DECOWELL®
德克威尔自动化

工业信号处理专家

AX-3000系列运动控制器

CodeSys软件使用手册

德克威尔 • 工业智造可靠伙伴

网址：www.welllinkio.com
邮箱：sales@welllinkio.com
地址：南京市浦口区兰新路19号瑞创智造园13号楼

前 言

资料简介

感谢您购买德克威尔 AX-3000 运动控制器！

AX-3000 系列运动控制器是高性能中型 PLC，拥有小体积、模块化、高性能 等优点。其拥有强大的运动控制能力，可带 32 个伺服轴，同时提供各种应用库，包括基于 EtherCAT 的运动控制库，运动控制包括点对点运动，凸轮运动，插补运动等。另外集成了 Modbus TCP Master/Slave、Modbus RTU Master/Slave,且硬件上包含了 RS485、USB 等串口。支持本体高速计数、拓展 EX 系列 IO 等。

本手册面向德克威尔可编程逻辑控制器（以下简称中型 PLC）。主要介绍 PLC 编程软件 Codesys 对控制器编程时所需的网络配置、编程环境、编程语言、程序诊断等相关知识。

目 录

安全注意事项

安全声明

01. 在安装、操作、维护产品时，请先阅读并遵守本安全注意事项。

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

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

04. 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，因未遵循相关规定引发的功能异常或部件损坏等不在产品质量保证范围之内。

05. 因违规操作产品引发的人身安全事故、财产损失等，德克威尔不承担任何法律责任。

安全等级定义

提示

该标记表示 “对操作的描述进行必要的补充或说明”。

注意

该标记“未按要求操作造成的危险，会导致人身轻度或中度伤害和设备损坏”。

警告

该标记表示“由于没有按要求操作造成的危险，可能导致人身伤亡”。

控制系统设计时警告

01.应用时请务必设计安全电路，保证当外部电源掉电或扩展模块故障时，控制系统依然能安全工作；

02.输出电路中由于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。

控制系统设计时注意

01.务必在扩展模块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；

02.为使设备能安全运行，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；

03.扩展模块的继电器、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；

04.扩展模块设计应用于室内、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扩展模块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。

1. CodeSys 概述

1.1 Codesys 简介

Codesys 是面向 PLC 的编程组态软件，为 PLC 提供一套完整的配置、编程、调试和监控环境，可以灵活自由地处理功能强大的 IEC 语言。

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

- I/O 模块配置；
- EtherCAT 总线；
- Modbus RTU Master/Slave；
- Modbus TCP Master/Slave；
- 高速 I/O。

支持程序的编写、下载和调试等功能，并为编程者提供如下便利：

- 标准化编程（符合 IEC 61131-3 标准） 支持多种编程语言：结构化文本（ST）、梯形图（LD）、顺序功能图（SFC）、功能块图（FBD）和 IEC61131-3 扩展编程语言 连续功能图（CFC）。
- 灵活的功能块库、全面的功能块库并支持用户自定义库。
- 离线仿真功能，不需要连接 PLC 硬件完成程序调试仿真。
- 智能的调试查错功能预编译及编译查错，快速定位编程错误，诊断及日志。
- 采样跟踪过程变量的时序图建立。

1.2 CodeSys 与硬件的连接

编程设备可以通过以太网（可经过集线器、交换机等）与中型 PLC 相连，使用 CodeSys 软件编写用户程序，将程序下载到 PLC 后进行程序监控并控制 PLC。

1.3 软件的获取

AX3000 的用户编程软件 CodeSys 为免费软件，如需安装文件及中型 PLC 系列产品的参考资料等，可通过以下途径获取：

- 从德克威尔的各级经销商处获得软件,建议从经销商获取，从官网上获取下载较慢。
- 在 CodeSys 技术官网（http://www.codesys.cn/?ivk_sa=1024320u）的“下载专区”页面免费下载软件安装包.

然后可以下载。

软件安装环境要求

具备以下条件的台式 PC 或便携式 PC 机：

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

1.4 安装步骤

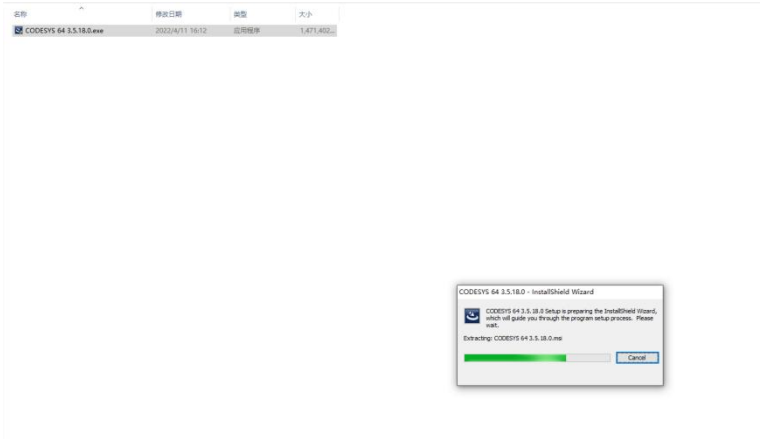
安装前准备：

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 5GB 以上，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。

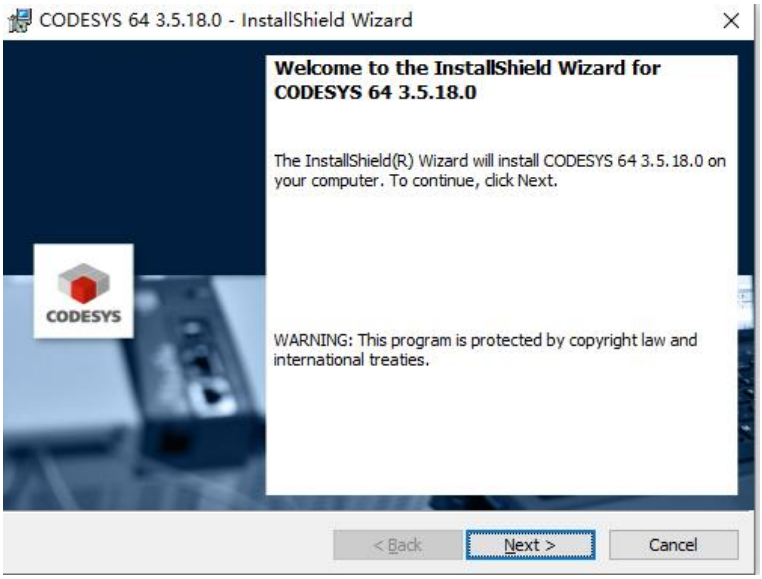
开始安装：

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

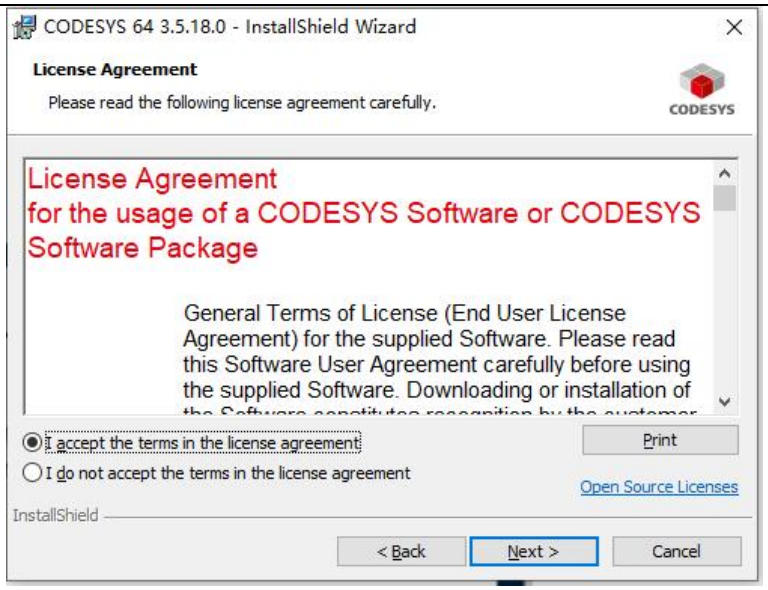
启动安装后，可以看到如下界面，表示进入安装准备阶段：



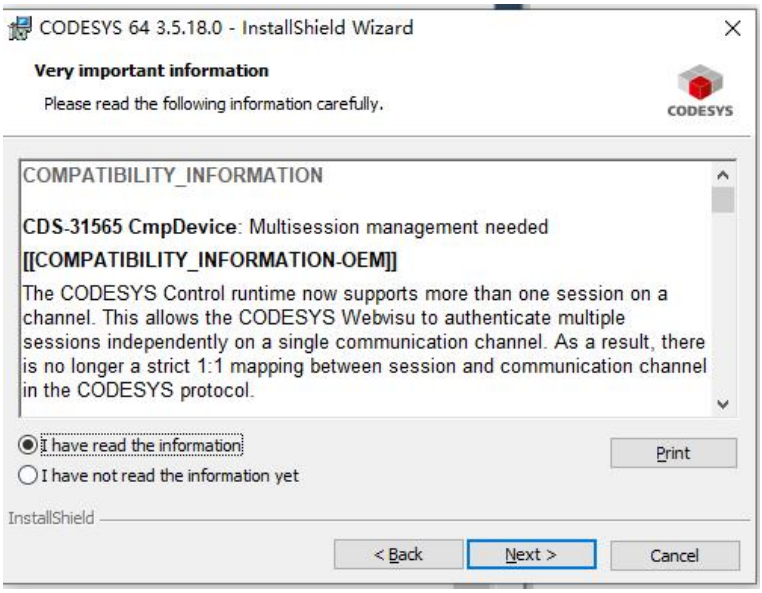
出现如下提示界面后，点击“Next”，开始安装



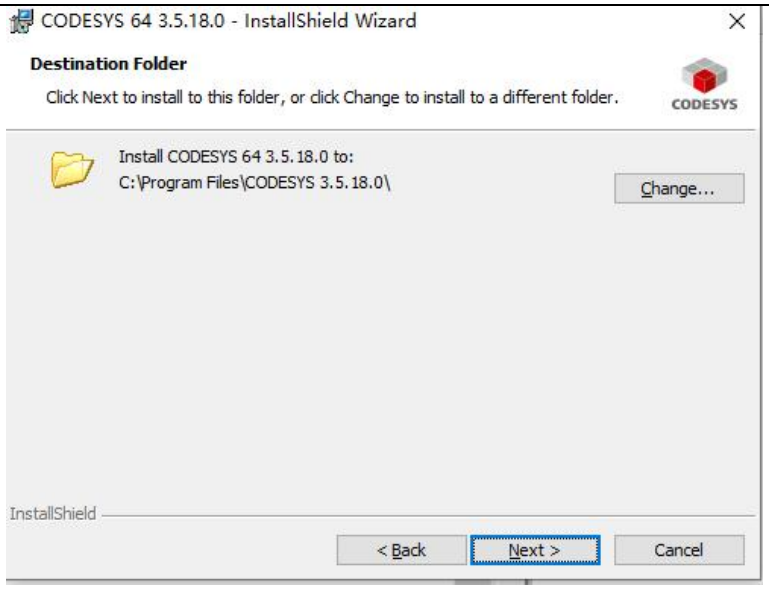
选择接收条款，然后点击 Next



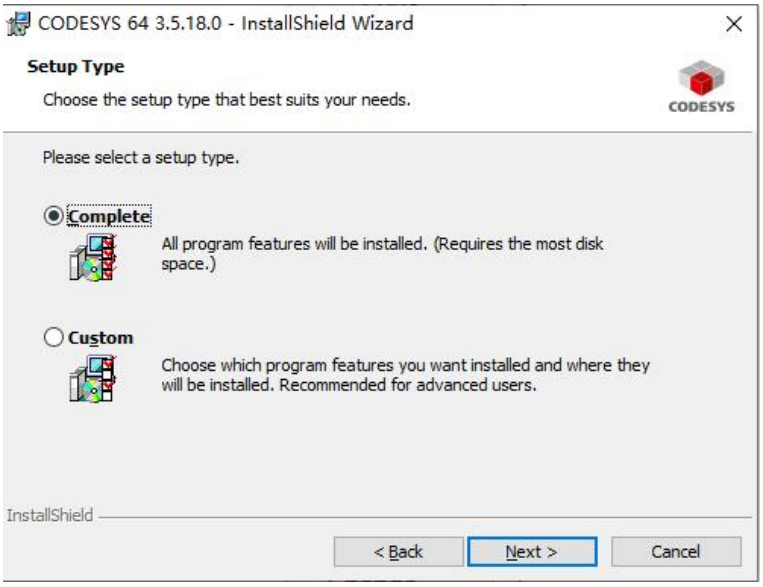
选择已经阅读了信息，然后点击 Next



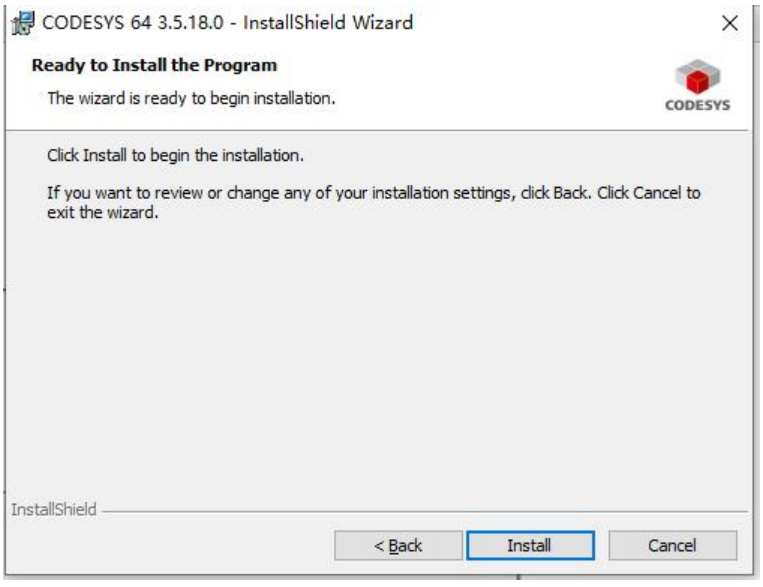
选择安装路径，然后点击 Next



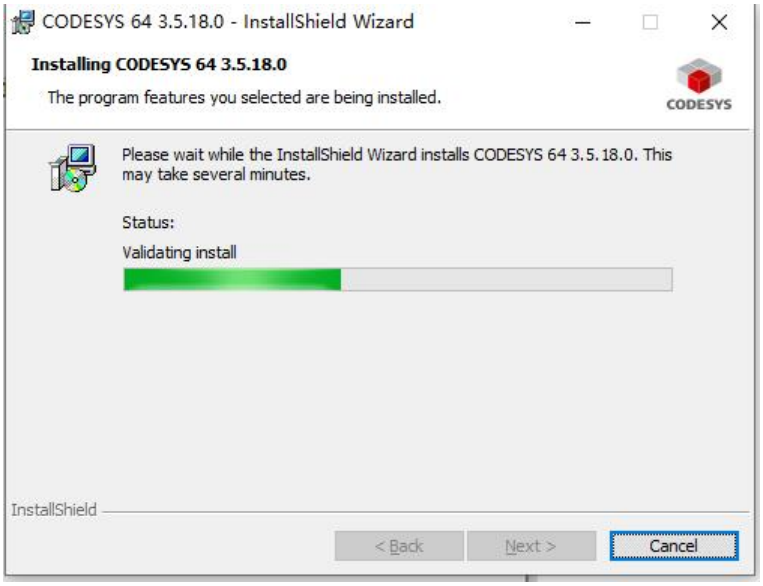
点击 Next



点击 Install



等待安装完成



1.5 卸载 CodeSys

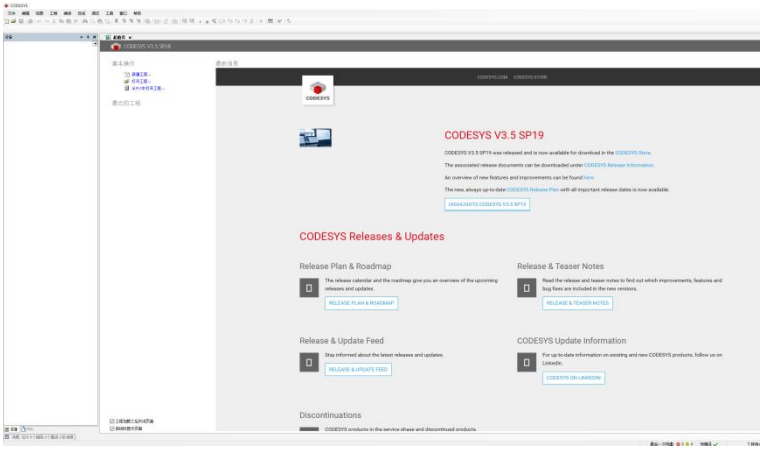
使用标准 Windows 系统卸载软件方法卸载 CodeSys 即可，具体步骤如下：

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

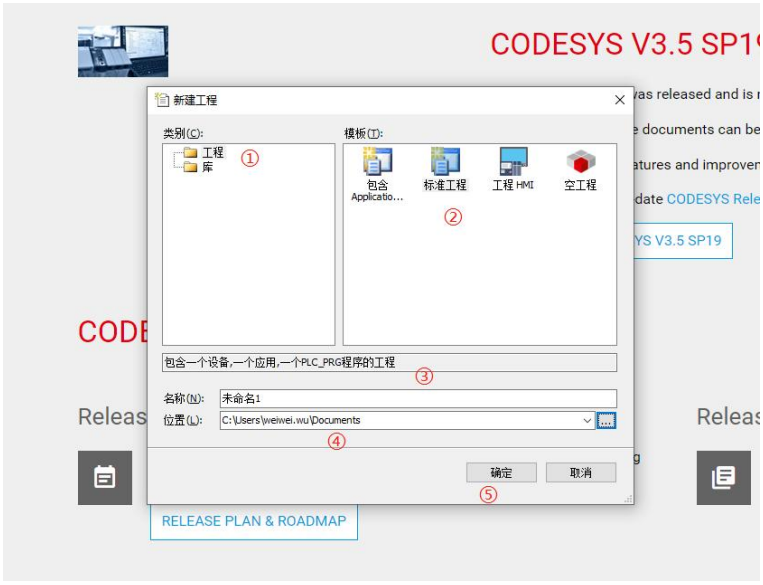
2. 快速入门

2.1 启动编程环境

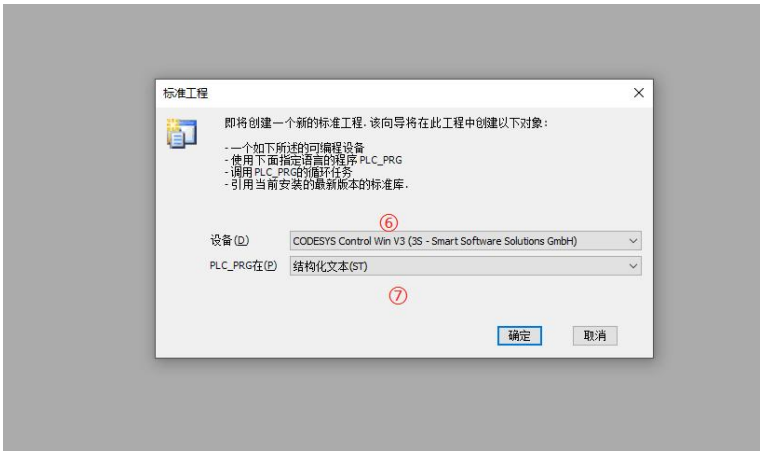
双击桌面编程软件图标 即可启动 CodeSys 编程环境，起始页显示画面如下：



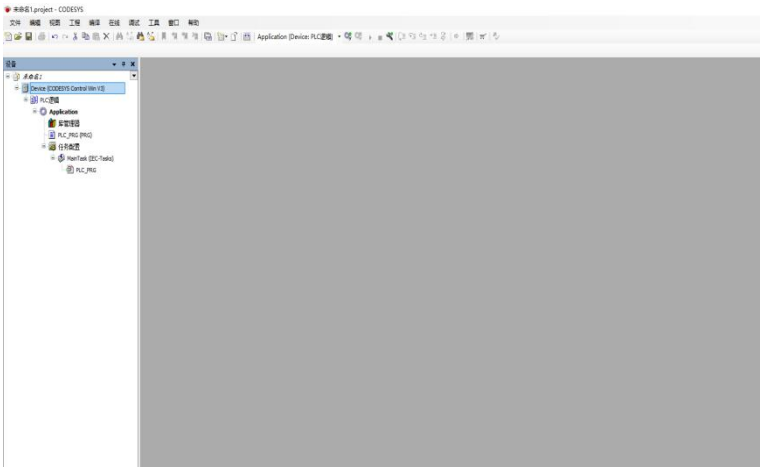
点击菜单栏左上角 新建工程或者选择“文件”-“新建工程”，选择工程类型“标准工程”选择设备类型和编程语言，并指定工程文件名及保存路径，如下图所示：



选择标准工程——>输入工程名称——>指定工程所在的位置——>点击确定——>创建成功



选择选择我司相对应的设备型号(需要提前添加我司设备描述文件)——>选择熟悉的编程语言——>点击“确定”后，进入系统组态配置与编程界面，常用的按钮与窗口分布如下图：

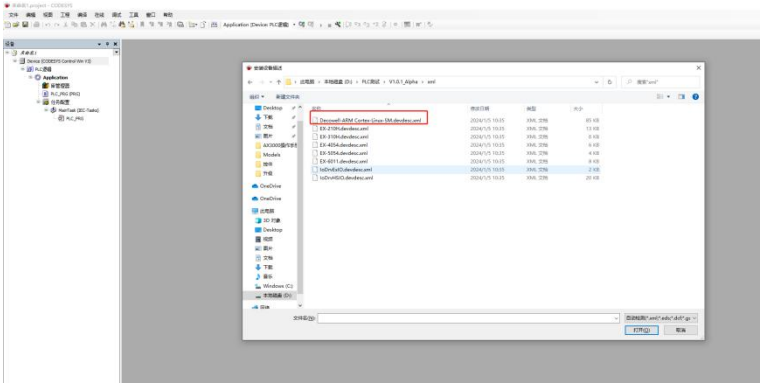


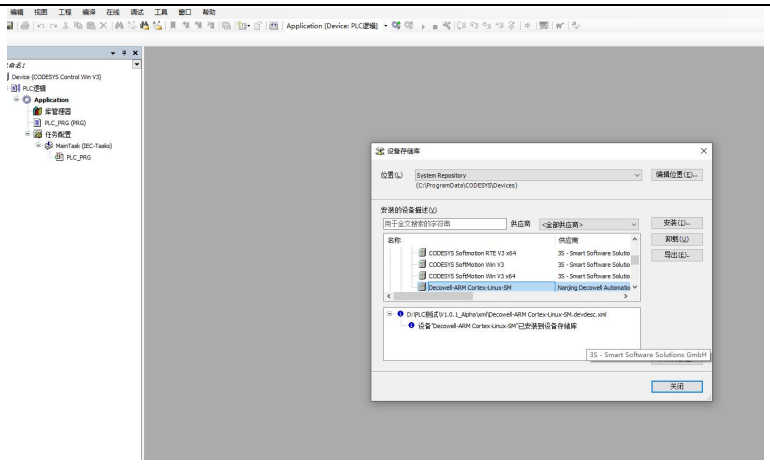
2.2 编写用户程序的典型步骤

2.2.1 基本操作

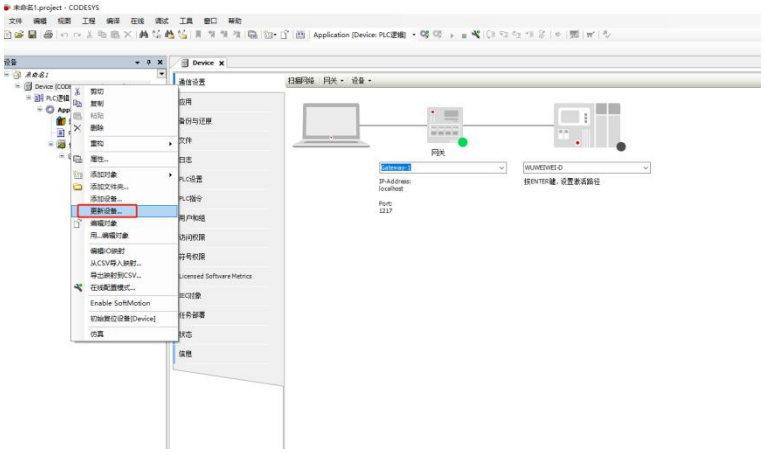
初次使用德克威尔 PLC 的用户需要注意，编写调试一个完整的用户程序需要 5 个步骤。

在设备库中添加安装 AX3000 设备描述文件

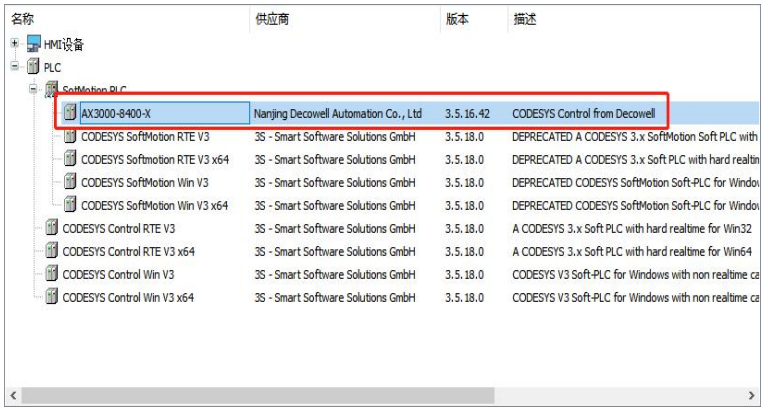




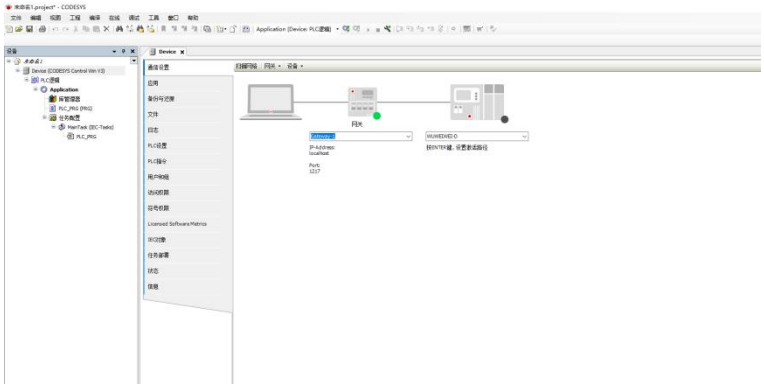
右击 Device，选择更新设备



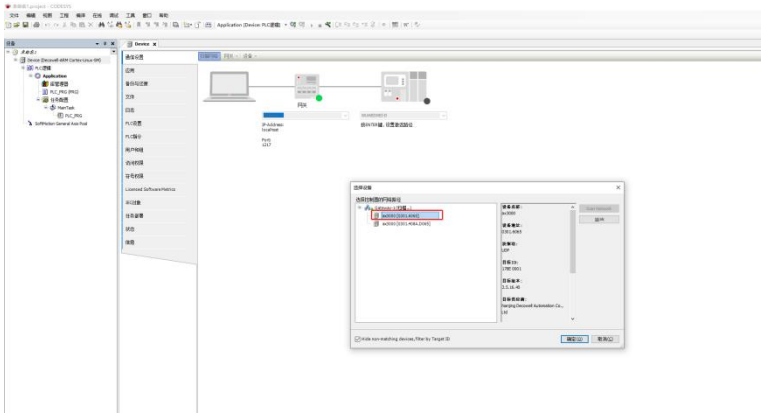
在 SotfMotion PLC 中选择 AX3000-8400-X，然后点击更新设备



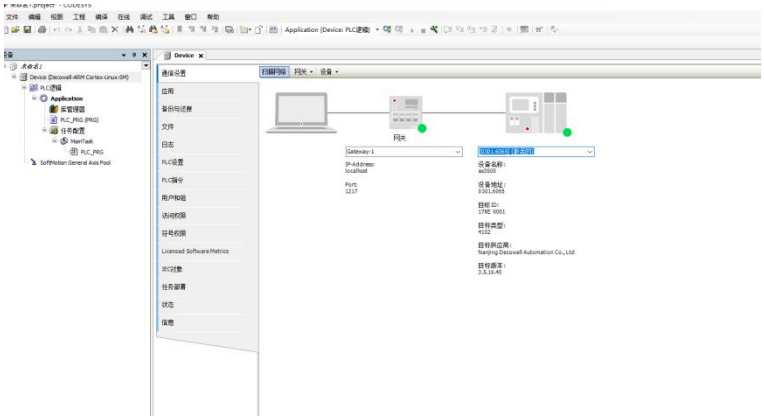
双击 Device 进入网络配置页面



扫描网络，点击确定

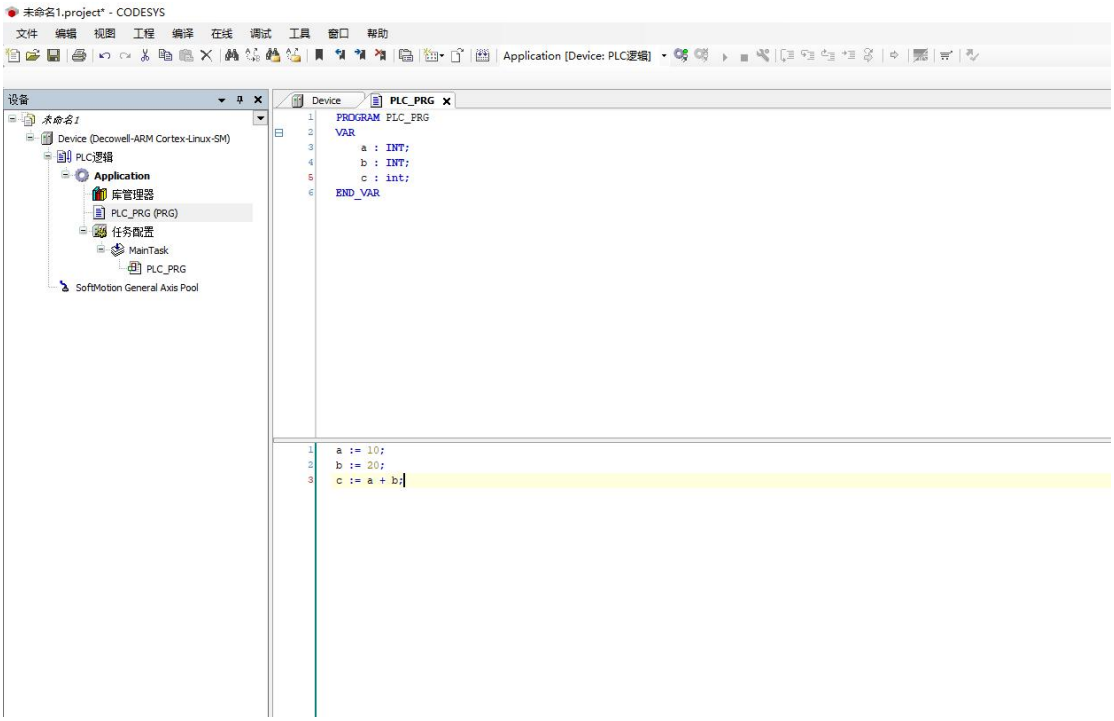


成功扫描到网络



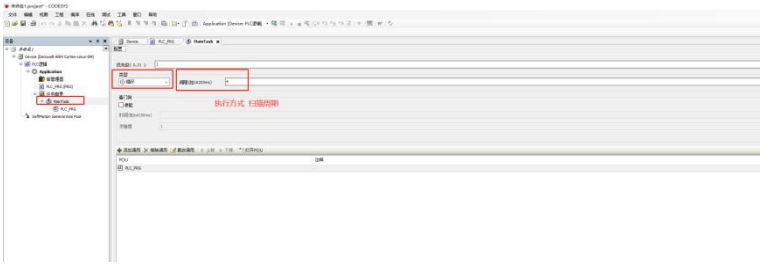
2.2.2 任务配置

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



2.2.3 配置用户程序的执行方式和运行周期

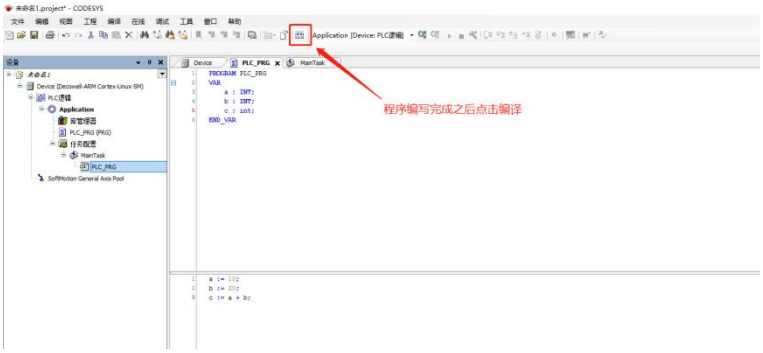
上文示例中编写的子程序默认为 4ms 执行一次，如要改为其他的执行方式，如反复执行、定时执行和执行周期等，可以按下图所示分别设置：



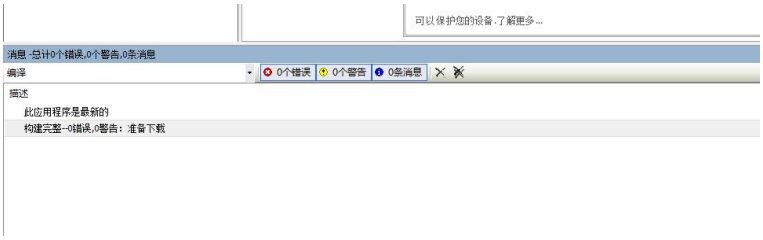
2.2.4 用户程序的编译、登录下载

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

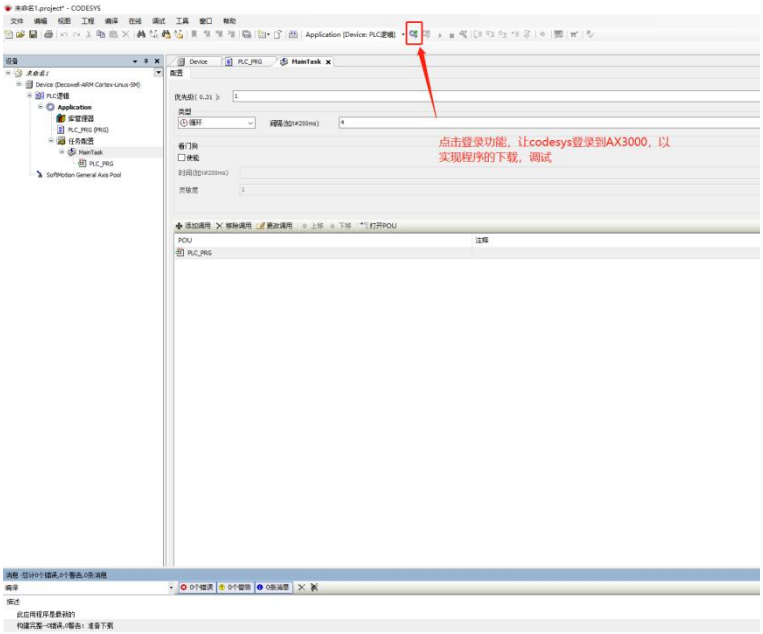
相关编译信息会显示在如下编译信息框中：



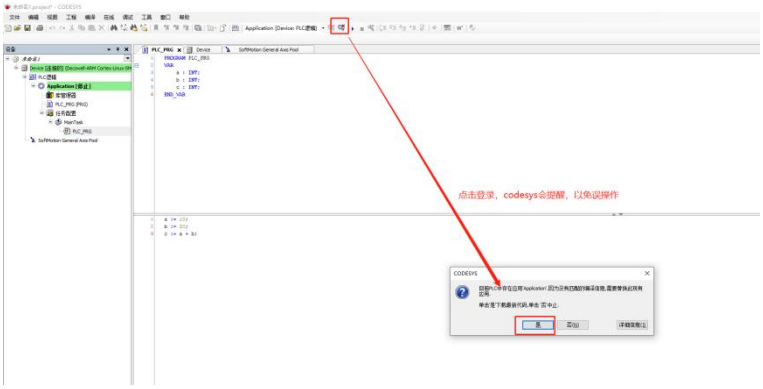
相关编译信息会显示在如下编译信息框中：



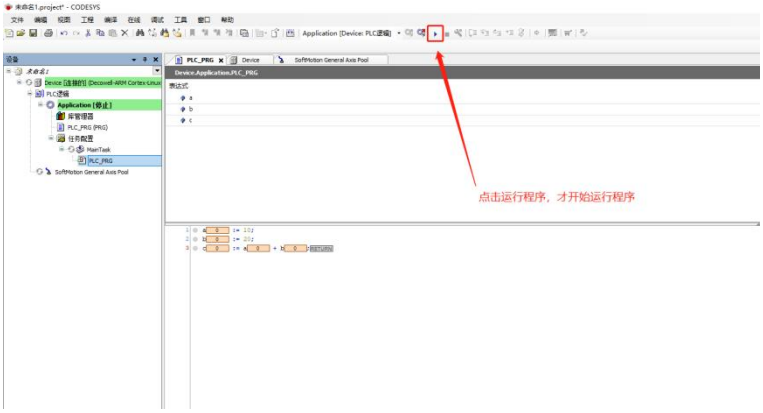
编译无误后，点击“在线”-“登录到”，如下图所示：



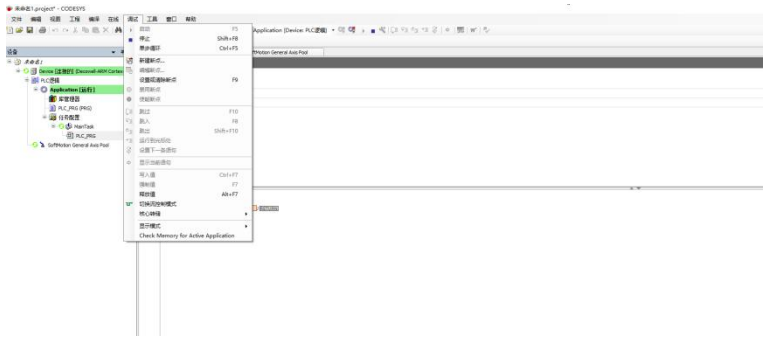
然后弹出如下对话框，选择是否创建程序并继续下载：



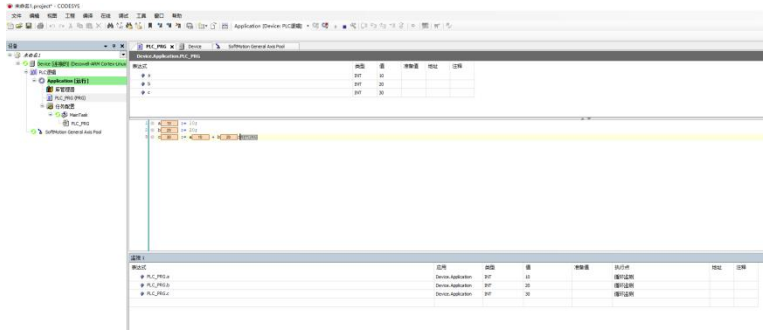
选择“是”，上位机与设备建立连接并保持，初始状态为“停止”，如下图所示：



点击“调试”-“启动”，设备进入运行状态，并开始执行用户程序。



正在运行的用户程序监控画面如下图所示：

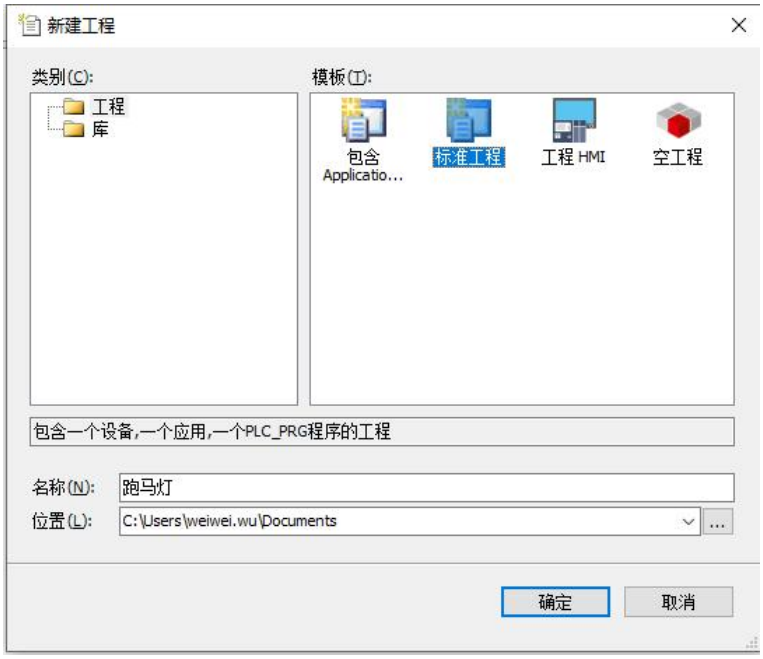


2.2.5 用 CodeSys 编写一个跑马灯样例工程

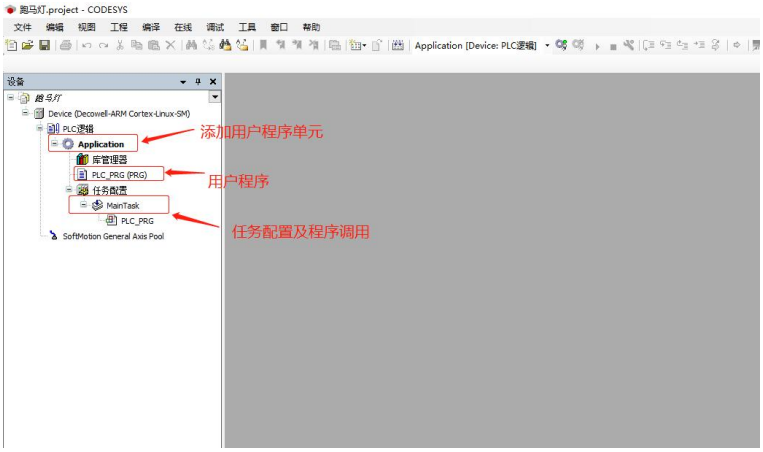
启动 CodeSys 编程环境

新建工程：

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

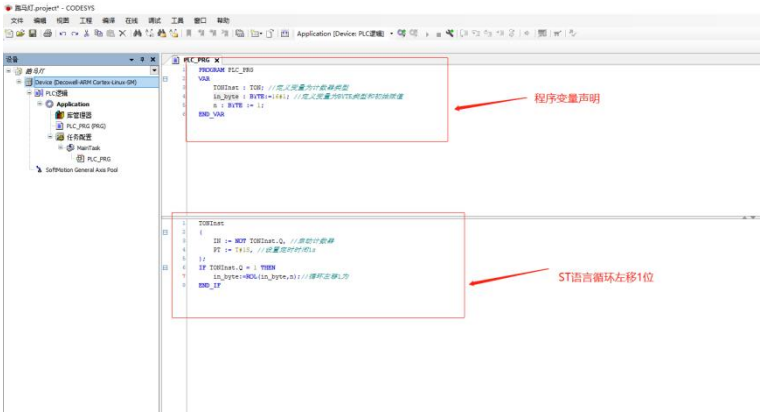


系统组态配置与编程界面

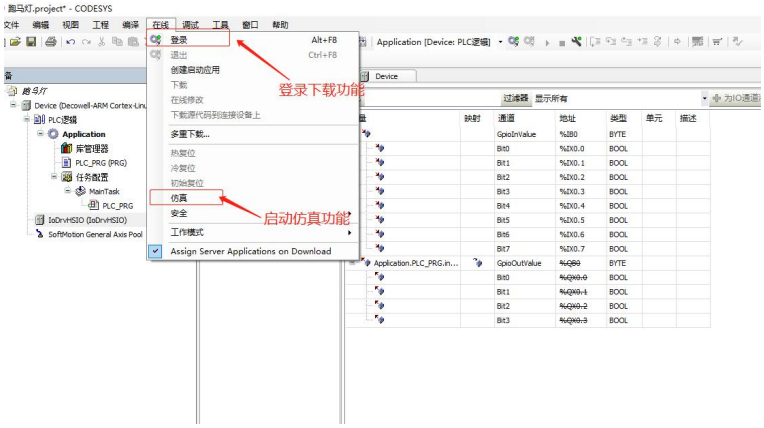


ST 语言编写“跑马灯样例程序”

双击打开“PLC_PRG”程序组织单元。



仿真调试

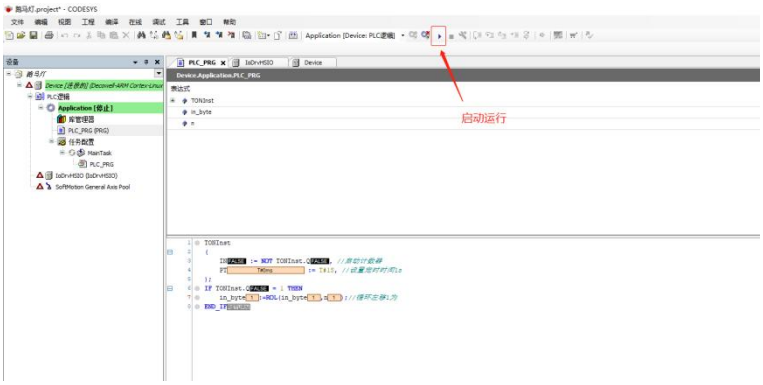


点击“仿真”进入仿真功能

在仿真模式下下载程序

点击“登录”，在仿真模式下下载程序。

下载完成后，启动运行 PLC



2.3 登录主模块

2.3.1 登录主模块的必备条件与操作简介

登录主模块”是指在 PC 上运行的 CodeSys 与 AX3000 主模块建立通讯，从而进行用户程序的运行、下载、启停和监控，以及程序参数查看或修改操作等。

● 目前可以通过通过 LAN 局域网登录到 AX3000

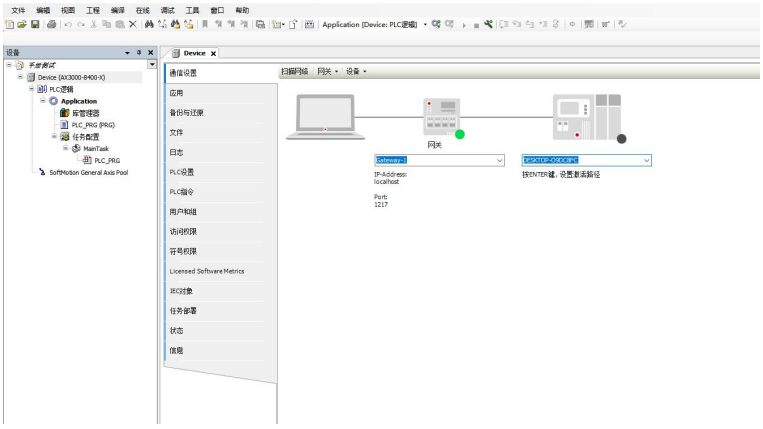
● PC 与 AX3000 之间可以通过网线进行 1 对 1 直连；也可以通过路由器、集线器无线连接 PLC，这种情况下， 一台 PC 可以与多台中型 PLC 联机，也可以多台 PC 访问同一个中型 PLC。

● PC 与 AX3000 两者的 IP 地址必需在同一个网段才能登录，否则 CodeSys 将无法扫描到 AX000。比如 AX000 的出厂默认 IP 地址为 192.168.1.101，若 PC 的 IP 地址为 192.168.1.xxx（这里 xxx 范围为 1~254，但 不得与 AX3000 的 IP 相同），那么 CodeSys 就可以扫描到 AX3000 并与之交互数据，进行用户程序下载、 运行监控等。若 AX3000 的 IP 被人为修改过，其地址与 PC 不在同一 IP 网段，二者将无法建立通讯，此时需要将 AX3000 的 IP 地址恢复为出厂地址 192.168.1.101，再将 PC 本机的地址修改为 192.168.1.xxx，二者建立 1 对 1 联机后，将 AX3000 的地址修改为所需的 IP 网段地址。

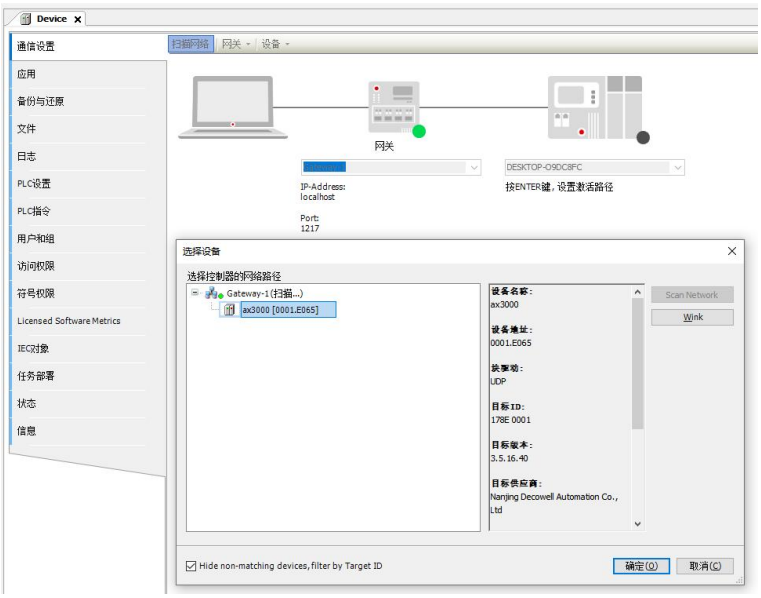
2.3.2 在 CodeSys 中扫描 AX3000 网络设备

PC 可通过 LAN 网络登录中型 AX3000

在 CodeSys 中，双击 Device(AX3000-8400-X)，弹出如下界面：



在该界面点击“扫描网络”标签，弹出如下界面，在窗口左侧点击其中的 AX3000 控制器，即可在窗口右侧可以看到其简介信息：



用户点击 wink，AX3000 数码管上闪烁 88，即为扫描到的 PLC 设备。

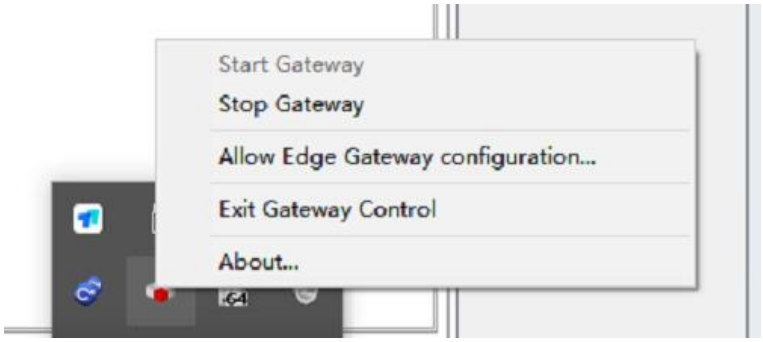
2.3.3 扫描不到设备的处理对策

如果在 CodeSys 中扫描不到 AX3000 设备，可能原因和对策如下：

1.CodeSys 网关没有启动。

解决方法：重新启动网关后，再进行扫描。

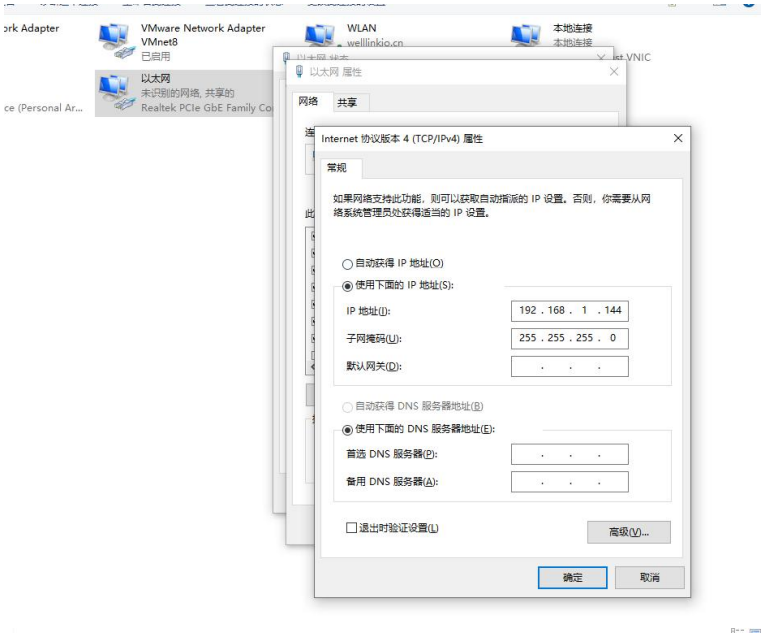
检查电脑右下方任务栏中 CodeSys 网关是否已经打开（以彩色显示），若为 Stop 状态，请点击启动



2. PC 的 IP 地址与 AX3000 的 IP 地址不在同一个网段。

解决方法：将 PC 的 IP 地址设置到 AX3000IP 地址的同一网段。如果忘记了 AX3000 的 IP 地址，建议将其恢复为默认 IP(使用 IP 复位功能)，再将 PC 的 IP 地址设置为 192.168.1.xxx 网段，即可正常扫描设备。

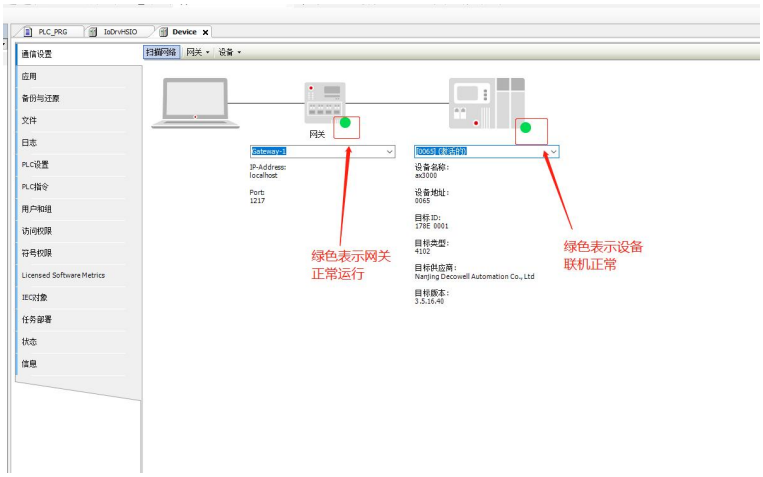
a. 首先在 PC 的资源管理器中，点击网络的本地连接，按下图进行 IP 地址的检查和修改：



b. 其次恢复 AX3000 的出厂默认 IP 地址（192.168.1.101）。AX3000 上电启动后，将 RUN/STOP 拨到 STOP 状位置，然后按住 MFK 键直到数码管显示 IP，松开按键；如果确认要复位 IP 地址，再按一下 MFK 键，数码管 开始显示 10，9，8... 倒计时：在计数到 0 之前按 MFK 键取消复位操作，倒计时结束，IP 复位完成，PLC 重新上电后，将使用新的 IP 地址。

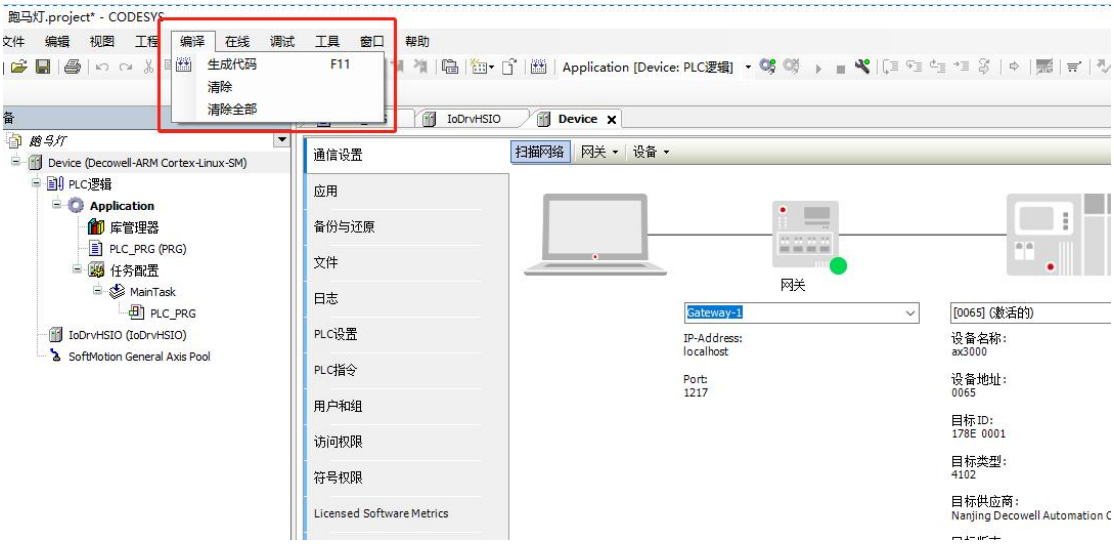
无论是复位 AX3000 的默认 IP，还是通过 CodeSys 后台软件修改 AX3000 的 IP 地址，然后重启 PLC，都会使修改后的 IP 地址立即生效。

一旦扫描联机成功，在设备扫描画面可看到如下网络状态信息：



3. 基本功能

3.1 编译命令



编译功能

- 检查程序：检查用户程序编写有无错误。
- 编译：把所有代码编译链接为 PLC 可以执行的代码。
- 重新编译：清除上次的编译信息，重新执行编译命名。
- 生成 Runtime 文件：用于研发测试，不做详细解释。
- 清除：清除上次的编译信息和下载信息。
- 清除全部：清除编译信息、下载信息、引用信息，所有的库、工程数据编译信息重新刷新。
- 打包用户程序：另做详细介绍。

生成代码更新为编译后，对于标签通讯中符号配置所要用到的生成文件功能，也点击编译按钮来实现，或者  点击快捷键

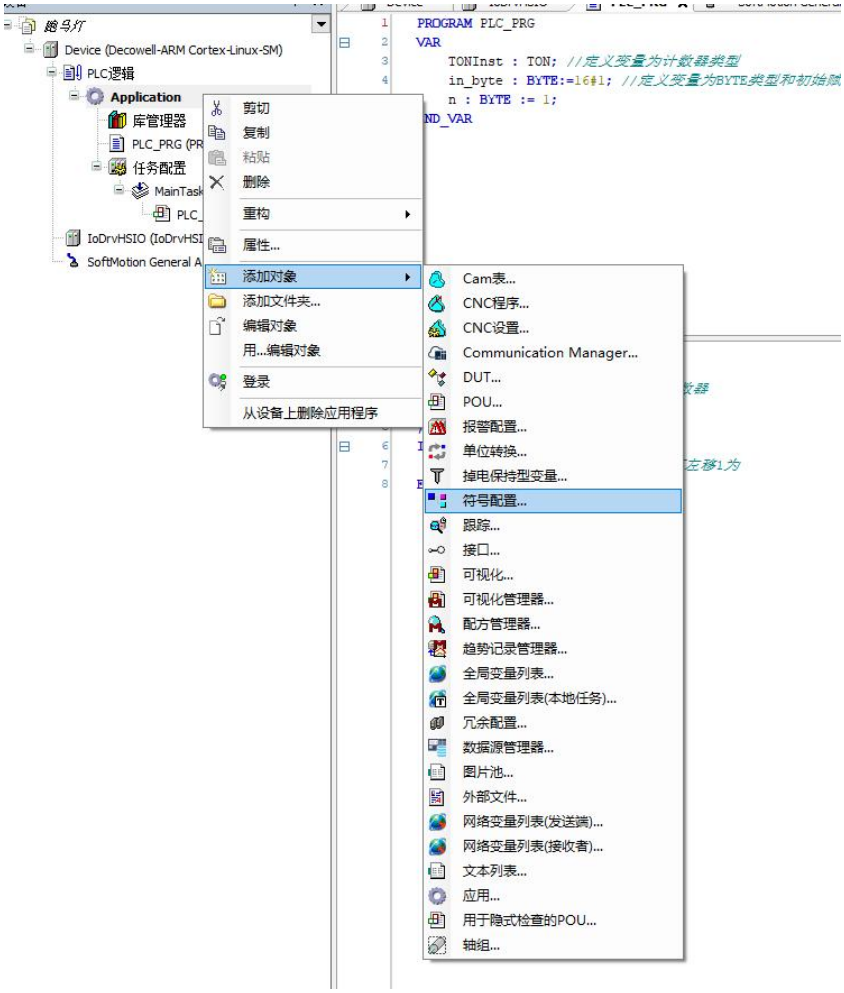
3.2 符号配置

使用符号配置为项目变量准备具有特定访问权限的符号。使用这些符号，可以从外部访问变量，比如 OPC 服务器。生成代码时，还会生成一个符号配置文件（工程目录下后缀为*.xml），命名方式为：<设备名称 name><设备名称<应用程序名称.xml，其中包含符号的说明。比如可以导入 HMI 中进行标签通信，访问变量。

3.2.1 添加符号配置

说明 生成符号配置要求编译没有任何错误。

选择工程树下 Application，右键选择添加对象，选择符号配置进行添加。



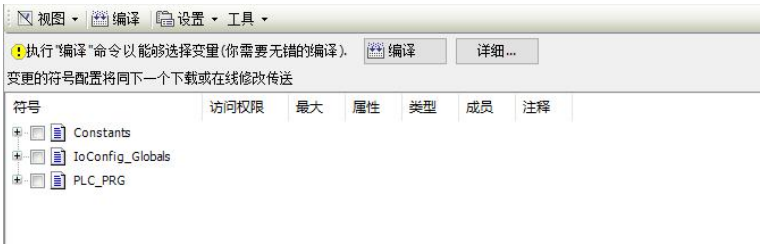
出现弹窗如下图所示：

勾选	说明
在XML中包含注释	导出的XML包含变量注释信息
支持OPC UA特征	支持OPC UA访问符号变量

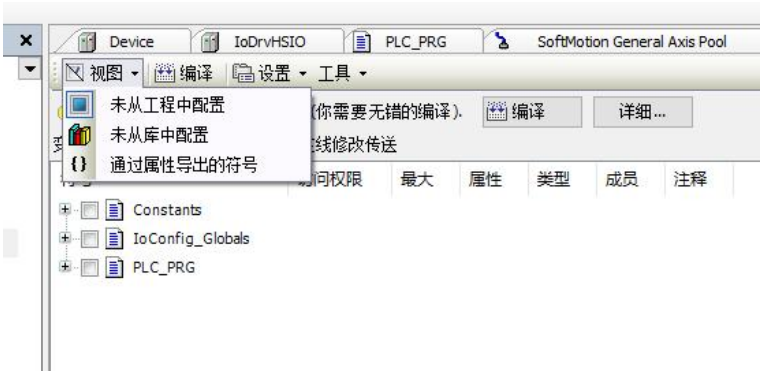
勾选	说明
兼容性布局	和类型成员定义偏移大小一致，如果类型成员不完全支持符号访问，偏移大小以实际编译偏移，中间会有空白
优化布局	根据选择的类型成员计算偏移，未选择的不计算偏移

符号配置界面介绍

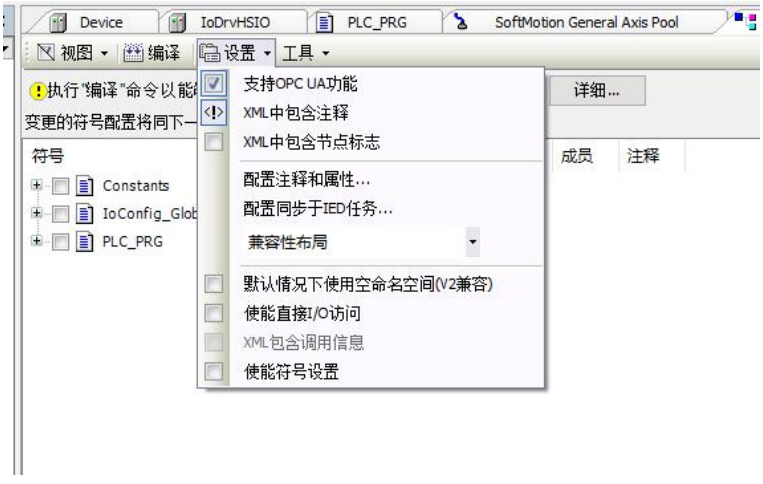
生成后符号配置界面如下图所示：



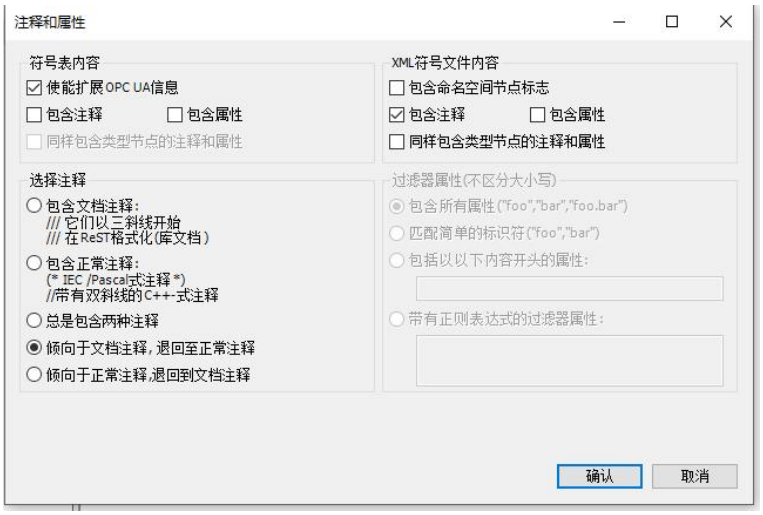
试图选项，可对下方符号标签进行显示。



设置选项，如下图所示。



配置注释和属性



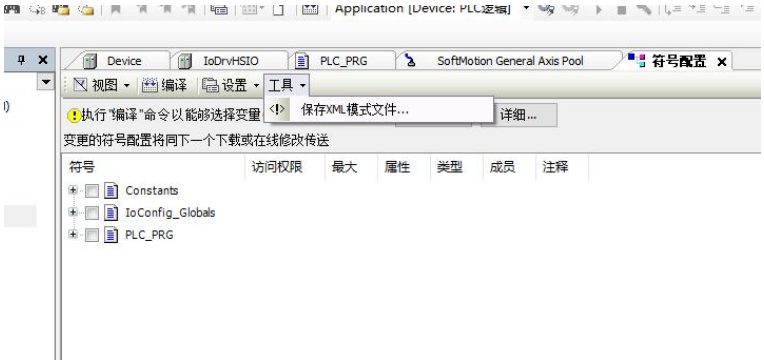
左上侧表示下载到 PLC 的数据，右上侧表示导出 XML 数据格式。

符号：一般表示变量，符号属性表示变量特性（Attribute 信息）。

注释格式：表示注释下载或者显示格式。

属性匹配：导出 XML 或者下载到 PLC 时，哪些属性信息是否包含。匹配规则三种为：所有属性只能是标识符格式，以特定字符串开始和正则表达式。

- **工具选项**。



导出 XML 数据模型，如果第三方解析离线符号，可以参考此数据模型。

3.2.1 符号配置过程样例

新建全局变量 A_0 、A_1 、A_2，并且在用户程序中至少应用一个变量，如下图所示。

说明如果单个全局变量表中任一变量都没有在用户程序中应用，则对应的变量表不会出现在符号配置中。

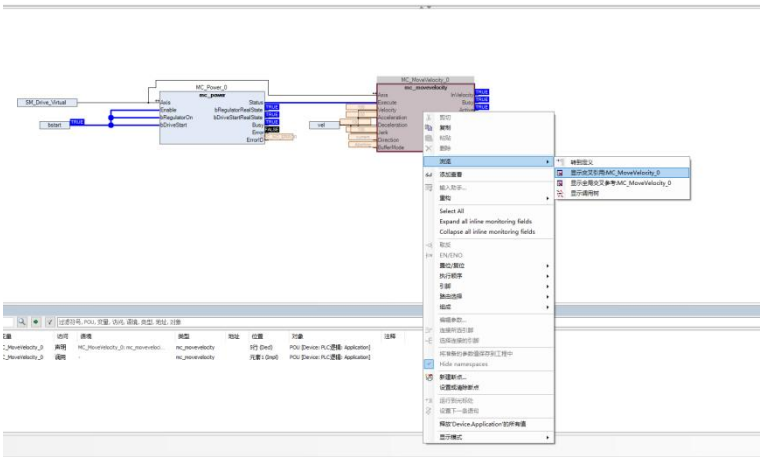
```
1  //{attribute 'qualified_only'}
2  VAR_GLOBAL
3      A_0 : ARRAY [0..9] OF INT;
4      A_1 : ARRAY [0..9] OF DINT;
5      A_2 : BOOL;
6  END_VAR
```

在 Application 中添加符号配置，并且勾选“在 XML 中包含注释”，点击上侧工具栏界面中选择-检查程序，对 应变量表及变量出现在符号配置中，对所需要配置的变量表进行勾选，并且配置好对应的访问权限（只读 、只写 、可读可写 ）后在上侧工具栏界面中选择-编译（生成代码）。

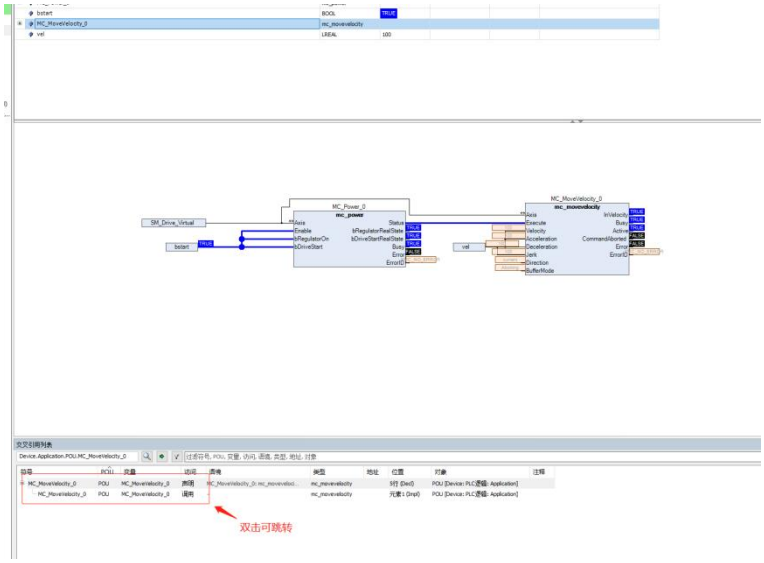
可在工程所在目录下找到生成后缀为 .xml 的文件。

3.3 交叉引用

通过“交叉引用”功能可以快速查找“目标对象”在整个工程中的调用位置。
找到需要“交叉引用”的对象，鼠标右键调出“交叉引用”。



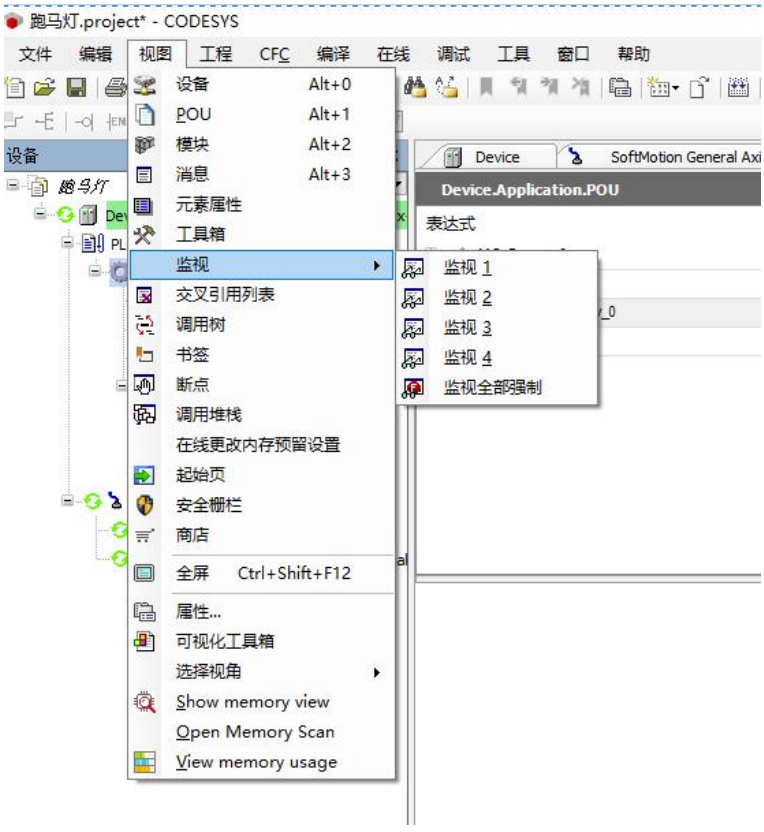
在工程下方“交叉引用表”查看“目标对象”在整个工程中的使用情况，双击交叉引用表中的信息可以跳转到工程中具体的使用位置。



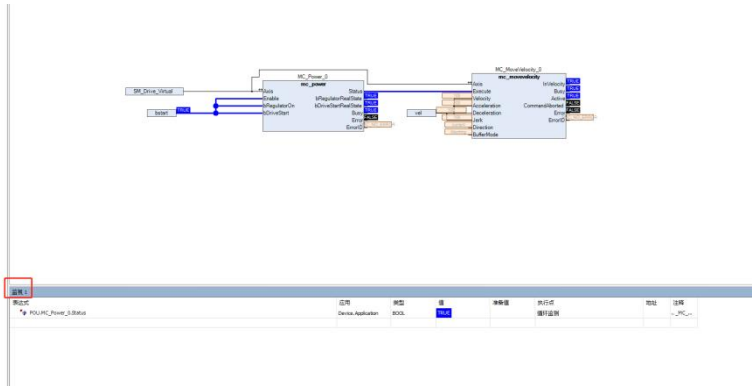
3.4 监控表

通过“监控表“功能可以对变量、地址进行监控，在程序运行中可以通过监控表查看监控变量的数据类型、当前值以及可以通过写入值为变量进行赋值。

在工具栏视图监视监控表中添加监视视图。



在工程栏下方添加想要监控的变量或者地址。



3.5 工程安全管理

3.5.1 加密工程文件

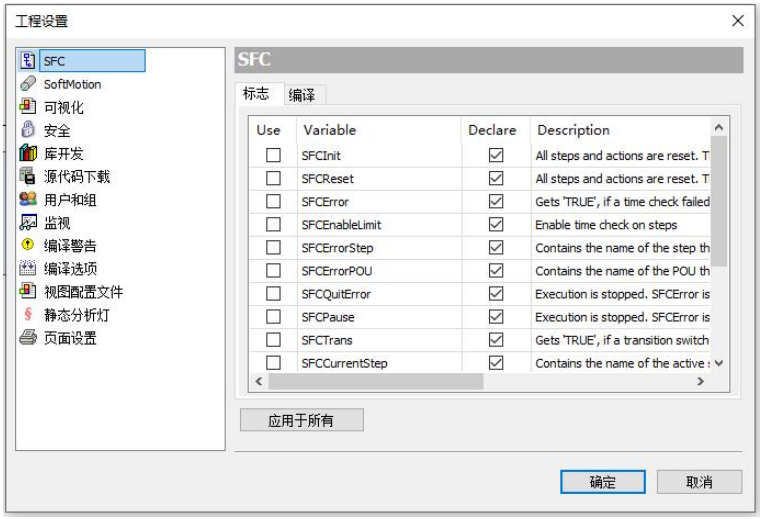
工程文件支持通过设置密码进行保护，防止未经允许的情况下被他人使用。设置工程文件密码后，用户在打 开此工程时需要输入密码进行校验，校验通过后才可正常使用该工程文件。

注 意

加密工程文件后，务必牢记该密码，如果遗忘密码，密码将无法找回，工程文件将会永久丢失。

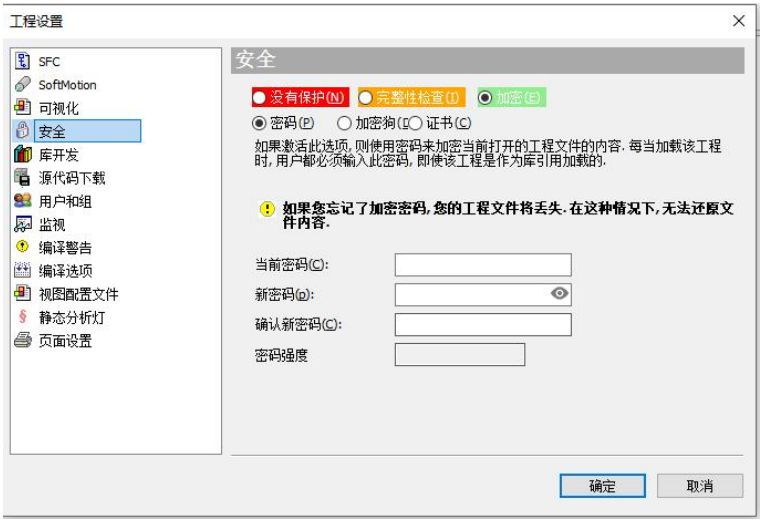
操作步骤

在菜单栏选择“工程 > 工程设置”，打开“工程设置”对话框。



单击“安全”，勾选“加密”复选框，勾选“密码“，完成加密工程文件。

说明 首次设置工程文件密码时，当前密码为空，无需输入当前密码



后续操作

重新打开该工程，打开“密码”对话框。

输入工程文件密码，单击“确定”。

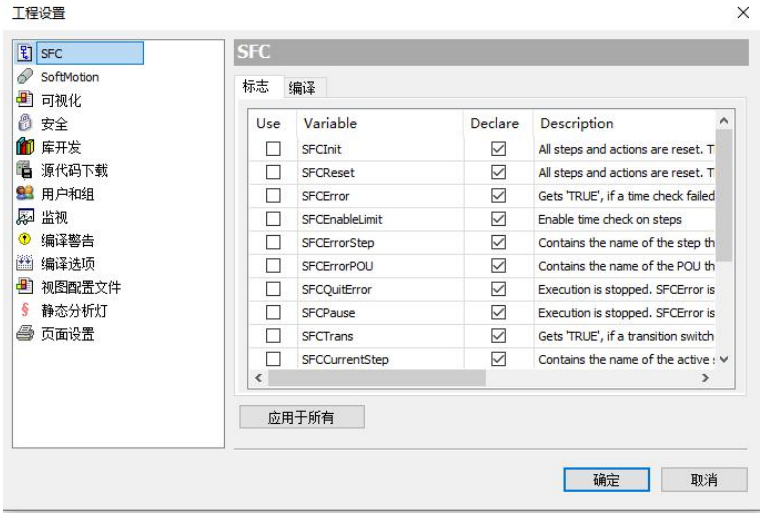
3.5.2 工程用户权限管理

该功能用于配置当前工程的用户权限，工程中不同功能的修改、浏览、添加或删除子项和移除操作可通过用户权限进行管理。

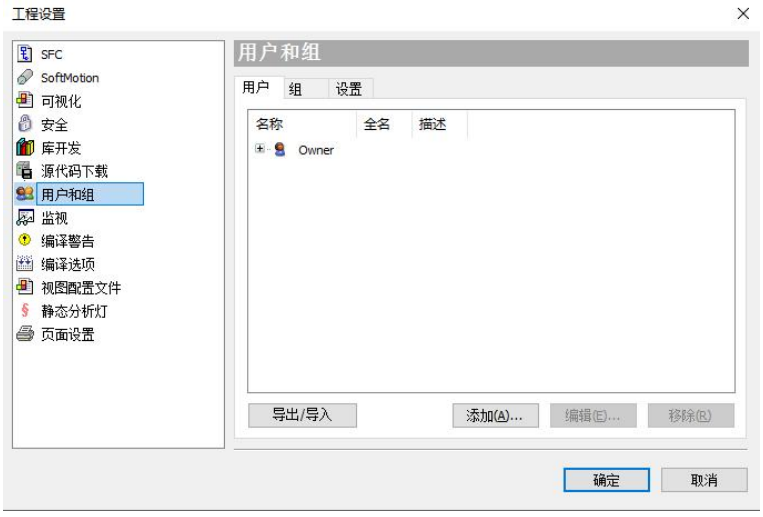
工程用户权限管理分为用户和组，系统默认两个组（Everyone 和 Owner）和一个用户（Owner），用户是组 的成员，组也可以是另一个组的成员。组 Everyone 和 Owner 不能被删除，只能重命名，所有新添加的用户都会自动分配到 Everyone 组中。用户 Owner 不能被删除，对应初始密码为空，可以重命名和修改密码，强烈建 议修改 Owner 的初始密码，否则任何用户都可以越过其他帐户使用 Owner 帐户登录，使分配的权限不起作用。

添加用户和组完成后，需要对设备树节点功能进行分配权限，权限以组为单位进行分配，默认分配 Everyone 组，缺省权限为授权；用户登录工程时将根据分配的权限进行管理工程。

在菜单栏选择“工程 > 工程设置”，打开“工程设置”对话框。



单击“用户和组”，进入“用户和组”管理界面。



添加用户

在“用户和组 > 用户”界面单击“添加”，填写用户相关信息，单击“确定”。

说明

- 新用户默认为“Everyone”组的成员，如需加到其他组，勾选组前复选框。
- 编辑用户：在“用户和组 > 用户”界面选中用户名，单击“编辑”，在打开的对话框中修改用户信息，单击“确定”。
- 删除用户：在“用户和组 > 用户”界面选中用户名，单击“删除”。

（可选）“用户名”输入“Owner”（初始密码为空），单击“登录”。

说明

工程打开首次添加/编辑/删除用户需要由“Owner”组用户授权（系统默认创建 Owner 用户），授权通过后无需再进行授权。

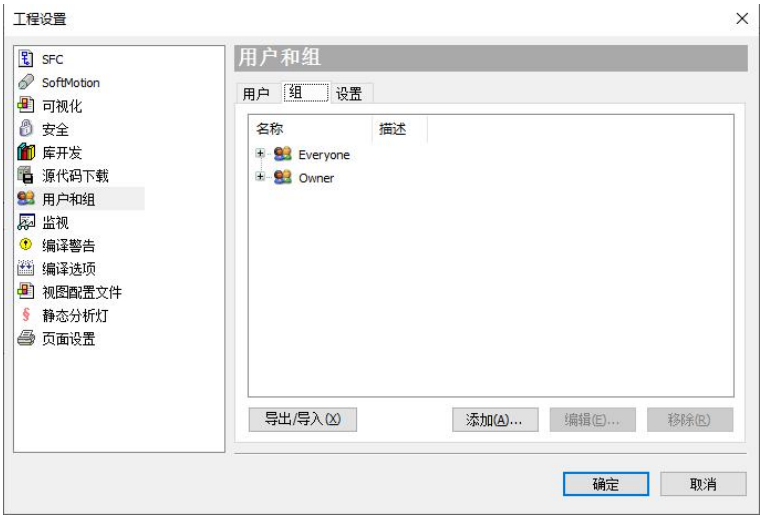
导出/导入用户

● 导出用户：在“用户和组 > 用户”界面单击“输出/输入”，选择“输出用户和群组”，在打开的界面中 选择保存至本地的路径，单击“保存”。

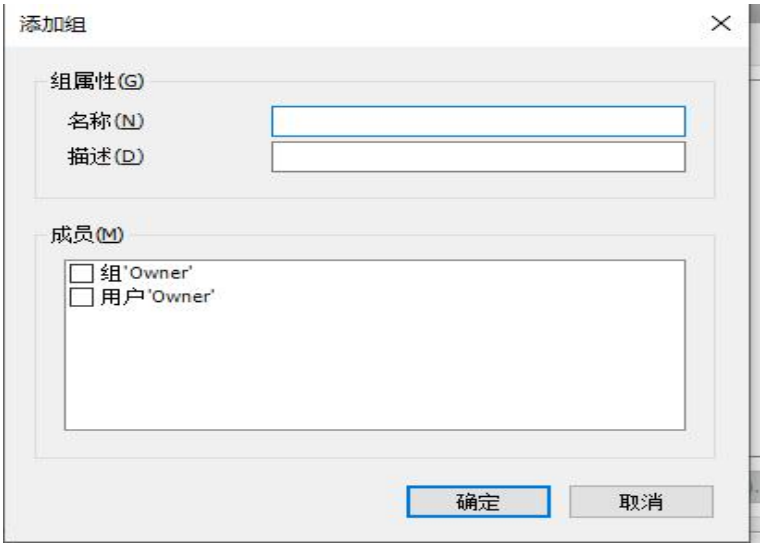
● 导入用户：在“用户和组 > 用户”界面单击“输出/输入”，选择“输入用户以及群组”，在打开的界面 中选择保存在本地路径的用户保存文件，单击“打开”。

添加组

在“用户和组”界面单击“组”，切换至组管理界面。



在“用户和组 > 组”界面单击“添加”，填写组相关信息，单击“确定”。



说明

● 系统默认创建“Everyone”组和“Owner”组，一个组也可以是另一个组的成员。

● 编辑组：在“用户和组 > 组”界面选中组名，单击“编辑”，在打开的对话框中修改组信息，单击“确定”。

● 删除组：在“用户和组 > 组”界面选中组名，单击“删除”。

（可选）“用户名”输入“Owner”（初始密码为空），单击“登录”。

说明

工程打开首次添加/编辑/删除组需要由“Owner”组用户授权（系统默认创建 Owner 用户），授权通过后无需再进行授权。

导出/导入组

● 导出用户：在“用户和组 > 组”界面单击“输出/输入”，选择“输出用户和群组”，在打开的界面中选 择保存至本地的路径，单击“保存”。

● 导入用户：在“用户和组 > 组”界面单击“输出/输入”，选择“输入用户以及群组”，在打开的界面中 选择保存在本地路径的用户保存文件，单击“打开”。

设置

在“用户和组”界面单击“设置”，切换至设置界面。

填写设置相关信息，单击“确定”。

参数名称	如何理解	如何设置
最大授权数	用户登录进行授权时，尝试使用密码进行登录授权的次数。超过该次数，则该用户帐户将被停用。	根据所需设置 缺省值：3
在不活动之后，自动退出	如果在此处指定的时间段（分钟）内未通过鼠标或键盘进行任何用户操作，该用户将自动退出。	根据所需设置 缺省值：10

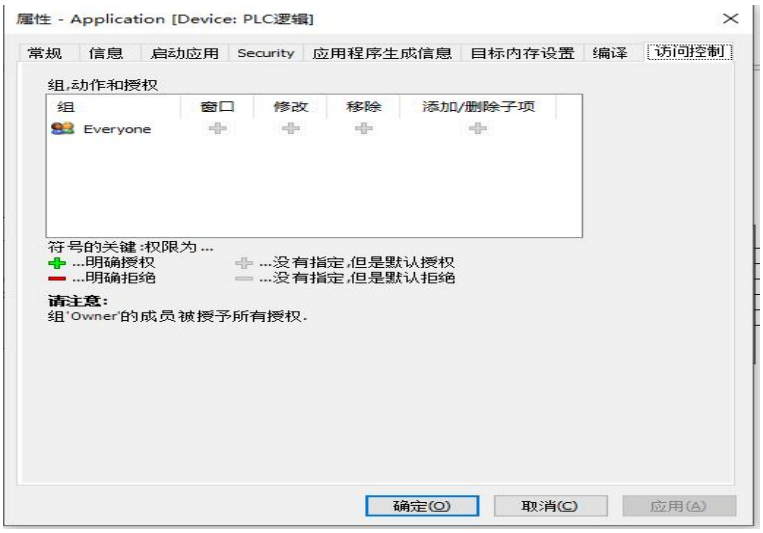
（可选）“用户名”输入“Owner”（初始密码为空），单击“登录”。

说明

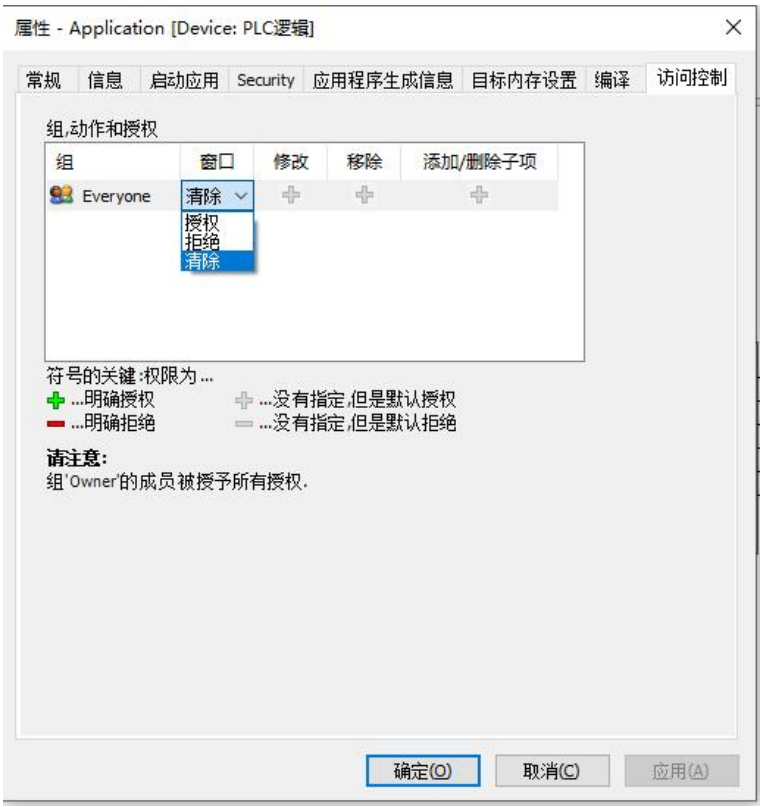
工程打开首次管理设置界面时需要由“Owner”组用户授权（系统默认创建 Owner 用户），授权通过后无需再进行授权。

分配权限

在左侧设备树中右键单击待授权的节点，以“Application”节点为例，选择“属性”，打开“属性”对话框。



单击“访问控制”页签，选择分配权限的组，在不同动作列双击“+”进行选择“授权”、“拒绝”或“清除”，单击“确定”，完成分配权限。



用户登录/退出工程

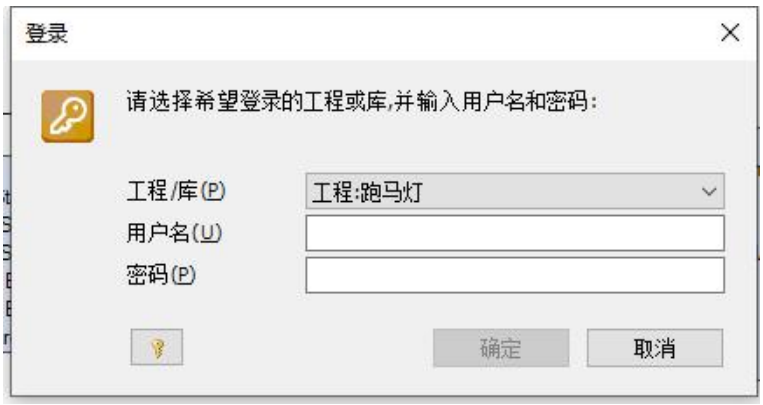
添加用户和组，以及分配权限完成后，用户登录工程才能使用相应节点功能。

用户登录/退出工程包含两种方式：

- 通过菜单栏

■ 用户登录工程

1. 在菜单栏选择“工程 > 用户管理 > 用户登录”，打开“登录”对话框。



2. 输入用户名和密码，单击“确定”，登录工程。

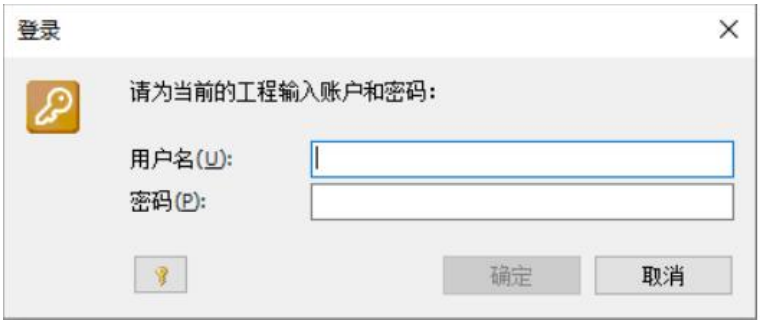
■ 用户退出工程

在菜单栏选择“工程 > 用户管理 > 用户退出”，退出工程。

- 通过状态栏

■ 用户登录工程

1. 在状态栏双击“当前用户：xx”区域，打开“登录”对话框。



2. 输入用户名和密码，单击“确定”，登录工程。

■ 用户退出工程

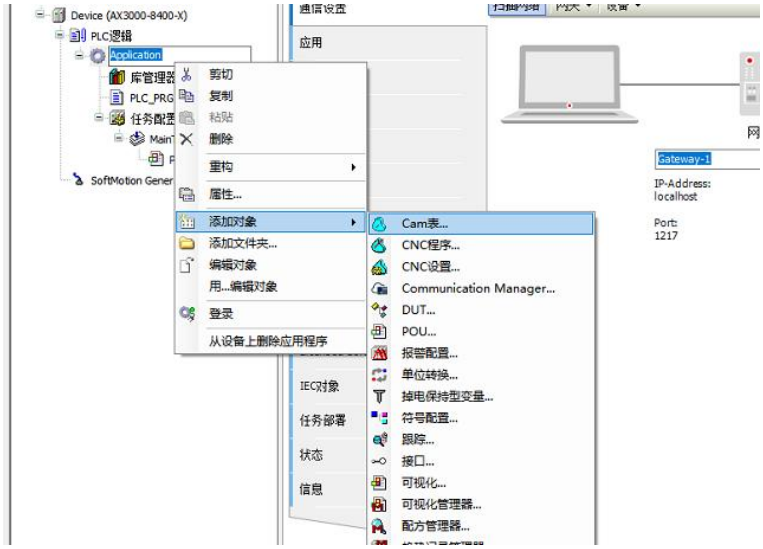
在状态栏双击“当前用户：xx”区域，单击“退出”，退出工程。

Application 添加对象

Application 支持添加对象功能，支持的功能如下，未列出的功能表示不支持。

- CAM 表
- DUT
- 程序组织单元
- 掉电保持变量
- 符号配置
- 跟踪
- 界面
- 全局变量列表
- 应用
- 用于隐含检查的 POU

在左侧设备树中右键单击“Application”，选择“添加对象 > XX”，例如选择“添加对象 > cam 表”。



CAM 表

CAM 表详细介绍，具体请参见《在线帮助》中“CoDeSys 编程系统 > SoftMotion > 对象编辑器 > Cam 编辑 器”章节。

DUT

DUT 详细介绍，具体请参见《在线帮助》中“CoDeSys 编程系统 >CODESYS Development System> 参考，用户接口 > 对象 > “DUT”对象”章节。

程序组织单元

程序组织单元详细介绍，具体请参见《在线帮助》中“CoDeSys 编程系统 >CODESYS Development System> 参考,用户接口 > 对象 > 对象‘POU’”章节。

符号配置 符号配置详细介绍，具体请参见《在线帮助》中“CoDeSys 编程系统 > CODESYS Development System > 参 考,用户接口 > 对象 > “符号配置”对象”章节。

跟踪

跟踪详细介绍，具体请参见《在线帮助》中“CoDeSys 编程系统 >CODESYS Development System> 参考，用户接口 > 对象 > “跟踪”对象”章节。

界面

界面详细介绍，具体请参见《在线帮助》中“CoDeSys 编程系统 >CODESYS Development System> 参考，用户接口 > 对象 > 对象‘POU’ > “接口”对象”章节。

全局变量列表

全局变量列表详细介绍，具体请参见《在线帮助》中“CoDeSys 编程系统 >CODESYS Development System> 参考,用户接口 > 对象 > “GVL”对象 - 全局变量列表”章节。

应用

应用详细介绍，具体请参见《在线帮助》中“CoDeSys 编程系统 >CODESYS Development System> 参考，用户接口 > 对象 > “应用”对象”章节。

用于隐含检查的 POU

用于隐含检查的 POU 详细介绍，具体请参见《在线帮助》中“CoDeSys 编程系统 > CODESYS Development System > 参考,用户接口 > 对象 > 对象‘隐式检查 POU’”章节。

4. 网络配置

4.1 设备组态

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

4.1.1 网络组态

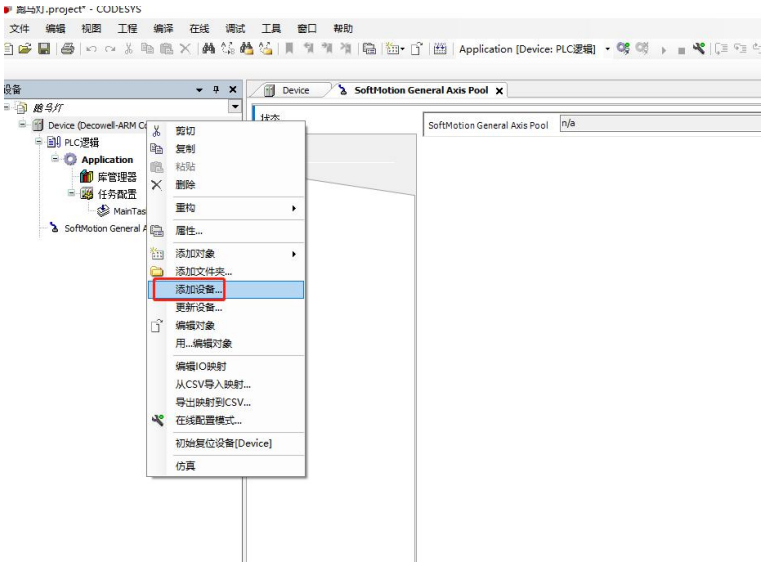
从总线型的网络拓扑视角设计，是设备组态的入口。

4.1.2 硬件组态

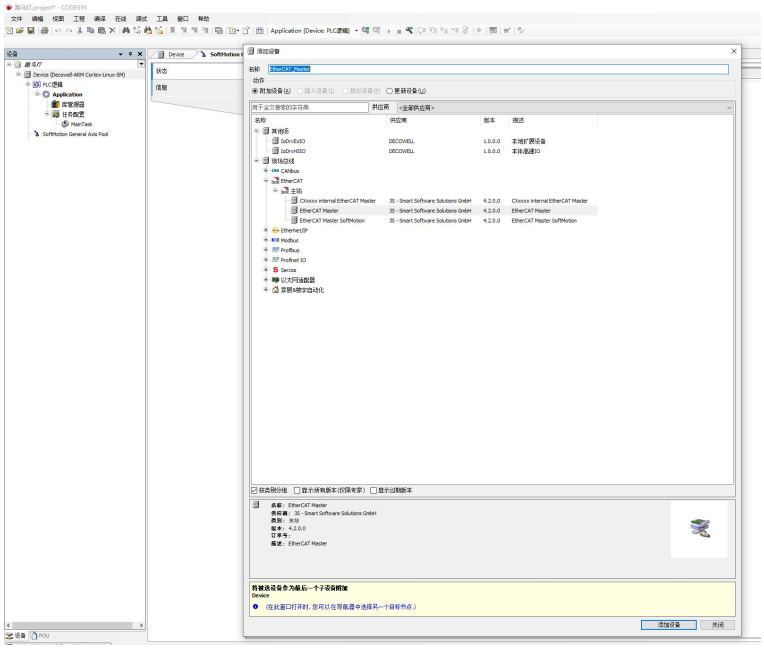
添加 AX3000 扩展 IO 模块。

网络组态（EtherCat 伺服轴）

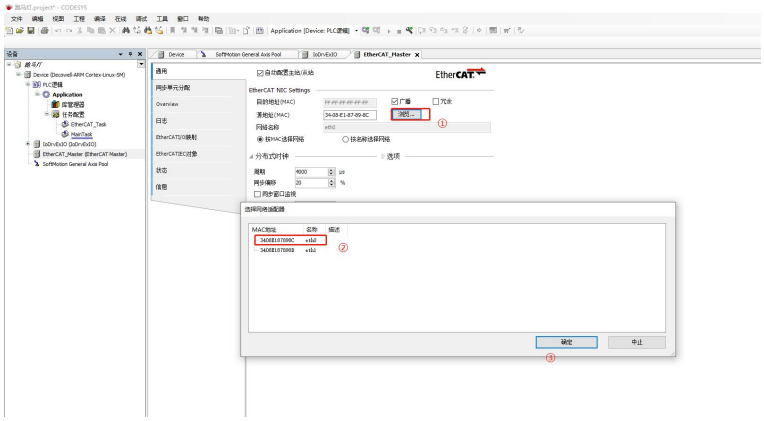
新建一个 CodeSys 工程后，在左侧设备树中双击“添加设备”的节点， 如下图所示。



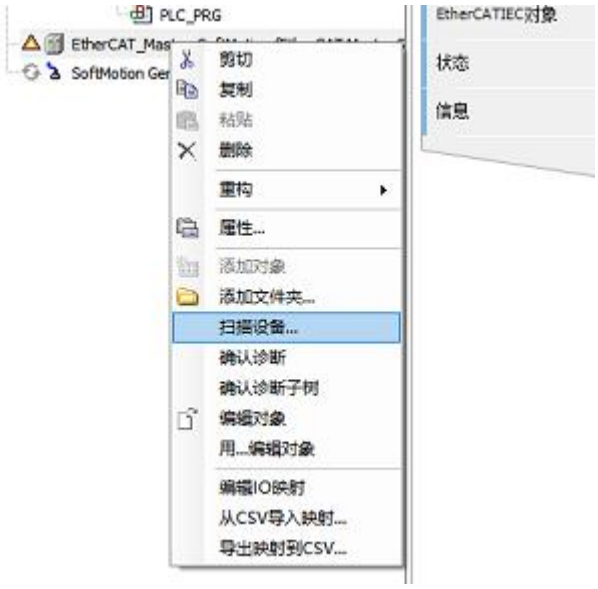
主站选择 EtherCat Master



选择 EtherCat 源地址

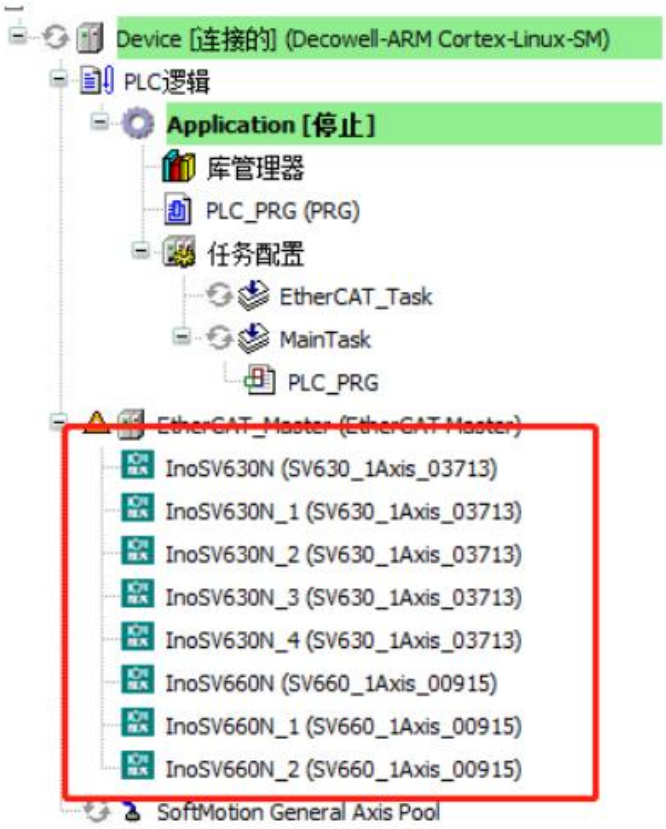


CodeSys 登录到 AX3000，右击 EtherCat_Master 选择扫描设备



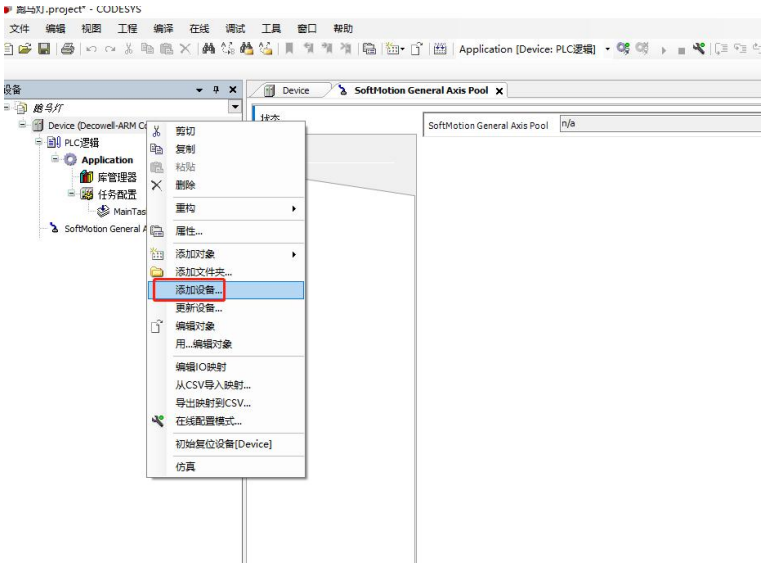
等待一分钟后，页面弹出扫描设备框

点击复制所有设备到工程

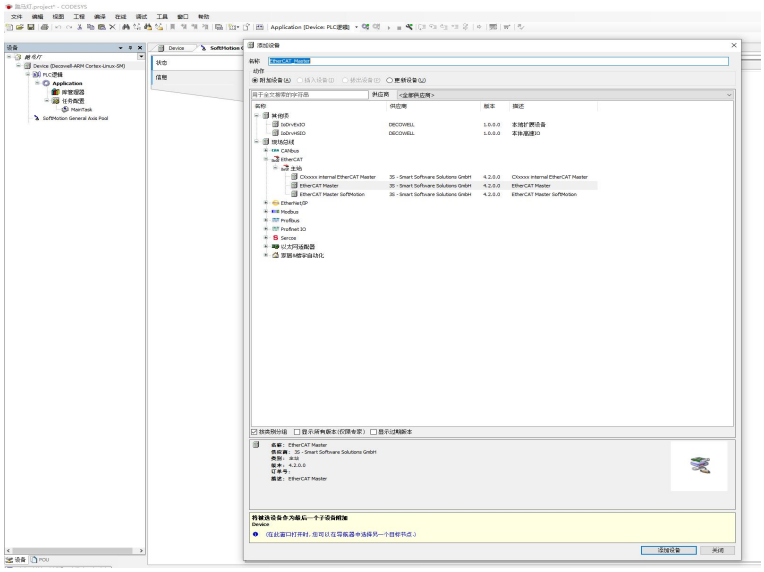


网络组态（EtherCat 扩展 IO）

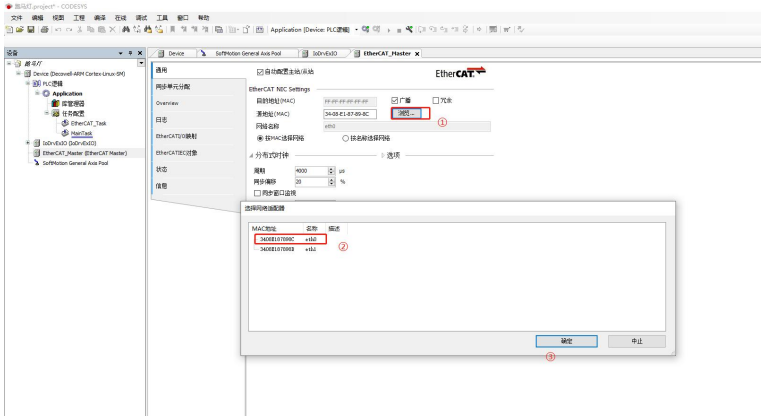
新建一个 CodeSys 工程后，在左侧设备树中双击“添加设备”的节点， 如下图所示。



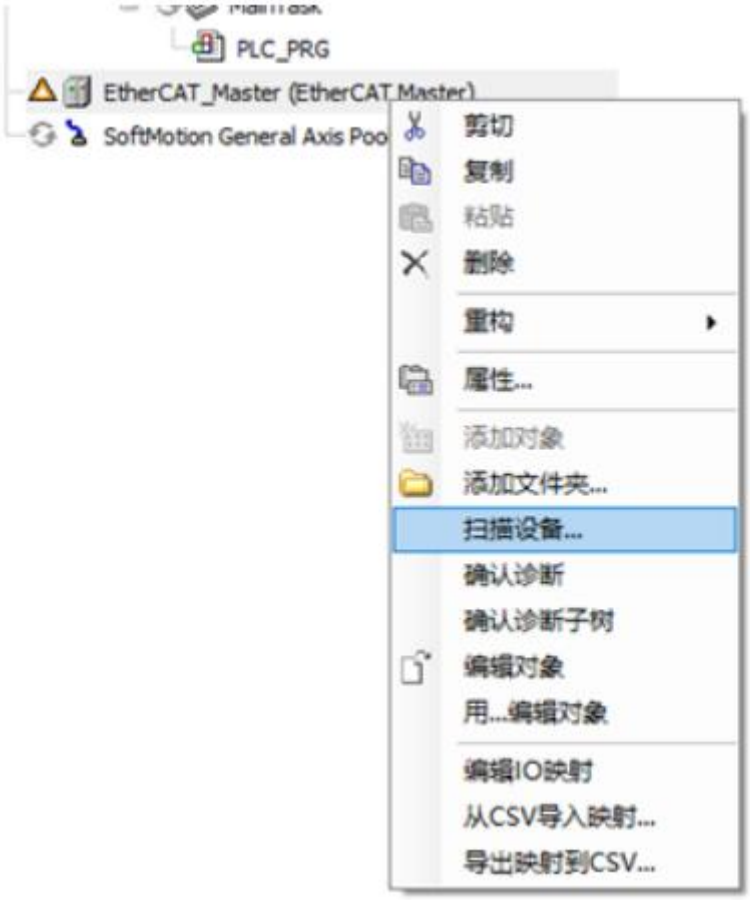
主站选择 EtherCat Master



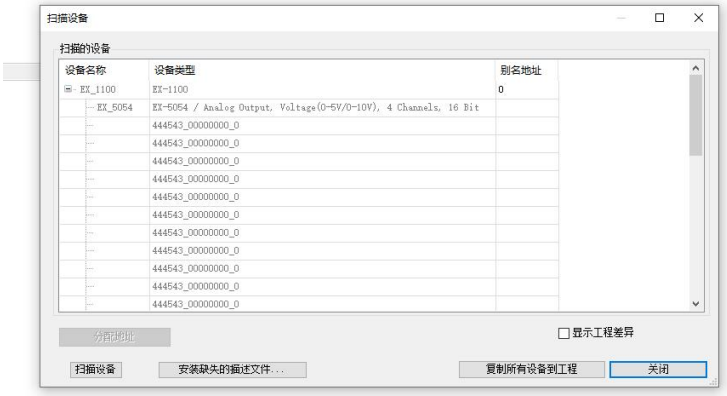
选择 EtherCat 源地址



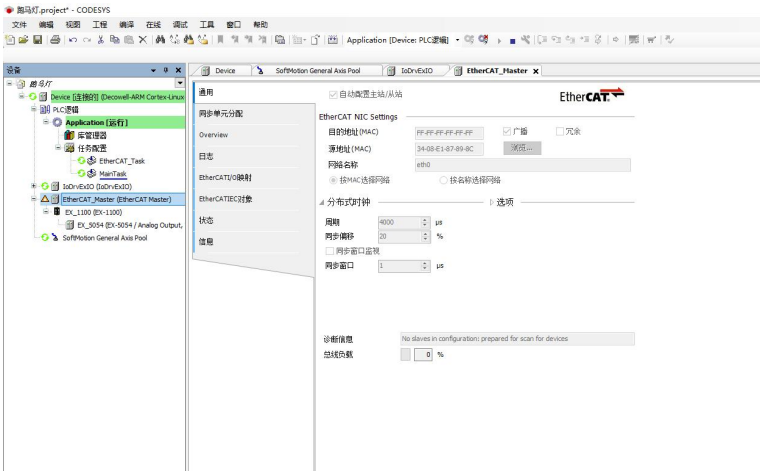
CodeSys 登录到 AX3000，右击 EtherCat_Master 选择扫描设备



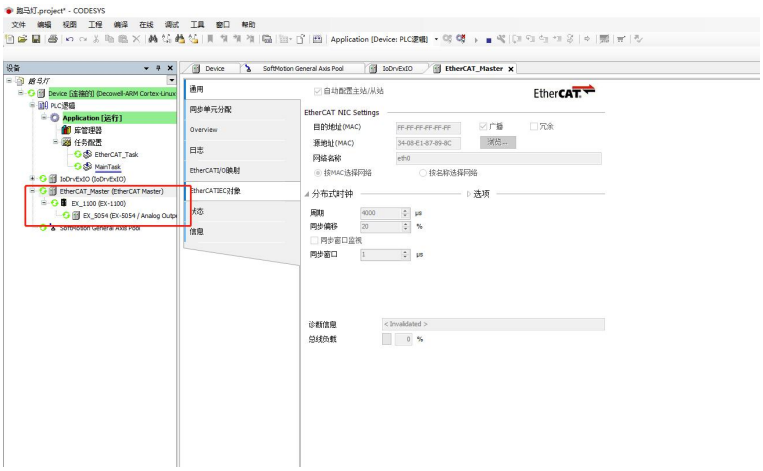
等待一分钟后，页面弹出扫描设备框



点击复制所有设备到工程

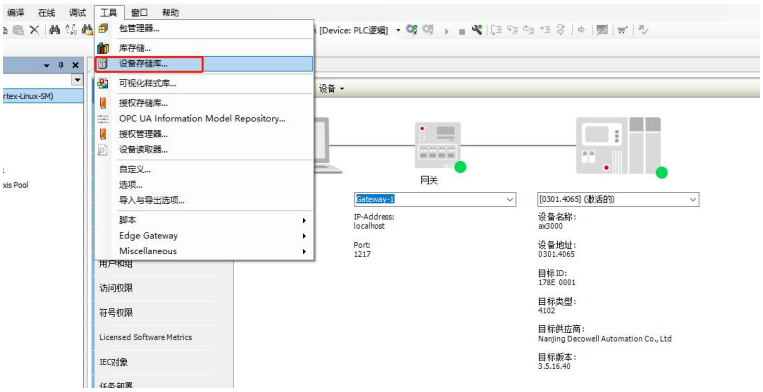


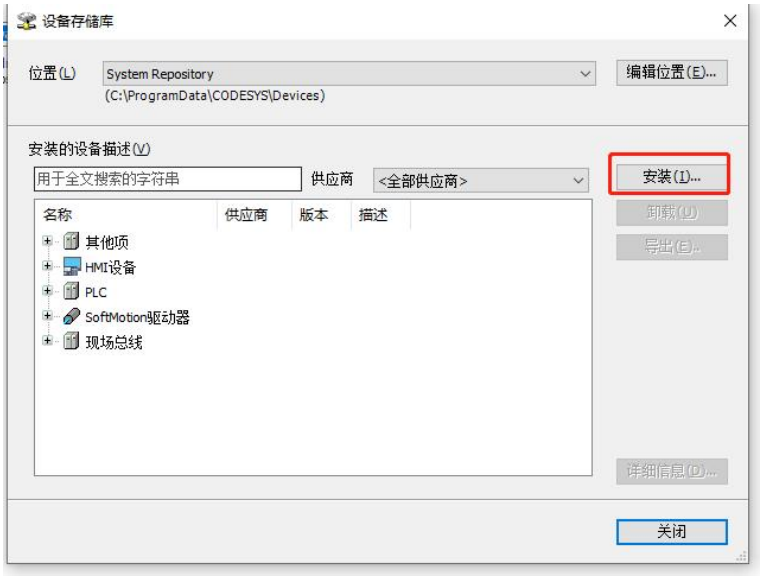
CodeSys 退出登录 AX3000，然后重新登录，可以成功组态到扩展 IO 设备



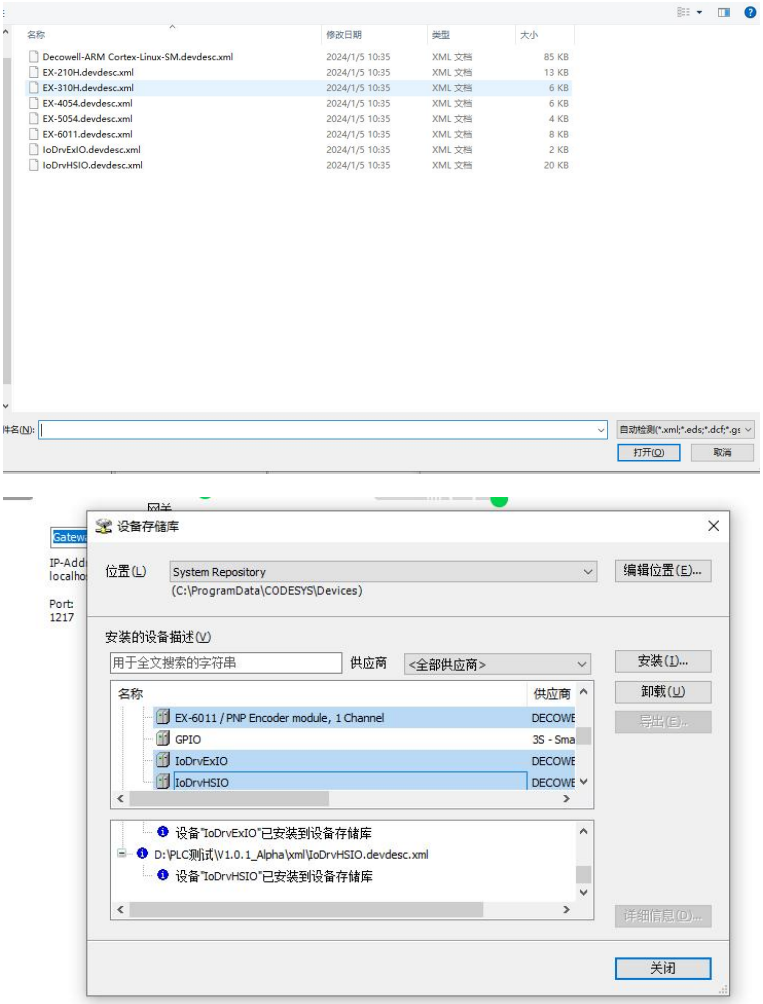
4.1.3 本体 IO 组态

在菜单栏“工具-设备存储库“，安装设备描述文件

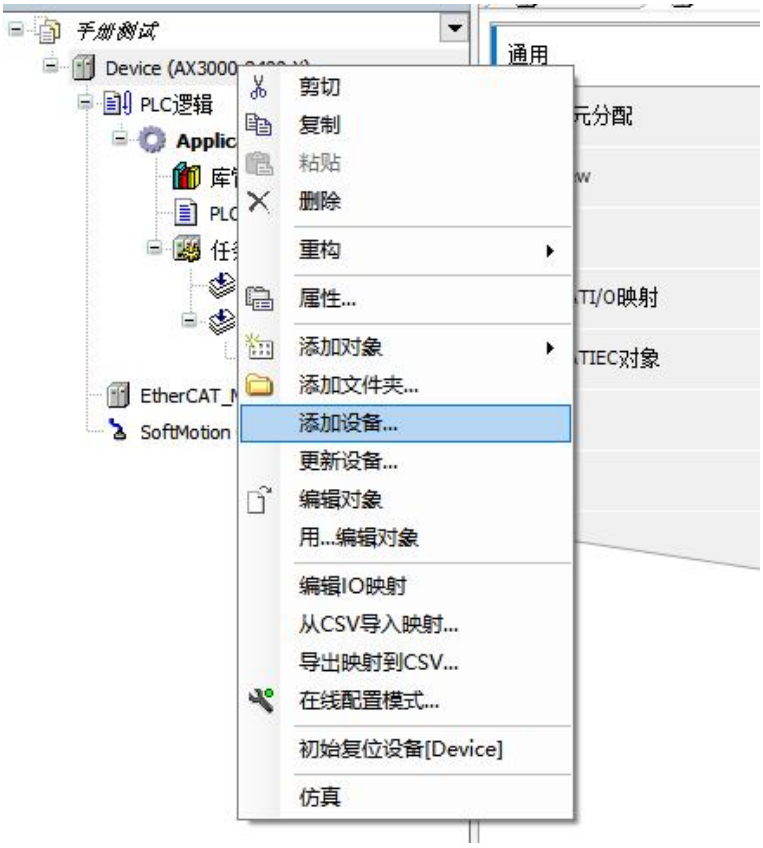




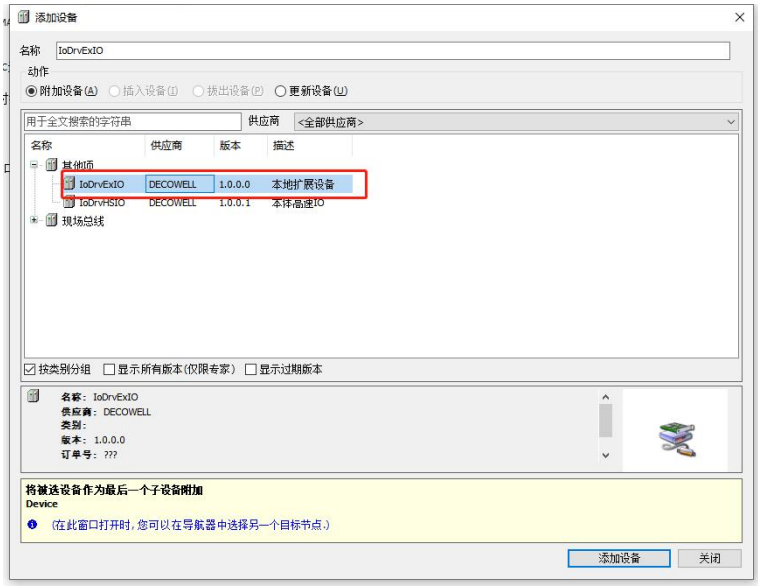
选择设备描述文件，点击安装



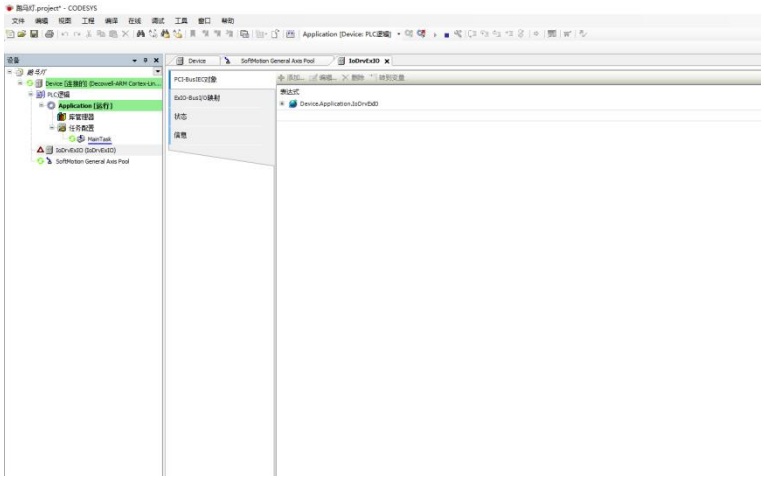
右击 Device（AX3000-8400-X），选择添加设备



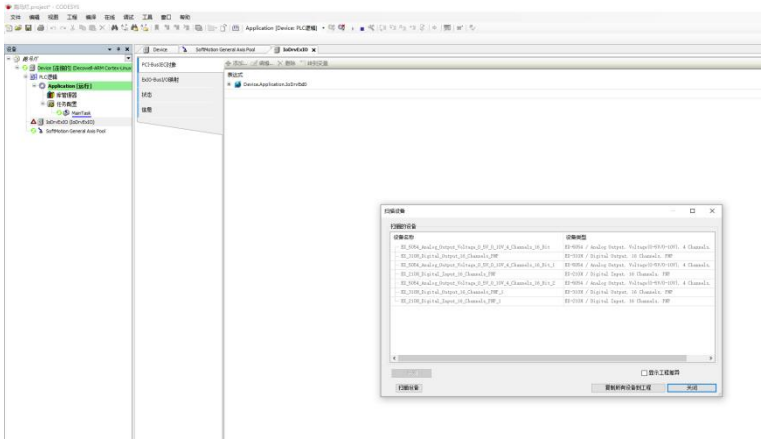
在其他项中选择本地扩展设备，然后点击添加设备



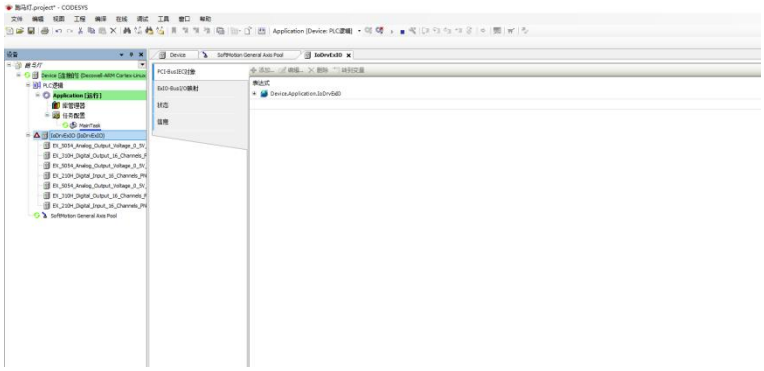
关闭添加设备窗口，使用 CodeSys 登录 AX3000



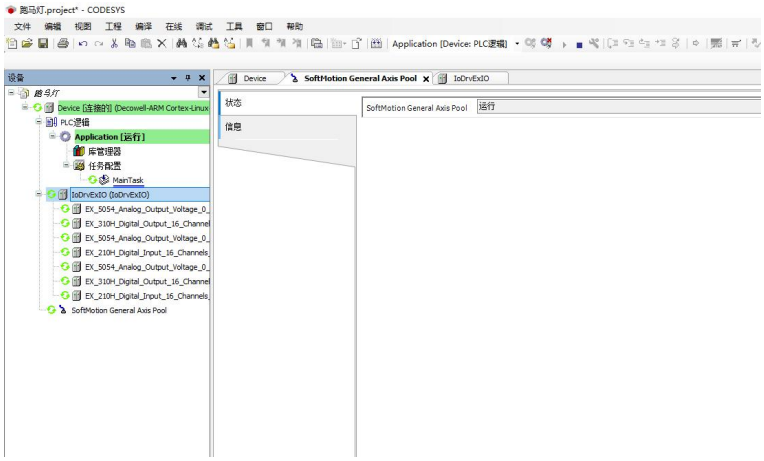
右击 IoDrvExIO，选择扫描设备



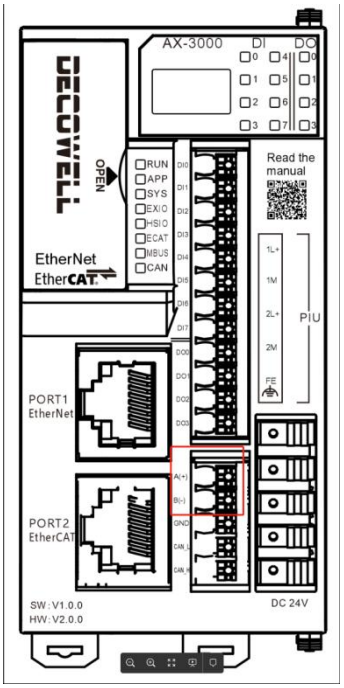
可以查看到有上述设备和 AX3000 进行了硬件组态，然后点击复制所有设备到工程



CodeSys 重新登录 AX3000，然后运行程序，IO 设备组态成功。



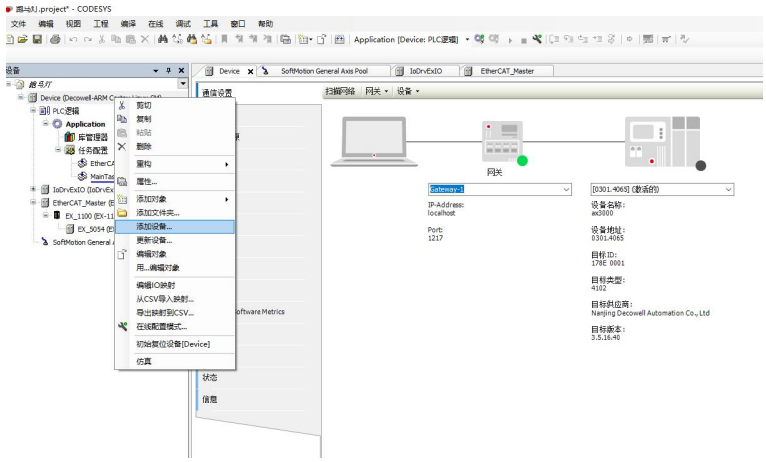
4.2 ModBus RTU



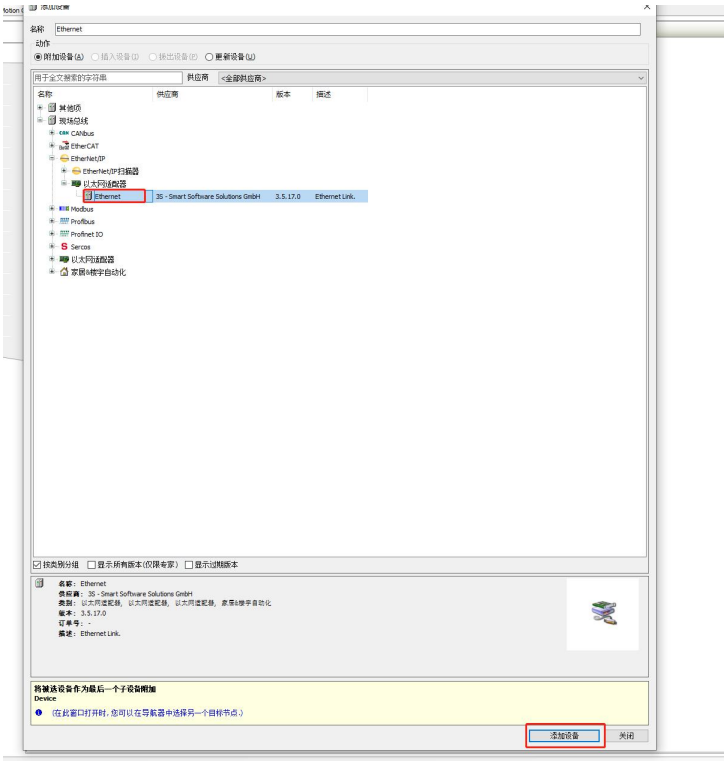
使用串口工具设备连接 AX3000 上的 A(+)和 B(-),可以使用 ModBus RTU 功能,让 AX3000 作为 ModBus_Master 或者 ModBus_Slave。

4.3 ModBus TCP

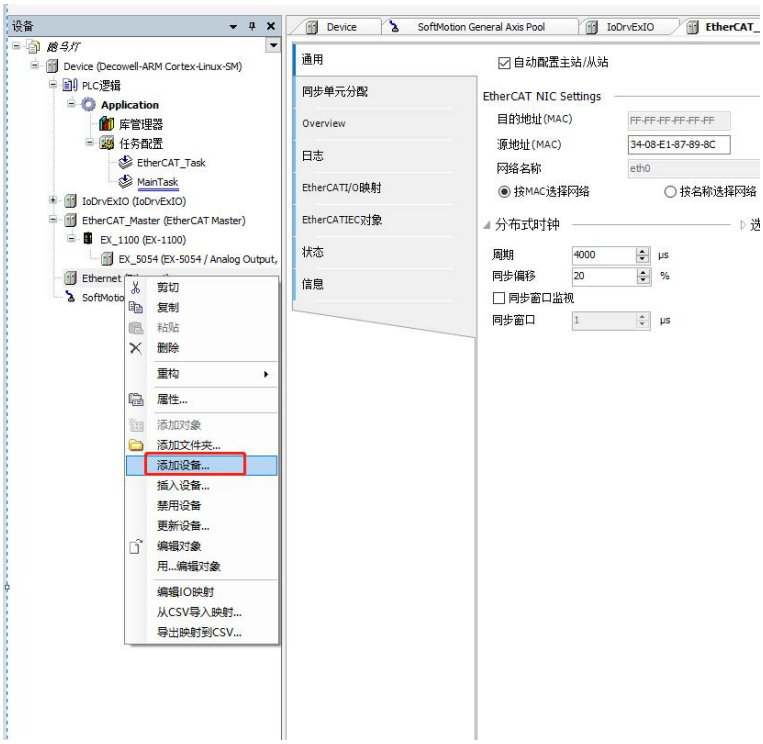
右击 Device（Decowell-ARM Cortex-Linux-SM），点击添加设备



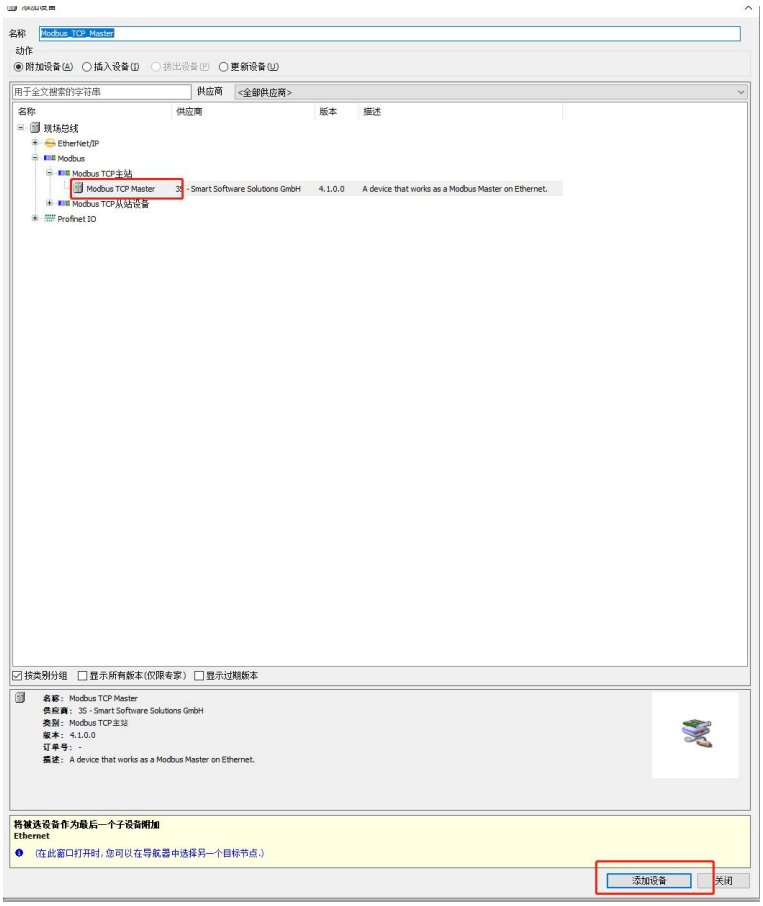
选择现场总线-EtherNet/IP-以太网适配器-Ethernet，然后点击添加设备



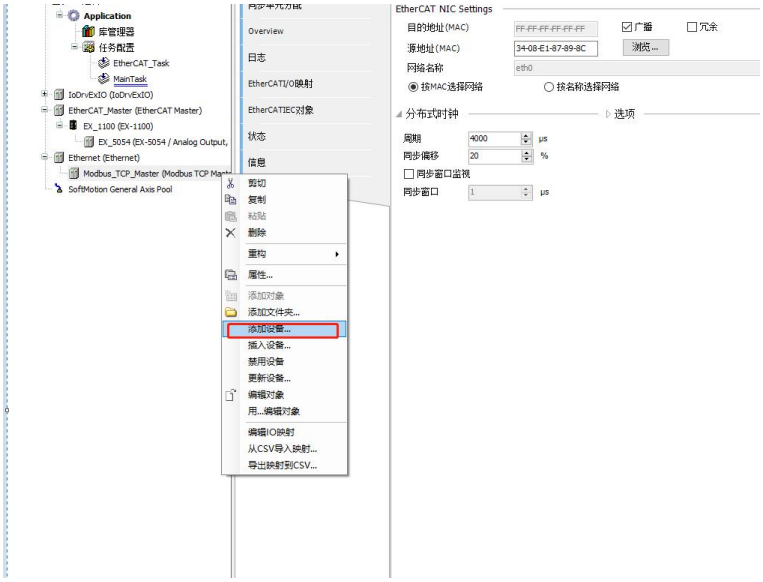
右击 Ethernet，点击添加设备



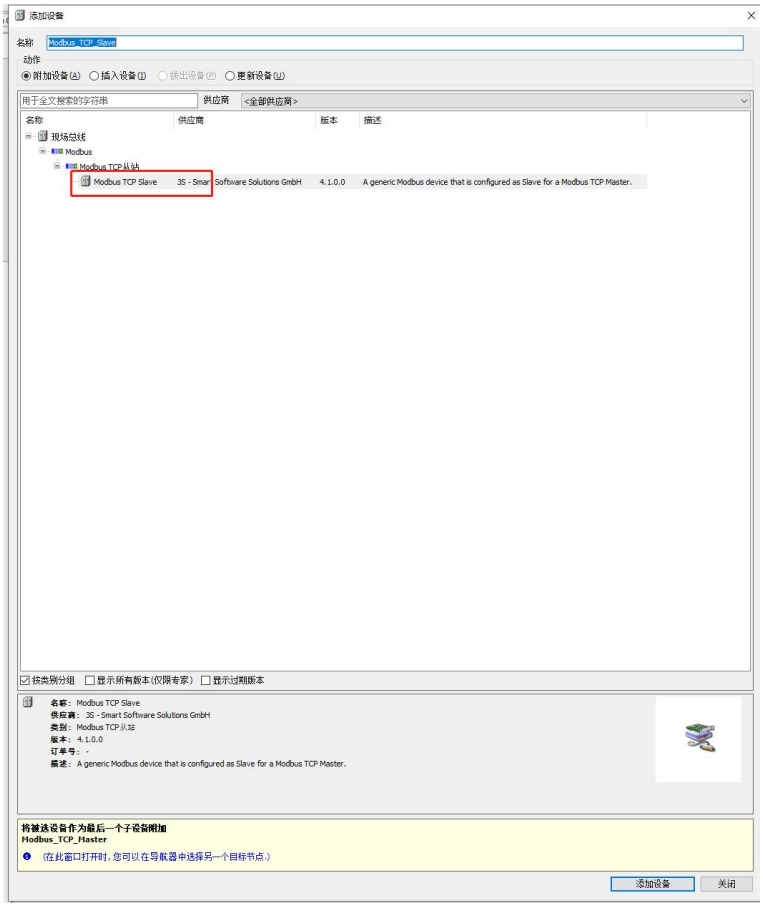
选择现场总线-Modbus-Modbus TCP 主站，选择 Modbus TCP Master，点击添加设备



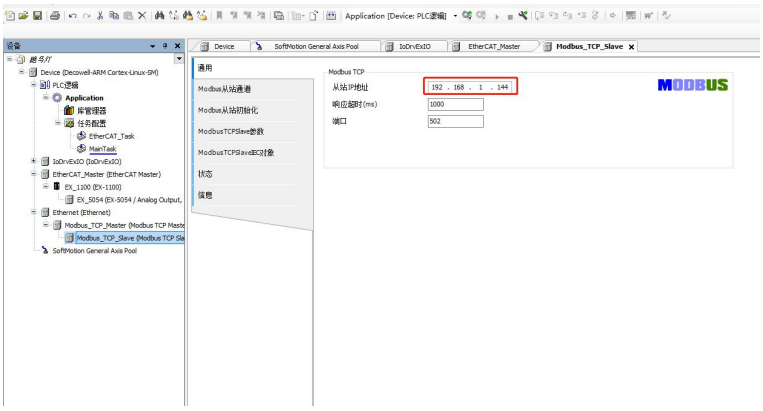
右击 Modbus_TCP_Master，选择添加设备



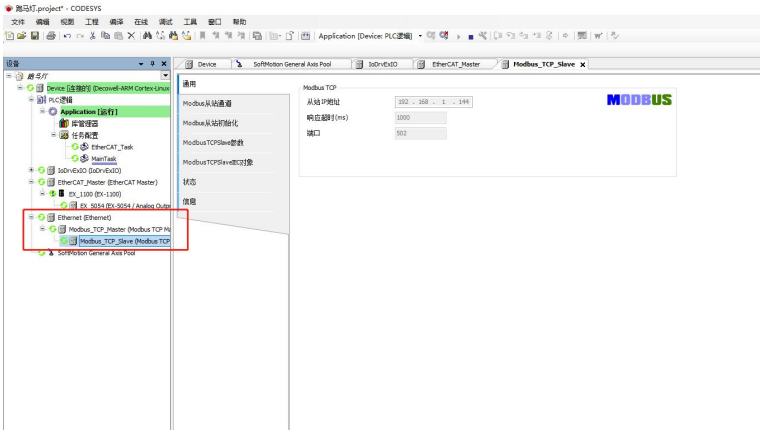
选择 Modbus TCP Slave，然后点击添加设备



在 Modbus TCP Slave 的通用中输入本地 IP 地址



点击登录 AX3000，然后启动，运行本地的 Modbus TCP Slave，可以成功组态



5. 编程基础

5.1 概述

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

与其他高级语言类似，CodeSys 也有常量和变量的概念。所谓常量就是数值不变的数；变量是由用户定义的标识符。变量的存储位置可由用户指定为%I 区、%Q 区、%M 区的特定地址，亦可不指定地址，由系统自行分配，用户不需要关注这些变量的存储位置。

5.2 直接地址

5.2.1 定义语法

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

语法：%< 存储器区前缀>< 大小前缀>< 数字>|.< 数字>

● < 存储器区前缀 >：编程系统支持以下存储区前缀

1. I：输入，物理输入，“传感器”
2. Q：输出，物理输出，“执行器”
3. M：存储位置

● < 大小前缀 >：编程系统支持以下大小前缀

1. X：bit，一位
2. B：Byte，一个字节
3. W：Word，一个字
4. D：Double Word，四个字节（双字）

● < 数字 >|.< 数字 >

第一个数字是存储区前缀的偏移地址，如果定义 BOOL 类型变量，需使用< 数字>.< 数字> 格式，“.”后 的数字是偏移地址的偏移位数。

示例：

%QX7.5 输出区域偏移 7 个字节的第六位（bit5）

%QB17 输出区域偏移 17 个字节

%IW215 输入区域偏移 215 个字

%MD48 内存区域偏移 48 个双字

iVar AT %IW10: WORD;//iVar 变量是字类型，映射到输入区域偏移 10 字的位置

说明

- 大小前缀为 X 类型变量代表的数据类型为 BOOL 型，偏移地址应具体到位。
- 大小前缀和数据类型是匹配的，大小前缀为 B 类型的变量应声明为一个字节的数据类型，如 BYTE，SINT，USINT；大 小前缀为 W 类型的变量应声明为一个字的数据类型，如 WORD，INT，UINT；大小前缀为 D 类型的变量应声明为一个双 字的数据类型，如 DWORD，DINT，UDINT。

5.2.2 PLC 直接地址存储区域

不同 PLC 提供的直接存储区域不同。对于 PLC 数据，%I、%Q 区地址不能掉电保存，对于%M 区可以掉电保存。AX3000 提供 128KB（Byte）的输入区域（I 区），128KB（Byte）输出区域（Q 区）和 512KB 存储区域（M 区），其中存储区域中的前 480KB 用户可以直接使用，后 32K 为系统使用区域（主要用作软元件），用户不要直接使用。编程时，用户可以直接访问地址，也可以定义变量后把变量映射到地址间接访问。存储区域定义及使用的地址范围如下表。

区域	用途	大小	地址范围
I区 (%I) 128KB	用户使用区域	64KWords	%IW0-%IW65535
Q区 (%Q) 128KB	用户使用区域	64KWords	%QW0-%QW65535
M区 (%M) 512KB	用户使用区域	240KWords	%MW0-%MW245759
	SD元件	10000Words	%MW245760-%MW255759
	SM元件	10000Bytes	%MB511520-%MB521519
	保留	2768Bytes	%MB521520-%MB524287

5.3 变量

5.3.1 概述

变量可以在 POU 的定义部分或者通过自动声明对话框定义，也可以在 DUT 或者 GVL 编辑器定义，通过变量类 型关键字来标识变量类型，例如通过 VAR 和 END_VAR 来标识它之间定义的变量为本地变量。
变量类型包括本地变量(VAR)，输入变量(VAR_INPUT)，输出变量(VAR_OUTPUT)，输入输出变量(VAR_IN_ OUT)，全局变量(VAR_GLOBAL)，临时变量(VAR_TEMP)，静态变量(VAR_STAT)，配置变量(VAR_ CONFIG)。

5.3.2 变量定义

变量可以在声明编辑器中定义。声明编辑器有两种显示形式：文本视图和表格视图。结构体和数组复杂数据 类型，支持变量定义支持数组元素注释，支持复杂数据类型地址递归显示。
文本声明如下图：

```
VAR_GLOBAL
END_VAR
VAR_GLOBAL RETAIN PERSISTENT
    {attribute 'ElemComment':='A_0([(ERT),9()]),A_1([10()])'}
    A AT%MW0:DUT := (A_0 := [2(FALSE), TRUE, 7(FALSE)]);
END_VAR
VAR_GLOBAL
        {attribute 'ElemComment':='A_0([(EEE),9()]),A_1([10()])'}
    B AT%MW200:DUT;
    C AT%MW400:DUT;
END_VAR
```

图5-1 文本声明

表格声明如下图：

类别	名称	地址	数据类型	初值	保持	常量	注释	特性
VAR_GLOBAL RETAIN PERSISTENT	A	%MW0	DUT	(A_0 := [2(FALSE), TRUE, 7(FALSE)])	☑	☐		
VAR_GLOBAL	B	%MW200	DUT		☐	☐		
VAR_GLOBAL	C	%MW400	DUT		☐	☐		

图5-2 表格声明

表格声明中，可以对变量的各类属性进行编辑设置。下表是对表格声明的具体说明。

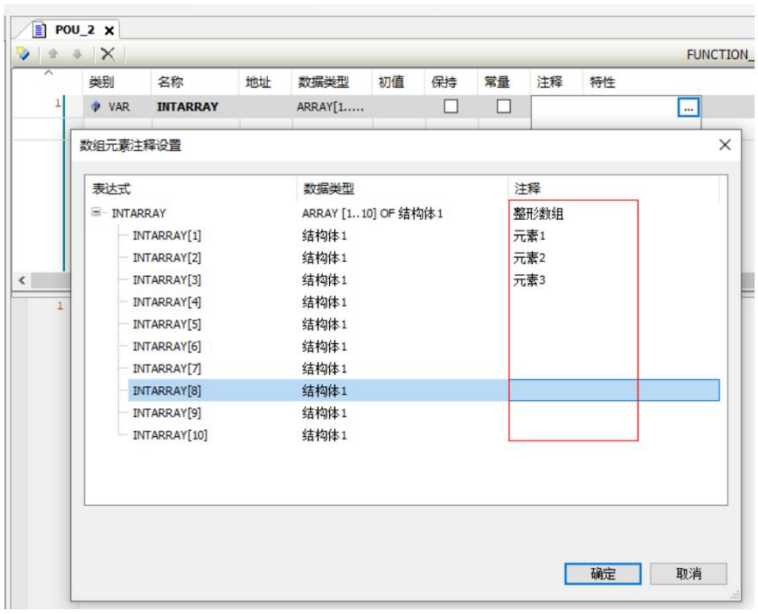
项目	描述
类别	变量的类别（如本地变量，输入变量，输出变量，临时变量等）
名称	变量的名称
地址	变量编译后的地址
数据类型	即变量的数据类型（如INT，BOOL等）
初值	变量的初始值
保持	标识变量是否为保持变量
常量	标识定义的变量是否为常量
注释	变量的注释
特性	变量的特性

变量定义支持数组元素注释和实例注释

数组元素注释

表格声明注释设置界面如下图所示。

双击注释空白处后点击 



设置完成后文本声明效果如下图所示（同样也可以用文本直接声明）：

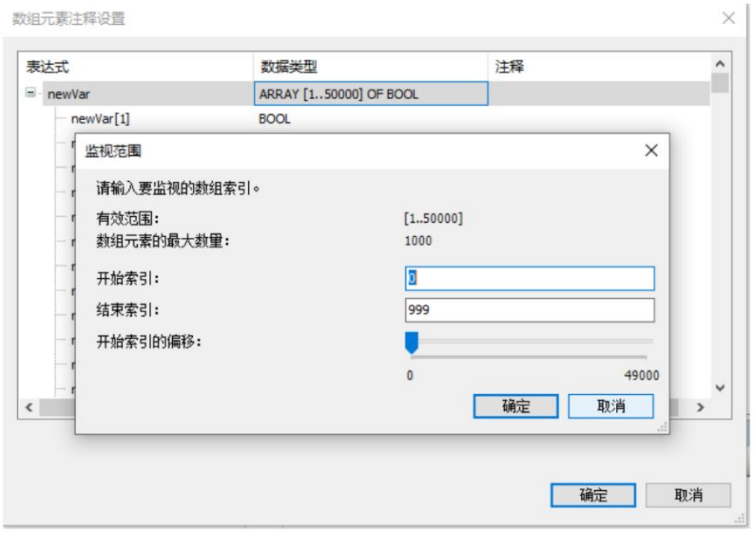
```
1  FUNCTION_BLOCK POU_2
2  VAR_INPUT
3  END_VAR
4  VAR_OUTPUT
5  END_VAR
6  VAR
7      // 整形数组
8      {attribute 'ElemComment':='(元素1),(元素2),(元素3),7()'}
9      INTARRAY: ARRAY[1..10] OF 结构体1;
10 END_VAR
11
```

- 可以通过表格和文本进行数组元素注释编辑。
- 表格在注释列，弹框（同初始值操作）显示当前元素及子元素注释编辑界面。
- 文本编辑器编辑格式如下：
  - 数组本身：使用标准的注释编辑方式。
  - 数组元素：{attribute'ElemComment':='1(子元素 1 注释),1(子元素 2 注释),n(相同子元素注释)'}。
- 如果是表格模式，声明数组类型变量时，默认注释为空（默认增加特性格式）。
- 表格模式的数组元素注释，只显示数组本身注释，不显示元素注释。
- 特性列中，去掉元素注释特性显示（数组元素注释是用特性来实现的，特性是标记在变量上的信息）。
- 表格视图数组长度变更时，存量数组元素的注释也会跟随保存。
- 表格视图中数组维度变更时，数组元素注释按扩展维度的最小下标对注释进行平移保存。

如数组 INT_ARRAY:ARRAY[1..2,2..3]维度变更为 ARRAY[1..2,2..3,3..4],原来的数组元素 INT_ARRAY[1,2]注释会迁移到新数组元素 INT_ARRAY[1,2,3]中。

如 INT_ARRAY:ARRAY[1..2,2..3]维度变更为 ARRAY[1..2],原来的数组元素 INT_ARRAY[1,2]注释会迁移 到新数组元素 INT_ARRAY[1]中。

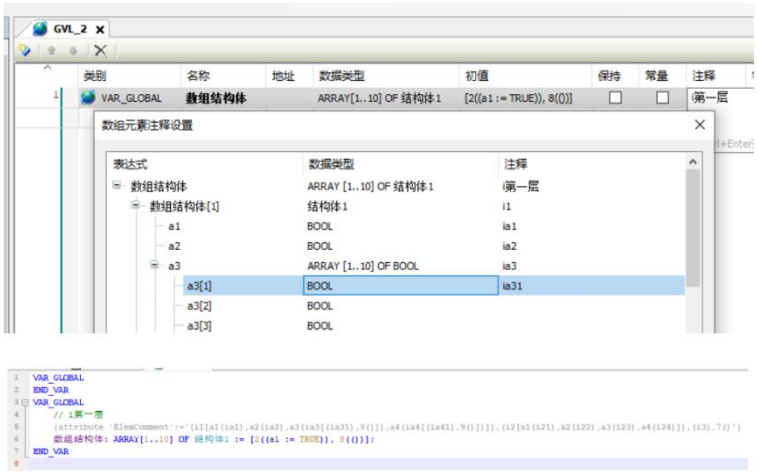
- 表格视图中，数据类型由数组类型变更为非数组类型时，会清空数组元素注释。
- 数组元素注释编辑界面最大显示 1000 个元素，双击数组所在的行中‘数据类型’列，可调节编辑显示范围。



实例注释

在 Prg（程序）和 GVL（全局变量表）中申明的变量或申明为 VAR_STAT（静态）类型的变量，可以不受限 制地展开变量内部成员编辑注释，保存注释时所有内部成员的注释会标记在该变量上，这种注释称为变量 的实例注释。

如下图所示，数据结构体内的成员注释都可以标记并保存在变量数组结构体上。



- 内部成员为 FB 类型时，只会显示输入、输出、输入输出变量，其他类型的变量不会展示。
- 表格中，数据类型由非数组类型变更为数组类型时，会清空实例注释。
- 变量的数组类型成员最大显示 1000 个元素，可调节编辑显示范围。

注释的显示

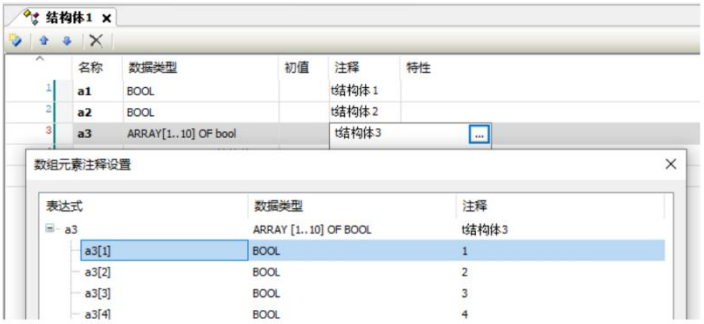
在初始值编辑界面，监控变量表界面，梯形图，鼠标悬停显示注释等涉及到变量注释显示的功能，注释显 示以实例注释优先，如果变量没有实例注释，则显示变量的类型注释。

梯形图中涉及到数组元素的注释显示的，以数组注释元素注释 合并显示；但同样采用上面的优先级顺序 规则进行显示。

例如下图显示数组元素 数据结构体[1].a3[1] 的注释。

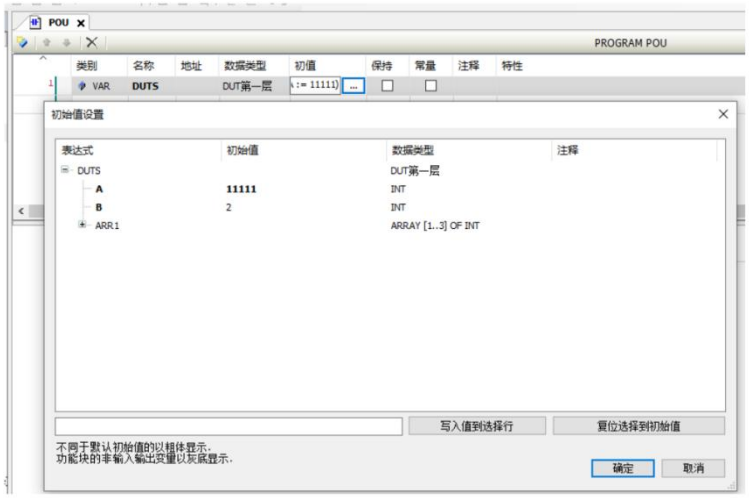
实例注释：

因为有实例注释，定义在类型中的数组注释和元素注释此时不显示。如下图所示。



变量的初始值设置

初始值设置界面如下。

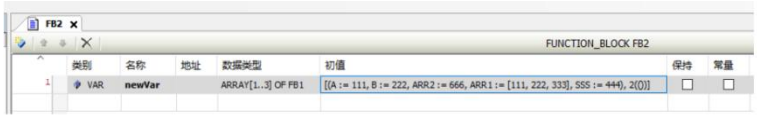


● 变量声明表格模式，点击”初值“列，可直接编辑初始值，也可以点展开按钮展开编辑。

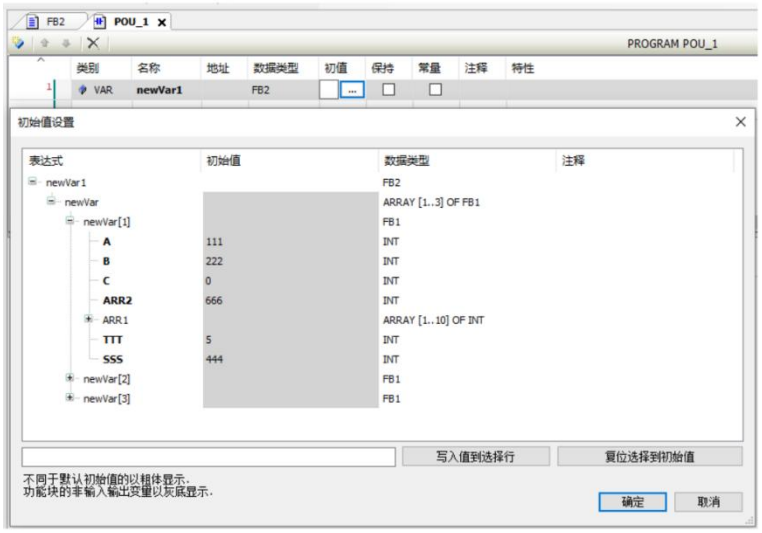
● 当设置的初始值与默认值不一样时，初始值以粗体显示。

● 初始值设置界面会把变量的成员逐层展开，中间变量不能直接设置，只有终端变量才能设置初始值，能设 置初始值的‘表达式’列以粗体显示。

● 功能块内部的不是输入、输出类型变量的成员以灰底色显示，并且不能编辑。例：FB2 中的成员 newVar 不是输入、输出类型的变量。



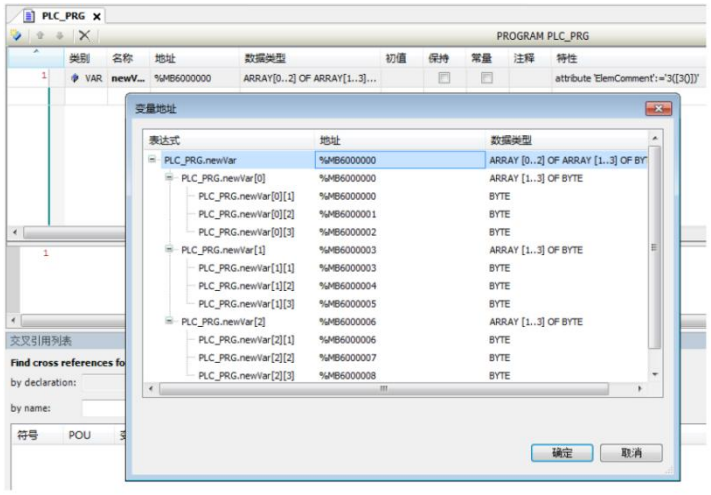
FB2 类型的 newVar 成员展开时，无法编辑初始值（显示灰色底）。



● 如果展开项目中有数组，数组最大显示 1000 个元素，可双击数组所在行的“数据类型”单元格调节编辑显示范围。

变量定义的子元素显示地址信息

地址显示界面如下：



● 变量声明表格模式，在地址列，编辑，如果变量是不包含功能块的块类型（数组，结构体，联合体、别名）弹出编辑框，显示子元素地址。

● 在地址列文本框中输入地址，地址显示界面只读。

- 弹出的地址编辑界面地址修改后，可以同步到变量地址中。
- 最大显示元素个数（数组中最大显示 1000 个元素，可调节显示范围）。

标识

标识即变量的名称。变量命名应注意以下事项：

- 不能包含空格或者特殊字符。
- 不能是预定义的关键字。
- 名称不区分大小写。
- 名称长度没有限制。
- 名称不能重复定义。

定义的本地变量名称可以和全局变量重名，默认使用本地变量，该变量可以用来表示全局变量，也可以通过全路径变量名来指定具体的变量。例如:本地变量 iVar:=1;全局变量.iVar:=2;全路径变量 globlist1.iVar：=3。 命名时需考虑部分命名建议：如变量名应准备表达其意义及数据类型，变量最好采用匈牙利命名法（变量名= 属性+类型+对象描述）。

AT 地址：

AT 地址为直接地址

数据类型：

数据类型分为标准数据类型和用户自定义数据类型。

标准数据类型 标准数据类型分为布尔类型、整形、浮点型、字符串、时间类型。

类型	关键字	范围	占用内存
布尔类型	BOOL	TRUE，FALSE，0，1	8bit
bit类型	bit	TRUE，FALSE，0，1，只能在结构体或者功能块中使用	1bit
整数	Byte	0~255	8bit
	WORD	0~65535	16bit
	DWORD	0~4294967295	32bit
	LWORD	0 ~2^{64}-1	64bit
	SINT	-128~127	8bit
	USINT	0~255	8bit
	INT	-32768~32767	16bit
	UINT	0~~65535	16bit
	DINT	-2147483648~2147483647	32bit
	UDINT	0~4294967295	32bit
	LINT	-2^{63}~2^{63}-1	64bit
	ULINT	0 ~2^{64}-1	64bit
浮点类型	REAL	1.401e-45~ 3.403e+38	32bit
	LREAL	2.2250738585072014e~308 - 1.7976931348623158e+308	64bit
字符串	STRING	只支持ASCII码字符（不支持中文字符）。默认最大长度80字符，超过最大长度会被去掉。可以声明字符最大长度，如str:STRING(35):=’This is a String’；字符串函数最大支持255字符	ASCII形式存储字符串，用一个字节存储结束符
	WSTRING	只支持UNICODE字符（支持中文字符）。默认最大长度80字符，超过最大长度会被去掉。可以声明字符最大长度，如wstr:WSTRING(35):=”This is a WString”；	UNICODE形式存储字符串，用两个字节存储结束符
时间	TIME	时间常量，日，时，分，秒，毫秒	32bit，内部按Dword处理
	TIME_OF_DAY(TOD)	一天的时间常量	32bit，内部按Dword处理
	DATE	日期常量，从1970年1月1日开始	32bit，内部按Dword处理
	DATE_ADN_TIME(DT)	日期和时间常量，从1970年1月1日开始	32bit，内部按Dword处理

用户自定义数据类型

用户自定义数据类型包括数组/结构体/枚举/联合/别名/子集/引用/指针。在中型 PLC 编程软件 CodeSys 中，可以通过右键应用-添加对象-DUT 来添加结构体、枚举、联合和别名 4 种自定义数据类型。

数组

语法：:ARRAY [..,..,..] OF 111, 112, 113 定义区域的下限，ul1, ul2 ul3 定义上限，数值必须为整数，elem. Type 为每个数组元素数据类型

初始化及示例

```
Card_game: ARRAY [1..13, 1..4] OF INT;
arr1 : ARRAY [1..5] OF INT := [1,2,3,4,5];
arr2 : ARRAY [1..2,3..4] OF INT := [1,3(7)]; (*数组值 1,7,7,7*)
arr3 : ARRAY [1..2,2..3,3..4] OF INT := [2(0),4(4),2,3]; (*数组值 0,0,4,4,4,4,2,3*)
arr1 : ARRAY [1..10] OF INT := [1,2]; (*数组部分初始化，没有初始化元素为默认值 0*)
```

数组结构初始化示例

结构定义:

```
TYPE STRUCT1
STRUCT
    p1:int;
    p2:int;
    p3:dword;
END_STRUCT
END_TYPE
```

数组结构初始化:

```
arr1:ARRAY[1..3] OF STRUCT1:= [(p1:=1,p2:=10,p3:=4723),(p1:=2,p2:=0,p3:=299)];
```

访问联合元素语法：

```
[Index1,Index2].
```

例子:

```
Card_game [9,2]
```

结构

语法：

```
TYPE | EXTENDS DUTTYPE:
STRUCT
    <declaration of Variables 1>
    …
    < declaration of Variables n>
END_STRUCT
END_TYPE
```

<structurename>是一个类型，可作为数据类型使用的。EXTENDS DUTTYPE 是可选项，表示继承了 DUTTYPE 中的成员，可以通过 structurename 类型变量访问 DUTTYPE 的成员。这里的 DUTTYPE 可以为结构 类型，联合类型或者别名。

初始化及示例

Polygonline 类型结构定义:

```
TYPE Polygonline:
STRUCT
    Start:ARRAY [1..2] OF INT;
    Point1:ARRAY [1..2] OF INT;
    Point2:ARRAY [1..2] OF INT;
    Point3:ARRAY [1..2] OF INT;
    Point4:ARRAY [1..2] OF INT;
    End:ARRAY [1..2] OF INT;
END_STRUCT
END_TYPE
```

初始化：

Poly_1:polygonline := (Start:=[3,3], Point1:=[5,2], Point2:=[7,3], Point3:=[8,5], Point4:=[5,7], End:= [3,5]); 访问结构元素语法：

< structurename>.< variable>

例子:

```
Poly_1.Start
```

枚举

枚举类型是由若干串常量组成的，这些常量被称为枚举类型值。

语法：

TYPE < identifier> : (<enum_0>,<enum_1>,…<enum_n>) |<base data type>;
END_TYPE

identifier:自定义的枚举类型；enum_n：枚举类型对应的常量值，每个常量可以声明其对应值，如果不声明 使用默认值；base data type 枚举常量对应数据类型，可以不用声明，默认为整数。

说明 同枚举变量一在多个库中同时存在时，需添加库名前缀，否则编译报错。

联合

语法：

```
TYPE <unionname>:UNION
    <declaration of variables 1>
    …
    <declaration of variables >
END_UNION
END_TYPE
```

<unionname>是一个类型，并且可作为数据类型使用的。联合中的所有变量具有相同的存储位置，联合类 型变量分配的空间大小为其中占用最大空间的变量分配的大小。

示例

```
TYPE union1: UNION
    a : LREAL;
    b : LINT;
END_UNION
END_TYPE
```

访问数组元素语法：

<unionname>.<variable>

示例

union1.a

别名

把一种数据类型用一种别名来表示。

语法：

TYPE <aliasname>:basetype END_TYPE

aliasname 为别名类型名，用作数据类型。basetype 可以是标准数据类型，也可以是用户自定义数据类型。

示例

```
TYPE alias1 : ARRAY[0..200] of Byte; END_TYPE
```

初始化及访问方式和其对应的基本类型一致。

子集

子集数据类型是其定义的基本数据类型的子集，可以通过增加 DUT 来增加一个子集类型，也可以直接声明一 个变量为子集类型。

DUT 对象语法：

TYPE <name>:<Inttype>(<ug>..<ug>)END_TYPE;

Inttype：是数据类型 SINT, USINT, INT, UINT, DINT, UDINT, Byte, WORD, DWORD (LINT, ULINT, LWORD)中的一种。

ug：是一个常数，必须符合基本类型对应的下边界范围。下边界本身包含在此范围内。

og：是一个常数，必须符合基本类型对应的上边界范围。上边界本身包含在此范围内。

DUT 对象声明示例

```
TYPE
    SubInt : INT (-4095..4095);
END_TYPE
```

变量直接声明示例

```
VAR
    i : INT (-4095..4095);
    ui : UINT (0..10000);
END_VAR
```

引用

引用是一个对象的别名，操作引用就如同操作对象。

语法：

<identifier>: REFERENCE TO <data type>

identifier：引用标示符。data type：引用对象的数据类型。

示例及初始化

```
ref_int : REFERENCE TO INT;
    a : INT;
    b : INT;
ref_int REF= a; (* ref_int 引用 a *)
ref_int := 12; (* a 值为 12 *)
b := ref_int * 2; (* b 值为 24 *)
ref_int REF= b; (* ref_int 引用 b *)
ref_int := a / 2; (* b 值为 6 *)
```

说明：不能引用 BIT 类型，即不允许定义 ref1:REFERENCE TO BIT。

指针:指针保存的是一个对象的地址，指针可以指向任何数据类型（bit 类型除外）

语法：

<identifier>: POINTER TO <data type>;

identifier：指针标示符。data type：指针指向的数据类型。

通过地址操作符对指针进行操作。地址操作符包括 ADR（获取变量地址）和^（变量地址对应的值） 示例及初始化

```
VAR
    pt:POINTER TO INT; (* 声明指向 INT 类型的指针 pt*)
    var_int1:INT := 5;
    var_int2:INT;
END_VAR

pt := ADR(var_int1); (* 变量 varint1 的地址分配给指针 pt *)
var_int2:= pt^; (* 通过^地址操作符获取指针对应的值*)
pt^:=33; (*给指针对应的变量 var_int1 赋值*)
```

初始值

变量初始化默认值为 0，用户可以在变量声明时通过赋值运算符“:=”来增加自定义初始化值。初始化值为有 效的 ST 表达式。ST 表达式由操作符，操作数，赋值表达式组成，操作符主要是加（+），减（-），乘（*），除（/）等组成；操作数主要是指常量，变量和函数；赋值表达式指 ST 表达式中的赋值运算符“:=”。因此， 初始化可以为常量，变量或者函数，只是要确保使用的变量已经被初始化。

示例：

```
VAR var1:INT := 12; (* 整形变量初始值 12*)
    x : INT := 13 + 8; (*常量表达式定义初始值*)
    y : INT := x + fun(4); (*初始值包含函数调用*)
    z : POINTER TO INT := ADR(y); (*指针通过地址函数 ADR 来初始化*)
END_VAR
```

说明

- 全局变量表（GVL）一般在 POU 本地变量定义之前已经初始化；
- 定义时初始化指针时，如果是在线修改默认，指针将不会被初始化（指针还是指向在线修改之前的变量）。

5.3.3 变量类型

主要的变量类型包括：本地变量（VAR）、输入变量（VAR_INPUT）、输出变量（VAR_OUTPUT）、输入输出变量（VAR_IN_OUT）、全局变量（VAR_GLOBAL）、临时变量（VAR_TEMP）、静态变量（VAR_ STAT）以及配置变量（VAR_CONFIG）。

变量类型声明语法：<type_key> | atribute_key

variable1;

variable2;

...

END_VAR

type_key：类型关键字，包括 VAR（本地变量），VAR_INPUT（输入变量），VAR_OUTPUT（输出变量）， VAR_IN_OUT（输入输出变量），VAR_GLOBAL（全局变量），VAR_TEMP（临时变量），VAR_STAT（静 态变量），VAR_CONFIG（配置变量）。 atribute_key：属性关键字，包括 RETAIN，PERSISTENT，CONSTANT，用于明确变量的范围。

本地变量（VAR）

在 POU 内部 VAR 和 END_VAR 之间的变量都为本地变量，不能被外部访问。

赋值格式：

本地变量:=值

示例

```
VAR
iLoc1:INT; (* 本地变量*)
END_VAR
```

输入变量（VAR_INPUT）

在 POU 内部 VAR_INPUT 和 END_VAR 之间的变量都为输入变量，可以在调用位置给输入变量赋值。

POU 调用格式：

本地变量:=调用者输入值

示例

```
VAR_INPUT
iIn1:INT; (* 输入变量*)
END_VAR
```

输出变量（VAR_OUTPUT）

在 POU 内部 VAR_OUTPUT 和 END_VAR 之间的变量都为输出变量。输出变量可以在调用时返回给调用者，调 用者可以做进一步处理。

POU 调用格式：

输出变量=>调用者匹配类型变量

示例

```
VAR_OUTPUT
iOut1:INT; (* 输出变量*)
END_VAR
```

输入输出变量（VAR_IN_OUT）

在 POU 内部 VAR_IN_OUT 和 END_VAR 之间的变量都为输入输出变量。输入输出变量不仅可以传入被调用的 POU 内，并且可以在被调用的 POU 内部修改。实际上传递给被调用 POU 内的变量是调用者变量的引用。

示例

```
VAR_IN_OUT
iInOut1:INT; (* 输入输出变量*)
END_VAR
```

Bit 类型直接变量示例：

```
VAR_GLOBAL
xBit0 AT %MX0.1 : BOOL;(*声明 Bit 类型直接变量*)
xTemp : BOOL; (*中间变量*)
END_VAR
//带有输入输出变量(xInOut)的功能块
FUNCTION_BLOCK FB_Test
VAR_INPUT
```

```
xIn : BOOL;
END_VAR
VAR_IN_OUT
xInOut : BOOL;
END_VAR
IF xIn THEN
xInOut := TRUE;
END_IF
//程序中调用功能块
PROGRAM Main
VAR
xIn : BOOL;
I1 : FB_Test;
I2 : FB_Test;
END_VAR
//使用 Bit 类型的直接地址变量，编译报错

//I1(xIn:=xIn, xInOut:=xBit0);
//通过中间变量 xTemp 把 xBit0 的值传递给功能块，然后把中间变量赋值给 xBit0
xTemp := xBit0;
I2(xIn:=xIn, xInOut:=xTemp);
xBit0 := xTemp;
```

输入输出常量(VAR_IN_OUT CONSTANT)只能读不能写，而输入变量在当前版本是能够修改的，即使增加了 常量属性，所以可以用输入输出常量来把变量属性变为不可修改。

输入输出常量示例：

```
PROGRAM PLC_PRG
```

```
VAR

sVarFits : STRING(16);

sValFits : STRING(16) := '1234567890123456';

iVar: DWORD;

END_VAR

POU(sReadWrite:='1234567890123456', scReadOnly:='1234567890123456', iVarReadWrite:=iVar);
//POU(sReadWrite:=sVarFits, scReadOnly:=sVarFits, iVarReadWrite:=iVar);

//POU(sReadWrite:=sValFits, scReadOnly:=sValFits, iVarReadWrite:=iVar);

//POU(sReadWrite:=sVarFits, scReadOnly:='23', iVarReadWrite:=iVar);


FUNCTION POU : BOOL

VAR_IN_OUT

sReadWrite : STRING(16); (* 在此 POU 内此字符串可读可写 *)

iVarReadWrite : DWORD; (*在此 POU 内此字此变量可读可写*)

END_VAR

VAR_IN_OUT CONSTANT

scReadOnly : STRING(16); (*在此 POU 内此字符串只读的*)

END_VAR

sReadWrite := 'string_from_POU';

iVarInPOU := STRING_TO_DWORD(scReadOnly);
```

全局变量（VAR_GLOBAL）

定义在 VAR_GLOBAL 和 END_VAR 之间的变量为全局变量。一般变量，常量，保留变量都可以声明为全局变 量。在 CodeSys 中可以通过右键应用-添加对象-添加全局变量表来添加全局变量表，然后在全 局变量表中添加全局变量。

```
VAR_GLOBAL
iGlobVar1:INT; (* 全局变量*)
END_VAR
```

临时变量（VAR_TEMP）

定义在 VAR_TEMP 和 END_VAR 之间的变量为临时变量，临时变量在每次调用时会进行初始化。

示例

```
VAR_TEMP
iTemp1:INT; (*临时变量*)
END_VAR
```

静态变量（VAR_STAT）

定义在 VAR_STAT 和 END_VAR 之间的变量都为静态变量。静态变量在第一次调用时被初始化，在每次调用完此 POU 后，变量值依然保持下来。

示例

```
VAR_STAT
iStat1:INT; (*静态变量*)
END_VAR
```

配置变量（VAR_CONFIG）

定义在 VAR_CONFIG 和 END_VAR 之间的变量都为配置变量。配置变量是直接变量，一般是映射到功能块定义的不确定地址直接变量。在功能块内可以定义一个不确定地址的变量，此变量的地址通过“*”来表示不确定 的地址（任意的地址），然后添加一个配置变量表（通过添加全局变量表方式），把所有功能块实例中不确 定地址变量添加到配置变量表中，在此变量表中把所有的不确定地址给明确下来，这样就可以集中管理所有 功能块中不确定地址变量。

功能块不确定地址变量定义语法:

<标识符> AT % <I|Q|M>*:<数据类型>

地址的最终确定是在全局变量列表的“变量配置”中完成：

示例

```
FUNCTION_BLOCK
```

```
locio VAR xLocIn AT %I*: BOOL := TRUE;

xLocOut AT %Q*: BOOL;

END_VAR
```

这里定义了两个 I/O-变量,一个本地输入变量(%I*) 和一个本地输出变量 (%Q*)。

然后添加“全局变量列表”对象(GVL)。在关键词 VAR_CONFIG 和 END_VAR 之间输入实例变量声明的具体 地址,这里的实例变量指包含 POU 完整的实例路径,具体地址对应于功能块中不确定指定的地址(%I*,% Q*),另外数据类型必须与功能块的声明一致。

配置变量定义语法:

<实例变量路径> AT % <I|Q|M><位置>:<数据类型>;

示例

```
PROGRAM PLC_PRG

VAR

locioVar1: locio;

locioVar2: locio;

END_VAR

VAR_CONFIG (*正确的变量配置表*)

PLC_PRG.locioVar1.xLocIn AT %IX1.0 : BOOL;

PLC_PRG.locioVar1.xLocOut AT %QX0.0 : BOOL;

PLC_PRG.locioVar2.xLocIn AT %IX1.0 : BOOL;

PLC_PRG.locioVar2.xLocOut AT %QX0.3 : BOOL;

END_VAR
```

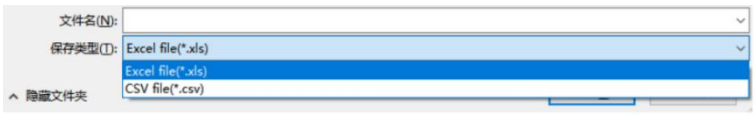
5.3.4 变量导入与导出

支持变量导入与导出，导出文件类型为“XLS 工作表 (.xls)”，呈现为 Excel 表格形式，可在外部进行增加、删减或其他对变量的编辑后再导入到 CodeSys 编程软件。

如下图所示。

	类别	名称	地址	数据类型	初值	保持	常量	注释	特性
1	VAR_GLOBAL RETAIN PERSISTENT	A_0		BOOL		☑	☐	变量A	
2	VAR_GLOBAL CONSTANT	A_1		INT		☐	☑		
3	VAR_GLOBAL	A_2		BYTE		☐	☐		
4	VAR_GLOBAL	A_3		WORD	100	☐	☐		
5	VAR_GLOBAL	A_4		DWORD		☐	☐		AAA
6	VAR_GLOBAL	A_5		ARRAY[0..9] OF REAL		☐	☐		

在变量表中增加部分变量，并右键选择导出变量表类型 Excel/CSV（CSV 是纯文本文件，excel 包含有格式信 息。CSV 文件较小，创建分发读取较方便，适合存放结构化信息。CSV 文件在 windows 平台默认的打开方式是 excel，本质是文本文件。），两种格式文件对变量的编辑没有区别。



打开导出的文件，并编辑（添加新变量 A_6、A_7、A_8、A_9）后导入，效果如下图所示。

Type	Name	Address	DataType	InitValue	Comment	Attribute
VAR_GLOBAL RETAIN PERSISTENT	A_0		BOOL		变量A	
VAR_GLOBAL CONSTANT	A_1		INT			
VAR_GLOBAL	A_2		BYTE			
VAR_GLOBAL	A_3		WORD	100		
VAR_GLOBAL	A_4		DWORD			
VAR_GLOBAL	A_5		ARRAY [0..9] OF REAL			
VAR_GLOBAL RETAIN PERSISTENT	A_6		BYTE			
VAR_GLOBAL CONSTANT	A_7		WORD			
VAR_GLOBAL	A_8		DWORD	200		
VAR_GLOBAL	A_9		ARRAY [0..9] OF REAL		变量B	

	类别	名称	地址	数据类型	初值	保持	常量	注释
1	VAR_GLOBAL RETAIN PERSISTENT	A_0		BOOL		☑	☐	变量A
2	VAR_GLOBAL CONSTANT	A_1		INT		☐	☑	
3	VAR_GLOBAL	A_2		BYTE		☐	☐	
4	VAR_GLOBAL	A_3		WORD	100	☐	☐	
5	VAR_GLOBAL	A_4		DWORD		☐	☐	
6	VAR_GLOBAL	A_5		ARRAY [0..9] OF REAL		☐	☐	
7	VAR_GLOBAL RETAIN PERSISTENT	A_6		BYTE		☑	☐	
8	VAR_GLOBAL CONSTANT	A_7		WORD		☐	☑	
9	VAR_GLOBAL	A_8		DWORD	200	☐	☐	
10	VAR_GLOBAL	A_9		ARRAY [0..9] OF REAL		☐	☐	变量B

5.4 常量

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

常量声明语法：

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

示例

```
VAR CONSTANT
c_iCon1:INT:=12;
```

END_VAR

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

类型	描述	示例
布尔类型	有两个值TRUE和FALSE（也可以用1和0），1表示TRUE，0表示FALSE	TRUE，FALSE，1
BIT类型	和布尔类型相似，只能在结构体（占用位数）或者功能块（映射BOOL类型的直接地址）中使用	TRUE，FALSE，0
整数	整数常量的数值可以是二进制、十进制、八进制和十六进制。如果整数值不是十进制值，可以用“进制”加符号“#”放在整数值前面来表示。十进制的10至15在十六进制中表示为A至F	十进制：66 二进制：2#101 八进制：8#72 十六进制：16#3A 类型常数： INT#22 BYTE#204
浮点类型	浮点常量用十进制小数和指数来表示，遵循标准的科学计数法格式	7.4 2.3e+9 REAL#3.12
ASCII字符串	ASCII字符串常量在两个单引号之间，可以包含空格和特殊字符。一个字符用一个字节表示，只支持ASCII码字符（不支持中文字符）。默认最大长度80字符，超过最大长度会被去掉。可以声明字符最大长度，如str:STRING(35):=’This is a String’；字符串函数最大支持255字符。	$做转义字符例： ‘$30’：0，字符0，16进制30对应的ASCII字符 $$：$，美元字符 $’：’，单引号
UNICODE字符串	UNICODE字符串常量在两个双引号之间，一个字符站两个字节，只支持UNICODE字符（支持中文字符）。默认最大长度80字符，超过最大长度会被去掉。可以声明字符最大长度，如wstr:WSTRING(35):=”This is a WString”；	“Unicode string”
时间	时间常量一般用来操作时间，由“T#”（或“t#”）加上“时间值”构成，时间值的单位包括天（d）、小时（h）、分（m）、秒（s）和毫秒（ms）。	t#12h34m15s;
时刻	一天的时间范围，语法：TOD#时间值。	TOD#15:36:30.123
日期	从1970年1月1日开始，语法：d#日期。	d#2015-02-12
日期时刻	日期常量和时刻常量合并起来称为日期时刻常量，由 从1970年1月1日开始，语法：dt#日期。	dt#2004-03-29-11:00:00

5.5 掉电保持变量

5.5.1 概述

注 意

仅在编译器3.5.11.71及以上版本支持掉电保持变量功能。可通过菜单栏选择“工程 > 工程设置”打开“工程设置”对话框，在“编译选项”界面查询编译器版本。

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

定义变量

掉电保持变量支持在全局变量表（GVL）、程序（PRG）、功能块（FB）、函数（FUN，仅限静态类别变量）中进行定义，不支持在方法（METH）、属性（Prop）、结构体（STRUCT）、联合体（Union）、枚举 （Enum）、别名（Alias）中进行定义，可通过文本方式进行定义，以在程序（PRG）中定义为例。

5.5.2 文本方式

在“PLC_PRG(PRG)”编程界面文本方式下，通过对变量增加“RETAIN PERSISTENT”或“PERSISTENT RETAIN”关键字，定义该变量为掉电保持变量。

6. 编程语言

6.1 CodeSys 支持的编程语言简介

编程软件支持下列 PLC 编程语言：

- Ladder diagram(LD) 梯形图
- Function block diagram(FBD)功能块图
- Structured text (ST) 结构化文本
- Sequential function chart (SFC)顺序功能图
- Continuous function chart (CFC)连续功能图

其中，LD、FBD、ST、SFC 是基于 IEC 61131-3 标准，CFC 是 IEC 61131-3 标准的一个扩展。

不论用户选用哪种语言，编程界面中的基本编辑方法是通用的，为编程带来很大方便。

- 标准的编辑器功能，如支持“复制”（Ctrl+C）、“粘贴”（Ctrl+V）和“删除”（Del）等快捷键；
- 标准的、按键多项选择；
- 支持功能键启动输入助手，系统根据具体环境提供相对应的输入提示或选择。

6.2 结构化文本语言(ST)

6.2.1 概述

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

例如：

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

6.2.2 表达式

表达式是一种结构，对它求值后，这个值可以在指令中使用。

表达式由操作符和操作数组成。一个操作数可以是一个常量，变量，功能调用或其他表达式。例如：

- 常量，例如：20，t#20s，’string’
- 变量，例如：iVar，Var1[2,3]
- 功能调用，值为调用返回值，例如：Fun1(1,2,4)

- 其它表达式：10+3，var1 OR var2，(x+y)/z，iVar1:=iVar2+22

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

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

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

下表记录了 ST 语言的操作符:

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

6.2.3 ST 指令

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

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

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

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

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

WHILE	WHILE循环	J:=1; WHILE J<= 100 AND ARR[J] <> 70 DO J:=J+2; END_WHILE;
REPEAT	REPEAT循环	J:=-1; REPEAT J:=J+2; UNTIL J= 101 OR ARR[J] = 70 END_REPEAT;
EXIT	退出循环	EXIT;
CONTINUE	继续循环下次执行	CONTINUE;
JMP	跳转	label: i:=i+1; JMP label;
;	空语句	;

赋值指令

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

- “:=”为一般赋值，右值直接赋给左值，左值和右值相等。

例如：Var1 := Var2 * 10;

完成执行后，Var1 值为 Var2 的 10 倍。

- “S=”为置位赋值，表示如果右值为 TRUE，左值变量变为 TRUE（置位），直到调用 R=命令来初始化。
- “R=”为复位赋值，表示如果右值为 TRUE，左值变量变为 FALSE（复位）。用于复位 S=指令置位的变 量。

例如：a S= b;

一旦 b 为 TRUE 后，a 会保持 TRUE，即使 b 变为 FALSE 后。

功能块的调用

语法：<FB 实例名>(FB 输入变量:=< 值和地址>|,<更多 FB 输入变量:=<值和地址>...更多 FB 输入变量>

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

注意：TON 功能块通过"TMR:TON"实例化。

实例化语法： <FB instance name>:<FB variable >;

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

RETURN 指令

RETURN 指令表示当前置条件为 TRUE 时，离开此 POU。

语法：

```
RETURN;
```

示例

```
IF b=TRUE THEN
RETURN;
END_IF;
a:=a+1;
```

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

IF 指令

通过 IF 关键字，可以判断执行条件，根据执行条件，执行相应的指令。

语法：

```
IF <布尔表达式 1> THEN
<IF_指令
{ELSIF <布尔表达式 2> THEN
<ELSIF_指令 1>

ELSIF <布尔表达式 n> THEN
<ELSIF_指令-1>
```

ELSE
<ELSE_指令>
END_IF;
{}内部分是可选的

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

示例

```
IF temp<17
THEN heating_on := TRUE;
ELSE heating_on := FALSE;
END_IF;
```

这里，当温度低于 17 度时加热打开，否则它保持关闭。

CASE 指令

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

语法:

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

CASE 指令根据以下流程处理：

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

示例

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

FOR 循环

通过 FOR 循环，可以编写重复处理逻辑。

语法：

FOR <INT_Var> := <INIT_VALUE> TO <END_VALUE> {BY <Step size>} DO

<instructions>

END_FOR;

{} 内的部分是可选的。

INT_Var是计数器，是整数类型，只要计数器<INT_Var>不大于<END_VALUE>，<Instructions>会被执行。在执行 <Instructions> 之前首先要检查该条件，如果 <INIT_VALUE>大于 <END_VALUE>，<instructions> 不会被执行。

当 <Instructions>执行一次后, <INT_Var>自动增加<Step size>。<Step size>可以是任意整数值，如果不写此参数，默认值为1。当<INT_Var>大于<END_VALUE>时，循环停止。

示例

FOR Counter:=1 TO 5 BY 1 DO

Var1:=Var1*2;

END_FOR;

Erg:=Var1;

假设Var1默认值是2，经过FOR循环后，它的值是32。

WHILE 循环

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

语法：

WHILE <boolean expression> DO

<instructions>

END_WHILE;

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

示例：

WHILE Counter<>0 DO

Var1:= Var1*2;

Counter := Counter-1;

END_WHILE

在一定意义上来说，WHILE 循环和 REPEAT 循环比 FOR 循环功能更强大，因为不需要在执行循环之前计算循环 次数。因此，在有些情况下，用 WHILE 循环和 REPEAT 循环两种循环就可以了。然而，如果清楚知道循环次 数，那么 FOR 循环更好。

REPEAT 循环

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

语法:

REPEAT

<instructions>

UNTIL <Boolean expression>

END_REPEAT;

执行逻辑:

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

示例:

REPEAT

Var1:=Var1*2;

Counter:=Counter-1;

UNTIL Counter=0;

END_REPEAT;

CONTINUE 语句

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

示例：

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

EXIT 语句

EXIT 指令用于退出 FOR, WHILE,或 REPEAT 循环。

JMP 语句

JMP 指令可用于无条件的跳转到指定标签处的代码行。

语法：

```
<label>:
JMP <label>;
```

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

示例：

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

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

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

注释

在结构化文本中有两种写注释的方法。

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

注释可以在ST编辑器声明或实现部分的任意地方。

注释的嵌套：注释可以放置在其他注释中。

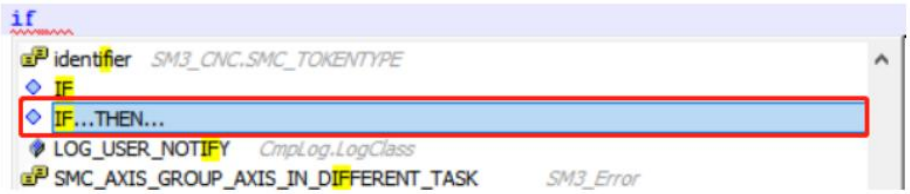
示例：

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

6.2.4 ST 编辑

智能输入：

● 关键字匹配 输入 ST 语句类型关键字，能够自动匹配，语句包含 IF 语句、WHILE、FOR、CASE、REPEATED，格式化模 板见附录语句模板。如下图所示，输入 IF 能够弹出响应联想语句。



● TAB 键快捷功能

■ 能够对功能块、函数、方法、动作、程序自动格式化输入、输出

■ 能够对功能块实例及实例的方法、动作自动格式化输入、输出

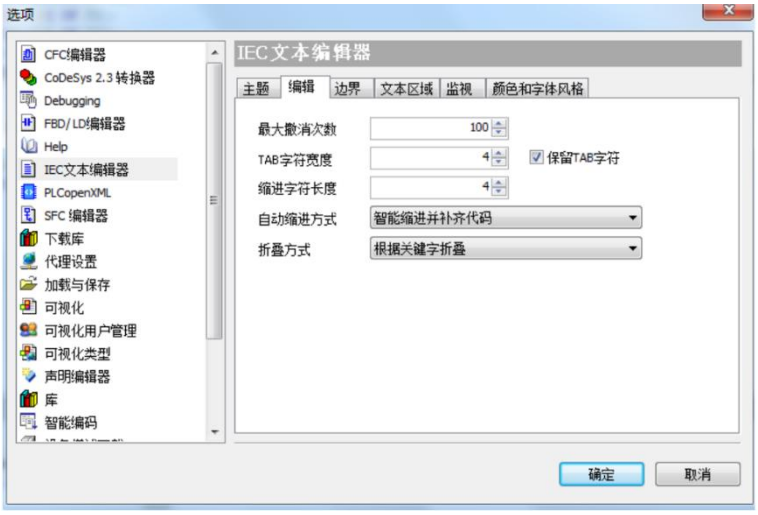
■ 能够对 IF、WHILE，FOR，CASE，REPEATED 语句自动格式化，格式化模板见附录语句模板，入功能 类型名、功能块实例等输入后按 Tab 键自动格式化

折叠和缩放功能：

● 折叠方式支持关键字：关键字包含 VAR、VAR_INPUT、VAR_GLOBAL、VAR_OUTPUT、VAR_IN_OUT、 VAR_TEMP、VAR_STAT、VAR_ESTERNAL、CASE、FOR、REPEATED、IF/ELSE/ELSIF、WHILE、 STRUCT、UNION、TYPE、__TRY、__CATCH、__FINALLY。

● 如果选择自动缩进功能中的智能缩进，根据上述的关键字，自动加 Tab 长度，如果选择智能缩进并自动补 齐，自动给关键字补齐结尾，如 VAR，FOR，WHILE 自动补齐结束标记，支持嵌套。

- 智能缩进时，如果上行是关键字，换行后自动增加 Tab 字符，如果非关键字，和上行同缩进。
- 块高亮显示。括号、中括号、WHILE、FOR、IF、ElSE、CASE、REPEAT、STRUCT、UNION、TYPE、 TRY 等类别之间显示块高亮信息，在文本边界和文本区域都有高亮显示标记。



IEC 文本编辑器界面颜色：

ST 界面颜色每种都是一个模板，通过模板配置，也可以通过【工具】-【选项】-IEC 文本编辑器-主题，颜色配 置包含基本配置、IEC61131 类型及标识符配置、在线配置及字体风格配置。

可根据用户喜好进行自定义设置，可设置参数具体如下表所示。

类别	说明
基本配置	主要设置基本界面基本颜色，主要包含： 背景色 前景色（默认文本颜色） 行高亮颜色 文本块高亮颜色 符号高亮颜色 光标颜色 焦点状态下选择文本背景颜色 非焦点状态下选择文本背景颜色 边界默认文本颜色 边界背景颜色 边界扩展背景 边界行高亮颜色 焦点状态分割线 非焦点状态分割线 折叠状态颜色 增量搜索颜色

类别	说明
IEC61131类型及标识符	主要设置数据类型和标识符颜色，主要包含： BOOL类型常数 时间类型常数 整数类型常数 浮点数类型常数 字符串类型常数 注释类型 特性类型 直接地址 全局变量 静态变量 输入变量 输出变量 输入输出变量 常量 临时变量 Persistent变量 Retain变量 外部变量 配置变量 FB 方法 动作 函数 结构体 枚举类型 枚举值 联合体 接口 操作符 关键字 错误
在线配置	监视框背景和文本 流控监视框背景和文本
字体风格	数据类型和标识符字体格式，如粗体，下划线等

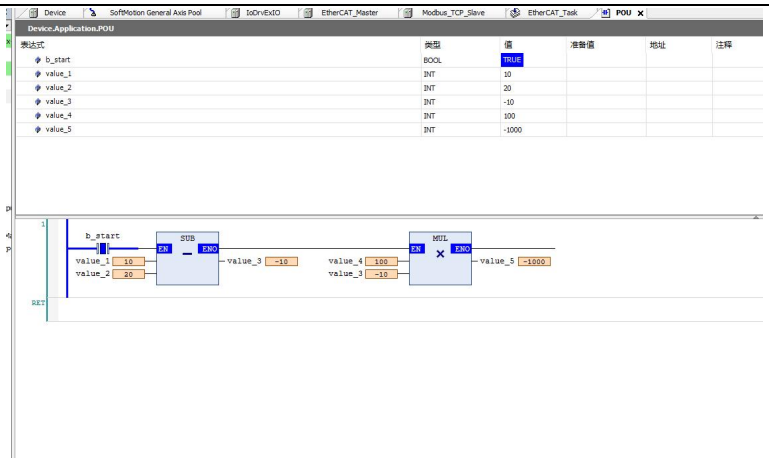
掉电保持、保持及常量类型和变量是修饰关系，两者只能选择一个，优先级关系如下：全局变量、输入、输 出、输入/输出掉电保持（保持）本地、临时常量静态变量、配置变量、外部变量、枚举变量种变量都可以颜 色设置，由于掉电和常量是特性，和变量类型是并列的，所以需要优先控制显示。

6.3 梯形图(LD)

6.3.1 概述

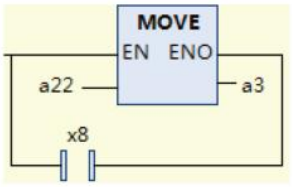
梯形图是图形化的编程语言，跟电路图的结构相近。梯形图包括一系列网络（也叫节，下文统一以“网络”代替），每个网络由左侧的竖线（电源轨，能流线）开始。一个网络由触点，线圈，运算块（函数、功能 块、程序、执行块、动作、方法）、跳转、标签和连接线等构成。

网络左侧母线为能流线，其状态永远为 TRUE，母线后会连接触点、运算块、线圈等元素。每个触点均分配布 尔变量。如果变量值为 TRUE，相当于开关闭合，条件会从沿着连接线从左向右传递，否则开关断开。在网络 右侧的线圈，接收到从左侧传来的“开”或“关”信号，相应的 TRUE 或 FALSE 会被写到与线圈关联的布尔变 量中。梯形图编辑界面如下图。



梯形图主要元素包括触点、线圈、运算块、分支、注释等。通过插入、拖拽、划线、复制粘贴操作在网络中 添加这些元素，形成梯形图执行逻辑。对于梯形图界面字体、操作数及注释显示可以通过【工具】-【选项】- 【FBD/LD 编辑器器】设置。

梯形图支持监控、写入值、强制值、断点等在线调试功能。

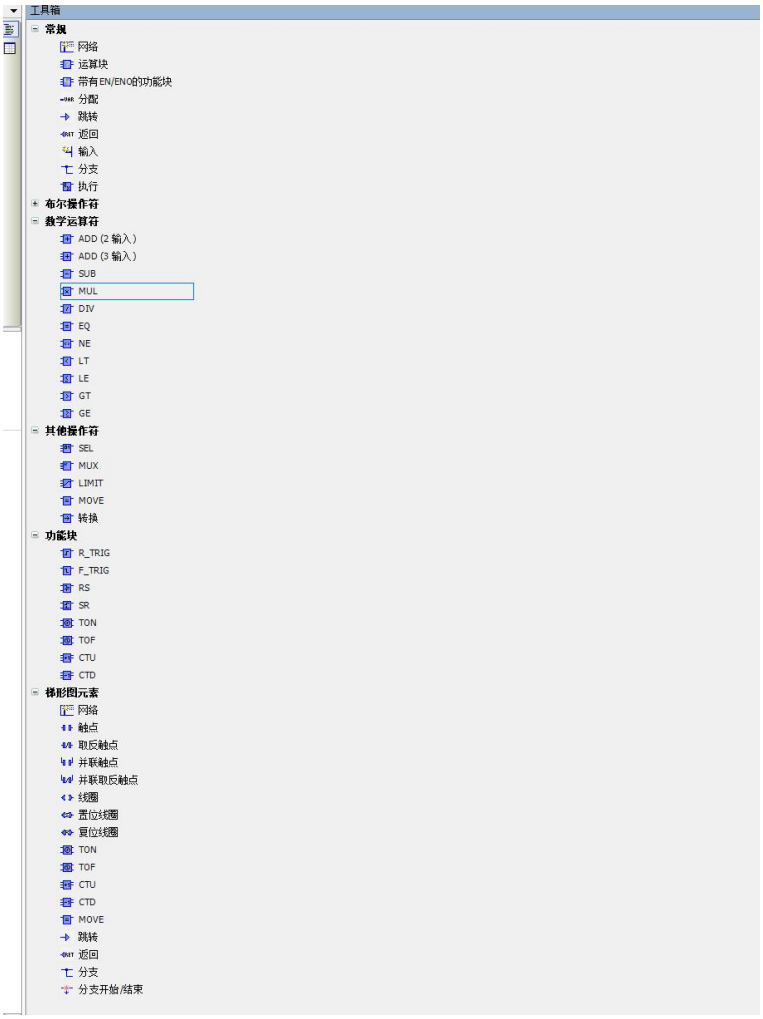


6.3.2 梯形图元素

梯形图元素包括网络、触点、线圈、运算块、执行块、分支、跳转、标号、返回。

触点、线圈、运算块输入输出都和操作数关联，操作数可以是变量、常量(TRUE、FALSE、1,2 等)、地址，具体见变量定义。

LD 元素位于工具箱（菜单命令【视图】-【工具箱】）中，如下图所示。在工具箱中除了常规下元素、梯形图元素和 IEC 标准操作符（如布尔操作符、数学操作符）外，还包括功能块，当前程序中定义的 POU。



网络

图标 

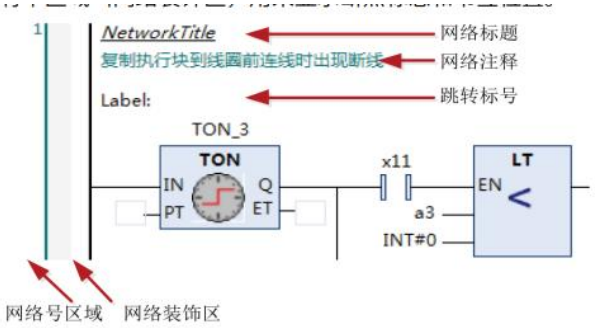
梯形图由一系列网络组成，其它所有的梯形图元素都位于网络内。每个网络都由左边的网络序号指示。

网络中可插入标题（网络总结性说明）和网络注释（网络比较详细的说明）。网络标题、网络注释是否显示通过选项配置(【工具】-【选项】-【FBD/LD 编辑器】-【常规】)来控制。

网络中可插入标签，标签位于网络标题和网络注释下面，作为跳转目标。

网络也可以处于注释状态，通过菜单命令【切换网络注释状态】来使能或者禁用网络。

网络序号和网络内容之间有个区域叫网络装饰区，用来显示断点标志和书签位置。



触点

图标 -

触点分常开触点和常闭触点。触点传递 ON（TRUE）和 OFF（FALSE）值，触点为 BOOL 型变量，如果变量值 为 TRUE，常开触点向右传递 ON（TRUE），否则传递 OFF（FALSE），常闭触点传递值相反。
触点可以增加延信号功能，选中触点右键菜单命令【边沿检测】，可以把触点变为上升沿触发触点（触点变量值由 FALSE 变为 TRUE 时触点向右传递 ON）或者下降沿触发触点（变量值由 TRUE 变为 FALSE 时触点向右传 递 ON）。

线圈

图标 -

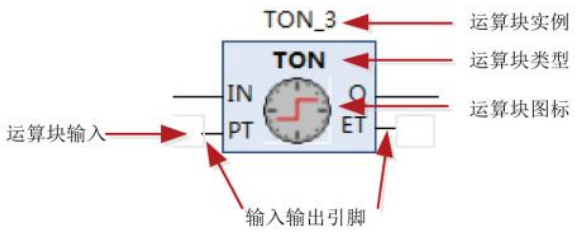
线圈位于网络的末尾。左侧逻辑运算结果，赋值给线圈变量。线圈变量只能为 BOOL 类型，TRUE 表示 （ON），FALSE 表示（OFF）。线圈仅支持向上或者向下插入并联线圈。
线圈分为线圈、取反线圈、置位线圈、复位线圈。通过右键菜单命令或者快捷键可以进行 4 种线圈类型之间的切换。

- 线圈：左侧逻辑运算结果直接赋值给线圈变量。
- 取反线圈：把左侧逻辑运算结果取反赋值给线圈变量。
- 置位线圈：如果线圈左侧的状态值为 ON（TRUE）,则把线圈变量的值设置为 ON（TRUE），并且一直保持这个状态，直到下次该变量被复位线圈重置为 OFF（False）。
- 复位线圈：对置位线圈进行复位。

运算块

图标 -

运算块可以是操作符、函数、功能块、程序、动作、方法。如果为功能块类型，则运算块框上面会增加编辑 框来显示功能块实例。
一个运算块至少包含一个输入和一个输出。运算块主要有由下图组成：

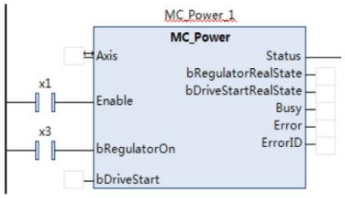


运算块分普通运算块和 En/Eno 运算块。

● EN/ENO 类型运算块：除了包含运算块本身所带的输入和输出外，还增加 EN 输入和 ENO 输出。EN/ENO 运算块执行逻辑为：当 EN 为 TRUE 时执行运算块逻辑，执行完成后 ENO 为 TRUE，如果 EN 为 FALSE，不执行运算块，ENO 为 FALSE。注意：EN/ENO 运算块输入连线只能连接在 EN 引脚，输出连线只能连接在 ENO 引脚。

● 运算块输入输出引脚：BOOL 型输入引脚可以增加取反、上升沿、下降沿信号。运算块输出连引脚可以增 加取反信号。

● 多输入连线运算块：有多个输入且多个输入均连接到能流线上，如下图为两个输入连线的多输入连线运算 块。由于多输入连线运算块有多个连线和能流线相连，所以多输入连线运算块不能再并联分支中，并且只 能在第一个分支中。



执行块

图标 –

执行块是一个可以插入内嵌 ST 的块，在块中可以编辑 ST 语句。执行块可以放大缩小，最大为 1000*400。

分支

图标 –

分支形成一个非闭合的并联逻辑。

标签

图标 –

标签标明跳转位置，位于网络头部，通过跳转元素跳转到标签位置。跳转标签是字符串，需符合标识符命名规则。

跳转

图标 –

当跳转元素左侧的输入为 TRUE 时，跳转到指定的标签位置执行。跳转元素位于网络的最右侧。

返回

图标 –

当返回元素左侧的输入为 TRUE 时，当前程序立即退出执行。返回元素位于网络的最右侧。

6.3.3 LD 编辑器选项

LD 编辑器选项用来控制 LD 界面的显示、单键命令设置、打印时显示方式。LD 编辑器选项通过菜单命令【工具】-【选项】-【FBD/LD】打开。LD 编辑器选项包含 3 个选项卡：常规、LD 和打印。

常规设置

常规选项卡设置如下图。

6.3.4 LD 编辑器常规选项卡

视图

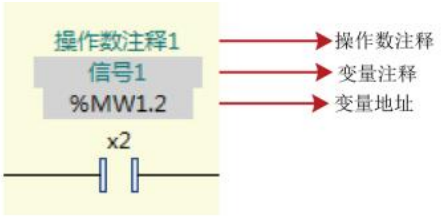
● 插入网络标题：如果激活此选项，梯形图每个网络可以插入标题、编辑标题。如果已经插入了标题会在当 前网络最上面增加一行，用来显示标题，如果没有标题，标题行不会显示。插入标题通过菜单命令实现。

● 显示网络注释：如果激活此选项，梯形图每个网络可以编辑网络注释。如果增加了网络注释，在网络标题 下增加一行显示网络注释，如果网络注释不存在，不会产生空白行显示网络注释。编辑网络注释通菜单命令实现。

● 显示功能块图标 - 如果激活此选项，如果运算块定义了图标，运算块中间会显示图标。标准操作符（如 ADD、SUB）和功能块（如 TON、TOF）都定义了图标，用户自定义的函数、功能块或者程序可以通过 【右键对象】-【属性】-【位图】-【点击此处选择与工程有关的位图】来添加图片，形成运算块图标。

● 显示操作数注释：如果激活此选项，梯形图界面每个操作数上都可以编辑和显示操作数注释。操作数是一个编程概念，如变量、常量、地址都是操作数。由于梯形图中不一定都使用变量，如常量或者地址，这时 可以通过操作数注释对它们进行注释。编辑操作数注释通过选择操作数字符串，然后右键菜单实现编辑。

● 显示变量注释：如果激活此选项，梯形图界面变量上，会显示变量声明时的注释。变量注释来自于变量声 明，不能编辑。



字体

点击样本文字，弹出编辑器字体选择框，设置梯形图文字字体。默认字体为微软雅黑，使用小五号字体。字体范围主要包括操作数字符、注释。执行块字体使用文本编辑器字体（ST 文本、变量声明文本）。

动作

- 新操作符的占位符：未实现
- 添加运算块时默认空引脚：如果激活此选项，新增加运算块时，运算块输入和输出引脚用空字符，如果未 激活此选项，运算块输入和输出引脚用”???”字符。

操作数固定大小设置

如果激活此选项，可以设置操作数固定宽带、操作费注释高度和变量注释高度。如下图。

操作数固定大小设置

- 操作数宽度：设置操作数固定字符个数，默认 15 个。
- 操作数注释高度：设置操作数注释固定高度行数，默认 1 行。
- 变量注释高度：设置变量注释固定高度行数，默认 1 行。

LD 选项设置

LD 选项卡设置如下图。

单键设置

单键功能，通过单个按键进行编辑操作，包括连线上执行的单键和元素上执行的单键。连线上执行单键命令 插入串行元素，元素上执行单键命令插入并行元素。

另外选择元素时可以对元素功能进行切换，如取反切换、延信号切换、置位/复位切换。取反切换作用对象为 触点、线圈，使用“/”按键；延信号切换作用对象为触点，使用空格按键；置位/复位切换作用对象为线圈 使用空格按键。

单键设置，设置单键功能按键字符。每种功能都有默认单键字符，但是可以根据个人喜好自行设置。

打印

打印选项卡设置如下图。

布局选项

打印大小适配方式：

- Poster，正常比例打印，打印时如果当前页面高度显示不了此网络使用，使用下一个页面打印，如果页面宽度显示了整个网络，使用下一个页面打印剩余的。
- Shrink to fit the widest network，表示打印时压缩显示内容使所有网络都一个页面宽度内显示，并且如 果一个页面高度显示不完整个网络，显示部分网络，剩下的网络在下一个页面显示。
- 避免元素切割：如果激活了此选项，表示当一个元素在两个页面之间都有显示，则把当前元素在下一个页面显示，只能在 Poster 打印方式设置。
- 相邻页上的标记连接：表示设置了避免元素切割后，为了表示前后连接关系，增加连接标记。

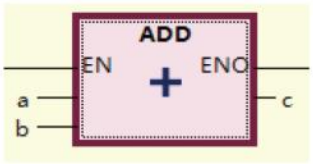
6.3.5 元素选择

选择是编辑的基础。选择对象可以是元件或者连线，选择可以单选也可以按下 Ctrl、Shift 多选，可以连续选择，也可以非连续选择。

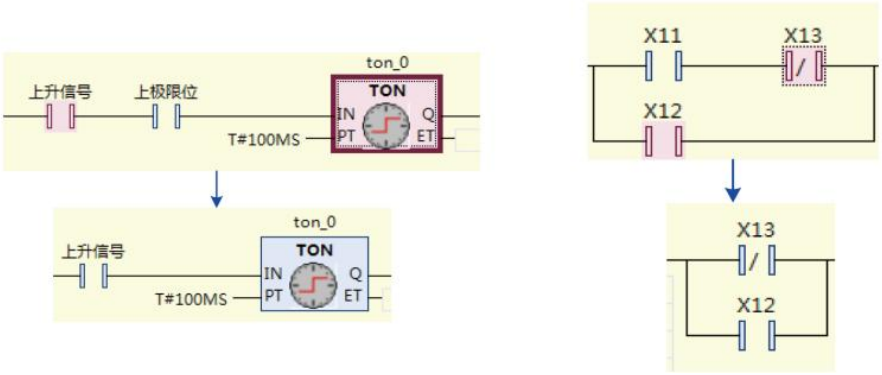
元素选择后元件会处于高亮状态，如触点被选中时显示形式为  ,其中外框的虚线表示触点处于焦点状态。选择的元素可以和其它元素粘贴并联，拖拽串并联。由于多选元素可能不连续，这就需要说明选择元素形成 的结果逻辑。选择结果逻辑的原则是保证原来逻辑的一致性。支持鼠标框选、Ctrl 和 Shift 多选、全选。

元素选择形成的结果逻辑

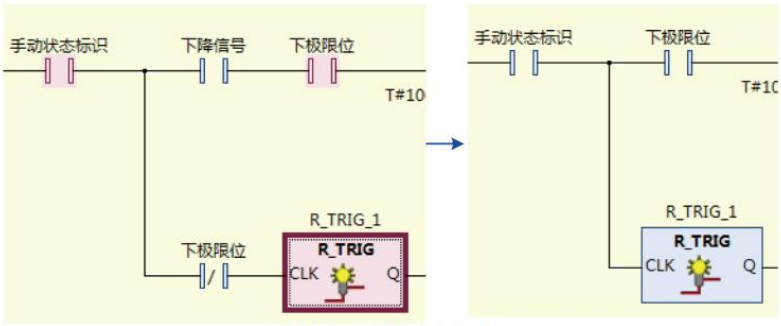
● 单选时：选择触点、线圈，只包含选择的触点、线圈；选择运算块包括运算块本身、无连线输入操作数和输出操作数。如下图，选择 ADD 的运算块，粘贴时会包含 ADD 运算块本身、输入 a、输入 b、输出 c。



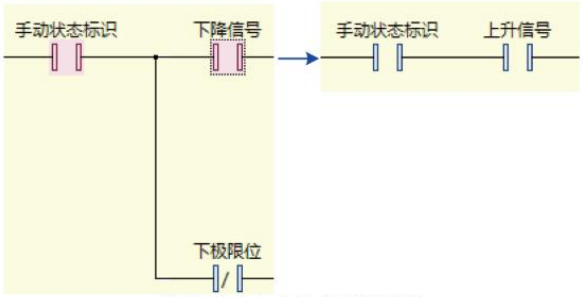
● 多选时：选择一条线上的串联（包含非连续选择）；选择并行线上元素（包含非连续选择），形成并行结果。



● 多选时：如果选择元素跨多分支，形成结果也跨多分支，不改变原来逻辑，但是如果只选择两个分支的， 结果会把两个分支元素串联。

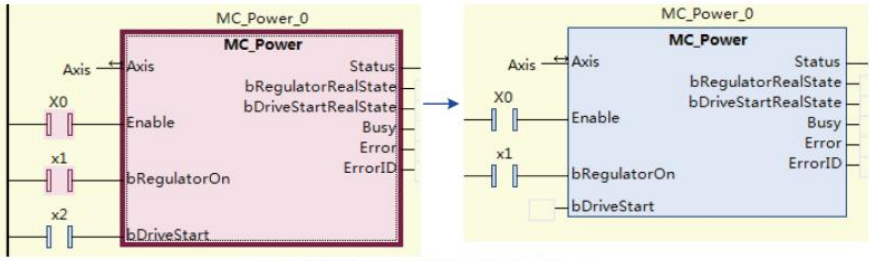


超过两个分支结果也多个分支

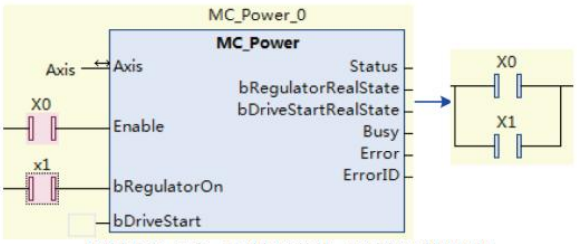


选择两个分支，两个分支结果串联

● 多选时：如果同时选择多输入连线运算块本身和多个输入连线上的元素，结果和选择一致；如果只选择多输入连线上元素，没有选择运算块，则把多个输入连线元素形成打开并联。



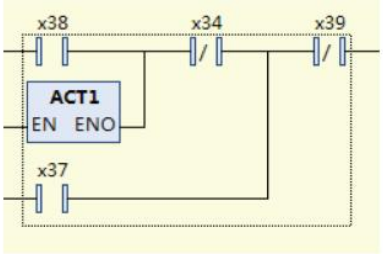
运算块和输入元素都选择，结果和选择一致



只选择输入元素，不选择运算块，形成打开并联逻辑

框选

从鼠标按下位置到鼠标松开位置之间矩形内的元素被选择。如下图：



Ctrl 和 Shift 多选

Ctrl 和 Shift 多选符合标准多选方式：

- Ctrl 多选：按下 Ctrl 时，如果当前元素没有选择则把当前元素增加到选择列表中；如果已经选择，则从选择列表中去除。
- Shift 多选：从上次选择元素到本次选择元素矩形内的元素进行选择。

全选

使用 Ctrl+A 快捷键进行全选功能。全选会把所有网络选择。

6.3.6 标准编辑命令

梯形图标准编辑主要是我们常规编辑操作，如复制、粘贴、删除、剪切、撤销、恢复。使用标准的编辑快捷 键。

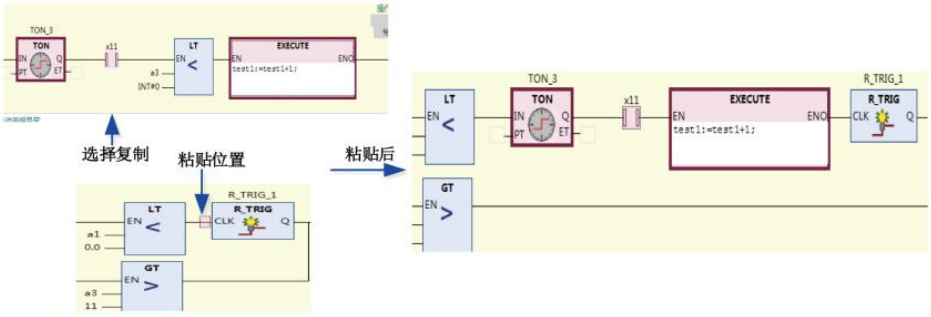
复制

对选择的元素进行复制。复制的结果也就是选择的元素。详见元素选择章节。
梯形图中可以复制的元素包括网络、触点、线圈、运算块、字符串、分支连线。
复制可以复制连续选择的元素，也可以复制非连续的，和选择有关。但是复制的元素只能单个网络内，如果 跨网络选择，只能复制焦点元素网络选择的数据。

粘贴
粘贴是对复制的元素进行粘贴。粘贴规则如下：

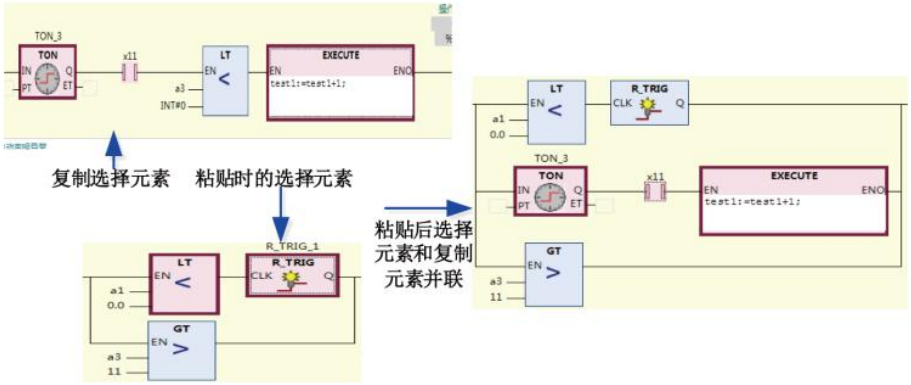
- 连线上粘贴。

在选择连线上粘贴时，连线位置插入复制元素，形成串联关系。



● 元素上粘贴。

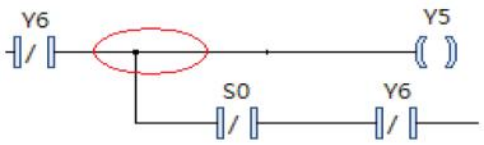
在选择元件上粘贴时，和所选择的批量元件生成并联关系。在形成并联关系时，被选择元件需要满足可并 联条件，才能粘贴，即选择的元件必现满足：在一条线上、必须连续、不能跨越分支、开始元素和结束元 素不能再并联分支内部和外部。



● 线圈位置粘贴。

由于线圈右侧不能存在元素，所以粘贴时有些特殊规则。跳转元素、返回元素规则和线圈相同，下面仅说 明线圈。

1. 如果选择为线圈(选择元素并联)，粘贴的元素位于一个新分支上，此分支位于线圈下面，如下图所示（选中 Y5 粘贴 S0 和 Y6，S0 和 Y6 在一个新分支上，分支位于 Y5 线圈下面）。

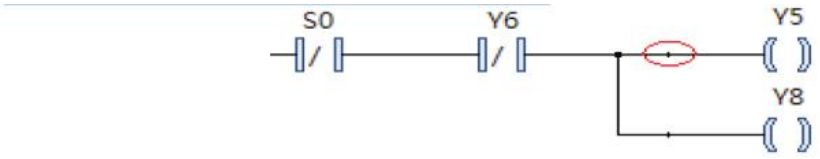


2. 如果选择为线圈之前的连线（选择连线串联），粘贴时，根据复制的元素内容，分为复制内容仅包含一 个分支和复制内容包含多个分支。

复制内容仅包含一个分支

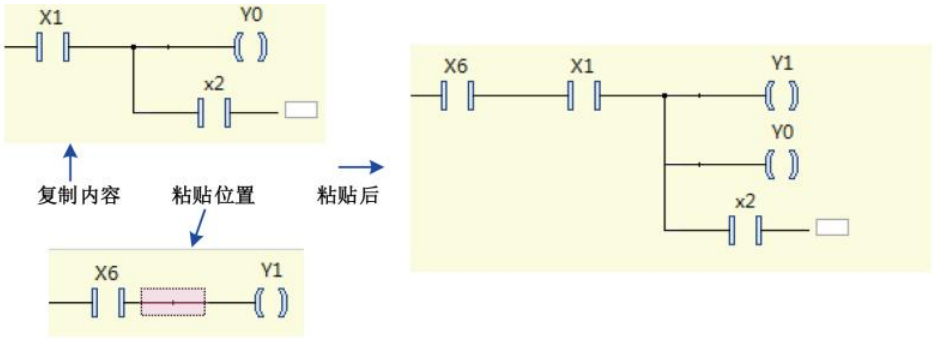
1. 如果复制内容不包含线圈，复制元素直接和线圈串联。

2. 如果复制元素包含线圈，则复制元素线圈之前的数据插入到连线选择位置，复制元素中的线圈和选择连线之后的线圈形成一个非闭合并联，如下图（选择 Y5 之前连线粘贴 S0、Y6、Y8，粘贴后，Y8 之前 S0、 Y6 串联在 Y5 之前连线上，Y8 和 Y5 通过分支并联）。

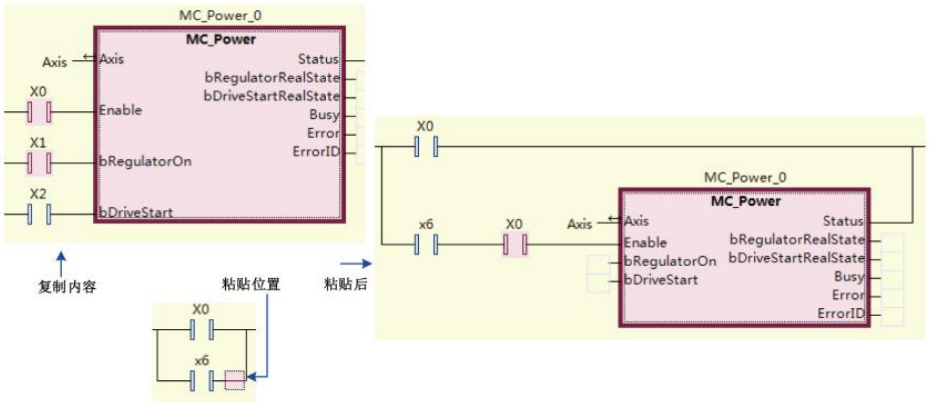


复制内容包含多分支

1. 如果复制内容水平最后一个分支不包含线圈，在选择连线位置插入复制内容。
2. 如果复制内容水平最后一个分支包含线圈，则把复制内容水平最后一个分支的线圈和选择连线之后的线圈并联，其它数据串并联不变，如下图：复制内容为 X1、Y0、X2 粘贴在 Y1 线圈之前的连线位置，粘贴 后，Y0 位于 Y1 线圈下，X1 位于 Y1 之前的连线上。

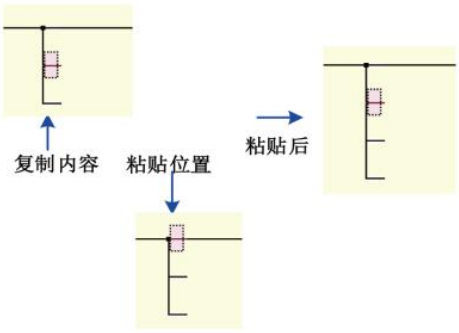


● 多输入连线运算块粘贴。 多输入连线运算块只能在第一个分支非并联位置，所以在粘贴位置，如果不能包含多连线运算块，会删除 复制运算块从第二个输入连线开始的连线元素；如下图，复制的运算块输入 X0、X1 触点，粘贴时 X1 被删除。



● 网络粘贴。可以单选或者多选网络，进行复制粘贴，只有选择了网络才能粘贴复制的网络。

● 单分支连线粘贴。单分支连线粘贴，用于增加单分支，和向上/下插入分支功能相同。复制单分支连线，可以在连线位置粘贴，粘贴在选择连线的下面，如下图所示：

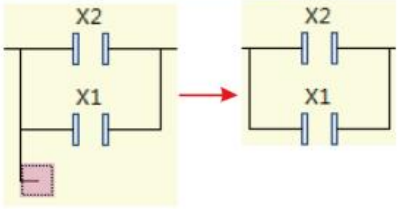


删除

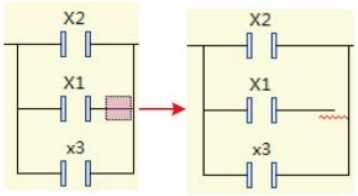
删除是对选择的元素进行删除。元素删除后，会选择下一个元素，以保证操作的连贯性。

删除可分为元素删除和连线删除：

- 元素删除 – 删除本身。
- 连线删除 – 删除分支连线、垂直连线及垂直连线相连的左侧连线。 删除分支连线– 空分支连线删除后，此分支也同时会被删除。



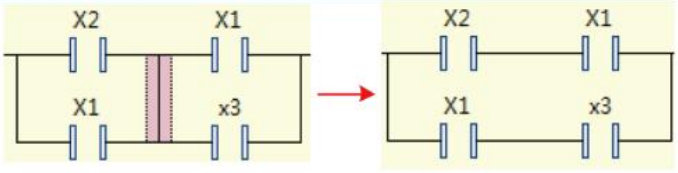
删除垂直连线相连的左侧连线 – 删除垂直连线左侧相连的连线，会把连线断开，分支打开。



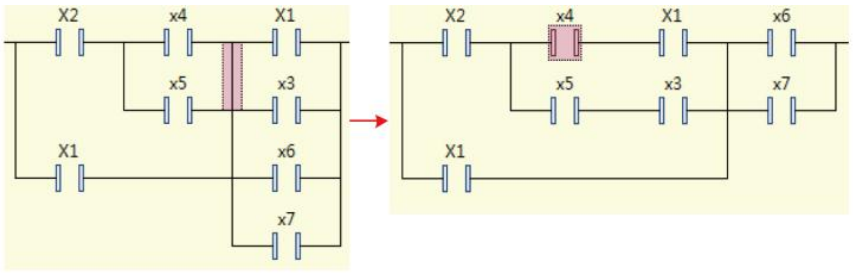
删除垂直连线 – 原则上，删除垂直连线后，此垂直连线不再存在。垂直连线删除后，会产生三个结果：合并分支、打开分支、分支左移。

合并分支

如果选择的垂直连线左右两侧都在分支内，删除垂直连线后，两侧的分支会合并。

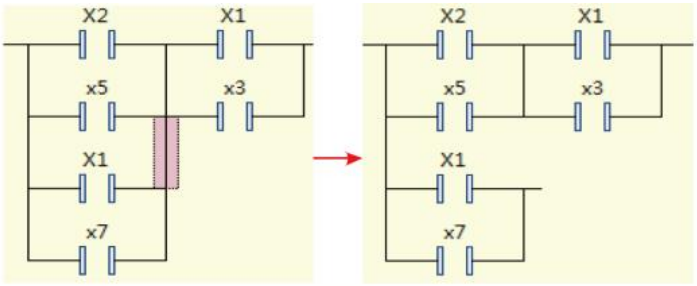


注意：如果删除垂直连线后产生桥式电路，这时会把右侧上下分支左移合并，右侧其它分支下移。 如下图所示，删除后 x1、x3 左移合并，x6、x7 上移。



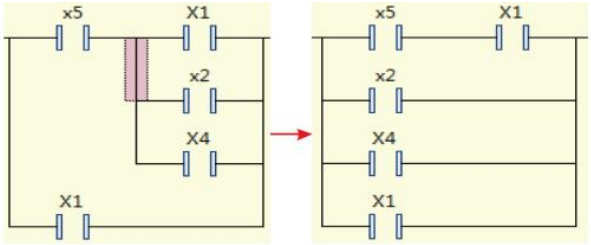
打开分支

如果选择的垂直连线左侧在分支内，右上存在分支，右下不存在，删除垂直连线后会把左侧选择之 下的分支断开。



分支左移

如果垂直连线右侧在分支内，左上存在分支，左下不存在，删除垂直连线后会把右侧选择之下的分支左移到上一个垂直连线上。



剪切

剪切功能是把选择的元素复制，以备粘贴，然后删除。

撤销/恢复

撤销：撤退到上一步编辑状态，并且恢复上一步选择元素。

恢复：恢复到下一步编辑状态，并且恢复下一步的选择元素。

6.3.7 LD 菜单命令

菜单命令可以通过右键菜单或者 LD 工具栏菜单执行的梯形图命令。

插入网络

包含插入网络命令和插入网络(在下方)两个菜单命令，可以通过拖动工具箱中“网络”来插入网络。

命令执行条件：先选择网络，在选择的网络上方/下方插入新网络。

插入网络：图标 -  快捷键：Ctrl + I，表示在选择网络上面插入一个空网络。

插入网络（在下方）：图标-  快捷键：Ctrl + T，表示在选择网络下方插入一个空网络。

切换网络注释状态

图标 -  快捷键：Ctrl + O，把一个网络在注释状态和非注释状态进行切换。

在注释状态，整个网络的代码无效，代码不会执行，执行块也不能编辑。

命令执行条件：选择网络。

插入网络头信息

网络头主要包含网络标题、网络注释和标号。

插入网络头相关命令包含插入标号、编辑网络标题和编辑网络注释。

- 编辑网络标题：图标 -  ，编辑选择网络的标题。
- 命令执行条件：选择网络，并且选项中“显示网络标题”使能
- 编辑网络注释：图标 -  ，编辑选择网络的注释。
- 命令执行条件：选择网络，并且选项中“显示网络注释”使能。
- 插入标号：图标 -  ，给选择网络插入跳转标签，作为跳转元素跳转位置。
- 命令执行条件：选择网络

插入运算块

包含插入运算块、插入空运算块、插入带EN/ENO的运算块、插入带有EN/ENO的空功能块和插入并行运算块（在下方）5个菜单命令，用来插入操作符、功能、功能块和程序，也可以通过拖动工具箱中“运算块”或者“带有EN/ENO的运算块”来插入运算块。

前四个命令用来插入串联运算块，最后一个命令用于插入和选择元素并联的运算块。

插入位置：

1. 选择水平连线，在水平连线处插入一个运算块。
2. 选择元素，在选择元素左侧插入运算块。

- 插入运算块：图标 快捷键：Ctrl + B，弹出输入助手选择一个要插入的运算块。
- 插入空运算块：图标 快捷键：Ctrl + Shift + B，插入一个空的运算块，不弹出输入助手。可以在运算块类型处输入运算块类型。
- 插入带EN/ENO运算块：图标 快捷键：Ctrl + Shift + E，弹出输入助手选择一个要插入的运算块，此运算块带有EN/ENO输入和输出。带EN/ENO的运算块，当EN为TRUE时运算块才执行，为FALSE时不执行，ENO和EN结果相同。
- 插入带有EN/ENO的功能块：插入一个空的运算块，不会弹出输入助手，此运算块带有EN/ENO输入和输出。
- 插入并行运算块（在下方）：在选择的元素下方插入一个空的运算块。选择元素可以是触点、运算块。

插入执行块

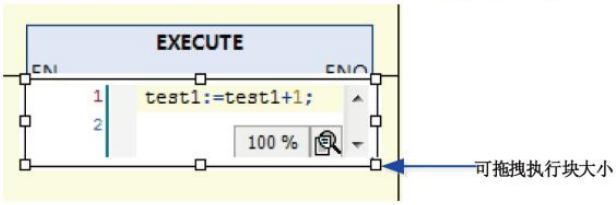
插入执行块：图标 - ，在当前选择位置插入串联执行块，也可以通过拖动工具箱中“执行块”来插入。

插入位置：

1. 选择水平连线，在水平连线处插入一个执行块；
2. 选择元素，在选择元素左侧插入运算块。

执行块是一个可以编辑ST语句的块，单键文本区域进行编辑；执行块只有EnEno输入和输出。

执行块大小拖拽：在编辑状态可以拖拽执行块边框，实现执行块大小控制，如下图：



插入输入

插入输入：图标 - 快捷键：Ctrl + Q，给可变输入运算块增加输入。

可变输入运算块：ADD、+、MUL、*、SEL、AND、&、OR、|、XOR、MAX、MIN、MUX。

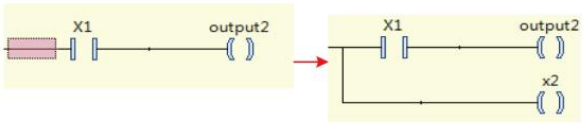
插入位置：当选择输入引脚时，在当前输入引脚之前加入一个输入；选择运算块，增加的输入引脚位于最后位置。

插入线圈

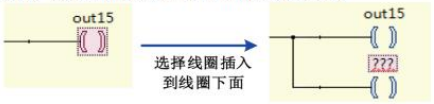
包含插入线圈、插入置位线圈和插入复位线圈三个菜单命令，也可以通过拖动工具箱中“线圈”、“置位线圈”和“复位线圈”元素来插入线圈。

- 命令执行条件：选择位置不能位于并联分支中，也不能位于多输入连线运算块输入位置。
- 插入线圈：图标 快捷键：Ctrl + Shift + A，在当前位置输出一个线圈。
- 插入位置：

1. 选择水平连线或者元素，在水平连线处或者元素左侧插入一个线圈，此线圈和连线通过非闭合分支处理。



2. 如果选择线圈、返回或者跳转，则在当前选择元素下面插入新线圈。



插入的线圈缺省变量为“???”，需要输入所需的变量或常量，可以使用输入助手（功能键<F2>），直接从变量列表中选择输入。

- 插入置位线圈：图标 - ，表示在当前位置插入一个置位线圈。操作方式和上述“插入线圈”相同。
- 插入复位线圈：图标 - ，表示在当前位置插入一个复位线圈。操作方式和上述“插入线圈”相同。

插入触点

包含插入触点、插入常闭触点、插入并联下触点和插入并联上触点四个菜单命令，也可以通过拖动工具箱中“触点”、“常闭触点”元素来插入触点。

- 插入触点：图标 - 快捷键：Ctrl + K，表示在当前位置前串联插入一个常开触点。
  插入位置：

1. 选择水平连线，在水平连线处插入一个触点；
2. 选择一个网络，那么新触点插入到最后；
3. 选择元素，新触点插入到元素左侧。

触点缺省变量名为“???”。点击文本输入所需的变量或常量，可以使用输入助手（功能键<F2>），直接从变量列表中选择输入。

- 插入常闭触点：图标 - ，表示在当前位置串联插入一个常闭触点。操作方式和上述“插入触点”相同。
- 插入并联下触点：图标 - ，快捷键：Ctrl + R，表示在选择元素下并联插入一个常开触点。选择元素可以是触点或者运算块。
- 插入并联上触点：图标 - ，快捷键：Ctrl + P，表示在选择元素上面并联插入一个常开触点。操作方式和上述“插入并联下触点”相同。

插入分支

包含插入分支、在上面插入分支两个菜单命令，也可以通过拖动工具箱中“分支”元素来插入分支。插入的分支是一个非闭合的线。

- 插入分支：图标 - ，快捷键：Ctrl + Shift + V，表示在所选连线位置插入一个分支。
  插入位置：

  1. 选择连线，在连线下方插入一条分支。
  2. 选择触点或者线圈，在选择元素之前插入。

  如下图，每个选择位置表示一个分支。

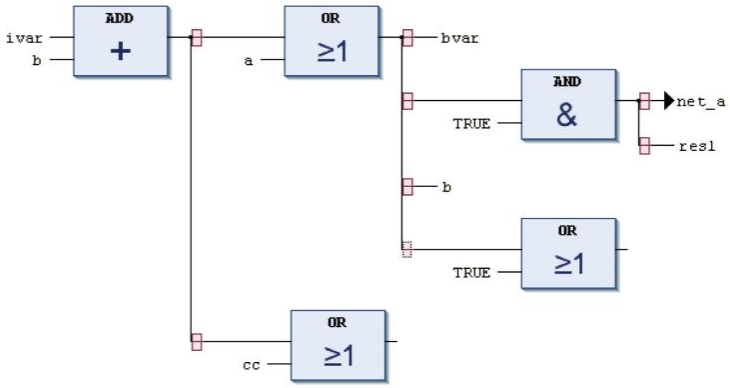


图6-3 分支标记

- 在上面插入分支：图标 - ，表示在所选分支上面增加一个分支。选择分支连线时，才能执行。

跳转和返回

包含插入跳转命令和插入返回两个菜单命令。跳转和返回属于程序执行顺序控制命令，正常程序按照网络顺序从上到下，从左到右依次执行。可以通过拖动工具箱中“跳转”来插入跳转元素或者工具箱中的“返回”来插入返回元素。

跳转、返回和线圈一样，必须在最右侧，所以插入跳转和插入返回的规则和插入线圈一样，详见插入线圈命令。

- 插入跳转：图标 - 快捷键：Ctrl + L，表示插入一个跳转元素，跳转到指定标号位置。
  跳转位置为网络中的标号，也就是说可以从一个网络跳转到另一个网络。当跳转前的输入条件满足时才能执行跳转。
- 插入返回：图标 - ，表示插入一个返回元素。当输入条件满足时，当前POU执行返回，返回到调用它的POU。

取反

图标 - 快捷键：Ctrl + N，对运算块输入、运算块输出、跳转条件、返回条件、触点值或者线圈取反。

取反命令可以在两种位置执行：

1. 元素取反：主要是触点和线圈。取反后触点和线圈内增加一个斜线（/）。
2. 连线取反：主要包含运算块输入连线、运算块输出连线、线圈输入连线、跳转输入连线、返回输入连线，取反后连线处增加一个圆圈。

可取反位置如下图：

图6-4 可“取反”位置

再次执行取反命令，返回到初始状态。

边沿检测

图标 - 快捷键：Ctrl + E，表示对触点、运算块输入连线、线圈输入连线、跳转元素输入连线、返回元素输入连线增加边沿触发功能。

上升沿检测相当于R_TRIG功能块，下降沿相当于F_TRIG功能块。

边沿检测命令可以在两种位置执行：

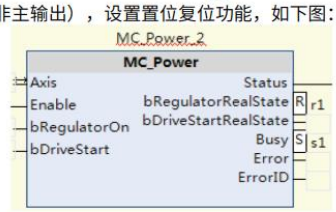
1. 触点边沿检测：选中触点执行边沿检测命令，触点增加边沿检测功能，表示上升沿；表示下降沿。
2. 连线增加边沿检测：运算块输入连线、线圈输入连线、跳转元素输入连线、返回元素输入连线，执行块边沿检测命令，连线上增加延信号符号，上升沿检测符号为；下降沿检测符号为。只有BOOL型输入连线才能添加边沿检测功能。

置位/复位

图标 - 快捷键：Ctrl + M，该命令用于增加置位或者复位输出功能。置位输出显示为“S”，复位输出显示为“R”。可以多次执行此命令，在置位、复位和正常输出之间切换。

置位/复位命令可以在两种位置执行：

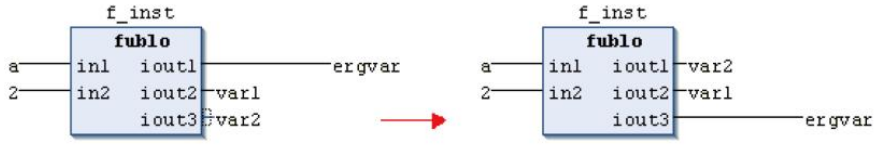
1. 选择线圈，执行此命令变为置位、复位线圈。置位线圈：。复位线圈：
2. 选择运算块BOOL型输出连线（非主输出），设置置位复位功能，如下图：



设置输出连接

图标 - 快捷键：Ctrl + W，当运算块有多个输出时，可以修改主输出的引脚（一个运算块只有一个主输出，主输出和后继的元素相连）。如下图：

选择要修改的输出引脚，执行此命令，修改输出连接。



更改输入、输出引脚显示

包含更新参数和删除未使用的FB调用参数两个菜单命令。

更新参数：图标 - 快捷键：Ctrl + U，表示更新所选的运算块的输入和输出参数。如果运算块的输入或者输出参数发生变化时，通过执行“更新参数”命令，更新运算块的输入和输出参数。

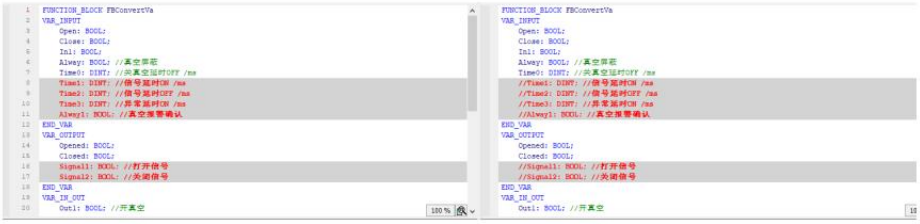
删除未使用的FB调用参数：图标 - ，删除未使用的运算块输入引脚和输出引脚，也就是一个运算块中输入或者输出为“???”或者空时，这些输入和输出将不再显示。

批量更新运算块

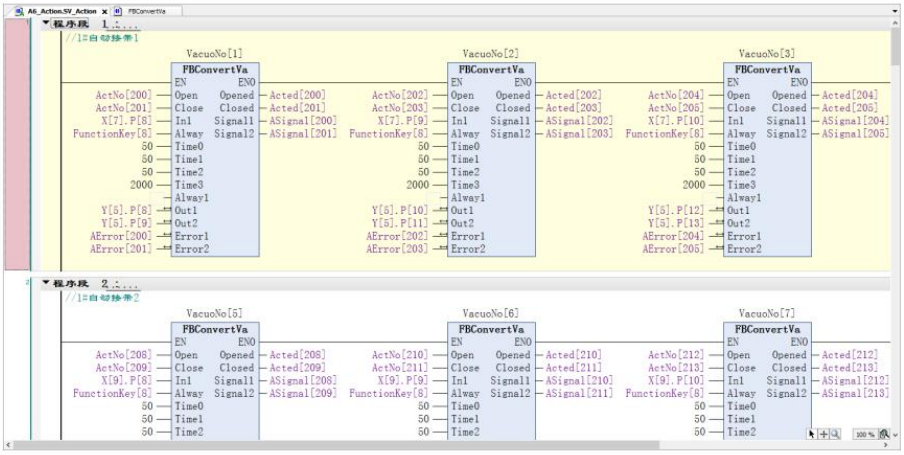
当前打开的是梯形图编辑器时，如果当前编辑器包含的任何运算块的输入或输出参数发生变化，单击工具栏上的“批量更新运算块”，将批量更新当前编辑器中的运算块的输入和输出参数。此命令执行时会对当前编辑器中所有运算块的参数先检查再更新。

示例：

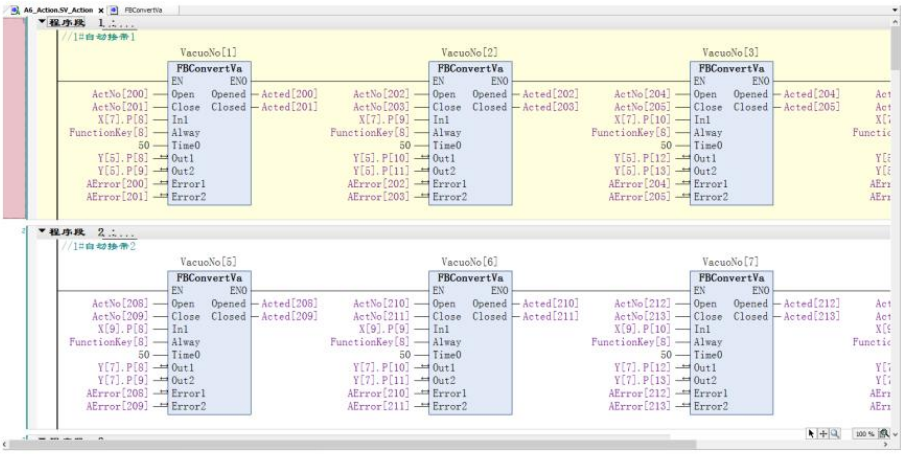
修改FBConVerTVa的输入输出参数：删除输入参数Time1，Time2，Time3，Alway1和输出参数Signal1，Signal2。



更新前：



更新后：



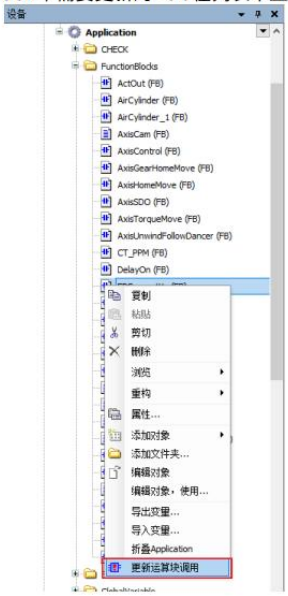
更新运算块调用

在设备树中选择POU节点（只包括函数、功能块、功能块方法、程序）后，在弹出的右键菜单中选择“更新运算块调用”，将更新工程中的所有梯形图中（ST暂不支持）调用该POU的运算块的输入输出引脚。

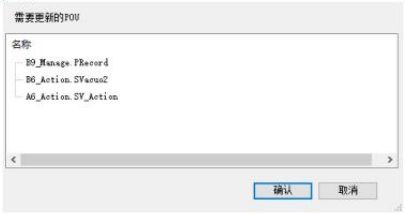
选择“更新运算块调用”后，会查找所有调用该POU的其他POU并对引脚重新计算。如果要做更新会弹出修改列表确认窗口，列出所有需要修改的POU，单击”确认“执行更新，单击”取消“不做更新；如果没有需要更新的POU，不会弹出该窗口。

示例：

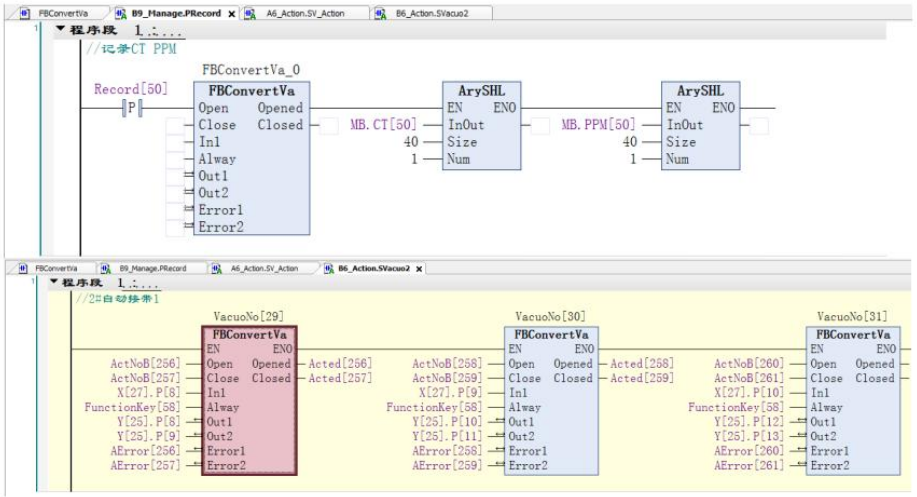
1. 在设备列表中右键单击“FBConvertVa”，选择“更新运算块调用”会将B9_Manage.PRecord，B6_Action.SVacuo2，A6_Action.SV_Action中需要更新的POU在列表中显示。

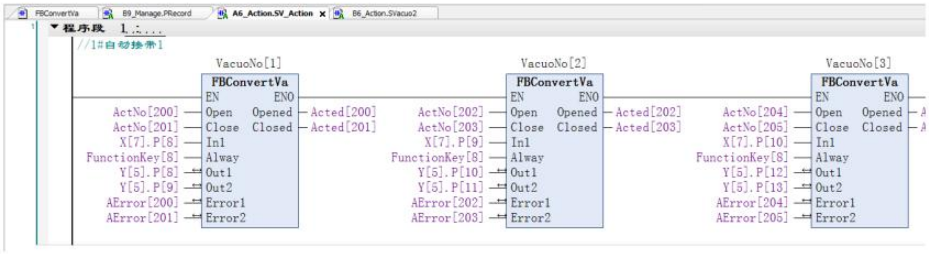


2. 在打开的对话框中单击“确认”。



所有调用FBConvertVa的运算块（B9_Manage.PRecord、B6_Action.SVacuo2、A6_Action.SV_Action）输入输出引脚将更新。





转换为 LD 语言

显示为梯形图逻辑：快捷键：Ctrl + 2。把FBD/IL转换为LD语言；由于FBD、IL暂时不再支持，旧工程可以通过此命令把FBD/IL转换为LD语言显示。

跳转网络

转到…：跳转到指定的网络。弹出跳转网络输入框，指定跳转的网络编号。

编辑操作数注释

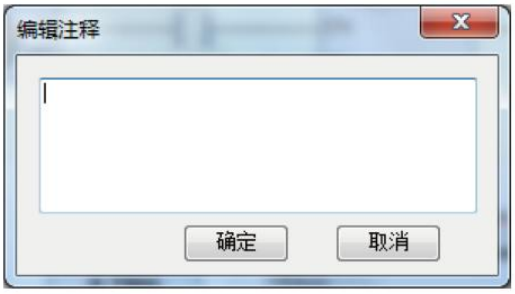
编辑操作数注释：编辑选择的操作数的注释。

命令执行条件：

- 在FBD/LD选项中，激活选项“显示操作数注释”。
- 需要选择操作数字符串。

操作数是逻辑概念，输入变量、常量、地址都是操作数，如运算块输入变量、触点关联变量、线圈关联变量、运算块实例等。

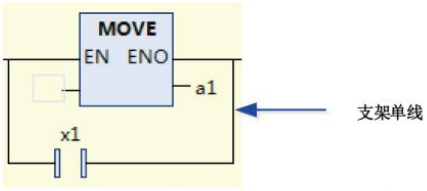
选择操作数字符串，执行此命令显示编辑操作数注释对话框，对操作数注释进行编辑，如下图。



并联模式切换

Toggle Parallel Mode：切换并联分支并联模式。并联模式分为顺序型并联分支和短路型并联分支。

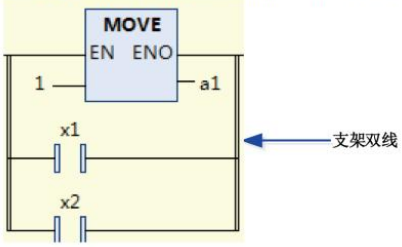
- 顺序型并联的并联支架使用单线，分支输出结果为单个分支输出取或操作，如下图，通过OR来形成分支结果。



- 短路型并联的并联支架使用双线，分支输出结果需要考虑每个分支是否包含非运算块。
  非运算块的分支作为条件，如果非运算块的分支有一个结果为True，都不去执行有运算块的分支（可以理解为触点短路运算块）。如下图，只有X1分支和X2分支结果都不是TRUE，才去执行第一个Move分支指令。

  非运算块分支需要同时满足以下条件：

  1. 此分支只包含触点或者操作符运算块。
  2. 触点不能具有延信号。
  3. 操作符运算块不能是EnEno类型，操作符运算块输入连线不能包含取反、或者延信号。



设置分支起点/终点

设置分支起点/终点：图标 -  ，快捷键：Ctrl + D。

设置分支启动/终点命令用于把启动和终止连接起来，功能同划线功能。

连接两个点，首先执行此命令，设置起点，这时起点位置显示  ，表示连接开始点，然后再选择终止点，执行此命令，这时会把启动和终点连接起来，具体连接逻辑见划线功能。

6.3.8 单键命令

单键命令通过单个字符快捷键进行快速编辑。可以在连线上执行单键命令，也可以在元素上执行单键命令。连线上执行单键插入串行元素；元素上单键命令用于插入并行元素或者元素功能切换。

每个命令对应的字符，可以在【选项】-【FBD/LD】-【LD】页面设置，具体见选项设置。

连线上执行的单键

- 插入触点：默认单键为“C”。
- 插入常闭触点：默认单键为“/”。
- 插入线圈：默认单键为“Q”。
- 插入复位线圈：默认单键为“R”。
- 插入置位线圈：默认单键为“S”。
- 插入空运算块：默认单键为“F”。
- 插入空EnEno运算块：默认单键为“E”。
- Set/Reset/延信号切换：默认单键为“空格”。用于运算块BOOL型输入、输出连线切换；当选择运算块BOOL型输入连线，执行延信号切换；选择运算块非主输出BOOL型连线，执行Set/Reset切换。

元素上执行的单键

- 插入并行触点：默认单键为“C”。选择的元素可以是触点、运算块。
- 插入并行空运算块：默认单键为“F”。选择的元素可以是触点、运算块。
- 插入并行空EnEno运算块：默认单键为“E”。 选择的元素可以是触点、运算块。
- 插入线圈：默认单键为“Q”。 选择的元素可以是线圈、返回、跳转元素。
- 元素取反切换：默认单键为“/”。选择触点进行常开和常闭触点切换；选择线圈对线圈进行取反切换。
- 元素Set/Reset/延信号切换：默认单键为“空格”。选择触点时，进行上升沿、下降沿和正常信号切换。选择线圈进行Set、Reset和正常线圈切换。

6.3.9 划线功能

划线功能主要是把划线起点和终点两个位置连接起来。划线功能首先要满足以下条件：

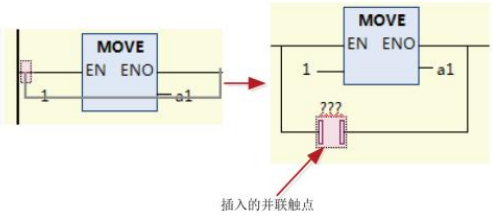
- 起点或者终点位置必须是连线，并且是可以选择的位置（能流线除外）。
- 运算块输入、输出引脚可以拖拽互换位置，所以靠近运算块引脚连线区域（大概11个像素）是拖拽引脚区域，不能划线。

从划线结果来看，划线功能分为三类：划线并联（增加并联分支），闭合分支、拆分分支。

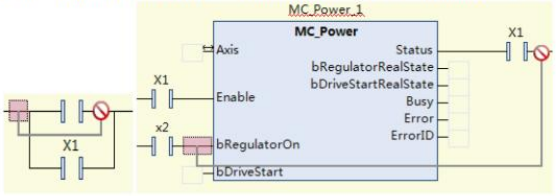
划线并联

当划线起点和终点在一个分支上时，自动增加一个触点和起点、终止并联，如下图：

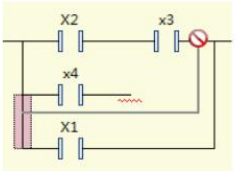
- 如果在一个分支内划线，在开始和结束位置之间自动并联一个触点；
- 如果从打开分支终止连线开始划线到其它分支，表示把打开分支闭合；
- 如果相邻两个上下划线，表示把上下两个分支拆分。



- 划线并联起点和结束点必须满足可并联条件，不能跨越并联分支内外、不能从多连线运算块输入到输出。

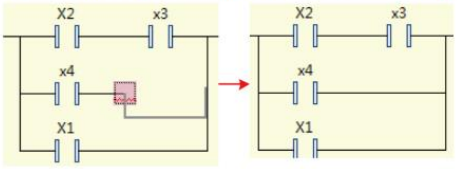


- 不能跨越打开分支。

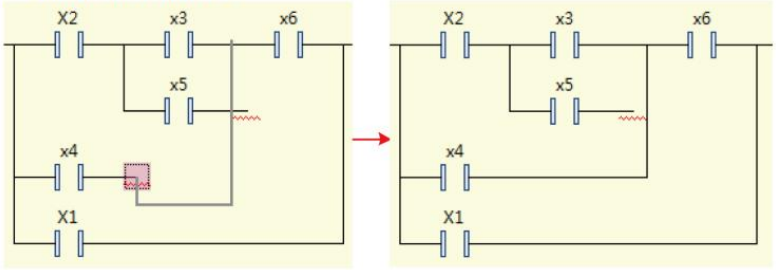


闭合分支

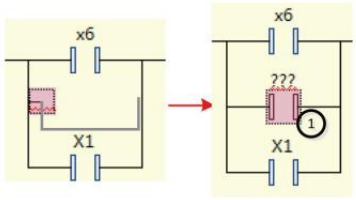
当划线起点为打开分支终止连线时，划线到另外一个可以闭合的连线上会把当前打开分支闭合，如下图：



- 闭合分支功能可以跨越打开的分支。

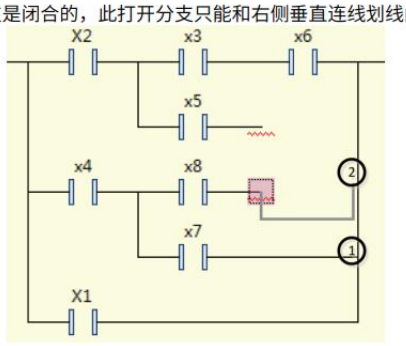


- 如果打开分支只有一个终止连线，闭合时，自动添加空触点。



①：自动添加空触点

- 如果打开分支对应的层级分支是闭合的，此打开分支只能和右侧垂直连线划线闭合。

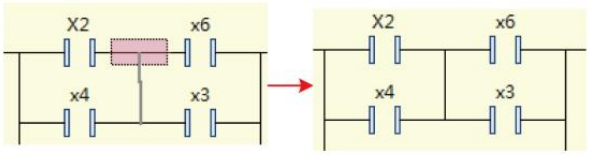


①：打开分支对应的层级分支是闭合的

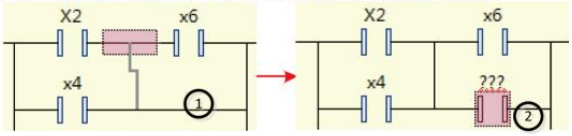
②：只能向右侧垂直连线划线闭合

拆分分支

当划线起点和终点为并联两相邻分支时，划线时会把起点、终点连线两侧拆分为两个并联。



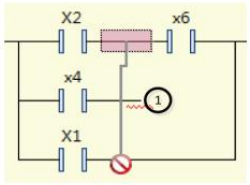
- 如果起点或者终点左侧或者右侧没有元素，自动在没有元素的一侧增加空触点。



①：右侧没有元素。

②：增加的空触点。

- 拆分分支不支持跨越打开分支。



①：打开的分支。

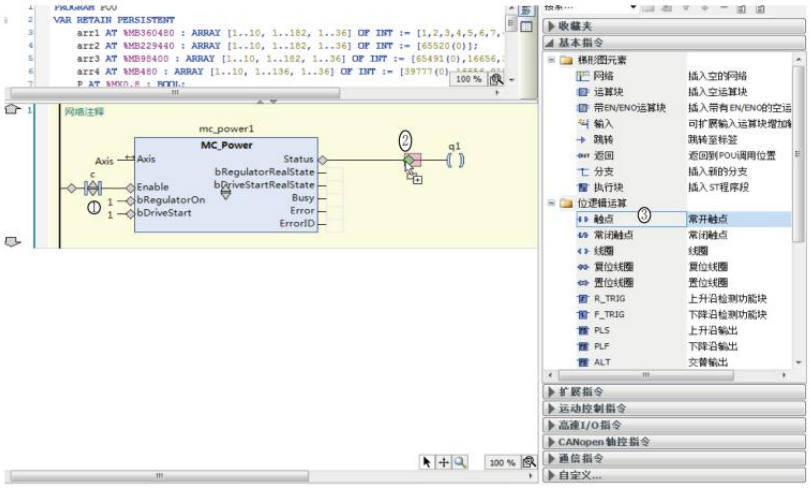
6.3.10 拖拽操作

梯形图可以进行元素拖拽，主要包括工具箱拖拽元素到网络、梯形图界面内元素拖拽及跨界面拖拽。

当拖拽元素时，梯形图界面会显示可拖拽位置。可以拖拽位置有三种显示形式：

- 菱形显示：使用菱形，表示可拖拽到当前位置串联插入。
- 上下三角显示：使用上下三角箭头，表示在当前元素上方或者下方插入并联元素。
- 上下箭头显示：使用上下箭头，表示向上或者向下增加一个网络，并拖拽到新添加的网络。

当拖拽元素到插入位置时，每种图形内部会变成绿色，如，表示要插入到此位置。拖拽显示如下图所示。



① 插入并联　　②拖拽上去后绿色，表示可以插入　　③插入触点

工具箱元素拖拽

工具箱中的元素可以拖拽到梯形图编辑器中。

工具箱元素主要包含基本指令、扩展指令、运动控制、高速I/O，CANOpen轴控指令、通讯指令、POUs；另外用户可以自定义类别并添加指令，还可以把指令添加到工具箱中。

梯形图元素在基本指令中。

POUS主要包含当前工程中定义的程序、功能块、函数、方法、动作。最大显示不能超过200个，如果工程中超过200个，为防止显示混乱，不再显示POUS的内容。

拖拽时，每个元素有限定的拖拽位置，可拖拽规则如下：

- 触点可以拖拽到触点、运算块（包含执行块）上并联，拖拽到连线上串联。
- 运算块可以拖拽到触点、运算块（包含执行块）上并联，拖拽到连线上串联。
- 线圈可以拖拽到非闭合并联、非多连线运算块输入连线上串联，可以拖拽到线圈、返回、跳转上面或者下面。

编辑界面元素拖拽

在梯形图界面，可以拖拽选择的元素从一个位置到另外一个位置。拖拽元素可以在本编辑界面内，也可以拖拽到其它梯形图编辑界面。

拖拽的元素是选择的元素（见元素选择章节），可以多选、单选。

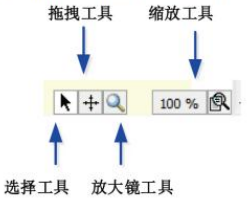
拖拽包含正常拖拽和复制式拖拽（按下Ctrl，拖拽）。正常拖拽，选择的元素拖拽过去后，会把原来选择的元素删除；复制式拖拽，选择的元素拖拽过去后，选项的元素保留。

拖拽功能都是按照标准操作方式实现。

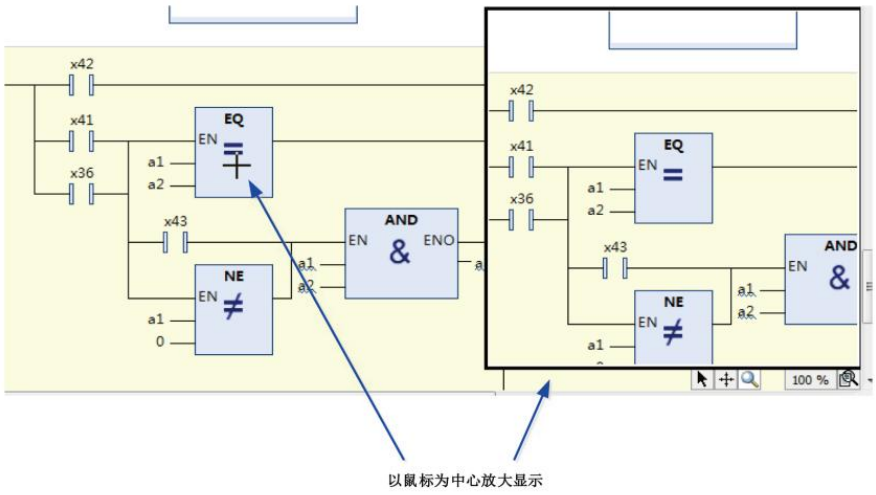
于多选或者单选的拖拽规则，和标准编辑命令（粘贴）一致。

6.3.11 图形显示工具

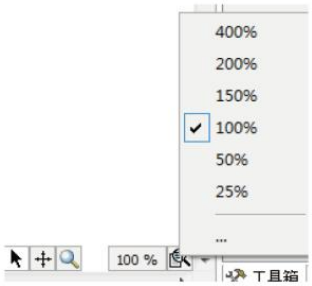
梯形图图形显示工具，用来控制梯形图显示模式，包含选择工具、拖拽工具、放大镜工具和缩放工具，默认梯形图是选择工具模式。图形显示工具位于梯形图界面右下侧，如下图：



- 选择工具
  选择工具是默认显示工具，在选择工具模式下，鼠标样式为，可以进行选择元素，从而进行编辑操作。
- 拖拽工具
  拖拽工具模式下，鼠标样式为，可以对区域进行拖拽显示操作。
- 放大镜工具
  放大镜模式下，鼠标样式为，以鼠标为中心，进行放大显示，有放大镜作用。如下图：



- 缩放工具
  缩放工具可以显示当前界面缩放比例，也可以设置缩放比例，如下图：



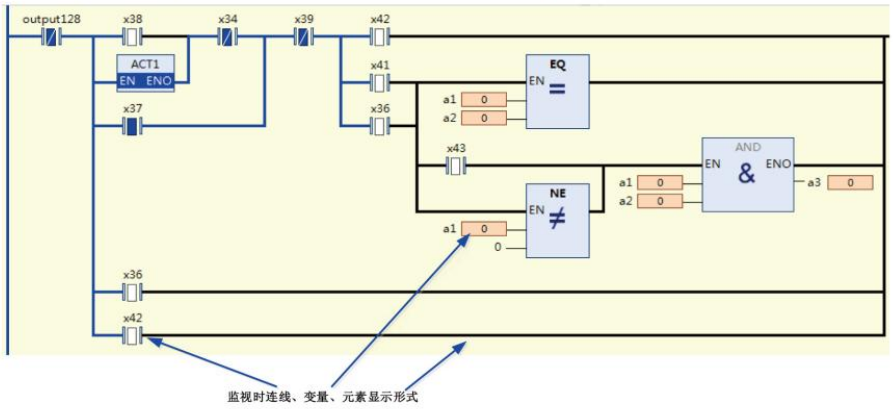
另外点击“…”，弹出缩放比例设置对话框，输入希望的缩放比例，如下图：

6.3.12 LD 调试

梯形图提供了强大的调试功能，除了已有的监控表监控，梯形图还提供了在线模式下的监视、操作数写入、强制值写入、断点和单步调试功能。

监视

在线模式下，梯形图界面中的连线、元素、操作数变量等通过特定的形式来表达执行结果。如下图。



- 监控连线

  1. 对于BOOL型值连线，当导通时（值为TRUE）时，显示蓝色粗线，没有导通时显示黑色粗线。
  2. 非BOOL型值连线（运算块输入、输出中的整形变量、时间类型变量、浮点数变量等），使用细线，并且值为零时使用黑色细线；不为零使用蓝色细线。

- 监控元素

  1. 触点导通时，常开触点显示或者常闭触点显示；触点不导通时，常开触点显示或者常闭触点显示。

2. 线圈导通时，正常线圈显示  或者取反线圈显示  ；线圈不导通时，正常线圈显示  或者取反线圈显示  。
3. 对于EnEno运算块，由于EnEno运算块只有En为True，才执行运算块本身逻辑，为了使EnEno运算块本身能够一目了然了解此运算块是否执行（运算块是否使能），对运算块类型文本显示做了区分，如果此运算块执行了（En输入为True），运算块类型显示黑色字体，没有执行显示灰色字体（运算块禁用），如下图：

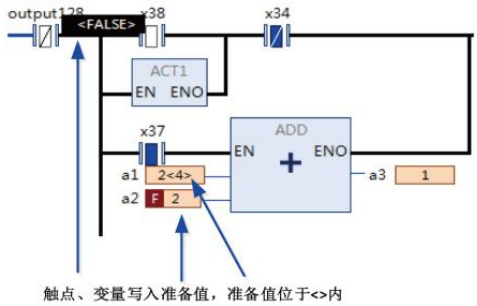
图6-6 运算块执行显示 运算块未执行显示

- 监控变量

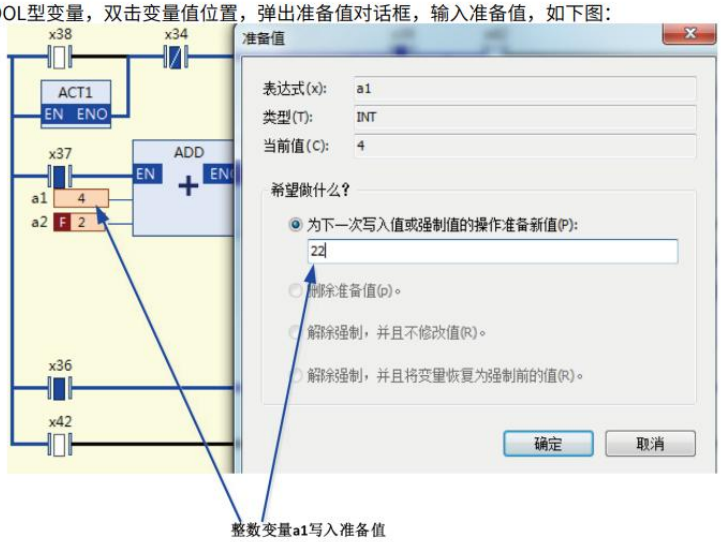
1. 监控变量根据类型不同显示宽度不同，以减少占用空间，对于不定长的，如字符串、枚举类型（显示枚举名），默认长度为12个字符，如果显示不完，使用…替换，通过信息提示来完整显示；对于定长的，如整数、浮点数等根据最大长度显示。
2. 可以拖拽监控变量到监控变量列表。
3. 可以更改变量显示模式，执行菜单命令：菜单【调试】-【显示模式】。

写入和强制

梯形图触点、线圈和变量可以写入准备值，然后执行调试菜单下的“写入值”命令或者“强制值”命令，给变量写入值或者强制值。在写值或者强制值之前，需写入准备值，如下图：



- 对于触点、线圈和BOOL型变量，通过双击元素位置或者变量值位置，进行TRUE、FALSE准备值切换。如双击触点或者线圈中间位置，准备值进行切换。
- 对于非BOOL型变量，双击变量值位置，弹出准备值对话框，输入准备值，如下图：

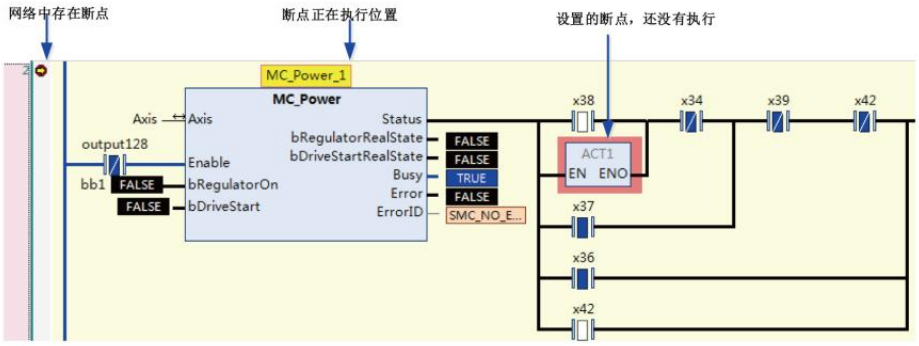


- 写入强制值后，值的最前方，增加 F 标识，标识此值是强制值。
- 把强制值释放，通过菜单命令【在线】-【释放值】

断点

LD支持断点功能，如果添加断点后，程序执行到断点位置自动中断，可以进行程序调试，支持跳入、跳过、跳出、运行到光标位置等操作。

添加断点后，添加断点的位置（元素）用一个浅红色的矩形框表示，当执行到断点位置，正在执行的断点位置用一个黄色的矩形框表示；如果网络中存在了断点，则网络装饰区域的圆点，如下图。



由于梯形图是图形化的，而断点在有逻辑语句的地方才可以添加，梯形图为了优化性能，并不是所有地方都会有逻辑语句，也就是，不是所有地方都能添加断点，例如触点位置、非EnEno操作符运算块位置不能添加断点。

断点一般在变量值可能发生变化的地方、程序的分支处或者另外一个POU调用的地方，如POU，输出变量赋值等地方。可以打开断点对话框（菜单【视图】-【断点】）查看所有可能的断点位置。

断点主要在可以添加到以下位置：

- 网络开始位置，表示网络中第一个可能的断点位置，给网络增加断点时，自动增加到第一个断点位置。
- 不包括非EnEno操作符的运算块，如FB、动作、程序调用、执行块等。
- 线圈、返回、跳转元素位置。

6.3.13 梯形图数据更新

梯形图数据升级有两种方式：

● 在工程打开时，弹出的工程版本信息对话框-【LD/FBD】选项卡，选择下图中所有更新标志，然后点击“确定”按钮；

● 通过菜单命令【工程】-【工程版本信息…】，弹出工程版本信息对话框，切换到【LD/FBD】选项卡，选 择下图中所有更新标志，然后点击“确定”按钮。

如果梯形图数据没有更新，在第一个网络，显示更新说明信息，如下图：

7 版本修订说明

版本号	修订章节	修订内容	修订日期	修订人
V1.0		初版	2024/6	胡坤

本手册如有参数更新，恕不另行通知。

南京德克威尔自动化有限公司
Nanjing Decowell Automation Co., Ltd.

全国服务热线

400-0969016

地址：南京市浦口区兰新路19号瑞创智造园13号楼

网址：www.welllinkio.com

邮箱：sales@welllinkio.com

德克威尔微信公众号

抖音官方账号