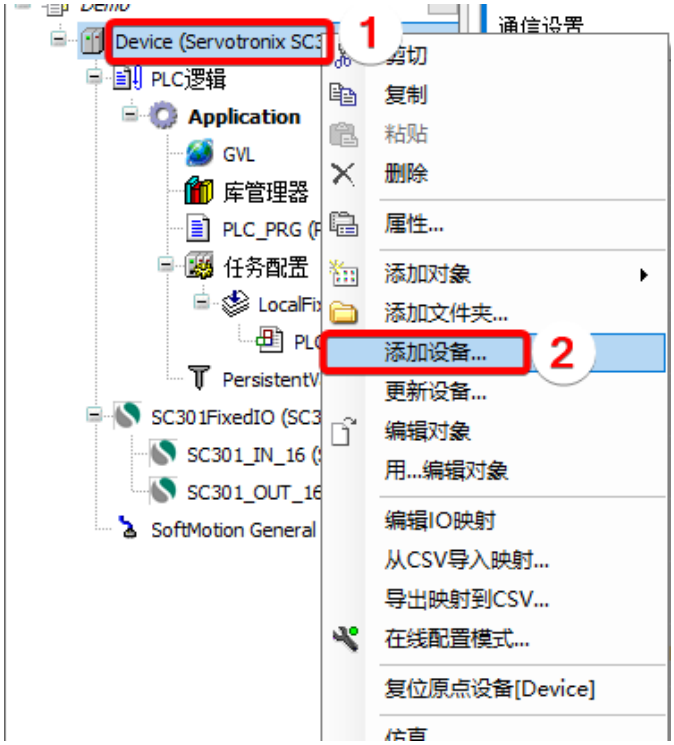


图 7-1 右键“Device”添加设备

如下图 7-2，然后在“添加设备…”页面中，选择“EtherCAT Master SoftMotion”，然后单击“添加设备”按钮，完成添加，然后关闭窗口。

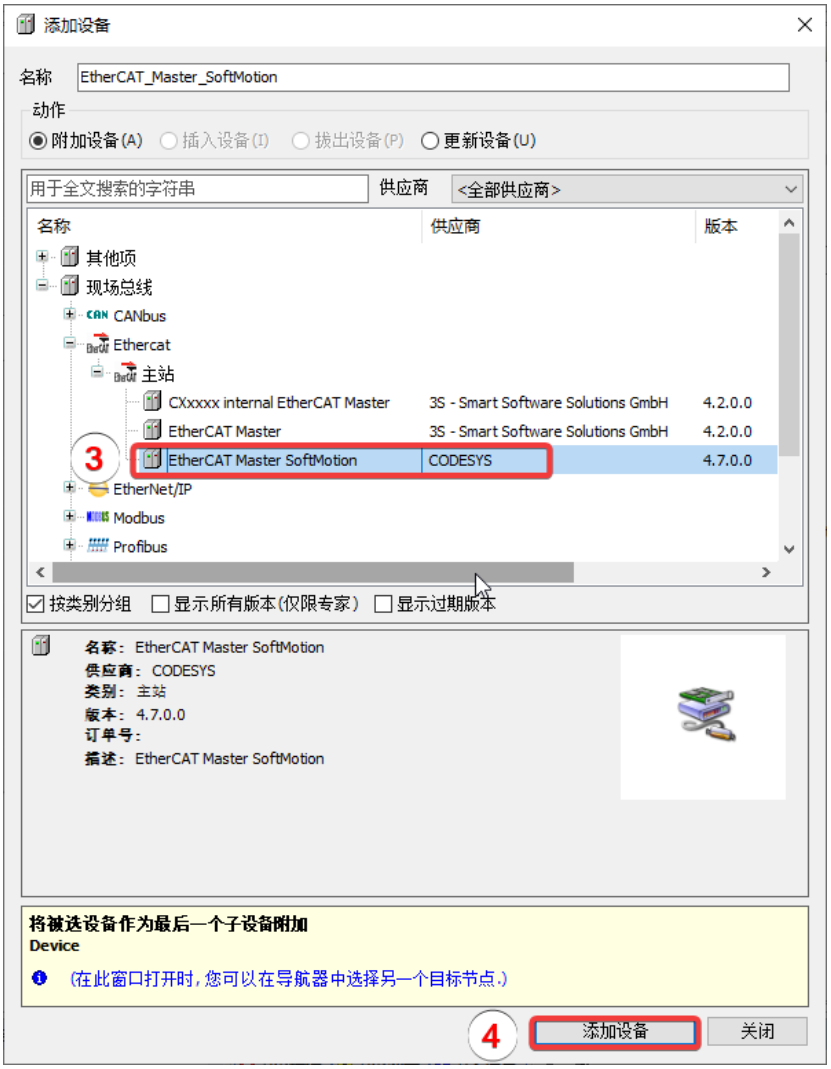


图 7-2 添加 EtherCAT Master SoftMotion

如下图 7-3，添加完成后，CODESYS 软件左侧工程窗口的设备树中会出现 EtherCAT_Master_SoftMotion，而且会在“任务配置”下自动创建一个名为 EtherCAT_Task(IEC-Tasks)的任务。

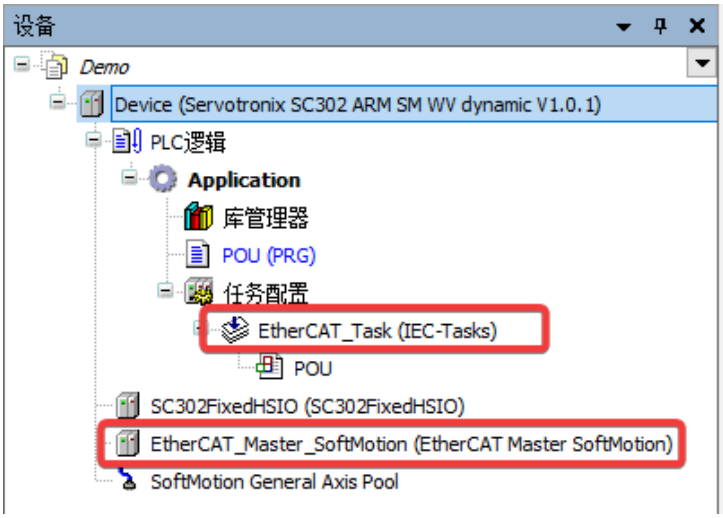


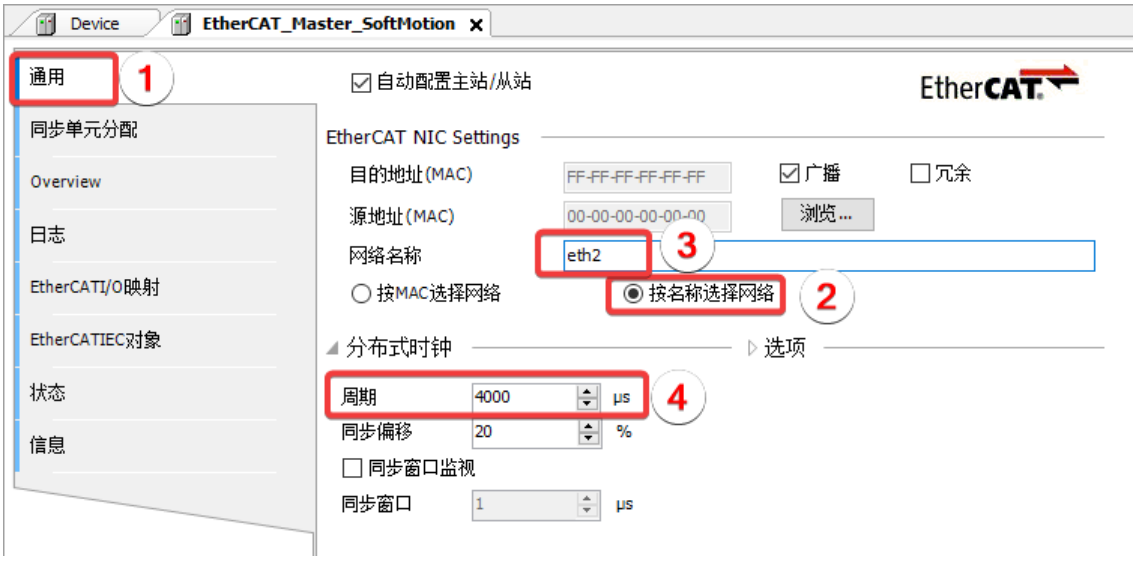
图 7-3 EtherCAT Master SoftMotion 添加成功

7.2 配置 EtherCAT_Master_SoftMotion

如图 7-4，双击设备树中的 EtherCAT_Master_SoftMotion，在弹出的页面中选择“通用”选项卡，然后 EtherCAT NIC Settings 中选择“按名称选择网络”，并在网络名称处填入“eth2”。“eth2”为 Port3 EtherCAT 网口对应的 SC301 控制器硬件接口编号，用于连接 EtherCAT 从站设备。

然后设置 EtherCAT 总线周期，SC301 控制器支持不同的扫描周期，用户可根据控制需求并结合连接从站的数量多少选择合适的总线周期。

通常情况下，总线周期时间越短，可以驱动的伺服数量越少，总线周期时间越长，可以驱动的伺服数量越多。一般应用场景下 2ms～4ms 均能满足，不建议低于 1ms。



7.3 登录控制器

点击“登录”按钮登录控制器，如下图 7-5。

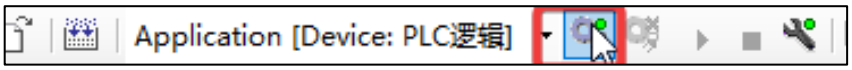


图 7-5 登录控制器

之后会弹出对话框，如下图 7-6，点击“是”。

图 7-6 下载确认

7.4 扫描设备

先登录 CODSYS，然后在设备树中右键单击 EtherCAT_Master_SoftMotion，选择“扫描设备…”，如图 7-7 所示。

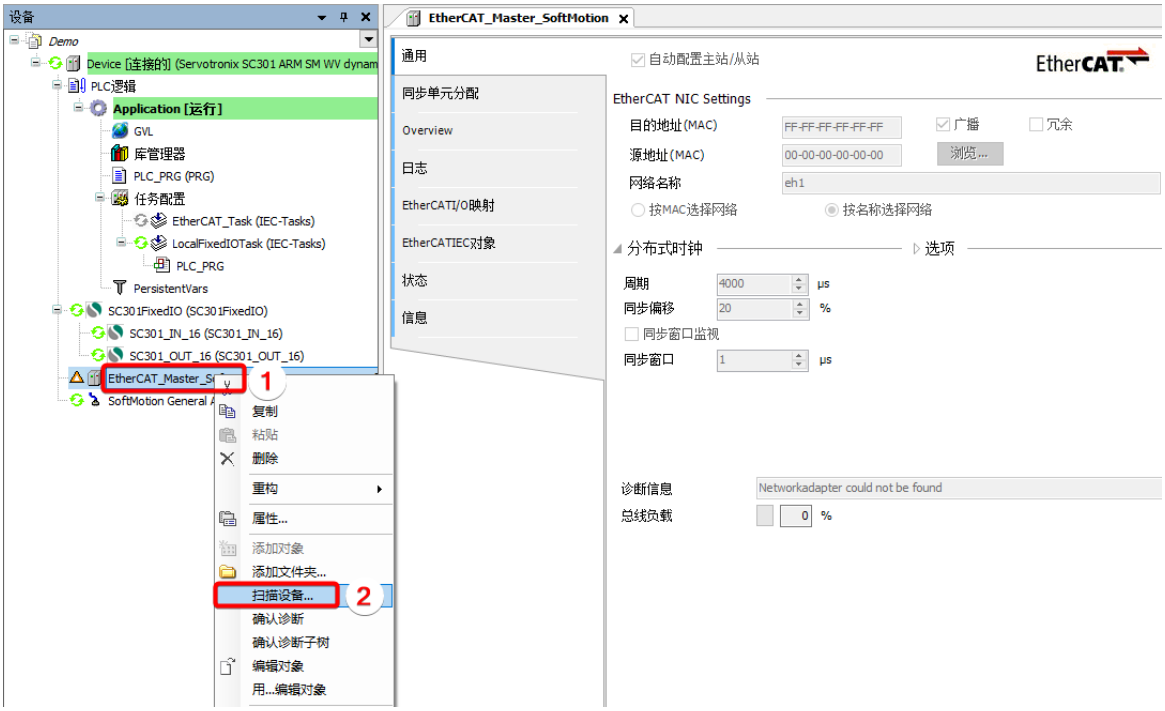


图 7-7 右键 EtherCAT_Master_SoftMotion 扫描设备

如下图 7-8，在弹出的界面中选中所有设备，单击“复制所有设备到工程”，添加成功后在 EtherCAT_Master_SoftMotion 下会出现对应的设备，至此伺服驱动器已经添加到工程中。

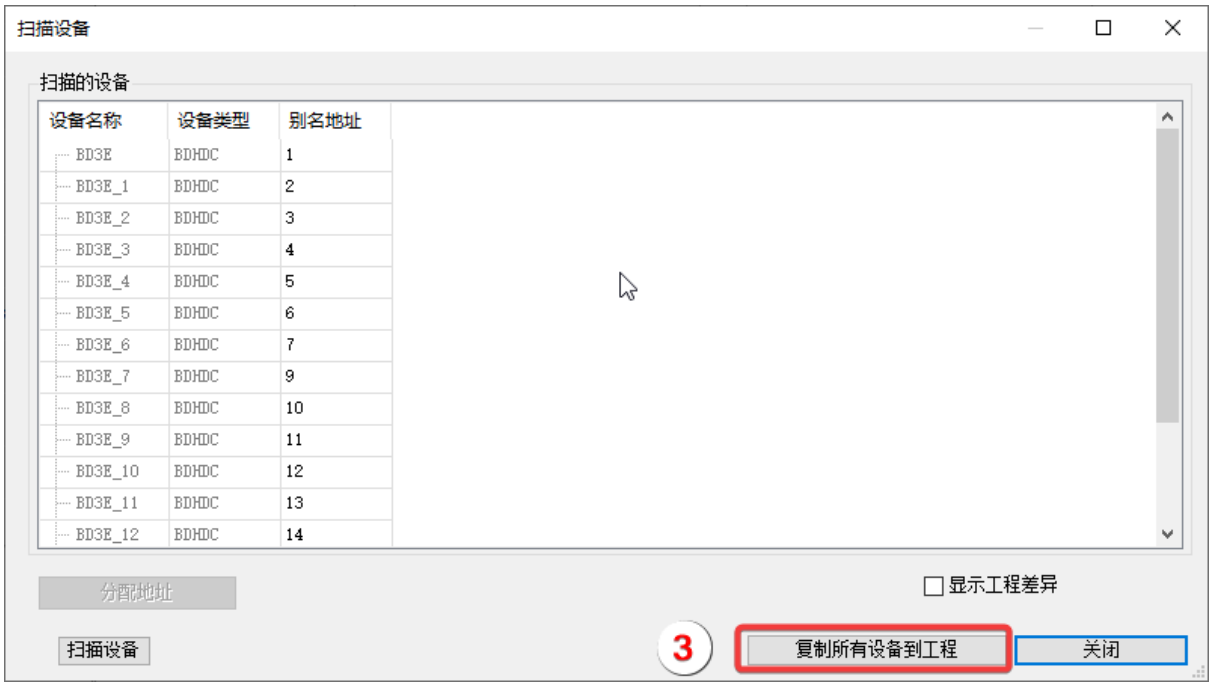


图 7-8 复制所有设备到工程

7.5 添加 SoftMotion CiA402 轴

如下图 7-9，在设备树窗口右键单击任意一个伺服设备，在弹出的窗口中选择“添加 SoftMotion CiA402 轴”，若有多个伺服设备，请按照此方法逐个添加。

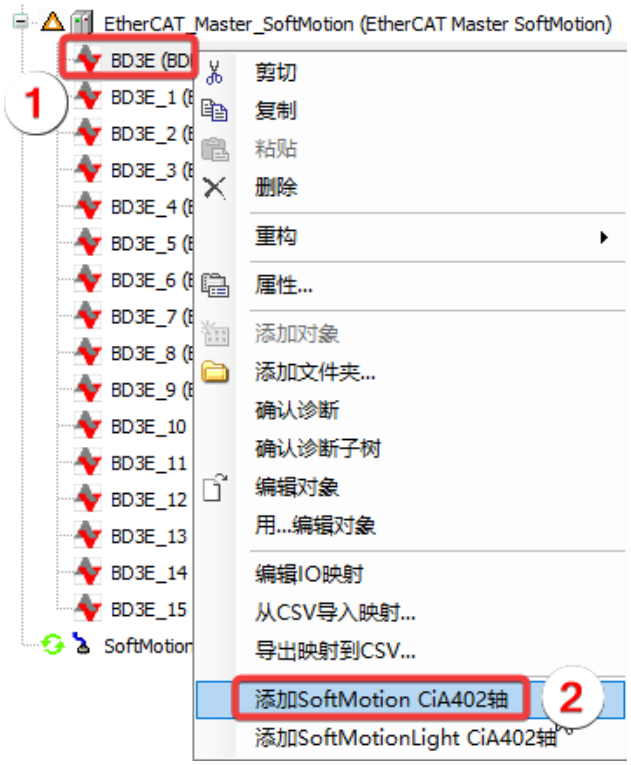


图 7-9 右键伺服设备添加 CiA402 轴

7.6 配置 CiA402 轴参数

根据实际的电机参数配置电机的分辨率，以及根据实际的需要配置减速比大小，如下图 7-10。

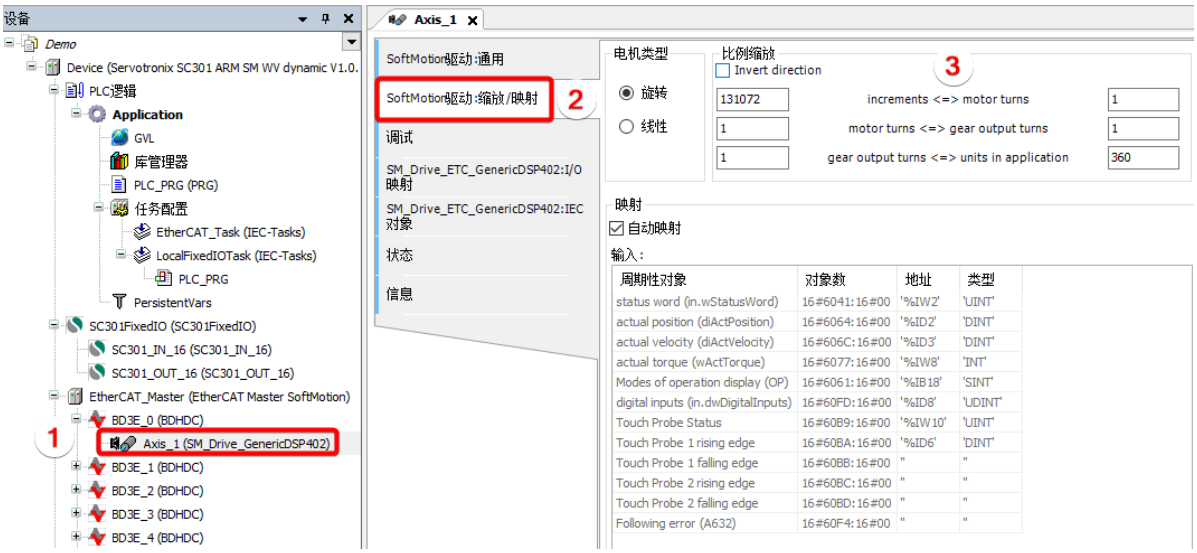


图 7-10 配置 CiA402 轴参数

8. 本体 I/O（SC301FixedIO）

本章节将重点介绍 SC301 本体上的 I/O（SC301FixedIO）的功能及使用方法。

SC301 本体支持 16 路数字量输入，均为普通输入；16 路数字量输出，均为普通输出。

如下图 8-1，红色方框口标记的是输入通道，绿色方框口标记的是输出通道。

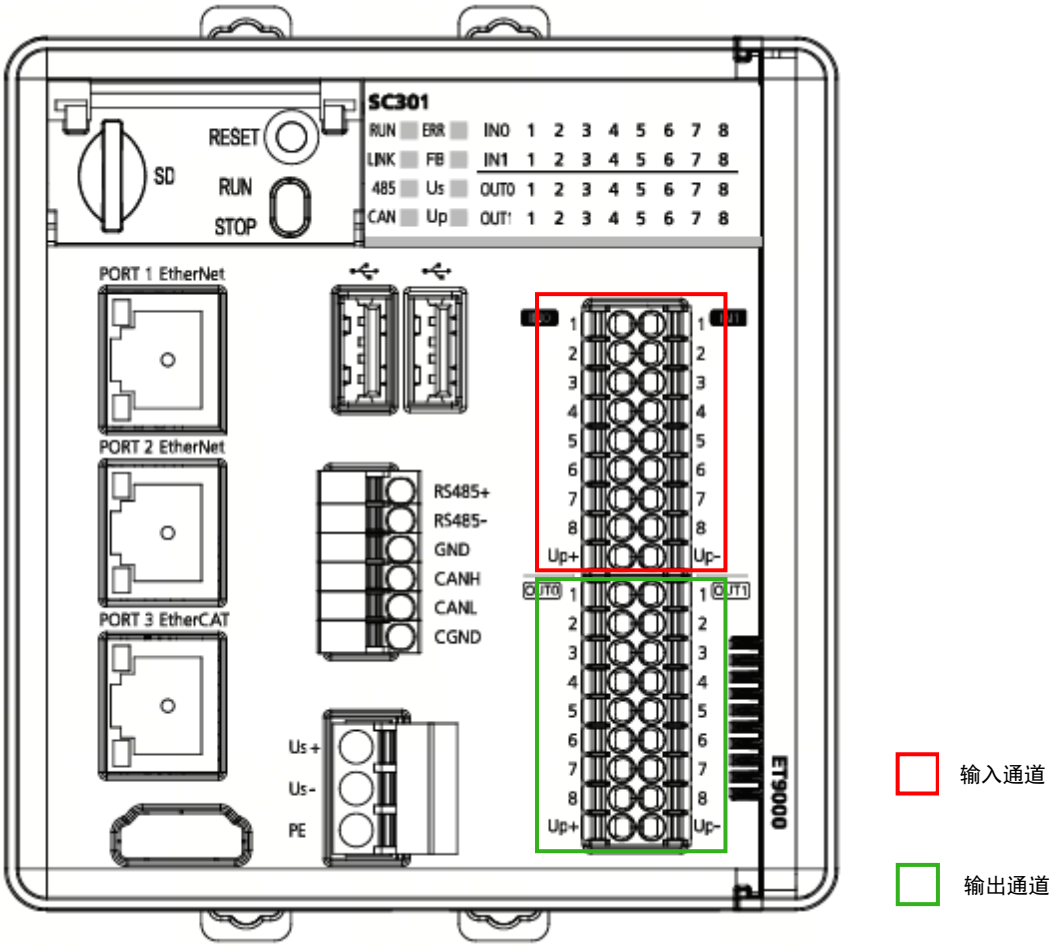


图 8-1 SC301 本体 I/O 输入/输出分布说明

8.1 下载和安装 SC301FixedIO 设备描述文件

在使用 SC301FixedIO 之前需要先在 CODESYS 软件上安装其对应的设备描述文件，设备描述文件可在高创传动官网 www.servotronix.cn 首页，依次单击“资料下载中心”->“运动控制器&I/O 模块”->“紧凑型运动控制器”->“SC301”->“设备描述文件”，在弹出的页面进行选择下载，并保存在合适位置。

如下图 8-2，SC301FixedIO 设备描述文件名称为：Servotronix SC301 Fixed IO Vx.x.x.ZIP，压缩包内包含设备描述文件和图标文件，其中设备描述文件格式为.XML，图标文件格式为.icon。后续版本可能会有更新，请注意版本变化。

图 8-2 SC301FixedIO 设备描述文件

将压缩包解压后（要确保 XML 文件和 icon 文件在同一个文件夹下），参考“**6.1 下载和安装 SC301 的设备描述文件**”章节安装 XML 文件。

XML 文件安装成功后，打开设备存储库页面，供应商选择“Servotronix Motion Control Ltd.”，点击“其他项”处的“+”展开即可看到，如下图 8-3 所示。

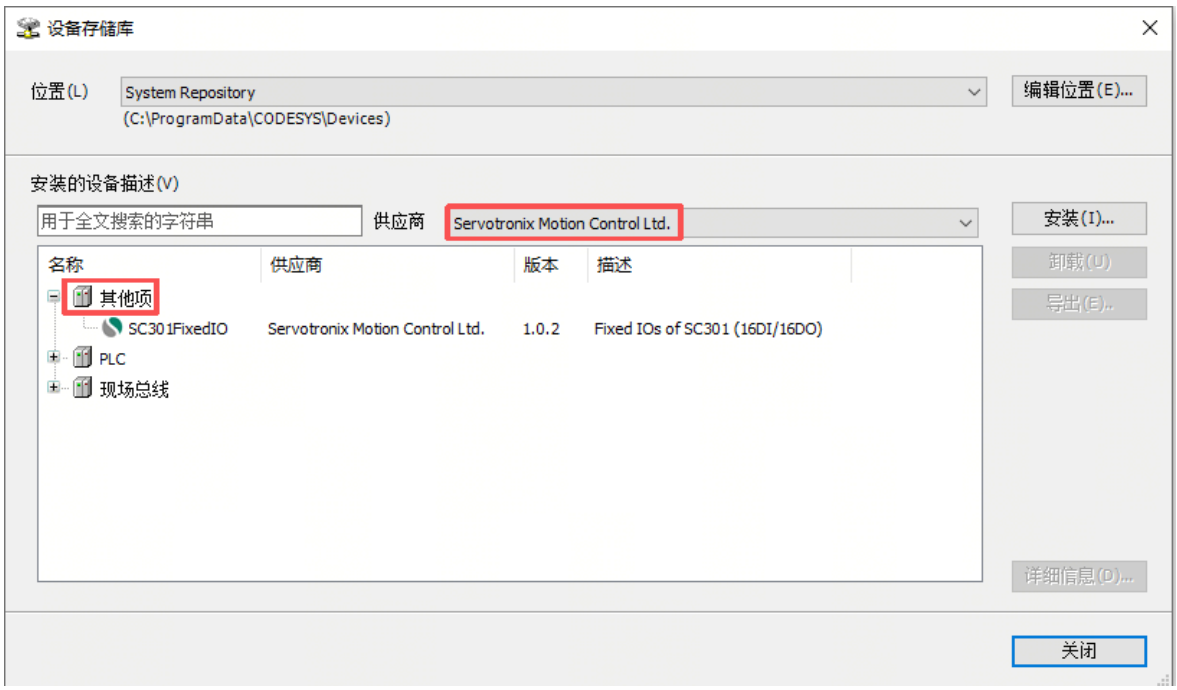


图 8-3 SC301FixedIO 在设备存储库中的位置

8.2 添加 SC301FixedIO 到工程

SC301FixedIO 设备描述文件安装完成以后，接下来将 SC301FixedIO 设备添加到工程中。

如下图 8-4，右键单击选择设备树中的 Device（Servotronix SC301 ARM SMWV dynamic Vx.x.x），在弹出的页面中选择“添加设备…”。

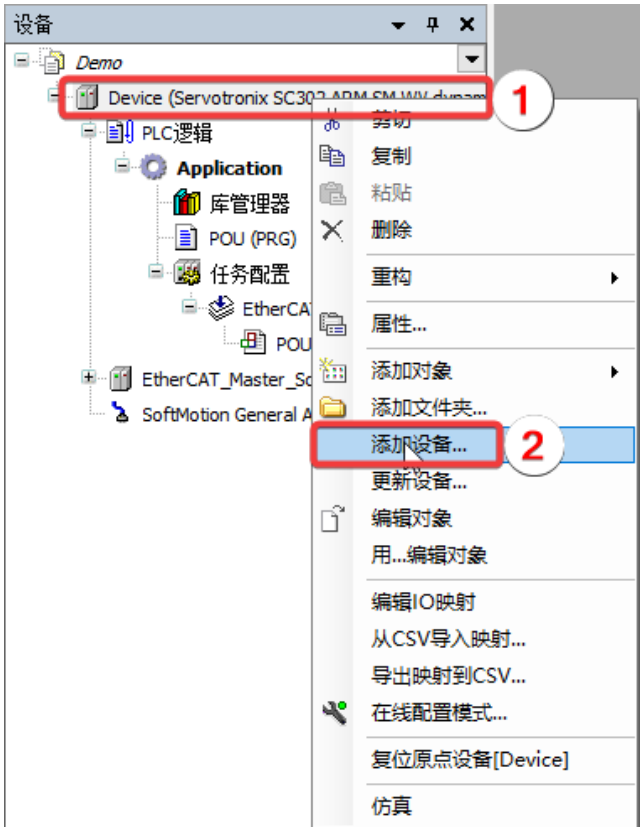


图 8-4 右键“Device”添加设备

如下图 8-5，在弹出的“添加设备”页面中选择 SC301FixedIO，最后点击“添加设备”按钮完成添加，然后关闭窗口。

图 8-5 添加 SC301FixedIO 设备

SC301FixedIO 设备添加完成后会出现在左侧的工程窗口，如下图 8-6 所示，分为 SC301_IN_16 和 SC301_OUT_16 两组。

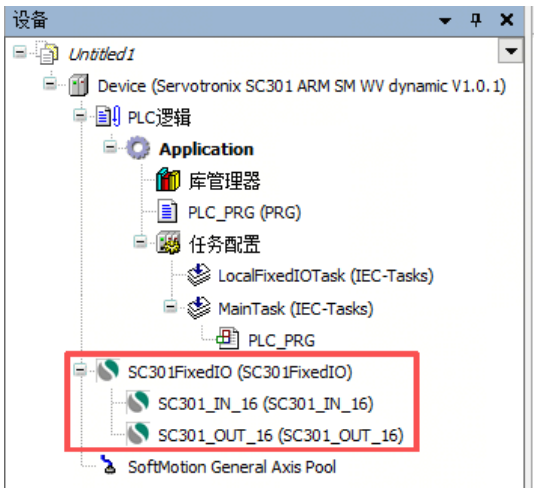


图 8-6 SC301FixedIO 在工程窗口中的位置

8.3 SC301FixedIO 过程数据映射

SC301FixedIO 过程数据的映射是为了将物理输入和输出信号与应用程序中的变量进行逐个映射匹配，从而能够实现编程访问。

CODESYS 会自动为 SC301FixedIO 的所有输入和输出通道物理信号分配地址。在 CODESYS 软件界面左侧工程窗口，双击已经添加到工程中的 SC301_IN_16，在弹出的窗口中单击“DI I/O 映射”选项卡，即可查看 16 个输入通道分配的物理地址和数据类型，如下图 8-7；双击 SC301_OUT_16，在弹出的窗口中单击“DO I/O 映射”选项卡，即可查看 16 个输出通道分配的物理地址和数据类型，如下图 8-8。输入/输出模块的通道物理信号，可以按照 Byte 访问，也可以按照 Bit 访问。

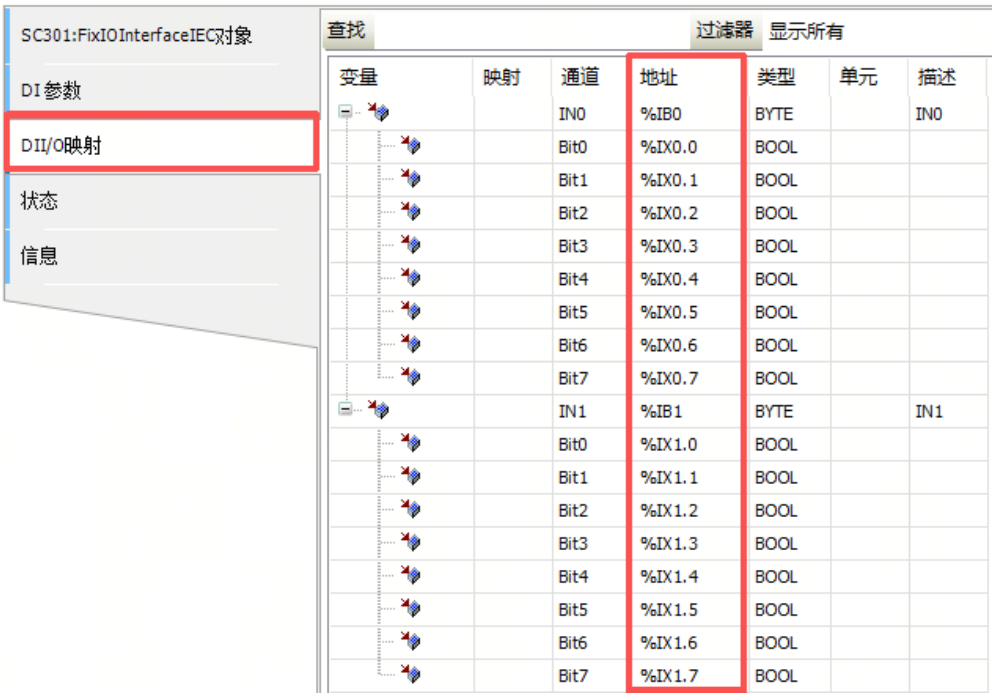


图 8-7 通过 DI I/O 映射查看地址和数据类型

图 8-8 通过 DO I/O 映射查看地址和数据类型

只有将这些输入/输出通道物理信号的地址与变量建立映射关系才可以进行访问，映射的方法主要有两种，任何一种皆可。

8.3.1 先定义变量，再与地址进行映射

【注意】：这是最常用最推荐的标准做法，特别适合将工程变量与物理输入/输出通道进行映射。映射时，需要注意数据类型匹配，否则会发生编译报错，且映射到输入端的变量是只读的，不可在程序中被赋值。同时，一个物理输入不能重复映射到两个不同变量。

如下图 8-9，首先在程序内部定义变量，变量的类型需要与“DII/O 映射”或“DOI/O 映射”表中的变量类型保持一致。

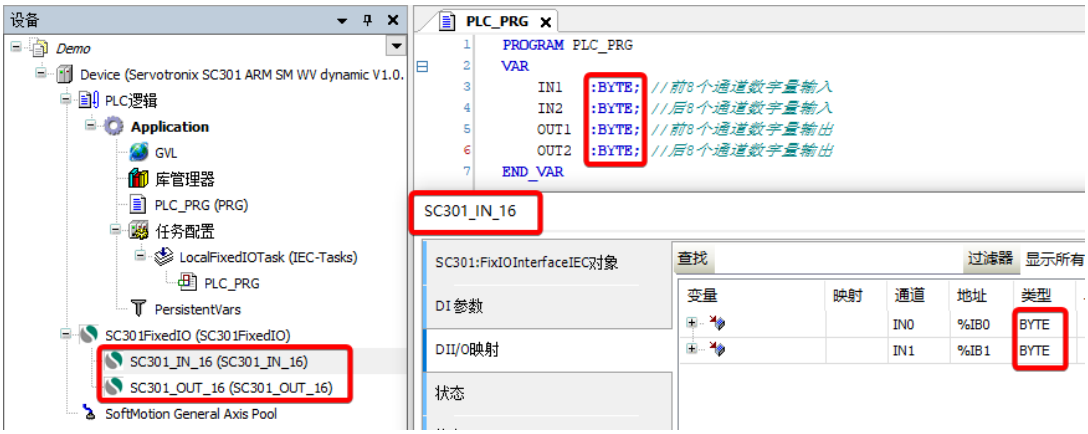


图 8-9 变量定义

如下图 8-10，然后点击需要映射的“DII/O 映射”或者“DOI/O 映射”选项卡，然后点击变量栏“…”，之后在弹出窗口中选择对应的变量，并点击确认关闭弹出的窗口。

之后就可以在程序中进行正常访问了。

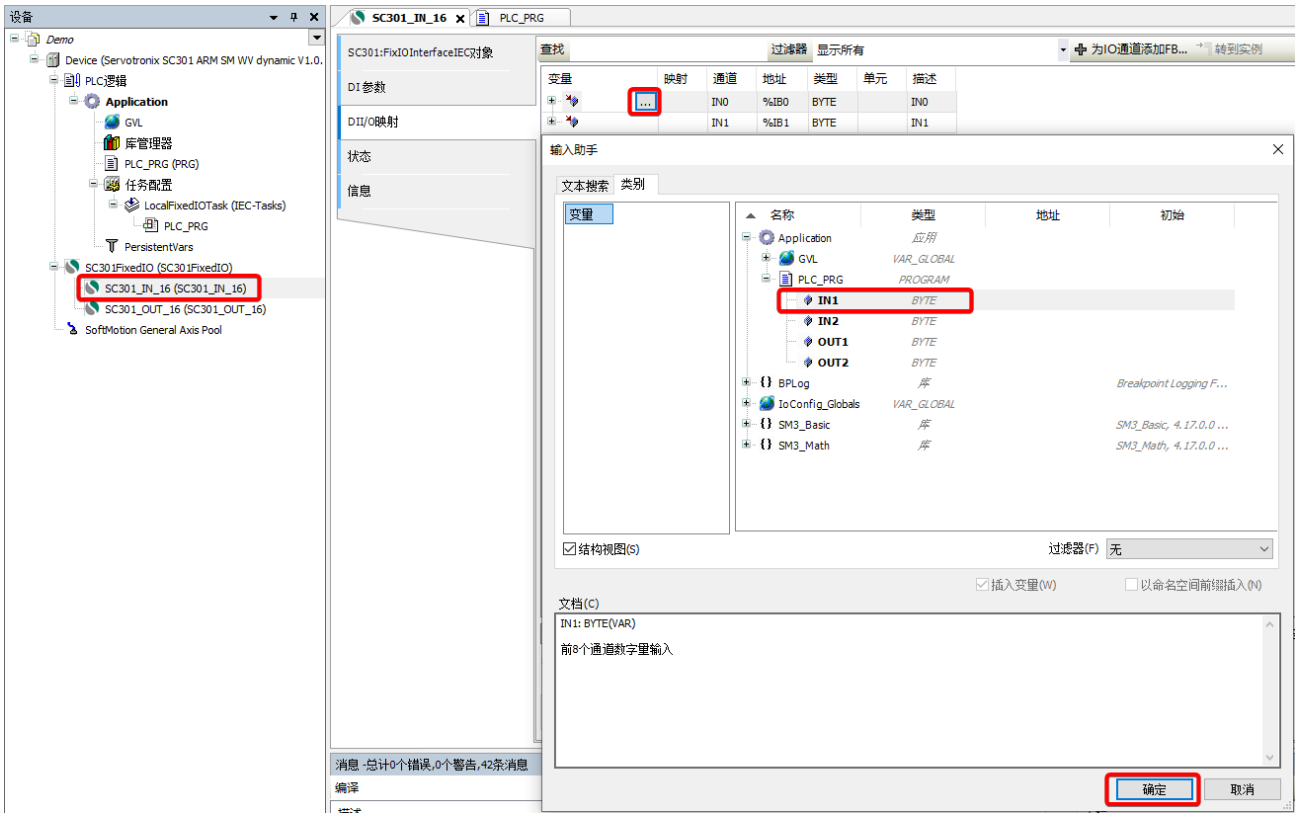


图 8-10 变量映射

8.3.2 变量定义的同时直接使用 AT 关键字与地址映射

首先在 SC301FixedIO 的“DII/O 映射”或者“DOI/O 映射”选项卡中查看各 I/O 通道分配的变量地址，然后在定义变量时，通过 AT 关键字，将变量定义时直接与物理地址进行映射，如下图 8-11。

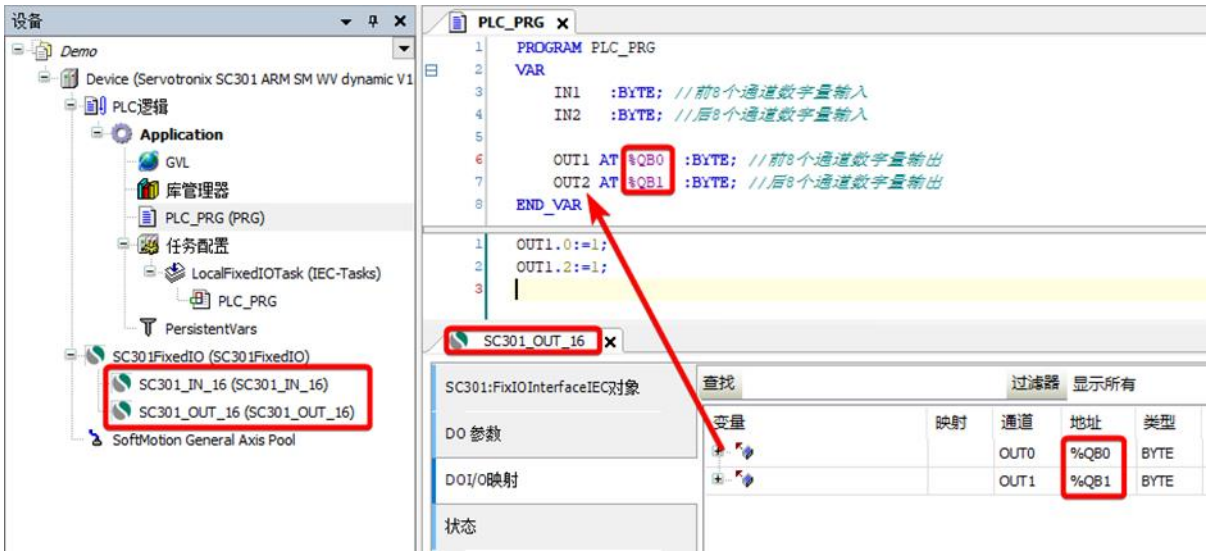


图 8-11 变量定义时直接映射地址

【注意】：

（1）本章节使用 AT 关键字可以定义局部变量或全局变量，但不能在功能块的输入/输出变量上使用。先定义变量再添加对应地址直接使用，与前一个章节的先定义变量，再与地址进行映射的方式相比，**本章节介绍的这种方式不够灵活，一旦 I/O 模块增加、减少或者 I/O 模块在工程中组态的顺序发生了变化，会导致输入和输出的物理通道地址发生变化，此时必须将变量与新分配的物理地址再次进行逐个映射才可以，而且非常容易疏忽造成地址冲突，所以不推荐这种方式。**

（2）同时需要注意地址语法。它必须以百分号 % 开头，包含 I（输入）、Q（输出）存储区前缀，并通过可选的大小前缀 X（位）、B（字节）、W（字）、D（双字）来指定数据类型，**需要确保定义变量的地址格式与映射地址一致。**

9. 扩展 I/O 模块

在一些复杂的应用场景，SC301 面临本体上的 I/O 点数不足的问题，此时可以通过添加扩展 I/O 模块实现。

SC301 扩展 I/O 模块的方式有 4 种，可以根据需要进行选择。

(1)选择高创传动 ET 系列扩展 I/O 系统进行扩展：直接安装于控制器右侧进行**本地集中扩展**。

(2)选择高创传动 ET 系列扩展 I/O 系统进行扩展：通过 EtherCAT 耦合器**远程集中扩展**。

(3)选择高创传动 TL 系列远程 I/O 系统进行扩展：通过 EtherCAT 耦合器**远程集中扩展**。

(4)选择第三方的远程 I/O 系统进行扩展：通过 EtherCAT 耦合器**远程集中扩展**。

ET 系列和 TL 系列 I/O 系统的选型请分别查阅附录 1 和附录 2，具体的使用方法请分别参考《ET 系列扩展 IO 系统用户手册》和《TL 系列远程 IO 系统用户手册》。

《ET 系列扩展 IO 系统用户手册》可在高创传动官网 www.servotronix.cn 首页，依次单击“资料下载中心”->“运动控制器&I/O 模块” ->“I/O 模块”->“ET 系列”->“用户手册”进行下载。

《TL 系列扩展 IO 系统用户手册》可在高创传动官网 www.servotronix.cn 首页，依次单击“资料下载中心”->“运动控制器&I/O 模块” ->“I/O 模块”->“TL 系列”->“用户手册”进行下载。

10. Modbus TCP/RTU 及自由通信

SC301 支持 Modbus TCP 主、从协议及 TCP/UDP 自由协议，Modbus RTU 主、从协议及串口自由协议，CANopen 协议、OPC-UA 协议，支持 PLCHandler 通信。其中 CANopen 协议、OPC-UA 协议和 PLCHandler 均是 CODESYS 系统内置的。Modbus TCP 主、从协议及 TCP/UDP 自由协议，Modbus RTU 主、从协议及串口自由协议均是由高创自主开发，并以编译库形式提供。

同时，在高创传动官网 www.servotronix.cn 首页，“资料下载中心”->“运动控制器&I/O 模块”->“紧凑型运动控制器” ->“SC301”->“通信示例工程”，还提供了通信示例工程供下载参考使用。

故本章将重点介绍如何使用编译库来进行 Modbus TCP 主、从协议及 TCP/UDP 自由协议，Modbus RTU 主、从协议及串口自由协议通信。

10.1 Modbus TCP 通信

在进行 Modbus TCP 通信协议之前，需要先安装高创传动提供的 Modbus TCP 编译库，TCP/UDP 自由协议通信无需安装。

10.1.1 下载和安装 Modbus TCP 编译库

Modbus TCP 编译库文件可在高创官网 www.servotronix.cn 首页，依次单击“资料下载中心”->“运动控制器&I/O 模块”->“紧凑型运动控制器”->“SC301”->“编译库”，在弹出的页面选择下载，并保存在计算机合适位置。

SC301 的 Modbus TCP 编译库文件，包括 Master 和 Slave 两个文件，其格式均为.complied-library。后续版本可能会有更新，请留意。

在使用 Modbus TCP 编译库接口进行编程之前，需要先进行编译库的安装，并添加到当前工程中，下面说明操作步骤。

如下图 10-1，点击菜单栏“工具”，在弹出的下拉列表中选择“库存储…”。

图 10-1 点击“工具”然后选择“库存储…”

如下图 10-2，在弹出的“库存储”页面先点击“安装…”按钮，然后在弹出的“选择库”对话窗口中找到存放 Modbus TCP Master 编译库和 Modbus TCP Slave 编译库文件的位置，右下角文件类型选择“编译的库文件”，之后选中要安装的库文件，点击右下角“打开（O）”按钮，即可安装成功。

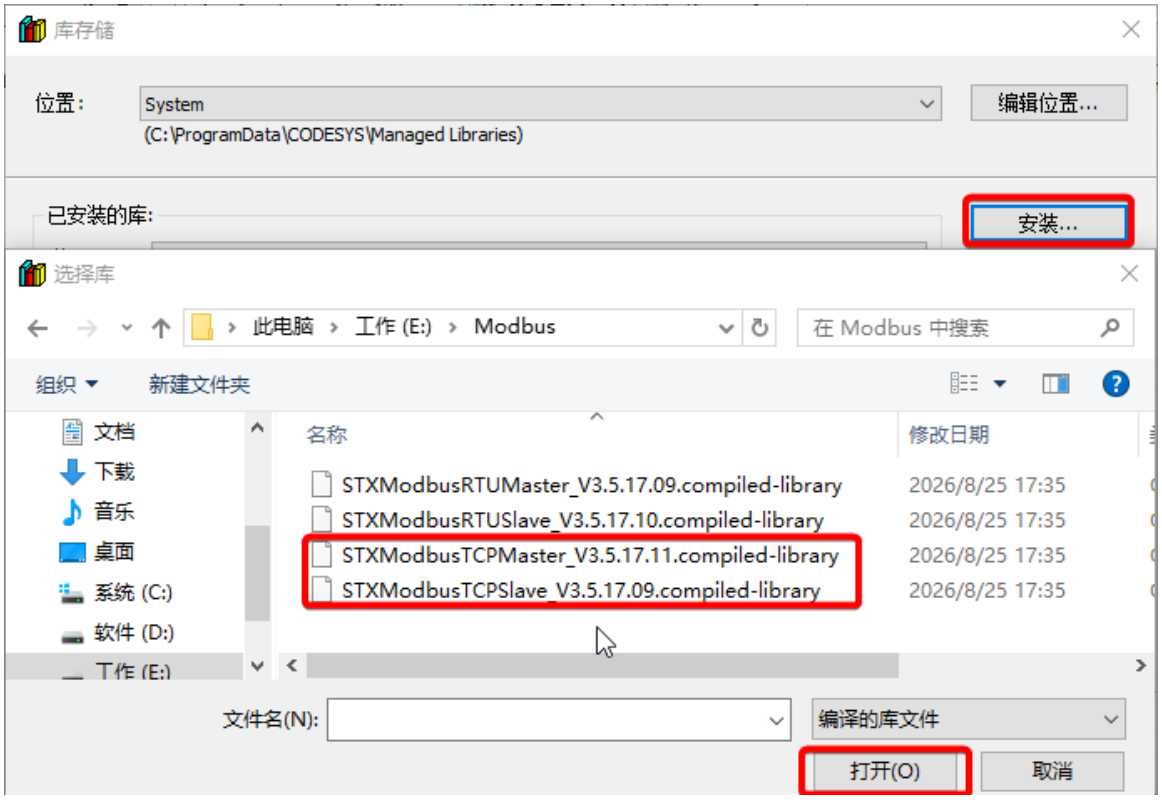


图 10-2 选择并安装 Modbus TCP 编译库

编译库安装成功后的位置如图 10-3 所示，然后点击“关闭”按钮。

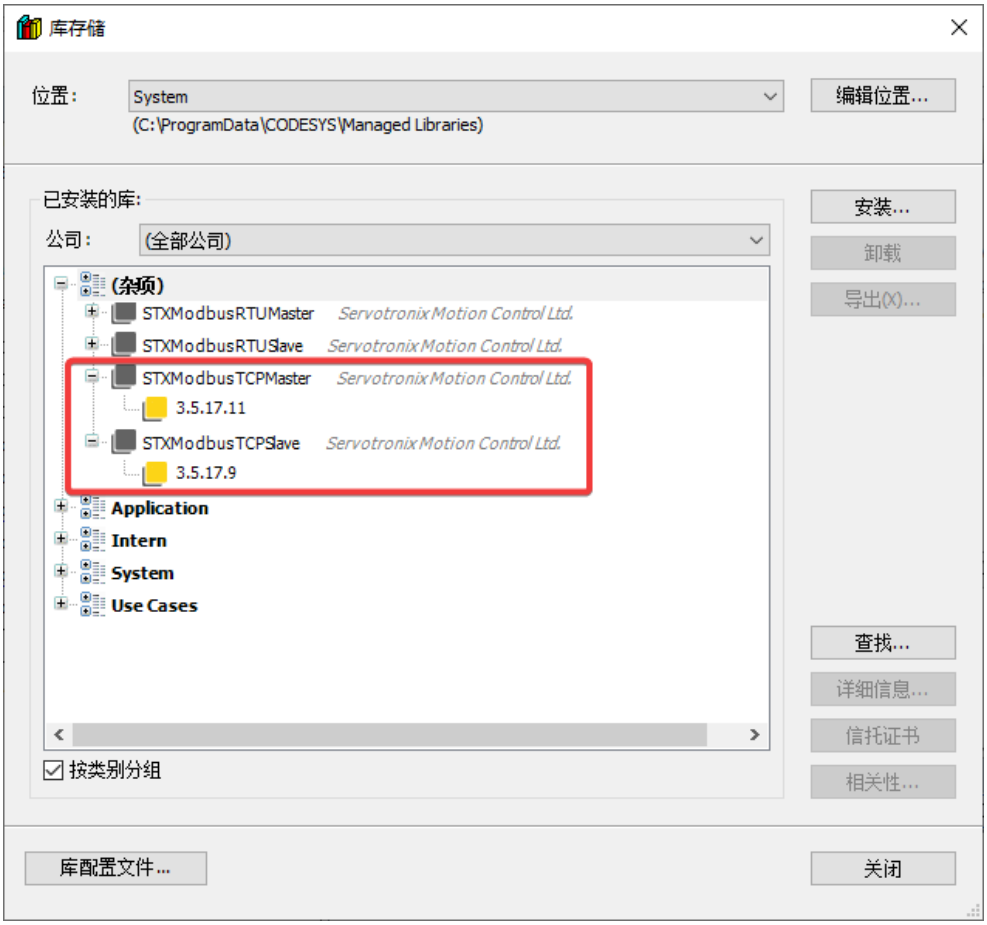


图 10-3 安装成功后 Modbus TCP 编译库文件所在位置

如下图 10-4，双击工程窗口的“库管理器”，会出现“库管理器”配置页面。

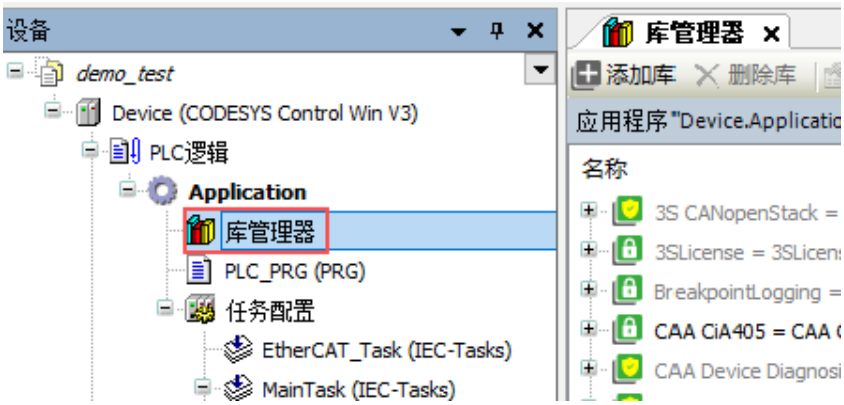


图 10-4 双击“库管理器”

如下图 10-5，点击“库管理器”页面左上角的“+添加库”，然后在弹出的“添加库”页面中点击“+（杂项）”展开，选中要添加到工程中的已经安装的库文件（可以同时选多个），最后点击“确定”，即添加成功。

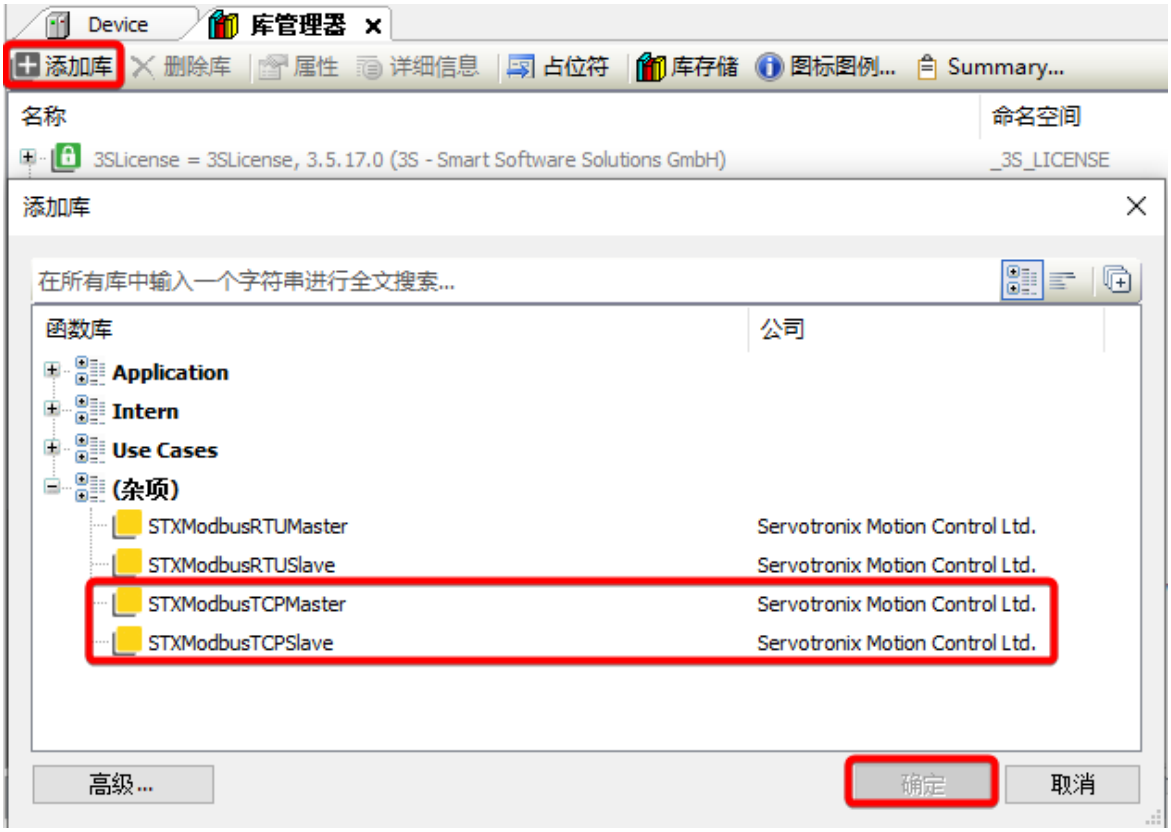


图 10-5 添加 Modbus TCP 编译库到当前工程

添加成功后，可以在下图 10-6 看到，Modbus TCP 编译库已经出现在应用程序当前使用的库列表中了。

之后就可以按照使用说明，进行正常编程和使用了。

图 10-6 Modbus TCP 编译库添加到工程后的效果呈现

10.1.2 作为 Modbus TCP Mater(主站)

通过编译库提供的编程接口来实现 Modbus TCP Master(主站)功能，针对 ModbusTCP 协议，功能块 ModbusTCPMaster 提供了 8 种 Method(方法)进行处理，如下图 10-7 所示。

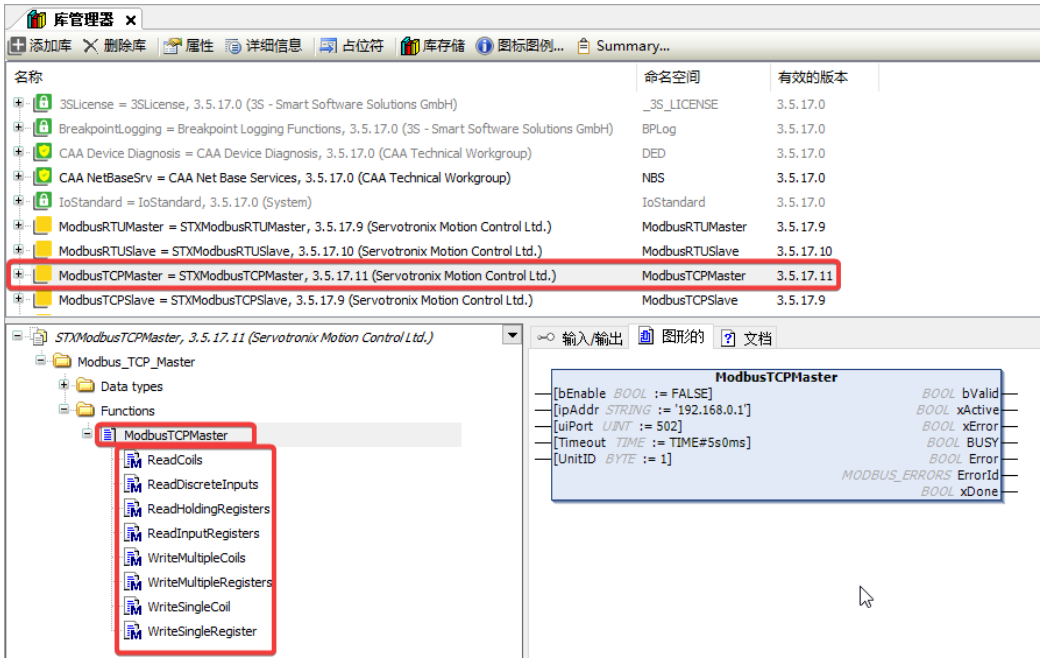


图 10-7 ModbusTCPMaster 编译库接口

下表 10-1 给出了 ModbusTCPMaster 提供的所有功能块及 Method(类似于函数)列表，下面分别说明。

表 10-1 ModbusTCPMaster 功能块列表

名称	类型	功能码	参数说明
ModbusTCPMaster	Function Block	--	ModbusTCP 主站功能块 （命名空间为 ModbusTCPMaster）
ReadCoils	Method	0x01	读线圈
ReadDiscreteInputs	Method	0x02	读离散输入
ReadHoldingRegisters	Method	0x03	读保持寄存器
ReadInputRegisters	Method	0x04	读输入寄存器
WriteMultipleCoils	Method	0x0F	写多个线圈
WriteMultipleRegisters	Method	0x10	写多个保持寄存器
WriteSingleCoil	Method	0x05	写单个线圈
WriteSingleRegister	Method	0x06	写单个保持寄存器

(1) ModbusTCPMaster

ModbusTCPMaster：ModbusTCPMaster 功能块。

ModbusTCPMaster 功能块指令图见下图 10-8。

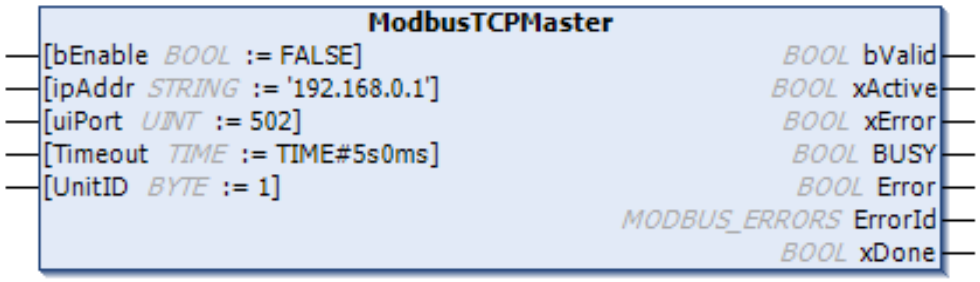


图 10-8 ModbusTCPMaster 指令图

ModbusTCPMaster 功能块输入/输出接口参数见下表 10-2。

表 10-2 ModbusTCPMaster 参数说明

参数名称	数据类型	IN/OUT	参数说明	取值范围	默认值
bEnable	BOOL	IN	功能块使能，使用中保持 TRUE	TRUE,FALSE	FALSE
ipAddr	STRING	IN	从站 IP 地址	取决于从站	192.168.0.1
uiPort	UINT	IN	从站端口号	取决于从站	502
Timeout	TIME	IN	连接、响应超时时间	遵循数据类型	T#5S
UnitID	BYTE	IN	从站站点号	1～255	1
bValid	BOOL	OUT	是否可运行在当前硬件平台上	TRUE,FALSE	FALSE
xActive	BOOL	OUT	主从站之间的 TCP 连接状态	TRUE,FALSE	FALSE
xError	BOOL	OUT	主从站之间 TCP 连接出错	TRUE,FALSE	FALSE
BUSY	BOOL	OUT	通信进行中	TRUE,FALSE	FALSE
Error	BOOL	OUT	Modbus 消息错误	TRUE,FALSE	FALSE
ErrorId	MODBUS_ERRORS 枚举	OUT	Modbus 消息错误 ID	参考定义	MODBUS_ERROR_NO_ERROR
xDone	BOOL	OUT	Modbus 消息交互成功	TRUE,FALSE	FALSE

当通信出现错误时，ERROR 标志位会置为 TRUE，并通过 ErrorId 给出具体的错误码，MODBUSTCPMaster 功能块中 MODBUS_ERRORS 的定义及含义见下表 10-3，根据该信息，可以协助进行问题排查。

表 10-3 MODBUSTCPMaster 功能块 MODBUS_ERRORS 枚举定义

名称	值	含义
MODBUSERROR_NO_ERROR	0	正常，无错误
MODBUSERROR_ILLEGAL_FUNCTION	1	非法功能码
MODBUSERROR_ILLEGAL_DATA_ADDRESS	2	寄存器地址错误
MODBUSERROR_ILLEGAL_DATA_VALUE	3	非法数据
MODBUSERROR_SLAVE_DEVICE_FAILURE	4	从设备故障，无法响应

名称	值	含义
MODBUSERROR_ACKNOWLEDGE	5	从设备已收到请求，需较长处理时间，主设备需要发送查询以确认完成
MODBUSERROR_SLAVE_DEVICE_BMSY	6	从设备忙
MODBUSERROR_NEGATIVE_ACKNOWLEDGE	7	从设备无法完成请求，可能因为某些条件未满足
MODBUSERROR_MEMORY_PARITY	8	从设备检测到存储器奇偶校验错误
MODBUSERROR_reserved9	9	保留
MODBUSERROR_GATEWAY_PATH_UNAVAILABLE	10	网关路径不可用
MODBUSERROR_GATEWAY_TARGET_DEVICE_FAILED_TO_RESPOND	11	网关目标设备失败
MODBUSERROR_CHARREC_TIMEOUT	20	接收等待超时
MODBUSERROR_ILLEGAL_DATA_SIZE	21	数据长度错误
MODBUSERROR_ILLEGAL_DEVICE_ADDRESS	22	非法设备地址
MODBUSERROR_ILLEGAL_DESTINATION_ADDRESS	23	非法目标地址
MODBUSERROR_ILLEGAL_DESTINATION_SIZE	24	非法目标长度
MODBUSERROR_NO_RESPONSE	25	无响应
MODBUSERROR_TXBUFFOVERRUN	102	发送缓存溢出
MODBUSERROR_SENDTIMEOUT	103	发送超时
MODBUSERROR_DATASIZEOVERRUN	107	数据长度溢出
MODBUSERROR_STRINGOVERRUN	110	字符串溢出
MODBUSERROR_INVALIDPOINTER	120	无效指针 NULL
MODBUSERROR_CRC	150	CRC 错误
MODBUSERROR_INVALIDMEMORYADDRESS	232	无效内存地址
MODBUSERROR_TRANSMITBUFFERTOOSMALL	233	发送缓存太小

(2) ReadCoils

ReadCoils：读线圈，其指令图及输入输出参数分别见图 10-9 和表 10-4。

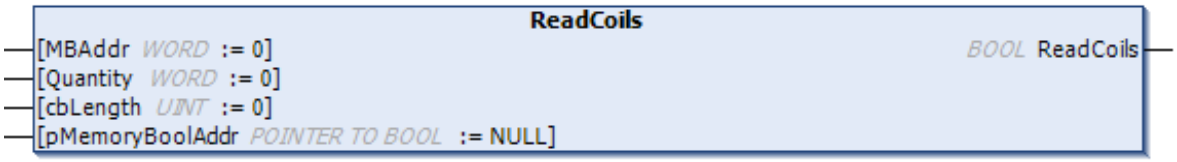


图 10-9 ReadCoils 指令图

表 10-4 ReadCoils 参数说明

参数名称	数据类型	IN/OUT	参数说明	取值范围	默认值
MBAddr	WORD	IN	从站 Modbus 寄存器起始地址	取决于从站	0
Quantity	WORD	IN	读线圈数量	1～2008	0
cbLength	UINT	IN	主站接收读取数据 buffer 大小（字节），需满足 cbLength >= Quantity	不小于读取数据大小	0
pMemoryBoolAddr	POINTER TO BOOL	IN	主站接收读取数据 buffer 的地址	非 NULL	NULL
ReadCoils	BOOL	OUT	不使用，无需关注	-	-

(3) ReadDiscreteInputs

ReadDiscreteInputs：读离散输入，其指令图和输入输出参数分别见图 10-10 和表 10-5。

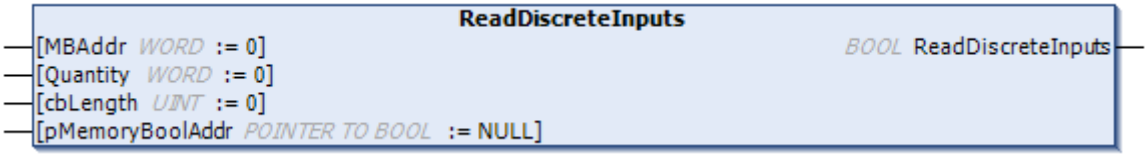


图 10-10 ReadDiscreteInputs 指令图

表 10-5 ReadDiscreteInputs 参数说明

参数名称	数据类型	IN/OUT	参数说明	取值范围	默认值
MBAddr	WORD	IN	从站 Modbus 寄存器起始地址	取决于从站	0
Quantity	WORD	IN	读离散输入数量	1～2008	0

参数名称	数据类型	IN/OUT	参数说明	取值范围	默认值
cbLength	UINT	IN	主站接收读取数据 buffer 大小（字节），需满足 cbLength >= Quantity	不小于读取数据大小	0
pMemoryBoolAddr	POINTER TO BOOL	IN	主站接收读取数据 buffer 的地址	非 NULL	NULL
ReadDiscreteInputs	BOOL	OUT	不使用，无需关注	-	-

(4) ReadHoldingRegisters

ReadHoldingRegisters：读保持寄存器，其指令图和输入输出参数分别见图 10-11 和表 10-6。

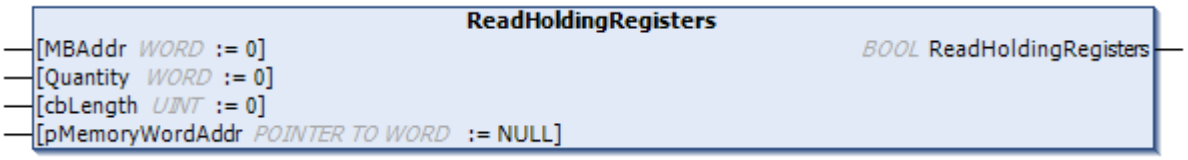


图 10-11 ReadHoldingRegisters 指令图

表 10-6 ReadHoldingRegisters 参数说明

参数名称	数据类型	IN/OUT	参数说明	取值范围	默认值
MBAddr	WORD	IN	从站 Modbus 寄存器起始地址	取决于从站	0
Quantity	WORD	IN	读保持寄存器数量	1～125	0
cbLength	UINT	IN	主站接收读取数据 buffer 大小（字节），需满足 cbLength >= 2*Quantity	不小于读取数据大小	0
pMemoryWordAddr	POINTER TO WORD	IN	主站接收读取数据 buffer 的地址	非 NULL	NULL
ReadHoldingRegisters	BOOL	OUT	不使用，无需关注	-	-

(5) ReadInputRegisters

ReadInputRegisters：读输入寄存器，其指令图和输入输出参数分别见图 10-12 和表 10-7。

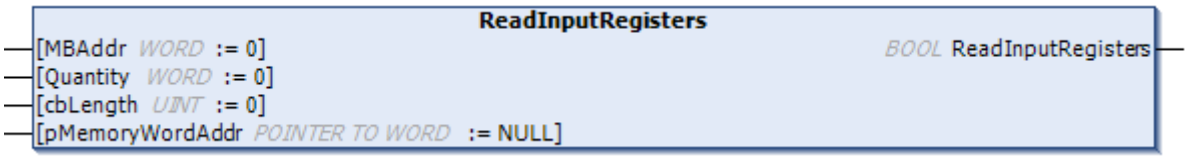


图 10-12 ReadInputRegisters 指令图

表 10-7 ReadInputRegisters 参数说明

参数名称	数据类型	IN/OUT	参数说明	取值范围	默认值
MBAddr	WORD	IN	从站 Modbus 寄存器起始地址	取决于从站	0
Quantity	WORD	IN	读输入寄存器数量	1～125	0
cbLength	UINT	IN	主站接收读取数据 buffer 大小（字节），需满足 cbLength >= 2*Quantity	不小于读取数据大小	0
pMemoryWordAddr	POINTER TO WORD	IN	主站接收读取数据 buffer 的地址	非 NULL	NULL
ReadInputRegisters	BOOL	OUT	不使用，无需关注	-	-

(6) WriteMultipleCoils

WriteMultipleCoils：写多个线圈，其指令图和输入输出参数分别见图 10-13 和表 10-8。

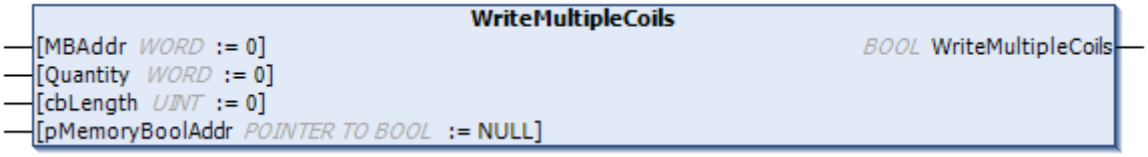


图 10-13 WriteMultipleCoils 指令图

表 10-8 WriteMultipleCoils 参数说明

参数名称	数据类型	IN/OUT	参数说明	取值范围	默认值
MBAddr	WORD	IN	从站 Modbus 寄存器起始地址	取决于从站	0
Quantity	WORD	IN	写多个线圈数量	1～1976	0
cbLength	UINT	IN	主站发送数据 buffer 大小（字节），需满足 cbLength >= Quantity	不小于读取数据大小	0
pMemoryBoolAddr	POINTER TO BOOL	IN	主站发送数据 buffer 的地址	非 NULL	NULL
WriteMultipleCoils	BOOL	OUT	不使用，无需关注	-	-

(7) WriteMultipleRegisters

WriteMultipleRegisters：写多个保持寄存器，其指令图和输入输出参数分别见图 10-14 和表 10-9。

图 10-14 WriteMultipleRegisters 指令图

表 10-9 WriteMultipleRegisters 参数说明

参数名称	数据类型	IN/OUT	参数说明	取值范围	默认值
MBAddr	WORD	IN	从站 Modbus 寄存器起始地址	取决于从站	0
Quantity	WORD	IN	写多个保持寄存器数量	1～123	0
cbLength	UINT	IN	主站发送数据 buffer 大小（字节）需满足 cbLength >= 2*Quantity	不小于读取数据大小	0
pMemoryWordAddr	POINTER TO WORD	IN	主站发送数据 buffer 的地址	非 NULL	NULL
WriteMultipleRegisters	BOOL	OUT	不使用，无需关注	-	-

(8) WriteSingleCoil

WriteSingleCoil：写单个线圈，其指令图和输入输出参数分别见图 10-15 和表 10-10。

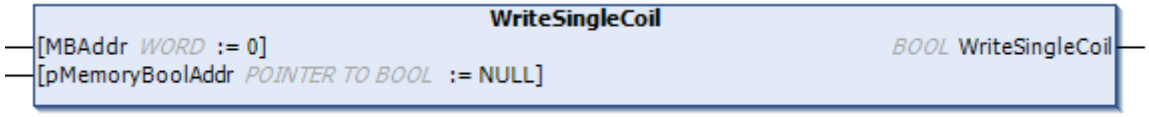


图 10-15 WriteSingleCoil 指令图

表 10-10 WriteSingleCoil 参数说明

参数名称	数据类型	IN/OUT	参数说明	取值范围	默认值
MBAddr	WORD	IN	从站 Modbus 寄存器起始地址	取决于从站	0
pMemoryBoolAddr	POINTER TO BOOL	IN	主站发送数据 buffer 的地址	非 NULL	NULL

参数名称	数据类型	IN/OUT	参数说明	取值范围	默认值
WriteSingleCoil	BOOL	OUT	不使用，无需关注	-	-

(9) WriteSingleRegister

WriteSingleRegister：写单个保持寄存器，其指令图和输入输出参数分别见图 10-16 和表 10-11。

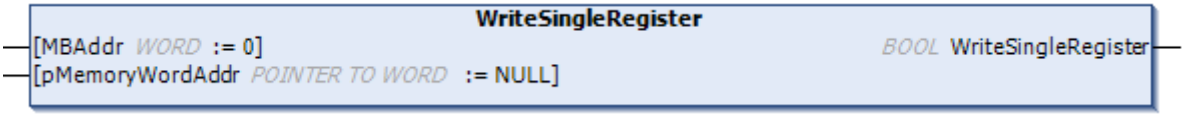


图 10-16 WriteSingleRegister 指令图

表 10-11 WriteSingleRegister 参数说明

参数名称	数据类型	IN/OUT	参数说明	取值范围	默认值
MBAddr	WORD	IN	从站 Modbus 寄存器起始地址	取决于从站	0
pMemoryWordAddr	POINTER TO WORD	IN	主站发送数据 buffer 的地址	非 NULL	NULL
WriteSingleRegister	BOOL	OUT	不使用，无需关注	-	-

图 10-17、图 10-18 提供了 ModbusTCPMaster 通信示例程序。

图 10-17 列出了该功能所需的基本变量定义，其中线圈、离散输入、保持寄存器、输入寄存器的数据 buffer 大小、定义的位置可根据实际需求进行调整。

图 10-18 的功能实现部分，首先使能 ModbusTCPMaster，当 TCP 连接成功后，根据状态机依次轮询执行 8 个 Modbus 功能函数。在实际应用中，根据实际情况选择需要的功能函数，但要避免在当前功能函数执行过程中（即 BUSY 为 TRUE 时），修改其输入参数、或者切换至其它功能函数。

当前功能函数执行完成（xDone 为 TRUE）、或者出错（Error 为 TRUE）后，可以切换至下一功能函数，且在该功能函数调用之前，根据需要变更其输入参数。

modbusTCP_Master_Demo

```
PROGRAM modbusTCP_Master_Demo
VAR
    mdbusTCPMaster_instance : ModbusTCPMaster.ModbusTCPMaster;      // 定义功能块实例
    slaveIP      : STRING := '192.168.0.50';                        // slave ip address
    step : BYTE := 0;

    coilsBuf         : ARRAY[0..99] OF BOOL;    // master buffer to store coils data
    discreteInputBuf : ARRAY[0..99] OF BOOL;    // master buffer to store discrete inputs data
    holdRegisterBuf  : ARRAY[0..99] OF WORD;    // master buffer to store holding registers data
    inputRegisterBuf : ARRAY[0..99] OF WORD;    // master buffer to store input registers data
END_VAR
```

图 10-17 ModbusTCPMaster 示例程序：变量定义

```
// 使能 modbusTCPMaster
mdbusTCPMaster_instance(bEnable:=TRUE, // keep TRUE in use
                        ipAddr:='192.168.0.' := slaveIP:='192.168.0.',
                        uiPort:=502 := 502,
                        Timeout:=T#5s := T#5S, // Modbus master waiting response timeout time
                        UnitID:=1 := 1, // slave unit id
                        bValid => , // TRUE: can run on the hardware platform (SC301/SC302/MC804)
                        xActive => , // TRUE: tcp connection success
                        xError => , // TRUE: tcp connection error
                        BUSY => , // TRUE: in modbus communication
                        Error => , // TRUE: modbus message error
                        ErrorId => , //       modbus message error id
                        xDone => // TRUE: modbus read or write operation completed, keep 1 cycle
);

// 硬件检查通过，TCP连接成功后：启动 ModbusTCP 通信
IF mdbusTCPMaster_instance.bValid=TRUE AND mdbusTCPMaster_instance.xActive=TRUE THEN
    CASE step[0] OF
        //=============================================================================================================
        // 读写线圈操作 (read or write coils operation)
        //=============================================================================================================
        0:  // 功能码0x01：读线圈 (read coils)
            mdbusTCPMaster_instance.ReadCoils(MBAddr := 0,                                  // 从站线圈modbus首地址（此次读操作）
                                              Quantity := 5,
                                              cbLength := SIZEOF(coilsBuf),                 // 内存大小（字节）
                                              pMemoryBoolAddr := ADR(coilsBuf[0]=TRUE));          // 存放线圈数据内存首地址
        1:  // 功能码0x0F：写多个线圈 (write multiple coils)
            mdbusTCPMaster_instance.WriteMultipleCoils(MBAddr := 5,                         // 从站线圈modbus首地址（此次写操作）
                                                       Quantity := 5,
                                                       cbLength := SIZEOF(coilsBuf) - 5,    // 内存大小（字节）
                                                       pMemoryBoolAddr := ADR(coilsBuf[5]=FALSE));  // 获取线圈数据内存首地址
        2:  // 功能码0x05：写单个线圈 (write single coil)
            mdbusTCPMaster_instance.WriteSingleCoil(MBAddr := 10,                           // 从站线圈modbus首地址（此次写操作）
                                                    pMemoryBoolAddr := ADR(coilsBuf[10]=FALSE));  // 获取线圈数据内存首地址

        //=============================================================================================================
        // 读写保持寄存器操作 (read or write holding register operation)
        //=============================================================================================================
        3:  // 功能码0x03：读保持寄存器 (read holding registers)
            mdbusTCPMaster_instance.ReadHoldingRegisters(MBAddr := 0,                       // 从站保持寄存器modbus首地址（此次读操作）
                                                         Quantity := 5,
                                                         cbLength := SIZEOF(holdRegisterBuf),   // 内存大小（字节）
                                                         pMemoryWordAddr := ADR(holdRegisterBuf[0]=11));  // 存放保持寄存器数据内存首地址
        4:  // 功能码0x10：写多个保持寄存器 (write multiple registers)
            mdbusTCPMaster_instance.WriteMultipleRegisters(MBAddr := 5,                     // 从站保持寄存器modbus首地址（此次写操作）
                                                           Quantity := 5,
                                                           cbLength := SIZEOF(holdRegisterBuf) - 5*2,  // 内存大小（字节）
                                                           pMemoryWordAddr := ADR(holdRegisterBuf[5]=6666));// 获取保持寄存器值内存首地址
        5:  // 功能码0x06：写单个保持寄存器 (write single register)
            mdbusTCPMaster_instance.WriteSingleRegister(MBAddr := 10,                       // 从站保持寄存器modbus首地址（此次写操作）
                                                        pMemoryWordAddr := ADR(holdRegisterBuf[10]=4444));  // 获取保持寄存器值内存首地址

        //=============================================================================================================
        // 读离散输入操作 (read discrete input operation)
        //=============================================================================================================
        6:  // 功能码0x02：读离散输入 (read discrete inputs)
            mdbusTCPMaster_instance.ReadDiscreteInputs(MBAddr := 0,                         // 从站离散输入modbus首地址（此次读操作）
                                                       Quantity := 5,
                                                       cbLength := SIZEOF(discreteInputBuf),    // 内存大小（字节）
                                                       pMemoryBoolAddr := ADR(discreteInputBuf[0]=TRUE));   // 存储离散输入数据内存首地址

        //=============================================================================================================
        // 读输入寄存器操作 (read input registers operation)
        //=============================================================================================================
        7:  // 功能码0x04：读输入寄存器 (read input registers)
            mdbusTCPMaster_instance.ReadInputRegisters(MBAddr := 0,                         // 从站输入寄存器modbus首地址（此次读操作）
                                                       Quantity := 5,
                                                       cbLength := SIZEOF(inputRegisterBuf),    // 存放内存大小（字节）
                                                       pMemoryWordAddr := ADR(inputRegisterBuf[0]=111));   // 存放输入寄存器值内存的首地址
    END_CASE

    // switch to next function command
    IF mdbusTCPMaster_instance.xDone=FALSE OR mdbusTCPMaster_instance.Error=FALSE THEN
        step[0] := step[0] + 1;
        IF (step[0] > 7) THEN
            step[0] := 0;
        END_IF
    END_IF

ELSE
    step[0] := 0;
END_IFRETURN
```

图 10-18 ModbusTCPMaster 示例程序：功能实现

10.1.3 作为 Modbus TCP Slave(从站)

如下图 10-19 所示，使用编译库提供的编程接口来实现 ModbusTCPSlave(从站)功能。在功能块 ModbusTCPSlave 中，4 种（线圈、离散输入、保持寄存器、输入寄存器）变量的 Modbus 访问地址分配是相互独立的，且寄存器起始地址均从 0 开始，最大可访问地址由相应变量分配的存储空间大小来决定。

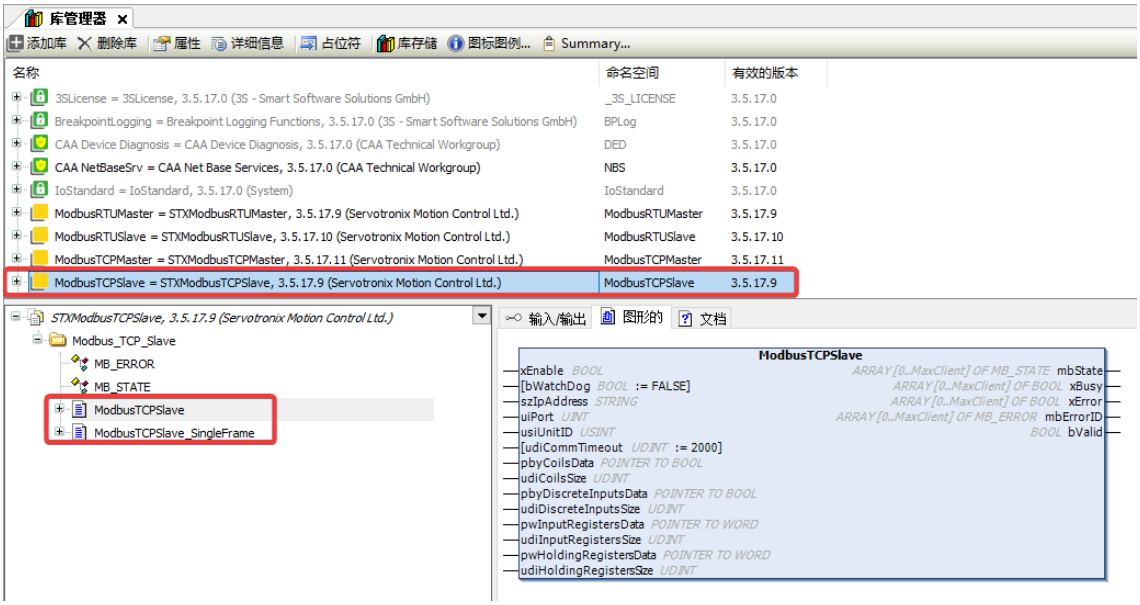


图 10-19 ModbusTCPSlave 编译库接口

表 10-12 ModbusTCPSlave 功能块列表

名称	类型	参数说明	注意
ModbusTCPSlave	Function Block	ModbusTCP 从站功能块，允许多连接，最大 21 个（命名空间为 ModbusTCPSlave）	**同一时刻二选一**
ModbusTCPSlave_SingleFrame	Function Block	ModbusTCP 从站功能块，单个连接（命名空间为 ModbusTCPSlave）	

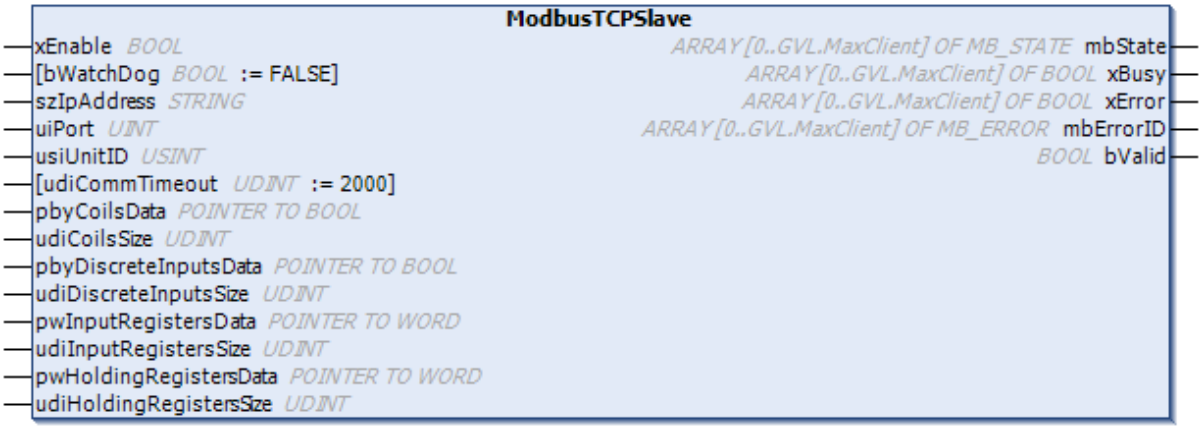


图 10-20 ModbusTCPSlave 指令图（多连接）

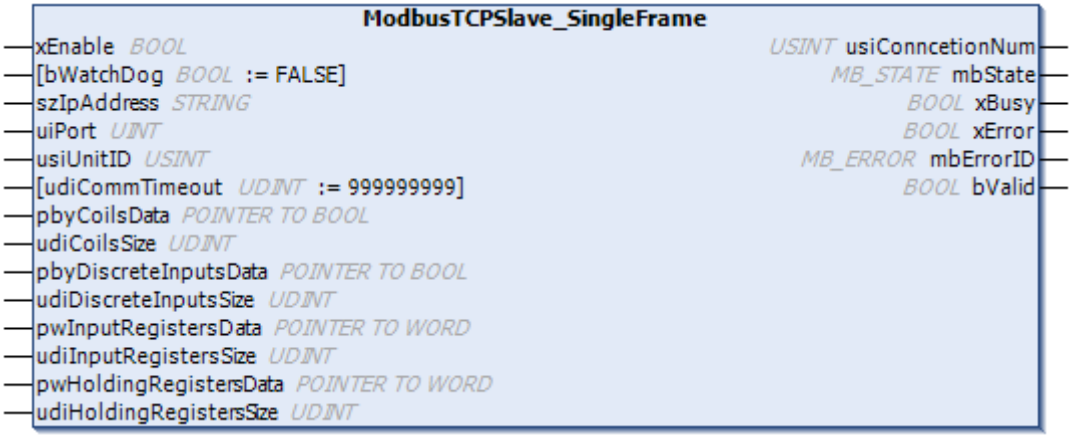


图 10-21 ModbusTCPSlave_SingleFrame 指令图（单连接）

表 10-13 ModbusTCPSlave 参数说明

参数名称	数据类型	IN/OUT	参数说明	取值范围	默认值
xEnable	BOOL	IN	功能块使能，使用中保持 TRUE	TURE, FALSE	FALSE
bWatchDog	BOOL	IN	保持 FALSE	TURE, FALSE	FALSE
szIpAddress	STRING	IN	从站（控制器）IP 地址	取决于从站	无
uiPort	UINT	IN	从站（控制器）端口号	取决于从站	0
usiUnitID	MSINT	IN	从站（控制器）站点号	1～255	0
udiCommTimeout	UDINT	IN	连接、响应超时时间，单位 ms	>0	2000

参数名称	数据类型	IN/OUT	参数说明	取值范围	默认值
pbyCoilsData	POINTER TO BOOL	IN	从站线圈数据区地址	非 NULL	NULL
udiCoilsSize	UDINT	IN	从站线圈数据区大小，单位 BOOL	不大于用户编程时定义的线圈区域大小	0
pbyDiscreteInputsData	POINTER TO BOOL	IN	从站离散输入数据区地址	非 NULL	NULL
udiDiscreteInputsSize	UDINT	IN	从站离散输入数据区大小，单位 BOOL	不大于用户编程时定义的离散输入区域大小	0
pwInputRegistersData	POINTER TO WORD	IN	从站输入寄存器数据区地址	非 NULL	NULL
udiInputRegistersSize	UDINT	IN	从站输入寄存器数据区大小，单位 WORD	不大于用户编程时定义的输入寄存器区域大小	0
pwHoldingRegistersData	POINTER TO WORD	IN	从站保持寄存器数据区地址	非 NULL	NULL
udiHoldingRegistersSize	UDINT	IN	从站保持寄存器数据区大小，单位 WORD	不大于用户编程时定义的保持寄存器区域大小	0
mbState	MB_STATE 数组	OUT	TCP 连接状态（最大 21 个连接）	参考定义	MB_IDLE
xBusy	BOOL 数组	OUT	通信状态（最大 21 个）	TRUE,FALSE	FALSE
xError	BOOL 数组	OUT	通信出错（最大 21 个）	TRUE,FALSE	FALSE
mbErrorID	MB_ERROR 数组	OUT	通信错误 ID（最大 21 个）	参考定义	MB_NO_ERROR
bValid	BOOL	OUT	是否可运行在当前硬件平台	TRUE,FALSE	FALSE

上表 10-13 中参数 MB_STATE 数组的定义见下表 10-14。

表 10-14 ModbusTCPSlave 通信状态 MB_STATE 数组定义

名称	值	含义
MB_IDLE	0x00	空闲
MB_SOCKET_CREATE_HANDLE	0x01	创建 socket
MB_SOCKET_BIND	0x02	绑定 socket

名称	值	含义
MB_SOCKET_LISTEN	0x03	监听 socket
MB_SOCKET_ACCEPT	0x04	接受 socket 连接
MB_SOCKET_CONNECTED	0x05	socket 已连接
MB_SOCKET_UNCONNECT	0x06	socket 未连接
MB_SERIAL_CREATE_HANDLE	0x11	创建串口
MB_SERIAL_CONNECTED	0x15	串口已连接
MB_SERIAL_UNCONNECT	0x16	串口未连接

上表 10-13 中参数 MB_ERROR 数组的定义见下表 10-15。

表 10-15 ModbusTCPSlave 通信错误 MB_ERROR 数组定义

名称	值	含义
MB_NO_ERROR	0x00	正常
MB_ILLEGAL_FUNCTION	0x01	非法功能码
MB_ILLEGAL_DATA_ADDRESS	0x02	寄存器地址错误
MB_ILLEGAL_DATA_VALUE	0x03	数据检查错误
MB_ILLEGAL_DEVICE_FAILURE	0x04	站号地址不匹配
MB_SOCKET_CREATE_HANDLE_ERROR	0x10	创建 socket 失败
MB_SOCKET_BIND_ERROR	0x11	绑定 socket 失败
MB_SOCKET_ACCEPT_ERROR	0x12	接受 socket 连接失败
MB_SOCKET_COMM_TIMEOUT	0x13	通信超时：主站设置超时时间比 Slave 超时时间短
MB_SOCKET_COMM_UNCONNECT	0x14	未建立 socket 连接
MB_SOCKET_DNOT_RUN_PLATFORM	0x15	该功能库无权限运行在该控制器平台上
MB_SERIAL_CREATE_HANDLE_ERROR	0x20	创建串口失败

图 10-22、图 10-23 提供了 ModbusTCPSlave 通信示例程序。

图 10-22 列出了作为从站所需的基本变量定义，其中线圈、离散输入、保持寄存器、输入寄存器的数据 buffer 大小、定义的位置可根据实际需求进行调整。

外部主站访问该从站不同类型变量（线圈、离散输入、保持寄存器、输入寄存器）数据时，Modbus 寄存器起始地址均从 0 开始，彼此相互独立，输入参数中各类型数据 buffer 的大小，决定了该类型数据可访问的最大 Modbus 寄存器地址。

图 10-23 的功能实现部分，只需要将 ModbusTCPSlave 保持使能状态，在 bValid 为 TRUE 的状态时，主站即可对其发起连接，连接成功后，ModbusTCPSlave 中的 mbState 会更新状态为已连接，此时可以响应主站的读、写请求。在通信过程中，该功能块会按照 Modbus 功能协议，自动对主站的请求进行应答，无需用户编程处理；用户只需对线圈、离散输入、保持寄存器、输入寄存器中的数据进行业务逻辑管理。

modbusTCP_Slave_Demo

```
PROGRAM modbusTCP_Slave_Demo
VAR
    mdbusTCPSlave_instance : ModbusTCPSlave.ModbusTCPSlave;     // 定义功能块实例
    slaveIP : STRING := '192.168.0.1';                          // modbus TCP Slave (SC30x controller) IP address

    coilsDataBuf           : ARRAY[0..99] OF BOOL;              // slave buffer to store coils data         (Modbus address starts from 0)
    discreteInputsDataBuf  : ARRAY[0..99] OF BOOL;              // slave buffer to store discrete inputs    (Modbus address starts from 0)
    inputRegistersDataBuf  : ARRAY[0..99] OF WORD;              // slave buffer to store input registers    (Modbus address starts from 0)
    holdingRegistersDataBuf : ARRAY[0..99] OF WORD;             // slave buffet to store holding registers  (Modbus address starts from 0)
END_VAR
```

图 10-22 ModbusTCPSlave 示例程序：变量定义

```
// 使能 modbusTCPSlave
mdbusTCPSlave_instance(xEnable TRUE := TRUE,
                       bWatchDog FALSE := FALSE,
                       szIpAddress '192.168.0. ▸ := slaveIP '192.168.0. ▸ ,
                       uiPort 502 := 502,
                       usiUnitID 1 := 1,                        // modbus TCP Slave id (set as need in 1-255)
                       udiCommTimeout 2000 := 2000,             // ms
                       pbyCoilsData 16#B57E5C14 := ADR(coilsDataBuf),
                       udiCoilsSize 100 := SIZEOF(coilsDataBuf)/SIZEOF(BOOL),
                       pbyDiscreteInputsData 16#B57E5C78 := ADR(discreteInputsDataBuf),
                       udiDiscreteInputsSize 100 := SIZEOF(discreteInputsDataBuf)/SIZEOF(BOOL),
                       pwInputRegistersData 16#B57E5CDC := ADR(inputRegistersDataBuf),
                       udiInputRegistersSize 100 := SIZEOF(inputRegistersDataBuf)/SIZEOF(WORD),
                       pwHoldingRegistersData 16#B57E5DA4 := ADR(holdingRegistersDataBuf),
                       udiHoldingRegistersSize 100 := SIZEOF(holdingRegistersDataBuf)/SIZEOF(WORD),
                       mbState => ,
                       xBusy => ,
                       xError => ,
                       mbErrorID => ,
                       bValid => );
```

图 10-23 ModbusTCPSlave 示例程序：功能实现

10.2 TCP/UDP 自由通信

当仅使用 TCP 进行自由协议通信时，无需安装、添加 ModbusTCP 编译库，但需要添加 CODESYS 提供的 CAA Net Base Services 通信库，并且该通信库中也包含 UDP 通信编程接口。图 10-24 至图 10-27 介绍了如何添加该通信库。

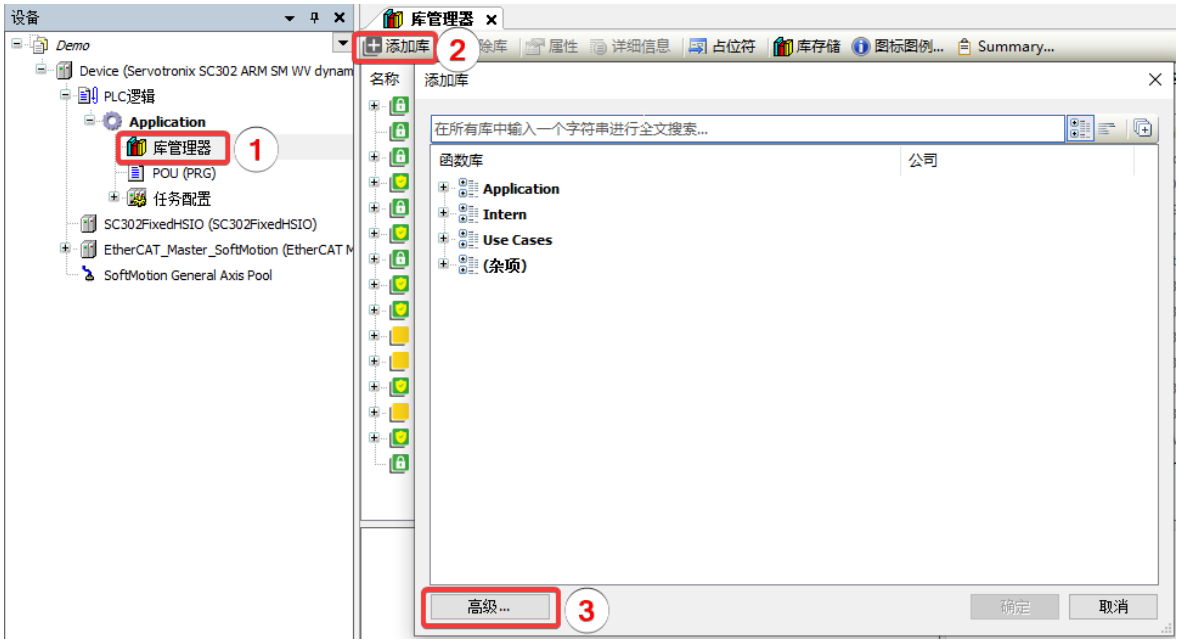


图 10-24 添加 CAA Net Base Services 通信库（一）

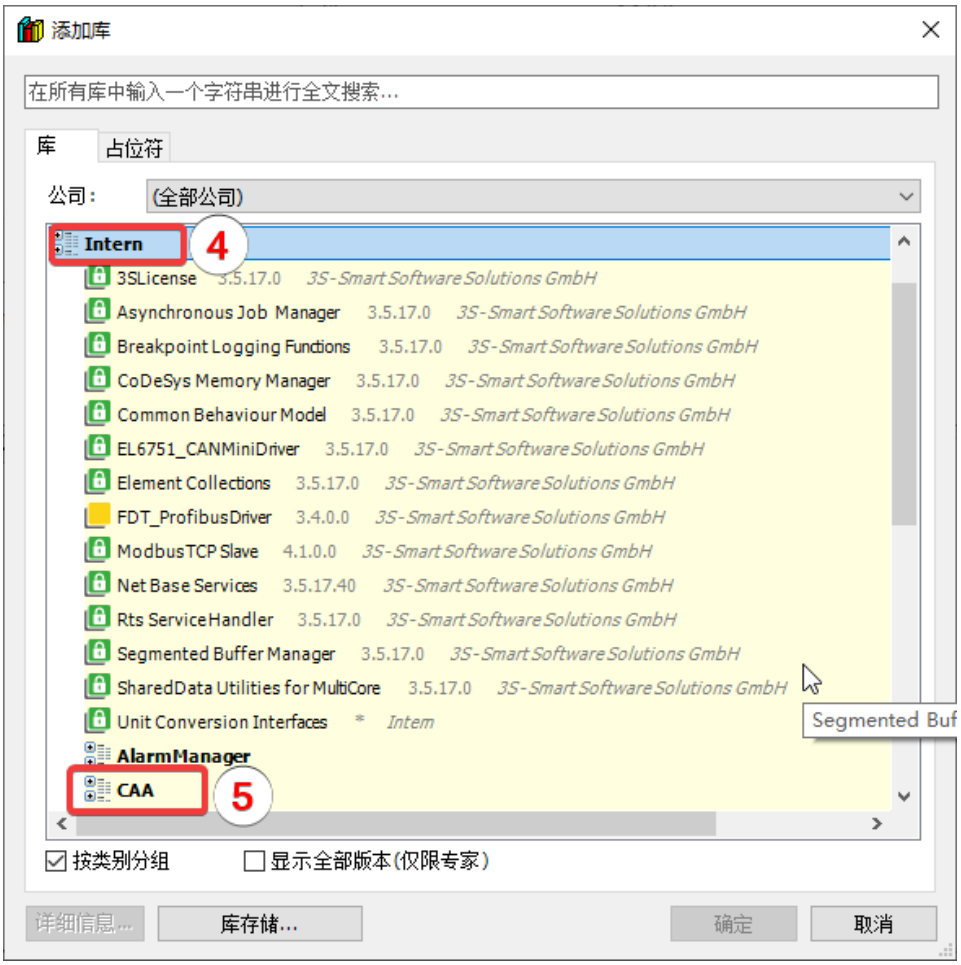


图 10-25 添加 CAA Net Base Services 通信库（二）

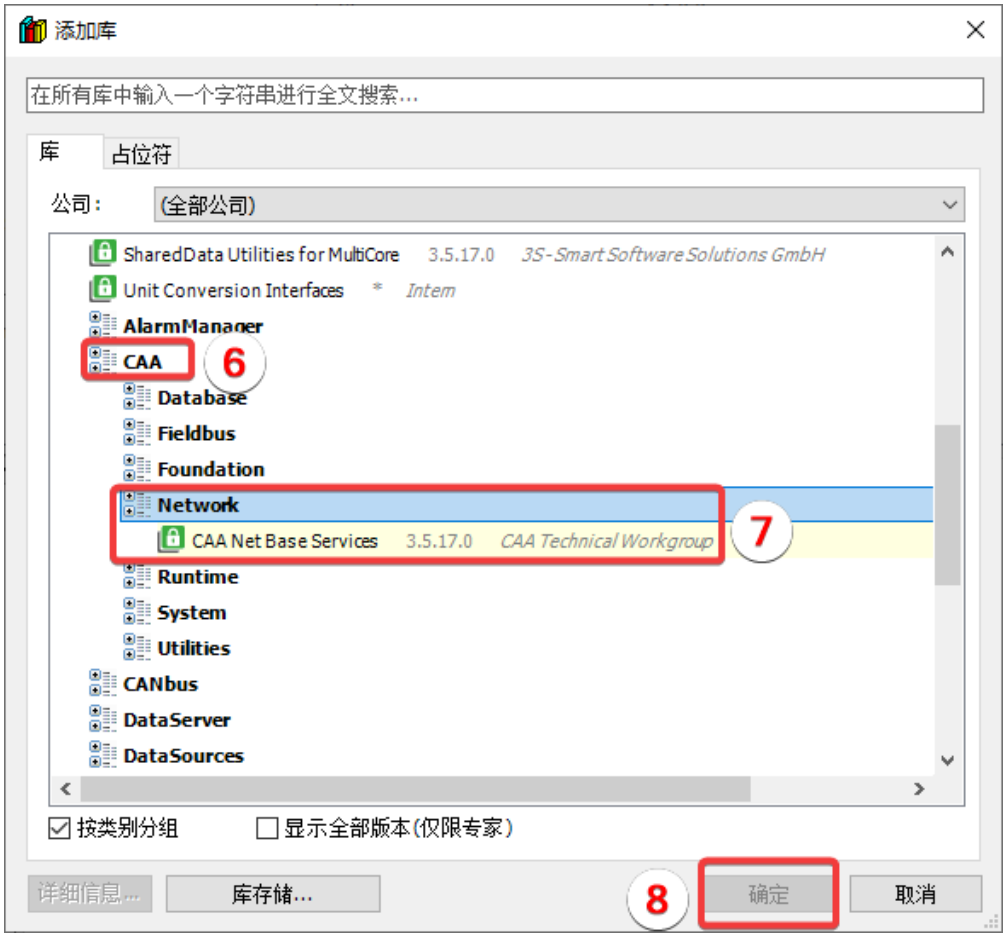


图 10-26 添加 CAA Net Base Services 通信库（三）

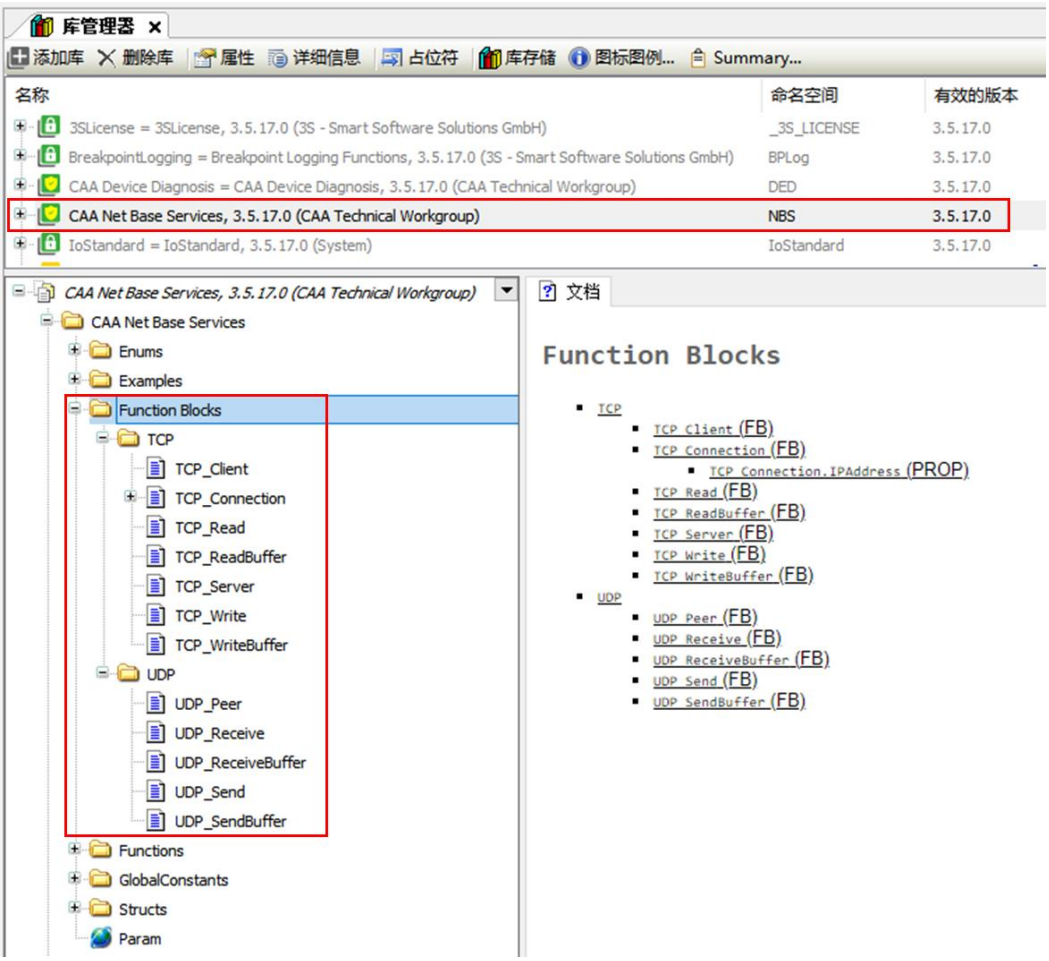


图 10-27 添加 CAA Net Base Services 通信库（四）

10.2.1 TCP Server

控制器作为 TCP Server，进行 TCP 通信时，可参考图 10-28、图 10-29 所示的通信示例程序。该示例程序实现的功能为，当控制器接收到客户端发送数据时，将接收数据原样返回给客户端。

freeTCP_Server_Demo

```
PROGRAM freeTCP_Server_Demo
VAR
    freeTCP_Server_instance     : NBS.TCP_Server;       // TCP服务端(控制器做服务器)
    freeTCP_Connection_instance : NBS.TCP_Connection;   // TCP链接管理(与服务端搭配使用)
    freeTCP_Read_instance       : NBS.TCP_Read;         // TCP数据读取
    freeTCP_write_instance      : NBS.TCP_Write;        // TCP数据写入

    localIP    : NBS.IP_ADDR := (sAddr:='192.168.0.1'); // 本地主机(控制器)地址
    localPort : UINT := 7575;                           // 本地主机(控制器)端口号

    connectEnable : BOOL := TRUE;                       // 发起TCP连接使能
    reconnect : BOOL := FALSE;                          // 重连触发使能
    delayCount : UINT := 0;                             // 重连延时等待

    recvBuffer : ARRAY[0..511] OF BYTE;
    readCount  : UDINT := 0;
    sendBuffer : ARRAY[0..511] OF BYTE;
    sendCount  : UDINT := 10;

    rxEnable   : BOOL := FALSE;                         // 读使能
    txExecute : BOOL := FALSE;                          // 发送，上升沿触发
END_VAR
```

图 10-28 freeTCP Server 通信示例程序：变量定义

```
//=============================================================================================================
// 使能TCP Server
//=============================================================================================================
freeTCP_Server_instance(xEnable TRUE := TRUE,
                        ipAddr := localIP,
                        uiPort 7575 := localPort 7575,
                        xDone => ,
                        xBusy => ,
                        xError => ,
                        eError => ,
                        hServer => );               // TCP Server句柄
//=============================================================================================================
// TCP Server使能成功后，使能 TCP Connection
//=============================================================================================================
IF freeTCP_Server_instance.hServer 3045873872 <> 0 THEN
    freeTCP_Connection_instance(xEnable TRUE := connectEnable TRUE,
                                hServer 3045873872 := freeTCP_Server_instance.hServer 3045873872,
                                xDone => ,
                                xBusy => ,
                                xError => ,
                                eError => ,
                                xActive => ,            // TCP连接有效
                                hConnection => );       // TCP连接句柄
ELSE
    freeTCP_Connection_instance(xEnable TRUE := FALSE);
END_IF
```

```
27  //=========================================================================================================
28  // TCP连接成功后，尝试读取是否接收到数据
29  //=========================================================================================================
30  IF freeTCP_Connection_instance.xActive TRUE THEN
31      rxEnable TRUE := TRUE;
32  ELSE
33      rxEnable TRUE := FALSE;
34  END_IF
35  freeTCP_Read_instance(xEnable TRUE := rxEnable TRUE,
36                        xDone => ,
37                        xBusy => ,
38                        xError => ,
39                        hConnection 3045886608 := freeTCP_Connection_instance.hConnection 3045886608,
40                        szSize 512 := SIZEOF(recvBuffer),
41                        pData 3046001172 := ADR(recvBuffer),
42                        eError => ,
43                        xReady => ,
44                        szCount 0 => readCount 0 );
45
46  //=========================================================================================================
47  // TCP连接成功后，如果接收到数据，则把接收数据返回
48  //=========================================================================================================
49  IF freeTCP_Connection_instance.xActive TRUE AND (readCount 0 > 0) AND NOT txExecute FALSE THEN
50      SysMemCpy(pDest := ADR(sendBuffer), pSrc := ADR(recvBuffer), udiCount := readCount 0 );
51      sendCount 35 := readCount 0 ;
52      txExecute FALSE := TRUE;
53      readCount 0 := 0;
54  END_IF
55  freeTCP_write_instance(xExecute FALSE := txExecute FALSE,
56                         udiTimeOut 1000000 := 1000000,
57                         xDone=> ,
58                         xBusy=> ,
59                         xError=> ,
60                         hConnection 3045886608 := freeTCP_Connection_instance.hConnection 3045886608,
61                         szSize 35 := sendCount 35 ,
62                         pData 3046001684 := ADR(sendBuffer),
63                         eError=> );
64  IF freeTCP_write_instance.xDone FALSE THEN
65      txExecute FALSE := FALSE;
66  END_IF
```

```
67  //=========================================================================================================
68  // 重连处理
69  //=========================================================================================================
70  IF freeTCP_Server_instance.hServer 3045873872 <> 0 THEN
71      IF connectEnable TRUE THEN
72          // 如果连接出错/超时，或者被Server关闭连接，尝试重连
73          IF freeTCP_Connection_instance.xError FALSE OR freeTCP_Connection_instance.xDone FALSE THEN
74              connectEnable TRUE := FALSE;
75              reconnect FALSE := TRUE;
76              delayCount 0 := 100;
77          END_IF
78      ELSE
79          IF reconnect FALSE THEN
80              IF delayCount 0 > 0 THEN
81                  delayCount 0 := delayCount 0 - 1;
82              ELSE
83                  reconnect FALSE := FALSE;
84                  connectEnable TRUE := TRUE;
85              END_IF
86          END_IF
87      END_IF
88  END_IFRETURN
```

图 10-29 freeTCP Server 通信示例程序：功能实现

10.2.2 TCP Client

当控制器作为 TCP Client，进行 TCP 通信时，可参考图 10-30、图 10-31 提供的通信示例程序。该示例程序实现的功能为，控制器每隔 1 秒将发送 buffer 内数据发送至 Server，并将接收到 Server 的数据保存至接收 buffer 内。

freeTCP_Client_Demo

```
PROGRAM freeTCP_Client_Demo
VAR
    freeTCP_Client_instance : NBS.TCP_Client;          // TCP客户端(控制器做客户端)
    freeTCP_Read_instance   : NBS.TCP_Read;            // TCP数据读取
    freeTCP_write_instance  : NBS.TCP_Write;           // TCP数据写入

    remoteIP : NBS.IP_ADDR :=( sAddr:='192.168.0.50'); // 远程主机IP地址
    remotePort : UINT := 8585;                         // 远程主机端口号

    connectEnable : BOOL := TRUE;                      // 发起TCP连接使能
    reconnect : BOOL := FALSE;                         // 重连触发使能
    delayCount : UINT := 0;                            // 重连延时等待

    recvBuffer : ARRAY[0..511] OF BYTE;
    readCount  : UDINT := 0;
    sendBuffer : ARRAY[0..511] OF BYTE := [16#30, 16#31, 16#32, 16#33, 16#34, 16#35, 16#36, 16#37, 16#38, 16#39];
    sendCount  : UDINT := 10;
    sendTimer : TON;

    rxEnable  : BOOL := FALSE;                         // 读使能
    txExecute : BOOL := FALSE;                         // 发送，上升沿触发
END_VAR
```

图 10-30 freeTCP Client 通信示例程序：变量定义

```
// 使能Client发起TCP连接
freeTCP_Client_instance(xEnable:= connectEnable,
                        xDone => ,
                        xBusy => ,
                        xError => ,
                        udiTimeOut := 1000000,      // 单位-us
                        ipAddr := remoteIP,         // 远程主机IP地址
                        uiPort := remotePort,       // 远程主机端口号
                        eError => ,
                        xActive => ,                // TCP连接有效
                        hConnection => );           // TCP连接句柄

//=============================================================================================
// TCP连接成功后，尝试读取是否接收到数据
//=============================================================================================
IF freeTCP_Client_instance.xActive THEN
    rxEnable := TRUE;
ELSE
    rxEnable := FALSE;
END_IF
freeTCP_Read_instance(xEnable := rxEnable,
                      xDone=> ,
                      xBusy=> ,
                      xError=> ,
                      hConnection := freeTCP_Client_instance.hConnection ,
                      szSize := SIZEOF(recvBuffer),          // 接收buffer大小
                      pData := ADR(recvBuffer),              // 接收buffer地址
                      eError=> ,
                      xReady=> ,                             // TRUE: 数据读取成功
                      szCount => readCount );                // 实际读取数据长度
//=============================================================================================
// TCP连接成功后，1s发送一次数据
//=============================================================================================
IF freeTCP_Client_instance.xActive AND NOT txExecute THEN
    sendTimer(IN := TRUE, PT := T#1S, Q =>, ET => );
    IF sendTimer.Q THEN
        txExecute := TRUE;
        sendTimer(IN := FALSE, PT := T#1S, Q =>, ET => );
    END_IF
ELSE
    sendTimer(IN := FALSE, PT := T#1S, Q =>, ET => );
END_IF
freeTCP_write_instance(xExecute := txExecute,          // 上升沿触发
                       udiTimeOut := 1000000,          // 单位-us
                       xDone=> ,
                       xBusy=> ,
                       xError=> ,
                       hConnection := freeTCP_Client_instance.hConnection,
                       szSize := sendCount ,
                       pData := ADR(sendBuffer),
                       eError=> );
IF freeTCP_write_instance.xDone OR freeTCP_write_instance.xError THEN
    txExecute := FALSE;
END_IF
```

```
55
56  //=========================================================================================================
57  // 重连处理
58  //=========================================================================================================
59  IF connectEnable TRUE THEN
60      // 如果连接出错/超时，或者被Server关闭连接，尝试重连
61      IF freeTCP_Client_instance.xError FALSE OR freeTCP_Client_instance.xDone FALSE THEN
62          connectEnable TRUE := FALSE;
63          reconnect FALSE := TRUE;
64          delayCount 0 := 100;
65      END_IF
66  ELSE
67      IF reconnect FALSE THEN
68          IF delayCount 0 > 0 THEN
69              delayCount 0 := delayCount 0 - 1;
70          ELSE
71              reconnect FALSE := FALSE;
72              connectEnable TRUE := TRUE;
73          END_IF
74      END_IF
75  END_IFRETURN
```

图 10-31 freeTCP Client 通信示例程序：功能实现

10.2.3 UDP

控制器使用 UDP 通信时可参考图 10-32 和图 10-33 提供的示例程序。该示例程序实现的功能为，当控制器接收到 UDP 数据时，将接收数据按照指定 IP 地址和端口号转发出去。

freeUDP_Demo

```
PROGRAM freeUDP_Demo
VAR
    UDP_Peer_instance : NBS.UDP_Peer;
    UDP_Recv_instance : NBS.UDP_Receive;
    UDP_Send_instance : NBS.UDP_Send;
    localIP           : NBS.IP_ADDR := ( sAddr := '192.168.0.1' );  // 控制器自身IP
    localPort         : UINT := 9000;
    remoteIP          : NBS.IP_ADDR := ( sAddr := '192.168.0.50' ); // 远程IP
    remotePort        : UINT := 8500;

    recvFromIP        : NBS.IP_ADDR := ( sAddr := '0.0.0.0' );
    recvFromPort      : UINT := 0;

    recvBuffer        : ARRAY[0..1023] OF BYTE;
    recvLen           : UDINT := 0;
    sendBuffer        : ARRAY[0..1023] OF BYTE;
    sendLen           : UDINT := 0;

    peerEnable : BOOL := TRUE;          // 使能 UDP_Peer_instance
    recvEnable : BOOL := FALSE;         // 使能 UDP_Recv_instance
    sendExecute : BOOL := FALSE;        // 触发 UDP_Send_instance
END_VAR
```

图 10-32 UDP 通信示例程序：变量定义

```
//=============================================================================================
// UDP使能打开
//=============================================================================================
UDP_Peer_instance(xEnable:=TRUE := peerEnable:=TRUE,
                  xDone => ,
                  xBusy => ,
                  xError => ,
                  ipAddr := localIP,
                  uiPort:=9000 := localPort:=9000,
                  ipMultiCast := ,
                  eError => ,
                  xActive => ,
                  hPeer => );
//=============================================================================================
// 若接收到数据，将数据拷贝至发送bufer
//=============================================================================================
IF UDP_Peer_instance.xActive:=TRUE AND UDP_Peer_instance.hPeer:=3046005376 <> 0 THEN
    recvEnable:=TRUE := TRUE;
    IF recvLen:=0 > 0 AND NOT sendExecute:=FALSE THEN
        SysMemCpy(pDest := ADR(sendBuffer), pSrc := ADR(recvBuffer), udiCount := recvLen:=0);
        sendLen:=0 := recvLen:=0;
        sendExecute:=FALSE := TRUE;
        recvLen:=0 := 0;
    END_IF
ELSE
    recvEnable:=TRUE := FALSE;
END_IF
```

```
28  //=============================================================================================
29  // 检查是否接收到数据
30  //=============================================================================================
31  UDP_Recv_instance(xEnable:=TRUE recvEnable:=TRUE,
32                    xDone => ,
33                    xBusy => ,
34                    xError => ,
35                    hPeer:=3046005376 := UDP_Peer_instance.hPeer:=3046005376,
36                    szSize:=1024 := SIZEOF(recvBuffer),
37                    pData:=3046008456 := ADR(recvBuffer),
38                    eError => ,
39                    xReady => ,
40                    ipFrom => ,
41                    uiPortFrom => ,
42                    szCount => );
43  IF UDP_Recv_instance.xReady=FALSE THEN
44      recvLen:=0 := UDP_Recv_instance.szCount:=0;
45      recvFromIP := UDP_Recv_instance.ipFrom;
46      recvFromPort:=8585 := UDP_Recv_instance.uiPortFrom:=0;
47  ELSE
48      recvLen:=0 := 0;
49  END_IF
50  //=============================================================================================
51  // 发送数据
52  //=============================================================================================
53  UDP_Send_instance(xExecute:=FALSE := sendExecute:=FALSE,
54                    udiTimeout := ,
55                    xDone => ,
56                    xBusy => ,
57                    xError => ,
58                    hPeer:=3046005376 := UDP_Peer_instance.hPeer:=3046005376,
59                    ipAddr := remoteIP,
60                    uiPort:=8500 := remotePort:=8500,
61                    szSize:=0 := sendLen:=0,
62                    pData:=3046009480 := ADR(sendBuffer),
63                    eError => );
64  IF UDP_Send_instance.xDone=FALSE OR UDP_Send_instance.xError=FALSE THEN
65      sendExecute:=FALSE := FALSE;
66      sendLen:=0 := 0;
67  END_IFRETURN
```

图 10-33 UDP 通信示例程序：功能实现

10.3 Modbus RTU 通信

在进行 Modbus RTU 通信协议之前，需要先安装厂家提供的 Modbus RTU 编译库，串口自由协议通信无需安装。

10.3.1 下载和安装 Modbus RTU 编译库

Modbus RTU 编译库文件可在高创官网 www.servotronix.cn 首页，依次单击“资料下载中心”->“运动控制器&I/O 模块” ->“紧凑型运动控制器”->“SC301” ->“编译库”，在弹出的页面进行下载，并保存在合适位置。

SC301 的 Modbus RTU 编译库文件，包括 Master 和 Slave 两个文件，其格式均为.complied-library。后续版本可能会有更新，请留意。

在使用 Modbus RTU 编译库接口进行编程之前，需要先进行编译库的安装，并添加到当前工程中。详细操作步骤可以参考“**10.1.1 下载和安装 Modbus TCP 编译库**”章节，在此不再重复。

10.3.2 作为 Modbus RTU Master(主站)

使用编译库提供的编程接口来实现 Modbus RTU Master(主站)功能，针对 Modbus 协议，功能块 ModbusRTUMaster 提供了 8 种方法（Method）进行处理，如下图 10-34 所示。

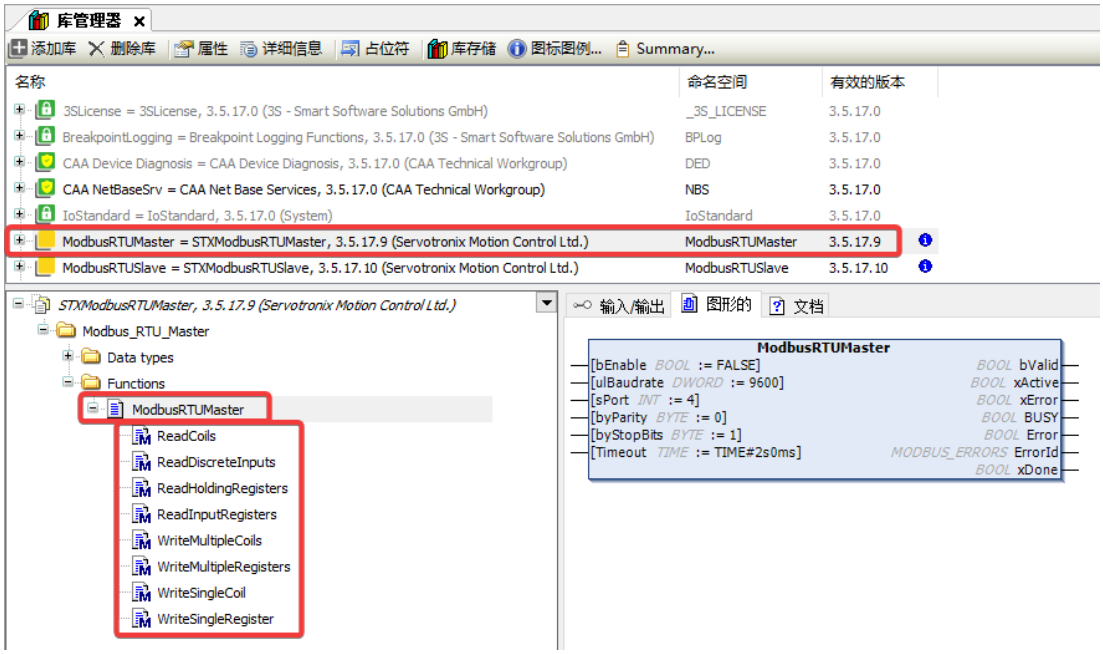


图 10-34 ModbusRTUMaster 编译库接口

下表 10-16 给出了 ModbusRTUMaster 提供的所有功能块及 Method(类似于函数)列表，下面分别说明。

表 10-16 ModbusRTUMaster 功能块及 Method 列表

名称	类型	功能码	参数说明
ModbusRTUMaster	Function Block	--	ModbusRTU 主站功能块 （命名空间为 ModbusRTUMaster）
ReadCoils	Method	0x01	读线圈
ReadDiscreteInputs	Method	0x02	读离散输入
ReadHoldingRegisters	Method	0x03	读保持寄存器
ReadInputRegisters	Method	0x04	读输入寄存器
WriteMultipleCoils	Method	0x0F	写多个线圈
WriteMultipleRegisters	Method	0x10	写多个保持寄存器
WriteSingleCoil	Method	0x05	写单个线圈
WriteSingleRegister	Method	0x06	写单个保持寄存器

(1) ModbusRTUMaster

ModbusRTUMaster：ModbusRTU 主站功能块。

ModbusRTUMaster 功能块指令图见下图 10-35。

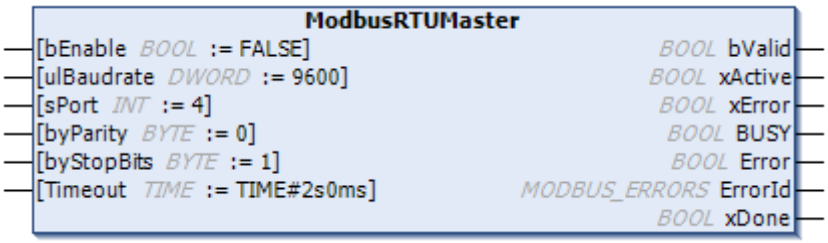


图 10-35 ModbusRTUMaster 指令图

ModbusRTUMaster 功能块输入/输出接口参数见下表 10-17。

表 10-17 ModbusRTUMaster 参数说明

参数名称	数据类型	IN/OUT	参数说明	取值范围	默认值
bEnable	BOOL	IN	功能块使能，使用中保持 TRUE	FALSE，TRUE	FALSE
ulBaudrate	DWORD	IN	波特率，单位 bps	4800、9600、19200、38400、57600、115200	9600
sPort	UINT	IN	串口端口号	0-disable; 1-COM1; 2-COM2; 3-COM3; 4-COM4 (RS485)	4
byParity	BYTE	IN	奇偶校验方式	0-无; 1-奇校验; 2-偶校验	0
byStopBits	BYTE	IN	停止位数	1-1bit; 2-1.5bit; 3-2bit	1
Timeout	TIME	IN	从站应答超时时间	遵循数据类型	T#2S
bValid	BOOL	OUT	是否可运行在当前硬件平台上	TRUE,FALSE	FALSE
xActive	BOOL	OUT	串口是否可用	TRUE,FALSE	FALSE
xError	BOOL	OUT	串口打开失败	TRUE,FALSE	FALSE
BUSY	BOOL	OUT	通信进行中	TRUE,FALSE	FALSE
Error	BOOL	OUT	Modbus 消息错误	TRUE,FALSE	FALSE
ErrorId	MODBUS_ERRORS 枚举	OUT	Modbus 消息错误 ID	见定义	MODBUS_ERROR_NO_ERROR
xDone	BOOL	OUT	Modbus 消息交互成功	TRUE,FALSE	FALSE

当通信出现错误时，ERROR 标志位会置为 TRUE，并通过 ErrorId 给出具体的错误信息，MODBUS_ERRORS 的定义及含义见表 10-18。

表 10-18 MODBUSRTUMaster 通信错误码 MODBUS_ERRORS 枚举定义

名称	值	含义
MODBUSERROR_NO_ERROR	0	正常
MODBUSERROR_ILLEGAL_FUNCTION	1	非法功能码
MODBUSERROR_ILLEGAL_DATA_ADDRESS	2	寄存器地址错误
MODBUSERROR_ILLEGAL_DATA_VALUE	3	非法数据

名称	值	含义
MODBUSERROR_SLAVE_DEVICE_FAILURE	4	从设备故障， 无法响应
MODBUSERROR_ACKNOWLEDGE	5	从设备已收到请求，需较长处理时间，主设备需要发送查询以确认完成
MODBUSERROR_SLAVE_DEVICE_BMSY	6	从设备忙
MODBUSERROR_NEGATIVE_ACKNOWLEDGE	7	从设备无法完成请求，可能因为某些条件未满足
MODBUSERROR_MEMORY_PARITY	8	从设备检测到存储器奇偶校验错误
MODBUSERROR_reserved9	9	保留
MODBUSERROR_GATEWAY_PATH_UNAVAILABLE	10	网关路径不可用
MODBUSERROR_GATEWAY_TARGET_DEVICE_FAILED_TO_RESPOND	11	网关目标设备失败
MODBUSERROR_CHARREC_TIMEOUT	20	接收等待超时
MODBUSERROR_ILLEGAL_DATA_SIZE	21	数据长度错误
MODBUSERROR_ILLEGAL_DEVICE_ADDRESS	22	非法设备地址
MODBUSERROR_ILLEGAL_DESTINATION_ADDRESS	23	非法目标地址
MODBUSERROR_ILLEGAL_DESTINATION_SIZE	24	非法目标长度
MODBUSERROR_NO_RESPONSE	25	无响应
MODBUSERROR_TXBUFFOVERRUN	102	发送缓存溢出
MODBUSERROR_SENDTIMEOUT	103	发送超时
MODBUSERROR_DATASIZEOVERRUN	107	数据长度溢出
MODBUSERROR_STRINGOVERRUN	110	字符串溢出
MODBUSERROR_INVALIDPOINTER	120	无效指针 NULL
MODBUSERROR_CRC	150	CRC 错误
MODBUSERROR_INVALIDMEMORYADDRESS	232	无效内存地址
MODBUSERROR_TRANSMITBUFFERTOOSMALL	233	发送缓存太小

(2) ReadCoils

ReadCoils：读线圈，其指令图及输入输出参数分别见图 10-36 和表 10-19。

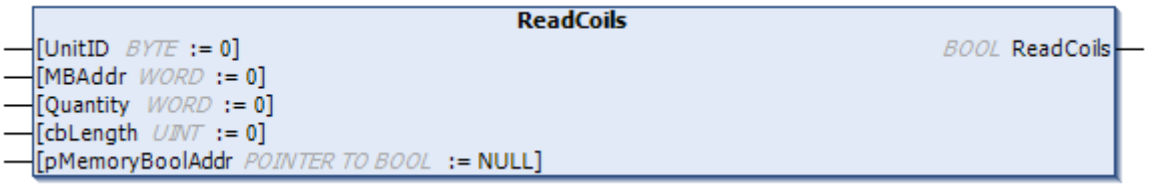


图 10-36 ReadCoils 指令图

表 10-19 ReadCoils 参数说明

参数名称	数据类型	IN/OUT	参数说明	取值范围	默认值
UnitID	BYTE	IN	从站站点号	1～255	0
MBAddr	WORD	IN	从站 Modbus 寄存器起始地址	取决于从站	0
Quantity	WORD	IN	读线圈数量（最大 2008）	1～2008	0
cbLength	UINT	IN	主站接收读取数据 buffer 大小（字节），需满足 cbLength >= Quantity	不小于读取数据	0
pMemoryBoolAddr	POINTER TO BOOL	IN	主站接收读取数据 buffer 的地址	非 NULL	NULL
ReadCoils	BOOL	OUT	不使用，无需关注	-	-

(3) ReadDiscreteInputs

ReadDiscreteInputs：读离散输入，其指令图和输入输出参数分别见图 10-37 和表 10-20。

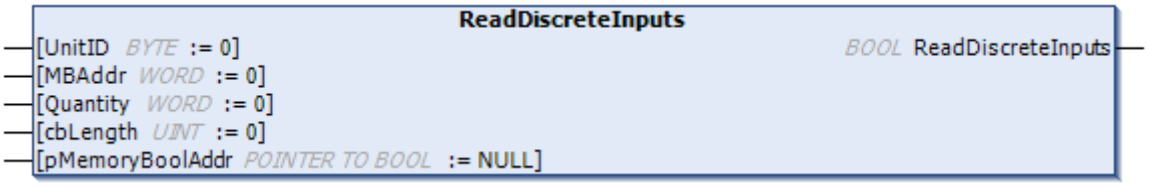


图 10-37 ReadDiscreteInputs 指令图

表 10-20 ReadDiscreteInputs 参数说明

参数名称	数据类型	IN/OUT	参数说明	取值范围	默认值
UnitID	BYTE	IN	从站站点号	1～255	0
MBAddr	WORD	IN	从站 Modbus 寄存器起始地址	取决于从站	0
Quantity	WORD	IN	读离散输入数量（最大 2008）	1～2008	0
cbLength	UINT	IN	主站接收读取数据 buffer 大小（字节），需满足 cbLength >= Quantity	不小于读取数据大小	0
pMemoryBoolAddr	POINTER TO BOOL	IN	主站接收读取数据 buffer 的地址	非 NULL	NULL
ReadDiscreteInputs	BOOL	OUT	不使用，无需关注	-	-

(4) ReadHoldingRegisters

ReadHoldingRegisters：读保持寄存器，其指令图和输入输出参数分别见图 10-38 和表 10-21。

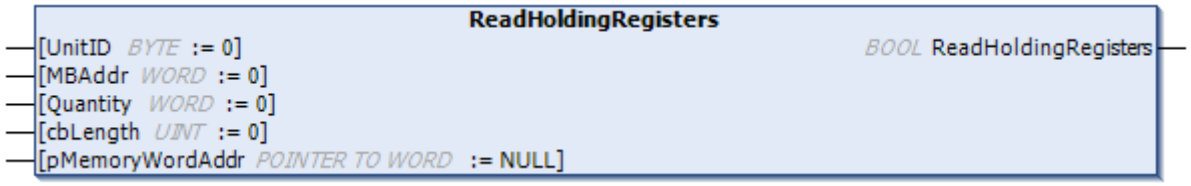


图 10-38 ReadHoldingRegisters

表 10-21 ReadHoldingRegisters 参数说明

参数名称	数据类型	IN/OUT	参数说明	取值范围	默认值
UnitID	BYTE	IN	从站站点号	1～255	0
MBAddr	WORD	IN	从站 Modbus 寄存器起始地址	取决于从站	0
Quantity	WORD	IN	读保持寄存器数量	1～125	0
cbLength	UINT	IN	主站接收读取数据 buffer 大小（字节），需满足 cbLength >= 2*Quantity	不小于读取数据占用大小	0
pMemoryWordAddr	POINTER TO WORD	IN	主站接收读取数据 buffer 的地址	非 NULL	NULL
ReadHoldingRegisters	BOOL	OUT	不使用，无需关注	-	-

(5) ReadInputRegisters

ReadInputRegisters：读输入寄存器，其指令图和输入输出参数分别见图 10-39 和表 10-22。

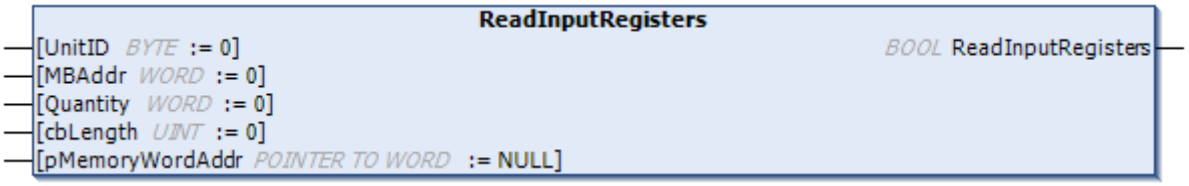


图 10-39 ReadInputRegisters 指令图

表 10-22 ReadInputRegisters 参数说明

参数名称	数据类型	IN/OUT	参数说明	取值范围	默认值
UnitID	BYTE	IN	从站站点号	1～255	0
MBAddr	WORD	IN	从站 Modbus 寄存器起始地址	取决于从站	0
Quantity	WORD	IN	读输入寄存器数量	1～125	0
cbLength	UINT	IN	主站接收读取数据 buffer 大小（字节），需满足 cbLength >= 2*Quantity	不小于读取数据大小	0
pMemoryWordAddr	POINTER TO WORD	IN	主站接收读取数据 buffer 的地址	非 NULL	NULL
ReadInputRegisters	BOOL	OUT	不使用，无需关注	-	-

(6) WriteMultipleCoils

WriteMultipleCoils：写多个线圈，其指令图和输入输出参数分别见图 10-40 和表 10-23。

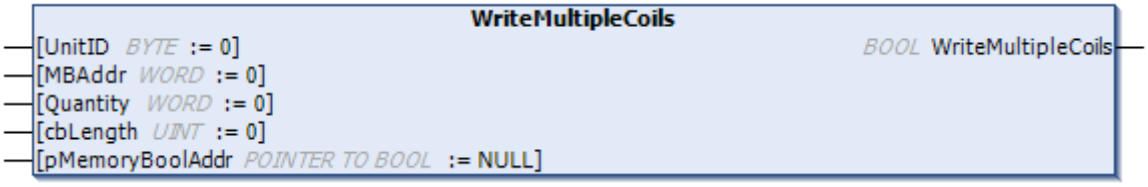


图 10-40 WriteMultipleCoils 指令图

表 10-23 WriteMultipleCoils 参数说明

参数名称	数据类型	IN/OUT	参数说明	取值范围	默认值
UnitID	BYTE	IN	从站站点号	1～255	0

参数名称	数据类型	IN/OUT	参数说明	取值范围	默认值
MBAddr	WORD	IN	从站 Modbus 寄存器起始地址	取决于从站	0
Quantity	WORD	IN	写多个线圈数量	1～1976	0
cbLength	UINT	IN	主站发送数据 buffer 大小（字节），需满足 cbLength >= Quantity	不小于发送数据大小	0
pMemoryBoolAddr	POINTER TO BOOL	IN	主站发送数据 buffer 的地址	非 NULL	NULL
WriteMultipleCoils	BOOL	OUT	不使用，无需关注	-	-

(7) WriteMultipleRegisters

WriteMultipleRegisters：写多个保持寄存器，其指令图和输入输出参数分别见图 10-41 和表 10-24。

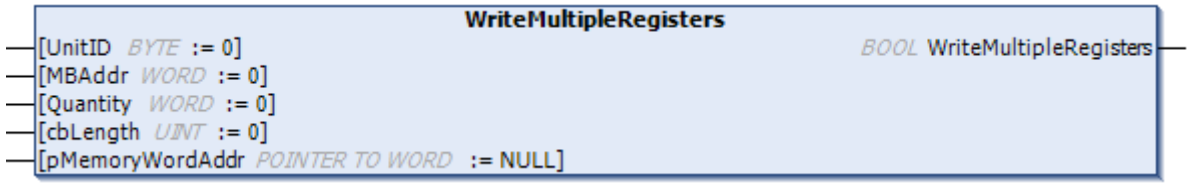


图 10-41 WriteMultipleRegisters 指令图

表 10-24 WriteMultipleRegisters 参数说明

参数名称	数据类型	IN/OUT	参数说明	取值范围	默认值
UnitID	BYTE	IN	从站站点号	1～125	0
MBAddr	WORD	IN	从站 Modbus 寄存器起始地址	取决于从站	0
Quantity	WORD	IN	写多个保持寄存器数量	1～123	0
cbLength	UINT	IN	主站发送数据 buffer 大小（字节）需满足 cbLength >= 2*Quantity	不小于发送数据占用大小	0
pMemoryWordAddr	POINTER TO WORD	IN	主站发送数据 buffer 的地址	非 NULL	NULL
WriteMultipleRegisters	BOOL	OUT	不使用，无需关注	-	-

(8) WriteSingleCoil

WriteSingleCoil：写单个线圈，其指令图和输入输出参数分别见图 10-42 和表 10-25。

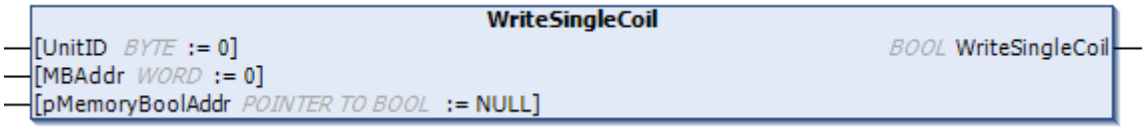


图 10-42 WriteSingleCoil 指令图

表 10-25 WriteSingleCoil 参数说明

参数名称	数据类型	IN/OUT	参数说明	取值范围	默认值
UnitID	BYTE	IN	从站站点号	1～255	0
MBAddr	WORD	IN	从站 Modbus 寄存器起始地址	取决于从站	0
pMemoryBoolAddr	POINTER TO BOOL	IN	主站发送数据 buffer 的地址	非 NULL	NULL
WriteSingleCoil	BOOL	OUT	不使用，无需关注	-	-

(9) WriteSingleRegister

WriteSingleRegister：写单个保持寄存器，其指令图和输入输出参数分别见图 10-43 和表 10-26。

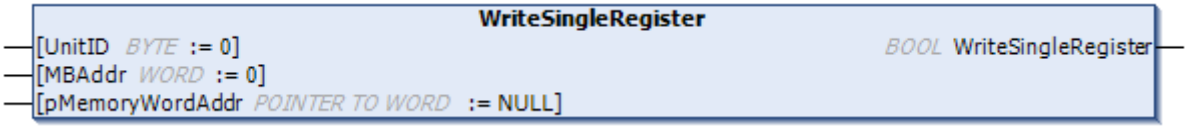


图 10-43 WriteSingleRegister 指令图

表 10-26 WriteSingleRegister 参数说明

参数名称	数据类型	IN/OUT	参数说明	取值范围	默认值
UnitID	BYTE	IN	从站站点号	1～255	0
MBAddr	WORD	IN	从站 Modbus 寄存器起始地址	取决于从站	0
pMemoryWordAddr	POINTER TO WORD	IN	主站发送数据 buffer 的地址	非 NULL	NULL
WriteSingleRegister	BOOL	OUT	不使用，无需关注	-	-

图 10-44 列出了该功能所需的基本变量定义，其中线圈、离散输入、保持寄存器、输入寄存器的数据 buffer 大小、定义的位置可根据实际需求进行调整。

图 10-45 的功能实现部分，使用状态机依次轮询执行 8 个 Modbus 功能函数。在实际应用中，根据实际需求，选择功能函数，但要避免在当前功能函数执行过程中（即 BUSY 为 TRUE 时），修改其输入参数、或者切换至其它功能函数。

当前功能函数执行完成（xDone 为 TRUE）、或者出错（Error 为 TRUE）后，可以切换至下一功能函数，且在该功能函数调用之前，根据需要变更其输入参数。

如果需要修改波特率，需要先关闭 ModbusRTUMaster 使能，修改波特率变量，然后再使能 ModbsRTUMaster 即可。

modbusRTU_Master_Demo

```
PROGRAM modbusRTU_Master_Demo
VAR
    mdbusRTUMaster_instance : ModbusRTUMaster.ModbusRTUMaster;          // 定义功能块实例
    baudRate     : DWORD := 115200;          // baud rate: 4800, 9600, 19200, 38400, 57600, 115200
    serialPort   : INT := 4;                 // serialPort: 0-disable, 1-COM1, 2-COM2, 3-COM3, 4-COM4(SC301&SC302 RS485), 5-COM5(SC302 RS232)
    parity       : BYTE := 0;                // parity: 0-no parity, 1-odd parity, 2-even parity
    stopBits     : BYTE := 1;                // stop bit: 1-1stopBit, 2-1.5stopBit, 3-2stopBit
    step : BYTE := 0;

    coilsBuf         : ARRAY[0..99] OF BOOL;    // master buffer to store coils data
    discreteInputBuf : ARRAY[0..99] OF BOOL;    // master buffer to store discrete inputs data
    holdRegisterBuf  : ARRAY[0..99] OF WORD;    // master buffer to store holding registers data
    inputRegisterBuf : ARRAY[0..99] OF WORD;    // master buffer to store input registers data
END_VAR
```

图 10-44 ModbusRTUMaster 示例程序：变量定义

```
// 使能 modbusRTUMaster
mdbusRTUMaster_instance(bEnable:=TRUE,
                        ulBaudrate:=115200 := baudRate:=115200,
                        sPort:=4 := serialPort:=4,
                        byParity:=0 := parity:=0,
                        byStopBits:=1 := stopBits:=1,
                        Timeout:=T#2s := T#2S,
                        bValid =>,                  // TRUE: can run on the hardware platform (SC301/SC302/MC804)
                        xActive =>,                 // TRUE: serial communication can be used
                        xError =>,                  // TRUE: serial communication/port cannot be used
                        BUSY =>,                    // TRUE: in modbus communication
                        Error =>,                   // TRUE: modbus message error
                        ErrorId =>,                 //       modbus message error id
                        xDone =>                    // TRUE: modbus read or write operation completed, keep 1 cycle
);

// 硬件检查通过，串口打开成功后：启动 ModbusRTU 通信
IF mdbusRTUMaster_instance.bValid=TRUE AND mdbusRTUMaster_instance.xActive=TRUE THEN
    CASE step:=7 OF
        //=============================================================================================================
        // 读写线圈操作 (read or write coils operation)
        //=============================================================================================================
        0:  // 功能码0x01：读线圈 (read coils)
            mdbusRTUMaster_instance.ReadCoils(UnitID := 1,                                  // 从站站点号
                                              MBAddr := 0,                                  // 从站线圈modbus首地址（此次读操作）
                                              Quantity := 5,
                                              cbLength := SIZEOF(coilsBuf),                 // 内存大小（字节）
                                              pMemoryBoolAddr := ADR(coilsBuf[0]=FALSE));       // 存放线圈数据内存首地址
        1:  // 功能码0x0F：写多个线圈 (write multiple coils)
            mdbusRTUMaster_instance.WriteMultipleCoils(UnitID := 1,                         // 从站站点号
                                                       MBAddr := 5,                         // 从站线圈modbus首地址（此次写操作）
                                                       Quantity := 5,
                                                       cbLength := SIZEOF(coilsBuf) - 5,    // 内存大小（字节）
                                                       pMemoryBoolAddr := ADR(coilsBuf[5]=TRUE));   // 获取线圈数据内存首地址
        2:  // 功能码0x05：写单个线圈 (write single coil)
            mdbusRTUMaster_instance.WriteSingleCoil(UnitID := 1,                            // 从站站点号
                                                    MBAddr := 10,                           // 从站线圈modbus首地址（此次写操作）
                                                    pMemoryBoolAddr := ADR(coilsBuf[10]=FALSE));    // 获取线圈数据内存首地址

        //=============================================================================================================
        // 读写保持寄存器操作 (read or write holding register operation)
        //=============================================================================================================
        3:  // 功能码0x03：读保持寄存器 (read holding registers)
            mdbusRTUMaster_instance.ReadHoldingRegisters(UnitID := 1,                       // 从站站点号
                                                         MBAddr := 0,                       // 从站保持寄存器modbus首地址（此次读操作）
                                                         Quantity := 5,
                                                         cbLength := SIZEOF(holdRegisterBuf),   // 内存大小（字节）
                                                         pMemoryWordAddr := ADR(holdRegisterBuf[0]=1));  // 存放保持寄存器数据内存首地址
        4:  // 功能码0x10：写多个保持寄存器 (write multiple registers)
            mdbusRTUMaster_instance.WriteMultipleRegisters(UnitID := 1,                     // 从站站点号
                                                           MBAddr := 5,                     // 从站保持寄存器modbus首地址（此次写操作）
                                                           Quantity := 5,
                                                           cbLength := SIZEOF(holdRegisterBuf) - 5*2,   // 内存大小（字节）
                                                           pMemoryWordAddr := ADR(holdRegisterBuf[5]=66));// 获取保持寄存器值内存首地址
        5:  // 功能码0x06：写单个保持寄存器 (write single register)
            mdbusRTUMaster_instance.WriteSingleRegister(UnitID := 1,                        // 从站站点号
                                                        MBAddr := 10,                       // 从站保持寄存器modbus首地址（此次写操作）
                                                        pMemoryWordAddr := ADR(holdRegisterBuf[10]=1010));  // 获取保持寄存器值内存首地址

        //=============================================================================================================
        // 读离散输入操作 (read discrete input operation)
        //=============================================================================================================
        6:  // 功能码0x02：读离散输入 (read discrete inputs)
            mdbusRTUMaster_instance.ReadDiscreteInputs(UnitID := 1,                         // 从站站点号
                                                       MBAddr := 0,                         // 从站离散输入modbus首地址（此次读操作）
                                                       Quantity := 5,
                                                       cbLength := SIZEOF(discreteInputBuf),    // 内存大小（字节）
                                                       pMemoryBoolAddr := ADR(discreteInputBuf[0]=FALSE));  // 存储离散输入数据内存首地址

        //=============================================================================================================
        // 读输入寄存器操作 (read input registers operation)
        //=============================================================================================================
        7:  // 功能码0x04：读输入寄存器 (read input registers)
            mdbusRTUMaster_instance.ReadInputRegisters(UnitID := 1,                         // 从站站点号
                                                       MBAddr := 0,                         // 从站输入寄存器modbus首地址（此次读操作）
                                                       Quantity := 5,
                                                       cbLength := SIZEOF(inputRegisterBuf),    // 存放内存大小（字节）
                                                       pMemoryWordAddr := ADR(inputRegisterBuf[0]=11));  // 存放输入寄存器值内存的首地址
    END_CASE

    // switch to next function command
    IF mdbusRTUMaster_instance.xDone=FALSE OR mdbusRTUMaster_instance.Error=FALSE THEN
        step:=2 := step:=2 + 1;
        IF (step:=2 > 7) THEN
            step:=2 := 0;
        END_IF
    END_IF

ELSE
    step:=2 := 0;
END_IFRETURN
```

图 10-45 ModbusRTUMaster 示例程序：功能实现

10.3.3 作为 Modbus RTU Slave(从站)

通过编译库提供的编程接口来实现 Modbus RTU Slave(从站)功能，如下图所示。在功能块 ModbusRTUSlave 中，4 种（线圈、离散输入、保持寄存器、输入寄存器）变量的 Modbus 地址分配是相互独立的，且寄存器起始地址均从 0 开始，最大可访问地址由相应变量分配的空间大小来决定。

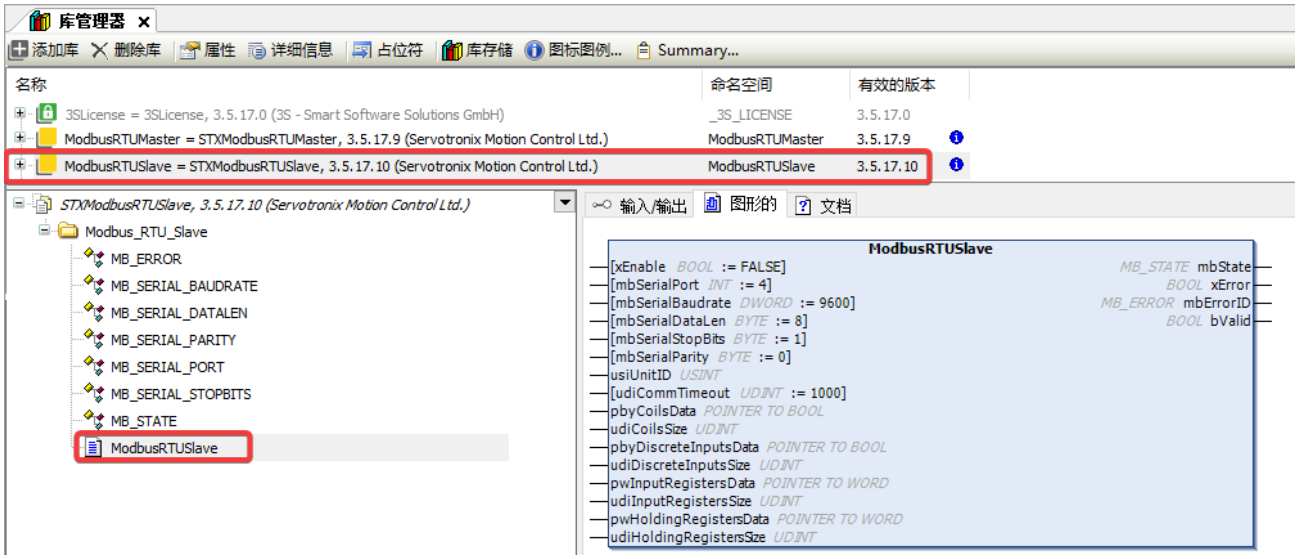


图 10-46 ModbusRTUSlave 编译库接口

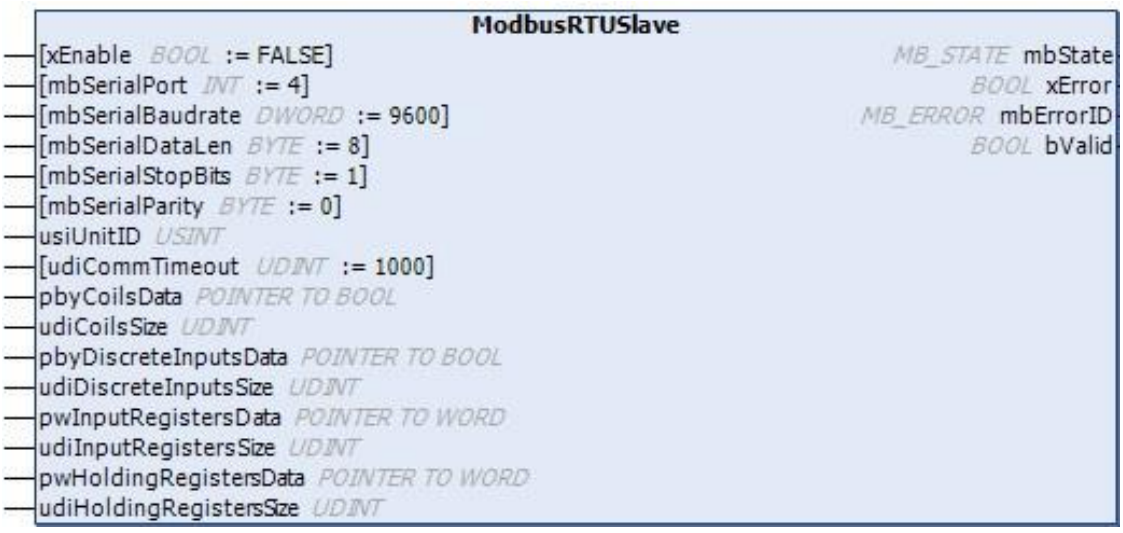


图 10-47 ModbusRTUSlave 指令图

表 10-27 ModbusRTUSlave 参数说明

参数名称	数据类型	IN/OUT	参数说明	取值范围	默认值
xEnable	BOOL	IN	功能块使能， 使用中保持 TRUE	TRUE,FALSE	FALSE
mbSerialPort	INT	IN	串口端口号	0-disable； 1-COM1； 2-COM2； 3-COM3； 4-COM4 (RS-485) 5-COM5 (RS-232)	4
mbSerialBaudrate	DWORD	IN	波特率，单位 bps	480、9600、 19200、38400、 57600、115200	9600
mbSerialDataLen	BYTE	IN	串口数据位数	8，7，6，5	8
mbSerialStopBits	BYTE	IN	停止位数	1-1bit； 2-1.5bit； 3-2bit	1
mbSerialParity	BYTE	IN	奇偶校验	0-无； 1-奇校验； 2-偶校验	0
usiUnitID	MSINT	IN	从站（控制器）站点号	1～255	0
udiCommTimeout	UDINT	IN	通信连接成功后的 超时时间（ms）	>0	1000
pbyCoilsData	POINTER TO BOOL	IN	从站线圈数据区地址	非 NULL	NULL
udiCoilsSize	UDINT	IN	从站线圈数据区大小， 单位 BOOL	不大于用户编程时定义的线圈区域大小	0
pbyDiscreteInputsData	POINTER TO BOOL	IN	从站离散输入数据区地址	非 NULL	NULL
udiDiscreteInputsSize	UDINT	IN	从站离散输入数据区大小，单位 BOOL	不大于用户编程时定义的离散输入区域大小	0
pwInputRegistersData	POINTER TO WORD	IN	从站输入寄存器数据区地址	非 NULL	NULL
udiInputRegistersSize	UDINT	IN	从站输入寄存器数据区大小，单位 WORD	不大于用户编程时定义的输入寄存器区域大小	0
pwHoldingRegistersData	POINTER TO WORD	IN	从站保持寄存器数据区地址	非 NULL	NULL

参数名称	数据类型	IN/OUT	参数说明	取值范围	默认值
udiHoldingRegistersSize	UDINT	IN	从站保持寄存器数据区大小，单位 WORD	不大于用户编程时定义的保持寄存器区域大小	0
mbState	MB_STATE 数组	OUT	连接状态	见定义	MB_IDLE
xError	BOOL	OUT	通信出错	TRUE,FALSE	FALSE
mbErrorID	MB_ERROR 数组	OUT	通信错误 ID	见定义	MB_NOERROR
bValid	BOOL	OUT	是否可运行在当前硬件平台	TRUE,FALSE	FALSE

上表 10-27 中参数 MB_STATE 数组定义见下表 10-28。

表 10-28ModbusRTUSlave 连接状态 MB_STATE 数组定义

名称	值	含义
MB_IDLE	0x00	空闲
MB_SOCKET_CREATE_HANDLE	0x01	创建 socket
MB_SOCKET_BIND	0x02	绑定 socket
MB_SOCKET_LISTEN	0x03	监听 socket
MB_SOCKET_ACCEPT	0x04	接受 socket 连接
MB_SOCKET_CONNECTED	0x05	socket 已连接
MB_SOCKET_UNCONNECT	0x06	socket 未连接
MB_SERIAL_CREATE_HANDLE_OK	0x11	创建串口成功
MB_SERIAL_CONNECTED	0x15	串口已连接
MB_SERIAL_UNCONNECT	0x16	串口未连接

上表 10-27 中参数 MB_ERROR 数组定义见下表 10-29。

表 10-29 ModbusRTUSlave 通信错误 MB_ERROR 数组定义

名称	值	含义
MB_NO_ERROR	0x00	正常
MB_ILLEGAL_FUNCTION	0x01	无效功能码
MB_ILLEGAL_DATA_ADDRESS	0x02	寄存器地址错误

名称	值	含义
MB_ILLEGAL_DATA_VALUE	0x03	数据检查错误
MB_ILLEGAL_DEVICE_FAILURE	0x04	站号地址不匹配
MB_SOCKET_CREATE_HANDLE_ERROR	0x10	创建 socket 失败
MB_SOCKET_BIND_ERROR	0x11	绑定 socket 失败
MB_SOCKET_ACCEPT_ERROR	0x12	接受 socket 连接失败
MB_SOCKET_COMM_TIMEOUT	0x13	通信超时：主站设置超时时间比 Slave 超时时间短
MB_SOCKET_COMM_UNCONNECT	0x14	未建立 socket 连接
MB_SOCKET_DNOT_RUN_PLATFORM	0x15	该功能库无权限运行在该控制器平台上
MB_SERIAL_CREATE_HANDLE_ERROR	0x20	创建串口失败
MB_SERIAL_DNOT_RUN_PLATFORM	0x21	该串口功能库无权限运行在该控制平台上
MB_SERIAL_COMM_TIMEOUT	0x22	串口通信超时

图 10-48 图 10-49 提供了 ModbusRTUSlave 应用示例程序。

图 10-48 列出了该功能所需的基本变量定义，其中线圈、离散输入、保持寄存器、输入寄存器的数据 buffer 大小、定义的位置可根据实际需求进行调整，该数据 buffer 的大小，决定了 Modbus 最大访问地址。

图 10-49 的功能实现部分，只需要将 ModbusRTUSlave 保持使能状态，在 bValid 为 TRUE 的状态时，主站即可对其发起连接，并进行 Modbus 功能访问。在应用过程中，该功能块会自动对主站的请求进行应答，用户只需对线圈、离散输入、保持寄存器、输入寄存器中的数据进行处理。

如果需要修改波特率，需要先关闭 ModbusRTUSlave 使能，修改波特率变量，然后再使能 ModbusRTUSlave 即可。

modbusRTU_Slave_Demo

```
PROGRAM modbusRTU_Slave_Demo
VAR
    mdbusRTUSlave_instance : ModbusRTUSlave.ModbusSerialSlave;          // 定义功能块实例
    baudRate      : DWORD := 115200;          // baud rate: 4800, 9600, 19200, 38400, 57600, 115200
    serialPort    : INT := 5;                 // serialPort: 0-disable, 1-COM1, 2-COM2, 3-COM3, 4-COM4(SC301&SC302 RS485), 5-COM5(SC302 RS232)
    dataLen       : BYTE := 8;                // 数据位数: 8,  7,  6, 5, 4
    parity        : BYTE := 0;                // parity: 0-no parity, 1-odd parity, 2-even parity
    stopBits      : BYTE := 1;                // stop bit: 1-1stopBit, 2-1.5stopBit, 3-2stopBit

    coilsDataBuf            : ARRAY[0..99] OF BOOL;             // slave buffer to store coils data         (Modbus address starts from 0)
    discreteInputsDataBuf   : ARRAY[0..99] OF BOOL;             // slave buffer to store discrete inputs    (Modbus address starts from 0)
    inputRegistersDataBuf   : ARRAY[0..99] OF WORD;             // slave buffer to store input registers    (Modbus address starts from 0)
    holdingRegistersDataBuf : ARRAY[0..99] OF WORD;             // slave buffet to store holding registers  (Modbus address starts from 0)
END_VAR
```

图 10-48 ModbusRTUSlave 示例程序：变量定义

```
// 使能 ModbusRTUSlave
mdbusRTUSlave_instance(xEnable:=TRUE,
                       mbSerialPort:=5 := serialPort:=5,
                       mbSerialBaudrate:=115200 := baudRate:=115200,
                       mbSerialDataLen:=8 := dataLen:=8,
                       mbSerialStopBits:=1 := stopBits:=1,
                       mbSerialParity:=0 := parity:=0,
                       usiUnitID:=1 := 1,
                       udiCommTimeout:=2000 := 2000,
                       pbyCoilsData:=16#B57E5F2C := ADR(coilsDataBuf),
                       udiCoilsSize:=100 := SIZEOF(coilsDataBuf)/SIZEOF(BOOL),
                       pbyDiscreteInputsData:=16#B57E5F90 := ADR(discreteInputsDataBuf),
                       udiDiscreteInputsSize:=100 := SIZEOF(discreteInputsDataBuf)/SIZEOF(BOOL),
                       pwInputRegistersData:=16#B57E5FF4 := ADR(inputRegistersDataBuf),
                       udiInputRegistersSize:=100 := SIZEOF(inputRegistersDataBuf)/SIZEOF(WORD),
                       pwHoldingRegistersData:=16#B57E60BC := ADR(holdingRegistersDataBuf),
                       udiHoldingRegistersSize:=100 := SIZEOF(holdingRegistersDataBuf)/SIZEOF(WORD),
                       mbState => ,
                       xError => ,
                       mbErrorID => ,
                       bValid => );RETURN
```

图 10-49 ModbusRTUSlave 示例程序：功能实现

10.4 串口自由通信

当仅仅使用串口进行自由协议通信时，无需安装、添加 Modbus RTU 编译库，但需要添加 CODESYS 提供的 SysCom 串口通信库。图 10-50 至图 10-53 介绍了如何添加该通信库。

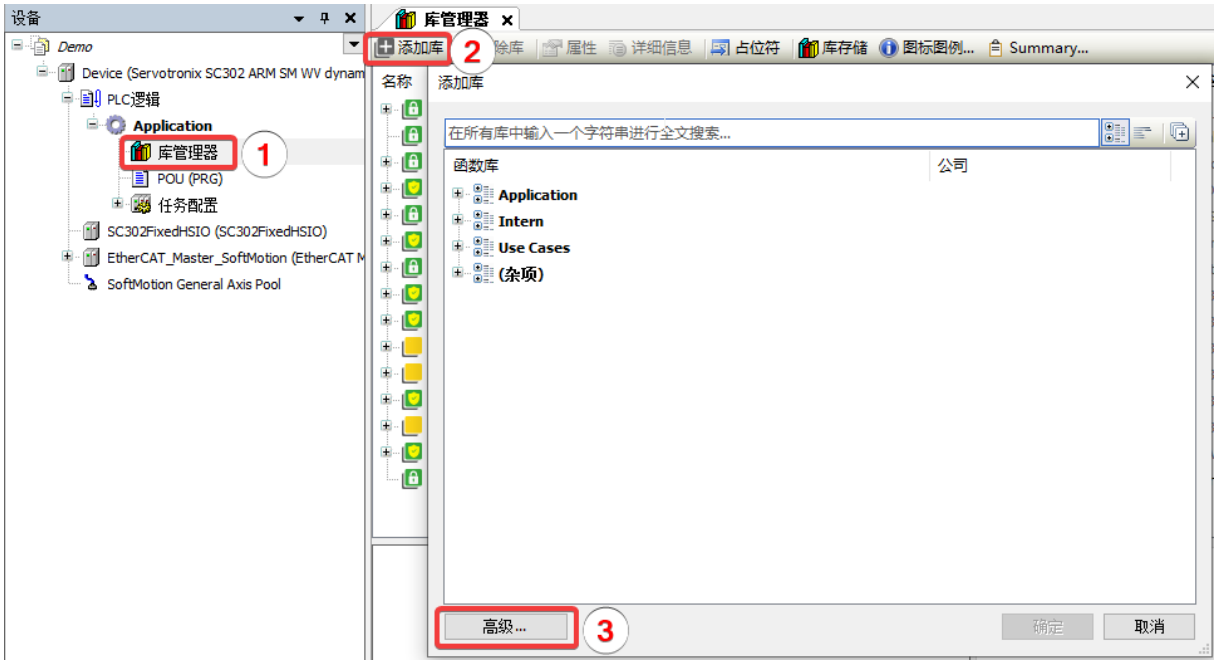


图 10-50 添加 SysCom 串口通信库（一）

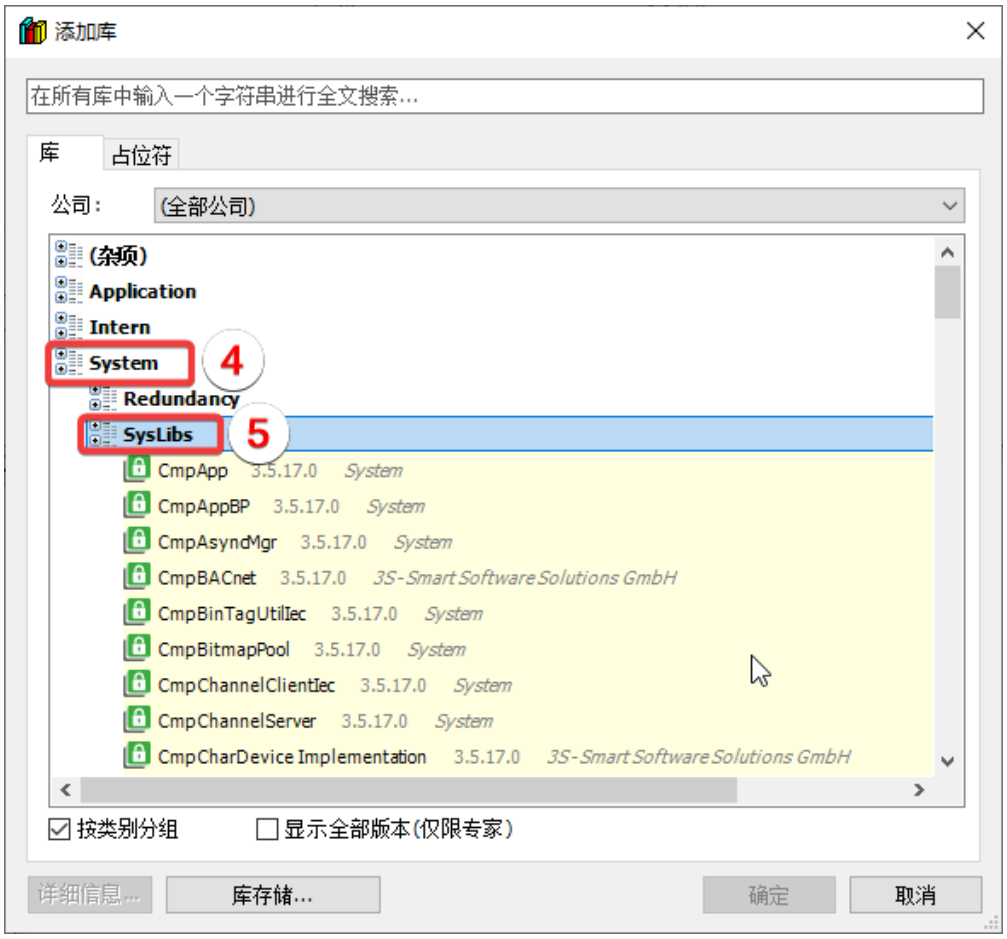


图 10-51 添加 SysCom 串口通信库（二）

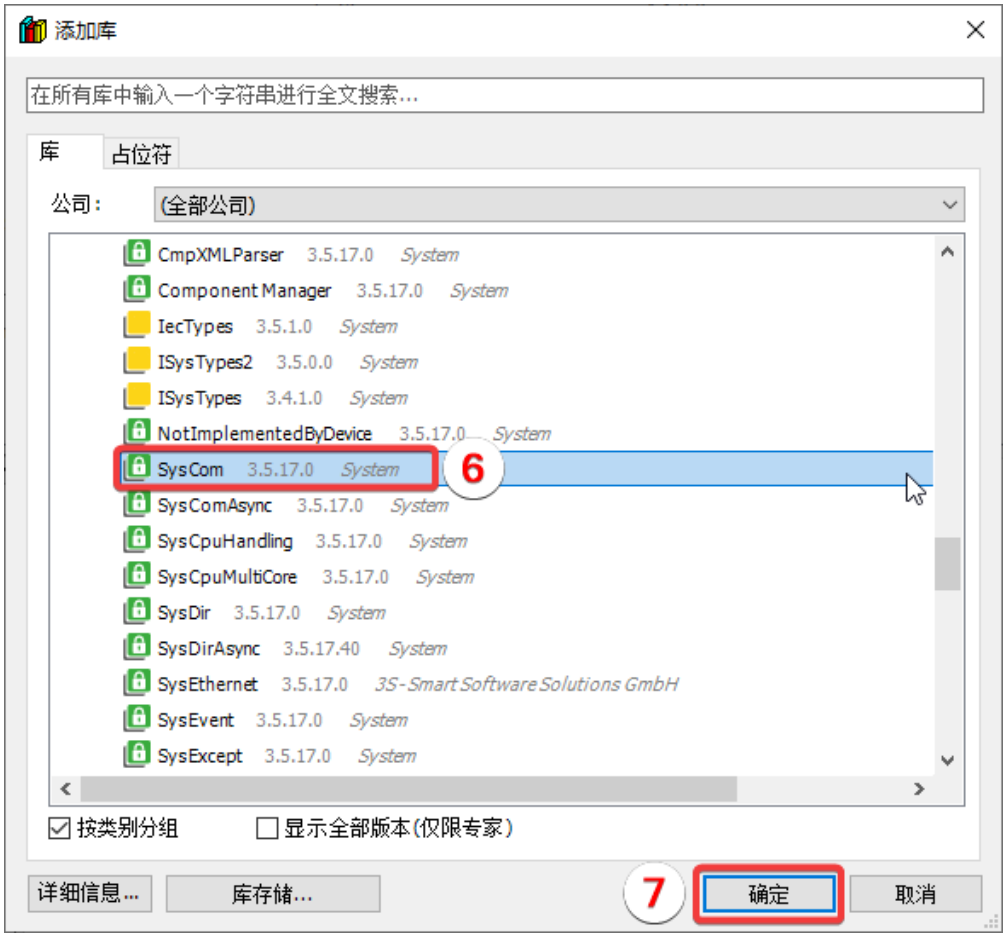


图 10-52 添加 SysCom 串口通信库（三）

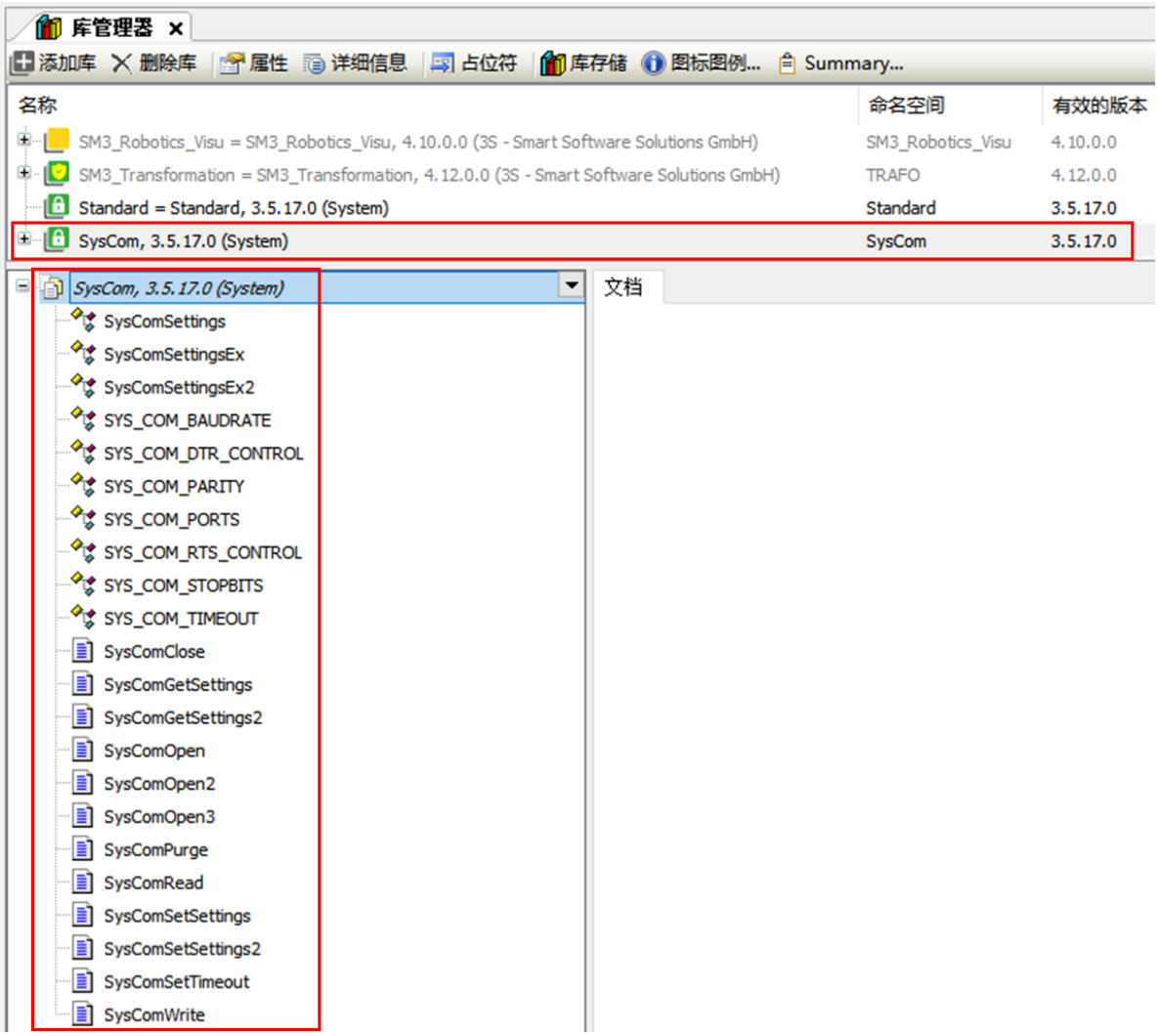


图 10-53 添加 SysCom 串口通信库（四）

图 10-54、图 10-55 提供了使用控制器 RS-232(COM5)端口，进行自由串口通信的示例程序。该示例程序实现的功能为：当控制器接收到数据后，将接收数据原样发出。

freeSerial_Demo

```
PROGRAM freeSerial_Demo
VAR
    comSetting   : SysCom.SysComSettings;
    comSettingEx : SysCom.SysComSettingsEx;
    hCom         : SysCom.SysTypes.RTS_IEC_HANDLE := SysCom.SysTypes.RTS_INVALID_HANDLE;
    openResult   : SysCom.SysTypes.RTS_IEC_RESULT;
    readResult   : SysCom.SysTypes.RTS_IEC_RESULT;
    writeResult  : SysCom.SysTypes.RTS_IEC_RESULT;
    comOpenStatus : BOOL := FALSE;  // TRUE: 串口打开成功

    baudrate    : DWORD := 19200;    // 波特率: 4800, 9600, 19200, 38400, 115200
    serialPort : INT := 4;           // 串口端口号: 0-disable, 1-COM1, 2-COM2, 3-COM3,
                                     //  4-COM4(SC301&SC302 RS485), 5-COM5(SC302 RS232), 6-COM6
    parity      : BYTE := 0;         // 奇偶校验: 0-无校验, 1-奇校验, 2-偶校验
    stopBits    : BYTE := 1;         // 停止位:  1-1个停止位, 2-1.5个停止位, 3-2个停止位

    recvCount : UDINT := 0;
    data : BYTE := 0;
    recvBuffer : ARRAY[0..127] OF BYTE;     // 根据实际需求调整大小
    recvPos : UDINT := 0;
    sendBuffer : ARRAY[0..127] OF BYTE;     // 根据实际需求调整大小
    sendLen : UDINT := 0;

    readDelay : TON;
END_VAR
```

图 10-54 串口自由通信示例程序：变量定义

```
//=============================================================================
// 初始化配置，打开串口
//=============================================================================
IF NOT comOpenStatus TRUE THEN
    comSetting.ulBaudrate SYS_BR_192 := baudrate 19200;
    comSetting.sPort SYS_COMPOR := serialPort 4;
    comSetting.byParity SYS_NOPARI := parity 0;
    comSetting.byStopBits SYS_ONESTO := stopBits 1;
    comSetting.ulBufferSize 256 := 256;
    comSetting.ulTimeout SYS_NOWAIT := SYS_NOWAIT 0;
    comSettingEx.bBinary TRUE := 1;
    comSettingEx.byByteSize 8 := 8;                    // 每字节数据位数: 4-8

    hCom 16#00000019 := SysComOpen2(pSettings   := ADR(comSetting),
                        pSettingsEx := ADR(comSettingEx),
                        pResult     := ADR(openResult 0));
    IF openResult 0 = CmpErrors.Errors.ERR_OK 0 AND hCom 16#00000019 > 0 THEN
        comOpenStatus TRUE := TRUE;
    END_IF
END_IF
//=============================================================================
// 检查是否接收到数据
//=============================================================================
IF comOpenStatus TRUE AND hCom 16#00000019 > 0 THEN
    IF recvPos 0 > 0 THEN
        SysMemSet(pDest := ADR(recvBuffer), udiValue := 0, udiCount := recvPos 0);
        recvPos 0 := 0;
    END_IF
    readDelay(IN TRUE := TRUE, PT T#5ms := T#5MS, Q => , ET => );
    IF readDelay.Q FALSE THEN
        readDelay(IN TRUE := FALSE);
        // 循环读取数据，直至读空或接收buffer满
        REPEAT
            recvCount 0 := SysComRead(hCom := hCom 16#00000019,
                        pbyBuffer:= ADR(data 0),
                        ulSize:= 1,
                        ulTimeout:= 0,
                        pResult:= ADR(readResult 0));
                (* 读取1字节数据，并存放至接收内存 *)
                IF recvCount 0 <> 0 THEN
                    recvBuffer[recvPos 0] 0 := data 0;
                    recvPos 0 := recvPos 0 + recvCount 0;
                END_IF
            UNTIL  (recvPos 0 = SIZEOF(recvBuffer))  OR recvCount 0 = 0
        END_REPEAT

        IF (recvPos 0 > 0) THEN
            SysMemCpy(pDest := ADR(sendBuffer), pSrc := ADR(recvBuffer), udiCount := recvPos 0);
            sendLen 0 := recvPos 0;
        END_IF
    END_IF
ELSE
    readDelay(IN FALSE := FALSE);
END_IF
//=============================================================================
// 如果接收数据不为空，将接收数据原样发送出去
//=============================================================================
IF comOpenStatus TRUE AND (hCom 16#00000019 > 0) AND (sendLen 0 > 0) THEN
    SysComWrite(hCom := hCom 16#00000019,
                pbyBuffer := ADR(sendBuffer),
                ulSize := sendLen 0,
                ulTimeout := 0,
                pResult := ADR(writeResult 0));
    IF writeResult 0 = CmpErrors.Errors.ERR_OK 0 THEN
        SysMemSet(pDest := ADR(sendBuffer), udiValue := 0, udiCount := sendLen 0);
        sendLen 0 := 0;
    END_IF
END_IFRETURN
```

图 10-55 串口自由通信示例程序：功能实现

11. 掉电保持

SC301 内置超级电容，无需外接 UPS 电源即可支持意外掉电数据保持。掉电保持型变量在 PLC 断电、复位或程序下载后仍能保留其值，最大支持 256KBytes 数据量。如果不需要掉电保持功能，则无需关心该功能。

【注意】：由于超级电容需要先充电后使用，如果控制器新上电不足 30S，由于充电时间不足，掉电保持数据可能失败。

在 COSESYS 软件中，掉电保持型变量的定义和声明需要使用特定关键字才有效，根据掉电保持型变量定义位置的不同，有 2 种不同的方法。

11.1 在全局变量表 GVL 中定义

掉电保持型变量在全局变量表 GVL 中定义，然后在掉电保持型变量表 PersistentVars 文件中声明并添加所有实例路径，具体步骤如下：

（1）创建全局变量表文件

如下图 11-1，在 CODESYS 项目中的工程窗口，右键单击“Application”节点，单击选择“添加对象”，然后在下拉列表中选择“全局变量列表…”，会创建 GVL 文件，用于定义全局变量。

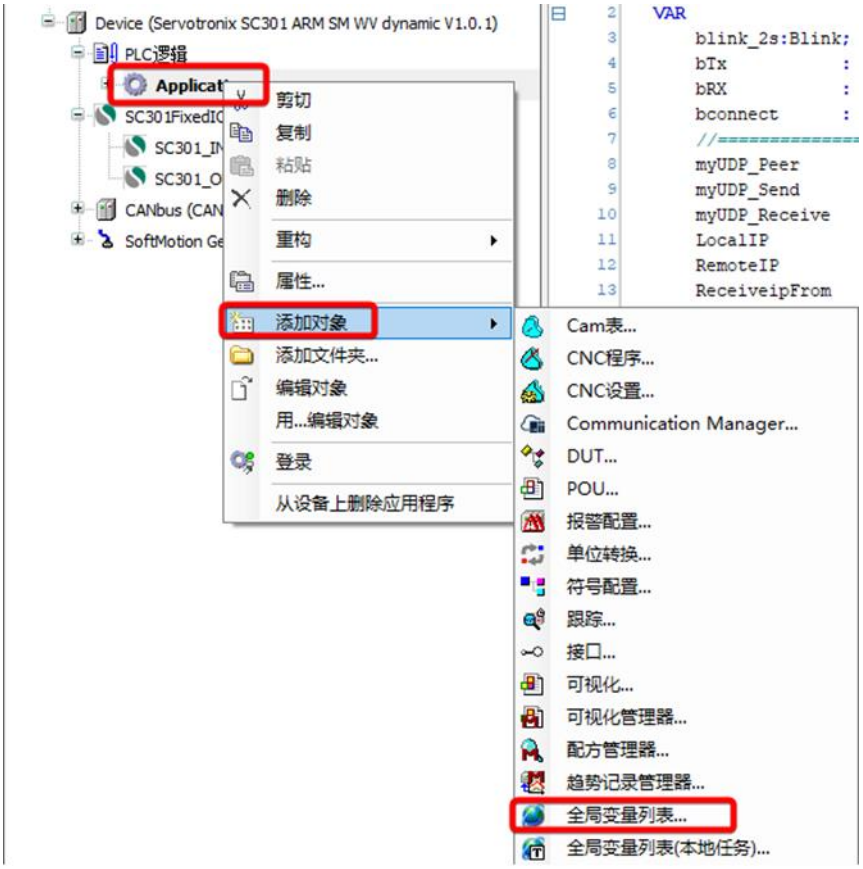


图 11-1 创建全局变量表

如下图 11-2，打开 GVL 文件，在该文件中进行全局变量定义。图中使用 VAR_GLOBAL 和 END_VAR 关键字定义的变量属于通用全局变量，掉电不保持。在关键字 **VAR_GLOBAL PERSISTENT RETAIN** 和 **END_VAR** 之间定义的变量才属于掉电保持型全局变量，例：

```
VAR_GLOBAL PERSISTENT RETAIN
test5: BOOL;  //BOOL 型变量
test6: INT;   //INT 型变量
test7: REAL;  //REAL 型变量
test8: ARRAY[1…10] OF REAL;  //REAL 型数组
END_VAR
```

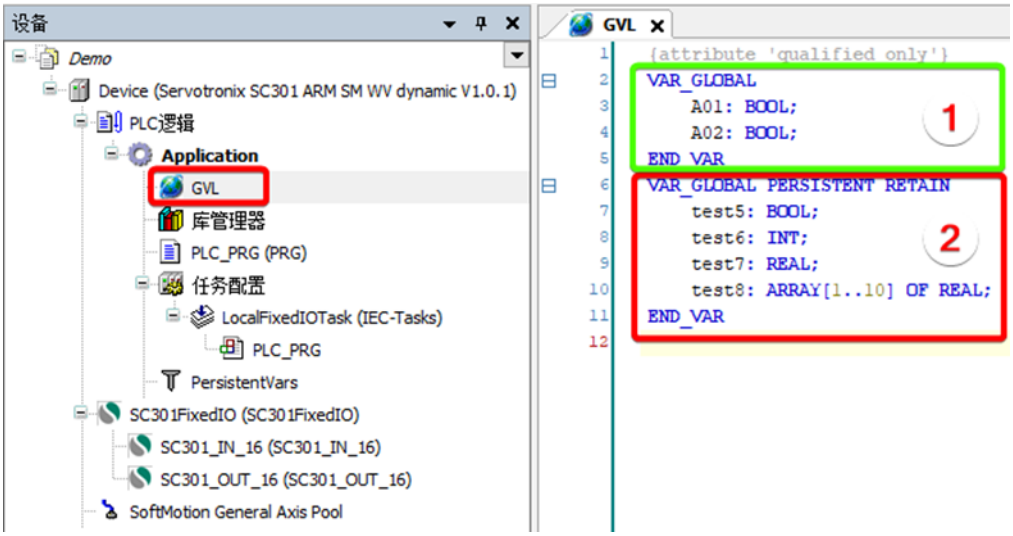


图 11-2 全局变量定义

（2）创建掉电保持型变量声明文件

如下图 11-3，在 CODESYS 项目中的工程窗口，右键单击“Application”节点，单击选择“添加对象”，然后在下拉列表中选择“掉电保持型变量…”，会创建 PersistentVars 文件，用于声明哪些变量需要在断电后保持其值。

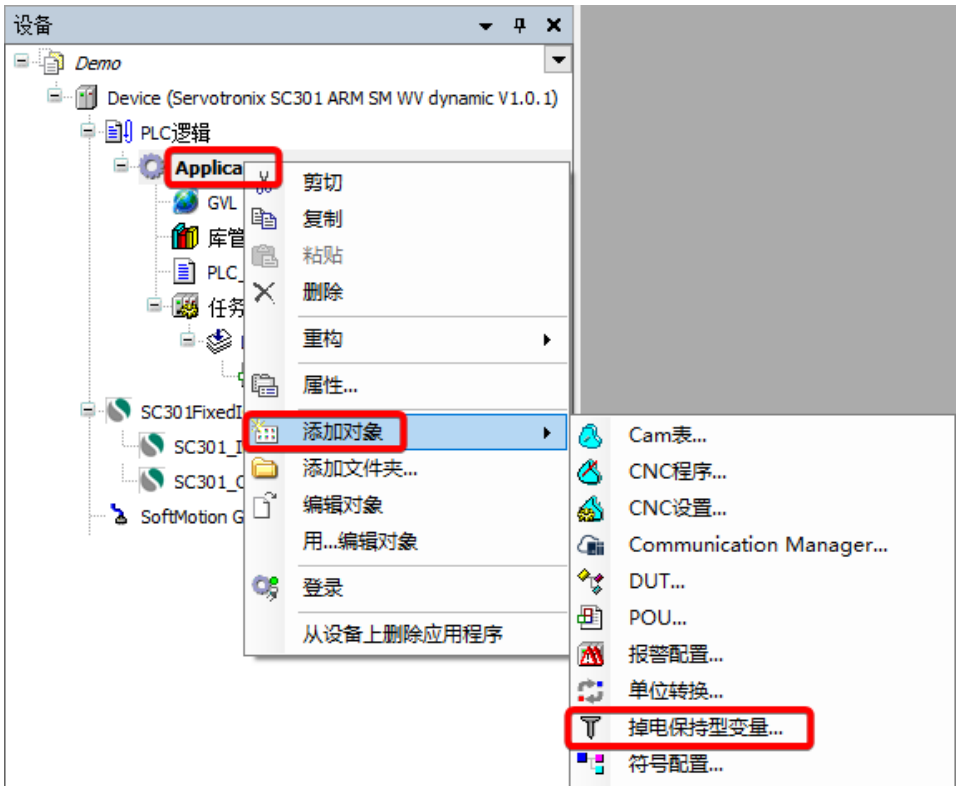


图 11-3 创建 PersistentVars 文件

（3）工程编译并添加实例化路径

打开 PersistentVars 文件，会看到在文件的上部，有一对关键字 **VAR_GLOBAL PERSISTENT RETAIN** 和 **END_VAR**，这两个关键字就是用于对在 GVL 中定义的掉电保持型变量进行声明使用的，是软件自动创建的。

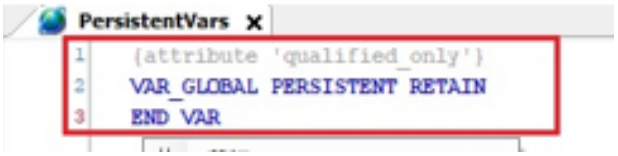


图 11-4 变量声明关键字

由于掉电保持变量的定义和声明不在同一个文件，所以声明需要通过“添加实例路径”实现，添加实例路径前需要先进行工程编译，如果没有经过工程编译，会弹出如下图 11-5 所示的错误提示。

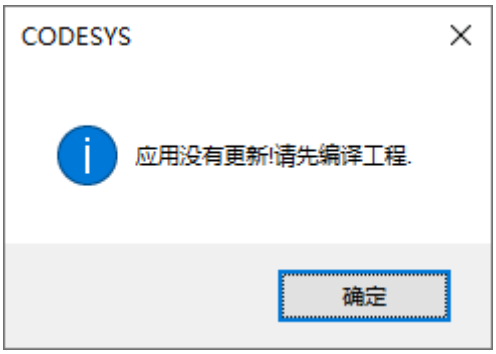


图 11-5 提示先编译工程

接下来进行工程编译，编译的作用是自动找到 GVL 中定义的掉电保持变量。编译通过后，如下图 11-6，在 PersistentVars 文件页面编辑区任意位置，右键选择“添加所有实例路径”。

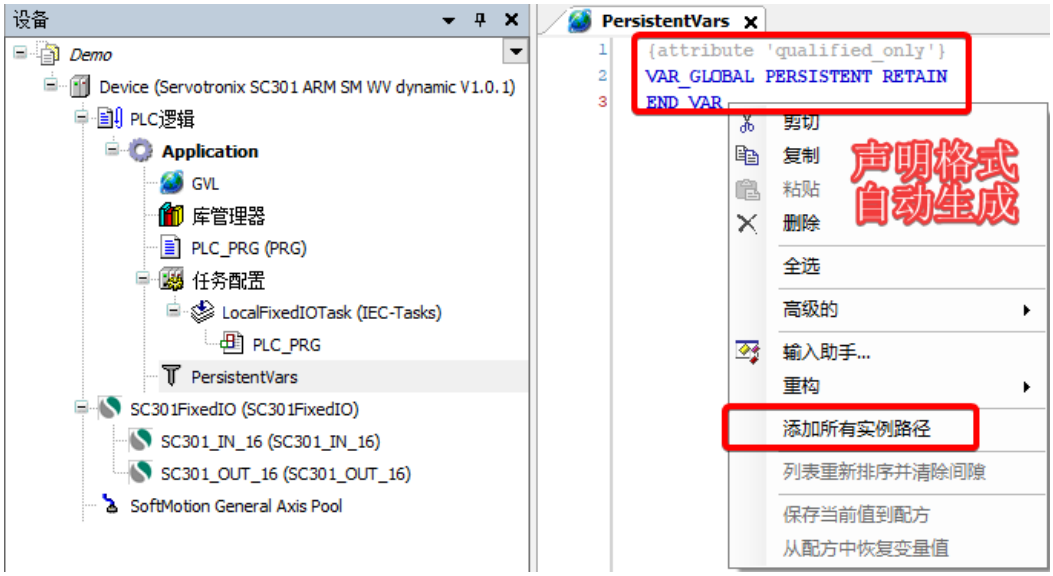


图 11-6 添加所有实例路径

之后，软件会自动将在全局变量表 GVL 中定义的且被 POU 引用的掉电保持变量添加到 PersistentVars 文件中的掉电保持变量声明区，完成后的界面如下图 11-7，图中红色方框内即为完成实例路径的变量声明。

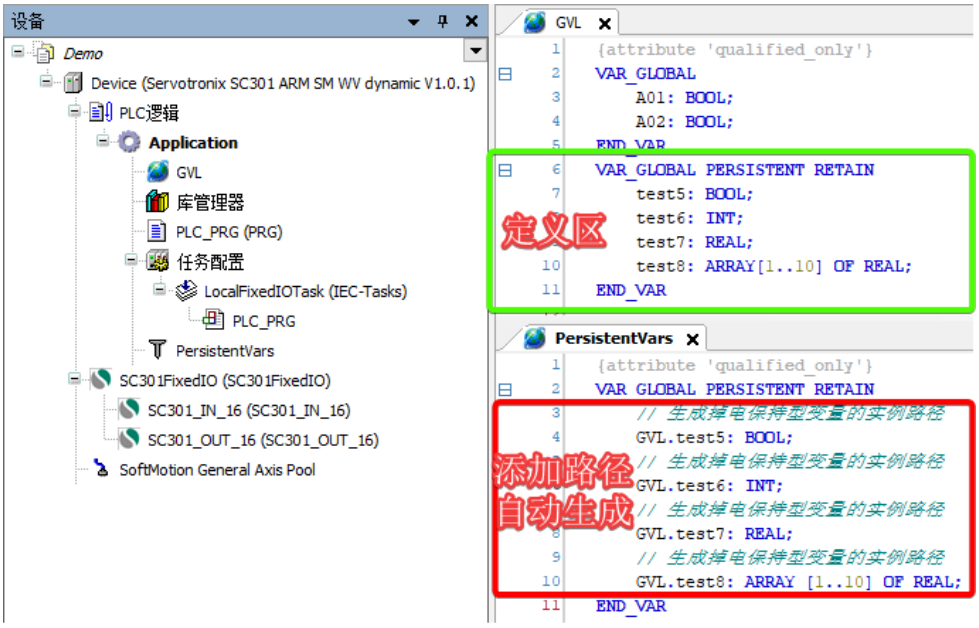


图 11-7 生成掉电保持型变量的实例路径

当有多个掉电保持变量分散定义在多个不同的全局变量表中时，通过一次添加实例路径的操作，就可完成所有的掉电保持变量声明。

【注意】：在添加实例路径的过程中，仅添加全局变量表中定义的且被 POU 引用的掉电保持变量。如果一个全局变量表中定义的所有掉电保持的变量，均没有任何 POU 引用，则添加所有实例路径会失效，也无法添加未调用的变量，这样可以减少系统资源消耗。此类定义但未被 POU 引用的掉电保持变量在后续需要使用时需要重新点击“添加所有实例路径”动作。

11.2 在掉电保持变量表 PersistentVars 文件中定义

掉电保持型变量在掉电保持变量表 PersistentVars 文件中定义，无需再添加所有实例路径，具体步骤如下：

（1）创建掉电保持型变量声明文件

如下图 11-8，在 CODESYS 项目中的工程窗口，右键单击“Application”节点，单击选择“添加对象”，然后在下拉列表中选择“掉电保持型变量…”，会创建 PersistentVars 文件，用于定义哪些变量需要在断电后保持其值。

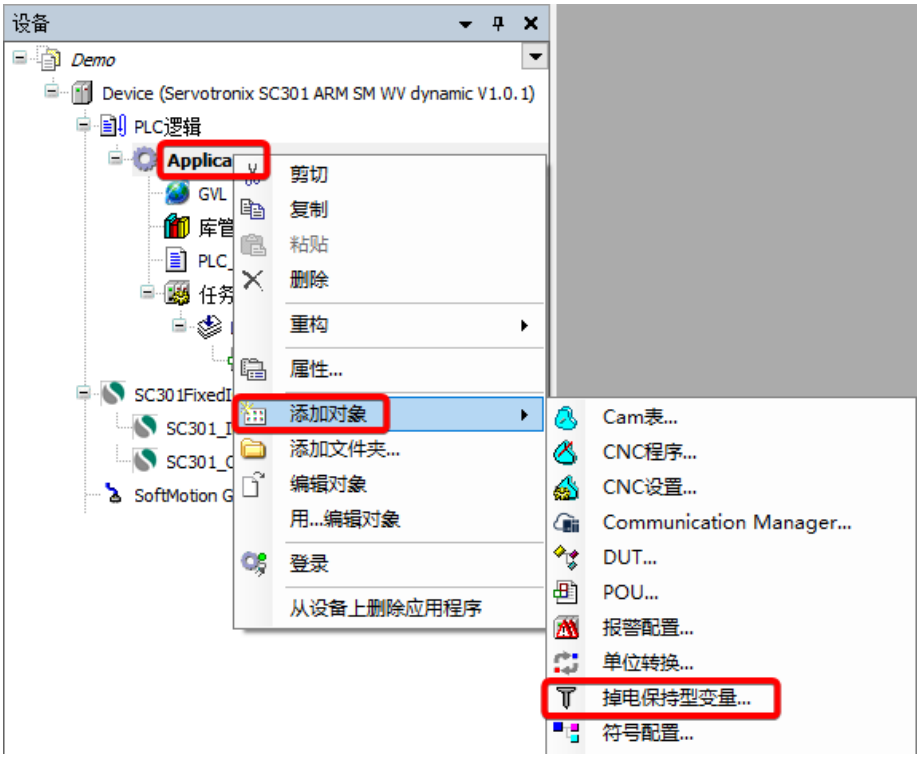


图 11-8 创建 PersistentVars 文件

如下图 11-9，打开 PersistentVars 文件，会看到在文件的上部，有一对关键字 **VAR_GLOBAL PERSISTENT RETAIN** 和 **END_VAR**，这两个关键字就是用于对掉电保持型变量进行定义使用的，是软件自动创建的。

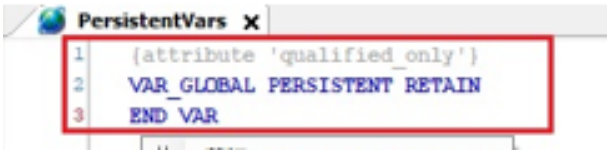


图 11-9 自动创建的掉电保持变量定义关键字

如下图 11-10，在关键字 **VAR_GLOBAL PERSISTENT RETAIN** 和 **END_VAR** 之间定义的所有可支持的类型数据类型如单个变量、数组、结构体等都具备掉电保持功能，由于掉电保持变量的定义和声明是在同一个文件中，因此无需再添加所有实例路径。定义完成以后，就可以在 POU 中被引用了。

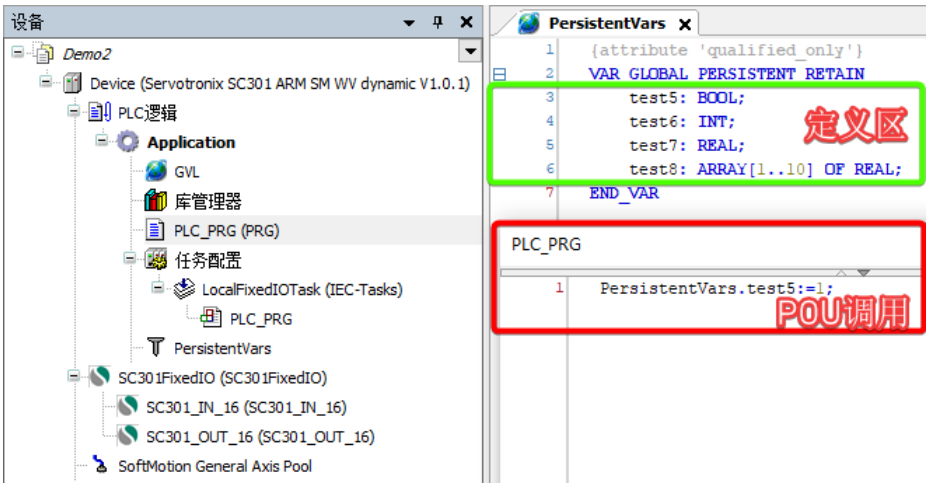


图 11-10 在 PersistentVars 文件中定义和声明

综上所述，上述两种方法都可以使用，但是存在以下区别，请根据需要选择：

（1）掉电保持变量在全局变量表 GVL 中定义（也可以分散定义在多个全局变量表中）：便于集中管理全局变量，但需要在 PersistentVars 文件中添加实例化路径进行声明。

（2）掉电保持变量在掉电保持变量表 PersistentVars 文件中定义和声明，便于单独管理，无需再添加实例化路径，但与其它全局变量不在同一个文件。

12. 配置网络及日期时间信息

原生的 CODESYS 编程软件，不支持直接在软件界面中对控制器的网络接口参数和日期时间信息进行读写。为此高创传动提供了 2 种访问方式：（1）通过配套工具软件 ServotronixControllerToolkit 进行操作；（2）在 CODESYS 软件中通过自定义 PLC 指令进行操作。对于两种访问方式，推荐使用更简单易用的 ServotronixControllerToolkit。

12.1 使用工具软件 ServotronixControllerToolkit

12.1.1 工具软件的获取与安装

ServotronixControllerToolkit 可在高创传动官网 www.servotronix.cn 首页，依次单击“资料下载中心”->“运动控制器&I/O 模块”->“控制器编程与工具软件”->“控制器工具软件”，在弹出的页面中点击“下载”按钮进行下载，并保存在合适位置。也可以联系高创销售或者技术支持人员获取。

【注意】：ServotronixControllerToolkit 软件目前有 V1.7、V1.8、V1.9 三个版本：V1.8 版本在 V1.7 版本基础上，增加了读写网关地址（Gateway）的功能，V1.9 版本在 V1.8 版本基础上，增加了对控制器 softMC803 和 TC304 的支持。如果不需要修改网关地址，则三个版本都可以使用，如果需要修改默认网关地址，则可以使用 V1.8 或 V1.9 版本。

12.1.2 工具软件介绍

ServotronixControllerToolkit 作为 CODESYS 软件的必要补充，支持目前高创所有在售的 CODESYS 控制器型号，且具有一键切换中英文功能。该软件主要实现以下 3 个功能：

（1）控制器网络接口参数（IP 地址、子网掩码、网关地址）的读取和写入；

（2）通过读写 RTC 实现控制器日期时间信息的读取和同步校正；

（3）查看控制器当前固件版本信息。

12.1.3 工具软件使用前必读

SC301 一共有 3 个网络通信接口，主要信息如下表 12-1 所示。更详细的信息请参考“**4.21 PORT1 EtherNet、4.20 PORT2 EtherNet、4.19 PORT3 EtherCAT**”章节内容。

表 12-1 SC301 网络通信接口信息汇总

网口	用途	默认 IP 地址	默认子网掩码	默认网关地址	网卡编号
PORT1	Ethernet	90.0.0.1	255.255.255.0	无	eth1
PORT2	Ethernet	192.168.0.1	255.255.255.0	192.168.0.171	eth0
PORT3	EtherCAT	192.168.39.220	255.255.255.0	无	eth2

（1）PORT1 Ethernet 接口的 IP 地址和子网掩码可以按需修改，但不支持配置网关地址，所以不支持跨网段通信。

（2）PORT2 Ethernet 接口的 IP 地址、子网掩码、网关地址均可以按需修改，所以支持跨网段通信。SC301 控制器固件 SC301_V1.0.10_20251121 及以上版本支持修改网关地址。如固件版本低于 SC301_V1.0.10_20251121 且有修改网关地址需求，需先进行固件升级，固件版本信息在工具软件可以查询。

（3）PORT3 EtherCAT 接口的 IP 地址和子网掩码均禁止修改，也无需修改。

（4）参数需要逐个修改，多个参数一次性修改后可以重启后同时生效；同一个参数可以多次修改，以最后一次修改为准，重启后生效。

（5）无论是使用 PORT1、PORT2、PORT3 的默认 IP 地址，还是使用自定义的 IP 地址，计算机所使用网口的 IP 地址必须与所连接 SC301 的网口的 IP 地址处于同一个网段。

12.1.4 建立与控制器的连接

在通过 ServotronixControllerToolkit 与 SC301 连接之前，需要知道 3 个网口（含 PORT3 EtherCAT）中至少 1 个网口的 IP 地址。通常情况下，PORT3 EtherCAT 网口的 IP 地址会一直维持出厂默认值，不会改变。

最糟糕的情况，当三个网口的网络通信参数均已遗失，可以参考“**4.5 RESET 按键**”章节内容，将所有网口的网络通信参数恢复到出厂默认值。

下面以 V1.9 版本软件为例，介绍工具软件与 SC301 的 PORT2 网口连接的过程。

（1）修改计算机 IP 地址。修改计算机所选网口的 IP 地址，使之与 SC301 的 PORT2 网口的 IP 地址处于同一网段。

（2）SC301 上电并连接网线。先将 SC301 上电，当绿色 RUN 运行指示灯常亮时，表示 SC301 已正常运行。然后通过网线将计算机和 SC301 的 PORT2 网口连接，当 PORT2 网口的橙色指示灯亮起时，表明网络物理连接已经成功。

（3）启动软件。运行后的 V1.9 版本软件中、英文界面分别如下图 12-1 和图 12-2。在软件界面的下方语言（Language）处根据需要选择中文或者 English。

图 12-1 高创控制器工具集软件中文界面

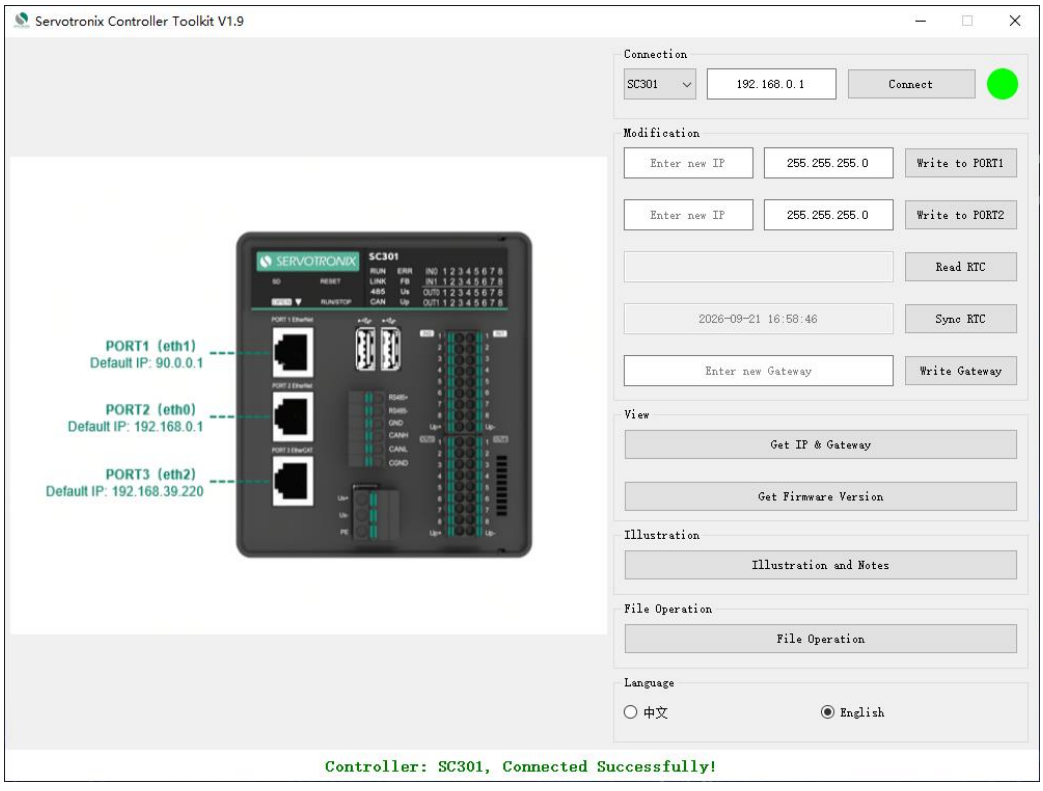


图 12-2 ServotronixControllerToolkit Software English Interface

（4）与控制器建立连接。如下图 12-3，在软件界面的“连接”栏，从型号下拉框中选择控制器型号“SC301”，并在 IP 地址输入框正确填写当前所选的 SC301 PORT2 网口的 IP 地址（本范例中输入默认的 192.168.0.1），然后点击“连接”按钮。待连接成功之后，右侧圆形的连接状态指示灯会变为绿色，同时界面底部还会有文字提示：“控制器：SC301，连接成功！”。

图 12-3 建立连接

如下图 12-4，如果连接失败，右侧圆形的连接状态指示灯会变为红色，同时界面底部还会有文字提示：“检测到控制器已断开，请重新连接控制器！”。此时需要检查网线是否连接牢固、计算机 IP 地址设置是否正确、控制器型号是否选对、所连接控制器网口的 IP 地址输入是否正确等，然后重新连接。

图 12-4 连接断开

12.1.5 读取网络接口参数信息

如下图 12-5 所示，无论使用 SC301 哪个网口与工具软件连接成功，点击软件中“**获取 IP 地址 & Gateway**”按钮会弹框显示所有网口的参数信息，如下图 12-6。如果固件版本不支持网关，则会进行提示，如图 12-7 所示。

【注意】：弹框显示的信息是控制器当前的有效信息，如果上电后有过修改但没有重启过，则不会显示。

图 12-5 读取所有网络接口参数信息

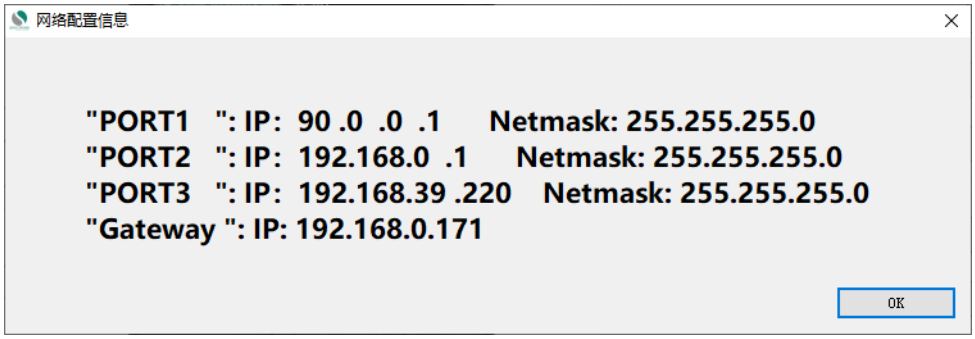


图 12-6 所有网口参数信息显示(固件支持网关)

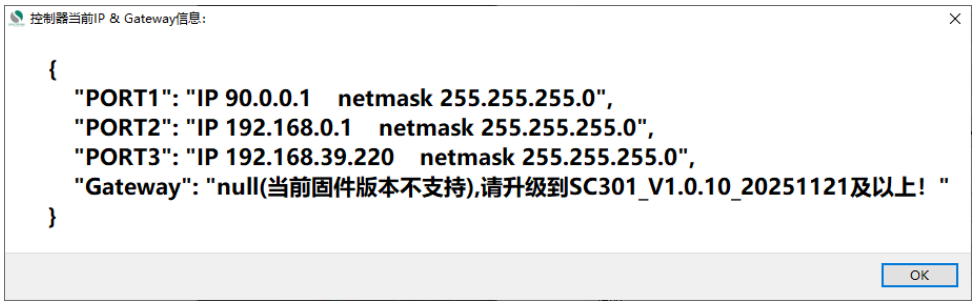


图 12-7 所有网口参数信息显示(固件不支持网关)

12.1.6 修改网络接口参数信息

（1）修改网络参数信息。如下图 12-8 所示，连接成功后，在 PORT1 或者 PORT2 的新 IP 地址输入框输入新的 IP 地址，点击“**写入 PORT1**”按钮会更新 PORT1 的 IP 及子网掩码地址；同理点击“**写入 PORT2**”按钮会更新 PORT2 的 IP 及子网掩码地址；点击“**写入 Gateway**”按钮会更新网关 Gateway 的地址。通常子网掩码无需修改。

【注意】：每次点击“写入”操作只能修改一个参数，同一个参数可以多次修改，以最后一次修改为准，重启后生效。

图 12-8 修改网络参数操作

在点击完“写入”操作按钮以后，在软件界面的下方会有相应的操作提示，如下图 12-9 所示，请按照提示信息进行操作。

（2）**重启控制器**。将控制器进行断电重启。【**注意**】：**断电重启时，需要等待SC301的指示灯全部熄灭之后再上电，否则可能导致断电重启失败。**

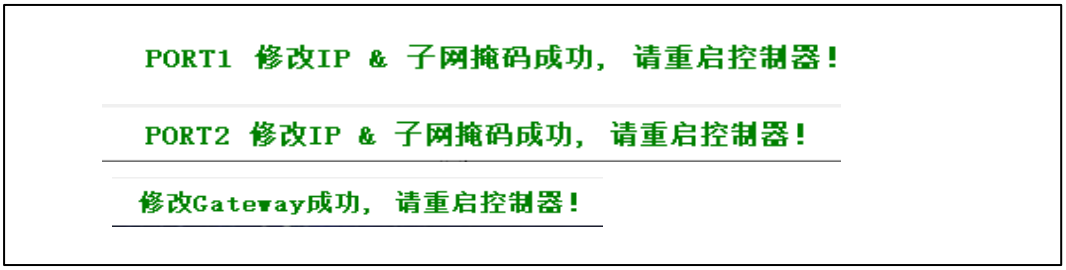


图 12-9 修改结果提示信息

（3）**确认是否修改成功**。重启之后，网络参数信息将会生效，可以再次连接通过读取确认是否修改成功。如果与计算机连接的 SC301 的网口参数没有修改，通过“**获取 IP & Gateway**”操作可查看其它网口参数是否修改成功。如果与计算机连接的 SC301 的网口 IP 地址做了修改，则需要先修改计算机网口的 IP 地址与 SC301 所连接网口的新 IP 地址处于同一个网段，然后连接控制器，再通过“**获取 IP 地址 & Gateway**”操作可查看网口参数是否修改成功。

如下图 12-10，在处于同一个网段的前提下，也可以使用 ping IP 地址的方式来快速判断网络是否具备通信成功的基础。

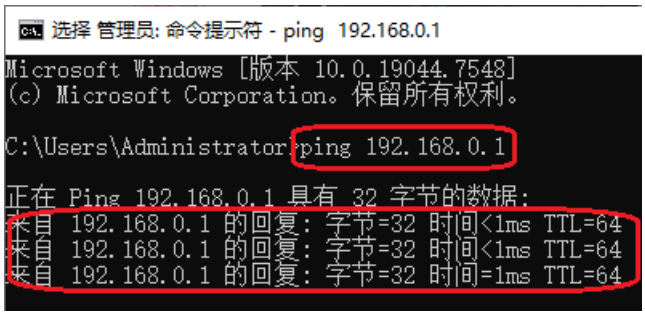


图 12-10 ping 命令确认控制器物理连接

12.1.7 IP 地址遗忘后的操作

（1）当 PORT1 或者 PORT2 网口的 IP 地址，有一个遗失后，可以通过已知的 IP 地址建立连接，然后点击“**获取 IP 地址 & Gateway**”按钮，获取另外一个遗

失的网口 IP 地址。

（2）通常，PORT3 EtherCAT 接口的 IP 地址和子网掩码均禁止修改，当 PORT1 和 PORT2 网口的 IP 地址全部遗失后，可以使用 PORT3 EtherCAT 接口建立连接后，使用“**获取 IP 地址 & Gateway**”按钮获取遗失的所有地址信息。

（3）当三个网口的网络通信参数均已遗失，可以参考“**4.5 RESET 按键**”章节内容，将所有网口的网络通信参数恢复到出厂默认值。

12.1.8 日期时间信息的读写

建立连接后，还可以通过 ServotronixControllerToolkit 软件对控制器的 RTC 进行日期时间信息的读取和校正。如下图 12-11，点击“读 RTC”按钮，可以将控制器当前的日期信息读取上来并显示；点击“同步 RTC”可以将当前计算机的日期时间信息写入控制器的 RTC，实现校正。

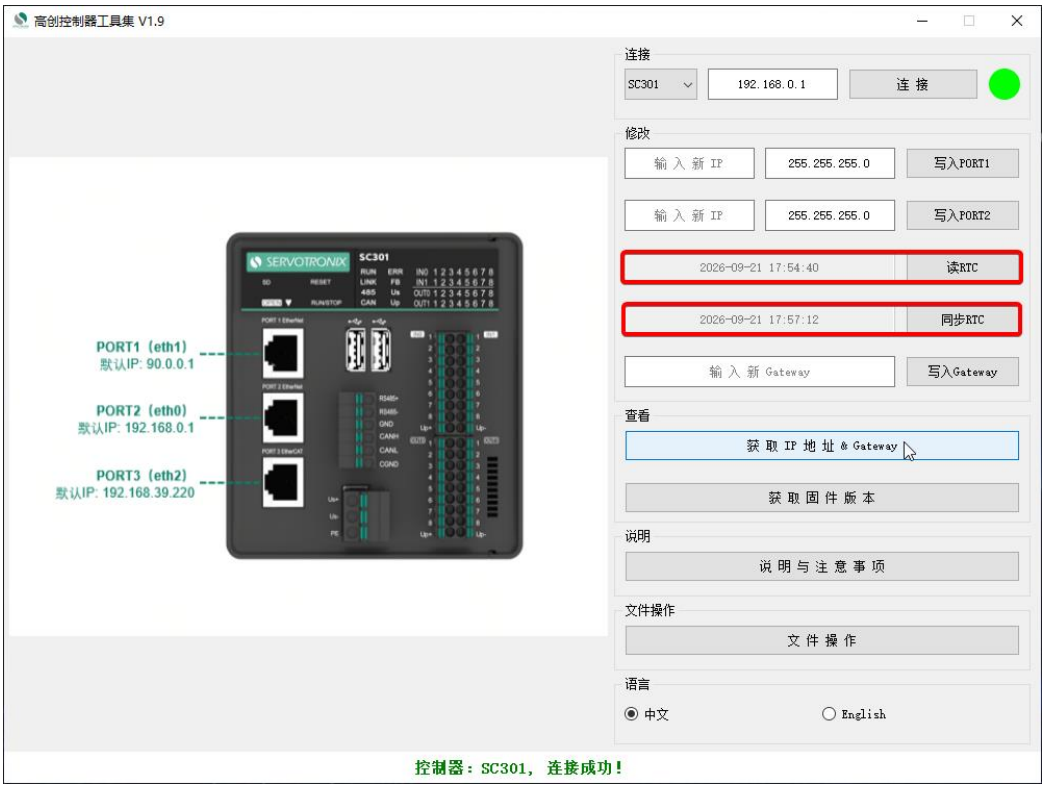


图 12-11 读写日期时间信息

12.1.9 获取控制器固件版本信息

建立连接后，还可以通过 ServotronixControllerToolkit 软件对控制器的固件版本信息读取。如下图 12-12，点击“获取固件版本”按钮，可以将控制器当前的固件版本信息读取上来并显示，如图 12-13。

图 12-12 获取固件版本

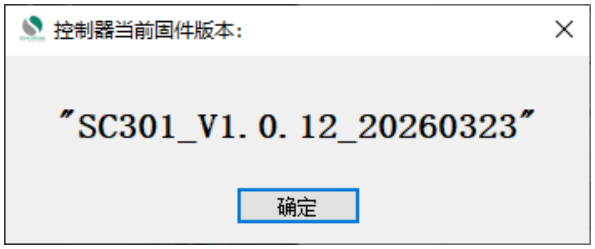


图 12-13 当前固件版本信息

12.2 使用自定义 PLC 指令

使用自定义 PLC 指令也可以对控制器的网络接口参数和日期时间信息进行读写。

（1）登录控制器。打开 CODESYS 软件，将其与 SC301 连接，并且登录到 SC301。

（2）打开 PLC 指令输入界面。如下图 12-14，双击工程窗口的“Device”，然后在出现的列表中选中“PLC 指令”，会出现指令输入界面。可以直接在指令输入框手动输入指令；也可以单击右下角的省略号在指令列表选中并插入指令，被插入的指令会出现在指令输入框中，如图 12-15。

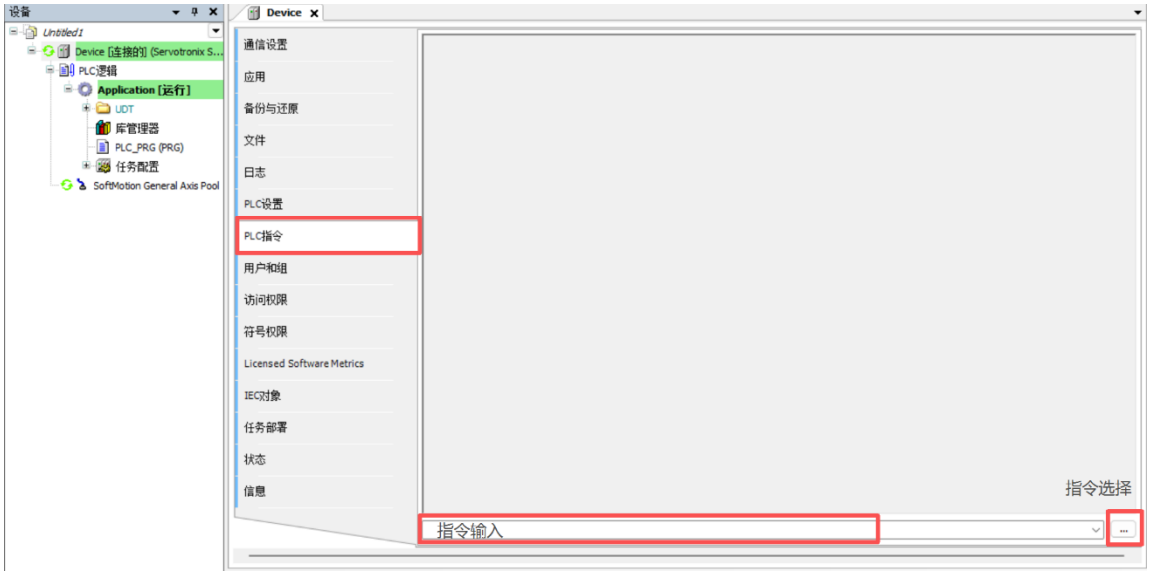


图 12-14 PLC 指令输入界面

图 12-15 插入指令

12.2.1 读取网络接口参数信息

读取网络接口通信参数的指令一共有两个：

GetIPAddress：读取某一个网口当前的 IP 地址及子网掩码。由于 SC301 有 3 个网口，故在指令输入时需要指定网卡编号，具体每个网口的网卡编号请查阅“**4.21 PORT1 EtherNet、4.20 PORT2 EtherNet、4.19 PORT3 EtherCAT**”章节内容，或者参考表 12-1。以 PORT2 为例，其网卡编号为 eth0，故读取该网口 IP 地址的指令格式为：**GetIPAddress eth0**。指令执行后，会在界面上返回当前网口的 IP 地址及子网掩码两个内容，如下图 12-16。

其余网口操作类似，只需要改变网卡编号参数。

图 12-16 GetIPAddress

GetGateway：获取 PORT2 EtherNet 的网关地址。由于仅有 PORT2 EtherNet 网口支持跨网段通信，故在读取时无需指定网卡编号，其指令格式为：**GetGateway**。指令执行后，会在界面上返回当前的网关地址信息，如下图 12-17。

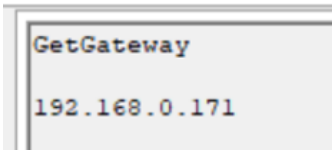


图 12-17 读取网关信息

12.2.2 修改网络接口参数信息

读取网络接口通信参数的指令一共有两个：

SetIPAddress：设置某一个网口的新 IP 地址和子网掩码。通常子网掩码无需修改，使用默认值 255.255.255.0 即可。由于 SC301 有 3 个网口，故在指令输入时需要指定网卡编号以及要设置的新 IP 地址及新子网掩码，具体每个网口的网卡编号请查阅“**4.21 PORT1 EtherNet、4.20 PORT2 EtherNet、4.19 PORT3 EtherCAT**”章节内容，或者参考表 12-1。如果子网掩码不修改就输入当前的子网掩码（其值可以通过指令 GetIPAddress 获取）。以 PORT2 为例，其网卡编号为 eth0，故设置该网口新 IP 地址及子网掩码的指令格式为：**SetIPAddress eth0 新 IP 地址 新子网掩码**，如设置新的 IP 地址为 192.168.0.2，子网掩码维持不变，则指令格式为：SetIPAddress eth0 192.168.0.2 255.255.255.0。设置成功后，会在界面上返回“OK”，如下图 12-18。

```
GetIPAddress eth0

192.168.0.1 255.255.255.0


----
SetIPAddress eth0 192.168.0.2 255.255.255.0

OK
```

图 12-18 先读取后设置新 IP 地址

SetGateway：设置 PORT2 EtherNet 新的网关地址。由于仅有 PORT2 EtherNet 网口支持跨网段通信，故在设置时无需指定网卡编号，其指令格式为：**SetGateway 新网关地址**。如设置新的网关地址为 192.168.0.123，则指令格式为：**SetGateway** 192.168.0.123。设置成功后，会在界面上返回“OK”，如下图 12-19。

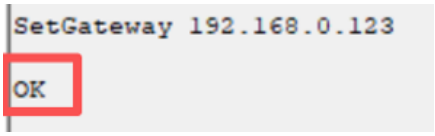


图 12-19 设置新的网关地址

【注意】：IP 地址、子网掩码及网关地址，任何一项修改后都必须要断电重启后才能生效。可以通过对应的 Get 指令来确认是否修改成功。如下图 12-20。

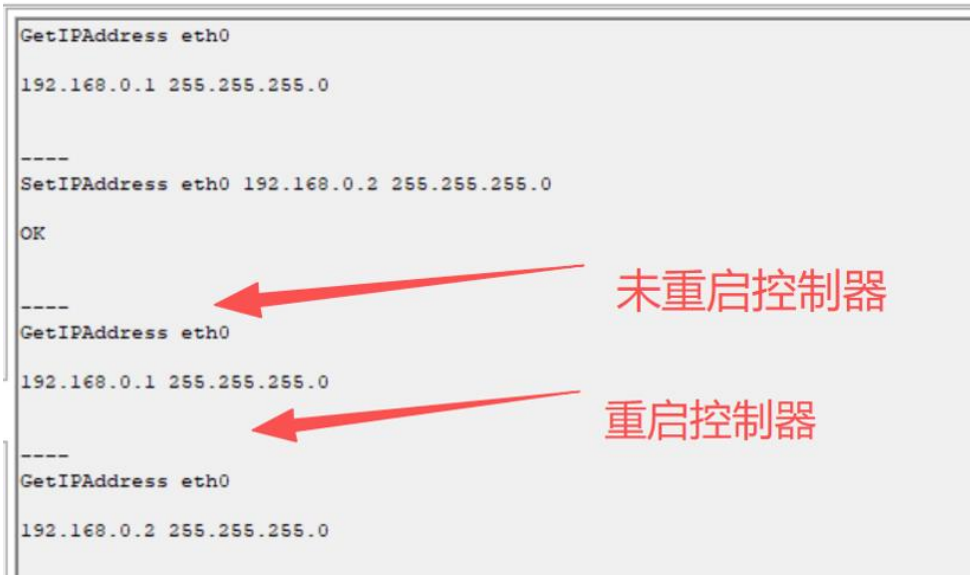


图 12-20 断电重启生效

12.2.3 读取日期时间信息

获取控制器当前日期信息时间的指令格式为 **GetDateTime**，如下图 12-21 所示，在指令输入框中输入 GetDateTime，则返回控制器当前日期信息。

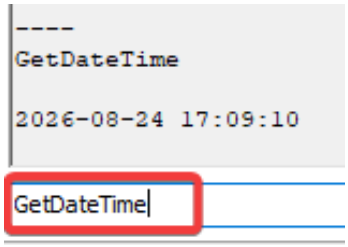


图 12-21 获取时间指令

12.2.4 同步日期时间信息

同步控制器当前日期信息时间的指令格式为 **SetDateTime**，如下图 12-22 所示，在指令输入框中输入 SetDateTime 2026-08-24 18:09:05，则将输入框中日期时间信息写入到控制器。

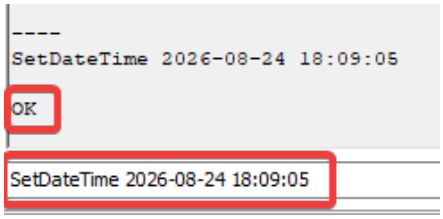


图 12-22 同步日期时间指令

【注意】：由于手动校准存在延时和滞后，故对时间精确度要求不高的情况下可以使用；当对时间精度要求较高时，推荐使用工具软件 ServotronixControllerToolkit 操作。

12.2.5 获取控制器固件版本信息

获取控制器固件版本信息的指令格式为 **ReadOSVersion**，如下图 12-23 所示，在指令输入框中输入 ReadOSVersion，则返回控制器当前固件版本信息。

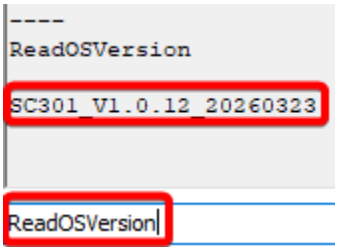


图 12-23 获取控制器固件版本指令

13. WebHMI

SC301 控制器提供了 WebHMI（WebVISU）的功能，因此可以通过 PORT1 Ethernet 和 PORT2 Ethernet 接口进行访问。由于 POR2 相对 POR1 具有更高的通信速率，因此更推荐使用 PORT2 访问。

打开计算机浏览器，在地址栏中按照如下格式输入即可访问：http://连接网口 IP 地址:8080/webvisu.htm。

如果 SC301 控制器的 PORT1 和 PORT2 的 IP 地址均为出厂模式值，则访问地址分别是：

PORT1：http://90.0.0.1:8080/webvisu.htm

PORT2：http://192.168.0.1:8080/webvisu.htm

如果修改了 IP 地址，则按照修改后的 IP 地址进行访问。

下图 13-1 是 WebHMI 画面的示例。

图 13-1 WebHMI 画面示例

14. PLCHandler 通信

14.1 PLCHandler 概述

PLCHandler 是一个 C++类，它提供了在客户端（例如可视化）和符合 CODESYS 标准的 PLC（控制器）之间进行通信的高级服务。PLCHandler 封装了完整的低层通信协议，并提供了 API 接口。该 API 接口提供对所有可用功能和服务的访问，其通信模型如下图 14-1 所示，图中可以看出，CODESYS 的 OPC-UA 通信也是基于 PLCHandler 构建的。

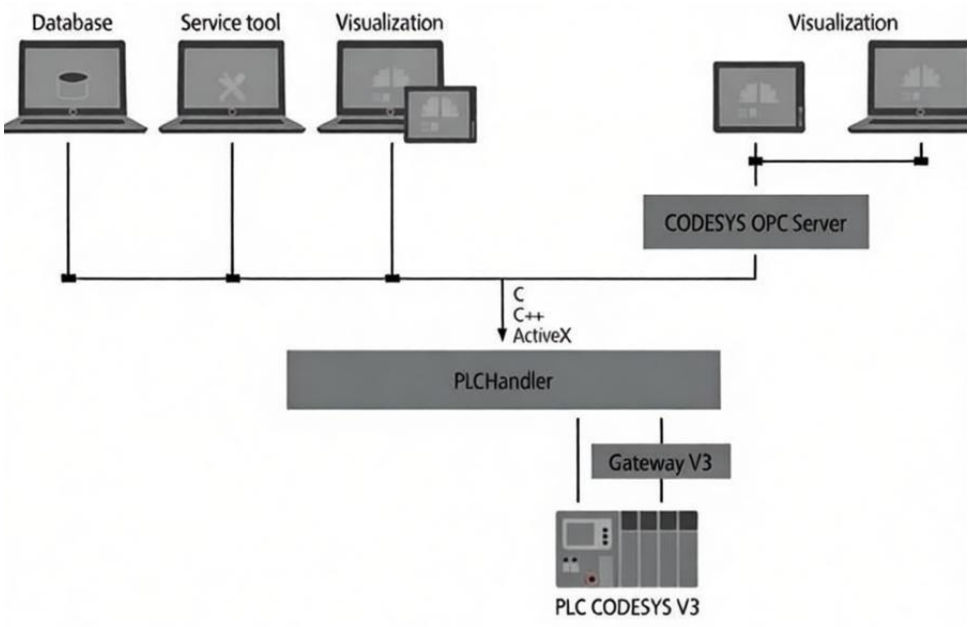


图 14-1 PLCHandler 通信模型

PLCHandler 提供以下功能和服务：

（1）建立或终止与 PLC 的通信。

（2）读取 PLC 的变量列表。

（3）循环读取 PLC 变量。

（4）同步读取 PLC 变量值。

（5）同步将变量值写入 PLC。

（6）实现与多个 PLC 同时通信。

（7）断开连接后自动重新连接 PLC。

（8）程序从 CODESYS 下载到 PLC 后自动重启。

（9）将信号事件（数据更改,状态更改）发送给客户端。

（10）获取、设置 PLC 应用程序的状态。

（11）访问 PLC 底层文件系统。

PLCHandler 以软件开发工具包（Software Development Kit，SDK）形式进行提供，包含所有 C++头文件、PLCHandler 库、示例配置文件和演示源代码等，也包括用于编译样本代码的工作区或生成文件。

典型的 PLCHandler 库作为静态链接库（例如 Windows 系统下：PLCHandlerLink.lib 或者 PLCHandlerLinkMFC.lib），封装了 C++类以及附加的纯 C 的接口。

对于 WINDOWS 平台，除了静态库，PLCHandler 还可以提供动态链接库（PLCHandlerDll.lib/dll）和 ActiveX-Control（PLCHandlerX.ocx）插件。

为了使 PLCHandler 能够在不同的操作系统上工作，需要使用 CODESYS Runtime System 中的系统组件。这些组件代表了硬件和操作系统抽象层，并且隐藏了 PLCHandler 的所有其他组件的处理器和操作系统具体的特性。每个系统组件都有其特定的功能，例如文件访问、访问堆内存、访问 RS-232 串行接口、访问操作系统任务等。

下图 14-2 说明了 PLCHandler 和 API 接口的内部架构。

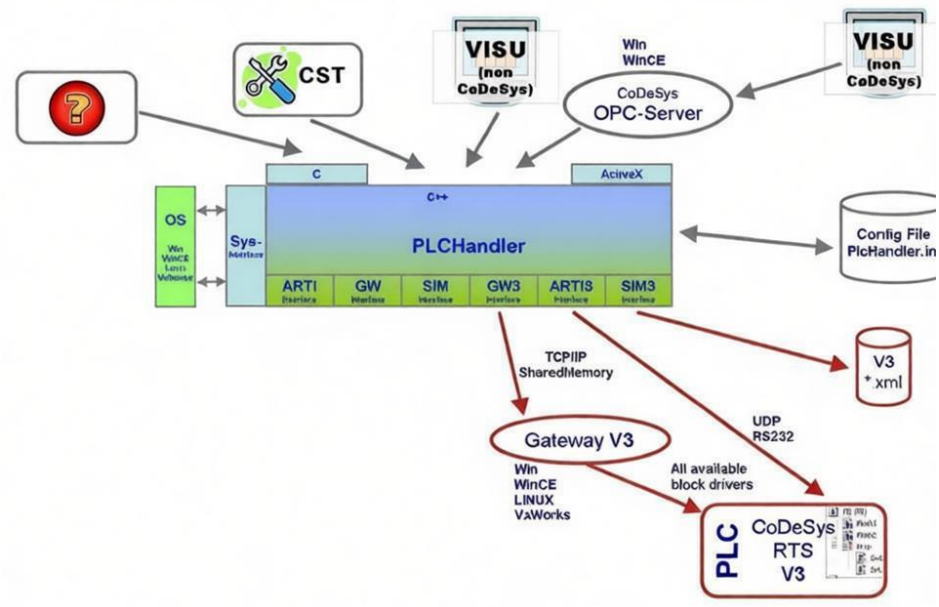


图 14-2 PLCHandler 和 API 接口内部架构

14.2 PLCHandler 通信的益处

PLCHandler 通信，又称为标签通信，可以用于以下场景：

（1）控制器与上位机（C、C#）通过 PLCHandler 数据交互。

（2）控制器与上位机（SCADA、Python）通过 OPC-UA 数据交互。

（3）控制器与支持 PLCHandler 的触摸屏（HMI）数据交互。

（4）多设备间符号化数据交互。

PLCHandler 这种标签通信的方式，摒弃了传统的通过变量映射地址、寄存器地址等进行访问的方式，直接通过变量名就可以访问，显著地带来以下方面的便利：

（1）免地址映射：摒弃传统 %IW、%QX 等硬地址，直接使用 MotorSpeed、Pressure 等语义化变量名通信，更加直观。

（2）支持复杂数据类型：兼容 BOOL、INT、REAL、STRING、STRUCT、ARRAY 等多种数据类型。

（3）降低错误率：修改变量名时自动同步，无需重新配置地址，减少人为失误。

（4）快速集成：一键生成 XML 标签文件，供 HMI/上位机直接导入，自动建立变量映射。

14.3 实现 PLCHandler 的准备工作

SC301 作为标准的基于 CODESYS V3 的控制器，能够支持 PLCHandler 通信，在实施之前，需要了解和做好以下准备事项：

（1）确认设备是否支持 PLCHandler 通信。首先要确认您的设备是否支持 PLCHandler 通信功能，这是前提条件。

（2）关于 PLCHandler 接口。如上图 14-2 中所示，SC301 控制器可支持 GW3 和 ARTI3 两种接口，GW3 依赖于 Gateway V3，ARTI3 不依赖 Gateway V3，但更推

荐使用 GW3 接口。如果实施 PLCHandler 通信的客户端计算机已经安装了 V3 以上版本的 CODESYS 软件，则无需再次安装 Gateway V3 软件；如果没有安装 CODESYS 软件，则需要单独安装 Gateway V3 软件。CODESYS 和 Gateway V3 软件安装包可以从高创传动官网下载，也可以联系技术支持人员获取。

（3）PLCHandler 开发包。为了降低使用难度，并提升开发效率，高创传动在 CODESYS 提供的内容基础上补充了内容，提供了开发包，开发包可以联系技术支持人员获取。开发包主要内容如下图 14-3 所示。

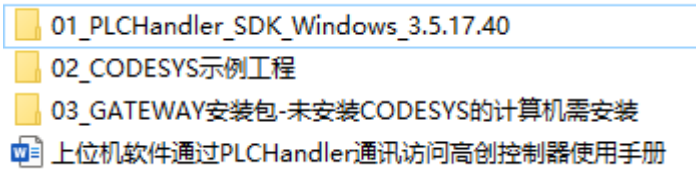


图 14-3 PLCHandler 开发包内容

开发包各部分的内容介绍如下表 14-1 所示。

表 14-1 PLCHandler 开发包内容说明

序号	项目	说明
1	01_PLCHandler_SDK_Windows_3.5.17.40	CODESYS 官方提供的 SDK，包含 C/C++头文件、PLCHandler 库、示例配置文件、演示源代码、编程手册、用于编译样本代码的工作区或生成文件。此外，本文件和发行信息均包含在内
2	02_CODESYS 示例工程	已经配置好的基于高创控制器的支持 PLCHandler 功能的 CODESYS 示例工程，以及用到的指令清单
3	03_GATEWAY 安装包- 未安装 CODESYS 的计算机需安装	CODESYS Gateway 3.5.17.40 安装包
4	04_上位机软件通过 PLCHandler 通讯访问高创控制器使用手册	指导如何根据开发包进行 PLCHandler 开发

14.4 PLCHandler 的具体实现

PLCHandler 的具体实现，由于篇幅所限，在此不再赘述，请参考我司提供的 PLCHandler 开发包资料，过程中如需指导，请联系我司技术支持人员。

15. 跨网段通信

跨网段通信是指在不同子网（网段）之间进行数据传输的过程。由于不同网段的设备通常使用不同的 IP 地址范围，无法直接进行通信。

SC301 控制器本身不支持跨网段通信，与之进行网络通信的设备 IP 地址需要与控制器的 IP 地址处于同一个网段内。但通过添加合适的网络设备（一般是路由器或三层交换机等网络设备）并经过正确设置后，也可以进行跨网段通信，其应用模式参考下图 15-1。

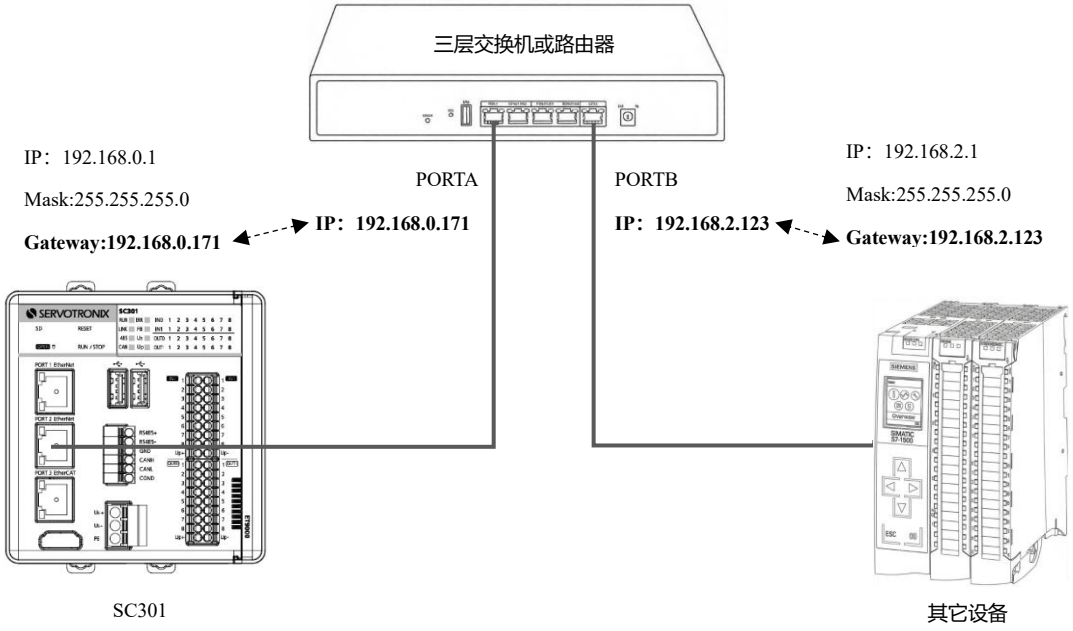


图 15-1 跨网段通信应用模式

15.1 基本配置方法与原理

以上图 15-1 所示的应用模式为例，一台路由器连接处于不同网段的 SC301 和另一台其它设备，路由器应具有不少于 2 个物理接口或子接口。SC301、其它设备及路由器应参考以下规则进行设计。

（1）SC301： IP 地址 192.168.0.1，子网掩码是 255.255.255.0，网关地址为

192.168.0.171，IP 地址与网关地址必须处于同一个网段。

（2）连接 SC301 的路由器的 PORTA：IP 地址 192.168.0.171，也就是 SC301 的网关地址。

（3）其它设备：IP 地址 192.168.2.1，子网掩码是 255.255.255.0，网关地址为 192.168.2.123，IP 地址与网关地址必须处于同一个网段。

（4）连接其它设备的 PORTB：IP 地址 192.168.2.123，也就是其它设备的网关地址。

（5）无需配置静态路由。

具体工作原理如下：

（1）路由器直连这两个网段，会自动在路由表中生成“Direct”（直连）路由条目。

（2）SC301 发送数据给其它设备时，发现目标 IP 地址不在同一网段，则将数据包发给网关（路由器 PORT A）。

（3）路由器查看路由表，发现目标网段对应 PORT B，于是从 PORTB 转发出去。

（4）其它设备接收到 SC301 的数据。

（5）其它设备将数据发送到 SC301 时，原理相同。

【注意】:

（1）应确保跨网段通信的两个设备的“默认网关”填写的是路由器对应接口的 IP 地址。

（2）若之前配置错误，需清除两个被连接设备和路由器的 ARP 缓存信息。

15.2 SC301 的配置方法

SC301 控制器有两个 Ethernet 接口，只有 PORT2 EtherNet 接口支持跨网段通信，其对应的网卡编号为 eth0，出厂默认的 IP 地址为 192.168.0.1，子网掩码为 255.255.255.0，网关地址为 192.168.0.171。

请参考本章节说明进行配置，如果需要修改该网口的任何一个网络通信参数，请参考“**12.配置网络及日期时间信息**”章节内容。

16. 使用 U 盘更新应用程序

通常情况下，应用程序在开发和调试期间，在 CODESYS 编程软件中经过编译后，通过网线或者串口直接下载到控制器中去运行，必要时进行单步调试和断点调试等，再对程序进行优化，比较方便。

但是应用程序开发完成和稳定后，再利用电脑通过网线下载应用程序就比较繁琐，尤其是需要对多台控制器下载相同的应用程序时效率较低，此时通过 U 盘对应用程序进行批量复制更新就非常方便。

16.1 生成应用程序

(1)按照下图 16-1 所示步骤，在 PC 上对已经调试好的 CODESYS 工程进行编译，创建启动应用，生成目标文件。

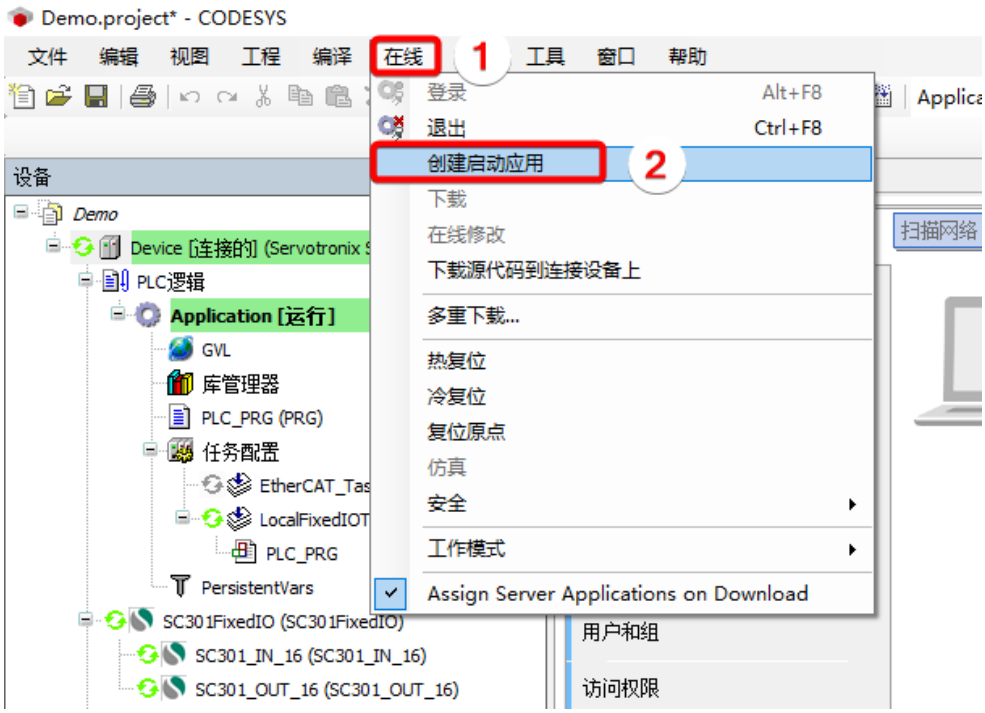


图 16-1 生成目标文件

(2)然后在弹出的对话框中，选择目标文件的保存路径，并对目标文件进行命名，本例中命名为 Application，如下图 16-2 所示。

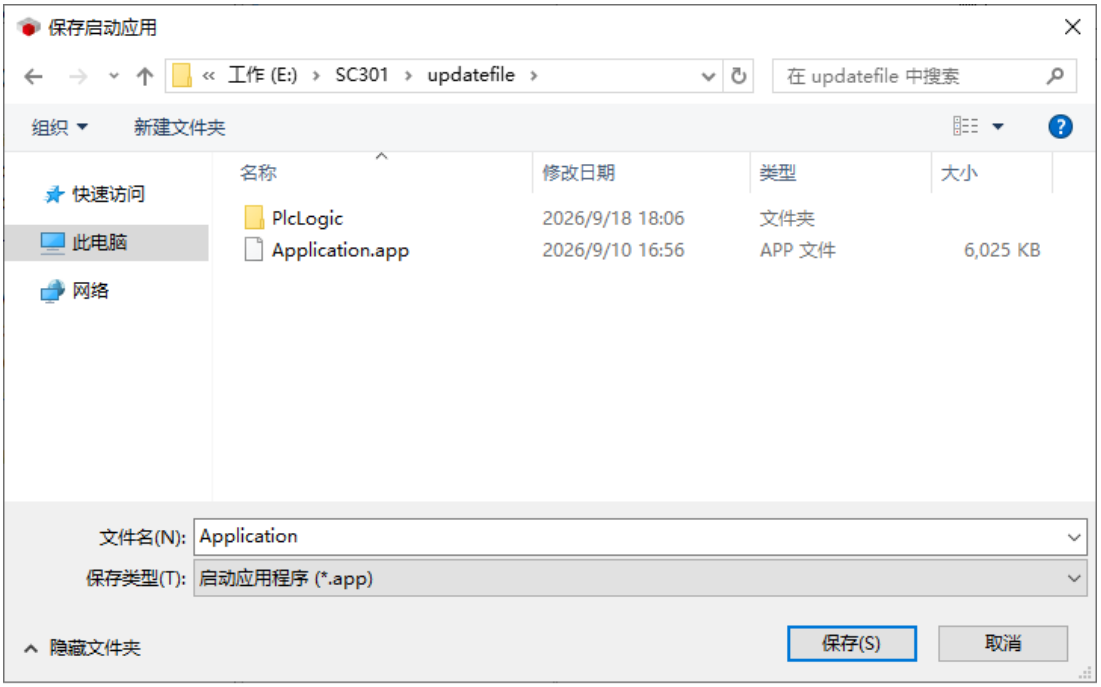


图 16-2 命名并保存目标文件

如下图 16-3 所示，Application.app、Application.crc 以及 Plclogic 文件夹即是编译完成后生成的目标文件，一共有 3 个。

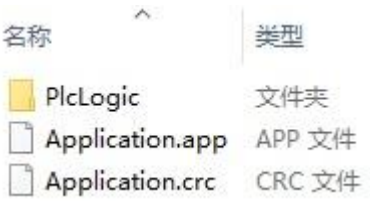


图 16-3 应用程序目标文件

16.2 在 U 盘内制作升级文件夹

(1)找一个市场上常见的主流品牌的 U 盘（较低概率下存在部分杂牌 U 盘无法识别的情况），并确保 U 盘文件系统格式 NTFS/FAT32，如不是，请重新进行格式

化。

(2)在 U 盘根目录下，新建一个名为 updatefile 的文件夹。

(3)将应用程序目标文件 Application.app、Application.crc 以及 Plclogic 文件夹一起拷贝到 updatefile 文件夹中，缺一不可。

16.3 应用程序更新操作

(1)将控制器断电。

(2)插入 U 盘。

(3)控制器上电，即可自动完成应用程序更新。更新完成并正常运行后，RUN 指示灯将呈现绿色常亮状态。RUN 指示灯的详细状态参考参考“**4.7 系统状态指示灯**”章节。

17. 固件更新

SC301 在市场销售和推广的过程中，在需要增加新功能、优化原有功能、解决新发现缺陷等情况下，高创传动会对固件进行更新。建议您及时更新固件，以便更及时、更好的获得产品体验。

更新固件通常通过将产品发回厂家进行，但也可以在厂家的协助下，在应用现场完成，SC301 支持通过工具软件、TF（MicroSD）卡 2 种固件更新方式。

【注意】：在某些情况下，固件版本与硬件版本存在一定的匹配性，贸然更新固件可能导致产品失效无法工作。因此在更新固件之前，建议跟高创技术支持人员进行确认为妥。

17.1 通过系统更新工具软件更新固件

SC301 可以通过<高创 CodeSys 运动控制器系统更新工具_Vx.x>软件来更新固件，其中 x.x 为软件的版本。新固件及该工具软件可以联系高创技术支持人员获取。

固件更新前的准备工作如下：

(1)控制器准备。待更新固件的 SC301 控制器，可以选择 PORT1 Ethernet 或者 PORT2 Ethernet 来进行固件更新。**两者的区别在于 PORT2 的通信速率更快，可以在更短的时间内完成固件更新。【注意】：PORT1 的默认 IP 地址为 90.0.0.1，子网掩码为 255.255.255.0，如果 IP 地址有修改，请以修改后的为准；PORT2 的默认 IP 地址为 192.168.0.1，子网掩码为 255.255.255.0，如果 IP 地址有修改，请以修改后的为准。**

(2)计算机准备。一台安装了 windows10 操作系统的计算机，计算机至少有一个网口，且计算机上已经成功地安装了<高创 CodeSys 运动控制器系统更新工具_Vx.x>软件。然后修改计算机所选网口的 IP 地址，设置成与 SC301 所选网口的 IP 地址处于同一网段，子网掩码设置为 255.255.255.0。

(3)连接网线。将 SC301 上电，并通过网线与计算机连接，当 SC301 的绿色

RUN 运行指示灯点亮之后，表示 SC301 已经正常运行。

(4)启动软件。运行《高创 CodeSys 运动控制器系统更新工具_Vx.x》软件（以下简称固件更新工具），其界面见下图 17-1。

图 17-1 固件更新工具软件界面

上述准备工作完成后，按照以下步骤操作更新固件：

(1)建立连接。在软件界面的“连接控制器”栏，选择 SC301，并在 IP 地址输入框正确填写当前连接的 SC301 网口的 IP 地址，然后点击“建立连接”按钮。待连接成功之后，更新工具的“连接状态”会显示“已连接”。“系统信息”栏会显示出当前 SC301 的固件版本，如下图 17-2。

图 17-2 建立连接并显示当前固件版本信息

(2) 选择固件包并确认更新。点击“更新包”栏的“选择”按钮，找到新固件存放的位置并选中，此时“开始更新”按钮变为可操作状态，然后点击“开始更新”。之后会弹出对话框进行二次提醒确认，点击“确认”按钮，即启动固件更新；若点击“取消”按钮，则放弃固件更新操作，如下图 17-3。

图 17-3 固件更新确认

(3) 重启控制器。如下图 17-4，固件更新完成后，会在“开始更新”栏的“更新状态”中显示：“已成功更新，请将控制器断电重启”，接下来请重启控制器。**【注意】：SC301 断电重启时，需要等待 SC301 的指示灯全部熄灭之后再上电，否则可能导致断电重启失败。**

图 17-4 固件更新完成

(4)确认。固件更新完成之后，可以重新打开更新工具并连接 SC301，可以获取更新后的 SC301 的固件版本，通过版本号确认固件升级成功。

17.2 通过 TF（MicroSD）卡更新固件

请按照以下步骤进行操作：

(1)写入镜像文件。通过读卡器等设备将待升级的新版本固件镜像写入到 TF 卡中。

(2)插卡。将 SC301 断电，打开翻盖，然后将 TF 卡插入 SC301 的 SD 卡插槽内（SD 卡插槽位于 RUN/STOP 拨动开关的旁边）。**【注意】：插入 TF 卡时，要确保 TF 卡的金手指的方向与卡槽的金手指的方向匹配，且操作时需要小心，以免 TF 卡没有插准掉入 SC301 的机壳内，难以取出**。

(3)按住 RESET 按键同时上电。用螺丝刀等比较细的工具按压住 SC301 的 RESET 按键的同时（用手指按压可能导致按压的不充分，不能触发固件升级）给 SC301 上电。

(4)松开 RESET 按键。待 SC301 的 RUN 运行指示灯变为红色时，松开 SC301 的 RESET 按键。

(5)固件升级完成。等待 SC301 的 RUN 运行指示灯再次变成绿色，表示固件升级完成。

(6)重启控制器。将 SC301 断电重启，之后新的固件就可以运行了。**【注意】：SC301 断电重启时，需要等待 SC301 的指示灯全部熄灭之后再上电，否则可能导致断电重启失败。**

18. 随机配件

SC301 控制器的随机配件为接线端子，打开产品包装后，在下图 18-1 所示位置取出使用。

图 18-1 接线端子在包装内存放位置

接线端子明细如下表 18-1 所示，如有缺失，请及时联系高创售后处理。

表 18-1 SC301 控制器接线端子明细

端子名称	位数	数量	高创物料编码
24VDC 电源输入端子	3（3*1）	1	11201707001376
RS-485&CAN 端子	6（6*1）	1	11201707001395
数字量输入端子	18（9*2）	1	11201707001375
数字量输出端子	18（9*2）	1	11201707001375

附录 1：ET 系列扩展 I/O 系统选型表

SC301 可以选择 ET 系列扩展 I/O 系统进行扩展，扩展方式有两种：本地集中扩展，或通过 EtherCAT 耦合器远程集中扩展。

表 附录 1-1 ET 系列扩展 I/O 系统选型表

型号	名称	说明
ET0100	EtherCAT 耦合器	ET 系列
ET1216	16 通道数字量混合输入输出模块	8 通道数字量输入：NPN/0V 有效 8 通道数字量输出：NPN/0V，0.5A/通道
ET1316	16 通道数字量混合输入输出模块	8 通道数字量输入：PNP/24V 有效 8 通道数字量输出：PNP/24V，0.5A/通道
ET9000	ET 终端盖板	搭配 ET0100 使用，每个耦合器需配一片

附录 2：TL 系列远程 I/O 系统选型表

SC301 可以选择 TL 系列远程 I/O 系统进行扩展，扩展方式为通过 EtherCAT 耦合器远程集中扩展。

表 附录 2-1 TL 系列远程 I/O 系统选型表

型号	名称	说明
TL0100	EtherCAT 耦合器	TL 系列
TL1016	16 通道数字量输入模块	PNP/24V 有效
TL1116	16 通道数字量输入模块	NPN/0V 有效
TL1516	16 通道数字量混合输入输出模块	8 通道数字量输入：PNP/24V 有效 & NPN/0V 有效 8 通道数字量输出：PNP/24V，0.5A/通道
TL2016	16 通道数字量输出模块	PNP/24V，0.5A/通道
TL2116	16 通道数字量输出模块	NPN/0V，0.5A/通道
TL3104	4 通道电压隔离输入模块	0～5VDC/0～10VDC/±5VDC/±10VDC, 15 位/16 位，**通道隔离**
TL3108	8 通道电压输入模块	0～5VDC/0～10VDC/±5VDC/±10VDC, 15 位/16 位
TL3204	4 通道电流输入模块	0&4～20mA，15 位，单端
TL3208	8 通道电流输入模块	0&4～20mA，15 位，单端
TL3703	3 通道热电阻输入模块	RTD-PT100
TL3704	4 通道热电阻隔离输入模块	RTD-PT100，**通道隔离**
TL3803	3 通道热电阻输入模块	RTD-PT1000
TL3904	4 通道热电偶输入模块	J / K/ E / T / S / R / B / N / C 型
TL4104	4 通道电压输出模块	0～5VDC/0～10VDC/±5VDC/±10VDC,16 位
TL4204	4 通道电流输出模块	0&4～20mA，16 位，单端
TL5102	2 通道编码器输入模块	5V 单端，输入频率≤1.5MHz
TL5202	2 通道编码器输入模块	24V 单端，输入频率≤500KHz
TL6001	1 通道串口扩展模块	支持 Modbus 主站/Modbus 从站/自由协议
TL9000	TL 终端盖板	搭配 TL0100 使用，每个耦合器需配一片
TL9410	电源扩展模块	SV：5V/2A FV:24V/8A（无状态，无需组态）

高精高速 科技创造美好未来

High precision and high speed, technology creates a better future.

高创传动科技开发（深圳）有限公司

Servotronix Motion Control Ltd.

地址：深圳市南山区西丽街道松坪山社区宝深路99 号科陆大厦B 座13B01

电话：400-111-8669

网址：www.servotronix.cn