Truhigh P500 PLC

编程指南及功能手册

V1.4

创恒
Truhigh

目录

第一章 功能与功能块概述

一、Multiprog 功能（FC）与功能块（FB）的定义与分类

功能和功能块都是用户编写的子程序，只是每一个功能块在运行中有独立的数据存储区，而功能则没有。也就是说功能块带记忆，有背景数据区，功能没有记忆，没有背景数据区。

功能有 n 个输入一个输出，功能块有 n 个输入 n 个输出。

使用中的具体差别和注意事项：

功能没有实例化名字，类似高级语言中的函数。功能的任何一个输入管脚都不能悬空。

功能块在应用后必须定义一个不能重复的实例化名字，类似高级语言中的类。功能块的输入和输出管脚可以根据实际情况悬空。请注意，在 Multiprog 中删除某个功能块后，该功能块的定义不能自动删除，必须到变量声明表格中手动删除。

在 Multiprog 库中的功能符号为  ，而功能块为  。在 TruHigh 自定义功能库中，没有将功能和功能块进行分类，用户在使用的时候请自行注意。

Multiprog 功能和功能块内容丰富，包含大量软件自带功能库，包括 BIT_UTIL（位运算功能库）、类型转换 FU（各种数据类型之间的转换）、字符串 FU（字符串）相关的一些运算、以及大部分数学运算功能块等。

创恒自定义功能库进一步完善了 Multiprog 功能库，并进行了详细的分类。包括 Math.mwt（数学运算）、Logic（逻辑运算），为区分 Multiprog 自带功能和功能块，创恒自定义功能和功能块的名字全部“_TH”结尾

二、创恒自定义功能库的安装

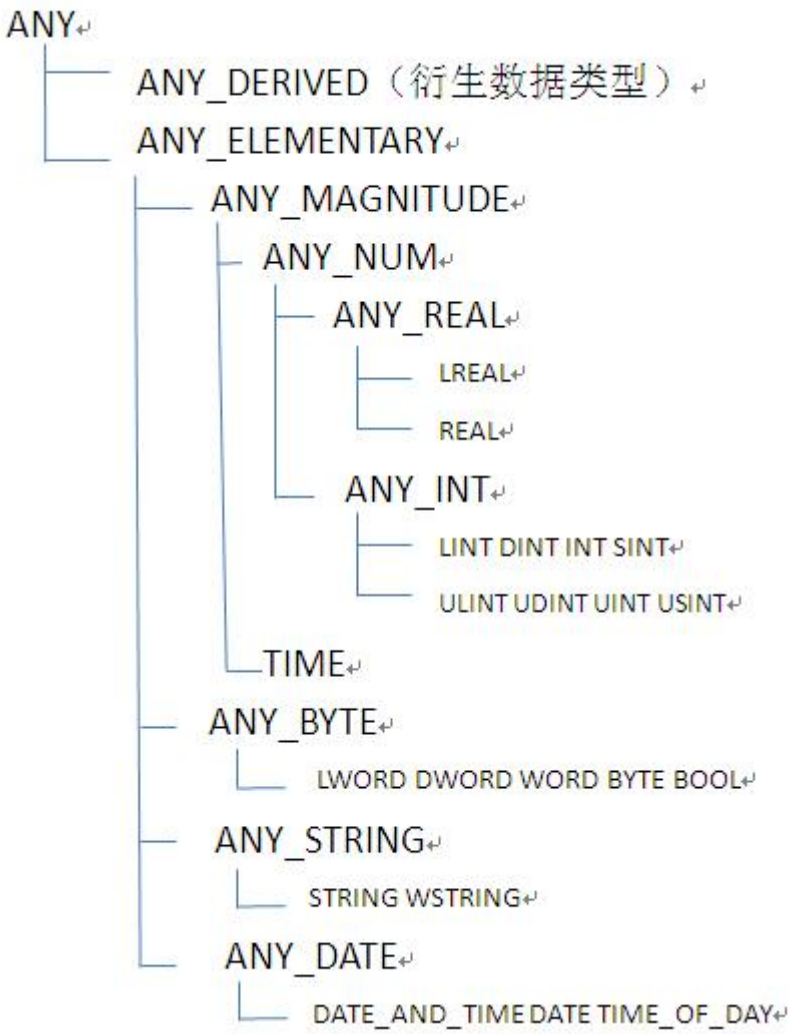
将创恒科技发布的软件包安装完毕之后，创恒自定义功能库就会自动安装进去了，安装位置在：安装目录\Truhigh\P500\MULTIPROG\plc\FW_LIB 或者 C:\ProgramData\PHOENIX CONTACT Software\MULTIPROG Express\5_50_10268\plc\FW_LIB。用户在使用 MODBUS 功能块或自由口功能块时，需要事先在项目目录树的“库”文件夹下，点击右键>插入固件库，将 TifsFwlib 创恒自定义功能库添加到项目树中。

若使用创恒自定义的 PID 功能库，需要事先在项目目录树的“库”文件夹下，点击右键>插入固件库，将 PROCNOS.FWL 添加进去。

三、Multiprog 数据类型

数据类型是包括了基本数据类型分级组的数据类型。 ANY_INT 包括 DINT、INT、SINT、UDINT、UINT 和 USINT 等基本数据类型。 如果一个功能可以与 ANY_INT 相连，则意味着可以处理 DINT、INT、SINT、UDINT、UINT 和 USINT 等数据类型的变量。

Multiprog 数据类型按以下形式组织：



数据类型	描述	大小	范围	缺省值
BOOL	布尔	1	0...1	0
SINT	短整数	8	-128...127	SINT#0
INT	整数	16	-32,768...32,767	0 或 INT#
DINT	双整数	32	-2,147,483,648 至 2,147,483,647	DINT#0

USINT	无符号短整数	8	0 至 255	USINT#0
UINT	无符号整数	16	0 至 65,535	UINT#0
UDINT	无符号双整数	32	0 至 4,294,967,295	UDINT#0
REAL	实数	32	-3.402823466 E+38（约 7 位数字） 至 -1.175494351 E-38（约 7 位数字） 以及 +1.175494351 E-38（约 7 位数字） 至 +3.402823466 E+38（约 7 位数字） 参见 REAL 和 LREAL 处理表单下方的提示。	0.0 或 REAL#0.0
LREAL	长实数	64	~ -1.798 E+308（约 15 位数字） 至 ~ -2.225 E-308（约 15 位数字） 以及 ~ +2.225 E-308（约 15 位数字） 至 ~ +1.798 E+308（约 15 位数字） 参见 REAL 和 LREAL 处理表单下方的提示。	LREAL#0.0
TIME	持续时间	32	0... 4,294,967,295 毫秒	t#0s
BYTE	长度为 8 的位串	8	0...255 （16#00...16#FF）	0
WORD	长度为 16 的位串	16	0...65,535 （16#00...16#FFFF）	0
DWORD	长度为 32 的位串	32	0...4,294,967,295 （16#00....16#FFFFFFFF）	0

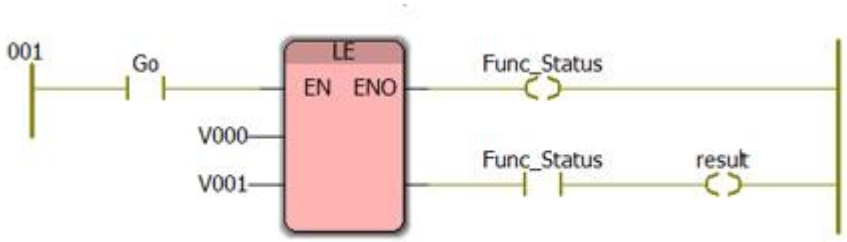
Multiprog 允许用户自定义复杂结构数据类型。在项目树的数据类型文件夹下，点击右键>插入>数据类型，用户可以自定义数组、结构体等复杂结构数据类型。

四、EN/ENO 的使用

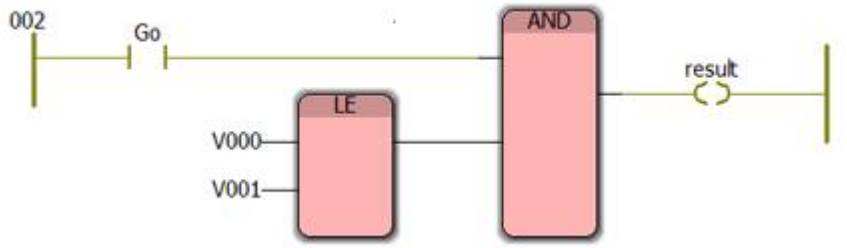
EN/ENO 可被用于有条件地执行各种标准功能。 EN（= enable）指派一个附加使能输入而 ENO（= enable output）指派一个附加使能输出，均可在梯形图（LD）和功能块图（FBD）中使用。 启动 EN/ENO 后，可有条件地执行某个特殊功能。

EN 是一个布尔型输入。 如果为 TRUE，执行该功能，在执行完成并未出现错误后，ENO 设为 TRUE，表明返回一个正确的值。 将其它功能或编码元素的 EN 输入连接到 ENO 输出，这样可依据是否正确执行了功能且未出现错误的情况来判断是否进行程序的后期执行。

如果 EN 为 FALSE，或在执行过程中出现错误（如被零除或超出输出数据类型范围），ENO 设置为 FALSE。



ENO 设置为 FALSE 的条件：使用 EN/ENO 时的具体行为取决于具体的功能。 如果某个功能使用 EN/ENO



后，表现出特定的结果，在相关的说明文档中有详尽的描述。没有表现出特定行为的功能块工作如下： 一旦 EN 为 TRUE 时，ENO 自动设置为 TRUE。 如果 EN 变为 FALSE，ENO 设置为 FALSE。

对逻辑功能的 ENO 输出求值：当 EN = FALSE，该逻辑功能的输出值保持上一次执行功能的有效值。 虽然编程系统没有强制要求，但是应用程序逻辑可能会要求对 ENO 输出进行求值。

例如以下示例中，在 EN=TRUE 时，逻辑运算输出为 TRUE，接着 EN=FALSE，功能块停止计算，逻辑运算的结果保持为 TRUE。随意一般用图示编程，如果 ENO = FALSE，则功能输出被“强行”置为 FALSE。

跟上图功能相同的不带 EN/ENO 的网络：

如果你对 EN/ENO 的功能不是非常熟悉，建议不要使用。

如何启用或禁用 IEC 61131 标准功能的 EN/ENO，在图形编辑器中启用或禁用 IEC 功能的 EN/ENO，步骤

如下：

选择“附加和选项...”菜单项。

在“选项”对话框中，打开“图形编辑器”选项卡。

选中“功能带 EN/ENO”的复选框以启用 EN/ENO。

不选，则为使用该功能但不带 EN 输入和 ENO 输出。

如果仅想针对某一特殊的工作单启用/禁用 EN/ENO，打开工作单并激活菜单项“对象和插入带 EN/ENO 的模块”。

第二章 Mulprog 软件自带功能

一、“功能”库

“功能”库里面包含了绝大部分数学运算和逻辑运算。

1、算数运算功能

1.1、ADD 任意类型数值加法

功能：两个变量，或者两个常量，或者一个变量与一个常量相加

输入数据类型：ANY_NUM

参数	数据类型	描述
IN1	ANY_NUM	被加数
IN2	ANY_NUM	被加数
OUT	ANY_NUM	和

注：在使用中，三个变量的值应该使用相同的数据类型，否则在编译时无法通过。运算的过程中若产生溢出，计算结果无法给出提示，过需要检测结果是否溢出，可以使用 **EN/ENO 功能。**

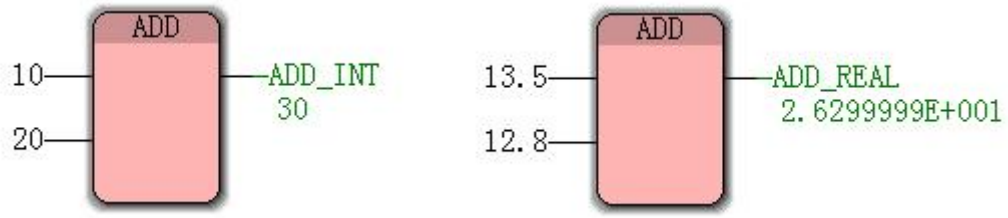
EN/ENO 行为：如果 EN/ENO 被激活（“选项|图形编辑器设置），只有当 EN 输入被置为 TRUE 时，才执行此功能。 如果 EN = FALSE，保留上一次执行功能的输出值。

在以下情况 ENO 输出为 FALSE：

输入变量的数据类型	ENO=FALSE，如果
ANY_INT	结果溢出，即超出数据类型的范围。
ANY_REAL	任何操作数代表一个无效值。结果为一个无效值。

应用举例： 如整数类型的 10 和 20 相加，输出整数类型 ADD_INT 为 30，

如实数类型的 13.5 和 12.8 相加，输出实数类型 ADD_REAL 为 26.3。



1.2、ADD_T_T 时间变量加法

功能：两个时间变量，或者两个时间常量，或者一个时间变量与一个时间常量相加。

这个算术功能实现了连接到输入 IN1 和 IN2 的 TIME 值的加法运算。

参数	数据类型	描述
IN1	TIME	被加数
IN2	TIME	被加数
OUT	TIME	和

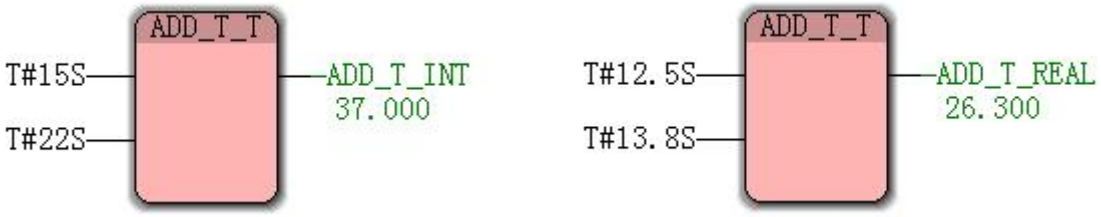
EN/ENO 行为

如果 EN/ENO 被激活（“选项|图形编辑器设置），只有当 EN 输入被置为 TRUE 时，才执行此功能。 如果 EN = FALSE，保留上一次执行功能的输出值。

如果结果超出 0 ms...4,294,967,295 ms 的范围，ENO 输出为 FALSE（TIME 的精度为 1ms）。

应用举例： 时间类型的 t#15s 和 t#22s 相加，输出 ADD_T_INT 为 t#37s，

时间类型的 t#12.5s 和 t#13.8s 相加，输出 ADD_T_REAL 为 t#26.3s。



1.3、SUB 任意类型数值减法

功能：两个变量，或者两个常量，或者一个变量与一个常量相减

输入数据类型：ANY_NUM

参数	数据类型	描述
IN1	ANY_NUM	被减数
IN2	ANY_NUM	减数
OUT	ANY_NUM	差

注：在使用中，三个变量的值应该使用相同的数据类型，否则在编译时无法通过。运算的过程中若产生溢出，计算结果无法给出提示，过需要检测结果是否溢出，可以使用 **EN/ENO 功能。**

EN/ENO 行为：

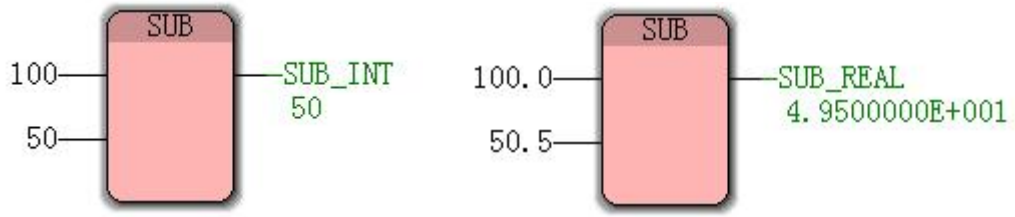
如果 EN/ENO 被激活（“选项|图形编辑器设置），只有当 EN 输入被置为 TRUE 时，才执行此功能。 如果 EN = FALSE，保留上一次执行功能的输出值。

在以下情况 ENO 输出为 FALSE：

输入变量的数据类型	ENO=FALSE，如果
ANY_INT	结果溢出，即超出数据类型的范围。
ANY_REAL	任何操作数代表一个无效值。 结果为一个无效值。

应用举例： 如整数类型的 100 减去 50，输出 SUB_INT 为 50，

如实数类型的 100.0 减去 50.5，输出 SUB_REAL 为 49.5。



1.4、SUB_T_T 时间变量减法

功能：两个时间变量，或者两个时间常量，或者一个时间变量与一个时间常量相减。

这个算术功能实现了连接到输入 IN1 和 IN2 的 TIME 值的加法运算。

参数	数据类型	描述
IN1	TIME	被减数
IN2	TIME	减数
OUT	TIME	差

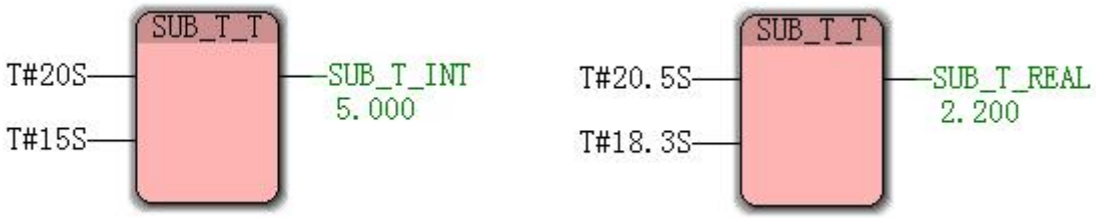
EN/ENO 行为

如果 EN/ENO 被激活（“选项|图形编辑器设置），只有当 EN 输入被置为 TRUE 时，才执行此功能。 如果 EN = FALSE，保留上一次执行功能的输出值。

如果结果超出 0 ms...4,294,967,295 ms 的范围，ENO 输出为 FALSE（TIME 的精度为 1ms）。

应用举例： 时间类型的 t#20s 和 t#15s 相减，输出 SUB_T_INT 为 t#5s，

时间类型的 t#20.5s 和 t#18,.3s 相减，输出 SUB_T_REAL 为 t#2.2s。



1.5、MUL 任意类型数值乘法

功能：两个变量，或者两个常量，或者一个变量与一个常量相乘

输入数据类型：ANY_NUM

参数	数据类型	描述
IN1	ANY_NUM	被乘数
IN2	ANY_NUM	乘数
OUT	ANY_NUM	乘积

注：在使用中，三个变量的值应该使用相同的数据类型，否则在编译时无法通过。运算的过程中若产生溢出，计算结果无法给出提示，过需要检测结果是否溢出，可以使用 **EN/ENO 功能。**

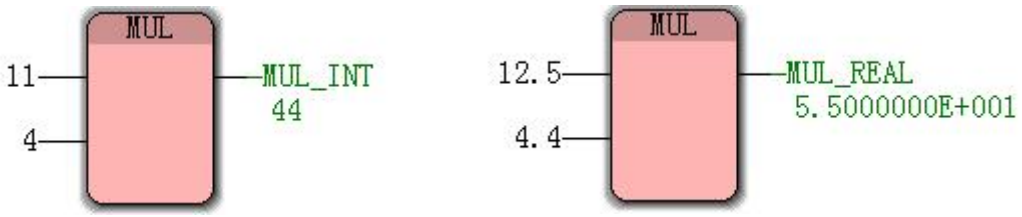
EN/ENO 行为：如果 EN/ENO 被激活（“选项|图形编辑器设置），只有当 EN 输入被置为 TRUE 时，才执行此功能。 如果 EN = FALSE，保留上一次执行功能的输出值。

在以下情况 ENO 输出为 FALSE：

输入变量的数据类型	ENO=FALSE，如果
ANY_INT	结果溢出，即超出数据类型的范围。
ANY_REAL	任何操作数代表一个无效值。 结果为一个无效值。

应用举例： 如整型的 11 与 4 相乘，输出 MUL_INT 为 44，

如实数类型的 11 与 4 相乘，输出 MUL_INT 为 44。



1.6、MUL_T_AI 任意类型数值乘法

功能：这个算术运算功能将 IN1 中的 TIME 值乘以 IN2 的 ANY_INT 值。

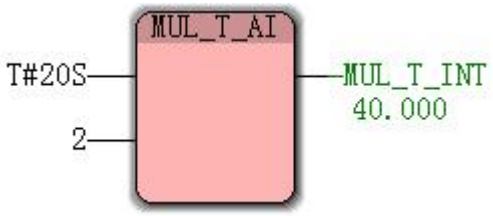
参数	数据类型	描述
IN1	TIME	第一输入值
IN2	ANY_INT	第二输入值
OUT	TIME	运算结果

注：计算结果无法给出提示，过需要检测结果是否溢出，可以使用 EN/ENO 功能。

EN/ENO 行为：如果 EN/ENO 被激活（“选项|图形编辑器设置），只有当 EN 输入被置为 TRUE 时，才执

行此功能。 如果 EN = FALSE，保留上一次执行功能的输出值。如果运算结果超出 0 ms...4,294,967,295 ms，ENO=FALSE。

应用举例：如时间类型的 t#10s 与整型的 2 相乘，输出结果 MuL_T_ INT 为时间类型 t#40s。



1.7、MUL_T_AN 任意整型数值乘法

功能：这个算术运算功能将 IN1 中的 TIME 值乘以 IN2 的 ANY_INT 值。

参数	数据类型	描述
IN1	TIME	第一输入值
IN2	ANY_INT	第二输入值
OUT	TIME	运算结果

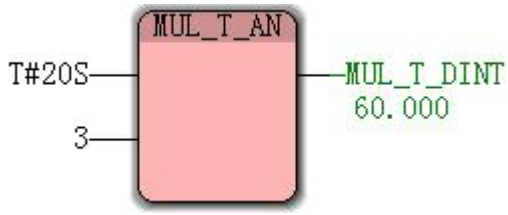
注：计算结果无法给出提示，过需要检测结果是否溢出，可以使用 EN/ENO 功能。

EN/ENO 行为：如果 EN/ENO 被激活（“选项|图形编辑器设置），只有当 EN 输入被置为 TRUE 时，才执行此功能。 如果 EN = FALSE，保留上一次执行功能的输出值。如果运算结果超出 0 ms...4,294,967,295 ms，ENO=FALSE。

在以下情况 ENO 输出为 FALSE：

操作数 IN2 的数据类型	ENO=FALSE，如果
ANY_INT	• 结果超出 0 ms...4,294,967,295 ms 的范围。 • 乘数是负值。
ANY_REAL	• 该操作数代表一个无效值。 • 乘数是负值。

应用举例：如时间类型的 t#20s 与双整型的 3 相乘，输出结果 MuL_T_DINT 为时间类型 t#60s。



1.8、MUL_T_R 任意类型数值乘法

功能：这个算术运算功能将 IN1 中的 TIME 值乘以 IN2 的 ANY_INT 值。

参数	数据类型	描述
IN1	TIME	第一输入值
IN2	ANY_INT	第二输入值
OUT	TIME	运算结果

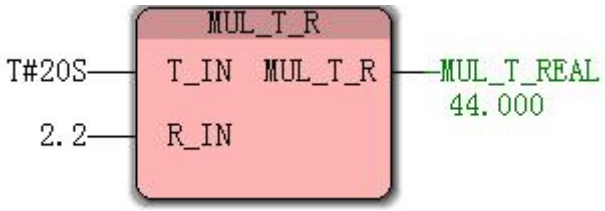
注：计算结果无法给出提示，过需要检测结果是否溢出，可以使用 EN/ENO 功能。

EN/ENO 行为：如果 EN/ENO 被激活（“选项|图形编辑器设置），只有当 EN 输入被置为 TRUE 时，才执行此功能。 如果 EN = FALSE，保留上一次执行功能的输出值。如果运算结果超出 0 ms...4,294,967,295 ms，ENO=FALSE。

在以下情况 ENO 输出为 FALSE：

- 结果超出 0 ms...4,294,967,295 ms 的范围。
- 该操作数 IN2 代表一个无效值。
- 乘数是负值。

应用举例：如时间类型的 t#20s 与实数类型的 2.2 相乘，输出结果 MuL_T_REAL 为时间类型 t#44s。



1.9、DIV 任意类型数值除法

功能：这个算术功能将连接到 IN1 的操作数除以连接到 IN2 的操作数。

参数	数据类型	描述
IN1	ANY_NUM	被除数
IN2	ANY_NUM	除数
OUT	ANY_NUM	输出数值

注：在使用中，三个变量的值应该使用相同的数据类型，否则在编译时无法通过。运算的过程中若产生溢出，计算结果无法给出提示，过需要检测结果是否溢出，可以使用 EN/ENO 功能。

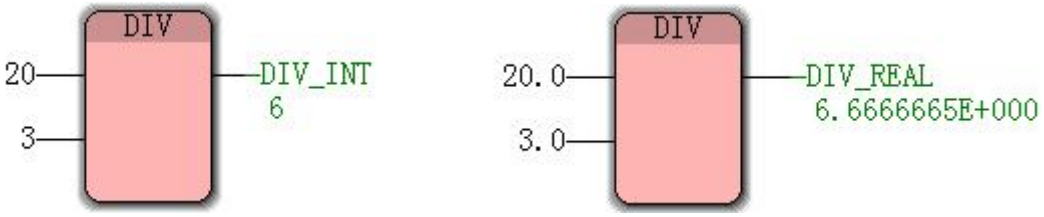
EN/ENO 行为:如果 EN/ENO 被激活（“选项|图形编辑器设置），只有当 EN 输入被置为 TRUE 时，才执行此功能。 如果 EN = FALSE，保留上一次执行功能的输出值。

在以下情况 ENO 输出为 FALSE

输入变量的数据类型	ENO=FALSE，如果
ANY_INT	发生被 0 除。
ANY_REAL	任何操作数代表一个无效值。 结果为一个无效值。 发生被 0 除。

应用举例： 如整数类型的 20 除以整数类型的 3，输出结果 DIV_INT 为实数类型 6，小数部分略去

如实数类型的 20.0 除以实数类型的 3.0，输出结果 DIV_REAL 为实数类型 6.6666666。



1.10、DIV_T_AI 任意类型数值除法

功能：这个算术运算功能将 IN1 中的 TIME 值乘以 IN2 的 ANY_INT 值。

参数	数据类型	描述
IN1	TIME	第一输入值
IN2	ANY_INT	第二输入值
OUT	TIME	运算结果

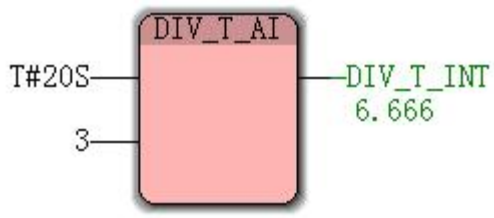
注：计算结果无法给出提示，过需要检测结果是否溢出，可以使用 EN/ENO 功能。

EN/ENO 行为：如果 EN/ENO 被激活（“选项|图形编辑器设置），只有当 EN 输入被置为 TRUE 时，才执行此功能。 如果 EN = FALSE，保留上一次执行功能的输出值。如果运算结果超出 0 ms...4,294,967,295 ms，ENO=FALSE。

在以下情况 ENO 输出为 FALSE：

操作数 IN2 的数据类型	ENO=FALSE，如果
ANY_INT	• 除数是负值。 • 发生被 0 除。

应用举例：如时间类型的 t#20s 除以双整型的 3，输出结果 DIV_T_INT 为 t#6.666s。



1.11、DIV_T_AN 任意类型数值除法

功能：这个算术运算功能将 IN1 中的 TIME 值乘以 IN2 的 ANY_INT 值。

参数	数据类型	描述
IN1	TIME	第一输入值
IN2	ANY_NUM	第二输入值
OUT	TIME	运算结果

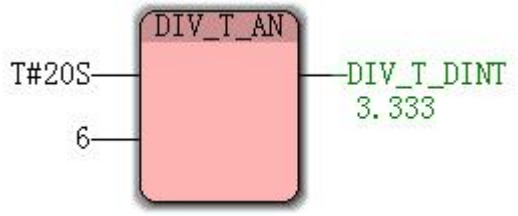
注：计算结果无法给出提示，过需要检测结果是否溢出，可以使用 EN/ENO 功能。

EN/ENO 行为：如果 EN/ENO 被激活（“选项|图形编辑器设置），只有当 EN 输入被置为 TRUE 时，才执行此功能。 如果 EN = FALSE，保留上一次执行功能的输出值。如果运算结果超出 0 ms...4,294,967,295 ms，ENO=FALSE。

在以下情况 ENO 输出为 FALSE：

操作数 IN2 的数据类型	ENO=FALSE，如果
ANY_INT	• 除数是负值。 • 发生被 0 除。
ANY_REAL	• 该操作数代表一个无效值。 • 该结果代表一个无效值。 • 除数是负值。 • 结果超出 0 ms...4,294,967,295 ms 的范围。 • 发生被 0 除。

应用举例：如时间类型的 t#20s 除以 6，输出结果 DIV_T_DINT 为 t#3.333s。



1.12、DIV_T_R 任意类型数值加法

功能：这个算术运算功能将 IN1 中的 TIME 值乘以 IN2 的 ANY_INT 值。

参数	数据类型	描述
IN1	TIME	第一输入值
IN2	ANY_INT	第二输入值
OUT	TIME	运算结果

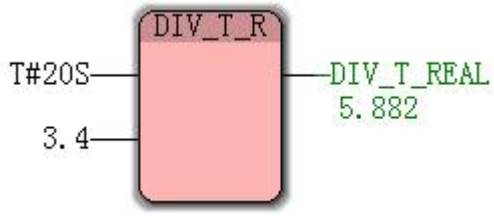
注：计算结果无法给出提示，过需要检测结果是否溢出，可以使用 EN/ENO 功能。

EN/ENO 行为：如果 EN/ENO 被激活（“选项|图形编辑器设置），只有当 EN 输入被置为 TRUE 时，才执行此功能。 如果 EN = FALSE，保留上一次执行功能的输出值。

在以下情况 ENO 输出为 FALSE：

- 结果超出 0 ms...4,294,967,295 ms 的范围。
- 发生被 0 除。
- 该操作数 IN2 代表一个无效值。
- 该结果代表一个无效值。
- 除数是负值。

应用举例：如时间类型的 t#20s 除以实数类型的 3.4，输出结果 DIV_T_REAL 为 t#5.882s。



1.13、EXPT 实数的乘幂运算

功能：这个数学运算计算以 IN1 的数中为底，以 IN2 中的数为指数的求幂运算。

参数	数据类型	描述
IN1	ANY_REAL	底数
IN2	ANY_NUM	指数
OUT	ANY_REAL	输出数值

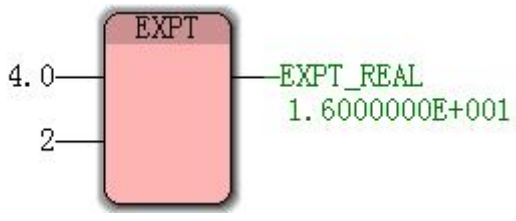
EN/ENO 行为：如果 EN/ENO 被激活（“选项|图形编辑器设置），只有当 EN 输入被置为 TRUE 时，才执行此功能。如果 EN = FALSE，保留上一次执行功能的输出值。

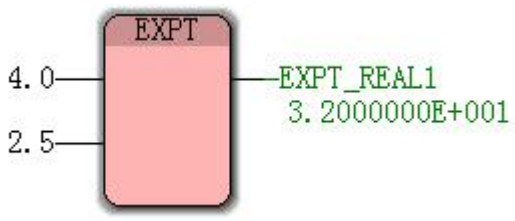
在以下情况 ENO 输出为 FALSE：

- 该操作数代表一个无效值。
- 结果为一个无效的值。

应用举例： 如实数类型的数值 4.0 的 2 次方，结果 EXPT_REAL 为实数类型的数值 16.0，

如实数类型的数值 4.0 的 2.5 次方，结果 EXPT_REAL1 为实数类型的数值 32.0。





1.14、MOD 模数除法操作符

功能：这个算术功能将 IN1 的操作数除以 IN2 的操作数，并在 OUT 中返回余数。

参数	数据类型	描述
IN1	ANY_INT	被除数
IN2	ANY_INT	除数
OUT	ANY_INT	余数

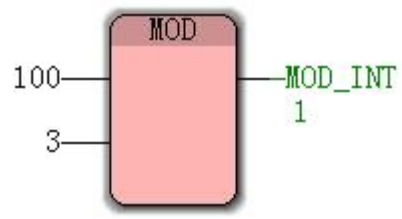
注：在使用中，三个变量的值应该使用相同的数据类型，否则在编译时无法通过。运算的过程中若产生溢出，计算结果无法给出提示，过需要检测结果是否溢出，可以使用 **EN/ENO 功能。**

EN/ENO 行为：如果 EN/ENO 被激活（“选项|图形编辑器设置），只有当 EN 输入被置为 TRUE 时，才执行此功能。如果 EN = FALSE，保留上一次执行功能的输出值。

在以下情况 ENO 输出为 FALSE：

输入变量的数据类型	ENO=FALSE，如果
ANY_INT	发生被 0 除。
ANY_REAL	• 任何操作数代表一个无效值。 • 结果为一个无效值。 • 发生被 0 除。

应用举例：如整型的 100 除以整型的 3，结果是 33，余数为 1，故输出结果 MOD_INT 为 1。



1.15、NEG 双精度补码

功能：这个算术功能对 IN1 中的操作数求反。

参数	数据类型	描述
IN1	ANY_NUM	输入值。
OUT	ANY_NUM	输出值。

注：在使用中，所有变量的值应该使用相同的数据类型，否则在编译时无法通过。运算的过程中若产生溢出，计算结果无法给出提示，过需要检测结果是否溢出，可以使用 **EN/ENO 功能。**

EN/ENO 行为：如果 EN/ENO 被激活（“选项|图形编辑器设置），只有当 EN 输入被置为 TRUE 时，才执

行此功能。如果 EN = FALSE，保留上一次执行功能的输出值。如果结果超出输出数据类型的范围，ENO 输出值设定为 FALSE。

应用举例： 如整型的 123 取反后，结果是-123，

整型的-123 取反后，结果是 123。

1.16、MOV 赋值功能

功能：这个算术功能将输入 IN 的操作数的值，移至输出 out 操作数。

参数	数据类型	描述
IN	ANY	输入值。
OUT	ANY	输出值。

注：在使用中，所有变量的值应该使用相同的数据类型，否则在编译时无法通过。运算的过程中若产生溢出，计算结果无法给出提示，过需要检测结果是否溢出，可以使用 **EN/ENO 功能。**

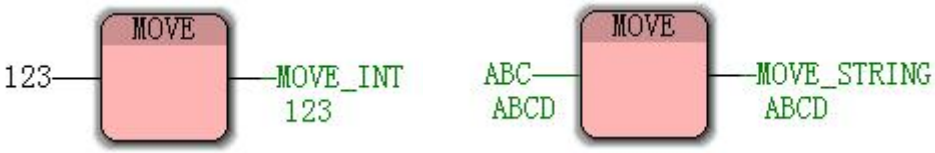
EN/ENO 行为：如果 EN/ENO 被激活（“选项|图形编辑器设置），只有当 EN 输入被置为 TRUE 时，才执行此功能。如果 EN = FALSE，保留上一次执行功能的输出值。

在以下情况 ENO 输出为 FALSE：

输入变量的数据类型	ENO=FALSE，如果
STRING	len(IN) > maxlen(OUT) （输入字符串的长度超出输出字符串所允许的最大长度。） 如果 ENO=FALSE，输出变量写为空字符串“”。

应用举例：如把整型的 123 赋值给变量 MOVE_INT，变量 MOVE_INT 的值就为整型的 10，

字符串 ABC 赋值给变量 MOVE_STRING，则变量 MOVE_STRING 的值就为字符串类型的 ABC。



2、数值功能

2.1、ABS 求绝对值运算功能

功能：这个数值功能可计算连接到 IN1 的操作数的绝对值。

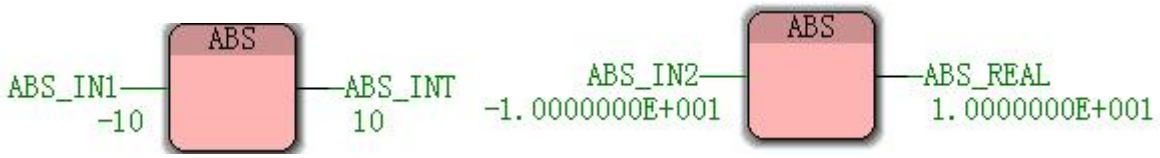
参数	数据类型	描述
IN1	ANY_NUM	输入值。
OUT	ANY_NUM	输出值。

注：在使用中，所有变量的值应该使用相同的数据类型，否则在编译时无法通过。计算结果无法给出提示，过需要检测结果是否溢出，可以使用 EN/ENO 功能。

EN/ENO 行为：如果 EN/ENO 被激活（“选项|图形编辑器设置），只有当 EN 输入被置为 TRUE 时，才执行此功能。 如果 EN = FALSE，保留上一次执行功能的输出值。如果结果超出输出数据类型的范围如 ABS（SINT#-128），ENO 输出值设定为 FALSE。

应用举例： 如整型的数值-10 求绝对值后，ABS_INT 的值为整型的数值 10，

实数类型的数值-10.0 求绝对值后，VAL_REAL 的值为 10.0。



2.2、SQRT 求平方根运算功能

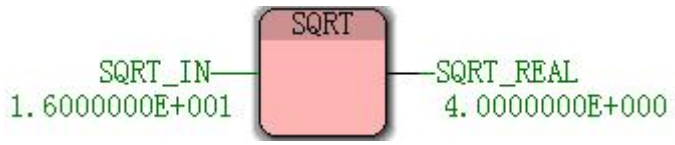
功能：这个数值功能可计算连接到 IN1 的操作数的平方根。

参数	数据类型	描述
IN1	ANY_REAL	输入值。
OUT	ANY_REAL	输出值。

注：计算结果无法给出提示，过需要检测结果是否溢出，可以使用 EN/ENO 功能。

EN/ENO 行为：如果 EN/ENO 被激活（“选项|图形编辑器设置），只有当 EN 输入被置为 TRUE 时，才执行此功能。 如果 EN = FALSE，保留上一次执行功能的输出值。如果连接到输入参数的操作数的值小于零或无效，ENO 输出为 FALSE。

应用举例： 如实数类型的数值 16.0 求开平方后，SQRT_REAL 的值为整型的数值 4.0。



2.3、LN 求自然对数运算功能

功能：这个数值功能可计算连接到 IN1 操作数的自然对数。

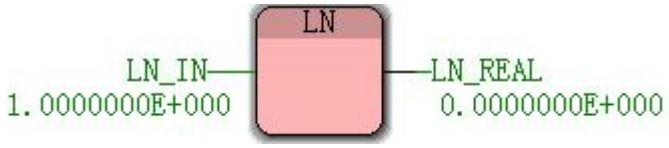
参数	数据类型	描述
IN1	ANY_REAL	输入值。
OUT	ANY_REAL	输出值。

注：计算结果无法给出提示，过需要检测结果是否溢出，可以使用 EN/ENO 功能。

EN/ENO 行为：如果 EN/ENO 被激活（“选项|图形编辑器设置），只有当 EN 输入被置为 TRUE 时，才执行此功能。 如果 EN = FALSE，保留上一次执行功能的输出值。

如果该操作数（输入值）代表一个无效值，操作数的值小于等于零，则 ENO 输出为 FALSE ，结果（输出值）为一个无效的值。

应用举例：如把实数类型的 1.0 求自然对数，变量 LN_REAL 的值就为实数类型的 1.0。



2.4、LOG 求基于 10 的对数运算功能

功能：这个数值功能计算连接到 IN1 操作数的以 10 为底的对数。

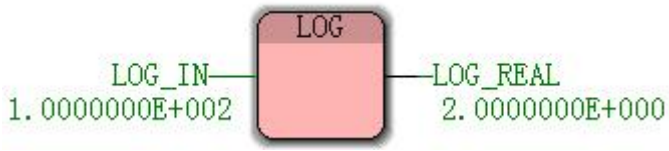
参数	数据类型	描述
IN1	ANY_REAL	输入值。
OUT	ANY_REAL	输出值。

注：计算结果无法给出提示，过需要检测结果是否溢出，可以使用 EN/ENO 功能。

EN/ENO 行为：如果 EN/ENO 被激活（“选项|图形编辑器设置），只有当 EN 输入被置为 TRUE 时，才执行此功能。 如果 EN = FALSE，保留上一次执行功能的输出值。

如果该操作数（输入值）代表一个无效值，操作数的值小于等于零，则 ENO 输出为 FALSE ，结果（输出值）为一个无效的值。

应用举例：如把实数类型的 100.0 求以 10 为底对数，变量 LOG_REAL 的值就为实数类型的 2.0。



2.5、EXP 求自然指数运算功能

功能：这个数值功能可计算连接到 IN1 操作数的自然指数。

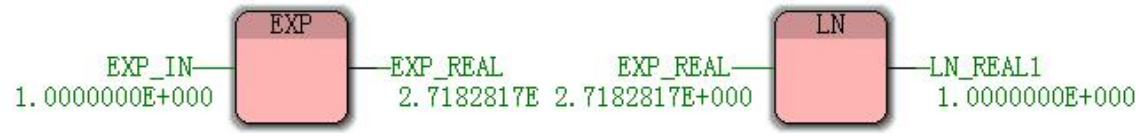
参数	数据类型	描述
IN1	ANY_REAL	e 的指数
OUT	ANY_REAL	输出值。

注：计算结果无法给出提示，过需要检测结果是否溢出，可以使用 EN/ENO 功能。

EN/ENO 行为：如果 EN/ENO 被激活（“选项|图形编辑器设置），只有当 EN 输入被置为 TRUE 时，才执行此功能。如果 EN = FALSE，保留上一次执行功能的输出值。

如果该操作数（e 的指数）代表一个无效值，则 ENO 输出为 FALSE ，结果（输出值）为一个无效的值。

应用举例：如把实数类型的 1.0 求自然指数，变量 EXP_REAL 的值就为实数类型的 2.7182817。



2.6、SIN 求正弦运算功能

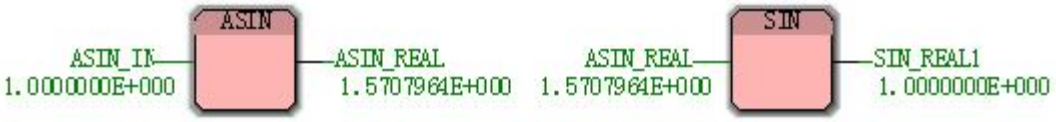
功能：这个数值功能可计算连接到 IN1 的操作数的正弦值。

参数	数据类型	描述
IN1	ANY_REAL	输入值以弧度表示。
OUT	ANY_REAL	输出值。

注：计算结果无法给出提示，过需要检测结果是否溢出，可以使用 EN/ENO 功能。

EN/ENO 行为：如果 EN/ENO 被激活（“选项|图形编辑器设置），只有当 EN 输入被置为 TRUE 时，才执行此功能。 如果 EN = FALSE，保留上一次执行功能的输出值。如果操作数为无效值，ENO 输出为 FALSE。

应用举例：



2.7、COS 求余弦运算功能

功能：这个数值功能可计算连接到 IN1 的操作数的余弦值。

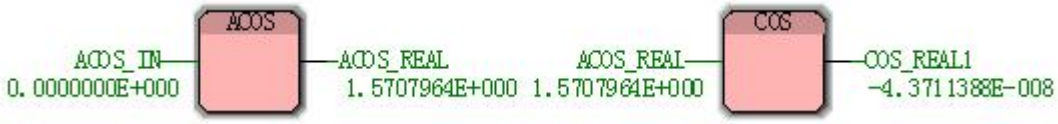
参数	数据类型	描述
IN1	ANY_REAL	输入值以弧度表示。

OUT	ANY_REAL	输出值。

注：计算结果无法给出提示，过需要检测结果是否溢出，可以使用 EN/ENO 功能。

EN/ENO 行为：如果 EN/ENO 被激活（“选项|图形编辑器设置），只有当 EN 输入被置为 TRUE 时，才执行此功能。 如果 EN = FALSE，保留上一次执行功能的输出值。如果操作数为无效值，ENO 输出为 FALSE。

应用举例：



2.8、TAN 求正切运算功能

功能：这个数值功能可计算连接到 IN1 的操作数的正切值。

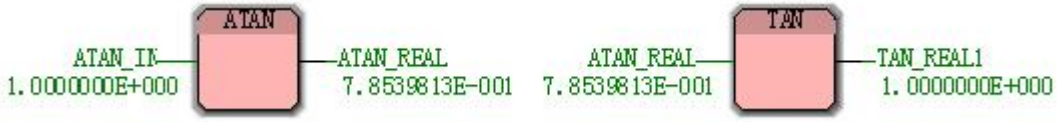
参数	数据类型	描述
IN1	ANY_REAL	输入值以弧度表示。
OUT	ANY_REAL	输出值。

注：计算结果无法给出提示，过需要检测结果是否溢出，可以使用 EN/ENO 功能。

EN/ENO 行为：如果 EN/ENO 被激活（“选项|图形编辑器设置），只有当 EN 输入被置为 TRUE 时，才执行此功能。 如果 EN = FALSE，保留上一次执行功能的输出值。

如果该操作数（输入值）代表一个无效值，则 ENO 输出为 FALSE ，结果（输出值）为一个无效的值。

应用举例：



2.9、ASIN 求反正弦运算功能

功能：这个数值功能可计算连接到 IN1 的操作数的反正弦。

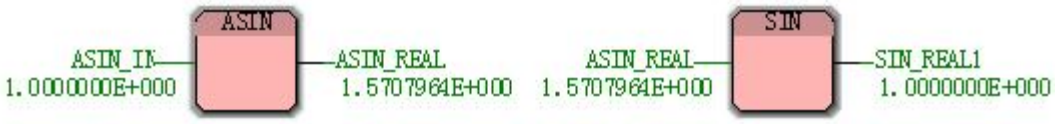
参数	数据类型	描述
IN1	ANY_REAL	输入值。
OUT	ANY_REAL	输出值以弧度表示。

注：计算结果无法给出提示，过需要检测结果是否溢出，可以使用 EN/ENO 功能。

EN/ENO 行为：如果 EN/ENO 被激活（“选项|图形编辑器设置），只有当 EN 输入被置为 TRUE 时，才执行此功能。 如果 EN = FALSE，保留上一次执行功能的输出值。

如果该操作数（输入值）代表一个无效值，即操作数的范围不在-1.0...+1.0，则 ENO 输出为 FALSE ，结果（输出值）为一个无效的值。

应用举例：



2.10、ACOS 求反余弦运算功能

功能：这个数值功能可计算连接到 IN1 的操作数的反余弦。

参数	数据类型	描述
IN1	ANY_REAL	输入值。
OUT	ANY_REAL	输出值以弧度表示。

注：计算结果无法给出提示，过需要检测结果是否溢出，可以使用 EN/ENO 功能。

EN/ENO 行为：如果 EN/ENO 被激活（“选项|图形编辑器设置），只有当 EN 输入被置为 TRUE 时，才执行此功能。 如果 EN = FALSE，保留上一次执行功能的输出值。

应用举例：

2.11、ATAN 求反正切运算功能

功能：这个数值功能可计算连接到 IN1 的操作数的反正切。

参数	数据类型	描述
IN1	ANY_REAL	输入值。
OUT	ANY_REAL	输出值以弧度表示。

注：计算结果无法给出提示，过需要检测结果是否溢出，可以使用 EN/ENO 功能。

EN/ENO 行为：如果 EN/ENO 被激活（“选项|图形编辑器设置），只有当 EN 输入被置为 TRUE 时，才执行此功能。 如果 EN = FALSE，保留上一次执行功能的输出值。

应用举例：

3、按位布尔运算功能

3.1、AND 逻辑与功能

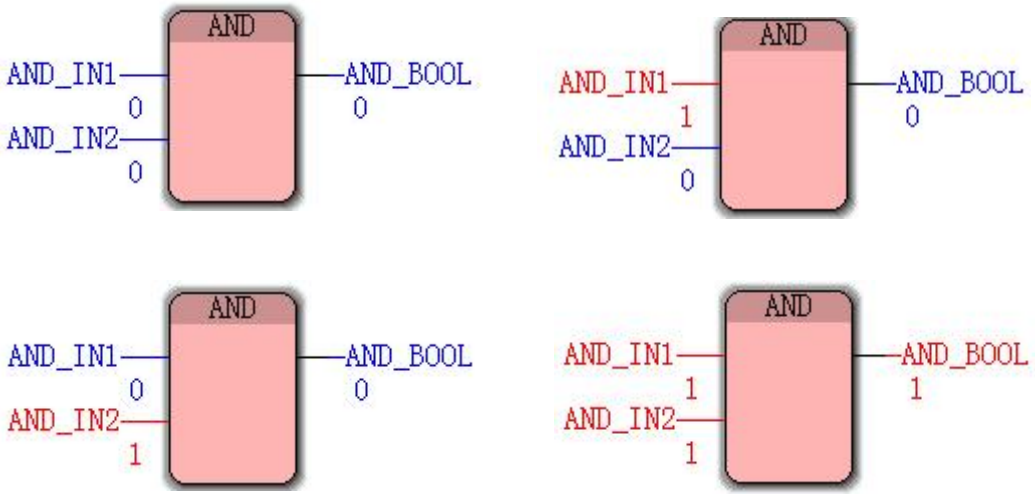
功能：这个逐位布尔运算功能执行对 IN1 和 IN2 中操作数的逻辑与运算。

参数	数据类型	描述
IN1	ANY_BIT	输入值。
IN2	ANY_BIT	输入值。
OUT	ANY_BIT	输出值。

注：在使用中，所有变量的值应该使用相同的数据类型，否则在编译时无法通过。输入 IN2 可以被复制。所有参数都可以被求反。

EN/ENO 行为：如果 EN/ENO 被激活（“选项|图形编辑器设置），只有当 EN 输入被置为 TRUE 时，才执行此功能。如果 EN = FALSE，保留上一次执行功能的输出值。依据程序逻辑，有可能需要对 ENO 输出求值。

应用举例：



3.2、OR 逻辑或功能

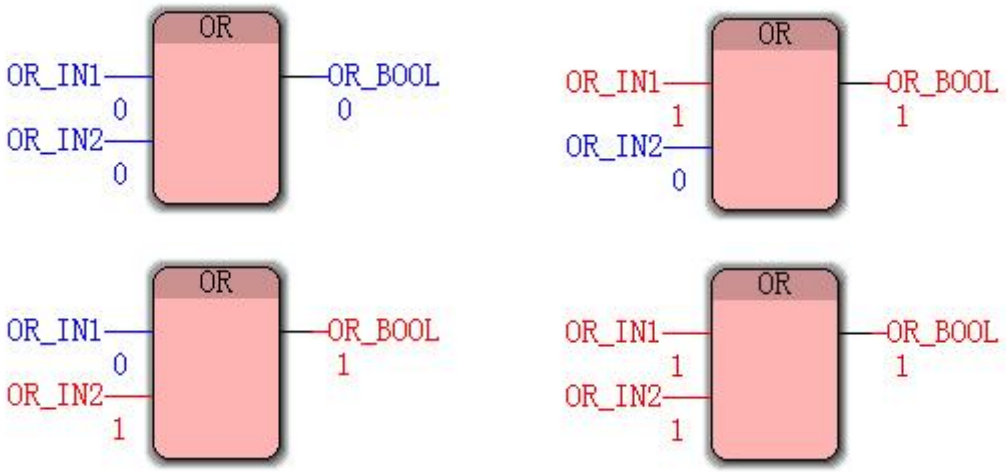
功能：这个逐位布尔运算功能执行对 IN1 和 IN2 中操作数的逻辑或运算。

参数	数据类型	描述
IN1	ANY_BIT	输入值。
IN2	ANY_BIT	输入值。
OUT	ANY_BIT	输出值。

注：在使用中，所有变量的值应该使用相同的数据类型，否则在编译时无法通过。输入 IN2 可以被复制。所有参数都可以被求反。

EN/ENO 行为：如果 EN/ENO 被激活（“选项|图形编辑器设置），只有当 EN 输入被置为 TRUE 时，才执行此功能。如果 EN = FALSE，保留上一次执行功能的输出值。依据程序逻辑，有可能需要对 ENO 输出求值。

应用举例：



3.3、NOT 按位取反功能

功能：这个逐位布尔运算功能执行对 IN1 和 IN2 中操作数的逻辑或运算。

参数	**数据类型**	**描述**
IN1	ANY_BIT	输入值。
OUT	ANY_BIT	输出值。

注：在使用中，所有变量的值应该使用相同的数据类型，否则在编译时无法通过。

EN/ENO 行为：如果 EN/ENO 被激活（“选项|图形编辑器设置），只有当 EN 输入被置为 TRUE 时，才执行此功能。如果 EN = FALSE，保留上一次执行功能的输出值。依据程序逻辑，有可能需要对 ENO 输出求值。

应用举例：

3.4、XOR 逻辑异或功能

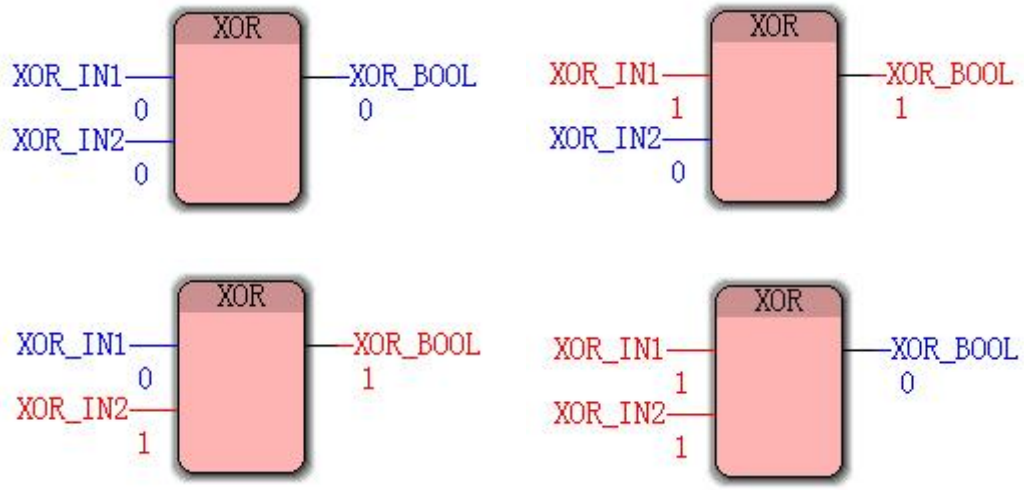
功能：这个逐位布尔运算功能执行对 IN1 和 IN2 中操作数的逻辑异或运算。

参数	**数据类型**	**描述**
IN1	ANY_BIT	输入值。
IN2	ANY_BIT	输入值。
OUT	ANY_BIT	输出值。

注：在使用中，所有变量的值应该使用相同的数据类型，否则在编译时无法通过。输入 IN2 可以被复制。所有参数都可以被求反。

EN/ENO 行为：如果 EN/ENO 被激活（“选项|图形编辑器设置），只有当 EN 输入被置为 TRUE 时，才执行此功能。如果 EN = FALSE，保留上一次执行功能的输出值。依据程序逻辑，有可能需要对 ENO 输出求值。

应用举例：



4、选择运算功能

4.1、LIMIT 范围限定功能

功能：这个选择功能将输入参数 IN 的值限定在输入参数 MN(最小)和 MX(最大)所定义范围。

OUT := MIN(MAX(IN,MN),MX)

功能 MIN 定义了输出值的最小值而功能 MAX 定义了输出值的最大值。

参数	数据类型	描述
MN	ELEMENTARY	最小值。
IN	ELEMENTARY	输入值。
MX	ELEMENTARY	最大值。
OUT	ELEMENTARY	输出值。

注：在使用中，所有变量的值应该使用相同的数据类型，否则在编译时无法通过。运算的过程中若产生溢出，计算结果无法给出提示，过需要检测结果是否溢出，可以使用 **EN/ENO 功能。**

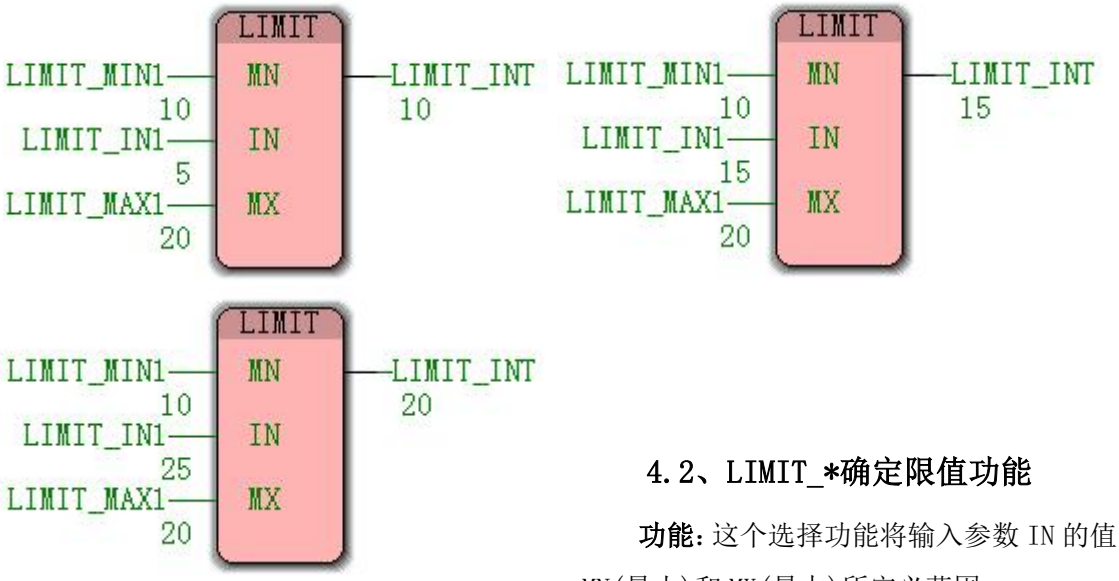
EN/ENO 行为：如果 EN/ENO 被激活（“选项|图形编辑器设置），只有当 EN 输入被置为 TRUE 时，才执行此功能。如果 EN = FALSE，保留上一次执行功能的输出值。依据程序逻辑，有可能需要对 ENO 输出求值。

在以下情况 ENO 输出为 FALSE：

输入变量的数据类型	ENO=FALSE，如果
ANY_INT	MN > MX（最小值大于最大值）。

ANY_REAL	• 任何操作数代表一个无效值。 • MN > MX（最小值大于最大值）。

应用举例：LIMIT_MIN1 和 LIMIT_MAX1 限定了 LIMIT_IN1 的上限和下限，当超过上限时，输出上限设定值，当超过下限时，输出下限设定值，当没有超限时，输出输入值。



4.2、LIMIT_*确定限值功能

功能：这个选择功能将输入参数 IN 的值限定在输入参数 MN(最小)和 MX(最大)所定义范围。

OUT := MIN(MAX(IN,MN),MX)

功能 MIN 定义了输出值的最小值而功能 MAX 定义了输出值的最大值。

“*”是所支持的数据类型的占位符。 该功能适用于 INT、DINT、SINT、REAL 和 STRING 数据类型。

参数	数据类型	描述
MN	INT、DINT、SINT、REAL、STRING	最小值。
IN	INT、DINT、SINT、REAL、STRING	输入值。
MX	INT、DINT、SINT、REAL、STRING	最大值。
OUT	INT、DINT、SINT、REAL、STRING	输出值。

注：在使用中，所有变量的值应该使用相同的数据类型，否则在编译时无法通过。运算的过程中若产生溢出，计算结果无法给出提示，过需要检测结果是否溢出，可以使用 **EN/ENO 功能。**

EN/ENO 行为：如果 EN/ENO 被激活（“选项|图形编辑器设置），只有当 EN 输入被置为 TRUE 时，才执行此功能。如果 EN = FALSE，保留上一次执行功能的输出值。

在以下情况 ENO 输出为 FALSE：

输入变量的数据类型	ENO=FALSE，如果
INT、DINT、SINT	MN > MX（最小值大于最大值）。

REAL	• 任何操作数代表一个无效值。 • MN > MX
STRING	• MN <= IN <= MX and (len(IN) > maxlen(OUT)) • MN <= IN <= MX and (len(IN) > maxlen(OUT)) • MN >= IN >= MX and (len(IN) > maxlen(OUT)) • MN > MX 与： len(...)=字符串实际长度 maxlen(...)=字符串最大长度 如果 ENO=FALSE，输出变量写为空字符串“”。

4.3、MAX 最大值设定功能

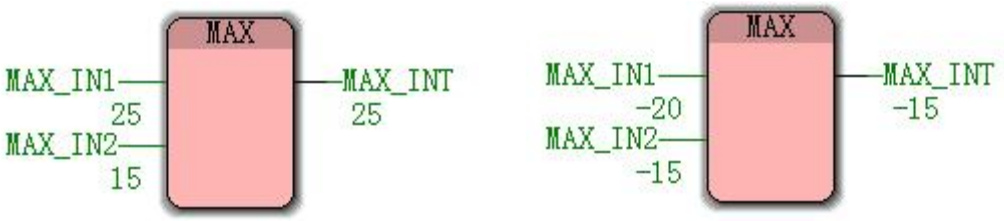
功能：这个选择功能定义了 IN1 和 IN2 中操作数的最大值。

参数	数据类型	描述
IN1	ANY_NUM	第一输入值。
IN2	ANY_NUM	第二输入值。
OUT	ELEMENTARY	输出值。

注：在使用中，所有变量的值应该使用相同的数据类型，否则在编译时无法通过。运算的过程中若产生溢出，计算结果无法给出提示，过需要检测结果是否溢出，可以使用 **EN/ENO 功能。**

EN/ENO 行为：如果 EN/ENO 被激活（“选项|图形编辑器设置），只有当 EN 输入被置为 TRUE 时，才执行此功能。如果 EN = FALSE，保留上一次执行功能的输出值。如果操作数为无效值，ENO 输出为 FALSE。依据程序逻辑，有可能需要对 ENO 输出求值。

应用举例：选择输入端 MAX_IN1 和 MAX_IN2 两个数值中比较大的值作为输出。



4.4、MAX_*确定最大值功能

功能：这个选择功能定义了 IN1 和 IN2 中操作数的最大值。

“*”是所支持的数据类型的占位符。 该功能适用于 INT、DINT、SINT、REAL、LREAL 和 STRING 数据

类型。

参数	数据类型	描述
IN1	INT、DINT、SINT、REAL、STRING	第一输入值。
IN2	INT、DINT、SINT、REAL、STRING	第二输入值。
OUT	INT、DINT、SINT、REAL、STRING	输出值。

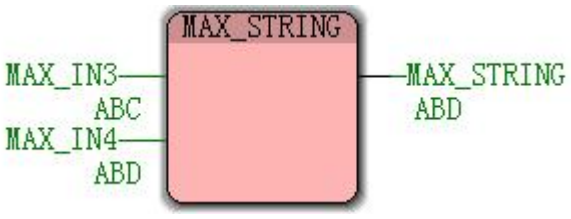
注：在使用中，所有变量的值应该使用相同的数据类型，否则在编译时无法通过。运算的过程中若产生溢出，计算结果无法给出提示，过需要检测结果是否溢出，可以使用 **EN/ENO 功能。**

EN/ENO 行为：如果 EN/ENO 被激活（“选项|图形编辑器设置），只有当 EN 输入被置为 TRUE 时，才执行此功能。如果 EN = FALSE，保留上一次执行功能的输出值。

在以下情况 ENO 输出为 FALSE：

输入变量的数据类型	ENO=FALSE，如果
ANY_REAL	任何操作数代表一个无效值。
STRING	max(len(IN1), len(IN2)) > maxlen(OUT) 与： len(...)=字符串实际长度 maxlen(...)=字符串最大长度 如果 ENO=FALSE，输出变量写为空字符串“”。

应用举例：选择输入端 MAX_IN3 和 MAX_IN4 两个数值中比较大的值作为输出。



4.5、MIN 最小值设定功能

功能：这个选择功能定义了 IN1 和 IN2 中操作数的最小值。

参数	数据类型	描述
IN1	ANY_NUM	第一输入值。
IN2	ANY_NUM	第二输入值。

OUT	ELEMENTARY	输出值。

注：在使用中，所有变量的值应该使用相同的数据类型，否则在编译时无法通过。运算的过程中若产生溢出，计算结果无法给出提示，过需要检测结果是否溢出，可以使用 **EN/ENO 功能。**

EN/ENO 行为：如果 EN/ENO 被激活（“选项|图形编辑器设置），只有当 EN 输入被置为 TRUE 时，才执行此功能。如果 EN = FALSE，保留上一次执行功能的输出值。如果操作数为无效值，ENO 输出为 FALSE。依据程序逻辑，有可能需要对 ENO 输出求值。

应用举例：选择输入端 MIX_IN1 和 MIX_IN2 两个数值中比较小的值作为输出。

4.6、MIN_*确定最小值功能

功能：这个选择功能定义了 IN1 和 IN2 中操作数的最大值。

“*”是所支持的数据类型的占位符。 该功能适用于 INT、DINT、SINT、REAL、LREAL 和 STRING 数据类型。

参数	数据类型	描述
IN1	INT、DINT、SINT、REAL、STRING	第一输入值。
IN2	INT、DINT、SINT、REAL、STRING	第二输入值。
OUT	INT、DINT、SINT、REAL、STRING	输出值。

注：在使用中，所有变量的值应该使用相同的数据类型，否则在编译时无法通过。运算的过程中若产生溢出，计算结果无法给出提示，过需要检测结果是否溢出，可以使用 **EN/ENO 功能。**

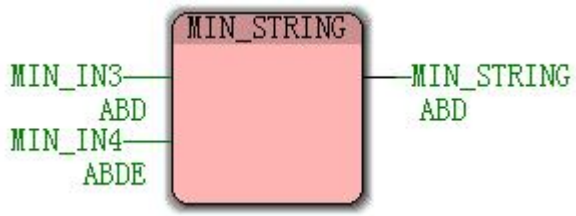
EN/ENO 行为：如果 EN/ENO 被激活（“选项|图形编辑器设置），只有当 EN 输入被置为 TRUE 时，才执行此功能。如果 EN = FALSE，保留上一次执行功能的输出值。

在以下情况 ENO 输出为 FALSE：

输入变量的数据类型	ENO=FALSE，如果
ANY_REAL	任何操作数代表一个无效值。
STRING	min(len(IN1), len(IN2)) > maxlen(OUT) 与： len(...)=字符串实际长度

	maxlen(...)=字符串最大长度 如果 ENO=FALSE，输出变量写为空字符串“”。

应用举例：选择输入端 MIX_IN3 和 MIX_IN4 两个数值中比较小的值作为输出。



4.7、SEL 二进制选择功能

功能：该选择功能可从输入 IN1 或 IN2 中选择一个值并根据赋予输入 G 的值将该值写入输出 OUT 中。

如果 G=FALSE，IN0 的输出值写入 OUT。

如果 G=TRUE，IN1 的输出值写入 OUT。

参数	数据类型	描述
G	BOOL	选择输入。
IN0	ANY	输入值。
IN1	ANY	输入值。
OUT	ANY	输出值。

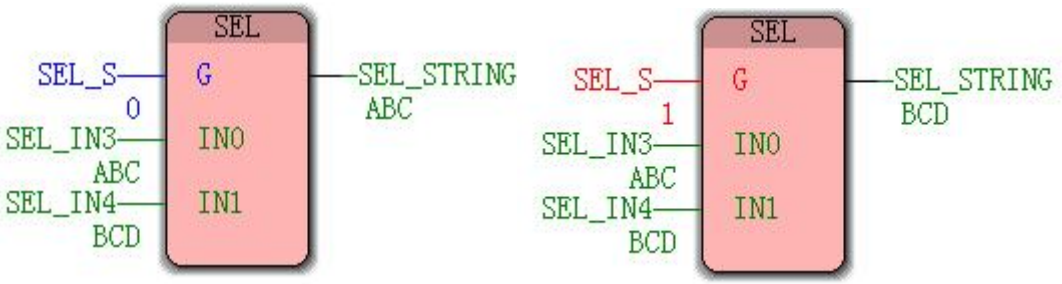
注：在使用中，变量 IN0 和 IN1 的值应该使用相同的数据类型，否则在编译时无法通过。另外 G 可以取反操作。运算的过程中若产生溢出，计算结果无法给出提示，过需要检测结果是否溢出，可以使用 EN/ENO 功能。

EN/ENO 行为：如果 EN/ENO 被激活（“选项|图形编辑器设置），只有当 EN 输入被置为 TRUE 时，才执行此功能。如果 EN = FALSE，保留上一次执行功能的输出值。依据程序逻辑，有可能需要对 ENO 输出求值。

在以下情况 ENO 输出为 FALSE：

输入变量的数据类型	ENO=FALSE，如果
STRING	G==FALSE && (len(IN0) > maxlen(OUT)) 与： len(...)=字符串实际长度 maxlen(...)=字符串最大长度

应用举例：当 SEL_S 为 1 时，选择 SEL_IN1 作为输出，当 SEL_S 为 1 时，选择 SEL_IN2 作为输出。



4.8、SEL_*确定二进制选择功能

功能：该选择功能可从输入 IN1 或 IN2 中选择一个值并根据赋予输入 G 的值将该值写入输出 OUT 中。

如果 G=FALSE，INO 的输出值写入 OUT。

如果 G=TRUE，IN1 的输出值写入 OUT。

“*”是所支持的数据类型的占位符。 该功能适用于 BOOL、BYTE、DINT、INT、SINT、REAL、LREAL、STRING、WORD、DWORD 以及 TIME 数据类型。

参数	数据类型	描述
G	BOOL	选择输入。
INO	BOOL、BYTE、DINT、INT、SINT、REAL、STRING、WORD、DWORD、TIME	输入值。
IN1	BOOL、BYTE、DINT、INT、SINT、REAL、STRING、WORD、DWORD、TIME	输入值。
OUT	BOOL、BYTE、DINT、INT、SINT、REAL、STRING、WORD、DWORD、TIME	输出值。

注：在使用中，变量 INO 和 IN1 的值应该使用相同的数据类型，否则在编译时无法通过。另外 G 可以取反操作。运算的过程中若产生溢出，计算结果无法给出提示，过需要检测结果是否溢出，可以使用 **EN/ENO 功能。**

EN/ENO 行为：如果 EN/ENO 被激活（“选项|图形编辑器设置），只有当 EN 输入被置为 TRUE 时，才执行此功能。如果 EN = FALSE，保留上一次执行功能的输出值。

在以下情况 ENO 输出为 FALSE：

输入变量的数据类型	ENO=FALSE，如果

STRING	G==FALSE && (len(IN0) > maxlen(OUT)) 与： len(...)=字符串实际长度 maxlen(...)=字符串最大长度 如果 ENO=FALSE，输出变量写为空字符串“”。

5、比较运算功能

5.1、EQ 等于功能

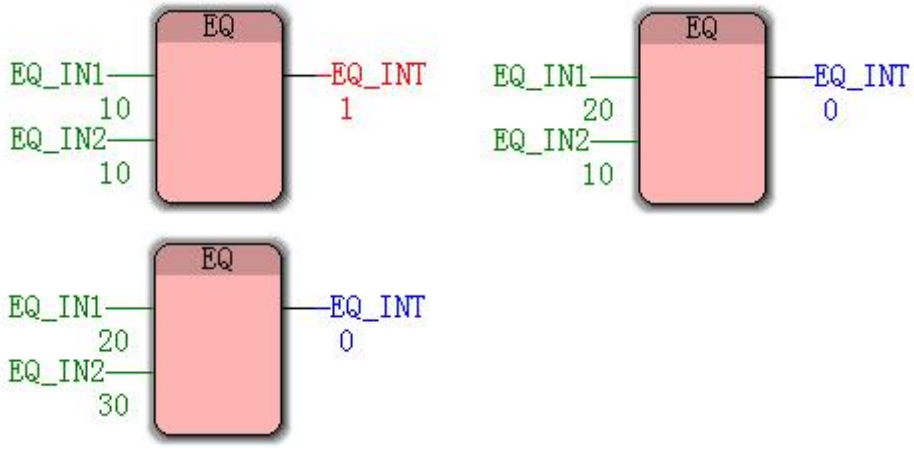
功能：该比较功能将输入 IN1 中的操作数与输入 IN2 的操作数进行比较，如果 IN1 等于 IN2，在 OUT 里设置为 TRUE，否则设置为 FALSE。 从左到右完成比较。

参数	数据类型	描述
IN1	ELEMENTARY	第一输入值。
IN2	ELEMENTARY	第二输入值。
OUT	BOOL	如果输入值相等，则为 TRUE。 如果输入值不相等，则为 FALSE。

注：输入 IN1、IN2 可以被复制。在使用中，变量 IN1 和 IN2 的值应该使用相同的数据类型，否则在编译时无法通过。另外 OUT 可以取反操作。运算的过程中若产生溢出，计算结果无法给出提示，过需要检测结果是否溢出，可以使用 **EN/ENO 功能。**

EN/ENO 行为：如果 EN/ENO 被激活（“选项|图形编辑器设置），只有当 EN 输入被置为 TRUE 时，才执行此功能。如果 EN = FALSE，保留上一次执行功能的输出值。依据程序逻辑，有可能需要对 ENO 输出求值。

应用举例：当 EQ_IN1 等于 EQ_IN2 时，输出为 1，当不相等时，输出为 0。



5.2、GE 大于等于功能

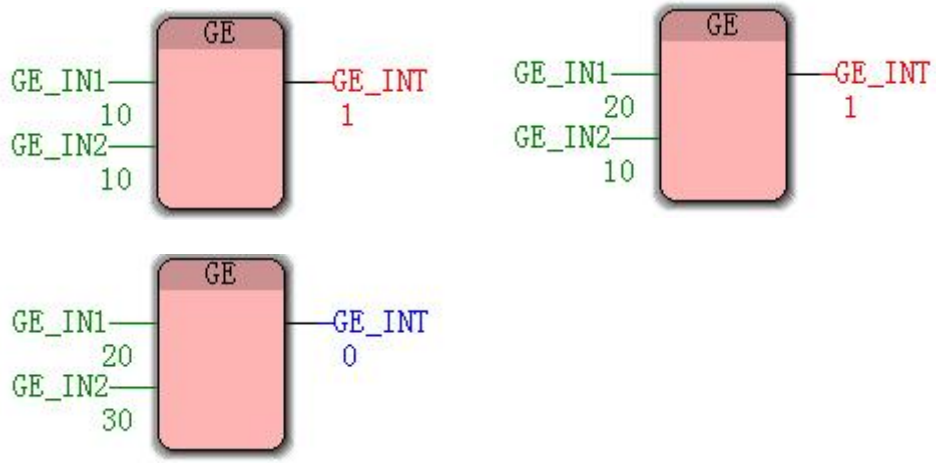
功能：该比较功能将输入 IN1 中的操作数与输入 IN2 的操作数进行比较，如果 IN1 大于或等于 IN2，在 OUT 里设置为 TRUE，否则设置为 FALSE。 从左到右完成比较。

参数	数据类型	描述
IN1	ELEMENTARY	第一输入值。
IN2	ELEMENTARY	第二输入值。
OUT	BOOL	如果 IN1 大于或者等于 IN2，则为真 TRUE。 如果 IN1 小于 IN2，则为 FALSE 假。

注：输入 IN1、IN2 可以被复制。在使用中，变量 IN1 和 IN2 的值应该使用相同的数据类型，否则在编译时无法通过。另外 OUT 可以取反操作。运算的过程中若产生溢出，计算结果无法给出提示，过需要检测结果是否溢出，可以使用 **EN/ENO 功能。**

EN/ENO 行为：如果 EN/ENO 被激活（“选项|图形编辑器设置），只有当 EN 输入被置为 TRUE 时，才执行此功能。如果 EN = FALSE，保留上一次执行功能的输出值。依据程序逻辑，有可能需要对 ENO 输出求值。

应用举例：当 GE_IN1 大于或者等于 GE_IN2 时，输出为 1，当 GE_IN1 小于 GE_IN2 时，输出为 0。



5.3、GT 大于功能

功能：该比较功能将输入 IN1 中的操作数与输入 IN2 的操作数进行比较，如果 IN1 大于 IN2，在 OUT 里设置为 TRUE，否则设置为 FALSE。 从左到右完成比较。

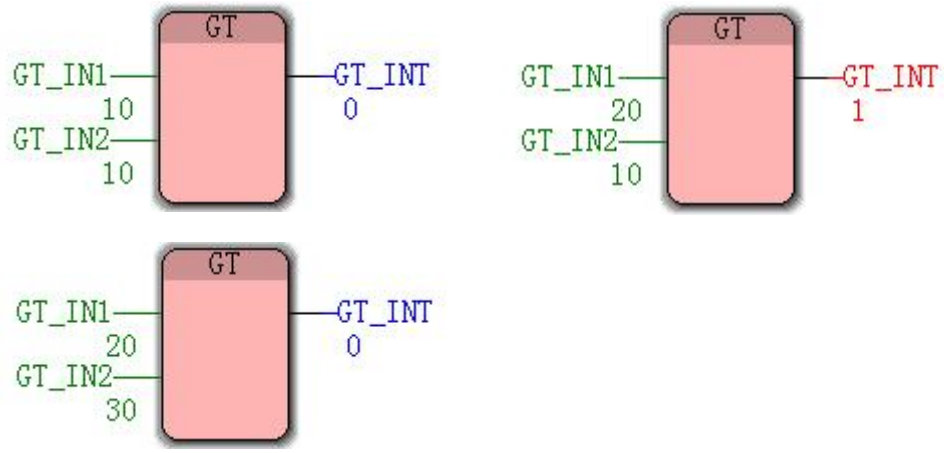
参数	数据类型	描述
IN1	ELEMENTARY	第一输入值。
IN2	ELEMENTARY	第二输入值。
OUT	BOOL	如果 IN1 大于 IN2，则为 TRUE 真。

		如果 IN1 小于或等于 IN2，则为 FALSE 假。

注：输入 IN1、IN2 可以被复制。在使用中，变量 IN1 和 IN2 的值应该使用相同的数据类型，否则在编译时无法通过。另外 OUT 可以取反操作。运算的过程中若产生溢出，计算结果无法给出提示，过需要检测结果是否溢出，可以使用 **EN/ENO 功能。**

EN/ENO 行为：如果 EN/ENO 被激活（“选项|图形编辑器设置），只有当 EN 输入被置为 TRUE 时，才执行此功能。如果 EN = FALSE，保留上一次执行功能的输出值。依据程序逻辑，有可能需要对 ENO 输出求值。

应用举例：当 GT_IN1 大于 GT_IN2 时，输出为 1，当 GT_IN1 小于或等于 GT_IN2 时，输出为 0。



5.4、LE 小于等于功能

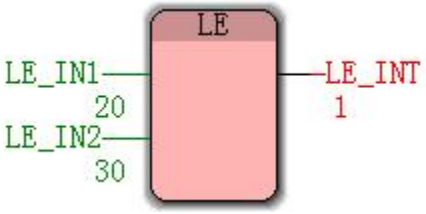
功能：该比较功能将输入 IN1 中的操作数与输入 IN2 的操作数进行比较，如果 IN1 小于或等于 IN2，在 OUT 里设置为 TRUE，否则设置为 FALSE。 从左到右完成比较。

参数	数据类型	描述
IN1	ELEMENTARY	第一输入值。
IN2	ELEMENTARY	第二输入值。
OUT	BOOL	如果 IN1 小于或者等于 IN2，则为 TRUE 真。 如果 IN1 大于 IN2，则为 FALSE 假。

注：输入 IN1、IN2 可以被复制。在使用中，变量 IN1 和 IN2 的值应该使用相同的数据类型，否则在编译时无法通过。另外 OUT 可以取反操作。运算的过程中若产生溢出，计算结果无法给出提示，过需要检测结果是否溢出，可以使用 EN/ENO 功能。

EN/ENO 行为：如果 EN/ENO 被激活（“选项|图形编辑器设置），只有当 EN 输入被置为 TRUE 时，才执行此功能。如果 EN = FALSE，保留上一次执行功能的输出值。依据程序逻辑，有可能需要对 ENO 输出求值。

应用举例：当 LE_IN1 小于或者等于 LE_IN2 时，输出为 1，当 LE_IN1 大于 LE_IN2 时，输出为 0。



5.5、LT 小于功能

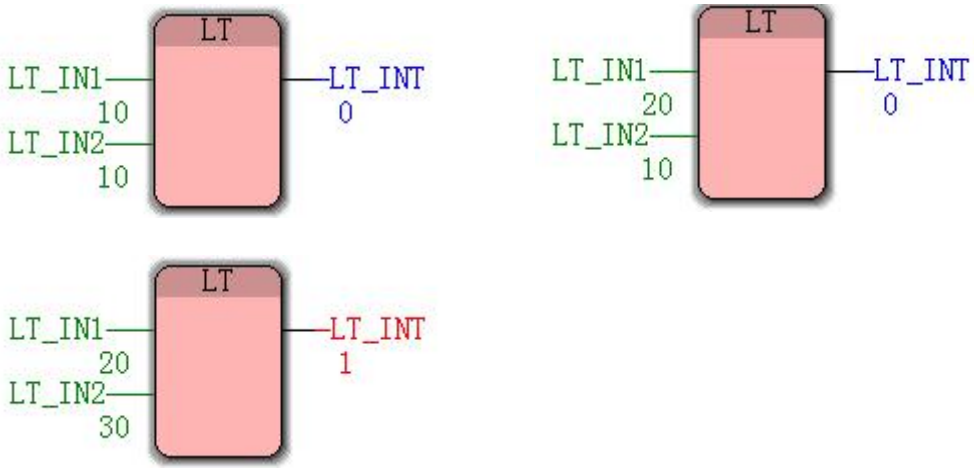
功能：该比较功能将输入 IN1 中的操作数与输入 IN2 的操作数进行比较，如果 IN1 小于 IN2，在 OUT 里设置为 TRUE，否则设置为 FALSE。 从左到右完成比较。

参数	数据类型	描述
IN1	ELEMENTARY	第一输入值。
IN2	ELEMENTARY	第二输入值。
OUT	BOOL	如果 IN1 小于 IN2，则为 TRUE 真。 如果 IN1 大于或等于 IN2，则为 FALSE 假。

注：输入 IN1、IN2 可以被复制。在使用中，变量 IN1 和 IN2 的值应该使用相同的数据类型，否则在编译时无法通过。另外 OUT 可以取反操作。运算的过程中若产生溢出，计算结果无法给出提示，过需要检测结果是否溢出，可以使用 **EN/ENO 功能。**

EN/ENO 行为：如果 EN/ENO 被激活（“选项|图形编辑器设置），只有当 EN 输入被置为 TRUE 时，才执行此功能。如果 EN = FALSE，保留上一次执行功能的输出值。依据程序逻辑，有可能需要对 ENO 输出求值。

应用举例：当 LT_IN1 小于 LT_IN2 时，输出为 1，当 LT_IN1 大于或者等于 LT_IN2 时，输出为 0。



5.6、NE 不等于功能

功能：该比较功能将输入 IN1 中的操作数与输入 IN2 的操作数进行比较，如果 IN1 不等于 IN2，在 OUT

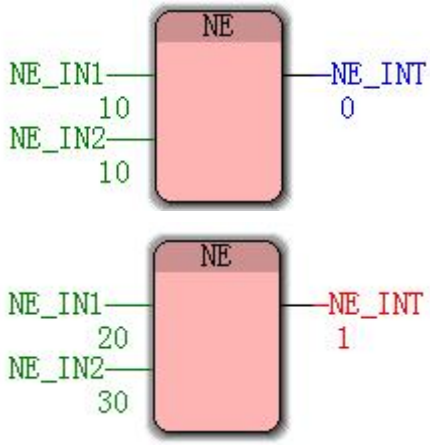
里设置为 TRUE，否则设置为 FALSE。 从左到右完成比较。

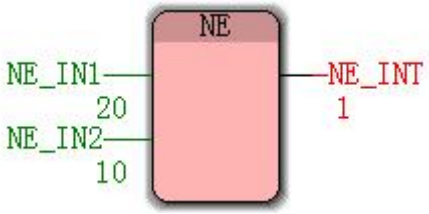
参数	数据类型	描述
IN1	ELEMENTARY	第一输入值。
IN2	ELEMENTARY	第二输入值。
OUT	BOOL	如果 IN1 不等于 IN2，则为 TRUE 真。 如果 IN1 等于 IN2，则为 FALSE 假。

注：输入 IN1、IN2 可以被复制。在使用中，变量 IN1 和 IN2 的值应该使用相同的数据类型，否则在编译时无法通过。另外 OUT 可以取反操作。运算的过程中若产生溢出，计算结果无法给出提示，过需要检测结果是否溢出，可以使用 EN/ENO 功能。

EN/ENO 行为：如果 EN/ENO 被激活（“选项|图形编辑器设置），只有当 EN 输入被置为 TRUE 时，才执行此功能。如果 EN = FALSE，保留上一次执行功能的输出值。依据程序逻辑，有可能需要对 ENO 输出求值。

应用举例：当 NE_IN1 不等于 NE_IN2 时，输出为 1，当 NE_IN1 等于 NE_IN2 时，输出为 0。



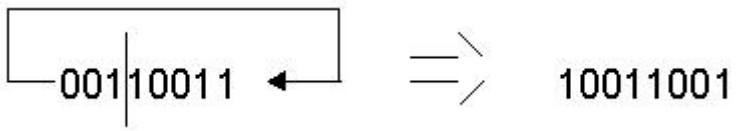


6、位串功能

6.1、ROL 循环左移功能

功能：这个位串功能执行了输入 IN 中操作数的逐位左移。N 规定了移动的位数。

下图为逐位左移 3 位：



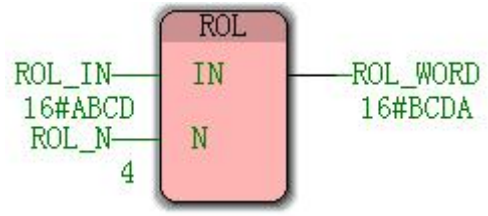
参数	数据类型	描述
IN	ANY_BIT	输入值。
N	ANY_INT	要移动的位数。
OUT	ANY_BIT	输出值。

注：当移动 N ＜ 0 时，功能输出值 0，这是因为参数“N”总是为不带符号的整数值。

将一个布尔直接量（TRUE 或 FALSE）连接到输入 IN 上，产生一个 PLC 错误。

EN/ENO 行为：如果 EN/ENO 被激活（“选项|图形编辑器设置），只有当 EN 输入被置为 TRUE 时，才执行此功能。如果 EN = FALSE，保留上一次执行功能的输出值。依据程序逻辑，有可能需要对 ENO 输出求值。

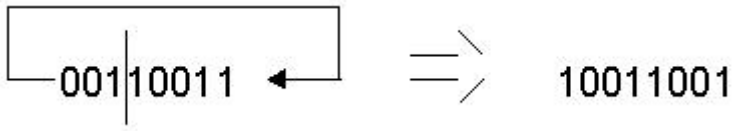
应用举例：ROL_IN 的操作数循环向左移 ROL_N 规定的位数，并输出结果。



6.2、ROL_*标记循环左移功能

功能：这个位串功能执行了输入 IN 中操作数的逐位左移。N 规定了移动的位数。

下图为逐位左移 3 位：



参数	数据类型	描述
IN	BYTE、WORD、DWORD	输入值。
N	INT	要移动的位数。
OUT	BYTE、WORD、DWORD	输出值。

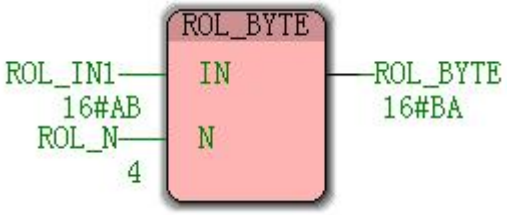
注：当移动 N ＜ 0 时，功能输出值 0，这是因为参数“N”总是为不带符号的整数值。

将一个布尔直接量（TRUE 或 FALSE）连接到输入 IN 上，产生一个 PLC 错误。

“*”是所支持的数据类型的占位符。 该功能适用于 BYTE、WORD 和 DWORD 数据类型。

EN/ENO 行为：如果 EN/ENO 被激活（“选项|图形编辑器设置），只有当 EN 输入被置为 TRUE 时，才执行此功能。如果 EN = FALSE，保留上一次执行功能的输出值。依据程序逻辑，有可能需要对 ENO 输出求值。

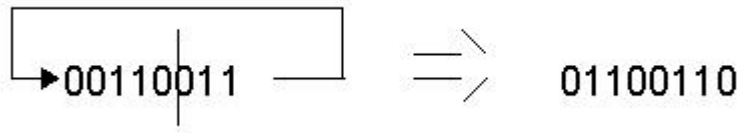
应用举例：ROL_IN1 的操作数循环向左移 ROL_N 规定的位数，并输出结果。



6.3、ROR 循环右移功能

功能：这个位串功能执行了输入 IN 中操作数的逐位右移。N 规定了移动的位数。

下图为逐位右移 3 位：



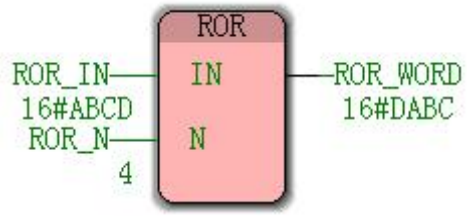
参数	数据类型	描述
IN	ANY_BIT	输入值。
N	ANY_INT	要移动的位数。
OUT	ANY_BIT	输出值。

注：当移动 N ＜ 0 时，功能输出值 0，这是因为参数“N”总是为不带符号的整数值。

将一个布尔直接量（TRUE 或 FALSE）连接到输入 IN 上，产生一个 PLC 错误。

EN/ENO 行为：如果 EN/ENO 被激活（“选项|图形编辑器设置），只有当 EN 输入被置为 TRUE 时，才执行此功能。如果 EN = FALSE，保留上一次执行功能的输出值。依据程序逻辑，有可能需要对 ENO 输出求值。

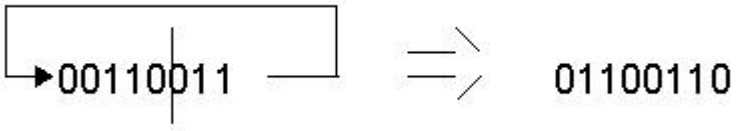
应用举例：ROR_IN 的操作数循环向右移 ROR_N 规定的位数，并输出结果。



6.4、ROR_*标记循环右移功能

功能：这个位串功能执行了输入 IN 中操作数的逐位右移。N 规定了移动的位数。

下图为逐位右移 3 位：



参数	数据类型	描述
IN	BYTE、WORD、DWORD	输入值。
N	INT	要移动的位数。
OUT	BYTE、WORD、DWORD	输出值。

注：当移动 N ＜ 0 时，功能输出值 0，这是因为参数“N”总是为不带符号的整数值。

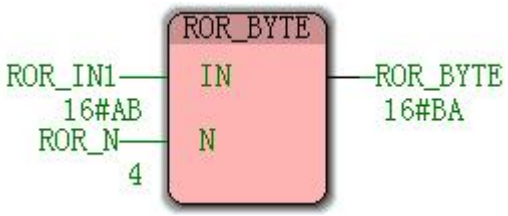
将一个布尔直接量（TRUE 或 FALSE）连接到输入 IN 上，产生一个 PLC 错误。

“*”是所支持的数据类型的占位符。 该功能适用于 BYTE、WORD 和 DWORD 数据类型。

EN/ENO 行为：

如果 EN/ENO 被激活（“选项|图形编辑器设置），只有当 EN 输入被置为 TRUE 时，才执行此功能。如果 EN = FALSE，保留上一次执行功能的输出值。依据程序逻辑，有可能需要对 ENO 输出求值。

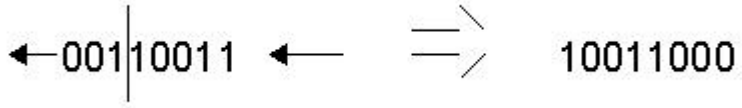
应用举例：ROR_IN1 的操作数循环向右移 ROR_N 规定的位数，并输出结果。



6.5、SHL 左移功能

功能：这个位串功能执行了输入 IN 中操作数的逐位左移。N 规定了移动的位数。 空位用 0 来填充。

下图为逐位左移 3 位：



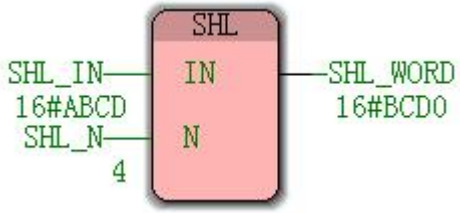
参数	数据类型	描述
IN	ANY_BIT	输入值。
N	ANY_INT	要移动的位数。
OUT	ANY_BIT	输出值。

注：当移动 N ＜ 0 时，功能输出值 0，这是因为参数“N”总是为不带符号的整数值。

将一个布尔直接量（TRUE 或 FALSE）连接到输入 IN 上，产生一个 PLC 错误。

EN/ENO 行为：如果 EN/ENO 被激活（“选项|图形编辑器设置），只有当 EN 输入被置为 TRUE 时，才执行此功能。如果 EN = FALSE，保留上一次执行功能的输出值。依据程序逻辑，有可能需要对 ENO 输出求值。

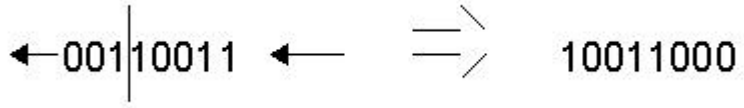
应用举例：SHL_IN 的操作数向左移 SHL_N 规定的位数，空白的位用 0 来填充，并输出结果。



6.6、SHL_*标记左移功能

功能：这个位串功能执行了输入 IN 中操作数的逐位左移。N 规定了移动的位数。 空位用 0 来填充。

下图为逐位左移 3 位：



参数	数据类型	描述
IN	BYTE、WORD、DWORD	输入值。
N	INT	要移动的位数。
OUT	BYTE、WORD、DWORD	输出值。

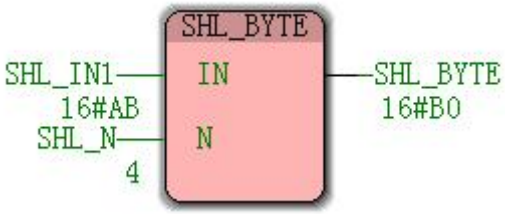
注：当移动 N ＜ 0 时，功能输出值 0，这是因为参数“N”总是为不带符号的整数值。

将一个布尔直接量（TRUE 或 FALSE）连接到输入 IN 上，产生一个 PLC 错误。

“*”是所支持的数据类型的占位符。 该功能适用于 BYTE、WORD 和 DWORD 数据类型。

EN/ENO 行为：如果 EN/ENO 被激活（“选项|图形编辑器设置），只有当 EN 输入被置为 TRUE 时，才执行此功能。如果 EN = FALSE，保留上一次执行功能的输出值。依据程序逻辑，有可能需要对 ENO 输出求值。

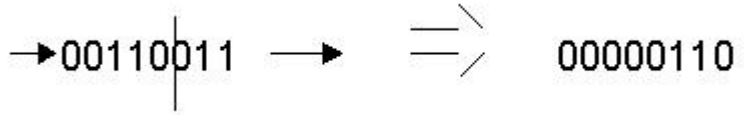
应用举例：SHL_IN1 的操作数向左移 SHL_N 规定的位数，空白的位用 0 来填充，并输出结果。



6.7、SHR 右移功能

功能：这个位串功能执行了输入 IN 中操作数的逐位右移。N 规定了移动的位数。 空位用 0 来填充。

下图为逐位右移 3 位：



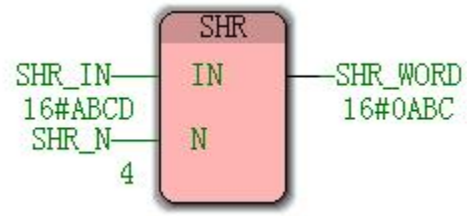
参数	数据类型	描述
IN	ANY_BIT	输入值。
N	ANY_INT	要移动的位数。
OUT	ANY_BIT	输出值。

注：当移动 N ＜ 0 时，功能输出值 0，这是因为参数“N”总是为不带符号的整数值。

将一个布尔直接量（TRUE 或 FALSE）连接到输入 IN 上，产生一个 PLC 错误。

EN/ENO 行为：如果 EN/ENO 被激活（“选项|图形编辑器设置），只有当 EN 输入被置为 TRUE 时，才执行此功能。如果 EN = FALSE，保留上一次执行功能的输出值。依据程序逻辑，有可能需要对 ENO 输出求值。

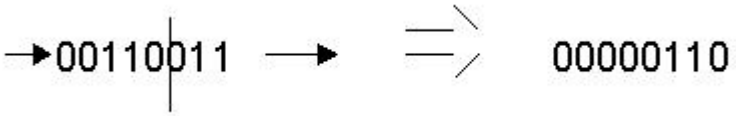
应用举例：SHR_IN 的操作数向右移 SHR_N 规定的位数，空白的位用 0 来填充，并输出结果。



6.8、SHR_*标记右移功能

功能：这个位串功能执行了输入 IN 中操作数的逐位右移。N 规定了移动的位数。 空位用 0 来填充。

下图为逐位右移 3 位：



参数	数据类型	描述
IN	BYTE、WORD、DWORD	输入值。
N	INT	要移动的位数。
OUT	BYTE、WORD、DWORD	输出值。

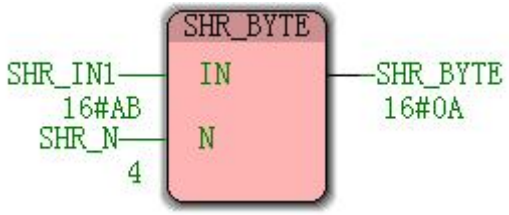
注：当移动 N ＜ 0 时，功能输出值 0，这是因为参数“N”总是为不带符号的整数值。

将一个布尔直接量（TRUE 或 FALSE）连接到输入 IN 上，产生一个 PLC 错误。

“*”是所支持的数据类型的占位符。 该功能适用于 BYTE、WORD 和 DWORD 数据类型。

EN/ENO 行为：如果 EN/ENO 被激活（“选项|图形编辑器设置），只有当 EN 输入被置为 TRUE 时，才执行此功能。如果 EN = FALSE，保留上一次执行功能的输出值。依据程序逻辑，有可能需要对 ENO 输出求值。

应用举例：SHR_IN1 的操作数向右移 SHR_N 规定的位数，空白的位用 0 来填充，并输出结果。



7、字符串功能

7.1、CONCAT 合并字符串功能

功能：该字符串功能可将 IN2 中的输入字符串串接到 IN1 输入字符串的尾端

参数	数据类型	描述
IN1	STRING	第一输入字符串。
IN2	STRING	第二输入字符串。
OUT	STRING	输出字符串。

注：**关于输入和输出字符串正确使用的重要注意事项**

正确使用：可以同时将第一输入的字符串 S1(赋予 IN1)作为输出字符串使用。 这种情况采用 ST 表示法 S1 := CONCAT(S1,S2);

错误使用：不能同时将输入字符串 S2(赋予 IN2)作为输出字符串使用。 这种情况采用 ST 表示法

S2 := CONCAT(S1,S2);

这种情况下，首先将 S1 中字符串的值复制到目标 S2。 不过此时原来 S2 中的值则被覆盖。 将出现运行期错误，并且显示“字符串错误！ 第二字符串等于输出字符串！”

补救：执行 CONCAT 功能之后，分配到输入的缓冲变量也可以连接到输出。

TempString := CONCAT(S1,S2);

S2 := TempString;

出错反应和处理：如果以下错误条件为真，则出现字符串错误：

len(IN1) + len(IN2) > maxlen(OUT)

IN2 == OUT

在这里“len(...)”代表实际字符串长度，而“maxlen(...)”是允许的最大字符串长度（maxlen(Default string) = 80）。

如果发生输入错误，则结果变量为空串“”，并且发生字符串错误。 在这种情况下，SPG 21 被调用，并且在错误目录内产生一个包含相应模块和相关行号的条目。 PLC 保持 RUN 状态。

对于 IPC_28、M68_28 及较早版本的字符串错误的纠错处理：在不同 PLC 版本内，对字符串错误的纠错处理是不同的。 对于 IPC_28、M68_28 及较早版本，会出现下面的情况：

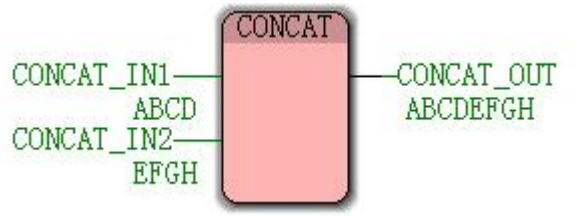
- 如果发生输入错误，则导致字符串出错。 相应的 PLC 将进入 STOP 状态。
- 如果值不在输出数据类型的有效范围内，则发生溢出。

EN/ENO 行为：如果 EN/ENO 被激活（“选项|图形编辑器设置），只有当 EN 输入被置为 TRUE 时，才执行此功能。如果 EN = FALSE，保留上一次执行功能的输出值。

如果 len(IN1) + len(IN2) > maxlen(OUT)，则 ENO 输出为 FALSE。（两个输入字符串的长度之和超出

输出字符串所允许的最大长度。）

应用举例：CONCAT_IN1 中的字符串 ABCD 和 CONCAT_IN2 中的字符串 EFGH 合并后，输出结果为 ABCDEFGH。



7.2、INSERT 插入字符串功能

功能：这个字符串功能在给定的字符串 IN1 里插入一个字符串 IN2。 IN2 被插入到 IN1 中的字符位置 P 之后。

参数	**数据类型**	**描述**
IN1	STRING	输入字符串。
IN2	STRING	要插入 IN1 里的第二字符串。
P	ANY_INT	字符串 IN2 的插入位置
OUT	STRING	输出字符串。

注：P 不能为 0。字符串中的第一位置为 1。如果想在另一个字符串之前插入一个字符串，请使用 CONCAT 功能。

无法将同一字符串同时作为输入和输出字符串。 在这种情况里，应该在输出中使用中间变量。 然后，将这个中间变量赋与输入变量。

出错反应和处理：如果以下错误条件为真，则出现字符串错误：

- len(IN1) + len(IN2) > maxlen(OUT)
- P <= 0
- P > len(IN1)
- IN2 == OUT

在这里“len(...)”代表实际字符串长度，而“maxlen(...)”是允许的最大字符串长度（maxlen(Default string) = 80）。

如果发生输入错误，则结果变量为空串“”，并且发生字符串错误。 在这种情况下，SPG 21 被调用，并且在错误目录内产生一个包含相应模块和相关行号的条目。 PLC 保持 RUN 状态。

注：这个功能只能用于 IPC_28、M68_28 和更高的版本。

对于 IPC_28、M68_28 及较早版本的字符串错误的纠错处理：在不同 PLC 版本内，对字符串错误的纠错处理是不同的。 对于 IPC_28、M68_28 及较早版本，会出现下面的情况：

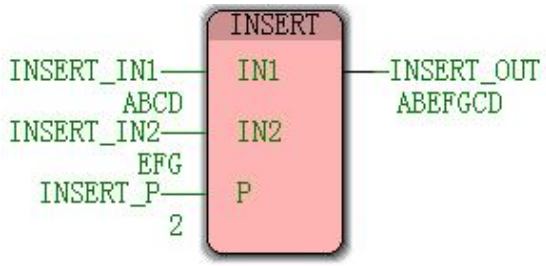
- 如果发生输入错误，则导致字符串出错。 相应的 PLC 将进入 STOP 状态。
- 如果值不在输出数据类型的有效范围内，则发生溢出。

EN/ENO 行为：如果 EN/ENO 被激活（“选项|图形编辑器设置），只有当 EN 输入被置为 TRUE 时，才执行此功能。 如果 EN = FALSE，保留上一次执行功能的输出值。

在以下情况 ENO 输出为 FALSE

- len(IN1) + len(IN2) > maxlen(OUT)
- P <= 0
- P > len(IN1)
- IN2 == OUT

应用举例：在 INSERT_IN1 中的字符串 ABCD 的第二位开始插入 INSERT_IN2 中的字符串 EFG，输出结果为 ABEFGCD。



7.3、DELETE 删除字符串功能

功能：这个字符串功能从给定的字符串里删除一个子串。 删除 IN 中的开始于字符位置 P 的 L 个字符。

参数	**数据类型**	**描述**
IN	STRING	输入字符串。
L	ANY_INT	要删除的字符数。
P	ANY_INT	要删除的第一字符的位置。
OUT	STRING	输出字符串。

出错反应和处理：如果以下错误条件为真，则出现字符串错误：

- len(IN) > maxlen(OUT)
- P + L > len(IN) + 1
- P <= 0
- L < 0

在这里“len(...)”代表实际字符串长度，而“maxlen(...)”是允许的最大字符串长度（maxlen(Default string) = 80）。

如果发生输入错误，则结果变量为空串“”，并且发生字符串错误。 在这种情况下，SPG 21 被调用，并且在错误目录内产生一个包含相应模块和相关行号的条目。 PLC 保持 RUN 状态。

注：这个功能只能用于 IPC_28、M68_28 和更高的版本。

对于 IPC_28、M68_28 及较早版本的字符串错误的纠错处理：在不同 PLC 版本内，对字符串错误的纠错处理是不同的。 对于 IPC_28、M68_28 及较早版本，会出现下面的情况：

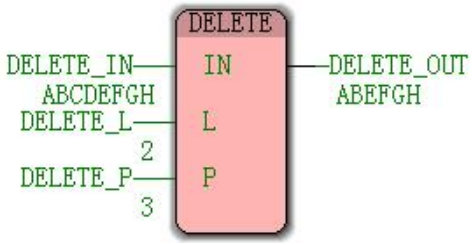
- 如果发生输入错误，则导致字符串出错。 相应的 PLC 将进入 STOP 状态。
- 如果值不在输出数据类型的有效范围内，则发生溢出。

EN/ENO 行为：如果 EN/ENO 被激活（“选项|图形编辑器设置），只有当 EN 输入被置为 TRUE 时，才执行此功能。 如果 EN = FALSE，保留上一次执行功能的输出值。

在以下情况 ENO 输出为 FALSE

- len(IN) > maxlen(OUT)
  （输入字符串的长度减去待删除子串的长度超出输出字符串所允许的最大长度。）
- P + L > len(IN) + 1
  （输入字符串的长度加 1 小于长度与位置之和再减去 1。）
- P <= 0
  （位置的值小于或等于零。）

应用举例：在 DELETE_IN1 中的字符串 ABCDEFGH 中，删除从第三个字符开始，长度为 2 的字符串，并输出结果为 ABEFGH。



7.4、REPLACE 替换字符串功能

功能：该字符串功能用输入 IN2 中的字符串替换输入 IN1 中的一个子串。 需要替换的字符数由输入 L 规定。替换起始位为字符位置 P。

参数	**数据类型**	**描述**
IN1	STRING	输入字符串。
IN2	STRING	要替换的子串。
L	ANY_INT	要替换的字符数。

P	ANY_INT	要替换的其实字符的位置。
OUT	STRING	输出字符串。

注：P 不能为 0，字符串中的第一位置是 1。

不能将同一字符串同时作为输入和输出字符串。 在这种情况里，应该在输出中使用中间变量。 然后，将这个中间变量赋与输入变量。

出错反应和处理：如果以下错误条件为真，则出现字符串错误：

- len(IN1) + len(IN2) > maxlen(OUT)
- IN2 == OUT
- P + L > len(IN) + 1
- P <= 0
- L < 0

在这里“len(...)”代表实际字符串长度，而“maxlen(...)”是允许的最大字符串长度（maxlen(Default string) = 80）。

如果发生输入错误，则结果变量为空串“”，并且发生字符串错误。 在这种情况下，SPG 21 被调用，并且在错误目录内产生一个包含相应模块和相关行号的条目。 PLC 保持 RUN 状态。

注：这个功能只能用于 IPC_28、M68_28 和更高的版本。

对于 IPC_28、M68_28 及较早版本的字符串错误的纠错处理：在不同 PLC 版本内，对字符串错误的纠错处理是不同的。 对于 IPC_28、M68_28 及较早版本，会出现下面的情况：

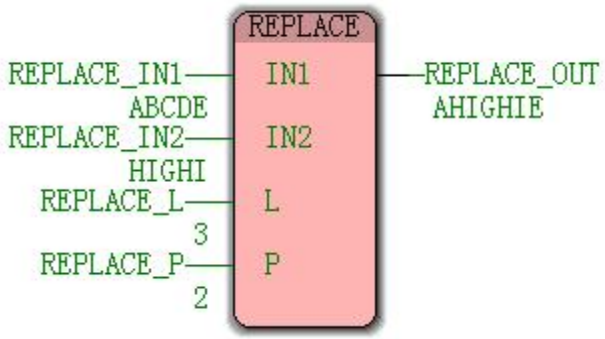
- 如果发生输入错误，则导致字符串出错。 相应的 PLC 将进入 STOP 状态。
- 假如值不在输出数据类型的有效范围内，则发生溢出。

EN/ENO 行为：如果 EN/ENO 被激活（“选项|图形编辑器设置），只有当 EN 输入被置为 TRUE 时，才执行此功能。 如果 EN = FALSE，保留上一次执行功能的输出值。

在以下情况 ENO 输出为 FALSE

- len(IN1) + len(IN2) > maxlen(OUT)
- IN2 == OUT
- P + L > len(IN1) +1
- P <= 0
- L < 0

应用举例：在 REPLACE_IN1 中的字符串 ABCDE 中，用 REPLACE_IN2 中的字符串 HIGHI 来替换从第二个字符开始，三个长度的字符串 BCD，并输出结果为 AHIGHIE。



7.5、LEN 计算字符串长度功能

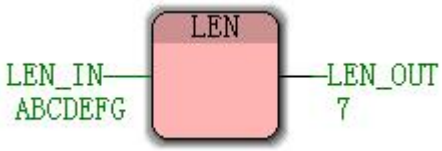
功能：这个字符串功能测定字符串的长度。

参数	**数据类型**	**描述**
IN1	STRING	输入字符串。
OUT	INT	IN1 中字符串的长度。

注：这个功能只能用于 IPC_28、M68_28 和更高的版本。

EN/ENO 行为：如果 EN/ENO 被激活（“选项|图形编辑器设置），只有当 EN 输入被置为 TRUE 时，才执行此功能。 如果 EN = FALSE，保留上一次执行功能的输出值。

应用举例：计算输入 LEN_IN 中字符串 ABCDEFG 的长度，并将结果 7 作为输出。



7.6、FIND 查找字符串中出现的一个字符功能

功能：这个字符串功能检测子串 IN2 在给定字符串 IN1 中的位置。 在 IN1 里第一次出现 IN2 字符的位置被保存到 OUT。

如果没出现 IN2 中的子串，那么 OUT:=0。

参数	**数据类型**	**描述**
IN1	STRING	输入字符串。
IN2	STRING	要查找的子串。
OUT	INT	第一次出现子串的字符位置。

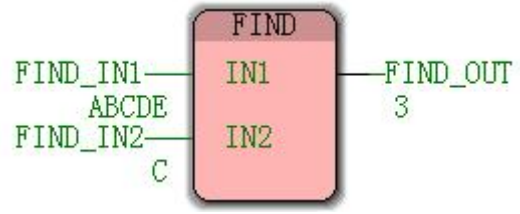
注：在字符串里的第一位置是 1。

这个功能只能用于 IPC_28、M68_28 和更高的版本。

EN/ENO 行为：如果 EN/ENO 被激活（“选项|图形编辑器设置），只有当 EN 输入被置为 TRUE 时，才执

行此功能。 如果 EN = FALSE，保留上一次执行功能的输出值。

应用举例：在 FIND_IN1 字符串 ABCDE 中查找 FIND_IN2 字符串 C，并将其位置 3 作为输出。



7.7、LEFT 取出字符串中最左边的几个字符

功能：该字符串功能可提取 IN 字符串中最左边的字符。IN 字符串中提取的字符数由输入 L 规定。

参数	**数据类型**	**描述**
IN	STRING	输入字符串。
L	ANY_INT	要提取的字符数。
OUT	STRING	输出字符串。

出错反应和处理：如果以下错误条件为真，则出现字符串错误：

- L > len(IN)
- L > maxlen(OUT)

在这里“len(IN)”代表实际字符串长度，而“maxlen(OUT)”是允许的最大字符串长度（maxlen(Default string) = 80）。

如果发生输入错误，则结果变量为空串“”，并且发生字符串错误。 在这种情况下，SPG 21 被调用，并且在错误目录内产生一个包含相应模块和相关行号的条目。 PLC 保持 RUN 状态。

注：这个功能只能用于 IPC_28、M68_28 和更高的版本。

对于 IPC_28、M68_28 及较早版本的字符串错误的纠错处理：在不同 PLC 版本内，对字符串错误的纠错处理是不同的。 对于 IPC_28、M68_28 及较早版本，会出现下面的情况：

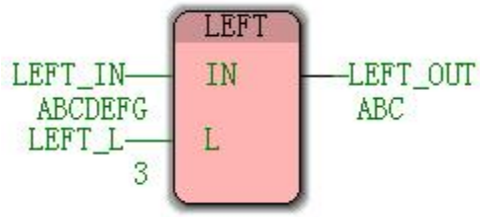
- 如果发生输入错误，则导致字符串出错。 相应的 PLC 将进入 STOP 状态。
- 如果值不在输出数据类型的有效范围内，则发生溢出。

EN/ENO 行为：如果 EN/ENO 被激活（“选项|图形编辑器设置），只有当 EN 输入被置为 TRUE 时，才执行此功能。 如果 EN = FALSE，保留上一次执行功能的输出值。

在以下情况 ENO 输出为 FALSE

- L > len(IN)
- L > maxlen(OUT)

应用举例：在 LEFT_IN 字符串 ABCDEFG 中取出最左边的 3 个字符，并将结果 ABC 作为输出。



7.8、MID 取出字符串中的几个字符

功能：该字符串功能可从输入 IN 中的字符串的中间位置提取子串。提取字符的数量由输入 L 规定。提取的起始位为字符位置 P。

参数	**数据类型**	**描述**
IN	STRING	输入字符串。
L	ANY_INT	要提取的字符数。
P	ANY_INT	要提取的第一个字符位置。
OUT	STRING	输出字符串。

注：P 不能为 0，字符串中的第一位置是 1。

不能将同一字符串同时作为输入和输出字符串。 在这种情况里，应该在输出中使用中间变量。 然后，将这个中间变量赋与输入变量。

出错反应和处理：如果以下错误条件为真，则出现字符串错误：

- L > maxlen(OUT)
- P <= 0
- P + L > len(IN) + 1

在这里“len(IN)”代表实际字符串长度，而“maxlen(OUT)”是允许的最大字符串长度（maxlen(Default string) = 80）。

如果发生输入错误，则结果变量为空串“”，并且发生字符串错误。 在这种情况下，SPG 21 被调用，并且在错误目录内产生一个包含相应模块和相关行号的条目。 PLC 保持 RUN 状态。

注：这个功能只能用于 IPC_28、M68_28 和更高的版本。

对于 IPC_28、M68_28 及较早版本的字符串错误的纠错处理：在不同 PLC 版本内，对字符串错误的纠错处理是不同的。 对于 IPC_28、M68_28 及较早版本，会出现下面的情况：

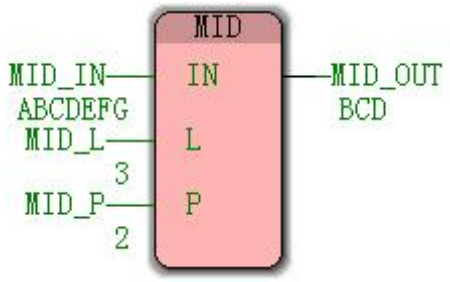
- 如果发生输入错误，则导致字符串出错。 相应的 PLC 将进入 STOP 状态。
- 如果值不在输出数据类型的有效范围内，则发生溢出。

EN/ENO 行为：如果 EN/ENO 被激活（“选项|图形编辑器设置），只有当 EN 输入被置为 TRUE 时，才执行此功能。 如果 EN = FALSE，保留上一次执行功能的输出值。

在以下情况 ENO 输出为 FALSE

- L > maxlen(OUT)
- P <= 0
- P + L > len(IN) + +1

应用举例：在 MID_IN 字符串 ABCDEFG 中取出从第 2 个字符开始，长度为 3 的字符，并将结果 BCD 作为输出。



7.9、RIGHT 取出字符串中最右边的几个字符

功能：该字符串功能可提取 IN 字符串中最右边的字符。IN 字符串中提取的字符数由输入 L 规定。

参数	**数据类型**	**描述**
IN	STRING	输入字符串。
L	ANY_INT	要提取的字符数。
OUT	STRING	输出字符串。

出错反应和处理：如果以下错误条件为真，则出现字符串错误：

- L > len(IN)
- L > maxlen(OUT)

在这里“len(IN)”代表实际字符串长度，而“maxlen(OUT)”是允许的最大字符串长度（maxlen(Default string) = 80）。

如果发生输入错误，则结果变量为空串“”，并且发生字符串错误。 在这种情况下，SPG 21 被调用，并且在错误目录内产生一个包含相应模块和相关行号的条目。 PLC 保持 RUN 状态。

注：这个功能只能用于 IPC_28、M68_28 和更高的版本。

对于 IPC_28、M68_28 及较早版本的字符串错误的纠错处理：在不同 PLC 版本内，对字符串错误的纠错处理是不同的。 对于 IPC_28、M68_28 及较早版本，会出现下面的情况：

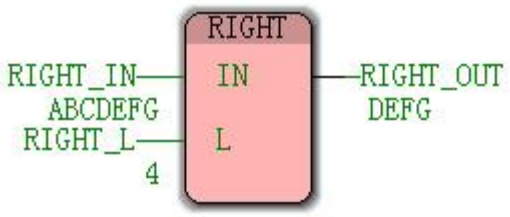
- 如果发生输入错误，则导致字符串出错。 相应的 PLC 将进入 STOP 状态。
- 如果值不在输出数据类型的有效范围内，则发生溢出。

EN/ENO 行为：如果 EN/ENO 被激活（“选项|图形编辑器设置），只有当 EN 输入被置为 TRUE 时，才执行此功能。 如果 EN = FALSE，保留上一次执行功能的输出值。

在以下情况 ENO 输出为 FALSE

- L > len(IN)
- L > maxlen(OUT)

应用举例：在 RIGHT_IN 字符串 ABCDEFG 中取出最右边的 4 个字符，并将结果 DEFG 作为输出。



7.10、GT_STRING 字符串大于

功能：该字符串比较功能将输入 IN1 中的字符串与输入 IN2 的字符串进行比较，如果 IN1 大于 IN2，在 OUT 里设置为 TRUE，否则设置为 FALSE。 从左到右完成比较。

从左到右完成比较。 字符“Z”大于字符“A″。 因此，字符串“Z”大于“AZ”，并且“AZ”大于“ABC”。

要比较长度不相等的两个字符串，应该用值为零的字符，将较短的字符串向右边扩展到与较长字符串的相同长度的位置。

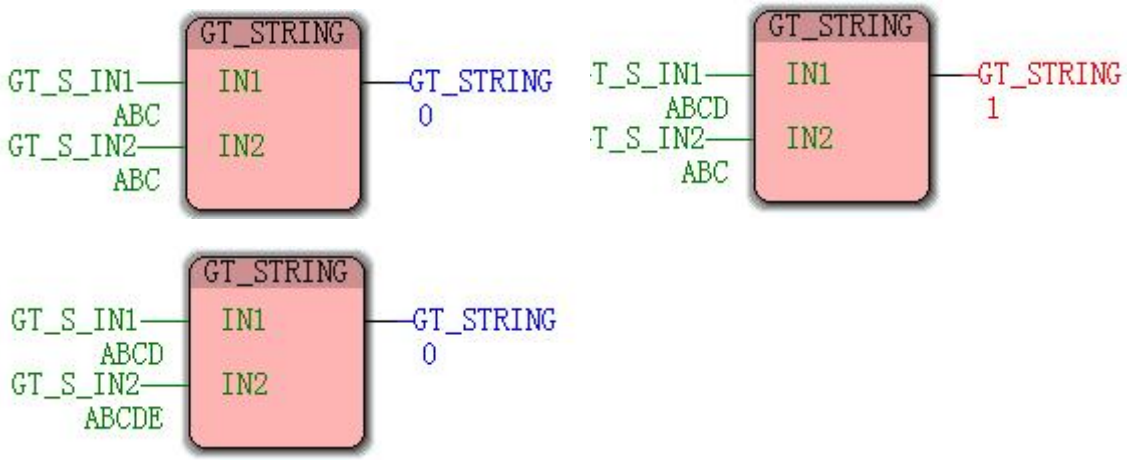
参数	**数据类型**	**描述**
IN1	STRING	第一输入字符串。
IN2	STRING	第二输入字符串。
OUT	BOOL	如果 IN1 大于 IN2，则为 TRUE。 如果 IN1 小于或等于 IN2，则为 FALSE。

注：输出 OUT 可以求反。

EN/ENO 行为：如果 EN/ENO 被激活（“选项|图形编辑器设置），只有当 EN 输入被置为 TRUE 时，才执行此功能。 如果 EN = FALSE，保留上一次执行功能的输出值。

依据程序逻辑，有可能需要对 ENO 输出求值。

应用举例：比较 GT_S_IN1 和 GT_S_IN2 的大小，若 GT_S_IN1 大于 GT_S_IN2，则输出为 1，若 GT_S_IN1 小于或者等于 GT_S_IN2，则输出为 0。



7.11、GE_STRING 字符串大于等于

功能：该字符串比较功能将输入 IN1 中的字符串与输入 IN2 的字符串进行比较，如果 IN1 大于或等于 IN2，在 OUT 里设置为 TRUE，否则设置为 FALSE。

从左到右完成比较。 字符“Z”大于字符“A”。 因此，字符串“Z”大于“AZ”，并且“AZ”大于“ABC”。

要比较长度不相等的两个字符串，应该用值为零的字符，将较短的字符串向右边扩展到与较长字符串的相同长度的位置。

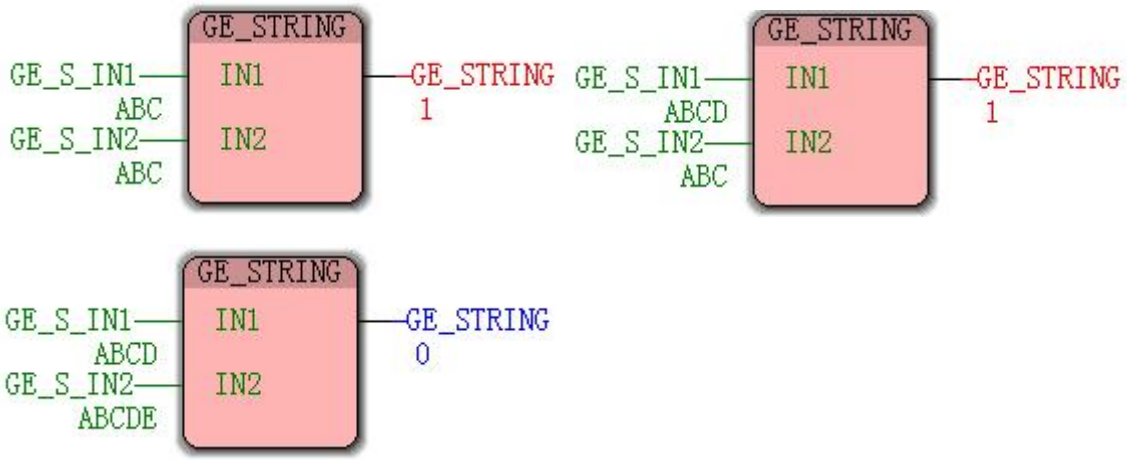
参数	**数据类型**	**描述**
IN1	STRING	第一输入字符串。
IN2	STRING	第二输入字符串。
OUT	BOOL	如果 IN1 大于或者等于 IN2，则为真 TRUE。 如果 IN1 小于 IN2，则为 FALSE 假。

注：输出 OUT 可以求反。

EN/ENO 行为：如果 EN/ENO 被激活（“选项|图形编辑器设置），只有当 EN 输入被置为 TRUE 时，才执行此功能。 如果 EN = FALSE，保留上一次执行功能的输出值。

依据程序逻辑，有可能需要对 ENO 输出求值。

应用举例：比较 GE_S_IN1 和 GE_S_IN2 的大小，若 GE_S_IN1 大于或者等于 GE_S_IN2，则输出为 1，若 GE_S_IN1 小于 GE_S_IN2，则输出为 0。



7.12、EQ_STRING 字符串等于

功能：该字符串比较功能将输入 IN1 中的字符串与输入 IN2 的字符串进行比较，如果 IN1 等于 IN2，在 OUT 里设置为 TRUE，否则设置为 FALSE。

从左到右完成比较。 字符“Z”大于字符“A”。 因此，字符串“Z”大于“AZ”，并且“AZ”大于“ABC”。

要比较长度不相等的两个字符串，应该用值为零的字符，将较短的字符串向右边扩展到与较长字符串的相同长度的位置。

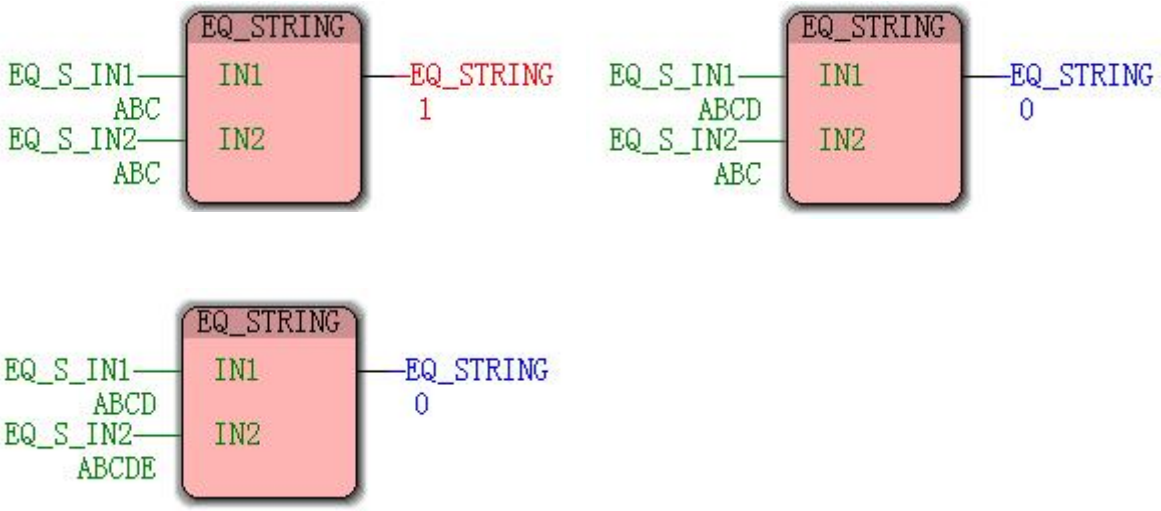
参数	数据类型	描述
IN1	STRING	第一输入字符串。
IN2	STRING	第二输入字符串。
OUT	BOOL	如果输入字符串相等，则为 TRUE。 如果输入字符串不相等，则为 FALSE。

注：输出 OUT 可以求反。

EN/ENO 行为：如果 EN/ENO 被激活（“选项|图形编辑器设置），只有当 EN 输入被置为 TRUE 时，才执行此功能。 如果 EN = FALSE，保留上一次执行功能的输出值。

依据程序逻辑，有可能需要对 ENO 输出求值。

应用举例:比较 EQ_S_IN1 和 EQ_S_IN2 的是否相等,若 EQ_S_IN1 等于 EQ_S_IN2,则输出为 1,若 EQ_S_IN1 不等于 EQ_S_IN，则输出为 0。



7.13、NE_STRING 字符串不等于

功能：该字符串比较功能将输入 IN1 中的字符串与输入 IN2 的字符串进行比较，如果 IN1 不等于 IN2，在 OUT 里设置为 TRUE，否则设置为 FALSE。

从左到右完成比较。 字符“Z”大于字符“A”。 因此，字符串“Z”大于“AZ”，并且“AZ”大于“ABC”。

要比较长度不相等的两个字符串，应该用值为零的字符，将较短的字符串向右边扩展到与较长字符串的相同长度的位置。

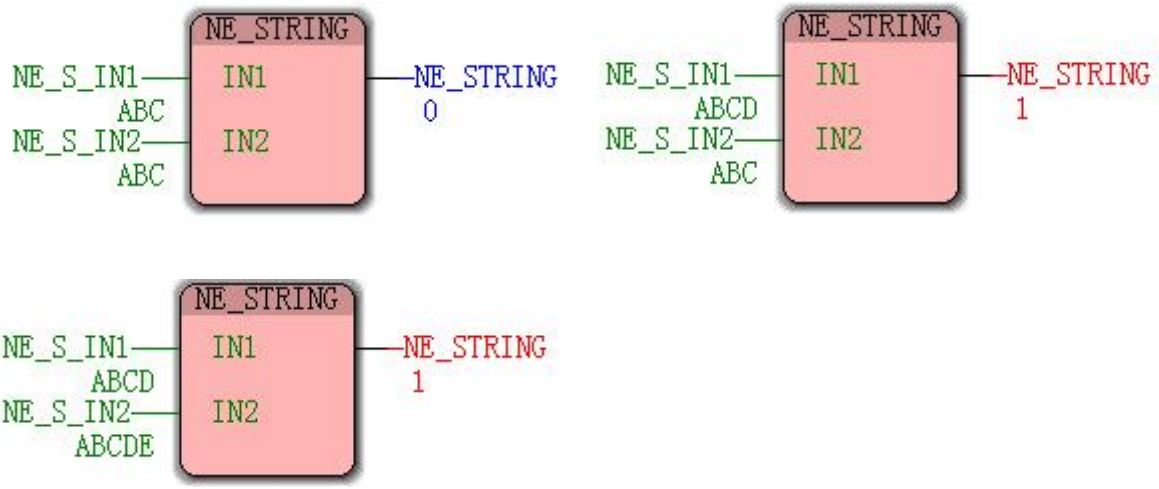
参数	数据类型	描述
IN1	STRING	第一输入字符串。
IN2	STRING	第二输入字符串。
OUT	BOOL	如果 IN1 和 IN2 是不相等的，则为 TRUE。 如果 IN1 和 IN2 相等，则为 FALSE。

注：输出 OUT 可以求反。

EN/ENO 行为：如果 EN/ENO 被激活（“选项|图形编辑器设置），只有当 EN 输入被置为 TRUE 时，才执行此功能。 如果 EN = FALSE，保留上一次执行功能的输出值。

依据程序逻辑，有可能需要对 ENO 输出求值。

应用举例:比较 NE_S_IN1 和 NE_S_IN2 的是否相等,若 NE_S_IN1 等于 NE_S_IN2,则输出为 0,若 NE_S_IN1 不等于 NE_S_IN，则输出为 1。



7.14、LE_STRING 字符串小于等于

功能：该字符串比较功能将输入 IN1 中的字符串与输入 IN2 的字符串进行比较，如果 IN1 小于或等于 IN2，在 OUT 里设置为 TRUE，否则设置为 FALSE。 从左到右完成比较。

从左到右完成比较。 字符“Z”大于字符“A”。 因此，字符串“Z”大于“AZ”，并且“AZ”大于“ABC”。

要比较长度不相等的两个字符串，应该用值为零的字符，将较短的字符串向右边扩展到与较长字符串的相同长度的位置。

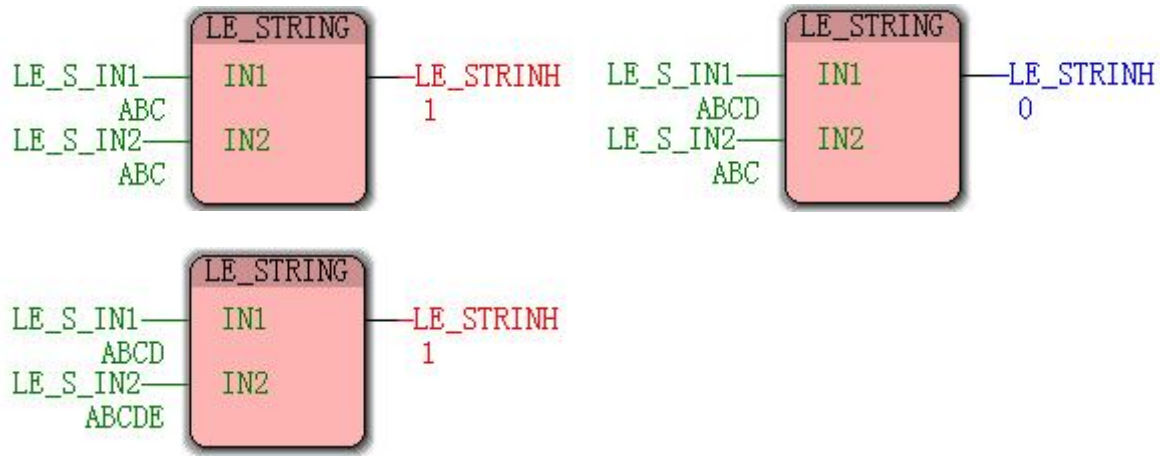
参数	数据类型	描述
IN1	STRING	第一输入字符串。
IN2	STRING	第二输入字符串。
OUT	BOOL	如果 IN1 小于或等于 IN2，则为真 TRUE。 如果 IN1 大于 IN2，则为 FALSE。

注：输出 OUT 可以求反。

EN/ENO 行为：如果 EN/ENO 被激活（“选项|图形编辑器设置），只有当 EN 输入被置为 TRUE 时，才执行此功能。 如果 EN = FALSE，保留上一次执行功能的输出值。

依据程序逻辑，有可能需要对 ENO 输出求值。

应用举例：比较 LE_S_IN1 和 LE_S_IN2 的大小，若 LE_S_IN1 大于 LE_S_IN2，则输出为 0，若 LE_S_IN1 小于或等于 LE_S_IN2，则输出为 1。



7.15、LT_STRING 字符串小于

功能：该字符串比较功能将输入 IN1 中的字符串与输入 IN2 的字符串进行比较，如果 IN1 小于 IN2，在 OUT 里设置为 TRUE，否则设置为 FALSE。

从左到右完成比较。 字符“Z”大于字符“A”。 因此，字符串“Z”大于“AZ”，并且“AZ”大于“ABC”。

要比较长度不相等的两个字符串，应该用值为零的字符，将较短的字符串向右边扩展到与较长字符串的相同长度的位置。

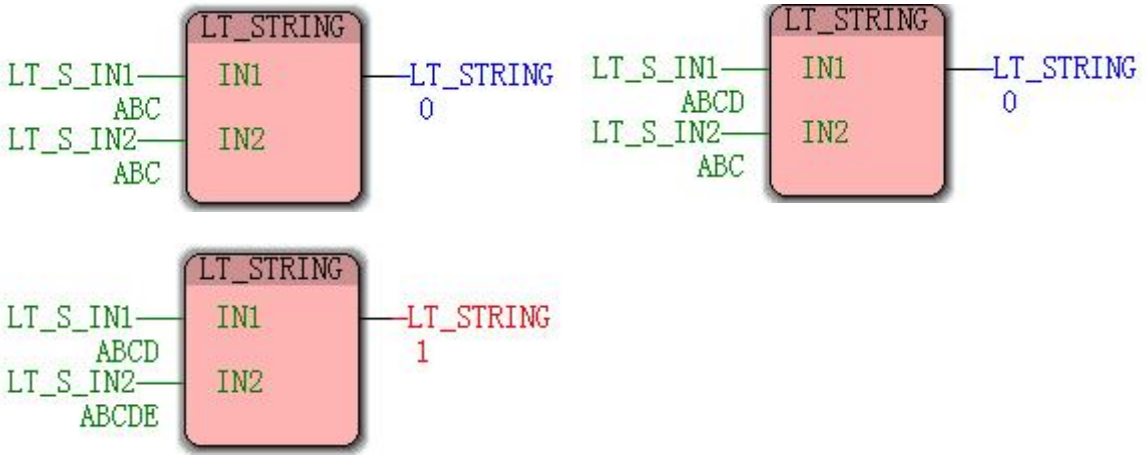
参数	数据类型	描述
IN1	STRING	第一输入字符串。
IN2	STRING	第二输入字符串。
OUT	BOOL	如果 IN1 小于 IN2，则为 TRUE。 如果 IN1 大于或等于 IN2，则为 FALSE。

注：输出 OUT 可以求反。

EN/ENO 行为：如果 EN/ENO 被激活（“选项|图形编辑器设置），只有当 EN 输入被置为 TRUE 时，才执行此功能。 如果 EN = FALSE，保留上一次执行功能的输出值。

依据程序逻辑，有可能需要对 ENO 输出求值。

应用举例：比较 LT_S_IN1 和 LT_S_IN2 的大小，若 LT_S_IN1 大于或等于 LT_S_IN2，则输出为 0，若 LT_S_IN1 小于 LT_S_IN2，则输出为 1。



8、类型转换功能

8.1、BYTE 型 BCD 数据的转换

BYTE 型 BCD 数据的转换包括以下三个指令：B_BCD_TO_SINT、B_BCD_TO_INT 和 B_BCD_TO_DINT。

这三个指令能将一个 BYTE 数据类型的 BCD（二进制编码的十进制数）输入值分别转换为一个 SINT、INT 和 DINT 数据类型的输出值，如 BCD 码 16#98 将被转换为十进制 98。

BCD 码每一位都不能大于 9，如 16#AB 的 BCD 码就是错误的。

B_BCD_TO_SINT

功能:这个类型转换功能将数据类型 BYTE 的 BCD 输入值(二进制编码的十进制数)转换为数据类型 SINT 的输出值。

输入	数据类型	描述
EN	BOOL	TRUE有效。如果不带EN/ENO端，则B_BCD_TO_SINT指令一直执行
IN	BYTE型BCD码	输入值，BCD码16#0~16#99，超出此范围转换失败
输出	数据类型	描述
ENO	BOOL	B_BCD_TO_SINT指令成功执行后，为TRUE，否则为FALSE
OUT	SINT	结果，有效范围0~99，转换失败后为-1

B_BCD_TO_INT

功能:这个类型转换功能将数据类型 BYTE 的 BCD 输入值（二进制编码的十进制数）转换为数据类型 INT

的输出值。

输入	数据类型	描述
EN	BOOL	TRUE有效。如果不带EN/ENO端，则B_BCD_TO_INT指令一直执行
IN	BYTE型BCD码	输入值，BCD码16#0~16#99，超出此范围转换失败
输出	数据类型	描述
ENO	BOOL	B_BCD_TO_INT指令成功执行后，为TRUE，否则为FALSE
OUT	INT	结果，有效范围0~99，转换失败后为-1

B_BCD_TO_DINT

功能:这个类型转换功能将数据类型 BYTE 的 BCD 输入值(二进制编码的十进制数)转换为数据类型 DINT 的输出值。

输入	数据类型	描述
EN	BOOL	TRUE有效。如果不带EN/ENO端，则B_BCD_TO_DINT指令一直执行
IN	BYTE型BCD码	输入值，BCD码有效范围16#0~16#99，超出此范围转换失败
输出	数据类型	描述
ENO	BOOL	B_BCD_TO_DINT指令成功执行后，为TRUE，否则为FALSE
OUT	DINT	结果，有效范围0~99，转换失败后为-1

8.2、WORD 型 BCD 数据的转换

WORD 型 BCD 数据的转换包括三个指令：W_BCD_TO_SINT、W_BCD_TO_INT 和 W_BCD_TO_DINT。

这三个指令能将一个 WORD 数据类型的 BCD(二进制编码的十进制数)输入值分别转换为一个 SINT、INT 和 DINT 数据类型的输出值，如 BCD 码 16#98 将被转换为十进制 98。BCD 码每一位都不能大于 9，如 16#AB 的 BCD 码就是错误的。

W_BCD_TO_SINT

功能:这个类型转换功能将数据类型 WORD 的 BCD 输入值(二进制编码的十进制数)转换为数据类型 SINT 的输出值。

输入	数据类型	描述
EN	BOOL	TRUE有效。如果不带EN/ENO端，则W_BCD_TO_SINT指令一直执行
IN	WORD型BCD码	输入值，BCD码有效范围16#0~16#127，超出此范围转换失败
输出	数据类型	描述
ENO	BOOL	W_BCD_TO_SINT指令成功执行后，为TRUE，否则为FALSE
OUT	SINT	结果，有效范围0~127，转换失败后为-1

W_BCD_TO_INT

功能：这个类型转换功能将数据类型 WORD 的 BCD 输入值（二进制编码的十进制数）转换为数据类型 INT 的输出值。

输入	数据类型	描述
EN	BOOL	TRUE有效。如果不带EN/ENO端，则W_BCD_TO_INT指令一直执行
IN	WORD型BCD码	输入值，BCD码有效范围16#0000~9999，超出此范围转换失败
输出	数据类型	描述
ENO	BOOL	W_BCD_TO_INT指令成功执行后，为TRUE，否则为FALSE
OUT	INT	结果，有效范围0~9999，转换失败后为-1

W_BCD_TO_DINT

功能：这个类型转换功能将数据类型 WORD 的 BCD 输入值(二进制编码的十进制数)转换为数据类型 DINT 的输出值。

输入	数据类型	描述
EN	BOOL	TRUE有效。如果不带EN/ENO端，则W_BCD_TO_DINT指令一直执行
IN	WORD型BCD码	输入值，BCD码有效范围16#0000~9999，超出此范围转换失败
输出	数据类型	描述
ENO	BOOL	W_BCD_TO_DINT指令成功执行后，为TRUE，否则为FALSE
OUT	DINT	结果，有效范围0~9999，转换失败后为-1

8.3、DWORD 型 BCD 数据的转换

DWORD 型 BCD 数据的转换包括以下三个指令：D_BCD_TO_SINT、D_BCD_TO_INT 和 D_BCD_TO_DINT。

这三个指令能将一个 DWORD 数据类型的 BCD（二进制编码的十进制数）输入值分别转换为一个 SINT、INT 和 DINT 数据类型的输出值，如 BCD 码 16#98 将被转换为十进制 98。

BCD 码每一位都不能大于 9，如 16#AB 的 BCD 码就是错误的。

D_BCD_TO_SINT

功能：这个类型转换功能将数据类型 DWORD 的 BCD 输入值（二进制编码的十进制数）转换为数据类型 SINT 的输出值。

输入	数据类型	描述
EN	BOOL	TRUE有效。如果不带EN/ENO端，则D_BCD_TO_SINT指令一直执行
IN	DWORD型BCD码	输入值，BCD码有效范围16#0~16#127，超出此范围转换失败
输出	数据类型	描述
ENO	BOOL	D_BCD_TO_SINT指令成功执行后，为TRUE，否则为FALSE
OUT	SINT	结果，有效范围0~127，转换失败后为-1

D_BCD_TO_INT

功能：这个类型转换功能将数据类型 DWORD 的 BCD 输入值（二进制编码的十进制数）转换为数据类型 INT 的输出值。

输入	数据类型	描述
EN	BOOL	TRUE有效。如果不带EN/ENO端，则D_BCD_TO_INT指令一直执行
IN	DWORD型BCD码	输入值，BCD码有效范围16#0~16#32767，超出此范围转换失败
输出	数据类型	描述
ENO	BOOL	D_BCD_TO_INT指令成功执行后，为TRUE，否则为FALSE
OUT	INT	结果，有效范围0~32767，转换失败后为-1

D_BCD_TO_DINT

功能：这个类型转换功能将数据类型 DWORD 的 BCD 输入值（二进制编码的十进制数）转换为数据类型 DINT 的输出值。

输入	数据类型	描述
EN	BOOL	TRUE有效。如果不带EN/ENO端，则D_BCD_TO_DINT指令一直执行
IN	DWORD型BCD码	输入值，BCD码有效范围16#0~16#99999999，超出此范围转换失败
输出	数据类型	描述
ENO	BOOL	D_BCD_TO_DINT指令成功执行后，为TRUE，否则为FALSE
OUT	DINT	结果，有效范围0~99999999，转换失败后为-1

8.4、BCD 型数据的转换

功能：这个类型转换功能将数据类型 DWORD 的 BCD 输入值（二进制编码的十进制数）转换为数据类型 DINT 的输出值。

输入	数据类型	描述
EN	BOOL	TRUE有效。如果不带EN/ENO端，则BCD_TO_DINT指令一直执行
IN	DWORD型BCD码	输入值，BCD码有效范围16#0~16#99999999，超出此范围转换失败
输出	数据类型	描述
ENO	BOOL	D_BCD_TO_DINT指令成功执行后，为TRUE，否则为FALSE
OUT	DINT	结果，有效范围0~99999999，转换失败后为-1

8.5、BOOL 型数据的转换

功能：BOOL 型数据的转换有 11 个指令，可以将 BOOL 型数据分别转换为 BYTE、WORD、DWORD、SINT、INT、DINT、USINT、UINT、UDINT、REAL 和 LREAL

指令	输入值	输出值	描述

BOOL_TO_BYTE	BOOL	BYTE	输入值取值范围FALSE或TRUE： 输入为FALSE时，输出为0； 输入为TRUE时，输出为1。
BOOL_TO_WORD	BOOL	WORD	
BOOL_TO_DWORD	BOOL	DWORD	
BOOL_TO_SINT	BOOL	SINT	
BOOL_TO_INT	BOOL	INT	
BOOL_TO_DINT	BOOL	DINT	
BOOL_TO_USINT	BOOL	USINT	
BOOL_TO_UINT	BOOL	UINT	
BOOL_TO_UDINT	BOOL	UDINT	
BOOL_TO_REAL	BOOL	REAL	
BOOL_TO_LREAL	BOOL	LREAL	

BOOL 型数据转换指令处理的数据类型（以 BOOL_TO_BYTE 为例）

输入	数据类型	描述
EN	BOOL	TRUE有效。如果不带EN/ENO端，则BOOL_TO_BYTE指令一直执行
IN	BOOL	输入值，FALSE或TRUE
输出	数据类型	描述
ENO	BOOL	BOOL_TO_BYTE指令成功执行后，为TRUE，否则为FALSE
OUT	BYTE	结果，IN为FALSE时，OUT=0，IN为TRUE时，OUT=1

8.6、BYTE 型数据的转换

功能：BYTE 型数据的转换有 11 个指令，可以将 BYTE 型数据分别转换为 BOOL、WORD、DWORD、SINT、INT、DINT、USINT、UINT、UDINT、REAL 和 LREAL 等类型。

输入值取值范围 0~255。

指令	输入值	输出值	描述
BYTE_TO_BOOL	BYTE	BOOL	IN=0时，OUT=FALSE；IN>0时，OUT=TRUE
BYTE_TO_WORD	BYTE	WORD	OUT=IN
BYTE_TO_DWORD	BYTE	DWORD	OUT=IN
BYTE_TO_SINT	BYTE	SINT	IN<=127时，OUT=IN；NI>127时，OUT=IN-256
BYTE_TO_INT	BYTE	INT	OUT=IN
BYTE_TO_DINT	BYTE	DINT	OUT=IN
BYTE_TO_USINT	BYTE	USINT	OUT=IN

BYTE_TO_UINT	BYTE	UINT	OUT=IN
BYTE_TO_UDINT	BYTE	UDINT	OUT=IN
BYTE_TO_REAL	BYTE	REAL	OUT=IN
BYTE_TO_LREAL	BYTE	LREAL	OUT=IN

BYTE 型数据转换指令处理的数据类型（以 BYTE_TO_BOOL 为例）

输入	数据类型	描述
EN	BOOL	TRUE有效。如果不带EN/ENO端，则BYTE_TO_BOOL指令一直执行
IN	BYTE	输入值，有效值0~255
输出	数据类型	描述
ENO	BOOL	BYTE_TO_BOOL指令成功执行后，为TRUE，否则为FALSE
OUT	BOOL	结果，输入0时，OUT=FALSE，其它情况下输出均为TRUE

8.7、WORD 型数据的转换

功能：WORD 型数据的转换有 11 个指令，可以将 WORD 型数据分别转换为 BOOL、BYTE、DWORD、SINT、INT、DINT、USINT、UINT、UDINT、REAL 和 LREAL 等类型。

输入值取值范围 0~65535。

指令	输入值	输出值	描述
WORD_TO_BOOL	WORD	BOOL	IN=0时，OUT=FALSE，其它情况下输出均为TRUE
WORD_TO_BYTE	WORD	BYTE	IN<=255时，OUT=IN；IN>255时，OUT=IN%256
WORD_TO_DWORD	WORD	DWORD	OUT=IN
WORD_TO_SINT	WORD	SINT	IN<=127时，OUT=IN；255>=IN>127时，OUT=IN-256；IN>255时，重复上述过程
WORD_TO_INT	WORD	INT	IN<=32767时，OUT=IN；IN>32767时，OUT=IN-65536
WORD_TO_DINT	WORD	DINT	OUT=IN
WORD _TO_USINT	WORD	USINT	IN<=255时，OUT=IN；IN>255时，OUT=IN%256
WORD _TO_UINT	WORD	UINT	OUT=IN
WORD _TO_UDINT	WORD	UDINT	OUT=IN
WORD _TO_REAL	WORD	REAL	OUT=IN
WORD_TO_LREAL	WORD	LREAL	OUT=IN

WORD 型数据转换指令处理的数据类型（以 WORD_TO_BOOL 为例）

输入	数据类型	描述
EN	BOOL	TRUE有效。如果不带EN/ENO端，则WORD_TO_BOOL指令一直执行
IN	WORD	输入值，有效值0~65535
输出	数据类型	描述
ENO	BOOL	WORD_TO_BOOL指令成功执行后，为TRUE，否则为FALSE
OUT	BOOL	结果，IN=0时，OUT=FALSE，其它情况下输出均为TRUE

8.8、DWORD 型数据的转换

功能：DWORD 型数据的转换有 11 个指令，可以将 DWORD 型数据分别转换为 BOOL、BYTE、WORD、SINT、INT、DINT、USINT、UINT、UDINT、REAL 和 LREAL 等类型。

输入值取值范围 0~4,294,967,295。

指令	输入值	输出值	描述
DWORD_TO_BOOL	DWORD	BOOL	IN=0时，OUT=FALSE，其它情况下输出均为TRUE
DWORD_TO_BYTE	DWORD	BYTE	IN<=255时，OUT=IN；IN>255时，OUT=IN%256
DWORD_TO_WORD	DWORD	WORD	IN<=65535时，OUT=IN；IN>65535时，OUT=IN%65536
DWORD_TO_SINT	DWORD	SINT	IN<=127时，OUT=IN；255>=IN>127时，OUT=IN-256；IN>255时，重复上述过程
DWORD_TO_INT	DWORD	INT	IN<=32767时，OUT=IN；IN>32767时，OUT=IN-65536；IN>65535时，重复上述过程
DWORD_TO_DINT	DWORD	DINT	IN<=2,147,483,647时，OUT=IN；IN>2,147,483,647时，OUT=IN%2,147,483,647
DWORD _TO_USINT	DWORD	USINT	IN<=255时，OUT=IN；IN>255时，OUT=IN%256
DWORD _TO_UINT	DWORD	UINT	IN<=65535时，OUT=IN；IN>65535时，OUT=IN%65535
DWORD _TO_UDINT	DWORD	UDINT	OUT=IN
DWORD _TO_REAL	DWORD	REAL	OUT=IN
DWORD_TO_LREAL	DWORD	LREAL	OUT=IN

DWORD 型数据转换指令处理的数据类型（以 DWORD_TO_BOOL 为例）

输入	数据类型	描述
EN	BOOL	TRUE有效。如果不带EN/ENO端，则DWORD_TO_BOOL指令一直执行
IN	DWORD	输入值，有效值0~4,294,967,295
输出	数据类型	描述
ENO	BOOL	DWORD_TO_BOOL指令成功执行后，为TRUE，否则为FALSE

OUT	BOOL	结果，IN=0时，OUT=FALSE，其它情况下输出均为TRUE

8.9、SINT 型数据的转换

功能：SINT 型数据的转换有 14 个指令，可以将 SINT 型数据分别转换为 B_BCD、W_BCD、D_BCD、BOOL、BYTE、WORD、DWORD、INT、DINT、USINT、UINT、UDINT、REAL 和 LREAL 等类型。

输入值取值范围-128~127。

指令	输入值	输出值	描述
SINT_TO_B_BCD	SINT	BYTE	0<=IN<=99时，OUT=INBCD；其它值时，OUT=16#FF
SINT_TO_W_BCD	SINT	WORD	0<=IN<=127时，OUT=INBCD；其它值时，OUT=16#FFFF
SINT_TO_D_BCD	SINT	DWORD	0<=IN<=127时，OUT=INBCD；其它值时，OUT=16#FFFF FFFF
SINT_TO_BOOL	SINT	BOOL	IN=0时，OUT=FALSE；其它值时，输出TRUE
SINT_TO_BYTE	SINT	BYTE	0<=IN<=127时，OUT=IN；-128<=IN<=-1时，OUT=IN+256
SINT_TO_WORD	SINT	WORD	0<=IN<=127时，OUT=IN；-128<=IN<=-1时，OUT=IN+256
SINT_TO_DWORD	SINT	DWORD	0<=IN<=127时，OUT=IN；-128<=IN<=-1时，OUT=IN+256
SINT_TO_INT	SINT	INT	0<=IN<=127时，OUT=IN；-128<=IN<=-1时，OUT=IN+256
SINT_TO_DINT	SINT	DINT	0<=IN<=127时，OUT=IN；-128<=IN<=-1时，OUT=IN+256
SINT_TO_USINT	SINT	USINT	0<=IN<=127时，OUT=IN；-128<=IN<=-1时，OUT=IN+256
SINT_TO_UINT	SINT	UINT	0<=IN<=127时，OUT=IN；-128<=IN<=-1时，OUT=IN+256
SINT_TO_UDINT	SINT	UDINT	0<=IN<=127时，OUT=IN；-128<=IN<=-1时，OUT=IN+256
SINT_TO_REAL	SINT	REAL	OUT=IN
SINT_TO_LREAL	SINT	LREAL	OUT=IN

SINT 型数据转换指令处理的数据类型（以 SINT_TO_B_BCD 为例）

输入	数据类型	描述
EN	BOOL	TRUE有效。如果不带EN/ENO端，则SINT_TO_B_BCD指令一直执行
IN	SIN	输入值，有效值-128~127
输出	数据类型	描述
ENO	BOOL	SINT_TO_B_BCD指令成功执行后，为TRUE，否则为FALSE
OUT	BCD	结果，0<=IN<=99时，OUT=IN_{BCD}；其它值时，OUT=16#FF

8.10、INT 型数据的转换

功能：INT 型数据的转换有 14 个指令，可以将 INT 型数据分别转换为 B_BCD、W_BCD、D_BCD、BOOL、

BYTE、WORD、DWORD、SINT、DINT、USINT、UINT、UDINT、REAL 和 LREAL 等类型。

输入值取值范围-32768~32767。

指令	输入值	输出值	描述
INT_TO_B_BCD	INT	BYTE	0<=IN<=99时，OUT=INBCD；其它值时，OUT=16#FF
INT_TO_W_BCD	INT	WORD	0<=IN<=9999时，OUT=INBCD；其它值时，OUT=16#FFFF
INT_TO_D_BCD	INT	DWORD	0<=IN<=32767时，OUT=INBCD；其它值时，OUT=16#FFFFFFFF
INT_TO_BOOL	INT	BOOL	IN=0时，OUT=FALSE；其它值时，OUT=TRUE
INT_TO_BYTE	INT	BYTE	0<=IN<=255时，OUT=IN；IN>255时，OUT=IN%256； -256<=IN<=-1时，OUT=IN+256；IN<-256时，输出重复255~0
INT_TO_WORD	INT	WORD	0<=IN<=32767时，OUT=IN；-32768<=IN<=-1时， OUT=IN+65536
INT_TO_DWORD	INT	DWORD	0<=IN<=32767时，OUT=IN；-32768<=IN<=-1时， OUT=IN+65536
INT_TO_SINT	INT	SINT	0<=IN<=127时，OUT=IN；128<=IN<=255时，OUT=IN-256； IN>=256时，OUT重复输出0~127，-128~-1；-128<=IN<=-1时，OUT=IN；-256<=IN<=-129，OUT=IN+256，IN<-256时OUT重复-1~-128，127~0
INT_TO_DINT	INT	DINT	OUT=IN
INT_TO_USINT	INT	USINT	0<=IN<=255时，OUT=IN；IN>255时，OUT=IN%256； -256<=IN<=-1时，OUT=IN+256；IN<-256时，输出重复255~0
INT_TO_UINT	INT	UINT	0<=IN<=32767时，OUT=IN；-32768<=IN<=-1时， OUT=IN+65536
INT_TO_UDINT	INT	UDINT	0<=IN<=32767时，OUT=IN；IN<0时转换失败
INT_TO_REAL	INT	REAL	OUT=IN
INT_TO_LREAL	INT	LREAL	OUT=IN

INT 型数据转换指令处理的数据类型（以 INT_TO_B_BCD 为例）

输入	数据类型	描述
EN	BOOL	TRUE有效。如果不带EN/ENO端，则INT_TO_B_BCD指令一直执行
IN	SIN	输入值，有效值-128~127
输出	数据类型	描述
ENO	BOOL	INT_TO_B_BCD指令成功执行后，为TRUE，否则为FALSE
OUT	BCD	结果，0<=IN<=99时，OUT=IN$_{BCD}$；其它值时，OUT=16#FF

8.11、DINT 型数据的转换

功能：DINT 型数据的转换有 16 个指令，可以将 DINT 型数据分别转换为 B_BCD、W_BCD、D_BCD、BOOL、BYTE、WORD、DWORD、SINT、INT、USINT、UINT、UDINT、REAL、LREAL、BCD 和 TIME 等类型。

输入值取值范围-2,147,483,648…2,147,483,647。

指令	输入值	输出值	描述
DINT_TO_B_BCD	DINT	BYTE	0<=IN<=99时，OUT=INBCD；其它值时，OUT=16#FF
DINT_TO_W_BCD	DINT	WORD	0<=IN<=9999时，OUT=INBCD；其它值时，OUT=16#FFFF
DINT_TO_D_BCD	DINT	DWORD	0<=IN<=99999999时，OUT=IN，其它值时，OUT=16#FFFFFFFF
DINT_TO_BOOL	DINT	BOOL	输入0时，输出FALSE；输入其它值时，输出TRUE
DINT_TO_BYTE	DINT	BYTE	0<=IN<=255时，OUT=IN；IN>255时，OUT=IN%256； -256<=IN<=-1时，OUT=IN+256；IN<-256时，输出重复255~0
DINT_TO_WORD	DINT	WORD	0<=IN<=65535时，OUT=65535；IN>65535时，OUT=IN%65536； -65536<=IN<=-1，OUT=IN+65536，输入再减小则输出重复65535~0
DINT_TO_DWORD	DINT	DWORD	输入0~2,147,483,647时，输出0~2,147,483,647；输入-2,147,483,648~-1，输出2,147,483,648~ 4,294,967,295
DINT_TO_SINT	DINT	SINT	输入0~127时，输出0~127；输入128~255，输出-128~-1；输入再增加则输出重复0~127，-128~-1；输入-1~-128，输出-1~-128；输入-129~-256，输出127~0；输入再减小则输出重复-1~-128，127~0
DINT_TO_INT	DINT	INT	输入-32768~32767时，输出-32768~32767；输入大于32767时，输出重复-32768~32767；输入小于-32768是，输出重复32767~-32768
DINT_TO_USINT	DINT	USINT	同DINT_TO_BYTE
DINT_TO_UINT	DINT	UINT	同DINT_TO_WORD
DINT_TO_UDINT	DINT	UDINT	同DINT_TO_DWORD
DINT_TO_REAL	DINT	REAL	输入0~2,147,483,647时，输出0~2,147,483,647；输入-2,147,483,648~-1，输出-2,147,483,648~-1，精度会降低
DINT_TO_LREAL	DINT	LREAL	输入0~2,147,483,647时，输出0~2,147,483,647；输入-2,147,483,648~-1，输出-2,147,483,648~-1，精度会降低
DINT_TO_BCD	DINT	BCD	输入值DINT 0~99999999，输出值BCD码 16#00000000~16#99999999。
DINT_TO_TIME	DINT	TIME	输出值单位为秒；输入值0~2,147,483,647，输出值

			0~2147483.647秒；输入值-2147483648~-1秒，输出值2147483.648~4294967.295秒

DINT 型数据转换指令处理的数据类型（以 DINT_TO_B_BCD 为例）

输入	数据类型	描述
EN	BOOL	TRUE有效。如果不带EN/ENO端，则DINT_TO_B_BCD指令一直执行
IN	SIN	输入值，有效值-128~127
输出	数据类型	描述
ENO	BOOL	DINT_TO_B_BCD指令成功执行后，为TRUE，否则为FALSE
OUT	BCD	结果，0<=IN<=99时，OUT=IN_{BCD}；其它值时，OUT=16#FF

8.12、SUINT 型数据的转换

功能：USINT 型数据的转换有 11 个指令，可以将 USINT 型数据分别转换为 BOOL、BYTE、WORD、DWORD、SINT、INT、DINT、UINT、UDINT、REAL 和 LREAL 等类型。

输入值取值范围 0~255。

指令	输入值	输出值	描述
USINT_TO_BOOL	USINT	BOOL	IN=0时，OUT=FALSE；IN>0时，OUT=TRUE
USINT_TO_BYTE	USINT	BYTE	OUT=IN
USINT_TO_WORD	USINT	WORD	OUT=IN
USINT_TO_DWORD	USINT	DWORD	OUT=IN
USINT_TO_SINT	USINT	SINT	0<=IN<=127时，OUT=IN；127<IN<256，OUT=IN-256输出-128~-1
USINT_TO_INT	USINT	INT	OUT=IN
USINT_TO_DINT	USINT	DINT	OUT=IN
USINT_TO_UINT	USINT	UINT	OUT=IN
USINT_TO_UDINT	USINT	UDINT	OUT=IN
USINT_TO_REAL	USINT	REAL	OUT=IN
USINT_TO_LREAL	USINT	LREAL	OUT=IN

USINT 型数据转换指令处理的数据类型（以 USINT_TO_BYTE 为例）

输入	数据类型	描述
EN	BOOL	TRUE有效。如果不带EN/ENO端，则USINT_TO_BYTE指令一直执行
IN	SIN	输入值，有效值0~255

输出	数据类型	描述
ENO	BOOL	USINT_TO_BYTE指令成功执行后，为TRUE，否则为FALSE
OUT	BCD	结果，OUT=IN

8.13、UINT 型数据的转换

功能：UINT 型数据的转换有 11 个指令，可以将 UINT 型数据分别转换为 BOOL、BYTE、WORD、DWORD、SINT、INT、DINT、USINT、UDINT、REAL 和 LREAL 等类型。

输入值取值范围 0~65535。

指令	输入值	输出值	描述
UINT_TO_BOOL	UINT	BOOL	IN=0时，OUT=FALSE；其它值时，OUT=TRUE
UINT_TO_BYTE	UINT	BYTE	IN<=255时，OUT=IN；IN>255时，OUT=IN%256
UINT_TO_WORD	UINT	WORD	OUT=IN
UINT_TO_DWORD	UINT	DWORD	OUT=IN
UINT_TO_SINT	UINT	SINT	IN<=127时，OUT=IN；255>=IN>127时，OUT=IN-256；IN>255时，重复上述过程
UINT_TO_INT	UINT	INT	IN<=32767时，OUT=IN；IN>32767时，OUT=IN-65536
UINT_TO_DINT	UINT	DINT	OUT=IN
UINT_TO_USINT	UINT	USINT	IN<=255时，OUT=IN；IN>255时，OUT=IN%256
UINT_TO_UDINT	UINT	UDINT	OUT=IN
UINT_TO_REAL	UINT	REAL	OUT=IN
UINT_TO_LREAL	UINT	LREAL	OUT=IN

UINT 型数据转换指令处理的数据类型（以 UINT_TO_BYTE 为例）

输入	数据类型	描述
EN	BOOL	TRUE有效。如果不带EN/ENO端，则UINT_TO_BYTE指令一直执行
IN	SIN	输入值，有效值0~65535
输出	数据类型	描述
ENO	BOOL	UINT_TO_BYTE指令成功执行后，为TRUE，否则为FALSE
OUT	BCD	结果，IN<=255时，OUT=IN；IN>255时，OUT=IN%256

8.14、UDINT 型数据的转换

功能：UDINT 型数据的转换有 11 个指令，可以将 UDINT 型数据分别转换为 BOOL、BYTE、WORD、DWORD、

SINT、INT、DINT、USINT、UINT、REAL 和 LREAL 等类型。

输入值取值范围 0~4,294,967,295。

指令	输入值	输出值	描述
UDINT_TO_BOOL	UDINT	BOOL	IN=0时，OUT=FALSE，其它情况下输出均为TRUE
UDINT_TO_BYTE	UDINT	BYTE	IN<=255时，OUT=IN；IN>255时，OUT=IN%256
UDINT_TO_WORD	UDINT	WORD	IN<=65535时，OUT=IN；IN>65535时，OUT=IN%65536
UDINT_TO_DWORD	UDINT	DWORD	OUT=IN
UDINT_TO_SINT	UDINT	SINT	IN<=127时，OUT=IN；255>=IN>127时，OUT=IN-256； IN>255时，重复上述过程
UDINT_TO_INT	UDINT	INT	IN<=32767时，OUT=IN；IN>32767时，OUT=IN-65536； IN>65535时，重复上述过程
UDINT_TO_DINT	UDINT	DINT	IN<=2,147,483,647时，OUT=IN； IN>2,147,483,647时，OUT=IN%2,147,483,647
UDINT_TO_USINT	UDINT	USINT	IN<=255时，OUT=IN；IN>255时，OUT=IN%256
UDINT_TO_UINT	UDINT	UDINT	IN<=65535时，OUT=IN；IN>65535时，OUT=IN%65535
UDINT_TO_REAL	UDINT	REAL	OUT=IN
UDINT_TO_LREAL	UDINT	LREAL	OUT=IN

UDINT 型数据转换指令处理的数据类型（以 UDINT_TO_BYTE 为例）

输入	数据类型	描述
EN	BOOL	TRUE有效。如果不带EN/ENO端，则UDINT_TO_BYTE指令一直执行
IN	SIN	输入值，有效值0~4,294,967,295
输出	数据类型	描述
ENO	BOOL	UDINT_TO_BYTE指令成功执行后，为TRUE，否则为FALSE
OUT	BCD	结果，IN<=255时，OUT=IN；IN>255时，OUT=IN%256

8.15、REAL 型数据的转换

功能：REAL 型数据的转换有 11 个指令，可以将 REAL 型数据分别转换为 BOOL、BYTE、WORD、DWORD、SINT、INT、DINT、USINT、UINT、UDINT 和 LREAL 等类型。

REAL 型转换分为两步。 首先对实数进行四舍五入（小于等于 0.49 的数值被舍去，等于或者大于 0.50 的数值进一位）。 在第二步里，四舍五入的输入值被转换到输出值相应数据类型。

指令	输入值	输出值	描述
REAL_TO_BOOL	REAL	BOOL	IN=0时，OUT=FALSE，其它情况下输出均为TRUE

REAL_TO_BYTE	REAL	BYTE	IN<=255时，OUT=IN；IN>255时，OUT=IN%256
REAL _TO_WORD	REAL	WORD	IN<=65535时，OUT=IN；IN>65535时，OUT=IN%65536；输入-1~-65536，输出65535~0，输入再减小则输出重复65535~0
REAL _TO_DWORD	REAL	DWORD	输入0~2,147,483,647时，输出0~2,147,483,647，输入再增加输出为2,147,483,647；输入 -1~-2,147,483,648，输出 4,294,967,295~2,147,483,648，输入再减小输出仍为2,147,483,648；精度会降低
IREAL _TO_SINT	REAL	SINT	输入0~127时，输出0~127；输入128~255，输出-128~-1；输入再增加则输出重复0~127, -128~-1；输入-1~-128，输出-1~-128；输入-129~-256，输出127~0；输入再减小则输出重复-1~-128, 127~0
REAL _TO_INT	REAL	INT	输入0~32767时，输出0~32767；输入32768~65535，输出-32768~-1；输入再增加则输出重复0~32767, 32768~-1；输入-1~-32768，输出-1~-32768；输入 -32769~-65536，输出32767~0；输入再减小则输出重复-1~-32768, 32767~0
REAL _TO_DINT	REAL	DINT	输入0~2,147,483,647时，输出0~2,147,483,647，输入再增加输出仍为2,147,483,647；输入 -1~-2,147,483,648，输出-1~-2,147,483,648，输入再减小输出仍为-2,147,483,648
REAL _TO_USINT	REAL	USINT	输入0~255时，输出0~255，输入再增加则输出重复 0~255；输入-1~-256，输出255~0，输出再减小则输出重复255~0
REAL _TO_UINT	REAL	UINT	输入0~65535时，输出0~65535，输入再增加则输出重复 0~65535；输入-1~-65536，输出65535~0，输入再减小则输出重复65535~0
REAL _TO_UDINT	REAL	UDINT	输入0~2,147,483,647时，输出0~2,147,483,647，输入再增加输出为2,147,483,647；输入 -1~-2,147,483,648，输出 4,294,967,295~2,147,483,648，输入再减小输出仍为2,147,483,648；精度会降低
REAL _TO_LREAL	REAL	LREAL	OUT=IN

REAL 型数据转换指令处理的数据类型（以 REAL_TO_BYTE 为例）

输入	数据类型	描述
EN	BOOL	TRUE有效。如果不带EN/ENO端，则REAL_TO_BYTE指令一直执行

IN	SIN	输入值，有效值0~4,294,967,295
输出	数据类型	描述
ENO	BOOL	REAL_TO_BYTE指令成功执行后，为TRUE，否则为FALSE
OUT	BCD	结果，IN<=255时，OUT=IN；IN>255时，OUT=IN%256

8.16、LREAL 型数据的转换

功能：LREAL 型数据的转换有11 个指令，可以将LREAL 型数据分别转换为BOOL、BYTE WORD、DWORD、SINT、INT、DINT、USINT、UINT、UDINT和REAL等类型。

LREAL 型转换分为两步。 首先对实数进行四舍五入（小于等于 0.49 的数值被舍去，等于或者大于 0.50 的数值进一位）。 在第二步里，四舍五入的输入值被转换到输出值相应数据类型。

指令	输入值	输出值	描述
LREAL_TO_BOOL	LREAL	BOOL	IN=0时，OUT=FALSE，其它情况下输出均为TRUE
LREAL_TO_BYTE	LREAL	BYTE	IN<=255时，OUT=IN；IN>255时，OUT=IN%256
LREAL _TO_WORD	LREAL	WORD	IN<=65535时，OUT=IN；IN>65535时，OUT=IN%65536；输入-1~-65536，输出65535~0，输入再减小则输出重复65535~0
LREAL _TO_DWORD	LREAL	DWORD	输入0~2,147,483,647时，输出0~2,147,483,647，输入再增加输出为2,147,483,647；输入 -1~-2,147,483,648，输出 4,294,967,295~2,147,483,648，输入再减小输出仍为2,147,483,648；精度会降低
LIREAL _TO_SINT	LREAL	SINT	输入0~127时，输出0~127；输入128~255，输出-128~-1；输入再增加则输出重复0~127，-128~-1；输入-1~-128，输出-1~-128；输入-129~-256，输出127~0；输入再减小则输出重复-1~-128，127~0
LREAL _TO_INT	LREAL	INT	输入0~32767时，输出0~32767；输入32768~65535，输出-32768~-1；输入再增加则输出重复0~32767， -32768~-1；输入-1~-32768，输出-1~-32768；输入-32769~-65536，输出32767~0；输入再减小则输出重复-1~-32768，32767~0
LREAL _TO_DINT	LREAL	DINT	输入0~2,147,483,647时，输出0~2,147,483,647，输入再增加输出仍为2,147,483,647；输入 -1~-2,147,483,648，输出-1~-2,147,483,648，输入再减小输出仍为-2,147,483,648
LREAL _TO_USINT	LREAL	USINT	输入0~255时，输出0~255，输入再增加则输出重复0~255；输入-1~-256，输出255~0，输出再减小则输出重复255~0
LREAL _TO_UINT	LREAL	UINT	输入0~65535时，输出0~65535，输入再增加则输出重复

			0~65535；输入-1~-65536，输出65535~0，输入再减小则输出重复65535~0
LREAL _TO_UDINT	LREAL	UDINT	输入0~2,147,483,647时，输出0~2,147,483,647，输入再增加输出为2,147,483,647；输入 -1~-2,147,483,648，输出 4,294,967,295~2,147,483,648，输入再减小输出仍为2,147,483,648；精度会降低
LREAL _TO_LREAL	LREAL	REAL	输入等于输出，精度会降低

LREAL 型数据转换指令处理的数据类型（以 LREAL_TO_BYTE 为例）

输入	数据类型	描述
EN	BOOL	TRUE有效。如果不带EN/ENO端，则LREAL_TO_BYTE指令一直执行
IN	SIN	输入值，有效值0~4,294,967,295
输出	数据类型	描述
ENO	BOOL	LREAL_TO_BYTE指令成功执行后，为TRUE，否则为FALSE
OUT	BCD	结果，IN<=255时，OUT=IN；IN>255时，OUT=IN%256

8.17、TRUNC 型数据的转换

功能：TRUNC 小数取整有 4 个指令，这个类型转换功能截去位于小数点后面的小数部分，以获得一个整数。可以将浮点型数据分别转换为 SINT、INT、DINT 等类型。

指令	输入值	输出值	描述
TRUNC	ANY_REAL	ANY_INT	小数的整数部分。输入负数时输出类型应为有符号整数
TRUNC_SINT	REAL	SINT	输入0~127时，输出0~127；输入128~255，输出-128~-1；输入再增加则输出重复0~127, -128~-1；输入-1~-128，输出-1~-128；输入-129~-256，输出127~0；输入再减小则输出重复-1~-128, 127~0
TRUNC _INT	REAL	INT	输入0~32767时，输出0~32767；输入32768~65535，输出-32768~-1；输入再增加则输出重复0~32767, -32768~-1；输入-1~-32768，输出-1~-32768；输入 -32769~-65536，输出32767~0；输入再减小则输出重复 -1~-32768, 32767~0
TRUNC _DINT	REAL	DINT	输入0~2,147,483,647时，输出0~2,147,483,647，输入再增加输出仍为2,147,483,647；输入 -1~-2,147,483,648，输出-1~-2,147,483,648，输入再减小输出仍为-2,147,483,648
TRUNC _LINT	REAL	LINT	输入0~2,147,483,647时，输出0~2,147,483,647，输入再增加输出仍为2,147,483,647；输入

			-1~-2,147,483,648，输出-1~-2,147,483,648，输入再减小输出仍为-2,147,483,648

TRUNC 型数据转换指令处理的数据类型（以 TRUNC 为例）

输入	数据类型	描述
EN	BOOL	TRUE有效。如果不带EN/ENO端，则TRUNC指令一直执行
IN	ANY_REAL	输入值
输出	数据类型	描述
ENO	BOOL	TRUNC指令成功执行后，为TRUE，否则为FALSE
OUT	ANY_INT	结果，输入实数的整数部分

8.18、TIME 型数据的转换

功能：TIME_TO_DINT 类型转换功能将一个 TIME 型的输入值转换为一个 DINT 型的输出值（任何时间值都将换算为毫秒值再将毫秒值转换为 DINT）。

TIME 型数据必须是以 T#为开始的一个无符号数，大于 T#2147483647 的时间值都将为负数，因为 DINT 类型为有符号数，其最大值为 2,147,483,647，例如，输入值 T#4294967295 毫秒将导致输出值为-1。

TIME_TO_DINT 指令处理的数据类型

输入	数据类型	描述
EN	BOOL	TRUE有效。如果不带EN/ENO端，则TIME_TO_DINT指令一直执行
IN	TIME	输入值，取值范围T#0~2147483647MS（等于T#0~2147483.647S）
输出	数据类型	描述
ENO	BOOL	TIME_TO_DINT指令成功执行后，为TRUE，否则为FALSE
OUT	DINT	结果，转换为DINT输出值为0~2147483647 T#2147483648<=IN<=T#4294967295时，OUT=IN-4294967296，为负值很少用

二、“功能块”库

1、双稳态功能块

1.1、SR 置位优先功能块

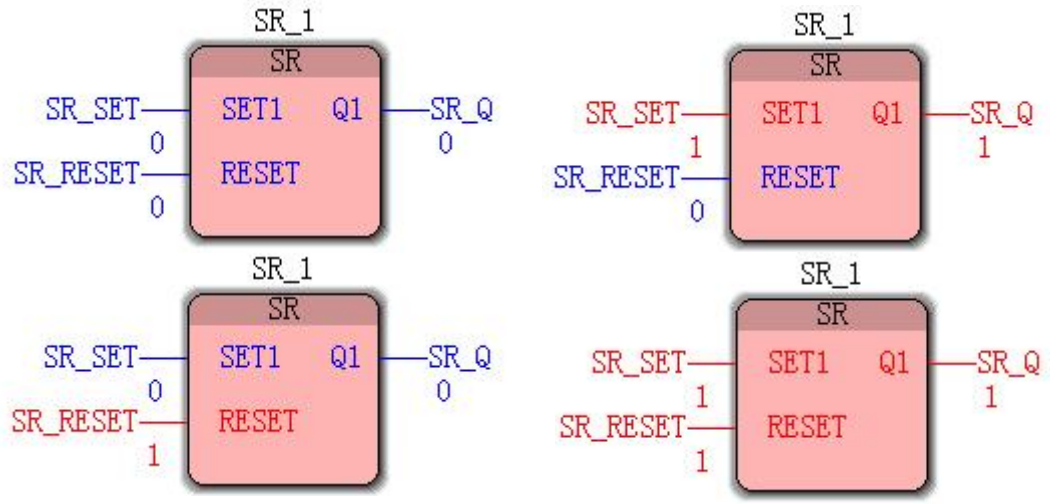
功能：这个双稳态功能块实现 Q1 输出端的优先置位。 如果输入 SET1=TRUE，则 Q1 输出端被置位。 即使 SET 为 FALSE，Q1 仍然保持置位状态。 如果 RESET=TRUE，则 Q1 输出端被复位。 如果两个输入均为 TRUE，则 Q1 输出端被 SET1 设置为 TRUE。如果是第一次调用此功能块，则 Q1 为 FALSE。

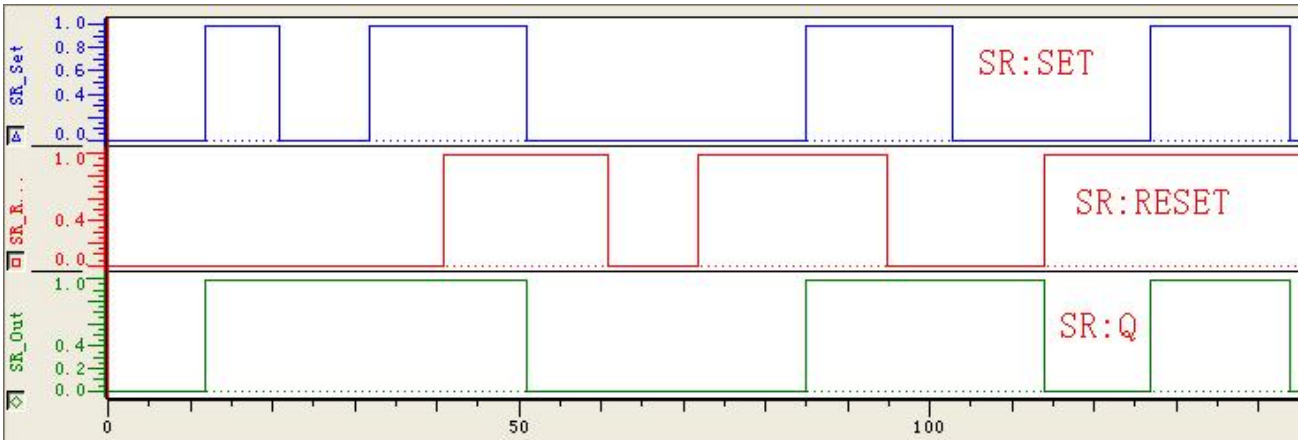
参数	**数据类型**	**描述**
SET1	BOOL	如果为 TRUE，则 Q1 为置位优先。
RESET	BOOL	如果为 TRUE，Q1 被复位。
Q1	BOOL	结果 SET=0，RESET1=0，Q1 保持上次状态； SET=1，RESET1=0，Q1=1； SET=0，RESET1=1，Q1=0； SET=1，RESET1=1，Q1=1；

注：功能块必须被实例化： 必须在需要使用该 FB 的 POU 变量工作单中声明所选择的实例名称（声明使用 VAR 关键字）。 该实例名称必须在 POU 内是唯一的。 在下面的例子中，使用了实例名称“SR_1”。

所有参数都可以被求反。

应用举例：这个双稳态功能块实现 Q1 输出端的优先置位。如果输入 SR_SET=TRUE，则 Q1 输出端被置位。即使 SR_SET 变为 FALSE(假)，Q1 仍然保持置位状态。如果 SR_RESET=TRUE(真)，则 Q1 输出端被复位。如果两个输入都=TRUE，则 Q1 输出端被 SR_SET 设置为 TRUE(真)。 如果是第一次调用此功能块，则 Q1 为 FALSE(假)。





1.2、RS 复位优先功能块

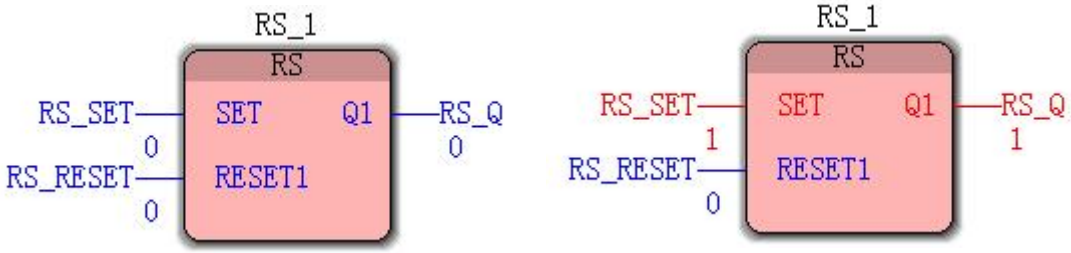
功能：这个双稳态功能块实现了 Q1 输出端的优先复位。 如果输入端 SET=TRUE，则输出端 Q1 被置位。即使 SET 为 FALSE，Q1 仍然保持置位状态。 如果 RESET1=TRUE，则 Q1 被复位。 如果两个输入均为 TRUE，则由 RESET1 将 Q1 输出端设置为 FALSE。。如果是第一次调用此功能块，则 Q1 为 FALSE。

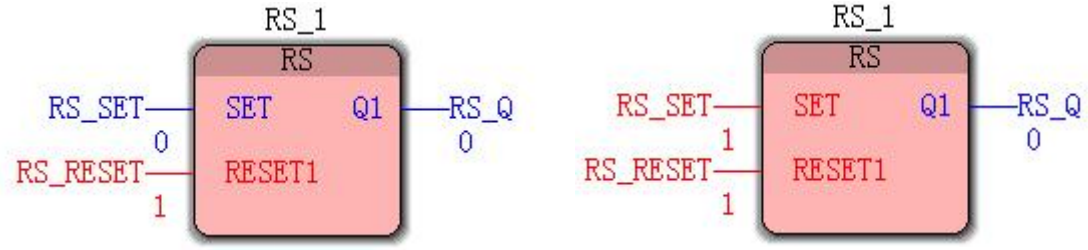
参数	**数据类型**	**描述**
SET1	BOOL	如果为 TRUE，则 Q1 为置位优先。
RESET	BOOL	如果为 TRUE，Q1 被复位。
Q1	BOOL	结果 SET=0，RESET1=0，Q1 保持上次状态； SET=1，RESET1=0，Q1=1； SET=0，RESET1=1，Q1=0； SET=1，RESET1=1，Q1=0；

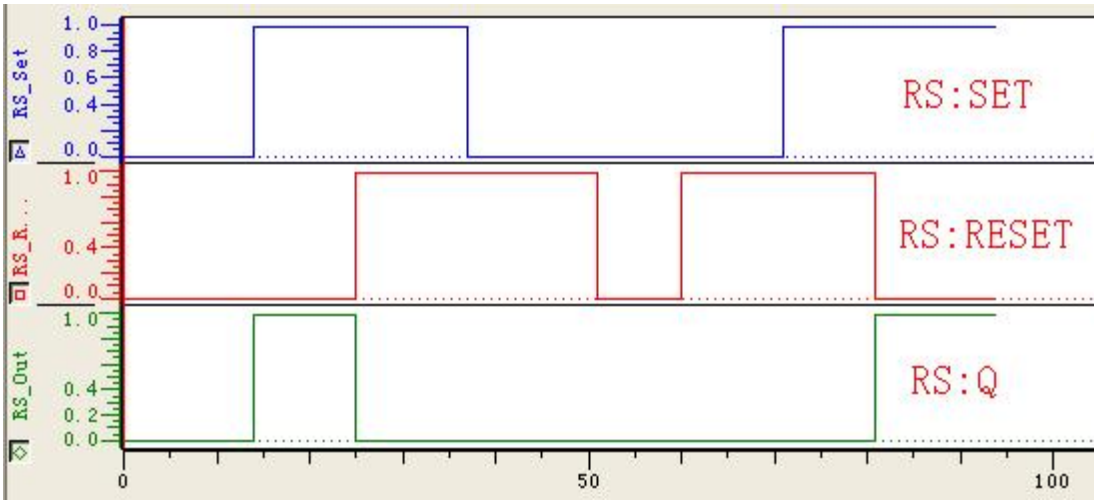
注：功能块必须被实例化： 必须在需要使用该 FB 的 POU 变量工作单中声明所选择的实例名称（声明使用 VAR 关键字）。 该实例名称必须在 POU 内是唯一的。 在以下例子中使用了实例名称“RS_1”。

所有参数都可以被求反。

应用举例：这个双稳态功能块实现了 Q1 输出端的优先复位。如果输入端 RS_SET=TRUE,,则输出端 Q1 被置位。即使 RS_SET 变为 FALSE(假)，Q1 仍然保持置位状态。如果 RS_RESET=TRUE(真)，则 Q1 被复位。如果两个输入都=TRUE(真)，则由 RS_RESET 将 Q1 输出端设置为 FALSE(假)。 如果是第一次调用此功能块，则 Q1 为 FALSE(假)。







2、边沿检测功能块

2.1、F_TRIG 下降沿检测器

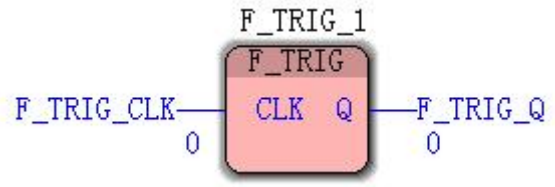
功能：这个边沿检测功能块检测下降沿。 如果在 CLK 输入端检测到一个下降沿，则 Q 输出端将由 FALSE 变为 TRUE。 直到下一次执行此功能块之前，Q 输出端都将保持为 TRUE。

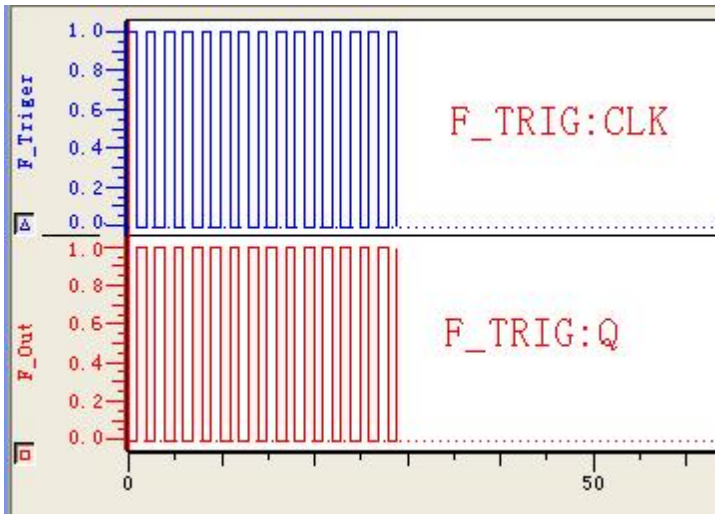
如果是第一次调用此功能块，则直到检测到第一个边沿之前，Q 输出端都将保持为 FALSE。

参数	**数据类型**	**描述**
CLK	BOOL	检测到下降沿。
Q	BOOL	当 CLK 出现下降沿时，Q=由 0 变为 1，直到下一次扫描到这条指令

注：功能块必须被实例化： 必须在需要使用该 FB 的 POU 变量工作单中声明所选择的实例名称（声明使用 VAR 关键字）。 该实例名称必须在 POU 内是唯一的。 在下面例子中使用了实例名称“F_TRIG_1”。

应用举例：这个边沿检测功能块检测下降沿。如果在 CLK 输入端检测到一个下降沿，则 Q 输出端将由 FALSE(假)变为 TRUE(真)。输出为一个周期的脉冲信号。如果是第一次调用此功能块，则直到检测到第一个边沿之前，Q 输出端都将一直为 FALSE(假)。





2.2、R_TRIG 上升沿检测器

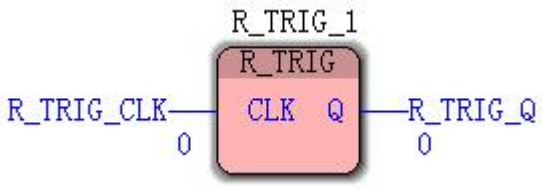
功能：这个边沿检测功能块检测上升沿。 如果在 CLK 输入端检测到一个上升沿，则 Q 输出端将由 FALSE 变为 TRUE。 直到下一次执行此功能块之前，Q 均保持为 TRUE。

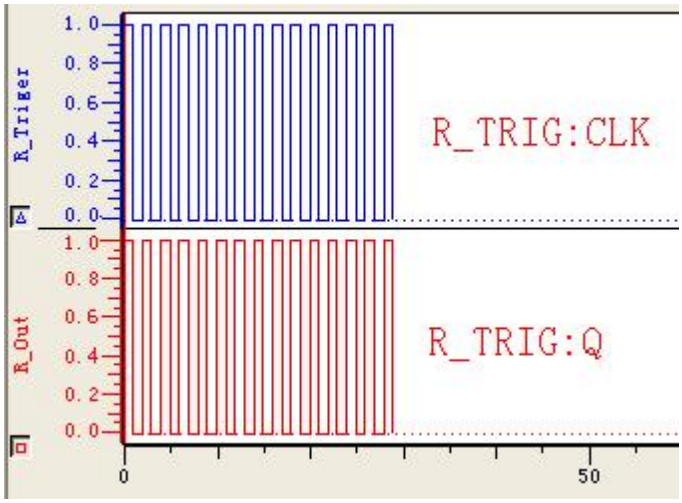
如果是第一次调用此功能块，则直到检测到第一个边沿之前，Q 输出端都将保持为 FALSE。

参数	**数据类型**	**描述**
CLK	BOOL	检测到下降沿。
Q	BOOL	当 CLK 出现上升沿时，Q=由 0 变为 1，直到下一次扫描到这条指令

注：功能块必须被实例化： 必须在需要使用该 FB 的 POU 变量工作单中声明所选择的实例名称（声明使用 VAR 关键字）。 该实例名称必须在 POU 内是唯一的。 在下面例子中使用了实例名称“R_TRIG_1”。

应用举例：这个边沿检测功能块检测上升沿。如果在 CLK 输入端检测到一个上升沿，则 Q 输出端由 FALSE(假)变为 TRUE(真)，输出为一个周期的脉冲信号。如果是第一次调用此功能块，则直到检测到第一个边沿之前，Q 输出端都将一直为 FALSE(假)。





3、计数器功能块

3.1、CTU 递增计数器

功能：这个记数器功能块递增计数。 假设在 CU 输入端有一个上升沿，并且 RESET=FALSE，则 CV 递增一。 如果达到了计数器的最终值（PV），则在 Q 输出端发出一个 TRUE 信号，并且功能块停止计数。

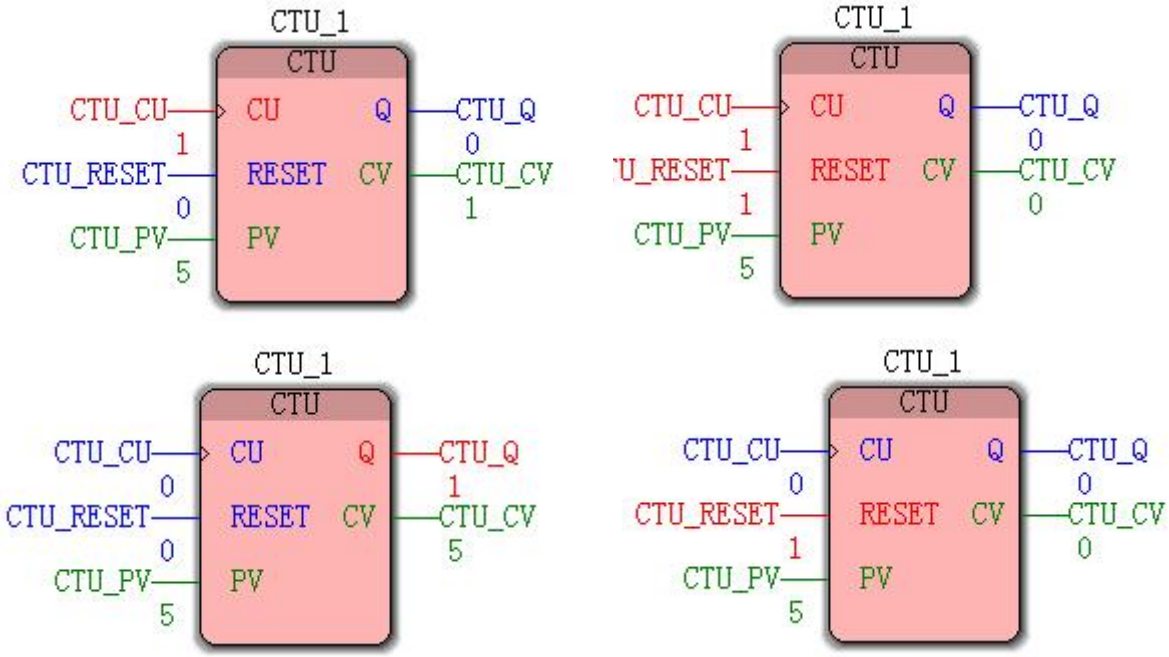
如果 RESET=TRUE，则用 0 初始化记数器。为了使能记数过程，RESET 输入端必须为 FALSE。 否则记数器将总被重新初始化。

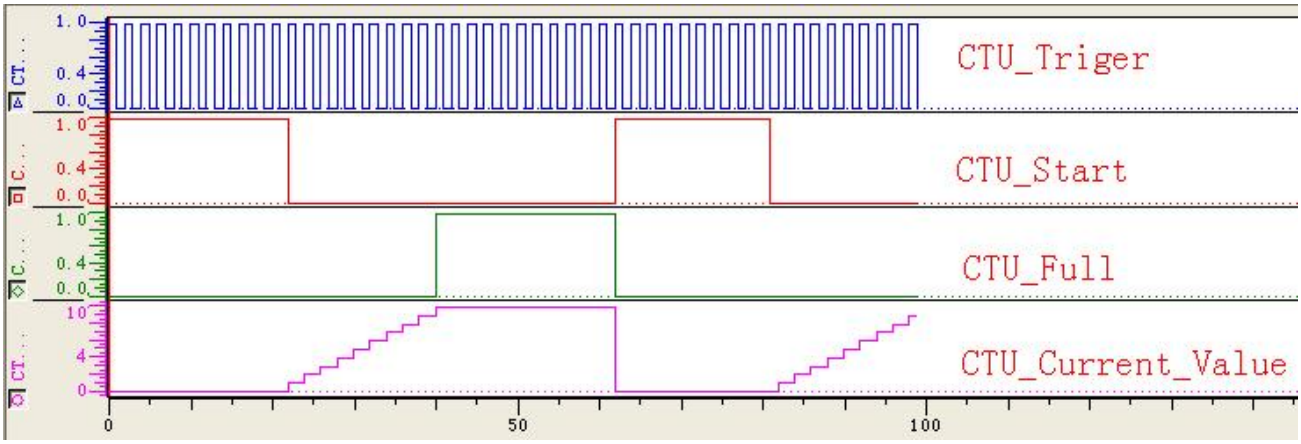
参数	数据类型	描述
CU	BOOL	如果检测到一个上升沿，则 CV 递增一。
RESET	BOOL	如果为 TRUE，则用 0 来初始化计数器。 如果为 FALSE，启用计数。
PV	INT	预置
Q	BOOL	如果 CV=PV，则为 TRUE。
CV	INT	计数器结果。

注：功能块必须被实例化： 必须在需要使用该 FB 的 POU 变量工作单中声明所选择的实例名称（声明使用 VAR 关键字）。 该实例名称必须在 POU 内是唯一的。 在下面的例子中使用了实例名称“CTU_1”。

Q 输出端可被取反。

应用举例：这个计数器功能块递增计数。假设在 CU 输入端有一个上升沿，并且 RESET=FALSE，则 CV 递增 1。如果达到了计数器的最后值（PV），则在 Q 输出端发出一个 TRUE 信号，并且功能块停止记数如果 RESET=TRUE，则用 0 初始化记数器。为了使能计数过程，RESET 输入端必须为 FALSE。否则计数器将总被重新初始化。





3.2、CTD 递减计数器

功能：这个计数器功能块递减记数。 假设在 CD 输入端有一个上升沿，且 LOAD=FALSE，则 CV 减一。 如果达到了计数器的最终值（PV），则在 Q 输出端发出一个 TRUE 信号，并且功能块停止计数。

如果 LOAD=TRUE，则按 PV 输入的值初始化计数器。 为了启用计数过程，LOAD 输入端必须为 FALSE。 否则将重新初始化计数器。

参数	数据类型	描述
CD	BOOL	如果检测到一个上升沿，则 CV 减一。
LOAD	BOOL	如果为 TRUE，计数器用 PV 来初始化。 如果为 FALSE，启用计数。
PV	INT	预置值。
Q	BOOL	如果 CV=0，则为 TRUE

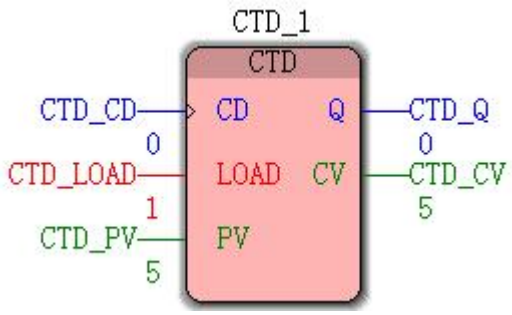
CV	INT	计数器结果。

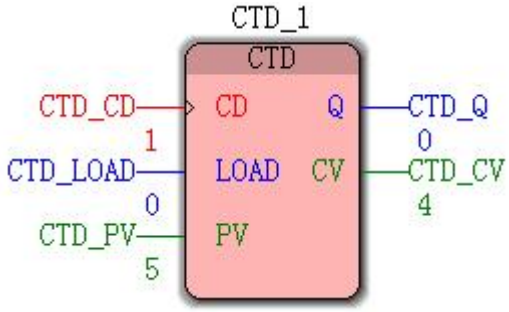
注：功能块必须被实例化： 必须在需要使用该 FB 的 POU 变量工作单中声明所选择的实例名称（声明使用 VAR 关键字）。 该实例名称必须在 POU 内是唯一的。 在下面的例子中使用了实例名称“CTD_1”。

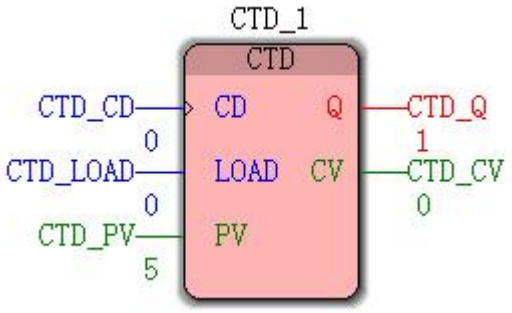
Q 输出端可被取反。

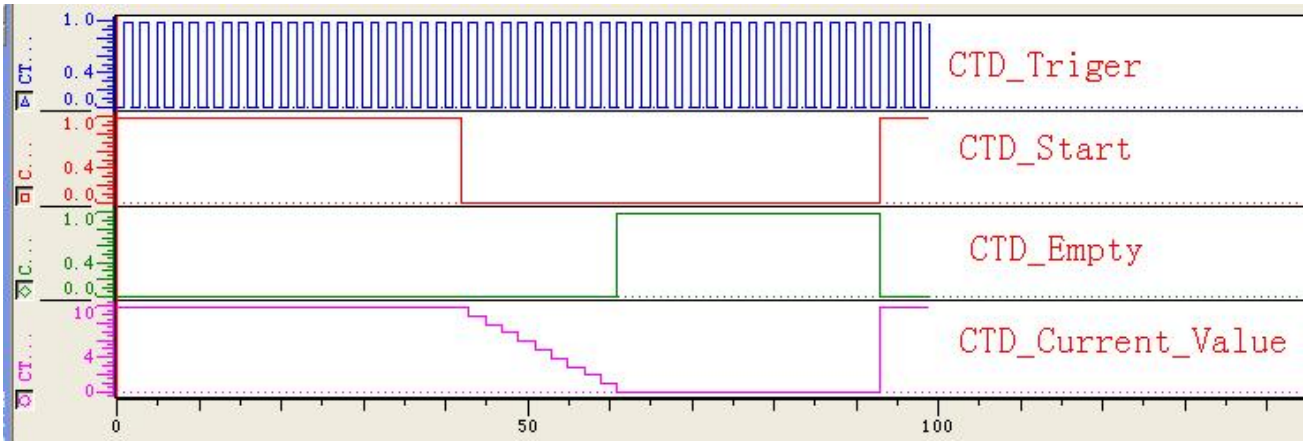
应用举例：这个计数器功能块递减计数。假设在 CD 输入端有一个上升沿，并且 LOAD=FALSE，则 CV 减 1。如果达到了计数器的最后值（0），则在 Q 输出端发出 TRUE 信号，并且功能块停止计数。

如果 LOAD=TRUE，则通过 PV 输入的值初始化记数器。为了使能记数过程，LOAD 输入端必须为 FALSE。否则计数器将总被重新初始化。









3.3、CTUD 增减计数器

功能：这个计数器功能块递增或者递减记数。 假设在 CU 输入端有一个上升沿，则 CV 递增一。 假设在 CD 输入端有一个上升沿，则 CV 递减一。 如果 CV = PV，OU 设置为 TRUE。 如果 CV = PV，OU 设置为 TRUE。

如果 RESET = TRUE，计数器初始化为 0。如果 LOAD = TRUE，计数器初始化为 PV。 为了启用计数过程，RESET 和 LOAD 输入端都必须为 FALSE。 否则将重新初始化计数器。

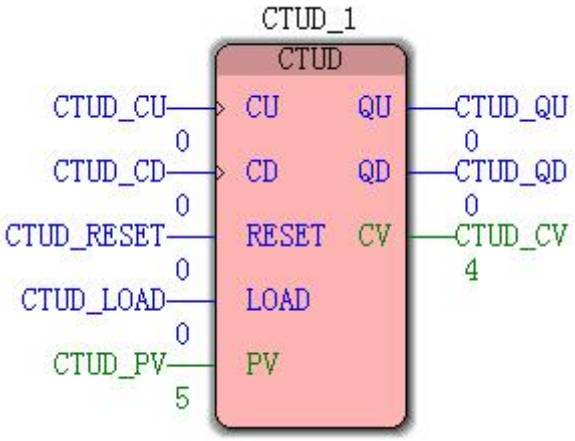
参数	**数据类型**	**描述**
CU	BOOL	如果检测到一个上升沿，则 CV 递增一。
CD	BOOL	如果检测到一个上升沿，则 CV 递减一。
RESET	BOOL	如果为 TRUE，则计数器初始化为零。 如果为 FALSE，启用计数。
LOAD	BOOL	如果为 TRUE，计数器用 PV 来初始化。 如果为 FALSE，启用计数。
PV	INT	预置值。
QU	BOOL	如果 CV=PV，则为 TRUE。
QD	BOOL	如果 CV=0，则为 TRUE。
CV	INT	计数器结果。

注：功能块必须被实例化： 必须在需要使用该 FB 的 POU 变量工作单中声明所选择的实例名称（声明使用 VAR 关键字）。 该实例名称必须在 POU 内是唯一的。 在下面的例子中使用了实例名称“CTUD_1”。

QU 和 QD 输出端都可以被取反。

应用举例：这个计数器功能块递增或者递减计数。假设在 CU 输入端有一个上升沿，则 CV 递增 1。假设在 CD 输入端有一个上升沿，则 CV 递减 1。如果 CV=PV，则 QU 输出端发出一个 TRUE 信号。如果 CV=0，则 QD 输出端发出一个 TRUE 信号。

如果输入端 RESET=TRUE，则用 0 初始化计数器。如果输入端 LOAD=TRUE，则用 PV 初始化计数器。为了使能计数过程，RESET 和 LOAD 输入端都必须为 FALSE。否则计数器将总被重新初始化。



4、定时器功能块

4.1、TON 延时接通定时器

功能：这个定时器功能块实现延时开定时。

如果输入 IN 从 TRUE 变为 FALSE，在延迟输入 PT 中的时长后开机。经过 PT 值的时长后，Q 值设置为 TRUE。在输出端 ET 上显示过程时间间隔。

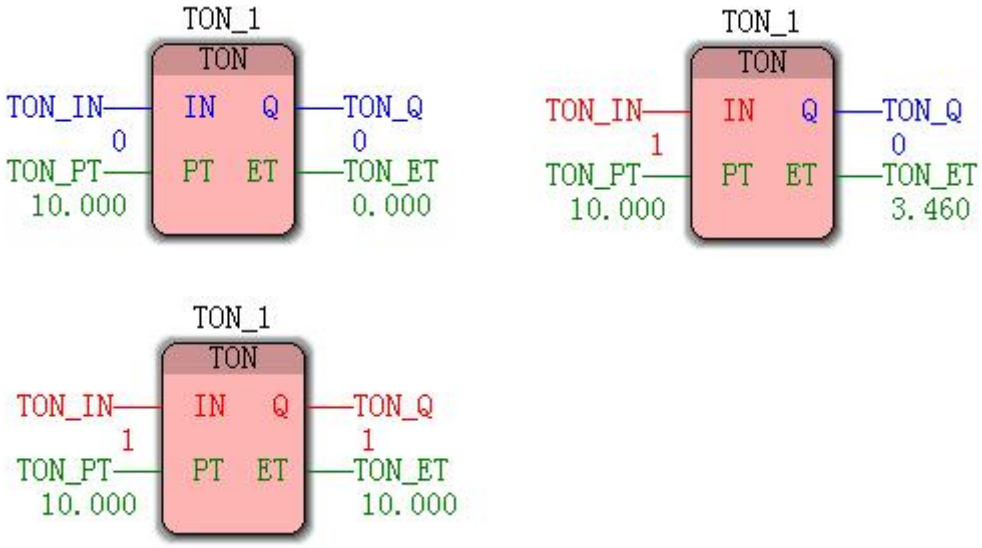
参数	**数据类型**	**描述**
IN	BOOL	如果检测到一个上升沿，则启动开-延迟定时功能。
PT	TIME	用于预设置延迟时间。
Q	BOOL	TRUE if IN = TRUE and ET > PT. TRUE if IN = TRUE and ET < PT.
ET	TIME	过程时间间隔。

注：功能块必须被实例化： 必须在需要使用该 FB 的 POU 变量工作单中声明所选择的实例名称（声明使用 VAR 关键字）。该实例名称必须在 POU 内是唯一的。在以下例子中使用了实例名称“WATCHDOG_TIMER”。

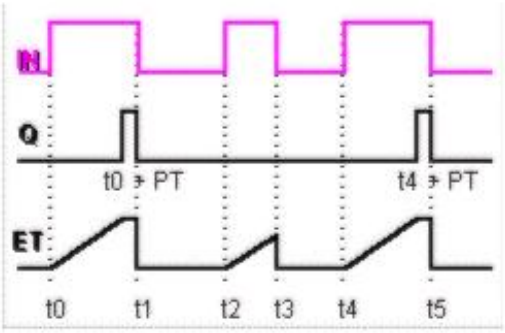
IN 输入端和 Q 输出端都可以被取反。

应用举例：这个定时器功能块实现开-延时定时。

如果 IN 输入端由 FALSE(假)变为 TRUE(真)，则接通将延迟一个在 PT 输入端处的时间间隔。在经过 PT 这段时间之后，Q 输出端变为 TRUE(真)。已经过去的时间显示在 ET 输出端。



时序图



4.2、TOF 延时断开定时器

功能：这个定时器功能块实现延时关定时功能。

如果输入 IN 从 TRUE 变为 FALSE，在延迟输入 PT 中的时长后关机。经过 PT 值的时长后，Q 值设置为 FALSE。在输出端 ET 上显示过程时间间隔。

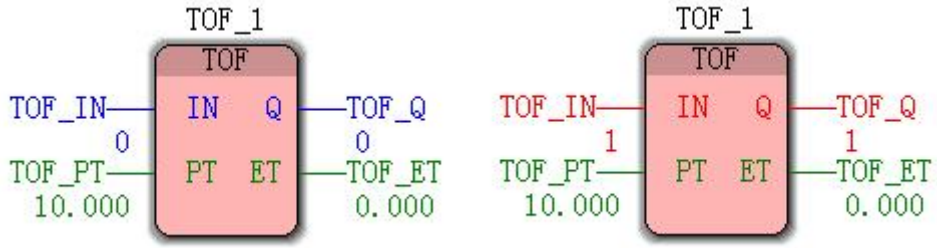
参数	**数据类型**	**描述**
IN	BOOL	如果检测到一个下降沿，则启动延迟关定时器。
PT	TIME	用于预设置延迟时间。
Q	BOOL	TRUE if IN = TRUE and ET < PT. TRUE if IN = TRUE and ET > PT.
ET	TIME	过程时间间隔。

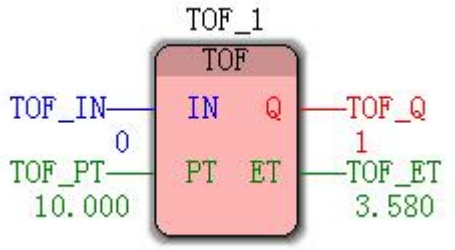
注：功能块必须被实例化： 必须在需要使用该 FB 的 POU 变量工作单中声明所选择的实例名称（声明使用 VAR 关键字）。 该实例名称必须在 POU 内是唯一的。 以下例子中，使用了实例名称“DELAY_TIMER”。

IN 输入端和 Q 输出端都可以被取反。

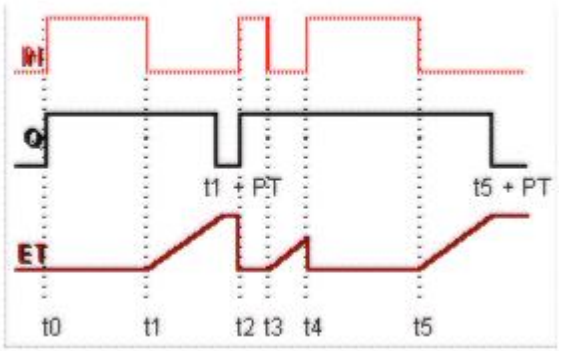
应用举例：这个定时器功能块实现关-延时定时功能。

如果 IN 输入端由 TRUE(真)变到 FALSE(假)，关断将延迟一个在 PT 输入端处的时间间隔。在经过 PT 这段时间之后，Q 输出端变为 FALSE(假)。已经过去的时间显示在 ET 输出端。





时序图



4.3、TP 脉冲定时器

功能：这个定时器功能块产生一个脉冲。

如果 IN 输入端由 FALSE 变为 TRUE，则在 Q 输出端产生一个时间间隔为 PT 的脉冲。过程时间显示于 ET 输出端。 如果 IN 获得另一个时间 TRUE 值，而 PT 时间段还没有结束，则它对在 Q 输出端产生的脉冲时间段没有影响。

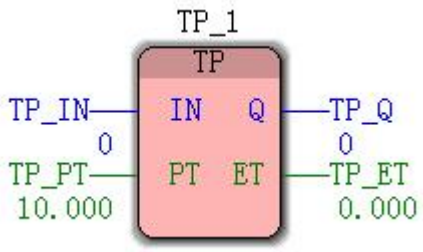
参数	**数据类型**	**描述**
IN	BOOL	如果检测到一个上升沿，则产生一个脉冲。
PT	TIME	预置的脉冲时间间隔。
Q	BOOL	TRUE if IN = TRUE and ET < PT. TRUE if IN = TRUE and ET > PT.
ET	TIME	过程时间间隔。

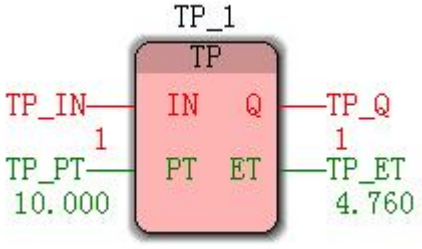
注：功能块必须被实例化： 必须在需要使用该 FB 的 POU 变量工作单中声明所选择的实例名称（声明使用 VAR 关键字）。 该实例名称必须在 POU 内是唯一的。 以下例子中，使用了实例名称“PULSE_TIMER”。

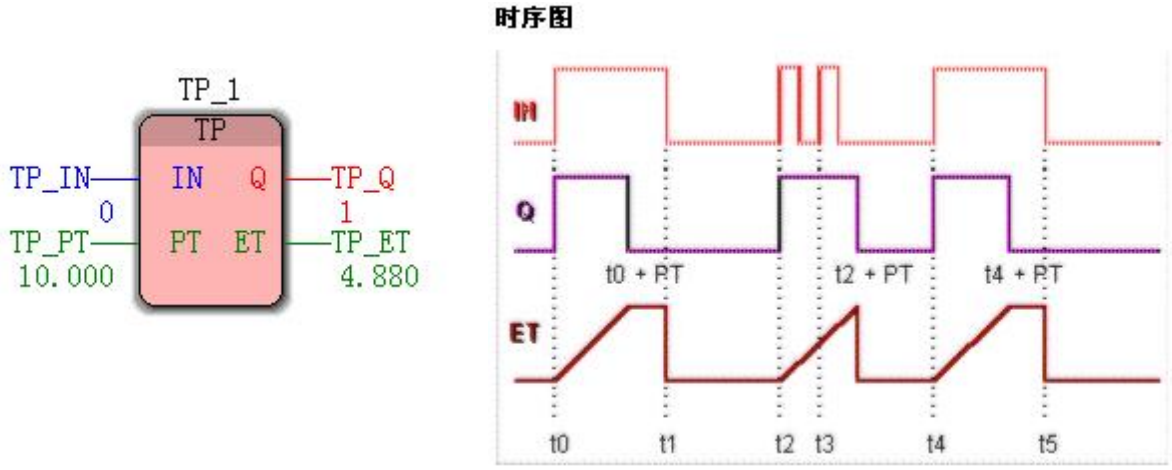
IN 输入端和 Q 输出端都可以被取反。

应用举例：这个定时器功能块产生一个脉冲。

如果 IN 输入端由 FALSE(假)变为 TRUE(真)，则在 Q 输出端产生一个时间间隔为 PT 的脉冲。已经流逝的时间显示于 ET 输出端。如果 IN 输入端又得到了用于另一个时间段的 TRUE 值，而 PT 时间段还没有流逝完，则它对在 Q 输出端产生的脉冲的时间间隔没有影响。







三、位操作函数

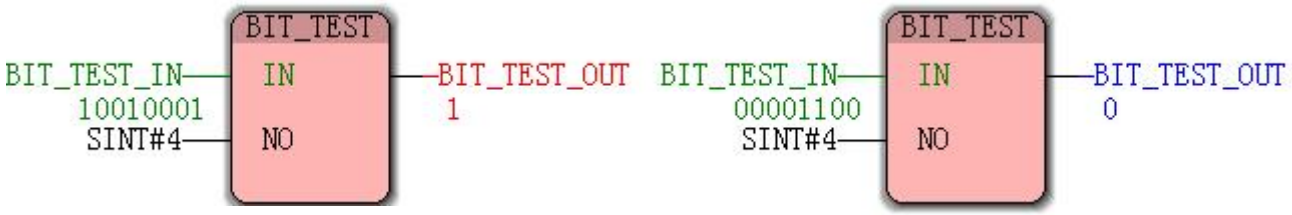
位操作函数在“BIT_UTIL”固件库里面，需要导入此固件库才能使用。

1、BIT_TEST 读取位串中的一个位的值

功能：这个位操作函数能用于读取位串中的单个位的值。

参数	**数据类型**	**描述**
IN	ANY_BIT	输入位串
NO	SINT	将被测试的位数
OUT	BOOL	被测试位的输出值

应用举例：在输入端 BIT_TEST_IN 中取第 4 位的值作为输出，从 0 位开始计算。

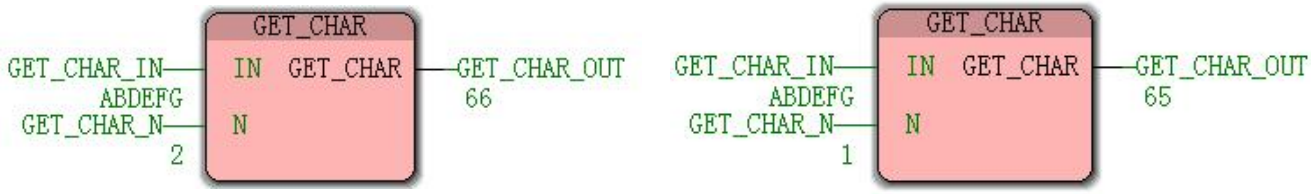


2、GET_CHAR 字符串里摘录一个字符

功能：这个位操作函数可用于从字符串中摘录一个字符。输出值表示该字符的 ASCII 码值。

参数	数据类型	描述
IN	STRING	输入字符串
N	INT	字符串内，从左边开始的字符序号
GET_CHAR	INT	所摘录字符的 ASCII 码值

应用举例：在输入端 GET_CHAR_IN 的字符串 ABEDFG 中字符串中摘录第 2 个字符，输出值表示该字符的 ASCII 码值。如 B 的 ASCII 码值为 66，A 的 ASCII 码值为 65，具体 ASCII 码值对照表详见附表一。

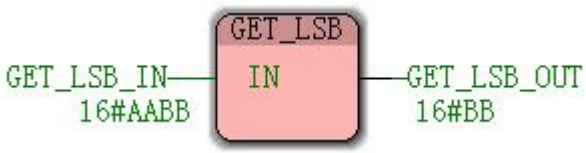


3、GET_LSB 读取位串的较低字节的值

功能：这个位操作函数可用于读取位串的较低字节(the less significant BYTE)的值。

参数	数据类型	描述
IN	WORD	输入位串
OUT	BYTE	较低字节的值

应用举例：读取 GET_LSB_IN 中低字节的值作为输出，如下图，读取 16#AABB 中的低字节，输出为 16#BB。

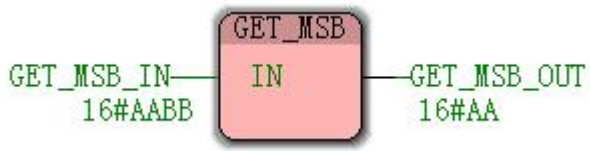


4、GET_MSB 读取位串的最高字节的值

功能：这个位操作函数可用于读取位串的最高字节的值。

参数	数据类型	描述
IN	WORD	输入位串
OUT	BYTE	最高字节的值

应用举例：读取 GET_MSB_IN 中高字节的值作为输出，如下图，读取 16#AABB 中的低字节，输出为 16#AA。



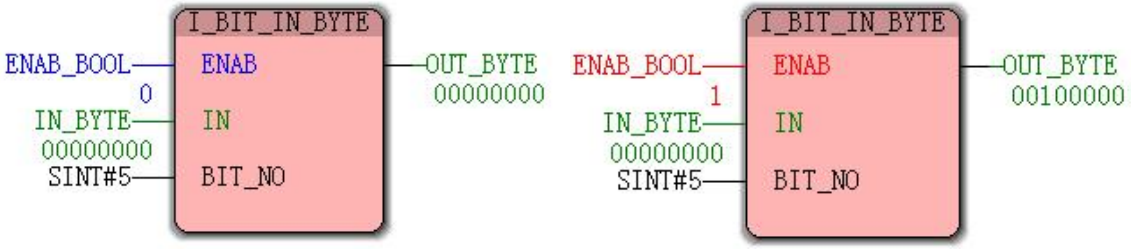
5、I_BIT_IN_*反转位串中的一个位

功能：这个位操作函数可用于反转位串中的一个位。

参数	数据类型	描述
ENAB	BOOL	使能函数的执行
IN	BYTE, WORD, DWORD	输入位串
BIT_NO	SINT	被反转的位的序号
OUT	BYTE, WORD, DWORD	输出位串

应用举例： 若 ENAB_BOOL=0，模块使能关闭，OUT_BYTE=IN_BYTE,

若 ENAB_BOOL=1，在 IN_BYTE 的第 5 位取反后输出，从第 0 位开始计算。



6、PARITY_*检查已置 1 的位数是奇数还是偶数

功能：这个位操作函数用于检查被置 1 的数量是奇数还是偶数。

参数	数据类型	描述
IN1	BYTE, WORD, DWORD	输入位串
OUT	BOOL	被置位的位数 TRUE = 偶数 FALSE = 奇数

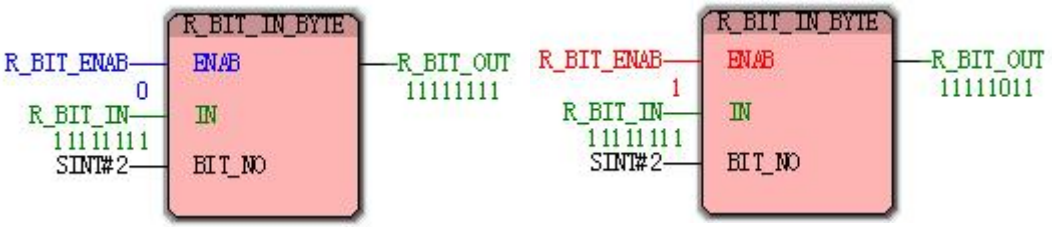
应用举例： 检查输入中二进制位置一的个数，若为偶数，输出 1，为奇数，输出 0。

7、R_BIT_IN_*复位位串中的一个位

功能：这个位操作函数可用于将位串中的一个位复位为 FALSE。

参数	数据类型	描述
ENAB	BOOL	使能函数的执行
IN	BYTE, WORD, DWORD	输入位串
BIT_NO	SINT	将被复位为 FALSE 的位的数目
OUT	BYTE, WORD, DWORD	输出位串

应用举例：当使能端为 1 时，复位输入中二进制位中所确定的位，若使能端为 0，则不动作。

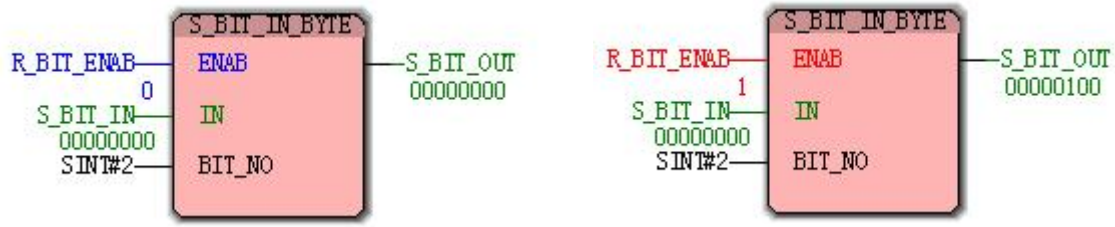


8、S_BIT_IN_*置位位串中的一个位

功能：这个位操作函数可用于将位串中的某一位设置为 TRUE。

参数	数据类型	描述
ENAB	BOOL	使能函数的执行
IN	BYTE, WORD, DWORD	输入位串
BIT_NO	SINT	将被设置为 TRUE 的位的序号
OUT	BYTE, WORD, DWORD	输出位串

应用举例：当使能端为 1 时，置位输入中二进制位中所确定的位，若使能端为 0，则不动作。

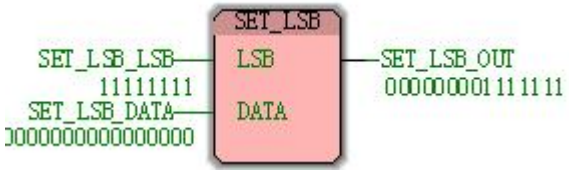


9、SET_LSB 向位串的较低字节写值

功能：这个位操作函数可用于向位串的较低字节(the less significant BYTE)写值。

参数	数据类型	描述
LSB	BYTE	将被写入较低字节上的值
DATA	WORD	输入位串
OUT	WORD	输出位串

应用举例：向位串的低字节中写入值。

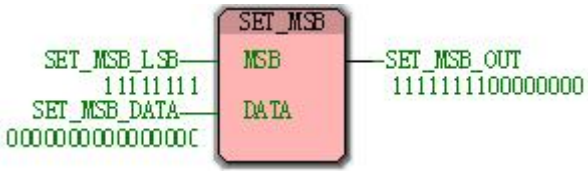


10、SET_MSB 向位串的最高字节写值

功能：这个位操作函数可用于向位串的最高字节(the most significant BYTE)上写值。

参数	数据类型	描述
MSB	BYTE	将被写到最高字节上的值
DATA	WORD	输入位串
OUT	WORD	输出位串

应用举例：向位串的高字节中写入值。



11、STRING_TO_BUFFER 将字符串的字符复制到缓冲区

功能：这个位操作函数将字符串的字符复制到缓冲区。

注意：要确保被复制到缓冲区的字符串不超过数组的长度。编程系统不做这类检查。

注意：只有在需要数组时，才应该使用这个函数。否则，应该使用 GET_CHAR 函数。

参数	数据类型	描述
STR_IN	STRING	输入字符串
BUFFER	BYTE	缓冲区
BUF_LEN	INT	复制到缓冲区的字符数

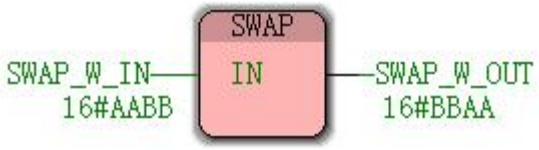
STRING_TO_BUFFER	INT	输出值

12、SWAP 交换位串的最高字节和较低字节

功能：这个位操作函数可用于交换位串的最高字节(the most significant BYTE)和较低字节(the less significant BYTE)。

参数	**数据类型**	**描述**
IN	WORD	输入位串
OUT	WORD	输出位串

应用举例：将输入位串的高字节和低字节互换，作为输出。



四、运行期系统的特定功能

运行期系统的特定功能在“PROCONS”固件库里面，需要导入此固件库才能使用。

1、COLD_RESTART 执行 PLC 冷重启

功能： 如果输入为 TRUE，这个运行期系统的特定功能将执行一个冷重启。冷重启过程中，所有数据初始化。在发生栈溢出、字符串错误或者被 0 除等问题时，该功能在相应的 SPG 中被调用，以便自动重启执行程序。发生异常后，会立即执行一个冷重启。如果是在普通 PLC 应用程序中调用，该功能首先完成当前周期的程序执行，然后执行冷重启。

参数	**数据类型**	**描述**
COND	BOOL	如果为 TRUE，则执行冷重启。
OUT	BOOL	如果 COND=TRUE 则为 TRUE，并可以进行冷重启。

注：使用该功能需谨慎。需确保发生被 0 除或字符串错误等异常情况时，重启执行程序不会引起任何破坏。

这个 ProConOS 功能在输入为 TRUE 的情况下，执行一个冷再启动(cold restart)。在冷再启动过程中，初始化所有数据。

应该在相应的 SPG 中调用这个功能，以便在发生堆栈溢出、字符串错误或者被 0 除等问题时，自动重新启动程序的执行。在发生异常之后，会立即执行一个冷再启动。

如果这个功能是在普通的 PLC 应用程序中调用的，以便执行冷再启动，则会首先完成当前周期的程序执行，然后执行冷再启动。

注意：要谨慎地使用这个功能。确保被 0 除或字符串错误等异常情况，不会通过重新启动程序的执行，而引起任何毁坏。

2、CONTINUE 继续执行程序

功能：如果输入为 TRUE，该运行期系统的特定功能将继续执行程序。在发生诸如看门狗出错等情况时，该功能应该在相应的 SPG 中调用，以自动继续执行程序。

参数	**数据类型**	**描述**
COND	BOOL	如果为 TRUE，则继续执行程序。
OUT	BOOL	如果 COND=TRUE 则为 TRUE，并可以执行程序。

注：这个功能不应在发生被 0 除（SPG 11）、栈溢出（SPG 12）、总线错误（SPG 20）和界限错误（SPG 19）等异常事件时使用。使用该功能需谨慎。需确保发生看门狗出错等异常情况时，继续执行程序不会引起任何破坏。

这个 ProConOS 功能在输入为 TRUE 的情况下，继续程序的执行。这个功能应该在相应的 SPG 中调用，以便在发生诸如监视定时器等错误的情况下，自动地继续程序的执行。

注意：这个功能不应该用于被 0 除（SPG 11）、栈溢出（SPG 12）、总线错误（SPG 20）和界限错误（SPG 19）等异常事件。

注意：要谨慎地使用这个功能。确保若继续程序的执行，诸如监视定时器错误等异常情况不会引起任何毁坏。

3、HOT_ RESTART 执行 PLC 热重启

功能：如果输入为 TRUE，该运行期系统的特定功能执行热重启。热重启过程中，数据不被初始化。

在发生栈溢出、字符串错误或者被 0 除等问题时，该功能在相应的 SPG 中被调用，以便自动重启执行程序。发生异常后，会立即执行热重启。

如果是在普通 PLC 应用程序中调用，该功能则首先完成当前周期的程序执行，然后执行热重启。

参数	**数据类型**	**描述**
COND	BOOL	如果为 TRUE，则执行热重启。
OUT	BOOL	如果 COND=TRUE 则为 TRUE，并可以进行热重启。

注：eCLR 不能使用 HOT_RESTART 功能，这是因为发生异常后，eCLR 运行期系统将处于仅可冷启动或暖启动的状态。使用该功能需谨慎。确保发生被 0 除或字符串错误等异常情况时，不会因为重启执行程序，而引起任何毁坏。

这个 ProConOS 功能在输入为 TRUE 的情况下执行一个热再启动(hot restart)。在热再启动的过程中，不初始化数据。

应该在相应的 SPG 中调用这个功能，以便在发生堆栈溢出、字符串错误或者被 0 除等问题时，自动重新启动程序的执行。在发生异常之后，立即执行热启动。

如果在常规的 PLC 应用程序中调用了此功能，以执行热再启动，则会首先完成当前循环的程序执行，然后才执行热再启动。

注意：要谨慎地使用这个功能。确保被 0 除或字符串错误等异常情况，不会通过重新启动程序的执行，而引起任何毁坏。

4、IMEMCPY 数据区域管理

功能：这个运行期系统的特定功能将数据从运行期系统的数据区域(源数据区域)复制到另一个运行期系统的数据区域(目标数据区域)。该数据区域是运行期系统存储器的分区，可根据相应的运行期系统逻辑存储器地址进行访问。这个功能通过连接到 SRC_OFF 和 DST_OFF 参数的变量进行复制并加索引。

参数	**数据类型**	**描述**
输入参数		
CNT	INT	需复制的字节数。
SRC	BYTE	位于源数据区域始端的变量。
SRC_OFF	INT	与源数据区域始端相关的以字节为单位的偏移量/索引。
DST	BYTE	位于目标数据区域始端的变量。
DST_OFF	INT	与目标数据区域始端相关的以字节为单位的偏移量/索引。
输出参数		
IMEMCPY	INT	错误码： 0 数据复制，未出现错误。 14 缓冲区超出运行期系统数据段。 15 目标区域为输入组。

注：针对这一类型的数据管理，建议使用 STRUCT 和 ARRAY 数据类型。这个功能可能只对特殊数据操作有效。如果从运行期系统的存储器区域“M”复制数据，或者将数据复制到该区域，需要注意该部分的存储器区域被编程系统用于符号变量的自动编址。因此建议只将此功能用于诸如输入、输出和用户标志等定位变量。

5、MEMCPY 数据区域管理

功能：这个运行期系统的特定功能将数据从运行期系统的数据区域(源数据区域)复制到另一个运行期系统的数据区域(目标数据区域)。该数据区域是运行期系统存储器的分区，可根据相应的运行期系统逻辑存储器地址进行访问。

参数	**数据类型**	**描述**
输入参数		
ERR	INT	错误码： 0 数据复制，未出现错误。 14 缓冲区超出运行期系统数据段。 15 目标区域为输入组。 注：这是位于功能块左侧的输出参数！
CNT	INT	需复制的字节数。
SRC	BYTE	位于源数据区域始端的变量。
DST	BYTE	位于目标数据区域始端的变量。
输出参数		
MEMCPY	WORD	该参数无任何功能。

注：针对这一类型的数据管理，建议使用 STRUCT 和 ARRAY 数据类型。这个功能可能只对特殊数据操作有效。如果从运行期系统的存储器区域“M”复制数据，或者将数据复制到该区域，需要注意该部分的存储器区域被编程系统用于符号变量的自动编址。因此建议只将此功能用于诸如输入、输出和用户标志等定位变量。

6、MEMSET 数据区域管理

功能：该运行期系统的特定功能将给定存储器区域设定为连接到 VAL 的操作数的值。

参数	**数据类型**	**描述**
输入参数		
ERR	INT	错误码： 0 数据复制，未出现错误。 14 缓冲区超出运行期系统数据段。 15 目标区域为输入组。 注：这是位于功能块左侧的输出参数！

VAL	BYTE	需设定的值
CNT	DINT	需设定的字节数。
DST	BYTE	目标缓冲区的首字节。
输出参数		
MEMSET	INT	该参数无任何功能。

注：针对这一类型的数据管理，建议使用 STRUCT 和 ARRAY 数据类型。这个功能可能只对特殊数据操作有效。如果从运行期系统的存储器区域“M”复制数据，或者将数据复制到该区域，需要注意该部分的存储器区域被编程系统用于符号变量的自动编址。因此建议只将此功能用于诸如输入、输出和用户标志等定位变量。

7、RD_*_BY_SYM 从 PDD 的符号变量中读取值

功能：该运行期系统的特定功能读取 PDD 内的符号变量值，变量的名称定义为字符串。对于局部变量，必须使用完整的实例名称“程序实例.功能块实例.变量名称”，实例路径以点号分隔。功能块实例为可选，取决于是否在“物理硬件”树中插入了功能块实例。对于全局变量，‘@GV’和变量名称以点号分隔。

参数	**数据类型**	**描述**
输入参数		
IN	STRING	包含了要读取的符号变量名称的字符串。
ERR	INT	错误码： 0 未出现错误。 1 在 PDD 中未发现变量。 2 数据类型不支持。 3 字符串错误。
输出参数		
RD_*_BY_SYM	BYTE, SINT, WORD, INT, DWORD, DINT, STRING, REAL, LREAL, TIME, UDINT, UINT, USINT	所读取的值。

注：必须使用{PDD}变量属性（pragma）来声明该变量。关于声明变量的信息，可参考编程系统的主帮助。在支持 PDD 的情况下，此功能仅适用 IPC_28、M68_28、ProConOS 2.9 及以上版本。

8、WARM_RESTART 执行 PLC 暖重启

功能：在输入为 TRUE 的情况下，该运行期系统的特定功能将执行一个暖重启。在暖重启过程中，只初始化非保持型数据。在发生栈溢出、字符串错误或者被 0 除等问题时，该功能在相应的 SPG 中被调用，以便自动重启执行程序。发生异常时该功能立即执行一个暖重启。如在普通 PLC 应用程序中调用，该功能会首先执行完当前周期的程序，然后执行暖重启。

参数	数据类型	描述
COND	BOOL	如果为 TRUE，则执行一个暖重启。
OUT	BOOL	如果 COND=TRUE 则为 TRUE，并可以进行暖重启。

注：使用该功能需谨慎。需确保发生被 0 除或字符串错误等异常情况时，重启执行程序不会引起任何破坏。

如果输入为TRUE，这个ProConOS功能执行一个暖再启动(warm restart)。在暖再启动过程中，只初始化非保持型数据。

应该在相应的SPG中调用这个功能，以便在发生堆栈溢出、字符串错误或者被0除等问题时，自动重新启动程序的执行。在发生异常之后，立即执行一个暖再启动。

如果在常规的PLC应用程序中调用此功能以执行一个暖再启动，则要首先完成当前循环的程序执行，然后再执行暖启动。

注意：要谨慎地使用这个功能。确保被 0 除或字符串错误等异常情况，不会通过重新启动程序的执行，而引起任何毁坏。注意：对于 RETAIN 变量，暖再启动后，变量数值为上次保存的数值。因此，暖再启动相当于断电重启。

9、WR_*_BY_SYM 向 PDD 的符号变量写入值

功能：该运行期系统的特定功能将一个值写入存储在 PDD 的某个变量中。对于局部变量，必须使用完整的实例名称“程序实例.功能块实例.变量名称”，实例路径以点号分隔。功能块实例为可选，取决于是否在“物理硬件”树中插入了功能块实例。对于全局变量，“@GV”和变量名之间以点号分隔。

参数	**数据类型**	**描述**
输入参数		
IN_STR	STRING	包含了要读取变量的符号名称的字符串。
IN_VAL	BYTE, SINT, WORD, DWORD, INT, DINT, STRING, REAL, LREAL, TIME, UDINT, UINT, USINT	需写入到变量的值。

输出参数		
WR_*_BY_SYM	INT	错误码： 0 未出现错误。 1 在 PDD 中未发现变量。 2 数据错误。 3 数据类型不支持。 4 字符串错误。

注：必须使用{PDD}变量属性（pragma）来声明该变量。关于声明变量的信息，可参考编程系统的主帮助。在支持 PDD 的情况下，此功能仅适用 IPC_28、M68_28、ProConOS 2.9 及以上版本。

五、运行期系统的特定功能块

运行期系统的特定功能块在“PROCONS”固件库里面，需要导入此固件库才能使用。

1、BUF 型转换为其它类型

BUF 型数据可分为 12 个功能块，可将基本数据类型从字节流复制到变量、数组或者用户自定义结构的元素中，即分别转换为 BYTE、WORD、DWORD、SINT、INT、DINT、USINT、UINT、UDINT、REAL、STRING、TIME 等数据，它主要用于数据传送或者在不同硬件平台上执行应用中的通讯。

ProConOS 中的 BUF 型数据转换指令需要在编辑向导中，从下拉列表选择“ProConOS”。

源数据必须为字节数组（数据类型为 BYTE），也可是 ANY_BIT（BOOL 除外）或 ANY_INT 型（被转换的字节数不能超出存放目标的数据类型）

BUF 型数据的转换指令

指令	输入值	输出值	描述	
BUF_TO_BYTE	BUF	BYTE	缓冲区的数据格式为 MOTOROLA 或 INTEL，根据需要，源数据也可以是 ANY_BIT(BOOL 除外）或 ANY_INT	被转换的字节数是1
BUF_TO_WORD	BUF	WORD		被转换的字节数是2
BUF_TO_DWORD	BUF	DWORD		被转换的字节数是4
BUF_TO_SINT	BUF	SINT		被转换的字节数是1
BUF_TO_INT	BUF	INT		被转换的字节数是2
BUF_TO_DINT	BUF	DINT		被转换的字节数是4
BUF_TO_USINT	BUF	USINT		被转换的字节数是1

BUF_TO_UINT	BUF	UINT		被转换的字节数是2
BUF_TO_UDINT	BUF	UDINT		被转换的字节数是4
BUF_TO_REAL	BUF	REAL	缓冲区的数据格式为MOTOROLA	输入为IEEE 754浮点数的十六进制码，被转换的字节数可以是4
BUF_TO_STRING	BUF	STRING	缓冲区的数据格式为MOTOROLA或INTEL	输入为字符的ASCII码，被转换的字节数可以是1、2…等正整数
BUF_TO_TIME	BUF	TIME	缓冲区的数据格式为INTEL	输入为以毫秒为单位的十六进制码，被转换的字节数是4，输出单位为秒

BUF_TO_*指令处理的数据类型

引脚		数据类型	描述
REQ	输入	BOOL	上升沿有效
BUF_FORMAT	输入	BOOL	TRUE表示缓冲区数据为MOTOROLA格式；FALSE表示INTEL
BUF_OFFS	输入	DINT	缓冲区中被转换的起始字节序号，0表示缓冲区第一个字节
BUF_CNT	输入	DINT	缓冲区中被转换的字节数
BUFFER	输入-输出	ARRAY	缓冲区，为一个字节数组
DST	输入-输出	数组或ANY	存放区，被转换的字节内容放在这里，DST的类型应与具体的BUF_TO_*类型一致，如BUF_TO_BYTE指令的DST必须是BYTE类型
DONE	输出	BOOL	转换完成后，置1，直到REQ为0
ERROR	输出	BOOL	转换正常，为0，否则置1
STATUS	输出	INT	如果转换不正常，则给出错误代码，具体见下表

注：MOTOROLA 和 INTEL 微处理器的数据存放顺序不同，INTEL 格式为高低字节排列，MOTOROLA 格式为低高字节排列。

错误代码	描述
0	转换过程正常完成
1	BUFFER和DST的输出-输出类型错误
2	超出缓冲区的长度，即将被复制的字节数BUF_CNT大于缓冲区BUFFER的可用字节数
3	超出存放区的长度，即将被复制的字节数BUF_CNT超出存放区长度
4	不支持这个数据类型
5	将要转换的字节长度与存放区的字节长度不对应，前者字节数必须能被后者的字节数整除
6	转换INTEL / MOTOROLA失败

7	字符串长度不适当，对于数据类型串，有必要做额外的检查
8	存放区数据类型错误
9	BUF_OFFS数值不正确
10	BUF_CNT数值不正确
11	缓冲区与存放区地址相同

注：缓冲区 BUFFER 的可用字节数——在缓冲区中从第 BUF_OFFS 个字节开始至最后一个字节

BUF_TO_STRING 指令处理的数据类型

输入变量	**数据类型**	**描述**
REQ	BOOL	上升沿有效
BUF_FORMAT	BOOL	TRUE表示为MOTOROLA格式；FALSE表示INTEL格式
BUF_OFFS	DINT	输入缓冲区中被转换的起始字节，0表示第一个字节
BUF_CNT	DINT	要被转换的字节数量
BUFFER	ARRAY	输入缓冲区，一般为字节数组
DST	STRING	输出，字符串类型
输出变量	**数据类型**	**描述**
DONE	BOOL	转换完成后，置1，直到REQ为0
ERROR	BOOL	转换正常，为0，否则置1
STATUS	INT	如果转换不正常
BUFFER	ARRAY	IN_OUT类型，与输入的BUFFER必须连接同一个变量名
DST	STRING	IN_OUT类型，与输入的DST必须连接同一个变量名

2、其它类型转换为 BUF 型

其它类型可转换为 BUF 型数据，可将变量、数组或者用户自定义结构的元素中的基本数据类型复制到字节流中，共育 12 个指令，分别将 BYTE、WORD、DWORD、SINT、INT、DINT、USINT、UINT、UDINT、REAL、STRING、TIME 等数据转换到 BUF 类型数据中，它主要用于数据传送或者在不同硬件平台上执行应用中的通讯。

ProConOS 中的其它类型转换为 BUF 型指令需要在编辑向导中，从下拉列表选择“ProConOS”。

存放区必须为字节数组，也可是 ANY_BIT（BOOL 除外）或 ANY_INT 型。这些指令的用法与上述“BUF 型转换为其它类型”相似，下面以 STRING_TO_BUF 为例说明。

功能：STRING_TO_BUF 指令用于将字符串复制到缓冲区，缓冲区为一个字节型数组。

STRING_TO_BUF 指令处理的数据类型

输入变量	数据类型	描述

REQ	BOOL	上升沿有效
BUF_FORMAT	BOOL	TRUE表示为MOTOROLA格式；FALSE表示INTEL格式
BUF_OFFS	DINT	输入字符串中被转换的起始字符，0表示第一个字符
BUF_CNT	DINT	要被转换的字符数量
SRC	STRING	输入字符串，也可是字符串常量，如’abcd’
BUFFER	ARRAY	输出缓冲区，一般为字节型数组
输出变量	数据类型	描述
DONE	BOOL	转换完成后，置1，直到REQ为0
ERROR	BOOL	转换正常，为0，否则置1
STATUS	INT	如果转换不正常，则给出错误代码，见下表
SRC	STRING	IN_OUT类型，与输入的SRC必须连接同一个变量名
BUFFER	ARRAY	IN_OUT类型，与输入的BUFFER必须连接同一个变量名

错误代码	描述
0	转换过程正常完成
1	BUFFER和DST的输出-输出类型错误
2	超出缓冲区的长度，即将被复制的字节数BUF_CNT大于缓冲区BUFFER的可用字节数
3	超出存放区的长度，即将被复制的字节数BUF_CNT超出存放区长度
4	不支持这个数据类型
5	将要转换的字节长度与存放区的字节长度不对应，前者字节数必须能被后者的字节数整除
6	转换INTEL / MOTOROLA失败
7	字符串长度不适当，对于数据类型串，有必要做额外的检查
8	存放区数据类型错误
9	BUF_OFFS数值不正确
10	BUF_CNT数值不正确
11	缓冲区与存放区地址相同

3、CLR_ERROR_CATALOG 删除整个错误目录

功能：该运行期系统的特定功能块删除整个错误目录。

参数	**数据类型**	**描述**
输入参数		
EXECUTE	BOOL	如果检测到上升沿，则删除错误目录。

输出参数		
DONE	BOOL	0 不能删除错误目录。 1 成功删除错误目录。

注：ProConOS 3.3.0044 以上版本适用 CLR_ERROR_CATALOG 功能块。功能块必须被实例化：必须在需要使用该 FB 的 POU 变量工作单中声明所选择的实例名称（声明使用 VAR 关键字）。该实例名称必须在 POU 内是唯一的。

4、CLR_OUT 将 I/O 映像的所有输出设置为零

功能：该运行期系统的特定功能块将 I/O 映像的输出设置为 0。如果程序停止执行，该功能块应该在 SPG2 中调用，以便将所有的输出设置为 0。

参数	数据类型	描述
EN	BOOL	如果为 TRUE，则将 I/O 映像的所有输出都设置为零。

注：功能块必须被实例化：必须在需要使用该 FB 的 POU 变量工作单中声明所选择的实例名称（声明使用 VAR 关键字）。该实例名称必须在 POU 内是唯一的。

5、DERIVAT 基于时间的偏差

功能：这个专用于特定运行期系统的功能块实现了对时间的微分。当 ENABLE=TRUE 时，执行该功能块下列程序片段描述了此功能块的功能：

工作原理：

```
IF RUN THEN
XOUT = (3*(XIN-X3)+X1-X2)/(10*TIME_TO_REAL(CYCLE));
X3 = X2;
X2 = X1;
X1 = XIN;
ELSE
XOUT = 0;
X1 = XIN;
X2 = XIN;
X3 = XIN;
END_IF
```

参数	数据类型	描述
输入参数		

ENABLE	BOOL	TRUE=执行该功能块。
RUN	BOOL	FALSE=功能块暂停执行，且输出为 0。
XIN	REAL	输入值。
CYCLE	TIME	采样周期应与执行此功能块的任务周期相对应。
输出参数		
XOUT	REAL	输出值。

注：连接到 CYCLE 上操作数的值，必须与执行此功能块的任务的周期相对应。

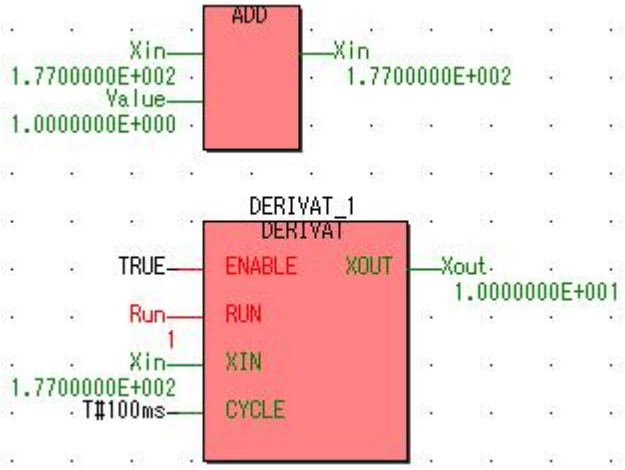
为什么除以 10 个时间周期?

（1）（XIN-X1）+（X1-X2）+（X2-X3）为 3⊿t = ⊿t+⊿t+⊿t

（2）（XIN-X2）+（X1-X3）为 4⊿t = 2⊿t+2⊿t

（3）（XIN-X3）为 3⊿t = 3⊿t

把（1）（2）（3）相加为 10⊿t=（3*（XIN-X3）+X1-X2）。所以，除以 10 个时间周期⊿t 进行微分。



6、EVENT_TASK 触发事件任务的执行

功能：通过使用这个运行期系统的特定功能块可以触发事件任务的执行。通过定义事件编号，功能块 Event_Task 产生一个事件。如果对应此事件编号的事件任务存在，则该事件任务被激活，即分配给该事件任务的程序处理一次。

参数	**数据类型**	**描述**
输入参数		
EXECUTE	BOOL	如果检测到一个上升沿，则生成一个事件。
EVENT_NO	UINT	要生成的事件编号。

输出参数		
ERROR	BOOL	错误码： 0 未出现错误。 1 事件编码不在有效值范围内。

注：功能块不验证给定编号的事件任务是否可用。该功能块无法验证事件任务是否处理成功。ProConOS 3.3.0004 以上版本适用 EVENT_TASK 功能块。功能块必须被实例化：必须在需要使用该 FB 的 POU 变量工作单中声明所选择的实例名称（声明使用 VAR 关键字）。该实例名称必须在 POU 内是唯一的。以下例子中，使用了实例名称“EVENT_TASK_1”。

7、FILE_CLOSE 关闭文件

功能：该运行期系统的特定功能块关闭一个已打开的文件。

参数	数据类型	描述
输入参数		
EXECUTE	BOOL	如果检测到上升沿，则关闭文件。
HANDLE	UINT	要关闭文件的文件句柄。
输出参数		
DONE	BOOL	0 功能块不能被执行。 1 功能块被执行。
ERROR	BOOL	0 关闭文件时未出现错误。 1 关闭文件时，出现错误。
ERRORID	UINT	关闭文件时，发生错误的错误编号： 0 没有可用的错误信息。 1 无效的文件句柄。 20 无法关闭文件。

注：当终止应用程序时，要想自动关闭已打开的文件，那么对于系统程序 2(SPG2)中的每个已打开的文件,必须调用运行期系统的特定功能块 FILE_CLOSE。输出 DONE、ERROR 和 ERRORID 的状态一直保留到 EXECUTE 输入检测到一个下降沿为止。 如果 ProConOS 版本为 3.3.0070 以上，可以使用 FILE_CLOSE 功能块。功能块必须被实例化:必须在需要使用该 FB 的 POU 变量工作单中声明所选择的实例名称(声明使用 VAR 关键字)。该实例名称必须在 POU 内是唯一的。

8、FILE_OPEN 打开/创建文件

功能：该运行期系统的特定功能块打开一个现有文件或创建一个新的文件。如果要打开的文件已经打开则不采取任何处理。在此种情况下 ERRORID 输出返回错误 ID 4。

参数	数据类型	描述
输入参数		
EXECUTE	BOOL	如果检测到上升沿，则打开/创建文件。
NAME	STRING	将要打开/创建的文件名称，包括路径。
输出参数		
DONE	BOOL	0 功能块不能被执行。 1 功能块被执行。
HANDLE	UINT	要打开/创建文件的文件句柄。
ERROR	BOOL	0 打开/创建文件时未出现错误。 1 打开/创建文件时出现错误。
ERRORID	UINT	打开/创建文件时，产生错误的错误编号： 0 没有可用的错误信息。 2 打开的文件数量已到达最大值。 4 文件已经被打开。 5 文件被写保护或者拒绝访问。 6 文件名称未定义。

注;只能用 FILE_CLOSE 功能块关闭已打开的文件。任何用户操作都无法自动关闭文件。最多可同时打开 8 个文件。 文件名称的数据类型必须是 STRING。文件名称的长度包括路径在内不得超过 80 个字符。不支持用户自定义的字符串。 即使在 EXECUTE 输入已非激活状态，功能块输出仍保留句柄的值。其它所有文件操作均要求已打开/创建文件的文件句柄。 输出 DONE、ERROR 和 ERRORID 的状态一直保留到 EXECUTE 输入检测到一个下降沿为止。 ProConOS 3.3.0070 以上版本适用 FILE_OPEN 功能块。功能块必须被实例化：必须在需要使用该 FB 的 POU 变量工作单中声明所选择的实例名称（声明使用 VAR 关键字）。该实例名称必

须在 POU 内是唯一的。

9、FILE_READ 从文件读取数据

功能：该运行期系统的特定功能块从文件读取数据。

参数	**数据类型**	**描述**
输入参数		
EXECUTE	BOOL	如果检测到上升沿，则从文件读数据。
HANDLE	UINT	要读取文件的文件句柄。
MAXLENGTH	UDINT	要读取的字符数。
输入/输出参数		
BUFFER	ANY	要读取数据的数据缓冲区。
输出参数		
DONE	BOOL	0 功能块不能被执行。 1 功能块被执行。
LENGTHREAD	UDINT	读取的字符数。
ERROR	BOOL	0 读取时，未出现错误。 1 读取时，出现错误。
ERRORID	UINT	读取时发生错误的错误编号： 0 没有可用的错误信息。 1 无效的文件句柄。 10 已达到数据的末尾。 12 要读取的字符数超出数据缓冲区大小。 22 读取不到任何数据。

注：可以用不同方法来声明要读数据的数据缓冲区。数据缓冲区类型是一个用户定义的数据类型，例如，Byte Array。数据类型声明如下：

```
TYPE
      FileBuffer :ARRAY [1..100] OF BYTE;
END_TYPE
```

在此情况下数据缓冲区长度为 100 字符。

字符串不能直接用作数据缓冲区。如果不能将所读数据处理成 String，那么首先要在数组中存储该数据，然后使用功能块 BUF_TO_STRING 将其转换成 String。 输出 DONE、LENGTHREAD、ERROR 和 ERRORID 的状态一直保留到 EXECUTE 输入检测到一个下降沿为止。 ProConOS 3.3.0070 以上版本适用 FILE_READ 功能块。功能块必须被实例化：必须在需要使用该 FB 的 POU 变量工作单中声明所选择的实例名称（声明使用 VAR 关键字）。该实例名称必须在 POU 内是唯一的。

10、FILE_REMOVE 删除文件

功能：该运行期系统的特定功能块删除文件。

参数	**数据类型**	**描述**
输入参数		
EXECUTE	BOOL	如果检测到上升沿，则删除文件。
NAME	STRING	将要删除的文件的名称，包括路径。
输出参数		
DONE	BOOL	0 功能块不能被执行。 1 功能块被执行。
ERROR	BOOL	0 删除文件时，未出现错误。 1 删除文件时，出现错误。
ERRORID	UINT	删除文件时，发生错误的错误编号： 0 没有可用的错误信息。 2 打开的文件数量已到达最大值。 3 无法找到文件。 5 文件打开，写保护或者被拒绝访问。 6 文件名称未定义。 21 文件无法删除。

注：文件名称的数据类型必须是 STRING。文件名称的长度包括路径在内不得超过 80 个字符。不支持用户自定义的字符串。 在使用 FILE_CLOSE 功能块删除文件之前，必须关闭要删除的文件。 输出 DONE、ERROR 和 ERRORID 的状态一直保留到 EXECUTE 输入检测到一个下降沿为止。 ProConOS 3.3.0070 以上版本适用 FILE_REMOVE 功能块。功能块必须被实例化：必须在需要使用该 FB 的 POU 变量工作单中声明所选择的实例名称（声明使用 VAR 关键字）。该实例名称必须在 POU 内是唯一的。

11、FILE_SEEK 将文件标记设置在文件中的任何位置

功能：该运行期系统的特定功能块在文件的任意位置设置文件标记。

参数	数据类型	描述
输入参数		
EXECUTE	BOOL	如果检测到上升沿，则设置新的位置。
HANDLE	UINT	要设置位置的文件的文件句柄。
POSITION	DINT	文件内的新位置。
MODE	UINT	定位模式 0 相对文件始端的定位。 1 相对当前位置的定位。 2 相对文件末尾的定位。
输出参数		
DONE	BOOL	0 功能块不能被执行。 1 功能块被执行。
ERROR	BOOL	0 设置位置时，未出现错误。 1 设置位置时，出现错误。
ERRORID	UINT	设置位置时，产生错误的错误编号： 0 没有可用的错误信息。 1 无效的文件句柄。 13 无效的定位模式，或者所指定的位置超出文件 24 位置不能被设置。

注：输出 DONE、ERROR 和 ERRORID 的状态一直保留到 EXECUTE 输入检测到一个下降沿为止。

Specifying a position beyond the end of file is possible.如果该位置被写入，则在文件末尾和当前位置之间补 0。 ProConOS 3.3.0070 以上版本适用 FILE_SEEK 功能块。功能块必须被实例化：必须在需要使用该 FB 的 POU 变量工作单中声明所选择的实例名称（声明使用 VAR 关键字）。该实例名称必须在 POU 内是唯一的。

12、FILE_TELL 确定文件标记在文档中的当前位置

功能：该运行期系统的特定功能块确定了文件标记的当前位置。

参数	数据类型	描述
输入参数		
EXECUTE	BOOL	如果检测到上升沿，则确定当前文件标记的位置。
HANDLE	UINT	需要确定文件位置的文件句柄。
输出参数		
DONE	BOOL	0 功能块不能被执行。 1 功能块被执行。
POSITION	DINT	在文件中的当前位置。
ERROR	BOOL	0 确定位置时未出现错误。 1 确定位置时出现错误。
ERRORID	UINT	确定位置时，出现错误的错误编号： 0 没有可用的错误信息。 1 无效的文件句柄。

注：输出 DONE、POSITION、ERROR 和 ERRORID 的状态一直保留到 EXECUTE 输入处检测到一个下降沿为止。 ProConOS 3.3.0070 以上版本适用 FILE_TELL 功能块。功能块必须被实例化：必须在需要使用该 FB 的 POU 变量工作单中声明所选择的实例名称（声明使用 VAR 关键字）。该实例名称必须在 POU 内是唯一的。

13、FILE_WRITE 往文件中写入数据

功能: 该运行期系统的特定功能块向文件写数据 002E

参数	数据类型	描述
输入参数		
EXECUTE	BOOL	如果检测到上升沿，则向文件写数据。
HANDLE	UINT	要写数据的文件的文件句柄。
LENGTH	UDINT	要写入的字符数。
输入/输出参数		
BUFFER	ANY	包含要写入数据的数据缓冲区。
输出参数		
DONE	BOOL	0 功能块不能被执行。

		1 功能块被执行。
LENGTHWRITTEN	UDINT	写入字符的数量。
ERROR	BOOL	0 写数据时未出现错误。 1 写入时出现错误。
ERRORID	UINT	写入出错的错误编号： 0 没有可用的错误信息。 1 无效的文件句柄。 11 没有内存可写入数据。 12 要写入的字符数超出数据缓冲区的大小。 23 无法写入任何数据

注：可以用不同方法来声明要写数据的数据缓冲区。数据缓冲区类型是一个用户定义的数据类型，例如，Byte Array。数据类型声明如下：

```
TYPE
       FileBuffer :ARRAY [1..100] OF BYTE;
END_TYPE
```

在此情况下数据缓冲区长度为 100 字符。

字符串不能直接用作数据缓冲区。如果需要将字符串存储在一个文件夹里，那么首先要使用功能块 STRING_TO_BUF 将其存储在一个数组中。

输出 DONE、LENGTHWRITTEN、ERROR 和 ERRORID 的状态一直保留到 EXECUTE 输入检测到一个下降沿为止。ProConOS 3.3.0070 以上版本适用 FILE_WRITE 功能块。

功能块必须被实例化：必须在需要使用该 FB 的 POU 变量工作单中声明所选择的实例名称（声明使用 VAR 关键字）。该实例名称必须在 POU 内是唯一的。

14 、FPID 比例+积分+导数控制器(第 2 序列)

功能：这个运行期系统的特定功能块是一个二阶的比例+积分+导数回路控制器。

可以通过设置变量 REMOTE，来选择远程（TRUE）或者本地（FALSE）设定点。变量 AUTO 声明 FPID 是自动起作用还是手动设置。

由于其内部互锁特征，可以将一个固定值设置为输出值（例如，PLC 上出错时）。

还可以声明一个低值（LOW）和一个高值（HIGH），以便设定输出限值。

积分部分避免了手动>自动及方向切换开关工作时遭受到冲击。SPR、SPL 及 X 的高和低范围必须是相同的。

手动和 X 的高和低的范围是 0..100%。

状态	0	1
设定点开关	局部	远程方向开关
自动开关	手动	自动
内部互锁开关	常规	互锁

Multiprog 软件提供的 FPID 和 PID 功能块可以实现 PID 控制。PID 功能块在 Proconos 库中，须先将库添加到 Project 中。

FPID/PID 参数	参数含义及设定范围	说明
FPID.REMOTE(BOOL)	远程(TRUE)或本地(FALSE)设定点之间的开关	
FPID.AUTO(BOOL) PID.AUTO(BOOL)	自动(TRUE)或手动(FALSE)控制之间的开关。手动时输出 Yman 的值	
FPID.DIRECTN(BOOL)	用于设定控制方向的开关（TRUE 意味着反作用控制，FALSE 意味着正作用控制）	
FPID.INTLCK(BOOL)	用于将 INTLCKV 所定义值设置为控制器输出的开关。万一 PLC 上有错误，这个开关可以被设置为 TRUE，然后，INTLCKV 将被设置为控制器输出	
FPID.Tscan(REAL) PID.CYCLE(TIME)	功能块调用之间的持续时间（单位为秒），意味着多长时间这个功能块再次被执行	若是 Cyclic Task，则 Intervaltime(ms)=TSCAN*1000
FPID(Yman)(REAL) PID.X0(REAL)	用于手动控制的输出值	可按阀门开度（%）设置，例如全开=100.0，半开=50.0，全关=0.0
FPID.SPR(REAL)	远程设定点（SV），当 REMOTE=TRUE 时起作用	
FPID.SPL(REAL) PID.SP(REAL)	本地设定点（SV），当 REMOTE=FALSE 时起作用	
FPID.X(REAL) PID.PV(REAL)	过程值（PV）	和模拟量输入模块的输入有关

FPID.KP(REAL) PID.KP(REAL)	比例常数（K），无单位	KP 不应设置为零，建议最小值 0.01;KP 值越大，比例作用越强
FPID.TI(REAL) PID.TR(REAL)	积分时间常数（单位秒）	TI=+∞时，没有积分作用；Ti 值越小，积分作用越强
FPID.TD(REAL) PID.TD(REAL)	微分时间常数（单位秒）	TD=0.0 时，没有微分作用；TD 值越大，微分作用越强
FPID.HIGH(REAL)	控制器输出的上限	可按阀门开度（%）设置，例如全开=100.0
FPID.LOW(REAL)	控制器输出的下限	可按阀门开度（%）设置，例如全关=0.0
FPID.INTLCKV(REAL)	内部互锁值，当 INTLCK=TRUE 时，Yout 等于此值	
FPID.Yout(REAL) PID.XOUT(REAL)	调节值（输出值 MV）	和模拟量输出模块 ADAM-5024 的输出有关联
PID.Enable(BOOL)	FALSE=控制被禁止 TRUE=控制被使能	当一个周期信号将控制此 PID 功能块的执行时，可能会使用这个输入参数

2、单回路控制系统中调节器正反作用的选择

任何一个控制系统在投运前，必须正确选择调节器的正反作用，使控制作用的方向对头，否则，在闭合回路中进行的不是负反馈而是正反馈，它将不断增大偏差，最终必将把被控变量引导到受其他条件约束的高端或低端极限值上。

在一个单回路控制系统中，只要调节器的放大系数 Kc、调节阀的放大系数 Kv、被控对象的放大系数 Ko 的乘积为正，就能实现负反馈控制。调节器、调节阀和对象放大系数正负号规定如下。

（1）调节器放大系数的正负号

对于调节器来说，按照统一的规定，测量值增加，输出增加，调节器放大系数 Kc 为负，称之为正作用。测量值增加，输出减小，Kc 为正，称之为反作用。

（2）调节阀的放大系数的正负号

调节阀的放大系数 Kv 定义为气开阀 Kv 为正，气关阀 Kv 为负。

（3）对象放大系数的正负号

对象的放大系数 Ko 定义为：如操纵变量增加，被控变量也增加，Ko 为正；操纵变量增加，被控变量减少，Ko 为负。由此可知，单回路控制系统调节器正

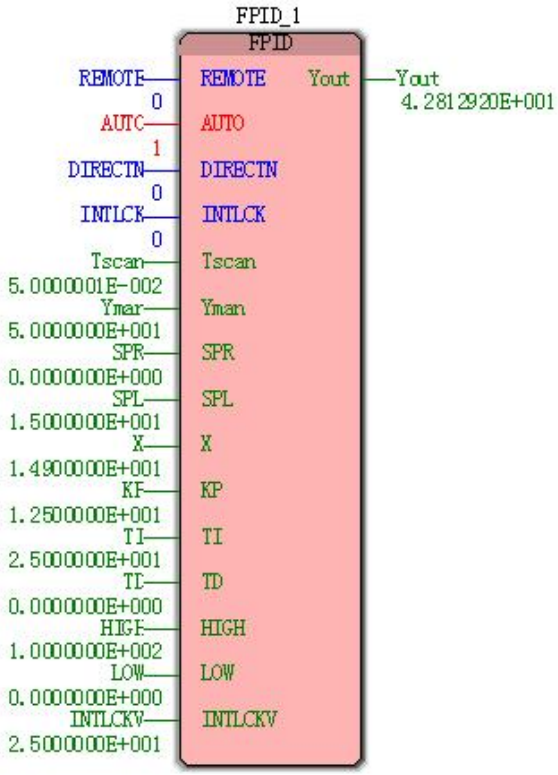
反作用的确定方法如下：首先确定对象放大系数 Ko 的正负号，然后根据调节阀选型为气开或气关确定调节阀放大系数 Kv 的正负号，最终由 Kc、Kv、Ko 乘积应为正，即可确定调节器的作用方式。

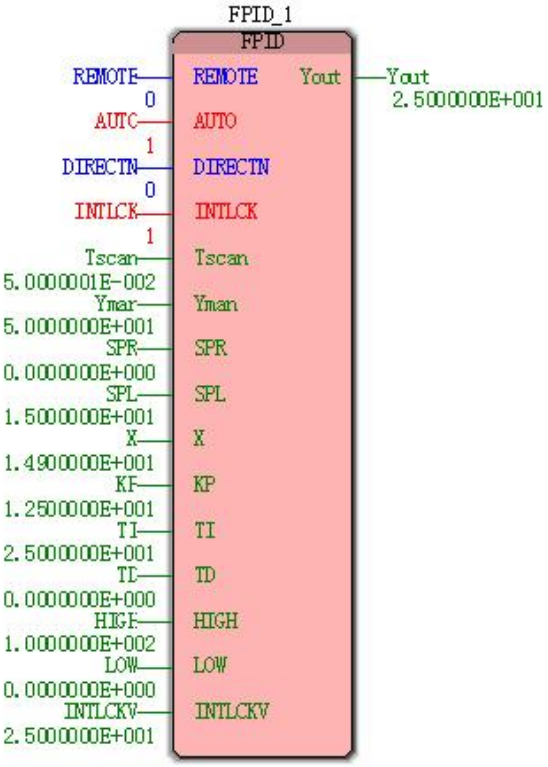
单回路控制系统调节器正反作用选择表

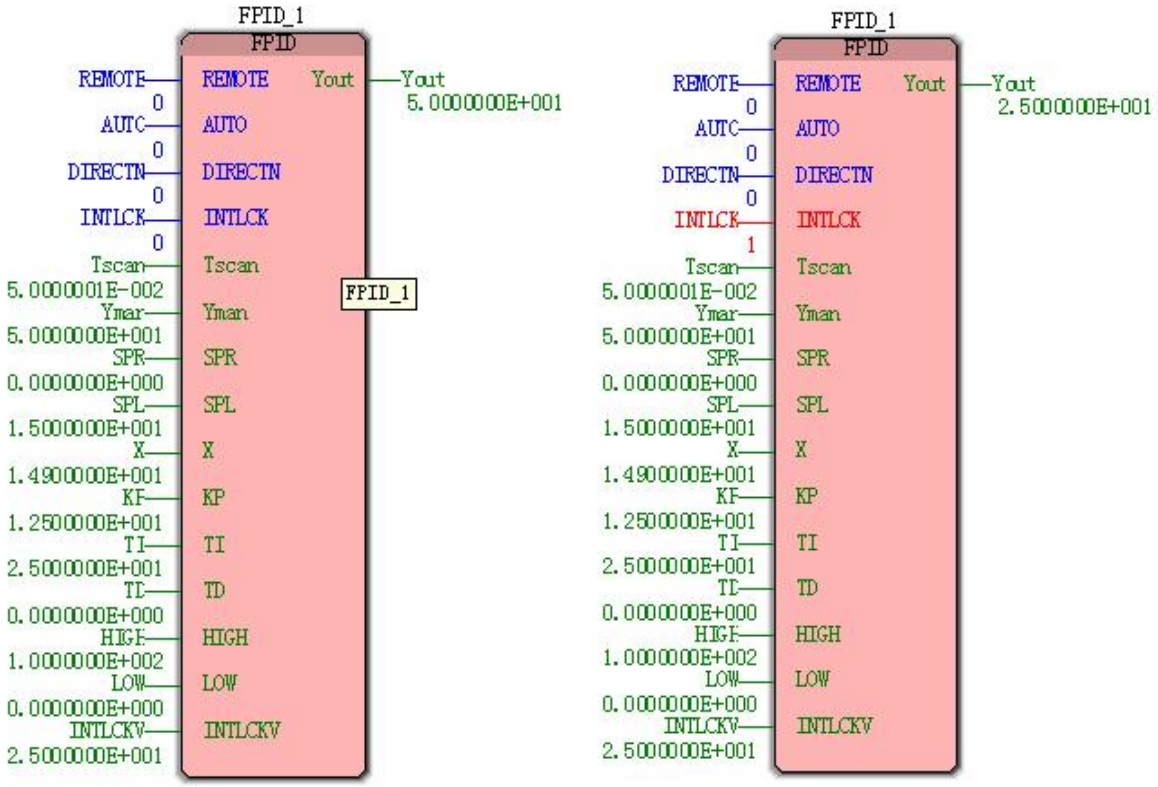
对象放大系数	调节阀	调节器	对象放大系数	调节阀	调节器
正号	气开	反作用	负号	气开	正作用
	气关	正作用		气关	反作用

3、 PID 控制应用举例

如下图，若 AUTO 为 1，Tscan=0.05s，功能块进行自动调节，设定值采用 SPL 的值。若 INTLCK=1，则 Yout=INTLCKV，若 AUTO=0，则则 Yout=Yman。







4、PID 参数的调整

PID 控制的效果就是看反馈（即被控对象 PV）是否跟随设定值 SV，是否响应快速、稳定，是否能够抑制闭环中的各种扰动而回复稳定，PV 值是否在理想的调节精度内，输出（MV）是否在一个稳定的开度范围内。

要调整 PID 参数，判断 PID 参数是否合适，必须能够连续观察 PV 相对于给定设置的响应变化曲线；而实际上 PID 的参数也是通过观察反馈波形而调试的。可以使用 Multiprog 软件中的逻辑分析仪，或使用带慢扫描记忆功能的示波器（如数字示波器），波形记录仪，或者在 PC 机上通过组态软件做的趋势曲线监控图等。

另外，调试 PID 参数是一个艰辛的过程，对于慢速反应系统还要花几小时甚至更长来观察响应的变化。

以下给出单回路 PI 控制系统参数的调整步骤可供参考：

（1）在开始调整之前，先进行开环测试，确认调节的精度、调节阀的理想调节开度、超调度等等。

对 ADAM-5510KW，也可先将其设置为配置模式（Dip Switch Bit7=ON Bit8=ON），用 ADAM-4000-5000 Utility 对输入模块进行观察、对输出模块进行手动调整，观察反馈信号是否稳定、输出通道是否工作正常，确保整个反馈系统可控。

（2）首先进行手动调节，调整输出使 PV 接近 SV。可以用 FPID 的手动模式。

（3）设置初始的 PID 参数，例如 KP=1.0，TI=最大（建议 1000.0），TD=0.0，TSCAN 可根据控制系统而定。

（4）调整 SV，使 SV=PV，并将 FPID 设为自动模式。

（5）给定一个阶跃（可增加 SV 的值 5-10%），并观察系统的反馈曲线，另外，从 YOUT 的曲线也能间接反映控制的效果。

（6）调整 KP 和 TI、TD 的参数，可参考如下：

KP：过大的 KP（即增益）会造成反馈参数（PV）和 YOUT 的震荡（此时减小 KP 值，同时可适当增大 TI 的值以减小积分效应）；过小的 KP 会使静态误差变大，即 PV 很长时间或不能到达 SV（此时增大 KP 值，同时可适当减小 TI 的值以增大积分效应）。

TI：当偏差值恒定时，积分时间决定了控制器输出的变化速率。积分时间越短，偏差得到的修正越快。但过短的积分时间有可能造成不稳定。积分时间的长度相当于在阶跃给定下，增益为“1”的时候，输出的变化量与偏差值相等，所需要的时间，也就是输出变化到二倍于初始阶跃偏差的时间。

如果将积分时间设为最大值，则相当于没有积分作用。当没有采用积分控制时，反馈值会一直达不到设定值。因为积分控制的作用在于消除纯比例调节系统固有的“静差”。没有积分控制的比例控制系统中，没有偏差就没有输出量，没有输出就不能维持反馈值与给定值相等。所以永远不能做到没有偏差。

所以，当反馈值迟迟达不到设定值时，应适当减小 TI 的值，并看调节曲线；一般情况下，减小 TI 的值，同时也应适当调整 KP 的值。当 TI 值过小，会导致系统震荡。在上述调试中，减小 TI 的值，即增大积分效应，同时应减小 KP 的值。

15、GET_ERROR 提供存储在 PLC 错误目录中错误的详细信息

功能：该运行期系统的特定功能块根据 PLC 错误目录的错误索引来界定某个特定错误，并摘出将该错误的可用信息。用于查找错误信息的索引必须应用到 ERRORNUMBER 输入。

通过使用 GET_ERROR_CATALOG 功能块，确定当前存储在 PLC 错误目录中的错误数量。例如 GET_ERROR_CATALOG FB 返回一个值为 3（当前在错误目录中有 3 个错误），并且摘出存储在错误目录中第一个错误的可用信息，则应该通过将值 1 赋予 ERRORNUMBER 输入来调用 GET_ERROR FB。

参数	**数据类型**	**描述**
输入参数		
EXECUTE	BOOL	上升沿启动读取操作。
ERRORNUMBER	UINT	存储在 PLC 错误目录中的错误索引，可用于确定信息。
输出参数		
DONE	BOOL	FALSE 没有可信息 TRUE 信息可用。
ISERROR	BOOL	FALSE 该条目未归为错误

		该条目归为错误。
ISWARNING	BOOL	FALSE 该条目未作为警告输入。 TRUE 该条目作为警告输入。
GROUPCODE	WORD	显示当前错误所属的错误组。
ERRORCODE	WORD	显示当前错误的错误 ID。 注：如果错误 ID 无效返回 0 值。
INFOAVAILABLE	BOOL	FALSE 该错误无其它可用的信息。 TRUE 该错误有其它可用的信息。
ERRORINFO	WORD	包含关于当前错误的其它信息，例如，任务编号。

注：在错误目录中的错误编号为 1 至 62。最新错误条目总是以编号 1 存储。

ProConOS 3.3.0044 以上版本适用 GET_ERROR 功能块。

功能块必须被实例化：必须在需要使用该 FB 的 POU 变量工作单中声明所选择的实例名称（声明使用 VAR 关键字）。该实例名称必须在 POU 内是唯一的。

16、GET_ERROR_CATALOG 提供 PLC 错误目录中当前内容的信息

功能：该运行期系统的特定功能块返回错误目录中当前内容的信息。

参数	**数据类型**	**描述**
输入参数		
EXECUTE	BOOL	如果检测到一个上升沿，那么确定错误目录的信息。
输出参数		
DONE	BOOL	FALSE 没有可用信息 TRUE 信息可用。
TABLESIZE	UINT	定义了最多可存储在错误目录中的错误数。
ERRORAMOUNT	UINT	显示错误目录中当前存储的错误数。

注：ProConOS 3.3.0044 以上版本适用 GET_ERROR_CATALOG 功能块。功能块必须被实例化：必须在需要使用该 FB 的 POU 变量工作单中声明所选择的实例名称（声明使用 VAR 关键字）。该实例名称必须在 POU 内是唯一的。

17、GET_ERROR_eCLR 通过错误索引在 PLC 错误目录中查找某个特定错误，并提取关于此错误的信息

功能：该运行期系统的特定功能块根据 PLC 错误目录的错误索引来界定某个特定错误，并摘出将该错误的可用信息。用于查找错误信息的索引必须应用到 ERRORNUMBER 输入。

通过使用 GET_ERROR_CATALOG 功能块，确定当前存储在 PLC 错误目录中的错误数量。假如 GET_ERROR_CATALOG FB 返回一个值为 3（当前在错误目录中有 3 个错误），并且要摘出存储在错误目录中第一个错误的可用信息，则应该通过将值 1 赋予 ERRORNUMBER 输入来调用 GGET_ERROR_eCLR FB。

参数	数据类型	描述
输入参数		
EXECUTE	BOOL	上升沿启动读取操作。
ERRORNUMBER	UINT	存储在 PLC 错误目录中的错误索引，可用于确定信息。
输出参数		
DONE	BOOL	TRUE 功能块执行完成 FALSE 执行未完成。
FOUND	BOOL	TRUE 错误目录中存有带特定错误索引的错误 FALSE 错误目录中不存在带特定错误索引的错
TIMESTAMP	UDINT	错误的时间戳。
SYSTEMERROR	BOOL	如果确定的错误为系统运行期错误则为 TRUE。该位从所应用的 ERRORNUMBER 中摘取。
ERRORLEVEL	UINT	界定摘自所应用的 ERRORNUMBER 的错误信息的警告等级。
LOGLEVEL	UINT	界定错误信息的日志等级。
ERRORCODE	UINT	显示当前错误的错误 ID。（错误目录中最多可有 65534 条错误。） 注：如果错误 ID 无效返回 0 值。
INFOTYPE	UINT	如有更多错误信息，则提供该信息的 ID。

注：功能块必须被实例化：必须在需要使用该 FB 的 POU 变量工作单中声明所选择的实例名称（声明使用 VAR 关键字）。该实例名称必须在 POU 内是唯一的。

18、GET_SYM 在 PDD 中搜索变量的符号名称

功能：该运行期系统的特定功能块搜索 PDD 中变量的符号名称。

对于局部变量，符号名称为完整的实例名称“程序实例.功能块实例.变量名称”，实例路径以点号分隔。功能块实例为可选，取决于是否在“物理硬件”树中插入了功能块实例。

如果支持现场名称，则该功能块也能用于搜索现场名称。现场名称是某个变量的附加描述，该变量用于描述工厂变量的现场拓扑结构。

如果功能块的 EN 输入为 TRUE，则会在 PDD 中搜索连接到 VARIABLE 输入的变量的符号名称。如果在 PDD 中找到了此变量，则符号名称和/或现场名称将被保存到“ymbolicName”和/或“FieldName”输出。如果输出的字符串相对于所连接的字符串变量来说太长，则会调用 SPG21 且 PLC 被设为停止状态。如果不支持 PDD 或者 MSFC，则只设置错误和状态标志。

参数	**数据类型**	**描述**
输入参数		
EN	BOOL	TRUE=执行该功能块。
VARIABLE	ANY	需要搜索符号名称或现场名称的变量。
输出参数		
ERROR	BOOL	TRUE 未发现符号名称。 FALSE 发现符号名称。
STATUS	INT	状态代码： 0 未发现符号名称或现场地址。 1 发现符号名称。 2 发现现场地址。 3 发现符号名称和现场地址。 4 符号名称字符串超长。 5 现场字符串超长。 6 不支持 PDD 或 MSFC。
SymbolicName	STRING	如发现，提供变量的符号名称。
FieldName	STRING	如发现，提供变量的现场名称

注：必须使用{PDD}变量属性（pragma）来声明该变量。关于声明变量的信息，可参考编程系统的主帮助。在系统支持 PDD 和 MSFC 的情况下，此功能块仅适用 IPC_28、M68_28、ProConOS 2.9 及以上版本。功

能块必须被实例化：必须在需要使用该 FB 的 POU 变量工作单中声明所选择的实例名称（声明使用 VAR 关键字）。该实例名称必须在 POU 内是唯一的。在以下例子中，使用了实例名称“GET_SYM_1”。

19、INTEGRAL 对时间的积分

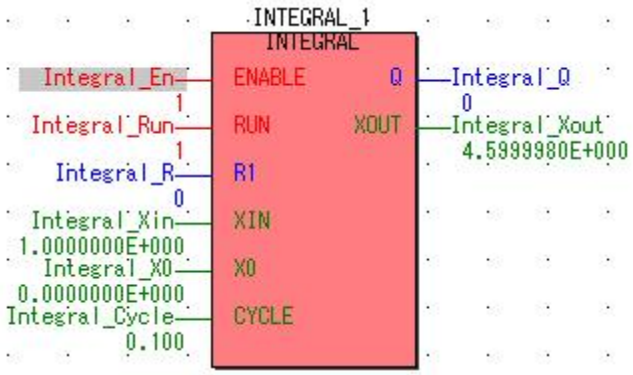
功能：该运行期系统的特定功能块执行对时间的积分。当 ENABLE 被设置为 TRUE 时，执行该功能块。

参数	数据类型	描述
输入参数		
ENABLE	BOOL	TRUE=执行该功能块。
RUN	BOOL	FALSE=暂停功能块的执行。
R1	BOOL	功能块被复位，且以初始值 X0 被初始化。
XIN	REAL	输入值
X0	REAL	初值
CYCLE	TIME	采样周期应与执行此功能块的任务周期相对应。
输出参数		
Q	BOOL	R1 的求反
XOUT	REAL	输出数值

注：连接到 CYCLE 上操作数的值，必须相应于使用了此功能块的任务的循环时间。功能块必须被实例化：必须在需要使用该 FB 的 POU 变量工作单中声明所选择的实例名称（声明使用 VAR 关键字）。该实例名称必须在 POU 内是唯一的。在以下例子中，使用了实例名称“INTEGRAL_1”。

工作原理：

```
IF R1 THEN
XOUT = X0;
ELSIF RUN THEN
XOUT = XOUT + XIN * TIME_TO_REAL(CYCLE);
END_IF
```



20、PID 比例+积分+导数控制(第 3 序列)

功能：这个运行期系统的特定功能块是一个三阶的比例+积分+导数回路控制器。该功能源于 INTEGRAL 和 DERIVAT 功能块的功能组合。

类似该功能块的回路控制器必须以固定的时间间隔来执行。通常有两种方法可满足这个要求。第一种方法是在 POU 内使用这个功能块，由一个任务以固定的时间间隔执行。第二种方法是结合一个周期信号来使用 ENABLE 输入参数，这个周期信号可以由定时器等功能块产生。仅对有较大时间间隔的慢速控制推荐这种方法。

下列程序片段描述了此功能块的功能：

```
VAR
      ERROR :REAL; (* PV -SP *)
      ITERM :INTEGRAL; (* FB for integral term *)
      DTERM :DERIVATIVE; （* 求导项用 FB*）
END_VAR
ITERM(ENABLE = TRUE, RUN = AUTO, R1 = NOT AUTO,
XIN = ERROR, XO = TR * (XO - ERROR), CYCLE = CYCLE);
DTERM(ENABLE = TRUE, RUN = AUTO, XIN = ERROR, CYCLE = CYCLE);
XOUT = KP * (ERROR+ITERM.XOUT/TR+DTERM.XOUT * TD);
```

参数	**数据类型**	**描述**
输入参数		
ENABLE	BOOL	FALSE=禁用控制 TRUE=启用 Control 当采用循环信号控制 PID 功能块的执行时可采用输入参数。
AUTO	BOOL	TRUE=自动模式 FALSE=手动模式 如果 AUTO = FALSE，输出值 XOUT 如下计算：XOUT = KP * XO
PV	REAL	过程的测量值。
SP	REAL	设定点。
XO	REAL	手动输出调整（对中转站很典型）。

KP	REAL	比例常量（放大系数）。这个常量必须为负值。
TR	REAL	复位时间常量（积分项的时间系数——XOUT 到达 KP 值后的时间，不考虑导数项的影响）。
TD	REAL	时间导数常量（导数项的时间系数——与比例项相比较，在导数项导致 XOUT=KP 之前的时间，不考虑积分项的影响）。
CYCLE	TIME	采样期，即 FB 再次计算的时间间隔。
输出参数		
XOUT	REAL	对过程的输出值。

注：连接到 CYCLE 上操作数的值，必须与使用此功能块的任务的周期相对应。功能块必须被实例化：必须在需要使用该 FB 的 POU 变量工作单中声明所选择的实例名称（声明使用 VAR 关键字）。该实例名称必须在 POU 内是唯一的。在以下例子中，使用了实例名称“PID_1”。、

21、PLC_STOP 停止 PLC

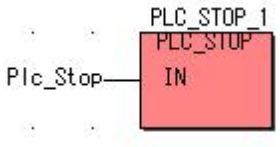
功能：如果在输入端发现上升沿，运行期系统的特定功能块将停止 PLC。PLC 进入停止状态，系统上报 PLC 出错。

点击资源控制对话框中高亮显示的“错误”按钮，即可将该错误上传到消息窗口的“PLC 错误”选项卡下，以通知出现异常情况。

双击提示工作单号（WN）的消息即可打开受影响的工作单进行分析。

参数	**数据类型**	**描述**
输入参数		
ENABLE	BOOL	FALSE=禁用控制 TRUE=启用 Control 当采用循环信号控制 PID 功能块的执行时可采用输入参数。

注：PLC 将立即停止。有一个例外：在某些 PLC 系统中，可以在一个特定错误任务内停止并重新启动 PLC。功能块必须被实例化：必须在需要使用该 FB 的 POU 变量工作单中声明所选择的实例名称（声明使用 VAR 关键字）。该实例名称必须在 POU 内是唯一的。在以下例子中，使用了实例名称“PLC_STOP_1”。



22、RD_BOOL_BY_SYM 从 PDD 的符号变量中读取一个值

功能：该运行期系统的特定功能块读取 PDD 内的一个符号变量的值，变量的名称赋给了一个字符串。对于局部变量，必须使用完整的实例名称“程序实例.功能块实例.变量名称”，实例路径以点号分隔。功能块实例为可选，取决于是否在“物理硬件”树中插入了功能块实例。

对于全局变量，“@GV”和变量名之间以点号分隔。

参数	数据类型	描述
输入参数		
IN	STRING	包含了要读取的符号变量名称的字符串。
ERR	INT	错误码： 0 未出现错误。 1 在 PDD 中未发现变量。 2 数据类型不支持。
输出参数		
OUT	BOOL	所读取的值

注；必须使用{PDD}变量属性（pragma）来声明该变量。关于声明变量的信息，可参考编程系统的主帮助。如果支持 PDD，此功能块仅适用于 IPC_28、M68_28、ProConOS 2.9 及更高版本。功能块必须被实例化：必须在需要使用该 FB 的 POU 变量工作单中声明所选择的实例名称（声明使用 VAR 关键字）。该实例名称必须在 POU 内是唯一的。在以下例子中，使用了实例名称“RD_BOOL_BY_SYM_1”。

23、RD_INPUT_GROUP 读取输入组（为 I/O 映像的一部分）的值

功能：该运行期系统的特定功能块读取输入组 I/O 映像的值。输入组由连接到 VAR 输入的变量地址指定。变量可以是这个组的任意变量。

这个功能块应该用于在调用时刷新 I/O 映像。

参数	数据类型	描述
输入参数		
MEMBER	BYTE	在要读取的输入组中指定的变量。
输出参数		
ERROR	INT	错误码： 0 未出现错误。 1 未发现该输入地址的驱动器。

		2 读取 I/O 映像时出错。 7 变量地址与 I/O 配置中的输入组不对应。

注：功能块必须被实例化：必须在需要使用该 FB 的 POU 变量工作单中声明所选择的实例名称（声明使用 VAR 关键字）。该实例名称必须在 POU 内是唯一的。在以下例子中，使用了实例名称“RD_INPUT_GROUP_1”。

24、RTC_S 将日期和时间写入字符串

功能：该运行期系统的特定功能块将当前日期和时间写入一个字符串。如果参数 IN 被设置为 TRUE，则会将 GMT 的实际日期和时间，按日期和时间的格式写入所连接的输出字符串中。如果 IN 被设置为 FALSE，则不更改字符串。

输出字符串按照 IEC 1131-3 要求前缀标志为日期和时间缩写。

范例：DT#1998-11-21-15:27:56.46

在不带实时时钟的系统上，所连接的输出字符串以零按日期和时间格式填充。

参数	**数据类型**	**描述**
输入参数		
EN	BOOL	如果为 TRUE，则当前的日期和时间被写入所连接的输出字符串中。
输出参数		
Q	BOOL	指示日期和时间被写入连接到 CDT 的字符串中。
CDT	STRING	包含日期和时间。

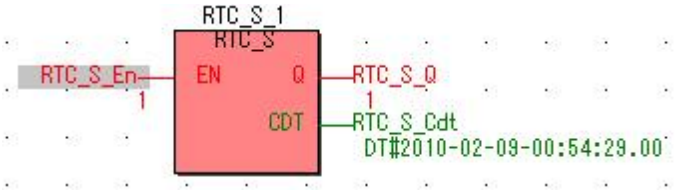
注：功能块必须被实例化：必须在需要使用该 FB 的 POU 变量工作单中声明所选择的实例名称（声明使用 VAR 关键字）。该实例名称必须在 POU 内是唯一的。在以下例子中，使用了实例名称“RTC_S_2”。

仅限带实时时钟的系统支持该功能块。在没有实时时钟的系统上，所连接的输出字符串用以日期和时间格式的零填充。

这个 ProConOS 功能块将日期和时间写入一个字符串。如果参数 IN 被设置为 TRUE，则会将根据 GMT 的实际的日期和时间，以日期和时间的格式写入所连结的输出字符串中。如果 IN 被设置为 FALSE，则不改变字符串。IEC 61131-3 的短前缀符号用于日期和时间输出字符串。DT#1998-11-21-15:27:56.46

注意：这个功能块仅支持带可用实时时钟的系统上。

在没有实时时钟的系统上，所连接的输出字符串用以日期和时间格式的零填充。



25、WR_BOOL_BY_SYM 向 PDD 内的符号变量写入一个值

功能：该运行期系统的特定功能块将值写入已插入 PDD 的变量。

对于局部变量，必须使用完整的实例名称“程序实例.功能块实例.变量名称”，实例路径以点号分隔。功能块实例为可选，取决于是否在“物理硬件”树中插入了功能块实例。

对于全局变量，“@GV”和变量名之间以点号分隔。

参数	数据类型	描述
输入参数		
IN_STR	STRING	包含了要读取变量的符号名称的字符串。
IN_VAL	BOOL	需写入到变量的值。
输出参数		
ERR	INT	错误码： 0 未出现错误。 1 在 PDD 中未发现变量。 2 数据错误。 3 数据类型不支持

注：必须使用{PDD}变量属性（pragma）来声明该变量。关于声明变量的信息，可参考编程系统的主帮助。如果支持 PDD，此功能块仅适用于 IPC_28、M68_28、ProConOS 2.9 及更高版本。功能块必须被实例化：必须在需要使用该 FB 的 POU 变量工作单中声明所选择的实例名称（声明使用 VAR 关键字）。该实例名称必须在 POU 内是唯一的。在以下例子中，使用了实例名称“WR_BOOL_BY_SYM_1”。

26、WR_OUTPUT_GROUP 写入输出组（为 I/O 映像的一部分）的值

功能：该特定运行期系统的专用于功能块向输出组的 I/O 映像写入值。输出组由连接到 VAR 输入的变量地址指定。变量可以是这个组的任意变量。

这个功能块应该用于在调用时刷新 I/O 映像。

参数	数据类型	描述
MEMBER	BYTE	在要读取的输出组中指定的变量。
ERROR	INT	错误码： 0 未出现错误。 1 未发现该输出地址下的驱动器。

		3 写入 I/O 映像时出错。 7 变量地址与 I/O 配置中的输出组不对应。

注：功能块必须被实例化：必须在需要使用该 FB 的 POU 变量工作单中声明所选择的实例名称（声明使用 VAR 关键字）。该实例名称必须在 POU 内是唯一的。在以下例子中，使用了实例名称“WR_OUTPUT_GROUP_1”。

27、WRITE_RETAIN 将保持型数据写入缓冲存储器区域内

功能：该运行期系统特定功能块用于在不具备或者无法直接访问非易失性存储器（non-volatile memory）的系统中将所有声明为保持型的变量保存到文件。 在其它系统上，WRITE_RETAIN 无效。
如果 IN 输入赋给一个上升沿，将会给正在执行实际写入操作的另一个任务发送一个消息。写入过程在非锁定状态下完成。

调用此功能块后，数据被复制到一个内部缓冲区。在下载 PLC 项目时分配该缓冲区的存储器。所分配的缓冲器大小与保持型数据区域大小相一致，在 PLC 应用项目（编程系统）的 VAR_CONF 声明中输入。PLC 复位或有新的项目下载时，释放该存储器。

完成写入过程后，连接到参数 DONE 的变量被设置为 TRUE。只有当 IN 赋予另一个上升沿以再次调用 WRITE_RETAIN FB 时，才将值改为 FALSE。如完成第二个写入过程，DONE 又被设置为 TRUE。

在 PLC 进行暖启动时，完成对保持型数据的读取。引导工程上传到 PLC 后，暖启动完成。在上传期间保存的保持型数据会与实际的项目数据进行比较。如果数据不同，则执行冷启动。

目前，不间断电源，如 NT 系统上的，仅限运行期系统的开发版支持。UPS 会产生一个掉电信号，以启动运行期系统中的写入过程。

参数	**数据类型**	**描述**
IN	BOOL	上升沿启动功能块的写操作。
DONE	BOOL	TRUE 值表示保持型变量已被写入。

注：一次只能执行一个 WRITE_RETAIN 功能块！

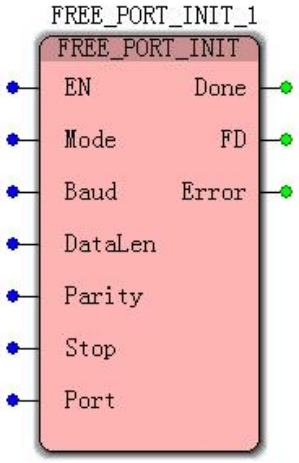
原因：根据保持型数据量的大小，写入操作可能要花费很长时间。因此 WRITE_RETAIN 功能块在调用后以较低优先级方式在服务任务中执行。如果在未完成第一次执行的情况下，再次调用该 FB，由于 DONE 的输出仍然为 FALSE，所以功能块忽略此次调用。只有当全部完成写入操作，DONE 变为 TRUE 后，才允许新的调用。功能块必须被实例化：必须在需要使用该 FB 的 POU 变量工作单中声明所选择的实例名称（声明使用 VAR 关键字）。该实例名称必须在 POU 内是唯一的。在以下例子中，使用了实例名称“WRITE_RETAIN_1”。

第三章 自定义功能块

一、TifsFwlib 库

1、自由口功能块

1.1 FREE_PORT_INIT 自由口初始化



功能：配置自由口所需的物理端口。

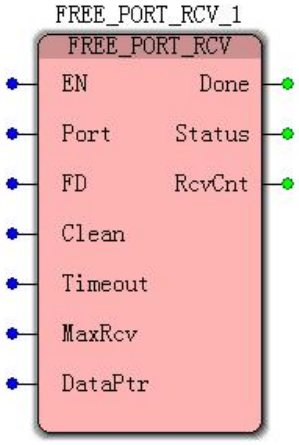
参数	数据类型	描述
EN	BOOL	表示配置并使用自由口功能块，在 EN 检测到上升沿时进行配置，在自由口使用期间，该标志必须一直保持使能
Mode	BYTE	通信模式，保留，默认为 0 即可
Baud	DWORD	自由口的波特率，可选择：1200、2400、4800、9600、19200、38400、57600、115200
DataLen	BYTE	自由口的数据位，5、6、7、8
Parity	BYTE	自由口的校验位，0 无校验；1 奇校验；2 偶校验；3 空格
Stop	BYTE	自由口的停止位，1、2
Port	BYTE	通信所使用的实际物理端口号，可选 0、1、2
Done	BOOL	配置完成标志，完成返回 True
FD	DINT	自由口初始化连接标识符
Error	BYTE	配置错误代码

错误代码：

0　无错误

1 预留
2 波特率错误
3 数据位错误
4 校验位错误
5 停止位错误
6 打开串口失败
7 配置串口失败
8 设置为 485 模式失败
9 端口被占用

1.2 FREE_PORT_RCV 自由口接收数据功能块



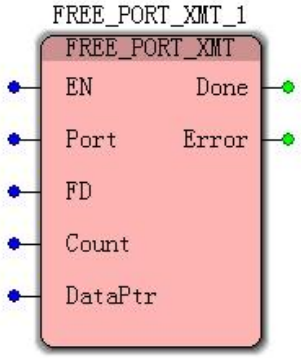
功能：给根据配置接收串行总线数据。

参数	数据类型	描述
EN	BOOL	在 EN 检测到上升沿后就根据设置条件接收数据，直到满足所设定条件或 EN 失能，接收结果由 Status 传出
Port	BYTE	FREE_PORT_INIT 配置过的端口号
FD	DINT	自由口连接标识符，由 FREE_PORT_INIT 配置产生
Clean	BOOL	清除已收到数据和接收状态，以便接收新的数据和产生新的接收状态
Stop	BYTE	结束字符，以该字符为结束接收判断（0-255）
Timeout	DWORD	接收超时时间，判断接收数据超时时间，单位 ms
MaxRcv	BYTE	设定接收最大字节数
DataPtr	ANY	接收数据的存放首地址
Done	BOOL	接收完成标志
Status	BYTE	接收状态代码
RcvCnt	BYTE	实际接收字节数

接收状态代码：

0 功能块无误或失能
1 串口未配置
2 接受的最大字节数 MaxRcv 超出定义缓冲区大小
3 FD 参数非法，0 或者 16#FFFFFFFF
4 接收数据存放地址非法

1.3 FREE_PORT_XMT 自由口发送数据功能块



功能：将指定数据发送到串行总线。

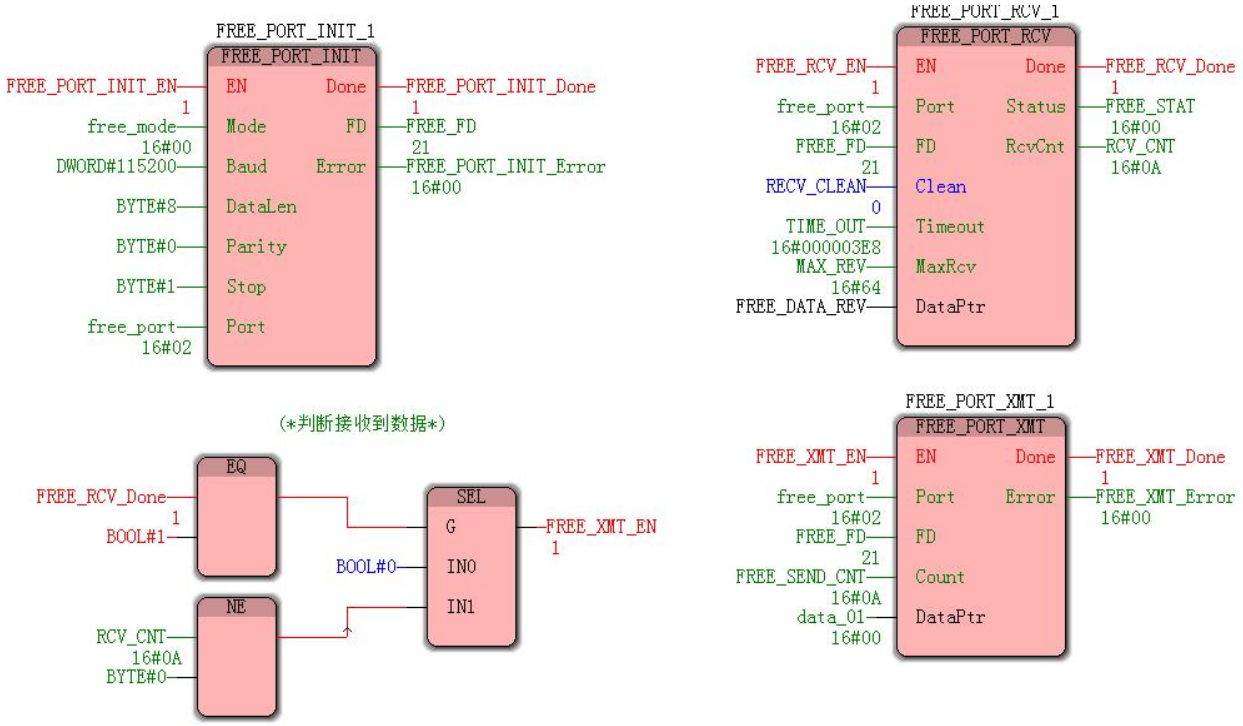
参数	数据类型	描述
EN	BOOL	使用自由口功能块发送数据，仅在 EN 检测到上升沿时发送一次
Port	BYTE	使用 FREE_PORT_INIT 配置过的端口号
FD	DINT	自由口连接标识符，由 FREE_PORT_INIT 配置产生
Count	INT	发送数据的字节数（最大 255 字节）
DataPtr	ANY	发送数据的首地址
Done	BOOL	发送完成标志
Error	BYTE	发送错误代码

错误代码：

0 发送成功
1 串口未配置
2 发送的字节数 Count 超出定义缓冲区大小
3 发送 0 了个字符
4 发送失败

1.4 应用举例

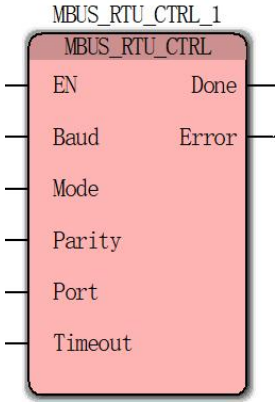
使用 FREE_PORT_INIT 将外部端口 0 配置为 115200 波特率、8 位数据位、无校验、1 位停止位。然后使用 FREE_PORT_RCV 读取总线上任意字符，PREE_DATA_REV 的首地址存放接收数据，data_01 的首地址存放发送数据，当接收到数据后使用 FREE_PORT_XMT 将读到的数据发送到串行总线。



由上图可知使用 FREE_PORT_INIT_1 配置端口 2 配置成功，之后通过 FREE_PORT_ECV_1 接收到了 10 个字节大小的数据，并使用 FREE_PORT_XMT_1 成功发出。

2、MODBUS RTU 主站

2.1 MBUS_RTU_CTRL 功能块



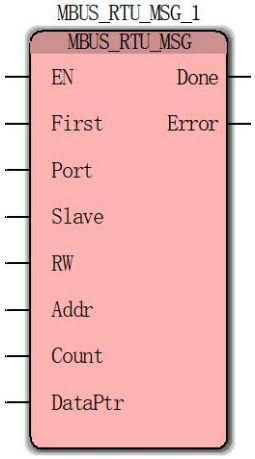
功能：modbus rtu 主站配置功能块，用于通讯端口的配置。

参数	数据类型	描述
EN	BOOL	用于开启 modbus rtu 配置，modbus rtu 功能使用期间需一直使能该变量
Mode	BYTE	通信模式，保留，默认为 0 即可
Baud	DWORD	波特率，可选：1200、2400、4800、9600、19200、38400、57600、115200
Parity	BYTE	校验位，0 无校验；1 奇校验；2 偶校验
Port	BYTE	通信所使用的实际物理端口号，可选 0、1、2
Timeout	WORD	通讯超时时间
Done	BOOL	配置完成标志
Error	BYTE	配置错误代码

错误代码：

0 无错误
1 波特率错误
2 模式参数错误（暂时未定义，保留）
3 奇偶校验参数错误
4 预留
5 超时参数错误（定义成了负数）
6 端口配置失败
7 连接从站失败
8 端口物理通讯模式配置失败（默认固定为 485 模式）
9 端口被占用

2.2 MBUS_RTU_MSG MODBUS RTU 功能块



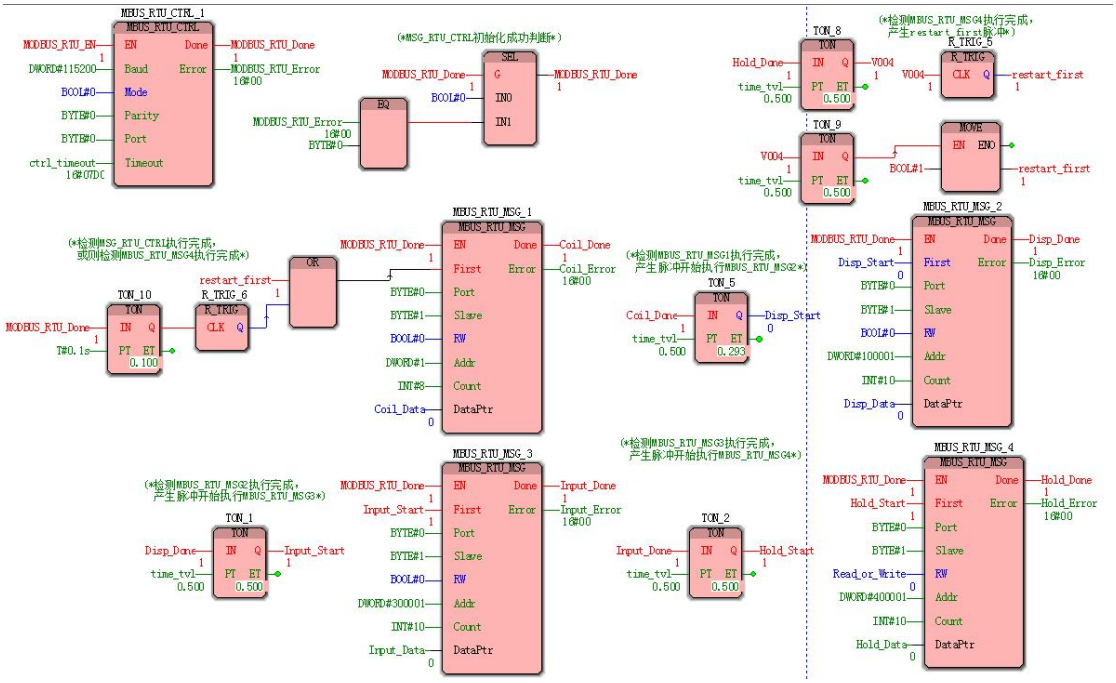
功能：modbus rtu 主站通讯功能块，用于与从站通讯

参数	数据类型	描述
EN	BOOL	通讯使能
First	BOOL	上升沿执行一次通讯，最好使用脉冲控制
Port	BYTE	通信所使用的实际物理端口号，可选 0、1、2，需使能 COM 口
Slave	BYTE	modbus 从机地址（0-255）
RW	BOOL	读写控制，0 读；1 写
Addr	DWORD	从站寄存器地址：1-65535（线圈寄存器）、100001-165535（离散寄存器）、300001-365535（输入寄存器）、400001-465535（保持寄存器）
Count	INT	操作寄存器个数（线圈寄存器最多：1920 个、离散寄存器最多：1920 个、输入寄存器最多：120 个、保持寄存器最多：120 个）
DataPtr	ANY	映射至本地的地址
Done	BOOL	配置完成标志
Error	BYTE	配置错误代码

错误代码：

0 无错误
1 端口未配置成功
2 从机地址配置失败
3 通讯超时
4 MBUS_RTU_CTRL 功能块未使能
5 modbus 寄存器地址参数错误
6 读写位参数非法，读写可为 0 或 1，离散寄存器和输入寄存器只能为 0
7 操作寄存器数目超限

2.3 应用举例



配置 modbus rtu 主站，使用外部端口 0 配置为 115200 波特率、无校验，1000ms 的通信延时。

使用 MBUS_RTU_MSG_1 操作线圈寄存器，将从站 1 的线圈寄存器 1 至 8 读取至 Coil_Data 地址；

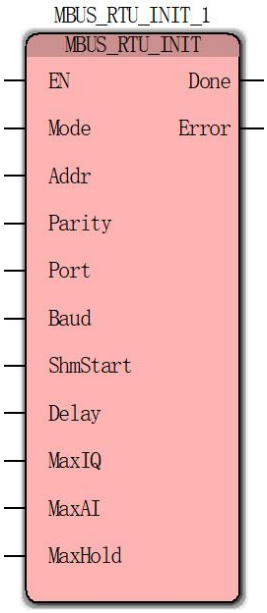
使用 MBUS_RTU_MSG_2 操作离散寄存器，将从站 1 的离散寄存器 1 至 10 数据读至 Disp_Data 地址；

使用 MBUS_RTU_MSG_3 操作输入寄存器，将从站 1 的输入寄存器 1 至 10 数据读至 Input_Data 地址；

使用 MBUS_RTU_MSG_4 操作保持寄存器，将从站 1 的保持寄存器 1 至 10 数据读取至 Hold_Data 地址开启在线调试，使能 MBUS_RTU_CTRL_1,CTRL_Done 为 True 则表示端口配置成功，分别对 Coil_Start、Disp_Start、Input_Start、Hold_Start 进行置 False 为 True 操作，即可完成一次相应寄存器的数据传输操作，为 True 期间，操作结果保持。

3、MODBUS RTU 从站

3.1 MBUS_RTU_INIT 功能块



功能：modbus rtu 从站配置功能块，用于配置物理端口与从站参数。

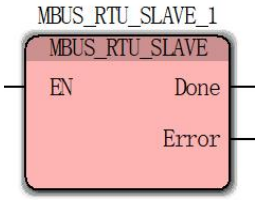
参数	数据类型	描述
EN	BOOL	从站配置使能，使用 modbus rtu 从站功能期间需一直使能
Mode	BYTE	模通信模式，保留，默认为 0 即可
Addr	BYTE	本机地址（从站）
Parity	BYTE	校验位，0 无校验；1 奇校验；2 偶校验
Port	BYTE	通信所使用的实际物理端口号，可选 0、1、2
Baud	DWORD	波特率，可选：1200、2400、4800、9600、19200、38400、57600、115200
ShmStart	ANY	本地 modbus 数据存放缓存的首地址
Delay	WORD	延迟发送时间，收到主站请求延迟 Delay ms 后返回数据
MaxIQ	WORD	最大线圈寄存器与离散寄存器数
MaxAI	WORD	最大输入寄存器数
MaxHold	WORD	最大保持寄存器数
Done	BOOL	配置完成标志
Error	BYTE	配置错误代码

注：modbus 数据会自 ShmStart 地址开始，依次存放 MaxIQ 个线圈寄存器、MaxIQ 个离散寄存器、MaxAI 个输入寄存器、MaxHold 个保持寄存器。

错误代码：

0 无错误
1 波特率参数错误
2 预留
3 奇偶校验参数错误
4 端口未使能
6 端口配置失败
7 主机连接失败
8 端口物理通讯模式配置失败（默认固定为 485 模式，）
9 端口被占用
10 寄存器使用超限
11 modbus 地址分配失败
12 ShmStart 映射地址不在 M3 共享缓冲区
13 站地址配置失败

3.2 MBUS_RTU_SLAVE 功能块



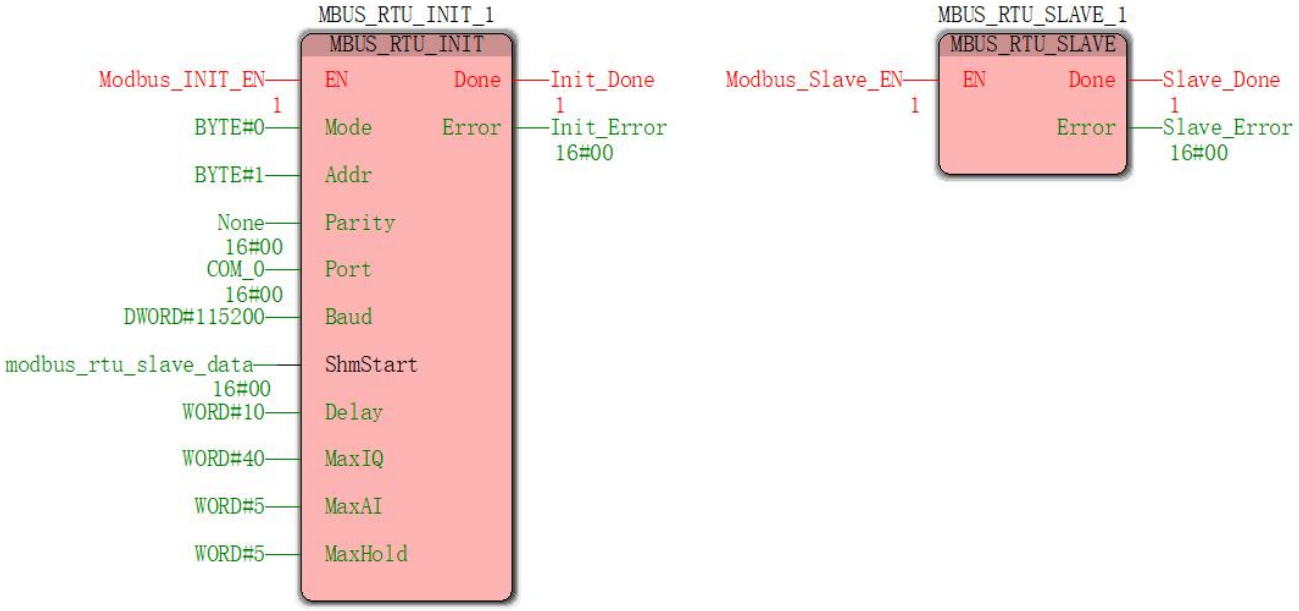
功能：modbus rtu 从站启动功能块，用于开启 modbus rtu 功能。

参数	数据类型	描述
EN	BOOL	modbus rtu 从站通讯使能，用来开启从站通讯
Done	BOOL	配置完成标志
Error	BYTE	配置错误代码

错误代码：

0 无错误
1 接收数据失败
2 返回数据失败

3.3 应用举例



使用 MBUS_RTU_INIT 将本机设置为 modbus 从站，从站地址为 1，使用外部端口为端口 0，波特率 115200、无校验，modbus 数据自本地%MX3.0.0 数据存放，开启 40 个线圈寄存器，40 个离散寄存器，5 个输入寄存器，5 个保持寄存器。

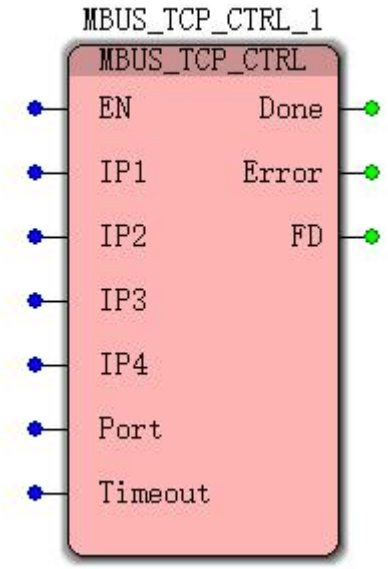
在线调试，首先使能 MBUS_RTU_INIT_1 功能块，输出无误，则配置成功，再使能 MBUS_RTU_SlAVE 功能块，即开启 modbus 通讯，等待 modbus rtu 主站连接，连接成功后 Slave_Done 置为 True，之后正常数据传输即可。

modbus 数据与本地地址映射（根据上述配置）

modbus 寄存器类型	modbus 寄存器地址		modbus 寄存器个数	本地地址（M3 区）
	PLC 地址	莫迪康地址		
线圈寄存器	1-40	0-39	40	%MX3.0.0
离散寄存器	1-40	0-39	40	%MX3.5.0
输入寄存器	1-5	0-4	5	%MB3.10
保持寄存器	1-5	0-4	5	%MB3.20

4、MODBUS TCP 主站

4.1 MBUS_TCP_CTRL 功能块



功能：MODBUS_TCP_CTRL 用于 CPU 作为主站与 MODBUS TCP 从站通讯时的配置功能块，程序调用 MODBUS_TCP_CTRL 功能块来初始化、监视或禁用 Modbus 通信。

在执行 MODBUS_TCP_MSG 功能块前，程序必须先执行 MODBUS_TCP_CTRL 且不出现错误。该功能块完成后，将“完成”(Done) 位置为 ON，然后再继续执行下一条指令。

EN 输入接通时，在每次扫描时均执行该指令。

MODBUS_TCP_CTRL 功能块参数介绍

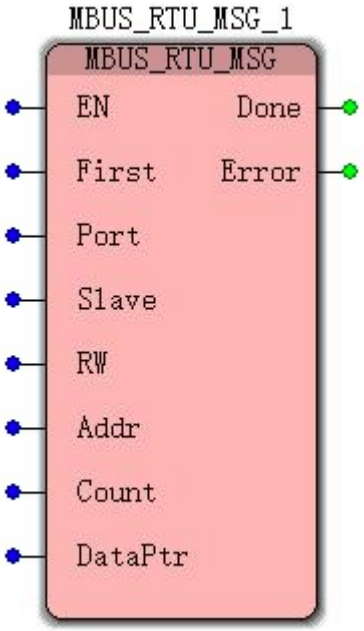
参数	数据类型	描述
EN	BOOL	用于开启 modbus tcp 配置，modbus tcp 功能使用期间需一直使能该变量
IP1-IP4	BYTE	从站地址 IP,BYTE 类型，例如 IP1=192,IP2=168,IP3=0,IP4=100 表示从站地址为 192.168.0.100；
Port	DWORD	从站端口号，DWORD 类型，默认值 502，具体数值要参考从站设备的端口号；
Timeout	WORD	超时时间，等待从站做出响应的毫秒数，WORD 类型，“超时”(Timeout) 值可以设置为 1 ms 到 65535 ms 之间的任何值。典型值是 1000 ms ，“超时”(Timeout) 参数无需设置过大，因为 Modbus Tcp 通讯的通讯速率足够块。“超时”(Timeout) 参数用于确定 Modbus 从站设备是否对请求做出响应。

Done	BOOL	设置完成标志，BOOL 类型，当设置完成并且没有错误后该标志自动置为“真”TRUE。
Error	BYTE	错误代码，BYTE 类型，当设置完成并且没有错误时该变量为 0，否则将作为错误代码。
FD	DWORD	PLC 连接标识号，DWORD 类型，每一个 MODBUS_TCP_CTRL 会产生唯一的一个标识号，是 MODBUS_TCP_MSG 功能块和 MODBUS_TCP_CTRL 建立关联的唯一标识，多个 MODBUS_TCP_CTRL 产生的 FD 值不同，多个 MODBUS_TCP_MSG 可使用同一个 FD 值。

MODBUS_TCP_CTRL 执行错误代码如下：

0 无错误
1 IP 地址参数错误（IP 地址全为 0，或者全为 255）
2 Port 端口参数错误
3 超时时间参数错误
4 TCP 连接失败
5 MODBUS 通讯内存分配失败
6 功能块使用个数超限（默认最大 16 个）

4.2 MBUS_TCP_MSG 功能块



功能：程序调用 MBUS_MSG 指令，启动对 Modbus 从站的请求并处理响应。

EN 输入和 First 输入同时接通时，MBUS_MSG 指令会向 Modbus 从站发起主站请求。发送请求、等待响应和处理响应通常需要多个 PLC 扫描时间。EN 输入必须接通才能启用发送请求，并且必须保持接通

状态，直到指令为 Done 位返回接通。

MODBUS_TCP_MSG 功能块参数介绍

参数	数据类型	描述
EN	BOOL	使能控制，BOOL 类型，在启动对 Modbus 从站的请求并处理响应时，EN 输入必须一直为“真”TRUE，否则将不执行对从站的访问。
First	BOOL	通讯请求，BOOL 类型，有新请求要发送时，将参数 First 设置为接通，并仅保持一个扫描周期。First 输入以脉冲方式通过边沿检测元素（例如，上升沿），这将导致程序发送请求一次，如果 EN 为“假”FALSE，则不检测 First 状态，有关详细信息，请参见示例程序。
FD	DWORD	PLC 连接标识号，DWORD 类型，每一个 MODBUS_TCP_MSG 对应唯一的标识号，是 MODBUS_TCP_MSG 功能块和 MODBUS_TCP_CTRL 建立关联的唯一标识。
Slave	BYTE	Modbus 从站设备地址，BYTE 类型，允许范围为 0 至 247，地址 0 是广播地址。仅将地址 0 用于写入请求。系统不会响应对地址 0 的广播请求。并非所有从站设备都支持广播地址。Truhigh P500 Modbus 从站库不支持广播地址。
RW	BOOL	读写指示，BOOL 类型，0 读取，1 写入，离散量输出（线圈）和保持寄存器支持读请求和写请求。离散量输入（触点）和输入寄存器仅支持读请求。
Addr	DWORD	从站寄存器地址，WORD 类型：1-65535（线圈寄存器）、100001-165535（离散寄存器）、300001-365535（输入寄存器）、400001-465535（保持寄存器）
Count	INT	读写从站元素个数，INT 类型，用于分配要在该请求中读取或写入的数据元素数。对于位数据类型，“Count”是位数，对于字数据类型，则表示字数，（线圈寄存器最多：1920 个、离散寄存器最多：1920 个、输入寄存器最多：120 个、保持寄存器最多：120 个）
DataPtr	Any	间接地址指针，Any 类型，指向所定义读写数据变量的首地址，对于读请求，将 DataPtr 设置为用于存储从 Modbus 从站读取的数据的第一个 CPU 存储单元。对于写请求，将 DataPtr 设置为要发送到 Modbus 从站的数据的第一个 CPU 存储单元。程序将 DataPtr 值以间接地址指针的形式传递到 MODBUS_TCP_MSG。
Done	BOOL	MODBUS_TCP_MSG 执行完成指示，BOOL 类型，程序已发送请求并接收响应后，Done 输出为 FALSE。响应完成或 MODBUS_TCP_MSG 指令因错误中止时，Done 输出为 TRUE。

Error	BYTE	错误代码，BYTE 类型，仅当 Done 输出为 TRUE 时，Error 输出才有效。

MODBUS_TCP_MSG 执行错误代码如下：

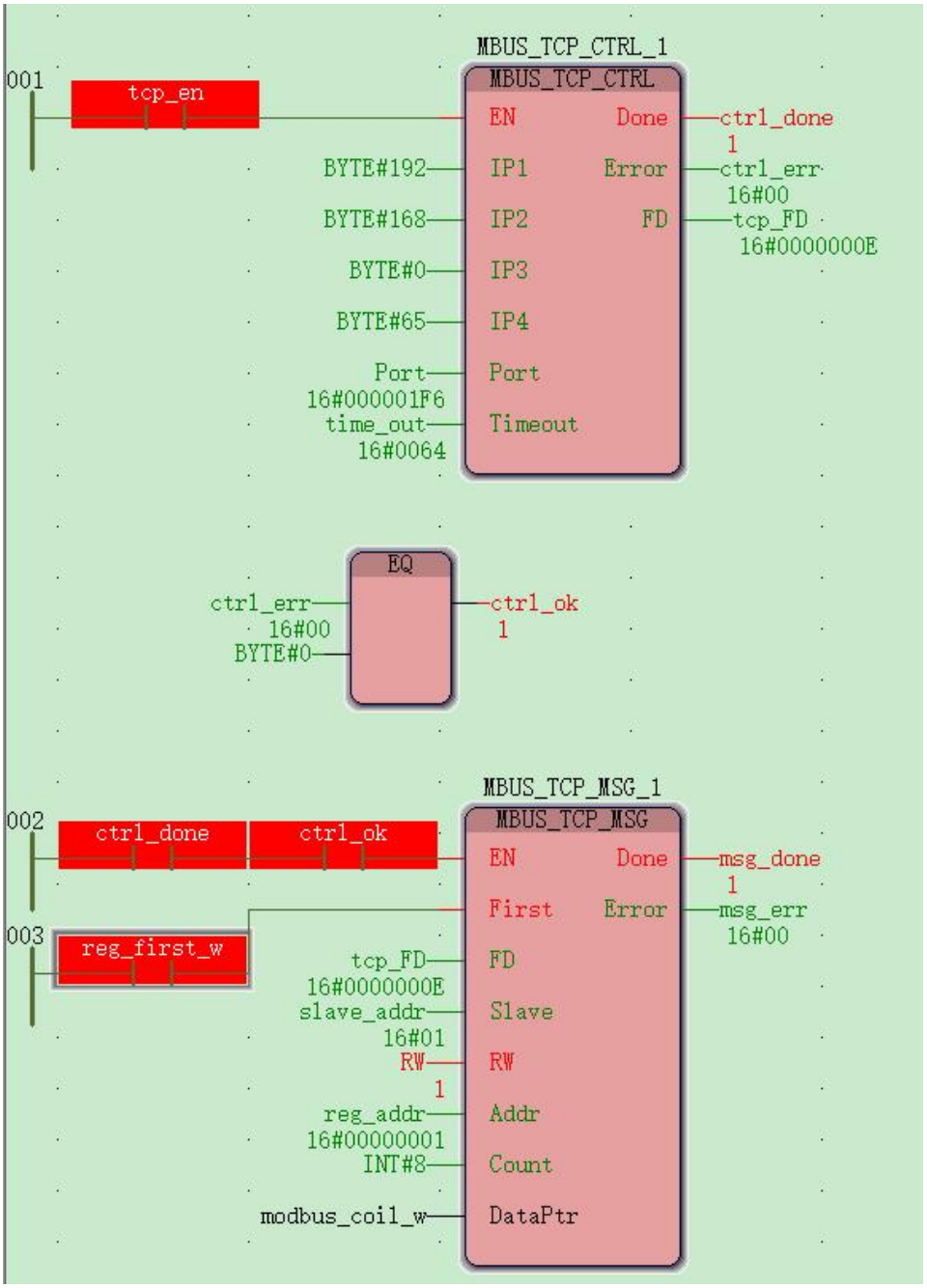
0 无错误
1 从机地址参数无效
2 通讯超时
3 FD 参数无效
4 MODBUS 寄存器地址参数错误（合法地址：1-65535、100001-165535、300001-365535、400001-465535）
5 读写位参数非法，读写可为 0 或 1，只读寄存器只能为 0
6 读写寄存器数目超限（线圈：1920 个、离散：1920 个、输入：120 个、保持：120 个）
7 MODBUS 通讯内存分配失败
8 功能块使用个数超限（默认最大 80 个）

4.3 MODBUS TCP 主站应用举例

从站地址为 192.168.0.65，访问端口号 Port 为 502（0x1F6）， 超时时间 Timeout 为 100ms（0x64）。

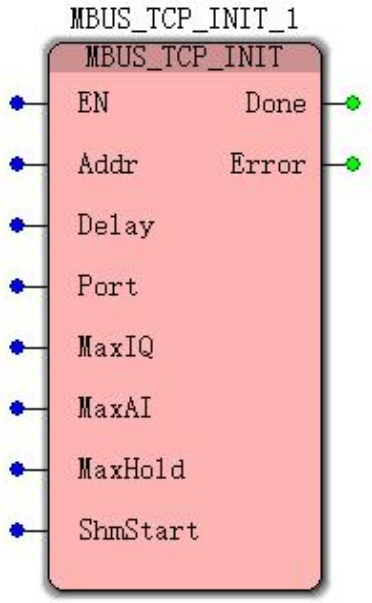
从站地址 Slave 为 1，写寄存器数量 Count 为 8，DataPtr 为 MODBUS_COIL_DATA 类型数组 modbus_coil_w。

```
TYPE
 MODBUS_COIL_DATA :ARRAY [1..100] OF BYTE;
END_TYPE
```


MBUS_TCP_CTRL_1
MBUS_TCP_CTRL
001
tcp_en
EN
Done
ctrl_done
1
BYTE#192
IP1
Error
ctrl_err
16#00
BYTE#168
IP2
FD
tcp_FD
16#0000000E
BYTE#0
IP3
BYTE#65
IP4
Port
Port
16#000001F6
time_out
Timeout
16#0064
EQ
ctrl_err
16#00
BYTE#0
ctrl_ok
1
MBUS_TCP_MSG_1
MBUS_TCP_MSG
002
ctrl_done
ctrl_ok
EN
Done
msg_done
1
First
Error
msg_err
16#00
003
reg_first_w
tcp_FD
16#0000000E
FD
slave_addr
16#01
Slave
RW
1
RW
reg_addr
16#00000001
Addr
INT#8
Count
modbus_coil_w
DataPtr

5、MODBUS TCP 从站

5.1 MBUS_TCP_INIT 功能块



功能： MODBUS_TCP_INIT 指令用于启用，初始化或禁用 Modbus 通信。在使用 MODBUS_TCP_SLAVE 指令之前，必须先无错误地执行 MODBUS_TCP_INIT。该指令完成后，立即置位“完成”(Done) 位，然后继续执行下一条指令。

EN 输入接通时，会在每次扫描时执行该指令。

MODBUS_TCP_INIT 功能块参数介绍

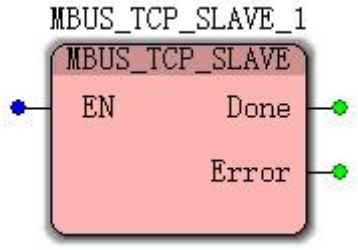
参数	数据类型	描述
EN	BOOL	使能控制，BOOL 类型，当 EN 输入接通时，在每次扫描时均执行该指令
Addr	BYTE	从站地址，BYTE 类型，站地址范围 1-247
Delay	WORD	延时响应时间，WORD 类型，通过使标准 Modbus 信息超时时间增加分配的毫秒数来延迟标准 Modbus 信息结束超时条件。在有线网络上运行时，该参数的典型值应为 0。如果使用具有纠错功能的调制解调器，则将延时设置为 50 至 100 ms 之间的值。如果使用扩频无线通信，则将延时设置为 10 至 100 ms 之间的值。“延时”(Delay) 值可以是 0 至 65535 ms。
Port	DWORD	TCP 通讯端口号，DWORD 类型，默认值为 502，作为 tcp server 通讯的端口号，具体端口号需要和主站协商确定。
MaxIQ	WORD	用于设置 Modbus 地址 0xxxx 和 1xxxx 可用的 I 和 Q 点数，WORD 类型。

		值为 0 时，将禁用所有对输入和输出的读写操作。数量限制详见寄存器地址及数量分配说明。
MaxAI	WORD	用于设置 Modbus 地址 3xxxx 可用的字输入（AI）寄存器数，WORD 类型。值为 0 时，将禁止读取模拟量输入。数量限制详见寄存器地址及数量分配说明。
MaxHold	WORD	用于设置 Modbus 地址 4xxxx 可访问的字保持寄存器数，WORD 类型。值为 0 时，将禁止读写保存寄存器。数量限制详见 5.3 寄存器地址及数量分配说明。
ShmStart	ANY	用于设置 Modbus 从站数据的起始地址，必须存放在 M3 区，合法地址为%MB3.0-%MB3.10239。
Done	BOOL	执行完成指示，BOOL 类型，MODBUS_TCP_INIT 执行完成时置位为“真”TRUE。
Error	BYTE	执行错误代码，BYTE 类型，仅当“完成”(Done) 接通时，该输出才有效。如果“完成”(Done) 关闭，则错误参数不会改变。

MODBUS_TCP_INIT 错误代码如下：

0 无错误
1 MODBUS 从站地址设置错误（站地址为 0 或者大于 247）
2 TCP Port 端口参数错误（端口为 0）
3 MODBUS 寄存器地址错误，或者寄存器分配超出 M3 区域（合法地址：%MB3.0--%MB3.10239）
4 MODBUS 通讯内存分配失败
5 TCP Server 打开失败（Port 端口被占用）
6 获取本地 IP 地址失败
7 功能块使用个数超限（默认 1 个）

5.2 MBUS_TCP_SLAVE 功能块



功能：MBUS_TCP_SLAVE 指令用于处理来自 Modbus 主站的请求，并且必须在每次扫描时执行，以便检查和响应 Modbus 请求。

EN 输入接通时，会在每次扫描时执行该指令。

MODBUS_TCP_SLAVE 功能块参数介绍

参数	数据类型	描述
EN	BOOL	使能控制，BOOL 类型，当 EN 输入接通时，每个扫描周期都会执行 MBUS_TCP_SLAVE 接收判断
Done	BOOLE	当 MBUS_TCP_SLAVE 指令响应 Modbus 请求时，“完成”(Done) 输出接通。如果未处理任何请求，“完成”(Done) 输出关闭。
Error	BYTE	执行错误代码，，BYTE 类型，仅当“完成”(Done) 接通时，该输出才有效。如果“完成”(Done) 关闭，则错误参数不会改变。

MBUS_TCP_SLAVE 错误代码如下：

0 正常
1 接收主机数据失败或者主机主动断开连接
2 向主机返回数据失败
3 MODBUS 寄存器内存申请失败
4 MODBUS 通讯任务创建失败

5.3 寄存器地址及数量分配说明

M3 区总大小为 10240 字节，0xxxx、1xxxx、3xxxx、4xxxx 分配总大小不能超过此限制，例如：

0xxxx 所占字节数：((MaxIQ + 7)/8)

1xxxx 所占字节数：((MaxIQ + 7)/8)

3xxxx 所占字节数：MaxAI*2

4xxxx 所占字节数：MaxHold*2

ShmStart 起始地址：%MB3.x

%MB3.x + ((MaxIQ + 7)/8) + ((MaxIQ + 7)/8) + MaxAI*2 + MaxHold*2 之和不能大于%MB3.10239。

各类型寄存器在 M3 区的存储顺序为先是线圈寄存器（0xxxx），然后是离散输入寄存器（1xxxx），之后是只读输入寄存器（3xxxx），最后是保存寄存器（4xxxx），例如：

MaxIQ = 50；

MaxAI = 60；

MaxHold = 60；

ShmStart = %MB3.100;

线圈寄存器（0xxxx）的起始地址为%MB3.100，长度占用(50 + 7)/8 = 7 字节；

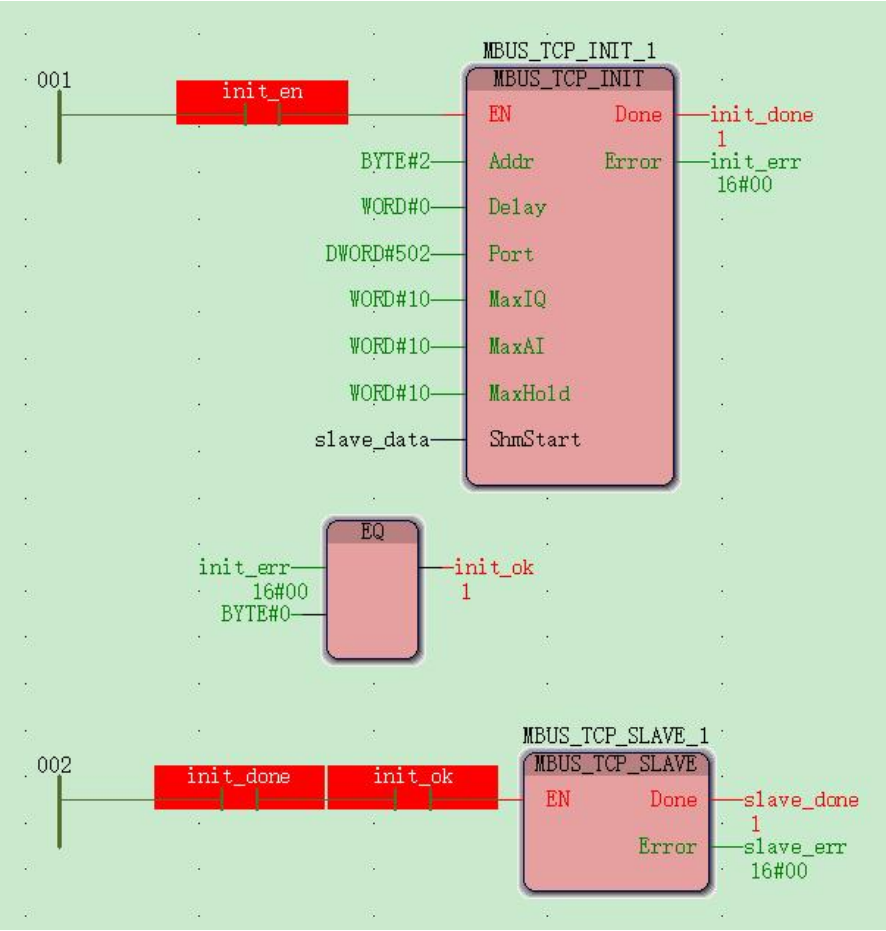
离散输入寄存器（1xxxx）的起始地址为%MB3.107，长度占用(50 + 7)/8 = 7 字节；

只读输入寄存器（3xxxx）的起始地址为%MB3.114，长度占用 60*2 = 120 字节；

保存寄存器（4xxxx）的起始地址为%MB3.234，长度占用 60*2 = 120 字节；

5.4 MODBUS TCP 从站应用举例

设置站地址 Addr 为 2，延时 Delay 为 0，端口号 Port 为 502，MaxIQ、MaxAI、MaxHold 分配数量为 10 个，M3 区 ShmStart 起始地址 slave_data 设为%MB3.0。



二、TruhighFwDev 库

1、DWORD 数据类型转换

功能：DWORD 型数据转换功能，首先需要导入 TruhighFwDev.FWL 固件库。

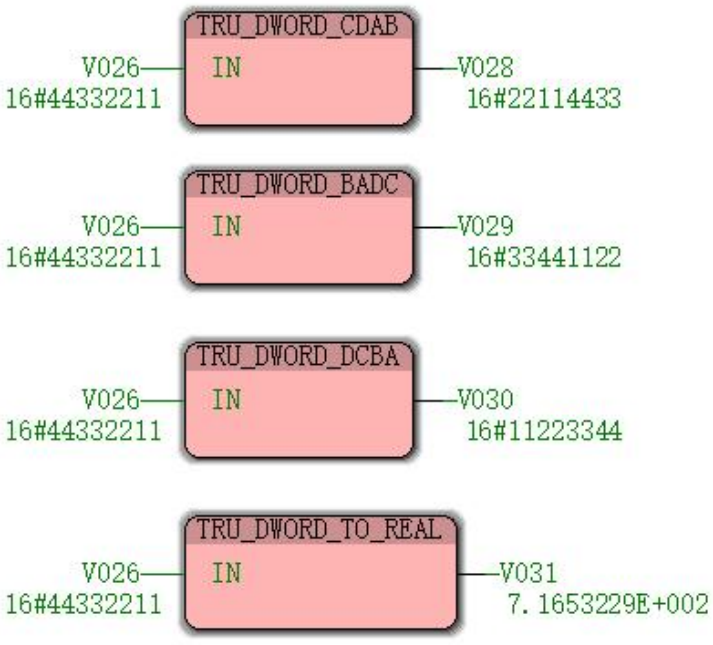
输入值取值范围 0~4,294,967,295。

指令	输入值	输出值	描述
TRU_DWORD_CDAB	DWORD	DWORD	若IN=0x12345678则OUT=0x56781234，高低字交换
TRU_DWORD_BADC	DWORD	DWORD	若IN=0x12345678则OUT=0x34127856，高低字节交换
TRU_DWORD_DCBA	DWORD	DWORD	若IN=0x12345678则OUT=0x78563412，字节倒序
TRU_DWORD_TO_REAL	DWORD	REAL	若IN=0x12345678输出为REAL类型，字节顺序不变

EN/ENO 行为：如果 EN/ENO 被激活（“选项|图形编辑器设置），只有当 EN 输入被置为 TRUE 时，才执行此功能。

如果 EN = FALSE，保留上一次执行功能的输出值。依据程序逻辑，有可能需要对 ENO 输出求值。

应用举例：V026 的高字（0x4433）和低字（0x2211）交换后输出。



2、REAL 数据类型转换

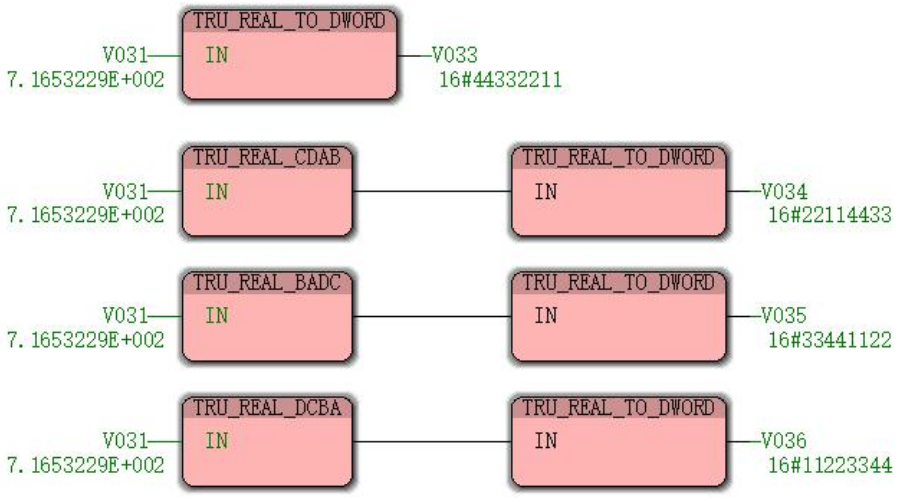
功能：REAL 数据类型转换，首先需要导入 TruhighFwDev.FWL 固件库。

指令	输入值	输出值	描述
TRU_REAL_CDAB	REAL	REAL	若IN对应的十六进制数据为0x11223344， 则OUT对应的十六进制数据为0x33441122，高低字交换
TRU_REAL_BADC	REAL	REAL	若IN对应的十六进制数据为0x11223344， 则OUT对应的十六进制数为0x22114433，高低字节交换
TRU_REAL_DCBA	REAL	REAL	若IN对应的十六进制数据为0x12345678， 则OUT对应的十六进制数据为0x78563412，字节倒序
TRU_REAL_TO_DWORD	REAL	DWORD	若IN对应的十六进制数据为0x12345678，字节顺序不变
TRU_REAL_TO_2INT	REAL	INT	若IN对应的十六进制数据为0x12345678， 则OUT1对应十六进制数据为0x1234， 则OUT2对应十六进制数据为0x5678。
TRU_2INT_TO_REAL	INT	REAL	若IN1对应十六进制数据为0x1234， IN2对应十六进制数据为0x5678， 则OUT对应的十六进制数据为0x12345678。
TRU_BYTES_TO_LREAL	BYTE	LREAL	把输入IN1-IN8八个字节对应到LREAL的八个字节，IN1在最高位，IN8在最低位。

EN/ENO 行为：如果 EN/ENO 被激活（“选项|图形编辑器设置），只有当 EN 输入被置为 TRUE 时，才执行此功能。

如果 EN = FALSE，保留上一次执行功能的输出值。依据程序逻辑，有可能需要对 ENO 输出求值。

应用举例：



3、除法运算

除法运算功能

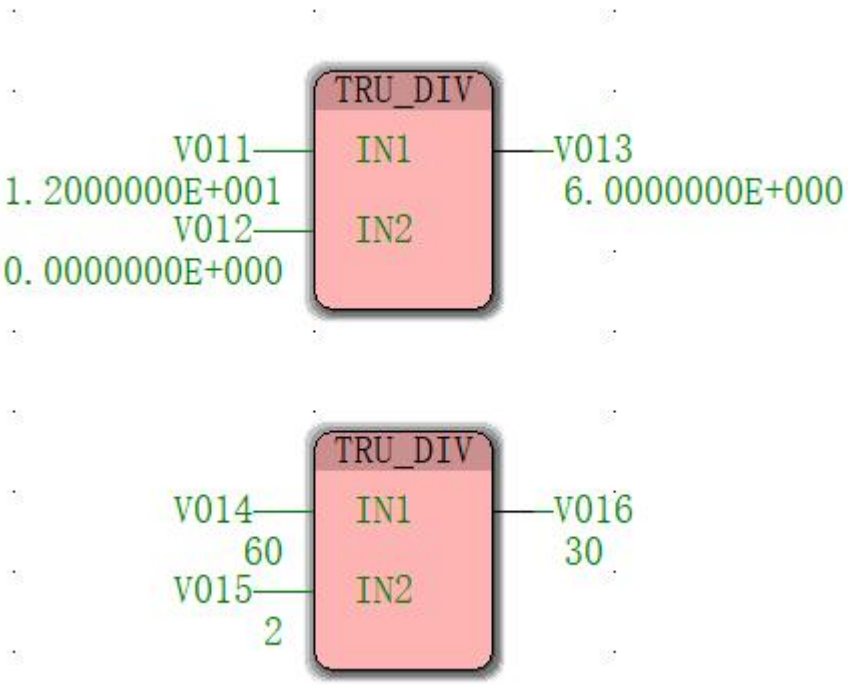
功能：DIV 除法运算处理。

指令	输入值	输出值	描述
TRU_DIV	ANY_NUM	ANY_NUM	IN2除数为0，则OUT保持不变。

EN/ENO 行为：如果 EN/ENO 被激活（“选项|图形编辑器设置），只有当 EN 输入被置为 TRUE 时，才执行此功能。

如果 EN = FALSE，保留上一次执行功能的输出值。依据程序逻辑，有可能需要对 ENO 输出求值。

应用举例：



附表一：ASCII 对照表

八进制	十六进制	十进制	字符	八进制	十六进制	十进制	字符
00	00	0	nul	100	40	64	@
01	01	1	soh	101	41	65	A
02	02	2	stx	102	42	66	B
03	03	3	etx	103	43	67	C
04	04	4	eot	104	44	68	D
05	05	5	enq	105	45	69	E
06	06	6	ack	106	46	70	F
07	07	7	bel	107	47	71	G
10	08	8	bs	110	48	72	H
11	09	9	ht	111	49	73	I
12	0a	10	nl	112	4a	74	J
13	0b	11	vt	113	4b	75	K
14	0c	12	ff	114	4c	76	L
15	0d	13	er	115	4d	77	M
16	0e	14	so	116	4e	78	N
17	0f	15	si	117	4f	79	O
20	10	16	dle	120	50	80	P
21	11	17	dc1	121	51	81	Q
22	12	18	dc2	122	52	82	R
23	13	19	dc3	123	53	83	S
24	14	20	dc4	124	54	84	T
八进制	十六进制	十进制	字符	八进制	十六进制	十进制	字符

25	15	21	nak	125	55	85	U
26	16	22	syn	126	56	86	V
27	17	23	etb	127	57	87	W
30	18	24	can	130	58	88	X
31	19	25	em	131	59	89	Y
32	1a	26	sub	132	5a	90	Z
33	1b	27	esc	133	5b	91	[
34	1c	28	fs	134	5c	92	\
35	1d	29	gs	135	5d	93	]
36	1e	30	re	136	5e	94	^
37	1f	31	us	137	5f	95	_
40	20	32	sp	140	60	96	’
41	21	33	!	141	61	97	a
42	22	34	″	142	62	98	b
43	23	35	#	143	63	99	c
44	24	36	$	144	64	100	d
45	25	37	%	145	65	101	e
46	26	38	&	146	66	102	f
47	27	39	`	147	67	103	g
50	28	40	(	150	68	104	h
51	29	41	)	151	69	105	i
52	2a	42	*	152	6a	106	j
53	2b	43	+	153	6b	107	k
八进制	**十六进制**	**十进制**	**字符**	**八进制**	**十六进制**	**十进制**	**字符**

54	2c	44	,	154	6c	108	l
55	2d	45	-	155	6d	109	m
56	2e	46	.	156	6e	110	n
57	2f	47	/	157	6f	111	o
60	30	48	0	160	70	112	p
61	31	49	1	161	71	113	q
62	32	50	2	162	72	114	r
63	33	51	3	163	73	115	s
64	34	52	4	164	74	116	t
65	35	53	5	165	75	117	u
66	36	54	6	166	76	118	v
67	37	55	7	167	77	119	w
70	38	56	8	170	78	120	x
71	39	57	9	171	79	121	y
72	3a	58	:	172	7a	122	z
73	3b	59	;	173	7b	123	{
74	3c	60	<	174	7c	124	|
75	3d	61	=	175	7d	125	}
76	3e	62	>	176	7e	126	~
77	3f	63	?	177	7f	127	del