HollySys

www.hollysys.com

Version 4.1.1

AutoThink

版权声明

本手册内容，包括文字、图表、标志、标识、商标、产品型号、软件程序、版面设计及其它内容等，均受《中华人民共和国著作权法》、《中华人民共和国商标法》、《中华人民共和国专利法》及与之适用的国际公约中有关著作权、商标权、专利权或其他财产所有权法律的保护，为北京和利时智能技术有限公司专属所有或持有。

由于本手册中所描述的设备有多种使用方法，用户以及设备使用责任人必须保证每种方法的可容许性。对由使用或错误使用这些设备造成的任何直接或间接损失，北京和利时智能技术有限公司将不负法律责任。

由于实际应用时的不确定因素，北京和利时智能技术有限公司不承担直接使用本手册中提供的数据的责任。

本手册仅供商业用户阅读，在未得到北京和利时智能技术有限公司书面授权的情况下，无论出于何种目的和原因，不得以任何形式（包括电子、机械或其它形式）传播或复制本手册的任何内容。违者我公司将依法追究其相关责任。

已核对本手册中的内容、图表与所述硬件设备相符，但误差难以避免，并不能保证完全一致。同时，会定期对手册的内容、图表进行检查、修改和维护，恕不另行通知。

HollySys、和利时、的字样和徽标均为北京和利时智能技术有限公司的商标或注册商标。

手册中涉及到的其他商标或注册商标属于它们各自的拥有者。

北京和利时智能技术有限公司版权所有。

地址：北京经济技术开发区地盛中路 2 号院

邮编：100176

商务电话：010-5898 1588

产品咨询热线：4008-111-999

传真：010-5898 1558

网址：http://www.hollysys.com/

Email：PLC@hollysys.com

新浪微博：http://weibo.com/hollysysplc

目录

北京和利时智能技术有限公司版权所有

北京和利时智能技术有限公司版权所有

北京和利时智能技术有限公司版权所有

北京和利时智能技术有限公司版权所有

北京和利时智能技术有限公司版权所有

北京和利时智能技术有限公司版权所有

北京和利时智能技术有限公司版权所有

北京和利时智能技术有限公司版权所有

北京和利时智能技术有限公司版权所有

第1章　关于本书

1.1 文档更新

版本	说明	更新日期
V1.0	新建	2020.12.31
V1.1	支持 CFC/SFC 语言组态 支持 LK226C/LK226CT1/LK616C/LK716C/LK412C/LK511C/LK432C/LK246C/LK239C 模块	2021.05.31
V1.2	支持 IP 地址扫描 支持 OPCUA 模块 支持导入示例工程 支持 LK238C/LK620C/LK410/LK411/LK412/LK430/LK432/LK411/LKL442/LK510/LK511/LK512/LK610/LK616/LK620/LK631/LK710/LK716/LK720 模块	2021.12.31
V1.3	支持通道地址修改 LD 语言支持拖拽移动	2022.03.31

1.2 文档用途

本手册帮助使用 AutoThink V4.1.1 软件的用户，了解和学习软件功能使用、工程组态方法以及常见问题处理，并指导用户正确、有效的使用该软件。

1.3 阅读对象

本手册适用于以下人员：

- 负责系统工程实施的工程人员；
- 负责系统维护的技术人员；
- 安装人员。

1.4 重要信息

在本手册中使用以下符号：

危险：该符号表示存在造成人身伤亡的潜在威胁。


电击：该符号表示存在造成电击的潜在威胁。


警告：该符号表示存在造成设备或环境损坏的潜在威胁。


提示：该符号表示有用的技巧或建议，以便更高效地使用产品。

SEE ALSO 该符号表示用于参考的信息。

1.5 产品文档目录


AutoThink V4.1.1 用户手册_工程组态


LKC 系列可编程控制器硬件手册

1.6 在线帮助手册

在线帮助为使用本软件的用户提供详细的内容支持。

通过以下两种方式打开帮助手册：

- 单击菜单栏“帮助”—“目录与索引”。
- 鼠标在软件界面，按下 **F1** 快捷键。

打开在线帮助时，推荐使用 Google Chrome 浏览器打开。使用 IE 浏览器打开时，请确保版本在 IE11 及以上版本。

1.7 缩略语

缩写	全称	说明
FA-AT	FA-AutoThinkV4	FA-AutoThinkV4 编程软件的简称
I/O	Input/Output	输入/输出
DC	Direct Current	直流电源

FPGA	Field Programmable Gate Array	现场可编程门阵列
TCP/IP	Transmission Control Protocol/Internet Protocol	传输控制协议/因特网互联协议
Profibus	Process Field Bus	过程现场总线
AI	Analog Input	模拟量输入
AO	Analog Output	模拟量输出
DI	Digital Input	开关量输入
DO	Digital Output	开关量输出
TC	Thermocouple assembly	热电偶
HSC	High Speed Counting	高速计数
RTD	Resistance Temperature Detector	热电阻
RTC	Real-Time Clock	实时时钟
HMI	Human Machine Interface	人机界面

1.8 缩略语

缩写	全称	名称
I/O	Input/Output	输入/输出
DC	Direct Current	直流电源
FPGA	Field Programmable Gate Array	现场可编程门阵列
TCP/IP	Transmission Control Protocol/Internet Protocol	传输控制协议/因特网互联协议
Profibus	Process Field Bus	过程现场总线
AI	Analog Input	模拟量输入
AO	Analog Output	模拟量输出
DI	Digital Input	开关量输入
DO	Digital Output	开关量输出
TC	Thermocouple assembly	热电偶
RTD	Resistance Temperature Detector	热电阻
RTC	Real-Time Clock	实时时钟
HMI	Human Machine Interface	人机界面

第2章　软件概述

AutoThinkV4 是自主可控 PLC 控制系统的上位编程软件，是和利时公司自主研发的跨平台编程工具。

AutoThinkV4 主要具有以下功能及特点：

- 组态简单直观：采用树形结构进行硬件配置、任务配置、程序块组态。
- 组态指令丰富，支持多种数据类型。
- 支持用户自定义库。
- 具有在线调试等功能。
- 具有密码保护功能。
- 强大的 ST 语言编辑及智能联想功能。
- 用工程来管理一个自动化系统的硬件和软件。

第3章　软件安装

3.1　安装环境

下表为计算机配置信息：

推荐使用的计算机配置表

环境	类型	型号
硬件环境	显示器	彩色 CRT 或液晶屏
	输入输出	标准键盘、鼠标
	USB 接口	至少 1 个 USB 2.0 接口
	显卡	分辨率支持 1280×720
	CPU	Intel Pentium 2.4 GHz 以上
	内存	512 MB 以上
	硬盘	10 GB 以上
软件环境	操作系统	Windows 7 Professional 32 位 Windows 7 Professional 64 位 Windows 10 Professional 64 位 以上操作系统仅支持中文系统环境

工程中使用的计算机配置应与上述配置相当，或高于上述配置。

3.2　在 Windows 环境安装

3.2.1　安装

3.2.1.1　概述

工程组态前，需要将 AutoThinkV4 软件安装到工程师站。

以下安装示意以 FA-AutoThinkV4.1.1 版本为例进行说明。

3.2.1.2　要求

- Windows 操作系统，中文环境
- 准备 AutoThinkV4 安装文件

3.2.1.3 步骤

安装 AutoThinkV4 软件，请按以下步骤操作：

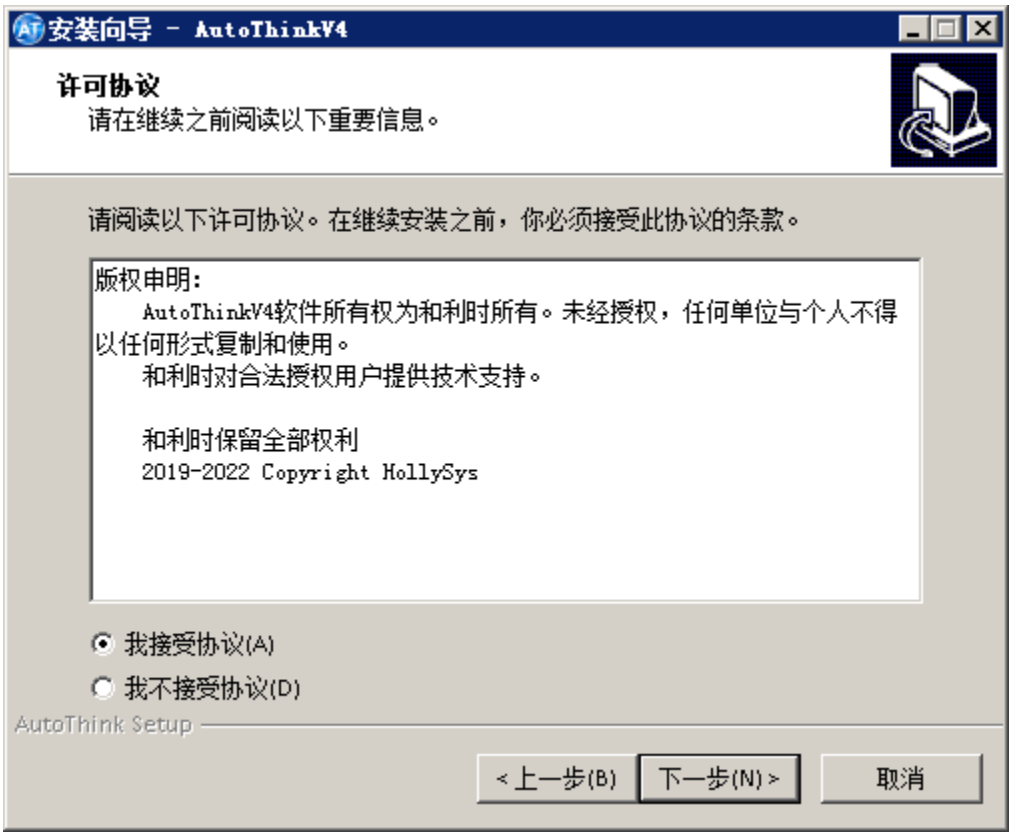
（1） 启动安装向导

单击**下一步(N)**。

安装向导

（2） 许可协议

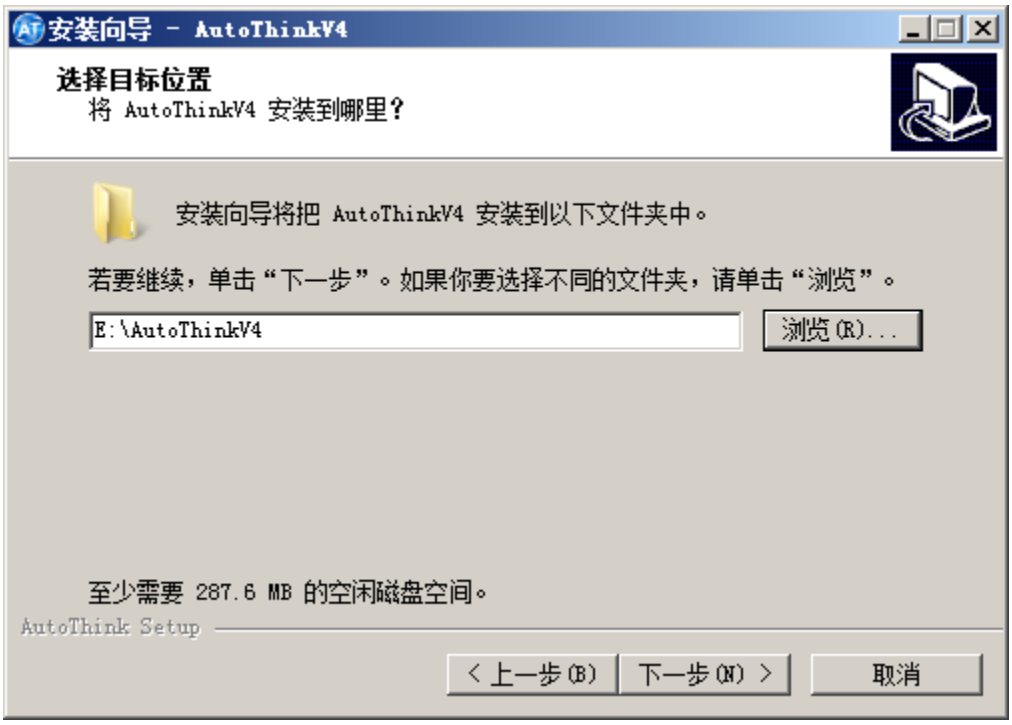
选择“我接受协议（A）”，单击**下一步(N)**。



许可协议

（3） 选择安装路径

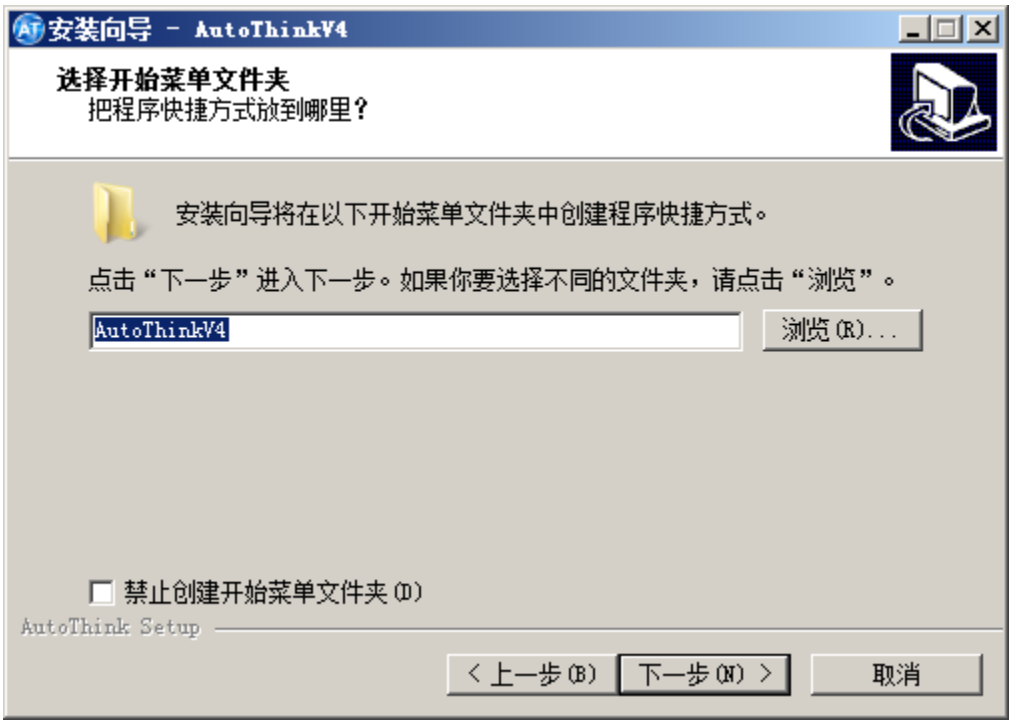
单击**浏览(R)…**选择安装路径，单击**下一步(N)**。



选择安装路径

（4） 选择开始菜单文件夹

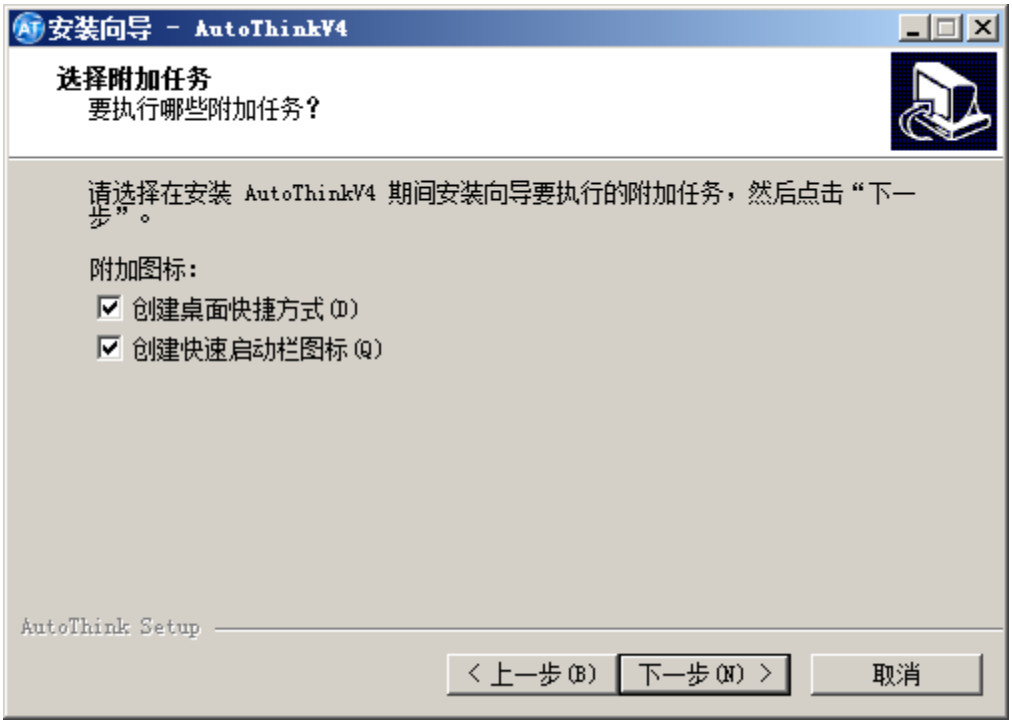
单击**浏览(R)…**选择文件夹，单击**下一步(N)**。



选择开始菜单文件夹

（5） 选择附加任务

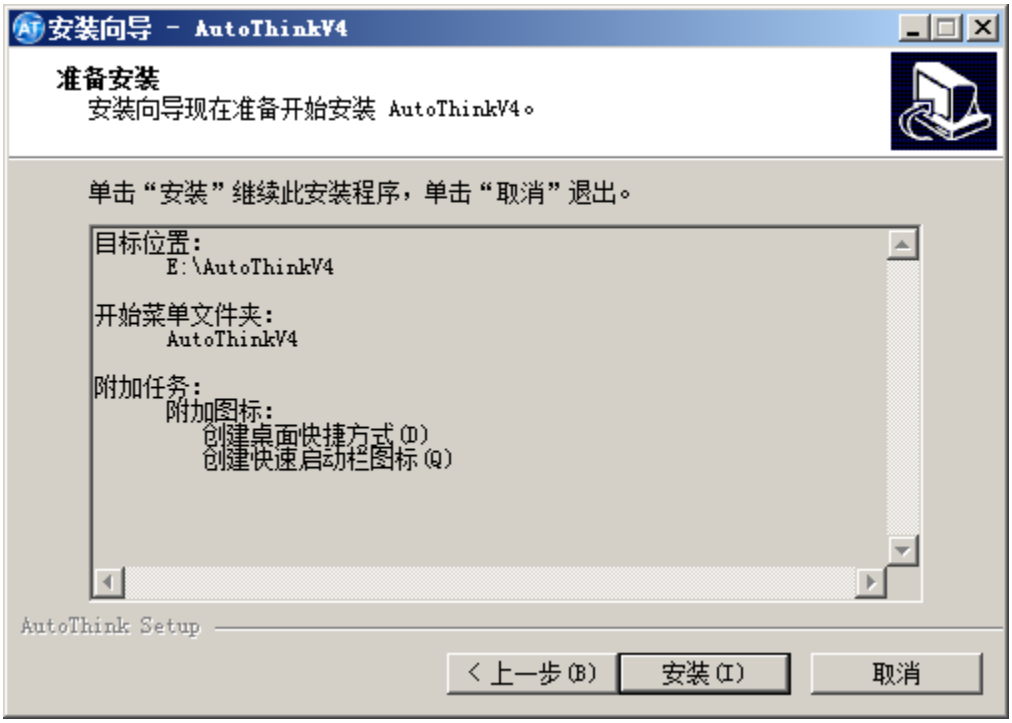
勾选需要创建的快捷图标，单击**下一步(N)**。



快捷方式创建

（6） 准备安装

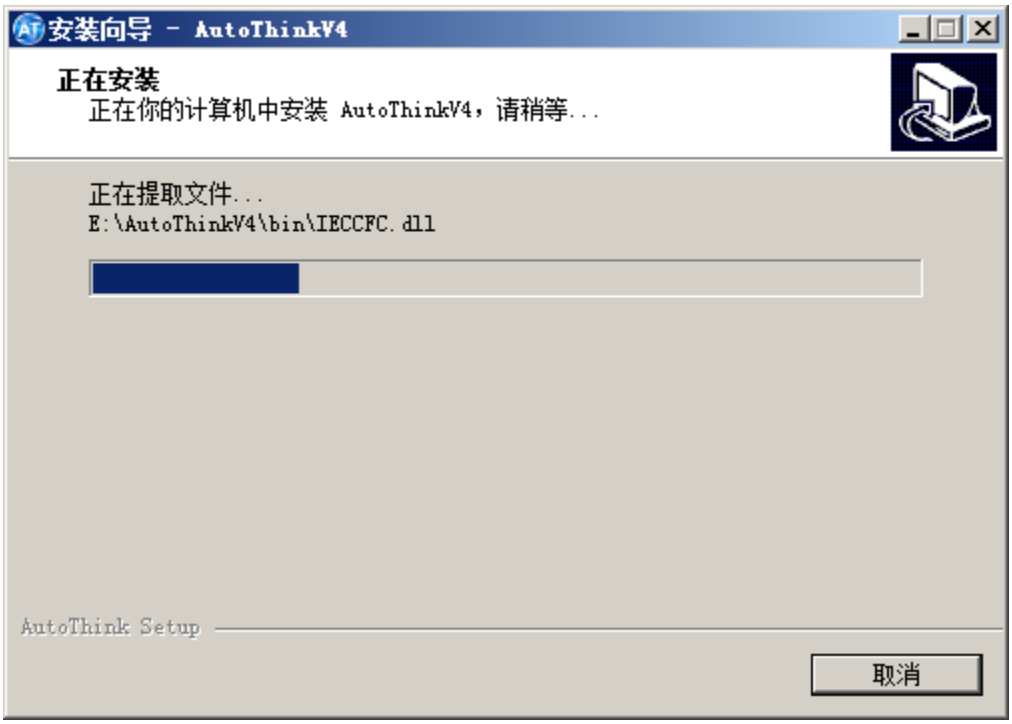
确认安装信息后，单击**安装(I)**。如果需要修改，请单击**上一步(B)**。



准备安装

（7） 安装过程

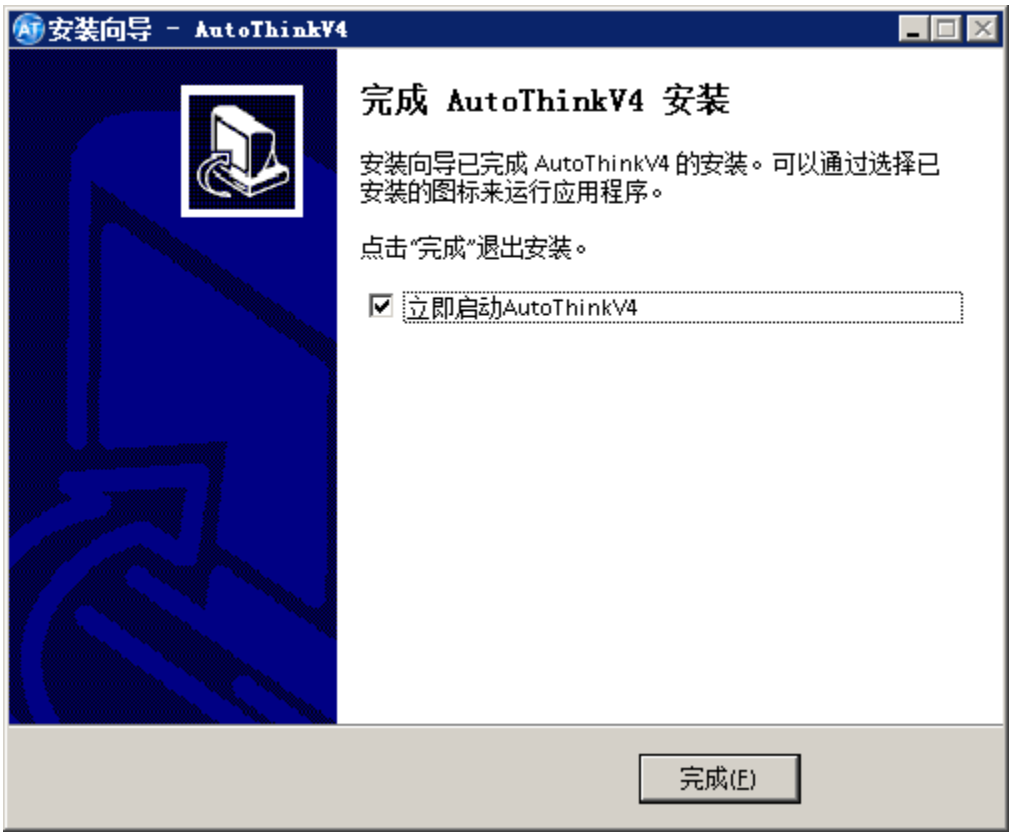
显示安装进度。若要取消安装，则单击**取消**。



安装过程

（8） 安装完成提示

安装完毕，弹出“完成 AutoThinkV4 安装”窗口。



完成安装

立即启动 AutoThinkV4：安装后，您需要立即运行 AutoThinkV4 软件，则勾选该项。

3.2.2 卸载

3.2.2.1 要求

AutoThinkV4 软件已安装

3.2.2.2 步骤

通过以下方式卸载 AutoThinkV4 软件：

- 桌面：单击“开始”菜单—“所有程序”—“AutoThinkV4”—“卸载 AutoThinkV4”。
- 在控制面板中卸载。
- 通过安装目录下 unins000.exe 文件卸载。

第4章　软件界面

4.1 启动 AutoThinkV4

4.1.1 概述

软件首次启动，如果 AutoThinkV4 软件安装时，勾选了“立即启动 AutoThinkV4”选项，则安装完成后，软件自动启动。也可通过快捷图标或程序手动启动。

支持同时开启多个 AutoThinkV4 软件进行组态。

4.1.2 要求

软件已安装完成

4.1.3 步骤

要启动 AutoThinkV4，请按以下方式执行：

- 桌面：单击“开始”菜单—“所有程序”—“AutoThinkV4”文件夹—“AutoThinkV4”软件；
- 快捷图标：双击桌面 AutoThinkV4 软件图标 

。

4.1.4 结果

软件窗口打开，你可以打开已保存的工程或者创建新工程。

4.2 退出 AutoThinkV4

4.2.1 步骤

要退出 AutoThinkV4，请按以下步骤操作：

（1） 在“文件”菜单中，选择“退出”命令。

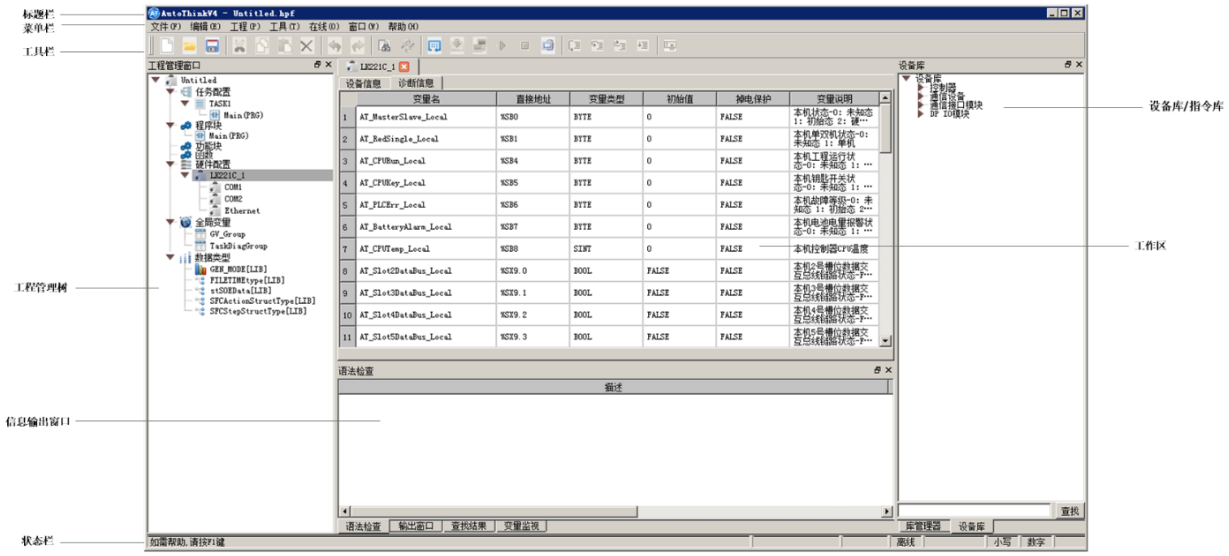
对于未保存的工程会弹出确认对话框。

（2） 确认是否未保存退出。

- 单击**是**，关闭 AutoThinkV4 软件，不保存工程修改。
- 单击**否**，取消退出操作。

4.3 软件窗口视图

AutoThinkV4 软件界面组成如下：



软件界面

4.4 标题栏

“标题栏”从左到右依次显示为：软件标识、工程名称、未保存标识、窗口控制按钮。

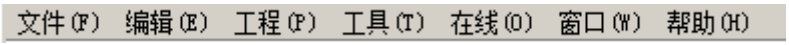
标题栏

- 软件标识：包含软件图标和软件名称。
- 工程名称：工程名.hpf，“.hpf”为工程源文件后缀。
- 未保存标识：“*”表示当前工程未保存，保存后“*”消失。
- 窗口控制按钮：分别控制软件窗口最小化、最大化、退出。
  - 最大化：单击**最大化**后，该按钮变为**还原**按钮，软件视图在当前窗口与最大化窗口间切换。
  - 最小化：窗口自动折叠，以图标形式显示在桌面的任务栏。
  - 退出：退出 AutoThinkV4。

4.5 菜单栏

4.5.1 菜单栏窗口

通过各菜单栏命令，完成相关操作任务。主菜单栏包括以下几组菜单：



菜单栏

- 文件：执行与工程文件有关的操作，例如打开、保存、关闭等。
- 编辑：对已有内容进行编辑、替换、查找等操作。
- 工程：对当前正在编辑的工程进行程编译、设置、查看等操作。
- 工具：提供上传/下传用户文件、辅助工具和工程目标文件加密操作。
- 在线：提供工程调试相关命令。
- 窗口：提供对当前打开窗口的排列方式、切换窗口显示或关闭等操作的命令。
- 帮助：提供软件信息和在线帮助。

快捷键使用说明：

- 菜单栏快捷键：使用快捷键，可不打开菜单，直接执行对应菜单命令。
- 菜单栏快捷字符：打开菜单栏后，使用（）中的快捷字符，可执行对应菜单命令。
- Alt+快捷字符：打开菜单栏后，使用 Alt+快捷字符，可执行对应菜单命令。

4.5.2 文件

菜单项	快捷键	工具栏图标	主要功能
新建	Ctrl+N		新建工程或库
打开	Ctrl+O		打开工程或库
关闭	——	——	关闭工程
保存	Ctrl+S		保存工程
另存为	F12	——	将当前工程备份到其他路径
最近工程列表	——	——	可从最近打开过的工程列表中选择工程打开
退出	——	——	关闭工程，并退出 AutoThinkV4 软件

4.5.3 编辑

菜单项	快捷键	工具栏图标	主要功能

撤销	Ctrl+Z		撤销在编辑器中所执行的动作
恢复	Ctrl+Y		恢复之前的撤销操作
剪切	Ctrl+X		剪切编辑器中的变量或程序
复制	Ctrl+C		复制编辑器中的变量或程序
粘贴	Ctrl+V		对剪切和复制的内容进行粘贴，整个工程有效
删除	Del		从工程中删除选中内容
查找	Ctrl+F		查找 POU、全局变量组、硬件配置中的文本
替换	Ctrl+H	——	查找 POU 中的文本并进行替换
输入助手	F2		组态用户程序时，通过 F2 快速选择变量、功能块、函数、操作符等

4.5.4 工程

菜单项	快捷键	工具栏图标	主要功能
编译	F11		工程修改后进行编译，首次编译为全编译，之后均为增量编译
全编译	——	——	对工程进行全编译，即工程初始化
导入变量	——	——	将 xml 格式的变量表导入到当前工程
导出变量	——	——	从当前工程导出全局变量组和程序（PRG）中的变量(系统全局变量不支持导出)
导入 POU	——	——	当需要复用其他工程的 POU 时，需要将其导入
导出 POU	——	——	从工程中导出 POU，将其导入到其他工程
导入示例工程	——	——	导入 LK238C 相关功能块的示例工程
选项	——	——	用于配置工程相关参数
导入设备描述文件	——	——	导入第三方设备描述文件
查看操作日志	——	——	查看工程的在线操作信息
权限设置	——	——	设置工程的访问权限
库自动更新	——	——	当软件版本升级或库文件更新后，需要更新工程库文件

4.5.5 工具

菜单项	主要功能
辅助工具	与控制器通讯，进行监视、下装和升级操作，可单独使用

工程目标文件加密	将工程文件与控制器 SN 号进行绑定，对该控制器参数进行独立设置、下装

4.5.6 在线

菜单项	快捷键	工具栏图标	主要功能
下装	——		将工程下装到控制器
下载用户工程	——	——	将工程源文件备份到控制器
上传用户工程	——	——	将控制器源工程文件上传到本地
在线	Ctrl+F8		进入在线监视
仿真	——		进入仿真模式
调试模式	——	——	使能断点调试模式
设置调试任务	——	——	调试模式下可用，设置需要调试的任务
断点调试窗口	——	——	勾选是否显示断点调试窗口
调用栈	——	——	勾选是否显示调用栈窗口
单步执行	F10		单步执行可打断点行
跳出	Ctrl+F9		从当前调试行跳出到调用处
跳进	Ctrl+F11		跳到调用程序第一行
断点运行	Ctrl+F5		单步执行使能断点行
运行	F5		钥匙开关在 REM 时，运行控制器任务
停止	Shift+F8		钥匙开关在 REM 时，停止控制器任务
复位	——	——	重启工程，除掉电保持数据外，所有数据被恢复成初始值
冷复位	——	——	重启工程，所有数据被恢复成初始值
热复位	——	——	重启工程，所有数据保持不变
清空控制器	——	——	清空下装到控制器中的所有工程文件和用户文件
写入	Ctrl+F7	——	在线写入变量值
强制	F7	——	在线强制变量值
全部释放	Shift+F7	——	解除所有变量的强制状态
强制变量表	Ctrl+Shift+F7	——	打开强制变量表，查看所有强制变量，并执行释放操作
通讯设置	——	——	下装前，设置控制器通讯参数
RTC 校时	——	——	校对控制器 RTC 时间

SOE 信息读取	——	——	读取 LK631C SOE 模块的通道变量历史事件信息

4.5.7 窗口

菜单项	主要功能
全部关闭	关闭工作区打开的所有窗口
工程管理窗口	选择是否显示工程管理窗口
语法检查	选择是否显示语法检查窗口
输出窗口	选择是否显示输出窗口
查找结果	选择是否显示查找结果窗口
变量监视	选择是否显示变量监视窗口
库管理窗口	选择是否显示库管理器窗口
设备库管理窗口	选择是否显示设备库窗口

4.5.8 帮助

菜单项	主要功能
目录与索引	打开帮助手册
关于 AutoThinkV4	查看软件版本信息和版权信息

4.6 工具栏

工具栏从左到右依次为主工具栏、在线工具栏、断点调试工具栏、输入助手、编程语言工具栏。编程语言工具栏根据当前打开的 POU 编程语言不同，分别显示为 LD 工具栏、ST 工具栏、CFC 工具栏和 SFC 工具栏。

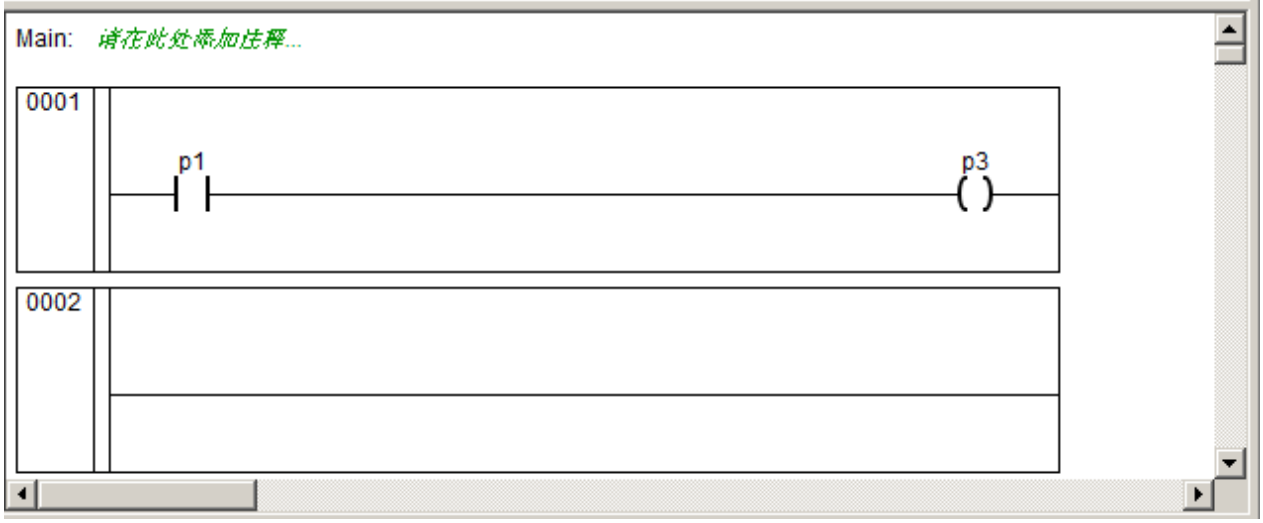
工具栏

4.7 工作区

工作区域为 POU、硬件配置、全局变量组、数据类型的组态区域，各窗口在工作区域顶端以标签页方式显示，通过单击标签页切换到对应窗口。

Main(PRG).ld

序号	变量名	直接地址	变量说明	变量类型	初始值
0001	p1			BOOL	FALSE
0002	p2			BOOL	FALSE
0003	p3			BOOL	FALSE
0004	p4			BOOL	FALSE

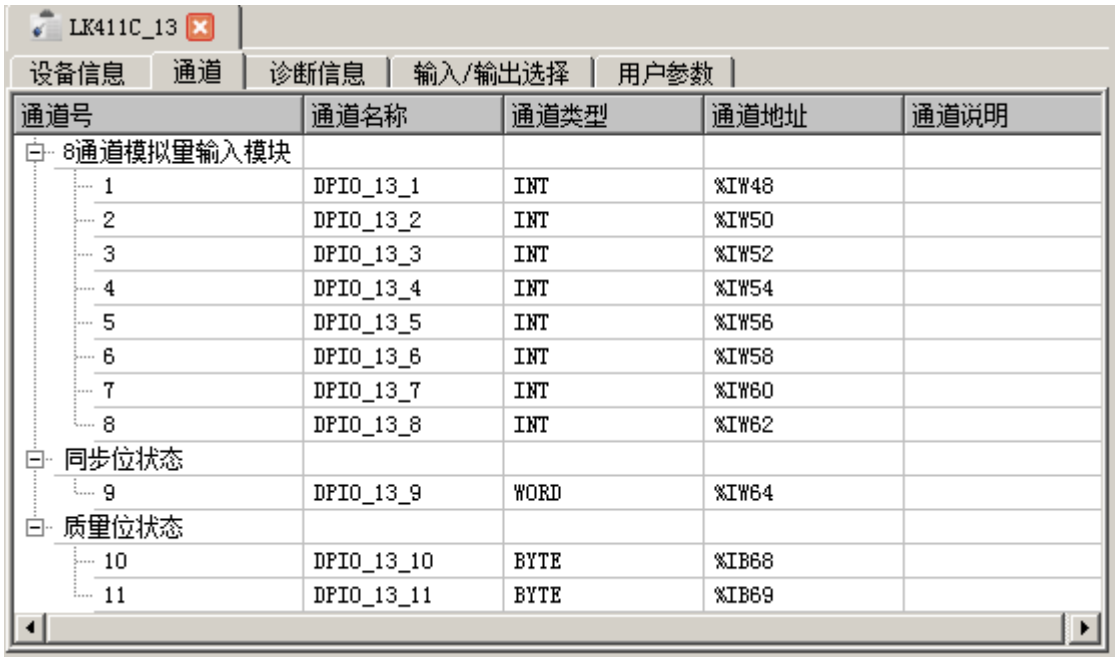


POU 窗口

GV_Group

序号	变量名	直接地址	变量说明	变量类型	初始值
0001	g1			BOOL	FALSE
0002	g2			BOOL	FALSE
0003	g3			BOOL	FALSE
0004	g4			BOOL	FALSE
0005	g5			BOOL	FALSE
0006	g6			BOOL	FALSE

全局变量区



通道号	通道名称	通道类型	通道地址	通道说明
8通道模拟量输入模块				
1	DPIO_13_1	INT	%IW48	
2	DPIO_13_2	INT	%IW50	
3	DPIO_13_3	INT	%IW52	
4	DPIO_13_4	INT	%IW54	
5	DPIO_13_5	INT	%IW56	
6	DPIO_13_6	INT	%IW58	
7	DPIO_13_7	INT	%IW60	
8	DPIO_13_8	INT	%IW62	
同步位状态				
9	DPIO_13_9	WORD	%IW64	
质量位状态				
10	DPIO_13_10	BYTE	%IB68	
11	DPIO_13_11	BYTE	%IB69	

硬件配置区

4.8 工程管理

工程管理采用树状结构图进行相应内容的管理，该种管理方式使整个工程的管理层次清楚、结构清晰、管理方便。工程管理树状图的根节点名称为当前工程名称，下面包括各种子节点：任务配置、程序块、功能块、函数、硬件配置、全局变量和数据类型。

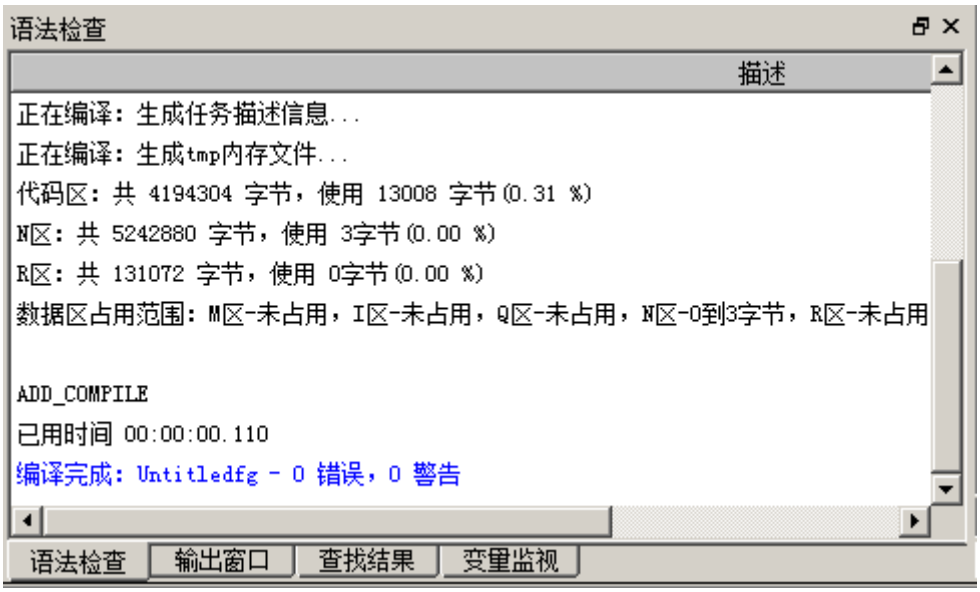
可通过“窗口”菜单，勾选是否显示工程管理窗口。

工程管理树

4.9 信息输出窗口

信息输出窗口包含语法检查、输出窗口、查找结果、变量监视 4 个标签窗口。通过“窗口”菜单，勾选是否加载各标签页。

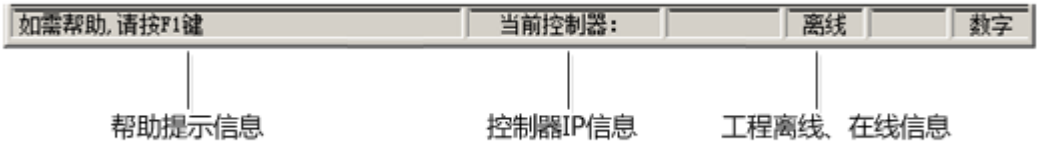
- 语法检查：上报编译过程中的错误、警告信息。
- 输出窗口：上报 PC 机与控制器间的通讯异常、错误服务信息、下装操作信息、在线操作信息。
- 查找结果：显示文本查找结果。
- 变量监视：添加监视变量并实时监视。



信息输出窗口视图

4.10 状态栏

状态栏用于显示工程相关信息，包括帮助信息、控制器 IP 以及工程状态信息等。



状态栏视图

4.11 调整窗口位置

4.11.1 概述

工程管理窗口、信息输出窗口、库管理器和设备库窗口可以设置为浮动窗口，并根据需要重新排列窗口位置。

4.11.2 要求

- 窗口已打开
- 窗口为固定状态

4.11.3 步骤

调整窗口位置，请按以下步骤操作：

（1） 单击窗口上的浮动按钮 。

此时，浮动按钮消失，窗口变为浮动状态。

（2） 鼠标选中窗口标题行，拖动窗口。

（3） 当拖动到目标窗口位置时，出现灰色虚框窗口，释放鼠标。

此时，窗口在目标位置变为固定窗口。

调整窗口

4.12 调整软件视图大小

4.12.1 概述

软件视图窗口可以最大化、最小化显示，还可以根据需要调整到适宜的窗口大小。

软件首次启动时以最大化窗口显示，单击**还原**按钮或双击标题栏，窗口调整为系统默认的尺寸大小。

4.12.2 最大化

要将调整后的窗口最大化，请按以下步骤操作

（1） 单击标题栏**最大化**按钮

窗口将以屏幕尺寸显示。

4.12.3 还原

最大化后，要再次恢复窗口，请按以下步骤操作：

（1） 单击标题栏**还原**按钮

窗口将恢复为最大化之前的视图。

4.12.4 最小化

要将窗口最小化，请按以下步骤操作

（2） 单击标题栏**最小化**按钮

窗口自动折叠，以图标形式显示在桌面的任务栏，单击软件图标，再次展开窗口。

4.12.5 调整大小

要手动调整窗口大小，请按以下步骤操作：

（1） 将鼠标放置在软件视图窗口的四个角，光标呈  。

（2） 拖拽鼠标到需要的窗口大小

窗口将在横向和纵向以同比例大小被放大或缩小。

还可通过拖动视图窗口的左右边框、上下边框分别在横向和纵向调整大小。

4.13 调整窗口显示比例

4.13.1 概述

固定软件视图大小的情况下，可以调整视图中各窗口的显示比例，如工程管理窗口、库管理器窗口、设备库窗口、信息输出窗口、工作区、编辑器窗口的变量定义区和程序逻辑区。

4.13.2 要求

软件视图窗口已打开

4.13.3 步骤

要调整软件界面各窗口显示比例，请按以下步骤操作：

（1） 将鼠标放在各窗口的共用边框上，光标呈  、  。

（2） 向光标显示方向拖动边框。

边框两侧的窗口沿光标方向被放大或缩小。

4.14 调整程序区显示比例

4.14.1 概述

程序区的显示比例可调整为 20%、50%、100%和 200%。当鼠标单击程序区时，工具栏“比例”按钮  被激活。仅 LD 编辑器支持。

4.14.2 要求

- 编辑器窗口已打开

4.14.3 步骤

要调整程序区显示比例，请按以下步骤操作：

（1） 鼠标单击程序区。

此时，工具栏“比例”按钮高亮显示。

（2） 单击工具栏按钮  。

将显示比例下拉菜单。

（3） 勾选对应比例进行设置。

程序区将显示为设置的比例大小。

第5章　工程管理

5.1 新建工程

5.1.1 概述

软件安装后，你需要创建一个工程来进行项目组态。工程文件以扩展名.hpf 格式被创建。

用户库创建方法同工程创建，这里以创建工程为例进行说明。

5.1.2 要求

软件已打开。

5.1.3 步骤

要创建新工程，请按以下步骤操作：

（1） 在“文件”菜单中，选择“新建”命令。

弹出“新建”对话框。

（2） 选择控制器型号。

创建库时，请选择“用户库”项。

（3） 输入工程名称和路径，或默认缺省设置。

- 工程名称不能以“.”或“\\”开始
- 名称不能使用“\”、“/”、“:”、“*”、“?”、“”、“<”、“>”、“|”、“空格”这 10 种字符。
- 工程名称不可为空。
- 工程名称长度不超过 32 字节。
- 工程名无效时，**确定**按钮不可操作。

（4） 单击**确定**按钮。

工程被创建，并显示在“工程管理窗口”。

5.2 保存工程

5.2.1 手动保存工程

5.2.1.1 概述

可以随时保存工程或将工程另存为其它名称。

5.2.1.2 保存

要保存工程，请按以下步骤操作：

（1） 在“文件”菜单中，选择“保存”命令。

工程以当前名称保存。

5.2.1.3 另存为

要以其它名称保存工程，请按以下步骤操作：

（1） 在“文件”菜单中，选择“另存为”命令。

弹出另存为对话框。

（2） 输入新的工程名。

- 工程名称不能以“.”或“\\”开始。
- 名称不能使用“\”、“/”、“:”、“*”、“?”、“”“、“<”、“>”、“|”、“空格”这 10 种字符。
- 工程名称不可为空。
- 工程名称长度不超过 32 字节。
- 工程名无效时，**确定**按钮不可操作。

（3） 单击**保存**按钮。

工程以新名称保存并打开。

如果工程名为空，则无法保存工程。

5.2.2 自动保存工程

5.2.2.1 概述

如果您设置了下装后自动保存，则每一次下装完成后，工程都将被保存一次。系统默认下装后自动保存。

5.2.2.2 要求

AutoThinkV4 软件在离线状态

5.2.2.3 步骤

要设置自动保存，请按以下步骤操作：

（1） 在“工程”菜单中，选择“选项”命令。

弹出“选项”窗口，默认显示“配置”页面。

（2） 勾选“下装后自动保存”。

（3） 单击**确定**。

下装完成后，“*.hpf”工程源文件被更新。

5.3 关闭工程

5.3.1 步骤

要关闭工程，请按以下步骤操作：

（1） 在“文件”菜单中，选择“关闭”命令。

对于未保存的工程会弹出确认对话框。

（2） 确认是否保存工程。

- 选择**是**，保存当前工程修改，并关闭工程。
- 选择**否**，仅关闭工程，不保存工程修改。
- 选择**取消**，取消关闭操作。

5.4 打开工程

5.4.1 概述

打开一个现有工程。打开工程前，如已经有工程被打开，请先保存关闭后，再打开。

5.4.2 条件

有已保存工程。

5.4.3 步骤

要打开一个现有工程，请按以下步骤操作：

（1） 在“文件”菜单中，选择“打开”命令。

弹出“打开工程文件”对话框。

（2） 在工程文件夹中，选择.hpf 工程文件。

（3） 单击**打开**按钮。

5.4.4 结果

工程被打开，显示在“工程管理窗口”。

5.5 工程配置

您可以在【工程】—【选项】菜单中配置工程组态、下装或在线监视时的执行项，当工程在执行某一操作时按配置选项执行。

可配置项如下：

- 监视周期：250 ms、500 ms、1000 ms 可选，默认 500 ms。
- 通迅日志记录类型：设置通迅日志是否进行记录以及记录的格式，默认为完全模式。完全模式下可记录详细的通讯信息。选择精简模式只记录通讯信息名称，不记录码值等信息。
- 下装后自动保存：设置工程下装后是否对工程进行自动保存，默认勾选。
- 下装后自动下载用户工程：缺省为勾选，执行工程下装操作后，自动下载工程源文件到控制器。
- 下装后初始化 M 区新增变量：设置下装后是否初始化 M 区的新增变量，默认不勾选。
- 调试回读：设置调试时是否将数据回读到控制站工程中，默认不勾选。
- 变量自动声明：设置在编写程序时，如果使用了未定义的变量，是否弹出变量自动声明对话框进行变量声明，默认选中状态。
- 全下装验证码确认：设置全下装时，是否需要输入验证码进行确认。
- 在线时进制显示：在线或仿真状态下，设置在线显示的数据格式。可以选择**二进制**、**十进制**或**十六进制**，默认为**十进制**显示。
- 变量导入：设置变量的导入方式，以及增量导入时同名变量的导入方式。
  - 导入方式：分清空和增量两种导入方式。选择**清空**方式，导入变量时，先清空对应 POU 或全局变量组中的变量，再执行导入。选择**增量**方式，导入变量时，导入增量变量，同名变量进行覆盖或跳过两种方式导入；
  - 同名变量：当导入方式为**增量**时可设。选择**覆盖**，导入变量时，会覆盖工程中的同名变量，并在信息输出窗口提示“XX 组：X 被覆盖！”。选择**跳过**，导入变量时，同名变量不做导入，并在信息输出窗口提示“XX 组：X 被跳过！”。

5.6 设置访问权限

5.6.1 概述

为确保工程操作安全，对工程设置密码保护，只有通过密码验证后，才能打开工程，并进行编辑和操作。

5.6.2 要求

工程已打开

5.6.3 步骤

要设置工程读写权限，请按以下步骤操作：

（1） 在“工程”菜单中，选择“权限设置”命令

将弹出“设置密码”对话框。

（2） 输入新密码

密码设置为 6~12 个字符，字符为除中文字符以外的所有字符。

（3） 再一次输入确认密码

（4） 单击**确定**按钮。

5.6.4 结果

工程已设置读写权限，下一次打开工程时，会提示输入密码。

设置操作权限后，请妥善保管密码，如密码忘记或丢失，请联系工程技术人员。

5.7 备份控制器工程

5.7.1 概述

工程下装到控制器后，需要将其本地工程源文件“*.hpf”单独备份到控制器 Flash 中，需要恢复工程时，再上传到本地即可。

5.7.2 要求

工程已编译下装、并保存

5.7.3 步骤

要备份控制器工程源文件，请按以下步骤操作：

（1） 在“在线”菜单栏中，选择“下载用户工程”命令

开始下载工程文件，并显示下载进度。

下载完成后，输出窗口显示“用户工程下载完成!”

5.8 恢复控制器工程

5.8.1 概述

当本地工程丢失、损坏，或与当前控制器工程版本不一致时，需要将控制器中备份的工程恢复到本地，与控制器工程版本保持一致。

5.8.2 要求

用户工程已备份到控制器中

5.8.3 步骤

要恢复控制器工程，请按以下步骤操作：

（1） 在“在线”菜单栏中，选择“上传用户工程”命令

将打开“选择保存路径”对话框。

（2） 选择存放工程文件的路径，单击**选择文件夹**按钮。

开始上传工程文件，并显示上传进度。

5.8.4 结果

上传完成后，输出窗口显示“用户工程上传完成!”。打开保存路径下的工程文件“*.hpf”即可编辑使用。

第6章　库管理

6.1 概述

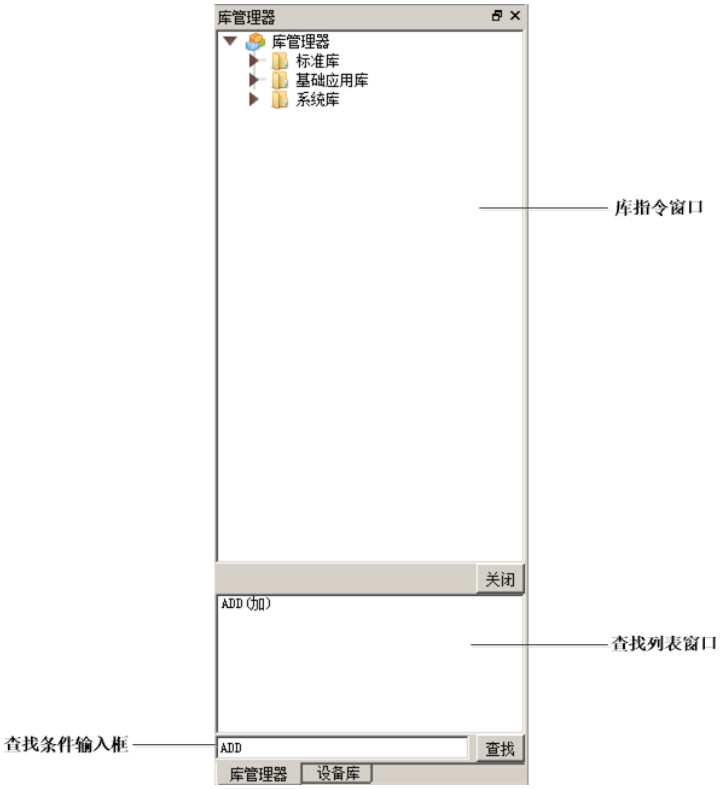
指令库分为系统库和用户库，系统库为系统默认配置的指令库，用户库为自定义库。

系统库按照功能，将其分为三大类：

- 标准库：IEC61131-3 标准规定的指令集、编译器通用的指令集。
- 基础应用库：具有通用性的应用库。
- 系统库：OS 提供的通用系统接口指令，RTS 提供的通用 PLC 系统接口指令。

用户可以自定义库详见新建工程。

6.2 库窗口



库管理器窗口视图

- 库指令窗口：显示已配置的指令库
- 查找条件输入框：输入指令名称进行模糊查找
- 查找列表窗口：显示查找结果

6.3 库配置

6.3.1 概述

所有的库文件需要使能后，才能在库管理器中加载并使用。

库管理器下缺省显示的库文件已使能，可以正常使用。如果您自定义了库指令，则需要手动配置使能。

6.3.2 要求

库管理器窗口已打开

6.3.3 步骤

要进行库配置，请按以下步骤操作：

（1） 在库管理器窗口，右击空白区域

（2） 单击“库配置”

将弹出“库配置”窗口。

（3） 勾选需要使能的库文件

（4） 单击**确定**按钮。

该库文件生效，用户可以调用已生效库文件下的指令。

6.4 更新库

6.5 概述

当更新库文件或升级软件版本后，打开旧版本工程时，需要更新库指令，否则无法正常使用。

当打开工程时，软件自动检测工程中库文件版本是否为最新，如果与 Library 目录下库文件不一致，则提示更新库文件。

此时，“库配置”窗口中，库文件状态显示“版本低”，编译时，“语法检查”窗口提示更新库。

6.6 要求

打开工程时提示更新库文件

6.7 步骤

要更新库，请按以下步骤操作：

（1） “工程”菜单中，选择“库自动更新”命令。

将弹出确认对话框。

（2） 单击**确定**。

库文件被更新，“库配置”窗口文件状态显示正常。

6.8 查找

6.8.1 要求

库管理器窗口已打开

6.8.2 规则

- 查找条件不区分字母大小写
- 搜索包含该查找条件的所有指令名称

6.8.3 步骤

要查找库指令，请按以下步骤操作：

（1） 在窗口底端的查找输入框中输入查找条件。

（2） 单击**查找**按钮

符合查找条件的指令被显示在查找列表窗口。

（3） 双击搜索到的指令进行定位

库管理器中对应指令将呈蓝色选中状态。

6.9 查看库信息

6.9.1 概述

通过此命令可以查看库指令变量信息，仅基础应用库和系统库支持查看。

6.9.2 要求

库管理器窗口已打开

6.9.3 步骤

要查看库指令信息，请按以下步骤操作：

（1） 在库管理器中，右击指令，选择“查看”命令。

在工作区显示该指令的变量定义和指令图例。

6.10 指令

6.10.1 标准库

6.10.1.1 数学运算

1. ADD（加）

第1步 功能说明

两个（或者多个）变量或常量相加。计算结果超出输出变量的取值范围将显示溢出值，对于 REAL 类型变量，运算超范围溢出，则输出显示无效值“1.#INF”或“-1.#INF”，无效值定义请参见指令系统使用注意事项。

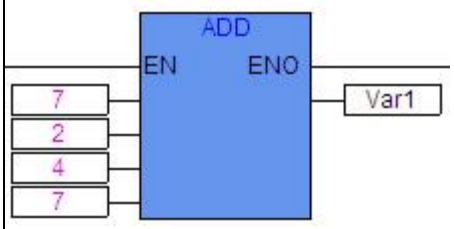
第2步 输入/输出数据类型

BYTE、WORD、DWORD、SINT、USINT、INT、UINT、DINT、UDINT、REAL、LREAL、TIME。

第3步 示例

变量定义

序号	变量名	直接地址	变量说明	变量类型	初始值	掉电保护
0001	Var1			INT	0	FALSE

编程语言	**程 序**
梯形图（LD）	 （*结果 Var1 为 20*）
结构化文本（ST）	Var1:=7+2+4+7; (*结果 Var1 为 20*)

2. SUB（减）

第1步 功能说明

两个变量或常量相减。

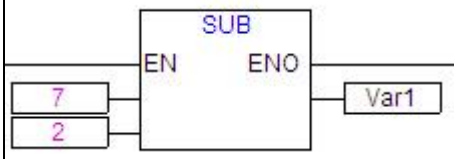
第2步 输入/输出数据类型

BYTE、WORD、DWORD、SINT、USINT、INT、UINT、DINT、UDINT、REAL、LREAL、TIME。

第3步 示例

变量定义

序号	变量名	直接地址	变量说明	变量类型	初始值	掉电保护
0001	Var1			INT	0	FALSE

编程语言	程序
梯形图（LD）	 （*结果 Var1 为 5*）
结构化文本（ST）	Var1:=7-2; （*结果 Var1 为 5*）

3. MUL（乘）

第1步 功能说明

两个（或者多个）变量或常量相乘。

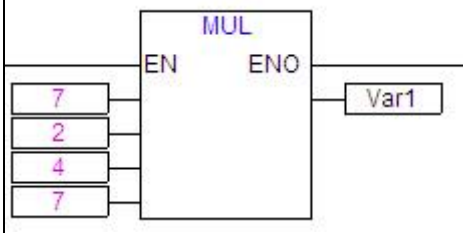
第2步 输入/输出数据类型

BYTE、WORD、DWORD、SINT、USINT、INT、UINT、DINT、UDINT、REAL、LREAL、TIME。

第3步 示例

变量定义

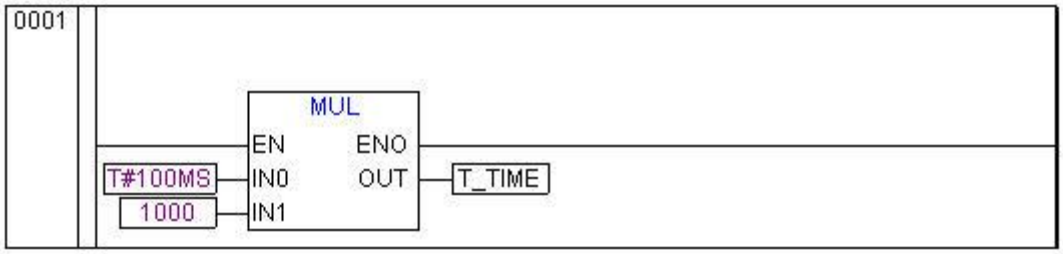
序号	变量名	直接地址	变量说明	变量类型	初始值	掉电保护
0001	Var1			INT	0	FALSE

编程语言	程 序
梯形图（LD）	 （*结果 Var1 为 392*）
结构化文本（ST）	Var1:=7*2*4*7; （*结果 Var1 为 392*）

对于 TIME 类型的变量或常量可以乘以整型变量或常量，结果为 TIME 类型变量。例如：在 AutoThinkV4 里支持 T#100MS 的时间常量*1000 的常量=时间。如下图所示。

序号	变量名	直接地址	变量说明	变量类型	初始值	掉电保护
0001	T_TIME			TIME	T#0MS	FALSE

Test:



TIME 类型乘法运算

对于 TIME 类型的乘法，只有第一个操作数为 TIME 类型数据，其余均为整型时，编译通过。其余情况报错，例如：T_TIME:=1000*T#100ms;

- LD 语言报错
  - 类型不匹配：不能将‘TIME’型转为‘WORD’型。
- ST 语言报错
  - 乘除运算：类型不匹配。

4. DIV（除）

第1步 功能说明

变量或常量相除。当除数为 0 时，结果等于被除数。

在工程中使用 DIV 指令时，可使用 CheckDivByte、CheckDivWord、CheckDivDWord 和 CheckDivReal 等指令检查除数是否为零，避免了除数为零的现象。

第2步 输入/输出数据类型

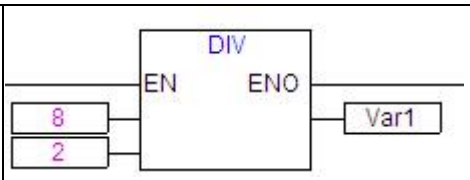
BYTE、WORD、DWORD、SINT、USINT、INT、UINT、DINT、UDINT、REAL、LREAL。

第3步 示例

变量定义

序号	变量名	直接地址	变量说明	变量类型	初始值	掉电保护
0001	Var1			INT	0	FALSE

编程语言	程 序

梯形图（LD）	 (*结果 Var1 为 4*)
结构化文本（ST）	Var1:=8/2; (*结果 Var1 为 4*)

5. MOD（取余）

第1步 功能说明

变量或常量相除取余，是一个整数。当除数为 0 时，结果等于被除数，结果符号跟被除数一致。

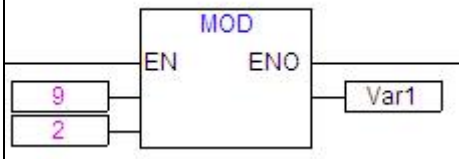
第2步 输入/输出数据类型

BYTE、WORD、DWORD、SINT、USINT、INT、UINT、DINT、UDINT。

第3步 示例

变量定义

序号	变量名	直接地址	变量说明	变量类型	初始值	掉电保护
0001	Var1			INT	0	FALSE

编程语言	**程序**
梯形图（LD）	 （*结果 Var1 为 1*）
结构化文本（ST）	Var1:=9 MOD 2; （*结果 Var1 为 1*）

6. SIZEOF（字节长度）

第1步 功能说明

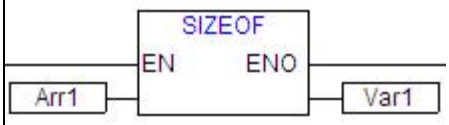
取得数据类型所需字节数。

第2步 示例

变量定义

序号	变量名	直接地址	变量说明	变量类型	初始值	掉电保护
0001	Arr1			ARRAY[0..4] OF INT		FALSE
0002	Var1			INT	0	FALSE

编程语言	**程序**

梯形图（LD）	 （*结果 Var1 为 10*）
结构化文本（ST）	Var1:=SIZEOF(arr1); （*结果 Var1 为 10*）

7. ABS（绝对值）

第1步 功能说明

把输入数据的绝对值赋予输出变量。

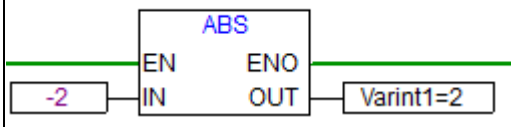
第2步 输入/输出数据类型

输入数据类型	**输出数据类型**
INT	INT、WORD、DWORD、DINT、UINT、REAL、LREAL
REAL	REAL
LREAL	LREAL
BYTE	INT、BYTE、WORD、DWORD、DINT、UINT、REAL、LREAL
WORD	WORD、DWORD、DINT、REAL、LREAL
DWORD	DWORD、DINT、REAL、LREAL
SINT	INT、BYTE、WORD、DWORD、DINT、UINT、REAL、LREAL
USINT	INT、BYTE、WORD、DWORD、DINT、UINT、REAL、LREAL
UINT	WORD、DWORD、DINT、UINT、REAL、LREAL
DINT	DWORD、DINT、REAL、LREAL
UDINT	DWORD、DINT、UDINT、REAL、LREAL

第3步 示例

变量定义

序号	变量名	直接地址	变量说明	变量类型	初始值	掉电保护
0001	Varint1			INT	0	FALSE

编程语言	程序
梯形图（LD）	
结构化文本（ST）	Varint1:=ABS(-2); (*结果 Varint1 为 2*)

8. SQRT（平方根）

第1步 功能说明

对输入数据求平方根，输入数据必须为非负数。

当输入为负数时，输出为无穷小。

第2步 输入/输出数据类型

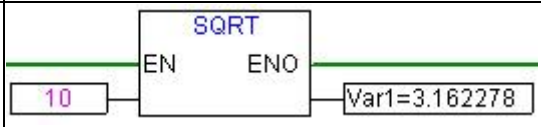
输入数据类型：BYTE、WORD、DWORD、INT、DINT、REAL、LREAL、SINT、USINT、UINT、UDINT。

输出数据类型：REAL、LREAL。

第3步 示例

变量定义

序号	变量名	直接地址	变量说明	变量类型	初始值	掉电保护
0001	Var1			REAL	0	FALSE

编程语言	程序
梯形图（LD）	
结构化文本（ST）	Var1:=SQRT(10); (*结果 Var1 为 3.162278*)

9. LOG（常用对数）

第1步 功能说明

对输入数据求以 10 为底的对数，输入数据必须为正数。

当输入≤0 时，输出为无穷小。

第2步 输入/输出数据类型

输入数据类型：BYTE、WORD、DWORD、INT、DINT、REAL、LREAL、SINT、USINT、UINT、UDINT。

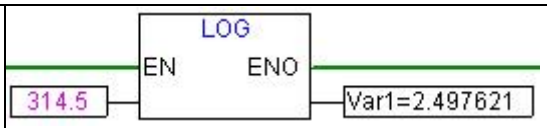
输出数据类型：REAL、LREAL。

第3步 示例

变量定义

序号	变量名	直接地址	变量说明	变量类型	初始值	掉电保护
0001	Var1			REAL	0	FALSE

编程语言	程序

梯形图（LD）	
结构化文本（ST）	Var1:=LOG(314.5); (*结果 Var1 为 2.497621 *)

10. LN（自然对数）

第1步 功能说明

对输入数据求自然对数，输入数据必须为正数。

当输入≤0 时，输出为无穷小；

第2步 输入/输出数据类型

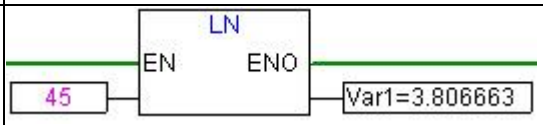
输入数据类型：BYTE、WORD、DWORD、INT、DINT、REAL、LREAL、SINT、USINT、UINT、UDINT。

输出数据类型：REAL、LREAL。

第3步 示例

变量定义

序号	变量名	直接地址	变量说明	变量类型	初始值	掉电保护
0001	Var1			REAL	0	FALSE

编程语言	程序
梯形图（LD）	
结构化文本（ST）	Var1:=LN(45)； （*结果 Var1 为 3.806663*）

11. EXP（指数）

第1步 功能说明

以输入数据为指数的幂计算，即 $y=e^x$，其中 x 为输入，y 为输出。

当输入＞709.8 时，输出值为无穷大“1.#INF”；当输入＜-745.2 时（MC＜-708.3），输出值为 0。

第2步 输入/输出数据类型

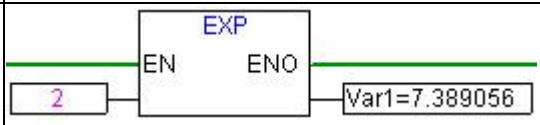
输入数据类型：BYTE、WORD、DWORD、INT、DINT、REAL、LREAL、SINT、USINT、UINT、UDINT。

输出数据类型：REAL、LREAL。

第3步 示例

变量定义

序号	变量名	直接地址	变量说明	变量类型	初始值	掉电保护
0001	Var1			REAL	0	FALSE

编程语言	程序
梯形图（LD）	
结构化文本（ST）	Var1:=EXP(2); (*结果 Var1 为 7.389056*)

12. SIN（正弦）

第1步 功能说明

求输入数据的正弦值，输入数据以弧度表示。

$$弧度(rad) = 角度 * \frac{\pi}{180^\circ}$$

第2步 输入/输出数据类型

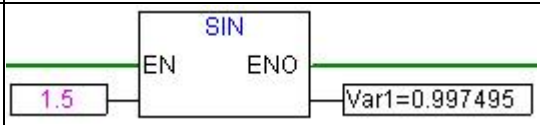
输入数据类型：BYTE、WORD、DWORD、INT、DINT、REAL、LREAL、SINT、USINT、UINT、UDINT。

输出数据类型：REAL、LREAL。

第3步 示例

变量定义

序号	变量名	直接地址	变量说明	变量类型	初始值	掉电保护
0001	Var1			REAL	0	FALSE

编程语言	程序
梯形图（LD）	
结构化文本（ST）	Var1:=SIN(1.5); (*结果 Var1 为 0.997495 *)

13. COS（余弦）

第1步 功能说明

求输入数据的余弦值，输入数据以弧度表示。

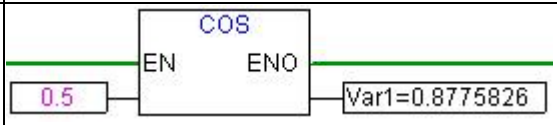
$$弧度(rad) = 角度 * \frac{\pi}{180^\circ}$$

第2步 输入/输出数据类型

输入数据类型：BYTE、WORD、DWORD、INT、DINT、REAL、LREAL、SINT、USINT、UINT、UDINT。

输出数据类型：REAL、LREAL。

变量定义						
序号	变量名	直接地址	变量说明	变量类型	初始值	掉电保护
0001	Var1			REAL	0	FALSE

编程语言	程序
梯形图（LD）	
结构化文本（ST）	Var1:=COS(0.5); (*结果 Var1 为 0.8775826 *)

14. TAN（正切）

第1步 功能说明

求输入数据的正切值，输入数据以弧度表示。

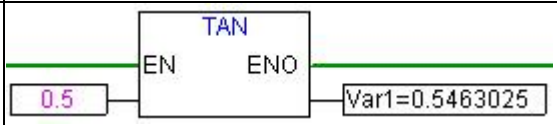
$$\text{弧度(rad)} = \text{角度} * \frac{\pi}{180^\circ}$$

第2步 输入/输出数据类型

输入数据类型：BYTE、WORD、DWORD、INT、DINT、REAL、LREAL、SINT、USINT、UINT、UDINT。

输出数据类型：REAL、LREAL。

变量定义						
序号	变量名	直接地址	变量说明	变量类型	初始值	掉电保护
0001	Var1			REAL	0	FALSE

编程语言	程序
梯形图（LD）	
结构化文本（ST）	Var1:=TAN(0.5); (*结果 Var1 为 0.5463025 *)

15. ASIN（反正弦）

第1步 功能说明

求输入数据的反正弦值。输入值取值：[-1,+1]。

输入不在[-1,1]范围时，输出为无穷小。

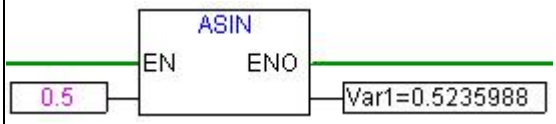
第2步 输入/输出数据类型

输入数据类型：BYTE、WORD、DWORD、INT、DINT、REAL、LREAL、SINT、USINT、UINT、UDINT。

输出数据类型：REAL、LREAL，输出数据以弧度表示。

变量定义

序号	变量名	直接地址	变量说明	变量类型	初始值	掉电保护
0001	Var1			REAL	0	FALSE

编程语言	程序
梯形图（LD）	
结构化文本（ST）	Var1:=ASIN(0.5); (*结果 Var1 为 0.5235988 *)

16. ACOS（反余弦）

第1步 功能说明

求输入数据的反余弦值。输入值取值：[-1,+1]。

第2步 输入/输出数据类型

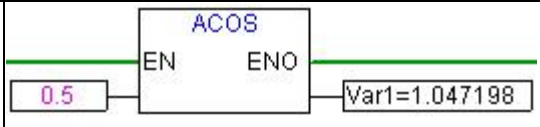
输入数据类型：BYTE、WORD、DWORD、INT、DINT、REAL、LREAL、SINT、USINT、UINT、UDINT。

输出数据类型：REAL、LREA，输出数据以弧度表示。

第3步 示例

变量定义

序号	变量名	直接地址	变量说明	变量类型	初始值	掉电保护
0001	Var1			REAL	0	FALSE

编程语言	程序
梯形图（LD）	
结构化文本（ST）	Var1:=ACOS(0.5); (*结果 Var1 为 1.047198 *)

17. ATAN（反正切）

第1步 功能说明

求输入数据的反正切值。

第2步 输入/输出数据类型

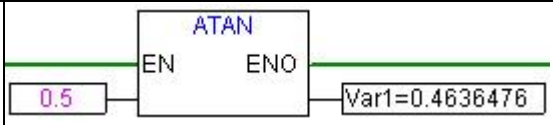
输入数据类型：BYTE、WORD、DWORD、INT、DINT、REAL、LREAL、SINT、USINT、UINT、UDINT。

输出数据类型：REAL、LREAL，输出数据以弧度表示。

第3步 示例

变量定义

序号	变量名	直接地址	变量说明	变量类型	初始值	掉电保护
0001	Var1			REAL	0	FALSE

编程语言	程序
梯形图（LD）	
结构化文本（ST）	Var1:=ATAN(0.5); (*结果 Var1 为 0.4636476 *)

18. EXPT（幂）

第1步 功能说明

计算以 x 为底，y 为指数的幂。x 为 IN1，y 为 IN2。

如果 x 和 y 都接近 0，则输出为 1。如果 x 接近 0，且 y＜0 时，输出为无穷小。当 x＜0，且 y 不是整数时，输出为非法浮点值“1.#QNAN”。运算结果溢出时，输出为无效值，无效值定义请参见指令系统使用注意事项。

第2步 输入/输出数据类型

输入数据的类型：BYTE、WORD、DWORD、INT、DINT、REAL、LREAL、SINT、USINT、UINT、UDINT。

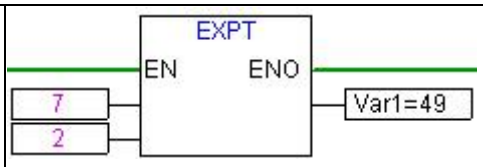
输出数据类型：REAL、LREAL。

第3步 示例

变量定义

序号	变量名	直接地址	变量说明	变量类型	初始值	掉电保护
0001	Var1			REAL	0	FALSE

编程语言	程序

梯形图（LD）	
结构化文本（ST）	Var1:=EXPT(7,2); (*结果 Var1 为 49 *)

19. MOVE（赋值）

第1步 功能说明

将一个常量或者变量的值赋给另外一个变量。

STRING 类型变量进行赋值运算时，MOVE 块连接的两个变量长度不相等时，以右侧变量的字符长度为准进行取舍。

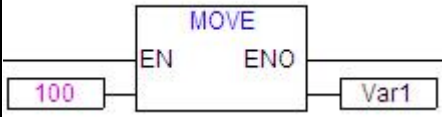
第2步 输入/输出数据类型

BYTE、WORD、DWORD、SINT、USINT、INT、UINT、DINT、UDINT、REAL、TIME、DT、BOOL、ARRAY、LREAL、STRING、TOD、DATE。

第3步 示例

变量定义

序号	变量名	直接地址	变量说明	变量类型	初始值	掉电保护
0001	Var1			INT	0	FALSE

编程语言	**程 序**
梯形图（LD）	 （*结果 Var1 为 100*）
结构化文本（ST）	Var1:=100; （*结果 Var1 为 100*）

6.10.1.2 逻辑运算

1. AND（与）

第1步 功能说明

变量或常量的相与运算。

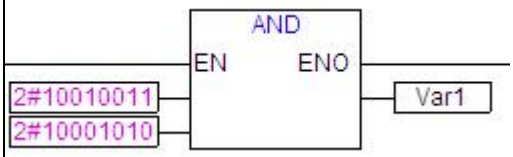
第2步 输入/输出数据类型

BOOL、BYTE、WORD、DWORD。

第3步 示例

变量定义

序号	变量名	直接地址	变量说明	变量类型	初始值	掉电保护
0001	Var1			BYTE	0	FALSE

编程语言	**程 序**
梯形图（LD）	 （*结果 Var1 为 2#10000010*）
结构化文本（ST）	Var1:=2#10010011 AND 2#10001010; (*结果 Var1 为 2#10000010*)

2. OR（或）

第1步 功能说明

变量或常量的相或运算。

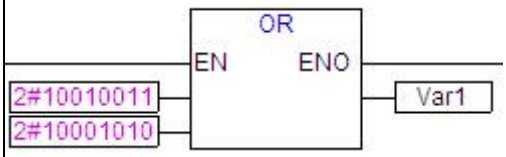
第2步 输入/输出数据类型

BOOL、BYTE、WORD、DWORD。

第3步 示例

变量定义

序号	变量名	直接地址	变量说明	变量类型	初始值	掉电保护
0001	Var1			BYTE	0	FALSE

编程语言	**程 序**
梯形图（LD）	 （*结果 Var1 为 2#10011011*）
结构化文本（ST）	Var1:=2#10010011 OR 2#10001010 ; （*结果 Var1 为 2#10011011*）

3. XOR（异或）

第1步 功能说明

变量或常量的异或运算。

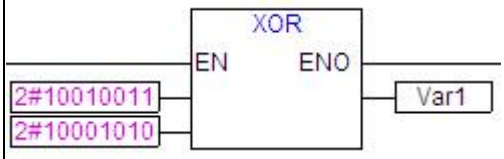
第2步 输入/输出数据类型

BOOL、BYTE、WORD、DWORD。

第3步 示例

变量定义

序号	变量名	直接地址	变量说明	变量类型	初始值	掉电保护
0001	Var1			BYTE	0	FALSE

编程语言	程 序
梯形图（LD）	 （*结果 Var1 为 2#00011001*）
结构化文本（ST）	Var1:=2#10010011 XOR 2#10001010 ; （*结果 Var1 为 2#00011001*）

4. NOT（非）

第1步 功能说明

变量或常量的取非运算，逐位取非。

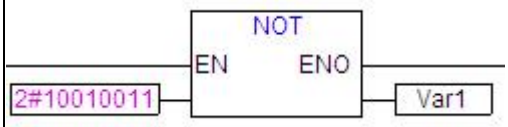
第2步 输入/输出数据类型

BOOL、BYTE、WORD、DWORD。

第3步 示例

变量定义

序号	变量名	直接地址	变量说明	变量类型	初始值	掉电保护
0001	Var1			BYTE	0	FALSE

编程语言	程 序
梯形图（LD）	 （*结果 Var1 为 2#01101100*）
结构化文本（ST）	Var1:= NOT 2#10010011 ; （*结果 Var1 为 2#01101100*）

6.10.1.3 比较运算

1. GT（大于）

第1步 功能说明

判断两个操作数的大小，当第一个数大于第二个数时输出 TRUE，否则输出为 FALSE。

第2步 输入/输出数据类型

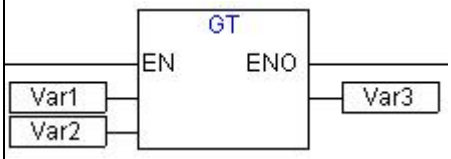
输入数据类型：BOOL、BYTE、WORD、DWORD、SINT、USINT、INT、UINT、DINT、UDINT、REAL、TIME、DATE、TOD、DT、LREAL、STRING。

输出数据类型：BOOL。

第3步 示例

变量定义

序号	变量名	直接地址	变量说明	变量类型	初始值	掉电保护
0001	Var1			INT	0	FALSE
0002	Var2			INT	0	FALSE
0003	Var3			BOOL	FALSE	FALSE

编程语言	程序
梯形图（LD）	 (*结果 Var3 为 FALSE*)
结构化文本（ST）	Var3:=20>30; (*结果 Var3 为 FALSE*)

2. LT（小于）

第1步 功能说明

判断两个操作数的大小，当第一个数小于第二个数时返回 TRUE，否则结果为 FALSE。

第2步 输入/输出数据类型

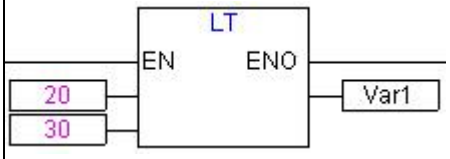
输入数据类型：BOOL、BYTE、WORD、DWORD、SINT、USINT、INT、UINT、DINT、UDINT、REAL、TIME、DATE、TOD、DT、LREAL、STRING。

输出数据类型：BOOL。

第3步 示例

变量定义

序号	变量名	直接地址	变量说明	变量类型	初始值	掉电保护
0001	Var1			BOOL	FALSE	FALSE

编程语言	程序
梯形图（LD）	 （*结果 Var1 为 TRUE*）

结构化文本（ST）	Var1:=20<30; （*结果 Var1 为 TRUE*）

3. GE（大于等于）

第1步 功能说明

判断两个操作数的大小，当第一个数大于等于第二个数时返回 TRUE，否则结果为 FALSE。

第2步 输入/输出数据类型

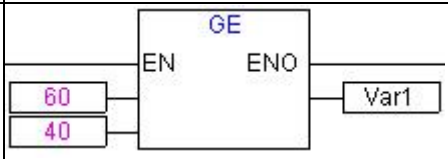
输入数据类型：BOOL、BYTE、WORD、DWORD、SINT、USINT、INT、UINT、DINT、UDINT、REAL、TIME、DATE、TOD、DT、LREAL、STRING；

输出数据类型：BOOL。

第3步 示例

变量定义

序号	变量名	直接地址	变量说明	变量类型	初始值	掉电保护
0001	Var1			BOOL	FALSE	FALSE

编程语言	程序
梯形图（LD）	 (*结果 Var1 为 TRUE*)
结构化文本（ST）	Var1:=60>=40; (*结果 Var1 为 TRUE*)

4. LE（小于等于）

第1步 功能说明

判断两个操作数的大小，当第一个数小于等于第二个数时返回 TRUE，否则结果为 FALSE。

第2步 输入/输出数据类型

输入数据类型：BOOL、BYTE、WORD、DWORD、SINT、USINT、INT、UINT、DINT、UDINT、REAL、TIME、DATE、TOD、DT、LREAL、STRING；

输出数据类型：BOOL。

第3步 示例

变量定义

序号	变量名	直接地址	变量说明	变量类型	初始值	掉电保护
0001	Var1			BOOL	FALSE	FALSE

编程语言	程序

梯形图（LD）	 (*结果 Var1 为 TRUE*)
结构化文本（ST）	Var1:=20<=30; (*结果 Var1 为 TRUE*)

5. EQ（等于）

第1步 功能说明

判断两个操作数是否相等，当第一个数等于第二个数时返回 TRUE，否则结果为 FALSE。

第2步 输入/输出数据类型

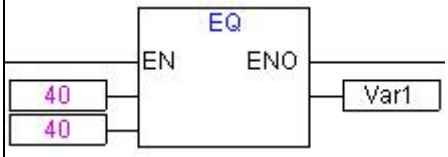
输入数据类型：BOOL、BYTE、WORD、DWORD、SINT、USINT、INT、UINT、DINT、UDINT、REAL、TIME、DATE、TOD、DT、LREAL、STRING；

输出数据类型：BOOL。

第3步 示例

变量定义

序号	变量名	直接地址	变量说明	变量类型	初始值	掉电保护
0001	Var1			BOOL	FALSE	FALSE

编程语言	程序
梯形图（LD）	 (*结果 Var1 为 TRUE*)
结构化文本（ST）	Var1:=40=40; (*结果 Var1 为 TRUE*)

6. NE（不等于）

第1步 功能说明

判断两个操作数是否不相等，当第一个数不等于第二个数时返回 TRUE，否则结果为 FALSE。

第2步 输入/输出数据类型

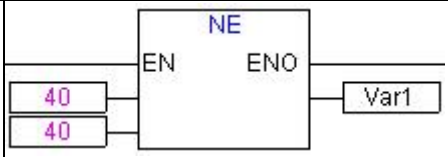
输入数据类型：BOOL、BYTE、WORD、DWORD、SINT、USINT、INT、UINT、DINT、UDINT、REAL、TIME、DATE、TOD、DT、LREAL、STRING；

输出数据类型：BOOL。

第3步 示例

变量定义

序号	变量名	直接地址	变量说明	变量类型	初始值	掉电保护
0001	Var1			BOOL	FALSE	FALSE

编程语言	程序
梯形图（LD）	 （*结果 Var1 为 FALSE*）
结构化文本（ST）	Var1:=40<>40; （*结果 Var1 为 FALSE*）

6.10.1.4 选择运算

1. SEL（二选一）

第1步 功能说明

通过选择开关在两个输入数据中选择一个作为输出，选择开关为 FALSE 时输出为第一个输入数据，选择开关为 TRUE 时输出为第二个输入数据。

第2步 指令格式

OUT := SEL(G, IN0, IN1)，其中 G 为选择开关，IN0 和 IN1 分别为第一个输入数据和第二个输入数据。

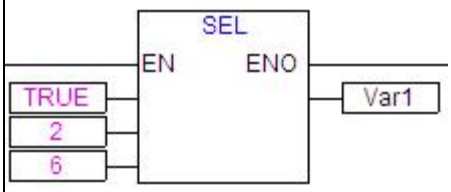
第3步 输入/输出数据类型

G 必须是 BOOL 类型，IN0、IN1 和输出数据可以是任意数据类型（不包括 STRING、数组、指针类型）。

第4步 示例

变量定义

序号	变量名	直接地址	变量说明	变量类型	初始值	掉电保护
0001	Var1			INT	0	FALSE

编程语言	**程 序**
梯形图（LD）	 （*结果 Var1 为 6 *）
结构化文本（ST）	Var1:=SEL(TRUE,3,4);

	（*结果 Var1 为 4 *）

2. MAX（取最大值）

第1步 功能说明

在两个输入数据中选择最大值作为输出。

第2步 指令格式

OUT:=MAX(IN0, IN1)，其中 IN0 和 IN1 分别为第 1 个输入数据和第 2 个输入数据，OUT 是输出数据。

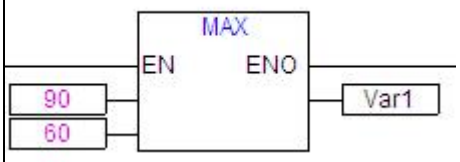
第3步 输入/输出数据类型

IN0，IN1 和 OUT 可以是任意数据类型（不包括 BOOL、STRING、数组、指针类型）。

第4步 示例

变量定义

序号	变量名	直接地址	变量说明	变量类型	初始值	掉电保护
0001	Var1			INT	0	FALSE
0002	Var2			INT	0	FALSE

编程语言	程 序
梯形图（LD）	 （*结果 Var1 为 90 *）
结构化文本（ST）	Var1:=MAX(90,60); （*结果 Var1 为 90 *） Var2:=MAX(40,MAX(90,60)); （*结果 Var2 为 90 *）

3. MIN（取最小值）

第1步 功能说明

在两个输入数据中选择最小值作为输出。

第2步 指令格式

OUT:=MIN(IN0, IN1)，其中 IN0 和 IN1 分别为第 1 个输入数据和第 2 个输入数据，OUT 是输出数据。

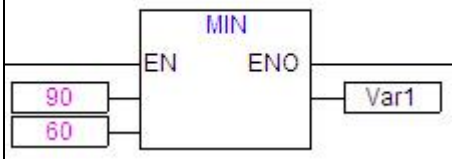
第3步 输入/输出数据类型

IN0， IN1 和 OUT 可以是任意数据类型（不包括 BOOL、STRING、数组、指针类型）。

第4步 示例

变量定义

序号	变量名	直接地址	变量说明	变量类型	初始值	掉电保护
0001	Var1			INT	0	FALSE
0002	Var2			INT	0	FALSE

编程语言	程 序
梯形图（LD）	 （*结果 Var1 为 60 *）
结构化文本（ST）	Var1:=MIN(90,30);　　（*结果 Var1 为 30 *） Var2:=MIN(MIN(90,30),60);　（*结果 Var2 为 30 *）

4. MUX（多选一）

第1步 功能说明

通过控制数在多个输入数据中选择一个作为输出。

第2步 指令格式

OUT:=MUX(K,IN0,…,INn)，其中 K 为控制数，IN0，…，INn 为输入数据，OUT 为输出结果。控制数为 K 时选择第 INk 个输入数据作为输出。

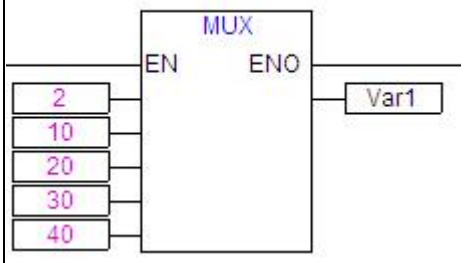
第3步 输入/输出数据类型

IN0，…，INn 和 OUT 可以是任意数据类型（不包括 BOOL、STRING、数组、指针类型），K 必须是 BYTE、WORD、DWORD、SINT、USINT、INT、UINT、DINT、UDINT。

第4步 示例

变量定义

序号	变量名	直接地址	变量说明	变量类型	初始值	掉电保护
0001	Var1			INT	0	FALSE

编程语言	程 序
梯形图（LD）	 （*2 为控制数据，对应于 30，所以输出 30 *）

结构化文本（ST）	Var1:=MUX(0,30,40,50,60,70,80); （*结果 Var1 为 30*）

5. LIMIT（限幅）

第1步 功能说明

判断输入数据是否在最小值和最大值之间，若输入数据在二者之间，则直接把输入数据作为输出数据进行输出。若输入数据大于最大值，则把最大值作为输出值。若输入数据小于最小值，则把最小值作为输出值。

第2步 指令格式

OUT := LIMIT(Min, IN, Max)。

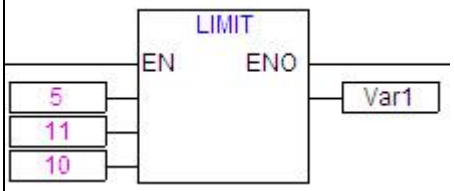
第3步 输入/输出数据类型

IN 和 OUT 可以是任意数据类型（不包括 BOOL、STRING、数组、指针类型）。

第4步 示例

变量定义

序号	变量名	直接地址	变量说明	变量类型	初始值	掉电保护
0001	Var1			INT	0	FALSE

编程语言	**程 序**
梯形图（LD）	 （*11 为输入数据，5 为最小值，10 为最大值，结果 Var1 为 10*）
结构化文本（ST）	Var1:=LIMIT(30,90,80); (*结果 Var1 为 80 *)

6.10.1.5 移位运算

1. SHL（左移）

第1步 功能说明

对操作数进行按位左移，左边移出的位不作处理，右边自动补 0。

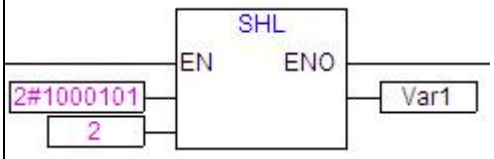
第2步 输入/输出数据类型

BYTE、INT、WORD、DWORD、SINT、UINT、USINT、DINT、UDINT。

第3步 示例

变量定义

序号	变量名	直接地址	变量说明	变量类型	初始值	掉电保护
0001	Var1			BYTE	0	FALSE

编程语言	程 序
梯形图（LD）	 （*结果 Var1 为 2#10100*）
结构化文本（ST）	Var1:=SHL(16#45,2); （*结果 Var1 为 16#14*）

2. SHR（右移）

第1步 功能说明

对操作数进行按位右移，右边移出的位不作处理，左边自动补 0。

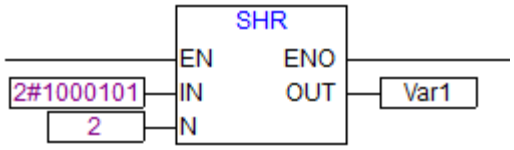
第2步 输入/输出数据类型

BYTE、INT、WORD、DWORD、SINT、UINT、USINT、DINT、UDINT。

第3步 示例

变量定义

序号	变量名	直接地址	变量说明	变量类型	初始值	掉电保护
0001	Var1			BYTE	0	FALSE

编程语言	程 序
梯形图（LD）	 （*结果 Var1 为 2#10001*）
结构化文本（ST）	Var1:=SHR(16#45,2); （*结果 Var1 为 16#11*）

3. ROL（循环左移）

第1步 功能说明

对操作数进行按位循环左移，左边移出的位直接补充到右边最低位。

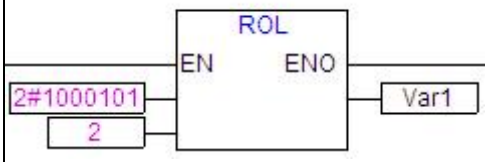
第2步 输入/输出数据类型

BYTE、INT、WORD、DWORD、SINT、UINT、USINT、DINT、UDINT。

第3步 示例

变量定义

序号	变量名	直接地址	变量说明	变量类型	初始值	掉电保护
0001	Var1			BYTE	0	FALSE

编程语言	程 序
梯形图（LD）	 （*结果 Var1 为 2#10101*）
结构化文本（ST）	Var1:=ROL(16#45,2);　（*结果 Var1 为 16#15*）

4. ROR（循环右移）

第1步 功能说明

对操作数进行按位循环右移，右边移出的位直接补充到左边最高位。

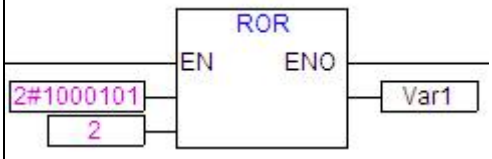
第2步 输入/输出数据类型

BYTE、INT、WORD、DWORD、SINT、UINT、USINT、DINT、UDINT。

第3步 示例

变量定义

序号	变量名	直接地址	变量说明	变量类型	初始值	掉电保护
0001	Var1			BYTE	0	FALSE

编程语言	程 序
梯形图（LD）	 （*结果 Var1 为 2#1010001*）
结构化文本（ST）	Var1:=ROR(16#45,2);　（*结果 Var1 为 16#51*）

6.10.1.6 数据类型转换

1. BOOL_TO_TYPE（BOOL 类型转换指令）

第1步 功能说明

把布尔数据类型转换为其它数据类型。

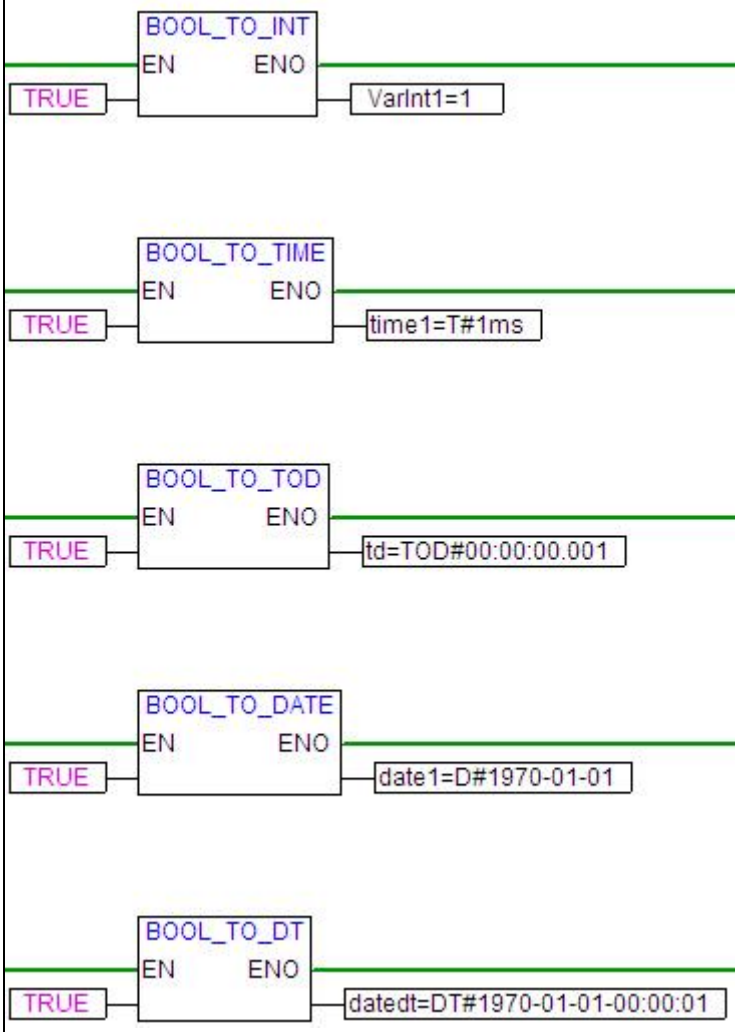
输出为数字类型时，如果输入是 TRUE，则输出 1，如果输入是 FALSE，则输出为 0。

输出为字符串类型时，如果输入是 TRUE，则输出字符串'TRUE'，如果输入是 FALSE，则输出为字符串'FALSE'。

第2步 示例

变量定义

序号	变量名	直接地址	变量说明	变量类型	初始值	掉电保护
0001	VarInt1			INT	0	FALSE
0002	time1			TIME	T#0MS	FALSE
0003	td			TOD	TOD#00:00:00	FALSE
0004	date1			DATE	D#1970-01-01	FALSE
0005	datedt			DT	DT#1970-01-01-00:00:00	FALSE

编程语言	程 序（部 分）
梯形图（LD）	
结构化文本（ST）	VarInt1:=BOOL_TO_INT(TRUE); (*结果为 1*) time1:=BOOL_TO_TIME(TRUE); (*结果为 T#1ms*) td:=BOOL_TO_TOD(TRUE); (*结果为 TOD#00:00:00.001*) date1:=BOOL_TO_DATE(TRUE); (*结果为 D#1970-01-01*) datedt :=BOOL_TO_DT(TRUE); (*结果为 DT#1970-01-01-00:00:01*)

2. BYTE_TO_TYPE（BYTE 类型转换指令）

第1步 功能说明

把字节类型转换为其他数据类型。

当 BYTE_TO_BOOL 时，输入不等于 0 时输出为 TRUE，当输入等于 0 时输出为 FALSE。

当 BYTE_TO_TIME、BYTE_TO_TOD 时，输入将以毫秒值进行转换。

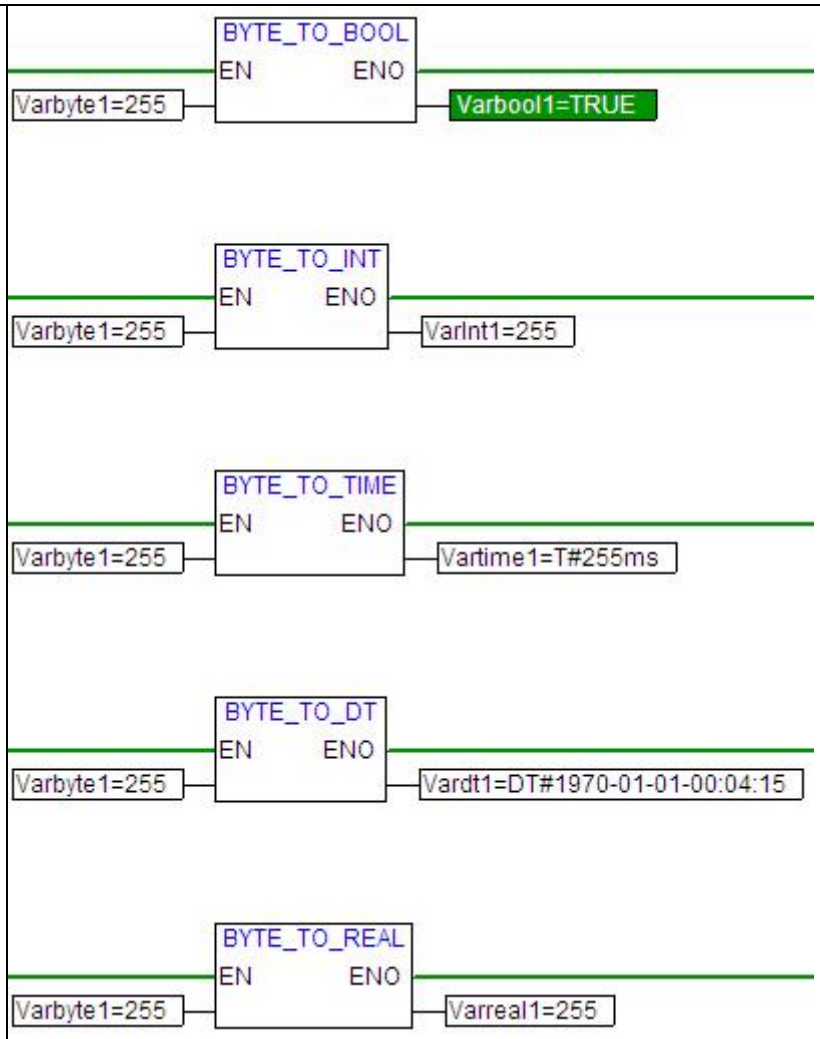
当 BYTE_TO_DATE、BYTE_TO_DT 时，输入将以秒值进行转换。

第2步 示例

变量定义

序号	变量名	直接地址	变量说明	变量类型	初始值	掉电保护
0001	Varbool1			BOOL	FALSE	FALSE
0002	Varbyte1			BYTE	0	FALSE
0003	Varint1			INT	0	FALSE
0004	Vartime1			TIME	T#0MS	FALSE
0005	Vardt1			DT	DT#1970-01-01-00:00:00	FALSE
0006	Varreal1			REAL	0	FALSE

编程语言	程 序（部 分）

梯形图（LD）	
结构化文本（ST）	Varbyte1:=16#FF; (*Varbyte1 取值*) Varbool1:=BYTE_TO_BOOL(Varbyte1); (*结果为 TRUE *) Varint1:=BYTE_TO_INT(Varbyte1); (*结果为 16# FF *) Vartime1:=BYTE_TO_TIME(Varbyte1); (*结果为 T#255ms *) Vardt1:=BYTE_TO_DT(Varbyte1); (*结果为 DT#1970-01-01-00:04:15 *) Varreal1:=BYTE_TO_REAL(Varbyte1); (*结果为 255 *)

3. DATE_TO_TYPE（DATE 类型转换指令）

第1步 功能说明

把日期型数据转换为其它类型数据，日期在内部以秒为单位存储，时间从 1970 年 1 月 1 日开始。

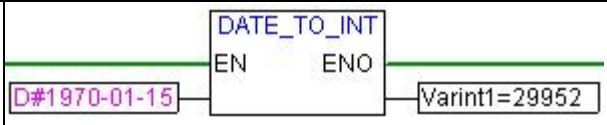
第2步 输入/输出数据类型

当 DATE_TO_BOOL 时，输入不等于 0 时，输出为 TRUE；当输入等于 0 时，输出为 FALSE。

第3步 示例

变量定义

序号	变量名	直接地址	变量说明	变量类型	初始值	掉电保护
0001	Varint1			INT	0	FALSE

编程语言	程序
梯形图（LD）	
结构化文本（ST）	VarInt1:=DATE_TO_INT(D#1970-01-15); (*结果为 29952*) 说明： 将 D#1970-01-15（十进制 14×24×3600=1209600=16#127500）保存为 INT 型变量，则会丢失高位数据，只显示低位数据 16#7500,转换十进制数为 29952

4. DINT_TO_TYPE（DINT 类型转换指令）

第1步 功能说明

双整数类型转换为其它数据类型。

第2步 输入/输出数据类型

当 DINT_TO_BOOL 时，输入不等于 0 时，输出为 TRUE；当输入等于 0 时，输出为 FALSE。

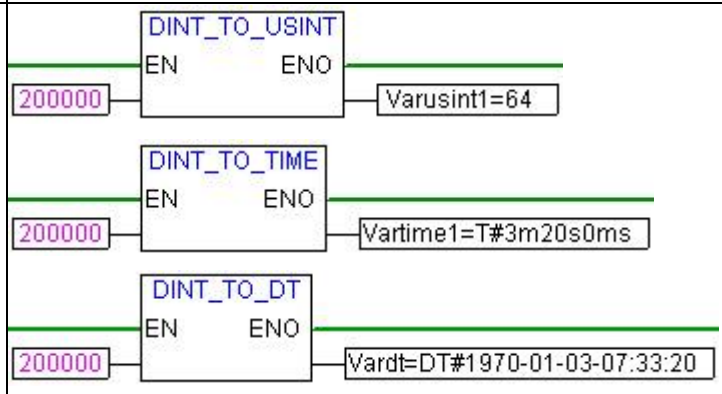
当 DINT_TO_TIME、DINT _TO_TOD 时，输入将以毫秒值进行转换。

当 DINT_TO_DATE、DINT _TO_DT 时，输入将以秒值进行转换。

第3步 示例

变量定义

序号	变量名	直接地址	变量说明	变量类型	初始值	掉电保护
0001	Vardint1			DINT	0	FALSE
0002	Varusint1			USINT	0	FALSE
0003	Vartime1			TIME	T#0MS	FALSE
0004	Vardt			DT	DT#1970-01-01-00:00:00	FALSE

编程语言	程序
梯形图（LD）	
结构化文本	Vardint1:=200000;

（ST）	Varusint1:=DINT_TO_USINT(Vardint1);　　(*结果 64*) 说明：如果将整数 200000（十六进制为 16#30D40）保存为 USINT 型变量，则会丢失高位数据，只显示低位数据 64（十六进制为 16#40） Vartime1:=DINT_TO_TIME(Vardint1);　　(*结果 T#3m20s0ms*) Vardt:=DINT_TO_DT(Vardint1);　　(*结果 DT#1970-01-03-07:33:20 *)

5. DT_TO_TYPE（DT 类型转换指令）

第1步 功能说明

把日期时间型数据转换为其它类型数据，日期在内部以秒为单位存储，时间从 1970 年 1 月 1 日开始。

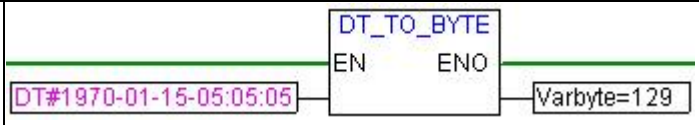
第2步 输入/输出数据类型

当 DT_TO_BOOL 时，输入不等于 0 时，输出为 TRUE；当输入等于 0 时，输出为 FALSE。

第3步 示例

变量定义

序号	变量名	直接地址	变量说明	变量类型	初始值	掉电保护
0001	Varbyte			BYTE	0	FALSE

编程语言	程序
梯形图（LD）	
结构化文本（ST）	Varbyte:=DT_TO_BYTE(DT#1970-01-15-05:05:05);　(*结果 Varbyte 为 129*) 说明： 将 DT#1970-01-15-05:05:05 转换成秒数，值为(((14*24+5)*60+5)*60+5)=1227905=16#12BC81，保存为 BYTE 型变量，则会丢失高 24 位数据，只显示低 8 位数据，16#81= 129

6. DWORD_TO_TYPE（DWORD 类型转换指令）

第1步 功能说明

把双字类型转换为其它数据类型。

当 DWORD_TO_BOOL 时，输入不等于 0 时，输出为 TRUE；当输入等于 0 时，输出为 FALSE。

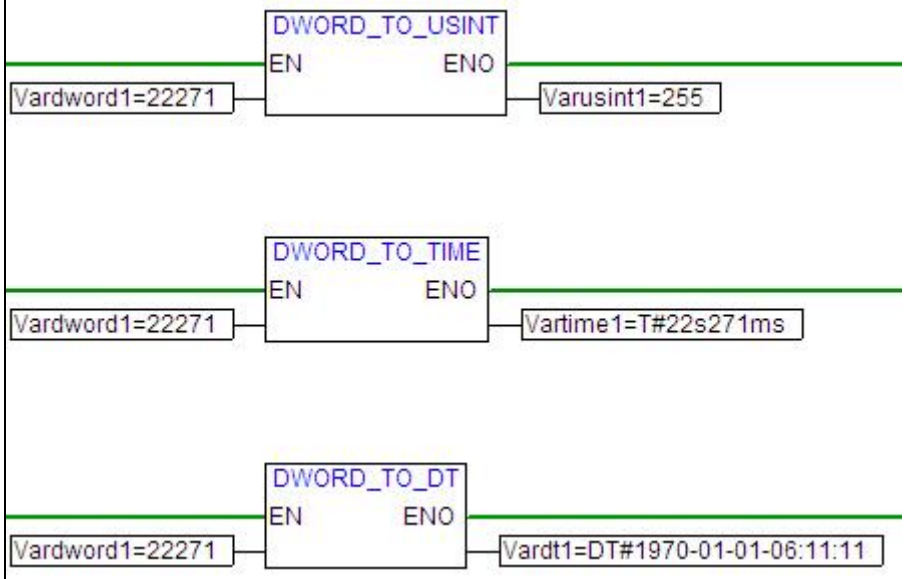
当 DWORD_TO_TIME、DWORD_TO_TOD 时，输入将以毫秒值进行转换。

当 DWORD_TO_DATE、DWORD_TO_DT 时，输入将以秒值进行转换。

第2步 示例

变量定义

序号	变量名	直接地址	变量说明	变量类型	初始值	掉电保护
0001	Varusint1			USINT	0	FALSE
0002	Vardword1			DWORD	0	FALSE
0003	Vartime1			TIME	T#0MS	FALSE
0004	Vardt1			DT	DT#1970-01-01-00:00:00	FALSE

编程语言	程序部分
梯形图（LD）	
结构化文本（ST）	Varword1:=16#56FF; (*Varword1 取值*) Varusint1:=DWORD_TO_USINT(Vardword1); (*结果 255 *) 说明：如果将整数 16#56FF（十进制为 22271）保存为 USINT 型变量，则会丢失高位数据，只显示低位数据 255（十六进制为 16#FF） Vartime1:=DWORD_TO_TIME(Vardword1); (*结果 T#22s271ms*) Vardt1:=DWORD_TO_DT(Vardword1); (*结果 DT#1970-01-01-06:11:11 *)

7. INT_TO_TYPE（INT 类型转换指令）

第1步 功能说明

把整型数据类型转换为其它数据类型。

当 INT_TO_BOOL 时，输入不等于 0 时，输出为 TRUE；当输入等于 0 时，输出为 FALSE。

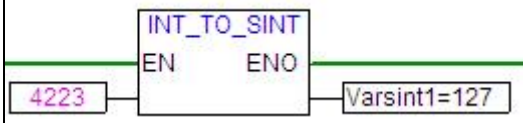
当 INT_TO_TIME、INT _TO_TOD 时，输入将以毫秒值进行转换。

当 INT_TO_DATE、INT _TO_DT 时，输入将以秒值进行转换。

第2步 示例

变量定义

序号	变量名	直接地址	变量说明	变量类型	初始值	掉电保护
0001	Varsint1			SINT	0	FALSE

编程语言	**程序部分**
梯形图（LD）	
结构化文本（ST）	VarSINT1 := INT_TO_SINT(4223);　(*结果 VarSINT1 为 127 *) 说明：如果将整数 4223（十六进制为 16#107F）保存为 SINT 型变量，则会丢失高位数据，只显示低位数据 127（十六进制为 16#7F）。

8. LREAL_TO_TYPE（LREAL 类型转换指令）

第1步 功能说明

把长实数转换为其它类型数据。

第2步 输入/输出数据类型

当 LREAL_TO_BOOL 时，输入不等于 0 时输出为 TRUE，当输入等于 0 时输出为 FALSE；当 LREAL_TO_TIME、LREAL_TO_TOD 时，输入将以毫秒值进行转换；

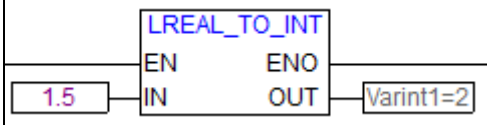
当 LREAL_TO_DATE、LREAL_TO_DT 时，输入将以秒值进行转换；

当 LREAL_TO_STRING 时，转换后的字符串长度最大为 16。

第3步 示例

变量定义

序号	变量名	直接地址	变量说明	变量类型	初始值	掉电保护
0001	Varint1			INT	0	FALSE
0002	Varint2			INT	0	FALSE
0003	Varint3			INT	0	FALSE
0004	Varint4			INT	0	FALSE

编程语言	程序
梯形图（LD）	
结构化文本（ST）	VarINT1:= LREAL_TO_INT(1.5);　(*结果 VarINT1 为 2*) VarINT2:= LREAL_TO_INT(1.4);　(*结果 VarINT2 为 1*) VarINT3:= LREAL_TO_INT(-1.5);　(*结果 VarINT3 为-2*) VarINT4:= LREAL_TO_INT(-1.4);　(*结果 VarINT4 为-1*)

9. REAL_TO_TYPE（REAL 类型转换指令）

第1步 功能说明

把实数转换为其它类型数据。

第2步 输入/输出数据类型

当 REAL_TO_BOOL 时，输入不等于 0 时输出为 TRUE，当输入等于 0 时输出为 FALSE；当 REAL_TO_TIME、REAL_TO_TOD 时，输入将以毫秒值进行转换；

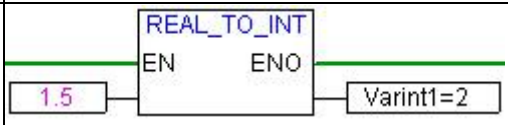
当 REAL_TO_DATE、REAL_TO_DT 时，输入将以秒值进行转换；

当 LREAL_TO_STRING 时，转换后的字符串长度最大为 16。

第3步 示例

变量定义

序号	变量名	直接地址	变量说明	变量类型	初始值	掉电保护
0001	Varint1			INT	0	FALSE
0002	Varint2			INT	0	FALSE
0003	Varint3			INT	0	FALSE
0004	Varint4			INT	0	FALSE

编程语言	程序
梯形图（LD）	
结构化文本（ST）	VarINT1:= REAL_TO_INT(1.5); (*结果 VarINT1 为 2*) VarINT2:= REAL_TO_INT(1.4); (*结果 VarINT2 为 1*) VarINT3:= REAL_TO_INT(-1.5); (*结果 VarINT3 为-2*) VarINT4:= REAL_TO_INT(-1.4); (*结果 VarINT4 为-1*)

10. SINT_TO_TYPE（SINT 类型转换指令）

第1步 功能说明

把短整型转换为其它数据类型。

当 SINT_TO_BOOL 时，输入不等于 0 时，输出为 TRUE；当输入等于 0 时，输出为 FALSE。

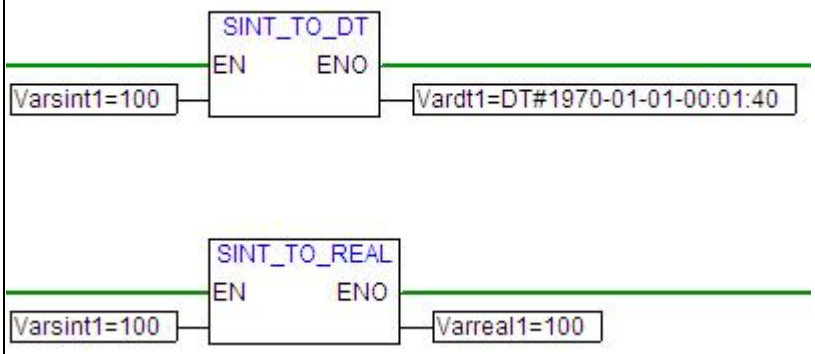
当 SINT_TO_TIME、SINT _TO_TOD 时，输入将以毫秒值进行转换。

当 SINT_TO_DATE、SINT _TO_DT 时，输入将以秒值进行转换。

第2步 示例

变量定义

序号	变量名	直接地址	变量说明	变量类型	初始值	掉电保护
0001	Varsint1			SINT	100	FALSE
0002	Vardt1			DT	DT#1970-01-01-00:00:00	FALSE
0003	Varreal1			REAL	0	FALSE

编程语言	**程 序（部 分）**
梯形图（LD）	
结构化文本（ST）	Varsint1:=100; (*Varsint1 取值*) Vardt1:=SINT_TO_DT(Varsint1); (*结果 DT#1970-01-01-00:01:40 *) Varreal1:=SINT_TO_REAL(Varsint1); (*结果为 100 .0*)

11. STRING_TO_TYPE（STRING 类型转换指令）

第1步 功能说明

把 STRING 类型变量转换为其它数据类型。

对于 STRING_TO_BOOL 指令，输入字符串'TRUE'或'true'，则输出 TRUE；除此之外，均输出 FALSE。

第2步 示例

变量定义

序号	变量名	直接地址	变量说明	变量类型	初始值	掉电保护
0001	Var_a			TIME	T#0MS	FALSE
0002	Var_b			BOOL	FALSE	FALSE
0003	Var_c			DINT	0	FALSE
0004	STRING01			STRING(80)	'4589763'	FALSE
0005	STRING02			STRING(80)	'1239876'	FALSE
0006	TIME01			TIME	T#0MS	FALSE
0007	DINT01			DINT	0	FALSE

编程语言	程序

结构化文本（ST）

序号	变量名	直接地址	变量说明	变量类型	在线值	掉电保护
0001	b			TIME	T#127ms	FALSE
0002	c			BOOL	TRUE	FALSE
0003	d			DINT	124378	FALSE

```
0001 b:=STRING_TO_TIME('T#127MS');
0002 c:= STRING_TO_BOOL ('TRUE');
0003 d:= STRING_TO_DINT ('124378');
0004
```

```
b = T#127ms
c = TRUE
d = 124378
```

12. TIME_TO_TYPE（TIME 类型转换指令）

第1步 功能说明

把时间型数据转换为其它类型数据，时间在内部以毫秒为单位存储成 DWORD 类型（对于 TIME_OF_DAY 变量从凌晨 00：00 开始）。

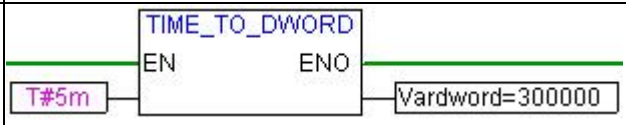
第2步 输入/输出数据类型

当 TIME_TO_BOOL 时，输入不等于 0 时，输出为 TRUE；当输入等于 0 时，输出为 FALSE。

第3步 示例

变量定义

序号	变量名	直接地址	变量说明	变量类型	初始值	掉电保护
0001	Vardword			DWORD	0	FALSE

编程语言	程序
梯形图（LD）	
结构化文本（ST）	Vardword:=TIME_TO_DWORD(T#5m); (*结果为 300000*)

13. TOD_TO_TYPE（TOD 类型转换指令）

第1步 功能说明

把时间型数据转换为其它类型数据，日期在内部以毫秒为单位进行转化。

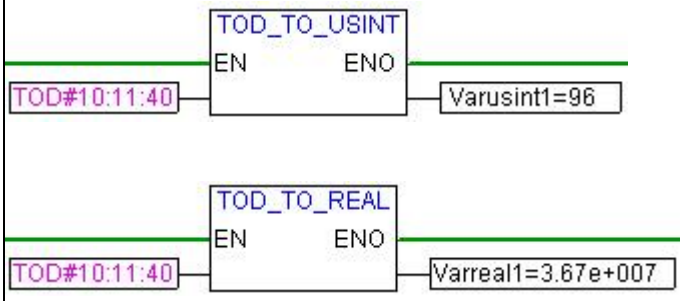
第2步 输入/输出数据类型

当 TOD_TO_BOOL 时，输入不等于 0 时，输出为 TRUE；当输入等于 0 时，输出为 FALSE。

第3步 示例

变量定义

序号	变量名	直接地址	变量说明	变量类型	初始值	掉电保护
0001	Vartod1			TOD	TOD#00:00:00	FALSE
0002	Varusint1			USINT	0	FALSE
0003	Vartime1			TIME	T#0MS	FALSE

编程语言	程序
梯形图（LD）	
结构化文本（ST）	Vartod1:=TOD#10:11:40; (*Vartod1 取值*) Varusint1:=TOD_TO_USINT(Vartod1); (*结果为 96*) Varreal1:=TOD_TO_REAL(Vartod1); (*结果为 3.67e+007*)

14. UDINT_TO_TYPE（UDINT 类型转换指令）

第1步　功能说明

无符号双整数类型转换为其它数据类型。

第2步　输入/输出数据类型

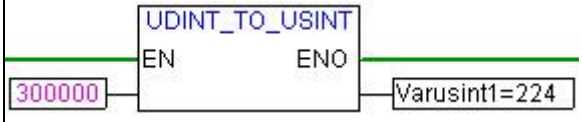
当 UDINT_TO_BOOL 时，输入不等于 0 时输出为 TRUE，当输入等于 0 时输出为 FALSE；当 UDINT_TO_TIME、UDINT _TO_TOD 时，输入将以毫秒值进行转换；

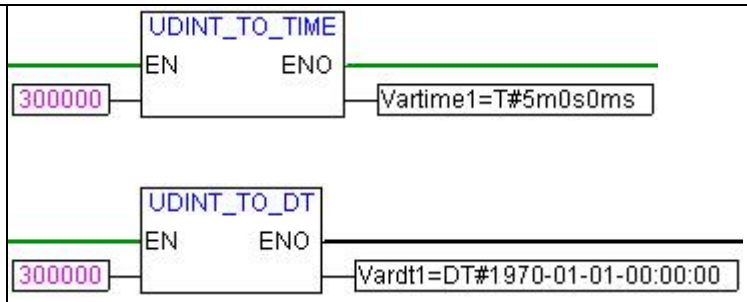
当 UDINT_TO_DATE、UDINT _TO_DT 时，输入将以秒值进行转换。

第3步　示例

变量定义

序号	变量名	直接地址	变量说明	变量类型	初始值	掉电保护
0001	Varudint1			UDINT	0	FALSE
0002	Varusint1			USINT	0	FALSE
0003	Vartime1			TIME	T#0MS	FALSE
0004	Vardt1			DT	DT#1970-01-01-00:00:00	FALSE

编程语言	程序
梯形图（LD）	

	
结构化文本（ST）	Varudint1:=300000; Varusint1:= UDINT_TO_USINT(Varudint1); (*结果 224*) 说明：如果将整数 300000（十六进制为 16#493E0）保存为 USINT 型变量，则会丢失高位数据，只显示低位数据 224（十六进制为 16#E0）。 Vartime1:=UDINT_TO_TIME(Varudint1); (*结果 T#5m0s0ms*) Vardt1:=UDINT_TO_DT(Varudint1); (*结果 DT#1970-01-01-00:00:00 *)

15. UINT_TO_TYPE（UINT 类型转换指令）

第1步 功能

无符号整数类型转换为其它数据类型。

当 UINT_TO_BOOL 时，输入不等于 0 时，输出为 TRUE；当输入等于 0 时，输出为 FALSE。

当 UINT_TO_TIME、UINT _TO_TOD 时，输入将以毫秒值进行转换。

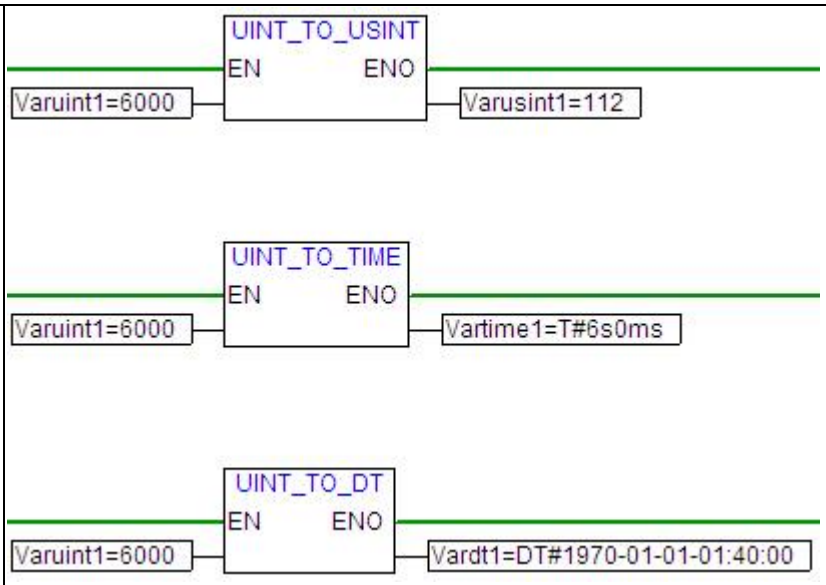
当 UINT_TO_DATE、UINT _TO_DT 时，输入将以秒值进行转换。

第2步 示例

变量定义

序号	变量名	直接地址	变量说明	变量类型	初始值	掉电保护
0001	Varuint1			UINT	6000	FALSE
0002	Varusint1			USINT	0	FALSE
0003	Vartime1			TIME	T#0MS	FALSE
0004	Vardt1			DT	DT#1970-0...	FALSE

编程语言	**程 序（部 分）**

梯形图（LD）	
结构化文本（ST）	Varuint1:=6000; Varusint1:=UINT_TO_USINT(Varuint1); 说明：如果将整数 6000（十六进制为 16#1770）保存为 USINT 型变量，则会丢失高位数据，只显示低位数据 112（十六进制为 16#70）。 Vartime1:=UINT_TO_TIME(Varuint1); Vardt1:=UINT_TO_DT(Varuint1);

16. USINT_TO_TYPE（USINT 类型转换指令）

第1步 功能说明

把无符号短整型转换为其它数据类型。

当 USINT_TO_BOOL 时，输入不等于 0 时，输出为 TRUE；当输入等于 0 时，输出为 FALSE。

当 USINT_TO_TIME、USINT _TO_TOD 时，输入将以毫秒值进行转换。

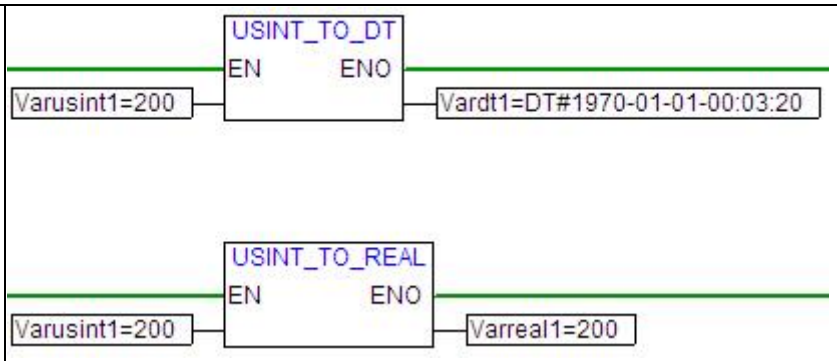
当 USINT_TO_DATE、USINT _TO_DT 时，输入将以秒值进行转换。

第2步 示例

变量定义

序号	变量名	直接地址	变量说明	变量类型	初始值	掉电保护
0001	Varusint1			USINT	0	FALSE
0002	Vardt1			DT	DT#1970-01-01-00:00:00	FALSE
0003	Varreal1			REAL	0	FALSE

编程语言	**程 序（部 分）**

梯形图（LD）	
结构化文本（ST）	Varusint1:=200;　(*Varusint1 取值*) Vardt1:=USINT_TO_DT(Varusint1);　(*结果 DT#1970-01-01-00:03:20 *) Varreal1:=USINT_TO_REAL(Varusint1);　(*结果为 200 .0*)

17. WORD_TO_TYPE（WORD 类型转换指令）

第1步　功能说明

把字类型转换为其它数据类型。

当 WORD_TO_BOOL 时，输入不等于 0 时输出为 TRUE，当输入等于 0 时输出为 FALSE。

当 WORD_TO_TIME、WORD_TO_TOD 时，输入将以毫秒值进行转换。

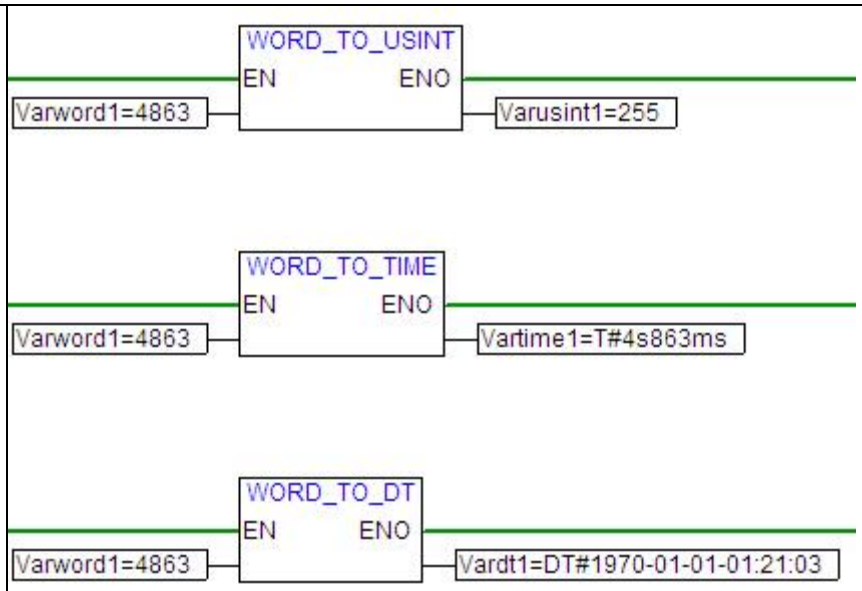
当 WORD_TO_DATE、WORD_TO_DT 时，输入将以秒值进行转换。

第2步　示例

变量定义

序号	变量名	直接地址	变量说明	变量类型	初始值	掉电保护
0001	Varusint1			USINT	0	FALSE
0002	Varword1			WORD	0	FALSE
0003	Vartime1			TIME	T#0MS	FALSE
0004	Vardt1			DT	DT#1970-01-01-00:00:00	FALSE

编程语言	程 序（部 分）

梯形图（LD）	
结构化文本（ST）	Varword1:=4863； (*Varword1 取值*) Varusint1:=WORD_TO_USINT(Varword1); (*结果 255 *) 说明：如果将整数 4863（十六进制为 16#12FF）保存为 USINT 型变量，则会丢失高位数据，只显示低位数据 255（十六进制为 16#FF）。 Vartime1:=WORD_TO_TIME(Varword1); (*结果 T#4s863ms*) Vardt1:=WORD_TO_DT(Varword1); (*结果 DT#1970-01-01-01:21:03 *)

18. TRUNC

第1步 功能说明

实现对浮点数取整数位功能。

第2步 输入/输出数据类型

输入：REAL；

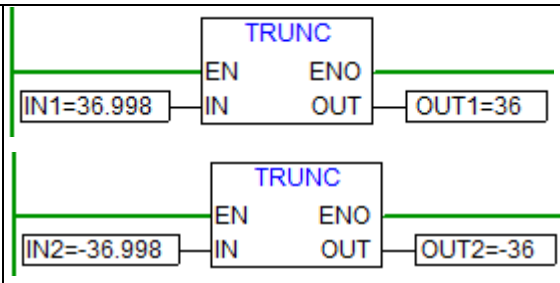
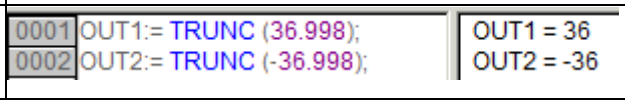
输出：DINT。

第3步 示例

变量定义

序号	变量名	直接地址	变量说明	变量类型	初始值	掉电保护
0001	IN1			REAL	0	FALSE
0002	OUT1			DINT	0	FALSE
0003	IN2			REAL	0	FALSE
0004	OUT2			DINT	0	FALSE

编程语言	程序

梯形图（LD）	
结构化文本（ST）	

6.10.1.7 指针

1. ADR（取地址）

第1步 功能说明

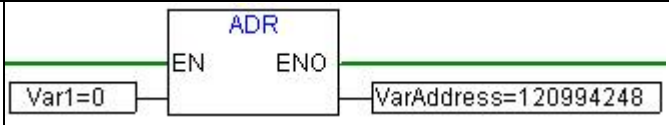
取得输入变量的内存地址，并输出。该地址可以在程序内当作指针使用，也可以作为指针传送给函数。

使用时务必保证指针类型与所指的变量类型匹配。

第2步 示例

变量定义

序号	变量名	直接地址	变量说明	变量类型	初始值	掉电保护
0001	Var1			BYTE	0	FALSE
0002	VarAddress			POINTER TO BYTE	0	FALSE

编程语言	程序
梯形图（LD）	
结构化文本（ST）	VarAddress:= ADR(Var1);

2. VAL（取值）

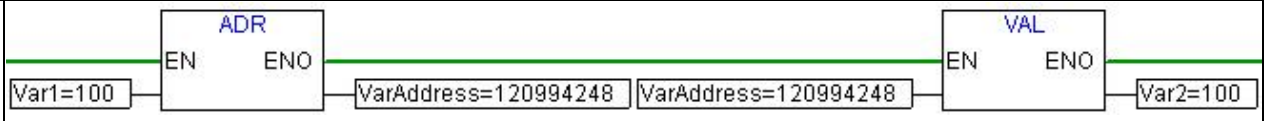
第1步 功能说明

取得指针所指地址的数据。

第2步 示例

变量定义

序号	变量名	直接地址	变量说明	变量类型	初始值	掉电保护
0001	Var1			BYTE	0	FALSE
0002	Var2			BYTE	0	FALSE
0003	VarAddress			POINTER TO BYTE	0	FALSE

编程语言	程序
梯形图（LD）	
结构化文本（ST）	VarAddress:= ADR(Var1); Var2:= VAL(VarAddress); (*结果 Var2 为 100 *)

6.10.1.8 字符串

1. Left

第1步 函数原型

STRING LEFT (STRING str, INT count)

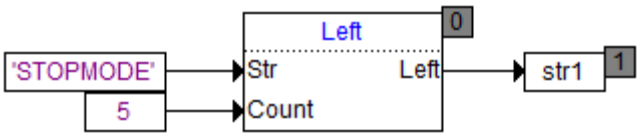
第2步 功能说明

返回指定字符串的最左边的 count 个字符构成的子字符串。

第3步 参数说明

	数据项	数据类型	参 数 描 述
输入端	Str	STRING	输入字符串，最大长度 128
	Count	BYTE、WORD、DWORD、SINT、USINT、INT、UINT、DINT 和 UDINT	要返回的子字符串长度
输出端	Left	STRING	返回子字符串。 (1) 若 str 为空，则返回空 (2) 若 count 小于或等于 0，则返回空 (3) 若 str 实际长度大于 128，取前 128 个字符 (4) 若 count 大于或等于 str 长度，则返回 str

第4步 示例



str1 的值为'STOPM'。

2. Right

第1步 函数原型

STRING RIGHT (STRING str, INT count)

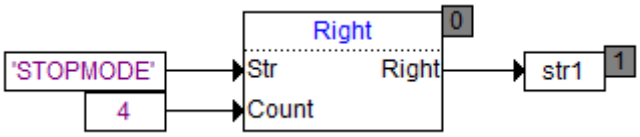
第2步 功能说明

返回指定字符串的最右边的 count 个字符构成的子字符串。

第3步 参数说明

	数据项	数据类型	参 数 描 述
输入端	Str	STRING	输入字符串，最大长度 128
	Count	BYTE、WORD、DWORD、SINT、USINT、INT、UINT、DINT 和 UDINT	要返回的子字符串长度
输出端	Right	STRING	返回子字符串。 (1) 若 str 为空，则返回空 (2) 若 count 小于或等于 0，则返回空 (3) 若 str 实际长度大于 128，取前 128 个字符 (4) 若 count 大于或等于 str 长度，则返回 str

第4步 示例



str1 的值为'MODE'。

3. Concat

第1步 函数原型

STRING CONCAT (STRING str1, STRING str2)

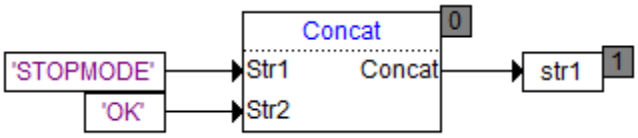
第2步 功能说明

连接两个字符串。

第3步 参数说明

	数据项	数据类型	参数说明
输入端	Str1	STRING	输入字符串，最大长度 128
	Str2	STRING	输入字符串，最大长度 128
输出端	Concat	STRING	返回子字符串。 (1) 输入字符串变量长度大于 128 时，取前 128 个字符 (2) 若 str1 长度+str2 长度大于 128，则返回空

第4步 示例



str1 的值为'STOPMODEOK'。

4. Delete

第1步 函数原型

STRING DELETE (STRING str, INT len, INT pos)

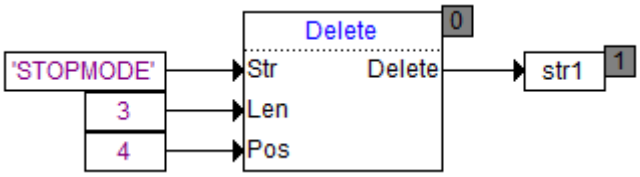
第2步 功能说明

从 str 第 pos 个字符开始删除 len 个字符，并返回结果。

第3步 参数说明

	数据项	数据类型	参 数 描 述
输入端	Str	STRING	输入字符串，最大长度 128
	Len	BYTE、WORD、SINT、USINT、INT、UINT	要删除的字符个数
	Pos	BYTE、WORD、SINT、USINT、INT、UINT	要删除的字符在 str 字符串中的起始位置
输出端	Delete	STRING	(1) 若 str 实际长度大于 128，取前 128 个字符； (2) 若 len 小于 0，则 pos=pos+len，len=-len (3) 若 pos 小于或等于 0，则 len=len+pos-1，pos=1 (4) 若 pos+len 大于 str 长度，则 len = str 长度-pos+1 (5) 若 pos 大于 str 长度或 len 小于等于 0，则返回 str（若其长度大于 128，则返回前 128 个字符）

第4步 示例



str1 的值为'STODE'。

5. Insert

第1步 函数原型

STRING INSERT (STRING str1, STRING str2, INT pos)

第2步 功能说明

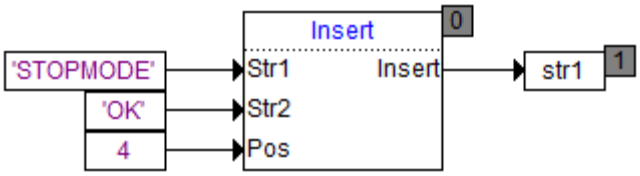
将 str2 插入到 str1 的第 pos 个字符之后，并返回结果。

第3步 参数说明

	数据项	数据类型	参 数 描 述
输入端	Str1	STRING	输入字符串，最大长度 128
	Str2	STRING	输入字符串，最大长度 128
	Pos	BYTE、WORD、SINT、USINT、INT、UINT	str2 字符串要插入到 str1 字符串中的起始位置
输出端	Insert	STRING	(1) 输入字符串变量长度大于 128 时，取前 128 个字符 (2) 若 str1 长度+str2 长度大于 128，则返回空

			(3) pos=0，插入到 str1 前面 (4) pos=1，插入到第一个字符之后 (5) 若 pos 大于 str1 长度或小于 0，则返回 str1（若其长度大于 128，则返回前 128 个）

第4步 示例



str1 的值为'STOPOKMODE'。

6. Mid

第1步 函数原型

STRING MID (STRING str, INT len, INT pos)

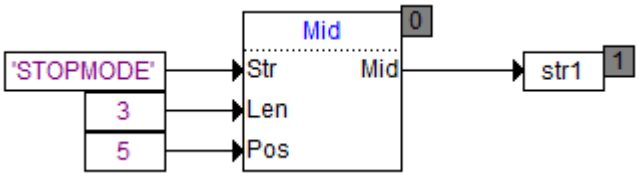
第2步 功能说明

返回字符串中从第 pos 个字符开始连续 len 个字符构成的子串。

第3步 参数说明

	数据项	数据类型	参 数 描 述
输入端	Str	STRING	输入字符串，最大长度 128
	Len	BYTE、WORD、SINT、USINT、INT、UINT	要返回的子字符串长度
	Pos	BYTE、WORD、SINT、USINT、INT、UINT	要返回的子字符串的起始位置
输出端	Mid	STRING	返回子字符串。 (1) 若 str 为空，则返回空 (2) 若 str 实际长度大于 128，取前 128 个字符 (3) 若 pos 小于或等于 0，则返回空 (4) 若 pos 大于 str 长度，则返回空 (5) 若 len 小于或等于 0，则返回空 (6) 若 len 大于实际可取的字符个数，则返回 pos 开始后面所有字符

第4步 示例



str1 的值为'MOD'。

7. Replace

第1步 函数原型

STRING REPLACE (STRING str1, STRING str2, INT len, INT pos)

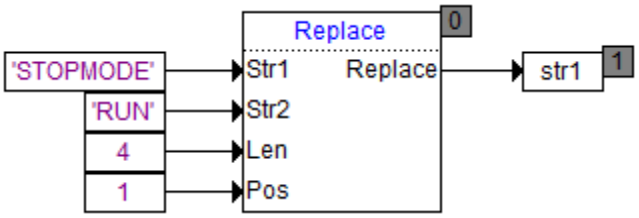
第2步 功能说明

用 str2 替换 str1 第 pos 个字符开始 len 个字符，并返回结果

第3步 参数说明

	数据项	数据类型	参数描述
输入端	Str1	STRING	输入字符串，最大长度 128
	Str2	STRING	输入字符串，最大长度 128
	Len	BYTE、WORD、SINT、USINT、INT、UINT	要替换的字符个数
	Pos	BYTE、WORD、SINT、USINT、INT、UINT	要替换的字符在 str1 字符串中的起始位置
输出端	Replace	STRING	(1) 输入字符串变量长度大于 128 时，取前 128 个字符 (2) 若 len 小于 0，pos=pos+len，len=-len (3) 若 pos 小于等于 0，len=len+pos-1，pos=1 (4) 若 pos+len 大于 str1 长度，len=str1 长度-pos+1 (5) 若 len 大于等于 0，则从 str1 第 pos 个字符 len 个字符开始替换为 str2 (6) 若替换后，str1 长度大于 128，则截取前 128 个返回

第4步 示例



str1 的值为'RUNMODE'。

8. Find

第1步 函数原型

INT FIND (STRING str1, STRING str2)

第2步 功能说明

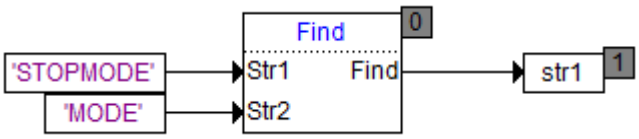
查找 str2 在 str1 中第一次出现的首字符位置并返回。

第3步 参数说明

	数据项	数据类型	参 数 描 述
输入端	Str1	STRING	输入字符串，最大长度 128

	Str2	STRING	输入字符串，最大长度 128
输出端	Find	WORD、DWORD、INT、UINT、DINT、UDINT	(1) 若 str2 在 str1 中没有出现，则返回 0 (2) 若 str1，str2 中有一个为空，则返回 0 (3) 输入字符串变量长度大于 128 时，取前 128 个字符

第4步 示例



str1 的值为 5。

9. Len

第1步 函数原型

INT LEN (STRING str)

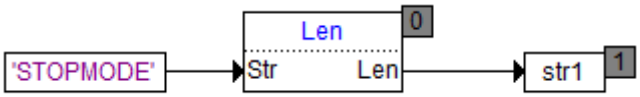
第2步 功能说明

返回输入字符串的长度。

第3步 参数说明

	数据项	数据类型	参 数 描 述
输入端	Str	STRING	输入字符串，最大长度 128
输出端	Len	WORD、DWORD、INT、UINT、DINT、UDINT	(1) 若 str 实际长度大于 128，取前 128 个字符 (2) 返回字符串的长度，不包括字符串结束符 (3) 若输入字符串为空，则返回 0

第4步 示例



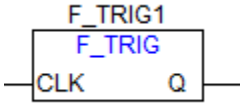
str1 的值应为 8。

6.10.1.9 沿触发器指令

1. F_TRIG（下降沿检测触发器）

第1步 功能说明

用于检测下降沿。



F_TRIG 功能块

第2步 逻辑关系

Q := NOT CLK AND NOT M;

M := NOT CLK;

M 是初始值为 TRUE 的一个中间变量，只要 CLK 是 TRUE，Q 和 M 保持 FALSE。CLK 是 FALSE，Q 首次返回 TRUE，M 设置为 TRUE。每次调用指令时，Q 返回 FALSE，当 CLK 检测到下降沿时，Q 为 TRUE。

第3步 参数说明

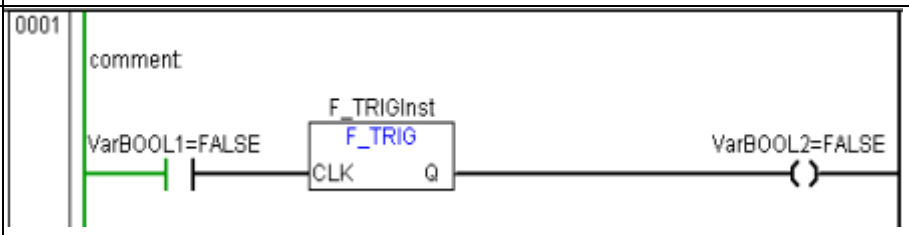
输入参数	数据类型	功能说明	参数值说明	默认值
CLK	BOOL	输入	下降沿触发信号	TRUE

输出参数	数据类型	功能说明	参数值说明	默认值
Q	BOOL	输出	当检测到下降沿时，Q=TRUE 每次调用指令时，Q=FALSE	0

第4步 示例

变量定义

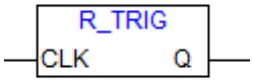
序号	变量名	直接地址	变量说明	变量类型	初始值	掉电保护
0001	F_TRIGInst			F_TRIG		FALSE
0002	VarBOOL1			BOOL	FALSE	FALSE
0003	VarBOOL2			BOOL	FALSE	FALSE

编程语言	程序
梯形图（LD）	
结构化文本（ST）	FTRIGInst(CLK:=VarBOOL1); VarBOOL2:=FTRIGInst.Q;

2. R_TRIG（上升沿检测触发器）

第1步 功能说明

用于检测上升沿。



R_TRIG 功能块

第2步 逻辑关系

Q := CLK AND NOT M;

M := CLK;

M 是初始值为 FALSE 的一个中间变量，只要 CLK 是 FALSE，Q 和 M 就是 FALSE。每次调用指令时，Q 返回 FALSE。当 CLK 检测到上升沿时，Q 返回 TRUE。

第3步 参数说明

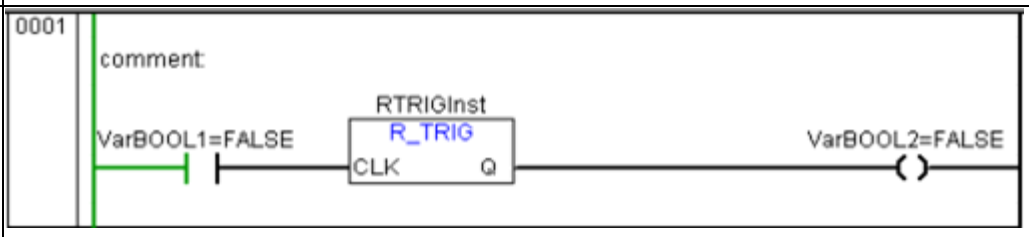
输入参数	数据类型	功能说明	参数值说明	默认值
CLK	BOOL	输入	上升沿触发信号	FALSE

输出参数	数据类型	功能说明	参数值说明	默认值
Q	BOOL	输出	当检测到上升沿时，Q=TRUE 每次调用指令时，Q=TRUE	0

第4步 示例

变量定义

序号	变量名	直接地址	变量说明	变量类型	初始值	掉电保护
0001	R_TRIGInst			R_TRIG		FALSE
0002	VarBOOL1			BOOL	FALSE	FALSE
0003	VarBOOL2			BOOL	FALSE	FALSE

编程语言	程序
梯形图（LD）	
结构化文本（ST）	RTRIGInst(CLK:=VarBOOL1); VarBOOL2:=RTRIGInst.Q;

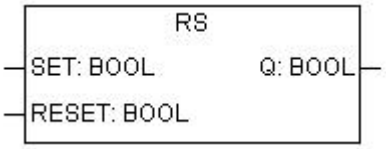
6.10.1.10双稳态指令

1. RS（复位优先双稳态器）

第1步 功能说明

复位双稳态功能算法（RS）实现复位优先的 RS 触发器功能，输入端 SET、RESET，输出端 Q 均为 BOOL 型变量。

详细功能说明如下：Q1 = RS (SET, RESET) 表示：Q1 = NOT RESET AND (Q1 OR SET)。



RS 功能块

指令的真值表如下表所示。

真值表

Reset	Set	Q(n+1)
0	0	Q(n)
0	1	1
1	0	0
1	1	0

第2步 参数说明

输入参数	数据类型	功能说明	默认值	掉电保护
Set	BOOL	置位	FALSE	FALSE
Reset	BOOL	复位	FALSE	FALSE

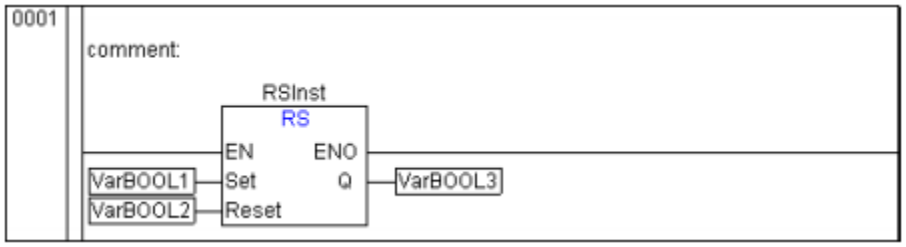
输出参数	数据类型	功能说明	默认值	掉电保护
Q	BOOL	输出端	FALSE	TRUE

第3步 示例

变量定义

序号	变量名	直接地址	变量说明	变量类型	初始值	掉电保护
0001	RSInst			RS		FALSE
0002	VarBOOL1			BOOL	FALSE	FALSE
0003	VarBOOL2			BOOL	FALSE	FALSE
0004	VarBOOL3			BOOL	FALSE	FALSE

编程语言	程序

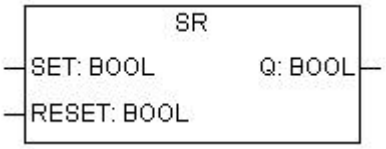
梯形图（LD）	 说明：VarBOOL3＝NOT VarBOOL2 AND (VarBOOL3 OR VarBOOL1)
结构化文本（ST）	RSInst(Set:=VarBOOL1, Reset:=VarBOOL2, Q=>VarBOOL3);

2. SR（置位优先双稳态器）

第1步 功能说明

双稳态功能算法（SR）实现置位优先的 RS 触发器功能，输入端 SET、RESET，输出端 Q 均为 BOOL 型变量。

详细功能如下：Q1 = SR (SET, RESET) 表示：Q1 = (NOT RESET AND Q1) OR SET。



SR 功能块

指令的真值表如下表所示。

真值表

Reset	Set	Q(n+1)
0	0	Q(n)
0	1	1
1	0	0
1	1	1

第2步 参数说明

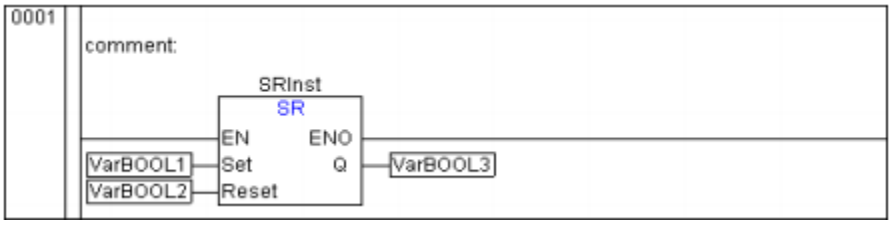
输入参数	说明	初始值（默认）	变量类型	掉电保护
Reset	复位	FALSE	BOOL	FALSE
Set	置位	FALSE	BOOL	FALSE

输出参数	说明	初始值（默认）	变量类型	掉电保护
Q	输出端	FALSE	BOOL	TRUE

第3步 示例

变量定义

序号	变量名	直接地址	变量说明	变量类型	初始值	掉电保护
0001	SRInst			SR		FALSE
0002	VarBOOL1			BOOL	FALSE	FALSE
0003	VarBOOL2			BOOL	FALSE	FALSE
0004	VarBOOL3			BOOL	FALSE	FALSE

编程语言	程序
梯形图（LD）	 说明：VarBOOL3＝（NOT VarBOOL2 AND VarBOOL3）OR VarBOOL1
结构化文本（ST）	SRInst(Set:= VarBOOL1, Reset:= VarBOOL2, Q=> VarBOOL3);

6.10.1.11计数器指令

1. CTD（递减计数器（INT））

第1步 功能说明

递减计数器（CTD）实现当输入 CD 存在上升沿时,计数值递减功能。每检测到一个信号上升沿，计数值 CV 减 1，直到达到其数据类型下限-32768。

递减计数过程如下：

（1） 初始化：

通过输入信号 Load 将输出 Q 和 CV 值赋值为初始设定状态。当 Load 为 TRUE，计数值 CV 被赋值 PV 设定值，Q 被赋值为 FALSE。此时，CD 输入信号不会触发计数，CV 保持当前 PV 值。

（2） 触发计数

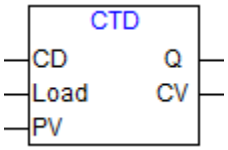
当 Load 变为 FALSE，CD 信号从 FALSE 变为 TRUE 时，CV 从 PV 设置值开始减 1。

（3） 输出置位

当 CV 值递减到 0 及 0 以下时，Q 输出被置为 TRUE,直到 Load 信号将其赋为初始值 FALSE。

（4） 停止计数

当 CV 值递减到其数据类型下限-32768 时，CD 输入信号不再触发计数，CV 保持下限值，直到 Load 信号将其赋值为初始设定值。



CTD 指令示意图

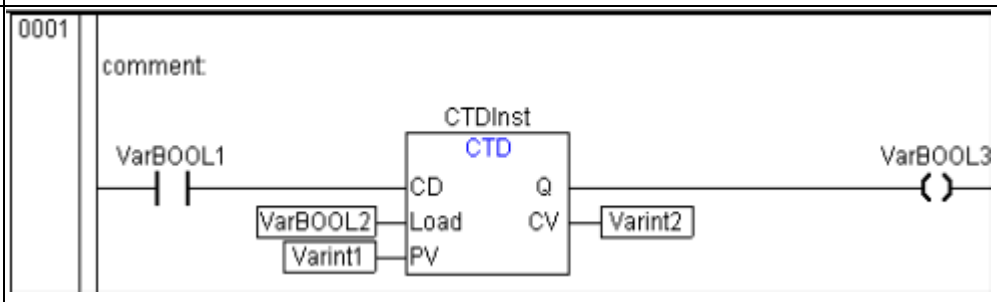
第2步 参数说明

参数类型	参数名称	数据类型	参数说明	初始值（默认）	掉电保护
输入参数	CD	BOOL	上升沿检测端	FALSE	FALSE
	Load	BOOL	初始化信号	FALSE	FALSE
	PV	INT	计数器设定值	0	TRUE
输出参数	Q	BOOL	计数标志	TRUE	FALSE
	CV	INT	当前计数值	0	TRUE

第3步 示例

变量定义

序号	变量名	直接地址	变量说明	变量类型	初始值	掉电保护
0001	CTDInst			CTD		FALSE
0002	VarBOOL1			BOOL	FALSE	FALSE
0003	VarBOOL2			BOOL	FALSE	FALSE
0004	Varint1			INT	0	FALSE
0005	VarBOOL3			BOOL	FALSE	FALSE
0006	Varint2			INT	0	FALSE

编程语言	程序
梯形图（LD）	
结构化文本（ST）	CTDInst(CD:=VarBOOL1, Load:=VarBOOL2, PV:=Varint1,

	Q=>VarBOOL3, CV=>Varint2);

2. CTD_DINT（递减计数器（DINT））

第1步 功能说明

递减计数器（CTD_DINT）实现当输入 CD 存在上升沿时,计数值递减功能。每检测到一个信号上升沿，计数值 CV 减 1，直到达到其数据类型下限-2,147,483,648。

递减计数过程如下：

（1） 初始化：

通过输入信号 Load 将输出 Q 和 CV 值赋值为初始设定状态。当 Load 为 TRUE，计数值 CV 被赋值 PV 设定值，Q 被赋值为 FALSE。此时，CD 输入信号不会触发计数，CV 保持当前 PV 值。

（2） 触发计数

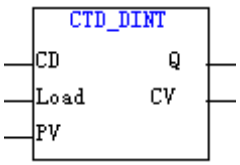
当 Load 变为 FALSE，CD 信号从 FALSE 变为 TRUE 时，CV 从 PV 设置值开始减 1。

（3） 输出置位

当 CV 值递减到 0 及 0 以下时，Q 输出被置为 TRUE,直到 Load 信号将其赋为初始值 FALSE。

（4） 停止计数

当 CV 值递减到其数据类型下限-2,147,483,648 时，CD 输入信号不再触发计数，CV 保持下限值，直到 Load 信号将其赋值为初始设定值。



CTD_DINT 指令示意图

第2步 参数说明

参数类型	参数名称	数据类型	参数说明	初始值（默认）	掉电保护
输入参数	CD	BOOL	上升沿检测端	FALSE	FALSE
	Load	BOOL	初始化信号	FALSE	FALSE
	PV	DINT	计数器设定值	0	TRUE
输出参数	Q	BOOL	计数标志输出	TRUE	FALSE
	CV	DINT	当前计数值	0	TRUE

第3步 示例

请参见 CTD（递减计数器（INT））。

3. CTD_UDINT（递减计数器（UDINT））

第1步 功能说明

递减计数器（CTD_UDINT）实现当输入 CD 存在上升沿时,计数值递减功能。每检测到一个信号上升沿，计数值 CV 减 1，直到达到其数据类型下限 0。

递减计数过程如下：

（1） 初始化：

通过输入信号 Load 将输出 Q 和 CV 值赋值为初始设定状态。当 Load 为 TRUE，计数值 CV 被赋值 PV 设定值，Q 被赋值为 FALSE。此时，CD 输入信号不会触发计数，CV 保持当前 PV 值。

（2） 触发计数

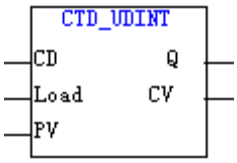
当 Load 变为 FALSE，CD 信号从 FALSE 变为 TRUE 时，CV 从 PV 设置值开始减 1。

（3） 输出置位

当 CV 值递减到 0 及 0 以下时，Q 输出被置为 TRUE,直到 Load 信号将其赋为初始值 FALSE。

（4） 停止计数

当 CV 值递减到其数据类型下限 0 时，CD 输入信号不再触发计数，CV 保持下限值，直到 Load 信号将其赋值为初始设定值。



CTD_UDINT 指令示意图

第2步 参数说明

参数类型	参数名称	数据类型	参数说明	初始值（默认）	掉电保护
输入参数	CD	BOOL	上升沿检测端	FALSE	FALSE
	Load	BOOL	初始化信号	FALSE	FALSE
	PV	UDINT	计数器设定值	0	TRUE
输出参数	Q	BOOL	计数标志输出	TRUE	FALSE
	CV	UDINT	当前计数值	0	TRUE

第3步 示例

请参见 CTD（递减计数器（INT））。

4. CTU（递增计数器（INT））

第1步 功能说明

递增计数器（CTU）实现当输入 CU 存在上升沿时,计数值递增功能。每检测到一个信号上升沿，计数值 CV 加 1，直到达到其数据类型上限 32767。

递增计数过程如下：

（1） 初始化：

通过输入信号 Reset 将输出 Q 和 CV 值复位为初始设定状态。当 Reset 为 TRUE，计数值 CV 被初始化为 0，Q 被初始化为 FALSE。此时，CU 输入信号不会触发计数，CV 保持初始值 0。

（2） 触发计数

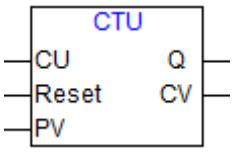
当 Reset 变为 FALSE，CU 信号从 FALSE 变为 TRUE 时，CV 从 0 开始加 1。

（3） 输出置位

当 CV 值递增到等于或大于 PV 设定值时，Q 输出被置为 TRUE,直到 Reset 信号将其复位为初始值 FALSE。

（4） 停止计数

当 CV 值递增到其数据类型上限 32767 时，CU 输入信号不再触发计数，CV 保持上限值，直到 Reset 信号将其复位为初始设定值。



CTU 指令示意图

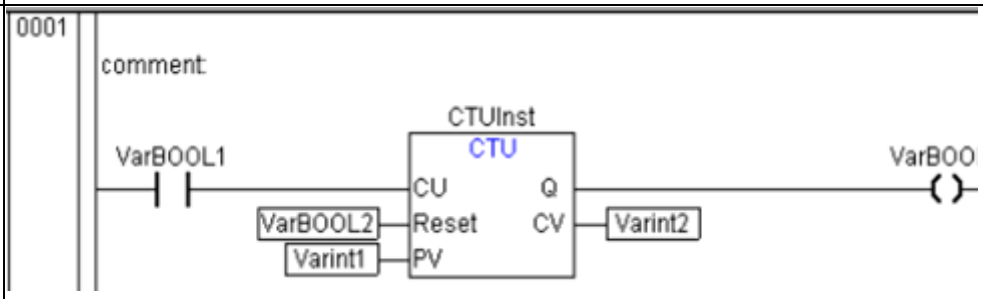
第2步 参数说明

参数类型	参数名称	数据类型	参数说明	初始值（默认）	掉电保护
输入参数	CU	BOOL	上升沿检测端	FALSE	FALSE
	Reset	BOOL	复位信号	FALSE	FALSE
	PV	INT	计数器设定值，用于置位输出信号	0	TRUE
输出参数	Q	BOOL	计数标志	TRUE	FALSE
	CV	INT	当前计数值	0	TRUE

第3步 示例

变量定义

序号	变量名	直接地址	变量说明	变量类型	初始值	掉电保护
0001	CTUInst			CTU		FALSE
0002	VarBOOL1			BOOL	FALSE	FALSE
0003	VarBOOL2			BOOL	FALSE	FALSE
0004	Varint1			INT	0	FALSE
0005	VarBOOL3			BOOL	FALSE	FALSE
0006	Varint2			INT	0	FALSE

编程语言	程序
梯形图（LD）	
结构化文本（ST）	CTUInst(CU:=VarBOOL1, Reset:=VarBOOL2, PV:=Varint1, Q=>VarBOOL3, CV=>Varint2);

5. CTU_DINT（递增计数器（DINT））

第1步 功能说明

递增计数器（CTU_DINT）实现当输入 CU 存在上升沿时,计数值递增功能。每检测到一个信号上升沿，计数值 CV 加 1，直到达到其数据类型上限 2,147,483,647。

递增计数过程如下：

（1） 初始化：

通过输入信号 Reset 将输出 Q 和 CV 值复位为初始设定状态。当 Reset 为 TRUE，计数值 CV 被初始化为 0，Q 被初始化为 FALSE。此时，CU 输入信号不会触发计数，CV 保持初始值 0。

（2） 触发计数

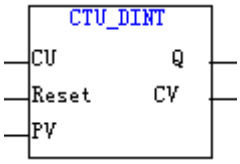
当 Reset 变为 FALSE，CU 信号从 FALSE 变为 TRUE 时，CV 从 0 开始加 1。

（3） 输出置位

当 CV 值递增到等于或大于 PV 设定值时，Q 输出被置为 TRUE,直到 Reset 信号将其复位为初始值 FALSE。

（4） 停止计数

当 CV 值递增到其数据类型上限 2,147,483,647 时，CU 输入信号不再触发计数，CV 保持上限值，直到 Reset 信号将其复位为初始设定值。



CTU_DINT 指令示意图

第2步 参数说明

参数类型	参数名称	数据类型	参数说明	初始值（默认）	掉电保护
输入参数	CU	BOOL	上升沿检测端	FALSE	FALSE
	Reset	BOOL	复位信号	FALSE	FALSE
	PV	DINT	计数器设定值，用于置位输出信号	0	TRUE
输出参数	Q	BOOL	计数标志	TRUE	FALSE
	CV	DINT	当前计数值	0	TRUE

第3步 示例

请参见 CTU（递增计数器（INT））。

6. CTU_UDINT（递增计数器（UDINT））

第1步 功能说明

递增计数器（CTU_UDINT）实现当输入 CU 存在上升沿时,计数值递增功能。每检测到一个信号上升沿，计数值 CV 加 1，直到达到其数据类型上限 4,294,967,295。

递增计数过程如下：

（1） 初始化：

通过输入信号 Reset 将输出 Q 和 CV 值复位为初始设定状态。当 Reset 为 TRUE，计数值 CV 被初始化为 0，Q 被初始化为 FALSE。此时，CU 输入信号不会触发计数，CV 保持初始值 0。

（2） 触发计数

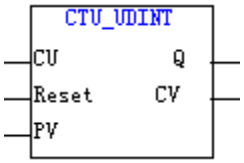
当 Reset 变为 FALSE，CU 信号从 FALSE 变为 TRUE 时，CV 从 0 开始加 1。

（3） 输出置位

当 CV 值递增到等于或大于 PV 设定值时，Q 输出被置为 TRUE,直到 Reset 信号将其复位为初始值 FALSE。

（4） 停止计数

当 CV 值递增到其数据类型上限 4,294,967,295 时，CU 输入信号不再触发计数，CV 保持上限值，直到 Reset 信号将其复位为初始设定值。



CTU_UDINT 指令示意图

第2步 参数说明

参数类型	参数名称	数据类型	参数说明	初始值（默认）	掉电保护
输入参数	CU	BOOL	上升沿检测端	FALSE	FALSE
	Reset	BOOL	复位信号	FALSE	FALSE
	PV	UDINT	计数器设定值，用于置位输出信号	0	TRUE
输出参数	Q	BOOL	计数标志	TRUE	FALSE
	CV	UDINT	当前计数值	0	TRUE

第3步 示例

请参见 CTU（递增计数器（INT））。

7. CTUD（递增递减计数器(INT)）

第1步 功能说明

递增递减计数器（CTUD）实现当输入存在上升沿时,计数值递增递减功能。当检测到 CU 信号上升沿，计数值 CV 加 1。当检测到 CD 信号上升沿，计数值 CV 减 1。递增或递减到其数据类型上限或下限值，计数值不再变化。

当同时检测到 CU 和 CD 上升沿时，计数值 CV 保值当前值不变。

递增计数过程如下：

（1） 初始化：

通过输入信号 Reset 将输出 Q 和 CV 值复位为初始设定状态。当 Reset 为 TRUE，计数值 CV 被初始化为 0，Q 被初始化为 FALSE。此时，CU/CD 输入信号不会触发计数，CV 保持初始值 0。

（2） 触发计数

当 Reset 变为 FALSE，CU 信号从 FALSE 变为 TRUE 时，CV 从 0 开始加 1。

（3） 输出置位

当 CV 值递增到等于或大于 PV 设定值时，Q 输出被置为 TRUE，直到 Reset 信号将其复位为初始值 FALSE。

（4） 停止计数

当 CV 值递增到其数据类型上限 32767 时，CU 输入信号不再触发计数，CV 保持上限值，直到 Reset 信号将其复位为初始设定值。

递减计数过程如下：

（1） 初始化：

通过输入信号 Load 将输出 Q 和 CV 值赋值为初始设定状态。当 Load 为 TRUE，计数值 CV 被赋值 PV 设定值，Q 被赋值为 FALSE。此时，CU/CD 输入信号不会触发计数，CV 保持当前 PV 值。

（2） 触发计数

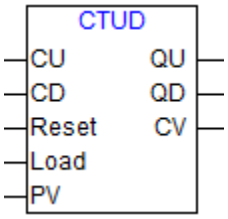
当 Load 变为 FALSE，CD 信号从 FALSE 变为 TRUE 时，CV 从 PV 设置值开始减 1。

（3） 输出置位

当 CV 值递减到 0 及 0 以下时，Q 输出被置为 TRUE,直到 Load 信号将其赋为初始值 FALSE。

（4） 停止计数

当 CV 值递减到其数据类型下限-32768 时，CD 输入信号不再触发计数，CV 保持下限值，直到 Load 信号将其赋值为初始设定值。



CTUD 指令示意图

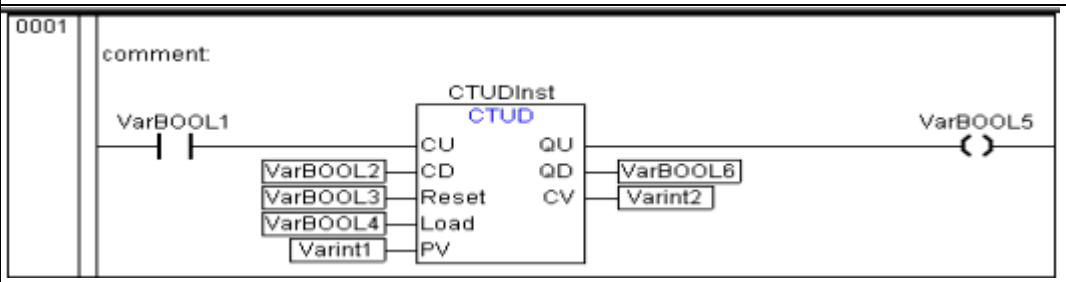
第2步 参数说明

参数类型	参数名称	数据类型	参数说明	初始值（默认）	掉电保护
输入参数	CU	BOOL	加计数器上升沿检测端	FALSE	FALSE
	CD	BOOL	减计数器上升沿检测端	FALSE	FALSE
	Reset	BOOL	加计数复位 CV 值	FALSE	FALSE
	Load	BOOL	减计数初始化 CV 值	FALSE	FALSE
	PV	INT	计数器设定值	0	TRUE
输出参数	QU	BOOL	加计数标志	TRUE	FALSE
	QD	BOOL	减计数标志	TRUE	FALSE
	CV	INT	当前计数值	0	TRUE

第3步 示例

变量定义

序号	变量名	直接地址	变量说明	变量类型	初始值	掉电保护
0001	CTUDInst			CTUD		FALSE
0002	VarBOOL1			BOOL	FALSE	FALSE
0003	VarBOOL2			BOOL	FALSE	FALSE
0004	VarBOOL3			BOOL	FALSE	FALSE
0005	VarBOOL4			BOOL	FALSE	FALSE
0006	VarBOOL5			BOOL	FALSE	FALSE
0007	VarBOOL6			BOOL	FALSE	FALSE
0008	Varint1			INT	0	FALSE
0009	Varint2			INT	0	FALSE

编程语言	程序
梯形图（LD）	
结构化文本（ST）	CTUDInst(CU:=VarBOOL1, CD:=VarBOOL2, Reset:=VarBOOL3, Load:=VarBOOL4, PV:=Varint1, QU=>VarBOOL5, QD=>VarBOOL6, CV=>Varint2);

8. CTUD_DINT（递增递减计数器（DINT））

第1步 功能说明

递增递减计数器（CTUD_DINT）实现当输入存在上升沿时,计数值递增递减功能。当检测到 CU 信号上升沿，计数值 CV 加 1。当检测到 CD 信号上升沿，计数值 CV 减 1。递增或递减到其数据类型上限或下限值，计数值不再变化。

当同时检测到 CU 和 CD 上升沿时，计数值 CV 保值当前值不变。

递增计数过程如下：

（1） 初始化：

通过输入信号 Reset 将输出 Q 和 CV 值复位为初始设定状态。当 Reset 为 TRUE，计数值 CV 被初始化为 0，Q 被初始化为 FALSE。此时，CU/CD 输入信号不会触发计数，CV 保持初始值 0。

（2） 触发计数

当 Reset 变为 FALSE，CU 信号从 FALSE 变为 TRUE 时，CV 从 0 开始加 1。

（3） 输出置位

当 CV 值递增到等于或大于 PV 设定值时，Q 输出被置为 TRUE,直到 Reset 信号将其复位为初始值 FALSE。

（4） 停止计数

当 CV 值递增到其数据类型上限 2,147,483,647 时，CU 输入信号不再触发计数，CV 保持上限值，直到 Reset 信号将其复位为初始设定值。

递减计数过程如下：

（1） 初始化：

通过输入信号 Load 将输出 Q 和 CV 值赋值为初始设定状态。当 Load 为 TRUE，计数值 CV 被赋值 PV 设定值，Q 被赋值为 FALSE。此时，CU/CD 输入信号不会触发计数，CV 保持当前 PV 值。

（2） 触发计数

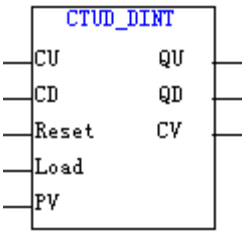
当 Load 变为 FALSE，CD 信号从 FALSE 变为 TRUE 时，CV 从 PV 设置值开始减 1。

（3） 输出置位

当 CV 值递减到 0 及 0 以下时，Q 输出被置为 TRUE，直到 Load 信号将其赋为初始值 FALSE。

（4） 停止计数

当 CV 值递减到其数据类型下限-2,147,483,648 时，CD 输入信号不再触发计数，CV 保持下限值，直到 Load 信号将其赋值为初始设定值。



CTUD_DINT 指令示意图

第2步 参数说明

参数类型	参数名称	数据类型	参数说明	初始值（默认）	掉电保护
输入参数	CU	BOOL	加计数器上升沿检测端	FALSE	FALSE
	CD	BOOL	减计数器上升沿检测端	FALSE	FALSE
	Reset	BOOL	加计数复位 CV 值	FALSE	FALSE
	Load	BOOL	减计数初始化 CV 值	FALSE	FALSE

	PV	DINT	计数器设定值	0	TRUE
输出参数	QU	BOOL	加计数标志	TRUE	FALSE
	QD	BOOL	减计数标志	TRUE	FALSE
	CV	DINT	当前计数值	0	TRUE

第3步 示例

请参见 CTUD（递增递减计数器（INT））。

9. CTUD_UDINT（递增递减计数器（UDINT））

第1步 功能说明

递增递减计数器（CTUD_UDINT）实现当输入存在上升沿时,计数值递增递减功能。当检测到 CU 信号上升沿，计数值 CV 加 1。当检测到 CD 信号上升沿，计数值 CV 减 1。递增或递减到其数据类型上限或下限值，计数值不再变化。

当同时检测到 CU 和 CD 上升沿时，计数值 CV 保值当前值不变。

递增计数过程如下：

（1） 初始化：

通过输入信号 Reset 将输出 Q 和 CV 值复位为初始设定状态。当 Reset 为 TRUE，计数值 CV 被初始化为 0，Q 被初始化为 FALSE。此时，CU/CD 输入信号不会触发计数，CV 保持初始值 0。

（2） 触发计数

当 Reset 变为 FALSE，CU 信号从 FALSE 变为 TRUE 时，CV 从 0 开始加 1。

（3） 输出置位

当 CV 值递增到等于或大于 PV 设定值时，Q 输出被置为 TRUE，直到 Reset 信号将其复位为初始值 FALSE。

（4） 停止计数

当 CV 值递增到其数据类型上限 4,294,967,295 时，CU 输入信号不再触发计数，CV 保持上限值，直到 Reset 信号将其复位为初始设定值。

递减计数过程如下：

（1） 初始化：

通过输入信号 Load 将输出 Q 和 CV 值赋值为初始设定状态。当 Load 为 TRUE，计数值 CV 被赋值 PV 设定值，Q 被赋值为 FALSE。此时，CU/CD 输入信号不会触发计数，CV 保持当前 PV 值。

（2） 触发计数

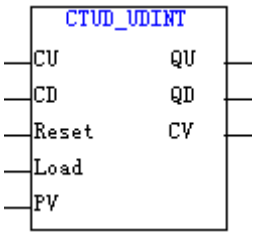
当 Load 变为 FALSE，CD 信号从 FALSE 变为 TRUE 时，CV 从 PV 设置值开始减 1。

（3） 输出置位

当 CV 值递减到 0 及 0 以下时，Q 输出被置为 TRUE，直到 Load 信号将其赋为初始值 FALSE。

（4） 停止计数

当 CV 值递减到其数据类型下限 0 时，CD 输入信号不再触发计数，CV 保持下限值，直到 Load 信号将其赋值为初始设定值。



CTUD 指令示意图

第2步 参数说明

参数类型	参数名称	数据类型	参数说明	初始值（默认）	掉电保护
输入参数	CU	BOOL	加计数器上升沿检测端	FALSE	FALSE
	CD	BOOL	减计数器上升沿检测端	FALSE	FALSE
	Reset	BOOL	加计数复位 CV 值	FALSE	FALSE
	Load	BOOL	减计数初始化 CV 值	FALSE	FALSE
	PV	UDINT	计数器设定值	0	TRUE
输出参数	QU	BOOL	加计数标志	TRUE	FALSE
	QD	BOOL	减计数标志	TRUE	FALSE
	CV	UDINT	当前计数值	0	TRUE

第3步 示例

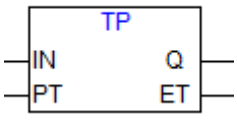
请参见 CTUD（递增递减计数器（INT））。

6.10.1.12 定时器指令

1. TP（普通定时器）

第1步 功能说明

当 IN 从 FALSE 变成 TRUE 时，Q 为 TRUE，ET 开始以毫秒计时，直到 ET 等于 PT 后 ET 保持常数。在 ET 开始计时后，IN 值无效。当 ET 等于 PT 后，若 IN 变为 FALSE,则 Q 为 FALSE，ET 为 0。



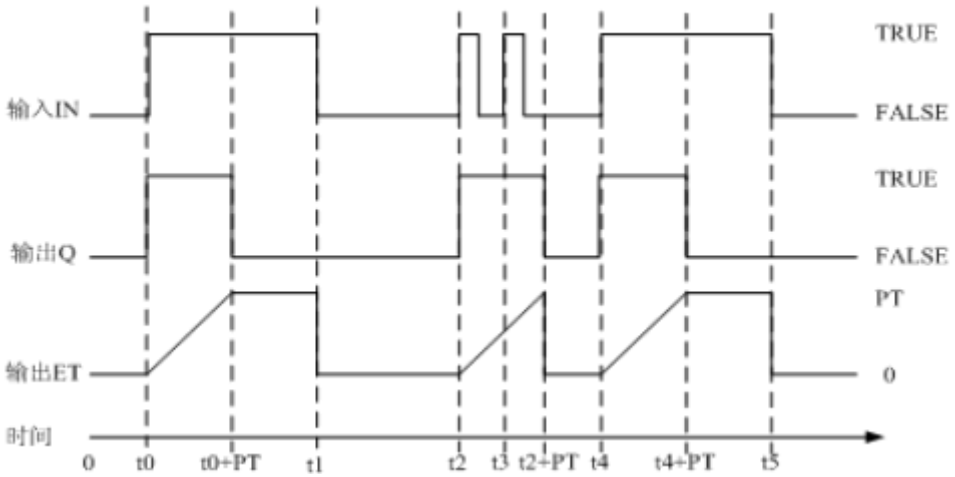
TP 功能块

第2步 参数说明

输入参数	数据类型	功能说明	参数值说明	默认值
IN	BOOL	定时器计时触发信号	当 IN 变成 TRUE 时，ET 以毫秒计时直到 ET 等于 PT，然后 ET 保持常数	FALSE
PT	TIME	定时时间值		0

输出参数	数据类型	功能说明	参数值说明	默认值
Q	BOOL	定时器输出	当 IN 为 TRUE 时，定时器开始工作，Q 为 TRUE，在 ET 小于等于 PT 前，IN 无效，ET 等于 PT 时，Q 为 FALSE	FALSE
ET	TIME	当前时间值	当 IN 变成 TRUE 时，ET 以毫秒计时直到 ET 等于 PT，然后 ET 保持常数。计时完毕后，当 IN 为 FALSE 时，ET 等于 0	0

第3步 时序图



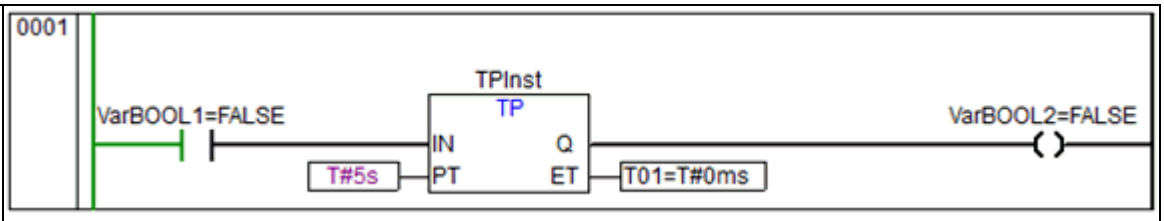
TP 时序图

第4步 示例

变量定义

序号	变量名	直接地址	变量说明	变量类型	初始值	掉电保护
0001	TPInst			TP		FALSE
0002	VarBOOL1			BOOL	FALSE	FALSE
0003	VarBOOL2			BOOL	FALSE	FALSE
0004	T01			TIME	T#0MS	FALSE

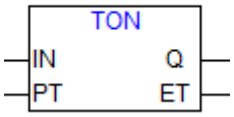
编程语言	程序

梯形图（LD）	
结构化文本（ST）	TPInst(IN:=VarBOOL1, PT:=T#5S, Q=>VarBOOL2, ET=>T01);

2. TON（通电延时定时器）

第1步 功能说明

当 IN 从 FALSE 变成 TRUE 时，ET 开始以毫秒计时，直到 ET 等于 PT 后 ET 保持常数，Q 变为 TRUE。ET 开始计时后，若 IN 变为 FALSE，则定时器停止计时，ET 变为 0。



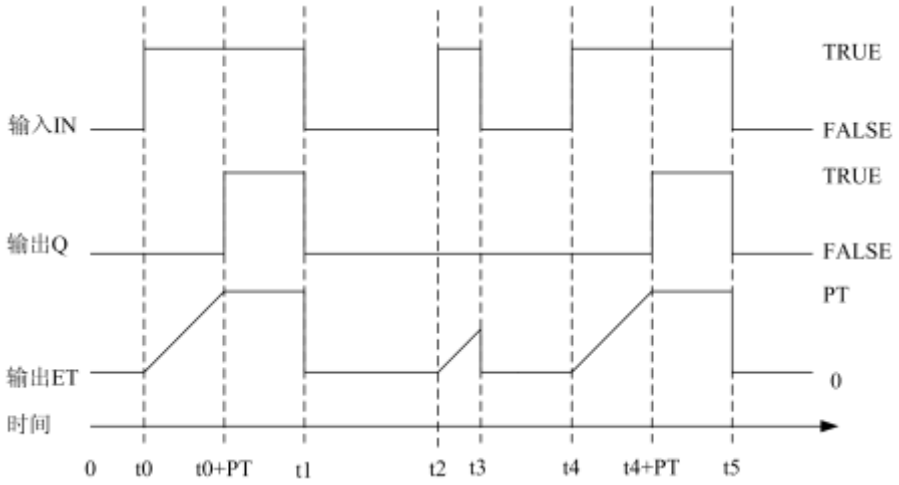
TON 功能块

第2步 参数说明

输入参数	数据类型	功能说明	参数值说明	默认值
IN	BOOL	定时器计时触发信号	当 IN 变成 TRUE 时，触发定时器开始计时。IN 为 FALSE 时停止计时	FALSE
PT	TIME	定时时间值		0

输出参数	数据类型	功能说明	参数值说明	默认值
Q	BOOL	定时器输出	当 IN 为 FALSE 时，Q 为 FALSE，ET 为 0。当 IN 为 TRUE 时，定时器开始工作，ET 等于 PT 时，Q 为 TRUE	FALSE
ET	TIME	当前时间值	当 IN 变成 TRUE 时，ET 以毫秒计时直到 ET 等于 PT，然后 ET 保持常数。无论何时，当 IN 为 FALSE 时，ET 等于 0	0

第3步 时序图

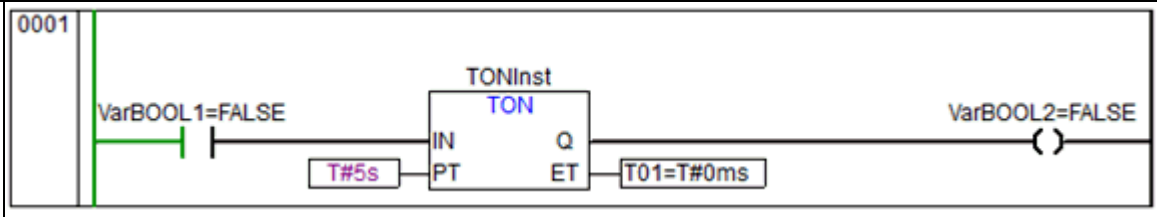


TON 时序图

第4步 示例

变量定义

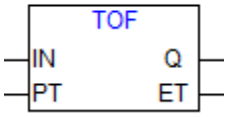
序号	变量名	直接地址	变量说明	变量类型	初始值	掉电保护
0001	TONInst			TON		FALSE
0002	VarBOOL1			BOOL	FALSE	FALSE
0003	VarBOOL2			BOOL	FALSE	FALSE
0004	T01			TIME	T#0MS	FALSE

编程语言	程序
梯形图（LD）	
结构化文本（ST）	TONInst(IN:=VarBOOL1, PT:=T#5S, Q=>VarBOOL2, ET=>T01);

3. TOF（断电延时定时器）

第1步 功能说明

当 IN 由 TRUE 变成 FALSE 时，ET 以毫秒计时直到 ET 等于 PT 时，Q 变为 FALSE，ET 保持常数。ET 开始计数后，若 IN 重新变为 TRUE 时，定时器停止计时，ET 变为 0，Q 保持值不变。



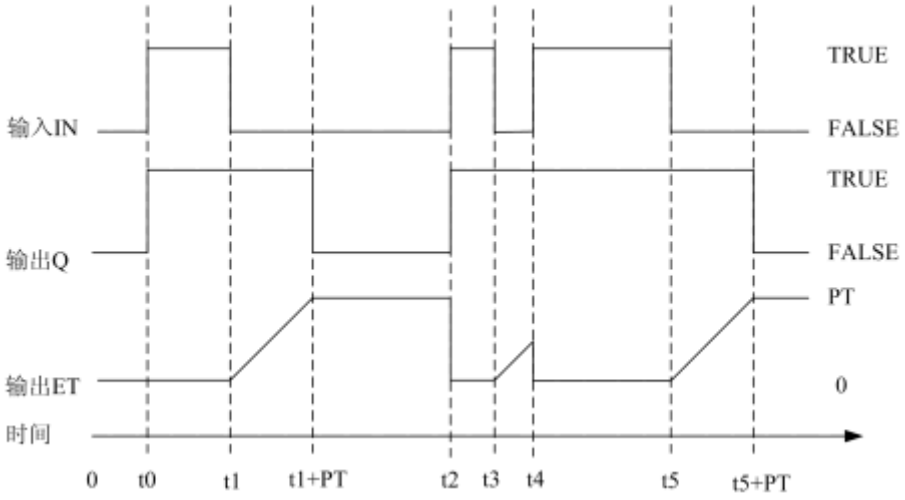
TOF 功能块

第2步 参数说明

输入参数	数据类型	功能说明	参数值说明	默认值
IN	BOOL	定时器计时触发信号	当 IN 由 TRUE 变成 FALSE 时，ET 以毫秒计时直到 ET 等于 PT，然后 ET 保持常数。Q 变为 FALSE	TRUE
PT	TIME	定时时间值		T#0MS

输出参数	数据类型	功能说明	参数值说明	默认值
Q	BOOL	定时器输出	当 IN 为 FALSE 且 ET 等于 PT 时，Q 为 FALSE。否则，Q 为 TRUE	TRUE
ET	TIME	当前时间值	当 IN 为 FALSE 时，ET 以毫秒计数直到 ET 等于 PT，然后 ET 保持常数。 无论何时，当 IN 为 TRUE 时，ET 等于 0，Q 为 TRUE	T#0MS

第3步 时序图

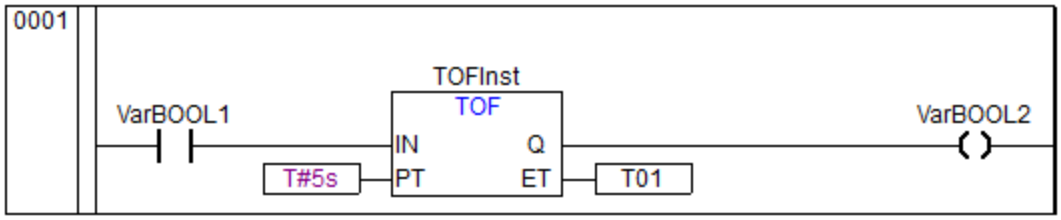


TOF 时序图

第4步 示例

变量定义

序号	变量名	直接地址	变量说明	变量类型	初始值	掉电保护
0001	TOFInst			TOF		FALSE
0002	VarBOOL1			BOOL	FALSE	FALSE
0003	VarBOOL2			BOOL	FALSE	FALSE
0004	T01			TIME	T#0MS	FALSE

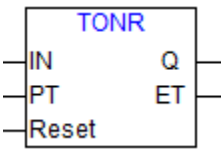
编程语言	程序
梯形图（LD）	
结构化文本（ST）	TOFInst(IN:=VarBOOL1, PT:=T#5S, Q=>VarBOOL2, ET=>T01);

4. TONR（保持型通电延时定时器）

第1步 功能说明

保持型通电延时定时器，可以累计输入信号的接通时间。

当 IN 变成 TRUE 时，ET 以毫秒计时直到 ET 等于 PT。如有外部触发导致在延时时间未达设定时间值前 IN 变成 FALSE，此时 ET 保持计时停止前的时间值，等到 IN 再次触发变成 TRUE 时，ET 继续计时，直到 ET 等于 PT，输出 Q 置位。RESET 为 1，则复位所有的输出值和中间变量，结束定时，为 FALSE 时，允许执行延时定时功能。



TONR 功能块

第2步 参数说明

输入参数	数据类型	功能说明	参数值说明	默认值
IN	BOOL	定时器计时触发信号	当 IN 变成 TRUE 时，定时器开始计时，IN 变为 FALSE 时，定时器停止计时	FALSE
PT	TIME	定时时间值	设定的延时时间值，请按照 TIME 数据类型的格式要求来设定， 单位可以是 d，h，m，s 和 ms，最小单位为 ms	0
Reset	BOOL	复位端	TRUE：复位所有的输出和中间变量，结束定时 FALSE：允许执行延时定时功能	FALSE

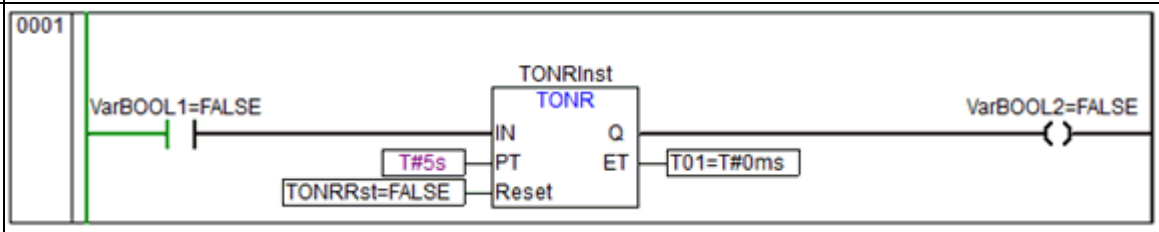
输出参数	数据类型	功能说明	参数值说明	默认值
Q	BOOL	定时器溢出标	当 Q 为 FALSE 时，说明 ET 未达设定值 PT。当 ET 等于 PT 时，Q 为	FALSE

		志	TRUE 计时开始时，无论 IN 为 TRUE 还是 FALSE，都不影响 Q 的输出，Q 的输出只取决于 ET 是否等于 PT 以及 RESET 端状态	
ET	TIME	当前时间值	当 IN 变成 TRUE 时，ET 以毫秒计时直到 ET 等于 PT 计时开始时，无论 IN 为 TRUE 还是 FALSE，只要 Q 为 FALSE，ET 会一直输出当前时间值，直到 ET 等于 PT	0

第3步 示例

变量定义

序号	变量名	直接地址	变量说明	变量类型	初始值	掉电保护
0001	TONRInst			TONR		FALSE
0002	VarBOOL1			BOOL	FALSE	FALSE
0003	VarBOOL2			BOOL	FALSE	FALSE
0004	TONRRst			BOOL	FALSE	FALSE
0005	T01			TIME	T#0MS	FALSE

编程语言	程序
梯形图（LD）	
结构化文本（ST）	TONRInst(IN:=VarBOOL1, PT:=T#5S, Reset:=TONRRst, Q=>VarBOOL2, ET=>T01);

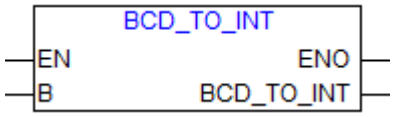
6.10.2 基础应用库

6.10.2.1 BCD 转换指令

1. BCD_TO_INT（BCD 码转整型）

第1步 功能说明

BCD 码转换为整型值算法（BCD_TO_INT）要求输入值是 BYTE 类型，而输出值则是 INT 类型。完成 BCD 码到整型值的转换。



BCD_TO_INT 功能块

第2步 参数说明

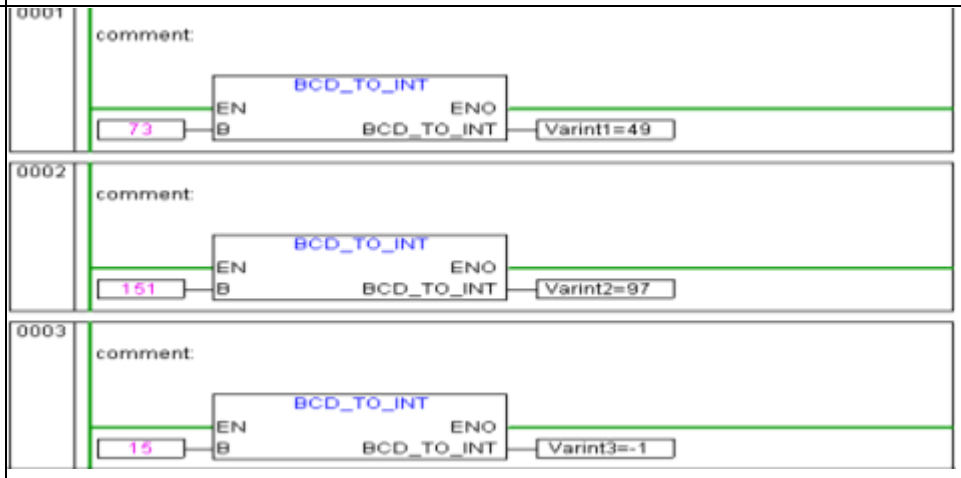
输入参数	数据类型	参数值说明	初始值（默认）
B	BYTE	需要转换的 BCD 码，输入 BCD 码的二进制形式（或者该二进制对应的十进制和十六进制），比如 BCD 码 49，表示为 2#100 1001（或者对应 10#73、16#49），则此处输入 2#100 1001（或者 10#73，16#49）	0

输出参数	数据类型	参数值说明	初始值（默认）
BCD_TO_INT	INT	转换后的 INT 值 如果输入的字节不是 BCD 码，输出是-1	0

第3步 指令使用举例

变量定义

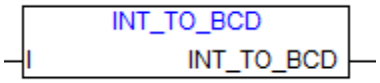
序号	变量名	直接地址	变量说明	变量类型	初始值	掉电保护
0001	Varint1			INT	0	FALSE
0002	Varint2			INT	0	FALSE
0003	Varint3			INT	0	FALSE

编程语言	程序
梯形图（LD）	
结构化文本（ST）	Varint1:=BCD_TO_INT(73); (*结果为 49*) Varint2:=BCD_TO_INT(151); (*结果为 97*) Varint3:=BCD_TO_INT(15); (*结果为-1，因为不是 BCD 码格式*)

2. INT_TO_BCD（整型转 BCD 码）

第1步 功能说明

整数值转换为 BCD 码算法（INT_TO_BCD）要求输入值是 INT 类型，而输出值则是 BYTE 类型。完成整型值到 BCD 码的转换。当整数值不能转换成 BCD 码字节时，输出数值 255。



INT_TO_BCD 功能块

第2步 参数说明

输入参数	数据类型	参数值说明	初始值（默认）
I	INT	需要转换的整型数据值 如果整数值为 49，则此处输入整型数据 49	0

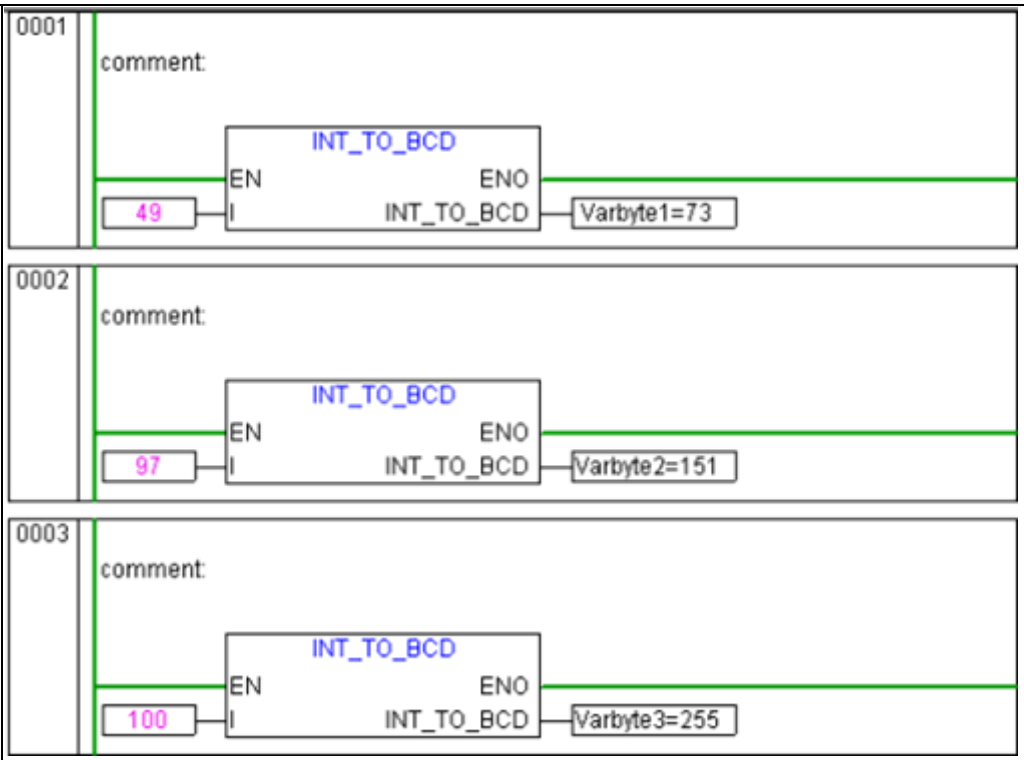
输出参数	数据类型	参数值说明	初始值（默认）
INT_TO_BCD	BYTE	转换后的 BCD 码 比如将 49 转换为 BCD 码为 2#100 1001，则此处输出 2#1001001（或者该二进制对应的 10#73、16#49）	0

第3步 指令使用举例

变量定义

序号	变量名	直接地址	变量说明	变量类型	初始值	掉电保护
0001	Varbyte1			BYTE	0	FALSE
0002	Varbyte2			BYTE	0	FALSE
0003	Varbyte3			BYTE	0	FALSE

编程语言	程序

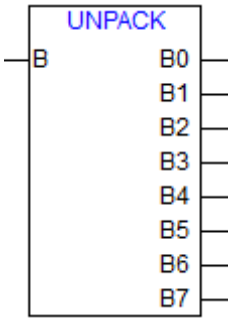
梯形图（LD）	
结构化文本（ST）	Varbyte1:=INT_TO_BCD(49); (*结果为 73*) Varbyte2:=INT_TO_BCD(97); (*结果为 151*) Varbyte2:=INT_TO_BCD(100); (*错误！输出：255*)

6.10.2.2 位_字节操作指令

1. UNPACK（位拆分指令）

第1步 功能说明

将输入 B 的第 7~0 位依次取出赋值给 B7~B0。与 PACK 指令相反。



UNPACK 功能块

第2步 参数说明

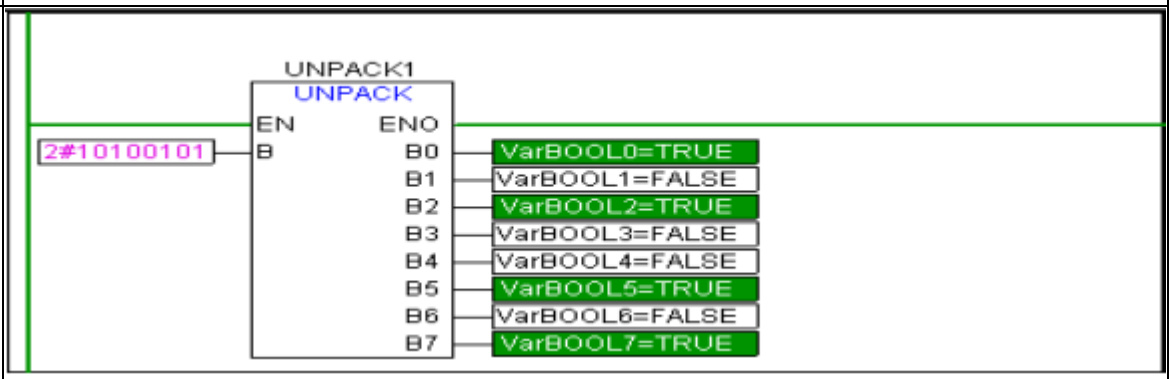
输入参数	说明	变量类型	初始值（默认）	掉电保护
B	BYTE 型输入变量	BYTE	0	FALSE

输出参数	说明	变量类型	初始值（默认）	掉电保护
B0	BOOL 型输出变量	BOOL	FALSE	FALSE
B1		BOOL	FALSE	FALSE
B2		BOOL	FALSE	FALSE
B3		BOOL	FALSE	FALSE
B4		BOOL	FALSE	FALSE
B5		BOOL	FALSE	FALSE
B6		BOOL	FALSE	FALSE
B7		BOOL	FALSE	FALSE

第3步 示例

变量定义

序号	变量名	直接地址	变量说明	变量类型	初始值	掉电保护
0001	UNPACK1			UNPACK		FALSE
0002	VarBOOL0			BOOL	FALSE	FALSE
0003	VarBOOL1			BOOL	FALSE	FALSE
0004	VarBOOL2			BOOL	FALSE	FALSE
0005	VarBOOL3			BOOL	FALSE	FALSE
0006	VarBOOL4			BOOL	FALSE	FALSE
0007	VarBOOL5			BOOL	FALSE	FALSE
0008	VarBOOL6			BOOL	FALSE	FALSE
0009	VarBOOL7			BOOL	FALSE	FALSE

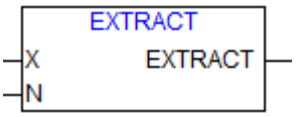
编程语言	程序
梯形图（LD）	
结构化文本（ST）	UNPACK1(B:=2#10101010, B0=>VarBOOL0, B1=>VarBOOL1,

	B2=>VarBOOL2, B3=>VarBOOL3, B4=>VarBOOL4, B5=>VarBOOL5, B6=>VarBOOL6, B7=>VarBOOL7);

2. EXTRACT（位提取指令）

第1步 功能说明

将输入变量 X 的指定第 N 位输出。



EXTRACT 功能块

第2步 参数说明

输入参数	说明	变量类型	初始值（默认）
X	被操作变量	DWORD	0
N	位置	BYTE	0

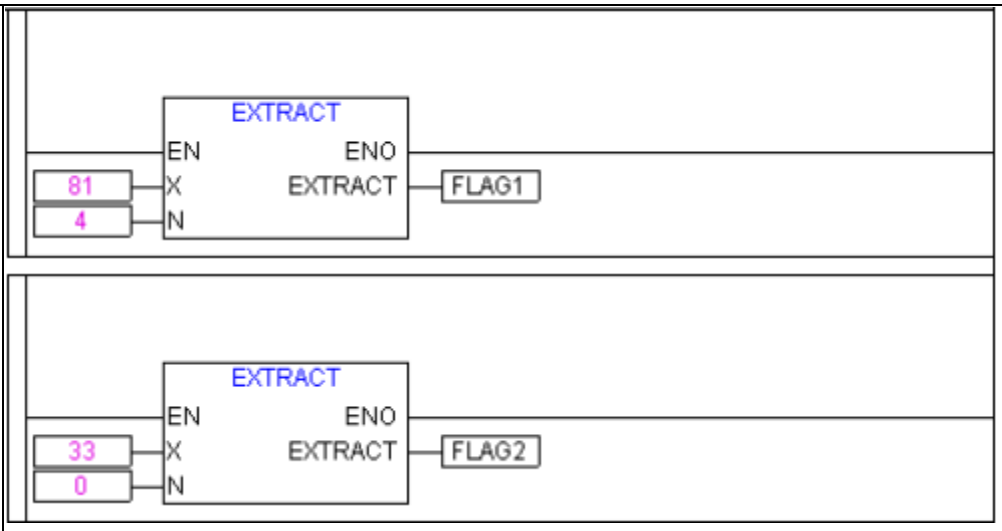
输出参数	说明	变量类型	初始值（默认）
EXTRACT	输出	BOOL	0

第3步 示例

变量定义

序号	变量名	直接地址	变量说明	变量类型	初始值	掉电保护
0001	FLAG1			BOOL	FALSE	FALSE
0002	FLAG2			BOOL	FALSE	FALSE

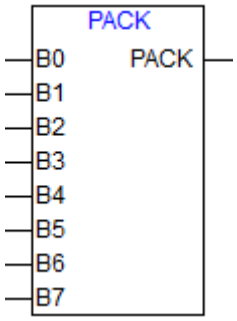
编程语言	程序

梯形图（LD）	 (*结果 FLAG1=TRUE，FLAG2=TRUE *)
结构化文本（ST）	FLAG1:= EXTRACT (81 ,4); (*结果：TRUE，因为 81 的二进制数是 1010001，所以第四位是 1*) FLAG2:= EXTRACT (33 ,0); (*结果：TRUE，因为 33 的二进制数是 100001，所以第 0 位是 1*)

3. PACK（位整合指令）

第1步 功能说明

将 B0~B7 按照从第 0 位到第 7 位的顺序组合成一个 BYTE 变量。与这个指令相对应的指令是 UNPACK。



PACK 功能块

第2步 参数说明

输入参数	变量类型	说明	初始值（默认）
B0	BOOL	第 0 位数据	0
B1	BOOL	第 1 位数据	0
B2	BOOL	第 2 位数据	0
B3	BOOL	第 3 位数据	0
B4	BOOL	第 4 位数据	0

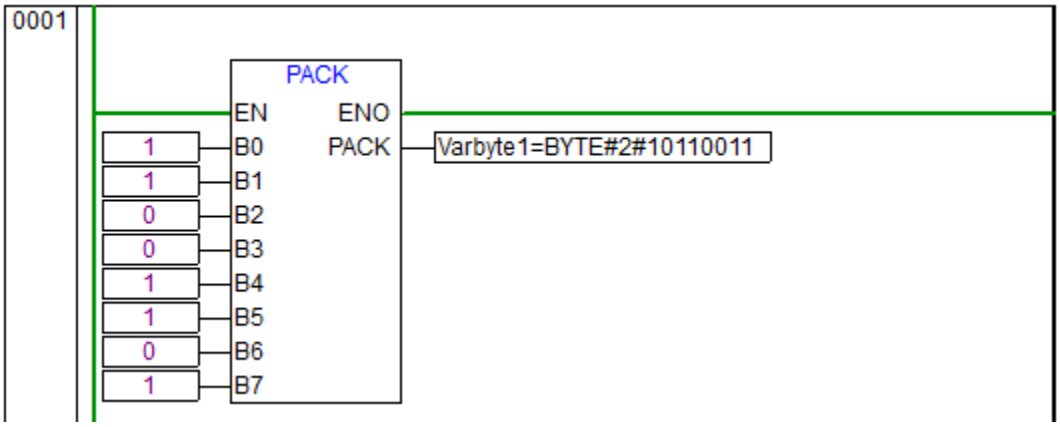
B5	BOOL	第 5 位数据	0
B6	BOOL	第 6 位数据	0
B7	BOOL	第 7 位数据	0

输出参数	变量类型	说明	初始值（默认）
Pack	BYTE	BYTE 字节由高到低：B7~B0	0

第3步 示例

变量定义

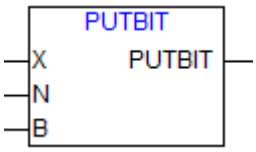
序号	变量名	直接地址	变量说明	变量类型	初始值	掉电保护
0001	Varbyte1			BYTE	0	FALSE

编程语言	程序
梯形图（LD）	
结构化文本（ST）	Varbyte1:= PACK(1,1,0,0,1,1,0,1); (*结果为 2#10110011*)

4. PUTBIT（位赋值命令）

第1步 功能说明

置变量指定位为指定值算法（PUTBIT）完成置输入值 X 指定的第 N 位为指定值 B。



PUTBIT 功能块

第2步 参数说明

输入参数	数据类型	说明	初始值（默认）
X	DWORD	被操作变量	0

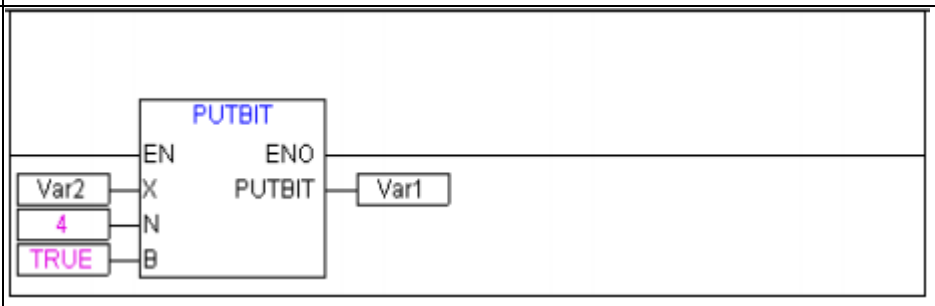
N	BYTE	位置	0
B	BOOL	指定值	FALSE

输出参数	数据类型	说明	初始值（默认）
PutBit	DWORD	输出完成指定值算法的变量值	0

第3步 示例

变量定义

序号	变量名	直接地址	变量说明	变量类型	初始值	掉电保护
0001	Var2			DWORD	0	FALSE
0002	Var1			DWORD	0	FALSE

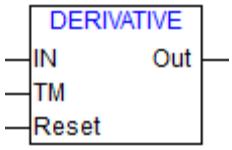
编程语言	程序
梯形图（LD）	
结构化文本（ST）	Var2:=38; (*二进制 100110*) Var1:=PUTBIT(Var2,4,TRUE); (*结果：54 = 2#110110*)

6.10.2.3 数学辅助功能

1. DERIVATIVE（微分）

第1步 功能说明

该指令对连续输入的变量进行微分运算。为了获得最好结果，DERIVATIVE 指令只对最新的四个输入值进行微分，减小因输入参数不精确产生的误差。



DERIVATIVE 功能块

第2步 微分迭代公式

$$OUT = \frac{3*[IN(k) - IN(k-3)] + IN(k-1) - IN(k-2)}{3*TM(k-2) + 4*TM(k-1) + 3*TM(k)} * 1000$$

k-3、k-2、k-1、k 为连续四次输入值的标记。

第3步 参数说明

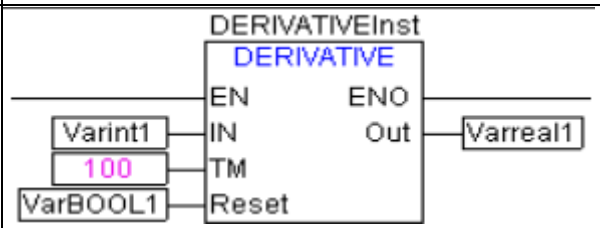
输入参数	数据类型	功能说明	初始值（默认）
IN	REAL	输入变量	0
TM	DWORD	时间常数	0
Reset	BOOL	重启信号	FALSE

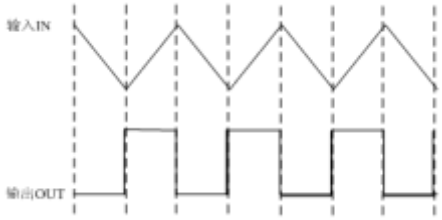
输出参数	数据类型	功能说明	初始值（默认）
OUT	REAL	微分算法的输出	0

第4步 示例

变量定义

序号	变量名	直接地址	变量说明	变量类型	初始值	掉电保护
0001	DERIVATVEInst			DERIVATIVE		FALSE
0002	Varint1			INT	0	FALSE
0003	VarBOOL1			BOOL	FALSE	FALSE
0004	Varreal1			REAL	0	FALSE

编程语言	程序
梯形图（LD）	
结构化文本（ST）	DERIVATIVEInst(IN:=Varint1, TM:=100, Reset:=VarBOOL1, Out=>Varreal1);

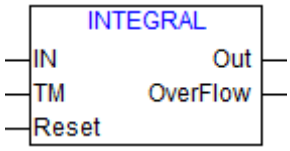


输入输出对应关系

2. INTEGRAL（积分）

第1步 功能说明

积分算法（INTEGRAL）近似求解输入函数的积分。



INTEGRAL 功能块

第2步 积分迭代公式

B(k)=B(k-1)+TM*IN(k)

OUT(k)=B(k)

k-1、k 为连续两次输入值的标记。

第3步 参数说明

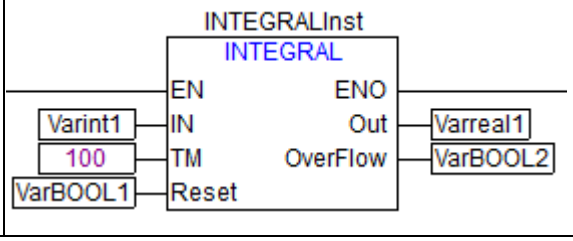
输入参数	数据类型	功能说明	初始值（默认）	掉电保护
IN	REAL	输入变量	0	FALSE
TM	DWORD	时间常数	0	TRUE
Reset	BOOL	重启信号	FALSE	TRUE

输出参数	数据类型	功能说明	初始值（默认）	掉电保护
Out	REAL	近似积分值	0	TRUE
OverFlow	BOOL	溢出标志	FLASE	TRUE

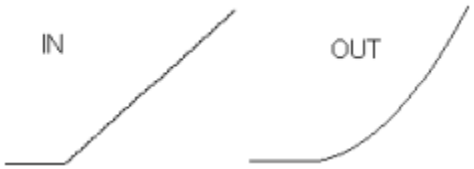
第4步 示例

变量定义

序号	变量名	直接地址	变量说明	变量类型	初始值	掉电保护
0001	INTEGRALInst			INTEGRAL		FALSE
0002	Varint1			INT	0	FALSE
0003	VarBOOL1			BOOL	FALSE	FALSE
0004	Varreal1			REAL	0	FALSE
0005	VarBOOL2			BOOL	FALSE	FALSE

编程语言	程序
梯形图（LD）	

结构化文本（ST）	INTEGRALInst(IN:=Varint1, TM:=100, Reset:=VarBOOL1, Out=>Varreal1, OverFlow=> VarBOOL2);

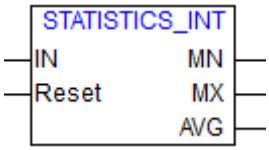


输入输出对应关系

3. STATISTICS_INT（整型统计）

第1步 功能说明

计算标准统计算法（STATISTICS_INT）统计所有 INT 型输入变量中的最小值、最大值和平均值。



STATISTICS_INT 功能块

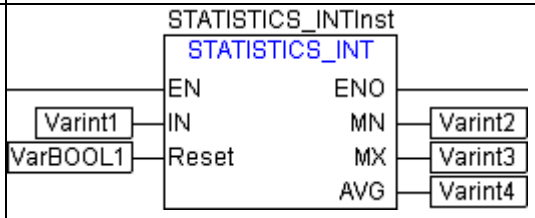
第2步 参数说明

输入参数	数据类型	功能说明	初始值（默认）
IN	INT	被操作变量	0
Reset	BOOL	初始化	FALSE

输出参数	数据类型	功能说明	初始值（默认）
MN	INT	最小值输出	32767
MX	INT	最大值输出	-32768
AVG	INT	平均值输出	0

变量定义

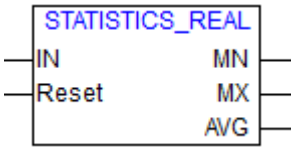
序号	变量名	直接地址	变量说明	变量类型	初始值	掉电保护
0001	STATISTICS_INTInst			STATISTICS_INT		FALSE
0002	Varint1			INT	0	FALSE
0003	VarBOOL1			BOOL	FALSE	FALSE
0004	Varint2			INT	0	FALSE
0005	Varint3			INT	0	FALSE
0006	Varint4			INT	0	FALSE

编程语言	程序
梯形图（LD）	
结构化文本（ST）	STATISTICS_INTInst(IN:= Varint1, Reset:= VarBOOL1, MN=>Varint2, MX=>Varint3, AVG=>Varint4);

4. STATISTICS_REAL（实型统计）

第1步 功能说明

统计所有 REAL 型输入变量中的最大值、最小值和平均值。



STATISTICS_ REAL 功能块

第2步 参数说明

输入参数	数据类型	功能说明	初始值（默认）
IN	REAL	被操作变量	0
Reset	BOOL	初始化	FALSE

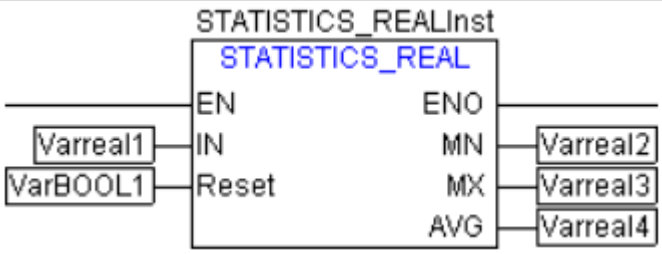
输出参数	数据类型	功能说明	初始值（默认）
MN	REAL	最小值输出	3E+38
MX	REAL	最大值输出	1E-37

AVG	REAL	平均值输出	0

第3步 示例

变量定义

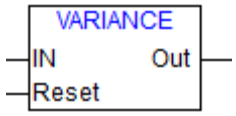
序号	变量名	直接地址	变量说明	变量类型	初始值	掉电保护
0001	STATISTICS_REALInst			STATISTICS_REAL		FALSE
0002	Varreal1			REAL	0	FALSE
0003	VarBOOL1			BOOL	FALSE	FALSE
0004	Varreal2			REAL	0	FALSE
0005	Varreal3			REAL	0	FALSE
0006	Varreal4			REAL	0	FALSE

编程语言	程序
梯形图（LD）	
结构化文本（ST）	STATISTICS_REALInst(IN:=Varreal1, Reset:=VarBOOL1, MN=>Varreal2, MX=>Varreal3, AVG=>Varreal4);

5. VARIANCE（平方偏差）

第1步 功能说明

计算输入值方差算法（VARIANCE）计算所有输入变量的方差。标准偏差可以由平方偏差的平方根得到。



VARIANCE 功能块

第2步 参数说明

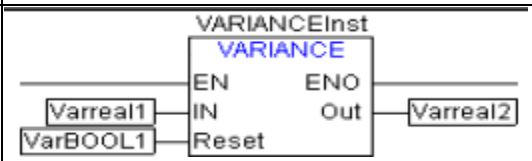
输入参数	数据类型	功能说明	初始值（默认）	掉电保护
IN	REAL	被操作变量	0	FALSE
Reset	BOOL	初始化	FALSE	TRUE

输出参数	数据类型	功能说明	初始值（默认）	掉电保护
Out	REAL	方差值输出	0	FALSE

第3步 示例

变量定义

序号	变量名	直接地址	变量说明	变量类型	初始值	掉电保护
0001	VARIANCEInst			VARIANCE		FALSE
0002	VarBOOL1			BOOL	FALSE	FALSE
0003	Varreal1			REAL	0	FALSE
0004	Varreal2			REAL	0	FALSE

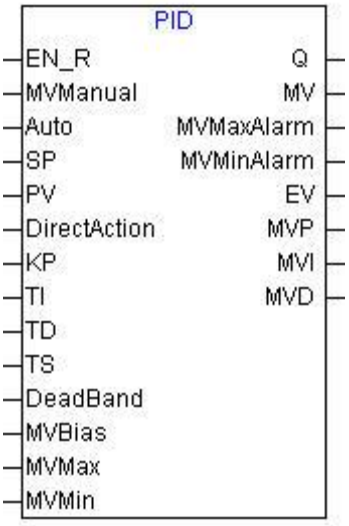
编程语言	程序
梯形图（LD）	
结构化文本（ST）	VARIANCEInst(IN:=Varreal1, Reset:=VarBOOL1, OUT=>Varreal2);

6.10.2.4 控制器指令

1. PID（比例积分微分控制器）

第1步 功能说明

比例积分微分控制器，对现场输入的过程值与设定值比较，对其进行 PID 调节控制。将调节的指令输出给现场设备从而达到及时响应的调节任务，同时对输出值进行限幅操作，以保证输出在许可的范围内操作，并对超限数值及时报警。



PID 功能块

1. 使能

   当使能端 EN_R 为真时，程序正常运行，否则程序返回不运算。

2. 判断设置值

   - 输出上下限 MVMax、MVMin 设置判断

     输出下限一定要小于输出上限，否则输出值超上下限标志均置位且程序返回不运算。

   - 积分时间 Ti、微分时间 Td 和死区范围 DeadBand 的设置判断

     积分时间 Ti、微分时间 Td 和死区范围 DeadBand 必须为非负数，否则按照初始值进行运算。

3. 计算

   - 手动模式

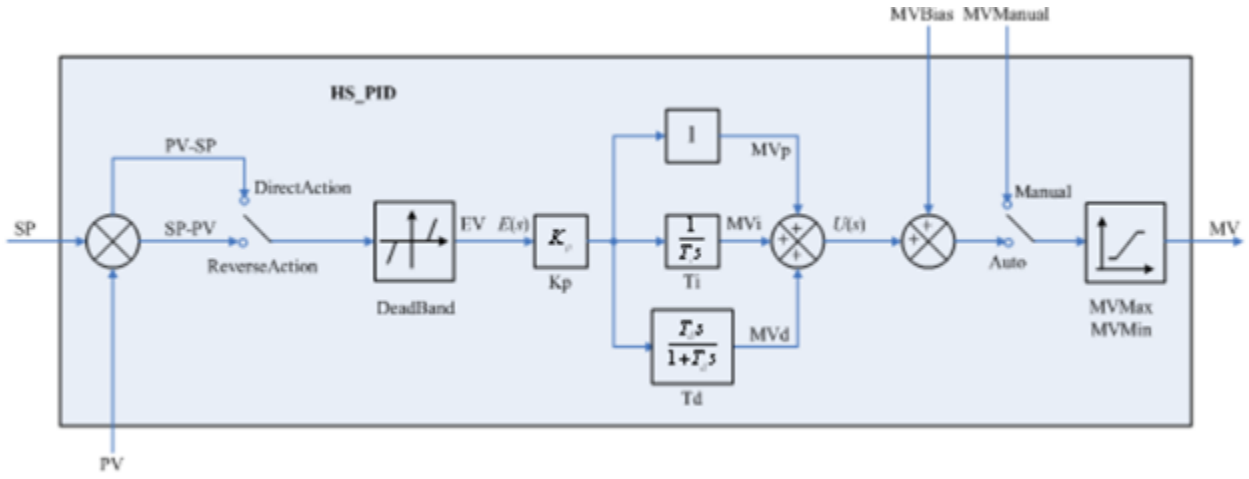
     输出 MV 等于手动输入值 MVManual，并且 SP 跟踪 PV。

   - 自动模式

     根据正反作用方式 DerectAction、设定值 SP、测量值 PV 和死区范围 DeadBand 计算偏差；再根据输出 MV、输出上下限值以及偏差的正负计算积分饱和系数。

     根据比例增益 Kp、积分时间 Ti、微分时间 Td 和积分饱和系数计算控制输出的比例分量 MVp、积分分量 MVi、微分分量 MVd 以及控制输出 MV。

     PID 功能块的逻辑结构如图所示。



PID 功能块逻辑结构图

第2步 参数说明

输入参数	数据类型	功能说明	参数值说明	默认值	掉电保护
EN_R	BOOL	使能	FALSE：无效 TRUE：上升沿设定、高电平有效	FALSE	TRUE
MVManual	REAL	手动输入值		0	TRUE
Auto	BOOL	“自动”模式选择	FALSE：手动 TRUE：自动	FALSE	TRUE
SP	REAL	设定值		0	TRUE
PV	REAL	过程值		0	FALSE
DirectAction	BOOL	“正作用”方式选择	FALSE：反作用（EV=SP-PV） TRUE：正作用（EV=PV-SP）	FALSE	TRUE
KP	REAL	比例增益		1	TRUE
TI	REAL	积分时间（S）	Ti>=0，如果积分时间为 0，表示没有积分环节	1	TRUE
TD	REAL	微分时间（S）	Td>=0，如果微分时间为 0，表示没有微分环节	0	TRUE
TS	REAL	运算周期（S）	Ts>0	0	TRUE
DeadBand	REAL	偏差死区限	（1）若 DeadBand=0，表示没有死区 （2）偏差绝对值大于该值时有效，否则偏差为 0	0	TRUE
MVBias	REAL	前馈控制量		0.0	TRUE
MVMax	REAL	输出值上限		+100	TRUE
MVMin	REAL	输出值下限		-100	TRUE

输出参数	数据类型	功能说明	参数值说明	默认值	掉电保护
Q	BOOL	使能状态标志	FALSE：无效	FALSE	TRUE

			TRUE：上升沿设定、高电平有效		
MV	REAL	控制量输出		0	TRUE
MVMaxAlarm	BOOL	输出值超上限报警	FALSE：未超上限 TRUE：已超上限	FALSE	TRUE
MVMinAlarm	BOOL	输出值超下限报警	FALSE：未超下限 TRUE：已超下限	FALSE	TRUE
EV	REAL	设定值与过程值的偏差	若有死区，EV 则为经死区处理后的偏差值	0	FALSE
MVP	REAL	比例分量		0	TRUE
MVI	REAL	积分分量		0	TRUE
MVD	REAL	微分分量		0	TRUE

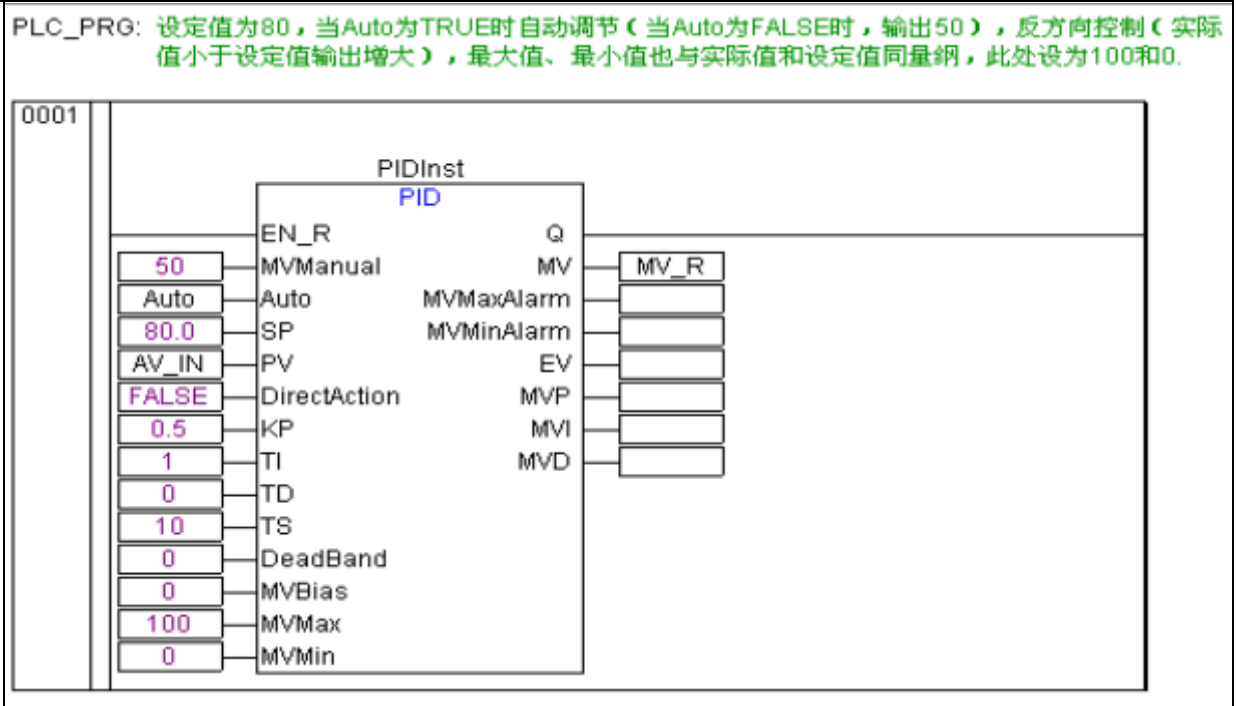
第3步 示例

变量定义

序号	变量名	直接地址	变量说明	变量类型	初始值	掉电保护
0001	PIDInst			PID		FALSE
0002	Auto			BOOL	FALSE	FALSE
0003	AV_IN			REAL	0	FALSE
0004	MV_R			REAL	0	FALSE

编程语言：程序

梯形图（LD）：



结构化文本：

```
PIDInst(
```

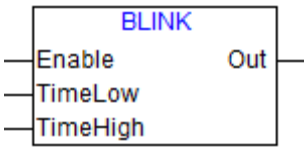
（ST）	EN_R:=TRUE, MVManual:=50, Auto:=Auto, SP:=80.0, PV:=AV_IN, DirectAction:=FALSE, KP:=0.5, TI:=1, TD:=0, TS:=10, DeadBand:=0, MVBias:=0, MVMax:=100.0, MVMin:=0, MV=>MV_R

6.10.2.5 信号发生器

1. BLINK（脉冲信号发生器）

第1步 功能说明

脉冲信号发生器算法（BLINK）完成按指定时间输出高电平和低电平的矩形波。



BLINK 功能块

第2步 参数说明

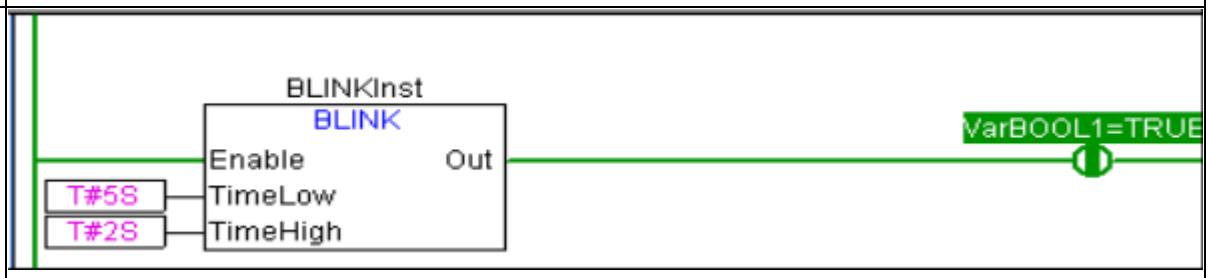
输入参数	说明	变量类型	初始值（默认）	掉电保护
Enable	使能端	BOOL	FALSE	TRUE
TimeLow	低电平时间	TIME	T#0S	TRUE
TimeHigh	高时平时间	TIME	T#0S	TRUE

输出参数	说明	变量类型	初始值（默认）	掉电保护
Out	输出	BOOL	FALSE	TRUE

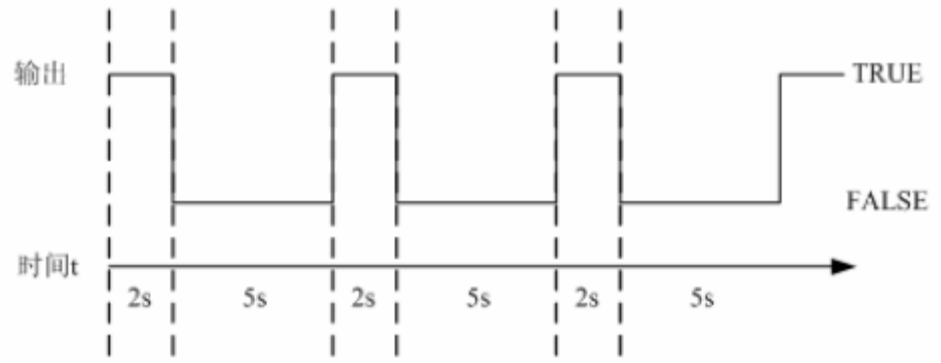
第3步 示例

变量定义

序号	变量名	直接地址	变量说明	变量类型	初始值	掉电保护
0001	BLINKInst			BLINK		FALSE
0002	VarBOOL1			BOOL	FALSE	FALSE

编程语言	程序
梯形图（LD）	
结构化文本（ST）	BLINKInst(Enable:=TRUE, TimeLow:=T#5S, TimeHigh:=T#2S, Out=>VarBOOL1);

当指令执行时，Out 所示波形输出如图所示。

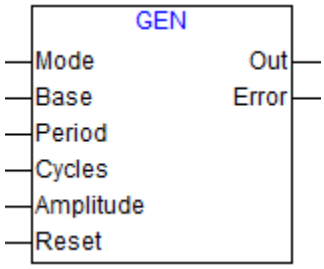


输出波形

2. GEN（典型周期信号发生器）

第1步 功能说明

产生典型周期函数算法（GEN）生成指定周期函数的类型、周期和振幅的信号。



GEN 功能块

- 初始化功能

  当 RESET=1 时，输出和相关计数变量清零。

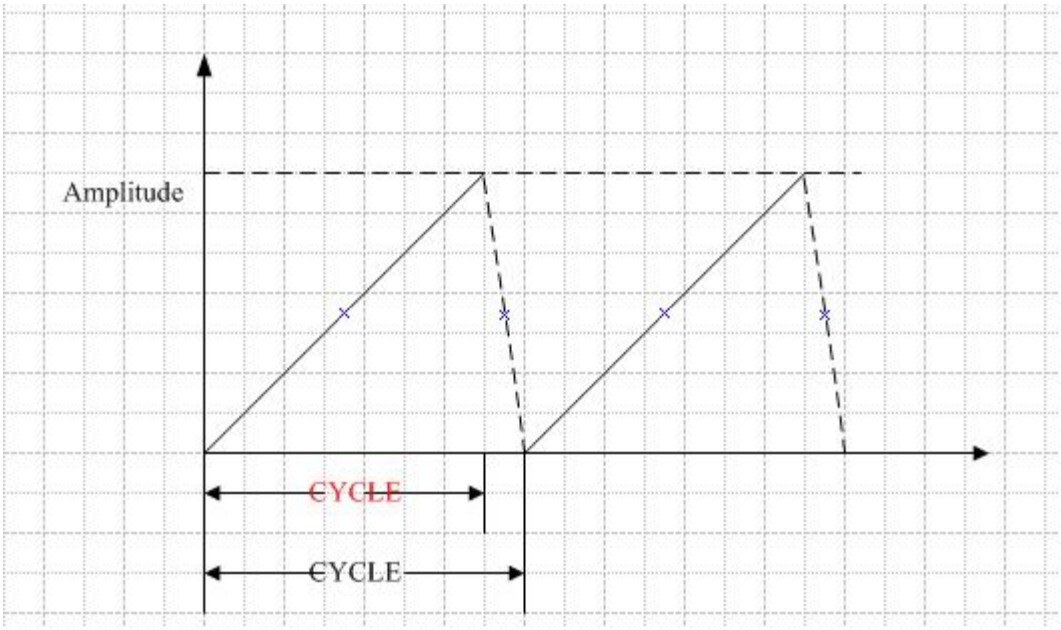
- 计算生成波形

  当 BASE=0 时，按照周期 CYCLES 和赋值 AMPLITUDE 生成 MODE 类型的波形。

  当 BASE=1 时，只是周期不同，周期变为 PERIOD。

- 主意事项

  - 当 BASE=0 且函数类型为 SAWTOOTH_RISE 或 SAWTOOTH_FALL 时，CYCLE 所示循环周期如下图黑色字体所示，而不是红色字体所示。（因为实际的信号产生中，不会有直上直下的波形，即一个时间点对应两个值的情况。）

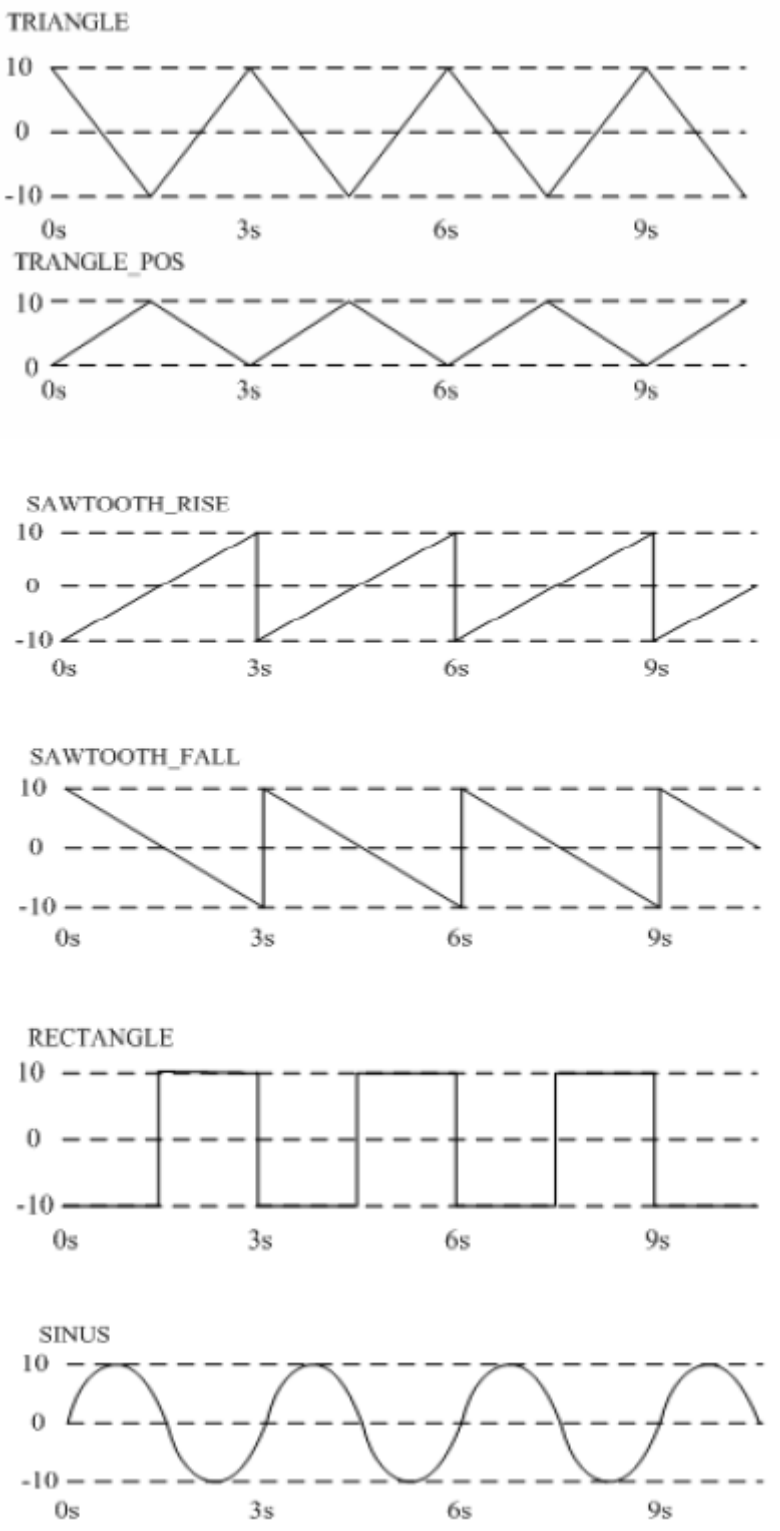


周期图

  - 当 BASE=1 且函数类型为 SINUS 或 COSINUS 时，如果 IEC 运算周期与波形循环周期 PERIOD 之比在 0.1 之内，可以实现全波形的采集；

    如果比值大于 0.1，有可能采集不到波峰波谷值，必然能采集到的最大值与最小值与比值大小相关。

    根据 Mode 处输入不同，产生的波形如下图所示。□


TRIANGLE
10
0
-10
0s
3s
6s
9s
TRANGLE_POS
10
0
0s
3s
6s
9s
SAWTOOTH_RISE
10
0
-10
0s
3s
6s
9s
SAWTOOTH_FALL
10
0
-10
0s
3s
6s
9s
RECTANGLE
10
0
-10
0s
3s
6s
9s
SINUS
10
0
-10
0s
3s
6s
9s

波形图

第2步 参数说明

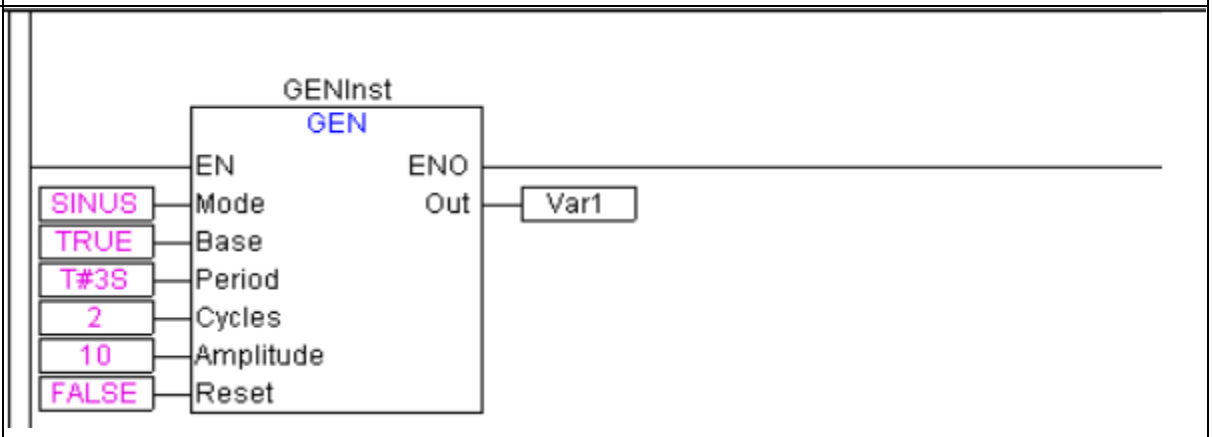
输入参数	变量类型	功能说明	参数值说明	初始值（默认）
Mode	GEN_MODE	指定要产生的信号类型	直接在 MODE 处输入 TRIANGLE、TRIANGLE_POS、SAWTOOTH_RISE、 SAWTOOTH_FALL、RECTANGLE、SINUS、COSINUS，则产生对应的波形	TRIANGLE
		TRIANGLE	三角波	
		TRIANGLE_POS	零起点三角波	
		SAWTOOTH_RISE	上升锯齿波	
		SAWTOOTH_FALL	下降锯齿波	
		RECTANGLE	方波	
		SINUS	正弦波	
		COSINU	余弦波	
Base	BOOL	循环方式选择	当 BASE 为 TRUE 时，信号发生器与定义的循环周期有关。当 BASE 为 FALSE 时， 信号发生器与特定的发生的个数有关	FALSE
Period	TIME	定义循环周期（Base 为真时有效）		T#1S
Cycles	INT	定义循环周期（Base 为假时有效）		1000
Amplitude	INT	信号的振幅		0
Reset	BOOL	初始化	当 RESET=TRUE 时，信号发生器被重新设置为 0	FALSE

输出变量	变量类型	功能说明	初始值（默认）
Out	INT	波形信号输出值	0
Error	BYTE	错误信息 1：没有初始 CYCLES 2：没有初始 PERIOD 3：CYCLES 被置为无效值 0 4：PERIOD 被置为无效值 0 5：Amplitude 为无效负值	0

第3步 示例

变量定义

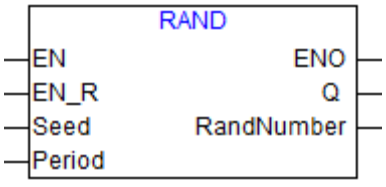
序号	变量名	直接地址	变量说明	变量类型	初始值	掉电保护
0001	GENInst			GEN		FALSE
0002	Var1			INT	0	FALSE

编程语言	程序
梯形图（LD）	
结构化文本（ST）	GENInst(Mode:=SINUS, Base:=TRUE, Period:=T#3S, Cycles:=2, Amplitude:=10, Reset:=FALSE, Out=>Var1);

3. RAND（随机数发生器）

第1步 功能说明

产生随机数。



RAND 功能块

第2步 参数说明

输入参数	数据类型	功能说明	参数值说明
EN_R	BOOL	使能	0：无效，停止产生随机数，RandNumber 保持最后一次产生的随机数不变 1：上升沿使能，且高电平有效
Seed	UINT	种子	种子是指计算随机序列的初始数值，取值范围为：0~65535 为保证随机数产生的持续性，seed 值只有在以下场景下才生效：全下装、控制器复位、AT 执行“冷复位”、“复位”

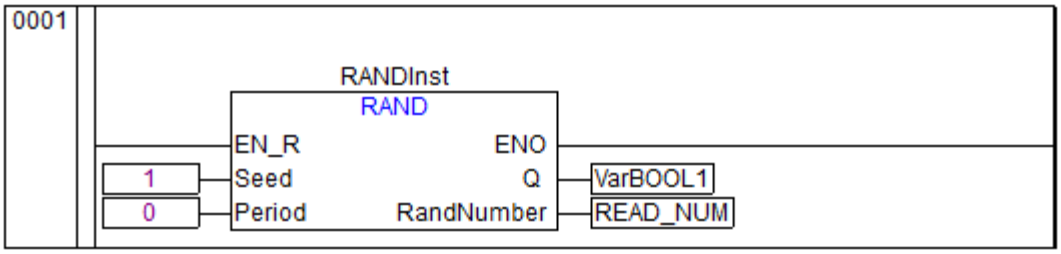
Period	UDINT	产生随机数的周期，单位为毫秒（ms）	0：每个扫描周期产生一个随机数 非 0：如设置为 200，则每 200ms 产生一个随机数 取值范围为：0~4294967295

输出参数	数据类型	功能说明	参数值说明
Q	BOOL	是否有效数据	0：无效数据 1：有效数据
RandNumber	UINT	随机数	0-65535

第3步 示例

变量定义

序号	变量名	直接地址	变量说明	变量类型	初始值	掉电保护
0001	RANDInst			RAND		FALSE
0002	READ_NUM			UINT	0	FALSE
0003	VarBOOL1			BOOL	FALSE	FALSE

编程语言	程序
梯形图（LD）	
结构化文本（ST）	RANDInst(EN_R:=TRUE, Seed:=1, Period:=0, Q=>VarBOOL1, RandNumber=>READ_NUM);

程序说明：

EN_R 置位时，产生随机数，每个周期产生 1 个随机数。

EN_R 复位时，停止产生随机数，RandNumber 保持最后一次产生的随机数不变。

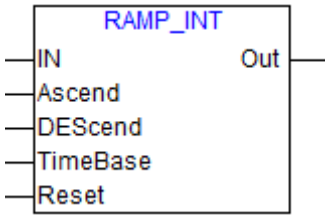
6.10.2.6 模拟量处理指令

1. RAMP_INT（整型限速）

第1步 功能说明

限制整型输入函数的升降速度。

输出 OUT 是输入 IN 经上升和下降限制整定后的函数值。当 TIMEBASE 设置为 t#0s，ASCEND 和 DESCEND 与时间间隔无关；当 TIMEBASE 设置为 t#0s 之外的值时，ASCEND 和 DESCEND 是时间间隔 TIMEBASE 时间长短的限制值。



RAMP_INT 功能块

第2步 参数说明

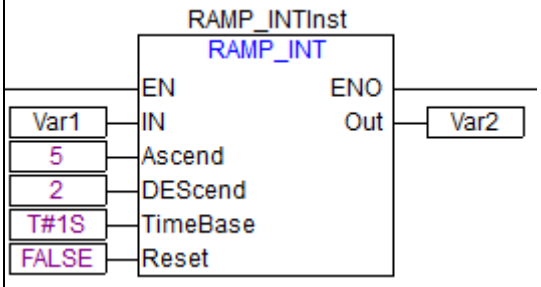
输入参数	数据类型	功能说明	初始值（默认）	掉电保护
IN	INT	被操作变量	0	FALSE
Ascend	INT	时间间隔内最大上升值	0	TRUE
DEScend	INT	时间间隔内最大下降值	0	TRUE
TimeBase	TIME	时间间隔	T#0S	TRUE
Reset	BOOL	重启信号	FALSE	TRUE

输出参数	数据类型	功能说明	初始值（默认）	掉电保护
Out	INT	经上升和下降限制后的值	0	TRUE

第3步 示例

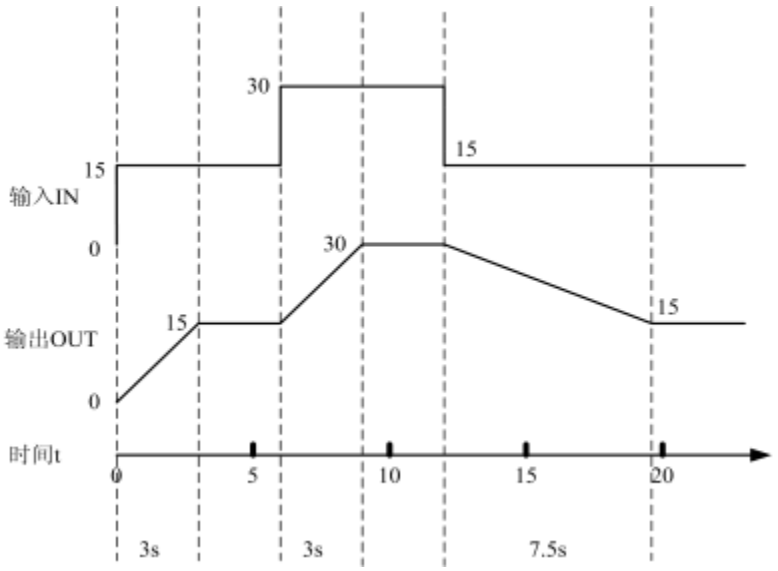
变量定义

序号	变量名	直接地址	变量说明	变量类型	初始值	掉电保护
0001	RAMP_INTInst			RAMP_INT		FALSE
0002	Var1			INT	0	FALSE
0003	Var2			INT	0	FALSE

编程语言	程序
梯形图（LD）	

结构化文本（ST）	RAMP_INTInst(IN:=Var1, Ascend:=5, DEScend:=2, TimeBase:=T#1S, Reset:=FALSE, OUT=>Var2);

上例中，当指令执行时，根据输入值 IN 的变化，对应的输出值如图所示。



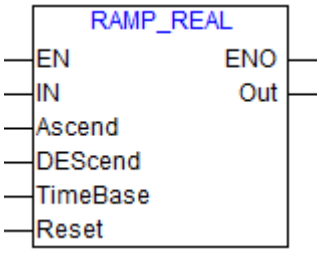
IN 与 OUT 对应关系

2. RAMP_REAL（实型限速）

第1步 功能说明

RAMP_REAL 指令可以限制实型输入函数的升降速度。

输出 OUT 是输入 IN 经上升和下降限制整定后的函数值。当 TIMEBASE 设置为 t#0s，ASCEND 和 DESCEND 与时间间隔无关；当 TIMEBASE 设置为 t#0s 之外的值时，ASCEND 和 DESCEND 是时间间隔 TIMEBASE 时间长短的限制值。



RAMP_REAL 功能块

第2步 参数说明

输入参数	数据类型	功能说明	初始值（默认）	掉电保护
IN	REAL	被操作变量	0	FALSE
Ascend	REAL	时间间隔内最大上升值	0	TRUE
DEScend	REAL	时间间隔内最大下降值	0	TRUE
TimeBase	TIME	时间间隔	T#0S	TRUE
Reset	BOOL	重启信号	FALSE	TRUE

输出参数	数据类型	功能说明	初始值（默认）	掉电保护
Out	REAL	经上升和下降限制后的值	0	TRUE

示例请参见 RAMP_INT（整型限速）。

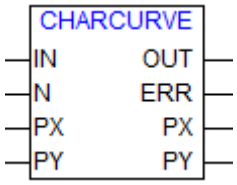
3. CHARCURVE（特征曲线）

第1步 功能说明

IN 是 DINT 类型，输入要操作的值。N 是字节类型，决定了特征曲线的点数且 2=<N<=11。在 DINT 类型数组 PX[0..10]和 PY[0..10]中产生一个特征行。DINT 型输出 OUT 为操作后的值。字节型变量 ERR 输出错误信息。

数组中的点 PX[0]..PX[N-1]必须根据从小到大存储，否则 ERR 的值为 1。如果 IN 不在 PX[0]和 PX[N-1]之间，ERR=2，输出包含相应的限制值 PY[0]或 PY[N-1]。

如果 N 超出了 2 和 11 之间的允许值，ERR=4。



CHARCURVE 指令示意图

第2步 参数说明

输入参数	数据类型	功能说明	初始值	掉电保护
IN	DINT	被操作变量	0	FALSE
N	BYTE	特征曲线的点数	0	TRUE

输出参数	数据类型	功能说明	初始值	掉电保护
OUT	DINT	操作后的值	0	TRUE
ERR	BYTE	错误信息	0	FALSE

其它参数	数据类型	功能说明	初始值	Retain
PX	ARRAY[0..10] OF DINT	特征曲线的 X 轴点列	0	TRUE
PY	ARRAY[0..10] OF DINT	特征曲线的 Y 轴点列	0	TRUE

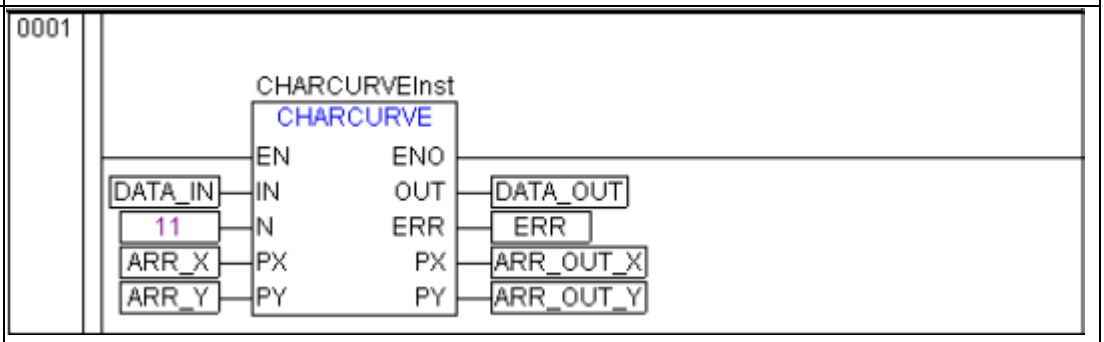
第3步 示例

变量定义

序号	变量名	直接地址	变量说明	变量类型	初始值	掉电保护
0001	CHARCURVEInst			CHARCURVE		FALSE
0002	ERR			BYTE	0	FALSE
0003	DATA_IN			INT	0	FALSE
0004	DATA_OUT			INT	0	FALSE
0005	ARR_X			ARRAY[0..10] OF DINT		FALSE
0006	ARR_Y			ARRAY[0..10] OF DINT		FALSE
0007	ARR_OUT_X			ARRAY[0..10] OF DINT		FALSE
0008	ARR_OUT_Y			ARRAY[0..10] OF DINT		FALSE

编程语言：程序

梯形图（LD）



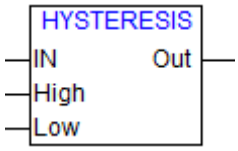
结构化文本（ST）

```
CHARCURVEInst(
IN:=DATA_IN,
N:=11,
PX:=ARR_X,
PY:=ARR_Y,
OUT=>DATA_OUT,
ERR=>ERR);
ARR_OUT_X:= CHARCURVEInst.PX;
ARR_OUT_Y:= CHARCURVEInst.PY;
```

4. HYSTERESIS（滞后）

第1步 功能说明

当输入值低于低限值时，输出为 TRUE；当输入值高于高限值时，输出为 FALSE；当输入值在高低限制之间时，输出保持。



HYSTERESIS 功能块

第2步 参数说明

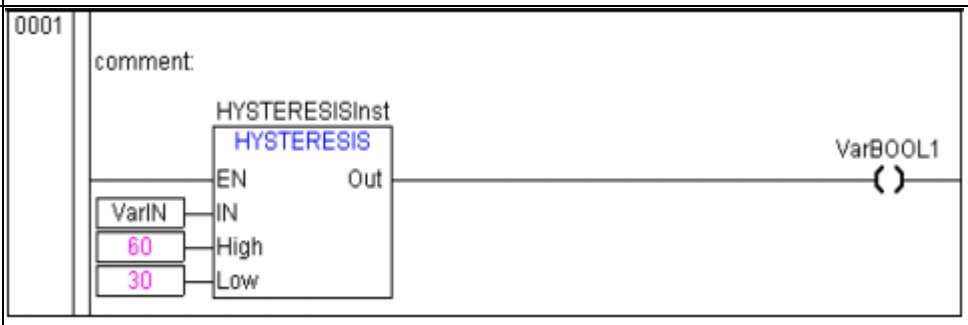
输入参数	说明	变量类型	初始值（默认）	掉电保护
IN	输入值	INT	0	FALSE
Low	低限值	INT	0	TRUE
High	高限值	INT	0	TRUE

输出参数	说明	变量类型	初始值（默认）	掉电保护
Out	输出，低下限值后为 TRUE，直到上限值后恢复为 FALSE	BOOL	FALSE	TRUE

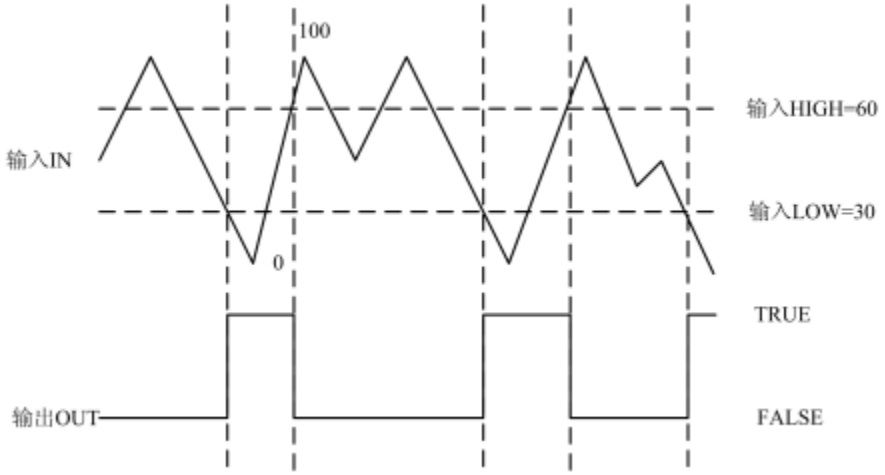
第3步 示例

变量定义

序号	变量名	直接地址	变量说明	变量类型	初始值	掉电保护
0001	HYSTERESISInst			HYSTERESIS		FALSE
0002	VarIN			INT	0	FALSE
0003	VarBOOL1			BOOL	FALSE	FALSE

编程语言	程序
梯形图（LD）	
结构化文本（ST）	HYSTERESISInst(IN:=VarIN, High:=60, Low:=30, Out=>VarBOOL1);

上例中，当指令执行时，根据输入值的变化，对应的输出值如图所示。

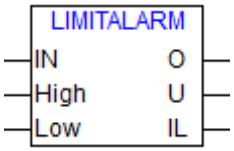


输入与输出变化关系

5. LIMITALARM（上下限报警）

第1步 功能说明

当输入值低于低限值时，输出 U 为 TRUE；当输入值高于高限值时，输出 O 为 TRUE；当输入值在高低限制之间时，输出 IL 为 TRUE。



LIMITALARM

第2步 参数说明

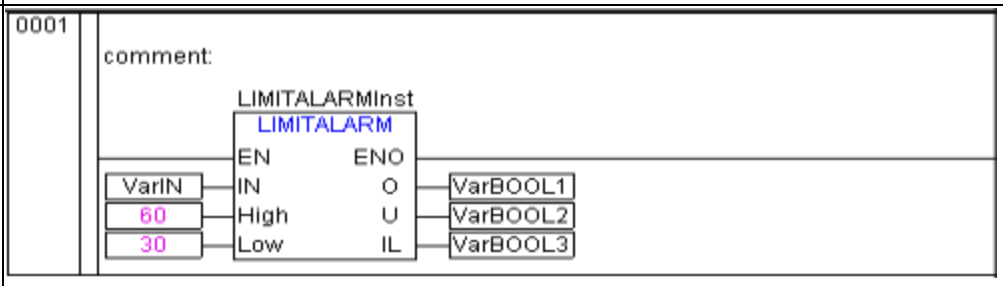
输入参数	说明	变量类型	初始值（默认）	掉电保护
IN	输入值	INT	0	FALSE
Low	低限值	INT	0	TRUE
High	高限值	INT	0	TRUE

输出参数	说明	变量类型	初始值（默认）	掉电保护
O	输入大于高限值	BOOL	FALSE	FALSE
U	输入小于低限值	BOOL	FALSE	FALSE
IL	输入在高低限之间	BOOL	FALSE	FALSE

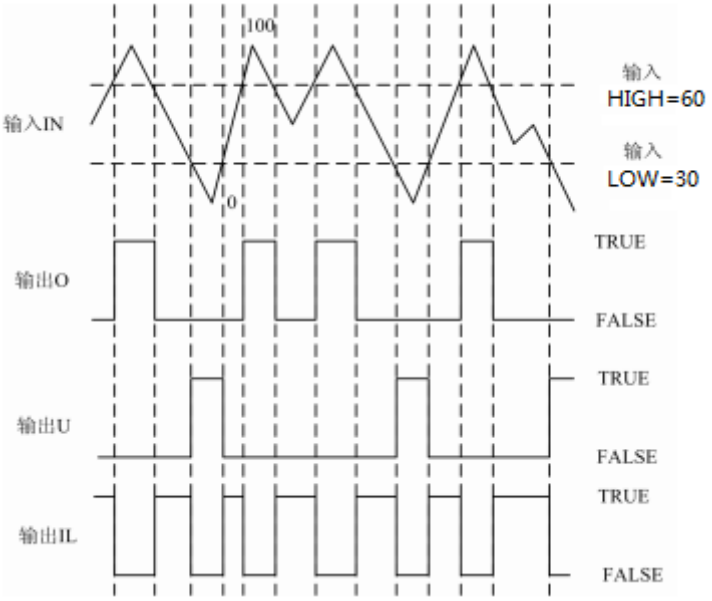
第3步 示例

变量定义

序号	变量名	直接地址	变量说明	变量类型	初始值	掉电保护
0001	VarIN			INT	0	FALSE
0002	VarBOOL1			BOOL	FALSE	FALSE
0003	VarBOOL2			BOOL	FALSE	FALSE
0004	VarBOOL3			BOOL	FALSE	FALSE

编程语言	程序
梯形图（LD）	
结构化文本（ST）	LIMITALARMInst(IN:=Var1 High:=60, Low:=30, O=>VarBOOL1, U=>VarBOOL2, IL=>VarBOOL3);

上例中，当指令执行时，根据输入值的变化，对应的输出值如图所示。

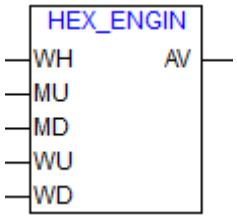


输入与输出变化关系

6. HEX_ENGIN（16 进制数转换为工程量数据）

第1步 功能说明

该指令一般用于模拟量输入数据处理，将输入的码值 WH 转换成对应的工程量 AV 值。



HEX_ENGIN 功能块

第2步 参数说明

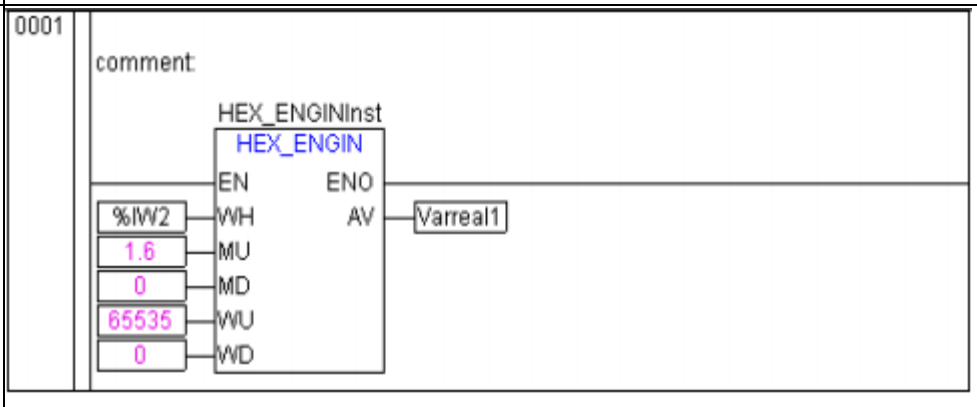
输入参数	说明	变量类型	初始值（默认）	掉电保护
WH	输入码值	DINT	0	FALSE
MU	工程量上限值	REAL	0	TRUE
MD	工程量下限值	REAL	0	TRUE
WU	PLC 的模拟量输入码值上限	DINT	0	TRUE
WD	PLC 的模拟量输入码值下限	DINT	0	TRUE

输出参数	说明	变量类型	初始值（默认）	掉电保护
AV	转换后的工程量	REAL	0	FALSE

第3步 示例

变量定义

序号	变量名	直接地址	变量说明	变量类型	初始值	掉电保护
0001	HEX_ENGINInst			HEX_ENGIN		FALSE
0002	Varreal1			REAL	0	FALSE

编程语言	程序
梯形图（LD）	
结构化文本（ST）	HEX_ENGINInst(WH:=%IW2,

	MU:=1.6, MD:=0, WU:=65535, WD:=0, AV=>Varreal1);

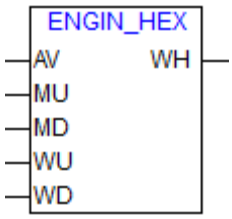
程序说明：

EN 置位时，进行转换，输入数据为%IW2，工程量上限是 1.6MPa，工程量下限是 0Mpa，对应模块码值范围为 0-65535。假设当%IW2 的值为 32767 时，对应输出值为 0.799。

7. ENGIN_HEX（工程量数据转换为 16 进制数据（WORD））

第1步 功能说明

该指令一般用于模拟量输出数据处理，将输入的工程量 AV 转换成相应的码值 WH。码值数据类型为 WORD。



ENGIN_HEX 功能块

第2步 参数说明

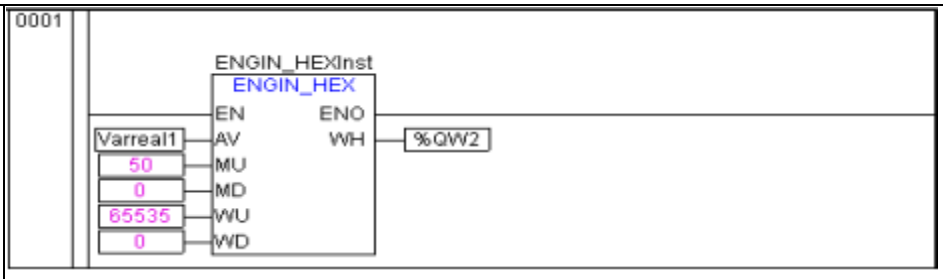
参数类型	参数名称	参数说明	变量类型	初始值（默认）	掉电保护
输入参数	AV	工程量当前值	REAL	0	FALSE
	MU	工程量上限值	REAL	0	TRUE
	MD	工程量下限值	REAL	0	TRUE
	WU	PLC 的模拟量输入码值上限，最大可设置为 65535	WORD	0	TRUE
	WD	PLC 的模拟量输入码值下限，最小可设置为 0	WORD	0	TRUE
输出参数	WH	转换后的码值	WORD	0	FALSE

第3步 示例

变量定义

序号	变量名	直接地址	变量说明	变量类型	初始值	掉电保护
0001	ENGIN_HEXInst			ENGIN_HEX		FALSE
0002	Varreal1			REAL	0	FALSE

编程语言	程序

梯形图（LD）	
结构化文本（ST）	ENGIN_HEXInst(AV:=Varreal1, MU:=50 MD:=0, WU:=65535, WD:=0, WH=>%QW2);

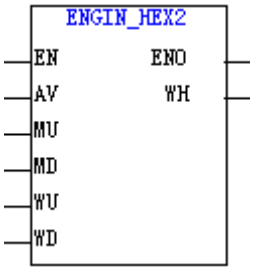
程序说明：

EN 置位时，进行转换，工程量实际数据为 Varreal1，工程量上限是 50Hz，工程量下限是 0Hz，对应模块范围为 0~65535，当 Varreal1 的值为 25Hz 时，对应输出值为 32768。将 32768 赋值给%QW2，则该通道模拟量输出值为 25Hz。

8. ENGIN_HEX2（工程量数据转换为 16 进制数据(INT)）

第1步 功能说明

该指令一般用于模拟量输出数据处理，将输入的工程量 AV 转换成相应的码值 WH。码值数据类型为 INT。



ENGIN_HEX2 功能块

第2步 参数说明

参数类型	参数名称	参数说明	变量类型	初始值（默认）	掉电保护
输入参数	AV	工程量当前值	REAL	0	FALSE
	MU	工程量上限值	REAL	0	TRUE
	MD	工程量下限值	REAL	0	TRUE
	WU	PLC 的模拟量输入码值上限，最大可设置为 32767	INT	0	TRUE
	WD	PLC 的模拟量输入码值下限，最小可设置为-32768	INT	0	TRUE

输出参数	WH	转换后的码值	INT	0	FALSE

第3步 示例

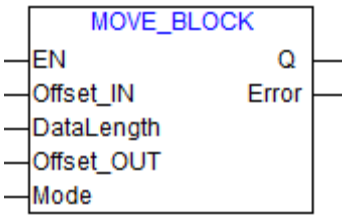
请参见 ENGIN_HEX（工程量数据转换为 16 进制数据(WORD)）。

6.10.2.7 数据块传送指令

1. MOVE_BLOCK（数据块传送）

第1步 功能说明

将 M 区数据从一个位置批量传送到 M 区的另一个位置。支持按字节（BYTE）、字（WORD）或双字（DWORD）批量传送。一次传送数据个数的最大值为 255 个双字，传送数据的长度由传送方式和传送数据的长度共同决定。



MOVE_BLOCK 功能块

第2步 参数说明

输入参数	数据类型	功能说明	参数值说明	默认值
EN	BOOL	使能	0:无效 1：使能	
Offset_In	DWORD	数据源区的偏移字地址	数据区的偏移字地址	0
DataLength	BYTE	传送数据的长度	0~255	0
Offset_Out	DWORD	数据目的区的偏移字地址	数据区的偏移字地址	0
Mode	BYTE	数据传送的方式	=0：表示字节传送； =1：表示字传送； =2：表示双字传送；	0

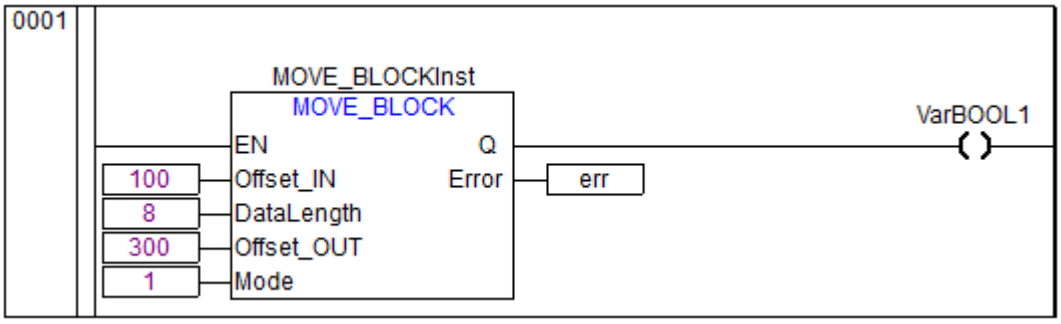
输出参数	数据类型	功能说明	参数值说明
Q	BOOL	使能输出	当数据未能传送/传送未正确时，Q=FALSE 当 数据传送正确时，Q=TRUE
Error	BYTE	错误信息	=0：数据传送正确 =1：数据源数据区偏移地址（Offset_IN）有误 =2：数据目的数据区偏移地址（Offset_OUT）有误

			=3：数据传送方式（Mode）有误

第3步 示例

变量定义

序号	变量名	直接地址	变量说明	变量类型	初始值	掉电保护
0001	MOVE_BLOCKInst			MOVE_BLOCK		FALSE
0002	err			BYTE	0	FALSE
0003	VarBOOL1			BOOL	FALSE	FALSE

编程语言	程序
梯形图（LD）	
结构化文本（ST）	MOVE_BLOCK(EN:=TRUE , Offset_IN:=100 , DataLength:=8 , Offset_OUT:=300 , Mode:=1 ,Q=>VarBOOL1 Error=> err);

程序说明：

EN 置位时，将 Offset_IN 所指向的 N 个字节的数据传送给 Offset_OUT 所指的目的地址。

6.10.3 系统库

6.10.3.1 时间相关指令

1. TimeGetTickCnt(获取开机计数)

第1步 函数原型

UDINT TimeGetTickCnt (void);

第2步 功能

获取控制器自开机以来的时间。时间范围为 0~2^{32} 毫秒，若超过上限，则从 0 开始计数。

第3步 参数说明

参数类型	参数名称	数据类型	参数说明
返回值	TimeGetTickCnt	UDINT	控制器开机以来所经过的时间，溢出后从零重新计时（ms）

2. TimeGetMs（获取 ms 级时间刻度值）

第1步 函数原型

UDINT TimeGetMs (void);

第2步 功能说明

获取到从开机开始的毫秒时间。

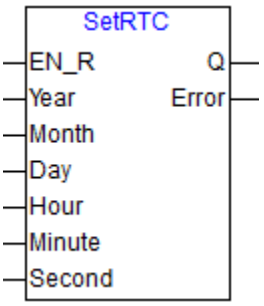
第3步 参数说明

参数类型	参数名称	数据类型	参数说明
返回值	TimeGetMs	UDINT	时间刻度值（单位为 ms）

3. SetRTC（设置实时时钟）

第1步 功能说明

将用户规定的时间设置到 PLC 实时时钟中。



SetRTC 功能块

第2步 参数说明

参数类型	输入参数	数据类型	参数说明	默认值
输入参数	EN_R	BOOL	使能 0：无效 1：上升沿有效	0
	Year	WORD	年，1971~2036	2011
	Month	BYTE	月，1-12	1
	Day	BYTE	日，1-31	1
	Hour	BYTE	时，0-23	0
	Minute	BYTE	分，0-59	0
	Second	BYTE	秒，0-59	0
输出参数	Q	BOOL	设置完成状态 0：未完成 1：已完成	

	Error	BYTE	参数设置错误 =0：设置 RTC 时间没有错误 非 0：表示设置 RTC 时间时出现错误 =1：设置年超出规定范围 =2：设置月不合理 =3：设置日不合理 =4：设置时不合理 =5：设置分不合理 =6：设置秒不合理 =200：模块重复定义并调用	

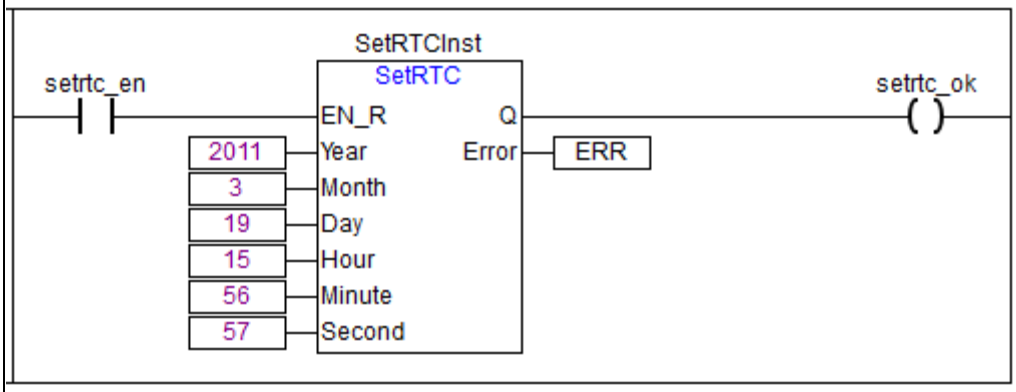
第3步 指令使用举例

变量定义

序号	变量名	直接地址	变量说明	变量类型	初始值	掉电保护
0001	setrtc_en			BOOL	FALSE	FALSE
0002	setrtc_ok			BOOL	FALSE	FALSE
0003	ERR			BYTE	0	FALSE
0004	SetRTCInst			SETRTC		FALSE

编程语言 | **程序**

梯形图（LD）



结构化文本（ST）

```
SETRTCInst(
EN_R:=set_rtc_en ,
Year:=2011 ,
Month:=3 ,
Day:= 19,
Hour:= 15,
Minute:= 56,
Second:=57 ,
Q=>set_rtc_ok
Error=>ERR );
```

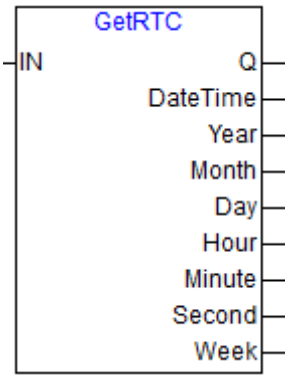
程序说明:

EN_R 置位时，如图所示的日期、时间将被设置到 PLC 硬件实时时钟之中，当前的实时时钟是 2011 年 3 月 19 日 15 点 56 分 57 秒。

4. GetRTC（读取实时时钟）

第1步 功能说明

该功能块读取 PLC 硬件实时时钟中的时间。



GetRTC 功能块

第2步 参数说明

参数类型	参数名称	数据类型	参数说明	默认值
输入参数	IN	BOOL	使能 0：无效 1：有效	0
输出参数	Q	BOOL	是否读出数据 0：未读出数据 1：正确取出数据	0
	DateTime	DT	日期/时间 DT#1971-01-01-00:00:00~ DT#2036-12-31-23:59:59	DT#1970-01-01-00:00:00
	Year	WORD	年，1971~2036	0
	Month	BYTE	月，1-12	0
	Day	BYTE	日，1-31	0
	Hour	BYTE	时，0-23	0
	Minute	BYTE	分，0-59	0
	Second	BYTE	秒，0-59	0
	Week	BYTE	星期，1-7	0

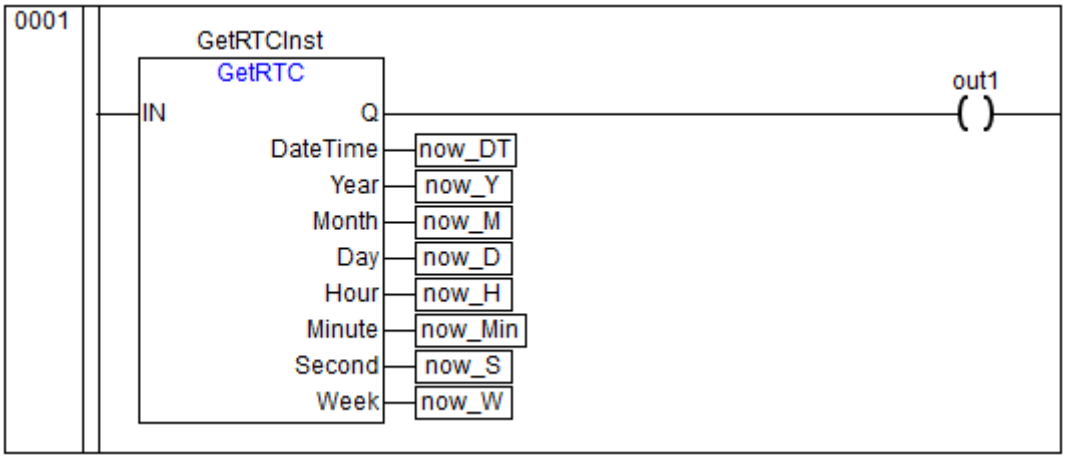
第3步 指令使用举例

变量定义

序号	变量名	直接地址	变量说明	变量类型	初始值	掉电保护
001	now_DT			DT	DT#1970-01-01-00:00:00	FALSE
002	now_Y			WORD	0	FALSE
003	now_M			BYTE	0	FALSE
004	now_D			BYTE	0	FALSE
005	now_H			BYTE	0	FALSE
006	now_Min			BYTE	0	FALSE
007	now_S			BYTE	0	FALSE
008	now_W			BYTE	0	FALSE
009	GetRTCInst			GETRTC		FALSE
0010	out1			BOOL	FALSE	FALSE

编程语言 | **程序**

梯形图（LD）



结构化文本（ST）

```
GETRTC(
IN:=TRUE ,
Q=>out1 ,
DateTime=>now_DT ,
Year=>now_Y ,
Month=>now_M ,
Day=>now_D ,
Hour=>now_H ,
Minute=>now_Min ,
Second=>now_S ,
Week=>now_W );
```

程序说明：

IN 置位时，DateTime 以 DT 型数据格式输出 PLC 实时时钟的日期、时间，Year 以 WORD 型数据格式输出年，Month、Date、Hour、Minute、Second、Week 分别以 BYTE 型数据格式输出月、日、时、分、秒、星期，Q 等于 TRUE，当前的实时时钟是：2012-03-19-15:56:57，星期一。

EN 复位时，Q 等于 FALSE，DateTime、Year、Month 、Date、Hour、Minute、Second、Week 保持最后一次输出值不变。

5. sysChangeUTCtime（读取实时时钟）

第1步 功能说明

UTC 时间转换成任意时区时间。

第2步 参数说明

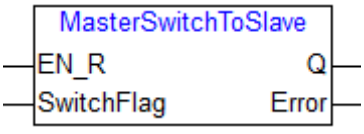
参数类型	输入参数	数据类型	参数说明	默认值
输入参数	IN	BOOL	使能 0：无效 1：上升沿有效	0
	DateTime	DWORD	输入的秒值	0
	MinutesWest	DINT	时区输入	0
输出参数	Q	BOOL	设置完成状态 0：未完成 1：已完成	0
	Year	WORD	年，1971~2036	0
	Month	BYTE	月，1-12	0
	Day	BYTE	日，1-31	0
	Hour	BYTE	时，0-23	0
	Minute	BYTE	分，0-59	0
	Second	BYTE	秒，0-59	0
	Week	BYTE	星期，1-7	0

6.10.3.2 冗余相关

1. MasterSwitchToSlave（主从切换）

第1步 功能说明

将主机切换为从机。



MasterSwitchToSlave 指令

第2步 参数说明

输入参数	数据类型	参数说明
EN_R	BOOL	使能 FALSE：无效（默认值） TRUE：上升沿有效
SwitchFlag	BOOL	FALSE：:不允许切换 TRUE：允许切换

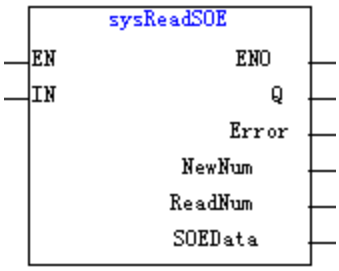
输出参数	数据类型	参数说明
Q	BOOL	输出，默认值 FALSE
Error	BYTE	错误，默认值 0 0：正确 128：单机系统 129：主从同步尚未完成 130：从机的钥匙开关在 PRG 状态

6.10.3.3 事件顺序记录指令

1. sysReadSOE(读取 SOE 记录)

第1步 功能说明

读取 SOE 模块的 SOE 数据，最多一次读取 50 条。



sysReadSOE 功能块

第2步 参数说明

输入参数	数据类型	参数说明
IN	BOOL	上升沿使能

输出参数	数据类型	参数说明
Q	BOOL	输出 TRUE：指令执行成功 FALSE：指令执行失败

Error	BYTE	错误码 0：正确 1：内部 SOE 队列错误 2：组态 SOE 数据接收缓冲区错误 3：未产生新 SOE 事件数据
NewNum	WORD	自上次读取后新增 SOE 事件条数
ReadNum	WORD	本次读取的 SOE 事件条数，每次读取不超过 50 条
SOEData	ARRAY [0 49] of STSOEDATA	存放 50 条 SOE 事件的缓存区，最多存放 50 条 SOE 事件 结构体 STSOEDATA 定义详见下表

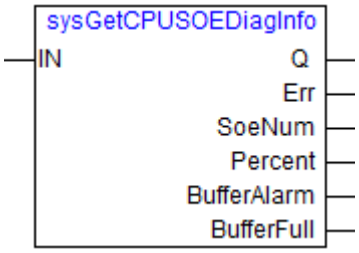
STSOEDATA 结构体成员

成员变量	数据类型	参数说明
Seconds	DWORD	SOE 事件发生的秒时间
T1	WORD	SOE 事件发生的百微秒时间
ulOffset	DWORD	SOE 事件发生的数据区偏移
ChannelData	BYTE	SOE 事件发生的通道信息 Bit6~Bit7：通道当前值 Bit5：变量大小 Bit4：新增有效位，占 1 个 bit，1 表示时钟有效，0 表示时钟失锁 Bit0~Bit3：数据区索引

2. sysGetCPUSOEDiagInfo (获取主控 SOE 诊断信息)

第1步 功能说明

获取主控 SOE 事件存储区诊断信息。



sysGetCPUSOEDiagInfo 指令

第2步 参数说明

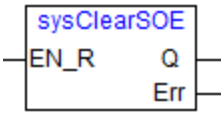
输出参数	类型	含义
IN	BOOL	使能输入信号，电平触发 1：有效 0：无效

Q	BOOL	输出完成标志 0：输出未完成 1：输出完成
Err	BYTE	错误值 0：获取 SOE 诊断信息成功
SoeNum	WORD	主控当前缓存的 SOE 条数
Percent	REAL	主控 SOE 缓存条数占最大存储条数的百分比，最大存储 6000 条
BufferAlarm	BOOL	主控 SOE 缓存条数超过最大存储的 70%时，置位告警标志 1
BufferFull	BOOL	主控 SOE 缓存条数达到最大存储的 100%时，置位告警标志 1

3. sysClearSOE (清除主控 SOE 数据)

第1步 功能说明

清空主控及 SOE 模块存储的 SOE 事件数据。



sysClearSOE 指令

第2步 参数说明

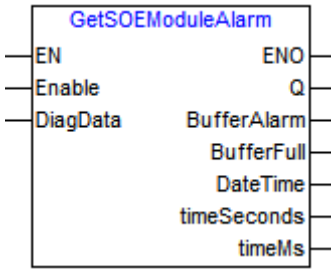
输入参数	类型	含义
EN_R	BOOL	使能信号，上升沿触发
Q	BOOL	输出完成标志 0：输出未完成 1：输出完成
Err	BYTE	错误值 0：清空 SOE 数据成功

6.10.3.4 模块扩展信息

1. GetSOEModuleAlarm(SOE 缓存告警信息查询功能块)

第1步 功能说明

获取 SOE 模块的告警信息，包括 SOE 事件记录条数 70%告警信息、100%告警信息以及事件记录时间信息。



GetSOEModuleAlarm 指令

第2步 参数说明

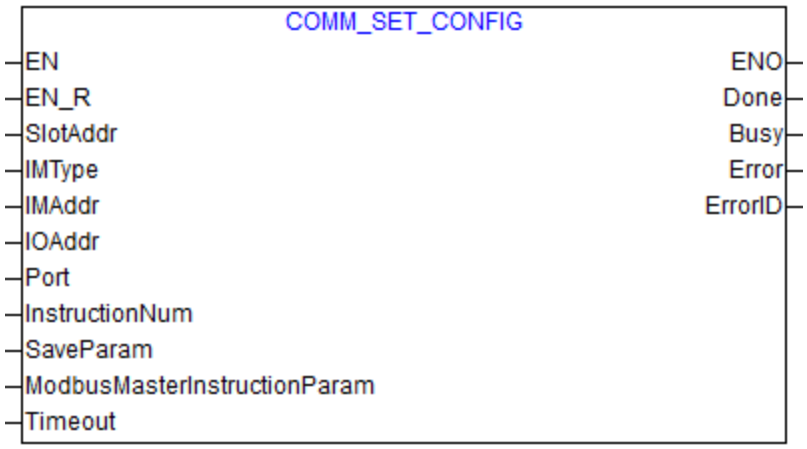
输入参数	类型	含义
Enable	BOOL	使能输入 0：无效 1：有效
DiagData	ARRAY[0..237] OF BYTE	诊断数据。当模块上报诊断数据时，记录一次 SOE 事件 模块在 POWERLINK 下组态时，您需要将 SOE 模块 S 区诊断变量的首地址+6 赋值给“DiagData”

输出参数	类型	含义
Q	BOOL	输出完成标志 0：输出未完成 1：输出完成
BufferAlarm	BOOL	70%告警标志 SOE 事件缓存条数超过最大存储的 70%时，置位告警标志 1，最大存储 3072 条
BufferFull	BOOL	100%告警标志 SOE 事件缓存条数超过最大存储的 100%时，置位告警标志 1，最大存储 3072 条
DataTime	DT	SOE 告警事件发生的记录时间
timeSeconds	DWORD	分钟及以上的事件记录时间对应的秒时间
timeMs	WORD	分钟以下的事件记录时间对应的毫秒时间

2. COMM_SET_CONFIG(LK238C 设置通讯指令参数功能块)

第1步 功能说明

设置 LK238C 模块的通讯指令参数，一次最多设置 16 条指令。



COMM_SET_CONFIG 指令

第2步 参数说明

输入参数	类型	含义
EN_R	BOOL	上升沿执行
SlotAddr	BYTE	通信板卡槽位（2~5）
IMType	BYTE	IO 模块所在背板使用的接口模块类型： 2：接口模块类型为 LK235C（当前唯一合法值）
IMAddr	DWORD	IO 模块所在背板使用的接口模块地址
IOAddr	BYTE	IO 模块所在背板的 DP 从站地址（2~125）
Port	BYTE	端口号（1~4）
InstructionNum	BYTE	配置指令数目（1~16）
SaveParam	BOOL	下发指令参数是否掉电保持
ModbusMasterInstructionParam	ARRAY[0..15] OF MASTERINSTRUCTIONPARAM	Modbus 主站指令参数结构体
Timeout	WORD	执行超时时间(单位:1ms)（0~65535），当设置为 0 时采用默认值 2000；该参数表示，控制器与串口模块通讯异常后多长时间报错，与实际通讯是否超时无关

输出参数	类型	含义
Done	BOOL	通讯参数配置完成
Busy	BOOL	正在进行通信参数配置
Error	BOOL	设置通信参数过程中发生错误
ErrorID	WORD	错误码： =0：无错误

		=1：IMType 参数错误 =2：SlotAddr 指定槽位范围错误 =3：SlotAddr 指定槽位的通信板卡类型不支持 =4：IOAddr 指定的 IO 模块 DP 从站地址错误 =5：IMAddr 指定的接口模块地址错误 =7：IOAddr 指定地址模块类型错误（此号站配置的不是 LK238C 模块，请检查 AddrType、SlotAddr、IMAddr 以及 IOAddr 参数配置是否正确） =8：输入/输出字节长度错误（请检查模块配置页面输入/输出模块） =9：Port 端口参数配置错误 =10：InstructionNum 配置指令个数错误 =11：配置指令参数长度超过输出字节长度（请检查模块配置页面输出模块） =12：ModbusMasterInstructionParam 参数配置错误 =13：该端口的用户参数配置的不是 Modbus 主站 =15：控制器与串口模块通讯异常

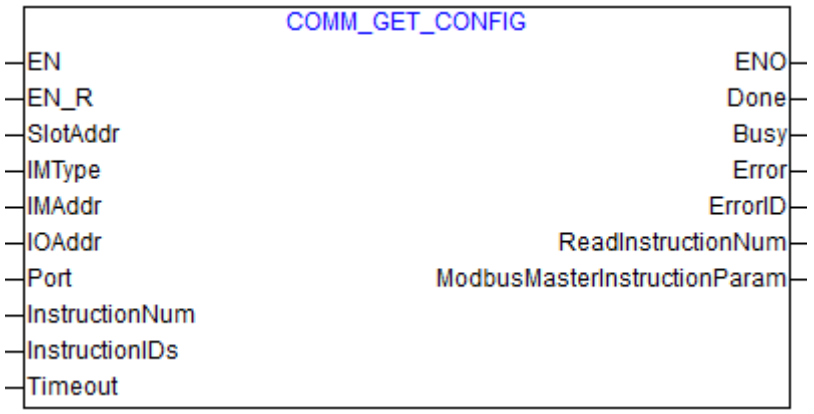
MasterInstructionParam 结构体成员

成员变量	数据类型	参数说明
InstructionID	BYTE	指令 ID 号（1~32）
InstructionStatus	BYTE	指令状态（0：指令无效，1：指令生效）
StationAddress	BYTE	从站地址（1~247）
FunctionCode	BYTE	0x01（读线圈）、0x02（读离散量输入）、0x03（读保持寄存器）、0x04（读输入寄存器）、0x05（写单个线圈）、0x06（写单个寄存器）、0x0F（写多个线圈）、0x10（写多个寄存器）
OffsetAddress	WORD	读写偏移地址
DataLength	WORD	读写开关量/模拟量数据长度 开关量：1~1600 模拟量：1~100
Timeout	WORD	执行超时时间（单位：1ms）
TriggerMode	BYTE	触发模式 0：周期触发，当参数配置成功，且协议端口使能，则 Modbus 按照 Period 设定的轮询周期开始通讯 1：事件触发，当参数配置成功，且协议端口使能，通过 COMM_RW_MODBUS_MASTER（LK238 Modbus 主站读写通讯数据功能块）中 EN_R 上升沿触发一次 Modbus 通讯
Period	BYTE	轮询周期时间（单位：10ms），当设置为 0 时采用默认轮询方式为死兜型

3. COMM_GET_CONFIG(LK238C 获取通讯指令参数功能块)

第1步 功能说明

获取 LK238C 模块的通讯指令参数，一次最多设置 16 条指令。



COMM_GET_CONFIG 指令

第2步 参数说明

输入参数	类型	含义
EN_R	BOOL	上升沿执行
SlotAddr	BYTE	通信板卡槽位（2~5）
IMType	BYTE	IO 模块所在背板使用的接口模块类型： 2：接口模块类型为 LK235C（当前唯一合法值）
IMAddr	DWORD	IO 模块所在背板使用的接口模块地址
IOAddr	BYTE	IO 模块所在背板的 DP 从站地址（2~125）
Port	BYTE	端口号（1~4）
InstructionNum	BYTE	配置指令数目（1~16）
InstructionIDs	ARRAY[0..15] OF BYTE	指令 ID 号（1~32）
Timeout	WORD	执行超时时间(单位:1ms)（0~65535），当设置为 0 时采用默认值 2000；该参数表示，控制器与串口模块通讯异常后多长时间报错，与实际通讯是否超时无关

输出参数	类型	含义
Done	BOOL	读取通讯参数完成
Busy	BOOL	正在进行读取参数配置
Error	BOOL	读取通信参数过程中发生错误
ErrorID	WORD	功能块错误码 =0：无错误 =1：IMType 协议类型错误 =2：SlotAddr 指定槽位范围错误 =3：SlotAddr 指定槽位的通信板卡类型不支持 =4：IOAddr 指定的 IO 模块 DP 从站地址错误 =5：IMAddr 指定的接口模块地址错误 =7：IOAddr 指定地址模块类型错误（此号站配

		置的不是 LK238C 模块，请检查 AddrType、SlotAddr、IMAddr 以及 IOAddr 参数配置是否正确） =8：输入/输出字节长度错误（请检查模块配置页面输入/输出模块） =9：Port 端口参数配置错误 =10：InstructionNum 参数配置错误 =11：获取指令参数长度超过输入字节长度（请检查模块配置页面输入模块） =12：InstructionIDs 参数配置错误 =13：该端口的用户参数配置的不是 Modbus 主站 =15：控制器与串口模块通讯异常
ReadInstructionNum	BYTE	读取配置总个数
ModbusMasterInstructionParam	ARRAY[0..15] OF MASTERINSTRUCTIONPARAM	Modbus 主站指令参数结构体

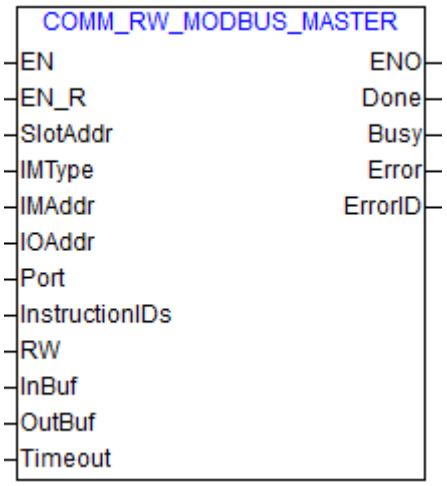
MasterInstructionParam 结构体成员

成员变量	数据类型	参数说明
InstructionID	BYTE	指令 ID 号（1~32）
InstructionStatus	BYTE	指令状态（0：指令无效，1：指令生效）
StationAddress	BYTE	从站地址（1~247）
FunctionCode	BYTE	0x01（读线圈）、0x02（读离散量输入）、0x03（读保持寄存器）、0x04（读输入寄存器）、0x05（写单个线圈）、0x06（写单个寄存器）、0x0F（写多个线圈）、0x10（写多个寄存器）
OffsetAddress	WORD	读写偏移地址
DataLength	WORD	读写开关量/模拟量数据长度 开关量：1~1600 模拟量：1~100
Timeout	WORD	执行超时时间（单位：1ms）
TriggerMode	BYTE	触发模式（0：周期触发，1：事件触发）
Period	BYTE	轮询周期时间（单位：10ms）

4. COMM_RW_MODBUS_MASTER(LK238C Modbus 主站读写通讯数据功能块)

第1步 功能说明

读取或写入 LK238C Modbus 主站数据。



COMM_RW_MODBUS_MASTER 指令

第2步 参数说明

输入参数	类型	含义
EN_R	BOOL	上升沿执行
SlotAddr	BYTE	通信板卡槽位（2~5）
IMType	BYTE	IO 模块所在背板使用的接口模块类型： 2：接口模块类型为 LK235C（当前唯一合法值）
IMAddr	DWORD	IO 模块所在背板使用的接口模块地址
IOAddr	BYTE	IO 模块所在背板的 DP 从站地址（2~125）
Port	BYTE	端口号（1~4）
InstructionIDs	BYTE	指令 ID 号（1~32）
RW	BOOL	读写方向（RW 为 0 时读数据，1 时写数据）
InBuf	POINTER TO BYTE	输入数据缓冲区
OutBuf	POINTER TO BYTE	输出数据缓冲区
Timeout	WORD	执行超时时间(单位:1ms)（0~65535），当设置为 0 时采用默认值 2000；该参数表示，控制器与串口模块通讯异常后多长时间报错，与实际通讯是否超时无关

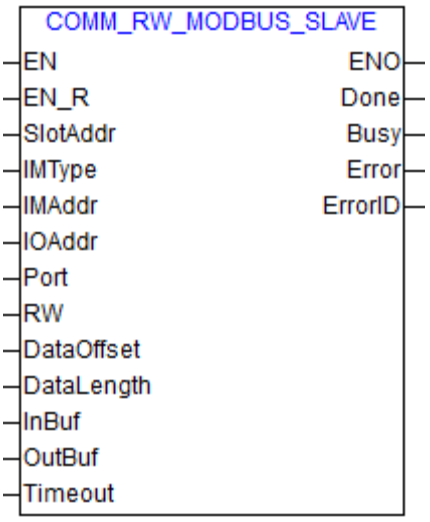
输出参数	类型	含义
Done	BOOL	读写主站数据完成
Busy	BOOL	正在进行读写主站数据
Error	BOOL	读写主站数据过程中发生错误
ErrorID	WORD	功能块错误码

		=0：无错误 =1：IMType 协议类型错误 =2：SlotAddr 指定槽位范围错误 =3：SlotAddr 指定槽位的通信板卡类型不支持 =4：IOAddr 指定的 IO 模块 DP 从站地址错误 =5：IMAddr 指定的接口模块地址错误 =7：IOAddr 指定地址模块类型错误（此号站配置的不是 LK238C 模块，请检查 AddrType、SlotAddr、IMAddr 以及 IOAddr 参数配置是否正确） =8：输入/输出字节长度错误（请检查模块配置页面输入/输出模块） =9：Port 端口参数配置错误 =10：InstructionIDs 参数配置错误 =15：控制器与串口模块通讯异常 主站错误信息 =1001:数据帧错误 =1002:超时 =1004:当前指令无效 =1005:端口未工作（当用户参数页面指定端口 Com Status 为 Disable） =1006:端口协议不匹配 =1128+1:从站不支持的功能码 =1128+2: 数据地址溢出 =1128+3: 数据范围溢出 =1128+4: 从站设备故障 =1128+6: 从站设备忙

5. COMM_RW_MODBUS_SLAVE(LK238C Modbus 从站读写通讯数据功能块)

第1步 功能说明

Modbus 从站读写通讯数据功能块。



COMM_RW_MODBUS_SLAVE 指令

第2步 参数说明

输入参数	类型	含义
EN_R	BOOL	上升沿执行
SlotAddr	BYTE	通信板卡槽位（2~5）
IMType	BYTE	IO 模块所在背板使用的接口模块类型： 2：接口模块类型为 LK235C（当前唯一合法值）
IMAddr	DWORD	IO 模块所在背板使用的接口模块地址
IOAddr	BYTE	IO 模块所在背板的 DP 从站地址（2~125）
Port	BYTE	端口号（1~4）
RW	BOOL	读写方向（RW 为 0 时读数据，1 时写数据）
DataOffset	WORD	读写数据偏移地址
DataLength	BYTE	读取数据长度
InBuf	POINTER TO BYTE	输入数据缓冲区
OutBuf	POINTER TO BYTE	输出数据缓冲区
Timeout	WORD	执行超时时间(单位:1ms)（0~65535），当设置为 0 时采用默认值 2000；该参数表示，控制器与串口模块通讯异常后多长时间报错，与实际通讯是否超时无关

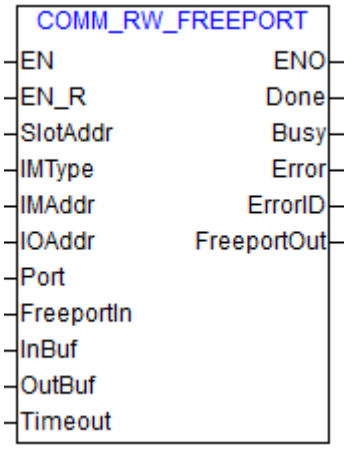
输出参数	类型	含义
Done	BOOL	读写从站数据完成
Busy	BOOL	正在进行读写从站数据
Error	BOOL	读写从站数据过程中发生错误
ErrorID	WORD	功能块错误码 =0：正确 =1：IMType 协议类型错误 =2：SlotAddr 指定槽位范围错误 =3：SlotAddr 指定槽位的通信板卡类型不支持 =4：IOAddr 指定的 IO 模块 DP 从站地址错误 =5：IMAddr 指定的接口模块地址错误 =7：IOAddr 指定地址模块类型错误（此号站配置的不是 LK238C 模块，请检查 AddrType、SlotAddr、IMAddr 以及 IOAddr 参数配置是否正确） =8：输入/输出字节长度错误（请检查模块配置页面输入/输出模块） =9：Port 端口参数配置错误 =10：读取数据长度超过输入字节长度（请检查模块配置页面输入模块） =11：写数据长度超过输出字节长度（请检查模块配置页面输出模块） =12：读写数据长度设定值过小（为 0） =13：读写数据长度超过 220 字节

		=14：偏移加长度大于 60*1024 =15：控制器与串口模块通讯异常 从站错误信息 =1001:CRC 校验错误； =1005:端口未工作（当用户参数页面指定端口 Com Status 为 Disable） =1006:端口协议不匹配 =1128+2: 数据地址溢出； =1128+3: 数据范围溢出；

6. COMM_RW_FREEPORT(LK238C 自由口协议读写通讯数据功能块)

第1步 功能说明

读取或写入 LK238C Modbus 从站数据。



COMM_RW_FREEPORT 指令

第2步 参数说明

输入参数	类型	含义
EN_R	BOOL	上升沿使能
SlotAddr	BYTE	通信板卡槽位（2~5）
IMType	BYTE	IO 模块所在背板使用的接口模块类型： 2：接口模块类型为 LK235C（当前唯一合法值）
IMAddr	DWORD	IO 模块所在背板使用的接口模块地址
IOAddr	BYTE	IO 模块所在背板的 DP 从站地址（2~125）
Port	BYTE	端口号（1~4）
FreeportIn	FREEPORTINPARAM	自由口协议输入数据
InBuf	POINTER TO BYTE	输入数据缓冲区

OutBuf	POINTER TO BYTE	输出数据缓冲区
Timeout	WORD	执行超时时间(单位:1ms)（0~65535），当设置为 0 时采用默认值 2000；该参数表示，控制器与串口模块通讯异常后多长时间报错，与实际通讯是否超时无关

输出参数	类型	含义
Done	BOOL	读写自由口数据完成
Busy	BOOL	正在进行读写数据
Error	BOOL	读写数据过程中发生错误
ErrorID	WORD	功能块错误码 =0：正确 =1：IMType 协议类型错误 =2：SlotAddr 指定槽位范围错误 =3：SlotAddr 指定槽位的通信板卡类型不支持 =4：IOAddr 指定的 IO 模块 DP 从站地址错误 =5：IMAddr 指定的接口模块地址错误 =7：IOAddr 指定地址模块类型错误（此号站配置的不是 LK238C 模块，请检查 AddrType、SlotAddr、IMAddr 以及 IOAddr 参数配置是否正确） =8：输入/输出字节长度错误（请检查模块配置页面输入/输出模块） =9：Port 端口参数配置错误 =10：发送数据长度超过输出长度字节（发送数据长度包括数据头的 10 字节长度） =11：接收数据长度超过输入长度字节（接收数据长度包括数据头的 7 字节长度） =12：发送长度设定异常（等于 0 或大于 210） =13：接收长度设定异常（大于 210） =14：端口未工作（当用户参数页面指定端口 Com Status 为 Disable；COMM_CTL_MODULE 功能块中，ComEnable 参数中对应端口未使能） =15：控制器与串口模块通讯异常 =16：端口协议不匹配 =17：接收使能和发送使能均设置为 0(收发模式下接收和发送至少有一个需要使能)
FreeportOut	FREEPORTOUTPARAM	自由口协议输出数据

FreeportInParam 结构体成员

成员变量	数据类型	参数说明
SendEn	BOOL	发送使能（高电平使能）
SendLength	BYTE	设置发送长度（1~210）
SendTimeout	BYTE	设置发送超时时间（1~255，单位：10ms）
RecvEn	BOOL	接收使能（用户参数中配置为收发模式，RecvEn 高电平使能；用户参数中配置为只收模

		式，RecvEn 上升沿使能，高电平持续收数据）
RecvLen	BYTE	设置接收长度（0~210）（当使能开始字符和结束字符的时候，接收长度可以设置为 0）
StartChar	BYTE	设置开始字符（1~255）
EndChar	BYTE	设置结束字符（1~255）
RecvTimeout	BYTE	设置接收超时时间（1~255，单位：10ms）
RecvMode	BYTE	接收模式 Bit0:超时时间使能 Bit1:忽略 Bit2:结束字符使能 Bit3:开始字符使能 Bit4-7:忽略
AckID	BYTE	主站确认包号

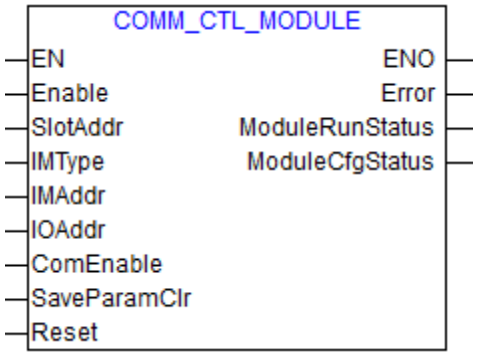
FreeportOutParam 结构体成员

成员变量	数据类型	参数说明
SendQ	BOOL	发送结束
SendErr	BYTE	发送状态 =0：正确 =2：超时
RecvQ	BOOL	接收结束
RecvCount	BYTE	接收字节数
RecvErr	BYTE	结束状态 =0：正确 =2：缺少开始条件（使能开始字符但开始字符设置为 0；使能结束字符，但是未使能开始字符，同时接收长度为 0） =3：缺少结束条件（使能结束字符但结束字符设置为 0；使能开始字符，但是未使能结束字符，同时接收长度为 0） =4：超时时间（Timeout）设定过小 =5：超时 =6：未找到结束字符
PackID	BYTE	LK238C 返回命令号

7. COMM_CTL_MODULE(LK238C 模块控制功能块)

第1步 功能说明

该功能块可以使能四个串口，清空 Modbus 主站指令，复位交互接口。



COMM_CTL_MODULE

第2步 参数说明

输入参数	类型	含义
Enable	BOOL	高电平使能
SlotAddr	BYTE	通信板卡槽位（2~5）
IMType	BYTE	IO 模块所在背板使用的接口模块类型： 2：接口模块类型为 LK235C（当前唯一合法值）
IMAddr	DWORD	IO 模块所在背板使用的接口模块地址
IOAddr	BYTE	IO 模块所在背板的 DP 从站地址（2~125）
ComEnable	MODULECONTROLPARAM	COM1~COM4 的工作使能位
SaveParamClr	BOOL	清空掉电保存的通讯指令参数
Reset	BOOL	复位控制

输出参数	类型	含义
Error	BYTE	功能块错误码 =0：正确 =1：IMType 协议类型错误 =2：SlotAddr 指定槽位范围错误 =3：SlotAddr 指定槽位的通信板卡类型不支持 =4：IOAddr 指定的 IO 模块 DP 从站地址错误 =5：IMAddr 指定的接口模块地址错误 =7：IOAddr 指定地址模块类型错误（此号站配置的不是 LK238C 模块，请检查 AddrType、SlotAddr、IMAddr 以及 IOAddr 参数配置是否正确）
ModuleRunStatus	MODULERUNSTATUSPARAM	模块工作状态
ModuleCfgStatus	MODULECFGSTATUSPARAM	模块配置状态

ModuleControlParam 结构体成员

成员变量	数据类型	参数说明

COM1_EN	BOOL	COM1 工作使能
COM2_EN	BOOL	COM2 工作使能
COM3_EN	BOOL	COM3 工作使能
COM4_EN	BOOL	COM4 工作使能

ModuleRunStatusParam 结构体成员

成员变量	数据类型	参数说明
ComRunStatus1	BOOL	COM1 工作状态
ComRunStatus2	BOOL	COM2 工作状态
ComRunStatus3	BOOL	COM3 工作状态
ComRunStatus4	BOOL	COM4 工作状态

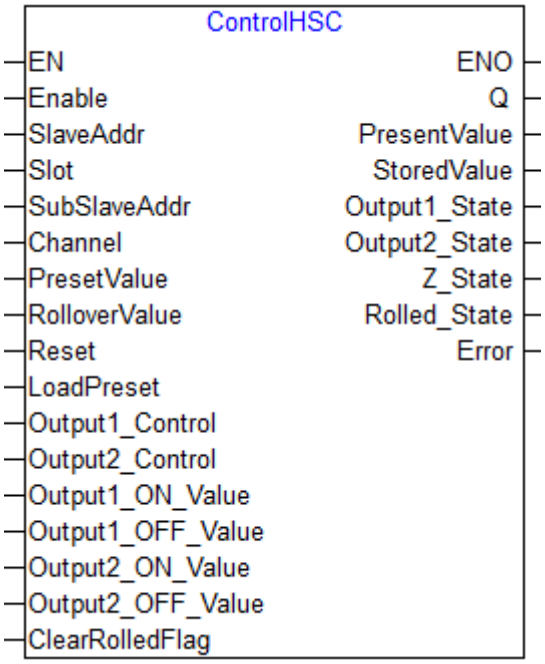
ModuleCfgStatusParam 结构体成员

成员变量	数据类型	参数说明
ComCfgStatus1	BOOL	COM1 配置状态
ComCfgStatus2	BOOL	COM2 配置状态
ComCfgStatus3	BOOL	COM3 配置状态
ComCfgStatus4	BOOL	COM4 配置状态

8. ControlHSC(双通道计数模块控制监视功能块)

第1步 功能说明

设置计数模块的参数以及读取计数模块的运算状态。



ControlHSC 指令

第2步 参数说明

输入参数	数据类型	功能描述	参数值说明
Enable	BOOL	使能输入	0：无效 1：有效
SlaveAddr	BYTE	站地址	从站地址，地址范围 2~125，默认值 2
Slot	BYTE	槽位号	槽位号，范围值 2~5，默认值 2
SubSlaveAddr	BYTE	从站站地址	LK235C 模块从站地址，地址范围 1~239，默认值 1
Channel	BYTE	通道号	=1，通道 1（计数器 1） =2，通道 2（计数器 2）
PresetValue	DWORD	预置值	计数器从该值开始，此值必须小于翻转值，否则会发生计数错误，预置值的取值范围 0~4,294,967,295
RolloverValue	DWORD	翻转值	计数模式下，作为计数的上限，取值范围为 1~4,294,967,295。在计数过程中，当计数值＝（翻转值－1）时，计数器回到 0，重新开始计数
Reset	BOOL	复位计数器	0：无效 0→1：上升沿，计数器复位并从 0 开始计数 1：两个输出通道输出 OFF，不按照组态设置输出
LoadPreset	BOOL	预置值控制位	0：无效 1：上升沿有效，计数器载入预置值并从预置值开始计数
Output1_Control	BYTE	输出控制	用于控制计数器各输出点的状态，输出 1 当前输出值修改：

			=0x00，根据计数结果输出 =0x02，强制修改 OUT1 输出值为 OFF =0x03，强制修改 OUT1 输出值为 ON
Output2_Control	BYTE	输出控制	用于控制计数器各输出点的状态，输出 2 当前输出值修改： =0x00，根据计数结果输出 =0x02，强制修改 OUT1 输出值为 OFF =0x03，强制修改 OUT1 输出值为 ON
Output1_ON_Value	DWORD	输出 ON 触发值	输出 1 输出 ON 触发值(0~4,294,967,295)
Output1_OFF_Value	DWORD	输出 OFF 触发值	输出 1 输出 OFF 触发值(0~4,294,967,295)
Output2_ON_Value	DWORD	输出 ON 触发值	输出 2 输出 ON 触发值(0~4,294,967,295)
Output2_OFF_Value	DWORD	输出 OFF 触发值	输出 2 输出 OFF 触发值(0~4,294,967,295)
ClearRolledFlag	BOOL	翻转标志清除位	清除翻转标志位

输出参数	**数据类型**	**功能描述**	**参数值说明**
Q	BOOL	输出完成标志	0：输出未完成 1：输出完成
PresentValue	DWORD	当前计数值	存放所选计数器的当前计数值
StoredValue	DWORD	存储计数值	所选计数器的当前存储计数值
Output1_State	BOOL	输出 1 状态	1：ON（通道闭合），即当前计数值达到 Output1_ON_Value 0：OFF（通道断开，停止输出），即当前计数达到 Output1_OFF_Value
Output2_State	BOOL	输出 2 状态	1：ON（通道闭合），即当前计数值达到 Output2_ON_Value 0：OFF（通道断开，停止输出），即当前计数达到 Output2_OFF_Value
Z_State	BOOL	Z 状态	0：低电平 1：高电平
Rolled_State	BOOL	翻转标志	计数器是否达到翻转值并翻转 0：未翻转 1：已翻转
Error	DWORD	错误信息	值为 0，无错误 Bit1：槽位不在范围内 2~5

			Bit2：模块 ID 不在范围内 3~4 Bit3：输入 IO 从站不在范围内 2~125（LK249C 或者 LK235C） Bit4：LK235C 站号地址不在范围内 1~239（LK241C） Bit5：此号站配置的不是计数模块 Bit6：通道值错位 Bit7：预设值错误 Bit8：输出控制位错误 Bit9：输出触发值错误

第7章　组态硬件

7.1 组态 PowerLink 协议

7.1.1 添加 POWERLINK 主站

7.1.1.1 概述

当通过 POWERLINK 协议组网时，需要有一个 POWERLINK 主站模块与 POWERLINK 从站模块通信，主站模块被安装在主控背板的通讯插槽中，同时，需要控制器下添加并配置主站参数。

7.1.1.2 要求

模块已安装

7.1.1.3 步骤

要添加 POWERLINK 主站，请按以下步骤操作：

（1） 在控制器树节点右键菜单中，选择“添加”命令

将弹出“添加”对话框。

（2） 选择 LK241C POWERLINK 主站模块。

（3） 根据 LK241C POWERLINK 主站通信模块在主控背板上的槽位号，输入设备地址。

（4） 单击**确定**按钮。

LK241C 主站模块被添加到控制器下。

7.1.1.4 下一步

添加 POWERLINK 主协议。

7.1.1.5 参考信息

配置设备地址

添加 POWERLINK 主协议

添加 POWERLINK 网关

添加 POWERLINK 从协议

添加 POWERLINK 从站

7.1.2 添加 POWERLINK 主协议

7.1.2.1 概述

主站模块通过 POWERLINK 协议与从站进行数据通信，需要为主站模块配置该协议。

7.1.2.2 要求

已添加主站模块

7.1.2.3 步骤

要添加 POWERLINK 协议，请按以下步骤操作：

（1） 在主站模块右键菜单中，选择“添加”命令。

弹出“添加”对话框。

（2） 选择“POWERLINK_Master”协议。

（3） 单击**确定**按钮。

POWERLINK_Master 协议被添加到主站模块下。

7.1.2.4 下一步

添加 POWERLINK 网关。

7.1.2.5 参考信息

添加 POWERLINK 主站

添加 POWERLINK 网关

关联任务

7.1.3 添加 Powerlink 网关

7.1.3.1 概述

POWERLINK 主站和 POWERLINK 从站通过网关连接，网关模块向上层主站上报 IO 从站数据信息，向下层 IO 从站传出控制器下发数据。需要在扩展背板通信槽中安装 LK235C 接口模块，同时，需要在 POWERLINK 主站协议下添加并配置网关参数。

最大可组态 128 个 LK235 接口模块。

7.1.3.2 要求

- 已添加“POWERLINK_Master”协议
- LK235C 已安装并设置站地址

7.1.3.3 步骤

要添加网关模块，请按以下步骤操作：

（1） 在“POWERLINK_Master”协议节点右键菜单中，选择“添加”命令。

将弹出“添加”对话框。

（2） 选择 LK235C POWERLINK 接口模块。

（3） 根据模块上地址拨码开关设置，输入设备地址。

（4） 单击**确定**按钮。

LK235C 接口模块被添加到“POWERLINK_Master”协议下。

7.1.3.4 下一步

添加 POWERLINK 从协议。

7.1.3.5 参考信息

设置 POWERLINK 网关设备地址

7.1.4 添加 PowerLink 从协议

7.1.4.1 概述

从站模块通过 POWERLINK 协议与主站进行数据通信，需要为从站模块配置 POWERLINK 协议。

7.1.4.2 要求

已添加网关模块

7.1.4.3 步骤

要添加 POWERLINK 从协议，请按以下步骤操作：

（1） 在网关模块右键菜单中，选择“添加”命令。

弹出“添加”对话框。

（2） 选择“POWERLINK_Slave”协议。

（3） 单击**确定**按钮。

POWERLINK_Slave 协议被添加到网关模块下。

7.1.4.4 下一步

添加 POWERLINK 网关

添加 POWERLINK 从站

7.1.5 添加 I/O 从站

7.1.5.1 概述

POWERLINK 网关下挂接 I/O 从站模块，根据扩展背板槽 IO 模块的安装顺序，依次添加 I/O。

每个 LK235C 网关模块最大支持 10 个 IO 模块。

7.1.5.2 要求

已添加“POWERLINK_Slave”协议

7.1.5.3 步骤

要添加 POWERLINK 从站，请按以下步骤操作：

（1） 在“POWERLINK_Slave”协议节点右键菜单中，选择“添加”命令。

将弹出“添加”对话框。

（2） 选择从站模块。

（3） 输入设备地址。

IO 模块设备地址与模块安装槽位和背板基地址有关，详见参考信息。

（4） 单击**确定**按钮。

从站模块被添加到“POWERLINK_Slave”协议下。

7.1.5.4 参考信息

配置设备地址

设置背板基地址

添加 POWERLINK 从协议

组态冗余 I/O

7.1.6 配置主站轮询周期

7.1.6.1 概述

POWERLINK 主站和 I/O 从站间周期性交互数据，主站按顺序周期性轮询每一个从站，将数据传输到从站，或接受从站数据。将所有的从站轮询一遍即为一个轮询周期。需要为主站配置轮询周期参数。

轮询周期与 I/O 从站的输入输出字节总数有关，默认值为 1000。

7.1.6.2 要求

“POWERLINK_Master”协议窗口已打开

7.1.6.3 步骤

要配置主站轮询周期，请按以下步骤操作：

（1） 在“设备参数”签页中，单击“轮询周期”的增大或减小按钮，调整到设定值。

7.1.7 配置网关通讯参数

7.1.7.1 概述

POWERLINK 主从站数据通讯时，需要设置以下通讯参数。

- 响应时间：POWERLINK 主站发送请求后，等待网关应答的时间。设置范围 300~1000us，默认值 750。
- 通信模式：保持默认值。
- 通讯故障模式：当发生通讯故障时，LK235C 挂载的 I/O 模块离线或保持在线值。
  - 复位：LK235C 与 LK241C 间发生通讯故障时，下挂 I/O 模块全部离线。
  - 保持：LK235C 与 LK241C 间发生通讯故障时，下挂 I/O 模块保持在线状态。

7.1.7.2 要求

LK235C 模块窗口已打开

7.1.7.3 步骤

要配置网关通讯参数，请按以下步骤操作：

（1） 在“设备参数”签页中，单击“响应时间”的增大或减小按钮，调整到设定值。

7.1.8 配置 DP 参数

7.1.8.1 DP 主站侧参数配置

默认勾选“使用默认值”，用户可通过切换不同波特率实现其他参数的优化配置。

勾选状态下，“波特率”除外的其他 DP 参数不可编辑。

去勾选“使用默认值”时，用户可对“TSL”、“重试最大次数”、“最小从站间隔参数”进行编辑设置。

“使用默认值”参数，用户由去勾选态切换至勾选态时，根据当前的“波特率”重置其他 DP 参数的默认值。

DP 主站侧参数配置

7.2 组态 Profibus-DP 协议

7.2.1 添加 Profibus-DP 主站

7.2.1.1 概述

当通过 Profibus-DP 组网时，需要有一个 Profibus-DP 主站模块与 I/O 从站通信，主站模块被安装在主控背板的通讯插槽中，同时，需要控制器下添加并配置主站参数。

7.2.1.2 要求

模块已安装

7.2.1.3 步骤

要添加 Profibus-DP 主站，请按以下步骤操作：

（1） 在控制器树节点右键菜单中，选择“添加”命令。

将弹出“添加”对话框。

（2） 选择 LK249C DP 主站通信模块。

（3） 根据 LK249C 模块在主控背板上的槽位号，输入设备地址。

（4） 单击**确定**按钮。

LK249C 主站模块被添加到控制器下。

7.2.1.4 下一步

添加 DP_Master 协议

7.2.1.5 参考信息

添加 DP_Master 协议

配置设备地址

7.2.2 添加 DP_Master 协议

7.2.2.1 概述

Profibus-DP 主站模块通过 DP_Master 协议与从站进行数据通信，需要为主站模块配置该协议。

7.2.2.2 要求

已添加主站模块

7.2.2.3 步骤

要添加 DP_Master 协议，请按以下步骤操作：

（1） 在 LK249C 模块右键菜单中，选择“添加”命令。

弹出“添加”对话框。

（2） 选择“DP_Master”协议。

（3） 单击**确定**按钮。

DP_Master 协议被添加到主站模块下。

7.2.2.4 下一步

添加 I/O 从站

7.2.2.5 参考信息

添加 I/O 从站

设置波特率

关联任务

7.2.3 添加 I/O 从站

7.2.3.1 概述

Profibus-DP 主站下挂接 I/O 从站模块，根据扩展背板槽 IO 模块的安装顺序，依次添加 I/O。

7.2.3.2 要求

已添加“DP_Master”协议

7.2.3.3 步骤

要添加 Profibus-DP 从站，请按以下步骤操作：

（1） 在“DP_Master”协议节点右键菜单中，选择“添加”命令。

将弹出“添加”对话框。

（2） 选择从站模块。

（3） 输入设备地址。

IO 模块设备地址与模块安装槽位和背板基地址有关，详见参考信息。

（4） 单击**确定**按钮。

从站模块被添加到“DP_Master”协议下。

7.2.3.4 参考信息

组态冗余 I/O

配置设备地址

7.2.4 设置波特率

7.2.4.1 概述

Profibus-DP 网络中，需要设置网络中数据传输速率。请根据传输距离、I/O 数据字节数，选择适宜的波特率。提供 187.5kbps、500kbps、1.5Mbps、3Mbps、6Mbps。选择不同的波特率，其它总线参数自动匹配为最佳值。

7.2.4.2 要求

“DP_Master”协议窗口已打开

7.2.4.3 步骤

要组态 Profibus-DP 网络波特率，请按以下步骤操作：

（1） 在“设备参数”签页中，单击波特率下拉按钮。

（2） 选择适宜的波特率。

7.3 组态 I/O 从站

7.3.1 导入第三方设备

7.3.1.1 导入设备描述文件

1. 概述

当使用第三方设备组态时，需要导入对应的设备描述文件。支持导入的设备描述文件格式为.gsd、.gse、.gsi、.gsf、.gsg、.gss。

导入规则：

- 当重复导入第三方设备时，需确认是否覆盖之前导入的设备描述文件。
- 当导入的第三方设备已在工程中被组态，则需要在“硬件配置”节点下删除该模块后，再进行导入。
- 组态第三方设备时，需要手动配置输入/输出数据。请参见配置输入/输出数据。

模块对应的设备描述文件作为系统文件，不允许修改或删除。

2. 要求

准备需要导入的设备描述文件

3. 步骤

要导入第三方设备，请按以下步骤操作：

（1） 单击“工程”菜单，选择“导入设备描述文件”—“DP 设备”命令。

将弹出“导入设备描述文件”对话框。

（2） 在存放路径下，选择要导入的设备描述文件。

（3） 单击“打开”按钮。

该设备描述文件被导入，将弹出提示框。

（4） 单击“确定”按钮。

“设备库”窗口的“第三方 DP 设备”节点下将显示导入的设备。

7.3.1.2 配置输入/输出数据

1. 概述

导入的第三方设备，没有配置输入/输出通道数据，需要手动进行配置。

2. 要求

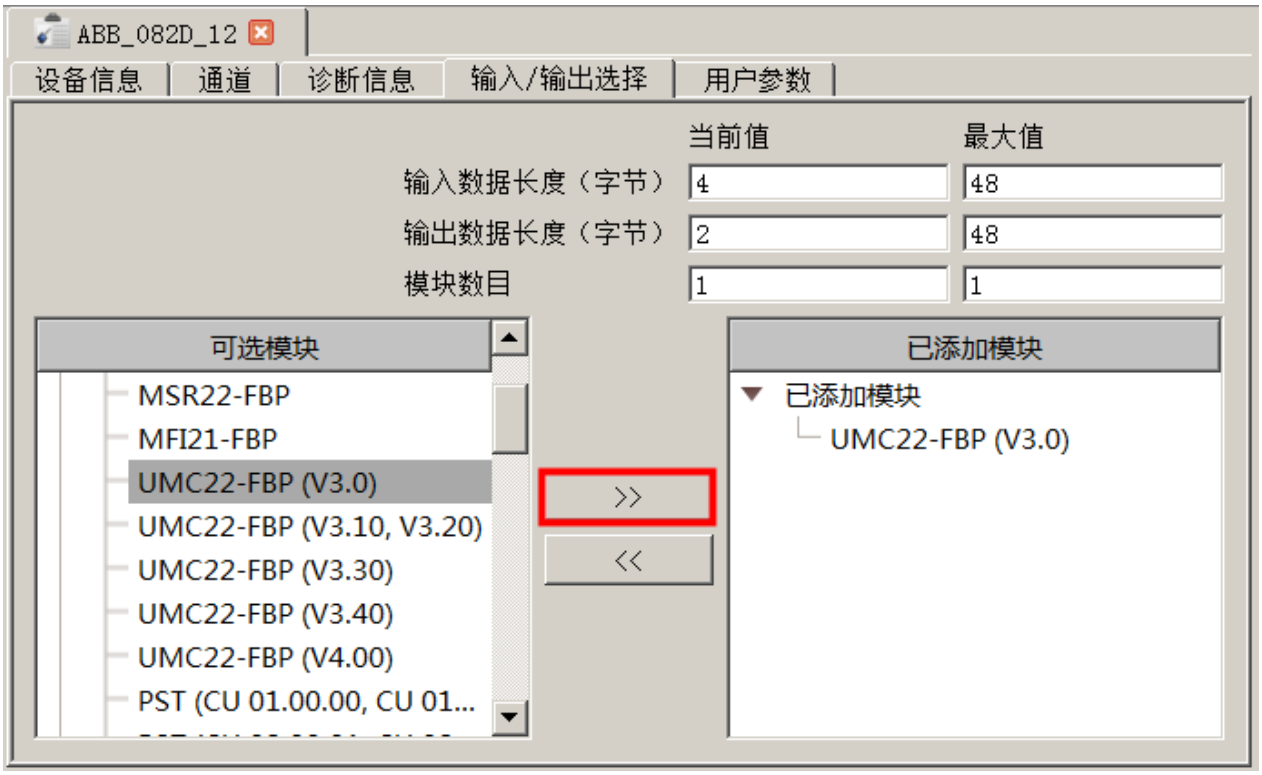
“硬件配置”下已添加第三方设备。

3. 步骤

要配置第三方设备输入/输出数据，请按以下步骤操作：

（1） “硬件配置”节点下，双击第三方设备。

将打开设备信息窗口。



添加输入/输出数据

（2） 在左侧“可选模块”列表框中，选择输入/输出模块。

（3） 单击添加按钮 >> 。

选中模块将被添加到右侧“已添加模块”列表框。

（4） 单击“通道”标签页，可查看已添加的输入/输出数据。

7.3.2 配置设备地址

7.3.2.1 概述

每个设备模块安装在背板上，均有一个唯一的设备地址与之对应，你需要在组态模块时，为每个模块配置一个设备地址。

设备地址与模块在背板上的安装位置有关：

- 主控背板上的模块，设备地址就是安装槽位号,从 0 开始，依次向右加 1。例如，6 槽背板上模块设备地址依次为 0~5，4 槽背板模块设备地址依次为 0~3。
- 扩展背板上的模块，设备地址由背板基地址和模块的槽位偏移地址决定，详细内容请参见设置背板基地址。

配置设备地址时，请按模块实际背板地址进行设置。

可以在添加模块时，进行设置。如果设置后，需要修改设备地址，请按下面的步骤修改。

7.3.2.2 要求

- 背板上已安装模块
- “硬件配置”节点下已添加模块

7.3.2.3 步骤

修改模块设备地址，请按以下步骤操作：

（1） 打开模块“设备信息”页面

（2） 单击“设备地址”栏中的地址

此时，地址栏出现光标。

（3） 删除旧地址，输入新地址

修改地址完成。

7.3.3 配置通道参数

7.3.3.1 使能通道状态

1. 概述

设置通道使能后，通道被激活，正常处理输入或输出信号，并能够检测通道故障。未使能通道的数据值默认为 0。建议未使用的通道设置为未使能。

2. 要求

- 通道接线已完成
- “用户参数”界面已打开

3. 步骤

设置通道使能，请按以下步骤操作：

（1） 选择“CHn State”参数。

（2） 双击参数值“Disable”。

出现编辑框。

（3） 单击下拉按钮。

显示可选参数值。

（4） 选择“Enable”。

下装后生效。

4. 结果

通道激活，能读取现场仪表输入信号，诊断通道故障状态。

7.3.3.2 使能通道故障诊断

1. 概述

LK610C/LK710C 模块使能通道诊断功能后，当通道发生故障时，会向控制器上报相关诊断信息。

对于 LK610C 模块，使能通道诊断后，您还需要在用户开关信号的两端并接电阻，阻值范围 39KΩ~43KΩ@24VDC，才能诊断通道故障。

2. 要求

- 通道已使能
- “用户参数”界面已打开

3. 步骤

使能通道故障诊断功能，请按以下步骤操作：

（1） 选择“CHn Diagnosis State”参数。

（2） 双击参数值“Disable Diagnosis”。

出现编辑框。

（3） 单击下拉按钮。

（4） 选择“Enable Diagnosis”。

4. 结果

通道发生故障时，检测故障信息并上报诊断。

7.3.3.3 使能通道 SOE 功能

LK631C 模块具有 SOE 功能，使能通道 SOE 功能后，如果通道发生信号跳变，如信号由 0 变为 1，或由 1 变为 0，则 SOE 记录该事件，通过通道数据上报。

1. 要求

“用户参数”界面已打开

2. 步骤

使能通道 SOE 功能，请按以下步骤操作：

（1） 选择“通道 SOE 状态”参数。

（2） 双击参数值“DI”。

出现编辑框。

（3） 单击下拉按钮。

（4） 选择“SOE”。

3. 结果

通道激活，能读取现场仪表输入信号，诊断通道故障状态。

7.3.3.4 设置超限报警

1. 使能超限报警

第1步 概述

对于具有超限报警功能的模块，使能超限报警后，当输入值超过设置的报警上限或报警下限时，通道上报超上限或超下限故障。当信号恢复至限定范围内时，上报**故障恢复**。

第2步 要求

“用户参数”界面已打开

第3步 步骤

要设置超限报警，请按以下步骤操作：

（1） 选择“CHn Upper Limit Exceeded Alarm”参数

（2） 双击参数值，下拉框中选择“Enable”。

使能超限报警

第4步 下一步

使能超限报警功能后，请设置超限报警值。

2. 设置报警限值

第1步 概述

在所选定的量程范围之内，用户可自行设定输入信号的报警上限和报警下限。

规则:

- 报警限值对应的量程单位与组态中配置的量程单位保持一致
- 报警上限必须大于报警下限

第2步 要求

- 超限报警已使能
- “用户参数”界面已打开

第3步 步骤

要设置报警限值，请按以下步骤操作：

（1） 选择“CHn Upper Limit Value”参数

当前值被选中。

（2） 输入已计算好的报警限值。

CH1 Upper Limit Value	18700	Unsigned16	18700	6720-25980
CH1 Lower Limit Value	8000	Unsigned16	8000	6720-25980

设置报警上下限

3. LK430C 超限报警功能

LK430C 模块具有超限报警功能。

LK430C 能测量的各种标准热电阻，如测量范围所示。

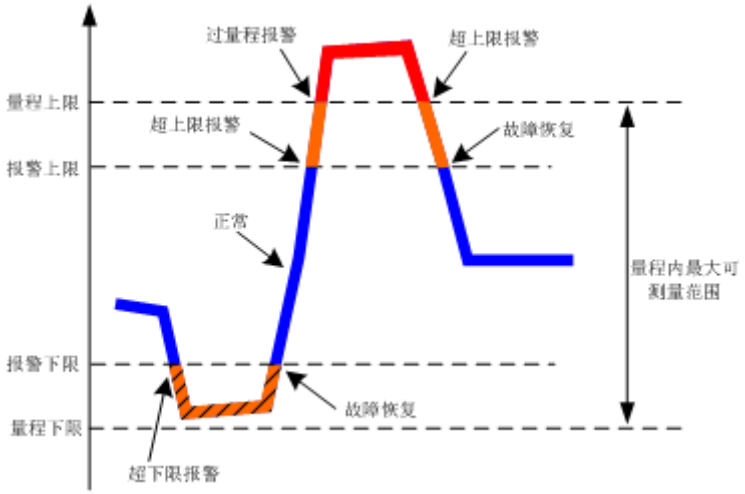
对于标准热电阻，无论 LK430C 的输出数据格式选择温度值还是电阻值，组态时超限报警上限和报警下限均设定为代表温度值的正整数代码，该报警上下限温度代码计算公式如下：

- 报警上限代码=报警上限温度值×10+10000
- 报警下限代码=报警下限温度值×10+10000

报警上限温度和报警下限温度的温度单位（摄氏度或华氏度），要与模块选用的温度单位(通过参数 **Temperature Units** 选择，默认摄氏度)保持一致。

报警上限、报警下限设定范围：6720~25980，且报警上限必须大于报警下限，否则 LK430C 不能正确上报诊断信息。

LK430C 模块只在超限发生和超限恢复时分别上报一次诊断数据。LK430C 模块是否进行超限报警、各个通道的报警上限和报警下限，组态可选。



LK430C 超限报警示意图

LK430C 组态成不同的测量数据格式，发生超限时模块的诊断处理方式有所不同，如表所示。信号恢复至正常范围内，通道诊断区上报 0xA0。

LK430C 超限报警的处理方式

测量数据格式	超限类型	超限处理
输出温度值	超上限	通道诊断区上报故障值 0xA7 ≤量程上限，通道测量数据上报当前温度值代码 ＞量程上限，通道测量数据上报量程内允许测量最大温度值代码
	超下限	通道诊断区上报故障值 0xA8 ≥量程下限，通道测量数据上报当前温度值代码 ＜量程下限，通道测量数据上报量程内允许测量最小温度值代码
输出电阻值	超上限	通道诊断区上报故障值 0xA7 ≤量程上限，通道测量数据上报当前电阻值代码 ＞量程上限，通道测量数据上报 0xFFFF
	超下限	通道诊断区上报故障值 0xA8 ≥量程下限，通道测量数据上报当前电阻值代码 ＜量程下限，通道测量数据上报 0x0000

7.3.3.5 设置故障输出模式

1. 概述

输出模块发生故障时，可以独立设置每个通道的故障输出模式，有两种输出模式供选择：

- 输出保持：输出保持故障前的值
- 输出故障模式安全值：输出采用用户组态的安全值。选择该模式后，需要为通道组态一个安全值。

各模块触发的故障类型请参见下表：

触发故障输出模式的故障类型

故障类型	LK710C/LK716C	LK511C/LK512C
系统电源掉电	√	×
DP 双网故障	√	√
模块单片机故障	√	√
通讯模块故障	√	√
控制器故障	√	√

2. 要求

- 通道已使能
- “用户参数”界面已打开

3. 步骤

设置故障输出模式，请按以下步骤操作：

（1） 选择“CHn Fault Mode State”参数

（2） 参数值下拉菜单中选择需要的输出模式，如下图所示：

设置输出故障模式

4. 下一步

设置故障输出模式完成后，请设置安全预设值。

7.3.3.6 设置故障安全值

1. 概述

模块发生故障时，通道值采用用户设定的安全值。

- DO 模块可设置为断开或闭合
- AO 模块可根据各模块量程设置对应的码值

2. 要求

- 通道已使能
- “用户参数”界面已打开
- 故障输出模式设置为“Fault Mode State”

3. 步骤

设置安全模式输出值，请按以下步骤操作：

（1） 选择“CHn Fault Mode”参数

（2） 参数值设置为需要的安全值，以 DO 模块为例，如下图所示：

设置输出值

4. 结果

模块发生故障时，输出用户组态的安全值。

7.3.3.7 设置断线上报值

1. 概述

LK430C/LK432C 模块具有断线检测功能，输入通道的任意一根信号线缆脱落，模块向控制器上报断线报警。

当某个通道发生断线时：

- 通道诊断区上报故障值 0xA6。
- 通道上报组态设定值。选择不同的数据格式，发生断线时上报的通道测量数据不同，如下表所示。

断线恢复后，通道诊断区上报 0xA0。LK430C 模块只在发生断线和断线恢复时分别上报一次诊断数据。

断线时通道上报数据说明

用户参数		测量数据说明
Data Format（数据格式）	Line Break Value（断线上报值）	
Code	0x0000	通道测量数据上报 0x0000，即上报量程下限值
	0xFFFF	通道测量数据上报 0xFFFF，即上报量程上限值
	Hold（默认）	通道测量数据保持断线前的正常数据
Temperature	0x0000	以通道 1 为例，端子 1、3 和 5： 1 端或 3 端断线，通道测量上报量程范围内最小温度码值 5 端断线，通道测量数据上报量程范围内最大温度码值
	0xFFFF	
	Hold（默认）	通道测量数据保持断线前的正常数据

2. 要求

“用户参数”界面已打开

3. 步骤

要设置断线上报值，请按以下步骤操作：

（1） 选择“Line Break Value”参数。

（2） 双击参数值，下拉框中选择需要上报的数据类型。

设置断线上报值

7.3.3.8 设置测量数据上报格式

1. 概述

LK430C/LK432C 模块可选择测量数据的上报格式，包括输出电阻值代码、温度值或电阻值，通过组态进行设置。

- Code:输出电阻值代码
- Temperature：输出温度值代码
- Resistance：输出电阻值（仅 LK432C 支持）

当所使用的热电阻类型，不在支持类型范围内时，请选择测量数据输出格式为电阻值。

LK430C/LK432C 模块每个通道的测量数据用 2 个字节的正整数（十进制 0~65535）表示，测量数据和物理量之间的转换公式如下：

- 选择输出电阻值：

  电阻值（Ω）=（阻值代码/65535）×满量程电阻值+量程内最小可测量电阻值；其中，满量程电阻值=量程范围内最大可测量电阻-最小可测量电阻。

  例如测量范围中，Cu50 最大可测量电阻范围 1~121.75 Ω，则满量程电阻值=121.75-1=120.75。

- 选择输出温度值：

  温度值（摄氏度或华氏度）=（温度代码-10000）/10

用户根据转换公式做简单运算后，即可得到现场实际的温度值或电阻值。

2. 要求

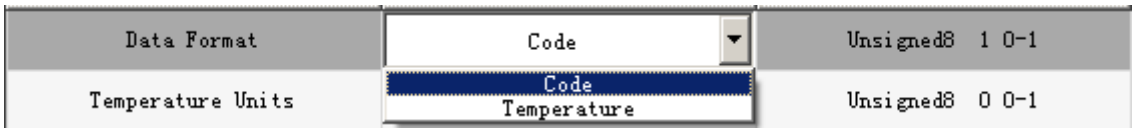
“用户参数”界面已打开

3. 步骤

要设置测量数据上报格式，请按以下步骤操作：

（1） 选择“Data Format”参数。

（2） 双击参数值，下拉框中选择“Code”或“Temperature”。



设置数据格式

7.3.3.9 组态热电阻类型

1. 概述

LK430C/LK432C 支持的各种标准热电阻，如表所示。

当用到一个下表中没有的特殊类型电阻时，选择测量数据输出格式为电阻值，就可以实现测量。

量程组态时选择下表中一个与该特殊电阻阻值范围相近的标准热电阻作为代用量程。比如：有一个阻值为 350Ω 的电阻需要测量，这时可以选择 Ni618 100Ω、Ni618 120Ω、Pt385 100Ω 或 Pt3916 100Ω 中之一作为代用量程。

LK430C/LK432C 支持的标准热电阻及测量范围一览表

热电阻类型	热电阻 测温范围（℃）	热电阻对应的 阻值范围（Ω）	量程 代号	最大可测量阻值范围（Ω）
Copper427 10 Ω	-200℃~260℃	3.69980~21.1574	192	1~121.75
Chinese_Cu 50 Ω	-50℃~150℃	39.243~82.136	193	
Nikel618 100 Ω	-60℃~250℃	69.5204~343.584	194	1~487
Nikel618 120 Ω	-60℃~250℃	83.4245~412.301	195	
Platinum385 100 Ω	-200℃~870℃	18.5201~396.311	196	
Platinum3916 100 Ω	-200℃~630℃	16.9960~327.744	197	
Nikel618 200 Ω	-60℃~250℃	139.041~687.168	198	2~1000
Nikel672 120 Ω	-80℃~320℃	66.6000~568.407	199	
Platinum385 200 Ω	-200℃~870℃	37.0402~792.622	200	
Platinum3916 200 Ω	-200℃~630℃	33.992~655.488	201	
Nikel618 500 Ω	-60℃~250℃	347.602~1717.92	202	4~2000
Platnium385 500 Ω	-200℃~870℃	92.6005~1981.56	203	
Platnium3916 500 Ω	-200℃~630℃	84.98~1638.72	204	

Platnium385 1000 Ω	-200℃~870℃	185.201~3963.11	205	8~4020
Platnium3916 1000 Ω	-200℃~630℃	169.960~3277.44	206	

2. 要求

“用户参数”界面已打开

3. 步骤

要设置热电阻类型，请按以下步骤操作：

（1） 选择“CHn Sensor Type”参数。

（2） 双击参数值，下拉框中选择热电阻型号。

请与实际连接热电阻型号保持一致。

7.3.3.10 设置热电阻温标

1. 概述

热电阻温标可组态配置：

- Celsius，摄氏温标
- Fahrenheit，华氏温标

2. 要求

“用户参数”界面已打开

3. 步骤

要设置热电阻温标，请按以下步骤操作：

（1） 选择“Temperature Units”参数。

（2） 双击参数值，下拉框中选择摄氏或华氏温标。

7.3.3.11 设置通道量程

1. LK410C 量程范围

输入信号与机器代码值对应关系

量程范围	对应十进制码值	扩展量程范围	对应十进制码值
0~5V	0~27648	0~5.92V	0~32767
1~5V	0~27648	0~5.7V	-6912~32767
0~10V	0~27648	0~10V	0~27648
-5V~+5V	-27648~27648	-5.92~5.92V	-32768~32767
-10V~+10V	-27648~27648	-10V~+10V	-27648~27648

信号与码值的换算公式：码值=（信号测量值-信号量程下限）×码值量程/信号量程+码值下限，其中：

- 码值量程=码值量程上限-码值量程下限
- 信号量程=信号量程上限-信号量程下限

举例：量程组态为 1~5V，对应码值为 0~27648，当前测量信号为 3V，则计算测量信号对应的码值为：

码值=（3-1）×27648/4=13824

2. LK411C 量程范围

输入信号与机器代码值对应关系

量程范围	对应十进制码值	扩展量程范围	对应十进制码值
0~20 mA	0~27648	0~23.7mA	0~32767
4~20 mA	0~27648	0~22.96mA	-6912~32767

信号与码值的换算公式：码值=（信号测量值-信号量程下限）×码值量程/信号量程+码值下限，其中：

- 码值量程=码值量程上限-码值量程下限
- 信号量程=信号量程上限-信号量程下限

举例：量程组态为 0~20 mA，对应码值为 0~27648，当前测量信号为 8mA，则计算测量信号对应的码值为：

码值=8×27648/20=11059

3. LK412C 量程范围

输入信号与机器代码值对应关系

量程范围		对应十进制码值
-10.25~10.25 V	-10.25~0 V	32768~65,535
	0~10.25 V	0~32767
0~10.25 V		0~65535
0~5.125 V		0~65535
0～20.58 mA		0~65535
4～20.58 mA		0~65535

信号与码值的换算公式：码值=（信号测量值-信号量程下限）×码值量程/信号量程+码值下限，其中：

- 码值量程=码值量程上限-码值量程下限

- 信号量程=信号量程上限-信号量程下限

举例：量程组态为 0~10.25 V，对应码值为 0~65535，当前测量信号为 5V，则计算测量信号对应的码值为：

码值=5×65535/10.25=31968

4. LK511C 量程范围

输出信号与机器代码值对应关系

量程范围	对应十进制码值
0~21 mA	0~65535
4~20 mA	0~65535

信号与码值的换算公式：码值=（信号测量值-信号量程下限）×码值量程/信号量程+码值下限，其中：

- 码值量程=码值量程上限-码值量程下限
- 信号量程=信号量程上限-信号量程下限

举例：量程组态为 4~20 mA，对应码值为 0~65535，当前测量信号为 8mA，则计算测量信号对应的码值为：

码值=（8-4）×65535/16=16384

5. LK512C 量程范围

输出信号与机器代码值对应关系

信号类型	量程范围	对应十进制码值	扩展量程范围	对应十进制码值
电流信号	0~20mA	0~27648	0~23.69mA	0~32767
	4~20mA	0~27648	4~22.96mA	0~32767
电压信号	0~5V	0~27648	0~5.92V	0~32767
	1~5V	0~27648	1~5.74V	0~32767
	0~10V	0~27648	0~11.85V	0~32767
	-5V~+5V	-27648~27648	-5.92V~+5.92V	-32768~32767
	-10V~+10V	-27648~27648	-11.85V~+11.85V	-32768~32767

信号与码值的换算公式：码值=（信号测量值-信号量程下限）×码值量程/信号量程+码值下限，其中：

- 码值量程=码值量程上限-码值量程下限
- 信号量程=信号量程上限-信号量程下限

举例：量程组态为 1~5V，对应码值为 0~27648，当前测量信号为 3V，则计算测量信号对应的码值为：

码值=（3-1）×27648/4=13824

6. 设置通道量程

第1步 概述

对于模拟量信号，根据通道连接的仪表信号类型和范围，为每个通道组态对应的量程。

第2步 要求

- 通道已使能
- “用户参数”界面已打开

第3步 步骤

设置通道量程，请按以下步骤操作：

（1） 选择“CHn Input Range”

（2） 参数值设置为仪表量程，如下图所示：

参数名称	参数值	参数说明
CH1 Input Range	1~5V	Unsigned8 0 0,3,4,8,22
CH2 Input Range	0~5V 1~5V 0~10V -5~+5V -10~+10V	Unsigned8 0 0,3,4,8,22
CH3 Input Range		Unsigned8 0 0,3,4,8,22

设置通道量程

7.3.3.12 设置工频滤波参数

1. 概述

对于 AI 信号，为消除现场信号工频干扰，需要组态通道滤波参数。

设置原则：

- 220VAC 工频干扰，滤波参数设置为 50Hz。
- 在使用 60Hz 交流电的场合，滤波参数设置为 60Hz。
- 在响应要求高，不考虑干扰的场合，可以设置无滤波。

2. 要求

- 通道已使能
- “用户参数”界面已打开

3. 步骤

设置工频滤波参数，请按以下步骤操作：

（1） 选择“Filter Mode”

（2） 双击“参数值”项，在下拉列表中选择需要的滤波值，如下图所示：

设置滤波参数

7.3.3.13 设置数字滤波参数

1. 要求

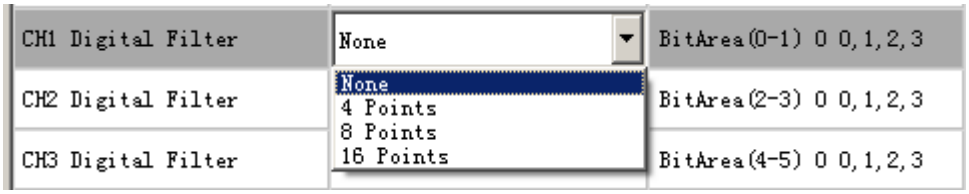
“用户参数”界面已打开

2. 步骤

要设置数字滤波，请按以下步骤操作：

（1） 选择“Chn Digital Filter”参数。

（2） 双击参数值，下拉框中选择滤波值，如下图所示。



设置滤波参数

7.3.3.14 设置通道数据采集周期

1. 概述

通过组态采样周期，对输入信号进行不同强弱的滤波。采样周期越大、信号越平滑，滤波效果越好，相对信号刷新越慢。需要根据对信号响应速度和质量要求，选择对应的采集周期，同时，也可根据任务属性中配置的“运行周期”（IEC）时间选择建议的采集周期。

2. 要求

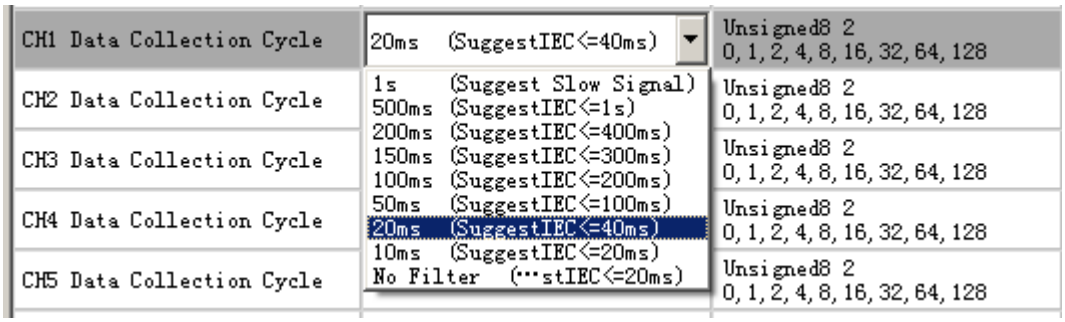
- 通道已使能
- “用户参数”界面已打开

3. 步骤

设置通道采集周期，请按以下步骤操作：

（1） 选择“CHn Data Collection Cycle”

（2） 参数值设置为需要的周期值，如下图所示：



设置采集周期

7.3.4 通道数据说明

7.3.4.1 LK410C 通道数据

LK410C 输入数据占用 10 个字。8WORD 通道输入数据，2 字节数据同步状态、2 字节通道质量位。

通道数据一览表

区域定义	数据类型	含义
输入数据（%IW）	INT	通道 1~通道 8 输入数据，每个通道占 1WORD
输入数据（%IW）	WORD	共 16bits，分别代表 16 通道数据同步状态 0：表示数据没有同步，此数据无效 1：表示数据同步，此数据有效
输入数据（%IB）	BYTE	bit0~bit7：分别代表通道 1~通道 8 信号质量，第二字节未使用 0：通道数据好 1：通道数据坏

7.3.4.2 LK411C 通道数据

LK411C 输入数据占用 10 个字。8WORD 通道输入数据，2 字节数据同步状态、2 字节通道质量位。

通道数据一览表

区域定义	数据类型	含义
输入数据（%IW）	INT	通道 1~通道 8 输入数据，每个通道占 1WORD
输入数据（%IW）	WORD	共 16bits，分别代表 16 通道数据同步状态 0：表示数据没有同步，此数据无效 1：表示数据同步，此数据有效
输入数据（%IB）	BYTE	bit0~bit7：分别代表通道 1~通道 8 信号质量，第二字节未使用 0：通道数据好 1：通道数据坏

7.3.4.3 LK412C 通道数据

LK412C 输入数据占用 6 个字,分别为 6 个通道输入数据。

通道数据一览表

区域定义	数据类型	含义
输入数据（%IW）	WORD	通道 1~通道 6 输入数据，每个通道占 1WORD

7.3.4.4 LK430C 通道数据

LK430C 输入数据占用 6 个字，分别采集 6 通道数据。

通道数据一览表

区域定义	数据类型	含义
输入数据（%IW）	WORD	通道 1~通道 6 输入数据，每个通道占 1WORD

7.3.4.5 LK432C 通道数据

LK432C 输入数据占用 17 个字节。8WORD 通道输入数据，1 字节通道质量位。

通道数据一览表

区域定义	数据类型	含义
输入数据（%IW）	WORD	通道 1~通道 8 输入数据，每个通道占 1WORD
输入数据（%IX）	BOOL	bit0~bit7：分别代表通道 1~通道 8 信号质量 0：通道数据好 1：通道数据坏

7.3.4.6 LK442C 通道数据

LK442C 输入数据占用 6 个字,分别为 6 个通道输入数据。

通道数据一览表

区域定义	数据类型	含义
输入数据（%IW）	WORD	通道 1~通道 6 输入数据，每个通道占 1WORD

7.3.4.7 LK511C 通道数据

LK511C 输出数据占用 6 个字。4WORD 通道输出数据，4 字节通道回读数据。

通道数据一览表

区域定义	数据类型	含义
输出数据（%QW）	WORD	通道 1~通道 4 输出数据，每个通道占 1WORD
回读数据（%IB）	BYTE	通道 1~通道 4 回读数据，每个通道占 1BYTE

7.3.4.8 LK512C 通道数据

LK512C 输出数据占用 8 个字，输出 8 通道电流或电压信号。输入数据占 2 个字节，读取模块主从状态。

通道数据一览表

区域定义	数据类型	含义
输出数据（%QW）	INT	通道 1~通道 8 输出数据，每个通道占 1WORD
输入数据（%IW）	WORD	读取模块主从状态 WORD 取值 0：当前模块为主模块 WORD 取值 1：当前模块为从模块

7.3.4.9 LK610C 通道数据

LK610C 输入数据占用 7 个字节。1 字节 SOE 事件状态，2 字节通道输入数据，2 字节数据同步状态、2 字节通道质量位。

通道数据一览表

区域定义	数据类型	含义
输入数据（%IB）	BYTE	SOE 事件状态 6：通道有 SOE 事件 0：无 SOE 事件 通过指令 sysReadSOE(读取 SOE 记录)可查看发生 SOE 事件的通道
输入数据（%IX）	BOOL	通道输入值，Bit0~Bit15 对应通道 1~通道 16 1：闭合 0：断开
输入数据（%IW）	WORD	共 16bits，分别代表 16 通道数据同步状态 0：表示数据没有同步，此数据无效 1：表示数据同步，此数据有效
输入数据（%IB）	BYTE	共 16bits，分别代表 16 通道信号质量 0：通道数据好 1：通道数据坏

7.3.4.10 LK616C 通道数据

LK616C 输入数据占用 8 个字节。4 字节通道输入数据，4 字节通道质量位。

通道数据一览表

区域定义	数据类型	含义

输入数据（%IX）	BOOL	通道 1~通道 32 输入数据，每个通道占 1 bit
输入数据（%IX）	BOOL	通道 1~通道 32 信号质量，每个通道占 1 bit 0：通道数据好 1：通道数据坏

7.3.4.11 LK631C 通道数据

LK631C 输入数据占用 14 bit。

通道数据一览表

区域定义	数据类型	含义
输入数据（%IX）	BOOL	通道 1~通道 14 输入数据，每个通道占 1 bit

7.3.4.12 LK710C 通道数据

LK710C 输出数据占用 2 个字节，控制 16 通道输出的闭合和断开。

通道数据一览表

区域定义	数据类型	含义
输出数据（%QX）	BOOL	通道输出值，Bit0~Bit15 对应通道 1~通道 16 1：闭合 0：断开

7.3.4.13 LK716C 通道数据

LK716C 输出数据占用 4 个字，输出通道数据。输入数据占 4 个字节，读取通道质量位。

通道数据一览表

区域定义	数据类型	含义
输出数据（%QX）	BOOL	通道 1~通道 32 输出数据，每个通道占 1 bit
输入数据（%IX）	BOOL	通道 1~通道 32 信号质量，每个通道占 1 bit 0：通道数据好 1：通道数据坏

7.4 组态 Modbus RTU 主站

7.4.1 概述

LK221C/LK221CT1 控制器可以做 Modbus RTU 主站或从站与外部设备进行通讯，有两个 RS485 串口接口，用于连接通讯线缆。

7.4.2 添加 Modbus RTU 主站协议

7.4.2.1 概述

控制器可作为 Modbus RTU 主站与外部设备进行数据通讯，有两个 RS485 串口，用于连接通讯线缆。需要为串口配置主站协议。

7.4.2.2 要求

工程已打开

7.4.2.3 步骤

要添加 Modbus RTU 主站协议，请按以下步骤操作：

（1） 在 COM 口树节点右键菜单中，选择“添加”命令。

将弹出“添加”对话框。

（2） 选择“ModbusRTU_Master”协议。

（3） 单击**确定**按钮。

“COM”节点下新增 ModbusRTU_Master 节点。

7.4.2.4 下一步

添加 ModbusSlave_RTU 从站设备

7.4.2.5 参考信息

关联任务

7.4.3 添加 ModbusSlave_RTU 从站设备

7.4.3.1 概述

需要在 Modbus RTU 主协议下配置 ModbusSlave_RTU 从站设备。

7.4.3.2 要求

已添加“ModbusRTU_Master”协议

7.4.3.3 步骤

要添加 ModbusSlave_RTU 从站设备，请按以下步骤操作：

（1） 在“ModbusRTU_Master”树节点右键菜单中，选择“添加”命令。

将弹出“添加”对话框。

（2） 选择“ModbusSlave_RTU”从站设备。

（3） 输入从站设备地址。

（4） 单击**确定**按钮。

7.4.3.4 下一步

配置 ModbusSlave_RTU 从站通讯参数。

7.4.4 配置串口通讯参数

7.4.4.1 要求

工程已打开

7.4.4.2 步骤

要配置串口通讯参数，请按以下步骤操作：

（1） 在“设备参数”签页中，单击参数下拉按钮，选择参数值。

参数	参数值	默认值	设置说明
波特率	9600bps、19200bps、38400bps、57600bps、115200bps	38400bps	设置时，保持主从站波特率、校验方式、停止位、数据位的一致。
校验方式	奇校验、偶校验、无校验	无校验	
停止位	1 位、2 位	1 位	
数据位	8 位	8 位	
通信类型	RS485	RS485	

7.4.5 配置 Modbus RTU 主站通讯参数

7.4.5.1 要求

“ModbusRTU_Master”协议窗口已打开

7.4.5.2 步骤

要配置 Modbus RTU 主站通讯参数，请按以下步骤操作：

（1） 在“设备参数”签页中，单击参数的增大或减小按钮，调整到设定值。

参数	参数值	默认值	设置说明
响应超时（ms）	10~65535	1000	主站发送请求帧后所允许的从站延时应答时间
重连次数	0~10	0	从站应答异常后主站重新发送请求的次数
ModbusRTU 报文结束时延（ms）	0~32767	0	接收 ModbusRTU 报文时，增加报文结束时延。建议保持默认值
串口通信类型	RTU、	RTU	--

	ASCII		
协议关联的任务名	任务名	No Task	为协议指定一个运行的 IEC 任务。建议不要关联到用户任务，您需要单独创建一个新任务，仅用于执行 Modbus 通讯协议

7.4.6 配置 ModbusSlave_RTU 从站通讯参数

7.4.6.1 要求

“ModbusSlave_RTU”从协议窗口已打开

7.4.6.2 步骤

要配置 ModbusSlave 从站通讯参数，请按以下步骤操作：

（1） 在“设备参数”签页中，单击参数的增大或减小按钮，调整到设定值。

参数	参数值	默认值	设置说明
从站地址	1~247	1	主站请求的从站地址
响应超时（ms）	0~65,535	0	主站发送请求帧后，所允许从站延时应答的时间，单位：毫秒 默认情况：从站默认响应超时时间为零，此时，以主站配置的“响应超时”时间为准 用户可单独配置某一个从站的响应超时时间，如果该参数配置大于零，则该从站的超时时间以当前从站配置为准

7.5 组态 Modbus RTU 从站

7.5.1 添加 Modbus RTU 从站协议

7.5.1.1 概述

当控制器 LK221C/LK221CT1 作从站时，需要配置 ModbusRTU_Slave 协议与主站通讯，可支持 32 个主站同时访问控制器。

7.5.1.2 要求

工程已打开

7.5.1.3 步骤

要添加 Modbus RTU 从站协议，请按以下步骤操作：

（1） 在 COM 口树节点右键菜单中，选择“添加”命令。

将弹出“添加”对话框。

（2） 选择“ModbusRTU_Slave”协议。

（3） 单击**确定**按钮。

“COM”节点下新增 ModbusRTU_Slave 节点。

7.5.1.4 下一步

配置从站通讯参数

7.5.1.5 参考信息

关联任务

7.5.2 配置串口通讯参数

7.5.2.1 要求

工程已打开

7.5.2.2 步骤

要配置串口通讯参数，请按以下步骤操作：

（1） 在“设备参数”签页中，单击参数下拉按钮，选择参数值。

参数	参数值	默认值	设置说明
波特率	9600bps、19200bps、38400bps、57600bps、115200bps	38400bps	设置时，保持主从站波特率、校验方式、停止位、数据位的一致。
校验方式	奇校验、偶校验、无校验	无校验	
停止位	1 位、2 位	1 位	
数据位	8 位	8 位	
通信类型	RS485	RS485	

7.5.3 配置从站通讯参数

7.5.3.1 要求

“ModbusRTU_Slave”协议窗口已打开

7.5.3.2 步骤

要配置 Modbus 从站通讯参数，请按以下步骤操作：

（1） 在“设备参数”签页中，单击参数的增大或减小按钮，调整到设定值。

参数	参数值	默认值	设置说明
从站地址	1~247	1	从站设备地址
ModbusRTU 报文结束时	0~32767	0	接收 ModbusRTU 报文时，增加报文结束时延。建议保持默认值

延（ms）			
协议关联的任务名	任务名	No Task	为协议指定一个运行的 IEC 任务。建议不要关联到用户任务，您需要单独创建一个新任务，仅用于执行 Modbus 通讯协议

7.6 组态 Modbus TCP 主站

7.6.1 添加 Modbus TCP 主站协议

7.6.1.1 概述

控制器和 LK246C 模块均可作为 Modbus TCP 主站或从站与外部设备通讯,在控制器和 LK246C 模块的【Ethernet】节点下完成 Modbus TCP 主从站的组态。控制器和 LK246C 模块添加 ModbusTCP_Master 数量约束如下表：

ModbusTCP_Master 可配置最大数量表

参数	LK221C（标准控制器）	LK221CT1(可信标准控制器)	LK226C（冗余控制器）	LK226CT1（可信冗余控制器）	LK246C（以太网通信处理器模块）
Modbus TCP 主协议可配置最大数量	1	1	8	8	16
Modbus TCP 主协议与 Modbus TCP 从协议可配置最大总数量	2	2	8	8	16

7.6.1.2 要求

工程已打开

7.6.1.3 步骤

要添加 Modbus TCP 主站协议，请按以下步骤操作：

（1） 在“Ethernet”树节点右键菜单中，选择“添加”命令。

将弹出“添加”对话框。

（2） 选择“ModbusTCP_Master”协议。

（3） 单击**确定**按钮。

“Ethernet”节点下新增 ModbusTCP_Master 节点。

7.6.1.4 下一步

添加 ModbusSlave_TCP 协议。

7.6.1.5 参考信息

添加 ModbusSlave_TCP 从站设备

配置 Modbus 主站通讯参数

配置 ModbusSlave 从站通讯参数

关联任务

7.6.2 添加 ModbusSlave_TCP 从站设备

7.6.2.1 概述

当添加 ModbusSlave_TCP 从站设备时，ModbusSlave_TCP 连接可配置数量和指令数量约束如下表:

ModbusSlave_TCP 连接可配置最大数量表

参数	LK221C（标准控制器）	LK221CT1(可信标准控制器)	LK226C（冗余控制器）	LK226CT1 （可信冗余控制器）	LK246C（以太网通信处理器模块）
单个 Modbus TCP 从协议连接可配置最大数量	32	32	32	32	32
Modbus TCP 从协议连接配置总数量	32	32	32	32	64
单个 Modbus TCP 从协议连接可配置最大指令条数	32	32	32	32	32

7.6.2.2 要求

已添加“ModbusTCP_Master”协议

7.6.2.3 步骤

要添加 ModbusSlave_TCP 从站设备，请按以下步骤操作：

（1） 在“ModbusTCP_Master”树节点右键菜单中，选择“添加”命令。

将弹出“添加”对话框。

（2） 选择“ModbusSlave_TCP”从站设备。

（3） 输入从站 IP 地址。

（4） 单击**确定**按钮。

7.6.2.4 下一步

配置 ModbusSlave 从站通讯参数。

7.6.2.5 参考信息

添加 Modbus TCP 主站协议

配置 Modbus 主站通讯参数

配置 ModbusSlave 从站通讯参数

添加 Modbus 指令

7.6.3 配置 Modbus 主站通讯参数

7.6.3.1 要求

“ModbusTCP_Master”协议窗口已打开

7.6.3.2 步骤

要配置 Modbus 主站通讯参数，请按以下步骤操作：

（1） 在“设备参数”签页中，单击参数的增大或减小按钮，调整到设定值。

参数	参数值	默认值	设置说明
响应超时（ms）	10~2,147,483,000	1000	ModbusTCP 主站发送请求帧后所允许的从站延时应答时间
EthernetSocket 超时时间（ms）	10~2,147,483,000	100	TCP/IP 连接 Socket 超时时间
重连次数	0~10	0	从站应答异常后主站重新发送请求的次数
协议关联的任务名	任务名	No Task	为协议指定一个运行的 IEC 任务。建议不要关联到用户任务，您需要单独创建一个新任务，仅用于执行 Modbus 通讯协议 如果当前协议配置的指令数量较多，则当前协议关联的任务的实际执行周期可能大于任务设定周期 对于 LK246C 模块，协议关联的任务周期为通信数据的刷新周期

7.6.4 配置 ModbusSlave 从站通讯参数

7.6.4.1 要求

“ModbusSlave_TCP”从协议窗口已打开

7.6.4.2 步骤

要配置 ModbusSlave 从站通讯参数，请按以下步骤操作：

（1） 在“设备参数”签页中，单击参数的增大或减小按钮，调整到设定值。

参数	参数值	默认值	设置说明
IP 地址	与从站一致	0.0.0.0	主站所连接从站的 IP 地址，根据从站设备 IP 地址进行设置

单元 ID	1~247	1	Modbus TCP 协议单元 ID，请与从站设备的组态保持一致
端口号	1~65,535	502	Modbus TCP 协议端口号，请与从站设备的组态保持一致
响应超时	0~2,147,483,000	0（ms）	主站发送请求帧后，所允许从站延时应答的时间，单位：毫秒 默认情况：从站默认响应超时时间为零，此时，以主站配置的“响应超时”时间为准 用户可单独配置某一个从站的响应超时时间，如果该参数配置大于零，则该从站的超时时间以当前从站配置为准

7.6.5 组态故障切换模式

7.6.5.1 概述

LK246C 模块发生网口故障时，触发主备切换。当软件中组态了触发切换的故障类型，当该故障发生时，将触发主备切换。

可组态的故障切换模式如下：

- 不切换
- 网口 1 故障
- 网口 2 故障
- 网口 1 或网口 2 故障
- 网口 1 且网口 2 故障

7.6.5.2 要求

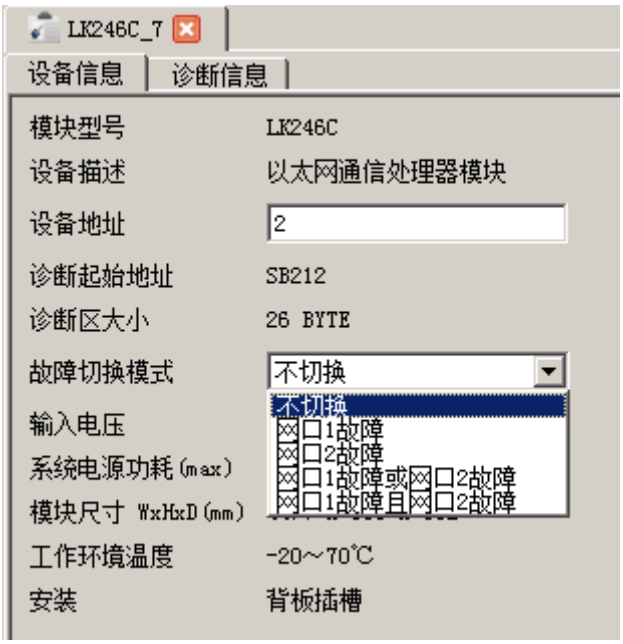
控制器下已添加 LK246C 模块

7.6.5.3 步骤

要组态故障切换模式，请按以下步骤操作：

（1） 双击控制器树节点下 LK246C 模块。

将打开 LK246C 配置界面。



LK246C 设备配置窗口

（2） 在“设备信息”标签中，单击“故障切换模式”下拉按钮，选择触发切换的故障类型。

7.7 组态 Modbus TCP 从站

7.7.1 添加 Modbus TCP 从站协议

7.7.1.1 概述

当添加 ModbusTCP_Slave 协议时，ModbusTCP_Slave 可配置数量、ModbusTCP_Slave 支持外部连接个数约束如下表：

ModbusTCP_Slave 可配置最大数量个数表

参数	LK221C（标准控制器）	LK221CT1(可 信标准控制器)	LK226C（冗余控制器）	LK226CT1 （ 可信冗余控制器）	LK246C（以太网通信处理器模块）
Modbus TCP 从协议可配置最大数量	1	1	8	8	16
Modbus TCP 主协议与 Modbus TCP 从协议可配置最大总数量	2	2	8	8	16

ModbusTCP_Slave 支持外部连接数量表

参数	LK221C（标准控制器）	LK221CT1(可信标准控制器)	LK226C（冗余控制器）	LK226CT1 （ 可信冗余控制器）	LK246C（以太网通信处理器模块）
单个 Modbus TCP 从	32	32	32	32	32

协议支持最大外部连接数量					
Modbus TCP 从协议支持外部连接总数量	32	32	32	32	64

7.7.1.2 要求

工程已打开

7.7.1.3 步骤

要添加 Modbus TCP 从站协议，请按以下步骤操作：

（1） 在“Ethernet”树节点右键菜单中，选择“添加”命令。

将弹出“添加”对话框。

（2） 选择“ModbusTCP_Slave”协议。

（3） 单击**确定**按钮。

“Ethernet”节点下新增 ModbusTCP_Slave 节点。

7.7.1.4 下一步

配置 Modbus 从站通讯参数

7.7.1.5 参考信息

关联任务

7.7.2 配置 Modbus 从站通讯参数

7.7.2.1 要求

“ModbusTCP_Slave”协议窗口已打开

7.7.2.2 步骤

要配置 Modbus 从站通讯参数，请按以下步骤操作：

（1） 在“设备参数”签页中，单击参数的增大或减小按钮，调整到设定值。

参数	参数值	默认值	设置说明
单元 ID	1~247	1	Modbus TCP 协议单元 ID
端口号	1~65,535	502	Modbus TCP 协议端口号
超时时间	0~2,147,483,000	2000（ms）	从站没有接收到主站发送数据的时间间隔，超过设定的时间，则从站断开与主站的通讯链接，单位：毫秒
远端设备 IP	连接服务器 IP	0.0.0.0	所连接服务器 IP 地址

远端设备个数	1~32	32	服务器下所允许连接的主站个数
协议关联的任务名	任务名	No Task	为协议指定一个运行的 IEC 任务。建议不要关联到用户任务，您需要单独创建一个新任务，仅用于执行 Modbus 通讯协议 从站协议对主站请求的响应速度与协议关联的任务周期有关，协议关联的任务周期越小，从站的响应速度越快 如果与当前从站建立连接的主站数量较多或者通信数据量较大，该从站关联的任务的实际执行周期可能大于任务设定周期 LK246C 模块作为从站时，如果主站为读指令，协议关联的任务周期为通讯数据刷新周期。主站为写指令时，协议关联的任务周期会影响从站响应速度，协议关联的任务周期越短，从站响应速度越快

7.7.3 功能码定义

控制器和 LK246C 模块作为从站时，支持 Modbus 功能码 01、02、03、04、05、06、15、16、23。

模块作为从站时的扩展协议，03 号功能码每条指令最大读取 4000 寄存器，16 号功能码每条指令最大写入 2000 寄存器。

支持的功能码列表

功能码		数据区	功能
01	读线圈 (0xxxx, 01H)	0xxxx	读 1~2000（0x07D0）个连续的逻辑线圈的状态（ON/OFF）
02	读离散量输入 (1xxxx, 02H)	1xxxx	读 1~2000（0x07D0）个连续的离散量输入的状态（ON/OFF）
03	读保持寄存器 (4xxxx, 03H)	4xxxx	读 1~4000（0x007D）个连续的保持寄存器的二进制值
04	读输入寄存器 (3xxxx, 04H)	3xxxx	读 1~125（0x007D）个连续的输入寄存器的二进制值
05	写单个线圈 (0xxxx, 05H)	0xxxx	强制单个线圈输出 ON 或 OFF
06	写单个寄存器 (4xxxx, 06H)	4xxxx	强制单个保持寄存器中写入二进制值
15	写多个线圈 (0xxxx, 0FH)	0xxxx	强制一组线圈（1~1968 个）的任意一个线圈为 ON 或 OFF
16	写多个寄存器 (4xxxx, 10H)	4xxxx	强制连续寄存器（1~2000 个）中写入二进制值
23	读/写多个寄存器(4xxxx, 17H)	4xxxx	在连续寄存器（1~118 个）中读取或写入二进制值

7.7.4 数据区与 Modbus 地址映射

控制器作从站时，Modbus TCP/Modbus RTU 通讯协议可以访问的从站数据区包括：输入区（I 区）、输出区（Q 区）、中间区（M 区）。

I 区和 Q 区 Modbus 地址范围 0~3000，通讯点数 3000。M 区 Modbus 地址范围 3000~65535，通讯点数 62535。主站指令偏移地址小于 3000，则访问从站 I 区和 Q 区数据，如果偏移地址大于 3000，则访问从站 M 区数据。

当访问从站 I 区、Q 区数据时，主站指令偏移+数据长度不能大于 3000。

这三个数据区，均可通过 BOOL 型或者 WORD 型数据访问。这些数据区与 Modbus 地址的映射关系如表所示。

数据区与 MODBUS 地址映射关系

数据区		类型	地址范围	MODBUS 地址	映射关系	X（寄存器类型）选择
I 区	%IX	BOOL	%IX0.0、…%IX0.7 %IX1.0、…%IX1.7 …%IX374.7	X0000、…X0007 X0008、…X0015 …X2999	IXm.n：m*8+n	只读，X 选 1
	%IW	WORD	%IW0、%IW2、…%IW5998	X0000、X0001、…X2999	IWm：m/2	只读，X 选 3
Q 区	%QX	BOOL	%QX0.0、…%QX0.7 %QX1.0、…%QX1.7 …%QX374.7	X0000、…X0007 X0008、…X0015 …X2999	QXm.n：m*8+n	读写，X 选 0
	%QW	WORD	%QW0、%QW2、…%QW5998	X0000、 X0001、……X2999	QWm：m/2	读写，X 选 4
M 区	%MX	BOOL	%MX0.0、…%MX0.7 %MX1.0、…%MX1.7 …%MX7816.7	X3000、…X3007 X3008、…X3015 …X65535	MXm.n： m*8+n+3000	只读，X 选 1 读写，X 选 0
	%MW	WORD	%MW0、%MW2、…%MW125070	X3000、 X3001、…X65535	MWm：m/2+3000	只读，X 选 3 读写，X 选 4

以下分 4 种情况举例说明该映射关系：

（a） 读 I 区（或写 Q 区）开关量数据：如读%IX2.6（或写%QX2.6）开始的一段数据时，则需参考 I 区（或 Q 区）BOOL 型数据的映射公式，在主站中读（或写）指令的偏移地址应设为：2*8+6=22；

（b） 读 I 区（或写 Q 区）模拟量数据：如读%IW8（或写%QW8）开始的一段数据时，则需参考 I 区（或 Q 区）WORD 型数据的映射公式，在主站中读（或写）指令的偏移地址应设为：8/2=4；

（c） 读写 M 区开关量数据：如读写%MX2.6 开始的一段数据时，则需参考 M 区 BOOL 型数据的映射公式，在主站中读写指令的偏移地址应设为 2*8+6+3000=3022；

（d） 读写 M 区模拟量数据：如读写%MW1000 开始的一段数据时，则需参考 M 区 WORD 型数据的映射公式，在主站中读写指令的偏移地址应设为 1000/2+3000=3500。

如果需要修改从站 Modbus 起始地址，请注意地址使用区间不能重叠，否则，会编译报错，提示数据超限。

7.8 组态 Modbus 指令

7.8.1 指令与从站数据映射

添加指令时，根据设置的偏移量和数据长度，读取从站设备对应偏移地址开始的数据段，或向对应偏移开始的地址段写入数据。

通过选择不同的功能码，访问从站不同的数据类型。

功能码与从站数据单元对应关系

可选指令	数据区	数据类型	对应从站数据单元	从站映射地址
读线圈 (0xxxx, 01H)	0xxxx	BOOL	线圈	指令偏移量：0~3000，访问 Q 区数据，地址从%QX0.0 开始，按字节存储 指令偏移量：＞3000，访问 M 区数据，地址从%MX0.0 开始，按字节存储
写单个线圈 (0xxxx, 05H)				
写多个线圈 (0xxxx, 0FH)				
读离散量输入 (1xxxx, 02H)	1xxxx	BOOL	离散量输入	指令偏移量：0~3000，访问 I 区数据，地址从%IX0.0 开始，按字节存储 指令偏移量：＞3000，访问 M 区数据，地址从%MX0.0 开始，按字节存储
读输入寄存器 (3xxxx, 04H)	3xxxx	WORD	输入寄存器	指令偏移量：0~3000，访问 I 区数据，地址从%IW0 开始，按字存储 指令偏移量：＞3000，访问 M 区数据，地址从%MW0 开始，按字存储
读保持寄存器 (4xxxx, 03H)	4xxxx	WORD	保持寄存器	指令偏移量：0~3000，访问 Q 区数据，地址从%QW0 开始，按字存储 指令偏移量：＞3000，访问 M 区数据，地址从%MW0 开始，按字存储
读/写多个寄存器(4xxxx, 17H)				
写单个寄存器 (4xxxx, 06H)				

写多个寄存器 (4xxxx, 10H)				

7.8.2 添加指令

7.8.2.1 概述

控制器通过 Modbus TCP 或 Modbus RTU 通讯时，Modbus 主站读取从站数据，或向从站写入数据。需要在主站下创建 Modbus 通信的读写指令，该指令将映射为 Modbus 通讯数据。

读写指令在 ModbusSlave 从站上配置，最大可配置 32 个指令。

7.8.2.2 要求

ModbusSlave 从站窗口已打开

7.8.2.3 步骤

要添加从站指令，请按以下步骤操作：

（1） 在“指令”标签空白区域，单击右键。

（2） 选择“添加”命令。

弹出指令配置对话框。

（3） 在“可选指令”列表框中，选择指令。

（4） 配置指令属性。

参数	参数值	默认值	设置说明
触发方式	周期、上升沿、高电平	周期	周期：主站周期性轮询从站，收发指令数据。如果当前协议关联的任务周期大于触发周期，则实际触发周期为协议关联的任务周期 上升沿：选择上升沿触发时，需要指定一个触发变量。当该变量有上升沿跳变时，主站轮询从站，开始收发指令数据 高电平：选择高电平触发时，需要指定一个触发变量。当该变量为高电平时，主站轮询从站，开始收发指令数据
周期时间	0~65,535	100	主站轮询从站的周期时间，触发方式为“周期”时可设置
触发变量	选择工程中 BOOL 型变量，包括 Modbus 变量、POU 变量、通道变量、全局变量。不包含诊断变量、复杂变量和中间临时变量	--	触发方式为“上升沿”或“高电平”时可设置 您需要组态触发该变量的控制逻辑，其中，高电平、上升沿信号的保持时间必须大于任务实际执行周期，否则可能检测不到触发信号 任务实际执行周期（最大值）=2×从站响应超时时间+关联任务的运行周期，当从站响应超时时间为 0

				时，请使用主站配置的响应超时进行计算
读寄存器	偏移	0~65,535	0	读取从站数据的地址偏移量 读取从站设备 Modbus 起始地址+偏移开始的地址数据，Modbus 起始地址在从站设备中配置
	长度	读线圈/读离散量输入：1~2000 读保持寄存器/读输入寄存器：1~125 读/写多个寄存器：1~118	1	读取的数据长度 从 Modbus 起始地址+偏移地址开始，读取设置长度的数据
	错误处理	保持、清零	保持	保持：在应答异常后保持当前数据 清零：在应答异常后将当前数据清零
写寄存器	偏移	0~65,535	0	写入从站数据的地址偏移量 向从站设备 Modbus 起始地址+偏移开始的地址中写入数据，Modbus 起始地址在从站设备中配置
	长度	写单个线圈：默认为 1 写单个寄存器：默认为 1 写多个线圈：1~1968 写多个寄存器：1~123 读/写多个寄存器：1~118	1	写入的数据长度 从 Modbus 起始地址+偏移地址开始，写入设置长度的数据

（5） 单击**确定**按钮。

“从站 I/O 映射”标签页中将创建对应的通道数据。

7.8.2.4 参考信息

指令参数

I/O 映射

功能码

7.8.3 I/O 映射

配置指令后，在【从站 I/O 映射】标签页中会映射对应的 Modbus 数据。

- MODBUS 起始地址=X0001+偏移地址，X 为由寄存器类型决定；
- 通道名称：初始值为 Sn_CH_通道号，n 为从站添加序号；
- 通道地址：通过组态直接使用通道地址，就可以访问 Modbus 从站数据。

7.8.4 ModbusTCP 指令诊断码

双击 ModbusSlave_TCP 树节点，打开信息参数页面，在【诊断信息】标签中查看指令诊断状态。当添加 Modbus 指令后，对应的诊断变量自动生成。


ModbusSlave_TCP_7

设备参数 | 指令 | 从站I/O映射 | 信息 | 诊断信息

	变量名	直接地址	变量类型	初始值	掉电保护	变量说明
1	AT_ModbusTCPOrder7_DiagState_1	%SB2024	BYTE	0	FALSE	第1条指令的状态诊断
2	AT_ModbusTCPOrder7_DiagState_2	%SB2025	BYTE	0	FALSE	第2条指令的状态诊断
3	AT_ModbusTCPOrder7_DiagState_3	%SB2026	BYTE	0	FALSE	第3条指令的状态诊断
4	AT_ModbusTCPOrder7_DiagState_4	%SB2027	BYTE	0	FALSE	第4条指令的状态诊断

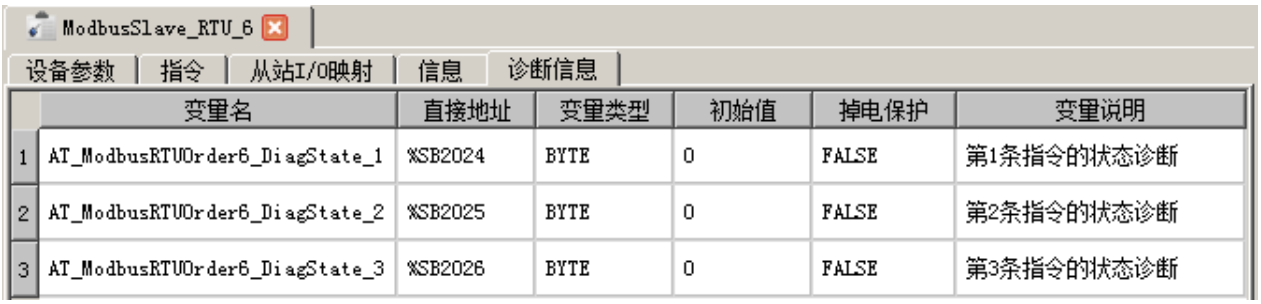
ModbusTCP 诊断信息

指令状态码值列表

指令状态码值	含义
0	无错误
2	超时
3	指令组态错误
4	功能码错误
64	发送的数据与接收的数据不匹配
16	单元 ID 不匹配（从站地址错误）
32	TCP 连接失败
33	发送请求报文失败
34	接收确认报文失败
128+1	从站不支持的功能码
128+2	数据地址溢出
128+3	数据范围溢出
128+4	从站设备故障
128+6	从站设备忙
128+15	从站其它故障

7.8.5 ModbusRTU 指令诊断码

双击 ModbusSlave_RTU 树节点，打开信息参数页面，在【诊断信息】标签中查看指令诊断状态。当添加 Modbus 指令后，对应的诊断变量自动生成。


ModbusSlave_RTU_6

设备参数 | 指令 | 从站I/O映射 | 信息 | 诊断信息

	变量名	直接地址	变量类型	初始值	掉电保护	变量说明
1	AT_ModbusRTUOrder6_DiagState_1	%SB2024	BYTE	0	FALSE	第1条指令的状态诊断
2	AT_ModbusRTUOrder6_DiagState_2	%SB2025	BYTE	0	FALSE	第2条指令的状态诊断
3	AT_ModbusRTUOrder6_DiagState_3	%SB2026	BYTE	0	FALSE	第3条指令的状态诊断

ModbusRTU 诊断信息

指令状态码值列表

指令状态码值	含义
0	无错误
1	校验码异常
2	超时
3	指令组态错误
4	校验码正常，功能码不正常
8	校验码正常，内容不正确
16	校验码正常，从站地址不正确
32	校验码正常，指令返回异常码

7.9 关联任务

7.9.1 概述

需要为控制器上运行的总线通讯协议指定一个协议任务，每个协议可指定不同的任务。当任务运行时，执行相关协议。

7.9.2 要求

- 已添加通信协议
- 协议窗口已打开

7.9.3 步骤

要关联任务，请按以下步骤操作：

（1） 在“设备信息”标签页中，选择协议关联的任务名。

7.10 OPCUA_Server 协议

7.10.1 LK246C OPCUA 配置

OPCUA 是一种工业通讯协议，包含了之前的 A&E, DA,OPC XML DA or HDA，只使用一个地址空间就能访问之前所有的对象，易于配置和使用，在协议和应用层集成了安全功能，更加安全。支持 LK246C 下的 OPCUA_Server 协议配置，主要分为 OPCUA_Server 配置、OPCUA 实时库 XML 生成、XML 转换二进制、OPCUA 二进制下装过程。

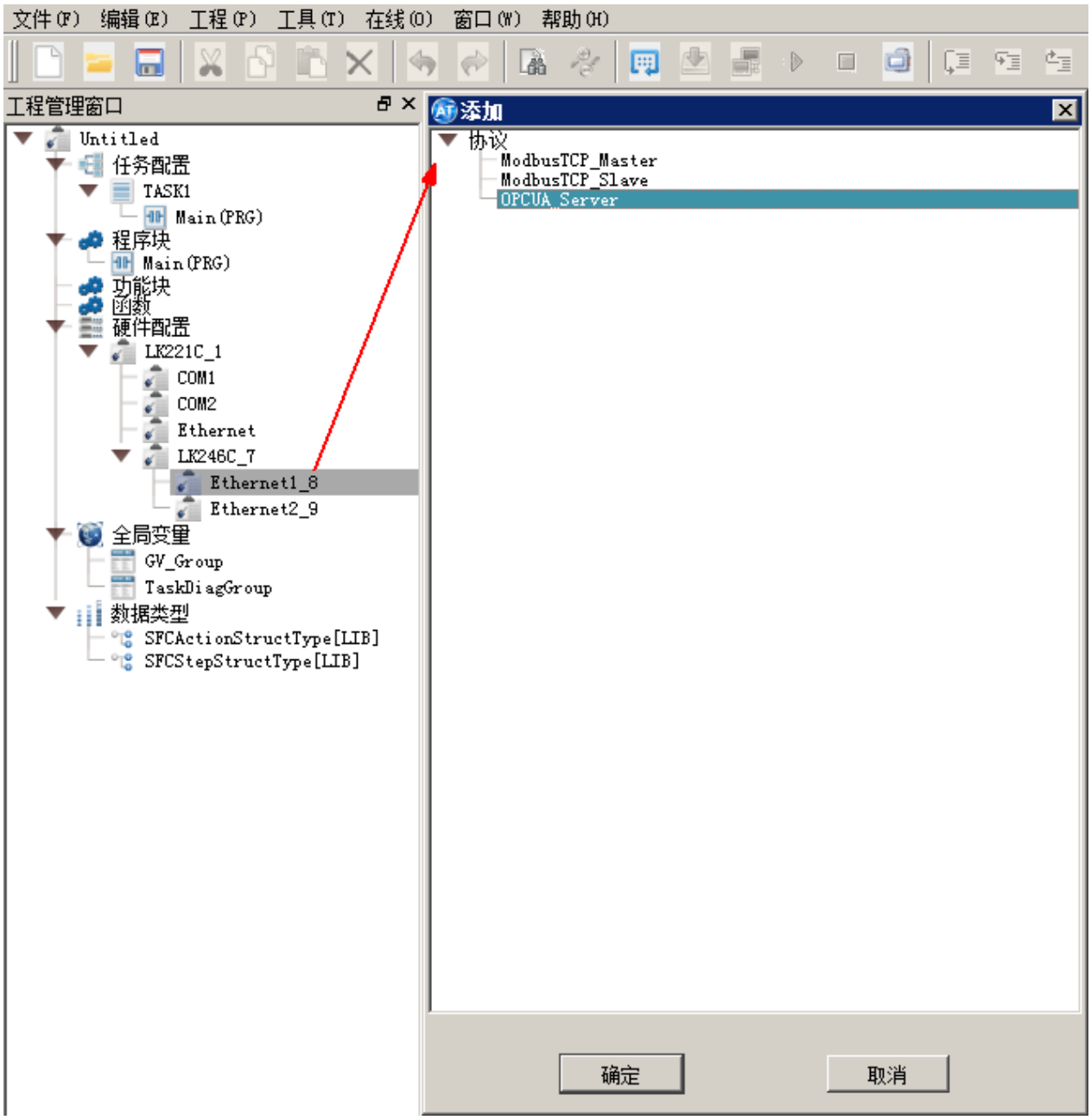
7.10.1.1 协议限制

LK246C 模块集成 OPCUA 协议功能。

- 整个工程只支持配置一个 OPCUA_Server 协议，仅支持一个 LK246C 模块下的一路网口可配置。LK246C 单板最多支持 16 路协议（Modbus 主+Modbus 从+OPCUA_Server）。
- OPCUA 变量可配置数目受制于所有变量占用字节大小，所有配置 OPCUA 变量最多占用 20000 字节。

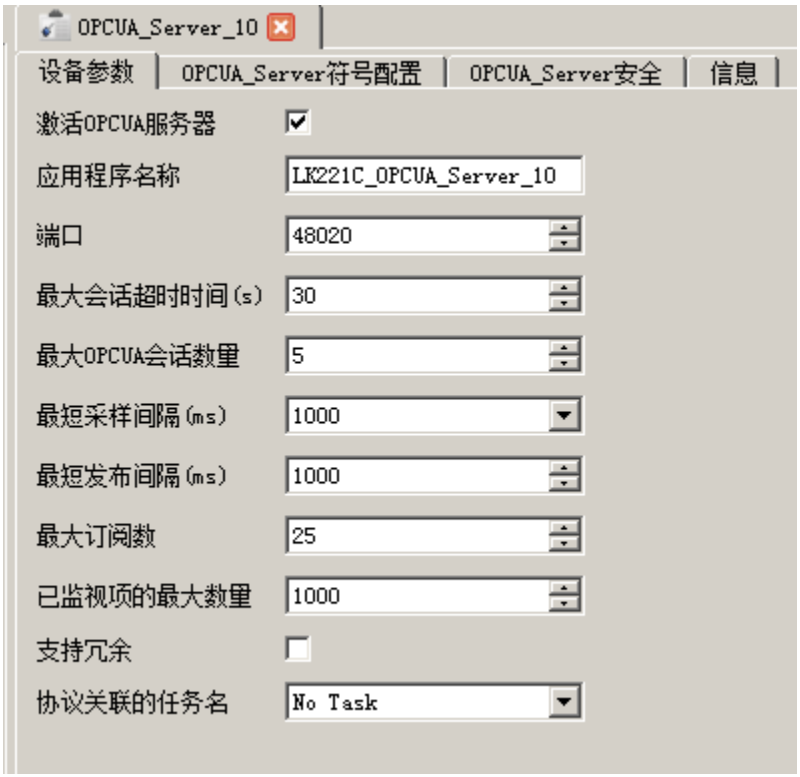
7.10.1.2 OPCUA 配置

（a） 在 LK246C 下的 Ethernet1、Ethernet2 节点下，右键“添加”，追加“OPCUA_Server”协议；



添加 OPCUA_Server 协议

（b） “OPCUA_Server”协议视图可通过双击“OPCUA_Server”工程管理树节点、右键“打开”两种方式打开视图。“激活 OPCUA 服务器”默认勾选。



设备参数

参数对应默认值及范围：

- 端口号（默认值 48020，值范围[1024..49151]）
- 最大会话超时时间 s（默认值 30，值范围[1..600000]）
- 最大 OPCUA 会话数量（默认值 5，值范围[1..10]）
- 最短采样间隔 ms（默认值 1000，值范围{50, 100, 250, 500, 1000, 2500, 5000，10000}）
- 最短发布间隔 ms（默认值 1000，值范围[500..3600000]）
- 最大订阅数（默认值 25，值范围[1..50]）
- 已监视项的最大数量（默认值 1000，值范围[100..13500]）

7.10.1.3 OPCUA 变量组态

LK246C 新增支持 OPCUA_Server 协议，仅支持单元素的数据类型: BOOL、BYTE、DINT、DWORD、INT、LREAL、REAL、SINT、UDINT、UINT、USINT、WORD、ENUM。

复杂数据类型 STRING、FB、Struct、Array、Pointer、TIME 及其子成员不支持 OPCUA 协议配置。

OPCUA 读写权限分为：“只读”权限、“读写”权限。

S 区默认为“只读”权限，不可设置为“读写”权限。

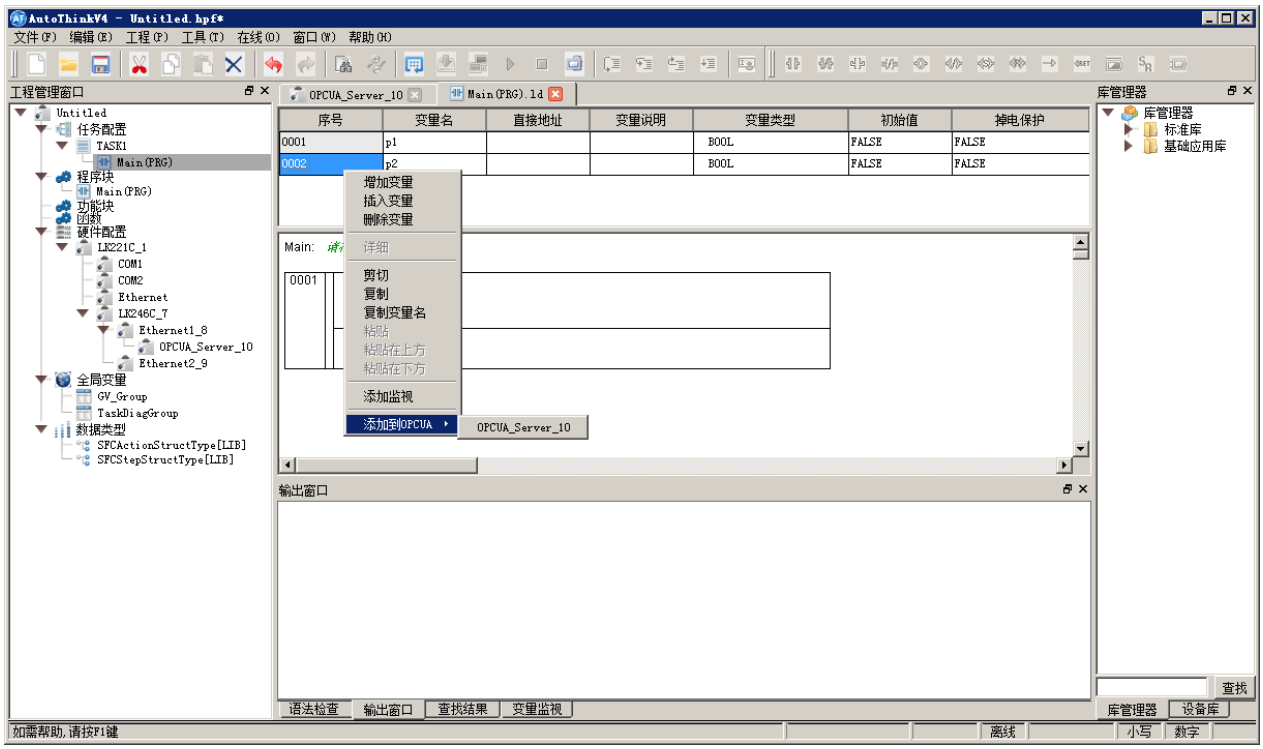
S 区以外的其他数据区变量默认为“读写权限”。

通过以下操作,将变量添加到“OPCUA_Server 符号配置”标签页。

（1） 在 POU、全局变量组或 I/O 模块通道标签页中，右击一个变量；

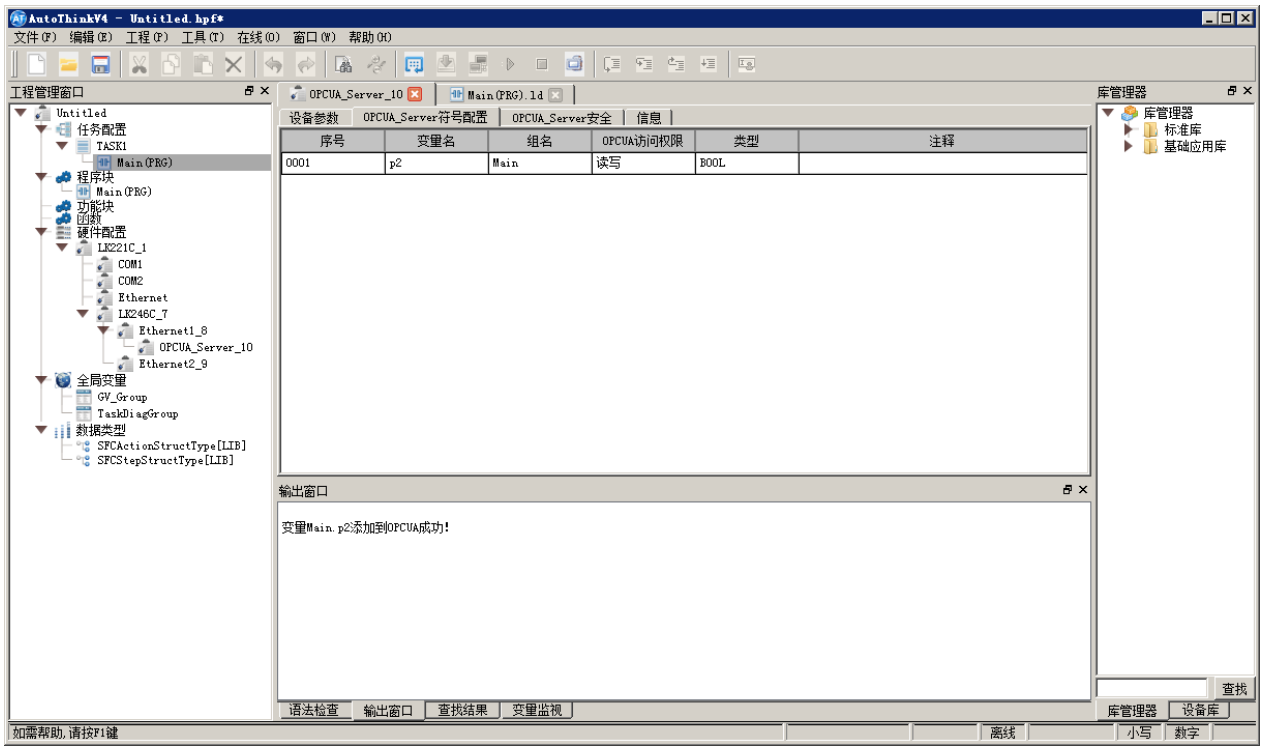
（2） 在右键菜单中，单击【添加到 OPCUA】-【OPCUA_Server_IO】命令；

输出窗口中显示变量添加到 OPCUA 成功。



添加 OPCUA 变量

（3） 双击 LK246C 下“OPCUA_Server_IO”树节点，打开设备参数界面，单击“OPCUA_Server”标签页，即可看到已添加的 OPCUA 变量。



“OPCUA_Server”标签页

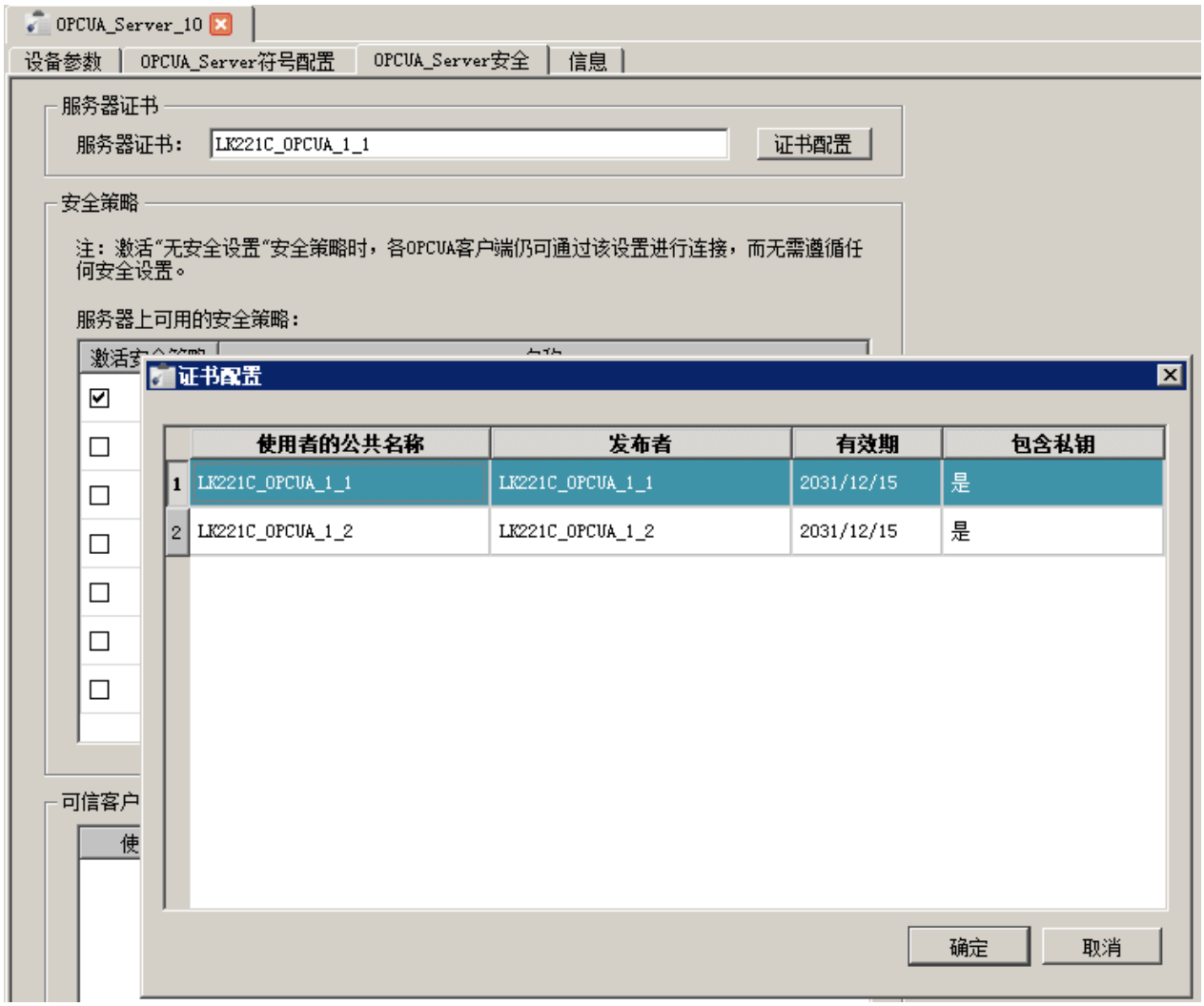
7.10.1.4 OPCUA 安全

用户可以通过赋予 OPCUA_Server 的匿名访问方式、用户名密码方式、证书等三种验证权限，可以单独配置，也可同时配置。

用户在客户端选择以上一种方式进行身份验证登录，进行数据访问、交互。

1. 服务器证书

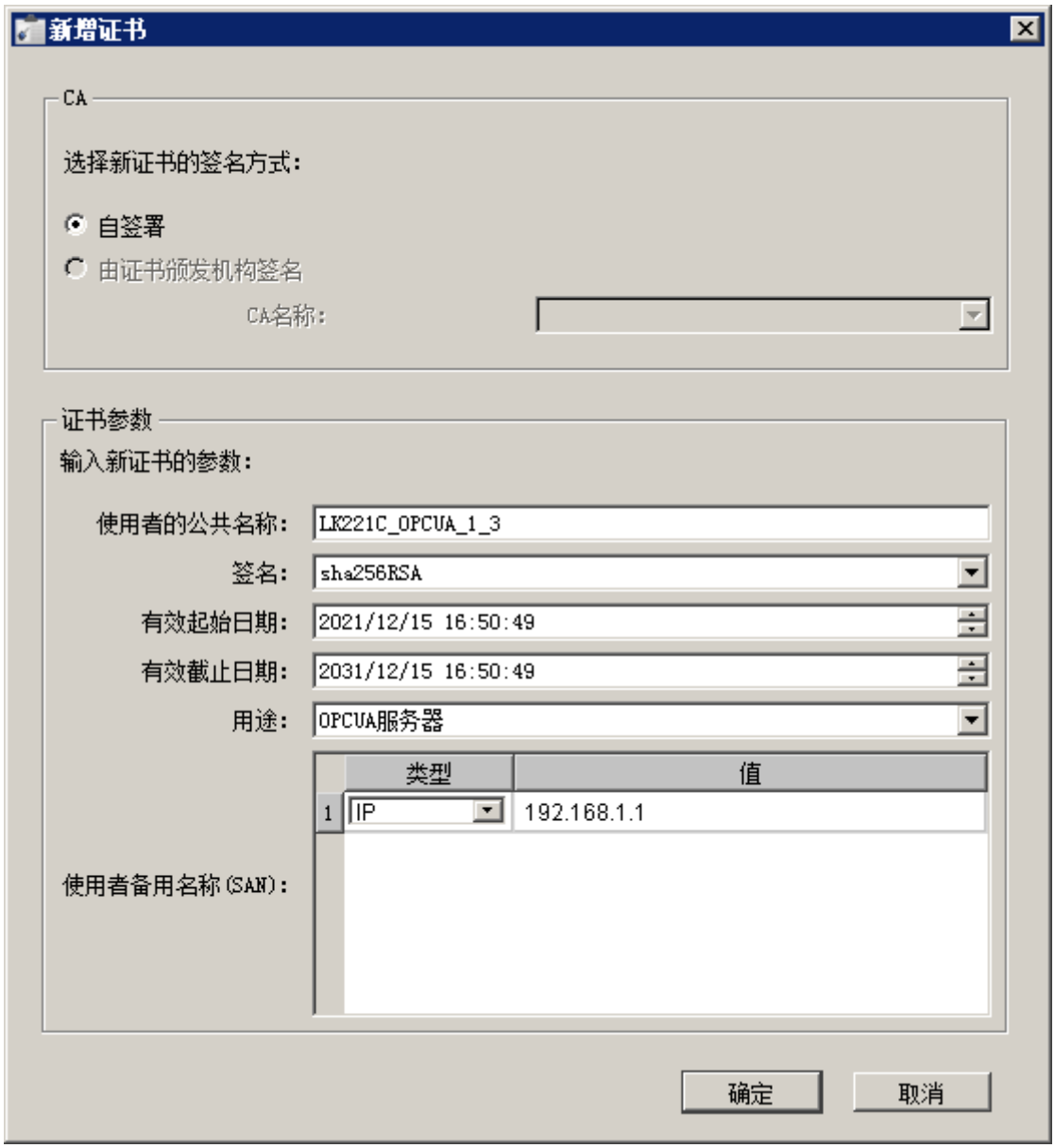
（4） 服务器证书配置及管理



OPCUA_Server 证书配置(a)

- 证书配置提供选择证书功能。
- 证书配置提供新增证书功能。
- 证书配置提供删除证书功能。
- 证书配置提供证书配置功能。

创建自签署 CA 证书（暂不支持证书颁发机构签名）



OPCUA_Server 证书配置(b)

- 使用者的公共名称即证书名。
- 签名：支持 sha1RSA、sha256RSA 两种方式。
- 有效起始日期：证书自指定日期起生效。
- 有效截止日期：指定日期到期后，证书将失效。
- 用途：现仅支持“OPCUA 服务器”证书用途。
- 使用者备用名称。

（5） 服务器证书生成并下装

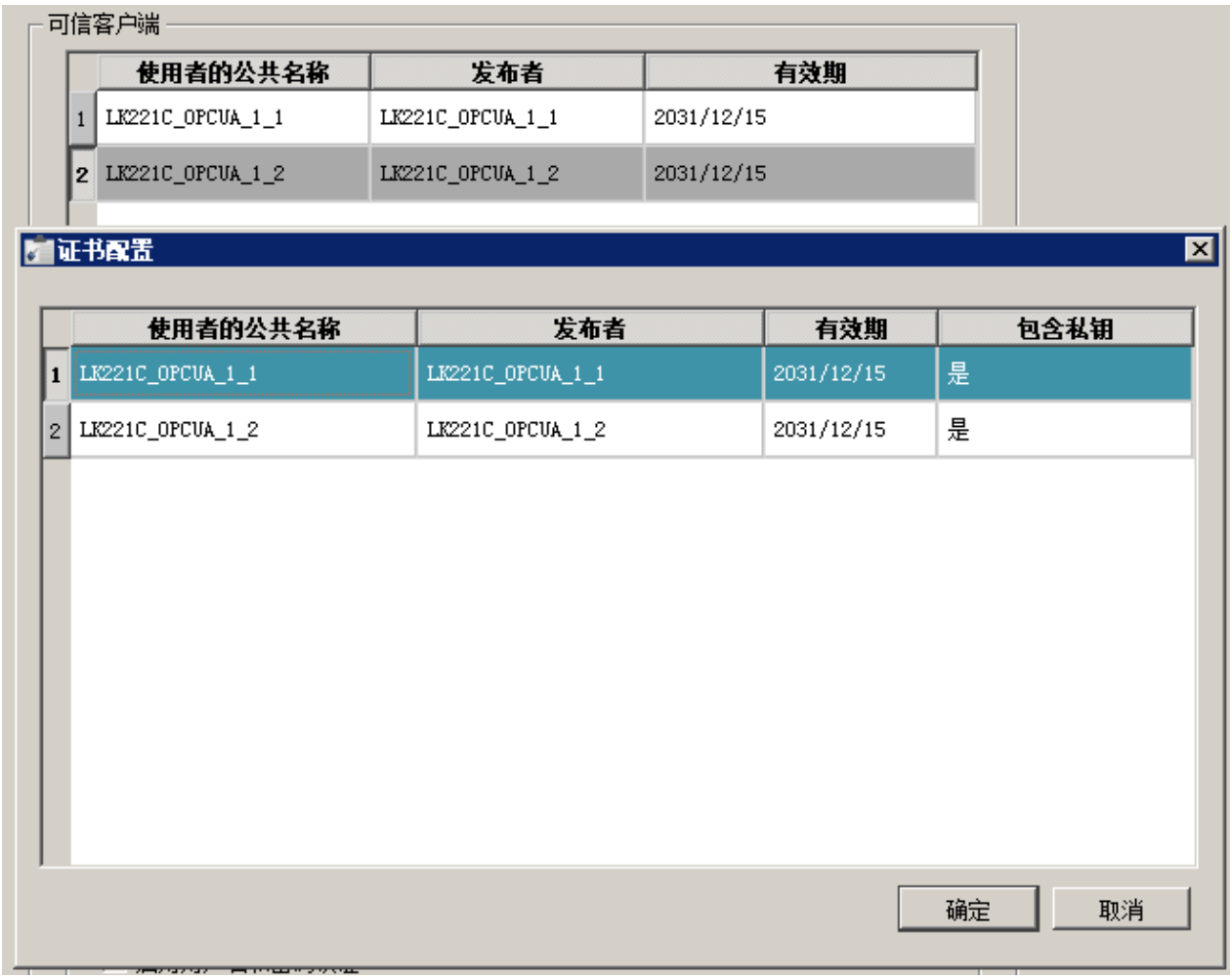
将与协议栈一致的开放源代码集成，并生成证书下发。

（6） 设备参数属性页新增应用程序名称

此应用程序名称默认是根据主控的型号和 OPCUA 模块的序号生成的，用户可以手动修改，长度为 32 个字符，可以包含大小写字母、数字和下划线，但必须以字母或下划线开头，此参数会随着硬件配置下装到给 OPCUA 服务器。生成服务器证书的可选名称的 URI 里包含此参数。

2. 可信客户端

提供可信客户端证书导入功能，用户选择对应客户端证书导入后，将该证书下发至 Server，可配置最多 64 个可信客户端。支持证书导出和导入功能，支持不含私钥的 Der 格式和包含私钥的 P12 格式的证书。



OPCUA_Server 可信客户端配置

3. 安全策略

因 Basic128Rsa15 和 Basic256 在嵌入式版本的 OPCUA_ Server 协议栈上不推荐该安全策略，故现支持如下所述安全策略。每种安全策略有“签名”、“签名和加密”两种方式。

安全策略表

Security Policy	URI

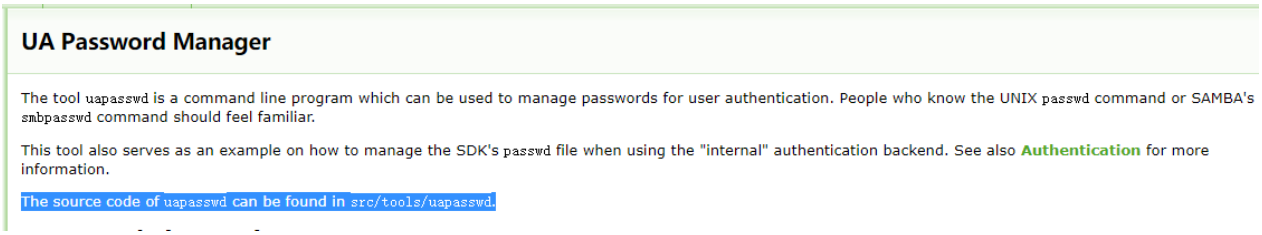
Aes256_Sha256_RsaPss	http://opcfoundation.org/UA/SecurityPolicy#Aes256_Sha256_RsaPss
Aes128_Sha256_RsaOaep	http://opcfoundation.org/UA/SecurityPolicy#Aes128_Sha256_RsaOaep
Basic256Sha256	http://opcfoundation.org/UA/SecurityPolicy#Basic256Sha256
None	http://opcfoundation.org/UA/SecurityPolicy#None

每种安全策略均可独立配置。上位机仅配置下发安全策略。

4. 用户身份认证

用户可通过“访客认证”和“用户密码认证”方式进行身份认证。两种认证方式可并行，由 UA 客户端选择认证方式。

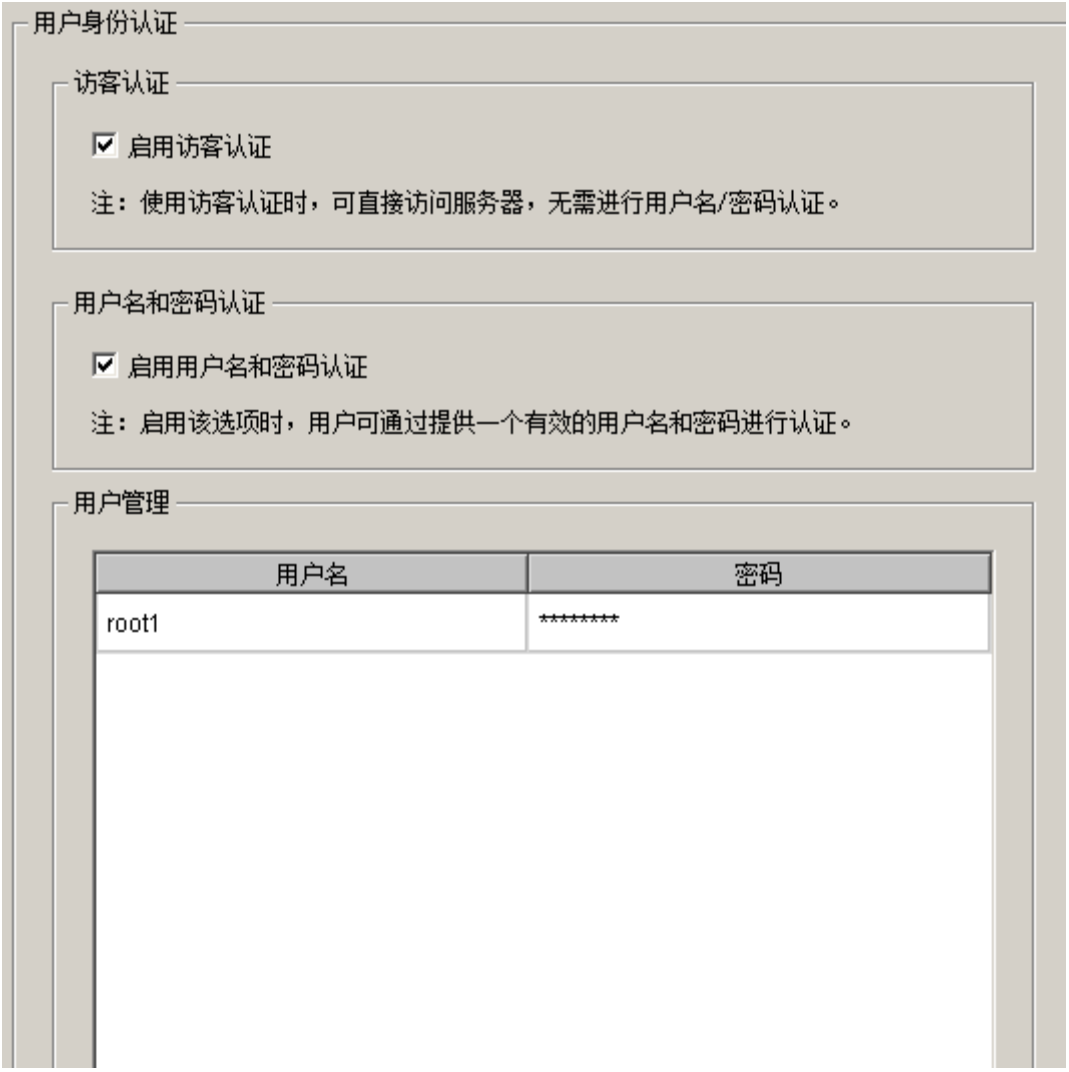
（1）访客认证。用户无需证明身份（匿名访问）。OPCUA 服务器不检查客户端用户权限。



访客认证

（2）用户密码认证（非匿名访问）。启用后需配置用户管理。OPCUA 服务器会检查客户端用户是否具备访问服务器的权限，通过用户名和密码进行身份验证，支持用户数最多 20 个。

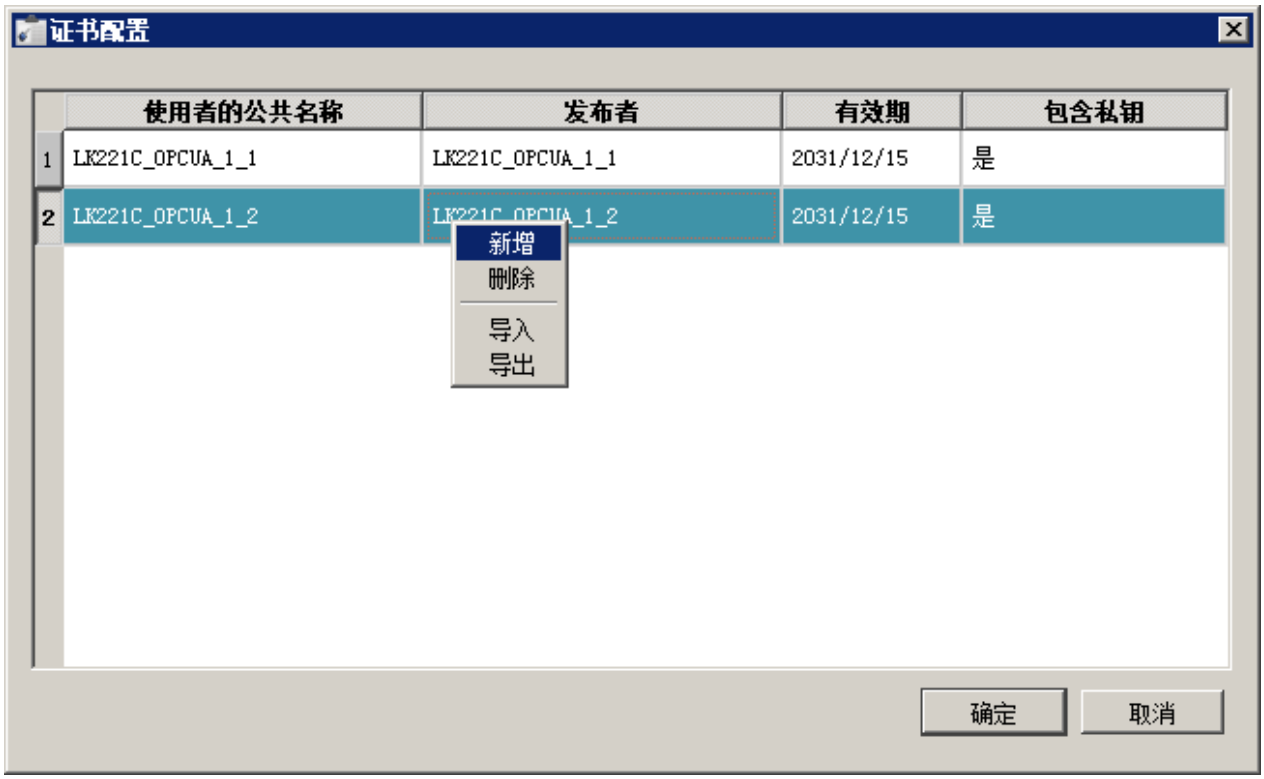
用户名要求为合法的标识符，由大小写字母和数字下划线组成，最长为 32 个字符，密码要求长度要求 6~32 个字符，必须包含字母、数字和特殊字符，必须包含大小写字符。



用户密码认证

5. 证书管理器

证书管理器提供证书管理功能，为 AT 所有需要证书的功能提供支持，此证书管理器是复用的，提供配置服务器证书和可信客户端证书功能。



证书配置

- 新增证书，生成自签署证书，用户可通过证书管理器新增生成自签署证书，证书管理器生成的证书会自动加载到证书列表中。
- 删除证书，用户可通过右键或 Delete 键删除已经加载到证书列表中的证书。
- 导入证书，支持导入第三方的或 AT 导出的不带私钥的.Der 格式和带私钥的 P12 格式的证书，证书导入成功后，自动加载到证书列表中。
- 导出证书，支持导出不带私钥的.Der 格式和带私钥的 P12 格式的证书，证书导出成功后，可提供给第三方软件使用。

7.10.1.5 OPCUA XML 导出

工程编译完成后，才可导出 OPCUA XML 文件，将生成的 XML 导出到用户指定的文件夹下。该导出功能仅为用户查看，将当前工程的 OPCUA 符号点表以 OPCUA XML 的格式导出。

7.11 修改通道地址

7.11.1 概述

系统为添加的 IO 模块和 Modbus 指令自动分配通道地址。通道地址以设置的输入/输出起始字节偏移作为起始地址依次分配。在添加模块时，系统自动分配一个起始字节偏移地址，默认从 0 开始，即%IB0 和%QB0。

如果需要修改通道地址，则修改输入模块或输出模块的起始字节偏移即可。

修改起始字节偏移地址需满足以下要求：

- 偏移地址满足四字节对齐。
- 以新的偏移地址计算，通道长度内的所有地址均未被使用。

7.11.2 要求

- 模块或 Modbus 指令已添加

7.11.3 步骤

要修改通道地址，请按以下步骤操作：

（1） 在“硬件配置”节点下，双击模块或 ModbusSlave_TCP 设备节点，打开设备属性窗口。

设置 IO 模块起始字节偏移

设置 Modbus 指令起始字节偏移

（2） 双击输入或输出起始字节偏移编辑框，输入偏移值。

在【通道】标签页或【从站 I/O 映射】标签页中可以查看以新偏移地址开始的通道地址。

第8章　组态工程逻辑

8.1 程序组织单元

8.1.1 POU 类型

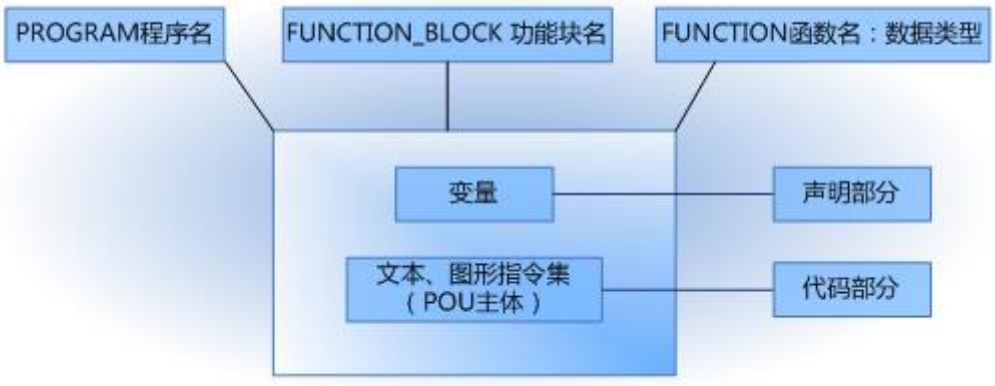
POU 有三种类型：程序、功能块、函数。

每个 POU 都由变量声明和逻辑程序两部分组成。

声明部分：局部变量声明在 POU 编辑器中的变量区进行变量的定义。

算法部分：即 POU 主体，用于实现逻辑运算。

POU 结构示意图如图所示。



POU 的结构图

- 程序（Program）

  程序是为了完成某项任务而编写的控制逻辑，或一组指令的集合。

- 功能块（Function Block）

  功能块可产生一个或多个数值可执行的逻辑元件。功能块可保留值于下一运算，输入一组数据后，可能得到不同的输出数据。

  功能块必须被程序调用后，才能运行。

- 函数（Function）

  函数用于实现数学运算，函数没有存储区保留参数值，需要在程序中定义对应的实参。在程序中，调用函数时，将执行函数中的数学运算，并将函数值返回给调用块。

  定义函数时，需要指定一个数据类型作为返回值（返回数据类型）。函数的计算结果赋给函数本身，即函数的输出变量就是函数名本身。

可以在程序的多个位置多次调用同一个函数。

在下面的描述中，程序类 POU 简称为程序，功能块类 POU 简称为功能块，函数类 POU 简称为函数。

8.1.2 创建 POU

8.1.2.1 概述

用户程序通过 POU 实现，系统默认包含程序 Main（PRG），可根据用户逻辑，创建不同的 POU。

程序、函数、功能块创建方法相同，唯一的区别是函数需要定义返回值类型。

8.1.2.2 要求

工程已打开

8.1.2.3 步骤

要创建 POU，请按以下步骤操作：

（1） 在程序块、功能块、函数树节点右键菜单中，选择“新增 POU”命令。

将弹出“新增 POU”对话框，POU 类型根据选择的节点自动默认。

（2） 输入 POU 名称。

如果名字不合法，**确定**按钮不可用。

命名遵循如下原则：

- POU 名只能包含字母、数字、下划线“_”，第一个字符必须是字母或者下划线。
- 不能与变量名、变量组名、POU 文件夹名、任务名、工程名、数据类型（自定义或系统缺省的）、关键字、指令库名或功能块名重名。
- POU 名不可为空。
- POU 名称长度最多不超过 32 字节，超出范围的部分无法输入。
- POU 名称不能定义为 Windows 预留的关键字，关键字包括：CON、PRN、AUX、NUL、COM0、COM1、COM2、COM3、COM4、COM5、COM6、COM7、COM8、COM9、LPT0、LPT1、LPT2、LPT3、LPT4、LPT5、LPT6、LPT7、LPT8、LPT9。

（3） 选择编程语言。

函数请继续执行（4）~（5）步，功能块和程序块请跳到第（6）步执行。

（4） 单击**返回类型**按钮。

将弹出返回类型选择对话框。

（5） 选择数据类型，单击**确定**按钮。

（6） 单击**确定**按钮。

新的 POU 被创建到对应树节点下。

8.1.3 复制和粘贴 POU

8.1.3.1 概述

复制的程序、功能块和函数可以粘贴到对应树节点下。

8.1.3.2 要求

有一个可用的 POU

8.1.3.3 步骤

要复制、粘贴 POU，请按以下步骤操作：

（1） 在 POU 右键菜单中，选择“复制 POU”命令。

（2） 在 POU 父节点的右键菜单中，选择“粘贴 POU”命令。

8.1.4 删除 POU

8.1.4.1 规则

删除 POU 时，请先了解以下规则：

- 删除程序，则调用任务下同步删除
- 删除功能块或函数，则调用 POU 中保留该功能块实例或函数，编译会报错

8.1.4.2 要求

有需要删除的 POU

8.1.4.3 步骤

要删除 POU，请按以下步骤操作：

（1） 在 POU 右键菜单中，选择“删除 POU”命令

将弹出确认对话框。

（2） 单击**是**按钮

POU 被删除。

8.1.5 重命名 POU

8.1.5.1 概述

重命名 POU 时，将同步更新该 POU 调用处。需要注意的是，ST 中调用的 POU 不支持更新。

8.1.5.2 要求

有一个可用的 POU

8.1.5.3 步骤

要重命名 POU，请按以下步骤操作：

（1） 在 POU 右键菜单中，选择“重命名”命令。

将弹出“重命名”对话框。

（2） 输入新名称。

命名规则请参见创建 POU。

（3） 单击**确定**按钮。

该 POU 名被更新。

8.1.6 导入 POU

8.1.6.1 概述

工程组态时，如果需要复用其他工程的 POU 程序，则将 POU 导入到当前工程，打开编辑即可。

导入时，如果当前工程中存在同名 POU，则提示进行覆盖。

导入 POU 时需注意以下几点，否则会导入失败。

- 工程最大支持 512 个 POU，超过该值后的 POU 无法导入。
- 一个 POU 最大网络节点 999 个，超过该值后的网络节点无法导入。
- 一次只支持导入 100 个 POU。
- 不支持导入网络节点上串联元素超过 32 个或者并联元素超过 16 个的 POU。
- 导入的 POU 名不能与算法库中指令名、工程管理树中文件夹名、系统自动生成的 POU 重名。
- 仅支持导入 ST 和 LD 语言的 POU。

8.1.6.2 要求

有至少一个可导入 POU

8.1.6.3 步骤

要导入 POU，请按以下步骤操作：

（1） 在“工程”菜单栏中，选择“导入 POU”命令。

将弹出文件选择窗口。

（2） 选择.xml 文件。

（3） 单击**打开**按钮

POU 被导入，“语法检查”窗口显示导入结果。

8.1.7 导出 POU

8.1.7.1 概述

在工程组态时，如果需要复用另一个工程的 POU，可以将 POU 从其他工程中导出，再将其导入到当前工程即可。

文件以.xml 格式被导出。

导出时需要注意，仅支持导出 ST 和 LD 语言的 POU。

8.1.7.2 要求

已创建至少一个 POU。

8.1.7.3 步骤

要导出 POU，请按以下步骤操作：

（1） 在“工程”菜单栏中，选择“导出 POU”命令。

将弹出“导出 POU”对话框。

（2） 勾选需要导出的 POU 名。

（3） 单击**浏览**，选择文件保存路径。

（4） 单击**导出**按钮。

保存成功后，“语法检查”窗口显示导出成功。

8.1.8 权限设置

8.1.8.1 概述

用户可以对 POU 设置读写权限。设置读写权限的 POU 需要密码验证后，才能进行编辑。

设置成功后，在下次打开该 POU 时，会提示用户输入读写密码。如果连续 3 次密码验证错误，则打开 POU 失败。

8.1.8.2 要求

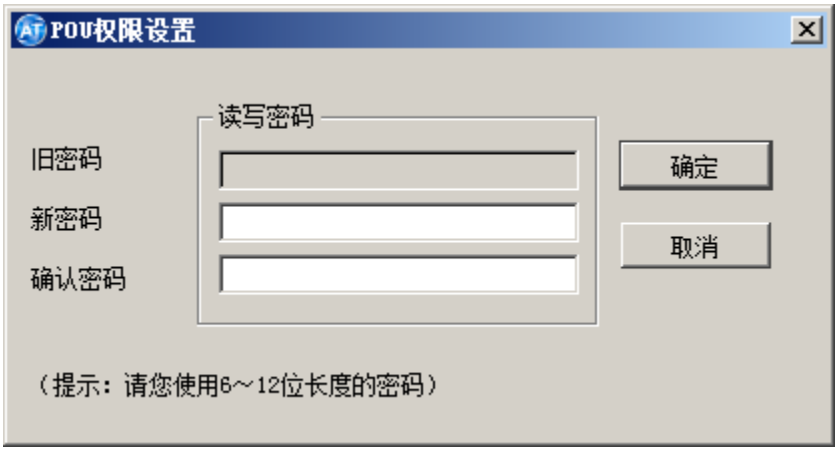
已创建 POU

8.1.8.3 步骤

要设置 POU 读写权限，请按以下步骤操作：

（1） 工程管理树上，右击 POU 名，单击【权限设置】。

弹出“POU 权限设置”对话框，如下图所示。



POU 权限设置对话框

（2） 输入新密码，并二次输入密码进行确认。

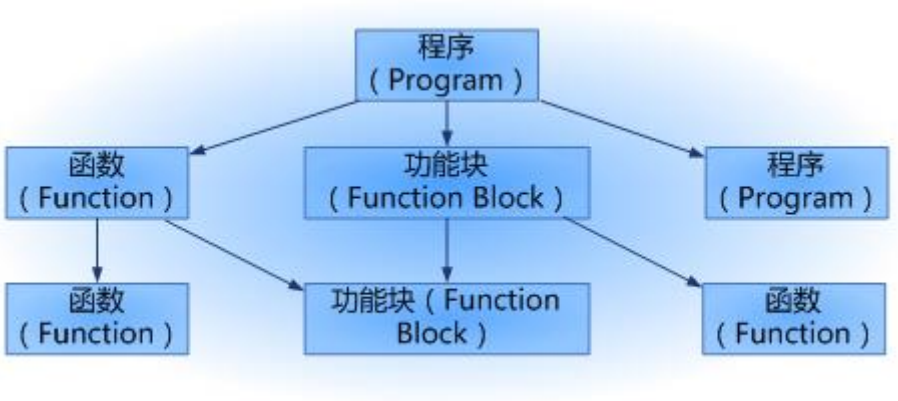
新密码和**确认新密码**框中输入字符完全相同，密码长度为 6~12 个字符，且区分大小写。

8.1.9 调用 POU

8.1.9.1 调用原则

POU 的调用要遵循以下原则，如下图所示。

- 程序可以调用函数、功能块和其他程序。
- 功能块可以调用函数和其他功能块。
- 函数可以调用功能块或函数。



POU 的调用关系

8.1.9.2 调用功能块

1. 概述

功能块包含系统自带功能块和自定义功能块两类，都需要在程序中调用，实现用户控制功能。

调用功能块时，首先需要为功能块声明一个变量实例。在程序中，调用该变量实例名，运行该功能块。

2. 要求

有一个调用 POU

3. LD 中调用

LD 编辑器调用功能块，请按以下步骤操作：

（1） 定义一个变量，变量类型选择自定义功能块或系统功能块。

（2） 在程序区，右键添加块元件。

（3） 将块元件名称修改为功能块名称。

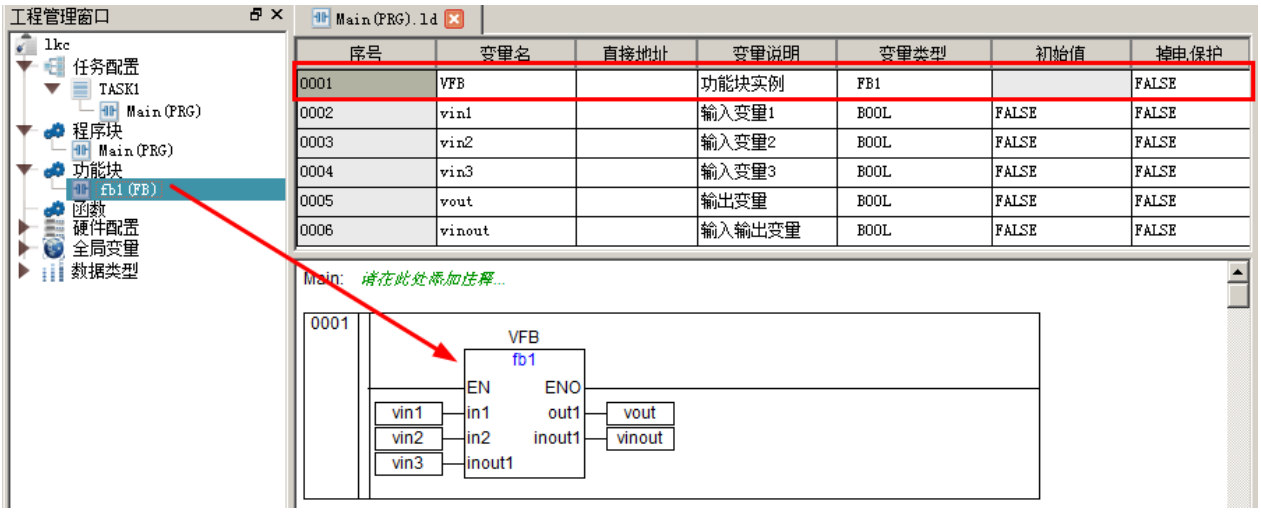
块元件显示为调用功能块，功能块上方显示变量名占位符“???”。

（4） 选中占位符，输入功能块变量实例名。

也可按 F2 选择变量实例。

（5） 功能块引脚关联变量名。

示例：在 Main 程序中调用功能块 fb1，需要声明功能块实例 VFB，该实例参与逻辑控制。



LD 功能块调用

4. ST 中调用

ST 编辑器调用功能块，请按以下步骤操作：

（1） 定义一个变量，变量类型选择自定义功能块或系统功能块。

（2） 在语句行输入“功能块实例名.变量名;”

也可按 F2 选择变量。

（3） 对功能块输入和输出变量赋实际变量。

示例：在 ff 程序中调用功能块 fb1，需要声明功能块实例 VFB，该实例参与逻辑控制。

ST 功能块调用

8.1.9.3 调用函数

1. 概述

函数包含系统自带函数块和自定义函数，在调用时，需要在程序中定义实参，实参个数和数据类型与形参保持一致。

2. 要求

有一个调用 POU

3. LD 中调用

LD 编辑器调用函数，请按以下步骤操作：

（1） 定义变量，作为函数实参。

（2） 在程序区，右键添加块元件。

（3） 选中块元件名称，修改为函数名称。

也可按 F2 选择变量实例。

（4） 为函数形参关联实参变量。

4. ST 中调用

ST 编辑器调用函数，请按以下步骤操作：

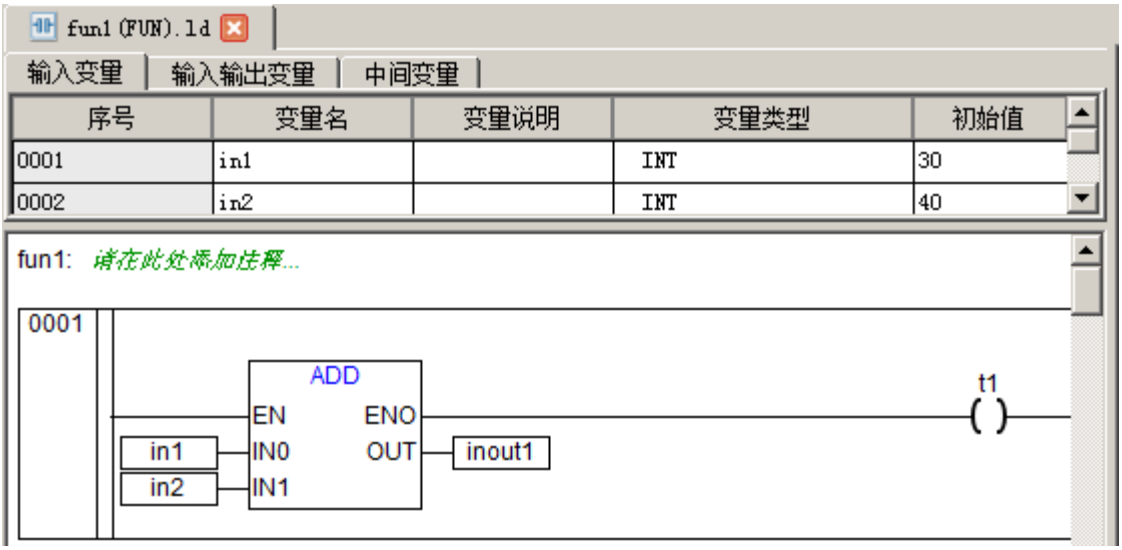
（1） 定义变量，作为函数实参。

（2） ST 语句行输入“函数名(实参 1，实参 2，...);”。

5. 示例

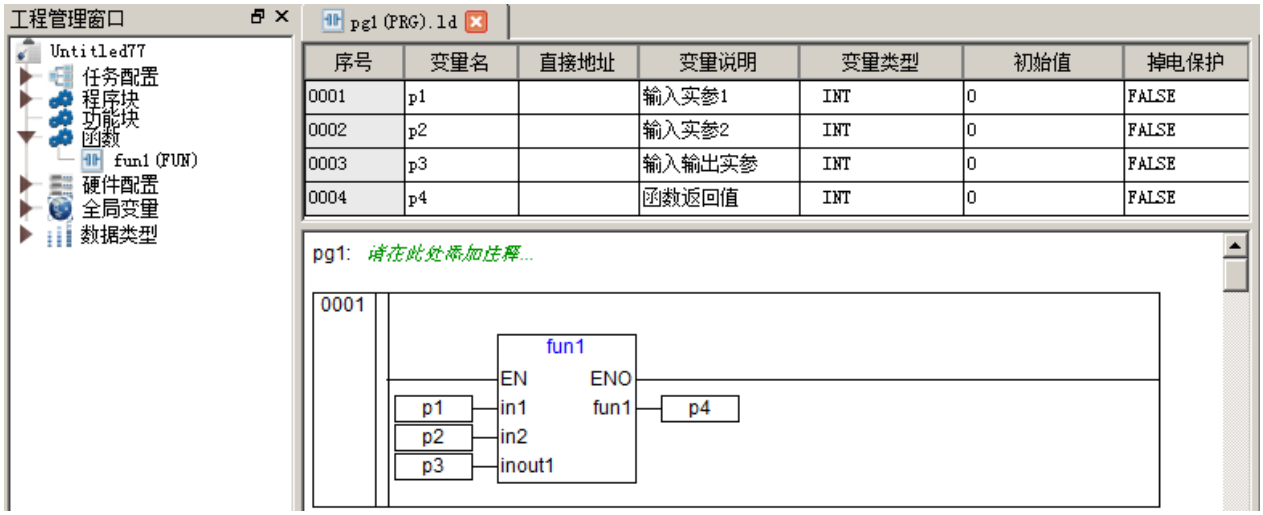
函数 fun1 实现加运算，形参列表如下：

变量类型	变量名	变量类型
输入变量	in1	INT
	in2	INT
输入输出变量	inout1	INT

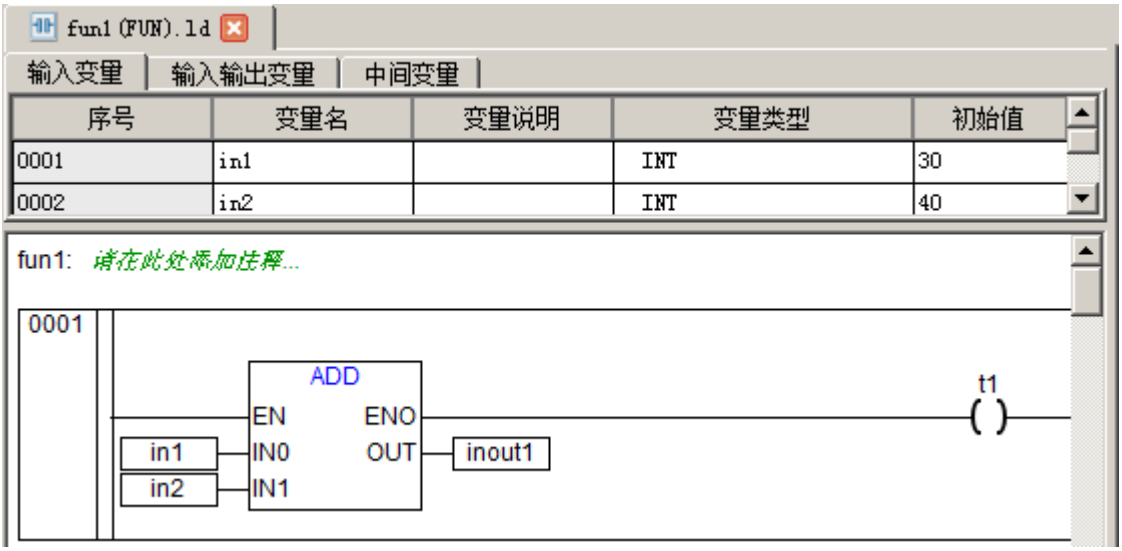


fun1 函数

定义实参 p1、p2、p3 和函数返回值变量 p4，将函数返回值赋给 p4。



LD 函数调用



ST 函数调用

8.1.9.4 调用程序

1. 要求

有两个可用的 POU

2. LD 中调用

LD 中程序调用程序，请按以下步骤操作：

（1） 在程序区，右键添加块元件。

（2） 将块元件名称修改为程序 POU 名称。

3. ST 中调用

ST 中程序调用程序，请按以下步骤操作：

（1） 在语句行输入“程序名（）;”

8.2 组态任务

8.2.1 创建一个任务

8.2.1.1 步骤

要新建一个任务，请按以下步骤操作：

（1） 在工程管理树，右击“任务配置”节点。

（2） 选择“新建任务”命令。

将弹出“新建任务”对话框。

（3） 按下面参数表设置任务属性。

任务属性设置

任务属性		设置说明
任务名称		名字由数字、字母、下划线组成，不能以数字开头。长度不超过 32 个字符
任务类型		周期运行：选择周期运行，控制器根据设置的周期时间和优先级，定时执行任务 状态触发：当关联的 BOOL 型变量值为 TRUE 时，任务开始执行，运行条件满足后等价于全速运行类任务 事件触发：当关联的 BOOL 型变量为上升沿时，任务被执行一次 全速运行：控制器循环执行任务。任务运行一次后的睡眠时间为（ms）：MAX(本次任务实际运行时间/10, T)，T 为项目级代码配置（LK221C 此值为 10） 状态和事件触发型任务，系统默认的执行时间为 10ms（max.），执行完一次后，剩余时间进入睡眠状态，直到执行时间结束，控制器重新检测任务触发信号，满足条件，则任务重新开始执行
优先级		多任务时，需要设置各个任务的执行顺序，从 0~31 优先级依次降低，任务按照优先级顺序执行，如果要某任务优先执行，需要不关联冗余任务并且优先级设置为最高优先级
关联冗余任务		为控制器冗余功能指定一个 IEC 任务，只允许关联到一个 IEC 任务
运行周期		任务类型为周期运行时，需要设置运行周期，可设置范围 1~60000ms
事件变量名		任务类型为状态或事件触发时，需要手动输入关联的事件变量，该变量只能为 PRG、全局变量组或者通道数据中的 BOOL 型变量
看门狗	使能	勾选后，启用看门狗功能
	时间	所有任务被执行一次的最大监控时间，设置范围 1~60000ms 当为周期运行任务时，设置值必须大于运行周期，其他任务类型保持默认值即可 看门狗总时间=一次看门狗时间*灵敏度，当任务实际执行时间超过看门狗总时间，则控制器复位用户工程，重新开始执行

	灵敏度	看门狗记录次数，设置次数 1~6 次

（4） 单击**确定**按钮。

8.2.2 添加任务调用

8.2.2.1 概述

程序块需要被任务调用后，才能执行。任务中调用的 POU 按照从上到下的顺序被执行。可通过程序块右键菜单或拖拽方式添加程序 POU 到任务节点下。

8.2.2.2 要求

- 程序块已创建

8.2.2.3 步骤

任务要调用程序块，请按以下步骤操作：

（1） 在工程管理树，右击程序 POU。

（2） 选择“追加”—任务名称。

POU 被添加到任务节点下。

也可选中程序 POU，拖拽到任务节点下。

（3） 选中任务下已被调用的程序 POU，按住鼠标左键+Shift，拖拽 POU 调整调用顺序。

8.2.3 修改任务属性

8.2.3.1 要求

AutoThinkV4 软件在离线状态

8.2.3.2 步骤

要修改任务属性，请按以下步骤操作：

（1） 在工程管理树，右击任务节点。

（2） 选择“编辑任务”命令。

将弹出“任务配置”对话框。

（3） 根据任务属性设置表修改参数值。

（4） 单击**确定**按钮。

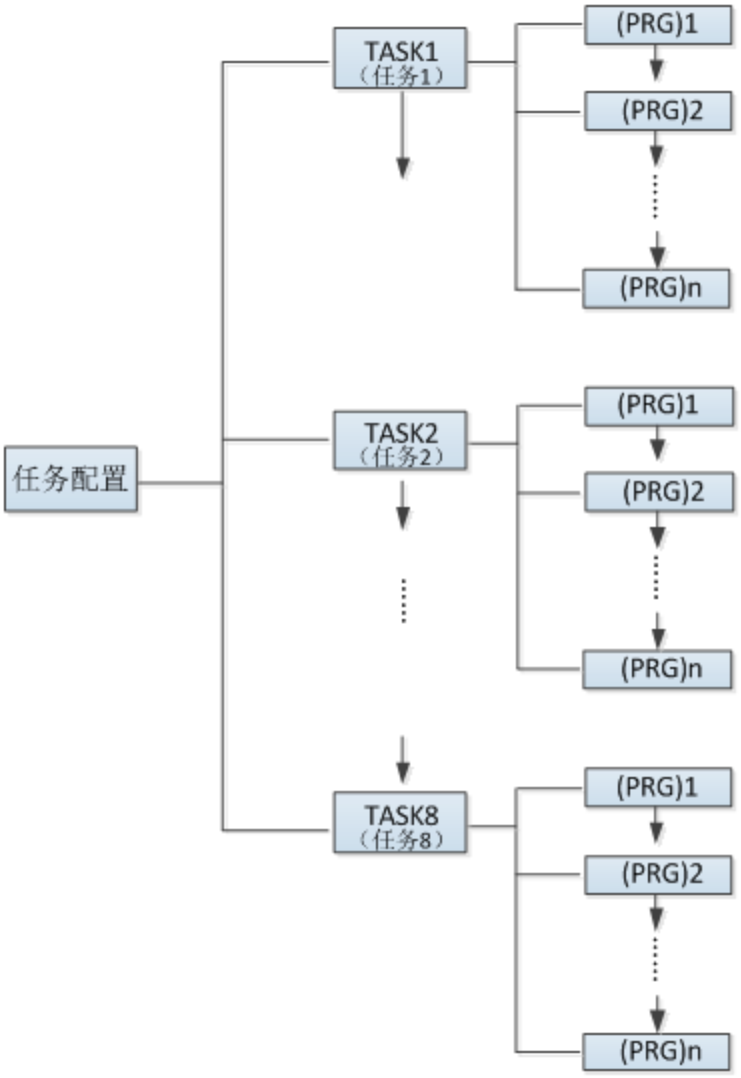
8.2.4 任务执行规则

8.2.4.1 概述

任务的执行应该遵循以下原则：

- 一旦任务的条件被满足，该任务就被执行；
- 如果多个任务的条件都有效，那么将按照任务优先级顺序执行任务；
- 任务中的程序，将按照从上到下的顺序被调用。

任务与程序的调用关系及执行顺序如图所示。



任务与程序的调用关系示意图

8.2.5 删除任务

8.2.5.1 步骤

要删除任务，请按以下步骤操作：

（1） 在任务配置节点下，右击任务名。

（2） 单击“删除任务”命令。

任务从“任务配置”节点下被移除。

8.3 数据存储

8.3.1 数据的含义

可编程控制器用于生产过程或设备的控制，其控制要求必须通过程序来实现，程序的基本元素是指令，指令由操作符和操作数构成。其中操作数分两类，一类是来自 I/O 模块的实际数据，例如行程开关的通（ON）或断（OFF）、温度的高低和流量的大小等；另一类是来自内部功能模块和程序的运算结果或中间结果。

数据的外部表现形式是变量名，例如行程开关的变量名为 SW1，当行程开关接通时，SW1=ON（或 1 或 TRUE）；当其断开时，SW1=OFF（或 0 或 FALSE）。数据的内部存储形式是 CPU 模块的存储器某位（bit）、字节（byte）或字（word）状态的变化。

例如当行程开关接通时，相应的存储器某位（bit）为 1；当行程开关断开时，相应的存储器某位（bit）为 0。当行程开关 SW1、SW2 接通时，其状态通过 DI 模块送到 CPU 模块存储器，例如输入区 I321 字节的位 0=1、位 1=1；再通过逻辑梯形图（LD）运算，其运算结果 LA1 存入 CPU 模块存储器，例如输出区 Q321 字节的位 0=1；最后通过 DO 模块输出，使灯 LA1 亮。

PLC 中的 DI 和 DO 数据

8.3.2 数据区

存储器中的数据区用于存储数据，该区分为输入区、输出区、中间区、全局变量区、保持区。

“工程属性”对话框中可查看各数据区大小。

工程中数据区大小

各个区的主要功能如下：

1. 输入区（I）

在每个扫描周期的首端，CPU 对输入点进行采样，并将采样值存储到内存储器的输入区。可按位（BIT）、字节（BYTE）、字（WORD）、双字（DWORD）来存取输入区。

2. 输出区（Q）

在每个扫描周期的末端，CPU 将内存储器的输出区的数据传送到物理输出点上。可按位（BIT）、字节（BYTE）、字（WORD）、双字（DWORD）来存取输出区。

3. 中间区（M）

中间区用来存储程序的中间结果、工作状态或其他控制信息。可按位（BIT）、字节（BYTE）、字（WORD）、双字（DWORD）来存取中间区。

%MB0~%MB3999，4000 字节具有默认掉电保持功能。以上地址区间的变量，即使变量定义时没有勾选掉电保持，也默认具有掉电保持属性。

4. 全局变量区（N）

随机区用来存储用户所定义的未指定地址的变量。

N 存储区也属于 PLC 的中间寄存器区，用于存储和管理中间过程产生的数据和状态。与 M 存储区不同的是，N 存储区只能通过变量的方式来访问和调用。

N 存储区中的变量地址，是系统自动分配而用户无法指定的。N 区中的变量数据类型不仅有位、字节、字和双字，还有 REAL、TIME、INT 等其他众多数据类型。另外，除了数据变量外，定义的功能块变量也存储在 N 存储区。

N 存储区可以读写，可以被输入和强制。

5. 保持区（R）

“保持区”存储掉电保持变量，若将数据或变量定义为保持型（掉电保护值为 TRUE），则当 CPU 模块掉电时，系统会自动保存该数据，待重新上电后，又会自动恢复该数据。

该区调用方式与 N 区一致，也是通过变量的方式访问，无法指定地址。

R 存储区变量可以读写，可以被输入和强制。

对于一个没有指定直接地址的变量，如果没有设置掉电保持，则该变量存储在 N 区；若设置掉电保持，则该变量存储于 R 区，具有掉电保持功能。

8.3.3 数据存储格式

数据存储格式分为字节、字、双字、位这 4 种方式，以中间区（M）为例，叙述数据存储格式。

1. 字节数据

字节数据的存储格式如下图所示。

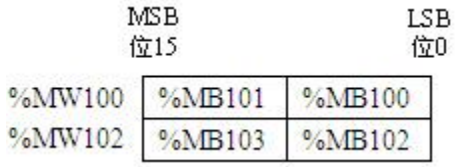
字节数据

其中 MSB 表示数据的最高位（位 7），LSB 表示数据的最低位（位 0）。

“%”表示地址，“M”为存储器标识，“B”表示字节数据，占 8 位。

2. 字数据

字数据的存储格式如下图所示。



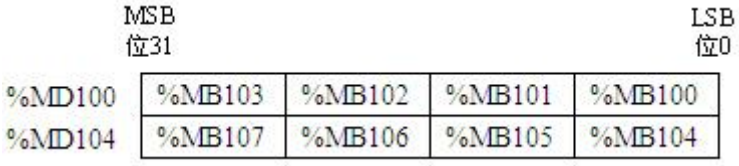
字数据

其中 MSB 表示数据的最高位（位 15），LSB 表示数据的最低位（位 0）。

“%”表示地址，“M”为存储器标识，“W”表示字数据，占 16 位。

3. 双字数据

双字数据的存储格式如下图所示。



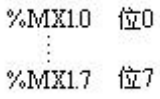
双字数据

其中 MSB 表示数据的最高位（位 31），LSB 表示数据的最低位（位 0）。

“%”表示地址，“M”为存储器标识，“D”表示双字数据，占 32 位。

4. 位数据

位数据的存储格式如下图所示。



位数据

“%”表示地址，“M”为存储器标识，“X”表示单个位，即占 1 位。

8.4 数据类型

8.4.1 标准数据类型

8.4.1.1 概述

软件支持的标准数据类型包括布尔型、整型、实型、时间型、数组、指针、字符串。

数据类型、上下限及存储位数详见下表。

数据类型描述表

类型	类型名称	数据下限	数据上限	存储位数
BOOL	布尔型	0	1	1 bit
BYTE	字节型	0	255	8 bit
WORD	字型	0	65,535	16 bit
DWORD	双字型	0	4,294,967,295	32 bit
SINT	短整型	-128	127	8 bit
USINT	无符号短整型	0	255	8 bit
INT	整型	-32,768	32,767	16 bit
UINT	无符号整型	0	65,535	16 bit

DINT	32 位整型	-2,147,483,648	2,147,483,647	32 bit
UDINT	无符号 32 位整型	0	4,294,967,295	32 bit
REAL	实型	-3.402823466E+38	-1.175494351E-38	32 bit
		1.175494351E-38	3.402823466E+38	
LREAL	长实型	-1.7976931348623158e+308	-2.2250738585072014e-308	64 bit
		2.2250738585072014e-308	1.7976931348623158e+308	

8.4.1.2 布尔

布尔型数据为逻辑量，占用存储器 1 位（bit)，其状态为“0”或“1”、“TRUE”或“FALSE”、“真”或“假”。

8.4.1.3 整型

整型数据分为 BYTE、WORD、DWORD、SINT、USINT、INT、UINT、DINT 和 UDINT。

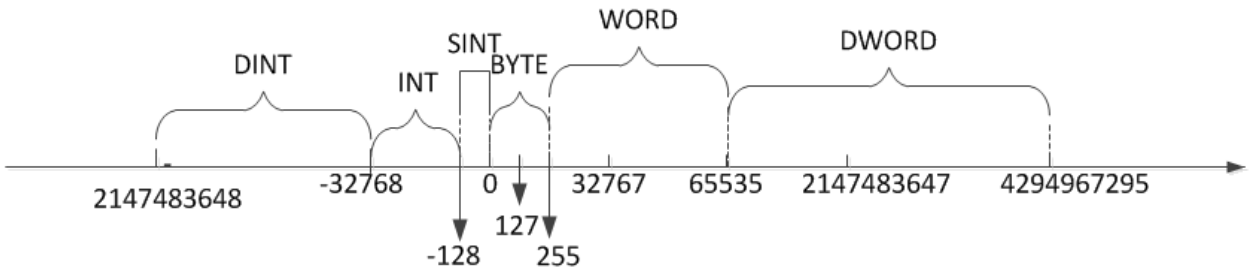
整型数据的特点是无小数位。

- 整数常量识别规则

  AutoThinkV4 中整数常量识别的顺序如下：

  BYTE（0～255）->SINT（-128～127）->USINT（0～255）->WORD（0～65535）->INT（-32768～32767）->UINT（0～65535）->DWORD（0～4294967295）->DINT（-2147483648～2147483647）->UDINT（0～4294967295）

  即正数优先被识别为无符号型数据，下图更形象的说明了这一点。



整数常量识别规则

- 类型优先级

  AutoThinkV4 中整型数据类型优先级由低到高如下：

  SINT（4）->BYTE（5）->USINT（6）->INT（7）->WORD（8）->UINT（9）->DINT（11）->DWORD（12）->UDINT（13）

- 类型提升规则

  - 支持数据类型提升的运算符：ADD、SUB、MUL、DIV、MOD、SEL、MAX、MIN、LIMIT、MUX。运算结果按操作数的最大类型进行提升，如小于平台机器字长，则再按

平台机器字长提升。对于 MOD 运算，提升后的有无符号由第一个操作数的有无符号类型决定。

- 比较运算/选择运算类型提升规则

  表达式：num1 operator num2

  运算符 MAX、MIN、LIMIT、GT、LT、GE、LE、EQ、NE 的运算结果数据类型由操作数的最大类型和平台机器字长决定，规则如下：

  1：如果最大类型为 U32(UDINT/DWORD)，则将操作数 num1 和 num2 均提升为 F64(LREAL)后参与比较/选择运算。

  2：其余情况,操作数 num1 和 num2 均提升为最大类型后参与比较/选择运算。

当范围较大的类型转换为较小的类型时，可能丢失信息。

不建议同时使用有符号数与无符号数进行运算，某些情况下达不到预期的结果。

- 使用注意事项：
  - AutoThinkV4 中正数整型常量被识别为无符号型数据，无符号型优先级高。
  - 不要使用有符号数与无符号数共同参与一个基本运算，除非运算结果符合预期。
  - 对于多个输入的基本指令，输入不要全为常量，除非结果符合预期。
  - 对于结果可能为负数的基本运算，若使用 16 位整数作为输入，最好使用 INT 型变量，若使用 32 位整数作为输入，最好使用 DINT 型变量。

8.4.1.4 实型

实型包含 REAL 和 LREAL，实型数据的特点是有小数位，既可表示成整数和小数形式，也可表示成指数形式。

LREAL 负数取值范围为 -1.7976931348623158e+308 到 -2.2250738585072014e-308，正数取值范围为 2.2250738585072014e-308 到 1.7976931348623158e+308。

当有实型数据参与运算时，有时会显示特殊字符，含义如下：

- 1#INF：表示无穷大数，分为正无穷和负无穷，它由非 0 的有限数除 0 得到，或者超过了实型的数据上、下限。实型除零不报错。
- 1#QNAN0：表示非数，它是由一些无意义的运算引起的。

8.4.1.5 时间型

时间型数据用于定义时间（TIME）、时刻（TOD）、日期（DATE）、日期时刻（DT）。

序号	变量名	直接地址	变量说明	变量类型	初始值	掉电保护
0001	TimeVar1		时间	TIME	T#0MS	FALSE
0002	TimeVar2		日期	DATE	D#1970-01-01	FALSE
0003	TimeVar3		日期时刻	DT	DT#1970-01-01-00:00:00	FALSE
0004	TimeVar4		时刻	TOD	TOD#00:00:00	FALSE

时间型数据

- 时间（TIME）

时间数据的标识符为 TIME，用于定义操作时间。定义时间的格式有 4 种：TIME#、time#、T#、t#，格式符#后为时间数据，依次为时、分、秒和毫秒。如图中 TimeVar1 所示。

使用时，时间的最高位允许超过其上限，而低位不允许超过其上限。

例：

正确的定义

TIME1 := T#14ms;

TIME1 := T#100S12ms; (*允许最高位超过其数据上限*)

TIME1 := t#13d12h34m15s;

错误的定义:

TIME1 := t#5m68s; (*低位超限*)

TIME1 := 15ms; (*缺少提示符 T#*)

TIME1 := t#4ms13d; (*输入顺序有误*)

- 日期（DATE）

日期的标识符为 DATE，用于存储日期。定义日期的格式有 4 种：D#、d#、DATE#、date#，格式符#后为日期数据，依次为年、月、日。如图中 TimeVar2 所示。

- 日期时刻（DT）

日期时刻的标识符为 DT，用于存储日期和时刻。定义日期时刻的格式有 4 种：DT#、dt#、DATE_AND_TIME#、date_and_time#，格式符#后为日期和时刻数据，依次为年、月、日、时、分、秒。 如图中 TimeVar3 所示。

- 时刻（TOD）

时刻数据的标识符为 TOD，用于存储时刻。定义时刻的格式有 4 种：TOD#、tod#、TIME_OF_DAY#、time_of_day#，格式符#后为时刻数据，依次为时、分、秒，其中秒为实数，即秒可有小数。如图中 TimeVar4 所示。

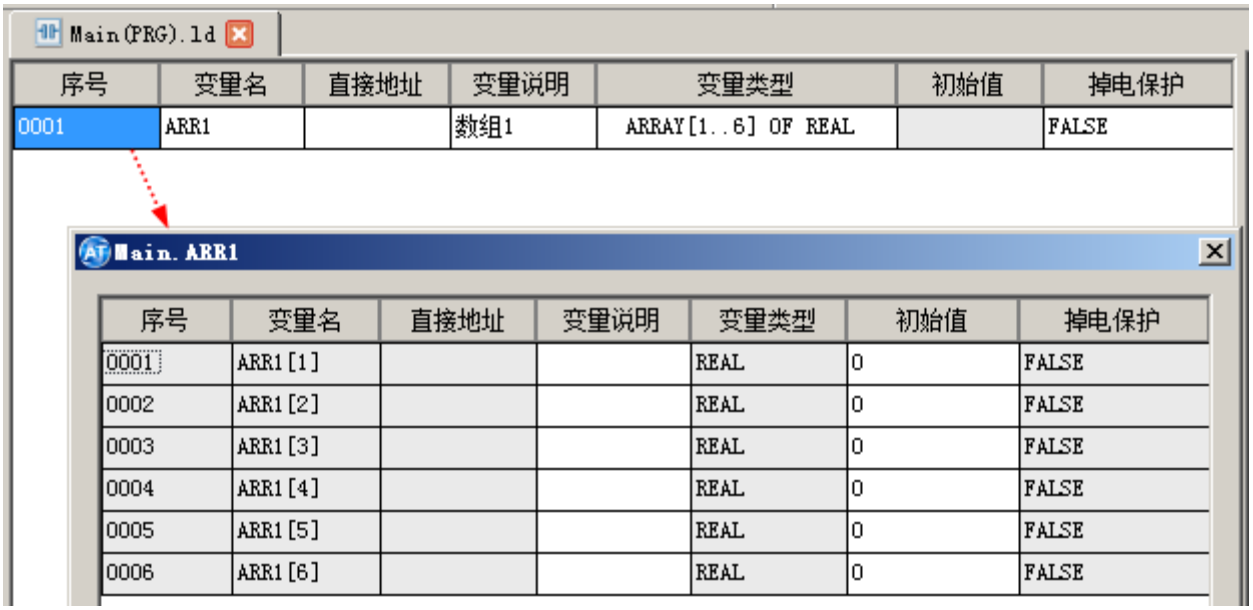
8.4.1.6 数组

数组（ARRAY）将同种类型的数据元素有序的组合成一个有机的整体，以便于引用。

可定定义多维数组。每个数组需定义起始下标和结束下标，缺省从 0 开始，定义数组时，Start 下标值必须小于等于 End 下标

数组元素下标用[]标识，例如 2×2 二维数组，起始下标为 1，结束下标为 2，则 4 个元素的下标分别为[1,1]、[1,2]、[2,1]、[2,2]。

例如：创建一个一位数组 ARR1，数组下标从 1~6，变量类型为 REAL，如下图所示。



为数组元素赋初始值

1. 要求

变量已添加

2. 步骤

要将变量类型定义为数组，请按以下步骤操作

（1） 双击变量类型

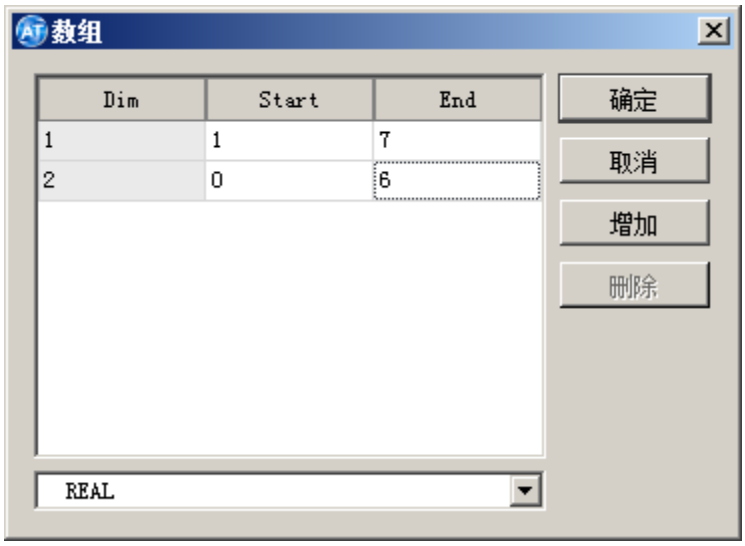
（2） 下拉框中选择 ARRAY 类型

将弹出数组定义对话框。

（3） 输入数组起始下表和结束下标

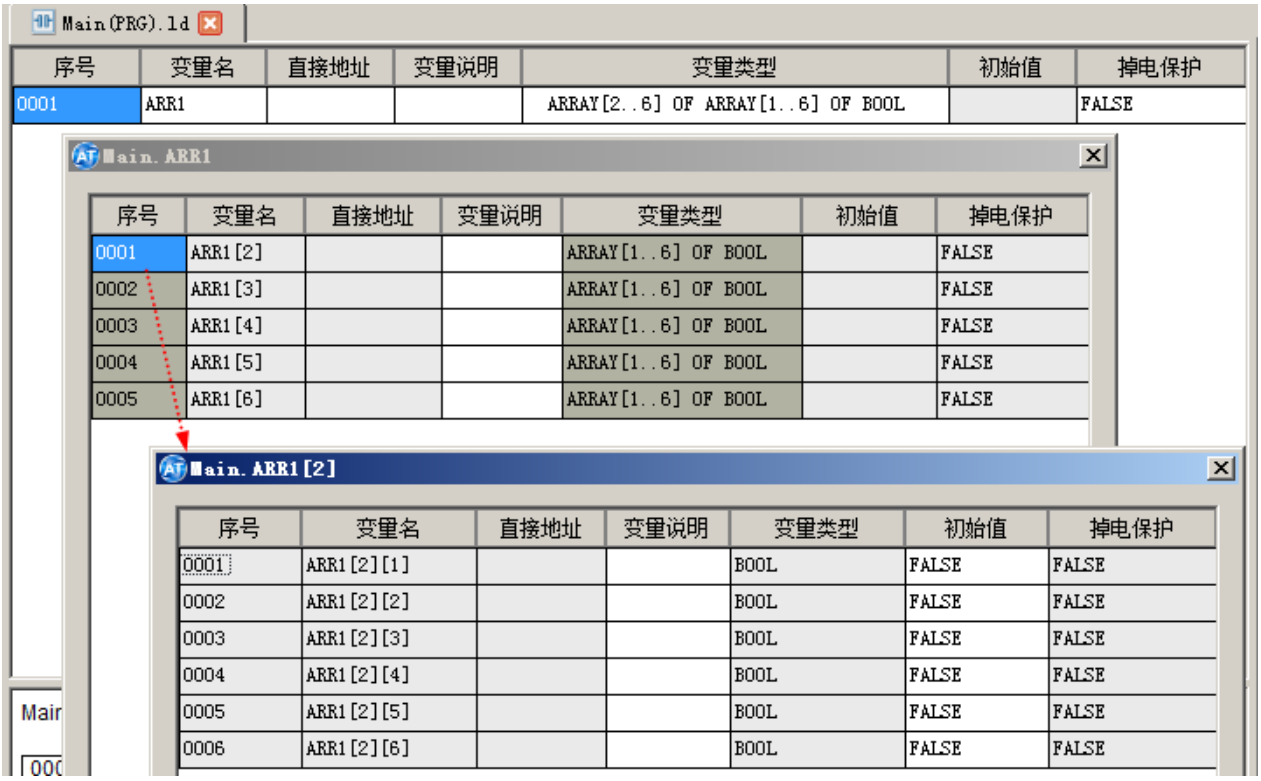
（4） 数据类型下拉框中，选择数组元素类型

（5） 单击**添加**按钮，添加多维数组



数组定义对话框

当数组元素类型为数组、结构体、功能块时，可双击变量序号，查看子变量元素。



数组子项定义窗口

3. 引用数组元素

程序中可引用数组元素，例如引用二维数组元素 ArrayVar[1,2]，以 ST 语言为例：

```
Var1 := ArrayVar[1,2]  (*二维数组元素[1,2]*)
```

在 ST 语言编辑器中对上述数组元素赋值，采用的格式为：“ARR1[1]:=10;”，或者定义另外一个数组变量 ARR2，且保证该数组维数（下标）以及数据类型与 ARR1 完全一样，这时，可以采用的格式为：“ARR1:= ARR2;”。

8.4.1.7 指针

指针是一个用来指示一个计算机内存地址的变量。在定义变量时，选择变量类型为 **POINTER**，弹出“设置指针指向的类型”对话框，在 **POINTER TO** 下拉列表中选择指针的指向类型。

指针类型定义对话框

这里，指针支持如下两种操作：

P := ADR(VAR1);　(*P 为指针变量，VAR1 为普通变量，这里为将 VAR1 的地址赋给变量 P*)

VAR1 := VAL(P);　(*P 为指针变量，VAR1 为普通变量，这里为将变量 P 的值赋给 VAR1*)

说明：

- 不支持对奇地址取值，如 p1:pointer to word，p1: = 12548637，对 p1 取值即 VAL 运算会引起控制器复位。
- 变量或功能块地址在程序运行时被存储在指针中。
- 指针可指向任何数据类型或功能块，甚至可用于自定义类型。地址操作符 ADR 的功能是分配变量或功能块的地址到指针。
- 取值操作符 VAL 的功能是对指针指向内存进行取值。
- 指针以字节为单位计算！可以通过使用说明 p ＝ p+SIZEOF（p 指向的变量）像在 C_Compiler 一样进行计算。

8.4.1.8 字符串

1. 字符串变量

字符串类型变量可包含任何一串字符。声明时变量的长度就决定为变量保留多大的存储空间。

用户选择变量的数据类型为 STRING 时，要求用户设置字符串数据的长度，如下图所示。

字符串长度设置

字符串变量的长度范围是 1~1000，表示可以输入 1~1000 个字符，每个字符占一个字节（8 bit）的地址空间。若没有指定，系统默认长度为 80 个字符。

用户在设置字符串类型变量的初始值时，需要在内容的始末添加英文单引号。如 'this is a string'。

2. 字符串常量

字符串常量的长度范围是 1~1000 个字符，用单引号括起来表示。你也可以在其中插入空格和特殊字符，特殊字符定义如下：

特殊字符定义

输入字符串	含义
$R	回车
$L 或$N	换行
$T	水平制表符
$’	单引号
$$	$符号
$P	换页

字符串型数据只能由 ASCII 码字符组成。

8.4.2 自定义数据类型

8.4.2.1 创建结构体类型

有时需要将不同类型的数据组合成一个有机的整体，以便于引用，结构体（STRUCT）就具有此功能。

结构体由多个成员（或分量）组成，每个成员有成员名及类型，类型可不同或相同。

1. 组态规则

- 不能直接或间接组态为自身结构体类型。
- 成员类型组态为数组时，数组元素类型不能组态为自身结构体类型。

- 程序中引用结构体（STRUCT）的成员，必须在变量区声明结构体。

对结构体进行实例声明后，通过“实例名.成员名”调用结构体中的成员。

2. 步骤

要创建结构体数据类型，请按以下步骤操作：

（1） 在“数据类型”节点右键菜单中，选择“添加类型”命令。

将弹出添加对话框。

（2） 输入数据类型名称。

命名规则请参考变量定义。

（3） 勾选“结构体”类型。

“数据类型”树节点下生成结构体类型

（4） 在结构体窗口，右键菜单中选择“增加变量”命令。

结构体成员被添加，缺省的成员名为“s*”，成员类型为 BYTE，可双击对应项进行修改。

成员类型为数组、结构体、功能块时，可双击“序号”查看对应成员的变量。

8.4.2.2 创建枚举类型

通过枚举，程序中的数字值可以转换为文本。

1. 组态规则

- 程序中可直接引用枚举的成员，不必在变量区声明。
- 程序中引用结构体（STRUCT）的成员，必须在变量区声明结构体。
- 枚举值范围是-32768~32767，枚举成员个数上限 65536。
- 一个工程中可定义多个枚举，枚举的成员名必须唯一

2. 步骤

要创建枚举数据类型，请按以下步骤操作：

（1） 在“数据类型”节点右键菜单中，选择“添加类型”命令。

将弹出添加对话框。

（2） 输入数据类型名称。

命名规则请参考变量定义。

（3） 勾选“枚举”类型。

“数据类型”树节点下生成枚举类型

（4） 在枚举窗口，右键菜单中选择“增加变量”命令。

枚举成员被添加，缺省的成员名为“e*”。

8.4.2.3 复制数据类型

1. 概述

枚举和结构体数据类型支持复制粘贴。在创建自定义数据类型时，可直接复制已存在的数据类型进行创建。

复制的数据类型默认以“NewType1”为名被创建。

2. 要求

- 有一个可复制的数据类型

3. 步骤

要复制数据类型，请按以下步骤操作：

（1） 右击枚举或结构体数据类型，选择“复制类型”命令。

复制数据类型

（2） 右击“数据类型”节点，选择“粘贴类型”命令。

粘贴数据类型

数据类型被复制，默认名为“NewType1”。

8.5 定义变量

8.5.1 变量概述

根据变量在工程中的使用范围，变量被分为全局变量和局部变量。

8.5.1.1 全局变量

全局变量可以在工程的多个地方同时使用，全局变量的名称在工程中必须唯一。全局变量包含全局变量组中的变量和通道变量。

8.5.1.2 局部变量

局部变量包含程序、函数和功能块中定义的变量，仅在所属 POU 中使用。如果需要在全局范围内引用局部变量，必须使用“POU 名称.局部变量名”的形式。

函数和功能块 POU 中的局部变量，按照用途可分为输入变量、输出变量、输入输出变量和中间变量。

- 输入变量（Input_Variable）：变量是从外部输入的，本功能块（或函数）对该变量只可读。
- 输出变量（Output_Variable）：变量是向外部输出的，本功能块对该变量可读写；其它功能块只可读。
- 输入输出变量（Input/Output_Variable）：兼有输入变量和输出变量的特性，其它功能块和本功能块对该变量都可读写。
- 中间变量：是块实现逻辑运算时所需要的临时变量；该变量不与外部连接，只在它所属的块中有效。在功能块查看信息中被称为参数（VAR_TEMP）。

8.5.2 变量命名规则

在工程中，变量组、变量的命名规则一致。

定义变量的名称只能由字母、数字、下划线组成，名称长度最多不超过 32 字节。

变量名的命名遵循如下规则：

- 变量名必须以字母或者下划线开头，不能以数字开头。
- 变量名识别下划线，例如 AB_CD 和 ABC_D 被认为是两个不同的变量名。
- 变量名不区分大小写。例如 VAR1、Var1 和 var1 表示相同的变量。
- 变量名不能为空，且不能包含空格。例如 AB CD 是错误的变量名。
- 变量名中不能包含特殊字符，如中划线“-”和加号“+”等。例如 AB-CD 和 AD+CD 是错误的变量名。
- 变量名不能与类型名（包括自定义类型）、POU 名、枚举名、任务名或类型转换函数名重名，不能以 AT_开头。
- 变量名不能与关键字相同。

8.5.3 变量寻址规则

为一个变量指定一个直接地址（变量地址），是指把一个变量连接到确定的内存储器地址，即是对输入区（I）、输出区（Q）和中间区（M）地址的映射。

直接地址使用规则：

- I 区、Q 区、S 区变量的直接地址与实际硬件通道地址关联，变量定义时，在直接地址中输入关联的硬件通道地址即可；
- M 区变量不关联实际通道地址，用户自己定义变量的直接地址；
- 直接地址定义时，采用字节对齐原则，即：输入的直接地址（首地址）必须满足能被变量类型的数据长度（按字节算）和寻址字节长度同时整除。如，定义 Byte 型变量，数据长度为 1 字节，按双字寻址（4 Byte），则直接地址必须能被 1 和 4 同时整除，例如：%ID8、%MD12 等；
- 寻址方式 X、B、W、D 只决定首地址对齐方式，实际寻址的数据长度由数据类型决定。例如，Real 型变量，数据长度为 4 字节，直接地址为%IW12，则实际读取数据为从%IW12 首地址开始的 4 字节数据；
- 时间型变量、结构体变量、数组元素为复杂类型（枚举、指针、字符串、结构体、数组、功能块、时间型）的数组不支持使用直接地址；
- 对于 WORD、BYTE 型通道数据或 Modbus 通讯数据，可通过直接地址按位或者字节读取相应数据。例如：通道%IW8，通过访问直接地址%IX8.0~%IX8.7、%IX9.0~%IX9.7，分别读取 16 位数据。

示例：在输出区（Q）定义的变量-OUT_SV2，变量地址设置为：%QX2.1（*Q 输出区，X 位寻址，位变量*），变量访问的输入格式符含义如下：

含义		标识
地址标识符		%
存储区位置		输入区-I、输出区-Q、中间区-M
寻址方式		位寻址-X（1 bit）
		字节寻址-B （1 Byte）
		字寻址-W （2 Bytes）
		双字寻址-D（4 Bytes）
首地址	字号	存储区的字号
	位号	存储区的位号。根据数据寻址方式的不同，位号可省略，如果有位号时，位号前必须加分隔符“.”，位号 0~7

使用 M 区地址时建议关联变量名，否则编译时不做处理，同时该地址数据不进行数据同步。如，变量定义区 M 区地址已使用到%MB100，用户组态时直接使用地址%MB101 而没有为其关联变量名，则在编译时，M 区的地址使用显示为%MB100，同时，%MB101 的数据不进行冗余同步。

8.5.4 定义一个变量

8.5.4.1 概述

组态用户逻辑前，需要创建全局变量或局部变量。变量定义时，可手动添加变量，也可以自动定义变量。

手动定义变量需要添加变量后，将该变量关联到程序元素。

自动定义变量是为元素直接关联变量时，再进行变量声明。

8.5.4.2 变量属性说明

定义变量时，需要定义变量相关属性，包括变量名称、变量说明、数据类型、关联直接地址、设置变量初始值、掉电保持属性等。

变量属性说明

变量属性	组态说明	设置场景
变量名称	一般定义为关联通道的仪表位号，或用户定义的中间变量名称。	手动定义变量/自动定义变量
直接地址	关联到模块的硬件通道地址，格式与通道地址保持一致。函数和功能块中的变量无直接地址属性。	
变量说明	说明该变量代表的含义，如，汽化炉压力、冷却塔温度。	
变量类型	如关联了直接地址，则变量类型与通道数据类型保持一致。 功能块和结构体在使用时，需要声明一个变量实例，变量类型选择已创建的功能块或结构体。	
初始值	程序初次运行时，采用变量的初始值进行计算，可根据工艺要求设置初始值。	
掉电保持	变量设置了掉电保持，则暖复位时，变量值保持不变。函数没有掉电保持属性。	
类别	变量类别根据当前所在 POU 不同，可选类别也不同。 1 程序 POU 中声明的变量，可选择局部变量或全局变量 2 函数 POU 中声明的变量，可选择输入变量、输入输出变量、中间变量、全局变量 3 功能块 POU 中声明的变量，可选择输入变量、输入输出变量、中间变量、输出变量、全局变量 全局变量类别，需要指定到某个全局变量组，确定后，该变量被添加到对应全局变量组下。 选择其他类别，变量将被添加到当前 POU 的对应类别下	自动定义变量

8.5.4.3 手动定义变量

1. 概述

组态用户逻辑前，需要手动创建变量，再将变量关联到用户逻辑的元素。

2. 要求

已打开全局变量组或 POU 窗口。

3. 步骤

要手动定义变量，请按以下步骤操作：

（1） 在全局变量组或 POU 窗口，右键单击空白区域。

（2） 在右键菜单中，选择“添加变量”命令。

变量被添加，全局变量默认为 gn，POU 局部变量被默认为 pn，n 为软件自动分配的序号。

（3） 手动设置变量属性。

4. 参考信息

变量定义规则

变量寻址规则

变量属性说明

8.5.4.4 自动定义变量

1. 概述

在组态用户逻辑过程中，定义变量。为元素关联变量名后，可自动弹出变量声明对话框，进行变量定义。

2. 要求

- 变量名未在当前 POU、全局变量组和通道变量中定义过
- 工程选项中已勾选“变量自动声明”选项

3. 步骤

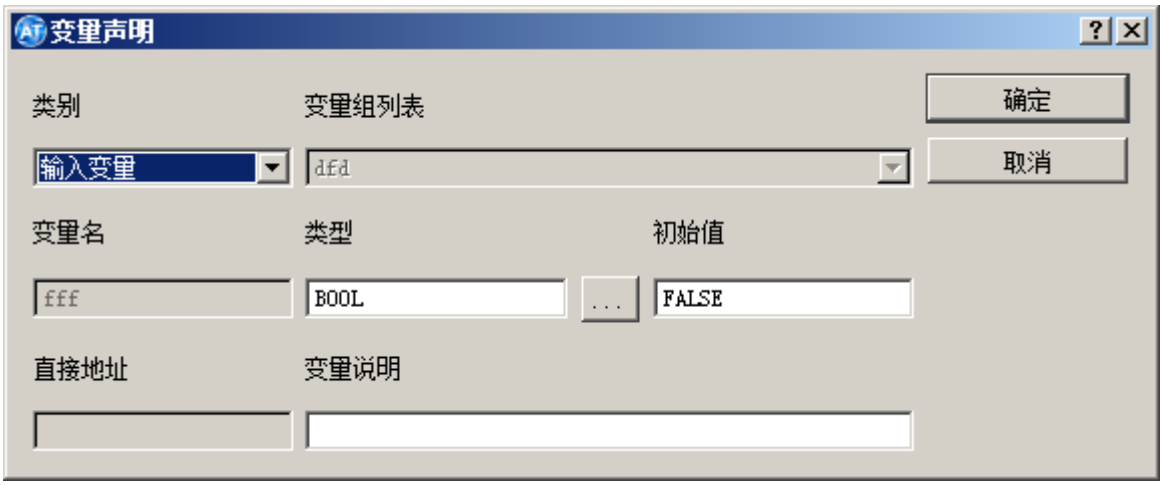
要自动定义变量，请按以下步骤操作：

（1） 在程序区，单击元素的变量名占位符“？？？”。

（2） 输入变量名。

（3） 按回车键。

此时，当前元素已关联这个变量名，同时弹出“变量声明”对话框。



变量声明对话框

（4） 设置变量属性。

变量将被添加到指定的变量类别下。

4. 参考信息

变量属性说明

变量命名规则

变量寻址规则

8.5.4.5 掉电保持

控制器提供掉电保持功能。控制器运行时，周期性备份掉电保持数据，并在掉电后，仍能保存这些数据。

以下数据具有掉电保持属性：

- R 区数据，大小为 128KBytes。
- M 区前 4000 个字节，即%MB0~%MB3999。
- AutoThinkV4 软件已设置掉电保持属性的变量。

8.5.4.6 为元素关联变量

1. 概述

手动添加的变量，需要将其关联到用户程序中的元素上，该变量才能参与逻辑运算、或输出逻辑值。

关联变量时，需要注意以下几点：

- 当手动关联变量时，如元素关联其他 PRG 局部变量，输入格式为：POU 名.变量名，其他变量类型直接输入变量名即可。

- 当关联复杂类型变量，如功能块、结构体时，输入格式为：实例名.变量名/成员名。
- 当关联通道变量时，通道数据类型必须和元素数据类型匹配，同时，输入数据只能关联输入元素、输出数据只能关联到输出元素。

2. 要求

已手动添加变量

3. 手动关联变量

将变量关联到逻辑元素，请按以下步骤操作：

（1） 在程序区，单击元素的变量名占位符“？？？”。

（2） 输入变量名

（3） 按回车键

当前元素已关联一个变量。

4. 自动关联变量

将变量自动关联到逻辑元素，请按以下步骤操作：

（1） 在程序区，选中元素的变量名占位符“？？？”。

（2） 按 F2 键

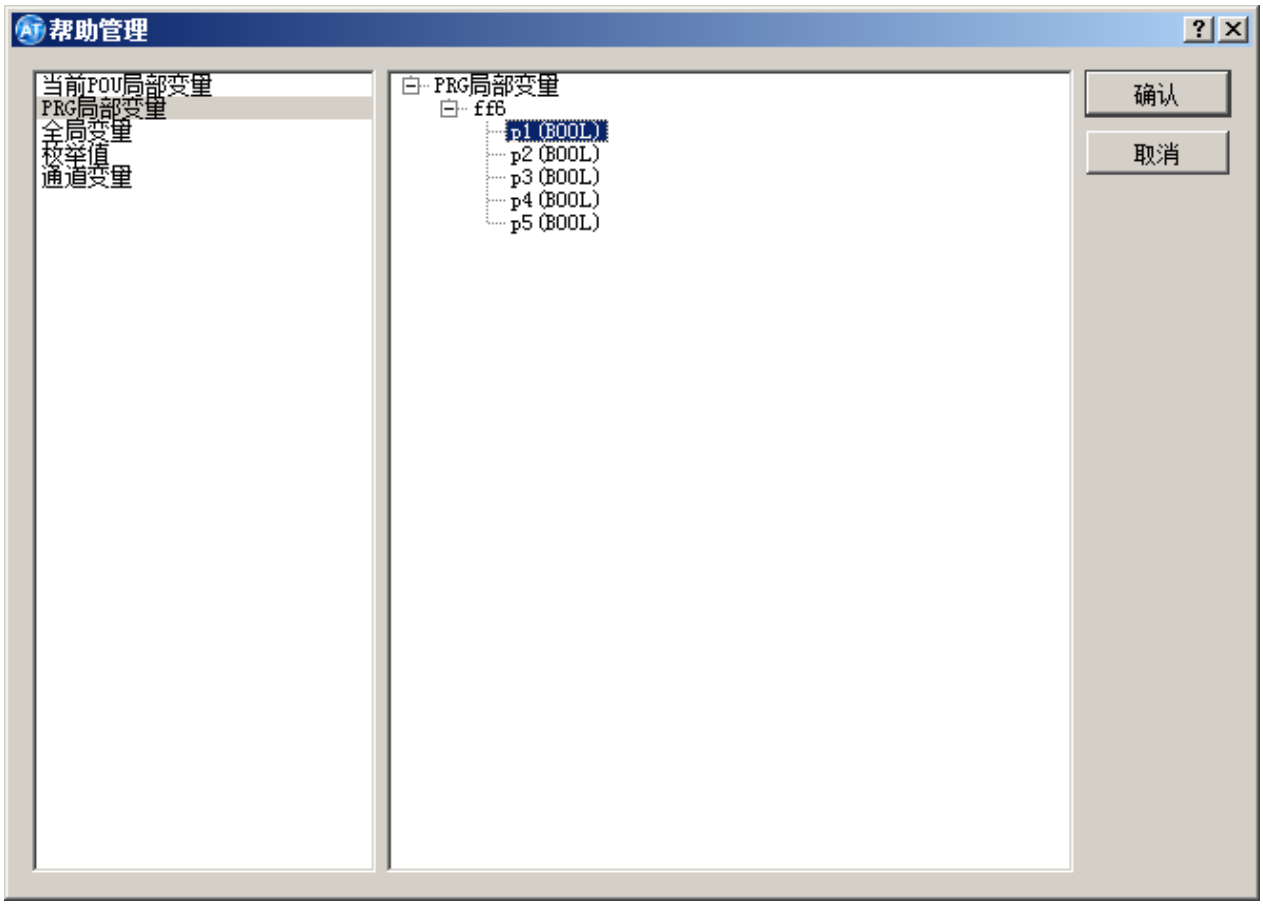
将弹出“帮助管理”窗口。

（3） 选择变量类型

可选择局部变量、全局变量、枚举值、通道变量。

（4） 选择某一变量，单击**确认**

占位符将显示为变量名。



选择变量

8.5.4.7 创建全局变量组

1. 概述

“全局变量”节点默认包含变量组 GV_Group 和 TaskDiagGroup。

- GV_Group：用于定义全局变量。
- TaskDiagGroup：为系统变量组，用于诊断任务运行状态，可直接使用其变量进行组态。

如默认的全局变量组 GV_Group 不够用，可自己创建全局变量组。

2. 步骤

要创建全局变量组，请按以下步骤操作：

（1） “全局变量”节点右键菜单中，选择“添加组”命令。

将弹出新建对话框。

（2） 输入变量组名称。

（3） 单击**确定**按钮。

全局变量节点下将生成新的变量组。

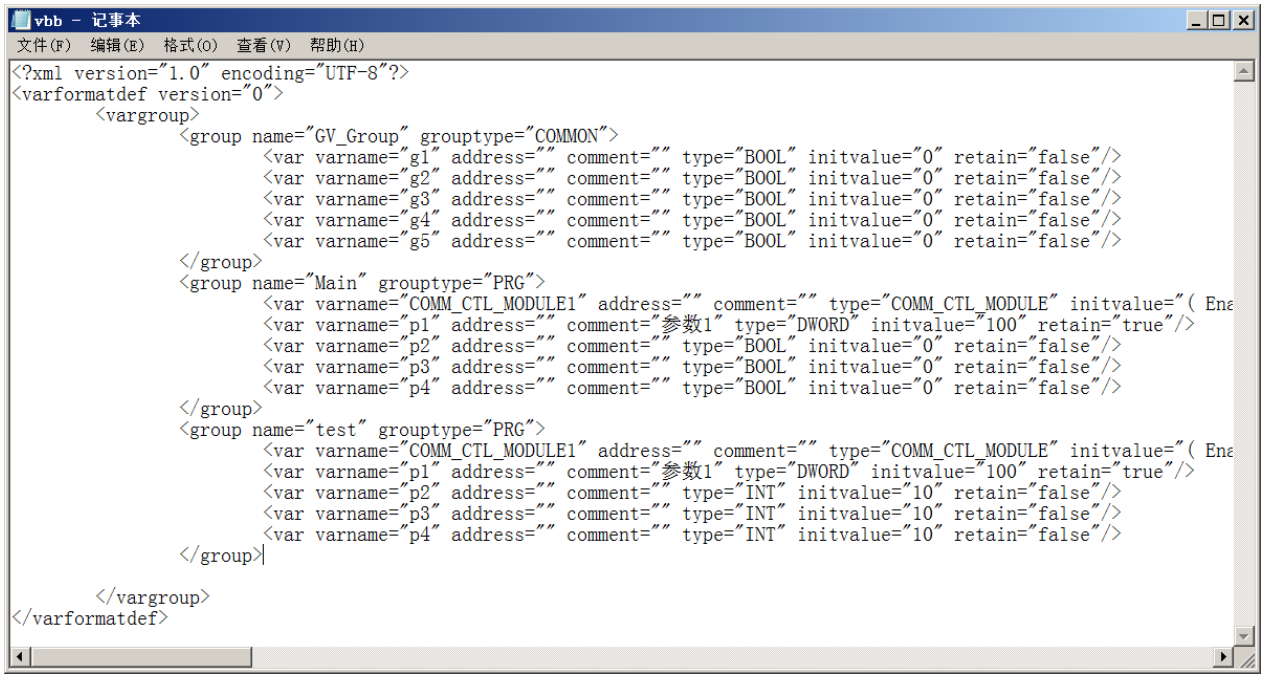
3. 参考信息

变量命名规则

8.5.5 导入变量

8.5.5.1 概述

将指定的变量表格导入到工程中，数据表格式如下图所示。



导入的数据表样式

可导入的数据分为两类：全局变量组中的变量和程序（PRG）中的变量。

1. 全局变量组中的变量

xml 文件名称无要求，文件中变量组均需以<group name="" grouptype="COMMON">行开始，以</group>行结束。执行导入操作时，如果工程中没有相应的变量组，则工程自动创建 xml 文件中对应的变量组，并且导入相应的数据。

2. 程序（PRG）中的局部变量

xml 文件名称无要求，文件中程序均需以<group name="" grouptype="PRG">行开始，以</group>行结束。执行导入操作时，如果工程中没有相应的程序，则 xml 文件中对应的程序变量不能被导入，在信息窗口中会提示“不存在 PRG 局部变量组：**，导入失败”。

导入变量前可选择变量的导入模式，缺省为**增量覆盖**导入。导入模式的选择请参见工程配置。

8.5.5.2 要求

导入的 xml 文件已编辑完成，并检查无误。

8.5.5.3 步骤

要导入变量，请按以下步骤操作：

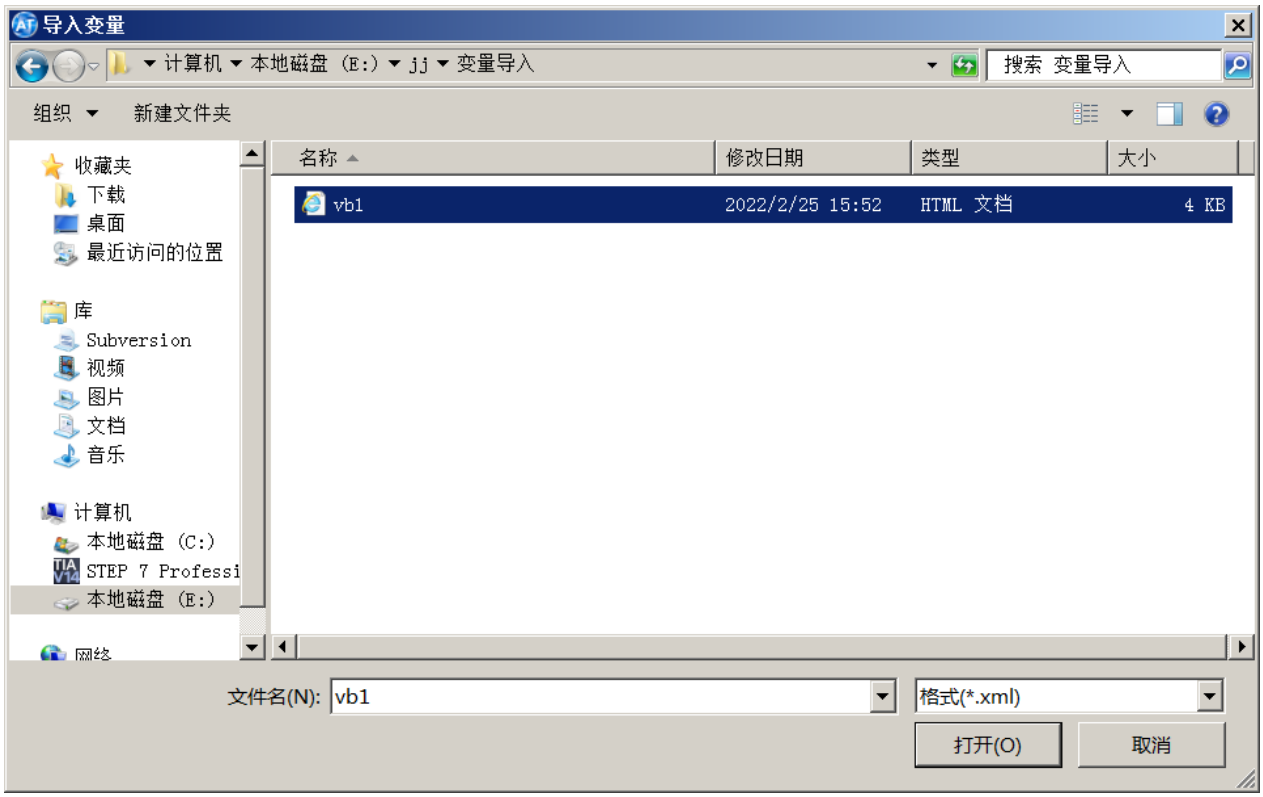
（1） 单击【工程】菜单。

（2） 选择【导入变量】命令。

将弹出导入模式确认提示框，如图所示。

导入变量提示框

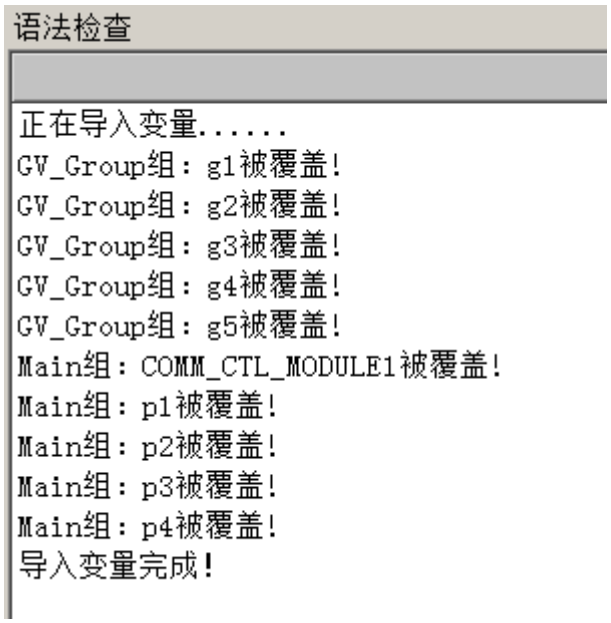
（3） 单击**确定**，将弹出文件选择对话框。



选择导入文件

（4） 选择要导入的 xml 文件，单击**打开**。

xml 文件中的变量导入到相应的程序中或者全局变量组中。在信息输出窗口提示导入信息。如图所示。



导入后信息输出窗口显示

8.5.6 导出变量

8.5.6.1 概述

如果需要对工程中的变量进行批量编辑、或导入到其他工程，您需要先导出工程中的变量。

可导出的变量包括全局变量组和程序（PRG）中的变量。您可以根据需要选择对应的变量组进行导出。需要注意的是，系统全局变量不支持导出。

变量以 xml 文件形式被导出，在 xml 文件中以变量组为节点显示导出变量。变量支持的所有属性均会导出，例如变量名、直接地址、变量说明、类型、初始值、掉电保护等属性等。

8.5.6.2 要求

工程已打开

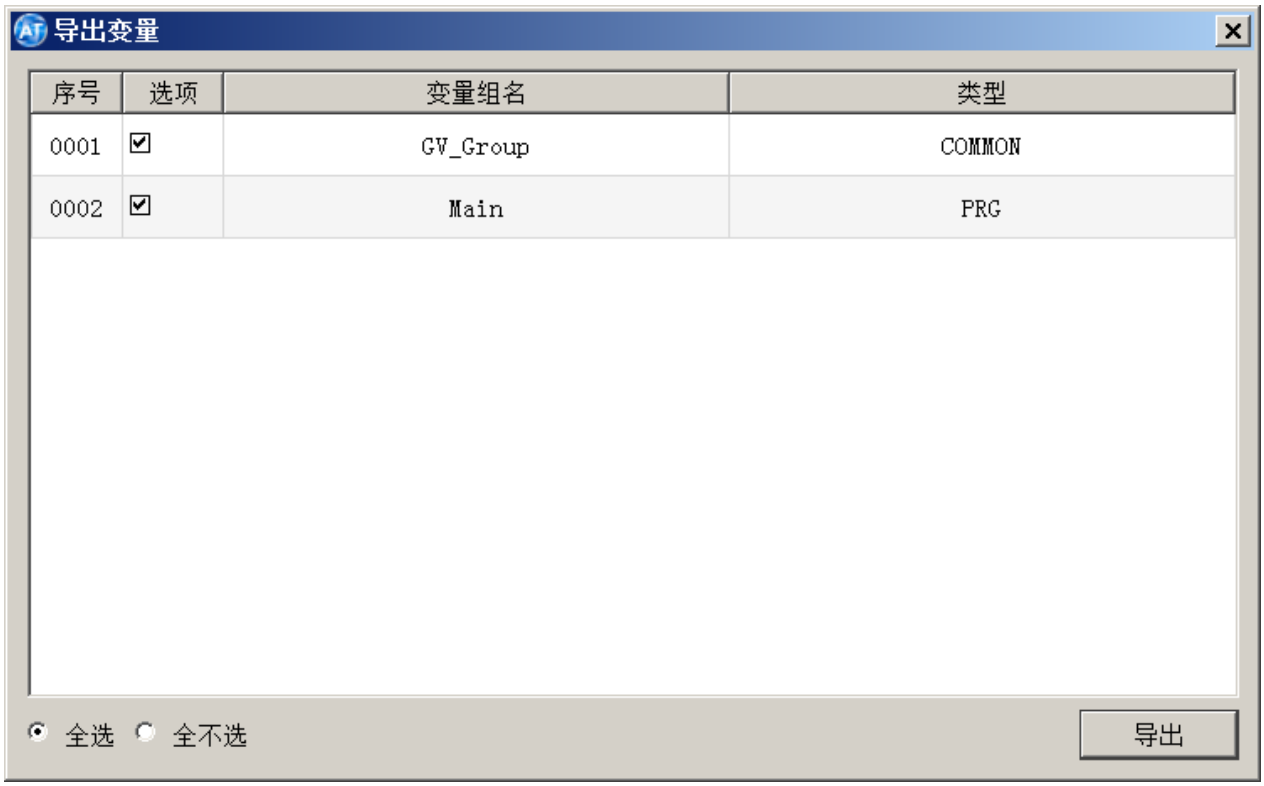
8.5.6.3 步骤

要导出变量，请按以下步骤操作：

（1） 单击【工程】菜单。

（2） 选择【导出变量】命令。

将弹出“导出变量”对话框。



导出变量对话框

（3） 勾选需要导出的变量组。

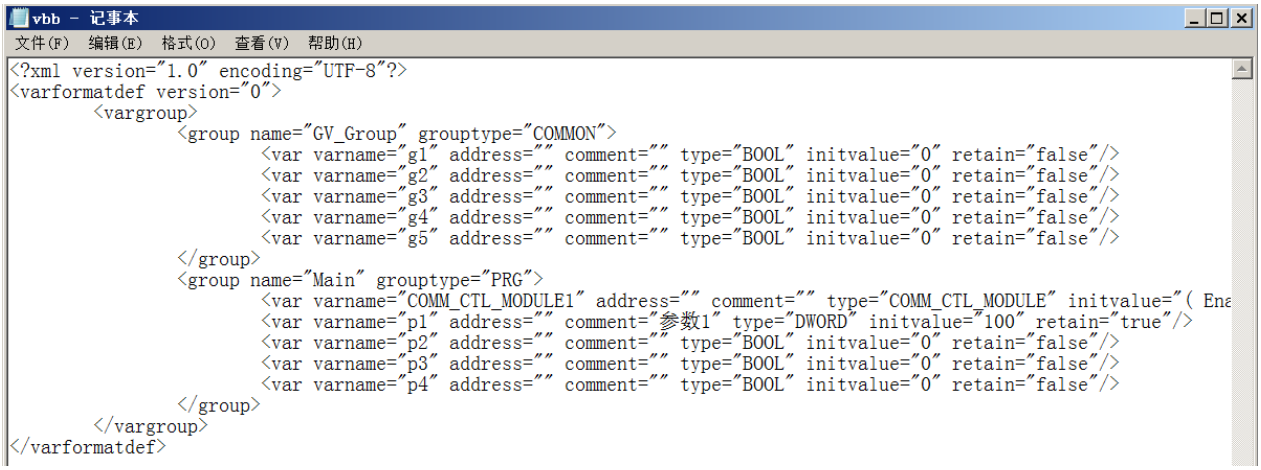
（4） 单击**导出**按钮。

将弹出“另存为”对话框。

（5） 输入文件名，并选择保存路径。

（6） 单击**保存**按钮。

导出成功后，“语法检查”窗口显示导出变量成功。



导出数据表样式

8.5.7 复制变量

可以复制一个或多个变量，粘贴到 POU 或全局变量组中。

全局变量组和 POU 之间，变量均可相互复制粘贴。

需要注意的是：复制变量不包含通道变量。

8.5.7.1 要求

有可复制变量

8.5.7.2 步骤

要复制变量，请按以下步骤操作：

（1） 在变量定义区，单击某一变量行。

（2） 按住 Shift 键，选择变量，在最后一个变量行单击。

这一段变量行被选中，呈蓝色高亮。

（3） 通过以下方式执行“复制”命令。

- 变量选中区：右键菜单—“复制”；
- 菜单栏：单击“编辑”-“复制”；
- 快捷键：**Ctrl+C**。

8.5.7.3 结果

变量被复制到剪贴板，请粘贴到对应 POU 或全局变量组。

8.5.8 复制变量名

8.5.8.1 概述

仅可以复制一个变量名，复制变量名包含 POU 局部变量、全局变量、诊断变量、通道变量。

注意：只读变量（如诊断变量）仅可复制，不可编辑。

8.5.8.2 要求

有可复制变量名

8.5.8.3 步骤

要复制变量名，请按以下步骤操作：

（1） 在变量定义区，选中变量所在行（任意列）的位置。

（2） 右键菜单选择“复制变量名”即可或双击变量名然后右键菜单选择复制。

8.5.9 粘贴变量

将复制的变量粘贴到 POU 或全局变量组中。

粘贴时，对于同名变量，系统自动分配新的变量名，POU 变量默认为 pn，全局变量默认为 gn，n 为系统自动分配的序号。

通过不同的命令，将变量粘贴到选中变量的不同位置。

- 粘贴：粘贴到当前最后一个变量之后。
- 粘贴到上方：粘贴到当前选中变量的上方。
- 粘贴到下方：粘贴到当前选中变量的下方。

8.5.9.1 要求

已复制变量

8.5.9.2 粘贴

要顺序粘贴变量，请按以下步骤操作：

（1） 通过以下方式执行“粘贴”命令。

- 变量定义区：右键菜单—“粘贴”；
- 菜单栏：“编辑”-“粘贴”；
- 快捷键：**Ctrl+V。**

8.5.9.3 粘贴在上方

要将变量粘贴到某一变量上方，请按以下步骤操作：

（1） 在变量定义区，选中某一变量。

（2） 通过以下方式执行粘贴命令。

- 变量定义区：右键菜单—“粘贴到上方”；

8.5.9.4 粘贴在下方

要将变量粘贴到某一变量下方，请按以下步骤操作：

（1） 在变量定义区，选中某一变量。

（2） 通过以下方式执行粘贴命令。

- 变量定义区：右键菜单—“粘贴到下方”；

8.5.9.5 参考信息

撤销

8.5.10 删除变量

如果需要清除一些没有使用的变量，请执行删除操作。如果变量已被关联到元素，删除变量后，会编译报错，需要手动删除元素关联的变量名。

系统默认配置的全局变量不允许删除。

8.5.10.1 要求

有需要删除的变量

8.5.10.2 步骤

要删除变量，请按以下步骤操作：

（1） 在变量定义区，单击某一变量行。

如果仅删除单个变量，请直接跳到第（3）步执行。

（2） 按住 Shift 键，选择变量，在最后一个变量行单击。

这一段变量行被选中，呈蓝色高亮。

（3） 通过以下方式执行“删除”命令。

- 变量选中区：右键菜单—“删除变量”；
- 菜单栏：“编辑”-“删除”；
- 快捷键：**Delete**。

8.5.10.3 参考信息

撤销

8.5.11 使能 SOE 功能

8.5.11.1 概述

LK631C 支持 SOE 功能，您可以通过使能通道变量的 SOE 功能，实时获取变量历史事件信息。

8.5.11.2 要求

已创建变量

8.5.11.3 步骤

使能 SOE 功能，请按以下步骤操作：

（1） 在 LK631C 模块“用户参数”中，使能 SOE 属性。

通道变量 SOE 使能

（2） 下装工程。

下装后，SOE 功能生效。

8.6 创建 LD 程序

8.6.1 LD 元素

8.6.1.1 节

节是 LD 程序的基本单元，每个 LD 语言编写的 POU 都是由“节”组成。节号缺省从 0001 开始，每个 POU 中最多能添加 999 个节。

在 LD 语言编程区的每个节的网络中，横向不超过 32 列，纵向不超过 16 行。

8.6.1.2 触点

每个触点代表一个 BOOL 型变量，可以是通道输入变量，如开关、按钮。也可以是内部变量。触点从左向右传递值。

┤ ├：常开触点，变量值为 TRUE 时，触点接通，能量流向右传递。否则，能量流在触点处断开。

8.6.1.3 线圈

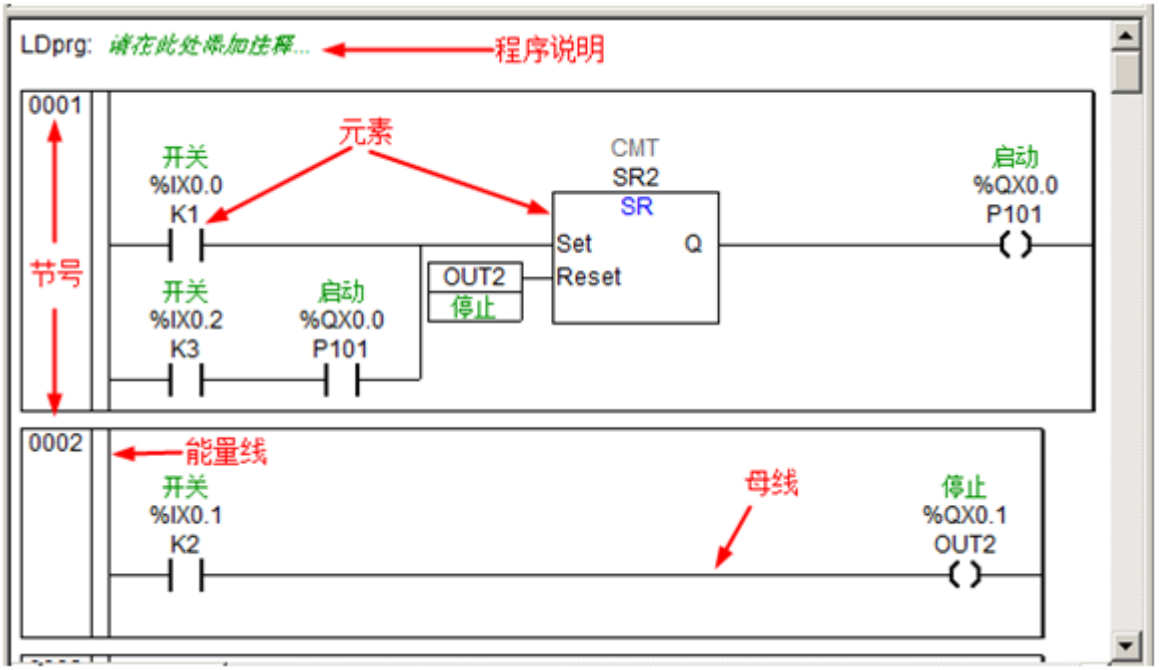
线圈代表逻辑运算的输出，用（）表示，传递从左到右的能量流，可以接通道输出变量，控制设备的启停，马达的开、停、关等。

8.6.1.4 块元件

块元件包括功能块、函数、程序这三种类型。详细说明请查看 POU 类型。

8.6.1.5 图例说明

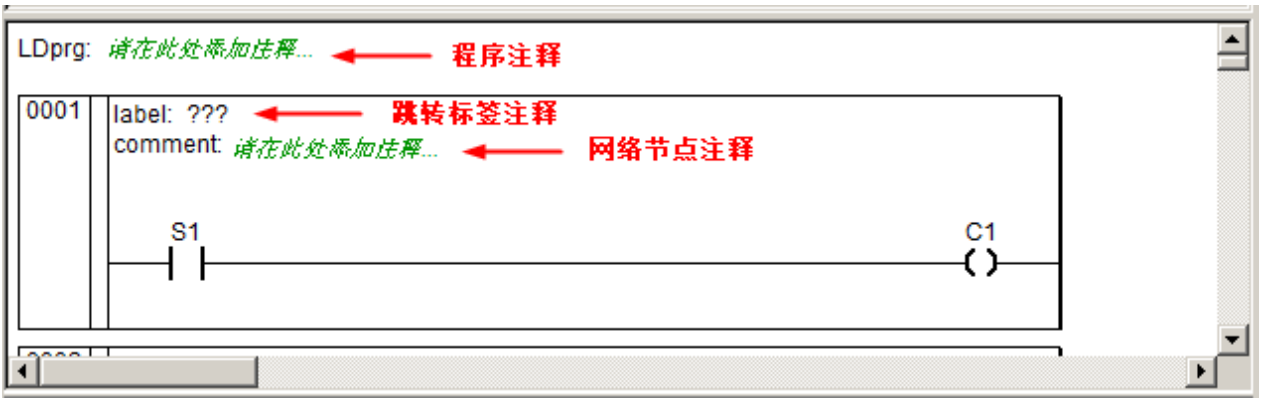
LD 程序由一系列的节组成，左右两边的能量线，以及中间母线上的元素组建了整个程序网络。



LD 语言编程元素图例说明

8.6.1.6 注释说明

- 程序注释：对 POU 程序进行说明，如“泵的启动/停止”。
- 跳转标签注释：用于标识跳转的目的程序段。默认不显示，需添加“跳转标号”才能显示。最多显示 32 个字符。
- 网络节点注释：对当前节程序进行说明。默认不显示，需要在节点右键菜单中选择“修改注释”命令才能显示注释区。



注释说明

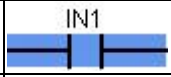
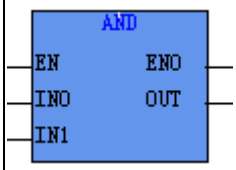
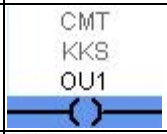
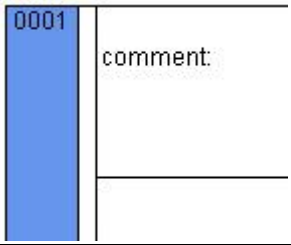
8.6.2 选中元素

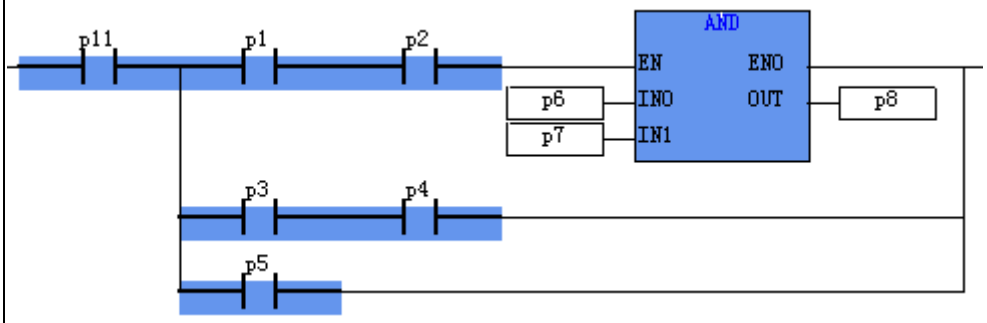
在 LD 中组态时，所有的操作，均需要选中元素才能执行。你可以选中单个元素或多个元素，执行相关操作。

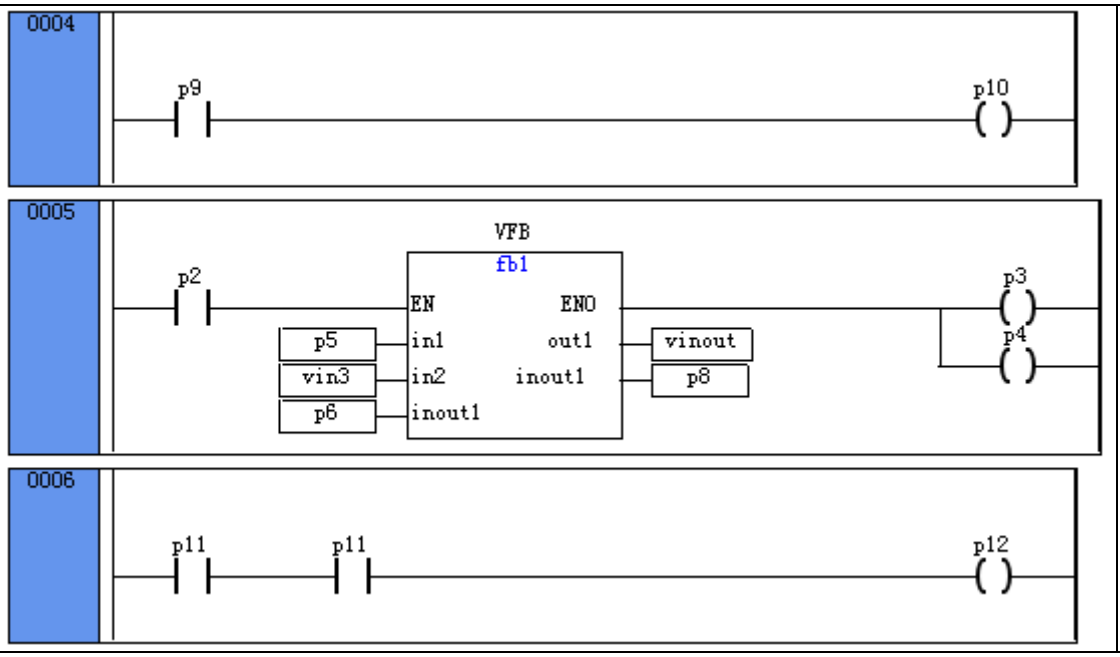
选中位置不同，可执行的操作命令也会不同。

光标选中元素时，呈现蓝色高亮状态。

选中位置及操作详见下表：

选中位置	图例	如何选中
文本域		鼠标单击元素的变量名或块名称
触点		鼠标单击触点
块元件		鼠标单击块元件
线圈		鼠标单击线圈
节		鼠标在节上单击

网络节点的注释	comment:	鼠标单击“请在此处添加注释...”
多个元素		步骤： 1 光标选中节上的一个元素作为始端元素。 2 按住 Shift 键，鼠标单击末端元素。 此时，两个元素间的所有元素均被选中。 注意： 3 这里的始端和末端元素指从上到下，从左到右顺序数，元素所在的位置。 4 选中多个元素执行操作时，要一直按下 Shift 键，直到操作命令被执行。

程序段		步骤： （1）选中程序节 （2）按住 Shift 键 （3）单击程序段尾节 这一段程序节被选中。 注意：在执行右键操作时，要一直按下 Shift 键，直到操作命令被执行。

在 LD 语言编程区，当添加块元件时，缺省添加的均为 AND 功能块，且带有使能输入端子 EN 和使能输出端子 ENO，ENO 输出值等于 EN 端的输入值。

8.6.3 插入前节

8.6.3.1 概述

在光标所在节的前面插入一个新节。

8.6.3.2 步骤

要在选中节前插入一个新节，请按以下步骤操作：

（1） 光标选中当前节。

（2） 右击节，选择“前节”命令。

此时，在当前节的前面插入一个新节，并呈蓝色高亮选中状态。

8.6.4 插入后节

8.6.4.1 概述

在光标所在节的后面插入一个新节。

8.6.4.2 步骤

要在选中节后插入一个新节，请按以下步骤操作：

（1） 光标选中当前节。

（2） 右击节，选择“后节”命令。

此时，在当前节的后面插入一个新节，并呈蓝色高亮选中状态。

8.6.5 插入触点

8.6.5.1 向前插入触点

1. 概述

在触点、块元件、线圈元素，或者并联逻辑的前面串联一个触点。

2. 要求

元素为选中状态

3. 步骤

要向前串联一个触点，请按以下步骤操作：

（1） 右击元素。

注意：如果当前选中的为并联逻辑，需要一直按住 Shift 键，在蓝色高亮区域右击鼠标。若右击其他空白区域，选中状态失效。

（2） 在右键菜单中选择“插入触点”命令。

当弹出右键菜单时，可以松开 Shift 键。

此时，触点被插入，并呈蓝色高亮选中状态。

4. 参考信息

选中元素

8.6.5.2 向后插入触点

1. 概述

在触点、块元件或并联逻辑的后面串联一个触点。

2. 要求

元素为选中状态

3. 步骤

要向后串联一个触点，请按以下步骤操作：

（1） 右击元素。

注意：如果当前选中的为并联逻辑，需要一直按住 Shift 键，在蓝色高亮区域右击鼠标。若右击其他空白区域，选中状态失效。

（2） 在右键菜单中选择“追加触点”命令或者单击工具栏“串联触点”图标  。

当弹出右键菜单时，可以松开 Shift 键。

此时，触点被插入，并呈蓝色高亮选中状态。

4. 参考信息

选中元素

8.6.5.3 在节的始端插入触点

1. 概述

在当前节的始端位置插入一个触点。

2. 要求

当前节为选中状态

3. 步骤

要在节的始端位置插入一个触点，请按以下步骤操作：

（1） 右击节。

（2） 在右键菜单中，选择“插入触点”命令。

此时，触点被插入，并呈蓝色高亮选中状态。

4. 参考信息

选中元素

8.6.5.4 并联一个触点

1. 概述

为触点、块元件或多个元素并联一个触点。

2. 要求

元素为选中状态

3. 步骤

要并联一个触点，请按以下步骤操作：

（1） 单击工具栏“并联触点”图标  。

此时，为当前选中元素并联一个触点，并呈蓝色高亮选中状态。

4. 参考信息

选中元素

8.6.5.5 串联置反触点

1. 概述

在触点、块元件的后面，或者线圈前面插入一个置反触点，也可以为并联逻辑串联一个置反触点。

2. 要求

元素为选中状态

3. 步骤

要插入一个置反触点，请按以下步骤操作：

（1） 右击元素。

注意：如果当前选中的为并联逻辑，需要一直按住 Shift 键，在蓝色高亮区域右击鼠标。若右击其他空白区域，选中状态失效。

（2） 在右键菜单中选择“置反触点”命令或单击工具栏图标  。

当弹出右键菜单时，可以松开 Shift 键。

此时，一个置反触点被插入，并呈蓝色高亮选中状态。

4. 参考信息

选中元素

8.6.5.6 并联置反触点

1. 概述

为触点、块元件或多个元素并联一个置反触点。

2. 要求

元素为选中状态

3. 步骤

要并联一个置反触点，请按以下步骤操作：

（1） 单击工具栏“并联置反触点”图标  。

此时，为当前选中元素并联一个置反触点，并呈蓝色高亮选中状态。

4. 参考信息

选中元素

8.6.6 插入线圈

8.6.6.1 插入线圈

1. 概述

在当前节母线尾端插入一个线圈，或在已有输出元素下方并联一个线圈。

2. 要求

当前节为选中状态

3. 步骤

要插入一个线圈，请按以下步骤操作：

（1） 右击节。

（2） 在右键菜单中，选择“线圈”命令或者单击工具栏图标  。

此时，在节的尾端插入一个线圈，并呈蓝色高亮选中状态。

若要继续并联线圈，请重复以上步骤。

4. 参考信息

选中元素

LD 元素

8.6.6.2 插入置位线圈

1. 概述

在当前节母线尾端插入一个置位线圈，或在已有线圈下方并联新的置位线圈。

2. 要求

当前节为选中状态

3. 步骤

要插入一个置位线圈，请按以下步骤操作：

（1） 右击节。

（2） 在右键菜单中，选择“置位线圈”命令或者单击工具栏图标  。

此时，在节的尾端插入一个置位线圈，并呈蓝色高亮选中状态。

若要继续并联置位线圈，请重复以上步骤。

4. 参考信息

选中元素

8.6.6.3 插入复位线圈

1. 概述

在当前节母线尾端插入一个复位线圈，或在已有线圈下方并联新的复位线圈。

2. 要求

当前节为选中状态

3. 步骤

要插入一个复位线圈，请按以下步骤操作：

（1） 右击节。

（2） 在右键菜单中，选择“复位线圈”命令或者单击工具栏图标  。

此时，在节的尾端插入一个复位线圈，并呈蓝色高亮选中状态。

若要继续并联复位线圈，请重复以上步骤。

4. 参考信息

选中元素

8.6.6.4 插入置反线圈

1. 概述

在当前节母线尾端插入一个置反线圈，或在已有线圈下方并联新的置反线圈。

2. 要求

当前节为选中状态

3. 步骤

要插入一个置反线圈，请按以下步骤操作：

（1） 右击节。

（2） 在右键菜单中，选择“置反线圈”命令或者单击工具栏图标  。

此时，在节的尾端插入一个置反线圈，并呈蓝色高亮选中状态。

若要继续并联置反线圈，请重复以上步骤。

4. 参考信息

选中元素

8.6.6.5 置位和复位线圈

1. 概述

通过置位/复位，将线圈变量置位为 1，设置为置位线圈和复位线圈。

置位(S)：当输入线圈的逻辑运算值为 1，则线圈变量被置为 TRUE，该变量将一直保持为 TRUE，直到被复位才变为 FALSE。

复位(R)：当输入线圈的逻辑运算值为 1，将线圈变量置为 FALSE，该变量将一直保持为 FALSE，直到被置位才变为 TRUE。

当置位和复位信号同时满足时，复位优先。

通过以下方式选择“置位/复位”命令：

- 程序区：线圈右键菜单—“置位/复位”；
- 快捷键：**Ctrl+T**；
- 工具栏：  。

2. 要求

线圈为选中状态

3. 置位线圈

要置位线圈，请按以下步骤操作：

（1） 右击线圈或复位线圈。

（2） 在右键菜单中，选择“置位/复位”命令。

线圈被设置为置位线圈。

4. 复位线圈

要复位线圈，请按以下步骤操作：

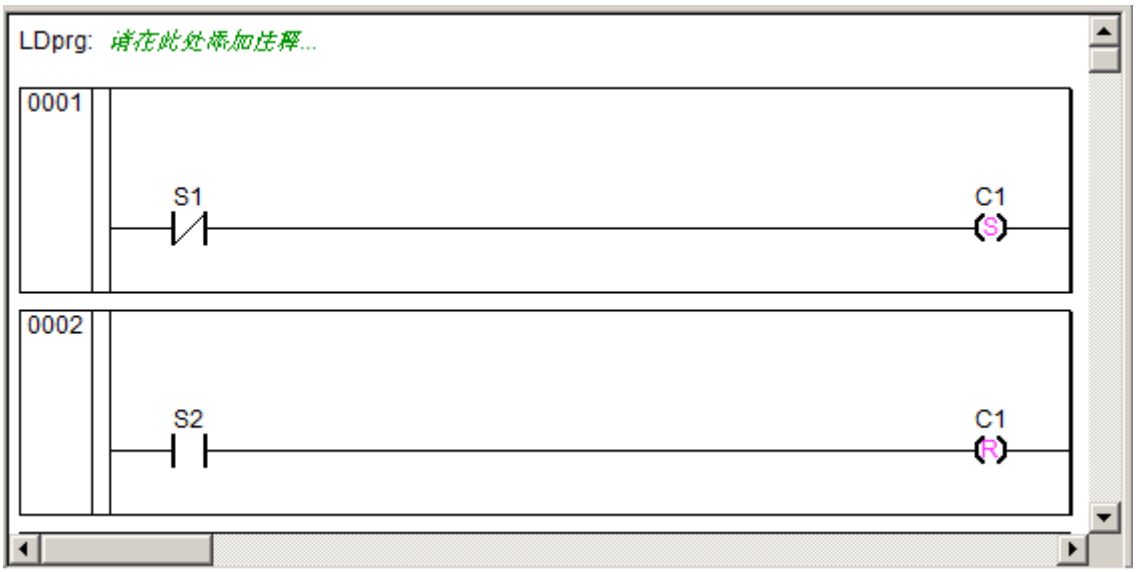
（1） 右击线圈或置位线圈。

（2） 在右键菜单中，选择“置位/复位”命令。

线圈被设置为复位线圈。

5. 示例

当 S1 为 FALSE 时，线圈变量 C1 被置位为 TRUE；只有当 S2 变为 TRUE，线圈 C1 被复位为 FALSE。如果线圈 C1 为复位状态，此时，S1 为 FALSE 时，线圈保持复位状态不变，直到 S2 变为 FLASE，线圈被再次置位为 TRUE。



置位复位操作

6. 参考信息

选中元素

8.6.7 插入块元件

8.6.7.1 串联块元件

1. 概述

为元素串联一个块元件，缺省为“AND”功能块，且该块元件具有使能输入引脚 EN，当 EN 为 TRUE 时，块元件才被执行。

选中区域不同，块元件插入的位置也会不同。

- 选中节，执行“块元件”命令，则块元件被插入到节的始端位置。
- 选中触点，执行“块元件”命令，则块元件被插入到触点之后。
- 选中线圈，执行“块元件”命令，则块元件被插入到线圈之前。

通过以下方式选择“块元件”命令：

- 程序区：右键菜单—“块元件”；
- 工具栏：  ；
- 快捷键：**Ctrl+F9**。

2. 要求

选中需要插入的区域

3. 步骤

要插入块元件，请按以下步骤操作：

（1） 右击选中区域。

（2） 在右键菜单中，选择“块元件”命令。

此时，插入一个块元件，并呈蓝色高亮选中状态。

4. 参考信息

选中元素

8.6.7.2 并联块元件

1. 概述

为线圈添加并行输出的块元件。

2. 要求

线圈为选中状态

3. 步骤

要输出一个块元件，请按以下步骤操作：

（1） 右击线圈。

（2） 在右键菜单中，选择“并联块元件”命令。

此时，线圈下方插入一个块元件，并呈蓝色高亮选中状态。

4. 参考信息

选中元素

8.6.7.3 增加输入引脚

1. 概述

块元件默认的输入引脚不够用时，可增加输入引脚数量。每次只能插入一个，输入引脚总数不超过 64 个。

增加输入引脚时，选择的区域不同，引脚插入位置也不同。

- 选中块元件，执行“多输入”命令，新引脚被插入到最后一个。
- 选中输入引脚，执行“多输入”命令，新引脚被插入到选中引脚的前面。

通过以下方式选择“多输入”命令：

- 程序区：块元件右键菜单—“高级”—“多输入（D）”；
- 程序区：输入引脚右键菜单—“多输入（D）”；
- 快捷键：**Ctrl+D**。

2. 要求

选中块元件或输入引脚

3. 步骤

要插入输入引脚，请按以下步骤操作：

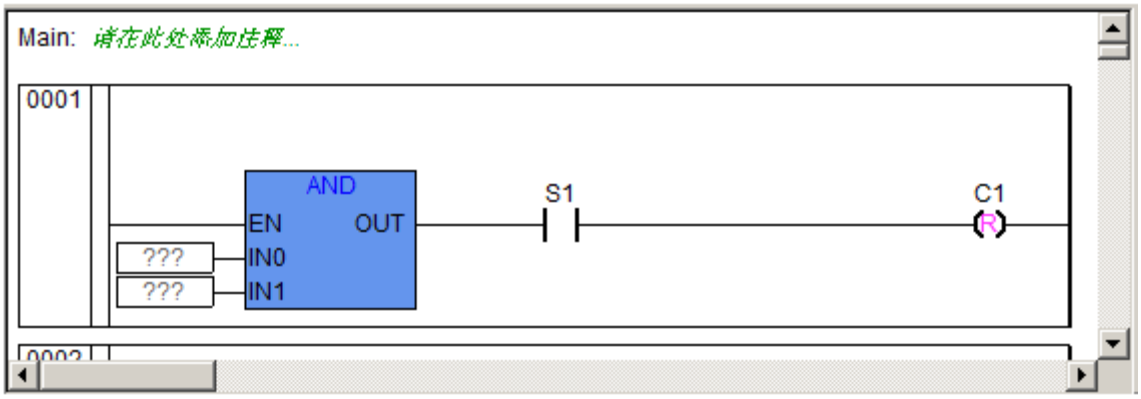
（1） 右击选中区域。

（2） 在右键菜单中，选择“多输入”命令。

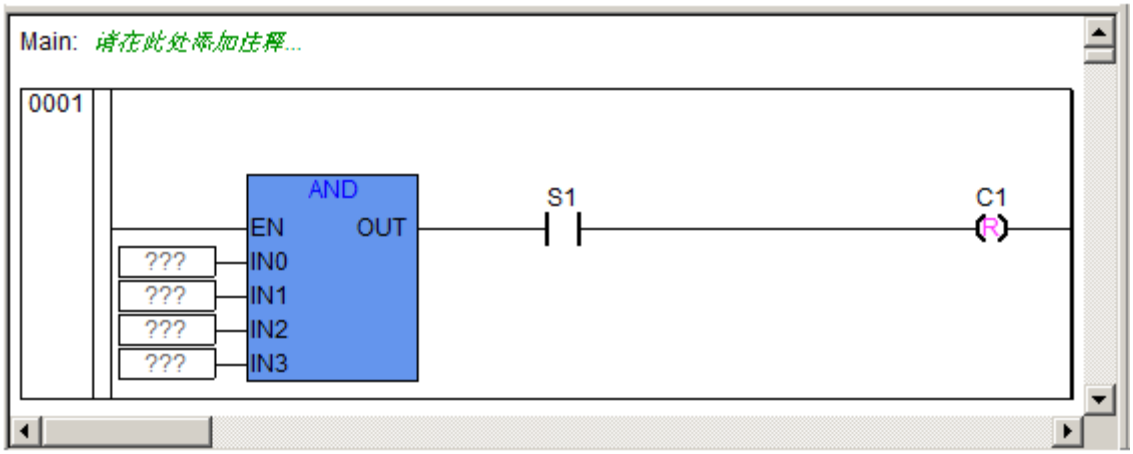
此时，插入一个新引脚，并呈蓝色高亮选中状态。

4. 示例

如图所示，对图（a）中选中的块元件，执行 2 次“多输入”命令后，增加 2 个输入引脚 IN2 和 IN3，如图（b）所示。



（a）



（b）

执行多输入命令

5. 参考信息

选中元素

8.6.7.4 显示或隐藏引脚

1. 概述

组态块元件时，可根据需要，设置使能引脚、输入和输出引脚的显示和隐藏。

功能块的全部引脚均可设置，其他块元件只能设置 EN 和 ENO 引脚。

2. 要求

已组态块元件

3. 隐藏引脚

要隐藏引脚，请按以下步骤操作：

（1） 右击块元件。

（2） 在右键菜单中，选择“高级”—“引脚（显示/隐藏）”命令。

将显示“设置引脚隐藏显示”对话框。

（3） 单击复选框，取消引脚勾选。

（4） 单击**确定**按钮。

块元件的对应引脚被隐藏，不显示。

4. 显示引脚

要隐藏引脚，请按以下步骤操作：

（1） 右击块元件。

（2） 在右键菜单中，选择“高级”—“引脚（显示/隐藏）”命令。

将显示“设置引脚隐藏显示”对话框。

（3） 勾选要显示的引脚。

（4） 单击**确定**按钮。

块元件的对应引脚被显示。

5. 参考信息

选中元素

8.6.8 添加跳转标号

8.6.8.1 概述

用户程序需要跳转到其他程序段时，通过跳转标签来指定跳转位置，程序跳转到目的程序段继续执行。

跳转标签最多可输入 32 个字符，字符要求同变量定义。

8.6.8.2 要求

目的程序段已组态

8.6.8.3 步骤

要添加跳转标号，请按以下步骤操作：

（1） 右击目的程序节。

（2） 在右键菜单中，选择“跳转标号”命令。

在程序段上方出现标识“lable”及跳转标签名称占位符“???”。

（3） 单击占位符“???”，输入跳转标签名称。

8.6.8.4 下一步

需要在跳转位置输入该跳转标签名。

8.6.8.5 参考信息

变量定义规则

跳转

8.6.9 跳转

8.6.9.1 概述

使用“跳转”命令中断程序的顺序执行，并跳转到目的程序段继续执行。目的程序段需要添加“跳转标号”进行标识，跳转位置需指定该跳转标号的名称，缺省为“???”。

跳转均在网络节的尾端，逻辑运算结果为 TRUE，则跳转到指定标号的程序段。如逻辑运算结果为 FALSE，则程序将顺序向下执行。

通过以下方式选择“跳转”命令。

- 程序区：当前节最后一个输入元素或任意输出元素的右键菜单—“跳转”；
- 工具栏：  。

8.6.9.2 要求

选中线圈

8.6.9.3 步骤

要增加跳转，请按以下步骤操作：

（1） 在中断程序节，右击线圈。

（2） 在右键菜单中，选择“跳转”命令。

线圈下方出现跳转箭头及跳转标签名称占位符“???”。

（3） 单击占位符，输入跳转标签名称。

8.6.9.4 参考信息

选中元素

添加跳转标号

返回

8.6.10 返回

8.6.10.1 概述

当调用 POU 时，可以利用“返回”命令，当条件满足后，被调用的 POU 不再继续执行，而返回调用 POU 中。

返回占位符默认为“Return”。

通过以下方式选择“返回”命令。

- 程序区：当前节最后一个输入元素或任意输出元素的右键菜单—“回”；
- 工具栏：  。

8.6.10.2 要求

选中节

8.6.10.3 步骤

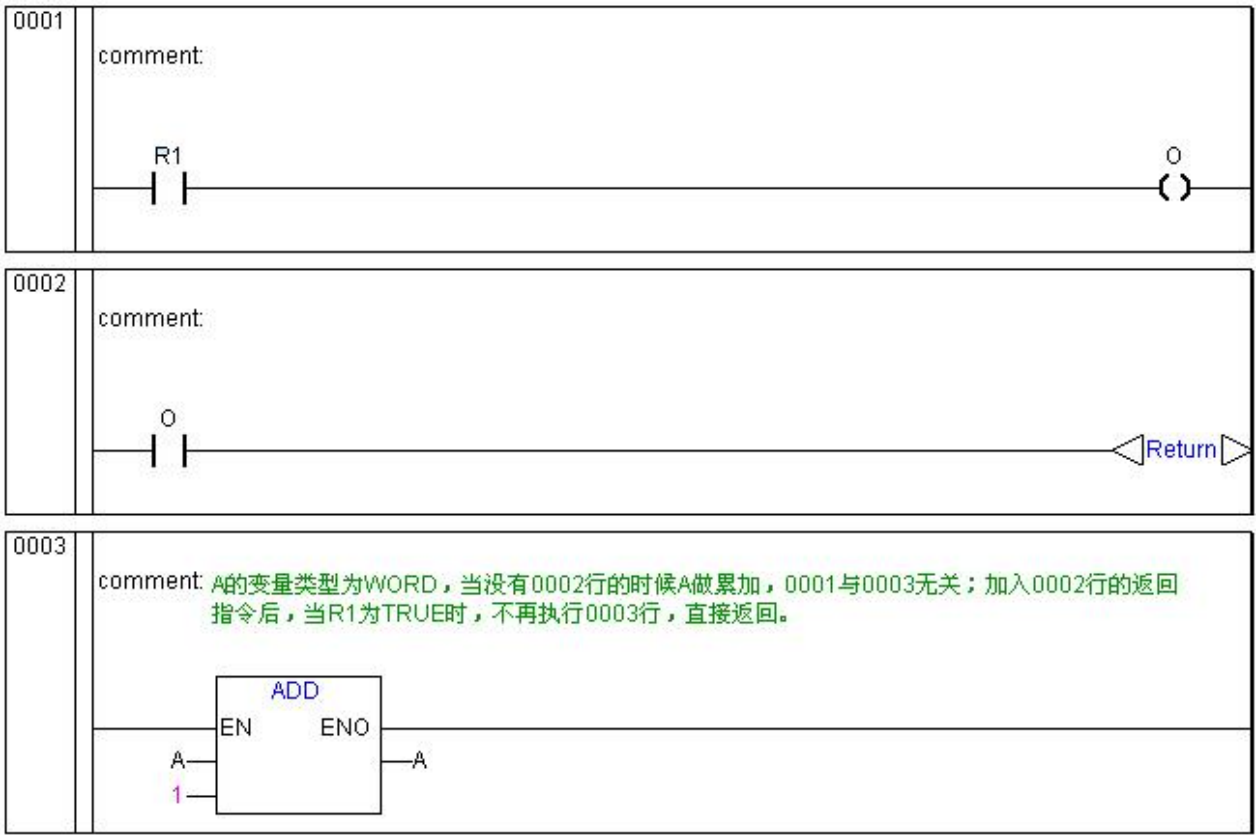
要插入返回，请按以下步骤操作：

（1） 右击节。

（2） 在右键菜单中，选择“返回”命令。

节的尾端插入返回符“Return”。

如下图 0002 节所示。当 0002 节的触点 O 闭合接通时，跳出程序，返回到调用它的 POU，不执行 0003 节及以下的程序。



返回示例

8.6.11 置反

8.6.11.1 概述

通过置反操作，可以将触点、线圈或者块元件的引脚取反。

通过以下方式选择置反命令：

- 程序区：右键菜单—“置反”；
- 工具栏：  ；
- 快捷键：**Ctrl+G**。

8.6.11.2 要求

元素为选中状态

8.6.11.3 步骤

要将元素取反，请按以下步骤操作：

（1） 右击元素。

（2） 在右键菜单中，选择“置反”命令。

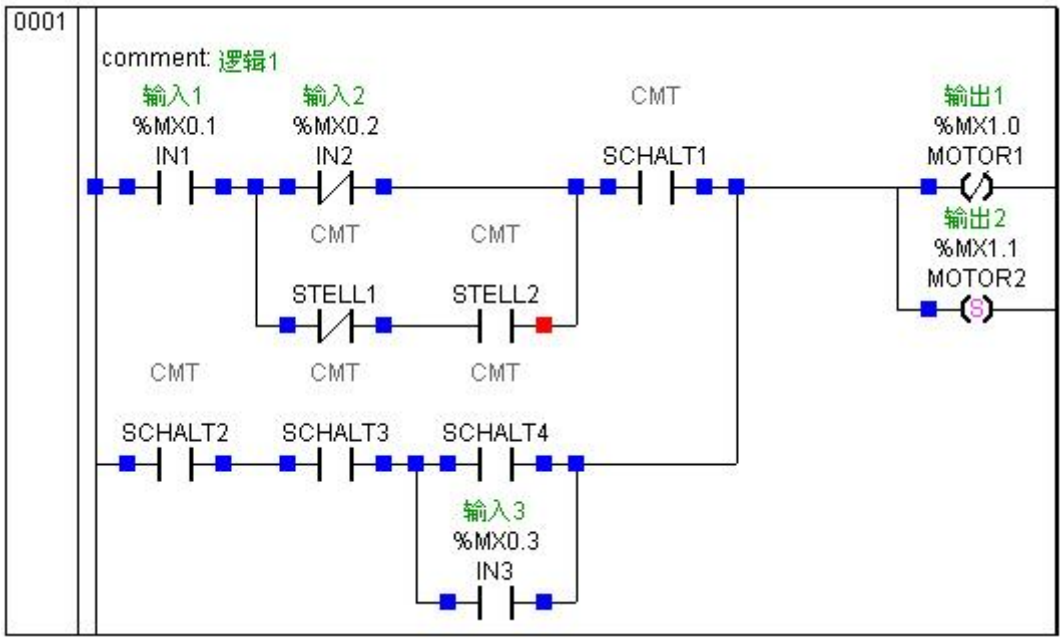
元素将被取反。

8.6.11.4 参考信息

选中元素

8.6.12 移动

选中元素，按住鼠标左键拖动，此时所有可以移动到的元件前后（线圈只有左侧的移动矩形框）处有亮蓝标识，如图所示，移动元件时，离鼠标最近的方框显示红色高亮，松开鼠标元件移动到该位置。



移动元素

8.6.13 添加程序注释

8.6.13.1 概述

使用程序注释，对每个网络节的程序内容进行说明。包括，程序节的工艺说明、逻辑功能说明、或者需要用户注意的特性。

8.6.13.2 要求

程序节已组态

8.6.13.3 步骤

要添加程序节注释，请按以下步骤操作：

（1） 右击目的程序节。

（2） 在右键菜单中，选择“修改注释”命令。

在程序节上方出现标识“comment”。

（3） 单击“请在此处添加注释...”，输入注释内容。

8.6.14 复制元素

对程序中的节、触点、线圈、块元件、多个元素或程序段可进行复制和粘贴操作。

8.6.14.1 要求

有可用的元素

8.6.14.2 步骤

要执行复制命令，请按以下步骤操作：

（1） 选中要复制的元素。

（2） 通过以下方式执行“复制”命令。

- 元素选中区：右键菜单—“复制”；
- 菜单栏：“编辑”-“复制”；
- 工具栏：  ；
- 快捷键：**Ctrl+C**。

注意：当复制程序段或多个元素时，如果通过右键菜单操作，请在选中后，一直按住 Shift 键，直到“复制”操作被执行。

8.6.14.3 结果

程序被复制到剪贴板中，请粘贴到目标位置。

8.6.14.4 参考信息

选中元素

8.6.15 剪切元素

对程序中的节、触点、线圈、块元件、多个元素或程序段可进行剪切和粘贴操作。

8.6.15.1 要求

有可用的元素

8.6.15.2 步骤

要执行剪切命令，请按以下步骤操作：

（1） 选中要剪切的元素。

（2） 通过以下方式执行“剪切”命令。

- 元素选中区：右键菜单—“剪切”；
- 菜单栏：“编辑”-“剪切”；
- 工具栏：  ；
- 快捷键：**Ctrl+X。**

注意：当剪切程序段或多个元素时，如果通过右键菜单操作，请在选中后，一直按住 Shift 键，直到“剪切”操作被执行。

8.6.15.3 结果

程序被剪切到剪贴板中，请粘贴到目标位置。

8.6.15.4 参考信息

选中元素

撤销

8.6.16 粘贴元素

8.6.16.1 概述

当执行复制和剪切操作后，需要将元素粘贴到程序某处。

- 粘贴节：将复制节粘贴到当前节之前。
- 粘贴触点和块元件：将复制元素粘贴到当前元素之前。
- 粘贴线圈：将线圈粘贴到节尾端，如当前节已有线圈，则并联该复制线圈。
- 向下或向右粘贴：对于触点和块元件，可以粘贴在当前选中元素的右边或并联到当前元素。

通过以下方式选择“粘贴”命令：

- 程序区：右键菜单—“粘贴”；
- 菜单栏：“编辑”-“粘贴”；
- 工具栏：  ；
- 快捷键：**Ctrl+V。**

8.6.16.2 要求

已复制或已剪切需要粘贴元素

8.6.16.3 粘贴

要粘贴元素，请按以下步骤操作：

（1） 右击元素。

（2） 在右键菜单中，选择“粘贴”命令。

8.6.16.4 向右粘贴

要将元素粘贴到右侧，请按以下步骤操作：

（1） 右击元素。

（2） 在右键菜单中，选择“右粘贴”命令。

8.6.16.5 向下粘贴

要并联复制元素，请按以下步骤操作：

（1） 右击元素。

（2） 在右键菜单中，选择“下粘贴”命令。

8.6.16.6 参考信息

选中元素

撤销

8.6.17 删除元素

8.6.17.1 要求

有可用的元素

8.6.17.2 步骤

要删除元素，请按以下步骤操作：

（1） 选中要删除的元素。

（2） 通过以下方式执行“删除”命令。

- 程序区：右键菜单—“删除”；
- 菜单栏：“编辑”-“删除”；
- 工具栏：  ；
- 快捷键：**Delete**。

注意：当删除程序段或多个元素时，如果通过右键菜单操作，请在选中后，一直按住 Shift 键，直到“删除”操作被执行。

8.6.17.3 参考信息

选中元素

撤销

8.7 创建 ST 程序

8.7.1 ST 元素

使用结构化文本（ST）的编程语言，可以执行多种操作，例如调用功能块、赋值和有条件地执行指令和重复任务。

ST 的编程语言由各种元素组成，具体如下：

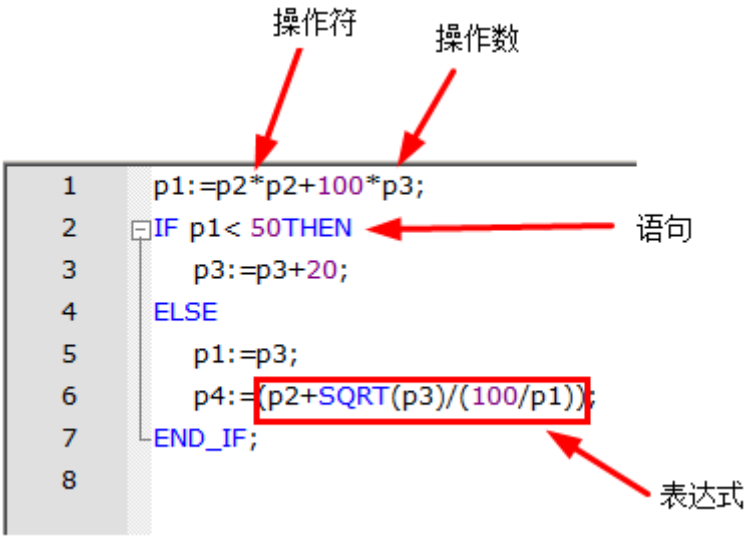
表达式：由操作数和操作符组成的结构，在执行表达式时会返回值。

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。

各元素的图例说明，如图所示。



ST 语言表示形式

ST 语言编辑时，单个 POU 的编辑内容不能超过 9999 行。

8.7.2 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：

12 （*常量*）

Var1 （*变量*）

Fun(a,b,c) （*函数调用*）

a + b （*操作符调用*）

I+（x*y*z） （*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-13。（A-B-C）*D，结果则为-16。

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。

8.7.3 ST 操作符

常见 ST 操作符及其执行的操作如下表所示。

常见 ST 操作符及其执行的操作

操作符	执行的操作	优先级
（表达式）	括号	高
函数名（参数列表）	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	低
AND	与	
XOR	异或	
OR	或	

8.7.4 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。

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));

8.7.5 ST 语句

8.7.5.1 赋值

赋值语句将一个表达式的求值结果分配给一个变量。

赋值运算符两边的数据类型必须相同，赋值语句的数据类型由左边变量的数据类型决定。

1. 语法

A := B;

其中，A 是变量，B 是表达式。表达式可以是数值、变量、运算值、函数或功能块。

2. 示例

- 将数值赋给变量
  - 语句 C := 25;

    用于将值 25 赋给变量 C。
- 将运算值赋给变量
  - 语句 X := (A+B-C)*D;

    用于将 (A+B-C)*D 的运算结果赋给变量 X。
- 将函数返回值赋给变量
  - 语句 B := C MOD A;

    用于调用 MOD（模数）函数，并将计算结果赋给变量 B。
- 将功能块输出赋值给变量

□ 语句 A := MY_TON.Q;

用于将 MY_TON 功能块（TON 功能块的实例）的 Q 输出值赋给变量 A。

8.7.5.2 IF 语句

IF 语句根据条件选择执行的程序分支。条件为布尔表达式，如果表达式值为 TRUE（真）时，条件满足，则执行 THEN 语句；如果表达式值为 FALSE（假）时，条件不满足，则根据分支执行相关语句。

1. 语法

IF<BOOL 表达式 1>THEN （判断 BOOL 表达式是否为真）

<IF 语句> （为真则执行该语句）

ELSIF<BOOL 表达式 2>THEN （可选，判断 BOOL 表达式是否为真）

<ELSIF 语句 1> （为真则执行该语句）

...

...

ELSIF<BOOL 表达式 n>THEN

<ELSIF 语句 n-1>

ELSE （可选，前面 IF 语句都不满足，执行该 ELSE 语句）

<ELSE 语句>

END_IF （IF 语句结束）

说明：

如果条件表达式 1 为真，则只执行 IF 指令，不执行其它语句。

否则，依次判断 ELSIF 的表达式值，直到其中的一个表达式为 TRUE，然后执行该表达式后的语句。

如果前面的条件表达式值都不为 TRUE，则执行 ELSE 后的语句。

条件表达式可以是 BOOL 变量、比较运算、逻辑运算。

IF 语句、ELSE 语句、ELSIF 语句的结构都遵从上面的语法结构，实际应用时，请根据用户程序进行选择。

2. 简单 IF 语句

简单 IF 语句为只有一个判断条件的 IF 结构，IF 判断条件为 TRUE，则执行语句，否则不执行。

第1步 语法

IF<BOOL 表达式>THEN （判断 BOOL 表达式是否为真）

<IF 语句>　　（为真则执行该语句）

END_IF　　（IF 语句结束）

第2步　示例

当温度值大于设定值或按下停止按钮，则停泵。

序号	变量名	直接地址	变量说明	变量类型	初始值	掉电保护
0001	Temp	%IW4	温度点	DINT	0	FALSE
0002	Temp_M		工程量温度值	REAL	0	FALSE
0003	Temp_SV		温度设定值	REAL	0	FALSE
0004	Stop_B	%IX56.0	停止按钮	BOOL	FALSE	FALSE
0005	PUMP	%QX0.1	停泵信号	BOOL	FALSE	FALSE
0006	HEX_ENGIN1			HEX_ENGIN		FALSE

```
1   HEX_ENGIN1(
2   WH:=Temp,
3   MU:=200,
4   MD:=50,
5   WU:=65535,
6   WD:=0,
7   AV=>Temp_M); (*将温度码值转换为对应的工程量值*)
8   IF Temp_M>=Temp_SV OR Stop_B =1 THEN (*当温度值大于设定值或停止按钮按下*)
9       PUMP:=0;
10  END_IF;
```

简单 IF 语句使用示例

3.　IF ELSE 语句

IF ELSE 语句是简单 IF 语句的扩展，一个 IF 语句中只能有一个 ELSE。

第1步　语法

IF<BOOL 表达式>THEN　　（判断 BOOL 表达式是否为真）

<IF 语句>　　（为真则执行该语句）

ELSE　　（前面 IF 语句都不满足，执行该 ELSE 语句）

<ELSE 语句>

END_IF　　（IF 语句结束）

第2步　示例

当罐子的低液位开关闭合时，入口阀打开，否则关闭。

序号	变量名	直接地址	变量说明	变量类型	初始值	掉电保护
0001	LSL101	%IX56.1	液位开关	BOOL	FALSE	FALSE
0002	LV101	%QX0.1	罐子入口阀	BOOL	FALSE	FALSE

```
1  IF LSL101 =1THEN   (*低液位开关闭合 *)
2       LV101:=1;  (*打开LV101阀 *)
3  ELSE
4     LV101:=0; (*关闭LV101阀 *)
5  END_IF;
```

IF-ELSE 语句示例

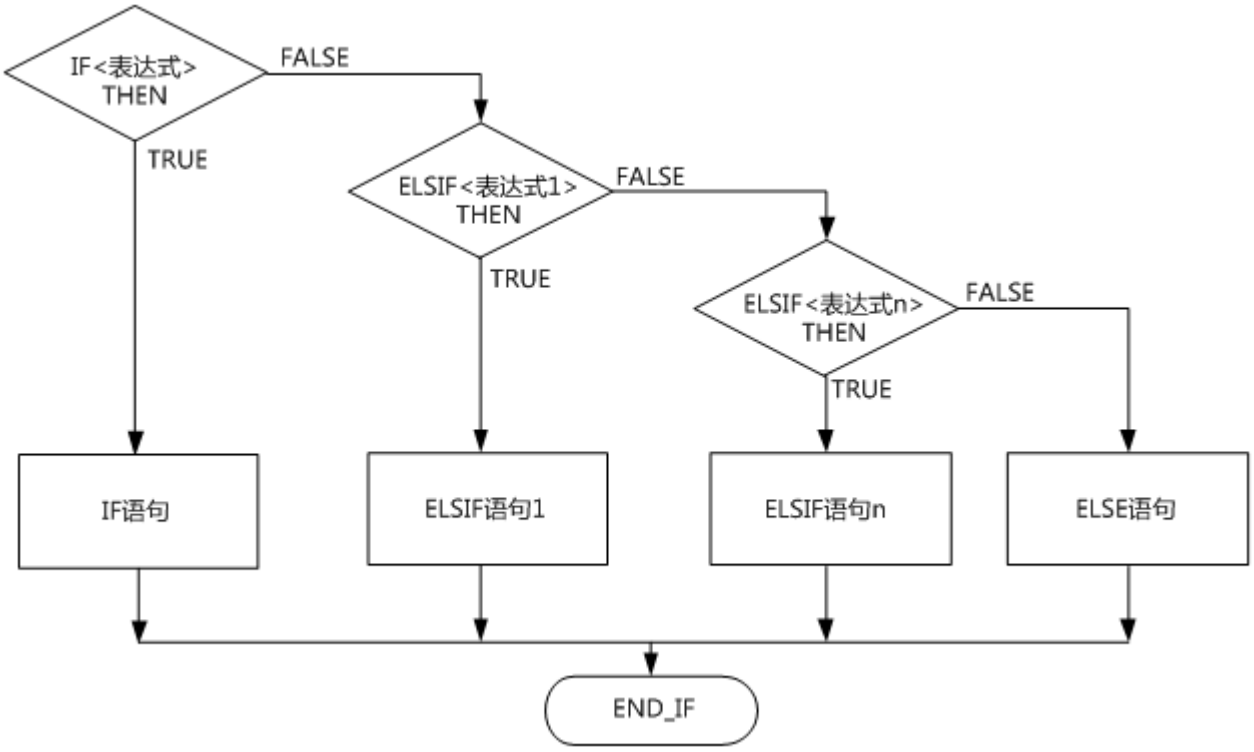
4. IF ELSIF 语句

在 IF-ELSE 语句中插入一个或多个 ELSIF 语句，可以实现多个条件的判断。

第1步 语法

参见语法。

- 仅当 IF 语句的布尔表达式值为 FALSE（假），并且 ELSIF 语句的布尔表达式值为 TRUE（真）时，才会执行语句或语句组。
- 如果 IF 语句的条件为 TRUE（真）或者 ELSIF 语句的条件为 FALSE（假），则不会执行该语句或语句组。



IF-ELSIF-ELSE 语句流程图

第2步 示例

当分子筛加热温度低于 190℃时，显示“加热温度低”；当分子筛加热温度在 190~200℃之间时，显示“加热 OK”，当分子筛加热温度超过 200℃时，显示“加热温度高”。

序号	变量名	直接地址	变量说明	变量类型	初始值	掉电保护
0001	TICA201	%IW8	温度点	DINT	0	FALSE
0002	TICA201_M		工程量温度值	REAL	0	FALSE
0003	Temp_shown		温度显示	STRING(80)	''	FALSE
0004	HEX_ENGIN1		量程转换功能块	HEX_ENGIN		FALSE

```
1   HEX_ENGIN1(
2   WH:=TICA201,
3   MU:=600,
4   MD:=0,
5   WU:=65535,
6   WD:=0,
7   AV=>TICA201_M); (*将温度码值转换为对应的工程量值*)
8   IF TICA201_M<190 THEN
9      Temp_shown:='加热温度低';
10     ELSIF TICA201_M >=190 AND TICA201_M<230 THEN
11     Temp_shown:='加热OK';
12     ELSE
13     Temp_shown:='加热温度高';
14  END_IF;
```

IF-ELSIF-ELSE 语句示例

5. IF 嵌套

IF 嵌套语句是在简单 IF 语句里面再插入一个或多个简单 IF 语句。嵌套的 IF 语句优先级低于上一层的优先级。在使用嵌套语句时，IF 与 END_IF 配套使用。

第1步 语法

IF<BOOL 表达式 1>THEN （判断 BOOL 表达式是否为真）

IF<BOOL 表达式 2>THEN （为真则执行嵌套的 IF 语句）

END_IF （嵌套的 IF 语句结束）

END_IF （上层 IF 语句结束）

第2步 示例

当主油泵 P101A 运行信号在，并且润滑油压力 PIA103 低于 0.18MPa，则启动辅助油泵 P101B。

序号	变量名	直接地址	变量说明	变量类型	初始值	掉电保护
0001	XL101	%IX56.3	运行信号	BOOL	FALSE	FALSE
0002	PIA103	%IW8	润滑油压力	DINT	0	FALSE
0003	PIA103_M		工程量值	REAL	0	FALSE
0004	P101B_S	%QX0.5	辅油泵启动信号	BOOL	FALSE	FALSE
0005	HEX_ENGIN1		工程量转换功能块	HEX_ENGIN		FALSE

```
1   HEX_ENGIN1(
2   WH:=PIA103,
3   MU:=3,
4   MD:=0,
5   WU:=65535,
6   WD:=0,
7   AV=>PIA103_M); (*将温度码值转换为对应的工程量值*)
8   IF XL101=1 THEN
9      IF PIA103_M<018 THEN
10        P101B_S:=1;
11     END_IF;
12  END_IF;
```

IF 嵌套语句示例

8.7.5.3 CASE 语句

1. 语法

CASE<条件变量> OF （判断条件变量的值）

<值 1>:<语句 1> （为值 1，则执行该语句）

<值 2>:<语句 2> （为值 2，则执行该语句）

<值 3, 值 4, 值 5>:<语句 3> （条件变量的值为值 3、值 4、值 5 时，执行该语句）

<值 6 值 10,>:<语句 4> （条件变量的值为值 6~值 10 之间的任何值时，执行该语句）

…

…

<值 n>:<语句 n>

ELSE<ELSE 语句> （条件变量的值与任何一个值都不匹配，则执行 ELSE 语句）

END_CASE （CASE 结束）

编写 CASE 语句时，需要注意以下事项：

- 条件变量必须是整型类型或枚举类型；
- CASE 子句中，值只能是整型常量或枚举常量，不能使用变量名或表达式；

- CASE 子句的值不能重叠使用，包括在一个子句中重叠使用或多个子句同时使用。

2. 规则

执行 CASE 语句时遵循下面的规则：

- 如果条件变量的值与<值 n>匹配，则执行<语句 n>；
- 如果条件变量没有任何匹配的值，则执行<ELSE 语句>；
- 如果条件变量的几个值都需要执行相同的指令，那么可以把几个值相继写在一起，并且用逗号分开。这样，共同的语句被执行；
- 如果条件变量在一定的取值范围内执行相同的语句，可以通过写入范围值的起始值、结束值，以操作符“..”分开。这样，共同的语句被执行。

3. 示例

氨罐的液位由液位计 LIA1001 测量，当液位＞90%时，画面显示“HIGH”，当液位在 25%~90%之间时，显示“OK”，当液位＜25%时，显示“LOW”，当液位低于 1%时，发出报警声。

序号	变量名	直接地址	变量说明	变量类型	初始值	掉电保护
0001	LIA1001	%IW12	现场液位计	DINT	0	FALSE
0002	LIA1001_M		工程量值	REAL	0	FALSE
0003	LIA1001_INT		整型液位值	INT	0	FALSE
0004	Level_s		液位显示信息	STRING(80)	''	FALSE
0005	L_ALARM	%QX0.6	液位低报警	BOOL	FALSE	FALSE
0006	HEX_ENGIN_LEVEL		工程量转换功能块	HEX_ENGIN		FALSE

```
1   HEX_ENGIN_LEVEL(
2   WH:=LIA1001,
3   MU:=100,
4   MD:=0,
5   WU:=65535,
6   WD:=0,
7   AV=>LIA1001_M);  (*将温度码值转换为对应的工程量值*)
8   LIA1001_INT:=REAL_TO_INT (LIA1001_M) ;(*将工程量值转换为整型数据类型*)
9   CASE LIA1001_INT OF
10      91..100:Level_s:='HIGH' ; (*液位在91%~100%，显示'HIGH'*)
11      25..90: Level_s:='OK' ; (*液位在25%~90%，显示'OK'*)
12      2..24: Level_s:='LOW' ; (*液位在2%~24%，显示'LOW'*)
13      0..1: L_ALARM:=TRUE ;(*液位低于1%，输出报警声*)
14  END_CASE;
```

CASE 示例

8.7.5.4 FOR 循环

FOR 语句用于循环次数可预先确定的程序设计，否则可使用 WHILE 或 REPEAT。

1. 语法

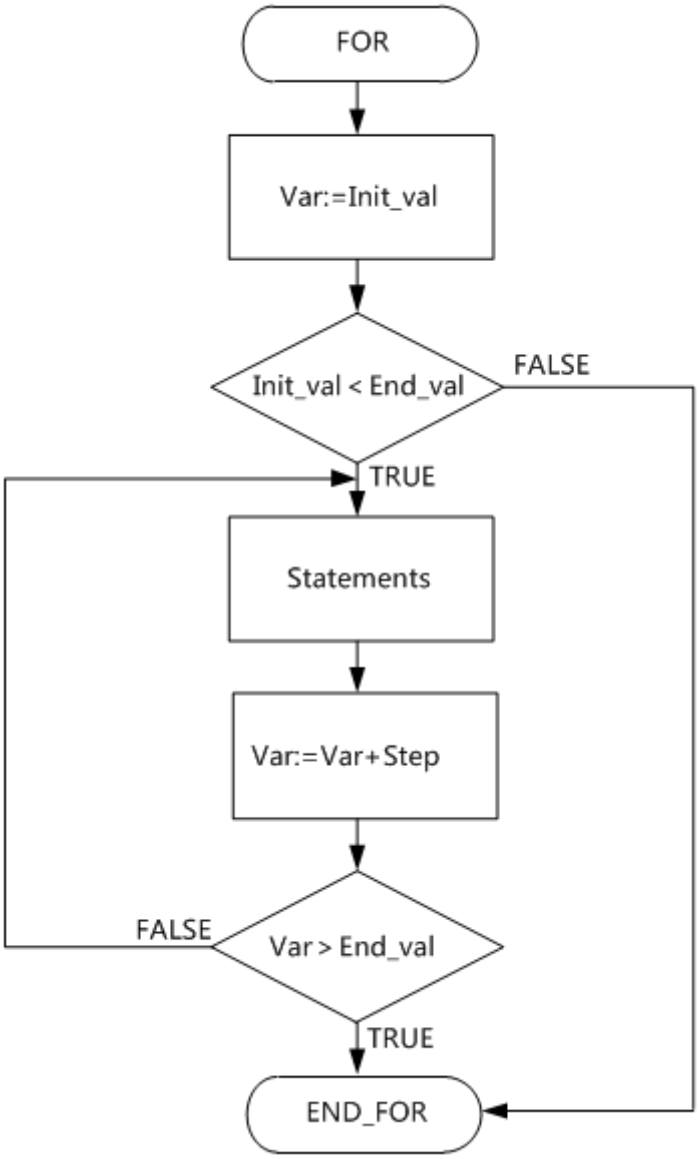
FOR<Var>:=<Init_val>TO<End_val>{BY<Step>} DO

<Statements>

END_FOR;

其中：

- <Var>：控制变量
- <Init_val>：控制变量起始值
- <End_val>：控制变量结束值
- {BY<Step>}：可选项，用于设置控制变量的递增值
- <Statements>：循环语句。
- END_FOR：FOR 循环结束标志。



FOR 循环流程图

2. 规则

- 当控制变量的值不超过结束值时，执行循环语句，每次循环之前，都会检查控制变量的值，如果超过结束值，则结束循环。
- 每次执行循环语句后，控制变量的值按照指定的递增值进行自加，递增值可以是任意的整数值，该值的设置为可选项，不设置时，该值缺省为 1。
- 控制变量、起始值和结束值必须具有相同的数据类型（DINT 或 INT），不可被重复语句改变。
- 起始值必须小于结束值，否则不会执行循环语句。

3. 示例

编写 FOR 循环示例，在单次任务下调用该 POU 程序。

序号	变量名	直接地址	变量说明	变量类型	初始值	掉电保护
0001	i			INT	0	FALSE
0002	Y			REAL	0	FALSE
0003	m			REAL	0	FALSE
0004	n			REAL	0	FALSE
0005	VAR1			REAL	0	FALSE

```
1  FOR i:=1 TO 5 DO (*当i≤5时，执行下面语句，i的循环递增值为1*)
2      Y:=Y*3;
3      m:=n/2;
4  END_FOR;
5  VAR1:=Y; (*循环结束后将Y的值赋给VAR1变量*)
```

FOR 示例

如果 Y 的初始值为 1、n 的初始值为 32，那么循环结束后，Y 的值为 243、m 的值为 1，i 的值为 6。

当在周期任务下调用该 POU 时，需要注意的是，每个 IEC 周期都会调用该 POU，所以 Y 和 m 的值循环计算。

8.7.5.5 WHILE 循环

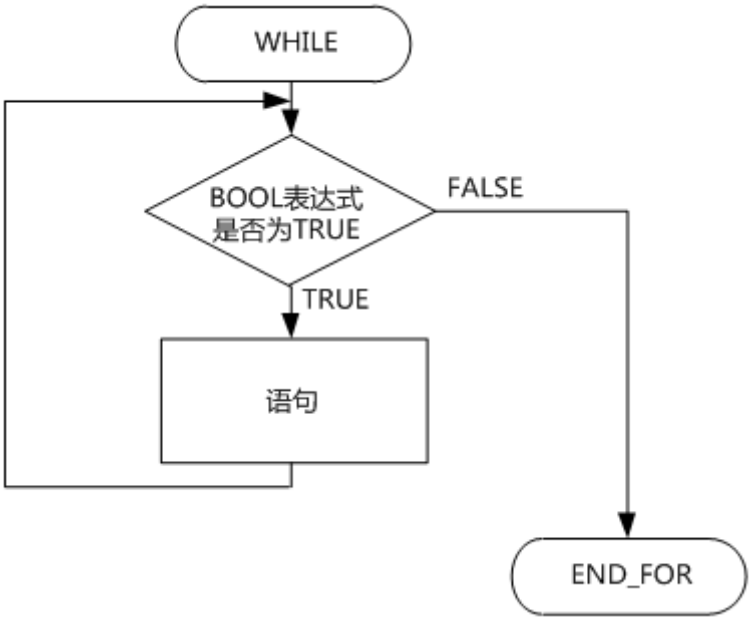
WHILE 循环用于不知道循环次数的程序设计，使用方法与 FOR 循环类似，不同的是 WHILE 循环的判断条件可以为任一表达式。

1. 语法

WHIILE<BOOL 表达式>DO （判断条件）

<语句> （循环语句）

END_WHILE; （WHILE 循环结束）



WHILE 循环流程图

2. 规则

- WHILE 语句执行时，首先检测 BOOL 表达式的值，当布尔表达式的值为 TRUE，则执行语句，当执行完语句后，再次检测表达式的值，为 TRUE，则再次执行语句，直到表达式的值不为 TRUE，结束循环。
- 如果布尔表达式初始值为 FALSE，那么语句不会被执行。
- 下列情况下不应使用 WHILE，因为它可能导致死循环，从而造成程序崩溃：
  - 布尔表达式的值始终为 TRUE；
  - WHILE 不能用于过程之间的同步；

    例如，不能用作具有外部定义的结束条件的“等待循环”。
  - WHILE 不能用在算法中；

    因为无法确保完成循环结束条件或执行 EXIT 语句。

3. 示例

序号	变量名	直接地址	变量说明	变量类型	初始值	掉电保护
0001	i			REAL	0	FALSE
0002	Value			REAL	0	FALSE

```
1  WHILE i< 4 DO
2      Value:=i*2;
3      i:=i+1;
4  END_WHILE;
```

WHILE 语句示例

假如 i 的初始值为 1，则循环结束后，Value 的值为 6，i 的值为 4。

8.7.5.6 REPEAT 循环

REPEAT 循环与 WHILE 循环不同，它在执行循环语句后，才检测结束条件。这意味着不管有没有满足结束条件，循环语句至少执行一次。

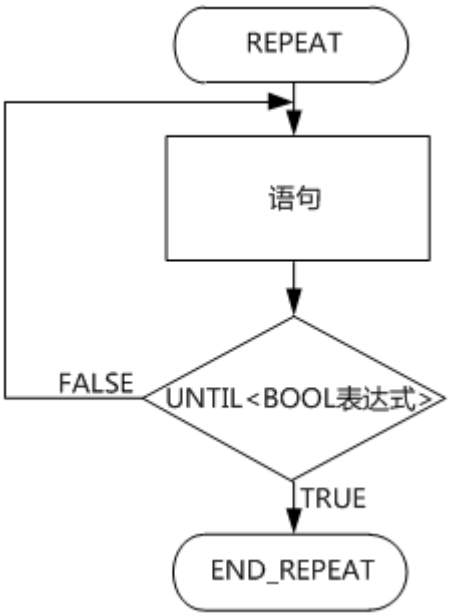
1. 语法

REPEAT （开始循环）

<语句> （循环语句）

UNTIL <BOOL 表达式> （退出条件）

END_REPEAT; （循环结束）



REPEAT 循环流程图

2. 规则

- 当 BOOL 表达式的值为 FALSE，则循环执行语句，直到 BOOL 表达式的值为 TRUE，退出 REPEAT 循环。

- 如果 BOOL 表达式的值在第一次执行时已经为 TRUE，那么语句只执行一次。
- 下列情况下不应使用 REPEAT，因为它可能导致死循环，从而造成程序崩溃：
  - BOOL 表达式的值永远都不为 TRUE；
  - REPEAT 不能用于过程之间的同步；

    例如，不能用作具有外部定义的结束条件的“等待循环”。
  - REPEAT 不能用在算法中；

    因为无法确保完成循环结束条件或执行 EXIT 语句。

3. 示例

序号	变量名	直接地址	变量说明	变量类型	初始值	掉电保护
0001	i			REAL	1	FALSE
0002	Value			REAL	0	FALSE

```
1  REPEAT
2      Value:=i*2;
3      i:=i+1;
4      UNTIL i>4
5  END_REPEAT;
```

REPEAT 示例

i 的初始值为 1，该 POU 被事件或状态型任务调用，任务运行一次后，Value 的值为 8，i 的值为 5。

需要注意的是，如果该 POU 被周期任务调用，则运行结果与单次任务调用的运行结果不同。每个 IEC 任务周期，该 POU 都被执行一次，每次执行时，Value 和 i 的值都会被执行一次，导致的运行结果是无限的循环计算 Value 和 i 的值。

8.7.5.7 EXIT 语句

1. 语法

EXIT 语句被用在 FOR 循环、WHILE 循环、REPEAT 循环语句中，无论结束条件是否满足，只要 EXIT 语句的终止条件满足时，立即退出循环语句。如果 EXIT 语句位于嵌套的循环语句内，则会退出 EXIT 所在的循环，跳转到内循环结束语句（END_FOR、END_WHILE 或 END_REPEAT）后的第一个语句继续执行。

2. 示例

如果 FLAG 的值为 0，执行语句后 SUM 将为 15。

如果 FLAG 的值为 1，执行语句后 SUM 将为 6。

序号	变量名	直接地址	变量说明	变量类型	初始值	掉电保护
0001	SUM			DINT	0	FALSE
0002	I			DINT	0	FALSE
0003	J			DINT	0	FALSE
0004	FLAG			BOOL	FALSE	FALSE

```
1  SUM:=0;
2  FOR I :=1 TO 3 DO
3     FOR J:= 1TO 2  DO
4        IF FLAG=1 THEN EXIT;
5        END_IF;
6        SUM:=SUM+J;
7     END_FOR;
8  SUM:=SUM +I;
9  END_FOR;
```

EXIT 语句示例

8.7.5.8 RETURN 语句

1. 语义

提供从函数、功能块或程序中提前退出的功能。

2. 示例

当 P1 为 TRUE 时，P2 进行自加运算；当 P1 为 FALSE 时，P2 停止自加，退出 IF 语句。

序号	变量名	直接地址	变量说明	变量类型	初始值	掉电保护
0001	p1			BOOL	FALSE	FALSE
0002	p2			DINT	10	FALSE

```
1  IF  p1=TRUE THEN
2     p2:=p2+1;
3  ELSIF  p1=FALSE THEN
4     RETURN;
5  END_IF;
```

RETURN 语句示例

8.7.5.9 函数和功能块控制语句

函数和功能块控制语句由调用功能块机制，以及在物理结束函数或功能块以前把控制返回到调用实体机制组成。

函数（FUN）求值应作为表达式求值部分被调用。

1. 调用功能块

功能块（FB）应通过语句调用，而语句由功能块实例名称和跟随其后带括号的变元表组成。

在下面的例子中，通过给两个参数 IN 和 PT 赋值来调用一个定时器，然后结果变量 Q 的值赋给变量 OUT。

结果变量被表示为：功能块名.变量名。

```
1  CMD_TMR (IN:= DM01, PT:=T#300ms);
2  OUT:=CMD_TMR.Q;
```

调用功能块示例

2. 调用函数

函数的文本调用应由函数名称及其后的变元表组成。变元应该用逗号来分隔，而该表的左右两边应该用括号界定。

函数调用规则：

- 函数输出变量的赋值应是空值或给变量赋值。
- 对 VAR_IN_OUT 变元的赋值应是变量。
- 对 VAR_INPUT 变元的赋值可以是空值、常量、变量或函数调用。在后一种情形，函数结果用作实际变元。

8.7.6 创建表达式

8.7.6.1 概述

组态表达式时，需要按从左向右的顺序添加操作数和操作符，您可以手动输入也可自动选择。手动输入时，ST 编辑器根据输入的字符自动联想关联的操作数和操作符。或者通过输入助手，选择操作数和操作符。

目前支持联想的字符范围：

- POU 名，包括程序、函数、功能块。
- 全局变量和 PRG 变量名。
- 结构体或功能块的变量成员。
- 函数仅联想到函数名，函数的变量名需要手动输入或通过输入助手选择。

8.7.6.2 规则

输入操作数时，请遵循以下规则：

- 功能块或结构体的变量成员格式：功能块或结构体变量名.成员变量名。在变量名后输入“.”自动联想成员变量名。

- 表达式中的字符为英文字符。
- 通过 Enter 键换行。

8.7.6.3 要求

已创建 ST 程序

8.7.6.4 步骤

要手动输入操作数，请按以下步骤操作：

（1） 光标当前位置输入操作数。

输入行下方显示联想的操作数，默认首个操作数被选中。

（2） 通过键盘上、下键选中操作数。

（3） 按 Enter 键。

或通过帮助管理选择操作数：

（1） 在光标处按 F2 键。

将弹出“帮助管理”窗口。

（2） 选择操作数或操作符。

（3） 单击**确认**按钮。

8.7.7 复制语句

如果需要复用 ST 语句行，请复制需要的语句或程序段，并粘贴到目标位置，可以是当前 POU 或其他 POU。

8.7.7.1 要求

有可用的 ST 程序段

8.7.7.2 步骤

要复制 ST 程序段，请按以下步骤操作：

（1） 按住鼠标左键，并拖动。

（2） 选择好程序段后，松开鼠标。

选中程序段区域蓝色高亮显示。

（3） 通过以下方式执行“复制”命令。

- 选中区域：右键菜单—“复制”；
- 菜单栏：“编辑”-“复制”；

- 快捷键：**Ctrl+C。**

8.7.7.3 结果

程序被复制到剪贴板中，请粘贴到目标位置。

8.7.8 剪切语句

如果要将当前语句剪切，并粘贴到其他位置或其他 POU 中，请执行剪切操作。

8.7.8.1 要求

有可用的 ST 程序段

8.7.8.2 步骤

要剪切 ST 程序段，请按以下步骤操作：

（1） 按住鼠标左键，并拖动。

（2） 选择好程序段后，松开鼠标。

选中程序段区域蓝色高亮显示。

（3） 通过以下方式执行“剪切”命令。

- 选中区域：右键菜单—“剪切”；
- 菜单栏：“编辑”-“剪切”；
- 快捷键：**Ctrl+X。**

8.7.8.3 结果

程序被剪切到剪贴板中，请粘贴到目标位置。

8.7.8.4 参考信息

撤销

8.7.9 粘贴语句

当执行复制和剪切操作后，需要将语句粘贴到程序某处。

8.7.9.1 要求

已复制或已剪切需要粘贴的语句

8.7.9.2 步骤

要粘贴语句行，请按以下步骤操作：

（1） 单击要粘贴语句行的位置。

（2） 通过以下方式执行“粘贴”命令。

- 程序区：右键菜单—“粘贴”；
- 菜单栏：“编辑”-“粘贴”；
- 快捷键：**Ctrl+V**。

8.7.9.3 参考信息

撤销

8.7.10 删除语句

8.7.10.1 要求

有可用的元素

8.7.10.2 步骤

要删除语句行，请按以下步骤操作：

（1） 按住鼠标左键，并拖动。

（2） 选择好程序段后，松开鼠标。

选中程序段区域蓝色高亮显示。

（3） 通过以下方式执行“删除”命令。

- 选中区域：右键菜单—“删除”；
- 菜单栏：“编辑”-“删除”；
- 快捷键：**Delete**。

8.7.10.3 参考信息

撤销

8.7.11 插入注释

8.7.11.1 概述

ST 程序中，使用注释可以对程序进行解释说明，或者用来禁用某一行或某一段程序。

注释语句行以“(*”开始，以“*）”结束，字体默认为淡绿色。

- 说明程序：（*程序的注释文本*）。
- 禁用程序：（*程序行*），被注释的程序行将不再执行，需要恢复执行时，取消注释即可。

8.7.11.2 要求

有可用的程序段

8.7.11.3 插入说明注释

要插入说明注释，请按以下步骤操作：

（1） 光标放在某一程序行或某一程序段前的空白行。

（2） 工具栏单击图标  。

当前行显示绿色注释标记（**）。

（3） 在两个*中间输入注释文本

8.7.11.4 禁用程序行

要禁用程序行，请按以下步骤操作：

（1） 光标放在某一语句行，或者鼠标选中某一段程序行。

（2） 工具栏单击图标  。

选中程序行被注释，显示绿色注释标记（**）。

8.7.11.5 启用程序行

要启用被注释的程序，请按以下步骤操作：

（1） 光标放在注释行括号内。

对于程序段，光标在注释行即可。

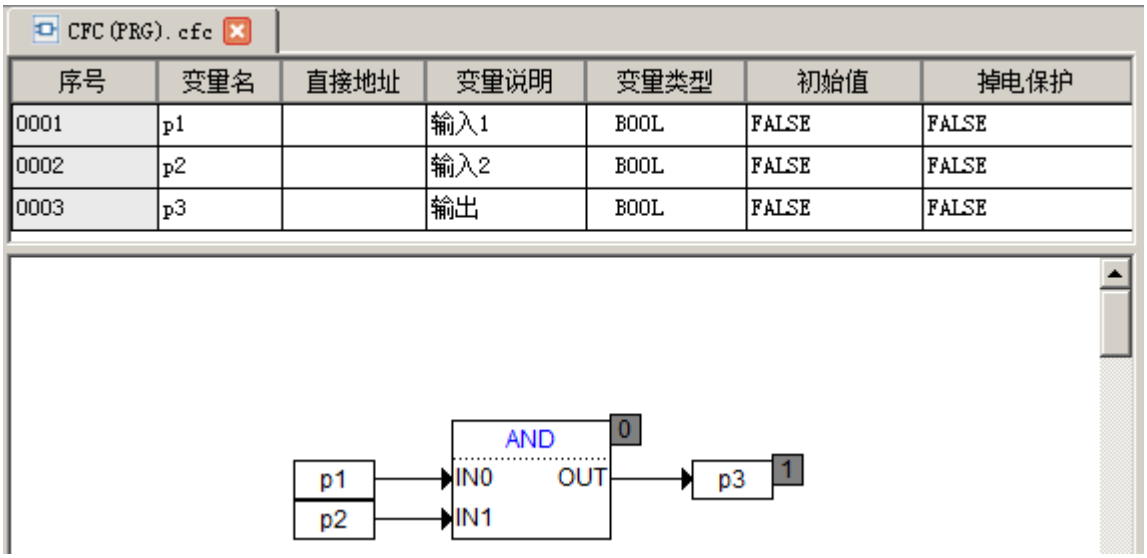
（2） 工具栏单击图标  。

禁用程序行被恢复。

8.8 创建 CFC 程序

CFC 是连续功能图（Continuous Function Chart）的简称。是一种可视化图形编辑语言，具有操作方便、显示直观的特点，适合用于连续过程控制的组态。

如下图所示，CFC 编辑器是一种图形编辑器，通过工具栏或右键菜单命令进行编辑。



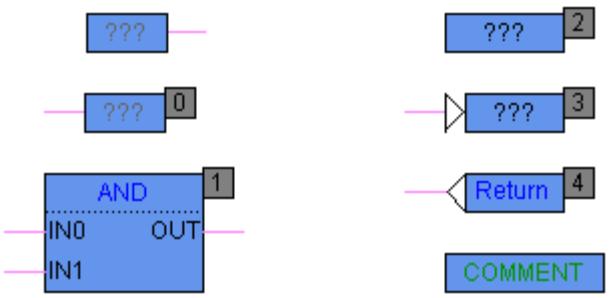
CFC 编程窗口

8.8.1 光标位置

每个文本都是光标可能的位置，选中的文本渐变为蓝色并且可以被修改。下面是光标可能位置的示例。

（1） 选中输入、输出、跳转、标签、返回、块元件和注释元件

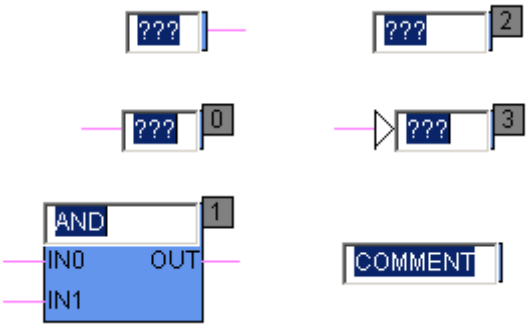
单击文本区域，即可选中，如图所示。



选中元件

（2） 选中输入、输出、跳转、标签、返回、块元件和注释元件的文本区域

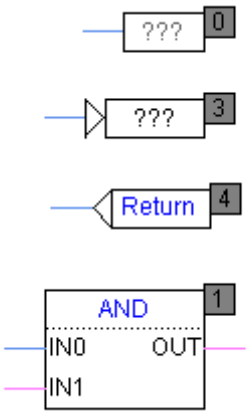
双击单击文本区域，即可选中，如图所示。



选中元件的文本区域

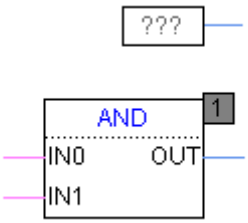
（3） 选中输出、跳转、标签块元件、返回元件的输入端

单击输入端，即可选中（呈蓝色），一次只能选中一个。如图所示。



选中输入端

（4） 选中输入元件、块元件的输出端



选中输出端

8.8.2 输入元件

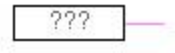
输入元件只有一个输出引脚，只能作为输入。可以为输入元件声明变量，在线时进行赋值。

通过以下方式执行：

- 程序区：右击空白区域，单击【输入元件】；
- 工具栏：  ；
- 快捷键：**Ctrl+I**。

这个命令用来插入一个输入，出现的文本“???”可以被选中并被变量或常量替代。

新插入的输入元件随鼠标移动，移动到合适的位置单击，插入成功。如图所示。



输入元件

8.8.3 输出元件

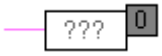
输出元件只有一个输入引脚，用来接收其他元件输出引脚的数据。输出元件关联一个执行序号，用来标示执行顺序。

通过以下方式执行：

- 程序区：右击空白区域，单击【输出元件】；
- 工具栏：  ；
- 快捷键：**Ctrl+U**。

这个命令用来插入一个输出，出现的文本“???”可以被选中并被变量替代。

新插入的输出元件随鼠标移动，移动到合适的位置单击，插入成功。如图所示。



输出元件

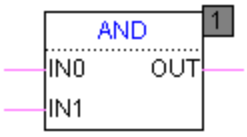
 输出元件右上方的序号 0 ，初始添加时表示添加顺序，之后可进行按数据流排序。具体内容详见调整元件执行顺序。

8.8.4 块元件

通过以下方式插入块元件。

- 程序区：右击空白区域，单击【块元件】；
- 工具栏：  ；
- 快捷键：**Ctrl+B**。

在程序编辑区插入块元件如图所示。

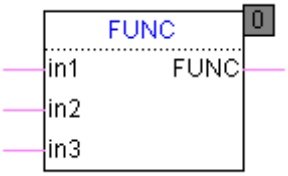


块元件

块元件函数、功能块和程序。本身包含 BOOL 类型的使能端 EN 和 ENO，且 EN 和 ENO 处于相同状态。如果块处于隐藏 EN 和 ENO 端状态，则 EN 和 ENO 状态为 TRUE；否则视输入状态而定。

（1） 函数

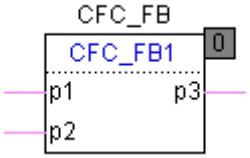
完成一定功能的函数，以块的形式参与 CFC 组态（供 CFC 用户调用）。函数块可以是库提供也可以是用户用 IEC 编程语言自定义的。函数存在若干输入变量和唯一返回值。



函数块

（2） 功能块

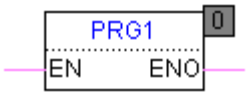
完成一定功能，以块的形式参与 CFC 组态（供 CFC 用户调用）。功能块可以是标准库提供也可以是用户用 IEC 编程语言自定义的。功能块存在若干输入和若干输出。



功能块

（3） 程序

程序块是完成一定功能的独立单元，以块的形式参与 CFC 组态（供 CFC 用户调用），能够直接被程序调用。程序块存在一个输入和一个使能输出。



程序块

8.8.4.1 多输入

使用此命令增加块元件的输入端。

选中块元件执行以下操作：

- 程序区：右击块元件，单击【高级】—【多输入】；
- 工具栏：  ；
- 快捷键：**Ctrl+D**。

执行多输入命令

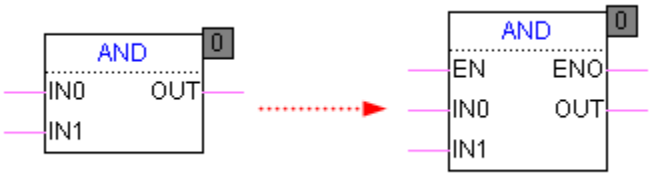
选中已添加的输入端，执行【删除】命令，即可删除引脚。

8.8.4.2 块元件使能

这个命令用来显示、隐藏选中块元件的使能输入端和使能输出端。重复激活这个命令可以使块元件在显示使能端、隐藏使能端状态之间转换。

选中块元件，通过以下方式执行【使能】命令。

- 程序区：右击块元件，单击【高级】—【使能】；
- 工具栏：  ；
- 快捷键：**Ctrl+E**。



显示使能端

CFC 程序默认情况下隐藏块元件使能端。

8.8.4.3 设置引脚属性

【设置引脚属性】选项设置块元件的输入、输出引脚的显示与隐藏。

通过以下方式设置：

- 程序区：右击块元件，单击【高级】—【设置引脚属性】。

8.8.4.4 重命名块元件

1. 概述

块元件默认以“AND”块被添加，您可以手动修改为需要的块元件名称。在输入块元件名称时，软件将自动联想所有与输入关键字匹配的块元件名称，供您选择，方便快速输入。

2. 要求

已创建一个块元件

3. 步骤

要重命名块元件，请按以下步骤操作：

（1） 双击块元件名。

光标显示在块元件名称处。

（2） 鼠标选中块元件名。

（3） 输入新名称。

自动显示所有关联的块元件。

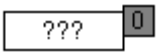
（4） 双击选中的块元件

块元件名称被修改。

8.8.5 标签元件

该元件的功能是标示程序的执行位置信息，是跳转指令跳转的目的地。标签指令不存在任何输入和输出引脚。

- 程序区：右击空白区域，单击【标签元件】；
- 工具栏：  ；
- 快捷键：**Ctrl+L**。



标签元件

文本域“???”中输入标号的标签名。该名称必须与跳转名一致。

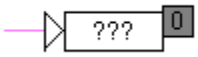
8.8.6 跳转元件

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。

通过以下方式插入跳转元件：

- 程序区：右击空白区域，单击【跳转元件】；
- 工具栏：  ；

- 快捷键：**Ctrl+J**。

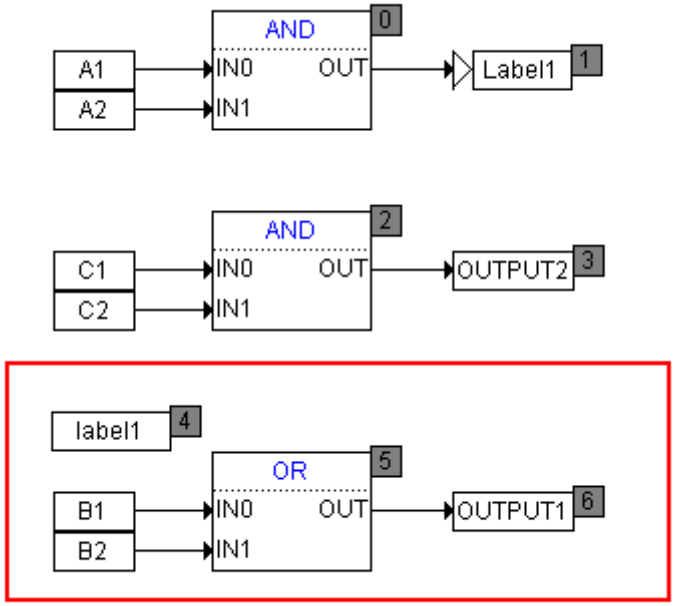


跳转元件

文本域“???”中输入要跳转到的标号名称，该名称必须与标签定义的名称一致。

跳转条件必须为 BIT 或 BOOL 类型（包括可以转换成 BOOL 的类型）的变量。

如下图所示，当跳转条件满足，CFC 程序将直接跳转到标签所在的红色标记区域，按照信号流方向执行程序。跳转标签之前的信号流 2 、 3 所在的程序行不执行。



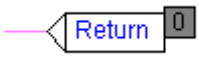
跳转与标签指令举例

8.8.7 返回元件

执行该指令后当前 POU 直接返回到调用该 POU 程序处。返回指令存在唯一输入引脚，输入值为 BOOL 类型。

通过以下方式插入返回元件：

- 程序区：右击空白区域，单击【返回元件】；
- 工具栏：  ；
- 快捷键：**Ctrl+R**。



返回元件

8.8.8 注释元件

注释元件不参与程序的执行，其作用就是在 CFC 程序界面插入表示说明的文本信息。相当于其他语言中的注释信息，该指令不存在任何输入输出信息。

通过以下方式插入注释元件：

- 程序区：右击空白区域，单击【注释元件】；
- 工具栏：  ；
- 快捷键：**Ctrl+K**。

注释元件

8.8.9 删除元件

该命令用于删除一个或多个 CFC 元件。

通过以下操作删除：

- 程序区：右击块元件，单击【删除】；
- 快捷键：**Delete**。

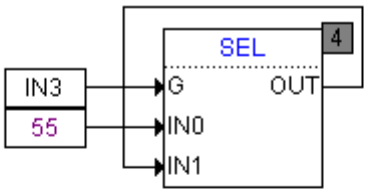
删除元件时，连同连线一起被删除。

8.8.10 增加连线

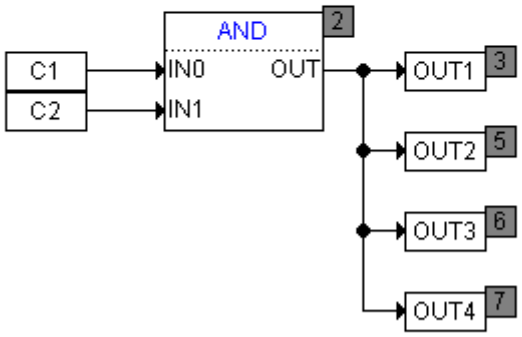
通过连线将块元件的引脚连接起来，组态用户逻辑。

拖动鼠标，连接元件的输入和输出，会自动创建连线。当元件移动的时候，连线线也会自动调整。

一个元件的输入引脚只能连一个变量（本元件的输出引脚或其他元件的输出引脚）。而一个元件的输出引脚可以连多个变量（本元件的输入变量或其他元件的输入变量），如图所示。



(a)

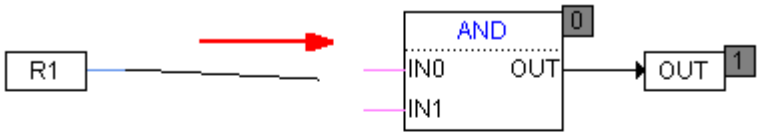


(b)

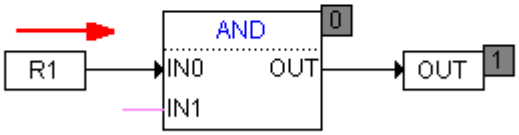
引脚信号连接方式

有两种方式连接 R1、AND 和 OUT 元件之间的连线，如图所示。

- 单击 R1 的输出引脚，如图（a）所示，拖拽到 AND 的输入引脚上，释放左键，如图（a）所示。
- 拖动 R1，将其输出引脚线与 AND 的输入引脚线重合，释放左键，连线完成，如图（b）所示。



(a)



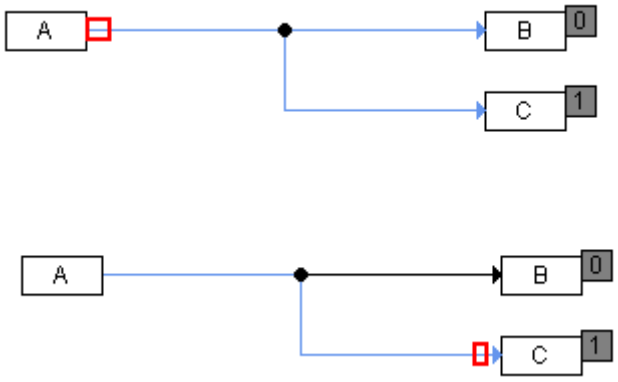
(b)

连线

8.8.11 删除连线

若要删除连线，可以选中引脚然后执行右键【删除】或键盘 **Delete**,连接即可被删除。

删除连线两种位置如图所示。选中 A 的输出引脚执行【删除】命令，将删除 A 输出引脚所连的全部连线。选中 C 出入引脚，执行【删除】命令，只删除 C 输入引脚与 A 输出引脚间的连线，A 与 B 之间的连线不受影响。

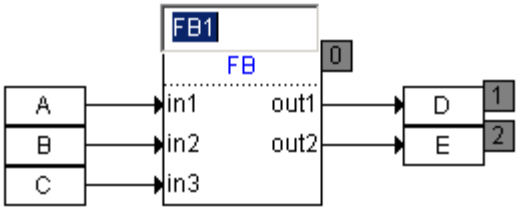


删除连线的选中位置

8.8.12 修改元件文本

每一个 CFC 元件文本都占据一定区域，鼠标左键双击文本区域就会弹出编辑元件文本的编辑框。通过编辑框用户可以修改元件的文本。

举例，如下图所示的 CFC 元件，现在要修改功能块文本 FB1：鼠标左键双击功能块文本 FB1，在原来文本处弹出编辑框如下图所示，用户即可在编辑框内输入新的文本。



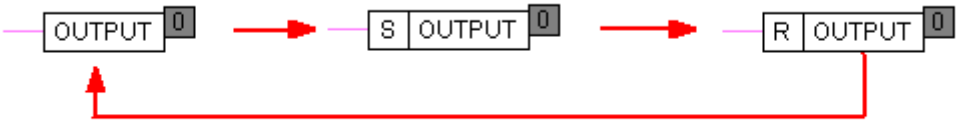
文本编辑

8.8.13 输出元件的置位复位

CFC 输出元件具有置位复位功能。被置位的输出元件可被复位，复位的输出元件能被还原。

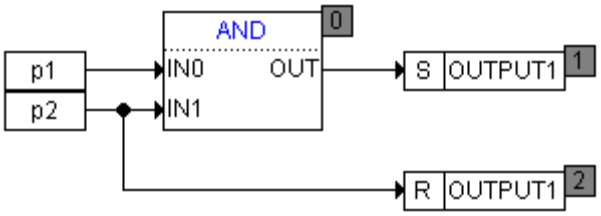
选中输出元件，连续执行以下操作，输出元件在置位、复位、还原状态之间转换。如图所示。

- 程序区：右击输出元件，单击【高级】—【置位/复位】；
- 工具栏：  ；
- 快捷键：**Ctrl+T**。



执行置位复位命令

如图所示，对于置位输出，AND 块元件的 OUT 为 TRUE，则 OUTPUT1 被置位为 TRUE，并保持置位状态不变，即使 OUT 变为 FALSE。只有当输入元件 p2 变为 FALSE，OUTPUT1 才被复位为 FALSE，并且保持复位状态不变，直到置位条件满足，再次被置位。



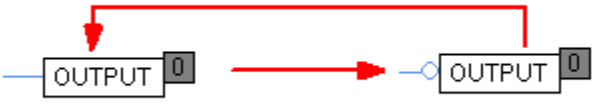
置位复位示例

8.8.14 引脚的置反和还原

置反操作是对 BOOL 值进行取反操作。选中引脚，执行【置反】命令，则被选中的引脚被置反。再次选中被取反的引脚，执行【置反】命令后，选中的引脚被还原。如图所示。

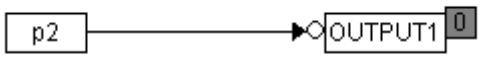
通过以下方式执行：

- 程序区：右击元件的引脚，单击【置反】；
- 工具栏：  ；
- 快捷键：**Ctrl+G**。



执行置反操作

如图所示，若 p2 为 TRUE，则经过取反操作后，OUTPUT1 的值为 FALSE。若 p2 为 FALSE，则 OUTPUT1 的值为 TRUE。

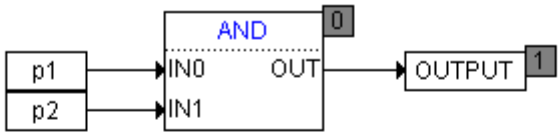


置反命令示例

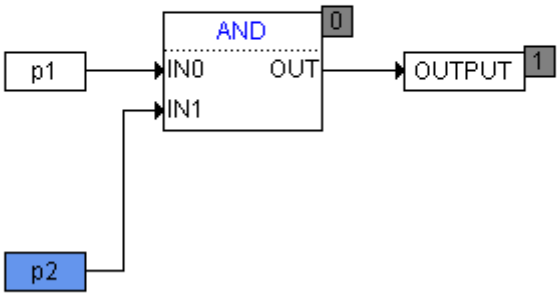
8.8.15 移动元件

CFC 中可在编辑区自由拖动元件。选中元件，拖动鼠标左键可实现元件的移动。如果选项中开启默认**碰撞检测**，即移动到的新位置已存在元件，则移动失败，恢复到移动前状态。

选中元件 p2，拖动到一个新的位置，实现了元件的移动，如下图所示。



(a)



(b)

移动元件

8.8.16 剪切、粘贴和复制

复制：选中要复制的元件，执行复制命令或键盘指令 **Ctrl+C**，即可把选中元件数据放到剪贴板上。如果选择多个元件，并且元件间存在连线，则保持原来连线。

剪切：选中要剪切的元件，执行剪切命令即可把选中元件数据放到剪贴板上，同时选中的元件被删除。

粘贴：将已复制或剪切在剪贴板上元件数据以元件形式添加到 CFC 编辑界面中。

通过以下方式执行复制/剪切操作。

- 程序区：右击元件，单击【复制】/【剪切】。

执行以上操作后，进行粘贴操作。

- 程序区：右击空白区域，单击【粘贴】。

8.8.17 撤销和恢复

撤消是取消上一次命令的执行，恢复到上一次执行操作命令之前。

恢复命令将操作恢复到执行撤销命令前。

所有 CFC 程序组态操作命令分为两类：不可以撤消恢复操作和可以撤消恢复操作。这里不包括对 POU 本身的操作比如 POU 重命名、禁止调用和允许调用等等。

不可以撤消恢复命令包括：选中元件、选中输入端、选中输出端、复制元件、查找。

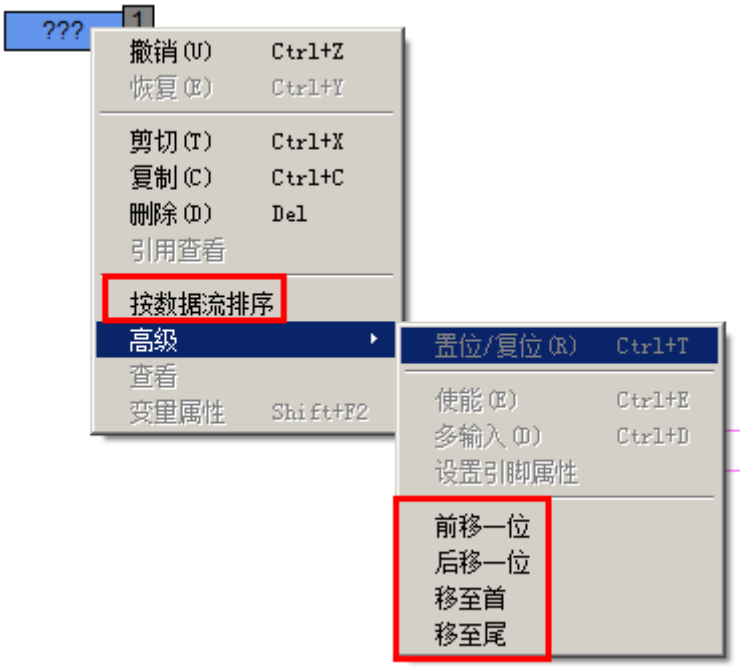
8.8.18 调整元件执行顺序

CFC 元件除了输入元件和注释都存在一个标示程序执行顺序的数字，显示在元件的右上角。

CFC 程序是按照各个元件的执行序号，从 0 号开始，依次完成赋值、计算、跳转、返回等程序流程。执行顺序编号默认是按照元件输入顺序排序的。元件执行序号在组态过程中始终保持从小到大并按自然数连续排列。

您可以选择按数据流顺序排序或者指定元件的执行顺序。

在元件右键菜单中选择排序方式，如下图所示。

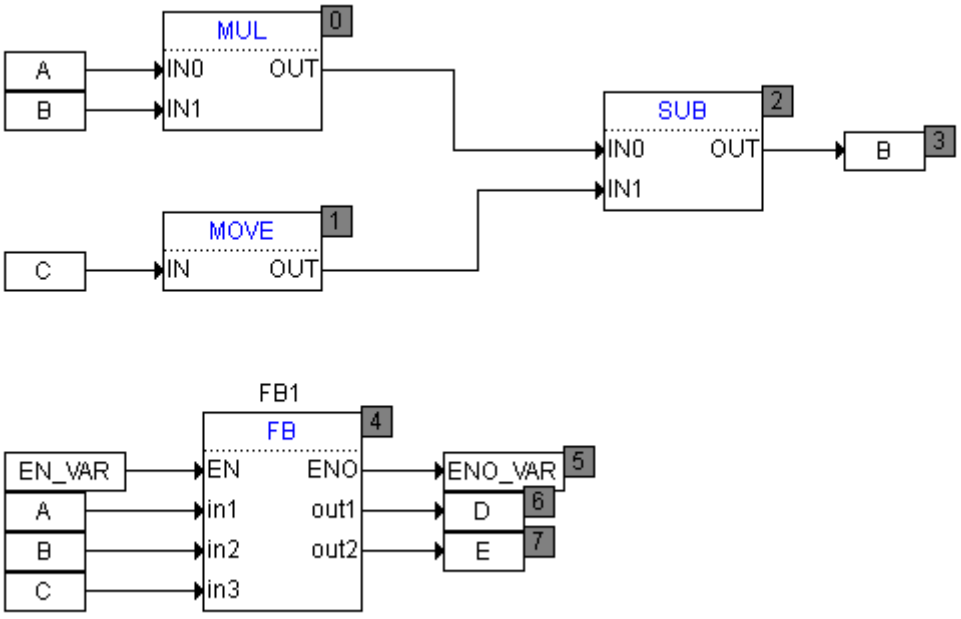


执行顺序选择菜单

（1） 按数据流排序

按数据流排序原则：以输入为基础，执行编号按照逆波兰式顺序排序。

举例：如下图所示，按照数据流排序的 CFC 程序。



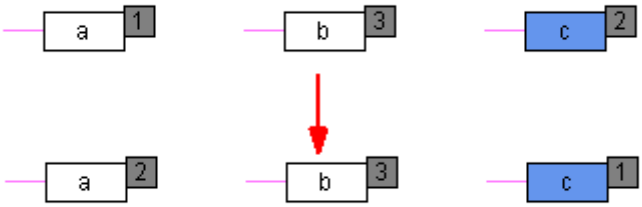
按数据流排序

（2） 前移一位

选中带有执行顺序的元件，执行该命令，选中元件与执行编号在其前面一位的元件进行执行编号交换。

选中最小编号的元件执行该命令后执行顺序保持不变。

如图所示，对输出元件 c 执行【前移一位】命令，其执行编号与其前一位执行编号互换。



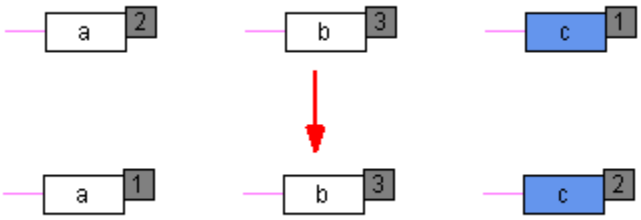
执行前移一位命令

（3） 后移一位

选中带有执行顺序的元件，执行该命令，选中元件与执行编号在其前后一位的元件进行执行编号交换。

选中最大编号的元件执行该命令后执行顺序保持不变。

如图所示，对输出元件 c 执行【后移一位】命令，其执行编号与其后一位执行编号互换。

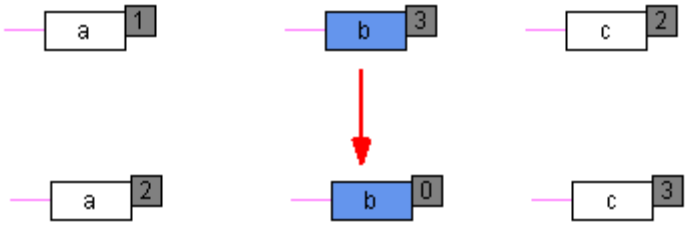


执行后移一位命令

（4） 移至首

将选中元件编号设置为 0，原元件执行编号前面的带执行编号元件的编号分别加 1。选中元件编号为 0 则执行该命令后无影响。

如图所示，对输出元件 b 执行【移至首】，其执行编号变为最小编号 0，其原执行编号前的编号 2、 1分别加 1。

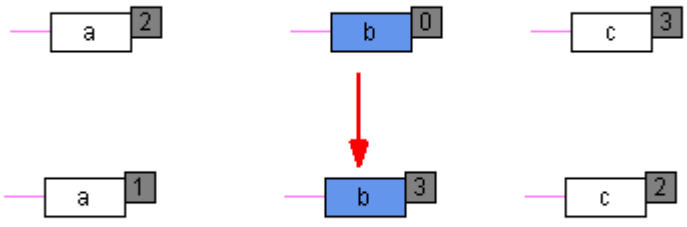


执行移至首命令

（5） 移至尾

将选中元件编号设置为现有编号范围中的最大值，原来元件执行编号后面的带执行编号元件的编号分别减 1。选中元件编号为最大则执行该命令后无影响。

如图所示，对输出元件 b 执行【移至尾】命令，其执行编号变为最大执行编号，其原执行编号后的编号 2、3分别减 1。



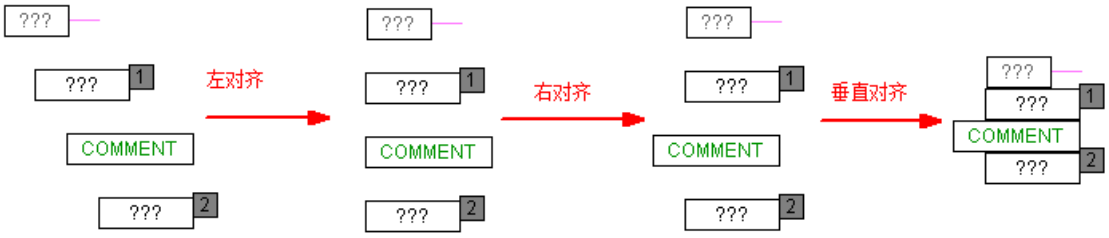
执行移至尾命令

8.8.19 元件对齐

对块元件进行水平方向上的左右对齐，以及垂直方向上的间距最小距离排布。

选中两个以上块元件，执行对齐操作。

- 程序区：右击选中的元件，单击【元件对齐】—【水平左对齐】/【水平右对齐】/【垂直对齐】。



对齐示例

水平左/右对齐：以选中元件中最左/右边位置的元件为基准，选中的其他元件在水平方向上与基准对齐。

垂直对齐：以选中元件中最上边位置的元件为基准，选中的其他元件在垂直方向上相互间隔最小距离排布，水平方向不做移动。

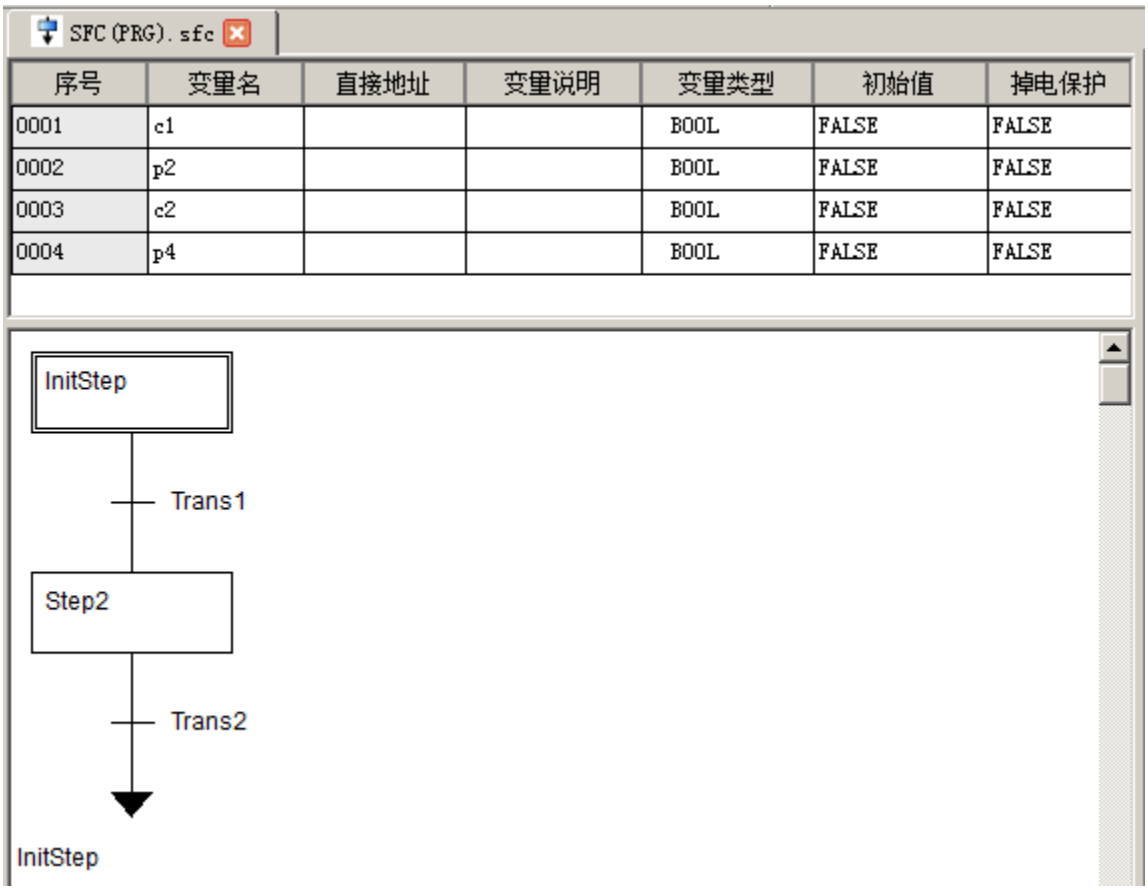
8.9 创建 SFC 程序

SFC 是顺序功能图（Sequential Function Chart）的简称，是 IEC61131-3 标准中的一种编程语言，用于建立和修改顺序控制。

SFC 能使单任务到复杂任务用一种简单的配置方法完成。这个结构用独立的元素表示用户的子程序，类似网络节点元素。

子任务是关联到转换和步的，通过程序的形式体现的。这些程序可以使用 CFC、LD、ST 语言编辑。转换表示激活后继步的使能条件。步周期地处理直到下一个转换条件满足。

转换通过线、分支和控制过程的独立元素连接起来。选择分支和并行分支有区别。如果是选择分支，一次只能是一个过程在运行；反之，过程同时有分支时，几个步并行处理。



序号	变量名	直接地址	变量说明	变量类型	初始值	掉电保护
0001	c1			BOOL	FALSE	FALSE
0002	p2			BOOL	FALSE	FALSE
0003	c2			BOOL	FALSE	FALSE
0004	p4			BOOL	FALSE	FALSE

SFC 编程窗口

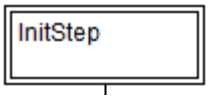
8.9.1 SFC 步元素

一个步代表一种状态，一个步要么是活动的要么是不活动的。任何时刻，程序组织单元（POU）的状态由一些活动步的设置及其内部变量的值来定义。

1. 初始步

每个 SFC 编辑的 POU 以初始步开始。这个步是进入 POU 的入口。

只有初始步总是在 SFC 的 POU 中存在的。

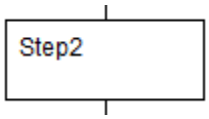


初始步

2. 步

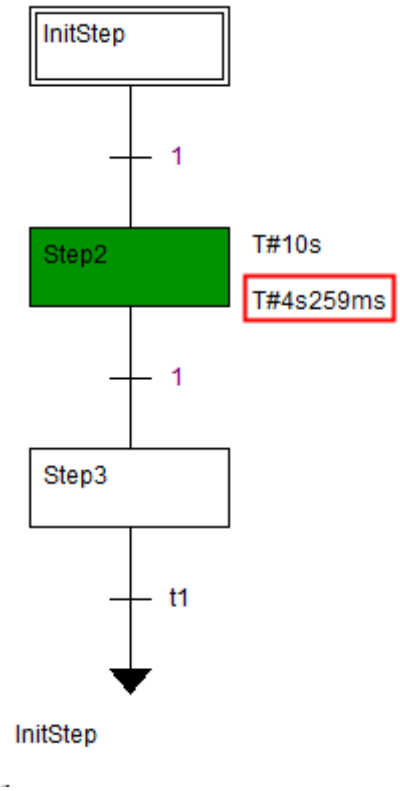
步说明了在这个工艺步控制了什么，步的动作描述了配置的程序，这些程序可以用 ST、LD 或 CFC 编辑。

插入一个步，步名需要在 SFC 的 POU 中明确。一个步最多可以关联 9 个动作。



步元素

- StepName.t(TIME)：步运行时间，当一个步被解除激活时，步所经历的时间值应保持在此步被解除激活时它所具有的值，当步被激活时，步所经历的时间值应复位为 t#0s，重新开始计时。如图所示，T#10s 为 Step2 步设置的最小时间，T#4s259ms 为 Step2.t 的值。当设置了步最小时间时，该步被激活，StepName.t 显示在如图所示位置。



StepName.t 步运行时间在线值

- StepName.x(BOOL)：步标志，是一个 BOOL 型元素，标示步当前的活动状态，1：步活动；0：步停用。如上图所示，当 Step2 之前的转换条件变为 True 时，Step2 被激活，此时 StepName.x 为 1，该步为活动状态。

步名称、步运行时间、步标志仅在当前 POU 有效，不支持其他 POU 引用。

8.9.2 SFC 动作元素

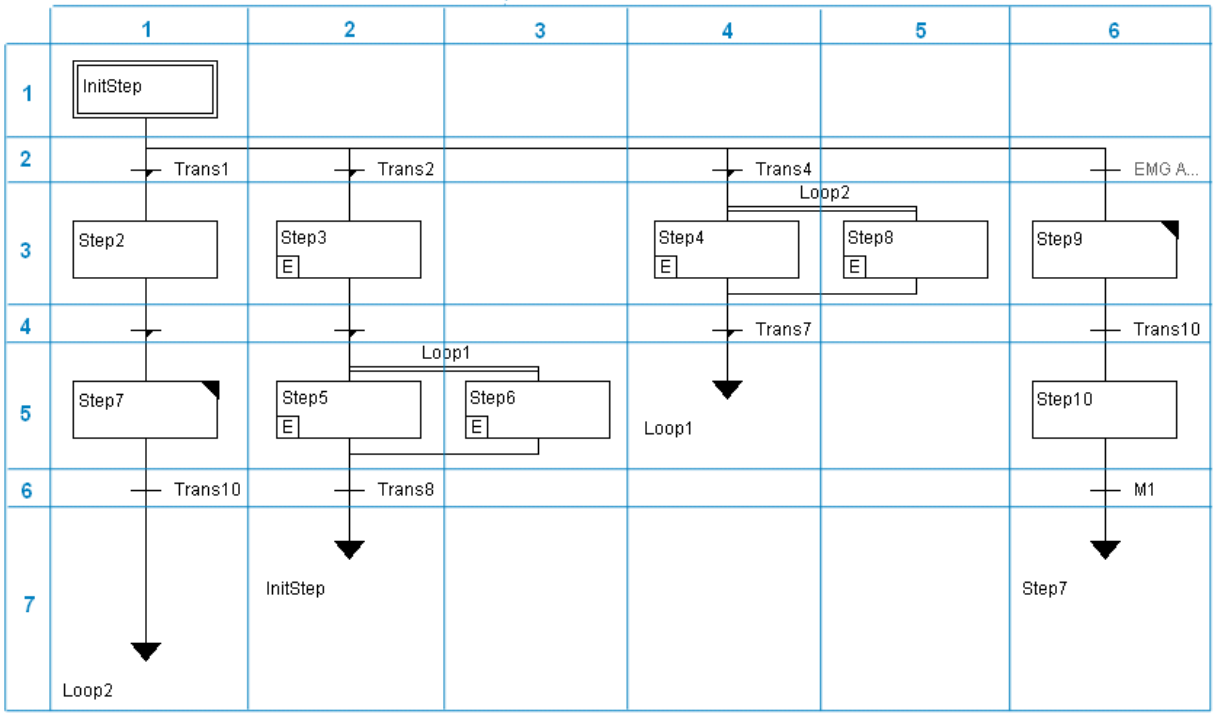
每个步应关联若干动作，具有零个相关动作的步应该被认为具有等待功能，即等待一个后继转换条件变为真。

动作可以是 ST 语言的一组语句、LD 语言的一组梯级、或 CFC 语言的一组程序。

当前仅支持 LD 语言的动作/转换。

8.9.3 SFC 组态操作举例

下面给出的示例能够帮助用户更多的了解 SFC 控制。示例中以行号、列号表示具体元素的位置。



组态操作示例

一个 SFC 控制是以初始步开始（第 1 行，第 1 列）。

初始步后面可以连接转换-步，也可以像这里连接选择分支。

选择分支包含 4 个子分支，分布于第 2 行的第 1、2、4、6 列。每个子分支都是以转换开始。当初始步执行完成后，按照从左到右的优先级执行某一子分支。

转换和步是成对添加或删除的。转换（步）可以添加转换、平行分支（动作、选择分支）。例如转换可以添加转换条件（如第 2 行，第 1 列、第 2 行，第 6 列、以及第 4 行，第 1 列）、平行分支等。

步 Step4（第 3 行，第 4 列）和 Step5（第 5 行，第 2 列）分别添加平行分支，依次分布在第 3 行，第 4、5 列和第 5 行，第 2、3 列，分支标号分别是 Loop2 和 Loop1。

平行分支用双水平线连接子分支。平行线上可以输入分支标号，主要用来标记跳转位置；每个子分支以步开始，表示当转换满足时，同时激活所有子分支的第一步。

如果前一步 InitStep（第 1 行，第 1 列）处于激活状态，此时 Trans4（第 2 行，第 4 列）条件满足，会同时激活步 Step4 和 Step8。

如果前一步 Step3（第 3 行，第 2 列）处于激活状态，此时转换条件（第 4 行，第 2 列）条件满足，会同时激活步 Step5 和 Step6。

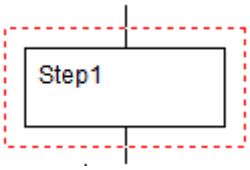
所有子分支都支持以跳转结束，跳转目标可以是步、并行分支标号、初始步（默认）。

这里的目标跳转位置有：

- 并行分支标号：第 7 行，第 1 列和第 5 行，第 4 列；
- 初始步：第 7 行，第 2 列；
- 步：第 7 行，第 6 列。

8.9.4 SFC 光标位置

通过单击来选中元素，选中的元素是用封闭的虚线框来标识。

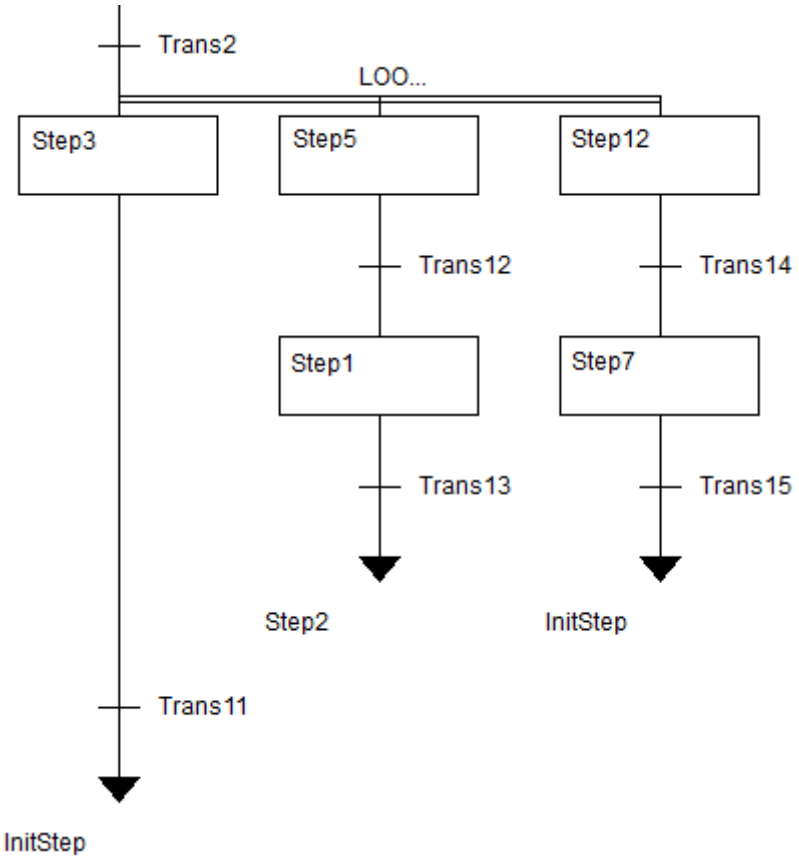


被选中的步

移动鼠标到元素上并按鼠标左键，可以选择一个元素（步、转换或跳转）。

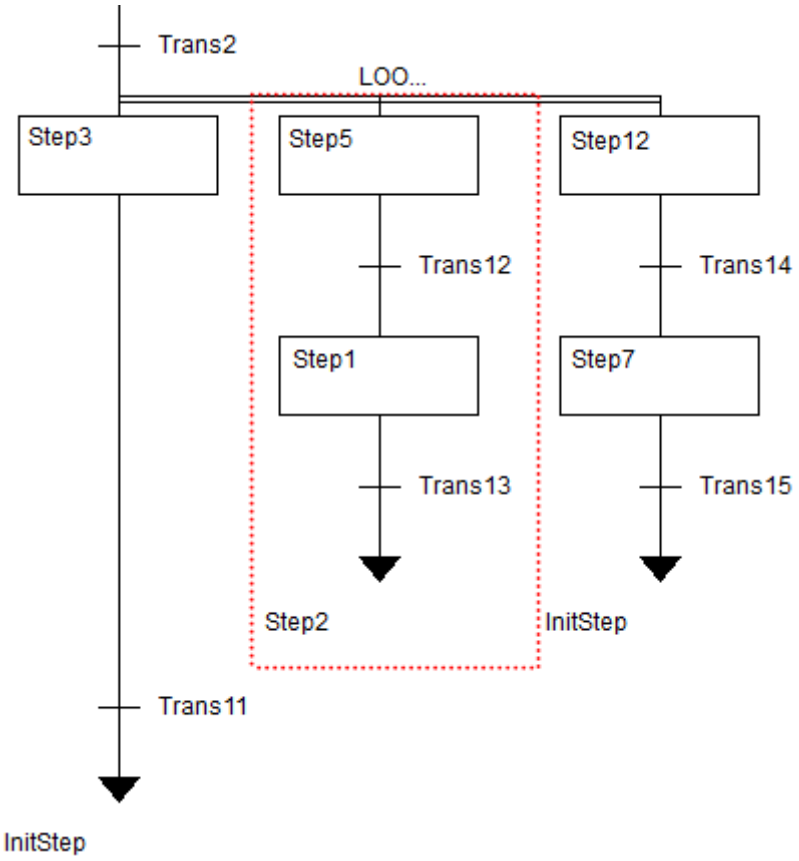
为了选择一组中多个元素，在选中的元素上按 **Shift**，并选择组的左下角或右下角的元素，这样就选中了包括这些元素的最小的组。

选择不同的元素组合，选择的结果可能会不同。



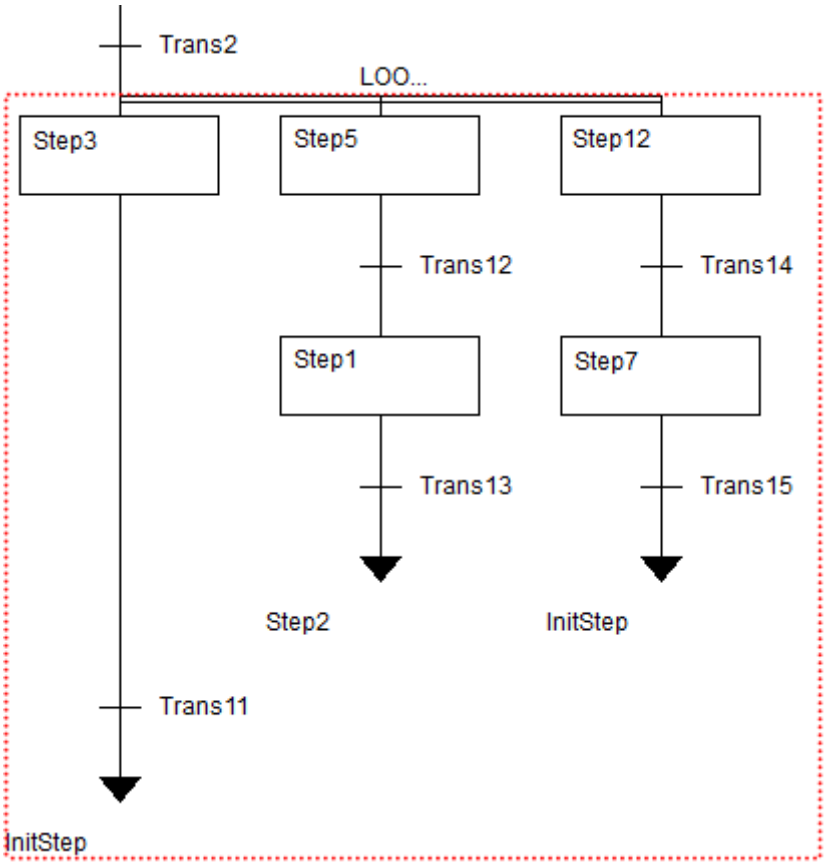
选中元素前

先选择元素 Step5，**Shift** 键按下时选择跳转元素 Step2，它们所在分支的所有内容被选中。



选中一组分支

先选择元素 Step3，按下 **Shift** 键的同时选择 Step3 所属分支（Trans11 和跳转 InitStep）外的其他任何元素，最终会得到同一种结果。



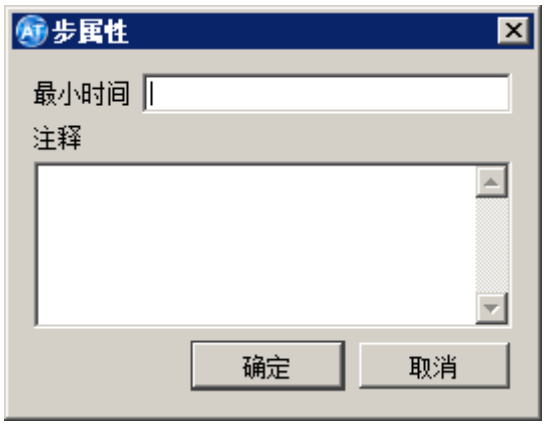
选中一组并行分支

步只能和它之前或随后的转换一起删除。

8.9.5 步属性

通过以下方式打开步属性：

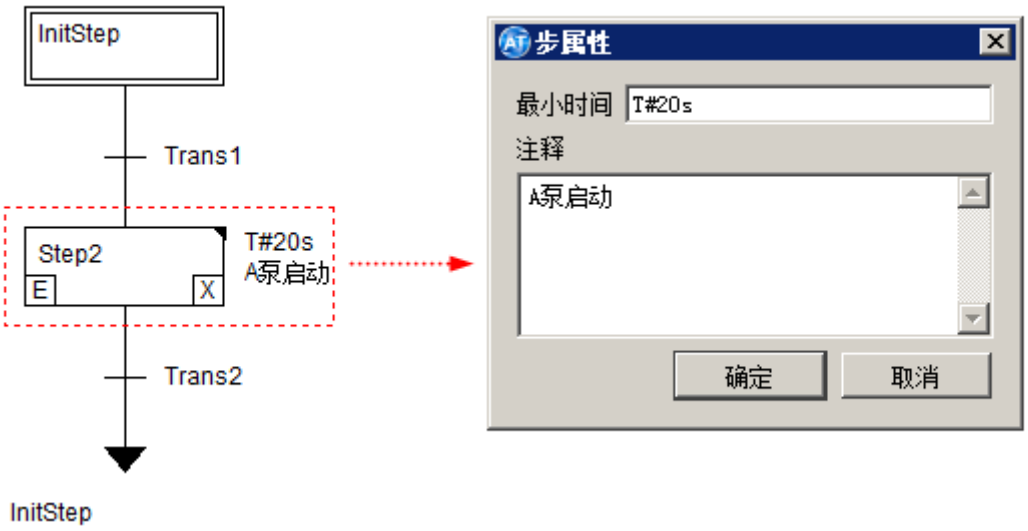
- 程序区：右击步，单击【步的属性】。



“步属性”窗口

最小时间：步运行时间的下限值，当步的运行时间小于**最小时间**时，即使其出口转移条件满足，程序也不会执行转移，一直到该步的执行时间超过**最小时间**后才执行转移。建议进行设置。

注释：步的相关说明，字符数不设限。确定设置内容后，本步的右边会显示设置的注释信息，如下图所示。显示的注释信息多少与设置的步高度和是否显示最小时间有关。



步属性设置示例

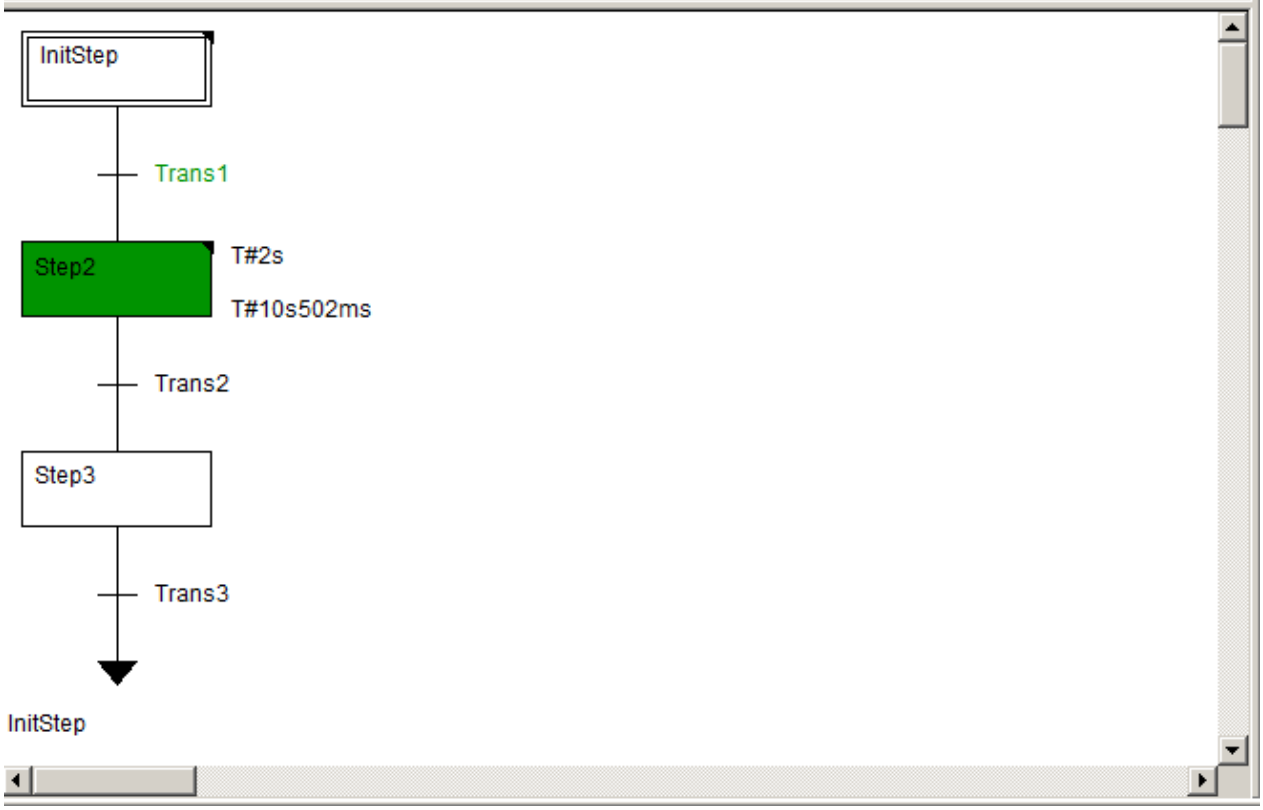
时间可以由 TIME 类型的变量表示，也可以用时间数据表示；常量的书写格式：T+“#”+时间值。

8.9.6 标志符

标识符用来控制 SFC POU 中的程序流，它在工程运行期间隐含创建。为了能读这些标识符，必须定义合适的局部变量。

- SFCInit：BOOL 类型变量。当这个变量值是 TRUE 时，顺序功能图返回到初始步，同时其它的 SFC 标志符也会被复位。初始步保持激活，但是不被执行。只有当 SFCInit 被重设置为 FALSE，顺序功能图才可以恢复正常执行。
- SFCPause：BOOL 类型变量。只有这个变量是 TRUE，顺序功能图停止执行，如下图所示。

序号	变量名	直接地址	变量说明	变量类型	在线值	掉电保护
0001	Trans1			BOOL	TRUE	FALSE
0002	Trans2			BOOL	FALSE	FALSE
0003	sfcinit			BOOL	FALSE	FALSE
0004	sfcreset			BOOL	FALSE	FALSE
0005	sfcpause			BOOL	TRUE	FALSE
0006	g1			INT	1	FALSE
0007	Trans3			BOOL	FALSE	FALSE



SFCPause 变量为 TRUE

- SFCReset：BOOL 类型变量。当该变量为 TRUE，则 SFC 程序被复位，返回到初始步，同时其它的 SFC 标志符也会被复位，初始步保持激活。与 SFCInit 不同之处在于，SFCReset 变量值为 TRUE 时，初始步的动作不会受影响，继续执行。同时，SFCReset 标识符可以在初始步中被复位到 FALSE。

8.9.7 插入步

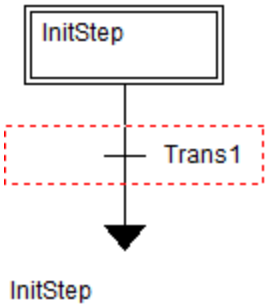
1. 向前插入步

选中转换或步，执行以下操作：

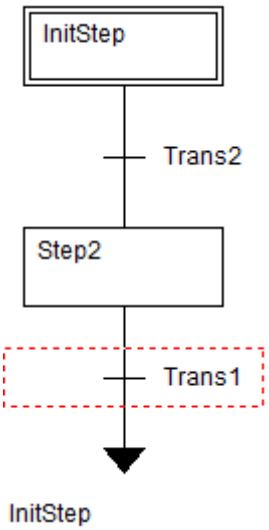
- 程序区：右击转换或步，单击【步—转换（前）】；

- 工具栏：  ；
- 快捷键：**Ctrl+T**。

选中转换时，在转换前插入一个新转换和一个新步。



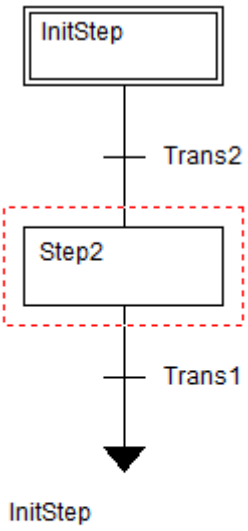
（a）



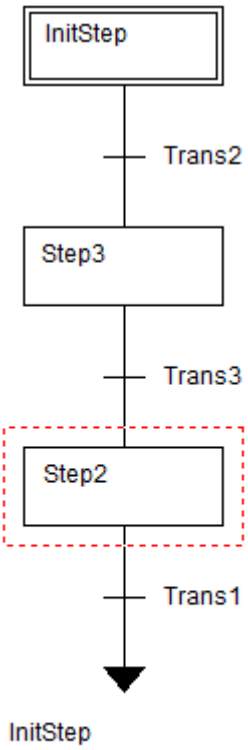
（b）

向前插入步-选中转换

选中步时，在步前插入一个新步和一个新转换，初始步（InitStep）不能向前插入步。



（a）



（b）

向前插入步-选中步

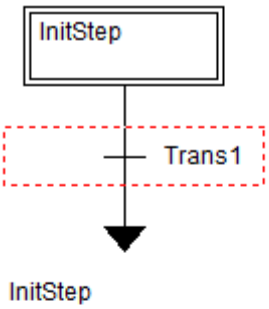
2. 向后插入步

选中步和转换，执行以下操作：

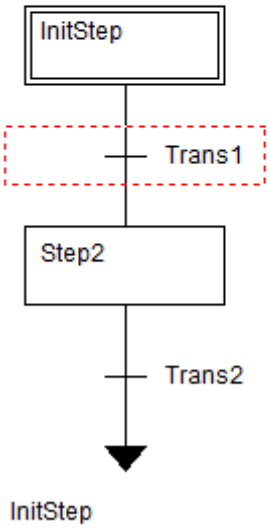
- 程序区：右击转换或步，单击【步—转换（后）】；

- 工具栏：  ；
- 快捷键：**Ctrl+E**。

选中转换时，在转换后插入一个新转换和一个新步。



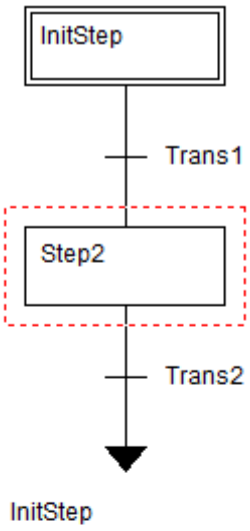
（a）



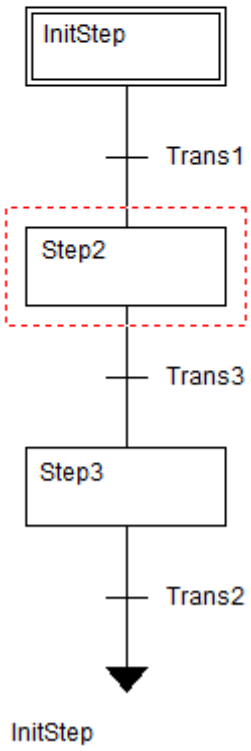
（b）

向后插入步-选中转换

选中步时，在步后插入一个新步和一个新转换。



（a）



（b）

向后插入步-选中步

实际上，插入步是插入“步-转换”对，使原有的 SFC 程序保持基本正确的逻辑流程。

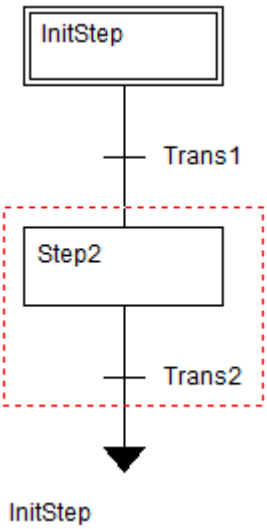
每个 SFCPOU 添加“步”的最大数是 400 个。

8.9.8 删除步

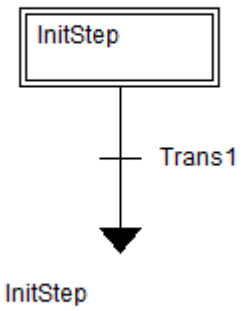
删除步也是删除“步-转换”对，使原有的 SFC 程序保持基本正确的逻辑流程。初始步不能被删除。

1. 删除单个步和转换对

选中需要删除的“步-转换”对：单击鼠标左键选中步 Step2 后，按下 **Ctrl** 或 **Shift** 键，同时单击转换 Trans2，同时选中 Step2 和 Trans2，执行删除命令。



（a）

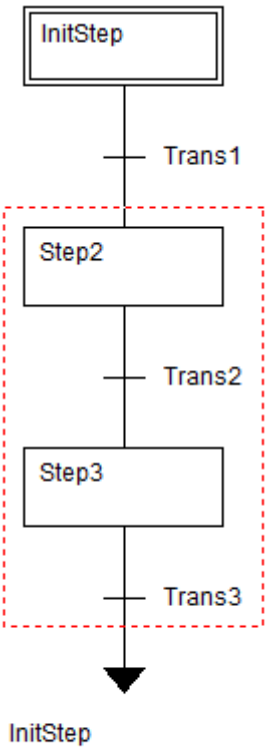


（b）

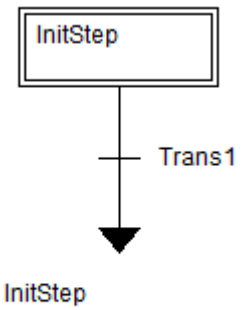
删除单个“步-转换”对

2. 删除多个“步-转换”对

选中需要删除的“步-转换”对：单击鼠标左键选中转换 Trans3 后，按住 **Ctrl** 键，依次单击步 Step3、转换 Trans2 和步 Step2，或按住 **Shift** 键，选中转换 Trans3 和步 Step2，执行删除命令。



（a）



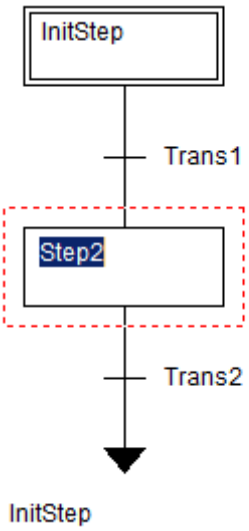
（b）

删除多个“步-转换”对

8.9.9 修改步名称

新建步时系统会自动为其分配名称，如需对其修改，新名称需要按照普通变量命名规则进行设置，并确保在工程中不重名。

双击步的名称文本所在的矩形区域，步名变为可编辑状态，输入新步名即可。



修改步名称

输入新名称后，在任意空白处单击或者 **Enter** 键确定，则编辑结束。

8.9.10 添加入口动作

入口动作，指本步刚进入激活状态时执行的动作，当前仅支持添加梯形图 LD 类型动作。

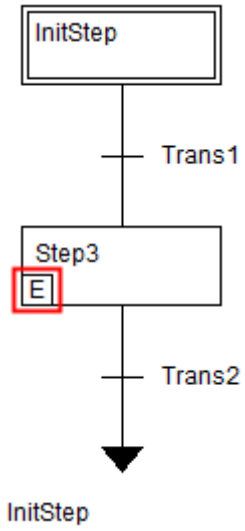
选中步，通过以下方式添加入口动作：

- 程序区：右击步，单击【添加入口动作】；
- 快捷键：**Ctrl+I**。

弹出选择动作语言类型的对话框。

入口动作语言类型

选择动作描述语言后，单击**确定**，完成了添加入口动作操作，在步的左下角会出现E标识。



入口动作标识

双击该标识，可以编辑步的入口动作。

初始步（InitStep）不能添加入口动作。

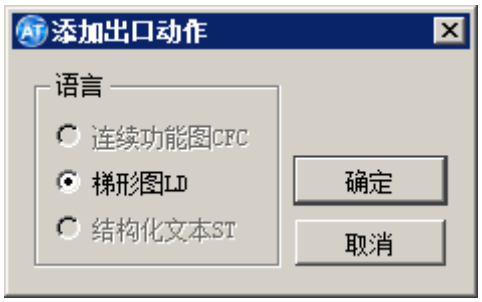
8.9.11 添加出口动作

出口动作，指本步即将退出激活状态时，执行的动作；当前仅支持添加梯形图 LD 类型动作。

选中步，通过以下方式添加出口动作：

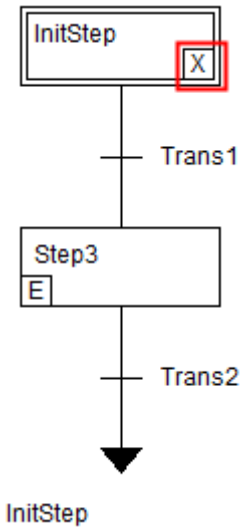
- 程序区：右击步，单击【添加出口动作】；
- 快捷键：**Ctrl+G**。

弹出选择动作语言类型的对话框。



出口动作语言类型

选择动作描述语言后，单击**确定**，完成添加出口动作操作，在步的右下角会出现X标识。



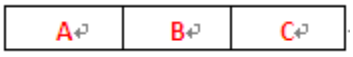
出口动作标识

双击该标识，可以编辑步的出口动作。

8.9.12 关联动作

通过【关联动作】操作，将 SFC POU 中增加的动作关联到某一步中。

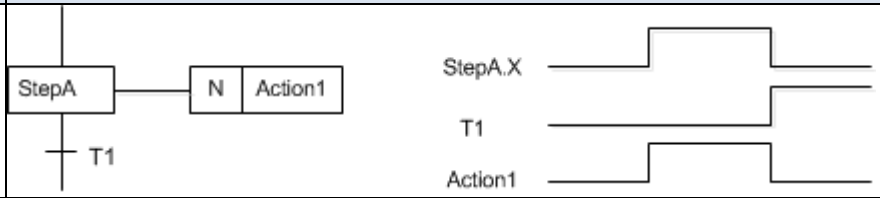
动作与步的关联关系是由动作限定符决定的。

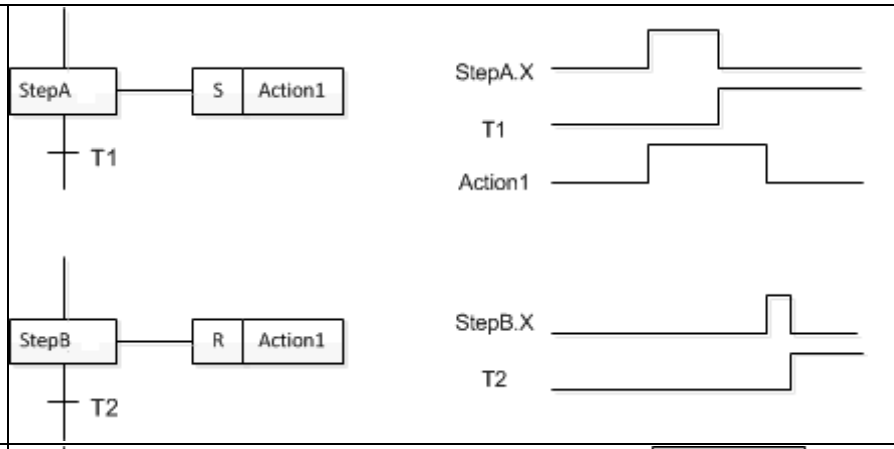
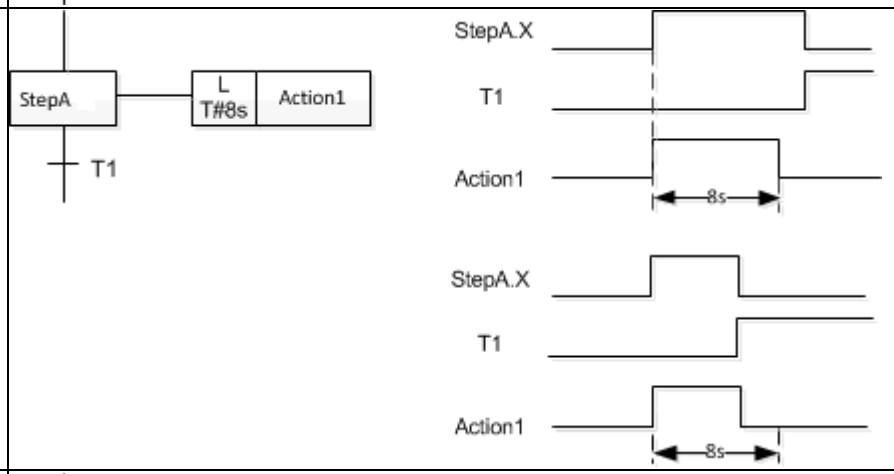
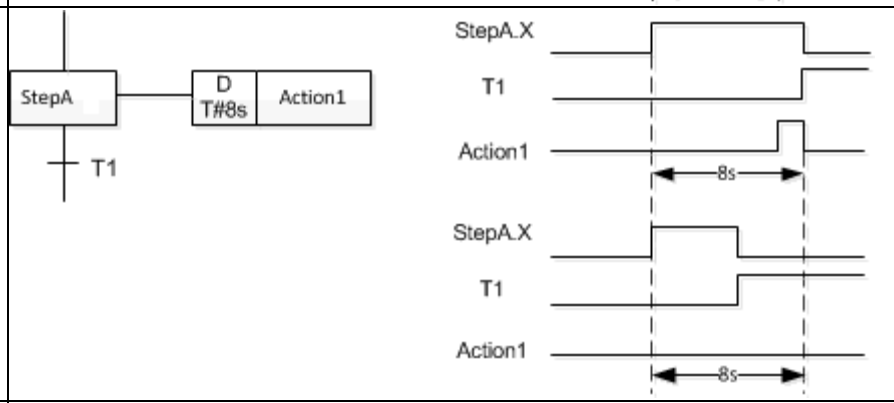


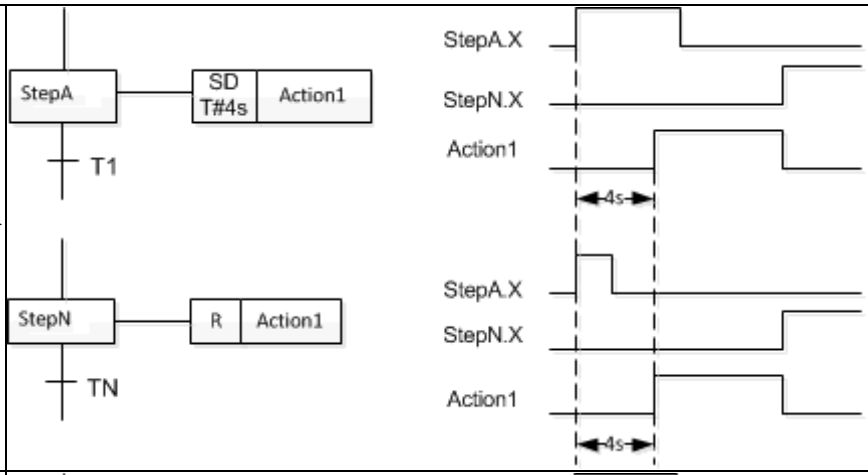
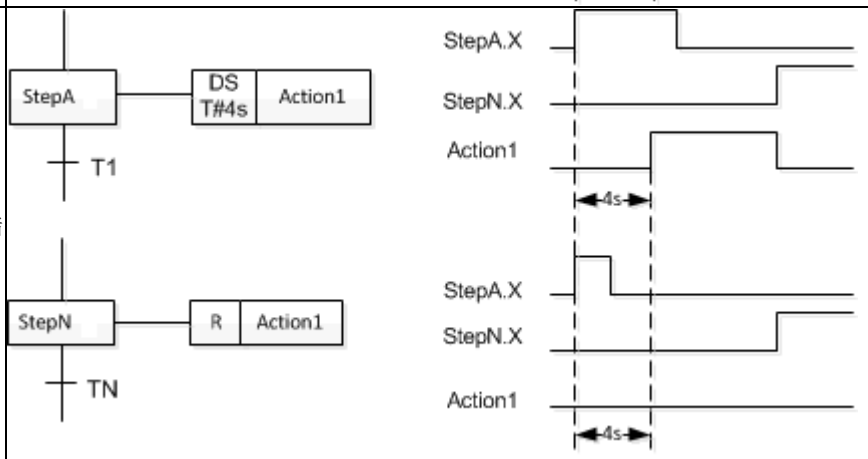
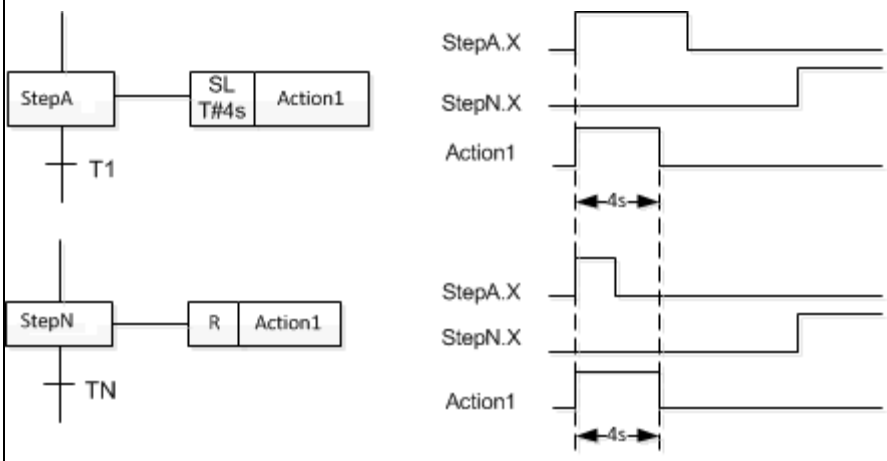
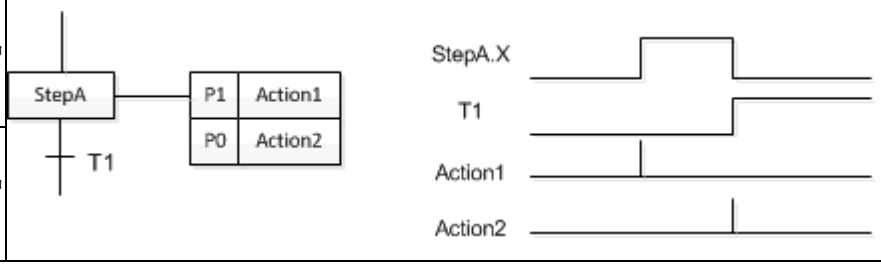
动作限定符

A 区域为动作限定符，自带下拉框，可以从中选择所需要的限定符；如果 A 区域选择与时间相关的限定符可在 B 区域对时间进行编辑，默认值为 T#1S；C 区域为一个动作名称， 当在 POU（SFC）中增加动作后（请参见增加动作），则动作名会在 C 区域的下拉框中显示，可以对其进行调用；该动作是否被激活与 A 区域的限定符有密切关系，如下表所示，列出了不同限定符和动作的执行关系。

动作限定符

序号	限定符	类型说明	说明	功能
1	N	非存储		动作执行跟步活动时间一样长。
2	S	设置存储		步活动时动作执行，即使步变成非活动也持续 执行，要取消执行，使用 R 限定符。
3	R	存储复位		当步变为活动，

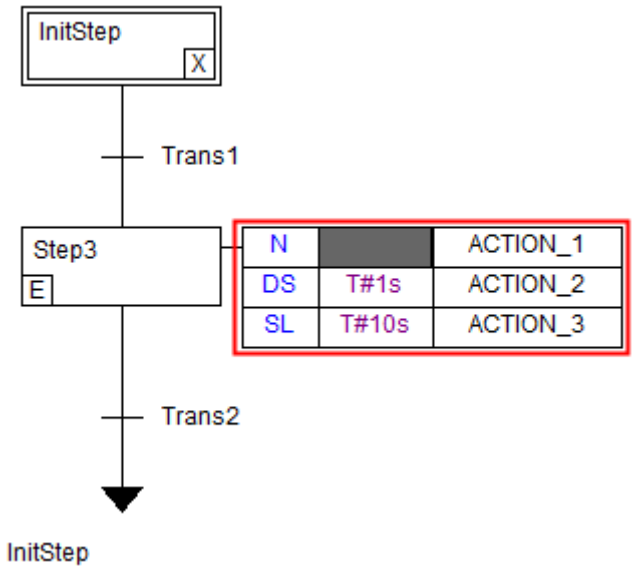
				且动作是由“S、SL、SD 或 DS”限定符，指定动作停止并复位。如果动作正在执行，但不是由“S、SL、SD 或 DS”限定符执行，动作只是复位（不停止）。
4	L	限时		当步活动后，动作执行直到用户指定时间到达。如果步变为非活动，动作执行将终止。
5	D	延时		从步活动开始，过去用户指定时间后动作执行。如果在用户指定时间到期之前步变为非活动，动作将不执行。
6	P	脉冲		若最终扫描功能使能，当步活动时动作执行 2 个扫描周期。若最终扫描功能非使能，当步活动时，动作执 行 1 个周期。

7	SD	存储和延时		步活动后直到用户指定时间到达，动作开始执行，即使步变成非活动也持续执行。要取消执行，使用 R 限定符。
8	DS	延时和存储		执行时类似"SD"。同 SD 不同，如果步在用户指定延迟时间到期之前变为非活动，动作将不操作。要取消执行，使用 R 限定符。
9	SL	存储和限时		步活动时动作执行，直到用户指定时间到期。与“L”不同的是，即使步变成非活动也持续执行。要取消执行，使用 R 限定符。
10	p1	上升沿脉冲		每当步变为活动时动作执行一次。
11	p0	下降沿脉冲		每当步变为非活动时动作执行一次。

选中步：

- 程序区：右击转换，单击【关联动作】；
- 快捷键：**Ctrl+K**。

即为步关联了一个新的动作。如步 Step3 关联了 3 个动作。

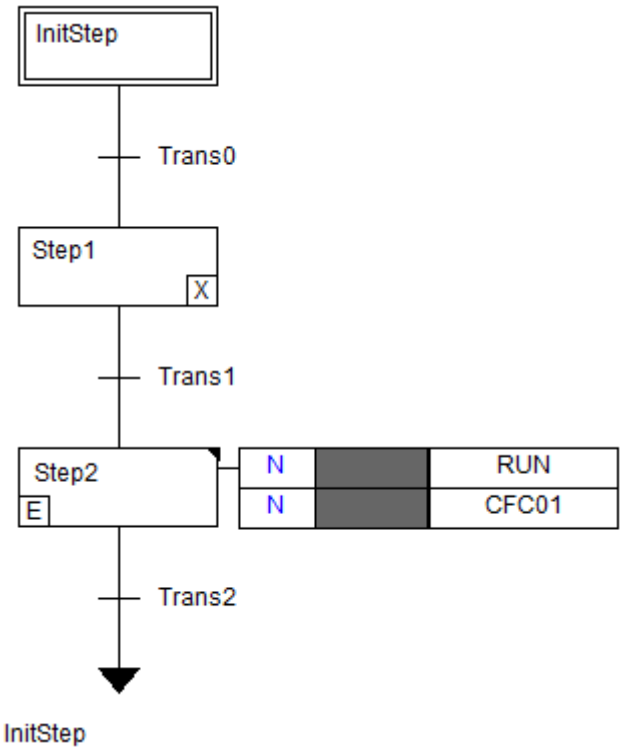


关联动作

每个“步”最多能关联 9 个动作。

8.9.13 SFC 步动作执行时序

如图所示，Step1 添加了出口动作，Step2 添加了入口动作、步动作、IEC 关联动作。



步执行动作示例

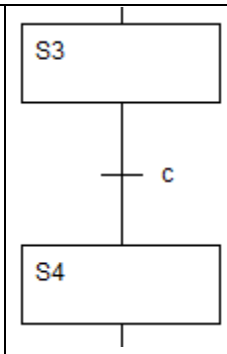
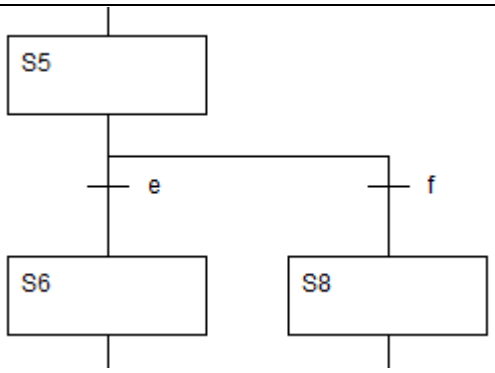
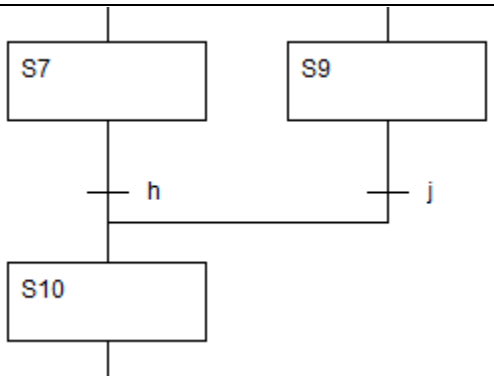
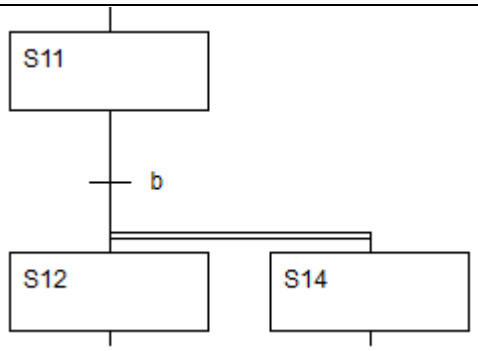
当前扫描周期内，步动作的执行如表所示。

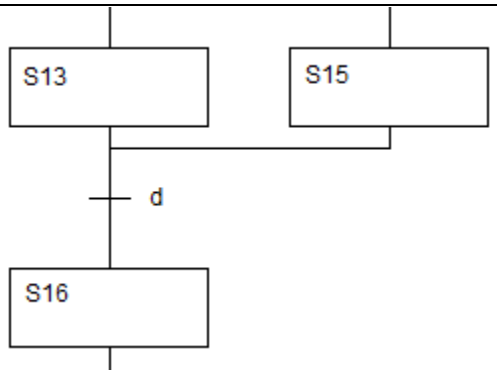
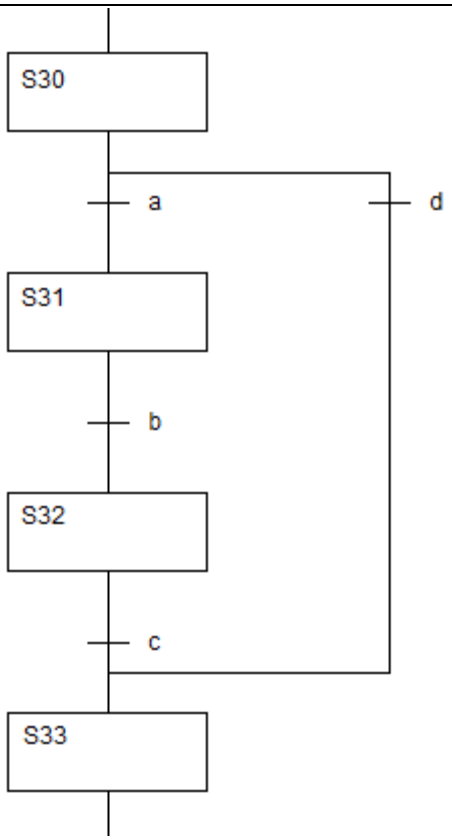
动作执行顺序

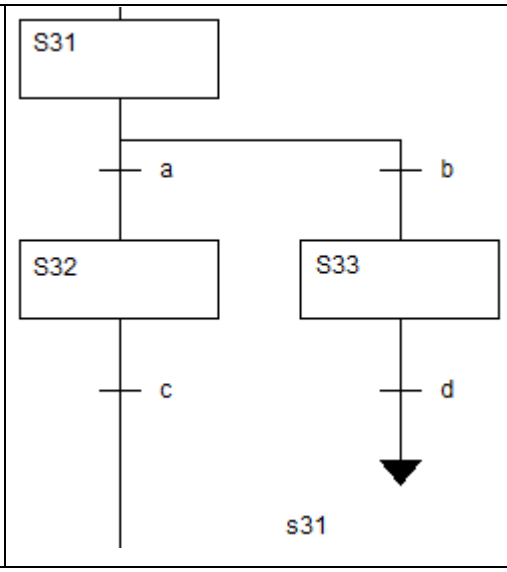
执行时序	动作	动作处理
1	Step1 出口动作	当 Step1.x=TRUE，且 Trans1=TRUE，该动作被执行。（它的入口动作和步动作已经在上一个周期被执行了）
2	Step2 入口动作	如果 Trans1=TRUE，且 Step2 被激活（即 Step2.x=TRUE），则入口动作被执行
3	Step2 步动作	Step2 被激活后，且入口动作已执行完，则执行步动作
4	Step2 IEC 关联动作	步动作执行完毕，IEC 关联动作按照字母顺序执行
5	转移，激活下一步	如果 Step2 是激活状态，Trans2 为 TRUE，且 Step2.t 大于最小时间，则后续步被激活

8.9.14 SFC 演变规则

序号	举例	规则

1		单个分支：连续地重复进行步—转换的交替。 举例： 只有当步 S3 处于活动状态且转换条件 c 为真时，从 S3 到 S4 的转移才会发生。
2		选择—分支：几个分支之间的选择可以用水平线下面的若干转换符号表示，有几个转换符号，就有几个可能的不同转移。 举例： 仅当 S5 活动且转换条件 e 为真时，从 S5 到 S6 的转移才会发生； 或仅当 S5 活动且 f 为真、e 为假时，从 S5 到 S8 的转移才会发生。
3		选择分支的汇集：选择分支的结束由水平线上面的若干转换符号表示，有几个转换符号，就有几个结束的选择路径。 举例： 仅当 S7 活动且转换条件 h 为真时，从 S7 到 S10 的转移才会发生； 或仅当 S9 活动且 j 为真时，从 S9 到 S10 的转移才会发生。
4		并列—分支：在并列分支的双水平线上面只可能有一个公用的转换符号。 举例： 仅当 S11 活动且公用转换相关的转换条件 b 为真时，从 S11 到 S12、S14、……的转移才会发生。S12、S14 等同时激活后，每个分支独立地执行。

5		并列分支—汇集：在并列分支的水平线下面只可能有一个公用的转换符号。 举例： 仅当与水平线上面连接的所有步都活动，且与公用转换相关的转换条件 d 为真时，从 S13、S15、……到 S16 的转移才会发生。
6		分支跳变：“分支跳变”是选择分支的一种特殊情况，在其中的一个或多个分支不包含步。 举例： 当 a 为假且 d 为真时，S30 到 S33 的演变才会发生，即跳过序列 S31 和 S32。

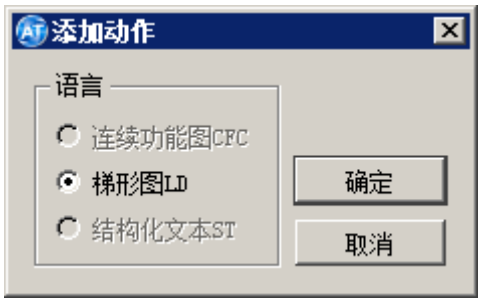
7		分支跳转：是选择分支或并列分支中的一种特殊情况，在其中的一个或多个分支跳转到其它步。 举例：当 S33 活动且 d 为真时，S33 到 S31 的转移才会发生。

8.9.15 添加动作/转换

8.9.15.1 添加步动作

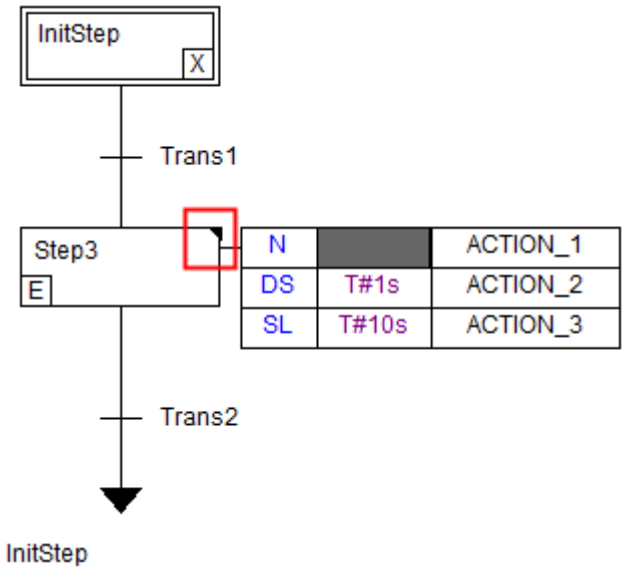
选中步：

- 程序区：右击步，单击【添加动作/转换】；
- 快捷键：**Ctrl+B**。



步动作语言类型

选择动作描述语言后，单击**确定**，完成添加步动作操作。在步的右上角会出现标识。



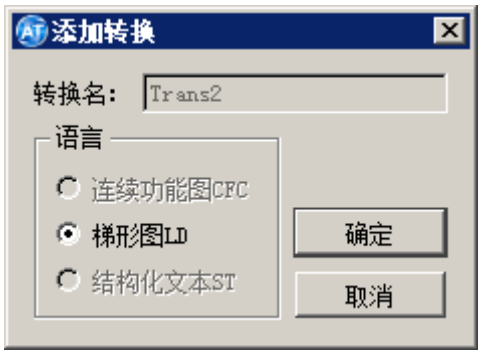
步动作标识

双击该标识，打开相应语言环境的步编辑页面，可以编辑步动作。

8.9.15.2 转换添加转换

选中转换条件，执行【添加动作\转换】命令。

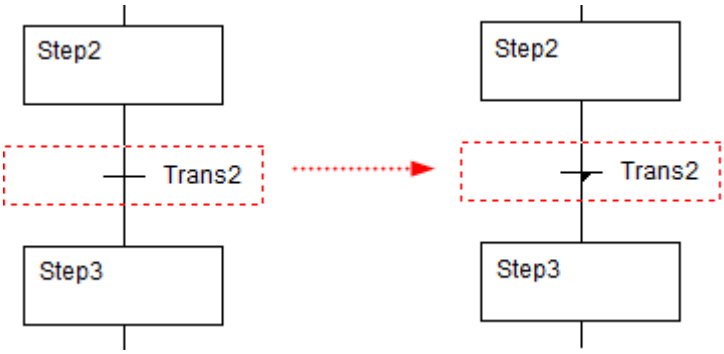
- 程序区：右击步，单击【添加动作/转换】；
- 快捷键：**Ctrl+B**。



转换语言

选择转换条件描述语言，单击**确定**即可。

转换在添加转换后，转换位置会出现 ┼ 标识。



已添加转换的标识

当添加转换后，就有了两个转换条件，这时只有用 LD 编辑的转换条件有效。

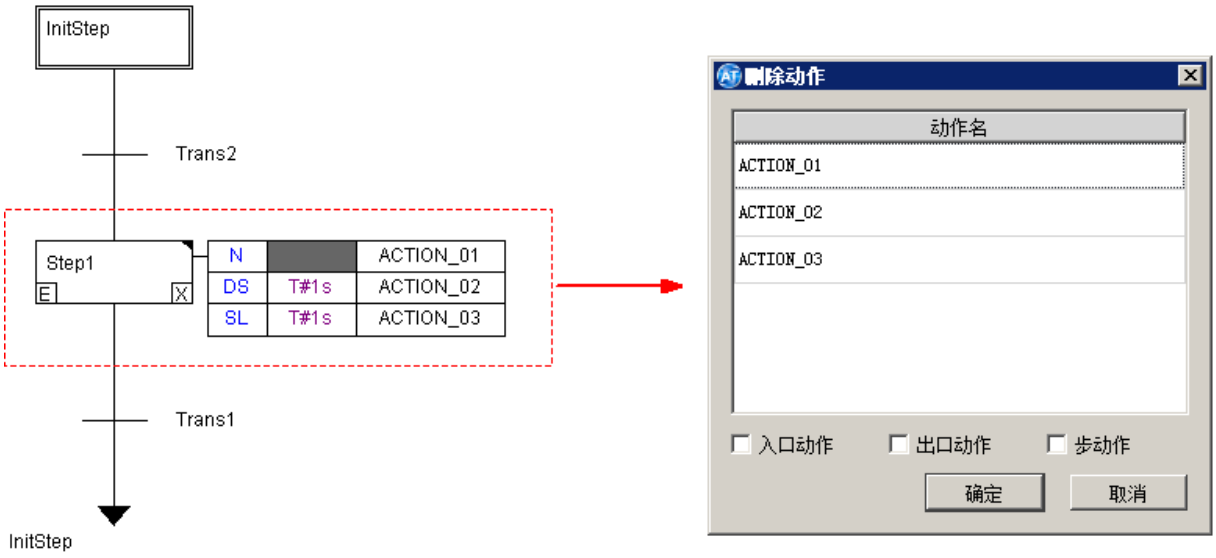
8.9.16 移除动作/转换

1. 移除步动作

对已添加动作的步，按实际需要进行移除。

选中步：

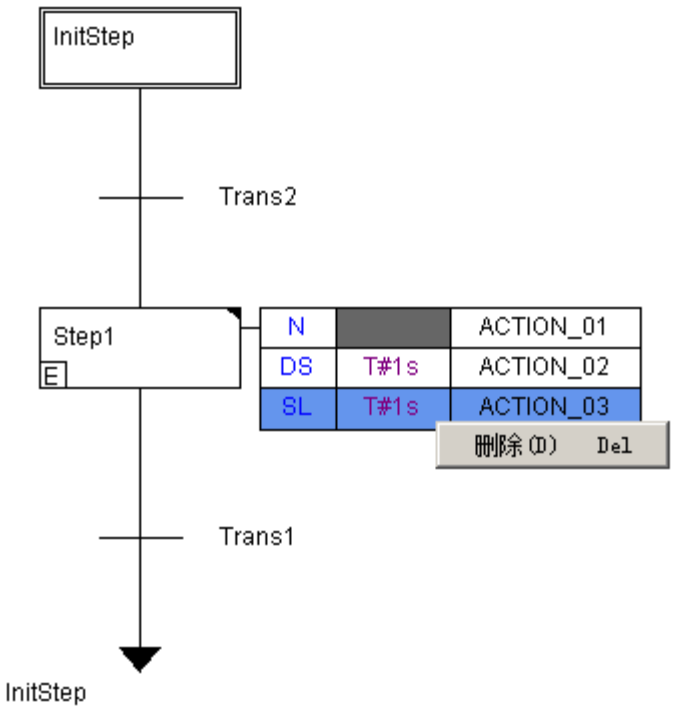
- 程序区：右击步，单击【移除动作/转换】；
- 快捷键：**Ctrl+U**。



移除动作

当步只有一个动作时，执行【移除动作/转换】命令后，会直接将动作删除，不会弹出“删除动作”对话框。

选择后，单击**确定**，完成移除操作。关联动作的移除也可以直接右击需要删除的动作，选择【删除】命令。



删除关联动作

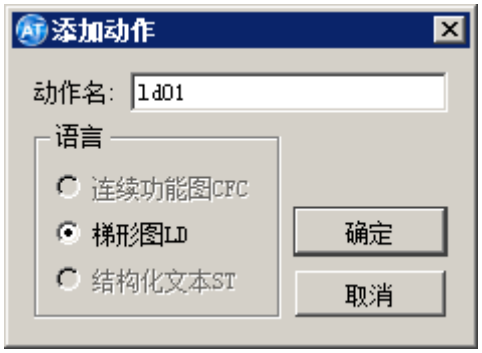
2. 移除转换

选中转换，右键菜单选择【移除动作\转换】命令，已添加的转换被删除。

8.9.17 增加动作

右击工程管理树下 SFC 语言的 POU 名，选择弹出菜单的【增加动作】项，可以为该 POU 增加一个动作。

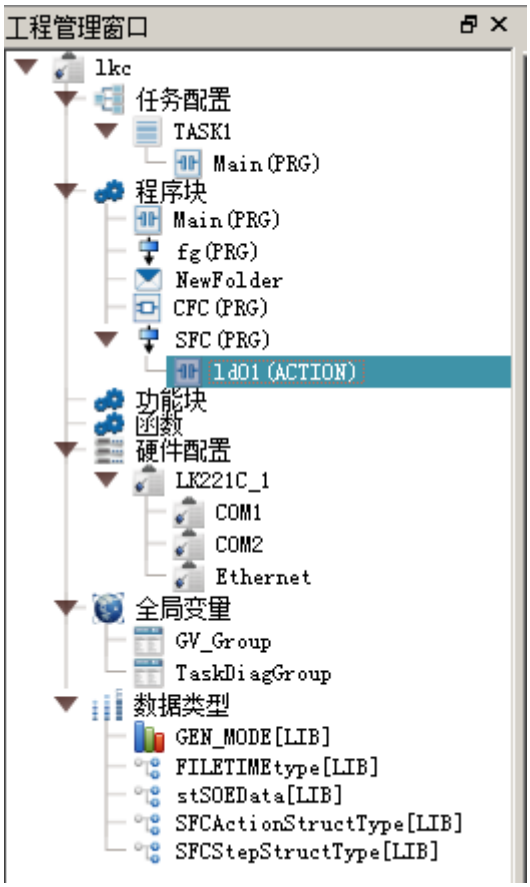
增加动作操作



添加动作

输入**动作名**，不超过 30 个字符，选择编辑语言后，单击**确定**，添加动作完成。在 POU 子目录会出现动作名。

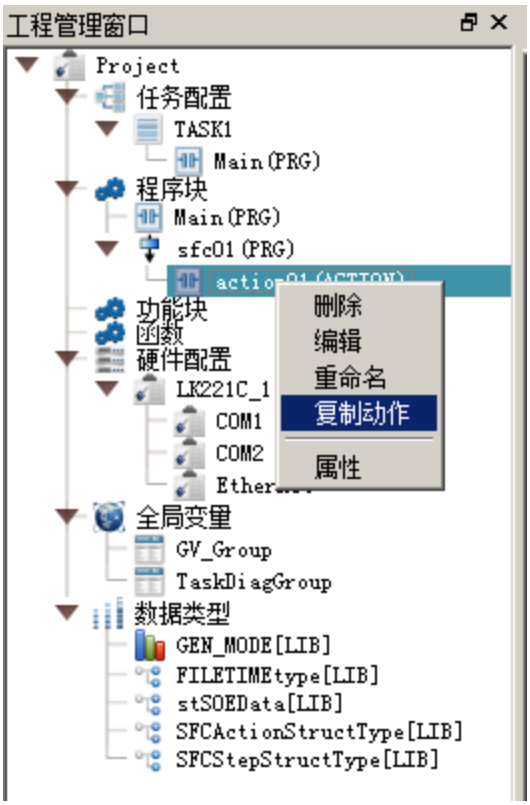
如下图所示，SFC(PRG)添加了一个编辑语言是 LD，动作名是 ld01(ACTION)的动作。



添加动作节点

8.9.18 复制动作

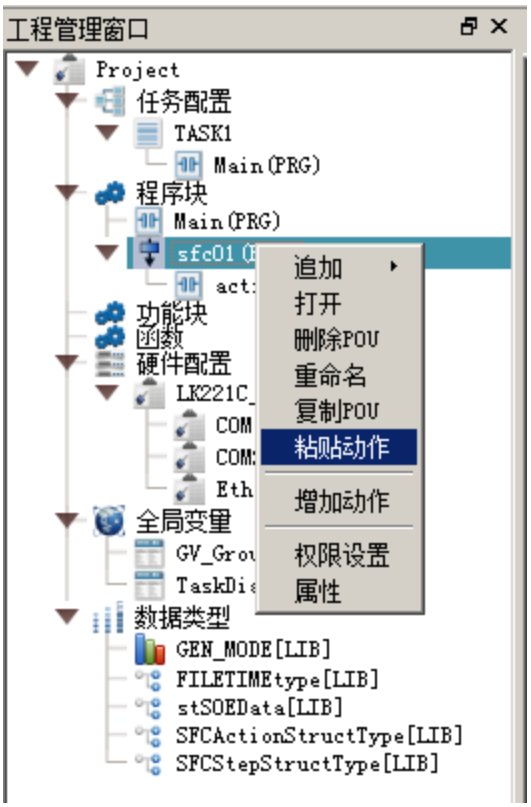
右击动作名，在快捷菜单中可以选择对该动作进行相关操作。



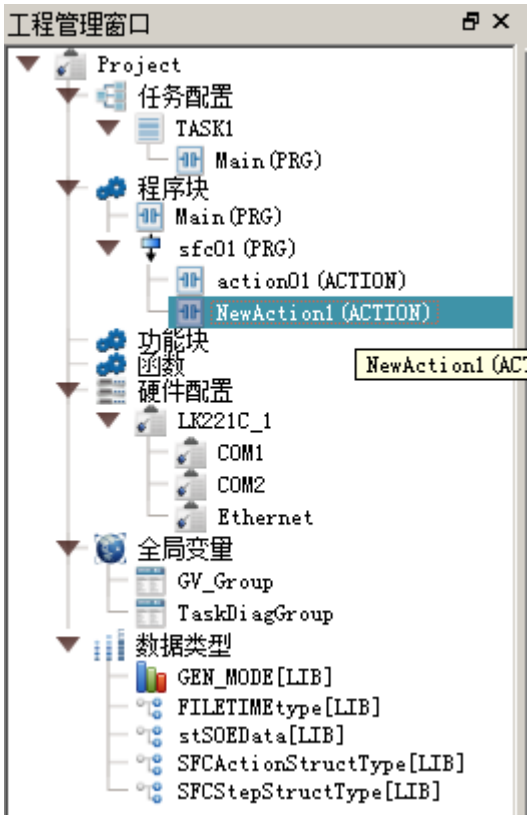
动作相关操作

执行其中的【复制动作】命令后，用户可以将复制的内容粘贴给对多个 SFC POU。

如图所示，复制 sfc01 的 IEC 动作 action01，执行【粘贴动作】命令后，系统会自动将动作名变为“NewAction*”。


工程管理窗口
Project
任务配置
TASK1
Main(PRG)
程序块
Main(PRG)
sfc01 (
act
功能块
函数
硬件配置
LK221C_
COM
COM
Eth
全局变量
GV_Gro
TaskDi
数据类型
GEN_MODE[LIB]
FILETIMEtype[LIB]
stSOEData[LIB]
SFCActionStructType[LIB]
SFCStepStructType[LIB]
追加
打开
删除POU
重命名
复制POU
粘贴动作
增加动作
权限设置
属性

（1）


工程管理窗口
Project
任务配置
TASK1
Main(PRG)
程序块
Main(PRG)
sfc01(PRG)
action01(ACTION)
NewAction1(ACTION)
功能块
函数
硬件配置
LK221C_1
COM1
COM2
Ethernet
全局变量
GV_Group
TaskDiagGroup
数据类型
GEN_MODE[LIB]
FILETIMEtype[LIB]
stSOEData[LIB]
SFCActionStructType[LIB]
SFCStepStructType[LIB]
NewAction1(AC

（2）

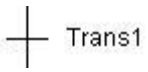
复制、粘贴动作操作

每个 SFC 关联的动作最大数是 1024 个，其中包含“步”动作（出\入口动作、步动作）、“转换”动作和 SFC POU 的 IEC 动作。

复制动作时，复制的内容不包含动作中局部变量的定义。

8.9.19 转换元素

转换表示控制从一个或多个前驱步沿相应的有向连接转换到一个或多个后继步所依据的条件。转换方向是从前驱步底部到后继步顶端。



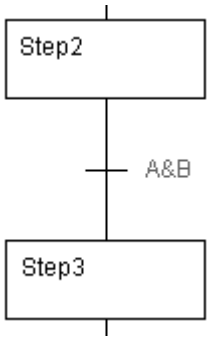
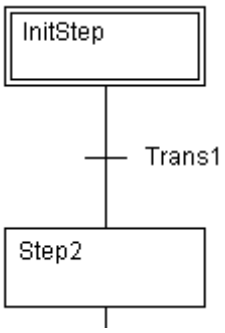
转换

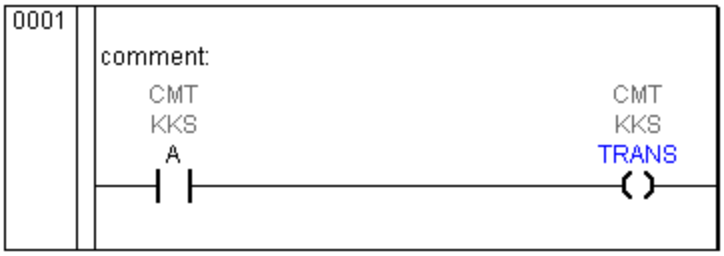
每个 SFC POU 添加“转换”的最大数是 400 个。

每个转换应有一个转换条件，它是一个单 BOOL 表达式的求值结果。总之，为真的转换条件应由符号“1”或关键字“TRUE”来表示。转换的求值可以由 LD、CFC 和 ST 得出。

在转换条件求值期间，如果给变量赋值而不是给转换，是一个错误情况。转换相关描述见下表。

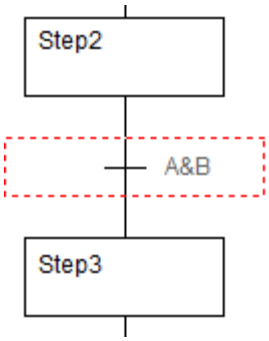
转换相关描述

序号	举例	说明
1		前驱步 ST 语言描述的转换条件，转换条件为单 BOOL 求值表达式 后继步
2		前驱步 转换名 后继步

3		LD 语言描述的转换条件
4		CFC 语言描述的转换条件
5	TRANS:=A&B;	ST 描述的转换条件

1. 单 BOOL 表达式做转换条件

直接在转换名称处写入单 BOOL 表达式作为转换条件。



转换条件-单 BOOL 表达式

2. 编辑已添加的转换

无论是转换条件还是已添加的转换，在需要对转换内容进行编辑时，双击转换条件或已添加转换的标识即可。

3. 剪切、复制、粘贴

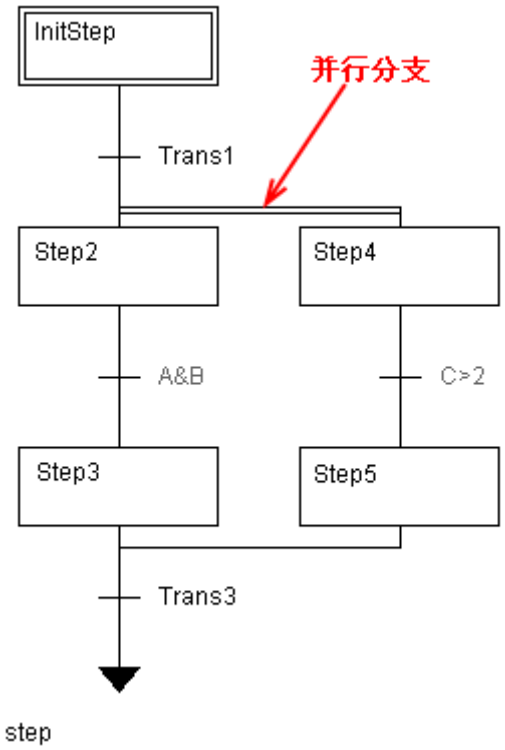
可以剪切和复制的单元包括：步与转换组成的序列对、并行分支和选择分支。复制的单元内容中不能包括初始步。

粘贴，剪贴板上的数据是以下几种单元的任意一种时，可以执行粘贴命令：

- 以步开始并且以步结束的单元；
- 以转换开始并且以转换结束的单元；
- 以步开始并且以转换结束的单元；
- 以转换开始并且以步结束的单元。

8.9.20 并行分支

两个或者两个以上以步起始并且以步结束的 SFC 单元，通过相互平行的双实线并行起来形成并行分支。



并行分支

当转换 TRANS1 为真并且初始步（InitStep）活动时，同时激活 Step2 和 Step4。

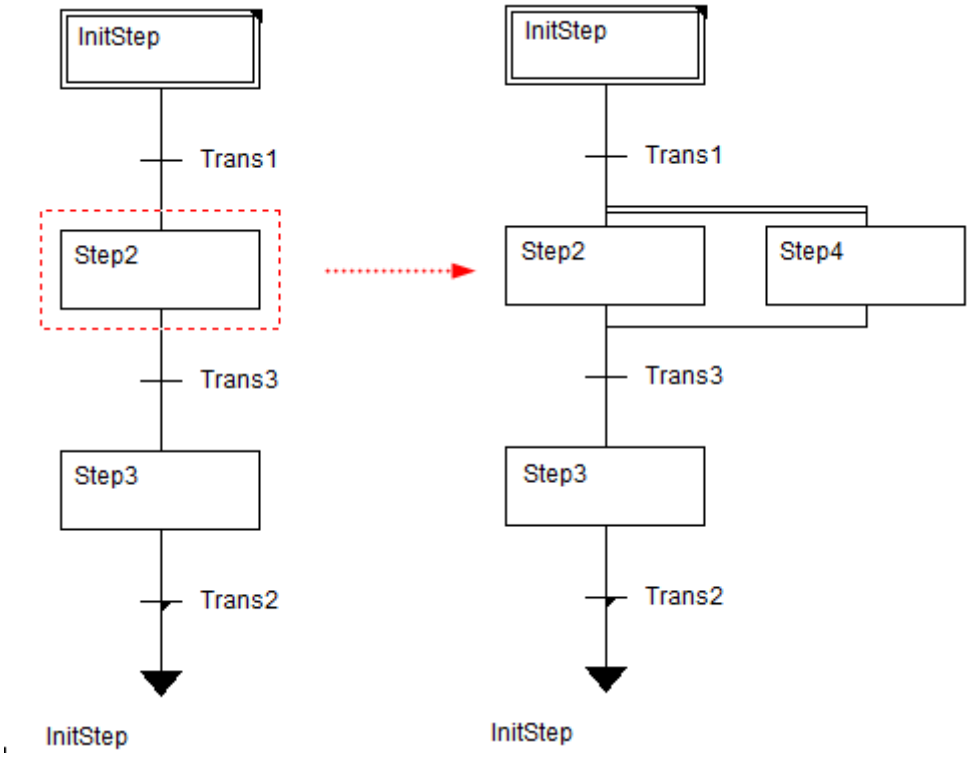
1. 增加并行分支

选中单个步或者一个以步开始且以步结束的单元，通过以下方式增加向左或向右的并行分支：

- 工具栏：、；
- 程序区：右击选中的步或单元，单击【并行分支（右）/（左）】。

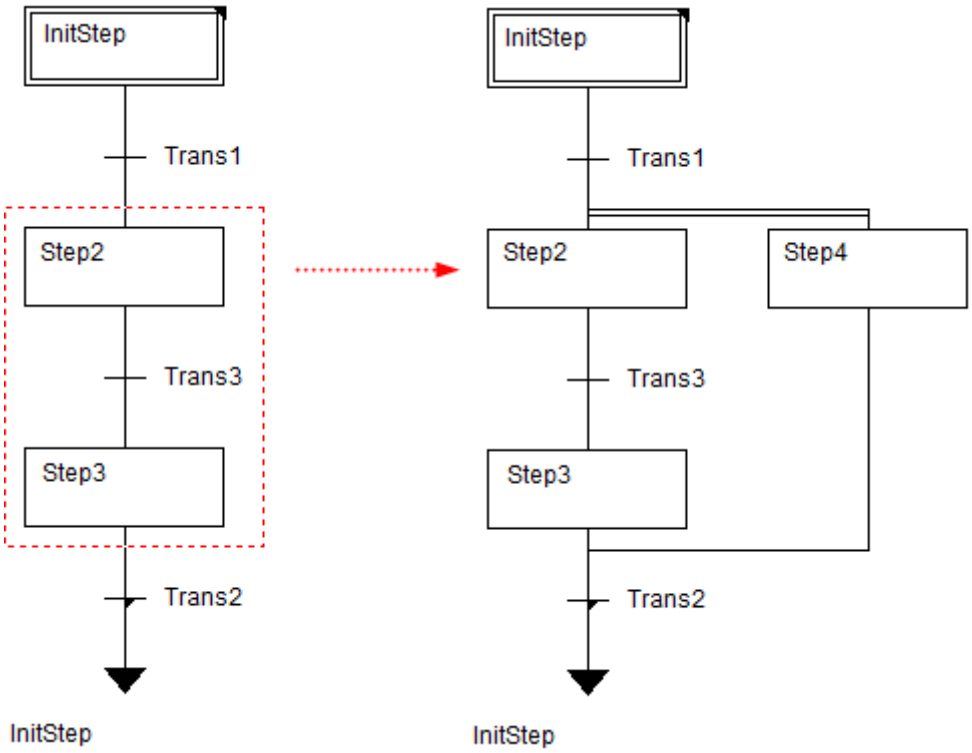
只介绍向右增加并行分支，向左增加并行分支类似。

选中单个步，增加向右并行分支，如图所示。



增加右并行分支—单个步

选中一个以步开始且以步结束的单元，如果是复选，需要按住 **Ctrl** 键或者 **Shift** 键选中步单元。向右增加并行分支，如图所示。



增加右并行分支—单元

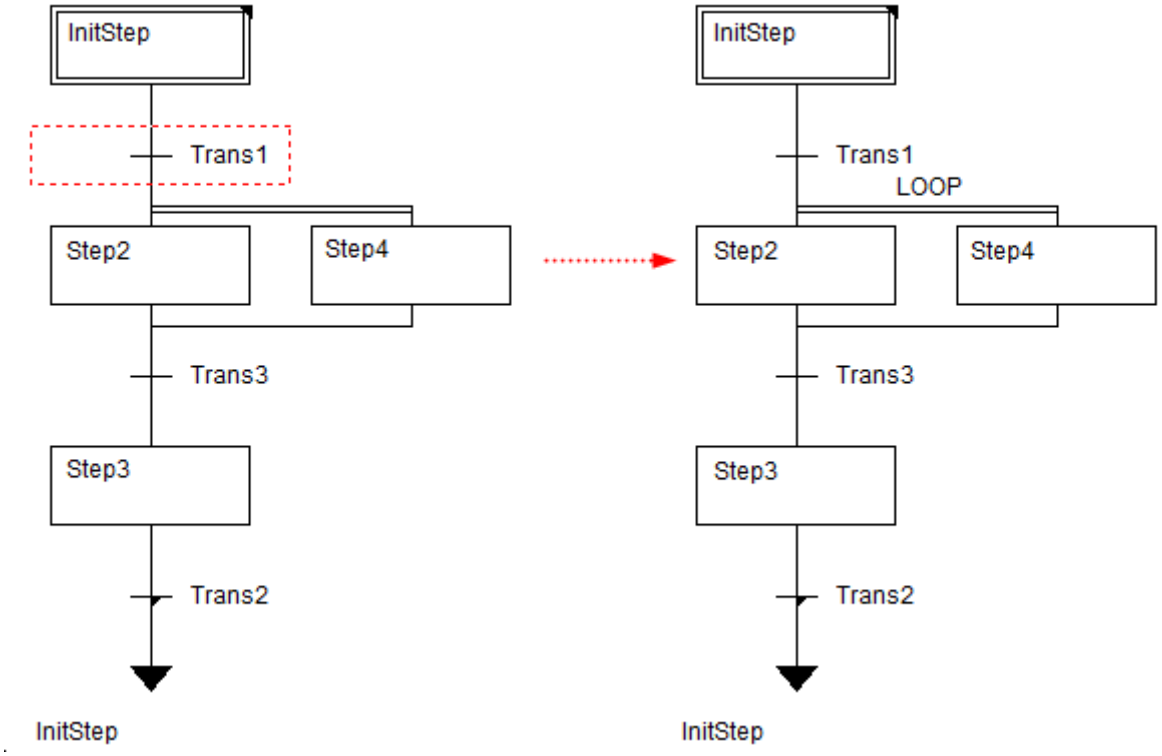
2. 并行分支标号

可以为整个并行分支添加一个标号，作为并行分支位置的标示，可以跳转到该位置。

选中并行分支前的转换：

- 程序区：右击转换，单击【增加并行分支标号】；
- 快捷键：**Ctrl+Q**。

如图所示，缺省标号名“LOOP”，可以对其进行修改，双击选中后，输入实际标号名（转换-跳转位置），在任意空白处单击鼠标左键或按 **Enter** 键完成重命名操作。



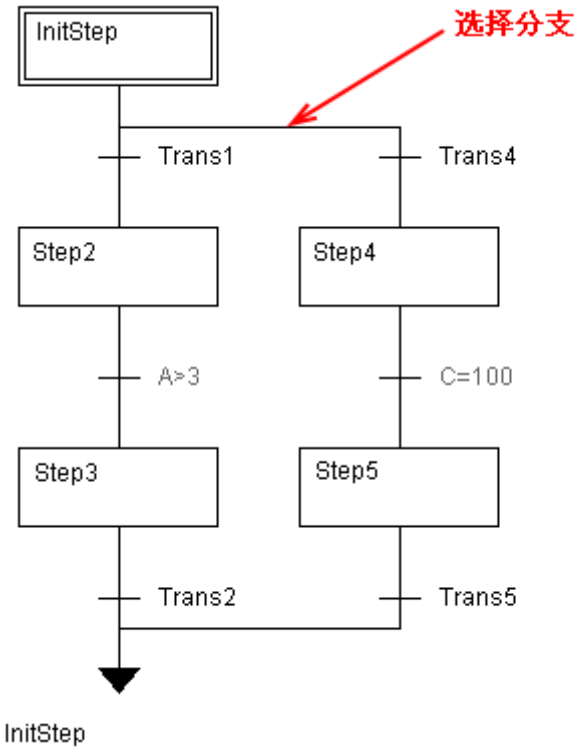
并行分支标号

3. 删除并行分支

选中并行分支（左或右）的所有元素，执行删除命令即可。

8.9.21 选择分支

两个或者两个以上以转换起始并且以转换结束的 SFC 单元，通过相互平行的单实线并行起来形成选择分支。



选择分支

初始步（InitStep）活动时，如果转换 Trans4 为真且 Trans1 为假，则激活 Step4，或者初始步活动，Trans1 为真则激活 Step2。

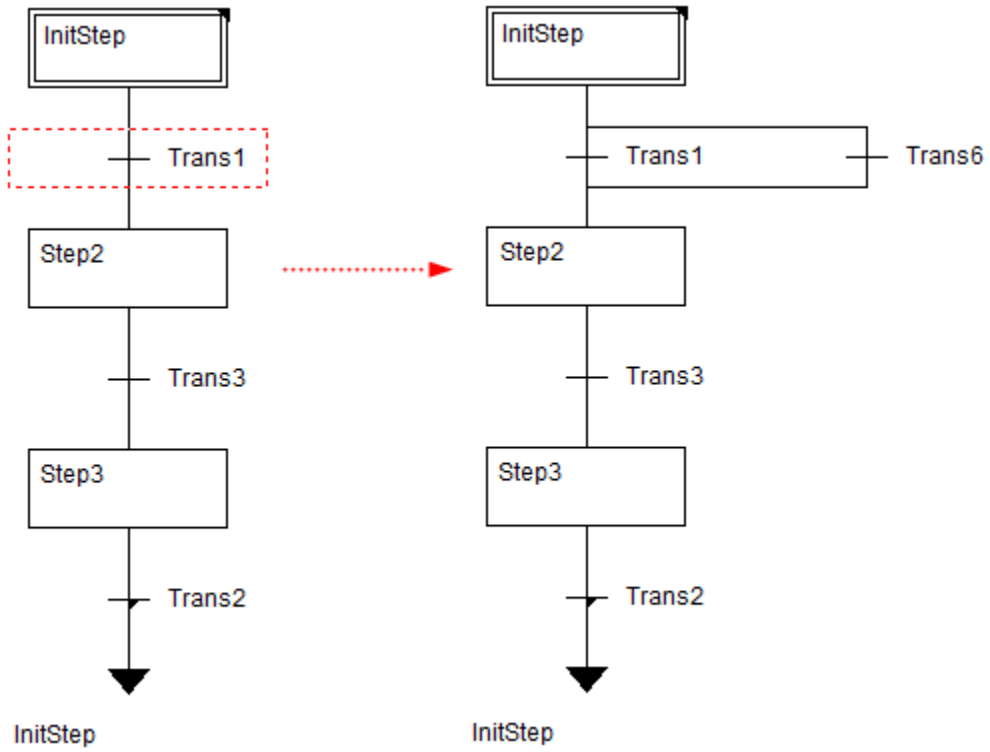
1. 增加选择分支

选中单个转换或者一个以转换开始并且以转换结束的单元，通过以下方式添加向右或向左的选择分支：

- 工具栏：  、  ；
- 程序区：右击选中的步或单元，单击【选择分支（右）/（左）】。

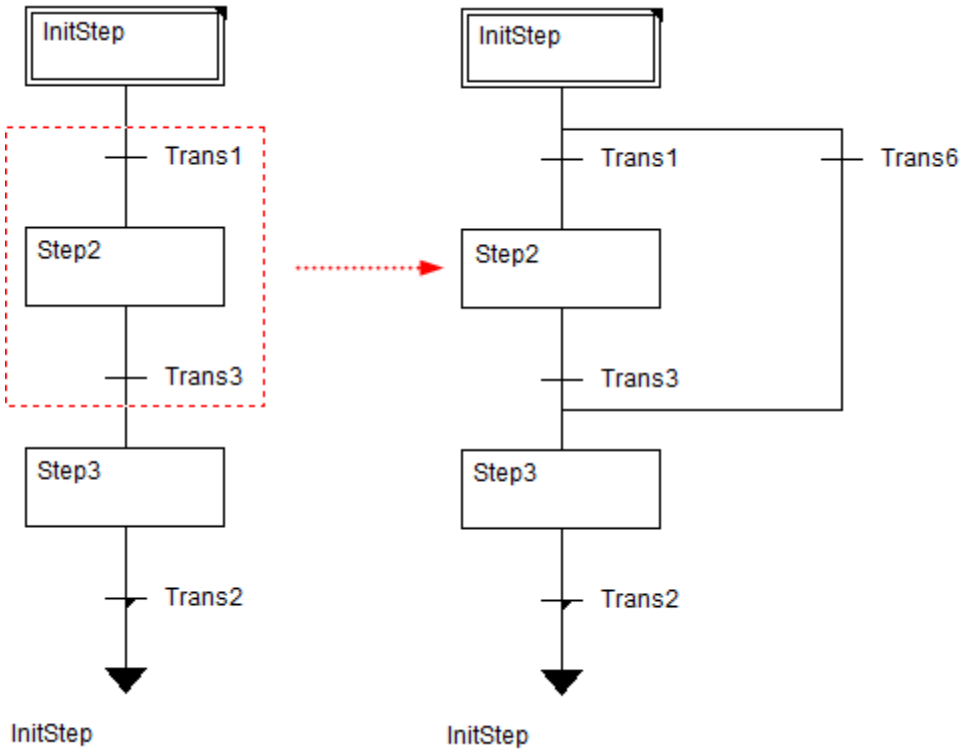
下面仅介绍向右增加选择分支，向左增加类似。

选中单个转换“Trans1”，增加选择分支前后如下图所示。



单个转换增加选择分支

选中一个以转换为起始并且以转换为结束的单元，如果是复选需要按住 **Ctrl** 键或者 **Shift** 键，执行增加选择分支前后对比如下图所示。



单元增加选择分支

如果分支的转换条件（如图中的 Trans1 和 Trans6）同时满足，则程序执行的优先级是从左到右。

2. 删除选择分支

选中选择分支的所有元素，执行【删除】命令。

8.9.22 跳转元素

SFC 程序的转移，通过该元素使 SFC 实现循环，SFC 语言编辑的 POU 至少包含一个跳转。

跳转是一种转移方法，可以从某一步直接转移到当前分支内部或外部的其他步。跳转用来代替一个步，并且由它所在位置的转换来执行，转换条件满足时，跳转才执行。

跳转元素

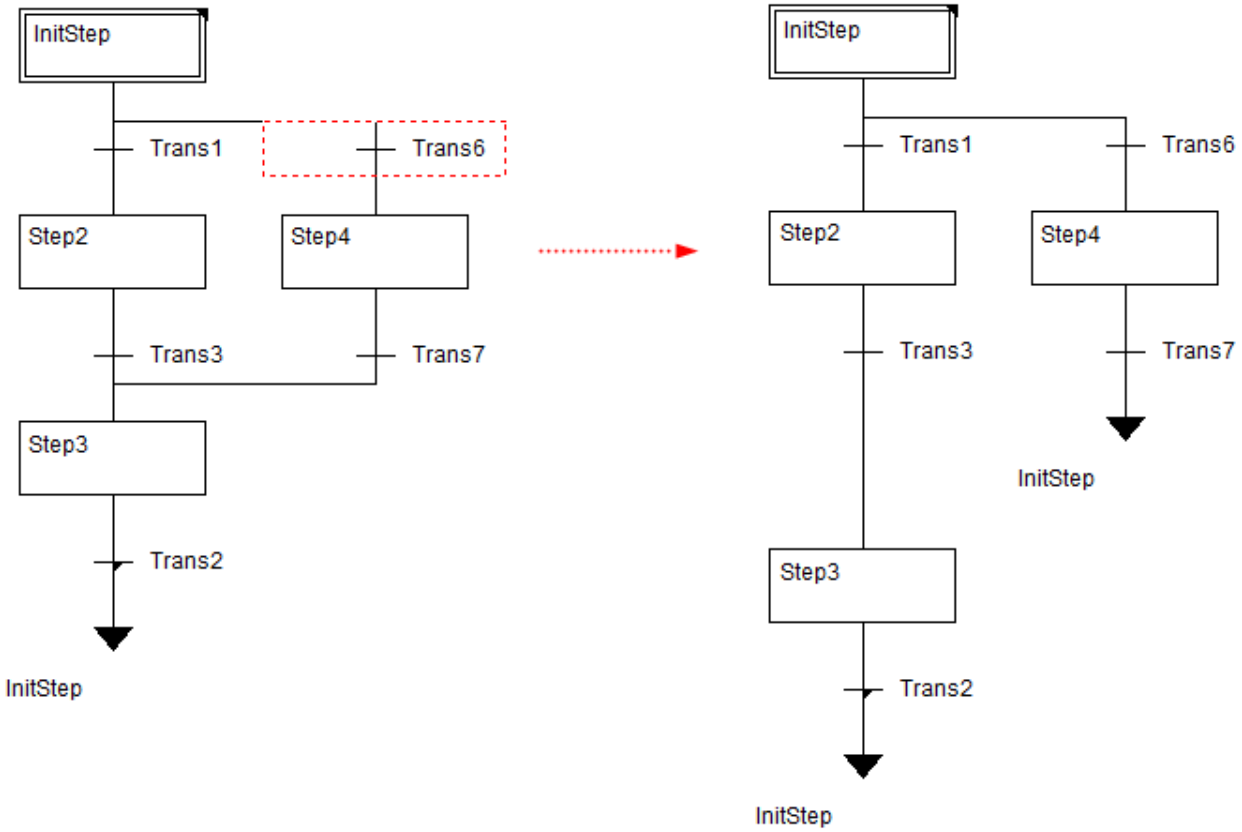
8.9.22.1 增加跳转

只能在选择分支和并行分支上插入跳转元素。跳转元素的跳转位置必须是步名或者并行分支标号名。

选中选择分支或并行分支上的一个元素，通过以下方式增加跳转：

- 程序区：右击分支上的步或转换，单击【跳转】；
- 工具栏：  ；
- 快捷键：**Ctrl+J**。

如图所示，在选择分支上插入跳转，跳转位置标示在跳转元素的左下角，默认显示 **InitStep** 表示默认跳转到初始步，选中跳转元素，双击跳转标号，可以写入需跳转到的步名或分支名称。

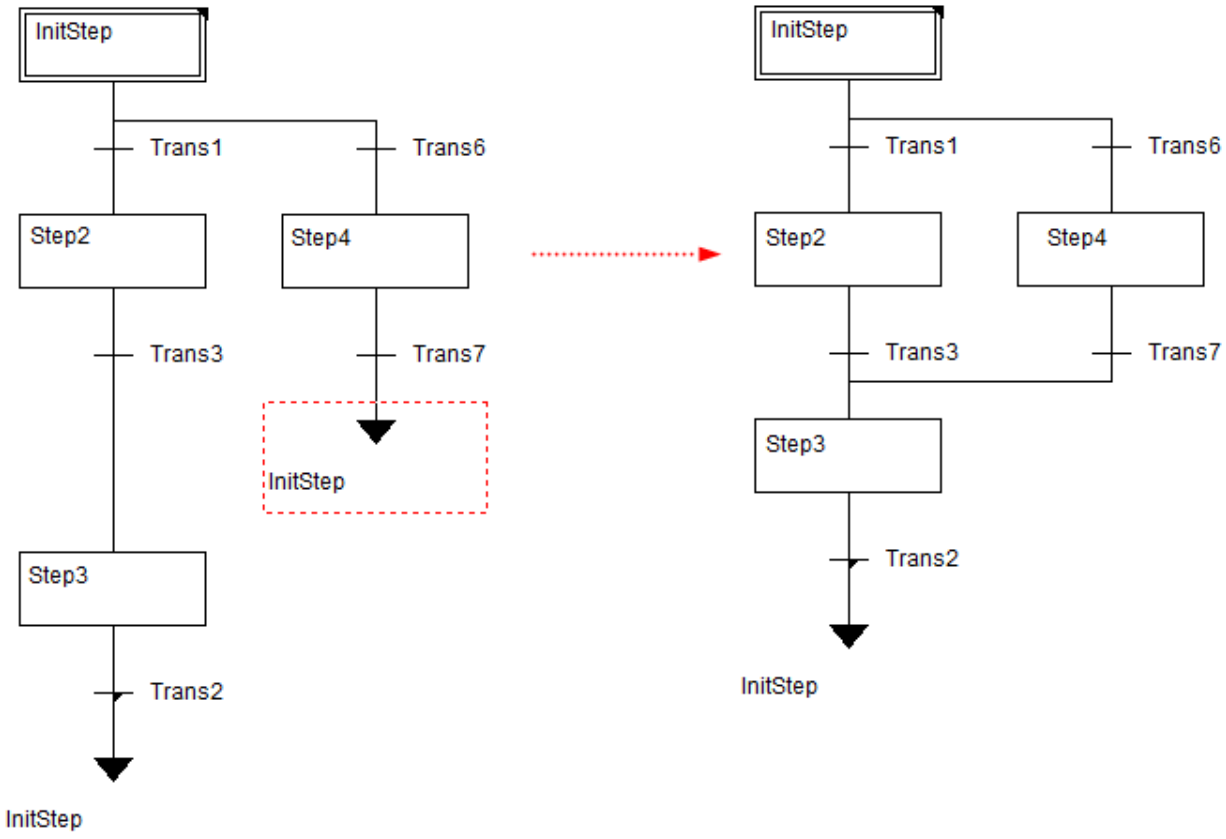


选择分支上插入跳转

“跳转”必须接在“转换”之后，所以在“步”后插入跳转时，选择【转换-跳转】命令，而在转换后可以直接插入“跳转”。

8.9.22.2 删除跳转

选中跳转分支，执行【删除】命令，如图所示。



删除跳转

8.9.23 转换跳转

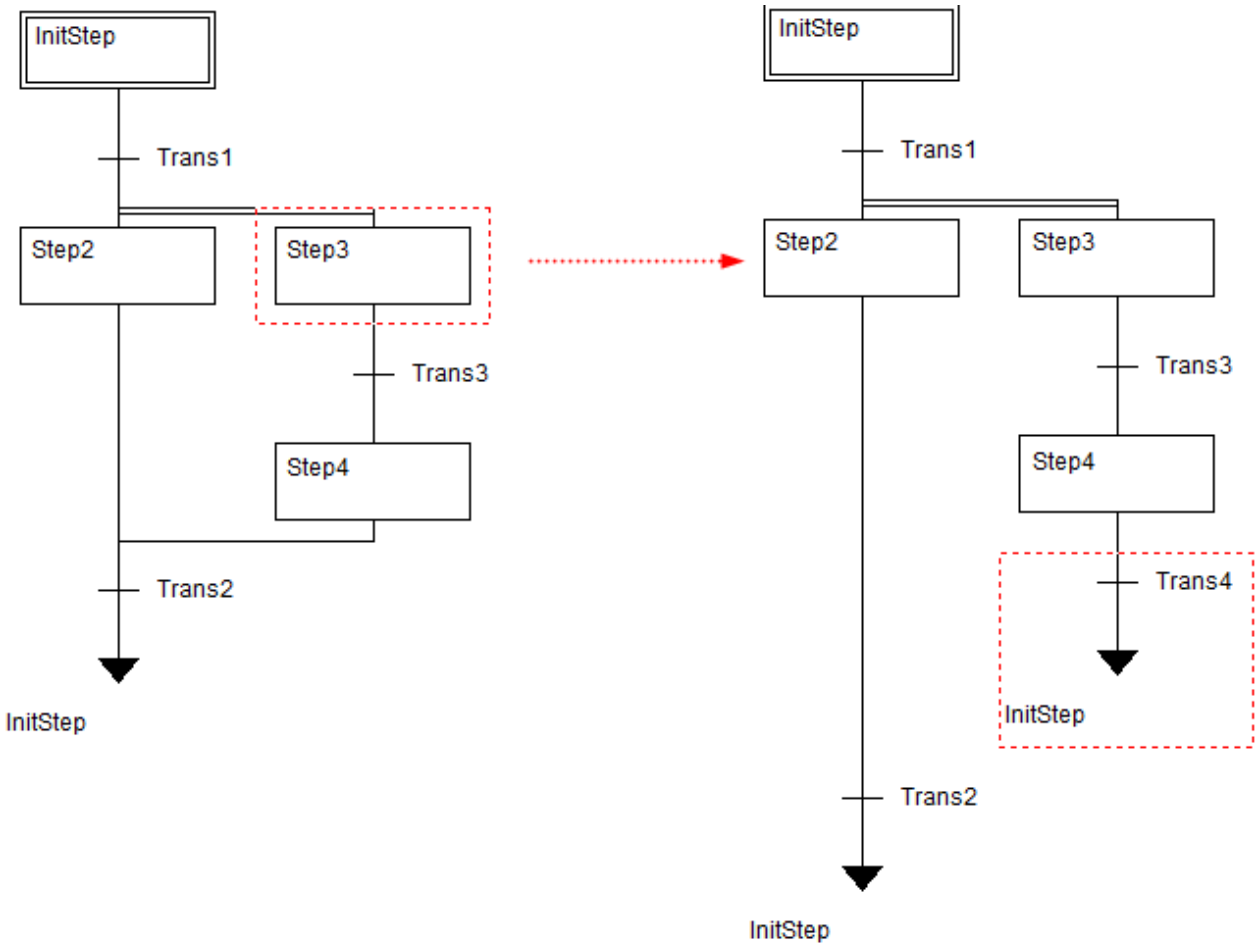
只能在并行分支上插入转换—跳转元素。跳转元素的跳转位置必须是步名或者并行分支标号名。转换—跳转元素插入到步之后。

1. 增加转换—跳转

选中并行分支上的一个元素，通过以下方式增加转换—跳转：

- 程序区：右击并行分支上的步或转换，单击【转换—跳转（R）】；
- 工具栏：  ；
- 快捷键：**Ctrl+R**。

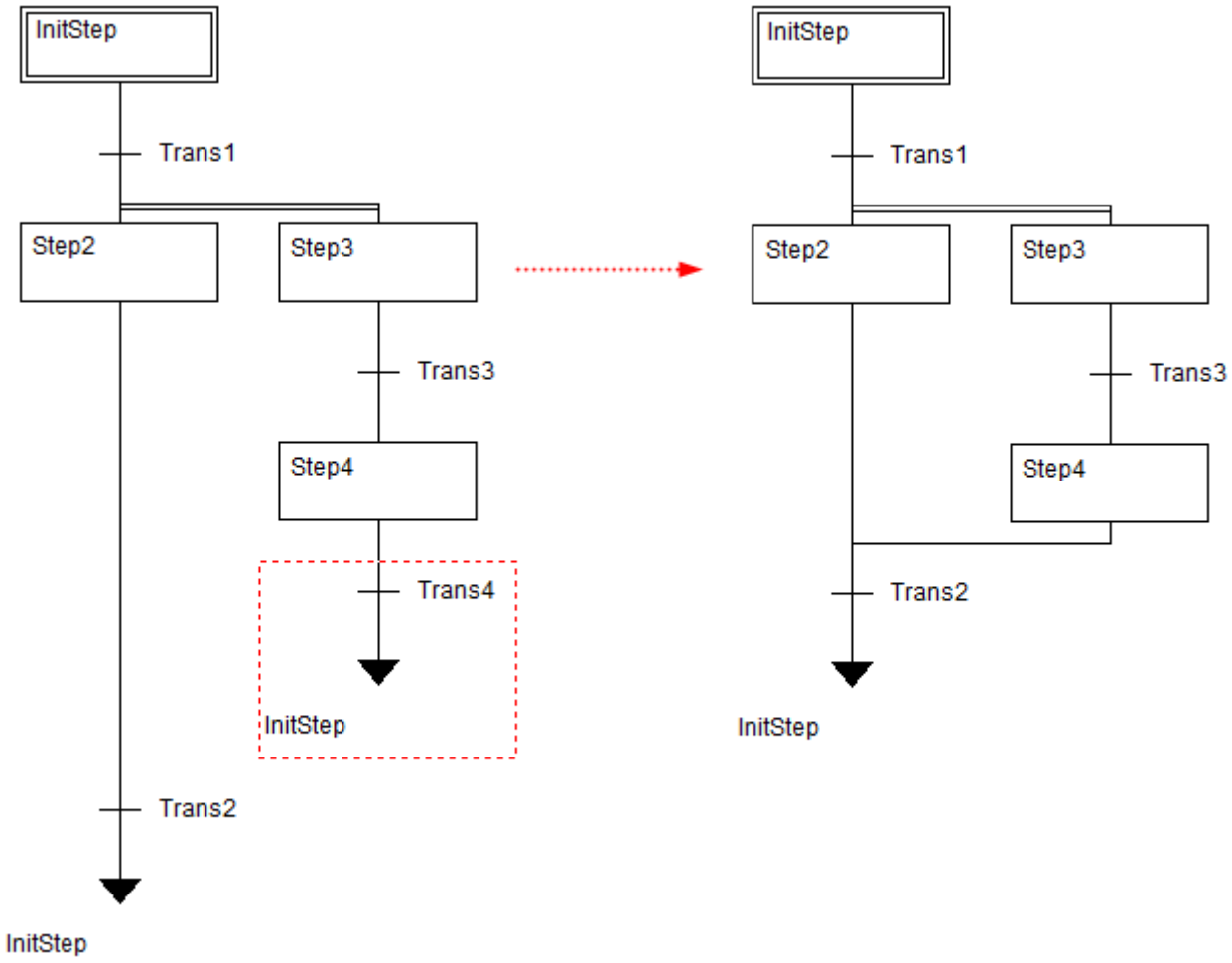
如图所示，跳转位置标示在转换—跳转元素的左下角，默认显示 **InitStep**，表示默认跳转到初始步，选中跳转元素，双击跳转标号，可以写入需跳转到的步名或分支名称。



并行分支插入转换—跳转

2. 删除转换—跳转

按下 **Shift** 键，选中转换—跳转，执行【删除】命令，转换—跳转被删除。如图所示。

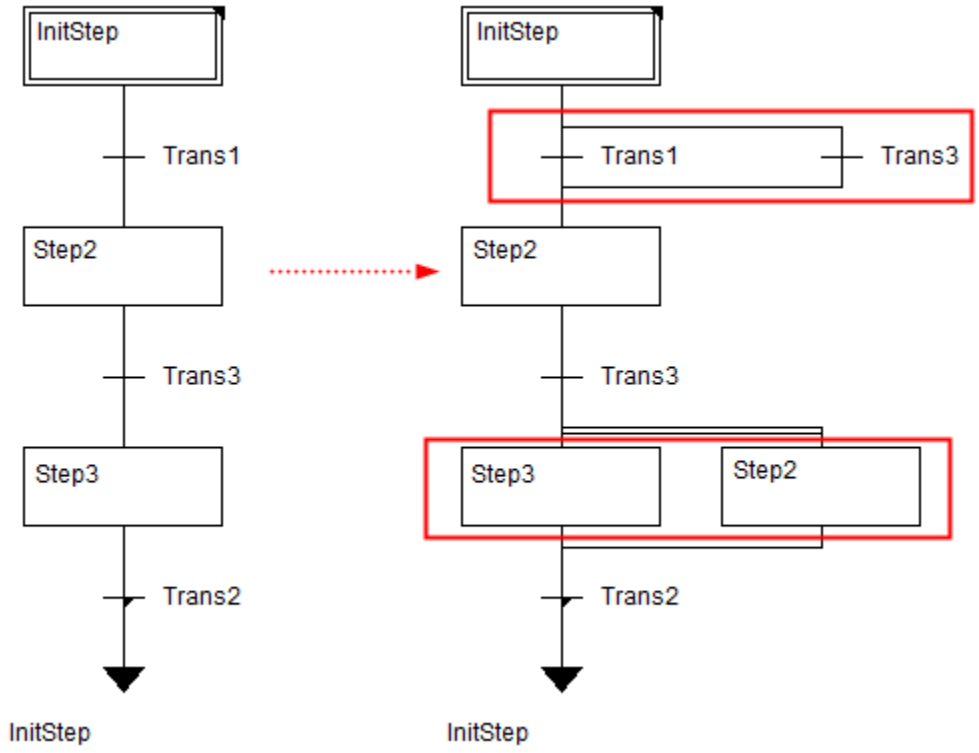


删除转换—跳转

8.9.24 粘贴到右边

将复制的单个步或转换，粘贴到某一步或转换的右边。

- 程序区：右击步或转换，单击【粘贴到右边】；
- 快捷键：**Ctrl+V**。



粘贴到右边

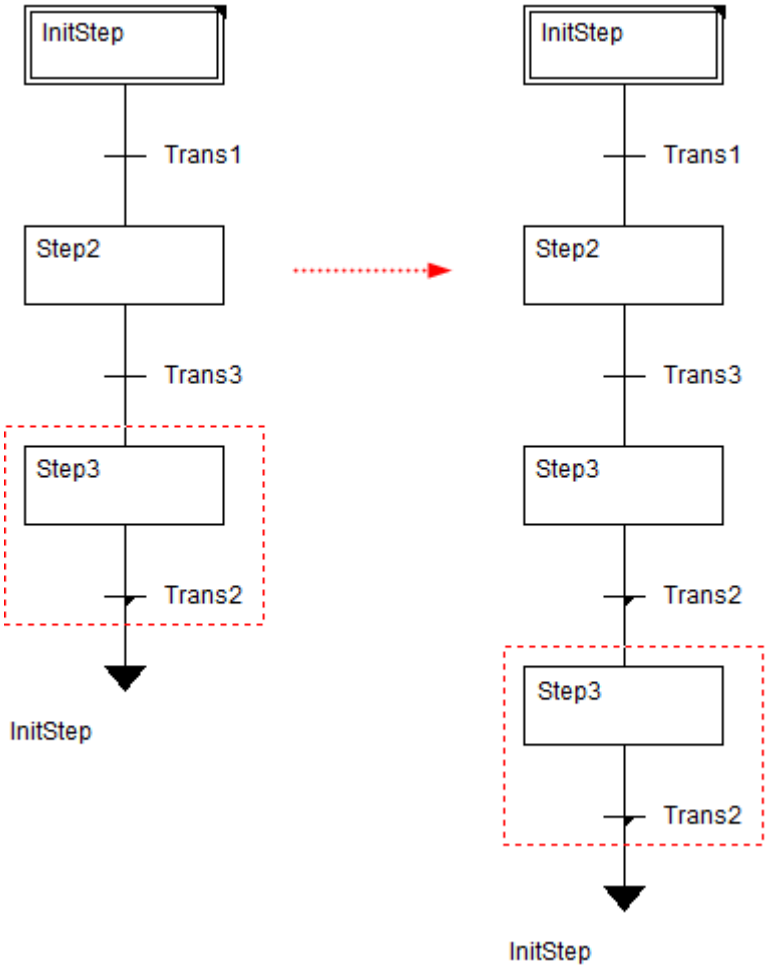
8.9.25 粘贴到后面

将复制的步-转换对，粘贴到某一转换的后面。

选中转换：

- 程序区：右击转换，单击【粘贴到后面】；
- 快捷键：**Ctrl+V**。

粘贴完成如下图所示。



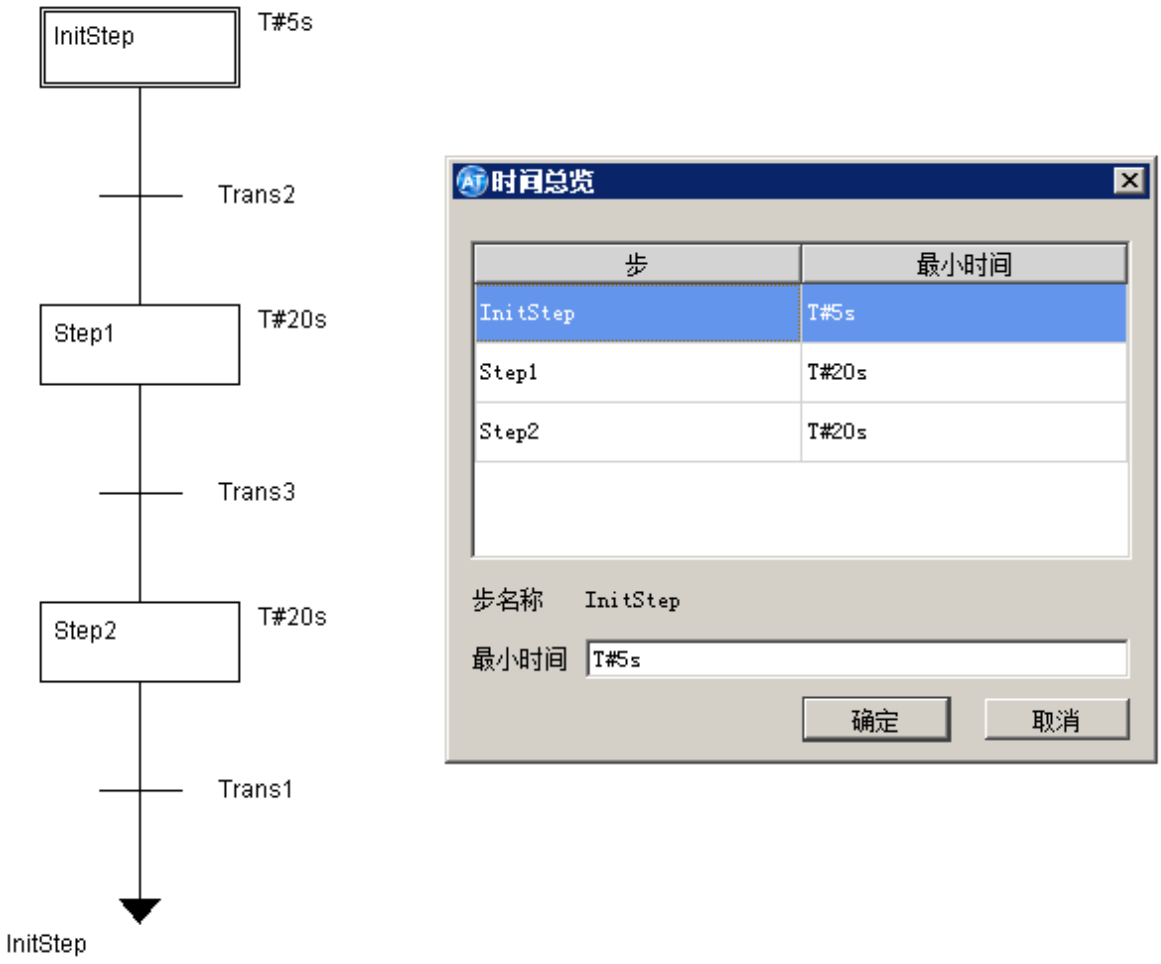
粘贴到后面

8.9.26 时间总览

通过以下方式打开：

- 程序区：右击空白区域，单击【时间总览（T）】。

在该窗口中，可以查看并修改本 SFC（PRG）程序页中的所有步时间的设置，如下图所示。

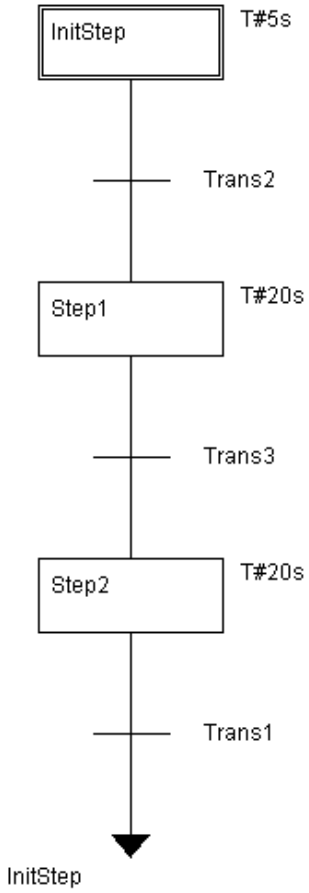


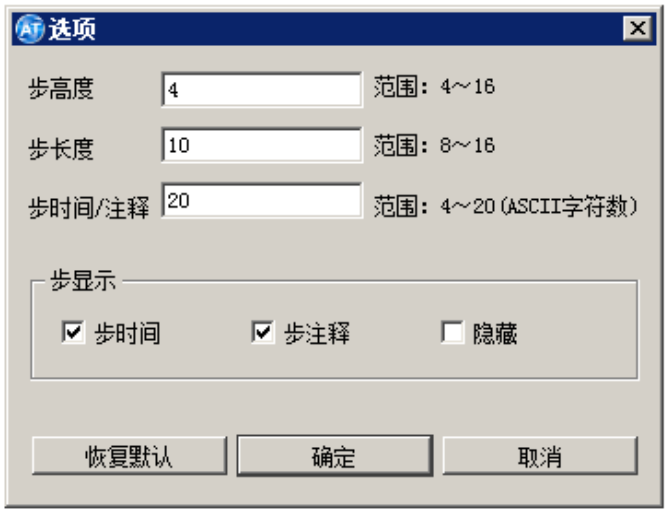
时间总览

8.9.27 选项

- 程序区：右击空白区域，单击【选项（O）】。

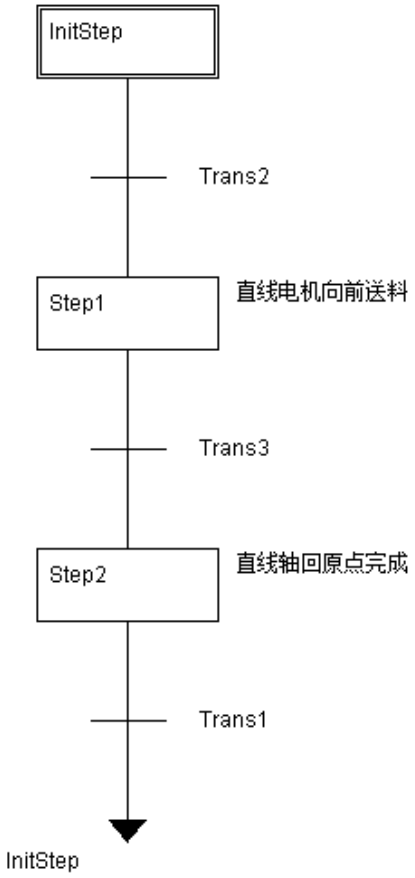
在该窗口中，可以设置步显示（标识框）的高度、长度、注释宽度等相关内容，如下图所示。

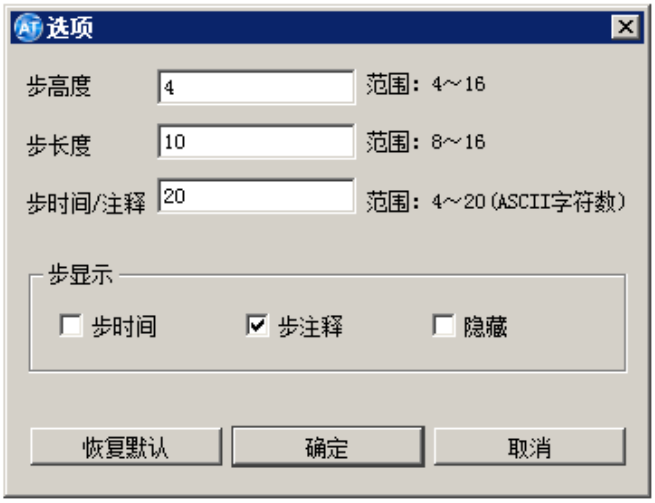




“选项”窗口

在【步的属性】中设置步的**最小时间**和**注释**后，在“选项“对话框的**步显示**中可以选择步的显示内容，缺省情况下，选择显示**步时间**；在需要显示每步的注释时，可以选择**步注释**，或者**隐藏**显示内容。如下图所示。





选择“步注释”

恢复默认：选择该按钮，可以将选项内容设置的当前值恢复为缺省值。

8.10 编辑

8.10.1 查找

在当前 POU 或所有 POU 中查找某一文本，通过设置搜索选项和搜索范围，进行精确查找。

8.10.1.1 要求

- 至少打开一个 POU 窗口
- “窗口”菜单中，已勾选“查找结果”命令

8.10.1.2 步骤

要查找某一文本，请按以下步骤操作：

（1） 在“编辑”菜单中，选择“查找”命令或单击工具栏按钮  。

将弹出“查找”对话框。

（2） 输入查找内容

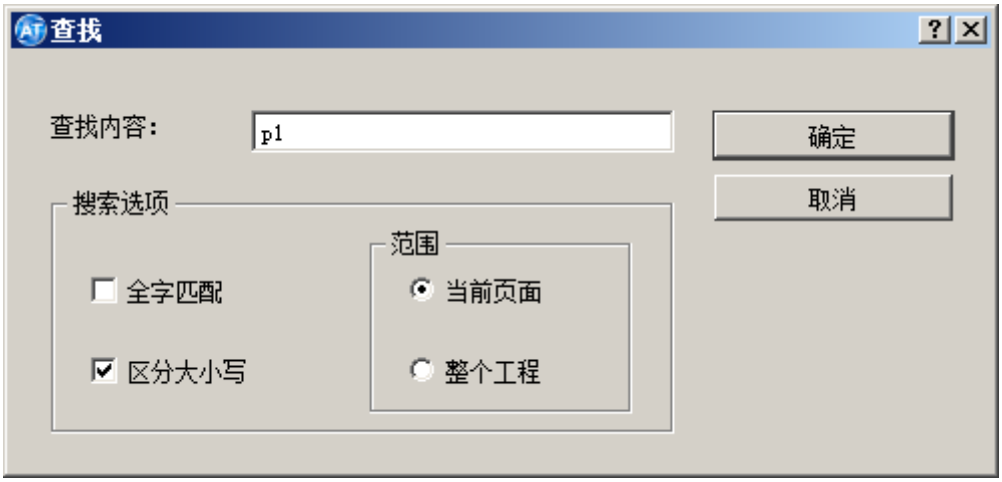
支持中英文字符。

（3） 勾选搜索选项。

- 全字匹配：进行精确查找，POU 中的文本必须与查找文本完全匹配，才能被检索到。
- 区分大小写：如果输入文本为英文字符，将区分字母大小写。
- 当前页面：在当前打开的视图窗口中搜索。
- 整个工程：在整个工程中搜索。

（4） 单击**确定**按钮

开始搜索。



设置查找参数

8.10.1.3 结果

在信息输出窗口的“查找结果”标签中显示查找结果，双击结果行，将自动定位到该文本区域。

8.10.2 替换

在当前 POU 或所有 POU 中查找某一文本，并将其替换为其他文本。

8.10.2.1 要求

至少打开一个 POU 窗口

8.10.2.2 步骤

要替换某一文本，请按以下步骤操作：

（1） 在“编辑”菜单中，选择“替换”命令。

将弹出“替换”对话框。

（2） 输入查找内容。

支持中英文字符。

（3） 输入替换文本。

（4） 勾选搜索选项。

- 全字匹配：进行精确查找，POU 中的文本必须与查找文本完全匹配，才能被检索到。
- 区分大小写：如果输入文本为英文字符，将区分字母大小写。
- 当前页面：在当前打开的视图窗口中搜索。
- 整个工程：在整个工程中搜索。

（5） 单击**全部替换**按钮.

所有查找文本都被替换为新文本。

如果仅替换其中一部分内容，请执行（6）~（7）步。

（6） 单击**查找下一个**按钮。

开始搜索，查找到的文本呈蓝色高亮选中状态。

（7） 单击**替换**按钮。

当前选中文本被替换为新文本，同时自动定位到下一个查找内容处。

继续替换，请重复执行上面第(6)步。

如全部替换完成，将弹出提示框。

（8） 单击**确定**按钮。

设置替换参数

8.10.3 撤销

如果想要取消程序编辑区中最近执行的一系列动作，请重复执行撤销命令。撤销操作仅对 POU 窗口打开后，执行的编辑操作有效，执行动作能被撤回到窗口打开那一次。

可撤销的操作包括复制粘贴、剪切、删除、替换、新增程序等。

8.10.3.1 要求

POU 窗口已打开，程序区有相关编辑操作

8.10.3.2 步骤

要执行撤销命令，请按以下步骤操作：

（1） 选择以下一种方方式，重复执行

- 菜单栏：“编辑”—“撤销”；
- 工具栏：  ；
- 快捷键：**Ctrl+Z**。

8.10.3.3 参考信息

恢复

8.10.4 恢复

在程序编辑区，撤销之前的一系列编辑操作后，如果想要恢复这些操作，请重复执行恢复命令。程序将被恢复到执行撤销命令前的状态。

8.10.4.1 要求

POU 窗口已打开，程序区已执行撤销操作

8.10.4.2 步骤

要执行恢复命令，请按以下步骤操作：

（1） 选择以下一种方方式，重复执行

- 菜单栏：“编辑”—“恢复”；
- 工具栏：  ；
- 快捷键：**Ctrl+Y**。

8.10.4.3 参考信息

撤销

第9章　编译工程

9.1 编译概述

工程下装前，需要对其进行编译，检查语法错误、并生成控制器可执行的二进制文件。编译通过后，在安装路径下生成.at 文件，用于控制器下装。

按照编译范围，编译分为全编译和增量编译。

- 全编译：将整个工程重新编译，生成新的组态逻辑信息和硬件配置信息。仅在首次执行“编译”操作、手动执行“全编译”操作或硬件配置修改时触发。
- 增量编译：只对修改和新增的内容进行编译，生成增量编译信息。首次编译通过后，除硬件配置修改会触发全编译外，其他所有工程修改触发的“编译”操作，均执行增量编译。

9.2 全编译

9.2.1 概述

全编译仅在工程首次执行“编译”操作或手动执行“全编译”操作时触发。

9.2.2 步骤

要执行全编译，请按以下步骤操作：

（1） 在“工程”菜单中，选择“全编译”命令。

如果是首次编译工程，选择“编译”命令，也会执行全编译。

9.2.3 结果

在“语法检查”窗口显示“FULL_COMPILE”编译完成、并报告编译错误和警告信息。

9.3 增量编译

9.3.1 概述

工程首次编译后，如果修改工程组态信息，执行“编译”操作，将触发增量编译。

9.3.2 要求

当前工程已首次编译通过

9.3.3 步骤

要执行增量编译，请按以下步骤操作：

（1） 在“工程”菜单中，选择“编译”命令。

9.3.4 结果

在“语法检查”窗口显示“ADD_COMPILE”编译完成、并报告编译错误和警告信息。

9.4 检查编译错误

9.4.1 概述

在信息输出窗口的“语法检查”标签页中，可查看编译是否成功、是否有编译错误信息。如发生错误，请修改后，重新编译。

9.4.2 要求

“窗口”菜单中，已勾选“语法检查”命令

9.4.3 检查错误

要检查编译错误，请按以下步骤操作：

（1） 在“语法检查”标签页，双击错误信息行。

光标自动定位到程序错误位置，对应元素呈选中状态。

（2） 修改错误。

（3） 重新进行编译。

9.4.4 清除信息

要清除信息，请按以下步骤操作

（1） 右击标签页空白区域。

（2） 在右键菜单中，选择“清除”命令。

所有的编译信息被删除。

第10章　下装

10.1 设置通讯参数

10.1.1 概述

下装前，需要建立本地计算机和控制器间的通讯连接。你需要分别设置本地计算机和控制器的通讯参数。

10.1.2 要求

本地计算机与控制器间通讯线连接正常

10.1.3 设置控制器通讯参数

要设置控制器通讯参数，请按以下步骤操作：

（1） 在“在线”菜单栏中，选择“通讯设置”。

“通讯设置”对话框被打开。

（2） 在 IP 地址栏中输入当前控制器 IP 地址。

控制器两个网口默认的 IP 地址分别为 192.168.0.1、192.168.1.1。

（3） 设置超时时间，区间为 20000~90000ms。

超时时间为通讯建立后，控制器没有响应的最大时间限值。超时时间内控制器没有响应，则在“输出窗口”提示通讯超时。

（4） 单击**确定**按钮。

控制器通讯参数设置完成。

10.1.4 设置本地计算机通讯参数

要设置本地计算机通讯参数，请按以下步骤操作：

（1） 在“本地连接状态”对话框中，单击**属性**按钮。

“本地连接属性”对话框被打开。

（2） 在项目列表框中双击“Internet 协议版本 4（TCP/IPv4）”。

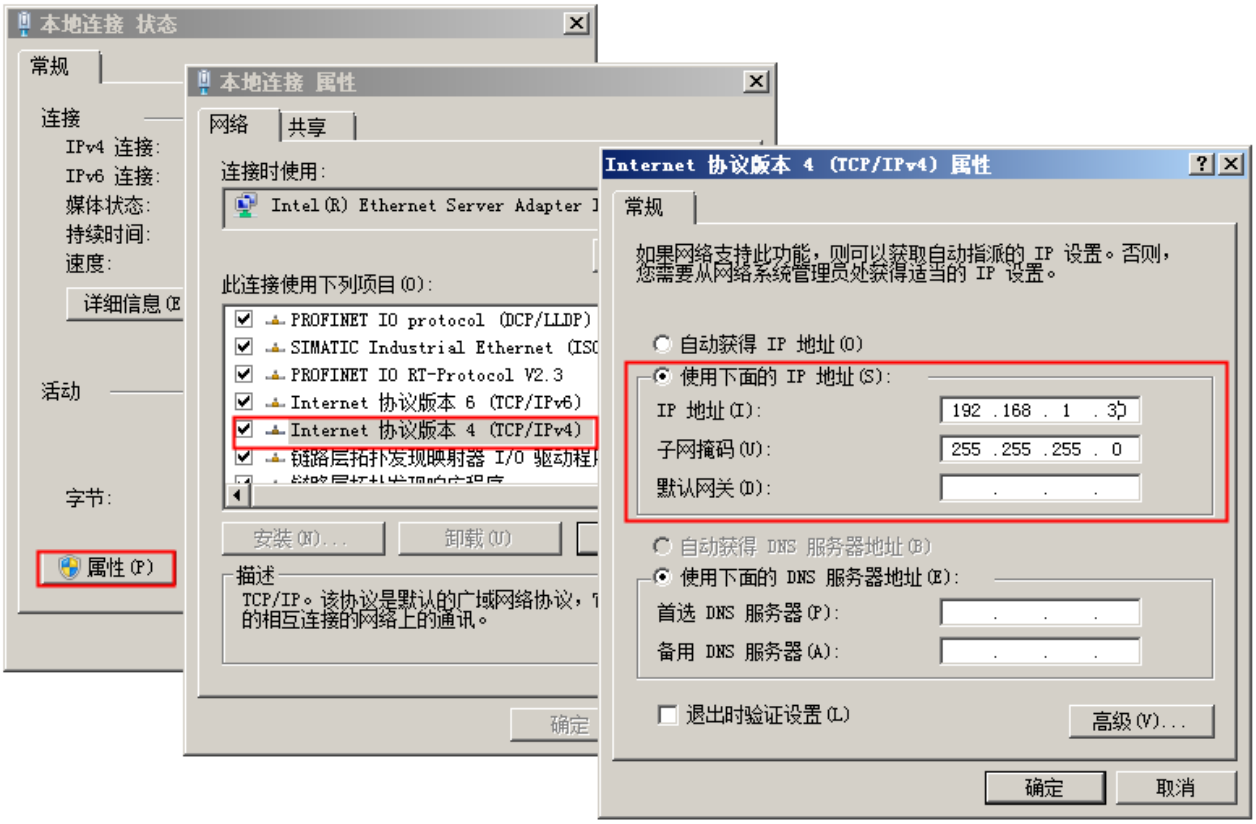
“Internet 协议版本 4（TCP/IPv4）属性”对话框被打开。

（3） 在 IP 地址框中输入本地 IP 地址，设置时，请与控制器 IP 地址保持在同一网段内。

（4） 设置子网掩码。

IP 地址第一字段 192 对应的子网掩码为 255.255.255.x。

（5） 单击**确定**按钮。



设置本地计算机通讯参数

10.1.5 结果

下装、在线时，本地计算机与控制器自动建立连接，并传输数据。

10.2 全下装

10.2.1 概述

全下装即初始化下装，将全部组态逻辑信息和硬件配置信息下装到控制器。下装时，所有变量恢复为初始值。

以下情况，执行下装操作时，将触发全下装：

- 工程首次下装。
- 当前工程与控制器工程不一致或版本不连续。
- 执行全编译操作。

10.2.2 要求

- 控制器钥匙开关拨到 PRG 或者 REM
- 工程已编译通过
- 通讯参数已设置完成

10.2.3 步骤

要下装工程，请按以下步骤操作：

（1） 在“在线”菜单中，选择“下装”命令

将弹出下装确认对话框。

（2） 单击**确认**按钮。

开始下装，并显示下装进度。


下装过程中不允许操作钥匙开关。

10.2.4 结果

下装完成后，输出窗口提示全下装成功。需要将钥匙开关拨到 RUN，运行工程。

如果钥匙开关在 REM 位置，下装前，工程处于运行状态，则全下装后工程自动运行。

10.3 增量下装

10.3.1 概述

增量下装即部分下装，仅仅将修改或新增的逻辑程序或变量下装到控制器。增量下装时，未发生变化的变量值保持不变，仅新增变量初始化。

增量下装由增量编译触发。

10.3.2 要求

- 控制器钥匙开关拨到 PRG 或者 REM
- 工程已编译通过
- 通讯参数已设置完成

10.3.3 步骤

要下装工程，请按以下步骤操作：

（1） 在“在线”菜单中，选择“下装”命令。

将弹出“待初始化新增变量”对话框。

（2） 单击**确认**按钮。

开始下装，并显示下装进度。

下装过程中不允许操作钥匙开关。

10.3.4 结果

下装完成后，输出窗口提示增量下装成功。需要将钥匙开关拨到 RUN，运行工程。

如果钥匙开关在 REM 位置，下装前，工程处于运行状态，则下装后工程自动运行。

10.4 检查通讯异常

下装时，“输出窗口”提示通讯异常信息，原因及处理措施详见下表。

异常信息及处理措施

通讯异常信息	原因	处理措施
提示“请确认以太网连接状态正常！”	通讯连接失败	1、检查通讯线缆是否正常连接 2、检查通讯参数是否设置正确
提示“通讯超时！”	通讯过程中，控制器无响应	1、请检查控制器指示灯、诊断信息，确认控制器是否发生错误 2、稍作等待后，请重新执行下装操作
提示“没有权限进行该操作！”	下装时，控制器钥匙开关处于 RUN 位置	1、重启控制器，并将钥匙开关拨到 PRG 或 REM 位置 2、重新执行下装操作
提示“已经有一个客户端与 RTS 连接！退出本次登录”	下装时，辅助工具与控制器已建立在线连接	小工具退出在线监视

10.5 多路连接

多个 AT 软件实例打开同一个工程时，可以同时连接控制器，监视工程，最多支持 5 路连接。

第一路连接为读写权限，第二路连接以及后续连接均为只读权限。当第一路连接断开时，其他路连接均保持连接且为只读权限。第一路读写权限连接断开，再进行连接时为受限权限。

基本特征：

- 只读权限不支持在线操作
- 读写权限时默认权限，支持所有操作
- 受限权限不支持增量下装/全下装、下载用户工程、清空控制器操作

第11章　在线调试

11.1 调试任务

11.1.1 在线监视

11.1.1.1 概述

进入在线，AutoThinkV4 软件与控制器建立数据通信，可以实时监视变量在线值、逻辑输出结果、任务运行状态及诊断信息等。

进入在线监视时，任务的运行和停止状态由控制器钥匙开关位置决定：

- 运行
  - 控制器钥匙开关在 RUN 位置，控制器任务运行。
  - 控制器钥匙开关在 REM 位置，进入在线后，通过 AutoThinkV4 软件控制用户任务运行。

    注意，钥匙开关从 RUN 或者 PRG 位置拨到 REM 后，任务运行状态保持之前模式的运行状态。
- 停止：
  - 控制器钥匙开关在 PRG 位置，控制器工程停止运行。进入在线后，任务状态为停止。
  - 控制器钥匙开关在 REM 位置，进入在线后，通过 AutoThinkV4 软件控制用户任务停止。

11.1.1.2 要求

工程已下装到控制器

11.1.1.3 启动在线

要启动在线监视，请按以下步骤操作：

（1） “在线”菜单中，选择“在线”命令。

输出窗口显示“进入监视状态！”

11.1.1.4 退出在线

要退出在线监视，请按以下步骤操作：

（1） 在“在线”菜单中，选择“退出在线”命令。

输出窗口显示“退出监视成功！退出本次登录！”此时，可编辑或修改控制器工程。

11.1.2 PRG 编程模式

在 PRG 编程模式，控制器任务停止运行，仅可编辑用户程序，无法调试和运行控制器工程。当您进行工程组态时，可将控制器模式置为 PRG。

PRG 模式，操作权限如下：

- 创建、修改和删除任务、程序。
- 下装用户工程。
- 复位和清除操作。
- 无法通过 FA-AT 编程软件运行控制器工程

11.1.3 REM 调试模式

11.1.3.1 概述

工程调试时或者控制器运行期间，需要修改用户程序、强制或写入变量值等操作，需要将控制器钥匙开关拨到 REM 调试模式。

- 从 RUN 模式拨到 REM 模式，用户程序保持继续运行；
- 从 PRG 模式拨到 REM 模式，用户程序保持原停止状态；
- 如果控制器启动前钥匙位置在 REM，系统程序启动完成后，运行状态与断电或复位前保持一致。

REM 模式，操作权限如下：

- 通过 FA-AT 编程软件控制工程的运行和停止。
- 强制变量。
- 写变量。
- 复位和清除操作。
- 下装用户工程。

11.1.3.2 参考信息

PRG 编程模式

RUN 运行模式

11.1.4 RUN 运行模式

工程调试完毕后，需要将控制器模式置为 RUN 模式。在运行模式，控制器扫描 I/O 数据和执行用户任务，只能监视在线数据，不允许操作。

- 不能修改和下装控制器工程。
- 不能强制。
- 不能写变量
- 不能进行复位和清除操作。
- 不能通过 FA-AT 编程软件停止控制器工程。

11.1.5 运行任务

11.1.5.1 概述

在调试模式（REM），通过 AutoThinkV4 软件启动控制器任务运行。

11.1.5.2 要求

- 控制器钥匙开关在 REM 位置
- AutoThinkV4 软件在监视状态
- 当前任务处于停止状态

11.1.5.3 步骤

要运行控制器任务，请按以下步骤操作：

（1） 在“在线”菜单中，选择“运行”命令或单击工具栏按钮  。

“输出窗口”显示开始运行。

11.1.5.4 结果

在调试模式，控制器工程开始运行，通过全局变量组 TaskDiagGroup 查看任务运行状态为运行。

11.1.6 停止任务

11.1.6.1 概述

在调试模式（REM），通过 AutoThinkV4 软件停止控制器任务运行。

11.1.6.2 要求

- 控制器钥匙开关在 REM 位置
- AutoThinkV4 软件在监视状态

- 当前任务处于运行状态

11.1.6.3 步骤

要停止控制器任务，请按以下步骤操作：

（1） 在“在线”菜单中，选择“停止”命令或单击工具栏按钮  。

“输出窗口”显示停止运行。

11.1.6.4 结果

在调试模式，控制器工程停止运行，通过全局变量组 TaskDiagGroup 查看任务运行状态为停止。

11.1.7 查看任务运行状态

11.1.7.1 概述

任务运行状态、运行模式、运行次数及运行时间诊断信息通过 S 区变量上报，可通过在线值，查看任务当前状态信息。

变量	变量说明	在线值
RunStatus	任务运行状态	0：当前任务正在运行 1：当前任务已停止 2：调试状态 3~5：保留 6：看门狗超时致任务停止运行 7：停止 IEC 任务错误 当 RunStatus 和 RunMod 的值为 0xFFFF 时，表示 任务被删除
RunMod	任务运行模式	0：不处理 1：持续运行 2：挂起任务 3：PLC 退出 4：PLC 停止
RunCount	任务运行计数	
RunCycleTime	任务最近一次运行时间	
RunCycleTimeMin	任务历史运行时间最小值	
RunCycleTimeMax	任务历史运行时间最大值	

11.1.7.2 要求

AutoThinkV4 软件在监视状态

11.1.7.3 步骤

要查看任务运行状态，请按以下步骤操作：

（1） 在“全局变量”树节点下，双击 TaskDiagGroup 变量组。

在工作区，打开“TaskDiagGroup”窗口。

11.1.8 断点调试

11.1.8.1 启用断点调试模式

1. 概述

断点调试模式默认为启用状态，如当前工程关闭了调试模式，请在断点调试前，启用断点调试模式。启用后，断点调试功能可用，可在 POU 中设置调试断点。目前只能调试 ST 类型的 POU 断点。

调试模式下，通过单步执行、设置调试任务、断点对话框、跳出、跳进、断点运行等操作调试程序。以上操作在线状态时可用。

2. 要求

AutoThinkV4 软件在监视状态

3. 步骤

要启用断点调试模式，请按以下步骤操作：

（1） 单击【在线】菜单，勾选【调试模式】菜单项。

勾选后，【断点调试窗口】菜单项可用。

11.1.8.2 设置调试任务

1. 概述

调试前，首先要设置需要调试的任务，当前仅支持单任务，默认 TASK1 为调试任务。

2. 要求

AutoThinkV4 软件在监视状态

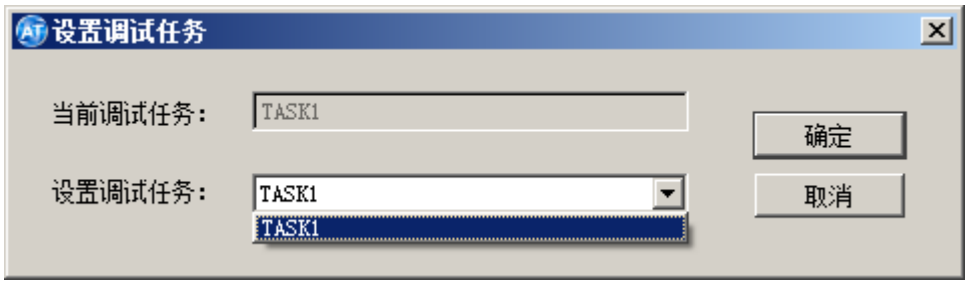
3. 步骤

要设置调试任务，请按以下步骤操作：

（1） 单击【在线】—【调试模式】菜单项。

（2） 单击“设置调试任务”下拉按钮，在任务列表中选择任务。

（3） 单击**确定**按钮。



调试任务设置窗口

11.1.8.3 断点设置

1. 概述

为需要调试的 POU 程序设置断点，离线或在线状态均可设置。在“断点调试”窗口中可以新建断点、删除断点、以及设置断点使能等。工程中最大可设置 1024 个断点。

“断点调试”窗口打开状态下，如果修改工程，则断点对话框变为不可编辑状态，您需要重新编译通过后，进行断点设置。

使能断点行显示红色实心圆，未使能断点行显示红色空心圆，如下图所示。

断点行显示

2. 要求

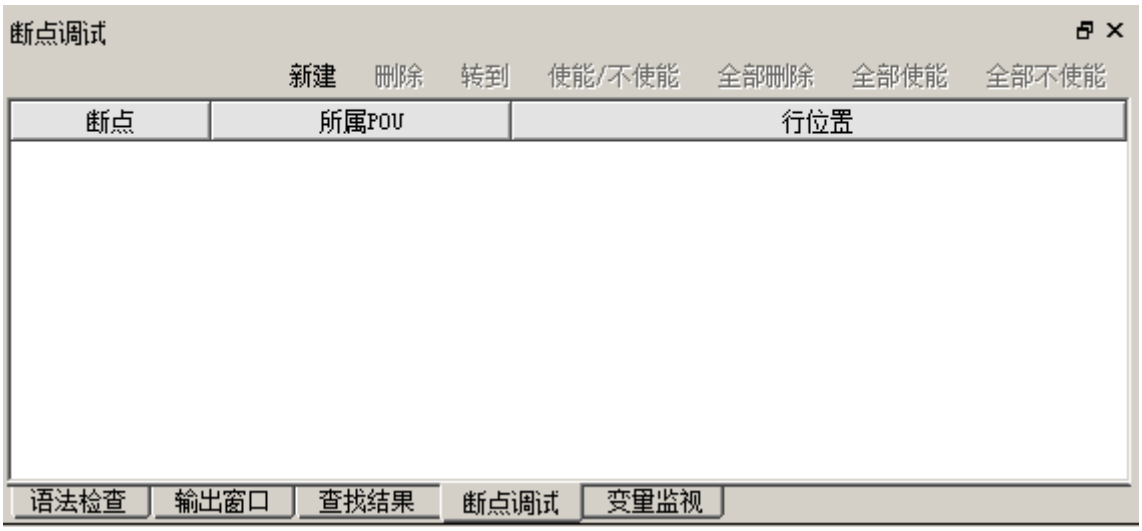
在线和离线状态均可设置。

3. 步骤

要设置断点，请按以下步骤操作：

（1） 单击【在线】菜单，勾选【断点调试窗口】。

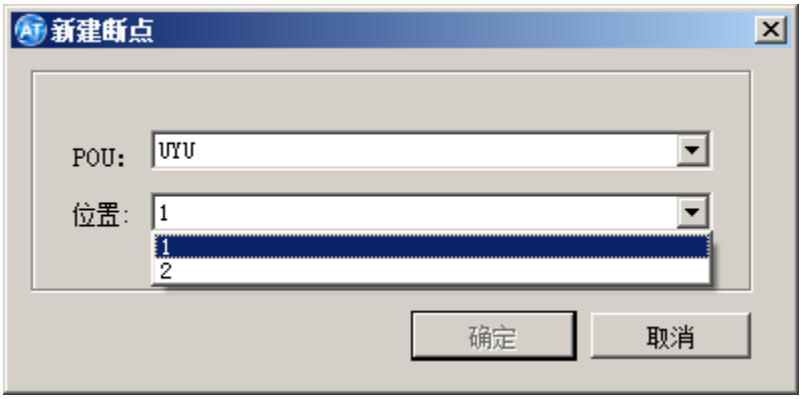
信息输出窗口栏将打开“断点调试”窗口。



断点调试窗口

（2） 单击**新建**按钮，添加断点。

- POU:POU 下拉列表将显示所有 ST 类型的 POU。
- 位置：位置下拉列表将显示 POU 中所有可打断点行。



（3） 选择 POU 和断点行后，单击**确定**按钮。

该语句行设置断点完成。

可通过窗口上方的按钮设置断点使能属性。

断点设置说明

窗口按钮	**说明**
新建	通过新建按钮，增加断点 在线监视状态下，单击某一语句行行号位置，为该行增加断点
删除	选中某一断点，单击**删除**，从“断点调试”窗口中删除该断点 在线监视状态下，单击某一断点行行号位置，将删除断点。“断点调试”窗口将同步更新断点状态
转到	选中某一断点，单击**转到**，则光标跳转到该断点行位置，并选中当前语句行

使能/不使能	使能：选中某一未使能断点，单击**使能/不使能**，将该断点使能。新建的断点默认使能 不使能：选中某一使能断点，单击**使能/不使能**，将该断点取消使能。不使能的断点不作为一个断点行，执行【断点运行】时，不会执行到该行
全部删除	单击**全部删除**，将删除“断点调试”窗口中所有断点
全部使能	对所有添加的断点全部使能
全部不使能	对所有添加的断点取消使能

11.1.8.4 单步执行

1. 概述

该命令用于在线模式下的程序步进调试。ST 程序中，【单步执行】以语句末尾的分号为步单位执行用户程序。

对于普通语句行（非函数、功能块或已调用的其他 ST 程序），【单步执行】与【跳进】的作用一致。

对于调用的函数、功能块、ST 语言 POU，如果该 POU 内部没有设置断点或设置的断点没有使能，则单步执行不会跳进这个 POU，而是把这个 POU 当作一个完整的步执行完。需要进入调用块中调试时，可通过【跳进】命令进入。

某一断点行【单步执行】的结果：

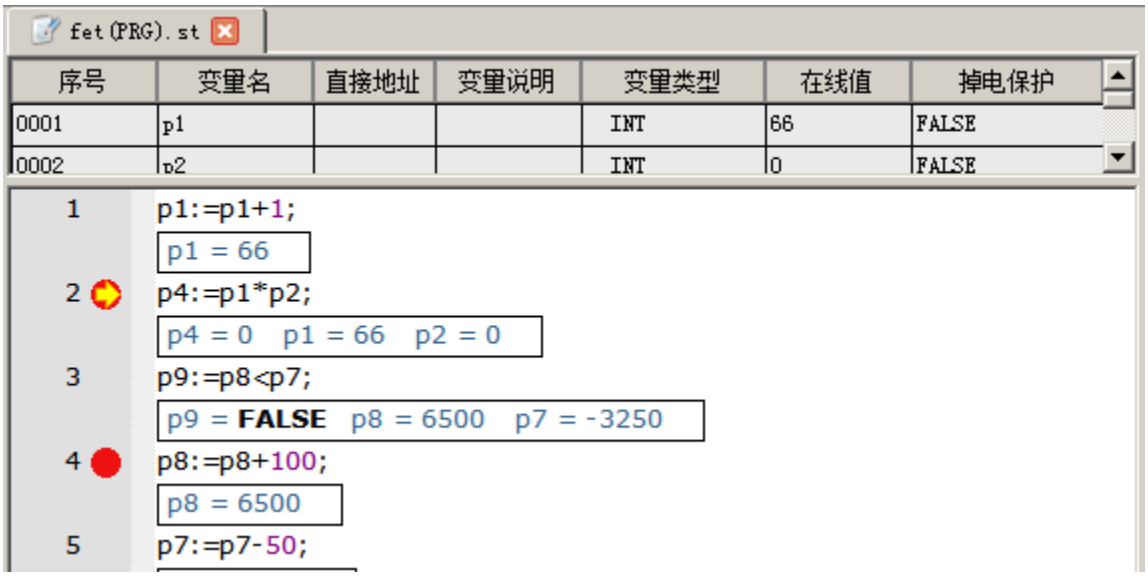
- 单击【单步执行】，程序执行完当前行，同时跳转到下一行停止。
- 当任务调用多个 PRG，在 POU 最后一行执行【单步执行】，则跳转到下一个 PRG 的断点处。如果任务下只有一个 POU，则跳转到本 POU 的第一步。
- 如果调用了函数、功能块或其他 ST 程序，执行【单步执行】操作，则会进入被调用程序内部的断点行。

通过以下方式执行：

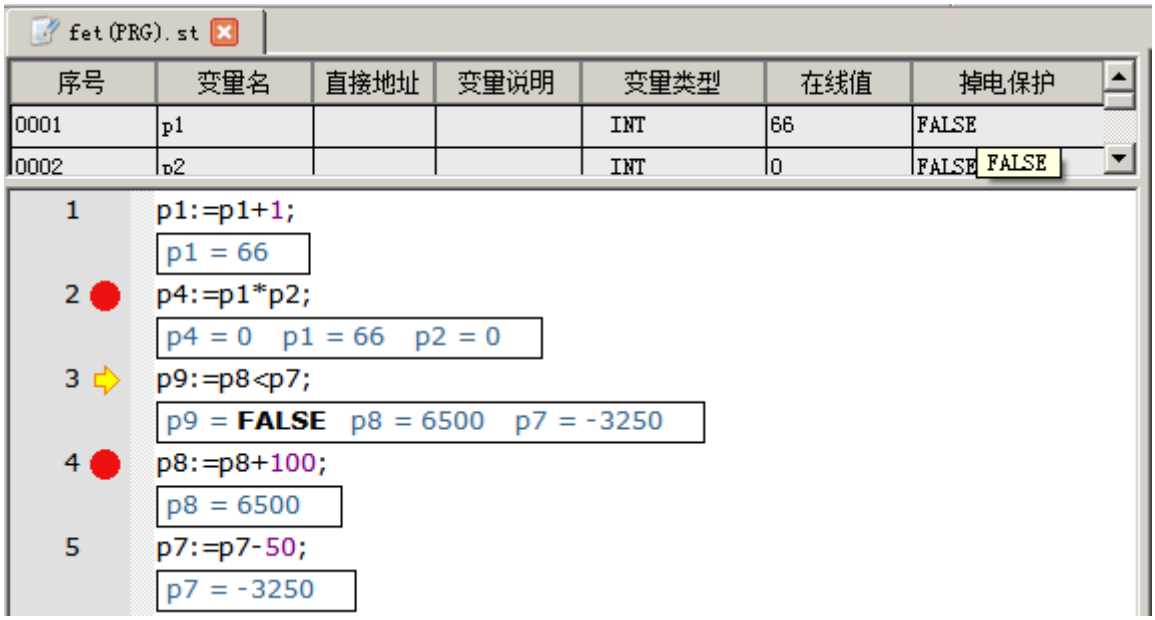
- 菜单栏：单击【在线】—【单步执行】；
- 工具栏：  ；
- 快捷键：**F10**。

2. 示例

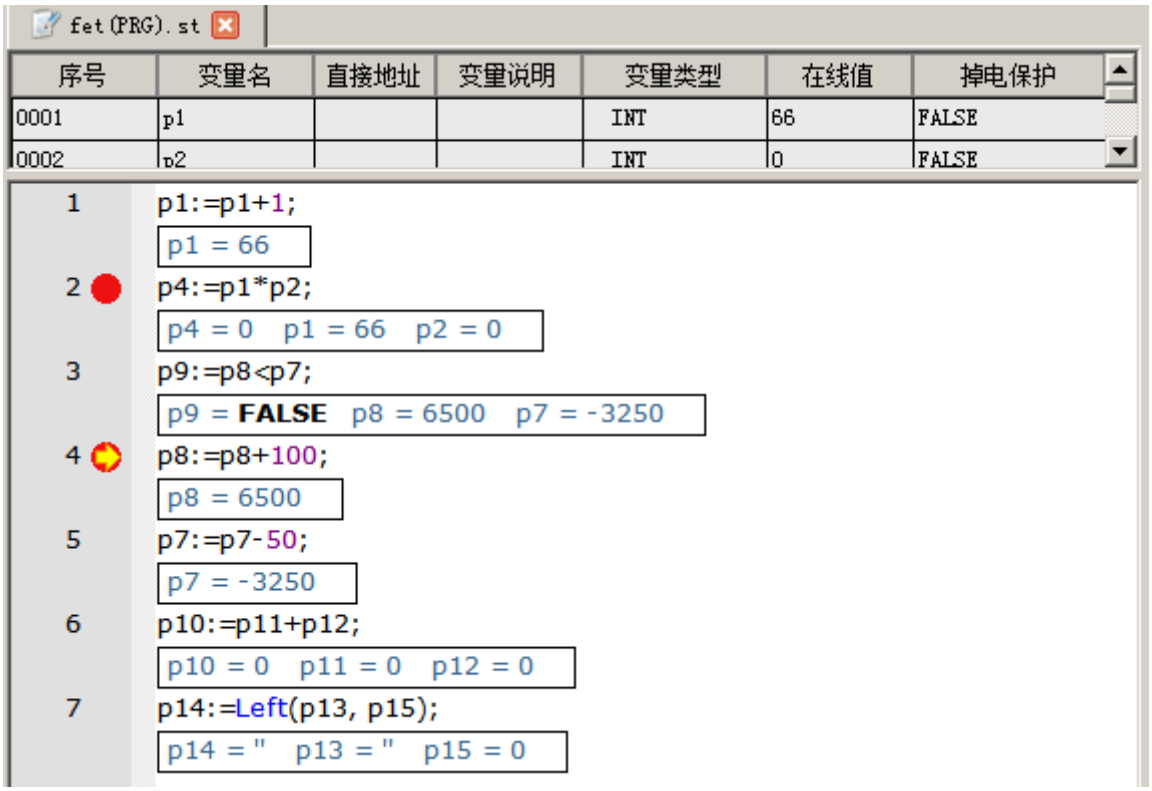
在下图（a）中，当前执行行为 2，执行【单步执行】，第 2 行语句被执行，同时跳转到第 3 行。如图（b）所示。第二次单步执行结果如图（c）所示。


fet(PRG).st
序号 变量名 直接地址 变量说明 变量类型 在线值 掉电保护
0001 p1 INT 66 FALSE
0002 p2 INT 0 FALSE
1 p1:=p1+1;
p1 = 66
2 p4:=p1*p2;
p4 = 0 p1 = 66 p2 = 0
3 p9:=p8<p7;
p9 = FALSE p8 = 6500 p7 = -3250
4 p8:=p8+100;
p8 = 6500
5 p7:=p7-50;

（a）


fet(PRG).st
序号 变量名 直接地址 变量说明 变量类型 在线值 掉电保护
0001 p1 INT 66 FALSE
0002 p2 INT 0 FALSE FALSE
1 p1:=p1+1;
p1 = 66
2 p4:=p1*p2;
p4 = 0 p1 = 66 p2 = 0
3 p9:=p8<p7;
p9 = FALSE p8 = 6500 p7 = -3250
4 p8:=p8+100;
p8 = 6500
5 p7:=p7-50;
p7 = -3250

（b）



（c）

单步执行示例

11.1.8.5 断点运行

1. 概述

断点运行以使能断点为步，执行程序语句。执行【断点运行】时，程序从当前行开始执行，直到下一个使能断点停止。

如果当前调试的 POU 及其调用的其它 POU 中都没有设置断点，执行【断点运行】时，任务退出调试模式。

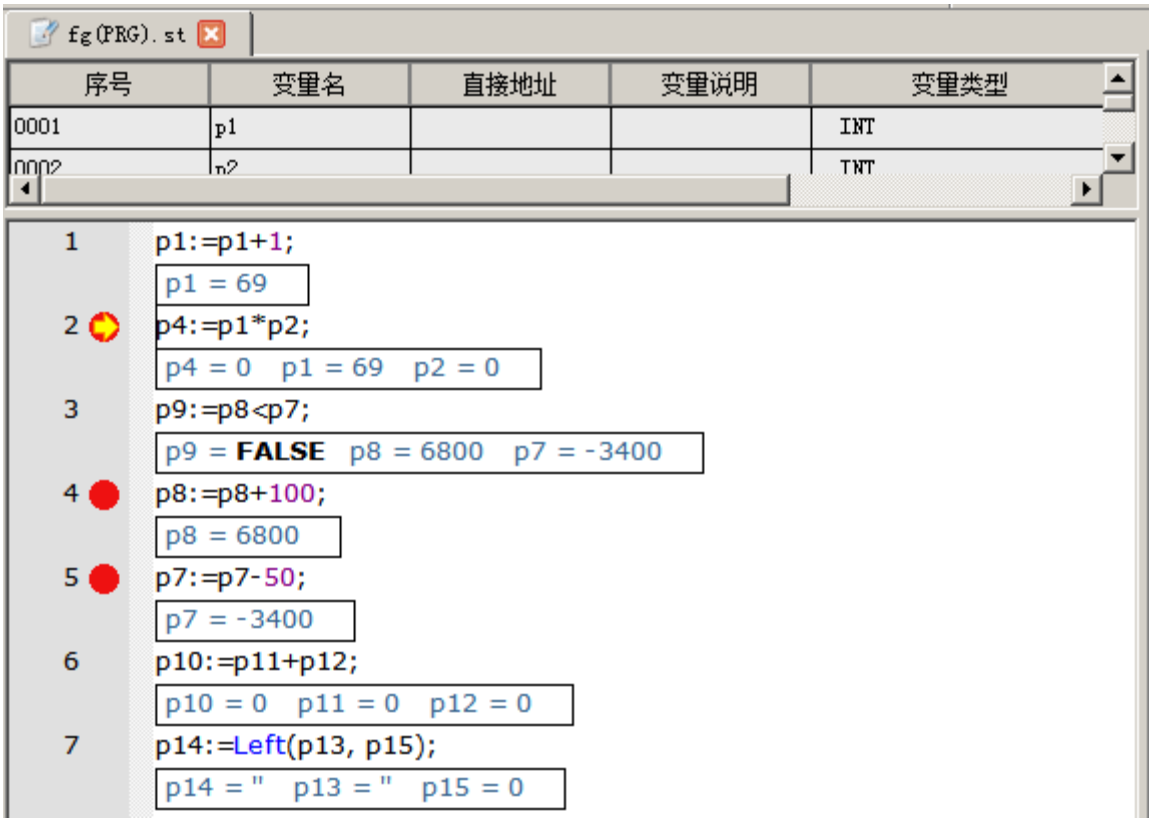
若调用了函数、功能块或其他 ST 程序，即使该语句行没有设置使能断点，但是函数、功能块、或 ST 程序内部设置有使能断点行，则执行【断点运行】时，会跳入到相应的使能断点处。

通过以下方式执行：

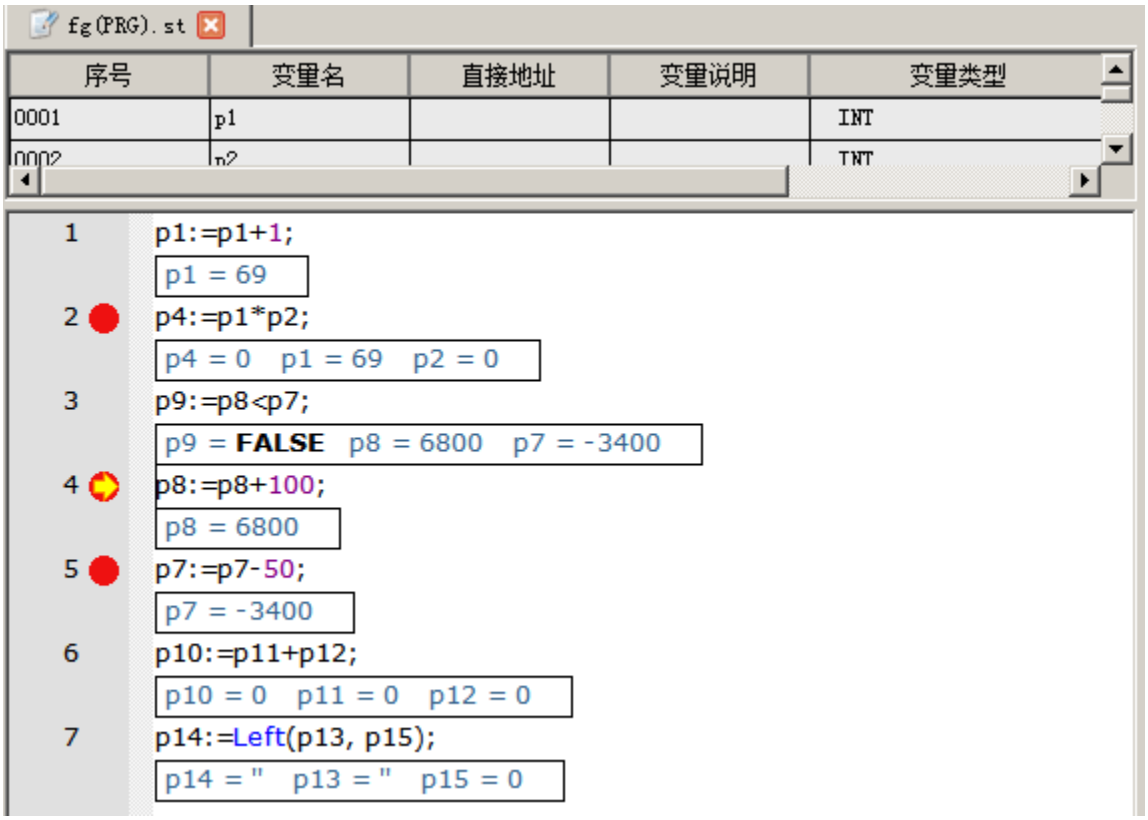
- 菜单栏：单击【在线】—【断点运行】；
- 工具栏：  ；
- 快捷键：**Ctrl+F5**。

2. 示例

在下图（a）中，当前执行行为 2，执行【断点运行】，程序从该行开始执行，直到下一个使能断点行 4 停止执行，如下图（b）所示。再次执行【断点运行】，程序从使能断点行 4 开始执行，直到下一个使能断点行 5 停止，如下图（c）所示。



（a）


fg(PRG).st
序号
变量名
直接地址
变量说明
变量类型
0001
p1
INT
1
p1:=p1+1;
p1 = 69
2
p4:=p1*p2;
p4 = 0 p1 = 69 p2 = 0
3
p9:=p8<p7;
p9 = FALSE p8 = 6800 p7 = -3400
4
p8:=p8+100;
p8 = 6800
5
p7:=p7-50;
p7 = -3400
6
p10:=p11+p12;
p10 = 0 p11 = 0 p12 = 0
7
p14:=Left(p13, p15);
p14 = " p13 = " p15 = 0

（b）


fg(PRG).st
序号
变量名
直接地址
变量说明
变量类型
0001
p1
INT
1
p1:=p1+1;
p1 = 71
2
p4:=p1*p2;
p4 = 0 p1 = 71 p2 = 0
3
p9:=p8<p7;
p9 = FALSE p8 = 7100 p7 = -3500
4
p8:=p8+100;
p8 = 7100
5
p7:=p7-50;
p7 = -3500
6
p10:=p11+p12;
p10 = 0 p11 = 0 p12 = 0
7
p14:=Left(p13, p15);
p14 = " p13 = " p15 = 0

（c）

断点运行示例

11.1.8.6 跳进

1. 概述

该命令用于在线模式下的程序步进调试。当程序中调用函数、功能块或其他 ST 程序时，通过“跳进”命令进入该函数、功能块或程序逻辑内部执行。

当被调试的 POU 中没有设置断点，在线运行时，仅【跳进】命令可操作，并通过该命令进入调试模式，同时激活【单步执行】、【跳出】、【断点运行】命令。执行【跳进】后，跳到即将执行的语句行，并红色高亮显示。

通过以下方式执行：

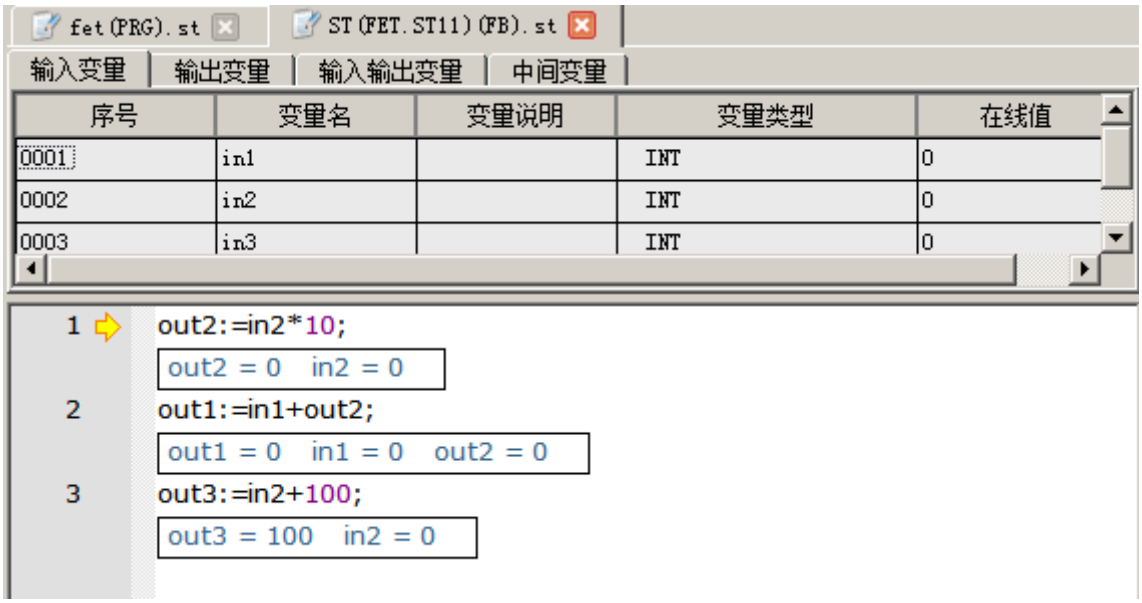
- 菜单栏：单击【在线】—【跳进】；
- 工具栏：  ；
- 快捷键：**Ctrl+F11**。

2. 示例

程序 fet(PRG)中调用功能块实例 ST11，当前执行行为 8，如图（a）所示。执行【跳进】操作，程序进入被调用的功能块内部，如图（b）所示。在功能块最后一步执行【跳进】，则程序执行该行，并返回到图（a）所示的当前功能块调用位置。


fet(PRG).st
序号
变量名
直接地址
变量说明
变量类型
0009
p9
BOOL
1 p1:=p1+1;
p1 = 6
2 p4:=p1*p2;
p4 = 0 p1 = 6 p2 = 0
3 p9:=p8<p7;
p9 = FALSE p8 = 600 p7 = -300
4 p8:=p8+100;
p8 = 600
5 p7:=p7-50;
p7 = -300
6 p10:=p11+p12;
p10 = 0 p11 = 0 p12 = 0
7 p14:=Left(p13, p15);
p14 = '' p13 = '' p15 = 0
8 ST11(
ST11
9 in1:=p16,
p16 = 0
10 in2:=p17,
p17 = 0

（a）



（b）

跳进示例

11.1.8.7 跳出

1. 概述

该命令用于在线模式下的程序步进调试。

如果正在调试的 POU（函数、功能块或已调用的其他 ST 程序）中没有设置断点或设置的断点不使能，执行【跳出】命令时，则会把这个 POU 剩下的语句一次执行完，然后返回到该 POU 被调用处。如果正在调试的 POU 中设置有使能断点，则会跳到下一个断点处停止。

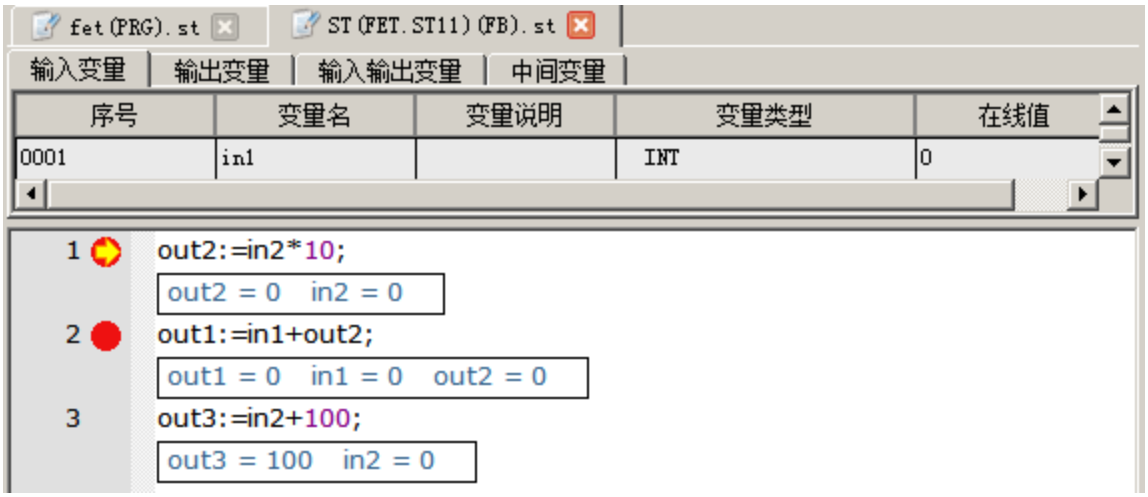
对于普通语句行（非函数、功能块或已调用的其他 ST 程序），【跳出】与【断点运行】的作用一致。

通过以下方式执行：

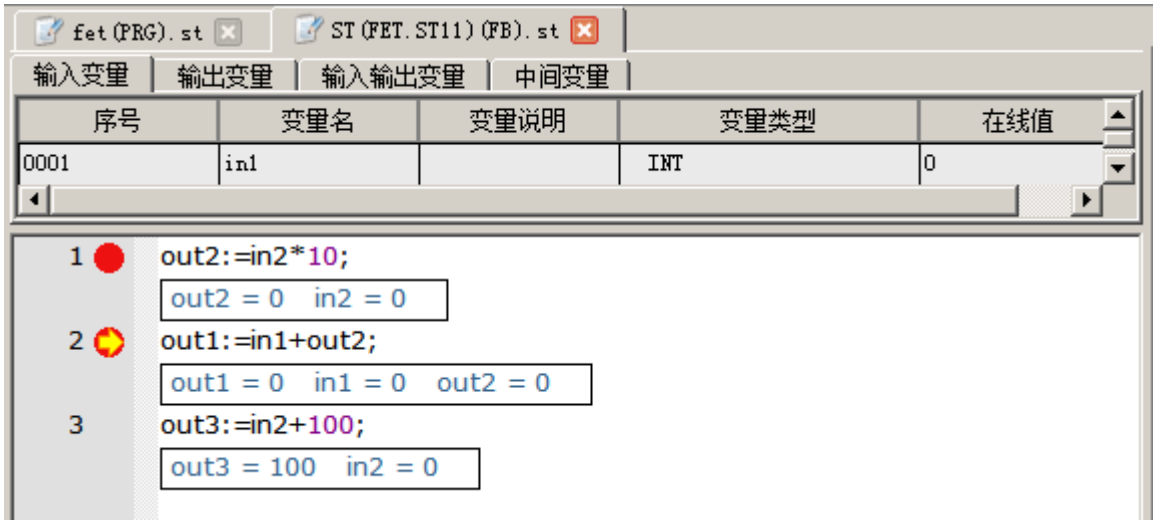
- 菜单栏：单击【在线】—【跳出】；
- 工具栏：  ；
- 快捷键：**Ctrl+F9**。

2. 示例

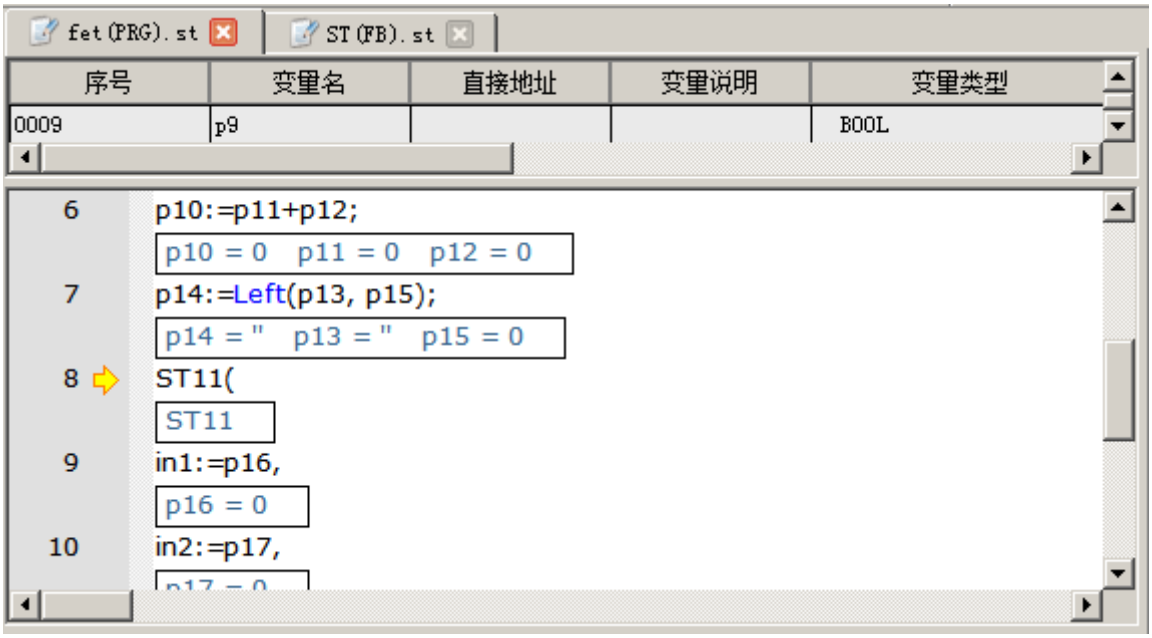
程序 fet(PRG)中调用功能块 ST11，并设置 ST11 的位置 1、2 为使能断点行。功能块 ST11 内部当前执行行为 1，如图（a）所示，执行【跳出】命令，程序执行完第 1 行并跳转到下一个使能断点行 2，如图（b）所示。再次执行【跳出】命令，程序返回到功能块调用处，如图（c）所示。


fet(PRG).st
ST(FET.ST11)(FB).st
输入变量
输出变量
输入输出变量
中间变量
序号
变量名
变量说明
变量类型
在线值
0001
in1
INT
0
1
out2:=in2*10;
out2 = 0 in2 = 0
2
out1:=in1+out2;
out1 = 0 in1 = 0 out2 = 0
3
out3:=in2+100;
out3 = 100 in2 = 0

（a）


fet(PRG).st
ST(FET.ST11)(FB).st
输入变量
输出变量
输入输出变量
中间变量
序号
变量名
变量说明
变量类型
在线值
0001
in1
INT
0
1
out2:=in2*10;
out2 = 0 in2 = 0
2
out1:=in1+out2;
out1 = 0 in1 = 0 out2 = 0
3
out3:=in2+100;
out3 = 100 in2 = 0

(b)



（c）

跳出示例

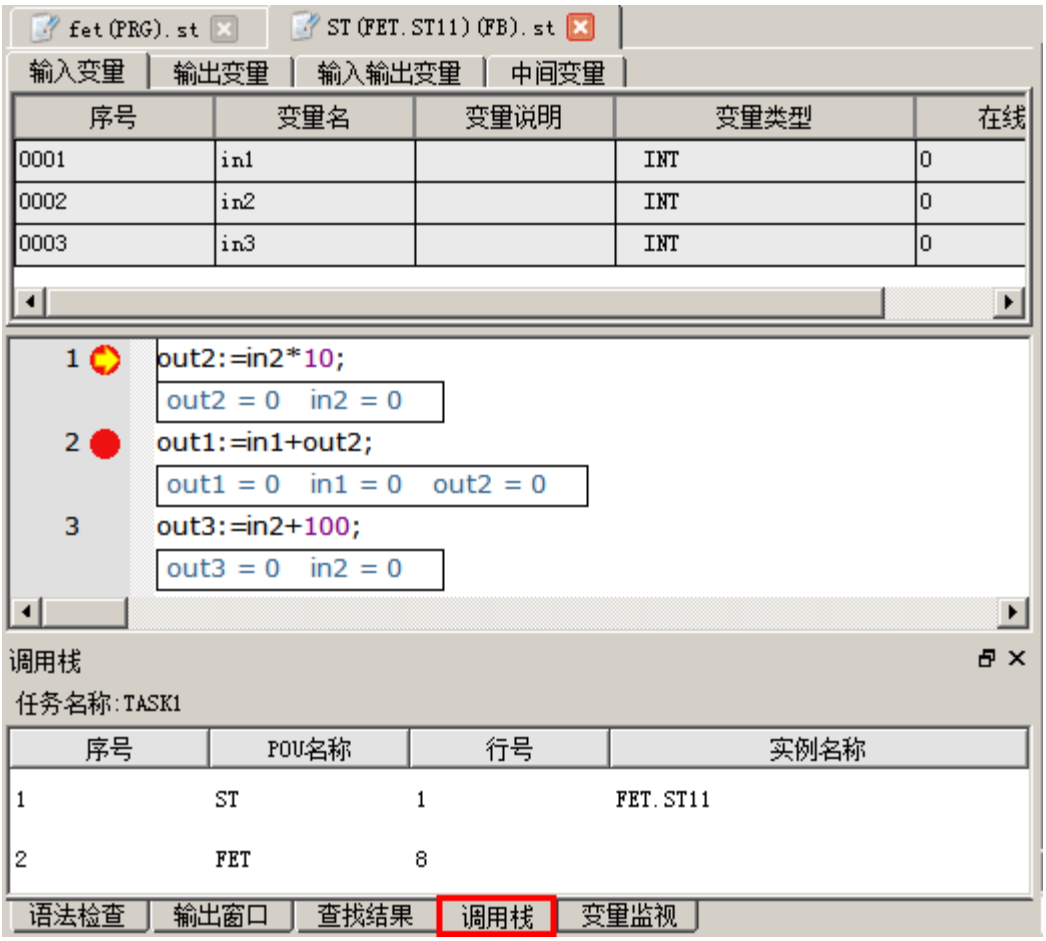
11.1.8.8 调用栈

1. 概述

该功能用于调试模式下，查看当前调试 POU、函数或功能块的历史调用栈列表。

进入在线后，【调用栈】显示在信息输出窗口；调用栈列表中，栈信息仅支持 ST 语言的 POU 栈信息显示。

调用栈信息列表从上到下，依次显示栈顶 POU 信息、中间调用 POU 信息、栈底 POU 信息。双击某行可打开该栈信息对应的 POU，并选中行号对应的逻辑行。



调用栈示例

11.2 调试变量

11.2.1 写入单个变量

11.2.1.1 概念

在线或仿真状态，对单个变量写入新值。写入后，变量将以新值参与逻辑运算。

写入值形式：

- 变量可写入十进制、二进制及十六进制数。十进制可直接输入写入值，二进制和十六进制写入值支持两种格式：“变量值类型#进制#变量值”和“进制#变量值”。
- 对于 BOOL 变量，默认的写入值为当前值取非。可写入值只能是 TRUE 或 FALSE。
- 对于枚举型的变量，可以写入的值只能是此枚举型定义时的枚举值。

11.2.1.2 要求

- 控制器钥匙开关在 REM 或 PRG

- AutoThinkV4 软件在监视状态

11.2.1.3 步骤

要给变量写值，请按以下步骤操作

（1） 在工作区，双击变量名。

将弹出变量调试窗口。

（2） 在“输入变量值”框中，输入写入值。

BOOL 型变量请选择 TRUE 或 FALSE。

（3） 单击**写入**按钮。

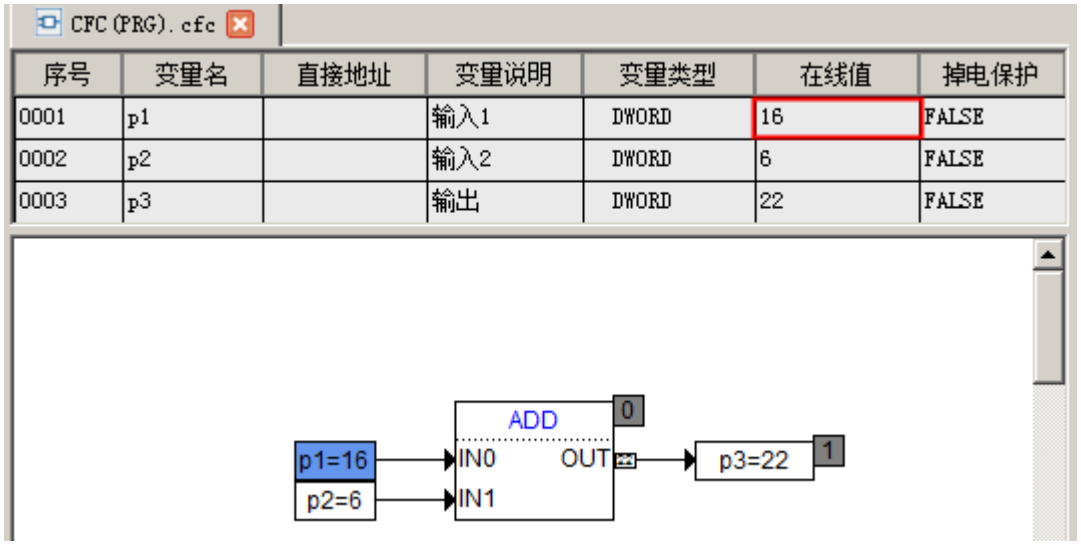
“输出窗口”显示在线值写入成功。

11.2.1.4 示例

如下图所示，仿真时，变量 p1 写入十六进制数 10，写入格式为：“int#16#10”。

调试变量时写入十六进制数字

下图为写入变量值之后程序的运行结果，p1 在线值以 10 进制显示为 16。

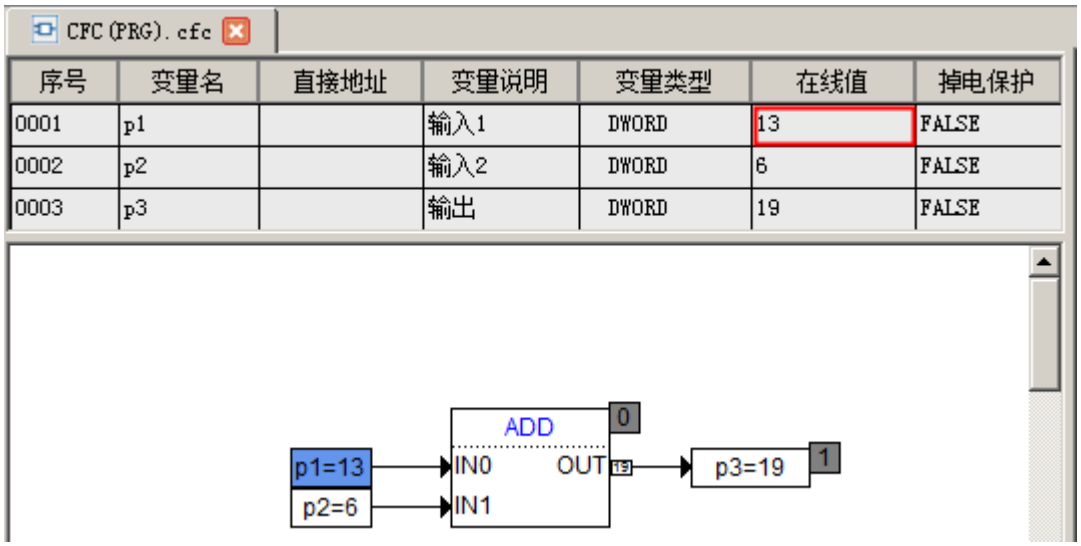


变量值写入之后的结果

以“进制#变量值”形式写入数值，如下图所示。变量 p1 写入二进制数 1101，写入格式为：“2#1101”。

以进制#变量值形式写入变量值

下图为写入变量值之后程序的运行结果，p1 在线值以 10 进制显示为 13。



变量值写入之后的结果

11.2.2 批量写入

11.2.2.1 概述

在线调试时，可以多个变量同时写入。需要将所有调试变量设置为待调试状态，再统一写入。

写入后，所有变量将以新值参与逻辑运算。

11.2.2.2 要求

- 控制器钥匙开关在 REM 或 PRG
- AutoThinkV4 软件在监视状态

11.2.2.3 步骤

要批量写入变量值，请按以下步骤操作：

（1） 在 POU 中，双击变量名。

将弹出变量调试窗口。

（2） 在“输入变量值”框中，输入待写入值。

（3） 单击**待调试**按钮。

在程序区和变量在线值列，以“<>”显示待写入值。

对所有待写入变量，按以上三步设置为待调试状态。

（4） 在“在线”菜单中，选择“写入”命令。

“输出窗口”显示所有待写入值均被成功写入，变量将以新值参与逻辑运算。

11.2.3 强制单个变量

11.2.3.1 概述

在线状态，对单个变量设置强制值。强制后，每个任务扫描周期，都会对变量写入强制值，以强制值参与逻辑运算或输出，直至强制被解除。

对于 BOOL 变量，默认的强制值为当前值取非。

11.2.3.2 要求

- 控制器钥匙开关在 REM 或 PRG
- AutoThinkV4 软件在监视状态

11.2.3.3 步骤

要强制单个变量，请按以下步骤操作：

（1） 在工作区，双击变量名。

将弹出变量调试窗口。

（2） 在“输入变量值”框中，输入强制值。

BOOL 型变量请选择 TRUE 或 FALSE。

（3） 单击**强制**按钮。

“输出窗口”显示在线值强制成功。

11.2.3.4 结果

在变量定义区和程序区被强制变量的变量名和强制值以红色字体显示。

11.2.4 批量强制

11.2.4.1 概述

在线状态，可以多个变量同时强制。需要将所有强制变量设置为待调试状态，再统一强制。

强制后，每个任务扫描周期，都会对变量写入强制值，以强制值参与逻辑运算或输出，直至强制被解除。

11.2.4.2 要求

- 控制器钥匙开关在 REM 或 PRG
- AutoThinkV4 软件在监视状态

11.2.4.3 步骤

要批量强制变量，请按以下步骤操作：

（1） 在 POU 中，双击变量名。

将弹出变量调试窗口。

（2） 在“输入变量值”框中，输入待强制值。

（3） 单击**待调试**按钮。

在程序区和变量在线值列，以“<>”显示待强制值。

对所有强制变量，按以上三步设置为待调试状态。

（4） 在“在线”菜单中，选择“强制”命令。

“输出窗口”显示所有待强制值均被成功强制。

11.2.4.4 结果

在变量定义区和程序区被强制变量的变量名和强制值以红色字体显示。

11.2.5 解除单个变量强制

11.2.5.1 概述

解除单个变量的强制状态，变量值将恢复为实时值。

11.2.5.2 要求

- 当前变量已被强制
- 请确保变量当前值为正常值

11.2.5.3 在调试对话框解除

要解除单个变量强制状态，请按以下步骤操作：

（1） 在工作区，双击变量名。

将弹出变量调试窗口。

（2） 单击**释放**按钮。

“输出窗口”显示变量释放成功。

11.2.5.4 在强制变量表解除

要解除单个变量强制状态，请按以下步骤操作：

（1） 在“在线”菜单中，选择“强制变量表”。

将弹出“强制变量列表”对话框。

（2） 选中变量，单击**释放**按钮。

“输出窗口”显示变量释放成功。

（3） 单击**关闭**按钮。

11.2.6 批量解除强制

11.2.6.1 概述

解除所有变量的强制状态，变量值将恢复为实时值。

11.2.6.2 要求

- 变量已被强制
- 请确保变量当前值为正常值

11.2.6.3 步骤

要批量解除变量强制状态，请按以下步骤操作：

（1） 在“在线”菜单中，选择“强制变量表”。

将弹出“强制变量列表”对话框。

（2） 选中变量，单击**全部释放**按钮。

“输出窗口”显示全部释放成功。

（3） 单击**关闭**按钮。

11.2.7 取消调试

11.2.7.1 概述

批量写入或强制时，需要先将变量设置为待调试状态，如果对于待调试状态的变量不需要做调试，可解除调试状态。

11.2.7.2 要求

- 变量为待调试状态

11.2.7.3 步骤

要取消调试，请按以下步骤操作：

（1） 在工作区，双击变量名。

将弹出变量调试窗口。

（2） 单击**取消调试**按钮。

在程序区和变量在线值列，取消待调试“<>”显示。

11.2.7.4 参考信息

批量写入

批量强制

11.2.8 仿真调试

11.2.8.1 概述

仿真是模拟控制器在线运行状态。您可以在工程投运前，通过仿真功能调试用户逻辑，排查逻辑组态错误。

仿真状态下，通过给变量写入仿真值，验证逻辑的运行结果是否正确。

11.2.8.2 要求

工程编译通过

11.2.8.3 步骤

进入仿真状态，请按以下步骤操作：

（1）“在线”菜单中，选择“仿真”命令。

输出窗口显示“进入监视状态！”，此时，工程在仿真模式。

11.2.8.4 参考信息

写入单个变量

11.2.9 监视变量

11.2.9.1 概述

工程运行中，如果需要对工程中部分变量进行统一监视和调试，可以将这些变量添加到“监视变量”窗口中，进行批量监控。

在线监视时，“监视变量”窗口中，双击变量的灰色区域，打开调试窗口进行调试。可直接在线修改变量名或添加变量。

监视状态下，变量的在线值根据您在“选项”菜单中设置，可显示为二进制、十进制或十六进制。

11.2.9.2 要求

- “窗口”菜单中已勾选“变量监视”命令

- AutoThinkV4 软件在离线状态

11.2.9.3 步骤

离线添加监视变量，请按以下步骤操作：

（1） 在“监视变量”窗口中，右击空白区域。

（2） 在右键菜单中，选择“增加变量”命令。

（3） 双击“变量名”参数项。

（4） 按 F2 键。

将弹出“帮助管理”窗口。

（5） 选择变量类型。

可选择局部变量、全局变量、通道变量。

（6） 选择某一变量，单击**确认。**

变量名被添加。

（7） 单击工具栏监视图标  。

显示变量类型和在线值。

11.2.10 设置监视周期

11.2.10.1概述

在线监视时，数据值刷新的快慢与监视周期有关。您可以根据数据实时性要求，选择合适的监视周期。软件提供 250ms、500ms、1000ms 周期值。

11.2.10.2要求

AutoThinkV4 软件在离线状态

11.2.10.3步骤

要设置监视周期，请按以下步骤操作：

（1） 在“工程”菜单中，选择“选项”命令。

弹出“选项”窗口，默认显示“配置”页面。

（2） 单击监视周期下拉按钮。

（3） 选择周期值。

11.3 控制器操作

11.3.1 控制器硬件复位

11.3.1.1 概述

控制器复位是将控制器重新启动，控制器所有程序重新开始运行，除掉电保护数据之外的所有数据被恢复成初始值。

复位控制器的结果：

- 控制器所有程序重新开始执行
- 除掉电保护数据之外的所有数据(I 区、Q 区、N 区、M 区 4000 开始的数据区)被恢复成初始值
- 变量强制属性保持
- 检查 SD 卡进行固件升级

以下两种情况会触发控制器复位：

- 控制器断电重启
- 钥匙开关复位

控制器发生断电重启时，会自动将工程数据备份到 FLASH 中。

系统正常运行时，禁止对主控制器进行复位操作！

11.3.1.2 要求

- 控制器钥匙开关拨到 REM
- FA-AT 编程软件在监视状态

11.3.1.3 步骤

通过钥匙开关复位，请按以下步骤操作：

（1） 把钥匙开关按 REM→RUN→REM→RUN→REM 顺序拨动。

该操作需在 1.5 s 之内完成。

（2） 将钥匙开关拨到 RUN，重新运行工程。

11.3.2 复位

11.3.2.1 概述

复位是重启用户工程，模拟控制器掉电重启状态，除掉电保持区数据外，所有数据被恢复成初始化值。

复位结果：

- 用户工程重新启动；
- 除掉电保护数据之外的所有数据(I 区、Q 区、N 区、M 区 4000 开始的数据区)被恢复成初始值；
- 变量强制属性保持。

11.3.2.2 要求

- 控制器钥匙开关拨到 REM 或 PRG
- FA-AT 编程软件在监视状态

11.3.2.3 步骤

要复位控制器，请按以下步骤操作：

（1） 在“在线”菜单中，选择“复位”命令。

输出窗口显示“复位控制器成功！”。

11.3.2.4 结果

在线后，除掉电保持数据外，所有变量在线值变为初始值。

11.3.3 冷复位

11.3.3.1 概述

冷复位是重启用户工程，模拟工程全下装后的控制器状态，将各内存区所有数据恢复到初始值。

冷复位结果：

- 用户工程重新启动；
- 所有数据被恢复成初始值；
- 变量强制属性不保持。

11.3.3.2 要求

- 控制器钥匙开关拨到 REM 或 PRG

- FA-AT 编程软件在监视状态

11.3.3.3 步骤

要冷复位控制器，请按以下步骤操作：

（1） 在“在线”菜单中，选择“冷复位”命令。

输出窗口显示“冷复位控制器成功！”。

11.3.3.4 结果

在线后，所有变量在线值变为初始值，强制属性取消。

11.3.4 热复位

11.3.4.1 概述

热复位是重启用户工程，重启后各数据区保持重启前状态。

热复位结果：

- 用户工程重新启动；
- 所有数据保持；
- 变量强制属性保持。

11.3.4.2 要求

- 控制器钥匙开关拨到 REM 或 PRG
- FA-AT 编程软件在监视状态

11.3.4.3 步骤

要热复位控制器，请按以下步骤操作：

（1） 在“在线”菜单中，选择“热复位”命令。

输出窗口显示“热复位控制器成功！”。

11.3.4.4 结果

在线后，所有变量在线值保持不变。

11.3.5 控制器校时

11.3.5.1 概述

您需要周期性校准控制器时间，使控制器时间与本地时间保持同步，为保证控制器记录的日志信息、诊断信息在时间上的准确性。

设置控制器时间时，可以手动设置为设定时间值，也可以同步为当前 PC 机系统时间。

11.3.5.2 要求

- 控制器钥匙开关拨到 REM 或 PRG
- AutoThinkV4 软件在离线状态

11.3.5.3 步骤

要校准控制器时间，请按以下步骤操作：

（1） 在“在线”菜单中，选择“RTC 校时”命令。

将弹出 RTC 校时对话框。

（2） 在“时区”下拉框中，选择本地时区。

（3） 在时间编辑框中，输入需要设定的时间。

（4） 单击**设置**按钮。

时间设定值被写入到控制器。

如果需要将控制器时间同步为 PC 机时间，请执行第（5）步。

（5） 单击**同步 PC 时间**按钮。

对话框上面显示的 PC 机系统时间被写入控制器，同时，时间编辑框中显示被同步到控制器的时间值。

11.3.5.4 参考信息

时区	**代表城市**
UTC	摩洛哥、巴黎
UTC+1	伦敦、柏林、尼日利亚、中非 、刚果
UTC+2	莫桑比克、南非、利比亚、埃及、土耳其、巴勒斯坦、以色列、叙利亚
UTC+3	莫斯科、马达加斯加、索马里、埃塞俄比亚
UTC+4	毛里求斯、亚美尼亚、伊拉克
UTC+5	巴基斯坦
UTC+6	俄罗斯
UTC+7	泰国、越南
UTC+8	中国 、蒙古、新加坡
UTC+9	日本、韩国、朝鲜
UTC+10	堪培拉、悉尼

UTC+11	所罗门群岛
东西十二区	俄罗斯、新西兰
UTC-11	纽埃岛
UTC-10	库克群岛
UTC-9	安克雷奇
UTC-8	西雅图、拉斯维加斯、洛杉矶
UTC-7	盐湖城
UTC-6	休斯敦 、芝加哥、圣保罗、墨西哥 >
UTC-5	纽约、华盛顿、渥太华、温哥华
UTC-4	智利
UTC-3	里约热内卢、乌拉圭、阿根廷
UTC-2	格陵兰
UTC-1	佛得角

11.3.6 清空控制器工程

11.3.6.1 概述

清空控制器工程文件或用户文件。

- 工程文件：包括.at 编译文件和.hpf 工程源文件，存放在控制器目录 ffx0/custom 下存放
- 用户文件：用户自定义文件，在控制器目录 ffx0/user 下存放

11.3.6.2 要求

- AutoThinkV4 软件在离线状态
- 工程处于停止状态

11.3.6.3 步骤

要清空控制器工程或用户文件，请按以下步骤操作：

（1） 在“在线”菜单中，选择“清空控制器”命令。

将弹出“清空控制器”对话框。

（2） 单击下拉框，选择需要清空的目标文件

（3） 单击**确定**按钮。

弹出确认删除对话框。

（4） 单击**是**按钮。

信息输出窗口中显示“清空控制器成功！”。

清空工程文件后，控制器 RUN 灯熄灭。此时，控制器中无工程，请重新编译下装用户工程。

11.3.7 恢复出厂设置

11.3.7.1 概述

将控制器状态恢复到出厂默认设置。

恢复出厂设置，将执行以下操作：

- 初始化用户数据
- 清除控制器用户文件/用户工程
- 清除控制器锁
- 清除静态路由表
- 将控制器 IP 恢复默认值，网口 1:192.168.0.1，网口 2:192.168.1.1
- 将子网掩码恢复默认值，255.255.255.0

11.3.7.2 要求

必须断开外部连接。

11.3.7.3 步骤

要恢复出厂设置，请按以下步骤操作：

（1） 在 1.5s 之内，将控制器钥匙开关按 REM→PRG→REM→PRG→REM 顺序拨动。

开始恢复出厂设置：FRC 灯和 BAT 灯同时闪烁，开始恢复出厂设置。

恢复出厂设置完成：FRC 灯和 BAT 灯停止闪烁，恢复出厂设置完成。

11.4 辅助工具使用

11.4.1 概述

通过辅助工具，可连接控制器进行在线调试和维护操作，而无需打开 AutoThinkV4。

主要功能：

- 获取控制器版本信息
- 在线监视主从控制器工作状态
- 设置控制器密码锁

- 在线修改控制器 IP 地址、路由信息
- 在线升级控制器工程和固件
- 读取控制器操作日志

通过以下两种方式使用：

- 在“工具”菜单中，选择“辅助工具”。
- 拷贝安装路径 V4.1.1→Tools 文件夹，然后打开 ATTools.exe 应用程序独立使用。

11.4.2 建立辅助工具与控制器连接

11.4.2.1 概述

对控制器进行操作前，首先需要建立辅助工具与控制器间的连接。通过 IP 地址访问控制器，网口 1 默认 IP：192.168.0.1，网口 2 默认 IP：192.168.1.1。

11.4.2.2 要求

辅助工具已打开

11.4.2.3 步骤

要建立辅助工具与控制器的连接，请按以下步骤操作：

（1） 单击控制器类型下拉按钮，选择控制器型号。

（2） 在 IP 编辑框输入当前控制器 IP 地址。

（3） 设置本地 PC 机 IP 地址。

设置时，与控制器 IP 地址在一个网段即可。

（4） 单击**测试连接**按钮

将检测辅助工具与控制器连接是否正常。如果 IP 地址不正确，则提示控制器连接失败。请检查 IP 地址正确后，再进行连接。

与控制器建立连接

11.4.3 获取控制器信息

11.4.3.1 概述

通过辅助工具获取控制器版本、协议库版本以及 MAC 地址信息。

控制器信息说明

基本信息	说明
HW	硬件版本
BOOT	启动程序版本
RTS	RTS 程序版本
DP_LIB	Profibus-DP 协议动态库版本
POWERLINK_LIB	POWERLINK 协议动态库版本
MODBUS_LIB	Modbus 协议动态库版本
SN	控制器单板序列号
ETH1 MAC	以太网口 1 的 MAC 地址
ETH2 MAC	以太网口 2 的 MAC 地址

11.4.3.2 要求

辅助工具与控制器已建立连接

11.4.3.3 步骤

要获取控制器版本信息，请按以下步骤操作：

（1） 在“控制器信息”标签窗口中，单击**读取信息**按钮。

控制器版本信息将显示在“基本信息”框中。

11.4.3.4 参考信息

建立辅助工具与控制器连接

11.4.4 监视控制器状态

11.4.4.1 概述

通过读取信息操作，辅助工具进入在线监视状态，实时监测控制器运行状态。“本机状态”显示当前连接控制器的状态信息，“对方机状态”显示冗余控制器状态信息。

控制器状态信息说明

控制器状态	状态信息	说明	处理措施
主从机状态	未知态	初始状态	——
	初始态	对方机控制器模块未插入	请检查控制器模块是否正常插入
	硬件 Ready 态	过程状态	联系厂家
	双机 Ready 态	过程状态	联系厂家
	单机 Ready 态	过程状态	联系厂家
	故障态	双机均为故障态：A/B 系开关设置冲突	重新设置 A/B 系拨码开关
	错误态	冗余通信链路故障时，对方机为错误态	检查冗余通信连接是否正常
	工程冗余	主机无工程	将工程下载到控制器
		控制器 Flash 空间不足	清空 Flash 或联系厂家 清空 Flash 后必须断电重启
	数据冗余	冗余任务运行周期与冗余数据量不匹配	重新设置冗余任务运行周期
	从机	当前控制器为主机	——
	主机	当前控制器为从机	——
	不合格主机	上一次冗余没有完成的控制器，在上电后控制器为不合格主机	1 通过钥匙开关复位 2 在“辅助工具”→“控制器信息”标签中单击清

			除按钮，再断电重启
单双机状态	未知态	初始状态	——
	单机	当前控制器为单机模式	——
	双机	当前控制器为双机模式	——
AB 机状态	未知态	初始状态	——
	A 机	当前控制器为 A 机	——
	B 机	当前控制器为 B 机	——
冗余模块工作网状态	未知态	冗余通信链路故障	检查冗余通信连接是否正常
	第一路通讯	第一路通讯网在工作	——
	第二路通讯	第二路通讯网在工作	——
工程运行状态	未知态	初始状态	——
	正在运行	当前控制器工程正在运行	——
	已停止	当前控制器工程已停止	——
钥匙开关状态	UNKNOWN	初始状态	——
	RUN	钥匙开关在 RUN 位置，控制器在运行模式	——
	REMOTE	钥匙开关在 REMOTE 位置，控制器在远程模式	——
	PRG	钥匙开关在 PRG 位置，控制器在编程模式	——

11.4.4.2 要求

辅助工具与控制器已建立连接

11.4.4.3 启动监视

要在线监视控制器状态，请按以下步骤操作：

（1） 在“控制器信息”标签窗口中，单击**读取信息**按钮。

辅助工具进入在线监视状态，冗余控制器状态信息将显示在“本机状态”和“对方机状态”下方。

11.4.4.4 退出监视

当前处于监视状态，若要退出监视，请按以下步骤操作：

（1） 在“控制器信息”标签窗口中，单击**停止**按钮。

辅助工具退出监视状态，控制器状态保持退出前状态。

11.4.4.5 参考信息

建立辅助工具与控制器连接

11.4.5 控制器加锁

11.4.5.1 概述

通过对控制器加锁，防止离线状态下对控制器进行相关操作，包括升级控制器、下传用户文件、下装、上传用户工程、下载用户工程、监视、清空控制器、IP 修改、工程升级、固件升级。

只能对主控制器加锁，锁密码自动冗余到从控制器。

请妥善保管锁密码，若忘记控制器锁密码，则只能通过恢复出厂设置的方式来清除密码。

11.4.5.2 要求

- 辅助工具与控制器已建立连接
- AutoThinkV4 软件在离线状态
- 辅助工具在离线状态

11.4.5.3 加锁

要对控制器加锁，请按以下步骤操作：

（1） 在“控制器锁”标签窗口中，单击**控制器锁**按钮。

（2） 在“控制器锁”对话框中，输入两次相同的 6-16 位数字“加锁码”，单击**确定**按钮。

控制器加锁成功。

11.4.5.4 解锁

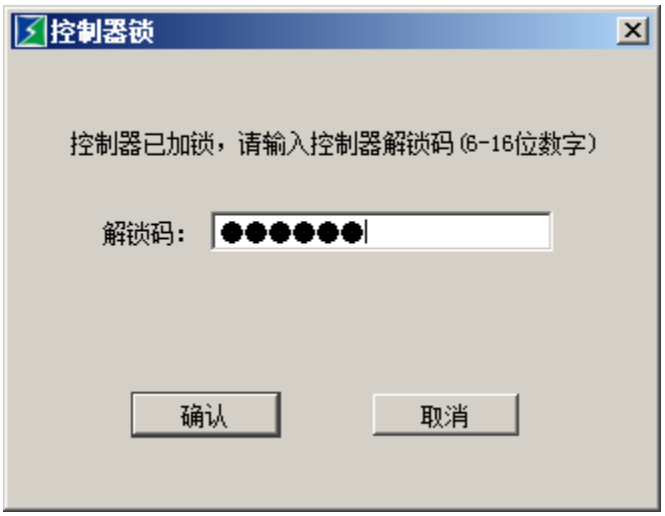
加锁后，如果需要对控制器执行相关操作，请先解锁。

要对控制器解锁，请按以下步骤操作：

（1） 在“控制器锁”标签窗口中，单击**控制器锁**按钮。

（2） 在“控制器锁”对话框中，输入之前设置的加锁码，单击**确定**按钮。

控制器解锁成功。



控制器解锁

11.4.5.5 参考信息

建立辅助工具与控制器连接

监视控制器状态

11.4.6 修改控制器 IP

11.4.6.1 概述

控制器网口 1 默认 IP 为 192.168.0.1、网口 2 默认 IP 为 192.168.1.1。通过辅助工具可修改控制器 IP 地址，并写入控制器。

11.4.6.2 要求

- 辅助工具与控制器已建立连接
- AutoThinkV4 软件在离线状态
- 辅助工具在离线状态
- 控制器在单机模式，并且任务处于停止状态

11.4.6.3 规则

- 控制器为冗余配置时，请在单机模式下，分别将 A、B 机 IP 地址设置为相同值。当冗余主从机建立后，主控制器 IP 地址为 IP 设置值，从控制器 IP 地址为主控制器 IP 地址加 1。
- 从控制器升主后，IP 地址自动减 1，主控制器降为从机后，IP 地址自动加 1。

例如，将 A、B 机两个网口 IP 地址均设置为 192.168.0.1 和 192.168.1.1，当冗余主从机建立后，主机 IP 地址为 192.168.0.1 和 192.168.1.1，从机 IP 地址为 192.168.0.2 和 192.168.1.2，此

时，即使发生主从切换，新主机 IP 地址仍然为 192.168.0.1 和 192.168.1.1，从机 IP 地址为 192.168.0.2 和 192.168.1.2。

- 当控制器为冗余配置时，主机 IP 地址的第四字段不能设置为 254，否则，无法与从机正常通讯。
- 同一网段内，多套控制器的 IP 地址不能连续。

11.4.6.4 步骤

要修改控制器 IP 地址，请按以下步骤操作：

（1） 在“网络配置”标签窗口中，单击 **IP 读取**按钮。

将从控制器读取到两个网口的 IP 地址、掩码和默认网关。

（2） 单击**确认**按钮。

（3） 勾选以太网口，输入新的 IP 地址和掩码。

（4） 单击 **IP 修改**按钮。

控制器 IP 地址立即生效。

11.4.6.5 参考信息

建立辅助工具与控制器连接

在线监视

停止任务

监视控制器状态

11.4.7 修改以太网通讯模块 IP

11.4.7.1 概述

控制器可通过以太网通讯模块 LK246C 扩展其通讯能力，如果需要独立设置某一通讯模块的 IP 地址和路由信息，请通过“控制器操作”辅助工具进行设置。

您可以通过控制器 IP 或 LK246C 的 IP 地址与控制器建立连接。控制器和 LK246C 默认 IP 相同，网口 1 默认 IP 地址为 192.168.0.1、网口 2 默认 IP 地址 192.168.1.1。

辅助工具与控制器建立连接后，可读取本机模块 IP 及路由信息。通过查找功能，搜索该控制器所连接的所有以太网通讯模块槽位。通过槽位号，设置对应模块 IP 及路由信息。

IP 设置规则：

- 不能与控制器 IP 地址相同。
- 不能与其他以太网通讯模块 IP 地址相同。

- 冗余配置时，请先将主、从机架上以太网通讯模块 IP 设置为一致。建立冗余机架后，主机架以太网通讯模块 IP 地址为设置值，从机架以太网通讯模块 IP 地址是主机架 IP 地址加 1。

11.4.7.2 要求

- 已安装以太网通讯模块
- “控制器操作”辅助工具与控制器已建立正常连接

11.4.7.3 步骤

修改以太网通讯模块 IP 地址，请按以下步骤操作：

（1） 槽位号默认为“本机”，单击**查找**按钮。

将查找该控制器的所有以太网通讯模块槽位号，显示在“槽位号”下拉框中。

（2） 单击“槽位号”下拉框，选择槽位号。

（3） 单击 **IP 读取**按钮。

“IP 信息”框中显示两路网口 IP 及掩码。

（4） 勾选网口，输入新的 IP 地址和掩码。

（5） 单击 **IP 修改**按钮。

11.4.8 配置路由

11.4.8.1 概述

通过小工具为控制器网口配置路由信息或修改路由。

11.4.8.2 要求

- 辅助工具与控制器已建立连接
- AutoThinkV4 软件在离线状态
- 辅助工具在离线状态
- 控制器在单机模式，并且任务处于停止状态

11.4.8.3 规则

路由设置规则同 PC 机路由设置。

11.4.8.4 添加路由

要添加路由表，请按以下步骤操作：

（1） 在“网络配置”标签窗口中，勾选需要配置的以太网口。

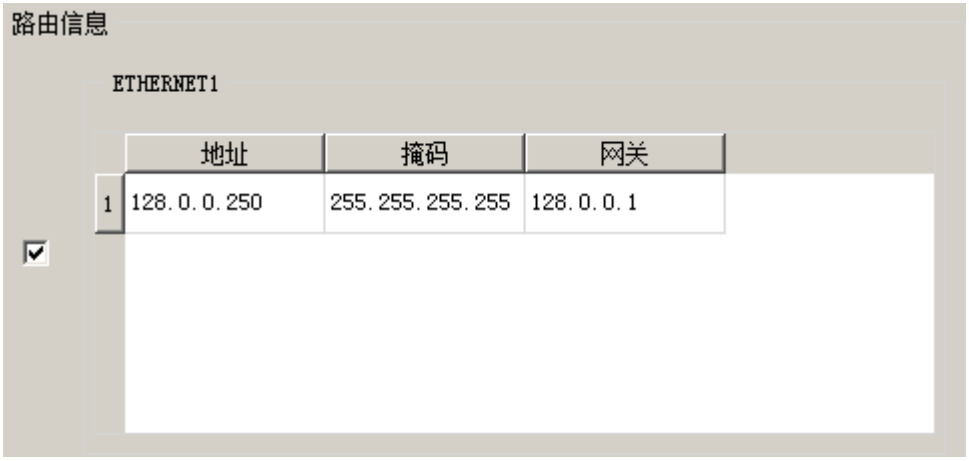
（2） 右击路由表区域，选择**添加**命令。

（3） 在“添加路由”对话框中，输入 IP 地址、子网掩码、网关信息，单击**确定**。

（4） 单击**路由修改**按钮。

路由信息立即生效。

如果控制器或 LK246C 模块与第三方设备在同一网段内，不建议添加路由信息。



添加路由信息

11.4.8.5 修改路由

要修改已添加的路由信息，请按以下步骤操作：

（1） 单击**路由读取**按钮。

（2） 将获取两个以太网口的路由信息，包含 IP 地址、子网掩码和网关。

（3） 勾选需要修改的以太网口，双击路由信息。

（4） 在“添加路由”对话框中，输入新的路由信息，单击**确定**。

（5） 单击**路由修改**按钮。

路由信息立即生效。

11.4.8.6 参考信息

建立辅助工具与控制器连接

在线监视

停止任务

监视控制器状态

11.4.9 工程升级

11.4.9.1 概述

当在没有用户源工程的情况下，如果想升级控制器工程，可通过辅助工具将编译后的工程文件传输到控制器，进行工程更新。

工程升级时，将“.at”文件或“.key”文件传输到控制器，进行工程更新。

- “.at”文件：用户源工程全编译后生成的工程文件。
- “.key”文件：“.at”文件加密后生成的加密工程文件。

工程升级规则：

- 升级时，如果加密文件中的 SN 与控制器 SN 号一致，则升级控制器工程；如果 SN 号不一致，则无法进行工程升级。

11.4.9.2 要求

- FA-AT 编程软件在离线状态
- 辅助工具在离线状态
- 任务处于停止状态

11.4.9.3 步骤

要进行工程升级，请按以下步骤操作：

（1） 在“工程升级”标签窗口中，单击按钮 . . 。

（2） 在安装路径 project 文件夹中,选择.at 或者.key 文件，单击**升级**按钮。

开始升级控制器工程，升级完成提示“工程升级成功!”。

（3） 单击 **OK** 按钮。

11.4.10 工程文件加密

11.4.10.1概述

通过对工程目标文件加密，使工程与控制器唯一关联，加密后，工程只能在关联的控制器中运行。

对编译后的.at 工程文件进行加密，使其与控制器 SN 号进行绑定，生成.key 加密文件。一个工程文件可绑定多个控制器 SN 号。

工程文件加密时，需要创建一个.csv 文件，用于将控制器 SN 号进行分组。每组最多 32 个 SN 号，生成一个.key 文件。

11.4.10.2要求

- 工程已编译通过
- 已通过获取控制器版本信息操作，获取到控制器 SN 号

11.4.10.3步骤

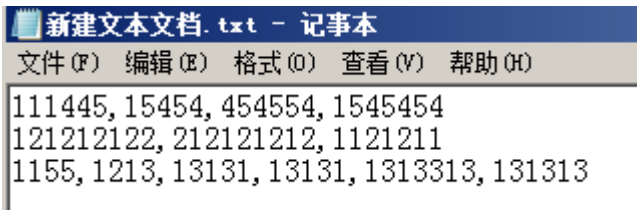
要批量加密工程文件，请按以下步骤操作：

（1） 新建一个.txt 文件。

（2） 在文件中，输入需要批量加密的控制器 SN 号。

输入规则如下：

- 每一行为一组，对应同一个.key 文件；
- 最多支持 32 组，可通过 Enter 键换行分组；
- 每组最多可输入 32 个 SN 号；
- SN 号之间用英文逗号隔开。



SN 号输入示例

（3） 保存文件。

（4） 将文件.txt 后缀名修改为.csv。

SN 文件被创建。

（5） 在“工具”菜单中，选择“工程目标文件加密”命令。

“工程文件加密”对话框被打开。

（6） 通过**打开**按钮，分别选择创建的 SN 文件和.at 工程文件。

（7） 单击**加密**按钮。

提示成功生成加密文件。

（8） 单击 **OK** 按钮。

将在对应的工程文件夹下，按组生成.key 文件。

.key 文件名默认为：工程名_每组首个 SN 号_每组 SN 个数_KEY。

11.4.10.4下一步

可以进行工程升级。

11.4.10.5参考信息

工程升级

获取控制器版本信息

11.4.11 在线升级固件

11.4.11.1概述

通过辅助工具，在线连接控制器，将新固件拷贝到控制器 Flash 中，重启控制器后，模块固件升级为最新版本。

固件升级后，控制器运行状态与升级前保持一致。

11.4.11.2要求

- FA-AT 编程软件在离线状态
- 辅助工具在离线状态
- 任务处于停止状态

11.4.11.3步骤

要通过辅助工具升级固件，请按以下步骤操作：

（1） 在“固件升级”标签窗口中，单击按钮..。

（2） 选择发行盘中.bin 固件文件。

（3） 单击**升级**按钮。

开始向控制器传输固件文件，传输完成后弹出确认框。

（4） 单击 **OK** 按钮。

开始升级，此时，控制器 FRC 指示灯和 BAT 指示灯同时慢闪，当 FRC 与 BAT 指示灯灭时，ERR 指示灯开始慢闪，当 ERR 指示灯灭后，控制器升级完成。

（5） 重启控制器。

11.4.11.4参考信息

使用 SD 卡升级固件

11.4.12 扫描 IP

11.4.12.1扫描 IP

1. 概述

通过小工具的 IP 扫描功能，可以查找局域网内所有控制器的 IP 地址。需要注意的是，IP 扫描不支持路由级联，如果局域网中接入了路由器，则路由器接入的设备 IP 无法扫描到。

如果 PC 机为多网卡，您需要在扫描前，检查接入网络的网卡型号，并在小工具中进行配置，以确保通讯口正确。

支持 LK246C 的 IP 扫描。

2. 要求

- 已确认网卡型号
- “控制器操作”窗口已打开

3. 步骤

扫描局域网内 IP 地址，请按以下步骤操作：

（1） 单击“IP 扫描”标签页。

（2） “通讯接口”下拉框中，选择网卡型号。

（3） 单击**查找 CPU** 按钮。

开始扫描，窗口中显示已扫描到的控制器及 IP 地址。

所有控制器查找结束后，提示扫描完成。

（4） 单击**确定**按钮。

4. 下一步

扫描完成后，您可以选择 IP 地址，检测其所属的控制器，或者设置 IP 地址和站名称。

5. 参考信息

检测控制器

修改站名称

11.4.12.2检测控制器

1. 概述

您可以通过点亮指示灯的方式，识别控制器的位置。点亮指示灯时，通过 MAC 地址识别所属控制器。

2. 要求

- “控制器操作”工具已打开

- IP 地址已被读取

3. 步骤

要检测控制器，请按以下步骤操作：

（1） 在“IP 扫描”标签页中，选择 IP 地址。

（2） 单击**闪烁指示灯**按钮。

此时，IP 所属控制器的 RUN 灯、ERR 灯、FRC 灯、BAT 灯同时闪烁；IP 所属 LK246C 的 RUN 灯、ERR 灯同时闪烁，30s 后自动停止闪烁。

您也可以在确认控制器后，单击**熄灭**按钮使其停止闪烁。

11.4.12.3修改站名称

1. 概述

您可以对局域网内的扫描到的控制器进行重命名。

2. 要求

- “控制器操作”工具已打开
- 已查找到控制器

3. 步骤

要修改站名称，请按以下步骤操作：

（1） 在“IP 扫描”标签页中，选择控制器一路 IP 地址。

（2） 单击**编辑**按钮。

站名称栏变为可编辑状态。

（3） 输入新名称。

- 名称为 a~z、A~Z、0~9、“-”、“.”字符的组合。
- 名称不得超过 20 个字符。

（4） 单击**下发**按钮。

站名称被修改。

11.5 查看日志信息

11.5.1 读取控制器日志

11.5.1.1 概念

通过辅助工具读取控制器中日志文件，供专业人员分析问题用。

11.5.1.2 要求

辅助工具已打开

11.5.1.3 步骤

要读取控制器日志文件，请按以下步骤操作：

（1） 在“日志读取”标签窗口中，单击按钮 .. 。

（2） 选择日志保存路径。

（3） 单击**读取**按钮。

控制器日志文件被下载到保存路径下，读取成功时会弹出提示框。

11.5.2 设置通讯日志类型

11.5.2.1 概述

在线状态，实时记录 AutoThinkV4 软件与控制器间的通讯信息，供技术人员分析问题使用。

支持完全记录模式、精简记录模式和不记录三种方式，可根据实际需要设置记录类型。

- 完全模式：日志记录所有通讯信息，包括数据交互时间、服务类型、通讯数据。
- 精简模式：日志记录部分通讯信息，包括数据交互时间、服务类型和数据字节数，不含具体数据。
- 不记录：不会生成日志文件。

11.5.2.2 要求

AutoThinkV4 软件在离线状态

11.5.2.3 步骤

要设置通讯日志记录类型，请按以下步骤操作：

（1） 在“工程”菜单中，选择“选项”命令。

弹出“选项”窗口，默认显示“配置”页面。

（2） 单击通讯日志记录类型下拉按钮。

（3） 选择记录类型。

（4） 单击**确定**。

11.5.2.4 结果

进入在线后，工程路径“comm_log”文件夹中将实时生成.log 日志文件。

11.5.3 查看操作日志

11.5.3.1 概述

操作日志记录程序的在线操作信息：操作时间、操作内容、记录的操作条数。

操作日志信息超过 5000 条将生成备份日志。同时，清空操作日志并重新开始记录。

日志文件存放在工程文件目录下，操作日志文件名称为“工程名.log”，备份日志文件名称为“工程名_日期.log”。

11.5.3.2 要求

- 工程有操作记录
- 工程在离线模式

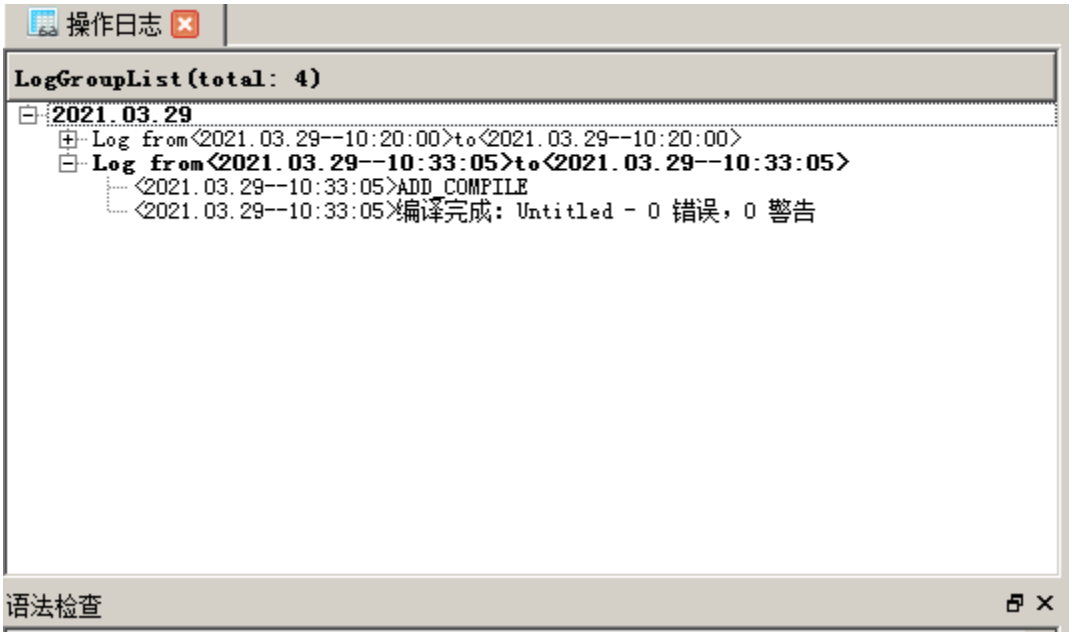
11.5.3.3 步骤

要查看操作日志，请按以下步骤操作：

（1） 在“工程”菜单中，选择“查看操作日志”命令

将弹出“操作日志”窗口。

单击日志信息前的“+”、“-”标记，可展开或收缩操作日志节点。



操作日志窗口

11.6 查看 SOE 信息

11.6.1 概述

控制器实时记录 LK631C SOE 模块的通道历史事件信息。当通道组态为支持 SOE 功能的 DI 信号时，每当通道值从 TRUE 变为 FALSE，或从 FALSE 变为 TRUE，均会记录一次值跳变，可保存 6000 条事件信息。

当您需要查看历史信息时，可从控制器读取 SOE 信息。SOE 信息将以“.csv”格式被保存在本地路径。

事件信息详见下表。

事件信息	信息说明
跳变时间	事件发生时间，精确到 0.1 毫秒
变量名	通道名称
变量组名	LK631C 模块节点名
变量说明	通道说明
跳变类型	通道值的变化状态，从 0 变为 1 或从 1 变为 0
时钟源质量位	1：时钟源的校时时间可信，SOE 事件以此时间值为准 0：时钟源的校时时间不可信，此时，SOE 事件的记录时间值不准确

控制器工程修改引起的历史事件不可追溯说明：

- LK631C 模块被删除后，之前的历史事件不可追溯。
- 通道变量重命名后，之前的历史事件显示为当前最新通道名。

11.6.2 要求

- LK631C 通道已使能 SOE 功能
- 当前工程已下装
- AutoThinkV4 离线状态

11.6.3 步骤

要查看 SOE 信息，请按以下步骤操作。

（1） 单击【在线】菜单。

（2） 选择【SOE 信息读取】命令。

（3） 将弹出“另存为”对话框。

（4） 输入文件名并选择保存路径。

（5） 单击**保存**按钮。

11.6.4 参考信息

使能 SOE 功能

第12章　运行工程

12.1 概述

工程调试完毕后，需要将控制器投入运行。在运行状态，控制器扫描 I/O 数据、执行用户任务、上报诊断信息。

12.2 要求

工程已下装到控制器

12.3 步骤

要运行控制器工程，请按以下步骤操作：

（1） 将控制器钥匙开关拨到 RUN。

（2） 在“在线”菜单中，选择“在线”命令。

“输出窗口“显示“进入监视状态！”

12.4 结果

控制器工程处于运行状态，通过全局变量组 TaskDiagGroup 查看任务运行状态为运行。

12.5 参考信息

RUN 运行模式

北京和利时智能技术有限公司

Beijing HollySys Intelligent Technologies Co., Ltd..

地址：北京经济技术开发区地盛中路2号院

邮编：100176

电话：010-5898 1588

热线：4008111999

传真：010-5898 1558

http://www.hollysys.com