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

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

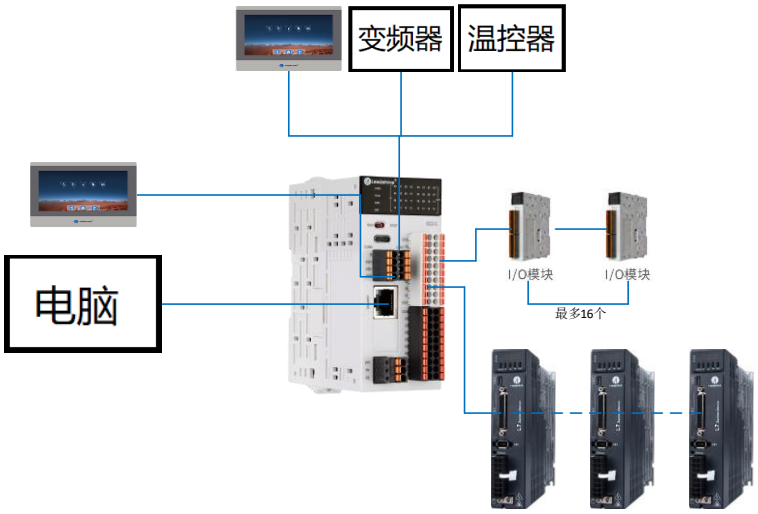
SC2 系列控制器电源电压为 DC24V，需要外部开关电源提供 24V 输入。建议给控制器独立配开关电源供电，避免与其它传感器或设备一起供电。

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。

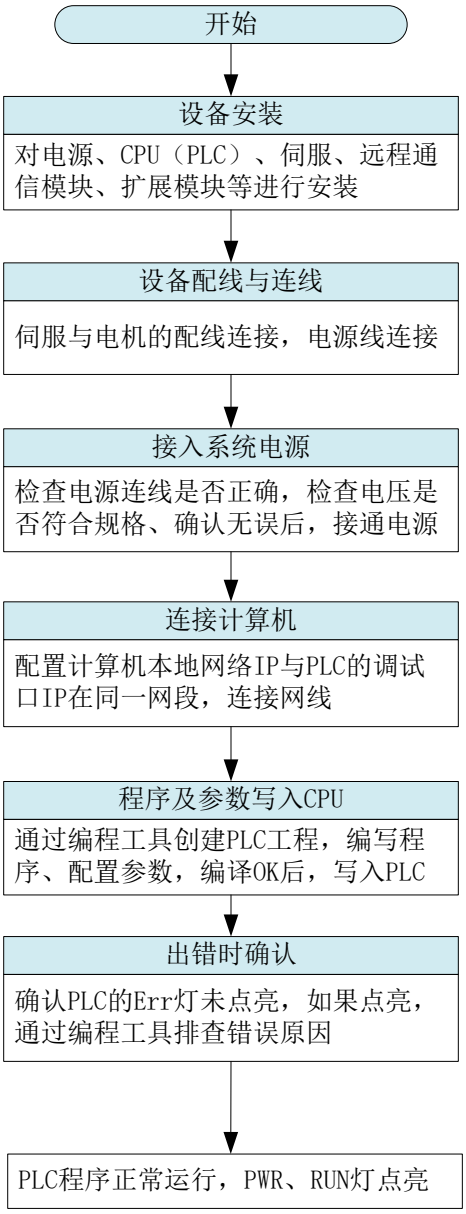
开关电源的负载能力参照下表 SC 系列产品的最大功耗：

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



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

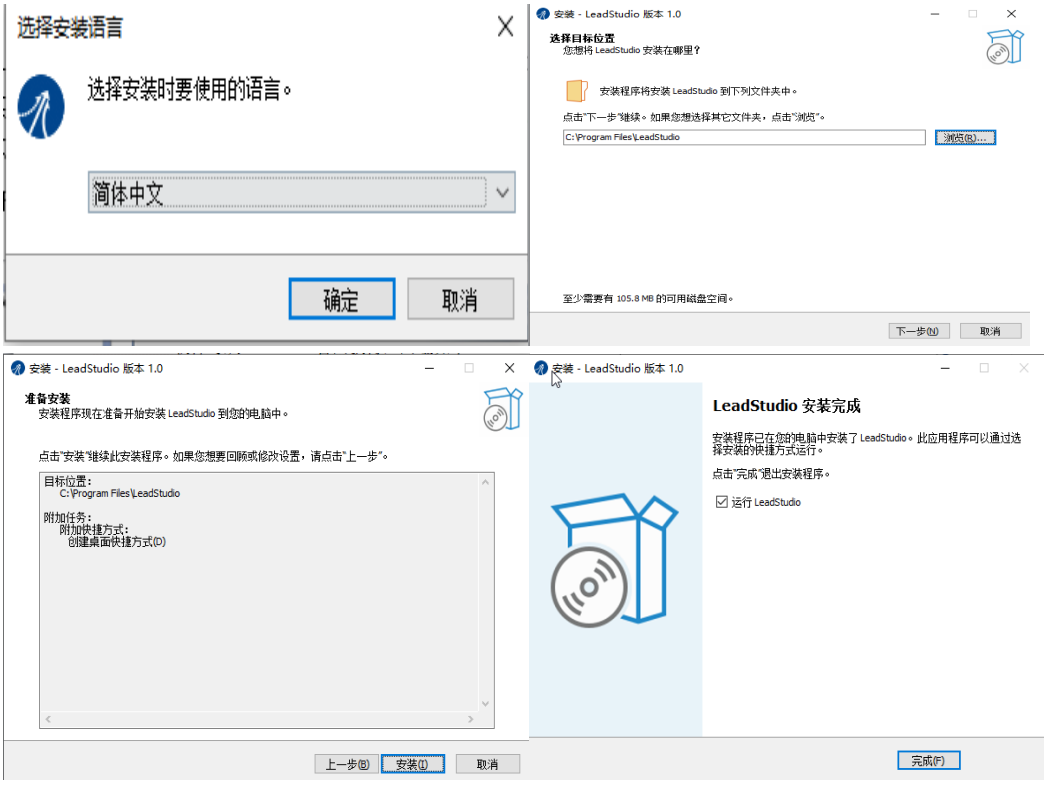
1.2.1 软件的获取与安装

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。



1.2.2 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：

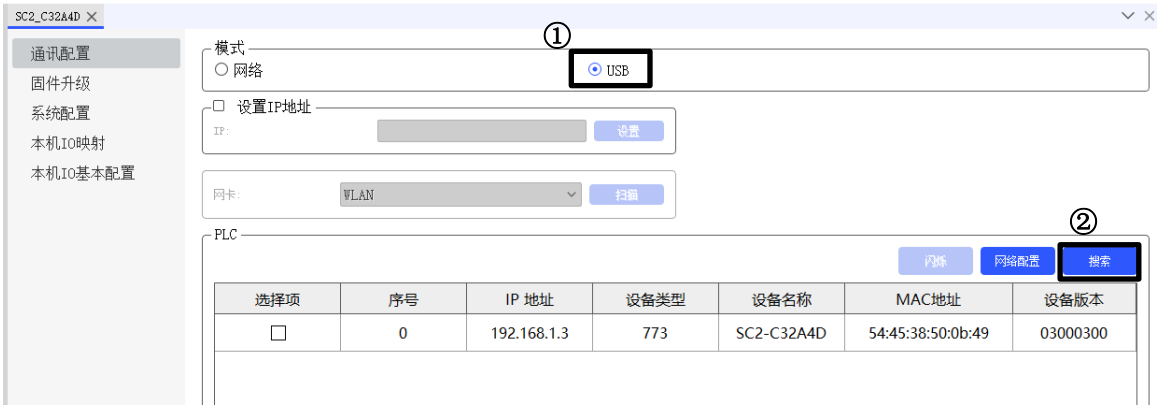
2）选择网络模式-----选择以太网网卡-----点击搜索，即可扫描出 PLC，如下图所示：

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：

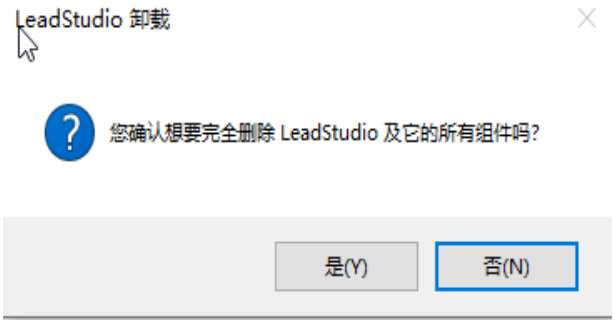


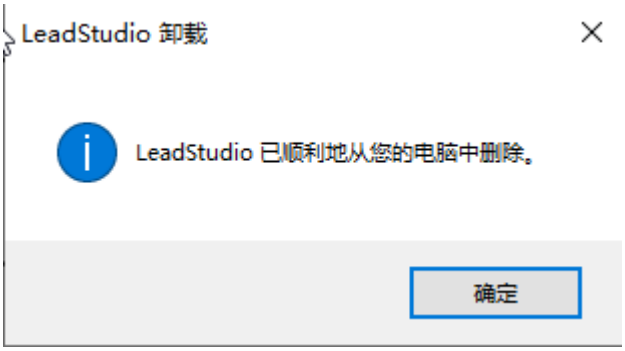
3）勾选扫描出来的 PLC，即可连接到 PLC，如下图所示：

1.2.3 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。


Microsoft Visual C++ 2013 Redistributable (x86)...
17.2 MB
2022/10/25
企业微信
2022/10/20
LeadStudio 版本 1.0
106 MB
2022/10/19
1.0
修改
卸载
Microsoft .NET SDK 6.0.305 (x64)
849 MB
2022/10/17


LeadStudio 卸载
您确认想要完全删除 LeadStudio 及它的所有组件吗?
是(Y)
否(N)

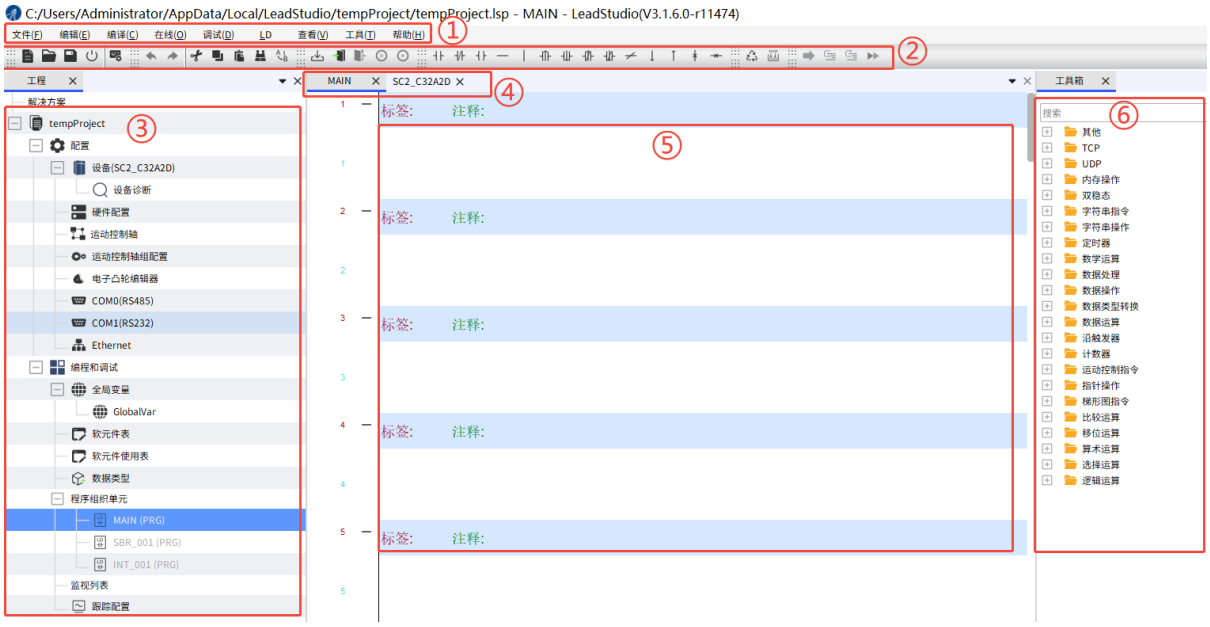

LeadStudio 卸载
LeadStudio 已顺利地从您的电脑中删除。
确定

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

包含文件操作、编辑操作以及在线操作等工程通用操作

② 快捷工具栏

包含工程保存、登录运行以及梯形图元素等快捷键操作

③ 设备树

包含硬件配置和用户编程区域

④ 页标签

可跳转已打开的工程页面

⑤ 梯形图编辑区

用户程序的编辑区域

⑥ 工具箱

存放指令的工具箱，用户可通过工具箱查找指令并拖放到程序中

左侧设备树详细如下：

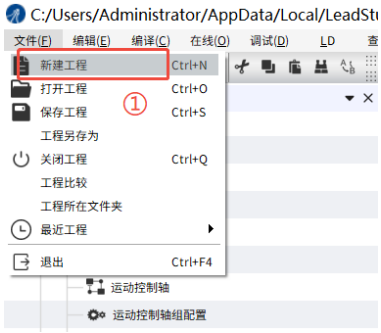
①设备配置：主要包含 PLC 的硬件配置（PLC 模块的配置）、通讯配置（PLC 串口和网口的通讯配置）以及轴运动配置（单轴、轴组以及凸轮配置）等

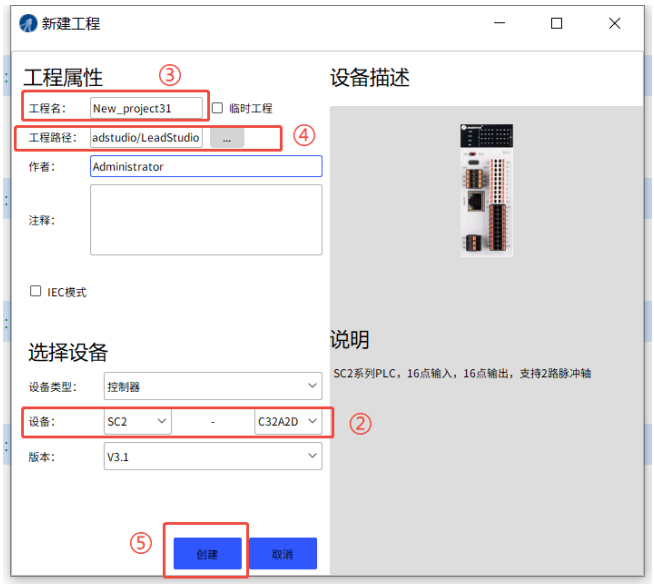
②编程调试：用户编程及调试的区域，主要包含程序组织单元、全局变量以及监控列表等

2.2 编程流程

1) 新建工程

打开 LeadStudio 软件-----新建工程-----选择对应的 PLC 型号-----设置工程名称-----选择保存路径----创建，如下图所示。





2) 编写程序

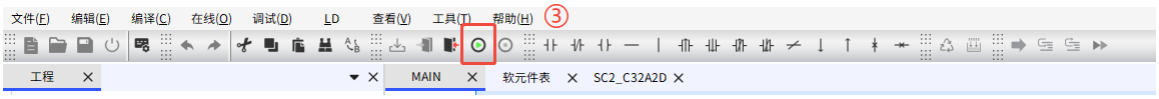
打开梯形图编辑区-----光标选中编辑位置-----键盘输入命令

说明：键盘命令输入与传统日系命令一致

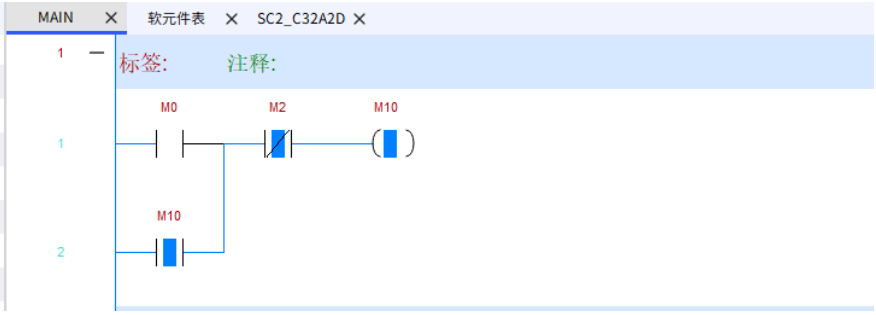
2.3 登录及运行

1）连接 PLC，参考章节 1.2.2

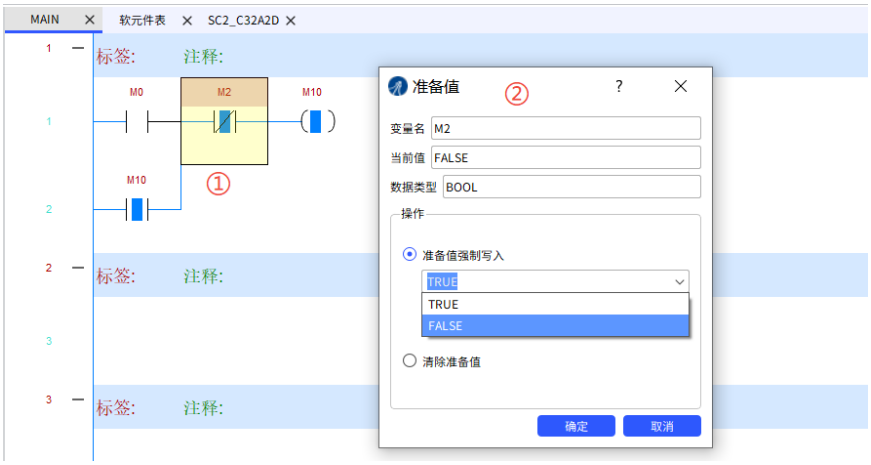
2）登录----下载-----运行，如下所示：



3）登录后可在监控梯形图状态



4）双击对应元件，可修改当前元件状态



3 编程基础

3.1 软元件

3.1.1 软元件范围

软元件种类：X ,Y, M, S, D, B, W, R

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

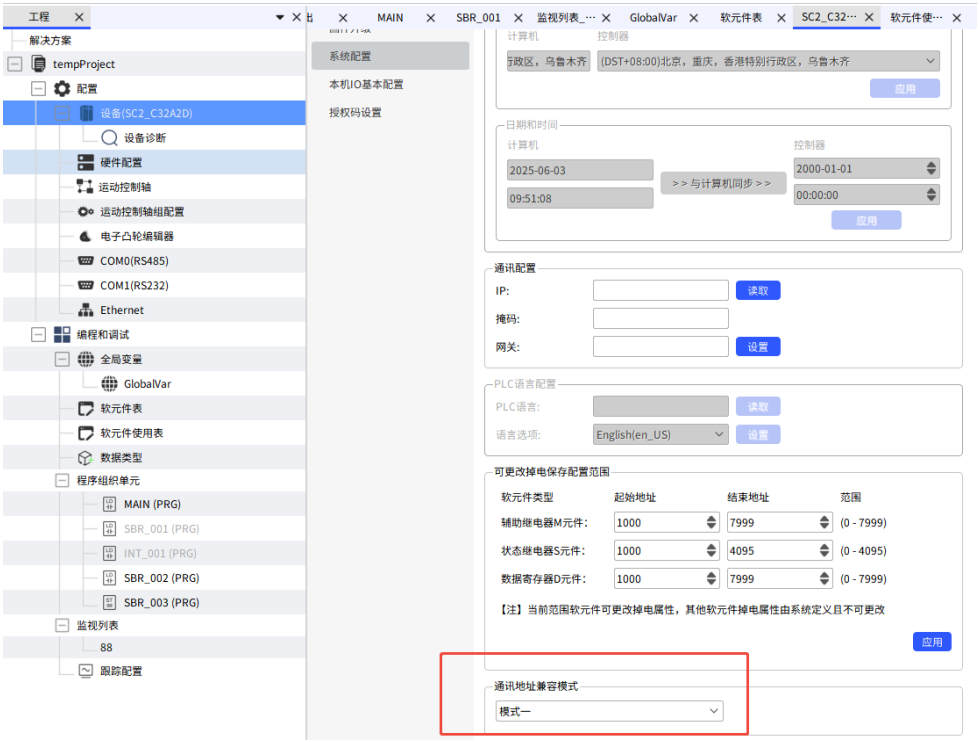
③特殊软元件

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

3.1.2 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），不同模式对应的软元件通讯地址有所不同；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

3.1.3 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

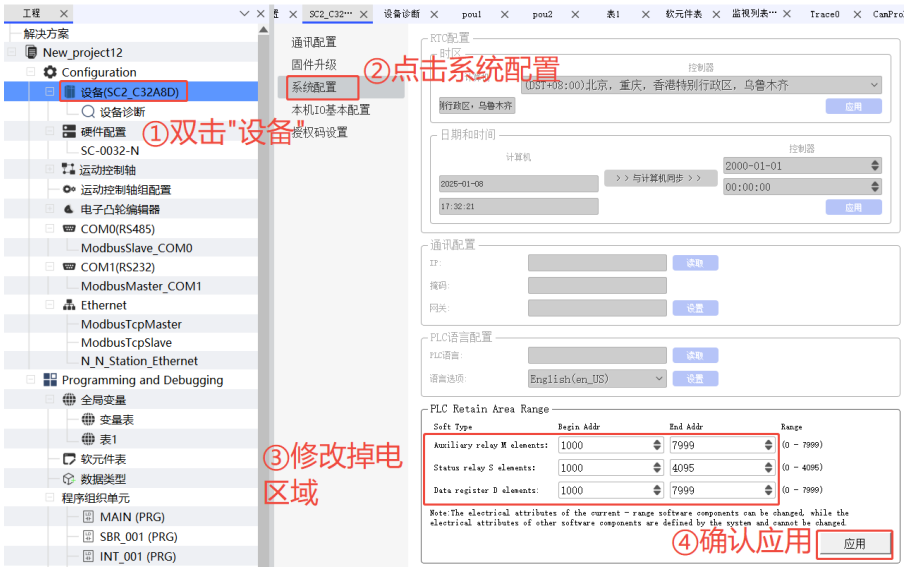
B: B1000-B32767 为保持继电器

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：

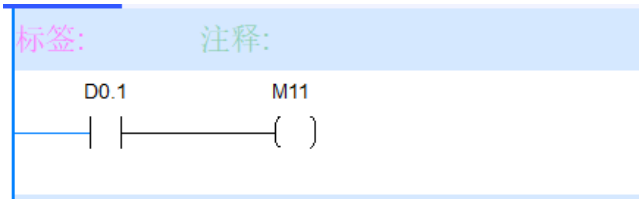


说明：①只支持软元件 S,M,D 的掉电区域修改。②软元件 M 和 S 的掉电范围必须为 8 的倍数（起始地址为 8 的倍数，且确保掉电范围内的个数也为 8 的倍数），否则将无法设置成功。

3.1.4 软元件使用技巧

1）字元件位访问

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

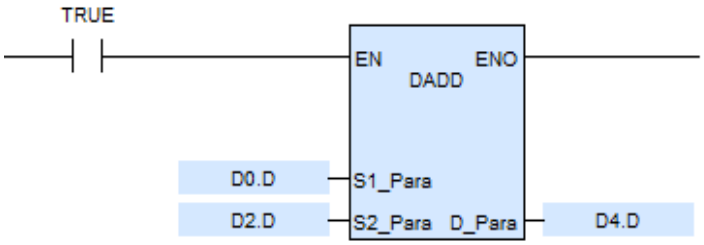
2）采用后缀区分寄存器数据类型

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：



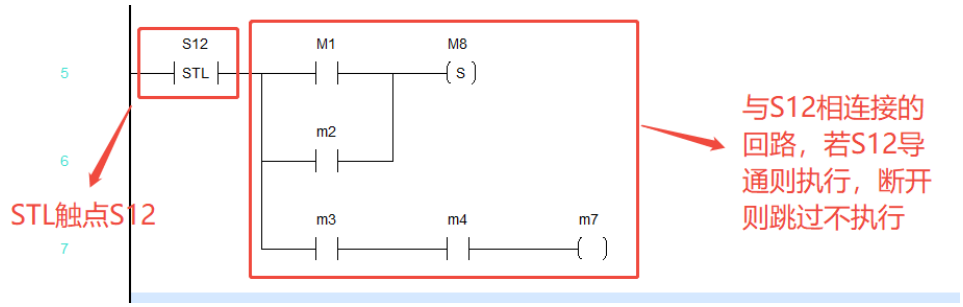
表示 32 位整型寄存器（D0 , D1）和(D2 , D3)相加，结果赋值给另一个 32 位整型寄存器(D4 , D5)

3.1.5 软元件 S 与步进指令 STL

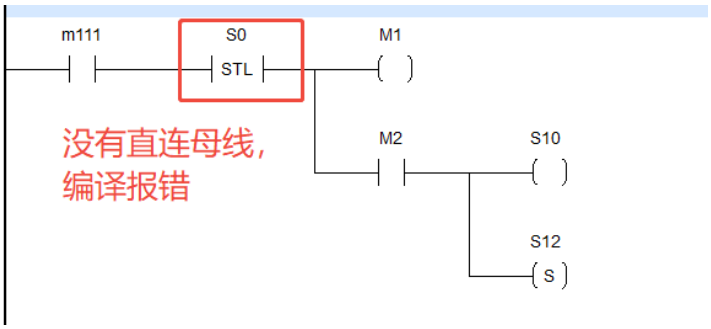
在基础指令中，S 可被当作 M 使用；在 STL 指令中，S 作为步进状态。

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：

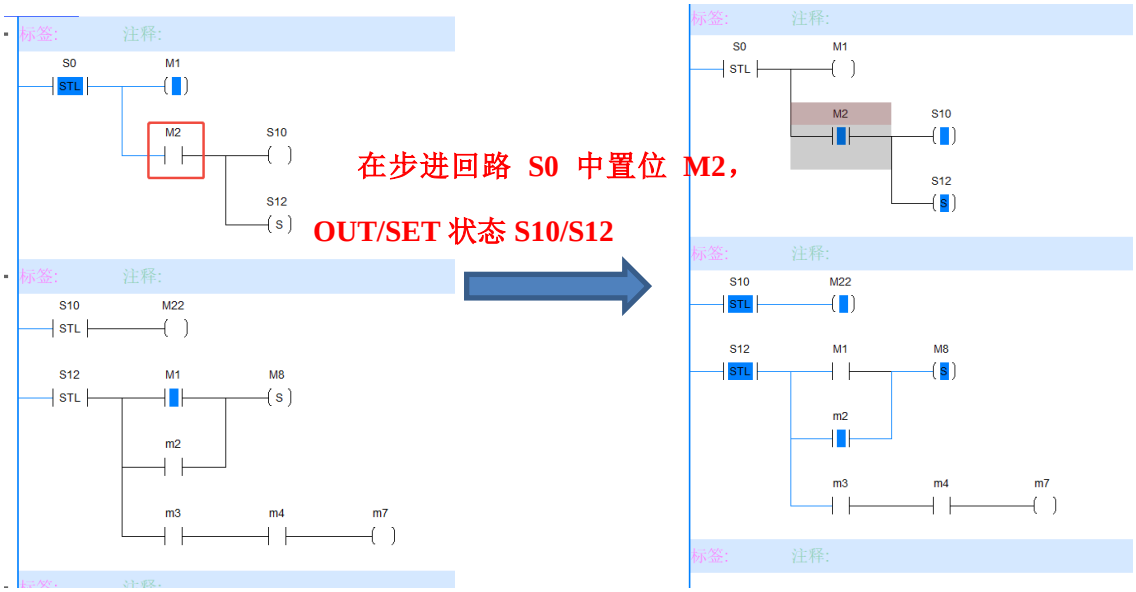


注：STL 触点 S 必须直连母线，否则将编译报错，如下所示：



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

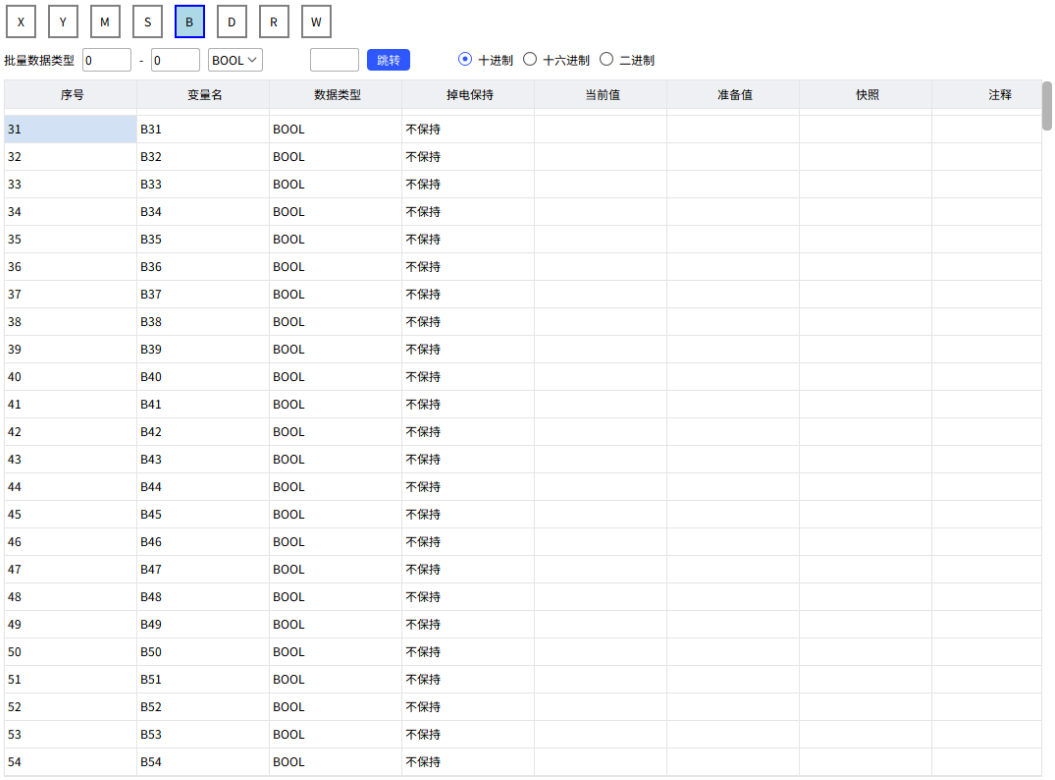
3.1.6 软元件表

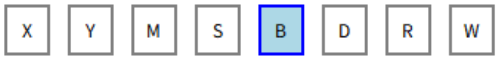
软元件表包含 PLC 所有的内部软元件，用户可在该表对所有软元件编辑注释，在线

情况下还可监控所有软元件的状态。

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：



通过点击表格上方的 

可切换软元件表显示的软元件类型

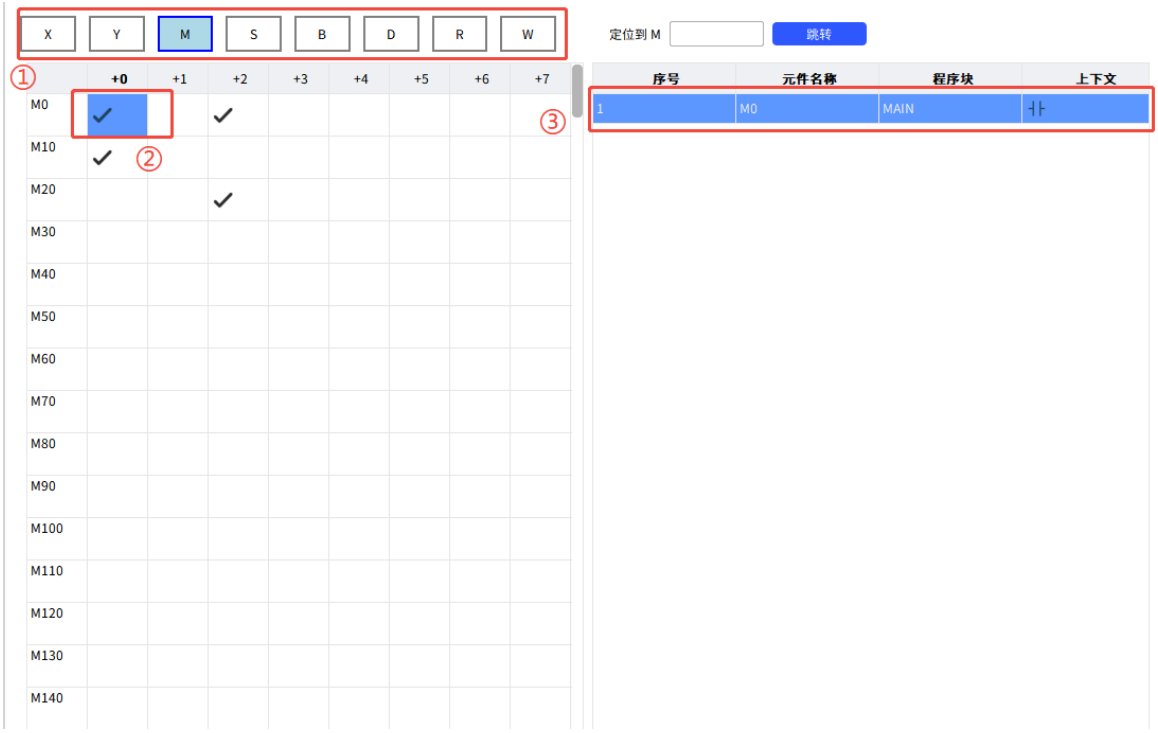
3.1.7 软元件使用表

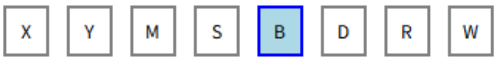
LeadStudioV3.1 以上版本支持软元件使用表功能，用户可通过软元件使用表查看

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

3.2 程序组织单元

3.2.1 概述

程序组织单元是用户编写程序的单元，在设备树中的位置如下：

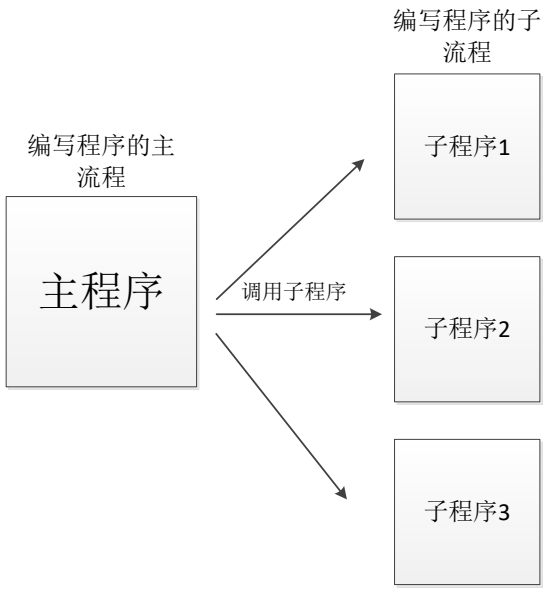
程序组织单元分为程序（PRG）和功能块（FB、FC）两大类，其中的程序（PRG）又可分为主程序、子程序以及中断程序;

新建工程默认添加一个主程序 MAIN,一个子程序 SBR_001,以及一个中断程序 INT_001

3.2.2 程序(PRG)

主程序（MAIN）

主程序为系统启动后一直循环运行的程序，是所有用户程序的核心中枢；编程时一般会把主流程和子流程的逻辑分别放在主程序和子程序中，并通过主程序去调用子程序来实现主流程到子流程的逻辑跳转



在整个工程中有且仅有一个主程序，用户无法添加或删除主程序；支持梯形图和 ST 语言编程（ST 语言通过在梯形图中插入执行块使用,参考下面章节 3.5.3）

主程序属性设置

用户可通过右键“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：

看门狗：防止程序跑飞或陷入死循环导致系统故障

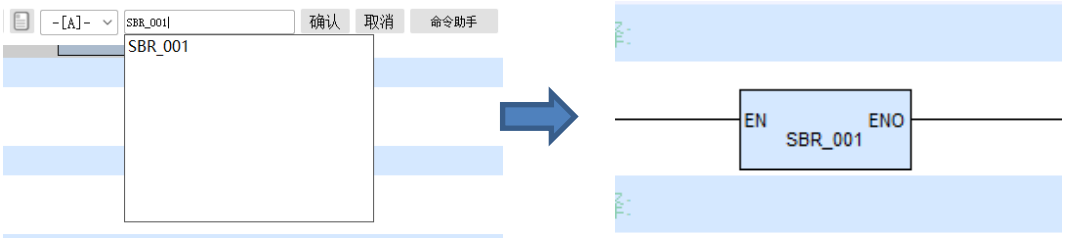
扫描周期：程序由上至下执行一次的固定周期时间，不勾选时则以惯性滑行来执行程序（即每次程序执行的扫描时间都不一致,根据程序实际执行时间而定）

子程序（SBR_）

子程序需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；合理利用子程序编程可以显著提高 PLC 的运行效率。

子程序调用方法

在主程序的键盘命令输入子程序名称即可实现调用，操作方便快捷



子程序支持嵌套调用，即主程序调用子程序 A，子程序 A 又调用另一个子程序 B，嵌套调用的方法同主程序调用一致。

子程序添加方法

子程序可添加多个，支持梯形图与 ST 语言编程，添加步骤如下：

①右键“程序组织单元”，选择“插入子程序”

②命名子程序名称，并选择子程序的编程语言

③添加成功，新增的子程序显示在程序组织单元下

子程序命名规则：SBR_序号，其中 SBR_为固定前缀不可更改，序号可通过重命名来更改。

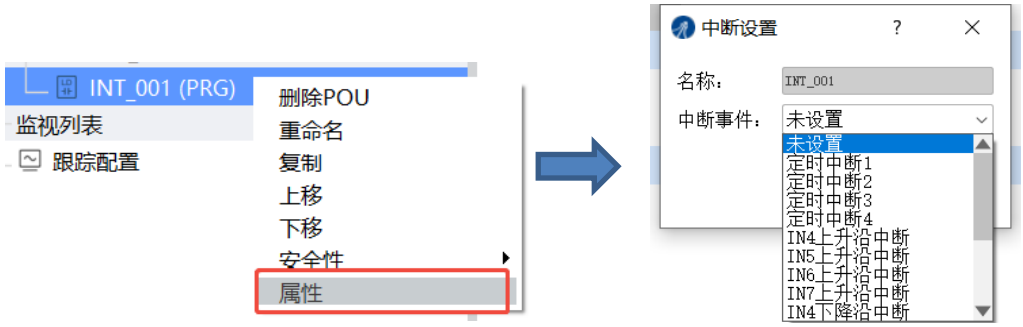
中断程序（INT_）

通过中断条件触发执行的程序，应用于需要立即执行程序予以响应的场合。

根据中断条件的不同，可分为三种类型中断：定时中断、外部输入中断以及高速比较中断。

中断条件设置方法

右键“INT_001(PRG)”，选择“属性”，下拉菜单选择中断触发事件，如下所示：



中断程序添加方法

①右键“程序组织单元”，插入中断程序

②命名并选择程序的编程语言

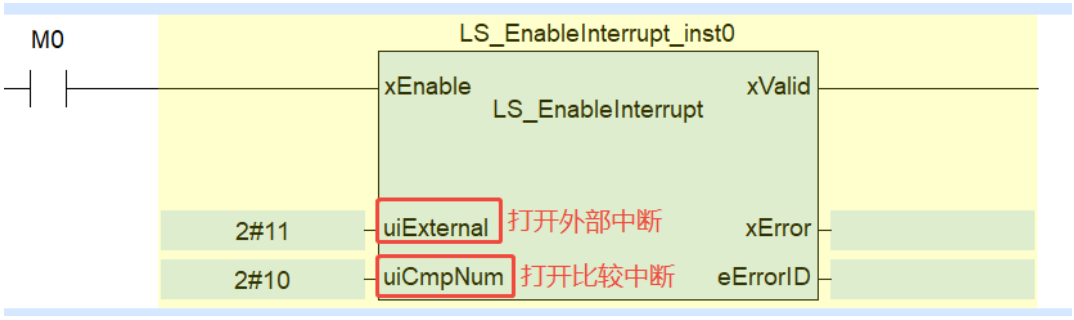
③添加成功，新增的中断程序显示在程序组织单元下

中断程序命名规则：INT_序号，其中 INT_为固定前缀不可更改，序号可通过重命名来更改。

说明事项

①定时中断只需设定好中断时间，PLC 运行后便会按照设定的时间定期执行中断程序

②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：



例如 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

③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等于指令设定的比较值时触发比较中断，详细参考 LeadStudio 指令手册

3.2.3 功能块 FB、函数 FC

用户可通过 FB、FC 块实现算法和工艺库的封装，避免大量重复性的编程工作，显著提高用户编程效率

FB/FC 添加方法

右键“程序组织单元”，插入 FB 或者 FC，选择编程语言以及数据类型，如下所示：

成功添加后，FB/FC 块会在显示在项目树的程序组织单元下

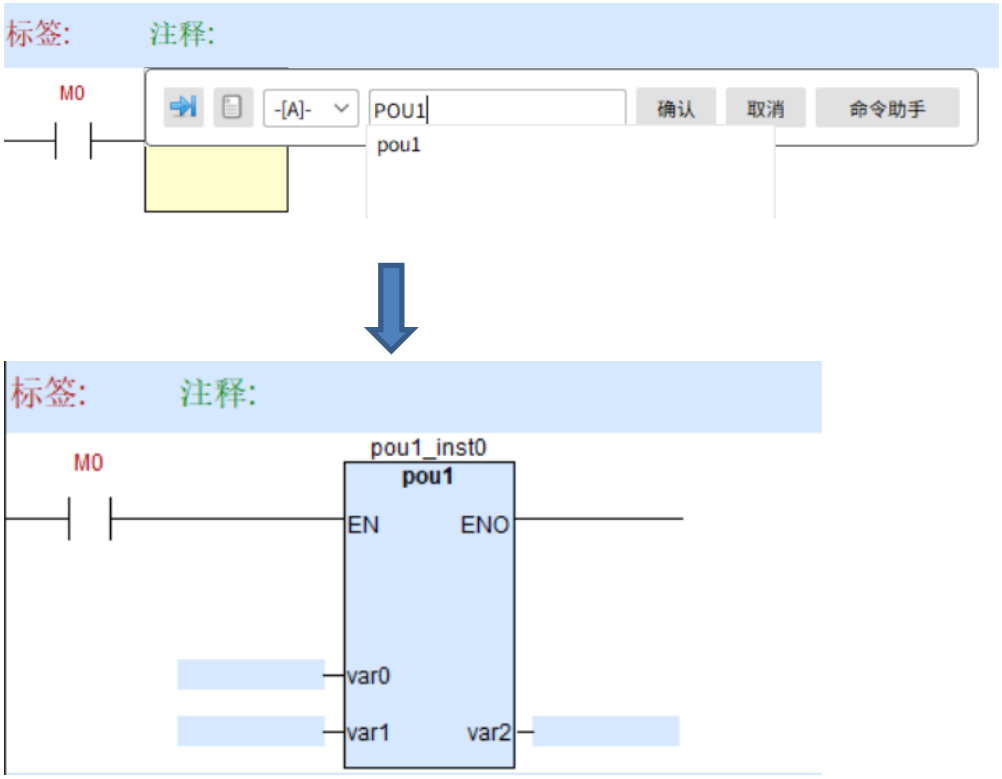
FB/FC 编程

FB/FC 需先声明局部变量，再进行程序编写；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。

FB/FC 调用方法

键盘命令输入 FB/FC 名称即可实现调用



说明：功能块 FB 调用时还需进行实例化，此处不赘述实例化相关内容

3.3 全局变量

变量是程序的基本单元，用户除了使用传统的软元件来完成编程外，也可通过创建变量来替代软元件完成编程。

变量相对于传统软元件，在编程方面会更加灵活，用户无需提前考虑软元件分配问题，灵活使用变量编程可显著提高用户的编程效率以及程序的可读性

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

添加变量表

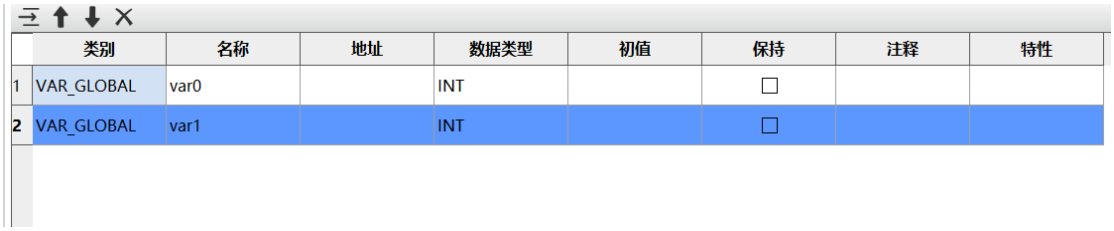
全局变量以表格的形式存在，用户可添加多个全局变量表。

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：

全局变量声明

全局变量有两种声明方法

方法一：在全局变量表中声明



	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		☐		

点击左上角的新建按钮  ，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

方法二：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，全局变量表自动中生成变量

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

变量地址映射

用户自定义的变量可以映射到软元件地址，从而实现与第三方设备的Modbus通讯。如下图所示，变量“var0”映射到软元件 D0；

说明：地址映射会自动占用连续的软元件，例如“var0”若为 DINT 类型，则自动占用软元件(D0,D1)

3.4 数据类型

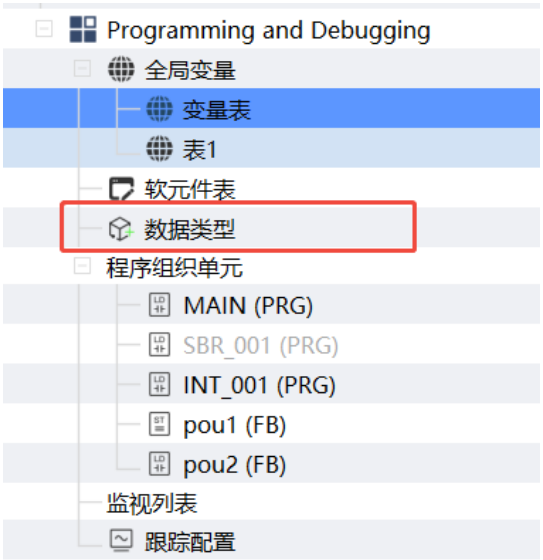
LeadStudio 基本数据类型如下

数据大类	数据类型	关键字	占用内存位数	取值范围
布尔类型	布尔	BOOL	8 bit	FALSE(0)或 TRUE(1)
整型	字节	BYTE	8 bit	0～255
	字	WORD	16 bit	0～65535
	双字	DWORD	32 bit	0～4294967295
	长字	LWORD	64 bit	0～(2^64-1)
	短整型	USINT	8 bit	-128～127
	无符号短整型	USINT	8 bit	0～255
	整型	INT	16 bit	-32768～32767
	无符号整型	UINT	16 bit	0～65535
	双整型	DINT	32 bit	-2147483648～2147483647
	无符号双整型	UDINT	32 bit	0～4294967295
	长整型	LINT	64 bit	-2^63～(2^63-1)
	无符号长整型	ULINT	64 bit	0～(2^64-1)

浮点	单精度	REAL	32 bit	1.175494351e-38～3.402823466e+38
	双精度	LREAL	64 bit	2.2250738585072014e-308～1.7976931348623158e+308
字符串	字符串	STRING	一个字符占用8bit	
	宽字符串	WSTRING	一个字符占用16bit	
时间	时间类型	TIME	32bit	T#0d0h0m0s0ms ～T#49d17h2m47s295ms
	长时间类型	LTIME	64bit	LTIME#0NS～LTIME#213503D23H34M33S709MS551US615NS
	日期类型	DATE	32bit	D#1970-1-1～D#2106-2-7
	一天的时间类型	TOD	32bit	TOD#0:0:0～TOD#23:59:59.999
	日期和时间类型	DT	32bit	DT#1970-1-1-0:0:0～DT#2106-2-7-6:28:15

除基本数据类型外，LeadStudio 还支持用户自定义数据类型，包括结构体和枚举

自定义数据类型在设备树中的位置如下：



添加自定义数据类型

可添加的数据类型：结构体、枚举

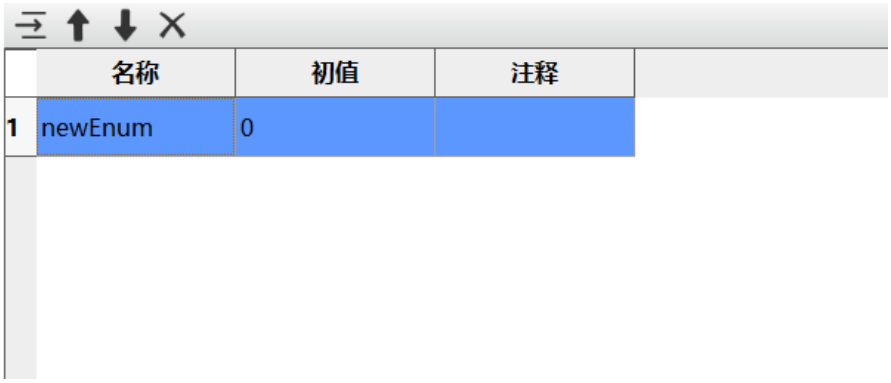
添加方法：右键“数据类型”，选择新建数据类型，输入类型名称，如下图所示，

编辑自定义数据类型

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

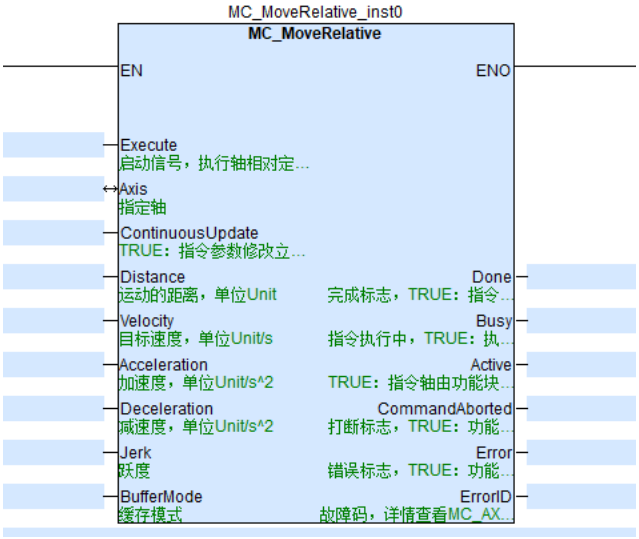
②枚举编辑



如上图所示，枚举编辑画面同样以表格形式展现，通过左上角按钮  可添加或删除枚举成员

3.5 指令功能块

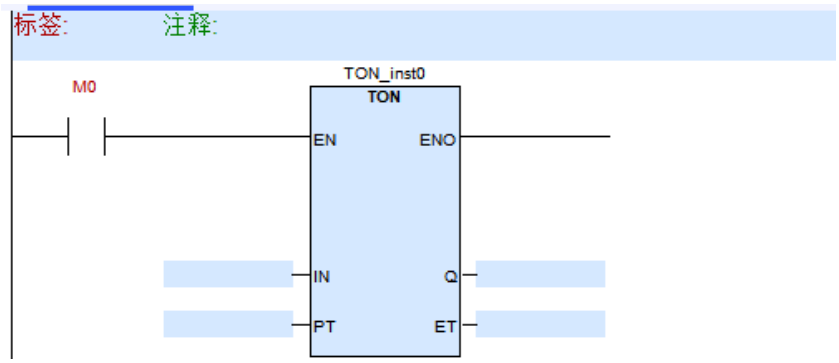
LeadStudio 的指令以“块”的形式展现，所以又称功能块；对比传统日系指令的展现形式，“块”的形式会更加直观，所有与指令相关的参数都可在“块”上展现，如下图的轴运动功能块，所有轴运动相关的参数都可体现在“块”上。



3.5.1 自动实例化功能块

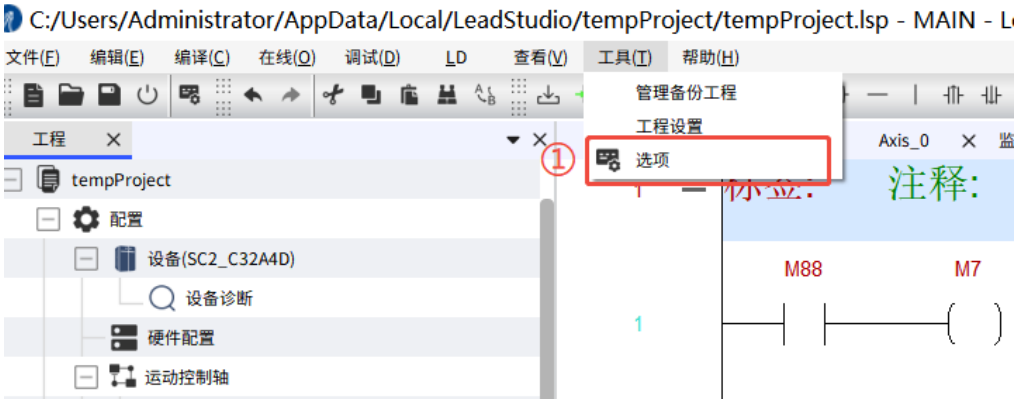
LeadStudio 的 FB 功能块与传统日系指令的主要区别为是否需要实例化，考虑实例化概念对于习惯日系编程的用户来说难以理解，LeadStudio V3.1 及以上版本在功能块实例化方面做出优化，用户在梯形图新增 FB 功能块时，系统会自动帮用户创建实例到“默认变量表”，无需用户手动实例化。

在梯形图输入功能块名称时，系统将自动生成实例名；用户可以不用考虑实例化的步骤，只需输入指令名称即可，如下图所示：



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；

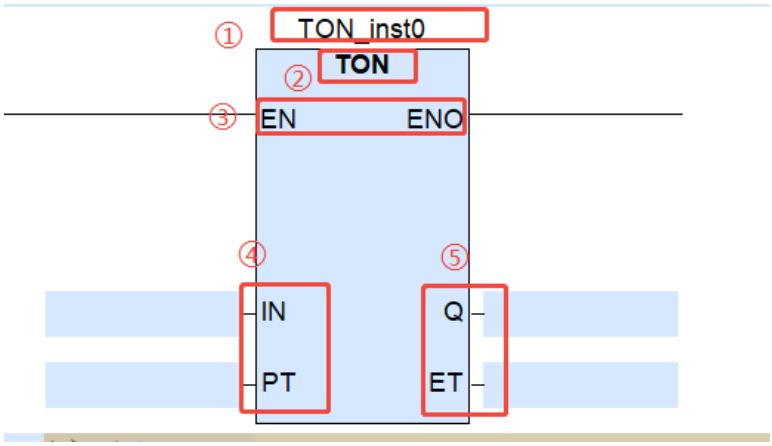


3.5.2 常用功能块

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。

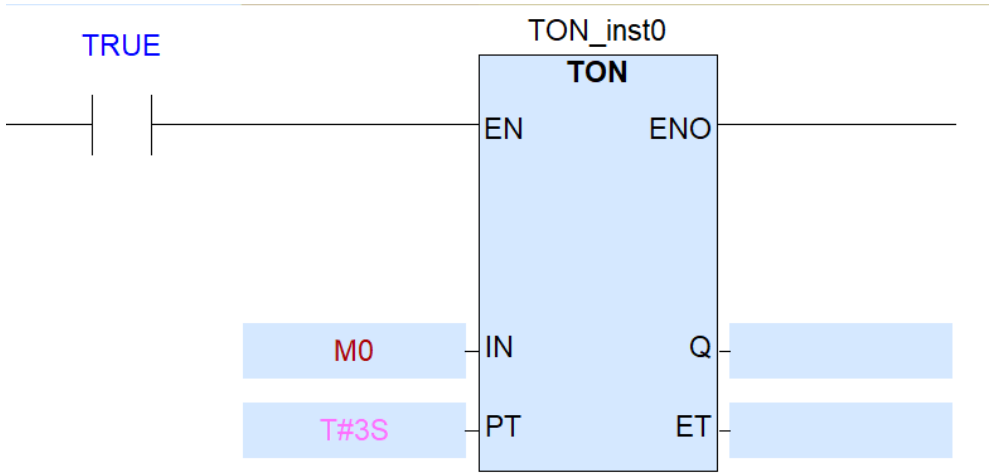
2）功能块使用方法

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），勾选方法见上述章节 3.5.1

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。

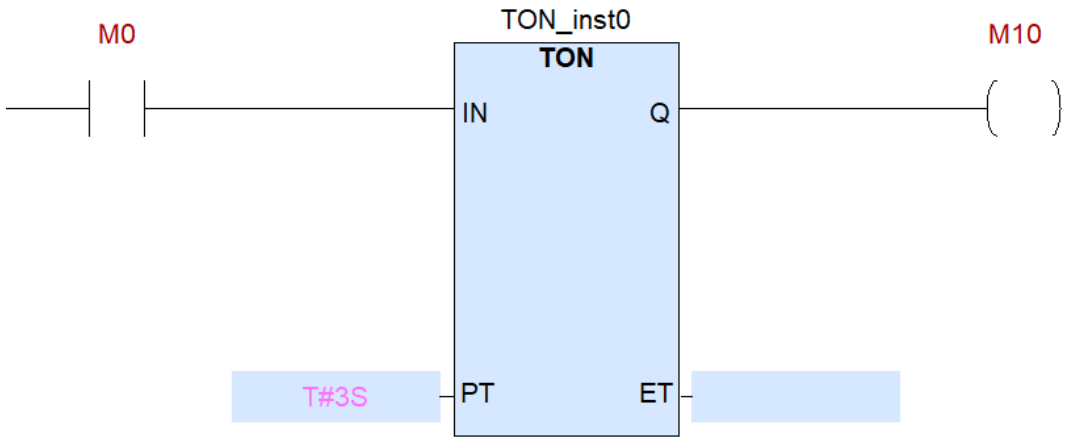
③为输入引脚 PT 分配 TIME 类型常量或变量，即设定定时器的预设时间。

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。

说明事项

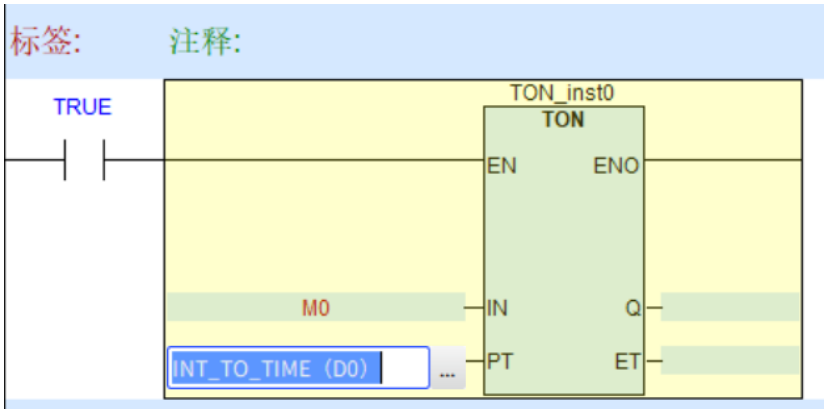
①功能块的输出引脚可不分配，使用 TON.ET、TON.Q 也可访问定时器的累计时间及完成状态；

②TIME 类型为 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”。

一般设定定时器的预设时间需要定义一个 TIME 类型变量,若用户想通过寄存器来设定，则需要使用“INT_TO_TIME”来转化寄存器类型，如下所示



此时 D0 的最小单位变为 1ms，假设 D0=1000,那么定时器的预设时间为 1000ms

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

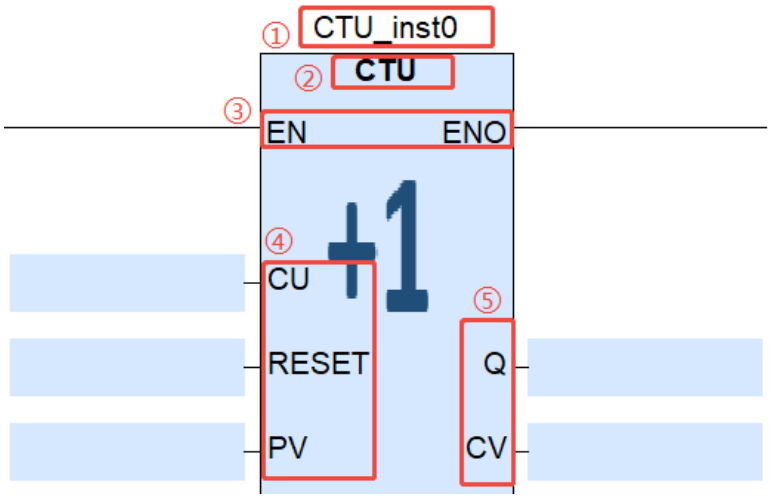
③定时功能块的预设时间与传统指令存在差别,功能块 TON 的预设时间由 TIME 类型的输入引脚 PT 设定（不需要时基），而传统指令以寄存器值*定时器时基来设定预设时间

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。

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。

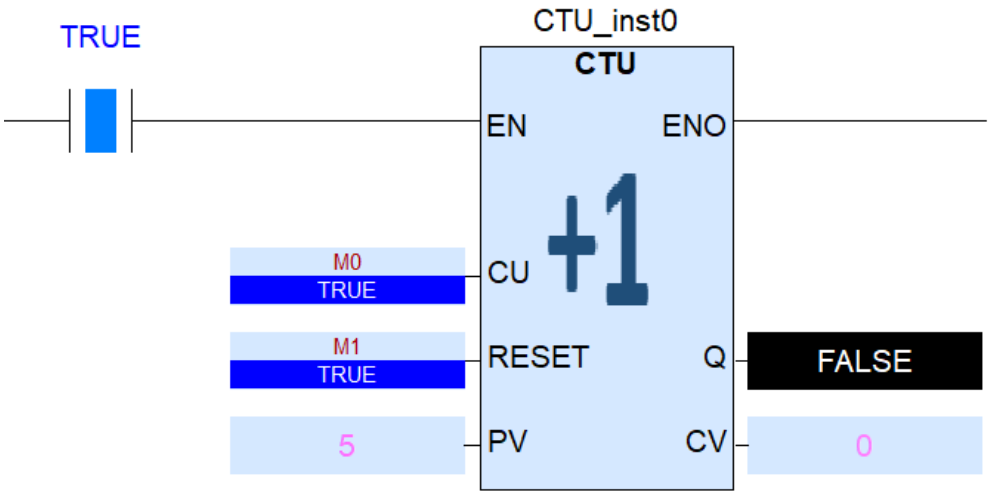
2）使用方法

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。

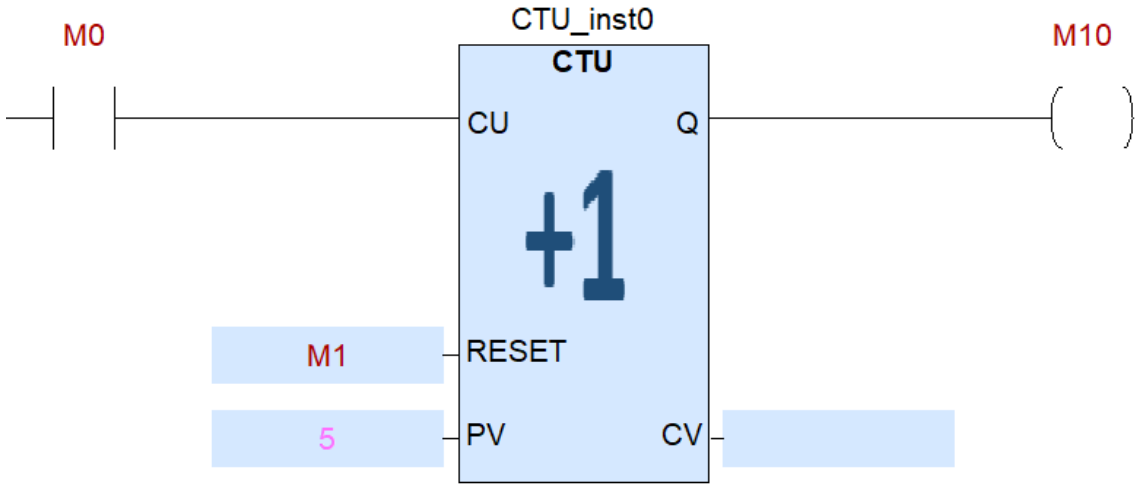
③为输入引脚 PV 分配 WORD 类型常量或变量，即设定计数器的预设计数值。

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。

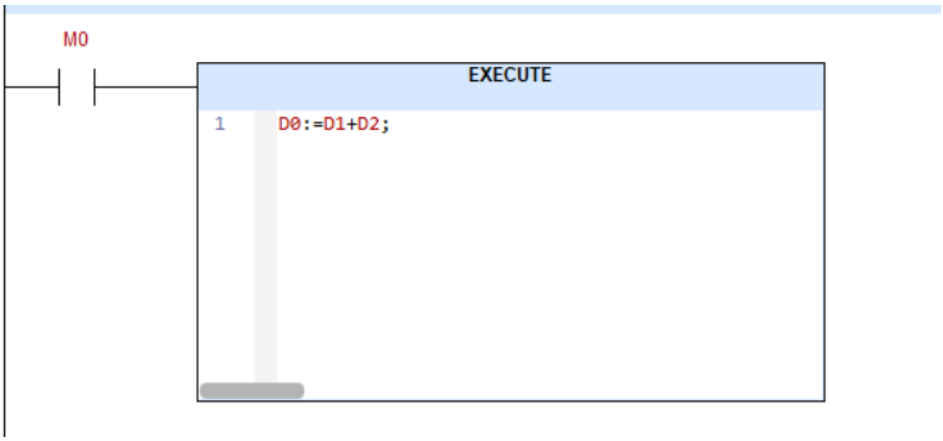
若用户习惯于传统指令的执行方式，可隐藏 EN/ENO 引脚，只需控制 CU 引脚的通断来触发计数

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。

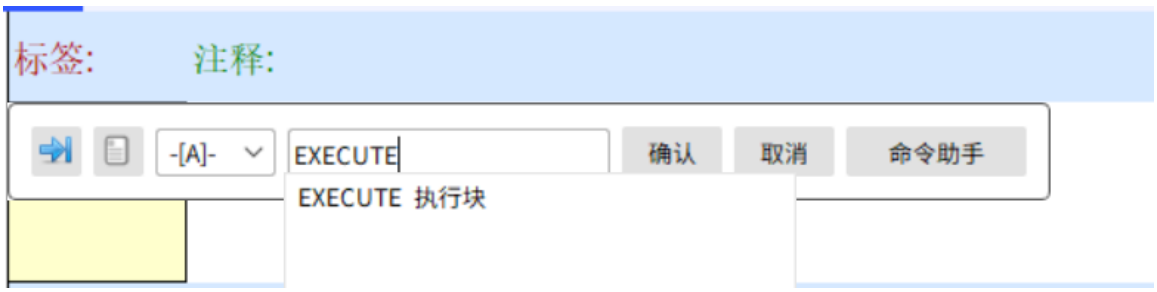
3.5.3 执行块

执行块是为了让用户可以在梯形图编程界面插入 ST 语言而设计的,主程序、子程序、中断程序以及 FB/FC 功能块均支持插入执行块。

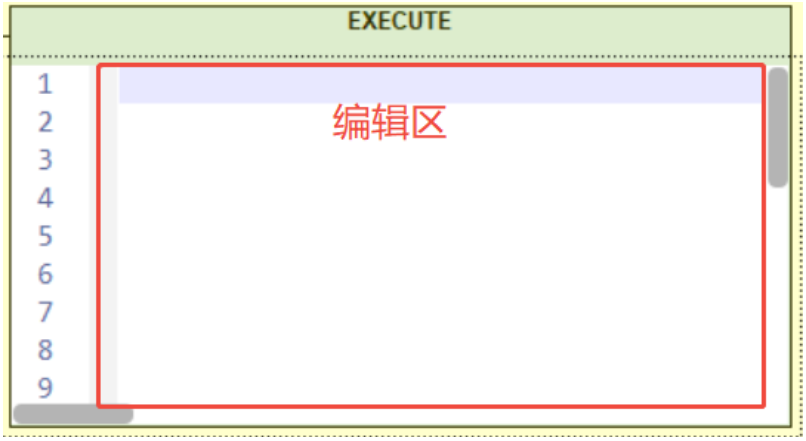


插入执行块

键盘命令输入“EXECUTE”



执行块编辑



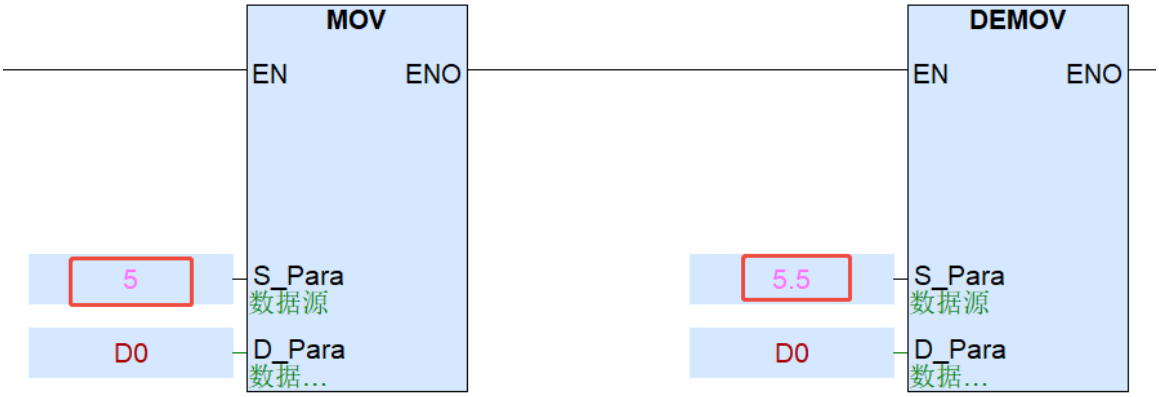
用户可在编辑区内编写 ST 语言，编辑方法与程序内的编辑方法一致

3.6 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#。

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。

✓ Ladder diagram(LD) 梯形图

✓ Structured text (ST) 结构化文本

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

1）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

2）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。

3）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

4）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。

5）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。

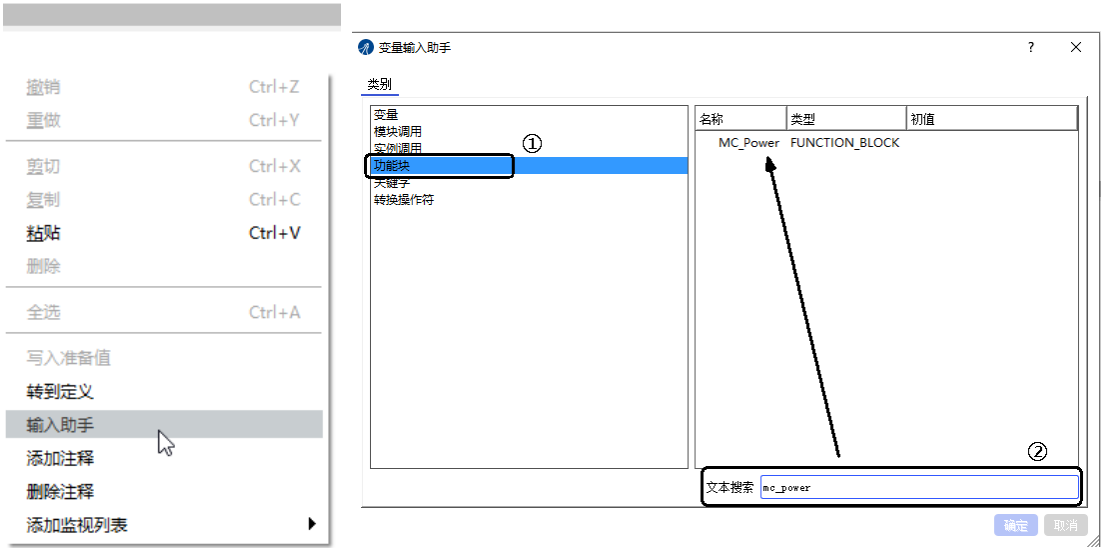
实例针对功能块而言，每个功能块实例就是一个独立的、可完成特定逻辑功能的活动对象。不同的程序、不同的任务都可以定义和调用功能块的应用实例，每个调用实例都占用独立的内存，保留独立的逻辑状态。

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。

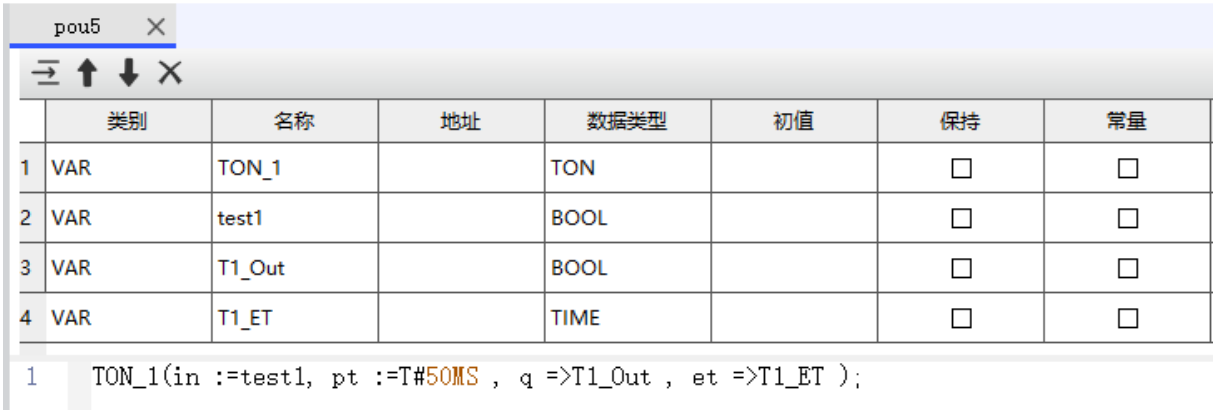


图：ST 调用功能块

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）

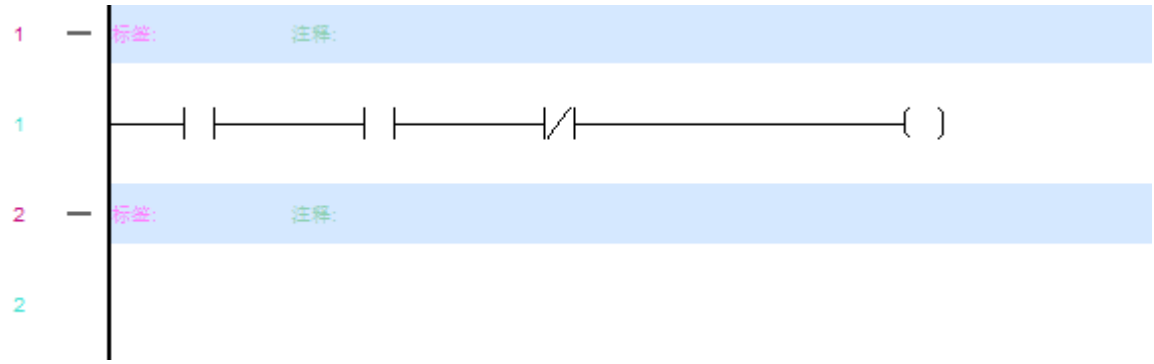
梯形图语言是 PLC 程序设计中最常用的语言。由于与电气设计人员所熟悉的传统继电器电路图类似，因此梯形图语言得到了广泛的应用。梯形图包含了一系列的网络（也称节），左右两边各有一个垂直的电流线（一般称为母线）限制其范围，在中间是由触点、线圈、运算块（函数、功能块）、跳转、标签和连接线组成的电路图。每一个网络的左边有一系列触点，这些触点根据布尔变量值的 TRUE 和 FALSE 来传递从左到右的“ON”和“OFF”的状态。每一个触点是一个布尔变量，如变量值为 TRUE，通过连接线从左到右传递“ON”状态。否则传递"OFF"的状态。在网络最右边的线圈，根据左边的状态获得“ON”或“OFF”的状态，并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

5 扩展模块

5.1 扩展模块类型

1）SC2-C 右扩展模块

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。

模块类型	型号	描述
数字量模块	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
	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
	SC-0016-R	16 路数字量输出，继电器输出，弹簧式接插件
	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
	SC-A0004-IV	4 路模拟量输出，支持电流/电压输入，弹簧式接插件

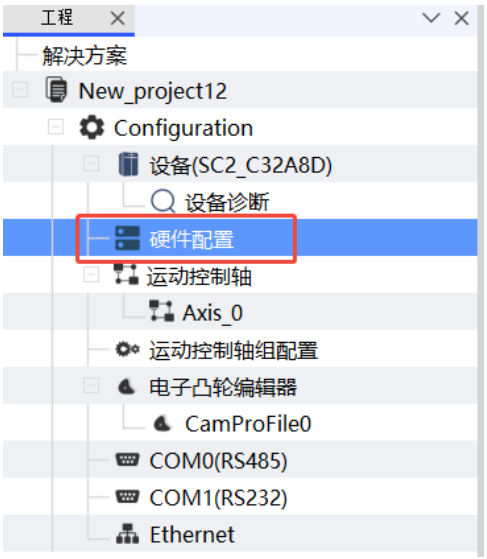
2）SC3U-C 右扩展模块

SC3U 系列 PLC 本体可扩展 16 路模块，可选择扩展数字量输入/输出、模拟量输入/输出以及热电偶/热电阻温度采集

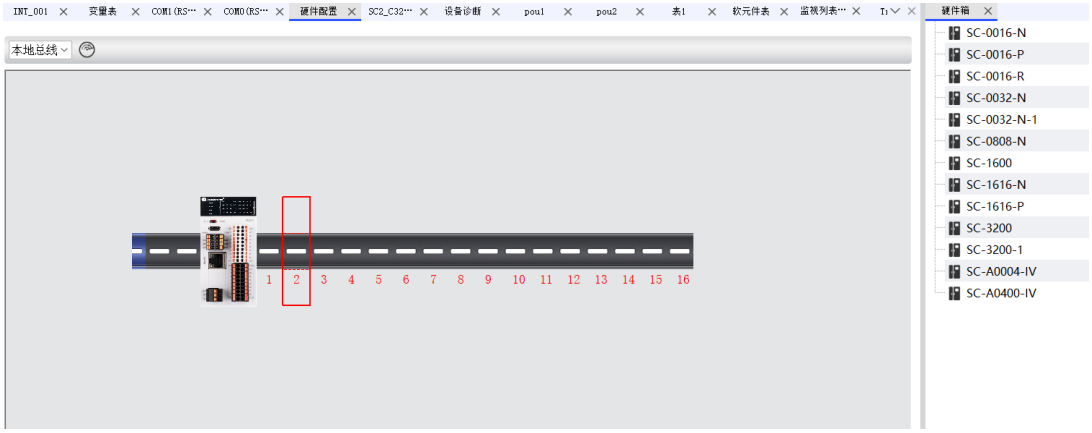
模块类型	型号	描述
数字量模块	SCU-0808-N	8 点输入，双极性；8 点输出，晶体管，漏型
	SCU-0808-R	8 点输入，双极性；8 点输出，继电器
	SCU-0808-P	8 点输入，双极性；8 点输出，晶体管，源型
	SCU-1600	16 点输入，晶体管，双极性
	SCU-0016-N	16 点输出，晶体管，漏型
	SCU-0016-R	16 点输出，继电器
	SCU-0016-P	16 点输出，晶体管，源型
	SCU-1616-N	16 点输入，双极性；16 点输出，晶体管，漏型
	SCU-1616-R	16 点输入，双极性；16 点输出，继电器
	SCU-3200	32 点输入，晶体管，双极性
	SCU-0032-N	32 点输出，晶体管，漏型
模拟量模块	SCU-A0400-IV	4 通道模拟量输入，16bit，电压/电流型
	SCU-A0004-IV	4 通道模拟量输出，16bit，电压/电流型
温度模块	SCU-T0400-TC	4 通道热电偶温度采集模块，精度 0.1 度
	SCU-T0400-TR	4 通道热电阻温度采集模块，精度 0.1 度

5.2 扩展模块组态

扩展模块组态在设备树中的位置如下：



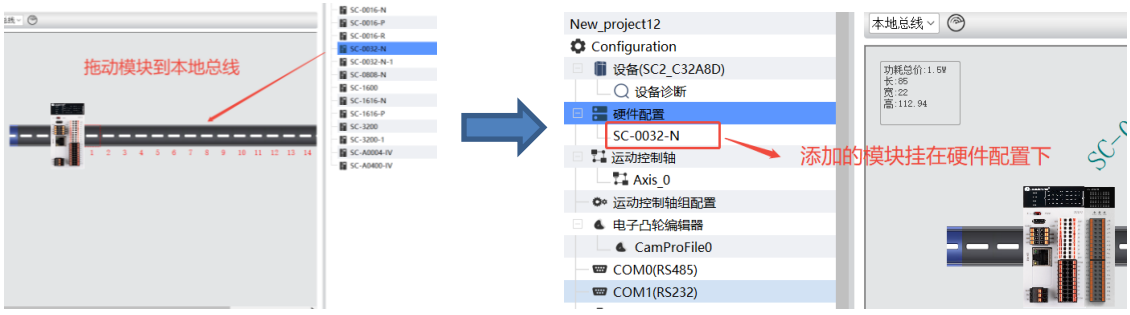
双击“硬件配置”可进入组态配置界面，如下图所示：



添加扩展模块组态

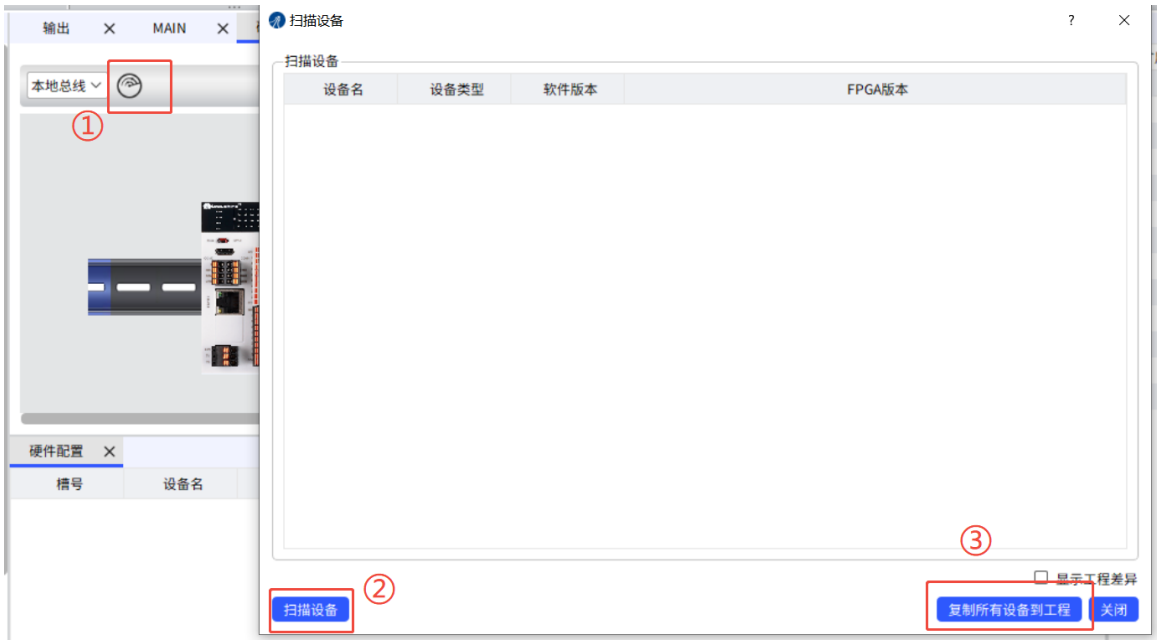
方式一：手动添加

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：



方式二：扫描添加

点击扫描图标---扫描设备---复制扫描到的设备到工程



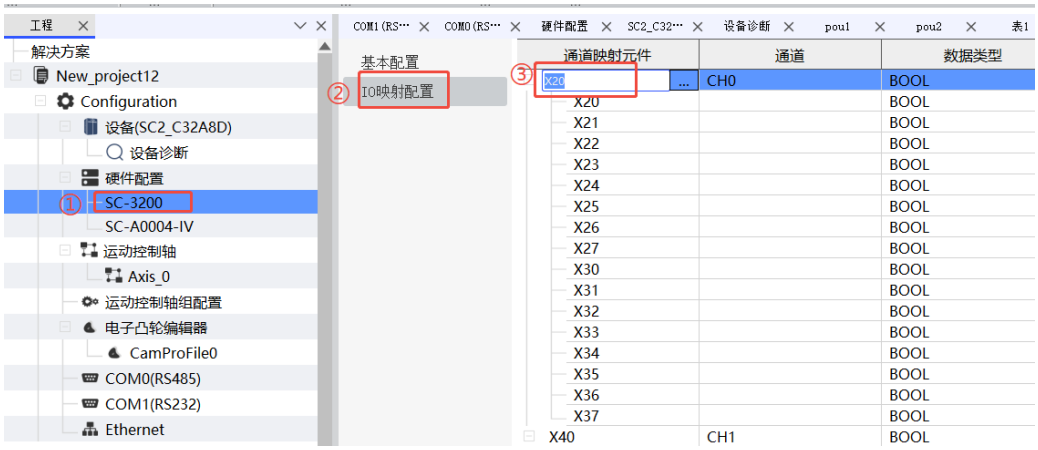
5.3 扩展模块的配置

IO 映射

①左键双击设备树下的扩展模块进入设置页面。

②左键单击“IO 映射配置”，进入 IO 映射配置界面。

③输入映射的软元件地址（数字量对应 X 或 Y，模拟量对应 D）。



参数设置

说明：数字量模块无需参数设置

1）模拟量输出模块参数设置

①左键双击设备树下的模拟量输出模块进入设置页面。

②左键单击“参数配置”，进入模块参数配置界面。

③设置通道使能，若勾选不使能则对应通道不输出。

④选择所需要的量程。

⑤通道滤波参数设置，根据实际需要选择滤波参数。

2）模拟量输入模块参数设置

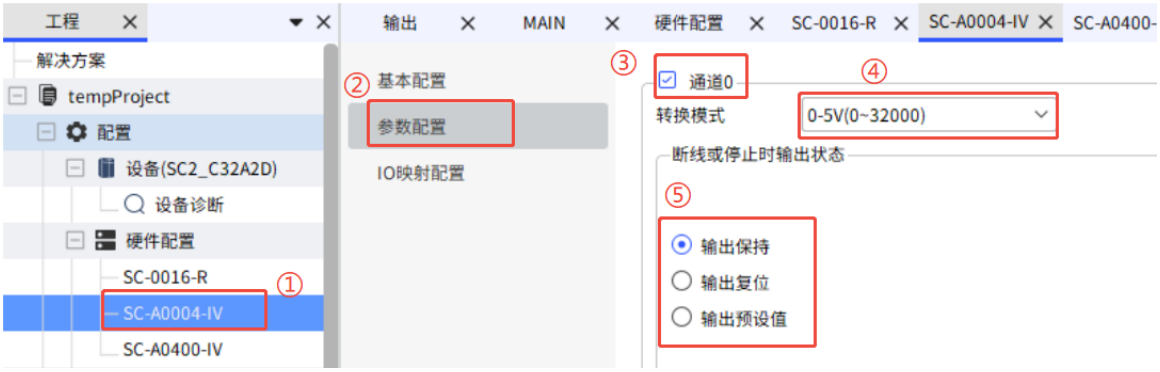
①左键双击设备树下的模拟量输入模块进入设置页面。

②左键单击“参数配置”，进入模块参数配置界面。

③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）。

④选择所需要的量程。

⑤根据实际需要设置断线后输出状态。



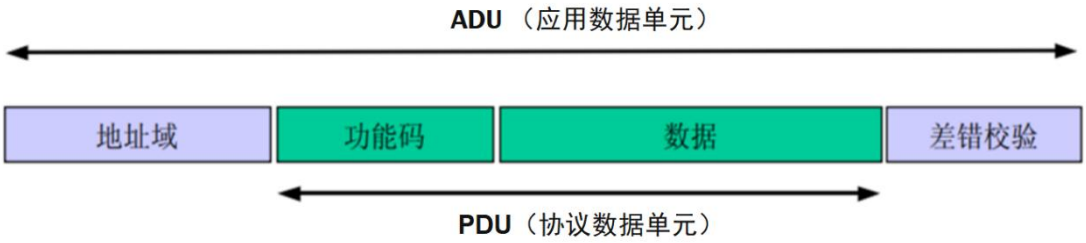
6 串口通讯

6.1 串口 Modbus 通信配置

6.1.1 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。

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

通信功能码

LeadStudio 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	位操作	单个或多个
02H	读离散输入状态线圈	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	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	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，可以读取软元件 M/Y/S/B。

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，读离散输入状态线圈，可以读取软元件 X。

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， 可以读取软元件 D/R。

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。

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，可以写软元件 M/Y/B/S。

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，可以写软元件 D/R。

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，可以写连续的多个软元件 M/Y/B/S。

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，可以写连续的多个软元件 D/R。

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

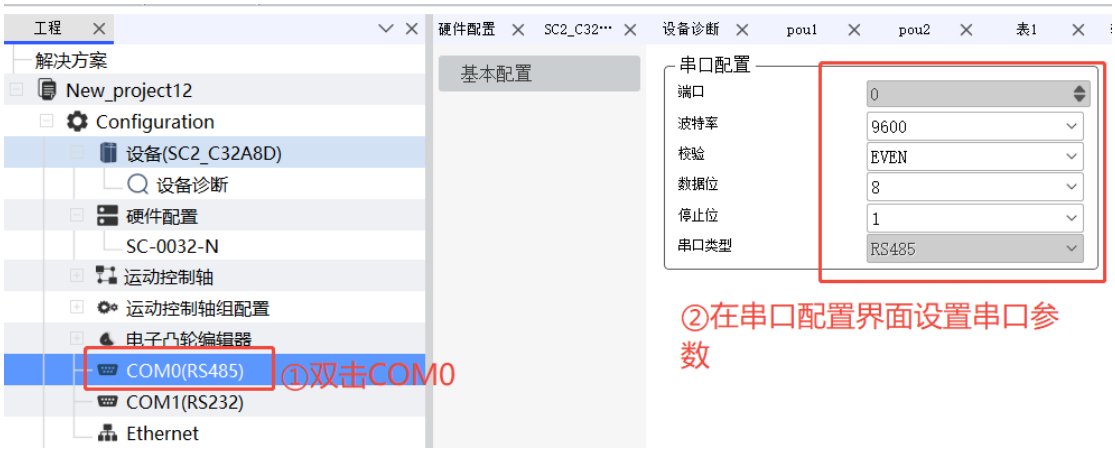
序号	描述	位/字操作	操作数量
1	从机地址	1 个字节	取值 1-247
2	0x10	1 个字节	写多个寄存器
3	寄存器起始地址	2 个字节	高位在前，低位在后，见寄存器编址
4	寄存器数量	2 个字节	高位在前，低位在后，N
5	CRC 校验	2 个字节	高位在前，低位在后

6.1.2 Modbus 主站通讯配置

以配置 485 串口为例，232 串口配置方法一致，区别在于 232 串口只可在主站下添加一个通讯从站

1）配置串口参数

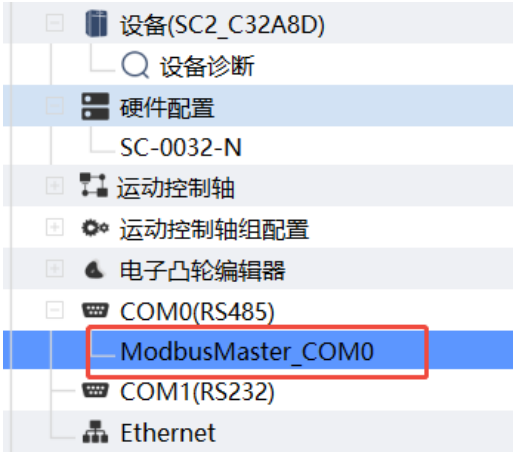
左键双击“COM0”，在打开的配置界面设置串口参数，如下图所示：



2）插入 Modbus 主站

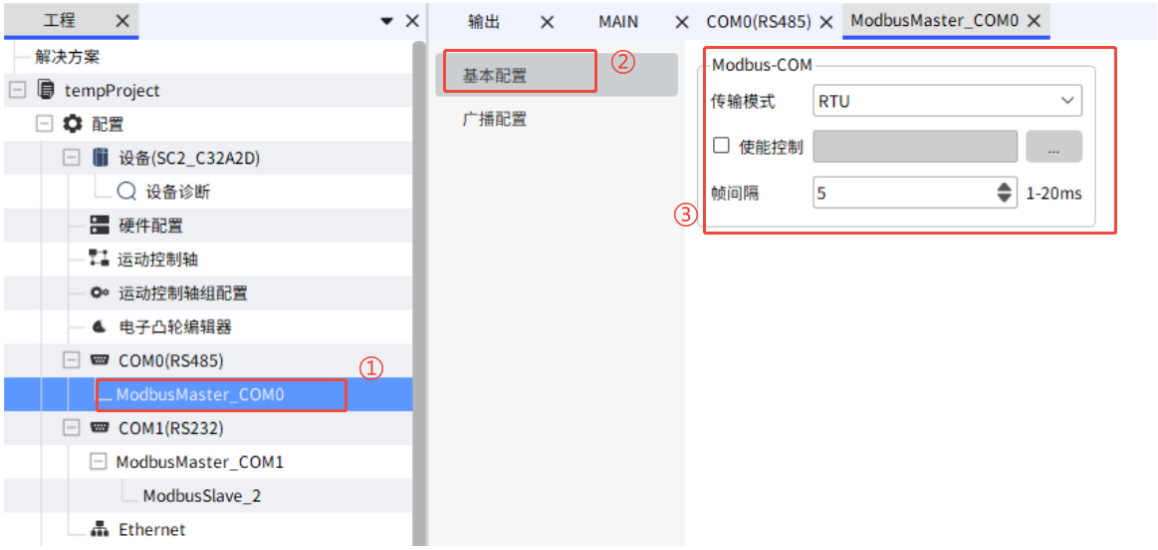
右键“COM0”，插入设备，选择 ModbusMaster_COM0，如下图所示：

添加成功后，MODBUS 主站会显示在 COM0 下，如下图所示：



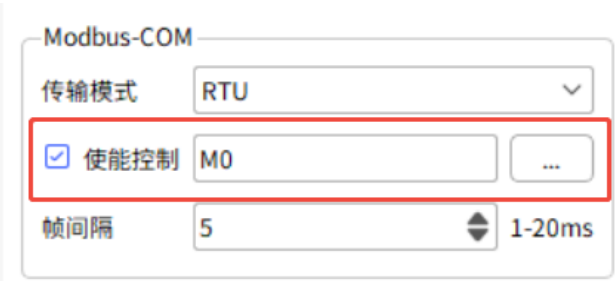
3）配置主站参数

左键双击上述步骤的 Modbus 主站，打开基本配置界面，设置主站传输协议、使能控制以及帧间隔等通讯参数，如下图所示：



传输模式：可选择 RTU/ASCI 通讯协议，根据通讯从站支持的协议来设定

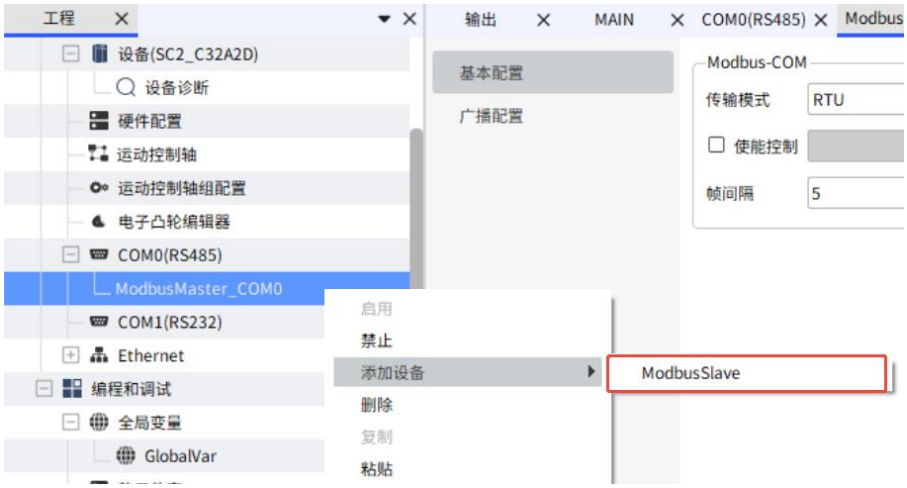
使能控制：可设置主站通讯的使能控制，不勾选时默认主站一直使能，勾选时根据选择的线圈来控制主站的通讯使能；如下图所示，只有当 M0 导通时主站才使能通讯；



帧间隔：主站通讯时发送数据帧的间隔时间，一般以默认 5ms 的时间来设定；若从站处理通讯时间较慢，可适当延长此时间来确保通讯正常。

4）在 Modbus 主站下添加通讯的从站

右键“ModbusMaster_COM0”，添加设备“ModbusSlave”，如下图所示：



添加成功后，MODBUS 从站会显示在“ModbusMaster_COM0”下，如下图所示：

5）配置从站参数

左键双击上述步骤的 Modbus 从站，打开基本配置界面，设置通讯从站的站号、超时响应以及使能控制等通讯参数，如下图所示：

站号：PLC 作为 Modbus 主站，与从站设备通讯的从站站号

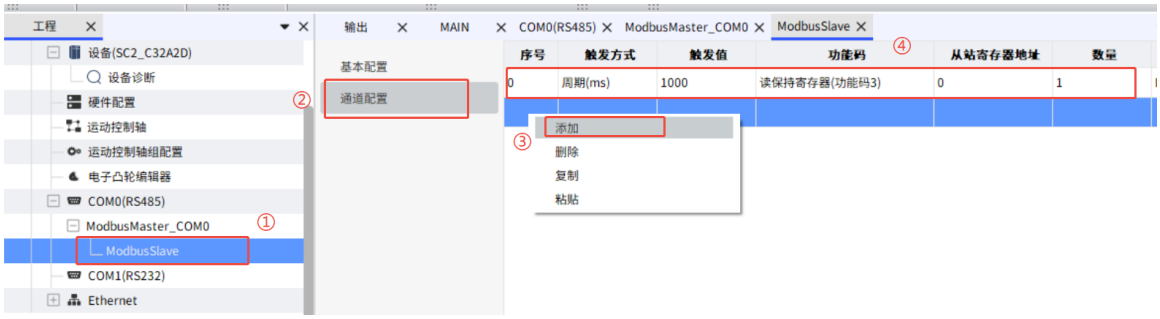
响应超时：PLC 发送通讯数据帧后，在响应超时时间内没有收到从站应答，判断通讯失败

使能控制：没有勾选时默认一直使能主站与该从站的通讯，勾选后根据选择的线圈

状态来控制从站通讯的使能

6）配置从站通讯通道

左键双击“ModbusSlave”，选择通道配置界面，右键添加通讯通道，配置通讯通道参数；如下图所示



通道配置相关参数如下表：

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	周期：根据触发值设定的时间来周期执行，单位 ms
		上升沿：根据触发值设定的线圈上升沿来触发执行
3	触发值	触发方式为周期时：设定触发的时间 触发方式为上升沿时：设定触发的线圈
4	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。
5	从站寄存	读\写从站的寄存器开始地址。

	器地址	
6	数量	读\写寄存器的个数
7	映射地址	读\写从站寄存器映射到本机的地址

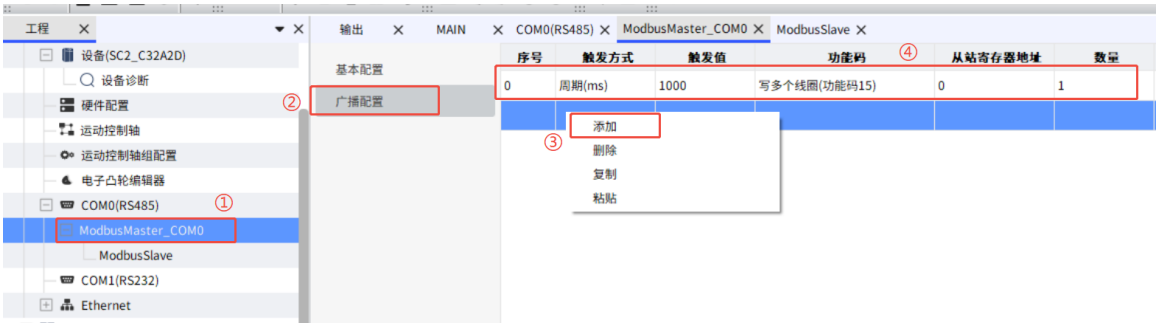
功能码详细说明如下表

功能码	访问类型	对应寄存器个数
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

7）配置主站广播

此步骤不是必选项，若不需要对所有通讯从站进行广播通讯，可忽略此步骤

左键双击“ModbusMaster_COM0”，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：



8）通讯故障排查

Modbus 主站与从站设备发生通讯错误时，可通过对应从站设备的名称访问错误代码。访问方法：DevicePOU.ModbusSlave.ErrorCode，其中 DevicePOU 为固定前缀，

ModbusSlave 为从站设备名称，ErrorCode 为错误码，如下图所示：

Modbus 常见错误代码如下表：

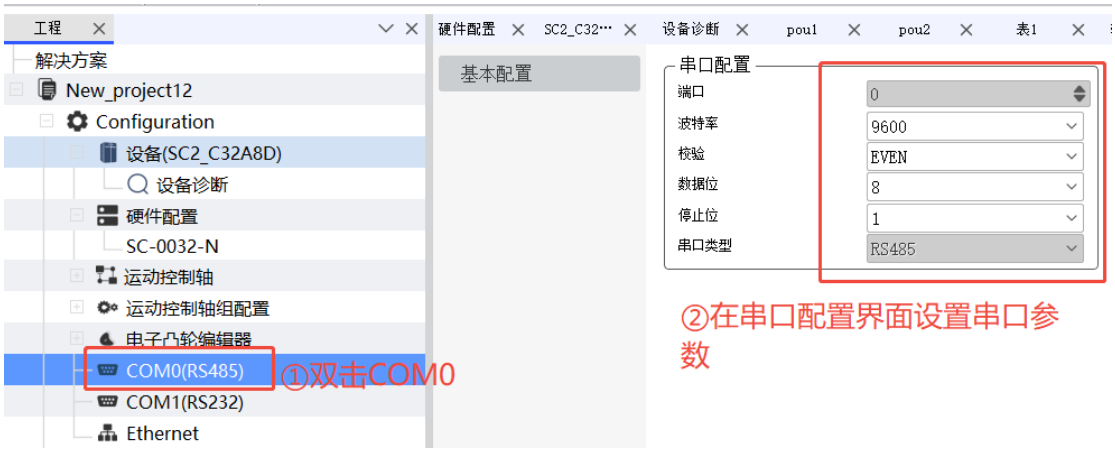
故障码	说明	故障码	说明
8012	CRC 错误	8105	数据地址超范围
8013	服务器（或从站）响应数据异常	8106	数据大小超范围
8014	服务器（或从站）异常响应	8107	缓冲区变量地址超范围
8015	无效功能码	8109	网络端口号超范围
8016	读/写线圈、寄存器数据过多	8110	未配置主站设备
8017	服务器（或从站）响应的站号与请求不一致	8111	内部错误
8101	从站号超范围	8112	连接超时
8102	重发次数超范围	8113	连接失败
8103	超时参数超范围	8114	Modbus 未配置正确
8104	功能码超范围	8115	Socket 连接不存在

6.1.3 Modbus 从站通讯配置

以配置 485 串口为例，232 串口配置方法一致

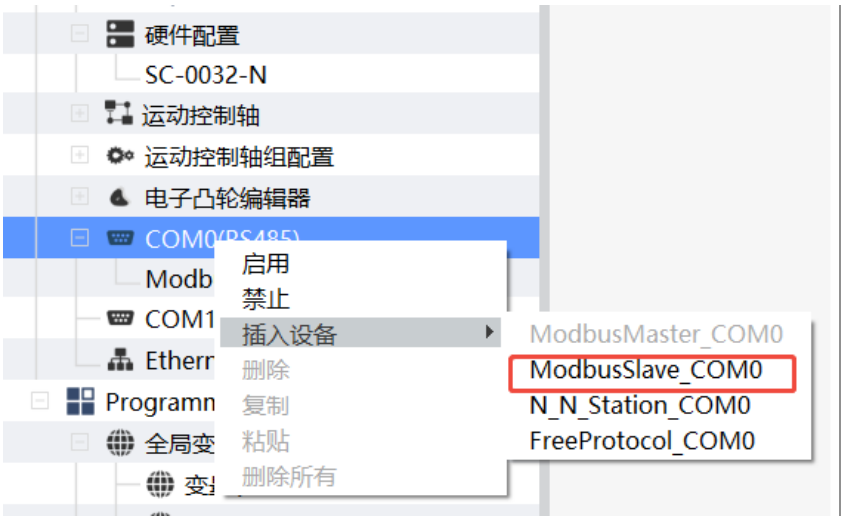
1）配置串口参数

左键双击“COM0”，在打开的配置界面设置串口参数；如下图所示：

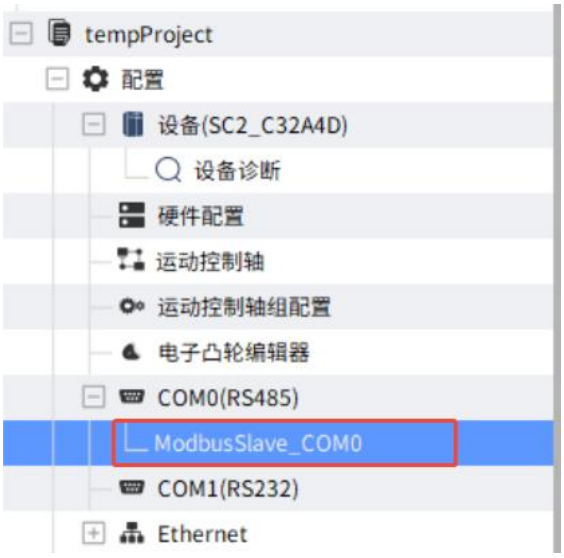


2）插入 MODBUS 从站

右键“COM0”，插入设备，选择 ModbusSlave_COM0，如下图所示：

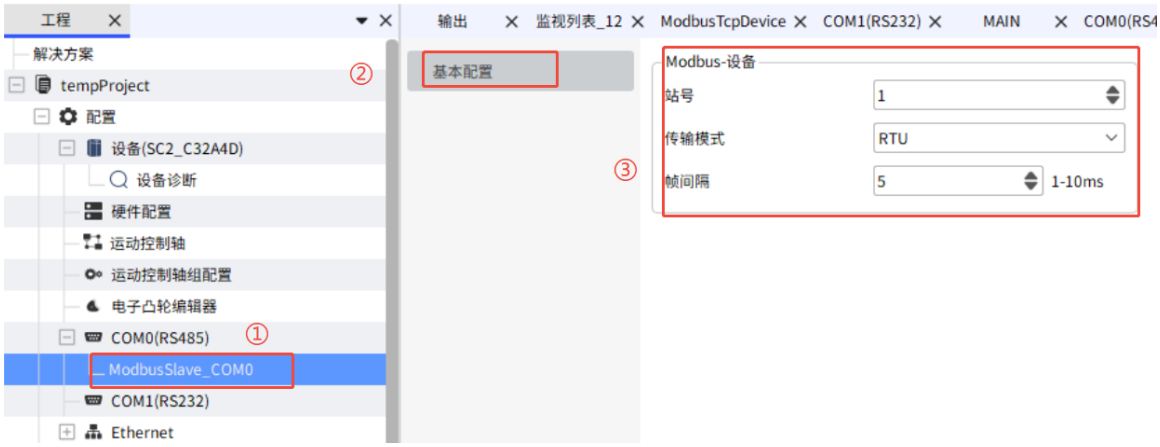


添加成功后，MODBUS 从站会显示在 COM0 下，如下图所示：



3）配置 Modbus 从站

左键双击上述步骤的 Modbus 从站，打开基本配置界面，设置从站站号、传输模式以及帧间隔等通讯参数，如下图所示：



站号：设定本机作为 Modbus 从站设备的通讯站号

传输模式：可选择 RTU/ASCI 通讯协议

帧间隔：从站响应主站发送数据帧的间隔时间，一般情况下按默认设置即可

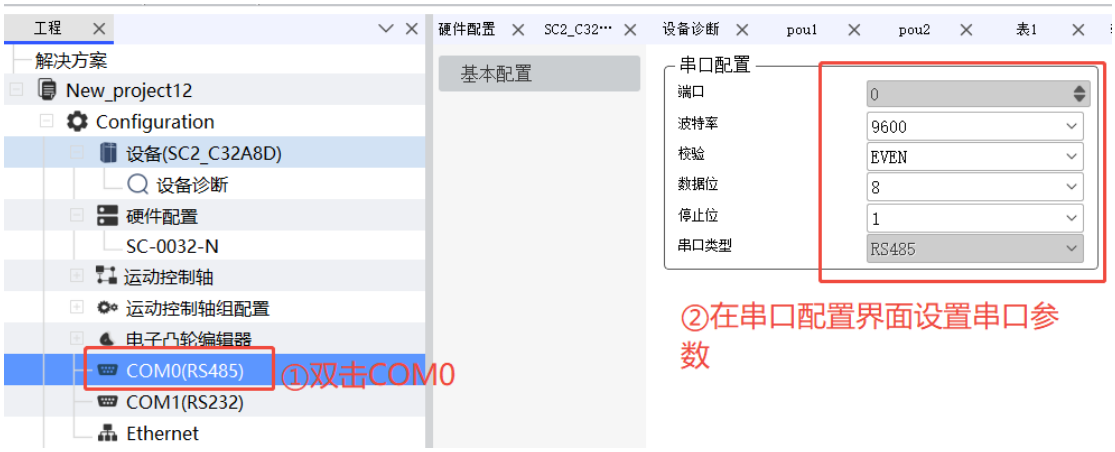
6.2 串口自由协议通讯

6.2.1 配置自由协议串口

以配置 485 串口为例，232 串口的配置方法一致

1）配置串口参数

左键双击“COM0”，在打开的配置界面设置串口参数；如下图所示：



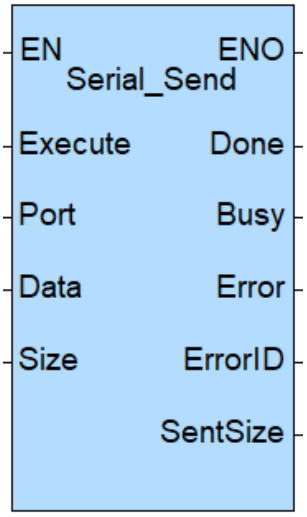
2) 插入自由串口

右键“COM0”，插入设备，选择“FreeProtocol_COM0”

6.2.2 功能块实现数据收发

说明：需要按上述步骤配置自由协议串口后，才可使用功能块实现数据收发

1）数据发送功能块



- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Execute	启动	BOOL	TRUE-FALSE	FALSE	上升沿触发指令
Port	端口	UINT	-	0	PLC 上的硬件端口，RS485 为 0,RS232 为 1

Data	发送数据指针	POINTER TO BYTE	-	-	指针，指向发送的数据
Size	设置发送数据的字节数	UINT	-	0	设置发送数据大小，单位：字节

输出变量

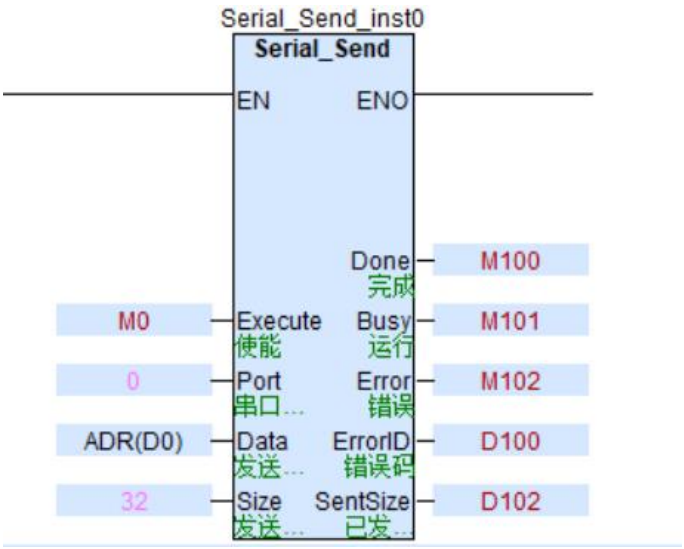
输出变量	名称	数据类型	有效范围	初始值	描述
Done	完成标志	BOOL	TRUE-FALSE	FALSE	如果指令完成，则为 TRUE。
Busy	忙标志	BOOL	TRUE-FALSE	FALSE	如果指令正在执行，则为 TRUE。
Error	错误标志	BOOL	TRUE-FALSE	FALSE	发生异常时，输出 TRUE。
ErrorID	错误代码	SERIAL_ERRID	-	0	故障码，详情查看 SERIAL_ERRID
SentSize	实际发送数据字节数	UINT	-	0	实际发送数据大小，单位：字节

- **功能说明**

①实现自由协议发送，指令触发后，通过指定端口发送指定长度的数据。

②输入参数Data为指针类型，指向本机端口发送数据的存放地址；需要使用ADR指令对寄存器进行取址

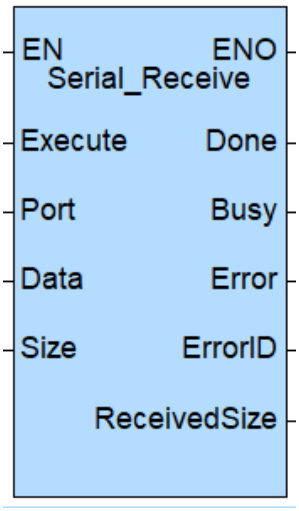
- **指令示例**



当M0上升沿触发后，将发送本机的D0-D16共32个字节的数据到串口COM0；

指令执行中M101置ON，执行完成M100置ON，发生错误M102置ON;

2）数据接收功能块



- **相关变量**

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Execute	启动	BOOL	TRUE-FALSE	FALSE	上升沿触发指令
Port	端口	UINT	-	0	PLC 上的硬件端口，RS485 为

					0,RS232 为 1
Data	读取数据指针	POINTER TO BYTE	-	-	指针，指向读取到的数据
Size	设置读取数据的字节数	UINT	-	0	设置读取数据大小，单位：字节

输出变量

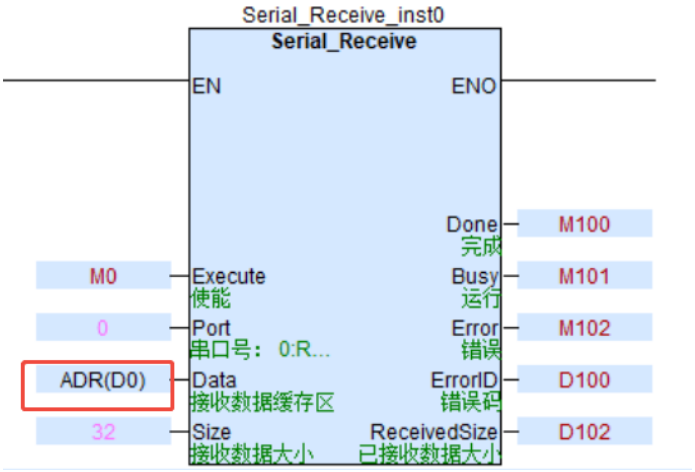
输出变量	名称	数据类型	有效范围	初始值	描述
Done	完成标志	BOOL	TRUE-FALSE	FALSE	如果指令完成，则为 TRUE。
Busy	忙标志	BOOL	TRUE-FALSE	FALSE	如果指令正在执行，则为 TRUE。
Error	错误标志	BOOL	TRUE-FALSE	FALSE	发生异常时，输出 TRUE。
ErrorID	错误代码	SERIAL_ERRID	-	0	故障码，详情查看 SERIAL_ERRID
ReceivedSize	实际接收数据的字节数	UINT	-	0	实际接收数据大小，单位：字节

- **功能说明**

①实现自由协议接收，指令触发后，通过指定端口接收指定长度的数据。

②输入参数Data为指针类型，指向本机端口接收数据的存放地址

- **指令示例**



当M0上升沿触发后，将从串口COM0接收32个字节的数据到本机地址的D0-D16；指令执行中M101置ON，执行完成M100置ON，发生错误M102置ON；

7 以太网通讯

7.1 ModbusTCP 通讯

7.1.1 ModbusTCP 通讯协议

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。

Modbus TCP 通讯帧格式如下

① 功能码 0x01，读线圈寄存器，可以读取软元件 M/Y/B/S。

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，可以读取软元件 X。

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，可以读取软元件 D/R。

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。

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，可以写软元件 M/Y/B/S

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，可以写软元件 D/R。

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），可以写连续的多个软元件 M/Y/B/S

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	每 8 个线圈合为一个字节，最后一个

		字节	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），可以写连续的多个软元件 D/R。

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

7.1.2 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) **添加 Modbus TCP 主站**

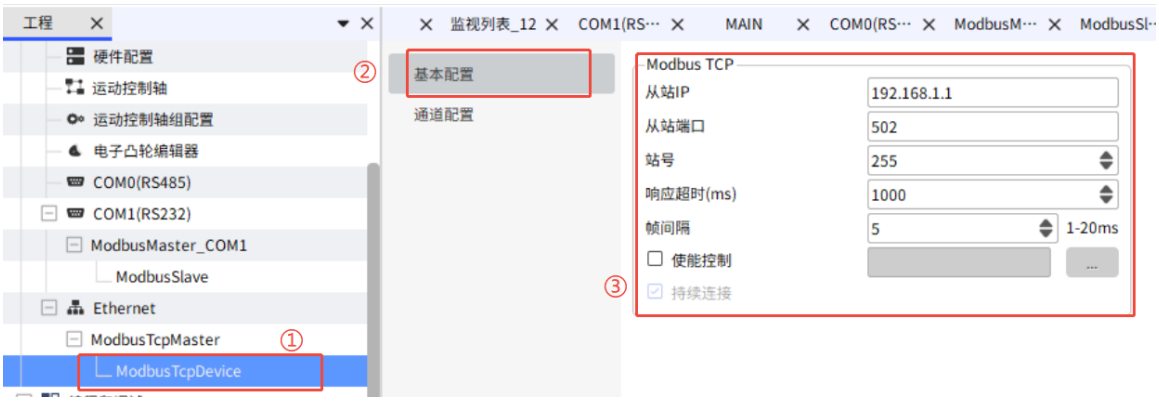
右键“Ethernet”，添加设备，选择 ModbusTCPMaster，如下图所示：

2) **在 ModbusTCP 主站下添加通讯的从站设备**

右键上述步骤添加的 MobdusTcp 主站，添加设备“ModbusTcpSlave”，如下图所示：

3) **配置从站设备参数**

双击“ModbusTcpDevice”，打开基本配置界面，配置从站 Modbus Tcp 参数，如下图所示：



从站 IP：对应远程通讯从站的 IP 地址

从站端口：默认使用 502 端口

站号：从站的通讯站号，在 ModbusTCP 中可忽略站号，按默认设置即可

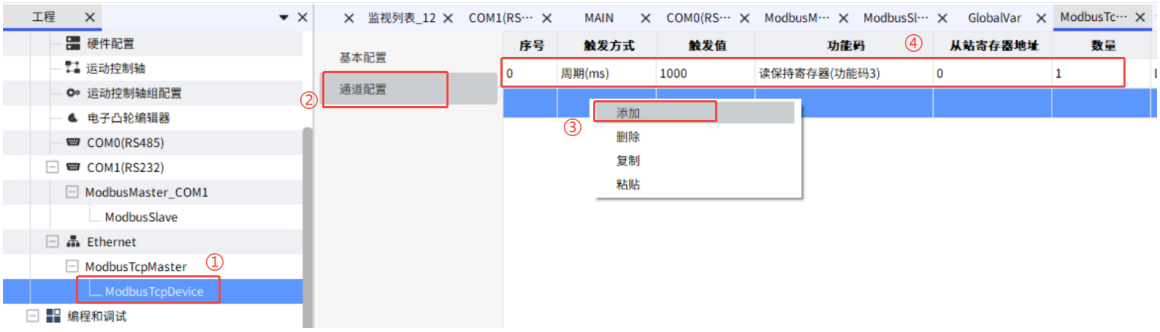
响应超时：PLC 发送通讯数据帧后，在响应超时时间内没有收到从站应答，判断通讯失败

帧间隔：主站通讯时发送数据帧的间隔时间，一般以默认 5ms 的时间来设定；若从站处理通讯时间较慢，可适当延长此时间来确保通讯正常。

使能控制：没有勾选时默认一直使能主站与该从站的通讯，勾选后根据选择的线圈状态来控制从站通讯的使能

4) **从站设备的通道配置**

双击“ModbusTcpDevice”，选择通道配置界面，右键添加通讯通道，配置通道的通讯参数，如下图所示：



通道参数的配置方法参考章节 6.1.2，与串口 Modbus 通道配置一致

5) **Modbus TCP 故障排查**

a) Modbus TCP 主站与从站设备通讯故障时，可通过对应从站设备的名称访问错误代码。

访问方法：DevicePOU.ModbusTcpDevice.ErrorCode，其中 DevicePOU 为固定前缀，ModbusTcpDevice 为从站设备名称，ErrorCode 为错误码；

Modbus 常见错误代码如下表：

故障码	说明	故障码	说明
8012	CRC 错误	8105	数据地址超范围
8013	服务器（或从站）响应数据异常	8106	数据大小超范围
8014	服务器（或从站）异常响应	8107	缓冲区变量地址超范围

8015	无效功能码	8109	网络端口号超范围
8016	读/写线圈、寄存器数据过多	8110	未配置主站设备
8017	服务器（或从站）响应的站号与请求不一致	8111	内部错误
8101	从站号超范围	8112	连接超时
8102	重发次数超范围	8113	连接失败
8103	超时参数超范围	8114	Modbus 未配置正确
8104	功能码超范围	8115	Socket 连接不存在

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

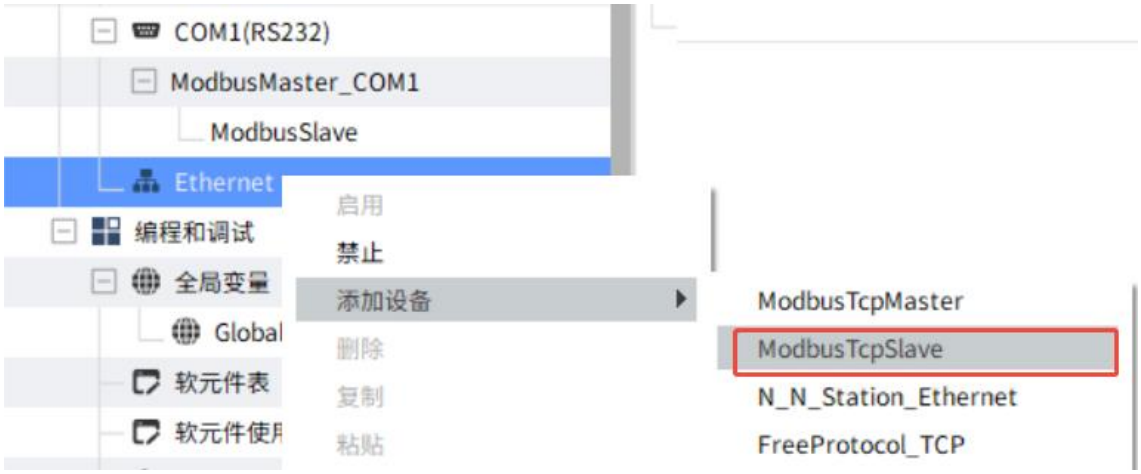
7.1.3 ModbusTCP 从站通讯

PLC 做 Modbus TCP 从站时，为 Modbus TCP 服务器（响应通讯的一方）。

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

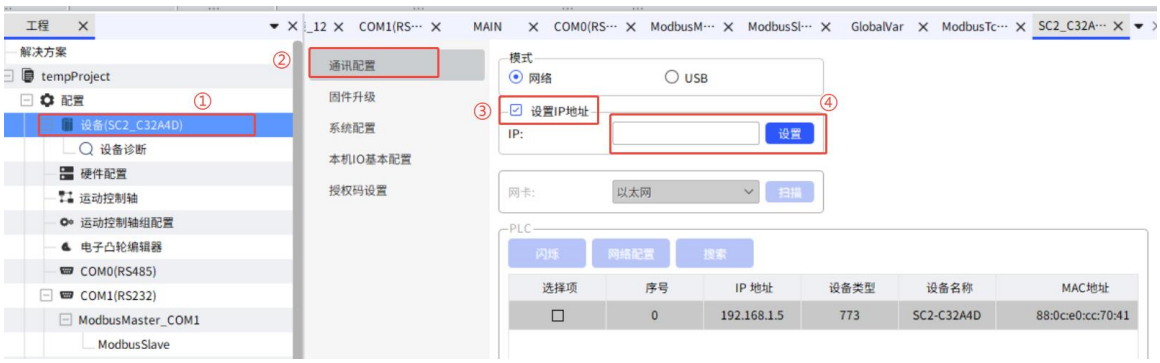
1) 添加 Modbus TCP 从站

右键“Ethernet”，添加设备，选择 ModbusTcpSlave，如下图所示：



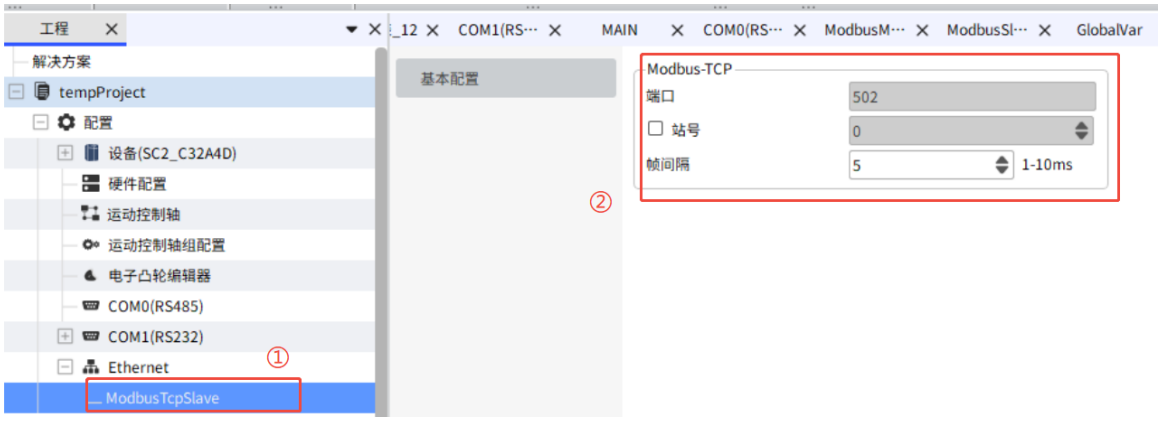
2) 设置 Modbus TCP 从站 IP，即本机的 IP 地址

左键双击“设备（名称）”，选择通讯配置页面，勾选设置 IP 地址，输入 IP 地址后点击设置



3) 设置 Modbus TCP 从站通讯参数

左键双击上述步骤添加的 Modbus TCP 从站，在基本配置界面中设置相关通讯参数



端口：默认 502 端口，无需设置

站号：ModbusTCP 通过 IP 地址识别从站，可忽略站号，无需设置

帧间隔：从站响应主站发送数据帧的间隔时间，一般情况下按默认设置即可

7.2 TCP/UDP 通讯

7.2.1 概述

1.TCP 简述

TCP（Transmission Control Protocol）是一种面向连接的、可靠的、基于字节流的传输层通信协议，应用程序在使用 TCP 协议收发数据之前，需要先建立 TCP 连接，在传送数据完毕后，释放已经建立的 TCP 连接；

TCP 提供可靠的传输服务，传送的数据无差错、不丢失、不重复、且按序到达。每一条 TCP 连接只能有两个端点，是点对点通信。

2.UDP 简述

UDP（User Datagram Protocol）是一个简单的面向消息的传输层协议，其特点是：简单、轻量化；但没有流控制，没有应答确认机制，不能解决丢包、重发、错序问题。

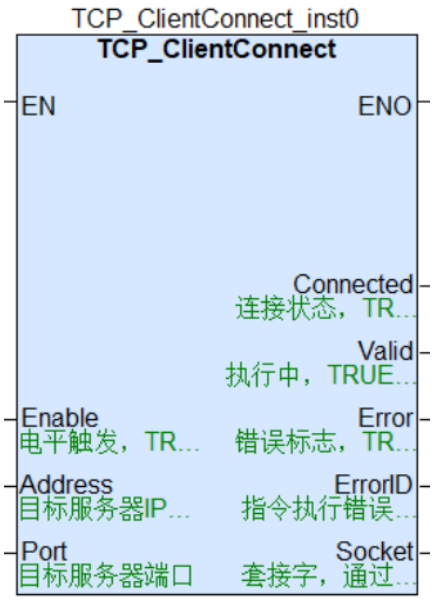
UDP 是面向无连接的通讯协议，UDP 数据包括目标端口号和源端口号信息，通讯不需要连接，不需要接收方确认，属于不可靠的传输，可能会出现丢包现象，实际应用中要求用户编程验证。

7.2.2 TCP_Client 通讯

1) 添加设备“FreeProtocol_TCP”

右键 Ethernet，添加设备，插入 FreeProtocol_TCP，如下图所示：

2）使用功能块“TCP_ClientConnect”创建 TCP 客户端通讯服务



指令说明

Enable 为 true 时使能 TCP 通讯客户端

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Enable	使能	BOOL	TRUE-FALSE	FALSE	电平触发指令。
Address	IP 地址	STRING[15]	-	''	目标服务器 IP 地址
Port	端口	UINT	-	0	目标服务器端口

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Connected	连接状态	BOOL	TRUE-FALSE	FALSE	连接状态，TRUE：已连接 TCP 服务器
Valid	执行中	BOOL	TRUE-FALSE	FALSE	执行中，TRUE：指令正在执行
Error	错误标志	BOOL	TRUE-FALSE	FALSE	错误标志，TRUE：指令执行错误
ErrorID	错误代码	ERRNO	-	0	指令执行错误时，输出错误码
Socket	套接字	INT	-	0	套接字，通过套接字提供的接口进行数据传输

3）使用功能块“TCP_Send / TCP_Receive”实现数据的发送和接收

①TCP_Send

指令说明

Execute 检测到上升沿时，发送本机指定地址的数据

输入变量

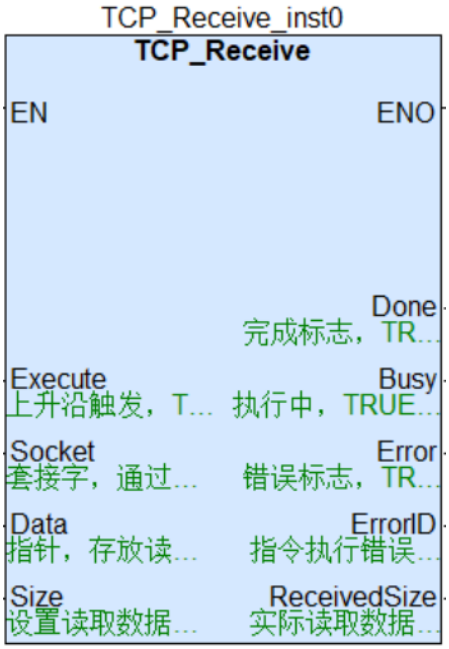
输入变量	名称	数据类型	有效范围	初始值	描述
Execute	发送数据	BOOL	TRUE-FALSE	FALSE	上升沿触发

					指令。
Socket	套接字	INT	-	0	套接字，通过套接字提供的接口进行数据传输
Data	发送数据指针	POINTER TO BYTE	-	0	指针，指向发送数据地址
Size	设置发送数据字节数	UINT	-	0	发送数据大小，单位：字节

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Done	完成信号	BOOL	TRUE-FALSE	FALSE	完成标志，TRUE：数据读取完成
Busy	执行中	BOOL	TRUE-FALSE	FALSE	执行中，TRUE：指令正在执行
Error	错误标志	BOOL	TRUE-FALSE	FALSE	错误标志，TRUE：指令执行错误
ErrorID	错误代码	ERRNO	-	0	指令执行错误时，输出错误码
SentSize	实际发送数据字节数	UINT	-	0	实际发送数据大小，单位：字节

②TCP_Receive



指令说明

Execute 检测到上升沿时，接收数据到本机指定的地址

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Execute	读取数据	BOOL	TRUE-FALSE	FALSE	上升沿触发指令。
Socket	套接字	INT	-	0	套接字，通过套接字提供的接口进行数据传输
Data	读取数据指针	POINTER TO BYTE	-	0	指针，指向读取到的数据
Size	设置读取数据字节数	UINT	-	0	设置读取数据大小，单位：字节

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Done	完成信号	BOOL	TRUE-FALSE	FALSE	完成标志，TRUE：数据读取完成

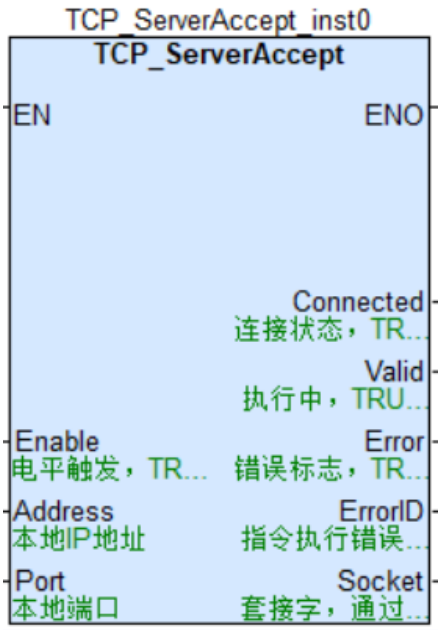
Busy	执行中	BOOL	TRUE-FALSE	FALSE	执行中，TRUE：指令正在执行
Error	错误标志	BOOL	TRUE-FALSE	FALSE	错误标志，TRUE：指令执行错误
ErrorID	错误代码	ERRNO	-	0	指令执行错误时，输出错误码
ReceivedSize	实际读取数据字节数	UINT	-	0	实际读取数据大小，单位：字节

7.2.3 TCP_Serve 通讯

1) 添加设备“FreeProtocol_TCP”

右键 Ethernet，添加设备，插入 FreeProtocol_TCP，如下图所示：

2）使用功能块“TCP_ServerAccept”创建 TCP 服务器端通讯服务



指令说明

Enable 为 true 时使能 TCP 通讯服务器端

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Enable	使能	BOOL	TRUE-FALSE	FALSE	电平触发指令。
Address	IP 地址	STRING[15]	-	''	本地 IP 地址
Port	端口	UINT	-	0	本地端口

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Connected	连接状态	BOOL	TRUE-FALSE	FALSE	连接状态，TRUE：TCP 客户端已连接
Valid	执行中	BOOL	TRUE-FALSE	FALSE	执行中，TRUE：指令正在执行
Error	错误标志	BOOL	TRUE-FALSE	FALSE	错误标志，TRUE：指令执行错误
ErrorID	错误代	ERRNO	-	0	指令执行错误时，输出

	码				错误码
Socket	套接字	INT	-	0	套接字，通过套接字提供的接口进行数据传输

3）使用功能块“TCP_Send / TCP_Receive”实现数据的发送和接收

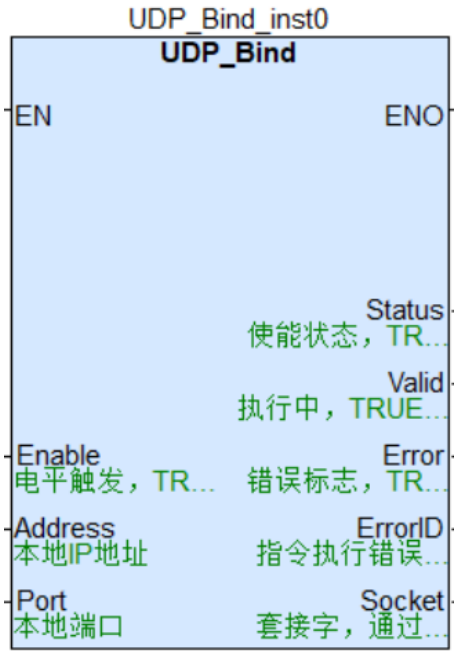
同 TCP_Client 通讯

7.2.4 UDP 通讯

1) 添加设备“FreeProtocol_TCP”

右键 Ethernet，添加设备，插入 FreeProtocol_TCP，如下图所示：

2）使用功能块“UDP_Bind”绑定 UDP 端口



指令说明

Enable 为 true 时使能 UDP 端口绑定

输入变量

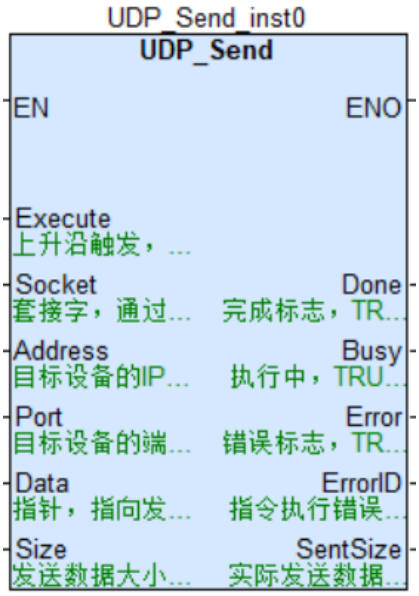
输入变量	名称	数据类型	有效范围	初始值	描述
Enable	使能	BOOL	TRUE-FALSE	FALSE	电平触发指令
Address	IP 地址	STRING[15]	-	''	本地 IP 地址
Port	端口	UINT	-	0	本地端口

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Status	状态	BOOL	TRUE-FALSE	FALSE	使能状态，TRUE：UDP 端口已绑定
Valid	执行中	BOOL	TRUE-FALSE	FALSE	执行中，TRUE：指令正在执行
Error	错误标志	BOOL	TRUE-FALSE	FALSE	错误标志，TRUE：指令执行错误
ErrorID	错误代码	ERRNO	-	0	指令执行错误时，输出错误码
Socket	套接字	INT	-	0	套接字，通过套接字提供的接口进行数据传输

3）使用功能块“UDP_Send / UDP_Receive”实现数据的发送和接收

①UDP_Send



指令说明

Execute 检测到上升沿时，发送本机指定地址的数据

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Execute	使能	BOOL	TRUE-FALSE	FALSE	上升沿触发指令。
Address	IP 地址	STRING[15]	-	''	本地 IP 地址
Port	端口	UINT	-	0	本地端口
Socket	套接字	INT		0	套接字，通过套接字提供的接口进行数据传输
Data	发送数据指针	POINTER TO BYTE	-	0	指针，指向发送数据地址
Size	设置发送数据字节数	UINT	-	0	发送数据大小，单位：字节

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述

Done	完成信号	BOOL	TRUE-FALSE	FALSE	完成标志，TRUE：数据读取完成
Busy	执行中	BOOL	TRUE-FALSE	FALSE	执行中，TRUE：指令正在执行
Error	错误标志	BOOL	TRUE-FALSE	FALSE	错误标志，TRUE：指令执行错误
ErrorID	错误代码	ERRNO	-	0	指令执行错误时，输出错误码
SentSize	实际发送数据字节数	UINT	-	0	实际发送数据大小，单位：字节

②UDP_Receive

指令说明

Execute 检测到上升沿时，接收数据到本机指定的地址

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Execute	读取数据	BOOL	TRUE-FALSE	FALSE	上升沿触发指令。
Socket	套接字	INT	-	0	套接字，通

					过套接字提供的接口进行数据传输
Data	读取数据指针	POINTER TO BYTE	-	0	指针，指向读取到的数据
Size	设置读取数据字节数	UINT	-	0	设置读取数据大小，单位：字节

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Done	完成信号	BOOL	TRUE-FALSE	FALSE	完成标志，TRUE：数据读取完成
Busy	执行中	BOOL	TRUE-FALSE	FALSE	执行中，TRUE：指令正在执行
Error	错误标志	BOOL	TRUE-FALSE	FALSE	错误标志，TRUE：指令执行错误
ErrorID	错误代码	ERRNO	-	0	指令执行错误时，输出错误码
Address	IP 地址	STRING[15]	-	''	接受数据来源 IP 地址
Port	端口	UINT	-	0	接受数据来源端口
ReceivedSize	实际接收数据的字节数	UINT	-	0	实际读取数据大小，单位：字节

8 多机互联通讯

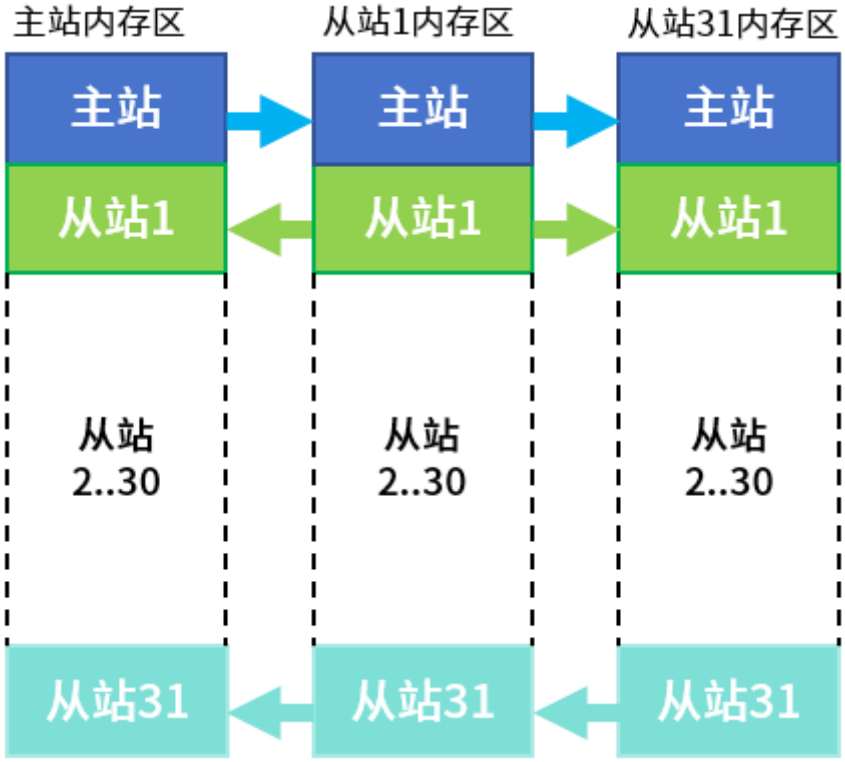
多机互联通讯可用于多台 PLC 之间的数据交互，传统的通讯方式配置相对繁琐，需要分开管理不同的 PLC 程序，LeadStudio 方案提供以下优势：

- 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

8.1 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			
		M	D
模式 0 0 个 M 元件 4 个 D 元件	主站	无	D0 ~ D3
	从站 1	无	D50 ~ D53
	从站 2	无	D100 ~ D103
	从站 3	无	D150 ~ D153
	从站 4	无	D200 ~ D203

	从站 5	无	D250 ~ D253
	从站 6	无	D300 ~ D303
	从站 7	无	D350 ~ D353
	从站 08	无	D400 ~ D403
	从站 09	无	D450 ~ D453
	从站 10	无	D500 ~ D503
	从站 11	无	D550 ~ D553
	从站 12	无	D600 ~ D603
	从站 13	无	D650 ~ D653
	从站 14	无	D700 ~ D703
	从站 15	无	D750 ~ D753
	从站 16	无	D800 ~ D803
	从站 17	无	D850 ~ D853
	从站 18	无	D900 ~ D903
	从站 19	无	D950 ~ D953
	从站 20	无	D1000 ~ D1003
	从站 21	无	D1050 ~ D1053
	从站 22	无	D1100 ~ D1103
	从站 23	无	D1150 ~ D1153
	从站 24	无	D1200 ~ D1203
	从站 25	无	D1250 ~ D1253
	从站 26	无	D1300 ~D1303
	从站 27	无	D1350 ~ D1353
	从站 28	无	D1400 ~ D1403
	从站 29	无	D1450 ~ D1453
	从站 30	无	D1500 ~ D1503
	从站 31	无	D1550 ~ D1553

模式 1 32 个 M 元件 4 个 D 元件	主站	M1000 ~M1031	D0 ~ D3
	从站 1	M1064 ~M1095	D50 ~ D53
	从站 2	M1128 ~M1159	D100 ~ D103
	从站 3	M1192 ~M1223	D150 ~ D153
	从站 4	M1256 ~M1287	D200 ~ D203
	从站 5	M1320 ~M1351	D250 ~ D253
	从站 6	M1384 ~M1415	D300 ~ D303
	从站 7	M1448 ~M1479	D350 ~ D353
	从站 08	M1512 ~M1543	D400 ~ D403
	从站 09	M1576 ~M1607	D450 ~ D453
	从站 10	M1640 ~M1671	D500 ~ D503
	从站 11	M1704 ~M1735	D550 ~ D553
	从站 12	M1768 ~M1799	D600 ~ D603
	从站 13	M1832 ~M1863	D650 ~ D653
	从站 14	M1896 ~M1927	D700 ~ D703
	从站 15	M1960 ~M1991	D750 ~ D753
	从站 16	M2024 ~M2055	D800 ~ D803
	从站 17	M2088 ~M2119	D850 ~ D853
	从站 18	M2152 ~M2183	D900 ~ D903
	从站 19	M2216 ~M2247	D950 ~ D953
	从站 20	M2280 ~M2311	D1000 ~ D1003
	从站 21	M2344 ~M2375	D1050 ~ D1053
	从站 22	M2408 ~M2439	D1100 ~ D1103
	从站 23	M2472 ~M2503	D1150 ~ D1153
	从站 24	M2536 ~M2567	D1200 ~ D1203
	从站 25	M2600 ~M2631	D1250 ~ D1253
	从站 26	M2664 ~M2695	D1300 ~D1303

	从站 27	M2728 ~M2759	D1350 ~ D1353
	从站 28	M2792 ~M2823	D1400 ~ D1403
	从站 29	M2856 ~M2887	D1450 ~ D1453
	从站 30	M2920 ~M2951	D1500 ~ D1503
	从站 31	M2984 ~M3015	D1550 ~ D1553
模式 2 64 个 M 元件 8 个 D 元件	主站	M1000 ~M1063	D0 ~ D7
	从站 1	M1064 ~M1127	D50 ~ D57
	从站 2	M1128 ~M1191	D100 ~ D107
	从站 3	M1192 ~M1255	D150 ~ D157
	从站 4	M1256 ~M1319	D200 ~ D207
	从站 5	M1320 ~M1383	D250 ~ D257
	从站 6	M1384 ~M1447	D300 ~ D307
	从站 7	M1448 ~M1511	D350 ~ D357
	从站 08	M1512 ~M1575	D400 ~ D407
	从站 09	M1576 ~M1639	D450 ~ D457
	从站 10	M1640 ~M1703	D500 ~ D507
	从站 11	M1704 ~M1767	D550 ~ D557
	从站 12	M1768 ~M1831	D600 ~ D607
	从站 13	M1832 ~M1895	D650 ~ D657
	从站 14	M1896 ~M1959	D700 ~ D707
	从站 15	M1960 ~M2023	D750 ~ D757
	从站 16	M2024 ~M2087	D800 ~ D807
	从站 17	M2088 ~M2151	D850 ~ D857
	从站 18	M2152 ~M2215	D900 ~ D907
	从站 19	M2216 ~M2279	D950 ~ D957
	从站 20	M2280 ~M2343	D1000 ~ D1007
	从站 21	M2344 ~M2407	D1050 ~ D1057

	从站 22	M2408 ~M2471	D1100 ~ D1107
	从站 23	M2472 ~M2535	D1150 ~ D1157
	从站 24	M2536 ~M2599	D1200 ~ D1207
	从站 25	M2600 ~M2663	D1250 ~ D1257
	从站 26	M2664 ~M2727	D1300 ~D1307
	从站 27	M2728 ~M2791	D1350 ~ D1357
	从站 28	M2792 ~M2855	D1400 ~ D1407
	从站 29	M2856 ~M2919	D1450 ~ D1457
	从站 30	M2920 ~M2983	D1500 ~ D1507
	从站 31	M2984 ~M3015	D1550 ~ D1557
模式 3 64 个 M 元件 32 个 D 元件	主站	M1000 ~M1063	D0 ~ D31
	从站 01	M1064 ~M1127	D50 ~ D81
	从站 02	M1128 ~M1191	D100 ~ D131
	从站 03	M1192 ~M1255	D150 ~ D181
	从站 04	M1256 ~M1319	D200 ~ D231
	从站 05	M1320 ~M1383	D250 ~ D281
	从站 06	M1384 ~M1447	D300 ~ D331
	从站 07	M1448 ~M1511	D350 ~ D381
	从站 08	M1512 ~M1575	D400 ~ D431
	从站 09	M1576 ~M1639	D450 ~ D481
	从站 10	M1640 ~M1703	D500 ~ D531
	从站 11	M1704 ~M1767	D550 ~ D581
	从站 12	M1768 ~M1831	D600 ~ D631
	从站 13	M1832 ~M1895	D650 ~ D681
	从站 14	M1896 ~M1959	D700 ~ D731
	从站 15	M1960 ~M2023	D750 ~ D781
	从站 16	M2024 ~M2087	D800 ~ D831

	从站 17	M2088 ~M2151	D850 ~ D881
	从站 18	M2152 ~M2215	D900 ~ D931
	从站 19	M2216 ~M2279	D950 ~ D981
	从站 20	M2280 ~M2343	D1000 ~ D1031
	从站 21	M2344 ~M2407	D1050 ~ D1081
	从站 22	M2408 ~M2471	D1100 ~ D1131
	从站 23	M2472 ~M2535	D1150 ~ D1181
	从站 24	M2536 ~M2599	D1200 ~ D1231
	从站 25	M2600 ~M2663	D1250 ~ D1281
	从站 26	M2664 ~M2727	D1300 ~D1331
	从站 27	M2728 ~M2791	D1350 ~ D1381
	从站 28	M2792 ~M2855	D1400 ~ D1431
	从站 29	M2856 ~M2919	D1450 ~ D1481
	从站 30	M2920 ~M2983	D1500 ~ D1531
	从站 31	M2984 ~M3015	D1550 ~ D1581

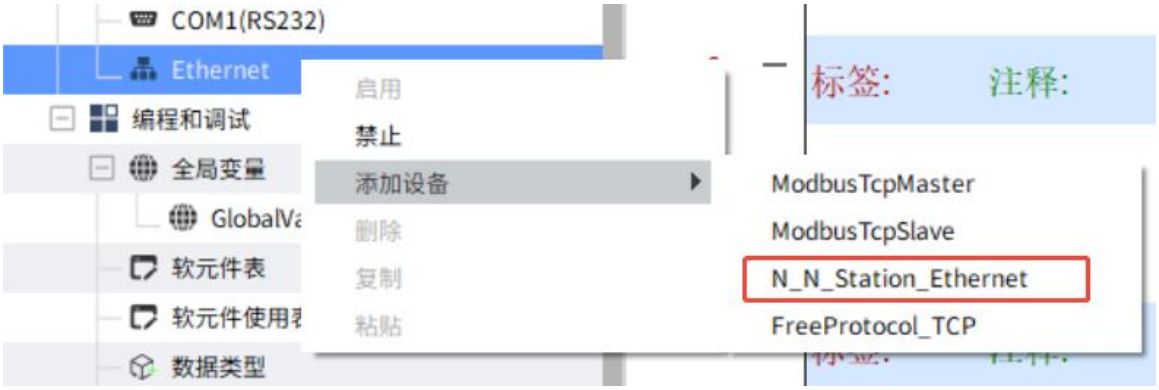
8.2 使用步骤

8.2.1 非自定义模式（通过软件进行配置）

1）添加 N_N 协议串口/网口

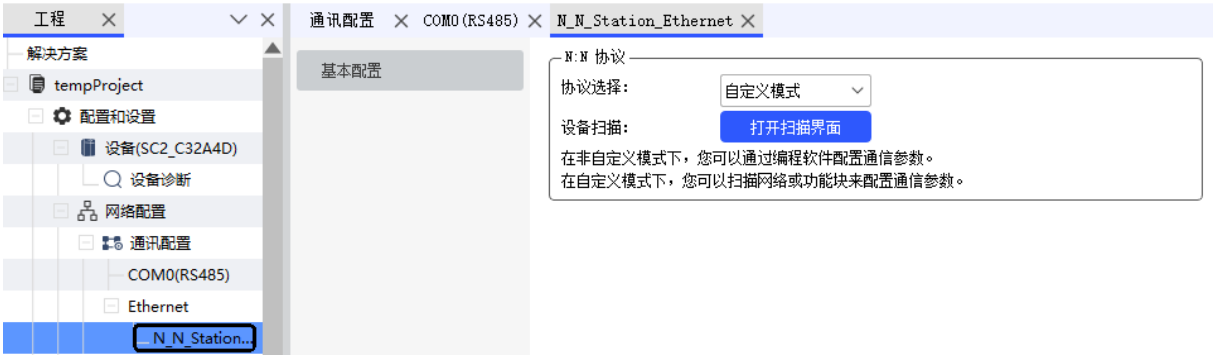
①使用串口模式（RS485），添加 N_N_Station_COM0

②使用以太网模式，添加 N_N_Station_Ethernet



注：以太网与串口模式无法同时使用

2）添加协议串口/网口后，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

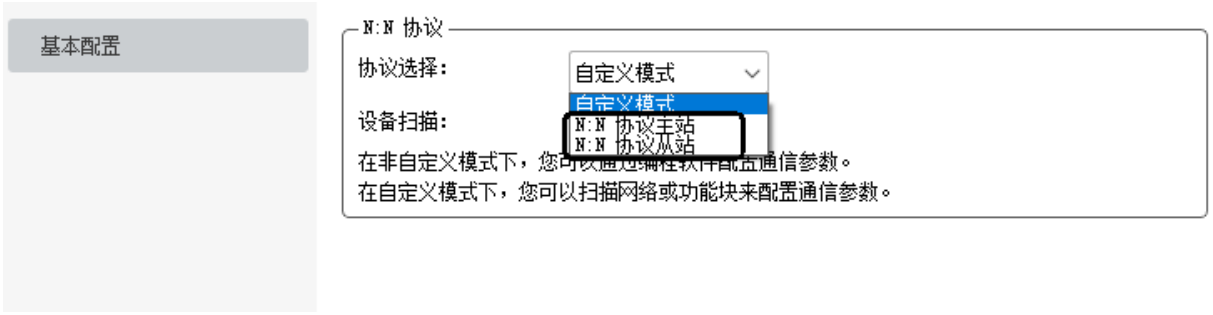


非自定义模式：通过编程软件配置通信参数

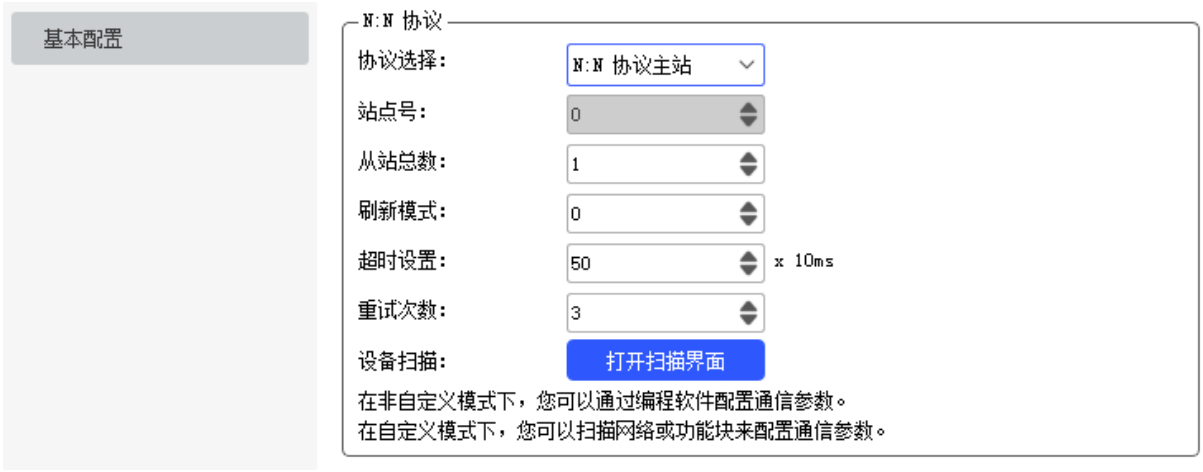
自定义模式：通过扫描（仅以太网支持）或功能块来配置通信参数

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

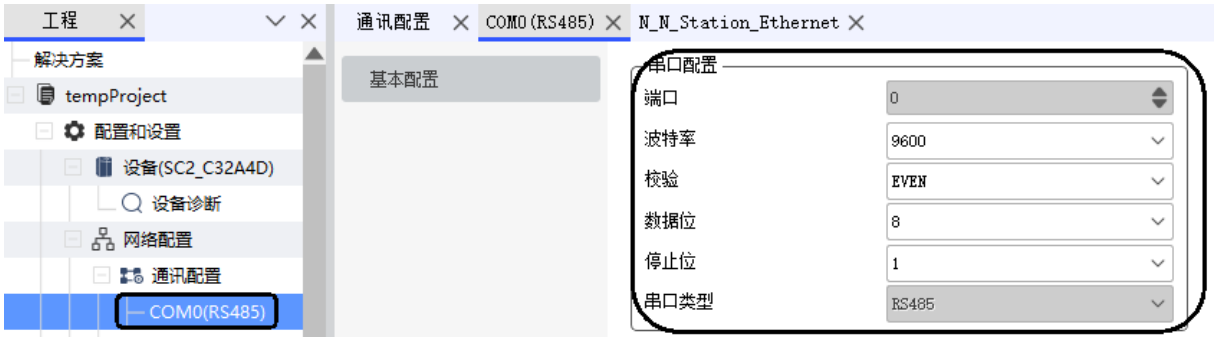
⑤重试次数

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



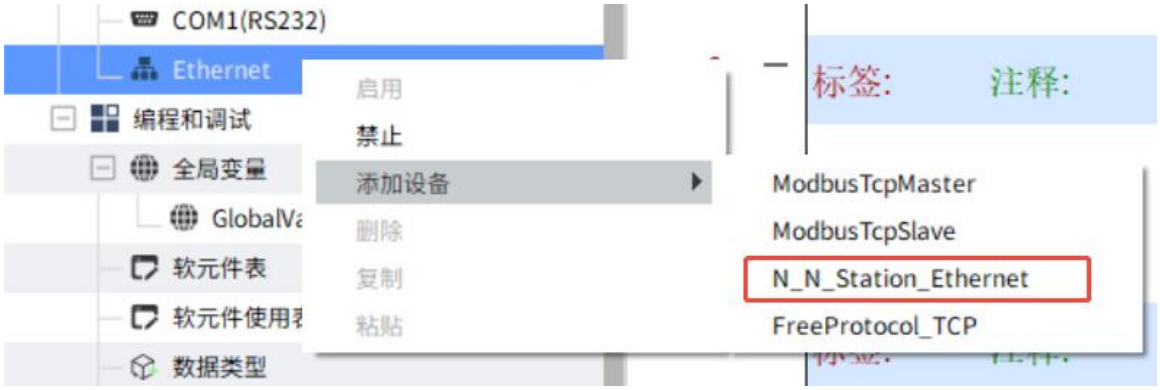
2）参数配置需要下载生效

8.2.2 自定义模式（通过扫描或功能块进行配置）

1）添加 N_N 协议串口/网口

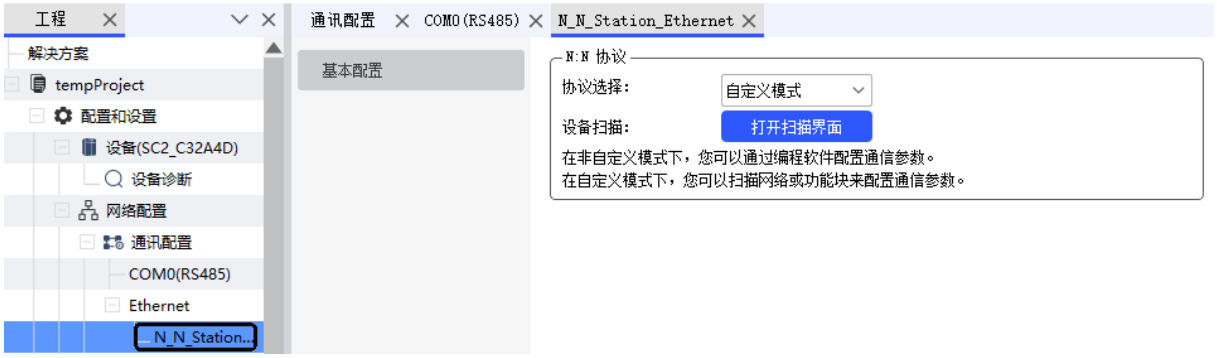
①使用串口模式（RS485），添加 N_N_Station_COM0

②使用以太网模式，添加 N_N_Station_Ethernet



注：以太网与串口模式无法同时使用

2）添加协议串口/网口后，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

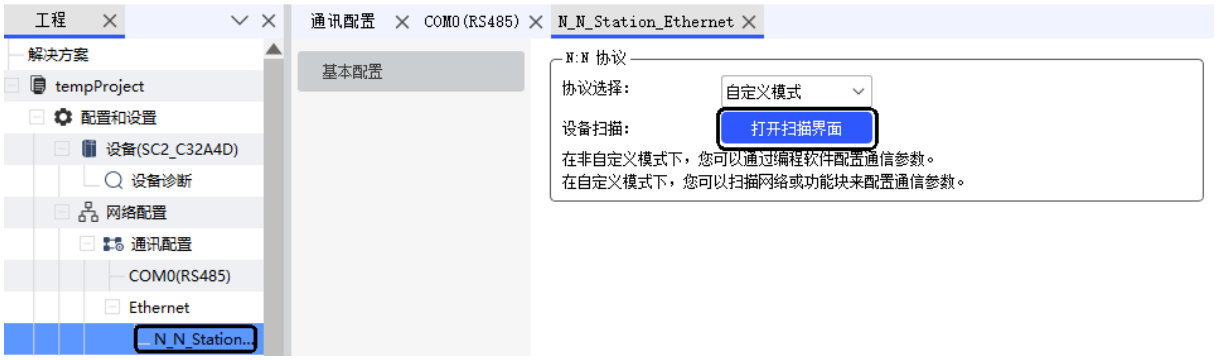


非自定义模式：通过编程软件配置通信参数

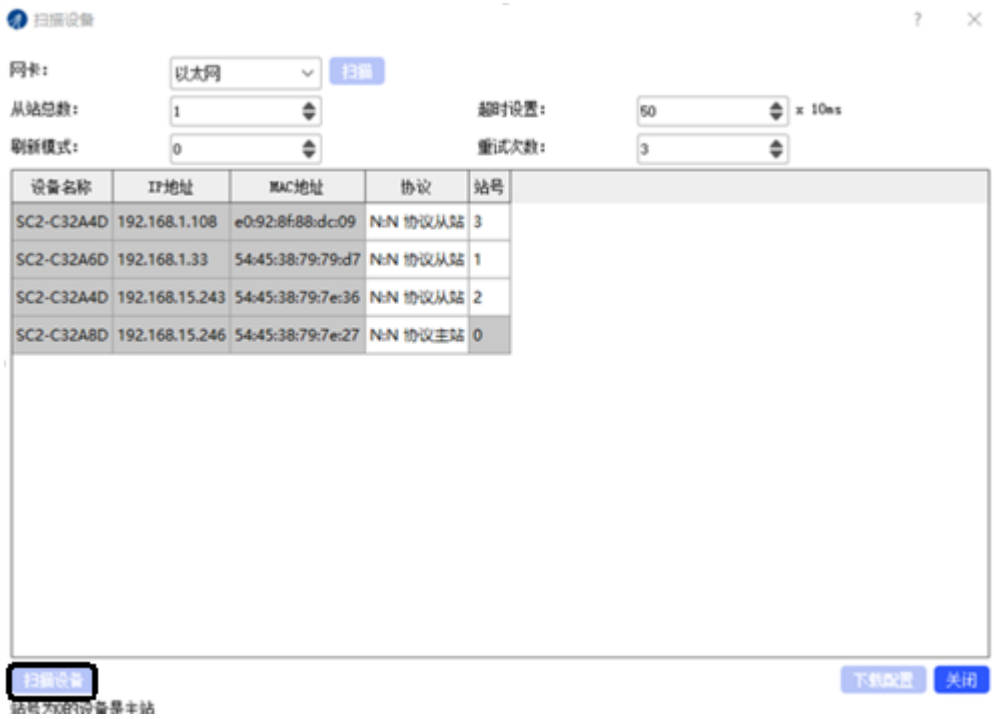
自定义模式：通过扫描（仅以太网支持）或功能块来配置通信参数

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-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。
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	LS_NNB	-	0	正常时数值为 0，发生异

	码	us_Error ID			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

8.3 故障诊断

可以通过结构体变量“NNBusDevice”查看 NN 协议相关的错误，该变量包含以下成员：

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

9 运动控制功能

9.1 概述

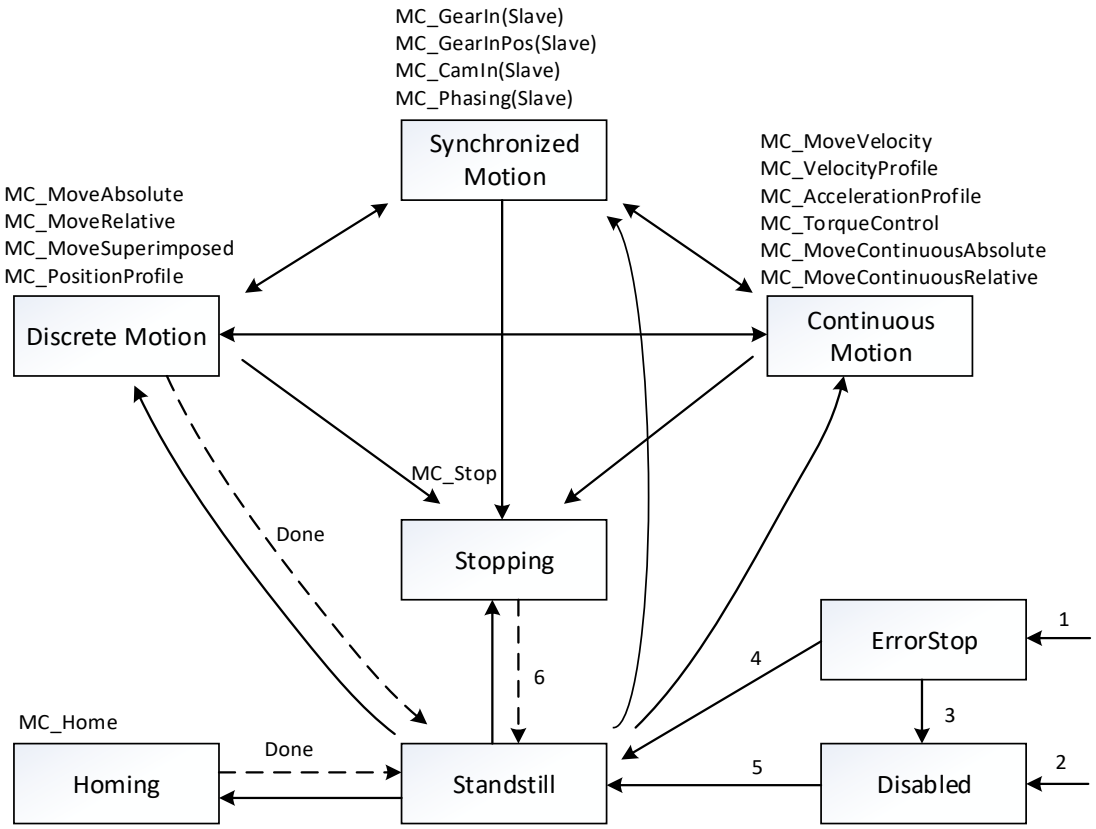
9.1.1 控制基本逻辑

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。

9.1.2 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：

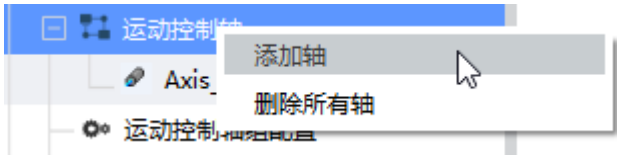
名称	值	注释
Power_off（Disabled）	0	轴未使能，需执行 MC_Power 指令
Errorstop	1	故障停止状态，需先执行 MC_Reset/MC_Power 指令
Stopping	2	停止中，等待停机操作完成
Standstill	3	使能状态，轴已停止运行
Discrete_Motion	4	轴处于离散运行状态
Continuous_Motion	5	轴处于连续运行中
Synchronized_Motion	6	轴处于同步运行中
Homing	7	轴处于回零运行中，等待归零操作执行完成

9.2 本地脉冲轴组态

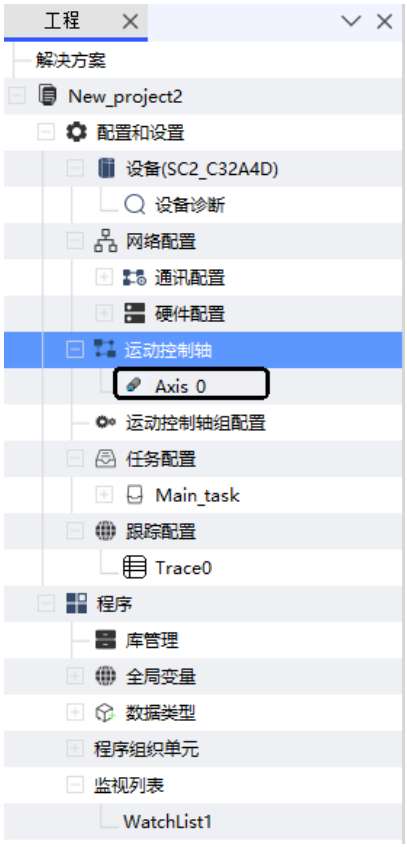
要正确控制运动控制轴，首先根据项目需要创建对应的轴，然后根据工况设定相关的轴配置参数。

添加运动控制轴的操作如下：

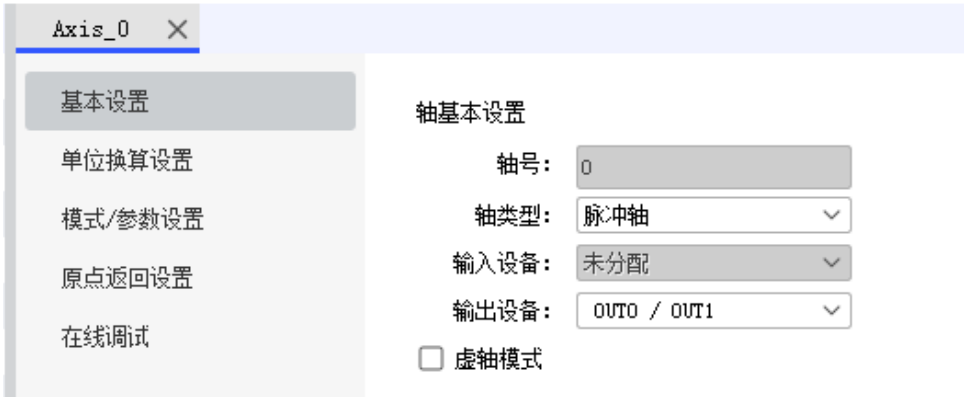
1) 右键“工程管理树”中的“运动控制轴”，选择“添加轴”；



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；

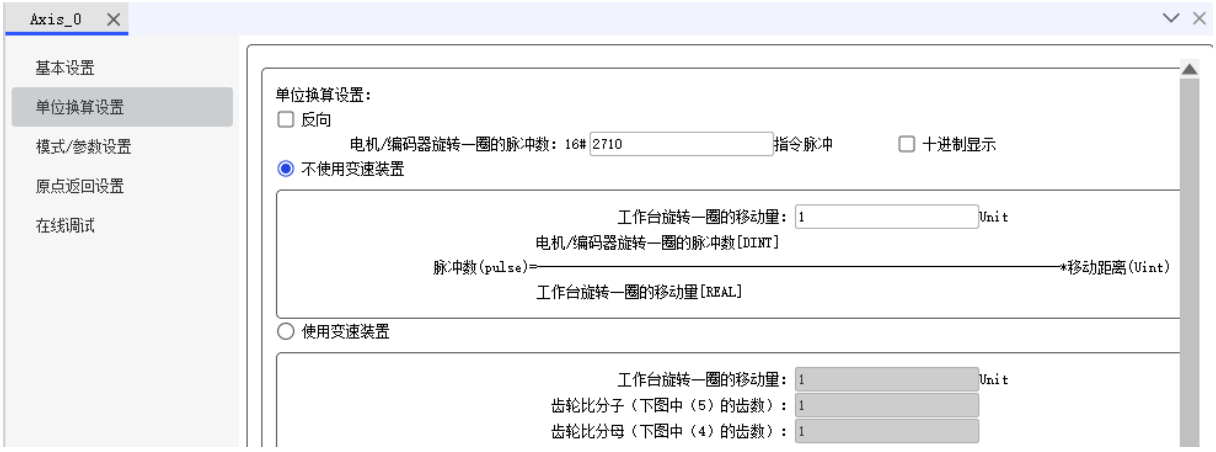


9.3 基本设置



- 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

9.4 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：

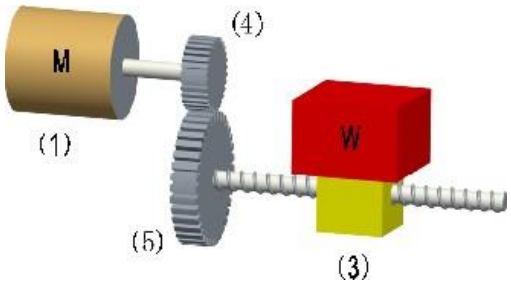
- 不使用变速装置

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：

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

- 使用变速装置

在线性模式下典型工况如下图所示：

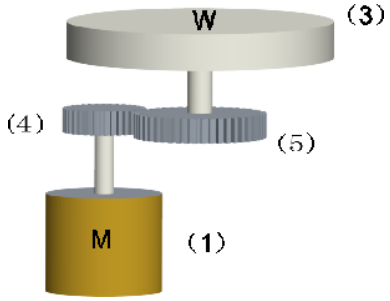


其中(1)为电机，(3)为工作台，(4)为齿轮比分子，(5)为齿轮比分母。

由用户单位到脉冲单位的计算公式如下：

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

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：

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

9.5 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；

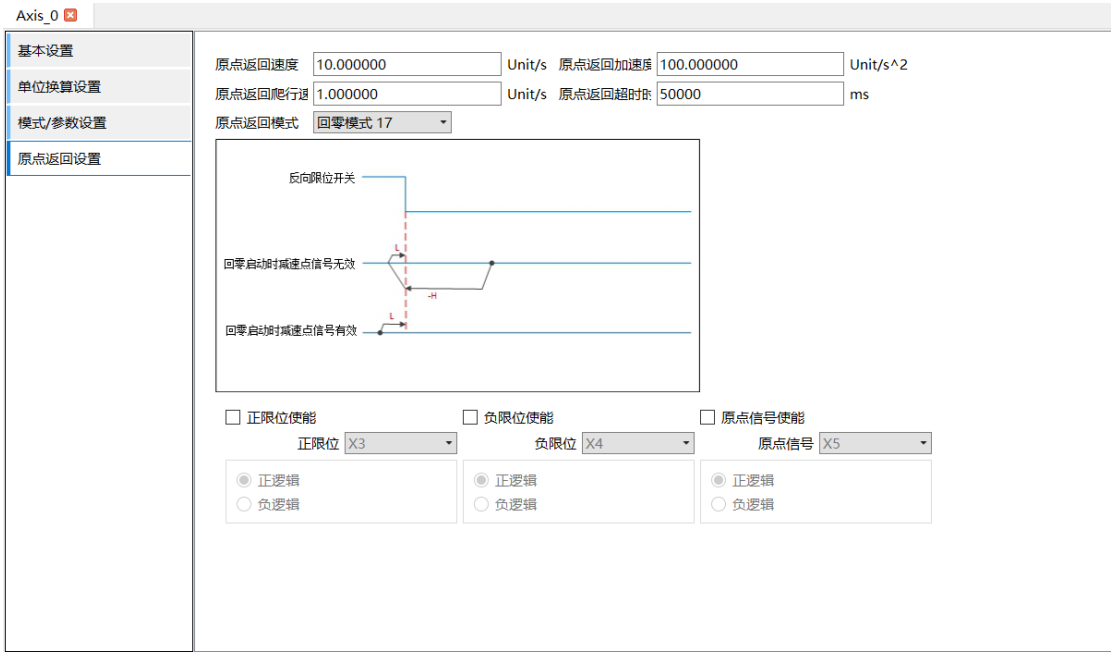
  最大加速度：运动控制轴运行的最大加速度限制；
- 输入信号设置：

  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。

9.6 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。

9.7 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。

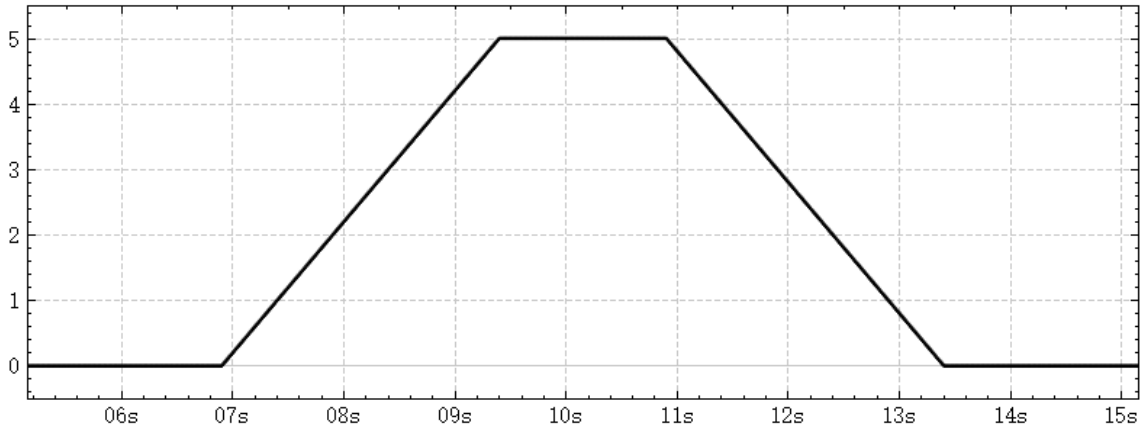
注：轴最大速度可在轴模式/参数界面中设置，见章节 12.5。

2.轴的起始速度和结束速度

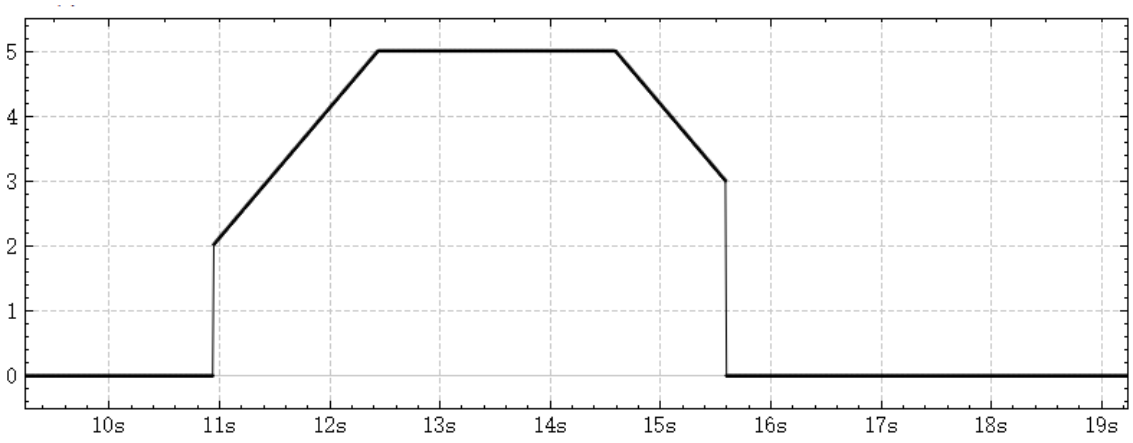
通过轴结构体 fStartVelocity 和 fEndVelocity 可分别设置轴的【起始速度】与【结束速度】，用户单位/s。

【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



9.8 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

9.9 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

10 电子凸轮

10.1 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。

10.1.1 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

10.1.2 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

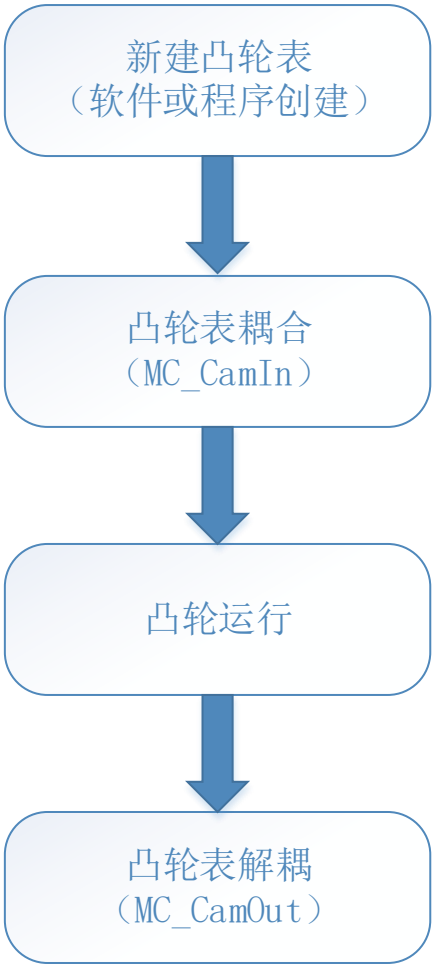
Cam_Node		
变量	数据类型	备注
fPhase	LREAL	主轴相位
fDistance	LREAL	从轴位置
fVelocity	LREAL	连接点速度
fAcceleration	LREAL	连接点加速度
wCurve	WORD	曲线类型 0-NA，1-直线，5-5 次曲线
align	Array[1..3] of WORD	保留

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

10.2 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。



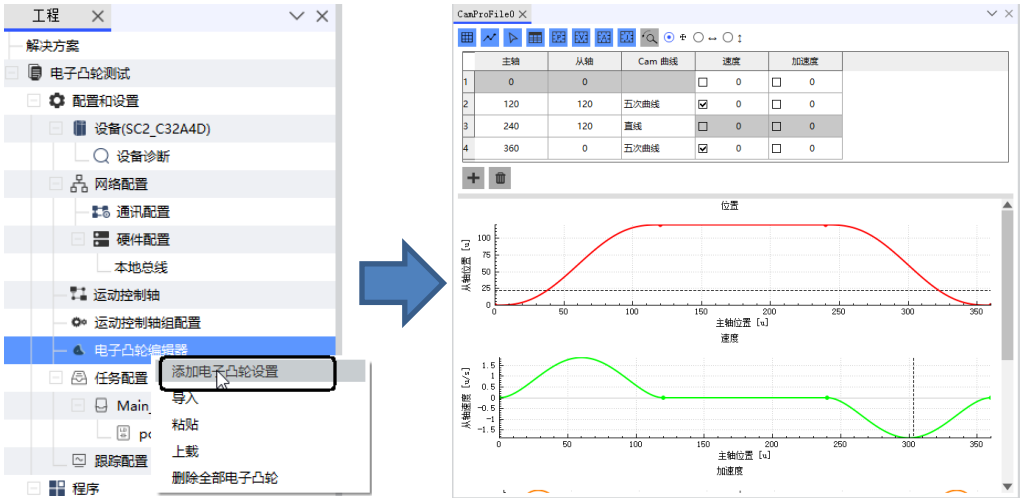
注：

- 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；

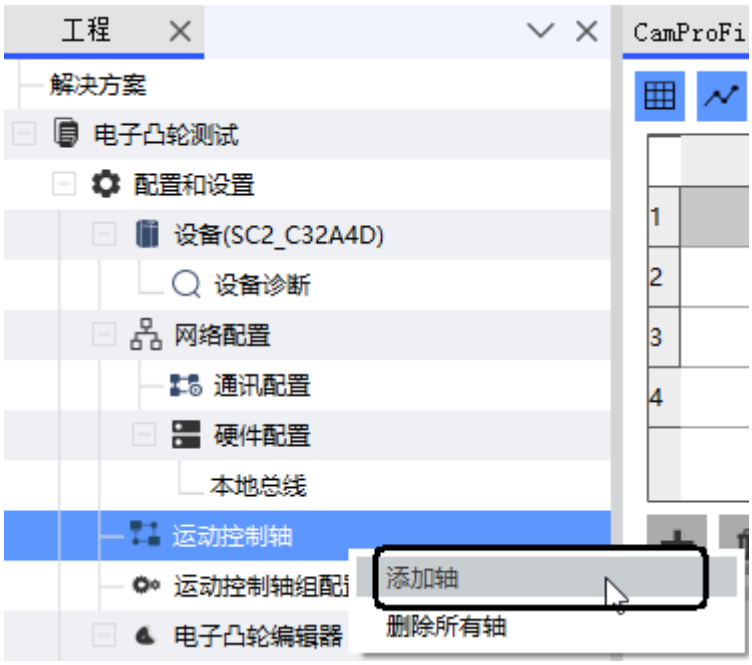
10.2.1 电子凸轮程序使用步骤

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：

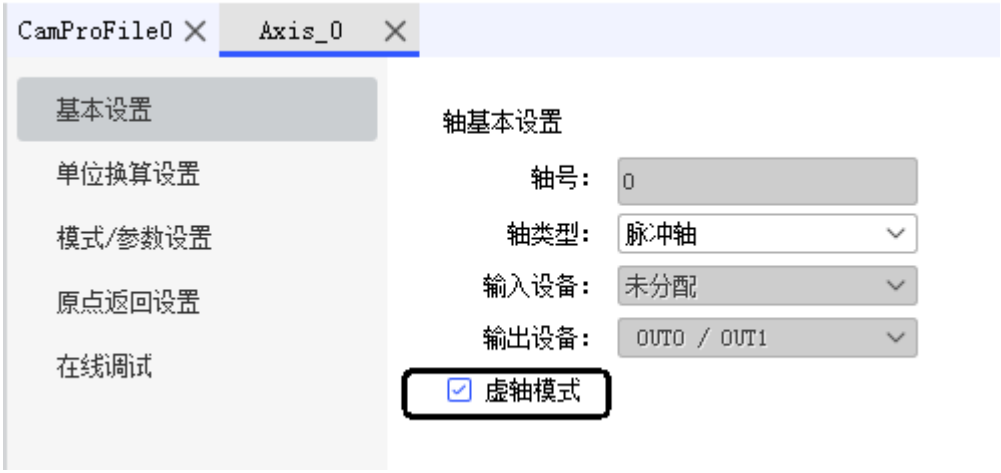
1）右键点击“电子凸轮编辑器”选择“添加电子凸轮设置”创建凸轮表，并根据需要修改凸轮表曲线。



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

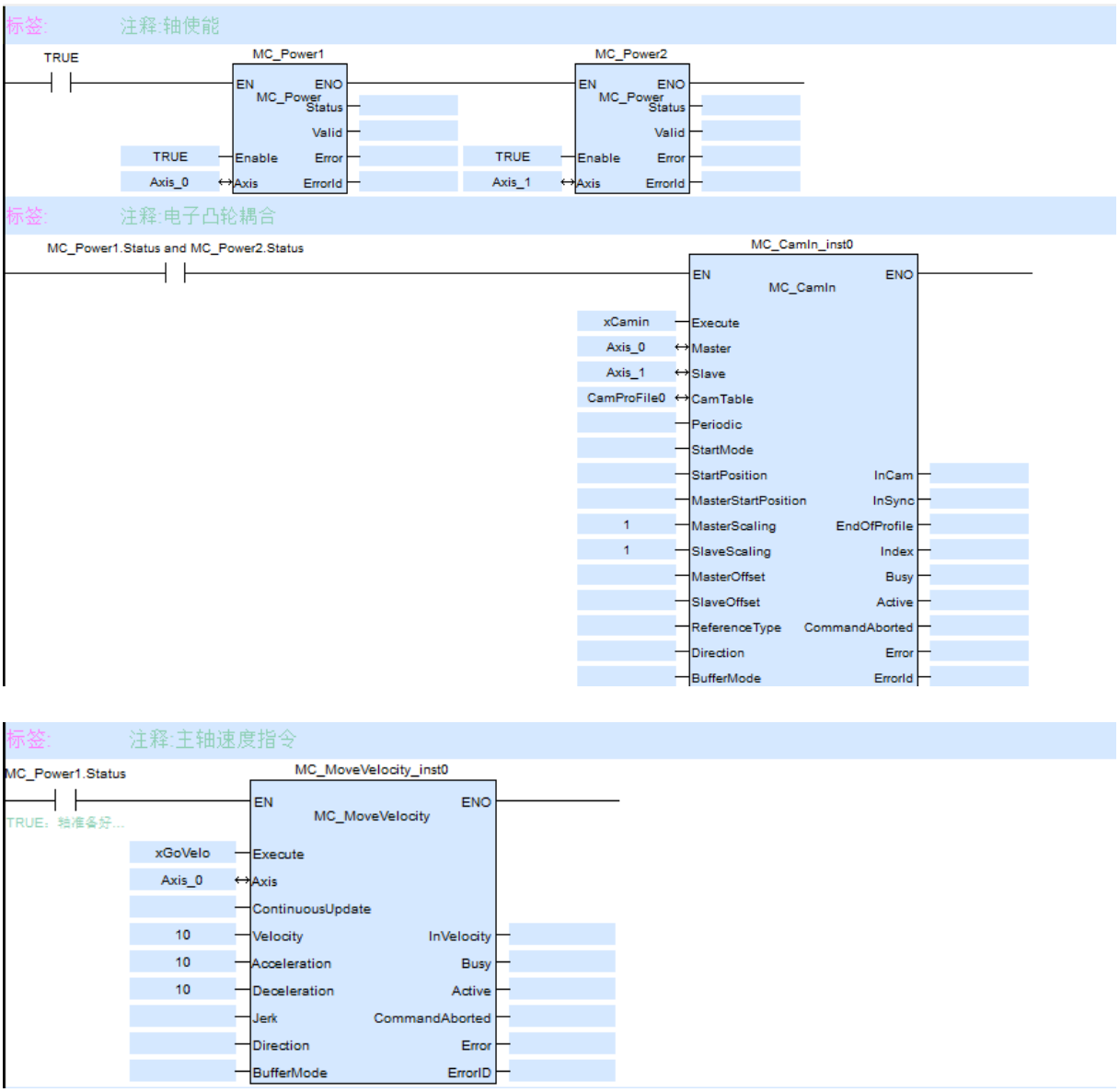


4）重复 3）操作，再次添加一个虚轴作为从轴使用

5）实例化功能块，声明相关变量。

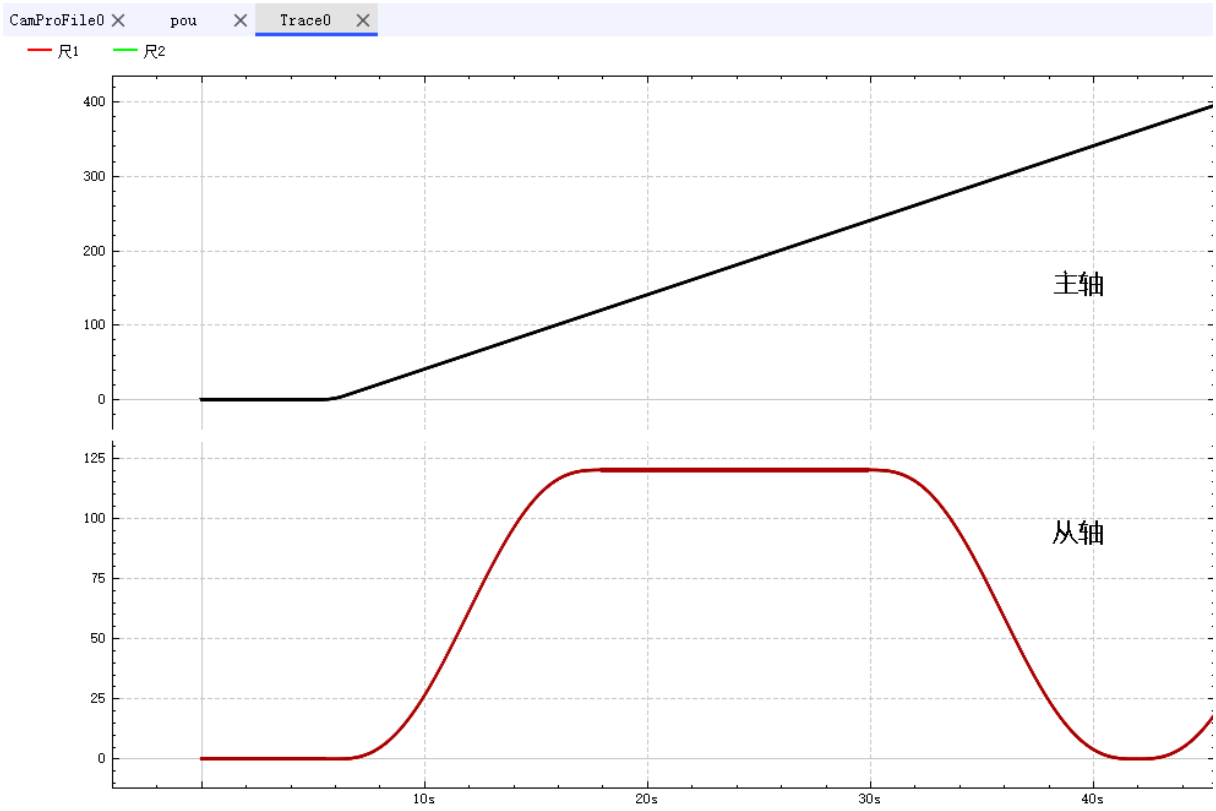
```
VAR
MC_Power1:MC_Power;
MC_Power2:MC_Power;
MC_CamIn_inst0:MC_CamIn;
xCamin:BOOL;
MC_MoveVelocity_inst0:MC_MoveVelocity;
xGoVelo:BOOL;
END_VAR
```

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：



10.3 凸轮表

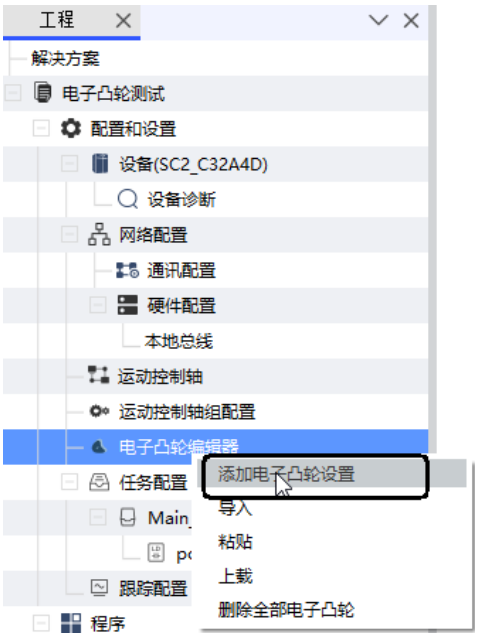
在凸轮功能中，将由主轴相位和从轴位移组成的一对数据称之为凸轮关键点，凸轮表即为多个凸轮关键点数据组成的数据组合。

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。

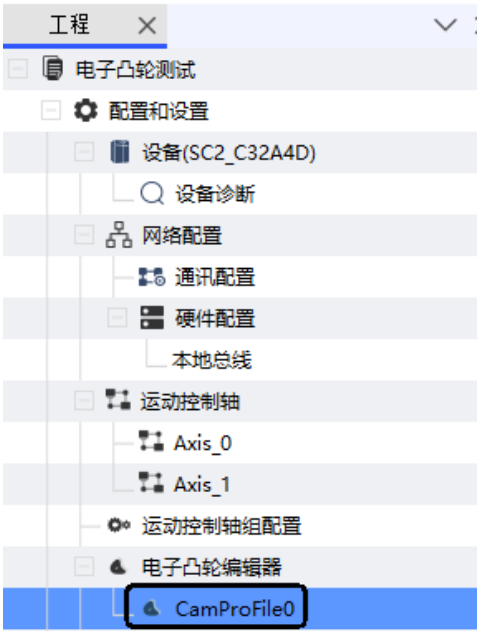
10.3.1 通过软件编辑凸轮表

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：

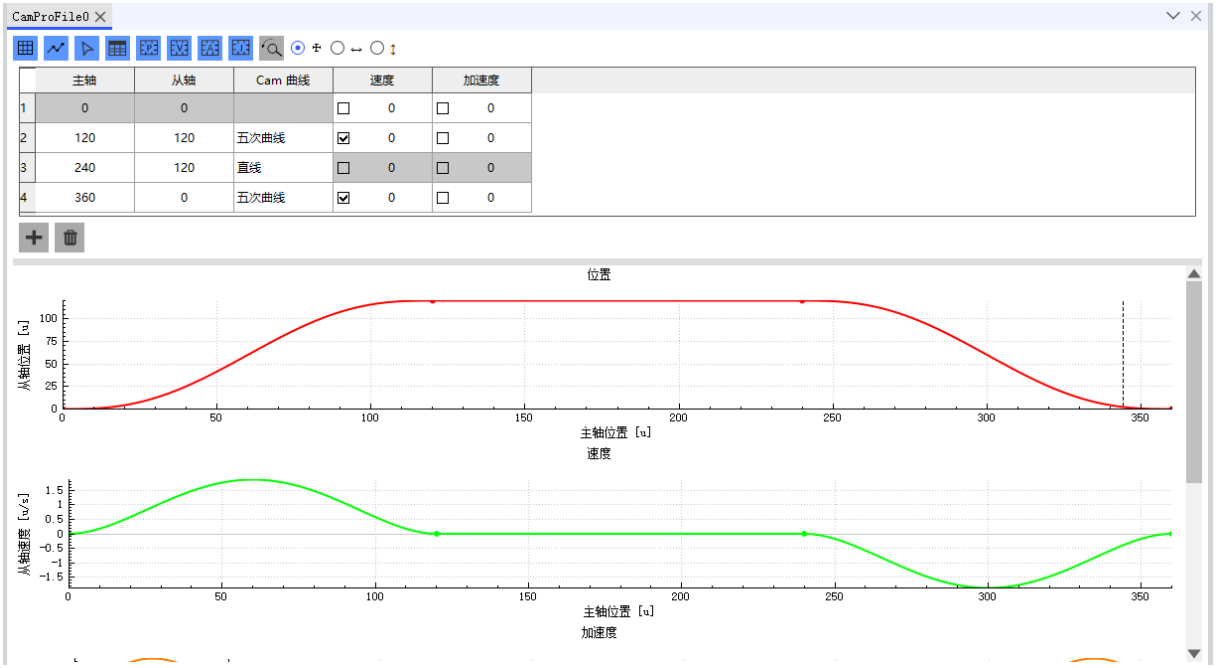
1）右键点击“电子凸轮编辑器”选择“添加电子凸轮设置”创建凸轮表.



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。

10.3.2 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

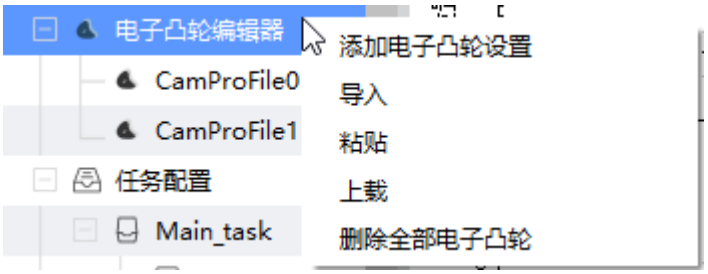
注：

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。

10.3.3 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。

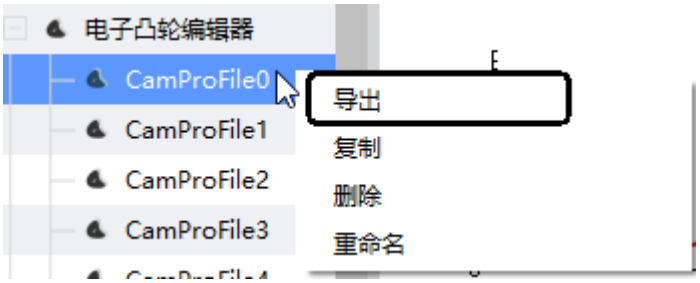


上载后的凸轮表会以 CamProFile+序号的方式进行命名，目前，只支持通过软件创建的凸轮表进行上传，通过程序创建的凸轮表暂时无法被上传。

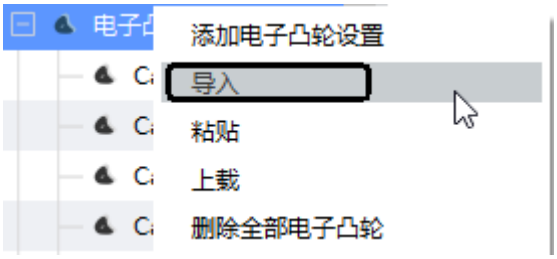
10.3.4 凸轮表导入导出

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。



11 高速计数器

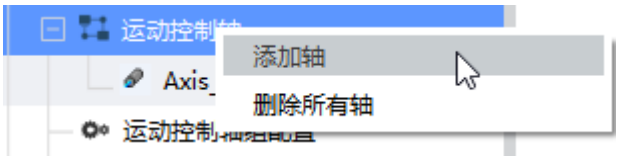
11.1 高速计数器轴简介

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。

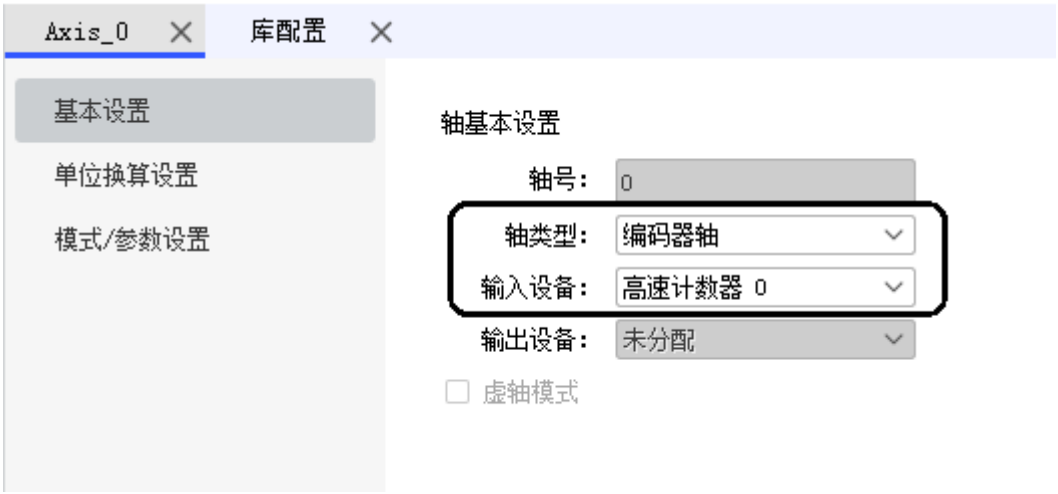
11.2 创建计数器轴

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。

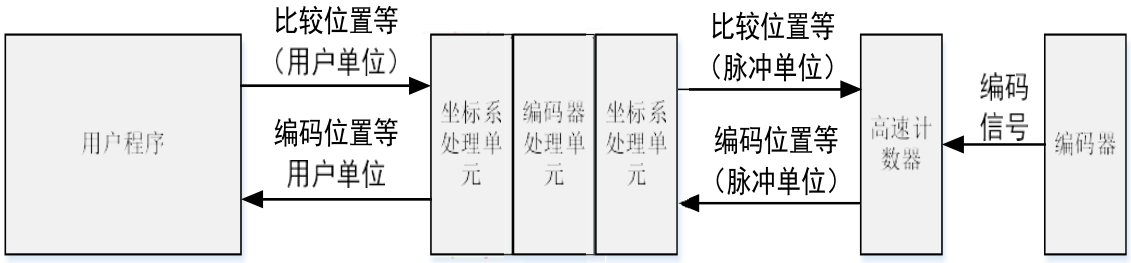


双击新添加的轴（如图 Axis_0）打开设置页面，在“基本设置”界面选择“本地编码器轴”作为轴类型， 选择“高速计数器”作为输入设备，即可将轴和计数器关联。轴号作为轴标识在程序中使用，实现对应计数器轴的控制。

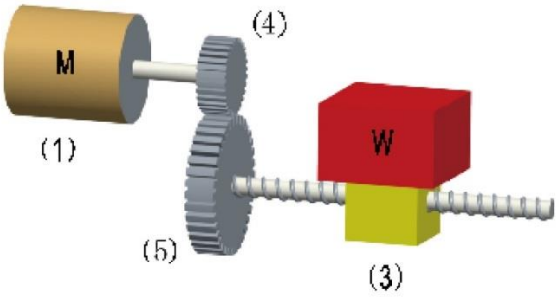


11.3 计数器轴用户单位与换算

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）。



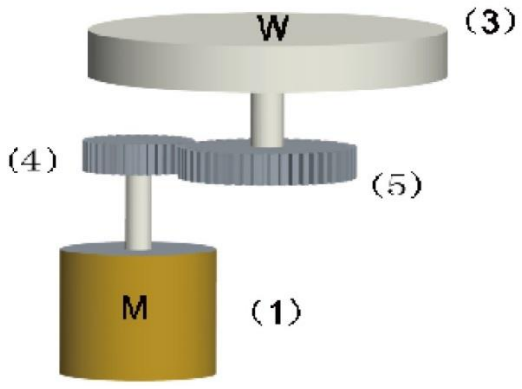
M：电机/编码器，W：工作台



轴类型为旋转模式：

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

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：

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

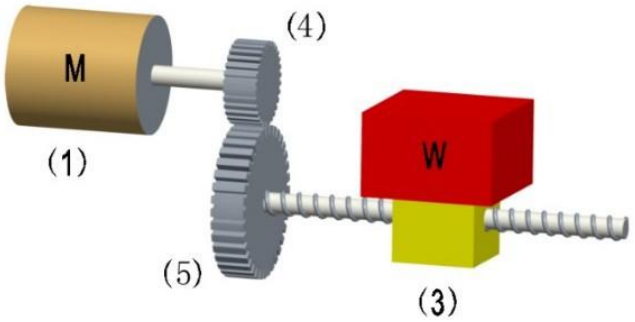
如编码器旋转一圈对应的工作轴旋转一圈，用户单位（Unit）为转，则电机/编码器旋转一圈的工作行程设置为 1 即可。

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：

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

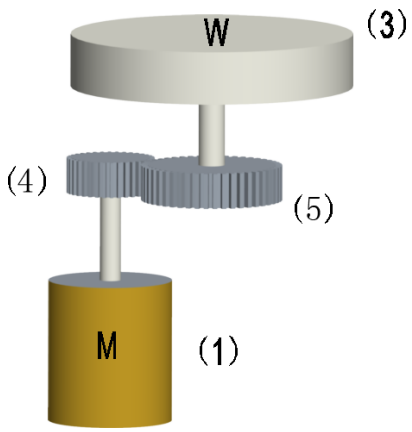
工作台旋转一圈的移动量: 1.000000 Unit

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：

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

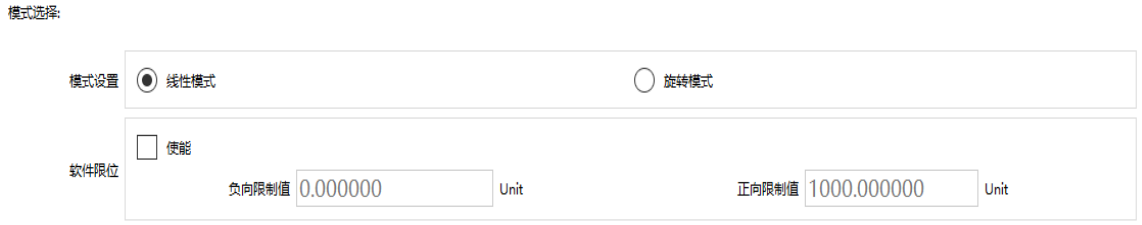
11.4 设置工作模式

11.4.1 线性模式

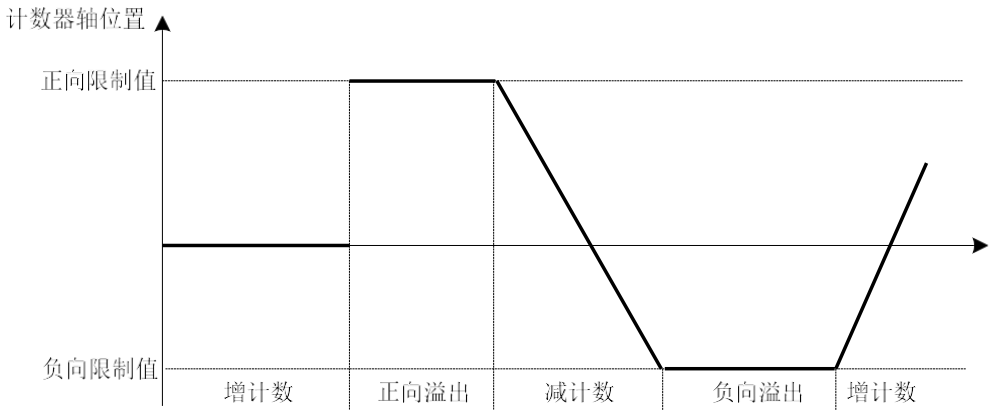
计数器轴的位置在负向限制值和正向限制值之间变化，计数器轴的位置达到限制值后，继续输入同向脉冲，计数器轴报溢出，同时计数器轴位置保持不变。计数器轴报溢出后，输入反向脉冲，计数器轴反向计数，溢出错误清除。

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。

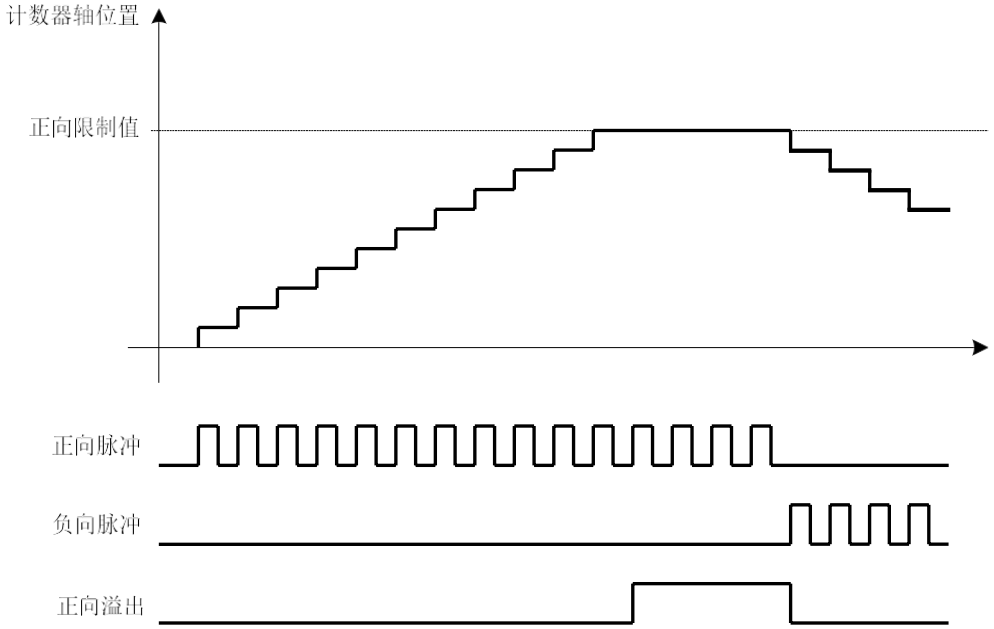


线性模式

正向脉冲计数

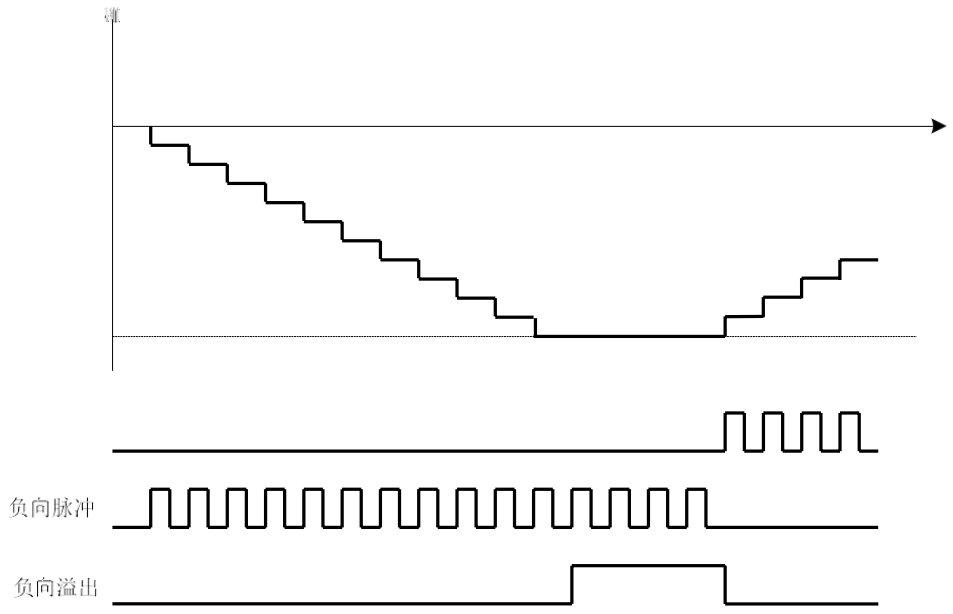
线性模式下，输入正向脉冲，计数器轴位置增计数达到限制值后，继续输入正向脉

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。



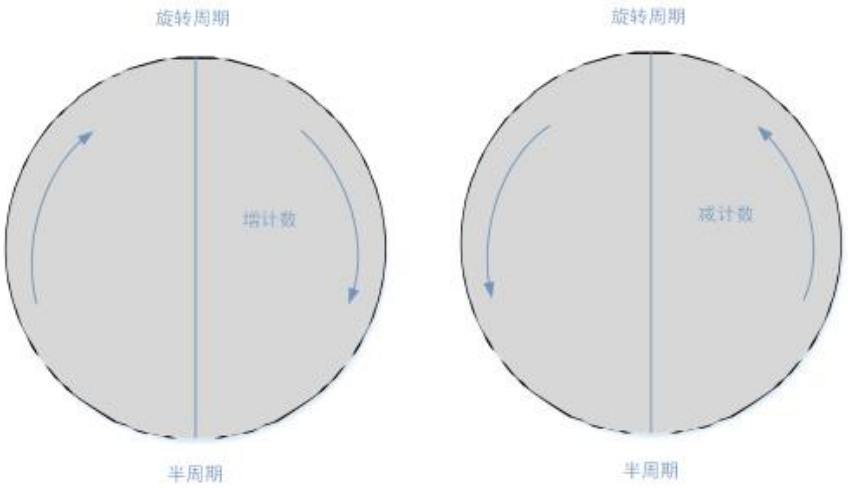
负向脉冲计数

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。



11.4.2 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]。



11.5 设置计数器参数

11.5.1 概述

参数设置主要包括计数模式、探针、预置、比较输出功能。

11.5.2 计数模式

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。

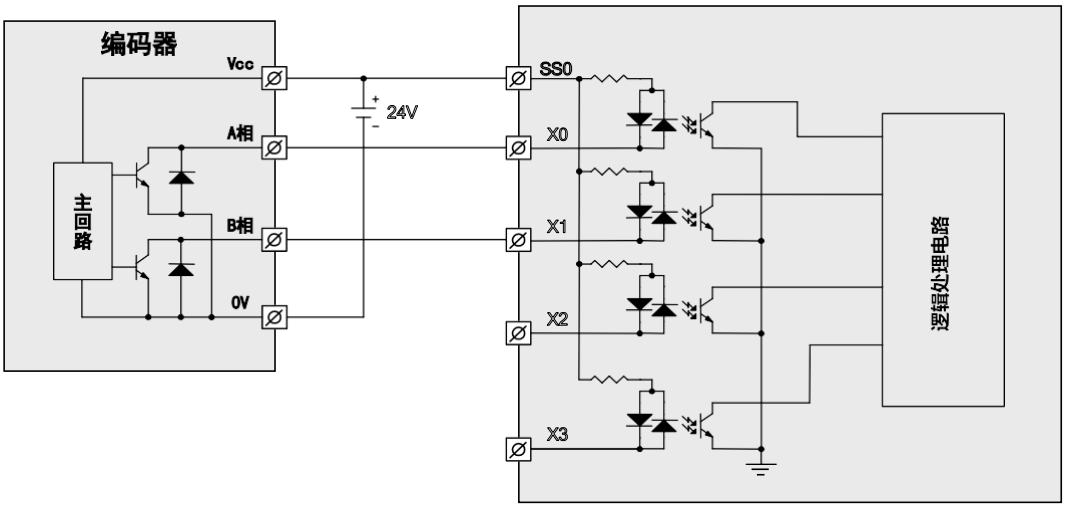
指令类别	IN0	IN1	IN 2	IN 3	IN 4	IN 5	IN 6	IN 7	内部 1ms	内部 1us
A/B 相	A 相	B 相	A 相	B 相	A 相	B 相	A 相	B 相	×	×
脉冲+方向	脉冲	方向	脉冲	方向	脉冲	方向	脉冲	方向	×	×
单相计数	脉冲	脉冲	脉冲	脉冲	脉冲	脉冲	脉冲	脉冲	脉冲	脉冲

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

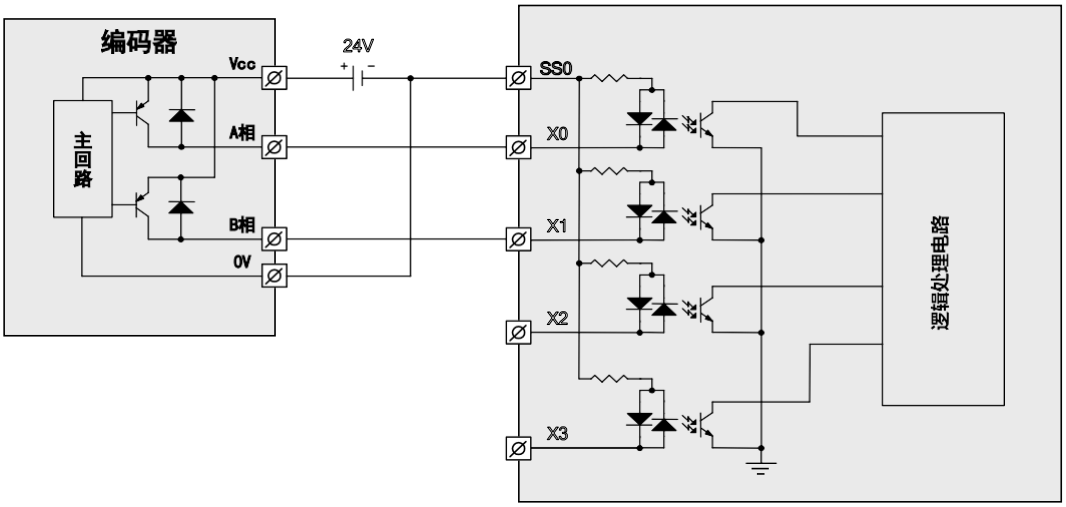
SC 系列 PLC 支持的 4 个计数器可以任意选择上面计数模式和信号源，不同计数器可以复用计数模式或信号源。

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。



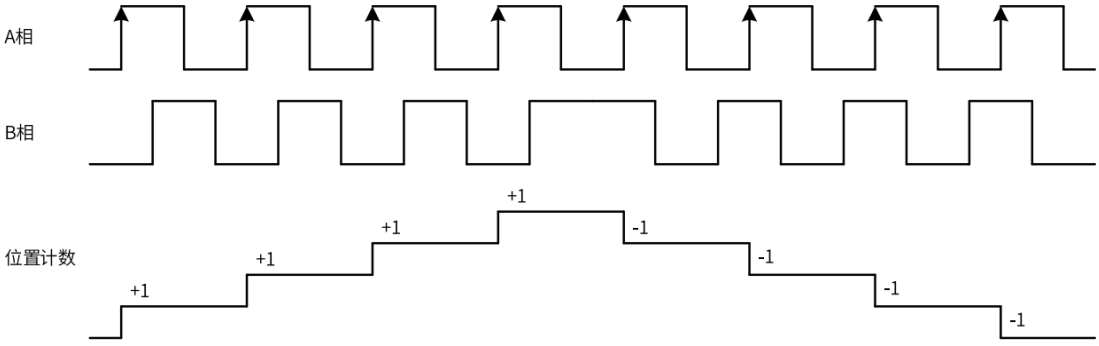
A/B 相编码器接线图--漏型输入接法



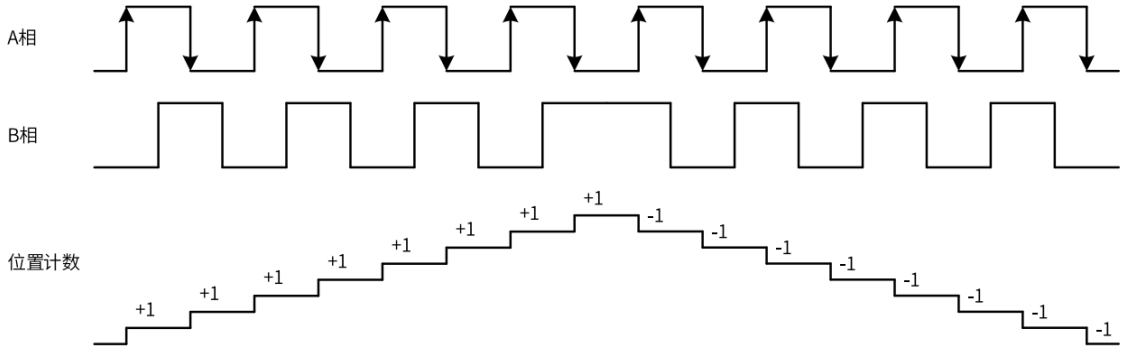
源型输入接法

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。

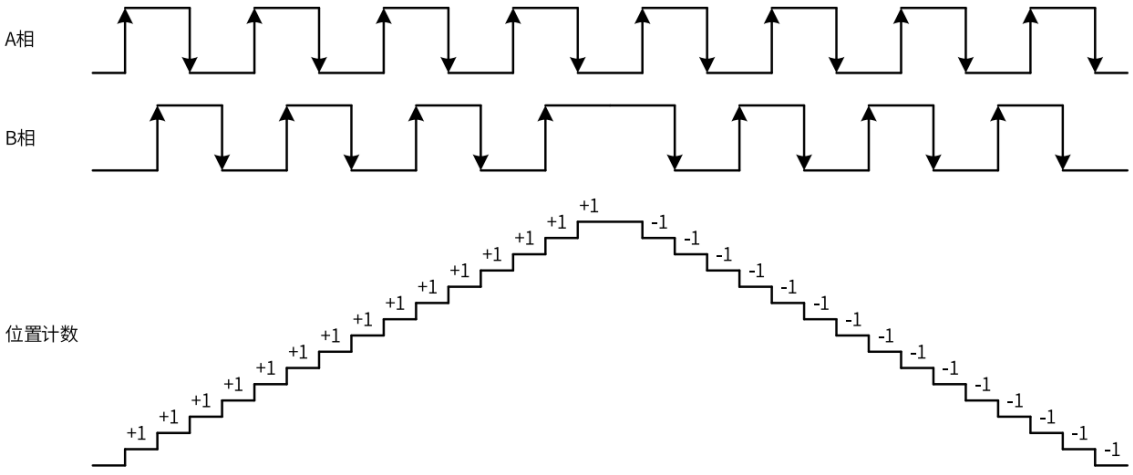
- A/B 相 1 倍频模式下，只对 A 相脉冲的上升沿计数，如下图所示：



- 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：

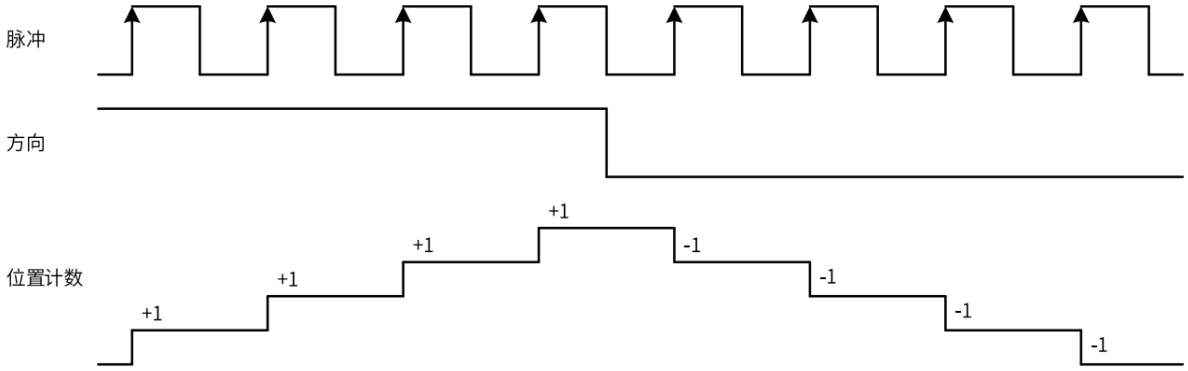


- A/B 相 4 倍频模式下，对 A 相脉冲和 B 相脉冲的上升/下降沿计数，如下图所示：



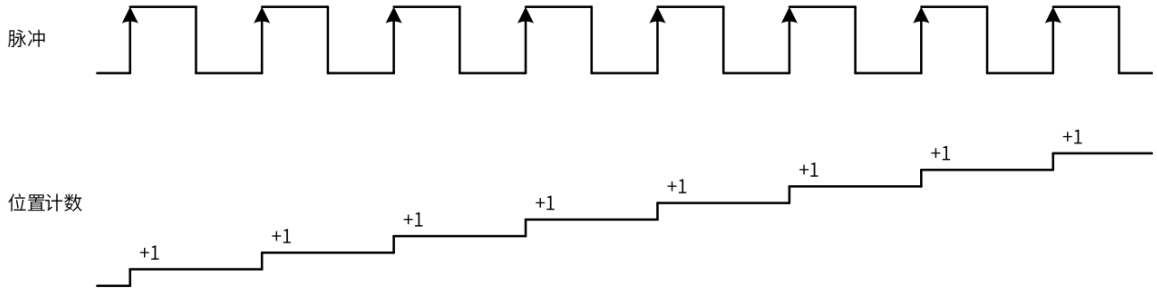
脉冲+方向模式

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：



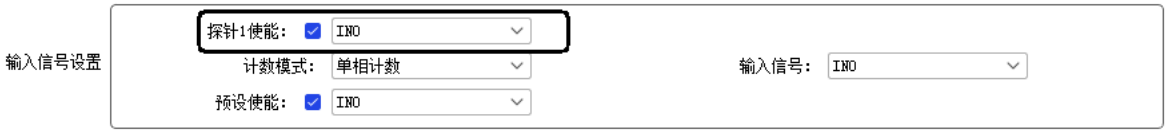
单相计数

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。



11.5.3 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。

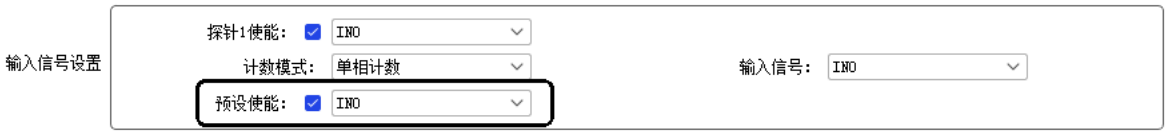


启用探针后，通过 LS_TouchProbe 功能块指令读取计数器轴的探针位置。

11.5.4 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。

11.5.5 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。

11.6 计数器轴指令应用

11.6.1 概述

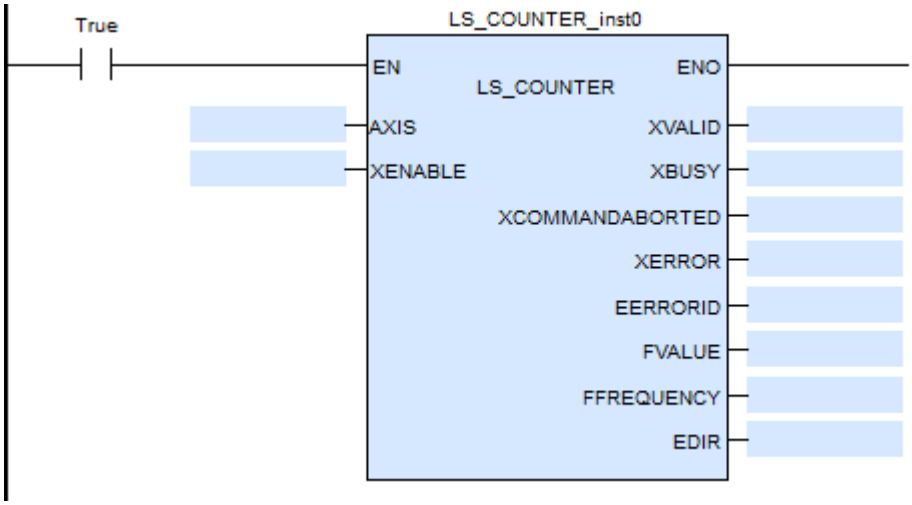
LeadStudio 软件中设置计数器轴后，配合功能块指令，可实现轴位置计数/速度测量、轴位置预置、轴位置锁存和比较功能。

11.6.2 轴位置计数/速度测量指令

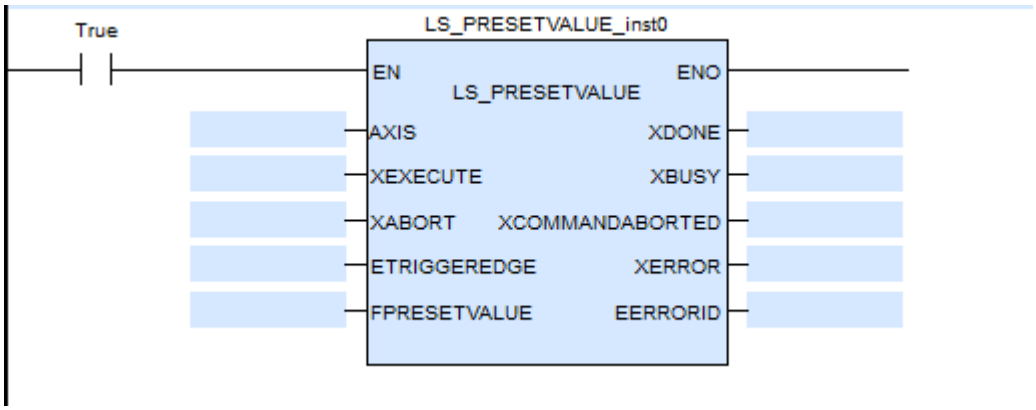
使用 LS_Counter 指令，可对计数器轴的位置计数和速度测量。

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。



11.6.3 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。

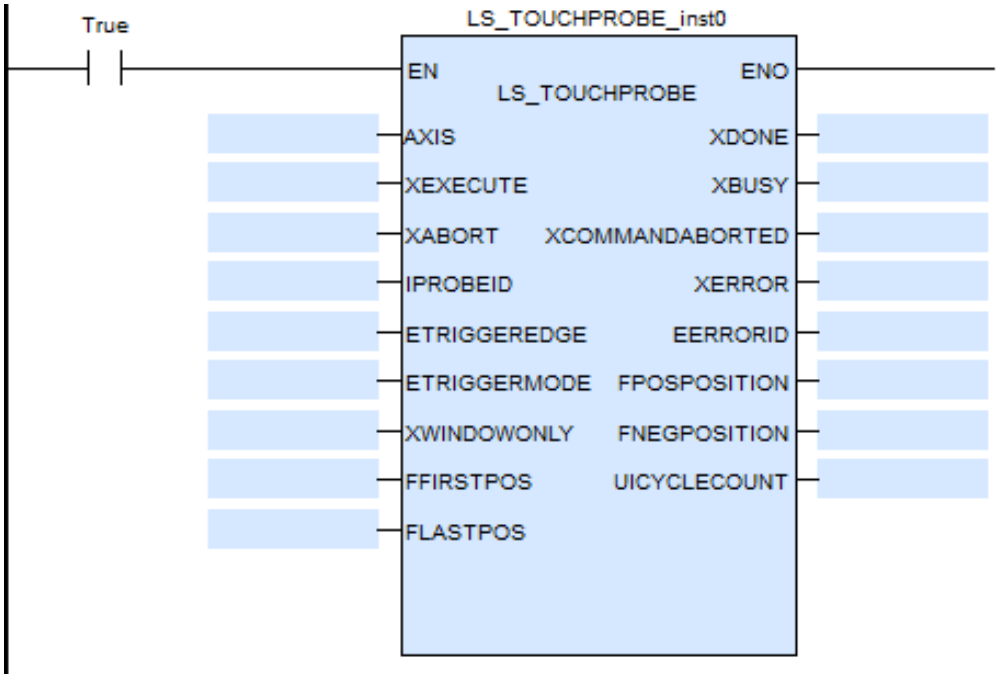
预置条件 EtriggerEdge 可选择指令上升沿触发或外部 X 输入触发。

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。

11.6.4 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。

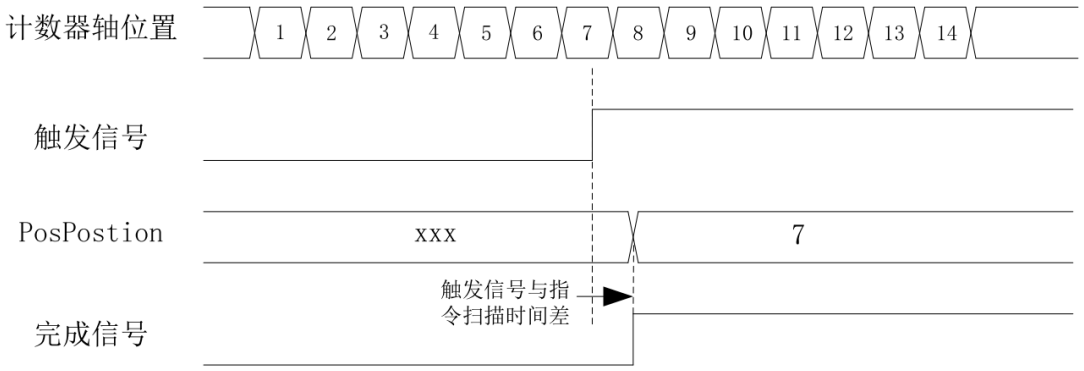
iProbeID	定义
0	表示使用探针 1

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

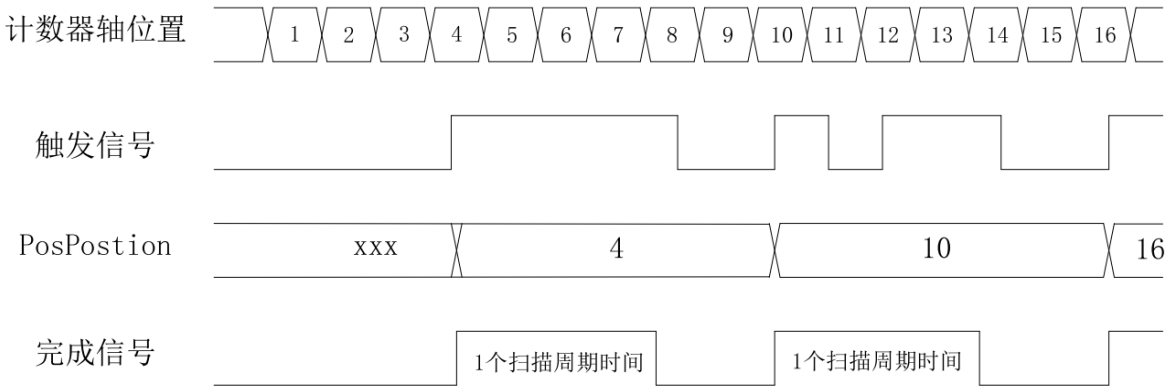
指令中 EtriggerMode 参数可设置单次触发和连续触发模式。

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。



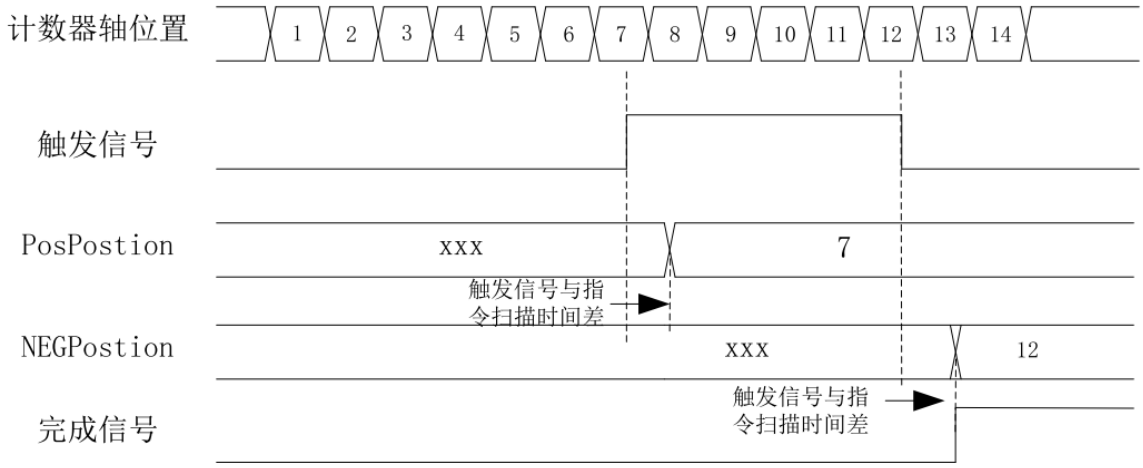
上升沿单次触发模式

- 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。

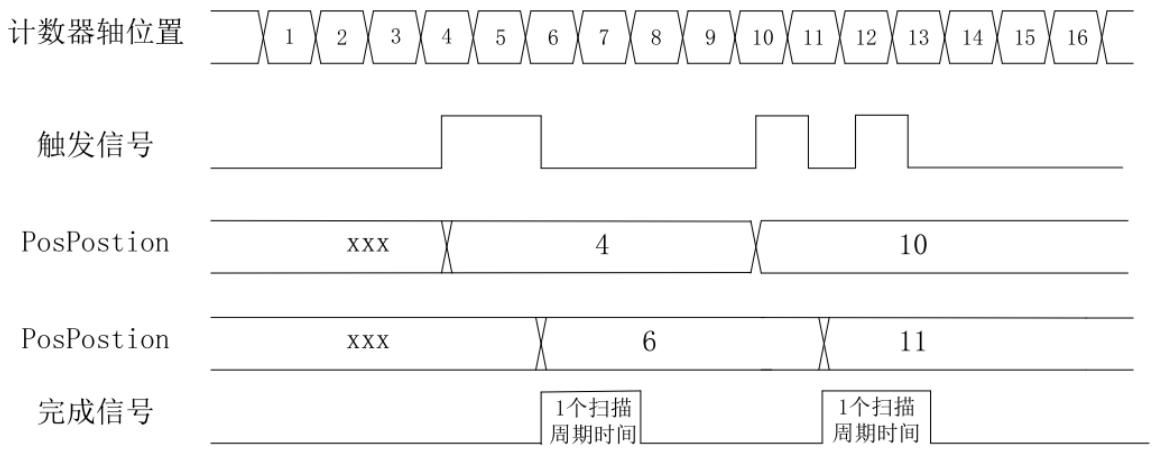


上升沿连续触发模式

- 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

11.7 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。

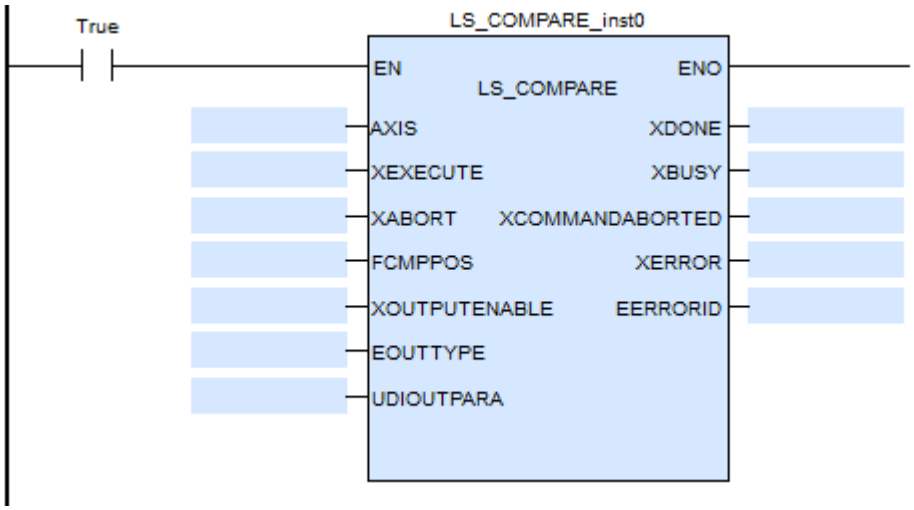
  当指令执行比较相等时，直接通过硬件电路控制设定的输出端子为 ON，持续输出宽度后输出变为 OFF。

11.7.1 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。

LS_Compare

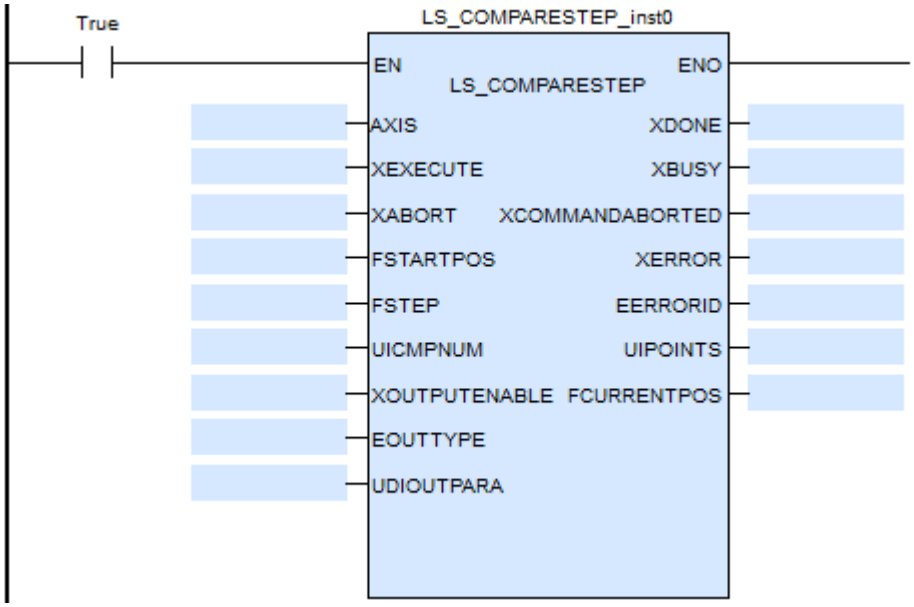
实现计数器轴与单个位置比较，指令能流有效时，计数器轴位置达到比较位置后，输出完成信号。



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。

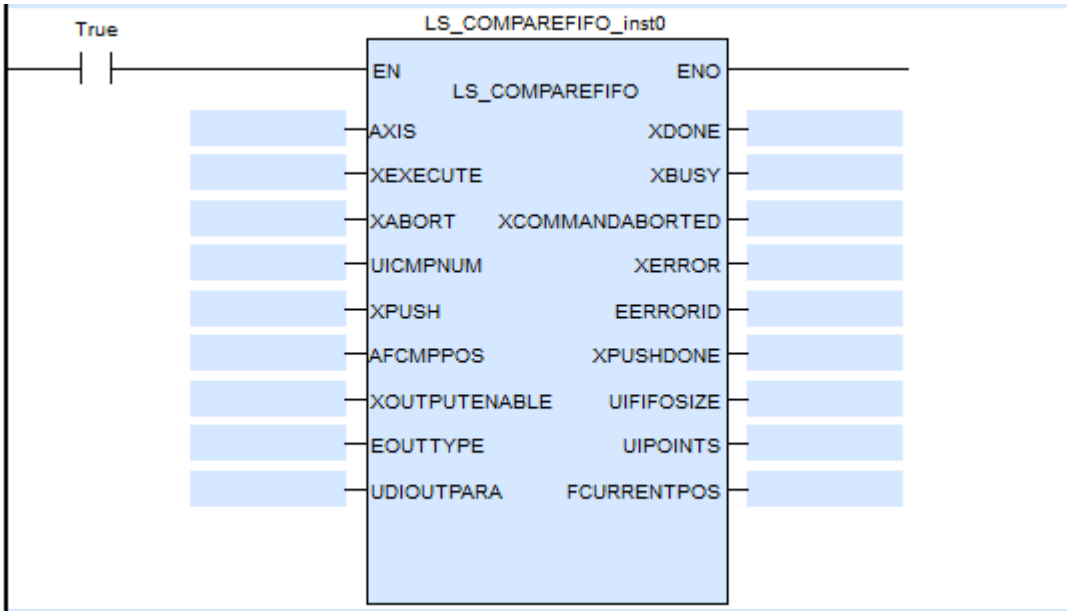
- fStep 大于 0 时，每次比较相等时，比较位置增加 fStep 间距，当 uiPoints 等于 uiCmpNum，当前比较位置即为最后一个比较位置。
- fStep 小于 0 时，每次比较相等时，比较位置减小 Step 间距，当 uiPoints 等于 uiCmpNum，当前比较位置即为最后一个比较位置。
- 输出参数 uiPoints 指示已完成比较相等的个数。



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。

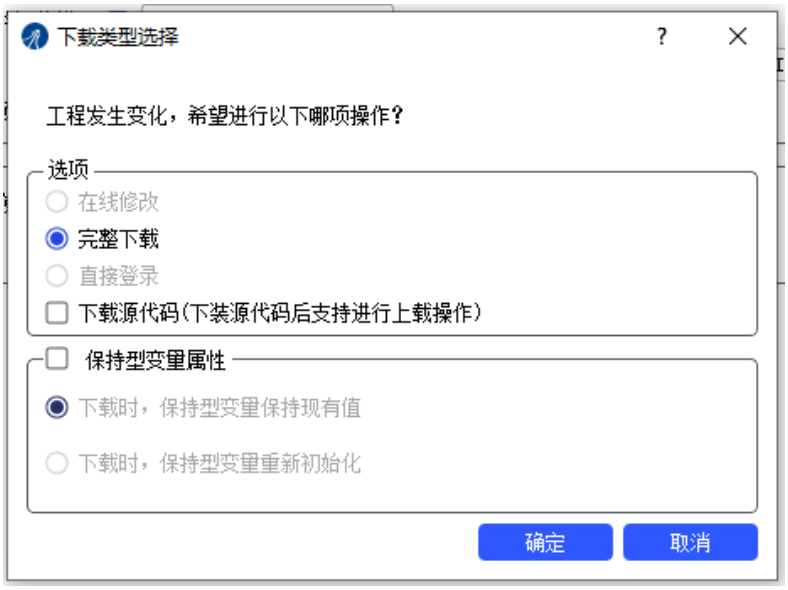
- 输出时间是指当比较一致后输出持续的时间，单位为 us，需要的输入参数为设置比较模式，将比较点压入，设置输出时间即可；
- 输出电平则满足比较一致后，持续输出，需要的输入参数为设置比较模式，将比较点和比较逻辑压入即可。
- 输出脉冲个数是指当比较一致后，持续计数器的脉冲计数个数后，关闭输出。比较点为指针类型，需指定比较点数 fCmpPos，注意比较点和比较逻辑电平数组长度必须大于比较点数。xPush 为动态压点，即在比较过程中支持压入比较点，且 FIFO 最大长度为 1000，若超过 1000 则会等待待比较点+FIFO 点≤1000 才会压入 FIFO 中。
- 比较功能块使用前，必须在对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。

12 运行调试

12.1 在线操作

12.1.1 登录 PLC

1) 鼠标点击工具栏中的快捷键“ ”编译 PLC 程序，再点击下载按钮“ ”，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)

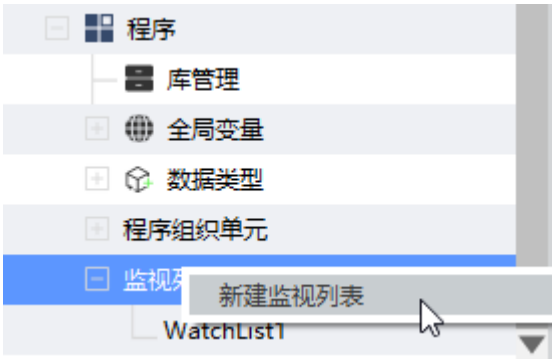


2) 点击工具栏中“ ” ----- “ ”，即可登录并运行 PLC，如下图所示：

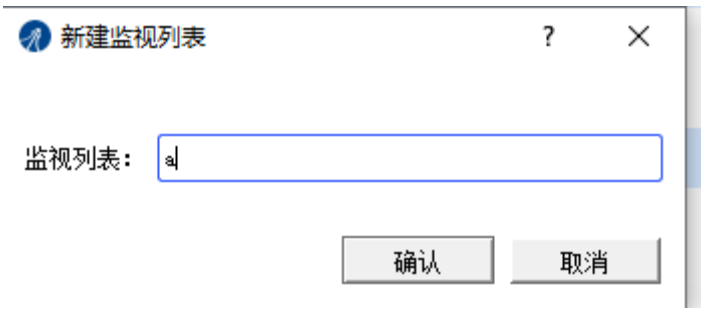
若点击“ ”前，PLC 没有下载程序或程序发生变动，则登录时会弹出上述步骤的下载类型选择框，用户需要下载后才可登录 PLC

12.1.2 添加变量监控

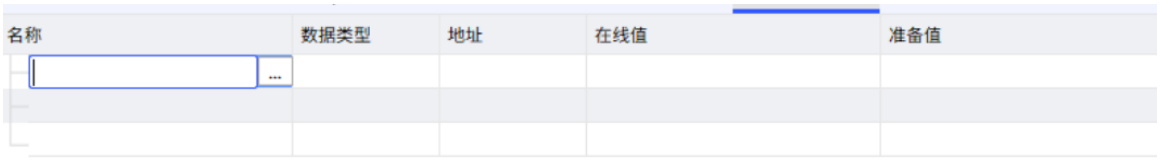
1) 右击监视列表-----新建监视列表，如下图所示：



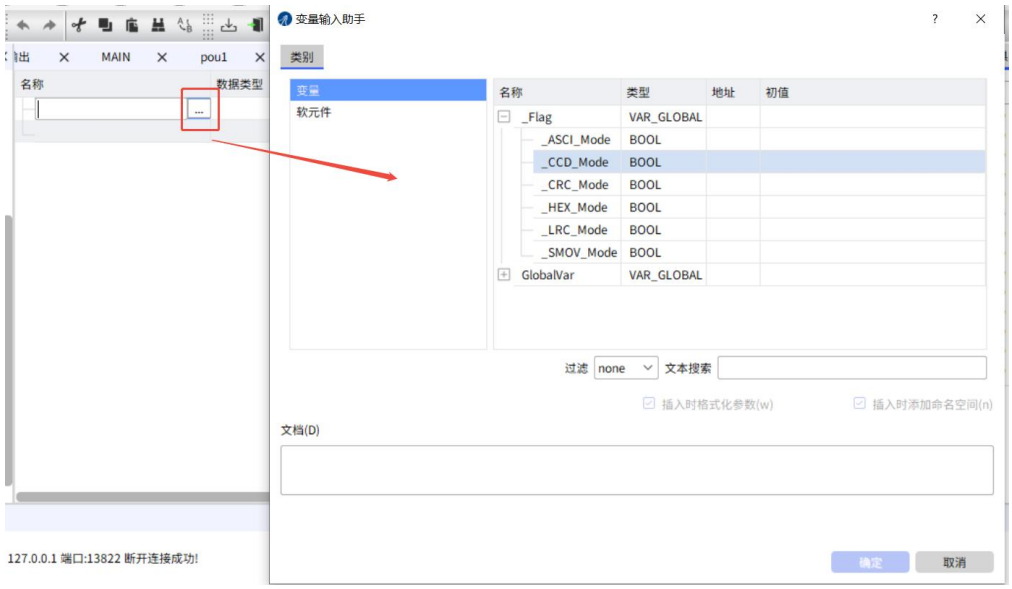
2) 给新建的监视列表设置名字



3) 键盘输入监控的变量或软元件名称



若变量名称过长不便键盘输入，也可通过输入助手完成变量添加



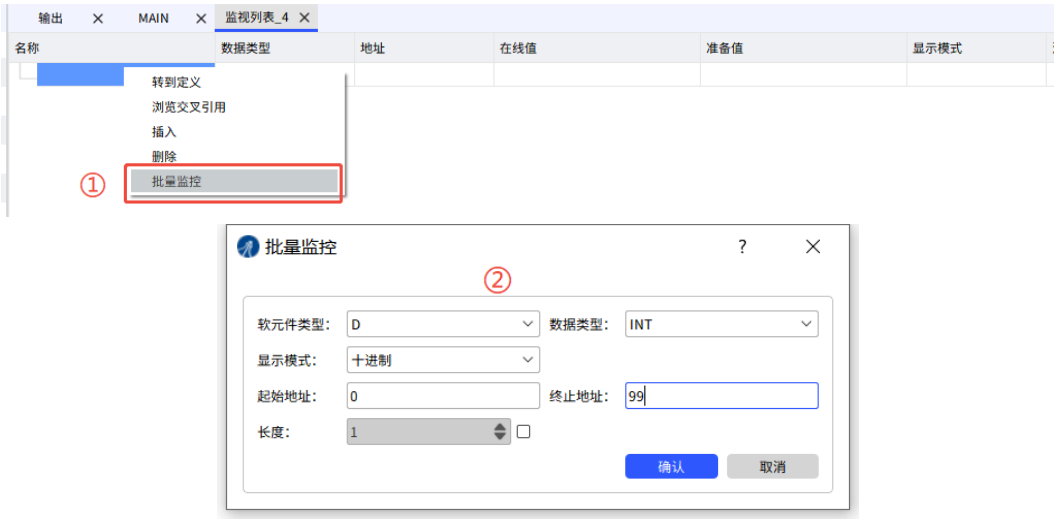
批量监控软元件

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；

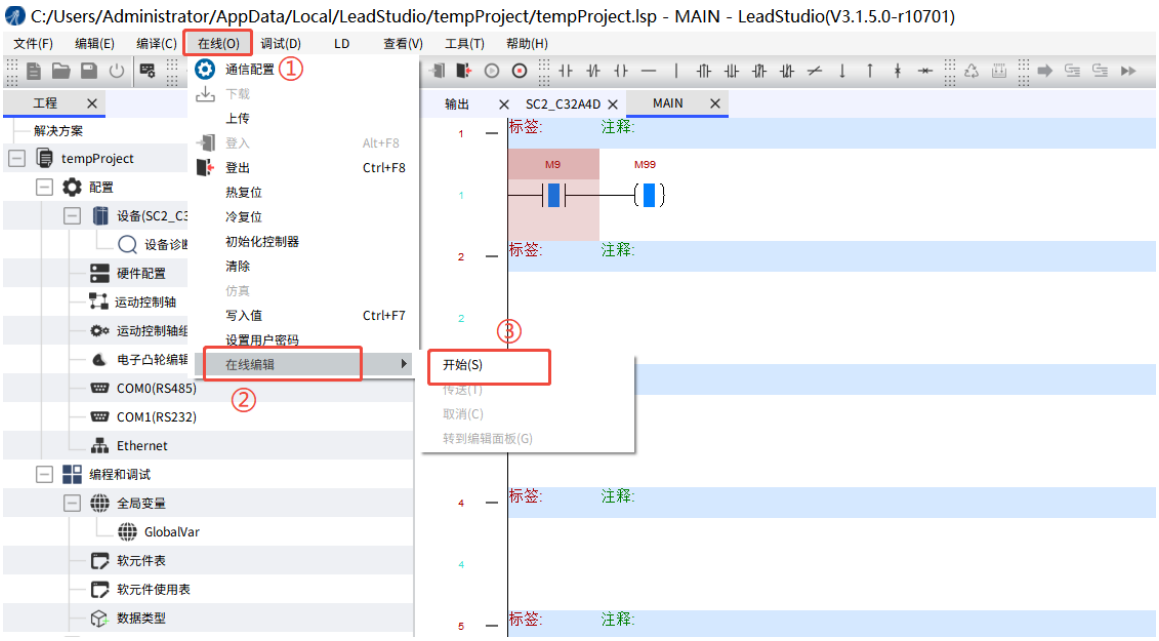
②选择监控的软元件类型及范围

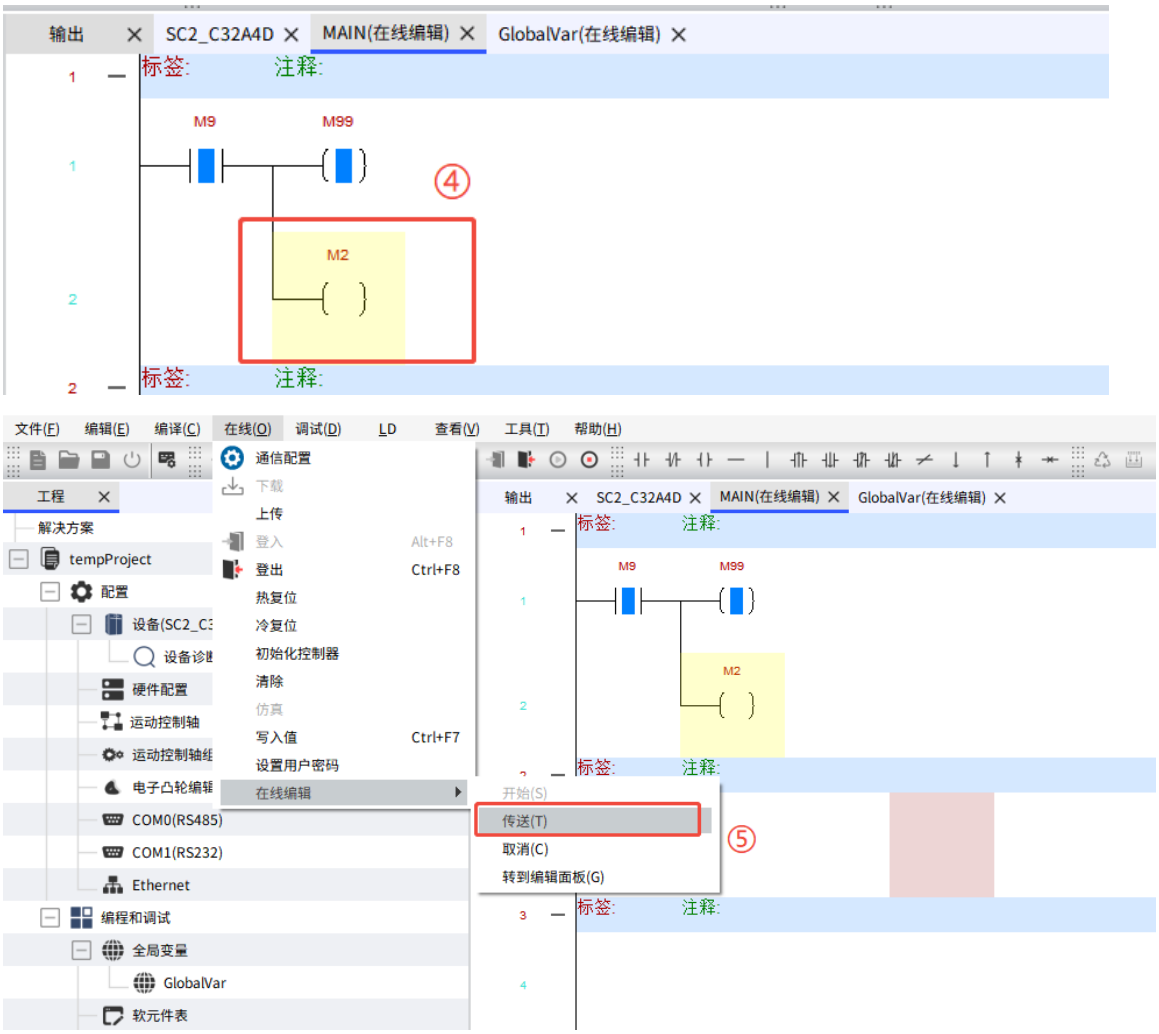


12.1.3 在线编辑

LeadStudioV3.1 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

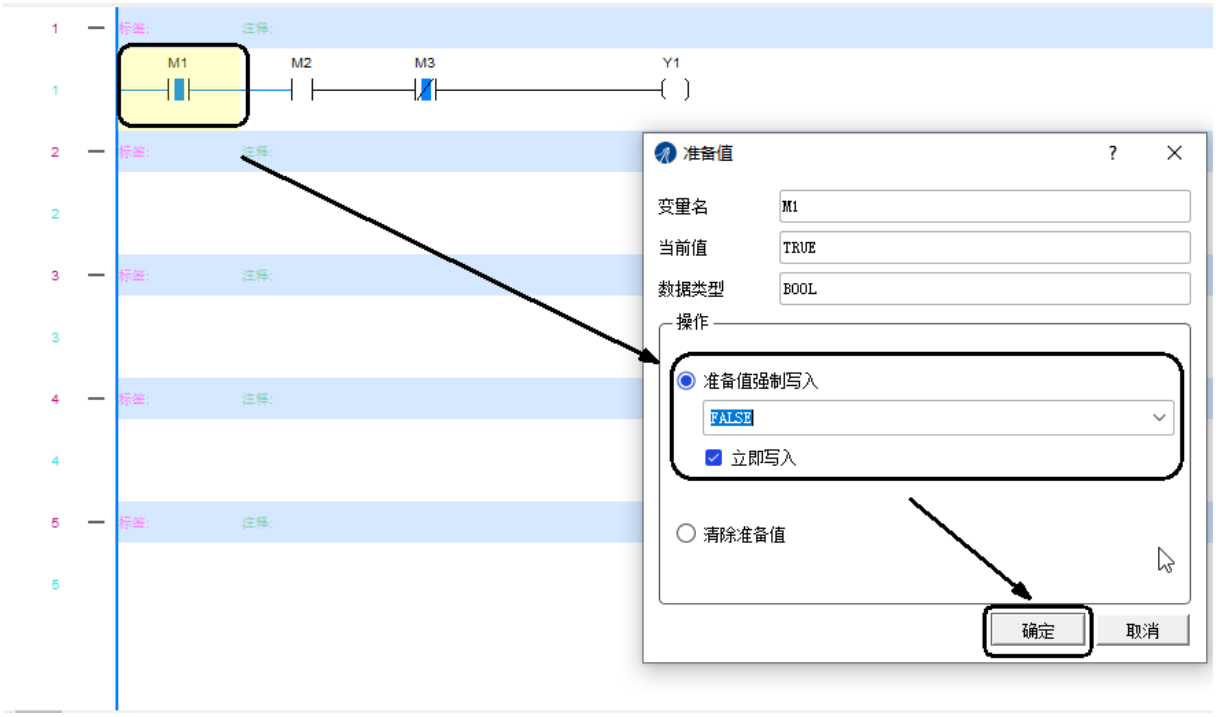




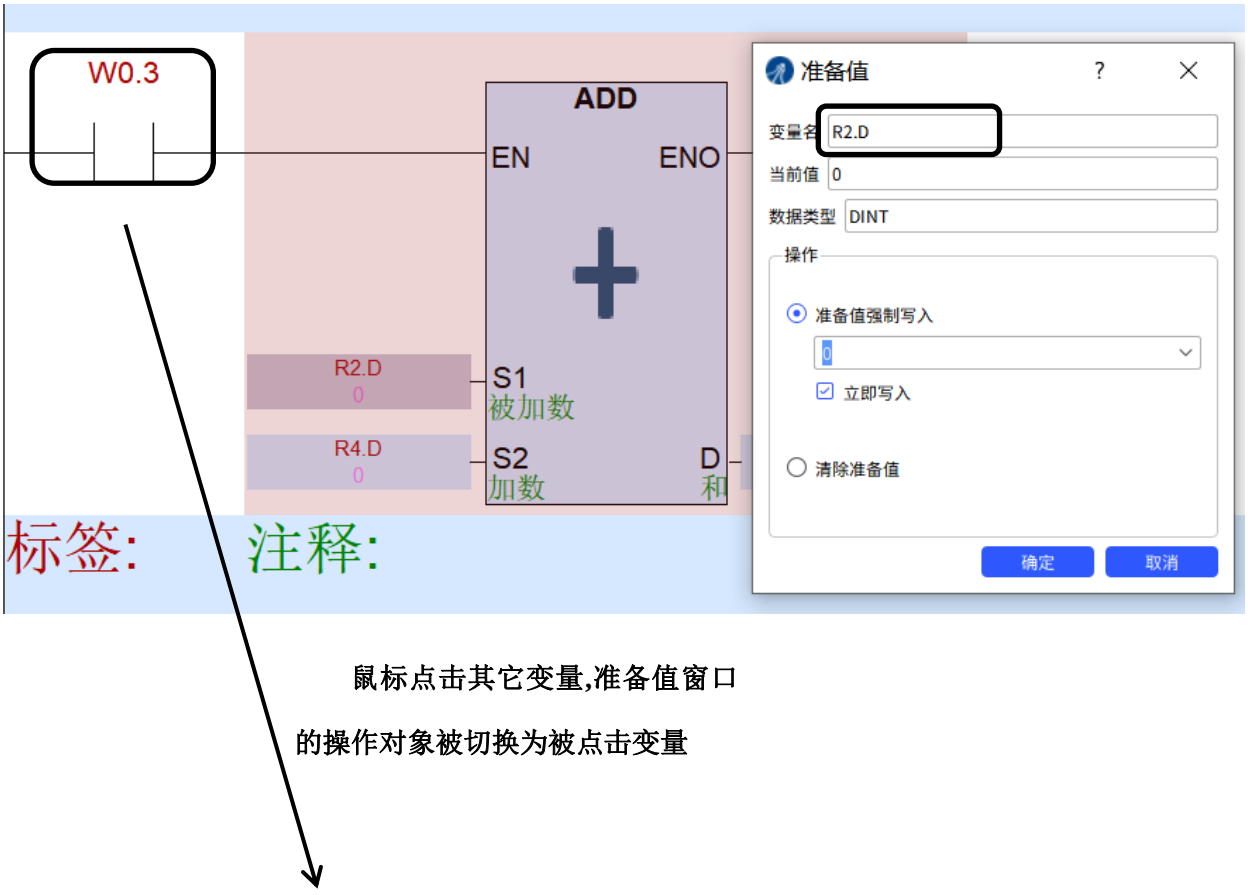
说明：只有在程序组织单元页面才可支持在线编辑功能，若在其它页面打开在线编辑则会弹窗提示“编辑器无法使用在线编辑”，如下所示：

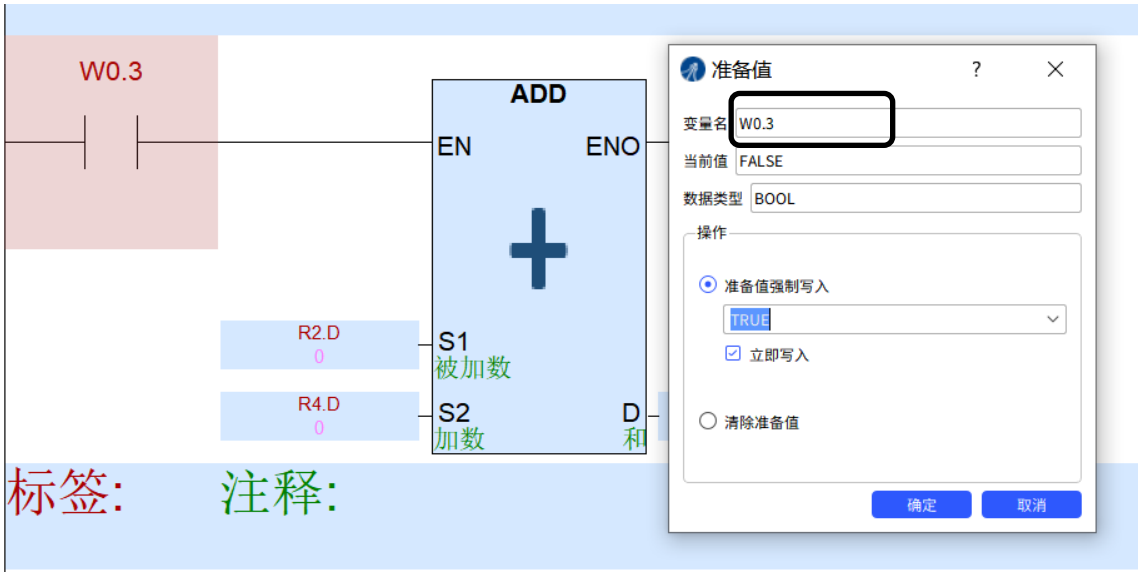
12.2 在线写入值

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。



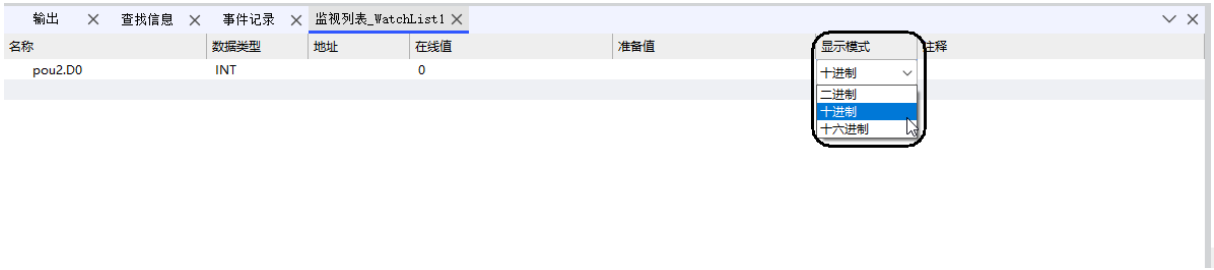
在准备值弹窗打开时，通过鼠标点击其它变量可快速切换准备值弹窗的操作对象，如下所示：





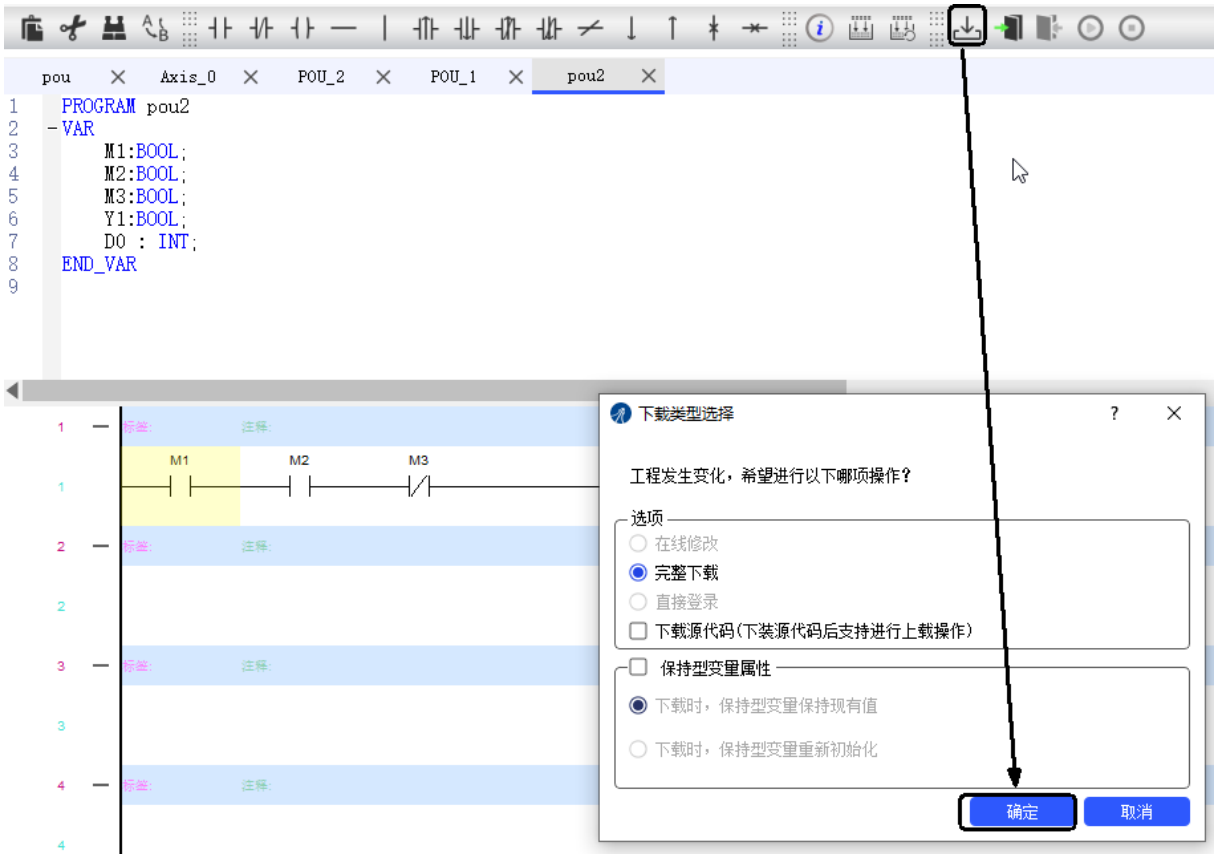
12.3 切换显示进制

监视列表的“显示模式”一列可以直接切换进制。

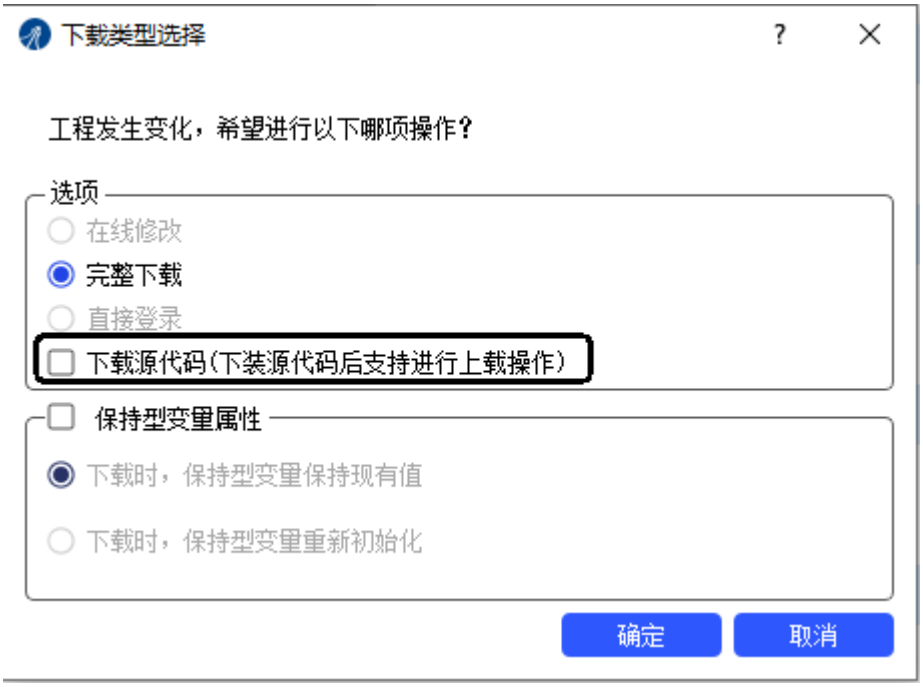


12.4 下载操作

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

12.5 复位功能（复位、冷复位）

点击在线-----复位和冷复位

12.6 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。

说明：快照值跟随工程保存。

12.6.1 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			☑

12.6.2 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：

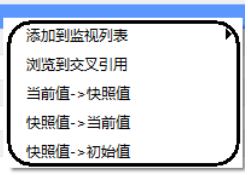
1）当前值->快照值：将变量的当前值同步到快照值中

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

12.6.3 快照值导入导出

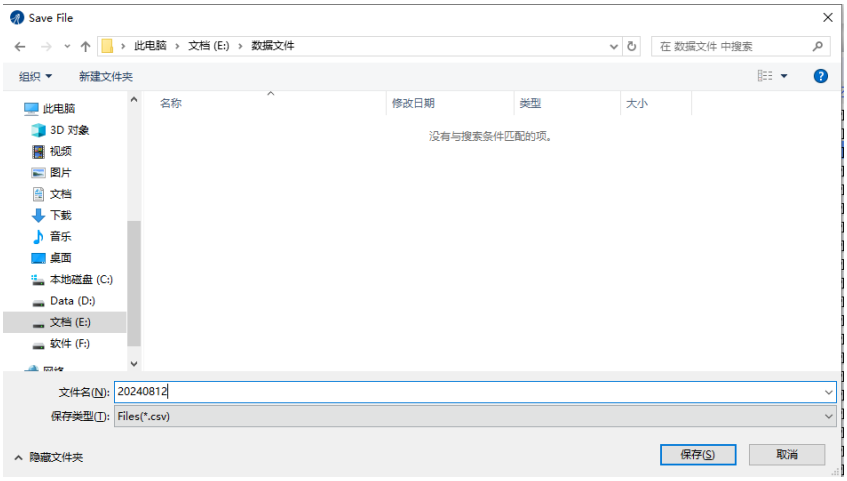
快照值支持导入导出，便于用户进行灵活的数据操作，导出快照的方式如下：

1）点击变量表右侧的  图标

GVL_HMI

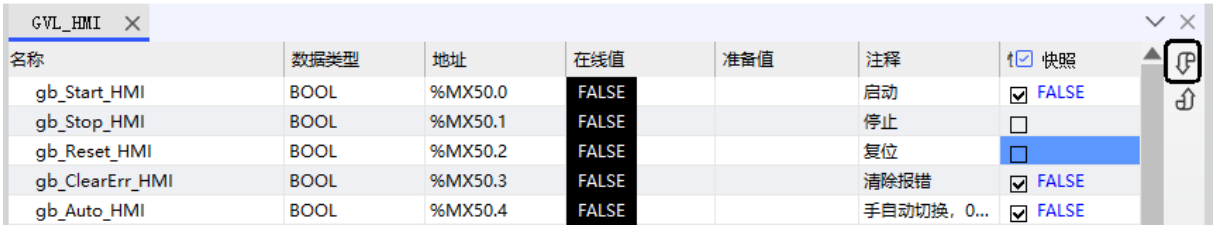
名称	数据类型	地址	在线值	准备值	注释	快照
gb_Start_HMI	BOOL	%MX50.0	FALSE		启动	☑ FALSE
gb_Stop_HMI	BOOL	%MX50.1	FALSE		停止	☐
gb_Reset_HMI	BOOL	%MX50.2	FALSE		复位	☐
gb_ClearErr_HMI	BOOL	%MX50.3	FALSE		清除报错	☑ FALSE
gb_Auto_HMI	BOOL	%MX50.4	FALSE		手自动切换，0...	☑ FALSE

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。

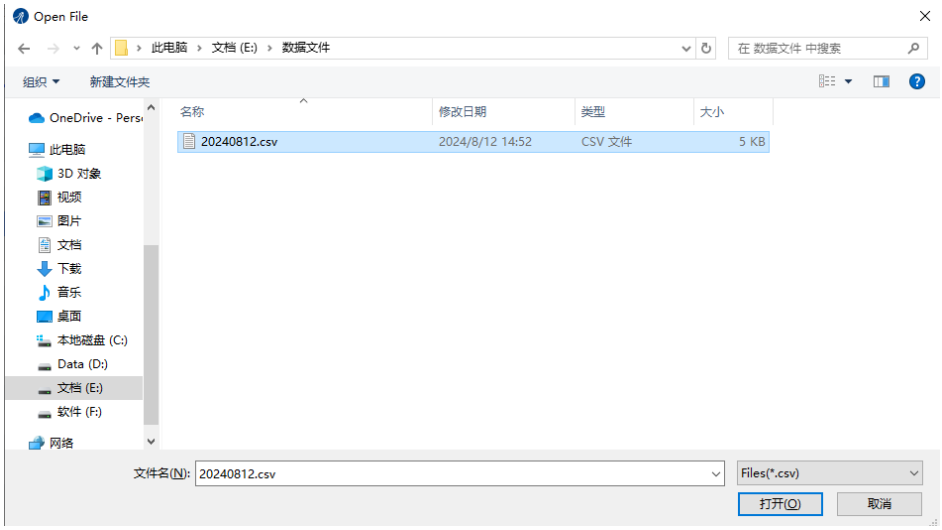


导入变量快照的方式如下：

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。

12.7 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。

离线与在线比较

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。

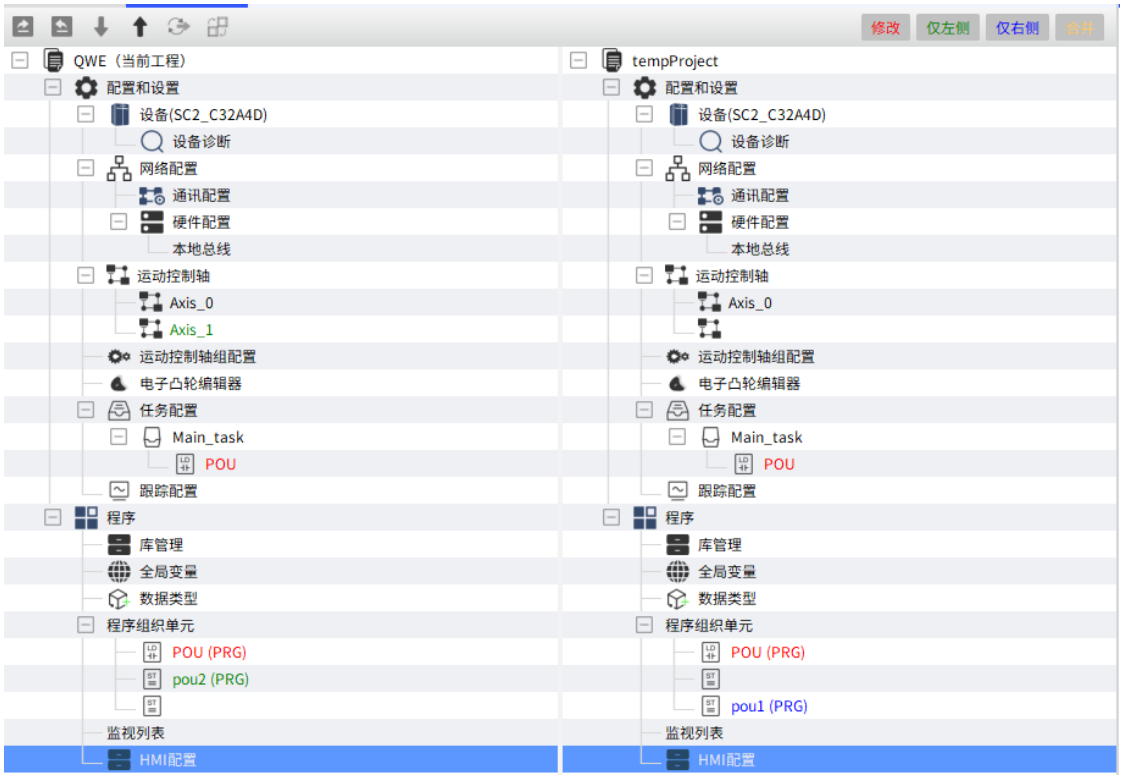
②登录状态下无法进行工程比较，且在工程比较期间限制用户执行登录PLC和编辑工程树对象。

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： V3.1及以上

12.7.1 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。

12.7.2 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”；

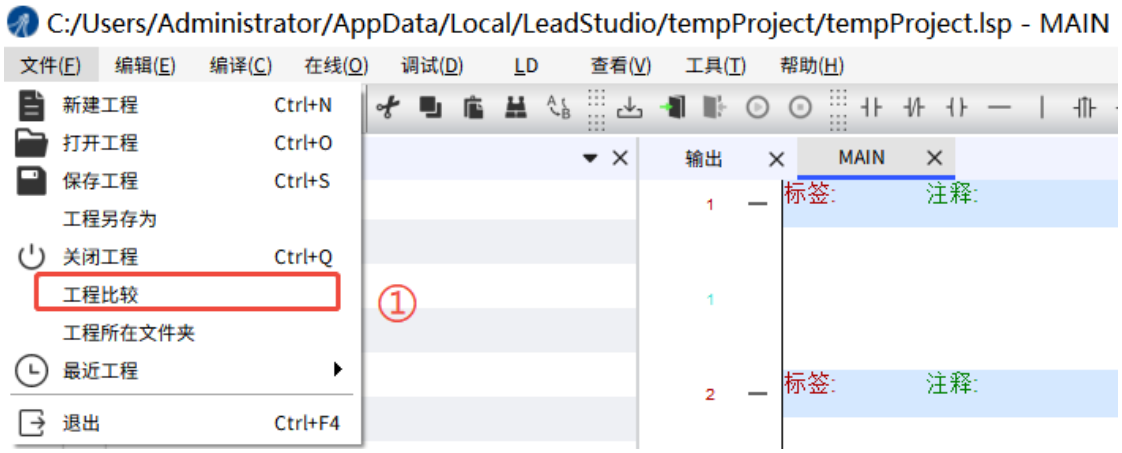
②选择对比工程类型（PLC内部程序或电脑本地程序）；

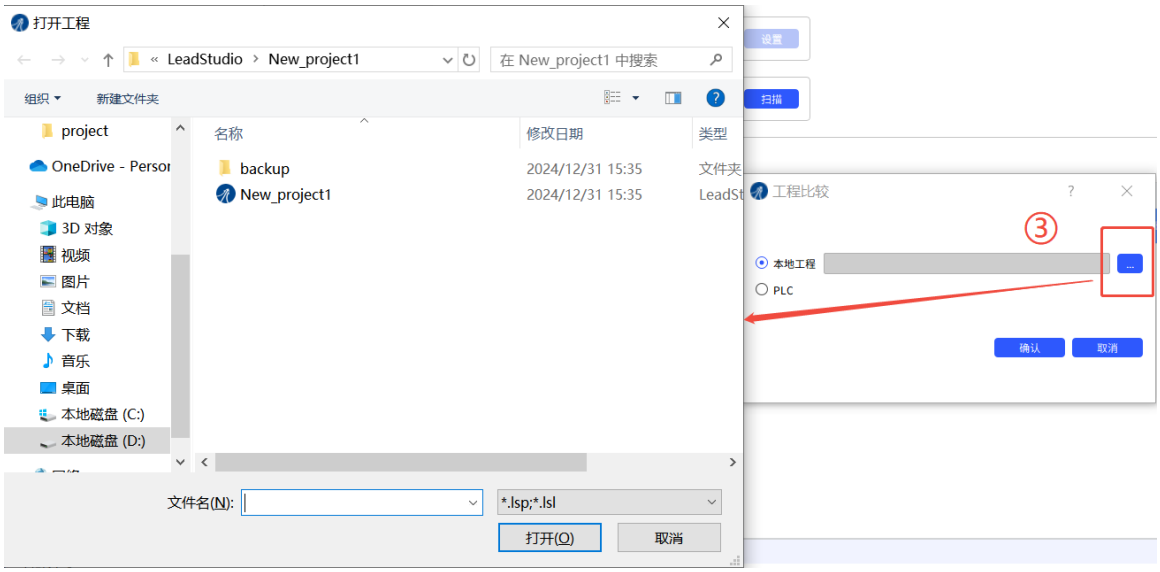
③若选择对比本地工程，需选择本地文件路径（选择对比在线PLC则需确保电脑与PLC已连接）；

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

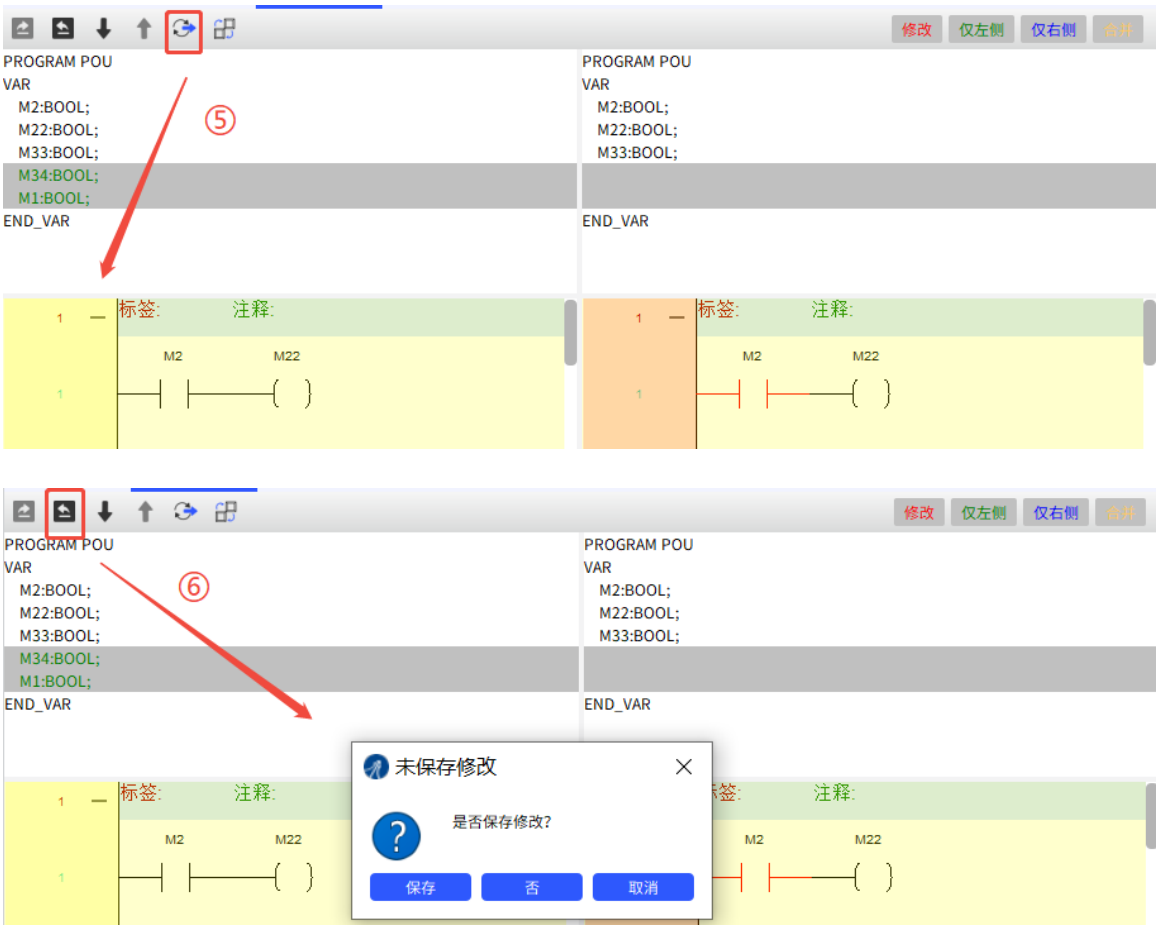
⑤在详细对比页面选中差异处，单击  进行行同步，或单击  进行块同步，以行同步为例。该差异处同步后以黄色色块标识。

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





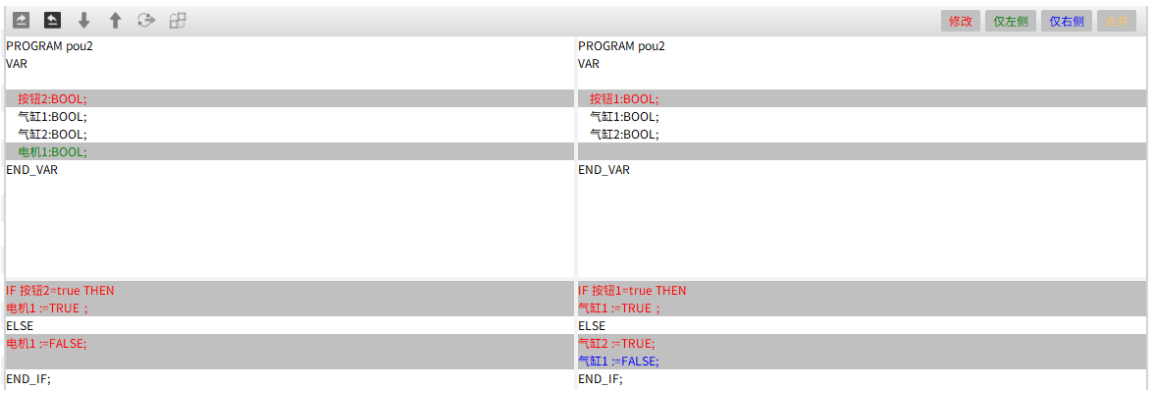

tempProject（当前工程）
配置和设置
设备(SC2_C32A4D)
设备诊断
网络配置
通讯配置
硬件配置
本地总线
运动控制轴
Axis_0
Axis_1
运动控制轴组配置
电子凸轮编辑器
任务配置
④
Main_task
POU
跟踪配置
程序
库管理
全局变量
数据类型
程序组织单元
POU (PRG)
pou2 (PRG)
监视列表
HMI配置
tempProject
Axis_0
pou1 (PRG)
PROGRAM POU
VAR
M2:BOOL;
M22:BOOL;
M33:BOOL;
M34:BOOL;
M1:BOOL;
END_VAR
标签:
注释:
M1
M22
M2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



12.7.3 详细比较与同步说明

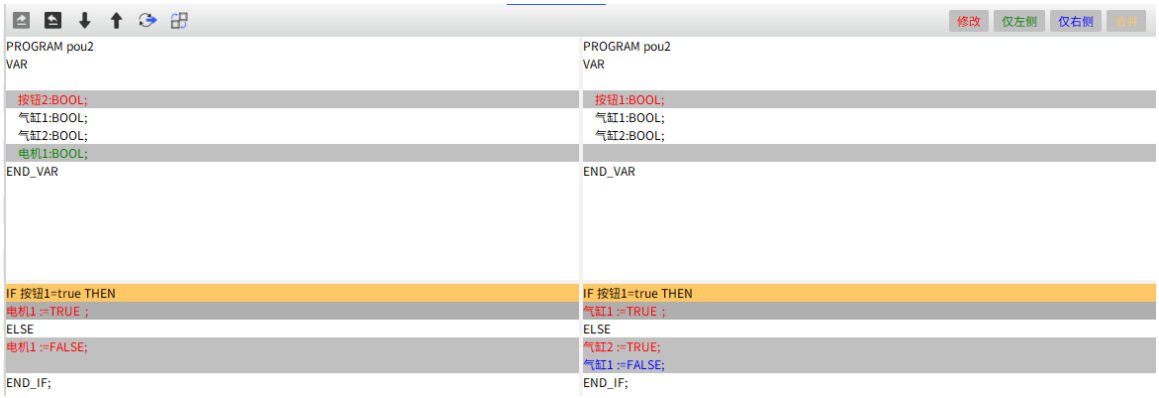
1. ST程序

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：

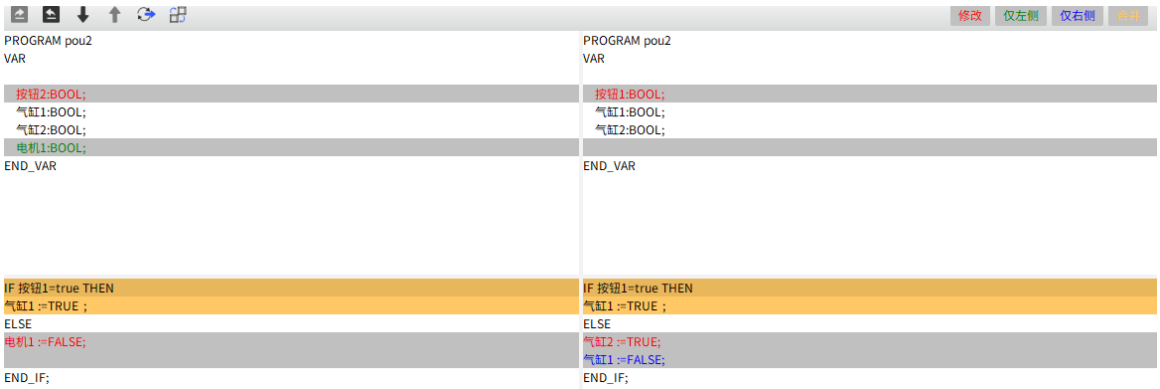


ST程序的行同步是将右侧比较工程文本光标所在行的内容合并到左侧当前工程中，块同步则是同步光标当前行以及前后连续差异行的内容，合并后同步部分内容将以黄色色块（底纹）标识。

ST行同步示例如下：

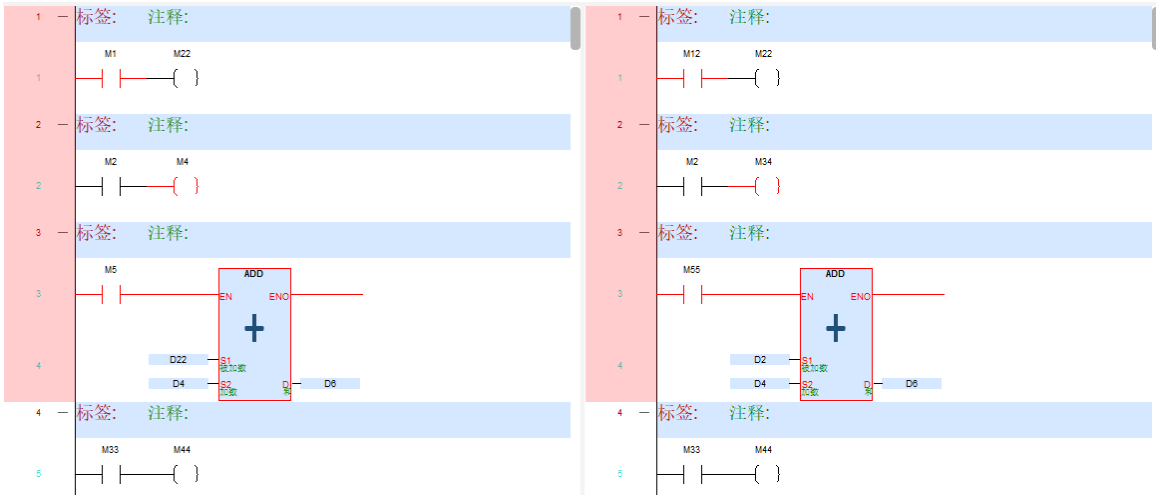


ST块同步示例如下：



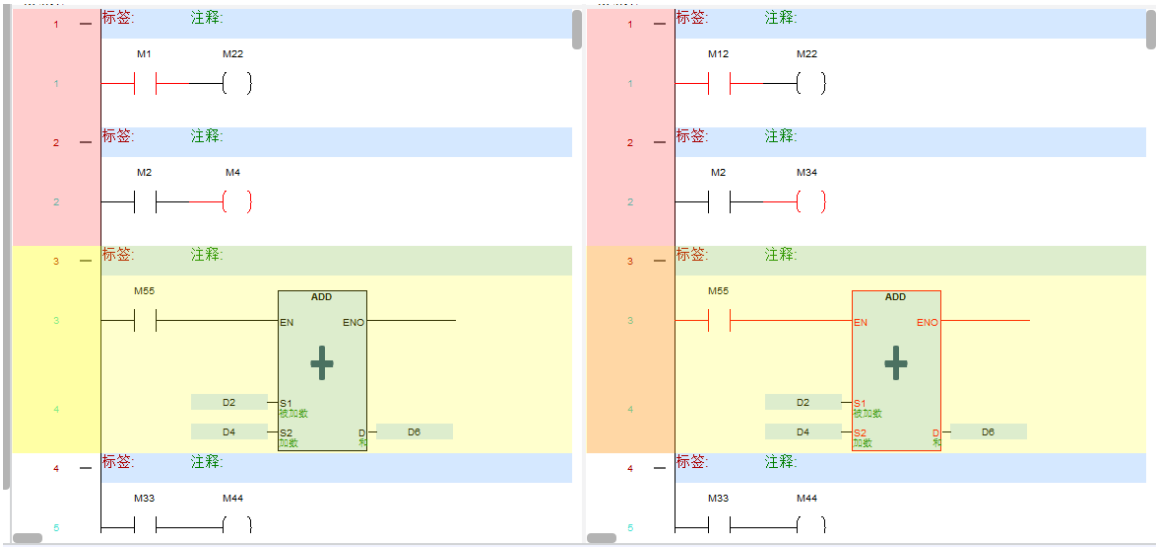
2.LD程序

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：

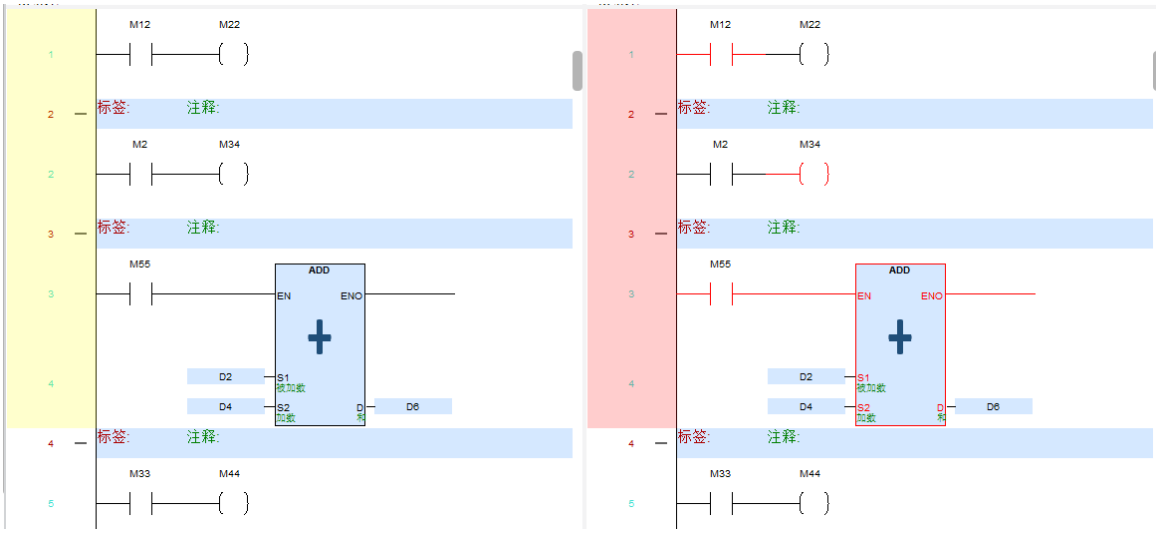


LD程序的行同步是对当前选择存在差异的网络进行内容的同步；块同步则是对当前选择存在差异的网络前后连续存在差异的网络进行同步操作。同步完成后，会合并比较工程相应的内容到当前工程中，网络同步内容左侧将以黄色色块标识。

LD行同步示例如下：



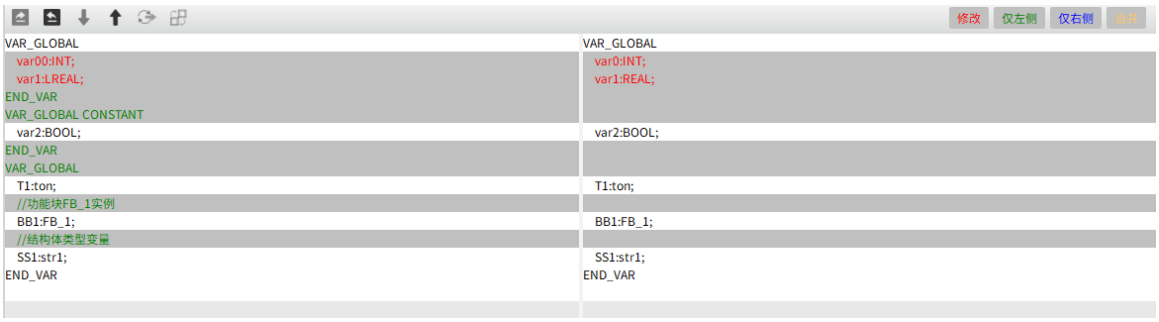
LD块同步示例如下：



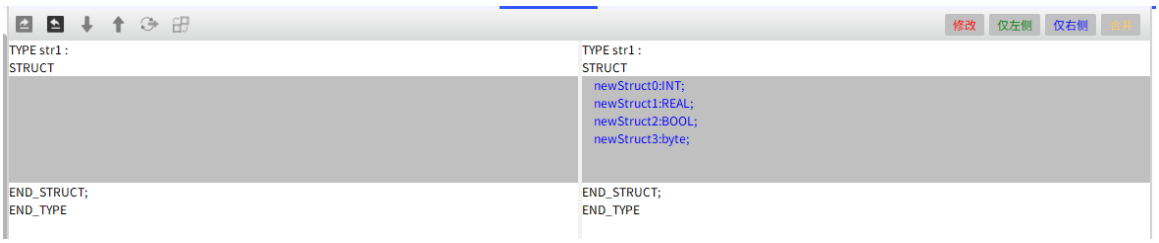
3. 变量表、数据类型

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

12.8 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

13 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

13.1 注意事项

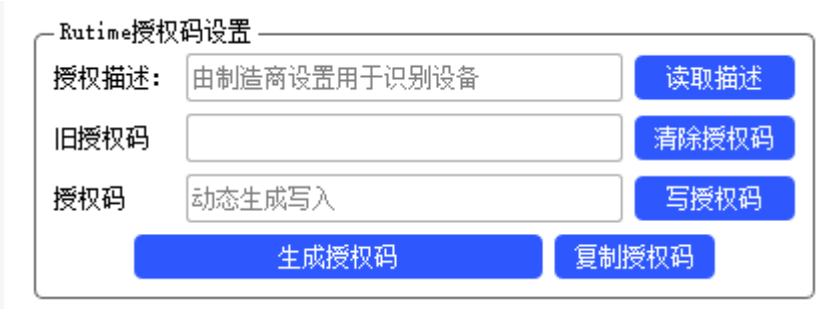
1）对工程设置授权码前建议对工程进行备份，防止遗忘授权码导致工程无法找回；

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！”；

13.2 授权码功能使用介绍

13.2.1 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）

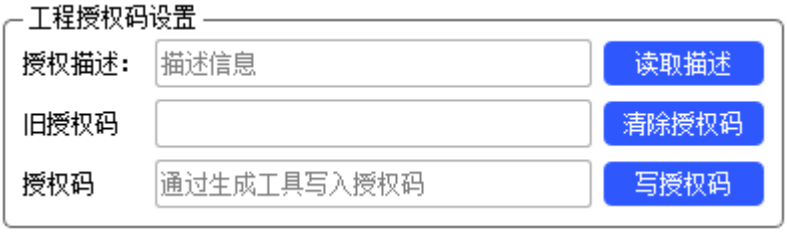
2）旧授权码：10 位数字，清除授权码、写授权码时使用，如果 Runtime 中存在授权码,要输入正确的旧授权码后才支持执行写授权码、清除授权码操作；（简单来说，这是 PLC 旧密码）

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；

13.2.2 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）

13.2.3 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”；

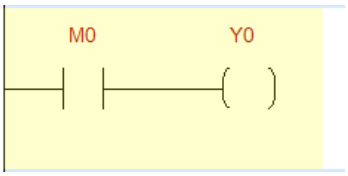
4）工程授权码只能通过清除授权码操作清除；

14 增量粘贴

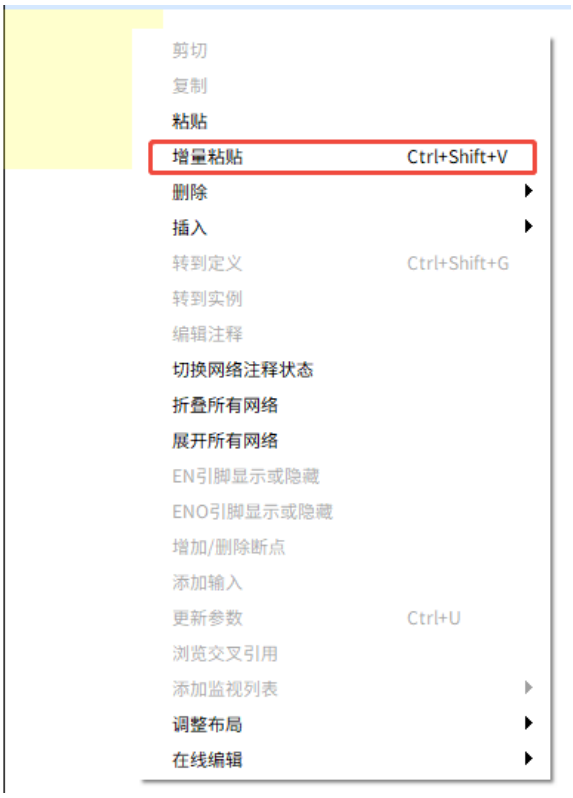
增量粘贴功能可在梯形图编程中对复制的元件进行多次连续粘贴操作；同时在粘贴的过程中可以对软元件号或数组下标进行指定操作。

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”）。

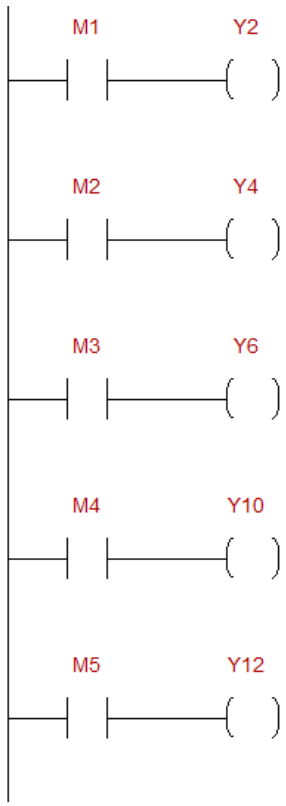


②选中需要增量粘贴的位置（选中位置为粘贴的第一行），在右键菜单中选择“增量粘贴”或者使用快捷键 Ctrl+Shift+V 键。



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