雷赛智能
Leadshine
稳定可靠的运动控制专家

2025/07

LeadStudio 编程及应用手册

- 非常感谢您本次购买雷赛产品
- 使用前请详细阅读此说明书，正确使用产品
- 请妥善保管此说明书

前言

资料简介

感谢您选用深圳市雷赛智能控制股份有限公司小型 PLC 产品。本手册提供了雷赛智能小型 PLC 的编程及应用说明。对于初次使用的用户，请认真阅读本手册。若对产品的功能应用和性能方面有所疑惑，请咨询我司技术支持人员以获得帮助。

由于产品的改进，手册内容可能持续更新。

技术热线：400-885-5501

版权说明

版本变更记录

修订日期	发布版本	变更内容
2023/12	V0.1	初版发行
2024/9	V1.0	1、新增以下章节： 1）电子凸轮章节 2）多机互联协议章节 3）HMI符号配置章节 4）数据快照章节

		5）HMI下载程序章节 2、细节优化 3、调整冷热复位对掉电保持变量的影响
2025/2	V1.1	1、新增第5章传统模式应用指导
2025/3	V1.2	1、细微勘误
2025/5	V1.3	1、新增产品型号SC3U
2025/7	V1.4	1、新增V3.1版本的新功能介绍 1）软元件使用表 2）在线编辑、增量下载 3）工程对比 4）批量添加软元件监控 5）轴的起跳速度、全局加减速 6）自动实例化FB 7）Modbus映射模式选择 8）授权码 9）新增软元件R,W,B 10）定时器、计数器使用说明 11）常数的表示方法 12）增量粘贴

目录

1 概览

1.1 产品简介

深圳市雷赛智能控制股份有限公司（简称“雷赛智能”或“雷赛”，股票代码：002979）小型 PLC 产品主要包括：

SC 系列超薄型运动控制 PLC

SCnU 系列面包型运动控制 PLC

本文所述的编程及应用手册主要应用于以上小型 PLC 产品

1.1.1 产品介绍

1）SC2 系列运动控制 PLC 产品

SC2 系列 PLC 自带 2-8 轴 200KHz 高速脉冲输出，具备完备的点位运动控制功能，支持任意 2 轴直线插补，支持对称和非对称 T 型、S 型曲线控制，适合于各轴点位动作的设备控制，如包装机、贴标机、接驳台，各种组装类设备等。

类型	型号	高速输出	高速输入	DI/DO	最多右扩展模块数	通讯口
SC2系列	SC2-C32A2D	2轴 200KHz	4路 200KHz	16DI/16DO	16个（有源）	1个RS232 1个RS485 1个以太网口
	SC2-C32A4D	4轴 200KHz	4路 200KHz	16DI/16DO	16个（有源）	1个RS232 1个RS485 1个以太网口
	SC2-C32A6D	6轴 200KHz	4路 200KHz	16DI/16DO	16个（有源）	1个RS232 1个RS485

						1个以太网口
	SC2-C32A8D	8轴 200KHz	4路 200KHz	16DI/16DO	16个（有源）	1个RS232 1个RS485 1个以太网口

2）SC3U 系列运动控制 PLC 产品

SC3U 系列产品是雷赛自主开发的新一代脉冲型运动控制小 PLC，支持 4～12 路本地脉冲轴。基于雷赛自研的 LeadStudio 编程平台，符合 IEC61131-3 标准，可通过 FB/FC 功能实现工艺的封装和复用。自带 RS485、RS232、以太网接口，可实现多层次网络通信。

类型	型号	高速输出	高速输入	DI/DO	最多右扩展模块数	通讯口
SC3U系列	SC3U-40A4/ SC3U-40AD	4轴 200KHz	4路 200KHz	24DI/16DO	16个（有源）	1个RS232 1个RS485 2个以太网口
	SC3U-40A6/ SC3U-406D	6轴 200KHz	6路 200KHz	24DI/16DO	16个（有源）	1个RS232 1个RS485 2个以太网口
	SC3U-60A6/ SC3U-60A6D	6轴 200KHz	6路 200KHz	36DI/24DO	16个（有源）	1个RS232 1个RS485 2个以太

						网口
	SC3U-60A8/ SC3U-60A8D	8轴 200KHz	8路 200KHz	36DI/24DO	16个（有源）	1个RS232 1个RS485 2个以太网口

1.1.2 系统应用架构

1) 电源连接

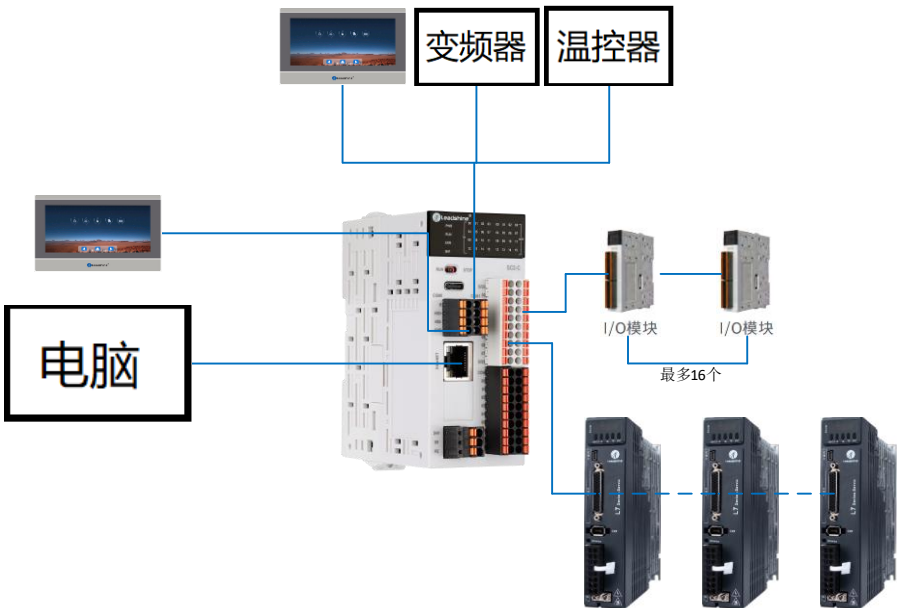
SC2 系列控制器电源电压为 DC24V，需要外部开关电源提供 24V 输入。建议给控制器独立配开关电源供电，避免与其它传感器或设备一起供电。

SC3U系列控制器电源电压为AC220V/DC24V， 例如SC3U-40A4为220V交流供电，SC3U-40AD为24V直流供电。

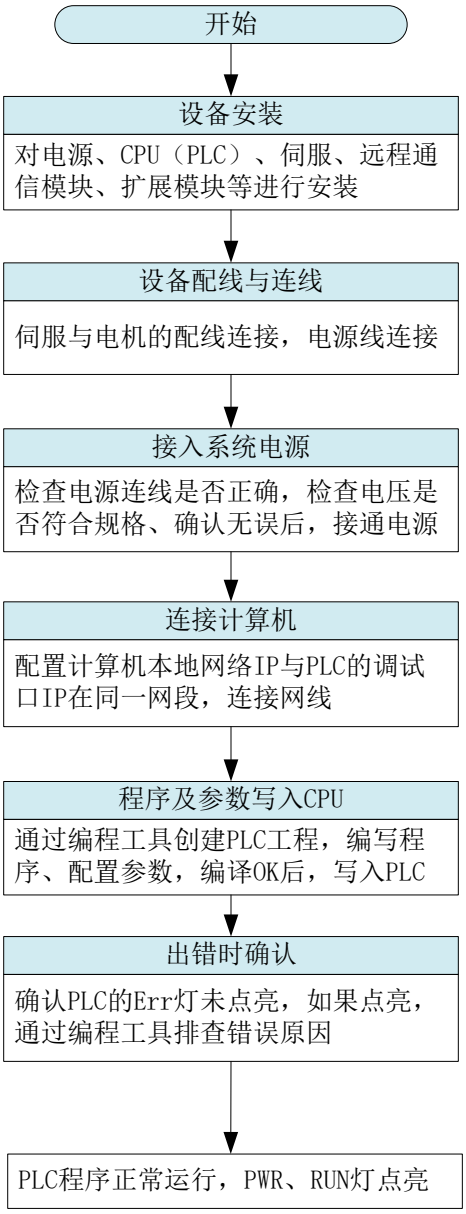
开关电源的负载能力参照下表 SC 系列产品的最大功耗：

产品系列	最大功耗
SC2 系列	20W
SC3U 系列	20W

2) IO 接线、本地模块说明


变频器
温控器
电脑
I/O模块
I/O模块
最多16个

1.1.3 基本使用流程


开始
设备安装
对电源、CPU（PLC）、伺服、远程通信模块、扩展模块等进行安装
设备配线与连线
伺服与电机的配线连接，电源线连接
接入系统电源
检查电源连线是否正确，检查电压是否符合规格、确认无误后，接通电源
连接计算机
配置计算机本地网络IP与PLC的调试口IP在同一网段，连接网线
程序及参数写入CPU
通过编程工具创建PLC工程，编写程序、配置参数，编译OK后，写入PLC
出错时确认
确认PLC的Err灯未点亮，如果点亮，通过编程工具排查错误原因
PLC程序正常运行，PWR、RUN灯点亮

1.2 LeadStudio 软件介绍

1.2.1 LeadStudio 简介

雷赛控制 LeadStudio 软件属于自研软件，包括 PLC 编程、PLC 配置、本地高速 IO 及编码器配置、扩展 IO 配置、EtherCat 等总线配置、运动控制，是一个功能丰富的自动化软件。

LeadStudio 软件还结合了雷赛特有的运动控制功能，将多种复杂运动控制功能封装成工艺库，供用户使用。

1）采用国际标准 IEC61131-3 架构，支持梯形图 LD、结构化文本 ST 两种编程语言。

2）具备方便的产品配置功能，可以轻松快速的实现包括 CPU 配置、通信配置、本地 IO 功能配置等。

3）提供丰富的运动控制功能库和行业工艺库，涵盖多种过程控制和运动控制的应用场景。

LeadStudio 软件从架构上可以分为 3 层：应用开发层、通信层和设备层。

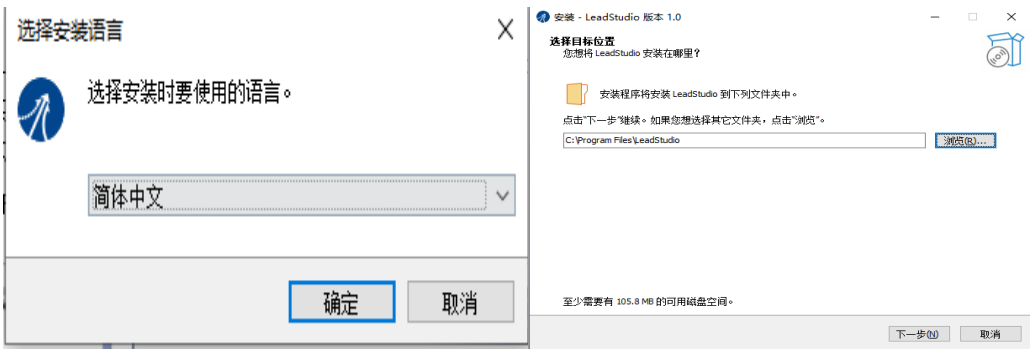
1.2.2 软件的获取与安装

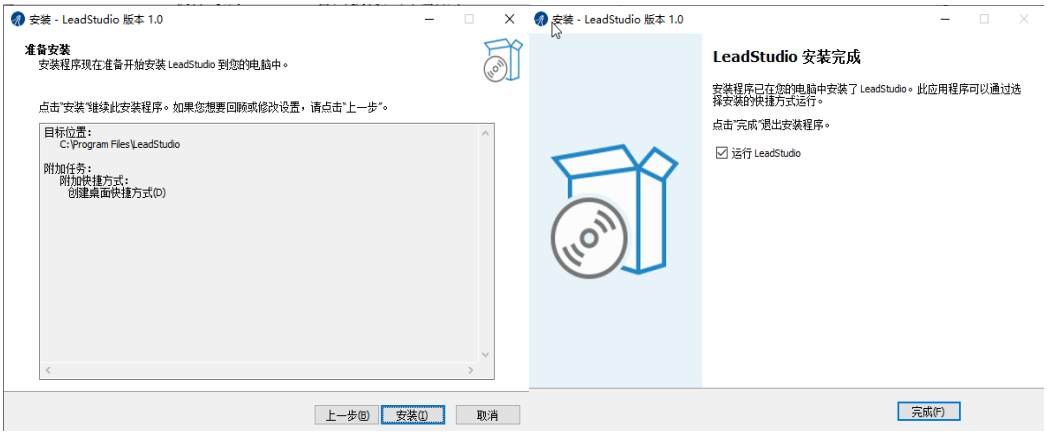
LeadStudio 软件可从雷赛智能官网（https://www.leisai.com）获得。请以最新发布版本为准。

LeadStudio 软件安装步骤如下：

点击安装文件，以 LeadStudio 开头的 exe 文件，例如“leadstudio_setup_r5308.exe”。

然后安装提示，如下图，用户选择所在磁盘空间大的目录作为安装路径，最后点击“安装”，表示成功安装。



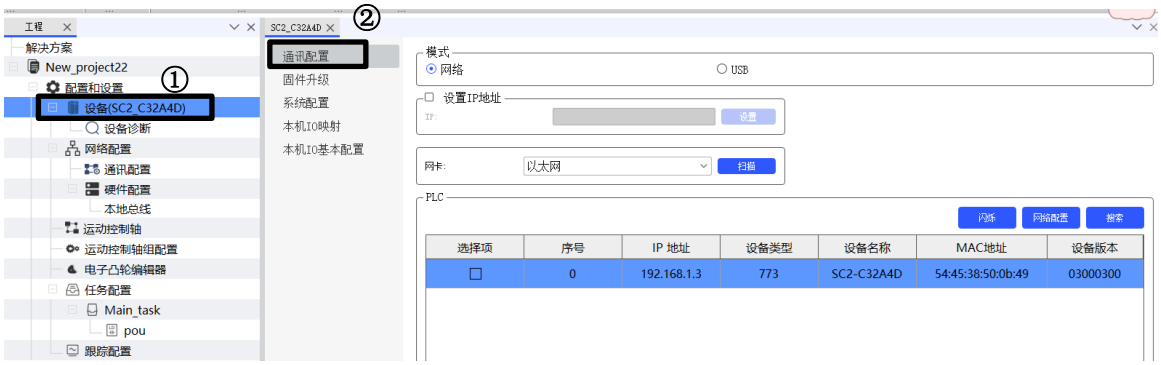


1.2.3 通过 LeadStudio 连接到 PLC

1.通过网线连接 PLC

用网线通过网口将电脑与 PLC 连接起来，然后配置电脑网口 IP 地址与 PLC 设备 IP 地址在同一网段

1）打开 LeadStudio 软件，双击左侧项目树的“设备”，选择“通讯配置”，打开如下界面：



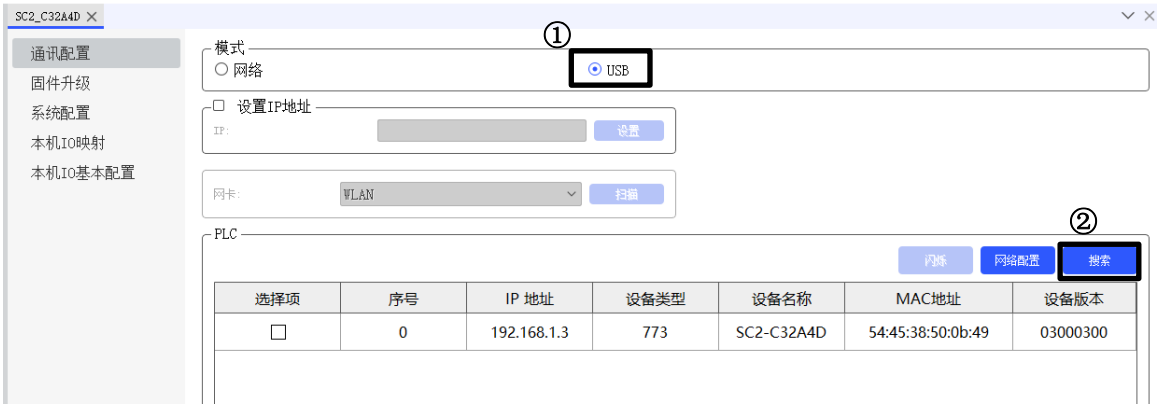
2）选择网络模式-----选择以太网网卡-----点击搜索，即可扫描出 PLC，如下图所示：

3）勾选扫描出来的 PLC，即可连接到 PLC，如下图所示：

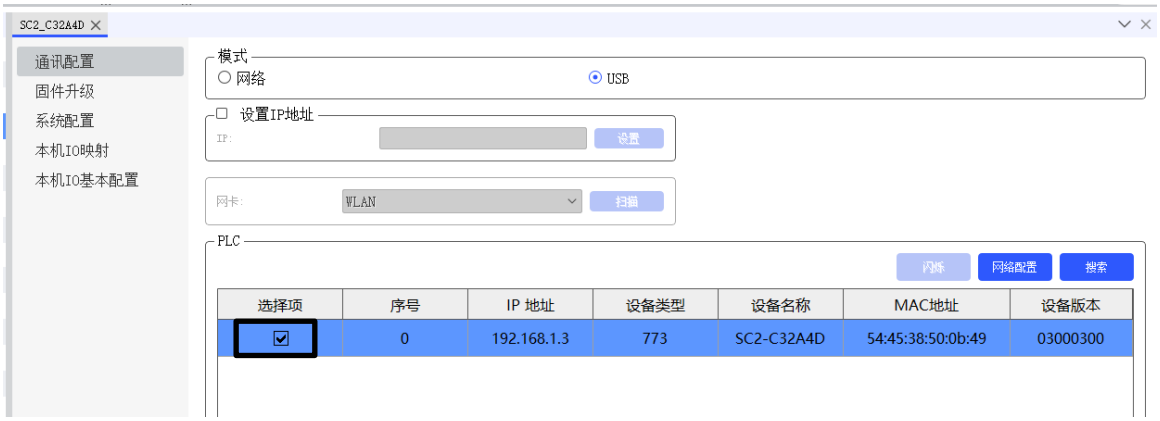
2.通过 USB 连接 PLC

1）打开 LeadStudio 软件，双击左侧项目树的“设备”，选择“通讯配置”，打开如下界面：

2）选择 USB 模式-----点击搜索，即可扫描出 PLC，如下图所示：

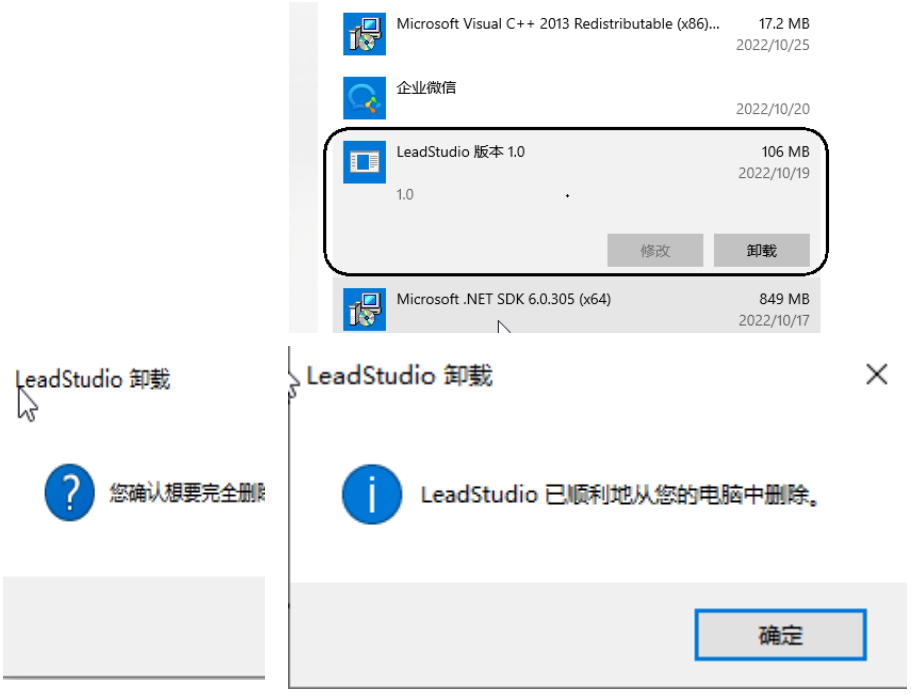


3）勾选扫描出来的 PLC，即可连接到 PLC，如下图所示：



1.2.4 LeadStudio 卸载

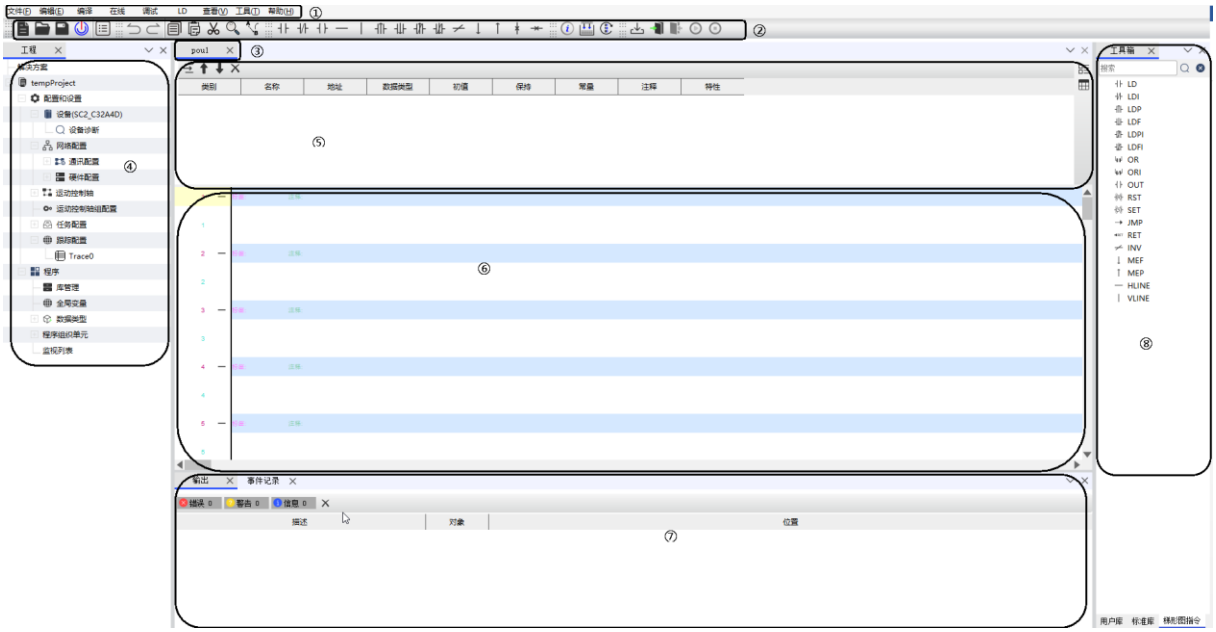
双击控制面板-----应用-----点击 LeadStudio-----点击卸载-----点击是-----确定，即可将 LeadStudio 软件卸载。



2 快速入门

2.1 启动编程环境

在 Windows 系统开始菜单或者桌面快速方式，双击打开 LeadStudio 软件。LeadStudio 的 PLC 工程界面如下：



① 菜单栏，用户可在上面新建工程，打开工程，登录设备，运行 PLC 等选项

② 工具栏，用户可在上面添加并联触点，线圈，上升沿，下降沿等常见的梯形图元件

③ 页标签

④ 设备树

⑤ 变量定义区

⑥ 代码编辑区

⑦ 设备组态和错误消息

⑧ 工具箱

左侧设备树如下：

① 工程名

② Device（设备扫描、固件升级）和设备诊断（诊断设备错误信息）

③ 网络配置（分为网络组态和硬件组态）

④ 运动控制轴（可添加轴）

⑤ 运动轴组配置（可右击配置轴组）

⑥ 任务配置（可右击添加全局变量），Main_Task(即为任务主程序，可在里面添加和设置任务执行时间和执行任务类型)

⑦ 跟踪配置

⑧ 库管理器（用户可在里面添加和卸载库）

⑨ 全局变量（可右击添加全局变量）

⑩ 数据类型（可右击添加结构体、枚举）

⑪ 程序组织单元（用户可以右击新建 POU）

⑫ 监视列表（可右击添加监视列表））

2.2 编写程序的典型步骤

初次使用雷赛小型 PLC 的用户，编写调试一个完整的用户程序需要 5 个步骤：

1) 按照 PLC 应用系统的硬件配置和网络架构进行硬件系统配置。

若用到本地 IO 模块，需要在硬件组态界面上，按照应用系统添加本地模块；

2) 根据应用系统的控制工艺编写用户程序；

3) 配置网络通信的同步周期，根据各任务的实时性要求，配置用户程序单元的执行周期；

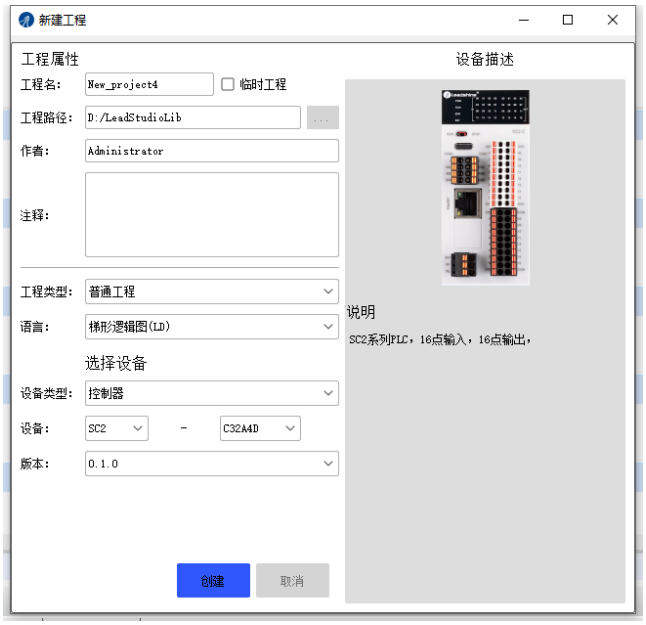
4) 将写好的 POU 按照需求放置到对应的任务中；

5) 在 LeadStudio 编程工具下登录 PLC，下载用户程序，仿真调试、排错，直到程序正确运行，功能正确。

2.3 编写一个样例程序

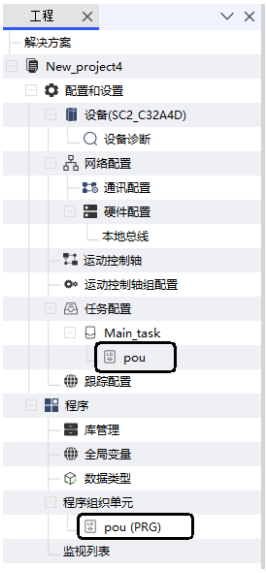
1) 新建工程

打开 LeadStudio 软件-----工程-----在设备类型里面选择“控制器”-----在设备里选中对应的 PLC 型号-----设置工程名称-----选择保存路径-----选择编程语言，如下图所示。



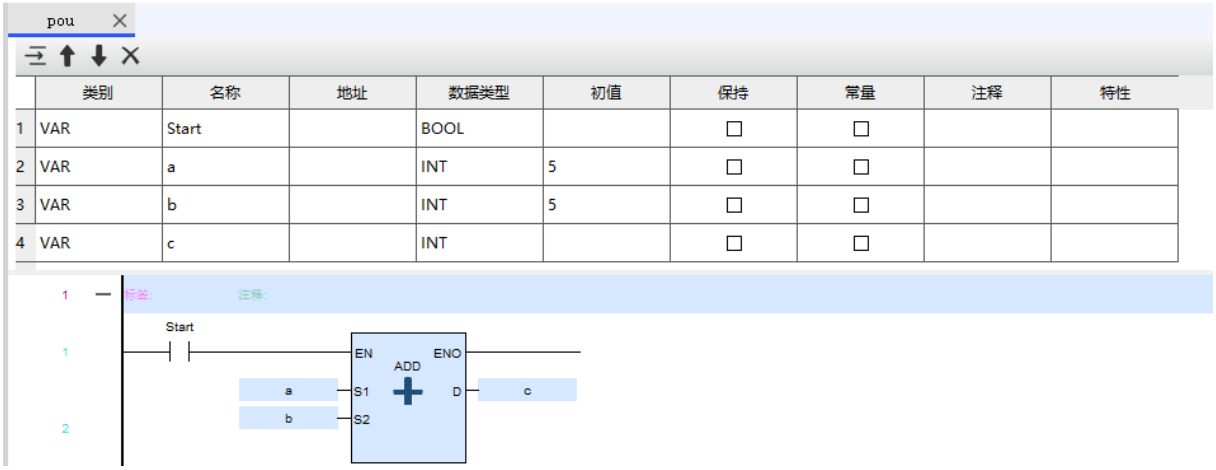
2) 创建程序

新建工程已经默认生成了一个程序“POU”，并且默认加入到了任务配置中，如下图，用户可以对其重新命名，或自行添加新的程序。



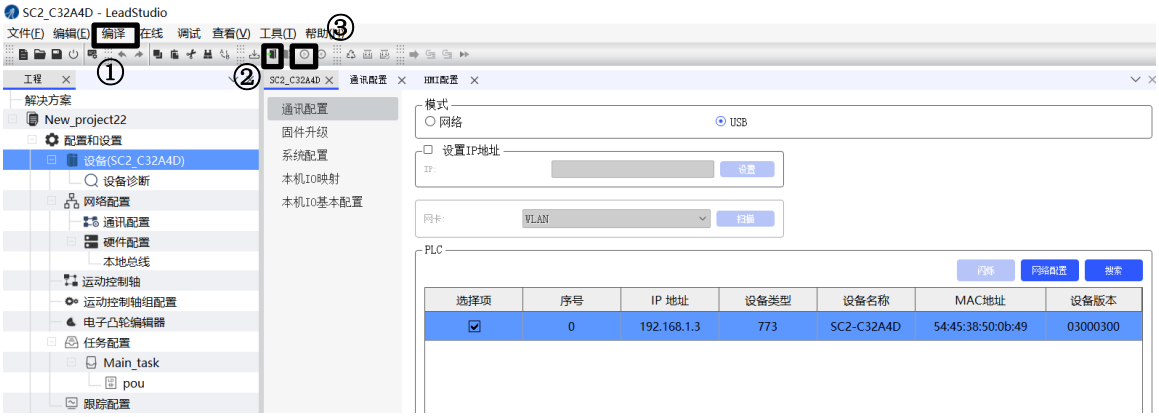
3) 编写程序

先在上面的变量定义区定义一个 BOOL 型变量 start、两个 INT 型变量 a 和 b（并且赋予初值 5 -----然后在代码编辑区，编写一个功能块 ADD（加法运算块），如下图所示:

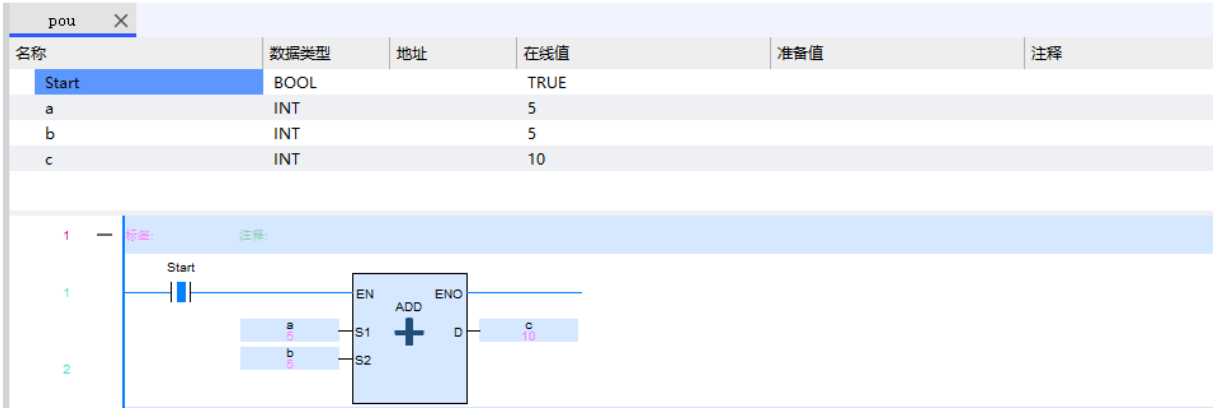


2.4 登录及运行

1）编译-----登录下载-----运行，如下所示：



2）打开程序编程界面，在线情况下可监控到 PLC 的当前变量值



3 编程基础

传统的 PLC 编程方式要求编程者先分配寄存器地址，然后在此基础上命名标签名。这样，编程者在编程之前必须先确定详细的 I/O 点数和中间寄存器数，然后分配相应的寄存器地址，最后再给这些寄存器地址命名标签名，LeadStudio 的编程方式不同于传统的 PLC 编程方式，编程者可以无需考虑先分配具体的寄存器地址，可以直接先声明变量名，在程序的执行代码中使用变量名即可，编程者可以随时将某个特定变量名链接到某个寄存器地址，也可以随时更改链接到该变量名的地址，使整个编程过程更加灵活和方便，大大节省了编程者的开发时间。

3.1 直接地址

3.1.1 直接地址定义

直接地址指映射到控制器设备具体寄存器的地址，包括输出单元 Q 区，输入单元 I 区，存储区单元 M 区, LeadStudio 变量的存储位置可以由用户指定为直接地址，也可以不指定地址，由系统自动分配。直接地址都可以按位，按字节，按字和按双字进行访问，前缀符号如下：

表：前缀符号

前缀符号	定义	数据类型
X	位（BIT）	BOOL
B	字节（BYTE）	BYTE
W	字（WORD）	WORD
D	双字（DWORD）	DWORD

直接地址的定义格式如下：

%< 地址区 >< 前缀符号 >< 数字 >.< 数字 >

SC2/SC2U 控制器直接地址寻址规则如下表，I 区、Q 区、M 区地址寻址规则相同。Word 型的起始地址，遵循的是起始地址为偶数 Byte 地址；DWord 型寄存器的起始地址，遵循的是起始地址为偶数 Word 地址对齐，其索引号为 2 倍关系的原则。这样方

便地址的计算。

表：M/Q/I 区直接地址寻址

M 区直接地址寻址规则			
按 BIT 寻址	**按 Byte 寻址**	**按 Word 寻址**	**按 DWORD 寻址**
%MX0.7~%MX0.0	%MB0	%MW0	%MD0
%MX1.7~%MX1.0	%MB1		
%MX2.7~%MX2.0	%MB2	%MW1	
%MX3.7~%MX3.0	%MB3		
%MX4.7~%MX4.0	%MB4	%MW2	%MD1
%MX5.7~%MX5.0	%MB5		
%MX6.7~%MX6.0	%MB6	%MW3	
%MX7.7~%MX7.0	%MB7		
…	…	…	…
Q 区直接地址寻址规则			
按 BIT 寻址	**按 Byte 寻址**	**按 Word 寻址**	**按 DWORD 寻址**
%QX0.7~%QX0.0	%QB0	%QW0	%QD0
%QX1.7~%QX1.0	%QB1		
%QX2.7~%QX2.0	%QB2	%QW1	
%QX3.7~%QX3.0	%QB3		
%QX4.7~%QX4.0	%QB4	%QW2	%QD1
%QX5.7~%QX5.0	%QB5		
%QX6.7~%QX6.0	%QB6	%QW3	
%QX7.7~%QX7.0	%QB7		

…	…	…	…
I 区直接地址寻址规则			
按 BIT 寻址	**按 Byte 寻址**	**按 Word 寻址**	**按 DWORD 寻址**
%IX0.7~%IX0.0	%IB0	%IW0	%ID0
%IX1.7~%IX1.0	%IB1		
%IX2.7~%IX2.0	%IB2	%IW1	
%IX3.7~%IX3.0	%IB3		
%IX4.7~%IX4.0	%IB4	%IW2	%ID1
%IX5.7~%IX5.0	%IB5		
%IX6.7~%IX6.0	%IB6	%IW3	
%IX7.7~%IX7.0	%IB7		
…	…	…	…

3.1.2 直接地址范围

不同系列的控制器提供的直接地址范围不同。

SC2 系列控制器编程系统提供 128KB（Byte）的输入区域（I 区），128KB（Byte）输出区域（Q 区）和 512KB 存储区域（M 区）。编程时，用户可以直接访问直接地址，也可以定义变量后把变量映射到直接地址间接访问。存储区域使用的地址范围如下表。

表：I/Q/M 区直接地址范围

区域	大小	地址范围
I 区	128KByte	%IW0~%IW65535
Q 区	128KByte	%QW0~QW65535
M 区	512KByte	%MW0~%MW262143

3.1.3 Modbus 地址范围

SC2/SC2U 系列控制器作为 Modbus 从站，支持与 HMI 或者上位机程序 ModbusTCP 或 ModbusRTU 主站通信，Modbus 主站侧访问的直接地址范围如下：

- 可访问的 I、Q 区范围

地址范围	功能码	起始地址	线圈数量	说明
%QX0.0~%QX8191.7	0X01,0x05,0x0f	0	65536	通用标准 Modbus 协议都可以访问
%IX0.0~%IX8191.7	0X02	0	65536	通用标准 Modbus 协议都可以访问

- 可访问的 M 区范围

地址范围	功能码	地址	寄存器数量	说明
%MW0~%MW65535	0x03,0x06,0x10	0~65535	65536	通用标准 Modbus 协议都可以访问

3.2 变量

3.2.1 变量定义

变量可以在 POU（程序组织单元）、GVL（全局变量表）的变量声明编辑器中定义或者通过自动声明对话框定义。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)。

变量声明格式基于 IEC61131-3，应注意以下几点：

1）变量首字符不能是数字，可以是字母或下划线。

2）变量名不区分大小写。

3）变量名不允许出现空格和特殊字符。

4）不能使用关键字、函数和功能块名称作为变量名 。

5）单行注释可以用 “//”表示；多行注释，可以选择“(* *)”。

具体格式如下：

<变量名称> {<直接地址>} <数据类型> {<初始值><注释内容><保持><常量>}

位于大括号{ }内为可选部分。

其中直接地址的变量通过输入寄存器（I）、输出寄存器（Q）和内存区（M）来区分变量存储区域，通过位（X）、字节（B）、字（W）、双字（D）来定义变量类型。

例如在变量声明编辑器中定义一个数字量输入，一个数字量输出，一个整型变量：


GVL

	类别	名称	地址	数据类型	初值	保持	常量	注释
1	VAR_GLOBAL	xIn1	%IX0.0	BOOL		☐	☐	
2	VAR_GLOBAL	xOut1	%QX0.0	BOOL		☐	☐	
3	VAR_GLOBAL	iVar		INT		☐	☐	

图：全局变量声明示例

3.2.2 变量类型

LeadStudio 变量类型符合 IEC61131-3 标准，根据应用范围定义变量属性，也提供了变量的附加属性

局部变量属性见下表：

表：外部访问权限

POU 类型	VAR	VAR_INPUT	VAR_OUTPUT	VAR_IN_OUT
Program	RO （只读）	RW （可读写）	RO （只读）	X （只能在调用程序时存取）
FC	X （只能在调用函数时存取）	X （只能在调用函数时存取）	X （只能在调用函数时存取）	X （只能在调用函数时存取）
FB	RO （只读）	RW （可读写）	RO （只读）	X （只能在调用功能块时存取）

VAR、VAR_INPUT、VAR_OUTPUT 和 VAR_IN_OUT 是在程序组织单元（POU）内部定义的局部变量类型，作用域为当前 POU 内 ； VAR_GLOBAL 是在全局变量表中定义的全局变量，在整个工程范围内都可读写。

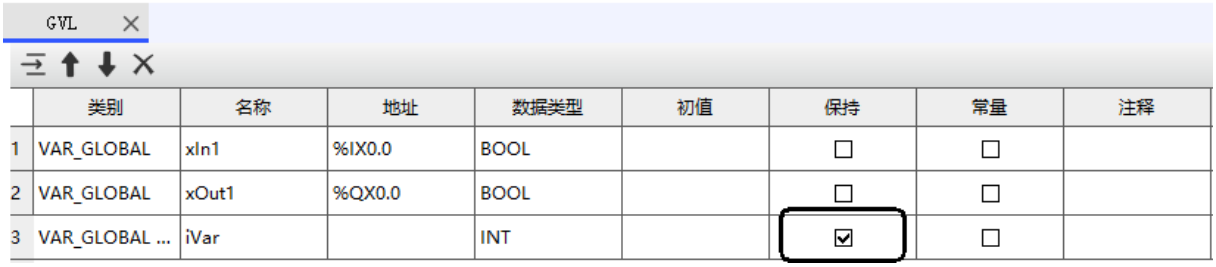
变量的附加属性见下表：

表：变量的附加属性

变量附加属性关键字	变量附加属性
保持	保持型变量，用于掉电保持
常量	常量

1） 保持

勾选变量保持标志。保持型变量在控制器正常关闭、打开（或收到在线命令“热复位”），甚至意外关闭之后仍能够保持原来的值。随着程序重新开始运行，存储的值能够继续发挥作用，举例如下：



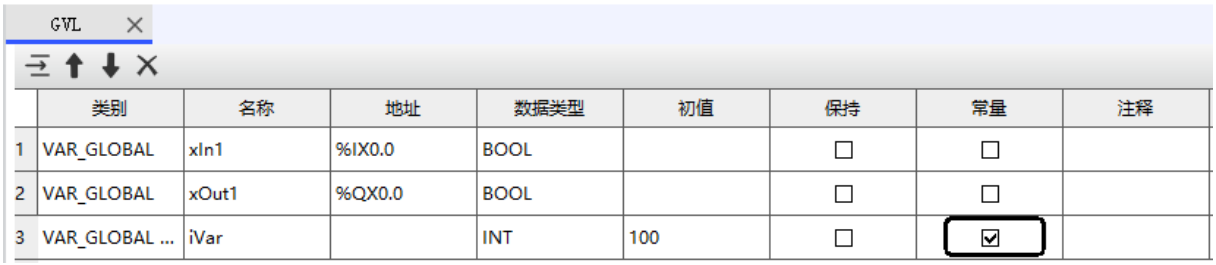
图：保持型变量声明示例

保持型变量在程序下载时可以选择将其重新初始化，或通过在线命令“冷复位”可复位保持型变量。内存存储位置：保持型变量仅仅存储在一个单独的内存区中。

2） 常量

在程序运行过程中只能读取数据而不能进行修改的量称为常量。可以将常量声明为局部常量，也可以声明为全局常量。

在实际应用中，通常可以将一些重要参数或系数设置成常量，这样可以有效地避免由于其他变量对其修改最终影响系统整体稳定性及安全性，举例如下：



图：常量声明示例

程序一旦开始运行，勾选常量标志的变量在程序运行过程中是不允许被修改的。

3.2.3 数据类型

无论声明的是变量还是常量，都必须使用数据类型。数据类型决定了它将占用多大的存储空间及存储何种类型的值。LeadStudio 的数据类型分为标准数据类型和扩展数据类型。

1） 标准数据类型

标准数据类型共分为 5 大类，分别为布尔类型、整型类型、实数类型、字符串类型和时间数据类型，如下表所示：

表：标准数据类型

数据类型	关键字	占用内存	取值范围
布尔	BOOL	8bit	FALSE(0)或 TRUE(1)，
整型	BYTE	8bit	0~255
	WORD	16bit	0~65535
	DWORD	32bit	0~4294967295
	SINT	8bit	-128~127
	USINT	8bit	0~255
	INT	16bit	-32768~32767
	UINT	16bit	0~65535
	DINT	32bit	-2147483648~2147483647
	UDINT	32bit	0~4294967295
实数	REAL	32bit	1.175494351e-38~3.402823466e+38
	LREAL	64bit	2.2250738585072014e-308~1.7976931348623158e+308
字符串	STRING	8×N bit	
时间	TIME	32bit	T#0ms~T#71582m47s295ms

	TIME_OF_DAY		TOD#0:0:0~TOD#23:59:59.999
	DATE		D#1970-1-1~D#2106-02-06
	DATE_AND_TIME		DT#1970-1-1-0:0:0~DT#2106-02-06-06:28:15

2） 扩展数据类型

扩展数据类型包括结构体、枚举体、指针、数组等。

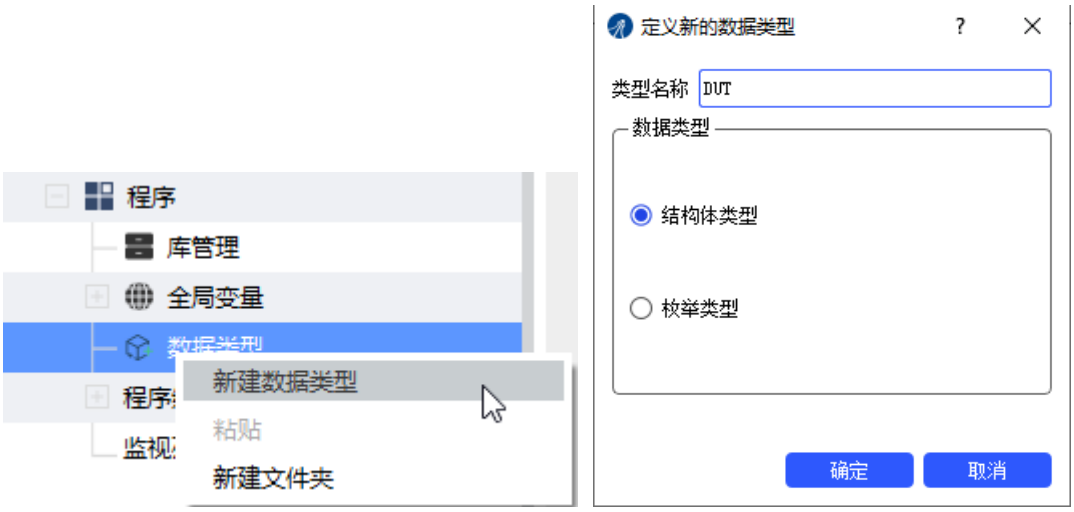
表：扩展数据类型

数据类型	关键字	占用内存	取值范围
结构体	STRUCT	根据定义	根据定义
枚举	ENUM	根据定义	根据定义
指针	POINTER TO	根据定义	根据定义
数组	ARRAY [?..?] OF	根据定义	根据定义

3.2.4 特殊数据类型

特殊数据类型也称为用户自定义数据类型，包括结构体、枚举体、指针、数组。

右键单击工程管理树中的“数据类型”，选择“新建数据类型”，在弹出对话框中，选择需要添加的数据类型，给定名称后，点击“确定”进行创建。

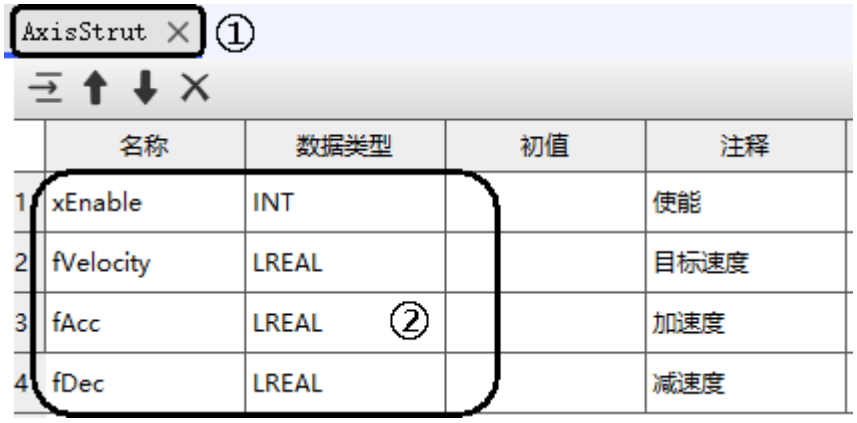


图：新建结构体/枚举体

1）结构体

结构体是由一系列相同或者不同类型的数据构成的数据集合，是一种用户自定义数据类型。用户有时候需要将不同数据类型的数据组合成一个整体对外引用。例如一台伺服轴的控制通常都包括轴使能、运行速度、加速度、减速度、目标位置等信息，这些信息都和这台伺服轴关联，以独立变量进行声明很难反应变量和轴的内在关系，此时就可以使用结构体来管理。

例如定义一个 AxisStruct 轴结构体如下：



图：AxisStruct 轴结构体

① 结构体名；②结构体元素定义。

结构体在程序中的调用需要首先在变量声明区实例化，之后可以直接在程序中使用声明好的结构体变量，结构体变量中的元素可以通过“.”索引的方式调用，如下所示，通过 Axis_1 实例访问元素：

图：结构体实例化调用

2）枚举

枚举是若干个被命名的整型常数的集合。枚举类型在程序中经常使用，如为了方便识别程序执行的步序，步序就可以定义为枚举。枚举默认为 INT 类型。枚举类型在没有

赋初值的情况下，从 0 开始按照 INT 整型数据递加。整数可以直接赋值给一个枚举类型。

枚举类型的定义如下：

图：枚举示例

①枚举名；②枚举元素定义。

枚举体在程序中的调用需要首先在变量声明区实例化，之后可以直接在程序中使用声明好的枚举体变量，如下所示：

图：枚举实例化调用

3）指针

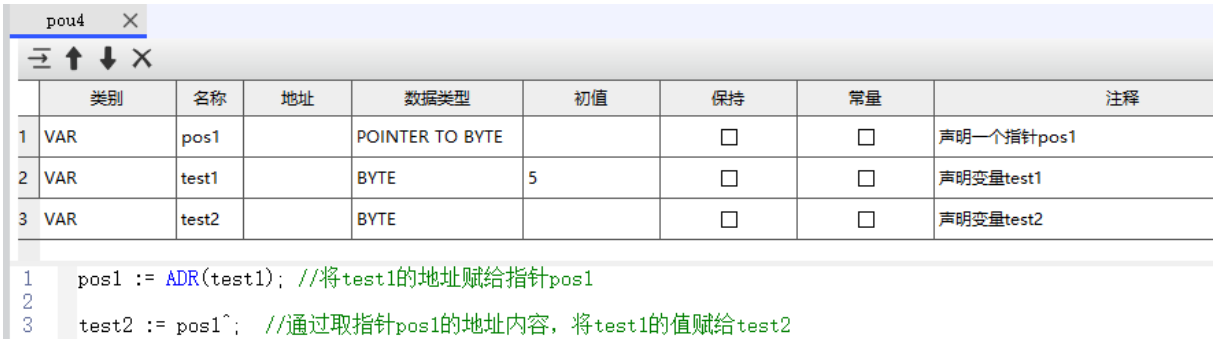
指针属于扩展数据类型，可以在 POU 的声明部分或者全局变量组中定义指针；指针用来在应用程序运行时存储变量、功能块和函数的地址。它可以指向上述的任何一个对象以及任意数据类型，包括用户定义数据类型。

指针类型变量的定义如下:

图：新建指针类型变量

指针变量的数据类型为：POINTER TO <基本数据类型，扩展数据类型，功能块，函数>；

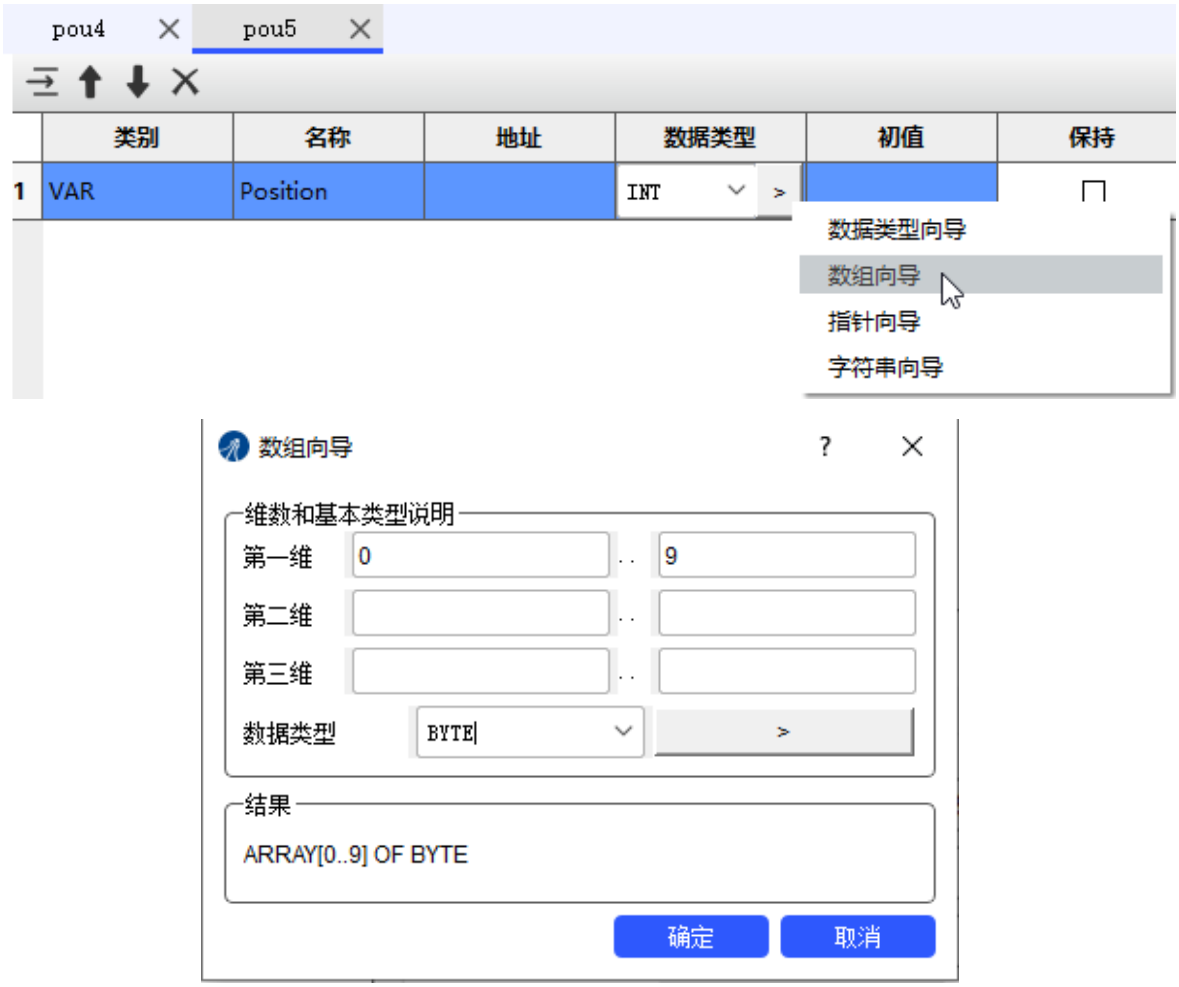
取指针地址内容即意味着读取指针当前所指地址中存储的数据。通过在指针标识符后添加内容操作符“^”，可以取得指针所指地址的内容； 通过地址操作符 ADR 可以将变量的地址赋给指针。如下图所示：



图：指针操作示例

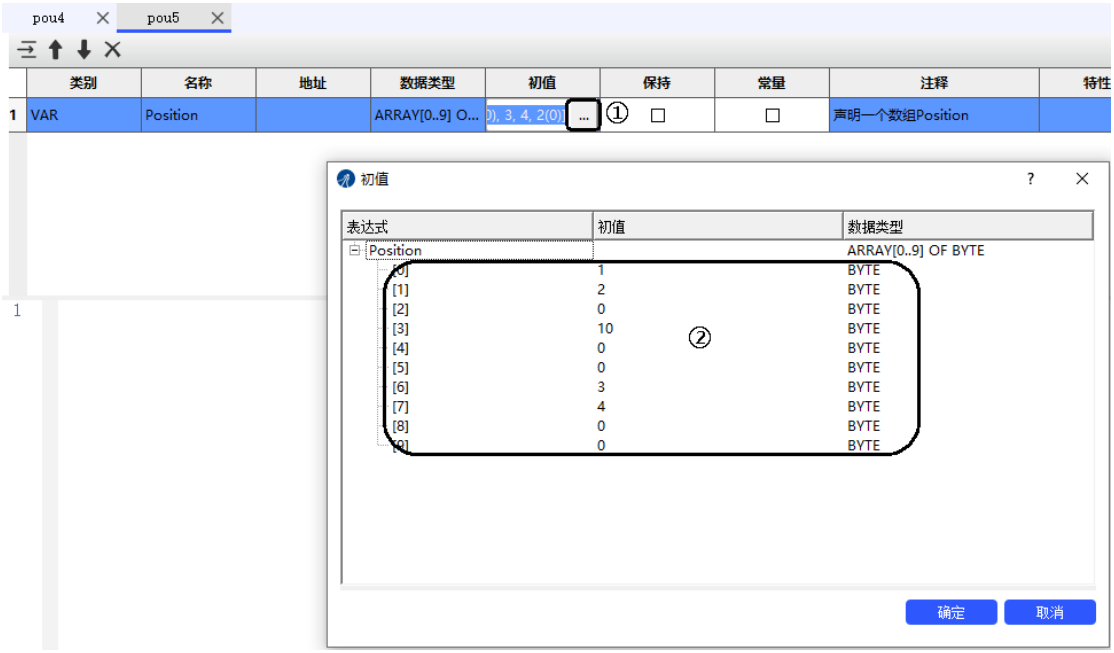
4）数组

一维、二维和三维数组属于基本的数据类型。可以在 POU 的声明部分或者全局变量组中定义数组。数组类型变量的定义如下：



图：新建数组类型变量

数组的初始化如下图所示：



图：数组初始化

3.2.5 掉电保持变量

在工业现场控制器实际运行的过程中，需要设备异常断电或关机以后，控制器内部的某些变量值能够保存在控制器内部，在下次上电开机后，这些数据可以被调出，并保持断电前的数据被程序使用。LeadStudio 在声明变量时可以勾选“保持”，即声明该变量为掉电保持型变量，当使用不同复位指令时，反应是不同的，详见下表：

在线命令	保持
掉电	√
复位	√
热复位	√
冷复位	×
程序下载	×/√

注： × = 恢复初始值； √ = 保留值； ×/√：可选择恢复初始值或保留值。

用户可以通过勾选“保持”标志来添加掉电保持变量。在变量声明器中勾选“保持”即可。

pou6

	类别	名称	地址	数据类型	初值	保持	常量	注释
1	VAR	xIn1	%ix0.0	BOOL		☐	☐	数字量输入
2	VAR	xOut1	%QX0.0	BOOL		☐	☐	数字量输出
3	VAR RETAIN	ivar		INT	100	☑	☐	整型变量

图：掉电保持型变量声明示例

3.2.6 全局变量和局部变量

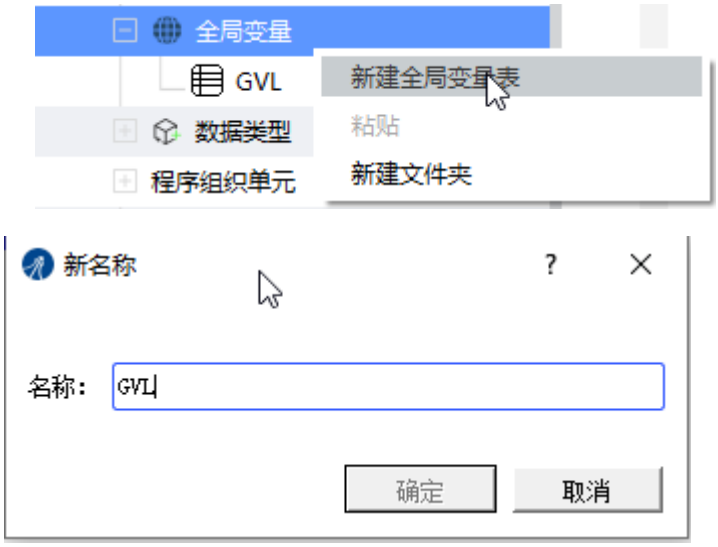
变量定义的范围确定了其在哪个程序组织单元（POU）中可以被调用，从范围上来划分，分为全局变量和局部变量。每个变量的范围由它被声明的位置和声明所使用的变量关键字所定义。

1）全局变量

在全局变量表中定义的变量为全局变量。全局变量可以被工程中其它程序组织单元所共用。需要注意的是，一个系统中不能有两个同名的全局变量，所有的全局变量都需

要在全局变量组中声明，全局变量实现了两个不同的程序和功能块之间非常便捷的数据交换。

用户可以通过全局变量表添加全局变量。在工程管理树中右击“全局变量”，选择“新建全局变量表”，在弹出的对话框中输入全局变量表名称，然后单击“确定”按钮即可。



图：新建全局变量组

在全局变量表中定义全局变量的示例如下：

GVL

	类别	名称	地址	数据类型	初值	保持	常量
1	VAR_GLOBAL	xIn1	%IX0.0	BOOL		☐	☐
2	VAR_GLOBAL	xOut1	%QX0.0	BOOL		☐	☐
3	VAR_GLOBAL	g_bEnable		BOOL		☐	☐
4	VAR_GLOBAL ...	iVar		INT	100	☐	☑

图：全局变量声明示例

2）局部变量

在一个程序组织单元（POU）内定义的变量为内部变量，它只在该 POU 内有效，这些变量也称为局部变量。用户可以在多个独立的程序中使用同名的局部变量，因为它们之间没有映射关系，互不影响。局部变量的定义如下所示，pos1 为 POU_T 内的一个局部变量。

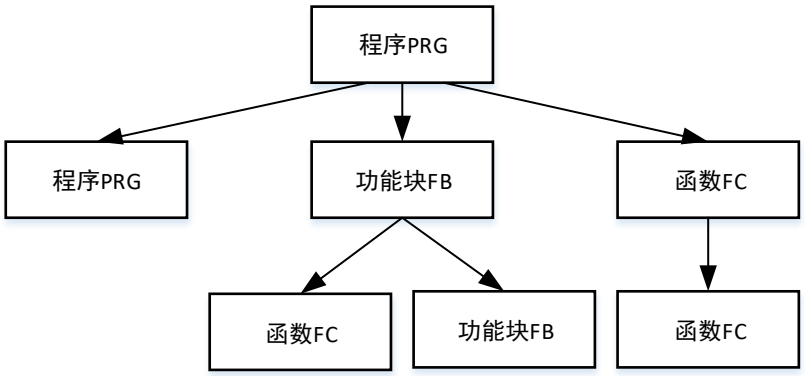
POU_T

	类别	名称	地址	数据类型	初值	保持	常量
1	VAR	pos1		LREAL		☐	☐

图：局部变量声明示例

3.3 新建 POU

POU 是程序组织单元，由声明区和代码区两部分组成，是用户程序的最小软件单元。LeadStudio 软件编程时创建的 POU，按功能划分，可以分为函数（FUN）、功能块（FB）和程序（PRG）。用户编写程序时，FUN、FB 和 PRG 三者之间可以相互调用。PRG 可以调用 FUN、FB 和 PRG；FB 可以调用 FB 和 FUN；FUN 仅能调用 FUN。三者之间的调用关系如下图：

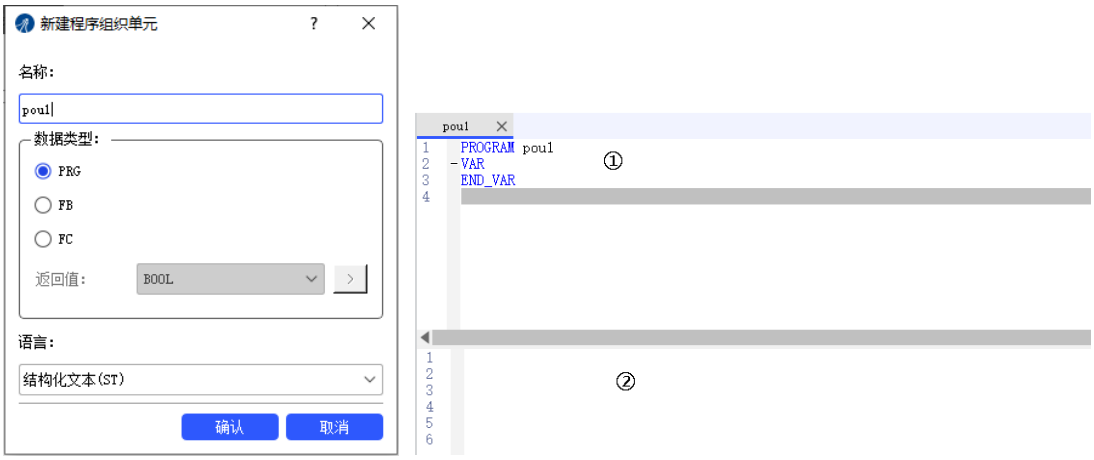


图：程序/功能块/函数调用关系

在工程管理树中，右键单击“程序组织单元”，选择 “新建 POU”，如下图所示：

图：新建 POU

在弹出对话框中，用户可以选择添加程序、功能块或函数，“实现语言”下拉可以选择对应的编程语言。



图：选择编程语言

①：变量声明区；②：代码编辑区。

创建 POU 之后，用户可以在代码区编写自己的程序代码，程序中使用的变量都需要在变量声明区定义。

3.4 功能块与函数

3.4.1 功能块

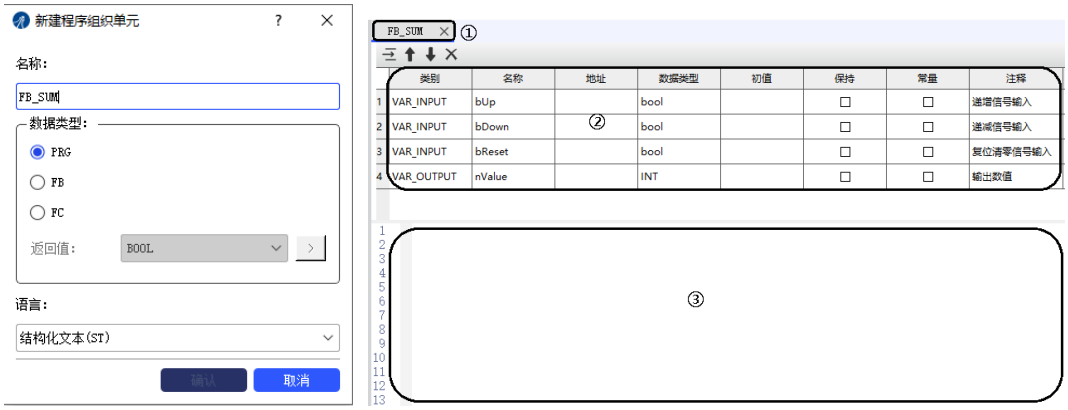
功能块（FB）是把程序中反复使用的部分程序抽象封装成一个通用程序块，它可以在程序中被任何一种编程语言重复调用。在编程中使用封装的功能块，不仅可提高程序的开发效率，也减少了编程中的错误，从而改善了程序质量。

功能块在执行时能够产生一个或多个值，功能块拥有自己的内部变量，这些内部变量在控制器内执行时需要分配内存，从而构成自身的状态特征。因此，对于相同参数的输入变量值，可能存在在不同的内部状态变量，所以会得到不同的计算结果。

功能块是抽象的数据类型，因此功能块是需要实例化后才能被程序调用和执行。

- 新建功能块

在工程管理树中，右键单击“程序组织单元”，选择“新建 POU”，在弹出对话框中，选择功能块，点击“确定”即完成新建功能块。自定义功能块的界面如下：



图：新建功能块

①功能块名称；②变量声明区；③程序编辑区。

- 功能块编程

功能块程序编写语言可以使用结构化文本 ST 或者梯形逻辑图 LD，在功能块程序里面，可以调用函数或功能块。

功能块编程应注意以下事项：

1）为确保功能块不依赖于硬件，功能块的变量声明中不允许使用固定地址变量（如%IX0.1、%QW2）作为局部变量，但在调用时可以给其赋值。

2）使用 VAR_INPUT 和 VAR_OUTPUT 会占用过多的内存，为此，在功能块编程时，

尽可能使用 VAR_IN_OUT 替代，减少对存储区的占用。

3）功能块需要实例化，才能被调用，同一个程序多次调用同一个功能块时，建议定义不同的实例。

我们以自定义一个递增/递减功能块为例说明，该功能块分为 3 个输入：增计数、减计数和复位，当前计数值为输出。功能块声明及代码如下：

FB_SUM

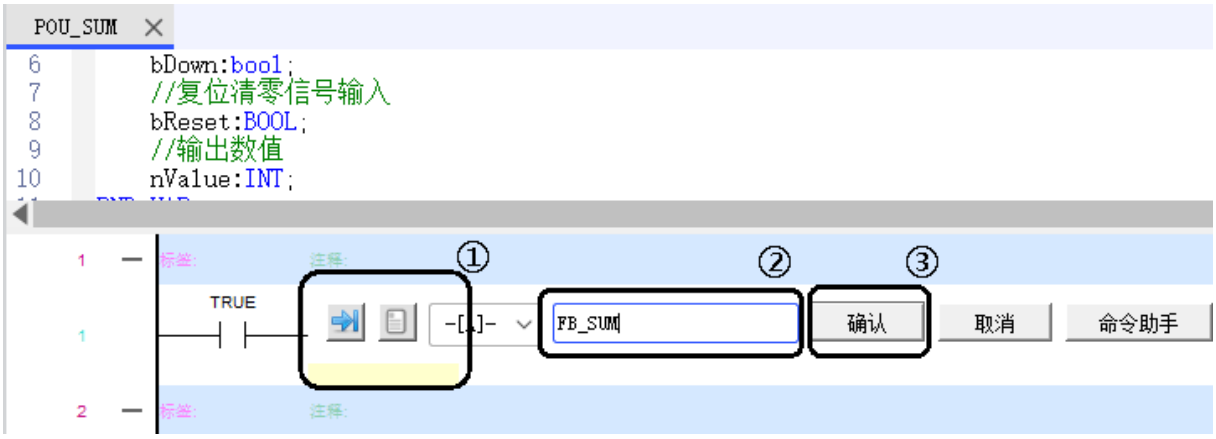
	类别	名称	地址	数据类型	初值	保持	常量	注释
1	VAR_INPUT	bUp		bool		☐	☐	递增信号输入
2	VAR_INPUT	bDown		bool		☐	☐	递减信号输入
3	VAR_INPUT	bReset		bool		☐	☐	复位清零信号输入
4	VAR_OUTPUT	nValue		INT		☐	☐	输出数值

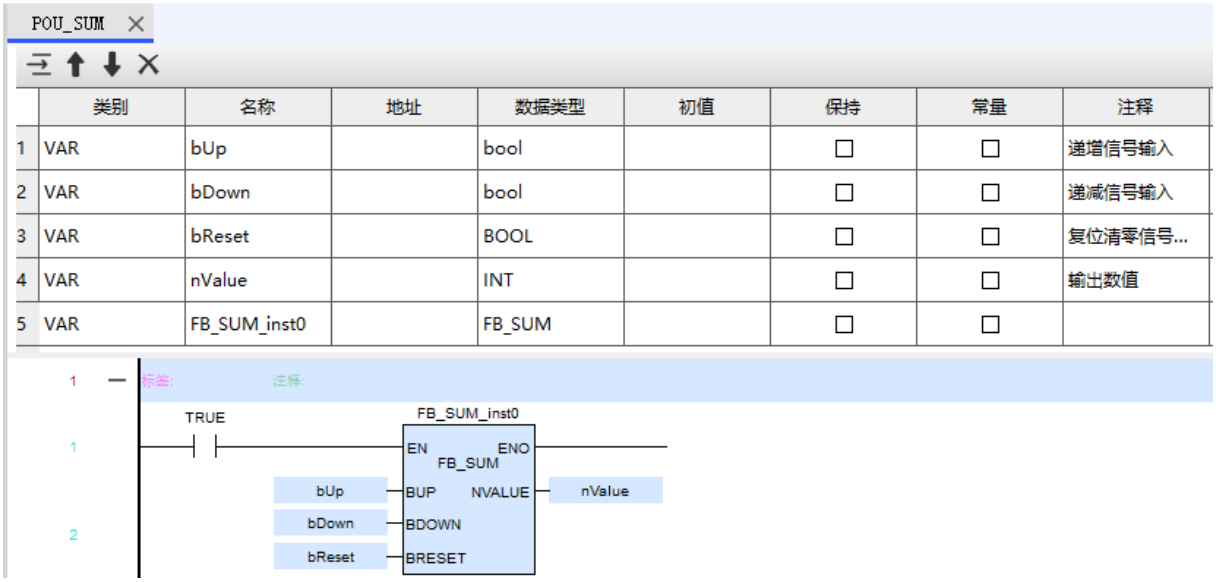
```
IF bUp THEN
    nValue := nValue+1;    //数据递增
END_IF

IF bDown THEN
    nValue := nValue-1;    //数据递减
END_IF

IF bReset THEN
    nValue := 0;           //数据清零
END_IF
```

图：功能块示例

- 功能块调用

功能块需要实例化后才能在程序中调用，为了方便查看功能块接口，我们新建 POU 程序的编程语言选择为 LD，在该 POU 中调用 FB_SUM 功能块。在程序编辑区通过双击梯形图“空白区”，输入功能块名称，点击确定，完成功能块实例化声明，填写完输入输出接口即可。

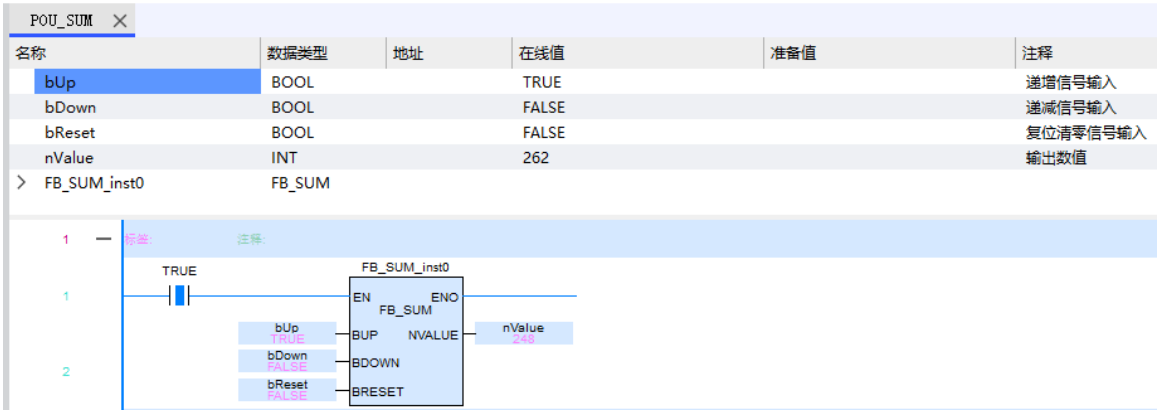




图：功能块实例调用

①双击空白区；②输入功能块名称；③单击确定。

其运行结果如下图。当输入信号 bUp 为 TRUE 时，输出值不断累加；当 Down 为 TRUE 时，输出值不断递减；当 bReset 为 TRUE 时，输出值清零。



图：功能块实例执行结果

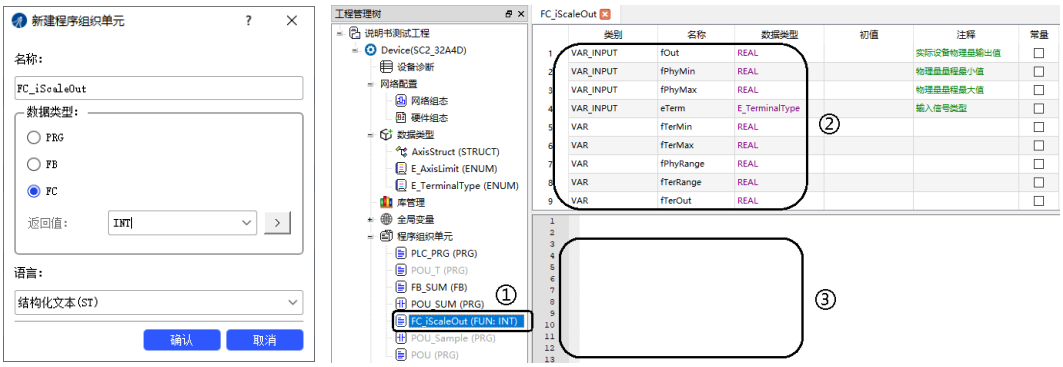
3.4.2 函数

函数（FC）是至少有一个输入变量、无私有数据、仅有一个返回值的基本算法单元，可以被函数、功能块、程序调用。函数的一个重要特性是它不能使用内部变量存储数值，也就是说，只要给定相同的输入参数，必定得到相同的输出，这是函数（FC）与功能块（FB）之间的主要区别。

- 新建函数

在工程管理树中，右键单击“程序组织单元”，选择“新建 POU”，在弹出对话框中，

选择函数，点击“确定”即完成新建函数。自定义函数的界面如下：



图：新建函数

①函数名称及其返回值类型；②变量声明；③程序编辑区。

- 函数编程

函数的编写语言可以使用结构化文本 ST 或者梯形逻辑图 LD，函数名即为函数的返回值，也可以理解为函数的输出值。

函数编程应注意以下事项：

1）函数可以拥有多个输入变量，但只能有一个返回值，由于没有限制返回值的数据类型，用户甚至可以将一个结构体作为返回值。

2）函数的重要特性是它不能在内部变量存储数值，这点与功能块截然不同。

3）函数没有指定的内存分配，不需要像功能块一样进行实例化。

4）函数只能调用函数，不能调用功能块。

5）函数的输入端口（VAR_INPUT）可以是空的、常数、变量或者函数调用。

我们以自定义模拟量输入信号转换为数字量信号为例说明，使用 PLC 时常会遇到一些实际问题，如很多情况下需要将实际的模拟量信号转换为数字量信号。常用的模拟量电流信号有 0~20mA，4~20mA，电压信号有 0~10V，-10~10V，通过这些输入参数的类型选定，将其转换为数字量值。FC_iScaleOut 函数声明及代码如下：

FC_iScaleOut

	类别	名称	地址	数据类型	初值	常量	注释
1	VAR_INPUT	fOut		REAL		☐	实际设备物理量输出值
2	VAR_INPUT	fPhyMin		REAL		☐	物理量量程最小值
3	VAR_INPUT	fPhyMax		REAL		☐	物理量量程最大值
4	VAR_INPUT	eTerm		E_TerminalType		☐	输入信号类型
5	VAR	fTerMin		REAL		☐	
6	VAR	fTerMax		REAL		☐	
7	VAR	fPhyRange		REAL		☐	
8	VAR	fTerRange		REAL		☐	
9	VAR	fTerOut		REAL		☐	

```
1   fTerMax := 32768.0;
2  -CASE eTerm OF
3       E_TerminalType.eTerm_0mA_20mA:
4       fTerMin := 0.0;
5       E_TerminalType.eTerm_4mA_20mA:
6       fTerMin := 0.0;
7       E_TerminalType.eTerm_0V_10V:
8       fTerMin := 0.0;
9       E_TerminalType.eTerm_neg0V_10V:
10      fTerMin := 32768.0;
11      ELSE
12      fTerMin := -32768.0;
13  END_CASE
14  fPhyRange := fPhyMax - fPhyMin;
15  fTerRange := fTerMax - fTerMin;
16 -IF fPhyRange > 0.0 AND fTerRange >0.0 THEN
17      fTerOut := fTerMin + (fTerRange*(fOut-fPhyMin)/fPhyRange);
18  ELSE
19      fTerOut := 0.0;
20  END_IF
21
22  FC_iScaleOut := REAL_TO_INT(fTerOut);
```

图：函数示例

示例代码中使用的枚举类型 E_TerminalType 定义如下：

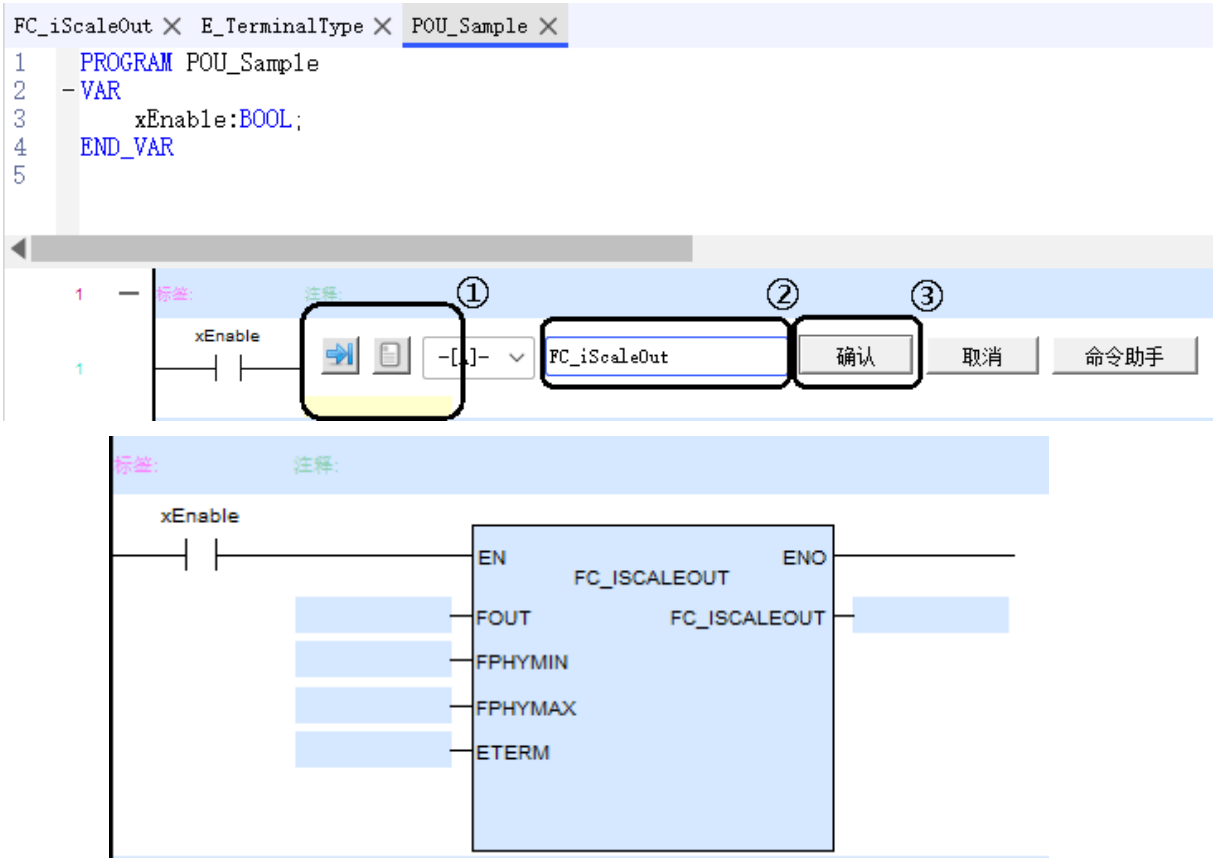
FC_iScaleOut　E_TerminalType

	名称	初值	注释
1	eTerm_0mA_2...	0	
2	eTerm_4mA_2...	1	
3	eTerm_0V_10V	2	
4	eTerm_neg0V...	3	

图：枚举类型 E_TerminalType 定义

- 函数调用

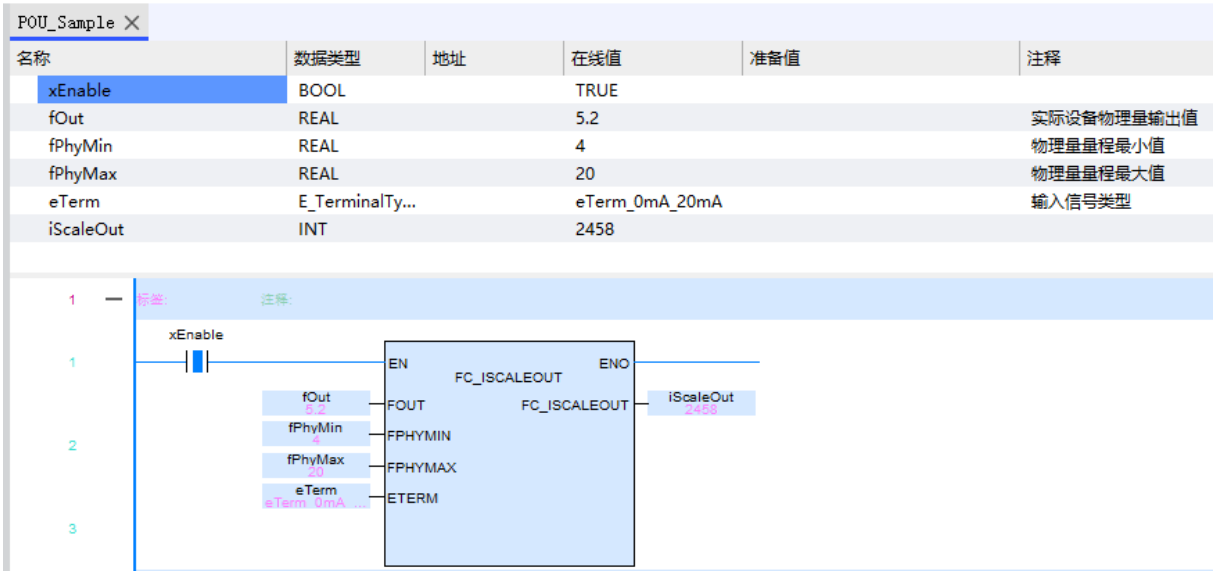
函数的调用无需实例化，为了方便查看函数接口，我们新建 POU_Sample 程序的编程语言选择为 LD，在 POU_Sample 中调用函数 FC_iScaleOut。在程序编辑区通过双击梯形图“空白区”，输入函数名称，点击确定，填写完输入输出接口即可。



图：函数调用

①双击空白区；②输入函数名称；③单击确定。

程序调用 FC_iScaleOut 函数结果如下图所示：



图：函数执行结果

通过函数和功能块的定义和使用，我们发现两者调用的表达式很相似，但两者还是具有明显的区别，主要区别见下表：

	函数（FUN）	功能块（FB）
内存分配	没有指定的内存分配地址	全部数据分配内存地址
输入/输出变量	只有一个返回值	多个输出变量或没有输出变量
调用关系	可以调用函数，但不能调用功能块	可以调用功能块或函数

3.5 定时器

3.5.1 定时器指令

LeadStudio 支持四种定时器类型，定时器 TP，通电延时定时器 TON，断电延时计时器 TOF，保持型通电延时定时器 TONR。如下表所示：

表：LeadStudio 支持的定时器类型

种类	指令	指令名称	功能说明
定时器	TP	定时器	仅在设定时间内输出 TRUE 的定时器
	TON	通电延时定时器	通电后经过设定时间输出 TRUE 的定时器
	TOF	断电延时计时器	断电后经过设定时间输出 FALSE 的定时器
	TONR	保持型通电延时定时器	通电后经过设定时间输出 TRUE 的定时器，断开后保持当前定时时间，再次接通的时候会继续从当前时间开始计时，用 RESET 复位输出

3.5.2 定时器 TP

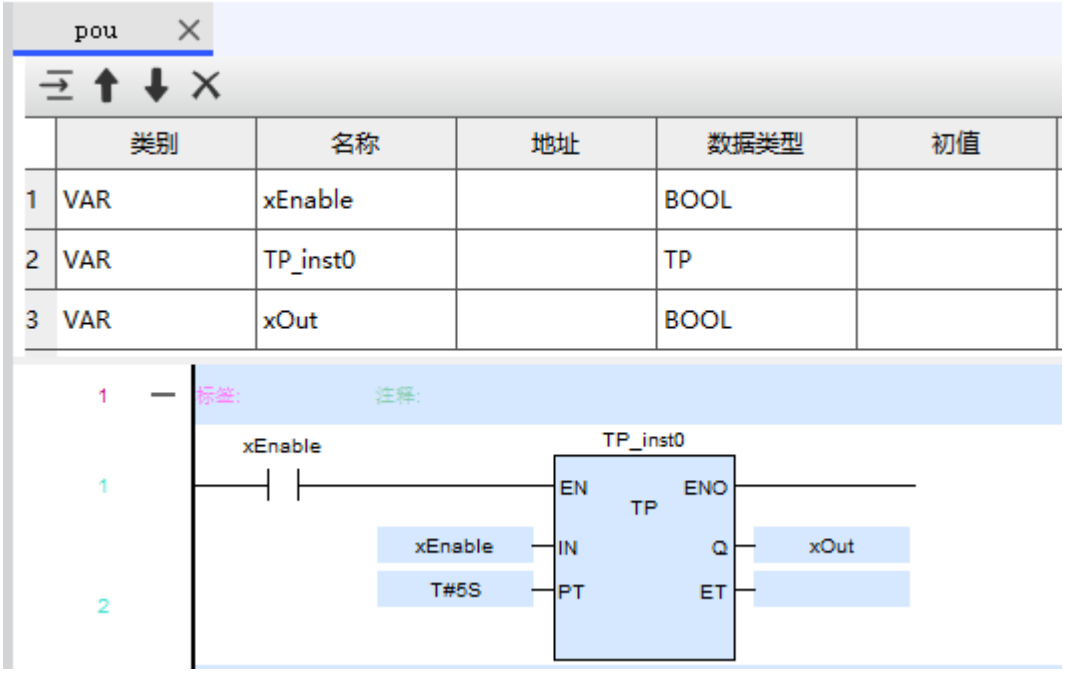
启动后，仅在设定时间内输出 TRUE 的定时器。

当检测到 IN 上升沿后，不论 IN 是否保持为 TRUE，输出 ET 以毫秒精度开始计时，定时器输出 Q 变为 TRUE，经过时间 ET 随着时间不断增加。

ET 到达设定时间 PT 后，定时器输出 Q 变为 FALSE。此时，ET 停止增加。

如果 IN 没有检测到上升沿，Q 输出为 FALSE。

定时器 TP 实例化调用示例如下：



图：定时器 TP 示例

3.5.3 通电延时定时器 TON

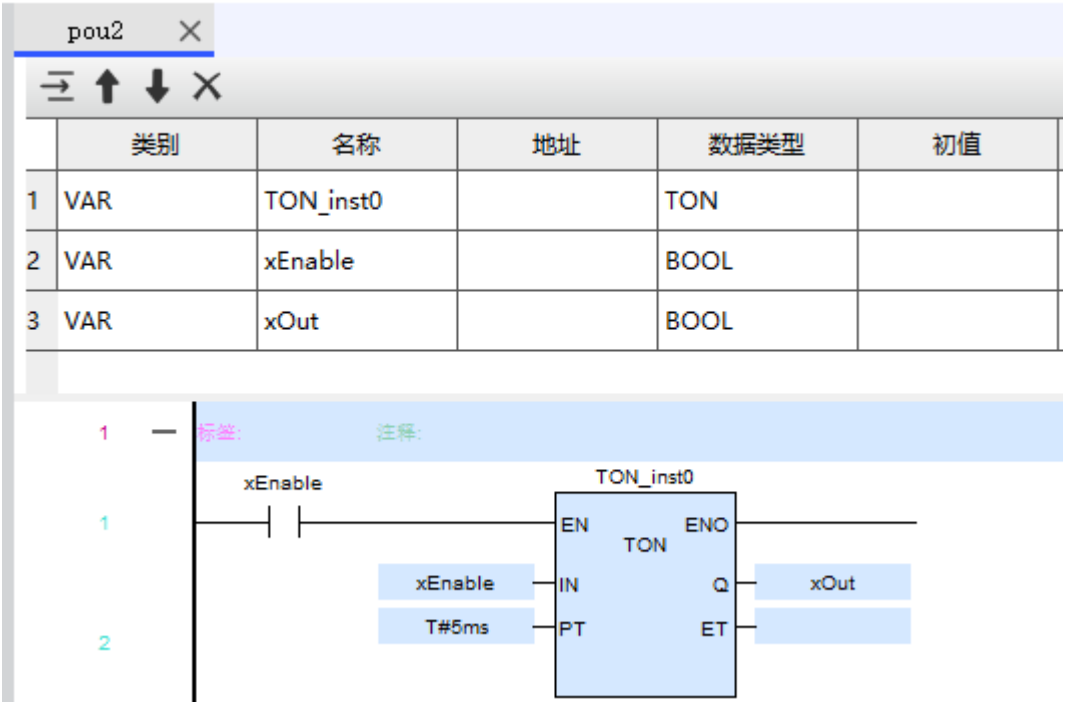
从启动起经过设定时间后，输出 TRUE 的定时器。

当 IN 为 FALSE 时,Q 为 FALSE，ET 为 0。

当检测到 IN 上升沿后，输出端 ET 以精确到毫秒级别开始计时，如果 IN 持续保持为 TRUE，继续正常计时，到达设定时间 PT 后，定时器状态输出 Q 为 TRUE。

如果计时到达设定时间 PT 之前，输入 IN 由 TRUE 变为 FALSE，则定时器状态输出 Q 为 FALSE，此时 ET 变为 0。

通电延时定时器 TON 实例化调用示例如下：



图：通电延时定时器 TON 示例

3.5.4 断电延时计时器 TOF

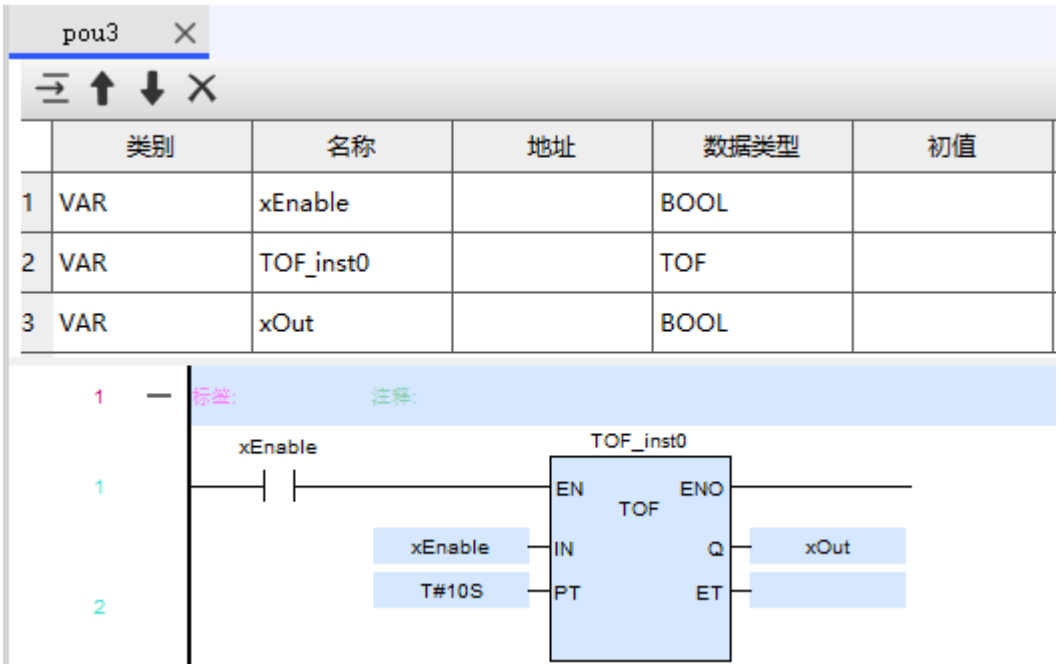
从启动起经过设定时间后，输出 FALSE 的定时器。

当 IN 为 TRUE 时,Q 为 TRUE，ET 为 0。

当检测到 IN 下降沿后，输出端 ET 以精确到毫秒级别开始计时，如果 IN 持续保持为 FALSE，继续正常计时，到达设定时间 PT 后，定时器状态输出 Q 为 FALSE。

如果计时到达设定时间 PT 之前，输入 IN 由 FALSE 变为 TRUE，则定时器状态输出 Q 还是为 TRUE，此时 ET 变为 0。

断电延时计时器 TOF 实例化调用示例如下：



图：断电延时计时器 TOF 示例

3.5.5 保持型通电延时定时器 TONR

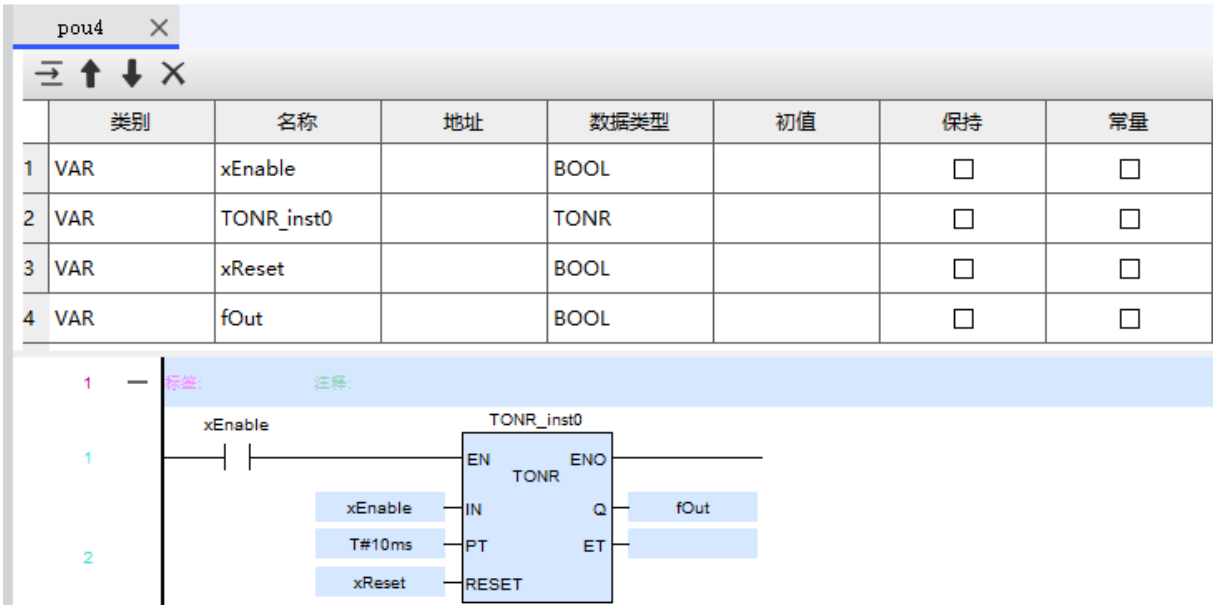
从启动起经过设定时间后，输出 TRUE 的定时器。

当 RESET 为 TRUE 时,Q 为 FALSE，ET 为 0。

当检测到 IN 上升沿后，输出端 ET 以精确到毫秒级别开始计时，如果 IN 持续保持为 TRUE，继续正常计时，到达设定时间 PT 后，定时器状态输出 Q 为 TRUE。

如果计时到达设定时间 PT 之前，输入 IN 由 TRUE 变为 FALSE，则定时器保持当前定时时间 ET，再次接通的时候会继续从当前时间开始计时。

保持型通电延时定时器 TONR 实例化调用示例如下：



图：保持型通电延时定时器 TONR 示例

4 编程语言

4.1 LeadStudio 支持的编程语言类型

LeadStudio 支持以下 2 种编程语言，编程者可以根据实际的需要任意选择编程语言。

- ✓ Ladder diagram(LD) 梯形图
- ✓ Structured text (ST) 结构化文本

编程语言的多样性是 LeadStudio 的一大优点。对于 LeadStudio 编程而言，建议编程者在选择编程语言时应根据实际的编程方便来选择编程语言，而不是在整个程序中仅使用 LD。

那么，编程者在选择编程语言时具体怎么选择呢？从优化程序和编程便利性的角度建议大家，涉及到算法部分，流程控制部分请选择 ST 语言，编写的程序往往简洁而高效，条理清晰，逻辑关系不会混乱；涉及到逻辑控制部分，功能块部分，请选择 LD 语言，编写的联锁，互锁等逻辑简单易懂。当然，在实际的编程时，用户也可以根据自己的使用习惯来选择编程语言，虽然实现的方法不同，但是都能得到同一个结果。

4.2 结构化文本（ST）

结构化文本 ST 是用结构化的描述文本来编写程序的一种编程语言，与 PASCAL 或 C 语言相似。ST 语言的特点是“高级文本编程”和“结构化”，适合于算法和结构较为复杂，其它编程语言 （如梯形图）实现比较困难的情况。具有高效、快捷、简洁的优点。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。

ST 编程语言各种元素具体如下：

- 表达式：由操作数和操作符组成的结构，在执行表达式时会返回值。
- 操作数：操作数表示变量，数值，地址，功能块等。
- 操作符：操作符是执行运算过程中所用的符号。
- 语句：语句用于将表达式返回的值赋给实际参数，并构造和控制表达式。

ST 程序执行顺序可以通过编排在 ST 编辑器中的语句顺序得到，从左到右，从上到下。这个顺序只能通过插入循环语句而改变。

ST 编辑器窗口如下：

FC_iScaleOut

	类别	名称	地址	数据类型	初值	常量	注释	特性
1	VAR_INPUT	fOut		REAL		☐	实际设备物理...	
2	VAR_INPUT	fPhyMin		REAL		☐	物理量量程最...	
3	VAR_INPUT	fPhyMax		REAL		☐	物理量量程最...	
4	VAR_INPUT	eTerm		E_TerminalType		☐	输入信号类型	
5	VAR	fTerMin		REAL		☐		
6	VAR	fTerMax		REAL		☐		
7	VAR	fPhyRange		REAL		☐		
8	VAR	fTerRange		REAL		☐		
9	VAR	fTerOut		REAL		☐		

```
fTerMax := 32768.0;
CASE eTerm OF
    E_TerminalType.eTerm_0mA_20mA:
    fTerMin := 0.0;
    E_TerminalType.eTerm_4mA_20mA:
    fTerMin := 0.0;
    E_TerminalType.eTerm_0V_10V:
    fTerMin := 0.0;
    E_TerminalType.eTerm_neg0V_10V:
    fTerMin := 32768.0;
    ELSE
    fTerMin := -32768.0;
END_CASE
fPhyRange := fPhyMax - fPhyMin;
fTerRange := fTerMax - fTerMin;
IF fPhyRange > 0.0 AND fTerRange >0.0 THEN
    fTerOut := fTerMin + (fTerRange*(fOut-fPhyMin)/fPhyRange);
ELSE
    fTerOut := 0.0;
END_IF

FC_iScaleOut := REAL_TO_INT(fTerOut);
```

图：ST 编辑器窗口

4.2.1 表达式

表达式是返回变量评估值的结构。在语句中需要使用该返回值。表达式由操作符和操作数组成。操作数可以是常量、变量、函数调用的返回值或其他表达式。举例：

123, t#10ms，‘abc’ （*常量*）

ivar1, global1 （*变量*）

Func(in1,in2) （*函数调用*）

a + b, a AND b, x*y （*操作符调用*）

计算表达式时将根据操作符的优先级所定义的顺序将操作符应用于操作数表。首先执行表达式中具有最高优先级的操作符，接着执行具有次优先级的操作符；以此类推，直到完成整个计算过程。优先级相同的操作符将根据它们在表达式中的书写顺序从左至

右执行。可使用括号更改此顺序。

例如，如果 A、B、C 和 D 的值分别为 1、2、3 和 4，并按以下方式计算：A+B-C*D，结果为-9。（A+B-C）*D，结果则为 0。如果操作符包含两个操作数，则先执行左边的操作数，例如在表达式 SIN(x)*COS(y)中，先计算表达式 SIN(x)，后计算 COS(y)，然后计算二者的乘积。

4.2.2 操作符

操作符是一种符号，它表示要执行的算术运算、要执行的逻辑运算、功能编辑调用。操作符是泛型的，即它们自动适应操作数的数据类型。

常见 ST 操作符及其执行的操作如下表所示：

表：常见 ST 操作符

操作符	执行的操作	优先级
（表达式）	括号	高 ↑ 低
函数名（参数列表，由逗号分隔）	调用函数	
EXPT	指数运算	
-，NOT	取反	
*	乘	
/	除	
MOD	取余数	
+	加	
-	减	
<，>，<=，>=	比较运算	
=	等于	
<>	不等于	
AND	与	

XOR	异或	
OR	或	

4.2.3 操作数

操作数可以是：地址、数值、变量、数组变量、结构体变量、数组/结构体变量的元素、功能调用、功能块输出等。处理操作数的语句中的数据类型必须相同。如果需要处理不同类型的操作数，则必须预先执行类型转换，否则可能存在数据丢失的情况。

在下面示例中，整数变量 Var1 在添加到实数变量 R1 中之前会先转换为实数变量。

R2:=R1+SIN (INT_TO_REAL (Var1));

4.2.4 语句

ST 编程语言常用的语句主要分为以下 4 类：

- 赋值语句：:=
- 条件语句：IF、CASE
- 循环语句：WHILE、CONTINUE、FOR、REPEAT
- 跳转语句：JMP
- 跳出语句：EXIT、RETURN

下文逐一说明各语句的使用方式。

1) 赋值语句

赋值语句用表达式的求值结果替代单元素或多元素变量的当前值。

语法：

A := B; （A 是变量名称；B 是变量、求值的表达式或 FB/FC）

将操作符 ”:=” 右边变量、表达式或 FB/FUN 的值赋给左边的变量，

两个变量（分别位于赋值操作符的左侧和右侧）的数据类型必须相同。数组是个特例。显式启用后，也可对长度不同的两个数组执行赋值操作。

示例：

1.将数值赋给变量。

iVar := 30;

用于将值 30 赋给变量 iVar。

2.将运算值赋给变量

X := (A+B-C)*D;

用于将 (A+B-C)*D 的运算结果赋给变量 X。

3.将 FUN/FB 的值赋给变量，赋值用于将 FC 返回值或 FB 的输出值赋给变量。

A := MY_TOF.Q;

用于将 MY_TOF 功能块（TOF 功能块的实例）的 Q 输出值赋给变量 A。（这不是功能块调用）。

B := C MOD A;

用于调用 MOD（模数）函数并将计算结果赋给变量 B。

2) IF 语句

使用 IF 指令可以检查条件，并根据此条件执行相应的语句。当相关的布尔表达式求值为 TRUE 时执行一组语句，条件为 FALSE 时不执行，结束语句，或者执行 ELSE 或者 ELSIF 后面的语句

语法：

常用的 IF 语句结构有以下 3 种：

a）IF 条件 A THEN //当条件 A 满足时，执行语句 A。

表达式 A ;

END_IF

b）IF 条件 A THEN //当条件 A 满足时，执行语句 A;否则，执行语句 B。

表达式 A;

ELSE

表达式 B;

END_IF

c）IF 条件 A THEN //当条件 A 满足时，执行语句 A。

表达式 A ;

ELSIF 条件 B THEN //当条件 B 满足时，执行语句 B。

表达式 B ;

…

ELSIF 条件 N-1 THEN　　//当条件 N-1 满足时，执行语句 N-1。

表达式 N-1;

ELSE　　//如果以上条件都不满足，则执行语句 N。

表达式 N ;

END_IF　　//指令结束。

示例：

```
IF fPhyRange > 0.0 AND fTerRange >0.0 THEN
    fTerOut := fTerMin + (fTerRange*(fOut-fPhyMin)/fPhyRange);
ELSE
    fTerOut := 0.0;
END_IF
```

图：IF 语句示例

3）CASE 语句

CASE 语句包含一个整型数据或枚举数据控制变量和多组表达式列表。每组都标记可应用的一个或多个整数值或枚举值的范围。如果控制变量与其中一个选择值相等，则执行该组表达式。否则，执行 ELSE 表达式。

OF 语句指示范围的开头。所有范围都不包含选择值时，才会在 CASE 语句内执行 ELSE 语句。END_CASE 关键字标记语句的结尾。

语法：

CASE 控制变量 OF

选择值 1:

表达式 1;

选择值 2:

表达式 2;

选择值 3,选择值 4:　　//控制变量等于选择值 3 或 4 时执行表达式 3

表达式 3;

选择值 5..选择值 9:　　//控制变量等于选择值 4~ 9 之内的值时执行表达式 4

表达式 4;

…

ELSE　　//可选项

ELSE 表达式;

END_CASE

当使用 IF 指令有过多分层，或者需要使用多个 ELSIF，才能完成程序功能时，使用 CASE 指令 替代 IF 指令，可以简化程序，并且能提高程序的可读性。

示例：

```
CASE eTerm OF
    E_TerminalType.eTerm_0mA_20mA:
    fTerMin := 0.0;
    E_TerminalType.eTerm_4mA_20mA:
    fTerMin := 0.0;
    E_TerminalType.eTerm_0V_10V:
    fTerMin := 0.0;
    E_TerminalType.eTerm_neg0V_10V:
    fTerMin := 32768.0;
    ELSE
    fTerMin := -32768.0;
END_CASE
```

图：CASE 语句示例

4）FOR 循环

FOR 语句用于在发生次数可预先确定的情况下。否则可使用 WHILE 或 REPEAT。FOR 语句会重复执行语句序列，直到遇到 END_FOR 语句为止。发生次数由起始值、结束值和控制变量决定。

语法：

FOR 控制变量 := 循环起始值 TO 循环结束值 { BY 递增步长} DO

表达式;

END_FOR

其中，{}内语句可根据需要省略，省略时步长默认为 1。若要中断循环指令，请执行 EXIT 指令。

示例：

```
FOR i:= 1 TO 9  BY  1 DO
    iTest := iTest +1;
END_FOR;
```

图：FOR 语句示例

此程序的循环控制变量为 i，循环开始时控制变量初值为 1，每一次循环 i+1；当 i 大于 9 时，执行完 FOR 循环内容后，退出循环，执行下一条语句。语句 iTest := iTest + 1 ; 一共执行 9 次；假设 iTest 的初始值是 1，那么循环结束后，iTest 的值为 10。

5）WHILE 循环

WHILE 循环与 FOR 循环使用方法类似。二者的不同之处是，WHILE 循环的结束条件不是指定的循环次数，而是任意的逻辑表达式。当该表达式叙述的条件满足时，执行循环。WHILE 语句可使一个表达式序列重复执行，直到其循环条件为假。如果从一开始该循环条件就为假，则根本不会执行该表达式组。DO 语句标识重复定义的结尾和语句的开头。可以使用 EXIT 提前终止循环。 END_WHILE 关键字标记语句的结尾。

语法：

WHILE 循环条件 DO

表达式;

END_WHILE

WHILE 循环执行前先检查<循环条件>是否为 TRUE，如果为 TRUE，则执行<表达式>；当执行完一次后，再次检查<循环条件>，如果仍为 TRUE，则再次执行，直到<循环条件>为 FALSE。如果一开始<循环条件>就为 FALSE，则不会执行 WHILE 循环里的指令。

示例：

```
WHILE iTest<>100 DO
    iTest := iTest+1;
END_WHILE;
```

图：WHILE 语句示例

6）REPEAT 循环

REPEAT 循环与 WHILE 循环一样，也是没有明确循环次数的循环。与 WHILE 循环的区别在于，REPEAT 循环在指令执行以后，才检查结束条件。因此无论结束条件怎样，循环至少执行一次。UNTIL 语句标记结束条件。可以使用 EXIT 提前终止循环。END_REPEAT 语句标记语句的结尾。

语法：

REPEAT

表达式;

UNTIL 循环结束条件

END_REPEAT

语句一直执行，直到循环结束条件为 TRUE 时，REPEAT 循环结束。如果一开始就为 TRUE，则循环只执行一次。

示例：

上述 WHILE 示例程序也可写为：

```
REPEAT
    iTest := iTest+1;
    UNTIL iTest > 100
END_REPEAT;
```

图：REPEAT 语句示例

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 循环更简单。

7）CONTINUE 语句

CONTINUE 语句用于在满足结束条件前终止循环语句的本次循环，进入下次循环（FOR、WHILE 或 REPEAT）。如果 CONTINUE 语句位于嵌套的重复语句内，则会继续最里层的循环。

语法：

CONTINUE;

示例：

```
FOR i:= 1 TO 9  BY  1 DO
    iTest := iTest +1;
    IF iTest1 = 10 THEN
        CONTINUE;
        END_IF
    iTest1 := iTest1+1;
END_FOR;
```

图：CONTINUE 语句示例

8）JMP 语句

跳转指令，无条件的跳转到 label 所在的位置执行程序。JMP 指令容易造成程序结构混乱，降低代码可读性，不建议使用。

语法：

label:

……

JMP label;

label 可以是任意确定的标识符，它被放置在程序的开始处。JMP 指令必须有一个跳转目标，也就是预定义的标签。当执行到 JMP 指令后，程会跳转到指定的标签处开始执行。

示例：

```
test2 := 0;
label : test2 := test2 +1;

IF test2<10 THEN
    JMP label;
    END_IF
```

图：JMP 语句示例

当 test2 <10 时，跳转回 label 所在行，执行 test2 := test2 + 1 ;。

这个例子中的功能同样可以通过使用 WHILE 或者 REPEAT 循环来实现。通常情况下，能够并且也应该避免使用跳转指令，因为这降低了代码的可读性。

9）EXIT 语句

EXIT 语句用于在满足结束条件前终止循环语句（FOR、WHILE 或 REPEAT）。如果 EXIT 语句位于嵌套的循环语句内，则会离开最里面的循环（EXIT 所在的循环）。接下来，将执行循环结尾（END_FOR、 END_WHILE 或 END_REPEAT）后的第一个语句。

语法：

EXIT;

示例：

```
FOR test2 := 1  TO 10 BY 1 DO
    test3 := test3 -1 ;
    IF test3 = 0 THEN
        EXIT;
        END_IF
        test4 := test4/test3;
END_FOR;
```

图：EXIT 语句示例

当 test3 等于 0 时，FOR 循环结束。

10）RETURN 语句

结束 POU、函数或功能块，将处理恢复到调用源。

语法：

RETURN;

执行本指令前，请设置该 POU 的返回值、输出变量的值。若经常使用本指令，处理流程将变的复杂。敬请注意。

示例：

```
IF test1 = true THEN
        RETURN;
END_IF
    test2 := test2 +1;
```

图：RETURN 语句示例

如果 test1 为 TRUE，将不会执行 test2 := test2 + 1 ，而是直接退出 POU。

4.2.5 调用功能块

库管理器中所有的函数和功能块都可以在 ST 语言中被调用。如果需要在 ST 中调用功能块，必须先对功能块进行实例声明。对功能块进行实例声明后，可通过“实例名. 参数名”，调用功能块中的参数；也可直接输入功能块的实例名称，并在随后的括号中给功能块的各参数分配数值或变量，参数之间以逗号隔开，功能块调用以分号结束。程序调用用户自行创建的功能块的方法与调用系统本身提供的标准功能块指令的方法相同。

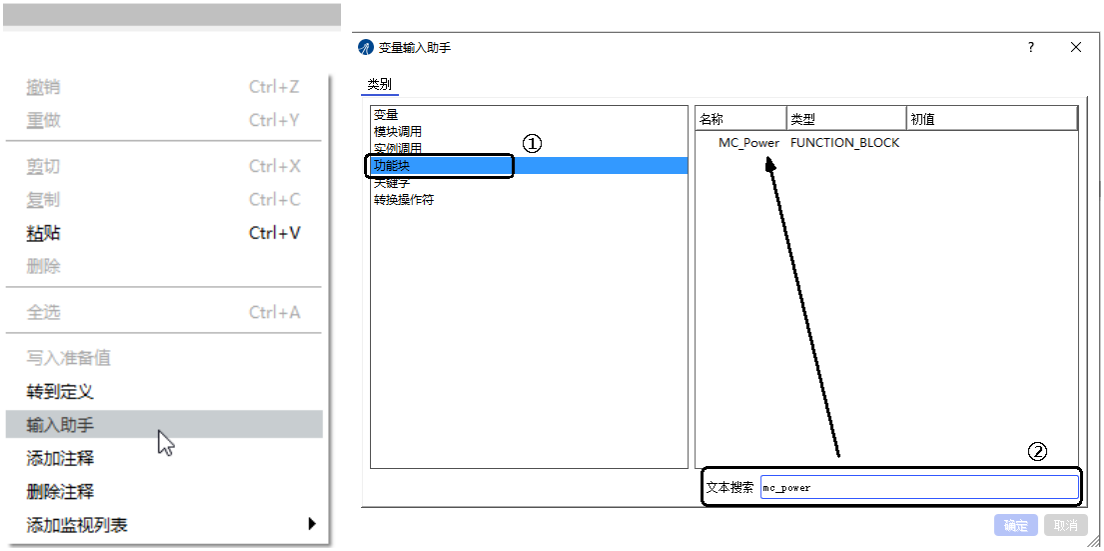
实例针对功能块而言，每个功能块实例就是一个独立的、可完成特定逻辑功能的活动对象。不同的程序、不同的任务都可以定义和调用功能块的应用实例，每个调用实例都占用独立的内存，保留独立的逻辑状态。

语法：

```
name of FB instance(                          //功能块实例名称
FB input variable := value or address,        //输入引脚
further FB input variable := value or address,
…,
FB output variables => value or address ,     //输出引脚
further FB output variables => value or address ,
… );
```

当用户不想一个个输入功能块参数引脚名称时，可以在 ST 编辑器中右击，选择①“输

入助手”→②“文本搜索”中输入功能块名称调出功能块，在弹出的自动声明框对功能块进行实例化声明，输入输出引脚将自动补全，用户只需为各参数引脚分配数值或变量。

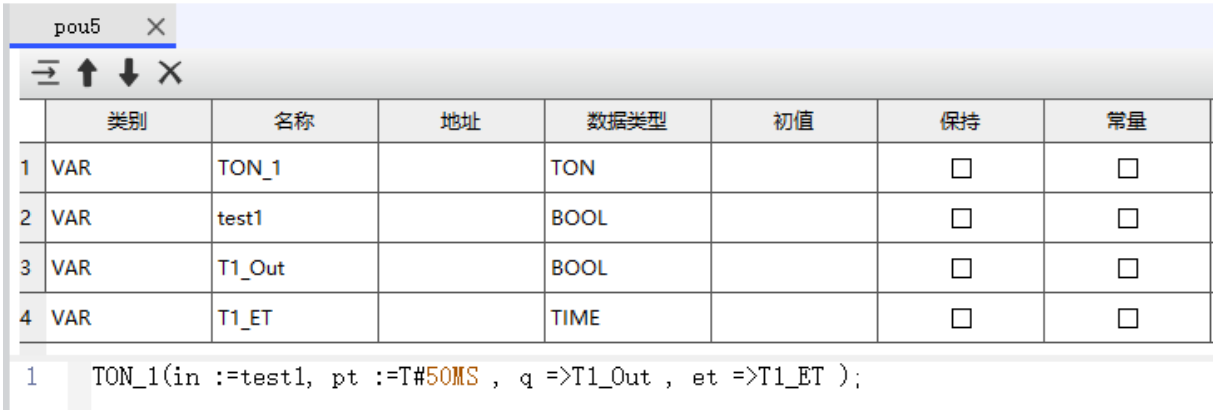


图：ST 调用功能块

用户也可在 ST 编辑区输入功能块名称后按“Tab”键，输入输出引脚也将自动补全。

示例：

在 ST 中调用 TON 定时器，假设其实例名为 TON_1:



图：ST 调用功能块 TON 示例

ST 程序调用函数的方法与调用功能块一样，只是函数没有实例名。故不再赘述。

4.2.6 注释

注释是 ST 程序中非常重要的一部分，它使程序更加具有可读性，同时不会影响程序的执行。 在 ST 编辑器的声明部分或执行部分的任何地方，都可以添加注释。 在 ST 语言中，有两种注释方法：

注释以 (* 开始，以 *) 结束。这种注释方法允许多行注释

注释以“//”开始，一直到本行结束。这是单行注释的方法。

例如：

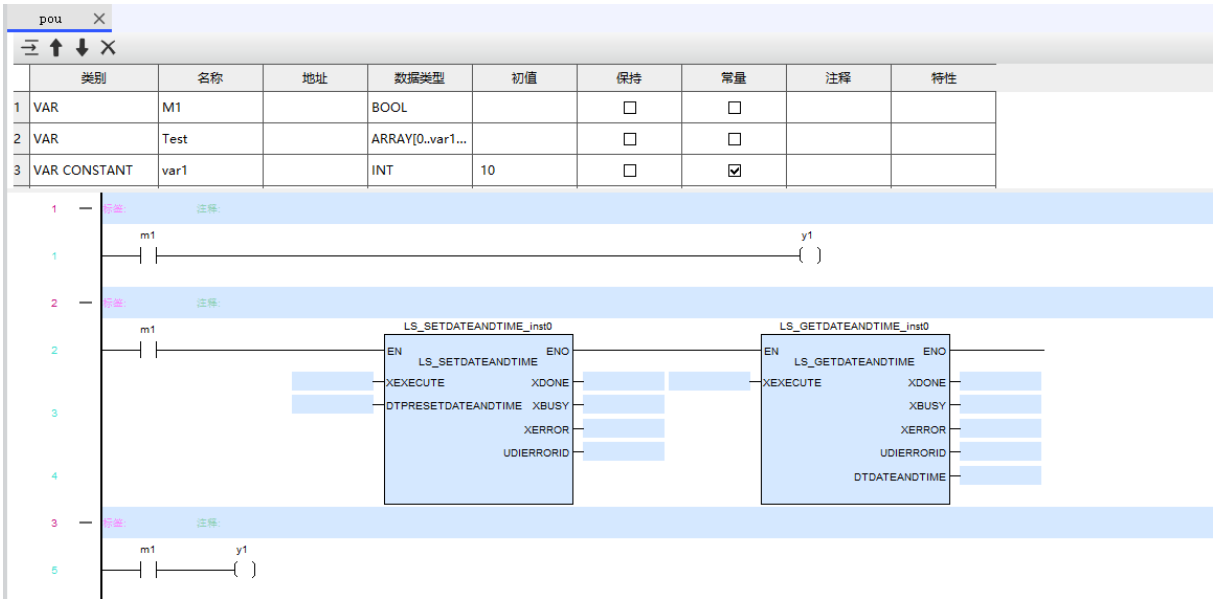
```
(*IF bUp THEN
    nValue := nValue+1;    //数据递增
END_IF*)

IF bDown THEN
//nValue := nValue-1;    //数据递减
END_IF
```

图：ST 注释示例

4.2 梯形图（LD）

梯形图语言是 PLC 程序设计中最常用的语言。由于与电气设计人员所熟悉的传统继电器电路图类似，因此梯形图语言得到了广泛的应用。梯形图包含了一系列的网络（也称节），左右两边各有一个垂直的电流线（一般称为母线）限制其范围，在中间是由触点、线圈、运算块（函数、功能块）、跳转、标签和连接线组成的电路图。每一个网络的左边有一系列触点，这些触点根据布尔变量值的 TRUE 和 FALSE 来传递从左到右的“ON”和“OFF”的状态。每一个触点是一个布尔变量，如变量值为 TRUE，通过连接线从左到右传递“ON”状态。否则传递"OFF"的状态。在网络最右边的线圈，根据左边的状态获得“ON”或“OFF”的状态，并相应地赋给一个布尔变量 TRUE 或 FALSE。

梯形图编辑界面如下图：



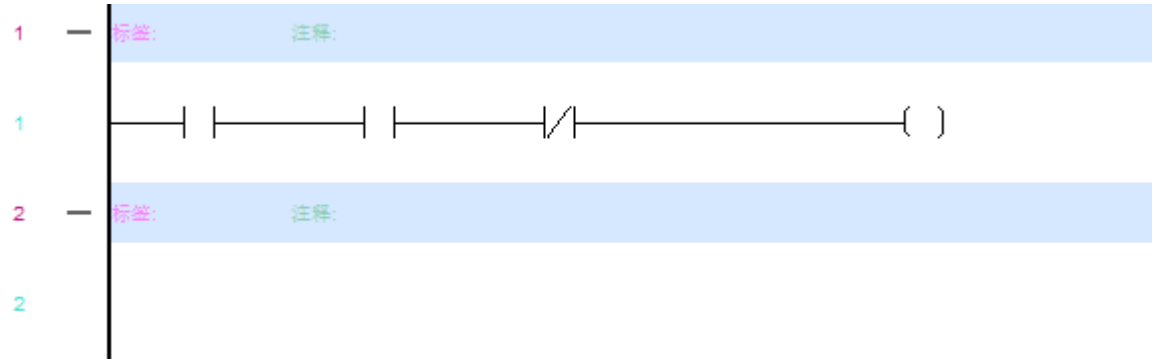
图：LD 梯形图编辑界面

4.2.1 梯形图元素

常用梯形图元素包括网络、触点、线圈、取反、上升沿/下降沿、横向连接线增加/删除、纵向连接线增加/删除、运算块、跳转、返回。

1） 网络

梯形图由一系列网络组成，网络由行与列划分出一个个小格，网络在左侧以垂直母线为界，其它所有的梯形图元素都位于网络右侧的小格中。网络中可插入标签和网络注释。



图：梯形图网络

2） 触点

触点分为常开和常闭触点。当选中一个已有触点后，再调用触点命令时，会覆盖已有触点。

常开触点符号：

常闭触点符号：

触点从左到右传递"ON" (TRUE) 或 "OFF" (FALSE)状态。一个布尔型的变量被分配到每个触点。如果此变量为真，常开触点将向右传递真，否则右部为假。常闭触点传递值相反。多个触点可串联也可并联，两个并联触点中只需一个触点为真，则平行线传输真，多个串联的触点要都为真，最后一个触点才能传输真，所以触点的串并联排列与串、并联电路相符。

3） 线圈

线圈分为线圈、置位线圈、复位线圈。

线圈符号：

置位线圈符号： 

复位线圈符号： 

线圈位于网络的右端，它们只能平行的排列，即向上或者向下插入并联线圈。从左到右的逻辑运算结果，赋值给线圈变量。线圈变量只能是布尔类型，它的值为"ON" (TRUE) 或 "OFF" (FALSE)。

4） 取反

取反符号： 

取反，对其左侧的触点运算结果取反，然后向右侧传递。如果取反元素左侧所有触点运算结果为 TRUE，则取反后为 FALSE，向右传递 FALSE；如果取反元素左侧所有触点运算结果为 FALSE，则取反后为 TRUE，向右传递 TRUE；

5） 上升沿/下降沿

上升沿符号：

下降沿符号：

上升沿，当其左侧触点运算结果从 FALSE 变为 TRUE 时，上升沿元素向右侧传递一个周期的 TRUE 信号，其他时候都为 FALSE；

下降沿，当其左侧触点运算结果从 TRUE 变为 FALSE 时，上升沿元素向右侧传递一个周期的 TRUE 信号，其他时候都为 FALSE；

6） 横向连接线增加/删除

横向连接线增加符号：

横向连接线删除符号： 

横向连接线增加，可以在选中的网络小格中添加横向连接线；

横向连接线删除，可以删除选中的网络小格中的横向连接线；

7） 纵向连接线增加/删除

纵向连接线增加符号：

纵向连接线删除符号： 

纵向连接线增加，可以在选中的网络小格中添加纵向连接线；

纵向连接线删除，可以删除选中的网络小格中的纵向连接线；

8） 运算块

运算块是一个复杂的元素，它可以是函数，如定时器、计数器，算术运算，也可以是功能块。如果为功能块，则运算块框上面会增加编辑框来显示功能块实例。运算块可以有输入和输出，并可以由用户系统库提供或用户自行编写。不管怎样至少要有一个布尔输入和输出。

运算块除了包含其本身所带的输入和输出外，还有 EN 输入和 ENO 输出。

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。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。

9） 标签

符号： 

标签标示程序的执行位置信息，是跳转指令跳转的目的地。标签是字符串，标签不存在任何输入和输出引脚，需符合标识符命名规则，标签与跳转元素搭配使用。

10） 跳转

符号：

该元素可以实现程序的转移执行。当执行该指令时跳转到该指令引用的标签处。跳转指令存在唯一的 BOOL 值输入引脚。

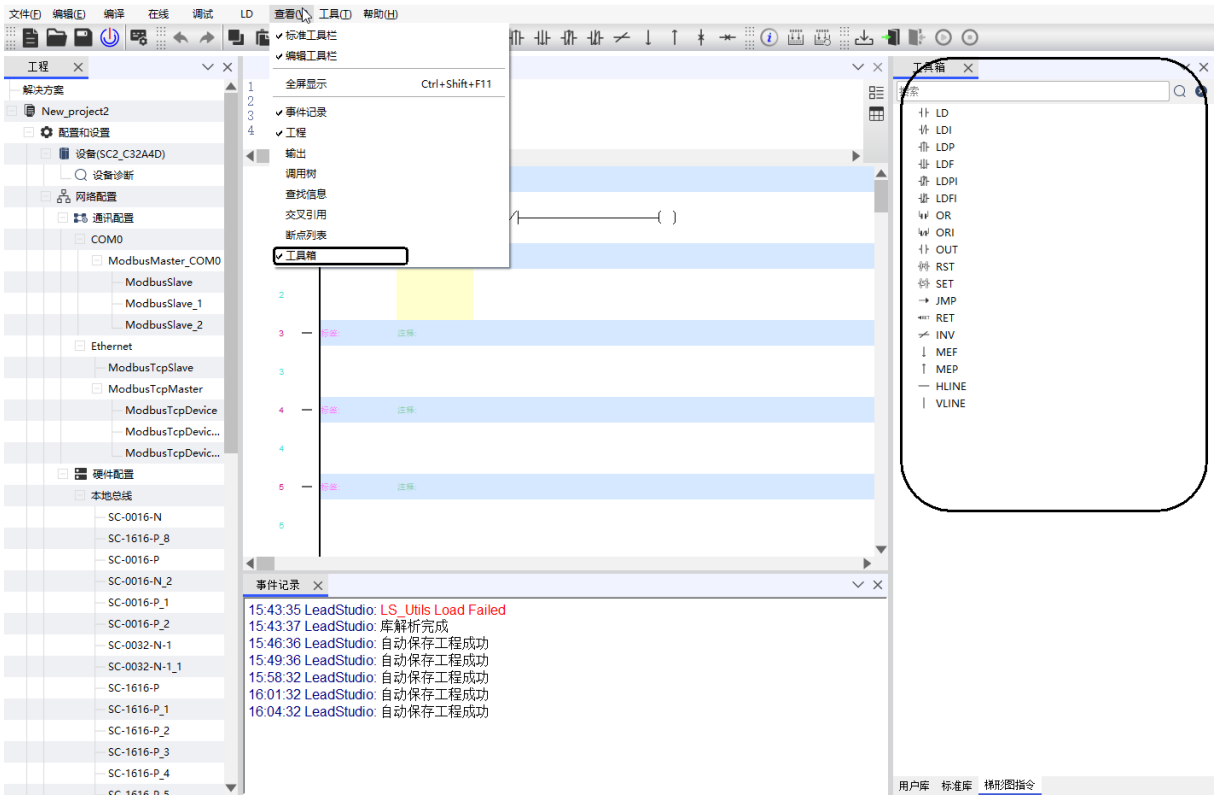
11） 返回

符号：

执行该操作后，当前 POU 直接返回到调用该 POU 程序指令处。返回指令存在唯一的输入引脚，输入值为 BOOL 类型。

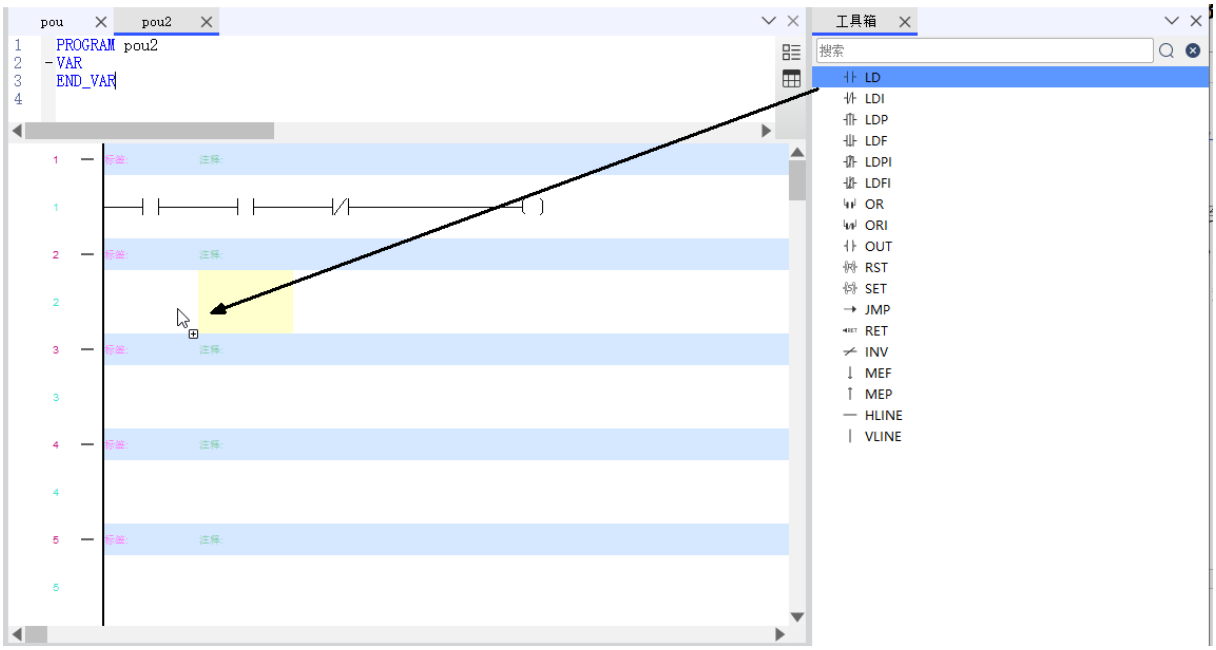
4.2.2 工具箱

LD 编辑器提供了工具箱，工具箱中的指令支持拖拽的插入方式。工具箱可以通过“窗口”菜单下“工具箱”子菜单打开。



图：工具箱

在 LD 编辑器中插入一个元素，要在工具箱中鼠标左击选择它，然后拖动到编辑器窗口。当按住鼠标拖动元素到编辑器窗口中时，在可以插入的位置以”+”符号指示。当放开鼠标时，则这个元素将插入到鼠标指定位置。双击该触点，可以输入变量名称。



图：LD 添加触点

5 传统模式应用指导

LeadStudioV3.0 及以上版本，支持传统模式编程环境。用户在新建工程时，若不勾选“IEC 模式”，则默认创建传统模式工程。

传统模式下，用户可使用软元件进行编程，更易于传统小 PLC 用户上手 LeadStudio 软件，帮助用户快速完成设备开发工作。

注：传统模式工程与 IEC 模式工程不支持互相转换。

5.1 传统模式与 IEC 模式的差异

项目	传统模式	IEC 模式
软元件 X、Y、D、M、S、B、W、R	支持	不支持
STL 指令	支持	不支持
软元件保持范围设置	支持	不支持
局部变量	FB/FC 支持	PRG/FB/FC 支持
监视列表	支持监控软元件	不支持监控软元件
软元件表	支持	不支持

库管理器	不支持	支持
特殊软元件（M8000/D8000）	支持	不支持
任务配置	无需任务配置	需任务配置
子程序、中断子程序	支持	不支持

5.2 工程项目树

软件左侧的工程项目树主要包括两大部分：设备配置和编程调试。如下图所示：

①设备配置：主要包含 PLC 的硬件配置（PLC 模块的配置）、通讯配置（PLC 串口和网口的通讯配置）以及轴运动配置（单轴、轴组以及凸轮配置）等

②编程调试：用户编程及调试的区域，主要包含程序组织单元、全局变量以及监控列表等

5.3 用户编程区域

5.3.1 程序组织单元

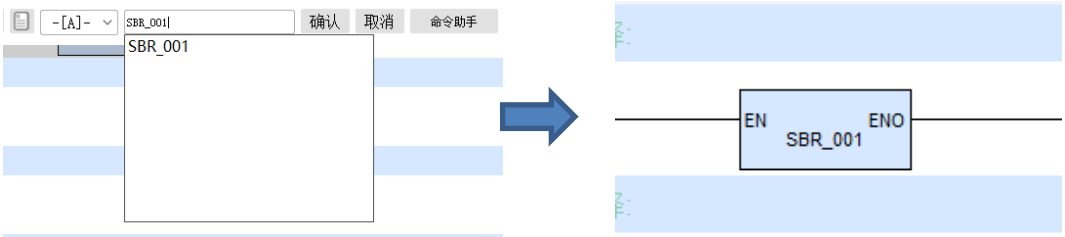
程序组织单元在设备树中的位置如下：

1）程序类型

程序组织单元有三种类型的程序：主程序、子程序以及中断程序，其中主程序只有一个，而子程序和中断程序可以添加多个。

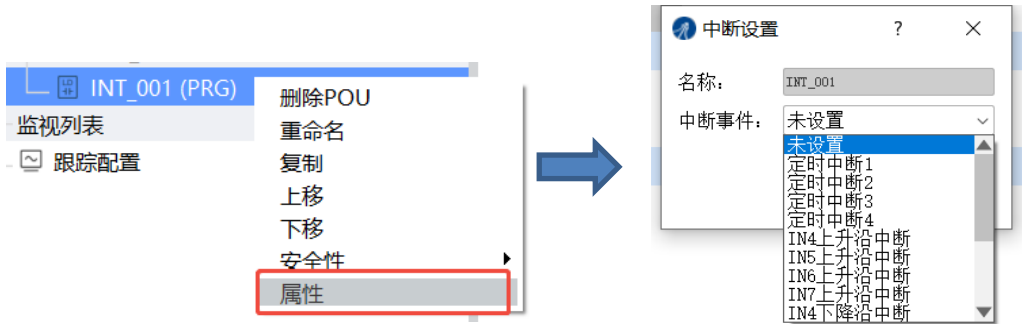
①主程序（MAIN）：PLC 启动后一直循环运行的程序，只支持梯形图编程。可通过右键“程序组织单元”，选择“属性”来配置主程序的看门狗时间和恒定扫描周期，如下图所示：

②子程序（SBR_001）：可通过主程序调用执行，不被主程序调用时 PLC 不会执行子程序。调用方法：在主程序中输入子程序名称，如下图所示：

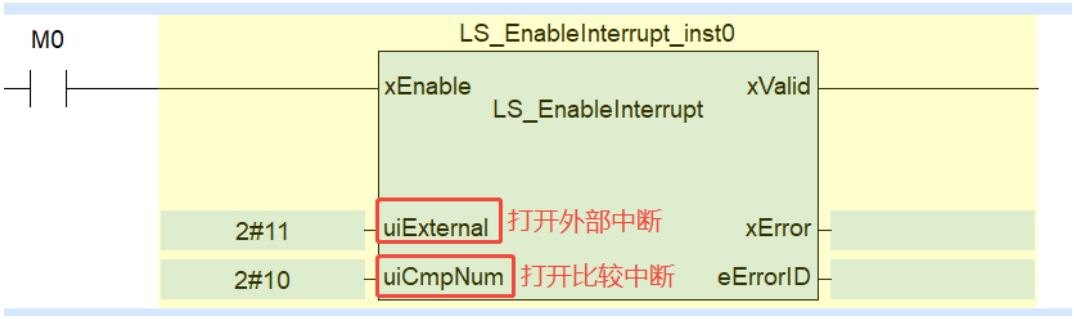


③中断程序（INT_001）：通过中断条件触发执行，中断条件设置方法：右键“中断

程序名称”，选择“属性”，下拉菜单选择中断触发事件，如下图所示：



此外，外部输入中断和比较中断需要指令 LS_EnableInterrupt 去控制使能，如下所示：



比如 uiExternal 输入 3（二进制为 2#11），代表输入端子 IN0 和 IN1 被打开；uiCmoNum 输入 2（二进制为 2#10），代表高速计数比较器 1 被打开

比较中断通过配合高速计数器比较指令 LS_Compare、LS_CompareFIFO 实现中断触发，当高速计数器的计数值大于指令设定的比较值时触发比较中断。

子程序和中断程序可以添加多个，添加方法：右键“程序组织单元”，插入子程序或中断程序，选择程序的编程语言（可选择 ST 或梯形图），如下图所示：

2）FB、FC 块

①添加方法

右键“程序组织单元”，插入 FB 或者 FC，选择编程语言（可选择 ST 或梯形图）以及数据类型，如下图所示：

成功添加后，FB/FC 块会在挂在项目树的程序组织单元下

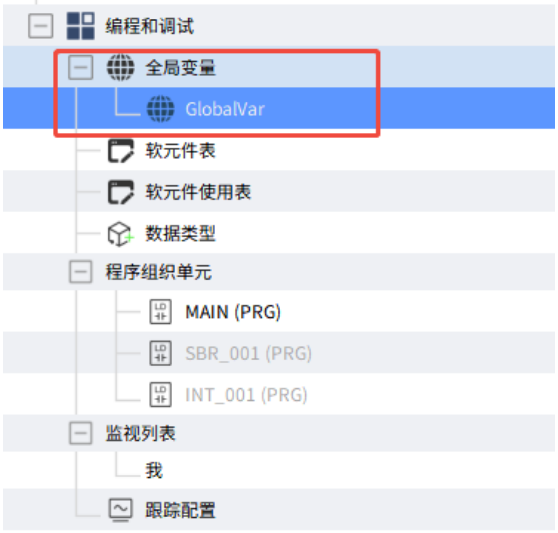
②局部变量声明

FB、FC 的局部变量声明区如下：

通过声明区左上角的四个按钮  可实现局部变量的添加、排序以及删除，按钮从左至右分别为新建、上移、下移、删除。

5.3.2 全局变量

全局变量表在设备树中的位置如下：

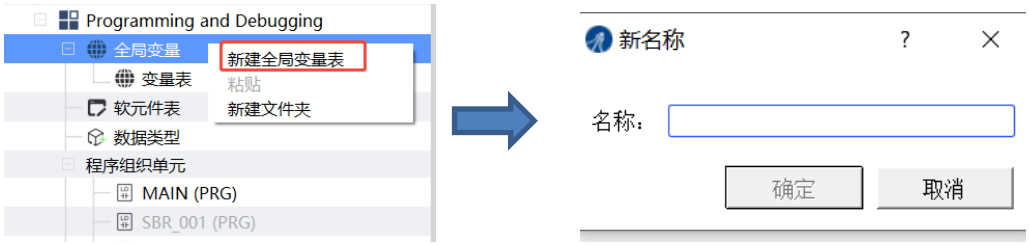


“GlobalVar”为系统默认变量表，自动实例化的变量会添加到该变量表，且该变量表无法被删除

1）添加变量表

全局变量以表格的形式存在，用户可添加多个全局变量表。

添加方法：右键“全局变量”，选择新建全局变量表，输入变量表名称，如下图所示：

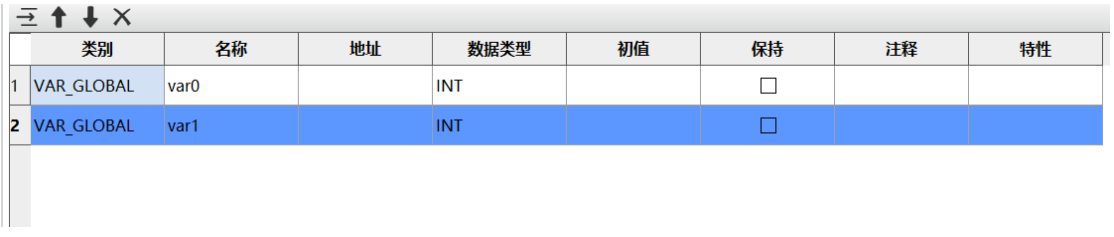


2）全局变量声明

全局变量有两种声明方法

①在全局变量表中声明

点击新建按钮  ，在表格中添加新变量，然后编辑变量的名称、数据类型以及地址等信息

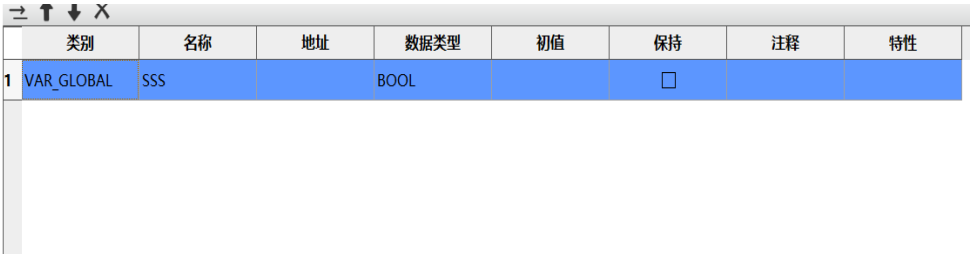


	类别	名称	地址	数据类型	初值	保持	注释	特性
1	VAR_GLOBAL	var0		INT		☐		
2	VAR_GLOBAL	var1		INT		☐		

②在程序（包括主程序、子程序以及中断程序）的变量声明窗口中声明

系统检测到未声明的变量，自动弹出声明窗口

声明确认后，全局变量表中生成变量



	类别	名称	地址	数据类型	初值	保持	注释	特性
1	VAR_GLOBAL	SSS		BOOL		☐		

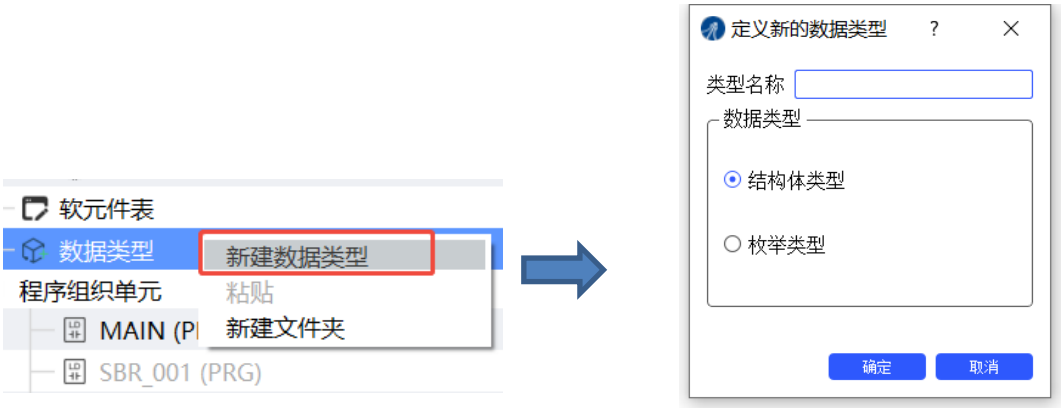
5.3.3 数据类型

数据类型在设备树中的位置如下：

1）添加数据类型

可添加的数据类型：结构体、枚举

添加方法：右键“数据类型”，选择新建数据类型，输入类型名称，如下图所示：

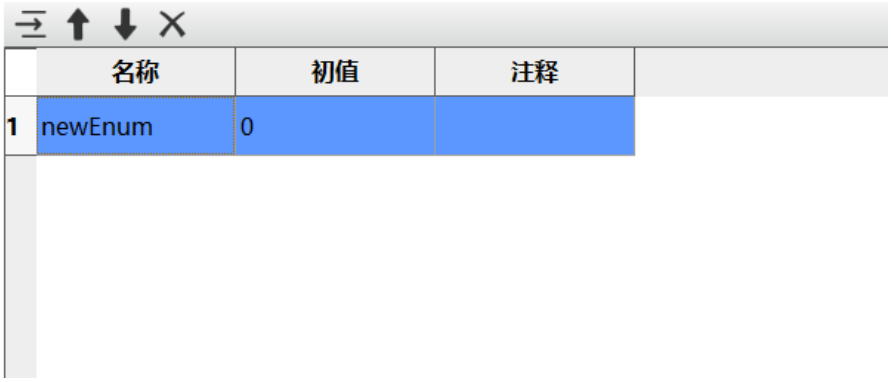


2）编辑数据类型

①结构体编辑

如上图所示，结构体编辑画面同全局变量表类似，都以表格形式展现，通过左上角按钮  可添加或删除结构体成员

②枚举编辑

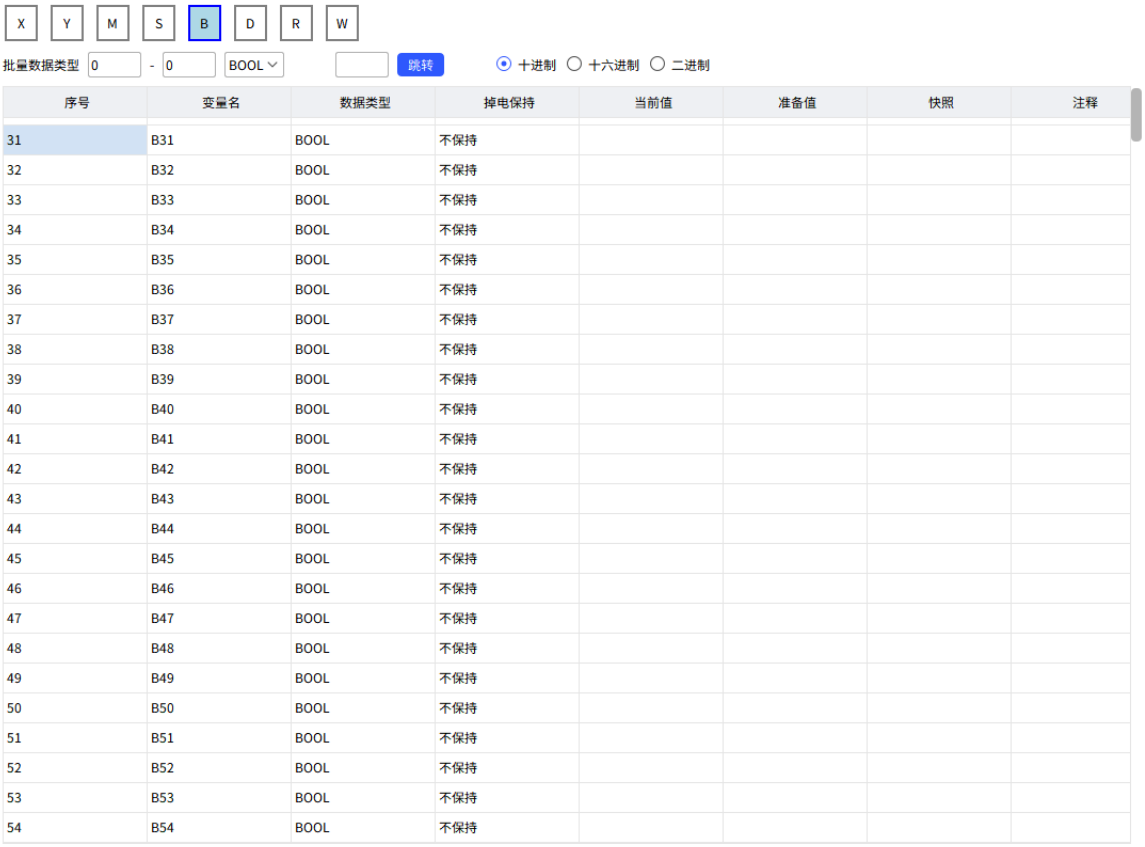


如上图所示，枚举编辑同结构和全局变量表类似，编辑方法参照结构体

5.3.4 软元件表

软元件表在设备树中的位置如下：

双击工程树的“软元件表”即可打开系统软元件表格，如下所示：



序号	变量名	数据类型	掉电保持	当前值	准备值	快照	注释
31	B31	BOOL	不保持				
32	B32	BOOL	不保持				
33	B33	BOOL	不保持				
34	B34	BOOL	不保持				
35	B35	BOOL	不保持				
36	B36	BOOL	不保持				
37	B37	BOOL	不保持				
38	B38	BOOL	不保持				
39	B39	BOOL	不保持				
40	B40	BOOL	不保持				
41	B41	BOOL	不保持				
42	B42	BOOL	不保持				
43	B43	BOOL	不保持				
44	B44	BOOL	不保持				
45	B45	BOOL	不保持				
46	B46	BOOL	不保持				
47	B47	BOOL	不保持				
48	B48	BOOL	不保持				
49	B49	BOOL	不保持				
50	B50	BOOL	不保持				
51	B51	BOOL	不保持				
52	B52	BOOL	不保持				
53	B53	BOOL	不保持				
54	B54	BOOL	不保持				

通过点击表格上方的 

可切换软元件表显示的软元件类型。

软元件表的功能：

①可监控 PLC 所有软元件的状态值

②可通过“快照”可实现软元件初值的保存和下载

③注释软元件

快照的使用方法：可在表格的快照一栏中输入值，通过鼠标右键可选择读内存和写内存；读内存：将软元件的当前值赋值给快照值，写内存：将快照值赋值给软元件的当前值。

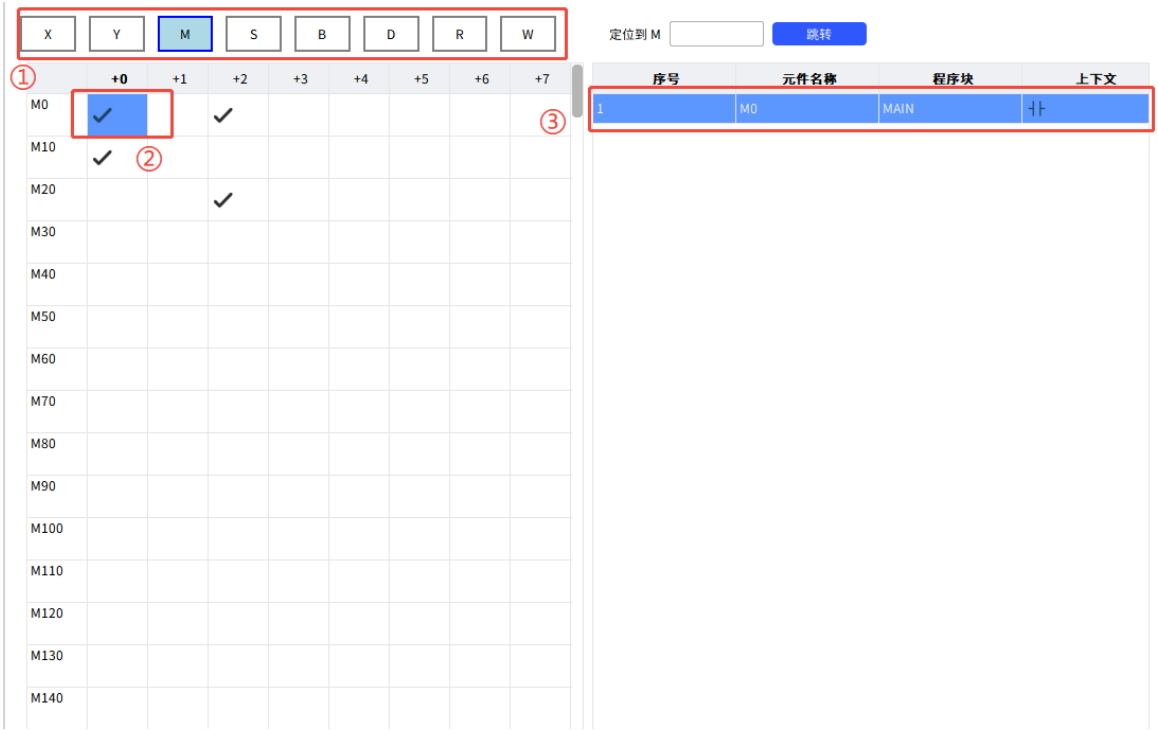
5.3.5 软元件使用表

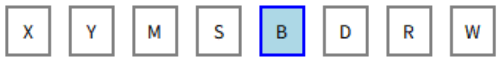
LeadSysV3.1 以上版本的传统模式，支持软元件使用表功能，用户可通过软元件使

用表查看 PLC 的软元件使用情况。

软元件使用表在设备树中的位置如下：

双击“软元件使用表”可查看系统软元件的使用情况，如下所示：



①通过 

可切换软元件使用表的软元件类型。

②若软元件在程序中被使用，则软元件的对应格子会显示“√”

③双击可跳转到软元件被使用的上下文位置

5.3.6 监视列表

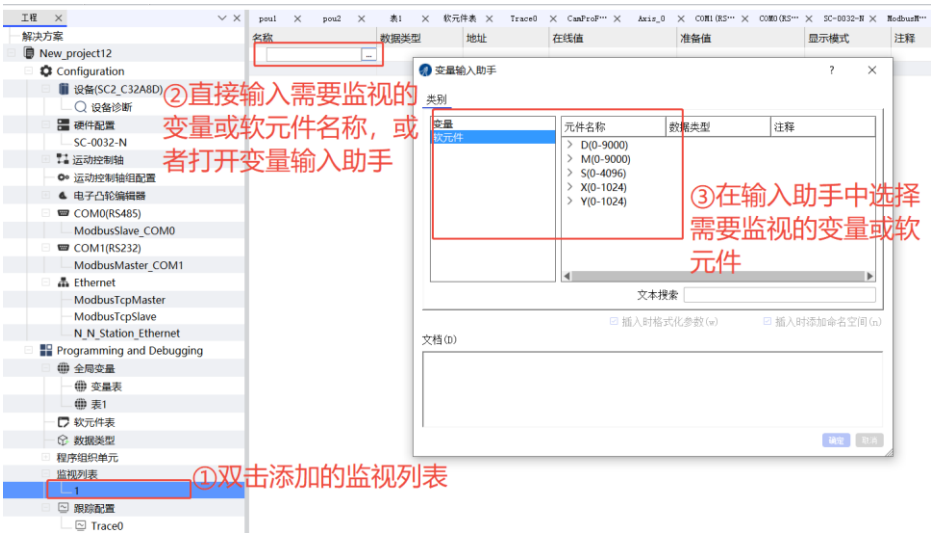
监视列表在设备树中的位置如下：

1）添加监视列表

添加方法：右键“监视列表”，点击新键监视列表，命名列表名称，如下图所示：

2）添加监视变量

添加方法如下图所示：



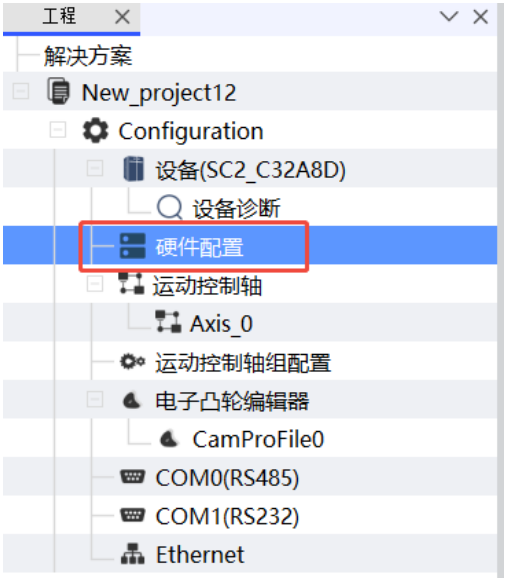
3）批量监控软元件

软件版本要求：V3.1 及以上

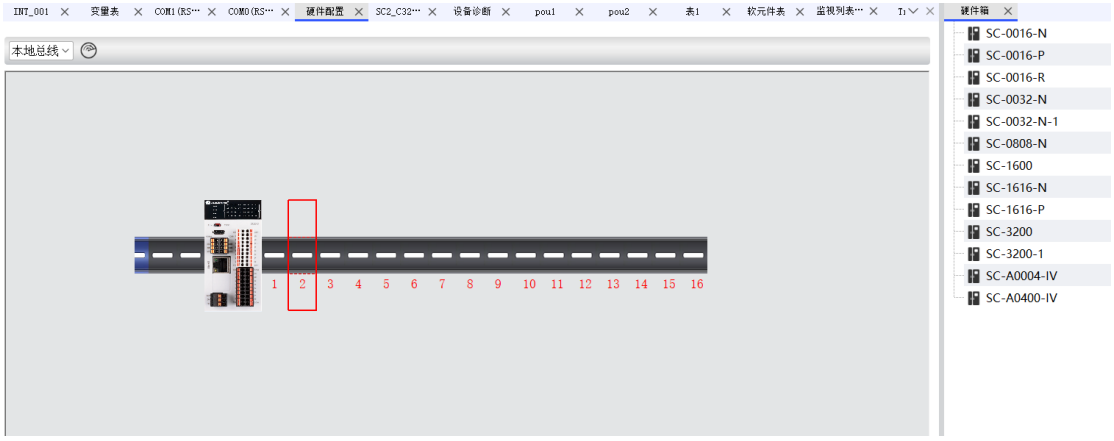
操作方法：①鼠标右键，点击批量监控；②选择监控的软元件类型及范围

5.4 设备配置区域

5.4.1 硬件配置

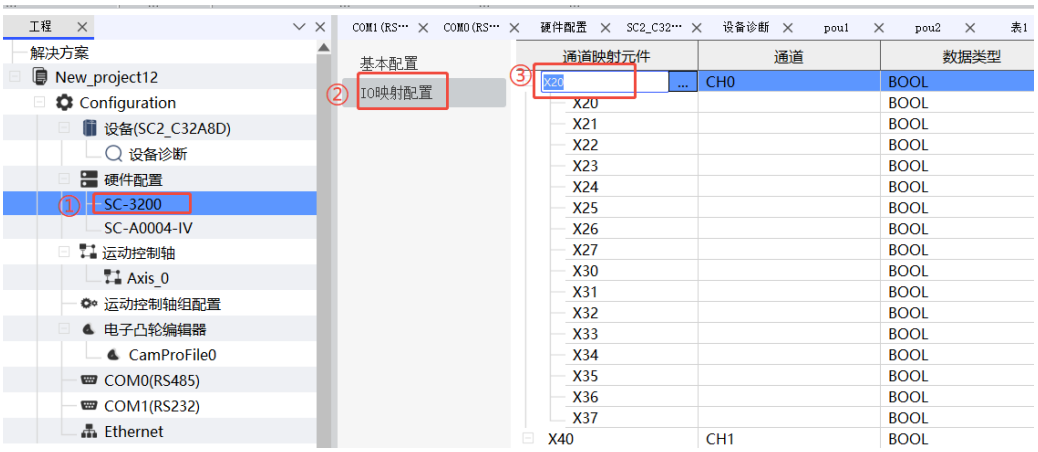


双击“硬件配置”可进入本地模块配置界面，如下图所示：



通过右侧的硬件箱，可以添加扩展模块到 PLC 本地总线上，如下图所示：

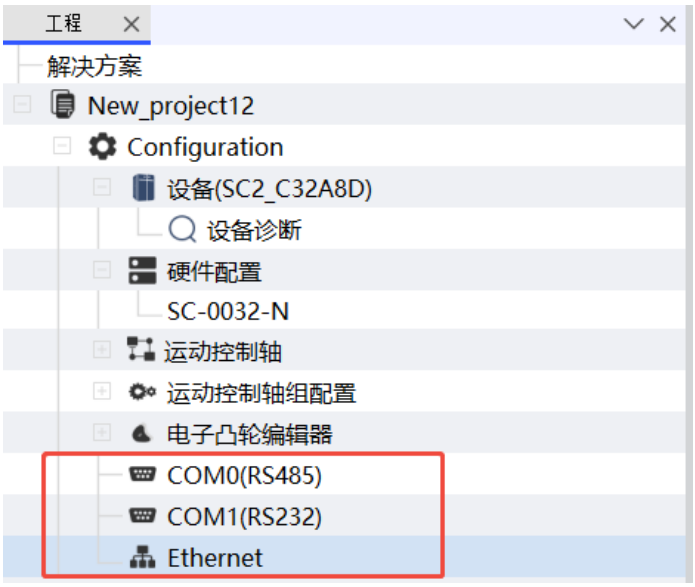
模块的配置方法：①在项目树双击已添加的模块；②点击“IO 映射配置”；③配置软元件地址（数字量对应 X 或 Y，模拟量对应 D）；如下图所示：



5.4.2 运动控制配置

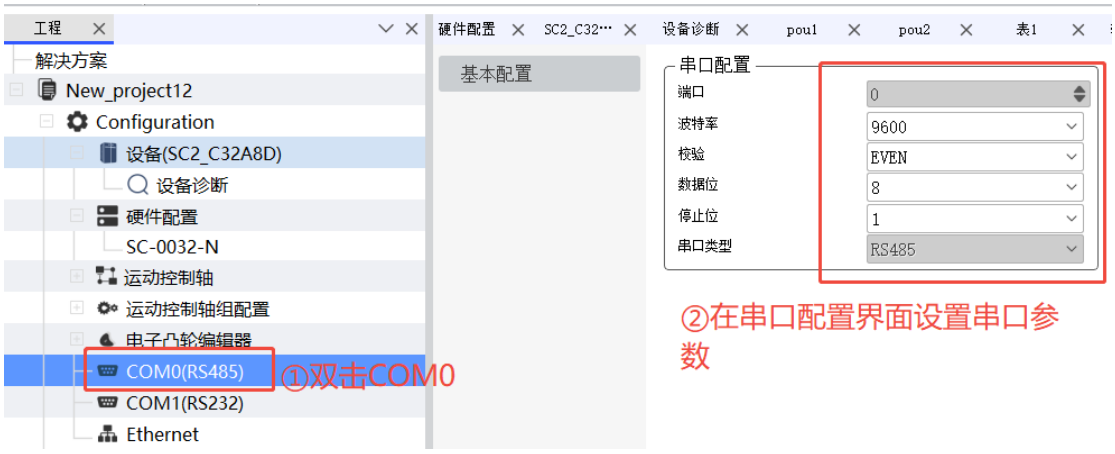
配置方法同 IEC 模式，具体设置方法参考 LeadStudio 编程及应用手册，第 12 章。

5.4.3 通讯配置



1）RS485 串口配置（COM0）

串口参数配置方法：双击“COM0”，在串口参数配置界面中设置串口参数；如下图所示：

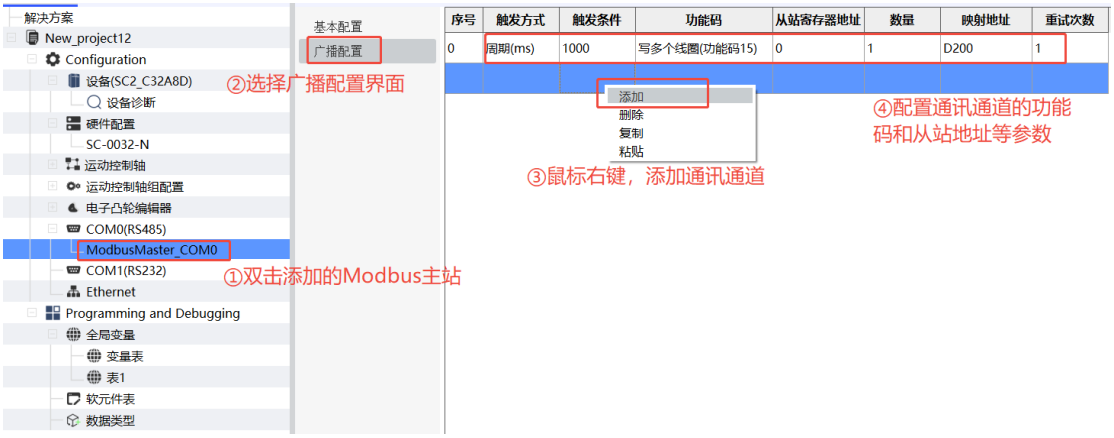


①设置 MODBUS 主站

添加 MODBUS 主站方法：右键“COM0”，插入设备，选择 ModbusMaster_COM0，如下图所示：

添加成功后，MODBUS 主站会挂在 COM0 下，如下图所示：

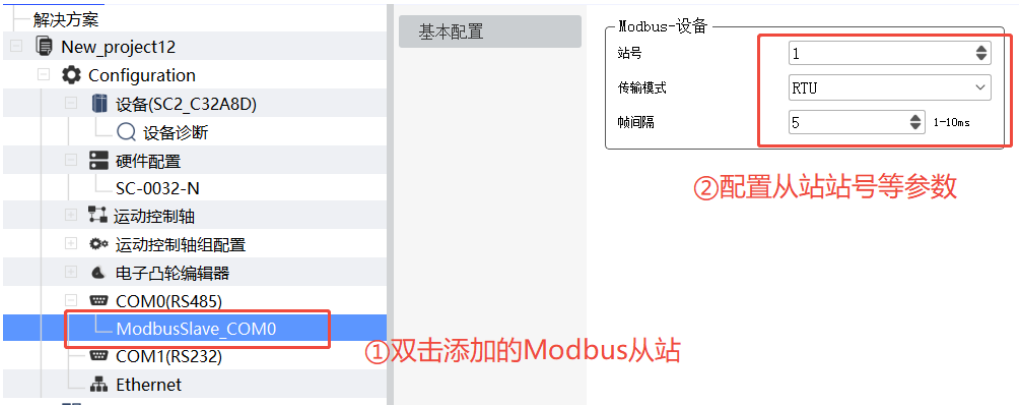
配置 Modbus 主站读写的方法：双击添加的 Modbus 主站，选择广播配置界面，鼠标右键添加通讯通道，配置通讯通道参数；如下图所示：



②设置 MODBUS 从站

添加 MODBUS 从站方法：同添加 MODBUS 主站的方法，如下图所示：

配置 Modbus 从站的方法：双击添加的 Modbus 从站，设置从站站号等参数；如下图所示：



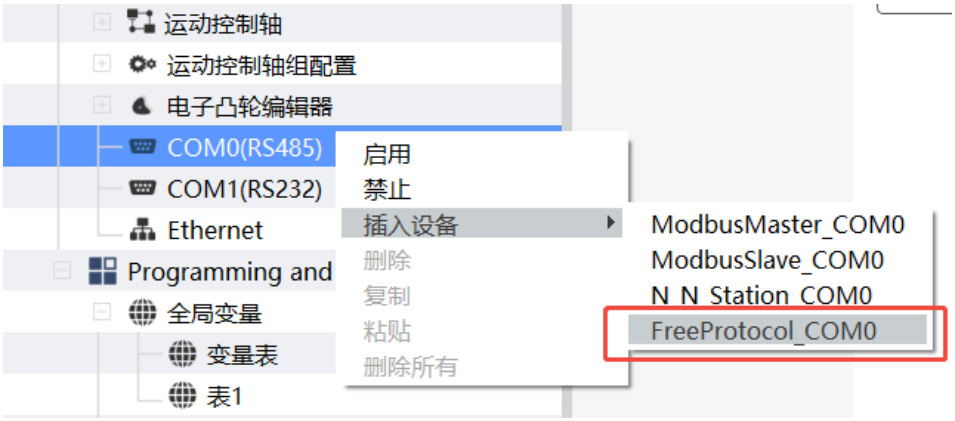
③设置自由串口

添加自由串口的方法：同添加 MODBUS 主站的方法，如下图所示：

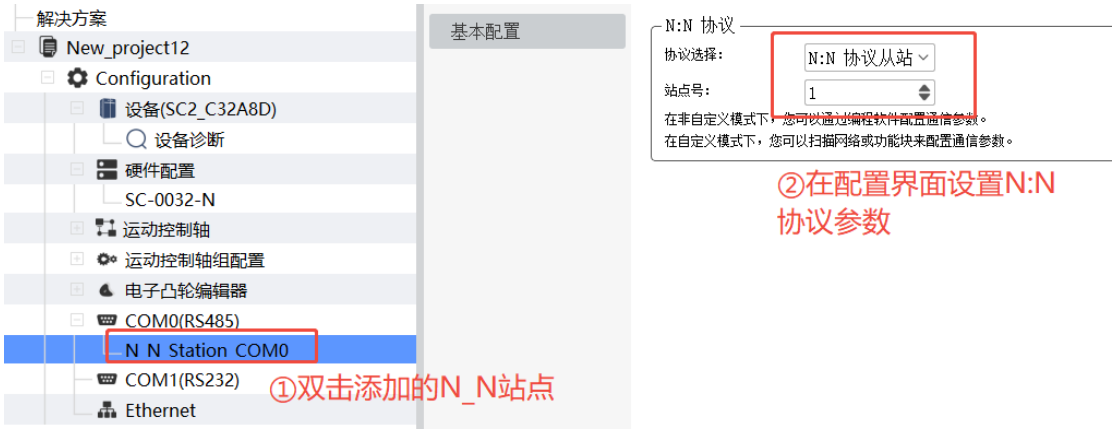
配置自由串口的方法：自由串口无需配置，只需设置好串口参数即可使用通讯功能块发送和接收字节数据。

④设置 N:N 协议串口（多机互联串口）

添加 N:N 协议串口的方法：同添加 MODBUS 主站的方法，如下图所示：



配置 N:N 协议串口的方法：双击添加的 N:N 站点，在配置界面设置 N:N 协议参数（具体设置方法参考 LeadStudio 编程及应用手册，第 10 章）



2）RS232 串口配置（COM1）

配置方法同 RS485 串口，唯一区别在于不支持添加 N:N 协议串口；

3）Ethernet 网口配置

配置方法同 RS485 串口，唯一区别在于 Ethernet 网口可以存在多种协议，而串口只能存在一种协议，如下图所示：

5.5 软元件

5.5.1 软元件说明

软元件种类：X ,Y, M, S, D, B, W, R

1）软元件范围

①位软元件：

软元件区域	范围	注释
X	0-1777(八进制)	输入
Y	0-1777(八进制)	输出
M	0-8999(十进制)	辅助继电器，8000-8999 为特殊继电器
B	0-32767(十进制)	辅助继电器
S	0-4095(十进制)	状态继电器，可作步进状态

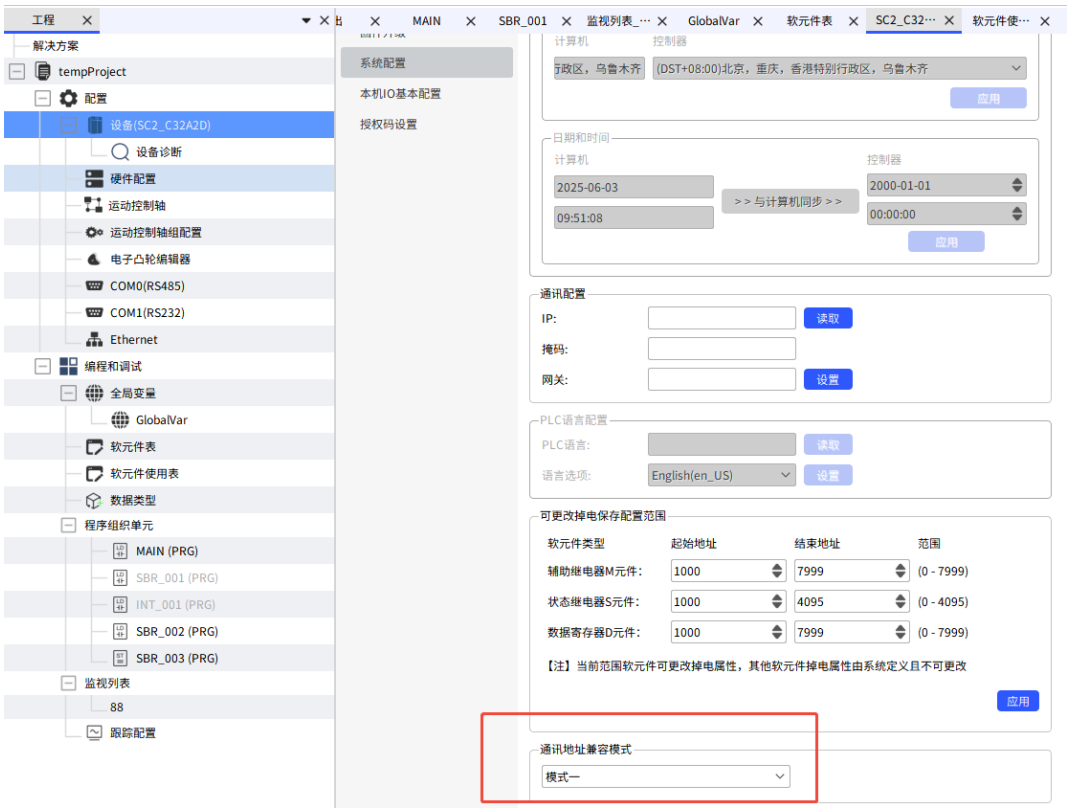
②字软元件

软元件区域	范围	注释
D	0-8999(十进制)	辅助字寄存器，8000-8999 为特殊寄存器
R	0-32767(十进制)	辅助字寄存器
W	0-32767(十进制)	辅助字寄存器，不支持

		Modbus 协议通讯

2）软元件通讯地址

LeadStudioV3.1 及以上版本支持两种通讯地址模式（V3.0 版本只支持模式一通讯地址），可在设备->系统配置->通讯地址兼容模式进行切换，系统默认为模式一



模式一可被 Modbus 主站访问的地址如下：

软元件	范围	地址(16 进制)
S	S0-1023	0x0000-0x03FF
X	X0-377(八进制)	0x0400-0x04FF
Y	Y0-377（八进制）	0x0500-0x05FF
M	M0-1535	0x0800-0x0DFF
	M1536-4095	0xB000-0xB9FF
D	D0-4095	0x1000-0x1FFF

	D4096-8999	0x9000-0xA327

模式二可被 Modbus 主站访问的地址如下：

软元件	范围	地址(16 进制)
M	M0-7999	0x0000-0x1F3F
B	B0-32767	0x3000-0xAFFF
S	S0-S4095	0xE000-0xEFFF
X	X0-X1777（八进制）	0xF800-0xFBFF
Y	Y0-Y1777（八进制）	0xFC00-0xFFFF
D	D0-D7999	0x0000-0x1F3F
R	R0-R32767	0x3000-0XAFFF
W	W0-W32767	不支持通过 Modbus 协议进行访问

3）软元件掉电保持区域

默认掉电保持区域：

M：M1000-M7999 为保持继电器

S： S1000-S4095 为保持状态继电器

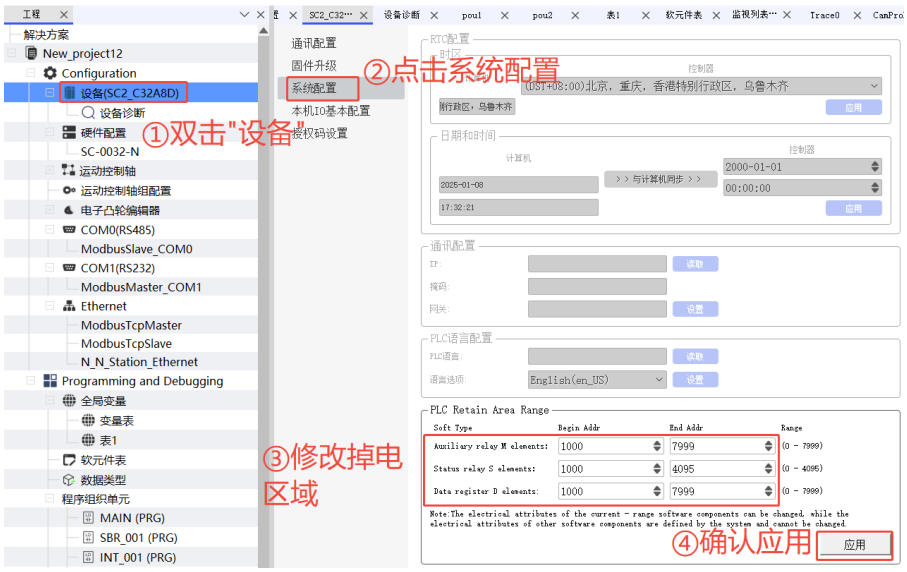
B: B1000-B32767 为保持继电器

D：D1000-D7999 为保持寄存器

R: R1000-R32767 为保持寄存器

W: W1000-W32767 为保持寄存器

用户可自由修改掉电保持区域，方法如下所示：



注：①只支持软元件 S,M,D 的掉电区域修改。 ②软元件 M 和 S 的掉电范围必须为 8 的倍数（起始地址为 8 的倍数，且确保掉电范围内的个数也为 8 的倍数），否则将无法设置成功。

4）特殊软元件

如下表所示：

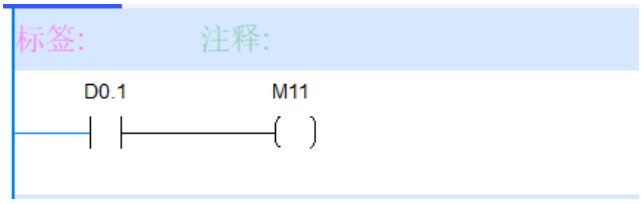
元件	功能
D8013	实时时钟秒（0-59）
D8014	实时时钟分（0-59）
D8015	实时时钟小时（0-23）
D8016	实时时钟日（1-31）
D8017	实时时钟月（1-12）
D8018	实时时钟年（2000-2099）
M8000	运行时常 ON
M8001	运行时常 OFF
M8002	程序运行的第一个周期为 ON
M8003	程序运行的第一个周期为 OFF
M8005	RTC 时钟的电池电压过低时

	动作
M8011	10ms 时钟周期的振荡时钟
M8012	100ms 时钟周期的振荡时钟
M8013	1S 时钟周期的振荡时钟
M8014	1 分钟时钟周期的振荡时钟

5.5.2 字软元件使用技巧

1）位访问

表示方法：以寄存器 D 为例，表示方法为 Dn.m，其中 0<=m<16，表示 Dn 寄存器的第 m 位，如下图所示（使用 D0 寄存器的第 1 位作为触点导通线圈）：



其他字软元件 R/W 的位访问方法同寄存器 D 一致，如 Rn.m/Wn.m

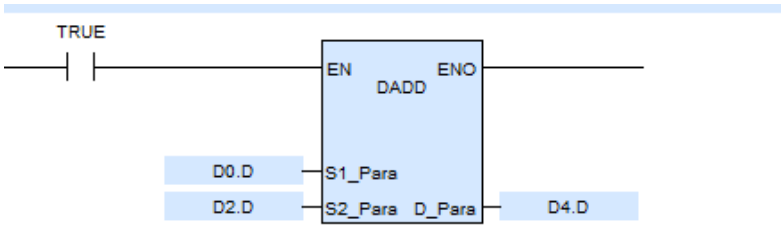
2）采用后缀区分数据类型

后缀区分的数据类型如下表所示：

后缀	功能
.U	无符号 16 位整型，占用一个寄存器
.S	有符号 16 位整型，若不加后缀，默认为此类型，占用一个寄存器
.UD	无符号 32 位整型，占用连续的两个寄存器
.UL	无符号 64 位整型，占用连续的四个寄存器
.D	有符号 32 位整型,占用连续的两个寄存器
.L	有符号 64 位整型,占用连续的四个寄存器
.F	32 位浮点数，占用连续两个寄存器，该后缀使用时，对应的寄存器地址必须 4 字节对齐
.DF	64 位浮点数，占用连续四个寄存器，该后缀使用时，对应的寄存器地

	址必须 8 字节对齐
.T	String 类型，长度为 80 个字符，占用连续的 41 个寄存器元件，字符串的最后自动存储 NUL(00H)
.DT	WString 类型，长度为 80 个字符，占用连续的 81 个寄存器元件，字符串的最后自动存储 NUL(0000H)

例如使用 32 位整型加法指令 DADD，就需要使用寄存器后缀“.D”来表示 32 位整型（占用两个寄存器），如下所示：



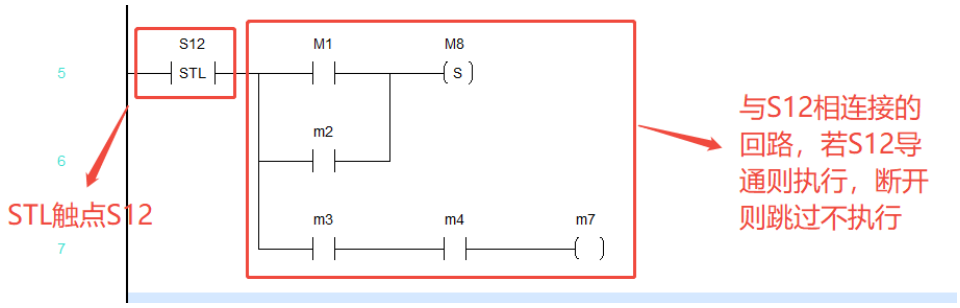
表示 32 位整型寄存器 D0 和 D2 相加，结果赋值给另一个 32 位整型寄存器 D4

5.5.3 软元件 S 与步进指令 STL

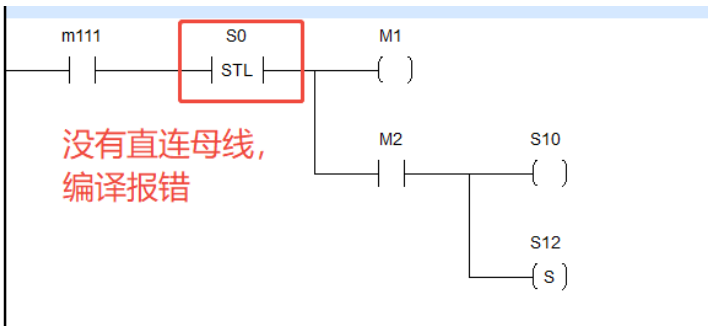
在基础指令中，S 可被当作 M 使用；在 STL 指令中，S 作为步进状态。

1）步进指令使用方法

若 STL 触点 S 导通，则与其相连的回路执行动作；若 S 触点断开，与其相连的回路将跳过不执行；如下图所示：

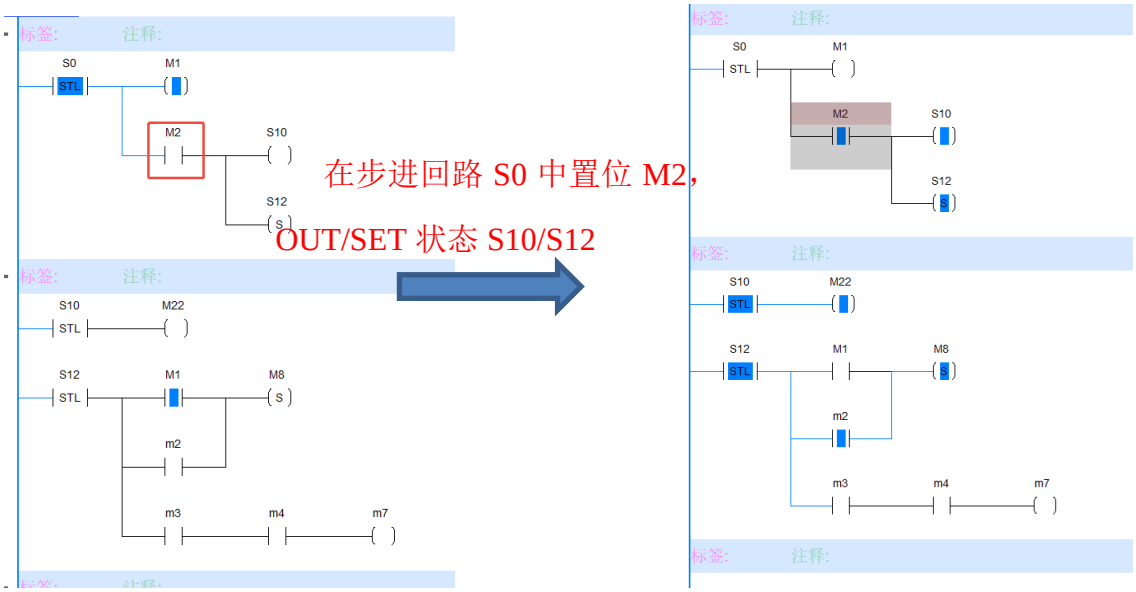


注：STL 触点 S 必须直连母线，否则将编译报错，如下所示：



2）步进跳转

在一个步进回路内，若 OUT/SET 了其它的状态 S，当前状态会被复位，并跳转到其它状态，如下所示：

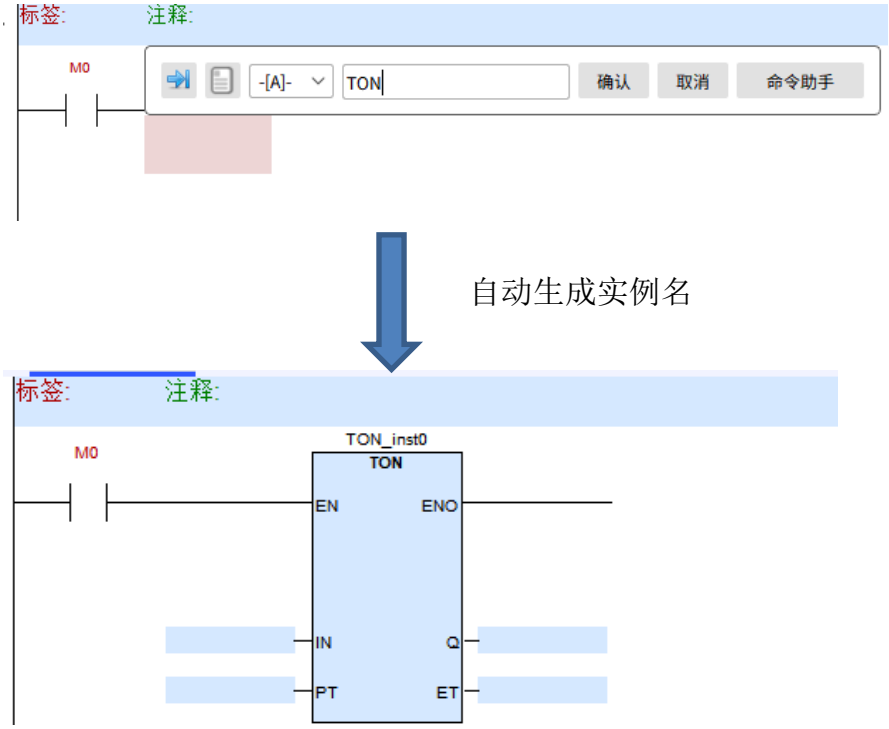


如上图所示，在回路 S0 中 OUT/SET 了状态 S10/S12 后，S0 状态被复位，步进回路跳转到了 S10 和 S12

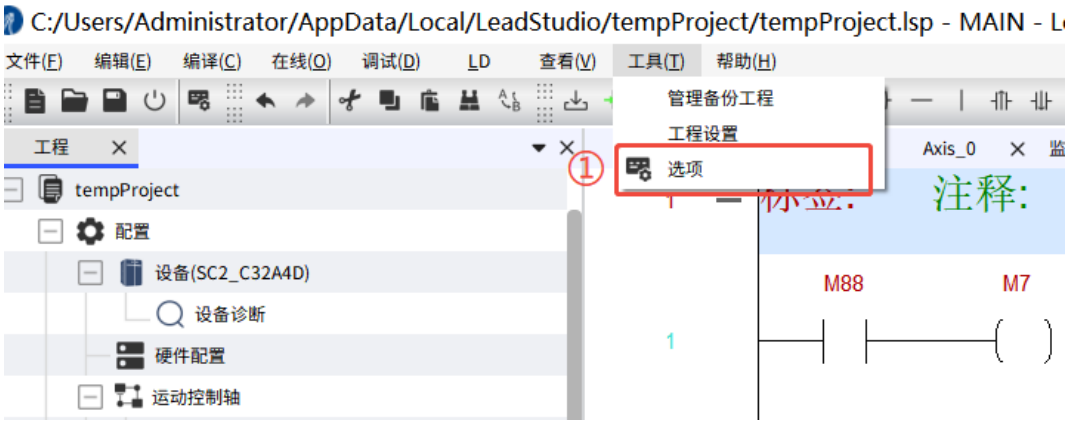
5.6 自动实例化 FB

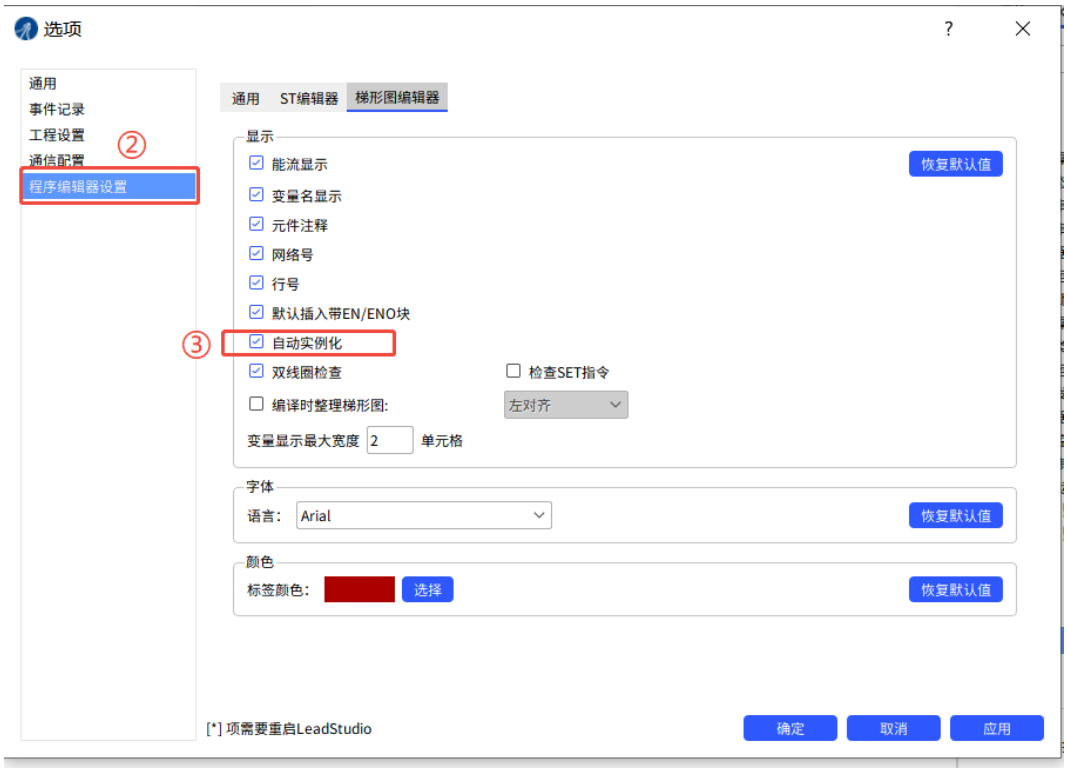
LeadStudioV3.1 及以上版本支持自动实例化 FB，用户在梯形图新增功能块时，系统会自动帮用户创建实例到“默认变量表”，无需用户手动实例化。

在梯形图输入 FB 名时，系统将自动生成实例名；若用户想自定义实例名称，则在梯形图中输入 FB 名+“空格”+“自定义名称”即可。



自动实例化功能可在配置选项中选择是否启用，勾选启用，不勾选则不启用。该选项系统默认勾选，操作步骤如下；



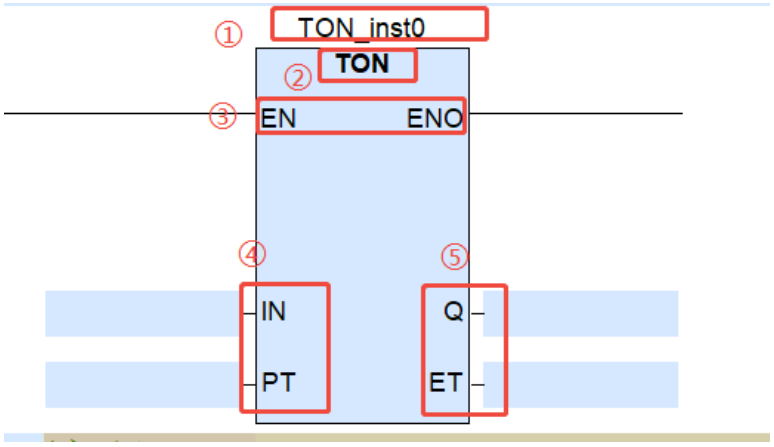


5.7 常用功能块

5.7.1 定时器功能块

以通电延时定时器 TON 为例

1）功能块表现形式



①定时器的实例名

使用定时器功能块时必须实例化，相当于声明一个定时器类型的变量，实例名即变量名。

②定时器名称

定时器功能块的名称，即指令名称。

③功能块的 EN/ENO 引脚

En 引脚导通后功能块才会被调用，可在右键选项隐藏 EN/ENO 引脚或在工具选项设置中默认隐藏该引脚（隐藏时默认一直调用功能块）。

④定时器的输入引脚

引脚 IN:BOOL 类型，输入有效时定时器开始计时；引脚 PT：TIME 类型，设定定时的预设时间。

⑤定时器的输出引脚

引脚 Q:BOOL 类型，定时到达预设时间置 ON；引脚 ET:TIME 类型，输出当前累计计时时间。

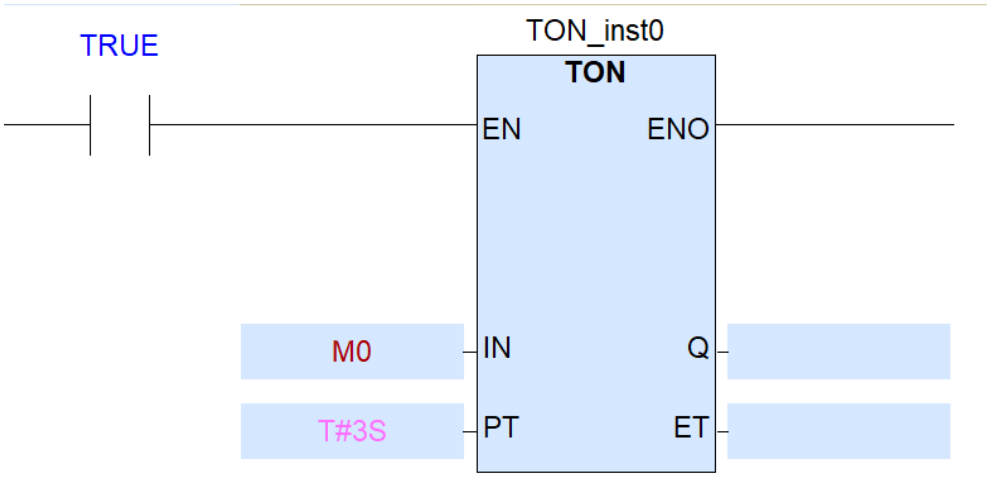
2）使用方法

键盘输入“TON”,梯形图在光标处生成功能块并自动声明实例，用户只需为功能块分配输入输出引脚即可。

说明：自动声明实例需勾选选项（选项默认勾选），勾选方法见章节 5.6

具体操作步骤如下（EN/ENO 引脚未隐藏与隐藏略有差别）；

EN/ENO 引脚未隐藏：



①使用功能块时需要一直导通 EN 引脚，功能块才可被调用。

②为输入引脚 IN 分配位元件或位变量，通过位元件或变量控制定时器的启停。

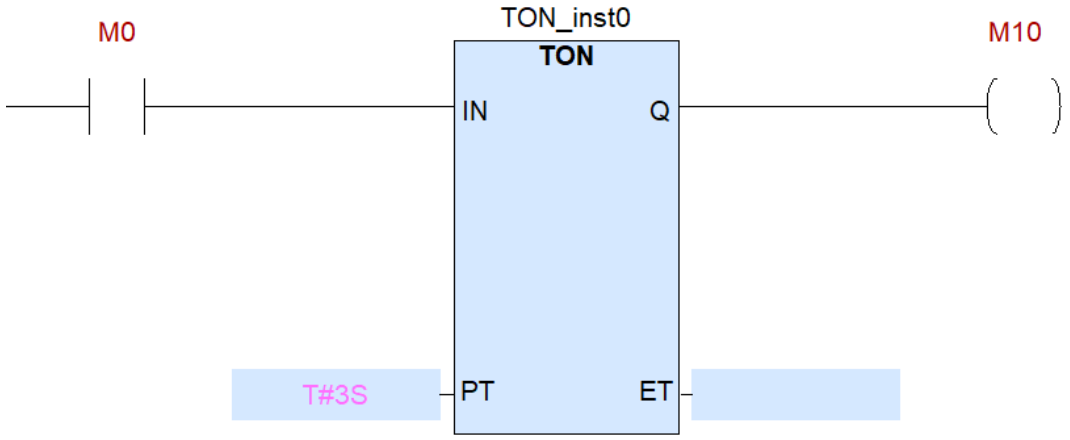
③为输入引脚 PT 分配 TIME 类型常量或变量，即设定定时器的预设时间。

④为输出引脚 Q 分配位元件或位变量、输出引脚 ET 分配 TIME 类型变量，引脚 Q 和 ET 代表定时器的完成状态和累计时间

⑤输入引脚 IN 分配的位元件置 ON 时，定时器启动定时，输出引脚 ET 累计当前定时时间；当定时时间到达 PT 引脚设定的预定时间时，定时器输出引脚 Q 置 ON。

⑥输入引脚 IN 分配的位元件置 OFF 时，定时器停止定时，输出引脚 ET 的累计时间清零，输出引脚 Q 置 OFF。

EN/ENO 引脚隐藏：



①通过与 IN 引脚连接的触点控制定时器的启停.

②为输入引脚 PT 分配 TIME 类型常量或变量，即设定定时器的预设时间。

③输出引脚 Q 后可连接线圈，受定时器的完成状态控制；为输出引脚 ET 分配 TIME 类型变量，输出当前的累计时间

④输入引脚 IN 连接的触点置 ON 时，定时器启动定时，输出引脚 ET 累计当前定时时间；当定时时间到达 PT 引脚设定的预定时间时，与输出引脚 Q 连接的线圈置 ON。

⑤输入引脚 IN 连接的触点置 OFF 时，定时器停止定时，输出引脚 ET 的累计时间清零，与输出引脚 Q 连接的线圈置 OFF。

说明：①功能块的输出引脚可不分配，使用 TON.ET、TON.Q 也可访问定时器的累计时间及完成状态；②TIME 类型：LeadStudio 表示时间的数据类型，单位如下表所示：

时间单位	符号
时	H

秒	S
毫秒	MS

TIME 类型数据的格式为“T#时间”，例如 50 毫秒表示为“T#50MS”，且时间单位支持混合使用，例如 1.5S 可表示为“T#1S500MS”。

3）与传统定时指令对比

传统定时指令一般形式如下：

“TMR　T0　D2”

TMR 为指令名称，T0 为定时器软元件，D2 为设定时间的寄存器

对比如下：

①定时器功能块的实例名相当于传统指令的软元件 T0,都可访问定时器的计时时间及完成状态，例如 “LD T0”（使用 T 元件在触点作判断）相当于“LD 实例名.Q”。

②EN/ENO 引脚是定时器功能块与传统指令的主要差别之一，定时器功能块需 EN 引脚导通才可被调用，但并非启动定时，需引脚 IN 输入有效才执行定时。

若用户习惯于传统指令的执行方式，可隐藏 EN/ENO 引脚，只需导通引脚 IN 即可触发定时

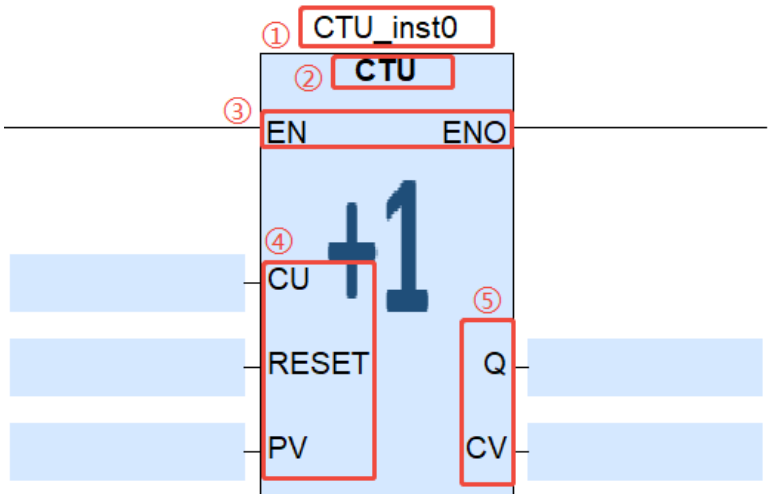
③定时功能块的预设时间与传统指令存在差别,功能块 TON 的预设时间由 TIME 类型的输入引脚 PT 设定（不需要时基），而传统指令以寄存器值*定时器时基来设定预设时间

④功能块 TON 有输出参数，可以将定时器的计时时间及完成状态输出给其他变量，而传统指令没有输出参数。

5.7.2 计数器功能块

以累加计数器 CTU 为例

1）功能块表现形式



①计数器的实例名

使用计数时器功能块时必须实例化，相当于声明一个计数器类型的变量，实例名即变量名。

②计数器名称

计数器功能块的名称，即指令名称。

③功能块的 EN/ENO 引脚

En 引脚导通后功能块才会被调用，可在右键选项隐藏 EN/ENO 引脚或在工具选项设置中默认隐藏该引脚（隐藏时默认一直调用功能块）。

④计数器的输入引脚

引脚 CU:BOOL 类型，上升沿触发计数值加 1；引脚 RESET：BOOL 类型，置 ON 时复位计数器；PV 引脚：WORD 类型，计数器的预设计数值

⑤计数器的输出引脚

引脚 Q:BOOL 类型，计数值等于预设计数值时置 ON；引脚 CV:WORD 类型，输出当前计数值。

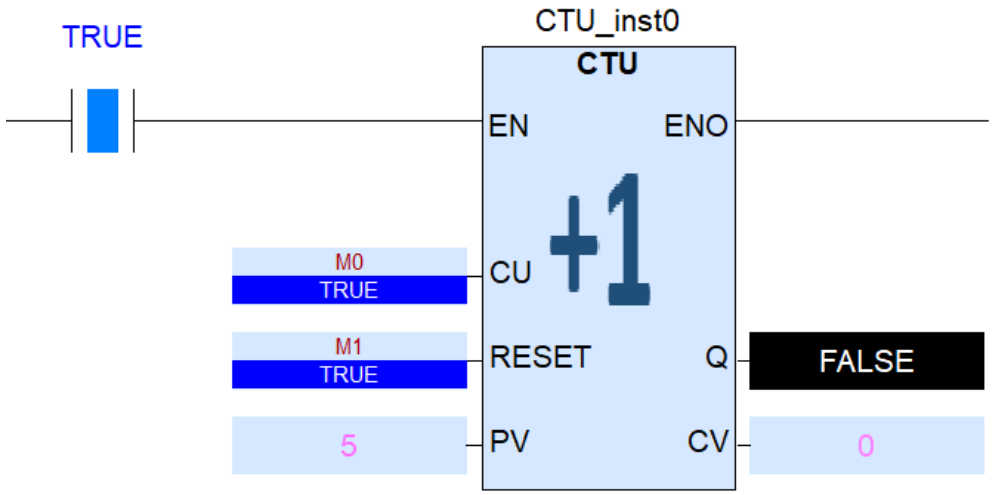
2）使用方法

键盘输入“CTU”,梯形图在光标处生成功能块并自动声明实例，用户只需为功能块分配输入输出引脚即可。

说明：自动声明实例需勾选选项（选项默认勾选），勾选方法见章节 5.6

具体操作步骤如下（EN/ENO 引脚未隐藏与隐藏略有差别）；

EN/ENO 引脚未隐藏：



①使用功能块时需要一直导通 EN 引脚，功能块才可被调用。

②为输入引脚 CU 分配位元件或位变量，通过位元件或变量的上升沿触发计数值加 1。

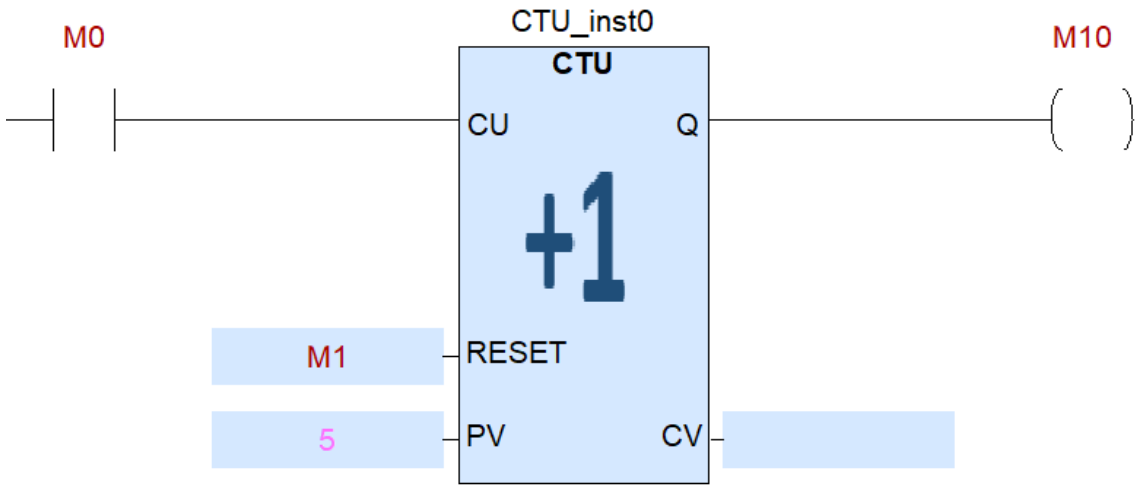
③为输入引脚 PV 分配 WORD 类型常量或变量，即设定计数器的预设计数值。

④为输出引脚 Q 分配位元件或位变量、输出引脚 CV 分配字元件或字变量，引脚 Q 和 CV 代表计数器的完成状态和计数值

⑤输入引脚 CU 检测到上升沿信号时，计数器计数值加 1；当计数值等于预设计数值时，计数器输出引脚 Q 置 ON。

⑥为输入引脚 RESET 分配位元件或位变量，该引脚置 ON 时将复位计数器，输出引脚 Q 置 OFF、CV 值清零，且此时计数器不再检测 CU 引脚的上升沿信号。

EN/ENO 引脚隐藏：



①通过与 CU 引脚连接的触点上升沿信号来触发计数器计数值加 1

②为输入引脚 PV 分配 WORD 类型常量或变量，即设定计数器的预设计数值。

③输出引脚 Q 后可连接线圈，受计数器的完成状态控制；为输出引脚 CV 分配字元件或字变量，输出计数器的计数值

④输入引脚 CU 连接的触点上升沿触发时，计数器计数值加 1；当计数值等于预设计数值时，与输出引脚 Q 连接的线圈置 ON。

⑤为输入引脚 RESET 分配位元件或位变量，该引脚置 ON 时将复位计数器，输出引脚 Q 后的线圈置 OFF、CV 值清零，且此时计数器不再检测 CU 引脚的上升沿信号。

说明：①功能块的输出引脚可不分配，使用 CTU.Q、CTU.CV 也可访问计数器的完成状态及计数值；

3）与传统计数指令对比

传统计数指令一般形式如下：

“CNT　C0　D0”

CNT 为指令名称，C0 为计数器软元件，D0 为预设计数值

对比如下：

①计数器功能块的实例名相当于传统指令的软元件 C0，都可访问计数器的计数值及完成状态，例如“LD C0”（使用 C 元件在触点作判断）相当于“LD 实例名.Q”。

②EN/ENO 引脚是计数器功能块与传统指令的主要差别之一，计数器功能块需 EN 引脚导通才可被调用，CU 引脚在功能块被调用后才可触发计数。

若用户习惯于传统指令的执行方式，可隐藏 EN/ENO 引脚，只需控制 CU 引脚的通断来触发计数

③功能块 CTU 有复位引脚 RESET,通过导通该引脚即可清除计数器，而传统指令没有复位功能，需要用户添加复位指令来清除计数值。

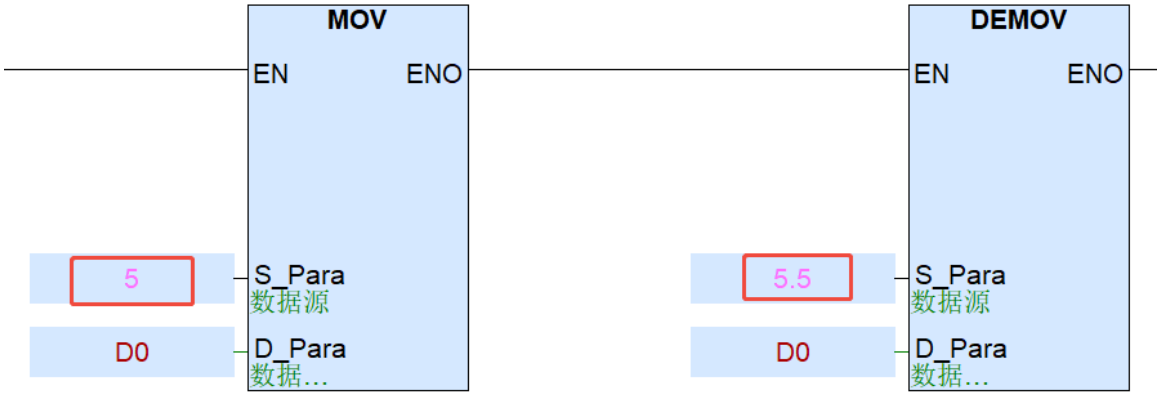
④功能块 CTU 有输出参数，可以将计数器的计数值及完成状态输出给其他变量，而传统指令没有输出参数。

5.8 常数的表示方法

LeadStudio 与传统日系平台一样通过前缀来表示不同类型的常数，但前缀符号有所区别，如下表所示；

常数类型	LeadStudio	传统日系
浮点数	REAL#	E
十进制整数	INT#	K
十六进制整数	16#	H

为提升用户编程效率，LeadStudio 通过用户输入的常数是否有小数点自动判别浮点数和十进制整数；此两种类型常数可免加前缀，如下所示：



此外，LeadStudio 还支持 2 进制、8 进制的整数类型常数，格式为 2#、8#。

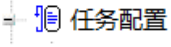
6 任务配置

在任务配置中，定义一个或多个任务来控制和执行控制器中的应用程序。每个应用程序必须包含一个“任务配置”。

“任务”被配置以后，可以使一系列程序或功能块周期性地执行或由一个特定的事件触发开始执行程序。一个任务可调用一个或多个程序块(POU)。程序按照任务下挂载顺序，从上到下依次执行。

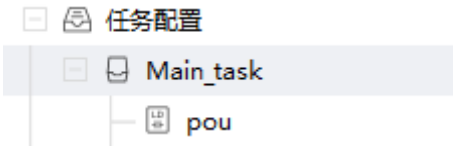
通过结合优先级和条件，可以定义处理任务的顺序。用户可以为每个任务配置看门狗，当任务的执行周期过长时控制器提示异常。

6.1 任务配置

新建工程后，LeadStudio 会在工程管理树中自动添加“任务配置”，如图 

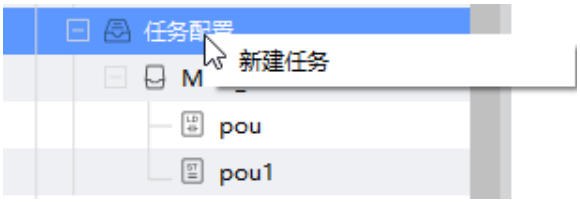
；

SC、SCU 系列产品会自动生成下列任务：



用户可根据需要自行添加任务，步骤如下：

1) 在工程管理树中，右键“任务配置”节点，选择“新建任务”命令。



2) 在弹窗的“新建任务”对话框中，输入任务名称和选择任务类型。其中任务名称应遵循以下规则：

- 名称长度最多不超过 32 字节；
- 只能包含字母、数字、下划线“_”，第一个字符必须是字母或者下划线，且不能为空；
- 不能与关键字、指令库、数据类型、任务、POU、全局变量组、全局变量、默认固件函数重名。

6.2 任务类型

在任务配置中，定义一个或多个任务来控制和执行控制器中的应用程序。每个应用程序必须包含一个“任务配置”。“任务”被配置以后，可以使一系列程序或功能块周期性地执行或由一个特定的事件触发开始执行程序。一个任务可调用一个或多个程序块（POU）。

任务类型及其执行逻辑见下表：

类型	执行逻辑	设置项
循环	按照设定的间隔循环执行	时间间隔

事件	设置的全局变量触发上升沿后，任务开始执行	触发变量
自由运行	在程序开始时和完整过程结束时以连续循环的方式自动开始处理任务	/
状态	设置的全局变量为 True 时，任务开始执行	触发变量
中断	可以配置为当输入状态变化，或高速计数器的读数匹配时，引起对应的 POU 执行一次	触发变量、编码器比较

通过结合优先级和条件，可以定义处理任务的顺序。用户可以为每个任务配置看门狗，当任务的执行周期过长时控制器提示异常。

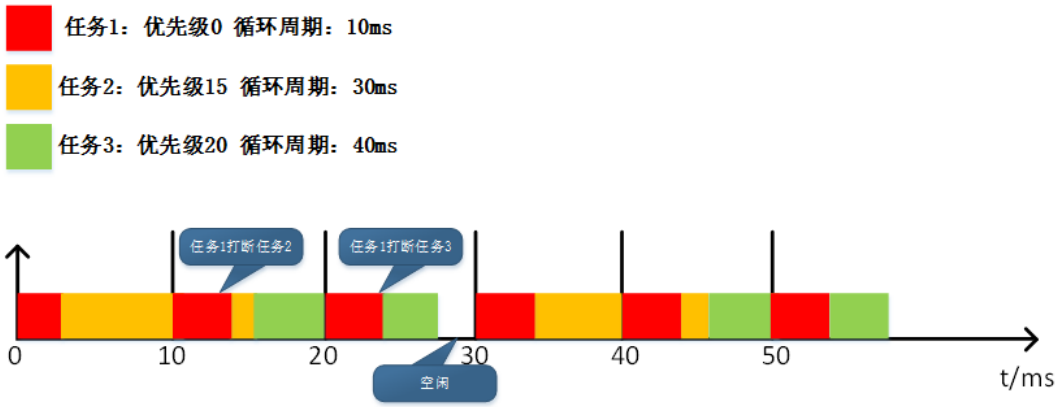
6.3 任务优先级

在 LeadStudio 中，一个工程可以添加多个任务，但是实际执行时，在同一时间只能执行其中一个任务，任务执行的先后受任务的优先级影响，优先级高的任务会优先执行，且可以打断低优先级的任务执行，待高优先级任务执行完成后，低优先级任务继续执行剩余的程序。

LeadStudio 中一共有 0..31 共 32 个优先级，数值越大优先级越低，0 级为最高优先级，当多个任务同时满足触发条件时，按照优先级顺序执行。

例：

程序中存在三个优先级不同的任务，并且具有不同的扫描周期，任务执行的时序图如下：



0~10ms：先执行任务 1，当任务 1 执行完成后执行任务 2；

10~20ms：任务 1 的循环周期到达，由于任务 1 的优先级较高，故直接打断任务 2 的执行，待任务 1 执行完成后，继续执行任务 2 剩余的程序，待任务 2 也执行完成，接着执行任务 3；20~30ms: 同上一周期，由于任务 1 的循环周期已到达，故任务 1 打断任务 3 执行，任务 1 执行完成后，接着执行任务 3 剩余程序，任务 3 执行完成后，控制器无其他任务需要执行，故处于空闲状态；

注意：

- 在任务优先级等级分配时，请勿分配具有相同优先级的任务；
- 尽量不在不同任务中调用同一 POU，否则执行结果可能与预期不符；
- Modbus 或 Socket 通讯类任务保持最高优先级且具有最小循环周期；
- 任务的循环周期不等于任务实际的执行时间；
- 此处只列出任务执行的时序，实际控制器程序运行时，还包含输入扫描、输出刷新、内务处理等阶段。

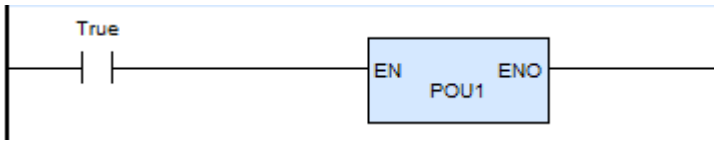
6.3.1 添加 POU 调用

在 LeadStudio 中 POU 需要在任务中调用之后才能执行，添加调用有三种方式：

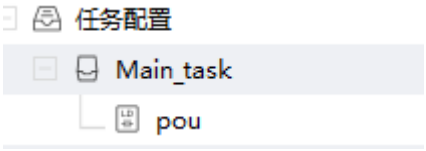
1) 直接拖拽 POU 到相应的任务添加：

2) 在其他 POU 中调用（用于调用其他 POU 的 POU 需在任务中执行）

pou 中的程序：



实际任务配置：



在该情况下，由于 POU1 在 pou 中被调用，故 POU1 仍会在任务 Main_Task 里执行。

6.3.2 判断 POU 是否被调用

如果一个 POU 没有在任何任务中调用，则编译后，该 POU 名称会显示为灰色 pou2 (PRG) ，当该 POU 被调用之后，则会显示为黑色 pou1 (PRG) 。用户可根据该方式直观的判断 POU 是否会在任务中执行。

6.3.3 POU 执行顺序

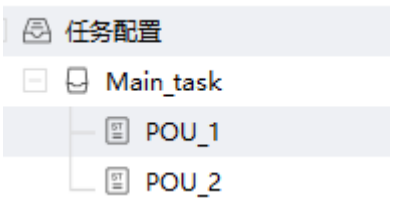
在任务中，POU 调用的顺序会影响 POU 的执行顺序，不同的执行顺序可能导致不同的结果。故在调用 POU 时，顺序也是用户需要注意的一个要点，下面举例进行简单说明：

1) 定义一个 Bool 型全局变量“ABC”，并新建两个 POU，分别命名为 POU1 和 POU2；

2) POU_1 中的代码为：ABC ：= False；

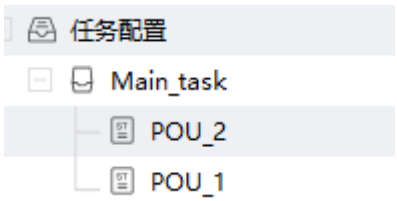
3) POU_2 中的代码为：ABC ：= True；

情况 1：任务配置为



该情况下，每周期结束时，变量 ABC 的值为 True；

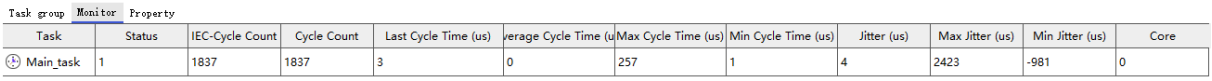
情况 2：任务配置为



该情况下，每周期结束时，变量 ABC 的值为 False。

6.4 任务执行周期监控

在线模式下，可以在“任务配置”里的“监视”选项卡对任务的状态、周期、抖动等数据进行监视。


Task group Monitor Property

Task	Status	IEC-Cycle Count	Cycle Count	Last Cycle Time (us)	verage Cycle Time (u	Max Cycle Time (us)	Min Cycle Time (us)	Jitter (us)	Max Jitter (us)	Min Jitter (us)	Core
Main_task	1	1837	1837	3	0	257	1	4	2423	-981	0

监视窗口的参数定义如下：

参数名	描述
任务	在任务配置中所定义的任务名
状态	分别有如下状态: 未创建:程序下载后一直未被建立，当使用事件触发任务可能会出现此状态。 创建:任务已经在实时系统中建立，但还未正式运行。 有效：任务正在被执行。 异常：任务出现异常。
IEC 运算执行计数	程序自开始运行至今的循环累积计数。
任务调度计数	已经运行的周期计数。取决于目标系统，它可以等于 IEC 循环计数，或者更大值，此时即使应用程序没运行，周期也一样被计数。
上周期执行时间（us）	上一个任务周期的执行时间。

平均执行时间（us）	任务平均所需的执行时间。
最大执行时间（us）	任务最大执行时间
最小执行时间（us）	任务最小执行时间
抖动（us）	上个周期测量到的抖动值
最大抖动（us）	测量得到的最大抖动时间
最小抖动（us）	测量得到的最小抖动时间
处理器核	当前任务在 CPU 中哪个核心运行，单核 CPU 显示为 0

通过分析如上各项时间的定义后，程序任务周期设置应遵循如下的时间设定关系，按照此设定方法可以更好的优化程序任务周期及看门狗时间，保障程序的稳定性和程序的实时性：

看门狗触发时间 > 固定周期循环时间 > 程序最大循环时间

循环时间比固定周期循环时间长的情况下，CPU 会检测出程序有超出计数，此时，会影响程序的实时性。如果程序循环时间比看门狗时间设定长的情况下，CPU 会检测出看门狗故障，会停止程序的执行。

7 扩展模块

7.1 扩展模块类型

SC2-C 系列 PLC 本体可直接扩展 R1 系列模块，单个 SC2-C 系列 PLC 最大可扩展 16 个 R1 系列模块。

表 6.1 R1 系列模块型号及描述

模块类型	型号	描述
数字量输入模块	SC-1600	16 路数字量输入，漏型（NPN）/源型（PNP）输入,DC24V 输入，弹簧式接插件
	SC-3200	32 路数字量输入，漏型（NPN）/源型（PNP）

		输入,DC24V 输入，弹簧式接插件
	SC-3200-1	32 路数字量输入，漏型（NPN）输入，DC24V 输入，MIL 接插件
数字量输出模块	SC-0016-N	16 路数字量输出，漏型（NPN）输出，弹簧式接插件
	SC-0016-P	16 路数字量输出，源型（PNP）输出，弹簧式接插件
	SC-0032-N	32 路数字量输出，漏型（NPN）输出，弹簧式接插件
	SC-0032-N-1	32 路数字量输出，漏型（NPN）输出，MIL 接插件
继电器输出模块	SC-0016-R	16 路数字量输出，继电器输出，弹簧式接插件
数字量输入输出模块	SC-0808-N	8 路数字量输入：漏型（NPN）/源型（PNP）输入,DC24V 输入，弹簧式接插件 8 路数字量输出：漏型（NPN）输出，弹簧式接插件
	SC-1616-N	16 路数字量输入：漏型（NPN）/源型（PNP）输入,DC24V 输入，弹簧式接插件 16 路数字量输出：漏型（NPN）输出，弹簧式接插件
	SC-1616-P	16 路数字量输入：漏型（NPN）/源型（PNP）输入,DC24V 输入，弹簧式接插件 16 路数字量输出：源型（PNP）输出，弹簧式接插件
模拟量输入模块	SC-A0400-IV	4 路模拟量输入，支持电流/电压输入，弹簧式接插件
模拟量输出模块	SC-A0004-IV	4 路模拟量输出，支持电流/电压输入，弹簧式接插件

7.2 扩展模块组态

SC2-C 系列 PLC 按照以下两种方式进行本体扩展模块组态：

方式一：手动添加

1) 打开“Lead Studio”编程软件，新建工程，选择编程语言。具体操作方法和第 2 章相同。

2) 添加所需扩展模块

① 在设备树栏中，鼠标左键双击“硬件配置”进入 PLC 本体模块配置界面；

② 在页面右侧硬件箱中，鼠标双击所需模块或单击后将模块拖动到配置页面即可完成本体模块增添，如图 6.2 所示；

③ 可在配置页面表可见已经选择增加的模块，如图 6.2 所示；

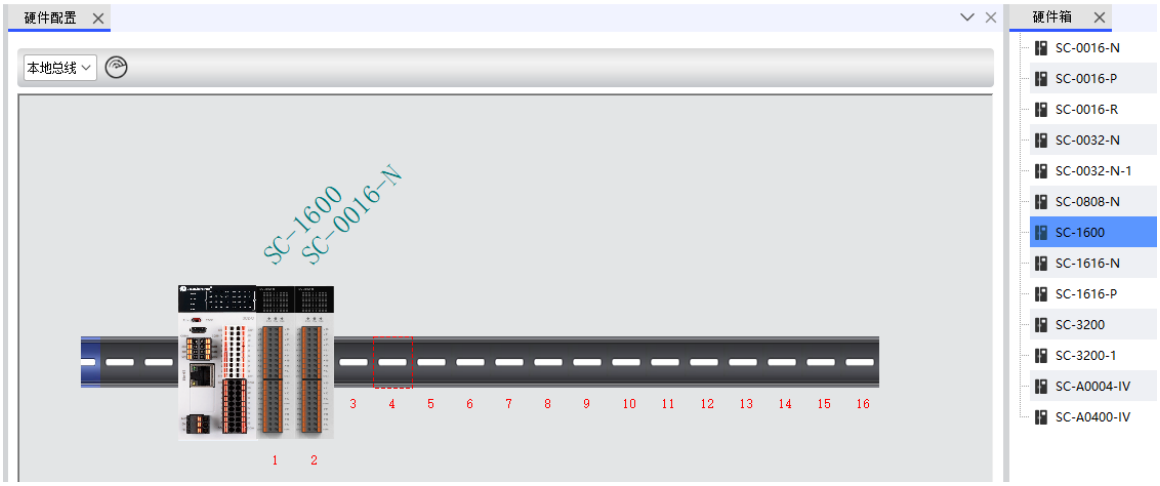


图 6.2　PLC 本体添加扩展模块

方式二：扫描添加

1）打开“LeadStudio”编程软件，新建工程，选择编程语言。

2）连接 PLC。具体操作方法和第 2 章相同。

3）扫描扩展模块

① 在设备树中，鼠标左键双击“硬件配置”进入 PLC 本体模块配置界面；

② 鼠标左键单击“扫描设备”，如图 6.3 所示

③ 鼠标左键单击“复制所有设备到工程中”，即可完成本体扩展模块的扫描，扫描结果

如图 6.4 所示；

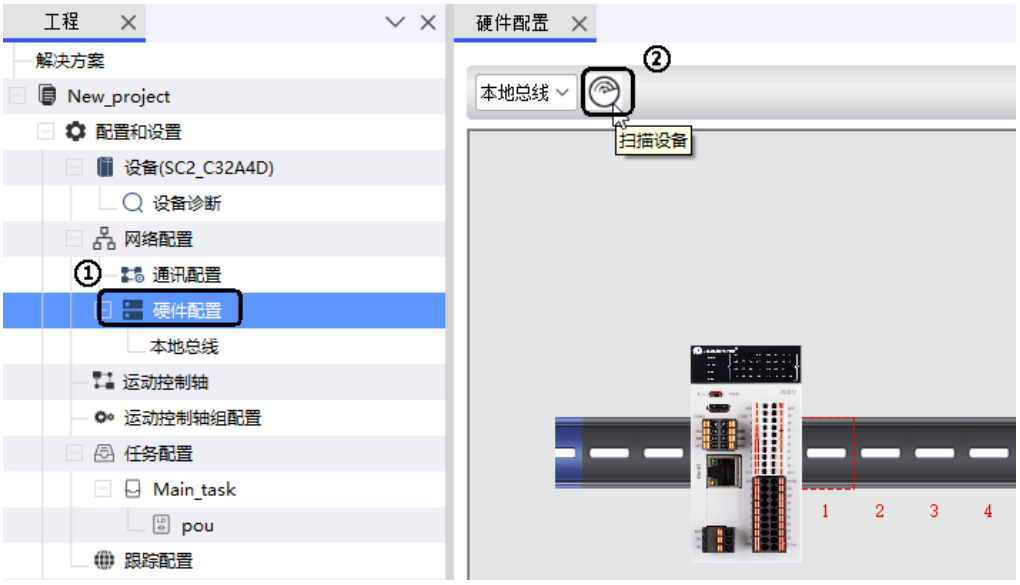


图 6.3　PLC 本体背板扫描

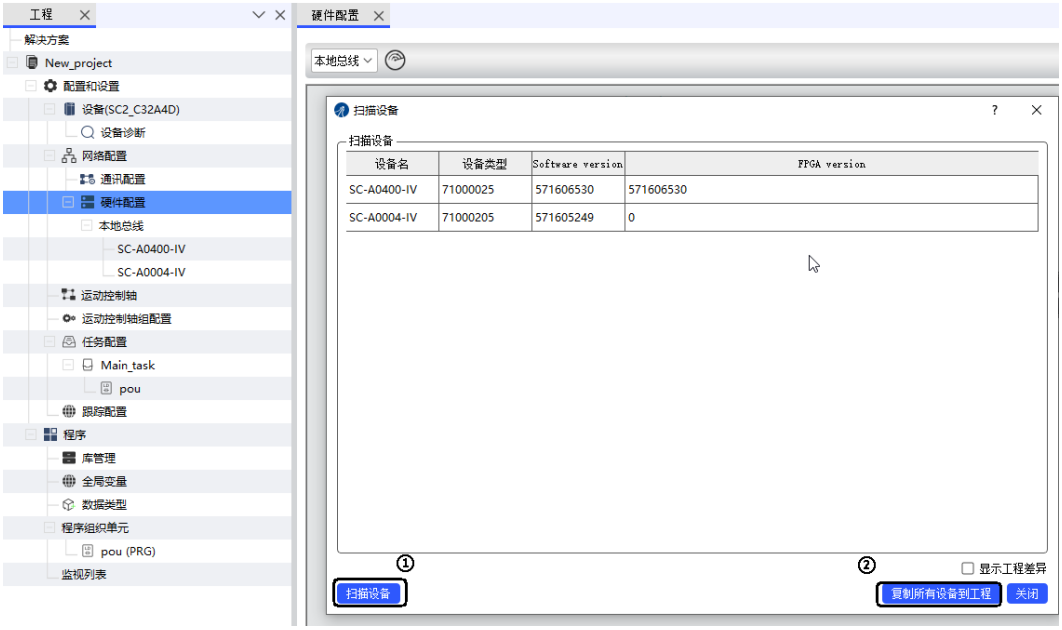


图 6.4　扫描结果

7.3 扩展模块的配置、映射

7.3.1 IO 模块的配置、映射

下面以 SC-0016-N 输出模块和 SC-1600 输入模块为例，介绍 IO 模块的配置、映射方法。（详细使用方法可参考“R1 系列经济型扩展模块用户手册”）

1) 添加所需要的模块，可参考 7.2 节，添加完成后如图 6.5 所示；

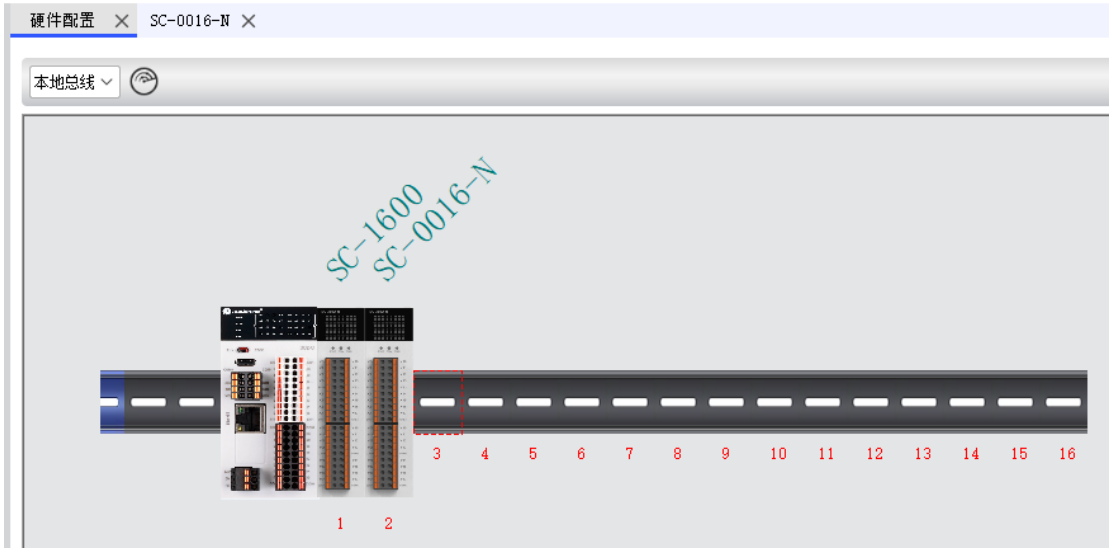


图 6.5 PLC 本体模块添加

2) 设置本地总线循环任务，在设备树中，鼠标左键双击“本地总线”进入 PLC 本地总线配置界面，选择本地总线循环任务，如图 6.6 所示；

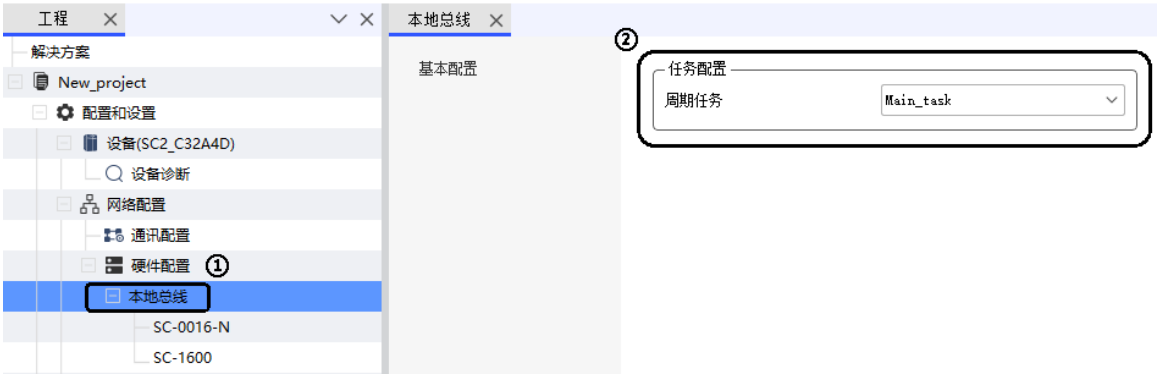


图 6.6 本地总线循环任务

3) 声明及建立输入、输出变量，与地址关联映射后，即可正常使用。

① 在设备树中，鼠标左键选择“POU(PRG)”进入编程页面新建变量，如图 6.7 所示；

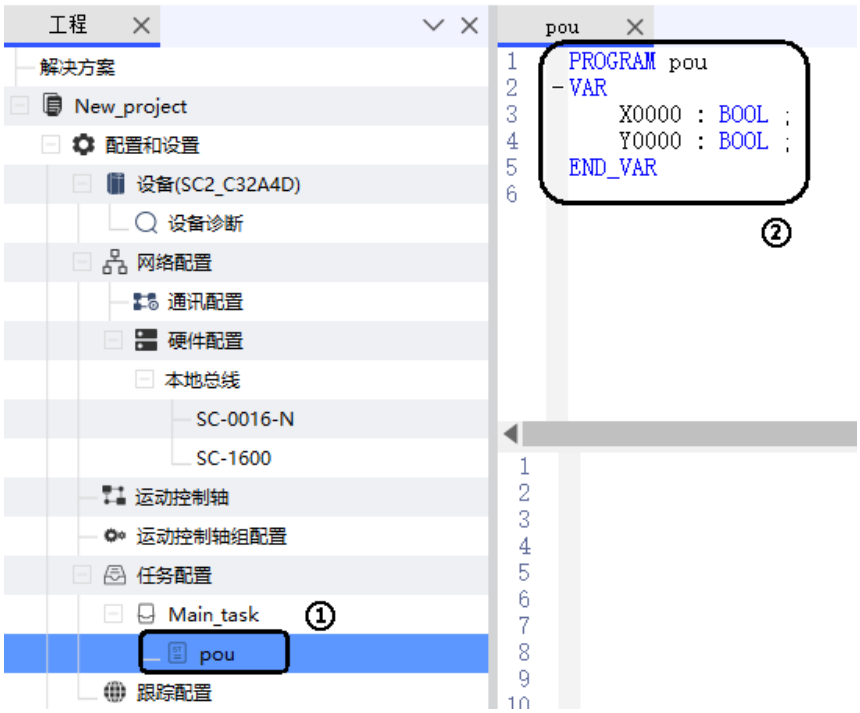


图 6.7 建立输入、输出变量 X000、Y000

② 鼠标左键双击“SC-0016-N”，选择“I/O 映射配置”，SC-0016-N 输出模块端口关联映射，如图 6.8 所示；

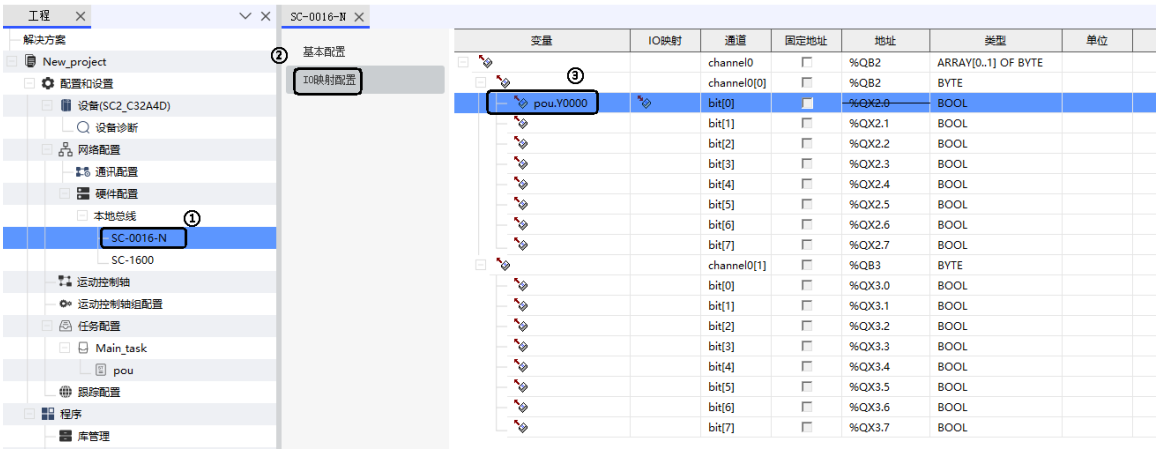


图 6.8 输出变量与输出口关联映射

③ 鼠标左键双击“SC-1600”，选择“I/O 映射配置”， SC-1600 输入模块端口关联映射，如图 6.9 所示。

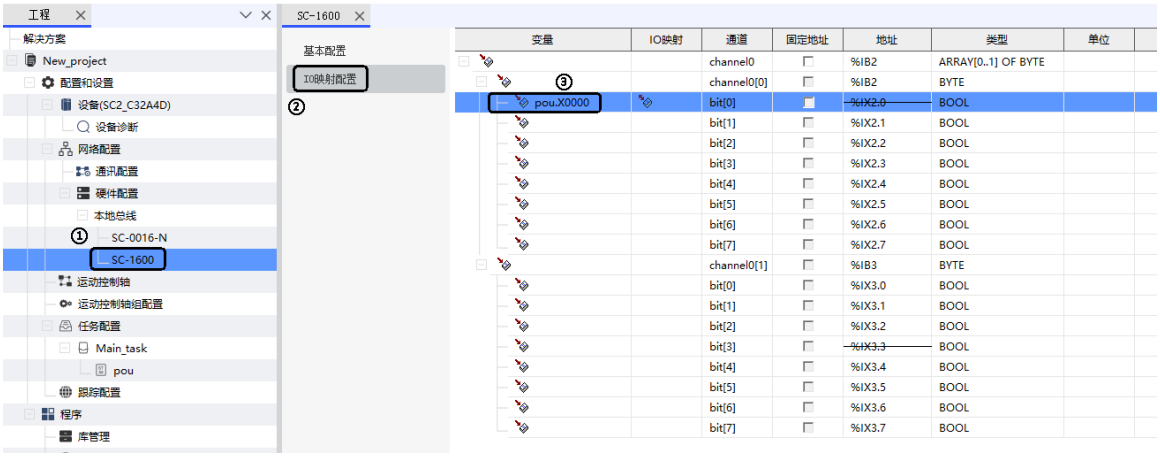


图 6.9 输入变量与输入口关联映射

7.3.2 模拟量模块的配置、映射

下面以 SC-A0400-IV 和 SC-A0004-IV 模块为例，介绍 AD、DA 模块的配置、映射方法。（详细使用方法可参考“R1 系列经济型扩展模块用户手册”）

1) 添加所需要的模块，可参考 7.2 节，添加完成后如图 6.10 所示；

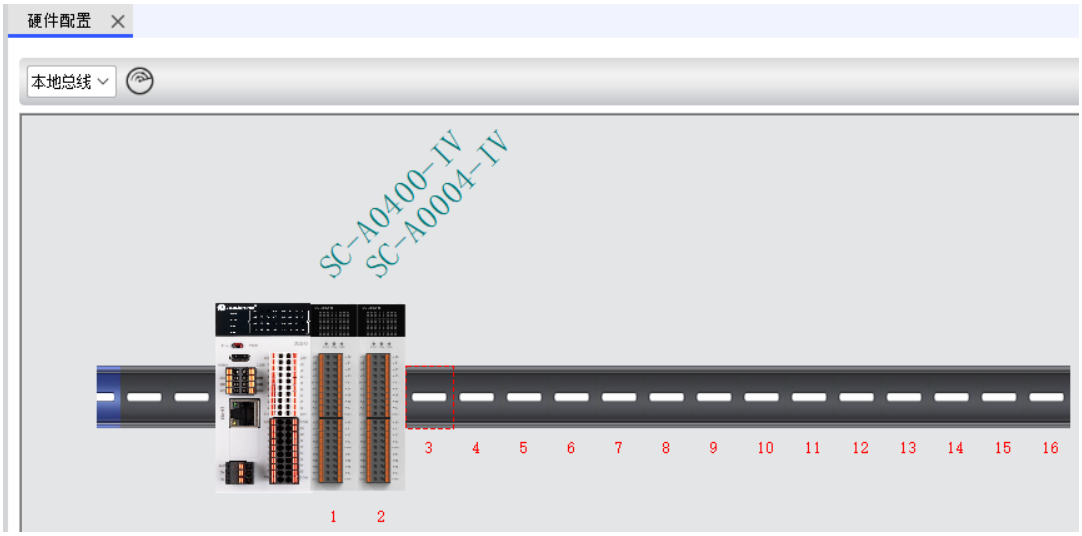


图 6.10　PLC 本体添加的 AD 和 DA 模块

2) 设置本地总线循环任务，在设备树中，鼠标左键双击“本地总线”进入 PLC 本地总线配置界面，选择本地总线循环任务，如图 6.11 所示

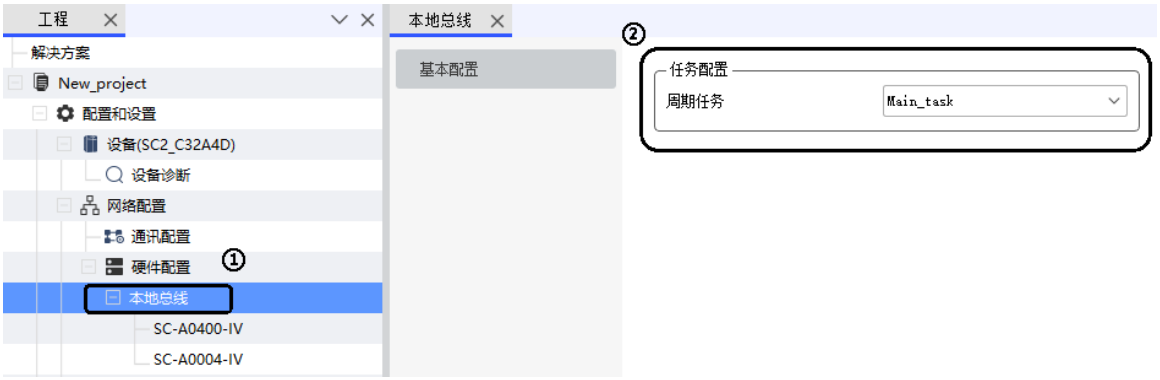


图 6.11 本地总线循环任务

3) 设置模块的参数

① 设置 SC-A0400-IV 模拟量输入模块的参数，左键双击设备树下“SC-A0400-IV”进入设置页面。

② 左键单击“参数配置”，进入模块参数配置界面，如图 6.12 所示。

③ 选择所需要的量程，本例程选择“-10V~10V”量程，如图 6.12 所示。

④ 通道滤波参数设置，根据实际需要选择滤波参数，如图 6.12 所示。

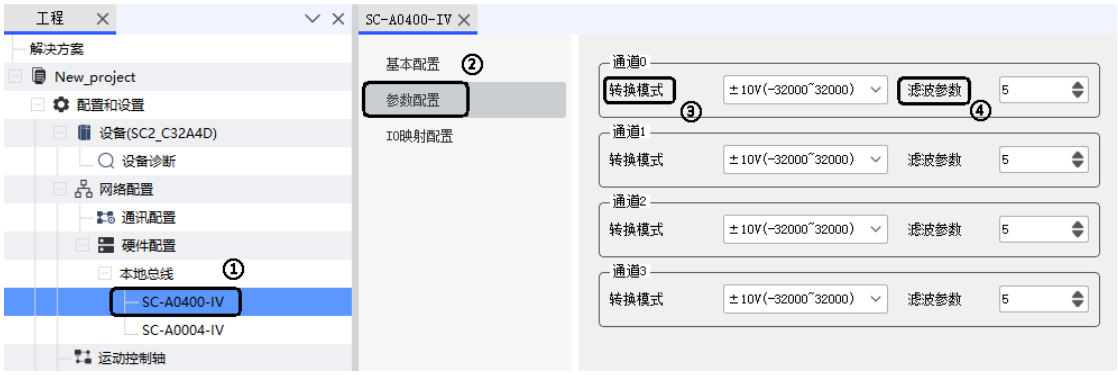


图 6.12 模拟量输入模块参数配置

⑤ 设置 SC-A0004-IV 模拟量输出模块的参数，左键双击设备树下“SC-A0004-IV”进入设置页面。

⑥ 左键单击“参数配置”，进入模块参数配置界面，如图 6.13 所示。

⑦ 设置通道使能，勾选“通道 1”则为通道 1 进行使能（必须勾选，否则无法使用），如图 6.13 所示。

⑧ 选择所需要的量程，本例程选择“-10V~10V”量程，如图 6.13 所示。

⑨ 根据实际需要设置断线后输出状态，如图 6.13 所示。

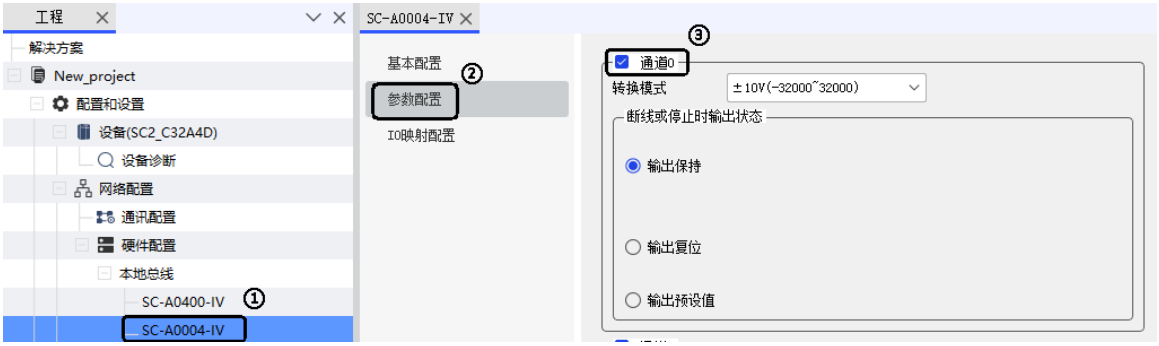


图 6.13 模拟量输出模块参数配置

4) 变量映射过程如图 6.14 及图 6.15 所示。

① 鼠标左键双击“SC-A0400-IV”及“SC-A0004-IV”；

② 鼠标左键单击“I/O 映射配置”进入变量映射界面；

③ 绑定变量 DA_D0 为模拟量输出模块通道 0 的输出变量；

④ 绑定变量 AD_D0 为模拟量输入模块通道 0 的输入变量。

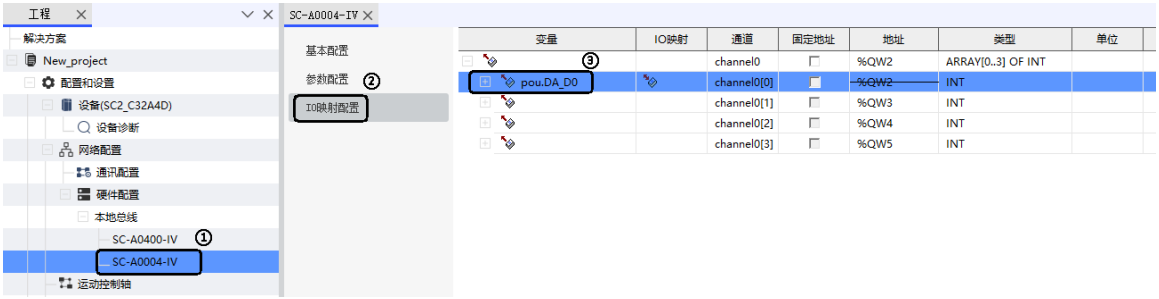


图 6.14 模拟量输出模块映射

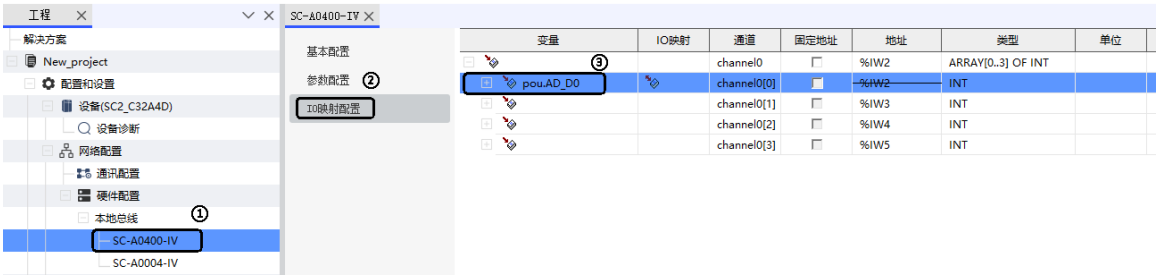


图 6.15 模拟量输入模块映射

与相关变量进行映射后，即可在程序中通过变量读取模拟量输入值、设置模拟量输出值。

8 串口通讯

8.1 基于 RS485 接口的 Modbus RTU 主从站通信

8.1.1 ModbusRTU 主从站通讯基础知识

本节介绍 PLC 基于 RS485 接口的 Modbus RTU 主从站通信。 通常以一个 PLC 为主站，变频器、 伺服电机驱动器和其他 PLC 等设备为从站。

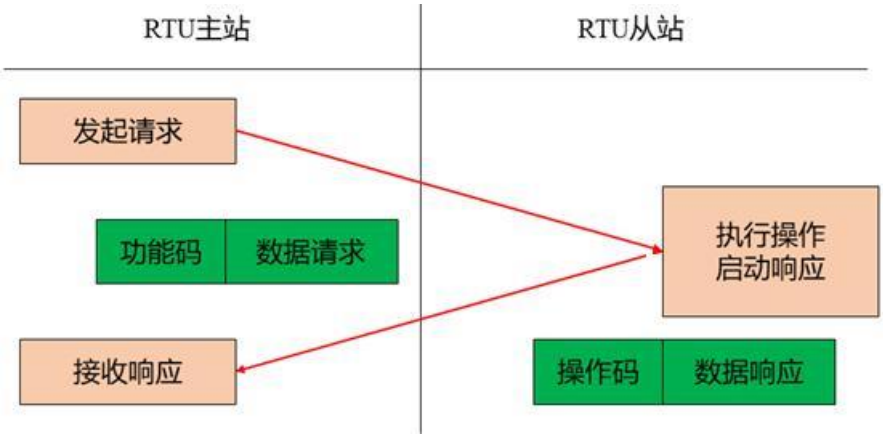
RTU（Remote Terminal Unit） 为远程终端单元。

Modbus RTU 主从站通讯过程如下图所示。 Modbus 主站的特点有：

1) 可以主动发出指令；

2) 具有唯一性， 通讯总线上只能有一个主站；

3) 可以对接多个 Modbus 从站。



图：Modbus RTU 主从站通讯过程

8.1.2 Modbus 通讯协议简介

Modbus 协议是一个主/从（master/slave） 架构的协议。 有一个节点是 master 节点，其他使用 Modbus 协议参与通信的节点是 slave 节点。 每一个 slave 设备都有一个唯一的地址。在通讯网络中，只有被指定为主节点的节点可以启动一个命令。

一个 Modbus 命令包含一个将执行该命令的设备的 Modbus 地址。 所有设备都会收到该命令，但只有指定地址的设备会执行该命令并回应指令。

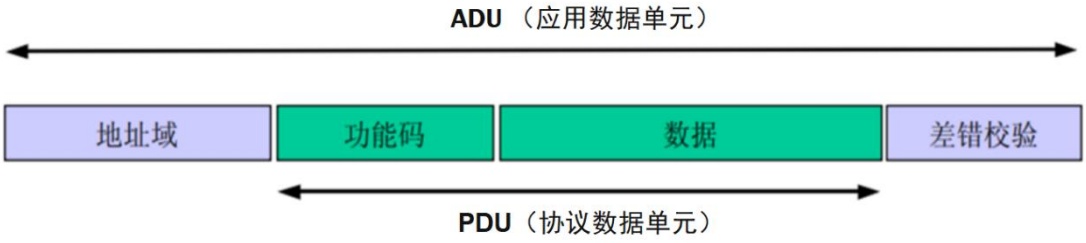
所有的 Modbus 命令包含了检查码，以确定到达的命令没有被破坏。基本的

Modbus 命令能指定一个 RTU 改变其寄存器的值，控制或者读取一个 I/O 端口， 以及指挥设备回送一个或者多个其寄存器中的数据。

Modbus 协议中按照通信访问的数据宽度划分数据类型，主要有 Bit 型和 Word 型两种数据。

依照行业惯例，本文中有时将 Bit 型变量称为“线圈” 或“触点” ，将 Word 型变量则称为“寄存器” 。

Modbus 协议通信数据帧格式，如下图所示：



图：Modbus 通信帧格式

例如：主站要读地址为 1 的从机中的 2 个寄存器的值，两个寄存器的地址为 0004、0005。该指令的功能码为 03。报文交互过程如下：

主站的请求命令的通讯帧内容为：01 03 00 04 00 02 85 ca，详见下表：

表：主站发出的通讯帧

从机地址	功能码	从站寄存器起始地址	读取寄存器个数	CRC 校验
01	03	00 04	00 02	85 ca

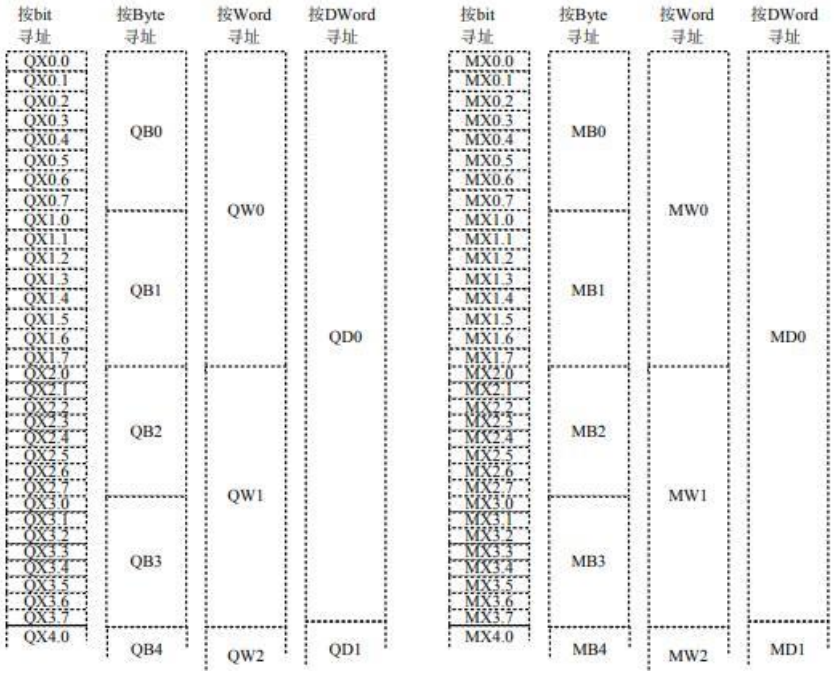
从站响应该请求，返回相应数据。该动作的功能码为 03。从站发出的通讯帧内容为：01 03 04 00 00 00 00 21 33，详见下表：

表：从站返回的通讯帧

从机地址	功能码	返回字节个数	寄存器 0005 数据	寄存器 0006 数据	CRC 校验
01	03	04	00 00	00 00	21 33

8.1.3 Modbus 协议可访问的内部地址

SC2/SC2U 系列 PLC 的变量区有 Q 区、I 区和 M 区这三种，分别都可以按位、按字节、按字和按双字进行访问。其中 Q 区、I 区及 M 区的内存大小均为为 16384 个字节(由于 Modbus 协议最大只支持 65536 个线圈，所以通过 Modbus 协议只能与前 8192 个字节的内部寄存器地址通讯)。寄存器地址索引规则如下图所示。



图：寄存器地址索引规则

为了方便地址的计算，Word 型寄存器的起始地址，遵循的是起始地址为偶数 Byte 地址；DWord 型寄存器的起始地址，遵循的是起始地址为偶数 Word 地址对齐，其索引号为 2 倍关系的原则。

I 区、Q 区、M 区地址编址规则相同。下表为 Q 区、M 区及 I 区的编址。

表：M/Q/I 区直接地址寻址

M 区直接地址寻址规则			
按 BIT 寻址	**按 Byte 寻址**	**按 Word 寻址**	**按 DWORD 寻址**
%MX0.0~%MX0.7	%MB0	%MW0	%MD0
%MX1.0~%MX1.7	%MB1		
%MX2.0~%MX2.7	%MB2	%MW1	

%MX3.0~%MX3.7	%MB3		
…	…	…	…
%MX8188.0~%MX8188.7	%MB8188	%MW4094	%MD2047
%MX8189.0~%MX8189.7	%MB8189		
%MX8190.0~%MX8190.7	%MB8190	%MW4095	
%MX8191.0~%MX8191.7	%MB8191		
Q 区直接地址寻址规则			
按 BIT 寻址	**按 Byte 寻址**	**按 Word 寻址**	**按 DWORD 寻址**
%QX0.0~%QX0.7	%QB0	%QW0	%QD0
%QX1.0~%QX1.7	%QB1		
%QX2.0~%QX2.7	%QB2	%QW1	
%QX3.0~%QX3.7	%QB3		
…	…	…	…
%QX8188.0~%QX8188.7	%QB8188	%QW4094	%QD2047
%QX8189.0~%QX8189.7	%QB8189		
%QX8190.0~%QX8190.7	%QB8190	%QW4095	
%QX8191.0~%QX8191.7	%QB8191		
I 区直接地址寻址规则			
按 BIT 寻址	**按 Byte 寻址**	**按 Word 寻址**	**按 DWORD 寻址**
%IX0.7~%IX0.0	%IB0	%IW0	%ID0
%IX1.7~%IX1.0	%IB1		
%IX2.7~%IX2.0	%IB2	%IW1	
%IX3.7~%IX3.0	%IB3		

…	…	…	…
%IX8188.0~%IX8188.7	%IB8188	%IW4094	%ID2047
%IX8189.0~%IX8189.7	%IB8189		
%IX8190.0~%IX8190.7	%IB8190	%IW4095	
%IX8191.0~%IX8191.7	%IB8191		

标准的 Modbus 协议都可以访问的 PLC 内部 I、Q 区范围如下表所示：

表：PLC 内部 I、Q 区范围

地址范围	功能码	起始地址	线圈数量	说明
QW0～QW4095 (QX0.0～QX8191.7)	0x01,0x05, 0x0f	0	65536	通用标准 Modbus 协议都可以访问
IW0～IW4095 (IX0.0～IX8191.7)	0x02,0x04	0	65536	通用标准 Modbus 协议都可以访问

M 区范围如下表所示：

表：PLC 内部 M 区范围

地址范围	功能码	起始地址	线圈数量	说明
MW0～MW65535	0x03,0x06, 0x10	0	65536	通用标准 Modbus 协议都可以访问

8.1.4 Modbus 通信功能码

SC2 支持 8 种 Modbus 常用功能码，可以分为位操作和字操作，以下表为 8 种 Modbus

协议常用功能码的简单介绍。

表：Modbus 协议常用功能码说明

功能码	描述	位/字操作	操作数量
01H	读线圈寄存器	位操作	单个或多个
02H	读离散输入状态寄存器	位操作	单个或多个
03H	读保持寄存器	字操作	单个或多个
04H	读输入寄存器	字操作	单个或多个
05H	写单个线圈寄存器	位操作	单个
06H	写单个保持寄存器	字操作	单个
0fH	写多个线圈寄存器	位操作	多个
10H	写多个保持寄存器	字操作	多个

1) 功能码 0x01，读线圈寄存器，可以读取 Q 区变量。

请求帧格式：从机地址+0x01+线圈起始地址+线圈数量+CRC 较验，如下表所示：

表：功能码 0x01 请求帧详解

序号	描述	位/字操作	操作数量
1	从机地址	1 个字节	取值 1-247
2	0x01	1 个字节	读线圈
3	线圈起始地址	2 个字节	高位在前，低位在后，见线圈编址
4	线圈数量 N	2 个字节	高位在前，低位在后
5	CRC 校验	2 个字节	高位在前，低位在后

响应帧格式：从机地址+0x01+字节数+线圈状态+CRC 校验，如下表所示：

表：功能码 0x01 响应帧详解

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

2) 功能码 0x02，读离散输入状态寄存器，可以读取 I 区变量。

请求帧格式：从机地址+0x02+输入线圈起始地址+线圈数量+CRC 较验，如下表所示：

表：功能码 0x02 请求帧详解

序号	描述	位/字操作	操作数量
1	从机地址	1 个字节	取值 1-247
2	0x02	1 个字节	读线圈
3	线圈起始地址	2 个字节	高位在前，低位在后，见线圈编址
4	线圈数量 N	2 个字节	高位在前，低位在后
5	CRC 校验	2 个字节	高位在前，低位在后

响应帧格式：从机地址+0x02+字节数+输入线圈状态+CRC 校验，如下表所示：

表：功能码 0x02 响应帧详解

序号	描述	位/字操作	操作数量
1	从机地址	1 个字节	取值 1-247
2	0x02	1 个字节	读线圈

3	字节数	1 个字节	值：[(N+7)/8]
4	线圈状态	[(N+7)/8] 个字节	每 8 个线圈合为一个字节，最后一个若不足 8 位，未定义部分填 0。前 8 个线圈在第一个字节，地址最小的线圈在最低位。依次类推
5	CRC 校验	2 个字节	高位在前，低位在后

3) 功能码 0x03，读保持寄存器， 可以读取 M 区变量。

请求帧格式：从机地址+0x03+寄存器起始地址+寄存器数量+CRC 较验，如下表所示：

表：功能码 0x03 请求帧详解

序号	描述	位/字操作	操作数量
1	从机地址	1 个字节	取值 1-247
2	0x03	1 个字节	读寄存器
3	寄存器起始地址	2 个字节	高位在前，低位在后，见寄存器编址
4	寄存器数量	2 个字节	高位在前，低位在后，N
5	CRC 校验	2 个字节	高位在前，低位在后

响应帧格式：从机地址+0x03+字节数+寄存器值+CRC 校验，如下表所示：

表：功能码 0x03 响应帧详解

序号	描述	位/字操作	操作数量
1	从机地址	1 个字节	取值 1-247
2	0x03	1 个字节	读寄存器
3	字节数	1 个字节	值：N*2
4	寄存器值	N*2 个字节	每两个字节表示一个寄存器值，高

			位在前低位在后。寄存器地址小的排在前面
5	CRC 校验	2 个字节	高位在前，低位在后

4) 功能码 0x04，读输入寄存器， 可以读取 I 区变量。

请求帧格式：从机地址+0x04+输入寄存器起始地址+寄存器数量+CRC 较验，如下表所示：

表：功能码 0x04 请求帧详解

序号	描述	位/字操作	操作数量
1	从机地址	1 个字节	取值 1-247
2	0x04	1 个字节	读输入寄存器
3	输入寄存器起始地址	2 个字节	高位在前，低位在后，见寄存器编址
4	寄存器数量	2 个字节	高位在前，低位在后，N
5	CRC 校验	2 个字节	高位在前，低位在后

响应帧格式：从机地址+0x04+字节数+寄存器值+CRC 校验，如下表所示：

表：功能码 0x04 响应帧详解

序号	描述	位/字操作	操作数量
1	从机地址	1 个字节	取值 1-247
2	0x04	1 个字节	读输入寄存器
3	字节数	1 个字节	值：N*2
4	寄存器值	N*2 个字节	每两个字节表示一个寄存器值，高位在前低位在后。寄存器地址小的排在前面

5	CRC 校验	2 个字节	高位在前，低位在后

5) 功能码 0x05，写单个线圈，可以写 Q 区变量。

请求帧格式：从机地址+0x05+线圈地址+线圈状态+CRC 较验，如下表所示：

表：功能码 0x05 请求帧详解

序号	描述	位/字操作	操作数量
1	从机地址	1 个字节	取值 1-247
2	0x05	1 个字节	写单线圈
3	线圈地址	2 个字节	高位在前，低位在后，见线圈编址
4	线圈状态	2 个字节	低位在前，高位在后。非 0 即为有效
5	CRC 校验	2 个字节	高位在前，低位在后

响应帧格式：从机地址+0x05+线圈地址+线圈状态+CRC 校验，如下表所示：

表：功能码 0x05 响应帧详解

序号	描述	位/字操作	操作数量
1	从机地址	1 个字节	取值 1-247
2	0x05	1 个字节	写单线圈
3	线圈地址	2 个字节	高位在前，低位在后，见线圈编址
4	线圈状态	2 个字节	低位在前，高位在后。非 0 即为有效
5	CRC 校验	2 个字节	高位在前，低位在后

6) 功能码 0x06，写单个寄存器，可以写 M 区变量。

请求帧格式：从机地址+0x06+寄存器地址+寄存器值+CRC 较验，如下表所示：

表：功能码 0x06 请求帧详解

序号	描述	位/字操作	操作数量
1	从机地址	1 个字节	取值 1-247
2	0x06	1 个字节	写单寄存器
3	寄存器地址	2 个字节	高位在前，低位在后，见寄存器编址
4	寄存器值	2 个字节	高位在前，低位在后。非 0 即为有效
5	CRC 校验	2 个字节	高位在前，低位在后

响应帧格式：从机地址+0x06+寄存器地址+寄存器值+CRC 较验，如下表所示：

表：功能码 0x06 响应帧详解

序号	描述	位/字操作	操作数量
1	从机地址	1 个字节	取值 1-247
2	0x06	1 个字节	写单寄存器
3	寄存器地址	2 个字节	高位在前，低位在后，见寄存器编址
4	寄存器值	2 个字节	高位在前，低位在后。非 0 即为有效
5	CRC 校验	2 个字节	高位在前，低位在后

7) 功能码 0x0f(15)，写多个线圈，可以写连续的多个 Q 区变量。

请求帧格式：从机地址+0x0f+线圈起始地址+线圈数量+字节数+线圈状态+CRC 检验，如下表所示：

表：功能码 0x0f 请求帧详解

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

响应帧格式：从机地址+0x0f+线圈起始地址+线圈数量+CRC 检验，如下表所示：

表：功能码 0x0f 请求帧详解

序号	描述	位/字操作	操作数量
1	从机地址	1 个字节	取值 1-247
2	0x0f	1 个字节	写个多线圈
3	线圈起始地址	2 个字节	高位在前，低位在后，见线圈编址
4	线圈数量	2 个字节	高位在前，低位在后。
5	CRC 校验	2 个字节	高位在前，低位在后。

8) 功能码 0x10(16)，写多个保持寄存器，可以写连续的多个 M 区变量。

请求帧格式：从机地址+0x10+寄存器起始地址+寄存器数量+字节数+寄存器值+CRC 检验，如下表所示：

表：功能码 0x10 请求帧详解

序号	描述	位/字操作	操作数量
1	从机地址	1 个字节	取值 1-247
2	0x10	1 个字节	写多个寄存器
3	寄存器起始地址	2 个字节	高位在前，低位在后，见寄存器编址
4	寄存器数量	2 个字节	高位在前，低位在后。数量为 N
5	字节数	1 个字节	值：N*2
6	寄存器值	N*2（N*4）	
7	CRC 校验	2 个字节	高位在前，低位在后

响应帧格式：从机地址+0x10+寄存器起始地址+寄存器数量+CRC 检验，如下表所示：

表：功能码 0x0f 请求帧详解

序号	描述	位/字操作	操作数量
1	从机地址	1 个字节	取值 1-247
2	0x10	1 个字节	写多个寄存器
3	寄存器起始地址	2 个字节	高位在前，低位在后，见寄存器编址
4	寄存器数量	2 个字节	高位在前，低位在后，N
5	CRC 校验	2 个字节	高位在前，低位在后

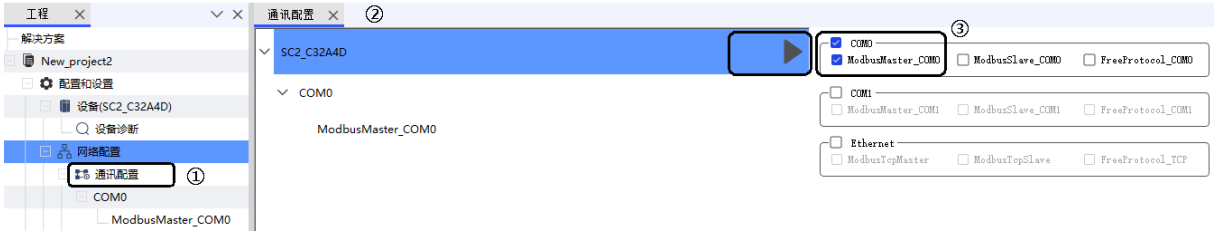
若想详细了解 Modbus 协议可参见《Modbus 协议规范》。

8.2 RS485 Modbus RTU 主站通讯

8.2.1 SC2 系列 PLC 的 RTU 主站配置

1) 使能 PLC 作为 Modbus RTU 主站

如下图所示，使能 PLC 作为 Modbus RTU 主站。



图：使能 PLC 作为 Modbus RTU 主站

① 左键双击网络配置下的“网络组态”；

② 左键单击 SC2-32A4D 旁的 “右箭头按钮”；

③ 左键单击“COM1”及“COM1 主站”，做为基于 RS485 端口的 Modbus RTU 主站使用；

2) 添加 Modbus RTU 从站设备

如下图，点击“ModbusMaster_COM0”，然后在“工具箱”里左键双击 “Modbus Slave”添加 Modbus RTU 从站

图：添加从站设备

3) Modbus RTU 主站配置

设置通信参数，如下图所示：

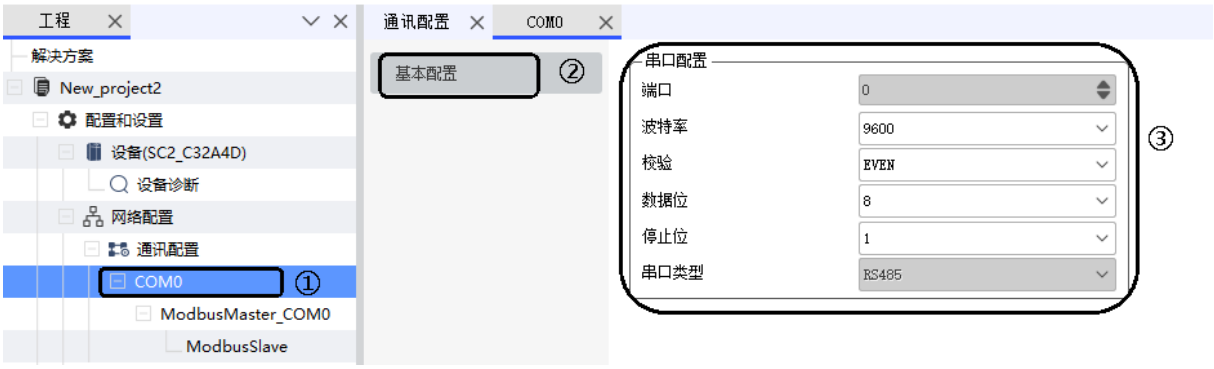


图 7.6　主站配置

① 双击“通讯配置”中的“COM0”进入串口配置界面；

② 在“基本配置”页面设置参数；

③ Modbus 主站配置相关参数与配置示例，如表 7.23、表 7.24 所示；

注意：Modbus 主从站通信参数配置必须一致，才能正常通信；

表：Modbus 主站配置相关参数

配置项	功能
波特率	通信时的速率，支持 4800-115200
奇偶校验	通信帧的校验方式，与停止位匹配使用
数据位	RTU 模式下为 8， ASCII 模式下为 7
停止位	奇/偶校验时为 1，无校验时为 2
传输模式	RTU

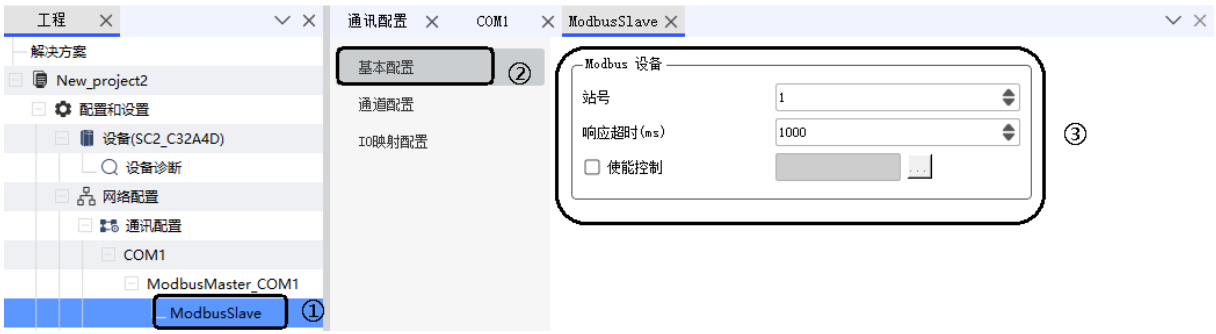
表：配置示例

配置项	配置值
波特率	115200
奇偶校验	EVEN（偶校验）
数据位	8
停止位	1

4) Modbus RTU 从站设备配置

① 基本设置

双击设备树中的主站设备“ModbusSlave”，在“基本配置”页面里设置从站的基本配置，如下图所示：



Modbus 从站配置

Modbus 从站配置相关参数，如下所示：

表：Modbus 从站配置相关参数

配置项	功能
从站站号	标识从站号，范围 1～247
超时时间	主站发帧后，超过该时间从站未响应，主站报接收超时

② Modbus 从站通讯设置

Modbus 从站设置完相关参数后，需要配置从站设备通讯数据的功能码、触发方式、读配置、写配置等参数，操作步骤如下图所示：

Modbus 从站通讯设置

Modbus 从站通讯设置

Modbus RTU 主站通讯设置相关参数以及功能码详细说明，如下表所示：

表：ModbusMaster_COM1 通讯设置相关参数

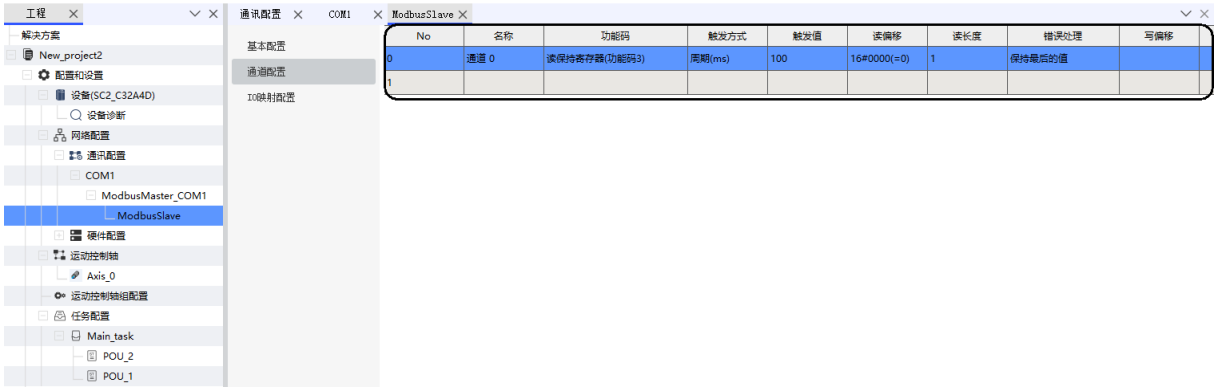
序号	配置项	功能
1	功能码	读线圈状态（功能码 01） 读离散寄存器（功能码 02） 读保持寄存器（功能码 03） 读输入寄存器（功能码 04） 写单个线圈（功能码 05） 写单个寄存器（功能码 06） 写多个线圈（功能码 15） 写多个寄存器（功能码 16）
2	触发方式	循环执行：周期触发的请求。
		电平触发：编程进行改变触发变量电平时触发请求。（触发变量在 IO 映射中设置）
3	重发次数	本次发生通信故障未获得从站返回帧， 则按重发次数进行重新发送。

4	注释	可以对数据进行描述的简短文本区域。
5	读/写偏移	读\写的寄存器位置开始地址。
6	读/写长度	读取寄存器的个数
7	错误处理	Keep the recent value：使数据保持最后一次的有效值。 Set Zero:数据设置成 0。

表：功能码详细说明

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

按照参数配置要求设置通讯参数，配置完成后如下图所示：



从站设备通讯数据

5) Modbus RTU 常见故障与错误码

Modbus RTU 主站连接从站时发生的主要故障如下表：

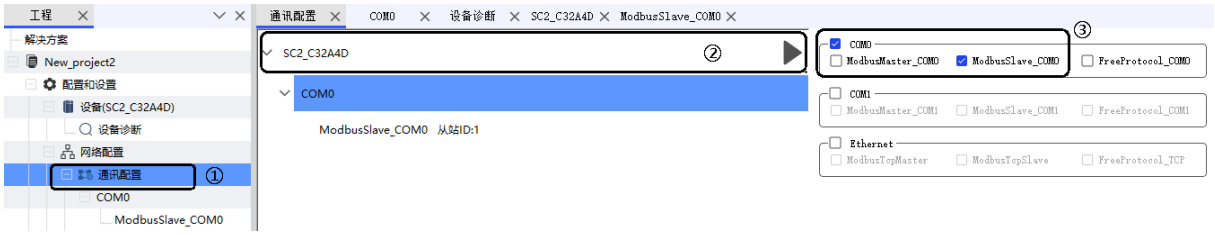
故障码	说明	故障码	说明
0	无错误	8008	存储奇偶性差错
9	设备掉线	8010	不可用网关路径
110	连接超时	8011	网关目标设备响应失败
8001	非法功能码	8012	CRC 错误
8002	非法数据地址	8013	服务器（或从站）响应数据异常
8003	非法数据值	8014	服务器（或从站）异常响应
8004	从站设备故障	8015	无效功能码
8005	从机应答超时	8016	读/写线圈、寄存器数据过多
8006	从站设备忙	8017	服务器（或从站）响应的站号与请求不一致
8007	否认应答		

8.3 RS485 Modbus RTU 从站通讯

8.3.1 SC2 系列 PLC Modbus RTU 从站的配置

1) 使能 PLC 作为 Modbus RTU 从站

如下图，设置 SC2 系列 PLC 为 Modbus RTU 从站。



图：使能 PLC 作为 Modbus RTU 从站

① 左键双击网络配置下的“网络组态”；

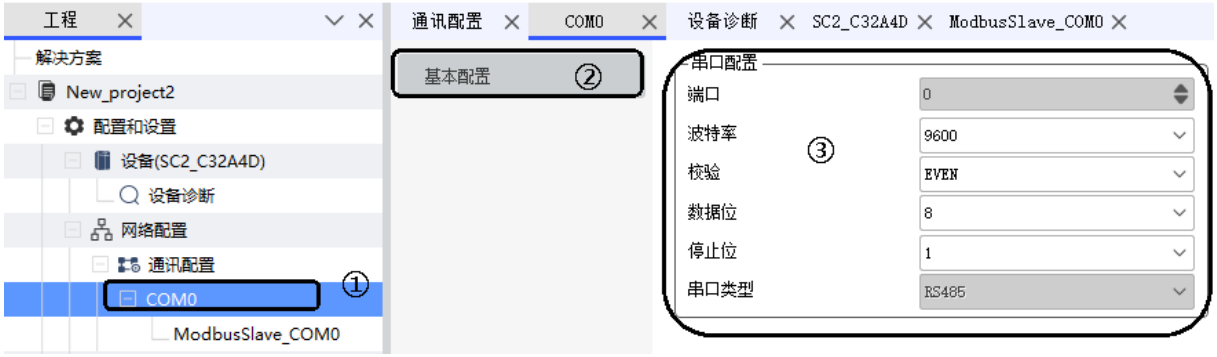
② 左键单击 SC2-C32A4D 旁的 “右箭头按钮”；

③ 左键单击“COM0”及“COM0 从站”，做为基于 RS485 端口的 Modbus RTU 从站使用。

2) Modbus RTU 从站配置

双击设备树中的“COM0”，在“基本配置”的“串口配置”里设置从站相关通讯参数，如下图所示。

Modbus 从站配置相关参数，请参见 7.2.1 小节内表：Modbus 从站配置相关参数。



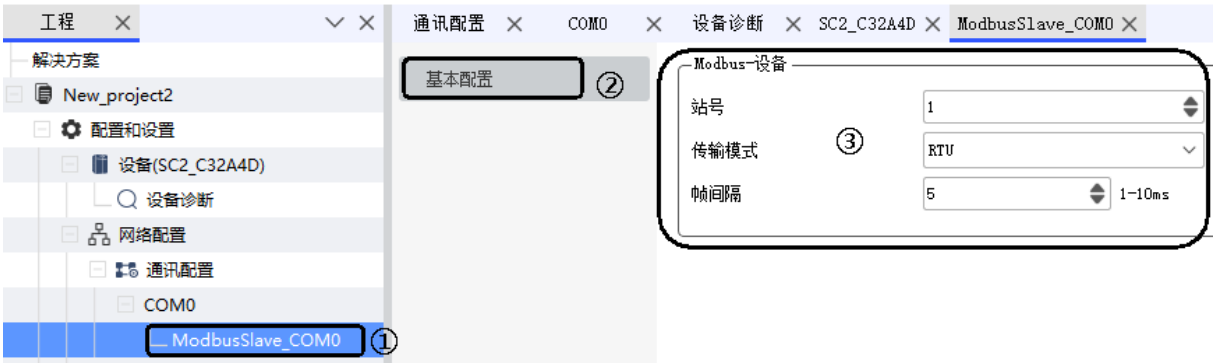
图：Modbus RTU 从站通讯参数设置

3) 从站站号设置

① 左键双击 “ModbusSlave COM0”；

② 在 “基本配置”页面中设置参数；

③ 在“Modbus 设备”中设置站号为 1。



图：Modbus RTU 从站站号设置

8.4 RS485 Modbus RTU 通讯例程

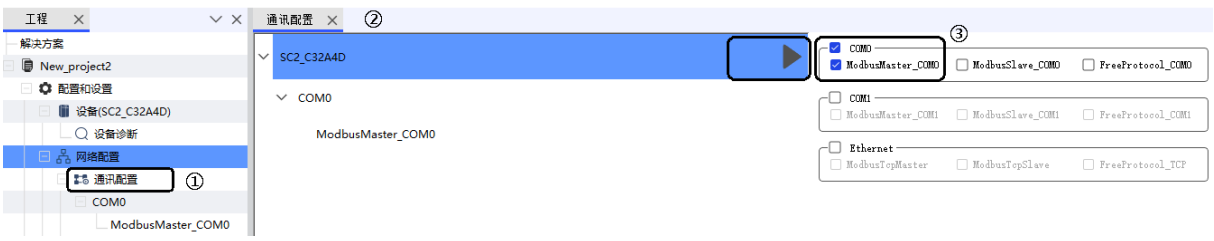
本例程采用两个 SC2 系列 PLC 分别作为 Modbus RTU 的主站及从站，通过读、写操作线圈和寄存器的方式，模拟传输 PLC 状态、读取当前坐标等数据，简单演示基于 RS485 端口的 Modbus RTU 通讯。

1) 硬件端口,如下图所示。分别将两台 SC2 系列 PLC 的 485+、485-及 GND 对接即可。

图：SC2 系列 PLC 的 RS485 端口

2) PLC 主站通讯配置

a) SC2 系列 PLC 主站设备通讯配置，如下图所示：



图：使能 PLC 作为 Modbus RTU 主站

① 左键双击网络配置下的“网络组态”；

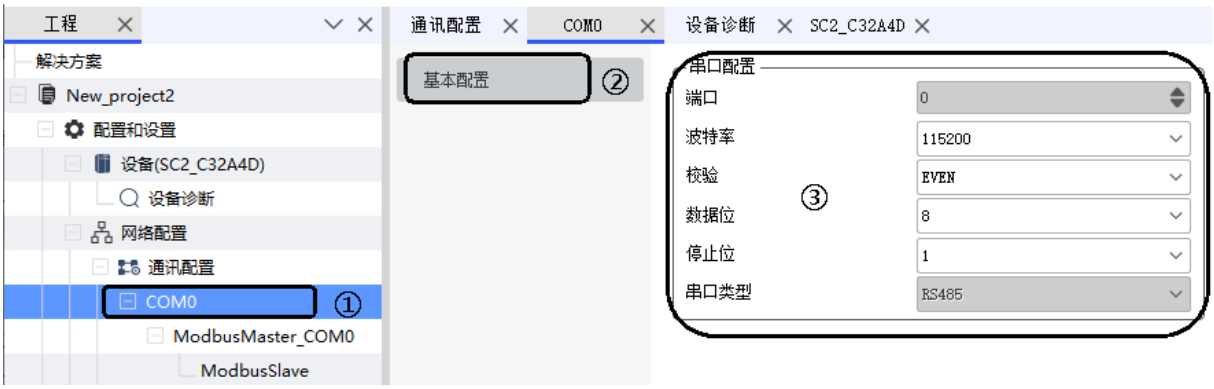
② 左键单击 SC2-C32A4D 旁的 “右箭头按钮”；

③ 左键单击“COM0”及“COM0 主站”，做为基于 RS485 端口的 Modbus RTU 主站使用；

b) 如下图，在“工具箱”里左键双击“Modbus RTU Slave”添加 Modbus RTU 从站设置。

图：添加从站设备

c) Modbus RTU 通信参数配置，如下图所示：



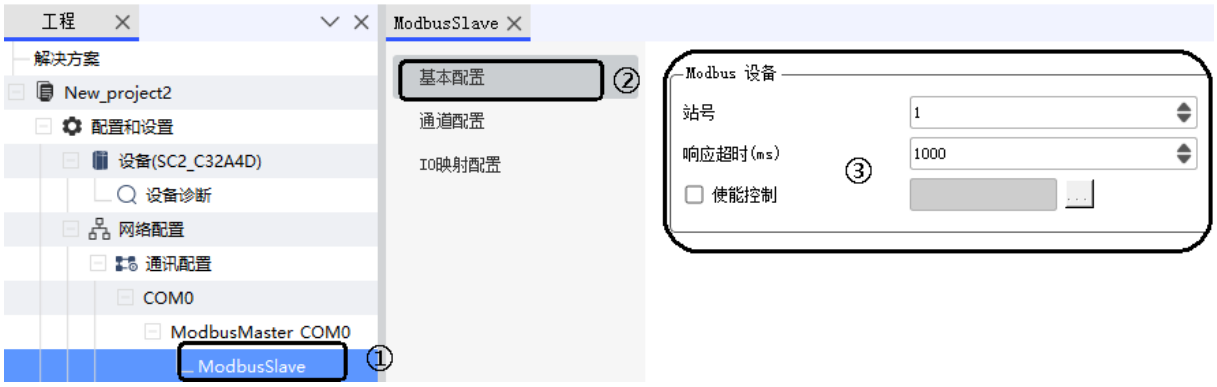
图：通信参数配置

① 左键双击 “COM0”；

② 在 “基本配置”页面中设置参数；

③ 在“串口配置”中设置成以下通信参数，“波特率：115200”，“校验：EVEN”，“数据位：8”，“停止位：1”；

d) 从站基本配置，如下图所示：



图：基本配置

① 左键双击“Modbus Slave”；

② 单击“基本配置”页面中设置参数；

③ “从站 ID：1”，“响应时间：1000”。

e) Modbus 通信功能码，如下图所示：

图：添加通信功能码

图：功能码设置

① 左键双击“Modbus RTU Slave0”；

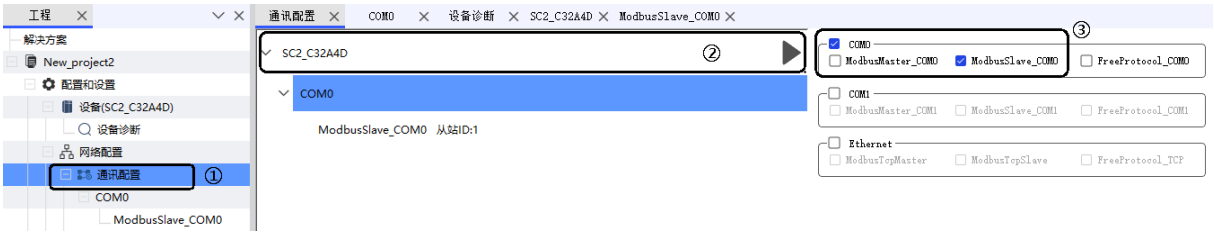
② 单击“通道指令配置”进入相关页面添加功能码；

③ 右键“添加”通道 0 功能码；

④ “功能码：03”，“触发方式：Cycle”，“周期：100ms”，“重发次数：1”，“读偏移：16#0000（首地址）”，“读长度：10”，“错误处理：keep the recent value”后单击“确定”，意味着可读取从站%MW0~%MW9 寄存器

3) PLC 从站通讯配置

a) 使能 PLC 作为 Modbus RTU 从站，如下图所示：



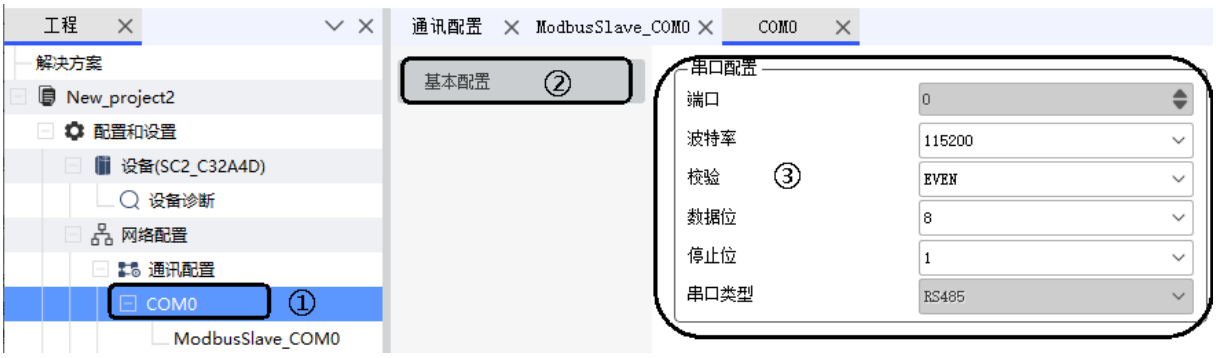
图：使能 PLC 作为 Modbus RTU 从站

① 左键双击网络配置下的“网络组态”；

② 左键单击 SC2-C32A4D 旁的 “右箭头按钮”；

③ 左键单击“COM0”及“COM0 从站”，做为基于 RS485 端口的 Modbus RTU 从站使用；

b) 使能 PLC 作为 Modbus RTU 从站，如下图所示：



图：Modbus RTU 从站通讯参数设置

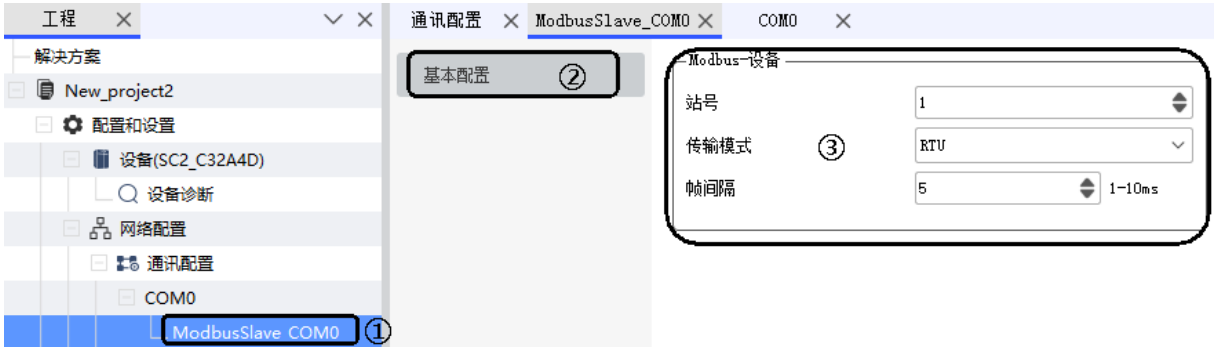
① 左键双击 “COM0”；

② 在 “基本配置”页面中设置参数；

③ 在“串口配置”中设置成以下通信参数，“波特率：115200”，“校验：EVEN”，“数

据位：8”，“停止位：1”；

c) Modbus RTU 从站站号设置，如下图所示：



图：Modbus RTU 从站站号设置

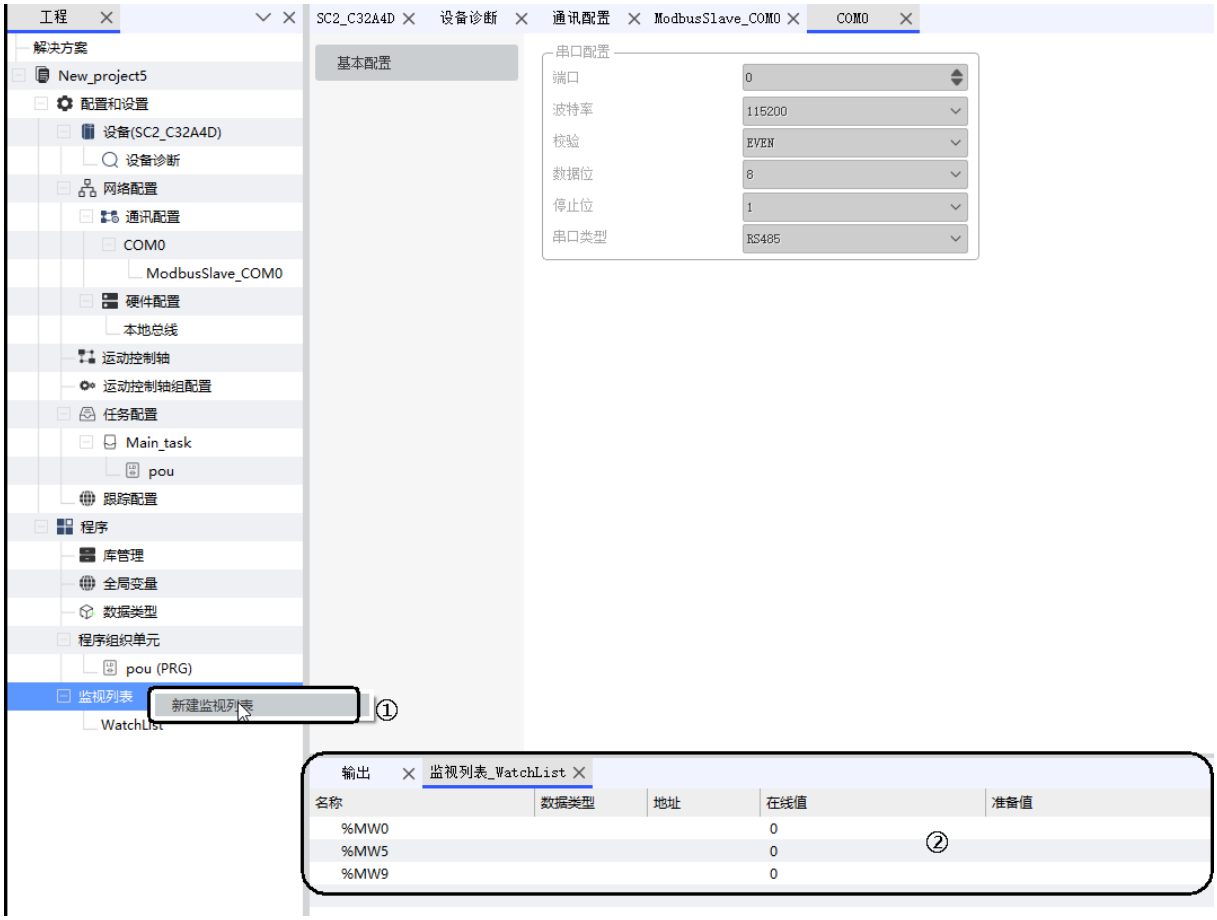
① 左键双击 “ModbusSlave COM0”；

② 在 “基本配置”页面中设置参数；

③ 在“Modbus 设备”中设置站号为 1；

4) 主站 PLC 读取从站 PLC 寄存器值

a) 打开从站 PLC 工程，对从站的寄存器赋值，如下图所示：



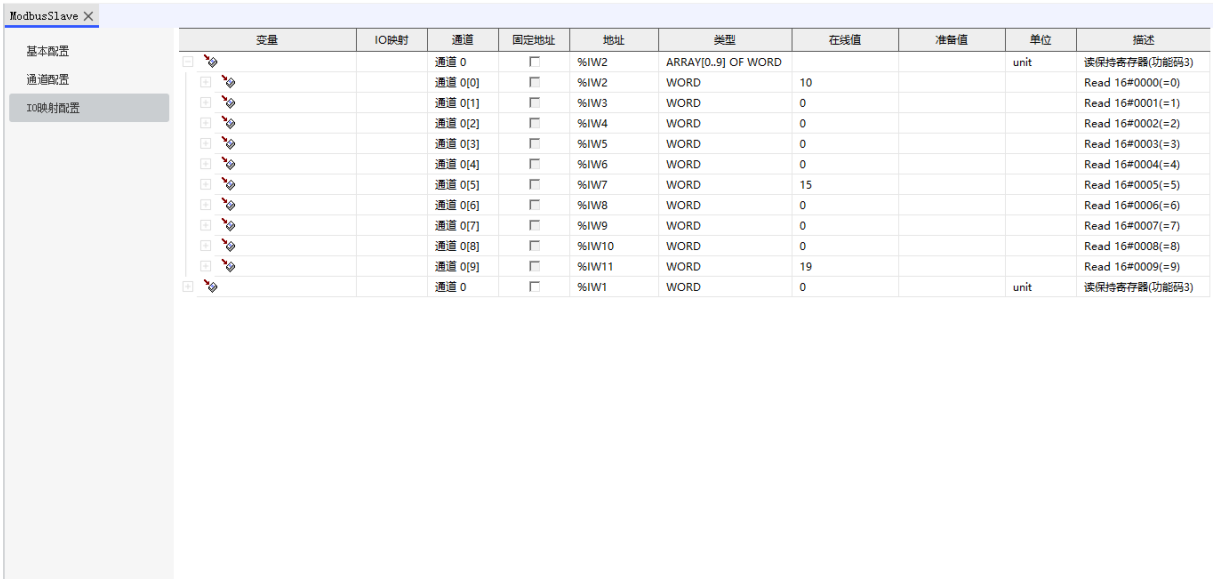
图：对从站寄存器赋值

① 右键单击“监视列表”，左键单击“新建监视列表”，创建 WatchList 监视；

② 在“监视列表”页面中增加相关参数“%MW0”、“MW5”、“%MW9”且进行赋值；

b) PLC 主站可读取从站的寄存器值，如下图所示。

双击“Modbus RTU Slave0”进入配置界面，单击“IO 映射配置”进入映射页面可见，从站寄存器读取成功。



图：主站读取从站寄存器

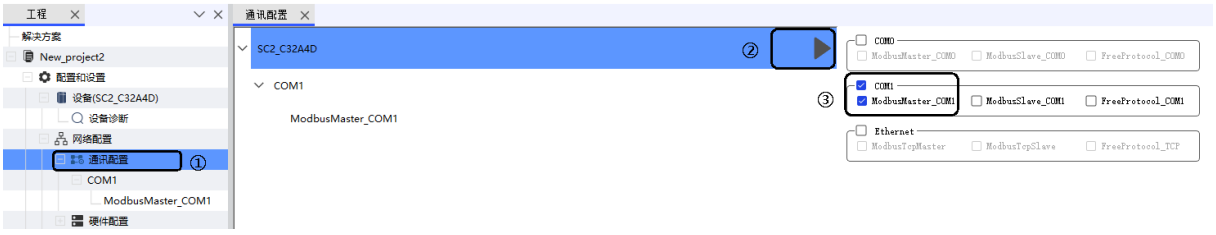
8.5 RS232 通讯配置

采用 RS232 接口进行 Modbus 协议的通讯，除硬件接线方法不同外，其他与 RS485 接口进行 Modbus 协议通讯的方法基本相同。

8.5.1 RS232 ModbusRTU 主站通讯及配置

1) 使能 SC2 系列 PLC 做为 Modbus RTU 主站

如下图所示，使能 SC2 系列 PLC 做主站。



图：使能 PLC 作为 Modbus RTU 主站

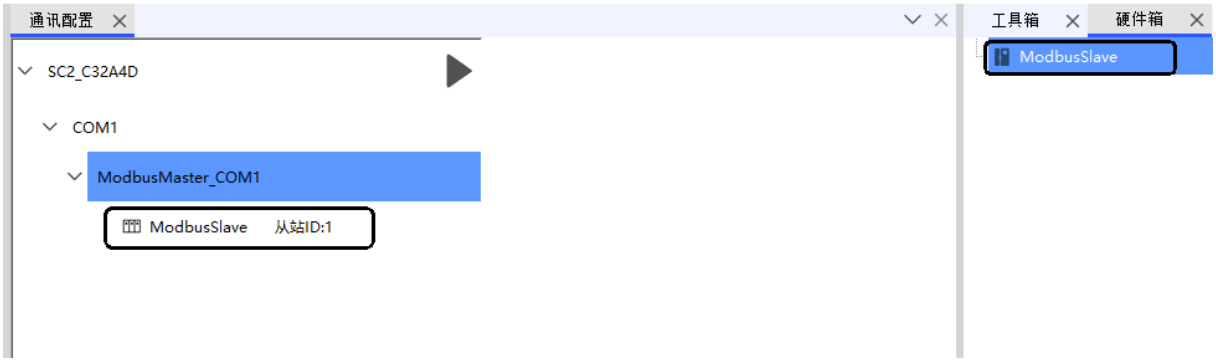
① 左键双击 “网络组态”；

② 左键单击 SC2-C32A4D 旁的 “右箭头按钮”；

③ 左键单击“COM1”及“COM1 主站”，做为基于 RS232 端口的 Modbus RTU 主站使用；

2) 添加 Modbus RTU 从站设备

如下图，在“工具箱”里左键双击的“Modbus Slave”添加 Modbus RTU 从站设备。



图：添加从站设备

3) Modbus RTU 主站配置

主站参数配置（串口通讯参数配置）与 SC 系列的 RS485 Modbus RTU 类似，见 7.2.1 章节。

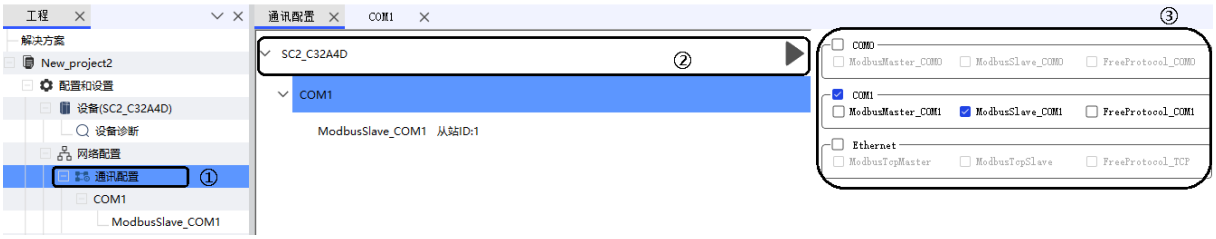
4) Modbus RTU 从站设备配置

从站设备配置（从站站号，功能码等）与 SC 系列的 RS485 Modbus RTU 类似，见 7.2.1 章节。

8.5.2 RS232 ModbusRTU 从站通讯及配置

1) 使能 SC2 系列 PLC 做为 Modbus RTU 从站

如下图所示，使能 SC2 系列 PLC 做为从站。



图：使能 PLC 作为 Modbus RTU 从站

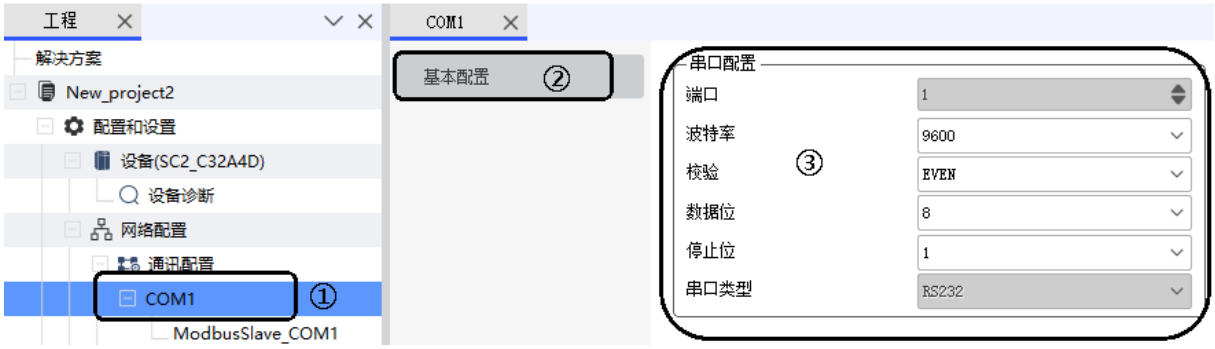
① 左键双击 “网络组态”；

② 左键单击 SC2-C32A4D 旁的 “右箭头按钮”；

③ 左键单击“COM1”及“COM1 从站”，做为基于 RS232 端口的 Modbus RTU 从站使用；

2) Modbus RTU 从站通讯参数设置

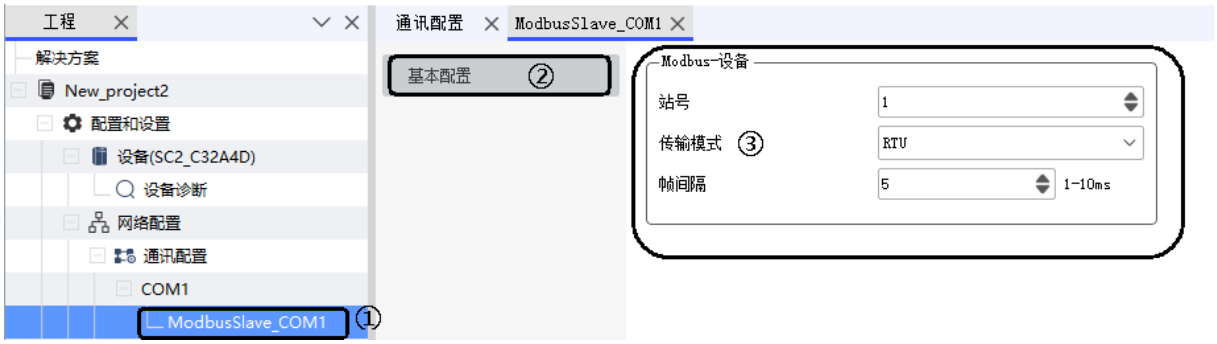
双击设备树中的“COM1”，在“基本配置”的“串口配置”里设置从站相关参数，相关参数如下图所示：



图： Modbus RTU 通讯参数配置

3) 从站站号设置

如下图所示，设置从站站号。



图：Modbus RTU 从站站号设置

① 左键双击 “ModbusSlave_COM1”；

② 在 “基本配置”页面中设置参数；

③ 在“Modbus 设备”中设置站号为 1；

8.5.3 RS232 ModbusRTU 通讯例程

除了添加 COM1 端口（RS232 端口）与 RS485 不同（COM0 为 RS485 端口），其余操作请参考 8.4 RS485 Modbus RTU 通讯例程章节。

9 以太网通讯

9.1 ModbusTCP 通讯

Modbus TCP 通讯与 Modbus RTU 通信链路不同，采用以太网链路，但 Modbus 应用层协议格式和使用方法基本相同。

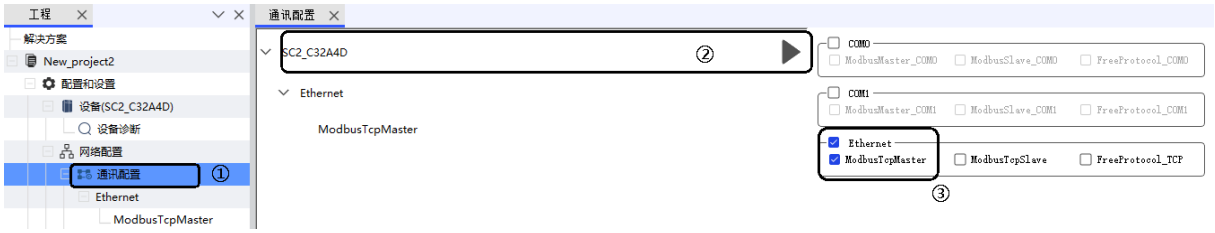
9.1.1 ModbusTCP 主站通讯

PLC 做 Modbus TCP 主站时，为 Modbus TCP 客户端（发起通讯的一方）。

本节介绍 2 个 SC2 系列 PLC 互做主从站进行通讯时，主站的参数设置及使用方法。

1) 使能 PLC 作为 Modbus TCP 主站

如下图所示，使能 SC2 系列 PLC 做主站。



图：使能 PLC 作为 Modbus TCP 主站

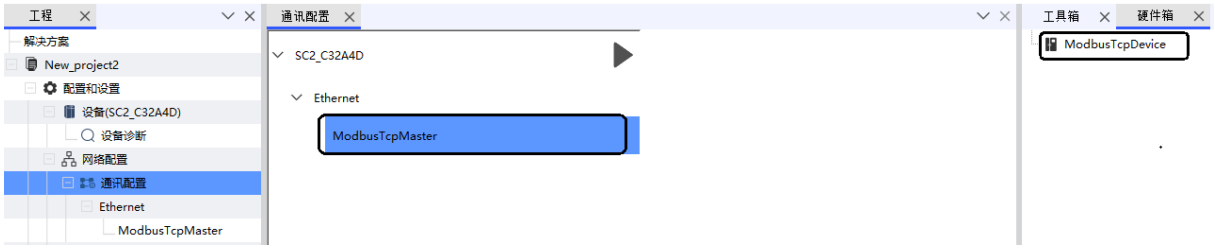
① 左键双击 “网络组态”；

② 左键单击 SC2-32A4D 旁的 “右箭头按钮”；

③ 左键单击“EtherNet”及“ModbusTcpMaster”，做为基于 EtherNet 端口的 Modbus TCP 主站使用；

2) 添加 Modbus TCP 从站设备

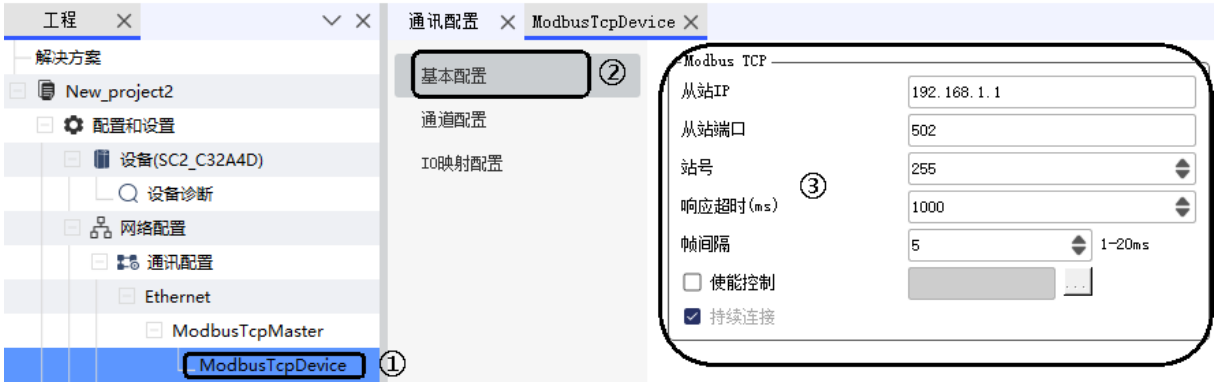
如下图，在“工具箱”里左键双击“Modbus TCP Device”添加 Modbus TCP 从站设备。



图：添加从站设备

3) Modbus TCP 从站配置

如下图及下表所示，配置从站通讯参数



图：从站通讯参数

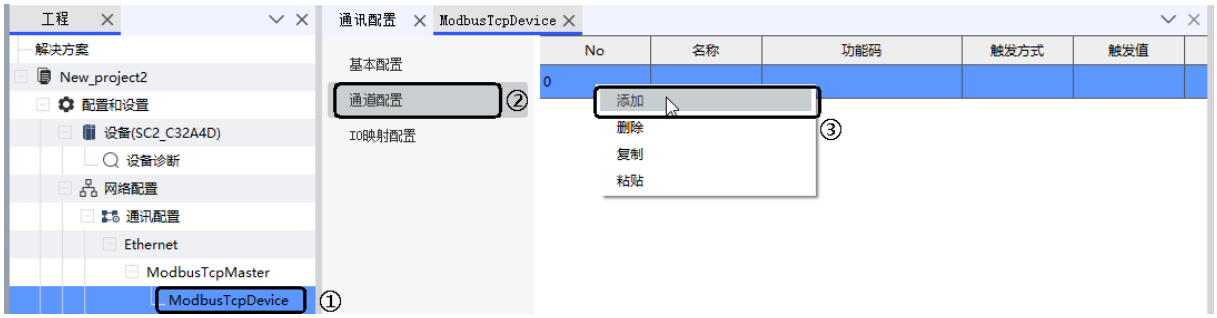
① 左键双击 “Modbus TCP Device”；

② 左键单击“基本配置”；

③ 在基本配置页面里设置从站的相关通讯配置参数；

4) Modbus TCP 从站参数设置

如下图所示，配置从站通讯参数



图：添加 Modbus 通道

图：设置功能码及其它参数

① 左键双击 “Modbus TCP Device”；

② 左键单击“通道配置”；

③ 右键 “添加”；

④ 在 “Modbus 通道” 页面里设置如图 8.5 所示参数，与 Modbus RTU 设置相同；

5) Modbus TCP 常见故障与错误码

a) Modbus TCP 主站连接 Modbus TCP 从站时发生的主要故障如下表所示：

表：Modbus TCP 错误码及说明

故障码	说明	故障码	说明
9	设备掉线	8007	从设备忙
104	连接断开	8008	存储奇偶性差错

110	响应超时	8010	不可用网关路径
113	访问的目标地址路由不可达	8011	网关目标设备响应失败
8001	非法功能码	8012	CRC 错误
8002	非法数据地址	8013	服务器(或从站)响应数据异常
8003	非法数据值	8014	服务器(或从站)异常响应
8004	从站设备故障	8015	无效功能码
8005	从机应答超时	8016	读/写线圈、寄存器个数过多
8006	从设备忙	8017	服务器(或从站)响应的站号与请求不一致

b) 错误响应帧格式

事务元标识符+协议标识符+长度+从机地址+（命令码+0x80） +错误码+CRC 校验。本错误帧适合所有的操作命令帧，如下表所示。

表：Modbus TCP 错误码及说明

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

c) Modbus TCP 通讯帧格式

① 功能码 0x01，读线圈寄存器，可以读取 Q 区变量。

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

表：功能码 0x01 请求帧详解

序号	描述	位/字操作	操作数量
1	事务元标识符	2 个字节	Modbus 请求/响应事务处理的识别码
2	协议标识符	2 个字节	0=Modbus 协议
3	长度	2 个字节	以下字节的数量
4	从机地址	1 个字节	取值 1-247
5	0x01	1 个字节	读线圈
6	线圈起始地址	2 个字节	高位在前，低位在后，见线圈编址
7	线圈数量 N	2 个字节	高位在前，低位在后

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

表：功能码 0x01 响应帧详解

序号	描述	位/字操作	操作数量
1	事务元标识符	2 个字节	Modbus 请求/响应事务处理的识别码
2	协议标识符	2 个字节	0=Modbus 协议
3	长度	2 个字节	以下字节的数量
4	从机地址	1 个字节	取值 1-247
5	0x01	1 个字节	读线圈
6	字节数	1 个字节	值：[(N+7)/8]
7	线圈状态	[(N+7)/8] 个字节	每 8 个线圈合为一个字节，最后一个若不足 8 位，未定义部分填 0。

			前 8 个线圈在第一个字节，地址最小的线圈在最低位。依次类推

② 功能码 0x02，读离散输入状态线圈寄存器，可以读取 I 区变量。

请求帧格式：事务元标识符+协议标识符+长度+从机地址+0x02+输入线圈状态起始地址+线圈数量，如下表所示。

表：功能码 0x02 请求帧详解

序号	描述	位/字操作	操作数量
1	事务元标识符	2 个字节	Modbus 请求/响应事务处理的识别码
2	协议标识符	2 个字节	0=Modbus 协议
3	长度	2 个字节	以下字节的数量
4	从机地址	1 个字节	取值 1-247
5	0x02	1 个字节	读输入状态离散线圈
6	线圈起始地址	2 个字节	高位在前，低位在后，见线圈编址
7	线圈数量 N	2 个字节	高位在前，低位在后

响应帧格式：事务元标识符+协议标识符+长度+从机地址 +0x01+字节数+离散线圈输入状态，如下表所示：

表：功能码 0x02 响应帧详解

序号	描述	位/字操作	操作数量
1	事务元标识符	2 个字节	Modbus 请求/响应事务处理的识别码
2	协议标识符	2 个字节	0=Modbus 协议
3	长度	2 个字节	以下字节的数量
4	从机地址	1 个字节	取值 1-247
5	0x02	1 个字节	读输入状态离散线圈

6	字节数	1 个字节	值：[(N+7)/8]
7	线圈状态	[(N+7)/8] 个字节	每 8 个线圈合为一个字节，最后一个若不足 8 位，未定义部分填 0。 前 8 个线圈在第一个字节，地址最小的线圈在最低位。依次类推

③ 读寄存器，功能码 0x03，可以读取 M 区变量。

请求帧格式：

事务元标识符+协议标识符+长度+从机地址+0x03+寄存器起始地址+寄存器数量， 如下表所示。

表：功能码 0x03 请求帧详解

序号	描述	位/字操作	操作数量
1	事务元标识符	2 个字节	Modbus 请求/响应事务处理的识别码
2	协议标识符	2 个字节	0=Modbus 协议
3	长度	2 个字节	以下字节的数量
4	从机地址	1 个字节	取值 1-247
5	0x03	1 个字节	读寄存器
6	寄存器起始地址	2 个字节	高位在前，低位在后，见寄存器编址
7	寄存器数量 N	2 个字节	高位在前，低位在后

响应帧格式：

事务元标识符+协议标识符+长度+从机地址+0x03+字节数+寄存器值，如下表所示：

表：功能码 0x03 响应帧详解

序号	描述	位/字操作	操作数量

1	事务元标识符	2 个字节	Modbus 请求/响应事务处理的识别码
2	协议标识符	2 个字节	0=Modbus 协议
3	长度	2 个字节	以下字节的数量
4	从机地址	1 个字节	取值 1-247
5	0x03	1 个字节	读寄存器
6	字节数	1 个字节	值：N*2
7	寄存器值	N*2 个字节	每两个字节表示一个寄存器值，高位在前低位在后。寄存器地址小的排在前面

④ 读输入寄存器，功能码 0x04，可以读取 I 区变量。

请求帧格式：

事务元标识符+协议标识符+长度+从机地址+0x04+输入寄存器起始地址+寄存器数量， 如下表所示：

表：功能码 0x04 请求帧详解

序号	描述	位/字操作	操作数量
1	事务元标识符	2 个字节	Modbus 请求/响应事务处理的识别码
2	协议标识符	2 个字节	0=Modbus 协议
3	长度	2 个字节	以下字节的数量
4	从机地址	1 个字节	取值 1-247
5	0x04	1 个字节	读输入寄存器
6	寄存器起始地址	2 个字节	高位在前，低位在后，见寄存器编址
7	寄存器数量 N	2 个字节	高位在前，低位在后

响应帧格式：

事务元标识符+协议标识符+长度+从机地址+0x04+字节数+输入寄存器值，如下表所示：

表：功能码 0x04 响应帧详解

序号	描述	位/字操作	操作数量
1	事务元标识符	2 个字节	Modbus 请求/响应事务处理的识别码
2	协议标识符	2 个字节	0=Modbus 协议
3	长度	2 个字节	以下字节的数量
4	从机地址	1 个字节	取值 1-247
5	0x04	1 个字节	读寄存器
6	字节数	1 个字节	值：N*2
7	寄存器值	N*2 个字节	每两个字节表示一个寄存器值，高位在前低位在后。寄存器地址小的排在前面

⑤ 写单个线圈，功能码 0x05，可以写 Q 区变量

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

表： 功能码 0x05 请求帧详解

序号	描述	位/字操作	操作数量
1	事务元标识符	2 个字节	Modbus 请求/响应事务处理的识别码
2	协议标识符	2 个字节	0=Modbus 协议
3	长度	2 个字节	以下字节的数量
4	从机地址	1 个字节	取值 1-247
5	0x05	1 个字节	写单线圈

6	线圈地址	2 个字节	高位在前，低位在后，见线圈编址
7	线圈状态	2 个字节	低位在前，高位在后，非 0 即为有效

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

表：功能码 0x05 响应帧详解

序号	描述	位/字操作	操作数量
1	事务元标识符	2 个字节	Modbus 请求/响应事务处理的识别码
2	协议标识符	2 个字节	0=Modbus 协议
3	长度	2 个字节	以下字节的数量
4	从机地址	1 个字节	取值 1-247
5	0x05	1 个字节	写单线圈
6	线圈地址	2 个字节	高位在前，低位在后，见线圈编址
7	线圈状态	2 个字节	低位在前，高位在后，非 0 即为有效

⑥ 写单个寄存器，功能码 0x06，可以写 M 区变量。

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

表：功能码 0x06 请求帧详解

序号	描述	位/字操作	操作数量
1	事务元标识符	2 个字节	Modbus 请求/响应事务处理的识别码
2	协议标识符	2 个字节	0=Modbus 协议
3	长度	2 个字节	以下字节的数量
4	从机地址	1 个字节	取值 1-247
5	0x06	1 个字节	写单寄存器

6	寄存器地址	2 个字节	高位在前，低位在后，见寄存器值编址
7	寄存器值	2 个字节	高位在前，低位在后。非 0 即为有效

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

表：功能码 0x06 响应帧详解

序号	描述	位/字操作	操作数量
1	事务元标识符	2 个字节	Modbus 请求/响应事务处理的识别码
2	协议标识符	2 个字节	0=Modbus 协议
3	长度	2 个字节	以下字节的数量
4	从机地址	1 个字节	取值 1-247
5	0x06	1 个字节	写单寄存器
6	寄存器地址	2 个字节	高位在前，低位在后，见寄存器值编址
7	寄存器值	2 个字节	高位在前，低位在后。非 0 即为有效

⑦ 写多个线圈，功能码 0x0f（15），可以写连续的多个 Q 区变量

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

表：功能码 0x0f 请求帧详解

序号	描述	位/字操作	操作数量
1	事务元标识符	2 个字节	Modbus 请求/响应事务处理的识别码
2	协议标识符	2 个字节	0=Modbus 协议
3	长度	2 个字节	以下字节的数量
4	从机地址	1 个字节	取值 1-247

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

响应帧格式：事务元标识符+协议标识符+长度+从机地址+0x0f+线圈数，如下表所示。

表：功能码 0x0f 响应帧详解

序号	描述	位/字操作	操作数量
1	事务元标识符	2 个字节	Modbus 请求/响应事务处理的识别码
2	协议标识符	2 个字节	0=Modbus 协议
3	长度	2 个字节	以下字节的数量
4	从机地址	1 个字节	取值 1-247
5	0x0f	1 个字节	写多个单线圈
6	线圈起始地址	2 个字节	高位在前，低位在后，见线圈编址
7	线圈数量	2 个字节	高位在前，低位在后。

⑧ 写多个寄存器，功能码 0x10（16）， 可以写连续的多个 M 变量。

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

表：功能码 0x10 请求帧详解

序号	描述	位/字操作	操作数量

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

响应帧格式：事务元标识符+协议标识符+长度+从机地址+0x10+线圈起始地址+线圈数量，如表下表所示。

表：功能码 0x10 响应帧详解

序号	描述	位/字操作	操作数量
1	事务元标识符	2 个字节	Modbus 请求/响应事务处理的识别码
2	协议标识符	2 个字节	0=Modbus 协议
3	长度	2 个字节	以下字节的数量
4	从机地址	1 个字节	取值 1-247
5	0x10	1 个字节	写多个单线圈
6	寄存器起始地址	2 个字节	高位在前，低位在后，见寄存器编址
7	寄存器数量	2 个字节	高位在前，低位在后，N

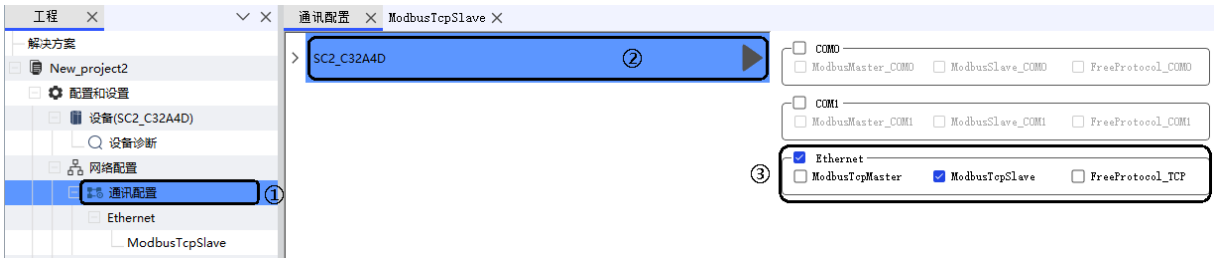
d) 其它相关参数配置

其它相关参数配置与 Modbus RTU 相同， 请参见 7.1、7.2 节内容。

9.1.2 ModbusTCP 从站通讯

1) 使能 PLC 作为 Modbus TCP 从站

如下图，设置 SC2 系列 PLC 为 Modbus TCP 从站。



图：使能 PLC 作为 Modbus TCP 从站

① 左键双击网络配置下的“网络组态”；

② 左键单击 SC2-C32A4D 旁的 “右箭头按钮”；

③ 左键单击“EtherNet”及“ModbusTcpSlave”，做为基于网口的 Modbus TCP 从站使用。

2) 设置 Modbus TCP 从站 IP

如下图所示,设置 Modbus TCP 从站的 IP。

图：设置 Modbus TCP 从站 IP

① 左键双击 “Device(SC2_C32A4D)”；

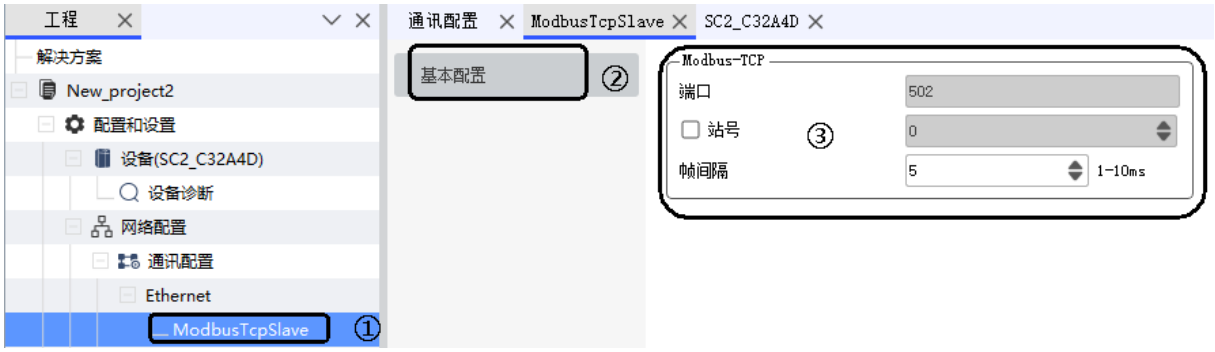
② 左键单击选择“自动扫描”进入设置页面；

③ 选中需要设置的 PLC

④ 设置 PLC 的 IP 后左键单击 “编辑”；

3) 设置 Modbus TCP 从站通讯参数

如下图所示,设置 Modbus TCP 从站的参数。



图：设置 Modbus TCP 从站通讯参数

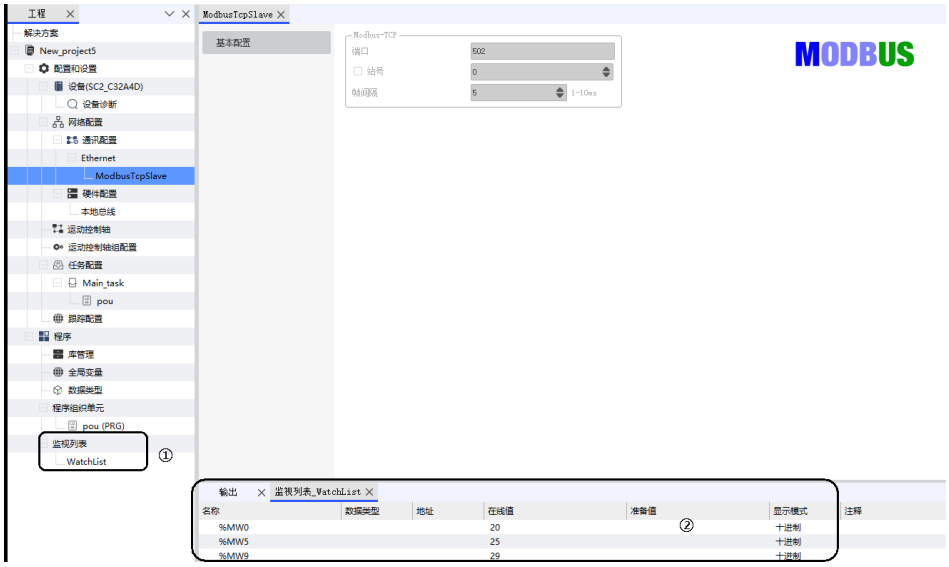
① 左键双击“EtherNet”下的“ModbusSlave”；

② 在“基本配置页面”内的“Modbus TCP”窗口设置参数；

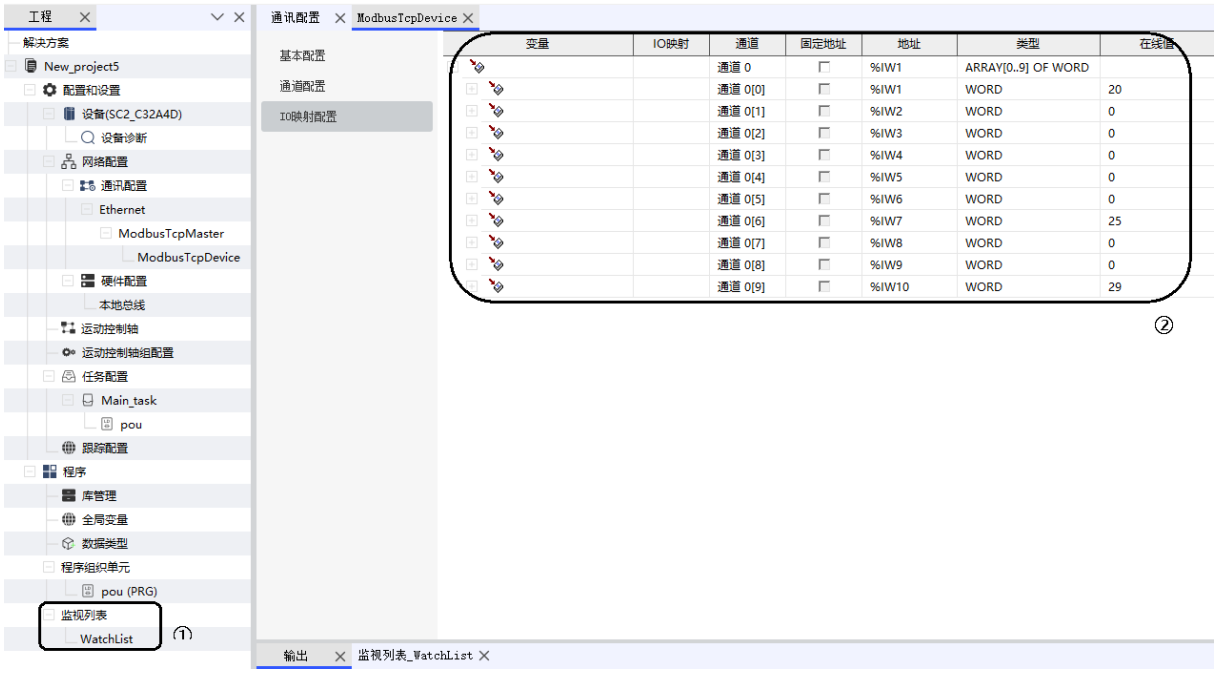
③ “端口”默认为 502（不可更改），勾选“站号”，可以修改站号；

4) 主站 PLC 读取从站 PLC 寄存器值

如下图所示，从站 PLC 寄存器赋值、主站 PLC 可读取从站寄存器数值。



图：Modbus TCP 从站寄存器赋值



图：Modbus TCP 主站读取寄存器值

① 在从站 PLC 创建 WatchList 监视，可参考 7.4 章节；

② 在“监视列表”页面中增加相关参数“%MW0”、“MW5”、“%MW9”且进行赋值，主站 PLC 可读取从站 PLC 寄存器数值；

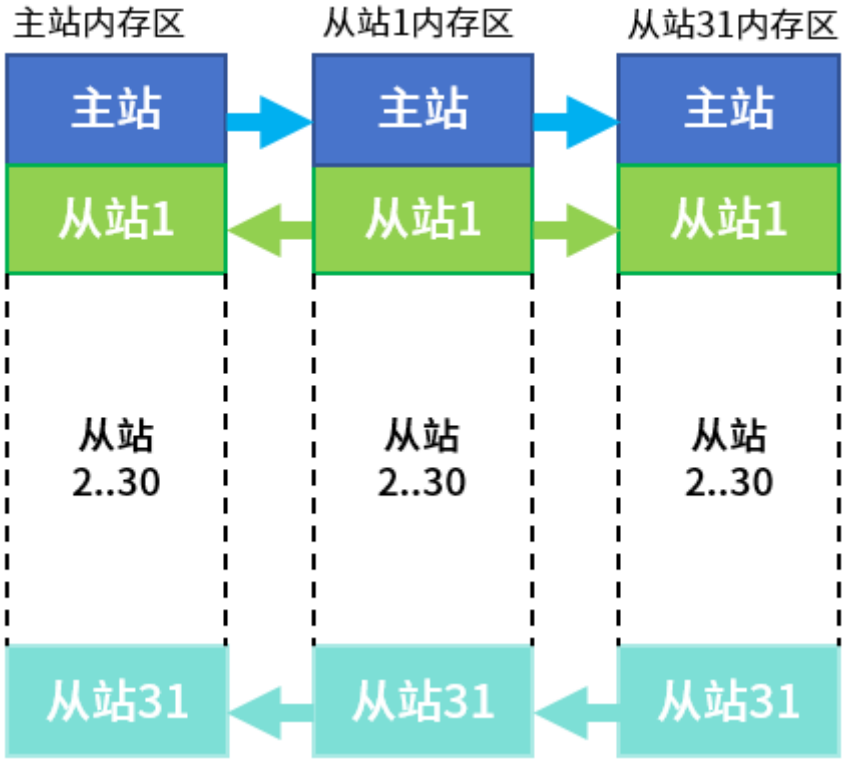
10 多机互联通讯

多机互联通讯可用于多台 PLC 之间的数据交互，传统的通讯方式配置相对繁琐，需要分开管理不同的 PLC 程序，LeadStudio 方案提供以下优势：

- 支持通过扫描进行配置，无需每个 PLC 管理一套程序；
- 最大支持 32 个从站之间进行数据交互，以太网模式下，单个站点通讯时间 4ms；
- 无需单独设置每个 PLC 的 IP 地址。
- 支持串口、以太网通讯方式

10.1 数据交互方式

多机互联通讯的数据交互方式可以参考下图，箭头代表数据传输的方向：



每个站点占用一片内存区域，本站点可以对该内存区域进行数据写入，其他站点中的该内存区域都为此站点的数据，可以从该区域读取目标站点的数据。

各站点占用的内存区域参考下表：

N:N 联机通信			
		MX	MW
模式 0 0 个 MX 元件 4 个 MW 元件	主站	无	MW500 ~ MW503
	从站 1	无	MW600 ~ MW603
	从站 2	无	MW700 ~ MW703
	从站 3	无	MW800 ~ MW803
	从站 4	无	MW900 ~ MW903
	从站 5	无	MW1000 ~ MW1003
	从站 6	无	MW1100 ~ MW1103
	从站 7	无	MW1200 ~ MW1203
	从站 08	无	MW1300 ~ MW1303
	从站 09	无	MW1400 ~ MW1403

	从站 10	无	MW1500 ~ MW1503
	从站 11	无	MW1600 ~ MW1603
	从站 12	无	MW1700 ~ MW1703
	从站 13	无	MW1800 ~ MW1803
	从站 14	无	MW1900 ~ MW1903
	从站 15	无	MW2000 ~ MW2003
	从站 16	无	MW2100 ~ MW2103
	从站 17	无	MW2200 ~ MW2203
	从站 18	无	MW2300 ~ MW2303
	从站 19	无	MW2400 ~ MW2403
	从站 20	无	MW2500 ~ MW2503
	从站 21	无	MW2600 ~ MW2603
	从站 22	无	MW2700 ~ MW2703
	从站 23	无	MW2800 ~ MW2803
	从站 24	无	MW2900 ~ MW2903
	从站 25	无	MW3000 ~ MW3003
	从站 26	无	MW3100 ~ MW3103
	从站 27	无	MW3200 ~ MW3203
	从站 28	无	MW3300 ~ MW3303
	从站 29	无	MW3400 ~ MW3403
	从站 30	无	MW3500 ~ MW3503
	从站 31	无	MW3600 ~ MW3603
模式 1 32 个 MX 元件 4 个 MW 元件	主站	MX500.0 ~ MX503.7	MW500 ~ MW503
	从站 1	MX510.0 ~ MX513.7	MW600 ~ MW603
	从站 2	MX520.0 ~ MX523.7	MW700 ~ MW703

	从站 3	MX530.0 ~ MX533.7	MW800 ~ MW803
	从站 4	MX540.0 ~ MX543.7	MW900 ~ MW903
	从站 5	MX550.0 ~ MX553.7	MW1000 ~ MW1003
	从站 6	MX560.0 ~ MX563.7	MW1100 ~ MW1103
	从站 7	MX570.0 ~ MX573.7	MW1200 ~ MW1203
	从站 08	MX580.0 ~ MX583.7	MW1300 ~ MW1303
	从站 09	MX590.0 ~ MX593.7	MW1400 ~ MW1403
	从站 10	MX600.0 ~ MX603.7	MW1500 ~ MW1503
	从站 11	MX610.0 ~ MX613.7	MW1600 ~ MW1603
	从站 12	MX620.0 ~ MX623.7	MW1700 ~ MW1703
	从站 13	MX630.0 ~ MX633.7	MW1800 ~ MW1803
	从站 14	MX640.0 ~ MX643.7	MW1900 ~ MW1903
	从站 15	MX650.0 ~ MX653.7	MW2000 ~ MW2003
	从站 16	MX660.0 ~ MX663.7	MW2100 ~ MW2103

	从站 17	MX670.0 ~ MX673.7	MW2200 ~ MW2203
	从站 18	MX680.0 ~ MX683.7	MW2300 ~ MW2303
	从站 19	MX690.0 ~ MX693.7	MW2400 ~ MW2403
	从站 20	MX700.0 ~ MX703.7	MW2500 ~ MW2503
	从站 21	MX710.0 ~ MX713.7	MW2600 ~ MW2603
	从站 22	MX720.0 ~ MX723.7	MW2700 ~ MW2703
	从站 23	MX730.0 ~ MX733.7	MW2800 ~ MW2803
	从站 24	MX740.0 ~ MX743.7	MW2900 ~ MW2903
	从站 25	MX750.0 ~ MX753.7	MW3000 ~ MW3003
	从站 26	MX760.0 ~ MX763.7	MW3100 ~ MW3103
	从站 27	MX770.0 ~ MX773.7	MW3200 ~ MW3203
	从站 28	MX780.0 ~ MX783.7	MW3300 ~ MW3303
	从站 29	MX790.0 ~ MX793.7	MW3400 ~ MW3403
	从站 30	MX800.0 ~ MX803.7	MW3500 ~ MW3503

	从站 31	MX810.0 ~ MX813.7	MW3600 ~ MW3603
模式 2 64 个 MX 元件 8 个 MW 元件	主站	MX500.0 ~ MX507.7	MW500 ~ MW507
	从站 1	MX510.0 ~ MX517.7	MW600 ~ MW607
	从站 2	MX520.0 ~ MX527.7	MW700 ~ MW707
	从站 3	MX530.0 ~ MX537.7	MW800 ~ MW807
	从站 4	MX540.0 ~ MX547.7	MW900 ~ MW907
	从站 5	MX550.0 ~ MX557.7	MW1000 ~ MW1007
	从站 6	MX560.0 ~ MX567.7	MW1100 ~ MW1107
	从站 7	MX570.0 ~ MX577.7	MW1200 ~ MW1207
	从站 08	MX580.0 ~ MX587.7	MW1300 ~ MW1307
	从站 09	MX590.0 ~ MX597.7	MW1400 ~ MW1407
	从站 10	MX600.0 ~ MX607.7	MW1500 ~ MW1507
	从站 11	MX610.0 ~ MX617.7	MW1600 ~ MW1607
	从站 12	MX620.0 ~ MX627.7	MW1700 ~ MW1707

	从站 13	MX630.0 ~ MX637.7	MW1800 ~ MW1807
	从站 14	MX640.0 ~ MX647.7	MW1900 ~ MW1907
	从站 15	MX650.0 ~ MX657.7	MW2000 ~ MW2007
	从站 16	MX660.0 ~ MX667.7	MW2100 ~ MW2107
	从站 17	MX670.0 ~ MX677.7	MW2200 ~ MW2207
	从站 18	MX680.0 ~ MX687.7	MW2300 ~ MW2307
	从站 19	MX690.0 ~ MX697.7	MW2400 ~ MW2407
	从站 20	MX700.0 ~ MX707.7	MW2500 ~ MW2507
	从站 21	MX710.0 ~ MX717.7	MW2600 ~ MW2607
	从站 22	MX720.0 ~ MX727.7	MW2700 ~ MW2707
	从站 23	MX730.0 ~ MX737.7	MW2800 ~ MW2807
	从站 24	MX740.0 ~ MX747.7	MW2900 ~ MW2907
	从站 25	MX750.0 ~ MX757.7	MW3000 ~ MW3007
	从站 26	MX760.0 ~ MX767.7	MW3100 ~ MW3107

	从站 27	MX770.0 ~ MX777.7	MW3200 ~ MW3207
	从站 28	MX780.0 ~ MX787.7	MW3300 ~ MW3307
	从站 29	MX790.0 ~ MX797.7	MW3400 ~ MW3407
	从站 30	MX800.0 ~ MX803.7	MW3500 ~ MW3507
	从站 31	MX810.0 ~ MX813.7	MW3600 ~ MW3607
模式 3 64 个 MX 元件 32 个 MW 元件	主站	MX500.0 ~ MX507.7	MW500 ~ MW531
	从站 01	MX510.0 ~ MX517.7	MW600 ~ MW631
	从站 02	MX520.0 ~ MX527.7	MW700 ~ MW731
	从站 03	MX530.0 ~ MX537.7	MW800 ~ MW831
	从站 04	MX540.0 ~ MX547.7	MW900 ~ MW931
	从站 05	MX550.0 ~ MX557.7	MW1000 ~ MW1031
	从站 06	MX560.0 ~ MX567.7	MW1100 ~ MW1131
	从站 07	MX570.0 ~ MX577.7	MW1200 ~ MW1231
	从站 08	MX580.0 ~ MX587.7	MW1300 ~ MW1331

	从站 09	MX590.0 ~ MX597.7	MW1400 ~ MW1431
	从站 10	MX600.0 ~ MX607.7	MW1500 ~ MW1531
	从站 11	MX610.0 ~ MX617.7	MW1600 ~ MW1631
	从站 12	MX620.0 ~ MX627.7	MW1700 ~ MW1731
	从站 13	MX630.0 ~ MX637.7	MW1800 ~ MW1831
	从站 14	MX640.0 ~ MX647.7	MW1900 ~ MW1931
	从站 15	MX650.0 ~ MX657.7	MW2000 ~ MW2031
	从站 16	MX660.0 ~ MX667.7	MW2100 ~ MW2131
	从站 17	MX670.0 ~ MX677.7	MW2200 ~ MW2231
	从站 18	MX680.0 ~ MX687.7	MW2300 ~ MW2331
	从站 19	MX690.0 ~ MX697.7	MW2400 ~ MW2431
	从站 20	MX700.0 ~ MX707.7	MW2500 ~ MW2531
	从站 21	MX710.0 ~ MX717.7	MW2600 ~ MW2631
	从站 22	MX720.0 ~ MX727.7	MW2700 ~ MW2731

	从站 23	MX730.0 ~ MX737.7	MW2800 ~ MW2831
	从站 24	MX740.0 ~ MX747.7	MW2900 ~ MW2931
	从站 25	MX750.0 ~ MX757.7	MW3000 ~ MW3031
	从站 26	MX760.0 ~ MX767.7	MW3100 ~ MW3131
	从站 27	MX770.0 ~ MX777.7	MW3200 ~ MW3231
	从站 28	MX780.0 ~ MX787.7	MW3300 ~ MW3331
	从站 29	MX790.0 ~ MX797.7	MW3400 ~ MW3431
	从站 30	MX800.0 ~ MX807.7	MW3500 ~ MW3531
	从站 31	MX810.0 ~ MX817.7	MW3600 ~ MW3631

10.2 使用步骤

10.2.1 非自定义模式（通过软件进行配置）

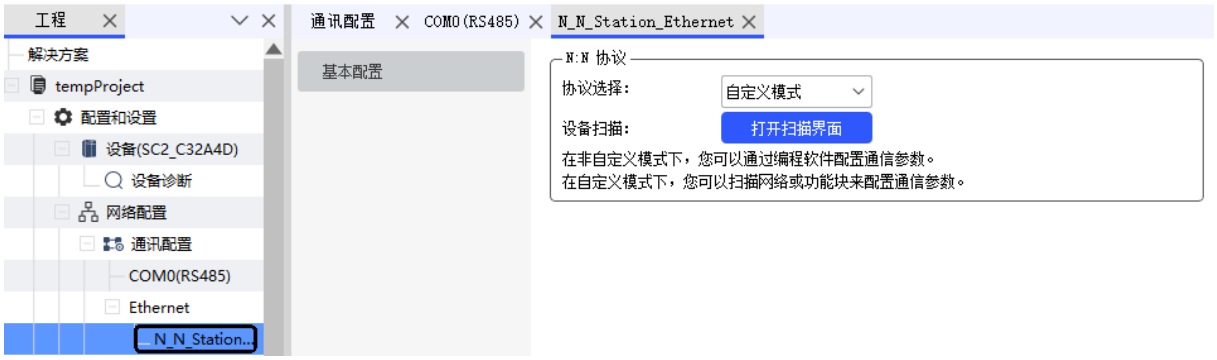
1）在通讯配置界面勾选 N_N_Station 协议

①使用串口模式（RS485），勾选 N_N_Station_COM0

②使用以太网模式，勾选 N_N_Station_Ethern

注：以太网与串口模式无法同时使用

2）勾选协议后，工程树中会生成多机互联通讯设备，双击可打开对应的配置界面

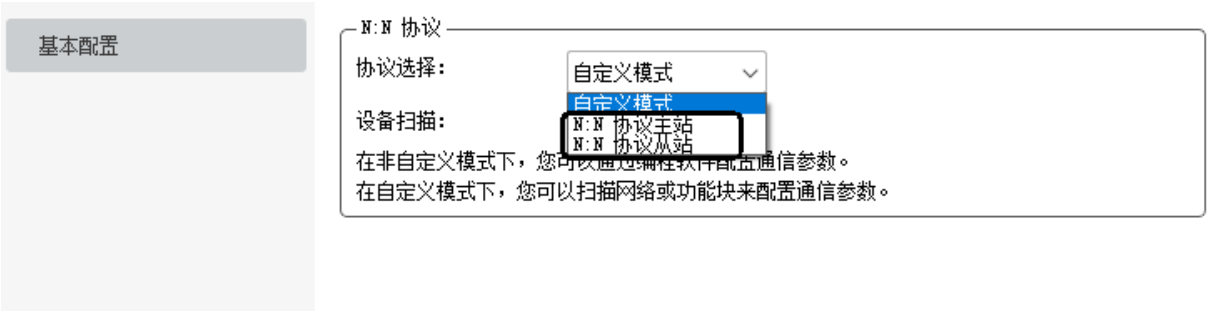


非自定义模式：通过编程软件配置通信参数

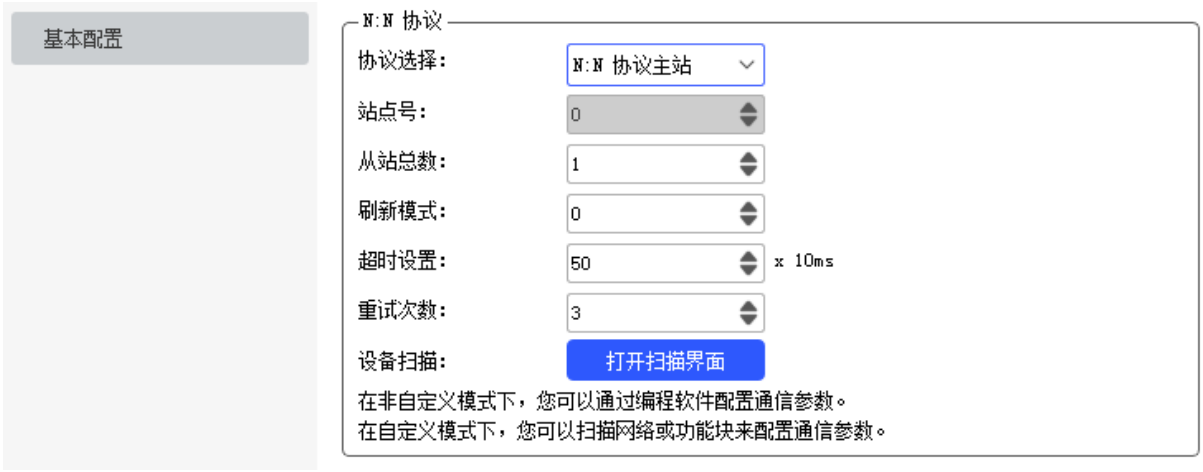
自定义模式：通过扫描（仅以太网支持）或功能块来配置通信参数

3）参数设置

非自定义模式下可选择 NN 协议主站或 NN 协议从站，单个网络能只能存在一个主站



主站设置参数如下：



①站点号：

主站为 0 且不支持修改；

②从站总数

支持设置 1-31

③刷新模式

支持 0-3

④超时设置

支持设置 5~6000 x 10ms

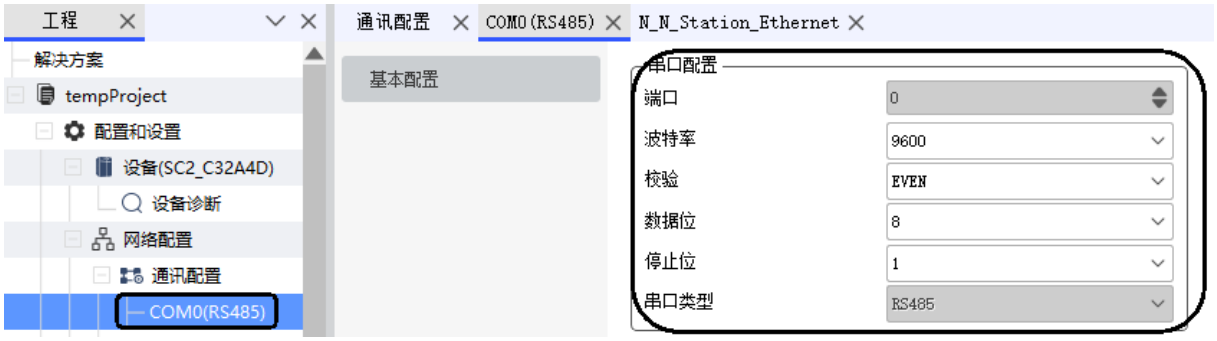
⑤重试次数

支持设置 0-10 次（默认 3）

从站只需要设置站号即可，范围为 1-31。

注：

1）串口模式下需要设置串口通讯参数，网络内的设备需要保证通讯参数一直



2）参数配置需要下载生效

10.2.2 自定义模式（通过扫描或功能块进行配置）

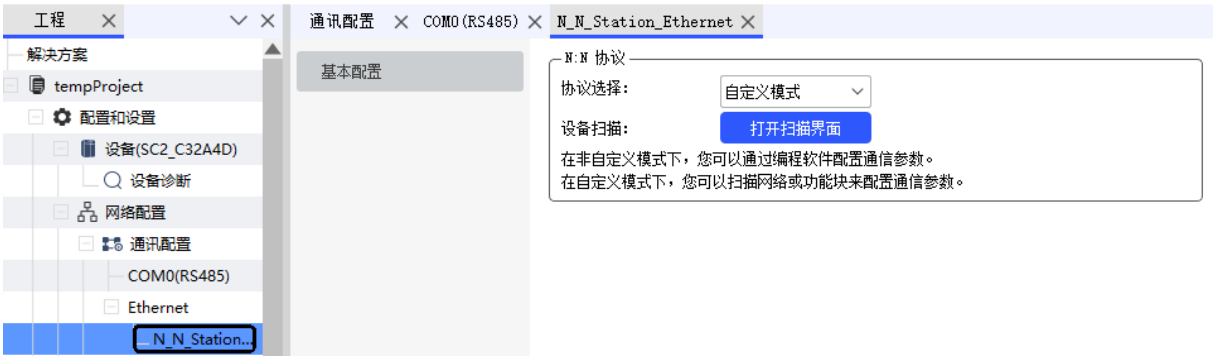
1）在通讯配置界面勾选 N_N_Station 协议，

①使用串口模式（RS485），勾选 N_N_Station_COM0

②使用以太网模式，勾选 N_N_Station_Ethern

注：以太网与串口模式无法同时使用

2）勾选协议后，工程树中会生成多机互联通讯设备，双击可打开对应的配置界面

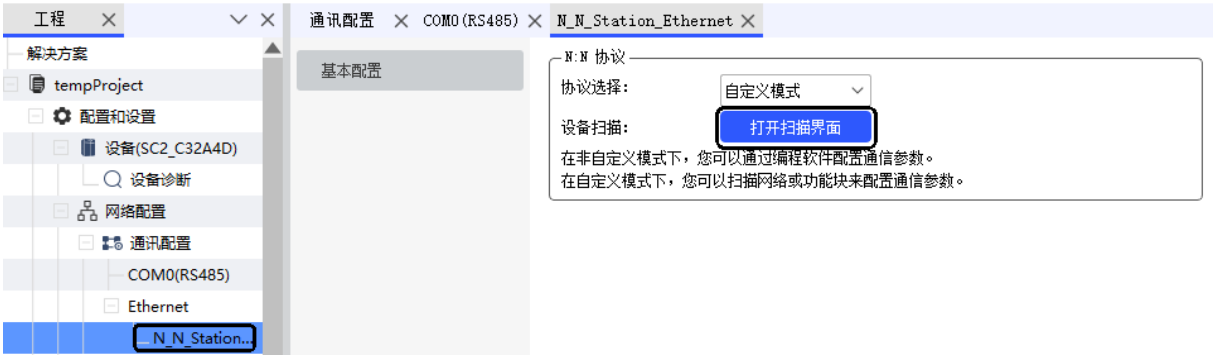


非自定义模式：通过编程软件配置通信参数

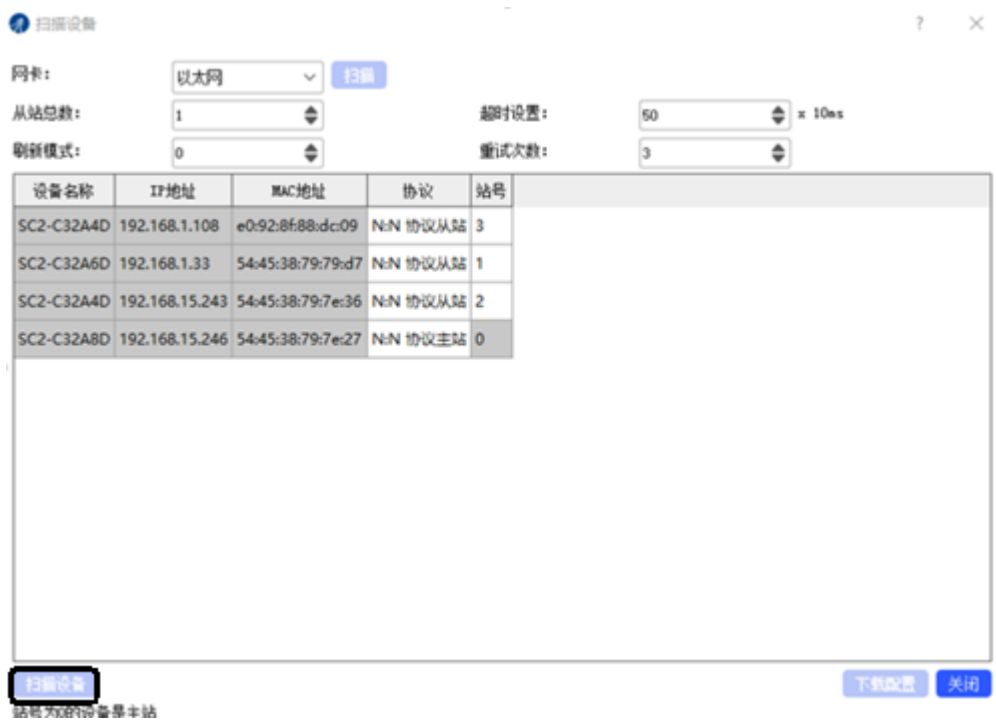
自定义模式：通过扫描（仅以太网支持）或功能块来配置通信参数

3）使用扫描方式进行配置

在 2）打开的配置界面中，点击“打开扫描界面”；



点击“打开扫描设备”按钮，可以将当前局域网内的设备全部扫描到列表中



用户可以在列表中编辑参数信息，编辑完成后点击“下载配置”按钮，即可完成配置参数的下发，配置下载后需要重新上下电、完整下载工程、冷热复位生效。

4）使用功能块方式进行配置

使用功能块进行配置读写时，使用的功能块参考下方介绍：

①LS_Get_NN_Param

指令说明：

xExecute 检测到上升沿时，读取对应的通讯口的 NN 协议通讯参数。

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
xExecute	触发	BOOL	TRUE-FALSE	FALSE	上升沿触发指令。

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
xDone	完成	BOOL	TRUE-FA	FALSE	如果功能块执行完

			LSE		成则为 TRUE。
xBusy	执行中	BOOL	TRUE-FALSE	FALSE	当功能块执行还没结束时为 TRUE。
xError	错误	BOOL	TRUE-FALSE	FALSE	在功能块内部发生错误的信号
nErrorID	错误代码	LS_NNBus_ErrorID	-	0	正常时数值为 0，发生异常时，输出报错代码。
byNNProtocol	协议	BYTE	0-2	0	0：自定义模式 1:N:N 协议主站 2:N:N 协议从站
byStation	站点号	BYTE	0-31	0	站号
bySlaveNum	从站总数	BYTE	0-31	0	从站总数
byMode	刷新模式	BYTE	0-3	0	0：模式 0 1：模式 1 2：模式 2 3：模式 3
iTimeOut	超时时间	INT	5-6000	50	超时时间(单位 10ms)
byRetryNum	重试次数	BYTE	0-10	3	重试次数

LS_NNBus_ErrorID 的描述参考下表

值	描述
0	无错误
1	没有有效的 NN 设备
2	保存参数失败
3	无效的协议类型
4	无效的站号
5	无效的从站数量
6	无效的刷新模式
7	无效的超时时间
8	无效的重发次数

②LS_Set_NN_Param

指令说明：

xExecute 检测到上升沿时，设置对应的通讯口的 NN 协议通讯参数，该指令设置参数掉电不保持（热复位、冷复位、重新上电不丢失）

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
xExecute	触发	BOOL	TRUE-FALSE	FALSE	上升沿触发指令。
byNNProtocol	协议	BYTE	0-2	0	0：自定义模式 1:N:N 协议主站 2:N:N 协议从站
byStation	站点号	BYTE	0-31	0	站号
bySlaveNum	从站总数	BYTE	0-31	0	从站总数

byMode	刷新模式	BYTE	0-3	0	0：模式 0 1：模式 1 2：模式 2 3：模式 3
iTimeOut	超时时间	INT	5-6000	50	超时时间(单位 10ms)
byRetryNum	重试次数	BYTE	0-10	3	重试次数

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
xDone	完成	BOOL	TRUE-FALSE	FALSE	如果功能块执行完成则为 TRUE。
xBusy	执行中	BOOL	TRUE-FALSE	FALSE	当功能块执行还没结束时为 TRUE。
xError	错误	BOOL	TRUE-FALSE	FALSE	在功能块内部发生错误的信号
nErrorID	错误代码	LS_NNBus_ErrorID	-	0	正常时数值为 0，发生异常时，输出报错代码。

LS_NNBus_ErrorID 的描述参考下表

值	描述
0	无错误
1	没有有效的 NN 设备
2	保存参数失败
3	无效的协议类型

4	无效的站号
5	无效的从站数量
6	无效的刷新模式
7	无效的超时时间
8	无效的重发次数

10.3 故障诊断

可以通过设备名称+“.”列出 NN 协议相关的变量，包含以下成员：

1）Stamp（TIME 类型）：故障发生的时间戳

2）ErrorCode（UINT 类型）：故障码

故障码参考下表：

值	错误
0	无错误
1000	设备打开失败
1001	设备内存分配失败
1002	主站设备超时
1003	从站设备超时
1004	从站站号冲突
1005	主站广播帧错误
1006	数据帧错误
1007	通信计数器错误

11 HMI 配置

HMI 配置用于快速建立 PLC 变量与 HMI 之间的通讯关系，LeadStudio 方案提供以下优势：

- 快速为变量映射地址，简化用户操作；
- 用户无需自行计算 Modbus 地址偏移，通过导入导出操作即可将变量地址关联

到 HMI 中

支持串口、以太网通讯方式

11.1 操作步骤

1）添加在 HMI 软件添加定制驱动；

2）将 PLC 中的变量添加到 LeadStudio 软件的 HMI 配置表中；

3）在 LeadStudio 软件中根据需要手动或自动分配地址；

4）从 LeadStudio 软件中导出 HMI 配置表；

5）导入 HMI 配置表到 HMI 软件中

下文将按照顺序介绍各步骤的详细操作

11.2 添加 HMI 驱动

11.2.1 MCGS

MCGS 定制驱动可在下方链接获取：

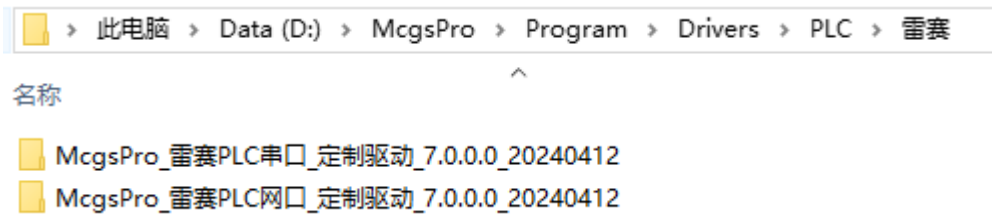
http://kzcp.leisai.com/Lead_Studio.html

添加 HMI 驱动的操作步骤如下：

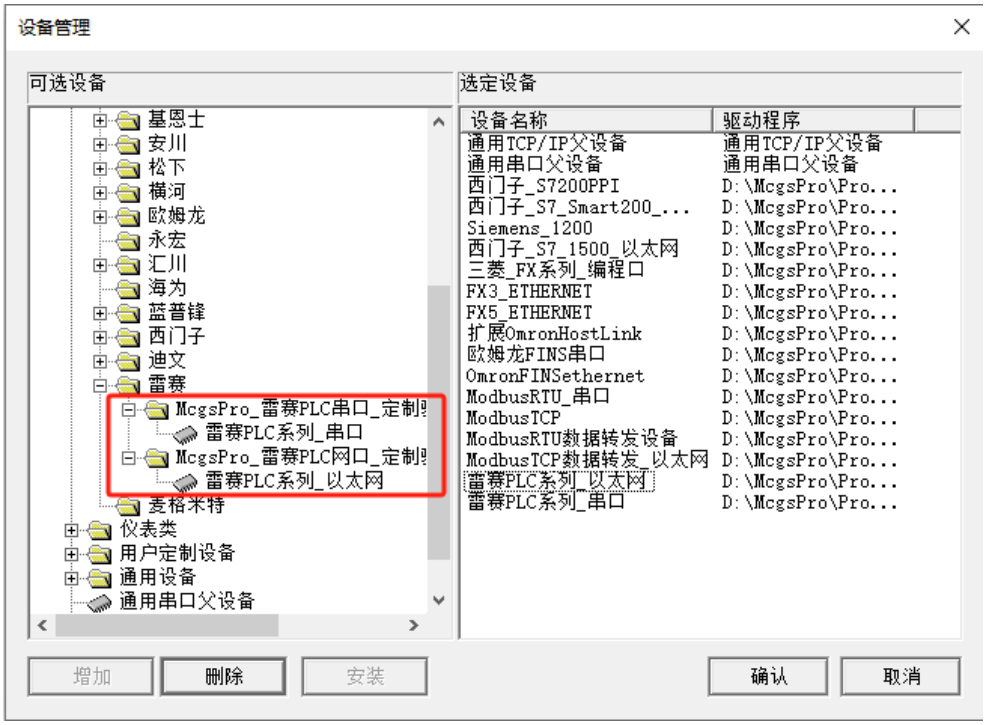
1）解压“McgsPro_雷赛 PLC 网口_定制驱动_7.0.0.0_20240412”及“McgsPro_雷赛 PLC 串口口_定制驱动_7.0.0.0_20240412”

2）在“自定义安装路径\McgsPro\Program\Drivers\PLC”中创建文件夹，命名为“雷赛”；

3）将解压完成的文件放置 2）中创建的文件夹里，可参考下图



4）重新打开 McgsPro 组态环境，在设备中可找到对应的驱动设备，则添加成功



11.3 添加 PLC 变量到 HMI 配置表

添加 PLC 变量到 HMI 配置表，有三种不同的操作方式：

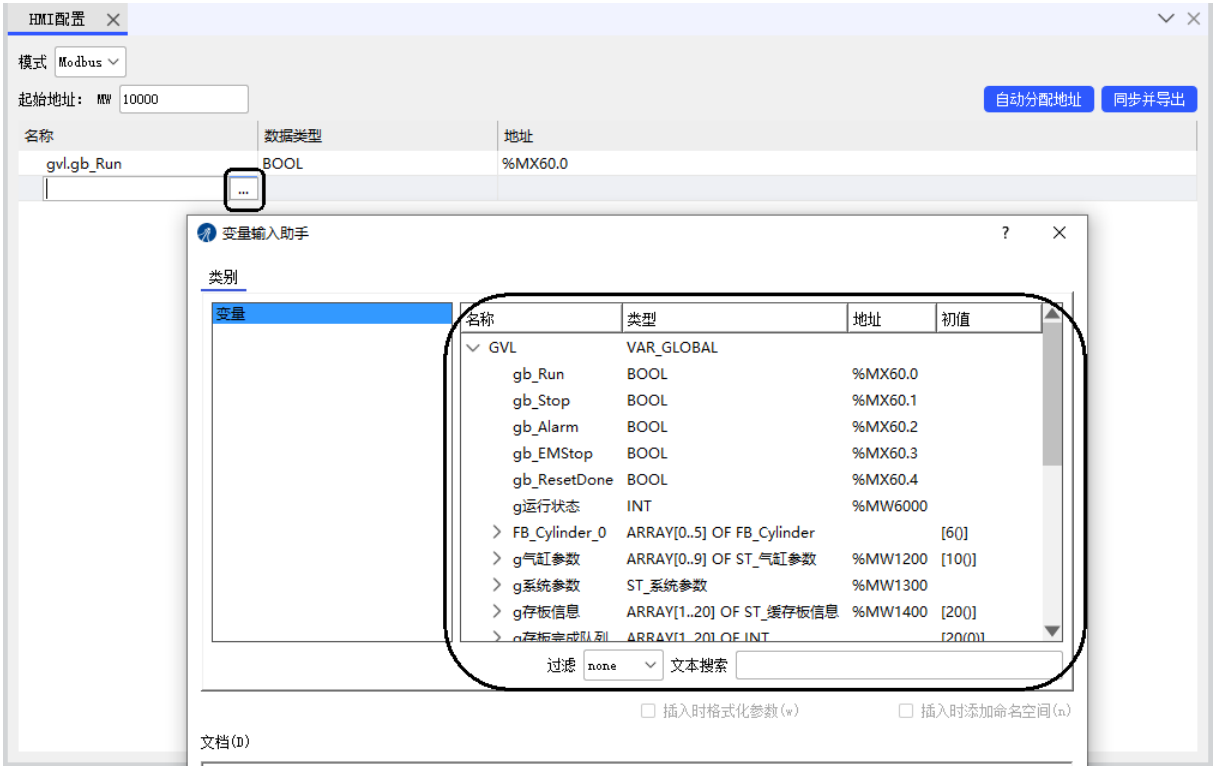
1）直接在输入框中输入变量进行添加

选中当前行，激活输入框，此时可以直接通过键盘输入变量名，注意此处需要输入完整的变量名称（包含全局变量或 POU 名）；

此方式支持直接输入 IQM 区地址；

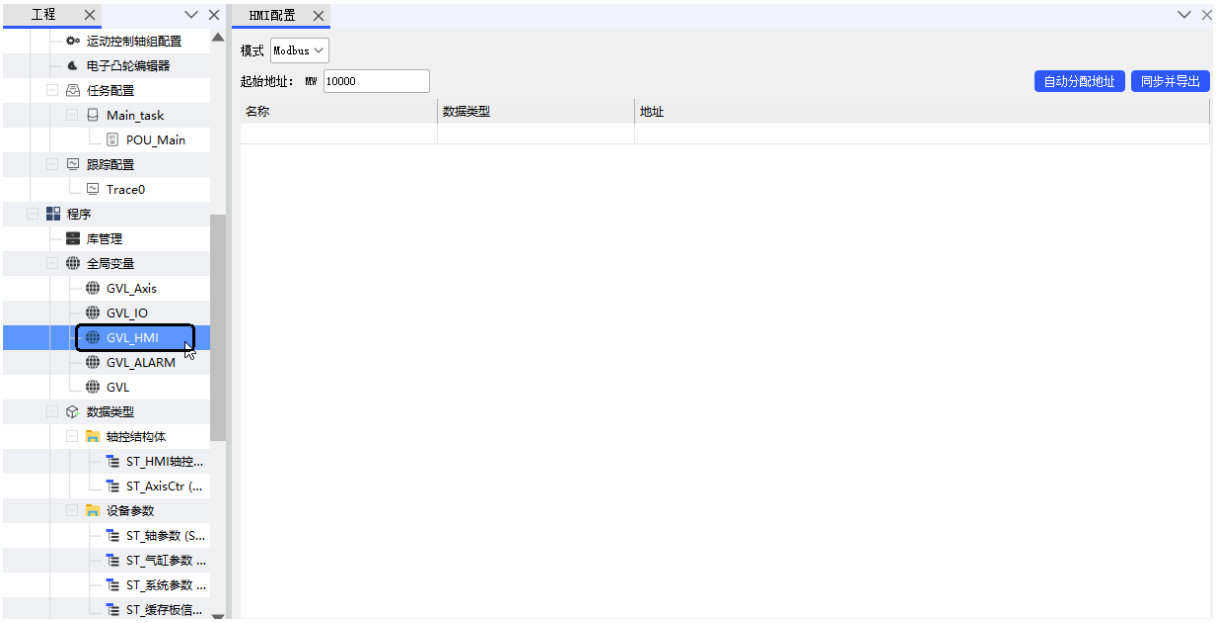
2）通过输入助手添加变量

选中当前行，激活输入框，此时可以看到输入框右侧有“...”按钮，点击该按钮可激活输入助手，用户可以在输入助手中选择需要插入的变量；

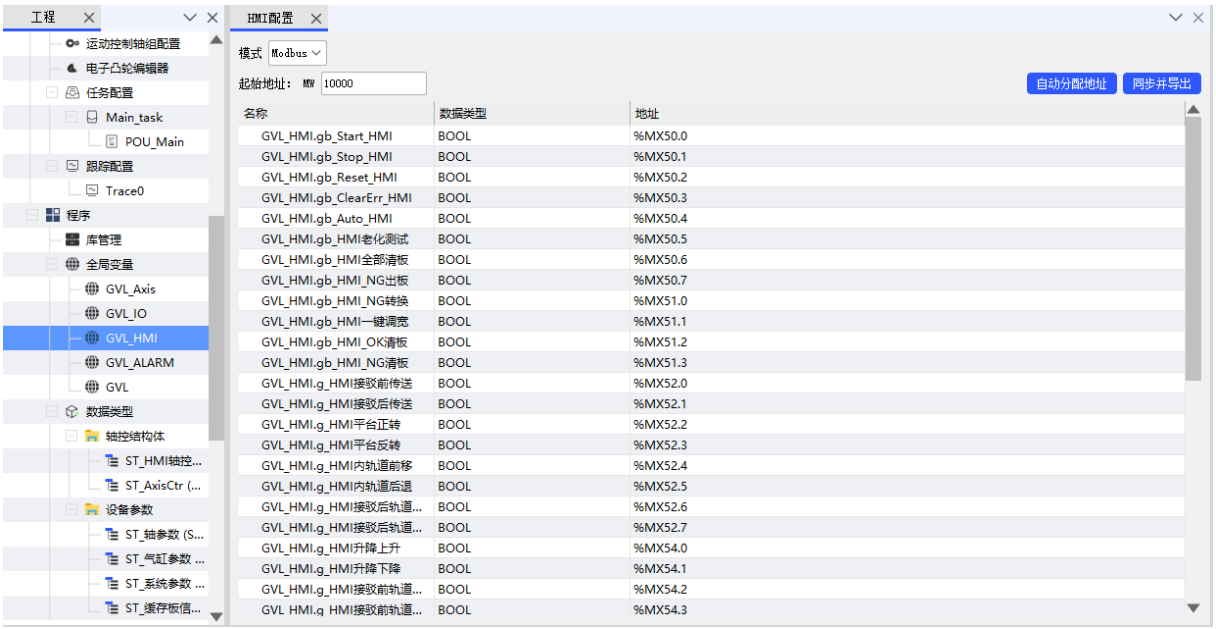


3）拖拽 POU 或全局变量表到 HMI 配置表，批量添加变量

鼠标左键选中需要添加的 POU 或变量表，长按并拖动到 HMI 配置表中，可完成变量表中所有变量的批量插入；



插入后：



11.4 分配 PLC 变量地址

PLC 变量地址支持手动或自动分配；

1）用户可以在变量表映射变量地址，在添加进 HMI 配置表时，会将该地址带入到 HMI 配置表中；

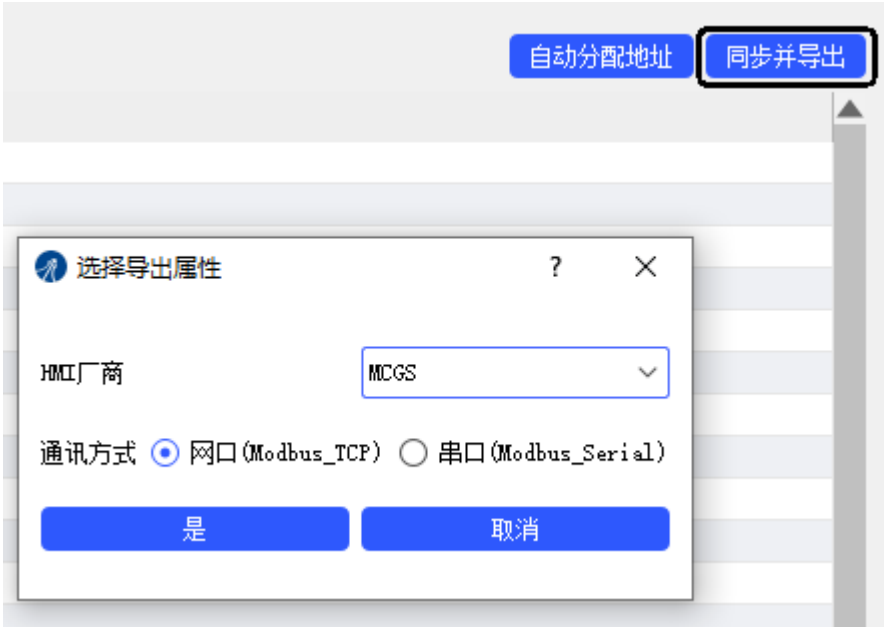
2）用户可以在 HMI 配置表修改或分配变量映射地址，在执行“同步并导出”操作时，软件会将修改同步到程序中；

3）用户可以在 HMI 配置表中点击“自动分配地址”，使软件自动分配变量映射地址，执行此操作时，软件将会从设定的“起始地址”开始，依次分配每个变量的地址（自动分配只针对没有映射地址的变量，如果变量当前存在映射地址，则不会参与到自动分配中），在执行“同步并导出”操作时，软件会将分配的地址同步到程序中；

11.5 导出 HMI 配置表

当所有变量地址分配完成后，用户可以点击“同步并导出”按钮，执行变量表导出

操作；

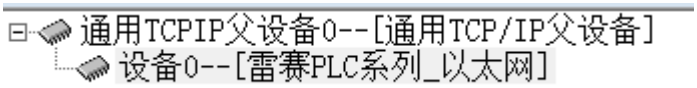


此界面用户可以选择与 PLC 通讯的 HMI 厂商以及通讯的方式，设定完成之后点击“是”，即可执行导出。

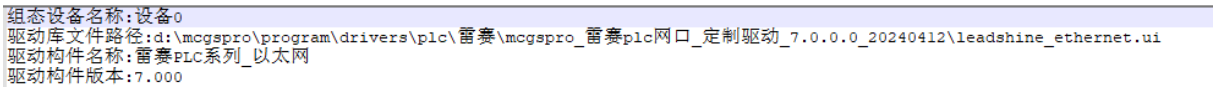
11.6 导入 HMI 配置表及相关设置

11.6.1 MCGS

1）在设备窗口根据实际使用的硬件配置，添加对应的设备，此处以 TCP 设备为例，添加“通用串口父设备”及“雷赛 PLC 系列_以太网”；

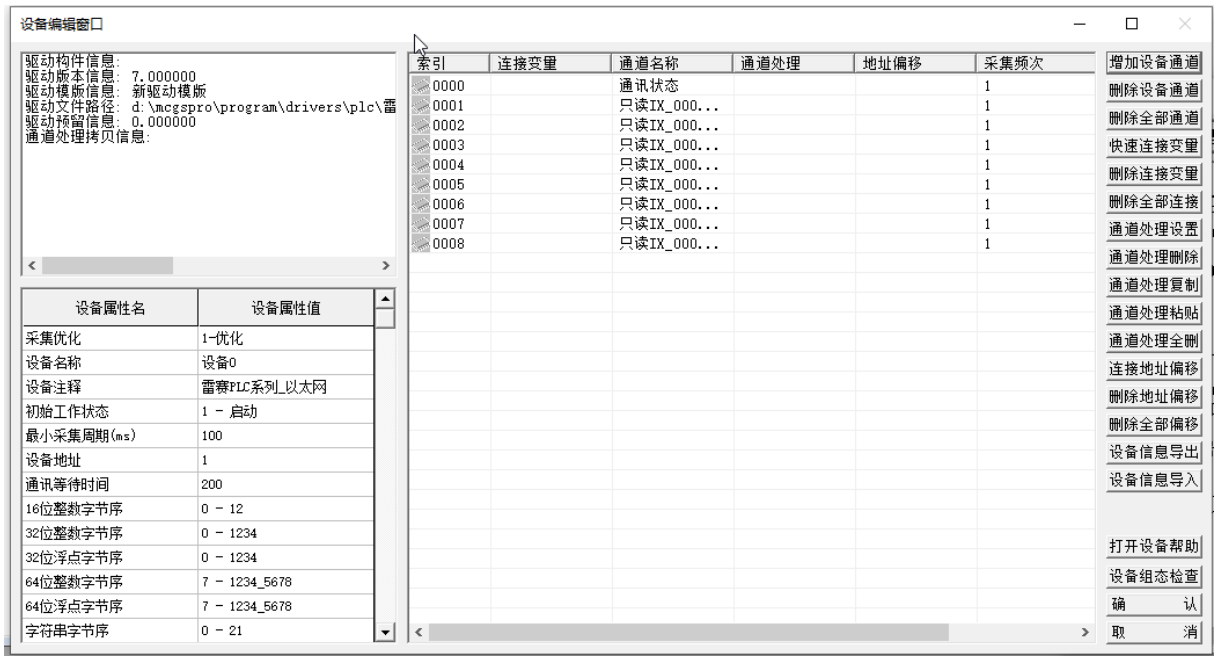


2）修改 HMI 配置表中的内容，匹配用户电脑，需要修改“组态设备名称”（与 HMI 组态软件中的设备名匹配）及“驱动库文件路径”（与驱动文件放置路径匹配）；

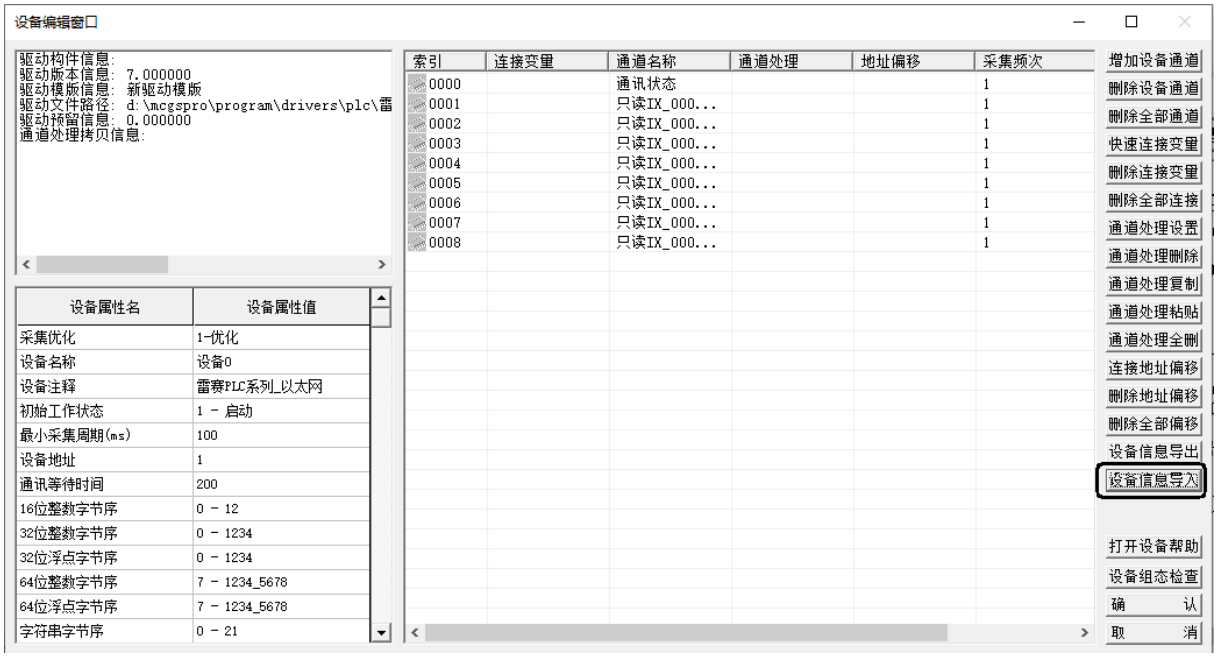


下图为与上图驱动库文件路径匹配的驱动文件放置路径

3）双击“设备 0”打开设备变量编辑窗口



4）在打开的编辑界面中，选择“设备信息导入”，并选择从 PLC 中导出的 HMI 配置表，即可完成变量信息导入。



5）设备属性设置参考下图，需要设置“32 位整数字节序”及“32 位浮点字节序”

设备属性名	设备属性值
采集优化	1-优化
设备名称	设备0
设备注释	雷赛PLC系列_以太网
初始工作状态	1 - 启动
最小采集周期(ms)	100
设备地址	1
通讯等待时间	200
16位整数字节序	0 - 12
32位整数字节序	2 - 3412
32位浮点字节序	2 - 3412
64位整数字节序	7 - 1234_5678
64位浮点字节序	7 - 1234_5678
字符串字节序	0 - 21

11.7 注意事项

1)该功能基于 Modbus 实现，故实际通讯时需要将 PLC 配置为对应的 Modbus 从站；

2）该功能基于 Modbus 实现，故存在以下限制：

①8 位变量类型（如 SINT)、64 位变量（如 LREAL)、WSTRING 类型不支持添加到 HMI 符号表中；

②若变量被分配到奇数地址，如 MB83，则不支持导出；

③未分配地址的变量，不会被导出；

④若结构体或数组中存在不支持的数据类型，仍可以导入到 HMI 符号表中，导出的时候，不支持的变量不导出到 CSV 文件中

3）由于 HMI 组态软件不支持特殊符号，故变量中的“.”、“[”、“]”等符号，在导入到 HMI 组态软件时，使用下划线进行代替，如“GVL_HMI.gb_HMI 启动”导入到 HMI 组态软件后会替换为“GVL_HMI_gb_HMI 启动”。

12 运动控制功能

12.1 概述

12.1.1 控制基本逻辑

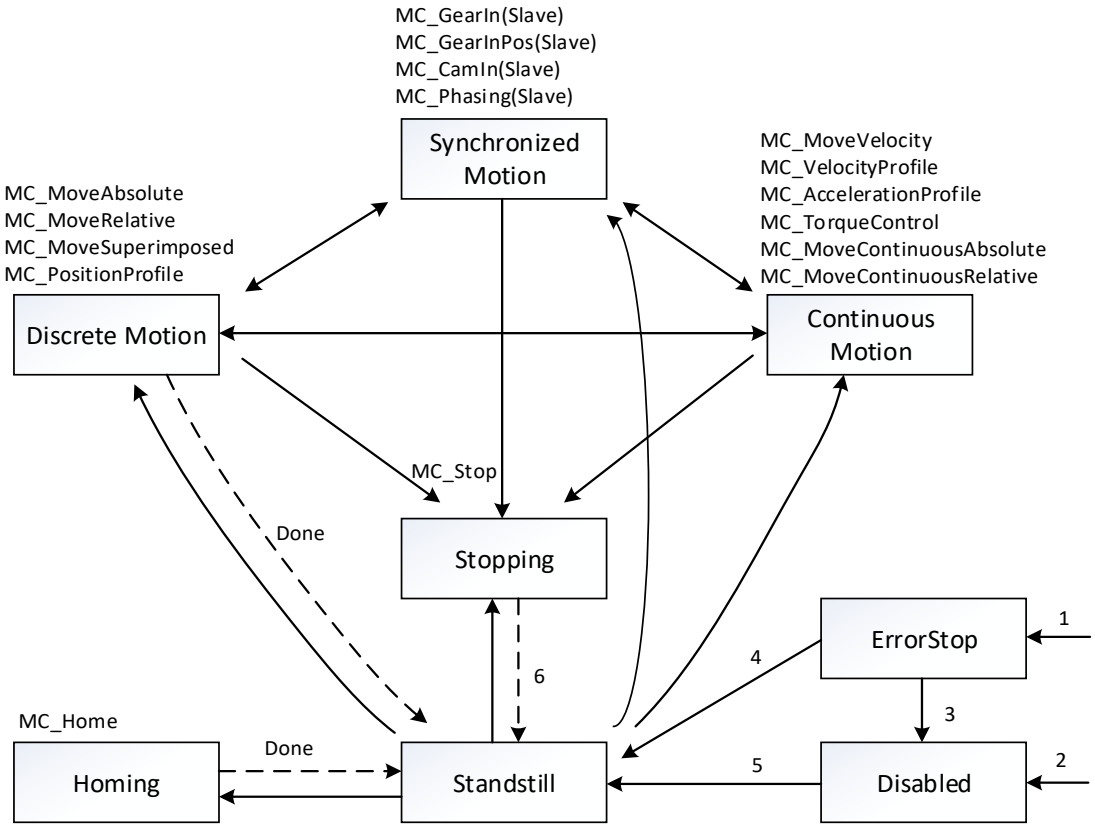
在 LeadStudio 运动控制系统中，将运动控制的对象称为轴。轴是连接驱动器和控制

器指令间的桥梁。LeadStudio 的运动控制轴用于控制符合 CIA402 协议的控制本地高速脉冲输出轴。其控制过程如下：

1) 在用户程序中启动运动控制指令，将在相应的 PLCopen 运动库和雷赛运动库中对运动算法进行分析。
2) 运动算法的分析结果，将按照循环周期的方式执行运动计算，生成的运动信息发送至本地脉冲轴的指令值，生成了目标位置、目标速度。
3) 生成的指令值，将按照同步周期进行发送。
4) 内部芯片根据接收到的周期发送的指令值，执行对应的脉冲输出。

12.1.2 PLCOPEN 状态机

LeadStudio 运动控制系统中基于 PLCOpen 状态机对轴的状态和运动进行管理，在每一个不同状态下完成不同的功能。轴从一个状态转移到另一个状态，需要运行对应的条件，如运行 MC 指令，或外部出现了故障，用户无法对其状态进行强制，编程时一定要按照逻辑要求，运行相关的指令，轴状态转移图如下：



序号	转移条件
1	检测到轴故障时立即进入该状态
2	轴无故障且 MC_Power.Enable=FALSE 时，进入该状态
3	调用 MC_Reset 复位轴故障且 MC_Power.Status=FALSE 时，进入该状态
4	调用 MC_Reset 复位轴故障且 MC_Power.Status=TRUE，MC_Power.Enable=TRUE 时，进入该状态
5	MC_Power.Enable=TRUE 且 MC_Power.Status=TRUE 时，进入该状态
6	MC_Stop.Done=TRUE 且 MC_Stop.Execute=FALSE 时，进入该状态

图中的 MC 功能块可以使轴状态转移到指定的状态，用户应熟悉上述轴状态图的转移条件，并在编程时，正确调用相应的 MC 指令，才能使轴正确运行。用户程序中，有时需要根据轴的状态，启动后续的控制逻辑，此时依据轴状态机的判断，相比于对 MC 功能块的 done 信号判断，更为准确可靠。

轴数据结构变量 (Axis.nAxisState) 为枚举变量，用来读取轴的当前运行状态，共有如下 8 种状态：

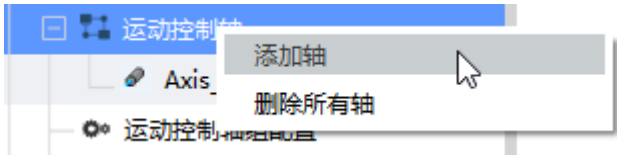
名称	值	注释
Power_off（Disabled）	0	轴未使能，需执行 MC_Power 指令
Errorstop	1	故障停止状态，需先执行 MC_Reset/MC_Power 指令
Stopping	2	停止中，等待停机操作完成
Standstill	3	使能状态，轴已停止运行
Discrete_Motion	4	轴处于离散运行状态
Continuous_Motion	5	轴处于连续运行中
Synchronized_Motion	6	轴处于同步运行中
Homing	7	轴处于回零运行中，等待归零操作执行完成

12.2 本地脉冲轴组态

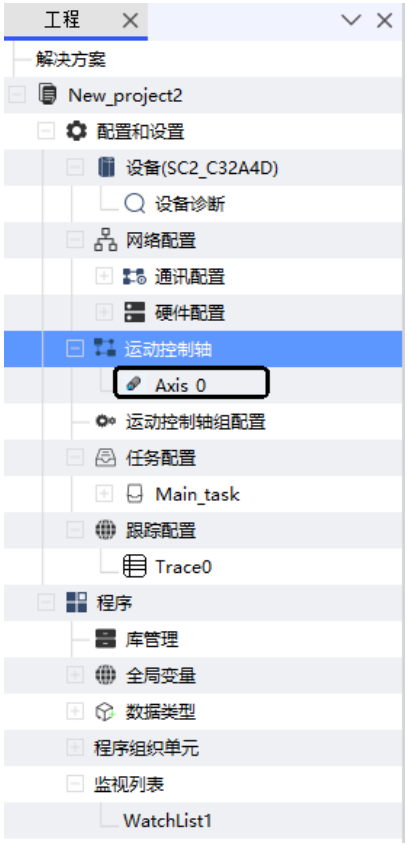
要正确控制运动控制轴，首先根据项目需要创建对应的轴，然后根据工况设定相关的轴配置参数。

添加运动控制轴的操作如下：

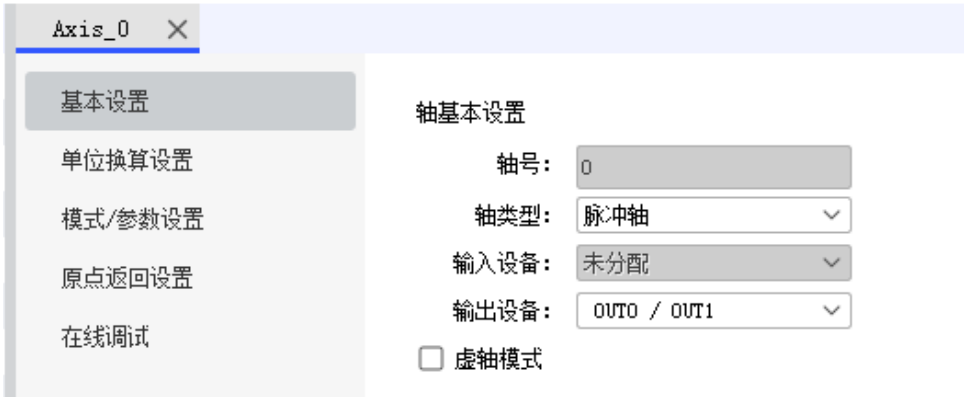
1) 右键“工程管理树”中的“运动控制轴”，选择“添加轴”；



2) 添加后的轴会出现在“运动控制轴”选项下，双击可打开对应的配置界面；

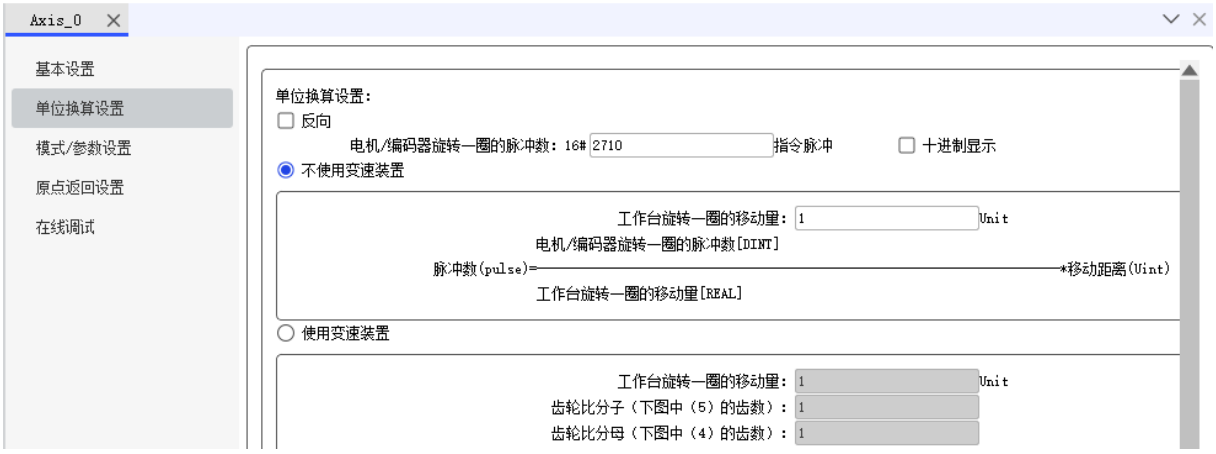


12.3 基本设置



- 轴号：每一个轴分配一个单独的编号，不可以手动修改。轴号具有唯一性。
- 轴类型：轴类型的选项有脉冲轴、编码器轴、虚轴。
- 输入设备：用于编码器轴。
- 输出设备：在脉冲轴模式下有效，SC2 支持选择 OUT0/OUT1，OUT2/OUT3，OUT4/OUT5，OUT6/OUT7

12.4 单位换算设置



- 反向：勾选后实际运行方向与控制方向相反。
- 电机/编码器旋转一圈脉冲数：根据实际电机的每转脉冲数，设定电机转 1 圈的脉冲数。
- 是否使用变速装置：指定是否使用电子齿轮比配置。
- 电机/编码器旋转一圈的移动量：在不使用变速装置时电机转 1 圈的工作台移动量。
- 工作台旋转一圈的移动量：使用变速装置时工件侧转 1 圈的移动量。

- 齿轮比分子：设定齿轮比分子。
- 齿轮比分母：设定齿轮比分母。

运动控制轴控制电机时采用脉冲单位，运动控制指令侧使用例如毫米、度、英寸等常见的度量单位，我们称之为用户单位（Unit）。

根据单位换算参数轴内部将两种单位进行相互转换。转换的模式分为以下几种情况：

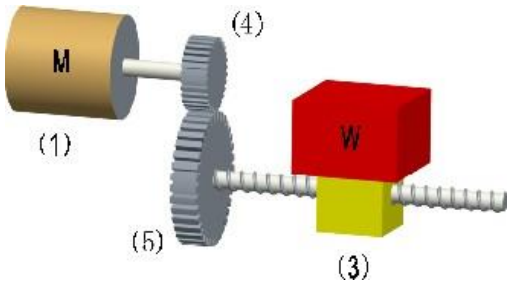
- 不使用变速装置

当不使用变速装置时，用户单位到脉冲单位的转换公式如下：

$$\text{脉冲数(pulse)} = \frac{\text{电机/编码器旋转一圈的脉冲数[DINT]}}{\text{电机/编码器旋转一圈的移动量[REAL]}} * \text{移动距离(Unit)}$$

- 使用变速装置

在线性模式下典型工况如下图所示：

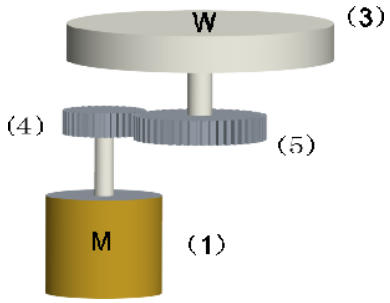


其中(1)为电机，(3)为工作台，(4)为齿轮比分子，(5)为齿轮比分母。

由用户单位到脉冲单位的计算公式如下：

$$\text{脉冲数(pulse)} = \frac{\text{电机/编码器旋转一圈的脉冲数[DINT]} * \text{齿轮比分子[DINT]}}{\text{电机/编码器旋转一圈的移动量[REAL]} * \text{齿轮比分母[DINT]}} * \text{移动距离(Unit)}$$

环形模式下典型工况如下图所示：



其中(1)为电机，(3)为工作台，(4)为齿轮比分子，(5)为齿轮比分母。

由用户单位到脉冲单位的计算公式如下：

$$脉冲数(pulse)=\frac{电机/编码器旋转一圈的脉冲数(DINT)*齿轮比分子[DINT]}{工作台旋转一圈的移动量[REAL]*齿轮比分母[DINT]}*移动距离(Unit)$$

12.5 模式/参数设置

- 编码器模式：目前只支持增量模式。
- 模式设置：支持设置线性模式、旋转模式。
- 软件限位：线性模式下有效，勾选“使能”复选框后，软件正负向限制值有效。
- 周期设置：旋转模式下有效，设置旋转模式下的旋转周期。
- 软件出错响应：

  限位减速度：如果软限位有效，轴按照该减速度减速停止；

  轴故障减速度：在轴运行期间如果运动指令出现故障导致轴必须切换到 errorstop 状态，则轴将按照该减速度减速停止。
- 轴速度限制：

  最大速度：运动控制轴运行的最大速度限制；

  最大加速度：运动控制轴运行的最大加速度限制；

- 输入信号设置：

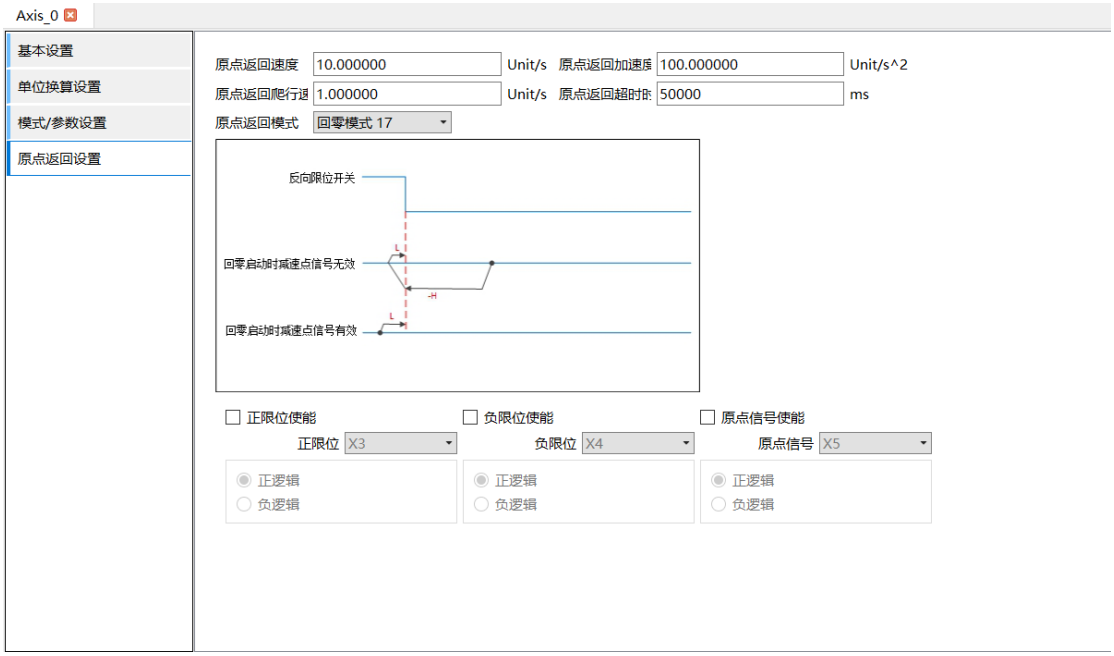
探针 1 使能：勾选后探针 1 有效，探针输入端子可在下拉菜单处配置，支持选择 IN0-IN7；

- 输出信号设置：

输出方式：配置脉冲输出方式，支持脉冲方向、单相脉冲；

输出端：配置脉冲方向时，该选项与基本设置界面设置相同，配置单相脉冲时，只占用脉冲输出端口。

12.6 原点返回设置



SC2 支持 CIA402 协议中规定的 17-35 号回原方式

原点返回设置配置如下

- 原点返回速度：设置原点返回速度，即回原时的高速阶段；
- 原点返回爬行速度：设置原点返回爬行速度，即回原时的低速阶段；
- 原点返回加速度：设置原点返回时的加速度；
- 原点返回超时时间：设置原点返回超时时间，超过该超时时间仍没有回原完成则回原错误。
- 正限位使能、负限位使能、原点使能：勾选后对应的信号有效；
- 正限位、负限位、原点信号：选择信号对应的输入引脚；

- 正逻辑、负逻辑：设置限位的逻辑，正逻辑时输入信号为 ON，限位触发，负逻辑时输入信号为 OFF，限位触发。

12.7 默认加减速、起跳速度

LeadStudioV2.7/V3.1 及以上版本支持设置轴的默认加减速、起跳速度

1.轴默认加减速时间

通过轴结构体 dwAccTime 和 dwDecTime 可分别设置轴的默认【加速时间】和【减速时间】，单位为 ms。

当轴运动指令的加减速参数设定为 0 时，轴将以默认的加减速度去执行轴运动；轴默认加减速度= 轴最大速度 / 轴默认加减速时间。

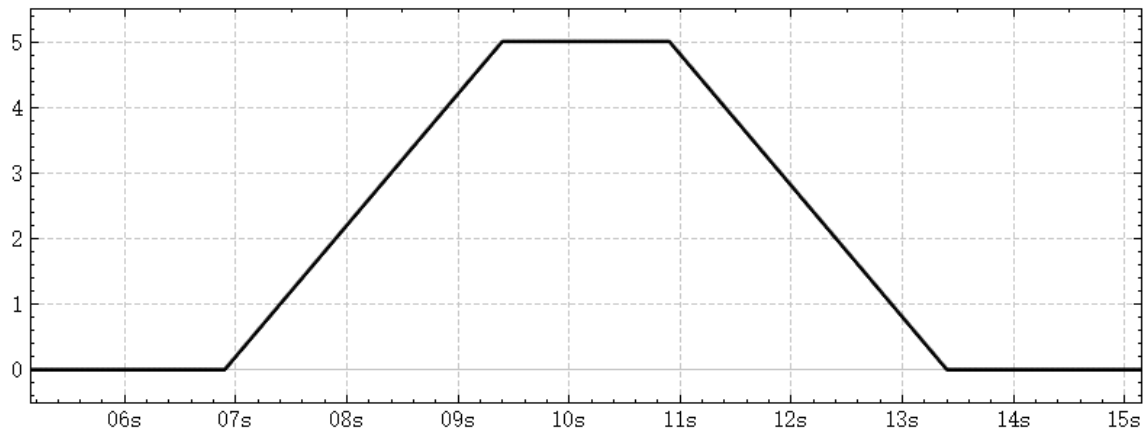
注：轴最大速度可在轴模式/参数界面中设置，见章节 12.5。

2.轴的起始速度和结束速度

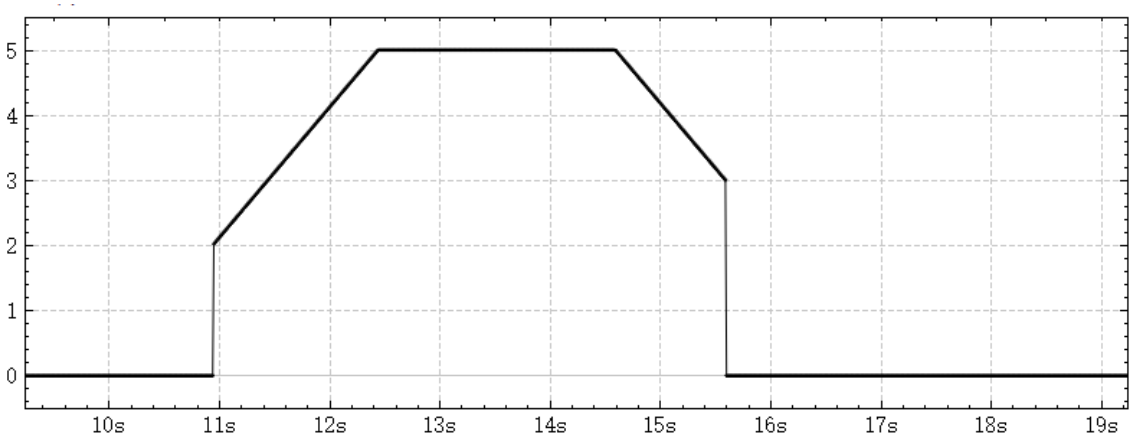
通过轴结构体 fStartVelocity 和 fEndVelocity 可分别设置轴的【起始速度】与【结束速度】，用户单位/s。

【起始速度】与【结束速度】是指轴运动指令开始执行时的轴起始速度与指令执行结束时的轴结束速度。其中，起始速度对位置指令和速度指令生效，而结束速度只对位置指令生效

【起始速度】与【结束速度】都为 0 时的，执行轴位移指令时的速度曲线



【起始速度】与【结束速度】都不为 0 时的，执行轴位移指令时的速度曲线



12.8 支持的运动指令

SC2 支持的运动指令参考下表，指令具体介绍可参考指令手册：

指令类别	名称	FB/FC	功能
单轴运动	MC_Power	FB	轴使能
	MC_Home	FB	轴回零
	MC_Stop	FB	轴停止
	MC_Halt	FB	轴暂停
	MC_MoveAbsolute	FB	绝对运动
	MC_MoveRelative	FB	相对运动
	MC_MoveAdditive	FB	附加运动
	MC_MoveSuperImposed	FB	叠加运动
	MC_HaltSuperImposed	FB	暂停叠加运动
	MC_MoveVelocity	FB	定速运动
	MC_MoveContinuousAbsolute	FB	指定结束速度的绝对运动
	MC_MoveContinuousRelative	FB	指定结束速度的相对运动

	MC_PositionProfile	FB	位置规划
	MC_VelocityProfile	FB	速度规划
	MC_SetPosition	FB	设置位置
	MC_SetOverride	FB	速度调节指令
	MC_ReadParameter	FB	读轴参数
	MC_ReadBoolParameter	FB	读轴布尔参数
	MC_WriteParameter	FB	写轴参数
	MC_WriteBoolParameter	FB	写轴布尔参数
	MC_SetAxisConfigPara	FB	设置轴配置参数
	MC_ReadActualPosition	FB	读轴实际位置
	MC_ReadActualVelocity	FB	读轴实际速度
	MC_ReadStatus	FB	读轴状态
	MC_ReadMotionState	FB	读轴运动状态
	MC_ReadAxisInfo	FB	读轴信息
	MC_ReadAxisError	FB	读轴错误
	MC_TouchProbe	FB	探针指令
	MC_AbortTrigger	FB	打断探针功能
	MC_Reset	FB	轴复位
	MC_MoveFeed	FB	中断定长指令
	MC_Jog	FB	轴点动
	MC_Inch	FB	轴寸动

12.9 轴结构体

SC2 的轴结构体参考下表：

结构体	成员	数据类型	RW	功能
AXIS_REF	instance	POINTER TO INT	/	轴实例，不可操作
	nAxisState	MC_AXIS_STATE	R	轴状态
	nAxisType	MC_AXIS_TYPE	R	轴类型
	bCommunication	BOOL	R	通讯状态
	fSetPosition	LREAL	R	轴设置位置，用户单位
	fSetVelocity	LREAL	R	轴设置速度，用户单位
	fSetAcceleration	LREAL	R	轴设置加速度，用户单位
	fSetTorque	LREAL	R	轴设置力矩，用户单位
	fActPosition	LREAL	R	轴实际位置，用户单位
	fActVelocity	LREAL	R	轴实际速度，用户单位
	fActAcceleration	LREAL	R	轴实际加速度，用户单位
	fActTorque	LREAL	R	轴实际力矩，用户单位
	diSetPosition	DINT	R	轴设置位置，脉冲单位
	diSetVelocity	DINT	R	轴设置速度，脉冲单位
	diSetAcceleration	DINT	R	轴设置加速度，脉冲单位
	diSetTorque	DINT	R	轴设置力矩，脉冲单位
	diActPosition	DINT	R	轴实际位置，脉冲单位
	diActVelocity	DINT	R	轴实际速度，脉冲单位
	diActAcceleration	DINT	R	轴实际加速度，脉冲单位
	diActTorque	DINT	R	轴实际力矩，0.1%

	wAxisError	WORD	R	轴错误
	wDriverError	WORD	R	驱动器错误
	fUnits	LREAL	R	综合齿轮比
	bHWplimit	BOOL	R	硬件正限位信号
	bHWnlimit	BOOL	R	硬件负限位信号
	bHome	BOOL	R	硬件原点信号
	bSWplimit	BOOL	R	软件正限位信号
	bSWnlimit	BOOL	R	软件负限位信号
	dilncrements	DINT	RW	电机转一圈脉冲数，脉冲单位
	fDistanceOfWorkBench	REAL	RW	电机转一圈工作台移动量，用户单位
	diNumerator	DINT	RW	齿轮比分子
	diDenominator	DINT	RW	齿轮比分母
	bInvertDirection	BOOL	RW	轴反向
	bVirtual	BOOL	RW	虚轴模式
	fSWLimitEnable	LREAL	RW	软限位使能
	fSWLimitPositive	LREAL	RW	正向软件限位
	fSWLimitNegative	LREAL	RW	负向软件限位
	nHomeMethod	INT	RW	原点回归模式（脉冲轴）
	fHomeVelocityFast	LREAL	RW	原点回归返回速度（脉冲轴）
	fHomeVelocitySlow	LREAL	RW	原点回归接近速度（脉冲轴）

	fHomeAcceleration	LREAL	RW	原点回归返回加速度（脉冲轴）
	diHomingTimeOut	DINT	RW	原点返回超时时间（脉冲轴），毫秒
	bDebugMode	BOOL	R	调试模式开关
	iDirection	INT	R	速度方向，-1-负向/0-停止/1-正向
	fFactorVel	LREAL	R	速度倍率
	fFactorAcc	LREAL	R	加速度倍率
	fFactorJerk	LREAL	R	跃度倍率
	fFactorTor	LREAL	R	力矩倍率
	iMovementType	INT	R	轴动作类型，0-旋转/1-直线
	fPositionPeriod	LREAL	R	旋转轴的周期，单位 Unit
	wImmediateStopErrCode	WORD	R	急停指令错误码
	bImmediateStopErr	BOOL	R	急停指令出错
	bImmediateStopBusy	BOOL	R	急停指令进行中
	bImmediateStopDone	BOOL	R	急停指令完成
	dwOwnerFBId	DWORD	R	轴指令的占用 FB 的 ID
	dwAccTime	DWORD	RW	轴默认加速时间，单位：ms
	dwDecTime	DWORD	RW	轴默认减速时间，单位：ms
	fStartVelocity	LREAL	RW	轴启动速度，用户单位/s

	fEndVelocity	LREAL	RW	轴结束速度，用户单位/s

13 电子凸轮

13.1 电子凸轮概述

电子凸轮是在机械凸轮的基础上发展起来的。从本质上讲，从动件的运动是一种计算机程序处理的结果。因此，凸轮机构可以看成一种函数关系，凸轮的转动作为函数的输入，而从动件的位移作为函数的输出。

相应地，电子凸轮设计者的目标就是利用计算机技术建立程序，实现凸轮从动件的运动轨迹。

电子凸轮相对于机械凸轮的优点：

1）轮廓更易设计，并且支持任意凸轮轮廓及轮廓拼接；避免了凸轮机构磨损，柔性、精度更高；

2）可动态生成和修改轮廓，在运行中通过修改关键点进行轮廓的动态修改；

3）可以设置若干个凸轮挺杆输出值；

4）支持虚拟轴、相位偏移等操作。

13.1.1 相关功能块

名称	FB/FC	功能
MC_CamIn	FB	电子凸轮耦合
MC_CamOut	FB	电子凸轮脱离
MC_GenerateCamTable	FB	更新凸轮表
MC_SaveCamTable	FB	保存凸轮表
MC_GetCamTablePhase	FB	获取主轴相位
MC_GetCamTableDistance	FB	获取从轴位移
MC_Phasing	FB	主轴相位偏移
MC_DigitalCamSwitch	FB	挺杆指令

13.1.2 相关数据结构

凸轮表结构体

CAM_TABLE_REF		
变量	数据类型	备注
wPoints	WORD	凸轮表关键点个数
sCamNode	Array[1..360] of Cam_Node	凸轮关键点数据

凸轮关键点结构体

Cam_Node		
变量	数据类型	备注
fPhase	LREAL	主轴相位
fDistance	LREAL	从轴位置
fVelocity	LREAL	连接点速度
fAcceleration	LREAL	连接点加速度
wCurve	WORD	曲线类型 0-NA，1-直线，5-5 次曲线
align	Array[1..3] of WORD	保留

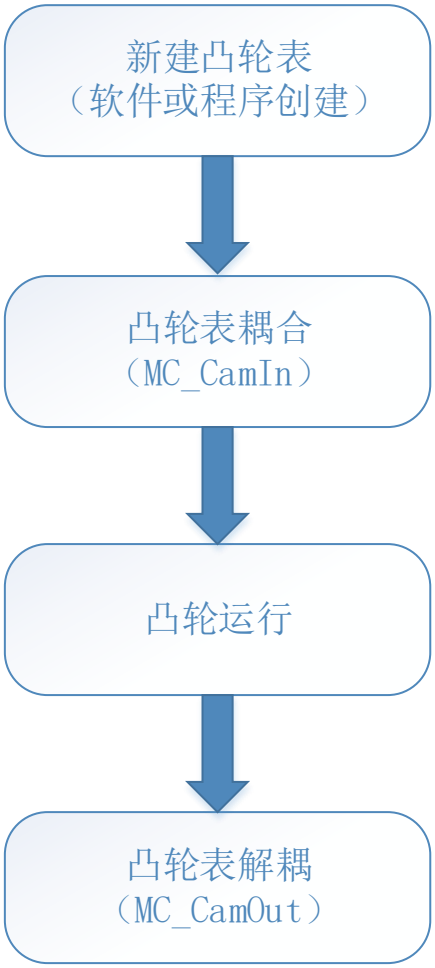
挺杆结构体

Digital_Switch		
变量	数据类型	备注
fPosition	LREAL	主轴相位位置
fParameter	LREAL	挺杆参数值
iMode	INT	挺杆模式：0-禁用，1- fParameter 表示位置，达到位置 OFF，2- fParameter 表示时间（ms），达到时间 OFF
iDirection	INT	挺杆方向：0-主轴

		运动正向，1-主轴运动负向

13.2 电子凸轮使用流程

下图描述了 LeadStudio 软件中电子凸轮的典型使用流程。



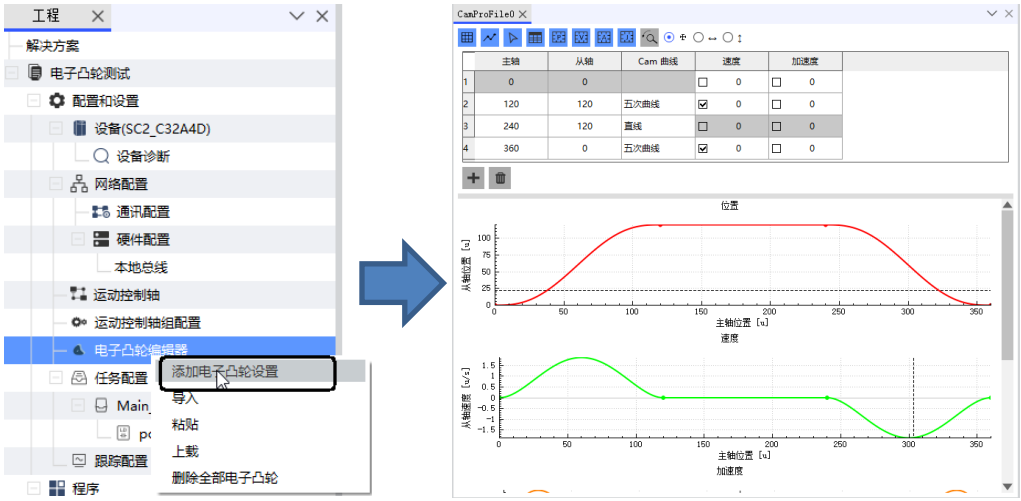
注：

- 在调用 MC_CamIn 前，需要先对轴进行使能操作；
- 调用 MC_CamOut 时，如果从轴状态机没有在 synchronized_motion 状态，执行后会报错；

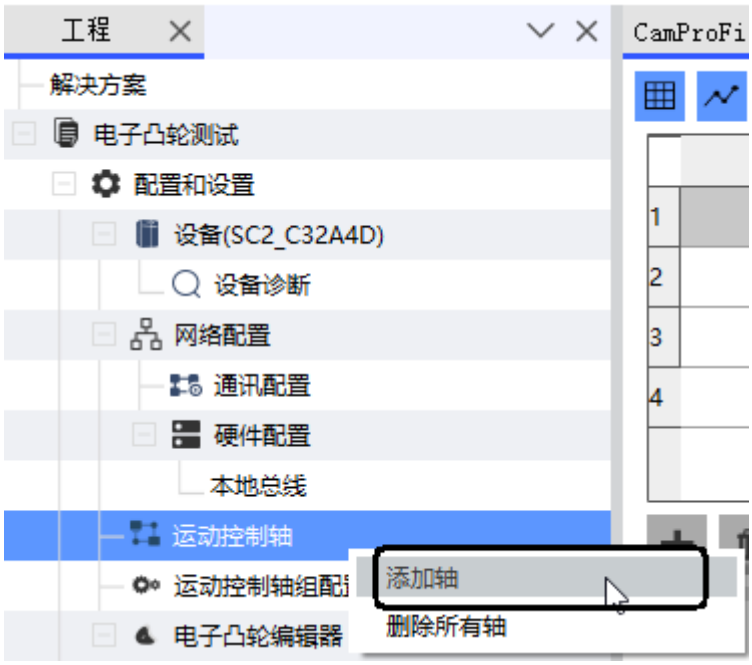
13.2.1 电子凸轮程序使用步骤

本节通过举例，简要说明如何使用电子凸轮功能，具体实现功能为：触发主轴运行速度指令，从轴跟随主轴实现凸轮运动。操作步骤如下：

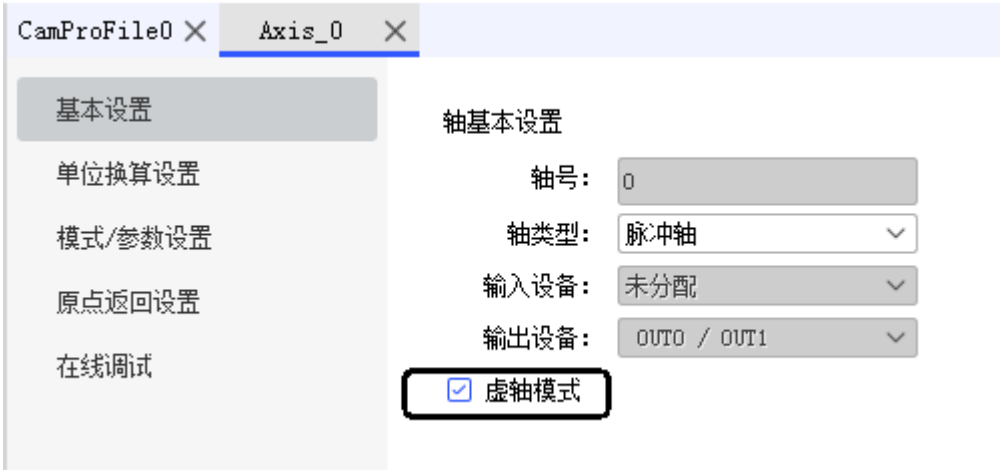
1）右键点击“电子凸轮编辑器”选择“添加电子凸轮设置”创建凸轮表，并根据需要修改凸轮表曲线。



2）添加运动控制轴（此处以虚轴为例），右键点击“运动控制轴”选择“添加轴”添加运动控制轴。



3）在轴基本设置界面勾选“虚轴模式”，将轴设置为虚轴



4）重复 3）操作，再次添加一个虚轴作为从轴使用

5）实例化功能块，声明相关变量。

VAR

MC_Power1:MC_Power;

MC_Power2:MC_Power;

MC_CamIn_inst0:MC_CamIn;

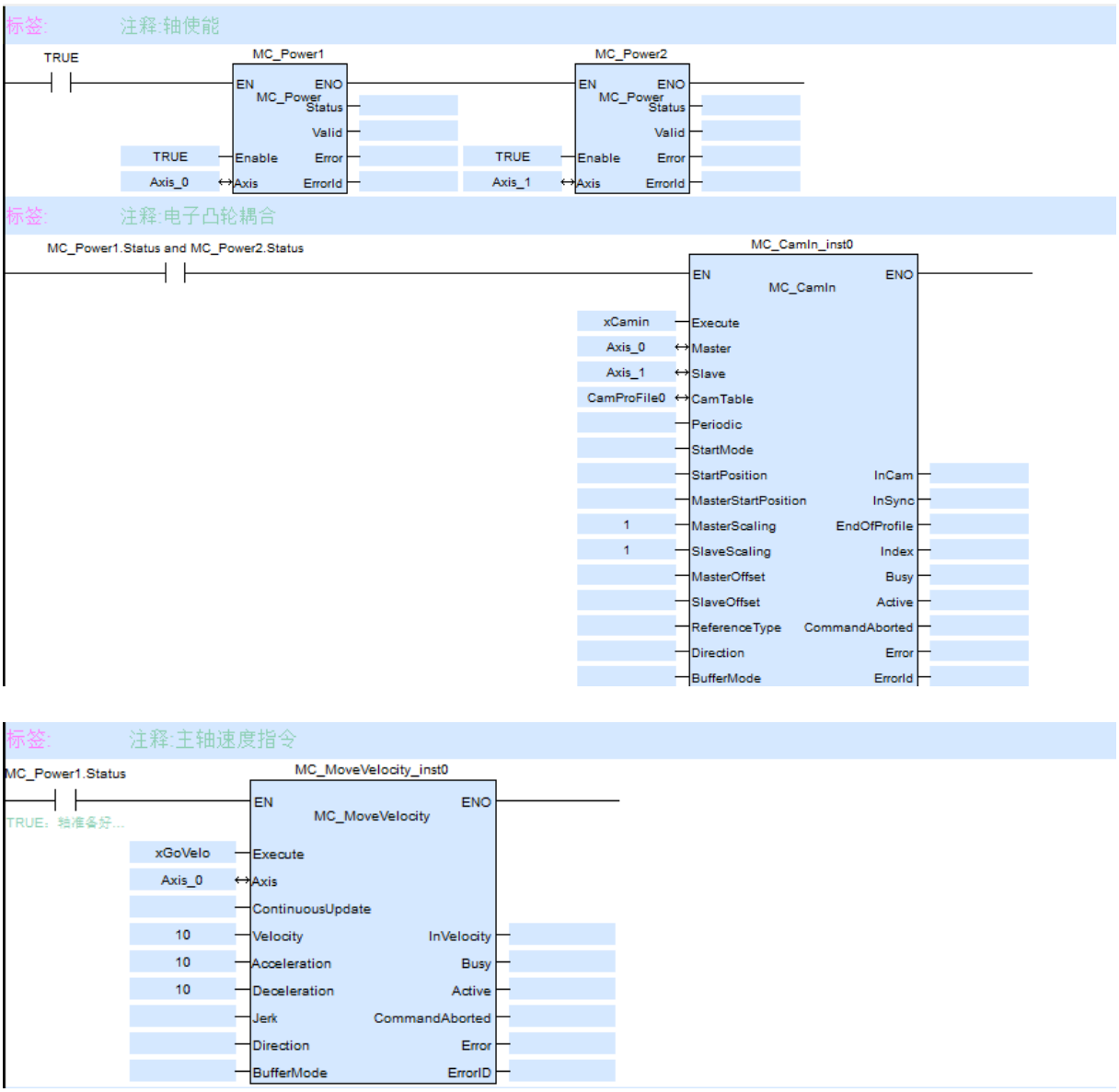
xCamin:BOOL;

MC_MoveVelocity_inst0:MC_MoveVelocity;

xGoVelo:BOOL;

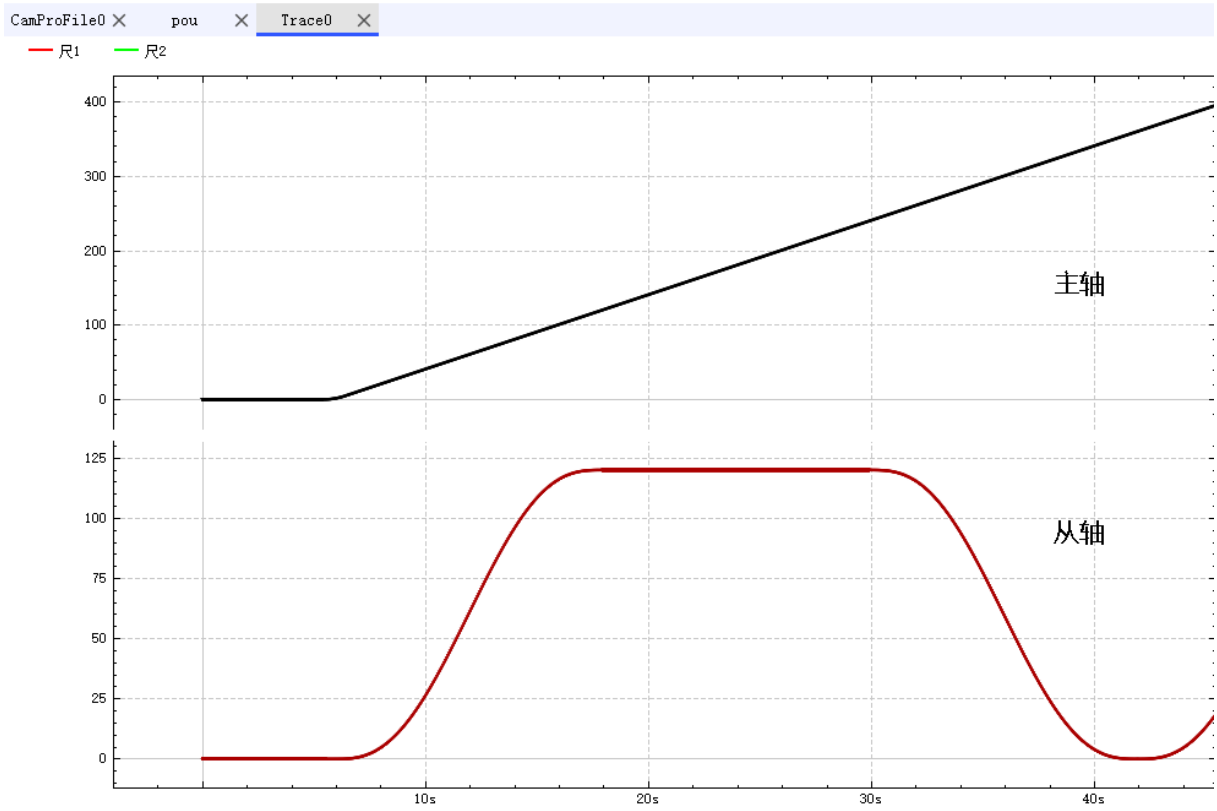
END_VAR

6）编写程序。



5）触发电子凸轮执行

触发 xCamIn 及 xGoVel 后，电子凸轮开始执行，运行曲线如下：



13.3 凸轮表

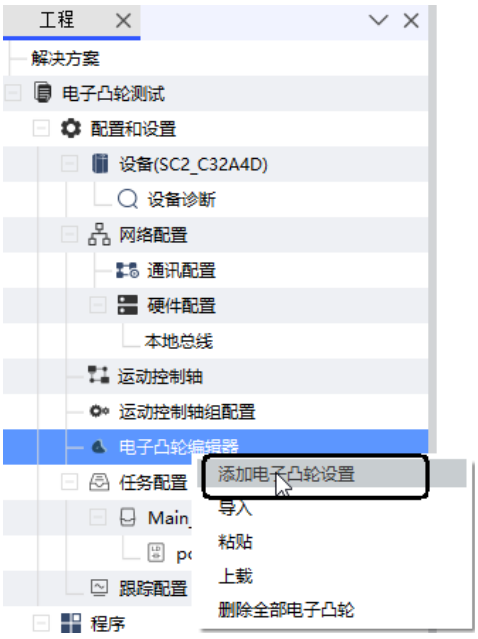
在凸轮功能中，将由主轴相位和从轴位移组成的一对数据称之为凸轮关键点，凸轮表即为多个凸轮关键点数据组成的数据组合。

凸轮动作中，根据主轴相位和设定的曲线类型计算出从轴的位移，从而控制从轴的动作。

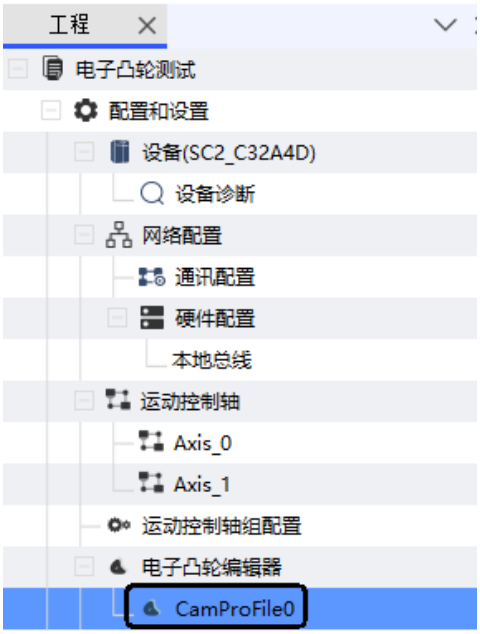
13.3.1 通过软件编辑凸轮表

LeadStudio 软件支持直接通过软件创建凸轮表，具体的操作步骤如下：

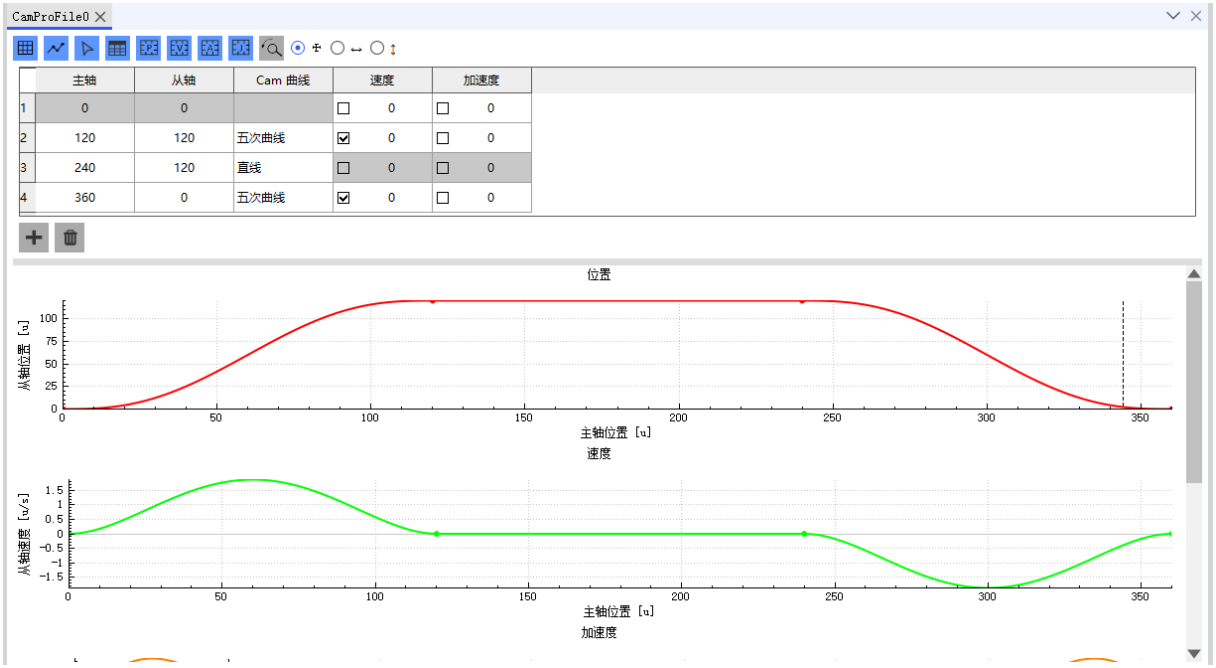
1）右键点击“电子凸轮编辑器”选择“添加电子凸轮设置”创建凸轮表.



2）双击 “CamProFile0”，打开凸轮表配置界面。



3）凸轮表配置页面用于显示凸轮表的位置曲线、速度曲线、加速度曲线、加加速度曲线，并且用户可以对关键点进行拖动，修改关键点的数据；



上方工具栏中各按钮对应的功能参考下表：

图标	功能
	隐藏/显示坐标轴中的网格（蓝色为显示）
	隐藏/显示曲线中的关键点标记（蓝色为显示）
	隐藏/显示坐标轴中的光标（蓝色为显示）
	隐藏/显示凸轮表编辑表格（蓝色为显示）
	隐藏/显示位置坐标图（蓝色为显示）
	隐藏/显示速度坐标图（蓝色为显示）
	隐藏/显示加速度坐标图（蓝色为显示）
	隐藏/显示加加速度坐标图（蓝色为显示）
	重置视图，单击触发
	自由拖拽，选中后可在各坐标图中自由拖拽关键点
	纵向锁定，选中后可在各坐标图中横向拖拽关键

	点
	横向锁定，选中后可在各坐标图中纵向拖拽关键点

凸轮关键点表格中各选项对应的功能参考下表：

选项	功能
	删除关键点
	插入关键点
主轴	主轴相位
从轴	从轴位移
Cam 曲线	凸轮曲线类型
速度	连接点速度比
加速度	连接点加速度比

注：

1、直线不支持修改连接点速度比、加速度比。

2、5 次曲线连接点速度比、加速度比，默认自动计算，速度比、加速度比勾选选框后可以手动修改，如直接通过坐标图拖动，也会勾选对应的选框及修改数值。

13.3.2 通过程序直接编辑凸轮表

本节讲述两种通过程序编辑凸轮表的情况，分别为通过软件生成凸轮表，在程序中修改关键点数据，以及直接通过程序生成凸轮表：

1） 通过软件生成凸轮表，在程序中修改关键点数据；

通过软件生成 CAM 表时，LeadStudio 会自动生成与变量表同名的 CAM_TABLE_REF 类型凸轮表数据其中包含 ARRAY[1..360] OF CAM_NODE 类型的凸轮表关键点数据 sCamNode，例：

上图新建凸轮表“CamProfile0”，则对应生成的凸轮表数据“CamProfile0”及凸轮表关键点数组 “CamProfile0.sCamNode”，用户可以直接在工程中修改。

如修改凸轮表主轴长度为 360，修改凸轮表关键点个数为 4，,修改第 3 个关键点位主轴相位 200，从轴位移 200，对应程序如下：

CamProfile0.fMaxPhase :=360;

CamProfile0.wPoints :=4;

CamProfile0.sCamNode[3].fPhase:=200

CamProfile0.sCamNode[3].fDistance:=200;

注：

1、凸轮表数据可以随时修改，但是正在运行的凸轮表只能通过重启动 MC_GenerateCamTable 或 MC_Camin 进行参数更新或凸轮表替换

2、凸轮表修改后，若重新上电，会恢复到修改之前的数据，如需要保存凸轮表的修改到掉电保持区域，需要调用 MC_SaveCamTable。

2） 直接通过程序生成凸轮表；

直接通过程序生成凸轮表需要三个步骤，下面举例说明：

第一步定义变量。

VAR

Cam : CAM_TABLE_REF;

END_ VAR;

第二步根据需求设置关键点并给 Cam 中的 sCamNode 参数赋值。

Cam.sCamNode[1].fPhase := 0; //设置第一个关键点的主轴相位

Cam.sCamNode[1].fDistance :=0; //设置第一个关键点的从轴位移

Cam.sCamNode[1].fVelocity := 1; //设置第一个关键点的连接点速度比

Cam.sCamNode[1].fAcceleration := 0; //设置第一个关键点的连接点加速度比

Cam.sCamNode[1].wCurve:= 5; //设置第一个关键点曲线类型为五次曲线

Cam.sCamNode[2].fPhase := 100; //设置第二个关键点的主轴相位

Cam.sCamNode[2].fDistance := 100; //设置第二个关键点的从轴位移

Cam.sCamNode[2].fVelocity := 1; //设置第二个关键点的连接点速度比

Cam.sCamNode[2].fAcceleration := 0; //设置第二个关键点的连接点加速度比

Cam.sCamNode[2].wCurve:= 5; //设置第二个关键点曲线类型为五次曲线

Cam.sCamNode[3].fPhase := 200; //设置第三个关键点的主轴相位

Cam.sCamNode[3].fDistance := 200; //设置第三个关键点的从轴位移

Cam.sCamNode[3].fVelocity := 1; //设置第三个关键点的连接点速度比

Cam.sCamNode[3].fAcceleration := 0; //设置第三个关键点的连接点加速度比

Cam.sCamNode[3].wCurve:= 5; //设置第三个关键点曲线类型为五次曲线

Cam.sCamNode[4].fPhase := 360; //设置第四个关键点的主轴相位

Cam.sCamNode[4].fDistance := 360; //设置第四个关键点的从轴位移

Cam.sCamNode[4].fVelocity := 1; //设置第四个关键点的连接点速度比

Cam.sCamNode[4].fAcceleration := 0; //设置第四个关键点的连接点加速度比

Cam.sCamNode[4].wCurve:= 5; //设置第四个关键点曲线类型为五次曲线

第三步设置凸轮属性

Cam.wPoints := 4; //设置关键点个数为 4 个关键点

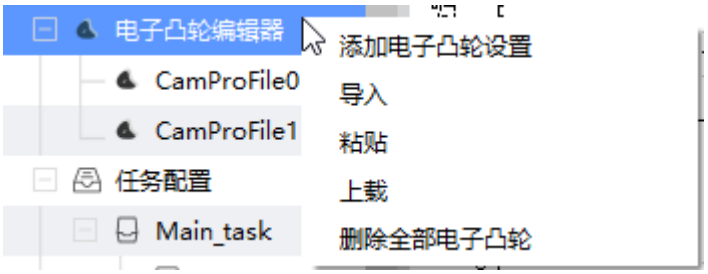
注：

1、凸轮表数据如需要掉电保持，请将 CAM_TABLE_REF 类型的凸轮表变量声明为保持型变量。

13.3.3 凸轮表上载

PLC 中的凸轮表，支持通过上载的方式上传到 LeadStudio 中

右键“电子凸轮编辑器”，点击“上载”选项，即可完成凸轮表上载操作。

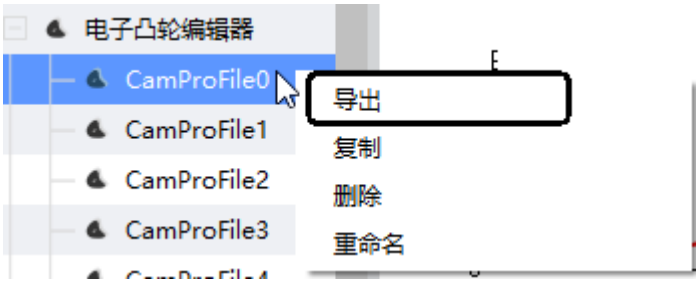


上载后的凸轮表会以 CamProFile+序号的方式进行命名，目前，只支持通过软件创建的凸轮表进行上传，通过程序创建的凸轮表暂时无法被上传。

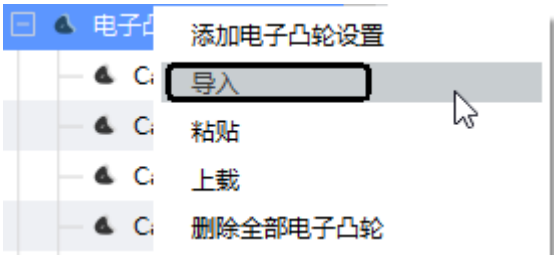
13.3.4 凸轮表导入导出

LeadStudio 中的凸轮表，支持导出成 CSV 类型的文件，同时支持符合条件的文件导入到凸轮表中。

导出操作：右键需要导出的凸轮表，在右键菜单中点击“导出”选项，设置好文件名及保存路径后，即可完成导出。



导入操作：右键 “电子凸轮编辑器”，在右键菜单中点击“导入”选项，选择需要导入的文件，即可完成导入。



14 高速计数器

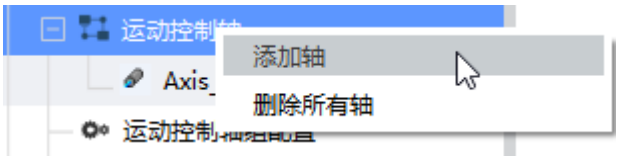
14.1 高速计数器轴简介

计数器在 LeadStudio 软件和工程应用中以编码器轴形式实现管理应用，计数器与轴关联后统称为计数器轴。SC 系列 PLC 支持最大 8 轴 32 位高速计数器，可实现 AB 相 1/2/4 倍频、CW/CCW、脉冲+方向和单相计数，计数信号源可选择外部脉冲输入或内部 1ms/1us 时钟计数；配合其他输入信号，可实现计数器的预置和锁存功能。

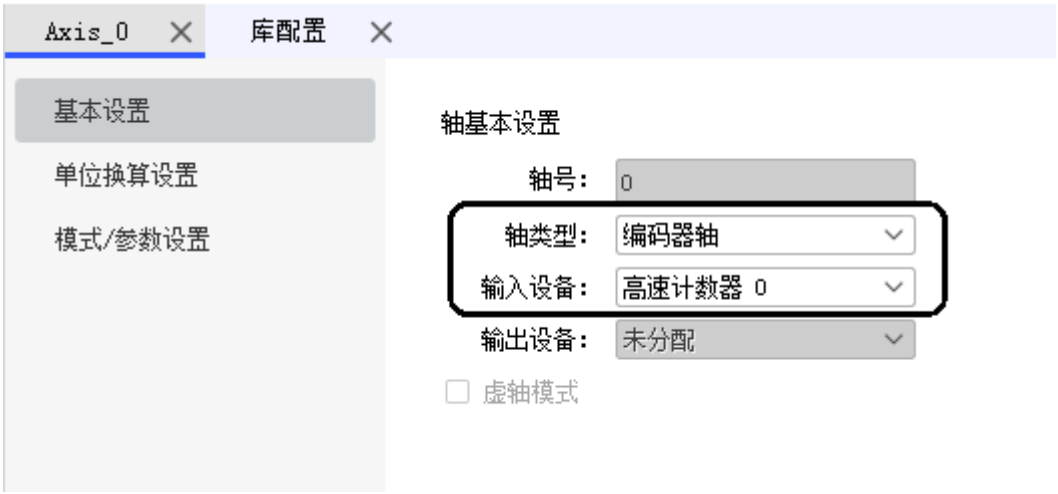
14.2 创建计数器轴

在 LeadStudio 编程软件中使用计数器前，需要先将计数器和轴关联。

在“工程管理”栏，右键单击运动控制轴，选择“添加轴”，创建一个运动控制轴。

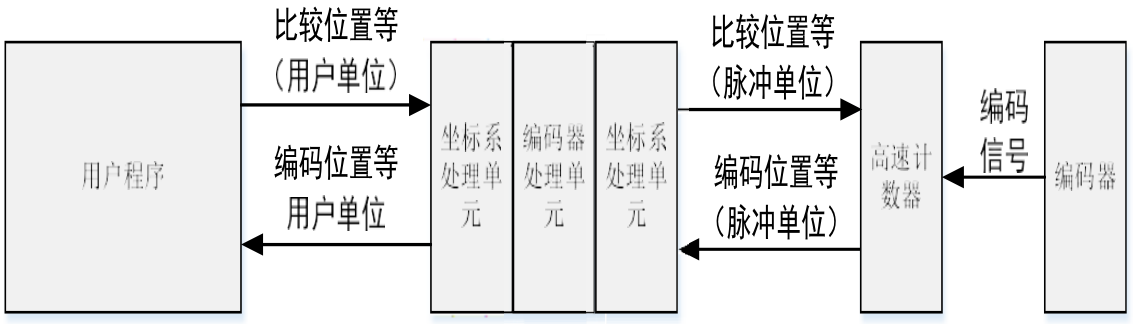


双击新添加的轴（如图 Axis_0）打开设置页面，在“基本设置”界面选择“本地编码器轴”作为轴类型， 选择“高速计数器”作为输入设备，即可将轴和计数器关联。轴号作为轴标识在程序中使用，实现对应计数器轴的控制。

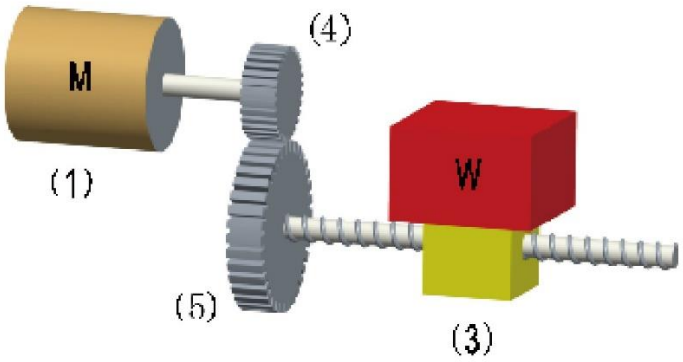


14.3 计数器轴用户单位与换算

高速计数器对编码器信号解码时采用脉冲单位，计数器指令则使用例如毫米、度、英寸等常见的度量单位，称之为用户单位（Unit）。通过单位换算可将脉冲数转换为为用户单位（Unit），用户单位（Unit）根据实际应用可定义为设备相关单位（毫米、转等）。



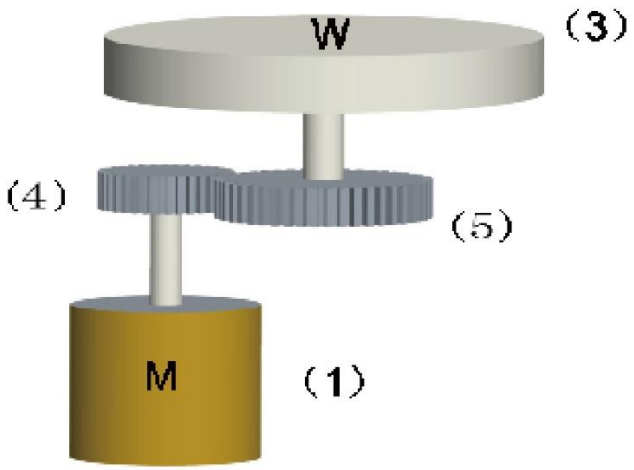
M：电机/编码器，W：工作台



轴类型为旋转模式：

$$\text{脉冲数(pulse)} = \frac{\text{电机/编码器旋转一圈的脉冲数(DINT)} * \text{齿轮比分子[DINT]}}{\text{工作台旋转一圈的移动里[REAL]} * \text{齿轮比分母[DINT]}} * \text{移动距离(Unit)}$$

M：电机/编码器，W：工作台



单位换算设置中，需要根据实际设备设置相应参数。

1) 电机/编码器旋转一圈的脉冲数设置：输入框内16#表示十六进制数，如编码器旋转一圈的脉冲数为10000个脉冲，对应十六进制值为2710，则输入16#2710。

2) 工作行程设置：工作行程设置可以不使用变速装置或使用变速装置。

当不使用变速装置时，用户单位到脉冲单位的转换公式如下：

$$脉冲数(pulse)=\frac{电机/编码器旋转一圈的脉冲数[DINT]}{工作台旋转一圈的移动量[REAL]}*移动距离(Unit)$$

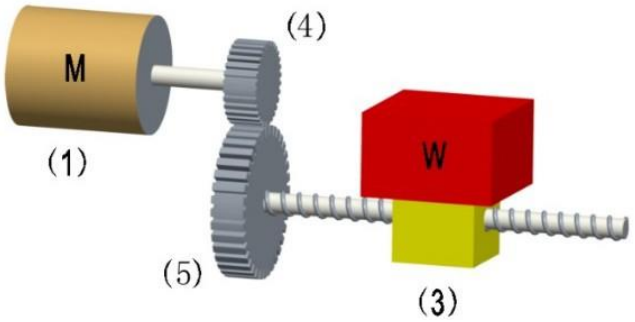
如编码器旋转一圈对应的工作轴旋转一圈，用户单位（Unit）为转，则电机/编码器旋转一圈的工作行程设置为 1 即可。

以雷赛 20 位编码器为例，设定参数如下： 电机/编码器旋转一圈脉冲数 = 1048576

电机/编码器旋转一圈的移动量 = 1，则当相对定位指令给定的目标位移为 10 时，运动控制轴实际发送的脉冲量为 10485760，此时电机旋转 10 圈。

- 使用变速装置

在线性模式下典型工况如下图所示：



其中(1)为伺服电机，(3)为工件，(4)为齿轮比分子，(5)为齿轮比分母。 由用户单位到脉冲单位的计算公式如下：

$$脉冲数(pulse)=\frac{电机/编码器旋转一圈的脉冲数[DINT]*齿轮比分子[DINT]}{工作台旋转一圈的移动量[REAL]*齿轮比分母[DINT]}*移动距离(Unit)$$

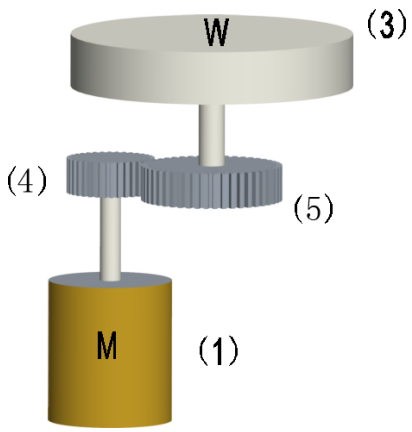
工作台旋转一圈的移动量: 1.000000 Unit

齿轮比分子(下图中(5)的齿数): 1

齿轮比分母(下图中(4)的齿数): 1

如伺服电机通过减速机连接丝杆带动工作台运动，丝杆旋转一圈的工作行程为 5mm，减速比为 20:10， 设置如下：

旋转（环形）模式下典型工况如下图所示：



其中(1)为伺服电机，(3)为工件，(4)为齿轮比分子，(5)为齿轮比分母。

由用户单位到脉冲单位的计算公式如下：

$$脉冲数(pulse)=\frac{电机/编码器旋转一圈的脉冲数(DINT)*齿轮比分子[DINT]}{工作台旋转一圈的移动量[REAL]*齿轮比分母[DINT]}*移动距离(Unit)$$

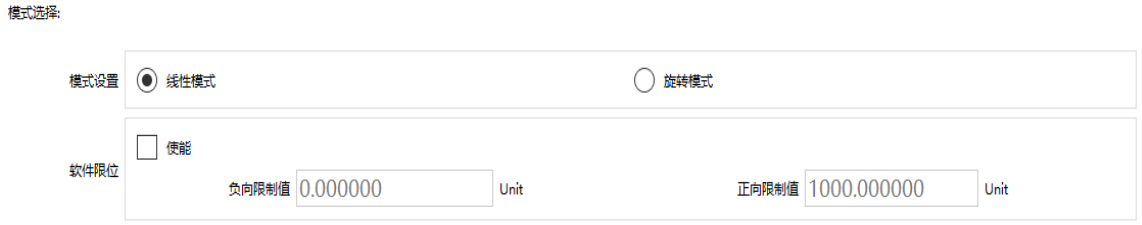
14.4 设置工作模式

14.4.1 线性模式

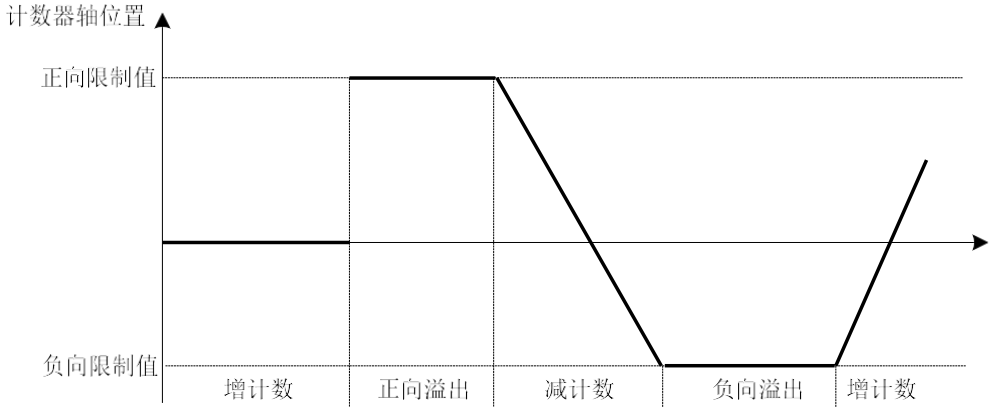
计数器轴的位置在负向限制值和正向限制值之间变化，计数器轴的位置达到限制值后，继续输入同向脉冲，计数器轴报溢出，同时计数器轴位置保持不变。计数器轴报溢出后，输入反向脉冲，计数器轴反向计数，溢出错误清除。

线性模式下，可以在界面中设置计数器轴的负向和正向位置限制值，位置单位为用户单位（Unit）。负向限制值必须小于或等于 0，正向限制值必须大于或等于 0。由于高速计数器为 32 位计数器，负向限制值与正向限制值换算为脉冲单位后必需在 32 位整数范围[-2147483648, 2147483647]。

在线性模式下，高速计数器在[负向限制值，正向限制值]的区间内工作。当方向为负向时，计数值向负方向减小，到达负向限制值后，计数值不再减小；当方向为正向时，计数值向正方向增加，到达正向限制值后，计数值不再增加。



在线性模式下，需要先打开使能开关，勾选“使能”之后，才能够正常使用负向限制和正向限制功能。使能开关默认关闭。

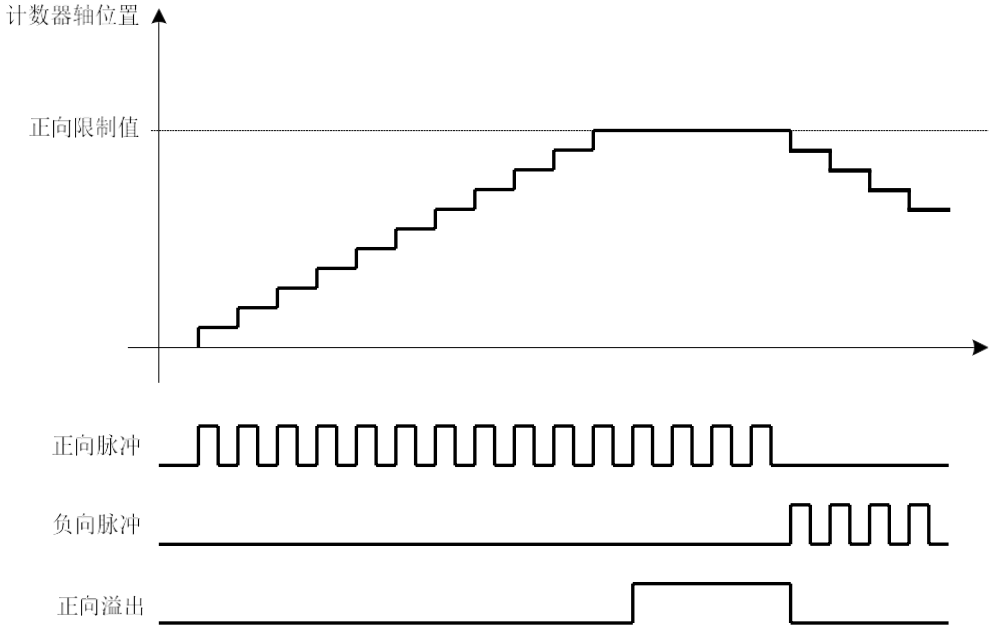


线性模式

正向脉冲计数

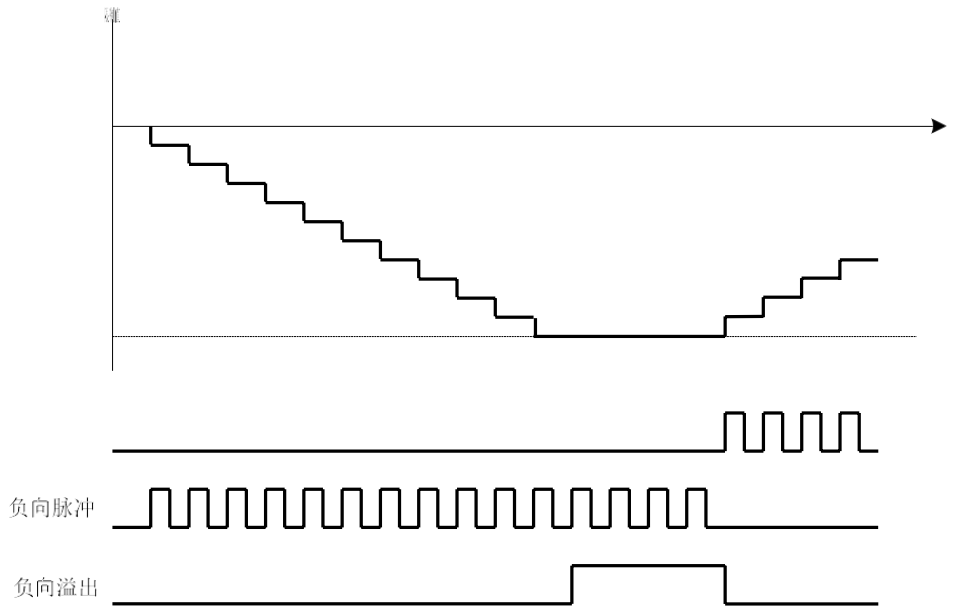
线性模式下，输入正向脉冲，计数器轴位置增计数达到限制值后，继续输入正向脉

冲，计数器轴报正向溢出错误，计数器轴位置值保持不变。输入负向脉冲，计数器轴位置减计数，正向溢出错误清除。



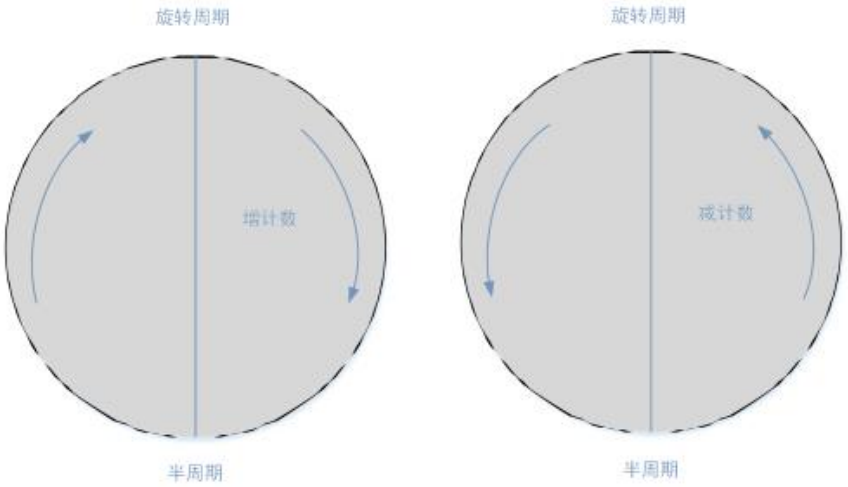
负向脉冲计数

线性模式下，输入负向脉冲，计数器轴位置减计数达到限制值后，继续输入负向脉冲，计数器轴报负向溢出错误，计数器轴位置值保持不变。输入正向脉冲，计数器轴位置增计数，负向溢出错误清除。



14.4.2 旋转模式

计数器轴的位置在旋转周期内循环变化，增计数时计数器轴的位置达到旋转周期最大值后变为 0，减计数时计数器轴的位置为 0 后，从旋转周期最大值往下减。旋转模式下，可以在界面中设置计数器轴的旋转周期，周期单位为用户单位（Unit）。由于高速计数器为 32 位计数器，旋转周期换算为脉冲单位后必需在 32 位整数范围[-2147483648, 2147483647]。



14.5 设置计数器参数

14.5.1 概述

参数设置主要包括计数模式、探针、预置、比较输出功能。

14.5.2 计数模式

14.5.2.1 概述

本地编码器轴支持多种信号计数模式，脉冲+方向、A/B 相（1/2/4 倍频）、单相计数。

信号源：根据不同的计数模式，可以选择不同的信号源。

各计数模式下支持的信号源输入端口如下表所示。不同的本地编码器轴可以选择相同的信号源输入端口。

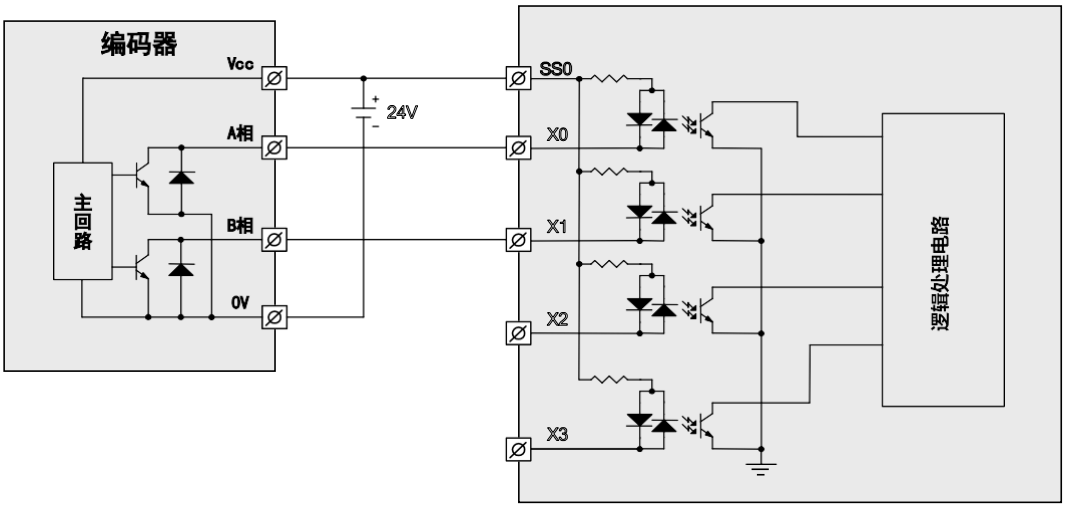
指令类别	IN0	IN1	IN2	IN3	IN4	IN5	IN6	IN7	内部 1ms	内部 1us
A/B 相	A 相	B 相	A 相	B 相	A 相	B 相	A 相	B 相	×	×
脉冲+方向	脉冲	方向	脉冲	方向	脉冲	方向	脉冲	方向	×	×
单相计数	脉冲	脉冲	脉冲	脉冲	脉冲	脉冲	脉冲	脉冲	脉冲	脉冲

当所选工作模式下需要两个输入信号时，**IN** 0 与 **IN** 1、**IN** 2 与 **IN** 3、**IN** 4 与 **IN** 5、**IN** 6 与 **IN** 7 分别为四组输入信号

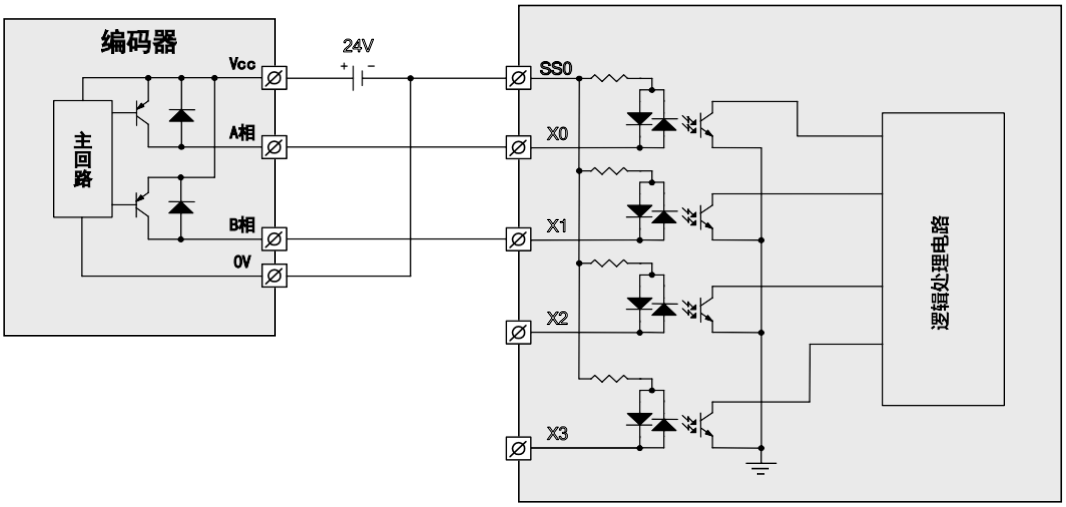
SC 系列 PLC 支持的 4 个计数器可以任意选择上面计数模式和信号源，不同计数器可以复用计数模式或信号源。

14.5.2.2 A/B 相模式

在 A/B 相模式下，编码器产生两个相位相差 90°的正交相位脉冲信号，即 A 相信号和 B 相信号。当 A 相信号超前 B 相信号时，计数器增计数；当 B 相信号超前 A 相信号时，计数器减计数。



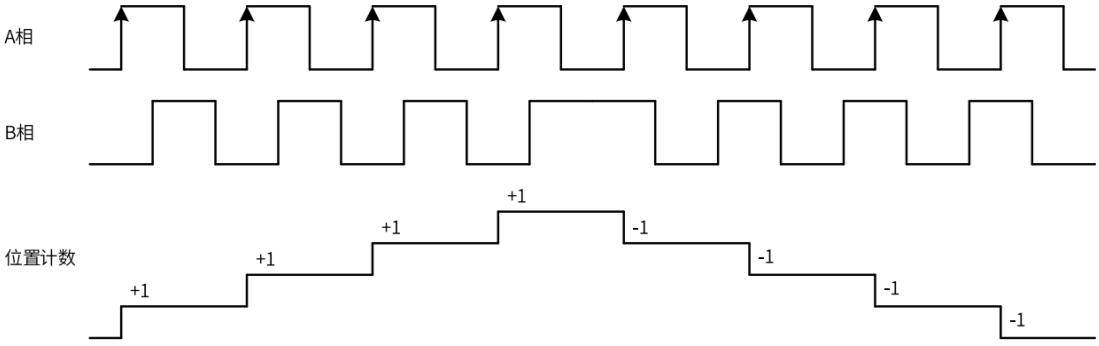
A/B 相编码器接线图--漏型输入接法



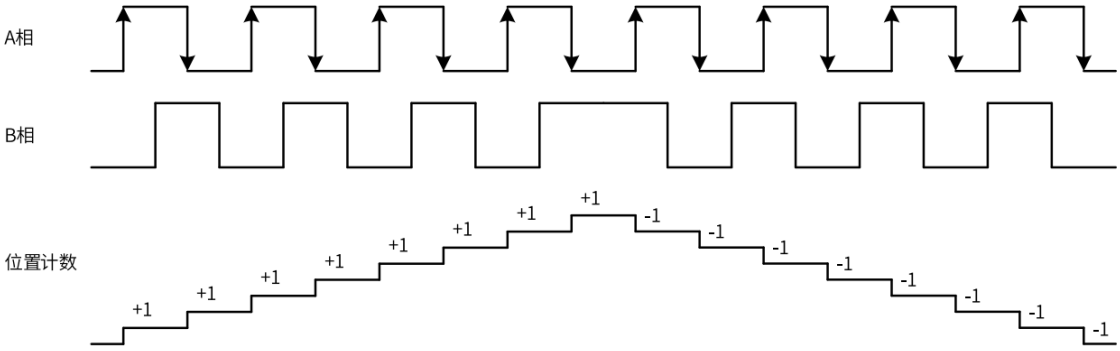
源型输入接法

A/B 相脉冲可以设置为 1 倍频、2 倍频或者 4 倍频模式工作。

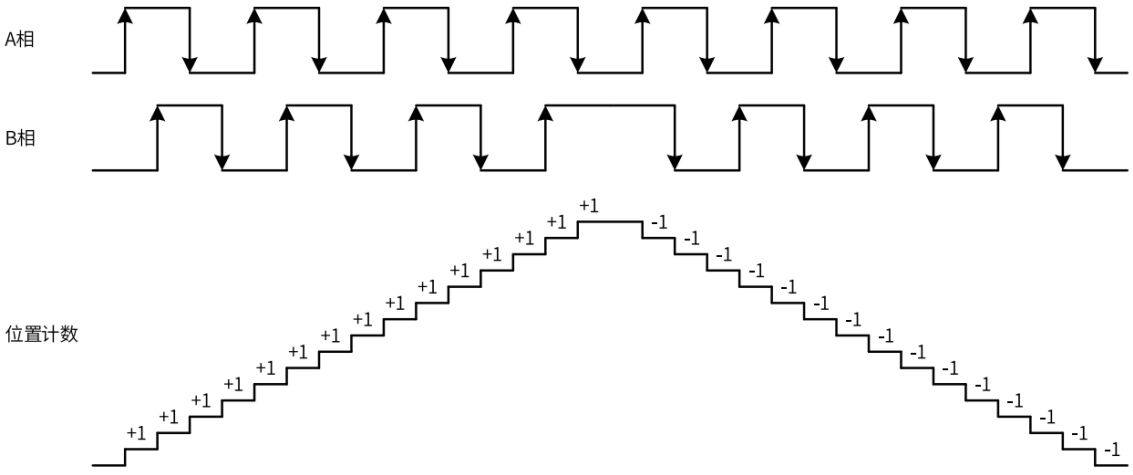
- A/B 相 1 倍频模式下，只对 A 相脉冲的上升沿计数，如下图所示：



- A/B 相 2 倍频模式下，对 A 相脉冲的上升/下降沿计数，如下图所示：

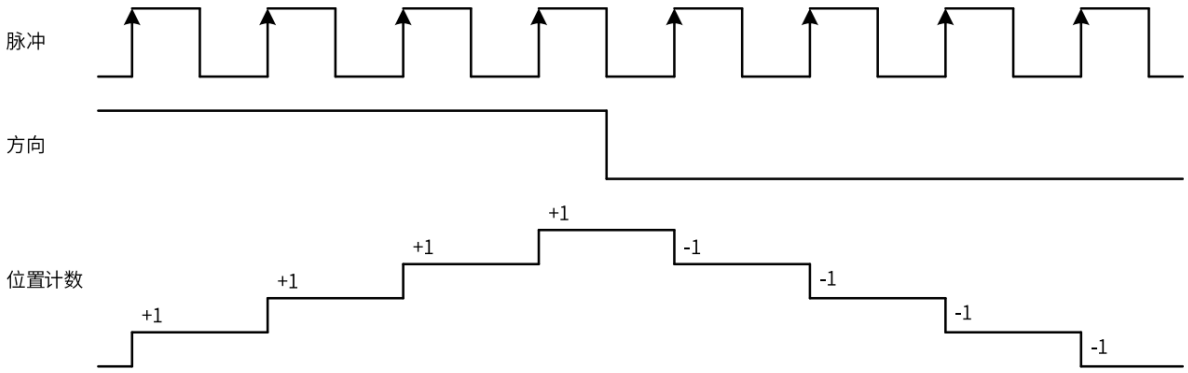


- A/B 相 4 倍频模式下，对 A 相脉冲和 B 相脉冲的上升/下降沿计数，如下图所示：



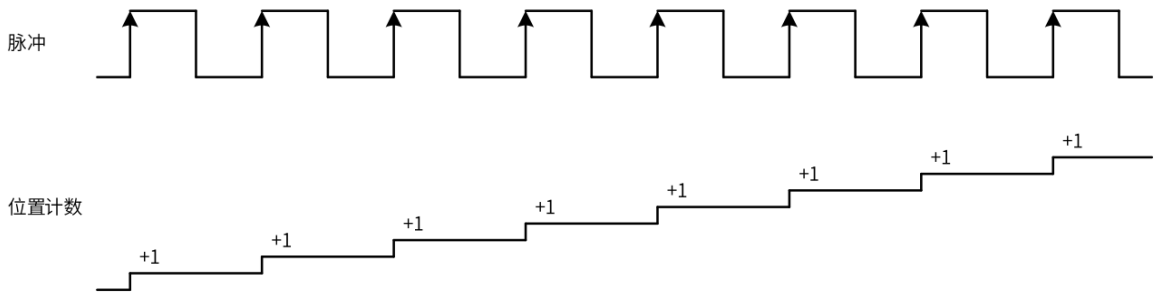
14.5.2.3 脉冲+方向模式

在该模式下，方向信号为 ON 时，高速计数器对脉冲信号增计数，方向信号为 OFF 时，高速计数器对脉冲信号减计数，如下图所示：



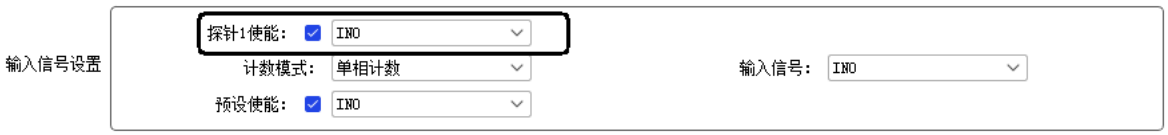
14.5.2.4 单相计数

在该模式下，高速计数器对脉冲信号增计数，在输入脉冲上升沿时，位置计数加 1。



14.5.3 探针端子设置

每个计数器支持 1 路外部输入锁存计数器当前值，实现探针功能。通过勾选探针使能启用外部输入的计数器轴位置锁存，输入端子可任意选择 IN0~IN7 输入。

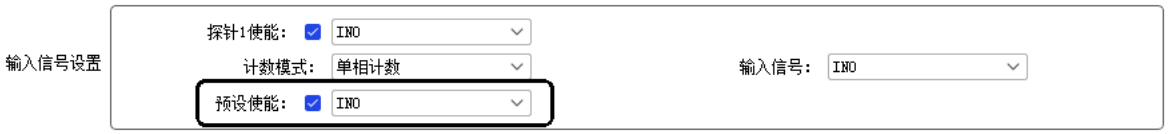


启用探针后，通过 LS_TouchProbe 功能块指令读取计数器轴的探针位置。

14.5.4 信号滤波和预置端子设置

通过信号滤波时间的设置，可以针对计数器轴的输入信号进行滤波，可以排除外部干扰对信号的影响。滤波时间单位为 100ns。

通过勾选预置使能启用外部输入预置计数器值，输入端子可任意选择 IN0~IN7。



启用预置功能后，通过 LS_Preset 功能块指令实现外部输入对编码器轴位置预置。

14.5.5 比较输出端子设置

勾选“比较输出使能”后，不通过软件处理即可实现比较相等时硬件输出，实时性高，输出响应可达微秒级别。

- 启动比较输出功能后，配合功能块指令，比较相等时通过硬件电路控制输出为 ON，输出端子可任意选择 OUT0~ OUT15
- 每个本地编码器轴配备一路比较输出功能，可根据需求配置输入端子与输出脉冲宽度。
- 配置完成后，通过 LS_Compare、LS_CompareFIFO、LS_CompareStep 功能块指令，实现轴位置比较输出。
- 指令可以选择输出模式，单位选择 ms 时，设置的时间范围为 0.1~65535ms。单位选择 Unit 时，应确保设置的值转换为脉冲单位后在 1~65535 的范围。

比较输出是直接通过硬件控制端口输出，不通过软件处理，因此不能通过程序中的软元件显示比较输出的状态。软元件和比较输出对输出端口控制为或关系，若软元件持续控制为 ON 状态，则实际端口输出保持为 ON。

14.6 计数器轴指令应用

14.6.1 概述

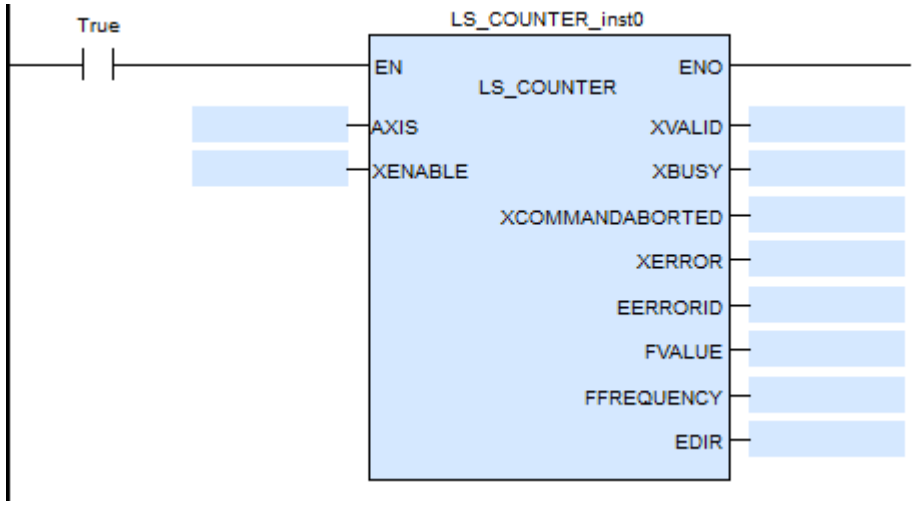
LeadStudio 软件中设置计数器轴后，配合功能块指令，可实现轴位置计数/速度测量、轴位置预置、轴位置锁存和比较功能。

14.6.2 轴位置计数/速度测量指令

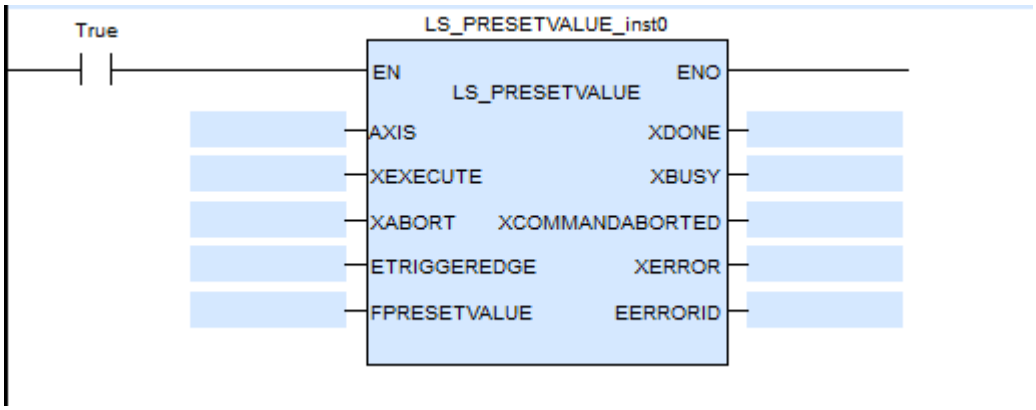
使用 LS_Counter 指令，可对计数器轴的位置计数和速度测量。

计数器轴位置值根据模式设置，在计数器轴模式的范围内变化，位置单位为 Unit。

计数器轴速度为当前实时速度，速度单位为 Unit/s。计数器轴可测量的最小速度为 1s 时间内计数器 1 个脉冲对应的速度，如计数器 1 个脉冲对应 0.01Unit，则可测量的最小速度为 0.01Unit/s。



14.6.3 轴位置预置指令



使用 LS_PresetValue 指令，根据预置条件，实现对计数器轴位置赋值。

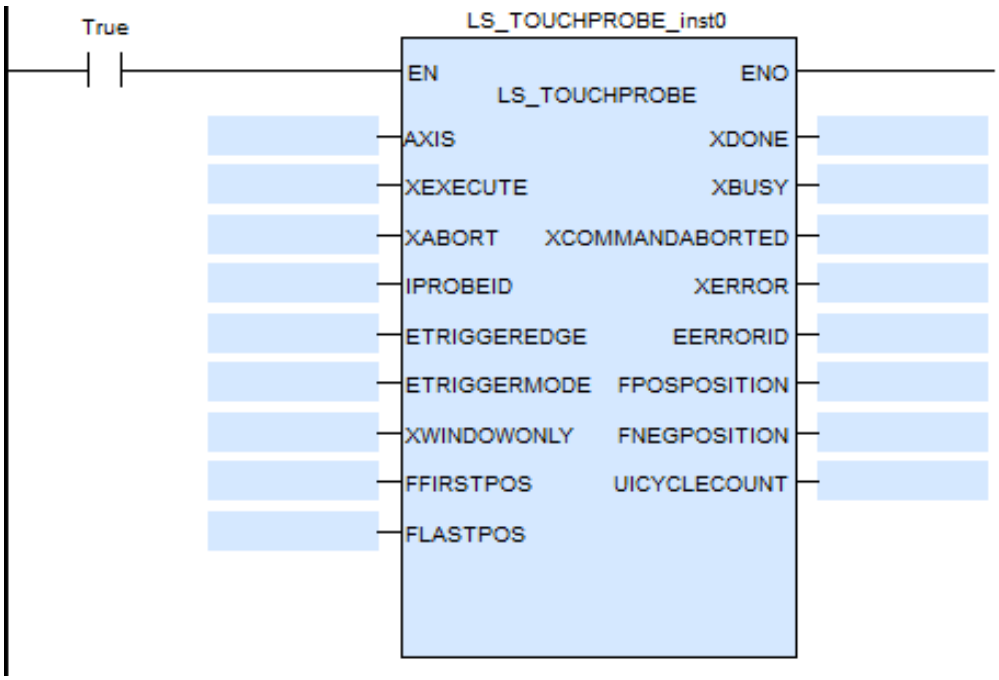
预置条件 EtriggerEdge 可选择指令上升沿触发或外部 X 输入触发。

EtriggerEdge	定义
0	指令上升沿触发
1	外部 DI 上升沿触发
2	外部 DI 下降沿触发
3	外部 DI 输入上升沿或下降沿触发

预置条件选择外部输入触发时，需要在计数器参数设置勾选预置功能，选择输入端子和触发条件，输入端子可任意设置选择 IN0~IN7，触发条件可选择上升沿或下降沿。

14.6.4 探针指令

使用 LS_TouchProbe 功能块指令，可在外部输入触发条件有效时，锁存计数器轴位置值。每个计数器轴支持 1 路探针，使用时，需要在计数器参数设置勾选对应的探针功能，选择输入端子和触发条件，输入端子可任意设置选择 IN0~IN7



参数 iProbeID 设置计数器使用的探针号。

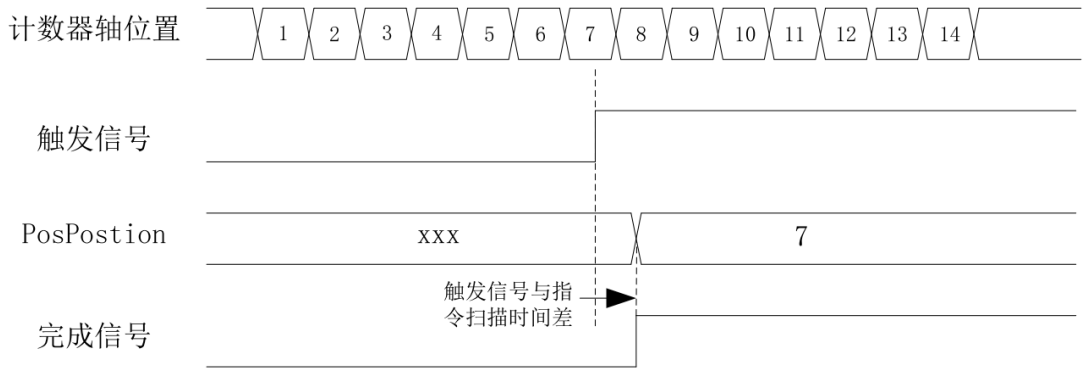
iProbeID	定义
0	表示使用探针 1

预置条件 EtriggerEdge 可选择指令上升沿触发或外部 X 输入触发。

EtriggerEdge	定义
0	上升沿触发
1	下降沿触发
2	外部输入上升沿或下降沿触发

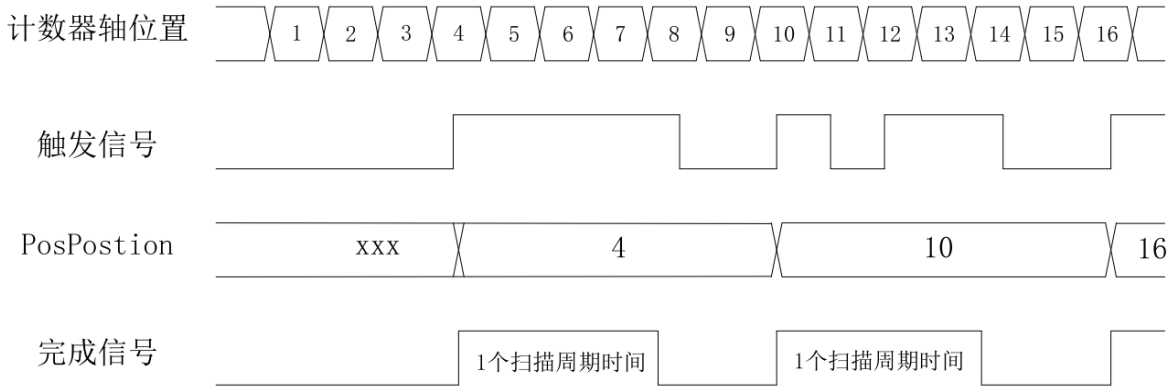
指令中 EtriggerMode 参数可设置单次触发和连续触发模式。

使用单次触发模式，功能块指令能流有效，外部输入触发条件有效时，锁存 1 次计数器轴位置，输出完成信号。探针位置根据触发边沿实时锁存计数器轴位置，不受程序执行影响。程序指令执行时，受扫描周期影响，程序扫描执行到锁存指令时，将锁存位置更新到指令输出参数中。



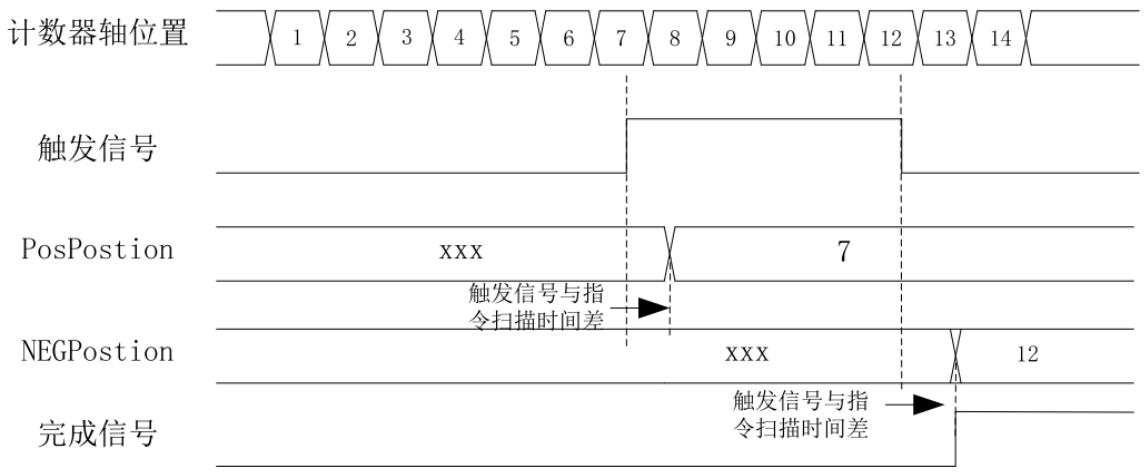
上升沿单次触发模式

- 使用连续触发模式，功能块指令能流有效，外部输入触发条件有效时，锁存计数器轴位置，输出完成信号，完成信号有效时间 1 个扫描周期。完成信号变为 OFF 后，外部输入触发条件有效，会继续锁存计数器轴位置，并输出有效时间为 1 个扫描周期的完成信号。在完成信号有效的 1 个扫描周期时间内，若外部输入触发条件有效，此时不会锁存计数器轴的位置。

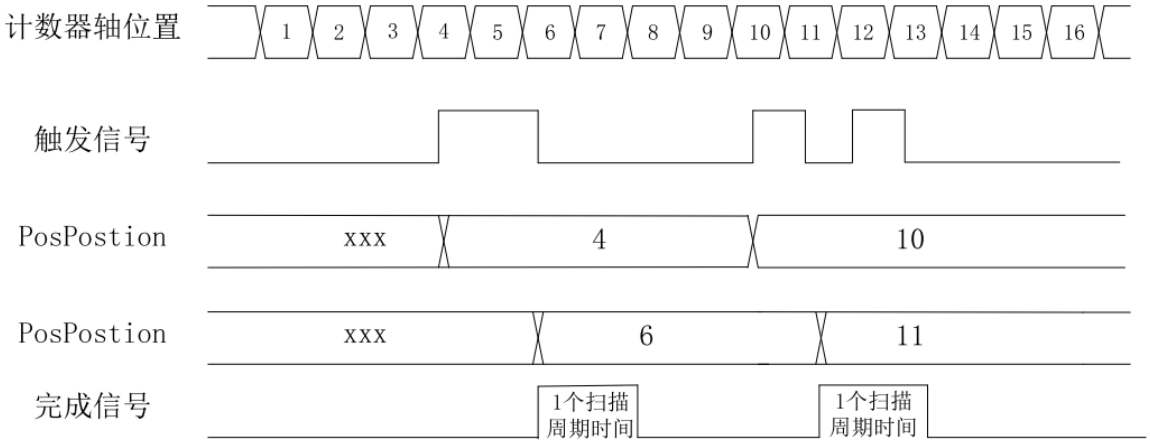


上升沿连续触发模式

- 使用双边沿触发模式时，当上升沿和下降沿都触发完成锁存后，输出完成后信号。单次触发模式时，完成信号持续到指令完成；连续触发模式时，完成信号有效时间1个扫描周期，完成信号有效的1个扫描周期内，不响应触发锁存信号。



上升下降沿单次触发模式



上升下降沿连续触发模式

14.7 高速硬件比较输出

计数器轴可实现位置比较硬件输出，计数器轴与比较位置相等时直接通过硬件电路控制输出为 ON，输出延时小于 1us。

- 设置计数器轴的比较输出功能

  计数器轴的参数设置界面中勾选比较输出使能

  输出端子可任意选择 OUT0~OUT15

脉冲输出宽度选择为时间单位时，输出脉冲宽度时间精度为 100us，最大可输出 6500ms 时间；脉冲输出宽度选择为用户单位（Unit），最大可输出 65535 脉冲对应的单

位宽度。

- 在比较指令中使能 xOutEnable 参数

使用 LS_Compare、LS_CompareStep、LS _ArrayFIFO 比较指令，将 xOutEnable 设为 TRUE，即在指令比较相等时关联硬件输出。

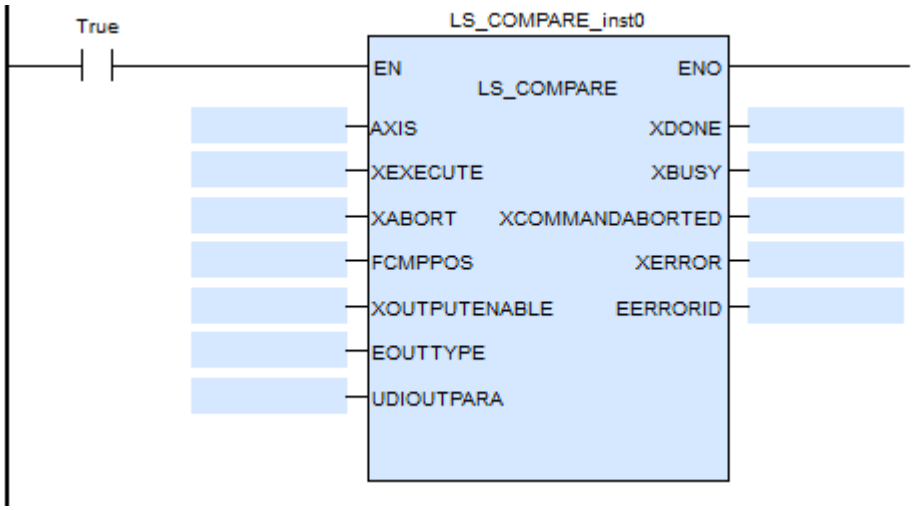
当指令执行比较相等时，直接通过硬件电路控制设定的输出端子为 ON，持续输出宽度后输出变为 OFF。

14.7.1 比较指令

使用 LS_Compare、LS_CompareStep、LS_CompareFIFO 可实现计数器轴的单个位置比较、等间距连续比较和多位置连续比较。

3.2.1.1 LS_Compare

实现计数器轴与单个位置比较，指令能流有效时，计数器轴位置达到比较位置后，输出完成信号。

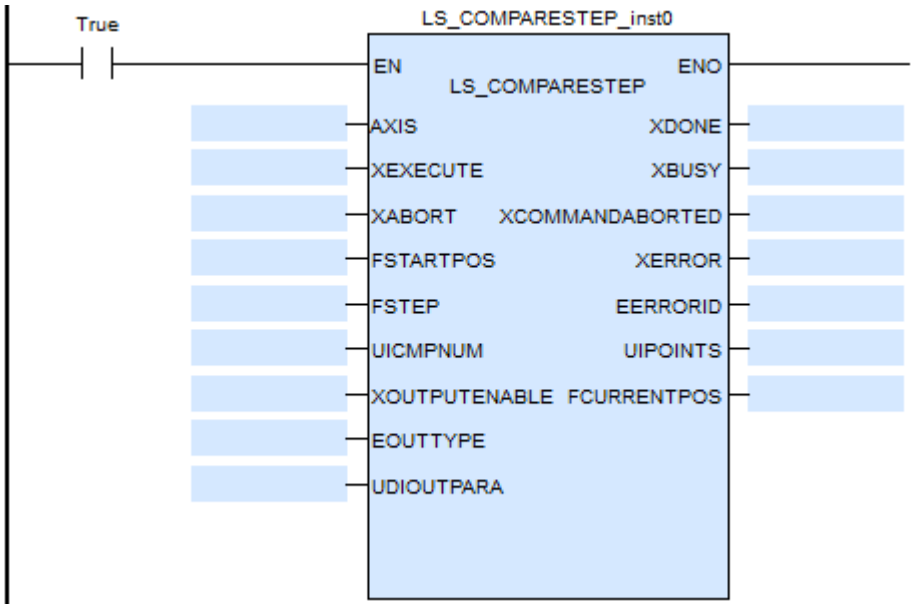


3.2.1.2 LS_CompareStep

实现计数器轴与等间距连续位置比较，指令能流有效时，计数器轴位置与 fStartPos

位置开始比较，比较相等后，比较位置增加或减小 Step 间距后继续比较，每次比较相等后，不会输出一个周期的完成信号，等间距比较完最后一个比较位置后，才会输出完成信号。

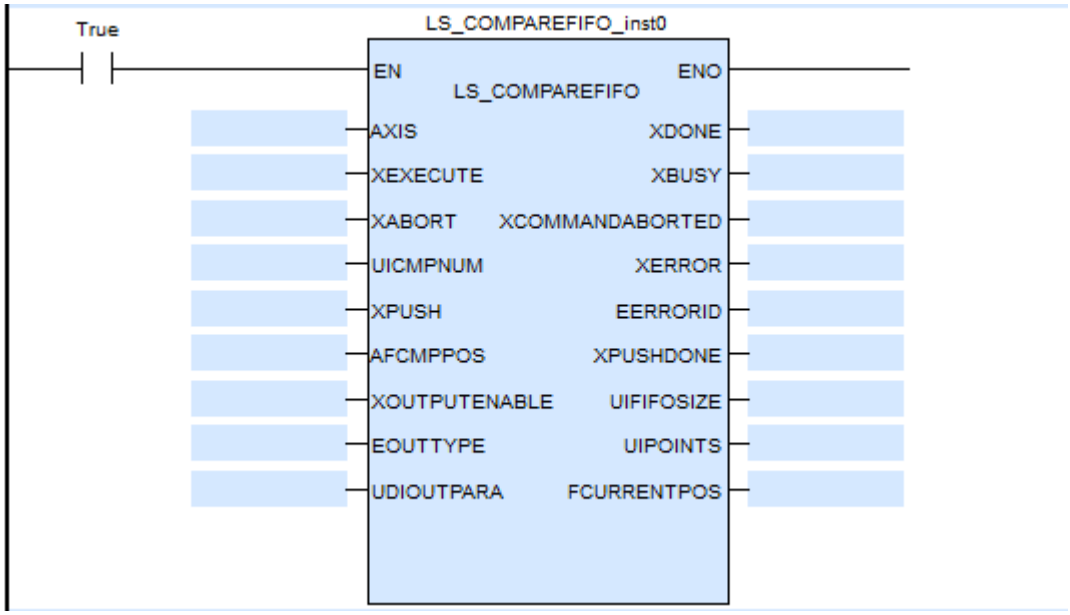
- fStep 大于 0 时，每次比较相等时，比较位置增加 fStep 间距，当 uiPoints 等于 uiCmpNum，当前比较位置即为最后一个比较位置。
- fStep 小于 0 时，每次比较相等时，比较位置减小 Step 间距，当 uiPoints 等于 uiCmpNum，当前比较位置即为最后一个比较位置。
- 输出参数 uiPoints 指示已完成比较相等的个数。



3.2.1.3 LS_CompareFIFO

实现计数器轴与数组多位置连续比较，指令能流有效时，计数器轴位置与数组第 1 个位置开始比较，比较相等后，与数组下一个位置值比较。直到比较完最后一个比较位置后，输出完成信号。

- 高速一维比较一致输出，FIFO 模式，最大 FIFO 数为 1000 个，可动态压入比较点



- 高速一维比较队列模式输出，支持 3 种类型：FIFO 输出时间、FIFO 输出电平和 FIFO 根据计数脉冲个数输出。FIFO-电平方式相对于 FIFO-时间方式的区别，在于一个是设置输出电平的高低，一个是让输出电平保持一定的时间后自动关闭。
- 输出时间是指当比较一致后输出持续的时间，单位为 us，需要的输入参数为设置比较模式，将比较点压入，设置输出时间即可；
- 输出电平则满足比较一致后，持续输出，需要的输入参数为设置比较模式，将比较点和比较逻辑压入即可。
- 输出脉冲个数是指当比较一致后，持续计数器的脉冲计数个数后，关闭输出。比较点为指针类型，需指定比较点数 fCmpPos，注意比较点和比较逻辑电平数组长度必须大于比较点数。xPush 为动态压点，即在比较过程中支持压入比较点，且 FIFO 最大长度为 1000，若超过 1000 则会等待待比较点+FIFO 点≤1000 才会压入 FIFO 中。
- 比较功能块使用前，必须在对应计数器界面上配置输出端口
- xPush 第一次执行指令时不需要压入，xDone 没有给出完成时，可以用 xPush 一直压数据，xPush 上升沿触发，xPushDone 未完成时，重复压入会导致指令错误。
- 连续比较同一个位置值，需要重新到达一次才触发比较输出。

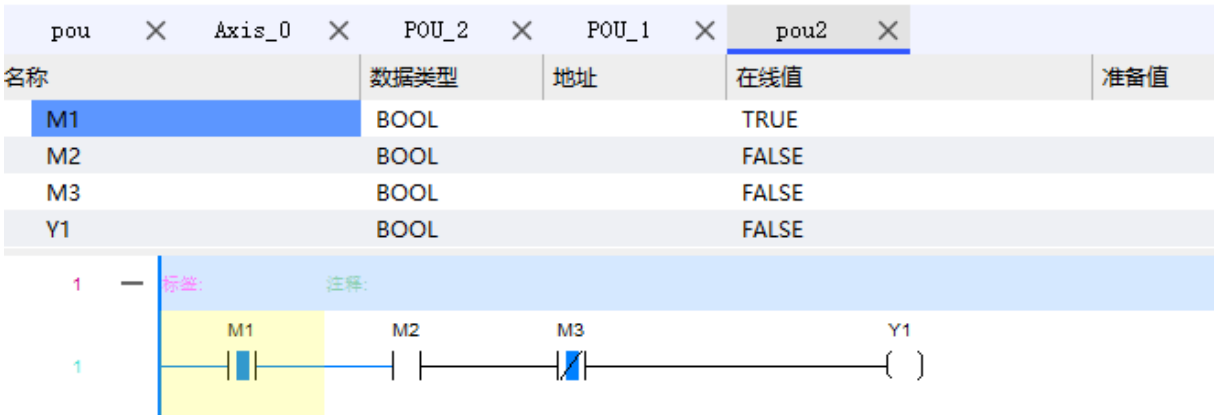
15 运行调试

15.1 在线操作

15.1.1 登录 PLC

1) 鼠标点击工具栏中的快捷键“ ”编译 PLC 程序，再点击登录按钮“ ”，即可把程序下载到 PLC，(注意：点击下载时，会弹出一个下载类型选择，用户可以根据自己的需要选择在线下载或完整下载，以及是否清除保持型变量的现有值或者初始化)

2) 点击“ ”-----“ ”，即可登录并运行 PLC，如下图所示：

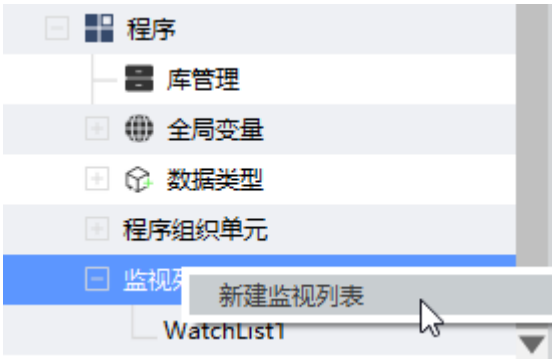


若点击“ ”前，PLC 没有下载程序或程序发生变动，则登录时会弹出上述步骤

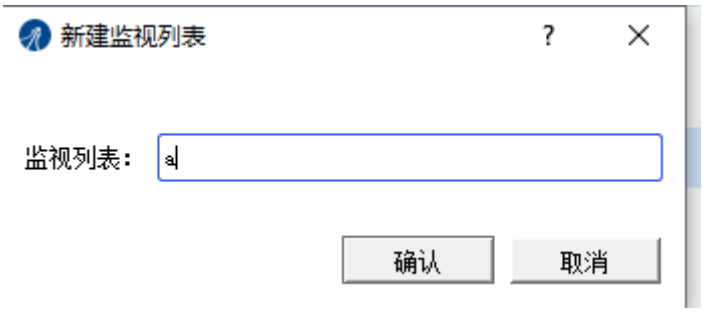
的下载类型选择框，用户需要下载后才可登录 PLC

15.1.2 添加变量监控

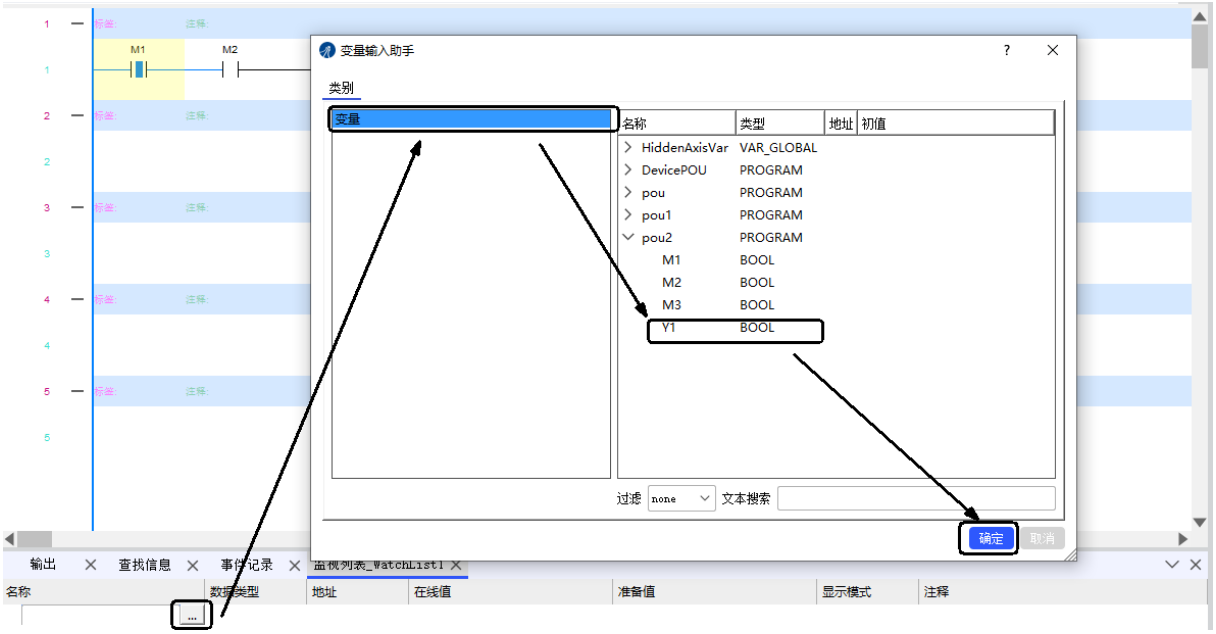
1) 右击监视列表-----新建监视列表，如下图所示：



2) 然后给新建的监视列表设置名字；



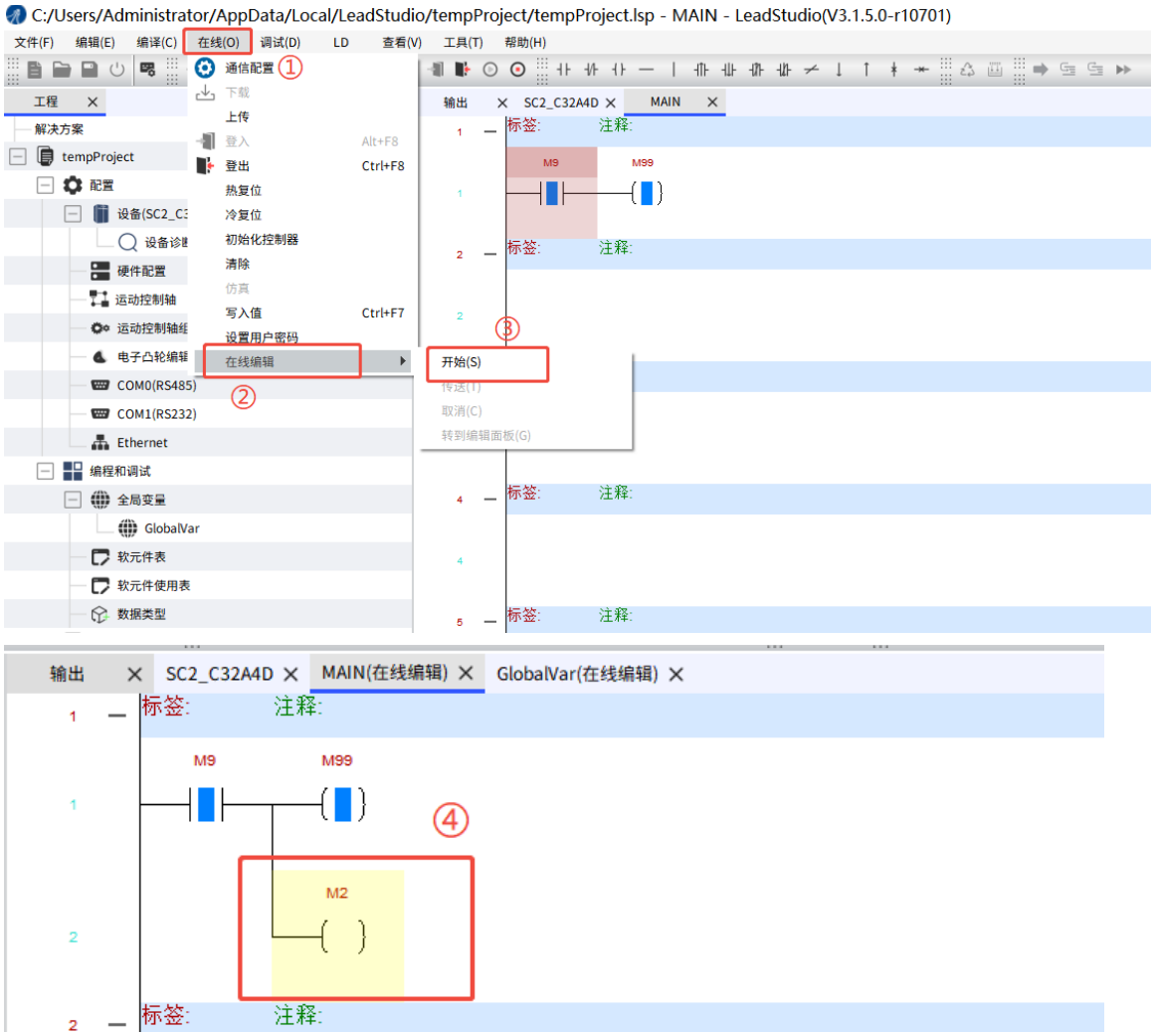
3) 点击监视列表的箭头-----在输入助手中展开 PLC_PRG-----选择需要监视的变量-----点击确定。

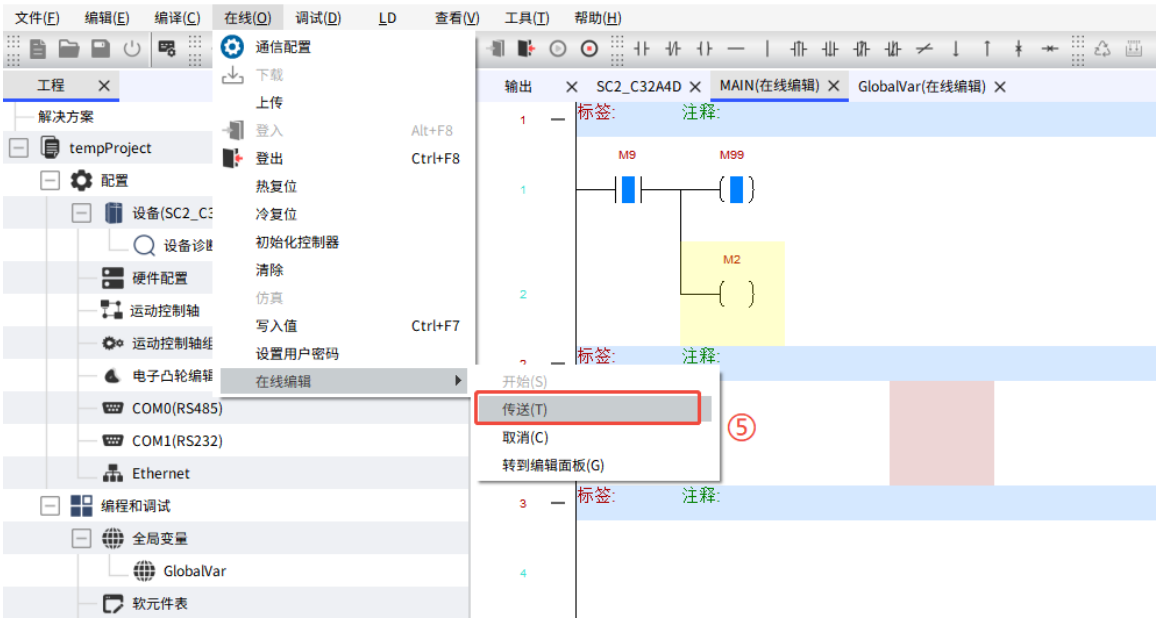


15.1.3 在线编辑

LeadStudioV2.7/V3.1 及以上版本支持在线编辑功能，用户可在不停机状态下在线修改并下载程序

操作步骤如下：①点击工具栏的“在线”；②在下拉框中选择“在线编辑”选项；③点击“开始”进入在线编辑状态；④在线状态下修改用户程序；⑤点击“传送”即可完成在线下载

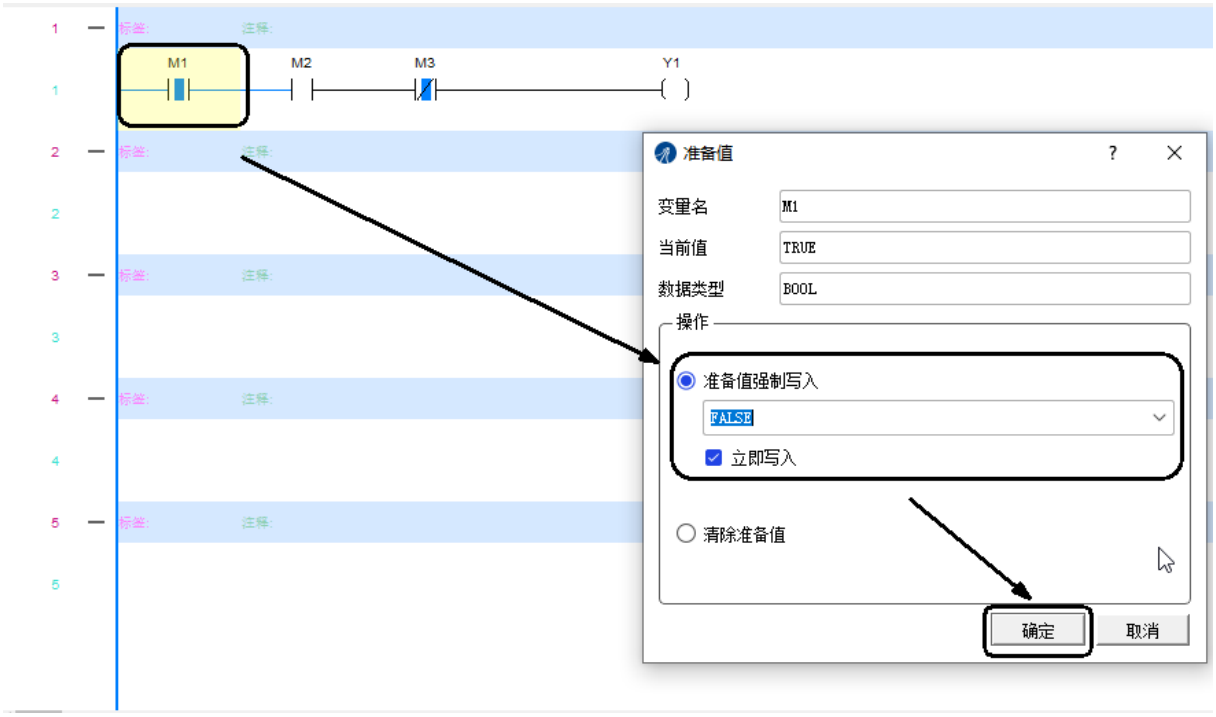




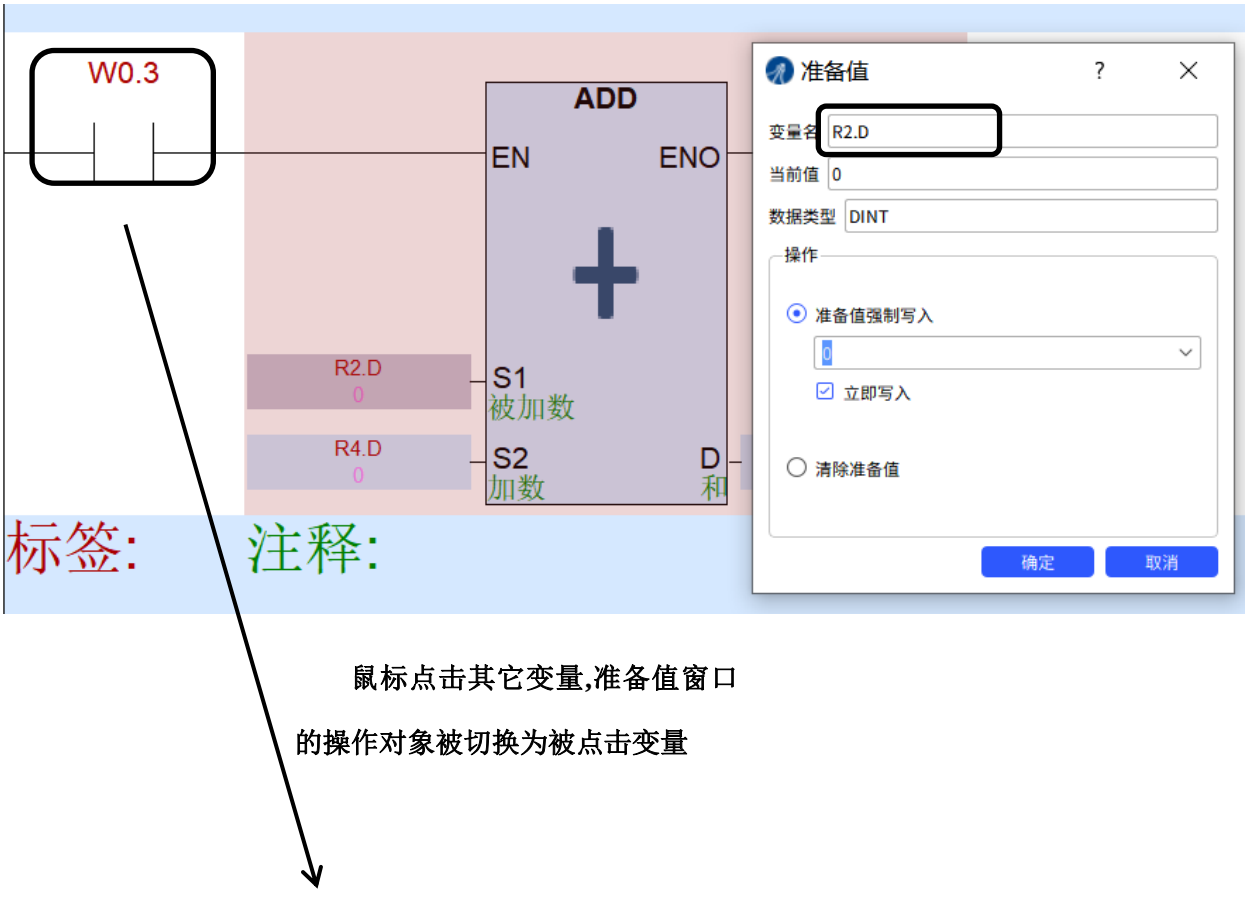
说明：只有在程序组织单元页面才可支持在线编辑功能，若在其它页面打开在线编辑则会弹窗提示“编辑器无法使用在线编辑”，如下所示：

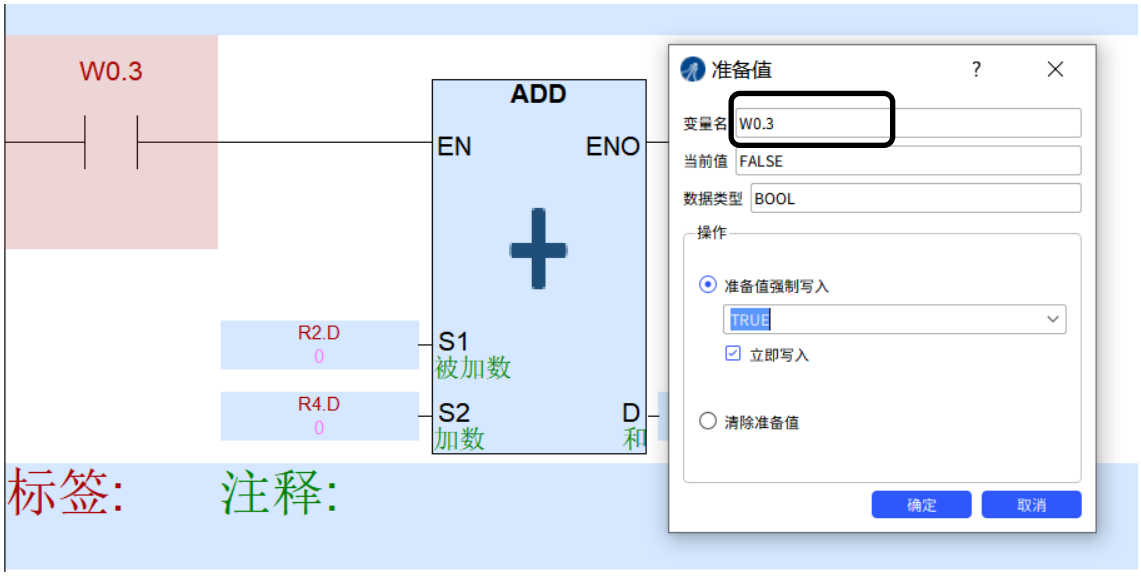
15.2 在线写入值

双击变量-----在准备值的界面中输入值-----点击确定，勾选立即写入时，变量会在点击“确定”后立即写入到变量中，如不勾选，则作为准备值，等待执行“写入准备值”后写入到变量中。



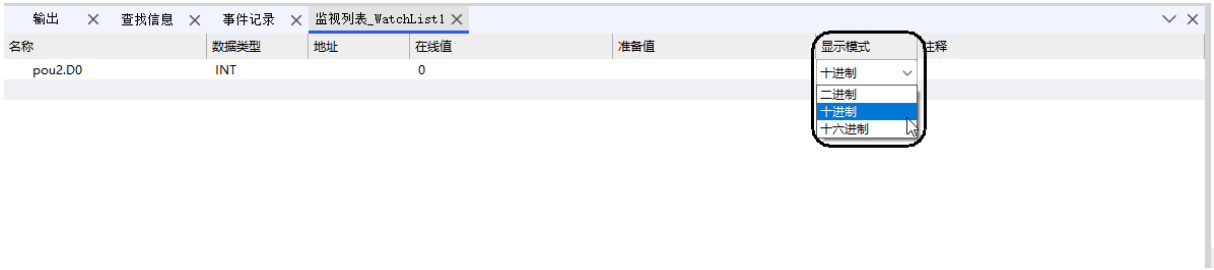
在准备值弹窗打开时，通过鼠标点击其它变量可快速切换准备值弹窗的操作对象，如下所示：





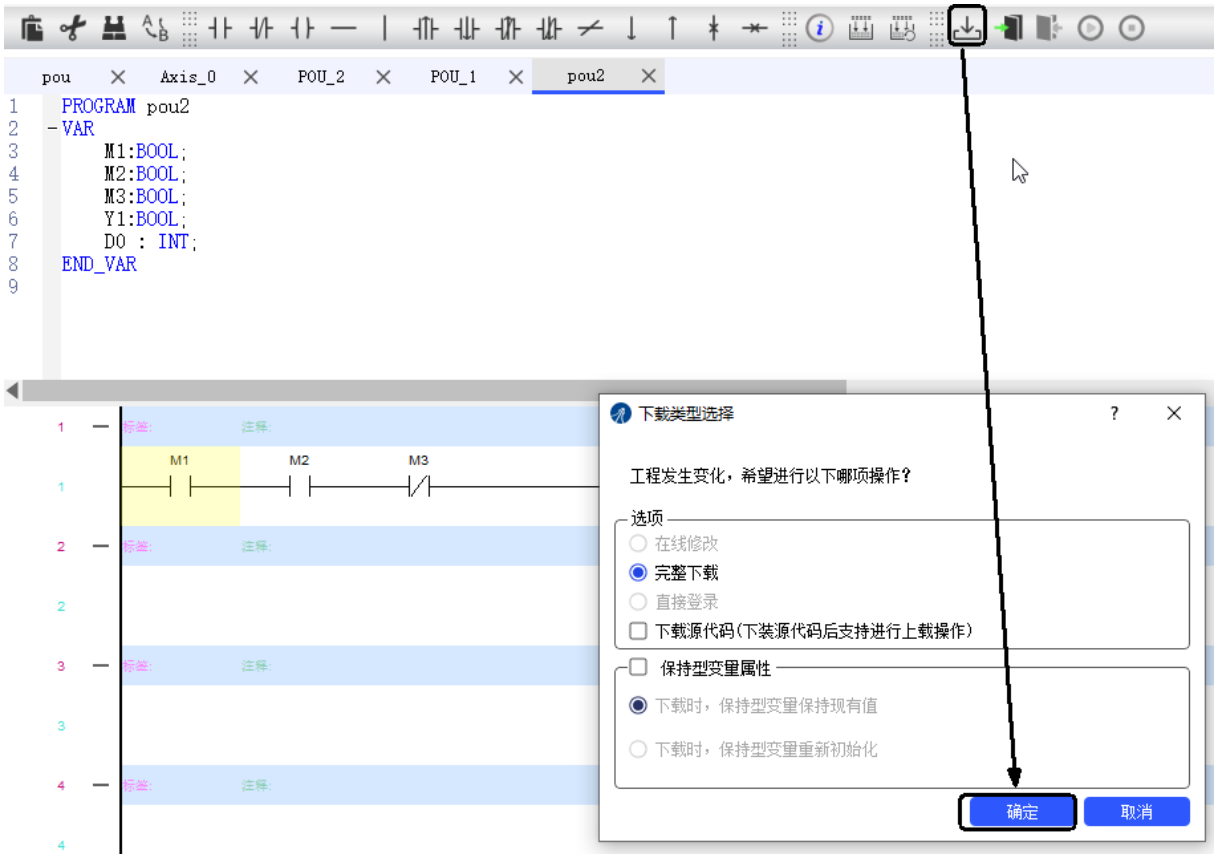
15.3 切换显示进制

监视列表的“显示模式”一列可以直接切换进制。

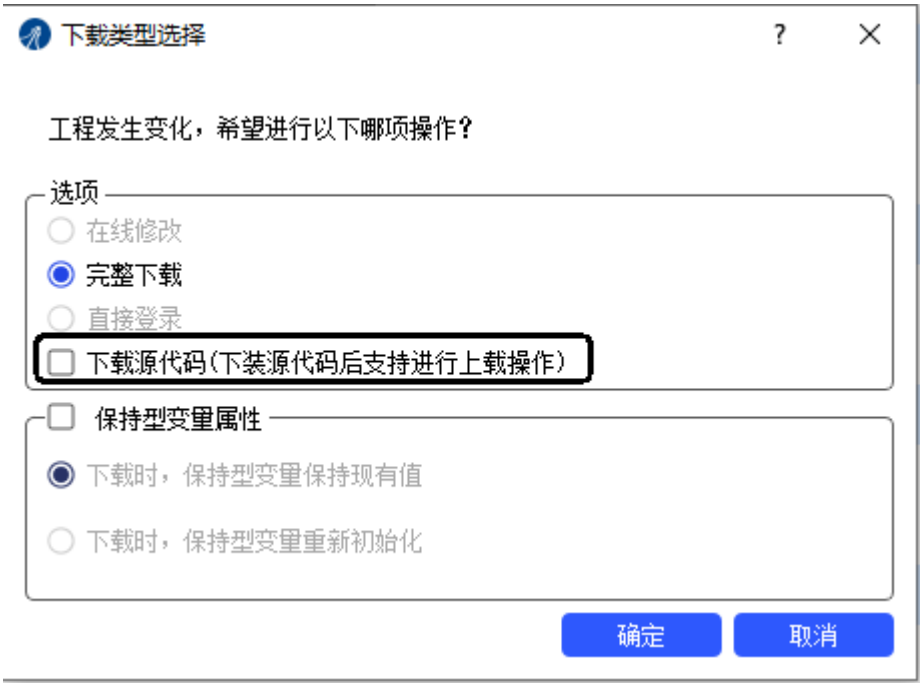


15.4 下载操作

1) 完整下载，点击下载-----确定，如下图所示：



2) 下载源代码，只需要在下载类型选择界面中勾选下载源代码，如下图所示：



3）在线修改，在下载类型选择界面中勾选在线修改即可在不停机状态下更新 PLC 程序(软件版本需在 V2.7/V3.1 以上)

说明：下表情况不支持在线修改

编辑对象	不支持在线下载
变量	1、原有的变量从用户定义的类型到另外一种用户定义类型的变化（包括结构体、FB、枚举等） 2、原有的变量从基础数据类型到用户定义类型的变化（包括结构体、FB、枚举等） 3、原有的变量用户定义类型（包括结构体、FB、枚举等）到基础数据类型的变化 4、变量从单一变量转对应的数组或从数组转单一变量（如 int 不能转数组/指针 int，但数组一维 int 可以转数组二维 int）* 5、单一变量、用户定义类型转 pointer to 变量类型*
库	库中的 pou、变量表、dut 等发生改动（若库中内容没有被调用，则改动不会影响在线修改功能）
轴	1、增加、删除轴 2、修改轴配置参数

	3、修改轴原点、限位等
轴组	1、增加、删除轴组 2、修改轴组配置参数 3、修改轴组组成
通讯配置	1、通讯协议变化 2、Modbus 通讯配置变化 3、通讯参数变化 4、Modbus 从站数量发生变化
IO 映射	1、映射地址发生变化 2、绑定变量发生变化
本地 IO	1、滤波参数发生变化 2、绑定刷新任务发生变化
本地模块	1、本地总线配置发生变化（数量及类型等） 2、模块的配置参数发生变化 3、使能失能模块
任务配置	1、增加、删除任务 2、任务优先级变化 3、任务配置（循环时间、触发变量、看门狗等）变化 4、调用 POU 的变化
工程树节点	轴、轴组、串口/以太网设备等增删不允许增量下载

15.5 复位功能（复位、冷复位）

点击在线-----复位和冷复位

15.6 数据快照

数据快照功能可用于快速将变量的当前值同步到其他PLC或程序的初始值中，典型的应用场景如下：

- 当程序异常时，可以获取当前变量数据分析问题。
- 变量数据保存后其他PLC使用，快速实现机台数据迁移
- 将当前时刻变量数据参数同步到初始值。

快照值跟随工程保存。

15.6.1 快照功能入口

在“登录”状态下，用户可以在全局变量表或局部变量表找到快照列，默认情况下快照列没有数据。

GVL_HMI

名称	数据类型	地址	在线值	准备值	注释	快照
gb_Start_HMI	BOOL	%MX50.0	FALSE		启动	☑
gb_Stop_HMI	BOOL	%MX50.1	FALSE		停止	☑
gb_Reset_HMI	BOOL	%MX50.2	FALSE		复位	☑
gb_ClearErr_HMI	BOOL	%MX50.3	FALSE		清除报错	☑
gb_Auto_HMI	BOOL	%MX50.4	FALSE		手自动切换，0...	☑
gb_HMI老化测试	BOOL	%MX50.5	FALSE			☑
gb_HMI全部清板	BOOL	%MX50.6	FALSE			☑
gb_HMI_NG出板	BOOL	%MX50.7	FALSE			☑
gb_HMI_NG转换	BOOL	%MX51.0	FALSE			☑
gb_HMI一键调宽	BOOL	%MX51.1	FALSE			☑
gb_HMI_OK清板	BOOL	%MX51.2	FALSE			☑
gb_HMI_NG清板	BOOL	%MX51.3	FALSE			☑
g_HMI接驳前传送	BOOL	%MX52.0	FALSE		//////	☑
g_HMI接驳后传送	BOOL	%MX52.1	FALSE			☑
g_HMI平台正转	BOOL	%MX52.2	FALSE			☑
g_HMI平台反转	BOOL	%MX52.3	FALSE			☑
g_HMI内轨道前移	BOOL	%MX52.4	FALSE			☑
g_HMI内轨道后退	BOOL	%MX52.5	FALSE			☑
g_HMI接驳后轨道前移	BOOL	%MX52.6	FALSE			☑
g_HMI接驳后轨道后退	BOOL	%MX52.7	FALSE			☑
g_HMI升降上升	BOOL	%MX54.0	FALSE			☑
g_HMI升降下降	BOOL	%MX54.1	FALSE			☑
g_HMI接驳前轨道前移	BOOL	%MX54.2	FALSE			☑
g_HMI接驳前轨道后退	BOOL	%MX54.3	FALSE			☑
g_HMI升降回零	BOOL	%MX54.4	FALSE			☑
g_HMI内轨道回零	BOOL	%MX54.5	FALSE			☑

15.6.2 快照值操作

右键变量表中的任意位置，在弹出的右键菜单中可以对快照值进行操作，存在以下三种操作：

1）当前值->快照值：将变量的当前值同步到快照值中

2）快照值->当前值：将变量的快照值同步到当前值中

3）快照值->初始值：将变量的快照值同步到初始值中

GVL_HMI

名称	数据类型	地址	在线值	准备值	注释	快照
gb_Start_HMI	BOOL	%MX50.0	FALSE		启动	☑
gb_Stop_HMI	BOOL	%MX50.1	FALSE		停止	☑
gb_Reset_HMI	BOOL	%MX50.2	FALSE		复位	☑
gb_ClearErr_HMI	BOOL	%MX50.3	FALSE		清除报错	☑
gb_Auto_HMI	BOOL	%MX50.4	FALSE		手自动切换，0...	☑
gb_HMI老化测试	BOOL	%MX50.5	FALSE			☑
gb_HMI全部清板	BOOL	%MX50.6	FALSE			☑
gb_HMI_NG出板	BOOL	%MX50.7	FALSE			☑
gb_HMI_NG转换	BOOL	%MX51.0	FALSE			☑
gb_HMI一键调宽	BOOL	%MX51.1	FALSE			☑
gb_HMI_OK清板	BOOL	%MX51.2	FALSE			☑
gb_HMI_NG清板	BOOL	%MX51.3	FALSE			☑
g_HMI接驳前传送	BOOL	%MX52.0	FALSE		//////	☑
g_HMI接驳后传送	BOOL	%MX52.1	FALSE			☑
g_HMI平台正转	BOOL	%MX52.2	FALSE			☑
g_HMI平台反转	BOOL	%MX52.3	FALSE			☑
g_HMI内轨道前移	BOOL	%MX52.4	FALSE			☑
g_HMI内轨道后退	BOOL	%MX52.5	FALSE			☑
g_HMI接驳后轨道前移	BOOL	%MX52.6	FALSE			☑
g_HMI接驳后轨道后退	BOOL	%MX52.7	FALSE			☑
g_HMI升降上升	BOOL	%MX54.0	FALSE			☑
g_HMI升降下降	BOOL	%MX54.1	FALSE			☑
g_HMI接驳前轨道前移	BOOL	%MX54.2	FALSE			☑
g_HMI接驳前轨道后退	BOOL	%MX54.3	FALSE			☑
g_HMI升降回零	BOOL	%MX54.4	FALSE			☑
g_HMI内轨道回零	BOOL	%MX54.5	FALSE			☑

注：只有勾选的变量才会执行对应的操作，未勾选变量的值不会受任何影响，勾选“快照”前的复选框可以全选变量表中的所有变量。

GVL_HMI

名称	数据类型	地址	在线值	准备值	注释	快照
gb_Start_HMI	BOOL	%MX50.0	FALSE		启动	☑ FALSE ①
gb_Stop_HMI	BOOL	%MX50.1	FALSE		停止	☑ FALSE
gb_Reset_HMI	BOOL	%MX50.2	FALSE		复位	☐ ②
gb_ClearErr_HMI	BOOL	%MX50.3	FALSE		清除报错	☐
gb_Auto_HMI	BOOL	%MX50.4	FALSE		手自动切换，0...	☐

① 受快照操作影响

② 不受快照操作影响

15.6.3 快照值导入导出

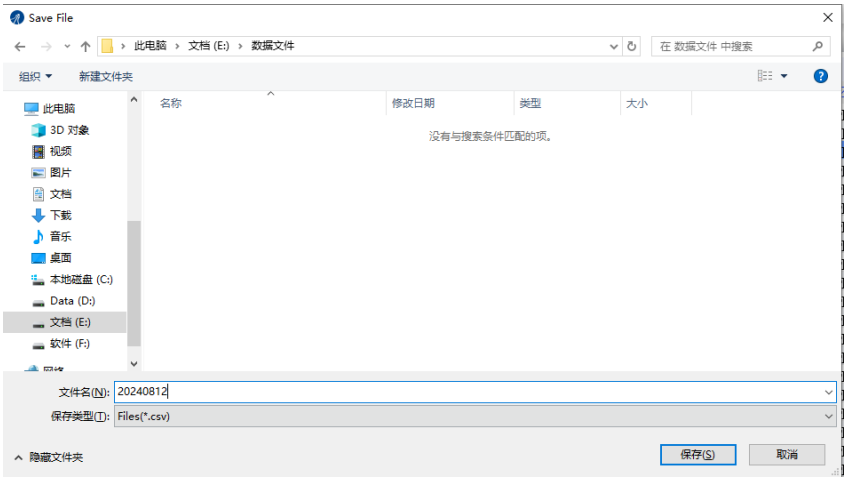
快照值支持导入导出，便于用户进行灵活的数据操作，导出快照的方式如下：

1）点击变量表右侧的  图标

GVL_HMI

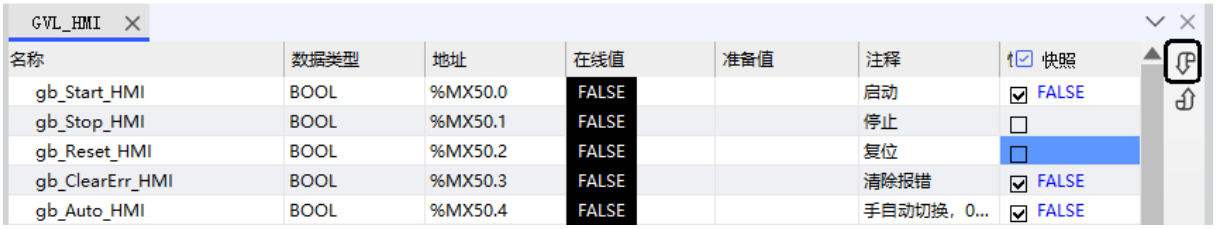
名称	数据类型	地址	在线值	准备值	注释	快照
gb_Start_HMI	BOOL	%MX50.0	FALSE		启动	☑ FALSE
gb_Stop_HMI	BOOL	%MX50.1	FALSE		停止	☐
gb_Reset_HMI	BOOL	%MX50.2	FALSE		复位	☐
gb_ClearErr_HMI	BOOL	%MX50.3	FALSE		清除报错	☑ FALSE
gb_Auto_HMI	BOOL	%MX50.4	FALSE		手自动切换，0...	☑ FALSE

2）在弹出的窗口中选择保存路径及输入文件名后，点击“确定”按钮即可完成导出。

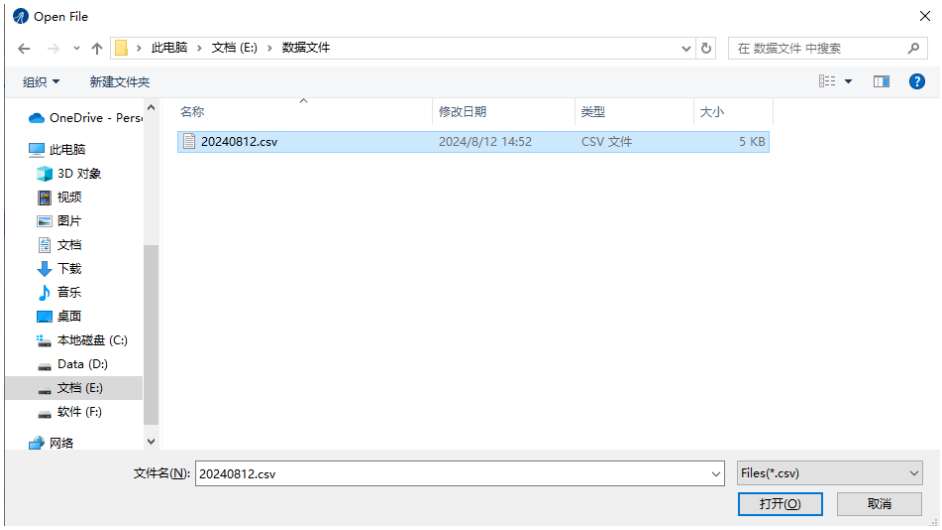


导入变量快照的方式如下：

1）点击变量表右侧的 图标



2）在弹出的窗口中选择需要导入的数据文件后后，点击“打开”按钮即可完成导出。



注：导出快照文件只针对被勾选的变量，导入快照文件时如果变量类型或变量名不匹配，则对应变量的快照值无法导入。

15.7 工程比较

工程比较用于比较两个工程各类对象之间的差异，其中数据类型、全局变量表、程序组织单元支持进行内容详细比较，其他工程树对象仅支持对象差异比较，内容详细比较差异部分可按行同步或块同步方式手动同步至当前打开的工程中。

工程比较分为离线比较和在线比较两种模式。

离线比较：当前打开工程与存放在本地的工程进行比较。

在线比较：当前打开工程与在线PLC内部的程序进行比较。

注意事项：

①只能在相同机型的工程文件之间进行比较，例如SC2_C32A4D机型工程文件只能与同样是SC2_C32A4D机型的工程文件之间比较，不能与SC2_C32A2D机型工程文件进行比较。

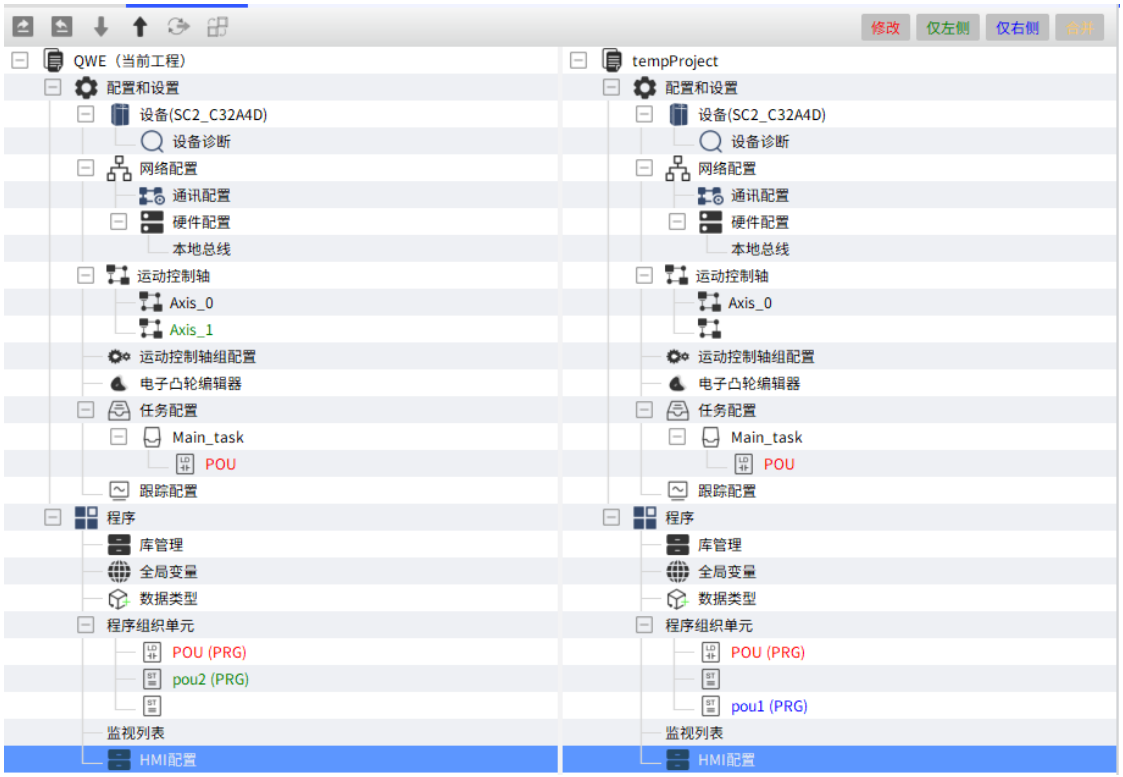
②登录状态下无法进行工程比较，且在工程比较期间限制用户执行登录PLC和编辑工程树对象。

③工程比较过程中，不能使用LeadStudio软件打开正在进行工程比较的工程文件。

④PLC必须下载源代码，方可与在线PLC内部的程序进行比较。

软件版本要求：V2.7/V3.1及以上

15.7.1 工程比较页面



左侧表示当前工程，右侧表示比较工程，工程比较界面工具栏图标功能或含义如下表所示。

	详细比较，单击该图标进入详细比较界面。
	工程树比较，单击该图标进入工程树比较界面。
	下一点差异，单击该图标跳转至下一点差异部分内容。
	上一点差异，单击该图标跳转至上一点差异部分内容。
	行同步，在详细比较界面中选中差异处，单击该图标可将右侧比较工程的该行同步至左侧当前工 程中；再次单击，将撤销行同步操作，恢复至行同步前的状态。 说明：该图标按钮仅在详细比较界面中有效。

	块同步，在详细比较界面中选中差异处，单击该图标可将右侧比较工程中当前行前后的连续差异 部分（即块同步内容覆盖至当前行前/后出现无差异部分为界）一起同步至左侧当前工程中。再次单击，将撤销块同步操作，恢复至块同步前的状态。 说明：该图标按钮仅在详细比较界面中有效。
	工程比较差异部分内容不同颜色字体含义标识，具体含义请参见下表。

工程比较差异部分使用不同颜色的字体区分，不同颜色的字体含义如下表所示。

字体颜色	含义
黑色字体	表示两侧内容一致。
绿色字体	表示该部分内容仅左侧当前工程中存在，右侧比较工程中以空单元形式进行填充对齐。
蓝色字体	表示该部分内容仅右侧比较工程中存在，左侧当前工程中以空单元形式进行填充对齐。
红色字体	表示该部分内容在左右两侧工程中都存在，但内容存在差异。
黄色字体	表示该部分内容为同步内容。 说明：黄色色块仅在详细比较界面中显示。

15.7.2 操作步骤

①软件左上角点击“文件”，在下拉框中选择“工程比较”；

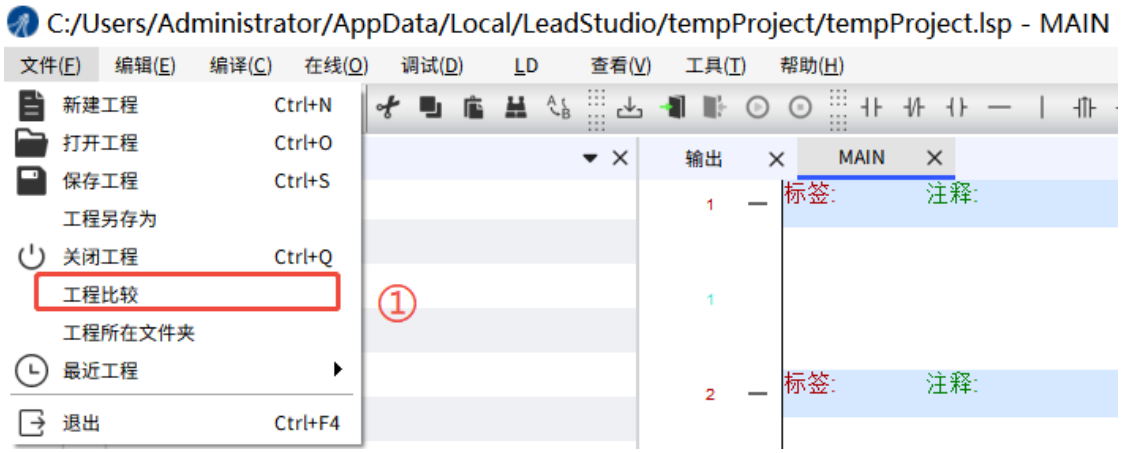
②选择对比工程类型（PLC内部程序或电脑本地程序）；

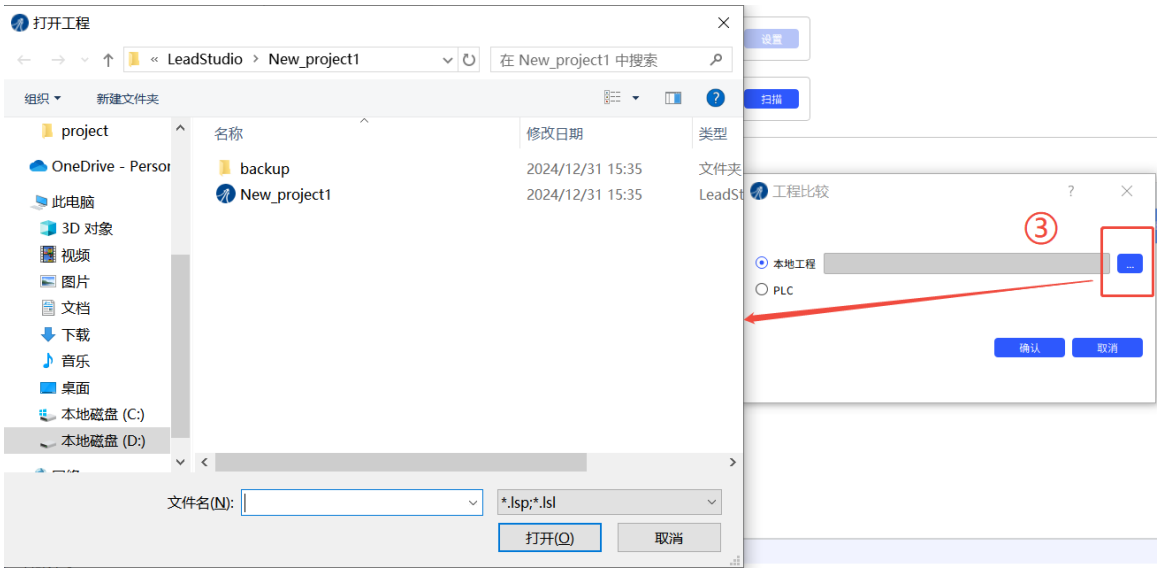
③若选择对比本地工程，需选择本地文件路径（选择对比在线PLC则需确保电脑与PLC已连接）；

④工程对比界面被打开，红色字体代表存在差异的部分，双击可进入详细对比界面

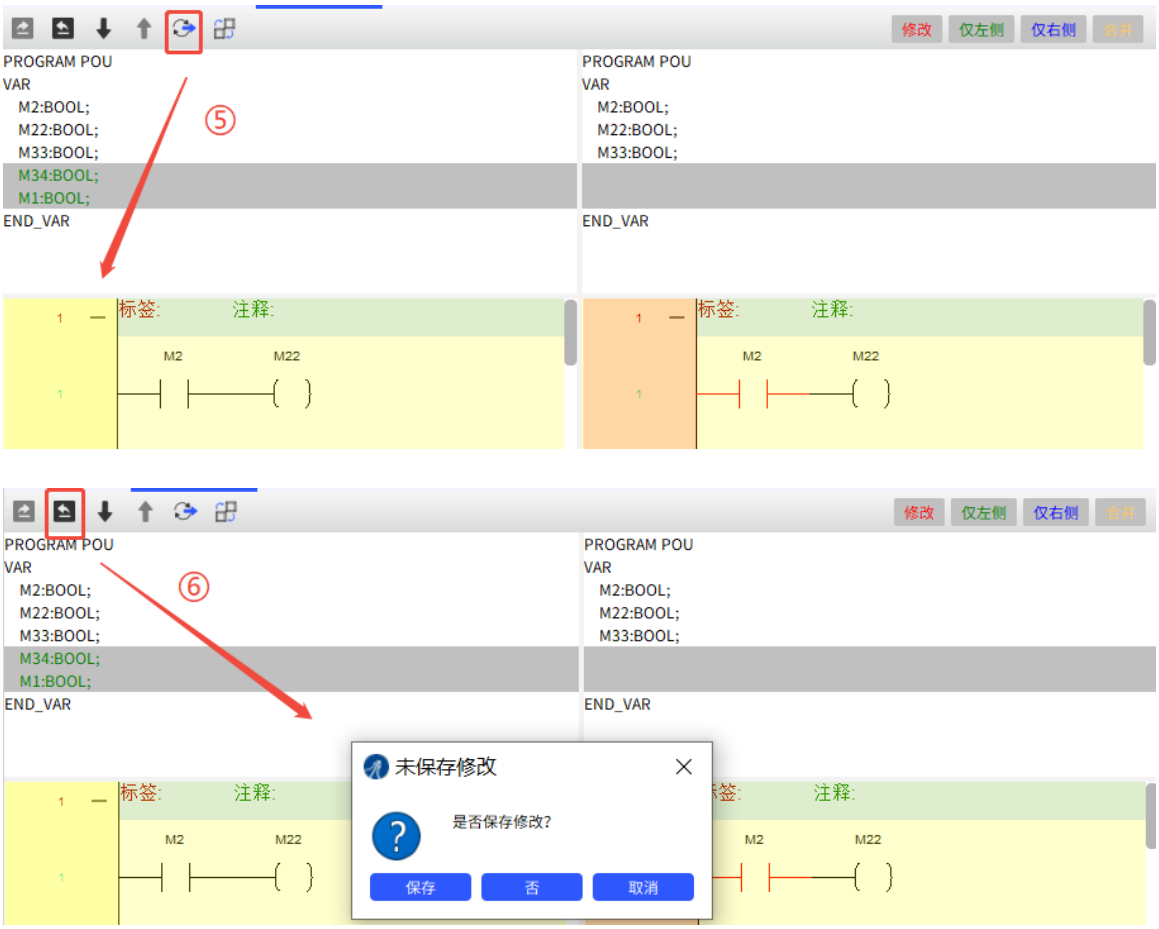
⑤在详细对比页面选中差异处，单击  进行行同步，或单击  进行块同步，以行同步为例。该差异处同步后以黄色色块标识。

⑥单击  ，打开是否需要保存同步内容提示框，点击“是”即可同步到当前工程





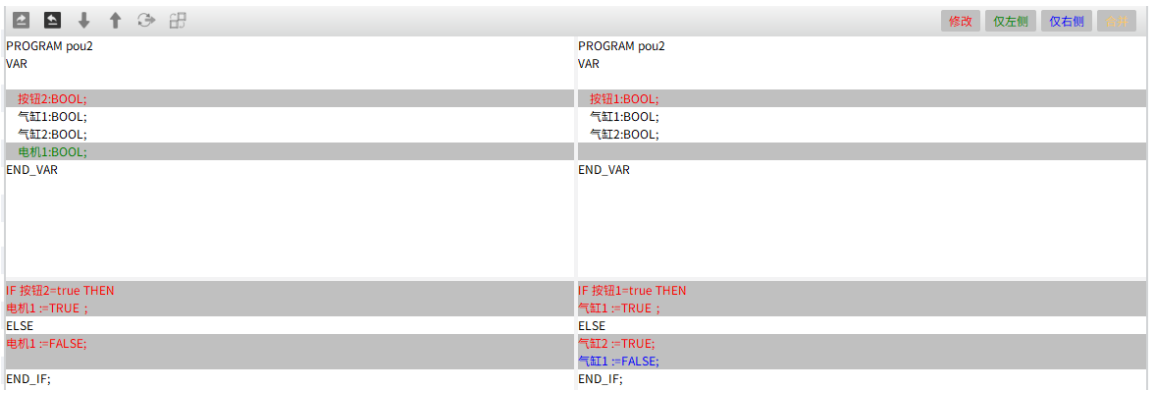

tempProject（当前工程）
配置和设置
设备(SC2_C32A4D)
设备诊断
网络配置
通讯配置
硬件配置
本地总线
运动控制轴
Axis_0
Axis_1
运动控制轴组配置
电子凸轮编辑器
任务配置
④
Main_task
POU
跟踪配置
程序
库管理
全局变量
数据类型
程序组织单元
POU (PRG)
pou2 (PRG)
监视列表
HMI配置
tempProject
Axis_0
pou1 (PRG)
PROGRAM POU
VAR
M2:BOOL;
M22:BOOL;
M33:BOOL;
M34:BOOL;
M1:BOOL;
END_VAR
标签:
注释:
M1
M22
M2
M34
M33



15.7.3 详细比较与同步说明

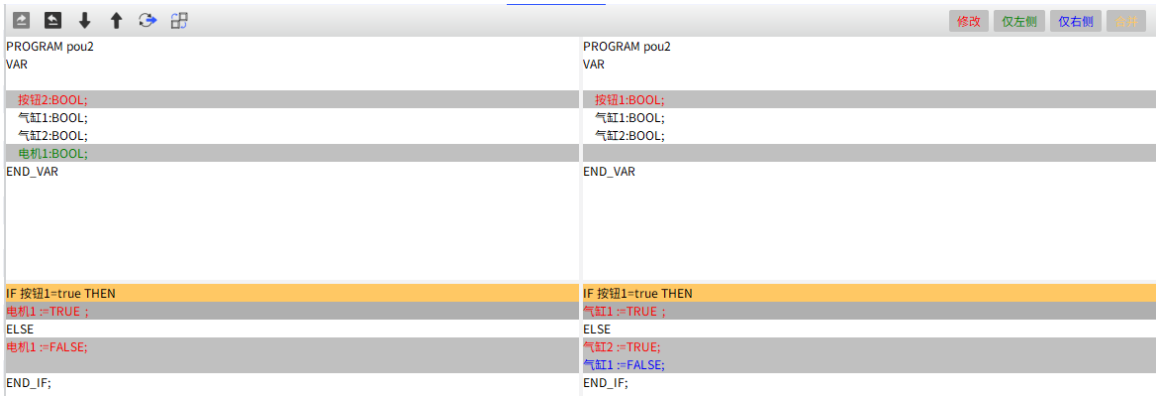
1. ST程序

ST程序之间的详细比较，会对程序的所有字符内容进行详细比较。程序文本中灰色底纹部分为两ST程序之间存在差异的部分，并用具体的字体颜色区分差异内容。详细比较示例如下：

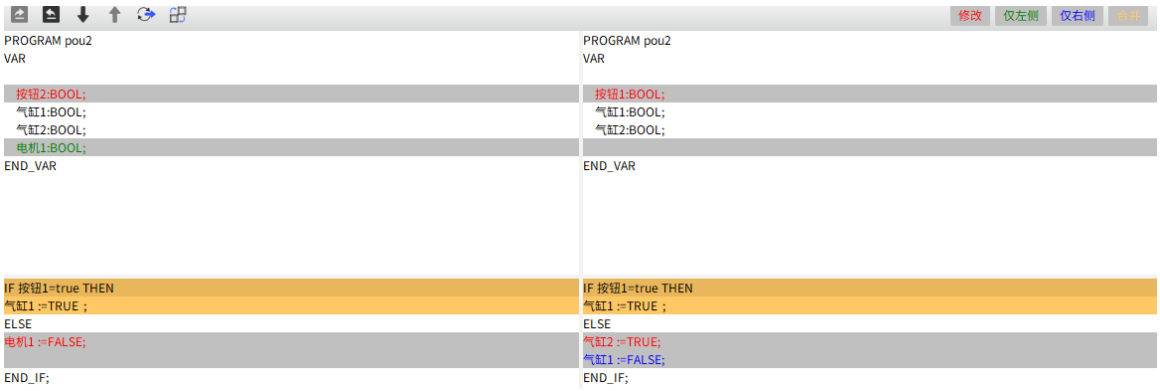


ST程序的行同步是将右侧比较工程文本光标所在行的内容合并到左侧当前工程中，块同步则是同步光标当前行以及前后连续差异行的内容，合并后同步部分内容将以黄色色块（底纹）标识。

ST行同步示例如下：

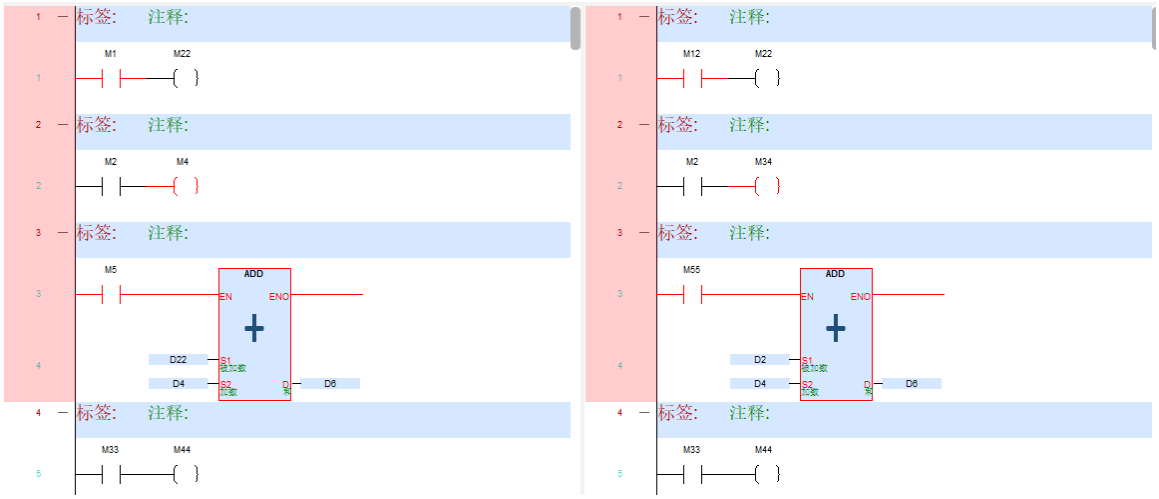


ST块同步示例如下：



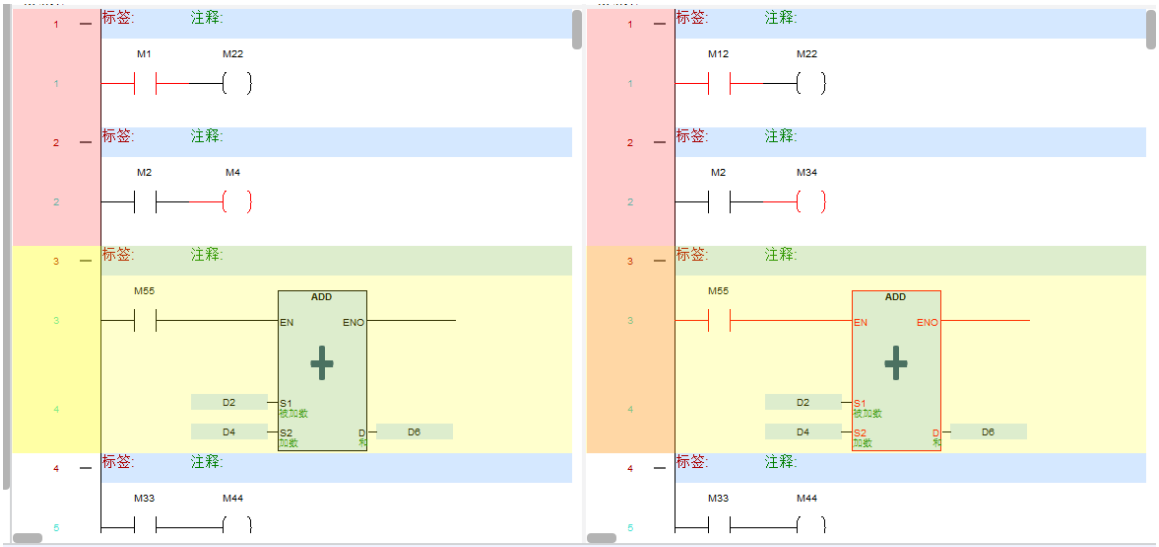
2.LD程序

LD程序之间的详细比较，以网络为单位进行详细比较。如果两工程对应的梯形图网络内容存在差异，将在相应的网络左侧标示不同的颜色，同时差异点标识会具体到行内元素，对具体存在差异的触点、功能块 （FB）、函数块（FC）等标示相应含义的颜色。详细比较示例如下：

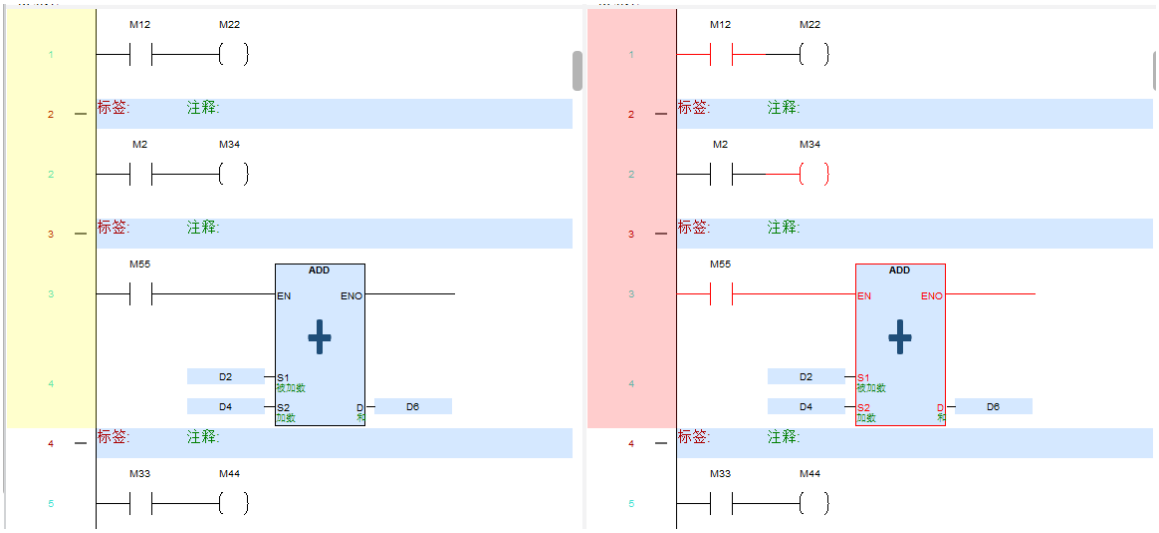


LD程序的行同步是对当前选择存在差异的网络进行内容的同步；块同步则是对当前选择存在差异的网络前后连续存在差异的网络进行同步操作。同步完成后，会合并比较工程相应的内容到当前工程中，网络同步内容左侧将以黄色色块标识。

LD行同步示例如下：



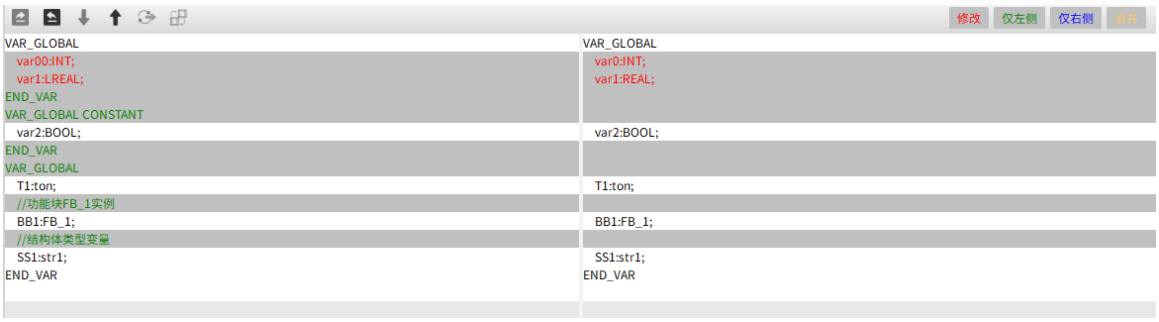
LD块同步示例如下：



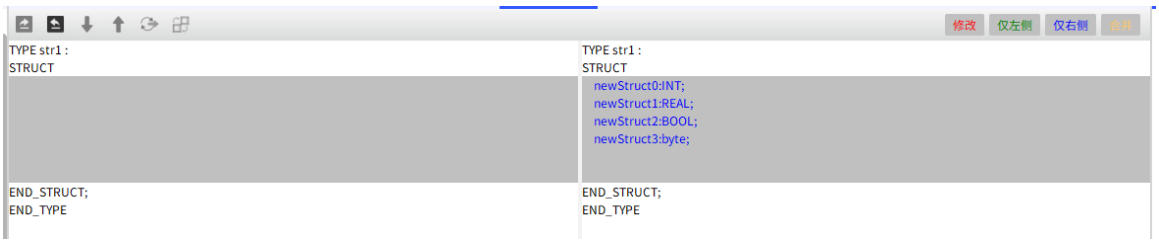
3. 变量表、数据类型

变量表、数据类型的详细比较同ST程序类似，会对变量声明文本的所有字符内容进行详细比较。底纹部分为两文本之间存在差异的部分，并用具体的字体颜色区分差异内容。

变量表详细比较示例：



数据类型详细比较示例：



变量表、数据类型的同步操作同ST程序，参考ST程序的行同步、块同步操作

16 HMI 下载程序

LeadStudio支持通过HMI更新PLC工程，将U盘连接到HMI上，即可更新PLC工程。PLC “通过HMI更新工程”的下载文件，用户自行在LeadStudio软件中生成“.hmilsp”格式的下载文件即可。

HMI需要与PLC通过Modbus方式建立连接，RS232、RS485、网口等通讯方式均可。

目前，该下载方式只支持雷赛HMI。

详细使用介绍可参考下方链接中的例程：

http://kzcp.leisai.com/product%20literature/Data%20download/PLC/SC/project/SC2-C_project_LeadStudio.rar

17 授权码

授权管理功能，用于绑定 PLC 与用户工程，PLC 中的授权码，必须与用户工程的授权码对应，程序才能运行。使用U盘、SD卡、HMI等方式升级的程序，同样受此限制。

PLC与用户工程授权码不一致时，PLC日志中输出“授权码不匹配！程序停止！”信息。

软件版本要求：V2.7/V3.1及以上

17.1 注意事项

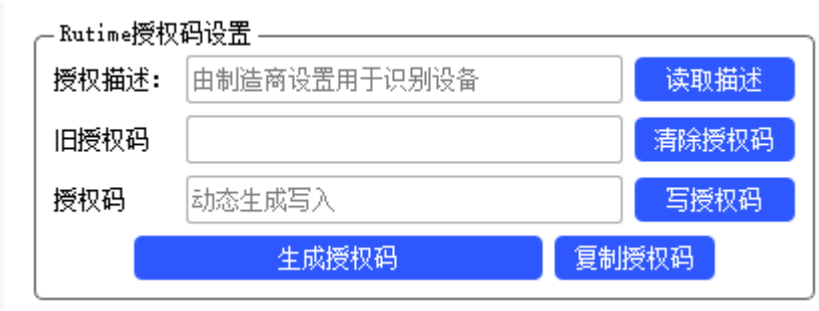
1）对工程设置授权码前建议对工程进行备份，防止遗忘授权码导致工程无法找回；

2）如 PLC Runtime 授权码遗忘，可通过初始化控制器清除 PLC Runtime 授权码；

3）如果工程设置了授权码，则 PLC Runtime 的授权码也必须设置成一致，否则程序无法运行，PLC 日志中报错“授权码不匹配！程序停止！”；

17.2 授权码功能使用介绍

17.2.1 Runtime 授权码设置



1）授权描述：用户定义的备注信息，支持中英文、数字，在执行写授权码时，同步写入，读取描述无需授权码；（简单来说，这是 PLC 密码提示）

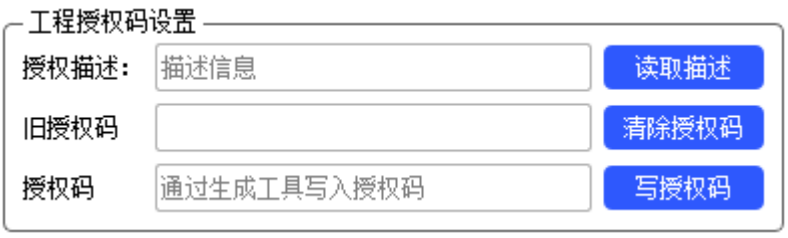
2）旧授权码：10 位数字，清除授权码、写授权码时使用，如果 Runtime 中存在授权码,要输入正确的旧授权码后才支持执行写授权码、清除授权码操作；（简单来说，这是 PLC 旧密码）

3）授权码：10 位数字，写授权码时使用，如果 Runtime 中存在授权码，则需要输入正确的旧授权码后才支持执行写授权码操作；（简单来说，这是 PLC 新密码）

4）生成授权码：生成随机的 10 位数字到授权码输入框中；（简单来说，这是让 LeadStudio 帮你随机生成 10 位密码）

5）复制授权码：复制授权码输入框中的内容；

17.2.2 工程授权码设置



1）授权描述：用户定义的备注信息，支持中英文、数字，在执行写授权码时，同步写入，读取无需授权码；（简单来说，这是工程密码提示）；

2）旧授权码：10 位数字，清除授权码、写授权码时使用，如果 Runtime 中存在授权码，则需要输入正确的旧授权码后才支持执行写授权码、清除授权码操作；（简单来说，这是工程旧密码）

3）授权码：10 位数字，写授权码时使用，如果工程中存在授权码，则需要输入正确的旧授权码后才支持执行写授权码操作；（简单来说，这是工程新密码）

17.2.3 其他说明

1）读取授权描述时，若描述为空，弹窗提示“授权描述为空”；

2）清除授权码时，若旧授权码校验失败，弹窗提示“旧授权码错误”

3）写授权码时，若旧授权码校验失败，弹窗提示“旧授权码错误，写授权码失败”；

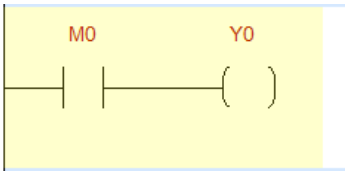
4）工程授权码只能通过清除授权码操作清除；

18 增量粘贴

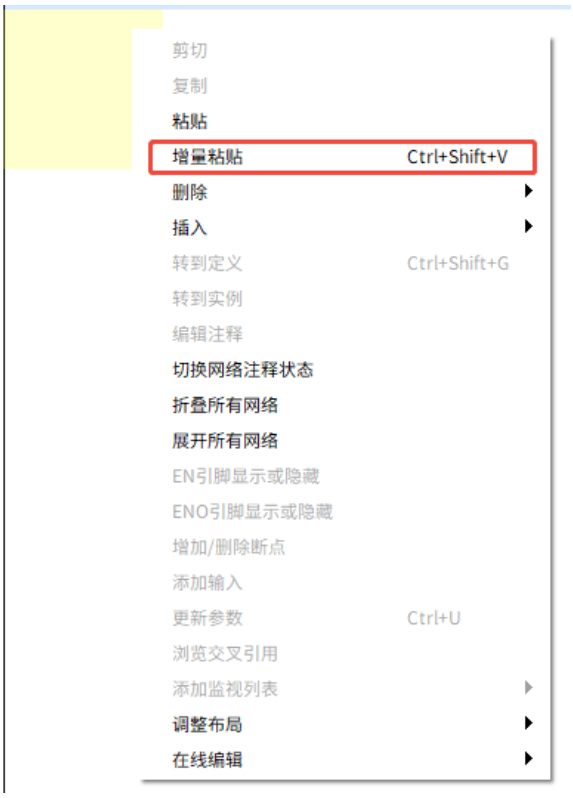
增量粘贴功能可在梯形图编程中对复制的元件进行多次连续粘贴操作；同时在粘贴的过程中可以对软元件号或数组下标进行指定操作。

1. 操作流程如下：

① 首先选中梯形图中需要增量粘贴的元件，执行复制操作（Ctrl+C 或者右键“复制”）。

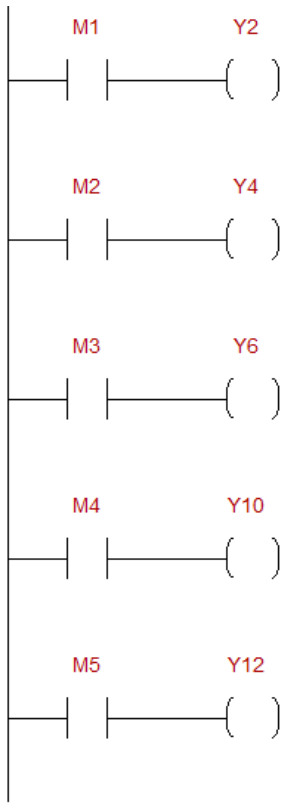


②选中需要增量粘贴的位置（选中位置为粘贴的第一行），在右键菜单中选择“增量粘贴”或者使用快捷键 Ctrl+Shift+V 键。



③在弹出的配置窗口中修改增量值和粘贴次数。

④单击“确定”后根据弹窗的配置结果进行粘贴操作。



2. 弹出窗口界面功能介绍：

①增量粘贴数(1~10)：设置粘贴次数，输入完成后点击窗口中的其他位置视为确认输入。

②增量后初值：支持编辑，可输入期望增量后的第一个元件，软件根据增量后的值

自动计算“增量数”，输入完成后点击窗口中的其他位置视为确认输入。

③增量后终值：不支持编辑，显示增量后的最后一个元件。

④增量数：支持编辑，可设置每次粘贴需要对目标元件进行增量的值，输入完成后点击窗口中的其他位置视为确认输入，寄存器的增量以一个寄存器为单位。

⑤位操作增量：当目标元件为软元件位操作时，此处选中后增量操作将对目标原件的位操作进行增量。

⑥批量设置增量数：可以批量设置“增量数”，输入完成后点击窗口中的其他位置视为确认输入。

3. 注意事项

①最多支持 100 个目标元件进行增量粘贴，若大于此数量，则触发“增量粘贴”时会提示报错

②除了位元件、寄存器元件、带下标的数组元素、FB 实例名、变量以及寄存器的位访问外的其他元素不支持增量粘贴，若复制内容中包含这类元素，则仍保留源数据进行粘贴，其他元件仍可以进行增量粘贴。

③若数组下标采用变量，则下标不增量，保留源数据进行粘贴，如当前复制的元件为“Test[i]”，则增量粘贴后的结果都为“Test[i]”。

④复制 FB 实例进行增量粘贴时，自动在当前实例名后增加“_+序号”，如当前复制的实例名为“Test”，则第一个增量粘贴的实例名为“Test_1”。（若为 FB 数组，则按照数组的规则进行增量粘贴）。

⑤数组变量执行增量粘贴时，若出现增量后数组下标大于数组范围的情况，则在点击“确定”时，弹出提示“数组下标超出定义的范围，不支持增量粘贴”。

客户咨询中心
目录索取·技术咨询·产品解惑
400-885-5521 销售热线
400-885-5501 技术热线

更多最新的雷赛资讯，请扫码关注！
公众号 视频号

成就客户 共创共赢

深圳市雷赛智能控制股份有限公司
China Leadshine Technology Co., Ltd.

深圳市南山区沙河西路3157号南山智谷产业园B栋15-20层
邮 编:518052
电 话:400-885-5521
网 址:www.leisai.com E-Mail:marketing@leisai.com

上海分公司
上海市嘉定区金园五路601号

北京办事处
北京市大兴区绿地启航国际3号楼1109室

山东办事处
济南市天桥区滨河商务中心D座2003室

苏州办事处
江苏省苏州市苏州工业园区金尚路1号仙峰大厦南楼7层

合肥办事处
合肥市蜀山区潜山路与高河东路交口绿地蓝海大厦A座1209室

武汉办事处
湖北省武汉市东湖新技术开发区长城园路2号海贝孵化器209

温州办事处
浙江省温州市瓯海区潘桥街道宁波路阳光城愉景嘉园8幢2604

青岛办事处
山东省青岛市城阳区金日紫都小区12号楼1单元301室

杭州办事处
浙江省杭州市余杭区瓶窑镇桂花溪园(南区)2幢1单元402