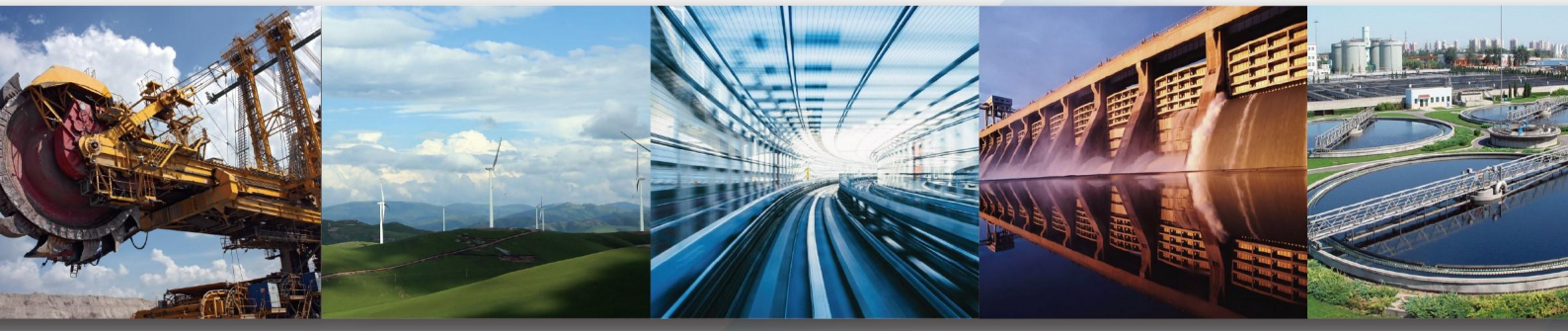
RUNPOWER®

蓝普锋科技

专注PLC研发及产业化

RPC3000 系列 PLC 指令手册



版权声明

前 言

RPC3000 系列 PLC 是北京蓝普锋科技有限公司推出的大型 PLC，包括多种 CPU 模块和扩展模块。同时，蓝普锋公司还推出了丰富的指令系统，方便用户进行控制系统设计。

RPC3000 系列 PLC 以其性能稳定、质量可靠、功能完善、组合灵活、扩展方便、实用性强等优点，广泛应用于中大型工程项目和高端装备配套，赢得了工控领域内广大用户的肯定与好评。

手册主要内容

《RPC3000 系列 PLC 指令手册》对蓝普锋公司研发的 RPC3000 系列 PLC 的指令系统进行了全面、详细的描述，是用户使用 RPC3000 系列 PLC 产品的重要参考资料，该手册包含如下内容：

（1）RPC3000 系列 PLC 基本指令

（2）RPC3000 系列 PLC 扩展指令

读者所需的基本知识

读者需要具备一定的 PLC 基础知识，对 RPC3000 系列 PLC 有一定的了解，并且熟悉 CODESYS 3.5 编程软件的使用方法。（注：关于 CODESYS 3.5 编程软件的使用方法，请参见 RPC 系列 PLC 软件手册。）

本手册适用范围

本手册包含的指令适用于 CODESYS 3.5 编程软件中文版或英文版。

指令手册使用入门

了解和使用本手册，请先按照 RPC 系列 PLC 软件手册对 CODESYS 3.5 编程软件进行安装。在指令学习过程中，需要通过 CODESYS 3.5 编程软件进行操作，为了快速的掌握指令的使用，需要了解和熟悉 CODESYS 3.5 编程软件。

如果已经熟悉 CODESYS 3.5 编程软件，可以直接通过目录查找相应指令。对于 CODESYS 3.5 编程软件的学习和使用请参考 RPC 系列 PLC 软件手册。

相关资料

《RPC 系列 PLC 软件手册》、《RPC3000 系列 PLC 硬件单页说明书》。

服务与支持

用户如果需要进一步帮助，请与蓝普锋公司销售部或技术部联系，联系方式请见本手册尾页。

目录

第一章 概述

蓝普锋公司研发的 RPC3000 系列 PLC 具有丰富的指令系统，通过 CODESYS 3.5 编程软件调用指令。为了使用户快速掌握指令的使用方法，本手册将全面描述 RPC3000 系列 PLC 指令（简称 RPC 指令）的功能及应用方法。

1.1. RPC 指令的特点

指令是 PLC 系统的核心模块 CPU 实现一类功能的一个命令或多个命令组合，指令系统是各类指令的集合。

RPC3000 系列 PLC 的编程语言符合国际标准 IEC（International Electrotechnical Commission）61131-3，故 RPC 指令支持以下六种语言：

（1）梯形图（LD）；

（2）指令表（IL）；

（3）功能块图（FBD）；

（4）顺序功能图（SFC）；

（5）结构文本（ST）；

（6）连续功能图（CFC）。

本文中的指令介绍将主要以 LD、ST 语言做举例说明。

CODESYS 3.5 编程软件为用户提供了丰富的指令，包括数学运算指令、数据转换指令、控制指令、通讯指令等。根据指令的特点，我们将上述指令分为基本指令和扩展指令。基本指令主要是指计算、赋值、逻辑、移位、选择、比较等等，是用户必须了解的指令。扩展指令主要包括通讯指令、硬件时钟指令，以及 RPC Math 库中包含的控制指令与数据处理指令等。

1.2. 软件中安装指令库

打开 CODESYS 3.5 编程软件，菜单栏中点击“工具”/“库存储...”，如图 1-2-1 所示。

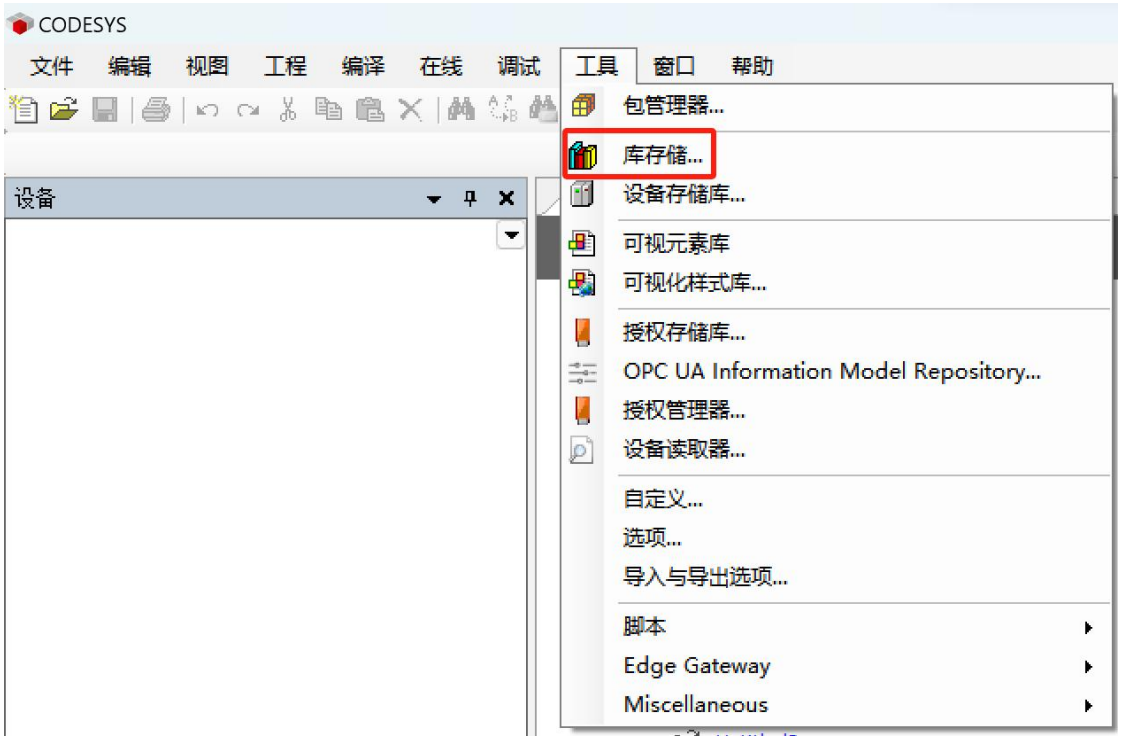


图 1-2-1 指令库安装步骤 1

点击弹出框“安装（I）...”，如图 1-2-2 所示。

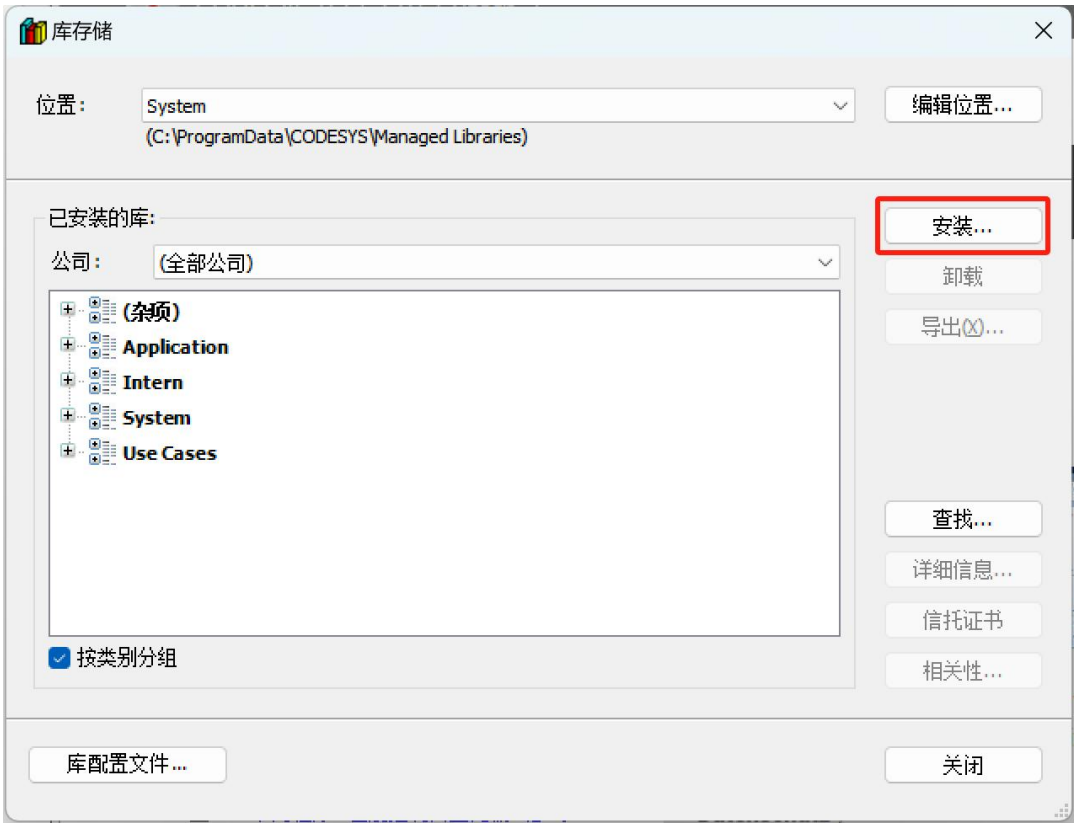


图 1-2-2 指令库安装步骤 2

选择库文件存储文件夹并打开，进行全部安装文件选择，如图 1-2-3 所示。（说明：请根据待安装库文件存放路径对应打开，选择安装文件进行安装。）

小提示：

1. 如果打开文件后无待安装库文件内容，可以修改弹出框右下角文件打开属性，如图 1-2-4 所示，选择“全部文件”。

2. 本章节内容仅为举例说明，文件名称可能与实际使用存在不同，请以实际指令库和软件为准。

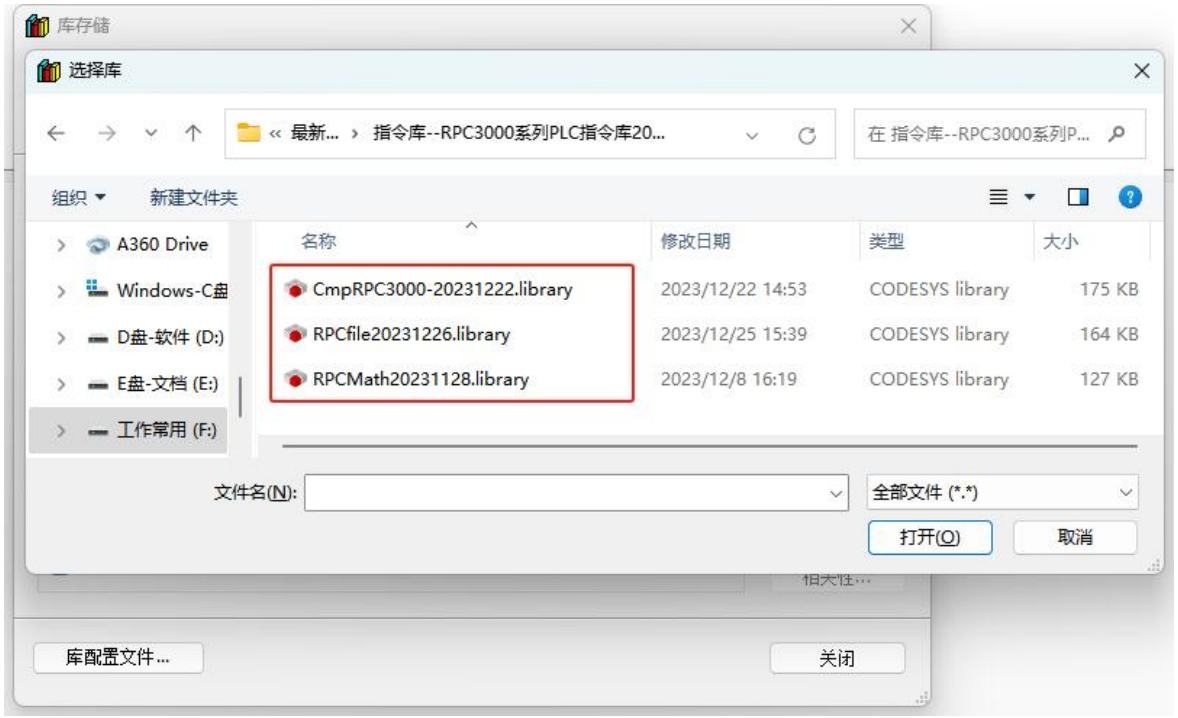


图 1-2-3 指令库安装步骤 3

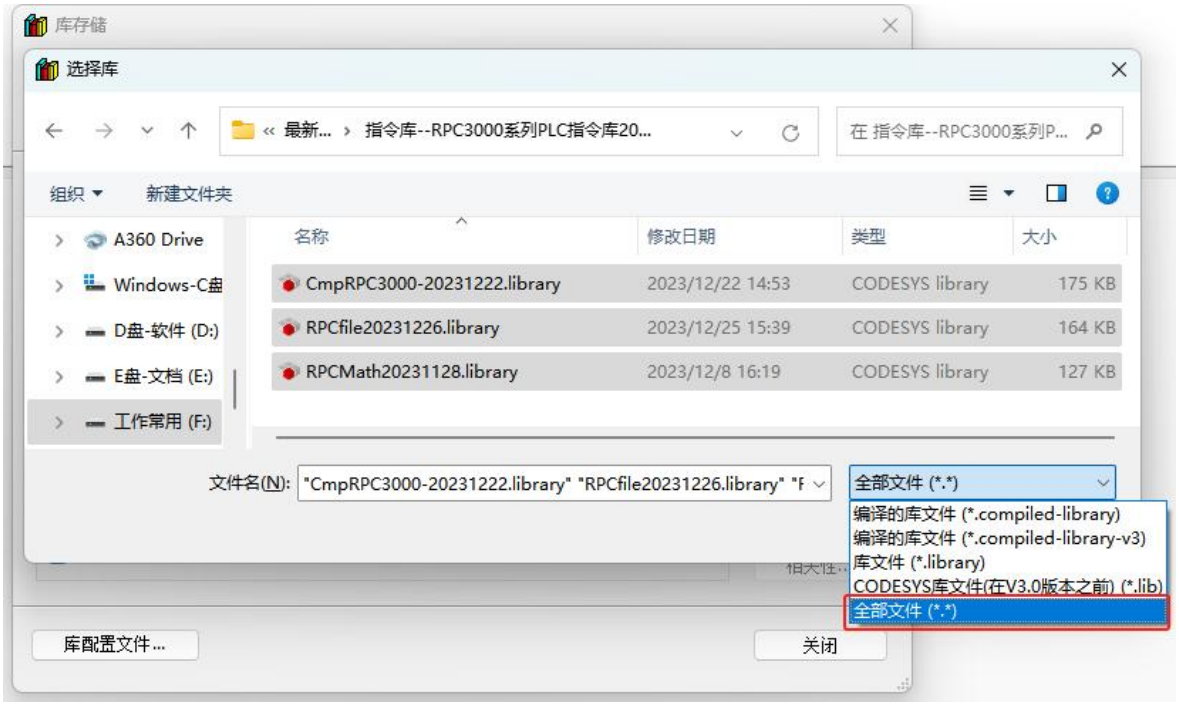


图 1-2-4 指令库安装步骤 4

在弹出框“选择库”中点击“打开（O）”，库文件自动进行安装，安装完成后显示对应库文件名称，如图 1-2-5 所示，点击关闭即可完成安装。也可以选择“卸载（U）”，卸载对应的指令库进行删除。

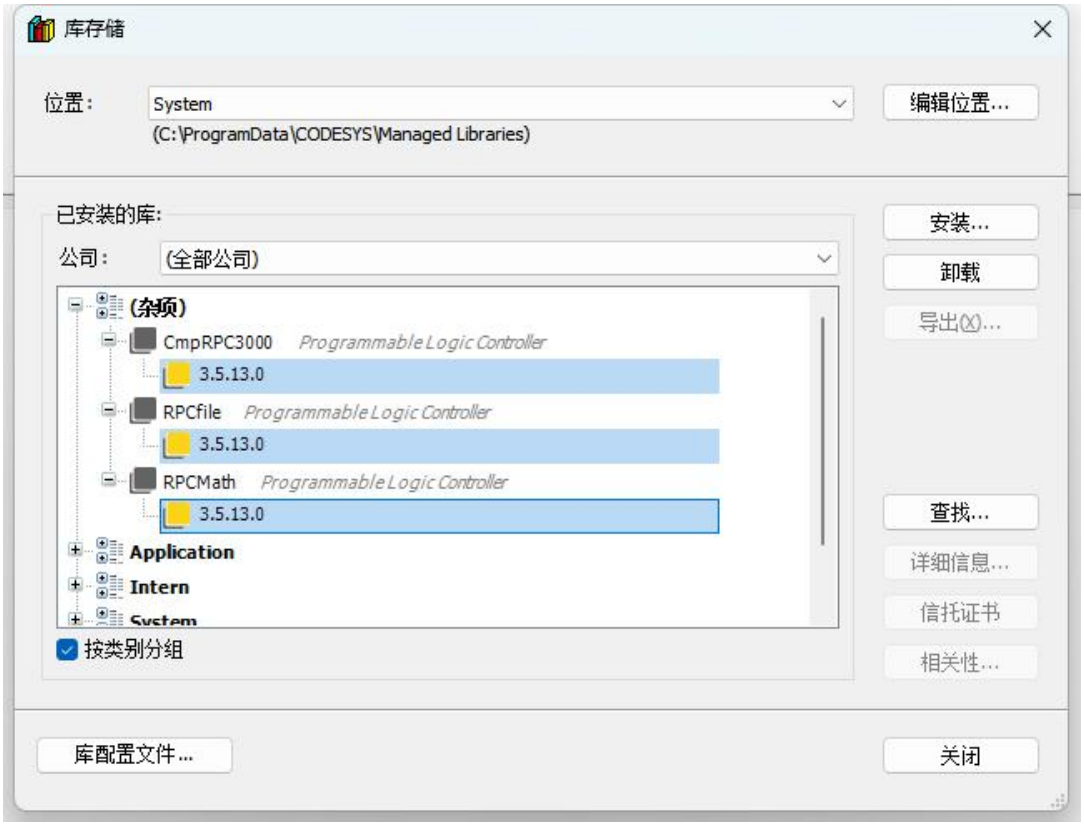


图 1-2-5 指令库安装步骤 5

初次安装编程软件，安装完指令库可能会遇到部分库缺失的情况，通过选择“窗口/库管理器”，即可打开“库管理器”。然后点击菜单栏“下载缺失的库”，如图 1-2-6 所示，点击下载即可完成。

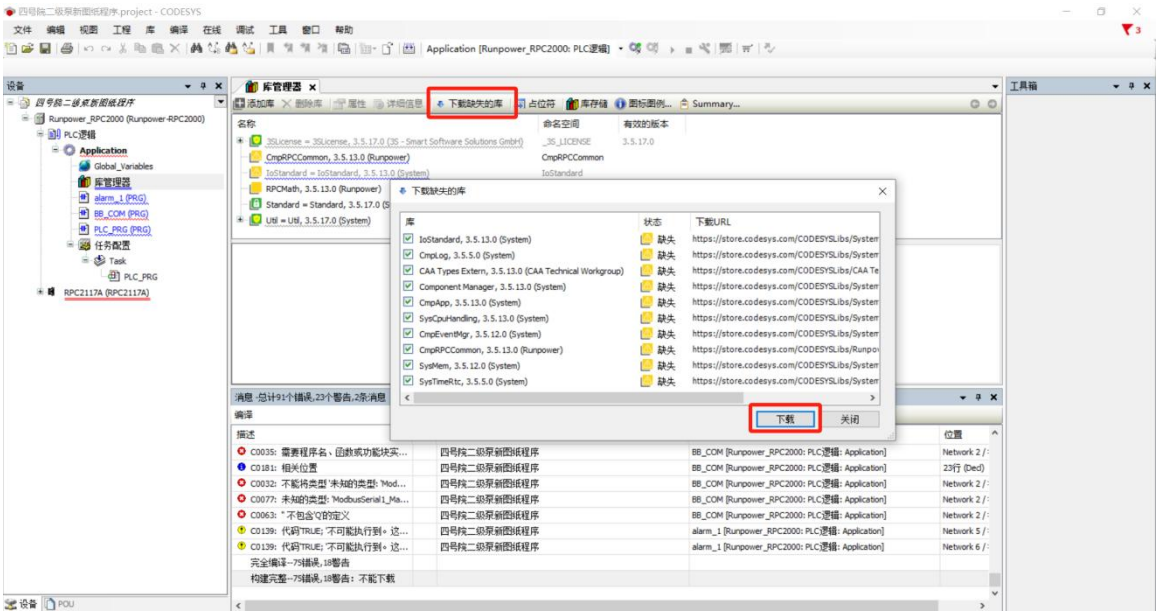


图 1-2-6 指令库安装步骤 6

1.3. 程序中添加指令库

启动 CODESYS 3.5 编程软件后，通过选择“窗口/库管理器”，即可打开“库管理器”。然后点击菜单栏“添加库”或者点击右键，从选项中选择“添加库”，如图 1-3-1 所示。

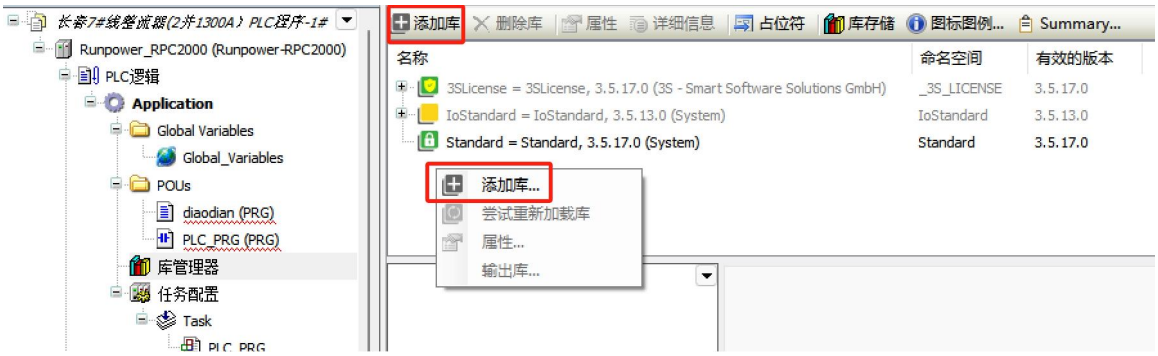


图 1-3-1

根据应用需要选择相应的库文件，如图 1-3-2 所示，点击“打开”即可打开相应的文件。

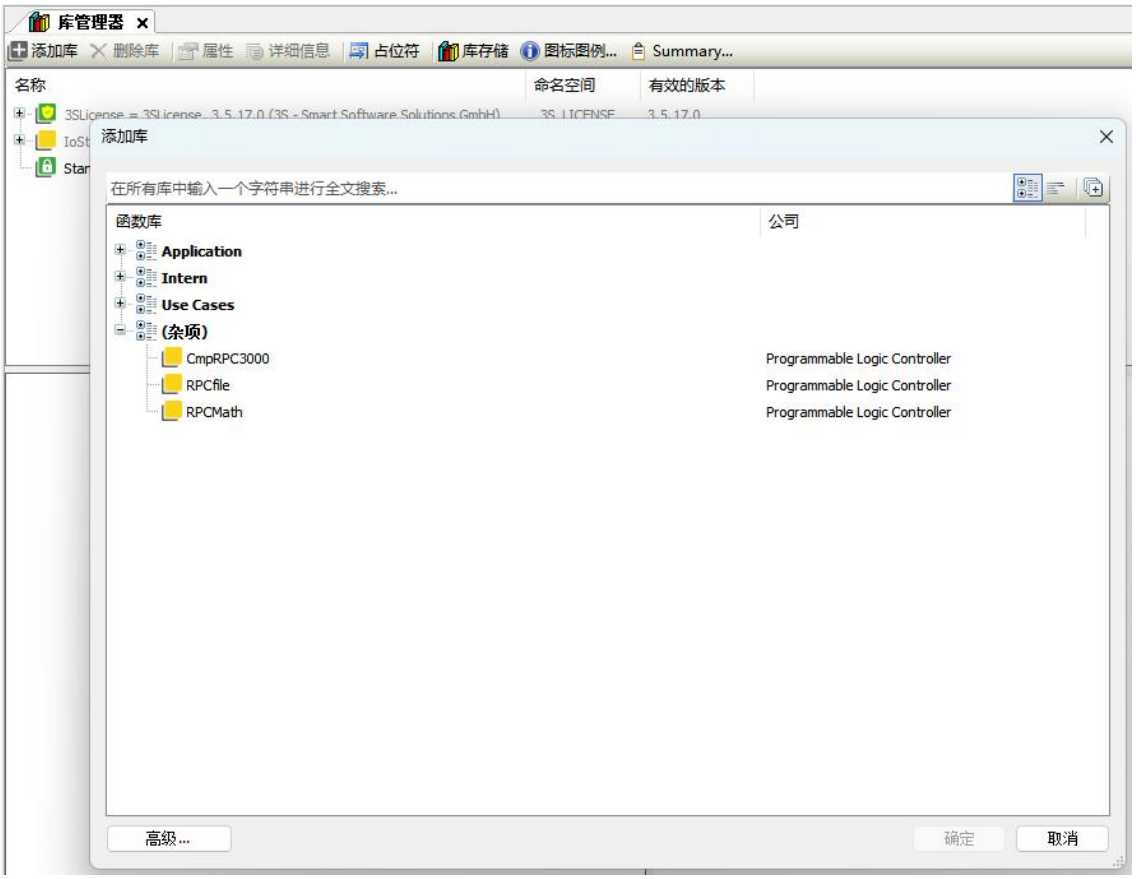


图 1-3-2

于是，选中的库被添加到列表中，如图 1-3-3 所示，库中各类指令显示在库下面的位置。

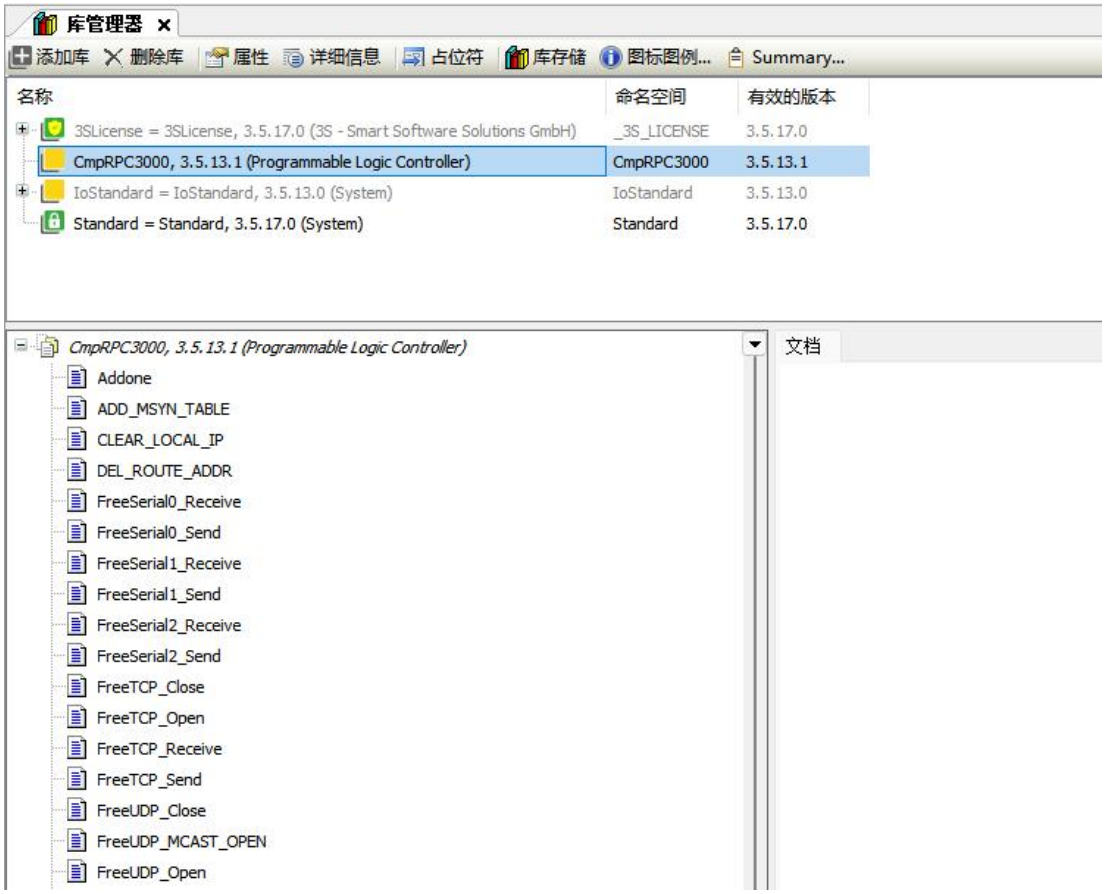


图 1-3-3

1.4. 指令使用须知

1.名词解释：

（1）上电/下装后首次使能有效：当使能端第一次由低电平跃升为高电平，且一直保持高电平，为上电/下装首次使能。在使能有效时，程序执行相关功能。若使能端电平再次由低变高，相关指令不会执行，除非软件重新下装或硬件重新上电。

（2）上升沿使能：当使能端由低电平变为高电平，且一直保持高电平时，为上升沿使能。

2.其它注意事项

（1）程序中使用数学运算指令时，如果输出数据所定义的范围小于运算结果的范围，将导致丢失高位数据。

（2）如果指令的实例名在掉电保持 RETAIN 区中声明，则指令的全部输入变量和输出变量均占用 RETAIN 内存区。为提高 PLC 资源利用效益，在 RETAIN 内存区尽量少声明实例。

（3）在用梯形图（LD）进行程序编制时，插入功能块和插入使能运算符具有不同的指令调用效果。在采用插入功能块调用指令时，无论使能端电平为低或高，系统将以一个输入扫描相应的指令代码。在采用插入使能运算符调用指令时，若使能端电平为低，系统不会扫描相应的指令代码。

第二章 存储区与变量

本章为用户讲述 RPC3000 系列 PLC 的存储区分配、地址寻址方式以及变量使用等概念。通过这些内容可以了解到 RPC3000 系列 PLC 管理数据的方式，也是程序编写的基础。

RPC3000 系列 PLC 的数据存储区共分为五类：I 区（输入区）、Q 区（输出区）、M 区（可寻址寄存器区）、N 区（随机寄存器区）、R 区（掉电保持区）。章节 2.1 会详细描述各存储区的特点和使用方法。

I 区（输入区）、Q 区（输出区）、M 区（可寻址寄存器区）为可寻址的数据区。章节 2.2 将讲述地址的寻址方法和存储规则。

和高级语言类似，RPC3000 系列 PLC 也有常量和变量的概念。所谓的常量就是数值不变的量。章节 2.3 详细讲述了常量的分类和使用方法。

IEC61131-3 标准提出了变量的概念，变量是指程序运行时，其数值可以改变的量。变量用于初始化、存储和数据处理。每个变量都有其固定的数据类型，变量的存储位置可以指定为可寻址的 I 区、Q 区或 M 区，亦可不指定地址，由系统自行分配至 N 区或 R 区。章节 2.4 详细讲述变量的类型和使用方法。

RPC3000 系列 PLC 还支持数组以及自定义数据类型，章节 2.5 和章节 2.6 将详细讲述。

2.1. 存储区分配

RPC3000 系列 PLC 的存储区，CPU 的内存被划分为不同的存储区，每一部分存储区都有各自的特点和使用规则。

存储区包括以下几大类：

输入存储区（I 区）

I 区最大可存储 8K 字节。扩展模块的数字量输入、模拟量输入都有占用输入存储区地址。

输入存储区通过寻址方式访问，可以按位、字节、字、双字访问。具体访问方法请参见 2.2 章节。

输入存储区是只读的，并且不能掉电保持。在仿真模拟时，输入存储区的地址可以被赋值，也可以被强制。但是在在线调试时，输入存储区需要通过接收外部信号赋值。

输出存储区（Q 区）

Q 区最大可存储 8K 字节。扩展模块的数字量输出、模拟量输出都要占用输出存储区地址。

输出存储区通过寻址方式访问，可以按位、字节、字、双字访问。具体访问方法请参见 2.2 章节。

输出存储区的数据是可读写的，并且不能掉电保持。在仿真模拟或者在线调试时，该数据区地址均可以被赋值或强制。

M 存储区

M 存储区是 PLC 的可寻址中间寄存器区，用于存储和管理中间过程产生的数据或状态。与上位机、触摸屏等交互的过程数据，需要定义在 M 存储区。

M 存储区通过寻址方式访问，可以按位、字节、字、双字访问。RPC3000 系列 PLC 的 M 区共有 64MB 字节，按字节来寻址，M 存储区的范围为 MB0~MB67108863。具体访问方法请参见 2.2 章节。

M 存储区的数据是可读写的。在仿真模拟或者在线调试时，该数据区地址均可以被输入或强制。

M 存储区的地址不具有掉电保持功能和自诊断地址，全部可以分配使用。

N 存储区

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，是不可寻址的。

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包括位、字节、字、双字、浮点、时间等。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 存储区大小为 16MB，即可以存储 16MB 的变量。N 存储区的数据是可以读写的，在仿真模拟或者在线调试时，都可以被输入和强制。N 存储区的数据是不能掉电保持的。

R 存储区

R 存储区属于掉电保持区，其调用方式与 N 区一致，也是通过变量的方式访问，无法指定地址。

R 存储区的最大存储空间为 256KB。R 存储区的数据是可以读写的，在仿真模拟或者在线调试时，都可以被输入和强制。

变量定义时，如果没有选择保持功能，则该变量存储在 N 区，若选择了保持功能或直接在保持型变量中定义，则该变量存储于 R 区，具有掉电保持功能，值得一提的是 R 区为永久掉电保持区。关于如何定义掉电保持区变量，详细说明请参见 2.4 章节。

2.2. 地址寻址方式

2.2.1. 地址访问格式

按 IEC61131-3 标准，直接地址以“%”开始，格式为 % 内存区范围 数据格式 地址编号，如下表 2-2-1 所示。

在“内存区范围”位置，字母“I”表示“输入 I 存储区”，字母“Q”表示“输出 Q 存储区”，字母“M”表示“中间 M 存储区”。

在“数据格式”位置，字母“X”表示“位（BIT）”，字母“B”表示“字节（BYTE）”，字母“W”表示“字（WORD）”，字母“D”表示“双字（DWORD）”。

在“地址编号”位置，输入各个存储区范围内数值，进行对应位置寄存器使用。

表 2-2-1 地址使用类型

内存区范围		数据格式	
I	输入区（Input）	X	单个位
Q	输出区（Output）	B	字节(8 位)
M	中间存储区（Memory location）	W	字（16 位）
		D	双字（32 位）

下面以 M 区为例，进行具体说明，如表 2-2-2 所示。

表 2-2-2 M 区地址使用格式

位寻址	格式	%MXm.n
	描述	X 表示是按位寻址； m 表示在 M 存储区中的字节编号； n 表示位于该字节的第几位，范围为 0~7
	数据类型	BOOL

	示例	%MX100.0、%MX105.5、%MX1000.7
字节寻址	格式	%MBm
	描述	B 表示按字节寻址 m 表示在 M 存储区中的字节编号
	数据类型	BYTE
	示例	%MB100、%MB101、%MB1000
字寻址	格式	%MWm
	描述	W 表示按字寻址 m 表示在 M 存储区中的字编号
	数据类型	WORD
	示例	%MW300、%MW301、%MW1000、%MW1001
双字寻址	格式	%MDm
	描述	D 表示按双字寻址 m 表示在 M 存储区中的双字编号
	数据类型	DWORD
	示例	%MD200、%MD201、%MD421、%MD422

对于 I 区、Q 区，则把表中的 M 替换为 I 或 Q 即可，如表 2-2-3 所示。

表 2-2-3 I 区和 Q 区地址使用格式

地址格式	对应地址（位地址 0~7）
%QX7.5	输出区的字节地址 7，第 5 位
%IW4	输入区的字地址 4，1 个字
%QB7	输出区的字节地址 7，1 个字节
%QW4	输出区的字地址 4，1 个字

表 2-2-4、表 2-2-5、表 2-2-6、表 2-2-7 为这三个数据区的范围，超过这个范围的地址视为错误的地址。

表 2-2-4 数据存储区及范围（按位）

存储区	范围（按位）
I 存储区	%IX0.0~%IX8191.7（最大为 8192 字节，具体大小根据 PLC 确定）
Q 存储区	%QX0.0~%QX8191.7（最大为 8192 字节，具体大小根据 PLC 确定）
M 存储区	%MX0.0~%MX67108863.7(最大为 64M 字节，具体大小根据 PLC 确定)

表 2-2-5 数据存储区及范围（按字节）

存储区	范围（按字节）
I 存储区	%IB0~%IB8191（最大为 8192 字节，具体大小根据 PLC 确定）
Q 存储区	%QB0~%QB8191（最大为 8192 字节，具体大小根据 PLC 确定）
M 存储区	%MB0~%MB67108863(最大为 64M 字节，具体大小根据 PLC 确定)

表 2-2-6 数据存储区及范围（按字）

存储区	范围（按字）
I 存储区	%IW0~%IW4095（最大为 4096 字，具体大小根据 PLC 确定）
Q 存储区	%QW0~%QW4095（最大为 4096 字，具体大小根据 PLC 确定）
M 存储区	%MW0~%MW33554431(最大为 8191 字，具体大小根据 PLC 确定)

表 2-2-7 数据存储区及范围（按双字）

存储区	范围（按双字）
I 存储区	部分特殊模块支持，以模块说明为准
Q 存储区	部分特殊模块支持，以模块说明为准
M 存储区	%MD0~%MD16777215

2.2.2. 地址存储映射关系

RPC3000 系列 PLC 的 I 区、Q 区和 M 区是按地址寻址方式访问的，这些存储区都有唯一的、明确的地址。用户可以通过地址寻址方式，即直接使用该存储区地址的方式，来读取和设置该存储区的值。

I 区、Q 区和 M 区的存储格式是一致的，所以这里以 M 区为例说明，如表 2-2-8 和表 2-2-9 所示。RPC3000 系列 PLC 所有直接寻址的存储区，是按字类型 WORD 为单位存储的。每个字（WORD）包含 2 个字节（BYTE），每 1 个字节包含 8 个位（BIT），每 2 个字节（BYTE）组成 1 个字（WORD），每 2 个字（WORD）组成 1 个双字（DWORD）。

表 2-2-8 M 存储区格式 1

位	%MX103.7	.....	%MX103.0	%MX102.7	.....	%MX102.0	%MX101.7	.....	%MX101.0	%MX100.7	.....	%MX100.0
字节	%MB103			%MB102			%MB101			%MB100		
字	%MW51						%MW50					
双字	%MD25											

表 2-2-9 M 存储区格式 2

位	%MX107.7	.....	%MX107.0	%MX106.7	.....	%MX106.0	%MX105.7	.....	%MX105.0	%MX104.7	.....	%MX104.0
字节	%MB107			%MB106			%MB105			%MB104		
字	%MW53						%MW52					
双字	%MD26											

例如： %MX100.0 表示%MB100 的第 0 位，%MX100.7 表示%MB100 的第 7 位。

%MB100 表示有%MX100.0～%MX100.7 八个位组成的字节。

%MW50 表示有%MB100 和%MB101 两个字节组成的字。

%MD25 表示有%MW50 和%MW51 两个字组成的双字。

这些存储区地址中的 BOOL、BYTE、WORD 和 DWORD 类型的数据，可以采用寻址的方式访问。INT、REAL 等其他数据类型，需要使用变量的方式访问，具体内容也请参见 2.4 章节。

要注意，按不同数据类型访问，数据存储区可能是重叠的。诸如%MW50 的数值为 3，则%MB100 的值也为 3，%MX100.0 的值为 TRUE，%MX100.1 的值也为 TRUE。又或者在程序中强制%MX100.0 为 TRUE，因为%MX100.0 是%MB100、%MW50 和%MD25 的第一个位，则同时，这些存储地址的值也被强制为 1。

对于 I 区和 Q 区，也是同样的存储方式。I 区和 Q 区的地址和实际 DI、DO、AI、AO 等点如何对应，请参

见软件手册说明。

2.3. 常量

在 PLC 编程的时候，经常会使用一些数值不变的参数，诸如定时器的时间、固定的比例参数等，这些数值不变的参数称为常量。RPC3000 系列 PLC 支持多种数据类型的常量，常见的常量：布尔型、时间型、数字型等，如表 2-3-1 所示。

表 2-3-1 常量的分类及表示方法

常量类型	表示方法	
布尔型	描述	布尔常量只有两个：逻辑值 TRUE 和 FALSE（也可表示为 1 和 0），TRUE 等价于 1，FALSE 等价于 0。
	示例	TRUE、0
整数型	描述	数字常量的数值可以是二进制、十进制、八进制和十六进制。如果整数值不是十进制值，可以用“进制”加符号“#”放在整数值前面来表示。十进制的 10 至 15 在十六进制中表示为 A 至 F。
	示例	10 (*十进制数 10*) 2#10111011 (*二进制数 10111011*) 8#33 (*八进制数 33*) 16#AB (*十六进制数 AB*)
实数型	描述	实数常量用十进制小数和指数来表示，遵循标准的科学计数法格式。实数常量的数据类型是 REAL、LREAL。
	示例	3.4 (*实数 3.4*) 1.23e+004 (*实数 1.23e+004*)
类型化	描述	在使用常量进行计算时，除实数型常量外，还可以使用类型化的常量。语法：<类型>#<常量>，可以使用的类型：BOOL，SINT， USINT，BYTE， INT，UINT，WORD， DINT，UDINT， DWORD， REAL， LREAL。
	示例	BOOL#1 (*布尔型数 1*) DINT#34 (*双整型数 34*)
时间型	描述	时间常量一般用来操作时间，由“T#”（或“t#”）加上“时间值”构成，时间值的单位包括天（d）、小时（h）、分（m）、秒（s）和毫秒（ms）。注意，它们的正确顺序为 d、h、m、s、ms。
	示例	T#15ms (*15 毫秒的一个时间常量*) T#111s12ms (*111 秒 12 毫秒的一个时间常量，高单位允许超限*） t#12h34m56s (*12 小时 34 分 56 秒的一个常量*) 下面是错误的时间常量： t#1m68s (*低单位不允许超限*) 12ms (*没有 T# *) t#2ms11d (*顺序错误*)
时刻型	描述	时刻常量用于存储当前时刻，由“TOD#”（“tod#”、“TIME_OF_DAY#”或“time_of_day#”）加上“时刻值”构成。时刻值的格式为：小时:分钟:秒（可以用实数形式输入秒）。
	示例	TOD#00:00:00 (*时刻常量为 0 点 0 时 0 分*) TIME_OF_DAY#10:30:30.123 (*时刻常量为 10 点 30 分 30.123 秒*)

日期型	描述	日期常量由“D#”(“d#”、“DATE#”或“date#”)加“日期值”构成。
	示例	DATE#2013-10-01 (*日期常量 2013 年 10 月 1 日*) d#2000-01-01 (*日期常量 2000 年 1 月 1 日*)
日期时刻型	描述	日期常量和时刻常量合并起来称为日期时刻常量，由“DT#”（“dt#”、“DATE_AND_TIME#”或“date_and_time#”）加上“日期时刻值”构成。
	示例	DT#2000-01-01-11:22:33 (*时刻日期常量为 2000 年 1 月 1 日 11 点 22 分 33 秒*) date_and_time#2013-10-01-00:00:00 (*时刻日期常量为 2013 年 10 月 1 日 0 点 0 分 0 秒*)
字符串型	描述	字符串常量在两个单引号之间，可以包含空格和特殊字符。当美元符号（$）位于字符串常量中时，字符将根据 ISO / IEC 8859-1 中的编码解释为十六进制代码，该代码也对应于 ASCII 代码。
	示例	'A and B' (*字符串 A and B *) ':-)' (*字符串:-)*) ‘$41’ (*字符 A*)

小提示：CODESYS 3.5 编程软件不区分大小写，诸如 T#10s 和 t#10s 属于同一常数，TRUE 和 true 均可以表示布尔型常量。

2.4. 变量

所谓变量，就是用字母、数字和下划线组成的一个标识符。

按照数据类型的不同，变量可以分为标准类型和用户自定义类型。其中标准类型包括布尔型（BOOL）、整型（INT）、WORD（字型）、实型（REAL）、字符串型（STRING）以及时间型（TIME）等。自定义类型包括结构体（STRUCT）和枚举(ENUM)。

按照使用范围的不同，变量可以分为全局变量和局部变量。局部变量只在定义该变量的程序中有效，其它程序不能引用。全局变量则可以被整个工程的任意程序引用，在整个工程中均有效。定义变量时，全局变量与局部变量最好不要重名，以免编程时造成混淆。

按照属性的不同，变量分为中间变量、输入型变量、输出型变量、输入输出型变量等。

按照能否掉电保护，变量分为保持型变量和非保持型变量。

2.4.1. 变量命名规则

变量命名必须遵循如下的规则：

☆ 必须以一个字母或者单一的下划线开始，随后是一定数量的字母、数字或下划线，最后以字符或数字结束。变量命名不限制长度，不允许有两个及两个以上连续的下划线，不允许有空格。

☆ 字母与大小写无关，DDBC 和 ddbc 被认为是同一个变量。

☆ 关键字不能用于变量名。CODESYS 3.5 编程软件定义了一些关键字，关键字是标准的标识符，其作用和命名已在系统中自动定义，CODESYS 3.5 编程软件的关键字如表 2-4-1 所示。

除以上命名规则外，CODESYS 3.5 编程软件还可以用中文进行变量名的命名。点击“工程”/“工程设置”/“编译选项”/勾选“允许标识符使用 unicode 字符”进行设置后，可以进行中文变量名称声明使用，如图 2-4-1 所示。

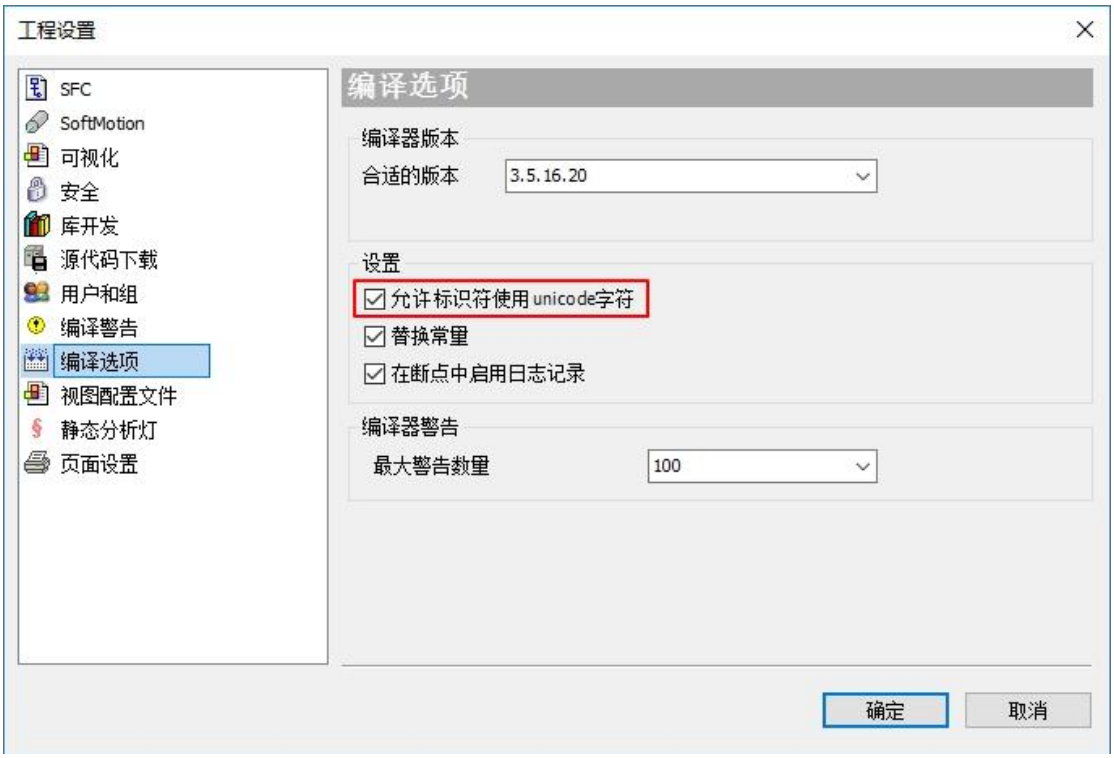


图 2-4-1 工程设置窗口

CODESYS 3.5 编程软件定义了一些关键字，关键字是标准的标识符，其作用和命名已在系统中自动定义，编程软件的部分关键词如表 2-4-1、表 2-4-2 所示。

表 2-4-1 关键字（软件）

ARRAY	FUNCTION	TYPE
AT	FUNCTION_BLOCK	VAR
CONSTANT	OF	VAR_ACCESS
END_FUNCTION	PERSISTENT	VAR_CONFIG
END_FUNCTION_BLOCK	PROGRAM	VAR_EXTERNAL
END_PROGRAM	READ_ONLY	END_GLOBAL
END_STRUCT	READ_WRITE	VAR_IN_OUT
END_TYPE	RETAIN	VAR_INPUT
END_VAR	STRUCT	VAR_OUTPUT

表 2-4-2 关键字（指令）

ABS	COS	INTERFACE
ABSTRACT	END__UNION	INTERNAL
ACOS	EXP	LIMIT
ACTION	EXPT	LN
ADR	EXTENDS	LOG
ASIN	FALSE	LOWER_BOUND
ATAN	FINAL	MAX
BITADR	IMPLEMENTS	METHOD
MIN	MOVE	MUX
ROR	ROL	PARAMS

SEL	REFERENCE	POINTER
SHL	PUBLIC	PRIVATE
SHR	PROTECTED	PROPERTY
SIN	TRUNC	TRUNC_INT
SIZEOF	TRUE	UNION
SQRT	TO	UPPER_BOUND
TAN	THIS	VAR_INST
VAR_STAT	VAR_TEMP	XSIZEOF

2.4.2. 变量数据类型

CODESYS 3.5 编程软件支持的变量数据类型包括标准类型和用户自定义类型。在 PLC 软件中可以查到所支持的标准类型，具体方法如下：选择“编辑”菜单下的“自动声明”，弹出“自动声明”对话框，选择“类型（T）”/“输入助手（I）：”即可，内容如下图 2-4-2、图 2-4-3 所示。

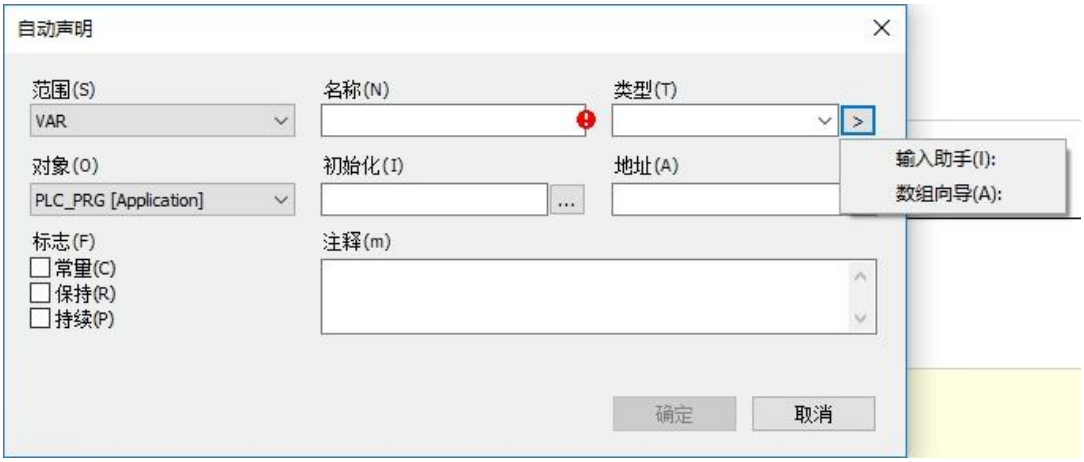


图 2-4-2 自动声明窗口

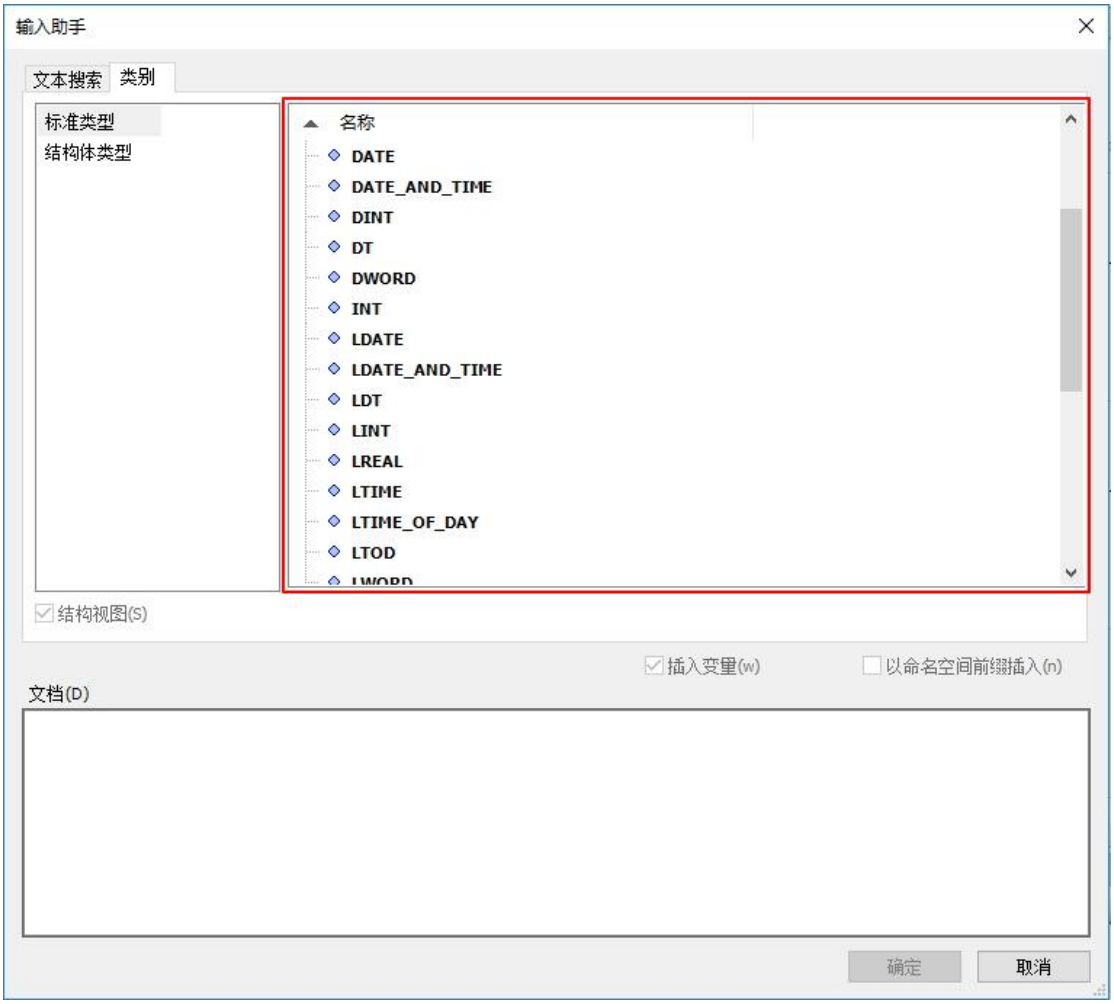


图 2-4-3 输入助手窗口

关于自定义数据类型，请参见 2.6 章节。

CODESYS 3.5 编程软件支持的标准数据类型及范围，如表 2-4-4 所示。

表 2-4-4 变量数据类型

类型	类型名称	数据下限	数据上限	存储空间	备注
BOOL	布尔型	0	1	1bit	
BYTE	字节型	0	255	8 Bit	
WORD	字型	0	65535	16 Bit	
DWORD	双字型	0	4294967295	32Bit	
SINT	短整型	-128	127	8 Bit	
USINT	无符号短整型	0	255	8 Bit	
INT	整型	-32768	32767	16 Bit	
UINT	无符号整型	0	65535	16 Bit	
DINT	长整型	-2147483648	2147483647	32 Bit	
UDINT	无符号长整型	0	4294967295	32 Bit	
REAL	实数型	-3.402823E+38	3.402823E+38	32Bit	单精度浮点数
TIME	时间型			32Bit	示例： Time1 : TIME := t#10s;

TOD	时刻型				示例： Tod1：TOD：= TOD#00:00:00；
DATE	日期型				示例： Date1：DATE：= D#2013-10-01;
DT	日期时刻型				示例： DT1：DT：= dt#2013-10-01-00:00:00;
STRING	字符串型				示例： Str:STRING(35):=‘Run’;
ARRAY	数组				示例： Arr1:ARRAY[1..6]OF BYTE:=1,2,3,4,5,6;

2.4.3. 变量定义

在使用变量之前，必须先对变量进行定义。在定义变量时，不但要定义数据类型，还要定义变量类型，变量类型包括：局部变量，输入变量、输出变量、全局变量等，VAR、VAR_INPUT、VAR_OUTPUT、VAR_IN_OUT、VAR_TEMP、VAR_STAT 、VAR_GLOBAL 为上述类型的标识符，具体说明如图 2-4-4、表 2-4-5 所示。

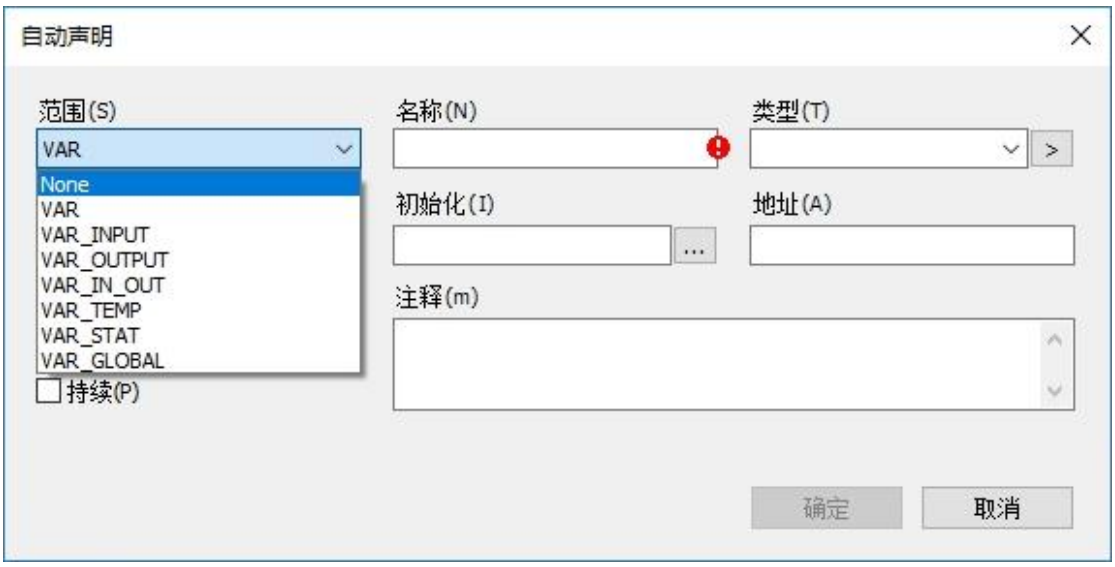


图 2-4-4 选项设置窗口

表 2-4-5 变量类型

标识符	变量类型
VAR	局部变量： 仅在定义该变量的程序中使用，在其余程序中可以定义相同名称的变量，被认为是两个变量
VAR_INPUT	输入变量：作为形参，用于调用程序时的参数传递。在调用程序时，可以将参数通过输入变量传递至子程序或其它 POU 中
VAR_OUTPUT	输出变量：作为形参，用于调用程序时的参数传递。在调用程序时，可以将参数通过输出变量传递至调用该 POU 的程序中
VAR_IN_OUTPUT	输入/输出变量：作为形参，同样用于调用程序时的参数传递，VAR_INPUT 和 VAR_OUTPUT 变量的组合
VAR_TEMP	临时变量：程序和功能块内部存储使用的变量
VAR_STAT	静态变量

VAR_GLOBAL	全局变量：可以在任何子程序中使用的变量。另外，不能定义两个名称相同的全局变量
VAR_EXTERNAL	外部变量：外部变量是导入到程序组织单元中的全局变量
VAR_INST	实例变量
VAR_CONFIG	配置变量
VAR_CONSTANT	常数变量：常量变量在全局变量列表或编程对象的声明部分中声明，在实现中仅以只读方式访问常数变量
VAR_PERSISTENT	持久变量：重新加载应用程序时，持久变量将保留其值。此外，在下载、热启动或冷启动之后，将还原这些值
RETAIN	保持变量，在局部变量、输入变量、输出变量、输入输出变量、全局变量后加关键字 RETAIN，不允许使用 AT 关键字分配输入，输出或内存地址
SUPER	功能块的指针，是一个特殊的变量，用于面向对象的编程
THIS	特殊变量，用于面向对象的编程，是功能块中指向自身属性的指针，可自动用于每个功能块

变量的声明有两种方式：自动声明和手动声明。

自动定义变量

系统支持变量自动定义功能。当程序中出现一个新变量时，系统会自动弹出对话框，要求进行变量定义，如图 2-4-5 所示。其中范围、名称和类型是必须选择或填写的。

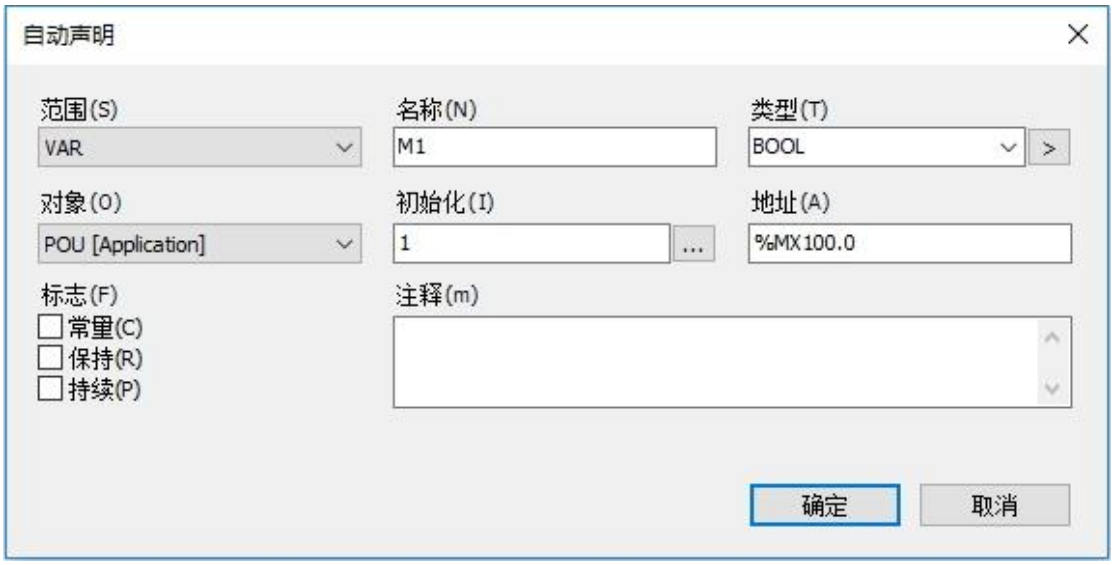


图 2-4-5 变量自动声明窗口

自动定义变量对话框的各项含义，如下所述：

✧ 范围（S）：类型选择。各类型区别请参见表 2-4-5，如：希望定义的变量在所有的 POU 中都能使用，则定义为全局变量，选择类型 VAR_GLOBAL。

✧ 名称（N）：声明变量的名称，即标识符。注意变量命名的规则。

✧ 类型（T）：数据类型选择。可以直接在输入框中输入，也可以点击下拉箭头按钮或右箭头按钮选择输入助手，然后在弹出的对话框中选择数据类型。

✧ 对象（O）：默认选项为“PLC_PRG[Application]”。如果需要定义一个全局变量，需要先添加全局变量表（支持添加多个全局变量表），再通过“对象（O）”进行变量表选择，如图 2-4-6 所示。

✧ 初始化（I）：可以根据需要预先为变量设定一个初始值。

✧ 地址（A）：定义变量的地址。

✧ 注释（m）：变量的说明解释。

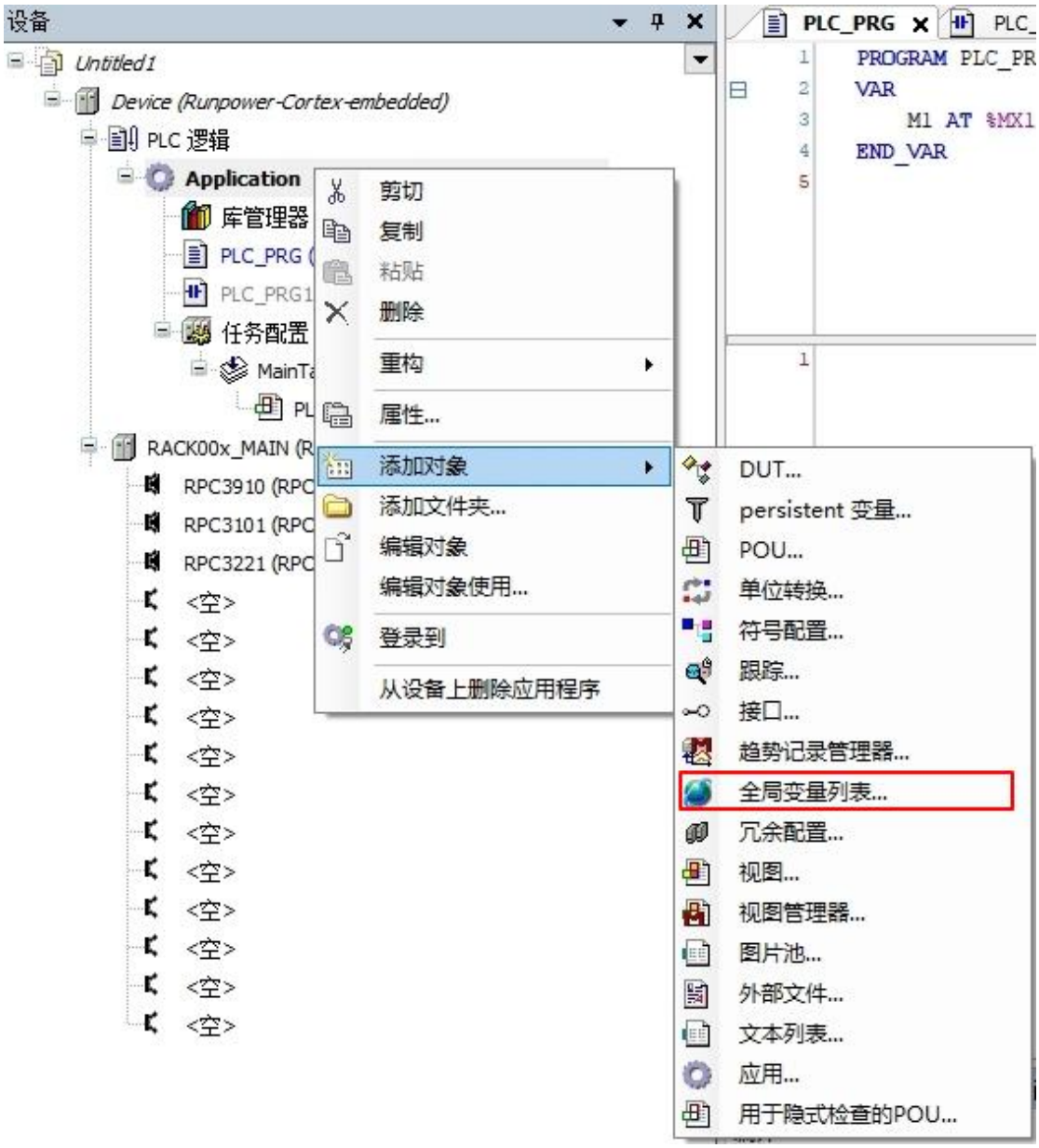


图 2-4-6 添加全局变量表

在自动定义变量时，需要注意以下几点：

✧ 变量可以被指定一个地址，地址的格式与 2.2 章节所讲述的一致。当变量指定地址时，变量存储于该地址所指定的数据区，如图 2-4-5 所示的变量定义，则该变量 M1 存储于 M 存储区，且其地址为%MX100.0。在程序中，采用直接寻址方式改变%MX100.0 中的值，则该变量 M1 的值相应改变。变量定义时，也可以不指定地址，则该变量存储于 N 存储区。

在定义时，可以设置变量的初始值，初始值是一个常量，其类型应与变量的类型一致。诸如定义一时间类型变量，则初始值应是一个时间常量，例如 t#10s。定义初始值后，当 PLC 在上电瞬间，变量被赋值为初始值。

自动定义变量后，将会在变量声明部分会显示刚定义的变量的声明。若定义了图 2-4-5 所示的变量，则在变量声明区有如下声明：

PROGRAM PLC_PRG

```
VAR
    M1 AT %MX100.0: BOOL := 1;
END_VAR
```

若定义的变量是全局变量，则会显示在设备选项中的全局变量表中，而不是在变量声明区。

✧ 变量自动定义时，在自动定义对话框右下角有三个选项：常量、保持和持续。当选择常量，则将该变量作为一个常量，程序中无法再改变其数值。当选择保持型变量时，表示将该变量设置为具有掉电保持功能，该变量存储在 R 存储区。当选择为保持型变量时，可以勾选持续[P]用于持久变量声明。

✧ 新建变量时，系统可以自动定义。但当变量被删除时，定义语句不会自动删除，继续保留在编辑器中，因此要注意变量不能定义重复。

手动声明变量

所谓的手动定义变量，就是不通过自动声明对话框进行定义，而是手动在变量声明区按变量声明的格式和规定添加变量。

变量声明的一般格式：

<变量名> {AT<地址>} : <数据类型> {: = <初始值>};

其中在{ }中的部分是可选的。

定义不同类型的变量，需要在不同的位置进行定义。诸如：定义局部变量，需要在 VAR 和 END_VAR 之间定义；定义输出变量，需要在 VAR_OUTPUT 和 END_VAR 之间定义。

变量声明区可以为表格方式也可以为文本方式，用户也可以在变量声明窗口右上角的文本按钮“  ”和表格按钮“  ”切换变量表的显示方式，如图 2-4-7 所示。表格方式便于排序，文本方式便于大规模复制粘贴。

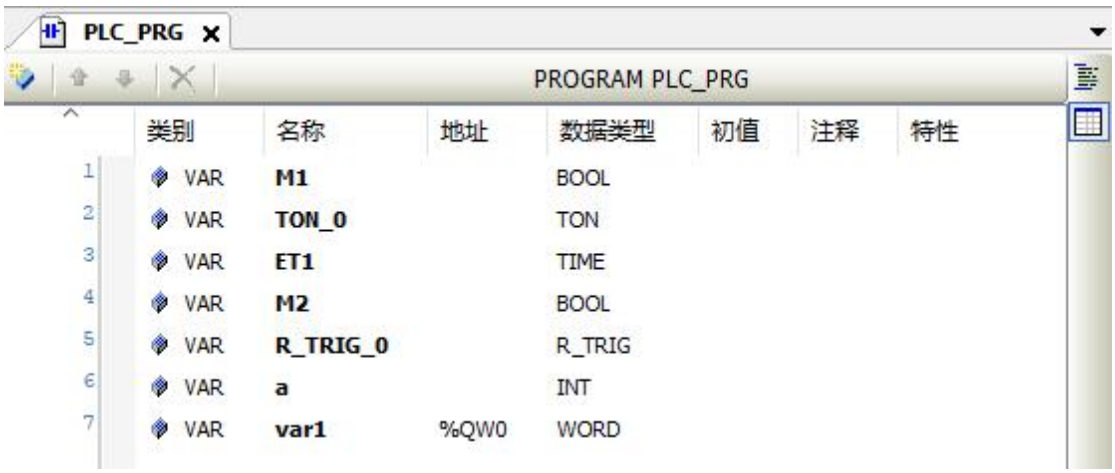


图 2-4-7 变量声明表

变量调用和地址调用方式的区别

变量调用为间接调用地址中的数据，需要预先采用“变量+地址”方式定义变量，与直接地址调用是有区别的。直接地址调用的数据类型可为 BOOL、BYTE、WORD、DWORD 等类型，而用“变量+地址”的方式调用，可以调用地址中 INT、REAL 等更多类型的数据。

例如：需要定义一个 REAL 型变量，其地址为%MD100。若直接使用地址%MD100，其数据类型为 DWORD 型而不是 REAL 型。此时就需要用“变量+地址”的方式定义一个数据类型为 REAL 型的变量，地址为%MD100，从而实现了在%MD100 上定义一个 REAL 型变量。

小提示：在输入变量名称时快捷输入，可以点击“工具”/“选项”/“编码助手”/勾选“键入时立即列出组件”进行设置后，输入变量相关字符后可以通过

下拉框选择对应变量，如图 2-4-8 所示。

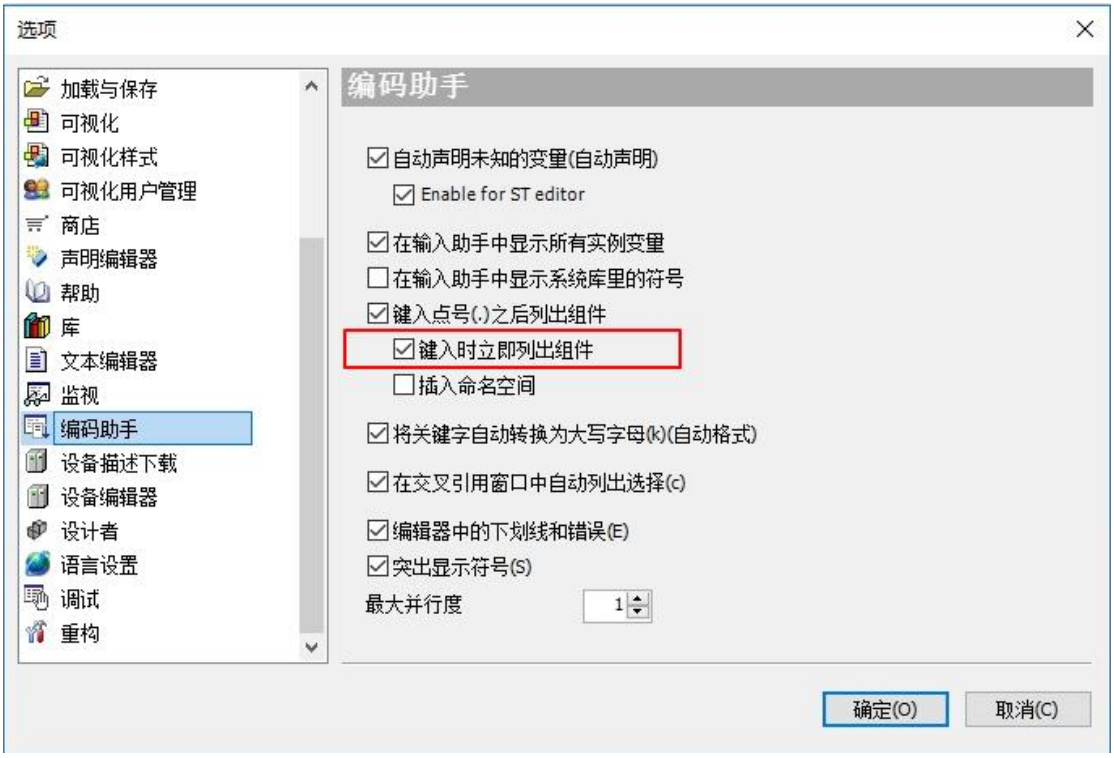


图 2-4-8 选项设置窗口

2.4.4. 保持型变量

在工程应用中，经常需要数据具有掉电保持功能，确保 PLC 断电后数据不丢失。

自动定义变量时，在自动定义对话框的右下角，选择“保持（R）”选项，变量就被自动定义为保持型变量。在手动定义时，将变量定义在 VAR_RETAIN 和 END_VAR 之间，也可以将变量定义为掉电保持变量。若将变量分配在 I、Q、M 区中，即使勾选“保持（R）”选项，变量也不具有掉电保持功能。

2.4.5. 指针变量

指针变量是一类特殊的变量。RPC3000 系列 PLC 中，所有的存储区都占用 CPU 的地址，这个地址被称为绝对地址。不管是 I 区、Q 区、M 区还是 N 区和 R 区，都占用 CPU 的一部分地址。而 I 区、Q 区和 M 区通过寻址访问的地址，诸如%MW200，可以认为是相对地址。指针是指数据区的绝对地址而不是相对地址。

相邻的两个相对地址，其绝对地址也是相邻的。因为 RPC3000 系列 PLC 的存储区是按字节存储的，因此假如%MB200 的绝对地址为 pt，则%MB201 的绝对地址为 pt+1，%MB210 的绝对地址就是 pt+10。

在编程时，假如利用指针的这种关系，可以很方便地实现一些比较复杂的功能。在使用之前，同样需要定义指针变量。指针变量的定义与其他数据类型定义类似，只是其数据类型为 POINTER TO <数据类型>。

指针定义的语法格式：

<指针名> : POINTER TO <数据类型/功能块>;

例如：

```
pt1:POINTER TO WORD;          (*定义一个字型数据的指针 pt1*)
Var_word1:WORD :=100;         (*定义字型变量 Var_word1，使其等于 100*)
Var_word2:WORD;               (*定义字型变量 Var_word2*)
pt1 := ADR(Var_word1);        (*取出 Var_word1 变量的地址，将地址值赋给 pt1*)
```

Var_word2:= pt1^; (*将指针 pt1 所指地址的值赋给 Var_word2，Var_word2=100*)

举例说明指针的用法：

在 M 数据区的%MB100 开始的地址中，存放了 100 个 BYTE 型变量，如果需要将这些值转移至%MB300 开始的 100 个字节内，进行批量赋值。

因为数据区比较长，采用指针方式，可以很方便地实现数据区的转移。

这里需要定义两个指针变量 pt1 和 pt2，一个指向%MB100，另一个指向%MB300。这里还需要用到一个取地址指令 ADR 和读取指针数值指令^，关于这两个指令的详细信息，请查看指令说明。

变量定义如下：

```
VAR
    m: INT;
    pt1: POINTER TO BYTE;
    pt2: POINTER TO BYTE;
END_VAR
```

其中变量 m 用于传输 100 个字节。

在这里采用 ST 方式编写程序，关于 ST 语言编程，请参见软件手册。

采用一个 FOR 循环语句，每次循环，pt1 和 pt2 的值均加 1（因为是 BYTE 类型指针），然后将 pt1 的值赋值给 pt2 即可。

具体程序如下：

```
pt1:=ADR(%MB100);
pt2:=ADR(%MB300);
FOR m:=1 TO 100 BY 1 DO
pt2^:=pt1^;
pt1:=pt1+1;
pt2:=pt2+1;
END_FOR
```

2.5. 数组

CODESYS 3.5 编程软件对数据的管理功能是非常强大的。不但支持多种数据类型，也支持多维数据。在编程时，可以根据基本数据类型来定义一维、二维和三维数组。数组可以采用自动定义，也可以在变量声明区手动定义。

数组的标识符为 ARRAY。数组定义的语法格式：

<数组名> : ARRAY [<L1>..<U1>, <L2>..<U2>, <L3>..<U3>] OF <基本数据类型>;

其中 L1、L2 和 L3 表示字段范围的最小值，U1、U2 和 U3 表示字段范围的最大值。字段范围必须是整数。假如是一维数组，则只需设置 L1 和 U1 即可；假如是二维数组，则需要设置 L1、U1 和 L2、U2；假如是三维数组，则 L1、U1、L2、U2 和 L3、U3 均需定义。

图 2-5-1 为定义一个数组元素数量为 8 的一个三维数组的自动定义对话框：

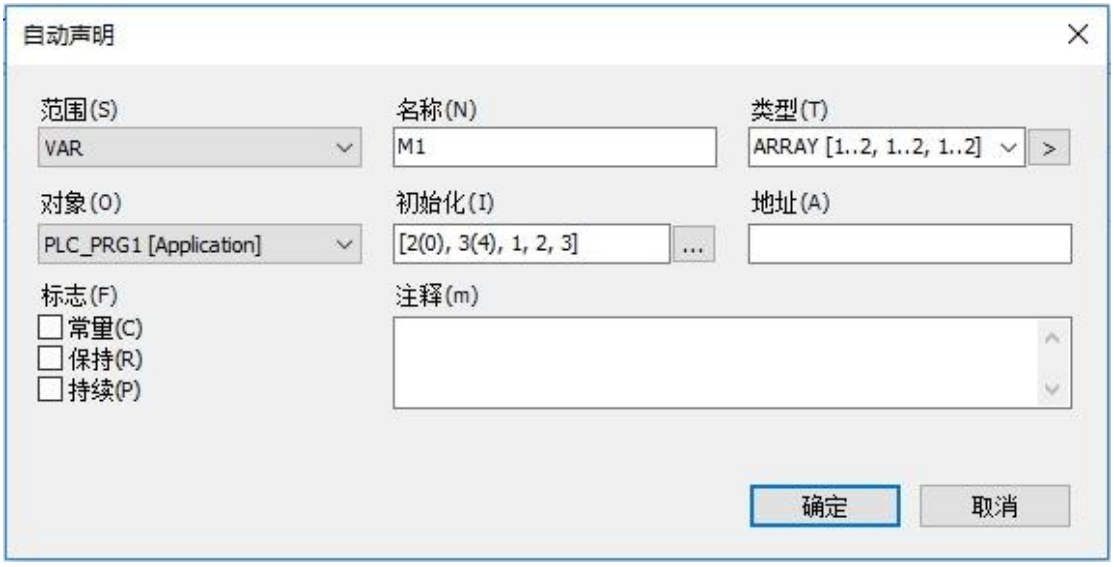


图 2-5-1 数组自动定义

在数组定义的同时，给数组中的元素赋值称为初始化数组。在数组定义时，可以初始化数组中所有元素，也可以不进行初始化。可以通过“初始化（I）”中“...”进行初始值输入，有可以按格式进行文本输入。

举例 1：数组的完全初始化

Arr1:ARRAY [1..7] OF BYTE:=[0, 1,2,3,4,5,6];

Arr2:ARRAY [1..2,1..3] OF INT := [1,2,3,3(5)] ; （*即 1,2,3,5,5,5 的缩写形式*）

Arr3:ARRAY [1..2,1..2,1..2] OF INT := [2(0),3(4),1,2,3]; （*即 0,0,4,4,4,1,2,3 的缩写形式*）

举例 2：结构数组定义，在设备选项卡中的“Application”右击“添加”/“DUT”中定义结构体 STRUCT1，如下图 2-5-2 所示：

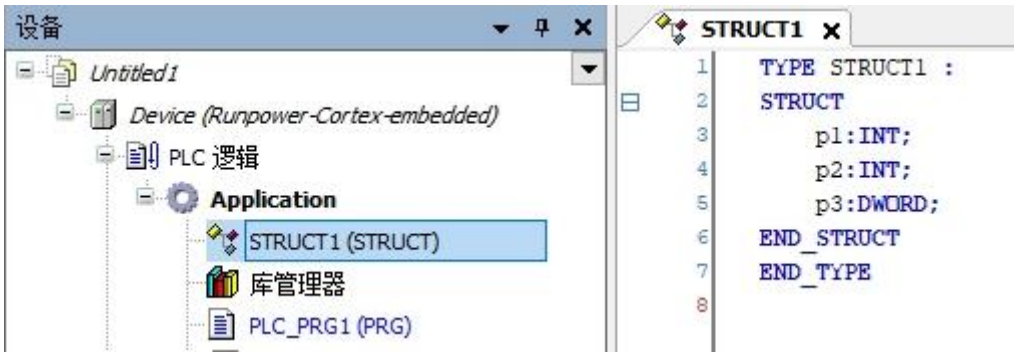


图 2-5-2 结构体定义

对结构数组进行定义和初始化，关于结构体声明详见 2.6 章节，举例如下：

A1: ARRAY[1..3] OF STRUCT1 := [(p1 := 1, p2 := 2, p3 := 3), (p1 := 4, p2 := 5, p3 := 6), (p1 := 7, p2 := 8, p3 := 9)];

举例 3：数组的部分初始化

Arr1:ARRAY [1..10] OF BYTE:= [1,2,3,4,5];

对于那些没有预先赋值的元素，按照基本数据类型的缺省初始值进行初始化。在此例中，元素[6]到[10]被初始化为 0。

2.6. 自定义数据类型

在一些实际应用场合，需要用到一些配方数据。每组配方包含很多参数，而这些参数的数据类型是不同的，诸如可能包含 REAL 型、WORD 型或 TIME 型等，那么这里使用自定义数据类型就能非常方便地实现配方数据

的管理。在“Application”右击“添加”/“DUT”中定义结构体，新建一个数据类型，命名规则同变量命名规则。命名完之后，该标志符就可以作为一个结构用来表示这个数据类型，通过在数据类型上添加对象来创建一个新的结构变量。

结构变量以关键字 TYPE 和 STRUCT 开始，以关键字 END_STRUCT 和 END_TYPE 结束。

定义结构的语法格式：

TYPE <结构名>:

STRUCT

<变量声明 1>

<变量声明 2>

…

<变量声明 n>

END_STRUCT

END_TYPE

<结构名>是一种可以在整个工程中被识别的数据类型，可以像标准数据类型一样引用，但不可以指定结构中变量的地址（即变量名之后不允许使用 AT 来指定该变量的地址）。

举例：

定义名为 Polygonline 的结构：

TYPE Polygonline:

STRUCT

Start:ARRAY [1..2] OF INT;

Point1:ARRAY [1..2] OF INT;

Point2:ARRAY [1..2] OF INT;

Point3:ARRAY [1..2] OF INT;

Point4:ARRAY [1..2] OF INT;

End:ARRAY [1..2] OF INT;

END_STRUCT

END_TYPE

结构体初始化及变量定义：

Poly_1: polygonline := (Start := [1, 2], Point1 := [3, 4], Point2 := [5, 6], Point3 := [6, 5], Point4 := [4, 3], End := [2, 1]);

Start1: INT;

结构体调用：

Start1:=Poly_1.Start[1];

结构成员的访问

<结构名>.<结构成员名>

举例：如果结构名为“Poly_1”，其中一个成员名为“Start”，则可以用下面的形式访问：Poly_1. Start

程序中使用举例如下图 2-6-1 所示。

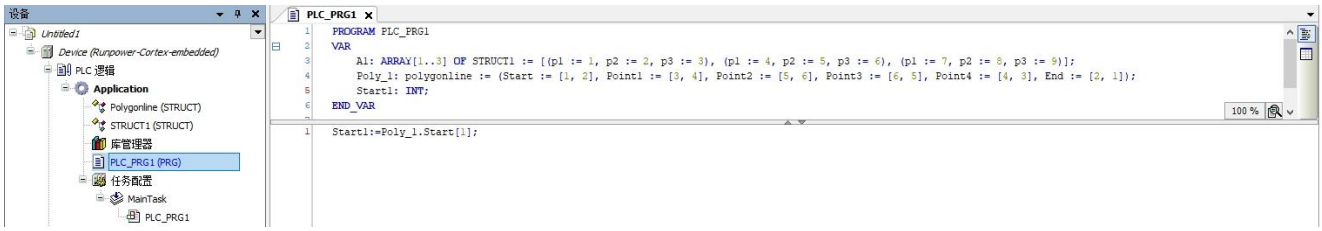
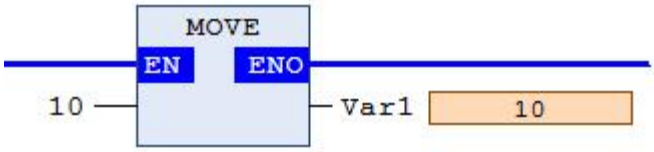


图 2-6-1 结构体定义举例

第三章 指令系统

3.1. 赋值指令

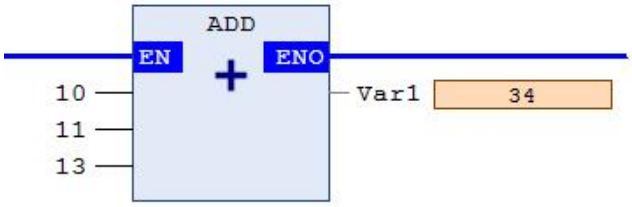
赋值指令 MOVE 用于将一个常量或者变量的值赋值给另外一个变量。

指令对应的输入和输出数据类型	
输入和输出的数据类型为 BOOL、BYTE、DATE、DATE_AND_TIME(DT)、DINT、DWORD、INT、LDATE、LDATE_AND_TIME(LDT)、LINT、LREAL、LTIME、LTIME_OF_DAY(LTOD)、LWORD、REAL、SINT、STRING、TIME、TIME_OF_DAY(TOD)、UDINT、UINT、USINT、WORD、WSTRING、ARRAY。	
变量定义	
类别 名称 地址 数据类型 初值 注释 特性 1 VAR Var1 INT	
编程语言	程序
梯形图（LD）	 (*结果 Var1 为 10*)
结构化文本（ST）	Var1:=10; (*结果 Var1 为 10*)

3.2. 算术运算指令

3.2.1. ADD——加法指令

加法指令用于对两个及两个以上的常量或变量，进行相加运算。

指令对应的输入和输出数据类型	
输入和输出的数据类型为 BOOL、BYTE、DATE、DATE_AND_TIME(DT)、DINT、DWORD、INT、LDATE、LDATE_AND_TIME(LDT)、LINT、LREAL、LTIME、LTIME_OF_DAY(LTOD)、LWORD、REAL、SINT、STRING、TIME、TIME_OF_DAY(TOD)、UDINT、UINT、USINT、WORD、WSTRING、ARRAY。	
变量定义	
类别 名称 地址 数据类型 初值 注释 特性 1 VAR Var1 INT	
编程语言	程序
梯形图（LD）	

	(*结果 Var1 为 34*)
结构化文本（ST）	Var1:=10+11+13; (*结果 Var1 为 34*)

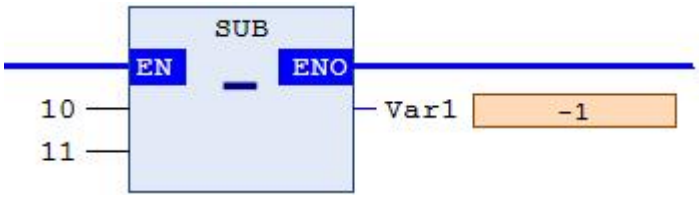
提示：

1. ADD 加法指令默认有两个输入端填写输入数据，可以在输入端处添加输入。

2. TIME 数据不同类型相加时，TIME+TIME = TIME，TOD+TIME = TOD，DT+TIME = DT，LTIME+LTIME = LTIME，LTOD+LTIME = LTOD，LDT+LTIME = LDT。

3.2.2. SUB——减法指令

减法指令用于对两个常量或变量，进行相减运算。

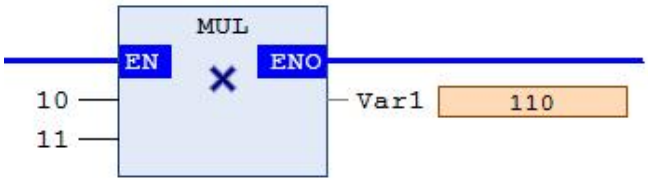
指令对应的输入和输出数据类型	
输入和输出的数据类型为 BOOL、BYTE、DATE、DATE_AND_TIME(DT)、DINT、DWORD、INT、LDATE、LDATE_AND_TIME(LDT)、LINT、LREAL、LTIME、LTIME_OF_DAY(LTOD)、LWORD、REAL、SINT、STRING、TIME、TIME_OF_DAY(TOD)、UDINT、UINT、USINT、WORD、WSTRING、ARRAY。	
变量定义	
类别 名称 地址 数据类型 初值 注释 特性 1 VAR Var1 INT	
编程语言	程序
梯形图（LD）	 (*结果 Var1 为-1*)
结构化文本（ST）	Var1:=10-11; (*结果 Var1 为-1*)

提示：TIME 数据类型的输入数据与其它输入数据相减时，TIME-TIME = TIME、DATE-DATE = TIME、TOD-TIME = TOD、TOD -TOD = TIME、DT -TIME = DT、DT -DT = TIME，LTIME+LTIME = LTIME，LTOD+LTIME = LTOD，LDT+LTIME = LDT。

3.2.3. MUL——乘法指令

乘法指令用于对两个及两个以上的常量或变量，进行相乘运算。

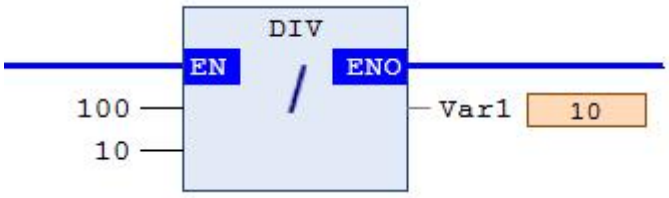
指令对应的输入和输出数据类型	
输入和输出的数据类型为 BYTE、DINT、DWORD、INT、LINT、LREAL、LWORD、REAL、SINT、UDINT、UINT、USINT、WORD。	
变量定义	
类别 名称 地址 数据类型 初值 注释 特性 1 VAR Var1 INT	
编程语言	程序

梯形图 （LD）	 (*结果 Var1 为 110*)
结构化文本 （ST）	Var1:=10*11; (*结果 Var1 为 110*)

提示：MUL 乘法指令默认有两个输入端填写输入数据，可以在输入端处添加输入。

3.2.4. DIV——除法指令

除法指令用于对两个常量或变量，进行相除运算。

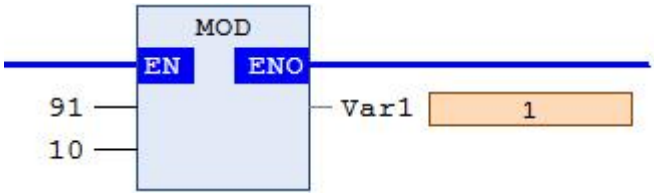
指令对应的输入和输出数据类型	
输入和输出的数据类型为 BYTE、DINT、DWORD、INT、LINT、LREAL、LWORD、REAL、SINT、UDINT、UINT、USINT、WORD。	
变量定义	
	
编程语言	**程序**
梯形图 （LD）	 (*结果 Var1 为 10*)
结构化文本 （ST）	Var1:=100/10; (*结果 Var1 为 10*)

提示：使用除法指令时需要注意不要出现除数为 0 的情况，除数为 0 会根据目标系统产生意料之外的结果。

3.2.5. MOD——取余指令

取余指令用于对一个变量或常量进行相除取余，其结果是一个整数。

指令对应的输入和输出数据类型	
输入和输出的数据类型为 BYTE、DINT、DWORD、INT、LINT、LREAL、LWORD、REAL、SINT、UDINT、UINT、USINT、WORD。	
变量定义	
	
编程语言	**程序**

梯形图 （LD）	 (*结果 Var1 为 1*)
结构化文本 （ST）	Var1:=91 MOD 10; (*结果 Var1 为 1*)

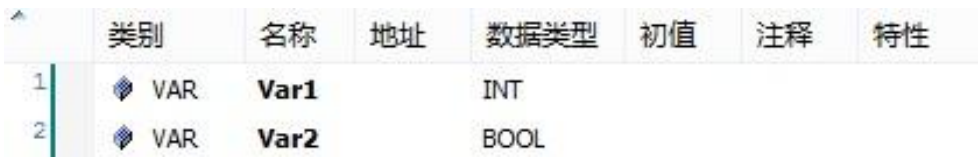
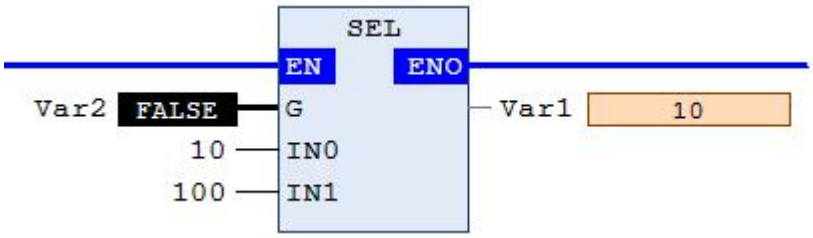
提示：对零取余后，结果为 0。

3.3. 选择指令

所有选择指令均可接受变量或常量输入。需要注意，输出数据的数据类型存储长度必须大于等于输入数据的数据类型类型存储长度。

3.3.1. SEL——二选一指令

二选一指令的格式为：OUT:=SEL(G, IN0, IN1)，其中 G 为控制位，IN0 和 IN1 分别为两个输入数据，OUT 为输出数据。通过控制位的值，选择两个输入数据中的其中一个作为输出数据，控制位的值为 0 时选择第一个输入数据，控制位的值为 1 时选择第二个输入数据。

指令对应的输入和输出数据类型	
该指令中控制位为 BOOL。 IN0、IN1 和 OUT 的数据类型为 BOOL、BYTE、DATE、DATE_AND_TIME(DT)、DINT、DWORD 、 INT 、 LDATE 、 LDATE_AND_TIME(LDT) 、 LINT 、 LREAL 、 LTIME 、 LTIME_OF_DAY(LTOD)、LWORD、REAL、SINT、STRING、TIME、TIME_OF_DAY(TOD)、UDINT、UINT、USINT、WORD、WSTRING、ARRAY。	
变量定义	
	
编程语言	**程序**
梯形图 （LD）	 (* Var2 为 FALSE，选择对应 IN0，Var1 结果为 10*)
结构化文本 （ST）	Var1:=SEL(Var2,10,100); (*Var1 结果为 10 *)

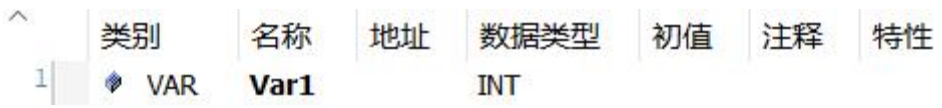
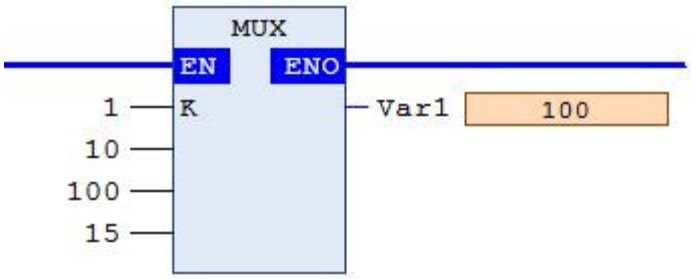
提示：

1.输入参数引脚 IN0, IN1 和输出参数眼角 OUT 需要使用相同的数据类型，特别是使用用户定义的数据类型时。

2.当控制位的值 G 为 1 时，软件不计算 IN0 的表达式。当控制位的值 G 为 0 时，软件不会计算 IN1 的表达式。

3.3.2. MUX——多选一指令

多选一指令 MUX 的功能是根据控制数的不同，选择两个及两个以上输入数据中的一个值作为输出。该指令格式为：OUT:=MUX(K,IN0,…,INn)，其中 K 为控制数，输入数据为 IN0~INn，OUT 为输出数据，输出的值等于输入数据 INk。

<table>
<tr><th colspan="2">指令对应的输入和输出数据类型</th></tr>
<tr><td colspan="2">输入和输出的数据类型为 BOOL、BYTE、DATE、DATE_AND_TIME(DT)、DINT、DWORD、INT、LDATE、LDATE_AND_TIME(LDT)、LINT、LREAL、LTIME、LTIME_OF_DAY(LTOD)、LWORD、REAL、SINT、STRING、TIME、TIME_OF_DAY(TOD)、UDINT、UINT、USINT、WORD、WSTRING、ARRAY。
控制数 K 必须是下面类型中的一种：BYTE、DINT、DWORD、INT、LINT、LWORD、SINT、UDINT、UINT、USINT、WORD。</td></tr>
<tr><th colspan="2">变量定义</th></tr>
<tr><td colspan="2"> 
</td></tr>
<tr><th>编程语言</th><th>程序</th></tr>
<tr><td>梯形图（LD）</td><td> 

(*Var1 结果为 100*)</td></tr>
<tr><td>结构化文本（ST）</td><td>Var1:=MUX(1,10,100,15);
(*Var1 结果为 100 *)</td></tr>
</table>

提示：

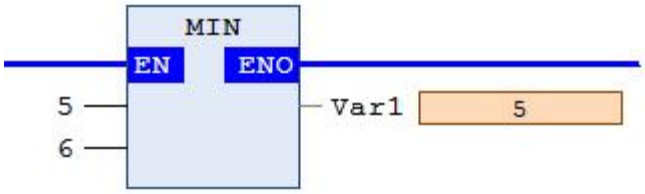
1. **MUX 多选一指令默认有两个输入端填写输入数据，可以在输入端处添加输入。**
2. **输入数据 IN0, ..., INn 和输出数据 OUT 需要使用相同的数据类型，特别是使用用户定义的数据类型时。**
3. **MUX 从一组输入参数引脚 IN0, ..., INn 中选择一个作为输出数据，控制数为 K，若选择第一个输入数据作为输出时，K=0。若 K 比 n 值大，那么软件传递输入数据的最后一个值(INn)。**
4. **出于对运行优化的考虑，软件只计算输入中的第 K 个表达式。**

3.3.3. MIN——取最小值指令

MIN 指令的功能为取出两个及两个以上输入数据中的最小值作为输出结果。其格式为：OUT:=MIN(IN0,…,INn)，IN0,…,INn 为输入数据，OUT 为输出数据。

<table>
<tr><th>指令对应的输入和输出数据类型</th></tr>
<tr><td>输入和输出的数据类型为 BOOL、BYTE、DATE、DATE_AND_TIME(DT)、DINT、DWORD、INT、LDATE、LDATE_AND_TIME(LDT)、LINT、LREAL、LTIME、LTIME_OF_DAY(LTOD)、LWORD、REAL、SINT、STRING、TIME、TIME_OF_DAY(TOD)、UDINT、UINT、USINT、WORD、WSTRING。</td></tr>
<tr><th>变量定义</th></tr>
</table>

	类别	名称	地址	数据类型	初值	注释	特性
1	VAR	Var1		INT			

编程语言	程序
梯形图（LD）	 (*Var1 结果为 5 *)
结构化文本（ST）	Var1:=MIN(5,6); (*Var1 结果为 5 *)

提示：MIN 取最小值指令默认有两个输入端填写输入数据，可以在输入端处添加输入。

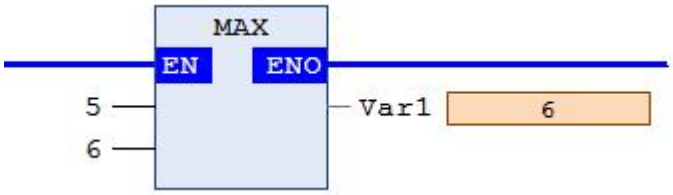
3.3.4. MAX——取最大值指令

MAX 指令的功能为取出两个及两个以上输入数据中的最大值作为输出结果。其格式为：OUT:=MAX (IN0,…,INn)，IN0,…,INn 为输入数据，OUT 为输出数据。

指令对应的输入和输出数据类型
输入和输出的数据类型为 BOOL、BYTE、DATE、DATE_AND_TIME(DT)、DINT、DWORD、INT、LDATE、LDATE_AND_TIME(LDT)、LINT、LREAL、LTIME、LTIME_OF_DAY(LTOD)、LWORD、REAL、SINT、STRING、TIME、TIME_OF_DAY(TOD)、UDINT、UINT、USINT、WORD、WSTRING。

变量定义

	类别	名称	地址	数据类型	初值	注释	特性
1	VAR	Var1		INT			

编程语言	程序
梯形图（LD）	 (*Var1 结果为 6 *)
结构化文本（ST）	Var1:=MAX(5,6); (*Var1 结果为 6 *)

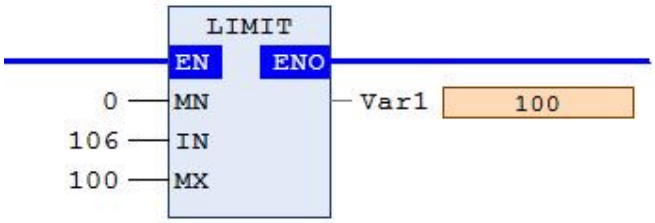
提示：MAX 取最大值指令默认有两个输入端填写输入数据，可以在输入端处添加输入。

3.3.5. LIMIT——极限值指令

极限值指令 LIMIT 用于控制输入数据在一定范围内，其格式为 OUT:=LIMIT(Min,IN,Max)。若输入数据介于最小值和最大值之间，输出数据等于输入数据；若输入数据小于最小值，输出数据等于最小值；若输入数据大于最大值，输出数据等于最大值。

指令对应的输入和输出数据类型	
输入和输出的数据类型为 BOOL、BYTE、DATE、DATE_AND_TIME(DT)、DINT、DWORD、INT、LDATE、LDATE_AND_TIME(LDT)、LINT、LREAL、LTIME、LTIME_OF_DAY(LTOD)、LWORD、REAL、SINT、STRING、TIME、TIME_OF_DAY(TOD)、UDINT、UINT、USINT、WORD、WSTRING。	
变量定义	

	类别	名称	地址	数据类型	初值	注释	特性
1	VAR	Var1		INT			

编程语言	程序
梯形图（LD）	 (*Var1 结果为 100 *)
结构化文本（ST）	Var1:=LIMIT(0,106,100); (*Var1 结果为 100 *)

提示：若最小值大于最大值，输出数据等于最大值。

3.4. 比较指令

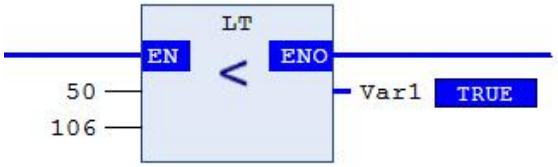
所有选择指令均可接受变量或常量输入。

3.4.1. LT——小于指令

小于指令 LT 用于比较两个输入数据的大小。当第一个输入数据小于第二个输入数据时，指令输出数据为 TRUE，否则输出数据为 FALSE。

指令对应的输入和输出数据类型	
输入的数据类型为 BOOL、BYTE、DATE、DATE_AND_TIME(DT)、DINT、DWORD、INT、LDATE、LDATE_AND_TIME(LDT)、LINT、LREAL、LTIME、LTIME_OF_DAY(LTOD)、LWORD、REAL、SINT、STRING、TIME、TIME_OF_DAY(TOD)、UDINT、UINT、USINT、WORD、WSTRING。 输出的数据类型为 BOOL。	
变量定义	

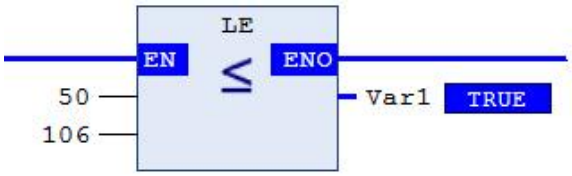
	类别	名称	地址	数据类型	初值	注释	特性
1	VAR	Var1		INT			

编程语言	程序
梯形图（LD）	 (*Var1 结果为 TRUE*)

结构化文本（ST）	Var1:=50<106； (*Var1 结果为 TRUE*)

3.4.2. LE——小于等于指令

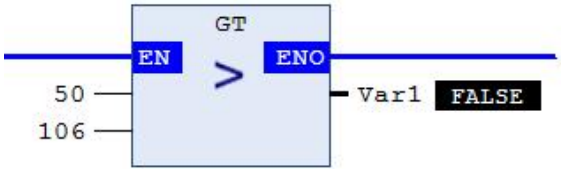
小于等于指令 LE 用于比较两个输入数据的大小。当第一个输入数据小于等于第二个输入数据时，指令输出数据为 TRUE，否则输出数据为 FALSE。

指令对应的输入和输出数据类型	
输入的数据类型为 BOOL、BYTE、DATE、DATE_AND_TIME(DT)、DINT、DWORD、INT、LDATE、LDATE_AND_TIME(LDT)、LINT、LREAL、LTIME、LTIME_OF_DAY(LTOD)、LWORD、REAL、SINT、STRING、TIME、TIME_OF_DAY(TOD)、UDINT、UINT、USINT、WORD、WSTRING。 输出的数据类型为 BOOL。	
变量定义	
类别 名称 地址 数据类型 初值 注释 特性 1 VAR Var1 INT	
编程语言	程序
梯形图（LD）	 (*Var1 结果为 TRUE*)
结构化文本（ST）	Var1:=50<=106； (*Var1 结果为 TRUE*)

3.4.3. GT——大于指令

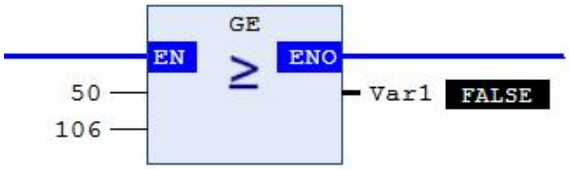
大于指令 GT 用于比较两个输入数据的大小。当第一个输入数据大于第二个输入数据时，指令输出数据为 TRUE，否则输出数据为 FALSE。

指令对应的输入和输出数据类型	
输入的数据类型为 BOOL、BYTE、DATE、DATE_AND_TIME(DT)、DINT、DWORD、INT、LDATE、LDATE_AND_TIME(LDT)、LINT、LREAL、LTIME、LTIME_OF_DAY(LTOD)、LWORD、REAL、SINT、STRING、TIME、TIME_OF_DAY(TOD)、UDINT、UINT、USINT、WORD、WSTRING。 输出的数据类型为 BOOL。	
变量定义	
类别 名称 地址 数据类型 初值 注释 特性 1 VAR Var1 INT	
编程语言	程序

梯形图 （LD）	 (*Var1 结果为 FALSE*)
结构化文本 （ST）	Var1:=50>106; (*Var1 结果为 FALSE*)

3.4.4. GE——大于等于指令

大于等于指令 GE 用于比较两个输入数据的大小。当第一个输入数据大于等于第二个输入数据时，指令输出数据为 TRUE，否则输出数据为 FALSE。

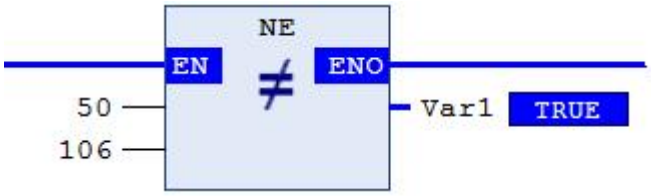
指令对应的输入和输出数据类型	
输入的数据类型为 BOOL、BYTE、DATE、DATE_AND_TIME(DT)、DINT、DWORD、INT、LDATE、LDATE_AND_TIME(LDT)、LINT、LREAL、LTIME、LTIME_OF_DAY(LTOD)、LWORD、REAL、SINT、STRING、TIME、TIME_OF_DAY(TOD)、UDINT、UINT、USINT、WORD、WSTRING。 输出的数据类型为 BOOL。	
变量定义	
	
编程语言	程序
梯形图 （LD）	 (*Var1 结果为 FALSE*)
结构化文本 （ST）	Var1:=50>=106; (*Var1 结果为 FALSE*)

3.4.5. NE——不等于指令

不等于指令 NE 用于比较两个输入数据的大小。当第一个输入数据不等于第二个输入数据时，指令输出数据为 TRUE，否则输出数据为 FALSE。

指令对应的输入和输出数据类型
输入的数据类型为 BOOL、BYTE、DATE、DATE_AND_TIME(DT)、DINT、DWORD、INT、LDATE、LDATE_AND_TIME(LDT)、LINT、LREAL、LTIME、LTIME_OF_DAY(LTOD)、LWORD、REAL、SINT、STRING、TIME、TIME_OF_DAY(TOD)、UDINT、UINT、USINT、WORD、WSTRING。 输出的数据类型为 BOOL。
变量定义

	类别	名称	地址	数据类型	初值	注释	特性
1	VAR	Var1		INT			

编程语言	程序
梯形图（LD）	 (*Var1 结果为 TRUE*)
结构化文本（ST）	Var1:=50<>106; (*Var1 结果为 TRUE*)

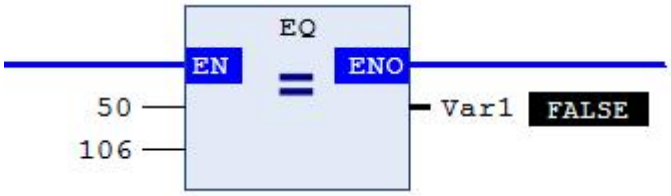
3.4.6. EQ——等于指令

等于指令 EQ 用于比较两个输入数据的大小。当第一个输入数据等于第二个输入数据时，指令输出数据为 TRUE，否则输出数据为 FALSE。

指令对应的输入和输出数据类型
输入的数据类型为 BOOL、BYTE、DATE、DATE_AND_TIME(DT)、DINT、DWORD、INT、LDATE、LDATE_AND_TIME(LDT)、LINT、LREAL、LTIME、LTIME_OF_DAY(LTOD)、LWORD、REAL、SINT、STRING、TIME、TIME_OF_DAY(TOD)、UDINT、UINT、USINT、WORD、WSTRING。 输出的数据类型为 BOOL。

变量定义

	类别	名称	地址	数据类型	初值	注释	特性
1	VAR	Var1		INT			

编程语言	程序
梯形图（LD）	 (*Var1 结果为 FALSE*)
结构化文本（ST）	Var1:=50=106; (*Var1 结果为 FALSE*)

3.5. 移位指令

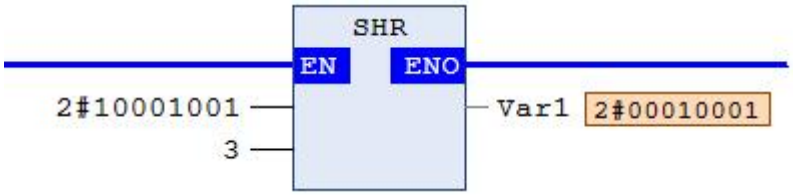
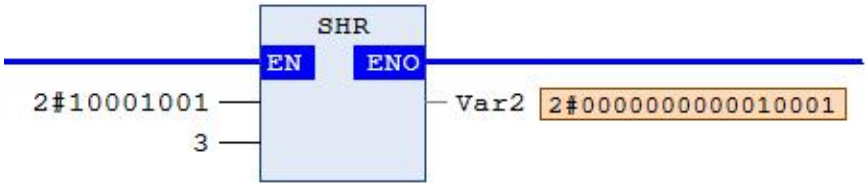
3.5.1. SHR——右移指令

右移指令是对输入数据进行按位右移，左边空缺位自动补 0，不需要处理右边移出的位。

指令对应的输入和输出数据类型
输入和输出的数据类型为 BYTE、DINT、DWORD、INT、LINT、LWORD、SINT、UDINT、UINT、USINT、WORD。

变量定义

	类别	名称	地址	数据类型	初值	注释	特性
1	VAR	Var1		BYTE			
2	VAR	Var2		WORD			

编程语言	程序
梯形图（LD）	 (*Var1 结果为 2#00010001 *)  (*Var2 结果为 2#0000000000010001 *)
结构化文本（ST）	Var1:=SHR(2#10001001,3); (*Var1 结果为 2#00010001*) Var2:=SHR(2#10001001,3); (*Var2 结果为 2#0000000000010001 *)

3.5.2. ROR——循环右移指令

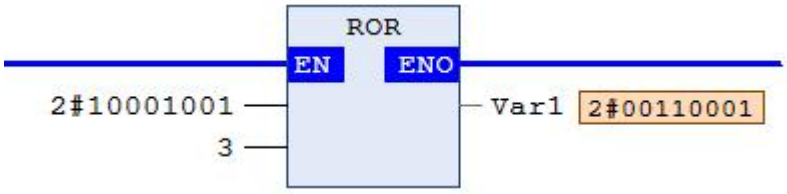
循环右移指令是对输入数据进行按位循环右移，右边移出的最低位循环送到左边的最高位处。

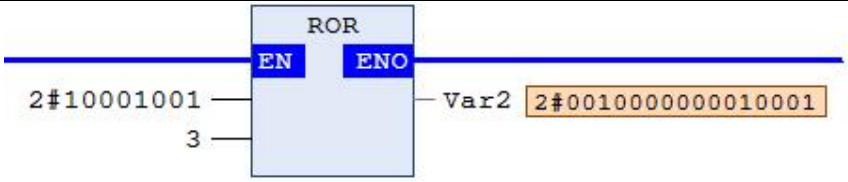
指令对应的输入和输出数据类型

输入和输出的数据类型为 BYTE、DINT、DWORD、INT、LINT、LWORD、SINT、UDINT、UINT、USINT、WORD。

变量定义

	类别	名称	地址	数据类型	初值	注释	特性
1	VAR	Var1		BYTE			
2	VAR	Var2		WORD			

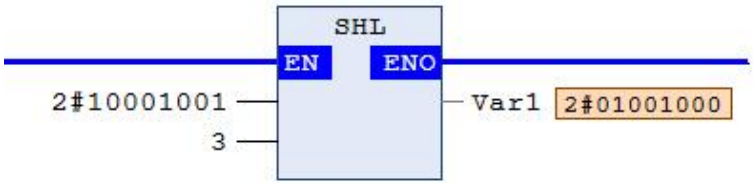
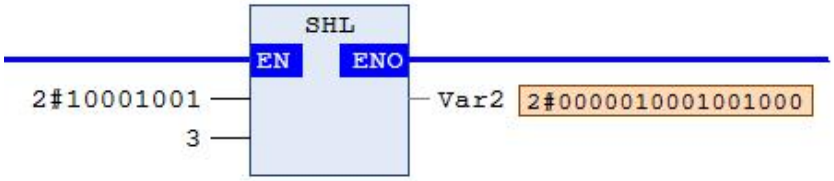
编程语言	程序
梯形图（LD）	 (*Var1 结果为 2#00110001 *)

	 (*Var2 结果为 2#0010000000010001 *)
结构化文本（ST）	Var1:=ROR(2#10001001,3); (*Var1 结果为 2#00110001 *) Var2:=ROR(2#10001001,3); (*Var2 结果为 2#0010000000010001 *)

提示：在循环右移过程中，若输出数据的数据类型的长度不一样，则输入数据相同时输出的数据结果不一样。

3.5.3. SHL——左移指令

左移指令是对输入数据进行按位左移，右边空缺位自动补 0，不需要处理左边移出的位。

指令对应的输入和输出数据类型	
输入和输出的数据类型为 BYTE、DINT、DWORD、INT、LINT、LWORD、SINT、UDINT、UINT、USINT、WORD。	
变量定义	
	
编程语言	**程序**
梯形图（LD）	 (*Var1 结果为 2#01001000 *)  (*Var2 结果为 2#0000010001001000 *)
结构化文本（ST）	Var1:=SHL(2#10001001,3); (*Var1 结果为 2#01001000 *) Var2:=SHL(2#10001001,3); (*Var2 结果为 2#0000010001001000 *)

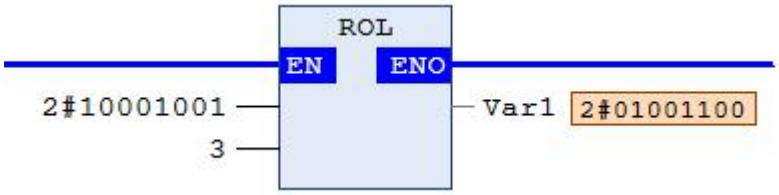
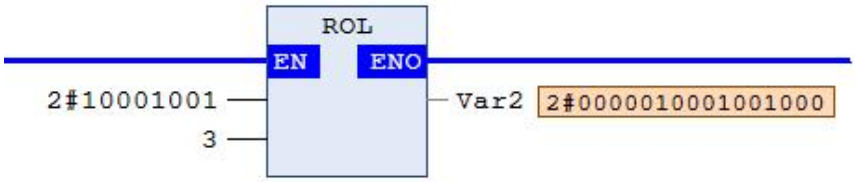
3.5.4. ROL——循环左移指令

循环左移指令是对输入数据进行按位循环左移，左边移出的最高位循环送到右边的最低位处。

指令对应的输入和输出数据类型
输入和输出的数据类型为 BYTE、DINT、DWORD、INT、LINT、LWORD、SINT、UDINT、UINT、USINT、WORD。

变量定义	

	类别	名称	地址	数据类型	初值	注释	特性
1	VAR	Var1		BYTE			
2	VAR	Var2		WORD			

编程语言	程序
梯形图（LD）	 (*Var1 结果为 2#01001100 *)  (*Var2 结果为 2#0000010001001000 *)
结构化文本（ST）	Var1:=ROL(2#10001001,3); (*Var1 结果为 2#01001100 *) Var2:=ROL(2#10001001,3); (*Var2 结果为 2#0000010001001000 *)

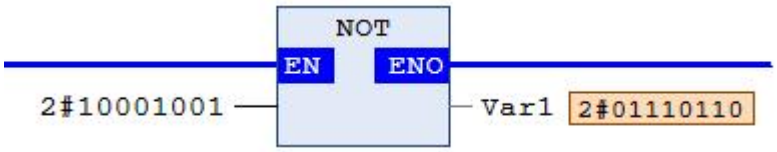
3.6. 逻辑运算指令

3.6.1. NOT——取非指令

取非指令用于对变量或常量的所有位进行取非运算。

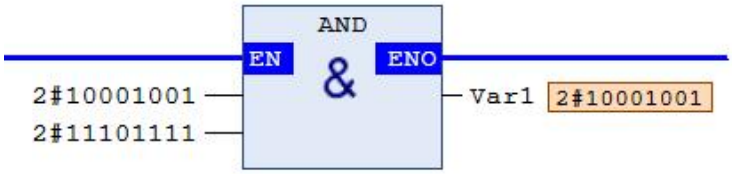
指令对应的输入和输出数据类型
输入和输出的数据类型为 BOOL、BYTE、DINT、DWORD、INT、LINT、LWORD、SINT、UDINT、UINT、USINT、WORD。
变量定义

	类别	名称	地址	数据类型	初值	注释	特性
1	VAR	Var1		BYTE			

编程语言	程序
梯形图（LD）	 (*Var1 结果为 2#01110110*)
结构化文本（ST）	Var1:=NOT 2#10001001; (*Var1 结果为 2#01110110*)

3.6.2. AND——与指令

与指令用于对两个及两个以上的变量或常量的所有位，按照由低位到高位的顺序进行与运算。

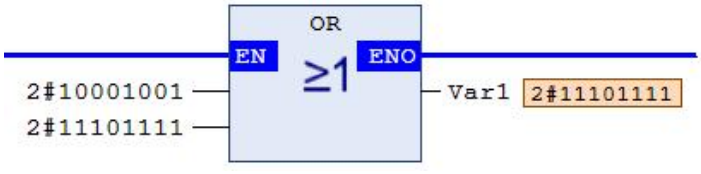
指令对应的输入和输出数据类型	
输入和输出的数据类型为 BOOL、BYTE、DINT、DWORD、INT、LINT、LWORD、SINT、UDINT、UINT、USINT、WORD。	
变量定义	
	
编程语言	程序
梯形图（LD）	 (*Var1 结果为 2#10001001*)
结构化文本（ST）	Var1:=2#10001001 AND 2#11101111; (*Var1 结果为 2#10001001*)

提示：

1. **AND 与指令默认有两个输入端填写输入数据，可以在输入端处添加输入。**
2. **输出数据的数据类型长度要大于或等于所有输入数据的数据类型长度。**

3.6.3. OR——或指令

或指令用于对两个及两个以上的变量或常量的所有位，按照由低位到高位的顺序进行或运算。

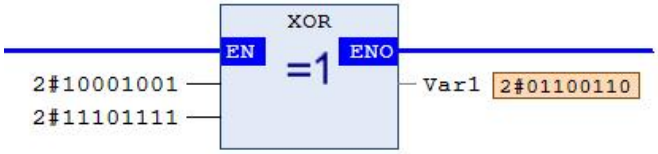
指令对应的输入和输出数据类型	
输入和输出的数据类型为 BOOL、BYTE、DINT、DWORD、INT、LINT、LWORD、SINT、UDINT、UINT、USINT、WORD。	
变量定义	
	
编程语言	程序
梯形图（LD）	 (*Var1 结果为 2#11101111*)
结构化文本（ST）	Var1:=2#10001001 OR 2#11101111; (*Var1 结果为 2#11101111*)

提示：

1. **OR 或指令默认有两个输入端填写输入数据，可以在输入端处添加输入。**
2. **输出数据的数据类型长度要大于或等于所有输入数据的数据类型长度。**

3.6.4. XOR——异或指令

异或指令用于对两个及两个以上的变量或常量的所有位，按照由低位到高位的顺序进行异或运算。

指令对应的输入和输出数据类型	
输入和输出的数据类型为 BOOL、BYTE、DINT、DWORD、INT、LINT、LWORD、SINT、UDINT、UINT、USINT、WORD。	
变量定义	
类别 名称 地址 数据类型 初值 注释 特性 1 VAR Var1 BYTE	
编程语言	程序
梯形图（LD）	 (*Var1 结果为 2#01100110*)
结构化文本（ST）	Var1:=2#10001001 XOR 2#11101111; (*Var1 结果为 2#01100110*)

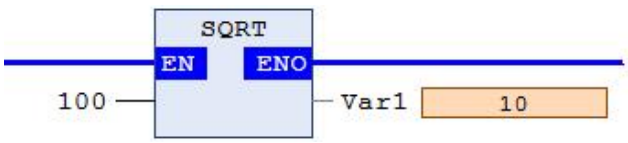
提示：

1. **XOR 异或指令默认有两个输入端填写输入数据，可以在输入端处添加输入。**
2. **输出数据的数据类型长度要大于或等于所有输入数据的数据类型长度。**

3.7. 初等数学运算指令

3.7.1. SQRT——平方根指令

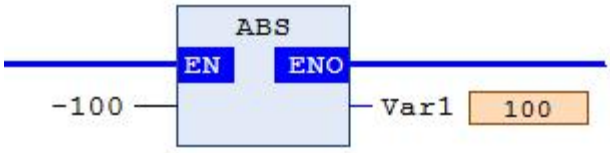
平方根指令用于求解输入数据的平方根，输出数据为运算结果。

指令对应的输入和输出数据类型	
输入的数据类型为 BYTE、DINT、DWORD、INT、LINT、LWORD、LREAL、REAL、SINT、UDINT、UINT、USINT、WORD。 输出的数据类型是 REAL、LREAL。	
变量定义	
类别 名称 地址 数据类型 初值 注释 特性 1 VAR Var1 REAL	
编程语言	程序
梯形图（LD）	 (*Var1 结果为 10*)
结构化文本（ST）	Var1:=SQRT(100); (*Var1 结果为 10*)

提示：当输入数据为负数时，输出数据的值为 0。

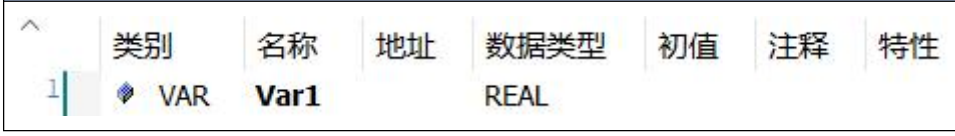
3.7.2. ABS——绝对值指令

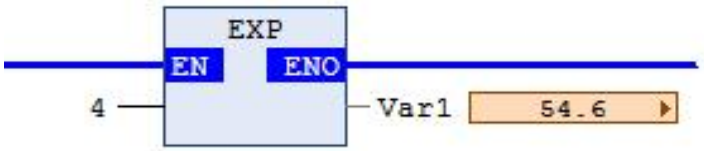
绝对值指令用于求解输入数据的绝对值，输出数据为运算结果。

输入数据类型	输出数据类型
BYTE	BYTE、DINT、DWORD、INT、REAL、SINT、UDINT、UINT、USINT、WORD
DINT	DINT、DWORD、REAL、UDINT
DWORD	DINT、DWORD、REAL、UDINT
INT	DINT、DWORD、INT、REAL、UDINT、UINT、WORD
REAL	REAL
SINT	BYTE、DINT、DWORD、INT、REAL、SINT、UDINT、UINT、USINT、WORD
UDINT	DINT、DWORD、REAL、UDINT
UINT	DINT、DWORD、INT、REAL、UDINT、UINT、WORD
USINT	BYTE、DINT、DWORD、INT、REAL、SINT、UDINT、UINT、USINT、WORD
WORD	DINT、DWORD、INT、REAL、UINT、WORD
变量定义	
	
编程语言	**程序**
梯形图（LD）	 (*Var1 结果为 100*)
结构化文本（ST）	Var1:=ABS(-100); (*Var1 结果为 100*)

3.7.3. EXP——指数指令

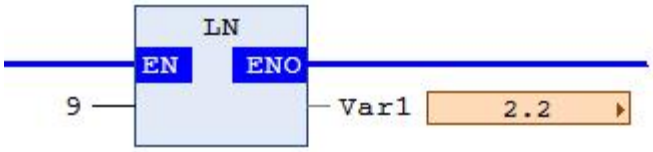
指数指令用于求解底数为 e、指数为输入数据的幂运算，即 ex，输出数据为运算结果。

指令对应的输入和输出数据类型
输入的数据类型为 BYTE、DINT、DWORD、INT、LINT、LWORD、LREAL、REAL、SINT、UDINT、UINT、USINT、WORD。 输出的数据类型为 REAL、LREAL。
变量定义


编程语言	程序
梯形图 （LD）	 (*Var1 结果为 54.6*)
结构化文本 （ST）	Var1:=EXP(4); (*Var1 结果为 54.6*)

3.7.4. LN——自然对数指令

自然对数指令用于求解输入数据的自然对数，输出数据为运算结果。

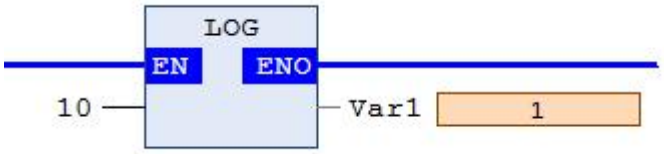
指令对应的输入和输出数据类型	
输入的数据类型为 BYTE、DINT、DWORD、INT、LINT、LWORD、LREAL、REAL、SINT、UDINT、UINT、USINT、WORD。 输出的数据类型为 REAL、LREAL。	
变量定义	
	
编程语言	程序
梯形图 （LD）	 (*Var1 结果为 2.2*)
结构化文本 （ST）	Var1:=LN(9); (*Var1 结果为 2.2*)

提示：当输入数据为负数时，输出数据的值为 0。

3.7.5. LOG——常用对数指令

常用对数指令用于求解输入数据以 10 为底的对数，输出数据为运算结果。

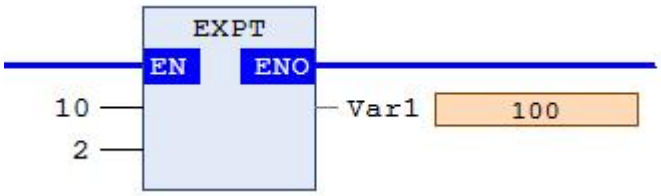
指令对应的输入和输出数据类型	
输入的数据类型为 BYTE、DINT、DWORD、INT、LINT、LWORD、LREAL、REAL、SINT、UDINT、UINT、USINT、WORD。 输出的数据类型为 REAL、LREAL。	
变量定义	
	
编程语言	程序

梯形图 （LD）	 (*Var1 结果为 1*)
结构化文本 （ST）	Var1:=LOG(10); (*Var1 结果为 1*)

提示：当输入数据为负数时，输出数据的值为 0。

3.7.6. EXPT——幂指令

幂指令用于对两个输入数据进行幂运算，第一个数据为幂运算的底数，第二个数据为幂运算的指数，输出数据为运算结果。

指令对应的输入和输出数据类型	
输入的数据类型为 BYTE、DINT、DWORD、INT、LINT、LWORD、LREAL、REAL、SINT、UDINT、UINT、USINT、WORD。 输出的数据类型为 REAL、LREAL。	
变量定义	
	
编程语言	**程序**
梯形图 （LD）	 (*Var1 结果为 100*)
结构化文本 （ST）	Var1:=EXPT(10,2); (*Var1 结果为 100*)

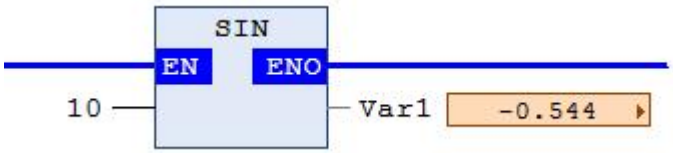
3.7.7. SIN——正弦指令

在软件中，所有的三角函数均采用弧度进行运算，包括正弦、余弦、正切、余切。若已知角度值，需换算成弧度值后再参与计算。应用上述三角函数的反函数进行运算时，其结果为弧度值。若要得到角度值，可对弧度值进行换算，弧度=角度/180*3.14。

正弦指令用于求解输入数据的正弦值，输出数据为运算结果。

指令对应的输入和输出数据类型
输入的数据类型为 BYTE、DINT、DWORD、INT、LINT、LWORD、LREAL、REAL、SINT、UDINT、UINT、USINT、WORD。 输出的数据类型为 REAL、LREAL。
变量定义

类别	名称	地址	数据类型	初值	注释	特性
1 VAR	Var1		REAL			

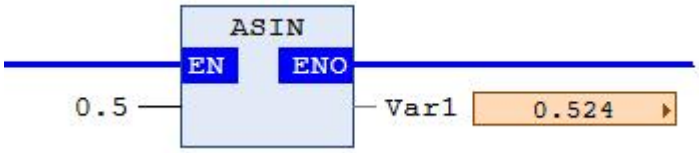
编程语言	程序
梯形图（LD）	 (*Var1 结果为-0.544*)
结构化文本（ST）	Var1:=SIN(10); (*Var1 结果为-0.544*)

3.7.8. ASIN——反正弦指令

反正弦指令用于求解输入数据的反正弦值，输出数据为运算结果。

指令对应的输入和输出数据类型
输入的数据类型为 BYTE、DINT、DWORD、INT、LINT、LWORD、LREAL、REAL、SINT、UDINT、UINT、USINT、WORD。 输出的数据类型为 REAL、LREAL。
变量定义

类别	名称	地址	数据类型	初值	注释	特性
1 VAR	Var1		REAL			

编程语言	程序
梯形图（LD）	 (*Var1 结果为 0.524*)
结构化文本（ST）	Var1:=ASIN(0.5); (*Var1 结果为 0.524*)

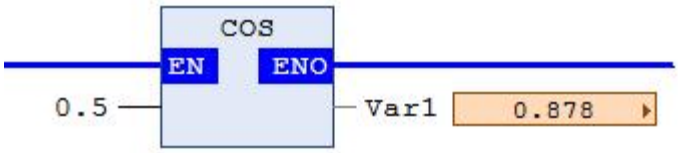
3.7.9. COS——余弦指令

余弦指令用于求解输入数据的与弦值，输出数据为运算结果。

指令对应的输入和输出数据类型
输入的数据类型为 BYTE、DINT、DWORD、INT、LINT、LWORD、LREAL、REAL、SINT、UDINT、UINT、USINT、WORD。 输出的数据类型为 REAL、LREAL。
变量定义

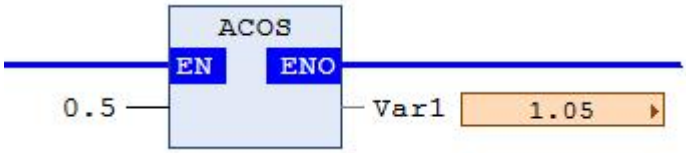
类别	名称	地址	数据类型	初值	注释	特性
1 VAR	Var1		REAL			

编程语言	程序

梯形图 （LD）	 (*Var1 结果为 0.878*)
结构化文本 （ST）	Var1:=COS(0.5); (*Var1 结果为 0.878*)

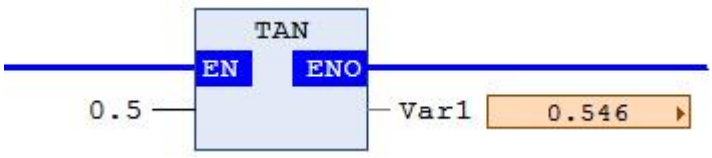
3.7.10. ACOS——反余弦指令

反余弦指令用于求解输入数据的反余弦值，输出数据为运算结果。

指令对应的输入和输出数据类型	
输入的数据类型为 BYTE、DINT、DWORD、INT、LINT、LWORD、LREAL、REAL、SINT、UDINT、UINT、USINT、WORD。 输出的数据类型为 REAL、LREAL。	
变量定义	
	
编程语言	程序
梯形图 （LD）	 (*Var1 结果为 1.05*)
结构化文本 （ST）	Var1:=ACOS(0.5); (*Var1 结果为 1.05*)

3.7.11. TAN——正切指令

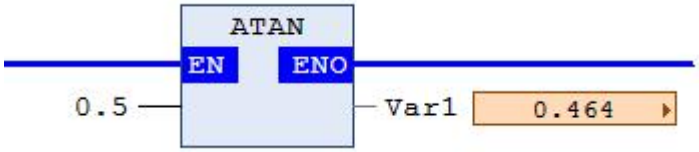
正切指令用于求解输入数据的正切值，输出数据为运算结果。

指令对应的输入和输出数据类型	
输入的数据类型为 BYTE、DINT、DWORD、INT、LINT、LWORD、LREAL、REAL、SINT、UDINT、UINT、USINT、WORD。 输出的数据类型为 REAL、LREAL。	
变量定义	
	
编程语言	程序
梯形图 （LD）	

	(*Var1 结果为 0.546*)
结构化文本（ST）	Var1:=TAN(0.5); (*Var1 结果为 0.546*)

3.7.12. ATAN——反正切指令

反正切指令用于求解输入数据的反正切值，输出数据为运算结果。

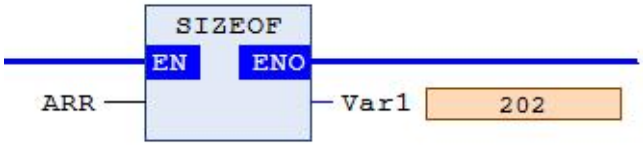
指令对应的输入和输出数据类型	
输入的数据类型为 BYTE、DINT、DWORD、INT、LINT、LWORD、LREAL、REAL、SINT、UDINT、UINT、USINT、WORD。 输出的数据类型为 REAL、LREAL。	
变量定义	
	
编程语言	程序
梯形图（LD）	 (*Var1 结果为 0.464*)
结构化文本（ST）	Var1:=ATAN(0.5); (*Var1 结果为 0.464*)

3.8. 地址运算指令

3.8.1. SIZEOF——数据类型大小指令

通过调用数据类型大小指令可获取各种数据类型的变量所占用的存储空间字节数。SIZEOF 数据类型大小指令的输入数据为需要计算所占用存储空间的变量，输出数据为计算得出的存储空间字节数，是一个无符号的值。

指令对应的输入和输出数据类型	
输入的数据类型为 BOOL、BYTE、DATE、DATE_AND_TIME(DT)、DINT、DWORD、INT、LDATE、LDATE_AND_TIME(LDT)、LINT、LREAL、LTIME、LTIME_OF_DAY(LTOD)、LWORD、REAL、SINT、STRING、TIME、TIME_OF_DAY(TOD)、UDINT、UINT、USINT、WORD、WSTRING、ARRAY。 输出的数据类型为 BYTE、DINT、DWORD、INT、LINT、LREAL、LWORD、REAL、SINT、UDINT、UINT、WORD。	
变量定义	
	
编程语言	程序

梯形图 （LD）	 (*Var1 结果为 202*)
结构化文本 （ST）	Var1:=SIZEOF(ARR); (*Var1 结果为 202 *)

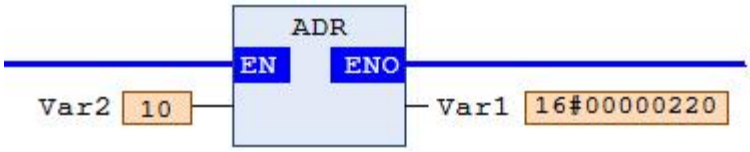
提示：

1. **DATE、DATE_AND_TIME(DT)、TIME、TIME_OF_DAY(TOD)等数据类型的输入数据所占用的地址空间均为 4 个字节，即双字。**
2. **LDATE、LDATE_AND_TIME(LDT)、LTIME、LTIME_OF_DAY(LTOD)等数据类型的输入数据所占用的地址空间均为 8 个字节，即四字。**
3. **STRING 数据类型的输入数据所占用的地址空间为 81 个字节。**
4. **WSTRING 数据类型的输入数据所占用的地址空间为 162 个字节。**

3.8.2. ADR——取地址指令

在 PLC 应用中，经常会遇到需要获取变量绝对地址的情况。取地址指令可获取变量的绝对地址，获取的地址可当作指针使用，既可参加指针运算，也可作为输入参数传输给函数。ADR 取地址指令的输入数据为需要获取绝对地址的变量，输出数据为获取到的变量绝对地址。

变量定义							
	类别	名称	地址	数据类型	初值	注释	特性
1	VAR	**Var1**		POINTER TO BYTE			
2	VAR	**Var2**		BYTE			

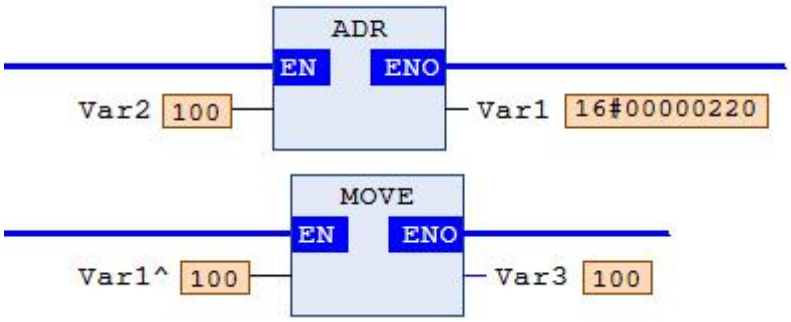
编程语言	程序
梯形图 （LD）	 (*Var1 结果为 16#00000220*)
结构化文本 （ST）	Var1:=ADR(Var2)； (*Var1 结果为 16#00000220*)

3.8.3. ^——取地址内容指令

取地址指令对应的指令为取地址内容指令。通过该指令取出绝对地址中的内容，其语法格式为：在指针变量后增加“^”符号。

变量定义							
	类别	名称	地址	数据类型	初值	注释	特性
1	VAR	**Var1**		POINTER TO BYTE			
2	VAR	**Var2**		BYTE			
3	VAR	**Var3**		BYTE			

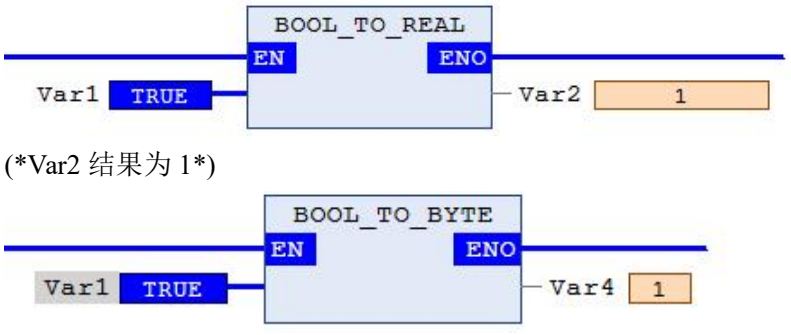
编程语言	程序

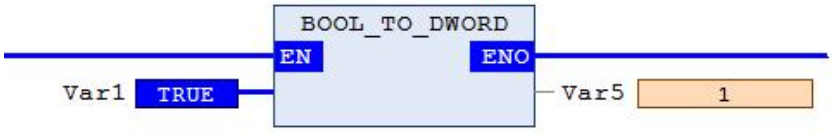
梯形图 （LD）	 (*Var3 结果为 100*)
结构化文本 （ST）	Var1:= ADR(Var2); Var3:= Var1^; (*Var3 结果为 100*)

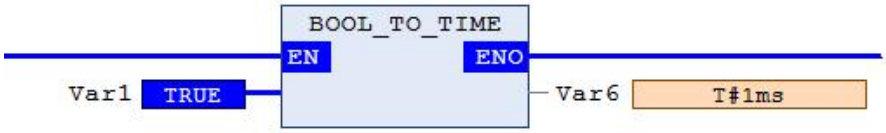
3.9. 转换指令

3.9.1. BOOL_TO_<TYPE>——布尔类型转换指令

布尔类型转换指令用于把布尔数据类型的输入数据转换为其它数据类型输出。

指令对应的输入和输出数据类型	
输入的数据类型为 BOOL。 输出的数据类型为 BYTE、DATE、DATE_AND_TIME(DT)、DINT、DWORD、INT、LDATE、LDATE_AND_TIME(LDT)、LINT、LREAL、LTIME、LTIME_OF_DAY(LTOD)、LWORD、REAL、SINT、STRING、TIME、TIME_OF_DAY(TOD)、UDINT、UINT、USINT、WORD、WSTRING。	
变量定义	
	
编程语言	**程序**
梯形图 （LD）	 (*Var2 结果为 1*) (*Var4 结果为 1*)

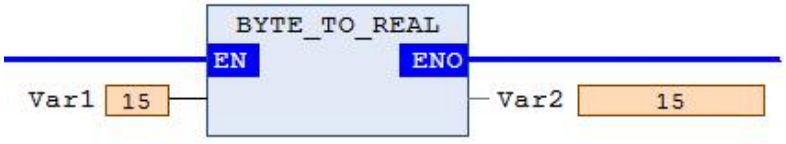
<table>
<tr><td></td><td> 

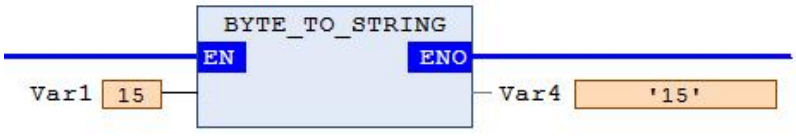
(*Var5 结果为 1*)
 

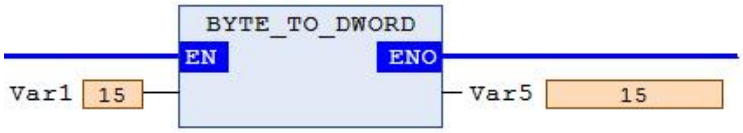
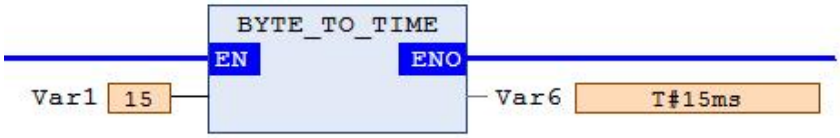
(*Var6 结果为 T#1ms*)</td></tr>
<tr><td>结构化文本
（ST）</td><td>Var2:=BOOL_TO_REAL(Var1);
(*Var2 结果为 1*)
Var4:=BOOL_TO_BYTE(Var1);
(*Var4 结果为 1*)
Var5:=BOOL_TO_DWORD(Var1);
(*Var5 结果为 1*)
Var6:=BOOL_TO_TIME(Var1);
(*Var6 结果为 T#1ms*)</td></tr>
</table>

3.9.2. BYTE_TO_<TYPE>——字节型转换指令

字节型转换指令用于把字节型的输入数据转换为其它数据类型输出。

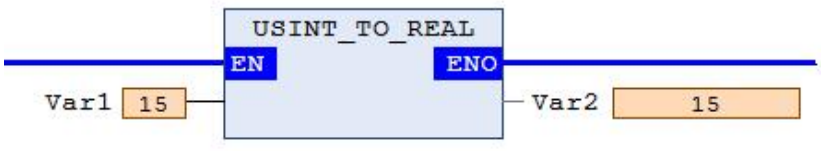
<table>
<tr><th colspan="2">指令对应的输入和输出数据类型</th></tr>
<tr><td colspan="2">输入的数据类型为 BYTE。
输出的数据类型为 BOOL、DATE、DATE_AND_TIME(DT)、DINT、DWORD、INT、LDATE、LDATE_AND_TIME(LDT)、LINT、LREAL、LTIME、LTIME_OF_DAY(LTOD)、LWORD、REAL、SINT、STRING、TIME、TIME_OF_DAY(TOD)、UDINT、UINT、USINT、WORD、WSTRING。</td></tr>
<tr><th colspan="2">变量定义</th></tr>
<tr><td colspan="2">
<table>
<tr><th></th><th>类别</th><th>名称</th><th>地址</th><th>数据类型</th><th>初值</th><th>注释</th><th>特性</th></tr>
<tr><td>1</td><td>VAR</td><td>Var1</td><td></td><td>BYTE</td><td></td><td></td><td></td></tr>
<tr><td>2</td><td>VAR</td><td>Var2</td><td></td><td>REAL</td><td></td><td></td><td></td></tr>
<tr><td>3</td><td>VAR</td><td>Var3</td><td></td><td>DATE</td><td></td><td></td><td></td></tr>
<tr><td>4</td><td>VAR</td><td>Var4</td><td></td><td>STRING</td><td></td><td></td><td></td></tr>
<tr><td>5</td><td>VAR</td><td>Var5</td><td></td><td>DWORD</td><td></td><td></td><td></td></tr>
<tr><td>6</td><td>VAR</td><td>Var6</td><td></td><td>TIME</td><td></td><td></td><td></td></tr>
</table>
</td></tr>
<tr><th>编程语言</th><th>程序</th></tr>
<tr><td>梯形图
（LD）</td><td> 

(*Var2 结果为 15*)
 
</td></tr>
</table>

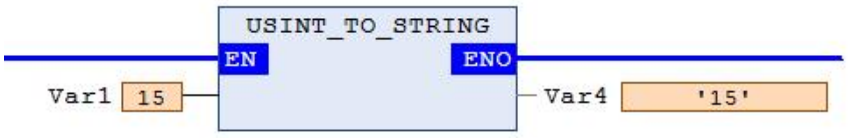
	(*Var4 结果为'15'*)  (*Var5 结果为 15*)  (*Var6 结果为 T#15ms*)
结构化文本（ST）	Var2:=BYTE_TO_REAL(Var1); (*Var2 结果为 15*) Var4:=BYTE _TO_STRING(Var1); (*Var4 结果为'15'*) Var5:=BYTE _TO_DWORD(Var1); (*Var5 结果为 15*) Var6:=BYTE _TO_TIME(Var1); (*Var6 结果为 T#15ms*)

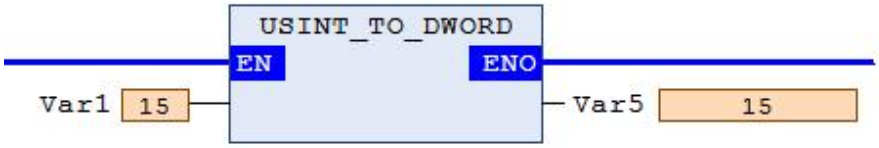
3.9.3. USINT_TO_<TYPE>——无符号短整型转换指令

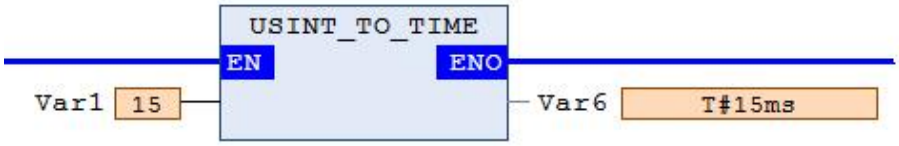
无符号短整型转换指令用于把无符号短整型的输入数据转换为其它数据类型输出。

指令对应的输入和输出数据类型	
输入的数据类型为 USINT。 输出的数据类型为 BOOL、BYTE、DATE、DATE_AND_TIME(DT)、DINT、DWORD、INT、LDATE、LDATE_AND_TIME(LDT)、LINT、LREAL、LTIME、LTIME_OF_DAY(LTOD)、LWORD、REAL、SINT、STRING、TIME、TIME_OF_DAY(TOD)、UDINT、UINT、WORD、WSTRING。	
变量定义	
	
编程语言	程序
梯形图（LD）	 (*Var2 结果为 15*)

	类别	名称	地址	数据类型	初值	注释	特性
1	VAR	Var1		USINT			
2	VAR	Var2		REAL			
3	VAR	Var3		DATE			
4	VAR	Var4		STRING			
5	VAR	Var5		DWORD			
6	VAR	Var6		TIME			

<table>
<tr><td></td><td>


(*Var4 结果为'15'*)


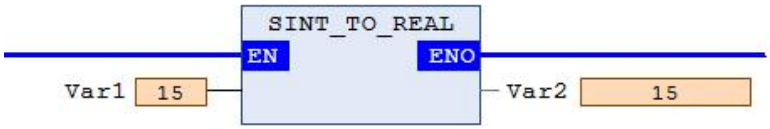
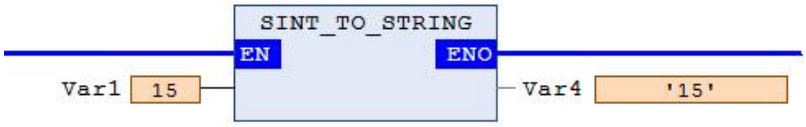
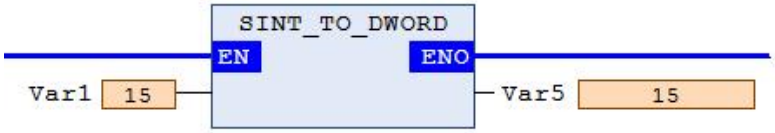
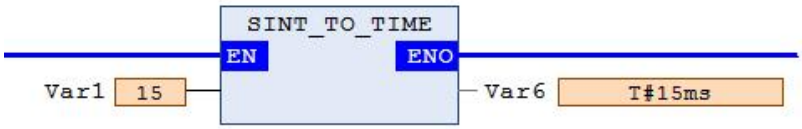
(*Var5 结果为 15*)


(*Var6 结果为 T#15ms*)</td></tr>
<tr><td>结构化文本
（ST）</td><td>Var2:=USINT_TO_REAL(Var1);
(*Var2 结果为 15*)
Var4:=USINT _TO_STRING(Var1);
(*Var4 结果为'15'*)
Var5:=USINT _TO_DWORD(Var1);
(*Var5 结果为 15*)
Var6:=USINT _TO_TIME(Var1);
(*Var6 结果为 T#15ms*)</td></tr>
</table>

3.9.4. SINT_TO_<TYPE>——短整型转换指令

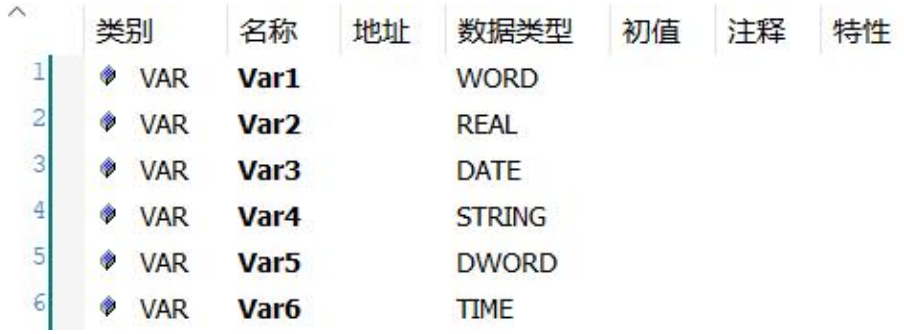
短整型转换指令用于把短整型的输入数据转换为其它数据类型输出。

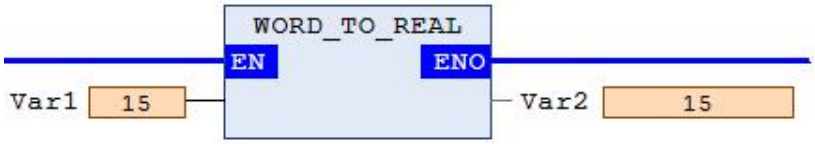
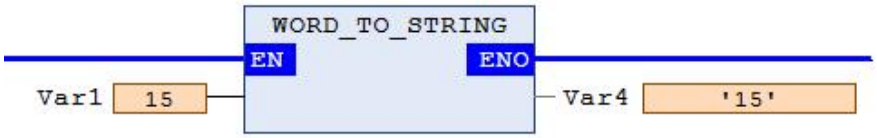
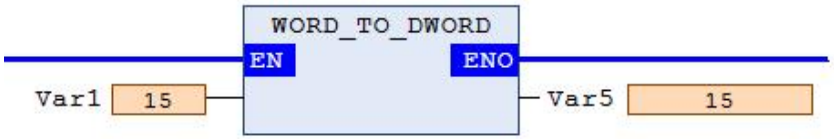
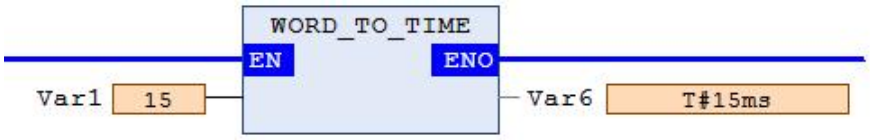
<table>
<tr><th colspan="2">指令对应的输入和输出数据类型</th></tr>
<tr><td colspan="2">输入的数据类型为 SINT。
输出的数据类型为 BOOL、BYTE、DATE、DATE_AND_TIME(DT)、DINT、DWORD、INT、LDATE、LDATE_AND_TIME(LDT)、LINT、LREAL、LTIME、LTIME_OF_DAY(LTOD)、LWORD、REAL、STRING、TIME、TIME_OF_DAY(TOD)、UDINT、UINT、USINT、WORD、WSTRING。</td></tr>
<tr><th colspan="2">变量定义</th></tr>
<tr><td colspan="2">
<table>
<tr><th></th><th>类别</th><th>名称</th><th>地址</th><th>数据类型</th><th>初值</th><th>注释</th><th>特性</th></tr>
<tr><td>1</td><td>VAR</td><td>Var1</td><td></td><td>SINT</td><td></td><td></td><td></td></tr>
<tr><td>2</td><td>VAR</td><td>Var2</td><td></td><td>REAL</td><td></td><td></td><td></td></tr>
<tr><td>3</td><td>VAR</td><td>Var3</td><td></td><td>DATE</td><td></td><td></td><td></td></tr>
<tr><td>4</td><td>VAR</td><td>Var4</td><td></td><td>STRING</td><td></td><td></td><td></td></tr>
<tr><td>5</td><td>VAR</td><td>Var5</td><td></td><td>DWORD</td><td></td><td></td><td></td></tr>
<tr><td>6</td><td>VAR</td><td>Var6</td><td></td><td>TIME</td><td></td><td></td><td></td></tr>
</table>
</td></tr>
<tr><th>编程语言</th><th>程序</th></tr>
</table>

梯形图 （LD）	 (*Var2 结果为 15*)  (*Var4 结果为'15'*)  (*Var5 结果为 15*)  (*Var6 结果为 T#15ms*)
结构化文本 （ST）	Var2:=SINT_TO_REAL(Var1); (*Var2 结果为 15*) Var4:=SINT _TO_STRING(Var1); (*Var4 结果为'15'*) Var5:=SINT _TO_DWORD(Var1); (*Var5 结果为 15*) Var6:=SINT _TO_TIME(Var1); (*Var6 结果为 T#15ms*)

3.9.5. WORD_TO_<TYPE>——字类型转换指令

字类型转换指令用于把字类型的输入数据转换为其它数据类型输出。

指令对应的输入和输出数据类型	
输入的数据类型为 WORD。 输出的数据类型为 BOOL、BYTE、DATE、DATE_AND_TIME(DT)、DINT、DWORD、INT、REAL、SINT、STRING、TIME、TIME_OF_DAY(TOD)、UDINT、UINT、USINT、WSTRING。	
变量定义	
	
编程语言	程序

梯形图 （LD）	 (*Var2 结果为 15*)  (*Var4 结果为'15'*)  (*Var5 结果为 15*)  (*Var6 结果为 T#15ms*)
结构化文本 （ST）	Var2:=WORD_TO_REAL(Var1); (*Var2 结果为 15*) Var4:=WORD _TO_STRING(Var1); (*Var4 结果为'15'*) Var5:=WORD _TO_DWORD(Var1); (*Var5 结果为 15*) Var6:=WORD _TO_TIME(Var1); (*Var6 结果为 T#15ms*)

3.9.6. DWORD_TO_<TYPE>——双字类型转换指令

双字类型转换指令用于把双字类型的输入数据转换为其它数据类型输出。

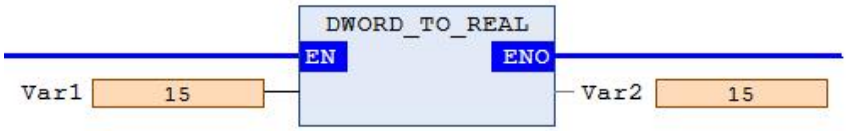
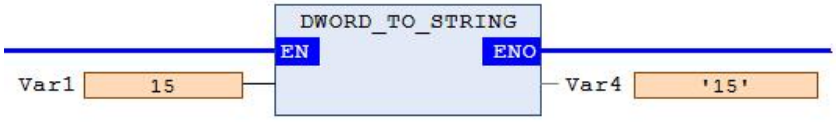
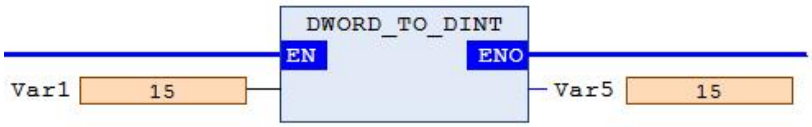
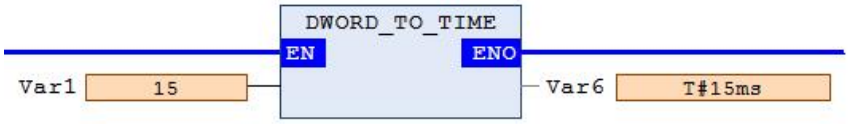
指令对应的输入和输出数据类型

输入的数据类型为 DWORD。

输出的数据类型为 BOOL、BYTE、DATE、DATE_AND_TIME(DT)、DINT、INT、LDATE、LDATE_AND_TIME(LDT)、LINT、LREAL、LTIME、LTIME_OF_DAY(LTOD)、LWORD、REAL、SINT、STRING、TIME、TIME_OF_DAY(TOD)、UDINT、UINT、USINT、WORD、WSTRING。

变量定义

	类别	名称	地址	数据类型	初值	注释	特性
1	VAR	**Var1**		DWORD			
2	VAR	**Var2**		REAL			
3	VAR	**Var3**		DATE			
4	VAR	**Var4**		STRING			
5	VAR	**Var5**		DINT			
6	VAR	**Var6**		TIME			

编程语言	程序
梯形图（LD）	 (*Var2 结果为 15*)  (*Var4 结果为'15'*)  (*Var5 结果为 15*)  (*Var6 结果为 T#15ms*)
结构化文本（ST）	Var2:=DWORD_TO_REAL(Var1); (*Var2 结果为 15*) Var4:=DWORD _TO_STRING(Var1); (*Var4 结果为'15'*) Var5:=DWORD _TO_DINT(Var1); (*Var5 结果为 15*) Var6:=DWORD _TO_TIME(Var1); (*Var6 结果为 T#15ms*)

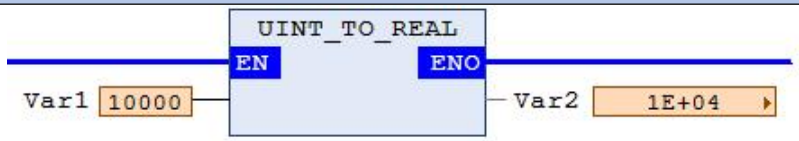
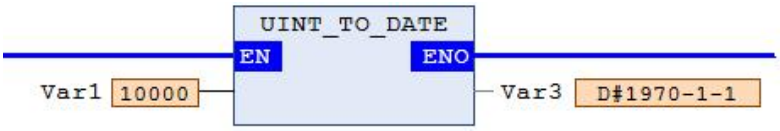
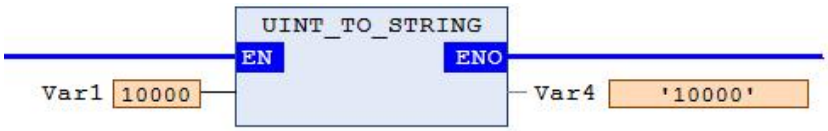
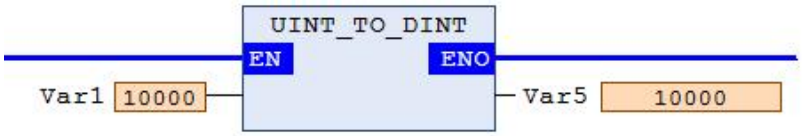
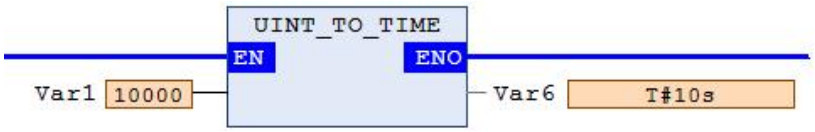
3.9.7. UINT_TO_<TYPE>——无符号整数类型转换指令

无符号整数类型转换指令用于把无符号整数类型的输入数据转换为其它数据类型输出。

指令对应的输入和输出数据类型
输入的数据类型为 UINT。 输出的数据类型为 BOOL、BYTE、DATE、DATE_AND_TIME(DT)、DINT、DWORD、INT、LDATE、LDATE_AND_TIME(LDT)、LINT、LREAL、LTIME、LTIME_OF_DAY(LTOD)、LWORD、REAL、SINT、STRING、TIME、TIME_OF_DAY(TOD)、UDINT、USINT、WORD、WSTRING。

变量定义

	类别	名称	地址	数据类型	初值	注释	特性
1	VAR	Var1		UINT			
2	VAR	Var2		REAL			
3	VAR	Var3		DATE			
4	VAR	Var4		STRING			
5	VAR	Var5		DINT			
6	VAR	Var6		TIME			

编程语言	程序
梯形图（LD）	 (*Var2 结果为 10000*)  (*Var3 结果为 D#1970-1-1*)  (*Var4 结果为'10000'*)  (*Var5 结果为 10000*)  (*Var6 结果为 T#10s*)
结构化文本（ST）	Var2:=UINT_TO_REAL(Var1); (*Var2 结果为 10000*) Var3:=UINT _TO_DATE(Var1); (*Var3 结果为 D#1970-1-1*) Var4:=UINT _TO_STRING(Var1); (*Var4 结果为'10000'*) Var5:=UINT _TO_DINT(Var1); (*Var5 结果为 10000*) Var6:=UINT _TO_TIME(Var1); (*Var6 结果为 T#10s*)

3.9.8. INT_TO_<TYPE>——整数类型转换指令

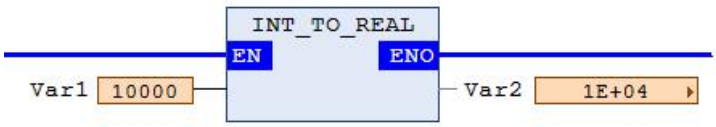
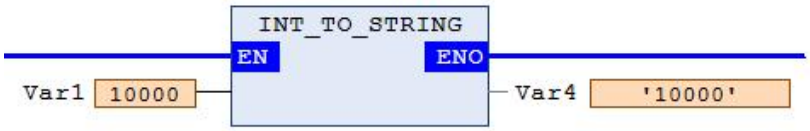
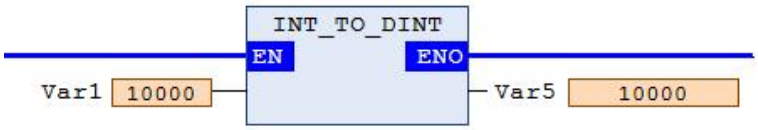
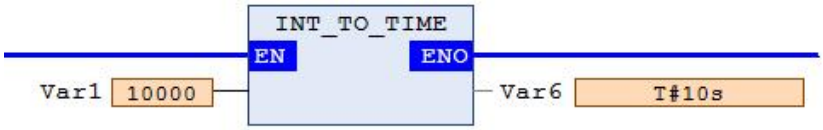
整数类型转换指令用于把整数类型的输入数据转换为其它数据类型输出。

指令对应的输入和输出数据类型
输入的数据类型为 INT。 输出的数据类型为 BOOL、BYTE、DATE、DATE_AND_TIME(DT)、DINT、DWORD、LDATE、LDATE_AND_TIME(LDT)、LINT、LREAL、LTIME、LTIME_OF_DAY(LTOD)、LWORD、

REAL、SINT、STRING、TIME、TIME_OF_DAY(TOD)、UDINT、UINT、USINT、WORD、WSTRING。

变量定义

	类别	名称	地址	数据类型	初值	注释	特性
1	VAR	**Var1**		INT			
2	VAR	**Var2**		REAL			
3	VAR	**Var3**		DATE			
4	VAR	**Var4**		STRING			
5	VAR	**Var5**		DINT			
6	VAR	**Var6**		TIME			

编程语言	程序
梯形图（LD）	 (*Var2 结果为 10000*)  (*Var4 结果为'10000'*)  (*Var5 结果为 10000*)  (*Var6 结果为 T#10s*)
结构化文本（ST）	Var2:=INT_TO_REAL(Var1); (*Var2 结果为 10000*) Var4:=INT _TO_STRING(Var1); (*Var4 结果为'10000'*) Var5:=INT _TO_DINT(Var1); (*Var5 结果为 10000*) Var6:=INT _TO_TIME(Var1); (*Var6 结果为 T#10s*)

3.9.9. UDINT_TO_<TYPE>——无符号双整数类型转换指令

无符号双整数类型转换指令用于把无符号双整数类型的输入数据转换为其它数据类型输出。

指令对应的输入和输出数据类型

输入的数据类型为 UDINT。

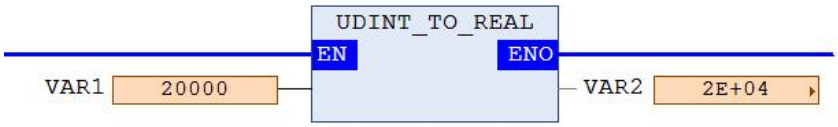
输出的数据类型为 BOOL、BYTE、DATE、DATE_AND_TIME(DT)、DINT、DWORD、INT、LDATE、LDATE_AND_TIME(LDT)、LINT、LREAL、LTIME、LTIME_OF_DAY(LTOD)、LWORD、REAL、SINT、STRING、TIME、TIME_OF_DAY(TOD)、UINT、USINT、WORD、WSTRING。

变量定义

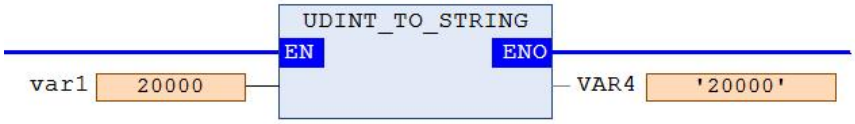
	类别	名称	地址	数据类型	初值	注释	特性
1	VAR	**Var1**		UDINT			
2	VAR	**Var2**		REAL			
3	VAR	**Var3**		DATE			
4	VAR	**Var4**		STRING			
5	VAR	**Var5**		DINT			
6	VAR	**Var6**		TIME			

编程语言 | **程序**

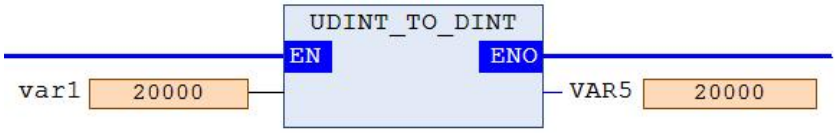
梯形图（LD）



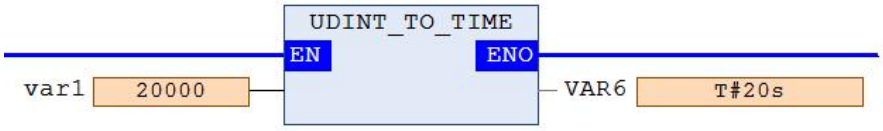
(*Var2 结果为 2E+04*)



(*Var4 结果为 '20000'*)



(*Var5 结果为 20000*)



(*Var6 结果为 T#20s*)

结构化文本（ST）

```
Var2:=UDINT_TO_REAL(Var1);
(*Var2 结果为 20000*)
Var4:=UDINT _TO_STRING(Var1);
(*Var4 结果为 '20000'*)
Var5:=UDINT _TO_DINT(Var1);
(*Var5 结果为 20000*)
Var6:=UDINT _TO_TIME(Var1);
(*Var6 结果为 T#20s*)
```

3.9.10. DINT_TO_<TYPE>——双整数类型转换指令

双整数类型转换指令用于把双整数类型的输入数据转换为其它数据类型输出。

指令对应的输入和输出数据类型

输入的数据类型为 DINT。

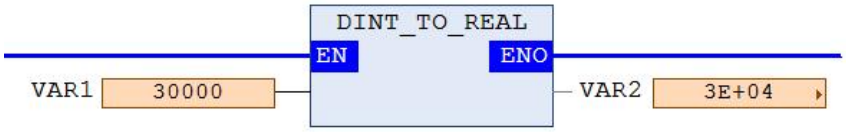
输出的数据类型为 BOOL、BYTE、DATE、DATE_AND_TIME(DT)、DWORD、INT、LDATE、LDATE_AND_TIME(LDT)、LINT、LREAL、LTIME、LTIME_OF_DAY(LTOD)、LWORD、REAL、SINT、STRING、TIME、TIME_OF_DAY(TOD)、UDINT、UINT、USINT、WORD、WSTRING。

变量定义

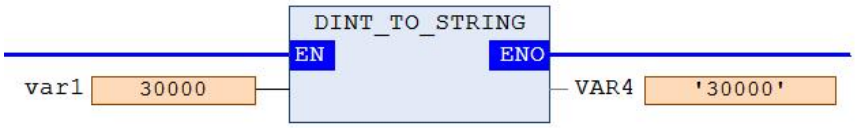
	类别	名称	地址	数据类型	初值	注释	特性
1	VAR	Var1		UDINT			
2	VAR	Var2		REAL			
3	VAR	Var3		DATE			
4	VAR	Var4		STRING			
5	VAR	Var5		WORD			
6	VAR	Var6		TIME			

编程语言 | **程序**

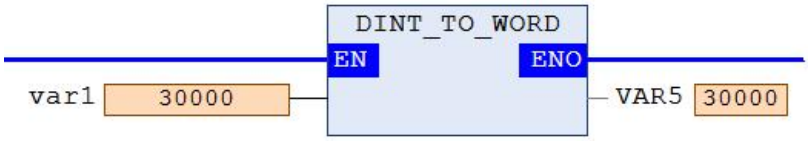
梯形图（LD）



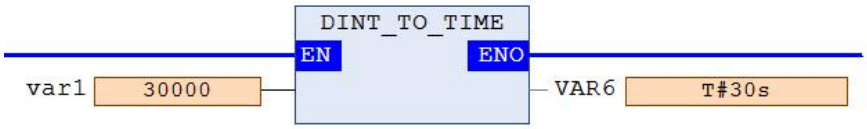
(*Var2 结果为 3E+04*)



(*Var4 结果为'30000'*)



(*Var5 结果为 30000*)



(*Var6 结果为 T#30s*)

结构化文本（ST）

```
Var2:=DINT_TO_REAL(Var1);
(*Var2 结果为 30000*)
Var4:=DINT _TO_STRING(Var1);
(*Var4 结果为'30000'*)
Var5:=DINT _TO_WORD(Var1);
(*Var5 结果为 30000*)
Var6:=DINT _TO_TIME(Var1);
(*Var6 结果为 T#30s*)
```

3.9.11. DATE_TO_<TYPE>——日期类型转换指令

日期类型的数据起始时间为 1970 年 1 月 1 日。日期类型转换指令用于把日期类型的输入数据转换为其它数据类型输出。

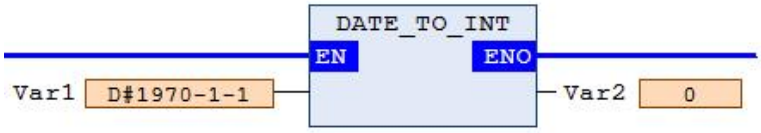
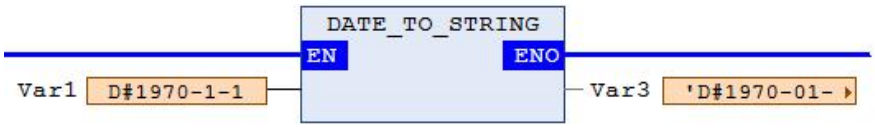
指令对应的输入和输出数据类型

输入的数据类型为 DATE。

输出的数据类型为 BOOL、BYTE、DATE_AND_TIME(DT)、DINT、DWORD、INT、LDATE、LDATE_AND_TIME(LDT)、LINT、LREAL、LTIME、LTIME_OF_DAY(LTOD)、LWORD、REAL、SINT、STRING、TIME、TIME_OF_DAY(TOD)、UDINT、UINT、USINT、WORD、WSTRING。

变量定义

	类别	名称	地址	数据类型	初值	注释	特性
1	VAR	Var1		DATE			
2	VAR	Var2		INT			
3	VAR	Var3		STRING			

编程语言	程序
梯形图（LD）	 (*Var2 结果为 0*)  (*Var3 结果为'D#1970-01-01'*)
结构化文本（ST）	Var2:=DATE_TO_INT(Var1); (*Var2 结果为 0*) Var3:=DATE _TO_STRING(Var1); (*Var3 结果为'D#1970-01-01'*)

提示：当转换指令为 DATE_TO_BOOL 时，输入为 D#1970-01-01 时输出为 FALSE，否则输出为 TRUE。

3.9.12. TIME_TO_<TYPE>——时间类型转换指令

时间类型转换指令用于把时间类型的输入数据转换为其它数据类型输出。

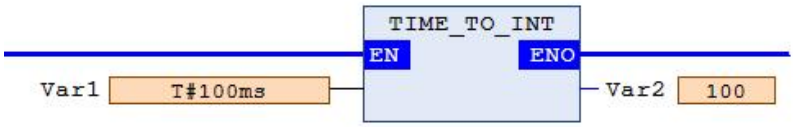
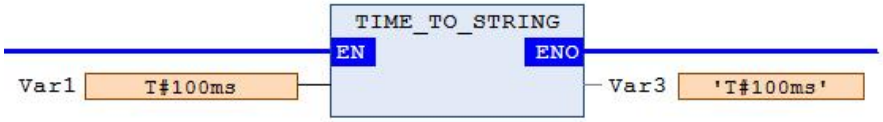
指令对应的输入和输出数据类型

输入的数据类型为 TIME。

输出的数据类型为 BOOL、BYTE、DATE、DATE_AND_TIME(DT)、DINT、DWORD、INT、LDATE、LDATE_AND_TIME(LDT)、LINT、LREAL、LTIME、LTIME_OF_DAY(LTOD)、LWORD、REAL、SINT、STRING、TIME_OF_DAY(TOD)、UDINT、UINT、USINT、WORD、WSTRING。

变量定义

	类别	名称	地址	数据类型	初值	注释	特性
1	VAR	Var1		TIME			
2	VAR	Var2		INT			
3	VAR	Var3		STRING			

编程语言	程序
梯形图（LD）	 (*Var2 结果为 100*)  (*Var3 结果为'T#100ms'*)
结构化文本（ST）	Var2:=TIME_TO_INT(Var1); (*Var2 结果为 100*) Var3:=TIME _TO_STRING(Var1); (*Var3 结果为'T#100ms'*)

提示：当转换指令为 TIME_TO_BOOL 时，输入为 T#0ms 时输出为 FALSE，否则输出为 TRUE。

3.9.13. DT_TO_<TYPE>——日期时间类型转换指令

日期时间类型转换指令用于把日期时间类型的输入数据转换为其它数据类型输出。

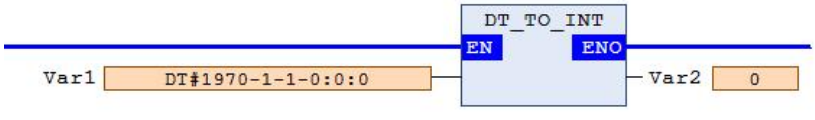
指令对应的输入和输出数据类型

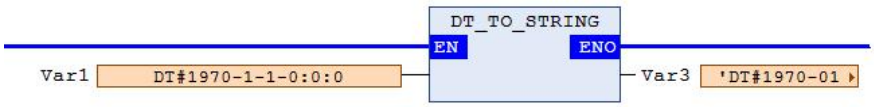
输入的数据类型为 DT。

输出的数据类型为 BOOL、BYTE、DATE、DINT、DWORD、INT、LDATE、LDATE_AND_TIME(LDT)、LINT、LREAL、LTIME、LTIME_OF_DAY(LTOD)、LWORD、REAL、SINT、STRING、TIME、TIME_OF_DAY(TOD)、UDINT、UINT、USINT、WORD、WSTRING。

变量定义

	类别	名称	地址	数据类型	初值	注释	特性
1	VAR	Var1		DT			
2	VAR	Var2		INT			
3	VAR	Var3		STRING			

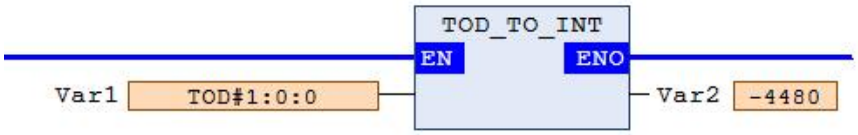
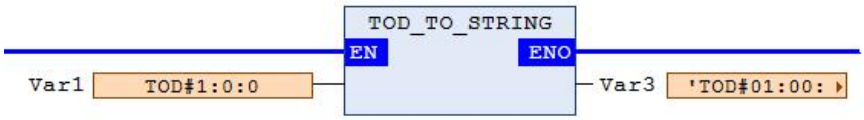
编程语言	程序
梯形图（LD）	 (*Var2 结果为 0*)

	 (*Var3 结果为'DT#1970-1-1-0:0:0'*)
结构化文本（ST）	Var2:=DT_TO_INT(Var1); (*Var2 结果为 0*) Var3:=DT _TO_STRING(Var1); (*Var3 结果为'DT#1970-1-1-0:0:0'*)

提示：当转换指令为 DT_TO_BOOL 时，输入为 DT#1970-1-1-0:0:0 时输出为 FALSE，否则输出为 TRUE。

3.9.14. TOD_TO_<TYPE>——时间日期类型转换指令

时间日期类型转换指令用于把时间日期类型的输入数据转换为其它数据类型输出。

指令对应的输入和输出数据类型	
输入的数据类型为 TOD。 输出的数据类型为 BOOL、BYTE、DATE、DATE_AND_TIME(DT)、DINT、DWORD、INT、LDATE、LDATE_AND_TIME(LDT)、LINT、LREAL、LTIME、LTIME_OF_DAY(LTOD)、LWORD、REAL、SINT、STRING、TIME、UDINT、UINT、USINT、WORD、WSTRING。	
变量定义	
	
编程语言	程序
梯形图（LD）	 (*Var2 结果为-4480*)  (*Var3 结果为'TOD#01:00:00'*)
结构化文本（ST）	Var2:=TOD_TO_INT(Var1); (*Var2 结果为-4480*) Var3:=TOD _TO_STRING(Var1); (*Var3 结果为'TOD#01:00:00'*)

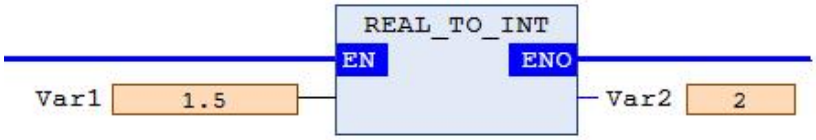
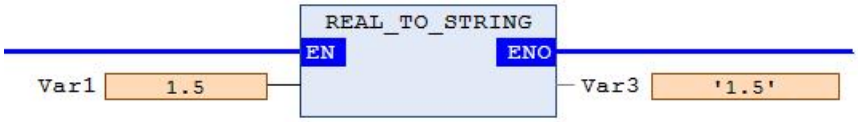
提示：当转换指令为 TOD_TO_BOOL 时，输入为 TOD#00:00:00 时输出为 FALSE，否则输出为 TRUE。

3.9.15. REAL_TO_<TYPE>——实数类型转换指令

实数类型转换指令用于把实数类型的输入数据转换为其它数据类型输出。把实数转换为其它类型数据(字符型数据类型除外)前，先要对实数进行四舍五入，变成整数后再转成其它数据类型。

指令对应的输入和输出数据类型	
输入的数据类型为 REAL。 输出的数据类型为 BOOL、BYTE、DATE、DATE_AND_TIME(DT)、DINT、DWORD、INT、LDATE、LDATE_AND_TIME(LDT)、LINT、LREAL、LTIME、LTIME_OF_DAY(LTOD)、LWORD、SINT、STRING、TIME、TIME_OF_DAY(TOD)、UDINT、UINT、USINT、WORD、WSTRING。	
变量定义	

	类别	名称	地址	数据类型	初值	注释	特性
1	VAR	Var1		REAL			
2	VAR	Var2		INT			
3	VAR	Var3		STRING			

编程语言	程序
梯形图 （LD）	 (*Var2 结果为 2*)  (*Var3 结果为'1.5'*)
结构化文本 （ST）	Var2:=REAL_TO_INT(Var1); (*Var2 结果为 2*) Var3:=REAL _TO_STRING(Var1); (*Var3 结果为'1.5'*)

提示：

1. 将数据类型从 REAL 转换成 BYTE、DINT、DWORD、INT、SINT、UDINT、UINT、USINT、WORD 时，如果 REAL 数的值超出了转换整数的范围，将收到根据目标系统产生的一个不确定的结果。

2. 当转换指令为 REAL_TO_BOOL 时，输入为 0 时输出为 FALSE，否则输出为 TRUE。

3.9.16. STRING_TO_<TYPE>——字符类型转换指令

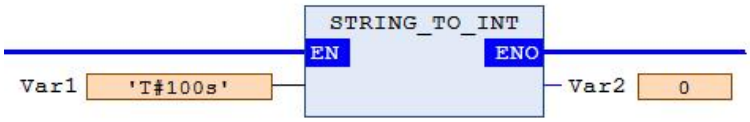
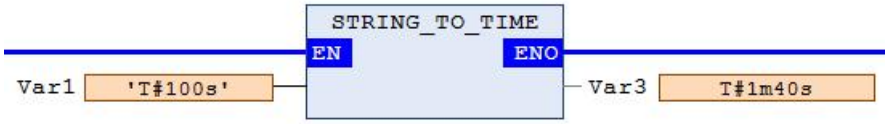
字符类型转换指令用于把字符类型的输入数据转换为其它数据类型输出。若被转换的字符串含有数值，转换结果取其数值。若被转换字符串不含数值，则转换结果为 0。

定义 STRING 操作数必须是目标类型的有效常量，不能包含无穷值、前缀、分组字符（_）和逗号。允许在一个数字后附加字符，例如：23xy，在一个数字之前添加字符是不允许的。输入数据必须是目标数据类型的有效值。

指令对应的输入和输出数据类型
输入的数据类型为 STRING。 输出的数据类型为 BOOL、BYTE、DATE、DATE_AND_TIME(DT)、DINT、DWORD、INT、LDATE、LDATE_AND_TIME(LDT)、LINT、LREAL、LTIME、LTIME_OF_DAY(LTOD)、LWORD、REAL、SINT、TIME、TIME_OF_DAY(TOD)、UDINT、UINT、USINT、WORD、WSTRING。

变量定义

	类别	名称	地址	数据类型	初值	注释	特性
1	VAR	Var1		STRING			
2	VAR	Var2		INT			
3	VAR	Var3		TIME			

编程语言	程序
梯形图（LD）	 (*Var2 结果为 0*)  (*Var3 结果为'T#1m40s'*)
结构化文本（ST）	Var2:=STRING_TO_INT(Var1); (*Var2 结果为 0*) Var3:=STRING _TO_TIME(Var1); (*Var3 结果为'T#1m40s'*)

提示：将 STRING 数据类型的字符正确的转换为 TIME 数据类型，需要把区分时间的 d、h、m、s、ms 等字符修改为小写。例如'T#100s'可以转换为'T#1m40s'，而'T#100S'则转换为'T#0s'。

3.9.17. TRUNC——截短转换指令

截断转换指令用于截取输入数据的实数整数部分，忽略实数的小数部分，转换为其它数据类型输出。

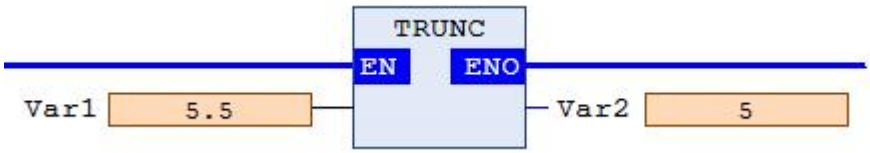
指令对应的输入和输出数据类型

输入的数据类型为 REAL、LREAL。

输出的数据类型为 DINT 、DWORD 、LINT、LWORD、UDINT。

变量定义

	类别	名称	地址	数据类型	初值	注释	特性
1	VAR	Var1		REAL			
2	VAR	Var2		DINT			

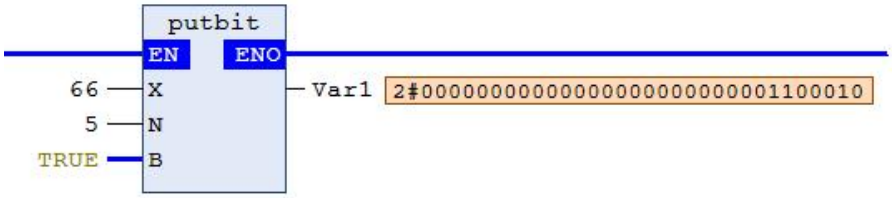
编程语言	程序
梯形图（LD）	 (*Var2 结果为 5 *)
结构化文本（ST）	Var2:=TRUNC(Var1); (*Var2 结果为 5 *)

提示：该指令的功能与四舍五入不同，若要通过四舍五入取整，请参照指令 REAL_TO_INT。

3.10. 位/字节操作指令（Util.compiled-library）

3.10.1. PUTBIT——位赋值指令

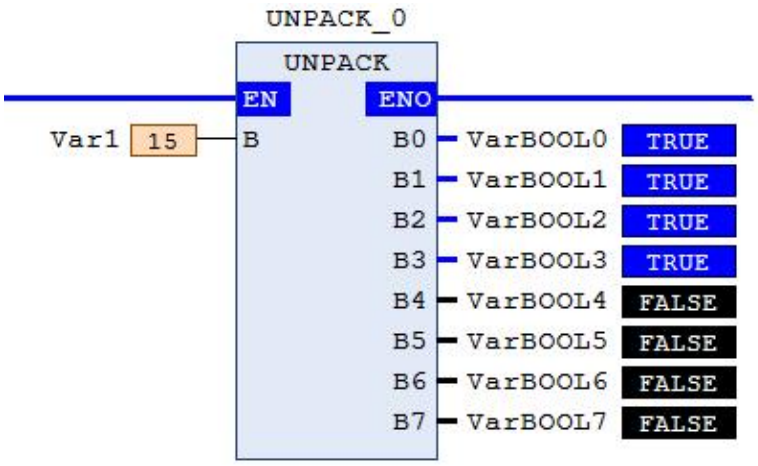
若要对输入数据的某一位进行赋值，可使用位赋值指令 PUTBIT。

指令对应的输入和输出数据类型	
输入 X 的数据类型为 DWORD，输入 N 的数据类型为 BYTE，输入 B 的数据类型为 BOOL。 输出的数据类型为 DWORD。	
变量定义	
	
编程语言	**程序**
梯形图 （LD）	
结构化文本 （ST）	Var1:=PUTBIT(66,5,TRUE); (*结果 Var1 为 2#00000000000000000000000001100010*)

3.10.2. UNPACK——位拆分指令

若要将字节型输入数据拆分成 8 个独立的位，可使用位拆分指令 UNPACK。位拆分指令 UNPACK 是位整合指令 PACK 的逆运算指令。

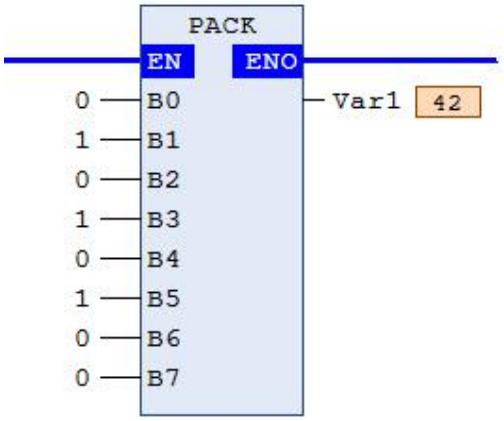
指令对应的输入和输出数据类型	
输入的数据类型为 BYTE。 输出 B0-B7 的数据类型为 BOOL。	
变量定义	
	
编程语言	**程序**

梯形图 （LD）	
结构化文本 （ST）	UNPACK_0(B:=Var1); VarBOOL0:=UNPACK_0.B0; VarBOOL1:=UNPACK_0.B1; VarBOOL2:=UNPACK_0.B2; VarBOOL3:=UNPACK_0.B3; VarBOOL4:=UNPACK_0.B4; VarBOOL5:=UNPACK_0.B5; VarBOOL6:=UNPACK_0.B6; VarBOOL7:=UNPACK_0.B7; (*Var1=2#00001111*) (*结果 VarBOOL0 为 TRUE*) (*结果 VarBOOL1 为 TRUE*) (*结果 VarBOOL2 为 TRUE*) (*结果 VarBOOL3 为 TRUE*) (*结果 VarBOOL4 为 FALSE*) (*结果 VarBOOL5 为 FALSE*) (*结果 VarBOOL6 为 FALSE*) (*结果 VarBOOL7 为 FALSE*)

3.10.3. PACK——位整合指令

若要将多个布尔型输入数据合成为字节型数据，可使用位整合指令 PACK。

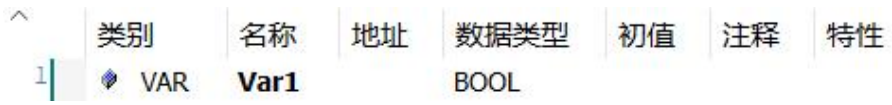
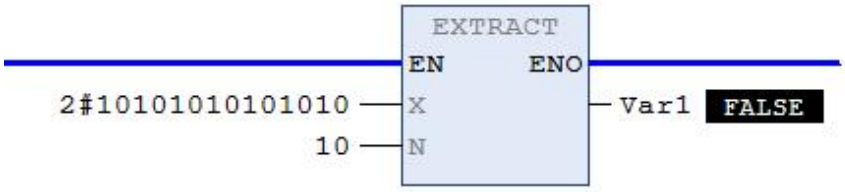
指令对应的输入和输出数据类型	
输入 B0-B7 的数据类型为 BOOL。 输出的数据类型为 BYTE。	
变量定义	
	
编程语言	程序

梯形图 （LD）	 (*结果 Var1 为 42，即 2#00101010*)
结构化文本 （ST）	. Var1:=PACK(0,1,0,1,0,1,0,0); (*结果 Var1 为 42，即 2#00101010*)

提示：在应用该指令时需注意，在 LD 语言中，字节的存储是由 B7 作为高位、B0 作为低位进行存储的。但在 IL 语言和 ST 语言中，先声明的部分是低位，结尾部分是高位，切勿弄错顺序。

3.10.4. EXTRACT——位提取指令

若要从输入数据中提取对应位的值，可使用位提取指令 EXTRACT。

指令对应的输入和输出数据类型	
输入 X 的数据类型为 DWORD，输入 N 的数据类型为 BYTE。 输出的数据类型为 BOOL。	
变量定义	
	
编程语言	**程序**
梯形图（LD）	 (*结果 Var1 为 FALSE*)
结构化文本 （ST）	Var1:=PUTBIT(2#10101010101010,10); (*结果 Var1 为 FALSE*)

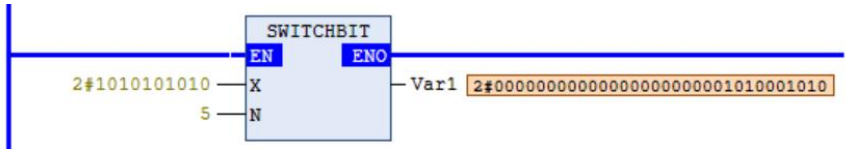
提示：该指令的功能是提取输入变量 X 的第 N 位输出，N 从零开始计算。

3.10.5. SWITCHBIT——位取反指令

若要从输入数据中取反对应位的值，可使用位取反指令 SWITCHBIT。

指令对应的输入和输出数据类型
输入 X 的数据类型为 DWORD，输入 N 的数据类型为 BYTE。 输出的数据类型为 DWORD。
变量定义

	类别	名称	地址	数据类型	初值	注释	属性
1	VAR	Var1		DWORD			

编程语言	程序
梯形图（LD）	 (*结果 Var1 为 2#00000000000000000000001010001010*)
结构化文本（ST）	Var1:=SWITCHBIT(X:=2#1010101010, N:=5);; (*结果 Var1 为 2#00000000000000000000001010001010*)

提示：该指令的功能是取反输入变量 X 的第 N 位输出，N 从零开始计算。

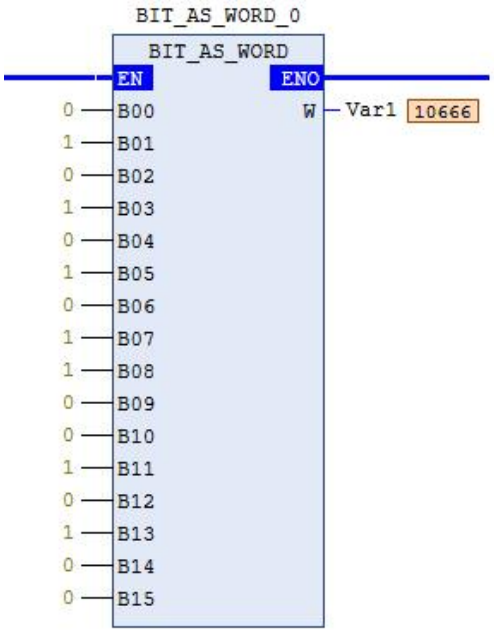
3.10.6. BIT_AS_WORD——位整合字指令

若要将多个布尔型输入数据合成为字类型数据，可使用位整合字指令 BIT_AS_WORD。

指令对应的输入和输出数据类型
输入 B00-B15 的数据类型为 BOOL。 输出的数据类型为 WORD。

变量定义

	类别	名称	地址	数据类型	初值	注释	属性
1	VAR	Var1		WORD			
2	VAR	BIT_AS_WORD_0		BIT_AS_WORD			

编程语言	程序
梯形图（LD）	 (*结果 Var1 为 10666，即 2#0010100110101010*)
结构化文本（ST）	BIT_AS_WORD0(B00:=0, B01:=1, B02:=0,

	B03:=1, B04:=0, B05:=1, B06:=0, B07:=1, B08:=1, B09:=0, B10:=0, B11:=1, B12:=0, B13:=1, B14:=0, B15:=0, W=>Var1); (*结果 Var1 为 10666，即 2#0010100110101010*)

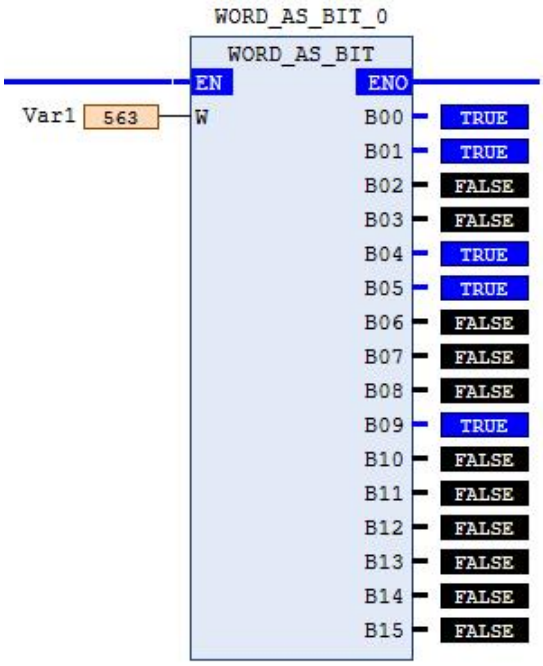
提示：在应用该指令时需注意，在 LD 语言中，字节的存储是由 B15 作为高位、B00 作为低位进行存储的。但在 IL 语言和 ST 语言中，先声明的部分是低位，结尾部分是高位，切勿弄错顺序。

3.10.7. WORD_AS_BIT——字提取位指令

若要将输入字类型的数据中提取对应位的值，可使用字提取位指令 WORD_AS_BIT。

指令对应的输入和输出数据类型	
输入 W 的数据类型为 WORD。 输出的数据类型为 BOOL。	
变量定义	

	类别	名称	地址	数据类型	初值	注释	属性
1	VAR	**Var1**		WORD			
2	VAR	**WORD_AS_BIT_0**		WORD_AS_BIT			

编程语言	程序
梯形图（LD）	 (*输入 Var1 为 563，结果为 2#0000001000110011*)

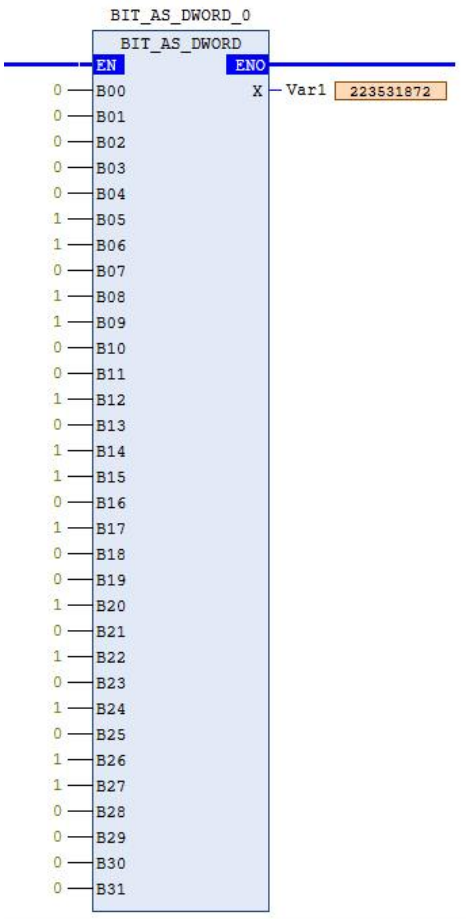
结构化文本（ST）	WORD_AS_BIT_0(W:=Var1 , B00=> , B01=> , B02=> , B03=> , B04=> , B05=> , B06=> , B07=> , B08=> , B09=> , B10=> , B11=> , B12=> , B13=> , B14=> , B15=>); (*输入 Var1 为 563，结果为 2#0000001000110011*)

提示：该指令的功能是提取输入变量 W，输出为位。

3.10.8. BIT_AS_DWORD——位整合双字指令

若要将多个布尔型输入数据合成为双字型数据，可使用位整合双字指令 BIT_AS_DWORD。

指令对应的输入和输出数据类型						
输入 B00-B31 的数据类型为 BOOL。 输出的数据类型为 DWORD。						
变量定义						
	类别	名称	地址	数据类型	初值	注释	属性
1	VAR	**Var1**		DWORD			
2	VAR	**BIT_AS_DWORD_0**		BIT_AS_DWORD			
编程语言	**程序**						

<table>
<tr>
<td>梯形图
（LD）</td>
<td> 

(*结果 Var1 为 223531872，即 2#00001101010100101101001101100000*)</td>
</tr>
<tr>
<td>结构化文本
（ST）</td>
<td>BIT_AS_WORD0(
B00:=0,
B01:=0,
B02:=0,
B03:=0,
B04:=0,
B05:=1,
B06:=1,
B07:=0,
B08:=1,
B09:=1,
B10:=0,
B11:=0,
B12:=1,
B13:=0,
B14:=1,
B15:=1,
B16:=0,
B17:=1,
B18:=0,
B19:=0,
B20:=1,
B21:=0,</td>
</tr>
</table>

	B22:=1, B23:=0, B24:=1, B25:=0, B26:=1, B27:=1, B28:=0, B29:=0, B30:=0, B31:=0, W=>Var1); (*结果 Var1 为 223531872，即 2#00001101010100101101001101100000*)

提示：在应用该指令时需注意，在 LD 语言中，字节的存储是由 B31 作为高位、B00 作为低位进行存储的。但在 IL 语言和 ST 语言中，先声明的部分是低位，结尾部分是高位，切勿弄错顺序。

3.10.9. DWORD_AS_BIT——双字提取位指令

若要将输入双字型的数据中提取对应位的值，可使用双字提取位指令 DWORD_AS_BIT。

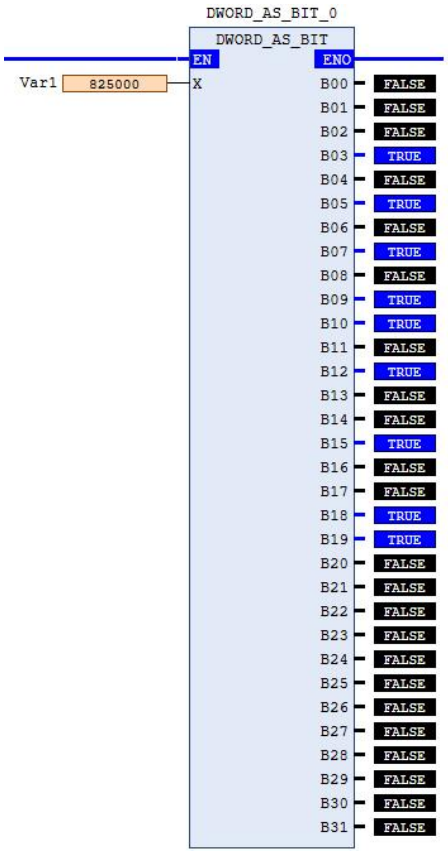
指令对应的输入和输出数据类型

输入 X 的数据类型为 DWORD。

输出的数据类型为 BOOL。

变量定义

	类别	名称	地址	数据类型	初值	注释	属性
1	VAR	**Var1**		DWORD			
2	VAR	**DWORD_AS_BIT_0**		DWORD_AS_BIT			

编程语言	程序
梯形图（LD）	

	(*输入 Var1 为 825000，结果为 2#00000000000011001001011010101000*)
结构化文本（ST）	DWORD_AS_BIT_0(W:=Var1 , B00=> , B01=> , B02=> , B03=> , B04=> , B05=> , B06=> , B07=> , B08=> , B09=> , B10=> , B11=> , B12=> , B13=> , B14=> , B15=> , B16=> , B17=> , B18=> , B19=> , B20=> , B21=> , B22=> , B23=> , B24=> , B25=> , B26=> , B27=> , B28=> , B29=> , B30=> , B31=>); (*输入 Var1 为 825000，结果为 2#00000000000011001001011010101000*)

提示：该指令的功能是提取输入变量 W，输出为位。

3.11. BCD 码转换指令（Util.compiled-library）

工程应用中大量采用 BCD 码，其数据存储格式为：每一字节 BCD 码代表 0 至 99 之间的一个两位整数，该整数的个位数存储在字节的第 3 位至第 0 位，该整数的十位数存储在字节的第 7 位至第 4 位。尽管 BCD 码的格式与 16 进制的格式比较相似，但它们的范围不同，BCD 码的范围为 0-99，而 16 进制的范围为 0-FF。

十进制 39 转换成 BCD 码，3 的二进制是 0011，9 的二进制是 1001，那么 39 转换 BCD 码为 00111001。十进制数 88 表示成 BCD 码为 10001000，表示成二进制为 2#0101 1000，表示成十六进制为 16#58。

3.11.1. INT_TO_BCD——整数型转 BCD 码指令

整数型转 BCD 码指令用于把整数类型的输入数据转换为 BCD 码输出。

指令对应的输入和输出数据类型

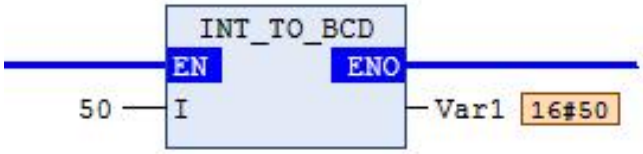
输入的数据类型为 INT。
输出的数据类型为 BYTE。

变量定义

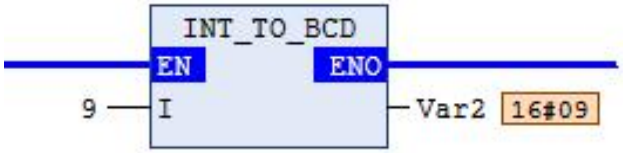
	类别	名称	地址	数据类型	初值	注释	特性
1	VAR	Var1		BYTE			
2	VAR	Var2		BYTE			
3	VAR	Var3		BYTE			

编程语言 | **程序**

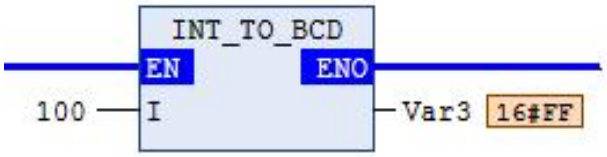
梯形图（LD）



(*结果 Var1 为 16#50*)



(*结果 Var2 为 16#09*)



(*错误！结果 Var3 为 16#FF*)

结构化文本（ST）

```
Var1:=INT_TO_BCD(50);
(*结果 Var1 为 16#50*)
Var2:=INT_TO_BCD(9);
(*结果 Var2 为 16#09*)
Var3:=INT_TO_BCD(100);
(*错误！结果 Var3 为 16#FF*)
```

提示：该指令对应的输入为 INT 型，输出为 BYTE 型，是转换后的 BCD 码值。若输入的数据不能转换成 BCD 码时，输出结果为 255，即 16#FF。

3.11.2. BCD_TO_INT——BCD 码转整数型指令

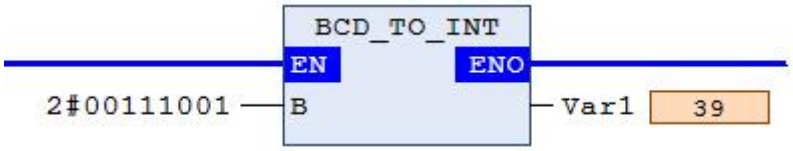
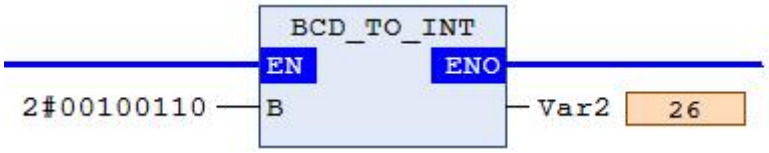
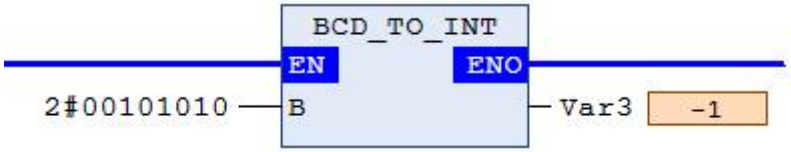
BCD 码转整数型指令用于把 BCD 码的输入数据转换为整数类型输出。

指令对应的输入和输出数据类型

输入的数据类型为 BYTE。
输出的数据类型为 INT。

变量定义

	类别	名称	地址	数据类型	初值	注释	特性
1	VAR	Var1		INT			
2	VAR	Var2		INT			
3	VAR	Var3		INT			

编程语言	程序
梯形图（LD）	 (*结果 Var1 为 39*)  (*结果 Var2 为 26*)  (*输出-1，因为后四位 1010 不是 BCD 码格式*)
结构化文本（ST）	Var1:=BCD_TO_INT(2#00111001); (*结果 Var1 为 39*) Var2:= BCD_TO_INT(2#00100110); (*结果 Var2 为 26*) Var3:= BCD_TO_INT(2#00101010); (*输出-1，因为后四位 1010 不是 BCD 码格式*)

提示：该函数用于将一个 BCD 位转化为一个 INT 值，如果要转换的字节不是 BCD 格式，那么结果为-1。

3.11.3. WORD_TO_BCD——字类型转 BCD 码指令

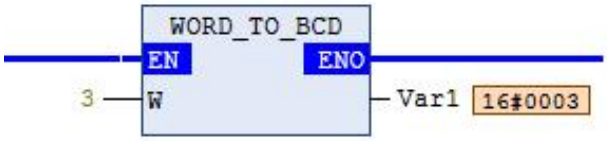
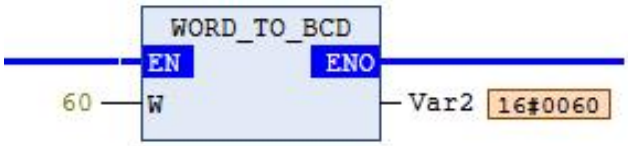
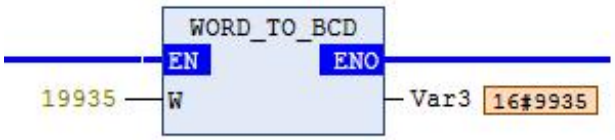
字类型转 BCD 码指令用于把字类型的输入数据转换为 BCD 码输出。

指令对应的输入和输出数据类型
输入的数据类型为 WORD。 输出的数据类型为 WORD。

变量定义

	类别	名称	地址	数据类型	初值	注释	属性
1	VAR	Var1		WORD			
2	VAR	Var2		WORD			
3	VAR	Var3		WORD			

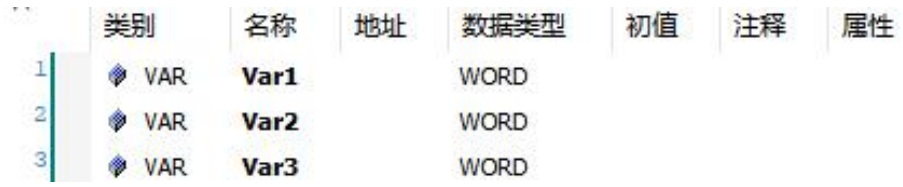
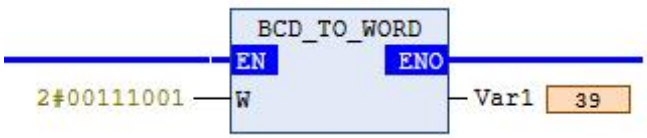
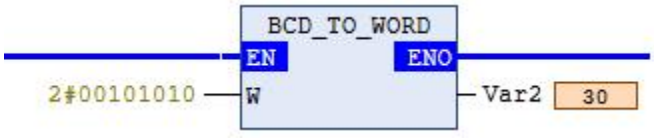
编程语言	程序

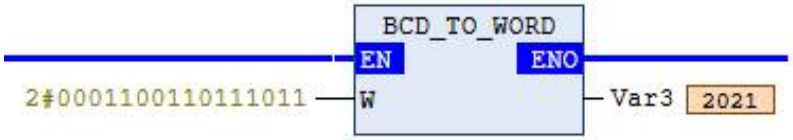
梯形图 （LD）	 (*结果 Var1 为 16#0003*)  (*结果 Var2 为 16#0060*)  (*错误！结果 Var3 为 16#9935*)
结构化文本 （ST）	Var1:=WORD_TO_BCD(3); (*结果 Var1 为 16#0003*) Var2:=WORD_TO_BCD(60); (*结果 Var2 为 16#0060*) Var3:=WORD_TO_BCD(19935); (*错误！结果 Var3 为 16#9935*)

提示：该指令对应的输入为 WORD 型，输出为 WORD 型，是转换后的 BCD 码值。输入数据的范围为 0~9999，若输入的数据超过 9999 不能转换成 BCD 码时，输出结果为后四位。

3.11.4. BCD_TO_WORD——BCD 码转字类型指令

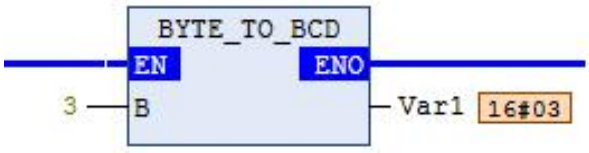
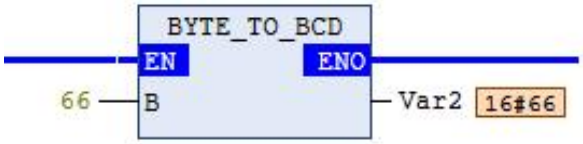
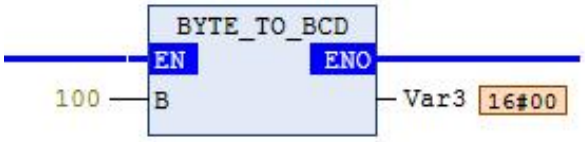
BCD 码转字类型指令用于把 BCD 码的输入数据转换为字类型输出。

指令对应的输入和输出数据类型	
输入的数据类型为 WORD。 输出的数据类型为 WORD。	
变量定义	
	
编程语言	程序
梯形图 （LD）	 (*结果 Var1 为 39*)  (*结果 Var2 为 30*)

	 (*结果 Var2 为 2021*)
结构化文本（ST）	Var1:=BCD_TO_WORD(2#00111001); (*结果 Var1 为 39*) Var2:= BCD_TO_WORD(2#00101010); (*结果 Var2 为 30*) Var3:= BCD_TO_WORD(2#0001100110111011); (*结果 Var2 为 2021*)

3.11.5. BYTE_TO_BCD——字节型转 BCD 码指令

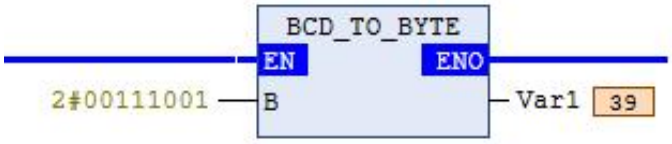
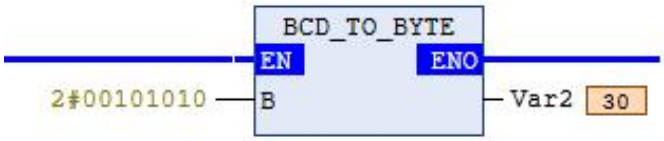
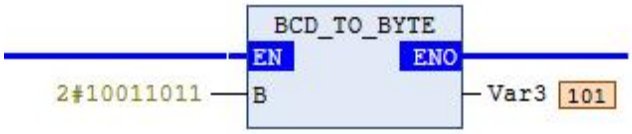
字节型转 BCD 码指令用于把字节型的输入数据转换为 BCD 码输出。

指令对应的输入和输出数据类型	
输入的数据类型为 BYTE。 输出的数据类型为 BYTE。	
变量定义	
	
编程语言	**程序**
梯形图（LD）	 (*结果 Var1 为 16#03*)  (*结果 Var2 为 16#66*)  (*错误！结果 Var3 为 16#00*)
结构化文本（ST）	Var1:=BYTE_TO_BCD(3); (*结果 Var1 为 16#03*) Var2:=BYTE_TO_BCD(66); (*结果 Var2 为 16#66*) Var3:=BYTE_TO_BCD(100); (*错误！结果 Var3 为 16#00*)

提示：该指令对应的输入为 BYTE 型，输出为 BYTE 型，是转换后的 BCD 码值。输入数据的范围为 0~99，若输入的数据超过 99 不能转换成 BCD 码时，输出结果为后两位。

3.11.6. BCD_TO_BYTE——BCD 码转字节型指令

BCD 码转字类型指令用于把 BCD 码的输入数据转换为字节型输出。

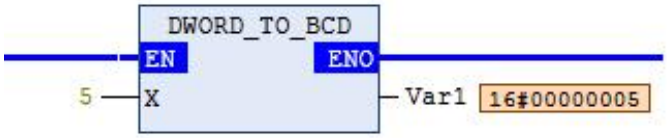
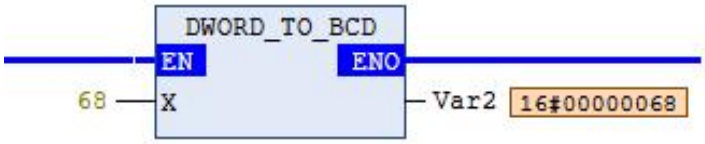
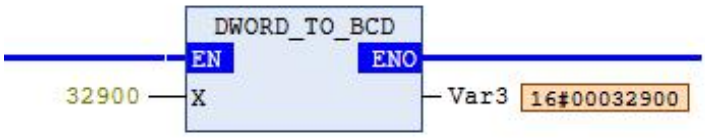
指令对应的输入和输出数据类型	
输入的数据类型为 BYTE。 输出的数据类型为 BYTE。	
变量定义	
	
编程语言	**程序**
梯形图 （LD）	 (*结果 Var1 为 39*)  (*结果 Var2 为 30*)  (*结果 Var2 为 101*)
结构化文本 （ST）	Var1:=BCD_TO_BYTE(2#00111001); (*结果 Var1 为 39*) Var2:= BCD_TO_BYTE(2#00101010); (*结果 Var2 为 30*) Var3:= BCD_TO_BYTE(2#10011011); (*结果 Var2 为 101*)

3.11.7. DWORD_TO_BCD——双字型转 BCD 码指令

双字型转 BCD 码指令用于把双字型的输入数据转换为 BCD 码输出。

指令对应的输入和输出数据类型
输入的数据类型为 DWORD。 输出的数据类型为 DWORD。
变量定义

	类别	名称	地址	数据类型	初值	注释	属性
1	VAR	Var1		DWORD			
2	VAR	Var2		DWORD			
3	VAR	Var3		DWORD			

编程语言	程序
梯形图（LD）	 (*结果 Var1 为 16#00000005*)  (*结果 Var2 为 16#00000068*)  (*结果 Var3 为 16#00032900*)
结构化文本（ST）	Var1:=WORD_TO_BCD(3); (*结果 Var1 为 16#0003*) Var2:=WORD_TO_BCD(60); (*结果 Var2 为 16#0060*) Var3:=WORD_TO_BCD(19935); (*结果 Var3 为 16#00032900*)

提示：该指令对应的输入为 WORD 型，输出为 WORD 型，是转换后的 BCD 码值。输入数据的范围为 0~99999999，若输入的数据超过 99999999 不能转换成 BCD 码时，输出结果为后八位。

3.11.8. BCD_TO_DWORD——BCD 码转双字型指令

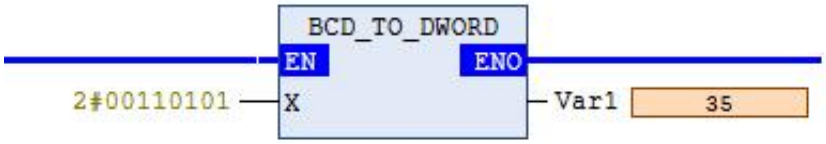
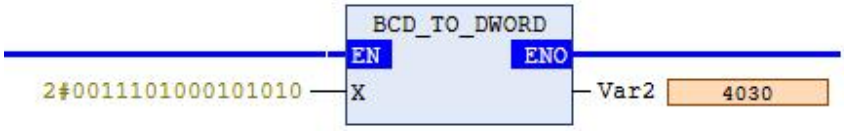
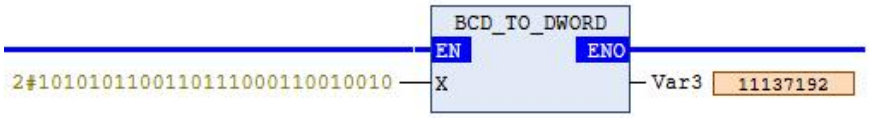
BCD 码转双字型指令用于把 BCD 码的输入数据转换为双字型输出。

指令对应的输入和输出数据类型
输入的数据类型为 DWORD。 输出的数据类型为 DWORD。

变量定义

	类别	名称	地址	数据类型	初值	注释	属性
1	VAR	Var1		DWORD			
2	VAR	Var2		DWORD			
3	VAR	Var3		DWORD			

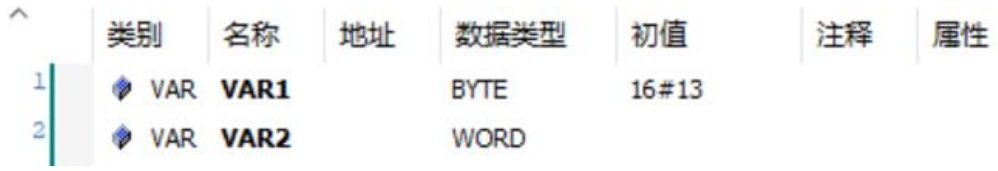
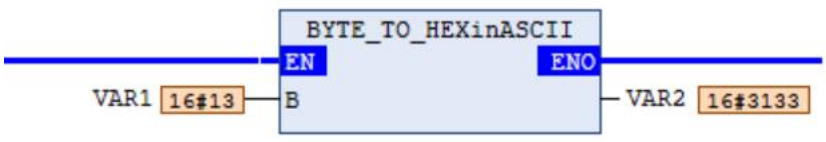
编程语言	程序

梯形图 （LD）	 (*结果 Var1 为 35*)  (*结果 Var2 为 4030*)  (*结果 Var2 为 11137192*)
结构化文本 （ST）	Var1:=BCD_TO_DWORD(2#00110101); (*结果 Var1 为 35*) Var2:= BCD_TO_DWORD(2#0011101000101010); (*结果 Var2 为 4030*) Var3:= BCD_TO_DWORD(2#10101011001101110000110010010); (*结果 Var2 为 11137192*)

3.12. ASCII 码转换指令（Util.compiled-library）

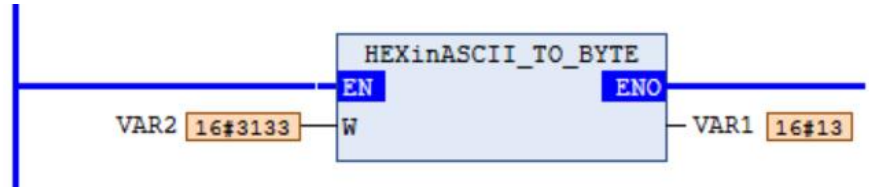
3.12.1. BYTE_TO_HEXinASCII——字节型转换为 ASCII 码指令

将 1 个字节的高 4 位与低 4 位分别转化为 ASCII 码并存储在 1 个字的高 8 位与低 8 位中。

指令对应的输入和输出数据类型	
输入的数据类型为 BYTE。 输出的数据类型为 WORD。	
变量定义	
	
编程语言	**程序**
梯形图 （LD）	 (*Var2 结果为 16#3133*)
结构化文本 （ST）	Var2:=BYTE_TO_HEXinASCII(B:=VAR1); (*Var2 结果为 16#3133*)

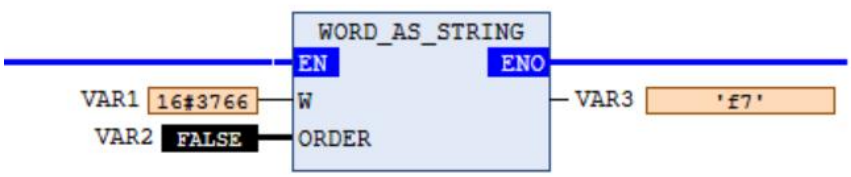
3.12.2. HEXinASCII_TO_BYTE——ASCII 码转换为字节型指令

将 1 个字的高 8 位与低 8 位 ASCII 码分别转化为 1 个字节的高 4 位与低 4 位。

指令对应的输入和输出数据类型	
输入的数据类型为 WORD。 输出的数据类型为 BYTE。	
变量定义	
	
编程语言	**程序**
梯形图 （LD）	 (*Var1 结果为 16#13*)
结构化文本 （ST）	Var1:=HEXinASCII_TO_BYTE(W:=VAR2); (*Var1 结果为 16#13*)

3.12.3. WORD_AS_STRING——字类型转 ASCII 码指令

将 1 个字的高 8 位与低 8 位分别转化为 ASCII 码并存储在字符类型中。

指令对应的输入和输出数据类型	
输入的数据类型为 WORD，高 8 位与低 8 位的 16 进制数范围分别为（20H-7FH）。 输出的数据类型为 STRING(2)。 该指令中控制位 ORDER 数据类型为 BOOL，将转换的高低字符对调。	
变量定义	
	
编程语言	**程序**
梯形图 （LD）	 (*Var1 结果为’f7’*)
结构化文本 （ST）	Var3:=WORD_AS_STRING(W:=VAR1, ORDER:=VAR2); (*Var1 结果为’f7’*)

3.13. 信号发生器指令（Util.compiled-library）

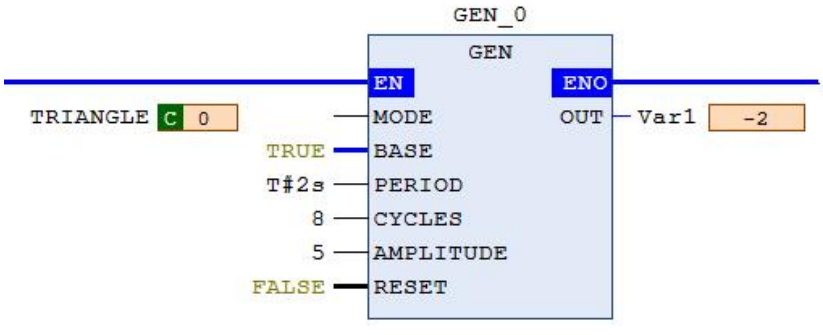
3.13.1. GEN——典型周期信号发生器

典型周期信号发生器指令用于生成典型的周期信号，包括三角波信号、零起点三角波信号、上升锯齿波信号、下降锯齿波信号、方波信号、正弦波信号和余弦波信号。

输入参数	数据类型	功能描述	参数值说明
MODE	GEN_MODE	指定要产生的信号类型	引脚输入名称及对应信号类型如下： 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 COSINUS，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。
PERIOD	TIME	循环周期	
CYCLES	INT	发生的个数	
AMPLITUDE	INT	信号的振幅	
RESET	BOOL	初始化	当 RESET=TRUE 时，信号发生器被重新设置为 0。
输出参数	**数据类型**	**功能描述**	**参数值说明**
OUT	INT	波形信号输出值	

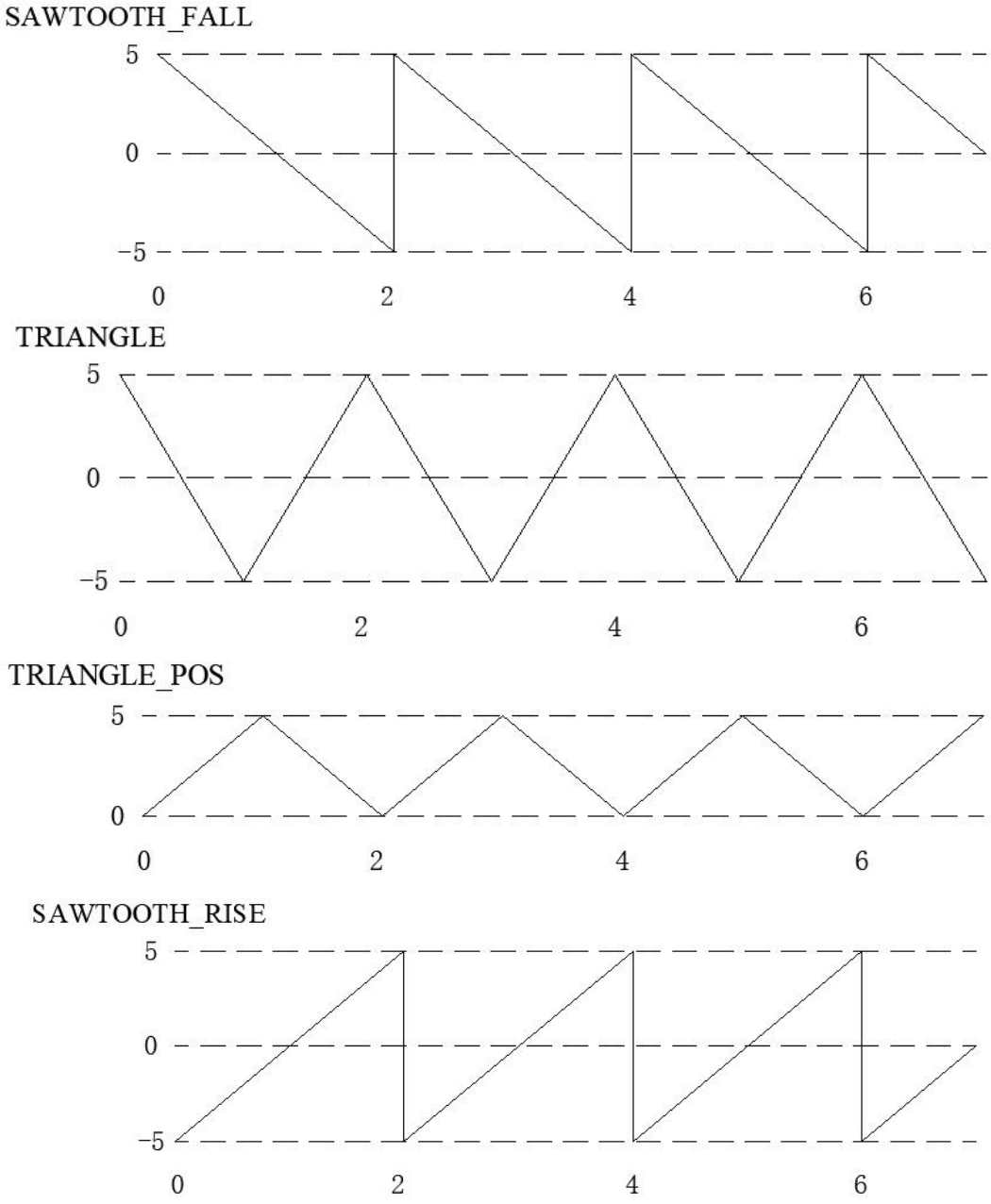
变量定义

	类别	名称	地址	数据类型	初值	注释	特性
1	VAR	Var1		INT			
2	VAR	GEN_0		GEN			

编程语言	程序
梯形图（LD）	

结构化文本（ST）	GEN_0(MODE:=TRIANGLE, BASE:=TRUE,PERIOD:=T#2s, CYCLES:=8, AMPLITUDE:=5, RESET:=FALSE); Var1:= GEN_0.OUT;

在 MODE 处输入不同的信号类型，得到不同的波形，如下所示。



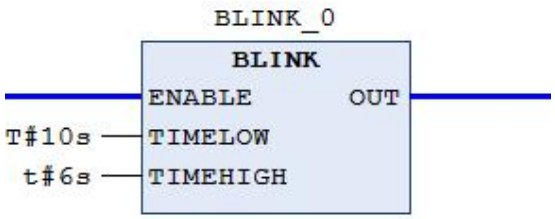
3.13.2. BLINK——脉冲信号发生器

信号发生指令 BLINK 用于产生脉冲信号。

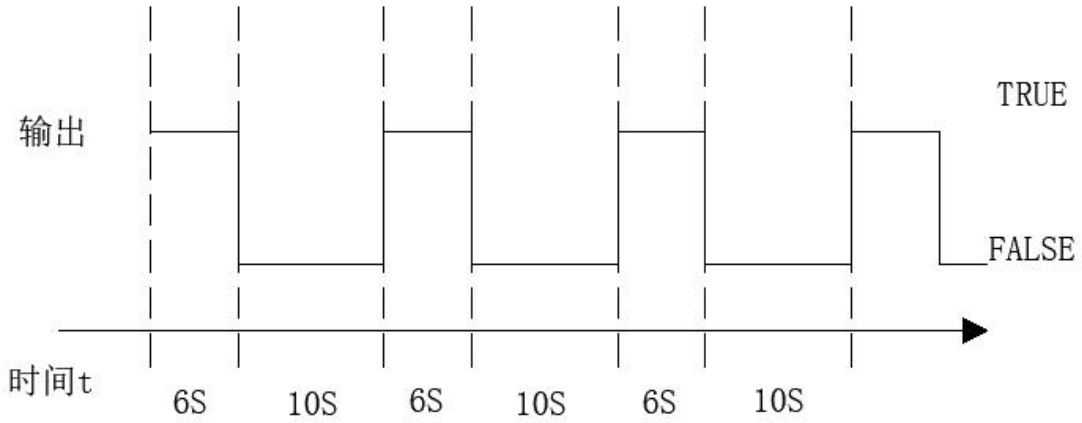
输入参数	数据类型	功能描述	参数值说明
ENABLE	BOOL	使能	TRUE 时开始工作
TIMELOW	TIME	输出低电平时间	

TIMEHIGH	TIME	输出高电平时间	
输出参数	**数据类型**	**功能描述**	**参数值说明**
OUT	BOOL	脉冲信号输出值	

变量定义						
	类别	名称	地址	数据类型	初值	注释	特性
1	VAR	Var1		BOOL			
2	VAR	BLINK_0		BLINK			

编程语言	程序
梯形图（LD）	
结构化文本（ST）	BLINK_0 (TIMELOW:=T#10s, TIMEHIGH:=T#6s); Var1:=BLINK_0.OUT;

指令执行时，输出下图所示的波形。



提示：

1. **系统设计时已确定，该指令输出的第一个电平一定是高电平。**
2. **当使能端断开时输出保持不变。**

3.13.3. FREQ_MEASURE——频率测量指令

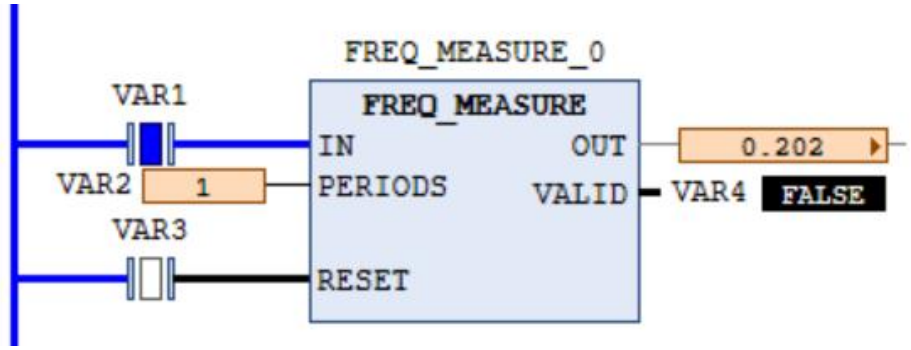
测量输入的布尔类型信号的频率值。

输入参数	数据类型	功能描述	参数值说明
IN	BOOL	输入信号	TRUE 时开始工作
PERIODS	INT	测量周期	范围 1 到 10，默认值为 1，输入信号两个上升沿的时间间隔为一个周期，将 N 个周期的频率值求平均值就得到输入信号的频率，此参数即为求平均值运算的周

			期次数
RESET	BOOL	复位	TRUE 时重新测量，正常执行时为 FALSE
输出参数	**数据类型**	**功能描述**	**参数值说明**
OUT	REAL	频率输出	输入信号的频率值，单位 Hz
VALID	BOOL	运算标志	第一次运算完成后或者当运算错误置 TRUE，其余时刻为 FALSE

变量定义

	类别	名称	地址	数据类型	初值	注释	属性
1	VAR	FREQ_MEASURE_0		FREQ_MEASURE			
2	VAR	VAR1		BOOL			
3	VAR	VAR2		INT	1		
4	VAR	VAR3		BOOL			
5	VAR	VAR4		BOOL			

编程语言	程序
梯形图（LD）	
结构化文本（ST）	FREQ_MEASURE_0(IN:=VAR1, PERIODS:=VAR2, RESET:=VAR3, OUT=> , VALID=>VAR4);

3.14. 函数操纵器指令（Util.compiled-library）

3.14.1. RAMP_INT——整型限速

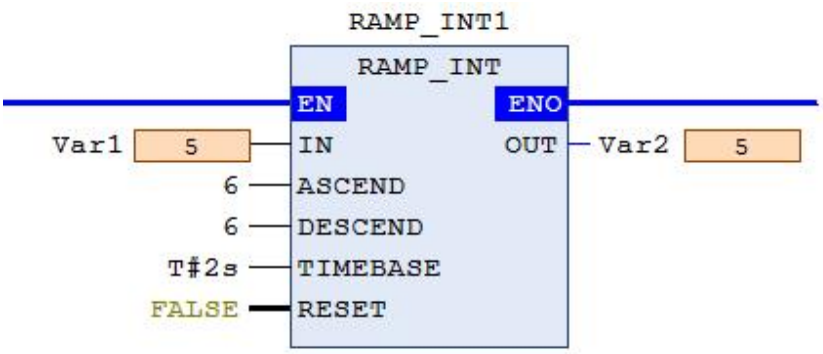
整型限速指令 RAMP_INT 用于对整型输入数据升降速度进行限制，防止输入数据变化过快。

输入参数	数据类型	功能描述	参数值说明
IN	BOOL	输入数据	若输入值大于上一周期的数值，按照时间基数和上升速率进行加法运算。若输入值小于上一周期的数值，按照时间基数和下降速率进行减法运算。
ASCEND	INT	上升速率	在时间基数内上升的数值
DESCEND	INT	下升速率	在时间基数内下降的数值

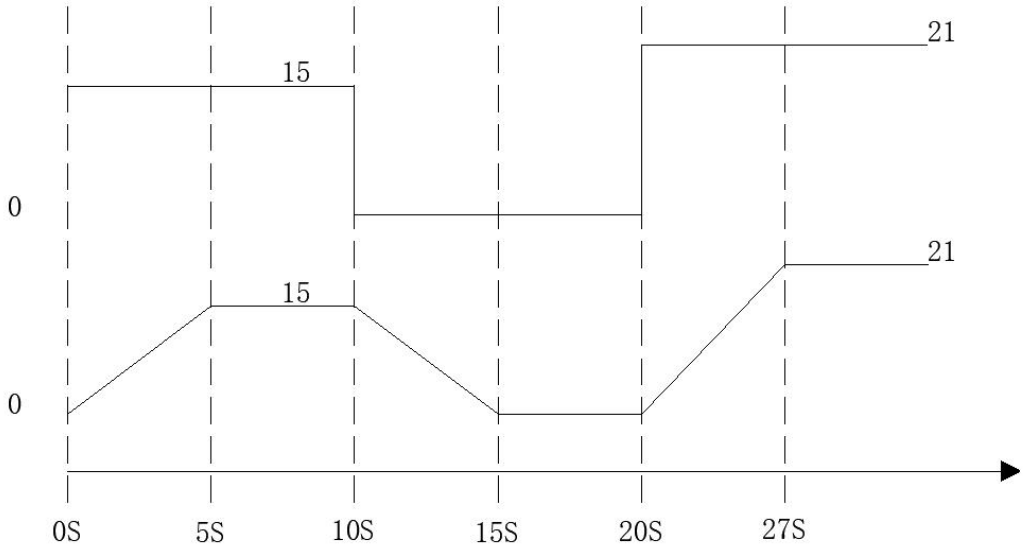
TIMEBASE	TIME	时间基数	按上升或者下降速率计算变化量的时间基数
RESET	BOOL	初始化	TRUE 时 RAMP_INT 被重新初始化
输出参数	**数据类型**	**功能描述**	**参数值说明**
OUT	INT	数据输出	

变量定义

	类别	名称	地址	数据类型	初值	注释	特性
1	VAR	**Var1**		INT			
2	VAR	**Var2**		INT			
3	VAR	**RAMP_INT1**		RAMP_INT			

编程语言	程序
梯形图（LD）	
结构化文本（ST）	RAMP_INT1(IN := Var1, ASCEND := 6, DESCEND :=6, TIMEBASE := T#2s, RESET := FALSE); Var2:=RAMP_INT1.OUT;

指令执行时，输出下图所示的波形。



3.14.2. RAMP_REAL——实型限速

实型限速指令 RAMP_REAL 用于对实型输入数据升降速度进行限制，防止输入数据变化过快。

输入参数	数据类型	功能描述	参数值说明
IN	REAL	输入数据	若输入值大于上一周期的数值，按照时间基数和上升速率进行加法运算。若输入值小于上一周期的数值，按照时间基数和下降速率进行减法运算。
ASCEND	REAL	上升速率	在时间基数内上升的数值
DESCEND	REAL	下升速率	在时间基数内下降的数值
TIMEBASE	TIME	时间基数	按上升或者下降速率计算变化量的时间基数
RESET	BOOL	初始化	TRUE 时 RAMP_REAL 被重新初始化
输出参数	**数据类型**	**功能描述**	**参数值说明**
OUT	REAL	数据输出	

该指令的应用举例可参见整数限速指令 RAMP_INT。

输入参数	数据类型	功能描述	参数值说明
IN	INT	输入 X 轴的数值	
N	BYTE	定义曲线所使用数组中的点数	（2≤N≤11） N=2 代表使用 P[0]. X、P[0]. Y， P[0]. X、P[0]. Y，2 个点。 N=11 代表使用 P[0]. X、P[0]. Y… P[10]. X、P[10]. Y，11 个点。
P	ARRAY[0..10] OF POINT		XY 平面上特征曲线的特征点
输出参数	**数据类型**	**功能描述**	**参数值说明**
OUT	INT	输出值	输入 X 值对应的输出 Y 值
ERR	BYTE	显示错误类型	ERR=1：数组中的点 P[0]..P[N-1]中的 X 值有错误。 ERR=2：输入值 IN 不在 P[0].X 和 P[N-1].X 之间。 若输入值小于 P[0].X，此时 OUT 的输出为 P[0].Y。 若输入值大于 P[N-1].X，此时 OUT 的输出为 P[N-1].Y。 ERR=4：输入 N 小于 2，或者大于 11。

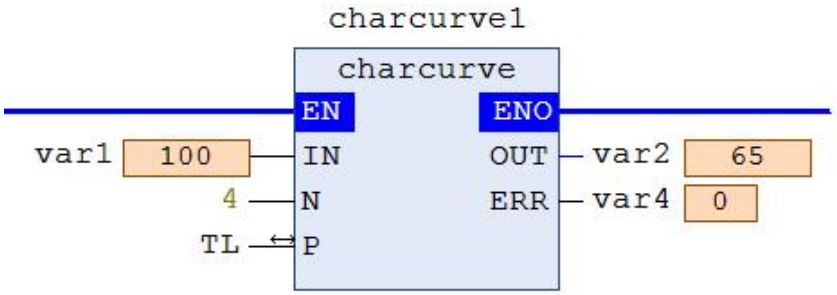
3.14.3. CHARCURVE——特征曲线

特征曲线指令 CHARCURVE 用于描述特征曲线上任意一点的位置。根据给定的二位数组，在 XY 平面上可绘制出一条特征曲线，该二位数组共有 N 个数据，分别为(x0，y0)、(x1，y1)、(x2，y2)…(XN-1，y N-1)。在此基础上，任意输入一个数值在 X0…XN-1 之间，可求取相应的 y 的数值。

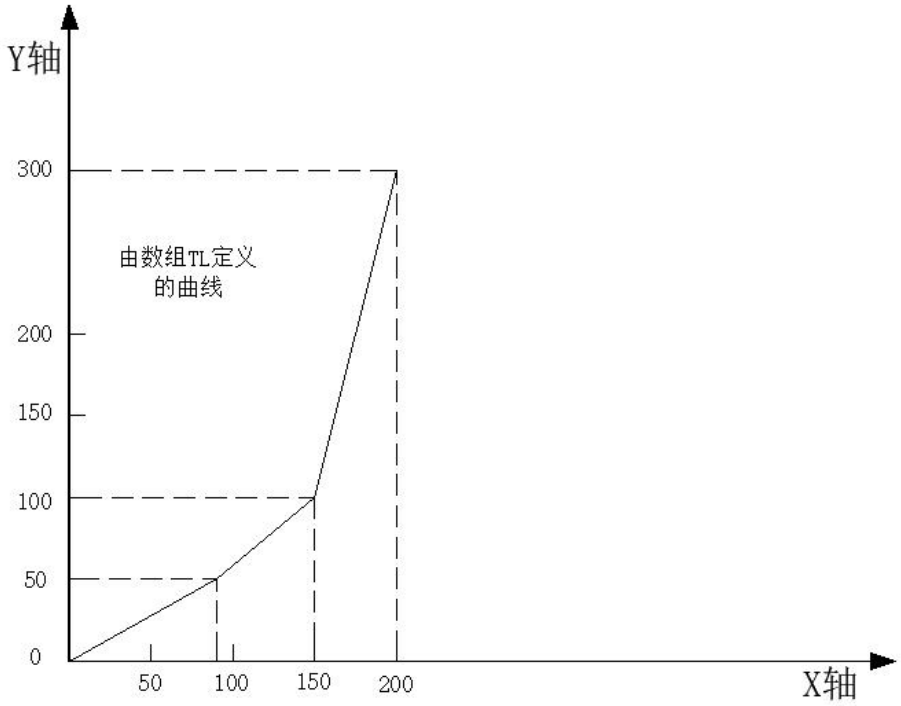
应用举例

变量定义

	类别	名称	地址	数据类型	初值	注释	特性
1	VAR	charcurve1		charcurve			
2	VAR	var1		INT			
3	VAR	var2		INT			
4	VAR	TL		ARRAY[0..10]OF POINT	[(X:=0,Y:=0),(X:=80,Y:=50),(X:=150,Y:=100),(X:=200,Y:=300)]		
5	VAR	var4		BYTE			

编程语言	程序
梯形图（LD）	
结构化文本（ST）	charcurve1 (IN := Var1, N := 4, P := TL); Var4:= charcurve1.ERR; Var2:= charcurve1.OUT;

上述程序运行时，可得到下图所示的特征曲线，输入 IN 值为 100，输出 OUT 的值为 65。



3.15. 高等数学运算指令（Util.compiled-library）

3.15.1. STATISTICS_INT——整型统计指令

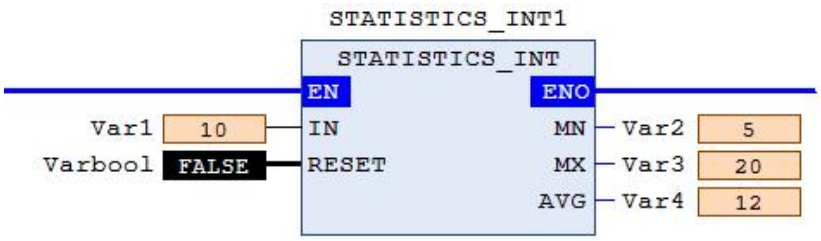
整型统计指令 STATISTIC_INT 用于统计输入整型数据的最大值、最小值和平均值。

输入参数	数据类型	功能描述	参数值说明
IN	INT	输入	

RESET	BOOL	初始化	为 TRUE 时，重新初始化
输出参数	**数据类型**	**功能描述**	**参数值说明**
MN	INT	最小值	
MX	INT	最大值	
AVG	INT	平均值	

变量定义

	类别	名称	地址	数据类型	初值	注释	特性
1	VAR	**Var1**		INT			
2	VAR	**Var2**		INT			
3	VAR	**Var3**		INT			
4	VAR	**Var4**		INT			
5	VAR	**Varbool**		BOOL			
6	VAR	**STATISTICS_INT1**		STATISTICS_INT			

编程语言	**程序**
梯形图（LD）	 (*整型统计指令是每改变一次 Var1 的值，计算一次*)
结构化文本（ST）	STATISTICS_INT1(IN := Var1, RESET := Varbool); Var2:=STATISTICS_INT1.MN; Var3:=STATISTICS_INT1.MX; Var4:=STATISTICS_INT1.AVG;

提示：当布尔型输入 RESET 为 TRUE 时，所有的变量值都将会被初始化。

3.15.2. STATISTICS_REAL——实型统计指令

实型统计指令 STATISTIC_REAL 用于统计输入实型数据的最大值、最小值和平均值。

输入参数	**数据类型**	**功能描述**	**参数值说明**
IN	REAL	输入	
RESET	BOOL	初始化	为 TRUE 时，重新初始化
输出参数	**数据类型**	**功能描述**	**参数值说明**
MN	REAL	最小值	
MX	REAL	最大值	
AVG	REAL	平均值	

该指令的应用举例可参见整型统计指令 STATISTICS_INT。

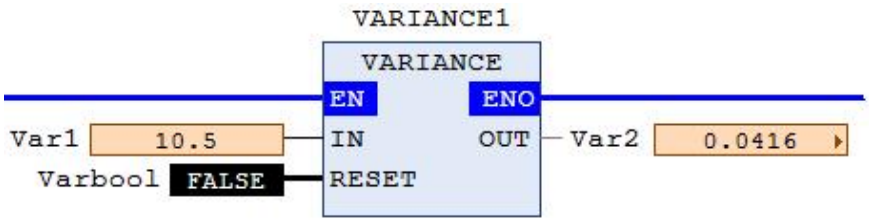
3.15.3. VARIANCE——平方偏差指令

平方偏差指令用于求解输入数据的反正弦值，输出数据为运算结果。

输入参数	数据类型	功能描述	参数值说明
IN	REAL	输入	
RESET	BOOL	复位	为 TRUE 时，指令复位
输出参数	**数据类型**	**功能描述**	**参数值说明**
OUT	REAL	平方偏差	

变量定义

	类别	名称	地址	数据类型	初值	注释	特性
1	VAR	**Var1**		REAL			
2	VAR	**Var2**		REAL			
3	VAR	**Varbool**		BOOL			
4	VAR	**VARIANCE1**		VARIANCE			

编程语言	程序
梯形图（LD）	 (*结果为 0.0416*)
结构化文本（ST）	VARIANCE1(IN:=Var1, RESET:=Varbool); Var2:=VARIANCE.OUT; (*结果为 0.0416*)

提示：得到一组数的平方偏差后，可对平方偏差值求平方根，得到该组数的标准偏差。

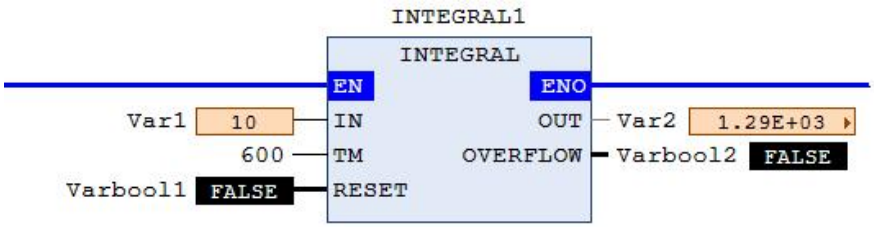
3.15.4. INTEGRAL——积分指令

积分指令用于求解输入数据的积分值，输出数据为运算结果。

输入参数	数据类型	功能描述	参数值说明
IN	REAL	连续输入的变量	
TM	DWORD	积分时间	单位秒
RESET	BOOL	复位信号	为 TRUE 时，重新启动指令
输出参数	**数据类型**	**功能描述**	**参数值说明**
OUT	REAL	积分运算结果	

变量定义

	类别	名称	地址	数据类型	初值	注释	特性
1	VAR	**Var1**		INT			
2	VAR	**Var2**		REAL			
3	VAR	**Varbool1**		BOOL			
4	VAR	**Varbool2**		BOOL			
5	VAR	**INTEGRAL1**		INTEGRAL			

编程语言	程序
梯形图（LD）	 (*结果为 1.29E+03*)
结构化文本（ST）	INTEGRAL1(IN := Var1, TM := 600, RESET := Varbool1); Var2:=INTEGRAL.OUT; Varbool2:=INTEGRAL.OVERFLOW; (*结果为 1.29E+03*)

提示：在 PLC 编程应用中，所有连续变化的输入量均需离散化后再计算、存储，故积分运算也是采用离散方式进行的，当积分周期足够小时可认为该计算接近实际连续过程。

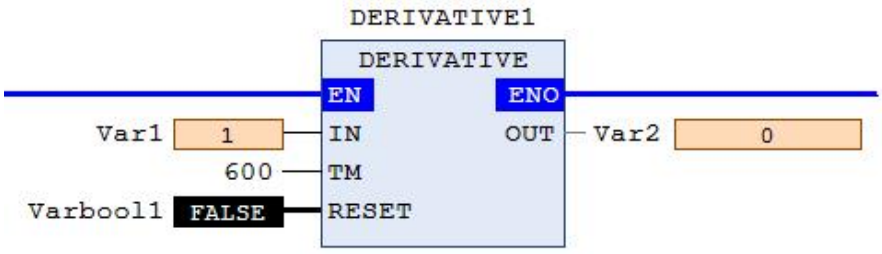
3.15.5. DERIVATIVE——微分指令

微分指令用于求解输入数据的微分值，输出数据为运算结果。

输入参数	数据类型	功能描述	参数值说明
IN	REAL	连续输入的变量	
TM	DWORD	微分时间	单位秒
RESET	BOOL	复位信号	为 TRUE 时，重新启动指令
输出参数	**数据类型**	**功能描述**	**参数值说明**
OUT	REAL	微分运算结果	

变量定义

	类别	名称	地址	数据类型	初值	注释	特性
1	VAR	**Var1**		INT			
2	VAR	**Var2**		REAL			
3	VAR	**Varbool1**		BOOL			
4	VAR	**Varbool2**		BOOL			
5	VAR	**DERIVATIVE1**		DERIVATIVE			

编程语言	程序
梯形图 （LD）	 (*结果为 0*)
结构化文本 （ST）	DERIVATIVE1(IN:=Var1, TM:=600, RESET:=Varbool1); Var2:=DERIVATIVE1.OUT; (*结果为 0*)

提示：微分指令对当前时刻的前 3 个运算周期的数据进行运算，以提高数据处理的准确性和平滑性，3 个周期所占权重分别为 30%、40%和 30%。

3.15.6. LIN_TRAFO——线性变换指令

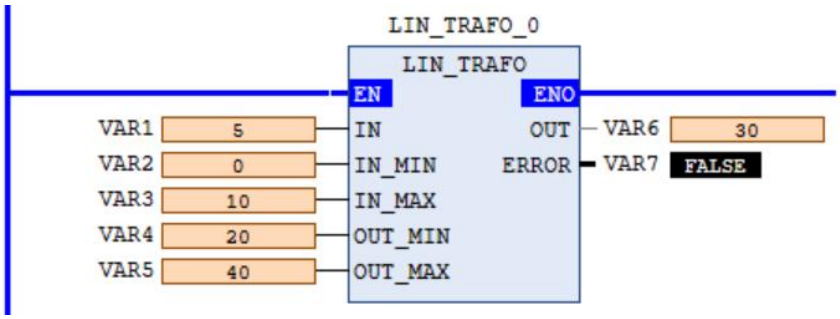
通过使用输入的最小值和最大值来实现输出值的线性近似值。

输入参数	数据类型	功能描述	参数值说明
IN	REAL	输入值	
IN_MIN	REAL	最小输入值	
IN_MAX	REAL	最大输入值	
OUT_MIN	REAL	最小输出值	
OUT_MAX	REAL	最大输出值	
输出参数	**数据类型**	**功能描述**	**参数值说明**
OUT	REAL	输出值	
ERROR	BOOL	错误代码	0：无错误 1：IN_MIN = IN_MAX 或 IN 超出输入范围

变量定义

	类别	名称	地址	数据类型	初值	注释	属性
1	VAR	**LIN_TRAFO_0**		LIN_TRAFO			
2	VAR	**VAR1**		REAL	5		
3	VAR	**VAR2**		REAL	0		
4	VAR	**VAR3**		REAL	10		
5	VAR	**VAR4**		REAL	20		
6	VAR	**VAR5**		REAL	40		
7	VAR	**VAR6**		REAL			
8	VAR	**VAR7**		BOOL			

编程语言	程序

梯形图 （LD）	 (*结果为 30*)
结构化文本 （ST）	LIN_TRAFO_0(IN:=VAR1, IN_MIN:=VAR2, IN_MAX:=VAR3, OUT_MIN:=VAR4, OUT_MAX:=VAR5, OUT=>VAR6, ERROR=>VAR7); (*结果为 30*)

3.16. 控制器指令（Util.compiled-library）

在对连续变化的过程参数进行调节时，目前应用最多的控制策略仍是经典的 PID 调节，即比例、积分、微分调节。

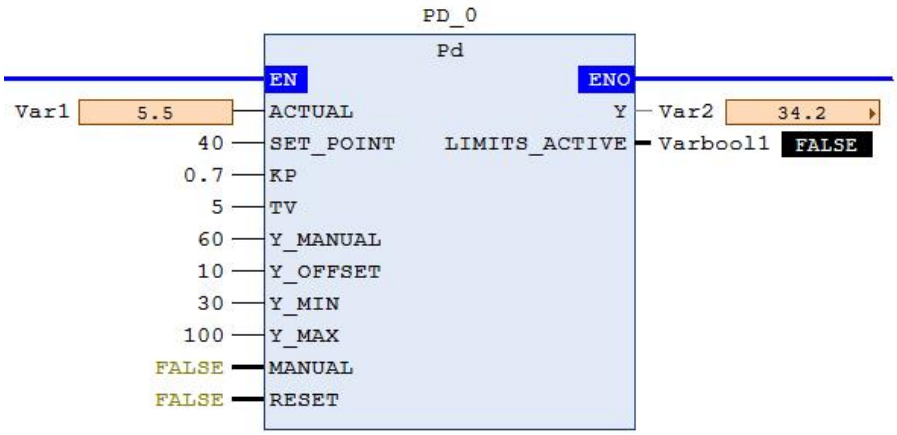
3.16.1. PD——比例微分控制器

比例微分控制器 PD 指令实现比例微分调节功能，该指令由测量值和设定值得到二者的差值，再根据差值求比例部分和微分部分。

输入参数	数据类型	功能描述	参数值说明
ACTUAL	REAL	测量值	
SET_POINT	REAL	设定值	
KP	REAL	比例系数	
TV	REAL	微分时间	单位秒
Y_MANUAL	REAL	手动值	MANUAL=TRUE 时，Y= Y_MANUAL
Y_OFFSET	REAL	输出偏移量	
Y_MIN	REAL	输出最小值	
Y_MAX	REAL	输出最大值	
MANUAL	BOOL	手自动选择	TRUE 为手动调节，FALSE 为自动调节
RESET	BOOL	重置	TRUE 时重置控制器，正常时应置 FALSE
输出参数	**数据类型**	**功能描述**	**参数值说明**
Y	REAL	输出值	
LIMITS_ACTIVE	BOOL	输出超限标志	超限时为 TRUE，不超限时为 FALSE

变量定义

	类别	名称	地址	数据类型	初值	注释	特性
1	VAR	Var1		REAL			
2	VAR	Var2		REAL			
3	VAR	Varbool1		BOOL			
4	VAR	PD_0		Pd			

编程语言	程序
梯形图（LD）	
结构化文本（ST）	PD_0(ACTUAL:=Var1, SET_POINT:=40, KP:=0.7, TV:=5, Y_MANUAL:=60, Y_OFFSET:=10, Y_MIN:=30, Y_MAX:=100, MANUAL:=FALSE, RESET:=FALSE, Y=>Var2, LIMITS_ACTIVE=>Varbool1);

3.16.2. PID——比例积分微分控制器

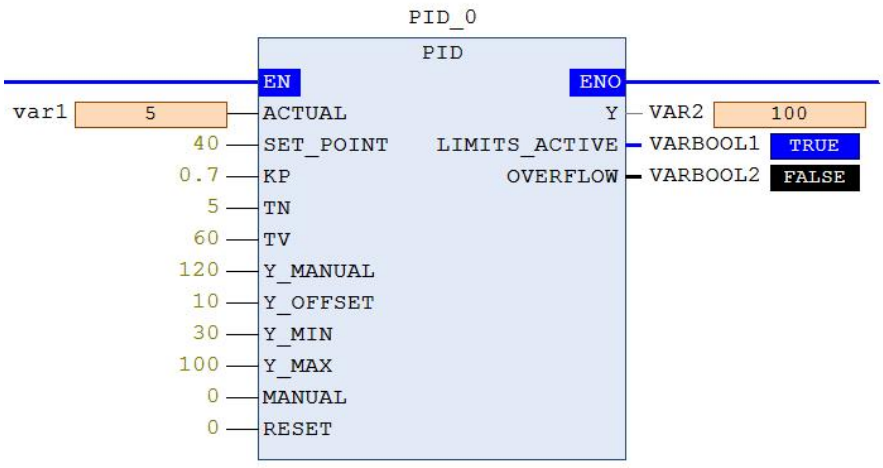
比例积分微分控制器 PID 指令实现比例积分微分调节功能，该指令由测量值和设定值得到二者的差值，再根据差值求比例部分、积分部分和微分部分。比例积分微分控制器中 TV=0 时，PID 控制器就变成了 PI 控制器。（建议使用 RPCMath 库中 RPCPID 指令）

输入参数	数据类型	功能描述	参数值说明
ACTUAL	REAL	测量值	
SET_POINT	REAL	设定值	
KP	REAL	比例系数	
TN	REAL	积分时间	单位为秒
TV	REAL	微分时间	单位为秒

Y_MANUAL	REAL	手动值	MANUAL=TRUE 时，Y= Y_MANUAL
Y_OFFSET	REAL	输出偏移量	
Y_MIN	REAL	输出最小值	
Y_MAX	REAL	输出最大值	
MANUAL	BOOL	手自动选择	TRUE 为手动调节，FALSE 为自动调节
RESET	BOOL	重置	TRUE 重置该控制器，正常时置 FALSE
输出参数	**数据类型**	**功能描述**	**参数值说明**
Y	REAL	输出值	
LIMITS_ACTIVE	BOOL	输出超限标志	输出值超限时为 TRUE
OVERFLOW	BOOL	输出溢出标志	积分溢出时为 TRUE

变量定义

	类别	名称	地址	数据类型	初值	注释	特性
1	VAR	**Var1**		REAL			
2	VAR	**Var2**		REAL			
3	VAR	**Varbool1**		BOOL			
4	VAR	**Varbool2**		BOOL			
5	VAR	**PID_0**		PID			

编程语言	程序
梯形图（LD）	
结构化文本（ST）	PID_0(ACTUAL:=Var1, SET_POINT:=40, KP:=0.7, TN:=5,TV:=60, Y_MANUAL:=120, Y_OFFSET:=10, Y_MIN:=30, Y_MAX:=100, MANUAL:= FALSE,

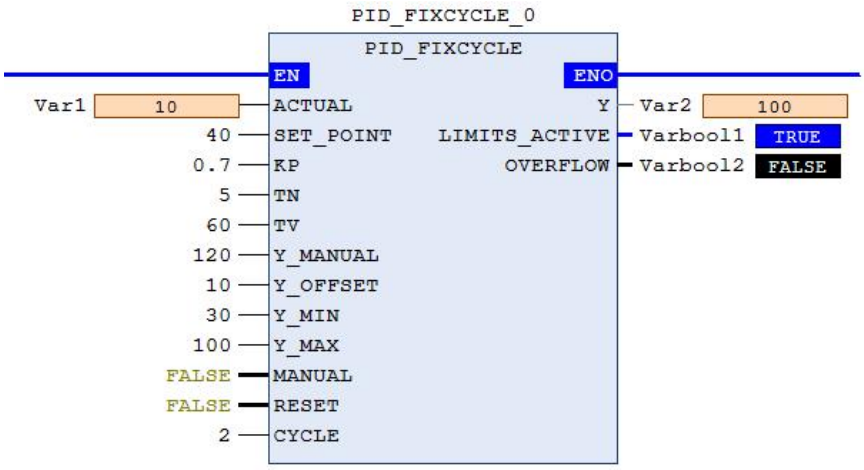
	RESET:= FALSE, Y=>Var2, LIMITS_ACTIVE=>Varbool1, OVERFLOW=>Varbool2);

3.16.3. PID_FIXCYCLE——比例积分微分控制器

该指令与前述的比例积分微分控制器功能基本相同，唯一的区别是，PID 调节周期是通过参数 CYCLE 传输进来的，该数据用来设定积分和微分的时间长短。

变量定义

	类别	名称	地址	数据类型	初值	注释	特性
1	VAR	**Var1**		REAL			
2	VAR	**Var2**		REAL			
3	VAR	**Varbool1**		BOOL			
4	VAR	**Varbool2**		BOOL			
5	VAR	**PID_FIXCYCLE_0**		PID_FIXCYCLE			

编程语言	程序
梯形图 （LD）	
结构化文本 （ST）	PID_FIXCYCLE_0(ACTUAL:=Var1, SET_POINT:=40, KP:= 0.7, TN:= 5, TV:=60, Y_MANUAL:=120, Y_OFFSET:=10, Y_MIN:=30, Y_MAX:= 100, MANUAL:=FALSE, RESET:=FALSE, CYCLE:=2, Y=>Var2, LIMITS_ACTIVE=>Varbool1, OVERFLOW=>Varbool2);

3.17. 数据异常处理指令（Util.compiled-library）

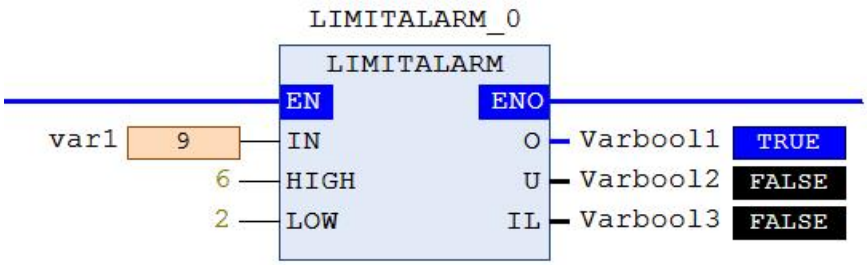
3.17.1. LIMITALARM——上下限报警

上下限报警指令 LIMITALARM 用于对输入数据的上下限值进行监视，若输入值超过上限值则发出上限报警信号，若输入值超过下限值则发出下限报警信号。

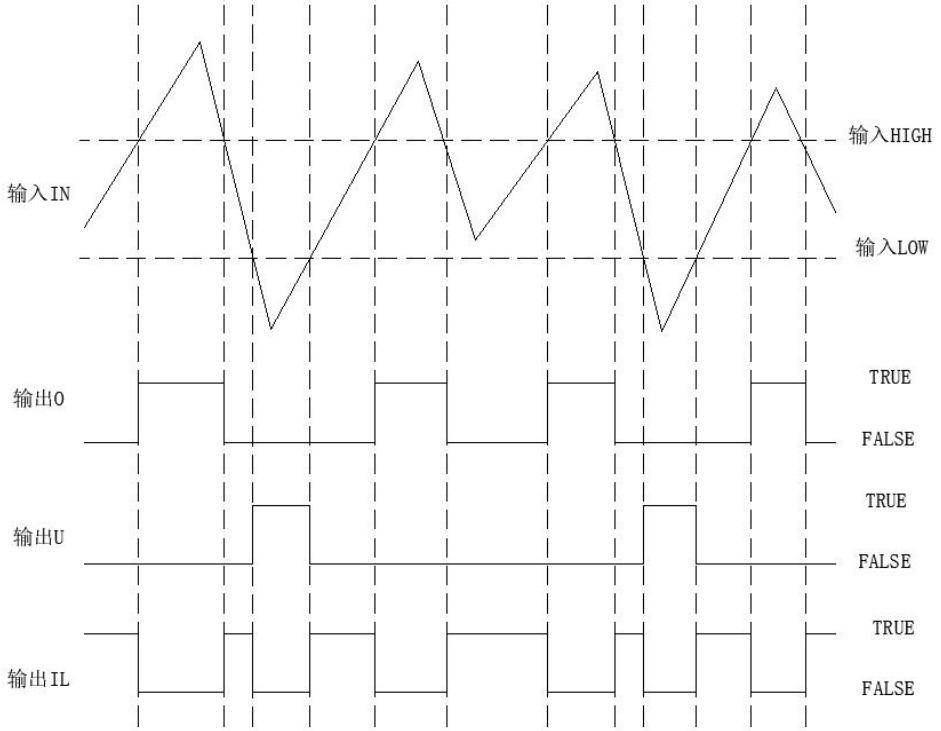
输入参数	数据类型	功能描述	参数值说明
IN	INT	输入值	
HIGH	INT	上限值	
LOW	INT	下限值	
输出参数	**数据类型**	**功能描述**	**参数值说明**
O	BOOL	输出值	输入超过上限时为 TRUE
U	BOOL	输出值	输入低于下限时为 TRUE
IL	BOOL	输出值	输入介于上下限之间时为 TRUE

变量定义

	类别	名称	地址	数据类型	初值	注释	特性
1	VAR	**Var1**		INT			
2	VAR	**Varbool1**		BOOL			
3	VAR	**Varbool2**		BOOL			
4	VAR	**Varbool3**		BOOL			
5	VAR	**LIMITALARM_0**		LIMITALARM			

编程语言	程序
梯形图（LD）	
结构化文本（ST）	LIMITALARM_0(IN:=var1, HIGH:=6, LOW:=2, O=>Varbool1, U=>Varbool2, IL=>Varbool3);

上述程序运行时，输入、输出对应关系如下图所示。



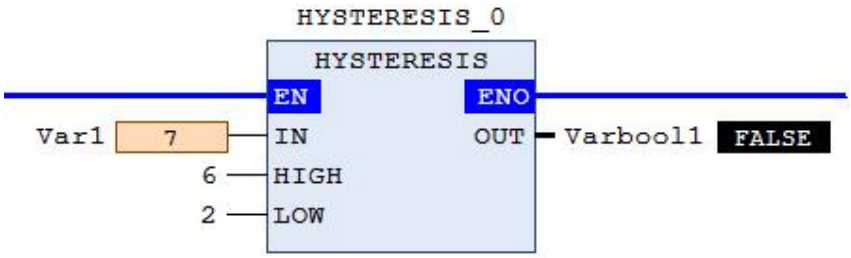
3.17.2. HYSTERESIS——滞后

滞后指令 HYSTERESIS 相当于在变量变化过程中设置了一个死区，当输入信号由低向高变化和由高向低变化时，输出状态发生变化的阈值不同。应用该指令后，一方面可以减少各类工业现场执行机构的动作频率，另一方面还能在一定程度上克服现场干扰信号的影响。

输入参数	数据类型	功能描述	参数值说明
IN	INT	输入值	
HIGH	INT	上限值	
LOW	INT	下限值	
输出参数	**数据类型**	**功能描述**	**参数值说明**
OUT	BOOL	输出值	输入低于下限值后为 TRUE，输入高于上限值后为 FALSE

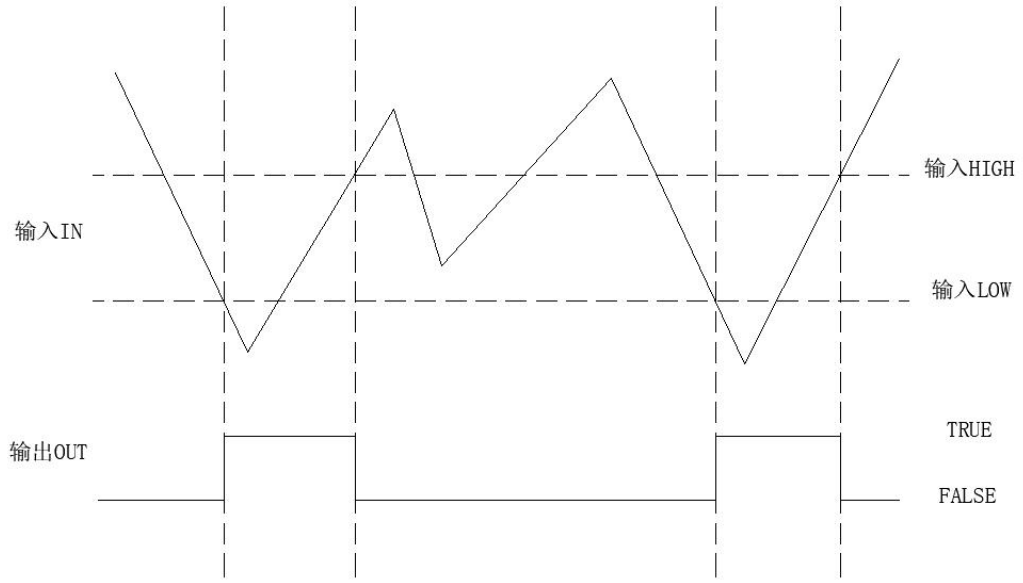
变量定义

	类别	名称	地址	数据类型	初值	注释	特性
1	VAR	**Var1**		INT			
2	VAR	**Varbool1**		BOOL			
3	VAR	**HYSTERESIS_0**		HYSTERESIS			

编程语言	程序
梯形图（LD）	

结构化文本（ST）	HYSTERESIS_0(IN :=Var1, HIGH:=6, LOW:=2, OUT=>Varbool1);

上述程序运行时，得到如下图所示结果。



3.18. 字符串处理指令（Standard. compiled-library）

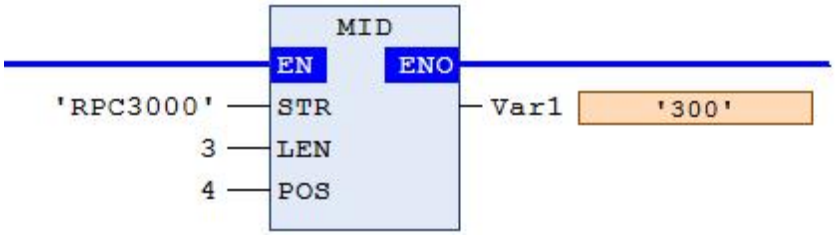
3.18.1. LEFT——左边取字符串指令

左边取字符串指令 LEFT 用于从字符串左边取一部分或全部字符串，该指令的语法格式为 LEFT(STR,SIZE)。

指令对应的输入和输出数据类型	
输入 STR 的数据类型为 STRING，输入 SIZE 的数据类型为 BYTE、INT、SINT、UINT、USINT、WORD。 输出的数据类型为 STRING。	
变量定义	
类别 名称 地址 数据类型 初值 注释 特性 1 VAR Var1 STRING	
编程语言	**程序**
梯形图（LD）	LEFT EN ENO 'RPC3000' STR 3 SIZE Var1 'RPC' (*结果为'RPC'*)
结构化文本（ST）	Var1:=LEFT('RPC3000',3); (*结果为'RPC'*)

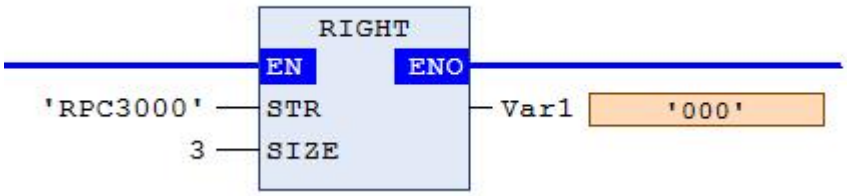
3.18.2. MID——中间取字符串指令

中间取字符串指令 MID 用于从字符串中间某一位置开始由左向右取一部分或全部字符串，该指令的语法格式为 MID(STR,LEN,POS)。

指令对应的输入和输出数据类型	
输入 STR 的数据类型为 STRING，输入 LEN 和 POS 的数据类型为 BYTE、INT、SINT、UINT、USINT、WORD。 输出的数据类型为 STRING。	
变量定义	
类别 名称 地址 数据类型 初值 注释 特性 1 VAR Var1 STRING	
编程语言	程序
梯形图（LD）	 (*结果为'300'*)
结构化文本（ST）	Var1:=MID('RPC3000',3,4); (*结果为'300'*)

3.18.3. RIGHT——右边取字符串指令

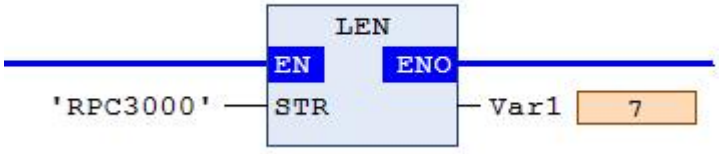
右边取字符串指令 RIGHT 用于从字符串右边取一部分或全部字符串，该指令的语法格式为 RIGHT(STR,SIZE)。

指令对应的输入和输出数据类型	
输入 STR 的数据类型为 STRING，输入 SIZE 的数据类型为 BYTE、INT、SINT、UINT、USINT、WORD。 输出的数据类型为 STRING。	
变量定义	
类别 名称 地址 数据类型 初值 注释 特性 1 VAR Var1 STRING	
编程语言	程序
梯形图（LD）	 (*结果为'000'*)

结构化文本（ST）	Var1:=RIGHT('RPC3000',3); (*结果为'000'*)

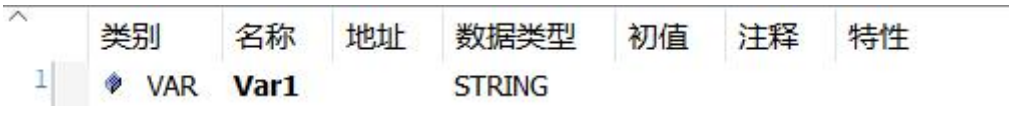
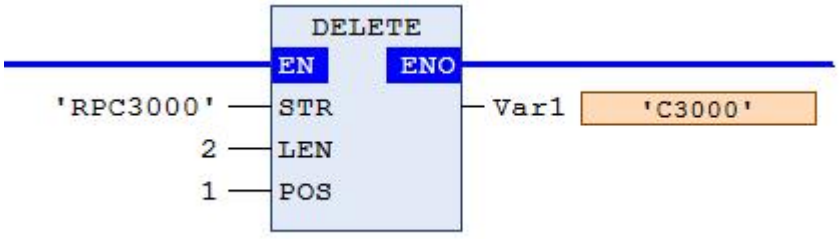
3.18.4. LEN——取字符串长度指令

取字符串长度指令 LEN 用于计算输入字符串的长度。

指令对应的输入和输出数据类型	
输入的数据类型为 STRING，输出的数据类型为 DINT、DWORD、INT、REAL、UDINT、UINT、USINT、WORD。	
变量定义	
	
编程语言	程序
梯形图（LD）	 (*结果为 7*)
结构化文本（ST）	Var1:=LEN('RPC3000'); (*结果为 7*)

3.18.5. DELETE——删除字符指令

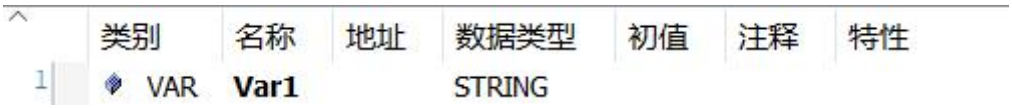
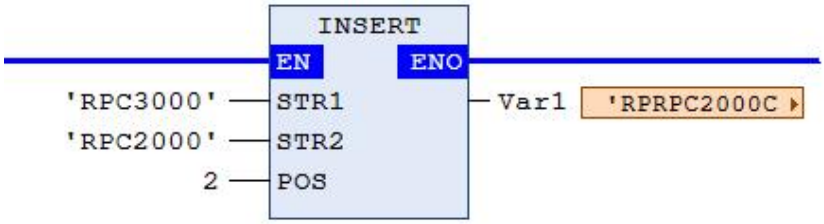
删除字符串指令 DELETE 用于从字符串的某一位置开始删除部分字符串或全部字符串，该指令格式为：DELETE(STR,LEN,POS)，其功能为从输入字符串从前往后数的位置 POS 处开始，连续删除 LEN 个字符。

指令对应的输入和输出数据类型	
输入 STR 的数据类型为 STRING，输入 LEN 和 POS 的数据类型为 BYTE、INT、SINT、UINT、USINT、WORD。 输出的数据类型为 STRING。	
变量定义	
	
编程语言	程序
梯形图（LD）	 (*结果为'C3000'*)

结构化文本（ST）	Var1:= DELETE('RPC3000',1,2); (*结果为' C3000'*)

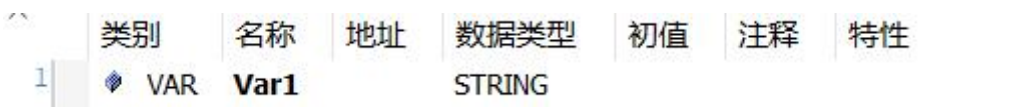
3.18.6. INSERT——插入字符串指令

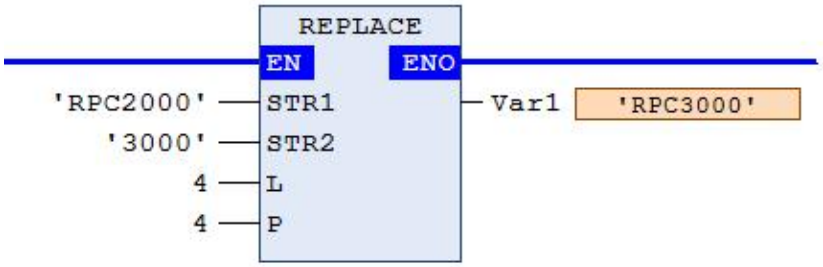
插入字符串指令用于把一个字符串插入到另一个字符串内，该指令的语法格式为：INSERT(STR1,STR2,POS)，其中 STR1 为需插入字符串的原始字符串，STR2 为插入的字符串，POS+1 为原始字符串从左向右开始计算的插入起始位置，即把 STR2 插入到 STR1 的 POS 位置之后。

指令对应的输入和输出数据类型	
输入 STR1 和 STR2 的数据类型为 STRING，输入 POS 的数据类型为 BYTE、INT、SINT、UINT、USINT、WORD。 输出的数据类型为 STRING。	
变量定义	
	
编程语言	**程序**
梯形图（LD）	 (*结果为'RPRPC2000C3000'*)
结构化文本（ST）	Var1:=INSERT (' RPC3000', ' RPC2000',2); (*结果为'RPRPC2000C3000'*)

3.18.7. REPLACE——替换字符串指令

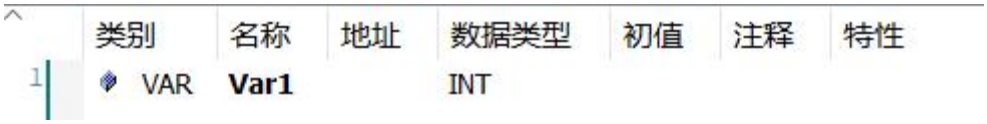
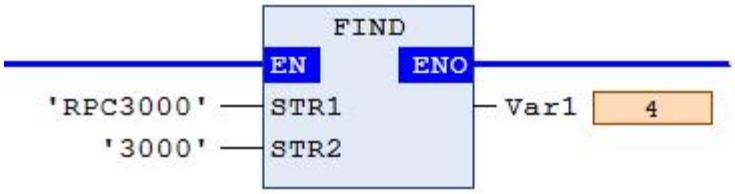
替换字符串指令 REPLACE 用于将一个字符串代替另一个字符串中的部分内容，该指令的语法格式为：REPLACE(STR1,STR2,L,P)，其功能为用字符串 STR2 代替字符串 STR1 中从位置 P 起始的 L 个字符。

指令对应的输入和输出数据类型	
输入 STR1 和 STR2 的数据类型为 STRING，输入 L 和 P 的数据类型为 BYTE、INT、SINT、UINT、USINT、WORD。 输出的数据类型为 STRING。	
变量定义	
	
编程语言	**程序**

梯形图 （LD）	 (*结果为'RPC3000'*)
结构化文本 （ST）	Var1:=REPLACE ('RPC2000','3000',4,4); (*结果为'RPC3000'*)

3.18.8. FIND——查找字符串指令

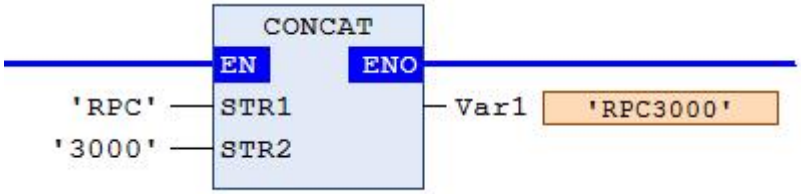
查找字符串指令 FIND 用于在一个字符串中查找既定内容，该内容由另一字符串确定，该指令的语法格式为：FIND(STR1,STR2)，输出结果为既定查找内容在第一个字符串中的起始位置。若第一个字符串中没有与第二个字符串完全相同的内容，输出结果为 0。

指令对应的输入和输出数据类型	
输入 STR1 和 STR2 的数据类型为 STRING。 输出的数据类型为 BYTE、INT、REAL、SINT、UINT、USINT、WORD。	
变量定义	
	
编程语言	**程序**
梯形图 （LD）	 (*结果为 4*)
结构化文本 （ST）	Var1:=FIND ('RPC3000', '3000'); (*结果为 4*)

3.18.9. CONCAT——合并字符串指令

合并字符串指令 CONCAT 用于把两个字符串合并成一个字符串。

指令对应的输入和输出数据类型
输入和输出的数据类型均为 STRING。
变量定义


编程语言	程序
梯形图 （LD）	 (*结果为'RPC3000'*)
结构化文本 （ST）	Var1:=CONCAT ('RPC', '3000'); (*结果为'RPC3000'*)

3.19. 计数器（Standard. compiled-library）

系统支持三类计数器，分别为递减计数器 CTD、递增计数器 CTU 和递增递减计数器 CTUD。

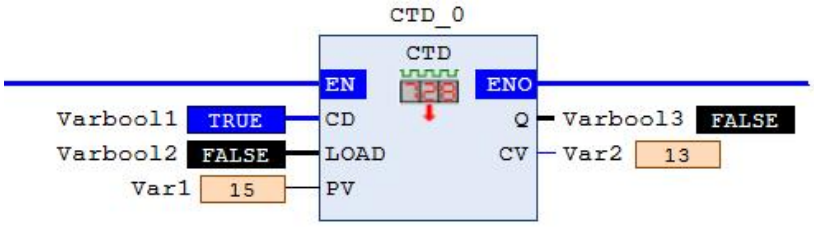
3.19.1. CTD——递减计数器

递减计数器指令 CTD 各参数的含义及相关说明见下表。

输入参数	数据类型	功能描述	参数值说明
CD	BOOL	计数输入	CV 大于 0 时，CD 每检测到 1 个上升沿 CV 减 1
LOAD	BOOL	初始化	为 TRUE 时，CV 被初始化为 PV
PV	WORD	计数器设定值	0-65535
输出参数	**数据类型**	**功能描述**	**参数值说明**
Q	BOOL	计数标志输出	当 CV 等于 0 时，Q 为 TRUE
CV	WORD	当前计数值	

变量定义

	类别	名称	地址	数据类型	初值	注释	特性
1	VAR	**Var1**		WORD			
2	VAR	**Var2**		WORD			
3	VAR	**Varbool1**		BOOL			
4	VAR	**Varbool2**		BOOL			
5	VAR	**Varbool3**		BOOL			
6	VAR	**CTD_0**		CTD			

编程语言	程序
梯形图 （LD）	
结构化文本	CTD_0(

（ST）	CD:=Varbool1, LOAD:=Varbool2, PV:=Var1, Q=>Varbool3, CV=>Var2);

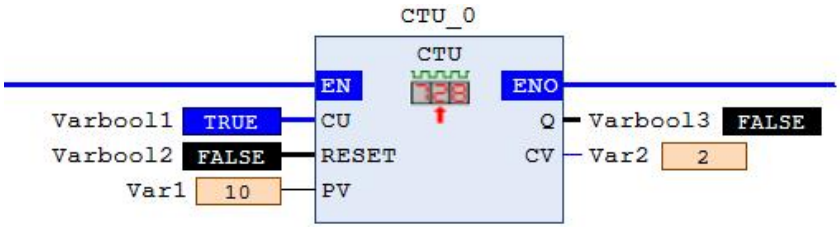
3.19.2. CTU——递增计数器

递增计数器指令 CTU 各参数的含义及相关说明见下表。

输入参数	数据类型	功能描述	参数值说明
CU	BOOL	计数输入	CU 每检测到 1 个上升沿，CV 加 1
RESET	BOOL	初始化	为 TRUE 时 CTU 被重新初始化
PV	WORD	计数器设定值	0-65535
输出参数	**数据类型**	**功能描述**	**参数值说明**
Q	BOOL	计数标志输出	CV 大于或等于 PV 时，Q 为 TRUE
CV	WORD	当前计数值	

变量定义

	类别	名称	地址	数据类型	初值	注释	特性
1	VAR	Var1		WORD			
2	VAR	Var2		WORD			
3	VAR	Varbool1		BOOL			
4	VAR	Varbool2		BOOL			
5	VAR	Varbool3		BOOL			
6	VAR	CTU_0		CTU			

编程语言	程序
梯形图（LD）	
结构化文本（ST）	CTU_0(CU:=Varbool1, LOAD:=Varbool2, PV:=Var1, Q=>Varbool3, CV=>Var2);

3.19.3. CTUD——递增递减计数器

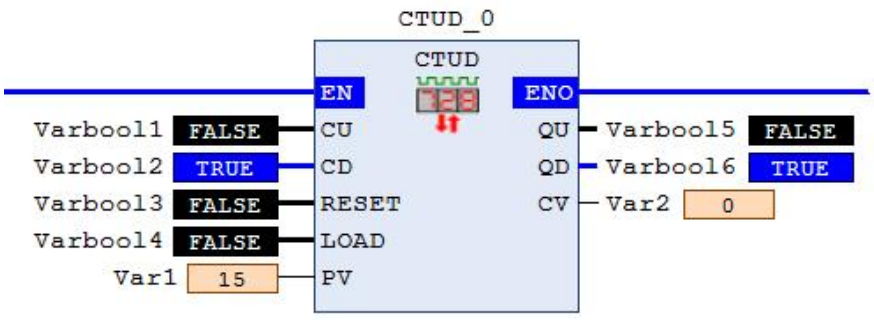
递增递减计数器指令 CTUD 各参数的含义及相关说明见下表。

输入参数	数据类型	功能描述	参数值说明
CU	BOOL	计数输入	CU 每检测到 1 个上升沿，CV 加 1

CD	BOOL	计数输入	CV 大于 0 时，CD 每检测到 1 个上升沿，CV 减 1
RESET	BOOL	复位输入	为 TRUE 时，CV 初始化为 0
LOAD	BOOL	初始化	为 TRUE 时，CV 被初始化为 PV
PV	WORD	计数器设定值	0-65535
输出参数	**数据类型**	**功能描述**	**参数值说明**
QU	BOOL	计数标志输出	当 CV 等于 PV 时，QU 为 TRUE
QD	BOOL	计数标志输出	当 CV 等于 0 时，QD 为 TRUE
CV	WORD	当前计数值	

变量定义

	类别	名称	地址	数据类型	初值	注释	特性
1	VAR	**Var1**		WORD			
2	VAR	**Var2**		WORD			
3	VAR	**Varbool1**		BOOL			
4	VAR	**Varbool2**		BOOL			
5	VAR	**Varbool3**		BOOL			
6	VAR	**Varbool4**		BOOL			
7	VAR	**Varbool5**		BOOL			
8	VAR	**Varbool6**		BOOL			
9	VAR	**CTUD_0**		CTUD			

编程语言	程序
梯形图（LD）	
结构化文本（ST）	CTUD_0(CU:=Varbool1, CD:=Varbool2, RESET:=Varbool3, LOAD:= Varbool4, PV:=Var1, QU=>Varbool5, QD=>Varbool6, CV=>Var2);

3.20. 定时器（Standard. compiled-library）

在软件中，有实时时钟指令 RTC，以及普通定时器 TP、通电延时定时器 TON、断电延时定时器 TOF 三类定时器指令。

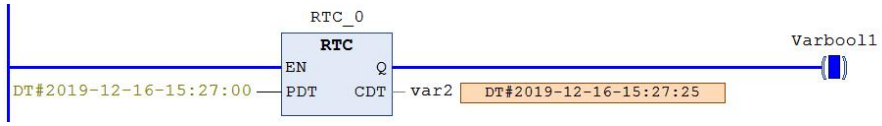
3.20.1. RTC——实时时钟

实时时钟指令 RTC 各参数的含义及相关说明见下表。

输入参数	数据类型	功能描述	参数值说明
EN	BOOL	使能信号	
PDT	DT	时间基值	
输出参数	**数据类型**	**功能描述**	**参数值说明**
Q	BOOL	定时器输出	EN 为“0”时，Q 为“0”；EN 为“1”时，Q 为“1”。
CDT	DT	当前时间值	EN 为“0”时，CDT 为 1970-01-01-00:00:00；EN 为“1”时，CDT 从 PDT 提供的时间开始计时。

变量定义

	类别	名称	地址	数据类型	初值	注释	特性
1	VAR	Var1		DT			
2	VAR	Varbool1		bool			
3	VAR	RTC_0		RTC			

编程语言	程序
梯形图（LD）	
结构化文本（ST）	RTC_0(EN:=TRUE, PDT:=DT#2019-12-16-15:27:00, Q=>Varbool1, CDT=>Var1);

提示：

1. **EN 为 TRUE (上升沿)，PDT 将接收时间给定值，并以秒为单位开始计时。**
2. **EN 为 TRUE 时，实时变化的计时值将传递给 CDT。EN 被复位为 FALSE 时，CDT 将被复位为系统初始值。**

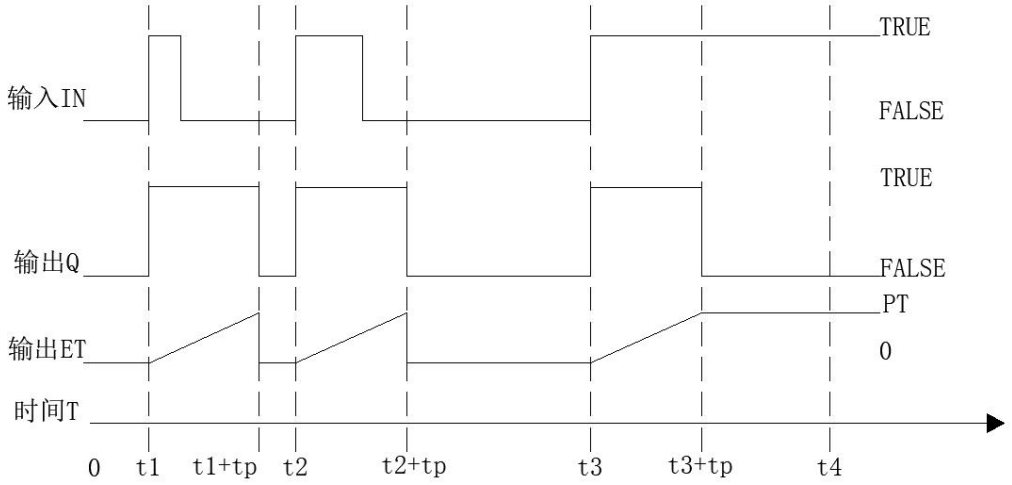
3.20.2. TP——普通定时器

普通定时器指令 TP 各参数的含义及相关说明见下表。

输入参数	数据类型	功能描述	参数值说明
IN	BOOL	定时器初始化信号	当 IN 由“0”变为“1”时，ET 以 ms 为单位计时，直至 ET 等于 PT 后 ET 保持不变
PT	TIME	定时时间值	

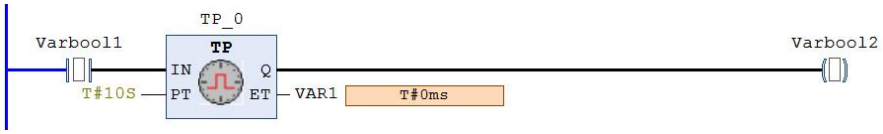
输出参数	数据类型	功能描述	参数值说明
Q	BOOL	定时器输出	IN 为“0”时，Q 为“0”，ET 为 0。IN 为“1”时，TP 开始计时，Q 为“1”。开始计时后，在 ET 小于等于 PT 前，IN 变化对计时过程无影响。ET 计时至等于 PT 时，Q 变为“0”。计时结束后，当 IN 变为“0”，ET 等于 0。
ET	TIME	当前时间值	

普通定时器 TP 的输入、输出关系如下图所示。



变量定义

	类别	名称	地址	数据类型	初值	注释	特性
1	VAR	Var1		TIME			
2	VAR	Varbool1		BOOL			
3	VAR	Varbool2		BOOL			
4	VAR	TP_0		TP			

编程语言	程序
梯形图（LD）	
结构化文本（ST）	TP_0(IN:=Varbool1, PT:=T#10s, Q=>Varbool2, ET=>Var1);

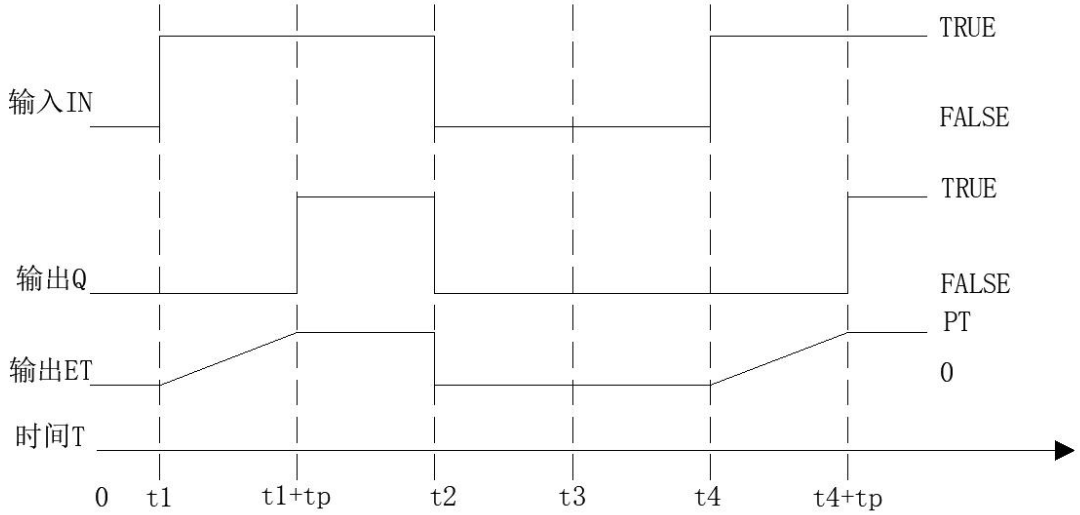
3.20.3. TON——通电延时定时器

通电延时定时器指令 TON 各参数的含义及相关说明见下表。

输入参数	数据类型	功能描述	参数值说明
IN	BOOL	定时器初始化信号	当 IN 由“0”变为“1”时，ET 以 ms 为单位计时，直至 ET 等于 PT 后 ET 保持不变。

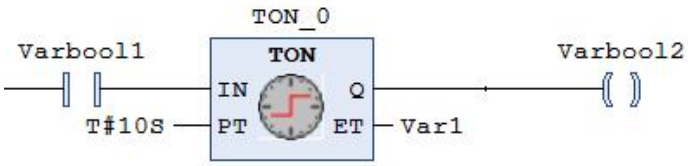
PT	TIME	定时时间值	
输出参数	**数据类型**	**功能描述**	**参数值说明**
Q	BOOL	定时器输出	IN 为“0”时，Q 为“0”，ET 为 0。IN 为“1”时，TP 开始计时，ET 等于 PT 后 Q 为“1”。开始计时后，在 ET 等于 PT 前，若 IN 由“1”变为“0”，则计时过程终止，ET 等于 0。
ET	TIME	当前时间值	

通电延时定时器 TON 的输入、输出关系如下图所示。



变量定义

	类别	名称	地址	数据类型	初值	注释	特性
1	VAR	**Var1**		TIME			
2	VAR	**Varbool1**		BOOL			
3	VAR	**Varbool2**		BOOL			
4	VAR	**TON_0**		TON			

编程语言	**程序**
梯形图（LD）	 (*Varbool1 触点为 TRUE，延时 10s 后 varbool2 输出为 TRUE*)
结构化文本（ST）	TON_0(IN:=Varbool1, PT:=T#10s, Q=>Varbool2, ET=>Var1);

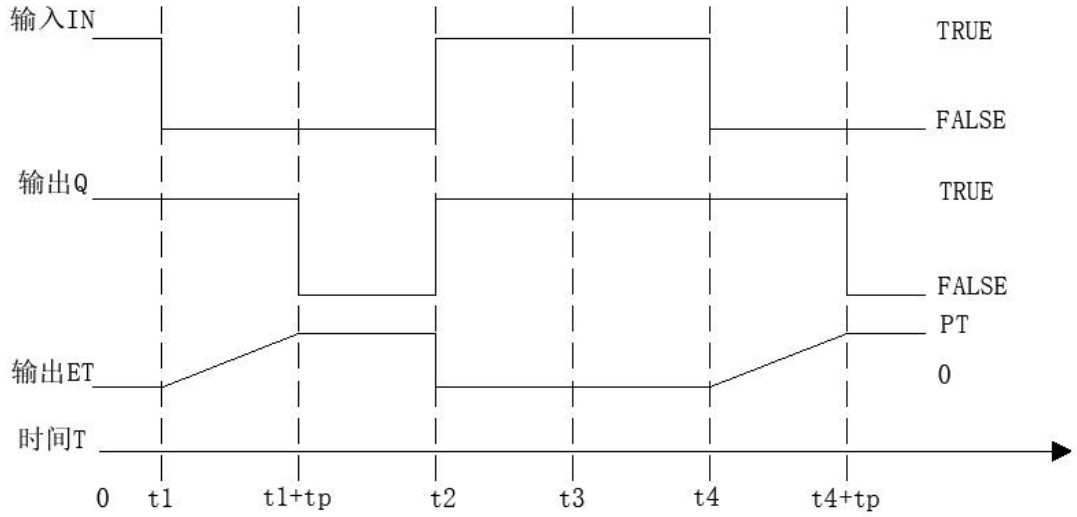
提示： 如果 IN 为 FALSE，Q 将会为 FALSE，且 ET 为 0。一旦 IN 为 TRUE，在 ET 端将会以毫秒为单位进行计时，直到 ET 等于 PT 值，其后将保持该常量值。当 IN 为 TRUE 并且 ET 等于 PT 时，Q 为 TRUE，否则 Q 为 FALSE。因此，以毫秒为单位计时，达到 PT 的输入值后，Q 会产生一个上升沿。

3.20.4. TOF——断电延时定时器

断电延时定时器指令 TOF 各参数的含义及相关说明见下表。

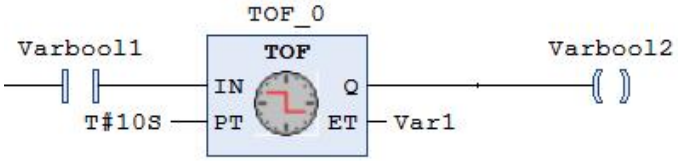
输入参数	数据类型	功能描述	参数值说明
IN	BOOL	定时器初始化信号	当 IN 由“1”变为“0”时，ET 以 ms 为单位计时，直至 ET 等于 PT 后 ET 保持不变。
PT	TIME	定时时间值	
输出参数	**数据类型**	**功能描述**	**参数值说明**
Q	BOOL	定时器输出	IN 为“1”时，Q 为“1”，ET 为 0。IN 为“0”时，TP 开始计时，ET 等于 PT 后 Q 为“0”。开始计时后，在 ET 等于 PT 前若 IN 由“0”变为“1”，则计时过程终止，ET 等于 0。
ET	TIME	当前时间值	

断电延时定时器 TOF 的输入、输出关系如下图所示。



变量定义

	类别	名称	地址	数据类型	初值	注释	特性
1	VAR	Var1		TIME			
2	VAR	Varbool1		BOOL			
3	VAR	Varbool2		BOOL			
4	VAR	TOF_0		TOF			

编程语言	程序
梯形图（LD）	 (* Varbool1 由“TRUE”变为“FALSE”，延时 10s 后 varbool2 断开*)
结构化文本（ST）	TOF_0(IN:=Varbool1, PT:=T#10s,

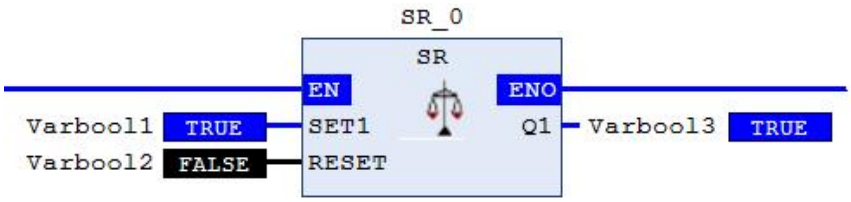
	Q=>Varbool2, ET=>Var1);

3.21. 双稳态指令（Standard. compiled-library）

3.21.1. SR——置位优先双稳态触发器

置位优先双稳态触发器指令 SR 的逻辑关系为：Q1:= SET1 OR (NOT RESET AND Q1)，其中 SET1 为置位信号，RESET 为复位信号，Q1 为输出信号。SR 触发器的真值表如下所示：

SET1	RESET	输出
0	0	保持原状态
1	0	1
0	1	0
1	1	1

指令对应的输入和输出数据类型	
输入和输出的数据类型均为 BOOL 型。	
变量定义	
	
编程语言	**程序**
梯形图 （LD）	
结构化文本 （ST）	SR_0(SET1:=Varbool1, RESET:=Varbool2, Q1=>Varbool3);

3.21.2. RS——复位优先双稳态触发器

复位优先双稳态触发器指令 RS 的逻辑关系为：Q1:= NOT RESET1 AND (Q1 OR SET)，其中 SET1 为置位信号，RESET 为复位信号，Q1 为输出信号。RS 触发器的真值表如下所示：

SET	RESET1	输出
0	0	保持原状态
1	0	1

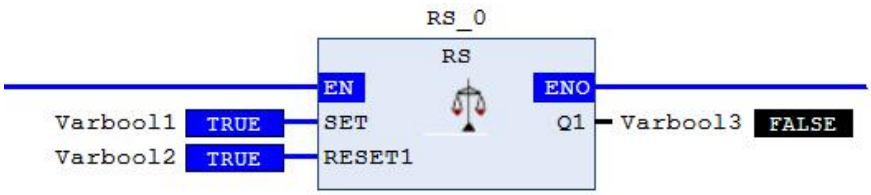
0	1	0
1	1	0

指令对应的输入和输出数据类型

输入和输出的数据类型为 BOOL 型。

变量定义

	类别	名称	地址	数据类型	初值	注释	特性
1	VAR	**Varbool1**		BOOL			
2	VAR	**Varbool2**		BOOL			
3	VAR	**Varbool3**		BOOL			
4	VAR	**RS_0**		RS			

编程语言	程序
梯形图（LD）	
结构化文本（ST）	RS_0(SET:=Varbool1, RESET1:=Varbool2, Q1=>Varbool3);

3.22. 触发器指令（Standard. compiled-library）

触发器分为两类，下降沿检测触发器 F_TRIG 用于检测下降沿，上升沿检测触发器 R_TRIG 用于检测上升沿。

3.22.1. F_TRIG——下降沿检测触发器

下降沿检测触发器 F_TRIG 检测到输入信号的下降沿时，将向输出端发送触发信号。该指令的语法格式为：

Q := NOT CLK AND NOT M;

M := NOT CLK;

在该指令的编程语言中看不到 M，M 是 BOOL 型中间变量，其初始值为“0”。当 CLK 是由“1”变为“0”时，Q 维持一个周期的高电平，下一周期 Q 又被 M 置为低电平。

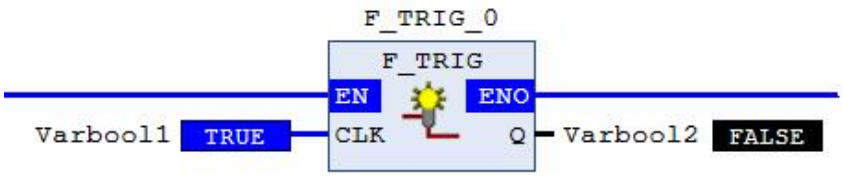
指令对应的输入和输出数据类型

输入和输出的数据类型为 BOOL 型。

变量定义

	类别	名称	地址	数据类型	初值	注释	特性
1	VAR	**Varbool1**		BOOL			
2	VAR	**Varbool2**		BOOL			
3	VAR	**F_TRIG_0**		F_TRIG			

编程语言	程序

梯形图（LD）	
结构化文本（ST）	F_TRIG_0(CLK:=Varbool1, Q=>Varbool2);

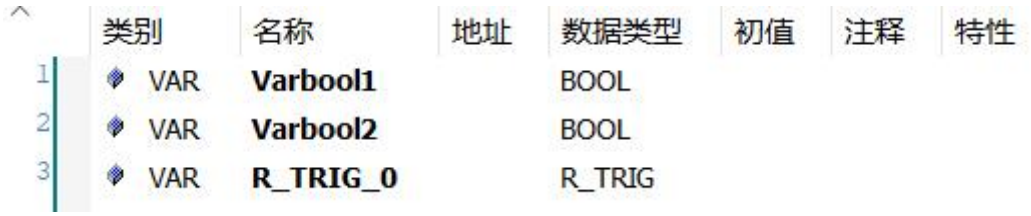
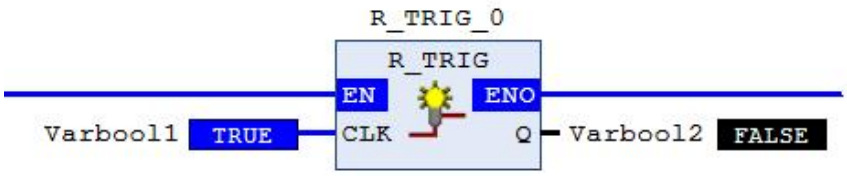
3.22.2. R_TRIG——上升沿检测触发器

上升沿检测触发器 R_TRIG 检测到输入信号的上升沿时，将向输出端发送触发信号。该指令的语法格式为：

Q :=CLK AND NOT M;

M :=CLK;

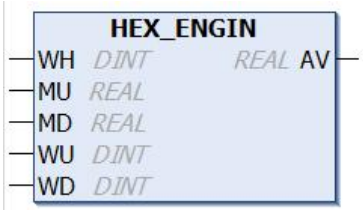
在该指令的编程语言中看不到 M，M 是 BOOL 型中间变量，其初始值为“1”。当 CLK 是由“0”变为“1”时，Q 维持一个周期的高电平，下一周期 Q 又被 M 置为低电平。

指令对应的输入和输出数据类型	
输入和输出的数据类型为 BOOL 型。	
变量定义	
	
编程语言	**程序**
梯形图（LD）	
结构化文本（ST）	R_TRIG_0(CLK:=Varbool1, Q=>Varbool2);

3.23. 数学指令（RPCMath. compiled-library）

3.23.1. HEX_ENGIN——模拟量输入数据转换为工程量数据

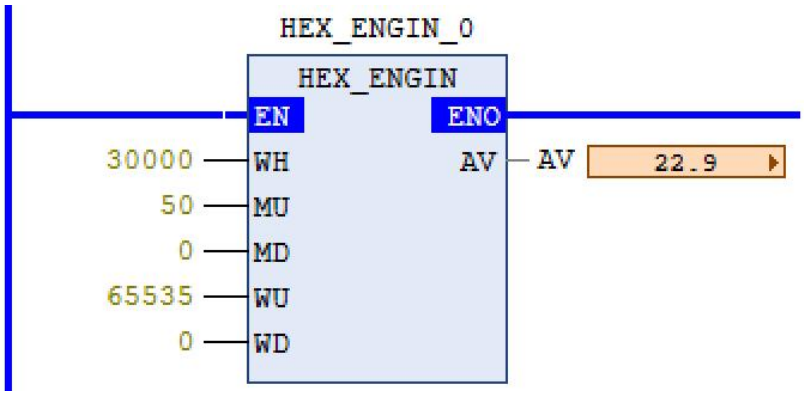
模拟量输入数据转换为工程量数据指令 HEX_ENGIN 在库文件中的图示表达与相关参数说明如下：



输入参数	数据类型	功能描述	参数值说明
WH	DINT	模拟量输入码值	
MU	REAL	工程量上限值	
MD	REAL	工程量下限值	
WU	DINT	模拟量输入码值上限	RPC3000 系列 PLC 模拟量的码值范围：0~65535
WD	DINT	模拟量输入码值下限	
输出参数	**数据类型**	**功能描述**	**参数值说明**
AV	REAL	工程量输出值	

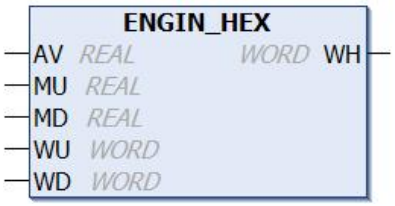
变量定义

	类别	名称	地址	数据类型	初值	注释	特性
1	VAR	**HEX_ENGIN_0**		HEX_ENGIN			
2	VAR	**AV**		REAL			

编程语言	程序
梯形图（LD）	
结构化文本（ST）	HEX_ENGIN_0(WH:=30000, MU:=50, MD:=0, WU:=65535, WD:=0, AV=>AV);

3.23.2. ENGIN_HEX——工程量数据转换为模拟量输出数据

工程量数据转换为模拟量输出数据指令 ENGIN_HEX 在库文件中的图示表达与相关参数说明如下：

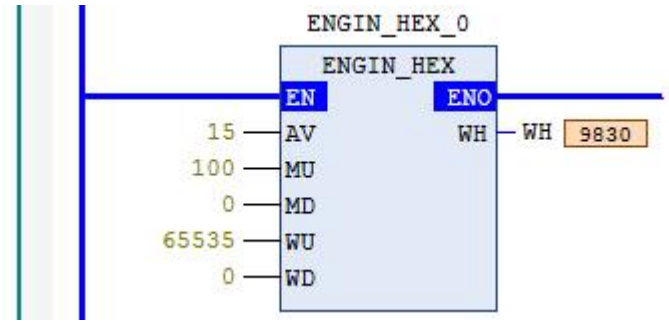


输入参数	数据类型	功能描述	参数值说明
AV	REAL	工程量输入值	

MU	REAL	工程量上限值	
MD	REAL	工程量下限值	
WU	WORD	模拟量输入码值上限	RPC3000 系列 PLC 模拟量的码值范围：0~65535
WD	WORD	模拟量输入码值下限	
输出参数	**数据类型**	**功能描述**	**参数值说明**
WH	WORD	模拟量输出码值	

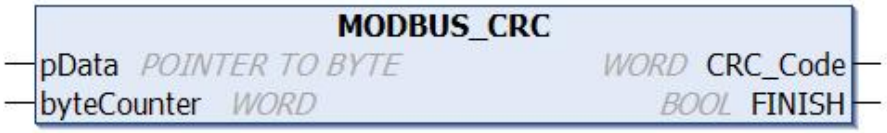
变量定义

	类别	名称	地址	数据类型	初值	注释	特性
1	VAR	**ENGIN_HEX_0**		ENGIN_HEX			
2	VAR	**WH**		WORD			

编程语言	**程序**
梯形图（LD）	
结构化文本（ST）	ENGIN_HEX_ 0(AV:=15, WU:=100, WD:=0, MU:=65535, MD:=0, WH=>WH);

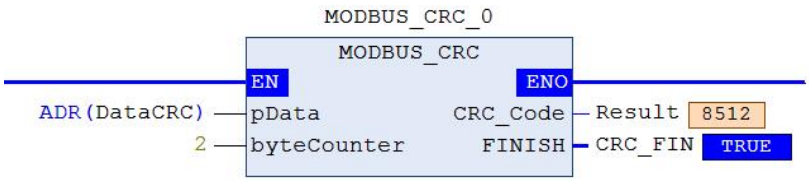
3.23.3. Modbus_CRC——生成 Modbus CRC 校验码

生成 Modbus CRC 校验码指令 Modbus_CRC 在库文件中的图示表达与相关参数说明如下：



输入参数	**数据类型**	**功能描述**	**参数值说明**
pData	POINT TO BYTE	指向校验数据的指针	
byteCounter	WORD	待校验数据的数量	以字节为计数单位
输出参数	**数据类型**	**功能描述**	**参数值说明**
CRC_Code	WORD	CRC 校验结果	
FINISH	BOOL	校验完成标志	0：未完成校验 1：完成校验

变量定义						
	类别	名称	地址	数据类型	初值	注释	特性
1	VAR	**MODBUS_CRC_0**		MODBUS_CRC			
2	VAR	**DataCRC**		ARRAY[1..2] OF BYTE	[16#1,16#3]		
3	VAR	**Result**		WORD	2		
4	VAR	**CRC_FIN**		BOOL			

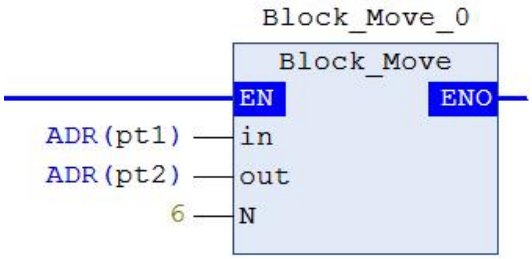
编程语言	程序
梯形图（LD）	
结构化文本（ST）	MODBUS_CRC_ 0(pData:=ADR(DataCRC), byteCounter:=2, CRC_Code =>Result， FINISH =>CRC_FIN);

3.23.4. BLOCK_MOVE——数据传送指令

数据传送指令 Block_Move 用于传输存储空间内连续存储的一批数据，该指令在库文件中的图示表达与相关参数说明如下：

输入参数	数据类型	功能描述	参数值说明
in	POINT TO BYTE	传输数据的起始地址	
out	POINT TO BYTE	接收数据的起始地址	
N	WORD	传输数据的字节长度	

变量定义						
	类别	名称	地址	数据类型	初值	注释	特性
1	VAR	**Block_Move_0**		Block_Move			
2	VAR	**pt1**		ARRAY[1..10] OF BYTE			
3	VAR	**pt2**		ARRAY[1..10] OF BYTE			

编程语言	程序
梯形图（LD）	

结构化文本（ST）	BLOCK_MOVE_ 0(in:=ADR(pt1), out:= ADR(pt2), N:= 6);

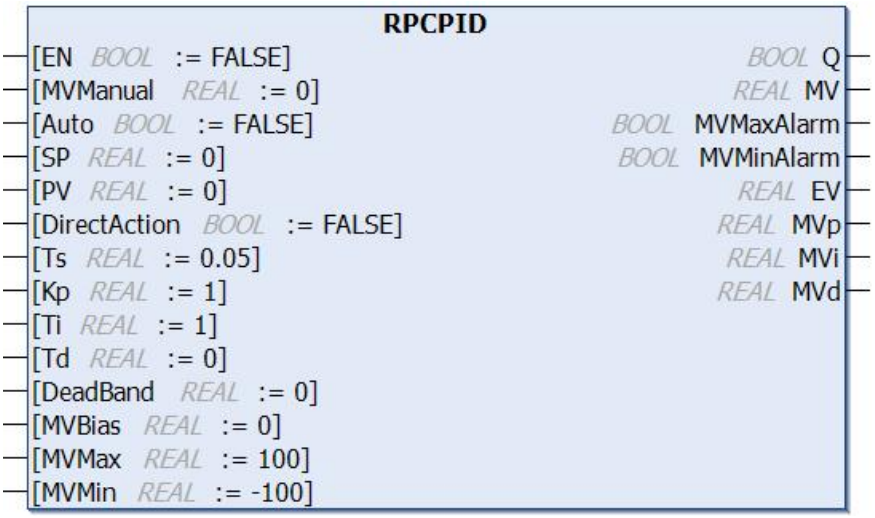
3.23.5. RPCPID——RPC 比例积分微分控制器

RPC 比例积分微分控制器 RPCPID 指令用于计算输入偏差信号的比例、积分和微分值，用于对生产过程进行稳定、高效的控制调节。当积分时间足够大时，PID 调节变为 PD 调节；当微分时间为 0 时，PID 调节变为 PI 调节。PID 控制器的传递函数为：

$$\frac{U(s)}{E(s)} = Kp\left(1+\frac{1}{Ti*s}+\frac{Td*s}{1+Td*s}\right)$$

其中 E（S）为输入偏差值，由过程值和设定值相减后得到，RPCPID 为反作用时由设定值减过程值，RPCPID 为正作用时由过程值减设定值。U（S）为 RPCPID 的调节输出，可在此基础上增加偏置值。Kp 为比例系数，Ti 为积分时间，Td 为微分时间。

该指令在库文件中的图示表达与相关参数说明如下：



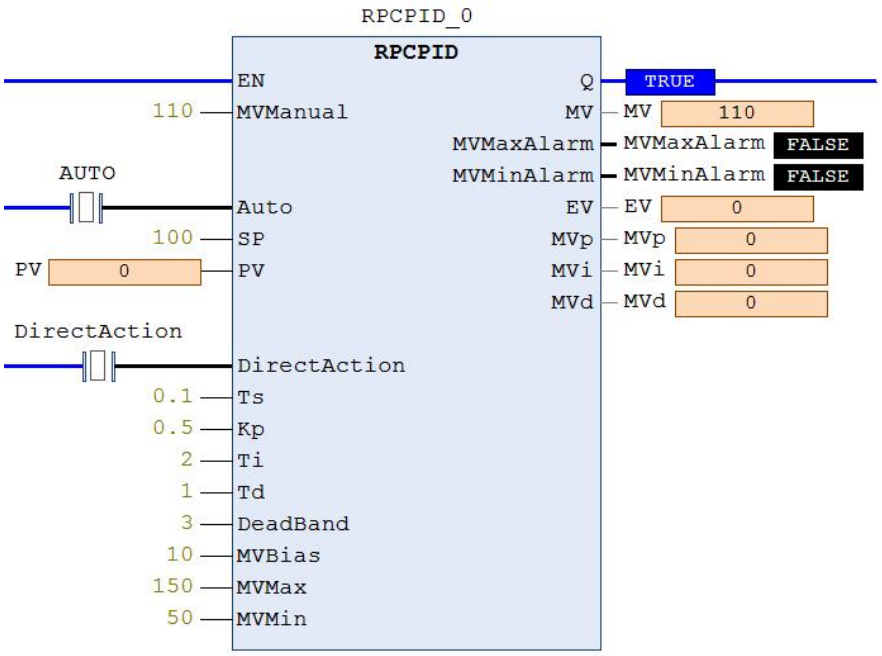
输入参数	数据类型	功能描述	参数值说明
EN	BOOL	使能控制端	0：无效 1：有效
MVManual	REAL	手动输出值	
Auto	BOOL	自动模式选择	0：手动 1：自动
SP	REAL	设定值	
PV	REAL	过程值	
DirectAction	BOOL	作用方式选择	0：反作用 1：正作用
Ts	REAL	运算周期	单位：秒
Kp	REAL	比例增益	

Ti	REAL	积分时间	单位：秒
Td	REAL	微分时间	单位：秒
DeadBand	REAL	偏差死区限值	
MVBias	REAL	前馈控制量	
MVMax	REAL	输出值上限	
MVMin	REAL	输出值下限	
输出参数	**数据类型**	**功能描述**	**参数值说明**
Q	BOOL	使能状态标志	0：无效 1：有效
MV	REAL	控制结果输出	
MVMaxAlarm	BOOL	上限报警	0：不超限 1：超限
MVMinAlarm	BOOL	下限报警	0：不超限 1：超限
EV	REAL	过程值与设定值的偏差	
MVp	REAL	比例分量	
MVi	REAL	积分分量	
MVd	REAL	微分分量	

变量定义

	类别	名称	地址	数据类型	初值	注释	特性
1	VAR	**RPCPID_0**		RPCPID			
2	VAR	**AUTO**		BOOL			
3	VAR	**PV**		REAL			
4	VAR	**DirectAction**		BOOL			
5	VAR	**MV**		REAL			
6	VAR	**MVMaxAlarm**		BOOL			
7	VAR	**MVMinAlarm**		BOOL			
8	VAR	**EV**		REAL			
9	VAR	**MVp**		REAL			
10	VAR	**MVi**		REAL			
11	VAR	**MVd**		REAL			

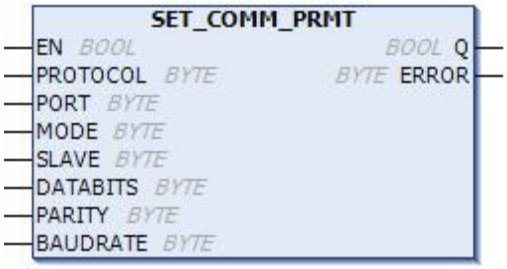
编程语言	**程序**

梯形图 （LD）	
结构化文本 （ST）	RPCPID_0(EN:=TRUE, MVManual:=110, Auto:= Auto, SP:=100, PV:= PV, DirectAction:= DirectAction, Ts:=0.1, Kp:=0.5, Ti:=2, Td:=1, DeadBand:=3, MVBias:=10, MVMax:=150, MVMin:=50, Q=> MV=> MV, MVMaxAlarm=> MVMaxAlarm, MVMinAlarm=> MVMinAlarm, EV=> EV, MVp=> MVp, MVi=> MVi, MVd=> MVd);

3.24. 串口通讯指令（CmpRPC3000-library）

3.24.1. SET_COMM_PRMT——设置串口通讯参数

设置串口通讯参数指令 SET_COMM_PRMT 在库文件中的图示表达和相关参数说明如下：

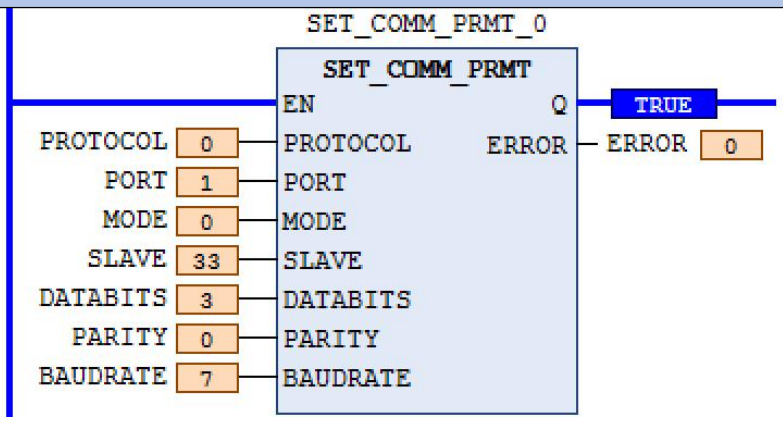


输入参数	数据类型	功能描述	参数值说明
EN	BOOL	使能	0：无效 1：上升沿使能 （上升沿触发，设置完成后不用重新上电）
PROTOCOL	BYTE	协议类型	0：Modbus-RTU 1：自由口
PORT	BYTE	RS232/RS485 串口号	0：串口号 0（RS485-0） 1：串口号 1（RS485-1） 2：串口号 2（RS485-2）
MODE	BYTE	Modbus-RTU 模式选择	0：从站 1：主站
SLAVE	BYTE	Modbus-RTU 模式的从站地址	1-247
DATABITS	BYTE	数据位	3，不允许设置为其他值（3 表示 8 位数据位）
PARITY	BYTE	校验位	0：无校验 1：偶校验 2：奇校验
BAUDRATE	BYTE	波特率	0：1200bps 1：2400bps 2：4800bps 3：9600bps 4：19200bps 5：38400bps 6：57600bps 7：115200bps
输出参数	**数据类型**	**功能描述**	**参数值说明**
Q	BOOL	操作结果	0：未完成 1：完成 （使能 EN 为 1，操作结果 Q 为 1，错误代码 ERROR 为 0，则串口通讯参数设置完成）
ERROR	BYTE	错误代码	0：无错误 Bit0：无效奇偶校验位

			Bit1：无效波特率 Bit3：无效从站地址 （字节中的 Bit0~Bit3 分别代表不同故障，每个故障显示互相独立。

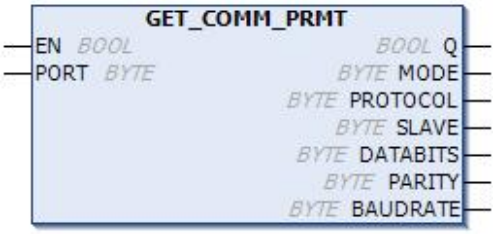
变量定义

	类别	名称	地址	数据类型	初值	注释
1	VAR	SET_COMM_PRMT_0		SET_COMM_PRMT		
2	VAR	PROTOCOL		BYTE	0	
3	VAR	PORT		BYTE	1	
4	VAR	MODE		BYTE	0	
5	VAR	SLAVE		BYTE	33	
6	VAR	DATABITS		BYTE	3	
7	VAR	PARITY		BYTE	0	
8	VAR	BAUDRATE		BYTE	7	
9	VAR	ERROR		BYTE		

编程语言	程序
梯形图（LD）	
结构化文本（ST）	SET_COMM_PRMT_0(EN:=TRUE , PROTOCOL:= PROTOCOL, PORT:= PORT, MODE:= MODE, SLAVE:= SLAVE, DATABITS:= DATABITS, PARITY:= PARITY, BAUDRATE:= BAUDRATE, Q=> , ERROR=>ERROR);　(*设置 RS485 接口 1 为 Modbus-RTU 从站，从站地址 33，数据位 8 位，校验位 None，波特率 115200*)

3.24.2. GET_COMM_PRMT——读取串口通讯参数

读取串口通讯参数指令 GET_COMM_PRMT 在库文件中的图示表达和相关参数说明如下：

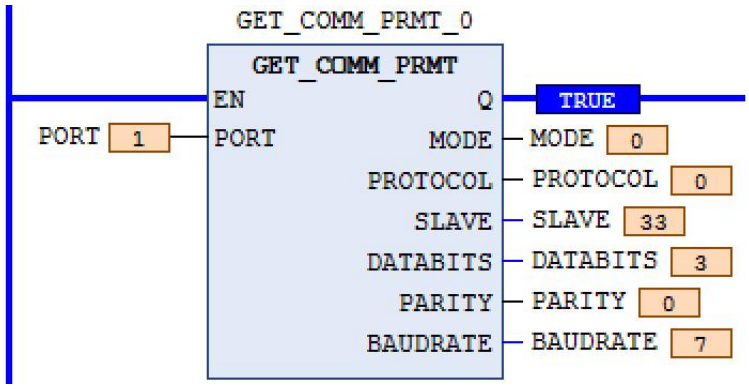


输入参数	数据类型	功能描述	参数值说明
EN	BOOL	使能	0：无效 1：高电平有效
PORT	BYTE	RS232/RS485 串口号	0：串口号 0（RS485-0） 1：串口号 1（RS485-1） 2：串口号 2（RS485-2）
输出参数	**数据类型**	**功能描述**	**参数值说明**
Q	BOOL	操作结果	0：未完成 1：完成
MODE	BYTE	Modbus-RTU 模式选择	0：从站 1：主站
PROTOCOL	BYTE	协议类型	0：Modbus-RTU 1：自由口
SLAVE	BYTE	Modbus-RTU 模式的从站地址	1-247
DATABITS	BYTE	数据位	3：表示 8 位数据位
PARITY	BYTE	校验位	0：无校验 1：偶校验 2：奇校验
BAUDRATE	BYTE	波特率	0：1200bps 1：2400bps 2：4800bps 3：9600bps 4：19200bps 5：38400bps 6：57600bps 7：115200bps

变量定义

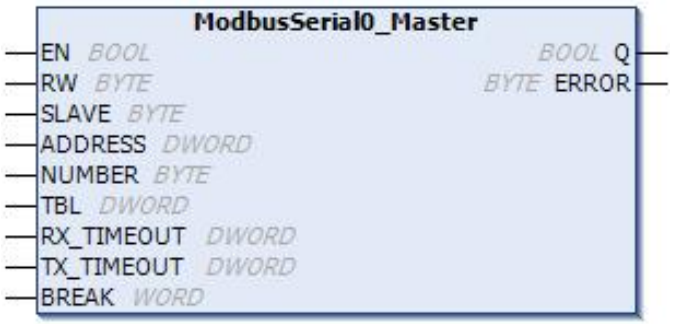
	类别	名称	地址	数据类型	初值	注释	属性
1	VAR	GET_COMM_PRMT_0		GET_COMM_PRMT			
2	VAR	PORT		BYTE	1		
3	VAR	MODE		BYTE			
4	VAR	PROTOCOL		BYTE			
5	VAR	SLAVE		BYTE			
6	VAR	DATABITS		BYTE			
7	VAR	PARITY		BYTE			
8	VAR	BAUDRATE		BYTE			

编程语言	程序

梯形图 （LD）	
结构化文本 （ST）	GET_COMM_PRMT_0(EN:= TRUE, PORT:= PORT, Q=> , MODE=> MODE, PROTOCOL=> PROTOCOL, SLAVE=> SLAVE, DATABITS=> DATABITS, PARITY=> PARITY, BAUDRATE=> BAUDRATE);(*获取 RS485 接口 1 的通讯参数*)

3.24.3. ModbusSerial0_Master——RS485_0 口 Modbus-RTU 主站功能块

RS485_0 口 Modbus-RTU 主站功能块指令 ModbusSerial0_Master 在库文件中的图示表达和相关参数说明如下：



输入参数	数据类型	功能描述	参数值说明
EN	BOOL	使能	0：无效 1：上升沿使能
RW	BYTE	读/写选择	0：读数据 1：写数据
SLAVE	BYTE	Modbus 从站地址	1~247
ADDRESS	DWORD	从站存放数据地址的首地址	首地址采用标准 Modbus-RTU 驱动的地址填写方式，即采用六位数（十进制 xxxxxx），其中十万位（十进制）可以选择为 0、1、3、4，分别是 0 代表从站开关量输出（读写位 bit 寄存器），1 代表开关量输入（只读位 bit 寄存器），3 代表模拟量输入（只读字 WORD 寄存器）、4 代表模拟量输出（读写字 WORD 寄存器），从站数据地址为后五位（十进制，个位数到万位数，范围是 0~65535）。

			例如： 000001 表示从站 Modbus 地址为 00001 的开关量输出点，405050 表示从站 Modbus 地址为 05050 的模拟量输出点。
NUMBER	BYTE	数据长度	范围：0~120 （位 bit 长度或字 WORD 长度） 特殊说明： 如果设定 NUMBER 为 0，在开关量（位 Bit）写操作时，主站发送报文通讯功能码为 05（十进制）；如果 NUMBER 为 1，在开关量（位 Bit）写操作时，主站发送报文通讯功能码为 15（十进制）； 如果设定 NUMBER 为 0，在模拟量（字 WORD）写操作时，主站发送报文通讯功能码为 06（十进制）；如果 NUMBER 为 1，在模拟量（字 WORD）写操作时，主站发送报文通讯功能码为 16（十进制）。
TBL	DWORD	PLC 主站存放数据地址的绝对地址	变量或 M/N 区寄存器绝对物理地址，可以通过 ADR 指令进行获取。 举例： （1）变量声明：ARR_01 AT %MW0: ARRAY [1..100] OF WORD; 取变量绝对地址：ADR（ARR_01）; 取 M 区绝对地址：ADR（%MW0）。 （2）变量声明：ARR_02AT %MD0: ARRAY [1..100] OF REAL; 取变量绝对地址：ADR（ARR_02）; 取 M 区绝对地址：ADR（%MD0）。
RX_TIME OUT	DWORD	接收超时时间（ms）	>20 ，数据发送完成后开始计时，每个扫描周期内都会记录一个时间值，当该扫描周期内接收到有效数据，会刷新该时间值，多个扫描周期未接收到有效数据，累计时间值超过接收超时时间，且未接收完所有的数据，指令报错
TX_TIME OUT	DWORD	发送超时时间（ms）	>20，发送超时时间内，发送数据未完成，指令报错
BREAK	WORD	数据接收完成判断时间（ms）	≥0，数据发送完成后，接收到第一个有效数据开始计时，数据接收未完成，且在 BREAK 时间未接收到下一帧有效数据，指令报错 （设置为 0，内部会设置为 100ms）
输出参数	**数据类型**	**功能描述**	**参数值说明**
Q	BOOL	操作结果	0：未完成 1：完成 （使能 EN 为 1，操作结果 Q 为 1，错误代码 ERROR 为 0，则单次通讯完成）

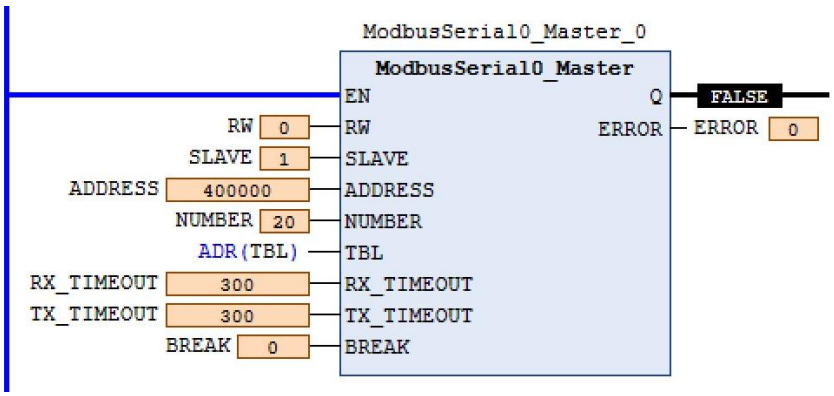
ERROR	BYTE	错误代码	0：无错误 Bit0：数据长度超出限制 Bit1：地址超出范围 Bit2：无效的功能码 Bit3：无效的从站地址 Bit4：无效的 TBL 值 Bit5：CRC 校验错误 Bit6：从站应答失败 Bit7：操作超时 （字节中的 Bit0~Bit7 分别代表不同故障，每个故障显示互相独立。

变量定义

	类别	名称	地址	数据类型	初值	注释	属性
1	VAR	ModbusSerial0_Master_0		ModbusSerial0_Master			
2	VAR	RW		BYTE	0		
3	VAR	SLAVE		BYTE	1		
4	VAR	ADDRESS		DWORD	400000		
5	VAR	NUMBER		BYTE	20		
6	VAR	TBL		ARRAY[1..20] OF WORD			
7	VAR	RX_TIMEOUT		DWORD	300		
8	VAR	TX_TIMEOUT		DWORD	300		
9	VAR	BREAK		WORD	0		
10	VAR	ERROR		BYTE			

编程语言 | **程序**

梯形图（LD）

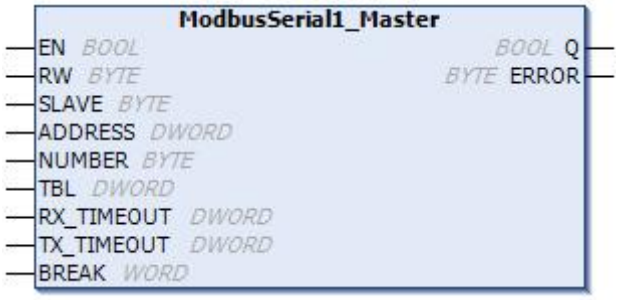


结构化文本（ST）

```
ModbusSerial0_Master_0(
EN:=TRUE ,
RW:=RW ,
SLAVE:=SLAVE ,
ADDRESS:=ADDRESS ,
NUMBER:=NUMBER,
TBL:=ADR(TBL) ,
RX_TIMEOUT:=RX_TIMEOUT ,
TX_TIMEOUT:=TX_TIMEOUT ,
BREAK:=BREAK ,
Q=> ,
ERROR=>ERROR );(*读取从站地址为 1 的设备中 Modbus 地址为 400000 开始的 20 个模拟量*)
```

3.24.4. ModbusSerial1_Master——RS485_1 口 Modbus-RTU 主站功能块

RS485_1 口 Modbus-RTU 主站功能块指令 ModbusSerial1_Master 在库文件中的图示表达和相关参数说明如下：



输入参数	数据类型	功能描述	参数值说明
EN	BOOL	使能	0：无效 1：上升沿使能
RW	BYTE	读/写选择	0：读数据 1：写数据
SLAVE	BYTE	Modbus 从站地址	1~247
ADDRESS	DWORD	从站存放数据地址的首地址	首地址采用标准 Modbus-RTU 驱动的地址填写方式，即采用六位数（十进制 xxxxxx），其中十万位（十进制）可以选择为 0、1、3、4，分别是 0 代表从站开关量输出（读写位 bit 寄存器），1 代表开关量输入（只读位 bit 寄存器），3 代表模拟量输入（只读字 WORD 寄存器）、4 代表模拟量输出（读写字 WORD 寄存器），从站数据地址为后五位（十进制，个位数到万位数，范围是 0~65535）。 例如： 000001 表示从站 Modbus 地址为 00001 的开关量输出点，405050 表示从站 Modbus 地址为 05050 的模拟量输出点。
NUMBER	BYTE	数据长度	范围：0~120 （位 bit 长度或字 WORD 长度） 特殊说明： 如果设定 NUMBER 为 0，在开关量（位 Bit）写操作时，主站发送报文通讯功能码为 05（十进制）；如果 NUMBER 为 1，在开关量（位 Bit）写操作时，主站发送报文通讯功能码为 15（十进制）； 如果设定 NUMBER 为 0，在模拟量（字 WORD）写操作时，主站发送报文通讯功能码为 06（十进制）；如果 NUMBER 为 1，在模拟量（字 WORD）写操作时，主站发送报文通讯功能码为 16（十进制）。
TBL	DWORD	PLC 主站存放数据地址的绝对地址	变量或 M/N 区寄存器绝对物理地址，通过 ADR 指令进行获取。 举例：

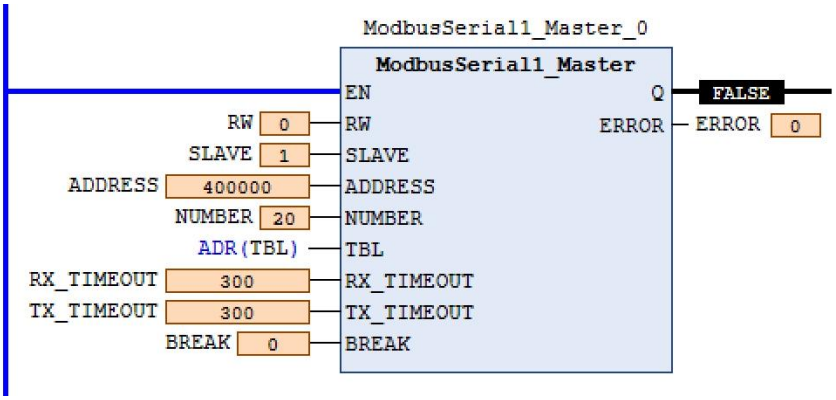
			（1）变量声明：ARR_01 AT %MW0: ARRAY [1..100] OF WORD; 取变量绝对地址：ADR（ARR_01）; 取 M 区绝对地址：ADR（%MW0）。 （2）变量声明：ARR_02AT %MD0: ARRAY [1..100] OF REAL; 取变量绝对地址：ADR（ARR_02）; 取 M 区绝对地址：ADR（%MD0）。
RX_TIME OUT	DWORD	接收超时时间（ms）	>20 ，数据发送完成后开始计时，每个扫描周期内都会记录一个时间值，当该扫描周期内接收到有效数据，会刷新该时间值，多个扫描周期未接收到有效数据，累计时间值超过接收超时时间，且未接收完所有的数据，指令报错
TX_TIME OUT	DWORD	发送超时时间（ms）	>20，发送超时时间内，发送数据未完成，指令报错
BREAK	WORD	数据接收完成判断时间（ms）	≥0，数据发送完成后，接收到第一个有效数据开始计时，数据接收未完成，且在 BREAK 时间未接收到下一帧有效数据，指令报错 （设置为 0，内部会设置为 100ms）
输出参数	**数据类型**	**功能描述**	**参数值说明**
Q	BOOL	操作结果	0：未完成 1：完成 （使能 EN 为 1，操作结果 Q 为 1，错误代码 ERROR 为 0，则单次通讯完成）
ERROR	BYTE	错误代码	0：无错误 Bit0：数据长度超出限制 Bit1：地址超出范围 Bit2：无效的功能码 Bit3：无效的从站地址 Bit4：无效的 TBL 值 Bit5：CRC 校验错误 Bit6：从站应答失败 Bit7：操作超时 （字节中的 Bit0~Bit7 分别代表不同故障，每个故障显示互相独立。）

变量定义

	类别	名称	地址	数据类型	初值	注释	属性
1	VAR	**ModbusSerial1_Master_0**		ModbusSerial1_Master			
2	VAR	**RW**		BYTE	0		
3	VAR	**SLAVE**		BYTE	1		
4	VAR	**ADDRESS**		DWORD	400000		
5	VAR	**NUMBER**		BYTE	20		
6	VAR	**TBL**		ARRAY[1..20] OF WORD			
7	VAR	**RX_TIMEOUT**		DWORD	300		
8	VAR	**TX_TIMEOUT**		DWORD	300		
9	VAR	**BREAK**		WORD	0		
10	VAR	**ERROR**		BYTE			

编程语言 | **程序**

梯形图（LD）

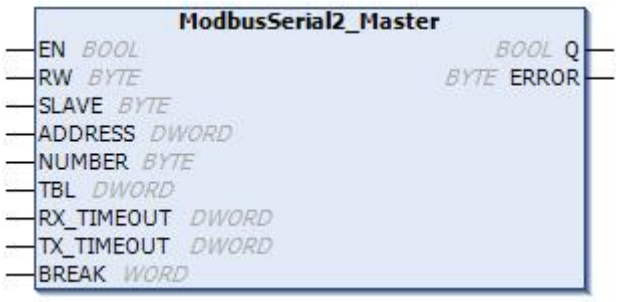


结构化文本（ST）

```
ModbusSerial1_Master_0(
EN:=TRUE ,
RW:=RW ,
SLAVE:=SLAVE ,
ADDRESS:=ADDRESS ,
NUMBER:=NUMBER,
TBL:=ADR(TBL) ,
RX_TIMEOUT:=RX_TIMEOUT ,
TX_TIMEOUT:=TX_TIMEOUT ,
BREAK:=BREAK ,
Q=> ,
ERROR=>ERROR );(*读取从站地址为 1 的设备中 Modbus 地址为 400000 开始的 20 个模拟量*)
```

3.24.5. ModbusSerial2_Master——RS485_2 口 Modbus-RTU 主站功能块

RS485_2 口 Modbus-RTU 主站功能块指令 ModbusSerial2_Master 在库文件中的图示表达和相关参数说明如下：



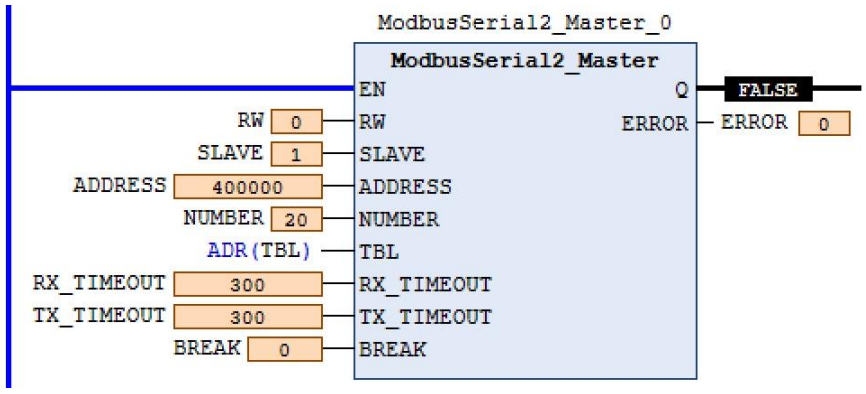
输入参数	数据类型	功能描述	参数值说明
EN	BOOL	使能	0：无效 1：上升沿使能
RW	BYTE	读/写选择	0：读数据 1：写数据
SLAVE	BYTE	Modbus 从站地址	1~247
ADDRESS	DWORD	从站存放数据地址的首地址	首地址采用标准 Modbus-RTU 驱动的地址填写方式，即采用六位数（十进制 xxxxxx），其中十万位（十进制）可以选择为 0、1、3、4，分别是 0 代表从站开关量输出（读写位 bit 寄存器），1 代表开关量输入（只读位 bit 寄存器），3 代表模拟量输入（只读字 WORD 寄存器）、4 代表模拟量输出（读写字 WORD 寄存器），从站数据地址为后五位（十进制，个位数到万位数，范围是 0~65535）。 例如： 000001 表示从站 Modbus 地址为 00001 的开关量输出点，405050 表示从站 Modbus 地址为 05050 的模拟量输出点。
NUMBER	BYTE	数据长度	范围：0~120 （位 bit 长度或字 WORD 长度） 特殊说明： 如果设定 NUMBER 为 0，在开关量（位 Bit）写操作时，主站发送报文通讯功能码为 05（十进制）；如果 NUMBER 为 1，在开关量（位 Bit）写操作时，主站发送报文通讯功能码为 15（十进制）； 如果设定 NUMBER 为 0，在模拟量（字 WORD）写操作时，主站发送报文通讯功能码为 06（十进制）；如果 NUMBER 为 1，在模拟量（字 WORD）写操作时，主站发送报文通讯功能码为 16（十进制）。
TBL	DWORD	PLC 主站存放数据地址的绝对地址	变量或 M/N 区寄存器绝对物理地址，通过 ADR 指令进行获取。 举例： （1）变量声明：ARR_01 AT %MW0: ARRAY [1..100] OF WORD; 取变量绝对地址：ADR（ARR_01）; 取 M 区绝对地址：ADR（%MW0）。 （2）变量声明：ARR_02AT %MD0: ARRAY [1..100] OF REAL; 取变量绝对地址：ADR（ARR_02）; 取 M 区绝对地址：ADR（%MD0）。
RX_TIME OUT	DWORD	接收超时时间（ms）	>20 ，数据发送完成后开始计时，每个扫描周期内都会记录一个时间值，当该扫描周期内接收到有效数据，会刷新该时间值，多个扫描周期未接收到有效数据，累计时间值超

			过接收超时时间，且未接收完所有的数据，指令报错
TX_TIMEOUT	DWORD	发送超时时间（ms）	>20，发送超时时间内，发送数据未完成，指令报错
BREAK	WORD	数据接收完成判断时间（ms）	≥0，数据发送完成后，接收到第一个有效数据开始计时，数据接收未完成，且在 BREAK 时间未接收到下一帧有效数据，指令报错 （设置为 0，内部会设置为 100ms）
输出参数	**数据类型**	**功能描述**	**参数值说明**
Q	BOOL	操作结果	0：未完成 1：完成 （使能 EN 为 1，操作结果 Q 为 1，错误代码 ERROR 为 0，则单次通讯完成）
ERROR	BYTE	错误代码	0：无错误 Bit0：数据长度超出限制 Bit1：地址超出范围 Bit2：无效的功能码 Bit3：无效的从站地址 Bit4：无效的 TBL 值 Bit5：CRC 校验错误 Bit6：从站应答失败 Bit7：操作超时 （字节中的 Bit0~Bit7 分别代表不同故障，每个故障显示互相独立。）

变量定义

	类别	名称	地址	数据类型	初值	注释	属性
1	VAR	ModbusSerial2_Master_0		ModbusSerial2_Master			
2	VAR	RW		BYTE	0		
3	VAR	SLAVE		BYTE	1		
4	VAR	ADDRESS		DWORD	400000		
5	VAR	NUMBER		BYTE	20		
6	VAR	TBL		ARRAY[1..20] OF WORD			
7	VAR	RX_TIMEOUT		DWORD	300		
8	VAR	TX_TIMEOUT		DWORD	300		
9	VAR	BREAK		WORD	0		
10	VAR	ERROR		BYTE			

编程语言	程序

梯形图 （LD）	
结构化文本 （ST）	ModbusSerial2_Master_0(EN:=TRUE , RW:=RW , SLAVE:=SLAVE , ADDRESS:=ADDRESS , NUMBER:=NUMBER, TBL:=ADR(TBL) , RX_TIMEOUT:=RX_TIMEOUT , TX_TIMEOUT:=TX_TIMEOUT , BREAK:=BREAK , Q=> , ERROR=>ERROR);(*读取从站地址为 1 的设备中 Modbus 地址为 400000 开始的 20 个模拟量*)

3.24.6. FreeSerial0_Send——RS485_0 口自由协议通讯数据发送

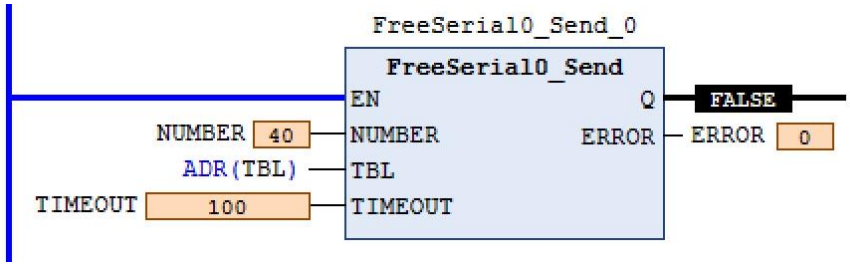
RS485_0 口自由协议通讯数据发送功能块指令 FreeSerial0_Send 在库文件中的图示表达和相关参数说明如下：

输入参数	数据类型	功能描述	参数值说明
EN	BOOL	使能	0：无效 1：上升沿使能
NUMBER	WORD	数据长度	≤280（字节数量）
TBL	DWORD	PLC 存放发送数据的绝对地址	变量或 M/N 区寄存器绝对物理地址，通过 ADR 指令进行获取。 举例： （1）变量声明：ARR_01 AT %MW0: ARRAY [1..100] OF WORD; 取变量绝对地址：ADR（ARR_01）; 取 M 区绝对地址：ADR（%MW0）。 （2）变量声明：ARR_01 AT %MD0: ARRAY [1..100] OF REAL; 取变量绝对地址：ADR（ARR_02）; 取 M 区绝对地址：ADR（%MD0）。

TIMEOUT	DWORD	发送超时时间（ms）	>50，发送超时时间内，发送数据未完成，指令报错
输出参数	**数据类型**	**功能描述**	**参数值说明**
Q	BOOL	操作结果	0：未完成 1：完成 （使能 EN 为 1，错误代码 ERROR 为 0，则单次发送数据完成）
ERROR	BYTE	错误代码	0：无错误 1：驱动超时错误 2：断开状态错误 3：帧错误 4：奇偶校验错误 5：超限错误 6：其它错误 128：发送超时

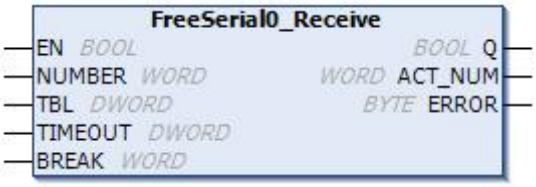
变量定义

	类别	名称	地址	数据类型	初值	注释	属性
1	VAR	FreeSerial0_Send_0		FreeSerial0_Send			
2	VAR	NUMBER		BYTE	40		
3	VAR	TBL		ARRAY[1..20] OF WORD			
4	VAR	TIMEOUT		DWORD	100		
5	VAR	ERROR		BYTE			

编程语言	程序
梯形图（LD）	
结构化文本（ST）	FreeSerial0_Send_0(EN:= TRUE, NUMBER:= NUMBER, TBL:= ADR(TBL), TIMEOUT:=TIMEOUT, Q=> , ERROR=> ERROR);(*自由口发送数组 TBL 里的前 40 个字节*)

3.24.7. FreeSerial0_Receive——RS485_0 口自由协议通讯数据接收

RS485_0 口自由协议通讯数据接收功能块指令 FreeSerial0_Receive 在库文件中的图示表达和相关参数说明如下：

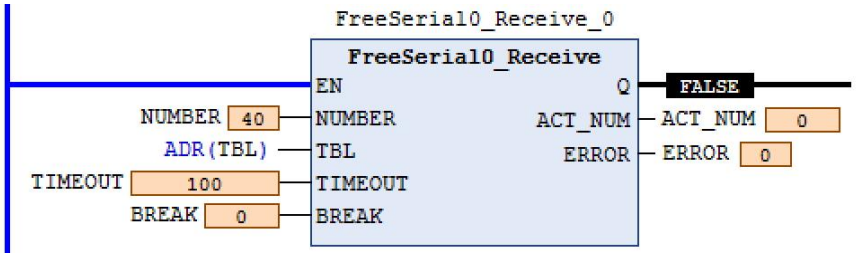


输入参数	数据类型	功能描述	参数值说明
EN	BOOL	使能	0：无效 1：上升沿使能
NUMBER	WORD	数据长度	≤280（字节数量）
TBL	DWORD	PLC 存放接收数据的绝对地址	变量或 M/N 区寄存器绝对物理地址，通过 ADR 指令进行获取。 举例： （1）变量声明：ARR_01 AT %MW0: ARRAY [1..100] OF WORD; 取变量绝对地址：ADR（ARR_01）; 取 M 区绝对地址：ADR（%MW0）。 （2）变量声明：ARR_02AT %MD0: ARRAY [1..100] OF REAL; 取变量绝对地址：ADR（ARR_02）; 取 M 区绝对地址：ADR（%MD0）。
TIMEOUT	DWORD	接收超时时间（ms）	>20 ，数据发送完成后开始计时，每个扫描周期内都会记录一个时间值，当该扫描周期内接收到有效数据，会刷新该时间值，多个扫描周期未接收到有效数据，累计时间值超过接收超时时间，且未接收完所有的数据，指令报错
BREAK	WORD	数据接收完成判断时间（ms）	≥0，数据发送完成后，接收到第一个有效数据开始计时，数据接收未完成，且在 BREAK 时间未接收到下一帧有效数据，指令报错 （设置为 0，内部会设置为 100ms）
输出参数	**数据类型**	**功能描述**	**参数值说明**
Q	BOOL	操作结果	0：未完成 1：完成 （使能 EN 为 1，接收的实际字节数 ACT_NUM >0，则单次接收数据完成）
ACT_NUM	WORD	接收的实际字节数	
ERROR	BYTE	错误代码	0：无错误 1：超时错误 2：中断状态错误 3：帧错误 4：奇偶校验错误 5：超限错误

			6：其它错误 128：接收缓冲区溢出错误

变量定义

	类别	名称	地址	数据类型	初值	注释	属性
1	VAR	FreeSerial0_Receive_0		FreeSerial0_Receive			
2	VAR	NUMBER		BYTE	40		
3	VAR	TBL		ARRAY[1..20] OF WORD			
4	VAR	TIMEOUT		DWORD	100		
5	VAR	BREAK		WORD			
6	VAR	ACT_NUM		WORD			
7	VAR	ERROR		BYTE			

编程语言	程序
梯形图（LD）	
结构化文本（ST）	FreeSerial0_Receive_0(EN:= TRUE, NUMBER:= NUMBER, TBL:= ADR(TBL), TIMEOUT:= TIMEOUT, BREAK:= BREAK, Q=> , ACT_NUM=> ACT_NUM, ERROR=> ERROR);(*自由口接收 40 个字节存入数组 TBL 中*)

提示：当自由口发送指令与接收指令同时被调用时，优先执行发送指令。

3.24.8. FreeSerial1_Send——RS485_1 口自由协议通讯数据发送

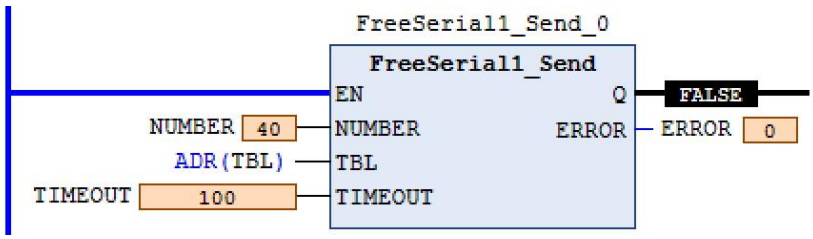
RS485_1 口自由协议通讯数据发送功能块指令 FreeSerial1_Send 在库文件中的图示表达和相关参数说明如下：

输入参数	数据类型	功能描述	参数值说明
EN	BOOL	使能	0：无效 1：上升沿使能
NUMBER	WORD	数据长度	≤280（字节数量）
TBL	DWORD	PLC 存放发送数据的绝对地址	变量或 M/N 区寄存器绝对物理地址，通过 ADR 指令进行获取。 举例： （1）变量声明：ARR_01 AT %MW0:

			ARRAY [1..100] OF WORD; 取变量绝对地址：ADR（ARR_01）; 取 M 区绝对地址：ADR（%MW0）。 （2）变量声明：ARR_02AT %MD0: ARRAY [1..100] OF REAL; 取变量绝对地址：ADR（ARR_02）; 取 M 区绝对地址：ADR（%MD0）。
TIMEOUT	DWORD	发送超时时间（ms）	>50，发送超时时间内，发送数据未完成，指令报错
输出参数	**数据类型**	**功能描述**	**参数值说明**
Q	BOOL	操作结果	0：未完成 1：完成 （使能 EN 为 1，错误代码 ERROR 为 0，则单次发送数据完成）
ERROR	BYTE	错误代码	0：无错误 1：驱动超时错误 2：断开状态错误 3：帧错误 4：奇偶校验错误 5：超限错误 6：其它错误 128：发送超时

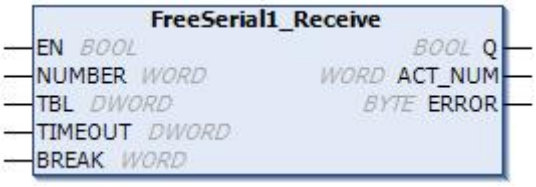
变量定义

	类别	名称	地址	数据类型	初值	注释	属性
1	VAR	FreeSerial1_Send_0		FreeSerial1_Send			
2	VAR	NUMBER		BYTE	40		
3	VAR	TBL		ARRAY[1..20] OF WORD			
4	VAR	TIMEOUT		DWORD	100		
5	VAR	ERROR		BYTE			

编程语言	程序
梯形图（LD）	
结构化文本（ST）	FreeSerial1_Send_0(EN:= TRUE, NUMBER:= NUMBER, TBL:= ADR(TBL), TIMEOUT:=TIMEOUT, Q=> , ERROR=> ERROR);(*自由口发送数组 TBL 里的前 40 个字节*)

3.24.9. FreeSerial1_Receive——RS485_1 口自由协议通讯数据接收

RS485_1 口自由协议通讯数据接收功能块指令 FreeSerial1_Receive 在库文件中的图示表达和相关参数说明如下：

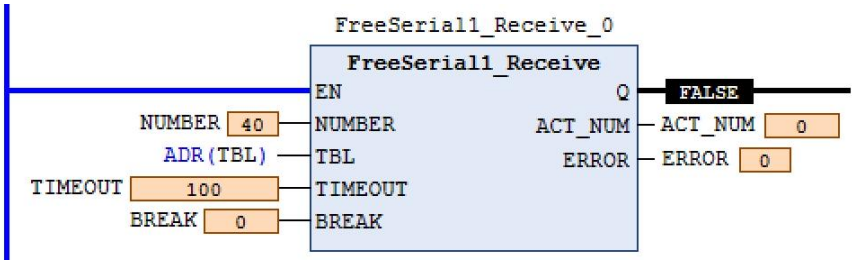


输入参数	数据类型	功能描述	参数值说明
EN	BOOL	使能	0：无效 1：上升沿使能
NUMBER	WORD	数据长度	≤280（字节数量）
TBL	DWORD	PLC 存放接收数据的绝对地址	变量或 M/N 区寄存器绝对物理地址，通过 ADR 指令进行获取。 举例： （1）变量声明：ARR_01 AT %MW0: ARRAY [1..100] OF WORD; 取变量绝对地址：ADR（ARR_01）; 取 M 区绝对地址：ADR（%MW0）。 （2）变量声明：ARR_02AT %MD0: ARRAY [1..100] OF REAL; 取变量绝对地址：ADR（ARR_02）; 取 M 区绝对地址：ADR（%MD0）。
TIMEOUT	DWORD	接收超时时间（ms）	>20 ，数据发送完成后开始计时，每个扫描周期内都会记录一个时间值，当该扫描周期内接收到有效数据，会刷新该时间值，多个扫描周期未接收到有效数据，累计时间值超过接收超时时间，且未接收完所有的数据，指令报错
BREAK	WORD	数据接收完成判断时间（ms）	≥0，数据发送完成后，接收到第一个有效数据开始计时，数据接收未完成，且在 BREAK 时间未接收到下一帧有效数据，指令报错 （设置为 0，内部会设置为 100ms）
输出参数	**数据类型**	**功能描述**	**参数值说明**
Q	BOOL	操作结果	0：未完成 1：完成 （使能 EN 为 1，接收的实际字节数 ACT_NUM >0，则单次接收数据完成）
ACT_NUM	WORD	接收的实际字节数	
ERROR	BYTE	错误代码	0：无错误 1：超时错误 2：中断状态错误

			3：帧错误 4：奇偶校验错误 5：超限错误 6：其它错误 128：接收缓冲区溢出错误

变量定义

	类别	名称	地址	数据类型	初值	注释	属性
1	VAR	**FreeSerial1_Receive_0**		FreeSerial1_Receive			
2	VAR	**NUMBER**		BYTE	40		
3	VAR	**TBL**		ARRAY[1..20] OF WORD			
4	VAR	**TIMEOUT**		DWORD	100		
5	VAR	**BREAK**		WORD			
6	VAR	**ACT_NUM**		WORD			
7	VAR	**ERROR**		BYTE			

编程语言	程序
梯形图（LD）	
结构化文本（ST）	FreeSerial1_Receive_0(EN:= TRUE, NUMBER:= NUMBER, TBL:= ADR(TBL), TIMEOUT:= TIMEOUT, BREAK:= BREAK, Q=> , ACT_NUM=> ACT_NUM, ERROR=> ERROR);(*自由口接收 40 个字节存入数组 TBL 中*)

提示：当自由口发送指令与接收指令同时被调用时，优先执行发送指令。

3.24.10. FreeSerial2_Send——RS485_2 口自由协议通讯数据发送

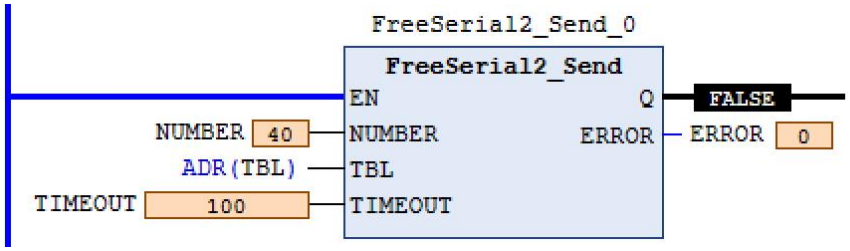
RS485_2 口自由协议通讯数据发送功能块指令 FreeSerial2_Send 在库文件中的图示表达和相关参数说明如下：

输入参数	数据类型	功能描述	参数值说明
EN	BOOL	使能	0：无效 1：上升沿使能
NUMBER	WORD	数据长度	≤280（字节数量）

TBL	DWORD	PLC 存放发送数据的绝对地址	变量或 M/N 区寄存器绝对物理地址，通过 ADR 指令进行获取。 举例： （1）变量声明：ARR_01 AT %MW0: ARRAY [1..100] OF WORD; 取变量绝对地址：ADR（ARR_01）； 取 M 区绝对地址：ADR（%MW0）。 （2）变量声明：ARR_02AT %MD0: ARRAY [1..100] OF REAL; 取变量绝对地址：ADR（ARR_02）； 取 M 区绝对地址：ADR（%MD0）。
TIMEOUT	DWORD	发送超时时间（ms）	>50，发送超时时间内，发送数据未完成，指令报错
输出参数	**数据类型**	**功能描述**	**参数值说明**
Q	BOOL	操作结果	0：未完成 1：完成 （使能 EN 为 1，错误代码 ERROR 为 0，则单次发送数据完成）
ERROR	BYTE	错误代码	0：无错误 1：驱动超时错误 2：断开状态错误 3：帧错误 4：奇偶校验错误 5：超限错误 6：其它错误 128：发送超时

变量定义

	类别	名称	地址	数据类型	初值	注释	属性
1	VAR	**FreeSerial2_Send_0**		FreeSerial2_Send			
2	VAR	**NUMBER**		BYTE	40		
3	VAR	**TBL**		ARRAY[1..20] OF WORD			
4	VAR	**TIMEOUT**		DWORD	100		
5	VAR	**ERROR**		BYTE			

编程语言	程序
梯形图（LD）	
结构化文本（ST）	FreeSerial2_Send_0(EN:= TRUE,

	NUMBER:= NUMBER, TBL:= ADR(TBL), TIMEOUT:=TIMEOUT, Q=> , ERROR=> ERROR);(*自由口发送数组 TBL 里的前 40 个字节*)

3.24.11. FreeSerial2_Receive——RS485_2 口自由协议通讯数据接收

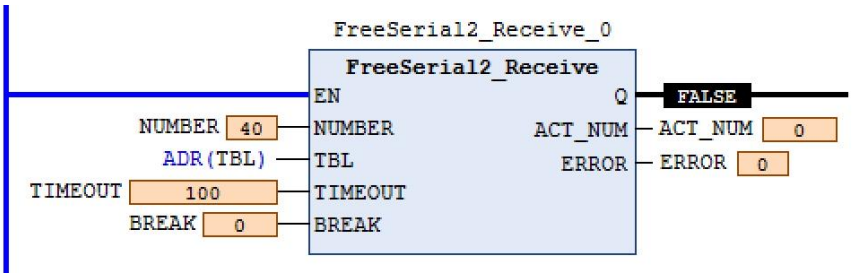
RS485_2 口自由协议通讯数据接收功能块指令 FreeSerial2_Receive 在库文件中的图示表达和相关参数说明如下：

输入参数	数据类型	功能描述	参数值说明
EN	BOOL	使能	0：无效 1：上升沿使能
NUMBER	WORD	数据长度	≤280（字节数量）
TBL	DWORD	PLC 存放接收数据的绝对地址	变量或 M/N 区寄存器绝对物理地址，通过 ADR 指令进行获取。 举例： （1）变量声明：ARR_01 AT %MW0: ARRAY [1..100] OF WORD; 取变量绝对地址：ADR（ARR_01）; 取 M 区绝对地址：ADR（%MW0）。 （2）变量声明：ARR_02AT %MD0: ARRAY [1..100] OF REAL; 取变量绝对地址：ADR（ARR_02）; 取 M 区绝对地址：ADR（%MD0）。
TIMEOUT	DWORD	接收超时时间（ms）	>20 ，数据发送完成后开始计时，每个扫描周期内都会记录一个时间值，当该扫描周期内接收到有效数据，会刷新该时间值，多个扫描周期未接收到有效数据，累计时间值超过接收超时时间，且未接收完所有的数据，指令报错
BREAK	WORD	数据接收完成判断时间（ms）	≥0，数据发送完成后，接收到第一个有效数据开始计时，数据接收未完成，且在 BREAK 时间未接收到下一帧有效数据，指令报错 （设置为 0，内部会设置为 100ms）
输出参数	**数据类型**	**功能描述**	**参数值说明**
Q	BOOL	操作结果	0：未完成 1：完成 （使能 EN 为 1，接收的实际字节数 ACT_NUM >0，则单次接收数据完成）

ACT_NUM	WORD	接收的实际字节数	
ERROR	BYTE	错误代码	0：无错误 1：超时错误 2：中断状态错误 3：帧错误 4：奇偶校验错误 5：超限错误 6：其它错误 128：接收缓冲区溢出错误

变量定义

	类别	名称	地址	数据类型	初值	注释	属性
1	VAR	FreeSerial2_Receive_0		FreeSerial2_Receive			
2	VAR	NUMBER		BYTE	40		
3	VAR	TBL		ARRAY[1..20] OF WORD			
4	VAR	TIMEOUT		DWORD	100		
5	VAR	BREAK		WORD			
6	VAR	ACT_NUM		WORD			
7	VAR	ERROR		BYTE			

编程语言	程序
梯形图（LD）	
结构化文本（ST）	FreeSerial2_Receive_0(EN:= TRUE, NUMBER:= NUMBER, TBL:= ADR(TBL), TIMEOUT:= TIMEOUT, BREAK:= BREAK, Q=> , ACT_NUM=> ACT_NUM, ERROR=> ERROR);(*自由口接收 40 个字节存入数组 TBL 中*)

提示：当自由口发送指令与接收指令同时被调用时，优先执行发送指令。

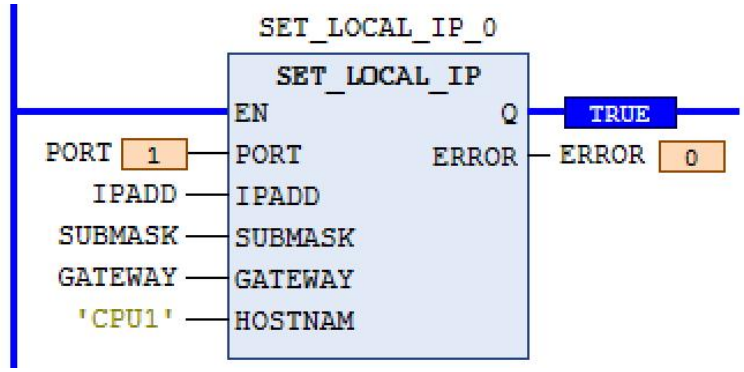
3.25. 以太网口通讯指令（CmpRPC3000-library）

3.25.1. SET_LOCAL_IP——设置以太网口 TCP/IPv4 参数

设置以太网口 TCP/IPv4 参数指令 SET_LOCAL_IP 在库文件中的图示表达和相关参数说明如下：

输入参数	数据类型	功能描述	参数值说明
EN	BOOL	使能	0：无效 1：上升沿使能 （上升沿触发，需要注意，重新上电后修改生效）
PORT	BYTE	以太网网口硬件编号	1：以太网网口 1 2：以太网网口 2 3：软冗余以太网网口号(IPADD 输入引脚为软冗余对方 CPU 设备以太网网口 2 的 IP 地址，其余引脚输入功能无效)
IPADD	ARRAY [0..3] OF WORD	IP 地址	数组变量定义，如[192.168.1.10]中的'192.168.1.10'为 IP 地址
SUBMASK	ARRAY [0..3] OF WORD	子网掩码	数组变量定义，如[255.255.255.0]中的'255.255.255.0'为子网掩码
GATEWAY	ARRAY [0..3] OF WORD	网关	数组变量定义，如[192.168.1.1]中的'192.168.1.1'为网关
HOSTNAM	STRING(64)	主机名	字符串，长度≤64，例如定义变量'CPU1'
输出参数	**数据类型**	**功能描述**	**参数值说明**
Q	BOOL	操作结果	0：未完成 1：完成 （使能 EN 为 1，操作结果 Q 为 1，错误代码 ERROR 为 0，则网口通讯参数设置完成）
ERROR	BYTE	错误代码	0：无错误 Bit0：无效以太网口号 Bit1：无效 IP 地址 Bit2：无效子网掩码 Bit3：无效网关 （字节中的 Bit0~Bit3 分别代表不同故障，每个故障显示互相独立。）
变量定义			

	类别	名称	地址	数据类型	初值	注释	属性
1	VAR	SET_LOCAL_IP_0		SET_LOCAL_IP			
2	VAR	PORT		BYTE	1		
3	VAR	IPADD		ARRAY [0..3] OF WORD	[192, 168, 1, 4]		
4	VAR	SUBMASK		ARRAY [0..3] OF WORD	[3(255), 0]		
5	VAR	GATEWAY		ARRAY [0..3] OF WORD	[192, 168, 1, 1]		
6	VAR	ERROR		BYTE			

编程语言	程序
梯形图（LD）	
结构化文本（ST）	SET_LOCAL_IP_0(EN:= TRUE, PORT:=PORT, IPADD:= IPADD, SUBMASK:= SUBMASK, GATEWAY:= GATEWAY, HOSTNAM:= 'CPU1', Q=> , ERROR=> ERROR);(*设置 PLC 以太网口 1 的 IP 地址为 192.168.1.4、子网掩码为 255.255.255.0、网关为 192.168.1.1*)

3.25.2. GET_Current_IP——读取以太网口 TCP/IPv4 参数

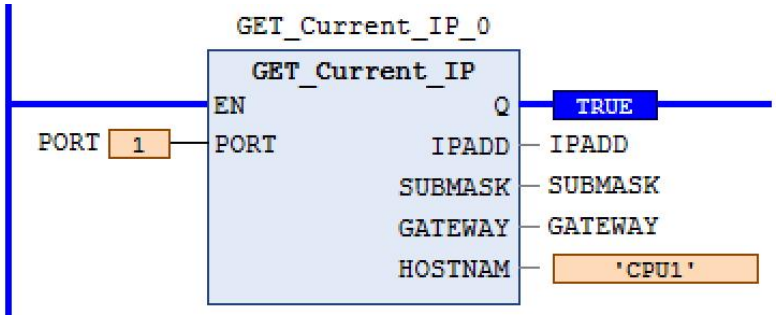
读取以太网口 TCP/IPv4 参数指令 GET_LOCAL_IP 在库文件中的图示表达和相关参数说明如下：

输入参数	数据类型	功能描述	参数值说明
EN	BOOL	使能	0：无效 1：上升沿使能
PORT	BYTE	以太网网口硬件编号	1：以太网网口 1 2：以太网网口 2 3：软冗余以太网网口号
输出参数	**数据类型**	**功能描述**	**参数值说明**
Q	BOOL	操作结果	0：未完成 1：完成
IPADD	ARRAY [0..3] OF	IP 地址	数组变量定义，如[192.168.1.10]中的

	WORD		'192.168.1.10'为 IP 地址
SUBMASK	ARRAY [0..3] OF WORD	子网掩码	数组变量定义，如[255.255.255.0]中的'255.255.255.0'为子网掩码
GATEWAY	ARRAY [0..3] OF WORD	网关	数组变量定义，如[192.168.1.1]中的'192.168.1.1'为网关
HOSTNAM	STRING(64)	主机名	字符串，长度≤64，例如定义变量'CPU1'

变量定义

	类别	名称	地址	数据类型	初值	注释	属性
1	VAR	GET_Current_IP_0		GET_Current_IP			
2	VAR	PORT		BYTE	1		
3	VAR	IPADD		ARRAY [0..3] OF WORD			
4	VAR	SUBMASK		ARRAY [0..3] OF WORD			
5	VAR	GATEWAY		ARRAY [0..3] OF WORD			

编程语言	程序
梯形图（LD）	
结构化文本（ST）	GET_CURRENT_IP_0 (EN:= TRUE, PORT:= PORT, Q=> , IPADD=> IPADD, SUBMASK=> SUBMASK, GATEWAY=> GATEWAY, HOSTNAM=>);(*获取 PLCIP 地址*)

3.25.3. CLEAR_LOCAL_IP——清除以太网口 TCP/IPv4 参数

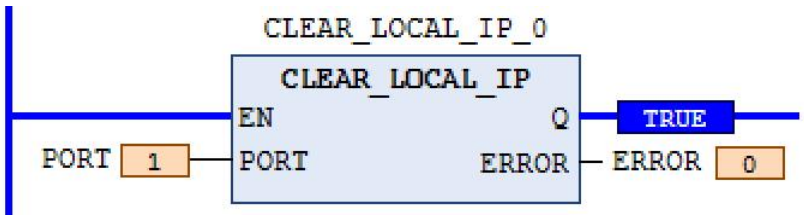
清除以太网口 TCP/IPv4 参数指令 CLEAR _LOCAL_IP 在库文件中的图示表达和相关参数说明如下：

输入参数	数据类型	功能描述	参数值说明
EN	BOOL	使能	0：无效 1：上升沿使能
PORT	BYTE	以太网网口硬件	1：以太网网口 1

		编号	2：以太网网口 2 3：软冗余以太网网口号
输出参数	**数据类型**	**功能描述**	**参数值说明**
Q	BOOL	操作结果	0：未完成 1：完成 （使能为 1，操作结果为 1，错误代码为 0，则网口通讯参数清除完成）
ERROR	BYTE	错误代码	0：无错误 1：无效网口编号

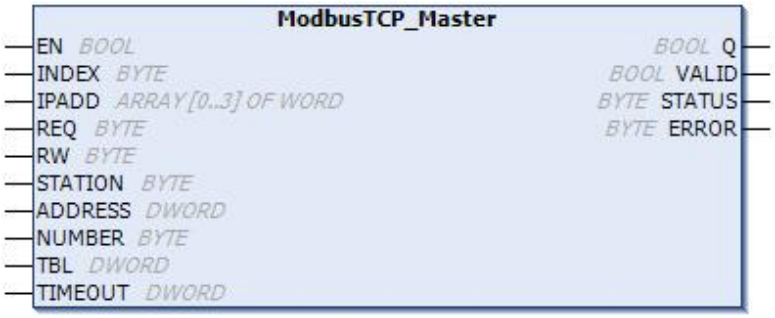
变量定义

	类别	名称	地址	数据类型	初值	注释	属性
1	VAR	CLEAR_LOCAL_IP_0		CLEAR_LOCAL_IP			
2	VAR	PORT		BYTE	1		
3	VAR	ERROR		BYTE			

编程语言	**程序**
梯形图（LD）	
结构化文本（ST）	CLEAR_LOCAL_IP_0(EN:= TRUE, PORT:= PORT, Q=> , ERROR=> ERROR);

3.25.4. ModbusTCP_Master——以太网口 Modbus-TCP 主站功能块

以太网口 Modbus-TCP 主站功能块指令 ModbusTCP_Master 在库文件中的图示表达和相关参数说明如下：



输入参数	**数据类型**	**功能描述**	**参数值说明**
EN	BOOL	使能	0：关闭 socket 1：打开 socket 连接 IP 地址 （上升沿触发，后续操作保持高电平）
INDEX	BYTE	套接字（socket）ID	0~19

IPADD	ARRAY [0..3] OF WORD	从站 IP 地址	数组变量定义，如[192.168.1.10]中的'192.168.1.10'为 IP 地址
REQ	BYTE	请求连接	0：无效 1：高电平有效
RW	BYTE	读/写选择	0：读数据 1：写数据
STATION	BYTE	单元标识符	从站设备地址：0~247
ADDRESS	DWORD	从站存放数据地址的首地址	首地址采用标准 Modbus-TCP 驱动的地址填写方式，即采用六位数（十进制 xxxxxx），其中十万位（十进制）可以选择为 0、1、3、4，分别是 0 代表从站开关量输出（读写位 bit 寄存器），1 代表开关量输入（只读位 bit 寄存器），3 代表模拟量输入（只读字 WORD 寄存器）、4 代表模拟量输出（读写字 WORD 寄存器），从站数据地址为后五位（十进制，个位数到万位数，范围是 0~65535）。 例如： 000001 表示从站 Modbus 地址为 00001 的开关量输出点，405050 表示从站 Modbus 地址为 05050 的模拟量输出点。
NUMBER	BYTE	数据长度	范围：0~120 （位 bit 长度或字 WORD 长度） 特殊说明： 如果设定 NUMBER 为 0，在开关量（位 Bit）写操作时，主站发送报文通讯功能码为 05（十进制）；如果 NUMBER 为 1，在开关量（位 Bit）写操作时，主站发送报文通讯功能码为 15（十进制）； 如果设定 NUMBER 为 0，在模拟量（字 WORD）写操作时，主站发送报文通讯功能码为 06（十进制）；如果 NUMBER 为 1，在模拟量（字 WORD）写操作时，主站发送报文通讯功能码为 16（十进制）。
TBL	DWORD	PLC 主站存放数据地址的绝对地址	变量或 M/N 区寄存器绝对物理地址，通过 ADR 指令进行获取。 举例： （1）变量声明：ARR_01 AT %MW0: ARRAY [1..100] OF WORD; 取变量绝对地址：ADR（ARR_01）; 取 M 区绝对地址：ADR（%MW0）。 （2）变量声明：ARR_02AT %MD0: ARRAY [1..100] OF REAL; 取变量绝对地址：ADR（ARR_02）;

			取 M 区绝对地址：ADR（%MD0）。
TIMEOUT	DWORD	超时时间（ms）	>50，打开连接超时时间与接收数据超时时间
输出参数	**数据类型**	**功能描述**	**参数值说明**
Q	BOOL	打开连接操作结果	0：未完成 1：完成 （使能 EN 为 1，打开连接操作结果 Q 为 1，错误代码 ERROR 为 0，则连接服务端 IP 地址完成）
VALID	BOOL	获取有效数据	0：无效 1：有效 （使能 EN 为 1，打开连接操作结果 Q 为 1，错误代码 ERROR 为 0，请求连接 REQ 为 1，则单次获取有效数据完成）
STATUS	BYTE	操作状态	0：关闭套接字（socket） 1：打开套接字（socket） 2：建立连接 3：连接 IP 地址 4：参数检查 5：组帧 6：超时计数 7：发送数据 8：接收状态 9：接收有效数据 10：接收错误 11：连接被关闭
ERROR	BYTE	错误代码	错误代码的参数值说明需结合操作状态的参数值，详细说明如下表

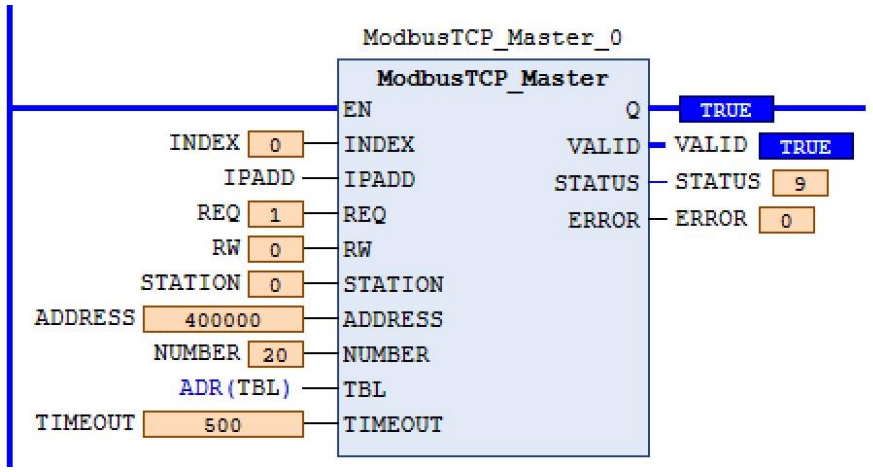
STATUS-操作状态	**ERROR-错误代码**	**参数值说明**	**内部后续动作**
0	0	关闭套接字（socket）	
1	0	打开套接字（socket）	
	其它值	详见附录 A	自动关闭套接字，延时 1 秒，重新打开套接字
2	0	设置套接字（socket）参数	
	其它值	详见附录 A	自动关闭套接字，延时 1 秒，重新打开套接字
3	0	连接 IP 地址	
	其它值	详见附录 A	自动关闭套接字，延时 1 秒，重新打开套接字
4	0	功能块参数检查	
	Bit0	数据长度超限	

	Bit1	地址超限	
	Bit2	无效功能码	
	Bit3	无效 TBL 值	
5	0	组帧	
6	0	发送请求	
	1	发送请求超时	
7	其它值	发送请求错误，详见附录 A	自动关闭套接字，延时 1 秒，重新打开套接字
8	其它值	接收状态查询，详见附录 A	自动关闭套接字，延时 1 秒，重新打开套接字
9	0	接收到有效应答	
	Bit0	MBAP 帧头或功能码错误	
	Bit1	数据长度错误	
	Bit2	数据错误	
10	其它值	接收错误，详见附录 A	自动关闭套接字，延时 1 秒，重新打开套接字
11	1	对方关闭连接	自动关闭套接字，延时 1 秒，重新打开套接字

变量定义

	类别	名称	地址	数据类型	初值	注释	属性
1	VAR	**ModbusTCP_Master_0**		ModbusTCP_Master			
2	VAR	**INDEX**		BYTE	0		
3	VAR	**IPADD**		ARRAY [0..3] OF WORD	[192, 168, 1, 130]		
4	VAR	**REQ**		BYTE	1		
5	VAR	**RW**		BYTE	0		
6	VAR	**STATION**		BYTE	0		
7	VAR	**ADDRESS**		DWORD	400000		
8	VAR	**NUMBER**		BYTE	20		
9	VAR	**TBL**		ARRAY[1..20] OF WORD			
10	VAR	**TIMEOUT**		DWORD	500		
11	VAR	**VALID**		BOOL			
12	VAR	**STATUS**		BYTE			
13	VAR	**ERROR**		BYTE			

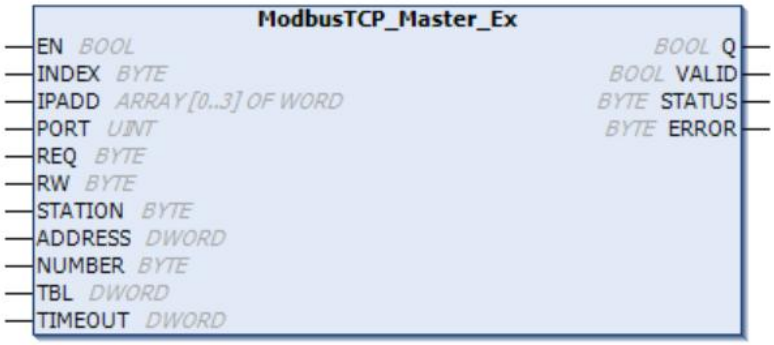
编程语言 **程序**

梯形图（LD）	
结构化文本（ST）	ModbusTCP_Master_0(EN:= TRUE, INDEX:= INDEX, IPADD:= IPADD, REQ:= REQ, RW:= RW, STATION:= STATION, ADDRESS:= ADDRESS, NUMBER:= NUMBER, TBL:= ADR(TBL), TIMEOUT:= TIMEOUT, Q=> , VALID=> VALID, STATUS=> STATUS, ERROR=> ERROR);

注意：在 1 个程序中最多可以有 20 个实例。

3.25.5. ModbusTCP_Master_Ex——以太网口 Modbus-TCP 主站功能块

以太网口 Modbus-TCP 主站功能块指令 ModbusTCP_Master 在库文件中的图示表达和相关参数说明如下：



输入参数	数据类型	功能描述	参数值说明
EN	BOOL	使能	0：关闭 socket 1：打开 socket 连接 IP 地址 （上升沿触发，后续操作保持高电平）
INDEX	BYTE	套接字（socket）ID	0~19
IPADD	ARRAY	从站 IP 地址	数组变量定义，如 [192.168.1.10] 中的

	[0..3] OF WORD		'192.168.1.10'为 IP 地址
PORT	UINT	目标端口号	0~65535
REQ	BYTE	请求连接	0：无效 1：高电平有效
RW	BYTE	读/写选择	0：读数据 1：写数据
STATION	BYTE	单元标识符	从站设备地址：0~247
ADDRESS	DWORD	从站存放数据地址的首地址	首地址采用标准 Modbus-TCP 驱动的地址填写方式，即采用六位数（十进制 xxxxxx），其中十万位（十进制）可以选择为 0、1、3、4，分别是 0 代表从站开关量输出（读写位 bit 寄存器），1 代表开关量输入（只读位 bit 寄存器），3 代表模拟量输入（只读字 WORD 寄存器）、4 代表模拟量输出（读写字 WORD 寄存器），从站数据地址为后五位（十进制，个位数到万位数，范围是 0~65535）。 例如： 000001 表示从站 Modbus 地址为 00001 的开关量输出点，405050 表示从站 Modbus 地址为 05050 的模拟量输出点。
NUMBER	BYTE	数据长度	范围：0~120 （位 bit 长度或字 WORD 长度） 特殊说明： 如果设定 NUMBER 为 0，在开关量（位 Bit）写操作时，主站发送报文通讯功能码为 05（十进制）；如果 NUMBER 为 1，在开关量（位 Bit）写操作时，主站发送报文通讯功能码为 15（十进制）； 如果设定 NUMBER 为 0，在模拟量（字 WORD）写操作时，主站发送报文通讯功能码为 06（十进制）；如果 NUMBER 为 1，在模拟量（字 WORD）写操作时，主站发送报文通讯功能码为 16（十进制）。
TBL	DWORD	PLC 主站存放数据地址的绝对地址	变量或 M/N 区寄存器绝对物理地址，通过 ADR 指令进行获取。 举例： （1）变量声明：ARR_01 AT %MW0: ARRAY [1..100] OF WORD; 取变量绝对地址：ADR（ARR_01）; 取 M 区绝对地址：ADR（%MW0）。 （2）变量声明：ARR_02AT %MD0: ARRAY [1..100] OF REAL; 取变量绝对地址：ADR（ARR_02）; 取 M 区绝对地址：ADR（%MD0）。

TIMEOUT	DWORD	超时时间（ms）	>50，打开连接超时时间与接收数据超时时间
输出参数	**数据类型**	**功能描述**	**参数值说明**
Q	BOOL	打开连接操作结果	0：未完成 1：完成 （使能 EN 为 1，打开连接操作结果 Q 为 1，错误代码 ERROR 为 0，则连接服务端 IP 地址完成）
VALID	BOOL	获取有效数据	0：无效 1：有效 （使能 EN 为 1，打开连接操作结果 Q 为 1，错误代码 ERROR 为 0，请求连接 REQ 为 1，则单次获取有效数据完成）
STATUS	BYTE	操作状态	0：关闭套接字（socket） 1：打开套接字（socket） 2：建立连接 3：连接 IP 地址 4：参数检查 5：组帧 6：超时计数 7：发送数据 8：接收状态 9：接收有效数据 10：接收错误 11：连接被关闭
ERROR	BYTE	错误代码	错误代码的参数值说明需结合操作状态的参数值，详细说明如下表

STATUS-操作状态	**ERROR-错误代码**	**参数值说明**	**内部后续动作**
0	0	关闭套接字（socket）	
1	0	打开套接字（socket）	
	其它值	详见附录 A	自动关闭套接字，延时 1 秒，重新打开套接字
2	0	设置套接字（socket）参数	
	其它值	详见附录 A	自动关闭套接字，延时 1 秒，重新打开套接字
3	0	连接 IP 地址	
	其它值	详见附录 A	自动关闭套接字，延时 1 秒，重新打开套接字
4	0	功能块参数检查	
	Bit0	数据长度超限	
	Bit1	地址超限	

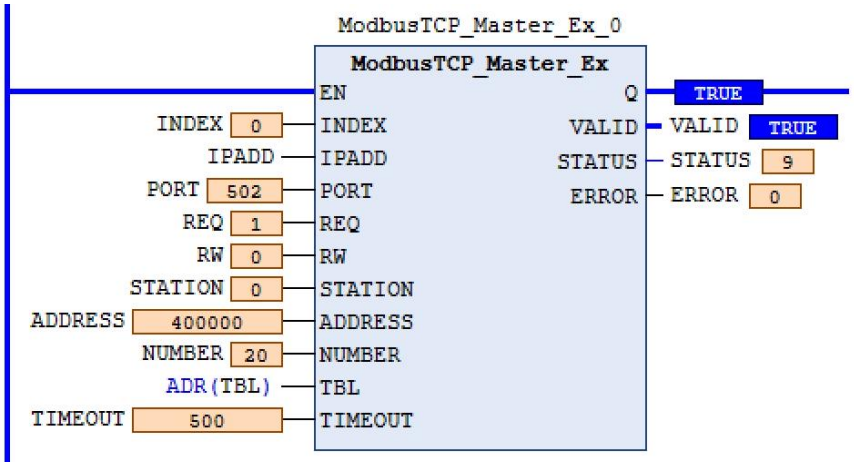
	Bit2	无效功能码	
	Bit3	无效 TBL 值	
5	0	组帧	
6	0	发送请求	
	1	发送请求超时	
7	其它值	发送请求错误，详见附录 A	自动关闭套接字，延时 1 秒，重新打开套接字
8	其它值	接收状态查询，详见附录 A	自动关闭套接字，延时 1 秒，重新打开套接字
9	0	接收到有效应答	
	Bit0	MBAP 帧头或功能码错误	
	Bit1	数据长度错误	
	Bit2	数据错误	
10	其它值	接收错误，详见附录 A	自动关闭套接字，延时 1 秒，重新打开套接字
11	1	对方关闭连接	自动关闭套接字，延时 1 秒，重新打开套接字

变量定义

	类别	名称	地址	数据类型	初值	注释	属性
1	VAR	ModbusTCP_Master_Ex_0		ModbusTCP_Master_Ex			
2	VAR	INDEX		BYTE	0		
3	VAR	IPADD		ARRAY [0..3] OF WORD	[192, 168, 1, 130]		
4	VAR	PORT		UINT	502		
5	VAR	REQ		BYTE	1		
6	VAR	RW		BYTE	0		
7	VAR	STATION		BYTE	0		
8	VAR	ADDRESS		DWORD	400000		
9	VAR	NUMBER		BYTE	20		
10	VAR	TBL		ARRAY[1..20] OF WORD			
11	VAR	TIMEOUT		DWORD	500		
12	VAR	VALID		BOOL			
13	VAR	STATUS		BYTE			
14	VAR	ERROR		BYTE			

编程语言 | **程序**

梯形图（LD）



结构化文本（ST）	ModbusTCP_Master_Ex_0(EN:= TRUE, INDEX:= INDEX, IPADD:= IPADD, PORT:= PORT, REQ:= REQ, RW:= RW, STATION:= STATION, ADDRESS:= ADDRESS, NUMBER:= NUMBER, TBL:= ADR(TBL), TIMEOUT:= TIMEOUT, Q=> , VALID=> VALID, STATUS=> STATUS, ERROR=> ERROR);

注意：在 1 个程序中最多可以有 20 个实例。

3.25.6. FreeTCP_Open——以太网口自由协议链接打开

以太网口自由协议链接打开指令 FreeTCP_Open 在库文件中的图示表达和相关参数说明如下：

输入参数	数据类型	功能描述	参数值说明
EN	BOOL	使能	0：无效 1：打开 socket （上升沿触发，开启本地端口，连接远程 IP 和远程端口）
INDEX	BYTE	套接字（socket）ID	无效
IPADD	ARRAY [0..3] OF WORD	目标 IP 地址	数组变量定义，如 [192.168.1.10] 中的 '192.168.1.10'为 IP 地址
L_PORT	WORD	本地端口号	0~65535
R_PORT	WORD	目标端口号	0~65535
TIMEOUT	DWORD	超时时间（ms）	>50，超时时间内，链接打开未完成，指令报错
输出参数	**数据类型**	**功能描述**	**参数值说明**
Q	BOOL	操作结果	0：未完成 1：完成 （使能 EN 为 1，错误代码 ERROR 为 0，则链接打开完成）

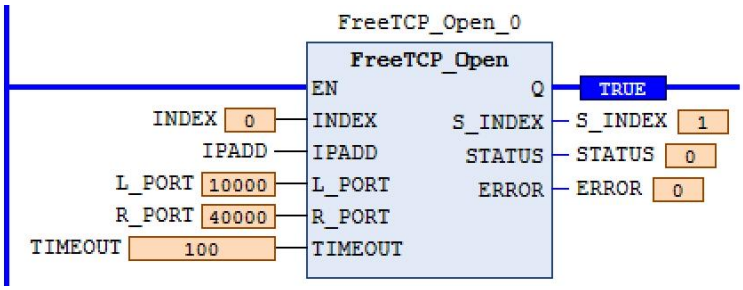
S_INDEX	BYTE	套接字（socket）ID	1-15
STATUS	BYTE	操作状态	0：查找可使用套接字（socket） 1：打开套接字（socket） 2：设置没有阻碍 5：连接远端主机
ERROR	BYTE	错误代码	错误代码的参数值说明需结合操作状态的参数值，详细说明如下表

STATUS-操作状态	ERROR-错误代码	参数值说明
0	0	查找可使用套接字（socket）
	1	无可使用的套接字（socket）
1	0	打开套接字（socket）
	其它值	详见附录 A
2	0	设置没有阻碍
	其它值	详见附录 A
3	0	绑定本地端口
	其它值	详见附录 A
4	0	连接远端主机
	其它值	详见附录 A
5	0	获取端口 r/w 状态
	其它值	详见附录 A
6	0	设置挂起错误命令
	其它值	详见附录 A
7	0	显示挂起错误
	其它值	详见附录 A

变量定义

	类别	名称	地址	数据类型	初值	注释	属性
1	VAR	FreeTCP_Open_0		FreeTCP_Open			
2	VAR	INDEX		BYTE			
3	VAR	IPADD		ARRAY [0..3] OF WORD	[192, 168, 1, 130]		
4	VAR	L_PORT		WORD	10000		
5	VAR	R_PORT		WORD	40000		
6	VAR	TIMEOUT		DWORD	100		
7	VAR	S_INDEX		BYTE			
8	VAR	STATUS		BYTE			
9	VAR	ERROR		BYTE			

编程语言	程序

梯形图 （LD）	
结构化文本 （ST）	FreeTCP_Open_0(EN:=TRUE, INDEX:=INDEX, IPADD:=IPADD , L_PORT:=L_PORT , R_PORT:=R_PORT , TIMEOUT:=TIMEOUT , Q=> , S_INDEX=>S_INDEX , STATUS=>STATUS , ERROR=>ERROR); (*开启 socket 和本地端口并连接目标 IP 和端口*)

注意：在 1 个程序中最多可以有 15 个实例。

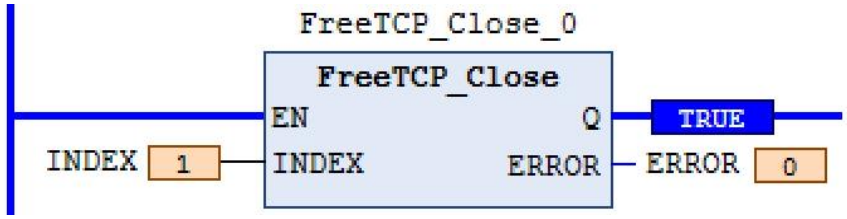
3.25.7. FreeTCP_Close——以太网口自由协议链接关闭

以太网口自由协议链接关闭指令 FreeTCP_Close 在库文件中的图示表达和相关参数说明如下：

输入参数	数据类型	功能描述	参数值说明
EN	BOOL	使能	0：无效 1：关闭 socket （上升沿触发，关闭本地端口，断开连接）
INDEX	BYTE	套接字（socket）ID	1~15
输出参数	**数据类型**	**功能描述**	**参数值说明**
Q	BOOL	操作结果	0：未完成 1：完成
ERROR	BYTE	错误代码	0：无错误 255：无效的接口或索引 其它值：详见附录 A

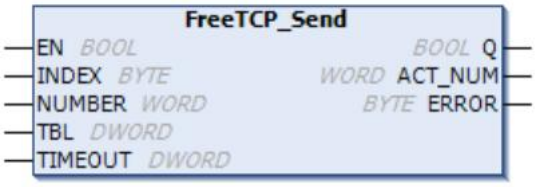
变量定义

	类别	名称	地址	数据类型	初值	注释
1	VAR	FreeTCP_Close_0		FreeTCP_Close		
2	VAR	INDEX		BYTE	0	
3	VAR	ERROR		BYTE		

编程语言	程序
梯形图（LD）	
结构化文本（ST）	FreeTCP_Close_0(EN:=TRUE, INDEX:=INDEX, Q=> , ERROR=>ERROR); (*关闭 socket 并断开连接*)

3.25.8. FreeTCP_Send——以太网口自由协议通讯数据发送

设置以太网口自由协议通讯数据发送指令 FreeTCP_Send 在库文件中的图示表达和相关参数说明如下：

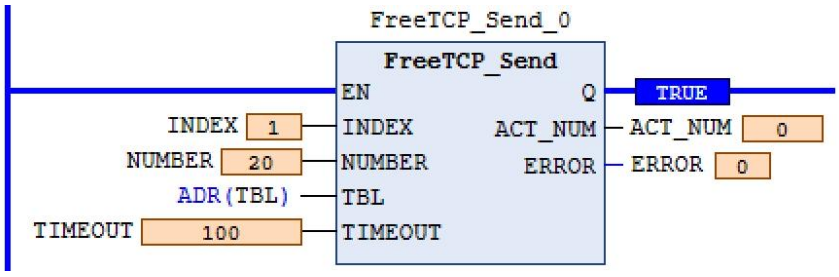


输入参数	数据类型	功能描述	参数值说明
EN	BOOL	使能	0：无效 1：发送数据 （上升沿触发）
INDEX	BYTE	套接字（socket）ID	1~15
NUMBER	WORD	数据长度	≤1460（字节数量）
TBL	DWORD	PLC 存放发送数据的绝对地址	变量或 M/N 区寄存器绝对物理地址，通过 ADR 指令进行获取。 举例： （1）变量声明：ARR_01 AT %MW0: ARRAY [1..100] OF WORD; 取变量绝对地址：ADR（ARR_01）; 取 M 区绝对地址：ADR（%MW0）。 （2）变量声明：ARR_02AT %MD0: ARRAY [1..100] OF REAL; 取变量绝对地址：ADR（ARR_02）; 取 M 区绝对地址：ADR（%MD0）。
TIMEOUT	DWORD	发送超时时间（ms）	>50，发送超时时间内，发送数据未完成，指令报错
输出参数	**数据类型**	**功能描述**	**参数值说明**
Q	BOOL	操作结果	0：未完成 1：完成 （使能 EN 为 1，发送的实际字节数 ACT_NUM >0，错误代码 ERROR 为 0，

			则单次发送数据完成）
ACT_NUM	WORD	发送的实际字节数	
ERROR	BYTE	错误代码	0：无错误 1：无效数据地址 2：无效套接字（socket） 255：发送数据超时 其它值：详见附录 A

变量定义

	类别	名称	地址	数据类型	初值	注释	属性
1	VAR	FreeTCP_Send_0		FreeTCP_Send			
2	VAR	INDEX		BYTE	1		
3	VAR	NUMBER		WORD	40		
4	VAR	TBL		ARRAY[1..20] OF WORD			
5	VAR	TIMEOUT		DWORD	100		
6	VAR	ACT_NUM		WORD			
7	VAR	ERROR		BYTE			

编程语言	程序
梯形图（LD）	
结构化文本（ST）	FreeTCP_Send_0(EN:=TRUE, INDEX:=INDEX, NUMBER:=NUMBER , TBL:=ADR(TBL), TIMEOUT:= TIMEOUT , Q=> , ACT_NUM=>ACT_NUM , ERROR=>ERROR); (*发送数据*)

3.25.9. FreeTCP_Receive——以太网口自由协议通讯数据接收

设置以太网口自由协议通讯数据接收指令 FreeTCP_Receive 在库文件中的图示表达和相关参数说明如下：

输入参数	数据类型	功能描述	参数值说明
EN	BOOL	使能	0：无效

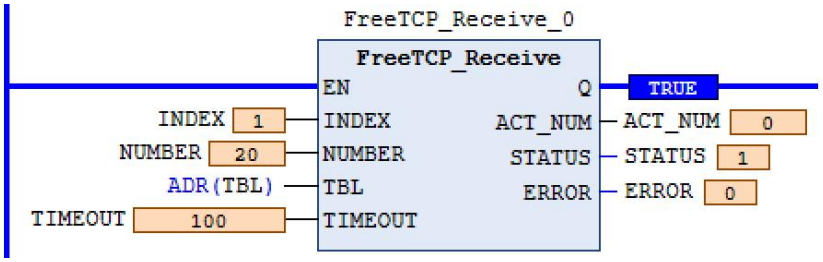
			1：接收数据 （上升沿触发）
INDEX	BYTE	套接字（socket）ID	1~15
NUMBER	WORD	数据长度	≤1460（字节数量）
TBL	DWORD	PLC 存放接收数据的绝对地址	变量或 M/N 区寄存器绝对物理地址，通过 ADR 指令进行获取。 举例： （1）变量声明：ARR_01 AT %MW0: ARRAY [1..100] OF WORD; 取变量绝对地址：ADR（ARR_01）; 取 M 区绝对地址：ADR（%MW0）。 （2）变量声明：ARR_02AT %MD0: ARRAY [1..100] OF REAL; 取变量绝对地址：ADR（ARR_02）; 取 M 区绝对地址：ADR（%MD0）。
TIMEOUT	DWORD	接收超时时间（ms）	>50 ，接收超时时间内，接收数据未完成，指令报错
输出参数	**数据类型**	**功能描述**	**参数值说明**
Q	BOOL	操作结果	0：未完成 1：完成 （使能 EN 为 1，接收的实际字节数 ACT_NUM >0，错误代码 ERROR 为 0，则单次接收数据完成）
ACT_NUM	WORD	接收的实际字节数	
STATUS	BYTE	操作状态	0：指令未运行 1：检查接口 2：第一次检查状态 3：第一次接收 4：检查状态 5：接收 6：服务器关闭 7：接收超时
ERROR	BYTE	错误代码	错误代码的参数值说明需结合操作状态的参数值，详细说明如下表

STATUS-操作状态	**ERROR-错误代码**	**参数值说明**
0	0	指令未运行
1	0	检查套接字（socket）
	1	无效表地址
	2	无效套接字（socket）
2	0	第一次检查状态

	其它值	详见附录 A
3	0	第一次接收
	其它值	详见附录 A
4	0	检查状态
	其它值	详见附录 A
5	0	接收
	其它值	详见附录 A
6	1	服务器关闭
7	1	接收超时

变量定义

	类别	名称	地址	数据类型	初值	注释	属性
1	VAR	FreeTCP_Receive_0		FreeTCP_Receive			
2	VAR	INDEX		BYTE	1		
3	VAR	NUMBER		WORD	40		
4	VAR	TBL		ARRAY[1..20] OF WORD			
5	VAR	TIMEOUT		DWORD	100		
6	VAR	ACT_NUM		WORD			
7	VAR	STATUS		BYTE			
8	VAR	ERROR		BYTE			

编程语言	程序
梯形图（LD）	
结构化文本（ST）	FreeTCP_Receive_0(EN:=TRUE, INDEX:=INDEX, NUMBER:=NUMBER , TBL:= ADR(TBL) , TIMEOUT:=TIMEOUT , Q=> , ACT_NUM=>ACT_NUM , STATUS=>STATUS , ERROR=>ERROR); (*接收数据*)

3.25.10. FreeUDP_Open——UDP 单播链接打开

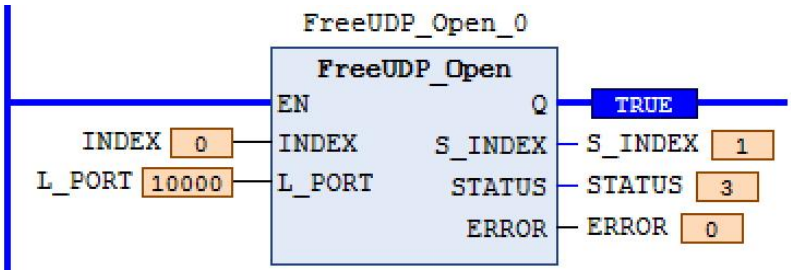
UDP 单播链接打开指令 FreeUDP_Open 在库文件中的图示表达和相关参数说明如下：

输入参数	数据类型	功能描述	参数值说明
EN	BOOL	使能	0：无效 1：打开 socket （上升沿触发，开启本地端口）
INDEX	BYTE	套接字（socket）ID	无效
L_PORT	WORD	本地端口号	0~65535
输出参数	**数据类型**	**功能描述**	**参数值说明**
Q	BOOL	操作结果	0：未完成 1：完成 （使能 EN 为 1，错误代码 ERROR 为 0，则链接打开完成）
S_INDEX	BYTE	套接字（socket）ID	1~15
STATUS	BYTE	操作状态	0：查找可使用套接字（socket） 1：打开套接字（socket） 2：设置没有阻碍 3：绑定本地端口
ERROR	BYTE	错误代码	错误代码的参数值说明需结合操作状态的参数值，详细说明如下表

STATUS-操作状态	ERROR-错误代码	参数值说明
0	0	查找可使用套接字（socket）
	1	无可使用的套接字（socket）
1	0	打开套接字（socket）
	其它值	详见附录 A
2	0	设置没有阻碍
	其它值	详见附录 A
3	0	绑定本地端口
	其它值	详见附录 A
4	0	连接远端主机
	其它值	详见附录 A

变量定义

	类别	名称	地址	数据类型	初值	注释	属性
1	VAR	FreeUDP_Open_0		FreeUDP_Open			
2	VAR	INDEX		BYTE			
3	VAR	L_PORT		WORD	10000		
4	VAR	S_INDEX		BYTE			
5	VAR	STATUS		BYTE			
6	VAR	ERROR		BYTE			

编程语言	程序
梯形图（LD）	
结构化文本（ST）	FreeUDP_Open_0(EN:=TRUE , INDEX:=INDEX, L_PORT:=L_PORT , Q=> , S_INDEX=>S_INDEX , STATUS=>STATUS , ERROR=>ERROR); (*开启 socket 和本地端口*)

注意：在 1 个程序中最多可以有 15 个实例。

3.25.11. FreeUDP_Open_Ex——UDP 单播链接打开

UDP 单播链接打开指令 FreeUDP_Open_Ex 在库文件中的图示表达和相关参数说明如下：

输入参数	数据类型	功能描述	参数值说明
EN	BOOL	使能	0：无效 1：打开 socket （上升沿触发，开启本地端口）
INDEX	BYTE	套接字（socket）ID	无效
L_IPADD	ARRAY [0..3] OF WORD	IP 地址	设定 UDP 单播发送数据，接收端接收的网口 IP 地址，数组变量定义，如[192.168.1.10]中的'192.168.1.10'为 IP 地址

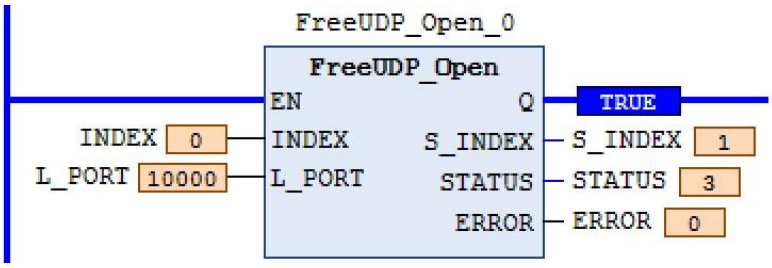
L_PORT	WORD	本地端口号	0~65535
输出参数	**数据类型**	**功能描述**	**参数值说明**
Q	BOOL	操作结果	0：未完成 1：完成 （使能 EN 为 1，错误代码 ERROR 为 0，则链接打开完成）
S_INDEX	BYTE	套接字（socket）ID	1~15
STATUS	BYTE	操作状态	0：找到可使用套接字（socket） 1：打开套接字（socket） 2：设置没有阻碍 3：绑定本地端口
ERROR	BYTE	错误代码	错误代码的参数值说明需结合操作状态的参数值，详细说明如下表

STATUS-操作状态	**ERROR-错误代码**	**参数值说明**
0	0	查找可使用套接字（socket）
	1	无可使用的套接字（socket）
1	0	打开套接字（socket）
	其它值	详见附录 A
2	0	设置没有阻碍
	其它值	详见附录 A
3	0	绑定本地端口
	其它值	详见附录 A
4	0	连接远端主机
	其它值	详见附录 A

变量定义

	类别	名称	地址	数据类型	初值	注释	属性
1	VAR	FreeUDP_Open_0		FreeUDP_Open			
2	VAR	INDEX		BYTE			
3	VAR	L_PORT		WORD	10000		
4	VAR	S_INDEX		BYTE			
5	VAR	STATUS		BYTE			
6	VAR	ERROR		BYTE			

编程语言	**程序**

梯形图 （LD）	
结构化文本 （ST）	FreeUDP_Open_0(EN:=TRUE , INDEX:=INDEX, L_PORT:=L_PORT , Q=> , S_INDEX=>S_INDEX , STATUS=>STATUS , ERROR=>ERROR); (*开启 socket 和本地端口*)

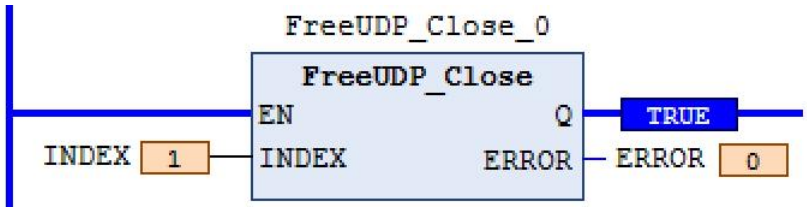
注意：在 1 个程序中最多可以有 15 个实例。

3.25.12. FreeUDP_Close——UDP 单播链接关闭

UDP 单播套接字关闭指令 FreeUDP_Close 在库文件中的图示表达和相关参数说明如下：

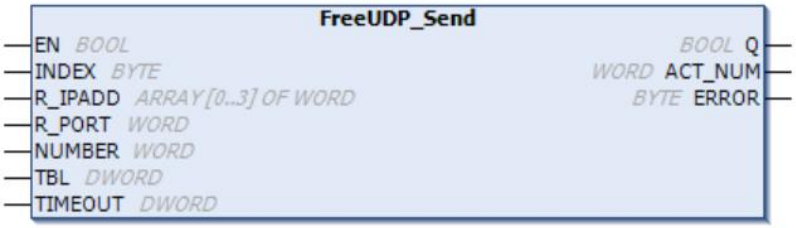
输入参数	数据类型	功能描述	参数值说明
EN	BOOL	使能	0：无效 1：关闭 socket （上升沿触发，关闭本地端口）
INDEX	BYTE	套接字（socket）ID	1~15
输出参数	**数据类型**	**功能描述**	**参数值说明**
Q	BOOL	操作结果	0：未完成 1：完成 （使能 EN 为 1，错误代码 ERROR 为 0，则链接关闭完成）
ERROR	BYTE	错误代码	0：无错误 255：无效套接字（socket） 其它值：详见附录 A

变量定义						
	类别	名称	地址	数据类型	初值	注释	属性
1	VAR	**FreeUDP_Close_0**		FreeUDP_Close			
2	VAR	**INDEX**		BYTE	1		
3	VAR	**ERROR**		BYTE			
编程语言	**程序**						

梯形图（LD）	
结构化文本（ST）	FreeUDP_Close_0(EN:=TRUE , INDEX:=INDEX, Q=> , ERROR=>ERROR); (*关闭 socket*)

3.25.13. FreeUDP_Send——UDP 单播通讯数据发送

设置 UDP 单播通讯数据发送指令 FreeUDP_Send 在库文件中的图示表达和相关参数说明如下：



输入参数	数据类型	功能描述	参数值说明
EN	BOOL	使能	0：无效 1：发送数据 （上升沿触发）
INDEX	BYTE	套接字（socket）ID	1~15
R_IPADD	ARRAY [0..3] OF WORD	目标 IP 地址	数组变量定义，如[192.168.1.10]中的'192.168.1.10'为 IP 地址
R_PORT	WORD	目标端口号	0~65535
NUMBER	WORD	数据长度	≤1460（字节数量）
TBL	DWORD	PLC 存放发送数据的绝对地址	变量或 M/N 区寄存器绝对物理地址，通过 ADR 指令进行获取。 举例： （1）变量声明：ARR_01 AT %MW0: ARRAY [1..100] OF WORD; 取变量绝对地址：ADR（ARR_01）； 取 M 区绝对地址：ADR（%MW0）。 （2）变量声明：ARR_02AT %MD0: ARRAY [1..100] OF REAL; 取变量绝对地址：ADR（ARR_02）； 取 M 区绝对地址：ADR（%MD0）。
TIMEOUT	DWORD	发送超时时间（ms）	>50，发送超时时间内，发送数据未完成，指令报错
输出参数	**数据类型**	**功能描述**	**参数值说明**
Q	BOOL	操作结果	0：未完成

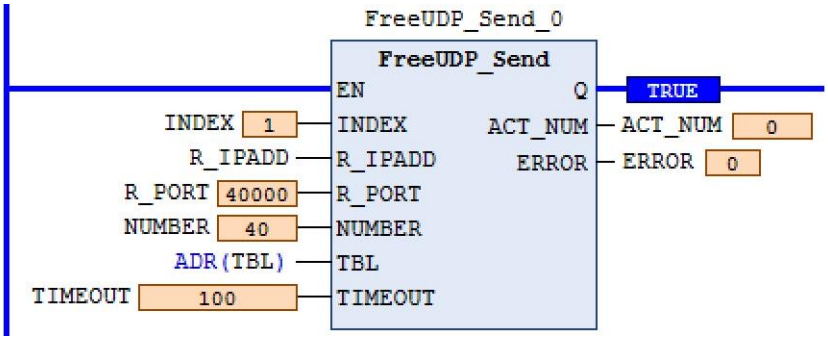
			1：完成 （使能 EN 为 1，发送的实际字节数 ACT_NUM >0，错误代码 ERROR 为 0，则单次发送数据完成）
ACT_NUM	WORD	发送的实际字节数	
ERROR	BYTE	错误代码	0：无错误 1：无效数据地址 2：无效套接字（socket） 254：网线未连接 255：发送数据超时 其它值：详见附录 A

变量定义

	类别	名称	地址	数据类型	初值	注释	属性
1	VAR	FreeUDP_Send_0		FreeUDP_Send			
2	VAR	INDEX		BYTE	1		
3	VAR	R_IPADD		ARRAY [0..3] OF WORD	[192, 168, 1, 130]		
4	VAR	R_PORT		WORD	40000		
5	VAR	NUMBER		WORD	40		
6	VAR	TBL		ARRAY[1..20] OF WORD			
7	VAR	TIMEOUT		DWORD	100		
8	VAR	ACT_NUM		WORD			
9	VAR	ERROR		BYTE			

编程语言 | **程序**

梯形图（LD）

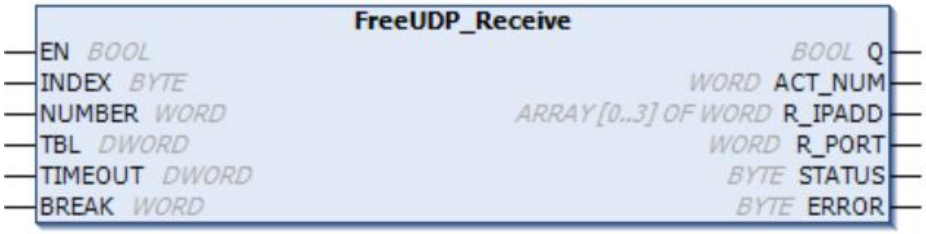


结构化文本（ST）

```
FreeUDP_Send_0(
EN:=TRUE ,
INDEX:=INDEX,
R_IPADD:= R_IPADD,
R_PORT:= R_PORT,
NUMBER:= NUMBER ,
TBL:= ADR(TBL) ,
TIMEOUT:= TIMEOUT ,
Q=> ,
ACT_NUM=>ACT_NUM ,
ERROR=>ERROR ); (*发送数据*)
```

3.25.14. FreeUDP_Receive——UDP 单播通讯数据接收

设置 UDP 单播通讯数据接收指令 FreeUDP_Receive 在库文件中的图示表达和相关参数说明如下：



输入参数	数据类型	功能描述	参数值说明
EN	BOOL	使能	0：无效 1：接收数据 （上升沿触发）
INDEX	BYTE	套接字（socket）ID	1~15
NUMBER	WORD	数据长度	≤1460（字节数量）
TBL	DWORD	PLC 存放接收数据的绝对地址	变量或 M/N 区寄存器绝对物理地址，通过 ADR 指令进行获取。 举例： （1）变量声明：ARR_01 AT %MW0: ARRAY [1..100] OF WORD; 取变量绝对地址：ADR（ARR_01）; 取 M 区绝对地址：ADR（%MW0）。 （2）变量声明：ARR_02AT %MD0: ARRAY [1..100] OF REAL; 取变量绝对地址：ADR（ARR_02）; 取 M 区绝对地址：ADR（%MD0）。
TIMEOUT	DWORD	接收超时时间（ms）	>50，接收超时时间内，接收数据未完成，指令报错
BREAK	WORD	数据接收完成判断时间（ms）	无效，默认设置为 0
输出参数	**数据类型**	**功能描述**	**参数值说明**
Q	BOOL	操作结果	0：未完成 1：完成 （使能 EN 为 1，接收的实际字节数 ACT_NUM >0，错误代码 ERROR 为 0，则单次接收数据完成）
ACT_NUM	WORD	接收的实际字节数	
R_IPADD	ARRAY [0..3] OF WORD	目标 IP 地址	数组变量定义，如 [192.168.1.20] 中的 '192.168.1.20'为 IP 地址
R_PORT	WORD	目标端口号	0~65535
STATUS	BYTE	操作状态	0：指令未运行 1：检查接口

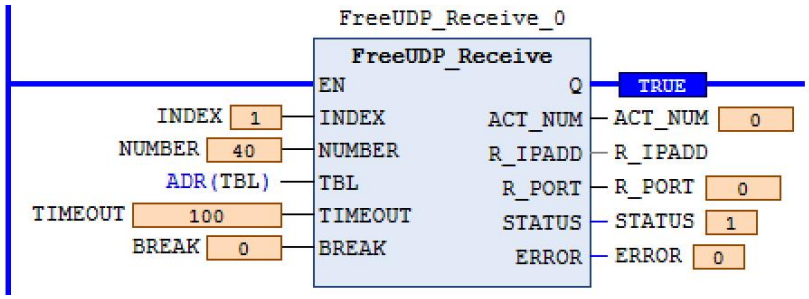
			2：第一次检查状态 3：第一次接收 4：检查状态 5：接收 6：服务器关闭 7：接收超时
ERROR	BYTE	错误代码	错误代码的参数值说明需结合操作状态的参数值，详细说明如下表

STATUS-操作状态	ERROR-错误代码	参数值说明
0	0	指令未运行
1	0	检查套接字（socket）
	1	无效表地址
	2	无效套接字（socket）
2	0	第一次检查状态
	其它值	详见附录 A
3	0	第一次接收
	其它值	详见附录 A
4	0	检查状态
	其它值	详见附录 A
5	0	接收
	其它值	详见附录 A
6	1	服务器关闭
7	1	接收超时

变量定义

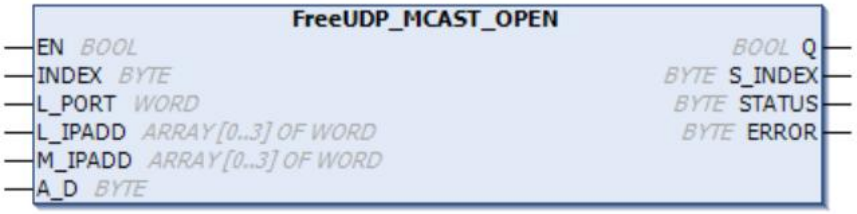
	类别	名称	地址	数据类型	初值	注释	属性
1	VAR	FreeUDP_Receive_0		FreeUDP_Receive			
2	VAR	INDEX		BYTE	1		
3	VAR	NUMBER		WORD	40		
4	VAR	TBL		ARRAY[1..20] OF WORD			
5	VAR	TIMEOUT		DWORD	100		
6	VAR	BREAK		WORD	0		
7	VAR	ACT_NUM		WORD			
8	VAR	R_IPADD		ARRAY [0..3] OF WORD			
9	VAR	R_PORT		WORD			
10	VAR	STATUS		BYTE			
11	VAR	ERROR		BYTE			

编程语言	程序

梯形图（LD）	
结构化文本（ST）	FreeUDP_Receive_0(EN:=TRUE , INDEX:=INDEX, NUMBER:=NUMBER , TBL:= ADR(TBL) , TIMEOUT:=TIMEOUT , BREAK:=BREAK , Q=> , ACT_NUM=>ACT_NUM , R_IPADD=>R_IPADD , R_PORT=>R_PORT , STATUS=>STATUS , ERROR=>ERROR); (*接收数据*)

3.25.15. FreeUDP_MCAST_Open——UDP 组播链接打开

UDP 组播链接打开指令 FreeUDP_MCAST_Open 在库文件中的图示表达和相关参数说明如下：



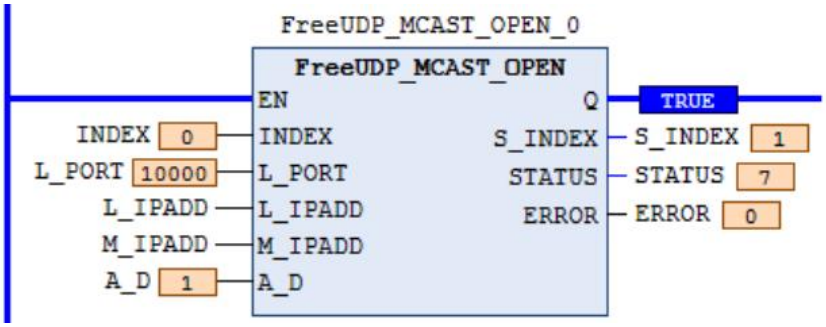
输入参数	数据类型	功能描述	参数值说明
EN	BOOL	使能	0：无效 1：打开 socket （上升沿触发，开启本地端口）
INDEX	BYTE	套接字（socket）ID	无效
L_PORT	WORD	本地端口号	0~65535
L_IPADD	ARRAY [0..3] OF WORD	IP 地址	打开 UDP 组播链接的 CPU 网口当前 IP 地址，数组变量定义，如 [192.168.1.10] 中的'192.168.1.10'为 IP 地址
M_IPADD	ARRAY [0..3] OF WORD	组播 IP 地址	数组变量定义，如[224.0.1.1]中的'224.0.1.1'为组播 IP 地址
A_D	BYTE	功能添加/删除	1，不允许设置为其他值（0：

			删除；1 添加；）
输出参数	**数据类型**	**功能描述**	**参数值说明**
Q	BOOL	操作结果	0：未完成 1：完成 （使能 EN 为 1，错误代码 ERROR 为 0，则链接打开完成）
S_INDEX	BYTE	套接字（socket）ID	1~15
STATUS	BYTE	操作状态	0：找到可使用套接字（socket） 1：打开套接字（socket） 2：设置没有阻碍 3：设置网口 IP 地址 4：设置本地端口 5：设置接收缓冲区 6：设置组播 IP 地址 7：绑定本地端口号
ERROR	BYTE	错误代码	错误代码的参数值说明需结合操作状态的参数值，详细说明如下表

STATUS-操作状态	**ERROR-错误代码**	**参数值说明**
0	0	查找可使用套接字（socket）
	1	无可使用的套接字（socket）
1	0	打开套接字（socket）
	其它值	详见附录 A
2	0	设置没有阻碍
	其它值	详见附录 A
3	0	设置网口 IP 地址
	其它值	详见附录 A
4	0	设置本地端口
	其它值	详见附录 A
5	0	设置接收缓冲区
	其它值	详见附录 A
6	0	设置组播 IP 地址
	其它值	详见附录 A
7	0	绑定本地端口号
	其它值	详见附录 A

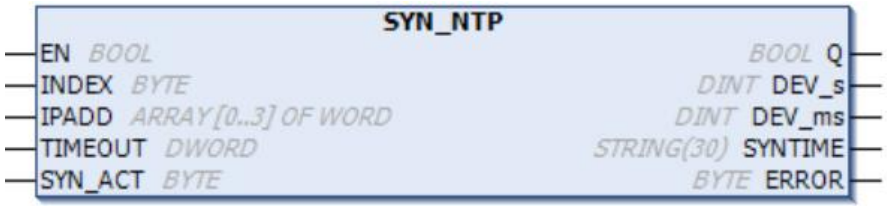
变量定义

	类别	名称	地址	数据类型	初值	注释	属性
1	VAR	FreeUDP_MCAST_OPEN_0		FreeUDP_MCAST_OPEN			
2	VAR	INDEX		BYTE			
3	VAR	L_PORT		WORD	10000		
4	VAR	L_IPADD		ARRAY [0..3] OF WORD	[192, 168, 1, 4]		
5	VAR	M_IPADD		ARRAY [0..3] OF WORD	[230, 220, 10, 21]		
6	VAR	A_D		BYTE	1		
7	VAR	S_INDEX		BYTE			
8	VAR	STATUS		BYTE			
9	VAR	ERROR		BYTE			

编程语言	程序
梯形图（LD）	
结构化文本（ST）	FreeUDP_MCAST_OPEN_0(EN:=TRUE, INDEX:=INDEX , L_PORT:=L_PORT , L_IPADD:=L_IPADD , M_IPADD:=M_IPADD , A_D:=A_D , Q=> , S_INDEX=>S_INDEX , STATUS=>STATUS , ERROR=>ERROR); (*开启 socket 和本地端口*)

3.25.16. SYN_NTP——NTP 同步时钟

NTP 同步时钟指令 SYN_NTP 在库文件中的图示表达和相关参数说明如下：



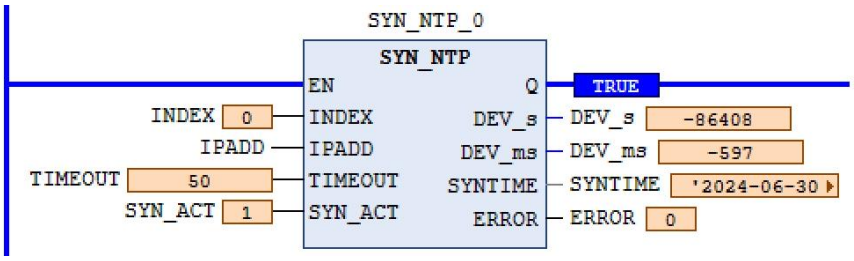
输入参数	数据类型	功能描述	参数值说明
EN	BOOL	使能	0：无效 1：上升沿使能
INDEX	BYTE	套接字（socket）ID	无效
IPADD	ARRAY [0..3] OF WORD	NTP 服务器的 IP 地址	数组变量定义，例如[192.168.1.10]中的'192.168.1.10'

TIMEOUT	DWORD	同步时钟超时时间	单元 ms，小于 100
SYN_ACT	BYTE	是否同步到系统时钟	0：不同步到 RTC 时钟 其它：同步到 RTC 时钟
输出参数	**数据类型**	**功能描述**	**参数值说明**
Q	BOOL	操作结果	0：未完成 1：完成 （使能 EN 为 1，错误代码 ERROR 为 0，则单次 NTP 时钟同步完成）
DEV_s	DINT	NTP 同步时钟与系统时钟的秒差值	单位 s，当 NTP 同步的时钟大于系统时钟，差值为正数，否则为负数
DEV_ms	DINT	NTP 同步时钟与系统时钟的毫秒差值	单位 ms，范围-999~999，当 NTP 同步的时钟毫秒值大于系统时钟，差值为正数，否则为负数
SYNTIME	STRING(30)	系统时间值	指令上升沿使能后无错误修改一次当前值，例如字符串的值'2023-04-24 13-05-21 01'，代表 NTP 同步时钟的值为 2023 年 4 月 24 日 13 时 15 分 21 秒星期一
ERROR	BYTE	错误代码	0：无错误 1：打开套接字建立连接失败 2：绑定端口号失败 3：发送 NTP 请求报文失败 4：套接字读取状态查询失败 5：套接字数据接收错误 6：指令操作超时 7：套接字数据接收异常

变量定义

	类别	名称	地址	数据类型	初值	注释	属性
1	VAR	SYN_NTP_0		SYN_NTP			
2	VAR	INDEX		BYTE			
3	VAR	IPADD		ARRAY [0..3] OF WORD	[192, 168, 1, 130]		
4	VAR	TIMEOUT		DWORD	50		
5	VAR	SYN_ACT		BYTE	1		
6	VAR	DEV_s		DINT			
7	VAR	DEV_ms		DINT			
8	VAR	SYNTIME		STRING(30)			
9	VAR	ERROR		BYTE			

编程语言	**程序**

梯形图 （LD）	
结构化文本 （ST）	SYN_NTP_0(EN:= TRUE, INDEX:=INDEX, IPADD:=IPADD, TIMEOUT:=TIMEOUT, SYN_ACT:=SYN_ACT, Q=> , DEV_s=>DEV_s, DEV_ms=>DEV_ms, SYNTIME=>SYNTIME, ERROR=>ERROR);

注意：在一个程序中只能有 1 个实例。

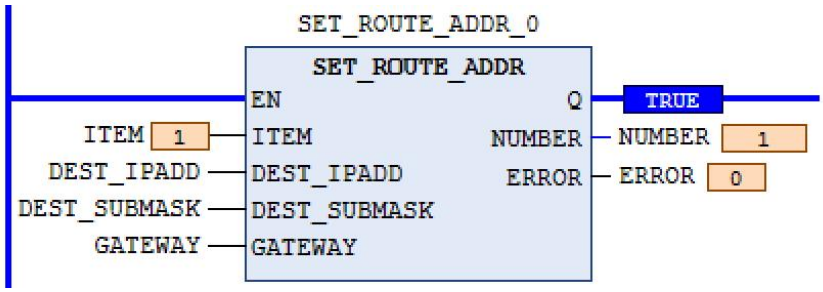
3.25.17. SET_ROUTE_ADDR——设置当前网络路由地址

设置当前网络路由地址指令 SET_ROUTE_ADDR 在库文件中的图示表达和相关参数说明如下：

输入参数	数据类型	功能描述	参数值说明
EN	BOOL	使能	0：无效 1：上升沿使能
ITEM	BYTE	网络配置选项	0：默认路由 1：静态路由
DEST_IPADD	ARRAY [0..3] OF WORD	路由 IP 地址	数组变量定义，如[192.168.1.10]中的'192.168.1.10'为 IP 地址
DEST_SUBMASK	ARRAY [0..3] OF WORD	路由子网掩码	数组变量定义，如[255.255.255.0]中的'255.255.255.0'为子网掩码
GATEWAY	ARRAY [0..3] OF WORD	路由网关	数组变量定义，如[192.168.1.1]中的'192.168.1.1'为网关
输出参数	**数据类型**	**功能描述**	**参数值说明**
Q	BOOL	操作结果	0：未完成 1：完成
NUMBER	WORD	实例数量	
ERROR	BYTE	错误代码	0：无错误 1:无效的函数参数或内存分配错误

变量定义

	类别	名称	地址	数据类型	初值	注释	属性
1	VAR	SET_ROUTE_ADDR_0		SET_ROUTE_ADDR			
2	VAR	ITEM		BYTE	1		
3	VAR	DEST_IPADD		ARRAY [0..3] OF WORD	[192, 168, 1, 10]		
4	VAR	DEST_SUBMASK		ARRAY [0..3] OF WORD	[3(255), 0]		
5	VAR	GATEWAY		ARRAY [0..3] OF WORD	[192, 168, 10, 1]		
6	VAR	NUMBER		WORD			
7	VAR	ERROR		BYTE			

编程语言	程序
梯形图（LD）	
结构化文本（ST）	SET_ROUTE_ADDR_0(EN:=TRUE, ITEM:=ITEM , DEST_IPADD:= DEST_IPADD , DEST_SUBMASK:= DEST_SUBMASK , GATEWAY:=GATEWAY , Q=> , NUMBER=>NUMBER , ERROR=>ERROR);

3.25.18. GET_ROUTE_ADDR——获取当前网络路由地址

获取当前网络路由地址指令 GET_ROUTE_ADDR 在库文件中的图示表达和相关参数说明如下：

输入参数	数据类型	功能描述	参数值说明
EN	BOOL	使能	0：无效 1：上升沿使能
ITEM	BYTE	网络配置选项	0：默认路由 1：静态路由
INDEX	BYTE	实例的索引	1~n
输出参数	**数据类型**	**功能描述**	**参数值说明**
Q	BOOL	操作结果	0：未完成

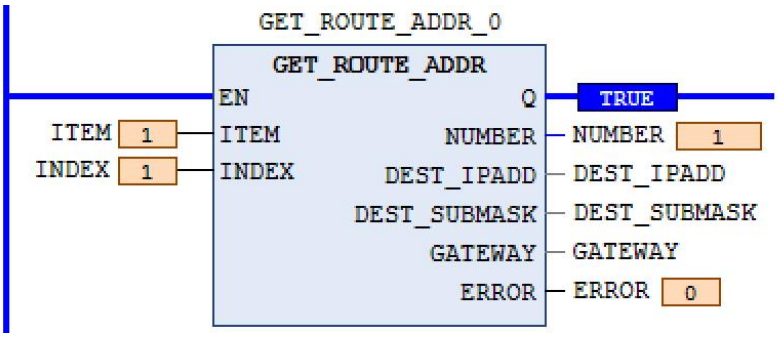
			1：完成
NUMBER	WORD	实例数量	
DEST_IPADD	ARRAY [0..3] OF WORD	路由 IP 地址	数组变量定义，如[192.168.1.10]中的'192.168.1.10'为 IP 地址
DEST_SUBMASK	ARRAY [0..3] OF WORD	路由子网掩码	数组变量定义，如[255.255.255.0]中的'255.255.255.0'为子网掩码
GATEWAY	ARRAY [0..3] OF WORD	路由网关	数组变量定义，如[192.168.1.1]中的'192.168.1.1'为网关
ERROR	BYTE	错误代码	0：无错误 1：无效的参数或无效网络配置选项

变量定义

	类别	名称	地址	数据类型	初值	注释	属性
1	VAR	GET_ROUTE_ADDR_0		GET_ROUTE_ADDR			
2	VAR	ITEM		BYTE	1		
3	VAR	INDEX		BYTE	1		
4	VAR	DEST_IPADD		ARRAY [0..3] OF WORD			
5	VAR	DEST_SUBMASK		ARRAY [0..3] OF WORD			
6	VAR	GATEWAY		ARRAY [0..3] OF WORD			
7	VAR	NUMBER		WORD			
8	VAR	ERROR		BYTE			

编程语言 | **程序**

梯形图（LD）

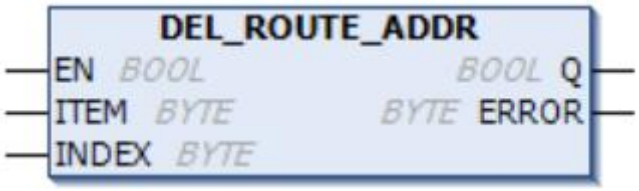


结构化文本（ST）

```
GET_ROUTE_ADDR_0 (
EN:=TRUE,
ITEM:=ITEM ,
INDEX:=INDEX ,
Q=> ,
NUMBER=>NUMBER ,
DEST_IPADD=> DEST_IPADD ,
DEST_SUBMASK=> DEST_SUBMASK ,
GATEWAY=>GATEWAY ,
ERROR=>ERROR );
```

3.25.19. DEL_ROUTE_ADDR——清除当前网络路由地址

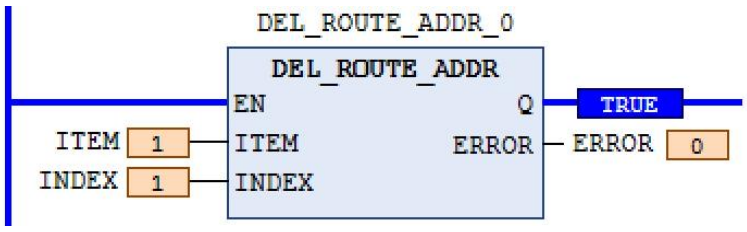
清除当前网络路由地址指令 DEL_ROUTE_ADDR 在库文件中的图示表达和相关参数说明如下：



输入参数	数据类型	功能描述	参数值说明
EN	BOOL	使能	0：无效 1：上升沿使能
ITEM	BYTE	网络配置选项	0：默认路由 1：静态路由
INDEX	BYTE	实例的索引	1~n
输出参数	**数据类型**	**功能描述**	**参数值说明**
Q	BOOL	操作结果	0：未完成 1：完成
ERROR	BYTE	错误代码	0：无错误 1：无效的函数参数

变量定义

	类别	名称	地址	数据类型	初值	注释	属性
1	VAR	DEL_ROUTE_ADDR_0		DEL_ROUTE_ADDR			
2	VAR	ITEM		BYTE	1		
3	VAR	INDEX		BYTE	1		
4	VAR	ERROR		BYTE			

编程语言	程序
梯形图（LD）	
结构化文本（ST）	DEL_ROUTE_ADDR(EN:=TRUE, ITEM:=ITEM , INDEX:=INDEX , Q=> , ERROR=>ERROR);

3.26. 冗余相关指令（CmpRPC3000-library）

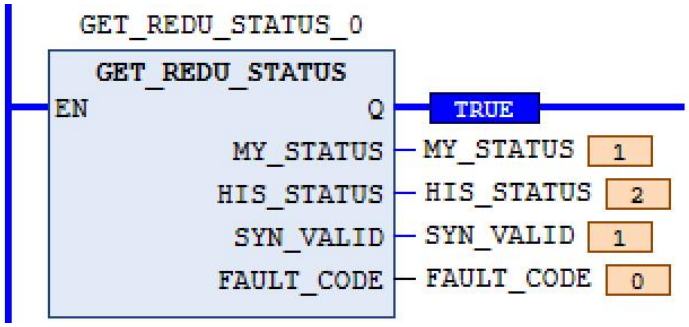
3.26.1. GET_REDU_STATUS——获取冗余状态

获取冗余状态指令 GET_REDU_STATUS 在库文件中的图示表达和相关参数说明如下：

输入参数	数据类型	功能描述	参数值说明
EN	BOOL	使能	0：无效 1：有效
输出参数	**数据类型**	**功能描述**	**参数值说明**
Q	BOOL	操作结果	0：未完成 1：完成
MY_STATUS	BYTE	当前 CPU 主备状态	1：主 CPU 2：备用 CPU 0：无
HIS_STATUS	BYTE	冗余对方 CPU 主备状态	1：主 CPU 2：备用 CPU 0：非冗余状态
SYN_VALID	BYTE	数据同步状态	0：未同步 1：正在同步
FAULT_CODE	BYTE	最后一次冗余操作错误码	0：无错误 其它：有错误

变量定义

	类别	名称	地址	数据类型	初值	注释	特性
1	VAR	GET_REDU_STATUS_0		GET_REDU_STATUS			
2	VAR	MY_STATUS		BYTE			
3	VAR	HIS_STATUS		BYTE			
4	VAR	SYN_VALID		BYTE			
5	VAR	FAULT_CODE		BYTE			

编程语言	程序
梯形图（LD）	
结构化文本（ST）	GET_REDU_STATUS_0 (EN:= TRUE, Q=>, MY_STATUS => MY_STATUS,

	HIS_STATUS => HIS_STATUS, SYN_VALID => SYN_VALID, SYN_VALID => SYN_VALID);

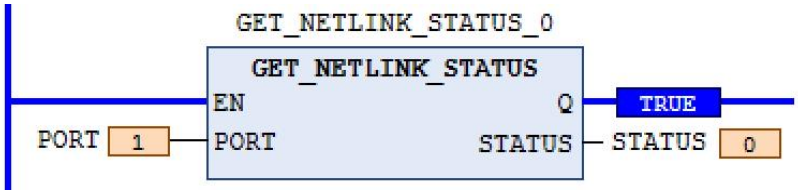
3.26.2. GET_NETWORK_STATUS——获取网线连接状态

获取网线连接状态指令 GET_NETWORK_STATUS 在库文件中的图示表达和相关参数说明如下：

输入参数	数据类型	功能描述	参数值说明
EN	BOOL	使能	0：无效 1：高电平有效
PORT	BYTE	以太网网口硬件编号	1：以太网网口 1 2：以太网网口 2
输出参数	**数据类型**	**功能描述**	**参数值说明**
Q	BOOL	操作结果	0：未完成 1：完成
STATUS	BYTE	网线连接状态	0：有连接 1：无连接

变量定义

	类别	名称	地址	数据类型	初值	注释	属性
1	VAR	GET_NETLINK_STATUS_0		GET_NETLINK_STATUS			
2	VAR	PORT		BYTE	1		
3	VAR	STATUS		BYTE			

编程语言	程序
梯形图（LD）	
结构化文本（ST）	GET_NETLINK_STATUS_0 (EN:=TRUE, PORT:=PORT, Q=>, STATUS => STATUS);

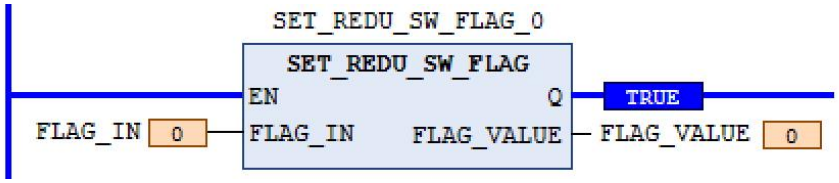
3.26.3. SET_REDU_SW_FLAG——设置冗余切换开关标志

设置冗余切换开关标志指令 SET_REDU_SW_FLAG 在库文件中的图示表达和相关参数说明如下：

输入参数	数据类型	功能描述	参数值说明
EN	BOOL	使能	0：无效 1：上升沿使能
FLAG_IN	BYTE	冗余切换开关标志	0 或 1，根据主备 CPU 的冗余切换开关标志的数值来切换主备 CPU。 例如：通过主备 CPU 的网口网线连接情况切换主备 CPU，0 代表网口网线连接中，1 代表网口网线未连接。 主 CPU 的标志为 0，备用 CPU 的标志为 1，不切换；主 CPU 的标志为 0，备用 CPU 的标志为 0，不切换；主 CPU 的标志为 1，备用 CPU 的标志为 1，不切换；主 CPU 的标志为 1，备用 CPU 的标志为 0，主备 CPU 切换。
输出参数	**数据类型**	**功能描述**	**参数值说明**
Q	BOOL	操作结果	0：未完成 1：完成
FLAG_VALUE	BYTE	主从切换标志当前值	

变量定义

	类别	名称	地址	数据类型	初值	注释	属性
1	VAR	SET_REDU_SW_FLAG_0		SET_REDU_SW_FLAG			
2	VAR	FLAG_IN		BYTE			
3	VAR	FLAG_VALUE		BYTE			

编程语言	程序
梯形图（LD）	
结构化文本（ST）	SET_REDU_SW_FLAG_0 (EN:=TRUE, FLAG_IN:= FLAG_IN, Q=>, FLAG_VALUE => FLAG_VALUE);

3.26.4. ADD_MSYN_TABLE——添加 M 区冗余同步数据表

添加 M 区冗余同步数据表指令 ADD_MSYN_TABLE 在库文件中的图示表达和相关参数说明如下：

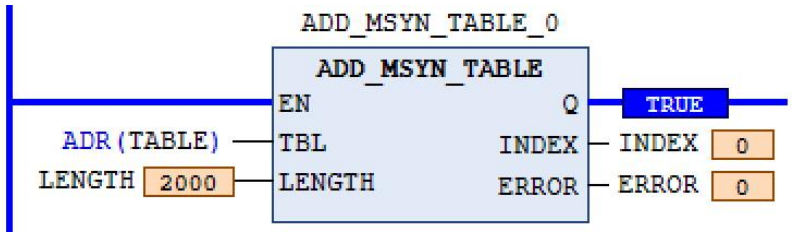
将 M 区的数据添加到同步数据表中，每个扫描周期同步一次，只对冗余主 CPU 有效。

输入参数	数据类型	功能描述	参数值说明
EN	BOOL	使能	0：无效 1：高电平有效
TBL	DWORD	PLC 主站存放数据地址的绝对地址	变量或 M/N 区寄存器绝对物理地址，通过 ADR 指令进行获取。 举例： （1）变量声明：ARR_01 AT %MW0: ARRAY [1..100] OF WORD; 取变量绝对地址：ADR（ARR_01）; 取 M 区绝对地址：ADR（%MW0）。 （2）变量声明：ARR_02AT %MD0: ARRAY [1..100] OF REAL; 取变量绝对地址：ADR（ARR_02）; 取 M 区绝对地址：ADR（%MD0）。
LENGTH	UINT	数据长度	≤2000（字节长度）
输出参数	**数据类型**	**功能描述**	**参数值说明**
Q	BOOL	操作结果	0：未完成 1：完成
INDEX	BYTE	同步数据表索引	≤255
ERROR	BYTE	错误代码	0：无错误 Bit0：无效数据表空间 Bit1：无效数据长度 Bit2：无效 TBL 值 Bit3：无可用表空间 （字节中的 Bit0~Bit3 分别代表不同故障，每个故障显示互相独立。）

变量定义

	类别	名称	地址	数据类型	初值	注释	属性
1	VAR	ADD_MSYN_TABLE_0		ADD_MSYN_TABLE			
2	VAR	TABLE	%MW1000	ARRAY[1..1000] OF WORD			
3	VAR	LENGTH		UINT	2000		
4	VAR	INDEX		BYTE			
5	VAR	ERROR		BYTE			

编程语言	程序

梯形图 （LD）	
结构化文本 （ST）	ADD_MSYN_TABLE(EN:=TRUE, TBL:= ADR(TABLE), LENGTH:=LENGTH, Q=> , INDEX=>INDEX, ERROR=>ERROR);

3.26.5. GET_SYS_REDUDIAG——获取冗余系统运行诊断信息

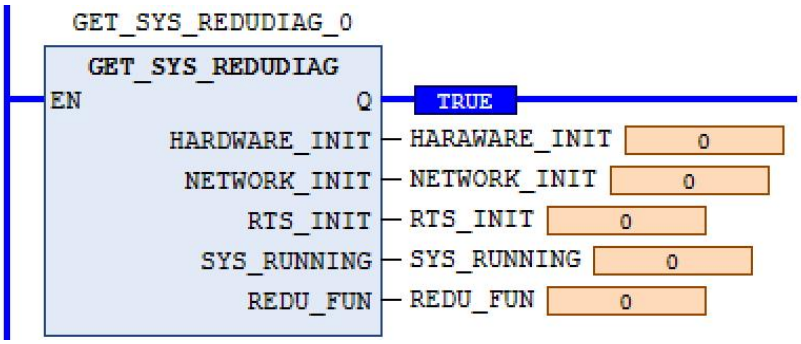
获取冗余系统运行诊断信息指令 GET_SYS_REDUDIAG 在库文件中的图示表达和相关参数说明如下：

输入参数	数据类型	功能描述	参数值说明
EN	BOOL	使能	0：无效 1：上升沿使能
输出参数	**数据类型**	**功能描述**	**参数值说明**
Q	BOOL	操作结果	0：未完成 1：完成
HARDWARE_INIT	UDINT	硬件初始化诊断信息	详见附录 B
NETWORK_INIT	UDINT	网络初始化诊断信息	详见附录 B
RTS_INIT	UDINT	运行时初始化诊断信息	详见附录 B
SYS_RUNNING	UDINT	系统运行诊断信息	详见附录 B
REDU_FUN	UDINT	冗余功能诊断信息	详见附录 B

变量定义

	类别	名称	地址	数据类型	初值	注释	属性
1	VAR	GET_SYS_REDUDIAG_0		GET_SYS_REDUDIAG			
2	VAR	HARDWARE_INIT		UDINT			
3	VAR	NETWORK_INIT		UDINT			
4	VAR	RTS_INIT		UDINT			
5	VAR	SYS_RUNNING		UDINT			
6	VAR	REDU_FUN		UDINT			

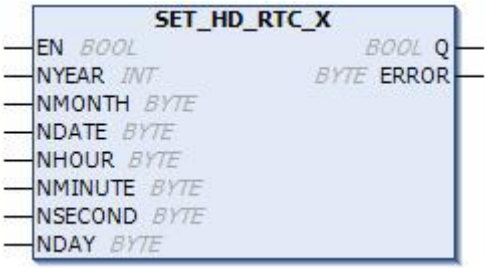
编程语言	程序

梯形图 （LD）	
结构化文本 （ST）	GET_SYS_REDUDIAG_0(EN:=TRUE, Q=> , HARDWARE_INIT=>HARDWARE_INIT, NETWORK_INIT=>NETWORK_INIT, RTS_INIT=>RTS_INIT, SYS_RUNNING=>SYS_RUNNING, REDU_FUN=>REDU_FUN);

3.27. 硬件设备指令（CmpRPC3000-library）

3.27.1. SET_HD_RTC_X——设置实时时钟（普通数据格式）

设置实时时钟（普通数据格式）指令 SET_HD_RTD_X 在库文件中的图示表达和相关参数说明如下：

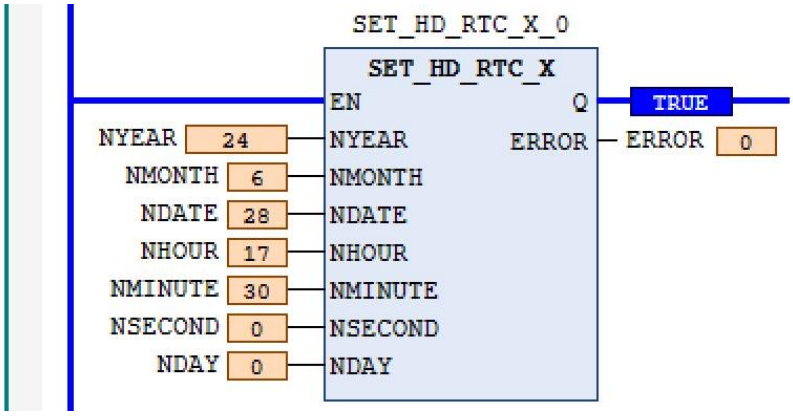


输入参数	数据类型	功能描述	参数值说明
EN	BOOL	使能	0：无效 1：上升沿使能
NYEAR	INT	年	0~99，设置年份为 2000
NMONTH	BYTE	月	1~12
NDATE	BYTE	日	1~31
NHOUR	BYTE	时	0~23
NMINUTE	BYTE	分	0~59
NSECOND	BYTE	秒	0~59
NDAY	BYTE	星期	0，通过设置日期自动计算星期值
输出参数	**数据类型**	**功能描述**	**参数值说明**
Q	BOOL	操作结果	0：未完成 1：完成 （使能 EN 为 1，操作结果 Q

			为 1，错误代码 ERROR 为 0，则时钟设置完成）
ERROR	BYTE	错误代码	0：没有错误 Bit0：无效的年 Bit1：无效的月 Bit2：无效的日 Bit3：无效的星期 Bit4：无效的时 Bit5：无效的分 Bit6：无效的秒 Bit7：写入错误 （字节中的 Bit0~Bit7 分别代表不同故障，每个故障显示互相独立。）

变量定义

	类别	名称	地址	数据类型	初值	注释	特性
1	VAR	SET_HD_RTC_X_0		SET_HD_RTC_X			
2	VAR	NYEAR		INT			
3	VAR	NMONTH		BYTE			
4	VAR	NDATE		BYTE			
5	VAR	NHOUR		BYTE			
6	VAR	NMINUTE		BYTE			
7	VAR	NSECOND		BYTE			
8	VAR	NDAY		BYTE			
9	VAR	ERROR		BYTE			

编程语言	程序
梯形图（LD）	
结构化文本（ST）	SET_HD_RTC_X_0(EN:= TRUE, NYEAR:= NYEAR, NMONTH:= NMONTH, NDATE:= NDATE, NHOUR:= NHOUR, NMINUTE:= NMINUTE,

	NSECOND:= NSECOND, NDAY:= NDAY, Q=> , ERROR=> ERROR);

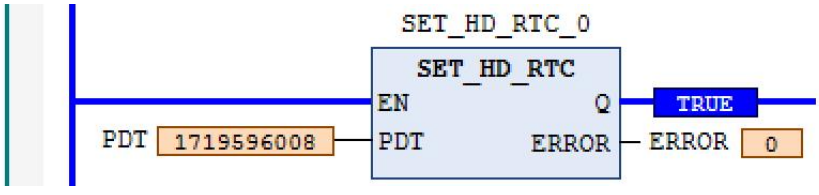
3.27.2. SET_HD_RTC——设置实时时钟（类格林威治时间格式）

设置实时时钟（类格林威治时间格式）指令 SET_HD_RTC 在库文件中的图示表达和相关参数说明如下：

输入参数	数据类型	功能描述	参数值说明
EN	BOOL	使能	0：无效 1：上升沿使能
PDT	DWORD	类格林威治时间	从 1970 年 1 月 1 日 12:00:00 开始的秒值（格林威治时间是从 1970 年 1 月 1 日 8:00:00 开始的秒值）
输出参数	**数据类型**	**功能描述**	**参数值说明**
Q	BOOL	操作结果	0：未完成 1：完成 （使能 EN 为 1，操作结果 Q 为 1，错误代码 ERROR 为 0，则时钟设置完成）
ERROR	BYTE	错误代码	0：没有错误 Bit7：写入错误

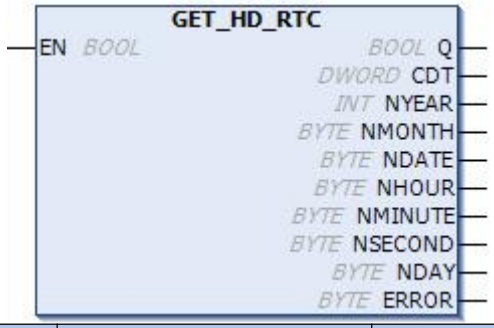
变量定义

	类别	名称	地址	数据类型	初值	注释	特性
1	VAR	**SET_HD_RTC_0**		SET_HD_RTC			
2	VAR	**PDT**		DWORD			
3	VAR	**ERROR**		BYTE			

编程语言	程序
梯形图（LD）	
结构化文本（ST）	SET_HD_RTC_0(EN:= TRUE, PDT:= PDT, Q=> , ERROR=> ERROR);

3.27.3. GET_HD_RTC——读取实时时钟日期、时间、星期

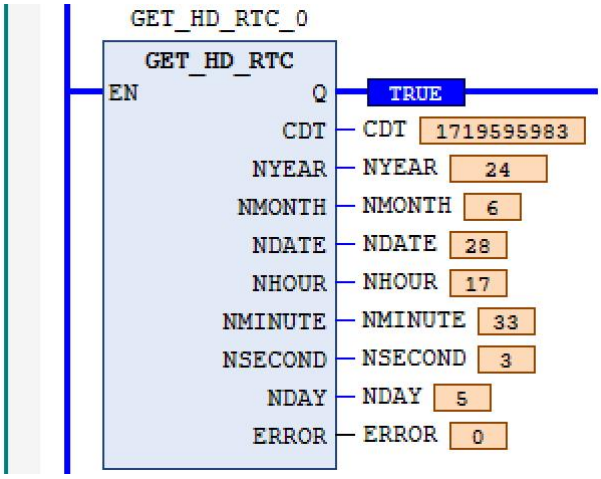
读取实时时钟日期、时间、星期指令 GET_HD_RTC 在库文件中的图示表达和相关参数说明如下：



输入参数	数据类型	功能描述	参数值说明
EN	BOOL	使能	0：无效 1：高电平有效
输出参数	**数据类型**	**功能描述**	**参数值说明**
Q	BOOL	操作结果	0：未完成 1：完成
CDT	DWORD	类格林威治时间	从 1970 年 1 月 1 日 12:00:00 开始的秒值（格林威治时间是从 1970 年 1 月 1 日 8:00:00 开始的秒值）
NYEAR	INT	年	0~99，取年份后两位值
NMONTH	BYTE	月	1~12
NDATE	BYTE	日	1~31
NHOUR	BYTE	时	0~24
NMINUTE	BYTE	分	0~60
NSECOND	BYTE	秒	0~60
NDAY	BYTE	星期	0~6，分别代表星期日-星期六
ERROR	BYTE	故障代码	0：无错误 1：操作异常 12：操作超时 13：内部未确认

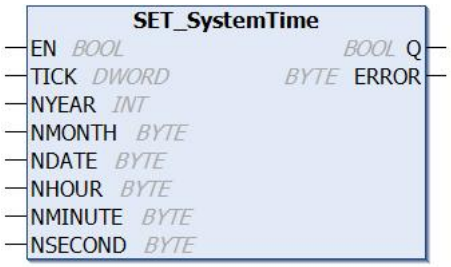
变量定义

	类别	名称	地址	数据类型	初值	注释	特性
1	VAR	GET_HD_RTC_0		GET_HD_RTC			
2	VAR	CDT		DWORD			
3	VAR	NYEAR		INT			
4	VAR	NMONTH		BYTE			
5	VAR	NDATE		BYTE			
6	VAR	NHOUR		BYTE			
7	VAR	NMINUTE		BYTE			
8	VAR	NSECOND		BYTE			
9	VAR	NDAY		BYTE			
10	VAR	ERROR		BYTE			

编程语言	程序
梯形图（LD）	
结构化文本（ST）	GET_HD_RTC_0(EN:= TRUE, Q=> , CDT=>CDT, NYEAR=> NYEAR, NMONTH=> NMONTH, NDATE=> NDATE, NHOUR=> NHOUR, NMINUTE=> NMINUTE, NSECOND=> NSECOND, NDAY=> NDAY, ERROR=> ERROR);

3.27.4. SET_SystemTime——设置系统时钟

设置系统时钟指令 SET_SystemTime 在库文件中的图示表达和相关参数说明如下：

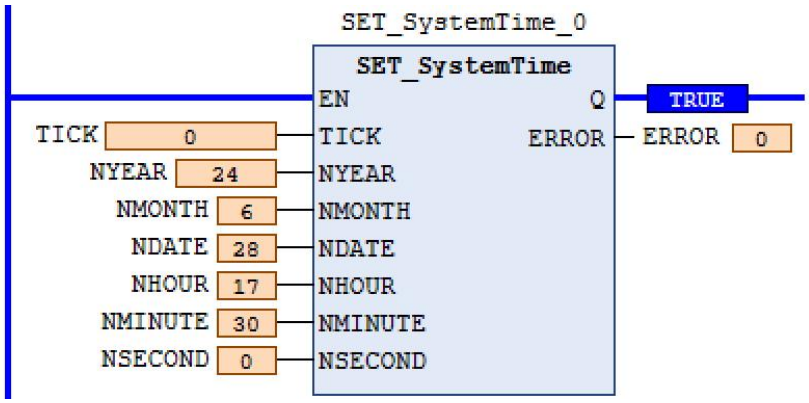


输入参数	数据类型	功能描述	参数值说明
EN	BOOL	使能	0：无效 1：上升沿使能
TICK	DWORD	系统刻度计数器	0-999，单位 ms，设置其他值无效
NYEAR	INT	年	0~99，设置年份为 2000
NMONTH	BYTE	月	1~12
NDATE	BYTE	日	1~31
NHOUR	BYTE	时	0~23
NMINUTE	BYTE	分	0~59
NSECOND	BYTE	秒	0~59
输出参数	**数据类型**	**功能描述**	**参数值说明**
Q	BOOL	操作结果	0：未完成 1：完成 （使能 EN 为 1，错误代码 ERROR 为 0，则时钟设置完成）
ERROR	BYTE	错误代码	0：没有错误 Bit0：无效的年 Bit1：无效的月 Bit2：无效的日 Bit3：无效的星期 Bit4：无效的时 Bit5：无效的分 Bit6：无效的秒 （字节中的 Bit0~Bit6 分别代表不同故障，每个故障显示互相独立。）

变量定义

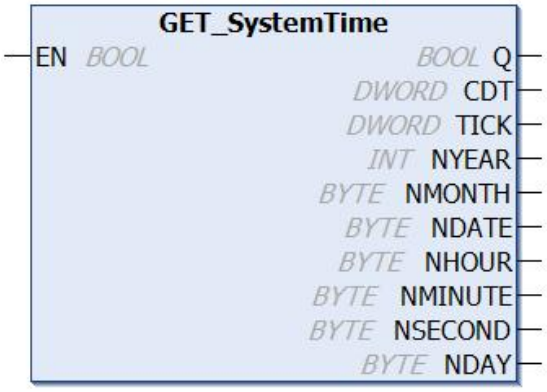
	类别	名称	地址	数据类型	初值	注释	属性
1	VAR	SET_SystemTime_0		SET_SystemTime			
2	VAR	TICK		DWORD			
3	VAR	NYEAR		INT			
4	VAR	NMONTH		BYTE			
5	VAR	NDATE		BYTE			
6	VAR	NHOUR		BYTE			
7	VAR	NMINUTE		BYTE			
8	VAR	NSECOND		BYTE			
9	VAR	ERROR		BYTE			

编程语言	程序

梯形图 （LD）	
结构化文本 （ST）	SET_SystemTime_0(EN:=TRUE TICK:= TICK, NYEAR:=NYEAR, NMONTH:=NMONTH, NDATE:=NDATE, NHOUR:=NHOUR, NMINUTE:=NMINUTE, NSECOND:=NSECOND, Q=> , ERROR=>ERROR);

3.27.5. GET_SystemTime——读取系统时钟

读取系统时钟指令 GET_SystemTime 在库文件中的图示表达和相关参数说明如下：



输入参数	数据类型	功能描述	参数值说明
EN	BOOL	使能	0：无效 1：高电平有效
输出参数	**数据类型**	**功能描述**	**参数值说明**
Q	BOOL	操作结果	0：未完成 1：完成
CDT	DWORD	类格林威治时间	从 1970 年 1 月 1 日 12:00:00 开始的秒值（格林威治时间是从 1970 年 1 月 1 日 8:00:00 开始的秒值）
TICK	DWORD	系统刻度计数器	0-999
NYEAR	INT	年	2000~2099

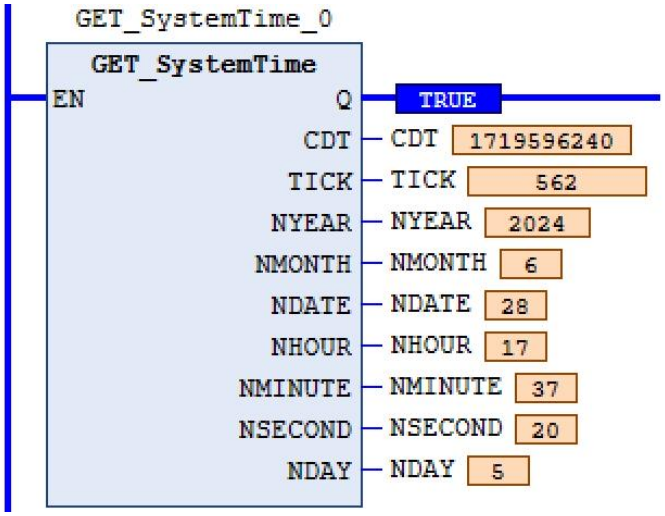
NMONTH	BYTE	月	1~12
NDATE	BYTE	日	1~31
NHOUR	BYTE	时	0~24
NMINUTE	BYTE	分	0~60
NSECOND	BYTE	秒	0~59
NDAY	BYTE	星期	0~6，分别代表星期日-星期六

变量定义

	类别	名称	地址	数据类型	初值	注释	属性
1	VAR	GET_SystemTime_0		GET_SystemTime			
2	VAR	CDT		DWORD			
3	VAR	TICK		DWORD			
4	VAR	NYEAR		INT			
5	VAR	NMONTH		BYTE			
6	VAR	NDATE		BYTE			
7	VAR	NHOUR		BYTE			
8	VAR	NMINUTE		BYTE			
9	VAR	NSECOND		BYTE			
10	VAR	NDAY		BYTE			

编程语言 | **程序**

梯形图（LD）

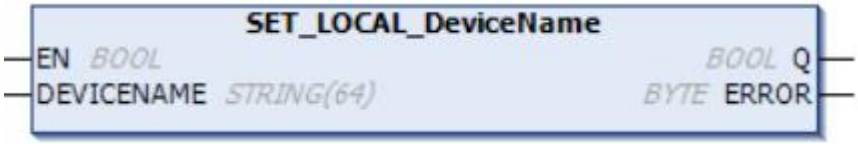


结构化文本（ST）

```
GET_SystemTime_0(
EN:=TRUE,
Q=> ,
CDT=>CDT,
TICK=>TICK,
NYEAR=>NYEAR,
NMONTH=>NMONTH,
NDATE=>NDATE,
NHOUR=>NHOUR,
NMINUTE=>NMINUTE,
NSECOND=>NSECOND,
NDAY=>NDAY);
```

3.27.6. SET_LOCAL_DeviceName——设置当前设备名称

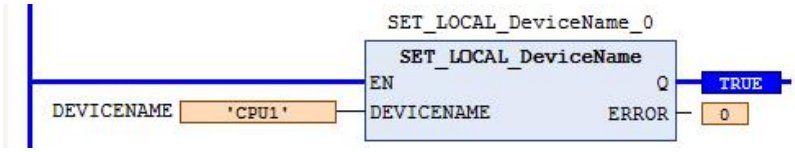
设置当前设备名称指令 SET_LOCAL_DeviceName 在库文件中的图示表达和相关参数说明如下：



输入参数	数据类型	功能描述	参数值说明
EN	BOOL	使能	0：无效 1：上升沿使能
DEVICENAME	STRING(64)	设备名称	字符串，长度≤64，例如定义变量'CPU1'
输出参数	**数据类型**	**功能描述**	**参数值说明**
Q	BOOL	操作结果	0：未完成 1：完成
ERROR	BYTE	故障代码	0：无错误 Bit0：无效的设备名称

变量定义

	类别	名称	地址	数据类型	初值	注释	属性
1	VAR	SET_LOCAL_DeviceName_0		SET_LOCAL_DeviceName			
2	VAR	DEVICENAME		STRING(64)	'CPU1'		

编程语言	程序
梯形图（LD）	
结构化文本（ST）	SET_LOCAL_DeviceName_0 (EN:= TRUE, DEVICENAME:= DEVICENAME , Q=>, ERROR=>);

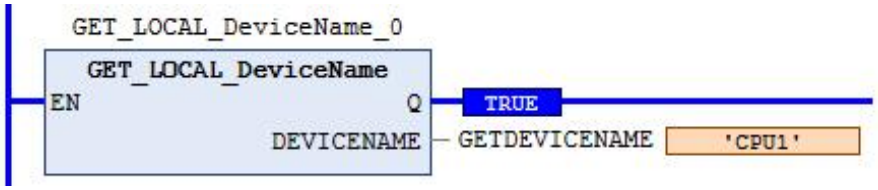
3.27.7. GET_LOCAL_DeviceName——获取当前设备名称

获取当前设备名称指令 GET_LOCAL_DeviceName 在库文件中的图示表达和相关参数说明如下：

输入参数	数据类型	功能描述	参数值说明
EN	BOOL	使能	0：无效 1：上升沿使能
输出参数	**数据类型**	**功能描述**	**参数值说明**

Q	BOOL	操作结果	0：未完成 1：完成
DEVICENAME	STRING(64)	设备名称	字符串，长度≤64，例如定义变量'CPU1'

变量定义						
	类别	名称	地址	数据类型	初值	注释	特性
1	VAR	GET_LOCAL_DeviceName_0		GET_LOCAL_DeviceName			
2	VAR	GETDEVICENAME		STRING(64)			

编程语言	程序
梯形图（LD）	
结构化文本（ST）	GET_LOCAL_DeviceName_0 (EN:= TRUE, Q=>, DEVICENAME => GETDEVICENAME);

3.27.8. GET_SYS_DIAG——获取系统运行诊断信息

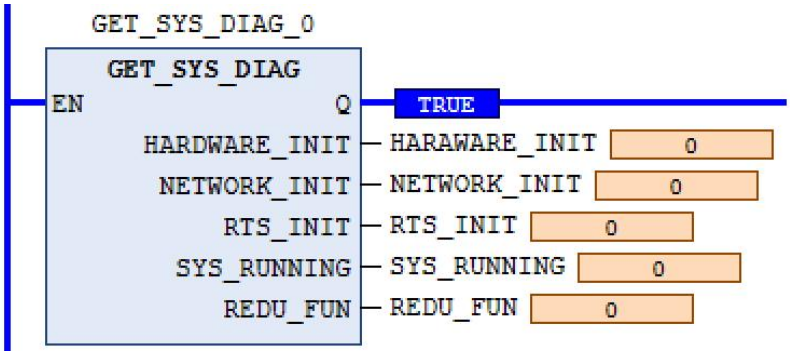
获取系统运行诊断信息指令 GET_SYS_DIAG 在库文件中的图示表达和相关参数说明如下：

输入参数	数据类型	功能描述	参数值说明
EN	BOOL	使能	0：无效 1：上升沿使能
输出参数	**数据类型**	**功能描述**	**参数值说明**
Q	BOOL	操作结果	0：未完成 1：完成
HARDWARE_INIT	UDINT	硬件初始化诊断信息	详见附录 C
NETWORK_INIT	UDINT	网络初始化诊断信息	详见附录 C
RTS_INIT	UDINT	运行时初始化诊断信息	详见附录 C

SYS_RUNNING	UDINT	系统运行诊断信息	详见附录 C
REDU_FUN	UDINT	冗余功能诊断信息	详见附录 C

变量定义

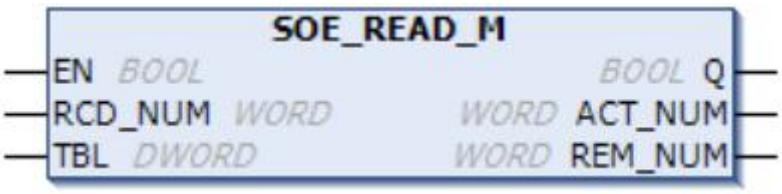
	类别	名称	地址	数据类型	初值	注释	属性
1	VAR	GET_SYS_DIAG_0		GET_SYS_DIAG			
2	VAR	HARDWARE_INIT		UDINT			
3	VAR	NETWORK_INIT		UDINT			
4	VAR	RTS_INIT		UDINT			
5	VAR	SYS_RUNNING		UDINT			
6	VAR	REDU_FUN		UDINT			

编程语言	程序
梯形图（LD）	
结构化文本（ST）	GET_SYS_DIAG_0(EN:=TRUE, Q=> , HARDWARE_INIT=>HARDWARE_INIT, NETWORK_INIT=>NETWORK_INIT, RTS_INIT=>RTS_INIT, SYS_RUNNING=>SYS_RUNNING, REDU_FUN=>REDU_FUN);

3.28. 特殊指令（RPCCommon.compiled-library）

3.28.1. SOE_READ_M——读取 SOE 数据记录态

读取 SOE 数据记录指令 SOE_READ_M 在库文件中的图示表达和相关参数说明如下：

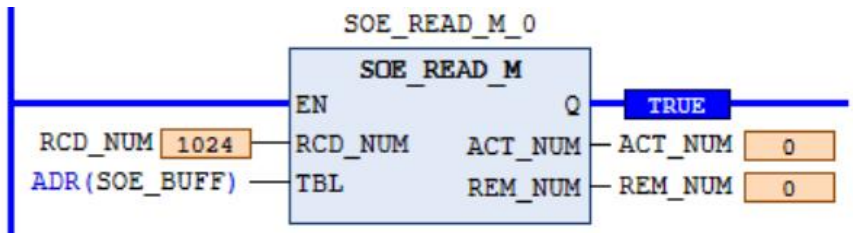


输入参数	数据类型	功能描述	参数值说明
EN	BOOL	使能	0：无效 1：上升沿使能，获取当前 SOE

			数据队列内的数据记录 1：高电平有效，显示当前 SOE 数据队列内未读取的数据记录
RCD_NUM	WORD	单次读取 SOE 数据记录数	≤1024
TBL	DWORD	读取 SOE 数据记录存放数据地址的绝对地址	变量绝对物理地址，数据类型为结构体，可以通过 ADR 指令进行获取。 举例： （1）变量声明：SOE_BUFF: ARRAY[0..1023] OF SOE_DATA; 取变量绝对地址：ADR（SOE_BUFF）。
输出参数	**数据类型**	**功能描述**	**参数值说明**
Q	BOOL	操作结果	0：未完成 1：完成
ACT_NUM	WORD	单次读取 SOE 数据记录数	≤1024
REM_NUM	WORD	未读取 SOE 数据记录数	使能 EN 为 1，显示当前 SOE 数据队列内未读取的数据记录

变量定义

	类别	名称	地址	数据类型	初值	注释	属性
1	VAR	SOE_READ_M_0		SOE_READ_M			
2	VAR	RCD_NUM		WORD	1024		
3	VAR	SOE_BUFF		ARRAY[0..1023] OF SOE_DATA			
4	VAR	ACT_NUM		WORD			
5	VAR	REM_NUM		WORD			

编程语言	**程序**
梯形图（LD）	
结构化文本（ST）	SOE_READ_M_0(EN:=TRUE, RCD_NUM:=RCD_NUM, TBL:=ADR(SOE_BUFF), Q=> , ACT_NUM=>ACT_NUM, REM_NUM=>REM_NUM);

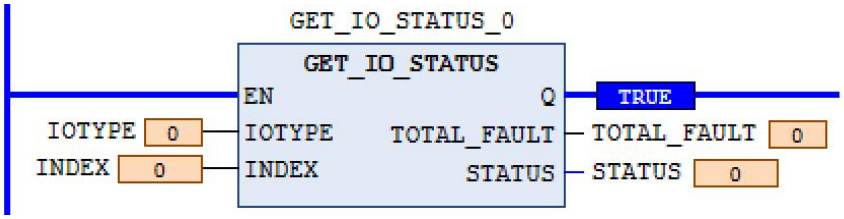
3.28.2. GET_IO_STATUS——读取指定模块状态

读取指定模块状态指令 GET_IO_STATUS 在库文件中的图示表达和相关参数说明如下：

输入参数	数据类型	功能描述	参数值说明
EN	BOOL	使能	0：无效 1：有效
IOTYPE	BYTE	模块类型	0：本地总线 IO 模块 1：高总线 IO 模块 2：高总线主站模块 3：高总线从站模块 4：远程总线 IO 模块 5：远程总线从站模块 6：通讯模块
INDEX	WORD	模块索引	<512
输出参数	**数据类型**	**功能描述**	**参数值说明**
Q	BOOL	操作结果	0：未完成 1：完成
TOTAL_FAULT	BOOL	总故障	
STATUS	BYTE	模块状态	0：没有错误 Bit0：模块离线 Bit1：配置参数错误 Bit2：通讯超时 Bit3：通信数据错误 Bit4：模块未就绪 Bit5：模块无配置参数 Bit6：背板总线 1 故障 Bit7：背板总线 2 故障 Bit8：CE 总线主/从卡 COM1 端口通讯故障 Bit9：CE 总线主/从卡 COM2 端口通讯故障 （字节中的 Bit0~Bit9 分别代表不同故障，每个故障显示互相独立。）

变量定义

	类别	名称	地址	数据类型	初值	注释	属性
1	VAR	**GET_IO_STATUS_0**		GET_IO_STATUS			
2	VAR	**IOTYPE**		BYTE			
3	VAR	**INDEX**		WORD			
4	VAR	**TOTAL_FAULT**		BYTE			
5	VAR	**STATUS**		WORD			

编程语言	程序
梯形图（LD）	
结构化文本（ST）	GET_IO_STATUS_0(EN:= TRUE, IOTYPE:= IOBYTE, INDEX:= INDEX, Q=> , TOTAL_FAULT=> TOTAL_FAULT, STATUS=> STATUS);

第四章 附录

4.1. 附录 A 通讯指令输出引脚 ERROR-错误代码部分故障码含义

ERROR-错误代码	故障码含义
6	设备未配置
9	文件描述符（套接字 socket）不可用
12	存储器分配错误
13	权限被拒绝
22	无效的参数
23	系统中存在过多已打开的文件
24	文件描述符表已满
35	非阻塞模式套接字（socket）被阻塞
36	请求已接收且被挂起
37	操作已在进行中
38	描述符不是一个套接字（socket）
40	DGRAM 类型的套接字（socket），发送的包超过最大包大小
41	套接字（socket）的协议类型错误
42	协议无效
44	套接字（socket）类型不支持
45	操作不支持
46	协议族不支持
48	地址已被使用
49	不能分配请求的地址
50	套接字（socket）操作失败，因为网络关闭
53	软件引起的连接终止
54	连接被对方复位
55	无足够空间处理相关请求
56	套接字（socket）已被连接
57	套接字（socket）未被连接
58	套接字（socket）已经关闭，不能发送
60	操作超时
61	连接被拒绝
64	主机已关闭
65	主机不可达，没有到主机的路由
255	其它错误

ERROR-错误代码	故障码含义
6	设备未配置
9	文件描述符（套接字 socket）不可用
12	存储器分配错误
13	权限被拒绝
22	无效的参数
23	系统中存在过多已打开的文件
24	文件描述符表已满
35	非阻塞模式套接字（socket）被阻塞
36	请求已接收且被挂起
37	操作已在进行中
38	描述符不是一个套接字（socket）
40	DGRAM 类型的套接字（socket），发送的包超过最大包大小
41	套接字（socket）的协议类型错误
42	协议无效
44	套接字（socket）类型不支持
45	操作不支持
46	协议族不支持
48	地址已被使用
49	不能分配请求的地址
50	套接字（socket）操作失败，因为网络关闭
53	软件引起的连接终止
54	连接被对方复位
55	无足够空间处理相关请求
56	套接字（socket）已被连接
57	套接字（socket）未被连接
58	套接字（socket）已经关闭，不能发送
60	操作超时
61	连接被拒绝
64	主机已关闭
65	主机不可达，没有到主机的路由
255	其它错误

4.2. 附录 B GET_SYS_REDUDIAG 指令输出参数代码说明

诊断代码	代码说明	故障影响
输出引脚 HARDWARE_INIT-硬件初始化诊断信息诊断		
0	模块插稳判断超时错误	查询时返回错误信息，导致读机槽地址不可靠
1	铁电存储器(SPI 接口)初始化错误	导致不能读写功能块定义的 IP 地址、以太网冗余第三 IP 地址、读固件日期、读写应用程序校验字（将不能从 FLASH 里加载应用程序）
2	读固件日期时间错误	查询时返回错误信息
3	硬件 RTC(IIC 接口)初始化错误	查询时返回错误信息，导致不能正确读取日期时间，不能初始化系统时间，但不影响功能块的 RTC 操作
4	读硬件 RTC 日期时间错误	查询时返回错误信息，将不能正确设定系统时间
5	FPGA 配置错误	如果系统配置高速总线模块，将禁止高速总线 IO 操作；在非以太网冗余情况下禁止启动背板/机架冗余线程，禁止高速总线 IO 操作和冗余操作
6	冗余等待信号量创建错误	不能进行冗余同步
7	冗余继续运行信号量创建错误	不能进行冗余同步(禁止调用信号量的挂起和发送函数)
8	Nand FLASH 初始化错误	禁止文件相关操作
9	Nand FLASH 文件系统注册失败	禁止文件相关操作
10	PRUICSS 初始化配置错误	停止本地总线 IO 操作
11	PRUICSS 初始化指令存储器加载错误	停止本地总线 IO 操作
12	PRUICSS 初始化数据存储器加载错误	停止本地总线 IO 操作
13	ftp 文件服务器创建失败	文件系统相关功能块执行返回错误
14	ftp 文件服务器 Socket 操作失败	文件系统相关功能块执行返回错误
15	ftp 文件服务器异常退出	文件系统相关功能块执行返回错误
输出引脚 NETWORK_INIT -网络初始化诊断信息		
0	UDP 接口无效 IP 地址	查询时返回错误信息
1	UDP 接口无效子网掩码	查询时返回错误信息
2	UDP 接口无效网络接口索引	查询时返回错误信息
3	UDP 套接字创建失败	查询时返回错误信息
4	无有效的网络接收接口	查询时返回错误信息

5	TCP 接口无效 IP 地址	查询时返回错误信息
6	TCP 端口绑定失败	查询时返回错误信息
7	TCP 端口 Socket 创建失败	查询时返回错误信息
8	TCP 块驱动注册失败	查询时返回错误信息
9	TCP 套接字设置失败	查询时返回错误信息
10	TCP 接口内存分配异常	查询时返回错误信息
11	TCP 套接字 Connect 失败	查询时返回错误信息
12	路由器初始化失败	查询时返回错误信息
13	路由器注册无效网络地址	查询时返回错误信息
14	路由器无效子网络	查询时返回错误信息
16	CBUS 总线初始化 Socket 打开错误	停止 CBUS 总线 IO 操作
17	CBUS 总线初始化发 Socket 设置错误	停止 CBUS 总线 IO 操作
18	CBUS 总线初始化 IP 地址端口绑定错误	停止 CBUS 总线 IO 操作
19	ModbusTCP 主 Socket 打开错误	屏蔽相关操作
20	ModbusTCP 主 Socket 设置错误	屏蔽相关操作
21	ModbusTCP 主 Socket 端口绑定错误	屏蔽相关操作
22	ModbusTCP 主 Socket 端口监听错误	屏蔽相关操作
输出引脚 RTS_INIT -运行时初始化诊断信息		
0	CM 模块功能加载错误	
1	CM 模块组件初始化失败	
2	CM 模块有未初始化的组件	
3	Retain 数据区分配失败	
4	Retain 数据恢复失败	
5	代码或数据区分配失败	
6	加载启动工程失败	
7	无有效的应用工程	
8	Retain 数据与应用工程	

	不匹配	
9	启动应用工程失败	
10	加载 Retain 数据失败	
11	应用工程目标不匹配	
12	应用工程格式不支持	
13	加载应用工程失败	
16	参数配置本地总线停止错误	停止本地总线配置和扫描
17	参数配置高速总线停止错误	停止高速总线配置和扫描
18	参数配置 CPU 模块配置错误	停止总线配置和扫描
19	参数配置 LBUS 总线配置错误	停止总线配置和扫描
20	参数配置 HBUS 总线配置错误	停止总线配置和扫描
21	参数配置 CBUS 模块配置错误	停止总线配置和扫描
22	参数配置高速总线冗余状态改变错误	停止高速总线配置和扫描
23	参数配置高速总线写配置命令错误	停止高速总线配置和扫描
24	参数配置本地总线冗余状态改变错误	停止本地总线配置和扫描
25	参数配置本地总线写配置命令错误	停止本地总线配置和扫描
26	参数配置高速总线读状态命令错误	停止高速总线配置和扫描
27	参数配置高速总线写输出命令错误	停止高速总线配置和扫描
输出引脚 SYS_RUNNING -系统运行诊断信息		
0	网络接收无效地址类型	
1	UDP 数据接收异常	
2	UDP 数据发送异常	
3	TCP 数据接收无效报文长度	
4	TCP 数据接收无效 Magic 值	
5	TCP 数据接收底层返回错误	

6	TCP 数据接收读数据长度错误	
7	TCP 数据发送失败	
8	路由模块 CheckQueue 错误	
9	路由模块 SendToNetwork 错误	
10	UDP 模块网络地址交换错误	
11	ModbusTcp 服务器 socket 选择错误	
12	ModbusTcp 服务器 socket accept	
13	ModbusTcp 服务器发送失败	
14	ModbusTcp 服务器接收失败	
15	应用执行异常	
16	参数配置 CPU 模块配置错误	
17	在线改变执行异常	
18	在线改变执行间隙异常	
19	应用复位异常	
20	创建启动工程失败由于通讯超时	
21	IEC 任务异常	
22	IO 扫描本地总线停止错误	
23	IO 扫描本地总线冗余状态改变错误	
24	IO 扫描高速总线冗余状态改变错误	
25	IO 扫描高速总线写命令错	
26	CBUS Socket 发送数据读状态错误	
27	CBUS Socket 有未处理的错误	
28	CBUS 发送数据错误	
29	CBUS 发送数据被丢弃	
30	CBUS Socket 接收数据	

	读状态错误	
输出引脚 REDU_FUN -冗余功能诊断信息		
0	以太网冗余同步套接字创建失败	禁止启动冗余线程，标记冗余同步任务异常退出；禁止冗余数据发送
1	以太网冗余同步套接字选项设置失败	禁止启动冗余线程，标记冗余同步任务异常退出；禁止冗余数据发送
2	以太网冗余同步套接字绑定失败	禁止启动冗余线程，标记冗余同步任务异常退出；禁止冗余数据发送
3	以太网冗余同步读接收状态错误	
4	以太网冗余同步接收数据错误	
5	以太网冗余同步发送数据错误	
6	冗余同步任务异常退出	
8	冗余同步数据应答数据错误	
9	冗余同步数据应答功能码错误	
10	冗余同步数据应答长度错误	
11	冗余同步数据应答超时错误	
12	冗余同步 M 数据同步应答数据错误	
13	冗余同步 M 数据同步应答功能码错误	
14	冗余同步 M 数据同步应答长度错误	
15	冗余同步 M 数据同步应答超时错误	
16	冗余同步主机静态数据同步超时	
17	冗余同步动态数据接收超时错误	
18	冗余同步动态 M 数据过限错误	
19	冗余同步本地总线停止失败	
20	冗余同步本地总线冗余状态改变失败	
21	冗余同步高速总线停止	

	失败	
22	冗余同步高速总线冗余状态改变失败	
23	冗余同步主卡或通讯卡冗余切换错误	

4.3. 附录 C GET_SYS_DIAG 指令输出参数代码说明

诊断代码	代码说明	故障影响
输出引脚 HARDWARE_INIT-硬件初始化诊断信息诊断		
0	模块插稳判断超时错误	查询时返回错误信息，导致读机槽地址不可靠
1	铁电存储器(SPI 接口)初始化错误	导致不能读写功能块定义的 IP 地址、以太网冗余第三 IP 地址、读固件日期、读写应用程序校验字（将不能从 FLASH 里加载应用程序）
2	读固件日期时间错误	查询时返回错误信息
3	硬件 RTC(IIC 接口)初始化错误	查询时返回错误信息，导致不能正确读取日期时间，不能初始化系统时间，但不影响功能块的 RTC 操作
4	读硬件 RTC 日期时间错误	查询时返回错误信息，将不能正确设定系统时间
5	FPGA 配置错误	如果系统配置高速总线模块，将禁止高速总线 IO 操作；在非以太网冗余情况下禁止启动背板/机架冗余线程，禁止高速总线 IO 操作和冗余操作
6	冗余等待信号量创建错误	不能进行冗余同步
7	冗余继续运行信号量创建错误	不能进行冗余同步(禁止调用信号量的挂起和发送函数)
8	Nand FLASH 初始化错误	禁止文件相关操作
9	Nand FLASH 文件系统注册失败	禁止文件相关操作
10	PRUICSS 初始化配置错误	停止本地总线 IO 操作
11	PRUICSS 初始化指令存储器加载错误	停止本地总线 IO 操作
12	PRUICSS 初始化数据存储器加载错误	停止本地总线 IO 操作
13	ftp 文件服务器创建失败	文件系统相关功能块执行返回错误
14	ftp 文件服务器 Socket 操作失败	文件系统相关功能块执行返回错误
15	ftp 文件服务器异常退出	文件系统相关功能块执行返回错误
输出引脚 NETWORK_INIT -网络初始化诊断信息		
0	UDP 接口无效 IP 地址	查询时返回错误信息

1	UDP 接口无效子网掩码	查询时返回错误信息
2	UDP 接口无效网络接口索引	查询时返回错误信息
3	UDP 套接字创建失败	查询时返回错误信息
4	无有效的网络接收接口	查询时返回错误信息
5	TCP 接口无效 IP 地址	查询时返回错误信息
6	TCP 端口绑定失败	查询时返回错误信息
7	TCP 端口 Socket 创建失败	查询时返回错误信息
8	TCP 块驱动注册失败	查询时返回错误信息
9	TCP 套接字设置失败	查询时返回错误信息
10	TCP 接口内存分配异常	查询时返回错误信息
11	TCP 套接字 Connect 失败	查询时返回错误信息
12	路由器初始化失败	查询时返回错误信息
13	路由器注册无效网络地址	查询时返回错误信息
14	路由器无效子网络	查询时返回错误信息
16	CBUS 总线初始化 Socket 打开错误	停止 CBUS 总线 IO 操作
17	CBUS 总线初始化发 Socket 设置错误	停止 CBUS 总线 IO 操作
18	CBUS 总线初始化 IP 地址端口绑定错误	停止 CBUS 总线 IO 操作
19	ModbusTCP 主 Socket 打开错误	屏蔽相关操作
20	ModbusTCP 主 Socket 设置错误	屏蔽相关操作
21	ModbusTCP 主 Socket 端口绑定错误	屏蔽相关操作
22	ModbusTCP 主 Socket 端口监听错误	屏蔽相关操作
输出引脚 RTS_INIT -运行时初始化诊断信息		
0	CM 模块功能加载错误	
1	CM 模块组件初始化失败	
2	CM 模块有未初始化的组件	
3	Retain 数据区分配失败	

4	Retain 数据恢复失败	
5	代码或数据区分配失败	
6	加载启动工程失败	
7	无有效的应用工程	
8	Retain 数据与应用工程不匹配	
9	启动应用工程失败	
10	加载 Retain 数据失败	
11	应用工程目标不匹配	
12	应用工程格式不支持	
13	加载应用工程失败	
16	参数配置本地总线停止错误	停止本地总线配置和扫描
17	参数配置高速总线停止错误	停止高速总线配置和扫描
18	参数配置 CPU 模块配置错误	停止总线配置和扫描
19	参数配置 LBUS 总线配置错误	停止总线配置和扫描
20	参数配置 HBUS 总线配置错误	停止总线配置和扫描
21	参数配置 CBUS 模块配置错误	停止总线配置和扫描
22	参数配置高速总线冗余状态改变错误	停止高速总线配置和扫描
23	参数配置高速总线写配置命令错误	停止高速总线配置和扫描
24	参数配置本地总线冗余状态改变错误	停止本地总线配置和扫描
25	参数配置本地总线写配置命令错误	停止本地总线配置和扫描
26	参数配置高速总线读状态命令错误	停止高速总线配置和扫描
27	参数配置高速总线写输出命令错误	停止高速总线配置和扫描
输出引脚 SYS_RUNNING -系统运行诊断信息		
0	网络接收无效地址类型	
1	UDP 数据接收异常	
2	UDP 数据发送异常	

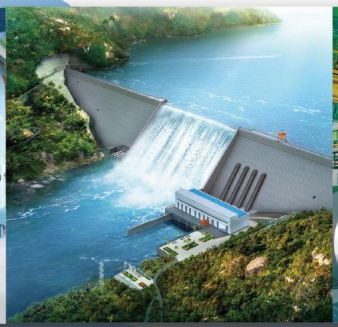
3	TCP 数据接收无效报文长度	
4	TCP 数据接收无效 Magic 值	
5	TCP 数据接收底层返回错误	
6	TCP 数据接收读数据长度错误	
7	TCP 数据发送失败	
8	路由模块 CheckQueue 错误	
9	路由模块 SendToNetwork 错误	
10	UDP 模块网络地址交换错误	
11	ModbusTcp 服务器 socket 选择错误	
12	ModbusTcp 服务器 socket accept	
13	ModbusTcp 服务器发送失败	
14	ModbusTcp 服务器接收失败	
15	应用执行异常	
16	参数配置 CPU 模块配置错误	
17	在线改变执行异常	
18	在线改变执行间隙异常	
19	应用复位异常	
20	创建启动工程失败由于通讯超时	
21	IEC 任务异常	
22	IO 扫描本地总线停止错误	
23	IO 扫描本地总线冗余状态改变错误	
24	IO 扫描高速总线冗余状态改变错误	
25	IO 扫描高速总线写命令错	
26	CBUS Socket 发送数据	

	读状态错误	
27	CBUS Socket 有未处理的错误	
28	CBUS 发送数据错误	
29	CBUS 发送数据被丢弃	
30	CBUS Socket 接收数据读状态错误	
输出引脚 REDU_FUN -冗余功能诊断信息		
0	以太网冗余同步套接字创建失败	禁止启动冗余线程，标记冗余同步任务异常退出；禁止冗余数据发送
1	以太网冗余同步套接字选项设置失败	禁止启动冗余线程，标记冗余同步任务异常退出；禁止冗余数据发送
2	以太网冗余同步套接字绑定失败	禁止启动冗余线程，标记冗余同步任务异常退出；禁止冗余数据发送
3	以太网冗余同步读接收状态错误	
4	以太网冗余同步接收数据错误	
5	以太网冗余同步发送数据错误	
6	冗余同步任务异常退出	
8	冗余同步数据应答数据错误	
9	冗余同步数据应答功能码错误	
10	冗余同步数据应答长度错误	
11	冗余同步数据应答超时错误	
12	冗余同步 M 数据同步应答数据错误	
13	冗余同步 M 数据同步应答功能码错误	
14	冗余同步 M 数据同步应答长度错误	
15	冗余同步 M 数据同步应答超时错误	
16	冗余同步主机静态数据同步超时	
17	冗余同步动态数据接收超时错误	

18	冗余同步动态 M 数据过限错误	
19	冗余同步本地总线停止失败	
20	冗余同步本地总线冗余状态改变失败	
21	冗余同步高速总线停止失败	
22	冗余同步高速总线冗余状态改变失败	
23	冗余同步主卡或通讯卡冗余切换错误	

第 1 版




北京蓝普锋科技有限公司
Beijing Runpower Technology Co.,Ltd
地址：北京市海淀区西三旗建材城中路 12 号院 17 号楼 2 层
E-mails:Service@runpower.cn
电话：010-62740825
技术热线：18519861720
销售热线：18510991991