凌臣科技 LINGCHEN

资料编码：2304LC1200-4

凌臣PLC中型PLC指令手册

目 录

指令系统概述

可编程控制系统中，使CPU完成某种操作或实现某种功能的命令及多个命令的组

合称为指令，指令的集合称为指令系统。

1.1 指令分类

指令安照实现方式的不同分为功能和功能块两类。以功能方式实现的指令（以FUN 标注），在使用的时候无需声明。 以功能块方式实现的指令（以 FB 标注），在使用的时候需声 明实例名。常用基本指令的实现方式可参考下表。

常用功能指令	算术运算指令（如ADD加法指令）
	赋值指令（MOVE赋值指令）
	逻辑运算指令（如AND与指令）
	移位指令（如SHL左移指令）
	选择指令（如MAX取最大值指令）
	比较指令（如GT大于指令）
	类型转换指令（如REAL_TO_<TYPE>实数类型转换指令）
	初等数学运算指令（如ABS绝对值指令）
	地址运算指令（如ADR取地址指令）
常用功能块指令	双稳态指令（如 SR 置位优先双稳态器）
	触发器(如 R_TRIG 上升沿检测触发器)
	计数器（如 CTU递增计数器）
	定时器（如 TON通电延时定时器）
	信号发生器指令（如 BLINK脉冲信号发生器）
	高等数学运算指令（如 DERIVATIVE微分）
	控制器指令（如 PID 比例积分微分控制器）

1.2 指令库

库是指令的集合，所有的库文件为 库“名.lib ”，例如标准库(Standard.lib)、应用库(Util.lib)等。 建立工程时有些库文件会自动加载到工程当中（如标准库(Standard.lib)及与硬件配置对应的应用库），可直接调用。而附加库文件则需要用户手动添加后才可调用。

1.3 库文件管理器

Codesys 中通过库文件管理器来管理库文件，库文件管理器窗口分为“库列表”、“指令”、“变量声明”和“指令图形”这 4 部分。

库列表　列出当前工程中添加的所有库文件名。

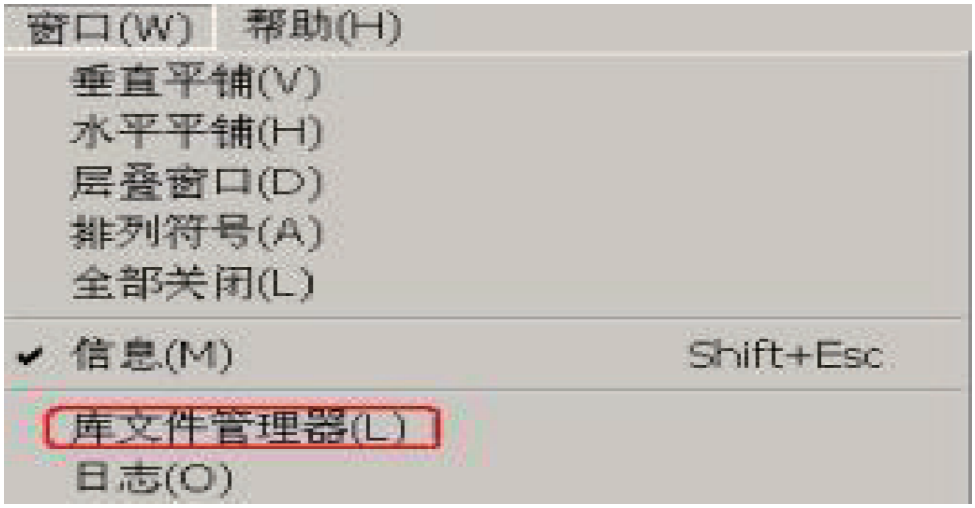
指令　列出该库文件中包含的指令并按类分别列出。

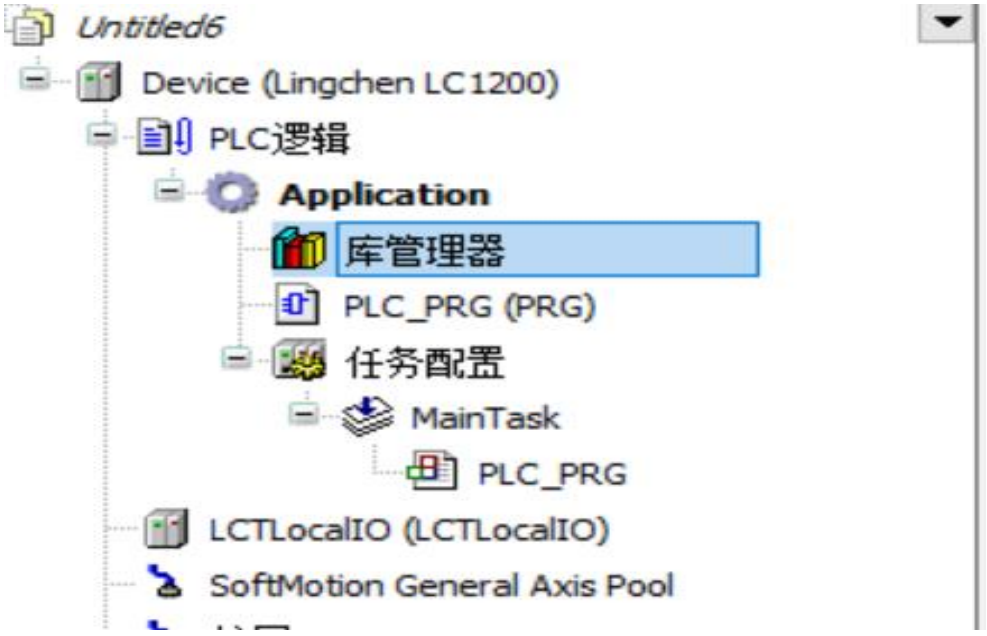
变量声明　列出所选指令的变量声明。

指令图形　显示所选指令的图形，左边为输入端，右边为输出端。

1. 打开库管理器

点击对象管理器中资源”，打开资源窗口，再双击　库“文件管理器”，如下图。



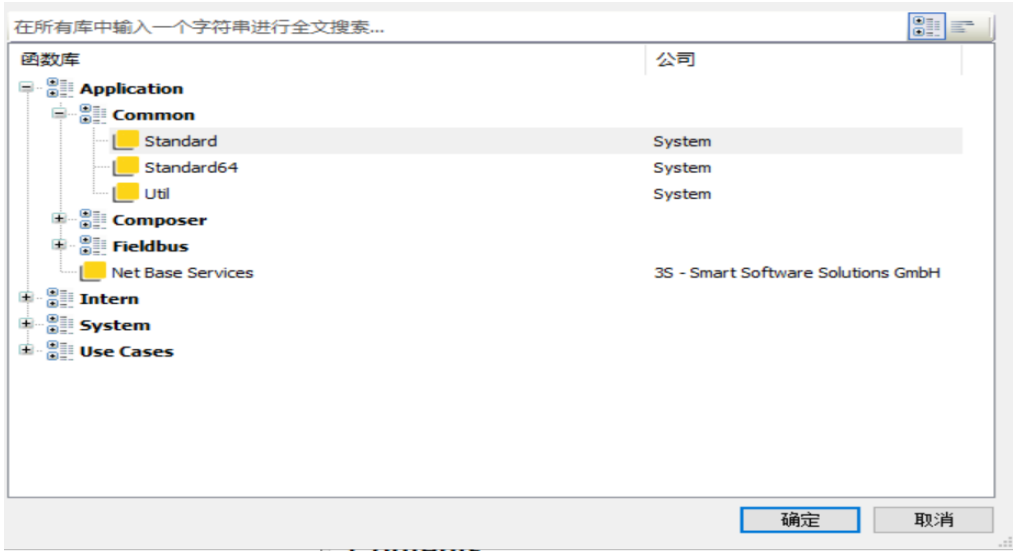


2. 指令库的添加

当库列表区的库文件已不能满足目前的编程需要时，则需要添加库。

在库文件管理器窗口的库列表位置选择鼠标右键菜单命令添加库，如下图。

选择所需要的库文件，点击确定即可打开对应的指令库。



注意：凡是添加到库文件管理器的库都会占用用户程序空间，所以建议用户只添加所使 用的库。

3. 删除库

选中库文件，选择鼠标右键菜单命令 删“ 除 ”，即可从工程和库文件管理器中删除已添加的 库。

第2章 基本指令

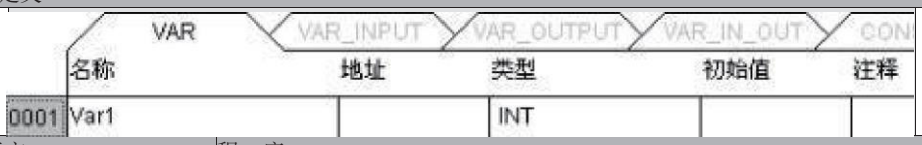
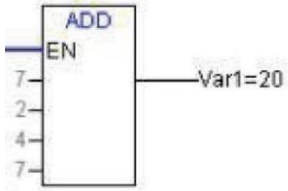
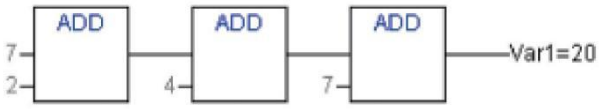
2.1 算术运算指令

2.1.1 ADD——加法指令

功能：两个（或者多个）变量或常量相加。

输入/输出数据类型： BYTE、WORD、DWORD 、SINT、USINT、INT、UINT 、DINT、 UDINT 、 REAL、 TIME。

指令使用举例

<table>
<tr><td colspan="2">变量定义
 
</td></tr>
<tr><td>编程语言</td><td>程 序</td></tr>
<tr><td>梯形图（LD）</td><td> 
</td></tr>
<tr><td>结构化文本（ST）</td><td>Var1:=7+2+4+7; （*结果Var1 为20*）</td></tr>
<tr><td>功能块（FBD）</td><td> 
</td></tr>
</table>

提示：

1.TIME型量也可以使用加法功能， 两个TIME 型量相加得到一个新的时间量。例如：t#45s + t#50s = t#1m35 s。

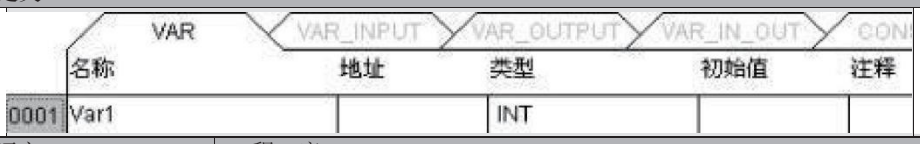
2.被选择的输出数据类型应可以存储输出结果， 否则可能引起数据错误。MUL 、SUB、DIV 指令同样。

2.1.2 MUL——乘法指令

功能：两个（或者多个） 变量或常量相乘。

输入/输出数据类型： BYTE、WORD、DWORD 、SINT、USINT、INT、UINT 、DINT、UDINT 、 REAL。

指令使用举例:

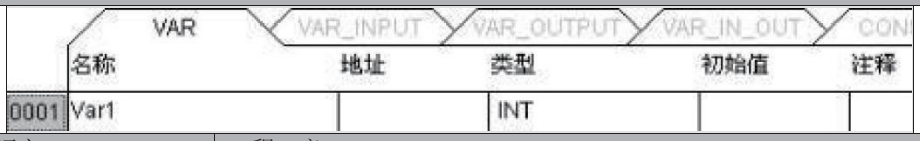
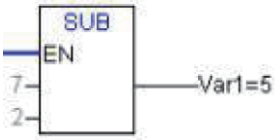
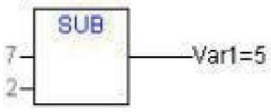
<table>
<tr><td colspan="2">变量定义
 
</td></tr>
<tr><td>编程语言</td><td>程 序</td></tr>
<tr><td>梯形图（LD）</td><td> 
</td></tr>
<tr><td>结构化文本（ST）</td><td>Var1:=7*2*4*7; （*结果 Var1 为 392*）</td></tr>
<tr><td>功能块（FBD）</td><td> 
</td></tr>
</table>

2.1.3 SUB——减法指令

功能：两个变量或常量相减。

输入/输出数据类型： BYTE、WORD、DWORD 、SINT、USINT、INT、UINT 、DINT、 UDINT 、REAL、 TIME 、 TOD。

指令使用举例

<table>
<tr><td colspan="2">变量定义
 
</td></tr>
<tr><td>编程语言</td><td>程 序</td></tr>
<tr><td>梯形图（LD）</td><td> 
</td></tr>
<tr><td>结构化文本（ST）</td><td>Var1:=7-2; （*结果 Var1 为 5*）</td></tr>
<tr><td>功能块（FBD）</td><td> 
</td></tr>
</table>

提示：

1.TIME 型量也可以使用减法功能，两个TIME 型量相减得到一个新的时间量。例如： t#1m35s - t#50s = t#45s，但时间结果不能有负值。

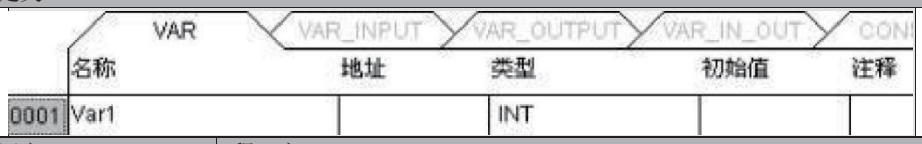
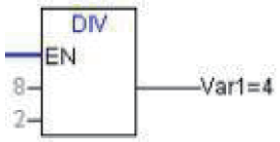
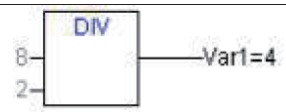
2.TOD型量也可以使用减法功能，两个TOD 型量相减得到一个新的 TIME 型数据，例如TOD#23:40:30- TOD#00:30:20=T#1390m10s0ms ,但时间结果不能有负值。

2.1.4 DIV——除法指令

功能：变量或常量相除。

输入/输出数据类型： BYTE、WORD、DWORD 、SINT、USINT、INT、UINT 、DINT、 UDINT 、 REAL。

指令使用举例

<table>
<tr><td colspan="2">变量定义
 
</td></tr>
<tr><td>编程语言</td><td>程 序</td></tr>
<tr><td>梯形图（LD）</td><td> 
</td></tr>
<tr><td>结构化文本（ST）</td><td>Var1:=8/2; （*结果 Var1 为 4*）</td></tr>
<tr><td>功能块（FBD）</td><td> 
</td></tr>
</table>

提示：

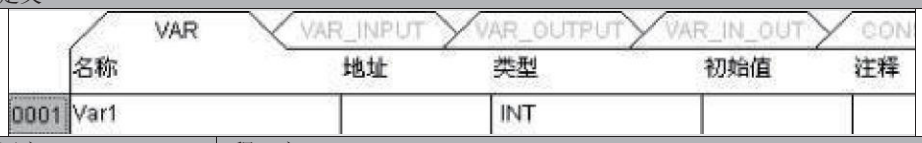
在工程中使用 DIV 指令时，可使用 CheckDivByte 、CheckDivWord 、CheckDivDWord 和 CheckDivReal 等指令（见 4.15 节） 检查除数是否为零， 避免了除数为零的现象。

2.1.5 MOD——取余指令

功能：变量或常量相除取余，是一个整数。

输入/输出数据类型： BYTE、WORD、DWORD 、SINT、USINT、INT、UINT 、DINT、 UDINT。

指令使用举例

<table>
<tr><td colspan="2">变量定义
 
</td></tr>
<tr><td>编程语言</td><td>程 序</td></tr>
<tr><td>梯形图（LD）</td><td> 
</td></tr>
<tr><td>结构化文本（ST）</td><td>Var1:=9 MOD 2; （*结果 Var1 为 1*）</td></tr>
<tr><td>功能块（FBD）</td><td> 
</td></tr>
</table>

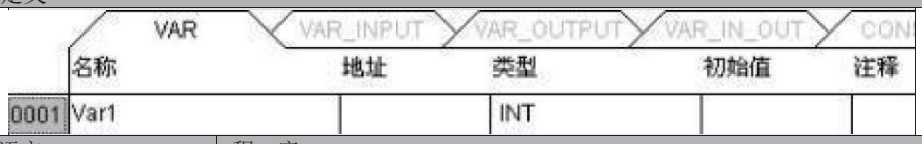
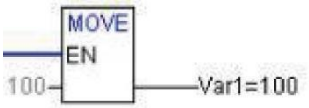
2.2 赋值指令

MOVE—赋值指令

功能：将一个常量或者变量的值赋给另外一个变量。

输入/输出数据类型： BYTE、WORD、DWORD、SINT、USINT、INT、UINT、DINT、 UDINT 、 REAL、 TIME 、 DT 、 BOOL、 STRING、 ARRAY。

指令使用举例

<table>
<tr><td colspan="2">变量定义
 
</td></tr>
<tr><td>编程语言</td><td>程 序</td></tr>
<tr><td>梯形图（LD）</td><td> 
</td></tr>
<tr><td>结构化文本（ST）</td><td>Var1:= 100; （ *结果 Var1 为 100*）</td></tr>
<tr><td>功能块（FBD）</td><td> 
</td></tr>
</table>

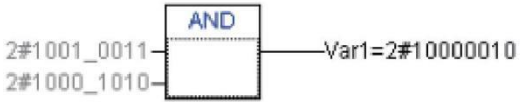
2.3 逻辑运算指令

2.3.1 AND——与指令

功能：变量或常量的相与运算。

输入/输出数据类型：BOOL、BYTE、WORD 和 DWORD。

指令使用举例

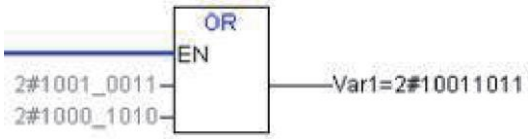
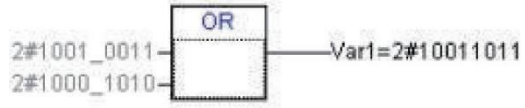
<table>
<tr><td colspan="2">变量定义
 
</td></tr>
<tr><td>编程语言</td><td>程 序</td></tr>
<tr><td>梯形图（LD）</td><td> 
</td></tr>
<tr><td>结构化文本（ST）</td><td>Var1:=2#1001_0011 AND 2#1000_ 1010;
（*结果 Var1 为 2#10000010*）</td></tr>
<tr><td>功能块（FBD）</td><td> 
</td></tr>
</table>

2.3.2 OR——或指令

功能：变量或常量的相或运算。

输入/输出数据类型：BOOL、BYTE、WORD 和 DWORD。

指令使用举例

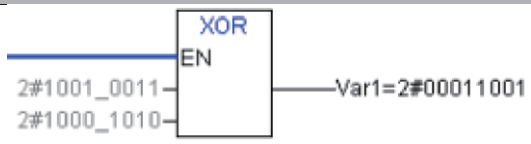
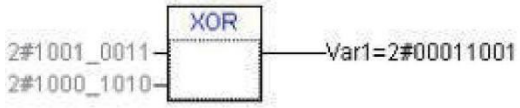
<table>
<tr><td colspan="2">变量定义
 
</td></tr>
<tr><td>编程语言</td><td>程 序</td></tr>
<tr><td>梯形图（LD）</td><td> 
</td></tr>
<tr><td>结构化文本（ST）</td><td>Var1:=2#1001_0011 OR 2#1000_ 1010 ;
（*结果 Var1 为 2#10011011*）</td></tr>
<tr><td>功能块（FBD）</td><td> 
</td></tr>
</table>

2.3.3 XOR——异或指令

功能：变量或常量的异或运算。

输入/输出数据类型：BOOL、BYTE、WORD 和 DWORD。

指令使用举例

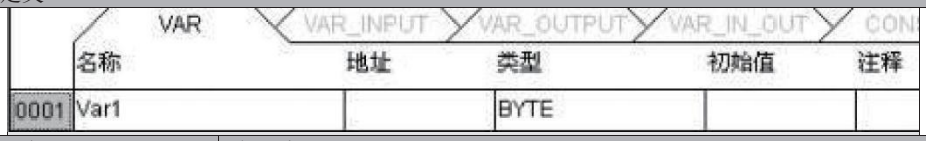
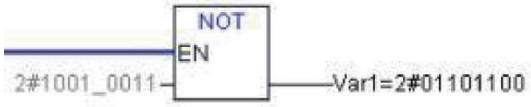
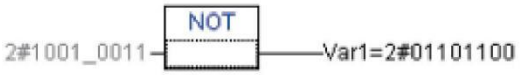
<table>
<tr><td colspan="2">变量定义
 
</td></tr>
<tr><td>编程语言</td><td>程 序</td></tr>
<tr><td>梯形图（LD）</td><td> 
</td></tr>
<tr><td>结构化文本（ST）</td><td>Var1:=2#1001_0011 XOR 2#1000_ 1010 ;
（*结果 Var1 为 2#00011001*）</td></tr>
<tr><td>功能块（FBD）</td><td> 
</td></tr>
</table>

2.3.4 NOT——取非指令

功能：变量或常量的取非运算， 逐位取非。

输入/输出数据类型：BOOL、BYTE、WORD 和 DWORD。

指令使用举例

<table>
<tr><td colspan="2">变量定义
 
</td></tr>
<tr><td>编程语言</td><td>程 序</td></tr>
<tr><td>梯形图（LD）</td><td> 
</td></tr>
<tr><td>结构化文本（ST）</td><td>Var1:= NOT 2#1001_0011 ;
（*结果 Var1 为 2#01101100*）</td></tr>
<tr><td>功能块（FBD）</td><td> 
</td></tr>
</table>

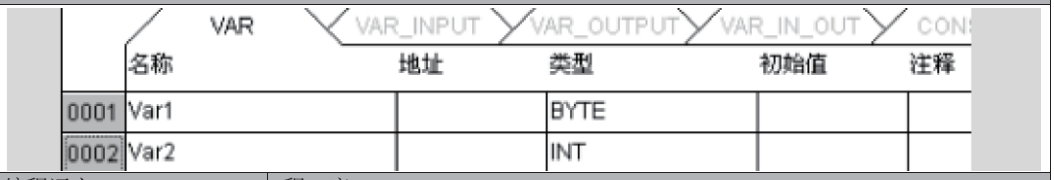
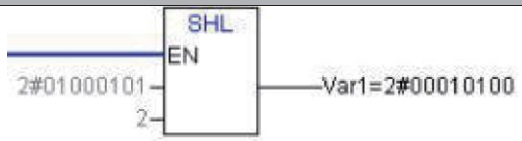
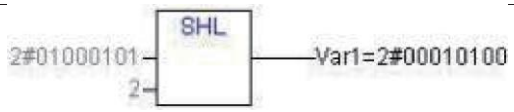
2.4 移位指令

2.4.1 SHL——左移指令

功能：对操作数进行按位左移， 左边移出的位不作处理，右边自动补0。

输入/输出数据类型：BYTE 、INT、WORD 、DWORD 、SINT 、UINT。

指令使用举例

<table>
<tr><td colspan="2">变量定义
 
</td></tr>
<tr><td>编程语言</td><td>程 序</td></tr>
<tr><td>梯形图（LD）</td><td> 
</td></tr>
<tr><td>结构化文本（ST）</td><td>Var1:=SHL(16#45,2); （*结果 Var1 为 16#14*）
Var2:=SHL(16#45,2); （*结果 Var2 为 16#0114*）
注意： 上面例子中， 虽然输入变量的值一样， 但因为输出数据类型的大小不同， 所以得到的结果 Var1 和 Var2 不同。</td></tr>
<tr><td>功能块（FBD）</td><td> 
</td></tr>
</table>

2.4.2 SHR——右移指令

功能：对操作数进行按位右移， 右边移出的位不作处理，左边自动补0。

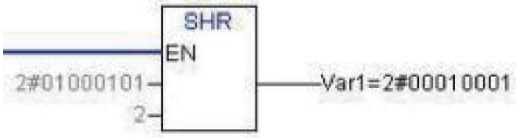
输入/输出数据类型：BYTE 、INT、WORD 、DWORD 、SINT 、UINT。

指令使用举例

变量定义

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CON

	名称	地址	类型	初始值	注释
0001	Var1		BYTE		
0002	Var2		INT		

编程语言	程　序
梯形图（**LD**）	
结构化文本（**ST**）	Var1:=SHR(16#45,2);　（*结果 Var1 为 16#11*） Var2:=SHR(16#45,2);　（*结果 Var2 为 16#0011*）
功能块（**FBD**）	

2.4.3　ROL——循环左移指令

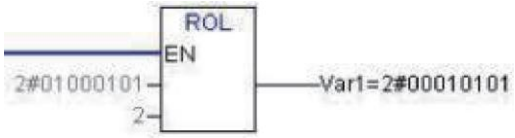
功能：对操作数进行按位循环左移，左边移出的位直接补充到右边最低位。输入输出数据类型：BYTE 、INT、WORD 、DWORD 、SINT 、UINT。

指令使用举例

变量定义

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CON

	名称	地址	类型	初始值	注释
0001	Var1		BYTE		
0002	Var2		INT		

编程语言	程　序
梯形图（**LD**）	
结构化文本（**ST**）	Var1:=ROL(16#45,2);　（*结果 Var1 为 16#15*） Var2:= ROL (16#45,2);　（ *结果 Var2 为 16#0114*） 注意：在循环左移过程中， 虽然输入变量的值一样，但因为输出数据类型的大小不同，所以得到的结果 Var1 和 Var2 不同。
功能块（**FBD**）	

2.4.4 ROR——循环右移指令

功能：对操作数进行按位循环右移，右边移出的位直接补充到左边最高位。 输入/输出数据类型：BYTE 、INT、WORD 、DWORD 、SINT 、UINT 。

指令使用举例

变量定义

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CONS

	名称	地址	类型	初始值	注释
0001	Var1		BYTE		
0002	Var2		INT		

编程语言	程 序
梯形图（**LD**）	
结构化文本（**ST**）	Var1:=ROR(16#45,2) （*结果 Var1 为 16#51*） Var2:= ROR (16#45,2) （*结果 Var2 为 16#4011*） 注意：在循环右移过程中， 虽然输入变量的值一样，但因为输出数据类型的大小不同， 所以得到的结果 Var1 和 Var2 不同。
功能块（**FBD**）	

2.5 选择指令

所有的选择指令在执行时均可以带有变量。为了能够更加清楚地说明问题， 以下各例只使 用常量。被选择的输入数据类型存储长度应不大于输出类型存储长度。

2.5.1 SEL—二选一指令

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

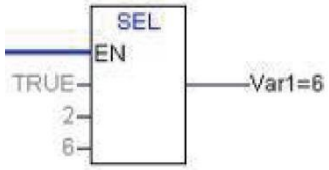
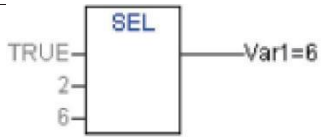
指令格式：OUT := SEL(G, IN0, IN1)，其中 G 为选择开关，IN0 和 IN1 分 别为第一个输入数据和第二个输入数据。输入/输出数据类型：G 必须是 BOOL 类型，IN0、IN1 和输出数据可以是任意数据类

指令使用举例

变量定义

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CON

	名称	地址	类型	初始值	注释
0001	Var1		INT		
0002	Var2		INT		

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

2.5.2　MAX——取最大值指令

功能：在两个输入数据中选择最大值作为输出。

指令格式：OUT:=MAX(IN0, IN1)，其中 IN0 和 IN1 分别为第 1 个输入数据和第 2 个

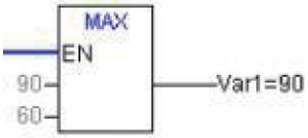
输入数据，OUT 是输出数据。输入/输出数据类型：IN0, IN1 和 OUT 可以是任意数据类型。

指令使用举例

变量定义

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CON

	名称	地址	类型	初始值	注释
0001	Var1		INT		
0002	Var2		INT		

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

2.5.3 MIN——取最小值指令

功能：在两个输入数据中选择最小值作为输出。

指令格式：OUT:=MIN(IN0, IN1)，其中 IN0 和 IN1 分别为第 1 个输入数据和第 2 个

输入数据，OUT 是输出数据。

输入/输出数据类型：IN0 ，IN1 和 OUT 可以是任意数据类型。

指令使用举例：

变量定义

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CON

	名称	地址	类型	初始值	注释
0001	Var1		INT		
0002	Var2		INT		

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

2.5.4 LIMIT——极限值指令

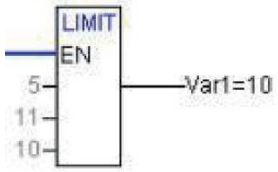
功能：判断输入数据是否在最小值和最大值之间，若输入数据在二者之间，则直接把 输入数据作为输出数据进行输出。若输入数据大于最大值，则把最大值作为输出值。

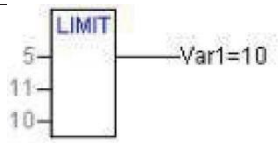
输入数据小于最小值，则把最小值作为输出值。

指令格式：OUT := LIMIT(Min, IN, Max)

输入/输出数据类型：IN 和 OUT 可以是任意数据类型。

指令使用举例

<table>
<tr><td colspan="2">变量定义
 
</td></tr>
<tr><td>编程语言</td><td>程　序</td></tr>
<tr><td>梯形图（LD）</td><td> 

（*11 为输入数据， 5 为最小值， 10 为最大值*）</td></tr>
<tr><td>结构化文本（ST）</td><td>Var1:=LIMIT(30,90,80);　(*结果 Var1 为 80 *)</td></tr>
<tr><td>功能块（FBD）</td><td> 

（*11 为输入数据， 5 为最小值， 10 为最大值*）</td></tr>
</table>

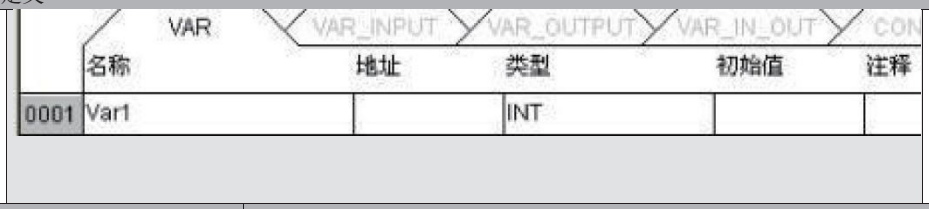
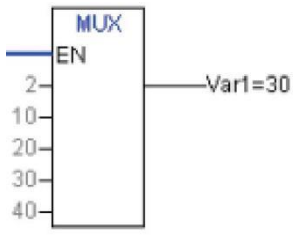
2.5.5　MUX—— 多选一指令

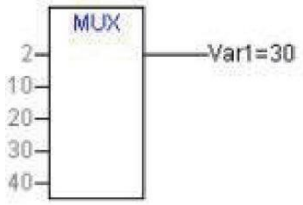
功能：通过控制数在多个输入数据中选择一个作为输出。

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

输入/输出数据类型：IN0, …, INn 和 OUT 可以是任意数据类型，K 必须是 BYTE 、 WORD 、DWORD 、 SINT 、USINT 、INT 、UINT 、DINT 或 UDINT。

指令使用举例

<table>
<tr><td colspan="2">变量定义
 
</td></tr>
<tr><td>编程语言</td><td>程　序</td></tr>
<tr><td>梯形图（LD）</td><td> 
</td></tr>
<tr><td>结构化文本（ST）</td><td>Var1:=MUX(0,30,40,50,60,70,80);（*结果 Var1 为 30*）</td></tr>
</table>

功能块（**FBD**）	 （*2 为控制数据，对应于 30，所以结果为 30*）

2.6 比较指令

所有的比较指令在执行时均可以带有变量。为了能够更加清楚地说明问题， 以下各例只使用常量。

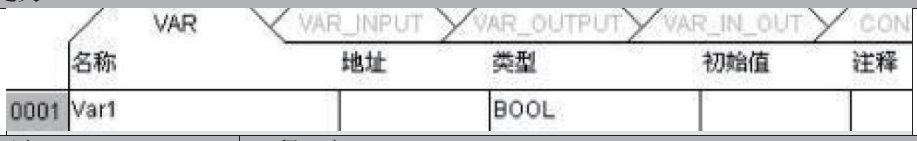
2.6.1 GT——大于指令

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

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

输出数据类型： BOOL。

指令使用举例

变量定义	
	
编程语言	程　序
梯形图（LD）	 （*结果 Var1 为 FALSE*）
结构化文本（ST）	Var1:=20>30;
功能块（**FBD**）	 （*结果 Var1 为 FALSE*）

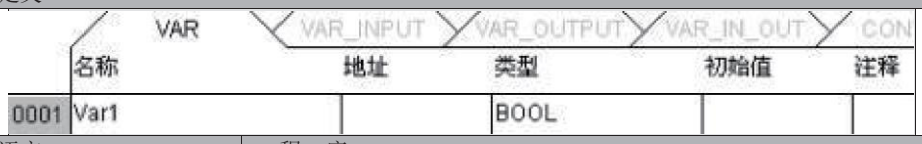
2.6.2 LT——小于指令

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

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

输出数据类型： BOOL。

指令使用举例

<table>
<tr><td colspan="2">变量定义</td></tr>
<tr><td colspan="2"> 
</td></tr>
<tr><td>编程语言</td><td>程 序</td></tr>
<tr><td>梯形图（LD）</td><td> 

（*结果 Var1 为 TRUE*）</td></tr>
<tr><td>结构化文本（ST）</td><td>VAR1:=20<30; （*结果 Var1 为 TRUE*）</td></tr>
<tr><td>功能块（FBD）</td><td> 

（*结果 Var1 为 TRUE*）</td></tr>
</table>

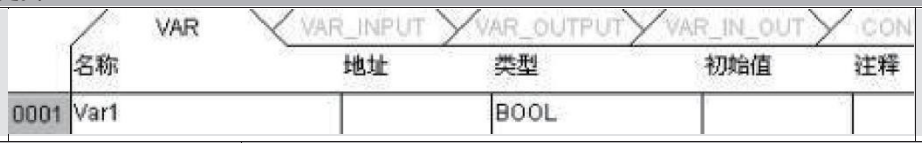
2.6.3 GE——大于等于指令

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

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

输出数据类型：BOOL。

指令使用举例

<table>
<tr><td colspan="2">变量定义</td></tr>
<tr><td colspan="2"> 
</td></tr>
<tr><td>编程语言</td><td>程 序</td></tr>
<tr><td>梯形图（LD）</td><td> 

（*结果 Var1 为 TRUE*）</td></tr>
<tr><td>结构化文本（ST）</td><td>VAR1:=60>=40; （*结果 Var1 为 TRUE*）</td></tr>
</table>

功能块（**FBD**）	 （*结果 Var1 为 TRUE*）

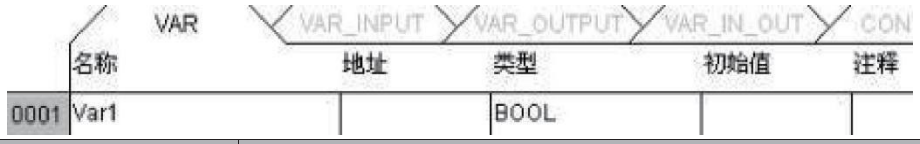
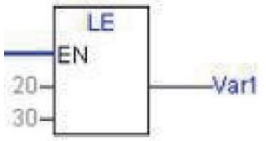
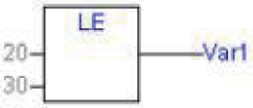
2.6.4 LE——小于等于指令

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

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

输出数据类型：BOOL。

指令使用举例

变量定义	
	
编程语言	程　序
梯形图（**LD**）	 （*结果 Var1 为 TRUE*）
结构化文本（**ST**）	VAR1:=20<=30; （*结果 Var1 为 TRUE*）
功能块（**FBD**）	 （*结果 Var1 为 TRUE*）

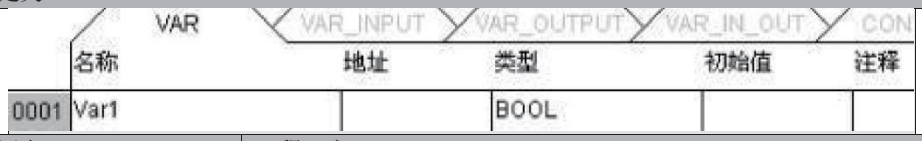
2.6.5 EQ——等于指令

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

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

输出数据类型：BOOL。

指令使用举例

<table>
<tr><td colspan="2">变量定义
 
</td></tr>
<tr><td>编程语言</td><td>程　序</td></tr>
<tr><td>梯形图（LD）</td><td> 

（*结果 Var1 为 TRUE*）</td></tr>
<tr><td>结构化文本（ST）</td><td>VAR1:=40=40;　　（*结果 Var1 为 TRUE*）</td></tr>
<tr><td>功能块（FBD）</td><td> 

（*结果 Var1 为 TRUE*）</td></tr>
</table>

2.6.6　NE——不等于指令

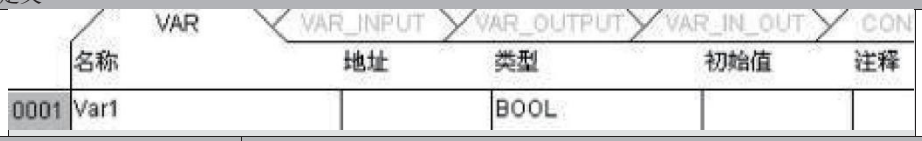
功能：判断两个操作数是否不相等，当第一个数不等于第二个数时返回 TRUE，否则结果为 FALSE。

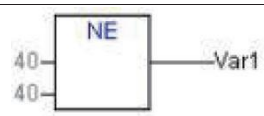
输入/输出数据类型：

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

输出数据类型：BOOL

指令使用举例

<table>
<tr><td colspan="2">变量定义
 
</td></tr>
<tr><td>编程语言</td><td>程　序</td></tr>
<tr><td>梯形图（LD）</td><td> 

（*结果 Var1 为 FALSE*）</td></tr>
<tr><td>结构化文本（ST）</td><td>VAR1:=40<>40;　　（*结果 Var1 为 FALSE*）</td></tr>
<tr><td>功能块（FBD）</td><td> 

（*结果 Var1 为 FALSE*）</td></tr>
</table>

2.7 数据类型转换指令

PowerPro 提供了 240 个数据类型转换指令，用于各种数据类型之间相互转换。

语法： <TYPE1>_TO_<TYPE2>

禁止将 “较大的”数据类型隐含地转换为 “较小的”数据类型使用，当从较大数据类 型转为较小数据类型时， 有可能丢失信息。

如果被转换的值超出目标数据类型的存储范围，则这个数的高字节将被忽略。例如将 INT 类型转换为 BYTE 类型，或者将 DINT 类型转换为 WORD 类型。

<TYPE>_TO_STRING 的转换中，字符串是从左边开始生成的。如果定义的字符串长 度小于<TYPE>的长度，右边部分会被截去。

表 4-7- 1 列出了所有的数据类型转换指令，本小节讲述数据类型转换指令。

BOOL_TO_<TYPE>	BYTE_TO_<TYPE>	DATE_TO_<TYPE>	DINT_TO_<TYPE>
BOOL_TO_BYTE	BYTE_TO_BOOL	DATE_TO_BOOL	DINT_TO_BOOL
BOOL_TO_DATE	BYTE_TO_DATE	DATE_TO_BYTE	DINT_TO_BYTE
BOOL_TO_DINT	BYTE_TO_DINT	DATE_TO_DINT	DINT_TO_DATE
BOOL_TO_DT	BYTE_TO_DT	DATE_TO_DT	DINT_TO_DT
BOOL_TO_DWORD	BYTE_TO_DWORD	DATE_TO_DWORD	DINT_TO_DWORD
BOOL_TO_INT	BYTE_TO_INT	DATE_TO_INT	DINT_TO_INT
BOOL_TO_REAL	BYTE_TO_REAL	DATE_TO_REAL	DINT_TO_REAL
BOOL_TO_SINT	BYTE_TO_SINT	DATE_TO_SINT	DINT_TO_SINT
BOOL_TO_STRING	BYTE_TO_STRING	DATE_TO_STRING	DINT_TO_STRING
BOOL_TO_TIME	BYTE_TO_TIME	DATE_TO_TIME	DINT_TO_TIME
BOOL_TO_TOD	BYTE_TO_TOD	DATE_TO_TOD	DINT_TO_TOD
BOOL_TO_UDINT	BYTE_TO_UDINT	DATE_TO_UDINT	DINT_TO_UDINT
BOOL_TO_UINT	BYTE_TO_UINT	DATE_TO_UINT	DINT_TO_UINT
BOOL_TO_USINT	BYTE_TO_USINT	DATE_TO_USINT	DINT_TO_USINT
BOOL_TO_WORD	BYTE_TO_WORD	DATE_TO_WORD	DINT_TO_WORD
DT_TO_<TYPE>	**DWORD_TO_<TYPE>**	**INT_TO_<TYPE>**	**WORD_TO_<TYPE>**
DT_TO_BOOL	DWORD_TO_BOOL	INT_TO_BOOL	WORD_TO_BOOL
DT_TO_BYTE	DWORD_TO_BYTE	INT_TO_BYTE	WORD_TO_BYTE
DT_TO_DATE	DWORD_TO_DATE	INT_TO_DATE	WORD_TO_DATE
DT_TO_DINT	DWORD_TO_DINT	INT_TO_DINT	WORD_TO_DINT
DT_TO_DWORD	DWORD_TO_DT	INT_TO_DT	WORD_TO_DT
DT_TO_INT	DWORD_TO_INT	INT_TO_DWORD	WORD_TO_DWORD
DT_TO_REAL	DWORD_TO_REAL	INT_TO_REAL	WORD_TO_INT
DT_TO_SINT	DWORD_TO_SINT	INT_TO_SINT	WORD_TO_REAL
DT_TO_STRING	DWORD_TO_STRING	INT_TO_STRING	WORD_TO_SINT
DT_TO_TIME	DWORD_TO_TIME	INT_TO_TIME	WORD_TO_STRING
DT_TO_TOD	DWORD_TO_TOD	INT_TO_TOD	WORD_TO_TIME
DT_TO_UDINT	DWORD_TO_UDINT	INT_TO_UDINT	WORD_TO_TOD
DT_TO_UINT	DWORD_TO_UINT	INT_TO_UINT	WORD_TO_UDINT
DT_TO_USINT	DWORD_TO_USINT	INT_TO_USINT	WORD_TO_UINT
DT_TO_WORD	DWORD_TO_WORD	INT_TO_WORD	WORD_TO_USINT
REAL_TO_<TYPE>	**SINT_TO_<TYPE>**	**STRING_TO_<TYPE>**	**TIME_TO_<TYPE>**
REAL_TO_BOOL	SINT_TO_BOOL	STRING_TO_BOOL	TIME_TO_BOOL
REAL_TO_BYTE	SINT_TO_BYTE	STRING_TO_BYTE	TIME_TO_BYTE
REAL_TO_DATE	SINT_TO_DATE	STRING_TO_DATE	TIME_TO_DATE
REAL_TO_DINT	SINT_TO_DINT	STRING_TO_DINT	TIME_TO_DINT
REAL_TO_DT	SINT_TO_DT	STRING_TO_DT	TIME_TO_DT
REAL_TO_DWORD	SINT_TO_DWORD	STRING_TO_DWORD	TIME_TO_DWORD
REAL_TO_INT	SINT_TO_INT	STRING_TO_INT	TIME_TO_INT
REAL_TO_SINT	SINT_TO_REAL	STRING_TO_REAL	TIME_TO_REAL
REAL_TO_STRING	SINT_TO_STRING	STRING_TO_SINT	TIME_TO_SINT
REAL_TO_TIME	SINT_TO_TIME	STRING_TO_TIME	TIME_TO_STRING
REAL_TO_TOD	SINT_TO_TOD	STRING_TO_TOD	TIME_TO_TOD
REAL_TO_UDINT	SINT_TO_UDINT	STRING_TO_UDINT	TIME_TO_UDINT
REAL_TO_UINT	SINT_TO_UINT	STRING_TO_UINT	TIME_TO_UINT
REAL_TO_USINT	SINT_TO_USINT	STRING_TO_USINT	TIME_TO_USINT
REAL_TO_WORD	SINT_TO_WORD	STRING_TO_WORD	TIME_TO_WORD
TOD_TO_<TYPE>	**UDINT_TO_<TYPE>**	**UINT_TO_<TYPE>**	**USINT_TO_<TYPE>**
TOD_TO_BOOL	UDINT_TO_BOOL	UINT_TO_BOOL	USINT_TO_BOOL
TOD_TO_BYTE	UDINT_TO_BYTE	UINT_TO_BYTE	USINT_TO_BYTE
TOD_TO_DATE	UDINT_TO_DATE	UINT_TO_DATE	USINT_TO_DATE
TOD_TO_DINT	UDINT_TO_DINT	UINT_TO_DINT	USINT_TO_DINT
TOD_TO_DT	UDINT_TO_DT	UINT_TO_DT	USINT_TO_DT
TOD_TO_DWORD	UDINT_TO_DWORD	UINT_TO_DWORD	USINT_TO_DWORD
TOD_TO_INT	UDINT_TO_INT	UINT_TO_INT	USINT_TO_INT
TOD_TO_REAL	UDINT_TO_REAL	UINT_TO_REAL	USINT_TO_REAL
TOD_TO_SINT	UDINT_TO_SINT	UINT_TO_SINT	USINT_TO_SINT
TOD_TO_STRING	UDINT_TO_STRING	UINT_TO_STRING	USINT_TO_STRING
TOD_TO_TIME	UDINT_TO_TIME	UINT_TO_TIME	USINT_TO_TIME
TOD_TO_UDINT	UDINT_TO_TOD	UINT_TO_TOD	USINT_TO_TOD
TOD_TO_UINT	UDINT_TO_UINT	UINT_TO_UDINT	USINT_TO_UDINT
TOD_TO_USINT	UDINT_TO_USINT	UINT_TO_USINT	USINT_TO_UINT
TOD_TO_WORD	UDINT_TO_WORD	UINT_TO_WORD	USINT_TO_WORD

表 4-7-1

2.7.1 BOOL_TO_<TYPE>——布尔类型转换指令

功能：把布尔数据类型转换为其它数据类型。

输入/输出数据类型（参见表 4-7- 1）：

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

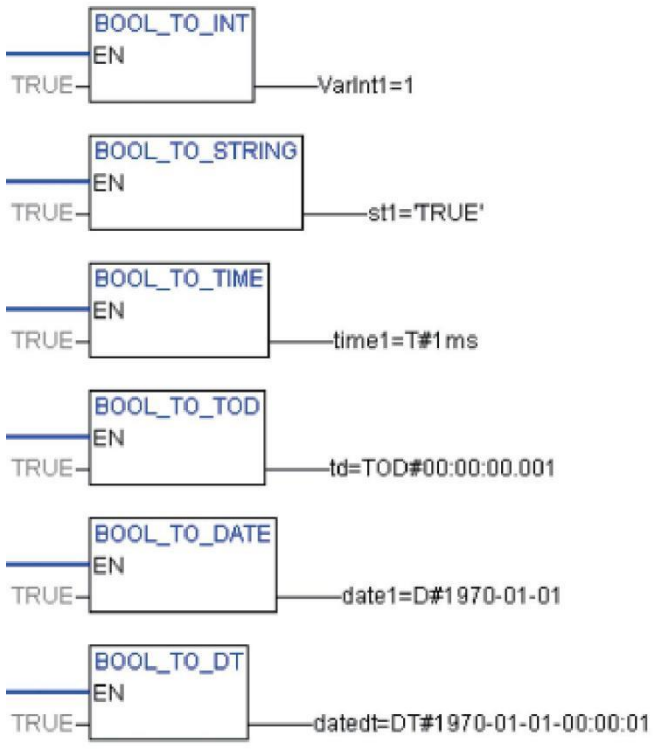
输出为字符串类型时，如果输入是 TRUE，则输出字符串'TRUE'，如果输入是 FALSE，则输出为字符串'FALSE'。

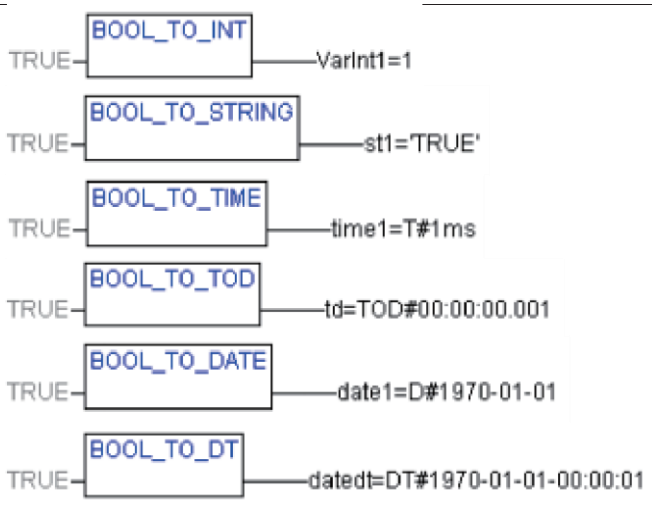
指令使用举例

变量定义

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CONSTANT

	名称	地址	类型	初始值	注释
0001	VarInt1		INT		
0002	st1		STRING		
0003	time1		TIME		
0004	td		TOD		
0005	date1		DATE		
0006	datedt		DT		

编程语言	程　序
梯形图（LD）	

功 能 块（**FBD**）	

2.7.2 BYTE_TO_<TYPE>——字节类型转换指令

功能：把字节类型转换为其他数据类型。

输入/输出数据类型（参见表 4-7- 1）：

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

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

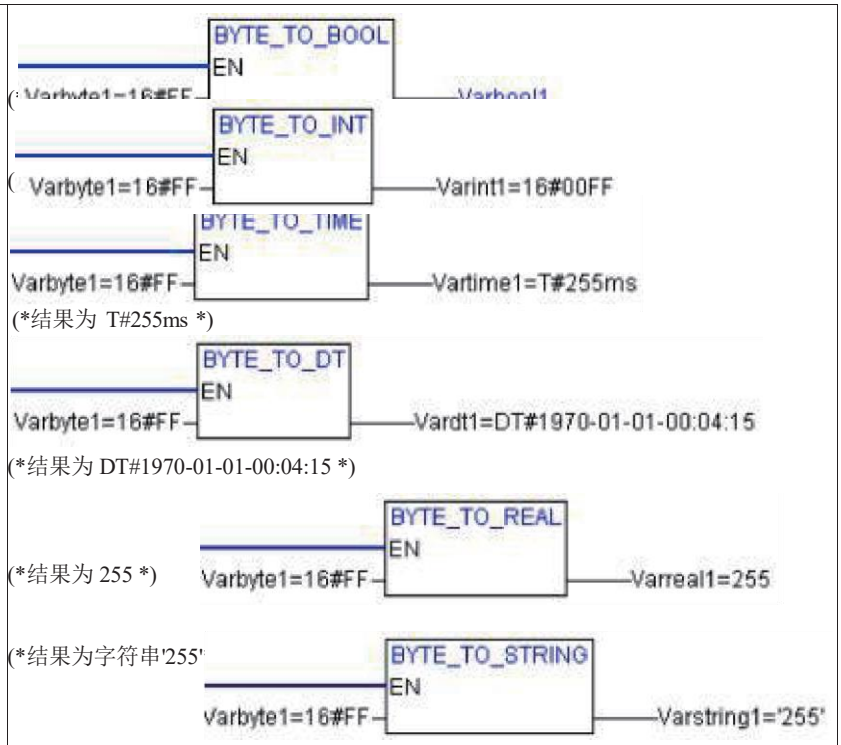
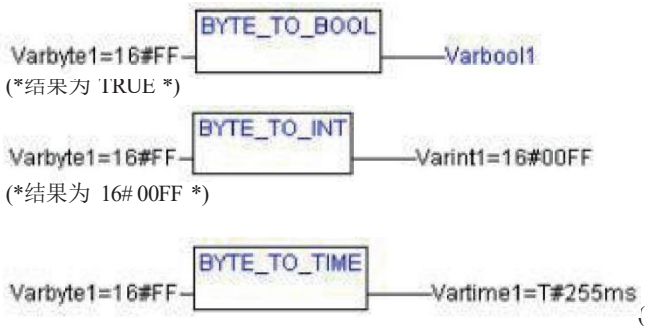
指令使用举

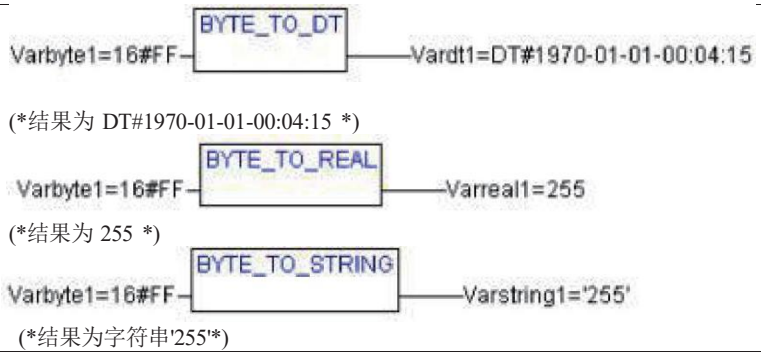
变量定义

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CONSTANT

	名称	地址	类型	初始值	注释
0001	Varbool1		BOOL		
0002	Varbyte1		BYTE		
0003	Varint1		INT		
0004	Vartime1		TIME		
0005	Vardt1		DT		
0006	Varreal1		REAL		
0007	Varstring1		STRING		

编程语言	程 序（部 分）

<table>
<tr>
<td>梯形图（LD）</td>
<td> 
</td>
</tr>
<tr>
<td>结 构 化 文 本 （ST）</td>
<td>Varbyte1:=16#FF (*Varbyte1 取值*)
Varbool1:=BYTE_TO_BOOL(Varbyte1); (*结果为 TRUE *)
Varint1:=BYTE_TO_INT(Varbyte1); (*结果为 16# FF *)
Vartime1:=BYTE_TO_TIME(Varbyte1); (*结果为 T#255ms *)
Vardt1:=BYTE_TO_DT(Varbyte1);
(*结果为 DT#1970-01-01-00:04:15 *)
Varreal1:=BYTE_TO_REAL(Varbyte1); (*结果为 255 *)
Varstring1:=BYTE_TO_STRING(Varbyte1); (*结果为字符串'255'*)</td>
</tr>
<tr>
<td>功能块（FBD）</td>
<td> 
</td>
</tr>
</table>



2.7.3 WORD_TO_<TYPE>——字类型转换指令

功能：把字类型转换为其它数据类型。

输入/输出数据类型（参见表 4-7- 1）：

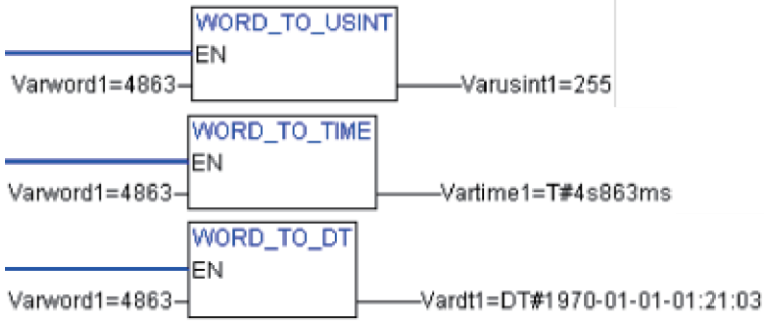
当 WORD_TO_BOOL 时，输入不等于 0 时输出为 TRUE，当输入等于 0 时,输出为FALSE；当WORD_TO_TIME 、WORD_TO_TOD 时，输入将以毫秒值进行转换；当 WORD_TO_DATE 、WORD_TO_DT 时，输入将以秒值进行转换。

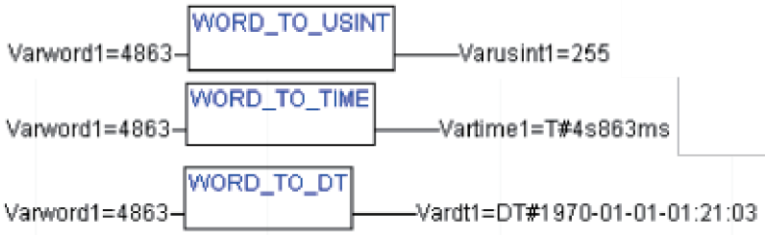
指令使用举例

变量定义

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CONSTANT

	名称	地址	类型	初始值	注释
0001	Varusint1		USINT		
0002	Varword1		WORD		
0003	Vartime1		TIME		
0004	Vardt1		DT		

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

功能块（**FBD**）	

2.7.4 DWORD_TO_<TYPE>——双字类型转换指令

功能：把双字类型转换为其它数据类型。

输入/输出数据类型（参见表 4-7- 1）：

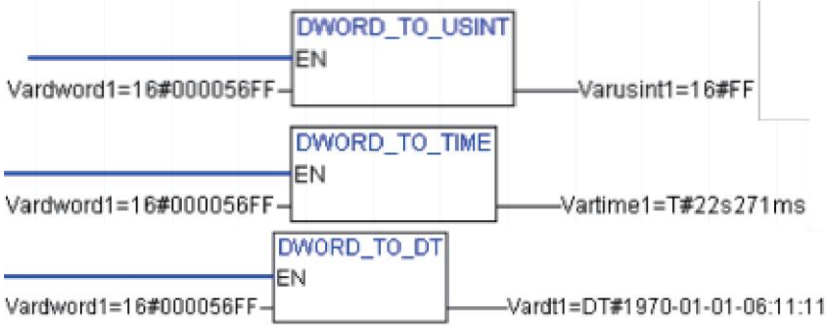
当 DWORD_TO_BOOL 时，输入不等于 0 时输出为 TRUE， 当输入等于 0 时输出为 FALSE；当 DWORD_TO_TIME 、DWORD_TO_TOD 时，输入将以毫秒值进行转换；当 DWORD_TO_DATE、DWORD_TO_DT 时，输入将以秒值进行转换。

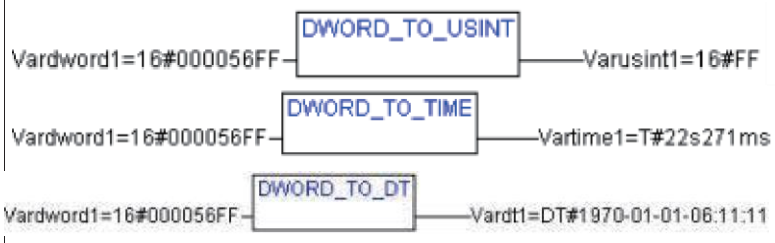
指令使用举例

变量定义

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CONSTANT

	名称	地址	类型	初始值	注释
0001	Varusint1		USINT		
0002	Vardword1		DWORD		
0003	Vartime1		TIME		
0004	Vardt1		DT		

编程语言	程 序（部 分）
梯形图（**LD**）	
结 构 化 文 本 （**ST**）	Varword1:= 16#56FF; (*Varword1 取值*) Varusint1:=DWORD_TO_USINT(Vardword1); (*结果 255 *) 说明： 如果将整数 16#56FF（十进制为 22271）保存为 USINT 型变量，则会丢失高位数 据，只显示低位数据 255 （十六进制为 16#FF）。 Vartime1:=DWORD_TO_TIME(Vardword1); (*结果 T#22s271ms*) Vardt1:=DWORD_TO_DT(Vardword1); (*结果 DT#1970-01-01-06:11:11 *)

功能块(**FBD**）	

2.7.5 SINT_TO_<TYPE>——短整型转换指令

功能：把短整型转换为其它数据类型。

输入/输出数据类型（参见表 4-7- 1）：

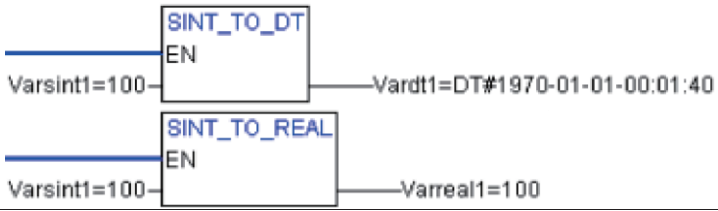
当 SINT_TO_BOOL时，输入不等于 0 时输出为TRUE，当输入等于 0 时输出为FALSE；

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

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

指令使用举例

变量定义					
VAR / VAR_INPUT / VAR_OUTPUT / VAR_IN_OUT / CONSTANT					
	名称	地址	类型	初始值	注释
0001	Varsint1		SINT		
0002	Vardt1		DT		
0003	Varreal1		REAL		

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

2.7.6 USINT_TO_<TYPE>——无符号短整型转换指令

功能：把无符号短整型转换为其它数据类型。

输入/输出数据类型（参见表 4-7- 1）：

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

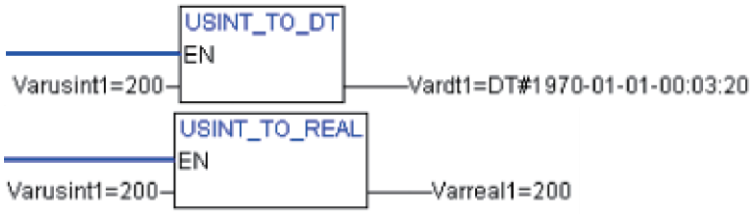
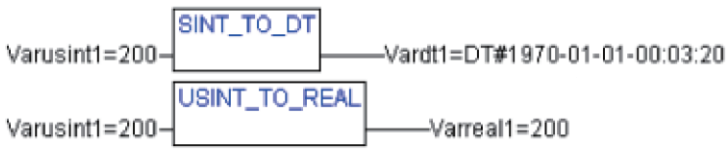
当 USINT_TO_DATE 、USINT _TO_DT 时，输入将以秒值进行转换。

指令使用举例

变量定义

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CONSTANT

	名称	地址	类型	初始值	注释
0001	Varusint1		USINT		
0002	Vardt1		DT		
0003	Varreal1		REAL		

编程语言	程 序（部 分）
梯形图（**LD**）	
结 构 化 文 本（**ST**）	Varusint1:=200; (*Varusint1 取值*) Vardt1:=USINT_TO_DT(Varusint1);(*结果 DT#1970-01-01-00:03:20 *) Varreal1:=USINT_TO_REAL(Varusint1); (*结果为 200 .0*)
功能块（**FBD**）	

2.7.7 INT_TO_<TYPE>——整数类型转换指令

功能：把整型数据类型转换为其它数据类型。

输入/输出数据类型（参见表 4-7- 1）：

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

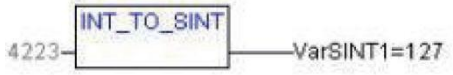
当 INT_TO_DATE、INT _TO_DT 时，输入将以秒值进行转换。

指令使用举例

变量定义

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CONS

	名称	地址	类型	初始值	注释
0001	VarSINT1		SINT		
0002	VarREAL1		REAL		

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

2.7.8 UINT_TO_<TYPE>——无符号整数类型转换指令

功能：无符号整数类型转换为其它数据类型。

输入/输出数据类型（参见表 4-7- 1）：

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

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

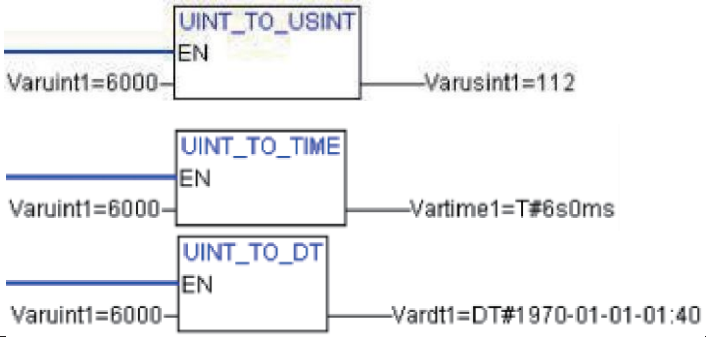
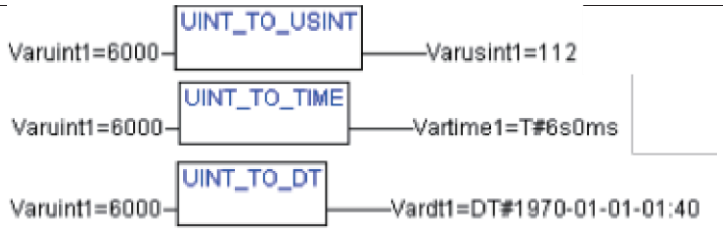
当 UINT_TO_DATE、UINT _TO_DT 时，输入将以秒值进行转换。

指令使用举例

变量定义

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CONSTANT

	名称	地址	类型	初始值	注释
0001	Varuint1		UINT		
0002	Varusint1		USINT		
0003	Vartime1		TIME		
0004	Vardt1		DT		

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

2.7.9 DINT_TO_<TYPE>——双整数类型转换指令

功能：双整数类型类型转换为其它数据类型。

输入/输出数据类型（参见表 4-7- 1）：

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

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

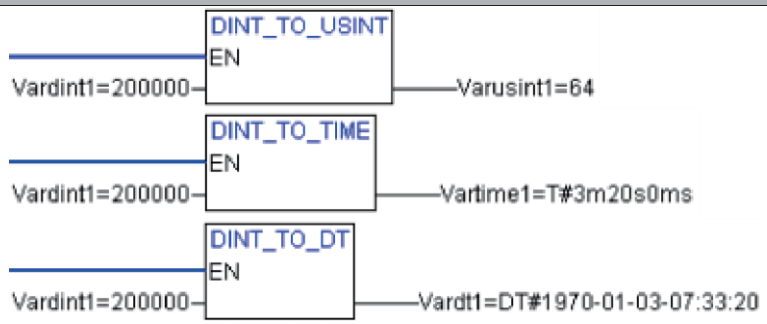
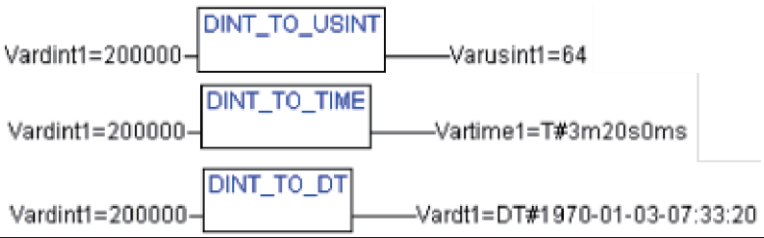
当 DINT_TO_DATE、DINT _TO_DT 时，输入将以秒值进行转换。

指令使用举例

变量定义

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CONSTANT

	名称	地址	类型	初始值	注释
0001	Varuint1		UINT		
0002	Varusint1		USINT		
0003	Vartime1		TIME		
0004	Vardt1		DT		

编程语言	程 序（部 分）
梯形图（**LD**）	
结构化文本（**ST**）	Vardint1:=200000; Varusint1:=DINT_TO_USINT(Vardint1);(*结果64*) 说明： 如果将整数200000（十六进制为16#30D40）保存为USINT 型变量， 则会丢失高位数据， 只显示低位数据64（十六进制为16#40）。 Vartime1:=DINT_TO_TIME(Vardint1); (*结果 T#3m20s0ms*) Vardt1:=DINT_TO_DT(Vardint1);(*结果DT#1970-01-03-07:33:20 *)
功能块（**FBD**）	

2.7.10 UDINT_TO_<TYPE>——无符号双整数类型转换指令

功能：无符号双整数类型转换为其它数据类型。

输入/输出数据类型（参见表 4-7- 1）：

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

指令使用举例

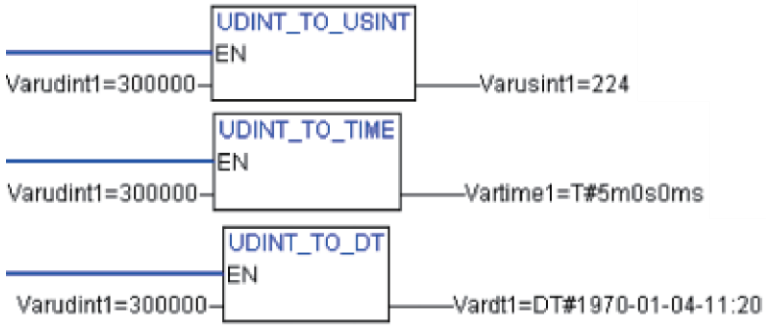
变量定义

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CONSTANT

	名称	地址	类型	初始值	注释
0001	Varudint1		UDINT		
0002	Varusint1		USINT		
0003	Vartime1		TIME		
0004	Vardt1		DT		

编程语言 | **程 序（部 分）**

梯形图（LD）



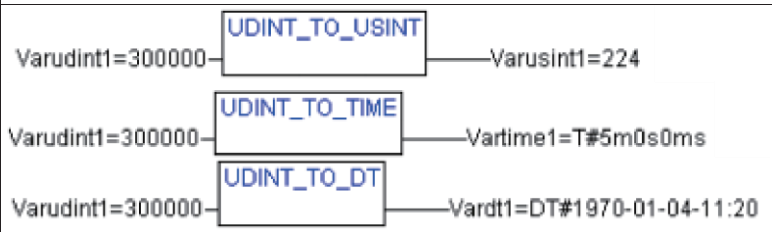
结构化文本（ST）

```
Varudint1:=300000;
Varusint1:= UDINT_TO_USINT(Varudint1);        (*结果 224*)
```

说明：如果将整数 300000（十六进制为 16#493E0）保存为 USINT 型变量，则会丢失高位数据， 只显示低位数据 224（十六进制为 16#E0）。

```
Vartime1:=UDINT_TO_TIME(Varudint1);           (*结果 T#5m0s0ms*)
Vardt1:=UDINT_TO_DT(Varudint1);
(*结果 DT#1970-01-04-11:20:00 *)
```

功能块（FBD）



2.7.11 REAL_TO_<TYPE>——实数类型转换指令

功能：把浮点数转换为其它类型数据。把浮点数转换为其它类型数据时，先将值四舍 五入成整数值，然后转成新的变量类型。

输入/输出数据类型（参见表 4-7- 1）：

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

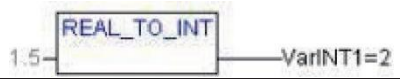
当 REAL_TO_DATE 、REAL_TO_DT 时，输入将以秒值进行转换。

指令使用举例

变量定义

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CONSTANT

	名称	地址	类型	初始值	注释
0001	Varint1		INT		
0002	Varint2		INT		
0003	Varint3		INT		
0004	Varint4		INT		

编程语言	程 序（部 分）
梯形图（LD）	
结构化文本（**ST**）	VarINT1:= REAL_TO_INT(1.5); (*结果 VarINT1 为 2*) VarINT2:= REAL_TO_INT(1.4); (*结果 VarINT2 为 1*) VarINT3:= REAL_TO_INT(-1.5); (*结果 VarINT3 为-2*) VarINT4:= REAL_TO_INT(-1.4); (*结果 VarINT4 为-1*)
功能块（**FBD**）	

2.7.12 TIME_TO_<TYPE>—— 时间类型转换指令

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

输入/输出数据类型（参见表 4-7- 1）：

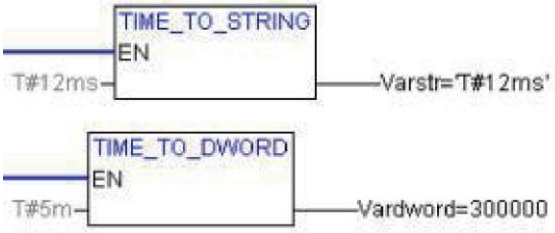
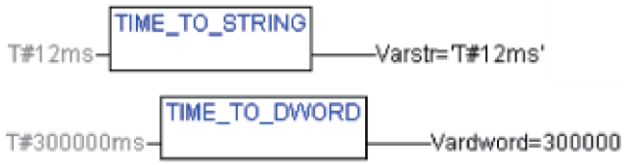
当 TIME_TO_BOOL 时，输入不等于 0 时输出为 TRUE，当输入等于 0 时输出为 FALSE。

指令使用举例

变量定义

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CONS

	名称	地址	类型	初始值	注释
0001	Varstr		STRING		
0002	Vardword		DWORD		

编程语言	程 序（部 分）
梯形图（**LD**）	
结构化文本（**ST**）	Varstr:=TIME_TO_STRING(T#12ms); (*结果为‘T#12ms’*) Vardword:=TIME_TO_DWORD(T#5m); (*结果为 300000*)
功能块（**FBD**）	

2.7.13 DATE_TO_<TYPE>—— 日期类型转换指令

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

输入/输出数据类型（参见表 4-7- 1）：

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

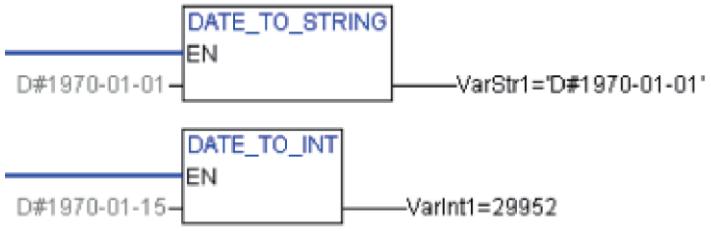
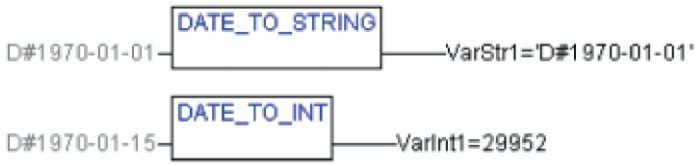
指令使用举例

变量定义

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CONS

	名称	地址	类型	初始值	注释
0001	Varint1		INT		
0002	VarStr1		STRING		

编程语言	程 序（部 分）

梯形图（**LD**）	
结构化文本（**ST**	VarStr1:=DATE_TO_STRING(D#1970-01-01); (*结果为'D#1970-01-01' *) VarInt1:=DATE_TO_INT(D#1970-01- 15); (*结果为 29952*) 说明： 将 D#1970-01-15 （十进制 14×24×3600=1209600= 16#127500）保存为 INT 型变量，则会丢失高位数据， 只显示低位数据 16#7500,转换十进制数为 29952。
功能块（**FBD**）	

2.7.14 DT_TO_<TYPE>—— 日期时间类型转换指令

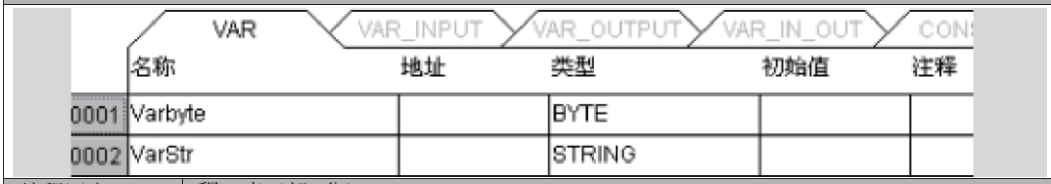
功能：把日期时间型数据转换为其它类型数据，日期在内部以秒为单位存储，时间从1970 年 1 月 1 日开始。

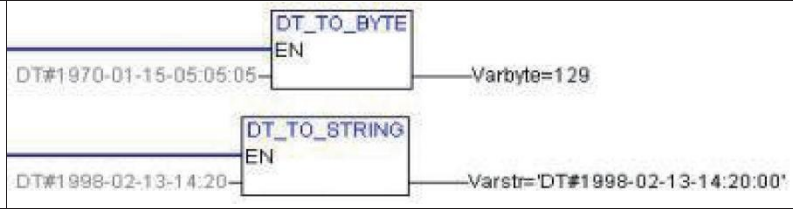
输入/输出数据类型（参见表 4-7- 1）：

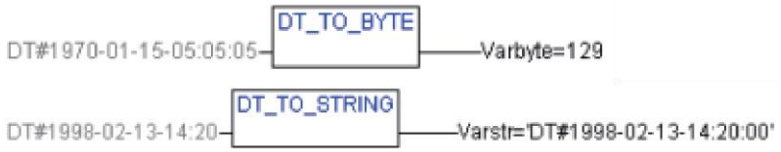
当 DT_TO_BOOL 时，输入不等于 0 时输出为 TRUE，当输入等于 0 时输出为 FALSE。

指令使用举例

变量定义



编程语言	程 序（部 分）
梯形图（**LD**）	

结构化文本（**ST**）	Varbyte:=DT_TO_BYTE(DT#1970-01-15-05:05:05); (*结果 Varbyte 为 129*) 说明：将 DT#1970-01-15-05:05:05 转换成秒数，值为(((14*24+5)*60+5)*60+5)=1227905=16#12BC81，保存为 BYTE 型变量，则会丢失高 24 位数据， 只显示低 8 位数据， 16#81= 129。 Varstr:=DT_TO_STRING(DT#1998-02-13-14:20); (*结果 Varstr 为‘DT#1998-02-13-14:20’)
功能块（**FBD**）	

2.7.15 TOD_TO_<TYPE>——时间类型转换指令

功能：把时间型数据转换为其它类型数据，日期在内部以毫秒为单位进行转化。 输入/输出数据类型（参见表 4-7- 1）：

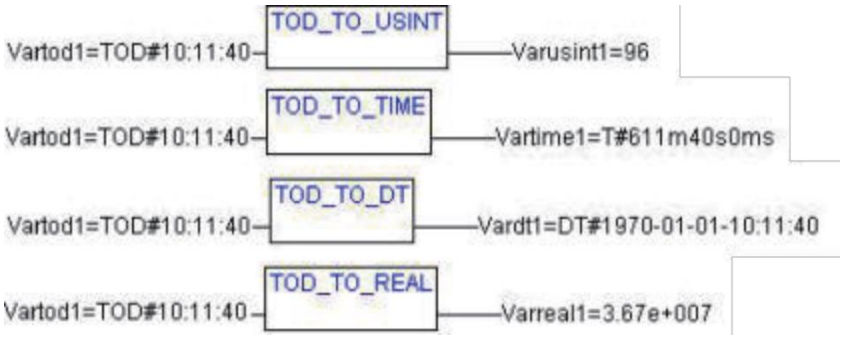
当TOD_TO_BOOL 时，输入不等于 0 时输出为TRUE，当输入等于0 时输出为FALSE。

指令使用举例

变量定义

	名称	地址	类型	初始值	注释
0001	Vartod1		TOD		
0002	varbool1		BOOL		
0003	Varusint1		USINT		
0004	Vartime1		TIME		
0005	Vardt1		DT		
0006	Varreal1		REAL		

编程语言	程 序（部 分）
梯形图（**LD**）	

结构化文本（**ST**）	Vartod1:=TOD#10:11:40; (*Vartod1 取值*) Varusint1:=TOD_TO_USINT(Vartod1); (*结果为 96*) Vartime1:=TOD_TO_TIME(Vartod1); (*结果为 T#611m40s0ms*) Vardt1:=TOD_TO_DT(Vartod1); (*结果为 DT#1970-01-01- 10:11:40*) Varreal1:=TOD_TO_REAL(Vartod1); (*结果为 3.67e+007*)
功能块（**FBD**）	

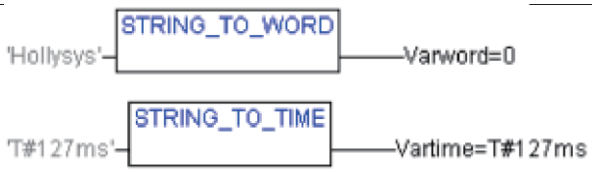
2.7.16 STRING_TO_<TYPE>——字符类型转换指令

功能：把字符串转换为其它类型数据，字符串型变量必须包含一个有效的目标变量值， 否则转换结果为 0。

输入/输出数据类型： （参见表 4-7- 1）

指令使用举例

变量定义

编程语言	程 序（部 分）
梯形图（**LD**）	
结构化文本（**ST**）	Varword:=STRING_TO_WORD('Hollysys'); (*结果为 0*) Vartime:=STRING_TO_TIME('T#127ms'); (*结果为 T#127ms*)
功能块（**FBD**）	

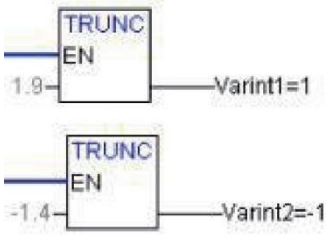
2.7.17 TRUNC——截短转换指令

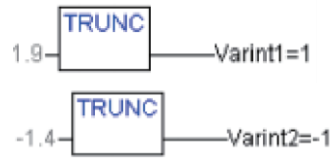
功能：该指令将数据截去小数部分，只保留整数部分。

输入/输出数据类型：输入为 REAL 型，输出为 INT 、WORD 、DWORD 型。

指令使用举例

变量定义

编程语言	程 序
梯形图（**LD**）	
结构化文本（**ST**）	Varint1:=TRUNC(1.9); (*结果 Varint1 为 1*) Varint2:=TRUNC(-1.4); (*结果 Varint2 为-1*)

功能块（**FBD**）	

提示：

当从较大数据类型转为较小数据类型时，有可能丢失信息。

本指令只是截取整数部分， 如果想四舍五入取整，可以使用 REAL_TO_INT 指令。

2.8 初等数学运算指令

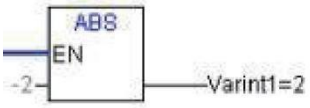
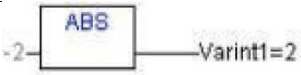
2.8.1 ABS——绝对值指令

功能：把输入数据的绝对值赋予输出变量。

输入/输出数据类型：见下表

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

指令使用举例

变量定义	
	
编程语言	程　序
梯形图（**LD**）	
结构化文本（**ST**）	i:=ABS(-2);　　(*结果 Varint1 为 2*)
功能块（**FBD**）	

2.8.2 SQRT——平方根指令

功能：对输入数据求平方根，输入数据为非负数。

输入/输出数据类型：输入数据类型可以是 BYTE、WORD、DWORD、INT、DINT、REAL、SINT、USINT、 UINT 、UDINT，输出数据必须是 REAL 类型。

指令使用举例

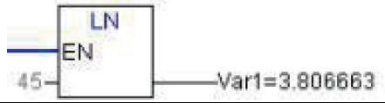
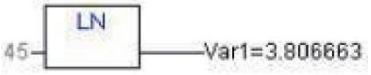
变量定义	
	
编程语言	程　序
梯形图（**LD**）	
结构化文本（**ST**）	Var1:=SQRT(10); (*结果 Var1 为 3.162278*)
功能块（**FBD**）	

2.8.3　LN—— 自然对数指令

功能：对输入数据求自然对数，输入数据必须为正数。

输入/输出数据类型：输入数据类型可以是 BYTE、WORD、DWORD、INT、DINT、REAL、SINT、USINT、 UINT 、UDINT，输出数据必须是 REAL 型。

指令使用举例

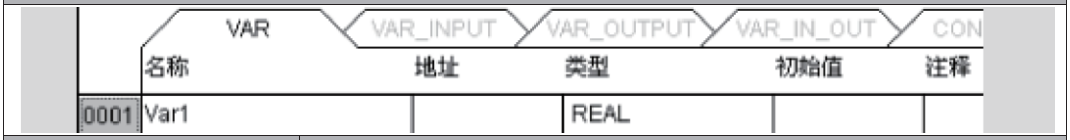
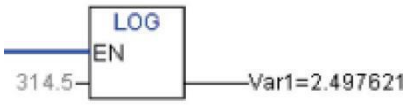
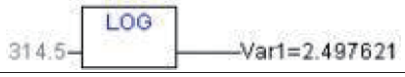
变量定义	
	
编程语言	程　序
梯形图（**LD**）	
结构化文本（**ST**）	Var1:=LN(45)；　(*结果 Var1 为 3.806663*)
功能块（**FBD**）	

2.8.4 LOG——常用对数指令

功能：对输入数据求以 10 为底的对数，输入数据必须为正数。

输入/输出数据类型：输入数据类型可以是 BYTE、WORD、DWORD、INT、DINT、REAL、SINT、USINT、 UINT 、UDINT，输出数据必须是 REAL 型。

指令使用举例

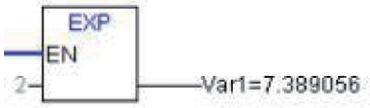
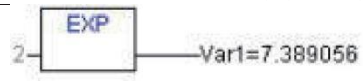
<table>
<tr><td colspan="2">变量定义
 
</td></tr>
<tr><td>编程语言</td><td>程 序</td></tr>
<tr><td>梯形图（LD）</td><td> 
</td></tr>
<tr><td>结构化文本（ST）</td><td>Var1:=LOG(314.5); (*结果 Var1 为 2.497621 *)</td></tr>
<tr><td>功能块（FBD）</td><td> 
</td></tr>
</table>

2.8.5 EXP——指数指令

功能：以输入数据为指数的幕计算，即 ye x ，其中 x 为输入，y 为输出。

输入/输出数据类型：输入数据类型可以是 BYTE、WORD、DWORD、INT、DINT、REAL、SINT、USINT、 UINT 、UDINT，输出数据必须是 REAL 型。

指令使用举例

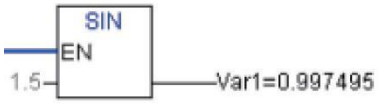
<table>
<tr><td colspan="2">变量定义
 
</td></tr>
<tr><td>编程语言</td><td>程 序</td></tr>
<tr><td>梯形图（LD）</td><td> 
</td></tr>
<tr><td>结构化文本（ST）</td><td>Var1:=EXP(2); （*结果 Var1 为 7.389056*）
EXP
ST Var1 （*结果 Var1 为 7.389056*）</td></tr>
<tr><td>功能块（FBD）</td><td> 
</td></tr>
</table>

2.8.6 SIN——正弦指令

功能：求输入数据的正弦值，输入数据以弧度表示。 弧度(rad) 角度* $\frac{\pi}{180^\circ}$

输入/输出数据类型：输入数据类型可以是 BYTE 、WORD 、DWORD 、INT 、DINT、REAL、 SINT 、USINT 、UINT 、UDINT，输出数据必须是 REAL 型。

指令使用举例

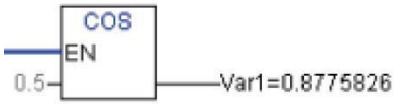
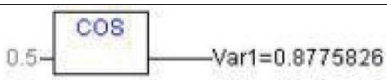
变量定义	
	
编程语言	程 序
梯形图（**LD**）	
结构化文本（**ST**）	Var1:=SIN(1.5); (*结果 Var1 为 0.997495 *)
功能块（**FBD**）	

2.8.7 COS——余弦指令

功能：求输入数据的余弦值，弧度(rad) 角度* $\frac{\pi}{180^\circ}$

输入/输出数据类型：输入数据类型可以是 BYTE 、WORD 、DWORD 、INT 、DINT、 REAL、 SINT 、USINT 、UINT 、UDINT，输出数据必须是 REAL 型。

指令使用举例

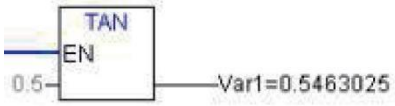
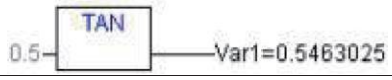
变量定义	
	
编程语言	程 序
梯形图（**LD**）	
结构化文本（**ST**）	Var1:=COS(0.5); (*结果 Var1 为 0.8775826 *)
功能块（**FBD**）	

2.8.8 TAN——正切指令

功能：求输入数据的正切值，弧度(rad) 角度* $\frac{\pi}{180^{\circ}}$

输入/输出数据类型：输入数据类型可以是 BYTE 、WORD 、DWORD 、INT 、DINT、REAL、 SINT 、USINT 、UINT 、UDINT，输出数据必须是 REAL 型。

指令使用举例

<table>
<tr><td colspan="2">变量定义</td></tr>
<tr><td colspan="2"> 
</td></tr>
<tr><td>编程语言</td><td>程　序</td></tr>
<tr><td>梯形图（LD）</td><td> 
</td></tr>
<tr><td>结构化文本（ST）</td><td>Var1:=TAN(0.5);　　(*结果 Var1 为 0.5463025 *)</td></tr>
<tr><td>功能块（FBD）</td><td> 
</td></tr>
</table>

2.8.9 ASIN——反正弦指令

功能：求输入数据的反正弦值。

输入数据类型可以是 BYTE、WORD、DWORD、INT、DINT、REAL、SINT、USINT、 UINT 、UDINT，输出数据必须是 REAL 型，输出数据以弧度表示。

指令使用举例

<table>
<tr><td colspan="2">变量定义</td></tr>
<tr><td colspan="2"> 
</td></tr>
<tr><td>编程语言</td><td>程　序</td></tr>
<tr><td>梯形图（LD）</td><td> 
</td></tr>
</table>

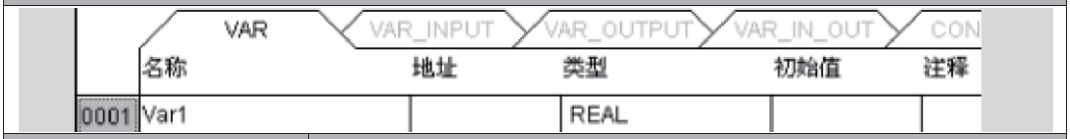
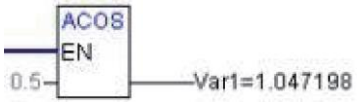
结构化文本（**ST**）	Var1:=ASIN(0.5);　(*结果 Var1 为 0.5235988 *)
功能块（**FBD**）	

2.8.10　ACOS——反余弦指令

功能：求输入数据的反余弦值。

输入数据类型可以是 BYTE、WORD、DWORD、INT、DINT、REAL、SINT、USINT、 UINT 、UDINT，输出数据必须是 REAL 型，输出数据以弧度表示。

指令使用举例

变量定义	
	
编程语言	程　序
梯形图（**LD**）	
结构化文本（**ST**）	Var1:=ACOS(0.5);　(*结果 Var1 为 1.047198 *)
功能块（**FBD**）	

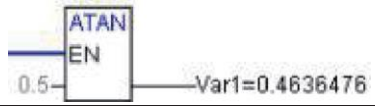
2.8.11　ATAN——反正切指令

功能：求输入数据的反正切值。

输入/输出数据类型：输入数据类型可以是 BYTE 、WORD 、DWORD 、INT 、DINT、 REAL、SINT 、USINT 、UINT 、UDINT，输出数据必须是 REAL 型，输出数据以弧 度表示。

指令使用举例

变量定义	
	
编程语言	程　序

梯形图（**LD**）	
结构化文本（**ST**）	Var1:=ATAN(0.5); (*结果 Var1 为 0.4636476 *)
功能块（**FBD**）	

2.8.12 EXPT——幕指令

功能：对输入数据求幕，输入数据 1 为幕底数，输入数据 2 为幕指数。

输入/输出数据类型：输入数据的类型可以是 BYTE、WORD、DWORD、INT、DINT、

REAL、 SINT 、USINT 、UINT 、UDINT，输出数据必须是 REAL 型。

指令使用举例

变量定义	
	
编程语言	程 序
梯形图（**LD**）	
结构化文本（**ST**）	Var1:=EXPT(7,2); (*结果 Var1 为 49 *)
功能块（**FBD**）	

2.9 地址运算指令

2.9.1 ADR——取地址指令

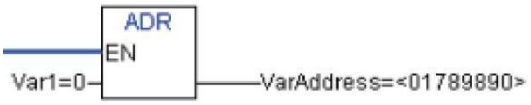
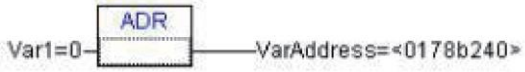
功能：取得输入变量的内存地址并输出。该地址可以在程序内当作指针使用，也可以作为指针传送给函数。

指令使用举例

变量定义

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CONS

	名称	地址	类型	初始值	注释
0001	Var1		BYTE		
0002	VarAddress		POINTER TO BYTE		

编程语言	程　序
梯形图（**LD**）	
结构化文本（**ST**）	VarAddress:= ADR(Var1);
功能块（**FBD**）	

2.9.2　Λ——取地址内容指令

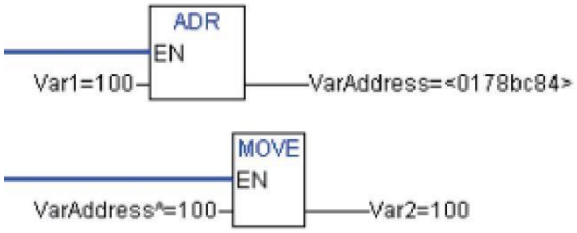
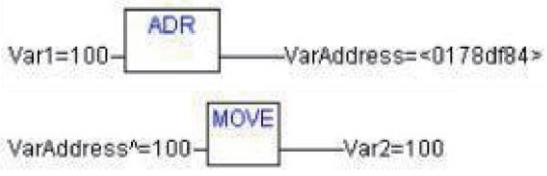
功能：在指针变量后增加 “ ^ ”符号，以取得该指针所指地址的数据。

指令使用举例

变量定义

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CON

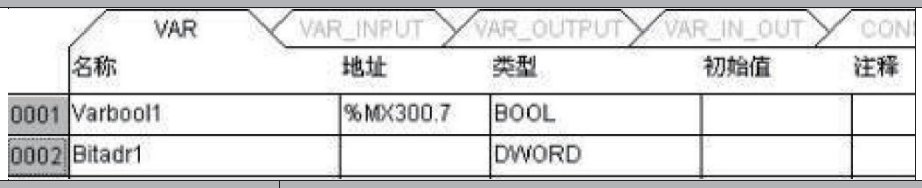
	名称	地址	类型	初始值	注释
0001	Var1		BYTE		
0002	Var2		BYTE		
0003	VarAddress		POINTER TO BYTE		

编程语言	程　序
梯形图（**LD**）	
结构化文本（**ST**）	VarAddress:= ADR(Var1); Var2:= VarAddress^;　(*结果 Var2 为 100 *)
功能块（**FBD**）	

2.9.3 BITADR——位地址指令

功能：取得 BOOL 量的位地址，下例中 MX300 .7 的地址为300*8+7=2407。

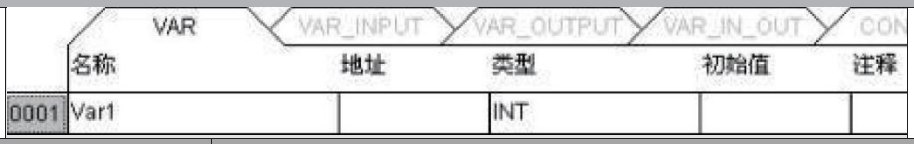
指令使用举例

<table>
<tr><td colspan="2">变量定义</td></tr>
<tr><td colspan="2"> 
</td></tr>
<tr><td>编程语言</td><td>程 序</td></tr>
<tr><td>梯形图（LD）</td><td> 
</td></tr>
<tr><td>结构化文本（ST）</td><td>Bitadr1:=BITADR(Varbool1);</td></tr>
<tr><td>功能块（FBD）</td><td> 
</td></tr>
</table>

2.9.4 INDEXOF——索引指令

功能：在 POU 中执行索引指令，可以寻找 POU 的索引号，其输入为POU的名称，输出为 INT 的数据。

指令使用举例

<table>
<tr><td colspan="2">变量定义</td></tr>
<tr><td colspan="2"> 
</td></tr>
<tr><td>编程语言</td><td>程 序</td></tr>
<tr><td>梯形图（LD）</td><td> 

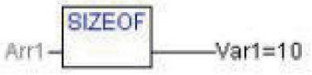
（POU2 可以是程序、功能块、函数）</td></tr>
<tr><td>结构化文本（ST）</td><td>Var1:=INDEXOF(POU2); （*结果 Var1 为 38*）</td></tr>
<tr><td>功能块（FBD）</td><td> 

（*结果 Var1 为 38*）</td></tr>
</table>

2.9.5 SIZEOF——数据类型大小指令

功能：取得数据类型所需字节数。

指令使用举例

<table>
<tr><td colspan="2">变量定义
 
</td></tr>
<tr><td>编程语言</td><td>程　序</td></tr>
<tr><td>梯形图（LD）</td><td> 
</td></tr>
<tr><td>结构化文本（ST）</td><td>Var1:=SIZEOF(arr1);　（*结果 Var1 为 10*）</td></tr>
<tr><td>功能块（FBD）</td><td> 
</td></tr>
</table>

2.10 调用指令

CAL—调用指令

功能：调用功能块或者程序。 在 IL 语言中使用 CAL 运算来调用功能块或者程序。被 调用功能块/程序的输入变量位于该功能块/程序名称右侧的括号内。

指令使用举例

在程序中调用名称为 Inst 的功能块，其输入变量 Par1 等于 0 、Par2 等于 TRUE。IL 中调用如下：

CAL Inst(Par1:=0 , Par2:=TRUE)

2.11 初始化操作指令

INI—初始化操作指令

功能：用于初始化在程序中使用的功能块程序的内部保持型变量。

指令使用举例

语法: <bool-Variable> := INI(<FB-instance, TRUE|FALSE)

在程序中调用名称为 FB-instance 的功能块，其输入变量分别为 Par1=FB-instance 、Par2= TRUE|FALSE，输出为初始化完成标志。

主程序 **PLC_PRG** 变量定义：

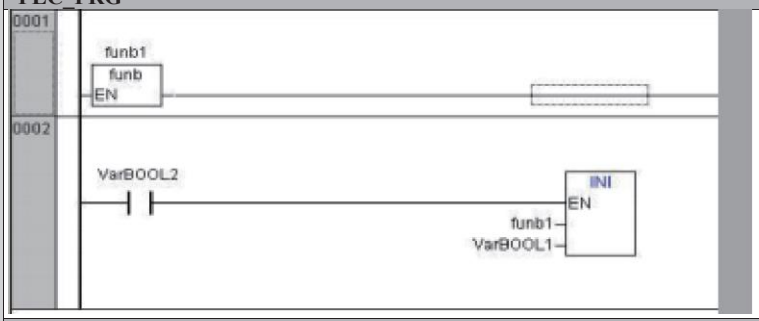
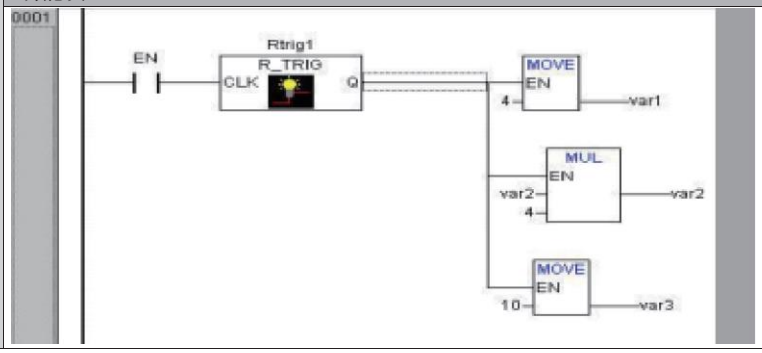
VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CONSTANT

	名称	地址	类型	初始值	注释
0001	VarBOOL2		BOOL		
0002	VarBOOL1		BOOL		
0003	funb1		funb		

funb（FB）变量定义：

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CONSTANT | RETAIN

	名称	地址	类型	初始值	注释
0001	Rtrig		R_TRIG		
0002	var1		INT	1	
0003	var2		INT	2	
0004	var3		INT	3	

编程语言	程　序
梯形图（**LD**）	**PLC_PRG**  功能块 **funb** 

程序说明：程序运行第一个扫描周期将变量 var1 、var2 、var3 的值改为 4 、 8 、 10，强制变量 VarBOOL2 接通，INI 指令执行，将三个保持型变量恢复为初始值 1 、2 、3。

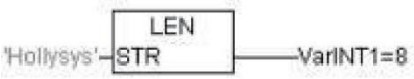
2.12 字符串处理指令（Standard.lib）

2.12.1 LEN——取字符串长度指令

功能：计算字符串的长度。

输入/输出数据类型： 输入是 STRING 类型，输出是 INT 类型。

指令使用举例

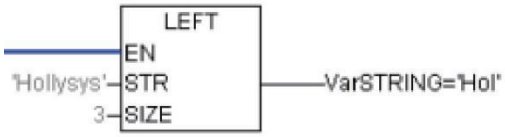
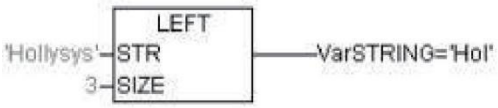
<table>
<tr><td colspan="2">变量定义
 
</td></tr>
<tr><td>编程语言</td><td>程 序</td></tr>
<tr><td>梯形图（LD）</td><td> 
</td></tr>
<tr><td>结构化文本（ST）</td><td>Var1INT1 := LEN (‘Hollysy’s); (结*果 Var1INT1 为 8*)</td></tr>
<tr><td>功能块（FBD）</td><td> 
</td></tr>
</table>

2.12.2 LEFT——左边取字符串指令

功能：从字符串左边取字符串。

指令格式： LEFT （STR,SIZE），其中输入 STR 是 STRING 类型， 为输入字符串，SIZE 是 INT 型，为从输入字符串左边开始获取的字符个数输出数据类型是 STRING 型。

指令使用举例

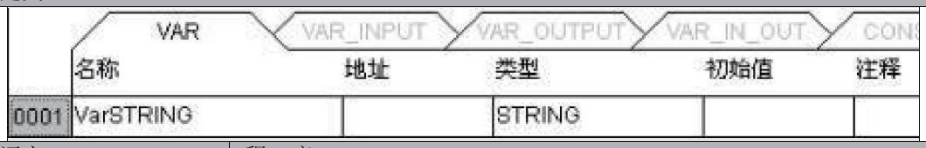
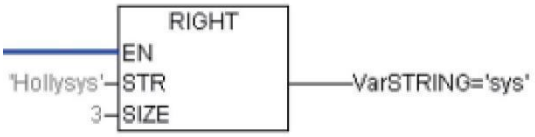
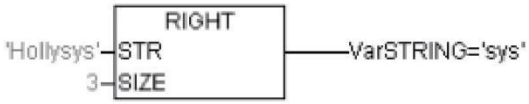
<table>
<tr><td colspan="2">变量定义
 
</td></tr>
<tr><td>编程语言</td><td>程 序</td></tr>
<tr><td>梯形图（LD）</td><td> 
</td></tr>
<tr><td>结构化文本（ST）</td><td>VarSTRING := LEFT (‘Hollysys’,3);结(*果为' Hol '*)</td></tr>
<tr><td>功能块（FBD）</td><td> 
</td></tr>
</table>

2.12.3 RIGHT——右边取字符串指令

功能：从字符串右边取字符串。

指令格式：RIGHT （STR,SIZE） ，其中输入 STR 是 STRING 类型，为输入字符串， SIZE 是 INT 型，为从字符串右边开始获取的字符个数，输出数据类型是 STRING 型。

指令使用举例

<table>
<tr><td colspan="2">变量定义
 
</td></tr>
<tr><td>编程语言</td><td>程 序</td></tr>
<tr><td>梯形图（LD）</td><td> 
</td></tr>
<tr><td>结构化文本（ST）</td><td>VarSTRING := RIGHT(‘Hollysys’,3); 结(*果为' sys '*)</td></tr>
<tr><td>功能块（FBD）</td><td> 
</td></tr>
</table>

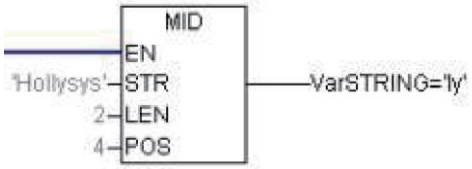
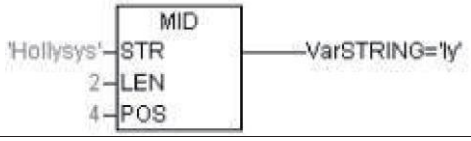
2.12.4 MID—— 取字符串指令

功能：从字符串中间取字符串。

指令格式：MID(STR,LEN,POS)，其中输入 STR 是 STRING 类型，为输入字符串。

LEN 和 POS 是 INT 型，该指令从 POS 开始从左往右获取 LEN 个字符，输出数据类型是 STRING 型。

指令使用举例

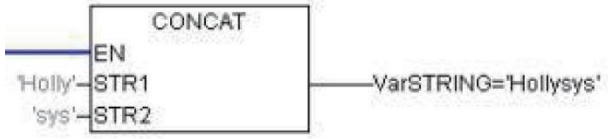
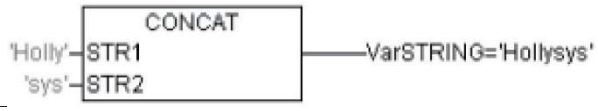
<table>
<tr><td colspan="2">变量定义
 
</td></tr>
<tr><td>编程语言</td><td>程 序</td></tr>
<tr><td>梯形图（LD）</td><td> 
</td></tr>
<tr><td>结构化文本（ST）</td><td>VarSTRING:= MID(‘Hollysys’,2,4); 结(*果为 ‘ly’*)</td></tr>
<tr><td>功能块（FBD）</td><td> 
</td></tr>
</table>

2.12.5 CONCAT——合并字符串指令

功能：把两个字符串按前后顺序结合成一个字符串，输入和输出都是 STRING 型。

指令使用举例

变量定义

编程语言	程　序
梯形图（**LD**）	
结构化文本（**ST**）	VarSTRING := CONCAT ('Holly', 'sys'); (*结果为'Hollysys'*)
功能块（**FBD**）	

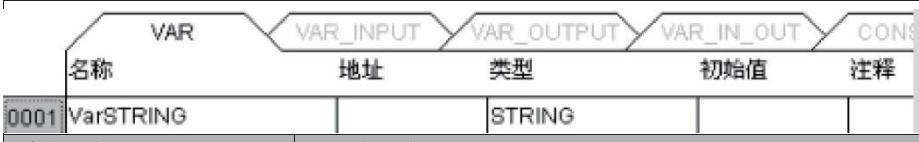
2.12.6 INSERT——插入字符串指令

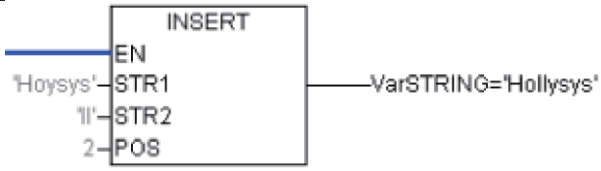
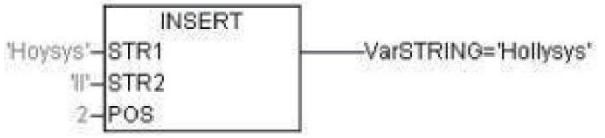
功能：把一个字符串插入到另一个字符串中。

指令格式：INSERT(STR1,STR2,POS)。输入 STR1 和 STR2 是 STRING 类型， POS 是 INT 型，该指令把 STR2 插入到 STR1 的 POS 位置之后。输出数据类型是 STRING 型。

指令使用举例

变量定义



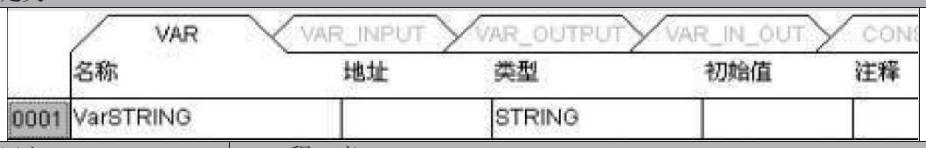
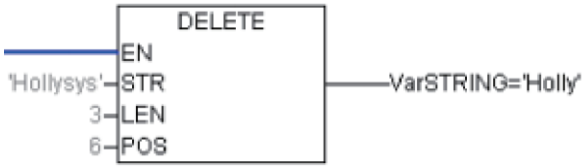
编程语言	程　序
梯形图（**LD**）	
结构化文本（**ST**）	VarSTRING := INSERT ('Hoysys', 'll',2);结(*果为 'Hollysys'*)
功能块（**FBD**）	

2.12.7 DELETE——删除字符指令

功能：从字符串中删除字符。

指令格式： DELETE(STR,LEN,POS)。输入 STR 是 STRING 类型，为输入字符串。 LEN 和 POS 是 INT 型， 该指令从输入字符的位置 POS 处开始从左往右删除 LEN 个字符， 输出数据类型是 STRING 型。

指令使用举例

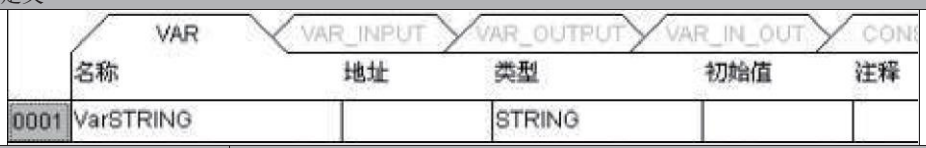
<table>
<tr><td colspan="2">变量定义</td></tr>
<tr><td colspan="2"> 
</td></tr>
<tr><td>编程语言</td><td>程 序</td></tr>
<tr><td>梯形图（LD）</td><td> 
</td></tr>
<tr><td>结构化文本（ST）</td><td>VarSTRING := DELETE ('Hollysys',3,6); (*结果为'Holly'*)</td></tr>
<tr><td>功能块（FBD）</td><td> 
</td></tr>
</table>

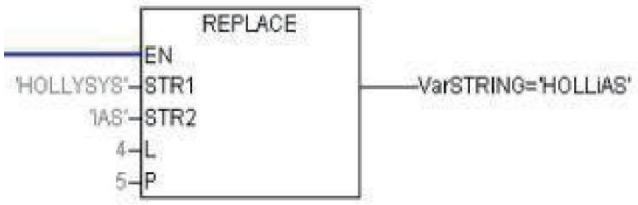
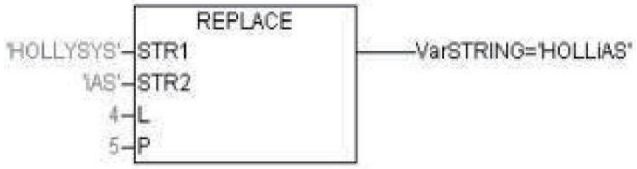
2.12.8 REPLACE——替换字符串指令

功能：用一个字符串替代另一字符串中的部分内容。

指令格式： REPLACE(STR1,STR2,L,P)。输入 STR1 和 STR2 是 STRING 类型，为输 入字符串。L 和 P 是 INT 型，该指令用 STR2 代替 STR1 中从 P 位置开始的 L 个字符。输出数据类型是 STRING 型。

指令使用举例

<table>
<tr><td colspan="2">变量定义</td></tr>
<tr><td colspan="2"> 
</td></tr>
<tr><td>编程语言</td><td>程 序</td></tr>
</table>

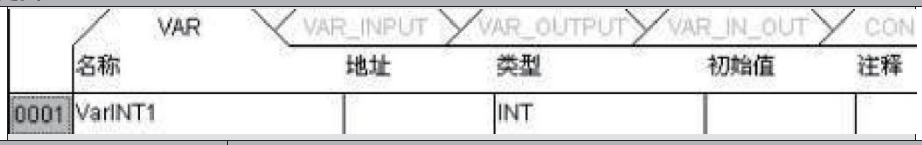
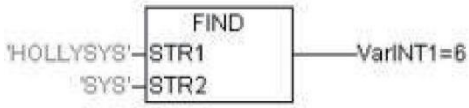
梯形图（**LD**）	
结构化文本（**ST**）	VarSTRING:= REPLACE ('HOLLYSYS', 'iAS',4,5);(*结果为'HOLLiAS'*)
功能块（**FBD**）	

2.12.9 FIND——查找字符串指令

功能：在一个字符串中查找与另一字符串完全相同的内容。

指令格式：FIND(STR1,STR2)。输入 STR1 和 STR2 都是 STRING 类型，为输入字符 串，返回数据类型为 INT 型。指令功能为在第一个字符串 STR1 中查找与字符串 STR2 完全相同的部分，返回该相同部分在字符串 STR1 的起始位置。 若没有完全相同的部 分，输出结果为 0.

指令使用举例

变量定义	
	
编程语言	程 序
梯形图（**LD**）	
结构化文本（**ST**）	VarINT1 := FIND ('HOLLYSYS', 'SYS'); (*结果为 6*)
功能块（**FBD**）	

2.13 BCD 码转换指令（Util.lib）

BCD 码的一个字节包含 0 到 99 之间的整数。每个十进制位对应 4 位，十位数存储在 4-7 位，个位数存储在 0-3 位。BCD 码格式和 16 进制表达方式很相似，差别在于 BCD 字节值是 0-99，而 16 进制是 0-FF。

BCD 码用 4 位二进制数表示一个十进制数位，整个十进制数用一串 BCD 码来表示。例如， 十进制数 59 表示成 BCD 码为 0101 1001，但表示成二进制为 2#111011。十进制 51 转换成 BCD 码， 5 的二进制是 0101，1 的二进制是 0001，那么 51 转换 BCD 码为 0101 0001。

2.13.1 BCD_TO_INT——BCD 码转整型指令

功能：该指令将 BCD 码转为 INT 值。

输入 B：BYTE 型，输入 BCD 码的二进制形式（或者该二进制对应的十进制和十六 进制）。比如 BCD 码 49，表示为 2#100 1001 （或者对应 10#73、 16#49） ，则此处输 入 2#100 1001 （或者 10#73，16#49）；

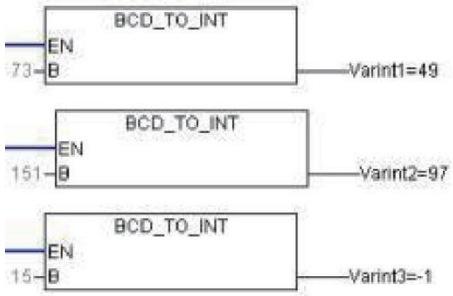
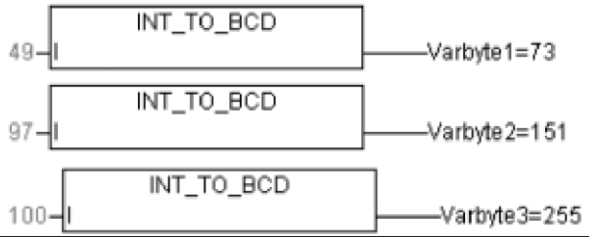
输出：INT 型，该 BCD 码所代表的实际值，如果输入的字节不是 BCD 码，输出是- 1。

指令使用举例

变量定义

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CONSTANT

	名称	地址	类型	初始值	注释
0001	Varint1		INT		
0002	Varint2		INT		
0003	Varint3		INT		

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

2.13.2 INT_TO_BCD——整型转 BCD 码指令

功能：将整数值转换成 BCD 码，当整数值不能转换成 BCD 码字节时，输出数值 255。

输入 I：INT 型，如果整数值为 49，则此处输入整型数据 49。

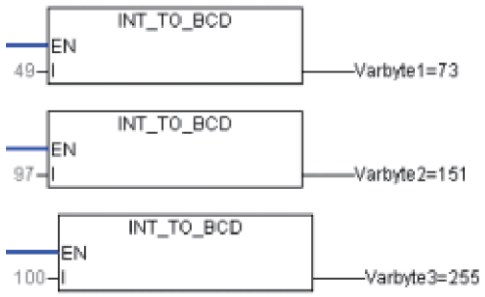
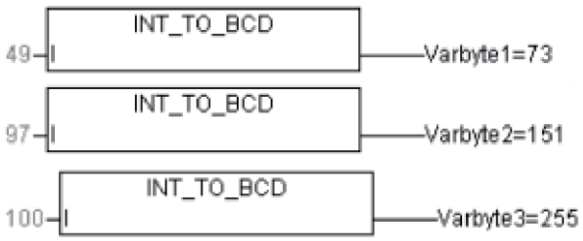
输出： BYTE 型， 转换完的 BCD 码的值，比如将 49 转换为 BCD 码为 2#100 1001， 则此处输出 2#100 1001 （或者该二进制对应的 10#73、 16#49）。

指令使用举例

变量定义

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CONSTANT

	名称	地址	类型	初始值	注释
0001	Varbyte1		BYTE		
0002	Varbyte2		BYTE		
0003	Varbyte3		BYTE		

编程语言	程 序
梯形图（**LD**）	
结构化文本（**ST**）	Varbyte1:=INT_TO_BCD(49); (*结果为 73*) Varbyte2:= INT_TO_BCD(97); (*结果为 151*) Varbyte3:= INT_TO_BCD(100); (*错误！ 输出：255*)
功能块（**FBD**）	

2.14 位/字节操作指令（Util.lib）

2.14.1 EXTRACT——位提取指令

功能：提取输入变量 X 二进制数的第 N 位（N=0, 1.. .）并输出该位数值。

输入/输出数据类型：

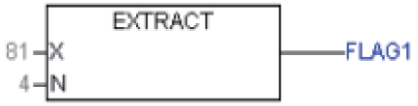
输入变量 X 是 DWORD 类型，N 是 BYTE 型；输出变量是 BOOL 类型。

指令使用举例

变量定义

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CONSTANT

	名称	地址	类型	初始值	注释
0001	FLAG1		BOOL		
0002	FLAG2		BOOL		

编程语言	程　序
梯形图（**LD**）	  (*结果 FLAG1=TRUE，FLAG2=TRUE *)
结构化文本（**ST**）	FLAG1:=EXTRACT(X:=81, N:=4); (*结果： TRUE，因为 81 的二进制数是 1010001，所以第四位是 1*) FLAG2:=EXTRACT(X:=33, N:=0); (*结果： TRUE，因为 33 的二进制数是 100001，所以第 0 位是 1*)
功能块（**FBD**	 (*结果 FLAG1=TRUE *)  (*结果 FLAG2=TRUE *)

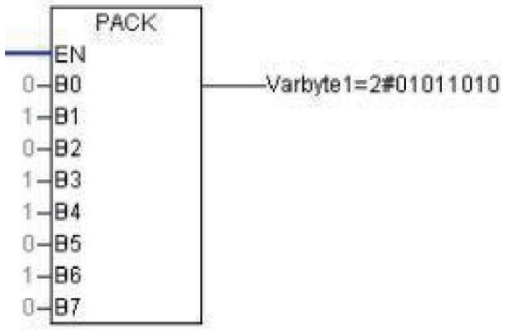
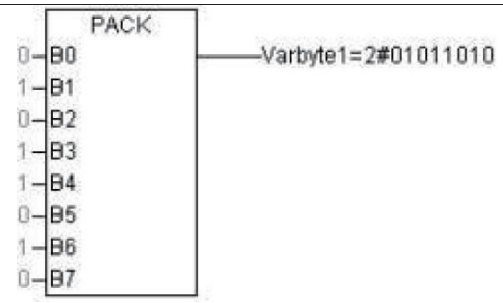
2.14.2 PACK——位整合指令

功能： 把输入位 B0、B1 、 ……、B7 合成为一个字节， 与这个指令相对应的指令是 UNPACK。

输入/输出数据类型：

输入 B0 、B1、 ……、B7 均为 BOOL 类型；输出数据为 BYTE 型。

指令使用举例

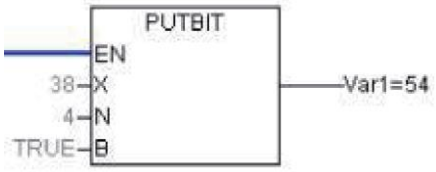
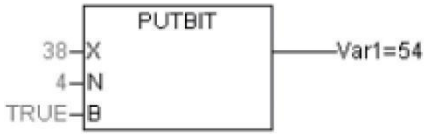
<table>
<tr><td colspan="2">变量定义
 
</td></tr>
<tr><td>编程语言</td><td>程 序</td></tr>
<tr><td>梯形图（LD）</td><td> 
</td></tr>
<tr><td>结构化文本 （ST）</td><td>Varbyte1:= PACK(0, 1,0, 1, 1,0, 1,0); (*结果为 2#01011010*)</td></tr>
<tr><td>功能块（FBD）</td><td> 
</td></tr>
</table>

2.14.3 PUTBIT——位赋值指令

功能：将输入变量 X 值的第 N 位（N=0, 1. . .） 赋值为 B，并输出 X 转变后的值。 输入/输出数据类型：

输入变量 X 为 DWORD 型、N 为 BYTE 型和 B 为 BOOL 型；输出为 DWORD 型。

指令使用举例

<table>
<tr><td colspan="2">变量定义</td></tr>
<tr><td colspan="2"> 
</td></tr>
<tr><td>编程语言</td><td>程 序</td></tr>
<tr><td>梯形图（LD）</td><td> 
</td></tr>
<tr><td>结构化文本（ST）</td><td>Var2:=38; (*二进制 100110*)
Var1:=PUTBIT(Var2,4,TRUE); (*结果： 54 = 2#110110*)</td></tr>
<tr><td>功能块（FBD）</td><td> 
</td></tr>
</table>

2.14.4 UNPACK——位拆分

功能： 将字节型的输入 B 拆分转换成 8 个 BOOL 类型的输出变量 B0, …,B7，与 PACK

指令相反。

输入/输出变量：

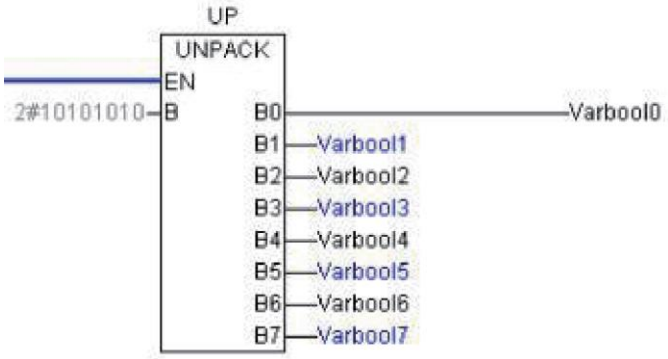
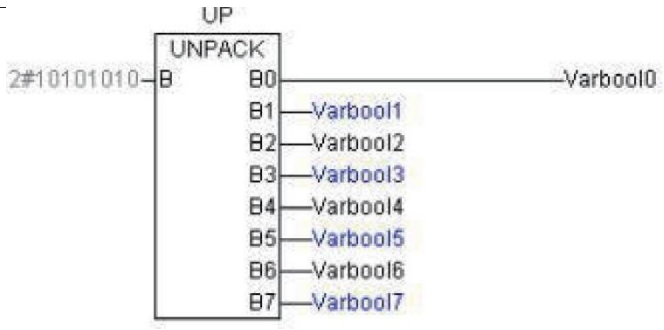
输入数据为 BYTE 型；输出 B0、B1 、 … …、B7 均为 BOOL 类型。

指令使用举例

变量定义

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CONSTANT

	名称	地址	类型	初始值	注释
0001	UP		UNPACK		
0002	Varbool0		BOOL		
0003	Varbool1		BOOL		
0004	Varbool2		BOOL		
0005	Varbool3		BOOL		
0006	Varbool4		BOOL		
0007	Varbool5		BOOL		
0008	Varbool6		BOOL		
0009	Varbool7		BOOL		

编程语言	程　序
梯形图（**LD**）	
结构化文本（**ST**）	UP(B:=2#10101010); Varbool0:=UP.B0; Varbool1:=UP.B1; Varbool2:=UP.B2; Varbool3:=UP.B3; Varbool4:=UP.B4; Varbool5:=UP.B5; Varbool6:=UP.B6; Varbool7:=UP.B7;
功能块（**FBD**）	

2.15 高等数学运算指令（Util.lib）

2.15.1 DERIVATIVE——微分

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

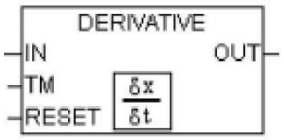
微分选代公式：

OUT = 3 * [IN(k) -IN(k -3)] +IN(k -1) -IN(k -2)

3 * TM(k -2) +4 * TM(k -1) +3 * TM(k)

k-3 、k-2、k- 1、k 为连续四次输入值的标记。

指令示例

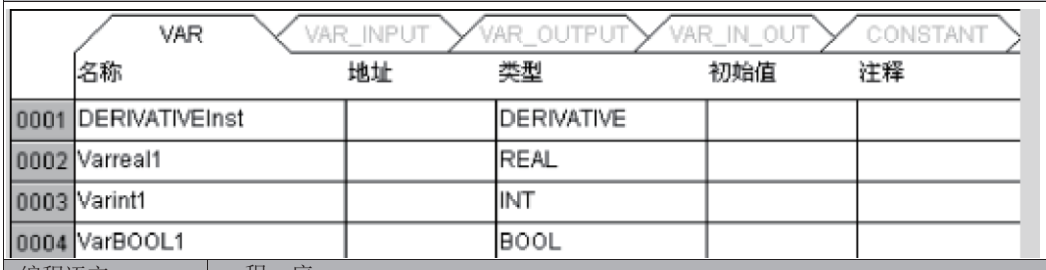
梯形图语言示例


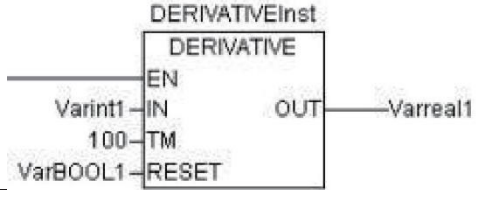
参数说明

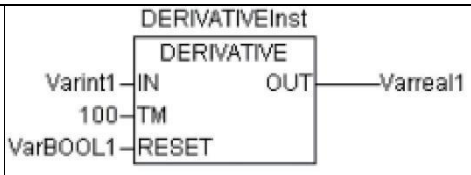
输入参数	数据类型	功能描述	参数值说明
IN	REAL	连续输入的变量	
TM	DWORD	微分时间	毫秒
RESET	BOOL	复位信号	值是 TRUE 时， 重新启动指令
输出参数	数据类型	功能描述	参数值说明
OUT	REAL	微分结果输出	

指令使用举例

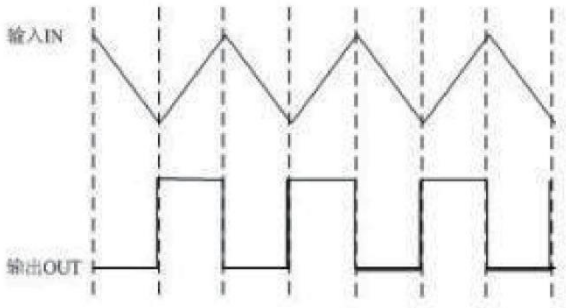
变量定义



编程语言	程 序
梯形图（**LD**）	
结构化文本（**ST**）	DERIVATIVEInst (IN:=Varint1,TM:= 100,RESET:=VarBOOL1); Varreal1:=DERIVATIVEInst.OUT;

功能块（**FBD**）	

输入输出对应关系如下图所示。



2.15.2 INTEGRAL——积分

功能：该指令对连续输入的变量进行积分运算。

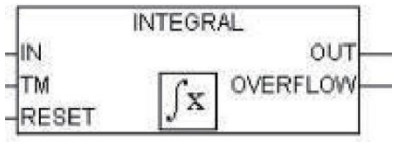
积分选代公式： A(k) =A(k -1) +TM * IN(k -1)

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

OUT(k) =(A(k) +B(k))/ 2

k-1 、k 为连续两次输入值的标记。

指令示例

梯形图语言示例


参数说明

输入参数	数据类型	功能描述	参数值说明
IN	REAL	连续输入的变量	
TM	DWORD	积分时间	
RESET	BOOL	复位信号	其值是 TRUE 时，重新启动指令
输出参数	数据类型	功能描述	参数值说明
OUT	REAL	积分结果输出	
OVERFLOW	BOOL	溢出标志	其值是 TRUE 时，说明计算溢出

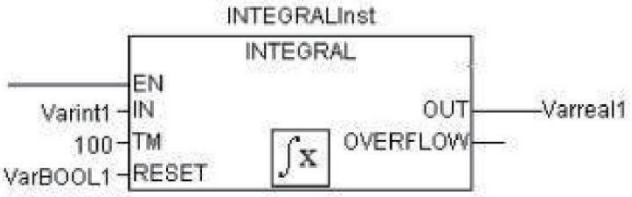
指令使用举

变量定义

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CONSTANT

	名称	地址	类型	初始值	注释
0001	INTEGRALInst		INTEGRAL		
0002	Varreal1		REAL		
0003	Varint1		REAL		
0004	VarBOOL1		BOOL		

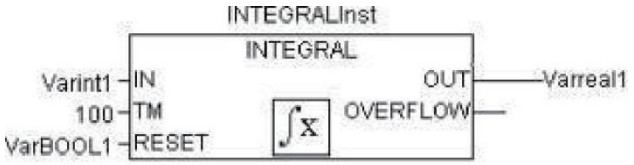
梯形图（**LD**）



结构化文本（**ST**）

```
INTEGRALInst(IN := Varint1, TM := 100, RESET := VarBOOL1);
Varreal1:=INTEGRALInst.OUT;
```

功能块（**FBD**）

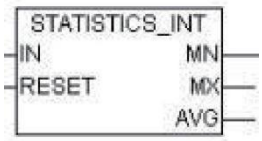


输入输出对应关系如下图所示。

2.15.3 STATISTICS_INT——整型统计

功能： 统计输入整型数据最小值、 最大值和平均值的计算。

指令示例

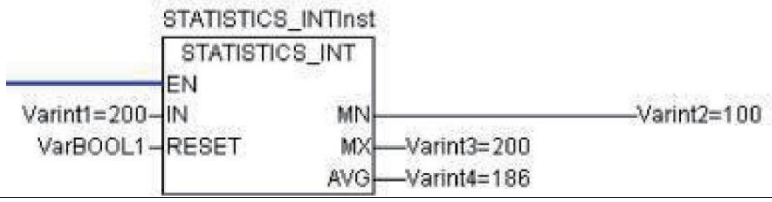
梯形图语言示例


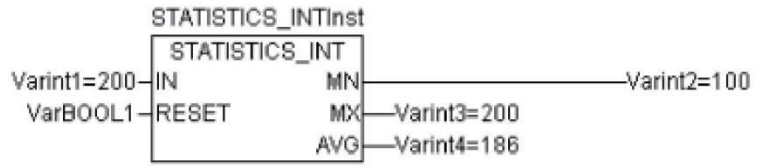
参数说明

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

指令使用举例

变量定义

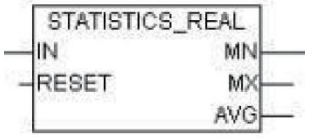
编程语言	程　序
梯形图（**LD**）	
结构化文本 （**ST**）	STATISTICS_INTInst(IN := Varint1, RESET := VarBOOL1); Varint3:=STATISTICS_INTInst.MX; Varint4:=STATISTICS_INTInst.AVG; Varint2:=STATISTICS_INTInst.MN;

功能块（**FBD**）	

2.15.4 STATISTICS_REAL——实型统计

功能：统计输入实型数据的最小值、最大值和平均值。

指令示例

梯形图语言示例


参数说明

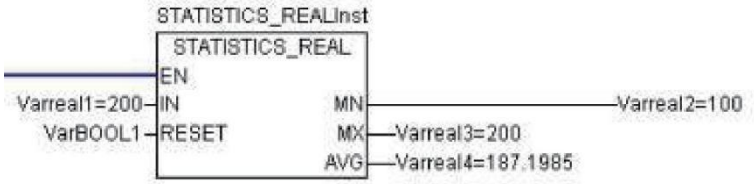
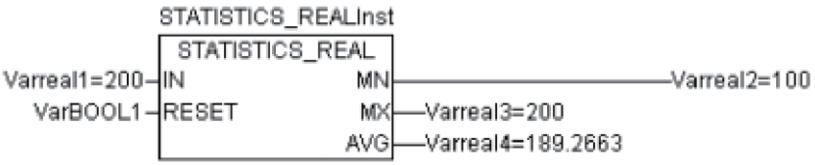
输入参数	数据类型	功能描述	参数值说明
IN	REAL	输入	
RESET	BOOL	初始化	其值是 TRUE 时，重新初始化
输出参数	数据类型	功能描述	参数值说明
MN	REAL	最小值	
MX	REAL	最大值	
AVG	REAL	平均值	

指令使用举例

变量定义

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CONSTANT

	名称	地址	类型	初始值	注释
0001	STATISTICS_REALInst		STATISTICS_REAL		
0002	VarBOOL1		BOOL		
0003	Varreal1		REAL		
0004	Varreal2		REAL		
0005	Varreal3		REAL		
0006	Varreal4		REAL		

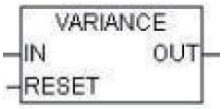
编程语言	程　序
梯形图（**LD**）	
结构化文本（**ST**）	STATISTICS_REALInst(IN :=Varreal1,RESET:=VarBOOL1); Varreal3:=STATISTICS_REALInst.MX; Varreal4:=STATISTICS_REALInst.AVG; Varreal2:=STATISTICS_REALInst.MN;
功能块（**FBD**）	

提示：

该指令与 STATISTICS_INT 功能相同，只是输入和输出数据类型为 REAL 型。

2.15.5 VARIANCE——平方偏差

功能：该指令计算变量输入值的平方偏差。标准偏差可以由平方偏差的平方根得到。　指令示例

梯形图语言示例


参数说明

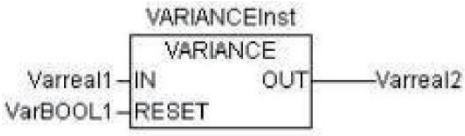
输入参数	数据类型	功能描述	参数值说明
IN	REAL	输入	
RESET	BOOL	复位	其值是 TRUE 时， 指令复位
输出参数	数据类型	功能描述	参数值说明
OUT	REAL	平方偏差	

指令使用举例

变量定义

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CONSTANT

	名称	地址	类型	初始值	注释
0001	VARIANCEInst		VARIANCE		
0002	VarBOOL1		BOOL		
0003	Varreal1		REAL		
0004	Varreal2		REAL		

编程语言	程　序
梯形图（**LD**）	
结构化文本（**ST**）	VARIANCEInst(IN := Varreal1, RESET := VarBOOL1); Varreal2:=VARIANCEInst.OUT;
功能块（**FBD**）	

2.16 控制器指令（Util.lib）

2.16.1 P—— 比例控制器

功能：该指令为比例控制器。

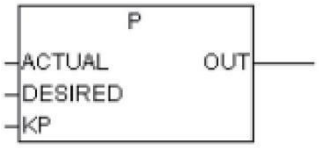
控制方程： OUT=ACTUAL+ （DESIRED-ACTUAL） *KP。

输入/输出数据类型：

输入的测量值 ACTUAL、设定值 DESIRED 和比例因子 KP 都是 REAL 型。输出值 OUT 是 REAL 型。

指令示例

梯形图语言示例



参数说明

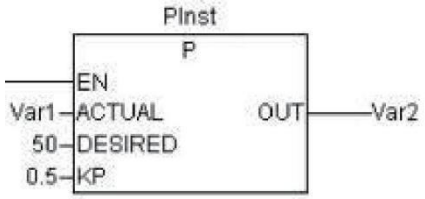
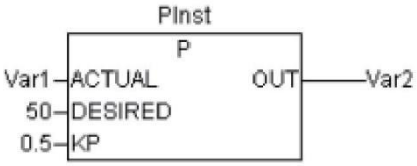
输入参数	数据类型	功能描述	参数值说明
ACTUAL	REAL	测量值	
DESIRED	REAL	设定值	
KP	REAL	比例系数	
输出参数	数据类型	功能描述	参数值说明
OUT	REAL	输出值	

指令使用举例

变量定义

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CONSTANT

	名称	地址	类型	初始值	注释
0001	PInst		P		
0002	Var1		REAL		
0003	Var2		REAL		

编程语言	程　序
梯形图（**LD**）	
结构化文本（**ST**）	PInst(ACTUAL := Var1, DESIRED := 50, KP := 0.5); Var2:=PInst.OUT;
功能块（**FBD**）	

2.16.2　PD—— 比例微分控制器

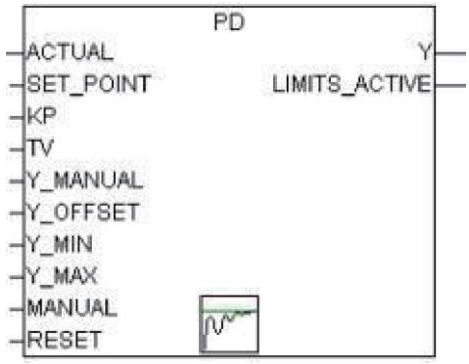
功能：该指令为比例微分控制器。

控制方程：Δ =SET _ POINT _ ACTUAL

Y = KP ＊ (Δ+ TV ＊ σ Δ/σ t) + Y_OFFSET

该指令会自动计算 $\frac{\sigma\Delta}{\sigma t}$ ，用户无需考虑。

指令示例

梯形图语言示例


参数说明

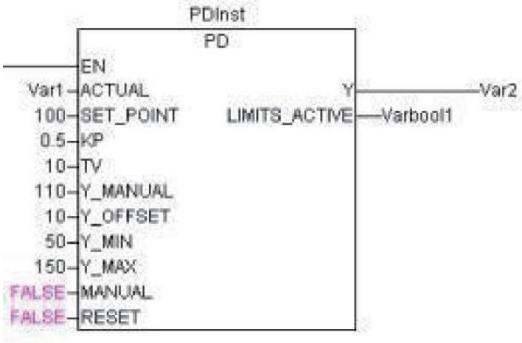
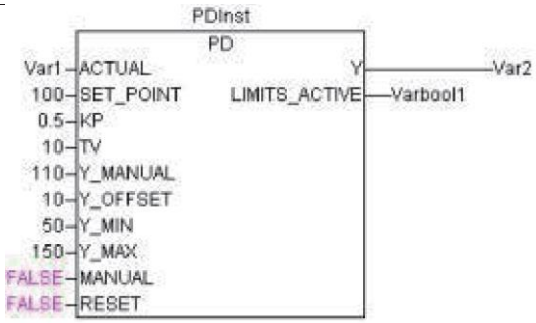
输入参数	数据类型	功能描述	参数值说明
ACTUAL	REAL	测量值	
SET_POINT	REAL	设定值	
KP	REAL	比例系数	
TV	REAL	微分时间	单位为秒（ s）
Y_MANUAL	REAL	手动值	MANUAL=TRUE 时， Y= Y_MANUAL
Y_OFFSET	REAL	输出值的偏移量	
Y_MIN	REAL	输出值的最小值	
Y_MAX	REAL	输出值的最大值	
MANUAL	BOOL	手自动选择	TRUE 时为手动调节， FALSE 时为自动调节
RESET	BOOL	重置	TRUE 时重置该控制器，正常运行时应置 FALSE
输出参数	数据类型	功能描述	参数值说明
Y	REAL	输出值	
LIMITS_ACTIVE	BOOL	输出超限标志	输出值 超限时等于 TRUE ， 即 超出（Y_MIN，Y_MAX）

指令使用举例

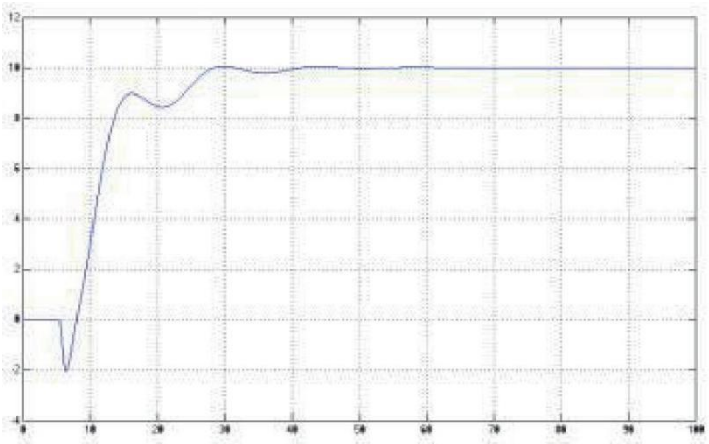
变量定义

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CONSTANT

	名称	地址	类型	初始值	注释
0001	PDinst		PD		
0002	Var1		REAL		
0003	Var2		REAL		
0004	Varbool1		BOOL		

编程语言	程　序
梯形图（**LD**）	
结构化文本（**ST**）	PDInst(ACTUAL := Var1, SET_POINT := 100, KP := 0.5, TV := 10, Y_MANUAL := 110, Y_OFFSET := 10, Y_MIN := 50, Y_MAX := 150, MANUAL := FALSE, RESET := FALSE); Varbool1:=PDInst.LIMITS_ACTIVE; Var2:=PDInst.Y;
功能块（**FBD**）	

调整波形如下图所示。



2.16.3 PID—— 比例积分微分控制器

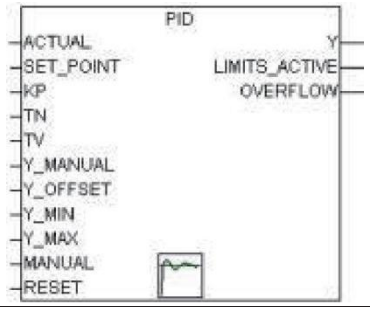
功能：该指令为比例积分微分控制器。 当 TV=0 时，PID 控制器作为 PI 控制器。 控制方程：

$$Y \quad KP * (\Delta + \frac{1}{TN} * \int \Delta(t)dt + TV * \frac{\sigma\Delta}{\sigma t}) + Y\_OFFSET$$

$$\Delta = SET\_POINT - ACTUAL$$

该指令会自动计算 $\int \Delta(t)dt$ 与 $\frac{\sigma\Delta}{\sigma t}$，用户无需考虑。

指令示例

梯形图语言示例


参数说明

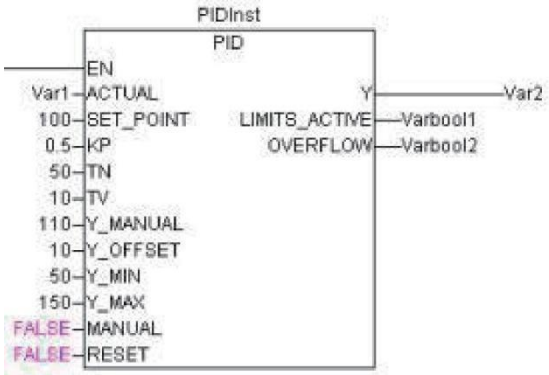
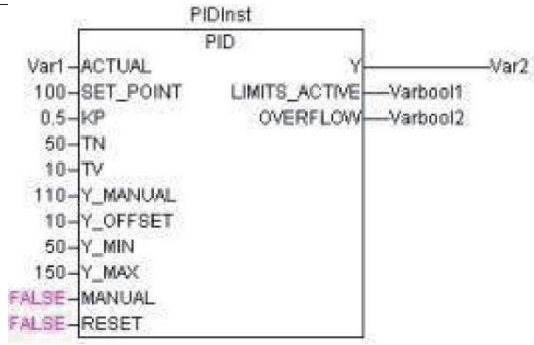
输入参数	数据类型	功能描述	参数值说明
ACTUAL	REAL	测量值	
SET_POINT	REAL	设定值	
KP	REAL	比例系数	
TN	REAL	积分时间	单位为秒（s）
TV	REAL	微分时间	单位为秒（s）
Y_MANUAL	REAL	手动值	MANUAL= TRUE 时， Y= Y_MANUAL
Y_OFFSET	REAL	输出值的偏移量	
Y_MIN	REAL	输出值的最小值	
Y_MAX	REAL	输出值的最大值	
MANUAL	BOOL	手自动选择	值为 TRUE 时为手动调节，值为 FALSE 时为自动调节
RESET	BOOL	重置	值为 TRUE 时重置该控制器，正常运行时应置 FALSE
输出参数	数据类型	功能描述	参数值说明
Y	REAL	输出值	
LIMITS_ACTIVE	BOOL	输出超限标志	输出 值超限时等 于 TRUE ， 即超出（Y_MIN，Y_MAX）
OVERFLOW	BOOL	输出溢出标志	积分溢出时等于 TRUE

指令使用举例

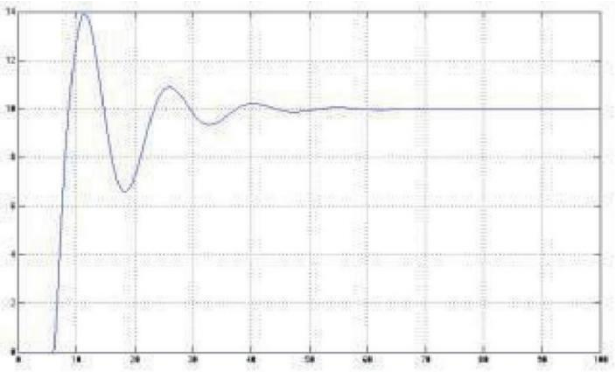
变量定义

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CONSTANT

	名称	地址	类型	初始值	注释
0001	PIDInst		PID		
0002	Var1		REAL		
0003	Var2		REAL		
0004	Varbool1		BOOL		
0005	Varbool2		BOOL		

编程语言	程　序
梯形图（**LD**）	
结构化文本（**ST**）	PIDInst(ACTUAL := Var1, SET_POINT := 100, KP := 0.5, TN := 50, TV := 10, Y_MANUAL := 110, Y_OFFSET := 10, Y_MIN := 50, Y_MAX := 150, MANUAL := FALSE, RESET := FALSE); Varbool1:=PIDInst.LIMITS_ACTIVE; Varbool2:=PIDInst.OVERFLOW; Var2:=PIDInst.Y;
功能块（**FBD**）	

调整波形如下图所示。



2.16.4 PID_FIXCYCLE—— 比例积分微分控制器

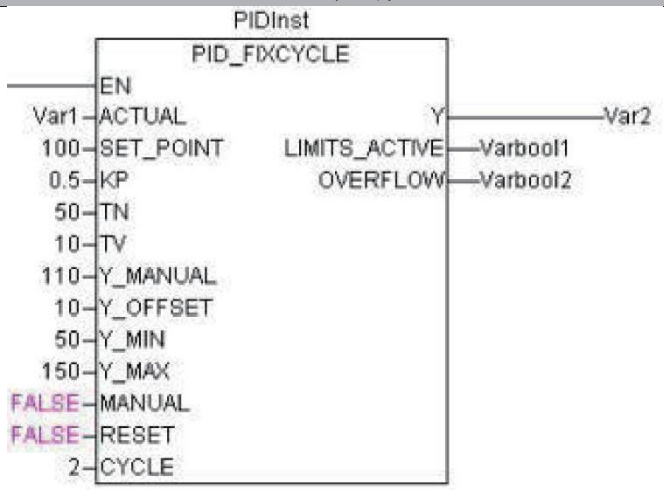
功能：该指令为比例积分微分控制器。与前面介绍过的 PID 控制器对比， 就是多了一 个采样周期的参数 CYCLE ，CYCLE 是一个 REAL 型的输入参数，是用来设定积分 和微分的时间步长， 单位是秒。控制方程和参数说明详见前面 PID 指令。

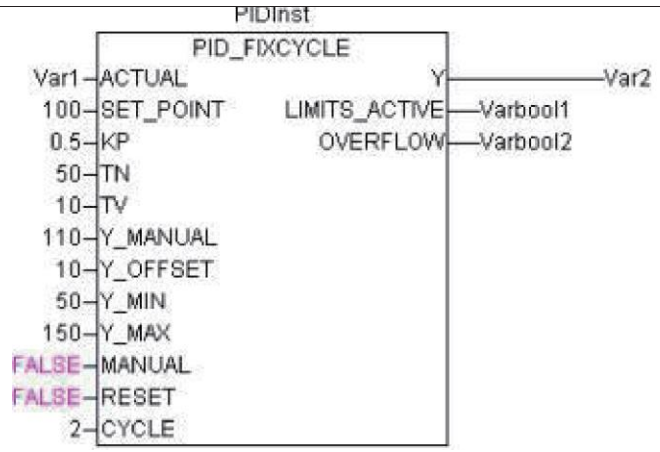
指令使用举例

变量定义

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CONSTANT

	名称	地址	类型	初始值	注释
0001	Var1		REAL		
0002	Var2		REAL		
0003	Varbool1		BOOL		
0004	Varbool2		BOOL		
0005	PIDInst		PID_FIXCYCLE		

编程语言	程 序
梯形图（**LD**）	
结构化文本（**ST**）	PIDInst(ACTUAL := Var1, SET_POINT := 100, KP := 0.5, TN := 50, TV := 10, Y_MANUAL := 110, Y_OFFSET := 10, Y_MIN := 50, Y_MAX := 150, MANUAL := FALSE, RESET := FALSE, CYCLE := 2); Varbool1:=PIDInst.LIMITS_ACTIVE; Varbool2:=PIDInst.OVERFLOW; Var2:= PIDInst.Y;

功能块（**FBD**）	

2.17 信号发生器指令（Util.lib）

2.17.1 BLINK——脉冲信号发生器

功能：该指令产生脉冲信号。脉冲信号发生器开始工作，输出高电平时间为

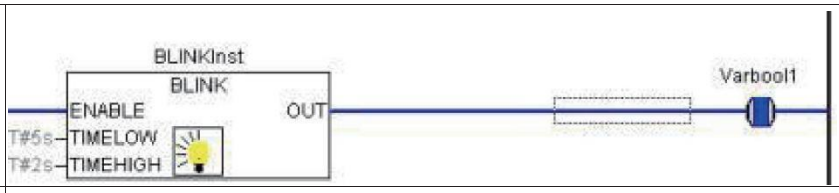
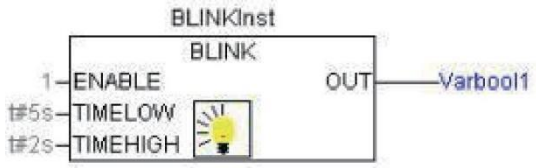
TIMEHIGH，输出低电平时间为 TIMELOW，周期循环输出。

参数说明

输入参数	数据类型	功能描述	参数值说明
ENABLE	BOOL	使能	TRUE 时，指令开始工作
TIMELOW	TIME	输出低电平时间	
TIMEHIGH	TIME	输出高电平时间	
输出参数	数据类型	功能描述	参数值说明
OUT	BOOL	脉冲信号输出值	

指令使用举例

变量定义				
VAR / VAR_INPUT / VAR_OUTPUT / VAR_IN_OUT / CONSTANT				
	名称	地址	类型	初始值	注释
0001	Varbool1		BOOL		
0002	PIDInst		PID_FIXCYCLE		
0003	BLINKInst		BLINK		
编程语言	程　序				

梯形图（**LD**）	
结构化文本（**ST**）	BLINKInst(TIMELOW := T#5s,TIMEHIGH := T#2s); Varbool1:=BLINKInst.OUT;
功能块（**FBD**）	

当指令执行时，OUT 如图 4-20- 1 所示波形输出。

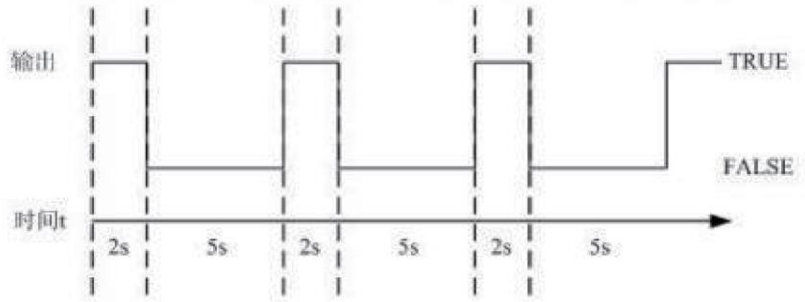


图 4-20-1

提示：

该指令输出的第一个电平为高是固定的，无法进行高低选择。

使能端断开，输出保持。

2.17.2 GEN——典型周期信号发生器

功能： 该指令用于生成典型的周期信号， 如：三角波、零起点三角波、 上升锯齿波、下降锯齿波、方波、正弦波和余弦波共七种。

输入参数	数据类型	功能描述	参数值说明
MODE	GEN_MODE	指定要产生的信号类型	直接在 MODE 处输入 TRIANGLE、TRIANGLE_POS、SAWTOOTH_RISE、SAWTOOTH_FALL 、RECTANGLE、SINUS 、COSINUS，则产生对应的波形
		TRIANGLE	三角波
		TRIANGLE_POS	零起点三角波
		SAWTOOTH_RISE	上升锯齿波
		SAWTOOTH_FALL	下降锯齿波
		RECTANGLE	方波
		SINUS	正弦波
		COSINU	余弦波
BASE	BOOL	循环方式选择	当 BASE 为 TRUE 时， 信号发生器与定义的循环周期有关。 当 BASE 为 FALSE 时，信号发生器与特定的发生的个数有关
PERIOD	TIME	循环周期	

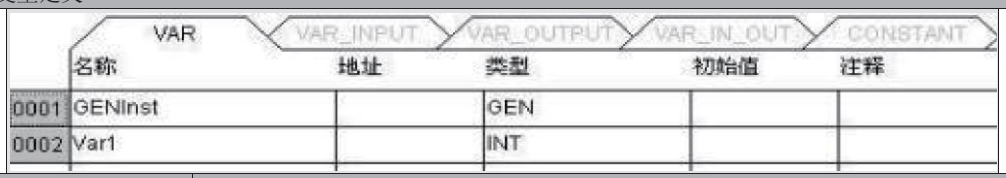
CYCLES	INT	发生的个数	
AMPLITUDE	INT	信号的振幅	
RESET	BOOL	初始化	当 RESET=TRUE 时，信号发生器被重新设置为 0
输出参数	数据类型	功能描述	参数值说明
OUT	INT	波形信号输出值	

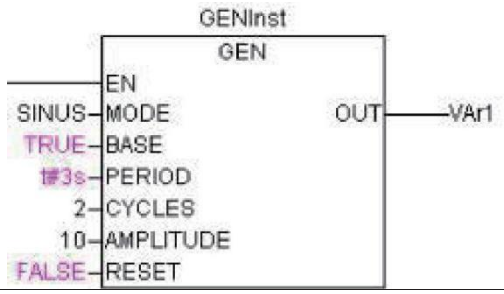
参数说明

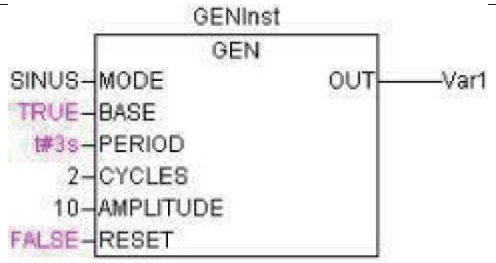
输入参数	数据类型	功能描述	参数值说明
MODE	GEN_MODE	指定要产生的信号类型	直接在 MODE 处输入 TRIANGLE、TRIANGLE_POS、SAWTOOTH_RISE、SAWTOOTH_FALL 、RECTANGLE、SINUS 、COSINUS，则产生对应的波形
		TRIANGLE	三角波
		TRIANGLE_POS	零起点三角波
		SAWTOOTH_RISE	上升锯齿波
		SAWTOOTH_FALL	下降锯齿波
		RECTANGLE	方波
		SINUS	正弦波
		COSINU	余弦波
BASE	BOOL	循环方式选择	当 BASE 为 TRUE 时， 信号发生器与定义的循环周期有关。 当 BASE 为 FALSE 时，信号发生器与特定的发生的个数有关
PERIOD	TIME	循环周期	
CYCLES	INT	发生的个数	
AMPLITUDE	INT	信号的振幅	
RESET	BOOL	初始化	当 RESET=TRUE 时，信号发生器被重新设置为 0
输出参数	数据类型	功能描述	参数值说明
OUT	INT	波形信号输出值	

指令使用举例

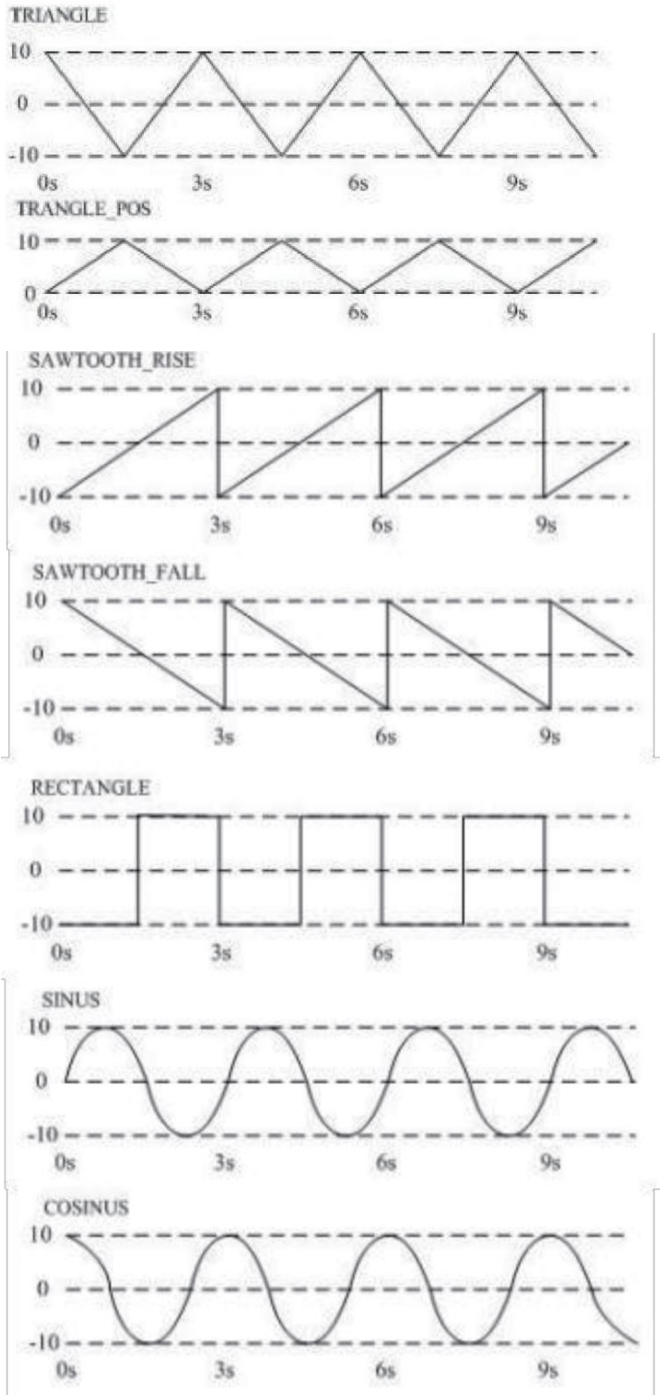
变量定义



编程语言	程　序
梯形图（**LD**）	
结构化文本（**ST**）	GENInst(MODE:=SINUS, BASE:=TRUE, PERIOD:=T#3s, CYCLES:=2, AMPLITUDE:= 10, RESET:=FALSE); Var1:=GENInst.OUT;

功能块（**FBD**）	

根据 MODE 处输入不同， 产生的波形如图 4-20-2 所示。



2.18 函数操纵器指令（Util.lib）

2.18.1 CHARCURVE——特征曲线

功能：输入的 POINT 类型数组 P[0..N- 1]在 XY 坐标图上定义了一条曲线，输入值 IN 为坐标图上的 X 轴上的点，输出值 OUT 为坐标图上该曲线所对应的 Y 轴的值。

参数说明

输入参数	数据类型	功能描述	参数值说明
IN	INT	输入坐标图上 X 轴的值	
N	BYTE	指定义曲线所使用数组中的点数	（2毫N毫11）
P	ARRAY[0..10] OF POINT		用来在 XY 坐标上定义曲线

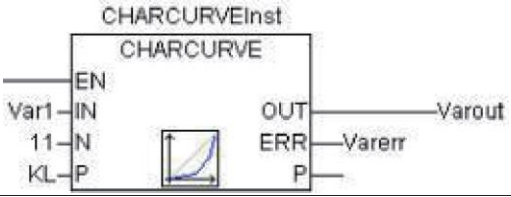
输出参数	数据类型	功能描述	参数值说明
OUT	INT	输出值	坐标图上曲线对应的 Y 轴的值
ERR	BYTE	显示错误类型	ERR=1：数组中的点 P[0]..P[N-1]中的 X 值有错误。 ERR=2：输入值 IN 不在 P[0].X 和 P[N-1].X 之间，即超出了数组定义曲线的 X 轴的范围。 此时 OUT 输出 IN 包含在限制值 P[0].Y 和 P[N-1].Y 之间所对应的数据。 ERR=4：输入N 小于 2，或者大于 11。
P	ARRAY[0..10] OF POINT	N 输出值	

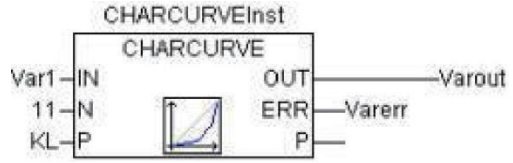
指令使用举例

变量定义

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CONSTANT | RETAIN

	名称	地址	类型	初始值	注释
0001	CHARCURVEInst		CHARCURVE		
0002	Var1		INT		
0003	Varout		INT		
0004	Varerr		BYTE		
0005	KL		ARRAY[0..10] OF POINT	(X:=0,Y:=0),(X:=250,Y:=50),(X:=500,Y:=150),(X:=750,Y:=400),7((X:=1000,Y:=1000));	

编程语言	程　序
梯形图（**LD**）	
结构化文本（**ST**）	CHARCURVEInst(IN := Var1, N := 11, P := KL); Varerr:=CHARCURVEInst.ERR; Varout:=CHARCURVEInst.OUT;

功能块（**FBD**）	

上例中， 当指令执行时， 根据输入值的变化，对应的输出值如图 4-21- 1 ，坐标曲线由数组 KL 确定， 输入 IN 为 X 轴上的值，输出 OUT 为该曲线对应的 Y 轴上的值。

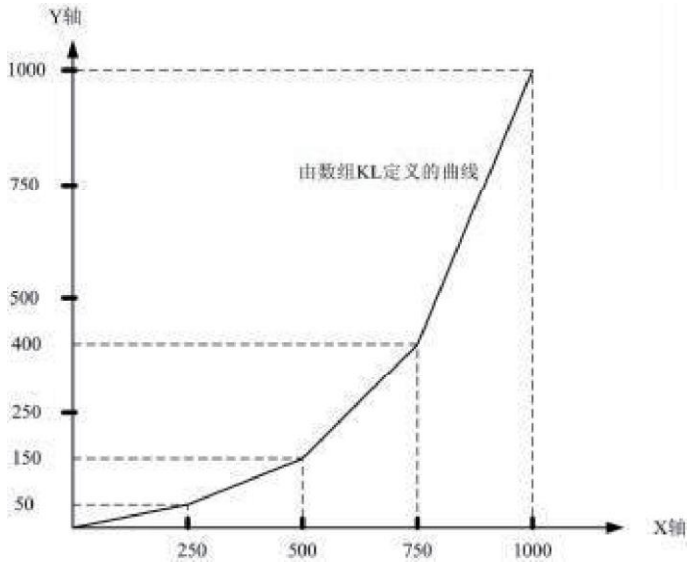


图 4-21-1

2.18.2 RAMP_INT——整型限速

功能：限制整型输入函数的升降速度。

参数说明

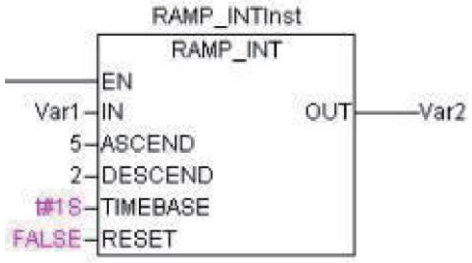
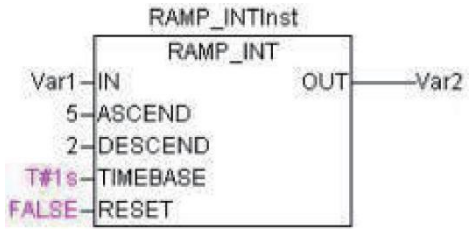
输入参数	数据类型	功能描述	参数值说明
IN	INT	目标值	若当前设定值大于先前值， 则按照设定的时间和上升速率进行加法运算，由 OUT 即时输出，若当前设定值小于先前值，则按照设定的时间和下降速率进行减法运算， 由 OUT 即时输出
ASCEND	INT	上升的增量	在规定时间（TIMEBASE） 内上升的数量
DESCEND	INT	下升的增量	在规定时间（TIMEBASE） 内下降的数量
TIMEBASE	TIME	时间基数	规定上升或者下降的速率
RESET	BOOL	初始化	设置为 TRUE 时，RAMP_INT 被重新初始化
输出参数	数据类型	功能描述	参数值说明
OUT	INT	即时数据输出	

指令使用举例

变量定义

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CONSTANT

	名称	地址	类型	初始值	注释
0001	RAMP_INTInst		RAMP_INT		
0002	Var1		INT		
0003	Var2		INT		

编程语言	程　序
梯形图（**LD**）	
结构化文本（**ST**）	RAMP_INTInst(IN := Var1, ASCEND := 5, DESCEND := 2, TIMEBASE := T#1000ms, RESET := FALSE); Var2:=RAMP_INTInst.OUT;
功能块（**FBD**）	

上例中，　当指令执行时，根据输入值 IN 的变化，　对应的输出值如图 4-21-2 所示。

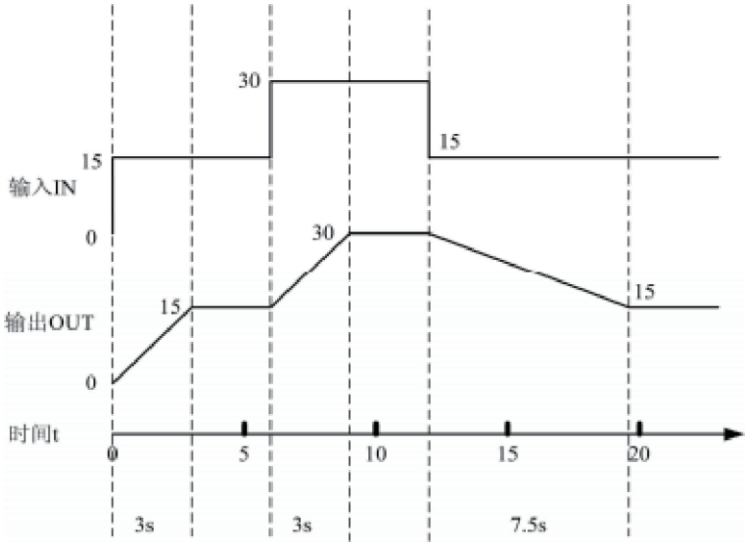


图 4-21-2

2.18.3 RAMP_REAL——实型限速

功能：限制实型输入函数的升降速度。

参数说明

输入参数	数据类型	功能描述	参数值说明
IN	REAL	目标值	若当前设定值大于先前值， 则按照设定的时间和上升速率进行加法运算，由 OUT 即时输出，若当前设定值小于先前值，则按照设定的时间和下降速率进行减法运算， 由 OUT 即时输出
ASCEND	REAL	上升的增量	在规定时间（TIMEBASE） 内上升的数量
DESCEND	REAL	下升的增量	在规定时间（TIMEBASE） 内下降的数量
TIMEBASE	TIME	时间基数	规定上升或者下降的速率
RESET	BOOL	初始化	设置为 TRUE 时，RAMP_INT 被重新初始化
输出参数	数据类型	功能描述	参数值说明
OUT	REAL	即时数据输出	

指令使用举例及输出图请参见 RAMP_INT 指令所述。

2.19 模拟量处理指令（Util.lib）

2.19.1 HYSTERESIS——滞后

功能：该指令的输入包括三个 INT 类型的数值 IN 、HIGH 和 LOW。如果 IN 小于下 限值 LOW，OUT 为 TRUE，保持至 IN 大于上限值 HIGH。此时， OUT 由 TRUE 变 为 FALSE，保持至 IN 小于下限值 LOW 。OUT 由 FALSE 变为 TRUE，保持至 IN 大 于上限值 HIGH ，OUT 变为 FALSE ，如此循环。

参数说明

输入参数	数据类型	功能描述	参数值说明
IN	INT	输入值	
HIGH	INT	上限值	
LOW	INT	下限值	
输出参数	数据类型	功能描述	参数值说明
OUT	BOOL	输出值	低下限值后为 TRUE，直到上限值后为恢复 FALSE

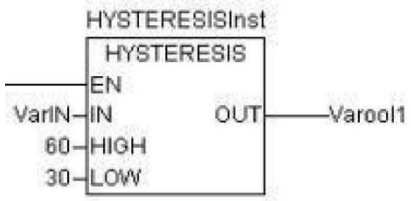
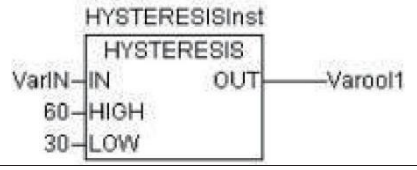
指令使用举例

变量定义

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CONSTANT

	名称	地址	类型	初始值	注释
0001	HYSTERESISInst		HYSTERESIS		
0002	Varool1		BOOL		
0003	VarIN		INT		

编程语言	程 序

梯形图（**LD**）	
结构化文本（**ST**）	HYSTERESISInst(IN := VarIN, HIGH := 60, LOW := 30); Varool1:=HYSTERESISInst.OUT;
功能块（**FBD**）	

上例中， 当指令执行时，根据输入值的变化，对应的输出值如图 4-22- 1 所示。

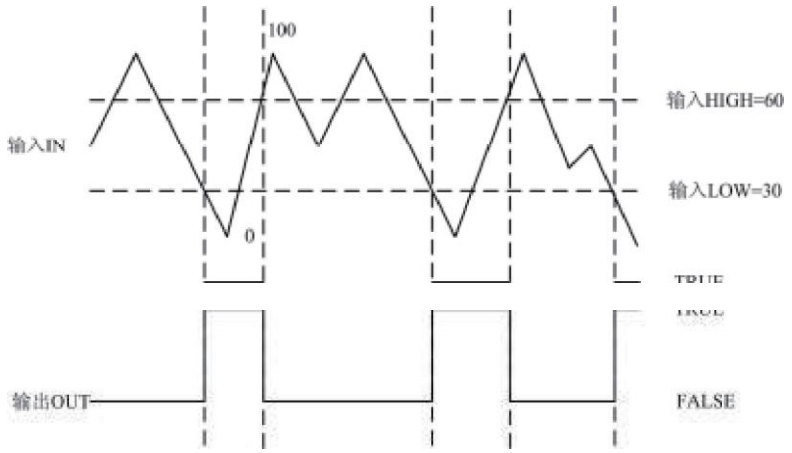


图 4-22-1

2.19.2 LIMITALARM——上下限报警

功能：如果 IN 超出上限 HIGH，则 O 为 TRUE ，U 和 IL 为 FALSE。如果 IN 低于下 限 LOW，则 U 为 TRUE ，O 和 IL 为 FALSE。如果 IN 在下限 LOW 和上限 HIGH 之间， 则 IL 为 TRUE ，O 和 U 为 FALSE。

参数说明

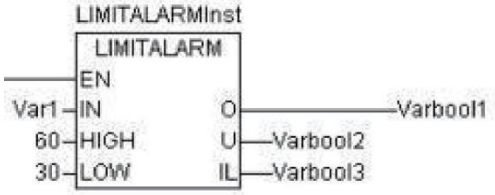
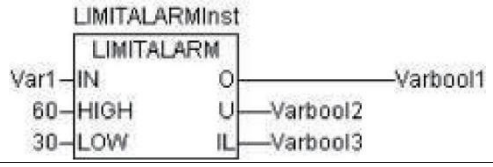
输入参数	数据类型	功能描述	参数值说明
IN	INT	输入值	
HIGH	INT	上限值	
LOW	INT	下限值	
输出参数	数据类型	功能描述	参数值说明
O	BOOL	输出值	
U	BOOL	输出值	
IL	BOOL	输出值	

指令使用举例

变量定义

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CONSTANT

	名称	地址	类型	初始值	注释
0001	LIMITALARMInst		LIMITALARM		
0002	Var1		INT		
0003	Varbool1		BOOL		
0004	Varbool2		BOOL		
0005	Varbool3		BOOL		

编程语言	程　序
梯形图（**LD**）	
结构化文本（**ST**）	LIMITALARMInst(IN := Var1, HIGH := 60, LOW := 30); Varbool2:=LIMITALARMInst.U; Varbool3:=　LIMITALARMInst.IL; Varbool1:=　LIMITALARMInst.O;
功能块（**FBD**）	

上例中，当指令执行时，根据输入值的变化，对应的输出值如下图 4-22-2 所示。

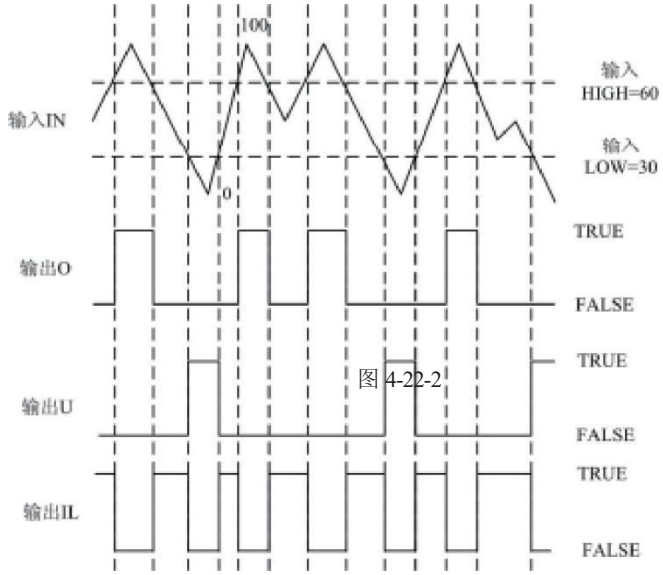


图 4-22-2

2.20 双稳态指令（Standard.lib）

2.20.1 SR——置位优先双稳态器

功能：置位双稳态触发器， 置位优先。

逻辑关系：Q1=(NOT RESET AND Q1) OR SET1

其中 SET1 为置位信号，RESET 为复位信号。

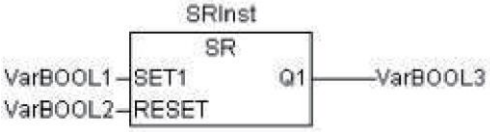
输入/输出数据类型：均为 BOOL 型。

指令使用举例

变量定义

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CONSTANT

	名称	地址	类型	初始值	注释
0001	SRInst		SR		
0002	VarBool1		BOOL		
0003	VarBool2		BOOL		
0004	VarBool3		BOOL		

编程语言	程　序
梯形图（**LD**）	 说明： VarBOOL3=（NOT VarBOOL2 AND VarBOOL3）OR VarBOOL1
结构化文本（**ST**）	SRInst(SET1:= VarBOOL1 , RESET:=VarBOOL2); VarBOOL3 := SRInst.Q1 ;
功能块（**FBD**）	

指令的真值表

指令	**SET1**	**RESET**	输出
SR	0	0	保持原状态
	1	0	1
	0	1	0
	1	1	1

2.20.2 RS——复位优先双稳态器

功能： 复位双稳态触发器，复位优先。

逻辑关系：Q1=NOT RESET1 AND (Q1 OR SET)

其中 SET 为置位信号，RESET1 为复位信号。

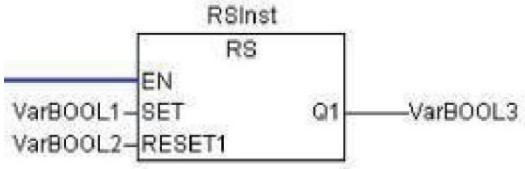
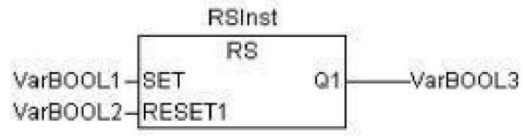
输入/输出数据类型：均为 BOOL。

指令使用举例

变量定义

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CONSTANT

	名称	地址	类型	初始值	注释
0001	RSInst		RS		
0002	VarBool1		BOOL		
0003	VarBool2		BOOL		
0004	VarBool3		BOOL		

编程语言	程 序
梯形图（**LD**）	 说明： VarBOOL3 =NOT VarBOOL2 AND (VarBOOL3 OR VarBOOL1)
结构化文本（**ST**）	RSInst(SET:= VarBOOL1 , RESET1:=VarBOOL2); VarBOOL3 := RSInst.Q1 ;
功能块（**FBD**）	

指令的真值表

指令	**SET**	**RESET1**	输出
RS	0	0	保持原状态
	1	0	1
	0	1	0
	1	1	0

2.21 触发器指令（Standard.lib）

触发器包含上升沿检测触发器 R_TRIG 和下降沿检测触发器 F_TRIG，分别用于检测上升沿 和下降沿。

2.21.1 R_TRIG——上升沿检测触发器

功能：用于检测上升沿。

逻辑关系： Q := CLK AND NOT M;

M := CLK;

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

输入/输出数据类型： CLK、Q 类型为 BOOL。

指令使用举例

变量定义

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CONSTANT

	名称	地址	类型	初始值	注释
0001	RTRIGInst		R_TRIG		
0002	VarBOOL1		BOOL		
0003	VarBOOL2		BOOL		

编程语言	程 序
梯形图（**LD**）	RTRIGInst R_TRIG CLK Q
结构化文本（**ST**）	RTRIGInst(CLK:= VarBOOL1); VarBOOL2 := RTRIGInst.Q;
功能块（**FBD**）	RTRIGInst R_TRIG VarBOOL1 CLK Q VarBOOL2

2.21.2 F_TRIG——下降沿检测触发器

功能：用于检测下降沿

逻辑关系：Q := NOT CLK AND NOT M;

M := NOT CLK;

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

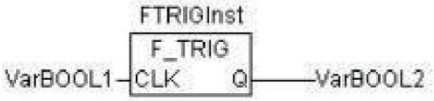
输入/输出数据类型：均为 BOOL 型。

指令使用举例

变量定义

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CONSTANT

	名称	地址	类型	初始值	注释
0001	FTRIGInst		F_TRIG		
0002	VarBOOL1		BOOL		
0003	VarBOOL2		BOOL		

编程语言	程 序
梯形图（**LD**）	
结构化文本（**ST**）	FTRIGInst(CLK:= VarBOOL1); VarBOOL2 := FTRIGInst.Q;
功能块（**FBD**）	

2.22 计数器（Standard.lib）

计数器包括递增计数器 CTU、递减计数器 CTD 和增减计数器 CTUD。

2.22.1 CTU——递增计数器

功能：递增计数器指令。

参数说明

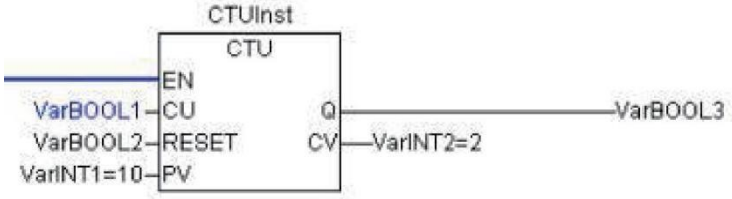
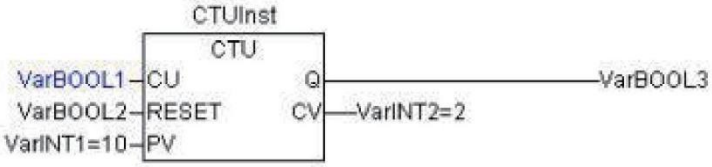
输入参数	数据类型	功能描述	参数值说明
CU	BOOL	计数输入	当 CU 有从 FALSE 到 TRUE 的上升沿， CV 加 1
RESET	BOOL	初始化	设置为 TRUE 时， CTU 被重新初始化
PV	WORD	计数器设定值	0-65535
输出参数	数据类型	功能描述	参数值说明
Q	BOOL	计数标志输出	当 CV 大于或等于上限 PV 时， Q 为 TRUE
CV	WORD	当前计数值	

指令使用举例

变量定义

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CONSTANT

	名称	地址	类型	初始值	注释
0001	CTUInst		CTU		
0002	VarBOOL1		BOOL		
0003	VarBOOL2		BOOL		
0004	VarBOOL3		BOOL		
0005	VarINT1		INT		
0006	VarINT2		INT		

编程语言	程　序
梯形图（**LD**）	
结构化文本（**ST**）	CTUInst(CU:= VarBOOL1, RESET:=VarBOOL2 , PV:= VarINT1); VarBOOL3 := CTUInst.Q; VarINT2 := CTUInst.CV;
功能块（**FBD**）	

2.22.2 CTD——递减计数器

功能：递减计数器。

参数说明

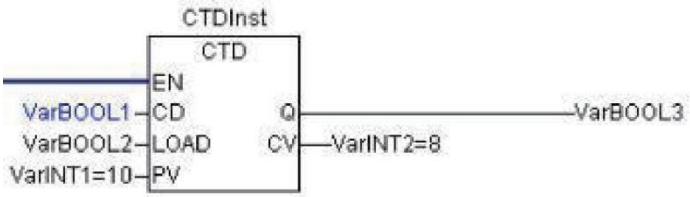
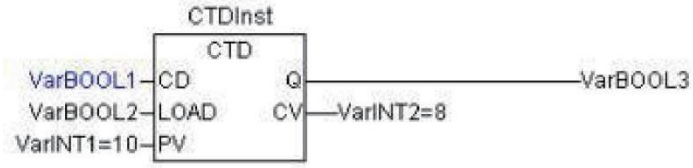
输入参数	数据类型	功能描述	参数值说明
CD	BOOL	计数输入	当 CD 有从 FALSE 到 TRUE 的上升沿， 若 CV 大于 0，CV 减 1（CV 的值不小于 0）
LOAD	BOOL	初始化	LOAD 为 TRUE 时， 计数变量 CV 装载为 PV
PV	WORD	计数器设定值	0-65535
输出参数	数据类型	功能描述	参数值说明
Q	BOOL	计数标志输出	当 CV 等于 0 时， Q 为 TRUE
CV	WORD	当前计数值	

指令使用举例

变量定义

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CONSTANT

	名称	地址	类型	初始值	注释
0001	CTDInst		CTD		
0002	VarBOOL1		BOOL		
0003	VarBOOL2		BOOL		
0004	VarBOOL3		BOOL		
0005	VarINT1		INT		
0006	VarINT2		INT		

编程语言	程　序
梯形图（**LD**）	
结构化文本（**ST**）	CTDInst(CD:= VarBOOL1, LOAD:=VarBOOL2 , PV:= VarINT1); VarBOOL3 := CTDInst.Q; VarINT2 := CTDInst.CV;
功能块（**FBD**）	

2.22.3 CTUD——递增递减计数器

功能：递增递减计数器。

参数说明

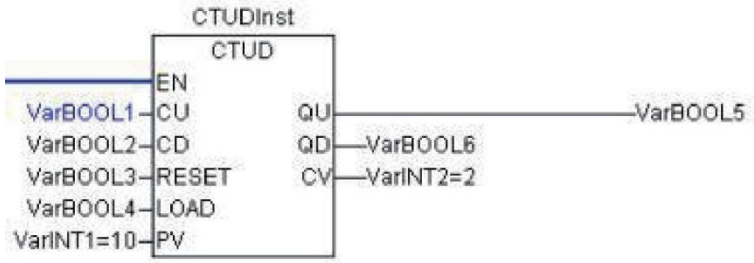
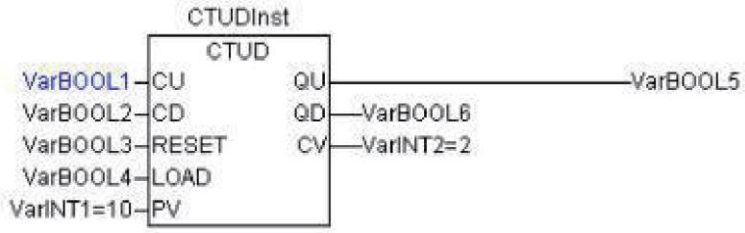
输入参数	数据类型	功能描述	参数值说明
CU	BOOL	计数输入	当 CU 有从 FALSE 到 TRUE 的上升沿， CV 加 1
CD	BOOL	计数输入	当 CD 有从 FALSE 到 TRUE 的上升沿， 若 CV 大于 0，CV 减 1（CV 的值不小于 0）
RESET	BOOL	复位输入	RESET 为 TRUE 时，计数变量 CV 初始化为 0
LOAD	BOOL	初始化	LOAD 为 TRUE 时， 计数变量 CV 装载为 PV
PV	WORD	计数器设定值	0-65535
输出参数	**数据类型**	**功能描述**	**参数值说明**
QU	BOOL	计数标志输出	当 CV 等于 PV 时， QU 为 TRUE
QD	BOOL	计数标志输出	当 CV 等于 0 时， QD 为 TRUE
CV	WORD	当前计数值	

指令使用举例

变量定义

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CONSTANT

	名称	地址	类型	初始值	注释
0001	CTUDInst		CUTD		
0002	VarBOOL1		BOOL		
0003	VarBOOL2		BOOL		
0004	VarBOOL3		BOOL		
0005	VarBOOL3		BOOL		
0006	VarBOOL4		BOOL		
0007	VarBOOL5		BOOL		
0008	VarBOOL6		BOOL		
0009	VarINT1		INT		
0010	VarINT2		INT		

编程语言	程　序
梯形图（**LD**）	
结构化文本（**ST**）	CTUDInst(CU:=VarBOOL1,CD:=VarBOOL2,RESET:=VarBOOL3, LOAD:=VarBOOL4,PV:=VarINT1) VarBOOL5 := CTUDInst.QU VarBOOL6 := CTUDInst.QD VarINT2 := CTUDInst.CV
功能块（**FBD**）	

2.23 定时器（Standard.lib）

定时器包括普通定时器 TP、通电延时定时器 TON、断电延时定时器 TOF 和实时时钟 RTC， 下面分别描述。

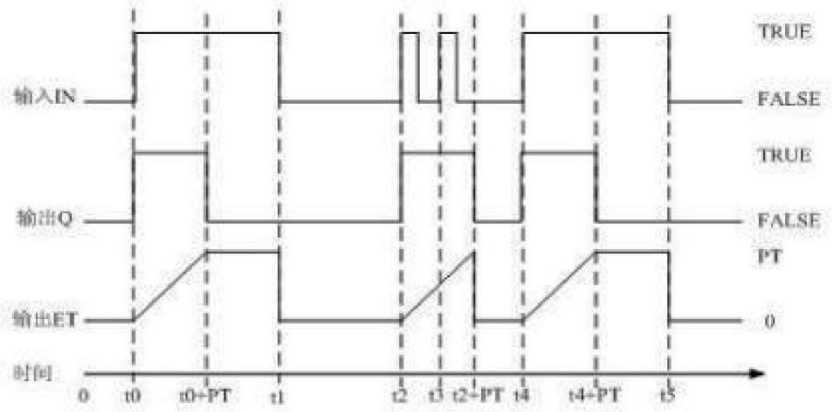
2.23.1 TP——普通定时器

功能：普通定时器。

参数说明

输入参数	数据类型	功能描述	参数值说明
IN	BOOL	定时器初始化信号	当 IN 变成 TRUE 时，ET 以毫秒计时直到 ET 等于 PT，然后 ET 保持常数
PT	TIME	定时时间值	
输出参数	数据类型	功能描述	参数值说明
Q	BOOL	定时器输出	若 IN 为 FALSE，则 Q 为 FALSE，ET 为 0。当 IN 为 TRUE 时，定时器开始工作，Q 为 TRUE，在 ET 小于等于 PT 前， IN 无效， ET 等于 PT 时， Q 为 FALSE
ET	TIME	当前时间值	当 IN 变成 TRUE 时，ET 以毫秒计时直到 ET 等于 PT，然后 ET 保持常数。 计时完毕后， 当 IN 为 FALSE 时， ET 等于 0

TP 时间顺序图

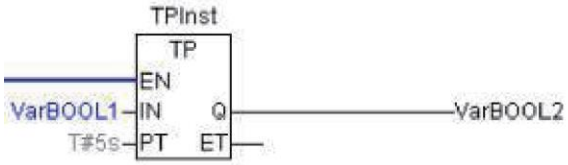
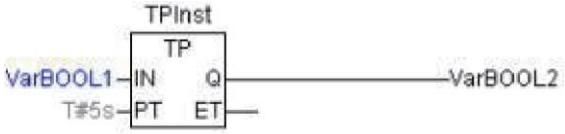


指令使用举例

变量定义	

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CONSTANT

	名称	地址	类型	初始值	注释
0001	TPInst		TP		
0002	VarBOOL1		BOOL		
0003	VarBOOL2		BOOL		

编程语言	程　序
梯形图（**LD**）	
结构化文本（**ST**）	TPInst(IN:= VarBOOL1,PT:= T#5s); VarBOOL2:=TPInst.Q;
功能块（**FBD**）	

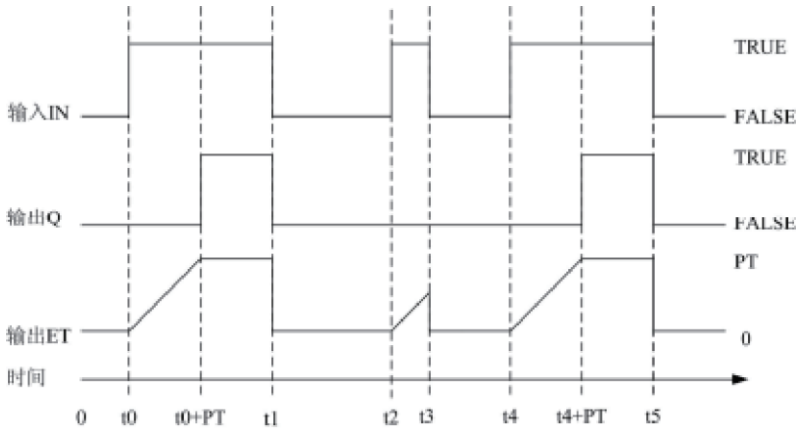
2.23.2　TON——通电延时定时器

功能：通电延时定时器。

参数说明

输入参数	数据类型	功能描述	参数值说明
IN	BOOL	定时器初始化信号	当 IN 变成 TRUE 时，ET 以毫秒计时直到 ET 等于 PT，然后 ET 保持常数
PT	TIME	定时时间值	
输出参数	数据类型	功能描述	参数值说明
Q	BOOL	定时器输出	当 IN 为 FALSE 时，Q 为 FALSE，ET 为 0。当 IN 为 TRUE 时， 定时器开始工作，ET 等于 PT 时，Q 为 TRUE
ET	TIME	当前时间值	当 IN 变成 TRUE 时，ET 以毫秒计时直到 ET 等于 PT，然后 ET 保持常数。无论何时， 当 IN 为 FALSE 时， ET 等于 0

TON 时间顺序图

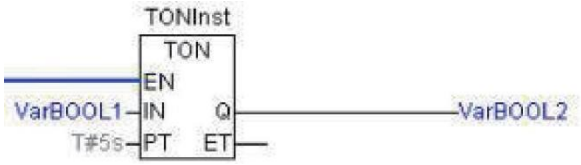
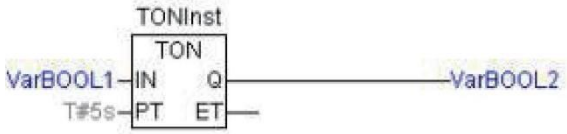


指令使用举例

变量定义

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CONSTANT

	名称	地址	类型	初始值	注释
0001	TONInst		TON		
0002	VarBOOL1		BOOL		
0003	VarBOOL2		BOOL		

编程语言	程　序
梯形图（**LD**）	
结构化文本（**ST**）	TONInst(IN:=VarBOOL1,PT:= T#5s);
功能块（**FBD**）	

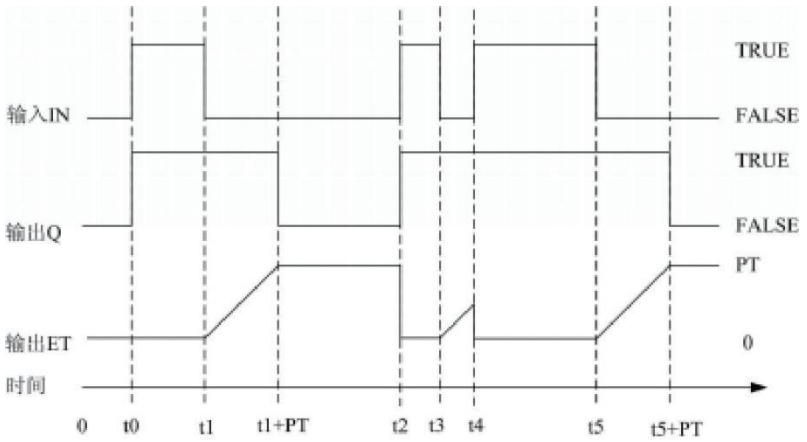
2.23.3　TOF——断电延时定时器

功能：断电延时定时器。

参数说明

输入参数	数据类型	功能描述	参数值说明
IN	BOOL	定时器初始化信号	当 IN 由 TRUE 变成 FALSE 时，ET 以毫秒计时直到 ET 等于 PT，然后 ET 保持常数
PT	TIME	定时时间值	
输出参数	数据类型	功能描述	参数值说明
Q	BOOL	定时器输出	当 IN 为FALSE 且 ET 等于 PT 时，Q 为 FALSE。否则，Q 为 TRUE
ET	TIME	当前时间值	当 IN 为 FALSE 时， ET 以毫秒计
			数直到 ET 等于 PT，然后 ET 保持常数。无论何时，当 IN 为 TRUE 时， ET 等于 0，Q 为 TRUE

TOF 时间顺序图

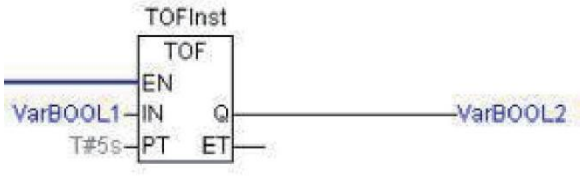
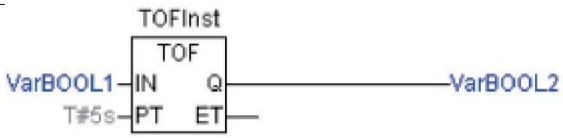


指令使用举例

变量定义

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CONSTANT

	名称	地址	类型	初始值	注释
0001	TOFInst		TOF		
0002	VarBOOL1		BOOL		
0003	VarBOOL2		BOOL		

编程语言	程　序
梯形图（**LD**）	
结构化文本（**ST**）	TOFInst(IN:= VarBOOL1,PT:=T#5s); VarBOOL2 :=TOFInst.Q;
功能块（**FBD**）	

2.23.4　RTC——实时时钟

功能：在给定时间启动，返回当前日期和时间。

参数说明

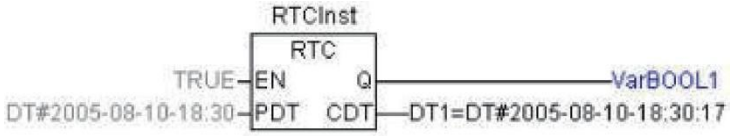
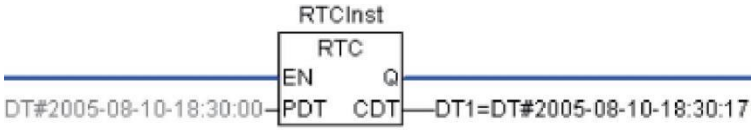
输入参数	数据类型	功能描述	参数值说明
EN	BOOL	使能信号	启动该指令
PDT	DT	时间基值	
输出参数	**数据类型**	**功能描述**	**参数值说明**
Q	BOOL	定时器输出	EN 为 FALSE 时， Q 为 FALSE，EN 为 TRUE 时， Q 为 TRUE
CDT	DT	当前时间值	EN 为 FALSE 时，CDT 为 1970-01-01-00:00:00，EN 为 TRUE 时， CDT 从 PDT 中的时间开始计时

指令使用举例

变量定义

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CONSTANT

	名称	地址	类型	初始值	注释
0001	RTCInst		RTC		
0002	DT1		DT		
0003	VarBOOL1		BOOL		

编程语言	程　序
功能块（**FBD**）	
结构化文本（**ST**）	RTCInst(PDT:=DT#2005-08-10-18:30:31); DT1:=RTCInst.CDT; VarBOOL1:=RTCInst.Q;
梯形图（**LD**）	

第三章： 扩展指令

3.1日期时间指令

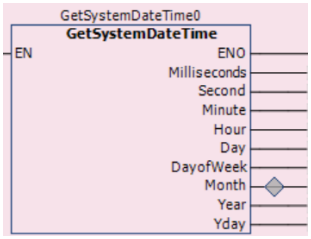
指令列表

指令类别	名称	FB/FC	功能
时间和日期	GetSystemTime	FB	获取系统当前时间ms/us/ns
	SetSystemDataTime	FB	设置系统当前时区、日期
	GetSystemDateTime	FB	获取系统当前时区、日期
	ADD_TOD_TIME	FC	时刻和时间的加法
	ADD_DT_TIME	FC	日期时刻和时间的加法
	SUB_TOD_TIME	FC	时刻和时间的减法
	SUB_TOD_TOD	FC	时刻减法
	SUB_DATA_DATA	FC	日期减法
	SUB_DT_DT	FC	日期时刻减法
	SUB_DT_TIME	FC	日期时刻减去时间
	GETDaysOfMonth	FC	月的天数获取
	GETDaysOfWeek	FC	星期获取
	GetMonthFromDays	FC	从天数获取月份
	GetWeekOfYear	FC	周数获取
	MULTIME	FC	时间乘法
	DIVTIME	FC	时间除法
	CheckLeapYear	FC	闰年判断
	TruncTime	FC	时间舍去
	TruncDT	FC	日期时刻舍去
	DT_To_DateStruct	FC	时刻分解
	DATE_TO_Sec	FC	日期转化为秒
	DateStruct_To_DT	FC	时刻组合
	GetSystemDate_SDT	FC	获取当前系统时间结构体
	DT_TO_SEC	FC	日期转换为秒
	SEC_TO_DT	FC	秒数转换为日期时刻
	TIME_TO_SEC_MS_NS	FC	时间转换为日期时刻
	Nanoseconds_To_Time	FC	纳秒转换为时间
	Sec_To_Time	FC	秒数转换为时间
	Sec_To_Tod	FC	秒数转换为时刻
	Tod_To_Sec	FC	时刻转换为秒数

	Sec_To_Date	FC	秒数转换为日期
	TruncTod	FC	时刻舍去

3.1.1 GetSystemDateTime 获取系统时间

①指令格式

指令	功能	FB/FC	LD 表现	ST表现
GetSystemDateTime	获取系统时间	FB		GetSystemDateTime_0(Milliseconds=> , Second=> , Minute=> , Hour=> , Day=> , DayofWeek=> , Month=> , Year=> , Yday=>);

②相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Milliseconds	毫秒	UINT	0~999	-	毫秒
Second	秒	UINT	0~59	-	秒
Minute	分	UINT	0~59	-	分
Hour	时	UINT	0~23	-	时
Day	天	UINT	1~31	-	天
DayofWeek	周几	STRING(5)	1~7	-	周几
Month	月	UINT	1~12	-	月
Year	年	UINT	1970~2038	-	年
Yday	当年第几天	UDINT	1~365	-	当年第几天

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Milliseconds	毫秒	UINT	0~999	-	毫秒
Second	秒	UINT	0~59	-	秒
Minute	分	UINT	0~59	-	分
Hour	时	UINT	0~23	-	时
Day	天	UINT	1~31	-	天
DayofWeek	周几	STRING(5)	1~7	-	周几
Month	月	UINT	1~12	-	月
Year	年	UINT	1970~2038	-	年
Yday	当年第几天	UDINT	1~365	-	当年第几天

数据类型

	布尔	位串	整数	实数	时刻、持续时间 日期、字符串

	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Milliseconds							√													
Second							√													
Minute							√													
Hour							√													
Day							√													
DayofWeek																				√
Month							√													
Year							√													
Yday								√												

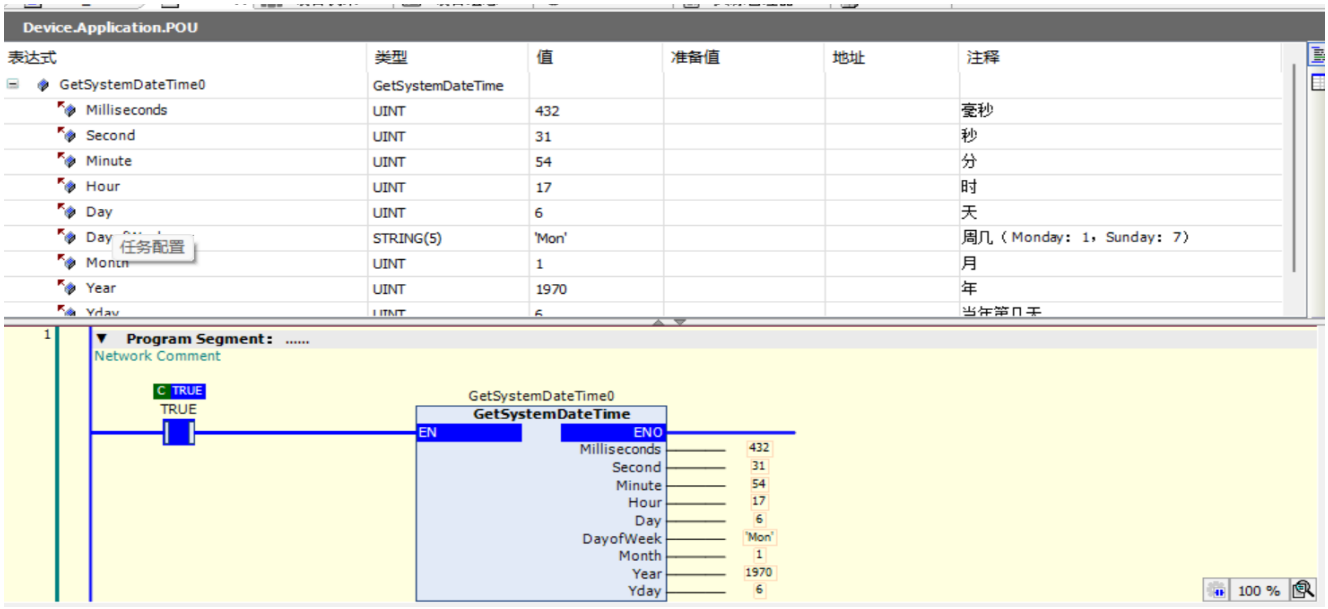
③功能说明

读取当前系统时间，以时间格式输出

④程序示例

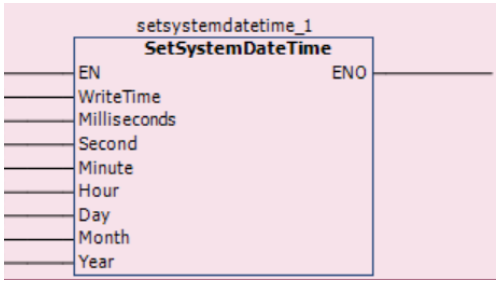
ST语言

LD语言



3.1.2 SetSystemDateTime 设置系统当前时区，日期

①指令格式

指令	功能	FB/FC	LD 表现	ST表现
SetSystemDateTime	设置系统当前时区，日期	FB		SetSystemDateTime_0(WriteTime:= , Milliseconds:= , Second:= , Minute:= , Hour:= , Day:= , Month:= , Year:=);

②相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
WriteTime	写入设置时间	BOOL	[FALSE,TRUE]	FALSE	写入设置时间
Milliseconds	毫秒	UINT	0~999	-	毫秒
Second	秒	UINT	0~59	-	秒
Minute	分	UINT	0~59	-	分
Hour	时	UINT	0~23	-	时
Day	天	UINT	1~31	-	天
Month	月	UINT	1~12	-	月
Year	年	UINT	1970~2038	-	年

输出变量

输入变量	名称	数据类型	有效范围	初始值	描述
WriteTime	写入设置时间	BOOL	[FALSE,TRUE]	FALSE	写入设置时间
Milliseconds	毫秒	UINT	0~999	-	毫秒
Second	秒	UINT	0~59	-	秒
Minute	分	UINT	0~59	-	分
Hour	时	UINT	0~23	-	时
Day	天	UINT	1~31	-	天
Month	月	UINT	1~12	-	月
Year	年	UINT	1970~2038	-	年

数据类型

	布尔	位串				整数								实数		时刻、持续时间 日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
WriteTime	√																			
Milliseconds							√													
Second							√													
Minute							√													
Hour							√													
Day							√													
Month							√													
Year							√													

③功能说明

设置当前系统时间，以时间格式输出

④程序示例

ST语言

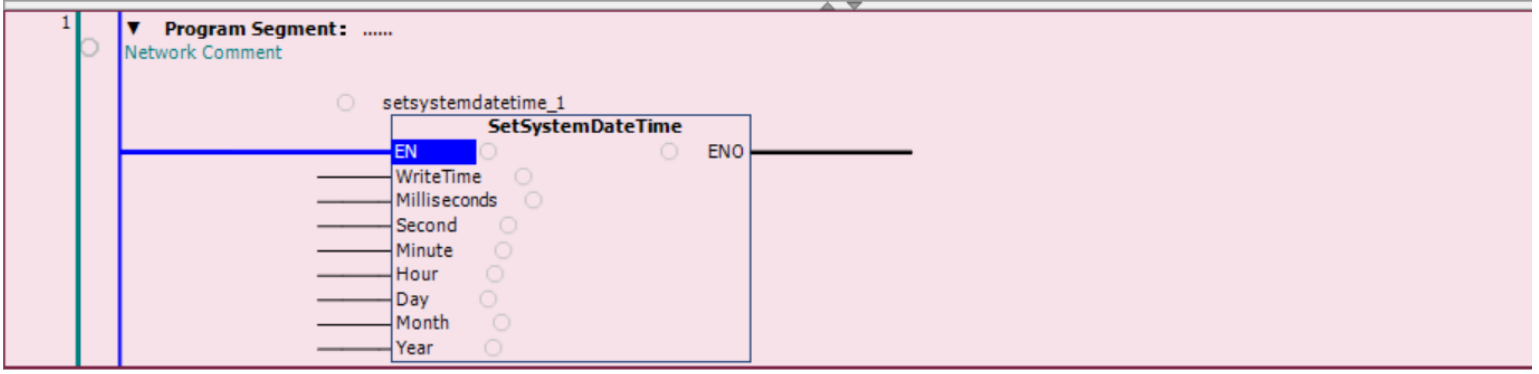
Device.Application.PLC_PRG

表达式	类型	值	准备值	地址	注释
SetSystemDateTime_0	SetSystemDateTime				
WriteTime	BOOL	FALSE			写入设置时间
Milliseconds	UINT	0			毫秒
Second	UINT	0			秒
Minute	UINT	0			分
Hour	UINT	0			时
Day	UINT	0			天
Month	UINT	0			月
Year	UINT	0			年

```
1  SetSystemDateTime_0(
2      WriteTime:= WriteTime,
3      Milliseconds:= Milliseconds,
4      Second:= Second,
5      Minute:= Minute,
6      Hour:= Hour,
7      Day:= Day,
8      Month:=Month ,
9      Year:=Year );
10 RETURN
```

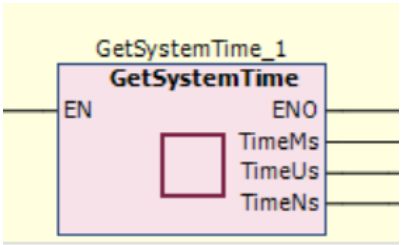
LD语言

表达式	类型	值	准备值	地址	注释
setsystemdatetime_1	SetSystemDateTime				
WriteTime	BOOL	FALSE			写入设置时间
Milliseconds	UINT	0			毫秒
Second	UINT	0			秒
Minute	UINT	0			分
Hour	UINT	0			时
Day	UINT	0			天
Month	UINT	0			月
Year	UINT	0			年



3.1.3 GetSystemTime 获取系统当前时间ms，us，ns

①指令格式

指令	功能	FB/FC	LD 表现	ST表现
GetSystemTime	获取系统当前时间ms，us，ns	FB		GetSystemTime_0(TimeMs=> , TimeUs=> , TimeNs=>);

②相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
TimeMs	系统时间，单位：MS	UDINT	-	-	系统时间，单位：MS
TimeUs	系统时间，单位：US	ULINT	-	-	系统时间，单位：US
TimeNs	系统时间，单位：NS	ULINT	-	-	系统时间，单位：NS

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
TimeMs	系统时间，单位：MS	UDINT	-	-	系统时间，单位：MS
TimeUs	系统时间，单位：US	ULINT	-	-	系统时间，单位：US
TimeNs	系统时间，单位：NS	ULINT	-	-	系统时间，单位：NS

数据类型

	布尔	位串				整数								实数		时刻、持续时间 日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
TimeMs												√								
TimeUs									√											
TimeNs									√											

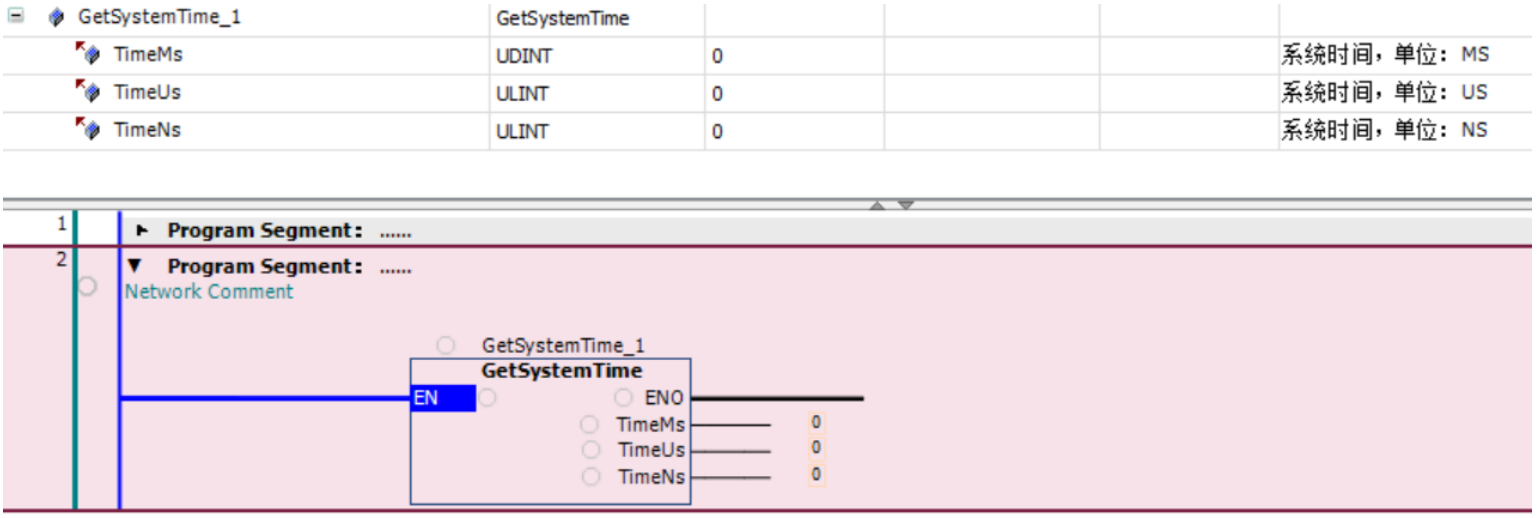
③功能说明

设置系统当前时间ms，ns，us

④程序示例

ST语言

LD语言



3.1.4 ADD_TOD_TIME 时刻和时间的加法

①指令格式

指令	功能	FB/FC	LD 表现	ST表现

ADD_TOD_TIME	时刻和时间的加法	FC		ADD_TOD_TIME(ADD_TOD_TIME=> , In1:= , In2:=);

②相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
In1	被加时刻	TOD	遵照数据类型	-	被加时刻
In2	相加时间	TIME	遵照数据类型	-	相加时间

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
ADD_TOD_TIME_1	相加的时间	TOD	遵照数据类型	–	相加的时间

数据类型

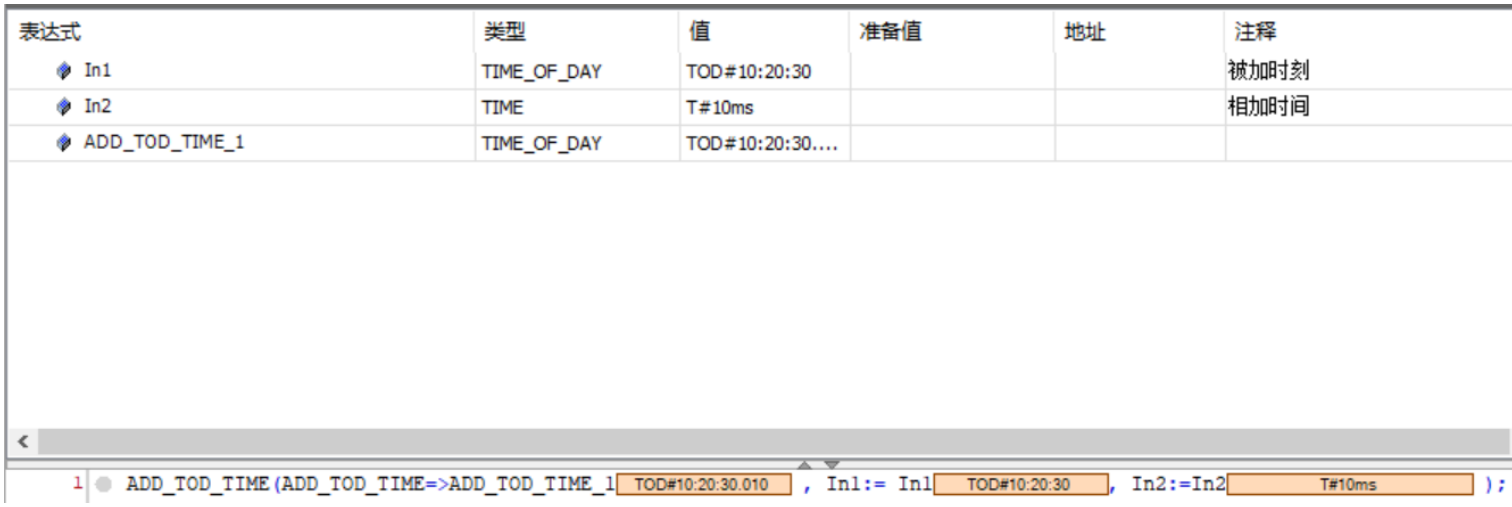
	布尔	位串				整数								实数		时刻、持续时间 日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In1																		√		
In2																√				
ADD_TOD_TIME_1																		√		

③功能说明

设置被加时刻和被加时间，输出结果

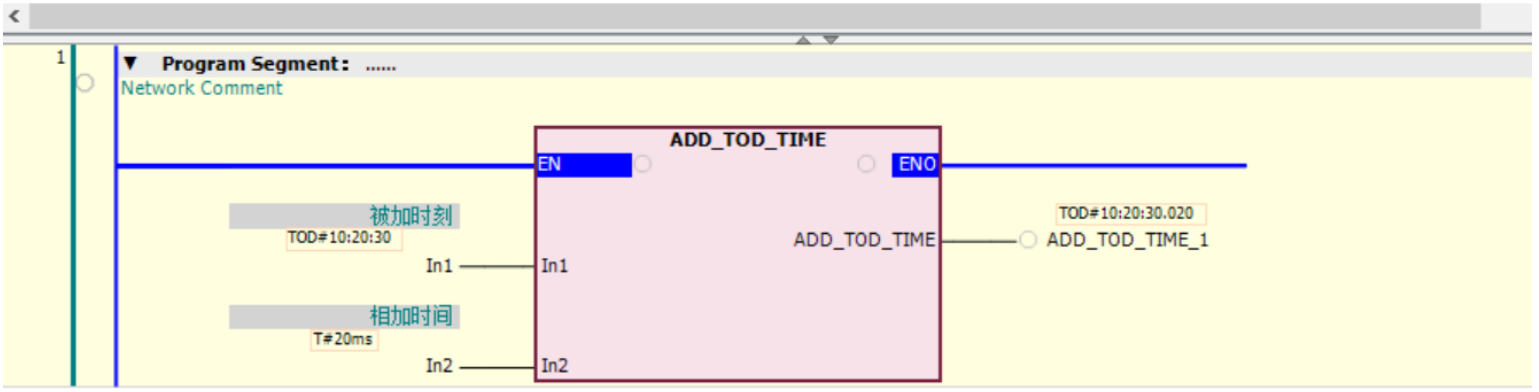
④程序示例

ST语言



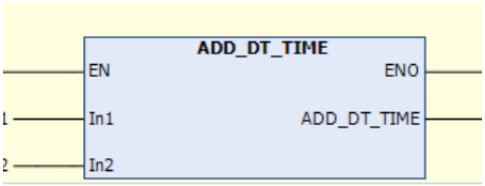
LD语言

表达式	类型	值	准备值	地址	注释
In1	TIME_OF_DAY	TOD#10:20:30			被加时刻
In2	TIME	T#20ms			相加时间
ADD_TOD_TIME_1	TIME_OF_DAY	TOD#10:20:30.020			



3.1.5 ADD_DT_TIME 日期时刻和时间的加法

①指令格式

指令	功能	FB/FC	LD 表现	ST表现
ADD_DT_TIME	日期时刻和时间的加法	FC		ADD_DT_TIME(ADD_DT_TIME=> , In1:= , In2:=);

②相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
In1	被加日期时刻	DT	遵照数据类型	-	被加日期时刻
In2	相加时间	TIME	遵照数据类型	-	相加时间

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
ADD_DT_TIME_1	日期时刻和时间的加法	DT	遵照数据类型	–	日期时刻和时间的加法

数据类型

	布尔	位串	整数	实数	时刻、持续时间 日期、字符串

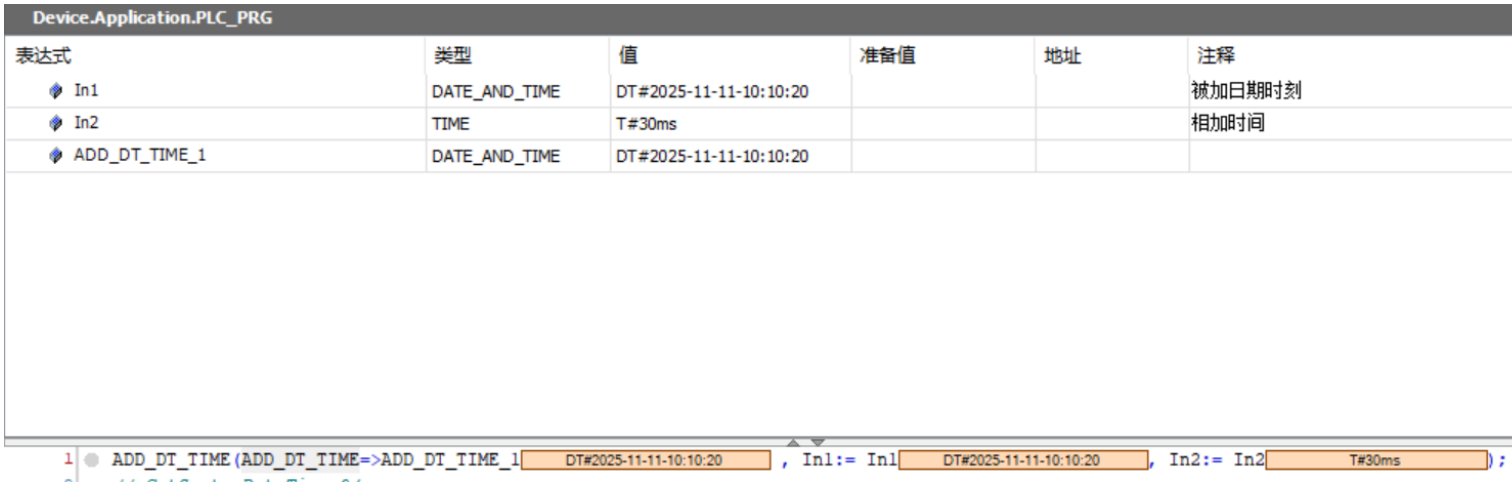
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In1																			√	
In2																√				
ADD_DT_TIME_1																			√	

③功能说明

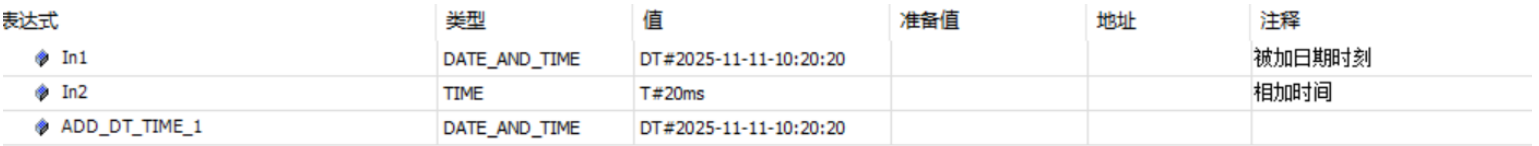
设置被加日期时刻和相加时间，输出日期时刻和时间的加法

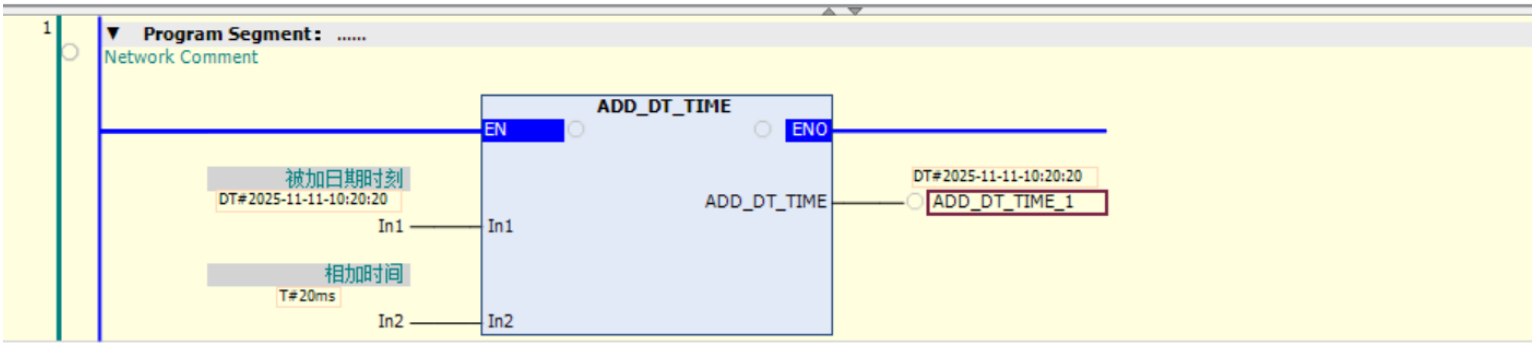
④程序示例

ST语言



LD语言

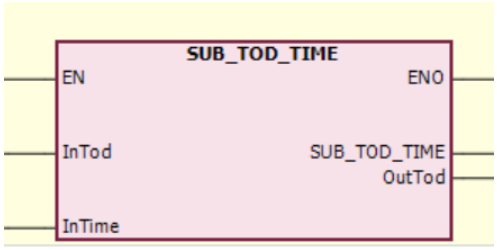




3.1.6 SUB_TOD_TIME 时刻和时间的减法

①指令格式

指令	功能	FB/FC	LD 表现	ST表现

SUB_TOD_TIME	设置 系统时间	FC		SUB_TOD_TIME(SUB_TOD_TIME=> , InTod:= , InTime:= , OutTod=>);

②相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
InTod	被减时刻	TOD	遵照数据类型	-	被减时刻
InTime	相减时间	TIME	遵照数据类型	-	相减时间

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
OutTod	相减结果时刻	TOD	遵照数据类型	–	相减结果时刻

数据类型

	布尔	位串				整数								实数		时刻、持续时间 日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
InTod																		√		
InTime																√				
OutTod																		√		

③功能说明

设置被减时刻和相减时间，输出相减结果

④程序示例

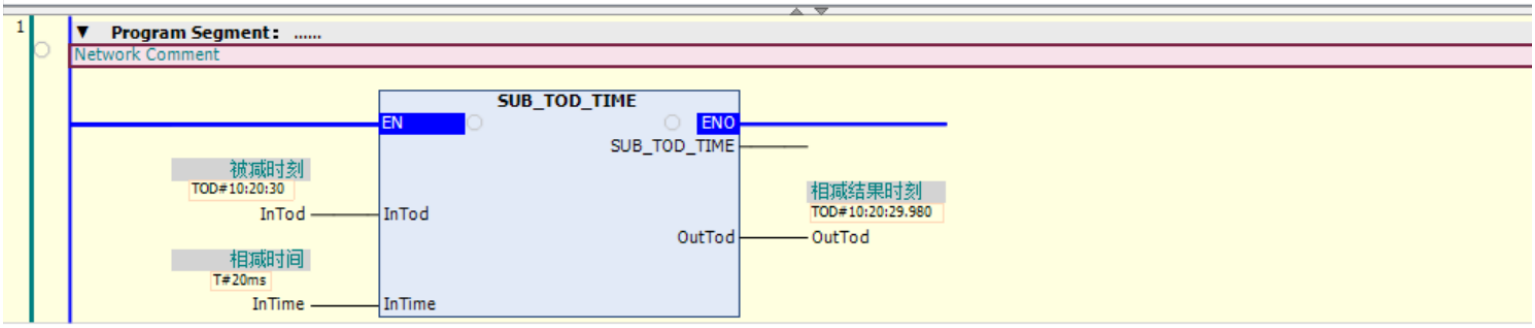
ST语言

表达式	类型	值	准备值	地址	注释
InTod	TIME_OF_DAY	TOD#10:10:20			被减时刻
InTime	TIME	T#20ms			相减时间
OutTod	TIME_OF_DAY	TOD#10:10:19.980			相减结果时刻

```
1  SUB_TOD_TIME(SUB_TOD_TIME=> , InTod:= InTod  TOD#10:10:20 , InTime:=InTime  T#20ms , OutTod=>OutTod  TOD#10:10:19.980 );
```

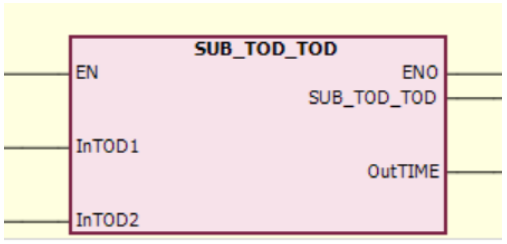
LD语言

表达式	类型	值	准备值	地址	注释
InTod	TIME_OF_DAY	TOD#10:20:30			被减时刻
InTime	TIME	T#20ms			相减时间
OutTod	TIME_OF_DAY	TOD#10:20:29.980			相减结果时刻



3.1.7 SUB_TOD_TOD 时刻减法

①指令格式

指令	功能	FB/FC	LD 表现	ST表现
SUB_TOD_TOD	设置 系统时间	FC		SUB_TOD_TOD(SUB_TOD_TOD=> , InTOD1:= , InTOD2:= , OutTIME=>);

②相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
InTOD1	被减时刻	TOD	遵照数据类型	-	被减时刻
InTOD2	相减时刻	TOD	遵照数据类型	-	相减时刻

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
OutTIME	相减结果时间	TIME	遵照数据类型	–	相减结果时间

数据类型

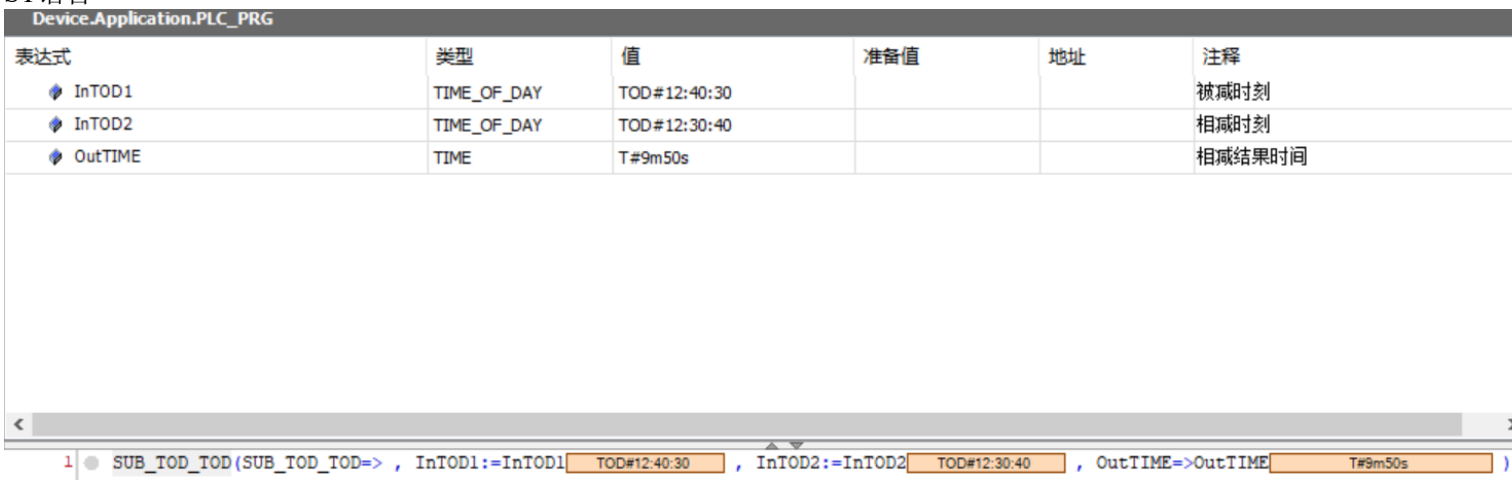
	布尔	位串				整数								实数		时刻、持续时间 日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
InTod1																		√		
InTOD2																		√		
OutTIME																√				

③功能说明

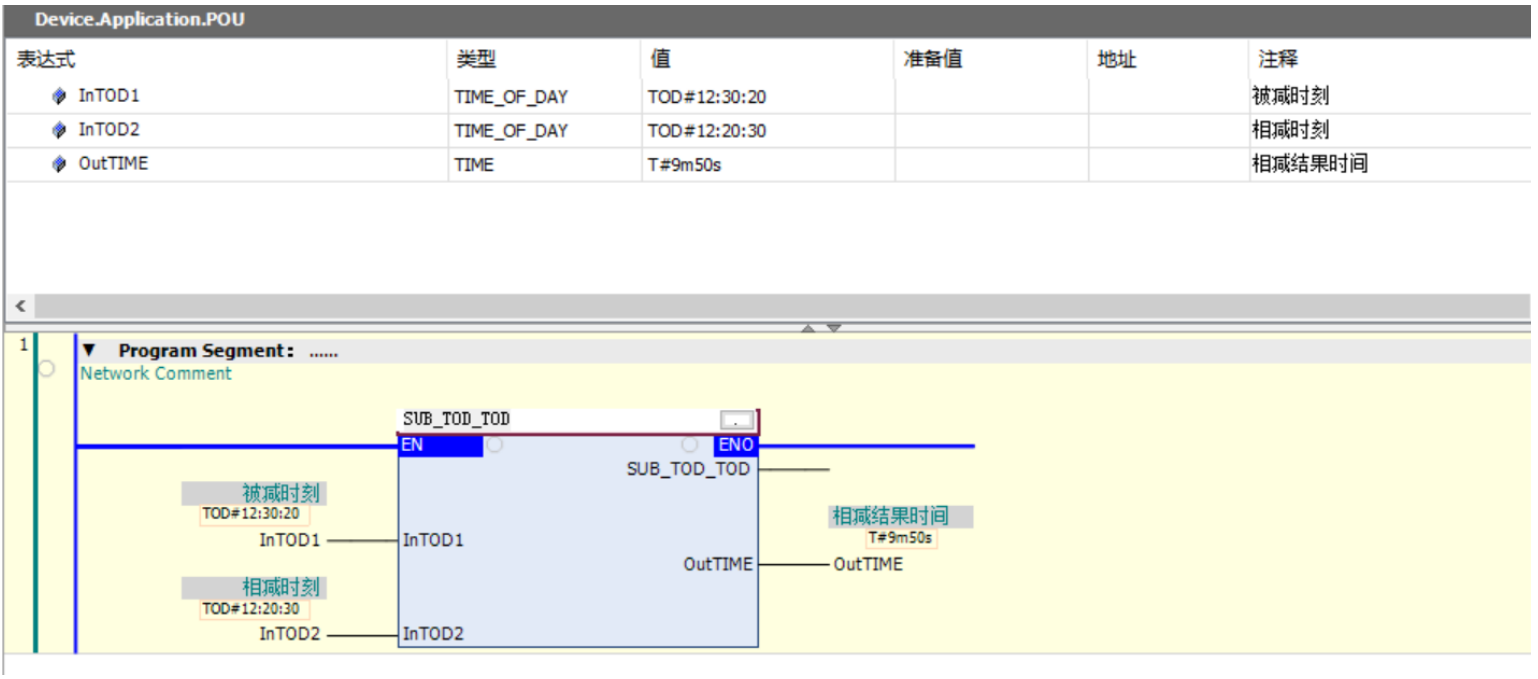
设置被减时刻和相减时刻，输出相减结果

④程序示例

ST语言

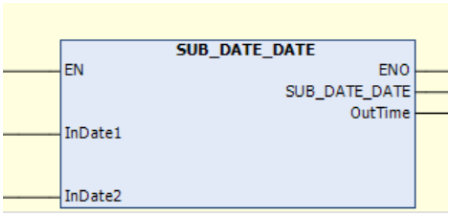


LD语言



3.1.8 SUB_DATE_DATE 日期减法

①指令格式

指令	功能	FB/FC	LD 表现	ST表现
SUB_DATE_DATE	日期减法	FC		SUB_DATE_DATE(SUB_DATE_DATE=> , InDate1:= , InDate2:= , OutTime=>);

②相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
InDate1	被减日期	DATE	遵照数据类型	-	被减日期
InDate2	相减日期	DATE	遵照数据类型	-	相减日期

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
OutTime	相减结果时间	TIME	遵照数据类型	–	相减结果时间

数据类型

	布尔	位串	整数	实数	时刻、持续时间 日期、字符串

	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
InDate1																	√			
InDate2																	√			
OutTIME																√				

③功能说明

设置被减日期和相减日期，输出相减结果

④程序示例

ST语言

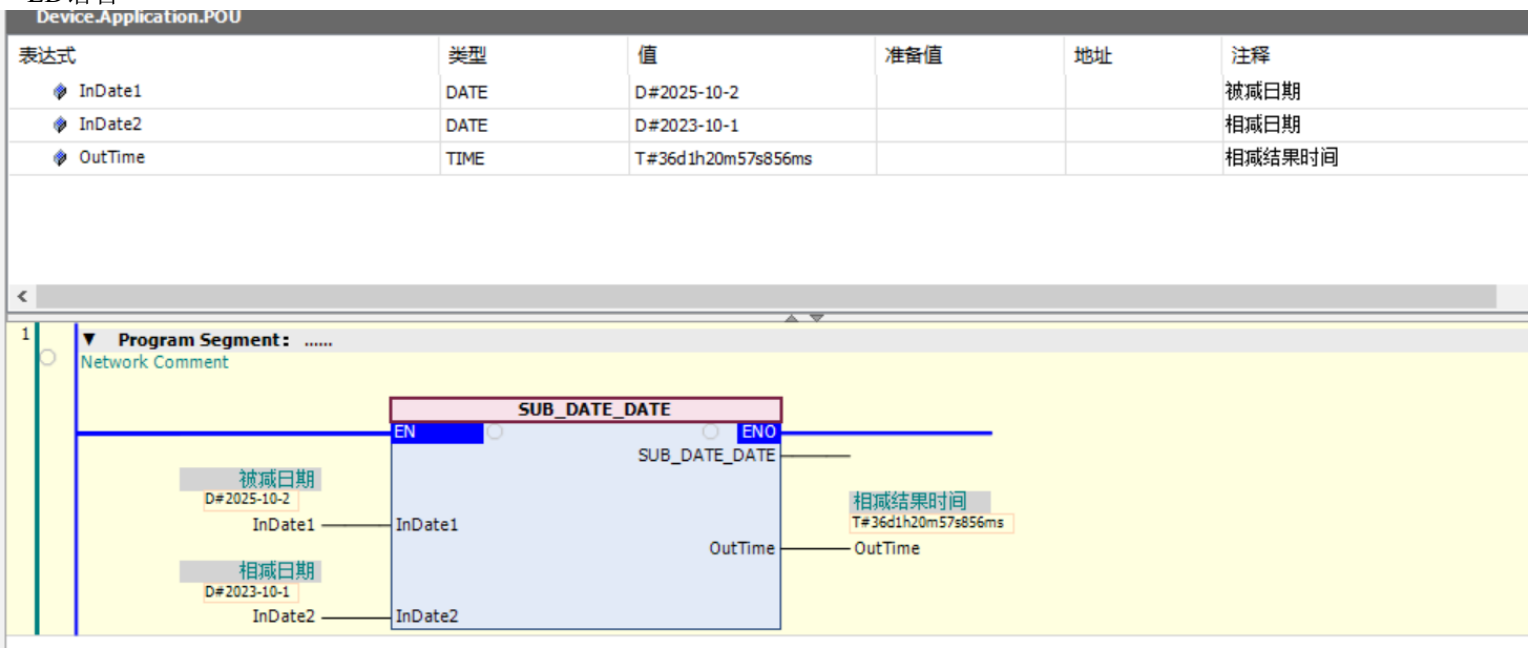
表达式	类型	值	准备值	地址	注释
InDate1	DATE	D#2025-10-2			被减日期
InDate2	DATE	D#2023-10-1			相减日期
OutTime	TIME	T#36d1h20m57s856ms			相减结果时间

```
1  SUB_DATE_DATE(SUB_DATE_DATE=> , InDate1:= InDate1 D#2025-10-2 , InDate2:=InDate2 D#2023-10-1 , OutTime=>OutTime T#36d1h20m57s856ms )
```

LD语言

Device.Application.POU

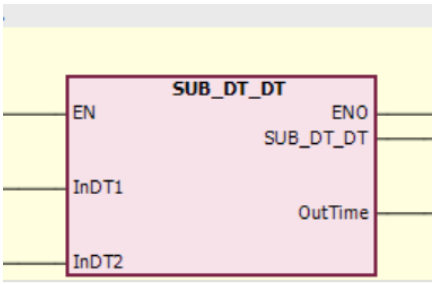
表达式	类型	值	准备值	地址	注释
InDate1	DATE	D#2025-10-2			被减日期
InDate2	DATE	D#2023-10-1			相减日期
OutTime	TIME	T#36d1h20m57s856ms			相减结果时间



3.1.9 SUB_DT_DT 日期时刻减法

①指令格式

指令	功能	FB/FC	LD 表现	ST表现

SUB_DT_DT	日期时刻减法	FC		SUB_DT_DT(SUB_DT_DT=> , InDT1:=, InDT2:=, OutTime=>);

②相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
InDT1	被减日期时刻	DT	遵照数据类型	-	被减日期时刻
InDT2	相减日期时刻	DT	遵照数据类型	-	相减日期时刻
OutTime	相减结果时间	TIME	遵照数据类型	-	相减结果时间

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
OutTime	相减结果时间	TIME	遵照数据类型	–	相减结果时间

数据类型

	布尔	位串				整数								实数		时刻、持续时间 日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
InDT1																			√	
InDT2																			√	
OutTIME																√				

③功能说明

设置被减日期时刻和相减日期时刻，输出相减结果

④程序示例

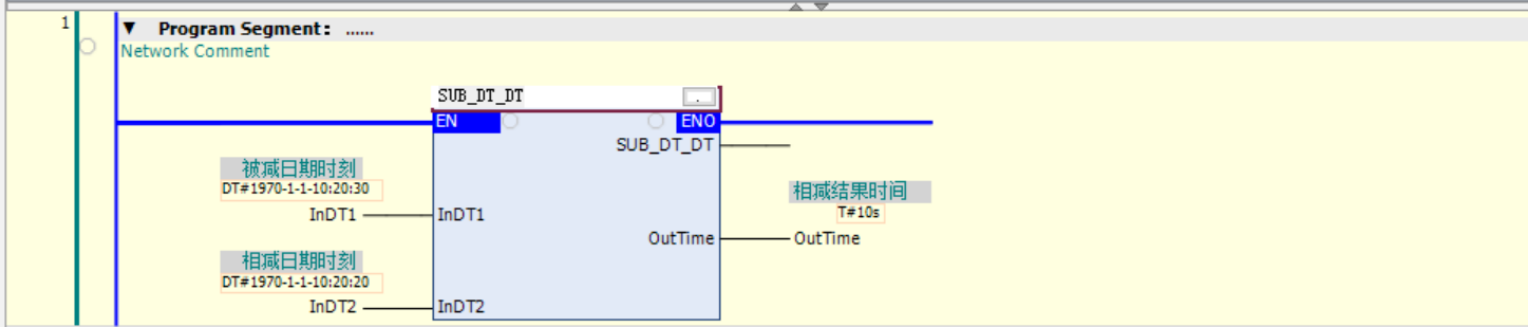
ST语言

表达式	类型	值	准备值	地址	注释
InDT1	DATE_AND_TIME	DT#1970-1-1-10:20:30			被减日期时刻
InDT2	DATE_AND_TIME	DT#1970-1-1-10:20:20			相减日期时刻
OutTime	TIME	T#10s			相减结果时间

```
1  SUB_DT_DT(SUB_DT_DT=> , InDT1:=InDT1 DT#1970-1-1-10:20:30 , InDT2:=InDT2 DT#1970-1-1-10:20:20 , OutTime=>OutTime T#10s
```

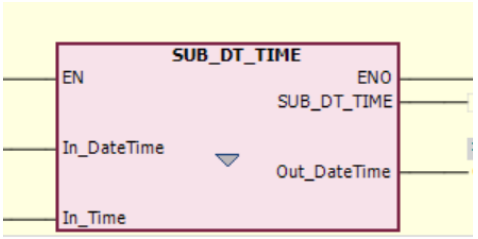
LD语言

表达式	类型	值	准备值	地址	注释
InDT1	DATE_AND_TIME	DT#1970-1-1-10:20:30			被减日期时刻
InDT2	DATE_AND_TIME	DT#1970-1-1-10:20:20			相减日期时刻
OutTime	TIME	T#10s			相减结果时间



3.1.10 SUB_DT_TIME 日期时刻减去时间

①指令格式

指令	功能	FB/FC	LD 表现	ST表现
SUB_DT_TIME	日期时刻减去时间	FC		SUB_DT_TIME(SUB_DT_TIME=> , In_DateTime:= , In_Time:= , Out_DateTime=>);

②相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
In_DateTime	被减日期时刻	DT	遵照数据类型	-	被减日期时刻
In_Time	相减时间	TIME	遵照数据类型	-	相减时间

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Out_DateTime	相减结果日期时刻	DT	遵照数据类型	–	相减结果日期时刻

数据类型

	布尔	位串	整数	实数	时刻、持续时间 日期、字符串

	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In_DateTime																			√	
In_Time																√				
Out_DateTime																			√	

③功能说明

设置被减日期时刻和相减时间，输出相减结果

④程序示例

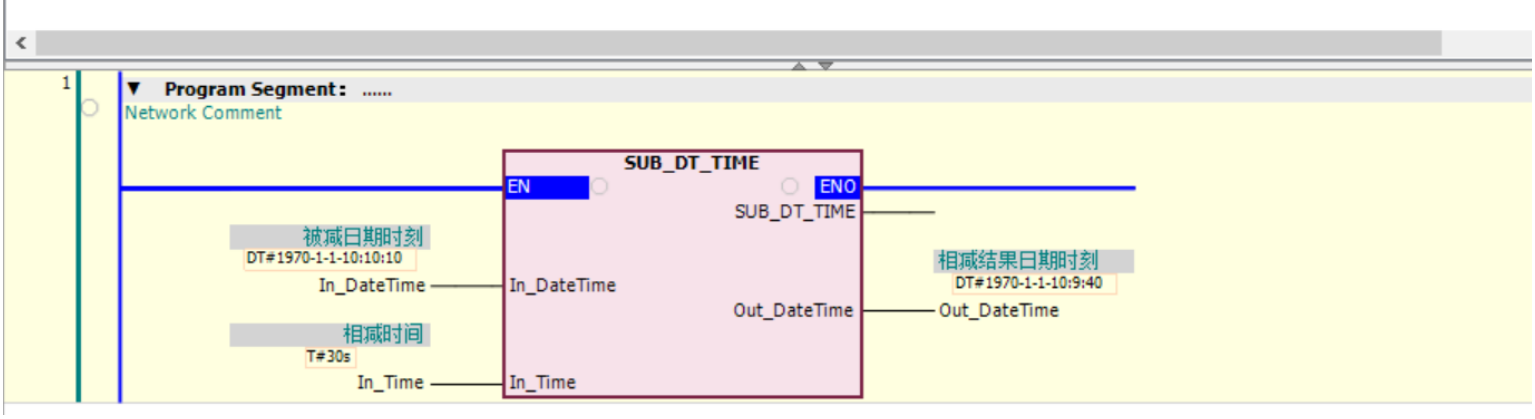
ST语言

表达式	类型	值	准备值	地址	注释
In_DateTime	DATE_AND_TIME	DT#1970-1-1-10:10:10			被减日期时刻
In_Time	TIME	T#30s			相减时间
Out_DateTime	DATE_AND_TIME	DT#1970-1-1-10:9:40			相减结果日期时刻

```
1  设备机架  TIME(SUB_DT_TIME=> , In_DateTime:=In_DateTime DT#1970-1-1-10:10:10 , In_Time:=In_Time T#30s , Out_DateTime=>Out_DateTime DT#1970-1-1-
```

LD语言

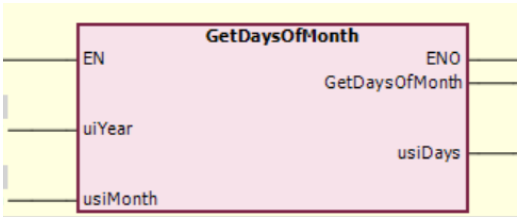
表达式	类型	值	准备值	地址	注释
In_DateTime	DATE_AND_TIME	DT#1970-1-1-10:10:10			被减日期时刻
In_Time	TIME	T#30s			相减时间
Out_DateTime	DATE_AND_TIME	DT#1970-1-1-10:9:40			相减结果日期时刻



3.1.11 GetDaysOfMonth 月的天数获取

①指令格式

指令	功能	FB/FC	LD 表现	ST表现

GetDaysOfMonth	月的 天数获取	FC		GetDaysOfMonth(GetDaysOfMonth=> , uiYear:= , usiMonth:= , usiDays=>);

②相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
uiYear	输入年份	UINT	1970 ~2106	-	输入年份
usiMonth	相减时间	USINT	1~12	-	相减时间

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
usiDays	输出范围	USINT	28~31	–	输出范围

数据类型

	布尔	位串				整数							实数		时刻、持续时间 日期、字符串					
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
uiYear							√													
usiMonth						√														
usiDays						√														

③功能说明

设置年份月份，输出对应天数

④程序示例

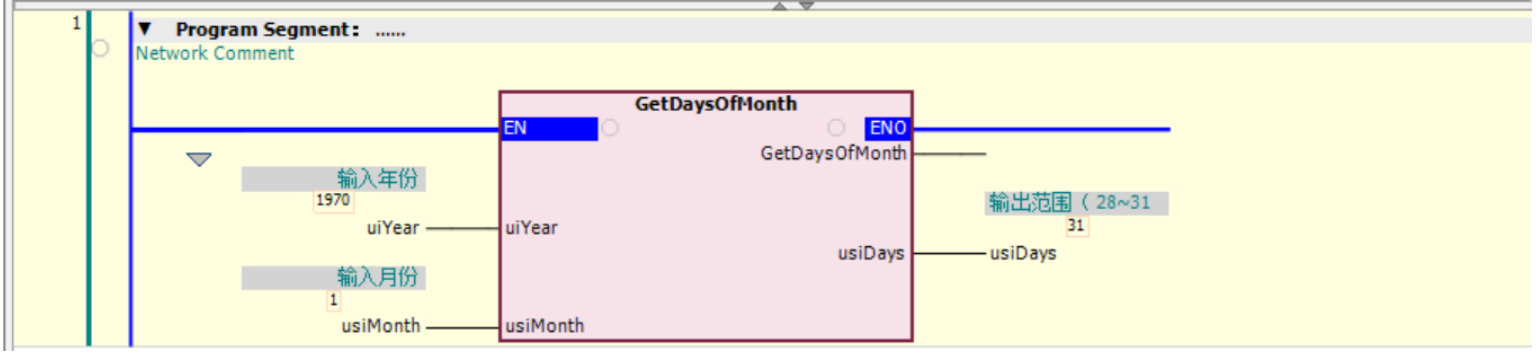
ST语言

表达式	类型	值	准备值	地址	注释
uiYear	UINT (1970..2106)	1970			输入年份
usiMonth	USINT (1..12)	1			输入月份
usiDays	USINT	31			输出范围（28~31

```
1  GetDaysOfMonth(GetDaysOfMonth=> , uiYear:=uiYear 1970 , usiMonth:=usiMonth 1 , usiDays=>usiDays 31 );
```

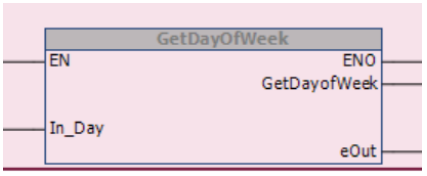
LD语言

表达式	类型	值	准备值	地址	注释
uiYear	UINT (1970..2106)	1970			输入年份
usiMonth	USINT (1..12)	1			输入月份
usiDays	USINT	31			输出范围（28~31）



3.1.12 GetDayOfWeek 星期获取

①指令格式

指令	功能	FB/FC	LD 表现	ST表现
GetDayOfWeek	星期获取	FC		GetDayOfWeek(GetDayofWeek=> , In_Day:= , eOut=>);

②相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
In_Day	年月日	ANY	DATE#1970-1-1 - DATE#2106-2-7 DT#1970-1-1-0:0:0 - DT#2106-2-7-6:28:15		年月日

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
eOut	星期结构体	eDAYOFWEEK	遵照数据类型	_MON	星期结构体

数据类型

	布尔	位串	整数	实数	时刻、持续时间 日期、字符串

	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In_Day																	√		√	
eOut	枚举类型_eDAYOFWEEK，枚举值参考功能说明																			

③功能说明

设置年月日，输出对应星期几

④程序示例

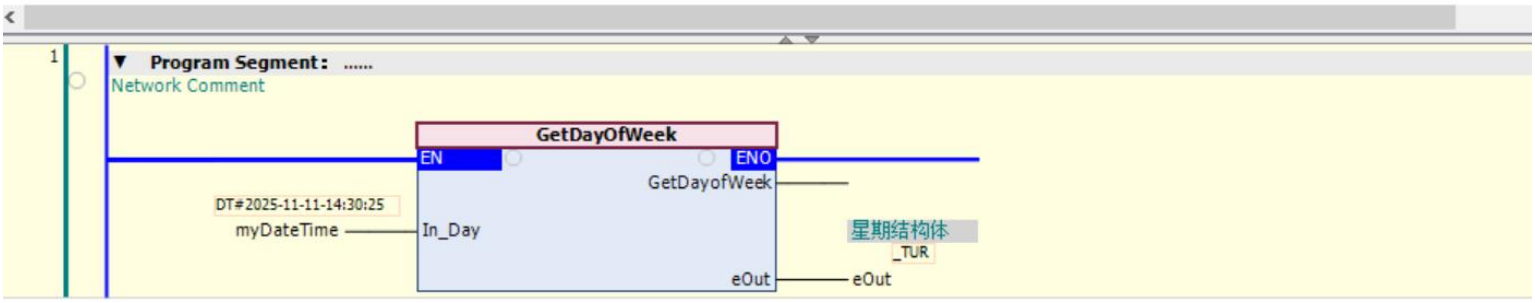
ST语言

表达式	类型	值	准备值	地址	注释
myDateTime	DATE_AND_TIME	DT#2025-11-11-14:30:25			SetSystemDateTime_0: SetSys
eOut	_EDAYOFWEEK	_TUR			星期结构体

```
1  GetDayOfWeek(GetDayofWeek=> , In_Day:= myDateTime DT#2025-11-11-14:30:25 , eOut=>eOut _TUR );
```

LD语言

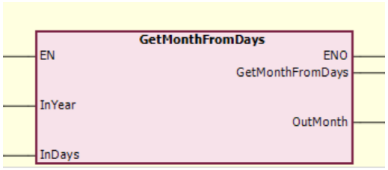
表达式	类型	值	准备值	地址	注释
myDateTime	DATE_AND_TIME	DT#2025-11-11-14:30:25			
eOut	_EDAYOFWEEK	_TUR			星期结构体



3.1.13 GetMonthFromDays 从天数获取月份

①指令格式

指令	功能	FB/FC	LD 表现	ST表现

GetMonthFromDays	从天数获取月份	FC		GetMonthFromDays(GetMonthFromDays=> , InYear:=, InDays:= , OutMonth=>);

②相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
InYear	输入年份	UINT	1970~2106	-	输入年份
InDays	自当年1月1日开始的日期	UINT	1~366	-	自当年1月1日开始的日期

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
OutMonth	输出月份	USINT	1~12	-	输出月份

数据类型

	布尔	位串				整数							实数		时刻、持续时间 日期、字符串					
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
InYear							√													
InDays							√													
OutMonth						√														

③功能说明

设置年份天数，输出对应月份

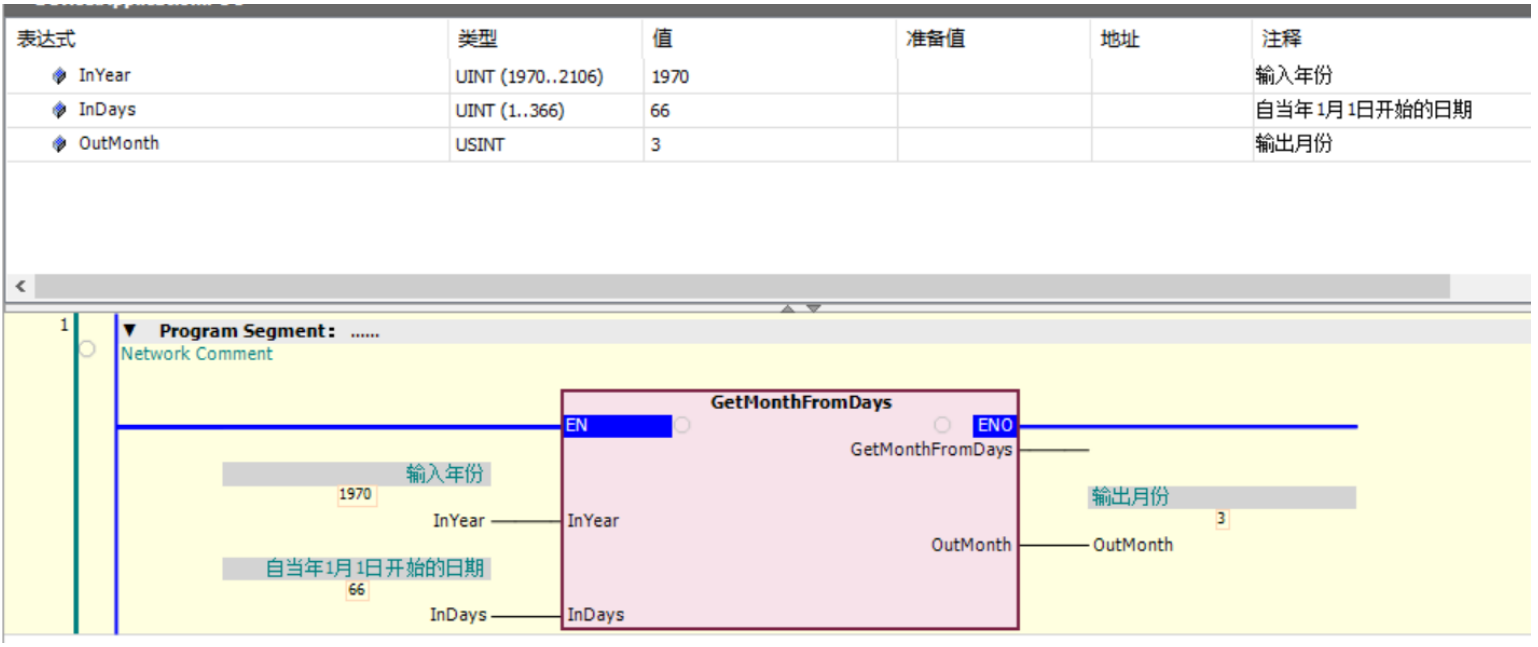
④程序示例

ST语言

表达式	类型	值	准备值	地址	注释
InYear	UINT (1970..2106)	1970			输入年份
InDays	UINT (1..366)	66			自当年 1月 1日开始的日期
OutMonth	USINT	3			输出月份

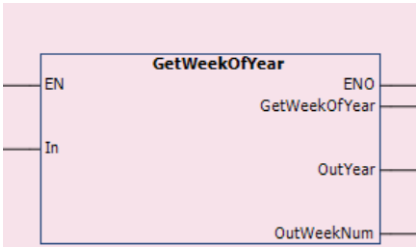
```
1  GetMonthFromDays(GetMonthFromDays=> , InYear:=InYear 1970 , InDays:= InDays 66 , OutMonth=>OutMonth 3 );
```

LD语言



3.1.14 GetWeekOfYear 周数获取

①指令格式

指令	功能	FB/FC	LD 表现	ST表现
GetWeekOfYear	周数获取	FC		GetWeekOfYear(GetWeekOfYear=>, In:= , OutYear=>, OutWeekNum=>);

②相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
IN	日期或日期时刻	DATE/DT	DATE#1970-1-1- DATE#2106-2-7 DT#1970-1-1-0:0:0 -DT#2106-2-7- 6:28:15	-	日期或日期时刻
OutYear	输出年份	UINT	1970~2106	-	输出年份

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
OutWeekNum	输出年份对应第几周	UINT	1~53	–	输出年份对应第几周

数据类型

	布尔	位串				整数							实数		时刻、持续时间 日期、字符串					
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
IN																	√		√	
OutYear							√													
OutWeekNum							√													

③功能说明

设置日期或日期时刻，输出年份对应第几周

④程序示例

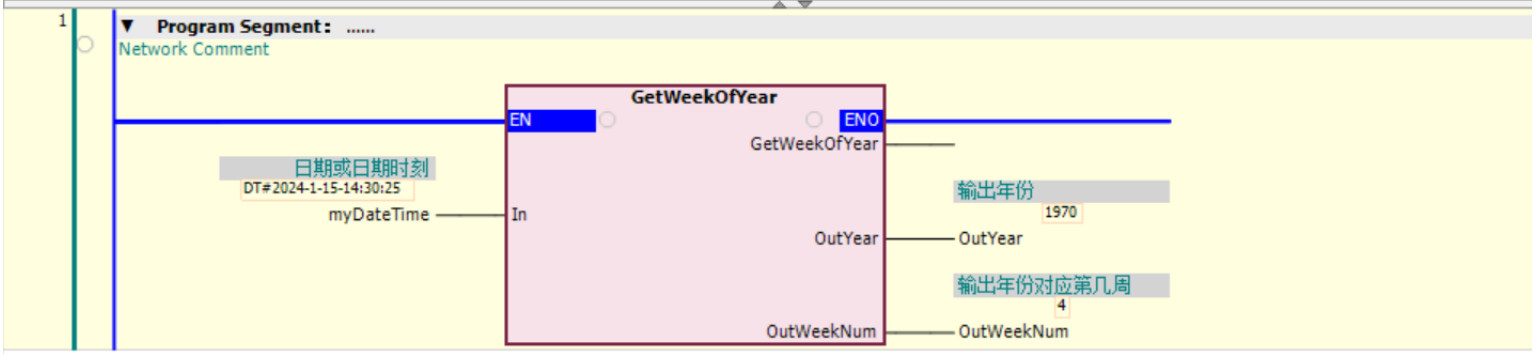
ST语言

表达式	类型	值	准备值	地址	注释
myDateTime	DATE_AND_TIME	DT#2024-1-15-14:30:25			日期或日期时刻
OutYear	UINT (UINT#1970.....	1970			输出年份
OutWeekNum	UINT (UINT#1..53)	4			输出年份对应第几周

```
1  GetWeekOfYear(GetWeekOfYear=> , In:= myDateTime DT#2024-1-15-14:30:25 , OutYear=>OutYear 1970 , OutWeekNum=>OutWeekNum 4 )
```

LD语言

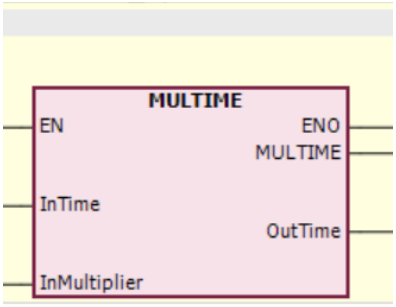
表达式	类型	值	准备值	地址	注释
myDateTime	DATE_AND_TIME	DT#2024-1-15-14:30:25			日期或日期时刻
OutYear	UINT (UINT#1970.....	1970			输出年份
OutWeekNum	UINT (UINT#1..53)	4			输出年份对应第几周



3.1.15 MULTIME 时间乘法

①指令格式

指令	功能	FB/FC	LD 表现	ST表现

MULTIME	时间乘法	FC		MULTIME(MULTIME=> , InTime:= , InMultiplier:= , OutTime=>);

②相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
InTime	被乘时间	TIME	遵照数据类型	-	被乘时间
TimeL	乘数，必须为非负数	-	遵照数据类型	-	乘数，必须为非负数

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
OutTime	相乘结果时间	TIME	遵照数据类型	_	相乘结果时间

数据类型

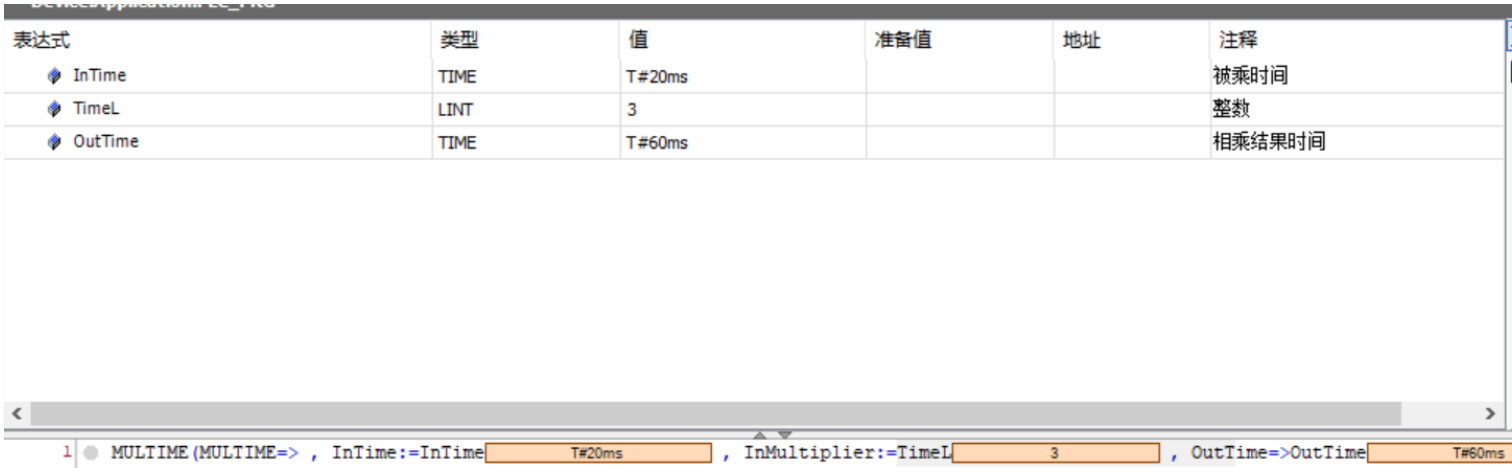
	布尔	位串				整数							实数		时刻、持续时间 日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
InTime																√				
TimeL						√	√	√	√	√	√	√	√	√	√					
OutTime																√				

③功能说明

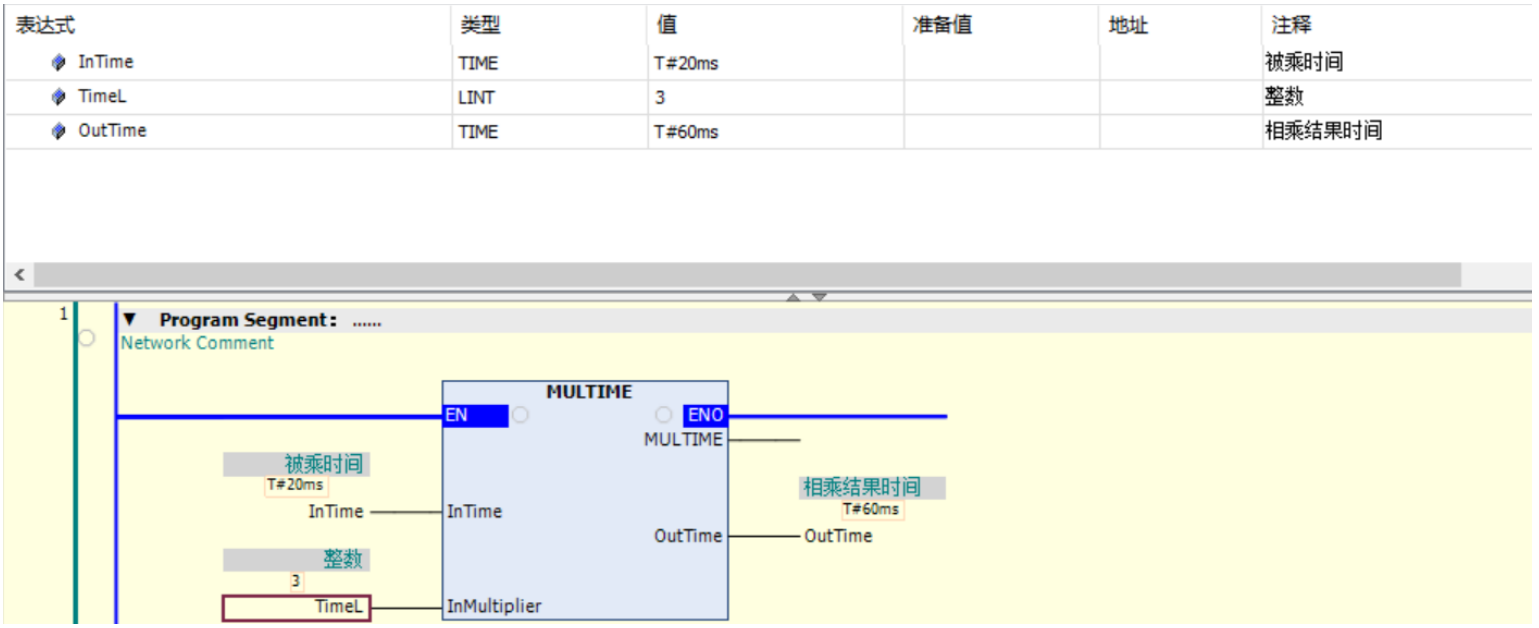
设置被乘时间，乘数，必须为非负数,输出相乘结果

④程序示例

ST语言

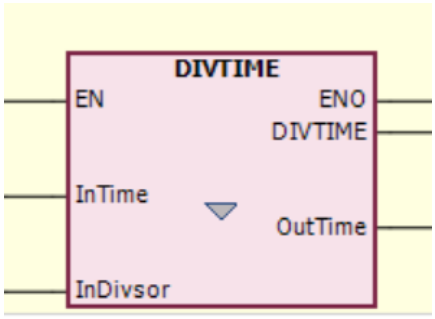


LD语言



3.1.16 DIVTIME 时间除法

①指令格式

指令	功能	FB/FC	LD 表现	ST表现
DIVTIME	时间除法	FC		DIVTIME(DIVTIME=> , InTime:=InTime , InDivsor:=InDivsor , OutTime=> OutTime);

②相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
InTime	被除时间	TIME	遵照数据类型	-	被除时间
InDivsor	除数，必须为正数	-	遵照数据类型	-	除数，必须为正数

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
OutTime	相除结果时间	TIME	遵照数据类型	_	相除结果时间

数据类型

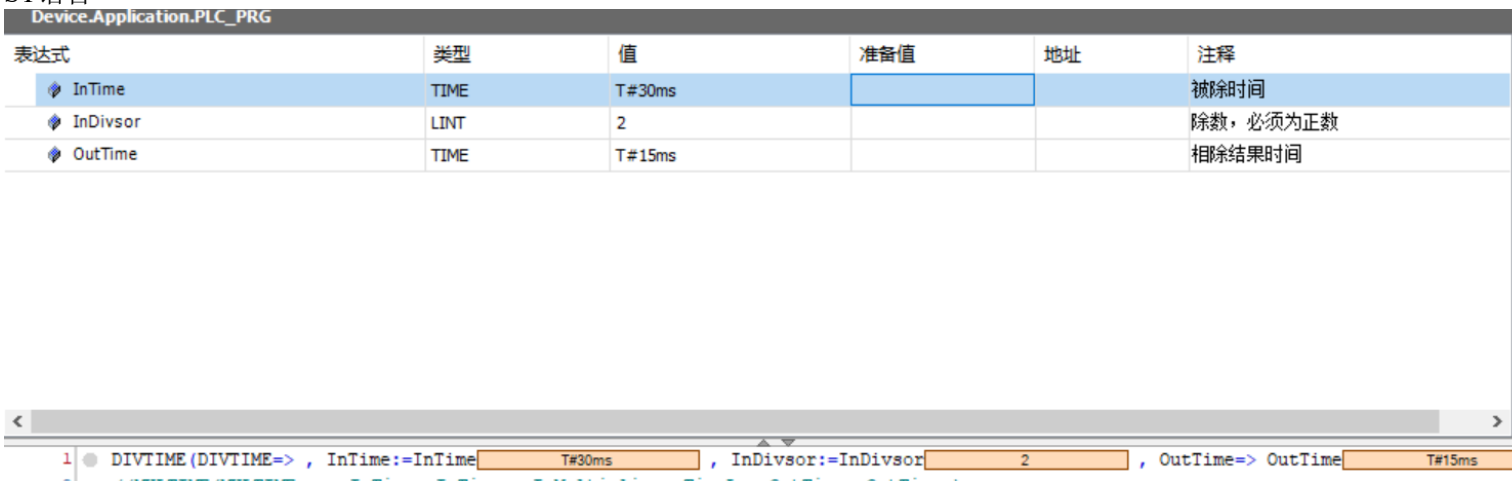
	布尔	位串				整数							实数		时刻、持续时间 日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
InTime																√				
InDivsor						√	√	√	√	√	√	√	√	√	√					
OutTime																√				

③功能说明

设置被除时间，除数，必须为正数,输出相除结果

④程序示例

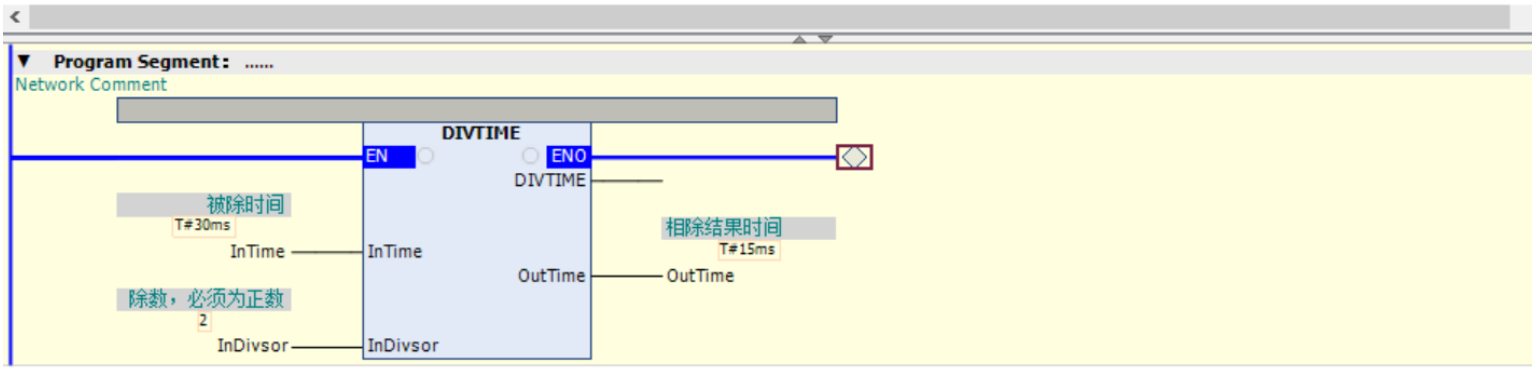
ST语言



LD语言

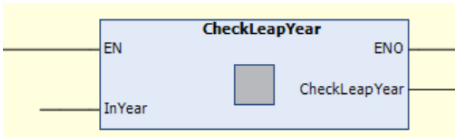
Device.Application.POU

表达式	类型	值	准备值	地址	注释
InTime	TIME	T#30ms			被除时间
InDivsor	LINT	2			除数，必须为正数
OutTime	TIME	T#15ms			相除结果时间
mc_power_1	MC_Power				



3.1.17 CheckLeapYear 闰年判断

①指令格式

指令	功能	FB/FC	LD 表现	ST表现
CheckLeapYear	闰年判断	FC		CheckLeapYear(CheckLeapYear=>, InYear:=);

②相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
InYear	输入年份	UINT	1970~2106	-	输入年份

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
CheckLeapYear_1	判断闰年	BOOL	遵照数据类型	–	判断闰年

数据类型

	布尔	位串				整数							实数		时刻、持续时间 日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING

InYear							√													
CheckLeapYear_1	√																			

③功能说明

输入年份，判断是否为闰年

④程序示例

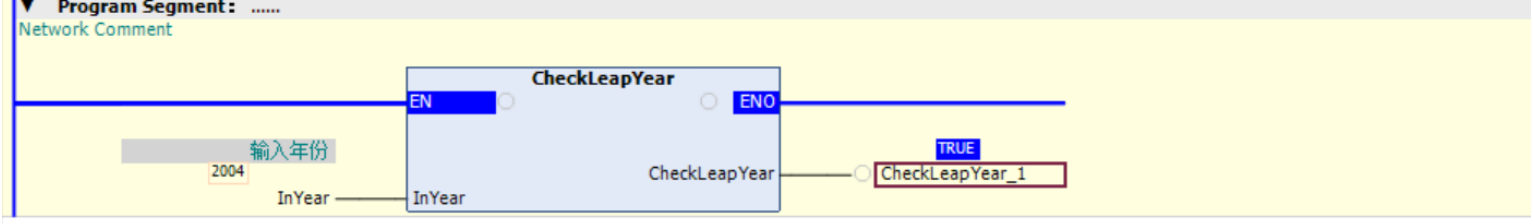
ST语言

表达式	类型	值	准备值	地址	注释
InYear	UINT (1970..2106)	2004			输入年份
CheckLeapYear_1	BOOL	TRUE			

```
1  CheckLeapYear(CheckLeapYear=>CheckLeapYear_1 , InYear:= InYear );
```

LD语言

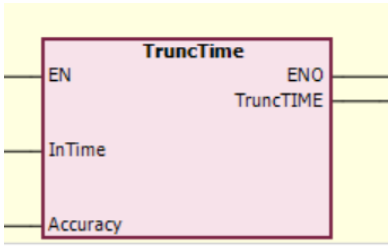
表达式	类型	值	准备值	地址	注释
InYear	UINT (1970..2106)	2004			输入年份
CheckLeapYear_1	BOOL	TRUE			



3.1.18 TruncTime 时间舍去

①指令格式

指令	功能	FB/FC	LD 表现	ST表现

TruncTime	时间除法	FC		DIVTIME(DIVTIME=> , InTime:= , InDivsor:= , OutTime=>);

②相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
InTime	输入时间	TIME	遵照数据类型	-	输入时间
Accuracy:	舍去单位	_eSUBSECOND	-	_SEC	舍去单位

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
TruncTIME_1	时间舍去结果	TIME	遵照数据类型	–	时间舍去结果

数据类型

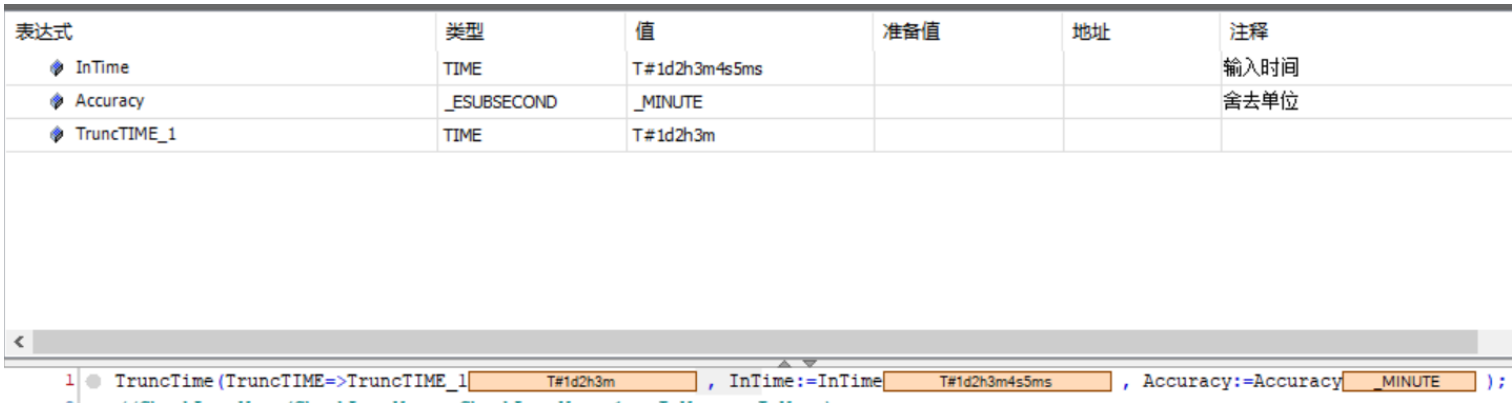
	布尔	位串				整数							实数		时刻、持续时间 日期、字符串					
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
InTime																√				
Accuracy:	枚举类型_eSUBSEC，枚举值参考功能说明																			
TruncTIME_1																√				

③功能说明

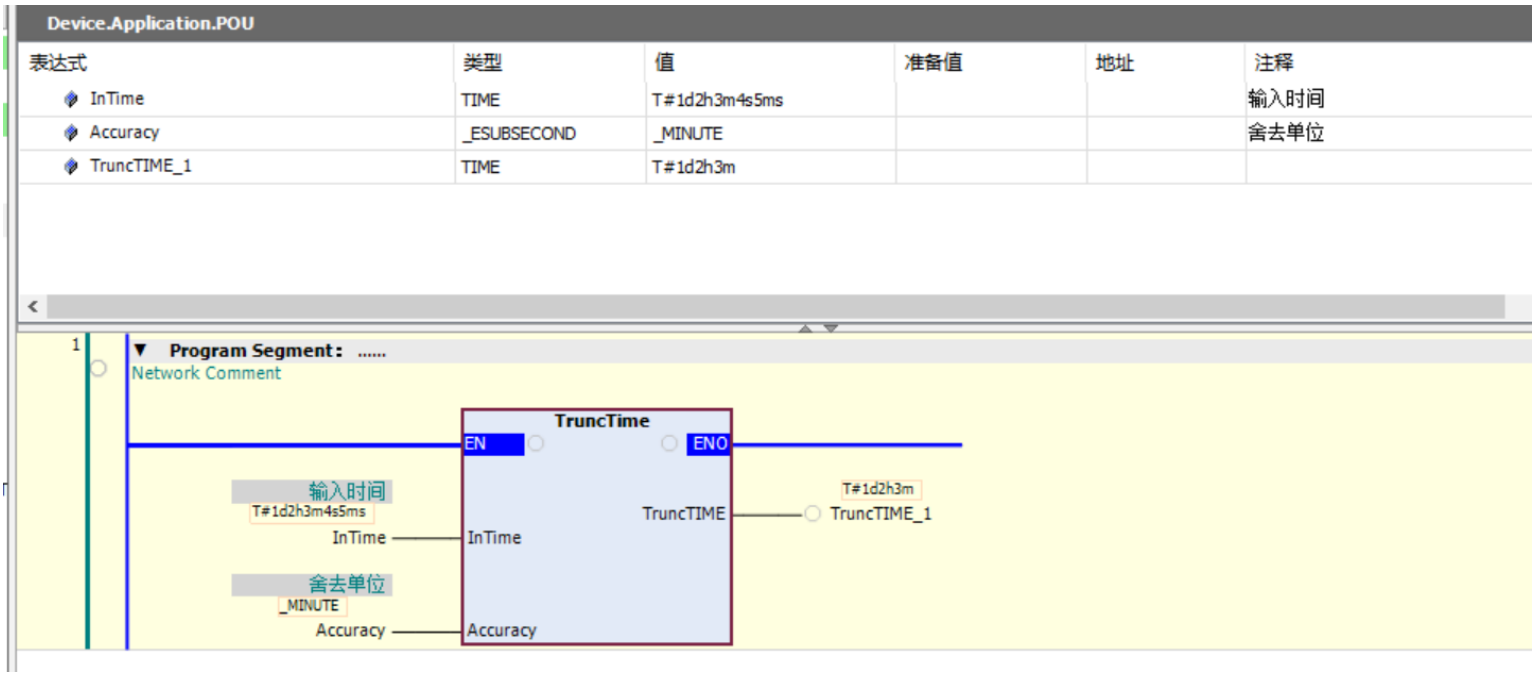
设置输入时间,舍去单位,输出舍去后的时间

④程序示例

ST语言

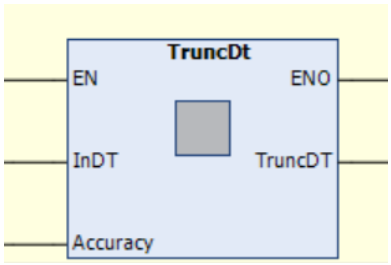


LD语言



3.1.19 TruncDt 日期时刻舍去

①指令格式

指令	功能	FB/FC	LD 表现	ST表现
TruncDt	时间除法	FC		TruncDt(TruncDT=> T, InDT:= , Accuracy:=);

②相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
InDT	输入日期时刻	DT	遵照数据类型	-	输入日期时刻
Accuracy	舍去单位	_eSUBSECOND	-	-	舍去单位

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
TruncDT_1	时间舍去结果	DT	遵照数据类型	–	TruncDT_1

数据类型

	布尔	位串	整数	实数	时刻、持续时间 日期、字符串

	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
InDT																			√	
Accuracy	枚举类型_eSUBSECOND																			
TruncDT_1																			√	

③功能说明

设置输入日期时刻,舍去单位,输出舍去后的时间

④程序示例

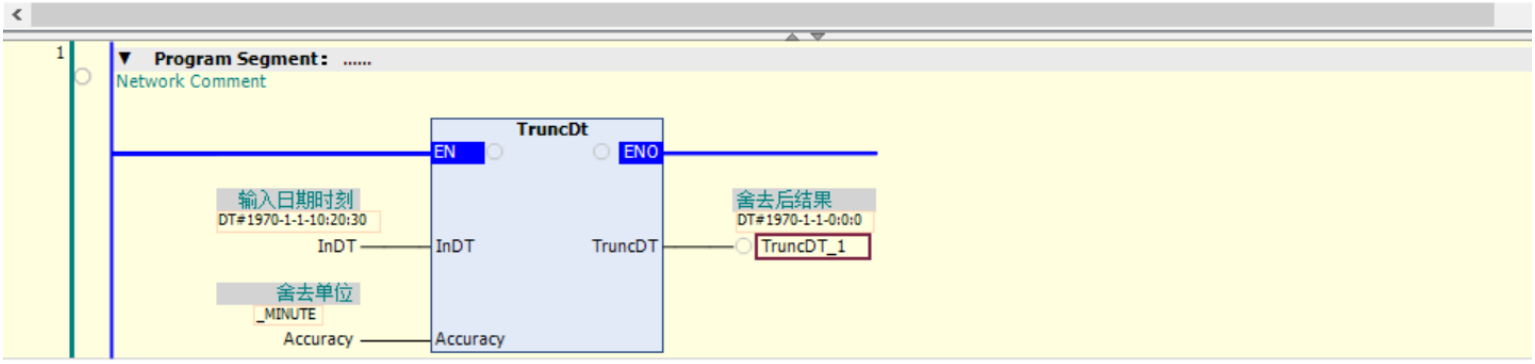
ST语言

表达式	类型	值	准备值	地址	注释
InDT	DATE_AND_TIME	DT#1970-1-1-10:20:30			输入日期时刻
Accuracy	_ESUBSECOND	_MINUTE			舍去单位
TruncDT_1	DATE_AND_TIME	DT#1970-1-1-0:0:0			舍去后结果

```
1  TruncDt(TruncDT=> TruncDT_1 DT#1970-1-1-0:0:0 , InDT:=InDT DT#1970-1-1-10:20:30 , Accuracy:=Accuracy _MINUTE );
```

LD语言

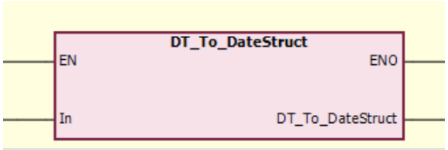
表达式	类型	值	准备值	地址	注释
InDT	DATE_AND_TIME	DT#1970-1-1-10:20:30			输入日期时刻
Accuracy	_ESUBSECOND	_MINUTE			舍去单位
TruncDT_1	DATE_AND_TIME	DT#1970-1-1-0:0:0			舍去后结果



3.1.20 DT_To_DateStruct 时刻分解

①指令格式

指令	功能	FB/FC	LD 表现	ST表现

DT_To_DateStruct	时刻分解	FC		DT_To_DateStruct(DT_To_DateStruct=>, In:=);

②相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
In	输入日期时刻	DT	遵照数据类型	-	输入日期时刻

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
DT_To_DateStruct_1	时刻分解	sDT	遵照数据类型	-	时刻分解

数据类型

	布尔	位串				整数							实数		时刻、持续时间 日期、字符串					
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In																			√	
DT_To_DateStruct_1																			√	

③功能说明

设置输入日期时刻,输出时刻分解后的结果

④程序示例

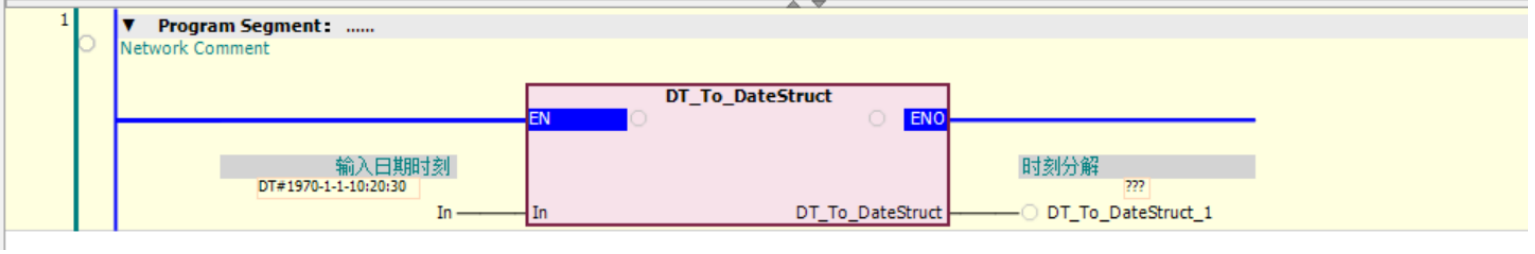
ST语言

In	DATE_AND_TIME	DT#1970-1-1-10:20:30			输入日期时刻
DT_To_DateStruct_1	_sDT				时刻分解
uiYear	UINT (1970..2106)	1970			年
uiMonth	UINT (1..12)	1			月
uiDay	UINT (1..31)	1			日
uiHour	UINT (0..23)	10			时
uiMinute	UINT (0..59)	20			分
uiSecond	UINT (0..59)	30			秒
uiMillisecond	UINT (0..999)	0			毫秒
uiDayofWeek	UINT (1..7)	1			星期几

```
1  DT_To_DateStruct(DT_To_DateStruct=>DT_To_DateStruct_1 , In:= In DT#1970-1-1-10:20:30 );
```

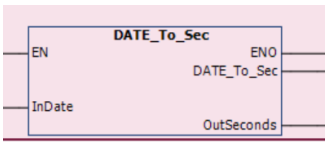
LD语言

表达式	类型	值	准备值	地址	注释
In	DATE_AND_TIME	DT#1970-1-1-10:20:30			输入日期时刻
DT_To_DateStruct_1	_sDT				时刻分解
uiYear	UINT (1970..2106)	1970			年
uiMonth	UINT (1..12)	1			月
uiDay	UINT (1..31)	1			日
uiHour	UINT (0..23)	10			时
uiMinute	UINT (0..59)	20			分
uiSecond	UINT (0..59)	30			秒
uiMillisecond	UINT (0..999)	0			毫秒
uiDayofWeek	UINT (1..7)	1			星期几



3.1.21 DATE_To_Sec 日期转化为秒

①指令格式

指令	功能	FB/FC	LD 表现	ST表现
DATE_To_Sec	日期转化为秒	FC		DATE_To_Sec(DATE_To_Sec=> , InDate:= , OutSeconds=>);

②相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
InDate	输入日期	DATE	遵照数据类型	DT#1970-1-1 0:0:0	输入日期

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
OutSeconds	输出秒	LINT	0 ～ 4294967295	-	输出秒

数据类型

	布尔	位串	整数	实数	时刻、持续时间 日期、字符串

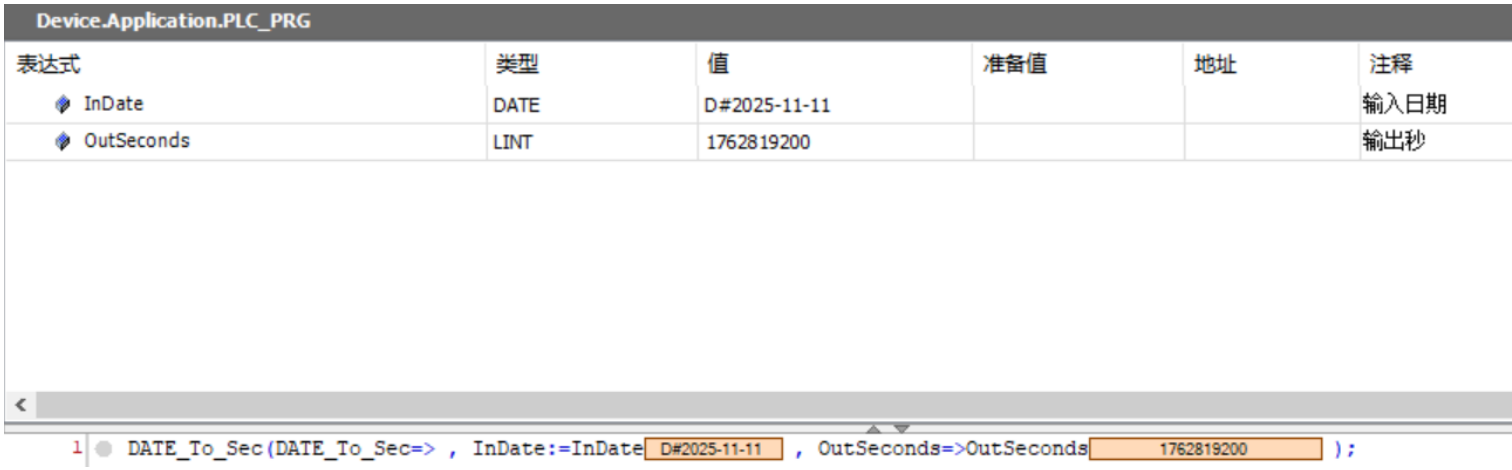
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
InDate																	√			
OutSeconds													√							

③功能说明

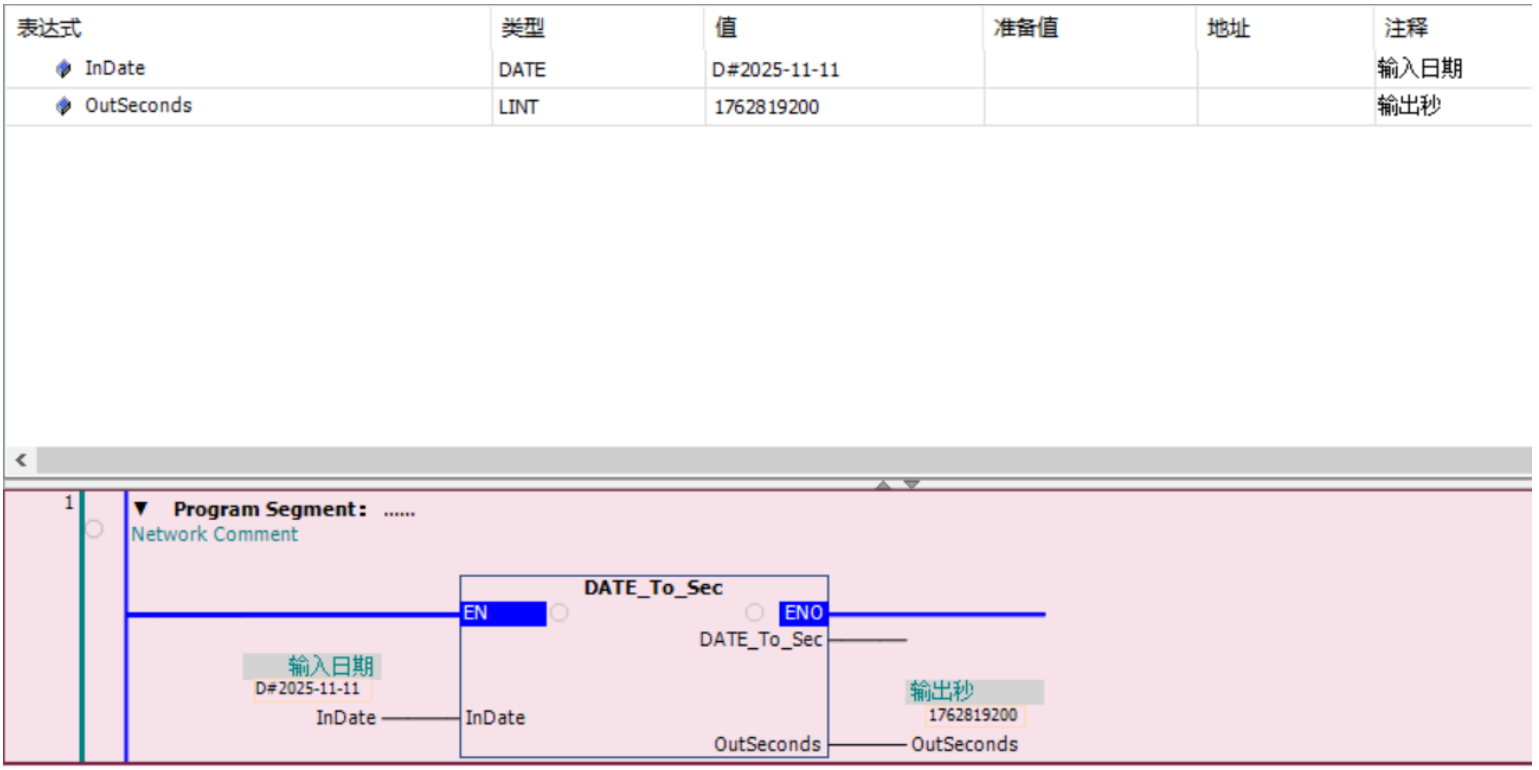
设置输入日期,将日期转换为秒

④程序示例

ST语言



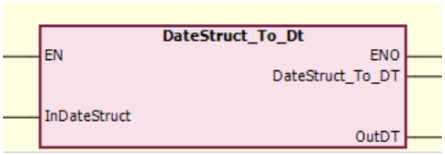
LD语言



3.1.22 DateStruct_To_Dt 时刻组合

①指令格式

指令	功能	FB/FC	LD 表现	ST表现

DateStruct_To_Dt	时刻组合	FC		DateStruct_To_Dt(DateStruct_To_DT=> , InDateStruct:= , OutDT=>);

②相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
InDateStruct	日期时刻结构体	sDT	-	-	日期时刻结构体

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
OutDT	合并后的日期时刻	DATE_AND_TIME	DT#1970-1-1-0:0:0 - DT#2106-2-7-6:28:15	DT#1970-1-1-0:0:0	合并后的日期时刻

_sDT 结构体说明

变量	名称	内容	数据类型	有效范围	初始值	单位
uiYear	年	年	UINT	1970 ~ 2106	1970	年

变量	名称	内容	数据类型	有效范围	初始值	单位
uiMonth	月	月	UINT	1 ~ 12	1	月
uiDay	日	日	UINT	1 ~ 31	1	日
uiHour	时	时	UINT	0 ~ 23	0	时
uiMinute	分	分	UINT	0 ~ 59	0	分
uiSecond	秒	秒	UINT	0 ~ 59	0	秒
uiMillisecond	毫秒	毫秒	UINT	0 ~ 999	0	毫秒
uiDayOfWeek	星期几	星期几	UINT	1 ~ 7	1	-

数据类型

	布尔	位串				整数							实数		时刻、持续时间 日期、字符串					
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
InDateStruct	_sDT结构体																			
OutDT																√	√			

③功能说明

设置输入日期时刻结构体,输出合并后的日期时刻

④程序示例

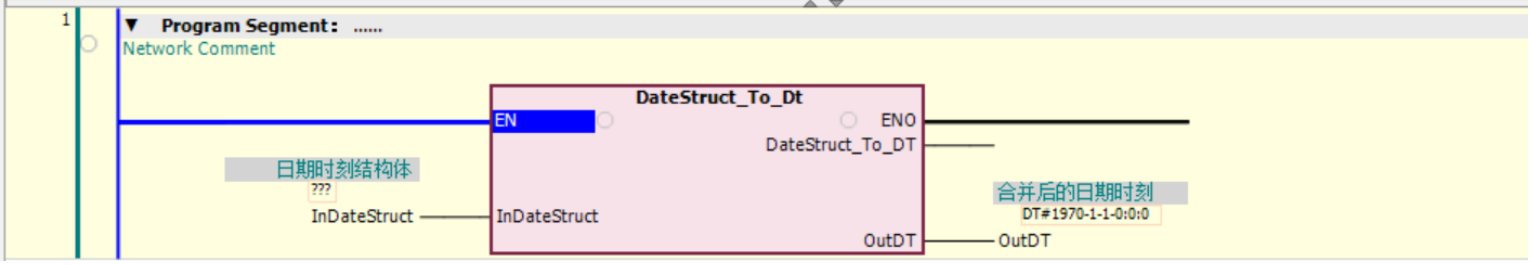
ST语言

InDateStruct	_sDT				日期时刻结构体
uiYear	UINT (1970..2106)	1970			年
uiMonth	UINT (1..12)	1			月
uiDay	UINT (1..31)	1			日
uiHour	UINT (0..23)	0			时
uiMinute	UINT (0..59)	0			分
uiSecond	UINT (0..59)	0			秒
uiMillisecond	UINT (0..999)	0			毫秒
uiDayofWeek	UINT (1..7)	1			星期几
OutDT	DATE_AND_TIME	DT#1970-1-1-0:0:0			合并后的日期时刻

```
1  DateStruct_To_Dt(DateStruct_To_DT=> , InDateStruct:=InDateStruct , OutDT=>OutDT DT#1970-1-1-0:0:0 );
```

LD语言

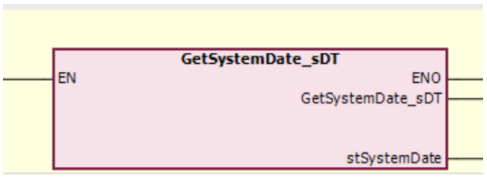
表达式	类型	值	准备值	地址	注释
InDateStruct	_sDT				日期时刻结构体
uiYear	UINT (1970..2106)	1970			年
uiMonth	UINT (1..12)	1			月
uiDay	UINT (1..31)	1			日
uiHour	UINT (0..23)	0			时
uiMinute	UINT (0..59)	0			分
uiSecond	UINT (0..59)	0			秒
uiMillisecond	UINT (0..999)	0			毫秒
uiDayofWeek	UINT (1..7)	1			星期几
OutDT	DATE_AND_TIME	DT#1970-1-1-0:0:0			合并后的日期时刻



3.1.23 GetSystemDate_sDT 获取当前系统时间结构体

①指令格式

指令	功能	FB/FC	LD 表现	ST表现

GetSystemDate_sDT	获取当前系统时间结构体	FC		GetSystemDate_sDT(GetSystemDate_sDT=> , stSystemDate=>);

②相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
-	-	-	-	-	-

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
stSystemDate	获取到的日期时刻结构体	sDT	–	–	获取到的日期时刻结构体

_sDT 结构体说明

变量	名称	内容	数据类型	有效范围	初始值	单位
uiYear	年	年	UINT	1970 ~ 2106	0	年
uiMonth	月	月	UINT	1 ~ 12	0	月
uiDay	日	日	UINT	1 ~ 31	0	日
uiHour	时	时	UINT	0 ~ 23	0	时
uiMinute	分	分	UINT	0 ~ 59	0	分
uiSecond	秒	秒	UINT	0 ~ 59	0	秒
uiMillisecond	毫秒	毫秒	UINT	0 ~ 999	0	毫秒
uiDayOfWeek	星期几	星期几	UINT	1 ~ 7	0	-

数据类型

	布尔	位串				整数							实数		时刻、持续时间 日期、字符串					
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
stSystemDate	_sDT结构体																			

③功能说明

获取当前系统时间结构体

④程序示例

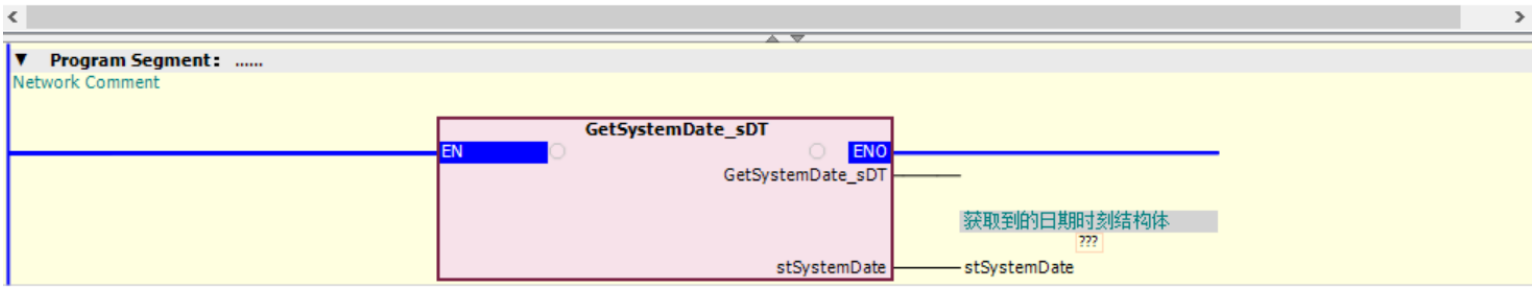
ST语言

表达式	类型	值	准备值	地址	注释
stSystemDate	_sDT				获取到的日期时刻结构体
uiYear	UINT (1970..2106)	2025			年
uiMonth	UINT (1..12)	11			月
uiDay	UINT (1..31)	12			日
uiHour	UINT (0..23)	9			时
uiMinute	UINT (0..59)	22			分
uiSecond	UINT (0..59)	29			秒
uiMillisecond	UINT (0..999)	525			毫秒
uiDayofWeek	UINT (1..7)	3			星期几

```
1  GetSystemDate_sDT(GetSystemDate_sDT=> , stSystemDate=>stSystemDate );
2
```

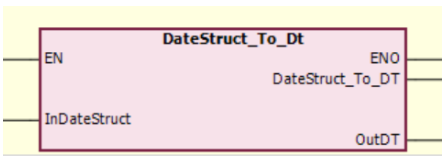
LD语言

表达式	类型	值	准备值	地址	注释
stSystemDate	_sDT				获取到的日期时刻结构体
uiYear	UINT (1970..2106)	2025			年
uiMonth	UINT (1..12)	11			月
uiDay	UINT (1..31)	12			日
uiHour	UINT (0..23)	9			时
uiMinute	UINT (0..59)	22			分
uiSecond	UINT (0..59)	42			秒
uiMillisecond	UINT (0..999)	497			毫秒
uiDayofWeek	UINT (1..7)	3			星期几



3.1.24 DT_To_Sec 日期转换为秒

①指令格式

指令	功能	FB/FC	LD 表现	ST表现
DT_To_Sect	日期转换为秒	FC		DateStruct_To_Dt(DateStruct_To_DT=> , InDateStruct:= , OutDT=>);

②相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
InDT	输入日期时刻	DT	遵照数据类型	DT#1970-1-1 0:0:0	输入日期时刻

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
OutSeconds	输出秒	LINT	0 ～ 4294967295	-	输出秒

数据类型

	布尔	位串				整数							实数		时刻、持续时间 日期、字符串					
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
InDT																			√	
OutSeconds													√							

③功能说明

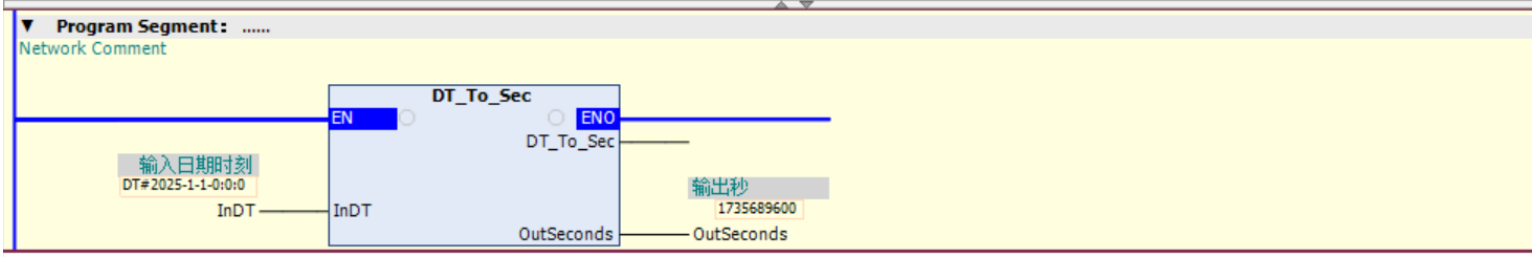
设置输入日期时刻结构体,输出合并后的日期时刻

④程序示例

ST语言

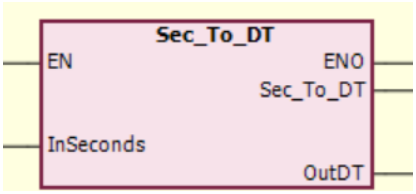
LD语言

表达式	类型	值	准备值	地址	注释
InDT	DATE_AND_TIME	DT#2025-1-1-0:0:0			输入日期时刻
OutSeconds	LINT	1735689600			输出秒



3.1.25 Sec_To_DT 秒数转换为日期时刻

①指令格式

指令	功能	FB/FC	LD 表现	ST表现
Sec_To_DT	秒数转换为日期时刻	FC		Sec_To_DT(Sec_To_DT=> , InSeconds:= , OutDT=>);

②相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
InSeconds	秒数	LINT	0 ～ 4294967295	-	秒数

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
OutDT	转换日期时刻	DT	遵照数据类型	–	转换日期时刻

数据类型

	布尔	位串	整数	实数	时刻、持续时间 日期、字符串

	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
InSeconds													√							
OutDT																			√	

③功能说明

设置输入秒数,输出转换日期时刻

④程序示例

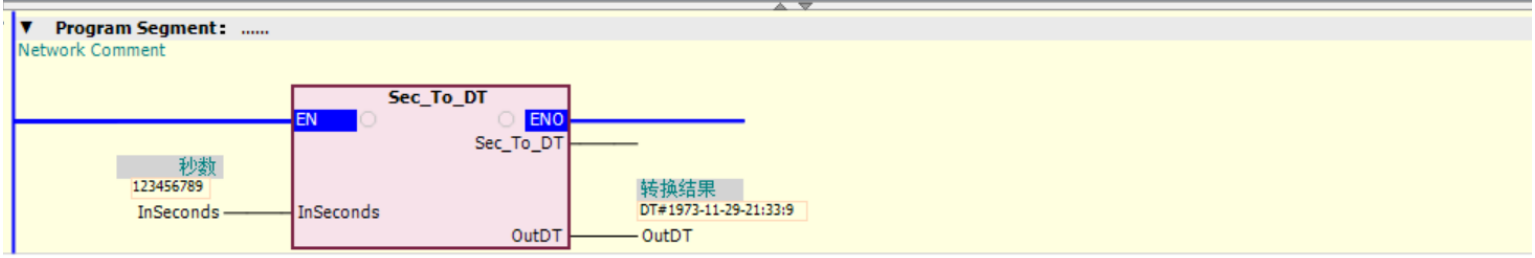
ST语言

表达式	类型	值	准备值	地址	注释
InSeconds	LINT	123456789			秒数
OutDT	DATE_AND_TIME	DT#1973-11-29-21:33:9			转换结果

```
1  Sec_To_DT(Sec_To_DT=> , InSeconds:=InSeconds 123456789 , OutDT=>OutDT DT#1973-11-29-21:33:9 );
```

LD语言

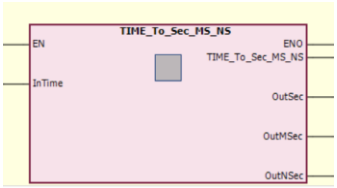
表达式	类型	值	准备值	地址	注释
InSeconds	LINT	123456789			秒数
OutDT	DATE_AND_TIME	DT#1973-11-29-21:33:9			转换结果



3.1.26 TIME_To_Sec_MS_NS 时间转换为秒数，毫秒，纳秒

①指令格式

指令	功能	FB/FC	LD 表现	ST表现

TIME_To_Sec_MS_NS	时间转换为秒数，毫秒，纳秒	FC		TIME_To_Sec_MS_NS(TIME_To_Sec_MS_NS=>, InTime:=, OutSec=> , OutMSec=> , OutNSec=>);

②相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
InTime	输入时间	TIME	遵照数据类型	T#0s	输入时间

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
OutSec	转换结果单位S	LINT	遵照数据类型	–	转换结果单位S
OutMSec	转换结果单位MS	LINT	遵照数据类型	–	转换结果单位MS
OutNSec	转换结果单位NS	LINT	遵照数据类型	–	转换结果单位NS

数据类型

	布尔	位串				整数							实数		时刻、持续时间 日期、字符串					
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
InTime																√				
OutSec													√							
OutMSec													√							
OutNSec													√							

③功能说明

设置输入时间,输出时间转换为秒数，毫秒，纳秒

④程序示例

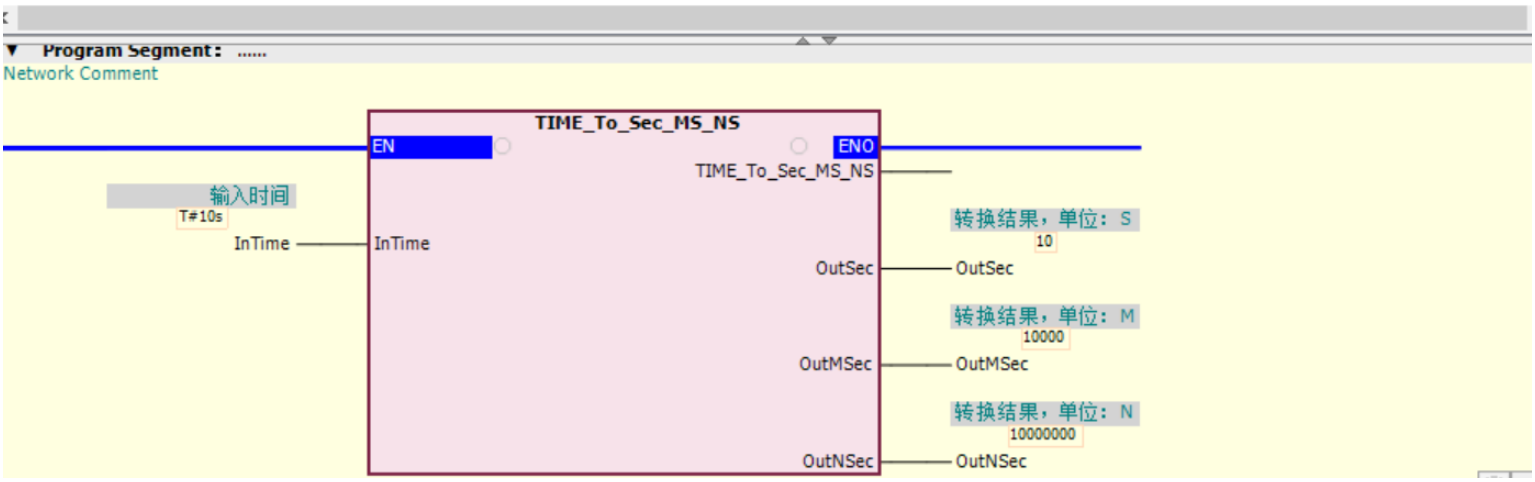
ST语言

表达式	类型	值	准备值	地址	注释
InTime	TIME	T#10s			输入时间
OutSec	LINT	10			转换结果，单位：S
OutMSec	LINT	10000			转换结果，单位：MS
OutNSec	LINT	10000000			转换结果，单位：NS

```
1  TIME_To_Sec_MS_NS(
2      TIME_To_Sec_MS_NS=> ,
3      InTime:= InTime(T#10s),
4      OutSec=>OutSec(10) ,
5      OutMSec=> OutMSec(10000),
6      OutNSec=>OutNSec(10000000) );
```

LD语言

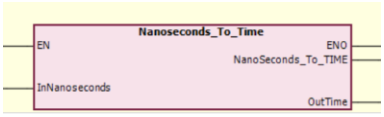
表达式	类型	值	准备值	地址	注释
InTime	TIME	T#10s			输入时间
OutSec	LINT	10			转换结果，单位：S
OutMSec	LINT	10000			转换结果，单位：MS
OutNSec	LINT	10000000			转换结果，单位：NS



3.1.27 Nanoseconds_To_Time 纳秒转换为时间

①指令格式

指令	功能	FB/FC	LD 表现	ST表现

Nanoseconds_To_Time	纳秒转换为时间	FC		Nanoseconds_To_Time(NanoSeconds_To_TIME=> , InNanoseconds:= , OutTime=>);

②相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
InNanoseconds	纳秒数	LINT	0 ～ 4294967295000000	0	纳秒数

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
OutTime	转换为时间	TIME	遵照数据类型	-	转换为时间

数据类型

	布尔	位串				整数							实数		时刻、持续时间 日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
InNanoseconds													√							
OutTime																√				

③功能说明

设置输入纳秒,输出转换后的时间

④程序示例

ST语言

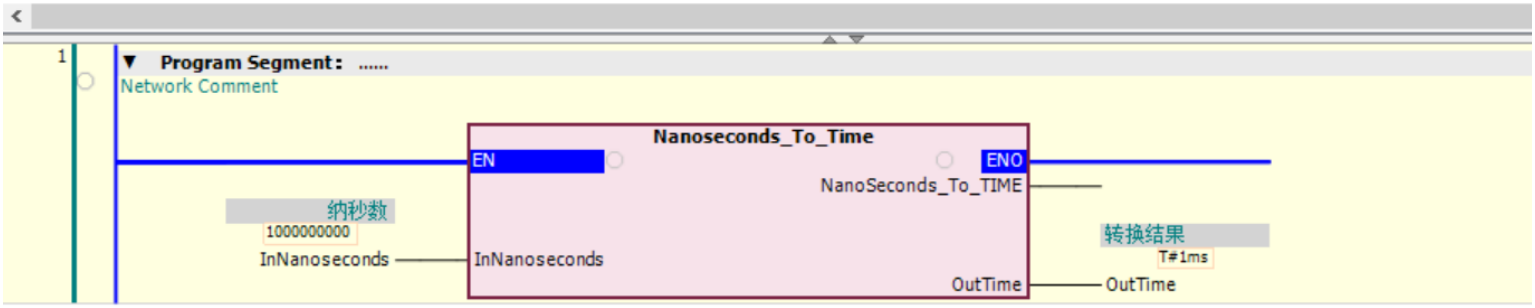
表达式	类型	值	准备值	地址	注释
InNanoseconds	LINT	10000000000			纳秒数
OutTime	TIME	T#10ms			转换结果

```
1  Nanoseconds_To_Time(NanoSeconds_To_TIME=> , InNanoseconds:=InNanoseconds 10000000000 , OutTime=>OutTime T#10ms
```

LD语言

Device.Application.POU

表达式	类型	值	准备值	地址	注释
InNanoseconds	LINT	1000000000			纳秒数
OutTime	TIME	T#1ms			转换结果



3.1.28 Sec_To_Time 秒数转换为时间

①指令格式

指令	功能	FB/FC	LD 表现	ST表现
Sec_To_Time	秒数转换为时间	FC		Sec_To_Time(Sec_To_TIME=> , InSeconds:= , OutTime=>);

②相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
InSeconds	秒数	LINT	0 ～ 4294967	0	秒数

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
OutTime	转换为时间	TIME	遵照数据类型	–	转换为时间

数据类型

	布尔	位串				整数							实数		时刻、持续时间 日期、字符串					
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
InSeconds													√							

OutTime																√				

③功能说明

设置输入秒数,输出转换后的时间

④程序示例

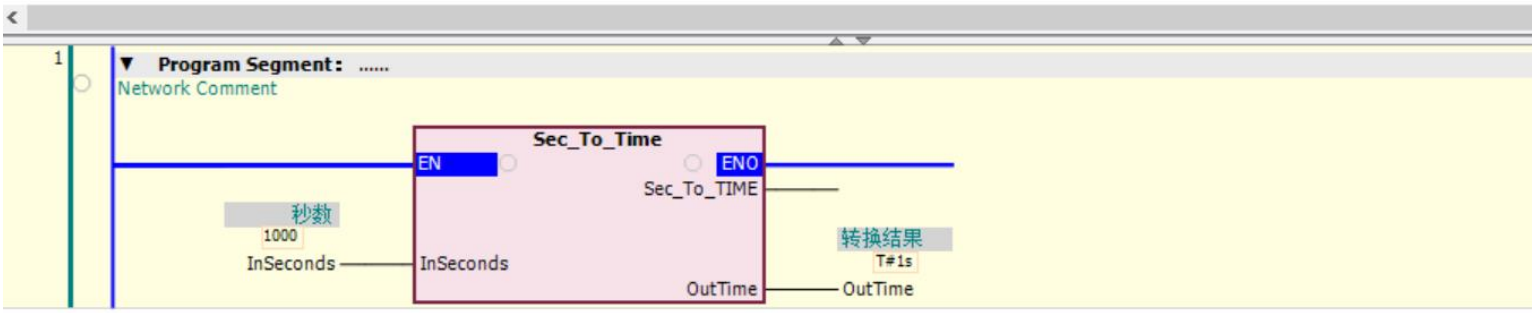
ST语言

表达式	类型	值	准备值	地址	注释
InSeconds	LINT	1000			秒数
OutTime	TIME	T#1s			转换结果

```
1  Sec_To_Time(Sec_To_TIME=> , InSeconds:=InSeconds 1000 , OutTime=>OutTime T#1s );
```

LD语言

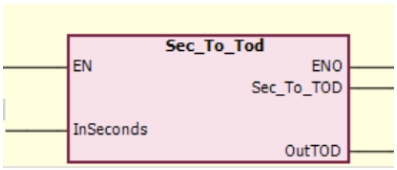
表达式	类型	值	准备值	地址	注释
InSeconds	LINT	1000			秒数
OutTime	TIME	T#1s			转换结果



3.1.29 Sec_To_Tod 秒数转换为时刻

①指令格式

指令	功能	FB/FC	LD 表现	ST表现

Sec_To_Tod	秒数转换为时刻	FC		Sec_To_Tod(Sec_To_TOD=> , InSeconds:= , OutTOD=>);

②相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
InSeconds	秒数	LINT	0 ～ 4294967295	0	秒数

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
OutTOD	转换为时刻	TIME_OF_DAY	遵照数据类型	-	时刻

数据类型

	布尔	位串				整数							实数		时刻、持续时间 日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATA	TOD	DT	STRING
InSeconds													√							
OutTOD																√				

③功能说明

设置输入秒数,输出转换后的时刻

④程序示例

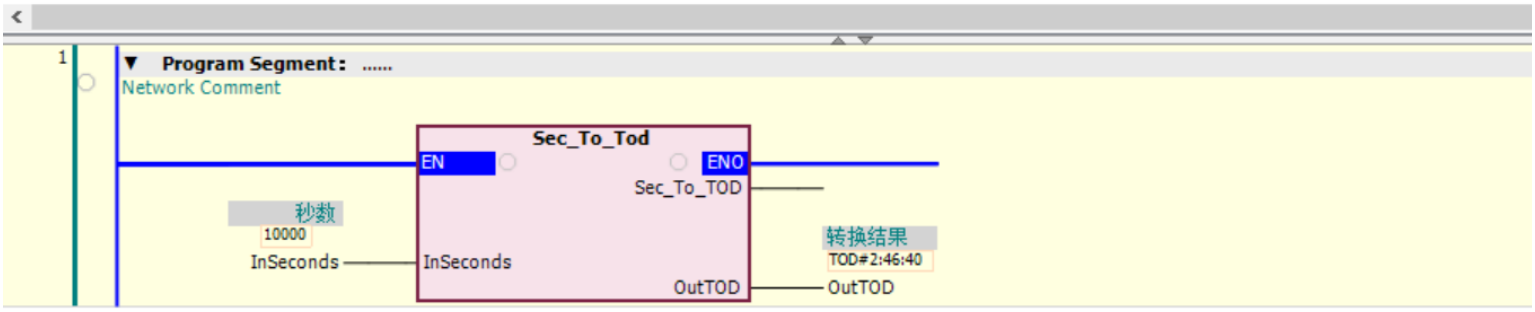
ST语言

表达式	类型	值	准备值	地址	注释
InSeconds	LINT	10000			秒数
OutTOD	TIME_OF_DAY	TOD#2:46:40			转换结果

```
1  Sec_To_Tod(Sec_To_TOD=> , InSeconds:=InSeconds 10000 , OutTOD=> OutTOD TOD#2:46:40 );
```

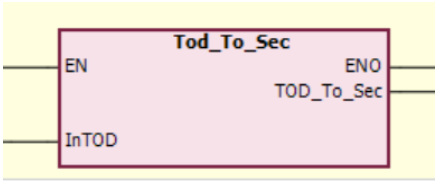
LD语言

表达式	类型	值	准备值	地址	注释
InSeconds	LINT	10000			秒数
OutTOD	TIME_OF_DAY	TOD#2:46:40			转换结果



3.1.30 Tod_To_Sec 时刻转换为秒数

①指令格式

指令	功能	FB/FC	LD 表现	ST表现
Tod_To_Sec	时刻转换为秒数	FC		Tod_To_Sec(TOD_To_Sec=> , InTOD:=);

②相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
InTOD	输入时刻	TOD	遵照数据类型	TOD#0:0:0	输入时刻

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
TOD_To_Sec_1	转换结果-秒	LINT	0 ～ 86399	-	转换结果-秒

数据类型

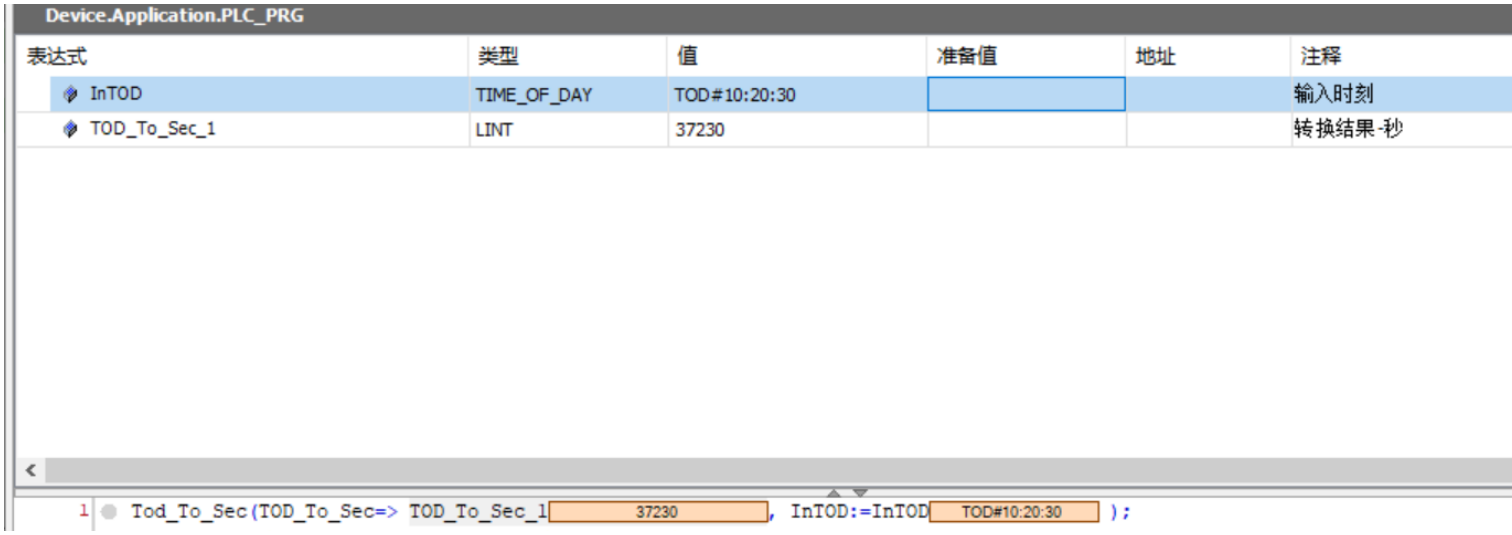
	布尔	位串				整数							实数		时刻、持续时间 日期、字符串					
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
InTOD																		√		
TOD_To_Sec_1													√							

③功能说明

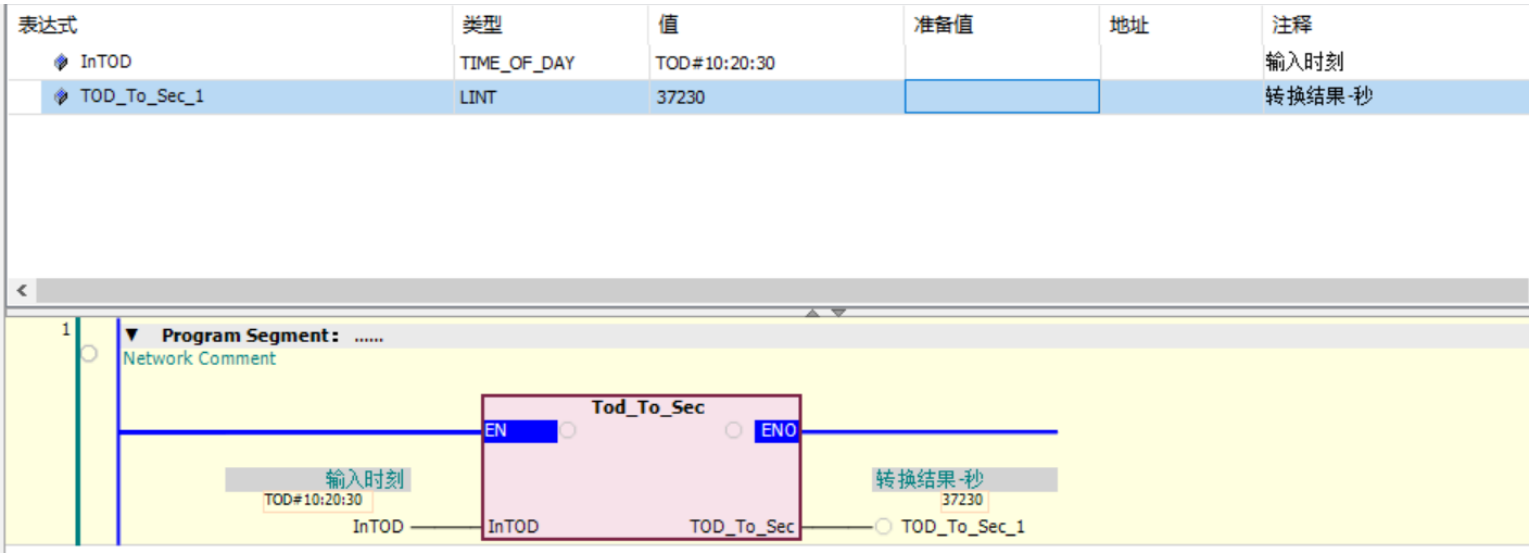
设置输入时刻,输出转换后的秒

④程序示例

ST语言

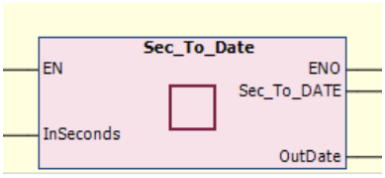


LD语言



3.1.31 Sec_To_Date 秒数转换为日期

①指令格式

指令	功能	FB/FC	LD 表现	ST表现
Sec_To_Date	秒数转换为日期	FC		Sec_To_Date(Sec_To_DATE=> , InSeconds:= , OutDate=>);

②相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
InSeconds	秒数	LINT	0 ～ 4294967295000000	-	秒数

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
OutDate	转换结果	DATE	遵照数据类型	-	转换结果

数据类型

	布尔	位串				整数							实数		时刻、持续时间 日期、字符串					
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
InSeconds													√							
OutDate																	√			

③功能说明

设置输入日期时刻结构体,输出合并后的日期时刻

④程序示例

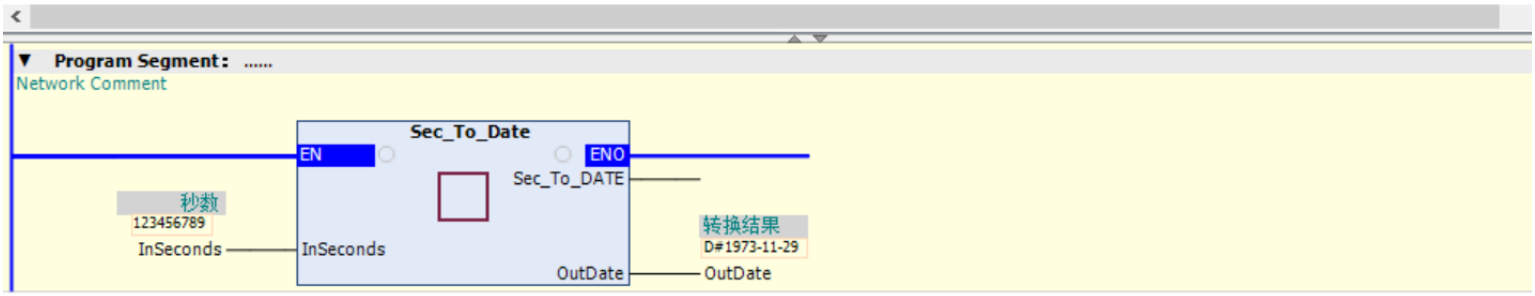
ST语言

表达式	类型	值	准备值	地址	注释
InSeconds	LINT	123456789			秒数
OutDate	DATE	D#1973-11-29			转换结果

```
1  Sec_To_Date(Sec_To_DATE=> , InSeconds:= InSeconds 123456789 , OutDate=>OutDate D#1973-11-29 );
```

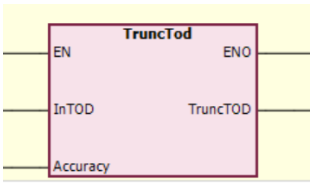
LD语言

表达式	类型	值	准备值	地址	注释
InSeconds	LINT	123456789			秒数
OutDate	DATE	D#1973-11-29			转换结果



3.1.32 TruncTod 时刻舍去

①指令格式

指令	功能	FB/FC	LD 表现	ST表现
TruncTod	时刻舍去	FC		TruncTod(TruncTOD=> , InTOD:= , Accuracy:=);;

②相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
InTOD	输入时刻	TOD	遵照数据类型	-	秒数
Accuracy	舍去单位	eSUBSECOND	-	-	舍去单位

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
TruncTOD_1	舍去后的时刻	TOD	遵照数据类型	–	转换结果

数据类型

	布尔	位串	整数	实数	时刻、持续时间 日期、字符串

	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
InTOD																		√		
Accuracy																	√			
TruncTOD_1																		√		

③功能说明

设置输入日期时刻结构体,输出合并后的日期时刻

④程序示例

ST语言

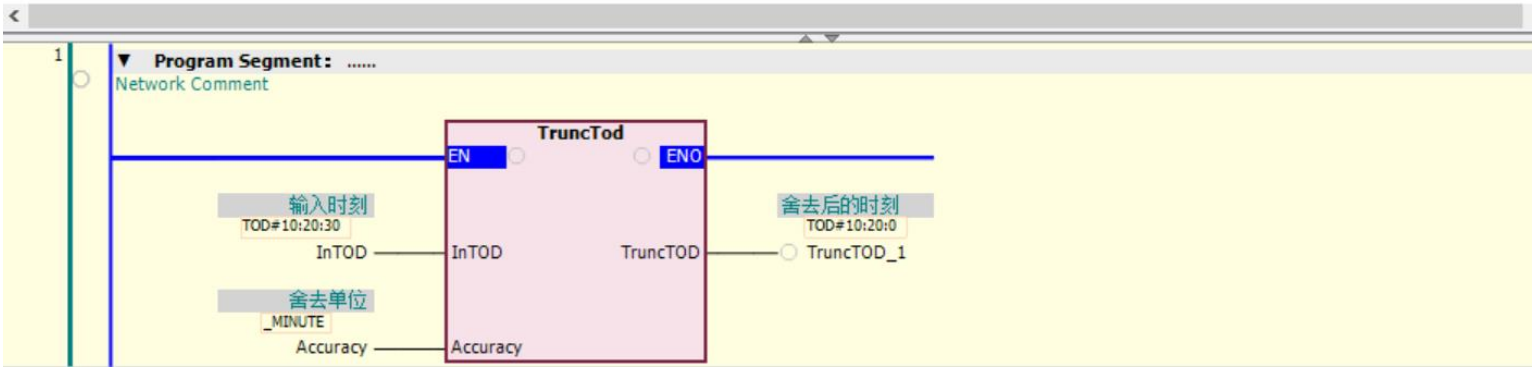
表达式	类型	值	准备值	地址	注释
InTOD	TIME_OF_DAY	TOD#10:20:30			输入时刻
Accuracy	_ESUBSECOND	_MINUTE			舍去单位
TruncTOD_1	TIME_OF_DAY	TOD#10:20:0			舍去后的时刻

```
1  TruncTod(TruncTOD=>TruncTOD_1 TOD#10:20:0 , InTOD:=InTOD TOD#10:20:30 , Accuracy:=Accuracy _MINUTE );
```

LD语言

Device.Application.POU

表达式	类型	值	准备值	地址	注释
InTOD	TIME_OF_DAY	TOD#10:20:30			输入时刻
Accuracy	_ESUBSECOND	_MINUTE			舍去单位
TruncTOD_1	TIME_OF_DAY	TOD#10:20:0			舍去后的时刻



3.2 字符串指令

指令列表

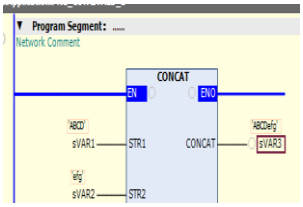
指令类别	名称	FB/FC	功能

字符串操作	CONCAT	FC	字符串连接
	DELETE	FC	字符串删除
	FIND	FC	字符串查找
	INSERT	FC	字符串插入
	REPLACE	FC	字符串替换
	LEFT	FC	从左边取字符串
	LEN	FC	获取字符串长度
	MID	FC	从中间取字符串
	RIGHT	FC	从右边取字符串
	AryToString	FC	将 BYTE 型数组转换为字符串
	HexStringToNum_DINT	FC	将 16 进制字符串格式转换为整数 DINT
	HexStringToNum_INT	FC	将 16 进制字符串格式转换为整数 INT
	将 16 进制字符串格式转换为整数 INT	FC	将 16 进制字符串格式转换为整数 SINT
	HexStringToNum_UDINT	FC	将 16 进制字符串格式转换为整数 UDINT
	HexStringToNum_UINT	FC	将 16 进制字符串格式转换为整数 UINT
	HexStringToNum_USINT	FC	将 16 进制字符串格式转换为整数 USINT
	NumToDecString	FC	将整数转换为固定长度的 10 进制字符串格式

	NumToHexString	FC	将整数转换为固定长度的 16 进制字符串格式
	StringToAry	FC	将字符串转换为 BYTE 型数组

3.2.1 CONCAT 字符串拼接

①指令格式

指令	功能	FB/FC	LD 表现	ST表现
CONCAT	将输入两字符串连接并从右侧输出。	FC		CONCAT(CONCAT=> , STR1:=sVAR1 , STR2:= sVAR2);

②相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
STR1	源数据 1	STRING	-	‘’	字符串高位
STR2	源数据 2	STRING	-	‘’	字符串低位

输出变量

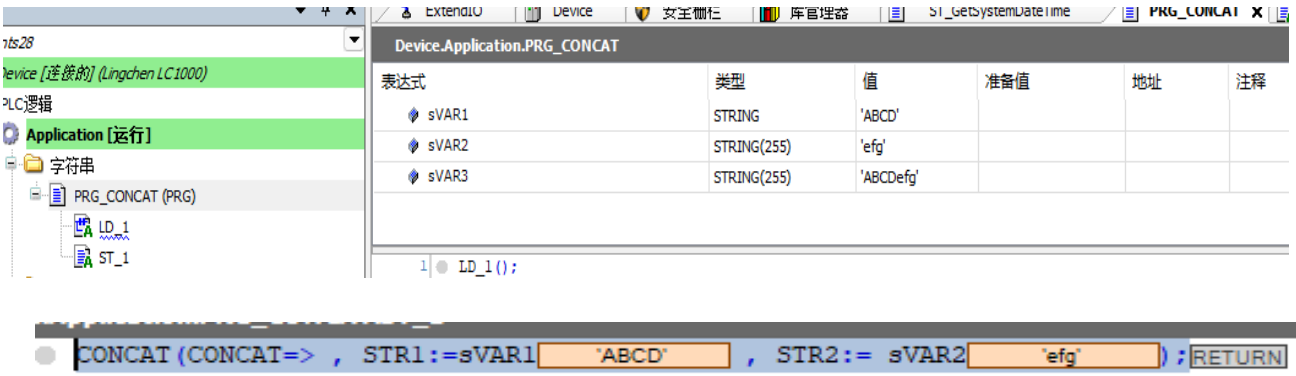
输出变量	名称	数据类型	有效范围	初始值	描述
CONCAT	返回值	STRING	_	‘’	合成的目标字符串

数据类型	布尔	位串				整数								实数		时刻、持续时间 日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATA	TOD	DT	STRING
STR1																				√
STR2																				√
CONCAT																				√

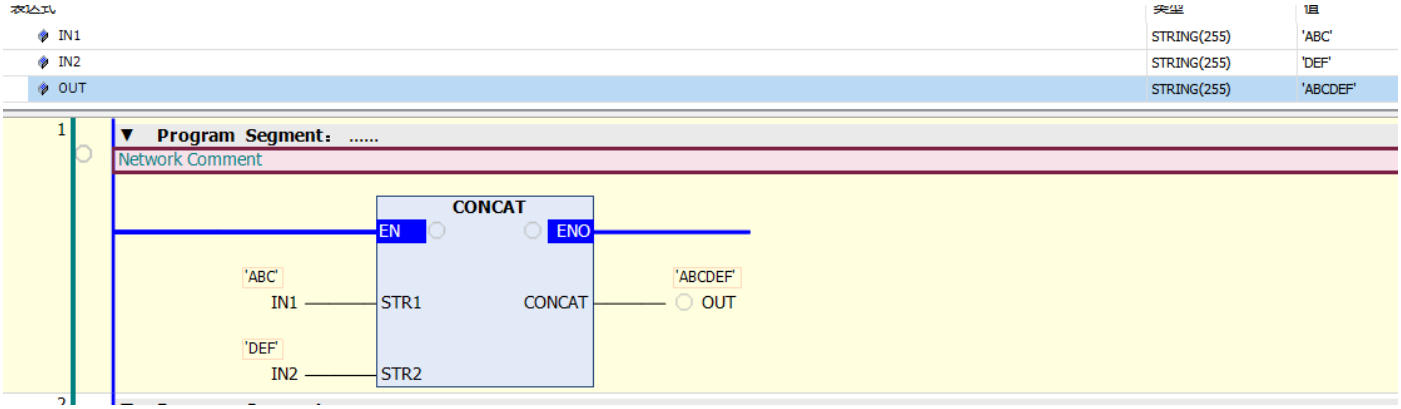
③程序示例

将输入字符串 STR1 与 STR2 进行连接，结果作为CONACT输出。

ST:



LD:

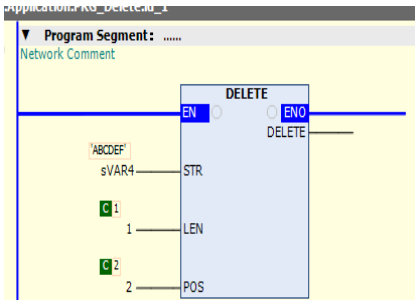


※注意事项

若拼接后字符串长度大于 255，将不会有输出。

3.2.2 Delete 字符串删除

①指令格式

指令	功能	FB/FC	LD 表现	ST表现
Delete	将输入字符串中的指定位置指定长度字符串删除，输出处理后字符串。	FC		DELETE(DELETE=>sVAR5 , STR:=sVAR4 , LEN:=1 , POS:=2);

②相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
STR	源数据	STRING	-	‘’	字符串
LEN	数据长度	Int	0~255	0	数据长度
POS	数据位置	Int	0~255	0	数据位置

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Delete	返回值	STRING	_	‘’	删除后的目标字符串

数据类型	布尔	位串				整数								实数		时刻、持续时间 日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATA	TOD	DT	STRING
STR																				√
POS											√									
LEN											√									
DELETE																				√

③程序示例

在输入字符串的 POS 位置后面删除 LEN 长度的字符串，生成一个新的字符串作为返回值，结果作为Delete输出。

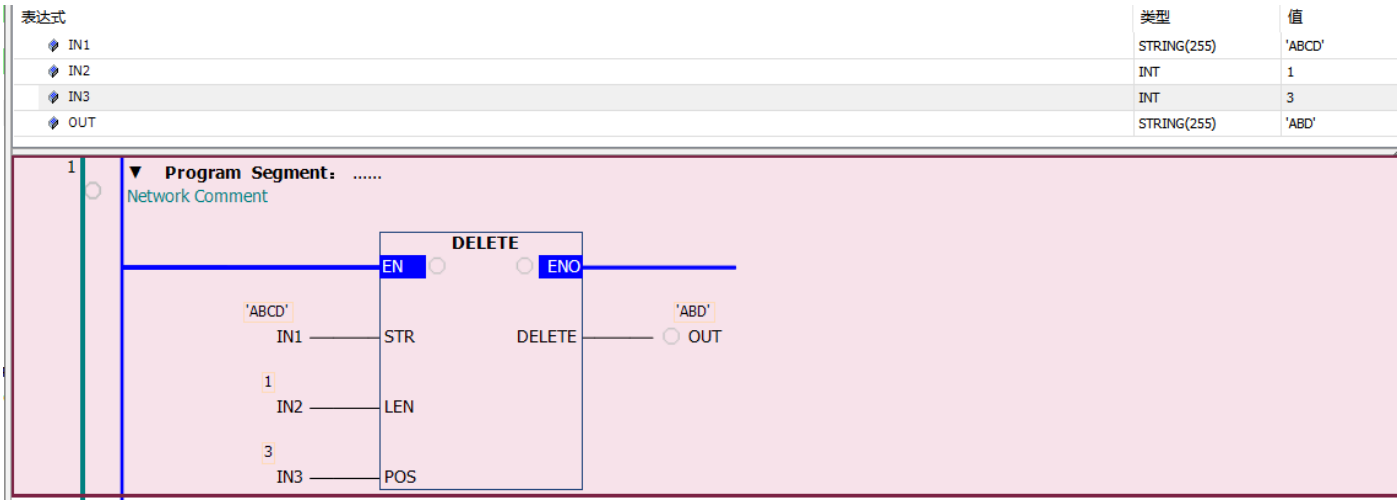
ST:

Device.Application.PRG_Delete

表达式	类型	值	准备值	地址	注释
sVAR4	STRING	'ABCDEF'			
sVAR5	STRING	'ACDEF'			
uiVAR2	INT	0			
uiVAR3	INT	0			

```
1  DELETE(DELETE=>sVAR5 'ACDEF' , STR:=sVAR4 'ABCDEF' , LEN:=1 , POS:=2 );RETURN
```

LD:

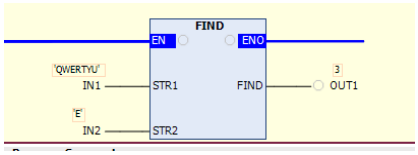


※注意事项

若POS=0，则从第一个字符开始删除。

3.2.3 FIND　字符串查找

①指令格式

指令	功能	FB/FC	LD 表现	ST表现
FIND	将STR2中的数据在STR1中寻找，将找到的数据的位置给到FIND。	FC		FIND(FIND=>OUT1 , STR1:=IN1 , STR2:=IN2);

②相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
STR1	源数据	STRING	-	‘’	用于搜索的目标字符串
STR2	查找数据	STRING	-	‘’	要查找位置的子字符串

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
FIND	查找位置	INT	0_255	0	STR1 中 STR2 首次出现的位置，未找到则为 0

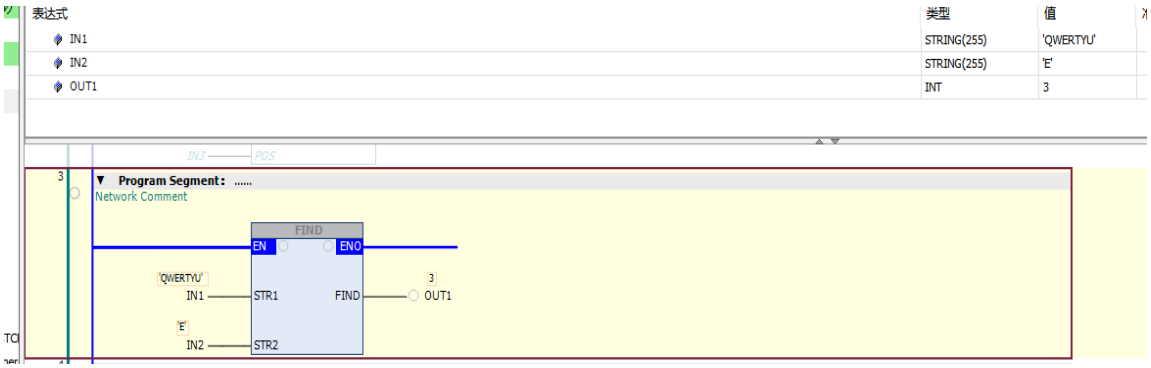
数据类型	布尔	位串				整数								实数		时刻、持续时间 日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATA	TOD	DT	STRING
STR1																				√
STR2																				√
FIND											√									

③程序示例

在输入字符串 STR1 中寻找 STR2 的值，找到相同的值后将地址位给到输出 FIND 中，如果找不到对应的值则 FIND 为0。

ST:

LD:

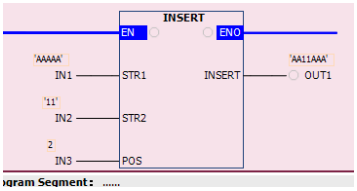


※注意事项

若STR1中不存在STR2的值，将不会有输出。

3.2.4 INSERT　字符串插入

①指令格式

指令	功能	FB/FC	LD 表现	ST表现
INSERT	将STR2中的值根据POS的位置插入到STR1中给到INSERT中	FC		INSERT(INSERT=>OUT1 , STR1:=IN1 , STR2:=IN2 , POS:=IN3);

②相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
STR1	源数据1	STRING	-	‘’	将 STR2 插入其中的字符串
STR2	插入数据	STRING	-	‘’	要插入到 STR1 中的字符串
POS	插入位置	INT	0_255	0	插入位置。如果 POS 大于 255 或小于 0，则结果为 STR1″0；在第一个字符 “1” 之前插入；在第一个字符之后插入。

输出变量

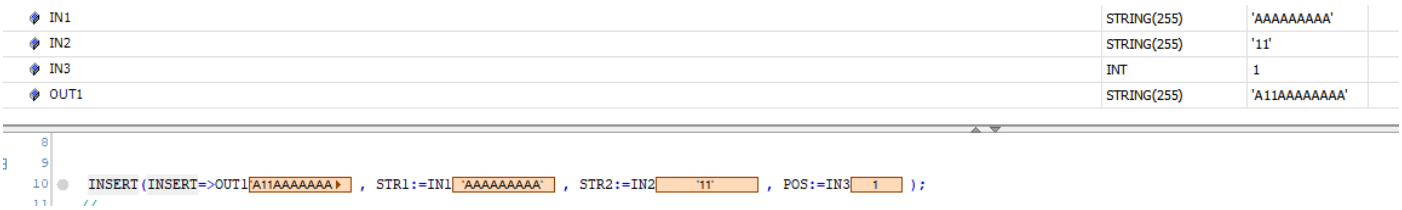
输出变量	名称	数据类型	有效范围	初始值	描述
INSERT	结果字符串	STRING	-	‘’	插入后的结果字符串

数据类型	布尔	位串				整数								实数		时刻、持续时间 日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATA	TOD	DT	STRING
STR1																				√
STR2																				√
POS											√									
INSERT																				√

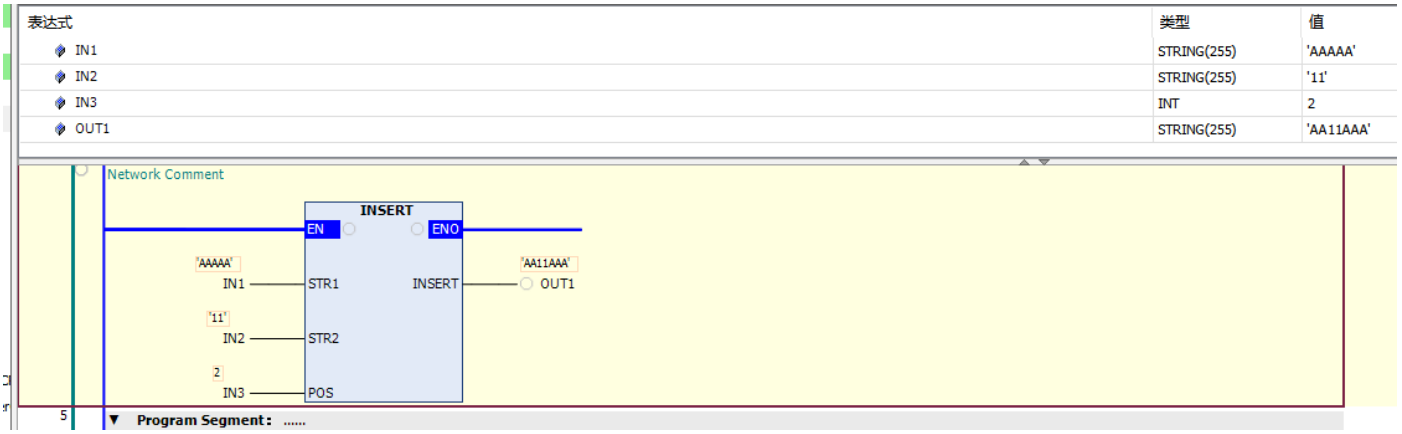
③程序示例

在输入字符串 SRR2 根据 POS 变量插入到 SRR1 中，最终得到的结果给到 INSERT 。

ST:



LD:



※注意事项

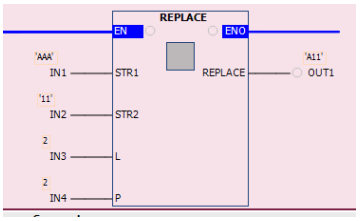
如果 POS 大于 255 或小于 0，则结果为 ‘STR2+STR1’。

若 POS 不小于字符串长度，则输出结果为 STR1 对应的数据。

3.2.5 REPLACE 字符串替换

①指令格式

指令	功能	FB/FC	LD 表现	ST表现

REPLACE	将STR2变量中的数据根据L和P来和STR1中的数据进行替换，将替换完的数据给到REPLACE中	FC		REPLACE(REPLACE=>OUT1 , STR1:=IN1 , STR2:=IN2 , L:=IN3 , P:=IN4);

②相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
STR1	源数据1	STRING	-	‘’	被替换部分的原字符串
STR2	源数据2	STRING	-	‘’	用于替换的字符串
L	字符数量	INT	0_255	0	从左数的字符数量
P	替换字符起始位置	INT	0_255	0	要替换的字符起始位置，=1或=e表示第一个字符

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
REPLACE	结果字符串	STRING	-	‘’	替换后的结果字符串

数据类型	布尔	位串				整数								实数		时刻、持续时间 日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATA	TOD	DT	STRING
STR1																				√
STR2																				√
L											√									
P											√									
REPLACE																				√

③程序示例

将STR2变量中的数据根据要在第几位和要替换几位来和STR1中的数据进行替换，将替换完的数据给到REPLACE中

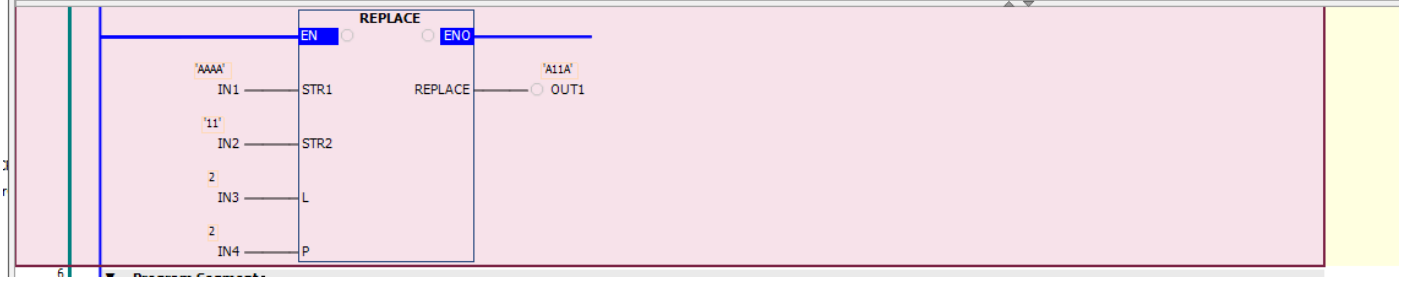
ST:

表达式	类型	值
IN1	STRING(255)	'AAAA'
IN2	STRING(255)	'11'
IN3	INT	1
IN4	INT	2
OUT1	STRING(255)	'A11AA'

```
12
13  REPLACE(
14      REPLACE=>OUT1 'A11AA' ,
15      STR1:=IN1 'AAAA' ,
16      STR2:=IN2 '11' ,
17      L:=IN3 1 ,
18      P:=IN4 2 );
19  //
```

LD:

IN1	STRING(255)	'AAAA'
IN2	STRING(255)	'11'
IN3	INT	2
IN4	INT	2
OUT1	STRING(255)	'A11A'

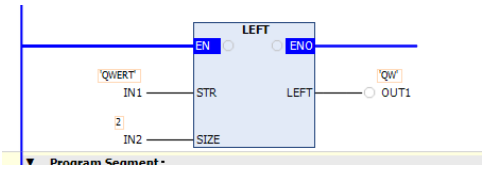


※注意事项

3.2.6 LEFT　从左边取字符串

①指令格式

指令	功能	FB/FC	LD 表现	ST表现

LEFT	获取字符串左侧开始设定位数的当前值	FC		LEFT(LEFT=>OUT1 , STR:=IN1 , SIZE:=IN2);

②相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
STR	源数据	STRING	-	‘’	被查询的字符串
SIZE	读取位数	INT	0_255	0	从左侧开始查询的位数

输出变量

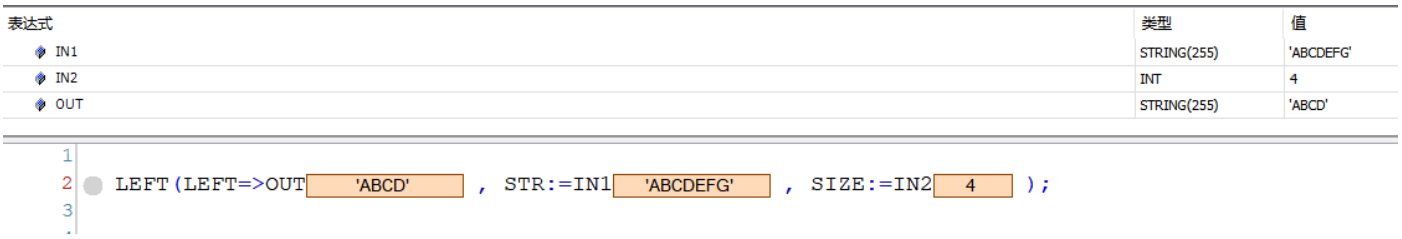
输出变量	名称	数据类型	有效范围	初始值	描述
LEFT	读取到的字符	STRING	-	‘’	最终得到的字符内容

数据类型	布尔	位串				整数								实数		时刻、持续时间 日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATA	TOD	DT	STRING
STR																				√
SIZE											√									
LEFT																				√

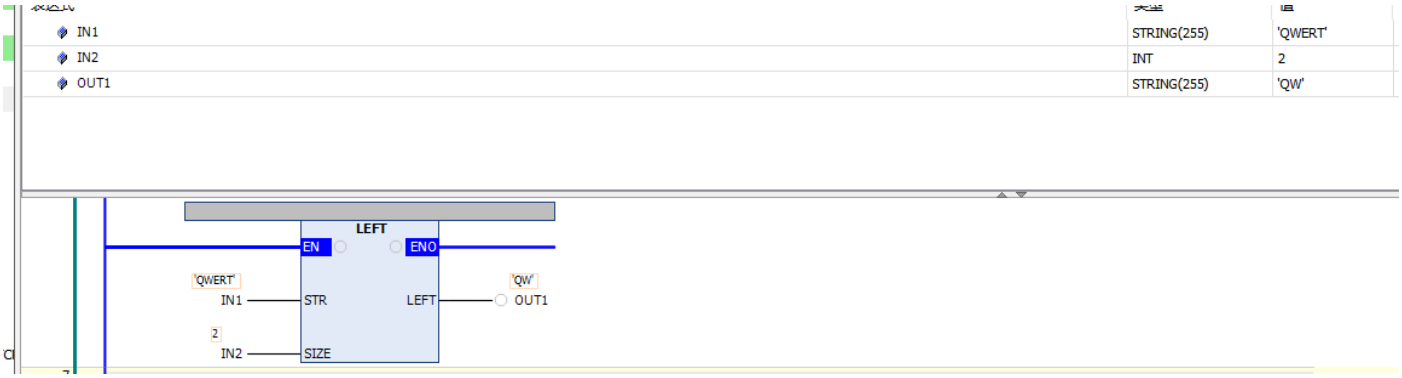
③程序示例

在STR中的左边第SIZE位(包括前面的数据)获取到的值给到LEFT

ST:



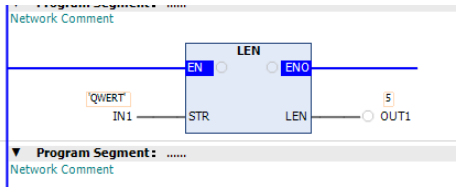
LD:



※注意事项 此指令是获取设定位数之前的所有数据并非单一字符

3.2.7 LEN 获取字符串长度

①指令格式

指令	功能	FB/FC	LD 表现	ST表现
LEN	获取字符串所用长度	FC		LEN(LEN=>OUT1 , STR:=IN1);

②相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
STR	源数据	STRING	-	‘’	被查询的字符串

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述

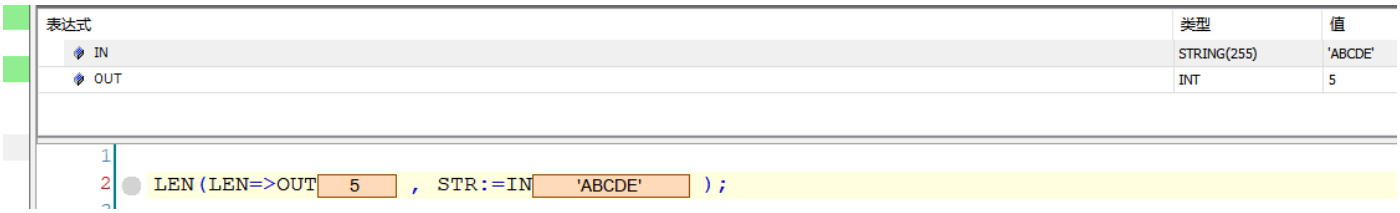
LEN	读取到的长度	INT	0-255	0	读取到的长度

数据类型	布尔	位串				整数								实数		时刻、持续时间 日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATA	TOD	DT	STRING
STR																				√
LEN											√									

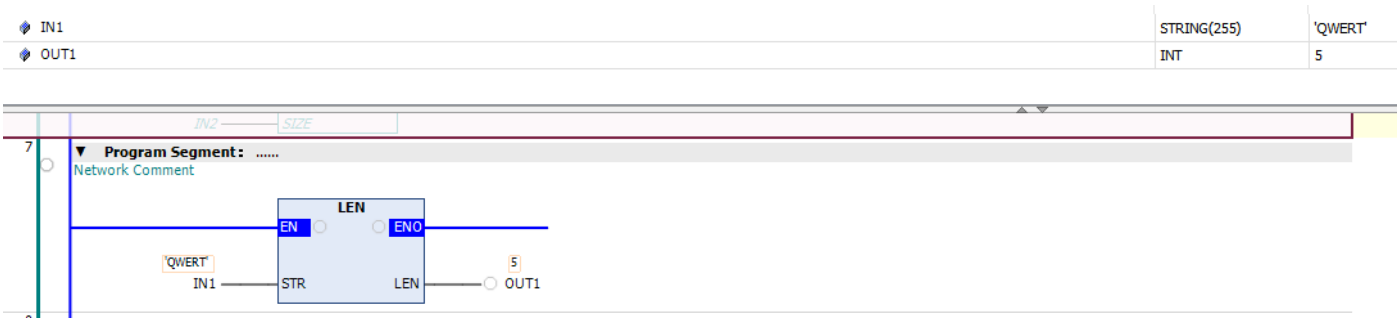
③程序示例

读取STR中使用的长读给到LEN中

ST:



LD:

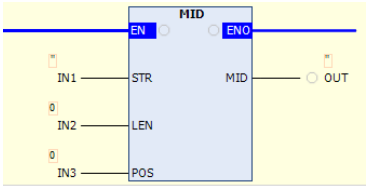


※注意事项

3.2.8 MID　　从中间取字符串

①指令格式

指令	功能	FB/FC	LD 表现	ST表现

MID	根据LEN和POS引脚在STR中取对应的值给到MID	FC		MID(MID=>OUT , STR:=IN1 , LEN:=IN2 , POS:=IN3);

②相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
STR	源数据1	STRING	-	‘’	被取值的源数据
LEN	取值数量	INT	0_255	0	要取的个数
POS	取值位置	INT	0_255	0	要取的位置

输出变量

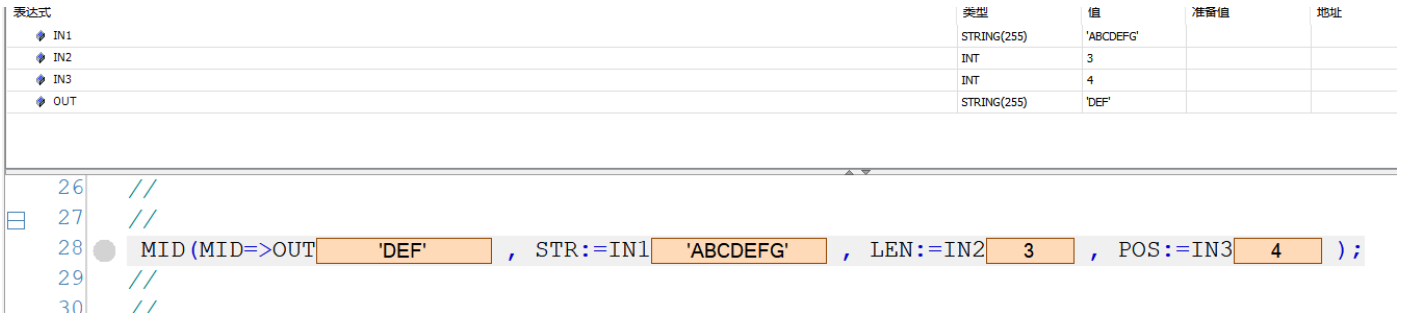
输出变量	名称	数据类型	有效范围	初始值	描述
MID	结果字符串	STRING	-	‘’	取值后的结果字符串

数据类型	布尔	位串				整数								实数		时刻、持续时间 日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATA	TOD	DT	STRING
STR																				√
LEN											√									
POS											√									
MID																				√

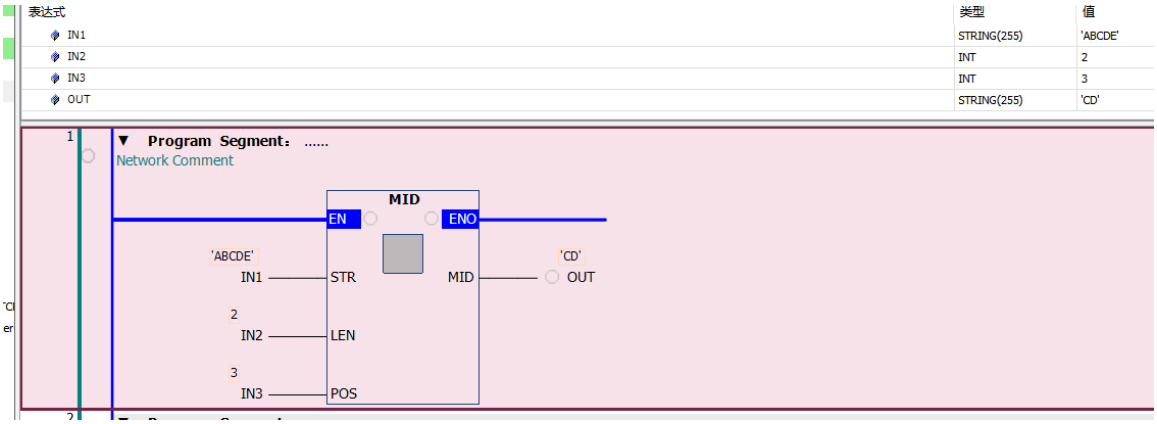
③程序示例

在输入字符串 STR 根据给到的取的位置和取几个来将拿到的值给到MID 。

ST:



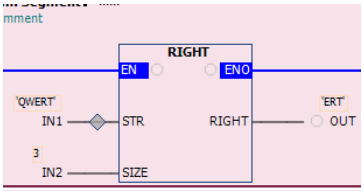
LD:



※注意事项

3.2.9 RIGHT 从右边取字符串

①指令格式

指令	功能	FB/FC	LD 表现	ST表现
RIGHT	获取字符串右侧开始设定位数的当前值	FC		RIGHT(RIGHT=>OUT , STR:=IN1 , SIZE:=IN2);

②相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
STR	源数据	STRING	-	‘’	被查询的字符串

SIZE	读取位数	INT	0_255	0	从右侧开始查询的位数

输出变量

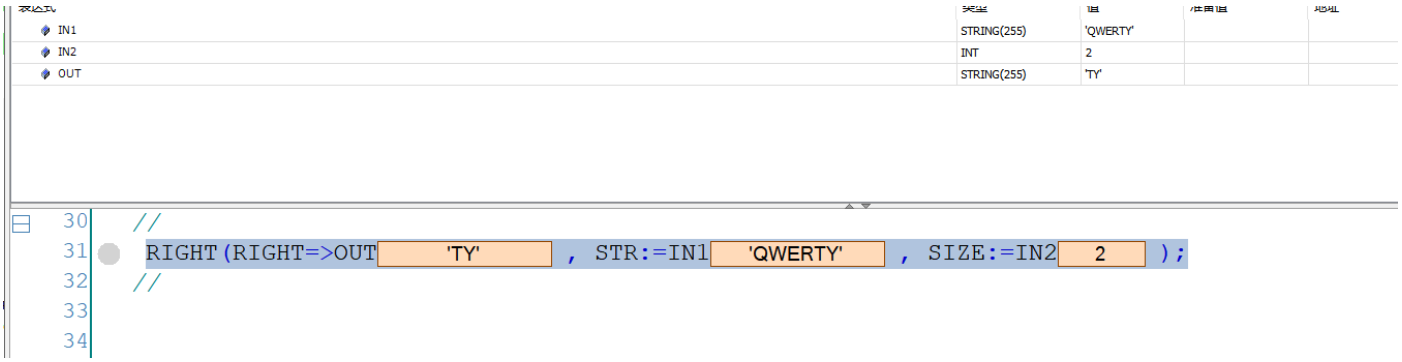
输出变量	名称	数据类型	有效范围	初始值	描述
RIGHT	读取到的字符	STRING	-	‘’	最终得到的字符内容

数据类型	布尔	位串				整数								实数		时刻、持续时间 日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATA	TOD	DT	STRING
STR																				√
SIZE											√									
RIGHT																				√

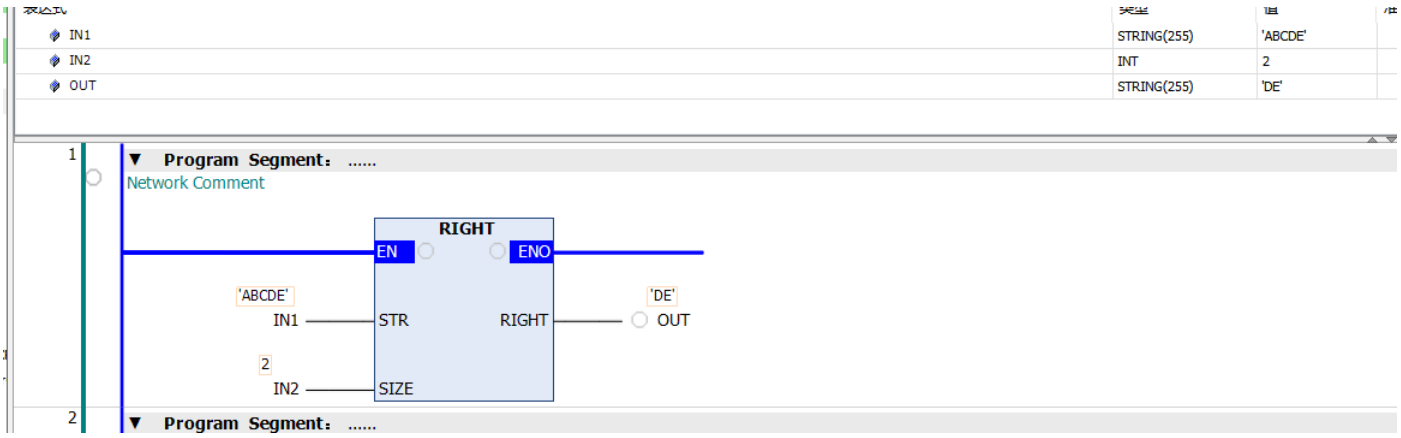
③程序示例

在STR中的右边第SIZE位(包括后面的数据)获取到的值给到RIGHT

ST:



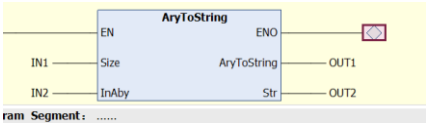
LD:



※注意事项 此指令是获取设定位数之后的所有数据并非单一字符

3.2.10 AryToString 将 BYTE 型数组转换为字符串

①指令格式

指令	功能	FB/FC	LD 表现	ST表现
AryToString	将 BYTE 型数组转换为字符串	FC		AryToString(AryToString=>OUT1 , Size:=IN1 , InAby:=IN2 , Str=>OUT2);

②相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
InAby	源数据	ARRAY[*..*]OF BYTE	-	-	BYTE型数组
Size	转换数量	UINT	0_255	0	要转换的数量

输出变量

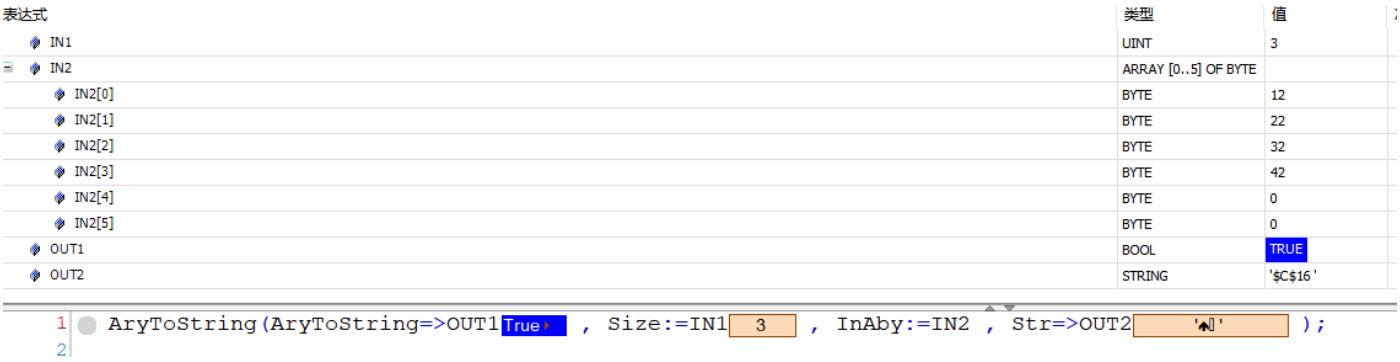
输出变量	名称	数据类型	有效范围	初始值	描述
AryToString	完成信号	BOOL	0-1	0	转换完成
Str	得到的结果	STRING	-	‘’	最终得到的字符内容

数据类型	布尔	位串				整数								实数		时刻、持续时间 日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATA	TOD	DT	STRING
InAby		√																		
Size							√													
AryToString	√																			
Str																				√

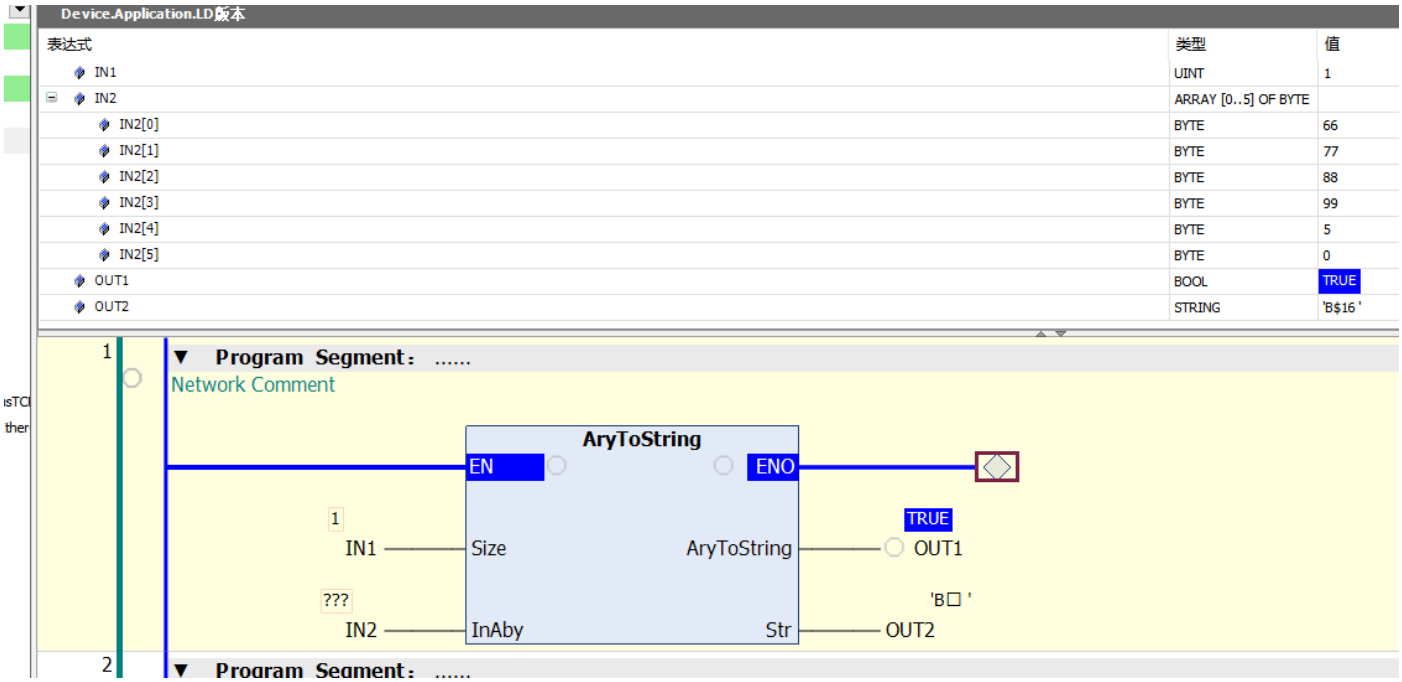
③程序示例

将InAby引脚BYTE型数组根据Size变量中的数量进行转换，转换成功后会给出个AryToString 完成信号将转换完成后的值给到Str中

ST:



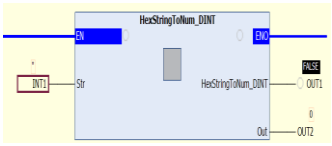
LD:



※注意事项
转换出来的值是ascii类型的

3.2.11 HexStringToNum_DINT　　将 16 进制字符串格式转换为整数 DINT

①指令格式

指令	功能	FB/FC	LD 表现	ST表现
HexStringToNum_DINT	将 16 进制字符串格式转换为整数 DINT	FC		HexStringToNum_DINT(HexStringToNum_DINT=>OUT1 , Str:=INT1 , Out=>OUT2);

②相关变量
输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Str	源数据	STRING	-	‘’	STRING类型变量

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
HexStringToNum_DINT	完成信号	BOOL	0-1	0	转换完成
Out	得到的结果	DINT	-	‘’	最终得到的DINT类型数据

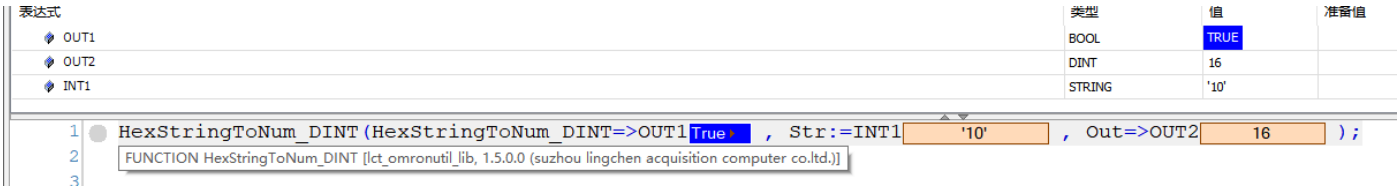
数据类型	布尔	位串				整数								实数		时刻、持续时间 日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATA	TOD	DT	STRING
Str																				√
HexStringToN	√																			

um_DINT																			
Out												√							

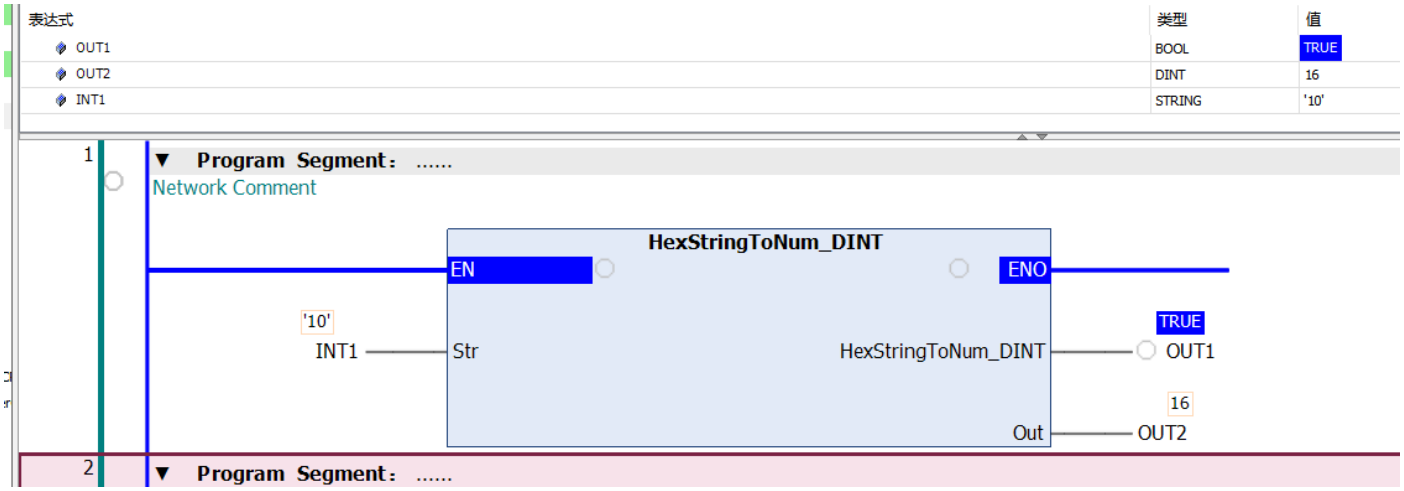
③程序示例

将STr字符STRING类型变量收到HexStringToNum_DINT完成信号后转换成DINT类型变量

ST:



LD:



※注意事项

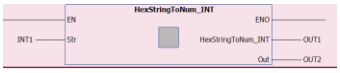
STr字符STRING类型变量只能是阿拉伯数字才能正常转换

STr字符STRING类型变量为16# Out变量收到的是10#

3.2.12 HexStringToNum_INT　　将 16 进制字符串格式转换为整数 INT

①指令格式

指令	功能	FB/FC	LD 表现	ST表现

HexStringToNum_INT	将 16 进制字符串格式转换为整数 INT	FC		HexStringToNum_INT(HexStringToNum_INT=>OUT1 , Str:=INT1 , Out=>OUT2);

②相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Str	源数据	STRING	-	‘’	STRING类型变量

输出变量

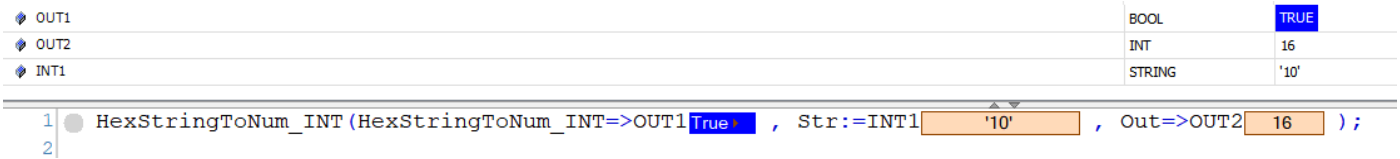
输出变量	名称	数据类型	有效范围	初始值	描述
HexStringToNum_INT	完成信号	BOOL	0-1	0	转换完成
Out	得到的结果	INT	-	‘’	最终得到的INT类型数据

数据类型	布尔	位串				整数								实数		时刻、持续时间 日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATA	TOD	DT	STRING
Str																				√
HexStringToNum_INT	√																			
Out											√									

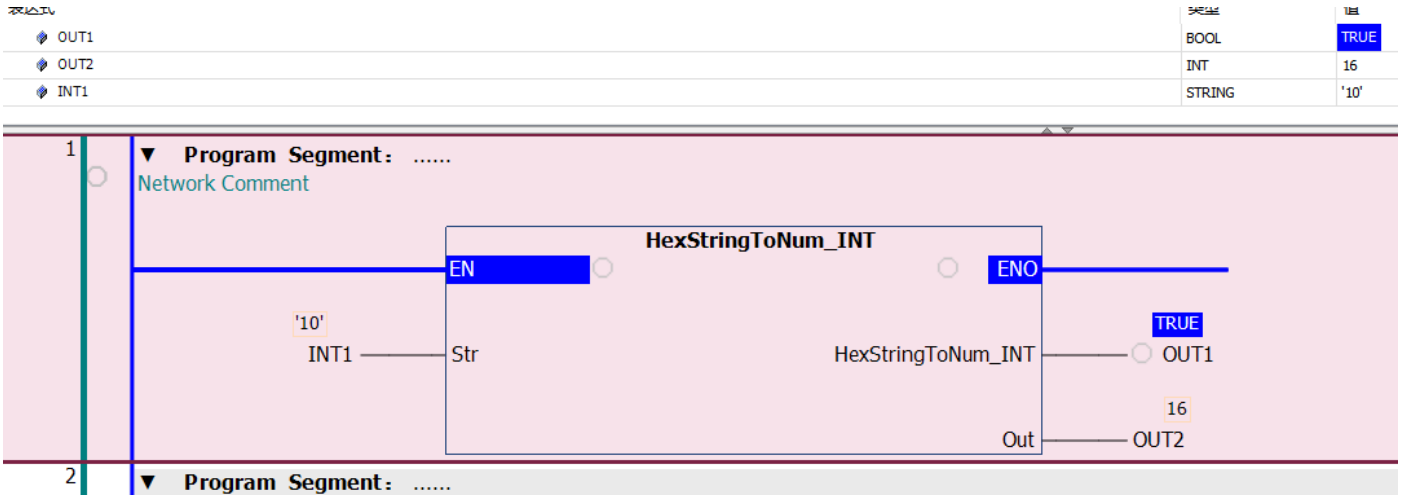
③程序示例

将STr字符STRING类型变量收到HexStringToNum_INT完成信号后转换成INT类型变量

ST:



LD:



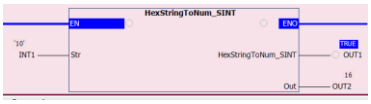
※注意事项

STr字符STRING类型变量只能是阿拉伯数字才能正常转换

STr字符STRING类型变量为16# Out变量收到的是10#

3.2.13 HexStringToNum_SINT　　将 16 进制字符串格式转换为整数 SINT

①指令格式

指令	功能	FB/FC	LD 表现	ST表现
HexStringToNum_SINT	将 16 进制字符串格式转换为整数 SINT	FC		HexStringToNum_SINT(HexStringToNum_SINT=>OUT1 , Str:=INT1 , Out=>OUT2);

②相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Str	源数据	STRING	-	‘’	STRING类型变量

输出变量

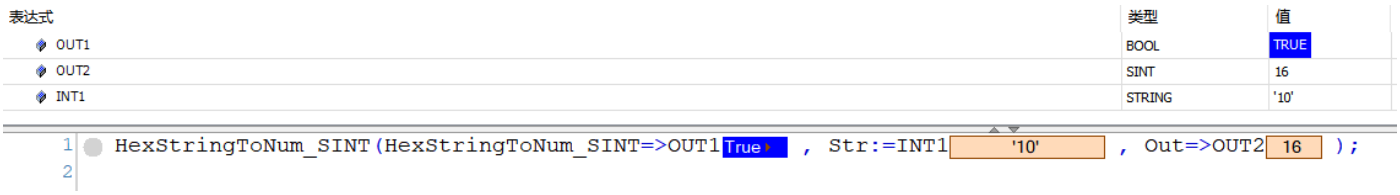
输出变量	名称	数据类型	有效范围	初始值	描述
HexStringToNum_SINT	完成信号	BOOL	0-1	0	转换完成
Out	得到的结果	INT	-	‘’	最终得到的SINT类型数据

数据类型	布尔	位串				整数								实数		时刻、持续时间 日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATA	TOD	DT	STRING
Str																				√
HexStringToNum_SINT	√																			
Out										√										

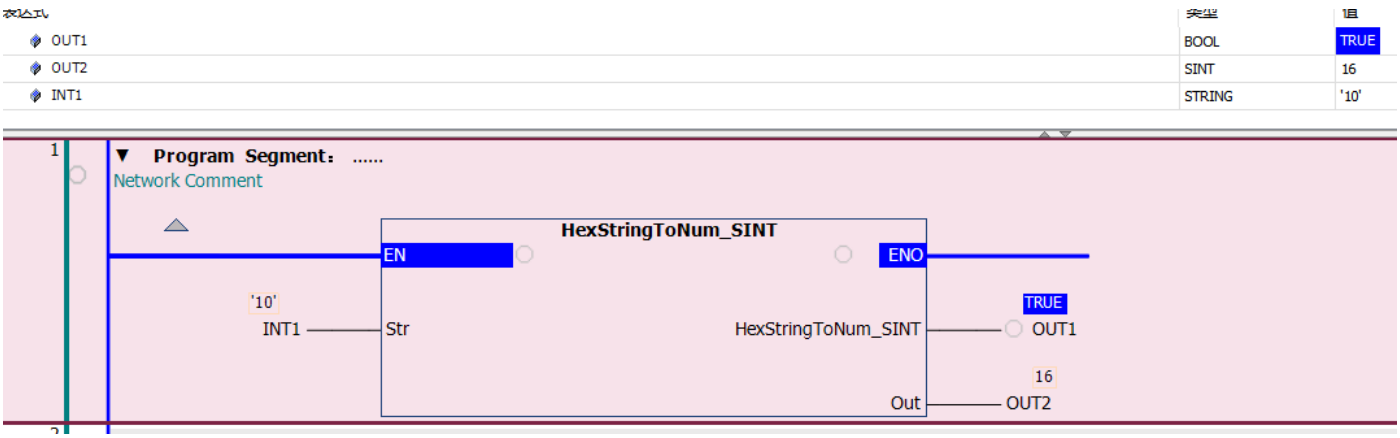
③程序示例

将STr字符STRING类型变量收到HexStringToNum_SINT完成信号后转换成INT类型变量

ST:



LD:



※注意事项

STr字符STRING类型变量只能是阿拉伯数字才能正常转换

STr字符STRING类型变量为16# Out变量收到的是10#

3.2.14 HexStringToNum_UDINT　　将 16 进制字符串格式转换为整数 UDINT

①指令格式

指令	功能	FB/FC	LD 表现	ST表现
HexStringToNum_UDINT	将 16 进制字符串格式转换为整数 UDINT	FC		HexStringToNum_UDINT(HexStringToNum_UDINT=>OUT1 , Str:=INT1 , Out=>OUT2);

②相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Str	源数据	STRING	-	‘’	STRING类型变量

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
HexStringToNum_UDINT	完成信号	BOOL	0-1	0	转换完成
Out	得到的结果	INT	-	‘’	最终得到的UDINT类型数据

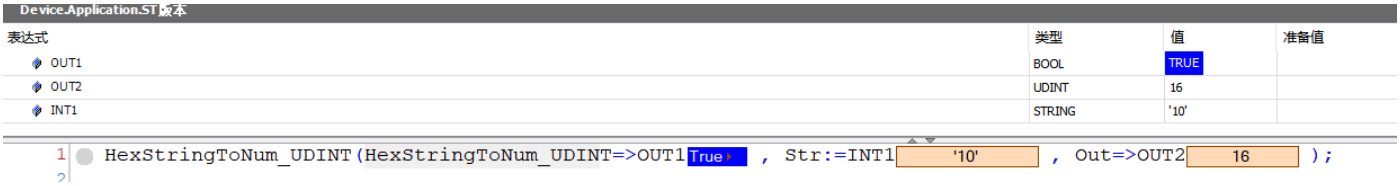
数据类型	布尔	位串				整数								实数		时刻、持续时间 日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATA	TOD	DT	STRING
Str																				√

HexStringToNum_UDINT	√																			
Out								√												

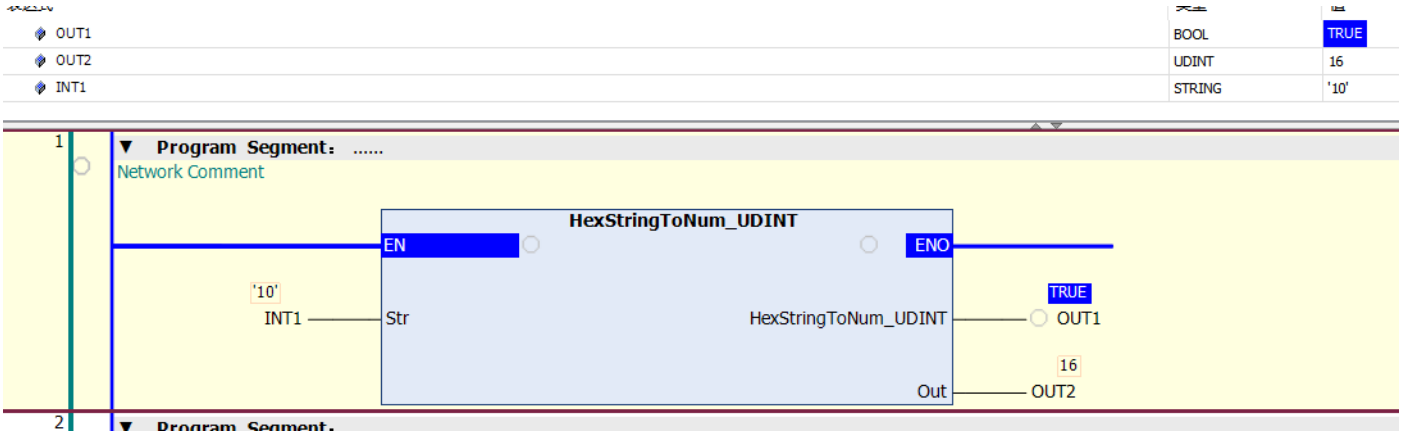
③程序示例

将STr字符STRING类型变量收到HexStringToNum_UDINT完成信号后转换成INT类型变量

ST:



LD:



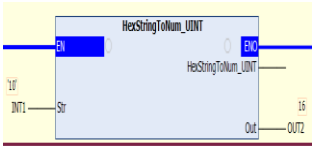
※注意事项

STr字符STRING类型变量只能是阿拉伯数字才能正常转换

STr字符STRING类型变量为16# Out变量收到的是10#

3.2.15 HexStringToNum_UINT　　将 16 进制字符串格式转换为整数 UINT

①指令格式

指令	功能	FB/FC	LD 表现	ST表现
HexStringToNum_UINT	将 16 进制字符串格式转换为整数 UINT	FC		HexStringToNum_UINT(HexStringToNum_UINT=>OUT1 , Str:=INT1 , Out=>OUT2);

②相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Str	源数据	STRING	-	‘’	STRING类型变量

输出变量

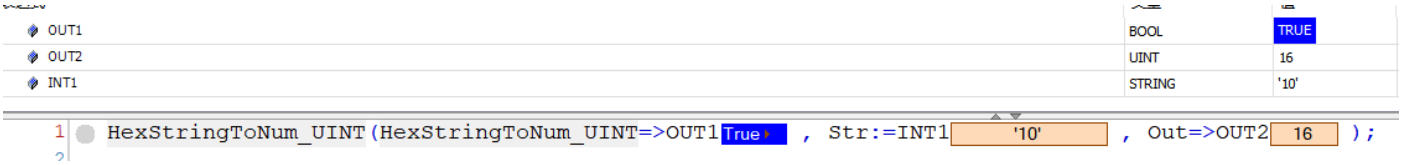
输出变量	名称	数据类型	有效范围	初始值	描述
HexStringToNum_UINT	完成信号	BOOL	0-1	0	转换完成
Out	得到的结果	INT	-	‘’	最终得到的UINT类型数据

数据类型	布尔	位串				整数								实数		时刻、持续时间 日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATA	TOD	DT	STRING
Str																				√
HexStringToNum_UINT	√																			
Out							√													

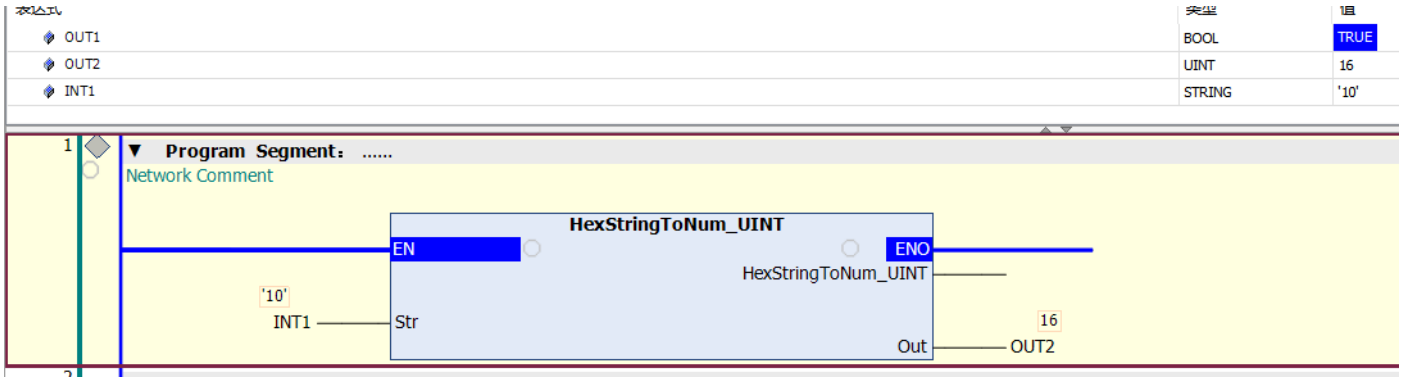
③程序示例

将STr字符STRING类型变量收到HexStringToNum_UINT完成信号后转换成UINT类型变量

ST:



LD:



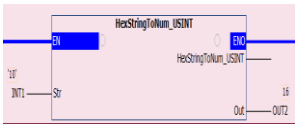
※注意事项

STr字符STRING类型变量只能是阿拉伯数字才能正常转换

STr字符STRING类型变量为16# Out变量收到的是10#

3.2.16 HexStringToNum_USINT　　将 16 进制字符串格式转换为整数 USINT

①指令格式

指令	功能	FB/FC	LD 表现	ST表现
HexStringToNum_USINT	将 16 进制字符串格式转换为整数 USINT	FC		HexStringToNum_USINT(HexStringToNum_USINT=>OUT1 , Str:=INT1 , Out=>OUT2);

②相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Str	源数据	STRING	-	‘’	STRING类型变量

输出变量

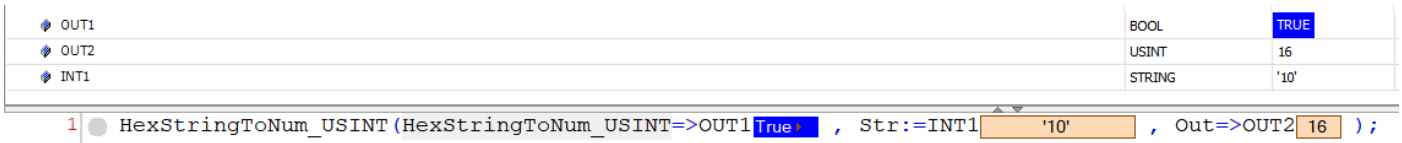
输出变量	名称	数据类型	有效范围	初始值	描述
HexStringToNum_USINT	完成信号	BOOL	0-1	0	转换完成
Out	得到的结果	INT	-	‘’	最终得到的USINT类型数据

数据类型	布尔	位串				整数								实数		时刻、持续时间 日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATA	TOD	DT	STRING
Str																				√
HexStringToNum_UINT	√																			
Out						√														

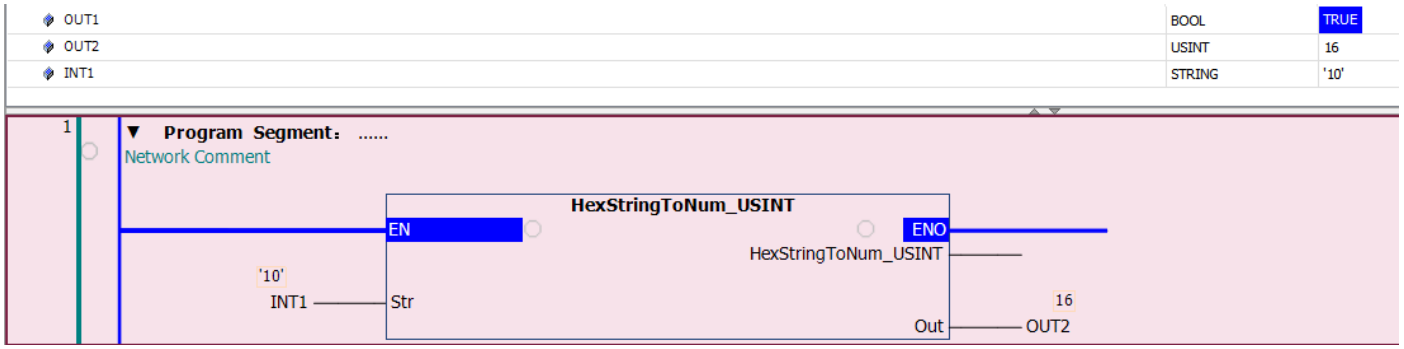
③程序示例

将STr字符STRING类型变量收到HexStringToNum_USINT完成信号后转换成USINT类型变量

ST:



LD:



※注意事项

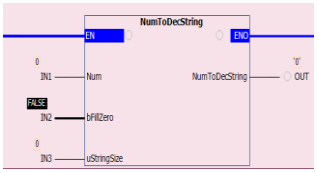
STr字符STRING类型变量只能是阿拉伯数字才能正常转换

STr字符STRING类型变量为16# Out变量收到的是10#

3.2.17 NumToDecString　　将整数转换为固定长度的 10 进制字符串格式

①指令格式

指令	功能	FB/FC	LD 表现	ST表现

NumToDecString	将整数转换为固定长度的 10 进制字符串格式	FC		NumToDecString(NumToDecString=>OUT , Num:=IN1 , bFillZero:=IN2 , uStringSize:=IN3);

②相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Num	源数据	LINT	-	0	LINT类型被转换数据
bFillZero	是否补零	BOOL	0-1	0	为1时在取uStringSize个字符数大于Num后在最前方以0代替不够的位数
uStringSize	字符数	USINT	-	0	转换的字符数

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
NumToDecString	得到的结果	STRING	-	‘’	最终得到的STRING类型数据

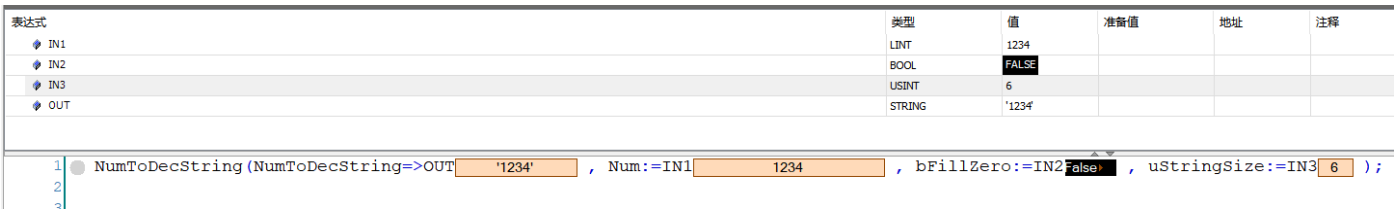
数据类型	布尔	位串				整数								实数		时刻、持续时间 日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATA	TOD	DT	STRING
Num													√							
bFillZero	√																			
uStringSize						√														

NumToDecString																			√

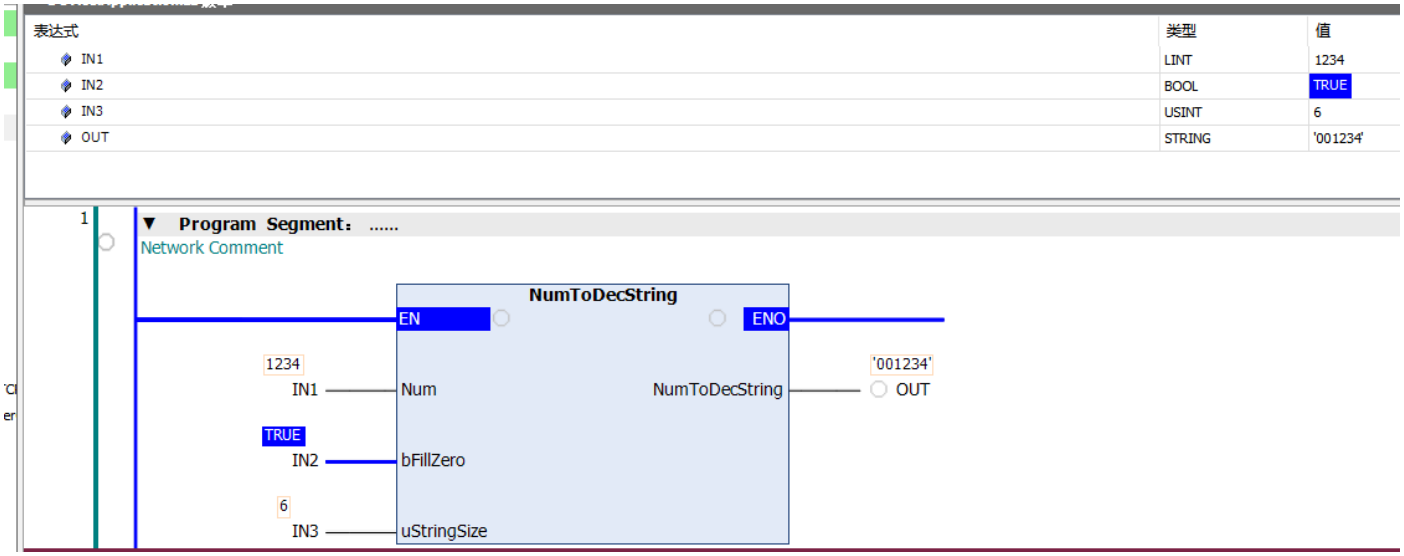
③程序示例

将Num引脚LINT类型数据判断需要转换uStringSize位（如果Num位数低于uStringSize可以选择bFillZero是否在数据最前端补零）给到NumToDecString引脚STRING类型

ST:（未补零）



LD:（补零）



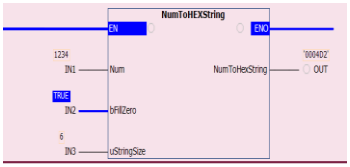
※注意事项

转换完成后NumToDecString收到的是10#类型的数据

3.2.18 NumToHEXString　　将整数转换为固定长度的 16 进制字符串格式

①指令格式

指令	功能	FB/FC	LD 表现	ST表现

NumToHEXString	将整数转换为固定长度的 16 进制字符串格式	FC		NumToHEXString(NumToHEXString=>OUT , Num:=IN1 , bFillZero:=IN2 , uStringSize:=IN3);

②相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Num	源数据	ULINT	-	0	ULINT类型被转换数据
bFillZero	是否补零	BOOL	0-1	0	为1时在取uStringSize个字符数大于Num后在最前方以0代替不够的位数
uStringSize	字符数	USINT	-	0	转换的字符数

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
NumToHEXString	得到的结果	STRING	-	‘’	最终得到的STRING类型数据

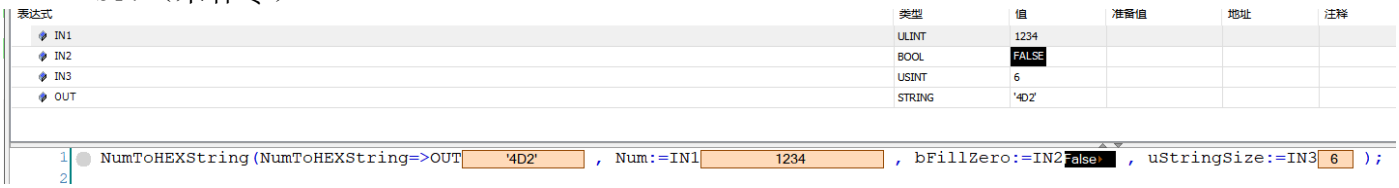
数据类型	布尔	位串				整数								实数		时刻、持续时间 日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATA	TOD	DT	STRING
Num									√											

bFillZero	√																		
uStringSize						√													
NumToHEXString																			√

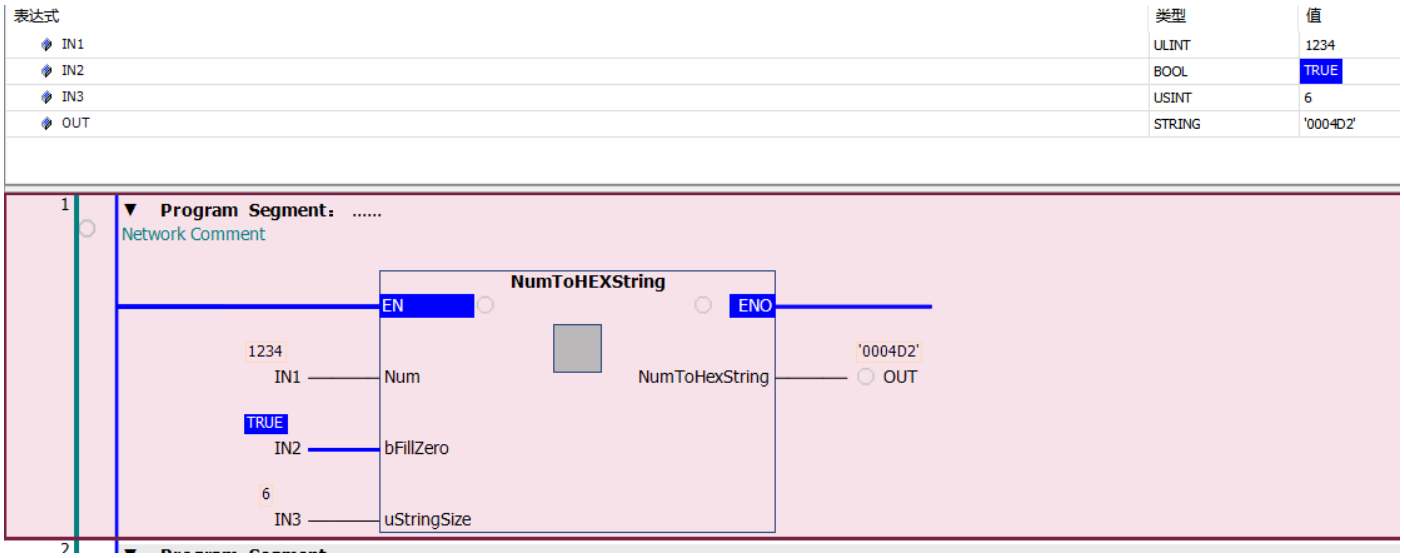
③程序示例

将Num引脚ULINT类型数据判断需要转换uStringSize位（如果Num位数低于uStringSize可以选择bFillZero是否在数据最前端补零）给到NumToDecString引脚STRING类型

ST:（未补零）



LD:（补零）



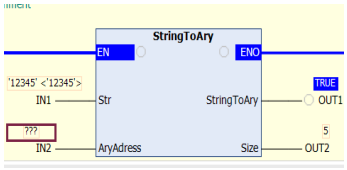
※注意事项

转换完成后NumToDecString收到的是16#类型的数据

3.2.19 StringToAry 将字符串转换为 BYTE 型数组

①指令格式

指令	功能	FB/FC	LD 表现	ST表现

StringToAry	将字符串转换为 BYTE 型数组	FC		StringToAry(StringToAry=>OUT1 , Str:=IN1 , AryAdress:=IN2 , Size=>OUT2);

②相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Str	源数据	STRING	-	‘’	STRING型变量
AryAdress	转换地址	ARRAY[*..*]OF BYTE	-	-	转换地址BYTE类型数组

输出变量

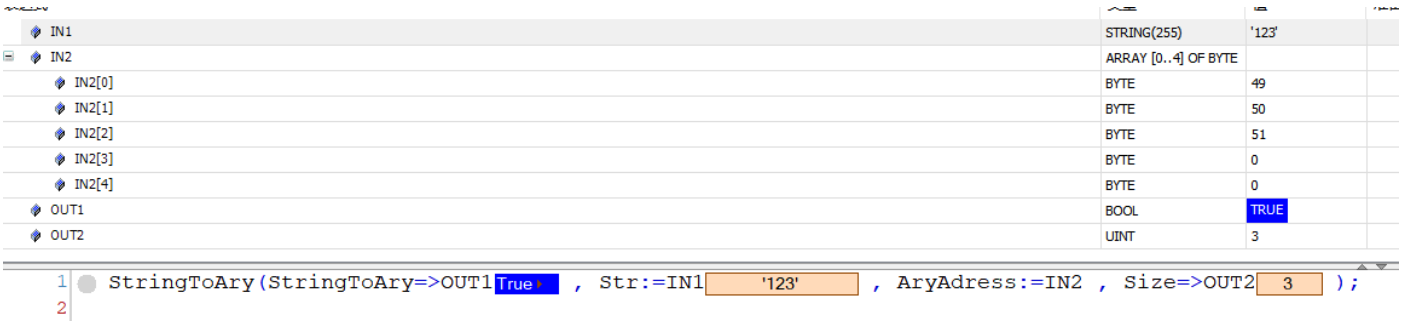
输出变量	名称	数据类型	有效范围	初始值	描述
StringToAry	完成信号	BOOL	0-1	0	转换完成
Size	转换的数量	UINT	-	0	转换完成得到的数量

数据类型	布尔	位串				整数								实数		时刻、持续时间 日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATA	TOD	DT	STRING
Str																				√
AryAdress		√																		
StringToAry	√																			
Size							√													

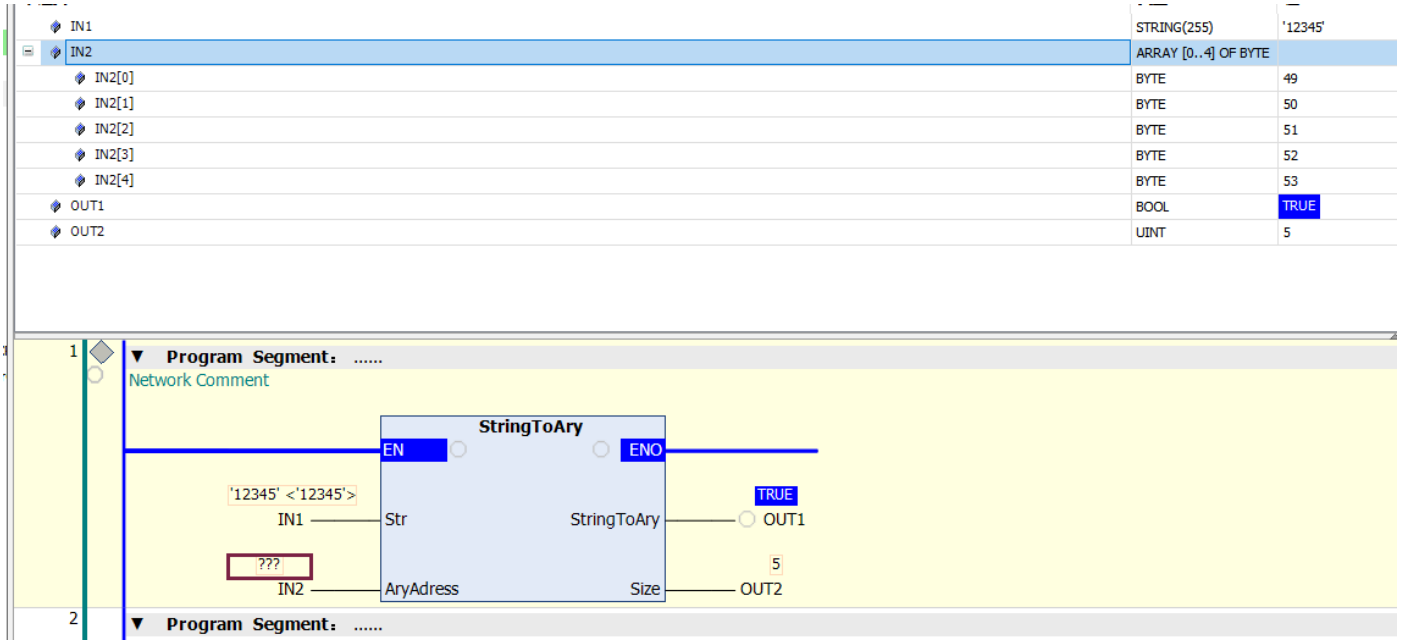
③程序示例

将Str引脚STRING型数据进行转换，转换到AryAdress引脚BYTE类型数组，转换成功后会给出个StringToAry完成信号将转换完成得到的转换数量给到Size中

ST:



LD:



※注意事项

转换出来的值是ascii类型的

3.3 数组逻辑相关

指令列表

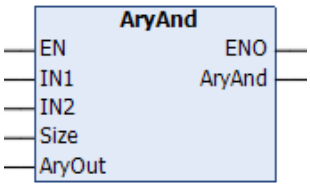
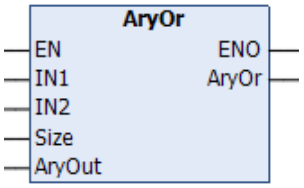
指令类别	名称	FB/FC	功能
数组逻辑相关	AryAnd	FC	数组与
	AryOr	FC	数组或
	AryXor	FC	数组异或
	AryXorN	FC	数组同或
	AryMax	FC	数组最大值检索
	AryMin	FC	数组最小值检索
	AryShl	FC	数组左移
	AryHR	FC	数组右移

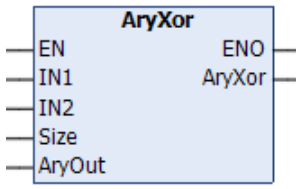
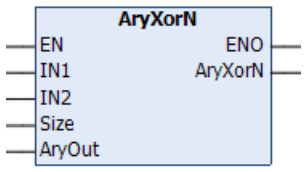
	ArySearch	FC	数组检索指定元素
	AryAddV	FC	数组加法
	ArySubV	FC	数组减法
	AryMean	FC	数组的平均值
	ArySD	FC	数组的标准差
	AryClr	FC	数组清零
	AryCpy	FC	数组复制
	RecMax	FC	结构体数组最大值检索
	RecMin	FC	结构体数组最小值检索
	RecSearch	FB	结构体数组检索指定元素
	RecRangeSearch	FC	结构体数组检索指定范围元素
	RecSort	FC	结构体数组元素排序
	RecNum	FC	结构体数组元素检索位置

3.3.1 AryAnd/AryOr/AryXor/AryXorN——数组与/或/异或/同或

对数组间各元素的布尔、位串的每一位进行运算

①指令格式

指令	功能	FB/FC	LD 表现	ST表现
AryAnd	数组与	FC		AryAnd(AryAnd=> , IN1:= , IN2:= , Size:= , AryOut:=);
AryOr	数组或	FC		AryOr(AryOr=> , IN1:= , IN2:= , Size:= , AryOut:=);

AryXor	数组异或	FC		AryXor(AryXor=> , IN1:= , IN2:= , Size:= , AryOut:=);
AryXorN	数组同或	FC		AryXorN(AryXorN=> , IN1:= , IN2:= , Size:= , AryOut:=);

②相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
IN1[]、IN2[]	运算对象数组	-	遵照数据类型	-	运算对象数组
Size	元素数	UINT	遵照数据类型	1	运算对象元素数

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
AryOut[]	元素数	UINT	遵照数据类型	1	运算对象元素数

数据类型

	布尔	位串				整数								实数		时刻、持续时间、日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
IN1[]	√	√	√	√	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
IN2[]	√	√	√	√	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Size	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
AryOut[]	√	√	√	√	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

运算的输入输出关系

AryAnd

两者均为 TRUE 时，运算结果为 TRUE。除此之外，运算结果为 FALSE。

IN1[] 的元素位	IN2[] 的元素位	AryOut[]的位
FALSE	FALSE	FALSE
FALSE	TRUE	FALSE
TRUE	FALSE	FALSE
TRUE	TRUE	TRUE

AryOr

有 TRUE 时，运算结果为 TRUE。除此之外，运算结果为 FALSE。

IN1[] 的元素位	IN2[] 的元素位	AryOut[]的位
FALSE	FALSE	FALSE
FALSE	TRUE	TRUE
TRUE	FALSE	TRUE
TRUE	TRUE	TRUE

AryXor

两者的值相同时，运算结果为 FALSE。不同时，运算结果为 TRUE。

IN1[] 的元素位	IN2[] 的元素位	AryOut[]的位
FALSE	FALSE	FALSE
FALSE	TRUE	TRUE
TRUE	FALSE	TRUE
TRUE	TRUE	FALSE

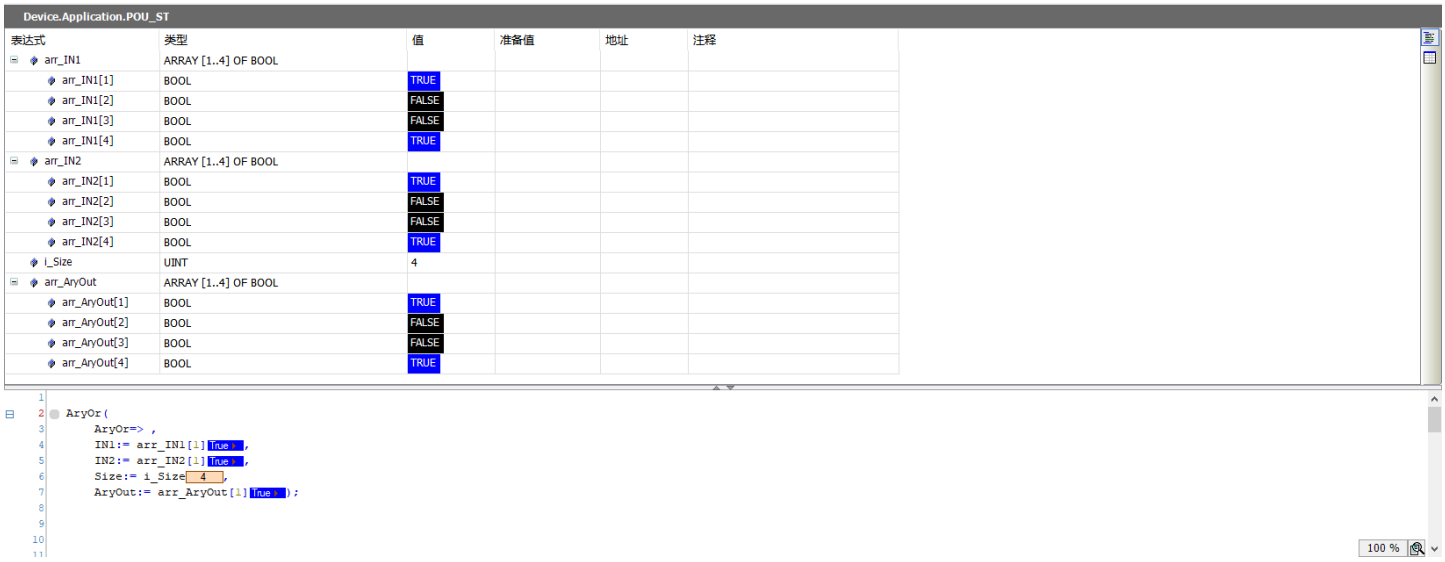
AryXorN

两者的值相同时，运算结果为 TRUE。不同时，运算结果为 FALSE。

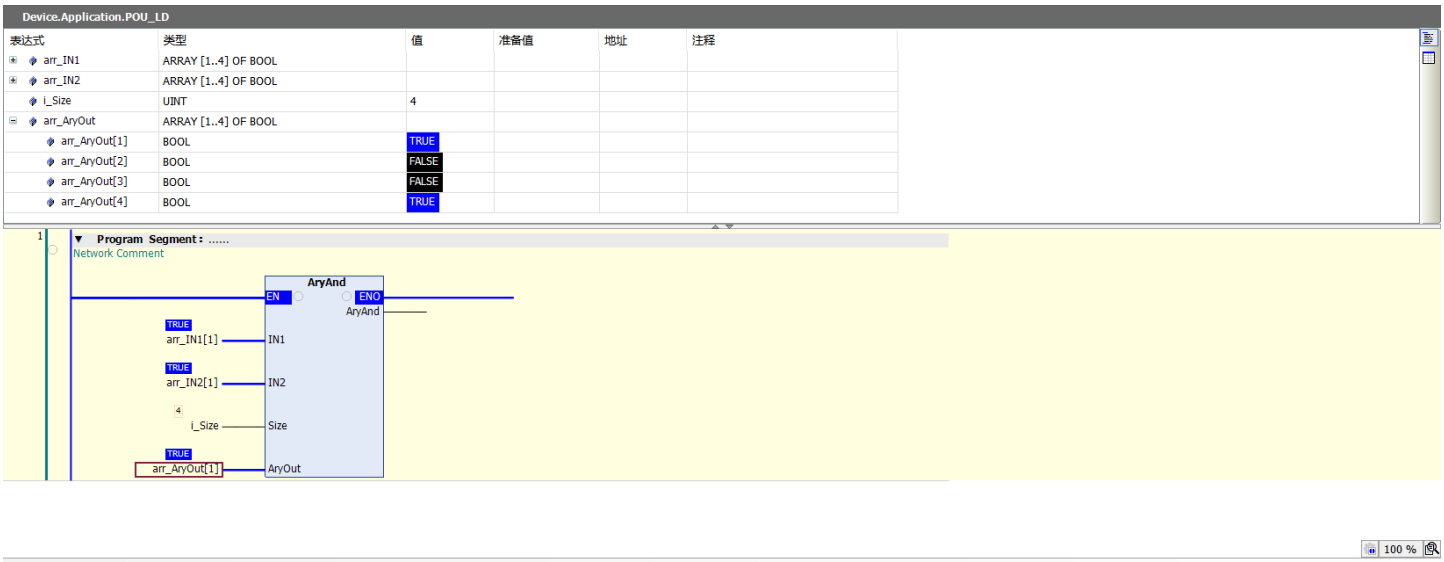
IN1[] 的元素位	IN2[] 的元素位	AryOut[]的位
FALSE	FALSE	TRUE
FALSE	TRUE	FALSE
TRUE	FALSE	FALSE
TRUE	TRUE	TRUE

③程序示例

ST:AryOr 指令下,“Size”=UINT#4 时示例。



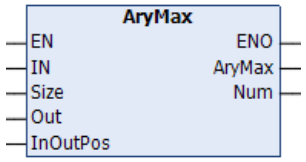
LD:AryOr 指令下,“Size”=UINT#4 时示例。



※注意事项
•In1[]、In2[]、AryOut[] 的数据类型应全部设为相同。若不同，编译时将会发生异常。
•“Size”的值超过 In1[]、In2[]、AryOut[] 中任一数组区域时，函数返回值为 FALSE，“AryOut” 的值不变。
•“Size”的值为 0 时，AryOut[] 的值无变化，函数返回值为 TRUE

3.3.2 AryMax/AryMin——数组最大值/最小值检索

检索 1 维数组最大 / 最小值
①指令格式

指令	功能	FB/FC	LD 表现	ST表现
AryMax	数组最大值检索	FC		AryMax(AryMax=> , IN:= , Size:= , Out:= , InOutPos:= , Num=>);
AryMin	数组最小值检索	FC		AryMin(AryMin=> , IN:= , Size:= , Out:= , InOutPos:= , Num=>);

②相关变量
输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
IN[]	检索对象数组	-	遵照数据类型	-	运算对象数组
Size	检索对象的元素数	UINT	遵照数据类型	1	检索对象的元素数
Out	检索结果	-	遵照数据类型		检索结果

输入输出变量

输入输出变量	名称	数据类型	有效范围	初始值	描述

InOutPos	检索元素	UINT	遵照数据类型	-	检索元素位置

输出变量

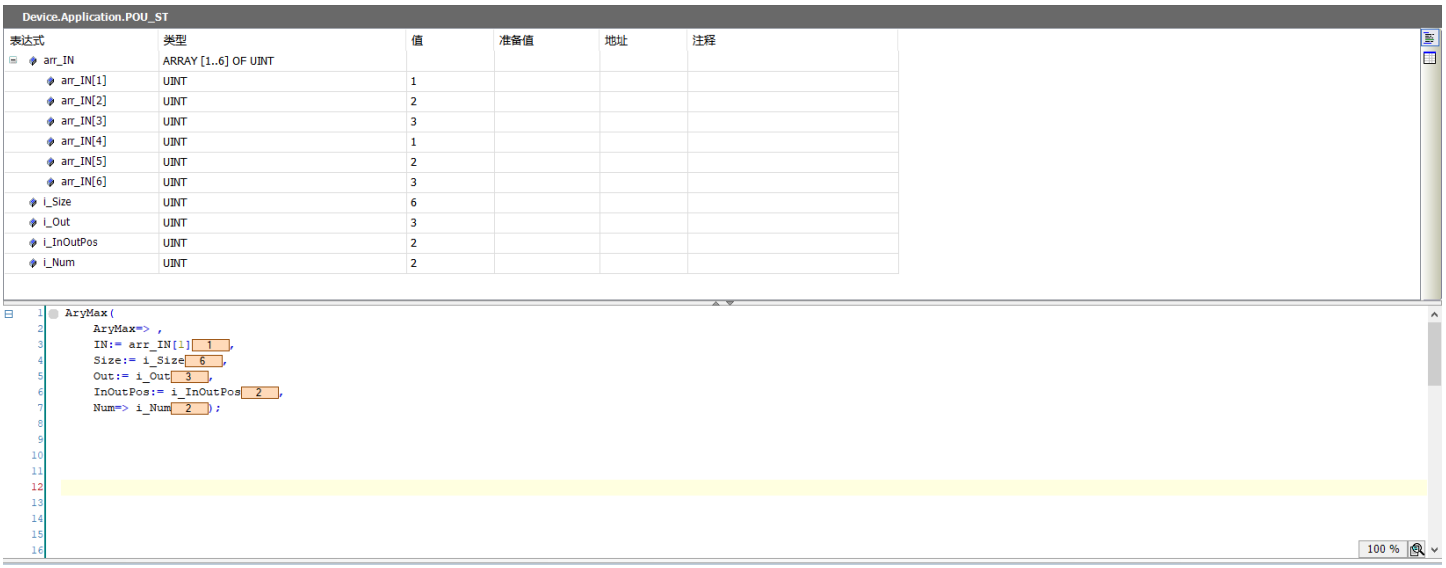
输出变量	名称	数据类型	有效范围	初始值	描述
Num	检索值的数量	UINT	遵照数据类型	-	检索值的数量

数据类型

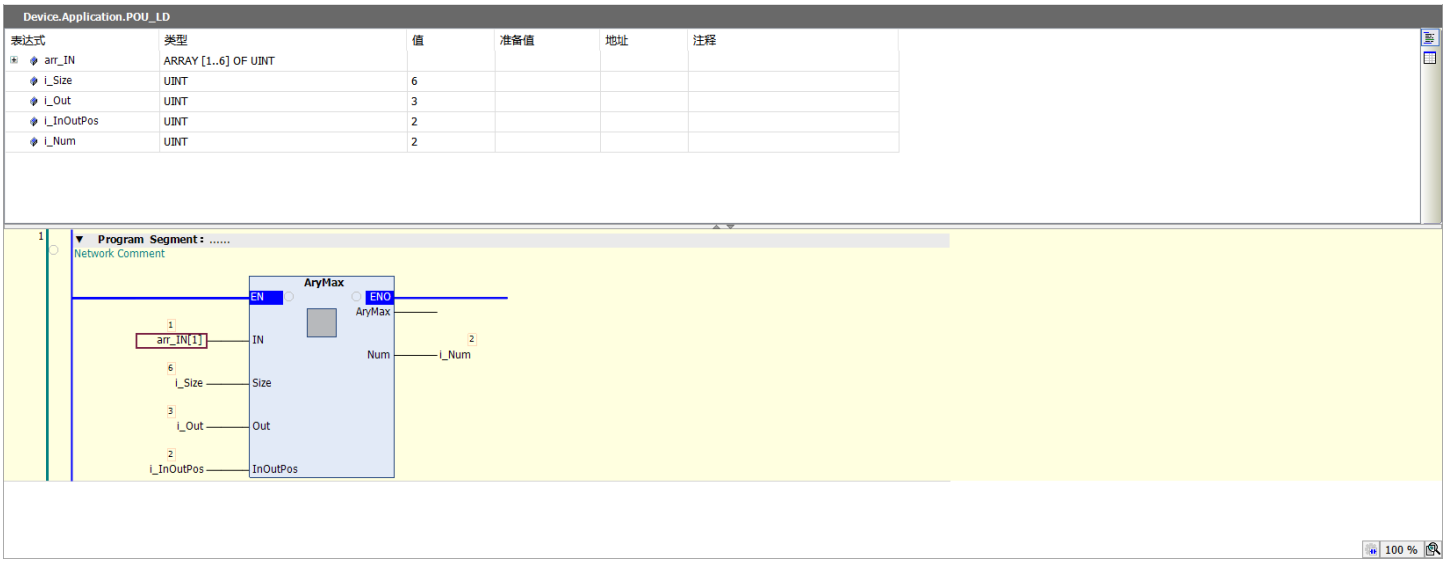
	布尔	位串				整数								实数		时刻、持续时间、日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
IN	-	-	-	-	-	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
Size	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
Out	-	-	-	-	-	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
InOutPos	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
Num	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-

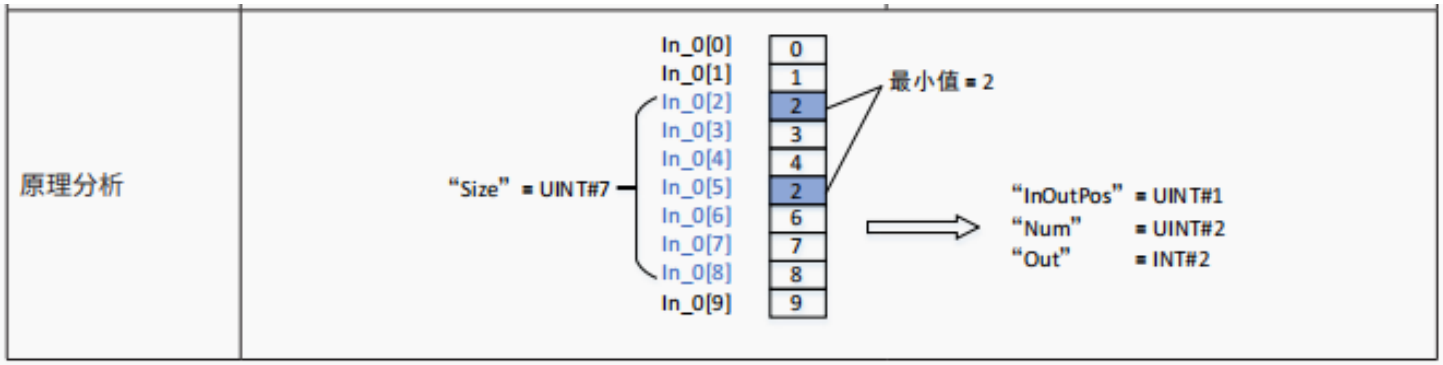
③程序示例

ST:AryMax 指令下,“Size”=UINT#6 时示例。



LD:AryMax 指令下，“Size”=UINT#6 时示例。





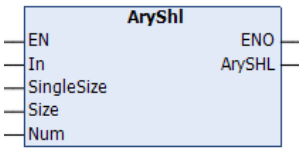
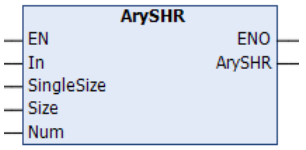
※注意事项

- “Size” 的值超过 In[ ] 的数组区域时，函数返回值的值为 FALSE，Out 不变。
- In[ ] 非 1 维数组时，编译时将会发生异常。
- 请确保“Out” 与检索对象数组 In[ ] 的元素为相同数据类型，否则编译时将会发生异常。
- 检索元素编号“InOutPos”为检索对象数组 In 中的元素编号。
- In[ ] 为实数时，因误差的影响，可能无法获取期望的结果。
- In[ ] 为不支持的数据类型时，编译时将会发生异常。

3.3.3 ArryShl/ArySHR——数组左移/右移

对多个数组元素进行移位。

①指令格式

指令	功能	FB/FC	LD 表现	ST表现
ArryShl	数组左移	FC		ArryShl(ArySHL=> , In:= , SingleSize:= , Size:= , Num:=);
ArySHR	数组右移	FC		ArySHR(ArySHR=> , In:= , SingleSize:= , Size:= , Num:=);

②相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
In[]	检索对象数组	-	遵照数据类型	-	运算对象数组
SingSize	单个元素字节长度	UINT	遵照数据类型	-	单个元素字节长度
Size	元素数	UINT	遵照数据类型	-	元素数
Num	移位位数	UINT	遵照数据类型	-	检索对象的元素数

数据类型

	布尔	位串				整数								实数		时刻、持续时间、日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In[]	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
SingSize	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
Size	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
Num	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-

功能说明

针对移位对象数组 InOut[ ] 的低位 / 高位 “Size” 个的元素，进行“Num”位的左 / 右移位。

移位后清除移位对象数组的溢出元素。

移位后将对象数组 InOut[ ] 的空缺位元素初始化，即把相应数据类型默认的初始值赋值给空缺位元素。

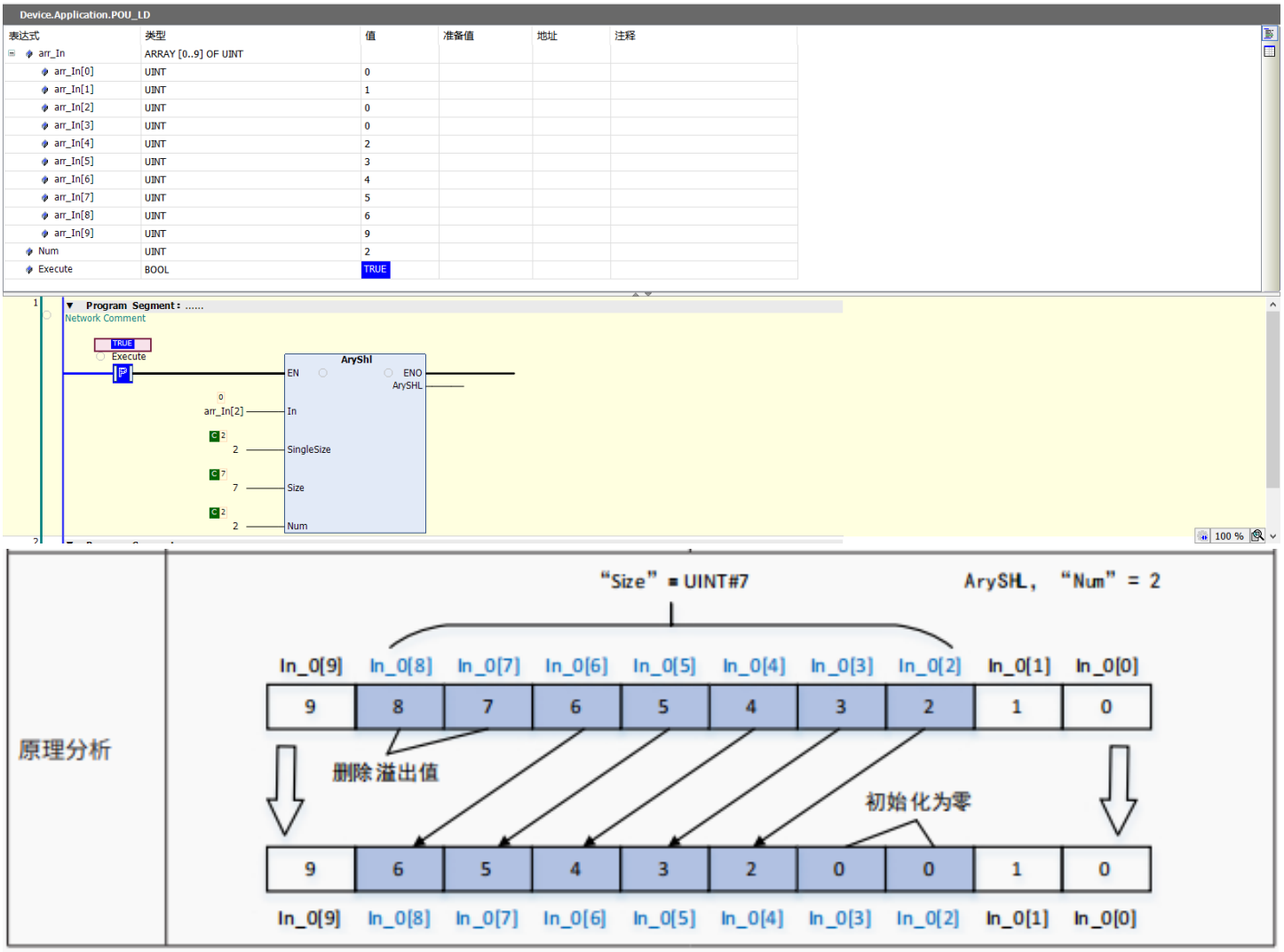
各数据类型默认的初始值如下表所示。

数据类型	描述
BOOL	FALSE
BYTE、WORD、DWORD、LWORD	16#0
USINT、UINT、UDINT、ULINT、SINT、INT、DINT、LINT、REAL、LREA	0
TIME	T#0ms
DATE	D#1970-1-1
TOD	TOD#0:0:0
DT	DT#1970-1-1-0:0:0
STRING	‘’

③程序示例

ST:

LD:



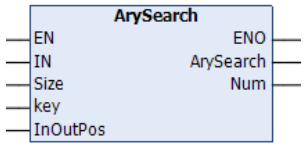
※注意事项

- 请确保 AryShl/ArySHR只执行1个扫描周期，比如：上升沿、下降沿作为执行条件，否则运行时会得不到正确的结果。

3.3.4 ArySearch——数组检索指定元素

在 1 维数组中检索指定值。

①指令格式

指令	功能	FB/FC	LD 表现	ST表现
ArySearch	数组检索指定元素	FC		ArySearch(ARySearch=> , IN:= , Size:= , key:= , InOutPos:= , Num=>);

②相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
IN[]	检索对象数组	-	遵照数据类型	-	运算对象数组
Size	检索对象的元素数	UINT	遵照数据类型	1	检索对象的元素数
Key	检索关键词	-	遵照数据类型		检索值

输入输出变量

输入输出变量	名称	数据类型	有效范围	初始值	描述
InOutPos	检索元素	UINT	遵照数据类型	-	检索元素位置

输出变量

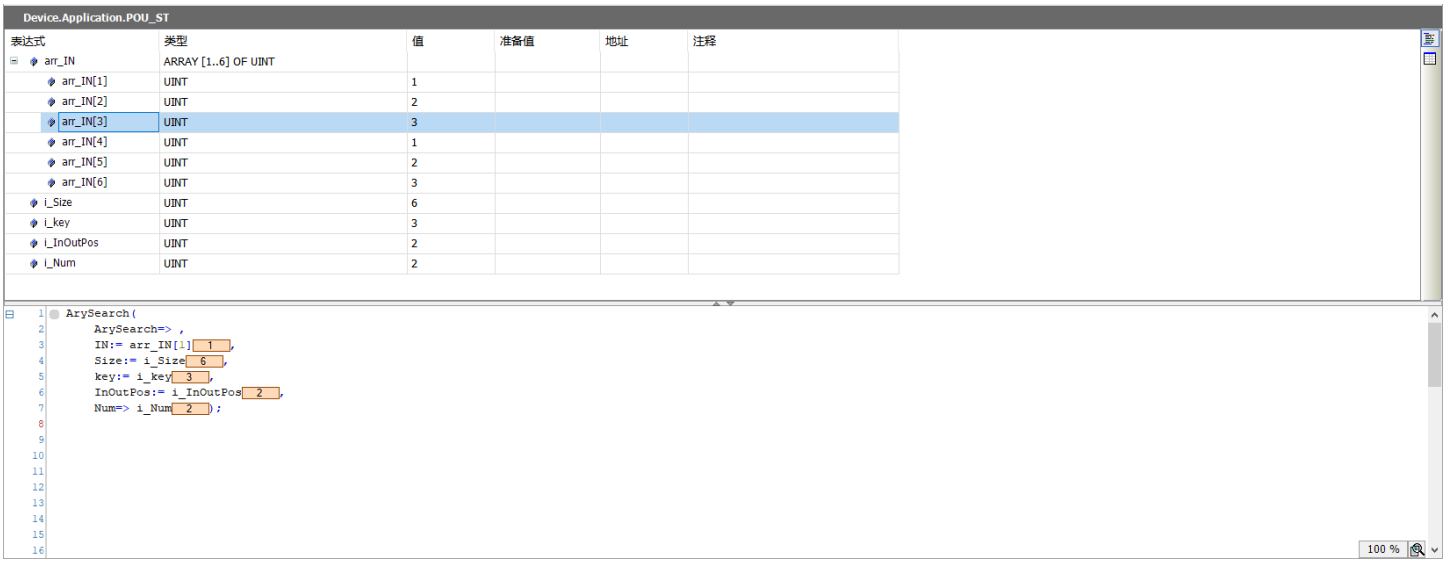
输出变量	名称	数据类型	有效范围	初始值	描述
Num	检索值的数量	UINT	遵照数据类型	-	检索值的数量

数据类型

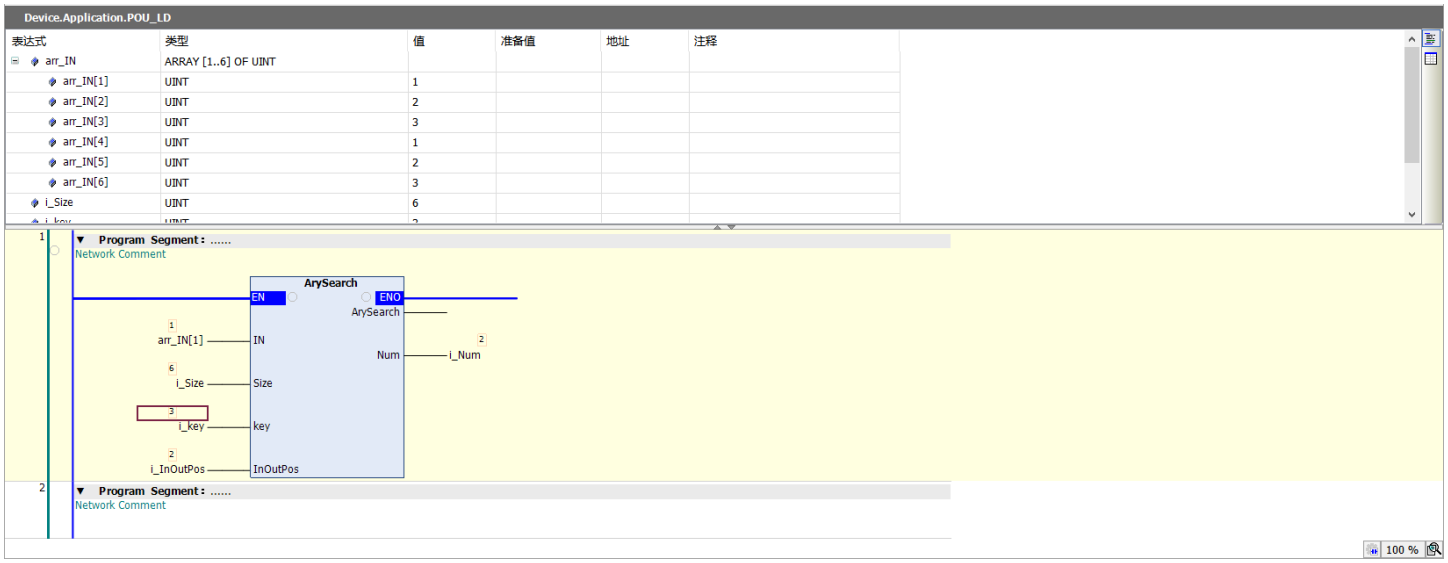
	布尔	位串				整数								实数		时刻、持续时间、日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
IN	-	-	-	-	-	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
Size	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
Key	-	-	-	-	-	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
InOutPos	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
Num	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-

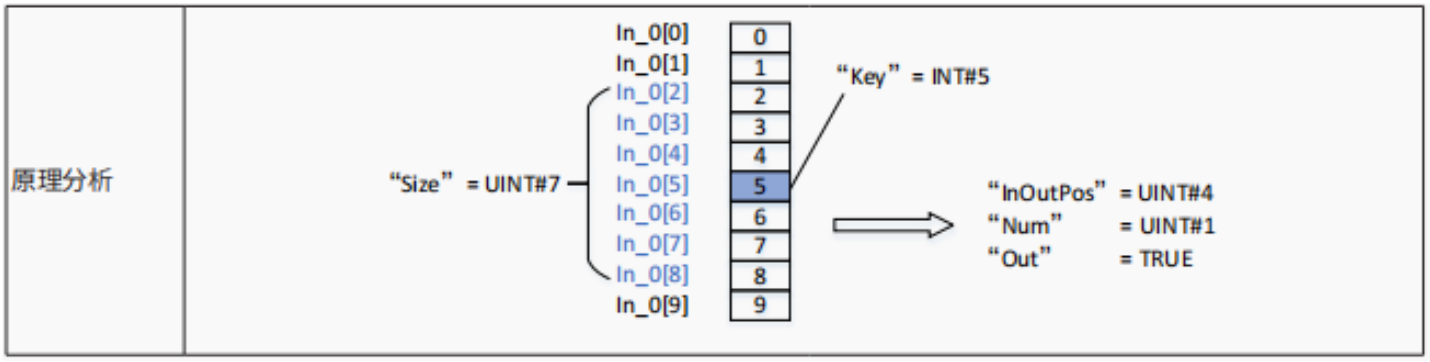
③程序示例

ST：



LD:





※注意事项

• “Size” 的值超过 AryOut[ ] 的数组区域时，函数返回值的值为 FALSE，“Num”的值初始化为 0。

•In[ ] 非 1 维数组时，编译时将会发生异常。

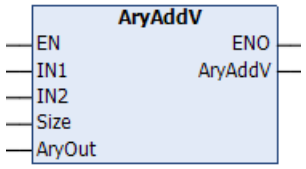
• 请确保“Key” 与检索对象数组 In[ ] 的元素为相同数据类型，否则编译时将会发生异常。

- 请确保“Key” 的输入参数为变量。
- 检索元素编号“InOutPos”为检索对象数组 In 中的元素编号。

3.3.5 AryAddV——数组加法

在数组的各要素上加相同数值。

①指令格式

指令	功能	FB/FC	LD 表现	ST表现
AryAddV	数组加法	FC		AryAddV(AryAddV=> , IN1:= , IN2:= , Size:= , AryOut:=);

②相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
IN1[]	对象数组	-	遵照数据类型	-	运算对象数组
IN2	相加值	-	遵照数据类型	-	相加值
Size	元素数	UINT	遵照数据类型	1	对象的元素数

输出变量

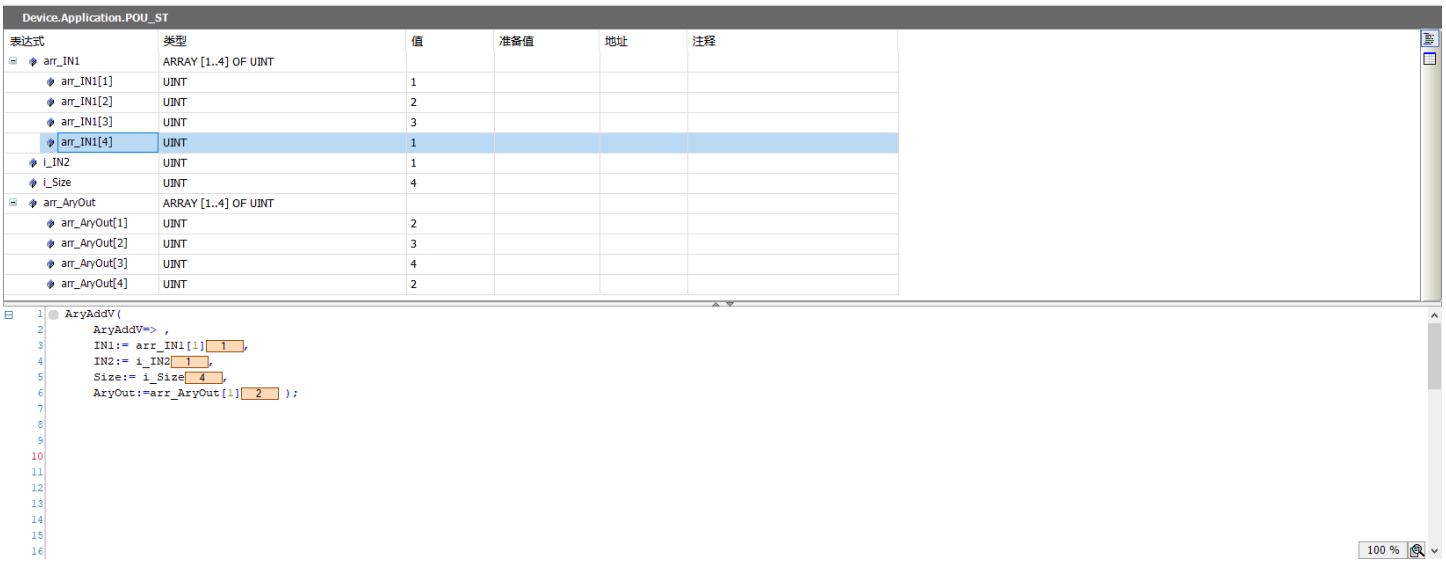
输出变量	名称	数据类型	有效范围	初始值	描述
AryOut[]	结果数组	-	遵照数据类型	-	运算结果数组数组

数据类型

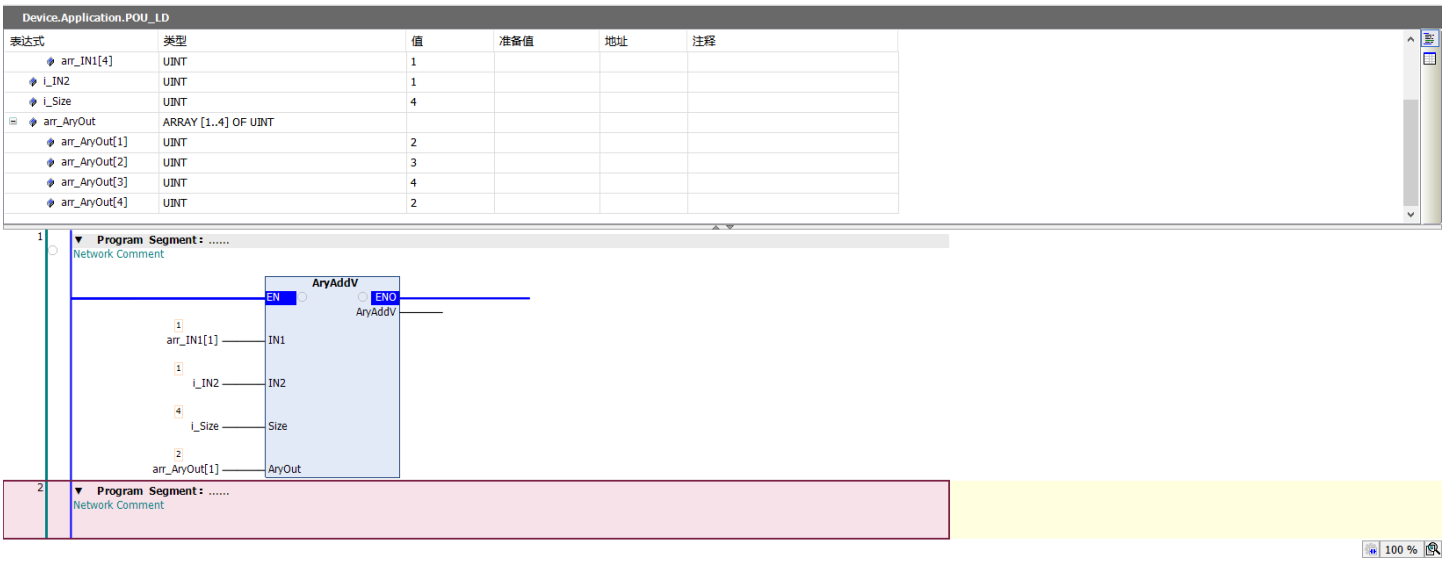
	布尔	位串				整数								实数		时刻、持续时间、日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
IN1[]	-	-	-	-	-	√	√	√	√	√	√	√	√	√	√	-	-	-	-	-
IN2	-	-	-	-	-	√	√	√	√	√	√	√	√	√	√	-	-	-	-	-
Size	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
AryOut[]	-	-	-	-	-	√	√	√	√	√	√	√	√	√	√	-	-	-	-	-

③程序示例

ST:



LD:



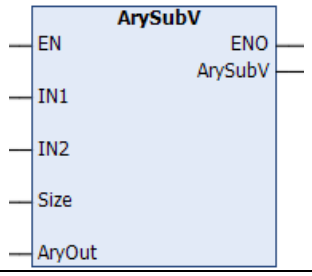
※注意事项

- “Size”的值超出 In1[] 或 AryOut[] 的数组区域时，函数返回值的值为 FALSE，AryOut[] 不变。
- “In[]”、“In2”、“AryOut”的数据类型应统一。若不相同，编译时将会发生异常。
- 若相加结果超出 AryOut[] 的有效范围，AryOut[] 的要素为非法值。此时，不会发生异常。此外，也不会破坏该要素的相邻存储区域。
- “Size”的值为 0 时，AryOut[] 的值无变化，函数返回值为 TRUE。

3.3.6 ArySubV——数组减法

在数组的各要素上减相同数值。

①指令格式

指令	功能	FB/FC	LD 表现	ST表现
ArySubV	数组减法	FC		ArySubV(ArySubV=> , IN1:= , IN2:= , Size:= , AryOut:=);

②相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
IN1[]	对象数组	-	遵照数据类型	-	运算对象数组
IN2	相加值	-	遵照数据类型	-	相加值
Size	元素数	UINT	遵照数据类型	1	对象的元素数

输出变量

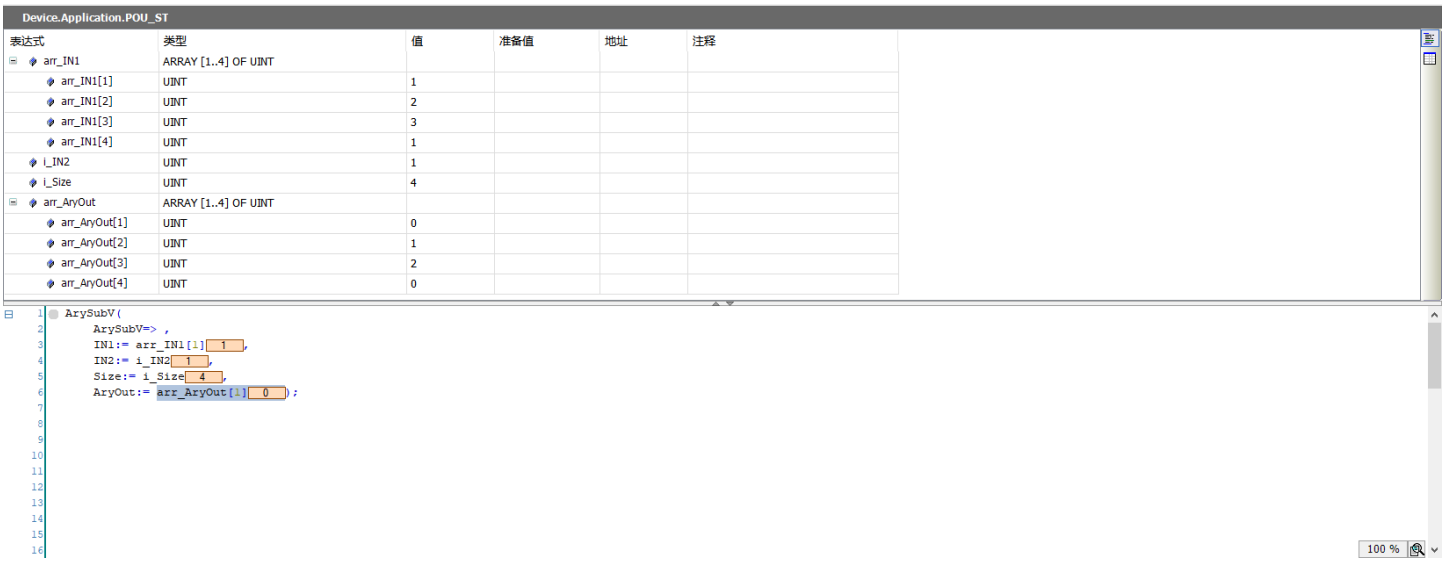
输出变量	名称	数据类型	有效范围	初始值	描述
AryOut[]	结果数组	-	遵照数据类型	-	运算结果数组数组

数据类型

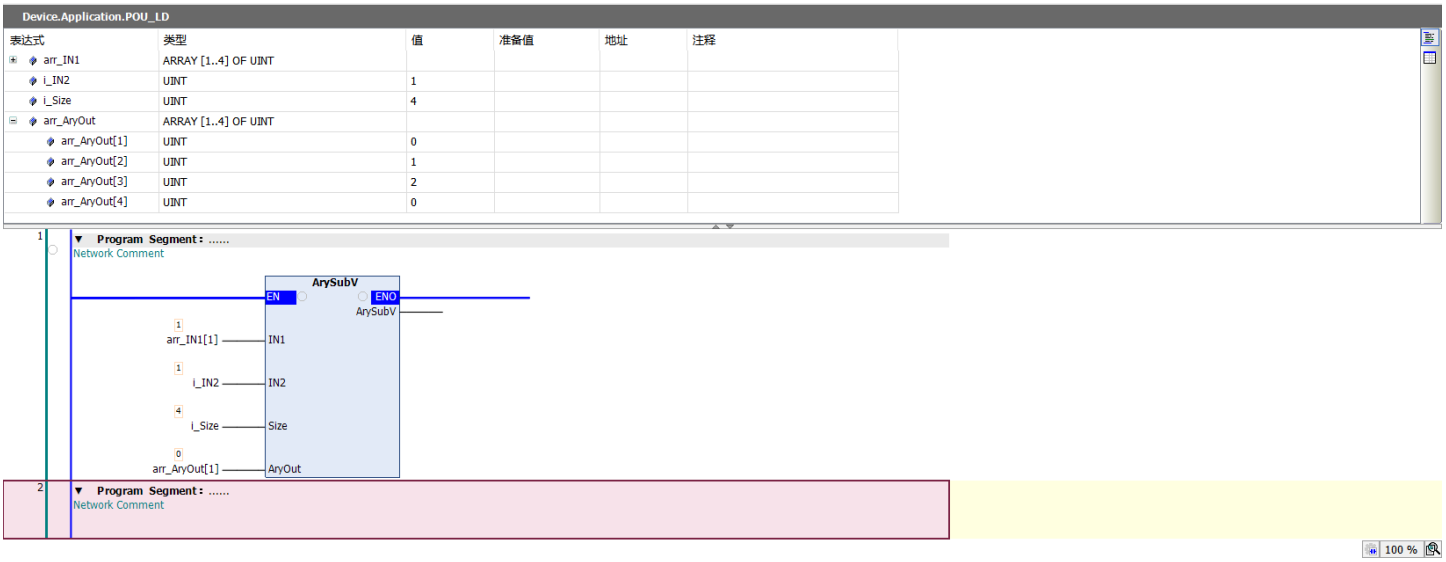
	布尔	位串				整数								实数		时刻、持续时间、日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
IN1[]	-	-	-	-	-	√	√	√	√	√	√	√	√	√	√	-	-	-	-	-
IN2	-	-	-	-	-	√	√	√	√	√	√	√	√	√	√	-	-	-	-	-
Size	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
AryOut[]	-	-	-	-	-	√	√	√	√	√	√	√	√	√	√	-	-	-	-	-

③程序示例

ST：



LD:



※注意事项

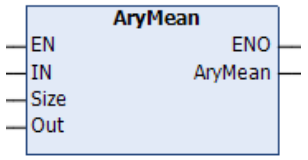
- “Size”的值超出 In1[] 或 AryOut[] 的数组区域时，函数返回值的值为 FALSE，AryOut[] 不变。
- “In[]”、“In2”、“AryOut”的数据类型应统一。若不相同，编译时将会发生异常。
- 若相加结果超出 AryOut[] 的有效范围，AryOut[] 的要素为非法值。此时，不会发生异常。此外，也不会破坏该要素的相邻存储区域。
- “Size”的值为 0 时，AryOut[] 的值无变化，函数返回值为 TRUE。

3.3.7 AryMean——数组的平均值

计算数组元素的平均值。

①指令格式

指令	功能	FB/FC	LD 表现	ST表现

AryMean	数组的平均值	FC		AryMean(AryMean=> , IN:= , Size:= , Out:=);

②相关变量

输入变量

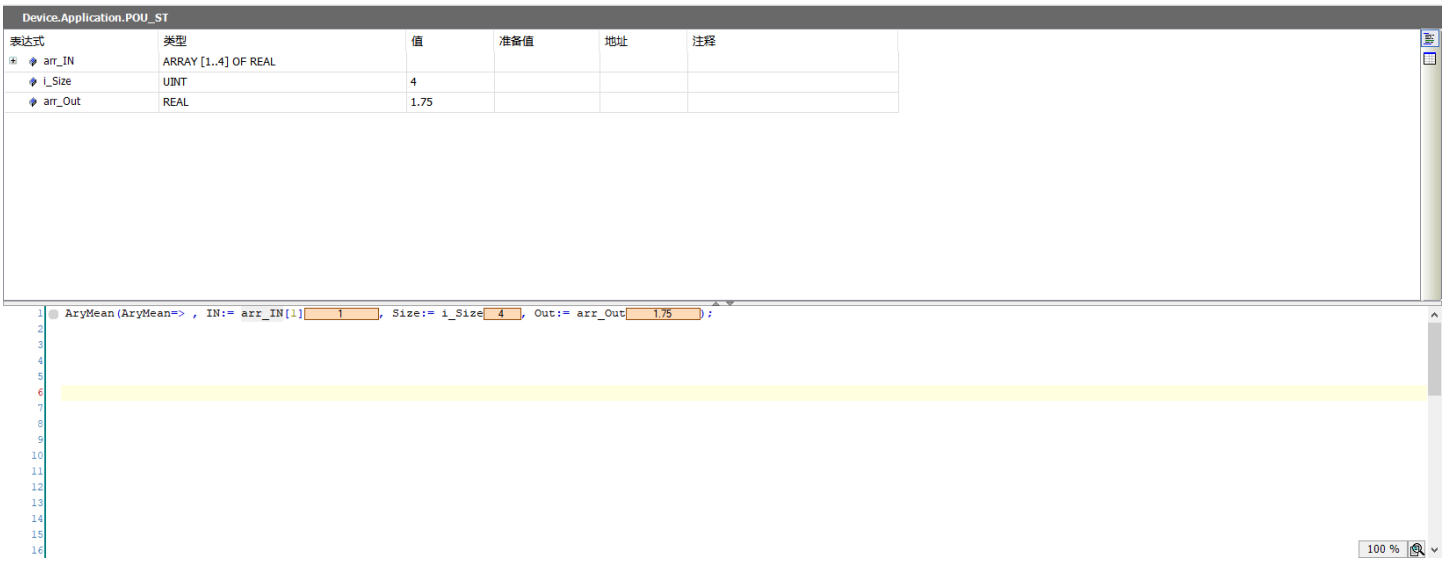
输入变量	名称	数据类型	有效范围	初始值	描述
IN1[]	对象数组	-	遵照数据类型	-	运算对象数组
Size	元素数	-	遵照数据类型	1	对象的元素数
Out	运算结果	-	遵照数据类型		运算结果

数据类型

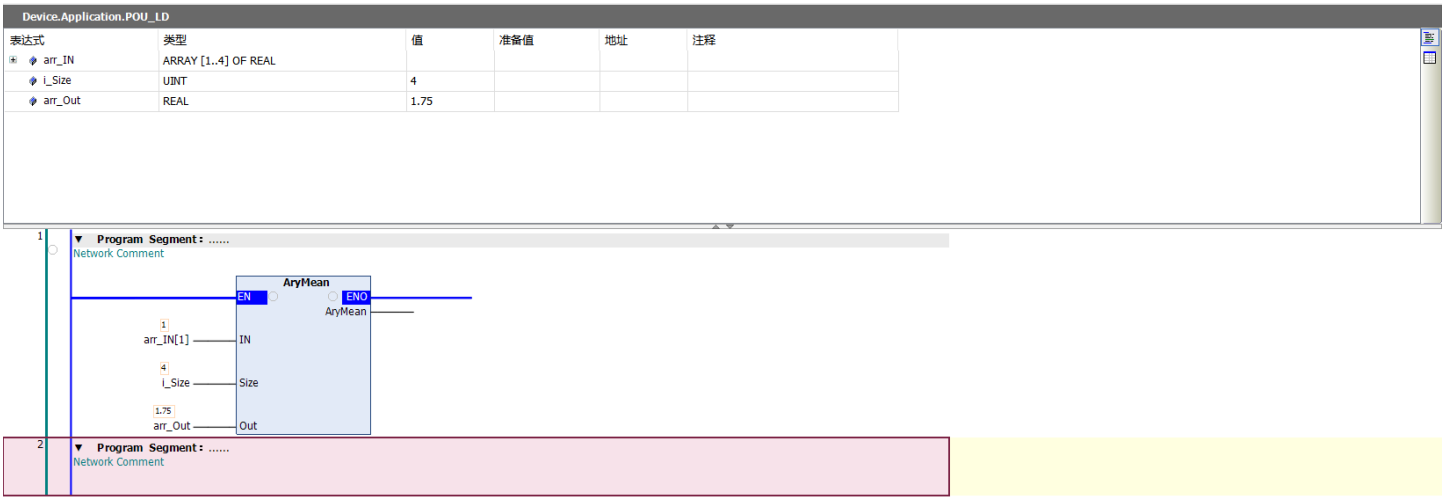
	布尔	位串				整数								实数		时刻、持续时间、日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
IN1[]	-	-	-	-	-	√	√	√	√	√	√	√	√	√	√	-	-	-	-	-
Size	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
Out	-	-	-	-	-	√	√	√	√	√	√	√	√	√	√	-	-	-	-	-

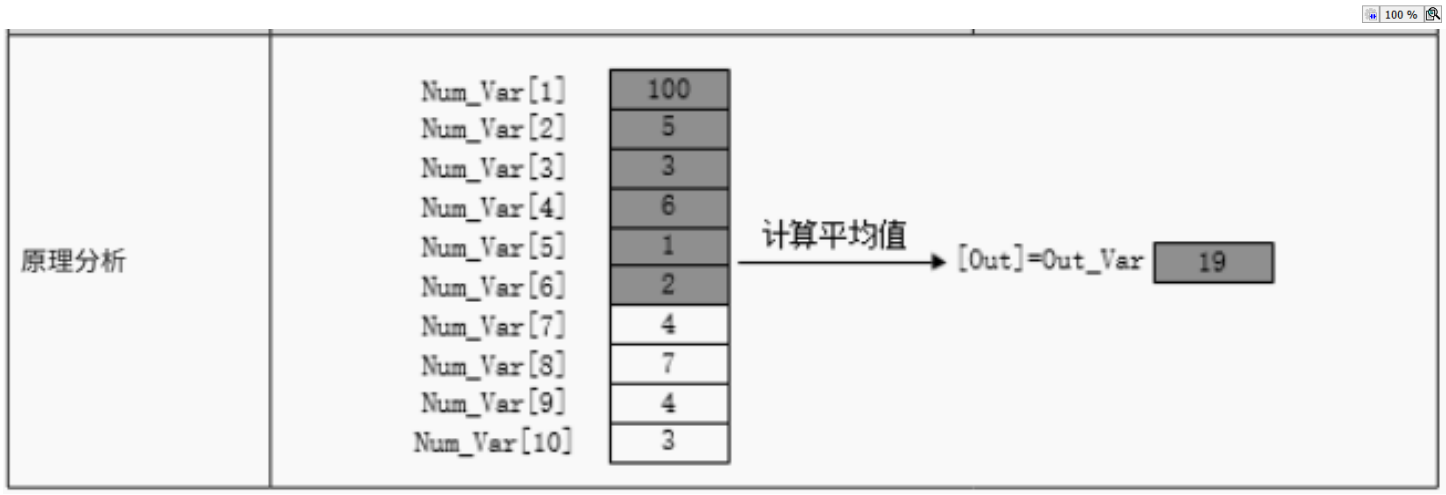
③程序示例

ST：



LD:





※注意事项

- 不支持 In[ ] 的值为 + ∞ /- ∞和非数时的四则运算。
- In[ ] 和“Out” 的数据类型不同时，编译时将会发生异常。
- “Size” 的值超过 In[ ] 的数组区域时，函数返回值为 FALSE，“Out”不变。

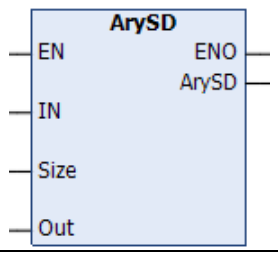
•In[ ]、“Out” 为整数型时，舍去平均值的小数点后的数字。

•“Size” 的值为 0 时，“Out” 的值为 0，函数返回值为 TRUE。

3.3.8 ArySD——数组的标准差

计算数组元素的标准差。

①指令格式

指令	功能	FB/FC	LD 表现	ST表现
ArySD	数组的标准差	FC		ArySD(ArySD=> , IN:= , Size:= , Out:=);

②相关变量

输入变量

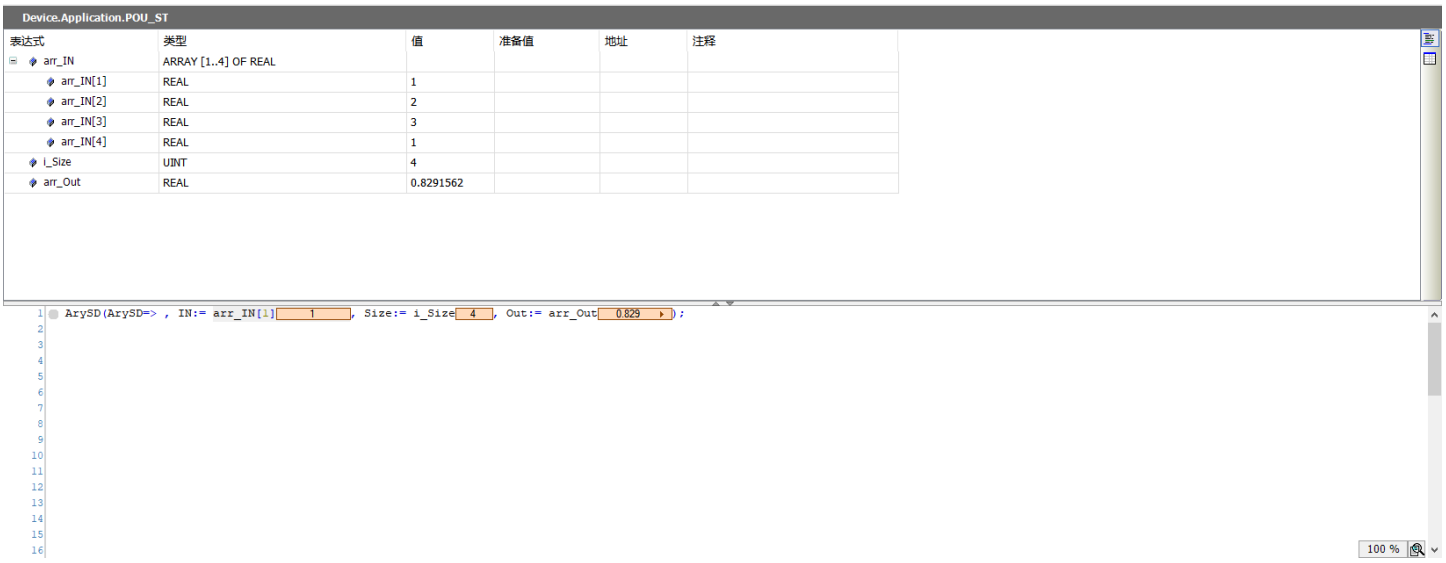
输入变量	名称	数据类型	有效范围	初始值	描述
IN1[]	对象数组	-	遵照数据类型	-	运算对象数组
Size	元素数	-	遵照数据类型	1	对象的元素数
Out	运算结果	-	遵照数据类型		运算结果

数据类型

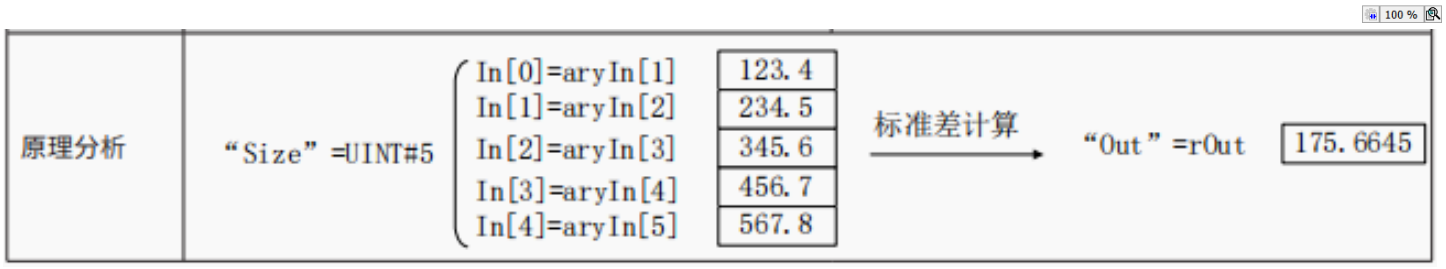
	布尔	位串				整数								实数		时刻、持续时间、日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
IN1[]	-	-	-	-	-	√	√	√	√	√	√	√	√	√	√	-	-	-	-	-
Size	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
Out	-	-	-	-	-	√	√	√	√	√	√	√	√	√	√	-	-	-	-	-

③程序示例

ST:



LD:



※注意事项

•In[ ] 和“Out” 的数据类型不同时，编译时将会发生异常。

•“Size” 的值超过 In[ ] 的数组区域时，函数返回值为 FALSE，“Out”不变。

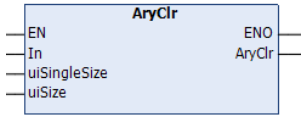
•In[ ]、“Out” 为整数型时，舍去平均值的小数点后的数字。

•“Size” 的值为 0 时，“Out” 的值为 0，函数返回值为 TRUE。

3.3.9 AryClr——数组清零

数组元素清零。

①指令格式

指令	功能	FB/FC	LD 表现	ST表现
AryClr	数组清零	FC		AryClr(AryClr=> , In:= , uiSingleSize:= , uiSize:=);

②相关变量

输入变量

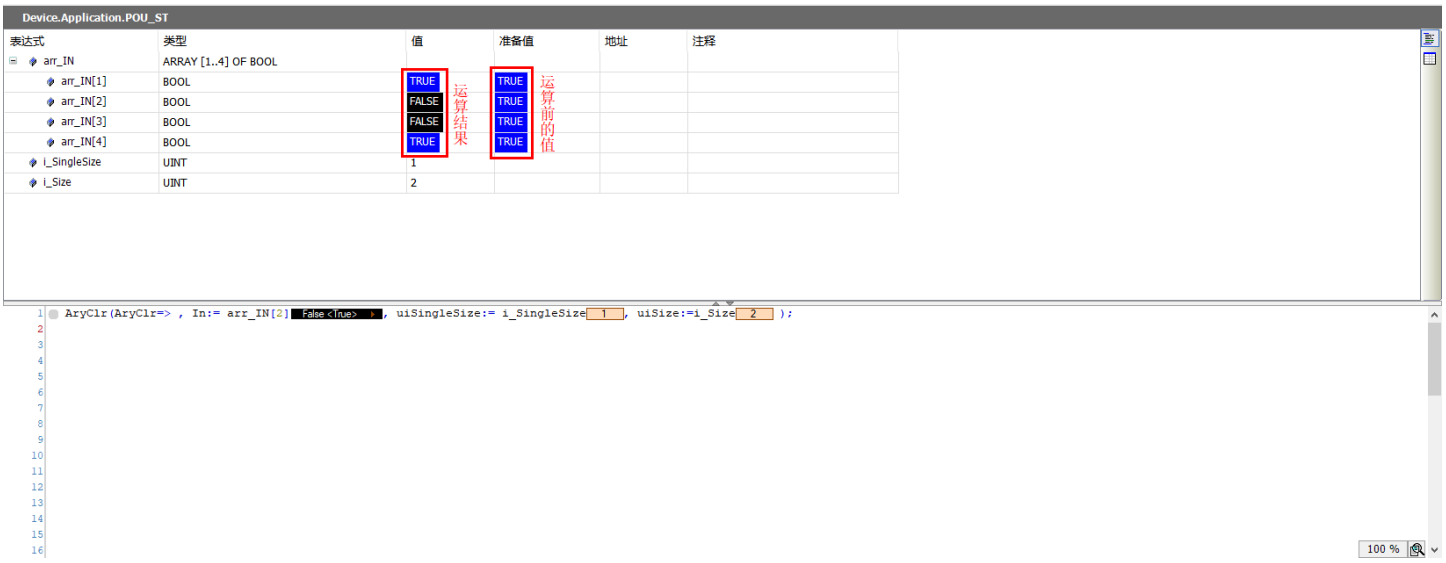
输入变量	名称	数据类型	有效范围	初始值	描述
IN[]	对象数组	-	遵照数据类型	-	运算对象数组
uiSingleSize	单个元素字节长度	-	遵照数据类型	1	单个元素字节长度
uiSize	元素数	-	遵照数据类型		元素数

数据类型

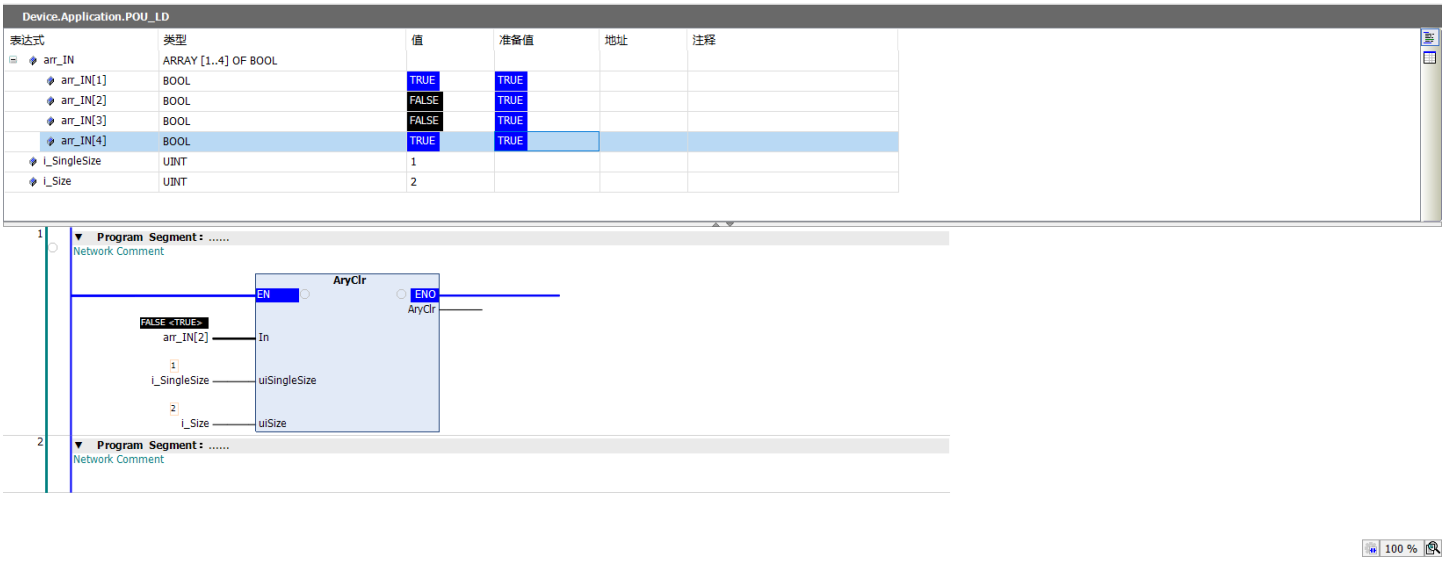
	布尔	位串				整数								实数		时刻、持续时间、日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
IN[]	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
uiSingleSize	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
uiSize	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-

③程序示例

ST：



LD:



3.3.10 AryCpy——数组复制

数组元素复制。

①指令格式

指令	功能	FB/FC	LD 表现	ST表现
AryCpy	数组复制	FC	AryCpy EN Source Des uiSingleSize uiSize ENO AryCpy	AryCpy(AryCpy=> , Source:= , Des:= , uiSingleSize:= , uiSize:=);

②相关变量

输入变量

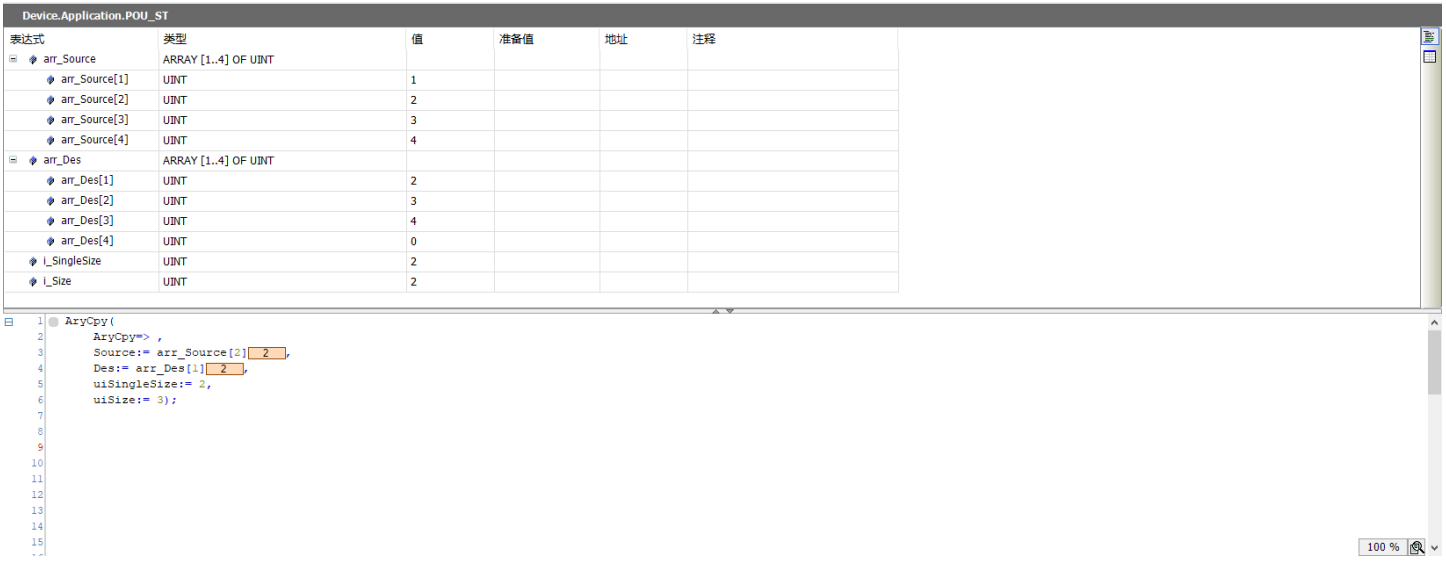
输入变量	名称	数据类型	有效范围	初始值	描述
Source[]	对象数组	-	遵照数据类型	-	运算对象数组
Des[]	结果数组	-	遵照数据类型	-	运算对象数组
uiSingleSize	单个元素字节长度	-	遵照数据类型	1	单个元素字节长度
uiSize	元素数	-	遵照数据类型		元素数

数据类型

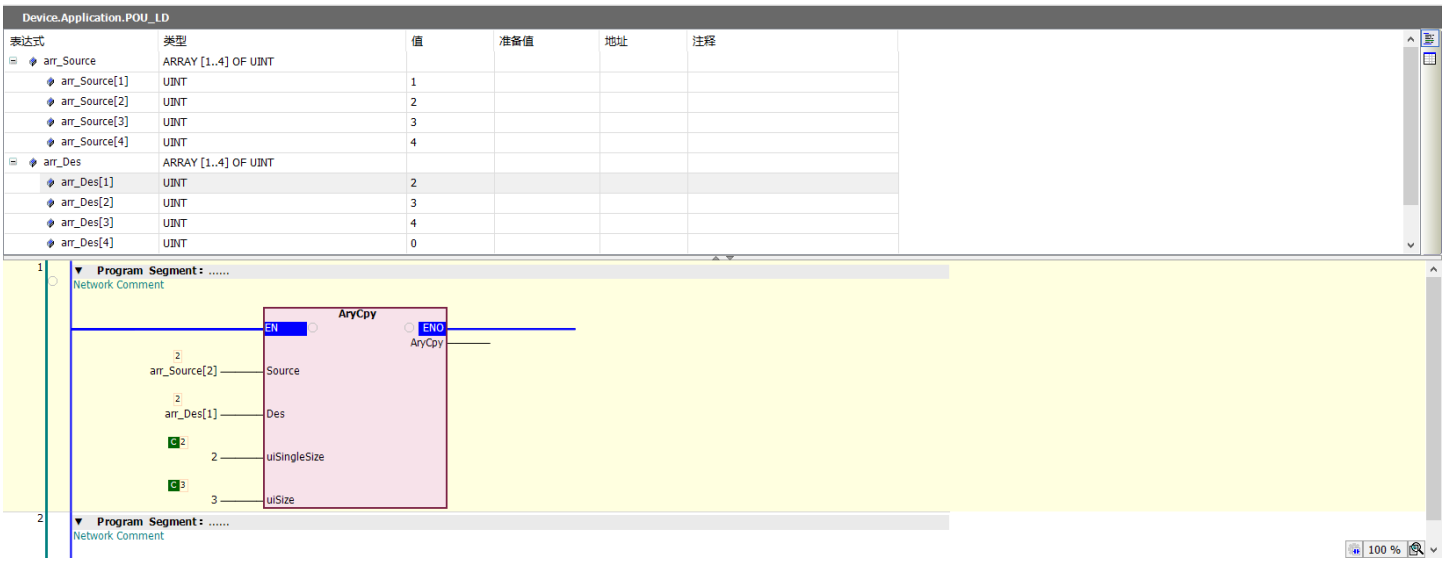
	布尔	位串				整数								实数		时刻、持续时间、日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Source[]	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
Des[]	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
uiSingleSize	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
uiSize	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-

③程序示例

ST：



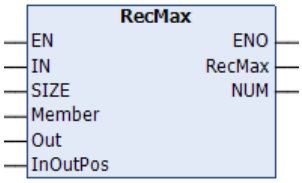
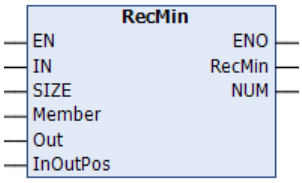
LD：



3.3.11 RecMax/RecMin——结构体数组最大值/最小值检索

检索结构体数组最大值/最小值检索

①指令格式

指令	功能	FB/FC	LD 表现	ST表现
RecMax	结构体数组最大值检索	FC		RecMax(RecMax=> , IN:= , SIZE:= , Member:= , Out:= , InOutPos:= , NUM=>);
RecMin	结构体数组最小值检索	FC		RecMin(RecMin=> , IN:= , SIZE:= , Member:= , Out:= , InOutPos:= , NUM=>);

②相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
IN[]	检索对象数组	-	遵照数据类型	-	运算对象数组
Member	元素成员	UINT		1	元素成员
SIZE	检索对象的元素数	UINT	遵照数据类型	1	检索对象的元素数
Out	检索结果	-	遵照数据类型		检索结果

输入输出变量

输入输出变量	名称	数据类型	有效范围	初始值	描述
InOutPos	元素位置	UINT	遵照数据类型	-	检索元素位置

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Num	检索值的数量	UINT	遵照数据类型	-	检索值的数量

数据类型

	布尔	位串				整数								实数		时刻、持续时间、日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
IN	-	-	-	-	-	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
Member	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
Size	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
Out	-	-	-	-	-	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
InOutPos	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
Num	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-

功能说明

从以结构体为要素的排列 In[] 的“Size”个元素中，即从 In[1] 到 In[Size] 中，检索结构体的检索对象成员“Member”的值。

将 In[] 中任一元素的检索对象成员作为参数传递到“Member”。

分别将检索结果的要素编号写入 InOutPos 中，检索要素的数量写入“Num”中。检索结果有 2 个以上时，将 In[] 中最低位的检索结果的要素编号带入 InOutPos 中。

③程序示例

ST:RecMax 指令下的示例。

Device.Application.POU_ST

表达式	类型	值	准备值	地址	注释
arr_IN	ARRAY [1..3] OF DUT_Stru				
arr_IN[1]	DUT_Stru				
arr	ARRAY [1..4] OF UINT				
arr[1]	UINT	10			
arr[2]	UINT	2			
arr[3]	UINT	3			
arr[4]	UINT	4			
arr_IN[2]	DUT_Stru				
arr_IN[3]	DUT_Stru				
arr	ARRAY [1..4] OF UINT				
arr[1]	UINT	10			
arr[2]	UINT	2			
arr[3]	UINT	3			
arr[4]	UINT	4			
i_Size	UINT	3			
i_Out	UINT	10			
i_NUM	UINT	3			

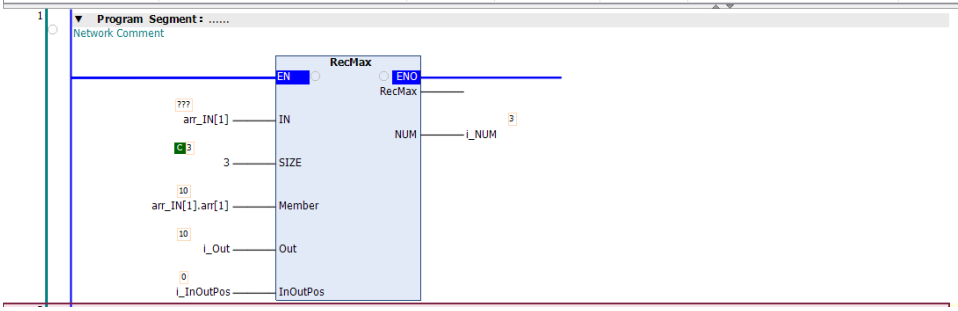
```
RecMax(
    RecMax=> ,
    IN:= arr_IN[1],
    SIZE:= 3,
    Member:= arr_IN[1].arr[1] 10 ,
    Out:= i_Out 10 ,
    InOutPos:= i_InOutPos 0 ,
    NUM=> i_NUM 3 );
```

100 %

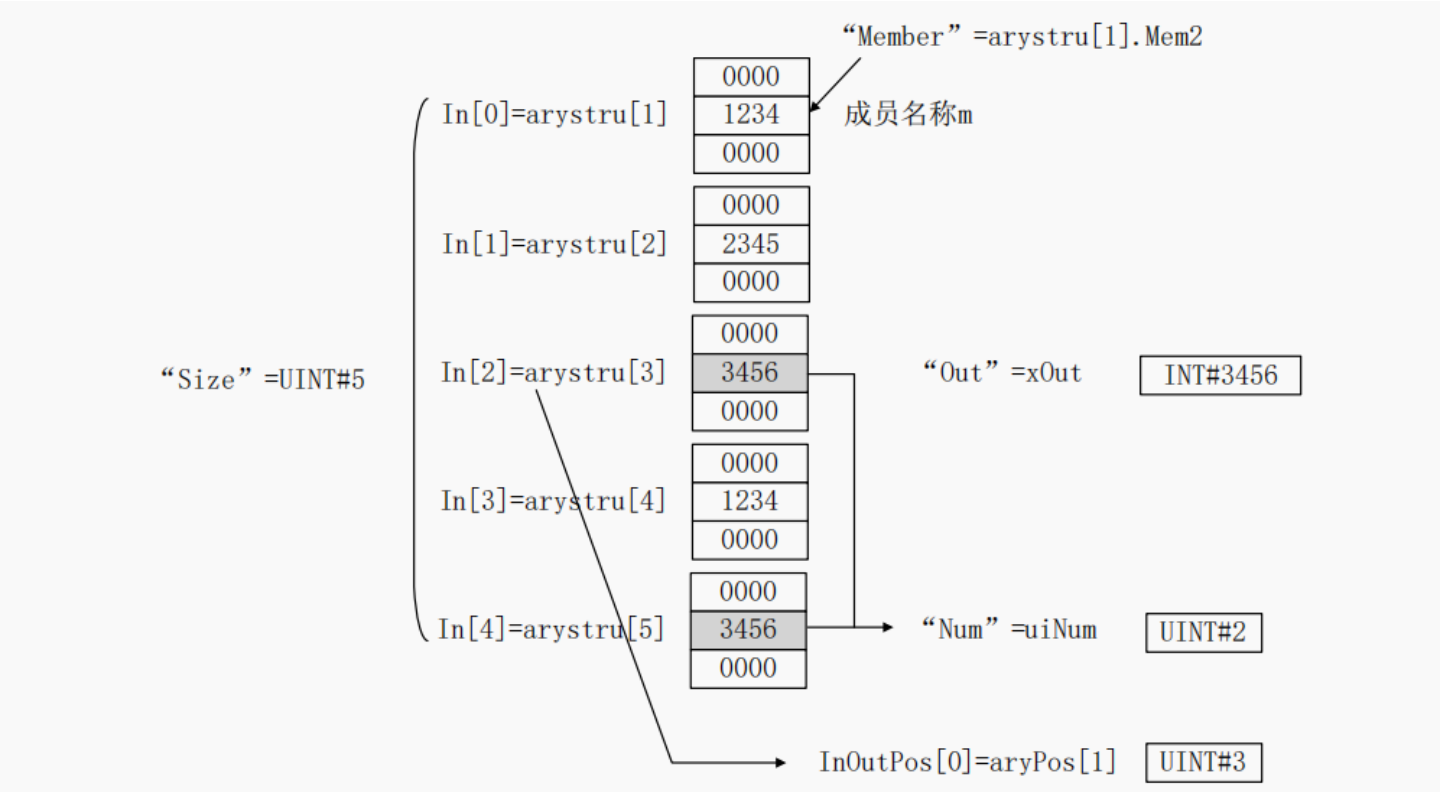
LD:RecMax 指令下的示例。

Device.Application.POU_LD

表达式	类型	值	准备值	地址	注释
arr_IN	ARRAY [1..3] OF DUT_Stru				
arr_IN[1]	DUT_Stru				
arr	ARRAY [1..4] OF UINT				
arr[1]	UINT	10			
arr[2]	UINT	2			
arr[3]	UINT	3			
arr[4]	UINT	4			
arr_IN[2]	DUT_Stru				
arr_IN[3]	DUT_Stru				
i_Size	UINT	3			



100 %



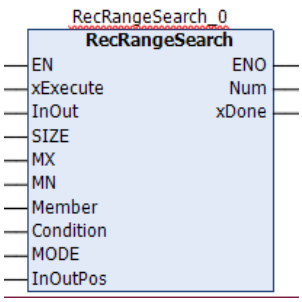
※注意事项

- “Size” 的值超过 In[ ] 的数组区域时，函数返回值的值为 FALSE，Out 不变。
- In[ ] 非 1 维数组时，编译时将会发生异常。
- 请确保“Out” 与检索对象数组 In[ ] 的元素为相同数据类型，否则编译时将会发生异常。
- 检索元素编号“InOutPos”为检索对象数组 In 中的元素编号。
- In[ ] 为实数时，因误差的影响，可能无法获取期望的结果。
- In[ ] 为不支持的数据类型时，编译时将会发生异常。

3.3.12 RecRangeSearch——结构体数组检索指定范围元素

从将结构体作为元素的排列中，以指定方法检索与检索条件范围匹配的元素。

①指令格式

指令	功能	FB/FC	LD 表现	ST表现
RecRangeSearch	结构体数组检索指定范围元素	FB		RecRangeSearch_0(xExecute:= , InOut:= , SIZE:= , MX:= , MN:= , Member:= , Condition:= , MODE:= , Num=> , xDone=> , InOutPos:=);

②相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
xExecute	上升沿执行	-	TRUE/FALSE	-	运算对象数组
InOut[]	检索对象数组	-	遵照数据类型	-	运算对象数组
Size	检索对象的元素数	UINT	遵照数据类型	-	检索对象的元素数
MX	检索上限	-	遵照数据类型	-	检索上限值
MN	检索下限	-	遵照数据类型	-	检索下限值
Member	元素成员	-	遵照数据类型	-	元素成员
MODE	模式	UINT	0,1,2	-	0：线性模式 1：升序模式 2：降序模式
Condition	范围条件	UINT	1，2,3,4	-	1：含上下限，闭区间 2：不含上限，左闭右开区间 3：不含下限，左开右闭区间 4：不含上下限，开区间

输入输出变量

输入输出变量	名称	数据类型	有效范围	初始值	描述
InOutPos	元素位置	UINT	遵照数据类型	-	检索元素位置

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Num	检索值的数量	UINT	遵照数据类型	-	检索值的数量
xDone	执行完成	BOOL	TRUE/FALSE	-	执行完成

数据类型

	布尔	位串				整数								实数		时刻、持续时间、日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
xExecute	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

InOut[]	-	-	-	-	-	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
Size	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
MX	-	-	-	-	-	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
MN	-	-	-	-	-	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
Member	-	-	-	-	-	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
MODE	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
Condition	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
InOutPos	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
Num	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
xDone	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

功能说明

从以结构体为要素的排列 In[] 的“Size”个元素中，即从 In[1] 到 In[Size] 中，检索结构体的检索对象成员“Member”的值。

将 In[] 中任一元素的检索对象成员作为参数传递到“Member”。

MODE=0 ，从当前元素顺序遍历整个数组，不依赖数据顺序，使用不需要排序。

MODE=1 ,升序二分查找时，在执行本指令前，In[] 的输入参数的排列元素需要事先升序排列。然后再执行本指令，到进行二分查找

MODE=2 ,降序二分查找时，在执行本指令前，In[] 的输入参数的排列元素需要事先降序排列。然后再执行本指令，到进行二分查找。

检索条件“Condition”的数据类型为列举型。

Condition	描述
1（EQ_BOTH）	“MN”≦“Member”≦“MX”
2（_EQ_MIN）	“MN”≦“Member”<“MX”
3（_EQ_MAX）	“MN”<“Member”≦“MX”
4（_NE_BOTH）	“MN”<“Member”<“MX”

③程序示例

ST: MODE=0 ,Condition=1 ,线性模式，含上下限，闭区间示例。

Device.Application.POU_ST

表达式	类型	值	准备值	地址	注释
RecRangeSearch_0	RecRangeSearch				
xExecute	BOOL	TRUE			
arr_IN	ARRAY [1..3] OF DUT_Stru				
arr_IN[1]	DUT_Stru				
arr	ARRAY [1..4] OF UINT				
arr[1]	UINT	10			
arr[2]	UINT	2			
arr[3]	UINT	3			
arr[4]	UINT	4			
arr_IN[2]	DUT_Stru				
arr_IN[3]	DUT_Stru				

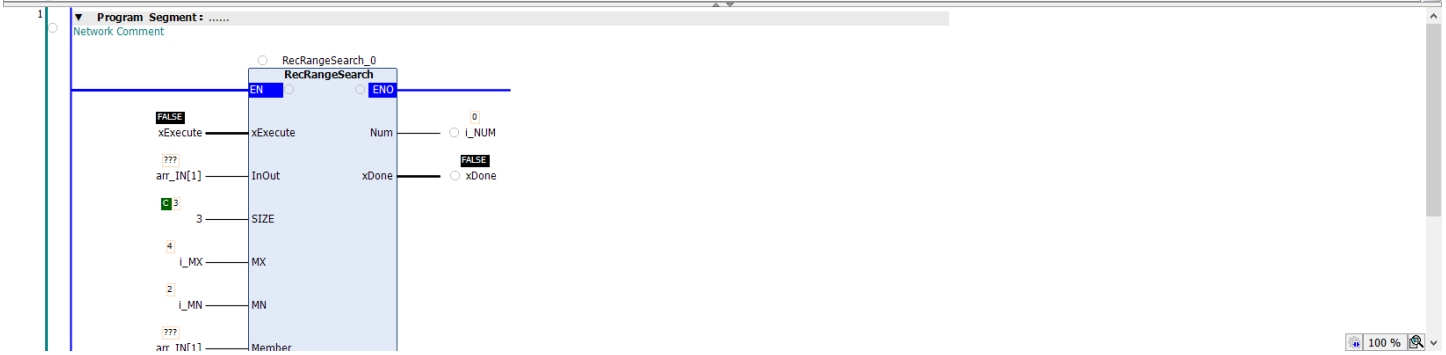
```
RecRangeSearch_0(
    xExecute:= xExecute,
    InOut:= arr_IN[1],
    SIZE:= 3,
    MX:= i_MX,
    MN:= i_MN,
    Member:= arr_IN[1],
    Condition:= EQ_BOTH,
    MODE:= LINER,
    Num=> i_NUM,
    xDone=> xDone,
    InOutPos:= i_InOutPos);
```

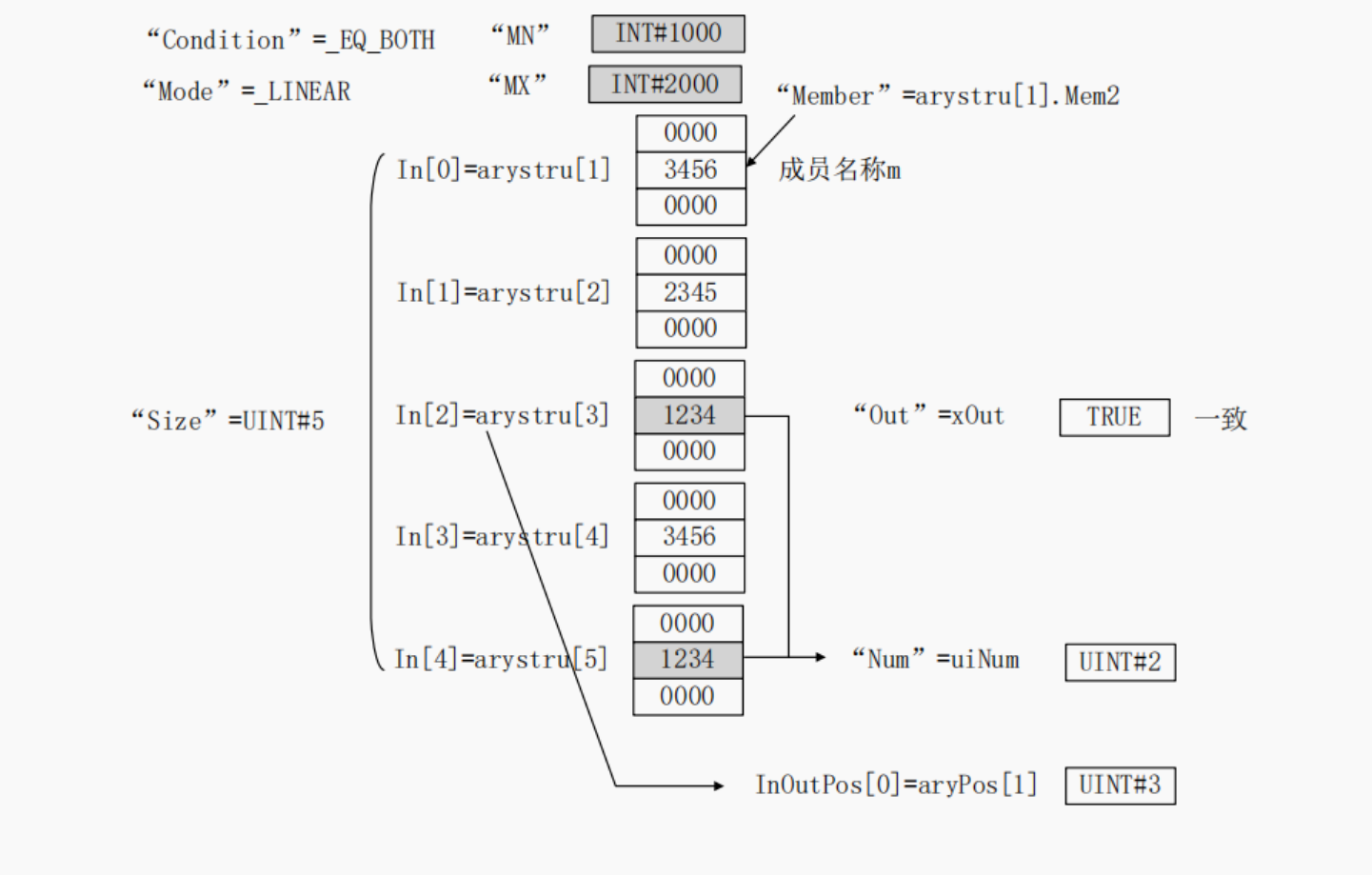
100 %

LD: MODE=0 ,Condition=1 ,线性模式，含上下限，闭区间示例。

Device.Application.POU_LD

表达式	类型	值	准备值	地址	注释
RecRangeSearch_0	RecRangeSearch				
xExecute	BOOL	FALSE			
arr_IN	ARRAY [1..3] OF DUT_Stru				
i_Size	UINT	3			
i_MX	UINT	4			
i_MN	UINT	2			
i_NUM	UINT	0			
i_InOutPos	UINT	0			





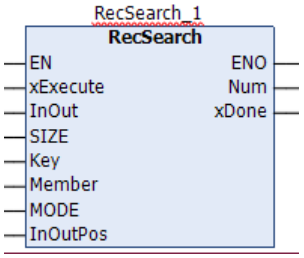
※注意事项

- “Size” 的值超过 In[ ] 的数组区域时，函数返回值的值为 FALSE，Out 不变。
- In[ ] 非 1 维数组时，编译时将会发生异常。
- 请确保“Out” 与检索对象数组 In[ ] 的元素为相同数据类型，否则编译时将会发生异常。
- 检索元素编号“InOutPos”为检索对象数组 In 中的元素编号。
- In[ ] 为实数时，因误差的影响，可能无法获取期望的结果。
- In[ ] 为不支持的数据类型时，编译时将会发生异常。

3.3.13 RecSearch——结构体数组检索指定元素

从将结构体作为元素的排列中，以指定方法检索与检索关键字匹配的元素。

①指令格式

指令	功能	FB/FC	LD 表现	ST表现
RecSearch	结构体数组检索指定元素	FB		RecSearch_0(xExecute:= , InOut:= , SIZE:= , Key:= , Member:= , MODE:= , Num=> , xDone=> , InOutPos:=);

②相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
xExecute	上升沿执行	-	TRUE/FALSE	-	运算对象数组
InOut[]	检索对象数组	-	遵照数据类型	-	运算对象数组
Size	检索对象的元素数	UINT	遵照数据类型	-	检索对象的元素数
Key	检索关键词	-	遵照数据类型	-	检索值
Member	元素成员	-	遵照数据类型	-	元素成员
MODE	模式	UINT	0,1,2	-	0：线性模式 1：升序模式 2：降序模式

输入输出变量

输入输出变量	名称	数据类型	有效范围	初始值	描述
InOutPos	元素位置	UINT	遵照数据类型	-	检索元素位置

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Num	检索值的数量	UINT	遵照数据类型	-	检索值的数量
xDone	执行完成	BOOL	TRUE/FALSE	-	执行完成

数据类型

	布尔	位串				整数								实数		时刻、持续时间、日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
xExecute	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
InOut[]	-	-	-	-	-	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
Size	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
Key	-	-	-	-	-	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√

Member	-	-	-	-	-	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
MODE	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
InOutPos	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
Num	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
xDone	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

功能说明

从以结构体为要素的排列 In[] 的“Size”个元素中，即从 In[1] 到 In[Size] 中，检索结构体的检索对象成员“Member”的值。

将 In[] 中任一元素的检索对象成员作为参数传递到“Member”。

分别将检索结果的要素编号写入 InOutPos 中，检索要素的数量写入“Num”中。检索结果有 2 个以上时，将 In[] 中最低位的检索结果的要素编号带入 InOutPos 中。

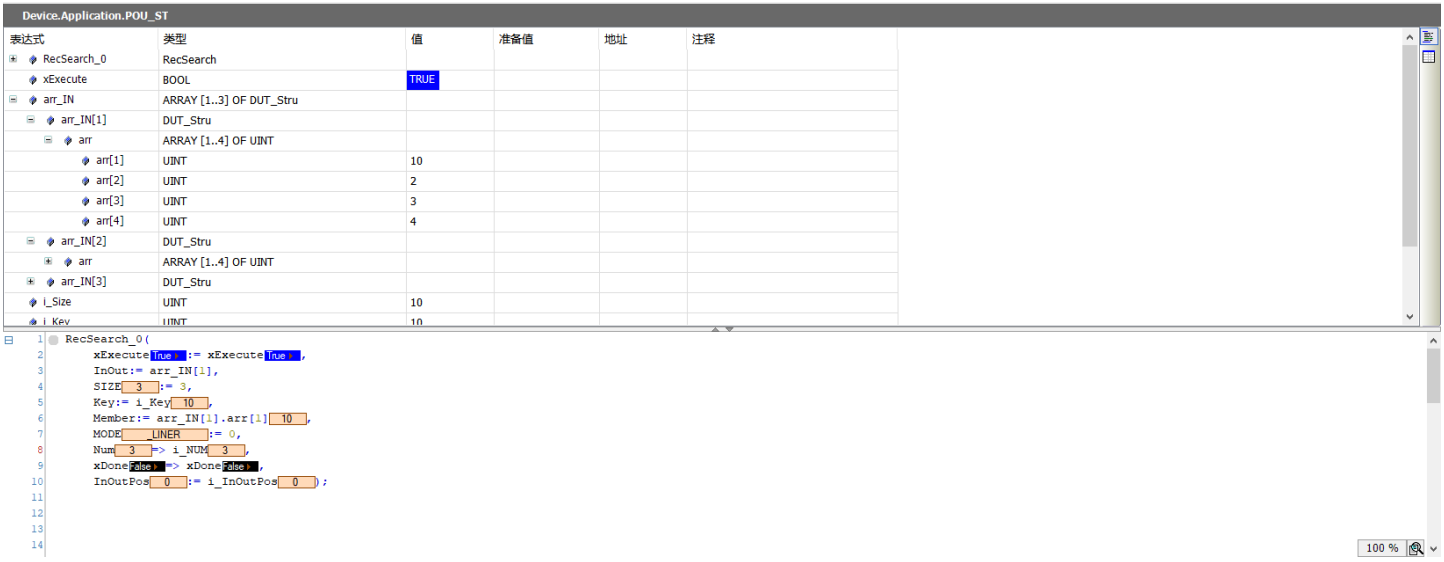
MODE=0 ， 从当前元素顺序遍历整个数组，不依赖数据顺序，使用不需要排序。

MODE=1 ,升序二分查找时，在执行本指令前，In[] 的输入参数的排列元素需要事先升序排列。然后再执行本指令，到进行二分查找

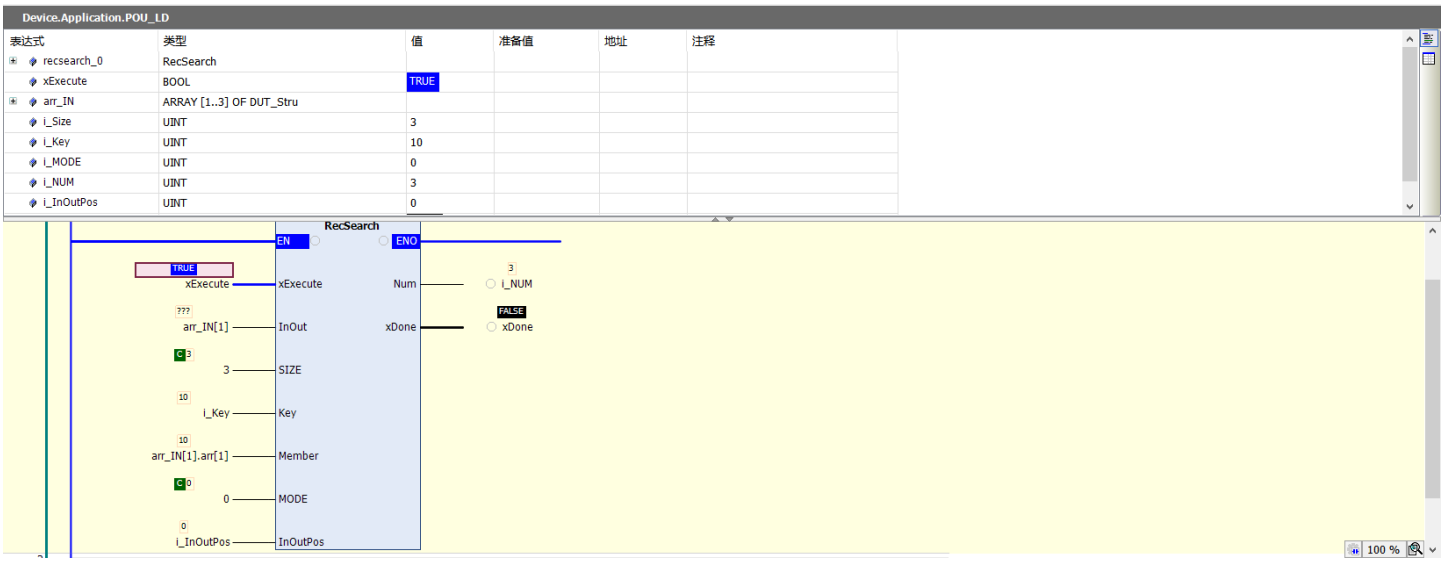
MODE=2 ,降序二分查找时，在执行本指令前，In[] 的输入参数的排列元素需要事先降序排列。然后再执行本指令，到进行二分查找。

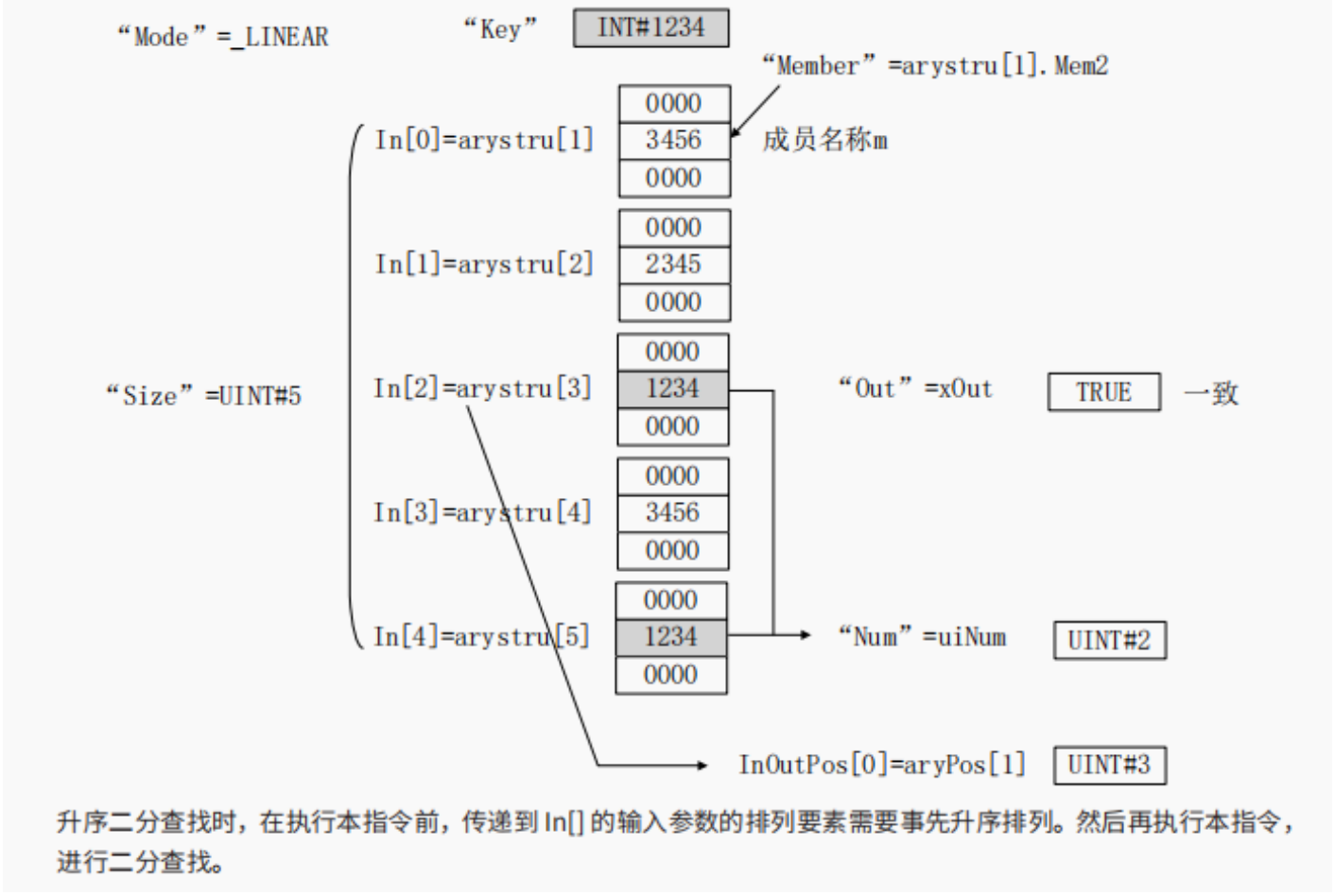
③程序示例

ST: MODE=0 ,线性模式示例。



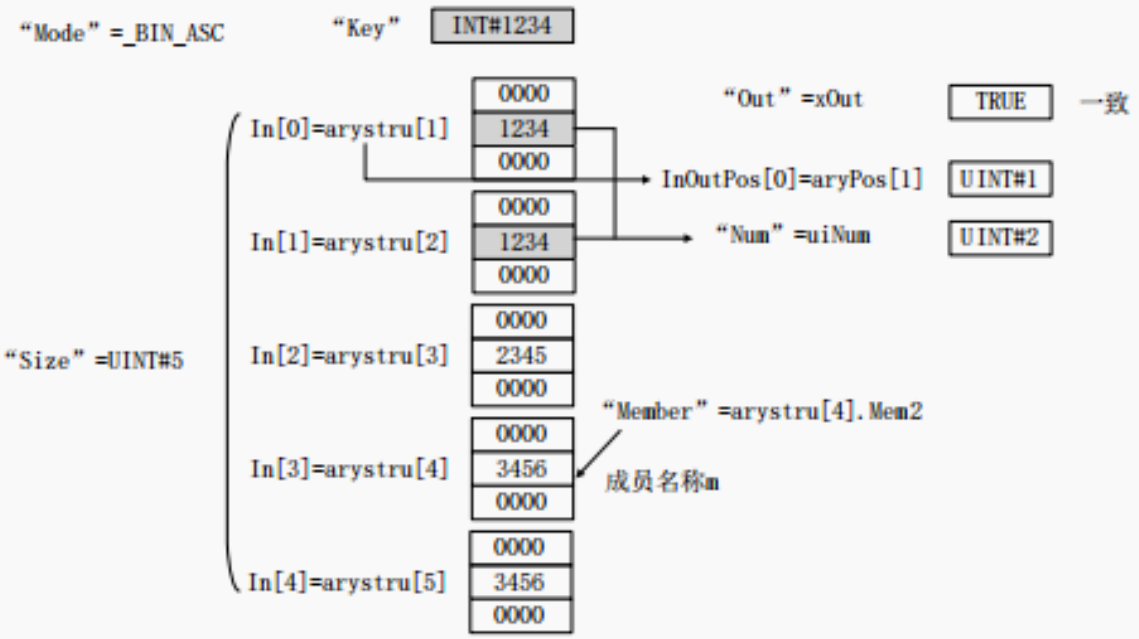
LD: MODE=0 ,线性模式示例





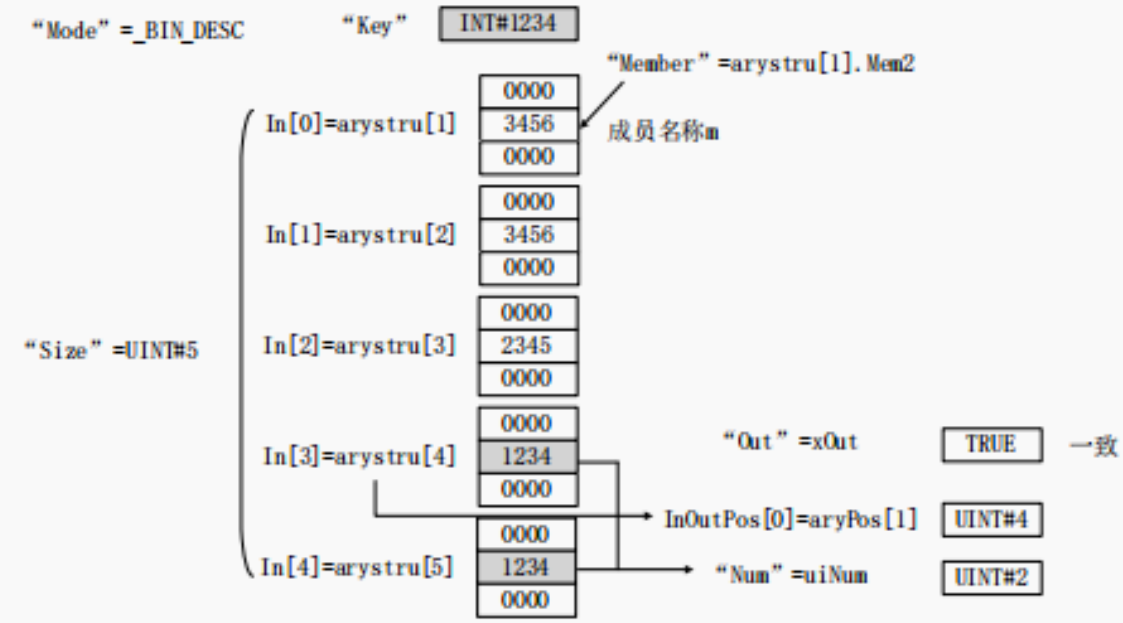
升序二分查找时，在执行本指令前，传递到 In[] 的输入参数的排列要素需要事先升序排列。然后再执行本指令，进行二分查找。

若对前面的示例进行升序二分查找，排列要素的顺序和处理结果如下。



降序二分查找时，在执行本指令前，传递到 In[] 的输入参数的排列要素需要事先降序排列。然后再执行本指令，进行二分查找。

若前面的示例进行降序二分查找，排列要素的顺序和处理结果如下。



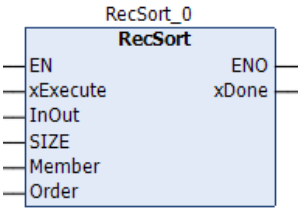
※注意事项

- "Size" 的值超过 In[ ] 的数组区域时，函数返回值的值为 FALSE，Out 不变。
- In[ ] 非 1 维数组时，编译时将会发生异常。
- 请确保 "Out" 与检索对象数组 In[ ] 的元素为相同数据类型，否则编译时将会发生异常。
- 检索元素编号 "InOutPos" 为检索对象数组 In 中的元素编号。
- In[ ] 为实数时，因误差的影响，可能无法获取期望的结果。
- In[ ] 为不支持的数据类型时，编译时将会发生异常。

3.3.14 RecSort——结构体数组元素排序

从将结构体作为元素的排列中排序。

①指令格式

指令	功能	FB/FC	LD 表现	ST表现
RecSor	结构体数组元素排序	FB		RecSort_0(xExecute:= , InOut:= , SIZE:= , Member:= , Order:= , xDone=>);

②相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
xExecute	上升沿执行	-	TRUE/FALSE	-	运算对象数组
InOut[]	检索对象数组	-	遵照数据类型	-	运算对象数组
Size	检索对象的元素数	UINT	遵照数据类型	-	检索对象的元素数
Member	元素成员	-	遵照数据类型	-	元素成员
Orde	排序方式	UINT	1,2	-	1：升序排序 2：降序排序

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
xDone	执行完成	BOOL	TRUE/FALSE	-	执行完成

数据类型

	布尔	位串				整数								实数		时刻、持续时间、日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
xExecute	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
InOut[]	-	-	-	-	-	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
Size	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-

Member	-	-	-	-	-	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
Orde	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
InOutPos	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
xDone	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

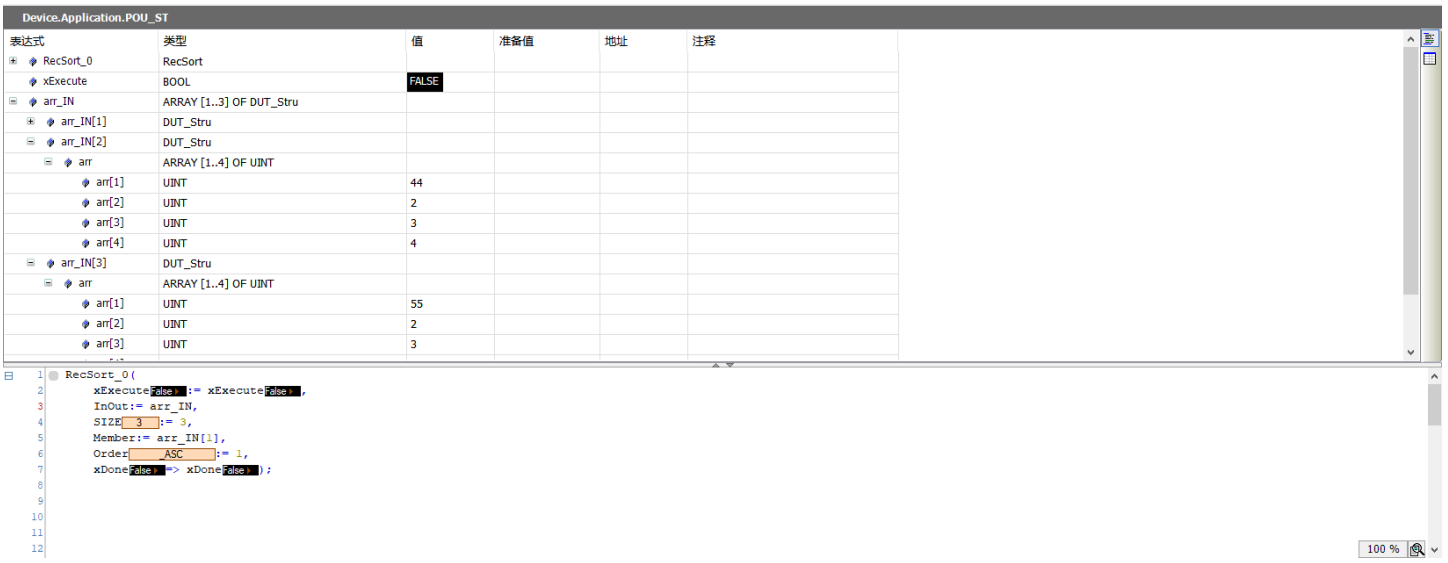
功能说明

从以结构体为要素的排列 In[] 的“Size”个元素中，即从 In[1] 到 In[Size] 中，检索结构体的检索对象成员“Member”的值。

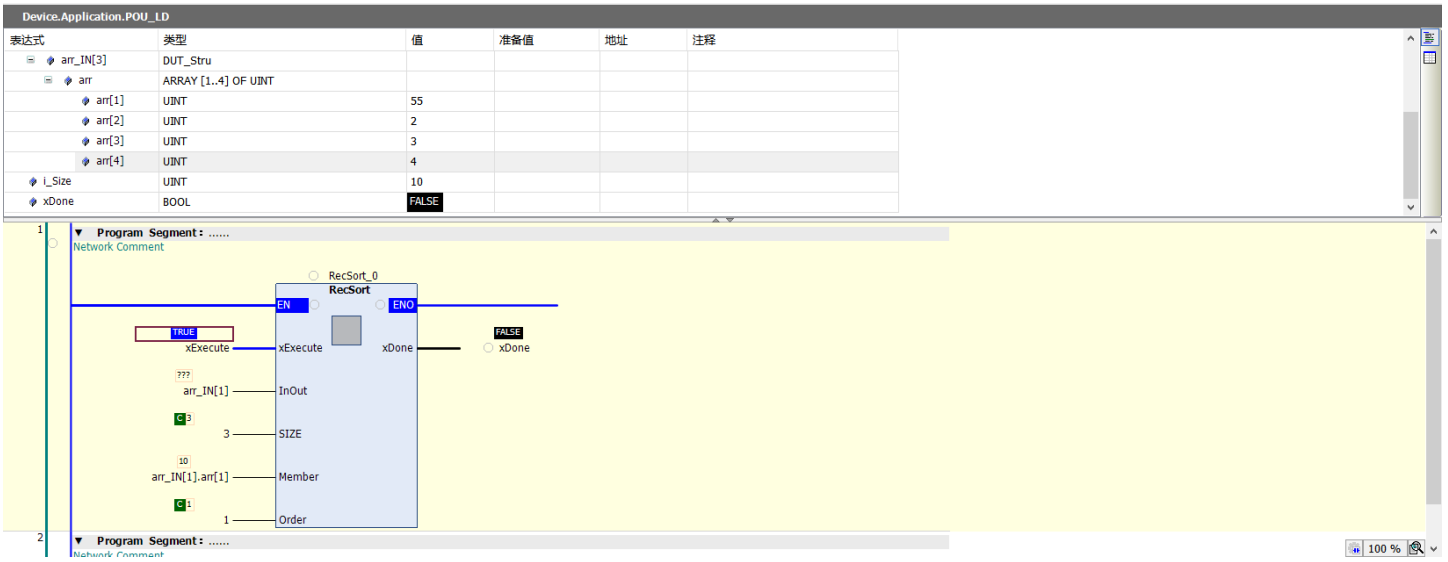
将 In[] 中任一元素的检索对象成员作为参数传递到“Member”。

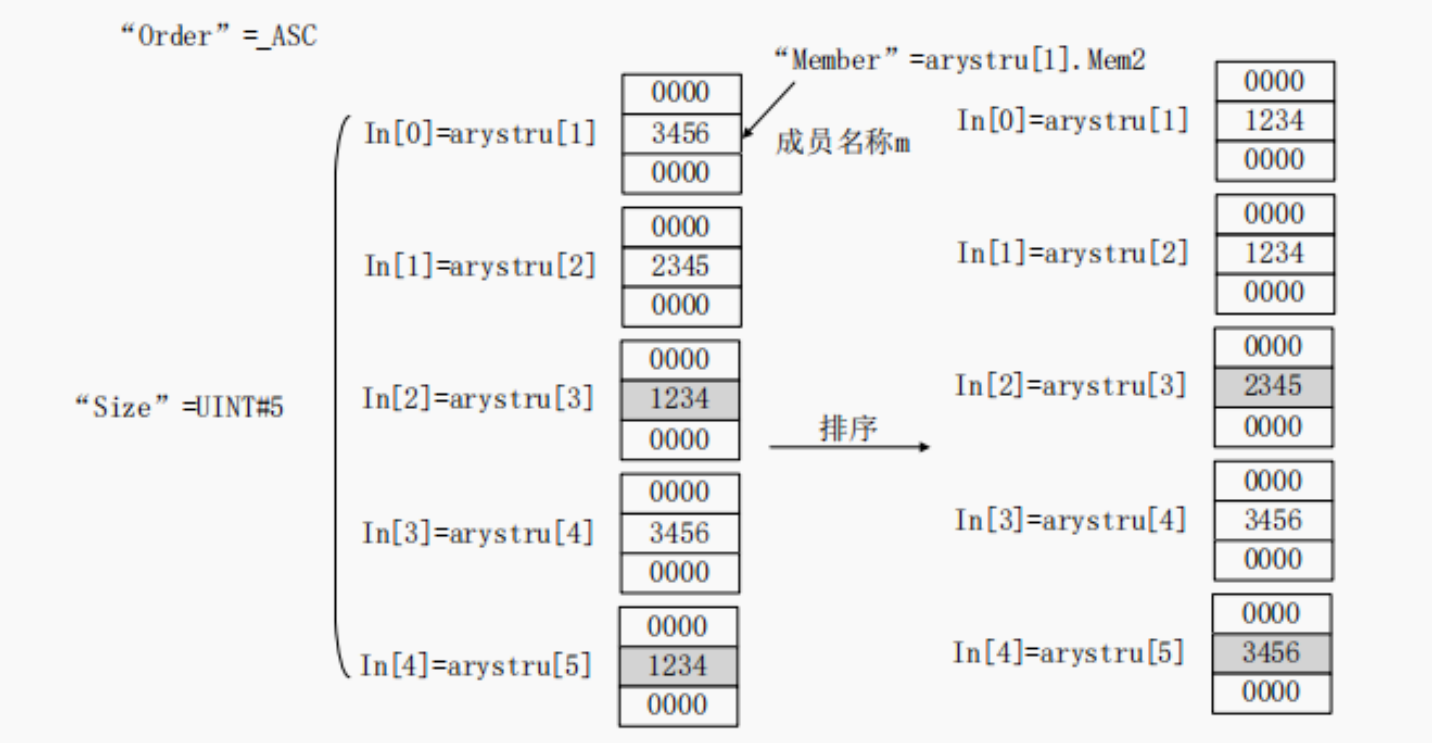
③程序示例

ST：Orde=1 ，升序排序示例。



LD：Orde=1 ，升序排序示例。





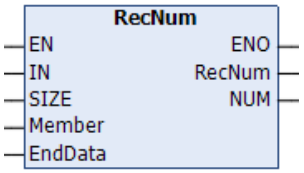
※注意事项

- “Size” 的值超过 In[ ] 的数组区域时，函数返回值的值为 FALSE，Out 不变。
- In[ ] 为实数时，因误差的影响，可能无法获取期望的结果。
- In[ ] 为不支持的数据类型时，编译时将会发生异常。

3.3.15 RecNum——结构体数组元素检索位置

结构体数组元素检索位置。

①指令格式

指令	功能	FB/FC	LD 表现	ST表现
RecSor	结构体数组元素排序	FC		RecNum(RecNum=> , IN:= , SIZE:= , Member:= , EndData:= , NUM=>);

②相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
In[]	检索对象数组	-	遵照数据类型	-	运算对象数组
Size	检索对象的元素数	UINT	遵照数据类型	-	检索对象的元素数
Member	元素成员	-	遵照数据类型	-	元素成员

EndData	结束条件	-	遵照数据类型	-	结束条件

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
NUM	计数结果	UINT	遵照数据类型	-	计数结果

数据类型

	布尔	位串				整数								实数		时刻、持续时间、日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In[]	-	-	-	-	-	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
Size	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
Member	-	-	-	-	-	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
EndData	-	-	-	-	-	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
NUM	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-

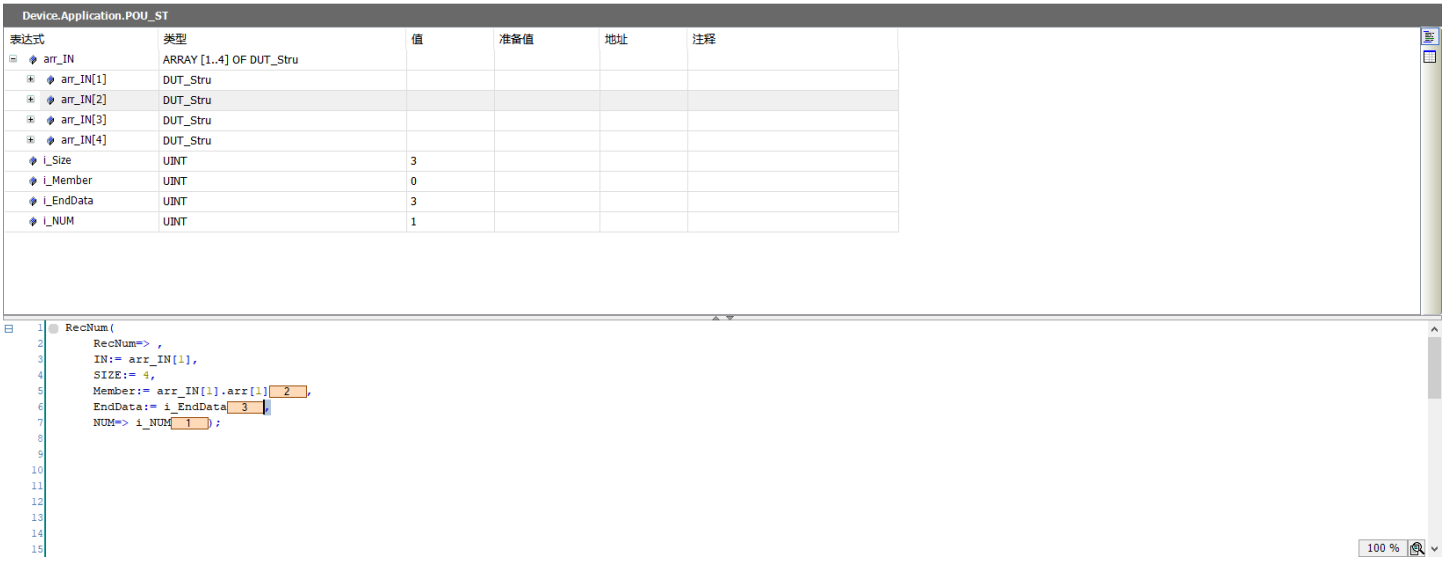
功能说明

从以结构体为要素的排列 In[] 的“Size”个元素中，即从 In[1] 到 In[Size] 中，检索结构体的检索对象成员“Member”的值。

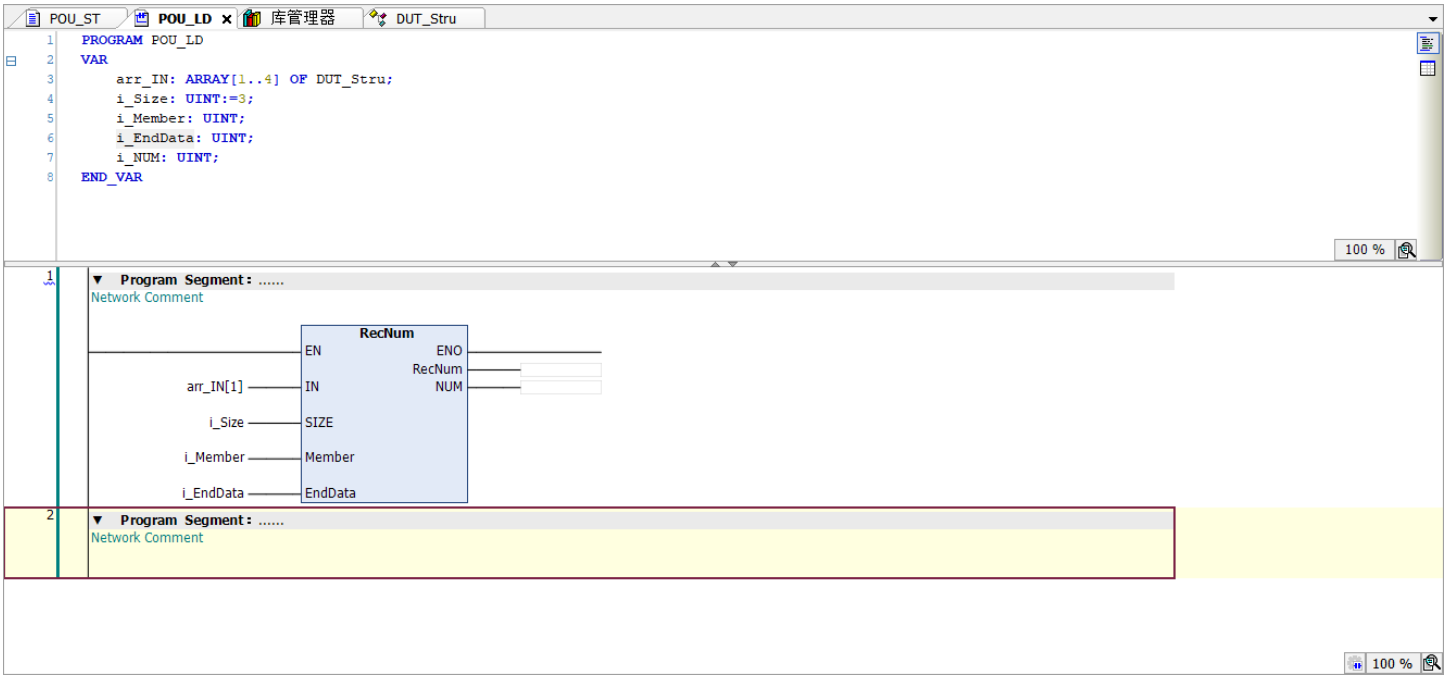
将 In[] 中任一元素的检索对象成员作为参数传递到“Member”。

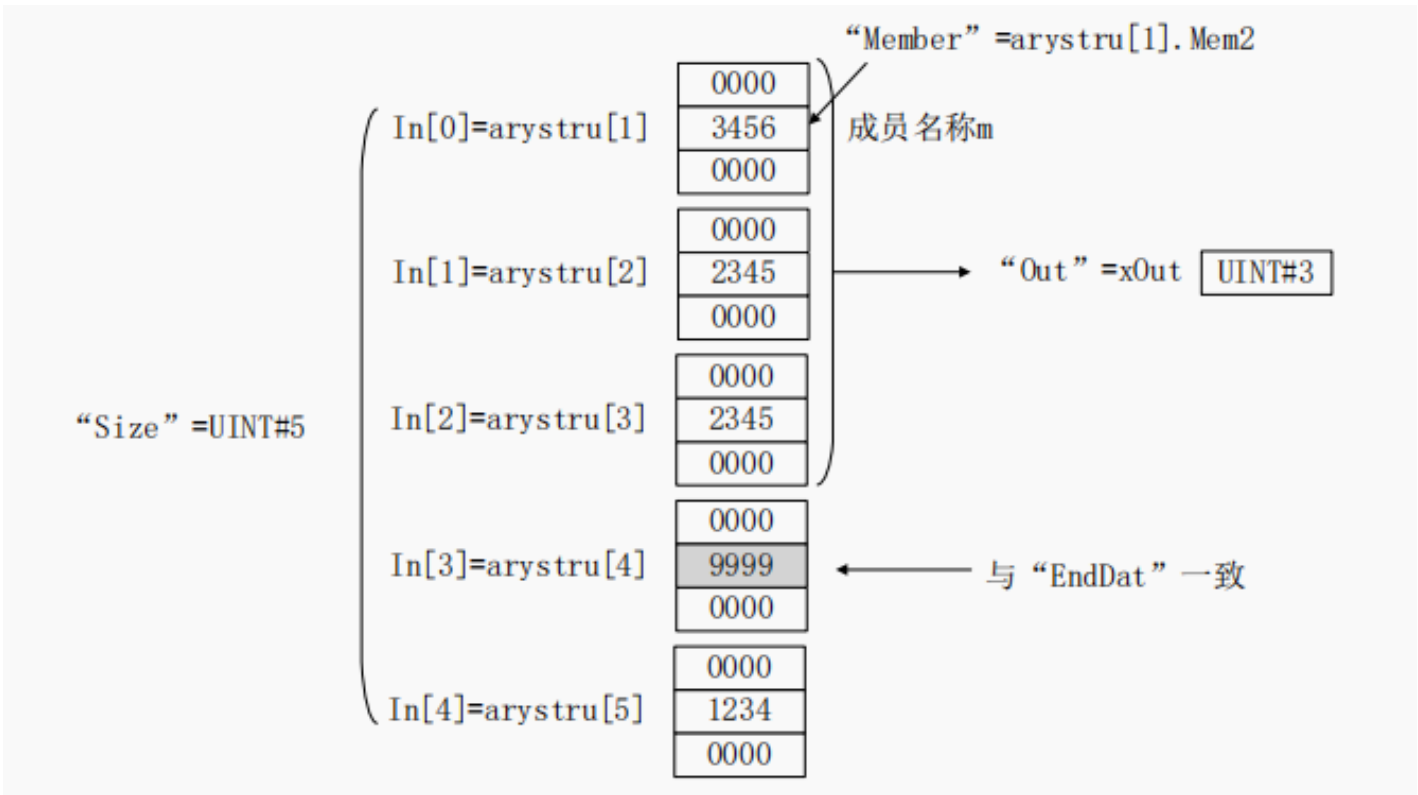
③程序示例

ST:



LD:





※注意事项

- “Size” 的值超过 In[ ] 的数组区域时，函数返回值的值为 FALSE，Out 不变。
- In[ ] 为实数时，因误差的影响，可能无法获取期望的结果。
- In[ ] 为不支持的数据类型时，编译时将会发生异常。

3.4 数组比较相关

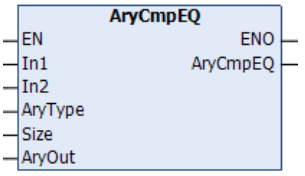
指令列表

指令类别	名称	FB/FC	功能
数组移位相关	AryCmpEQ	FC	数组批量比较EQ
	AryCmpEQV	FC	数组批量比较EQV
	AryCmpEQ_Element	FC	数组批量比较EQ_Element
	AryCmppNE	FC	数组批量比较NE
	AryCmpNEV	FC	数组批量比较NEV
	AryCmpNE_Element	FC	数组批量比较NE_Element
	AryCmpGE	FC	数组批量比较GE
	AryCmpGEV	FC	数组批量比较GEV
	AryCmpGT	FC	数组批量比较GT
	AryCmpGTV	FC	数组批量比较GTV
	AryCmpLE	FC	数组批量比较LE
	AryCmpLEV	FC	数组批量比较LEV
	AryCmpLT	FC	数组批量比较LT
	AryCmpLTV	FC	数组批量比较LTV
	TableCompare	FC	表比较
	ZoneCompare	FC	区域比较

3.4.1 AryCmpEQ——数组批量比较EQ

比较两个数组的各要素是否相等。

①指令格式

指令	功能	FB/FC	LD 表现	ST表现
AryCmpEQ	数组批量比较EQ	FC		AryCmpEQ(AryCmpEQ=> , In1:= , In2:= , AryType:= , Size:= , AryOut:=);

②相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
IN1[]	对象数组1	-	遵照数据类型	-	对象数组1
IN2[]	对象数组2	-	遵照数据类型	-	对象数组2
AryType:	数据类型	-	遵照数据类型	1	对象的元素
Size	比较元素数	UINT	遵照数据类型	1	比较元素数
AryOut:[]	比较结果	UINT	遵照数据类型	1	比较结果

数据类型

	布尔	位串				整数								实数		时刻、持续时间、日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
IN1[]	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	-	-	-	-	-
IN2[]	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	-	-	-	-	-
AryType	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	-	-	-	-	-
Size	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
AryOut:[]	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

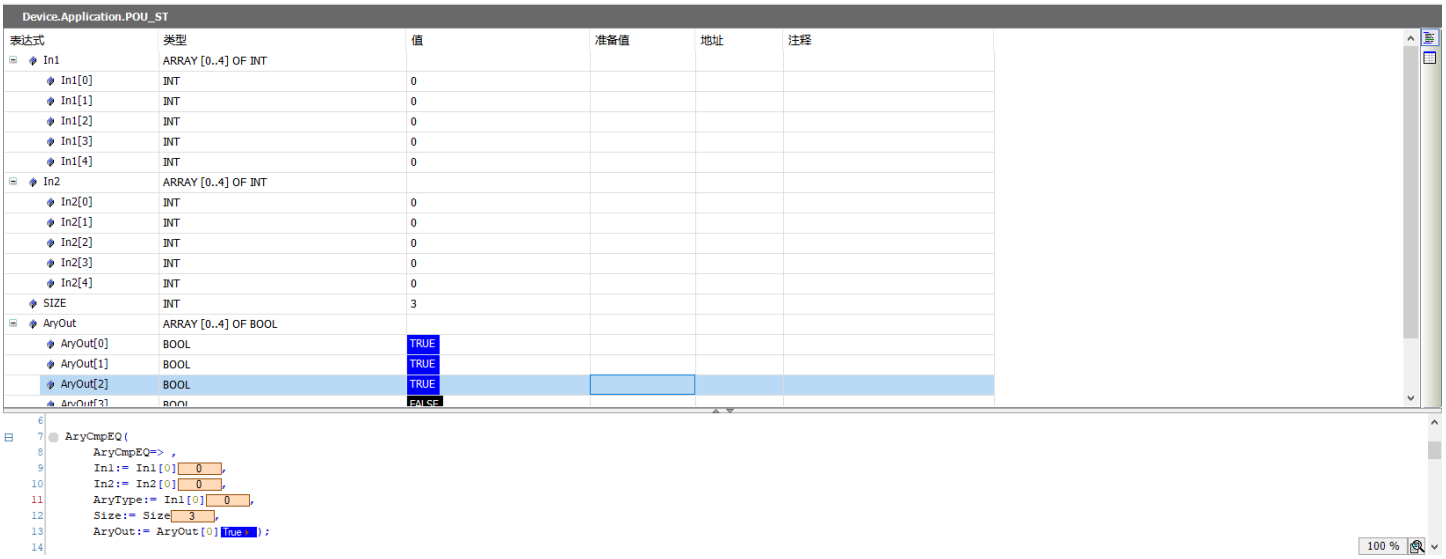
功能说明

对两个数组In1[0]～In1[“Size”-1]、In2[0]～In2[“Size”-1]的相同要素编号进行比较是否相等。比较结果保存到与比较结果排列AryOut[0]~AryOut[“Size”-1]对应的要素编号中。

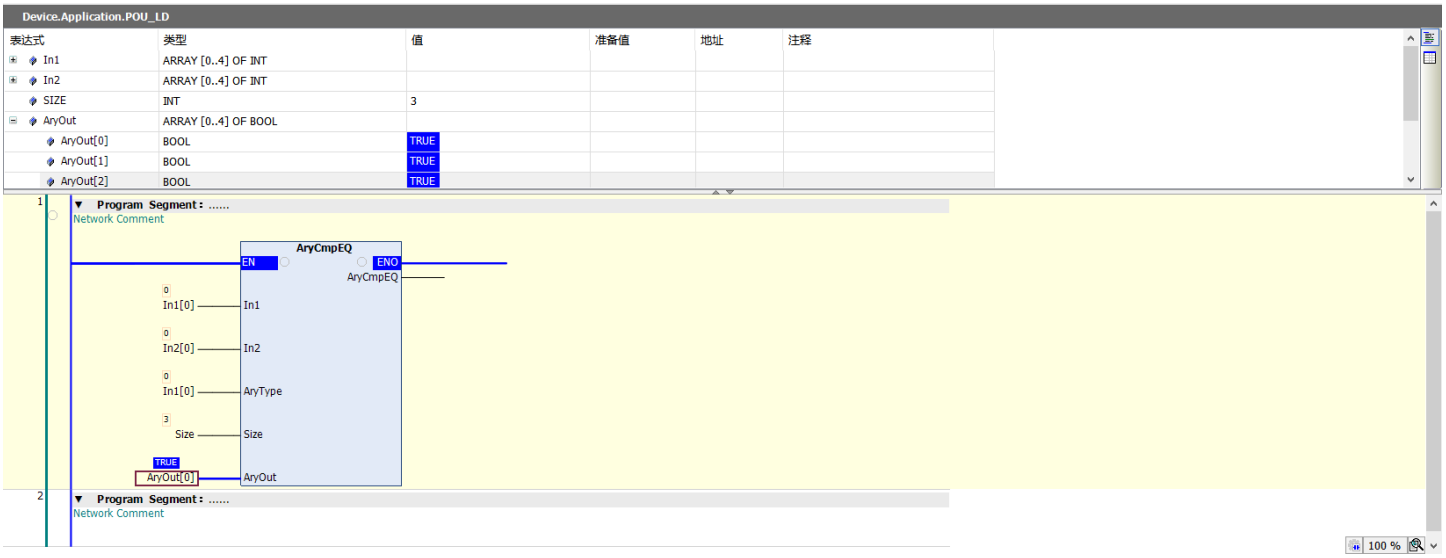
其中In1[i]=In2[]时，AryOut[i]为TRUE，否则为FALSE。

③程序示例

ST:



LD:



※注意事项

•In1[] 和 In2[] 的数据类型应统一，否则编译时将会发生异常。

•In1[]、In2[] 为实数、且包括除不尽的除法运算结果时，受取整误差的影响，处理结果可能与预计的不同。

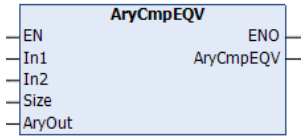
•“Size”的值为 0 时，函数返回值为 TRUE，AryOut[] 无变化。

•“Size”的值超出 In1[]、In2[]、AryOut[] 中任一数组区域时，函数返回值为 FALSE，AryOut[] 无变化。

3.4.2 AryCmpEQV——数组批量比较EQV

比较数组元素和数值是否相等。

①指令格式

指令	功能	FB/FC	LD 表现	ST表现
AryCmpEQV	数组批量比较EQV	FC		AryCmpEQV(AryCmpEQV=> , In1:= , In2:= , Size:= , AryOut:=);

②相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
In1[]	对象数组1	-	遵照数据类型	-	对象数组1
In2	对象2	-	遵照数据类型	-	对象数组2
Size	比较元素数	UINT	遵照数据类型	1	比较元素数
AryOut:[]	比较结果	UINT	遵照数据类型	1	比较结果

数据类型

	布尔	位串				整数								实数		时刻、持续时间、日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In1[]	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	-	-	-	-	-
In2	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	-	-	-	-	-
Size	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
AryOut:[]	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

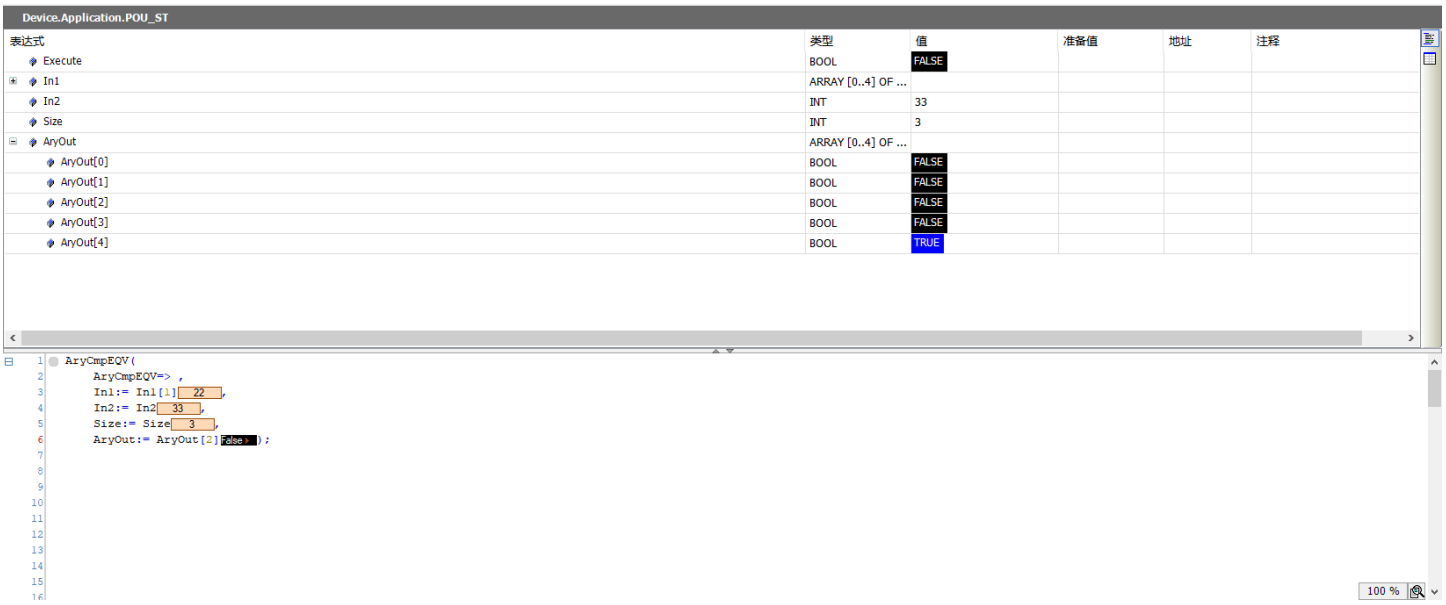
功能说明

对比较数组 In1[0] ～ In1[“Size”-1] 和比较变量“In2”进行比较是否相等。比较结果保存到与比较结果排列 AryOut[0] ～ AryOut[“Size”-1] 对应的要素编号中。

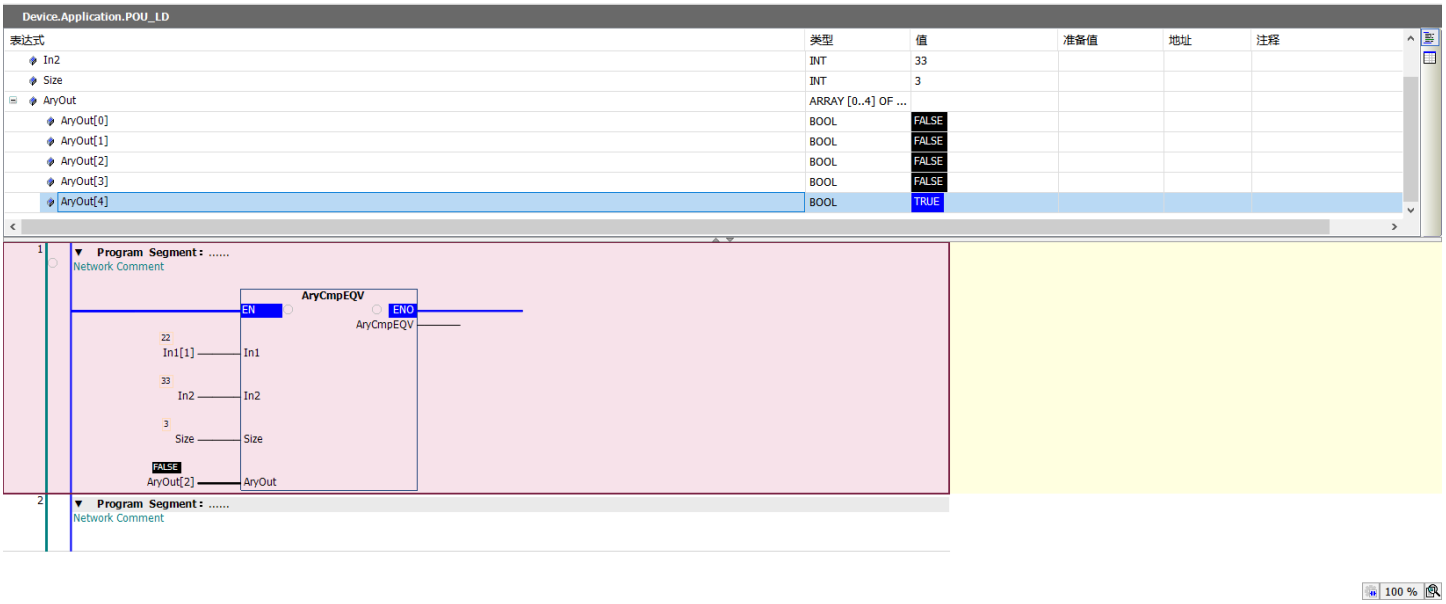
其中 In1[i] = In2 时，AryOut[i] 为 TRUE，否则为 FALSE。

③程序示例

ST:



LD:



※注意事项

•In1[] 和 In2 的数据类型应统一，否则编译时将会发生异常。

•In1[]、In2 为实数、且包括除不尽的除法运算结果时，受取整误差的影响，处理结果可能与预计的不同。

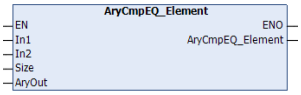
•“Size”的值为 0 时，函数返回值为 TRUE，AryOut[] 无变化。

•“Size”的值超出 In1[]、AryOut[] 中任一数组区域时，函数返回值为 FALSE，AryOut[] 无变化。

3.4.3 AryCmpEQ_Element——数组批量比较EQ_Element

比较数组元素和数值是否相等。

①指令格式

指令	功能	FB/FC	LD 表现	ST表现
AryCmpEQ_Element	数组批量比较EQ_Element	FC		AryCmpEQ_Element(AryCmpEQ_Element=> , In1:= , In2:= , Size:= , AryOut:=);

②相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
In1[]	对象数组1	-	遵照数据类型	-	对象数组1
In2[]	对象数组2	-	遵照数据类型	-	对象数组2
Size	比较元素数	UINT	遵照数据类型	1	比较元素数
AryOut:[]	比较结果	UINT	遵照数据类型	1	比较结果

数据类型

	布尔	位串				整数								实数		时刻、持续时间、日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In1[]	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	-	-	-	-	-
In2[]	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	-	-	-	-	-
Size	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
AryOut:[]	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

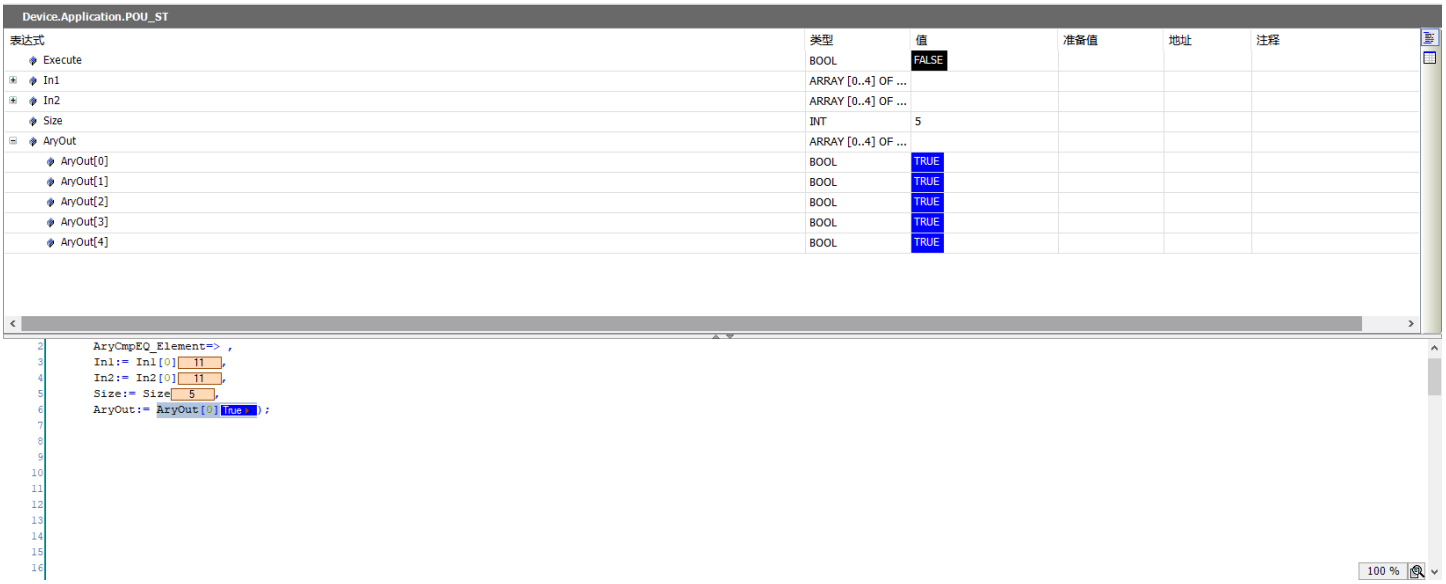
功能说明

对比较数组 In1[0] ～ In1[“Size”-1] 和比较变量“In2”进行比较是否相等。比较结果保存到与比较结果排列 AryOut[0] ～ AryOut[“Size”-1] 对应的要素编号中。

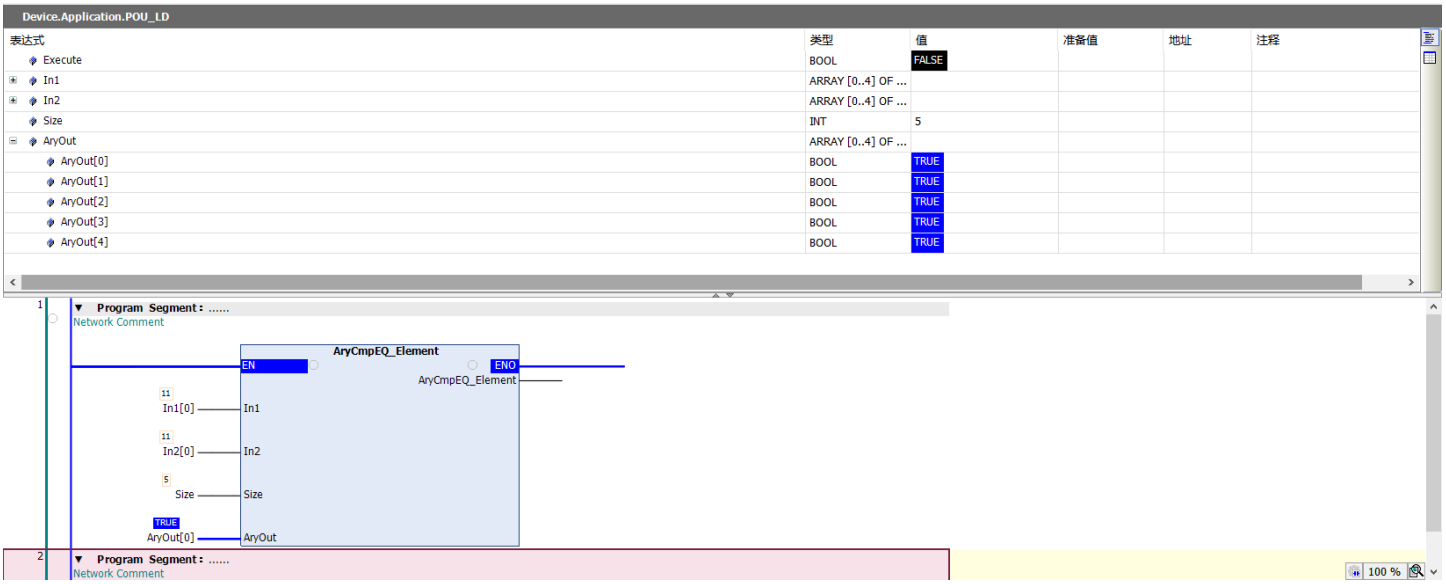
其中 In1[i] = In2 时，AryOut[i] 为 TRUE，否则为 FALSE。

③程序示例

ST:



LD:



※注意事项

•In1[] 和 In2 的数据类型应统一，否则编译时将会发生异常。

•In1[]、In2 为实数、且包括除不尽的除法运算结果时，受取整误差的影响，处理结果可能与预计的不同。

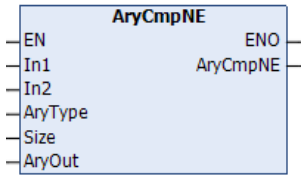
•“Size”的值为 0 时，函数返回值为 TRUE，AryOut[] 无变化。

•“Size”的值超出 In1[]、AryOut[] 中任一数组区域时，函数返回值为 FALSE，AryOut[] 无变化。

3.4.4 AryCmppNE——数组批量比较NE

比较两个数组的各要素是否不同。

①指令格式

指令	功能	FB/FC	LD 表现	ST表现
AryCmppNE	数组批量比较NE	FC		AryCmpNE(AryCmpNE=> , In1:= , In2:= , AryType:= , Size:= , AryOut:=);

②相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
In1[]	对象数组1	-	遵照数据类型	-	对象数组1
In2[]	对象数组2	-	遵照数据类型	-	对象数组2
AryType:	数据类型	-	遵照数据类型	1	对象的元素
Size	比较元素数	UINT	遵照数据类型	1	比较元素数
AryOut:[]	比较结果	UINT	遵照数据类型	1	比较结果

数据类型

	布尔	位串				整数								实数		时刻、持续时间、日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In1[]	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	-	-	-	-	-
In2[]	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	-	-	-	-	-
AryType	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	-	-	-	-	-
Size	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
AryOut:[]	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

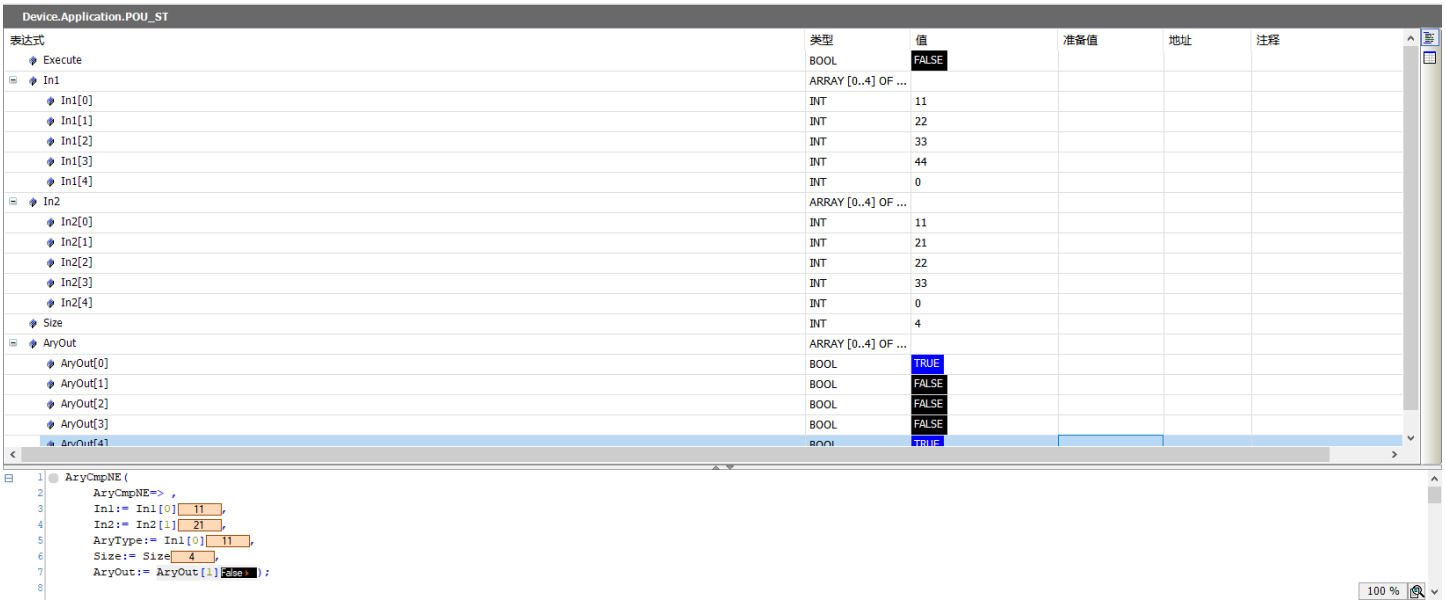
功能说明

对两个数组In1[0]～In1[“Size”-1]、In2[0]～In2[“Size”-1]的相同要素编号进行比较是否不同。比较结果保存到与比较结果排列 AryOut[0]～ AryOut[“Size”-1]对应的要素编号中。

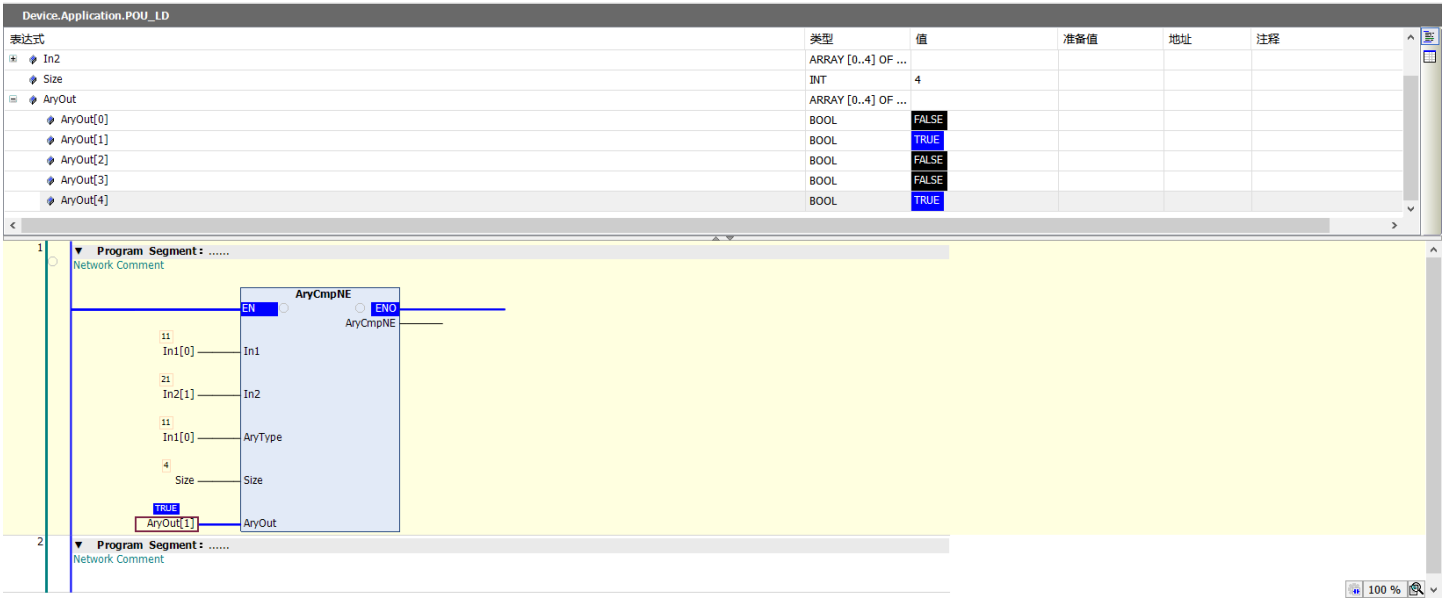
其中In1[i]<>In2[i]时，AryOut[i]为TRUE，否则为FALSE。

③程序示例

ST:



LD:



※注意事项

•In1[] 和 In2[] 的数据类型应统一，否则编译时将会发生异常。

•In1[]、In2[] 为实数、且包括除不尽的除法运算结果时，受取整误差的影响，处理结果可能与预计的不同。

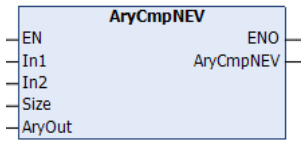
•“Size”的值为 0 时，函数返回值为 TRUE，AryOut[] 无变化。

•“Size”的值超出 In1[]、In2[]、AryOut[] 中任一数组区域时，函数返回值为 FALSE，AryOut[] 无变化。

3.4.5 AryCmpNEV——数组批量比较NEV

比较数组元素和数值是否不同。

①指令格式

指令	功能	FB/FC	LD 表现	ST表现
AryCmpNEV	数组批量比较NEV	FC		AryCmpNEV(AryCmpNEV=> , In1:= , In2:= , Size:= , AryOut:=);

②相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
In1[]	对象数组1	-	遵照数据类型	-	对象数组1
In2	对象2	-	遵照数据类型	-	对象数组2
Size	比较元素数	UINT	遵照数据类型	1	比较元素数
AryOut:[]	比较结果	UINT	遵照数据类型	1	比较结果

数据类型

	布尔	位串				整数								实数		时刻、持续时间、日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In1[]	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	-	-	-	-	-
In2	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	-	-	-	-	-
Size	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
AryOut:[]	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

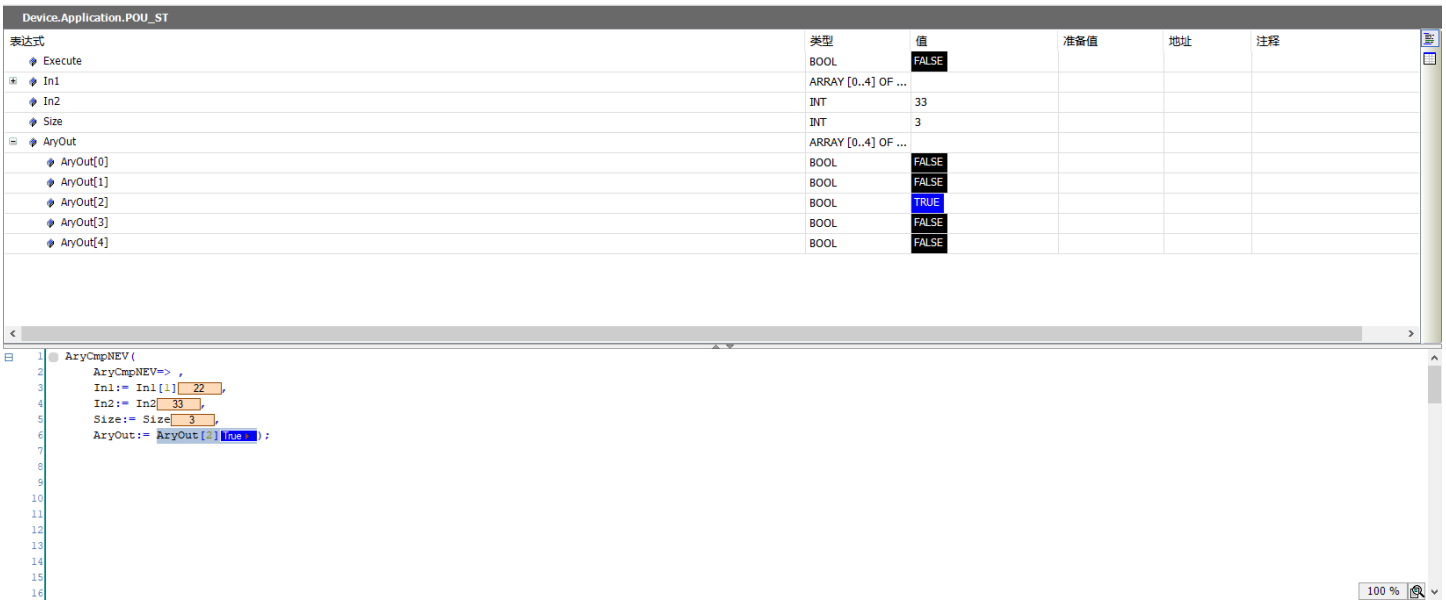
功能说明

对比较数组 In1[0] ～ In1[“Size”-1] 和比较变量“In2”进行比较是否不同。比较结果保存到与比较结果排列 AryOut[0] ～ AryOut[“Size”-1] 对应的要素编号中。

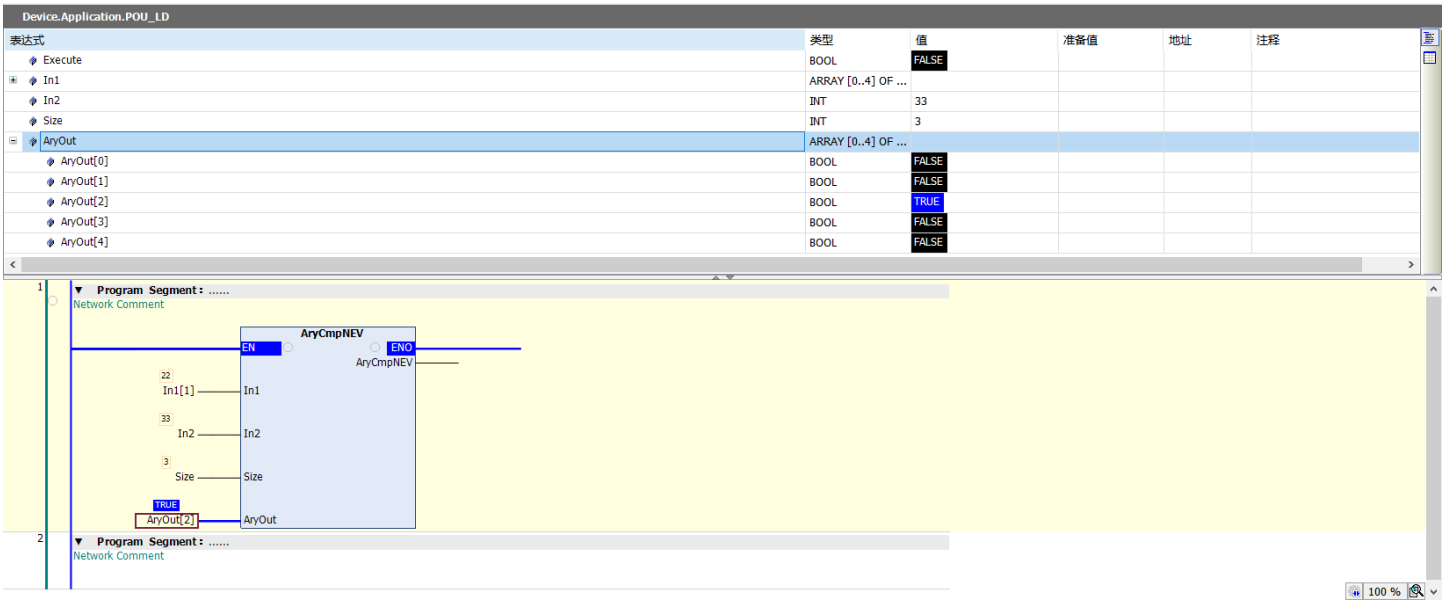
其中 In1[i] <> In2 时，AryOut[i] 为 TRUE，否则为 FALSE。

③程序示例

ST:



LD:



※注意事项

•In1[] 和 In2 的数据类型应统一，否则编译时将会发生异常。

•In1[]、In2 为实数、且包括除不尽的除法运算结果时，受取整误差的影响，处理结果可能与预计的不同。

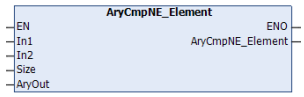
•“Size”的值为 0 时，函数返回值为 TRUE，AryOut[] 无变化。

•“Size”的值超出 In1[]、AryOut[] 中任一数组区域时，函数返回值为 FALSE，AryOut[] 无变化。

3.4.6 AryCmpNE_Element——数组批量比较NE_Element

比较数组元素和数值是否不同。

①指令格式

指令	功能	FB/FC	LD 表现	ST表现
AryCmpNE_Element	数组批量比较 NE_Element	FC		AryCmpNE_Element(AryCmpNE_Element=> , In1:= , In2:= , Size:= , AryOut:=);

②相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
In1[]	对象数组1	-	遵照数据类型	-	对象数组1
In2	对象2	-	遵照数据类型	-	对象数组2
Size	比较元素数	UINT	遵照数据类型	1	比较元素数
AryOut:[]	比较结果	UINT	遵照数据类型	1	比较结果

数据类型

	布尔	位串				整数								实数		时刻、持续时间、日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In1[]	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	-	-	-	-	-
In2	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	-	-	-	-	-
Size	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
AryOut:[]	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

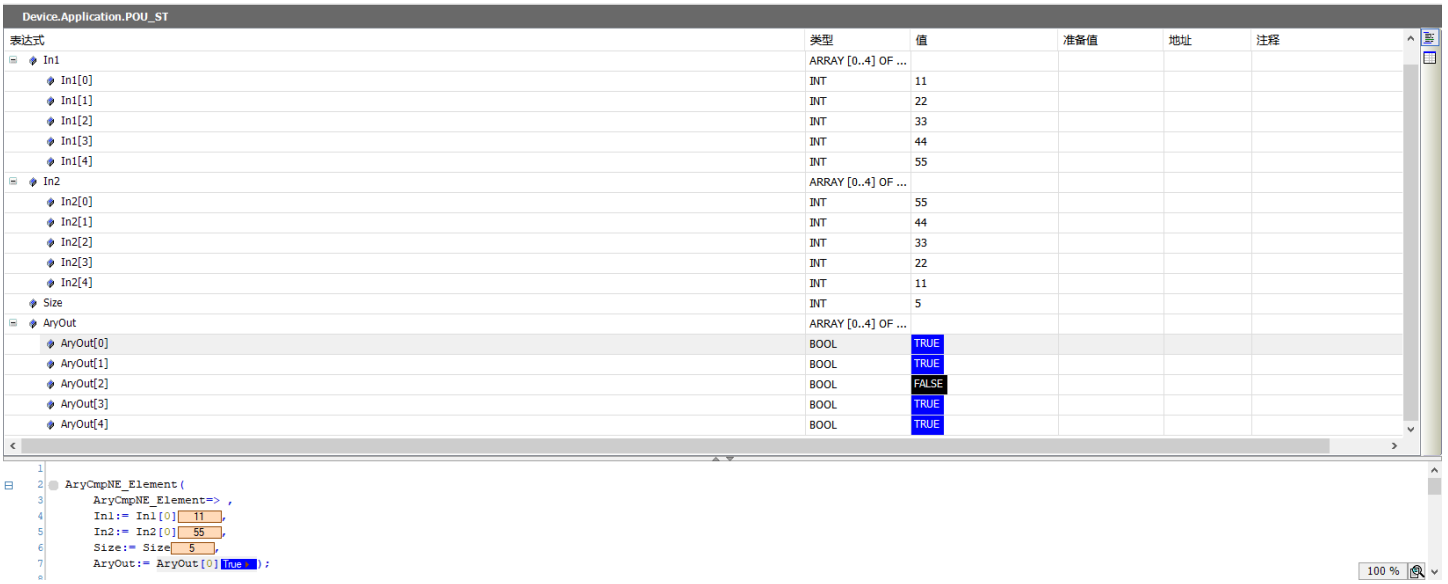
功能说明

对比较数组 In1[0] ～ In1[“Size”-1] 和比较变量“In2”进行比较是否不同。比较结果保存到与比较结果排列 AryOut[0] ～ AryOut[“Size”-1] 对应的要素编号中。

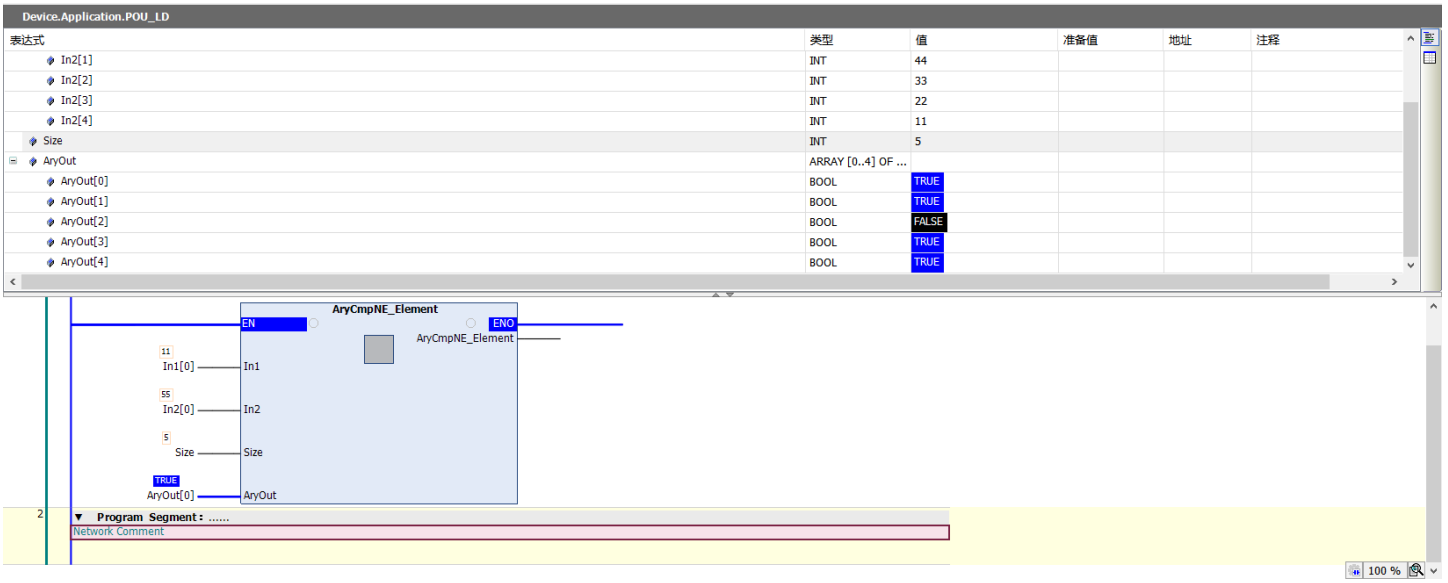
其中 In1[i] <> In2 时，AryOut[i] 为 TRUE，否则为 FALSE。

③程序示例

ST:



LD:



※注意事项

•In1[] 和 In2 的数据类型应统一，否则编译时将会发生异常。

•In1[]、In2 为实数、且包括除不尽的除法运算结果时，受取整误差的影响，处理结果可能与预计的不同。

•“Size”的值为 0 时，函数返回值为 TRUE，AryOut[] 无变化。

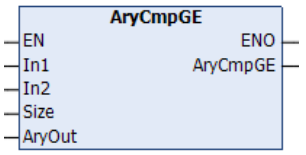
•“Size”的值超出 In1[]、AryOut[] 中任一数组区域时，函数返回值为 FALSE，AryOut[] 无变化。

3.4.7 AryCmpGE——数组批量比较GE

对两个数组的各要素进行大小比较（大于等于）。

①指令格式

指令	功能	FB/FC	LD 表现	ST表现

AryCmpGE	数组批量比较GE	FC		AryCmpGE(AryCmpGE=> , In1:= , In2:= , Size:= , AryOut:=);

②相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
In1[]	对象数组1	-	遵照数据类型	-	对象数组1
In2[]	对象数组2	-	遵照数据类型	-	对象数组2
Size	比较元素数	UINT	遵照数据类型	1	比较元素数
AryOut:[]	比较结果	UINT	遵照数据类型	1	比较结果

数据类型

	布尔	位串				整数								实数		时刻、持续时间、日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In1[]	-	-	-	-	-	√	√	√	√	√	√	√	√	√	√	-	-	-	-	-
In2[]	-	-	-	-	-	√	√	√	√	√	√	√	√	√	√	-	-	-	-	-
Size	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
AryOut:[]	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

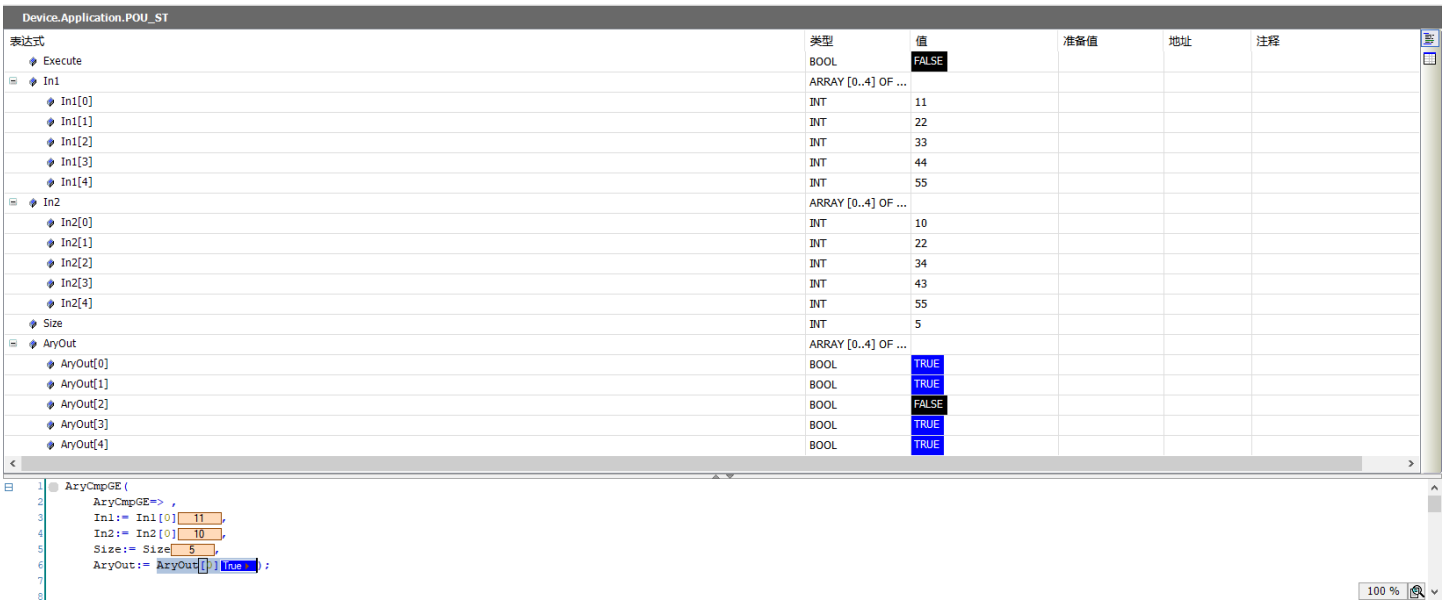
功能说明

对两个数组 In1[0] ～ In1[“Size”-1]、In2[0] ～ In2[“Size”-1] 的相同要素编号进行大小比较（大于等于）。比较结果保存到与比较结果排列 AryOut[0] ～ AryOut[“Size”-1] 对应的要素编号中。

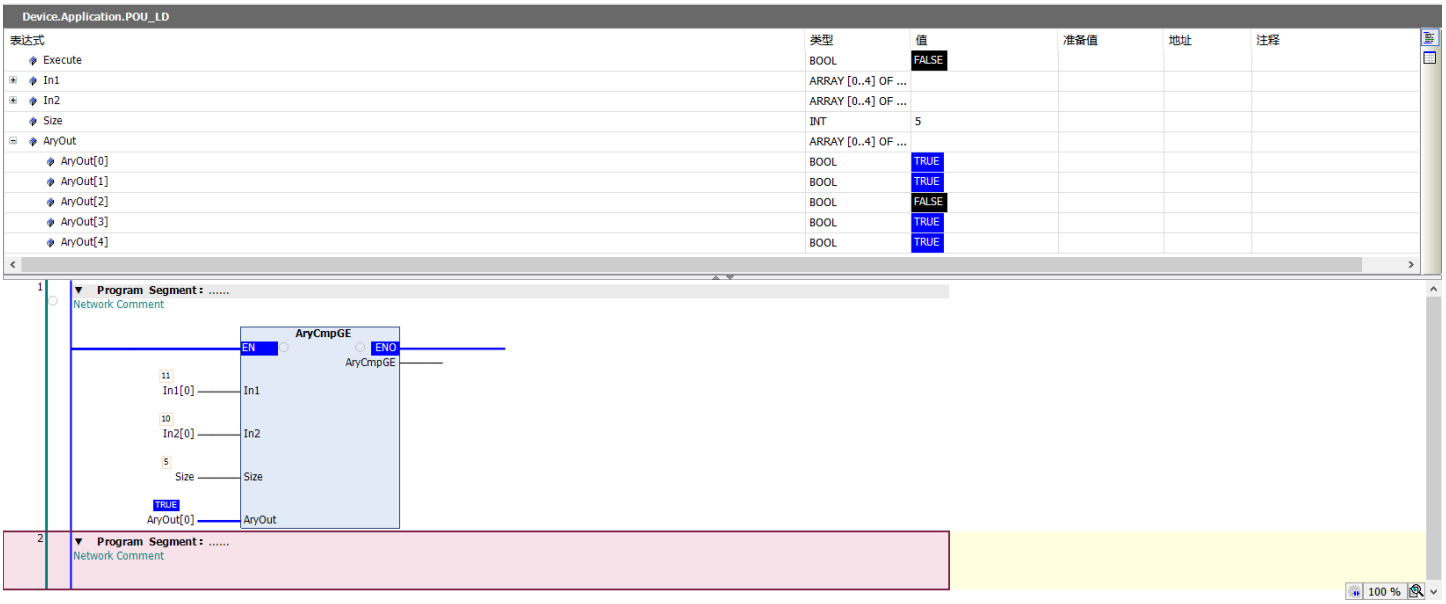
其中 In1[i] >= In2[i] 时，AryOut[i] 为 TRUE，否则为 FALSE。

③程序示例

ST:



LD:



※注意事项

In1[] 和 In2[] 的数据类型应统一，否则编译时将会发生异常。

In1[]、In2[] 为实数、且包括除不尽的除法运算结果时，受取整误差的影响，处理结果可能与预计的不同。

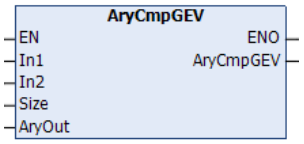
“Size”的值为 0 时，函数返回值为 TRUE，AryOut[] 无变化。

“Size”的值超出 In1[]、In2[]、AryOut[] 中任一数组区域时，函数返回值为 FALSE，AryOut[] 无变化。

3.4.8 AryCmpGEV——数组批量比较GEV

对数组的各要素和变量进行大小比较（大于等于）。

①指令格式

指令	功能	FB/FC	LD 表现	ST表现
AryCmpGEV	数组批量比较GEV	FC		AryCmpGEV(AryCmpGEV=> , In1:= , In2:= , Size:= , AryOut:=);

②相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
In1[]	对象数组1	-	遵照数据类型	-	对象数组1
In2	对象2	-	遵照数据类型	-	对象数组2
Size	比较元素数	UINT	遵照数据类型	1	比较元素数
AryOut:[]	比较结果	UINT	遵照数据类型	1	比较结果

数据类型

	布尔	位串				整数								实数		时刻、持续时间、日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In1[]	-	-	-	-	-	√	√	√	√	√	√	√	√	√	√	-	-	-	-	-
In2	-	-	-	-	-	√	√	√	√	√	√	√	√	√	√	-	-	-	-	-
Size	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
AryOut:[]	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

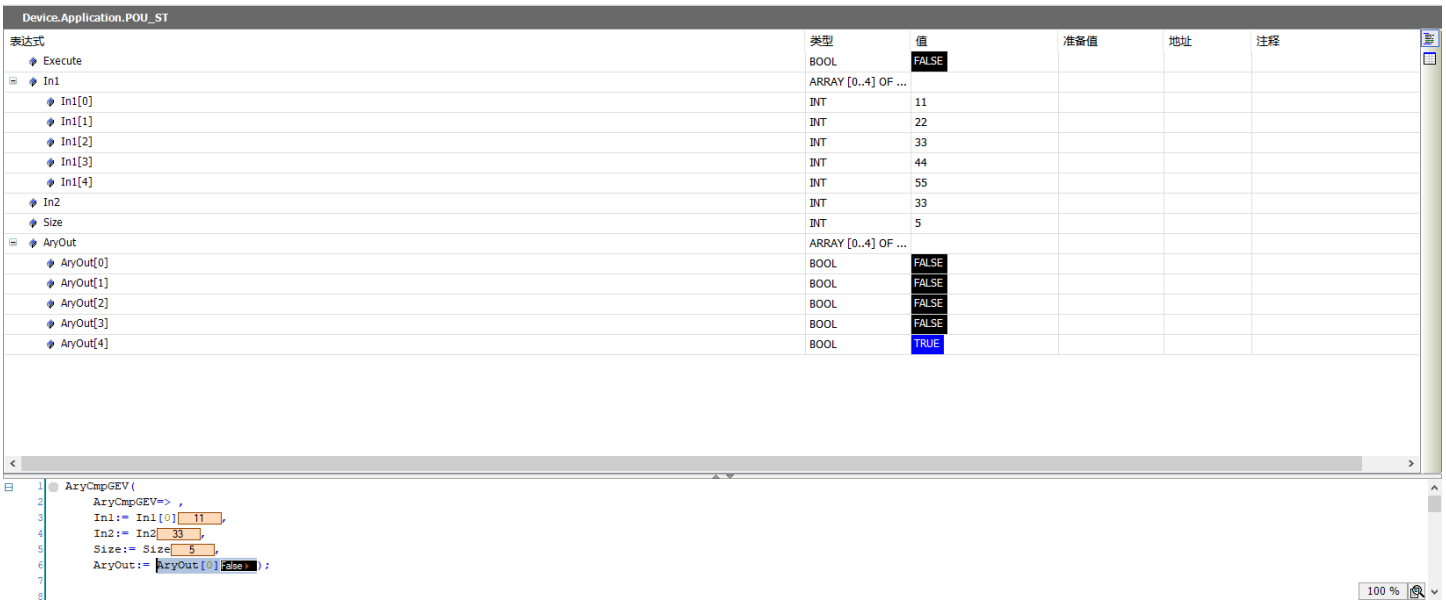
功能说明

对比较数组 In1[0] ～ In1[“Size”-1] 和比较变量“In2”进行大小比较（大于等于）。比较结果保存到与比较结果排列 AryOut[0] ～ AryOut[“Size”-1] 对应的要素编号中。

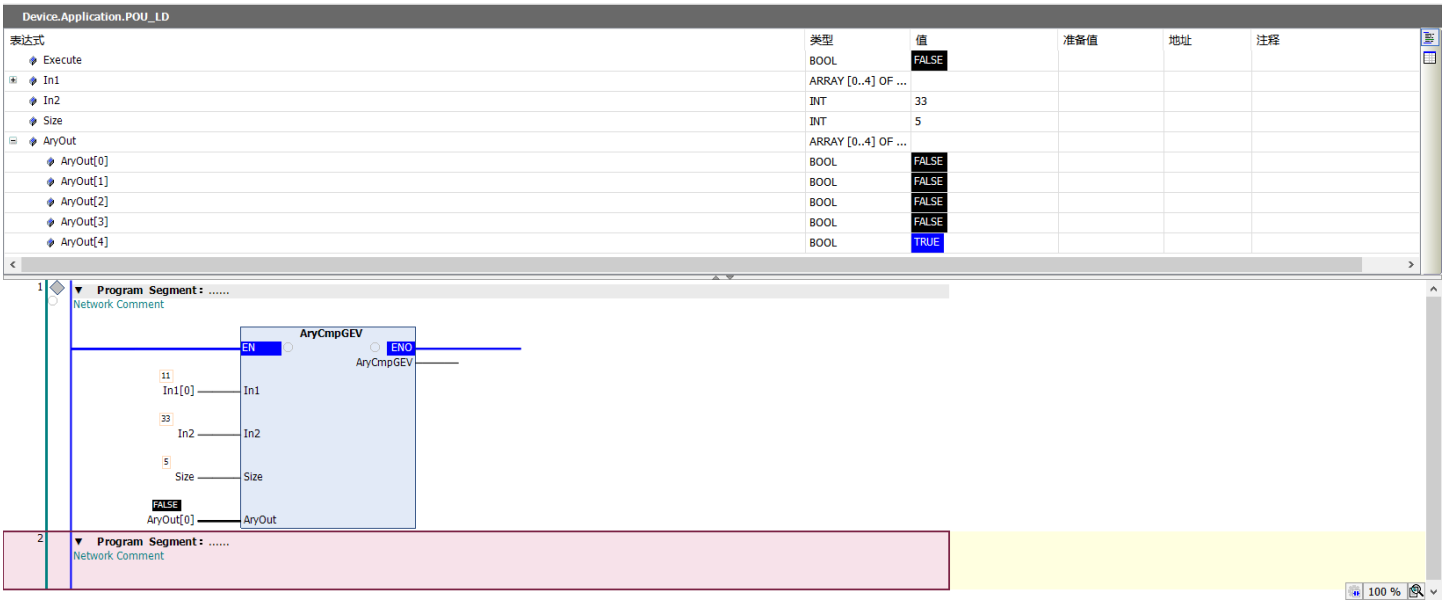
其中 In1[i] >= In2 时，AryOut[i] 为 TRUE，否则为 FALSE。

③程序示例

ST:



LD:



※注意事项

In1[] 和 In2 的数据类型应统一，否则编译时将会发生异常。

In1[]、In2 为实数、且包括除不尽的除法运算结果时，受取整误差的影响，处理结果可能与预计的不同。

“Size”的值为 0 时，函数返回值为 TRUE，AryOut[] 无变化。

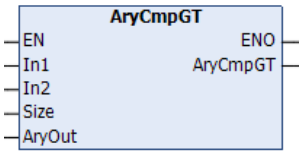
“Size”的值超出 In1[]、AryOut[] 中任一数组区域时，函数返回值为 FALSE，AryOut[] 无变化。

3.4.9 AryCmpGT——数组批量比较GT

对两个数组的各要素进行大小比较（大于）。

①指令格式

指令	功能	FB/FC	LD 表现	ST表现

AryCmpGT	数组批量比较GT	FC		AryCmpGT(AryCmpGT=> , In1:= , In2:= , Size:= , AryOut:=);

②相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
In1[]	对象数组1	-	遵照数据类型	-	对象数组1
In2[]	对象数组2	-	遵照数据类型	-	对象数组2
Size	比较元素数	UINT	遵照数据类型	1	比较元素数
AryOut:[]	比较结果	UINT	遵照数据类型	1	比较结果

数据类型

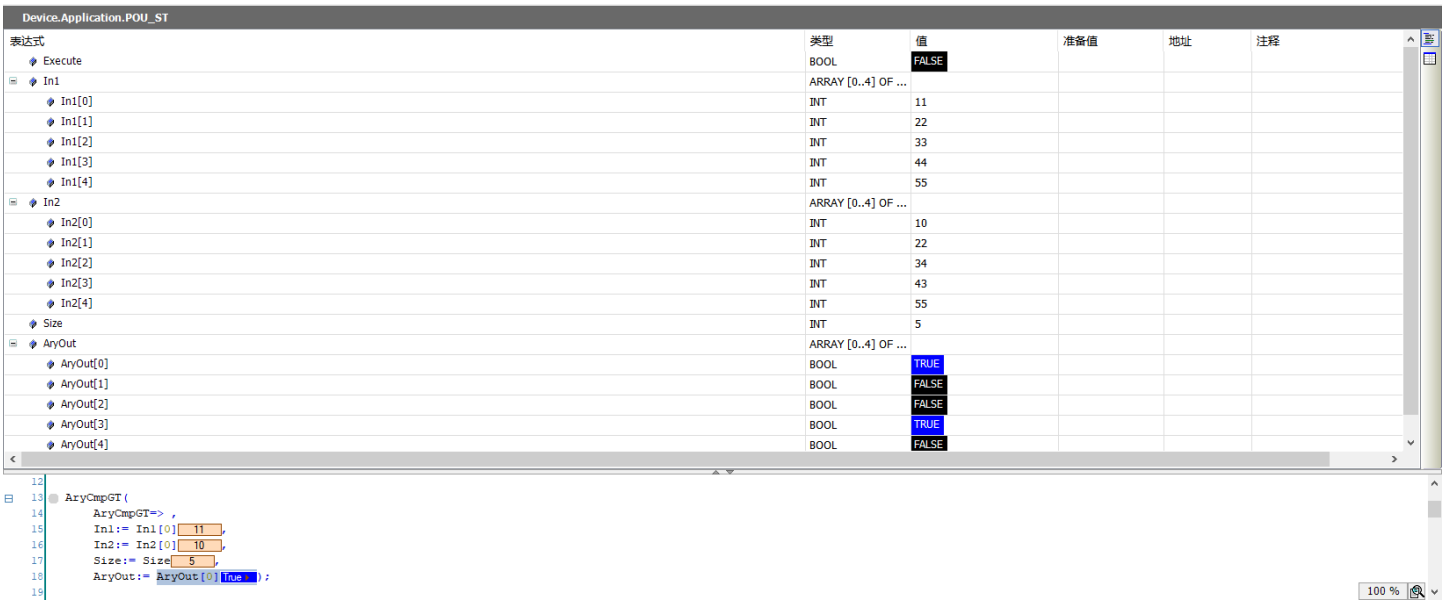
	布尔	位串				整数								实数		时刻、持续时间、日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In1[]	-	-	-	-	-	√	√	√	√	√	√	√	√	√	√	-	-	-	-	-
In2[]	-	-	-	-	-	√	√	√	√	√	√	√	√	√	√	-	-	-	-	-
Size	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
AryOut:[]	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

功能说明

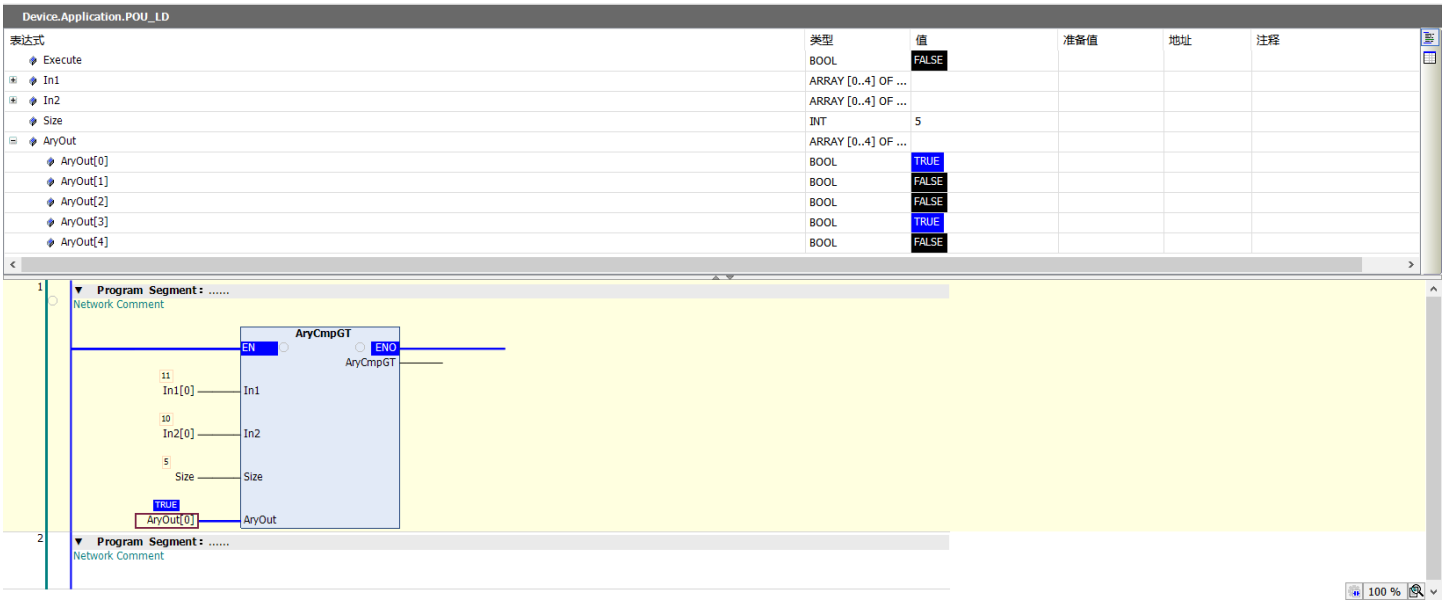
对两个数组 In1[0] ～ In1[“Size”-1]、In2[0] ～ In2[“Size”-1] 的相同要素编号进行大小比较（大于）。比较结果保存到与比较结果排列 AryOut[0] ～ AryOut[“Size”-1] 对应的要素编号中。

其中 In1[i] > In2[i] 时，AryOut[i] 为 TRUE，否则为 FALSE。③程序示例

ST:



LD:



※注意事项

In1[] 和 In2[] 的数据类型应统一，否则编译时将会发生异常。

In1[]、In2[] 为实数、且包括除不尽的除法运算结果时，受取整误差的影响，处理结果可能与预计的不同。

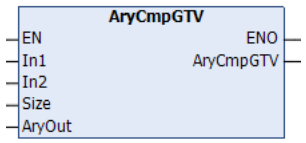
“Size”的值为 0 时，函数返回值为 TRUE，AryOut[] 无变化。

“Size”的值超出 In1[]、In2[]、AryOut[] 中任一数组区域时，函数返回值为 FALSE，AryOut[] 无变化。

3.4.10 AryCmpGTV——数组批量比较GTV

对数组的各要素和变量进行大小比较（大于）。

①指令格式

指令	功能	FB/FC	LD 表现	ST表现
AryCmpGTV	数组批量比较GTV	FC		AryCmpGTV(AryCmpGTV=> , In1:= , In2:= , Size:= , AryOut:=);

②相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
In1[]	对象数组1	-	遵照数据类型	-	对象数组1
In2	对象2	-	遵照数据类型	-	对象数组2
Size	比较元素数	UINT	遵照数据类型	-	比较元素数
AryOut:[]	比较结果	UINT	遵照数据类型	-	比较结果

数据类型

	布尔	位串				整数								实数		时刻、持续时间、日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In1[]	-	-	-	-	-	√	√	√	√	√	√	√	√	√	√	-	-	-	-	-
In2	-	-	-	-	-	√	√	√	√	√	√	√	√	√	√	-	-	-	-	-
Size	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
AryOut:[]	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

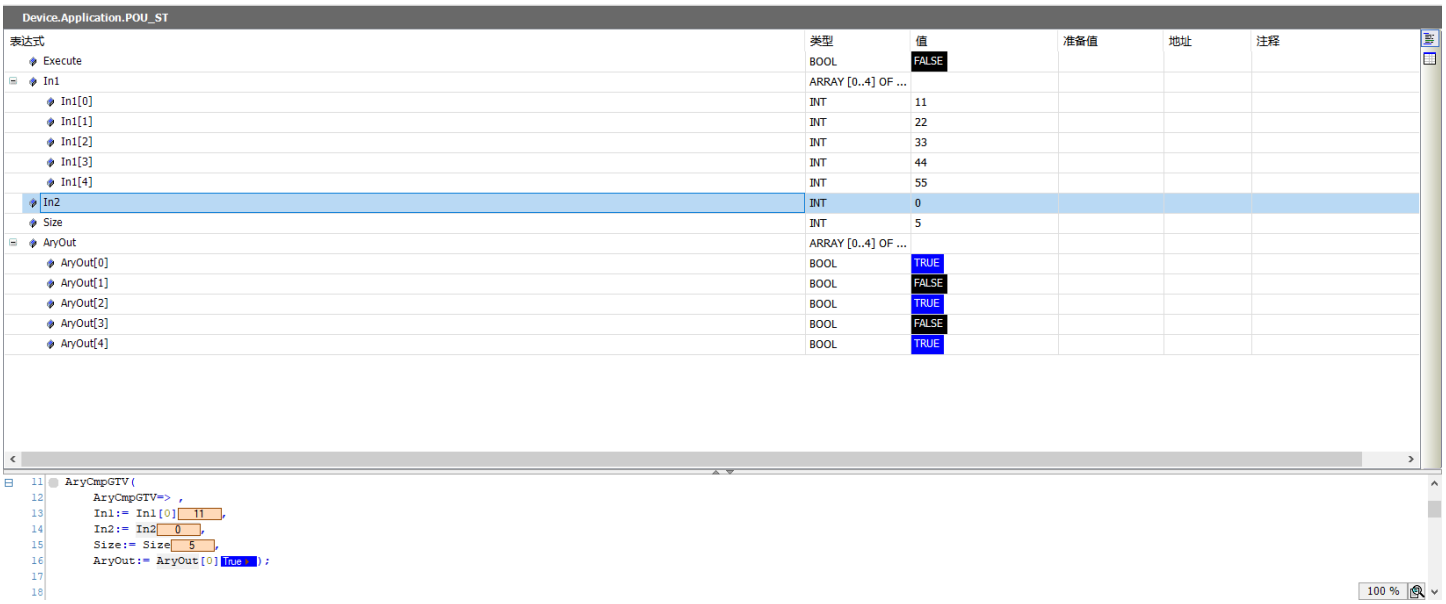
功能说明

对比较数组 In1[0] ～ In1[“Size”-1] 和比较变量“In2”进行大小比较（大于）。比较结果保存到与比较结果排列 AryOut[0] ～ AryOut[“Size”-1] 对应的要素编号中。

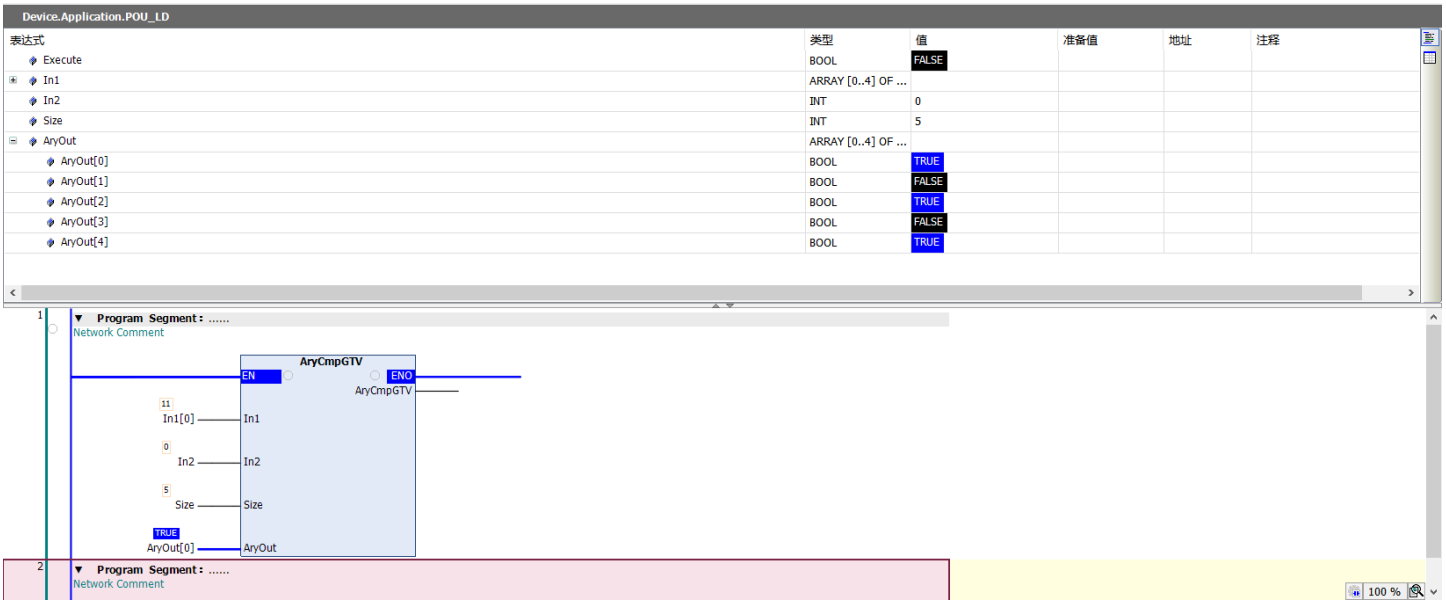
其中 In1[i] > In2 时，AryOut[i] 为 TRUE，否则为 FALSE。

③程序示例

ST:



LD:



※注意事项

In1[] 和 In2 的数据类型应统一，否则编译时将会发生异常。

In1[]、In2 为实数、且包括除不尽的除法运算结果时，受取整误差的影响，处理结果可能与预计的不同。

“Size”的值为 0 时，函数返回值为 TRUE，AryOut[] 无变化。

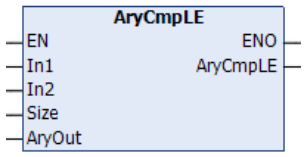
“Size”的值超出 In1[]、AryOut[] 中任一数组区域时，函数返回值为 FALSE，AryOut[] 无变化。

3.4.11 AryCmpLE——数组批量比较LE

对两个数组的各要素进行大小比较（小于等于）。

①指令格式

指令	功能	FB/FC	LD 表现	ST表现

AryCmpLE	数组批量比较LE	FC		AryCmpLE(AryCmpLE=> , In1:= , In2:= , Size:= , AryOut:=);

②相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
In1[]	对象数组1	-	遵照数据类型	-	对象数组1
In2[]	对象数组2	-	遵照数据类型	-	对象数组2
Size	比较元素数	UINT	遵照数据类型	-	比较元素数
AryOut:[]	比较结果	UINT	遵照数据类型	-	比较结果

数据类型

	布尔	位串				整数								实数		时刻、持续时间、日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In1[]	-	-	-	-	-	√	√	√	√	√	√	√	√	√	√	-	-	-	-	-
In2[]	-	-	-	-	-	√	√	√	√	√	√	√	√	√	√	-	-	-	-	-
Size	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
AryOut:[]	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

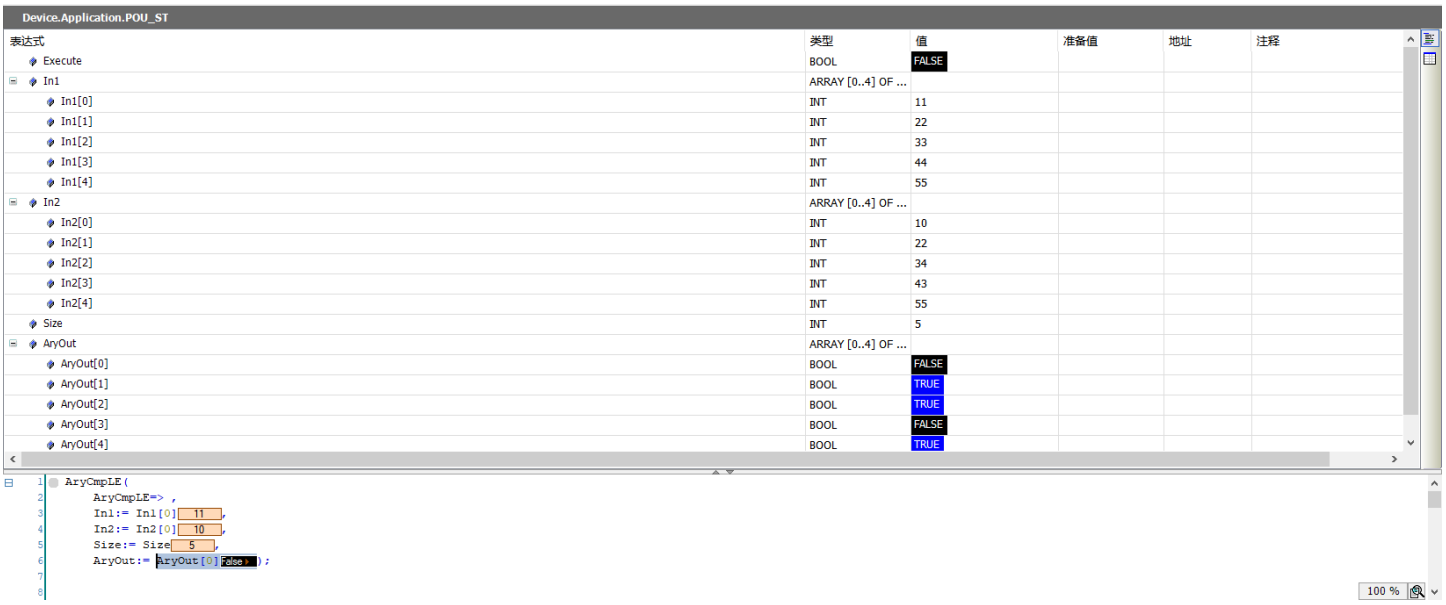
功能说明

对比较数组 In1[0] ～ In1[“Size”-1] 和比较变量“In2”进行比较是否不同。比较结果保存到与比较结果排列 AryOut[0] ～ AryOut[“Size”-1] 对应的要素编号中。

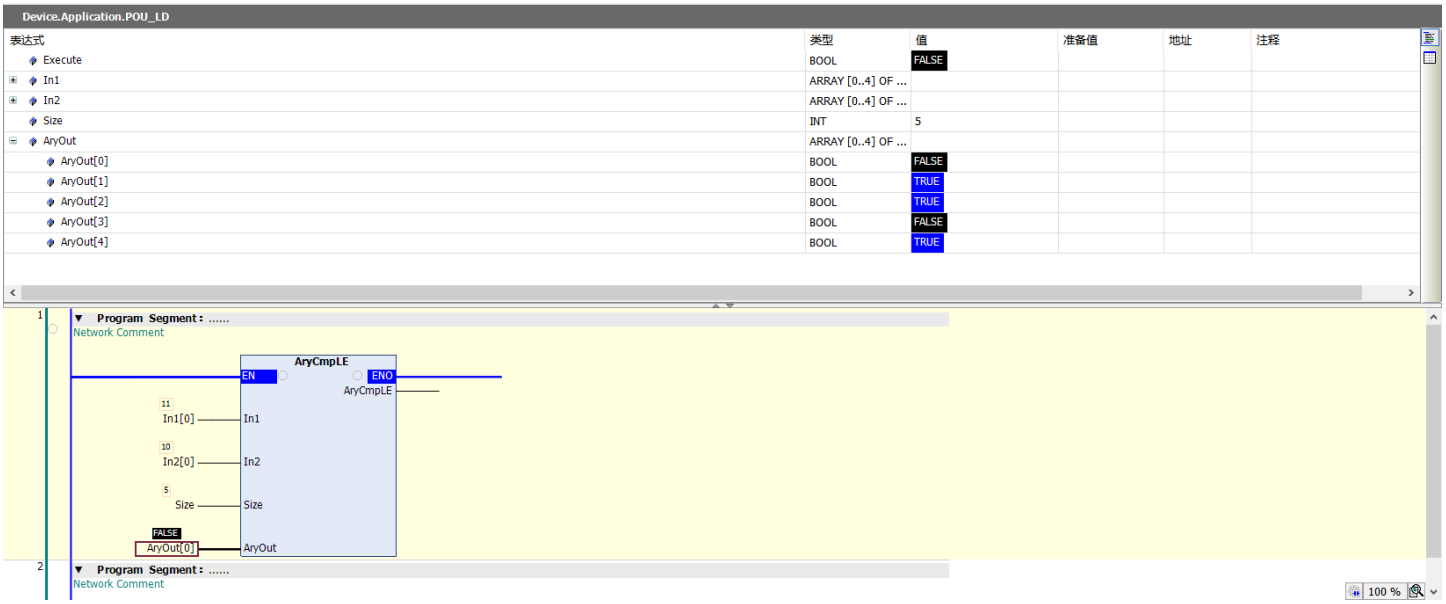
其中 In1[i] <> In2 时，AryOut[i] 为 TRUE，否则为 FALSE。

③程序示例

ST:



LD:



※注意事项

•In1[] 和 In2[] 的数据类型应统一，否则编译时将会发生异常。

•In1[]、In2[] 为实数、且包括除不尽的除法运算结果时，受取整误差的影响，处理结果可能与预计的不同。

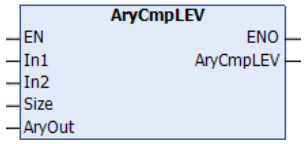
•“Size”的值为 0 时，函数返回值为 TRUE，AryOut[] 无变化。

•“Size”的值超出 In1[]、In2[]、AryOut[] 中任一数组区域时，函数返回值为 FALSE，AryOut[] 无变化。

3.4.12 AryCmpLEV——数组批量比较LEV

对数组的各要素和变量进行大小比较（小于等于）。

①指令格式

指令	功能	FB/FC	LD 表现	ST表现
AryCmpLEV	数组批量比较LEV	FC		AryCmpLEV(AryCmpLEV=> , In1:= , In2:= , Size:= , AryOut:=);

②相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
In1[]	对象数组1	-	遵照数据类型	-	对象数组1
In2	对象2	-	遵照数据类型	-	对象数组2
Size	比较元素数	UINT	遵照数据类型	-	比较元素数
AryOut:[]	比较结果	UINT	遵照数据类型	-	比较结果

数据类型

	布尔	位串				整数								实数		时刻、持续时间、日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In1[]	-	-	-	-	-	√	√	√	√	√	√	√	√	√	√	-	-	-	-	-
In2	-	-	-	-	-	√	√	√	√	√	√	√	√	√	√	-	-	-	-	-
Size	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
AryOut:[]	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

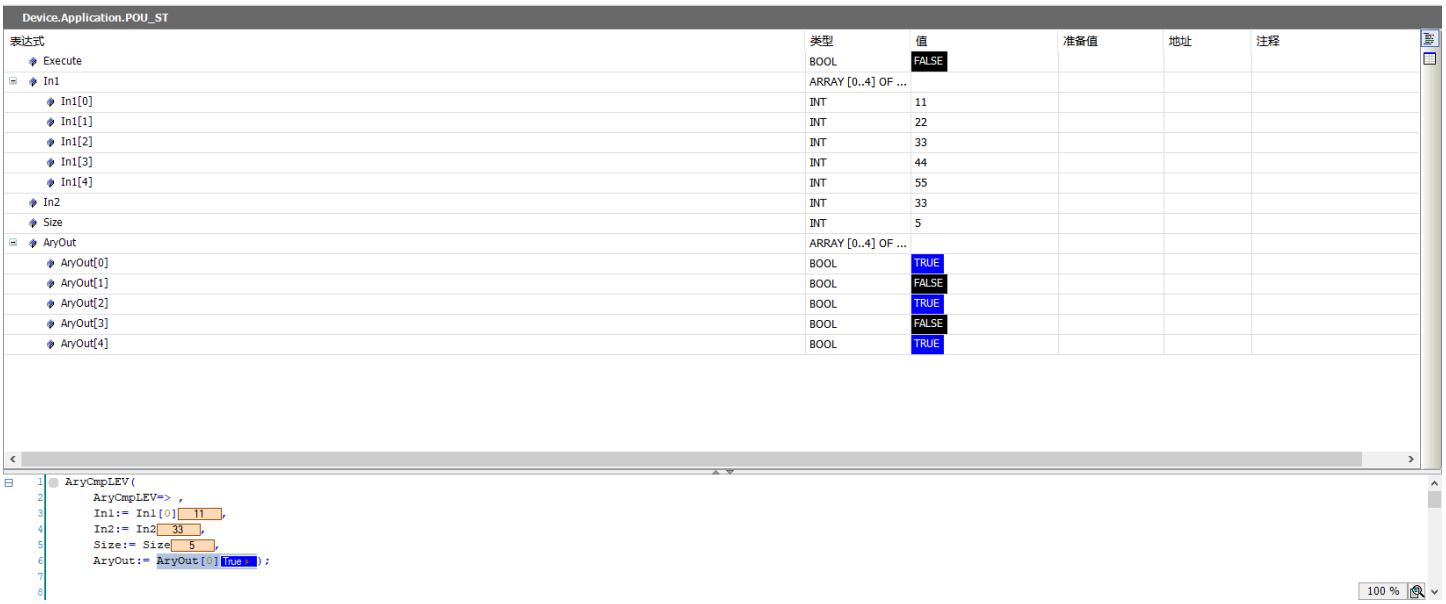
功能说明

对比较数组 In1[0] ～ In1[“Size”-1] 和比较变量“In2”进行大小比较（小于等于）。比较结果保存到与比较结果排列 AryOut[0] ～ AryOut[“Size”-1] 对应的要素编号中。

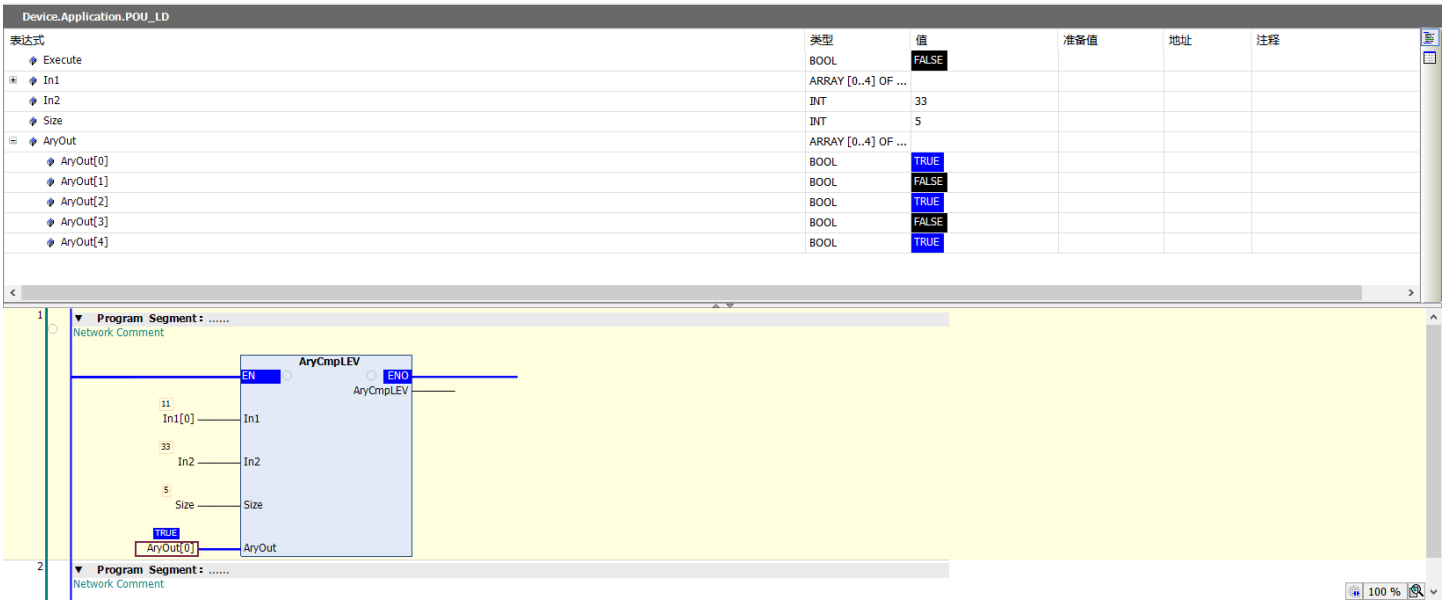
其中 In1[i] <= In2 时，AryOut[i] 为 TRUE，否则为 FALSE。。

③程序示例

ST:



LD:



※注意事项

•In1[] 和 In2[] 的数据类型应统一，否则编译时将会发生异常。

•In1[]、In2[] 为实数、且包括除不尽的除法运算结果时，受取整误差的影响，处理结果可能与预计的不同。

•“Size”的值为 0 时，函数返回值为 TRUE，AryOut[] 无变化。

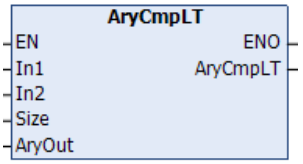
•“Size”的值超出 In1[]、In2[]、AryOut[] 中任一数组区域时，函数返回值为 FALSE，AryOut[] 无变化。

3.4.13 AryCmpLT——数组批量比较LT

对两个数组的各要素进行大小比较（小于）。

①指令格式

指令	功能	FB/FC	LD 表现	ST表现

AryCmpLT	数组批量比较LT	FC		AryCmpLT(AryCmpLT=> , In1:= , In2:= , Size:= , AryOut:=);

②相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
In1[]	对象数组1	-	遵照数据类型	-	对象数组1
In2[]	对象数组2	-	遵照数据类型	-	对象数组2
Size	比较元素数	UINT	遵照数据类型	-	比较元素数
AryOut:[]	比较结果	UINT	遵照数据类型	-	比较结果

数据类型

	布尔	位串				整数								实数		时刻、持续时间、日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In1[]	-	-	-	-	-	√	√	√	√	√	√	√	√	√	√	-	-	-	-	-
In2[]	-	-	-	-	-	√	√	√	√	√	√	√	√	√	√	-	-	-	-	-
Size	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
AryOut:[]	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

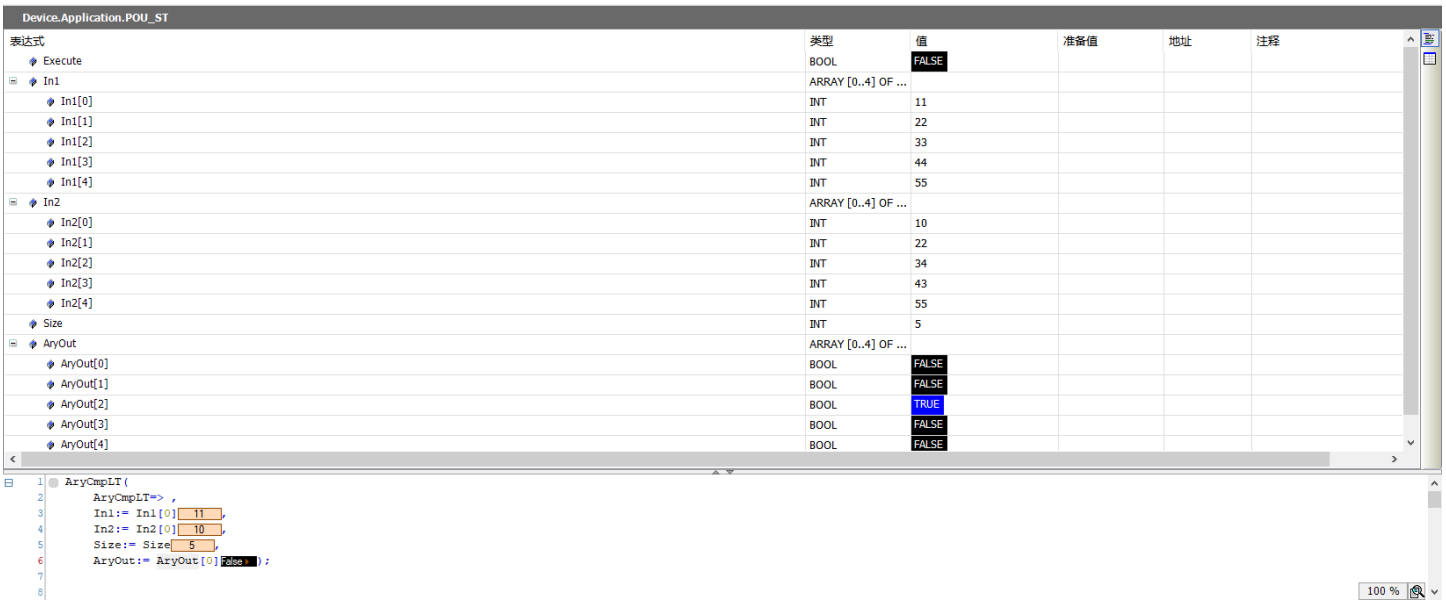
功能说明

对两个数组 In1[0] ～ In1[“Size”-1]、In2[0] ～ In2[“Size”-1] 的相同要素编号进行大小比较（小于）。比较结果保存到与比较结果排列 AryOut[0] ～ AryOut[“Size”-1] 对应的要素编号中。

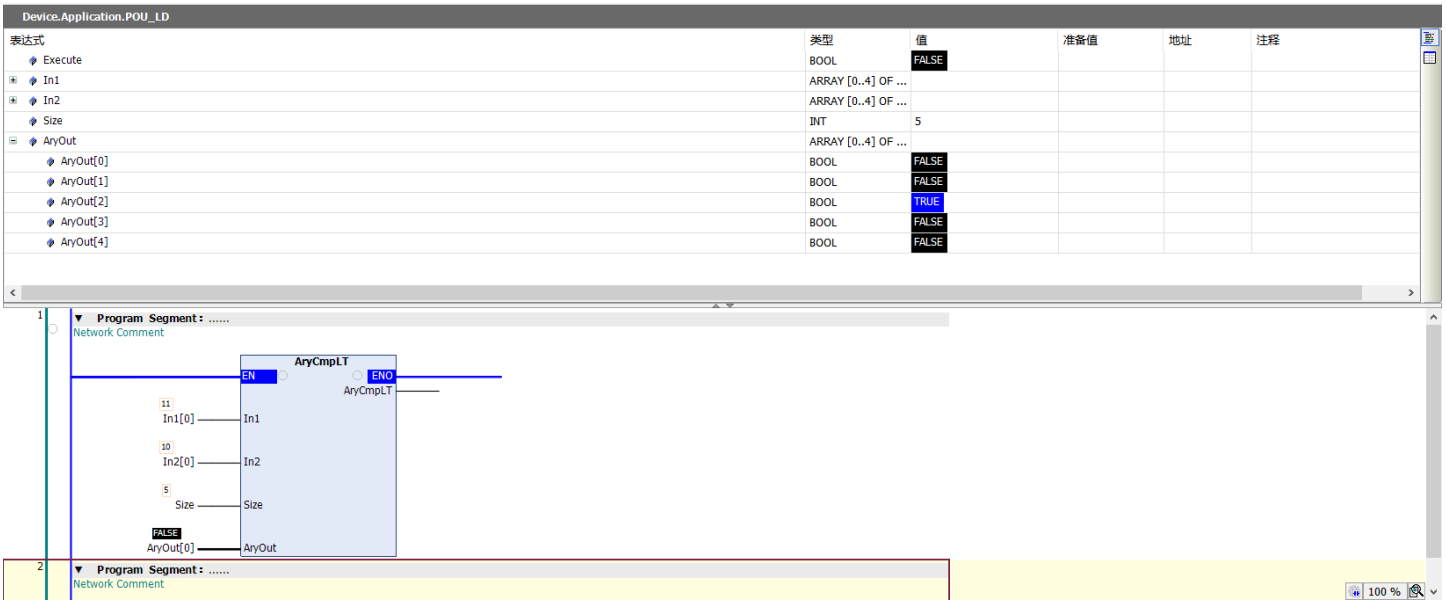
其中 In1[i] < In2[i] 时，AryOut[i] 为 TRUE，否则为 FALSE。

③程序示例

ST:



LD:



※注意事项

•In1[] 和 In2[] 的数据类型应统一，否则编译时将会发生异常。

•In1[]、In2[] 为实数、且包括除不尽的除法运算结果时，受取整误差的影响，处理结果可能与预计的不同。

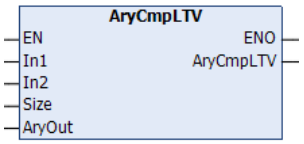
•“Size”的值为 0 时，函数返回值为 TRUE，AryOut[] 无变化。

•“Size”的值超出 In1[]、In2[]、AryOut[] 中任一数组区域时，函数返回值为 FALSE，AryOut[] 无变化。

3.4.14 AryCmpLTV——数组批量比较LTV

对数组的各要素和变量进行大小比较（小于）。

①指令格式

指令	功能	FB/FC	LD 表现	ST表现
AryCmpLTV	数组批量比较LTV	FC		AryCmpLTV(AryCmpLTV=> , In1:= , In2:= , Size:= , AryOut:=);

②相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
In1[]	对象数组1	-	遵照数据类型	-	对象数组1
In2	对象2	-	遵照数据类型	-	对象数组2
Size	比较元素数	UINT	遵照数据类型	-	比较元素数
AryOut:[]	比较结果	UINT	遵照数据类型	-	比较结果

数据类型

	布尔	位串				整数								实数		时刻、持续时间、日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In1[]	-	-	-	-	-	√	√	√	√	√	√	√	√	√	√	-	-	-	-	-
In2	-	-	-	-	-	√	√	√	√	√	√	√	√	√	√	-	-	-	-	-
Size	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
AryOut:[]	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

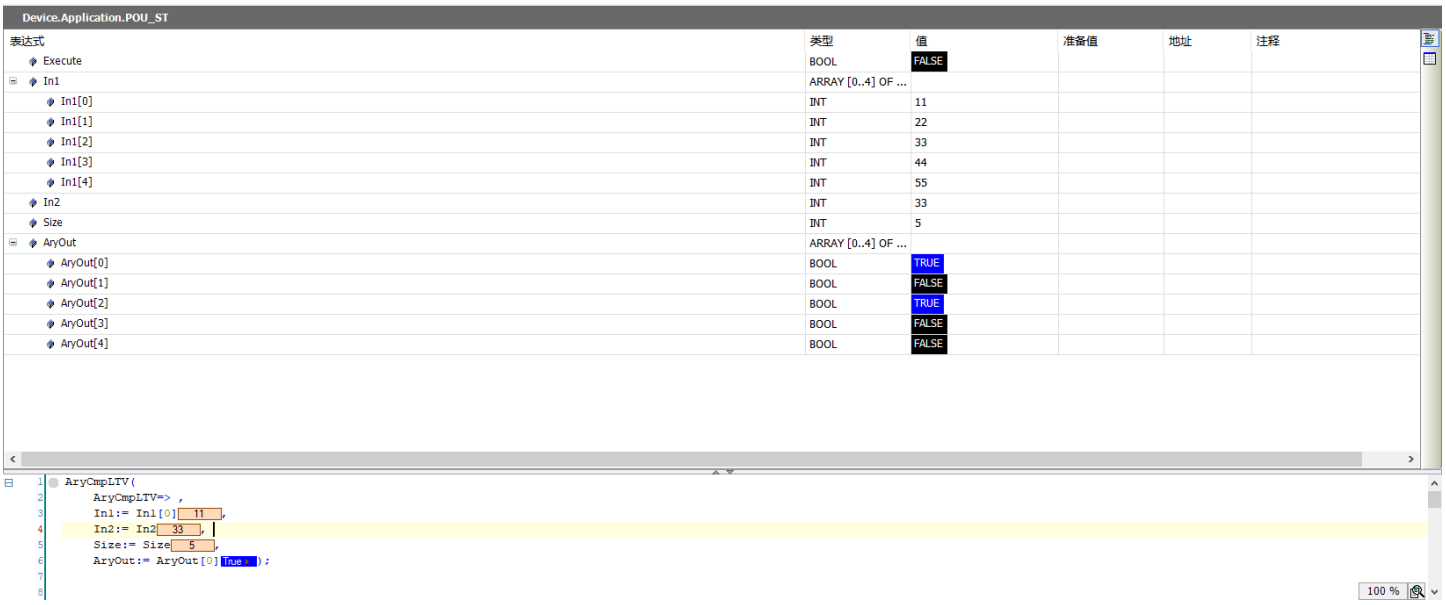
功能说明

对比较数组 In1[0] ～ In1[“Size”-1] 和比较变量“In2”进行大小比较（小于）。比较结果保存到与比较结果排列 AryOut[0] ～ AryOut[“Size”-1] 对应的要素编号中。

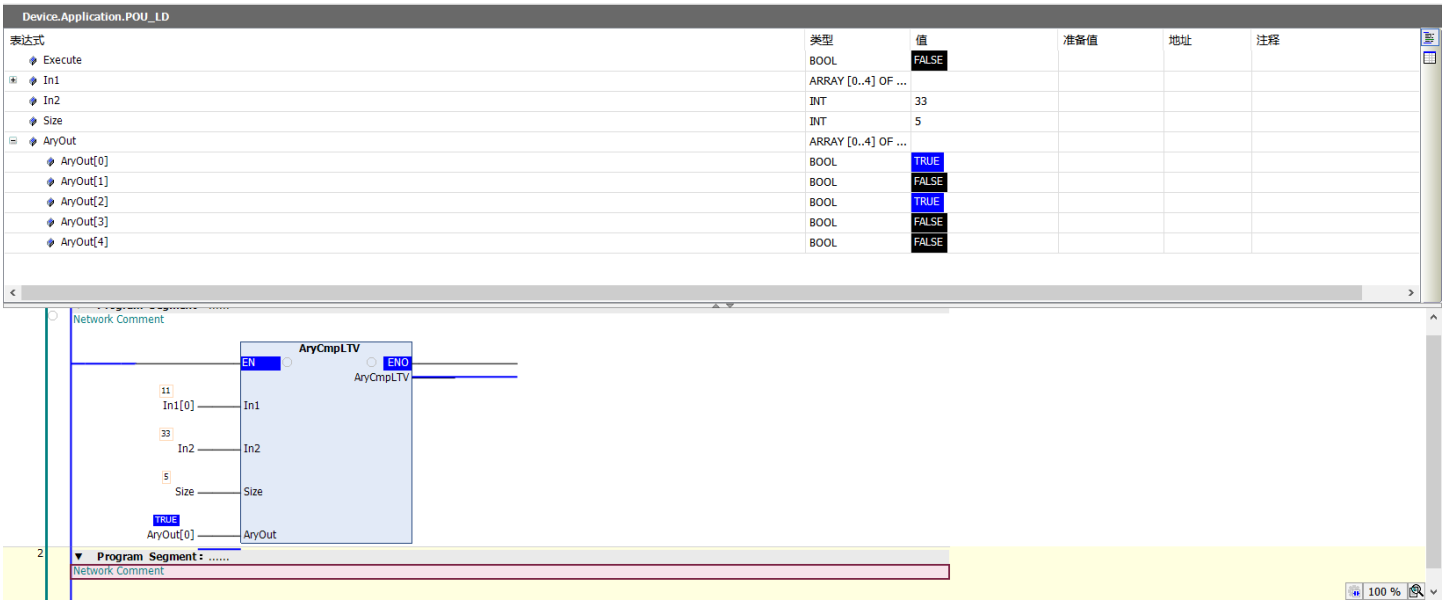
其中 In1[i] < In2 时，AryOut[i] 为 TRUE，否则为 FALSE。

③程序示例

ST:



LD:



※注意事项

•In1[] 和 In2[] 的数据类型应统一，否则编译时将会发生异常。

•In1[]、In2[] 为实数、且包括除不尽的除法运算结果时，受取整误差的影响，处理结果可能与预计的不同。

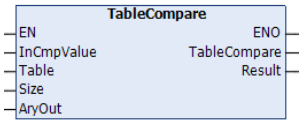
•“Size”的值为 0 时，函数返回值为 TRUE，AryOut[] 无变化。

•“Size”的值超出 In1[]、In2[]、AryOut[] 中任一数组区域时，函数返回值为 FALSE，AryOut[] 无变化。

3.4.15 TableCompare——表比较

对比较数据和比较表格中指定的多个定义区间进行比较。

①指令格式

指令	功能	FB/FC	LD 表现	ST表现
TableCompare	表比较	FC		TableCompare(TableCompare=> , InCmpValue:= , Table:= , Size:= , AryOut:= , Result=>);

②相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
InCmpValue	对象数组	-	遵照数据类型	-	对象数组
Table	对象表	-	遵照数据类型	-	对象表
Size	比较元素数	UINT	遵照数据类型	-	比较元素数
AryOut:[]	比较结果	BOOL	遵照数据类型	-	比较结果

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Result	执行完成	BOOL	TRUE/FALSE	-	执行完成

数据类型

	布尔	位串				整数								实数		时刻、持续时间、日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
TableCompare[]	-	-	-	-	-	√	√	√	√	√	√	√	√	√	√	-	-	-	-	-
In2[]	-	-	-	-	-	√	√	√	√	√	√	√	√	√	√	-	-	-	-	-
Size	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
AryOut:[]	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

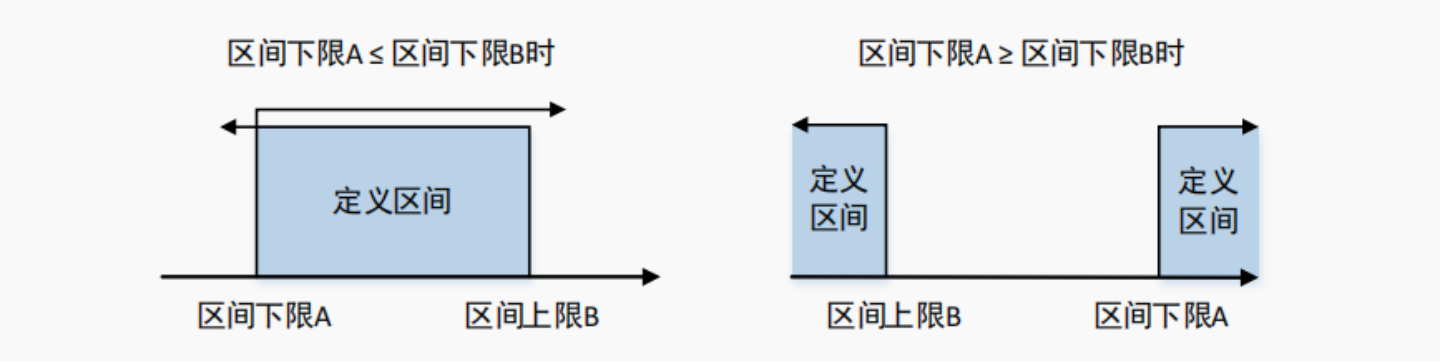
功能说明

对比较数据“In”和比较表格 Table[] 中指定的“Size”组的定义区间进行比较。

Table[]为 2 维数组，1 维元素是定义区间的编号，2 维第 0 个要素表示定义区间的区间下限值 A，第 1 个要。素表示定义区间的区间上限值B。

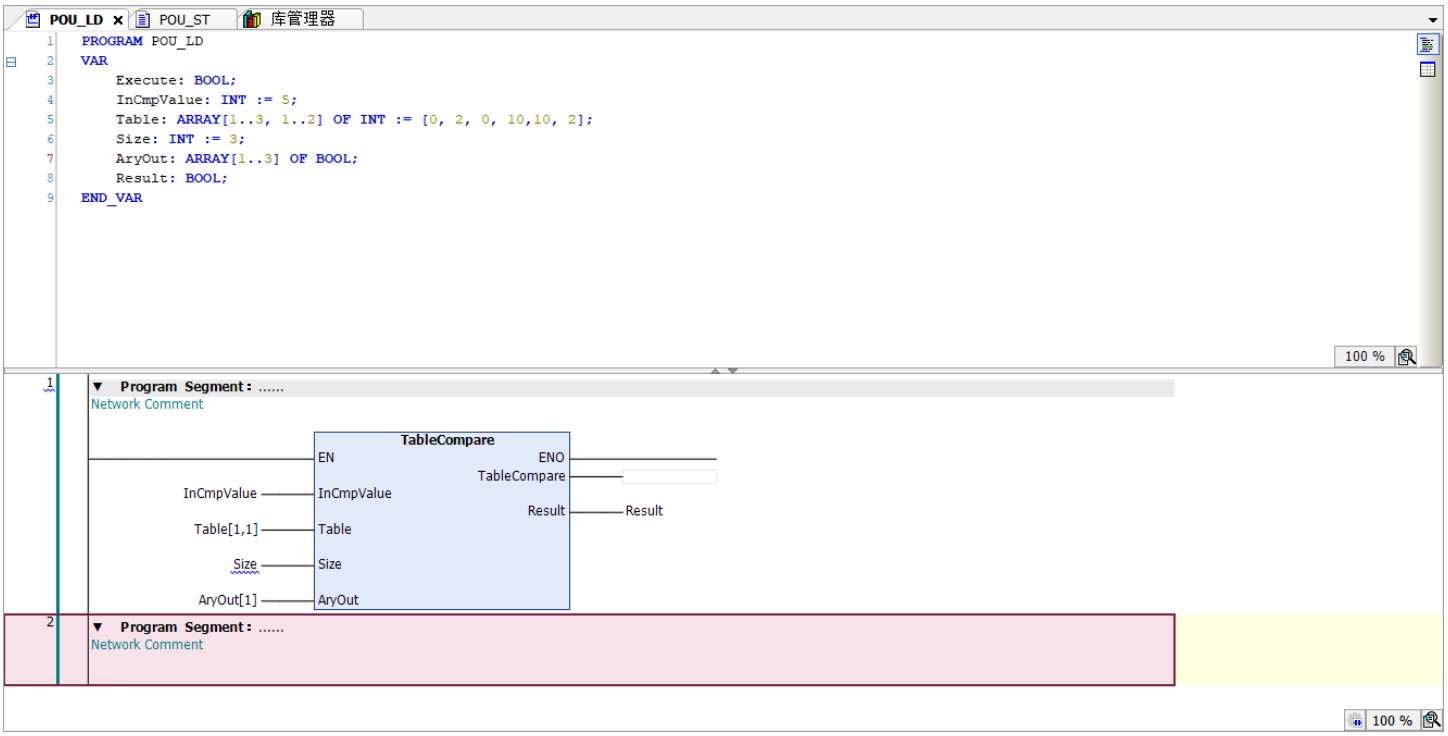
定义区间	区间下限值 A	区间上限值 B
区间 0	Table[0,0]	Table[0,1]
区间1	Table[1,0]	Table[1,1]
…	…	…
区间“Size”-1	Table[“Size”-1,0]	Table[“Size”-1,1]

用区间下限值 A 和区间上限值 B 指定定义区间的方法如下所示，区间下限值 A 及区间上限值 B 的值包含于定义区间内。

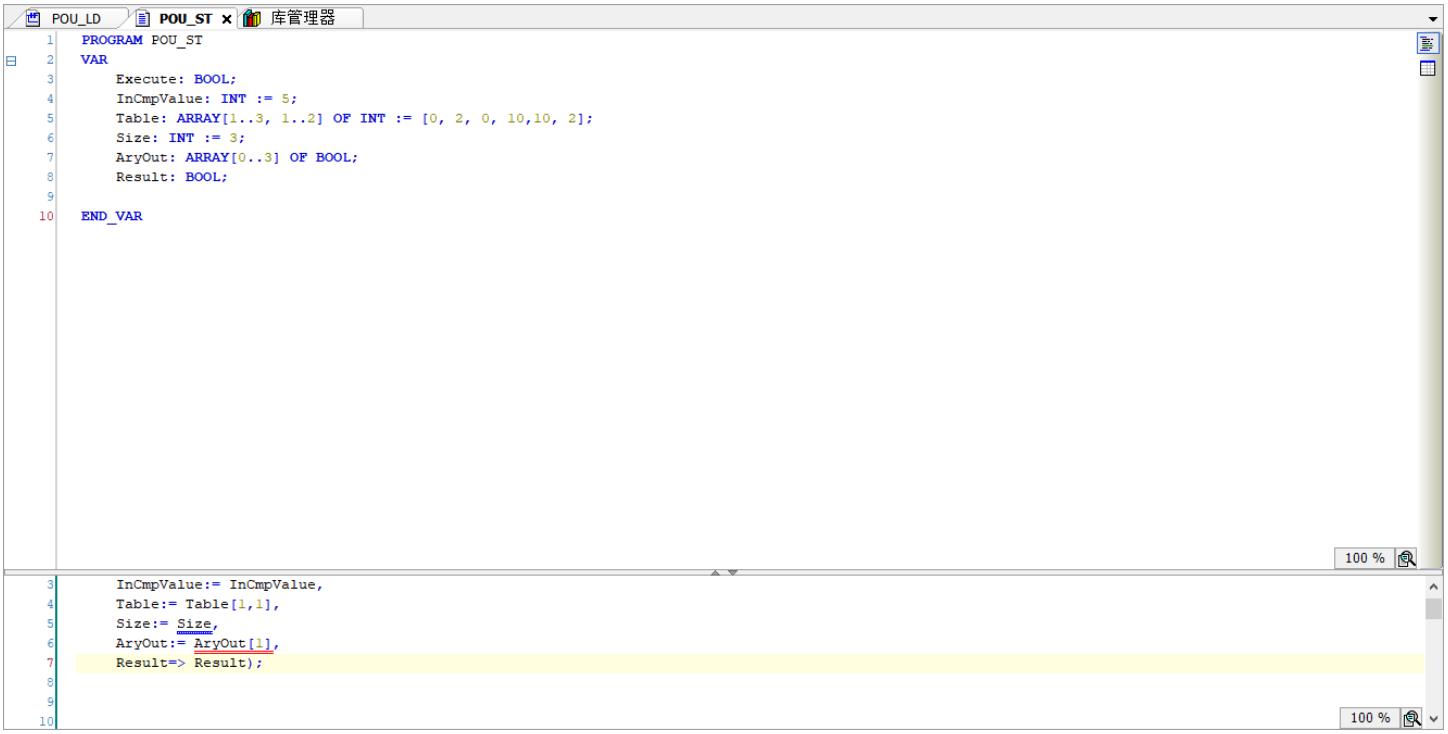


“In”和 Table[] Out[i] 的值为 TRUE 的比较结果保存于个别比较结果数组 ，否则为 FALSE。若 AryOut[] 的“ AryOut[] Size”个要素全部为 TRUE 中。若“In”在第 i ，则比较结果“ 个定义区间内，则 Out”的值 Ary-
为 TRUE，否则为 FALSE。

③程序示例
ST:



LD:



※注意事项

•“In”和 Table[] 要素的数据类型应统一，否则函数返回值为 FALSE。

•Table[] 请务必使用 2 维数组且第 2 维数组的大小为 2，否则输出结果是非预期的。

•AryOut[] 数组的大小大于“Size”时，比较结果将保存到 AryOut[0] ～ AryOut[“Size”-1] 中。其他数组 要素无变化。

• 比较对象为实数、且包括除不尽的除法运算结果时，受取整误差的影响，处理结果可能与预计的不同。

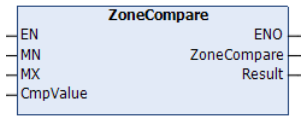
•“Size” 的值为 0 时，“Out” 的值为 FALSE，AryOut[] 无变化。

•“Size” 的值超出 AryOut[] 数组的大小，或者超出 Table[] 的第 1 维数组的大小时，可能导致程序执行异常， 甚至 PLC 死机。

3.4.16 ZoneCompare——区域比较

判断比较数据是否在指定的上限和下限值之间。

①指令格式

指令	功能	FB/FC	LD 表现	ST表现
ZoneCompare	区域比较	FC		ZoneCompare(ZoneCompare=> , MN:= , MX:= , CmpValue:= , Result=>);

②相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
MN	对象数组	-	遵照数据类型	-	对象数组
MX	比较数据	-	遵照数据类型	-	比较数据
CmpValue	比较元素数	-	遵照数据类型	-	比较元素数

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Result	比较结果	BOOL	TRUE/FALSE	-	比较结果

数据类型

	布尔	位串				整数								实数		时刻、持续时间、日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
MN	-	-	-	-	-	√	√	√	√	√	√	√	√	√	√	√	√	√	√	-
MX	-	-	-	-	-	√	√	√	√	√	√	√	√	√	√	√	√	√	√	-

CmpValue	-	-	-	-	-	√	√	√	√	√	√	√	√	√	√	√	√	√	√	-
Result	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

功能说明

判断比较数据“In”是否在上限值“MX”和下限值“MN”之间。

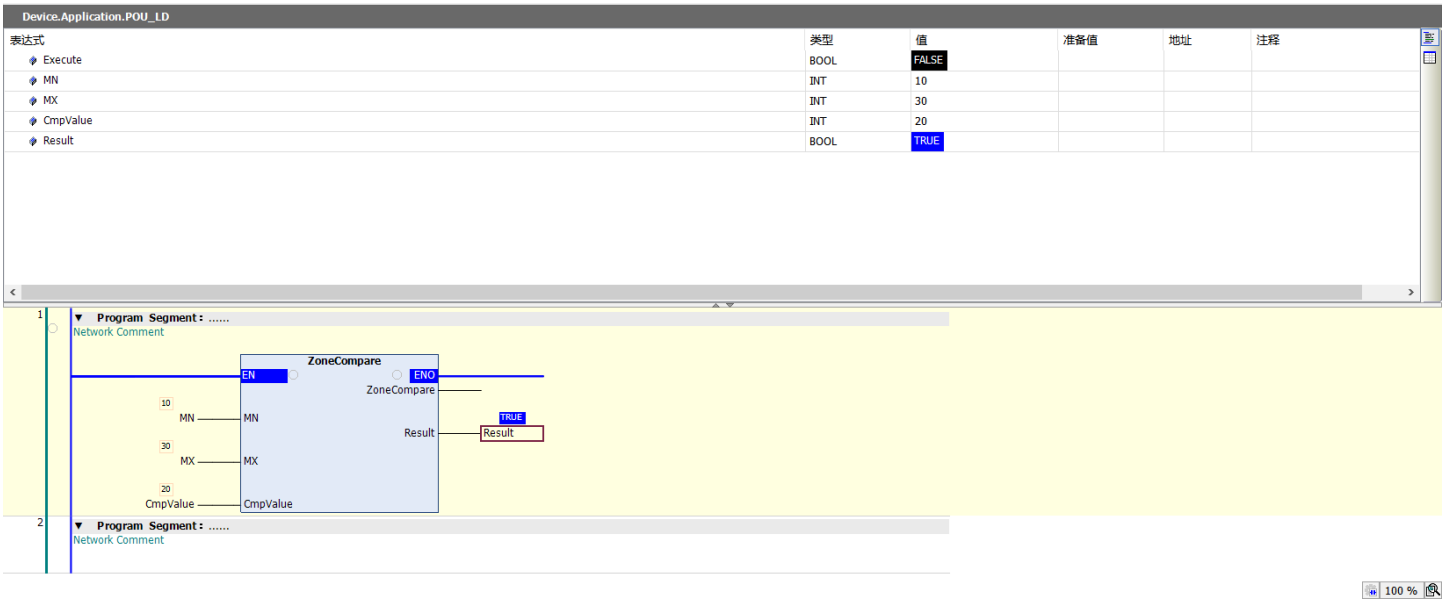
“MX”≥“In”≥“MN”时，“Out”的值为 TRUE，其他位 FALSE。

整数、实数类型以外值的大小关系判断如下

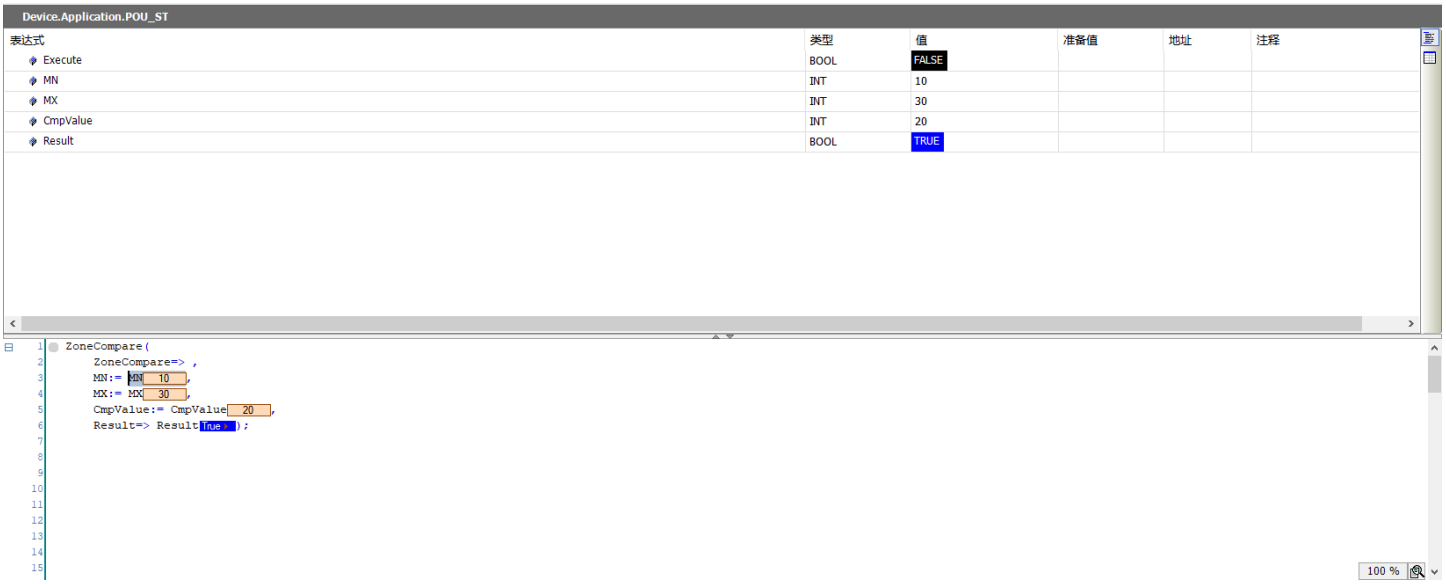
数据类型	大小关系
TIME	值较大者判断为大
DATA、TOD、DT	日期或时刻较后面者判断为大

③程序示例

ST:“MN”=INT#10、“In”=INT#20、“MX”=INT#3



LD:“MN”=INT#10、“In”=INT#20、“MX”=INT#3



※注意事项

- “In”、“MX”、“MN”的数据类型不同时，编译时将发生异常。
- “In”、“MX”、“MN”为实数，且包括除不尽的除法运算结果时，受取整误差的影响，处理结果可能与预计的不同。
- “In”的值为非数值时，“Out”的值为 FALSE。
- “MN”的值大于“MX”的值时，程序发生异常，函数返回值为 FALSE。
- “MX”、“MN”中任意一个为非数值时，程序发生异常，函数返回值为 FALSE。

3.5 数组移位相关

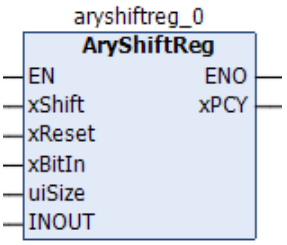
指令列表

指令类别	名称	FB/FC	功能
数组移位相关	AryShiftReg	FB	左移寄存器
	AryShiftRegLR	FB	左右移寄存器
	RCL	FC	带进位的循环左移位指令
	RCR	FC	带进位的循环右移位指令
	SFTL	FC	位数据向左拷贝
	SFTR	FC	位数据向右拷贝
	WSFR	FC	字数据向左拷贝
	WSFR	FC	字数据向右拷贝

3.5.1 AryShiftReg——左移寄存器

将数组要素组成的位列整体向左移位 1 位，在最低位中插入输入值。

①指令格式

指令	功能	FB/FC	LD 表现	ST表现
AryShiftReg	左移寄存器	FB		AryShiftReg_0(xShift:= , xReset:= , xBitIn:= , uiSize:= , INOUT:= , xPCY=>);

②相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
xShift	移位	BOOL	遵照数据类型	-	由 FALSE 变为 TRUE 时，执行移位
xReset:	复位	BOOL	遵照数据类型	-	由 FALSE 变为 TRUE 时，执行移位
xBitIn	输入值	BOOL	遵照数据类型	-	InOut[] 的最低位中插入的值
Size	元素数	UINT	遵照数据类型	-	元素数
INOUT[]	数组	-	遵照数据类型	-	数组

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
xPCY	进位标志	BOOL	遵照数据类型	-	进位标志中保存到的值

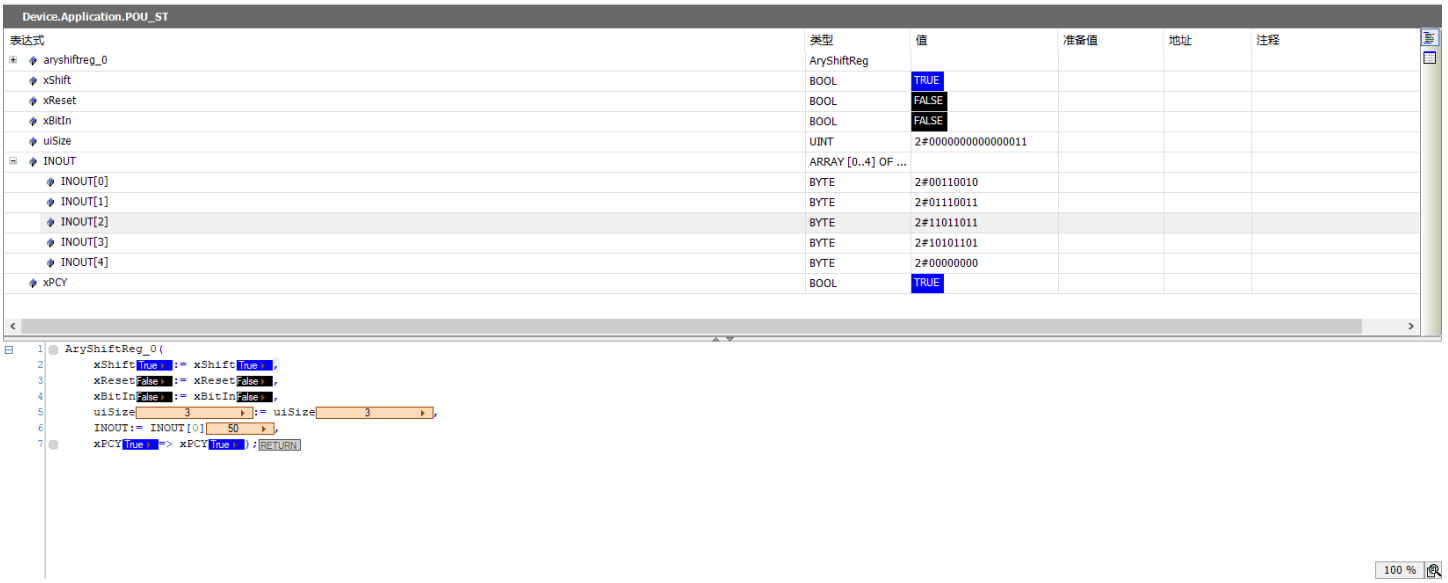
数据类型

	布尔	位串				整数								实数		时刻、持续时间、日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
xShift	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
xReset	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
xBitIn	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

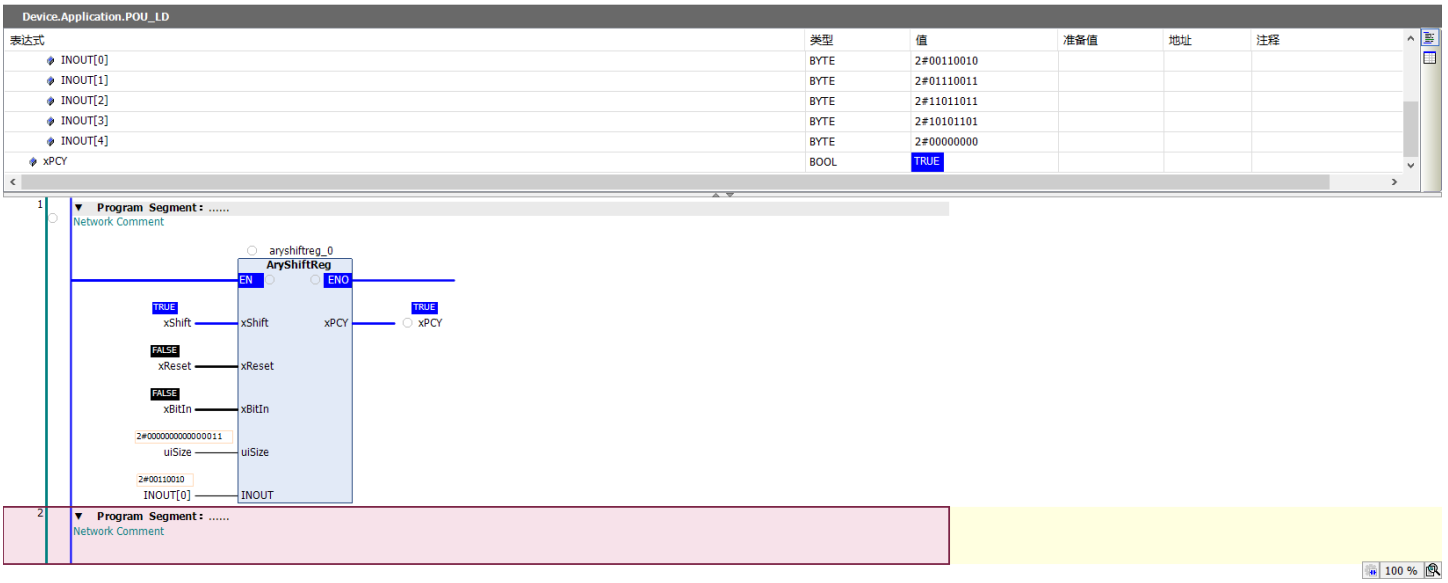
Size	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
INOUT[]	√	√	√	√	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
xPCY	√																			

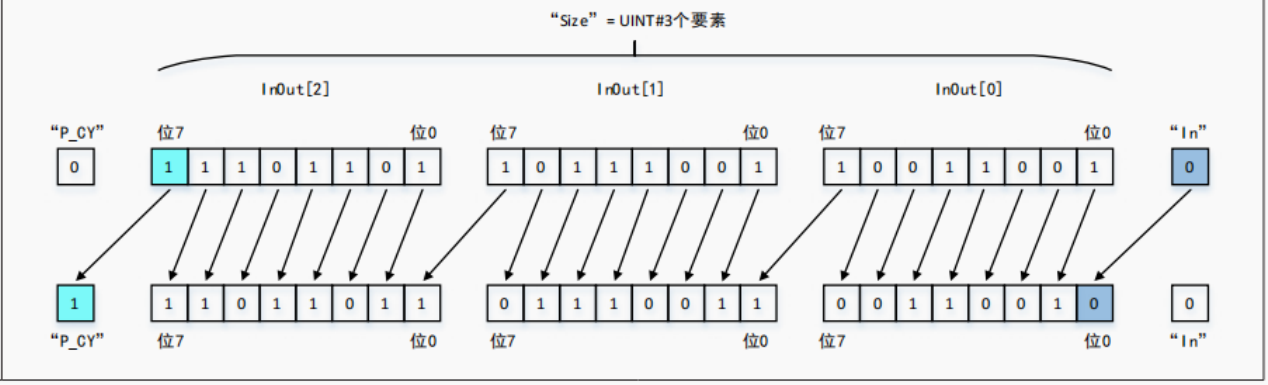
③程序示例

ST：



LD：



原理分析	

※注意事项

- “Reset” 为 TRUE 时，即使 “Shift” 从 FALSE 变为 TRUE，也不执行移位。
- “Size” 的值为 0 时，InOut[] 无变化。
- “Size” 的值超出 InOut[] 的数组区域时，InOut[] 无变化。

3.5.2 AryShiftRegLR——左右移寄存器

将由数组要素组成的位列整体向左（或右）移位 1 位，在最低位（或最高位）插入输入值。

①指令格式

指令	功能	FB/FC	LD 表现	ST表现
AryShiftRegLR	左右移寄存器	FB	aryshiftreglr_0 AryShiftRegLR EN xShiftL xShiftR xReset xBitIn uiSize INOUT ENO xPCY	AryShiftRegLR_0(xShiftL:= , xShiftR:= , xReset:= , xBitIn:= , uiSize:= , INOUT:= , xPCY=>);

②相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
xShiftL	左移位	BOOL	遵照数据类型	-	由 FALSE 变为 TRUE 时，执行左移位
xShiftR	右移位	BOOL	遵照数据类型	-	由 FALSE 变为 TRUE 时，执行右移位
xReset:	复位	BOOL	遵照数据类型	-	由 FALSE 变为 TRUE 时，执行移位
xBitIn	输入值	BOOL	遵照数据类型	-	InOut[] 的最低位中插入的值
Size	元素数	UINT	遵照数据类型	-	元素数
INOUT[]	数组	-	遵照数据类型	-	数组

输出变量

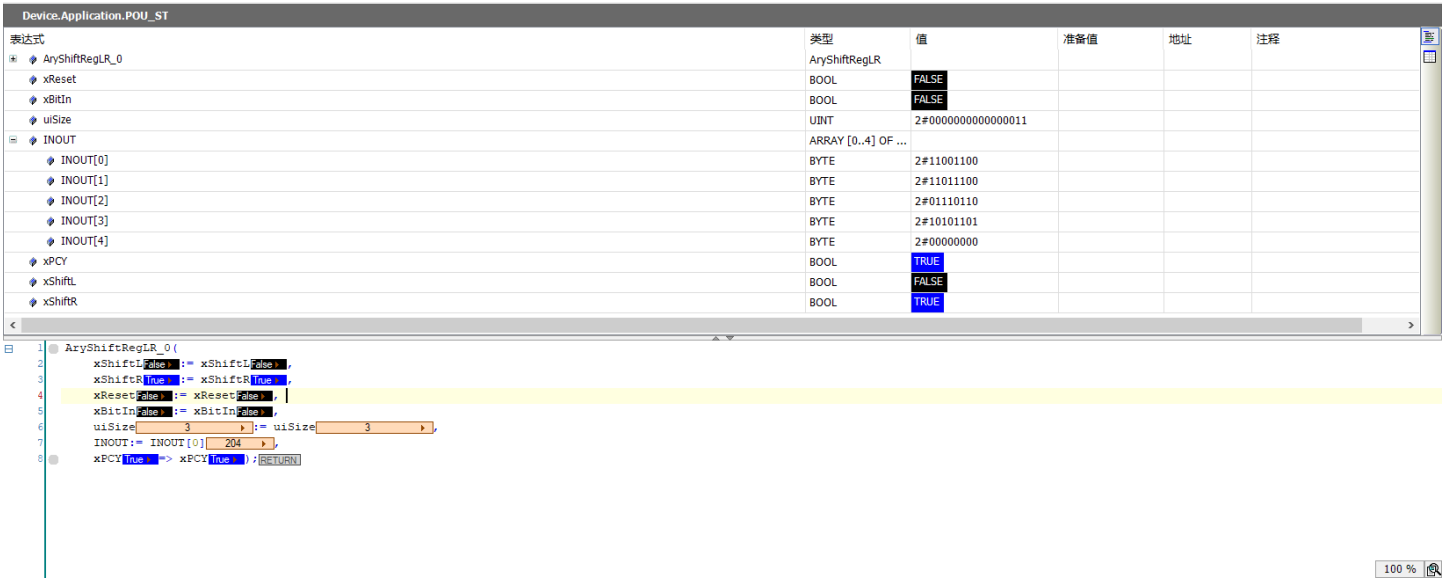
输出变量	名称	数据类型	有效范围	初始值	描述
xPCY	进位标志	BOOL	遵照数据类型	-	进位标志中保存到的值

数据类型

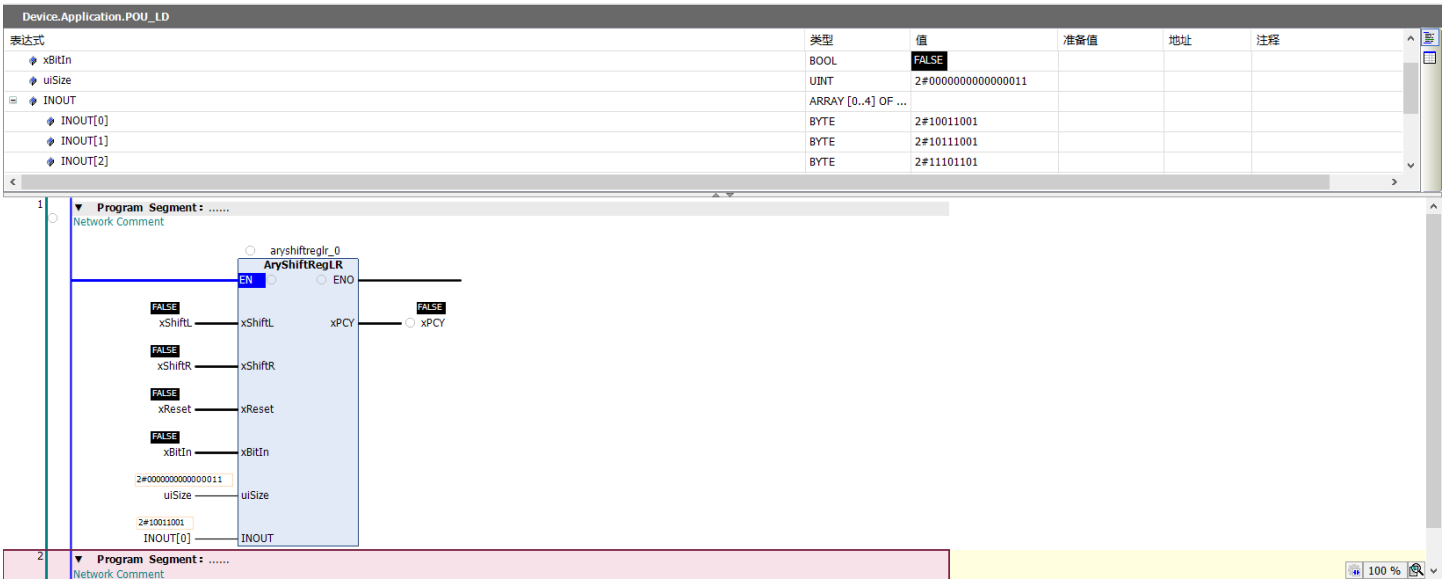
	布尔	位串				整数								实数		时刻、持续时间、日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
xShiftL	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
xShiftR	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
xReset	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
xBitIn	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Size	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
INOUT[]	√	√	√	√	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
xPCY	√																			

③程序示例

ST：



LD：

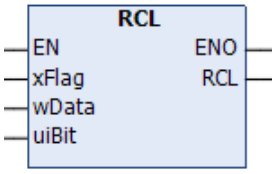
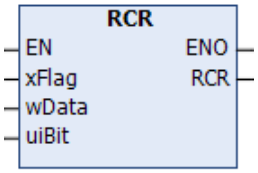


3.5.3 RCL/RCR——带进位的循环左移位指令/带进位的循环右移位指令

RCL：执行后对操作数进行可进位的循环左移，左边移出的位直接补充到右边最低位，进位时，进位标志位会置 on。

RCR：执行后对操作数进行可进位的循环右移，右边移出的位直接补充到左边最高位，进位时，进位标志位会置 on。

①指令格式

指令	功能	FB/FC	LD 表现	ST表现
RCL	带进位的循环左移位指令	FC		RCL(RCL=> , xFlag:= , wData:= , uiBit:=);
RCR	带进位的循环右移位指令	FC		RCR(RCR=> , xFlag:= , wData:= , uiBit:=);

②相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
xFlag	数据进位标志位	BOOL	遵照数据类型	-	数据进位标志位
wData	数据来源	WORD	遵照数据类型	-	数据来源

uiBit	循环移位数	UINT	1-1024	-	循环移位数

数据类型

	布尔	位串				整数								实数		时刻、持续时间、日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
xFlag	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
wData	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
uiBit	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-

③程序示例

1、RCL

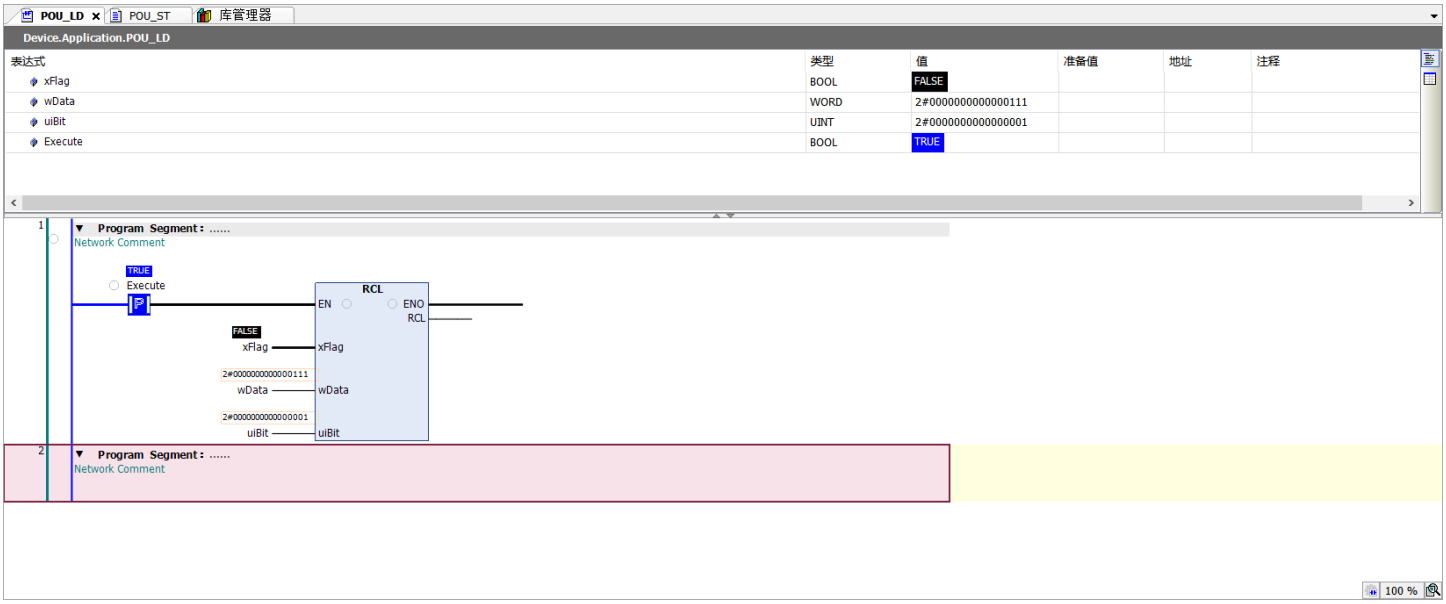
输入信号“源数据”为 2#1110000000000000，“移位数”为 1 时，触发第一次，输出信号“结果数据”为 2#1100000000000001，有效数据向左移 1 位，并且进位标志位置 ON。

2. ROR

输入信号“源数据”为 2#0000000000000011，“移位数”为 1 时，触发第一次，输出信号“结果数据”为 =2#0000000000000001，有效数据向右移 1 位，并且进位标志位置 ON。

ST:

LD:



3.5.4 SFTL/SFTR——位数据向左拷贝/位数据向右拷贝

SFTL：将源数组指定长度的位数据，连续向左拷贝到目标数组中。

SFTR：将源数组指定长度的位数据，连续向右拷贝到目标数组中。

①指令格式

指令	功能	FB/FC	LD 表现	ST表现
SFTL	位数据向左拷贝	FC	SFTL EN pbyDataSrc uiSizeSrc pbyDataDes uiSizeDes ENO SFTL	SFTL(SFTL=> , pbyDataSrc:= , uiSizeSrc:= , pbyDataDes:= , uiSizeDes:=);
SFTR	位数据向右拷贝	FC	SFTR EN pbyDataSrc uiSizeSrc pbyDataDes uiSizeDes ENO SFTR	SFTR(SFTR=> , pbyDataSrc:= , uiSizeSrc:= , pbyDataDes:= , uiSizeDes:=);

②相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
pbyDataSrc	源数据	POINTER_TO_BYTE	遵照数据类型	-	源数据
uiSizeSrc	数据源个数	BYTE	遵照数据类型	-	数据源个数
pbyDataDes	目标数据	POINTER_TO_BYTE	遵照数据类型	-	目标数据

uiSizeDes	目标数据个数	BYTE	遵照数据类型	-	目标数据个数

数据类型

	布尔	位串				整数								实数		时刻、持续时间、日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
pbyDataSrc	POINTER_TO_BYTE																			
uiSizeSrc	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
pbyDataDes	POINTER_TO_BYTE																			
uiSizeDes	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

③程序示例

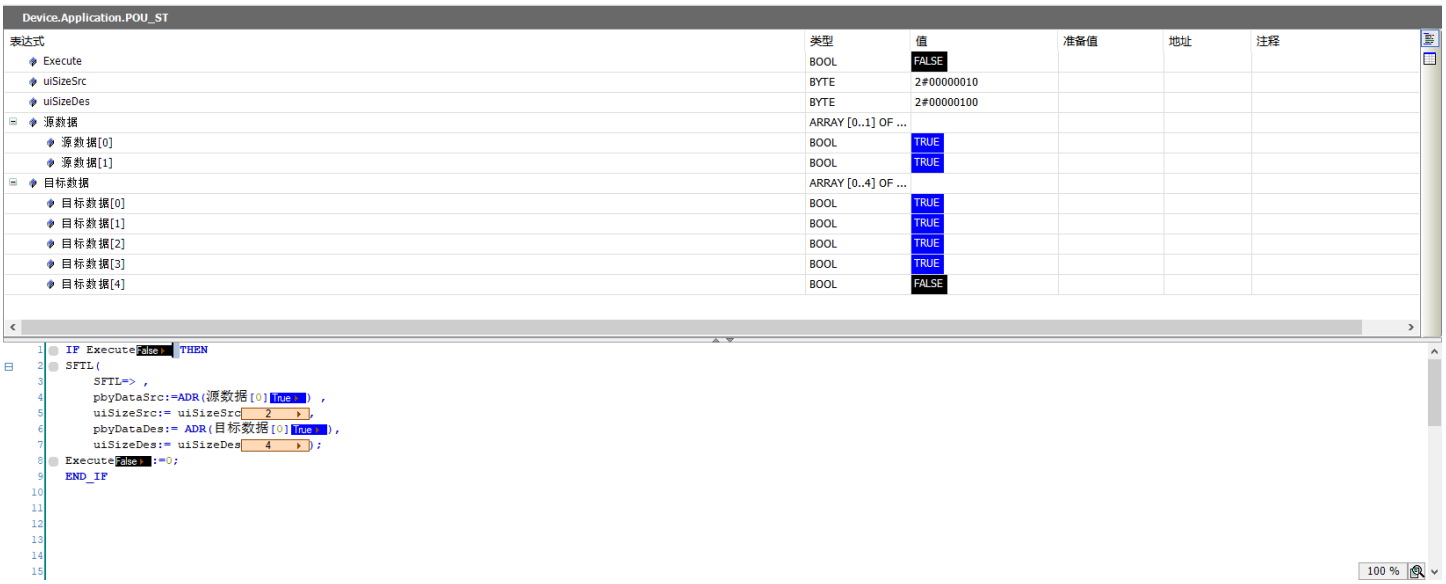
1、SFTL

将指针 pbyDataSrc（源数组）指向的位置为首地址的 uiSizeSrc（单次拷贝数组元素数）个位数据，拷贝pbyDataDes（目标数组）为首地址的低位区域，拷贝之前，pbyDataDes 的 uiSizeDes（偏移目标）个位数据全部向左移动 uiSizeSrc 位。图为触发两次之后的效果。

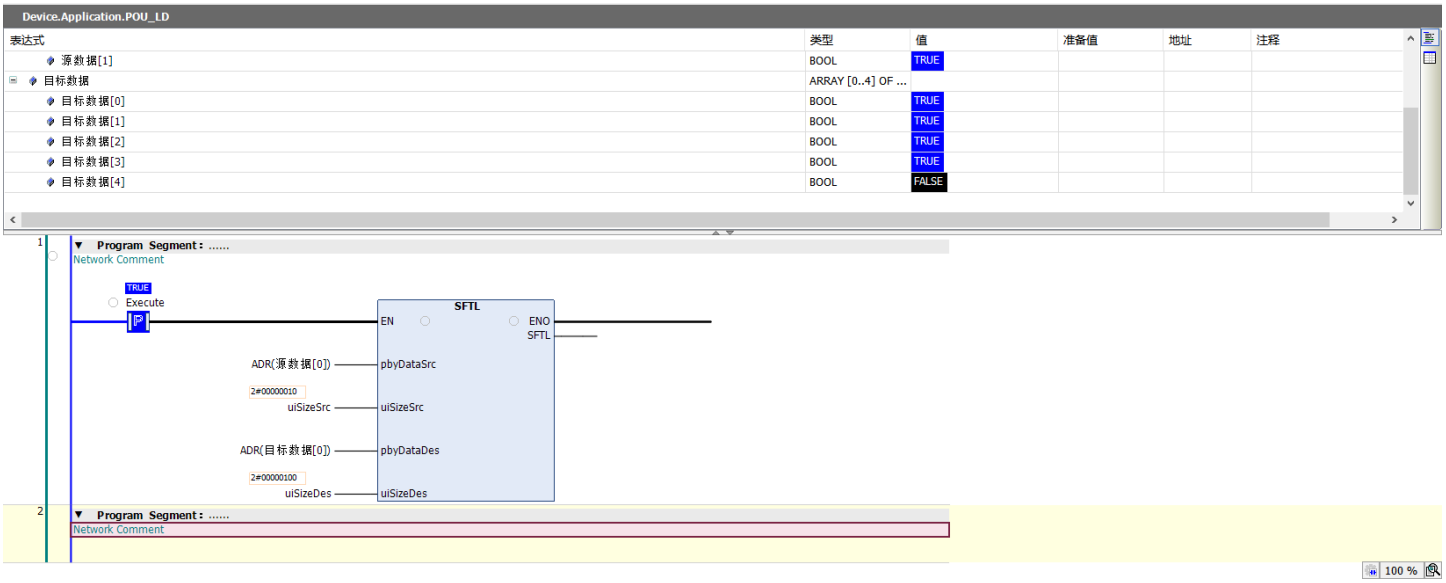
2. SFTR

将 pbyDataSrc（源数组）为首地地址的 uiSizeSrc（单次拷贝数组元素数）个位数据，拷贝 pbyDataDes（目标数组）为首地址的高位区域，拷贝之前，pbyDataDes 的 uiSizeDes（偏移目标）个位数据全部向右移动 uiSizeSrc 位。

ST:



LD:



※注意事项

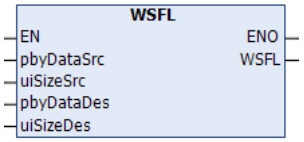
uiSizeSrc 必须不大于 uiSizeDes，否则 PLC 会异常停机。

3.5.5 WSFL/WSFR——字数据向左拷贝/字数据向右拷贝

WSFL：将源数组指定长度的字数据，连续向左拷贝到目标数组中。

WSFR：将源数组指定长度的字数据，连续向右拷贝到目标数组中。

①指令格式

指令	功能	FB/FC	LD 表现	ST表现
WSFL	字数据向左拷贝	FC		WSFL(WSFL=> , pbyDataSrc:= , uiSizeSrc:= , pbyDataDes:= , uiSizeDes:=);
WSFR	字数据向右拷贝	FC		WSFR(WSFR=> , pbyDataSrc:= , uiSizeSrc:= , pbyDataDes:= , uiSizeDes:=);

②相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
pbyDataSrc	源数据	POINTER_TO_BYTE	遵照数据类型	-	源数据
uiSizeSrc	数据源个数	BYTE	遵照数据类型	-	数据源个数

pbyDataDes	目标数据	POINTER_TO_BYTE	遵照数据类型	-	目标数据
uiSizeDes	目标数据个数	BYTE	遵照数据类型	-	目标数据个数

数据类型

	布尔	位串				整数								实数		时刻、持续时间、日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
pbyDataSrc	POINTER_TO_BYTE																			
uiSizeSrc	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
pbyDataDes	POINTER_TO_BYTE																			
uiSizeDes	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

③程序示例

1、WSFL

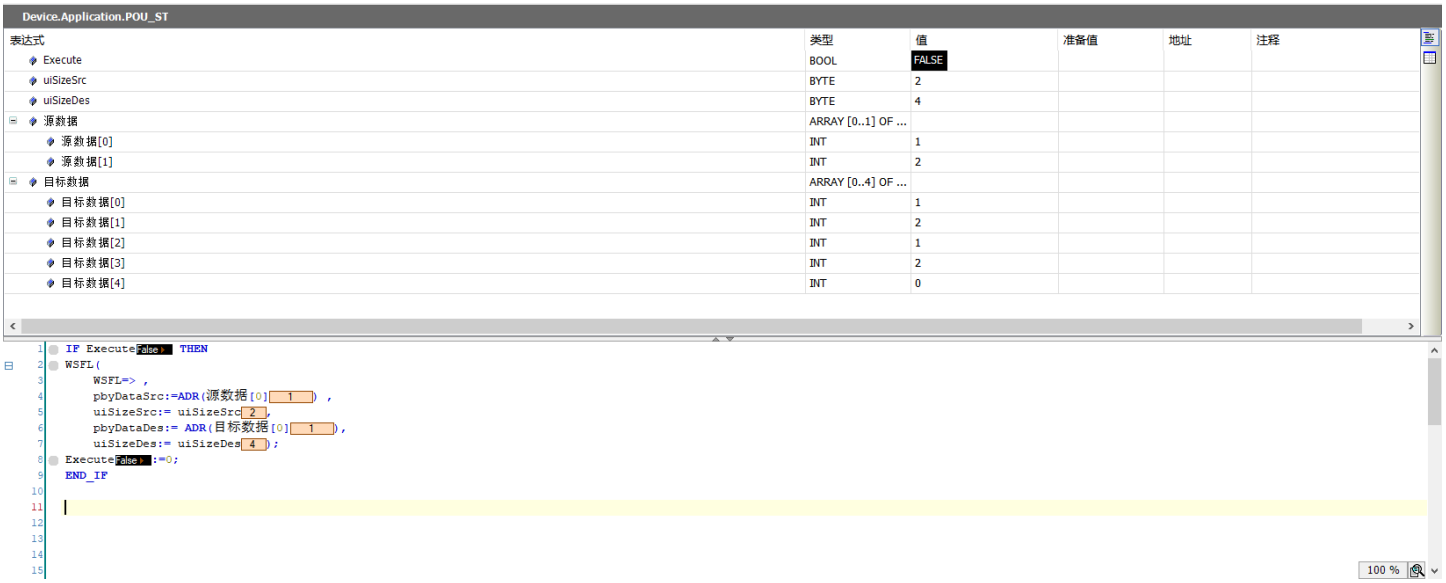
将 pwDataSrc（源数组）地址起始的 iSizeSrc（拷贝数组元素数）位变量与 pwDataDes（目标数组）地址起始的 uiSizeDes（偏移目标）变量，向左方向移动 uiSizeDes 位后，将结果保存在 pwDataDes 中。一般使用脉冲触发执行。

示例图为触发2次的效果。

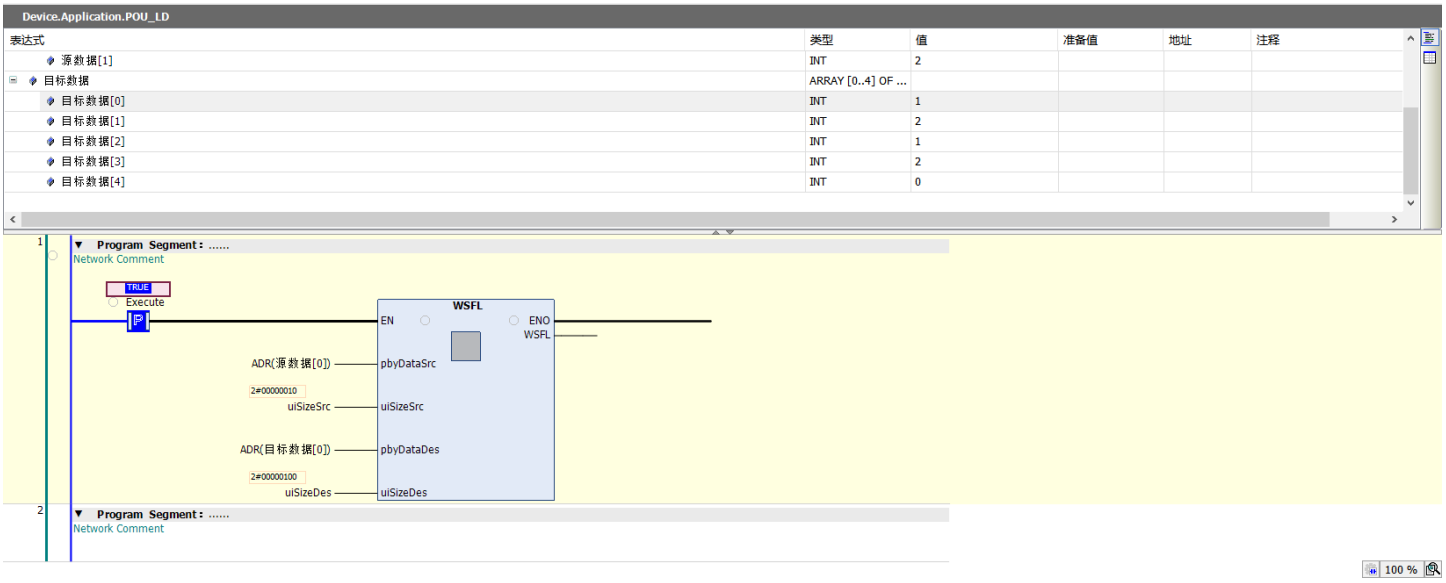
2、WSFR

将 pwDataSrc（源数组）地址起始的 iSizeSrc（拷贝数组元素数）位变量与 pwDataDes（目标数组）地址起始的 uiSizeDes(便宜目标) 变量，按向右方向移动 uiSizeDes 位后，将结果保 pwDataDes 中。一般使用脉冲触发执行。

ST:



LD:



※注意事项

uiSizeSrc 必须不大于 uiSizeDes，否则 PLC 会异常停机。

3.6 队列

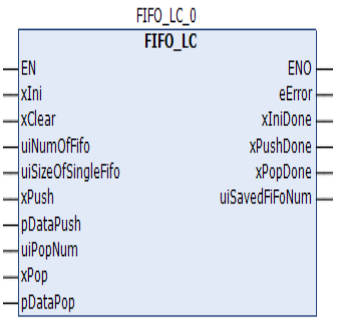
指令列表

指令类别	名称	FB/FC	功能
队列	FIFO_LC	FB	先入先出环形队列
	StackFIFO	FC	先入先出
	StackPush	FC	数据保存
	StackLIFO	FC	后入先出
	StackIns	FC	堆栈数据插入
	StackDel	FC	堆栈数据删除

3.6.1 FIFO_LC——先入先出环形队列

先入先出环形队列。

①指令格式

指令	功能	FB/FC	LD 表现	ST表现
FIFO_LC	先入先出环形队列	FB		FIFO_LC_0(xIni:= , xClear:= , uiNumOfFifo:= , uiSizeOfSingleFifo:= , xPush:= , pDataPush:= , uiPopNum:= , xPop:= , pDataPop:= ,

				eError=> , xIniDone=> , xPushDone=> , xPopDone=> , uiSavedFiFoNum=>);

②相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
xIni	初始化	BOOL	遵照数据类型	-	初始化辟内存，上电后初始化一次即可
xClear	队列清空	BOOL	遵照数据类型	-	清空触发
uiNumOfFifo	FIFO元素数量	UINT	遵照数据类型	-	FIFO通元素数量
uiSizeOfSingleFifo	单个FIFO元素长度	UINT	遵照数据类型	-	单个FIFO长度，单位字节
xPush	入队	BOOL	遵照数据类型	-	上升沿执行，数据入队
pDataPush	入队数据指针	POINTER TO BYTE	TRUE/FALSE	-	入队数据指针
uiPopNum	出队元素数量	UINT	遵照数据类型	-	出队元素数量
xPop	出队	BOOL	遵照数据类型	-	上升沿执行，数据出队
pDataPop	出队数据指针	POINTER TO BYTE	遵照数据类型	-	出队数据指针

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
eError	错误代码	ERR_FIFO	遵照数据类型	-	错误代码
xIniDone	初始化完成	BOOL	遵照数据类型	-	初始化完成
xPushDone	写入完成	BOOL	遵照数据类型	-	写入完成
xPopDone	读取完成	BOOL	遵照数据类型	-	读取完成
uiSavedFiFoNum	存入FIFO元素个数	UINT	遵照数据类型	-	存入FIFO元素个数

功能说明

创建一个先入先出的队列，包含初始化、入队列、出队列、清空功能，上升沿触发执行，一个周期执行完毕，并输出相应的标志位，如果未正常输出，请检查输入参数是否合法。

FIFO 功能块使用步骤：

1） 初始化申请队列缓存

触发 xInit 初始化 FIFO，此时 FIFO 创建缓存空间，如果成功，一个周期内 xInitDone 置 true。本例中在 PLC

※注意事项

2） 入队

触发 xPush 将 pbyDataPush 指向的数据数据，压入队列，如果成功，一个周期内 xPushDone 置位 TRUE，uiSavedFiFoNum（存入FIFO元素个数）加1。

3） 出队

触发 xPop 将 FIFO 中缓存队列的数据， uiPopNum个数的元素取出，存入pbyDataPop 指向的数据。如果成功，一个周期内 xPushDone 置位TRUE，uiSavedFiFoNum（存入FIFO元素个数）减少 uiSizePop个

。

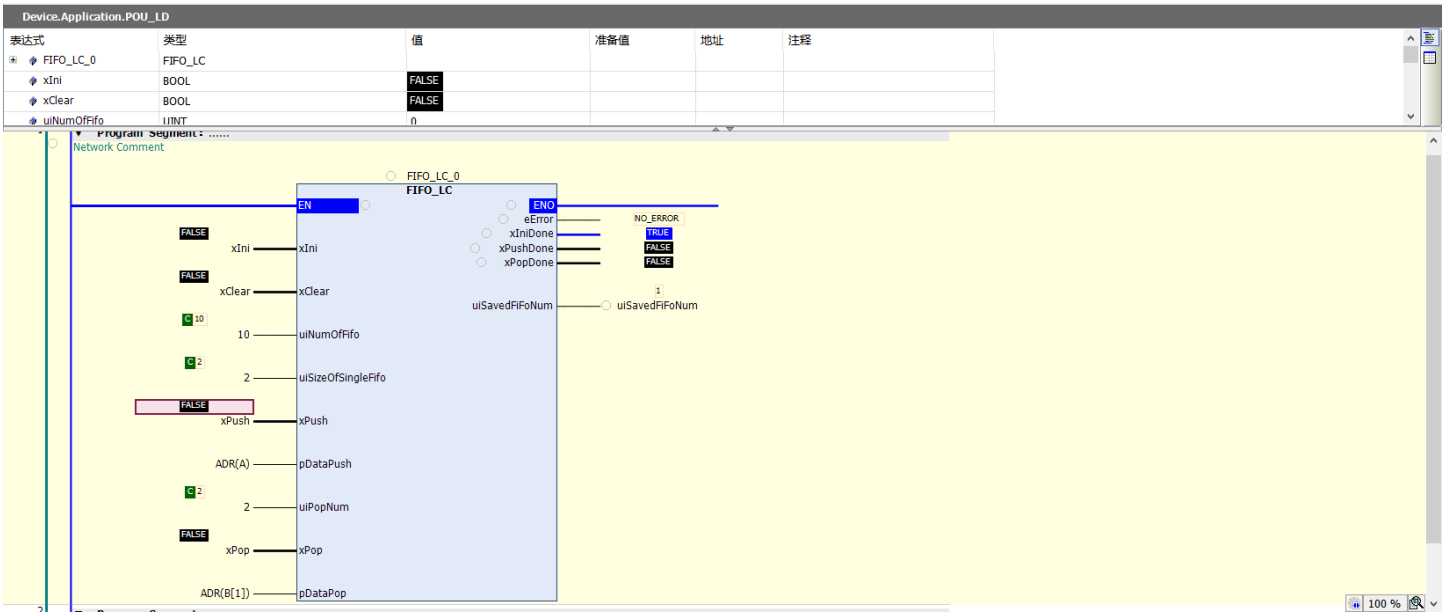
4） 队列清空

根据需要是否要清空队列。触发 xClear，如果成功，一个周期内 xClearDone 置 true。

③程序示例

ST: uiSizePop=2，每次触发 xPop 从数据队列取出最先压入的元素内容到 pbyDataPop 地址，同时队列中数据减少 2 个元素。

LD: uiSizePop=2，每次触发 xPop 从数据队列取出最先压入的元素内容到 pbyDataPop 地址，同时队列中数据减少 2 个元素,FIFO 中缓存队列元素不足，则取出无效。



3.6.2 StackFIFO——先入先出

取出堆叠最低位的值。

①指令格式

指令	功能	FB/FC	LD 表现	ST表现
StackFIFO	先入先出	FC		StackFIFO(StackFIFO=> , SIZE:= , InOut:= , OUTVal:= , Num:=);

②相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
SIZE	堆叠的要素数	UINT	遵照数据类型	-	堆叠的要素数
InOut[]	堆叠数组	-	遵照数据类型	-	堆叠数组
OUTVal	输出值	-	遵照数据类型	-	输出值

输入输出变量

输入输出变量	名称	数据类型	有效范围	初始值	描述
Num	堆叠中保存的要素	UINT	遵照数据类型	-	堆叠中保存的要素

数据类型

	布尔	位串				整数								实数		时刻、持续时间、日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
SIZE	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
InOut[]	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
	还可指定枚举体、结构体整体、结构体的 1 个成员																			
OutVal	与“InOut[]”要素的数据类型相同																			
Num	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-

功能说明

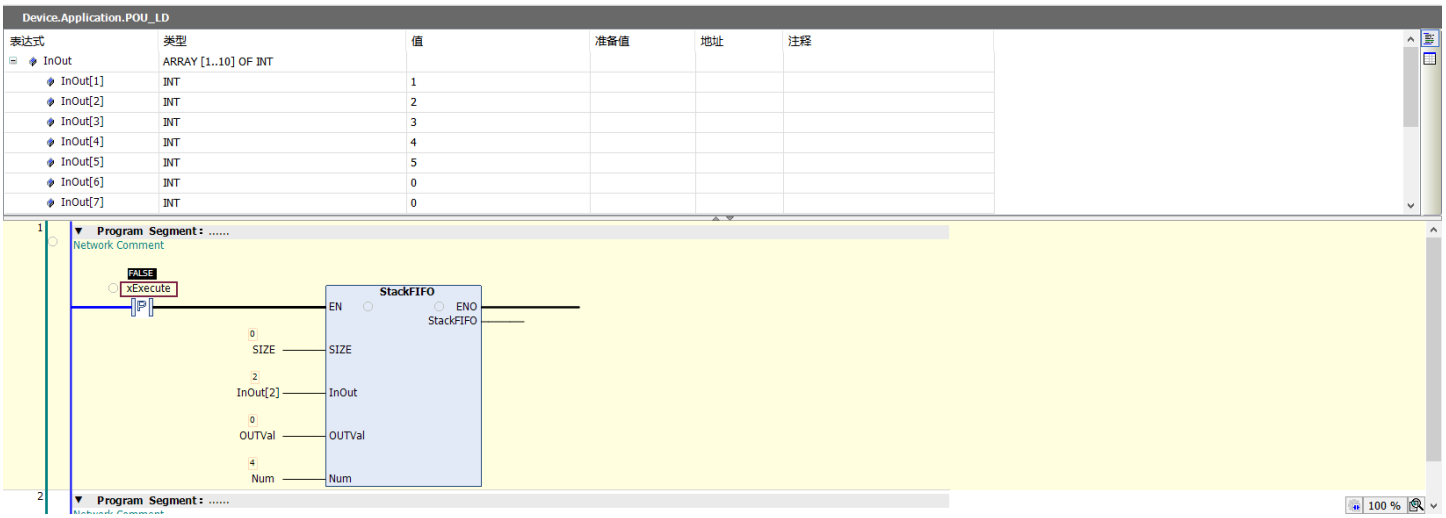
将目前已存有保存要素数“Num”个要素的数组InOut[]视为堆叠数组。从中取出堆叠最低位InOut[0]的值，代入输出值“OutVal”中。

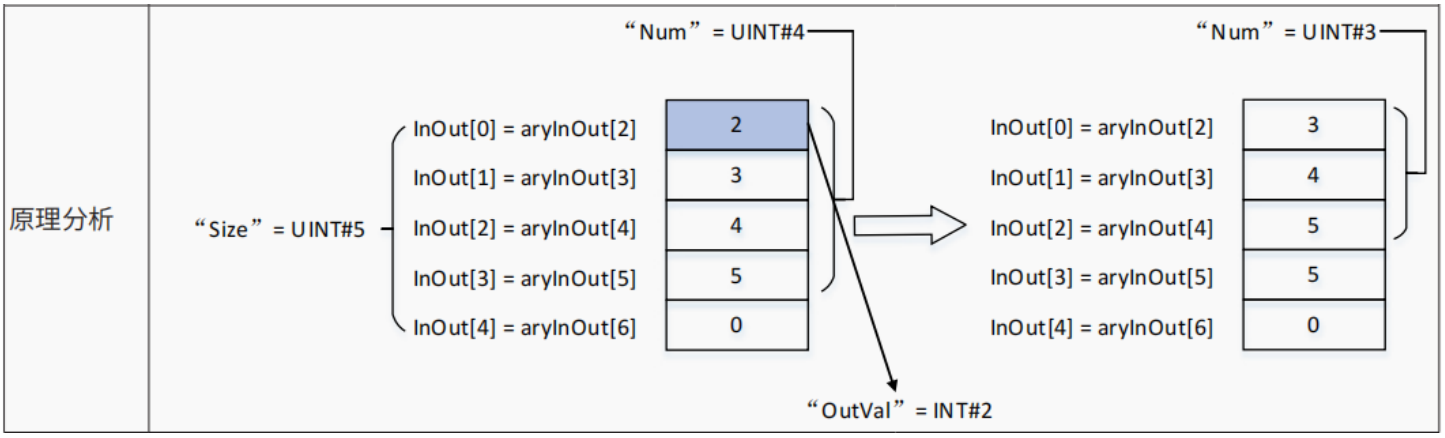
然后，将InOut[1]之后的“Num”-1个要素分别向堆叠数组的低位方向移动，“Num”减小1。

堆叠要素数“Size”为InOut中要作为堆叠使用的要素数。

ST:

LD:





※注意事项

• 请确保只执行1个扫描周期，比如：上升沿、下降沿作为执行条件，

否则运行时会得不到正确的结果。

“Size”的值超出 InOut[] 的数组区域时，函数返回值的值为 FALSE，InOut[]、“Num”、“OutVal”的值不变。

“OutVal”、InOut[] 为 STRING 型、InOut[] 的字节数超出“OutVal”的大小时，函数返回值的值为 FALSE，InOut[]、“Num”、“OutVal”的值不变。

“OutVal”和 InOut[] 要素的数据类型应一致，否则程序无法运行，函数返回值为 FALSE，InOut[]、“Num”、“OutVal”的值不变。

“Size”的值为 0 时，函数返回值为 TRUE，InOut[]、“Num”、“OutVal”的值无变化。

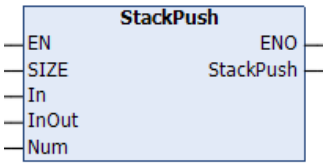
“Size”或“Num”的值为 0 以外且“Num”>“Size”时，函数返回值的值为 FALSE，InOut[]、“Num”、“OutVal”的值不变。

3.6.3 StackPush——数据保存

将值保存到堆叠中。

①指令格式

指令	功能	FB/FC	LD 表现	ST表现

StackPush	数据保存	FC		StackPush(StackPush=> , SIZE:= , In:= , InOut:= , Num:=);

②相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
SIZE	堆叠的要素数	UINT	遵照数据类型	-	堆叠的要素数
In	输入值	-	遵照数据类型		输入值
InOut[]	堆叠数组	-	遵照数据类型	-	堆叠数组

输入输出变量

输入输出变量	名称	数据类型	有效范围	初始值	描述
Num	堆叠中保存的要素	UINT	遵照数据类型	-	堆叠中保存的要素

数据类型

	布尔	位串				整数								实数		时刻、持续时间、日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
SIZE	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
In	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
	还可指定枚举体、结构体整体、结构体的 1 个成员																			
InOut[]	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
	还可指定枚举体、结构体整体、结构体的 1 个成员																			
Num	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-

功能说明

1） 将目前已存有保存要素数“Num”个要素的数组InOut[N]视为堆叠数组,将输入值“In”将覆盖掉下一个要素InOut[“Num+N”]。然后，“Num”增加 1。

2） 堆叠要素数“Size”为 InOut[] 中要作为堆叠使用的要素数,“Size”>=“Num”。

ST:

Device.Application.POU_ST

表达式	类型	值	准备值	地址	注释
InOut	ARRAY [1..10] OF INT				
InOut[1]	INT	1			
InOut[2]	INT	2			
InOut[3]	INT	3			
InOut[4]	INT	4			

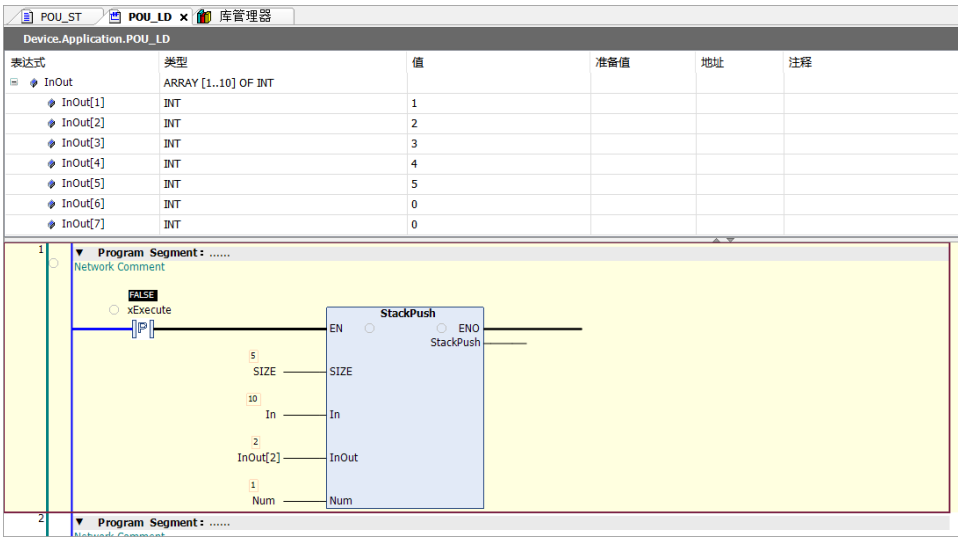
```
IF xExecute=FALSE THEN
StackPush(
    StackPush=> ,
    SIZE:= SIZE 5 ,
    In:= In 10 ,
    InOut:= InOut[2] 2 ,
    Num:= Num 1 );
xExecute=FALSE:=0;
END_IF
```

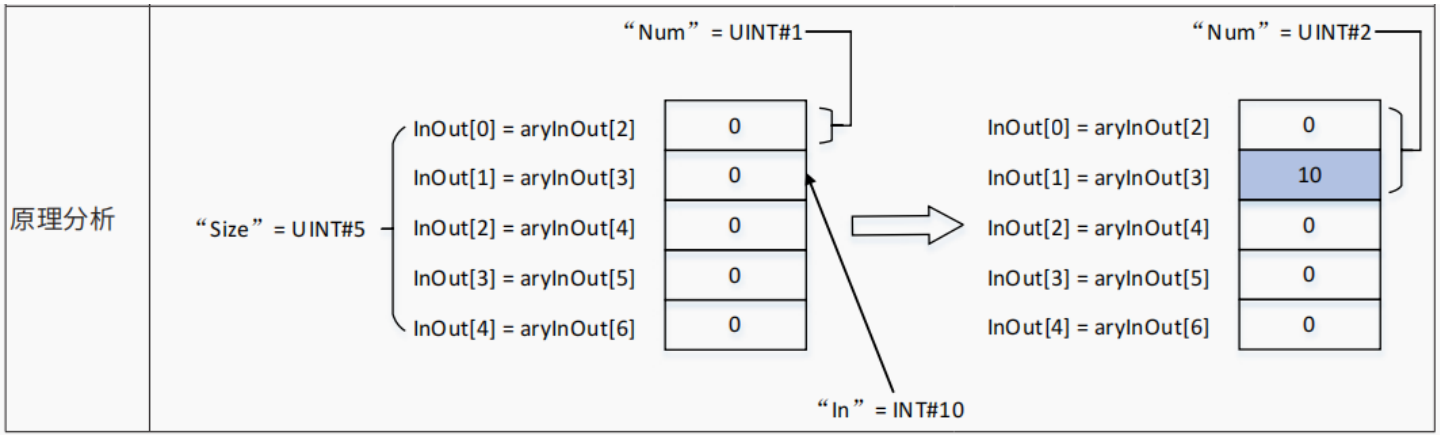
LD:

POU_ST | POU_LD | 库管理器

Device.Application.POU_LD

表达式	类型	值	准备值	地址	注释
InOut	ARRAY [1..10] OF INT				
InOut[1]	INT	1			
InOut[2]	INT	2			
InOut[3]	INT	3			
InOut[4]	INT	4			
InOut[5]	INT	5			
InOut[6]	INT	0			
InOut[7]	INT	0			





※注意事项

• 请确保只执行1个扫描周期，比如：上升沿、下降沿作为执行条件，

否则运行时会得不到正确的结果。

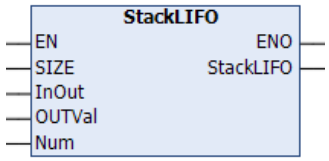
• “Size”的值超出 InOut[] 的数组区域时，函数返回值的值为 FALSE，InOut[ ] 不变。

•“In”、InOut[] 为 STRING 型、“In”的字节数超出 InOut[] 的大小时，函数返回值的值为 FALSE，InOut[] 不变。
•“In” 和 InOut[] 的数据类型不同时，编译时将发生异常。
•“Size” 的值为 0 时，InOut[]、“Num”的值无变化，函数返回值为 TRUE。
•“Size” 的值为 0 以外且“Num”≧“Size”时，程序无法运行，函数返回值为 FALSE。

3.6.4 StackLIFO——后入先出

取出堆叠最低位的值。

①指令格式

指令	功能	FB/FC	LD 表现	ST表现
StackLIFO	后入先出	FC		StackLIFO(StackLIFO=> , SIZE:= , InOut:= , OUTVal:= , Num:=);

②相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
SIZE	堆叠的要素数	UINT	遵照数据类型	-	堆叠的要素数
InOut[]	堆叠数组	-	遵照数据类型	-	堆叠数组
OUTVal	输出值	-	遵照数据类型	-	输出值

输入输出变量

输入输出变量	名称	数据类型	有效范围	初始值	描述
Num	堆叠中保存的要素	UINT	遵照数据类型	-	堆叠中保存的要素

数据类型

	布尔	位串				整数								实数		时刻、持续时间、日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
SIZE	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
InOut[]	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√

	还可指定枚举体、结构体整体、结构体的 1 个成员																			
OutVal	与“InOut[]”要素的数据类型相同																			
Num	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-

功能说明

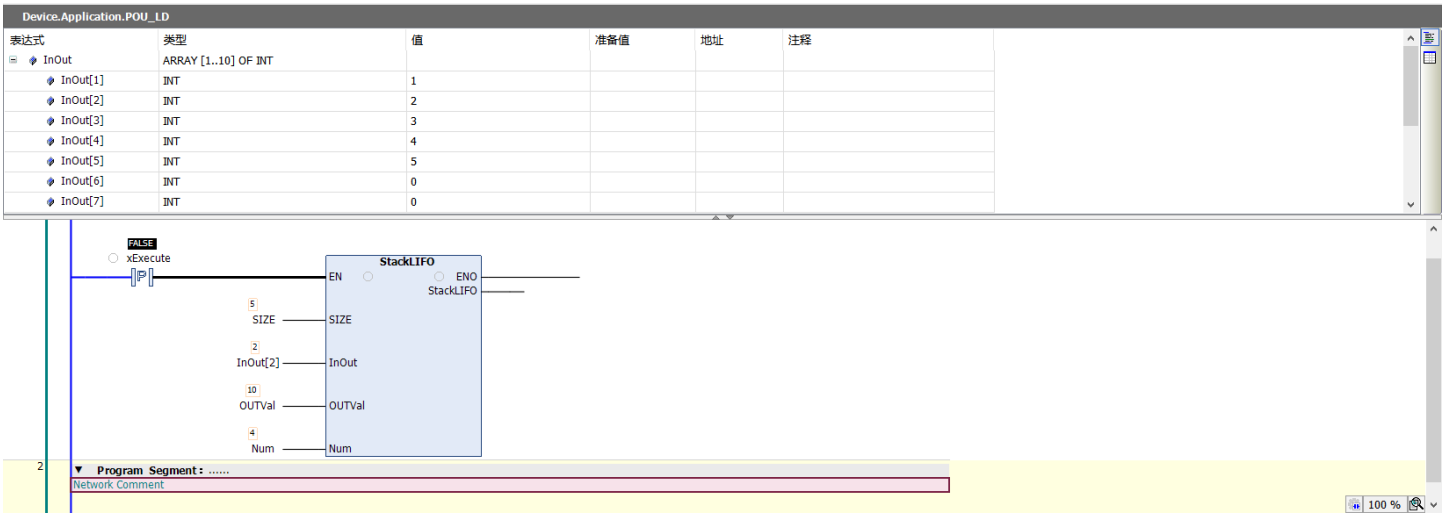
将目前已存有保存要素数“Num”个要素的数组InOut[]视为堆叠数组。从中取出堆叠最低位InOut[0]的值，代入输出值“OutVal”中。

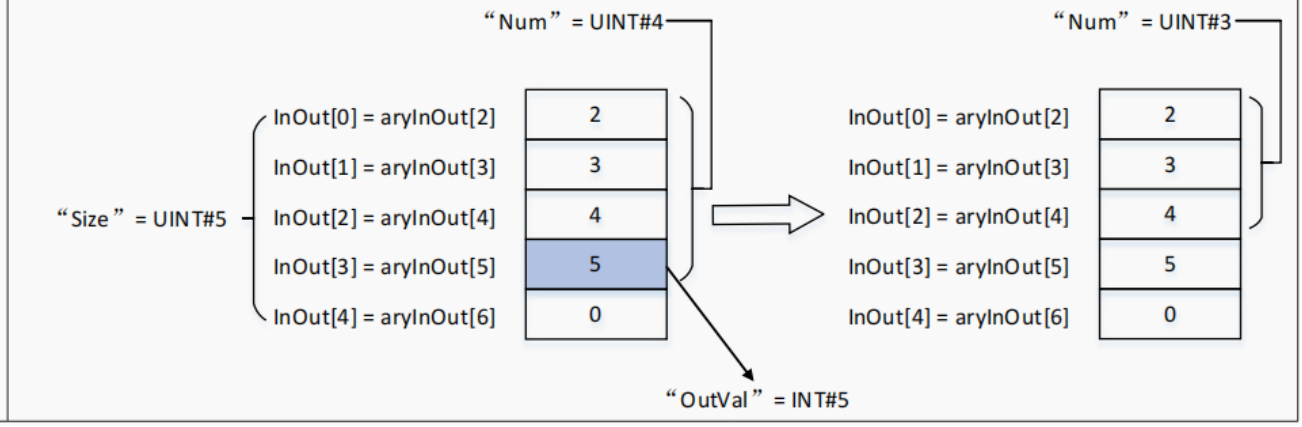
然后，将InOut[1]之后的“Num”-1个要素分别向堆叠数组的低位方向移动，“Num”减小1。

堆叠要素数“Size”为InOut中要作为堆叠使用的要素数。

ST:

LD:



项目	LD	ST
原理分析		

※注意事项

• 请确保只执行1个扫描周期，比如：上升沿、下降沿作为执行条件，

否则运行时会得不到正确的结果。

“Size”的值超出 InOut[] 的数组区域时，函数返回值的值为 FALSE，InOut[]、“Num”、“OutVal”的值不变。

“OutVal”、InOut[] 为 STRING 型、InOut[] 的字节数超出“OutVal”的大小时，函数返回值的值为 FALSE，InOut[]、“Num”、“OutVal”的值不变。

“OutVal”和 InOut[] 要素的数据类型应一致，否则程序无法运行，函数返回值为 FALSE，InOut[]、“Num”、“OutVal”的值不变。

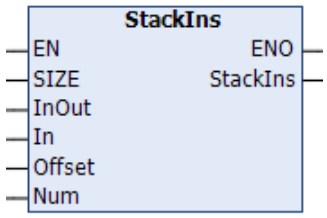
“Size”的值为 0 时，函数返回值为 TRUE，InOut[]、“Num”、“OutVal”的值无变化。

“Size”或“Num”的值为 0 以外且“Num”>“Size”时，函数返回值的值为 FALSE，InOut[]、“Num”、“OutVal”的值不变。

3.6.5 StackIns——堆栈数据插入

将值插入到堆叠的任意位置。

①指令格式

指令	功能	FB/FC	LD 表现	ST表现
StackIns	堆栈数据插入	FC		StackIns(StackIns=> , SIZE:= , InOut:= , In:= , Offset:= , Num:=);

②相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述

SIZE	堆叠的要素数	UINT	遵照数据类型	-	堆叠的要素数
In	输入值	-	遵照数据类型		输入值
InOut[]	堆叠数组	-	遵照数据类型	-	堆叠数组
Offset	偏置位置	UINT	遵照数据类型	-	堆叠的要素数

输入输出变量

输入输出变量	名称	数据类型	有效范围	初始值	描述
Num	堆叠中保存的要素	UINT	遵照数据类型	-	检索元素位置

数据类型

	布尔	位串				整数								实数		时刻、持续时间、日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
SIZE	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
In	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
	还可指定枚举体、结构体整体、结构体的 1 个成员																			
InOut[]	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
	还可指定枚举体、结构体整体、结构体的 1 个成员																			
Num	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
Offset	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-

功能说明

1） 将目前已存有保存要素数“Num”个要素的数组 InOut[]视为堆叠数组。将输入值“In”插入到偏置位置“Offset”确定的堆叠中的位置InOut[“Offset”]。

2） 更高位的要素，即InOut[“Offset”]到InOut[“Num”-1]，则分别向堆叠数组的高位方向移动。然后，“Num”增加1。

3） 堆叠要素数“Size”为 InOut[] 中要作为堆叠使用的要素数。

ST:

Device.Application.POU_ST

表达式	类型	值	准备值	地址	注释
InOut	ARRAY [1..10] OF INT				
InOut[1]	INT	1			
InOut[2]	INT	2			
InOut[3]	INT	3			
InOut[4]	INT	4			
InOut[5]	INT	5			
InOut[6]	INT	0			
InOut[7]	INT	0			

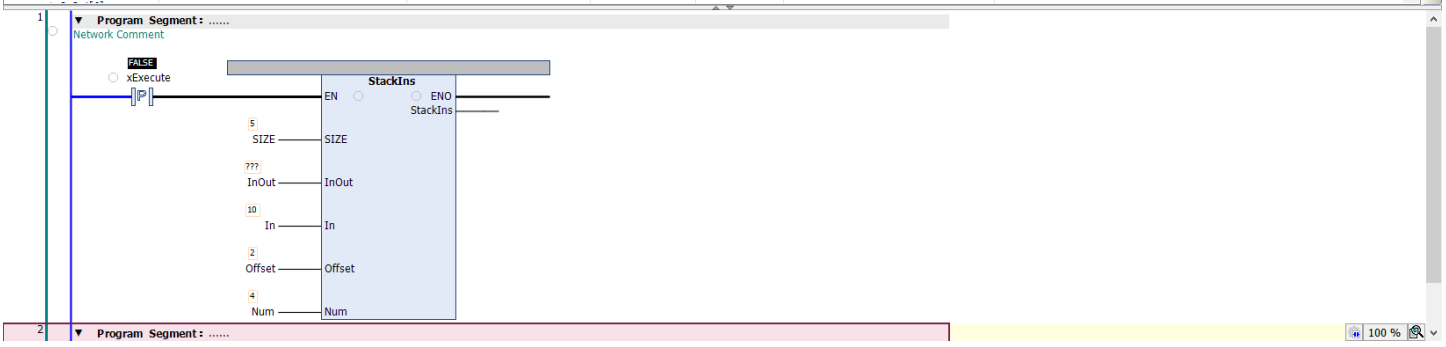
```
IF xExecute=FALSE THEN
StackIns(
    StackIns=> ,
    SIZE:= SIZE 5 ,
    InOut:= InOut[2] 2 ,
    In:= In 10 ,
    Offset:= Offset 2 ,
    Num:= Num 4 ) ;

xExecute=FALSE :=0;
END_IF
```

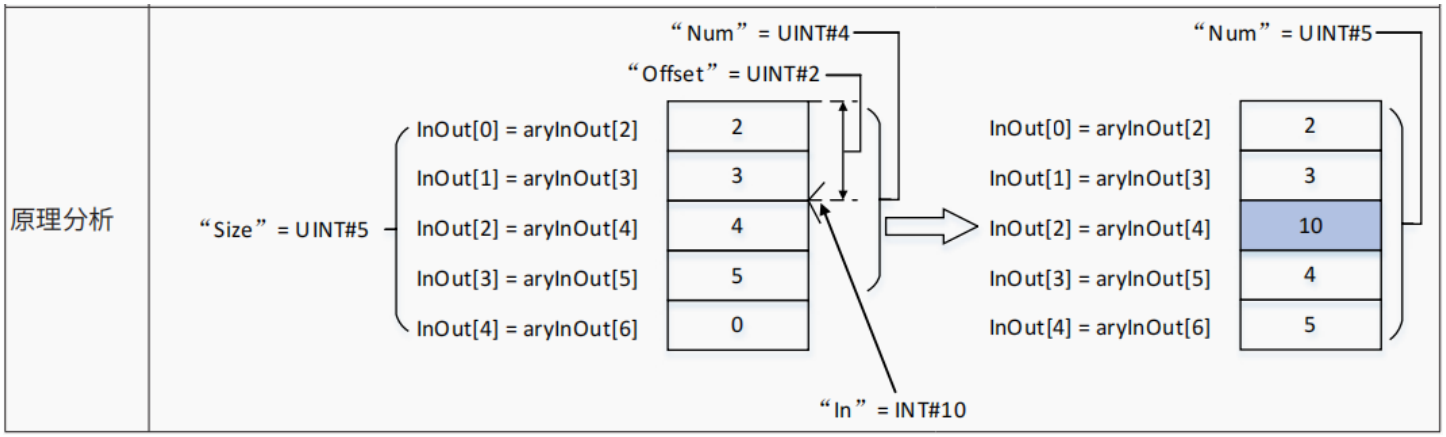
100 %

LD:

Device.Application.POU_LD

表达式	类型	值	准备值	地址	注释
InOut	ARRAY [1..10] OF INT				
InOut[1]	INT	1			
InOut[2]	INT	2			
InOut[3]	INT	3			
InOut[4]	INT	4			
InOut[5]	INT	5			





※注意事项

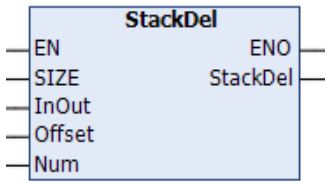
• 请确保只执行1个扫描周期，比如：上升沿、下降沿作为执行条件，否则运行时会得不到正确的结果。

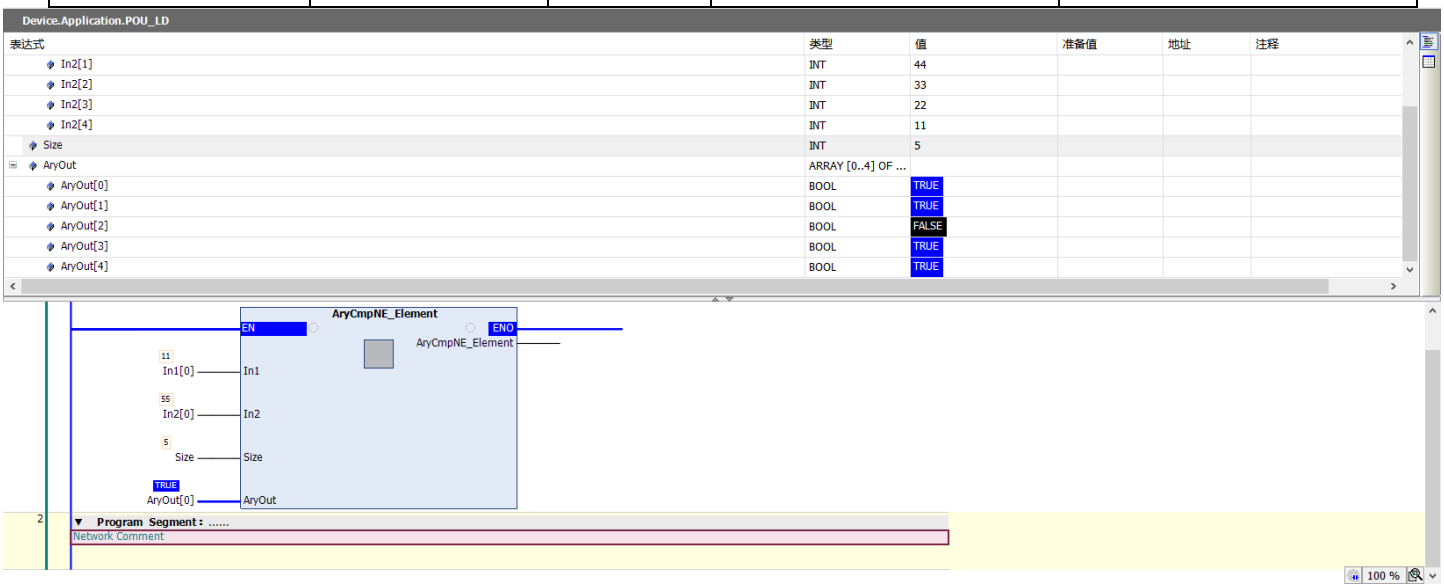
• “Size”的值超出 InOut[] 的数组区域时，函数返回值的值为 FALSE，InOut[]、“Num”的值不变。

• “In”、InOut[] 为 STRING 型、“In”的字节数超出 InOut[] 的大小时，函数返回值的值为 FALSE，InOut[]、“Num”
的值不变。
• “In” 和 InOut[ ] 的数据类型不同时，编译时将发生异常。
• “Size”的值为 0 时，函数返回值为 TRUE，InOut[]、“Num”的值不变。

3.6.6 StackDel——堆栈数据删除

删除堆叠中任意位置的值。
①指令格式

指令	功能	FB/FC	LD 表现	ST表现
StackDel	堆栈数据插入	FC		StackDel(StackDel=> , SIZE:= , InOut:= , Offset:= , Num:=);



②相关变量
输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
SIZE	堆叠的要素数	UINT	遵照数据类型	-	堆叠的要素数
In	输入值	-	遵照数据类型		输入值
InOut[]	堆叠数组	-	遵照数据类型	-	堆叠数组

输入输出变量

输入输出变量	名称	数据类型	有效范围	初始值	描述

Num	堆叠中保存的要素	UINT	遵照数据类型	-	检索元素位置

数据类型

	布尔	位串				整数								实数		时刻、持续时间、日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
SIZE	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
In	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
	还可指定枚举体、结构体整体、结构体的 1 个成员																			
InOut[]	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
	还可指定枚举体、结构体整体、结构体的 1 个成员																			
Num	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-

功能说明

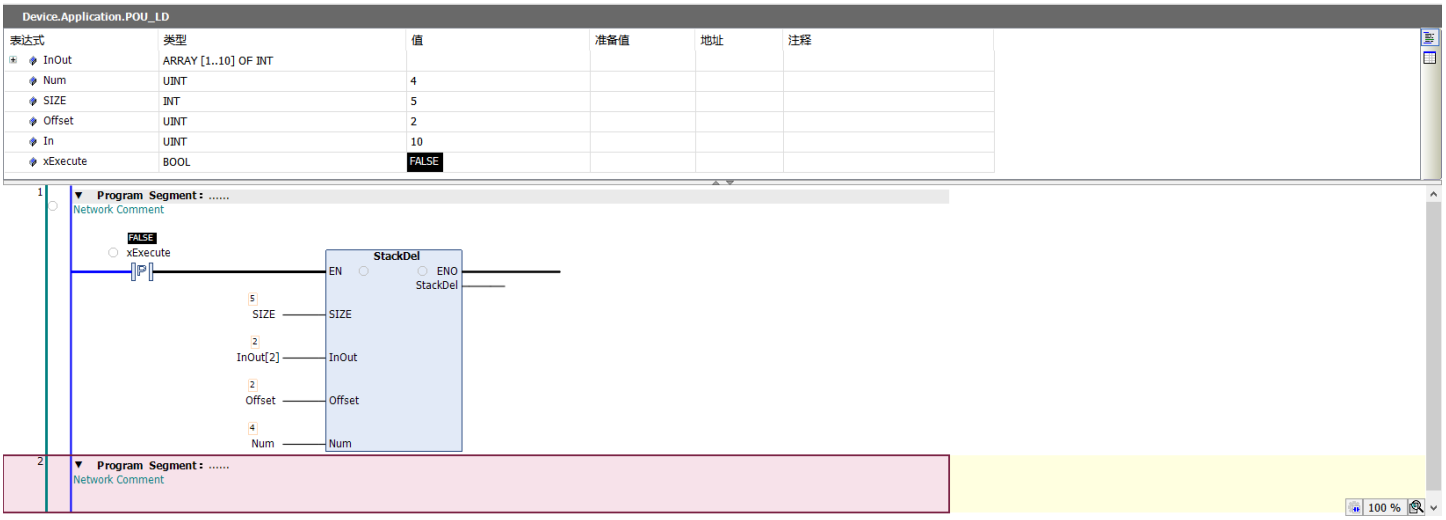
1） 将目前已存有保存要素数“Num”个要素的数组InOut[视为堆叠数组。从中删除偏置位置“Offset”确定的堆叠中的位置InOut[“Offset”]的值。

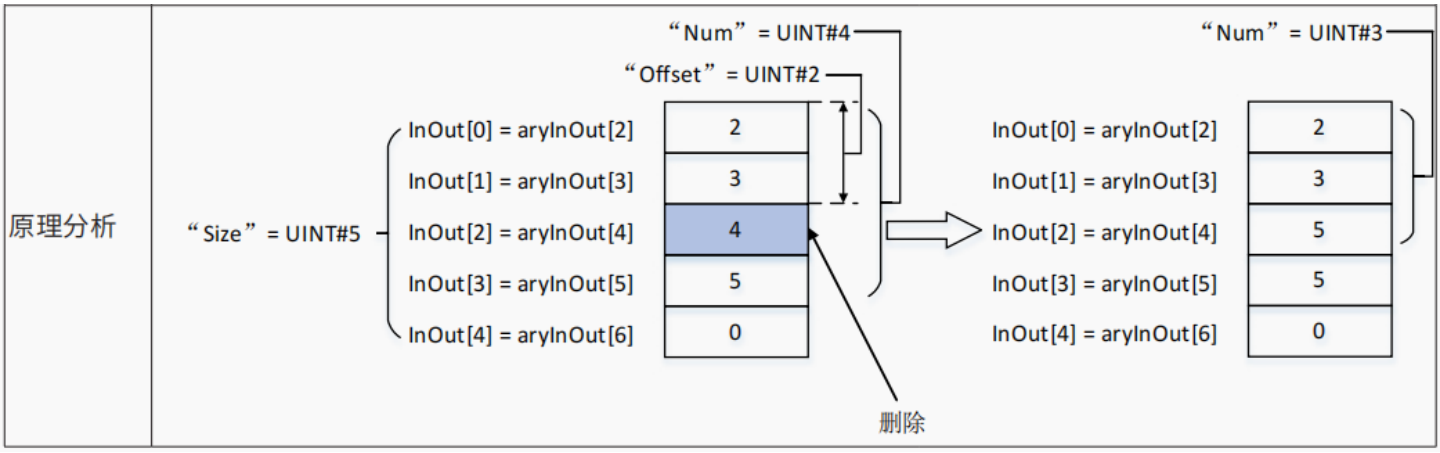
2） 更高位的要素，即InOut[“Offset”+1]到 InOut[“Num”-1]，则分别向堆叠数组的低位方向移动。然后，“Num”减小1。

3） 堆叠要素数“Size”为InOut1中要作为堆叠使用的要素数。

ST:

LD:





※注意事项

- 请确保只执行1个扫描周期，比如：上升沿、下降沿作为执行条件，

否则运行时会得不到正确的结果。

- “Size”的值超出 InOut[] 的数组区域时，函数返回值的值为 FALSE，InOut[]、“Num”的值不变。
- “Size”的值为 0 时，函数返回值为 TRUE，InOut[]、“Num”的值不变。

3.7 数据传递指令

指令列表

指令类别	名称	FB/FC	功能
数据传递操作	BMOV	FC	数组拷贝
	BON	FC	判断数据位状态
	BTOW	FC	数据拼接
	FMOV	FC	REAL 数组拷贝
	SRD	FC	读取指定长度的数据
	SFWR	FC	写入指定长度的数据
	LSWAP	FC	高低位数据交换
	WTOB	FC	16 位数据拆分高低位
	XCH	FC	将两个浮点数据进行交换

3.7.1 BMOV 数组拷贝

①指令格式

指令	功能	FB/FC	LD 表现	ST表现
BMOV	将源数据按指定长度拷贝到目标数据。	FC		BMOV(BMOV=>OUT , pByDataSrc:=ADR(IN1) , pByDataDes:= ADR(IN2), uiSize:=IN3);

②相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
pByDataSrc	数据复制起始地址	POINTER TO BYTE	-	-	被复制的源数据

uiSize	复制的数量	UINT	-	0	复制数组中的数量

输出变量

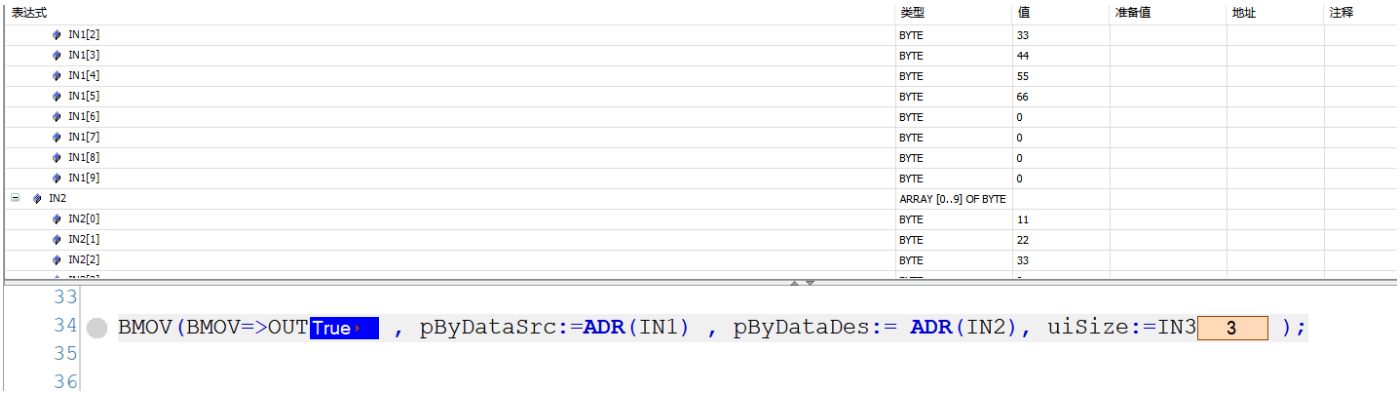
输出变量	名称	数据类型	有效范围	初始值	描述
pByDataDes	数据储存起始地址	POINTER TO BYTE	-	-	复制过来的数据
BMOV	复制完成信号	BOOL	0-1	0	完成信号

数据类型	布尔	位串				整数								实数		时刻、持续时间 日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATA	TOD	DT	STRING
pByDataSrc		√																		
uiSize							√													
pByDataDes		√																		
BMOV	√																			

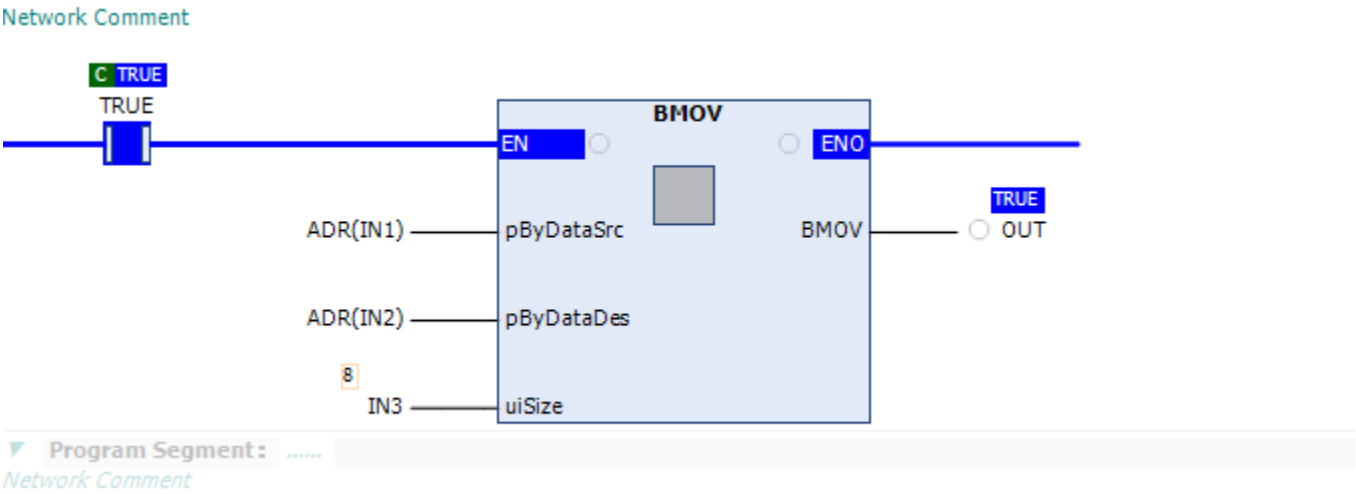
③程序示例

将 pbyDataSrc 指定起始地址的 uiSize 个变量值复制到 pbyDataDes 指定起始地址的 uiSize 。

ST:



LD:



※注意事项

uiSize是按照BYTE类型为数量的，如是其他类型数组要算此类型是几个BYTR组成的在乘上复制的数量

3.7.2 BON　判断数据位状态

①指令格式

指令	功能	FB/FC	LD 表现	ST表现
BON	判断数据 wData 的第 uiBit 位的状态，结果存入 BON 。	FC		BON(BON=>OUT , wData:=IN1 , uiBit:=IN2);

②相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
wData	源数据	WORD	-	0	被查询的源数据
uiBit	查询的位数	UINT	-	0	需要被查询位数例如WWORD有16位当前值是多少

					就判断此位置的状态是什么

输出变量

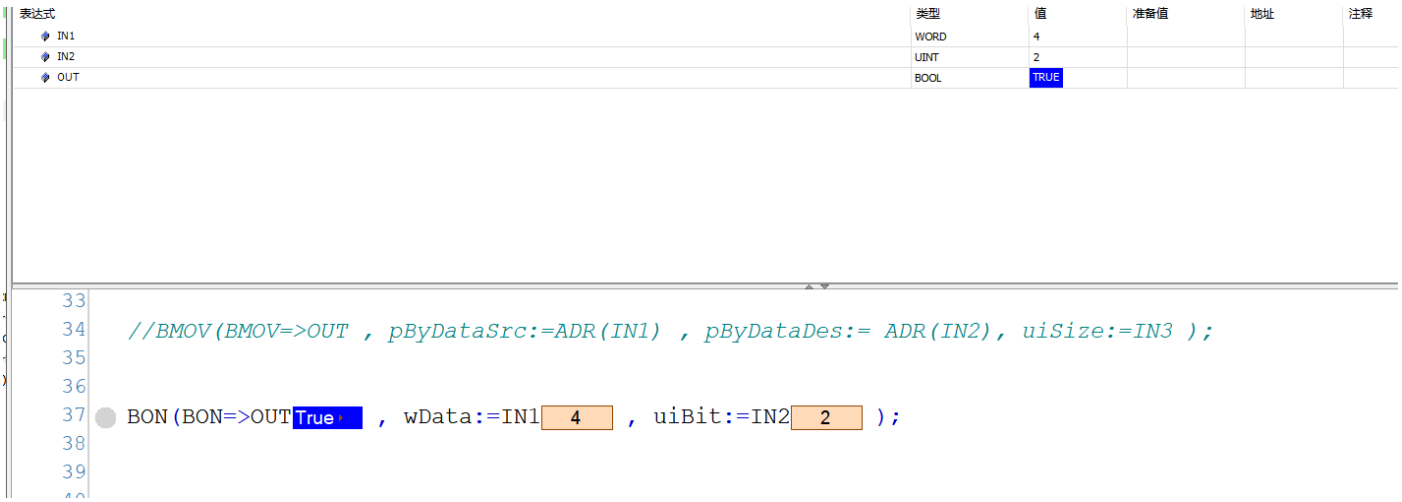
输出变量	名称	数据类型	有效范围	初始值	描述
BON	查询的结果	BOOL	-	-	查询的结果为1或0

数据类型	布尔	位串				整数								实数		时刻、持续时间 日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATA	TOD	DT	STRING
wData			√																	
uiBit							√													
BON	√																			

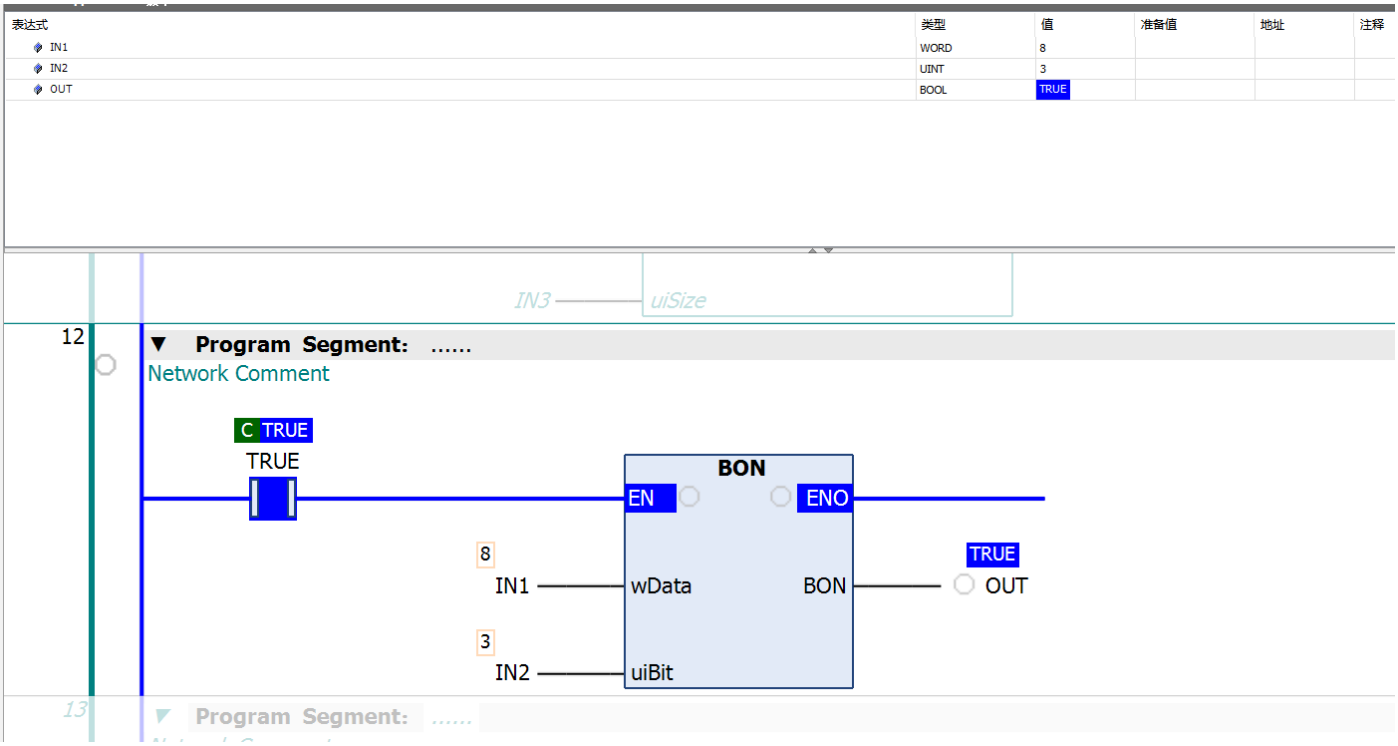
③程序示例

查询 wData变量的第uiBit个位的当前状态 。

ST:



LD:



※注意事项

3.7.3 BTOW 数据拼接

①指令格式

指令	功能	FB/FC	LD 表现	ST表现
BTOW	数据拼接	FC	BTOW pbyDataSrc POINTER TO BYTE uiSize UINT pwDataDes POINTER TO WORD BOOL BTOW	BTOW(BTOW=> , pbyDataSrc:= , uiSize:= , pwDataDes:=);

②相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
pbyDataSrc	源数据	POINTER TO WORD	-	0	要进行拼接数据
uiSize	拼接的位数	UINT	-	0	要准备几个pbyDataSrc拼接

					将低八位数据给到pwDataDes

输出变量

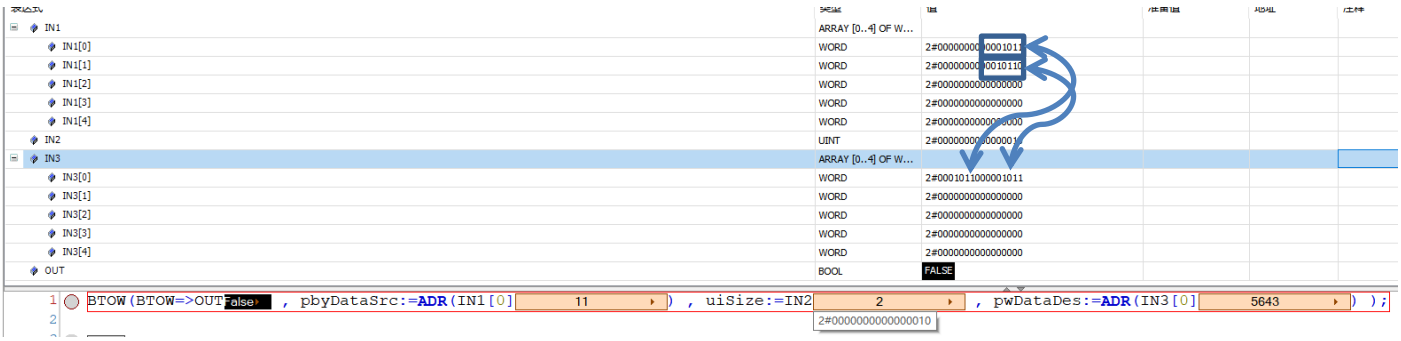
输出变量	名称	数据类型	有效范围	初始值	描述
pwDataDes	结果数据	POINTER TO WORD	-	-	拼接后收到的结果数据
BTOW	完成信号	BOOL	0-1	0	功能块完成信号

数据类型	布尔	位串				整数								实数		时刻、持续时间 日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATA	TOD	DT	STRING
pbyDataSrc		√																		
uiSize							√													
pwDataDes			√																	
BTOW	√																			

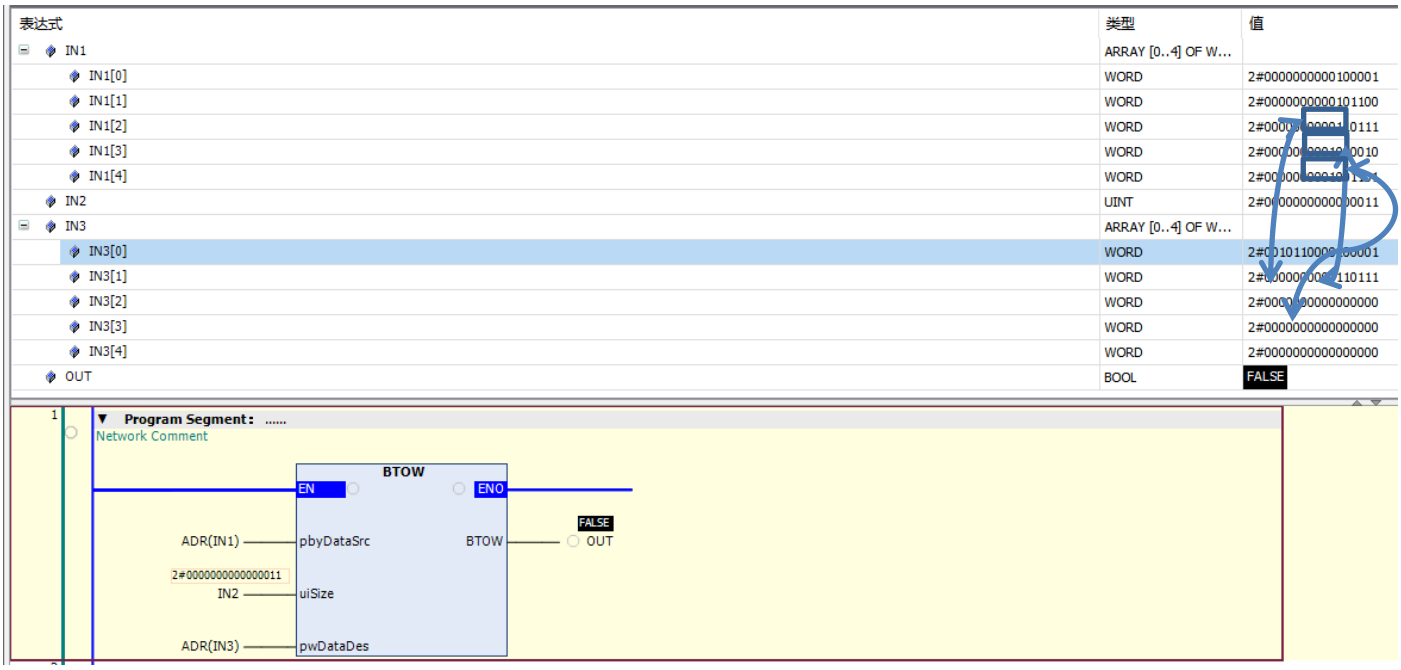
③程序示例

将准备的pbyDataSrc源数据uiSize值拼接将两个或者1个低八位数据组合给到pwDataDes 。

ST:



LD:



※注意事项

pbyDataSrc定义变量时要与pwDataDes变量类型相同，否则会读取错误导致结果不是自己想要的

pwDataDes变量收到的结果是pbyDataSrc变量数组对应位置的两个的低八位的值组合而成的

3.7.4 FMOV　REAL数组拷贝

①指令格式

指令	功能	FB/FC	LD 表现	ST表现
FMOV	REAL数组拷贝	FC		FMOV(FMOV=>OUT , fDataSrc:=IN1 , uiSize:=IN2 , pfDataDes:=ADR(IN3));

②相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
pfDataDes	源数据	REAL	-	0	被复制的源数据
FMOV	复制的数量	UINT	-	0	要复制到数组中的数量

输出变量

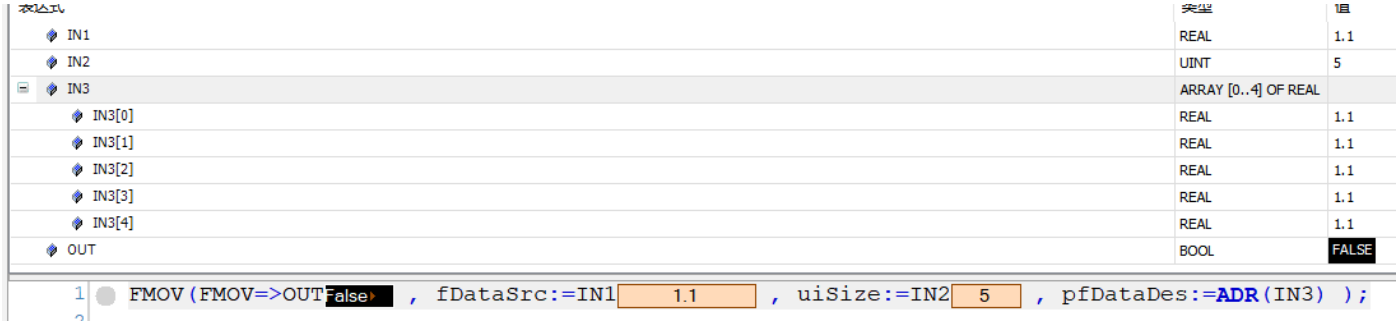
输出变量	名称	数据类型	有效范围	初始值	描述
pByDataDes	数据储存起始地址	POINTER TO REAL	-	-	复制过来的数据
BMOV	复制完成信号	BOOL	0-1	0	完成信号

数据类型	布尔	位串				整数								实数		时刻、持续时间 日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATA	TOD	DT	STRING
pfDataDes														√						
FMOV							√													
pByDataDes														√						
BMOV	√																			

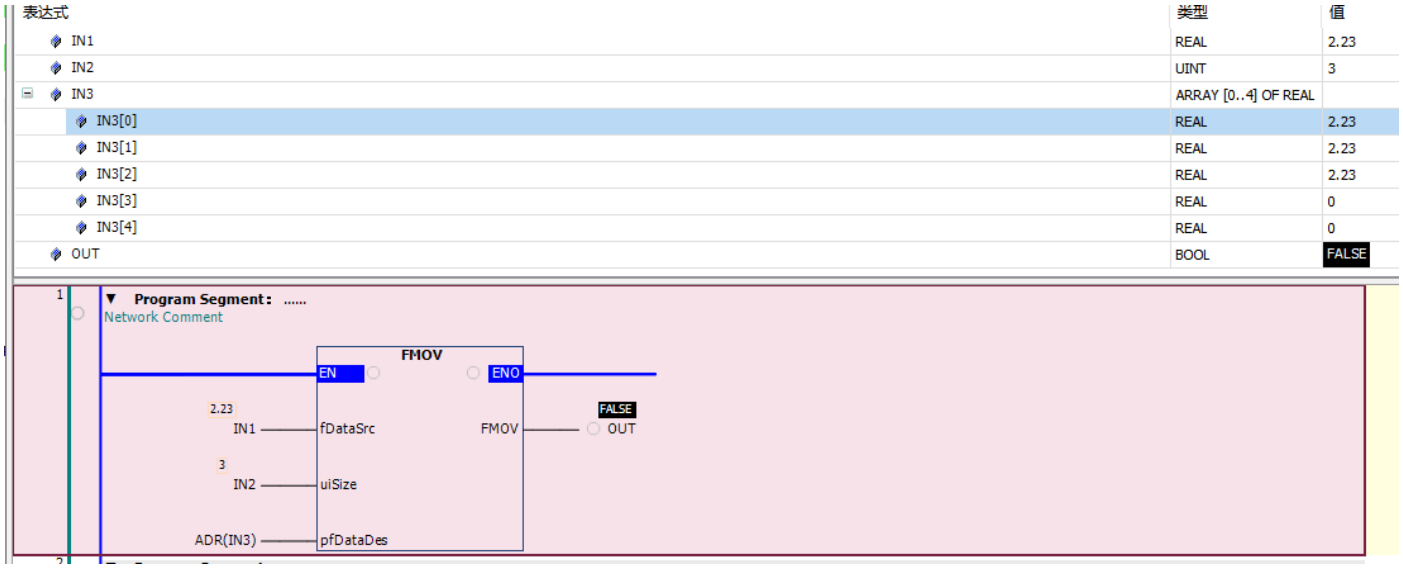
③程序示例

将pfDataDes复制下来根据FMOV要复制到pByDataDes数组中

ST:



LD:



※注意事项

此功能块只是将相同的一个变量付值到按照设定个数的数组里面

3.7.5 SFRD 读取指定长度的数据

①指令格式

指令	功能	FB/FC	LD 表现	ST表现
SFRD	读取指定长度的数据	FC		SFRD(SFRD=>OUT1 , pfDataSrc:=ADR(IN1) , uiSize:=IN2 , fDataDes=>OUT2);

②相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
pfDataSrc	源数据	ARRAY[*..*] OF REAL	-	0	被读取的源数据
uiSize	复制的数量	UINT	-	0	读取的长度

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
SFRD	完成信号	BOOL	0-1	0	完成信号
fDataDes	读出数据	REAL	-	0	从pfDataSrc拉出来的当前值

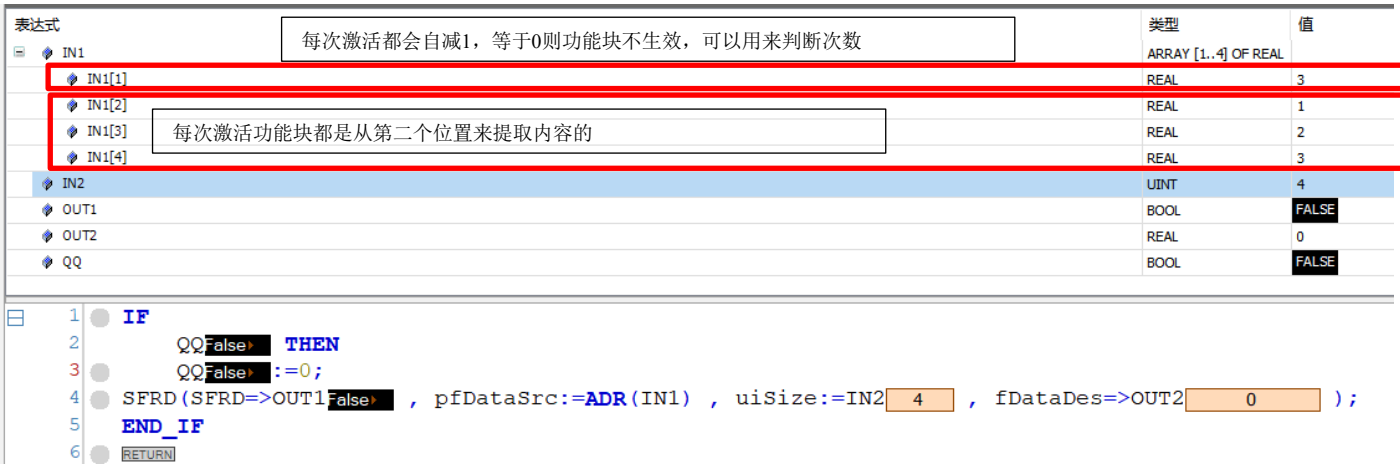
数据类型	布尔	位串				整数								实数		时刻、持续时间 日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATA	TOD	DT	STRING
pfDataSrc														√						
uiSize							√													
SFRD	√																			
fDataDes														√						

③程序示例

设置了个REAL类型的数组，数组的第一个变量不能小于1不能大于uiSize否则功能块不生效，每次激活功能块会将数组第二个变量依次提取到fDataDes中，提取的uiSize长度的变量集体前移，第一个变量每次又会自减1

ST:

1. 初始状态



2. 第一次触发

IN1	ARRAY [1..4] OF REAL	
IN1[1]	REAL	2
IN1[2]	REAL	2
IN1[3]	REAL	3
IN1[4]	REAL	3
IN2	UINT	4
OUT1	BOOL	FALSE
OUT2	REAL	1
QQ	BOOL	FALSE

每次激活功能块提取内容后整体数据都会集体向上移动一位，直到次数为零

```
IF
    QQ False THEN
    QQ False :=0;
SFRD(SFRD=>OUT1 False , pfDataSrc:=ADR(IN1) , uiSize:=IN2 4 , fDataDes=>OUT2 1 );
END_IF
RETURN
```

3.第二次触发

表达式	类型	值
IN1	ARRAY [1..4] OF REAL	
IN1[1]	REAL	1
IN1[2]	REAL	3
IN1[3]	REAL	3
IN1[4]	REAL	3
IN2	UINT	4
OUT1	BOOL	FALSE
OUT2	REAL	2
QQ	BOOL	FALSE

```
IF
    QQ False THEN
    QQ False :=0;
SFRD(SFRD=>OUT1 False , pfDataSrc:=ADR(IN1) , uiSize:=IN2 4 , fDataDes=>OUT2 2 );
END_IF
RETURN
```

4.第三次触发

IN1	ARRAY [1..4] OF REAL	
IN1[1]	REAL	0
IN1[2]	REAL	3
IN1[3]	REAL	3
IN1[4]	REAL	3
IN2	UINT	4
OUT1	BOOL	FALSE
OUT2	REAL	3
QQ	BOOL	FALSE

```
IF
    QQ False THEN
    QQ False :=0;
SFRD(SFRD=>OUT1 False , pfDataSrc:=ADR(IN1) , uiSize:=IN2 4 , fDataDes=>OUT2 3 );
END_IF
RETURN
```

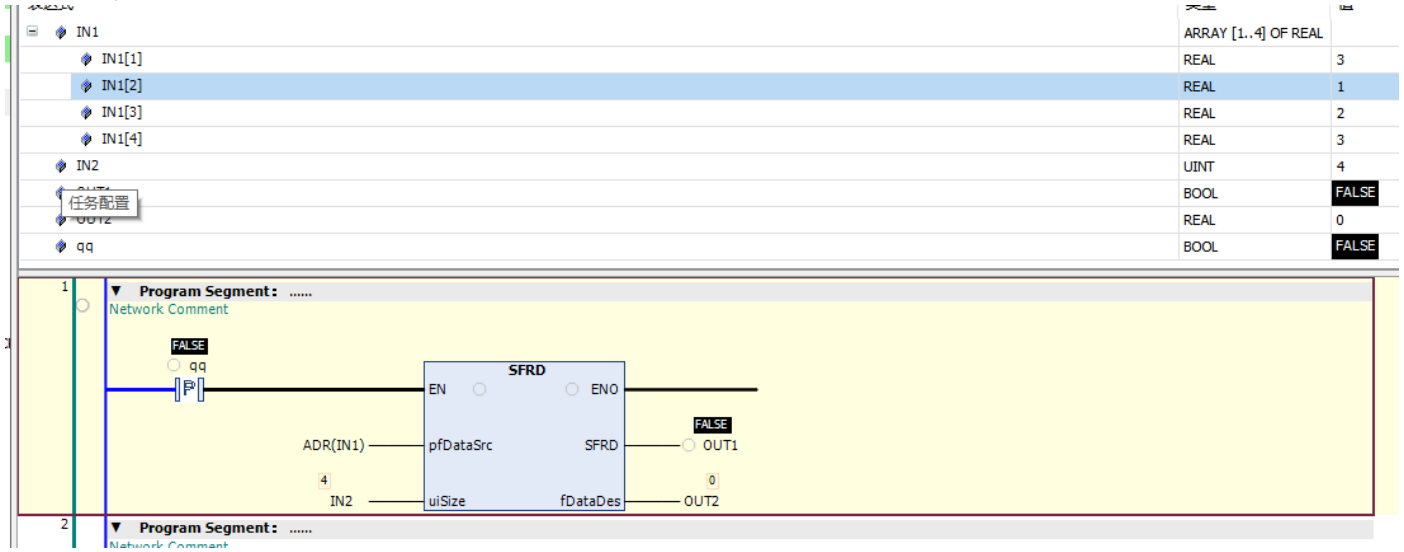
5.第四次触发

表达式	类型	值
IN1	ARRAY [1..4] OF REAL	
IN1[1]	REAL	0
IN1[2]	REAL	3
IN1[3]	REAL	3
IN1[4]	REAL	3
IN2	UINT	4
OUT1	BOOL	FALSE
OUT2	REAL	0
QQ	BOOL	FALSE

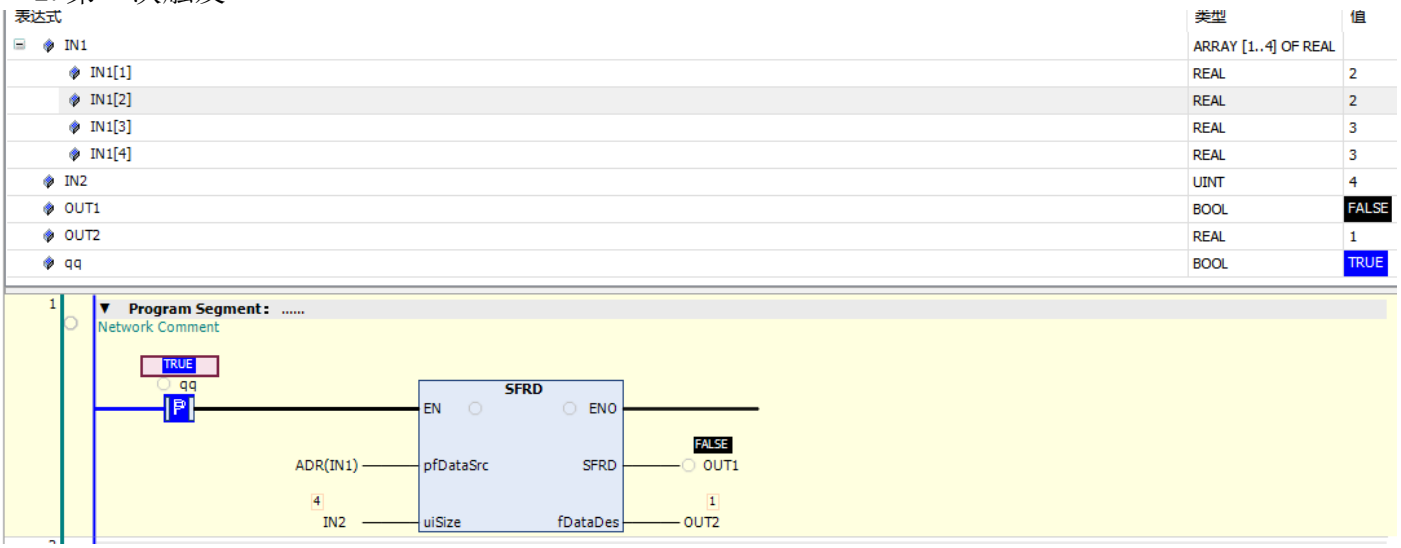
```
IF
    QQ False THEN
    QQ False :=0;
SFRD(SFRD=>OUT1 False , pfDataSrc:=ADR(IN1) , uiSize:=IN2 4 , fDataDes=>OUT2 0 );
END_IF
RETURN
```

LD:

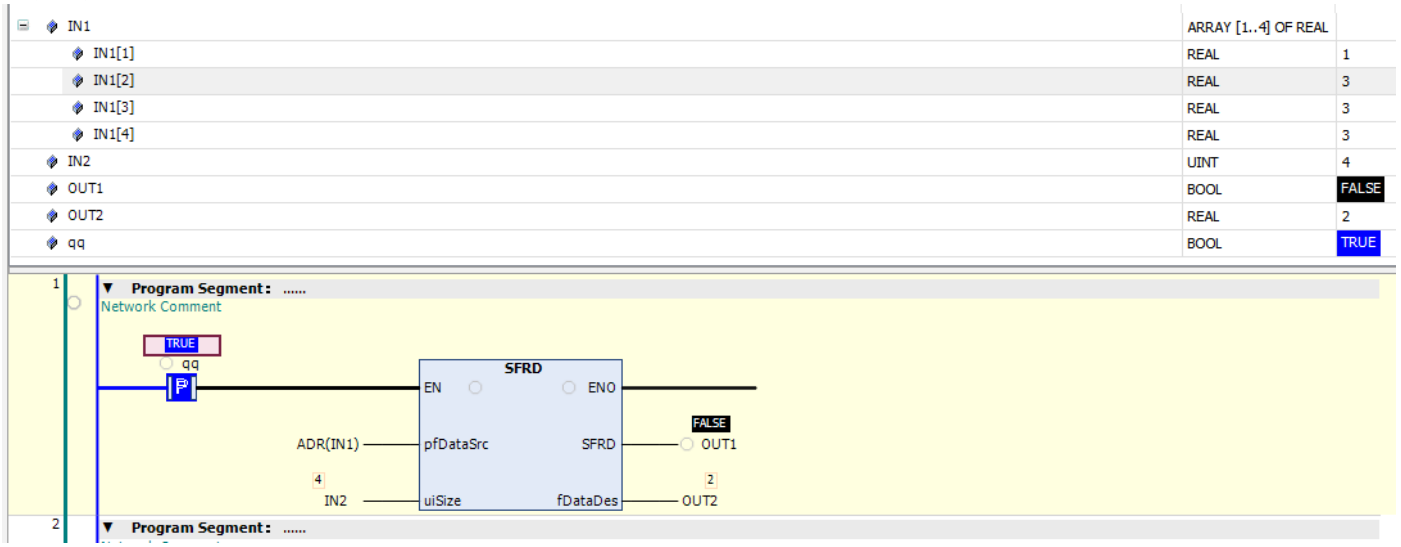
1. 初始状态



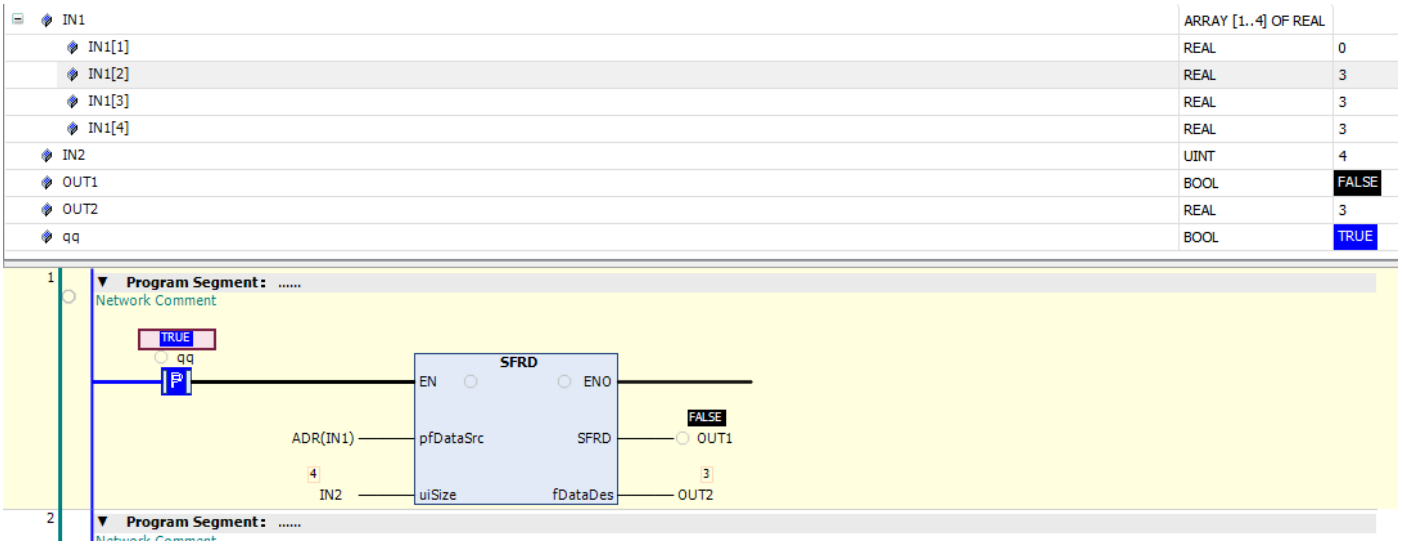
2. 第一次触发



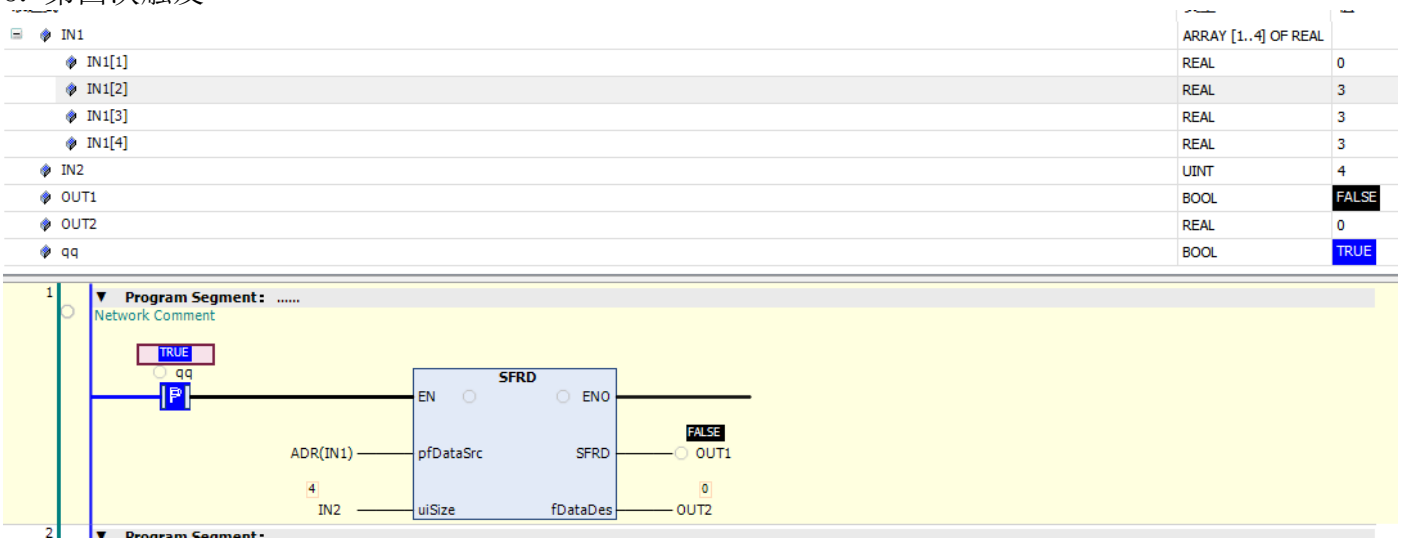
3.第二次触发



4.第三次触发



5. 第四次触发



※注意事项

3.7.6 SFWR　写入指定长度的数据

①指令格式

指令	功能	FB/FC	LD 表现	ST表现
SFWR	写入指定长度的数据	FC		SFWR(SFWR=>OUT , fDataSrc:=IN1 , pfDataDes:=IN2 , uiSize:=IN3);

②相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述

fDataSrc	源数据	REAL	-	0	要写入的源数据
uiSize	写入的数量	UINT	-	0	写入的长度

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
SFWR	完成信号	BOOL	0-1	0	完成信号
pfDataDes	读出数据	ARRAY[*..*] OF REAL	-	0	从fDataSrc得到的值

数据类型	布尔	位串				整数								实数		时刻、持续时间 日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATA	TOD	DT	STRING
fDataSrc														√						
uiSize							√													
SFWR	√																			
pfDataDes														√						

③程序示例

设置了个REAL类型的数组，每次激活功能块会将fDataSrc依次写入到数组pfDataDes第二个变量每次激活都会集体后移直到达到uiSize设定值，，第一个变量每次又会自加1

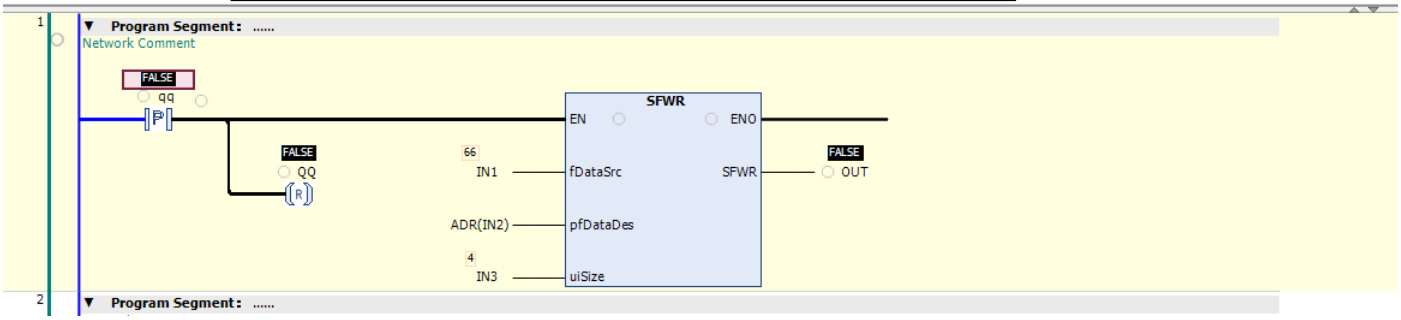
LD:

1.初始状态

表达式	类型	值
qq	BOOL	FALSE
IN1	REAL	66
IN2	ARRAY [1..4] OF REAL	
IN2[1]	REAL	0
IN2[2]	REAL	0
IN2[3]	REAL	0
IN2[4]	REAL	0
IN3	UINT	4
OUT	BOOL	FALSE

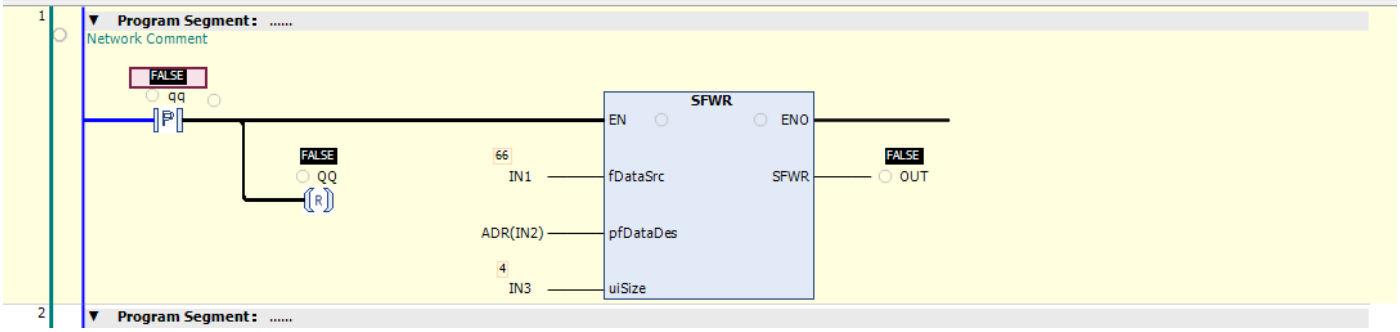
每次激活都会自加1，等于设定次数则功能块不生效，可以用来判断当前激活次数

每次激活功能块都会将要复制的数据依次往下赋值，直到达到设定次数



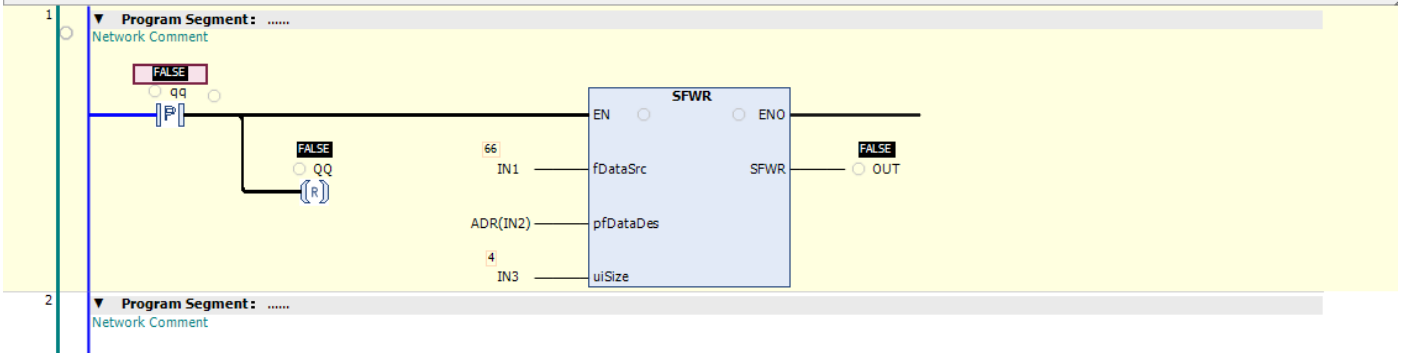
2. 第一次触发

表达式	类型	值
qq	BOOL	FALSE
IN1	REAL	66
IN2	ARRAY [1..4] OF REAL	
IN2[1]	REAL	1
IN2[2]	REAL	66
IN2[3]	REAL	0
IN2[4]	REAL	0
IN3	UINT	4
OUT	BOOL	FALSE

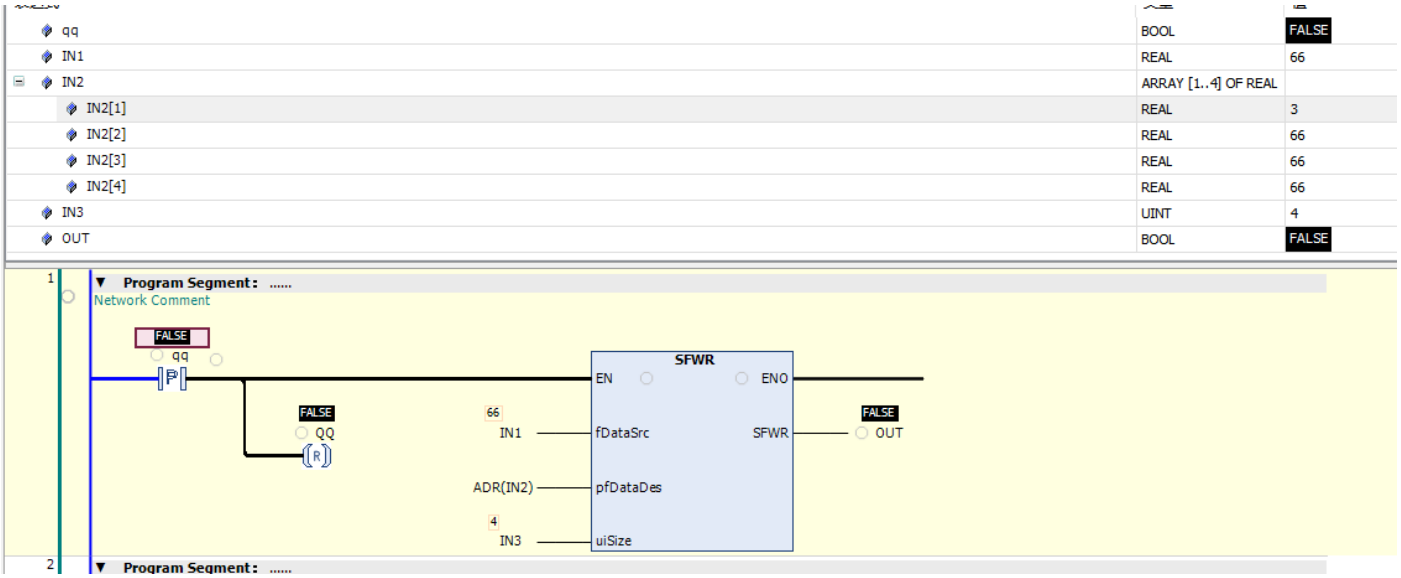


3.第二次触发

表达式	类型	值
qq	BOOL	FALSE
IN1	REAL	66
IN2	ARRAY [1..4] OF REAL	
IN2[1]	REAL	2
IN2[2]	REAL	66
IN2[3]	REAL	66
IN2[4]	REAL	0
IN3	UINT	4
OUT	BOOL	FALSE

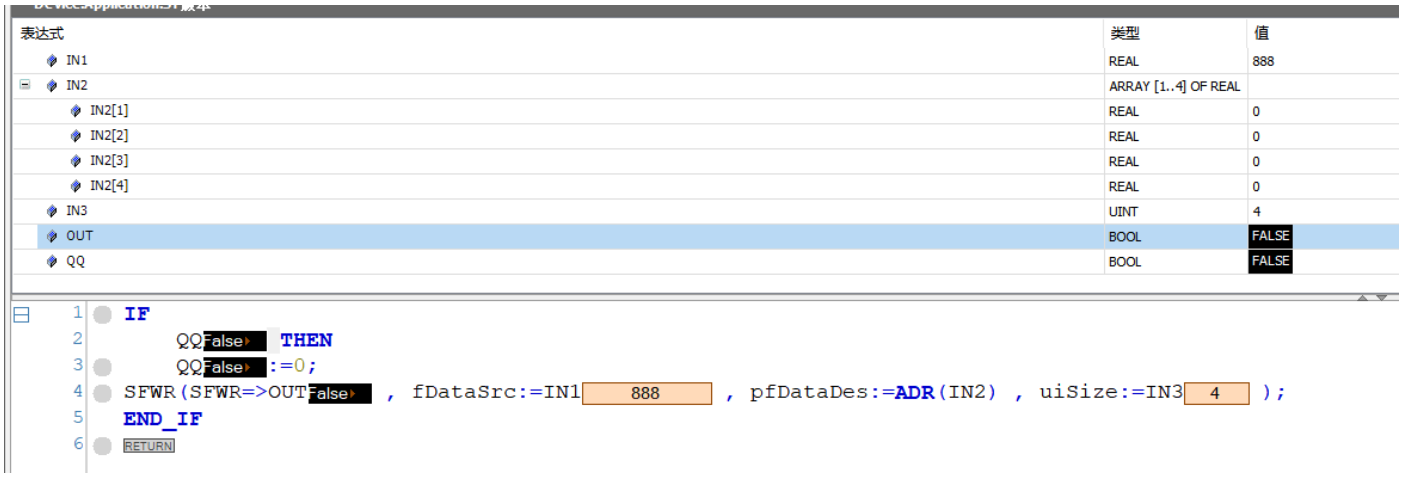


4.第三次触发

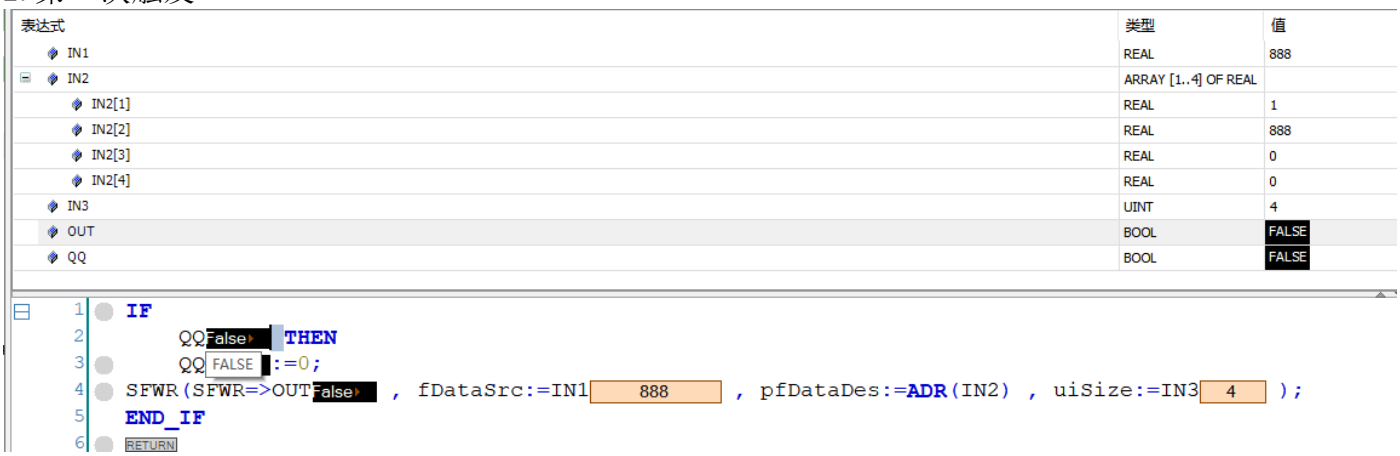


ST:

1. 初始状态



2. 第一次触发



3. 第二次触发

表达式	类型	值
IN1	REAL	888
IN2	ARRAY [1..4] OF REAL	
IN2[1]	REAL	2
IN2[2]	REAL	888
IN2[3]	REAL	888
IN2[4]	REAL	0
IN3	UINT	4
OUT	BOOL	FALSE
QQ	BOOL	FALSE

```
1  IF
2      QQFalse THEN
3      QQFalse:=0;
4  SFWR(SFWR=>OUTFalse , fDataSrc:=IN1 888 , pfDataDes:=ADR(IN2) , uiSize:=IN3 4 );
5  END_IF
6  RETURN
```

4.第三次触发

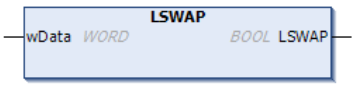
表达式	类型	值
IN1	REAL	888
IN2	ARRAY [1..4] OF REAL	
IN2[1]	REAL	3
IN2[2]	REAL	888
IN2[3]	REAL	888
IN2[4]	REAL	888
IN3	UINT	4
OUT	BOOL	FALSE
QQ	BOOL	FALSE

```
1  IF
2      QQFalse THEN
3      QQ FALSE :=0;
4  SFWR(SFWR=>OUTFalse , fDataSrc:=IN1 888 , pfDataDes:=ADR(IN2) , uiSize:=IN3 4 );
5  END_IF
6  RETURN
```

※注意事项

3.7.7 LSWAP　高低位数据交换

①指令格式

指令	功能	FB/FC	LD 表现	ST表现
LSWAP	高低位数据交换	FC		LSWAP(LSWAP=>OUT , wData:= IN1);

②相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述

wData	源数据	WORD	-	0	要写入的源数据

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
LSWAP	完成信号	BOOL	0-1	0	完成信号

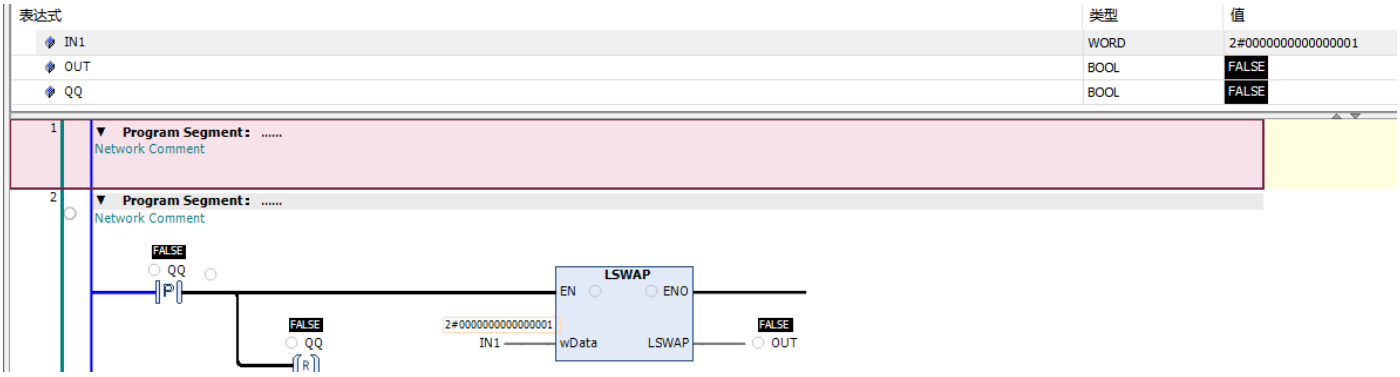
数据类型	布尔	位串				整数								实数		时刻、持续时间 日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATA	TOD	DT	STRING
wData			√																	
LSWAP	√																			

③程序示例

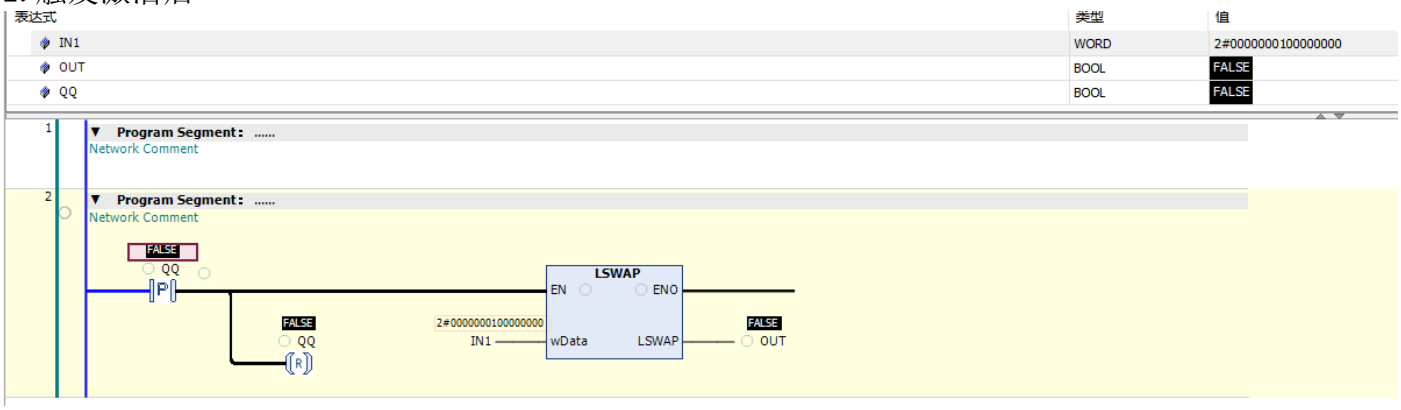
每次激活功能块都会将wData引脚的WORD类型值的低八位和高八位进行替换

LD:

1.初始状态

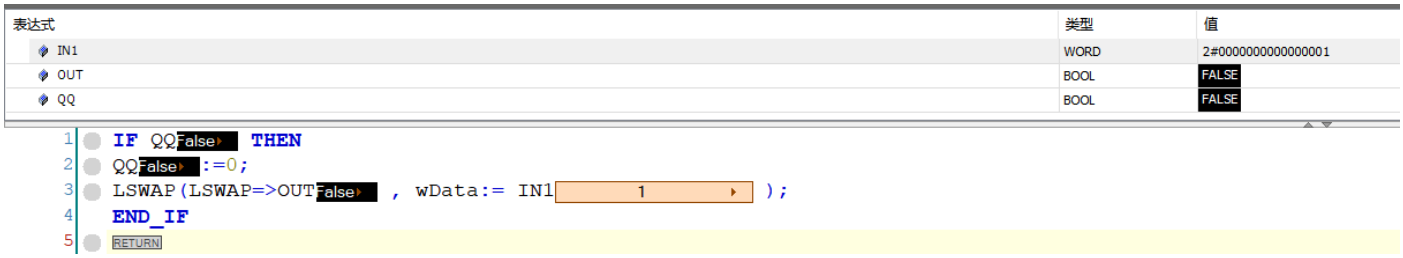


2.触发激活后

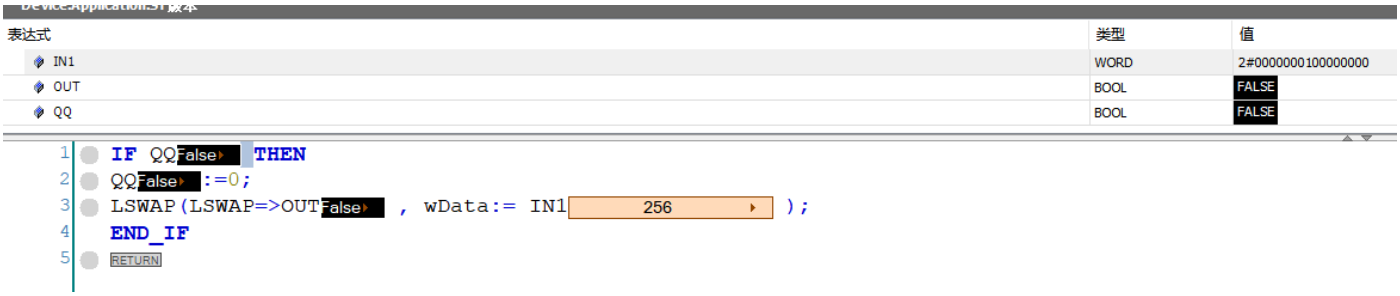


ST:

1. 初始状态



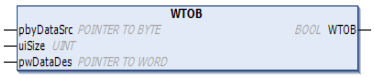
2. 触发激活后



※注意事项

3.7.8 WTOB 16 位数据拆分高低位

①指令格式

指令	功能	FB/FC	LD 表现	ST表现
WTOB	16 位数据拆分高低位	FC		WTOB(WTOB=>OUT , pbyDataSrc:=IN1 , uiSize:=IN2 , pwDataDes:=IN3);

②相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
pbyDataSrc	源数据	POINTER TO BYTE	-	0	被拆分的源数据
uiSize	拆分数量	UINT	-	0	要拆分的数量

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
pwDataDes	拆分后储存起始地址	POINTER TO WORD	-	-	拆分后储存起始地址

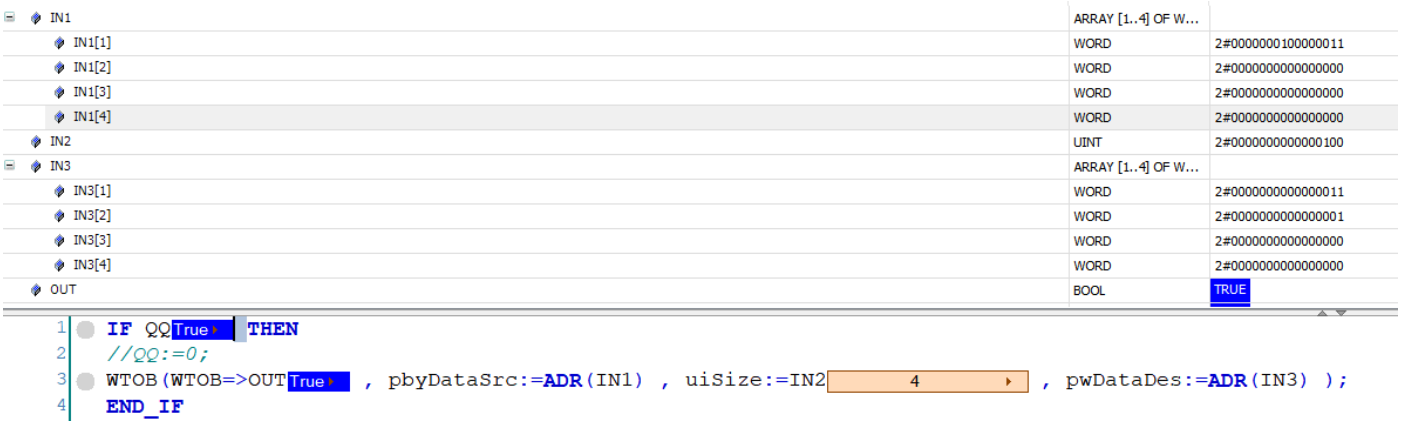
WTOB	完成信号	BOOL	-	0	完成信号

数据类型	布尔	位串				整数								实数		时刻、持续时间 日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATA	TOD	DT	STRING
pbyDataSrc		√																		
uiSize							√													
pwDataDes			√																	
WTOB	√																			

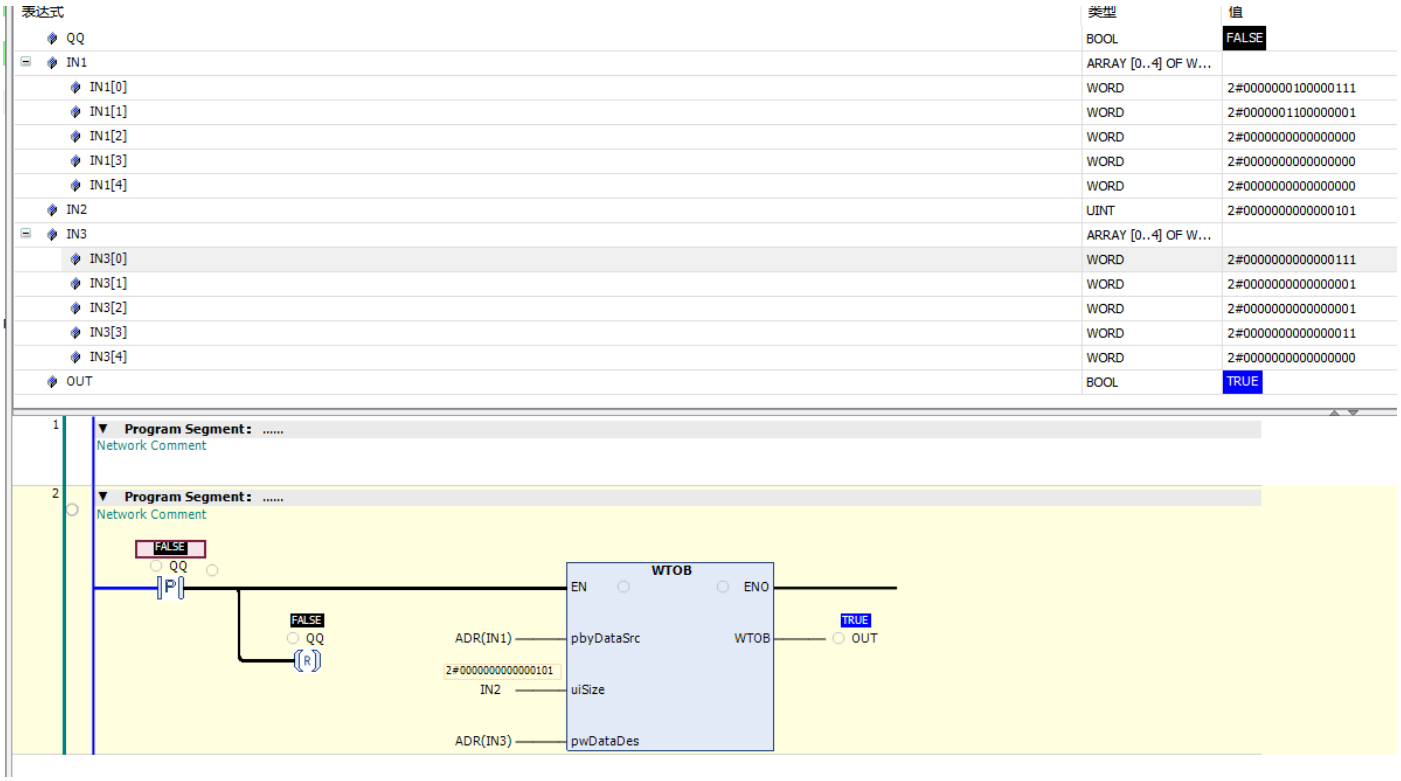
③程序示例

将pbyDataSrc指针指向的数据的高八位和低八位拆分出来给到两个pwDataDes变量储存起来

ST:



LD:

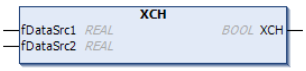


※注意事项

拆分出来的存储地址低8位在高八位的上面

3.7.9 XCH　将两个浮点数据进行交换

①指令格式

指令	功能	FB/FC	LD 表现	ST表现
XCH	将两个浮点数据进行交换	FC		XCH(XCH=>OUT , fDataSrc1:=IN1 , fDataSrc2:=IN2);

②相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
fDataSrc1	源数据1	REAL	-	0	要替换的数据1
FDataSrc2	源数据2	REAL	-	0	要替换的数据2

输出变量

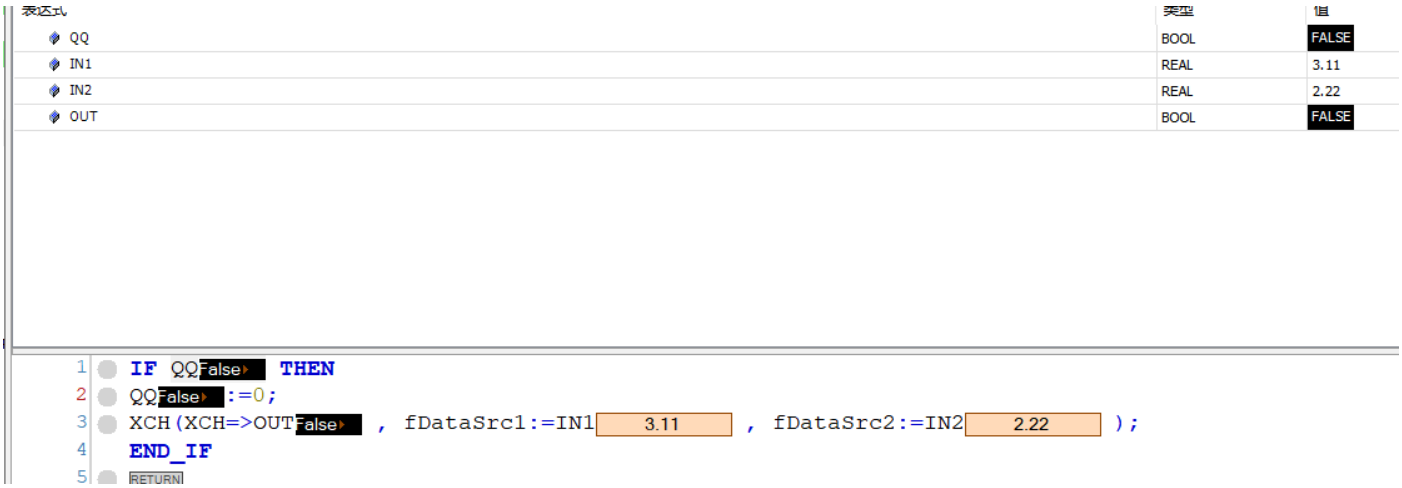
输出变量	名称	数据类型	有效范围	初始值	描述
XCH	完成信号	BOOL	-	0	完成信号

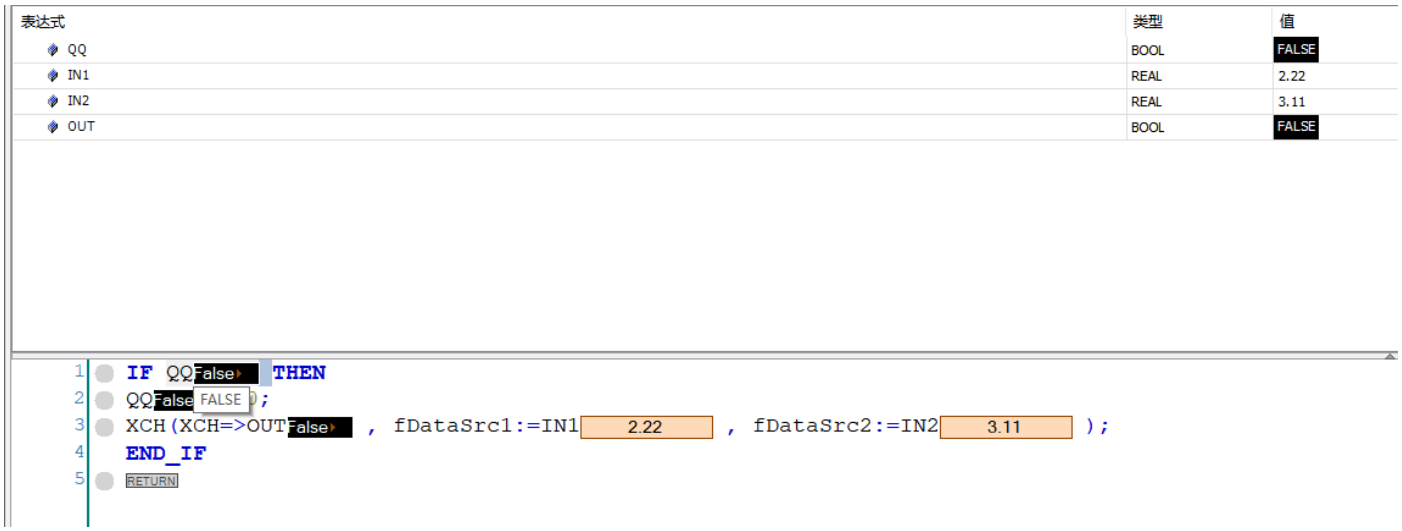
数据类型	布尔	位串				整数								实数		时刻、持续时间 日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATA	TOD	DT	STRING
fDataSrc1														√						
fDataSrc2														√						
XCH	√																			

③程序示例

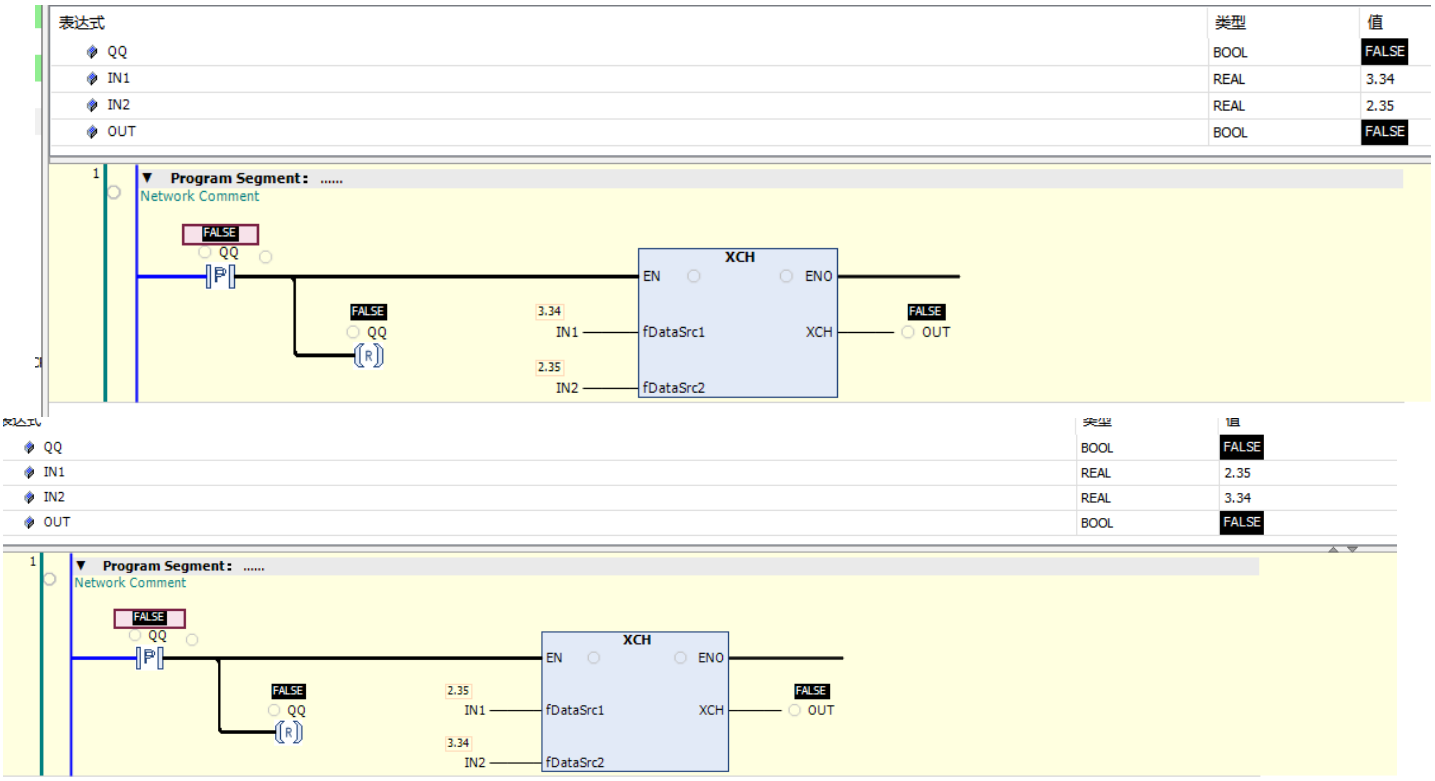
每次激活功能块将fDataSrc1与fDataSrc2内的数据进行替换

ST:





LD:



※注意事项

3.8 位操作相关

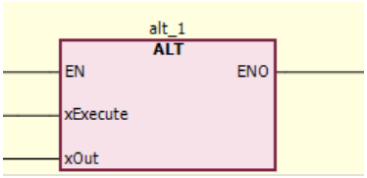
指令列表

指令类别	名称	FB/FC	功能
位操作	ALT	FB	交替输出
	BIT_AS_BYTE	FC	8bit 整合字节指令
	BIT_AS_DWORD	FC	32bit整合双字指令
	BIT_AS_WORD	FC	16bit整合字节指令

	BOUT	FC	位数据输出
	BRST	FC	位数据复位
	BYTE_AS_BIT	FC	位数据置位
	LGetBit	FC	判断一个16位整数某一位是否为1
	READ_NBit_Byte	FC	N位读取组BYTE
	READ_NBit_DWORD	FC	N位读取组DWORD
	READ_NBit_LWORD	FC	N位读取组LWORD
	READ_NBit_Word	FC	N位读取组Word
	WRITE_NBIT_BYTE	FC	将多个位写入位列中BYTE
	WRITE_NBIT_DWORD	FC	将多个位写入位列中DWORD
	WRITE_NBIT_LWORD	FC	将多个位写入位列中LWORD
	WRITE_NBIT_WORD	FC	将多个位写入位列中WORD
	WORD_AS_BIT	FC	字拆分指令

3.8.1 ALT 交替输出

①指令格式

指令	功能	FB/FC	LD 表现	ST表现
ALT	交替输出	FB		ALT(xExecute:= , xOut:=);

②相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
xExecute	输入信号	BOOL	[FALSE, TRUE]	FALSE	输入信号

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
xOut	输出信号	:BOOL	[FALSE, TRUE]	FALSE	输出信号

数据类型

	布尔	位串	整数	实数	时刻、持续时间 日期、字符串

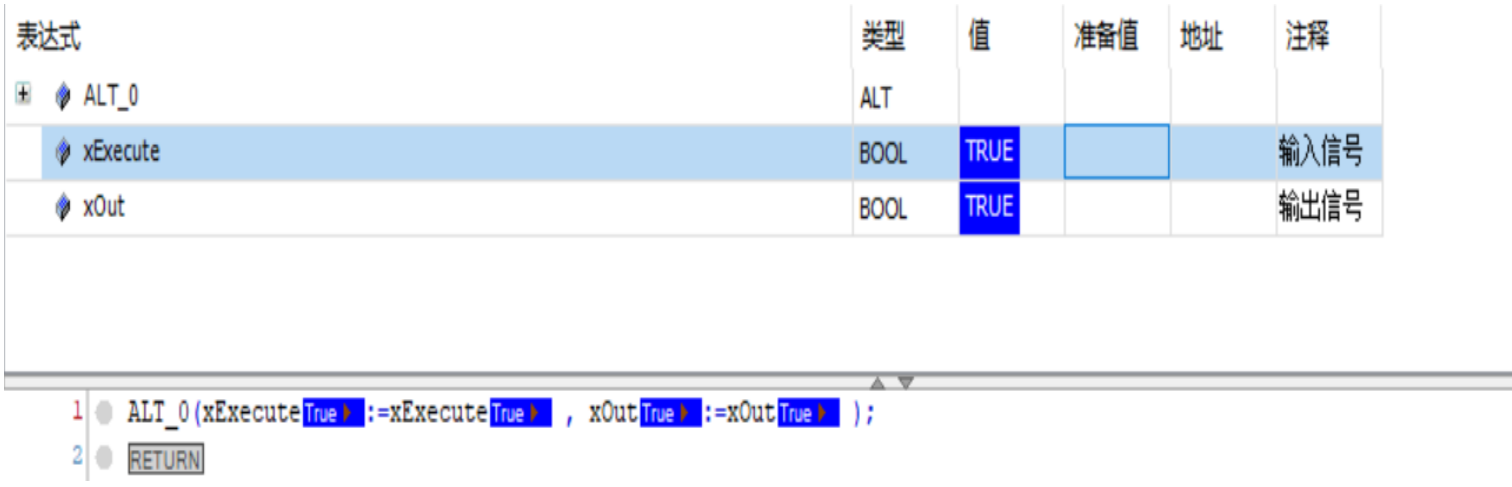
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
IxExecute	√																			
xOut	√																			

③功能说明

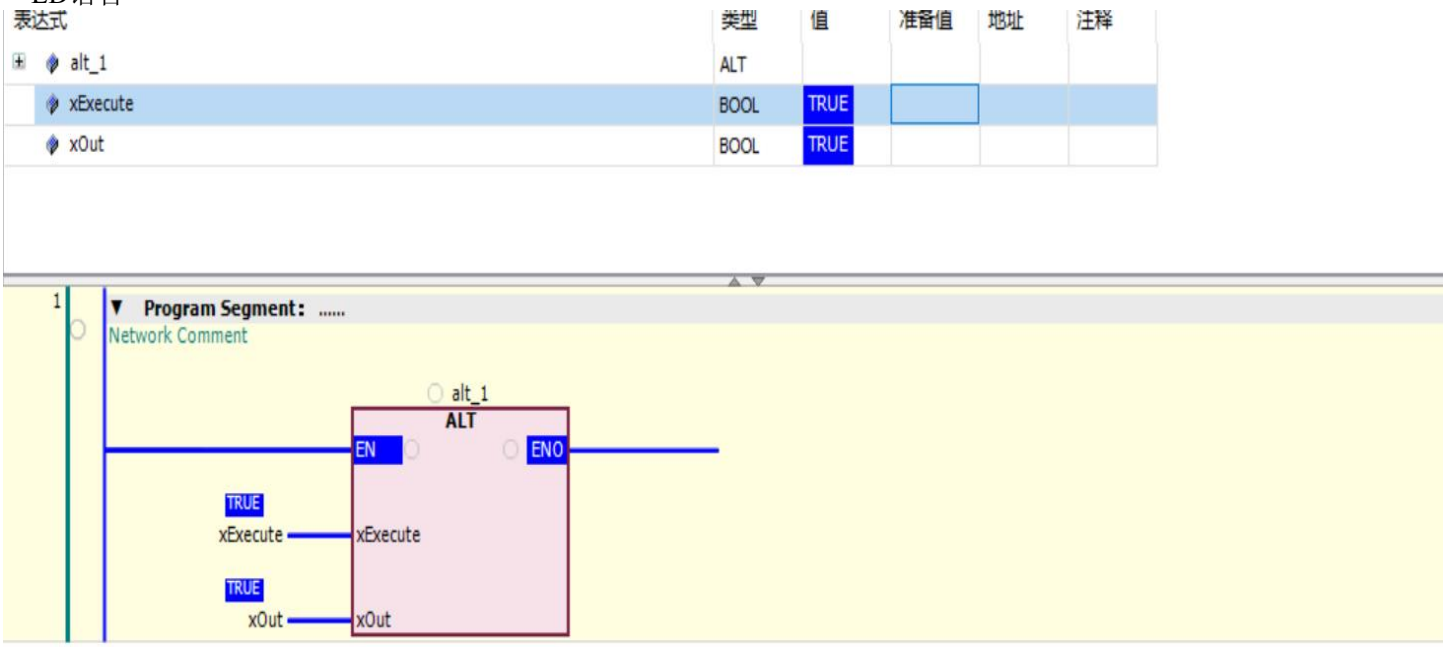
当ALT指令xExecute检测到上升沿时，xOut将反转其状态。

④程序示例

ST语言



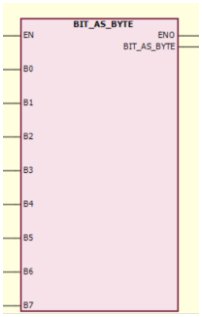
LD语言



3.8.2 BIT_AS_BYTE 8bit 整合字节指令

①指令格式

指令	功能	FB/FC	LD 表现	ST表现

BIT_AS_BYTE	8bit 整合字节指令	FC		BIT_AS_BYTE(BIT_AS_BYTE=> , B0:= , B1:= , B2:= , B3:= , B4:= , B5:= , B6:= , B7:=);

②相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
B0	Bit 0位	BOOL	[FALSE, TRUE]	FALSE	Bit 0位
B1	Bit 1位	BOOL	[FALSE, TRUE]	FALSE	Bit 1位
B2	Bit 2位	BOOL	[FALSE, TRUE]	FALSE	Bit 2位
B3:	Bit 3位	BOOL	[FALSE, TRUE]	FALSE	Bit 3位
B4	Bit 4位	BOOL	[FALSE, TRUE]	FALSE	Bit 4位
B5	Bit 5位	BOOL	[FALSE, TRUE]	FALSE	Bit 5位
B6	Bit 6位	BOOL	[FALSE, TRUE]	FALSE	Bit 6位
B7	Bit 7位	BOOL	[FALSE, TRUE]	FALSE	Bit 7位

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
BIT_AS_BYTE_1	返回值	BYTE	0~255	0	转换结果

数据类型

	布尔	位串				整数							实数		时刻、持续时间 日期、字符串					
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
B0~B7	√																			
BIT_AS_BYTE_1		√																		

③功能说明

将8个数据类型为BOOL的输入值转换为byte数据类型并从右侧输出端输出。

④程序示例

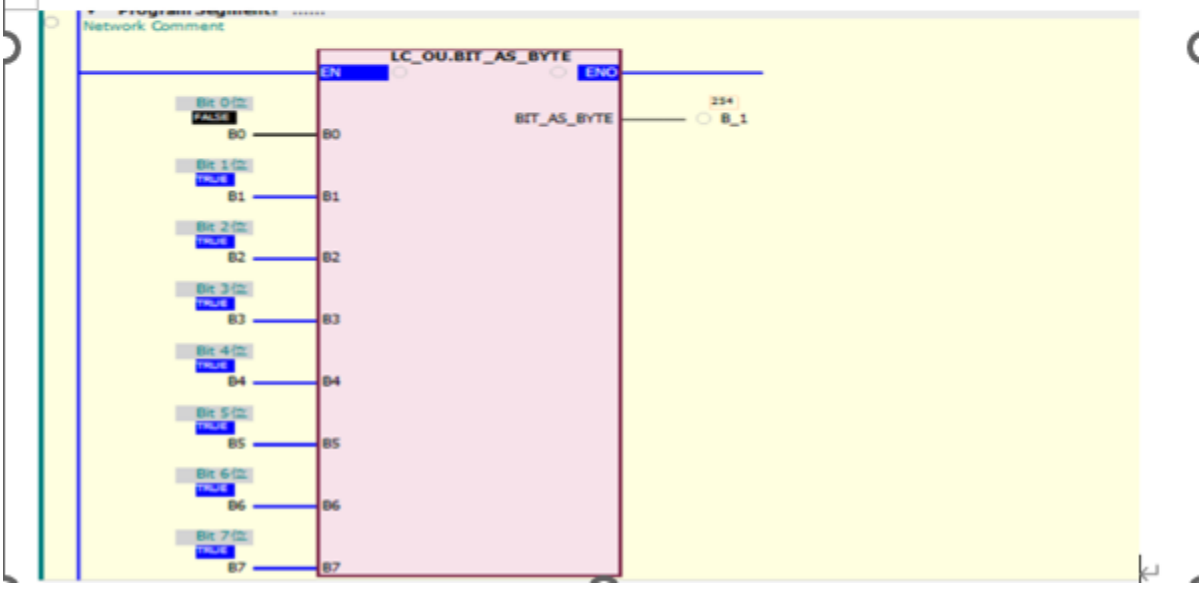
ST语言

表达式	类型	值	准备值	地址	注释
B0	BOOL	FALSE			Bit 0位
B1	BOOL	FALSE			Bit 1位
B2	BOOL	TRUE			Bit 2位
B3	BOOL	TRUE			Bit 3位
B4	BOOL	TRUE			Bit 4位
B5	BOOL	TRUE			Bit 5位
B6	BOOL	TRUE			Bit 6位
B7	BOOL	TRUE			Bit 7位
B_1	BYTE	252			

```
1   //ALT_0(xExecute:=xExecute , xOut:=xOut );
2   LC_OU.BIT_AS_BYTE(
3       BIT_AS_BYTE=>B_1 252 ,
4       B0:=B0 False ,
5       B1:= B1 False ,
6       B2:= B2 True ,
7       B3:=B3 True ,
8       B4:= B4 True ,
9       B5:=B5 True ,
10      B6:= B6 True ,
11      B7:= B7 True );RETURN
```

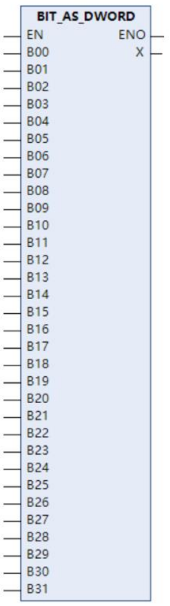
LD语言

表达式	类型	值	准备值	地址	注释
B0	BOOL	FALSE			Bit 0位
B1	BOOL	TRUE			Bit 1位
B2	BOOL	TRUE			Bit 2位
B3	BOOL	TRUE			Bit 3位
B4	BOOL	TRUE			Bit 4位
B5	BOOL	TRUE			Bit 5位
B6	BOOL	TRUE			Bit 6位
B7	BOOL	TRUE			Bit 7位
B_1	BYTE	254			



3.8.3 BIT_AS_DWORD 32bit整合双字指令

①指令格式

指令	功能	FB/FC	LD 表现	ST表现
BIT_AS_DWORD	32bit整合双字指令	FC		LC_OU.BIT_AS_DWORD(BIT_AS_DWORD=>X, B00:=, B01:= , B02:= , B03:= , B04:= , B05:= , B06:= , B07:= , B08:= ,B09:=,B10:= , B11:= , B12:= , B13:= , B14:= , B15:= , B16:=, B17:=, B18:=, B19:= , B20:= , B21:= , B22:= , B23:= , B24:= , B25:= , B26:= , B27:= , B28:= , B29:= , B30:= , B31:=);

②相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
xExecute	输入信号	BOOL	[FALSE, TRUE]	FALSE	输入信号

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
xOut	输出信号	:BOOL	[FALSE, TRUE]	FALSE	输出信号

数据类型

	布尔	位串				整数							实数		时刻、持续时间 日期、字符串					
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
IxExecute	√																			
xOut	√																			

③功能说明

当ALT指令xExecute检测到上升沿时，xOut将反转其状态。

④程序示例

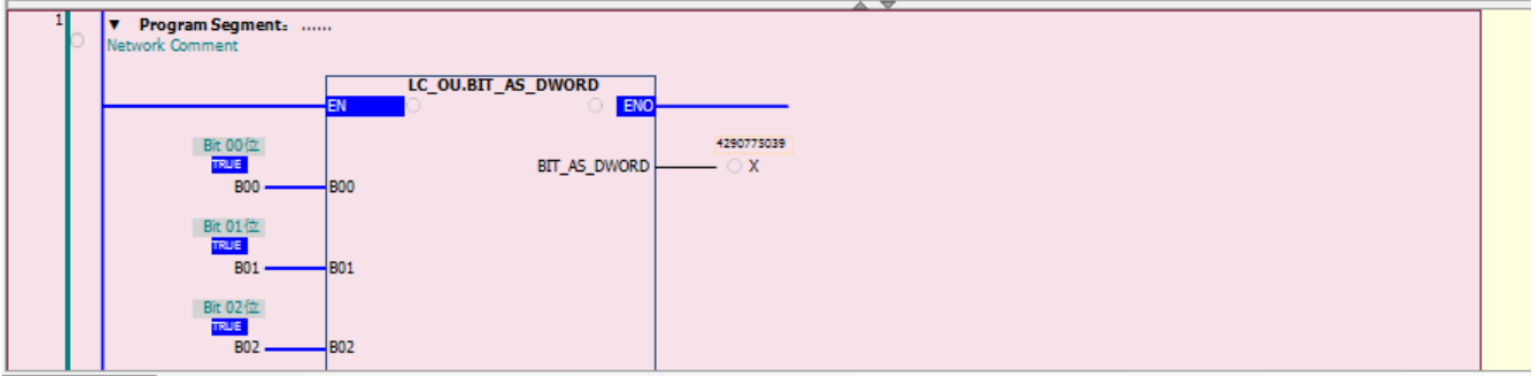
ST语言

表达式	类型	值	准备值	地址	注释
B24	BOOL	TRUE			Bit 24位
B25	BOOL	TRUE			Bit 25位
B26	BOOL	TRUE			Bit 26位
B27	BOOL	TRUE			Bit 27位
B28	BOOL	TRUE			Bit 28位
B29	BOOL	TRUE			Bit 29位
B30	BOOL	TRUE			Bit 30位
B31	BOOL	TRUE			Bit 31位
X	DWORD	429496...			

```
1  LC_OU.BIT_AS_DWORD(
2      BIT_AS_DWORD=>X 4294967295 ,
3      B00:= B00 True ,
4      B01:= B01 True ,
5      B02:= B02 True ,
6      B03:= B03 True ,
7      B04:= B04 True ,
8      B05:= B05 True ,
9      B06:= B06 True ,
10     B07:= B07 True ,
11     B08:= B08 True ,
12     B09:= B09 True ,
13     B10:= B10 True ,
14     B11:= B11 True ,
15     B12:= B12 True ,
```

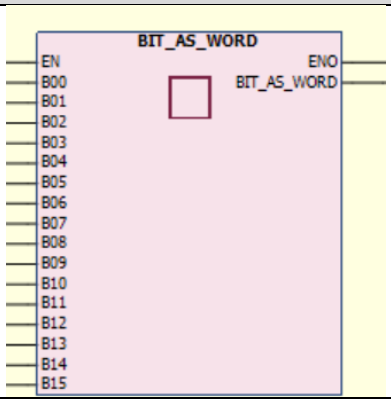
LD语言

表达式	类型	值	准备值	地址	注释
B22	BOOL	TRUE			Bit 22位
B23	BOOL	TRUE			Bit 23位
B24	BOOL	TRUE			Bit 24位
B25	BOOL	TRUE			Bit 25位
B26	BOOL	TRUE			Bit 26位
B27	BOOL	TRUE			Bit 27位
B28	BOOL	TRUE			Bit 28位
B29	BOOL	TRUE			Bit 29位
B30	BOOL	TRUE			Bit 30位
B31	BOOL	TRUE			Bit 31位
X	DWORD	429077...			



3.8.4 BIT_AS_WORD 16bit整合字节指令

①指令格式

指令	功能	FB/FC	LD 表现	ST表现
BIT_AS_WORD	16bit整 合字节指令	FC		LC_OU.BIT_AS_WORD(BIT_AS_WORD=>W , B00:= , B01:= , B02:= , B03:= , B04:= , B05:= , B06:= , B07:= , B08:= , B09:= , B10:= , B11:= , B12:= , B13:= , B14:= , B15:= ,);

②相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
B00-B15	数据源	BOOL	-	0	数据源

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
W	目标源	WORD	-	0	目标源

数据类型

	布尔	位串				整数							实数		时刻、持续时间 日期、字符串					
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
B00-B15	√																			
W			√																	

③功能说明

将输入端的16个bit结合成一个字从输出端输出。

④程序示例

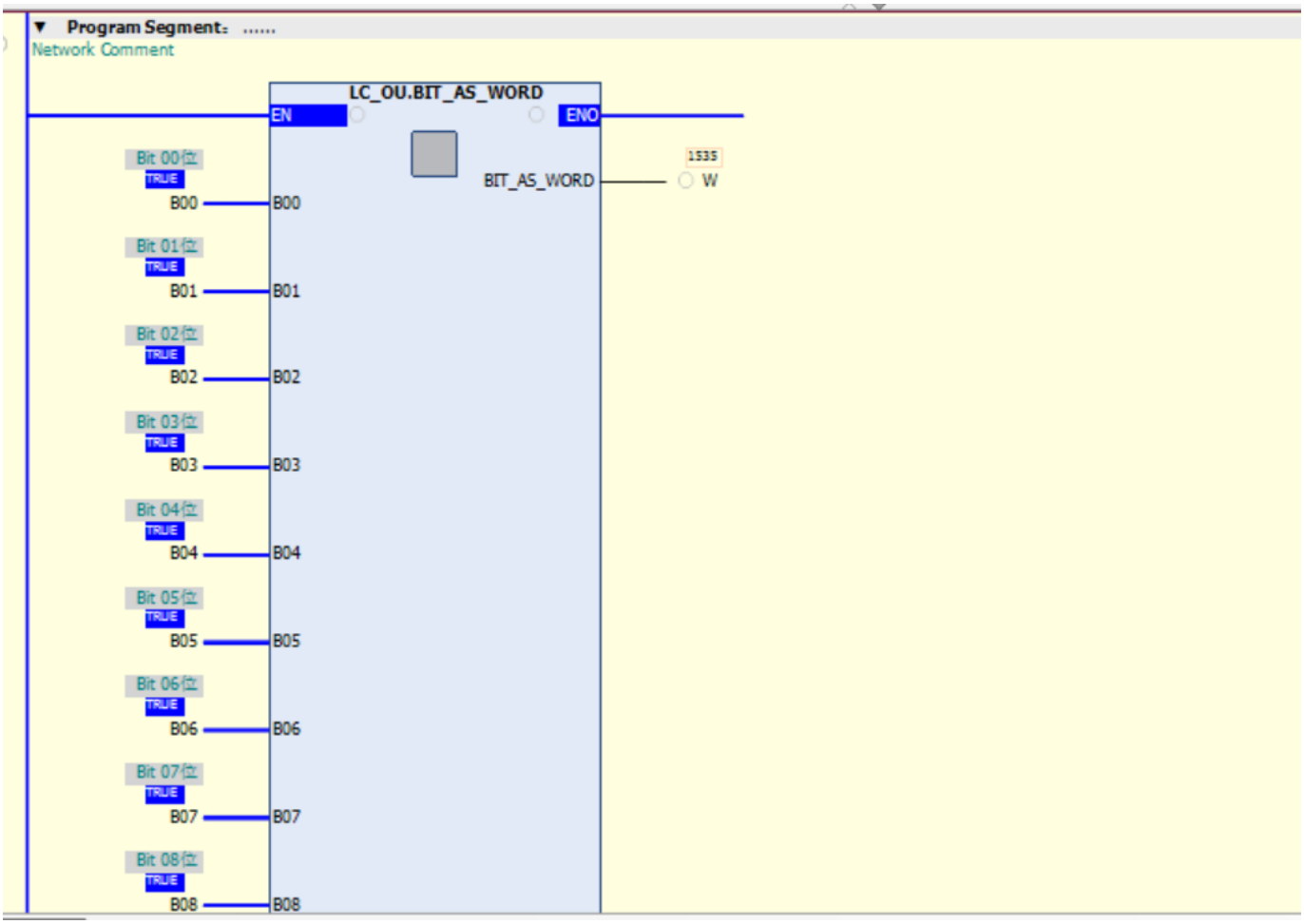
ST语言

表达式	类型	值	准备值	地址	注释
B00	BOOL	TRUE			Bit 00位
B01	BOOL	TRUE			Bit 01位
B02	BOOL	TRUE			Bit 02位
B03	BOOL	TRUE			Bit 03位
B04	BOOL	FALSE			Bit 04位
B05	BOOL	TRUE			Bit 05位
B06	BOOL	TRUE			Bit 06位
B07	BOOL	TRUE			Bit 07位
B08	BOOL	FALSE			Bit 08位
B09	BOOL	FALSE			Bit 09位
B10	BOOL	FALSE			Bit 10位
B11	BOOL	FALSE			Bit 11位
B12	BOOL	FALSE			Bit 12位
B13	BOOL	FALSE			Bit 13位
B14	BOOL	FALSE			Bit 14位
B15	BOOL	FALSE			Bit 15位
W	WORD	239			

```
LC_OU.BIT_AS_WORD(
    BIT_AS_WORD=>W 239 ,
    B00:= B00 True ,
    B01:= B01 True ,
    B02:= B02 True ,
    B03:= B03 True ,
    B04:= B04 False ,
    B05:= B05 True ,
    B06:= B06 True ,
    B07:= B07 True ,
    B08:= B08 False ,
    B09:= B09 False ,
    B10:= B10 False ,
    B11:= B11 False ,
    B12:= B12 False ,
    B13:= B13 False ,
    B14:= B14 False ,
    B15:= B15 False ,);
```

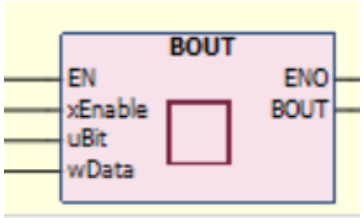
LD语言

表达式	类型	值	准备值	地址	注释
B00	BOOL	TRUE			Bit 00位
B01	BOOL	TRUE			Bit 01位
B02	BOOL	TRUE			Bit 02位
B03	BOOL	TRUE			Bit 03位
B04	BOOL	TRUE			Bit 04位
B05	BOOL	TRUE			Bit 05位
B06	BOOL	TRUE			Bit 06位
B07	BOOL	TRUE			Bit 07位
B08	BOOL	TRUE			Bit 08位
B09	BOOL	FALSE			Bit 09位
B10	BOOL	TRUE			Bit 10位
B11	BOOL	FALSE			Bit 11位
B12	BOOL	FALSE			Bit 12位
B13	BOOL	FALSE			Bit 13位
B14	BOOL	FALSE			Bit 14位
B15	BOOL	FALSE			Bit 15位
W	WORD	1535			



3.8.5 BOUT 位数据输出

①指令格式

指令	功能	FB/FC	LD 表现	ST表现
BOUT	位数据输出	FC		BOUT(BOUT=> , xEnable:= , uBit:=, wData:=);

②相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
xEnable	输入信号	BOOL	[FALSE, TRUE]	FALSE	电平触发有效开关
uBit	输入信号	UINT	遵照数据类型	0	需要操作的bit

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
wData	输出信号	WORD	遵照数据类型	0	需要操作的数据

数据类型

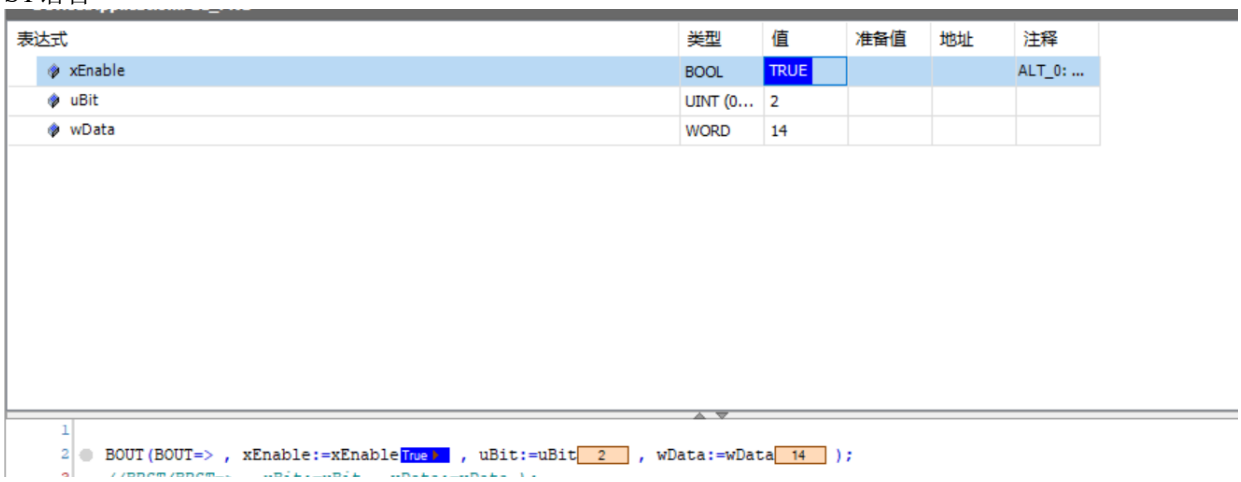
	布尔	位串				整数							实数		时刻、持续时间 日期、字符串					
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
IxExecute	√																			
uBit							√													
wData			√																	

③功能说明

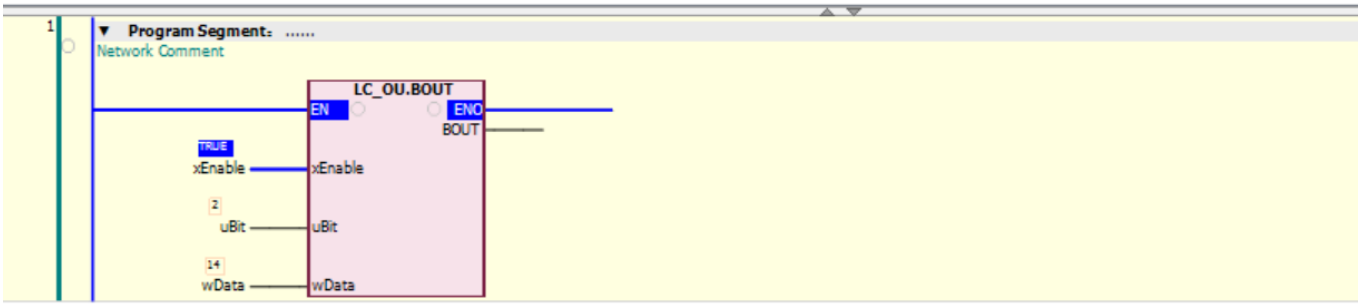
将wDate的初始值设为10（二进制#1010），当uiBit=2且xEnable =TRUE 情况如下，wData的bit2被置位， 结果wData=14（二进制2#1110）。

④程序示例

ST语言

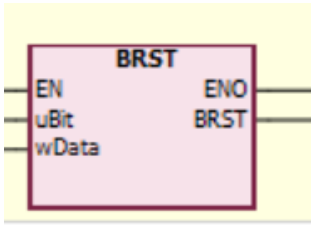


LD语言



3.8.6 BRST 位数据复位

①指令格式

指令	功能	FB/FC	LD 表现	ST表现
BRST	位数据复位	FC		BRST(BRST=> , uBit:= , wData:=);

②相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
uBit	输入信号	UINT	-	0	要置ON的bit位

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
wData	输出信号	WORD	遵照数据类型	0	需要操作的数据

数据类型

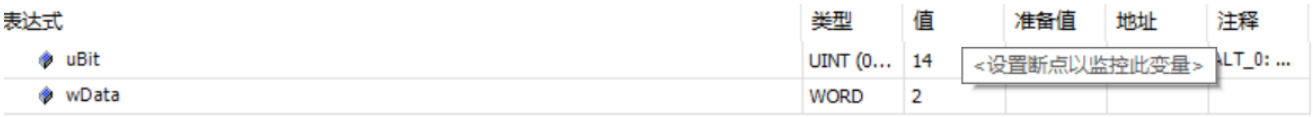
	布尔	位串				整数							实数		时刻、持续时间 日期、字符串					
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
uBit							√													
wData			√																	

③功能说明

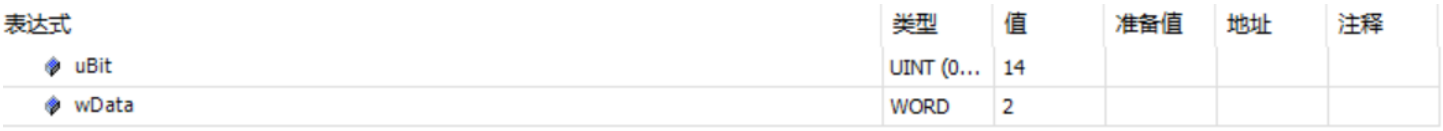
置位数据指定位,将被指定位设定为ON。不论后面BSET指令是否仍被驱动，设定位ON状态都会一直保持， 可利用BRST指令将该位设为OFF。

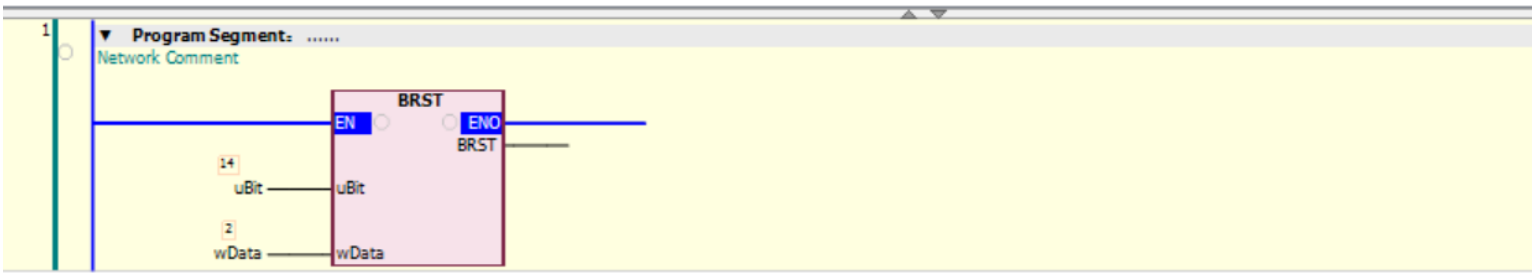
④程序示例

ST语言



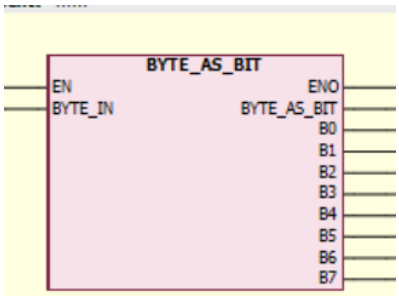
LD语言





3.8.7 BYTE_AS_BIT 位数据置位

①指令格式

指令	功能	FB/FC	LD 表现	ST表现
BYTE_AS_BIT	位数据置位	FC		BYTE_AS_BIT(BYTE_AS_BIT=> , BYTE_IN:=, B0=> , B1=> , B2=> , B3=> ,B4=> , B5=> , B6=> , B7=>);

②相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
BYTE_IN	返回值	BYTE	0-255	0	转换结果

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
B0-B7	输出信号	BOOL	TRUE-FALSE	FALSE	-

数据类型

	布尔	位串				整数							实数		时刻、持续时间 日期、字符串					
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
BYTE_IN		√																		
B0-B7	√																			

③功能说明

将数据类型为byte的输入值转换为数据类型为BOOL的8个值进行输出。

④程序示例

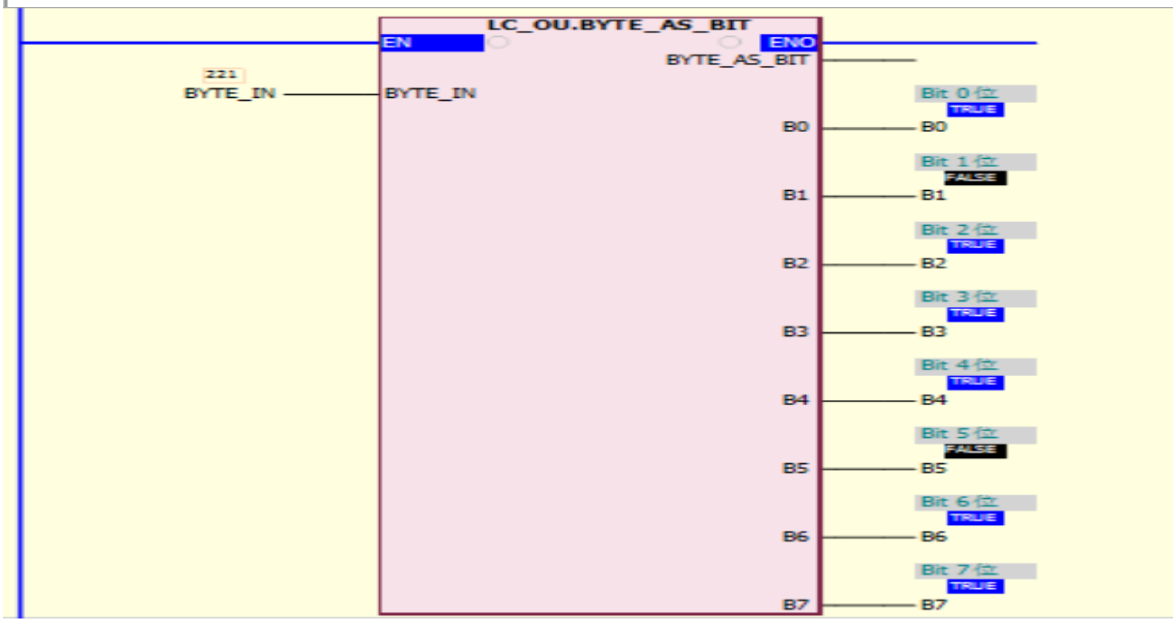
ST语言

表达式	类型	值	准备值	地址	注释
BYTE_IN	BYTE	221			ALT_0: ...
B0	BOOL	TRUE			Bit 0位
B1	BOOL	FALSE			Bit 1位
B2	BOOL	TRUE			Bit 2位
B3	BOOL	TRUE			Bit 3位
B4	BOOL	TRUE			Bit 4位
B5	BOOL	FALSE			Bit 5位
B6	BOOL	TRUE			Bit 6位
B7	BOOL	TRUE			Bit 7位

```
1  LC_OU.BYTE_AS_BIT(
2      BYTE_AS_BIT=> ,
3      BYTE_IN:= BYTE_IN221,
4      B0=> B0True,
5      B1=> B1False,
6      B2=>B2True ,
7      B3=>B3True ,
8      B4=> B4True,
9      B5=> B5False,
10     B6=>B6True ,
11     B7=> B7True);
12 //BRST(BRST=> , uBit:=uBit , xData:=xData );
```

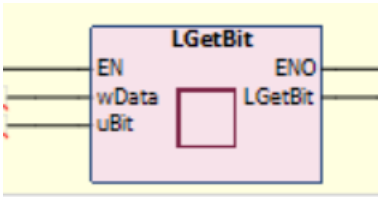
LD语言

表达式	类型	值	准备值	地址	注释
BYTE_IN	BYTE	221			
B0	BOOL	TRUE			Bit 0位
B1	BOOL	FALSE			Bit 1位
B2	BOOL	TRUE			Bit 2位
B3	BOOL	TRUE			Bit 3位
B4	BOOL	TRUE			Bit 4位
B5	BOOL	FALSE			Bit 5位
B6	BOOL	TRUE			Bit 6位
B7	BOOL	TRUE			Bit 7位



3.8.8 LGetBit 判断一个16位整数某一位是否为1

①指令格式

指令	功能	FB/FC	LD 表现	ST表现
LGetBit	判断一个16位整数某一位是否为1	FC		LGetBit(LGetBit=> , wData:= wData, uBit:= uBit);

②相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
xEnable	输入信号	BOOL	[FALSE, TRUE]	FALSE	电平触发有效开关

uBit	输入信号	UINT	遵照数据类型	0	需要操作的bit

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
wData	输出信号	WORD	遵照数据类型	0	需要操作的数据

数据类型

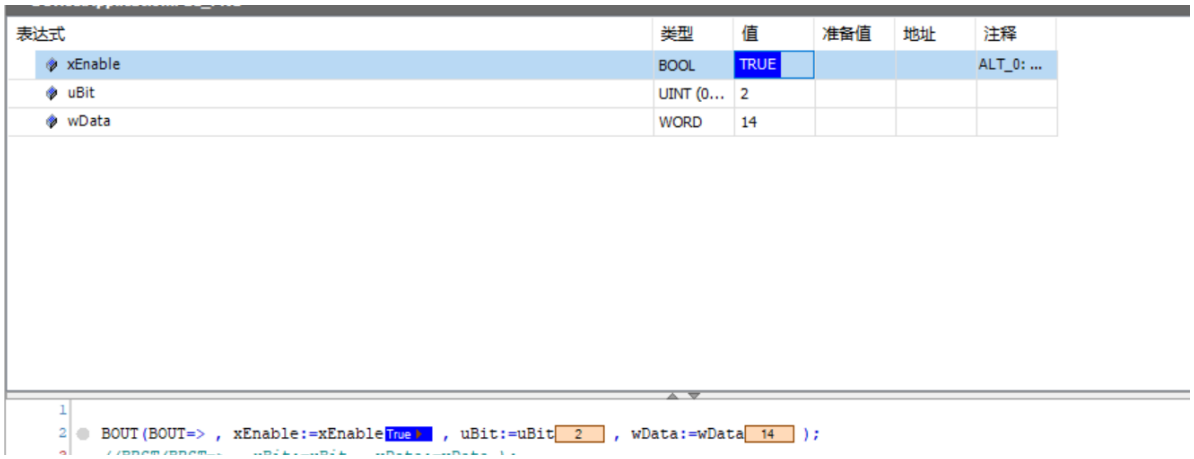
	布尔	位串				整数							实数		时刻、持续时间 日期、字符串					
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
IxExecute	√																			
uBit							√													
wData			√																	

③功能说明

将wDate的初始值设为10（二进制#1010），当uiBit=2且xEnable =TRUE 情况如下，wData的bit2被置位， 结果wData=14（二进制2#1110）。

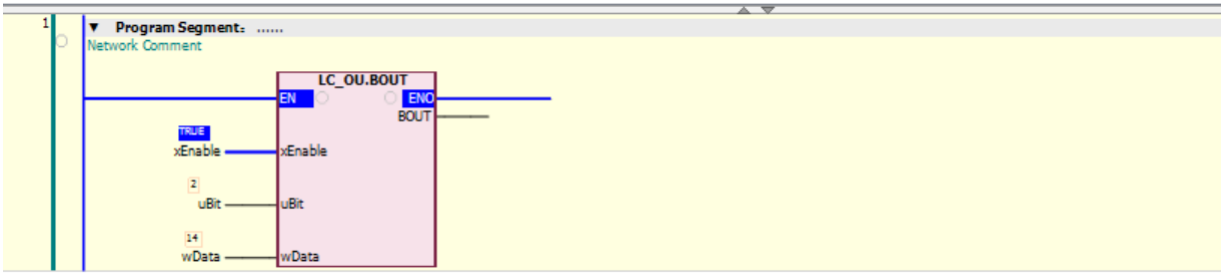
④程序示例

ST语言



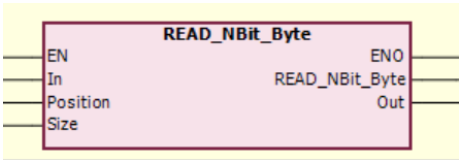
LD语言

表达式	类型	值	准备值	地址	注释
xEnable	BOOL	TRUE			
uBit	UINT (0...	2			
wData	WORD	14			



3.8.9 READ_NBit_Byte N位读取组BYTE

①指令格式

指令	功能	FB/FC	LD 表现	ST表现
READ_NBit_Byte	N位读取组BYTE	FC		READ_NBit_Byte(READ_NBit_Byte=> , In:= , Position:= , Size:= , Out=>);

②相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
In	输入BYTE类型数据	BYTE	遵照数据类型	–	输入BYTE类型数据
Position	读取位的位置	USINT	遵照数据类型	–	读取位的位置
Size	读取位的数量	USINT	遵照数据类型		读取位的数量

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Out	读取结果	BYTE	遵照数据类型	–	读取结果-

数据类型

	布尔	位串	整数	实数	时刻、持续时间 日期、字符串

	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In		√																		
Position						√														
Size						√														
Out		√																		

③功能说明

输入BYTE类型数据，读取位的位置，读取位的数量，读取结果。

④程序示例

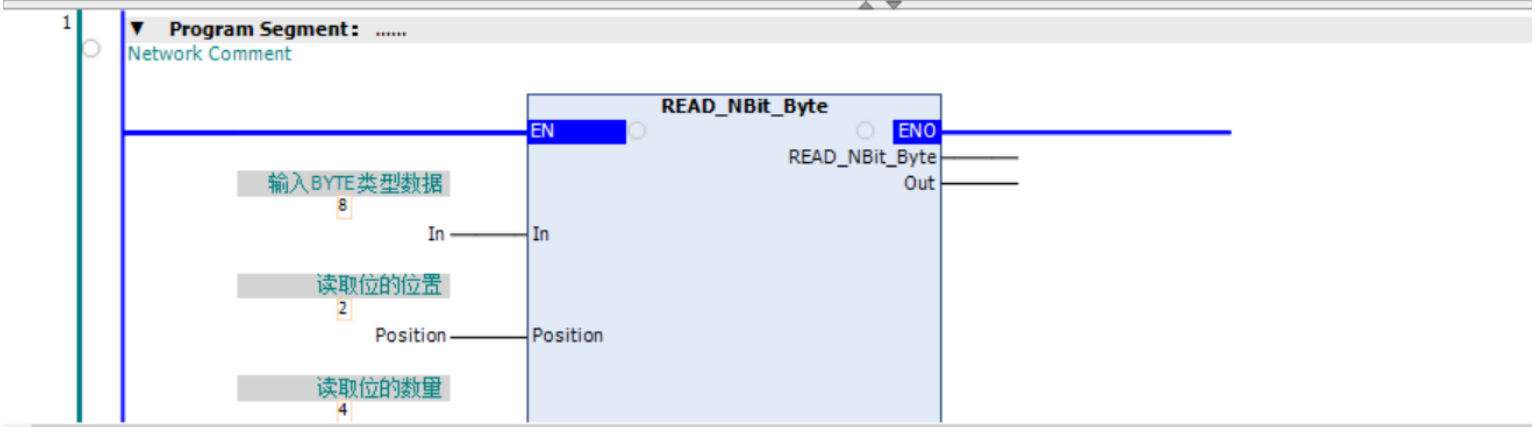
ST语言

表达式	类型	值	准备值	地址	注释
In	BYTE	16			输入 BY...
Position	USINT	3			读取位...
Size	USINT	4			读取位...
Out	BYTE	2			读取结...

```
READ_NBit_Byte(
    READ_NBit_Byte=> ,
    In:=In 16 ,
    Position:=Position 3 ,
    Size:=Size 4 ,
    Out=>Out 2 );
```

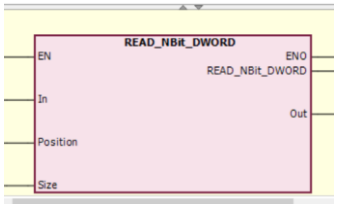
LD语言

表达式	类型	值	准备值	地址	注释
In	BYTE	8			输入 BY...
Position	USINT	2			读取位...
Size	USINT	4			读取位...
Out	BYTE	0			读取结...



3.8.10 READ_NBit_DWORD N位读取组DWORD

①指令格式

指令	功能	FB/FC	LD 表现	ST表现
READ_NBit_DWORD	N位读取组DWORD	FC		READ_NBit_DWORD(READ_NBit_DWORD=> , In:= , Position:= , Size:=, Out=>);

②相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
In	输入DWORD类型数据	DWORD	遵照数据类型	–	输入DWORD 类型数据
Position	读取位的位置	USINT	遵照数据类型	–	读取位的位置
Size	读取位的数量	USINT	遵照数据类型		读取位的数量

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Out	读取结果	DWORD	遵照数据类型	–	读取结果-

数据类型

	布尔	位串				整数							实数		时刻、持续时间 日期、字符串					
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In				√																
Position						√														
Size						√														
Out				√																

③功能说明

输入DWORD类型数据，读取位的位置，读取位的数量，读取结果。

④程序示例

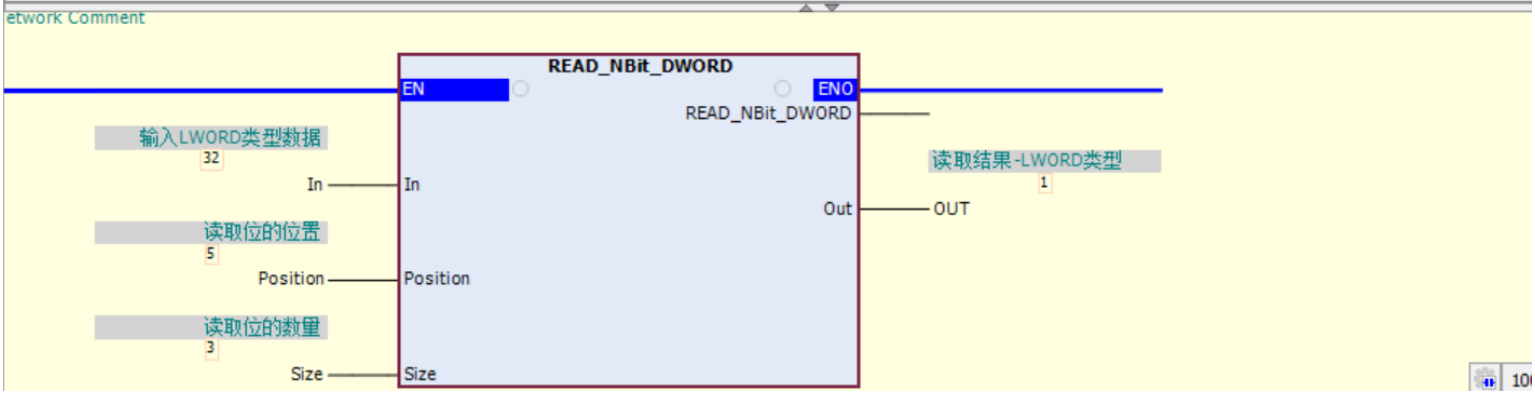
ST语言

表达式	类型	值	准备值	地址	注释
In	DWORD	32			输入 LW...
Position	USINT	5			读取位...
Size	USINT	3			读取位...
OUT	DWORD	1			读取结...

```
READ_NBit_DWORD(
    READ_NBit_DWORD=> ,
    In:=In 32 ,
    Position:= Position 5 ,
    Size:=Size 3 ,
    Out=>OUT 1 );
```

LD语言

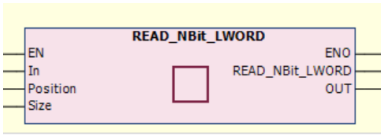
表达式	类型	值	准备值	地址	注释
In	DWORD	32			输入 LW...
Position	USINT	5			读取位...
Size	USINT	3			读取位...
OUT	DWORD	1			读取结...



3.8.11 READ_NBit_LWORD N位读取组LWORD

①指令格式

指令	功能	FB/FC	LD 表现	ST表现

READ_NBit_LWORD	N位读取组LWORD	FC		READ_NBit_LWORD(READ_NBit_LWORD=> , In:= , Position:= , Size:= , OUT=>);

②相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
In	输入 LWORD 类型数据	LWORD	遵照数据类型	–	输入 LWORD 类型数据
Position	读取位的位置	USINT	遵照数据类型	–	读取位的位置
Size	读取位的数量	USINT	遵照数据类型		读取位的数量

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Out	读取结果	LWORD	遵照数据类型	–	读取结果-

数据类型

	布尔	位串				整数							实数		时刻、持续时间 日期、字符串					
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In					√															
Position						√														
Size						√														
Out					√															

③功能说明

输入LWORD类型数据，读取位的位置，读取位的数量，读取结果。

④程序示例

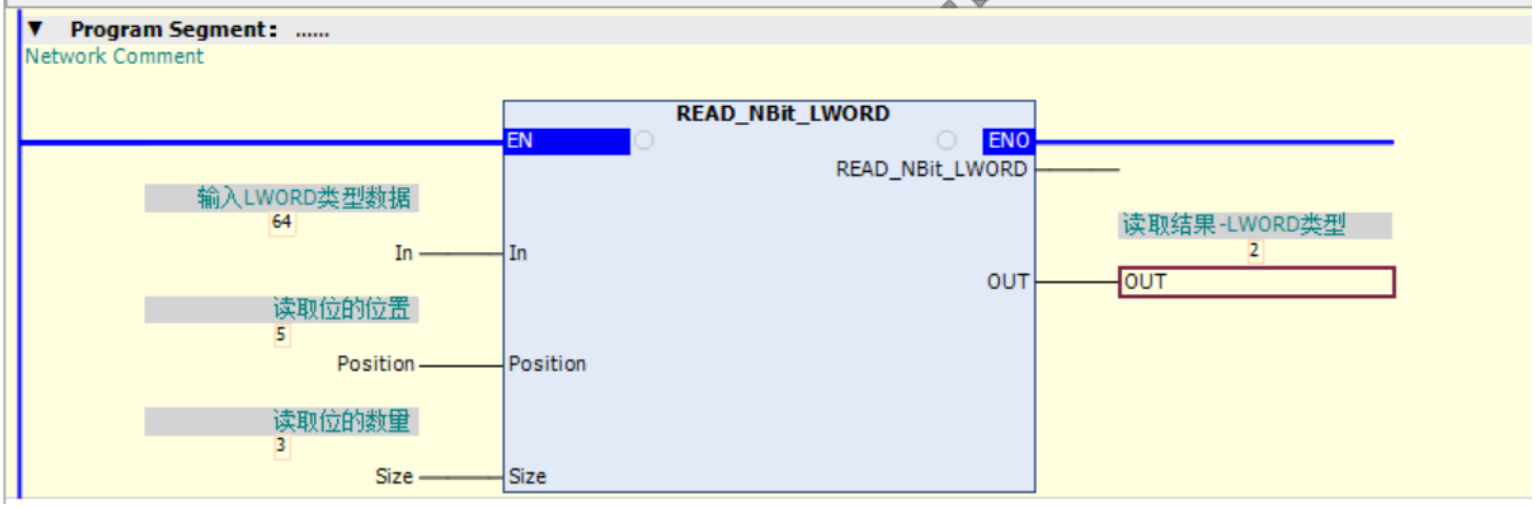
ST语言

表达式	类型	值	准备值	地址	注释
In	LWORD	64			输入LW...
Position	USINT	5			读取位...
Size	USINT	3			读取位...
OUT	LWORD	2			读取结...

```
READ_NBit_LWORD(
    READ_NBit_LWORD=> ,
    In:=In 64 ,
    Position:=Position 5 ,
    Size:= Size 3 ,
    OUT=>OUT 2 );
```

LD语言

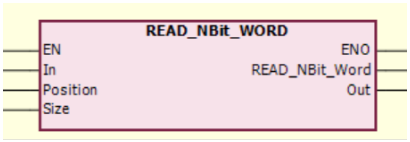
表达式	类型	值	准备值	地址	注释
In	LWORD	64			输入LW...
Position	USINT	5			读取位...
Size	USINT	3			读取位...
OUT	LWORD	2			读取结...



3.8.12 READ_NBit_Word N位读取组Word

①指令格式

指令	功能	FB/FC	LD 表现	ST表现

READ_NBit_Word	N位读取组Word	FC		READ_NBit_WORD(READ_NBit_Word=> , In:= , Position:= , Size:= , Out=>);

②相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
In	输入 Word 类型数据	BYTE	遵照数据类型	-	输入 Word 类型数据
Position	读取位的位置	USINT	遵照数据类型	-	读取位的位置
Size	读取位的数量	USINT	遵照数据类型	-	读取位的数量

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Out	读取结果	Word	遵照数据类型	-	读取结果-

数据类型

	布尔	位串				整数							实数		时刻、持续时间 日期、字符串					
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In			√																	
Position						√														
Size						√														
Out			√																	

③功能说明

输入Word类型数据，读取位的位置，读取位的数量，读取结果。

④程序示例

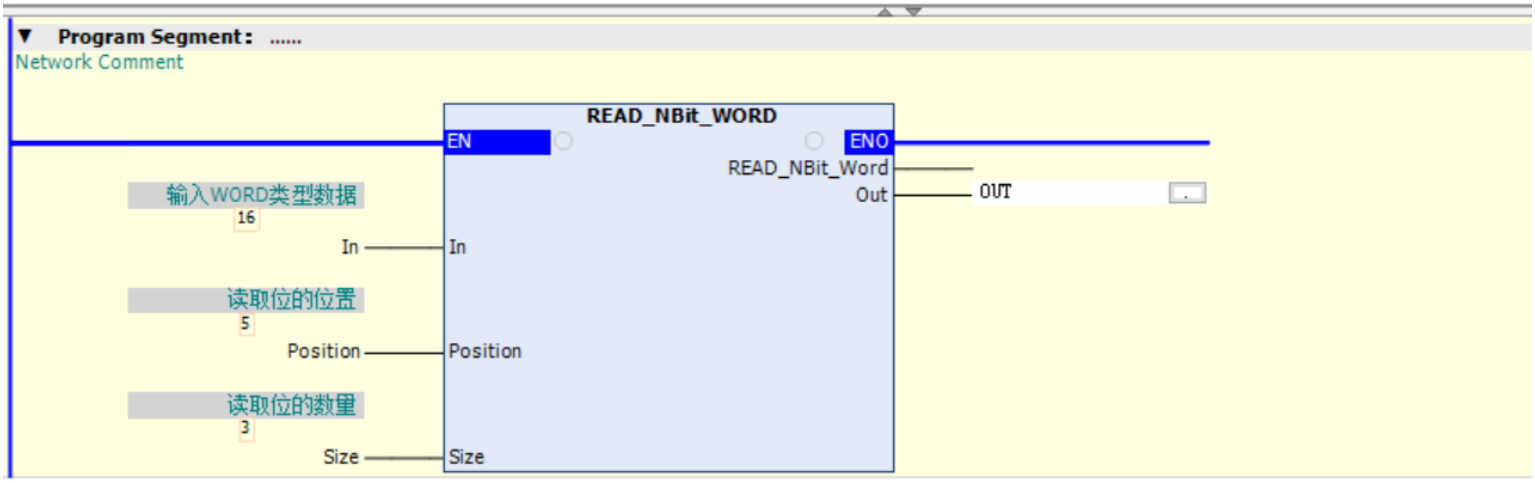
ST语言

表达式	类型	值	准备值	地址	注释
In	WORD	16			输入 W...
Position	USINT	5			读取位...
Size	USINT	3			读取位...
OUT	WORD	0			读取结...

```
1  READ_NBit_WORD(
2      READ_NBit_Word=> ,
3      In:= In 16 ,
4      Position:=Position 5 ,
5      Size:= Size 3 ,
6      Out=> OUT 0 );
```

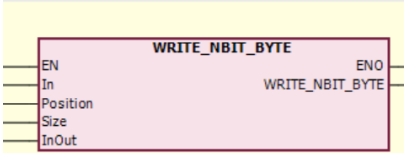
LD语言

表达式	类型	值	准备值	地址	注释
In	WORD	16			输入 W...
Position	USINT	5			读取位...
Size	USINT	3			读取位...
OUT	WORD	0			读取结...



3.8.13 WRITE_NBIT_BYTE 将多个位写入位列中BYTE

①指令格式

指令	功能	FB/FC	LD 表现	ST表现
WRITE_NBIT_BYTE	将多个位写入位列中BYTE	FC		WRITE_NBIT_BYTE(WRITE_NBIT_BYTE=> , In:= , Position:= , Size:= , InOut:=);

②相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
In	输入BYTE类型数据	BYTE	遵照数据类型	-	输入BYTE类型数据
Position	写入位的位置（写到InOut中的位置）	USINT	遵照数据类型	-	写入位的位置（写到InOut中的位置）
Size	写入位的数量（从输入In的低位算起）	USINT	遵照数据类型		写入位的数量（从输入In的低位算起）

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
InOut	输入输出BYTE类型数据	BYTE	遵照数据类型	-	输入输出BYTE类型数据

数据类型

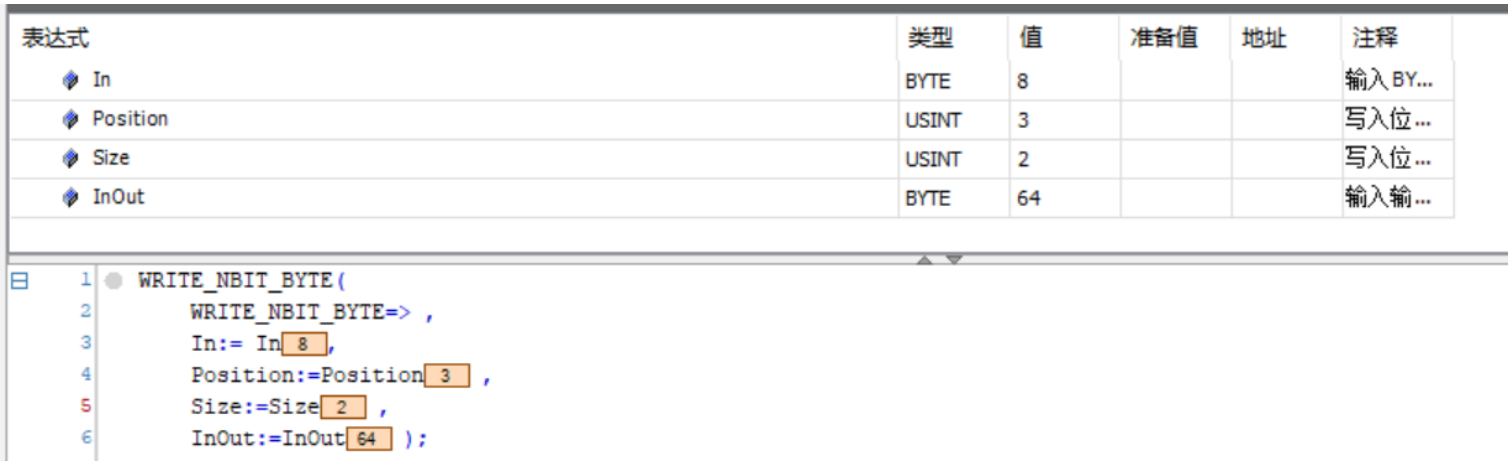
	布尔	位串				整数							实数		时刻、持续时间 日期、字符串					
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In		√																		
Position						√														
Size						√														
InOut		√																		

③功能说明

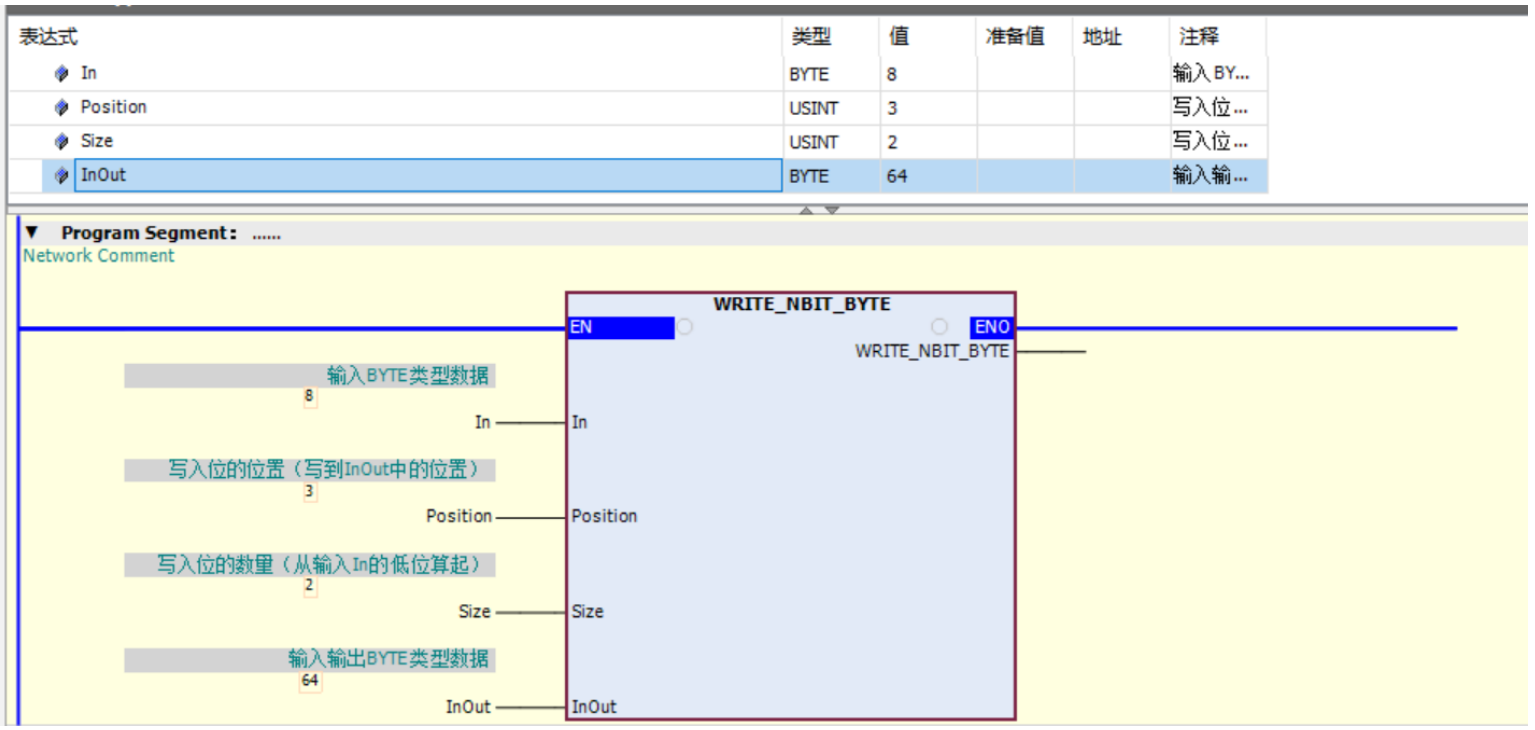
输入BYTE类型数据16#FF，写入位的位置（写到InOut中的位置），写入位的数量（从输入In的低位算起），输入输出BYTE类型数据。

④程序示例

ST语言

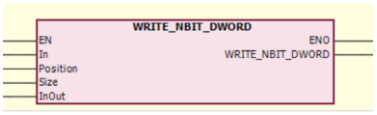


LD语言



3.8.14 WRITE_NBIT_DWORD 将多个位写入位列中DWORD

①指令格式

指令	功能	FB/FC	LD 表现	ST表现
WRITE_NBIT_DWORD	将多个位写入位列中 DWORD	FC		WRITE_NBIT_DWORD(WRITE_NBIT_DWORD=> , In:= , Position:= , Size:= , InOut:=);

②相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
In	输入 DWORD 类型数据	DWORD	遵照数据类型	-	输入 DWORD 类型数据
Position	读取位的位置	USINT	遵照数据类型	-	读取位的位置
Size	读取位的数量	USINT	遵照数据类型		读取位的数量

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
InOut	读取结果	DWORD	遵照数据类型	-	读取结果-

数据类型

	布尔	位串				整数							实数		时刻、持续时间 日期、字符串					
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In				√																
Position						√														
Size						√														
InOut				√																

③功能说明

输入DWORD类型数据16#FFFFFFFF，写入位的位置（写到InOut中的位置），写入位的数量（从输入In的低位算起），输入输出DWORD类型数据。

④程序示例

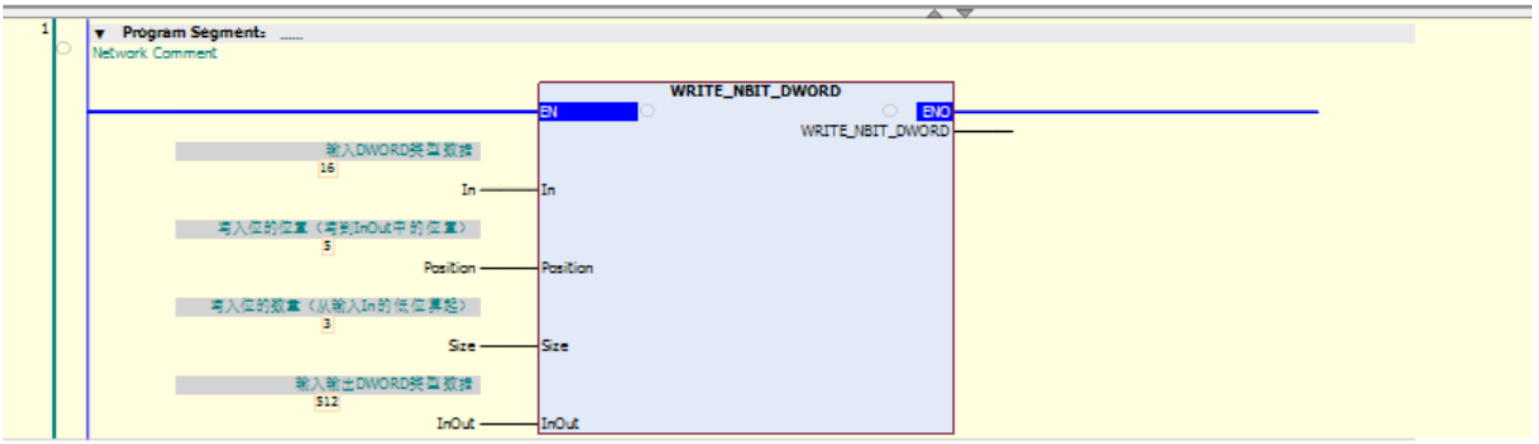
ST语言

表达式	类型	值	准备值	地址	注释
In	DWORD	16			输入 DW...
Position	USINT	5			写入位...
Size	USINT	3			写入位...
InOut	DWORD	512			输入输...

```
1  WRITE_NBIT_DWORD(
2      WRITE_NBIT_DWORD=> ,
3      In:= In 16,
4      Position:=Position 5 ,
5      Size:= Size 3,
6      InOut:=InOut 512 );
```

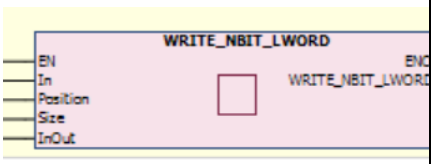
LD语言

表达式	类型	值	准备值	地址	注释
In	DWORD	16			输入 DW...
Position	USINT	5			写入位...
Size	USINT	3			写入位...
InOut	DWORD	512			输入输...



3.8.15 WRITE_NBIT_LWORD 将多个位写入位列中LWORD

①指令格式

指令	功能	FB/FC	LD 表现	ST表现
WRITE_NBIT_LWORD	将多个位写入位列中LWORD	FC		WRITE_NBIT_LWORD(WRITE_NBIT_LWORD=> , In:= , Position:= , Size:= , InOut:=);

②相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
In	输入 LWORD 类型数据	LWORD	遵照数据类型	-	输入 LWORD 类型数据
Position	读取位的位置	USINT	遵照数据类型	-	读取位的位置
Size	读取位的数量	USINT	遵照数据类型		读取位的数量

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
InOut	读取结果	LWORD	遵照数据类型	-	读取结果-

数据类型

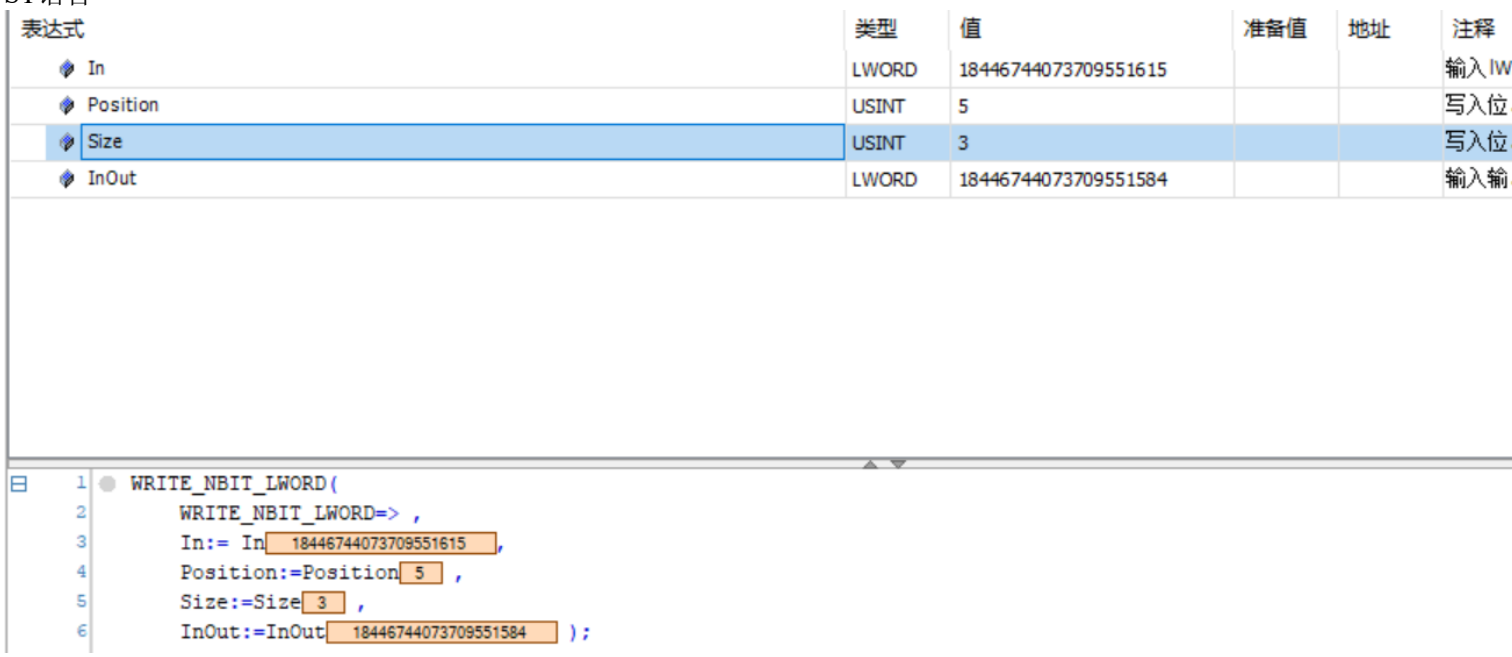
	布尔	位串				整数							实数		时刻、持续时间 日期、字符串					
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In:					√															
Position						√														
Size						√														
InOut					√															

③功能说明

输入LWORD类型数据16#FFFFFFFFFFFFFFFF，写入位的位置（写到InOut中的位置），写入位的数量（从输入In的低位算起），输入输出LWORD类型数据。

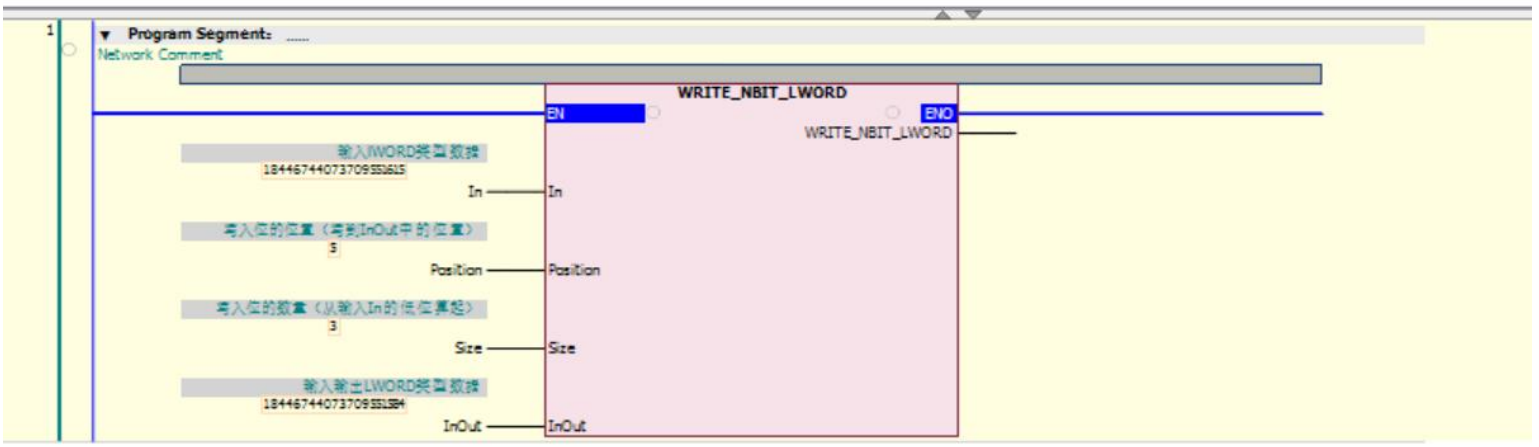
④程序示例

ST语言



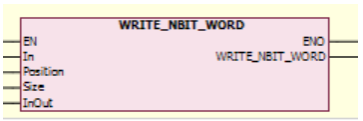
LD语言

In	LWORD	18446744073709551615	
Position	USINT	5	
Size	USINT	3	
InOut	LWORD	18446744073709551584	



3.8.16 WRITE_NBIT_WORD 将多个位写入位列中WORD

①指令格式

指令	功能	FB/FC	LD 表现	ST表现
WRITE_NBIT_WORD	将多个位写入位列中WORD	FC		WRITE_NBIT_WORD(WRITE_NBIT_WORD=> , In:= , Position:= , Size:= , InOut:=);

②相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
In	输入 WORD 类型数据	WORD	遵照数据类型	–	输入 WORD 类型数据
Position	读取位的位置	USINT	遵照数据类型	–	读取位的位置
Size	读取位的数量	USINT	遵照数据类型		读取位的数量

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述

InOut	读取结果	WORD	遵照数据类型	-	读取结果-

数据类型

	布尔	位串				整数							实数		时刻、持续时间 日期、字符串					
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In:			√																	
Position						√														
Size						√														
InOut			√																	

③功能说明

输入WORD类型数据16#FFFF，写入位的位置（写到InOut中的位置），写入位的数量（从输入In的低位算起），输入输出WORD类型数据。

④程序示例

ST语言

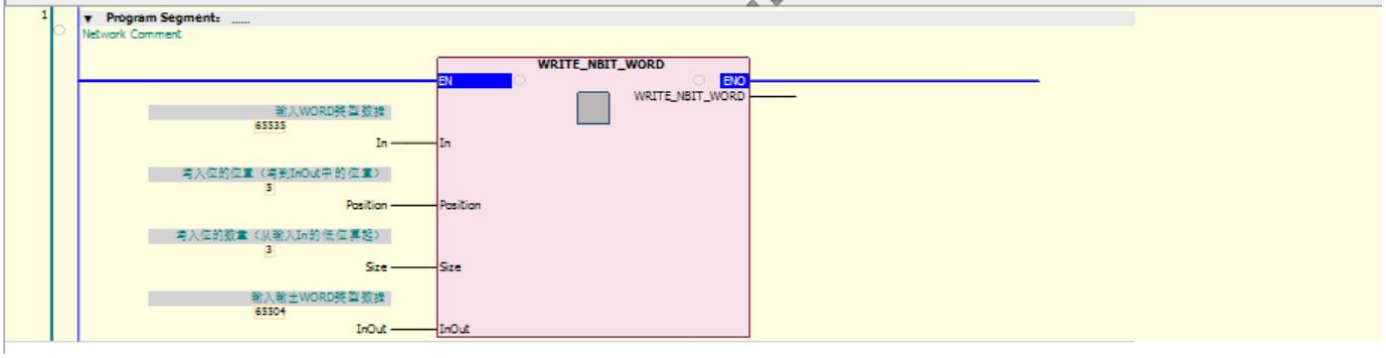
表达式	类型	值	准备值	地址	注释
In	WORD	65535			输入W...
Position	USINT	5			写入位...
Size	USINT	3			写入位...
InOut	WORD	65504			输入输...

```
1  WRITE_NBIT_WORD(
2      WRITE_NBIT_WORD=> ,
3      In:=In 65535 ,
4      Position:=Position 5 ,
5      Size:=Size 3 ,
6      InOut:=InOut 65504 );
```

LD语言

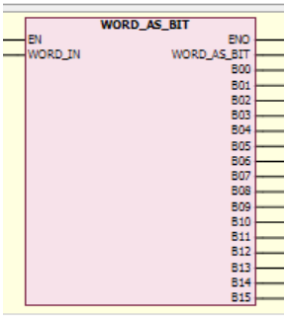
Device.Application.POU

表达式	类型	值	准备值	地址	注释
In	WORD	65535			输入W...
Position	USINT	5			写入位...
Size	USINT	3			写入位...
InOut	WORD	65504			输入输...



3.8.17 WORD_AS_BIT 字拆分指令

①指令格式

指令	功能	FB/FC	LD 表现	ST表现
WORD_AS_BIT	字拆分指令	FC		WORD_AS_BIT(WORD_AS_BIT=> , WORD_IN:= , B00=> , B01=> , B02=> , B03=> , B04=> , B05=> , B06=> ,B07=> , B08=> , B09=> , B10=> , B11=> , B12=> , B13=> , B14=> , B15=>);

②相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
WORD_IN	输入 WORD 类型数据	WORD	遵照数据类型	–	输入 WORD 类型数据

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
B00-B15	读取结果	BOOL	遵照数据类型	–	读取结果-

数据类型

	布尔	位串				整数							实数		时刻、持续时间 日期、字符串					
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
WORD_IN			√																	
B00-B15	√																			

③功能说明

输入WORD类型数据拆分后得到结果。

④程序示例

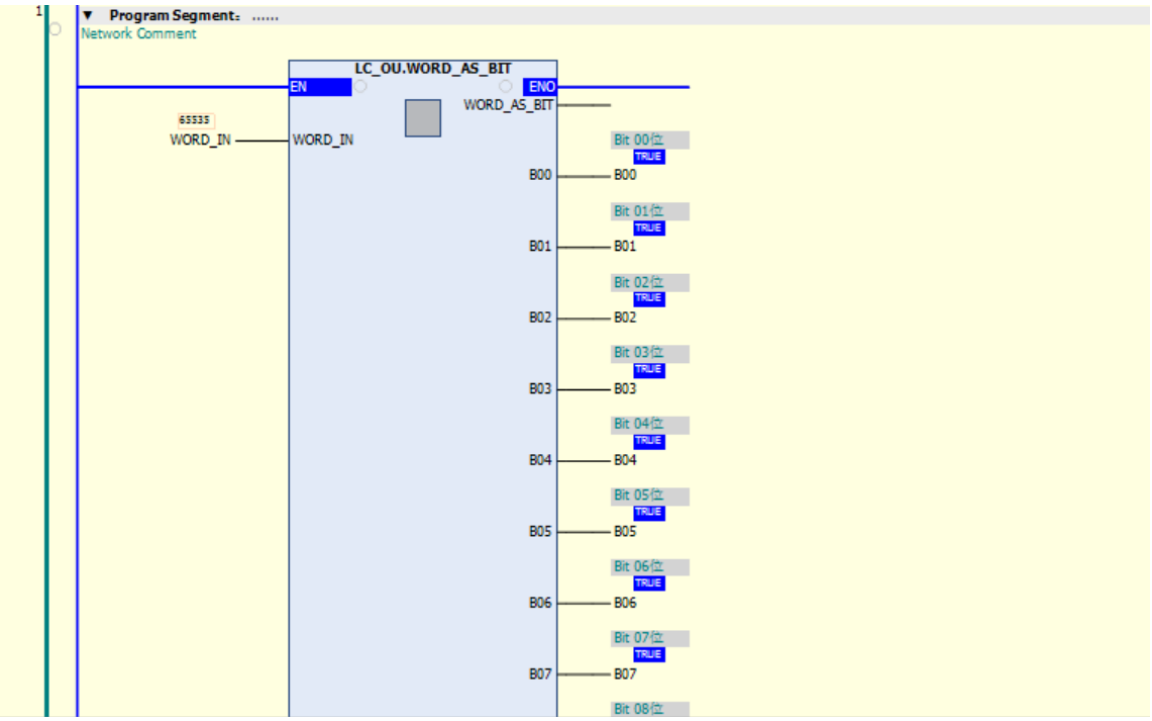
ST语言

表达式	类型	值	准备值	地址	注释
B00	BOOL	TRUE			Bit 00位
B01	BOOL	TRUE			Bit 01位
B02	BOOL	TRUE			Bit 02位
B03	BOOL	TRUE			Bit 03位
B04	BOOL	TRUE			Bit 04位
B05	BOOL	TRUE			Bit 05位
B06	BOOL	TRUE			Bit 06位
B07	BOOL	TRUE			Bit 07位
B08	BOOL	TRUE			Bit 08位
B09	BOOL	TRUE			Bit 09位
B10	BOOL	TRUE			Bit 10位
B11	BOOL	TRUE			Bit 11位
B12	BOOL	TRUE			Bit 12位
B13	BOOL	TRUE			Bit 13位
B14	BOOL	TRUE			Bit 14位
B15	BOOL	TRUE			Bit 15位
WORD_IN	WORD	65535			

```
LC_OU.WORD_AS_BIT(
    WORD_AS_BIT=> ,
    WORD_IN:=WORD_IN 65535 ,
    B00=>B00 True ,
    B01=> B01 True ,
    B02=>B02 True ,
    B03=>B03 True ,
    B04=> B04 True ,
    B05=> B05 True ,
    B06=> B06 True ,
    B07=> B07 True ,
    B08=> B08 True ,
    B09=>B09 True ,
    B10=>B10 True ,
    B11=> B11 True ,
    B12=>B12 True ,
    B13=>B13 True ,
    B14=>B14 True ,
    B15=>B15 True );
```

LD语言

表达式	类型	值	准备值	地址	注释
B00	BOOL	TRUE			Bit 00位
B01	BOOL	TRUE			Bit 01位
B02	BOOL	TRUE			Bit 02位
B03	BOOL	TRUE			Bit 03位
B04	BOOL	TRUE			Bit 04位
B05	BOOL	TRUE			Bit 05位
B06	BOOL	TRUE			Bit 06位
B07	BOOL	TRUE			Bit 07位
B08	BOOL	TRUE			Bit 08位
B09	BOOL	TRUE			Bit 09位
B10	BOOL	TRUE			Bit 10位
B11	BOOL	TRUE			Bit 11位
B12	BOOL	TRUE			Bit 12位
B13	BOOL	TRUE			Bit 13位
B14	BOOL	TRUE			Bit 14位
B15	BOOL	TRUE			Bit 15位
WORD_IN	WORD	65535			



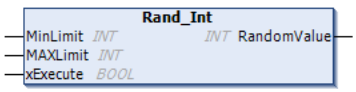
3.9 数学指令

指令列表

指令类别	名称	FB/FC	功能
字符串操作	Rand_Int	FB	随机数整型
	Rand_Real	FB	随机数浮点型

3.9.1 Rand_Int　随机数整型

①指令格式

指令	功能	FB/FC	LD 表现	ST表现
Rand_Int	随机数整型	FB		Rand_Int_1(MinLimit:= IN1, MAXLimit:=IN2 , xExecute:=IN3 , RandomValue=>OUT);

②相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
MinLimit	区间最小值	INT	-	0	设置要获取区间最小值
MAXLimit	区间最大值	INT	-	0	设置要获取区间最大值
xExecute	获取按钮	BOOL	0-1	0	上升沿触发，

输出变量

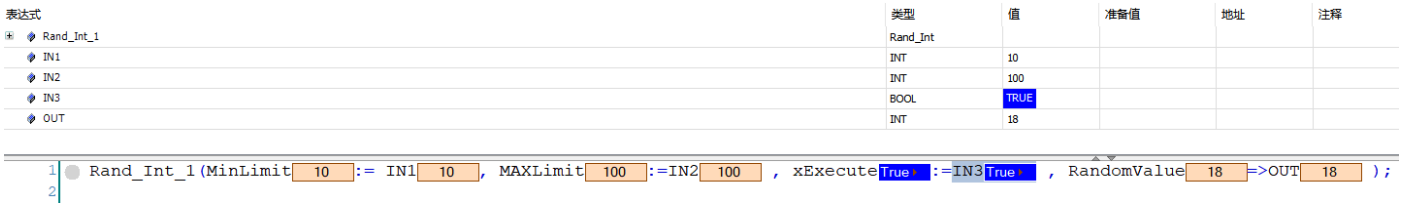
输出变量	名称	数据类型	有效范围	初始值	描述
RandomValue	得到的结果	INT	-	0	得到的随机数

数据类型	布尔	位串				整数								实数		时刻、持续时间 日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATA	TOD	DT	STRING
MinLimit											√									
MAXLimit											√									
xExecute	√																			
RandomValue											√									

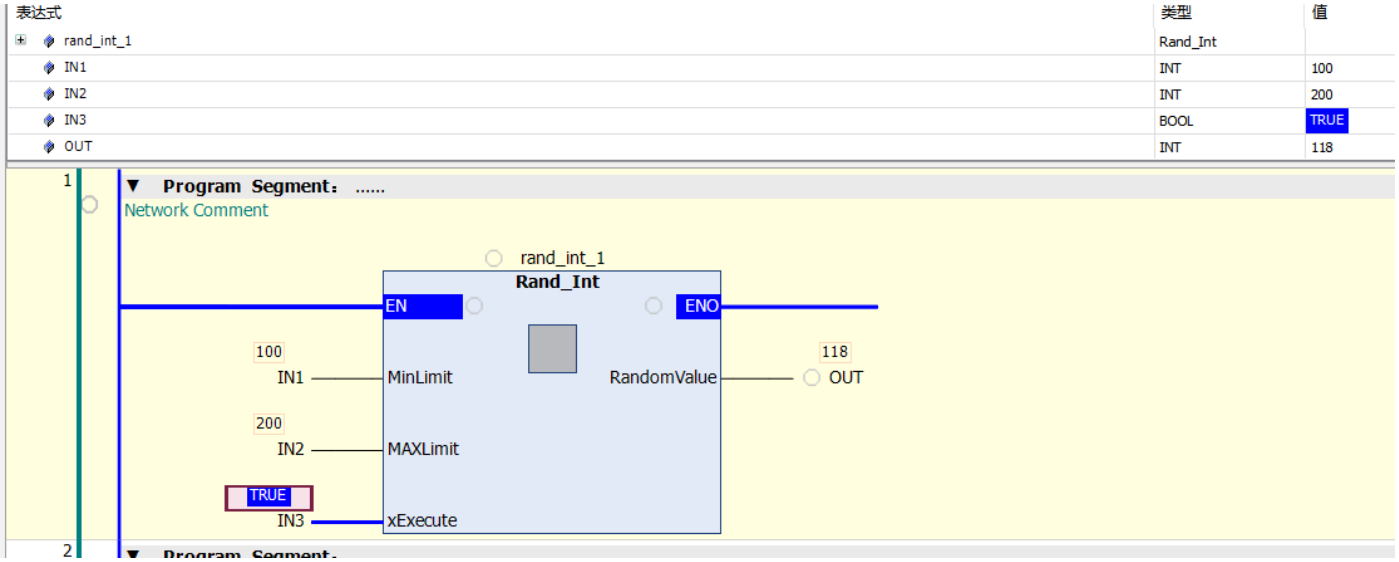
③程序示例

根据MinLimit最小值和MAXLimit最大值中的区间触发xExecute按钮每次取一个随机数给到RandomValue

ST:



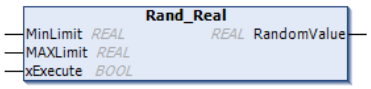
LD:



※注意事项

3.9.2 Rand_Real 随机数浮点型

①指令格式

指令	功能	FB/FC	LD 表现	ST表现
Rand_Real	随机数浮点型	FB		Rand_Real(MinLimit:=IN1 , MAXLimit:=IN2 , xExecute:=IN3 , RandomValue=>OUT);

②相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
MinLimit	区间最小值	REAL	-	0	设置要获取区间最小值
MAXLimit	区间最大值	REAL	-	0	设置要获取区间最大值
xExecute	获取按钮	BOOL	0-1	0	上升沿触发，

输出变量

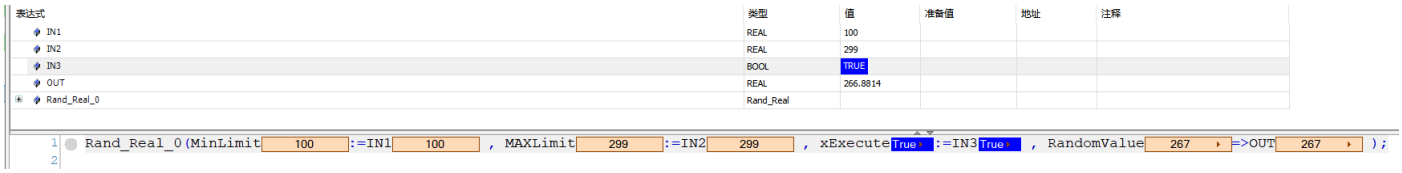
输出变量	名称	数据类型	有效范围	初始值	描述
RandomValue	得到的结果	REAL	32位	0	得到的随机数

数据类型	布尔	位串				整数								实数		时刻、持续时间 日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATA	TOD	DT	STRING
MinLimit														√						
MAXLimit														√						
xExecute	√																			
RandomValue														√						

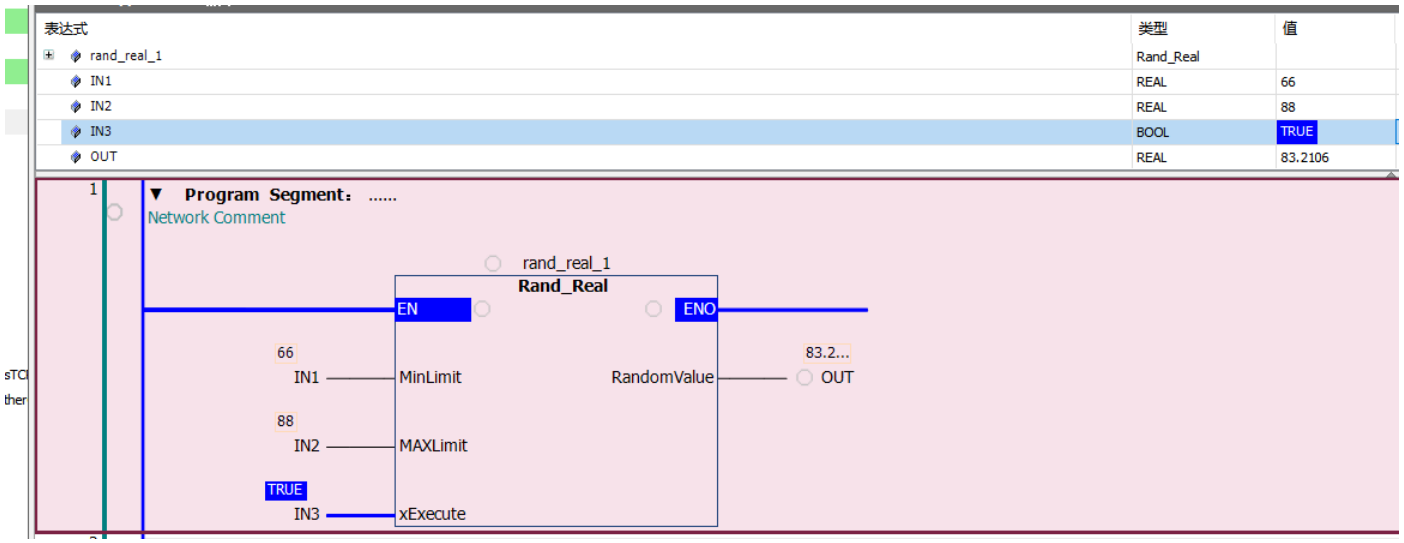
③程序示例

根据MinLimit最小值和MAXLimit最大值中的区间触发xExecute按钮每次取一个随机数给到RandomValue

ST:



LD:



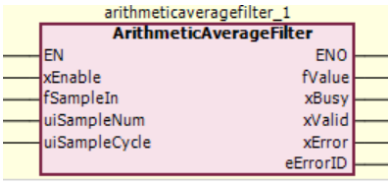
3.10 滤波

指令列表

指令类别	名称	FB/FC	功能
滤波	ArithmeticAverageFilter	FB	一维算数平均滤波
	RecursiveAverageFilter	FB	递推平均滤波
	WeightRecursiveAverageFilter	FB	加权递推平均滤波
	MedianFilter	FB	中位值滤波
	MedianAverageFilter	FB	中位值平均滤波
	LimitingFilter	FB	限幅滤波
	LimitingAverageFilter	FB	限幅平均滤波
	LimitingDebounceFilter	FB	限幅消抖滤波
	DebounceFilter	FB	消抖滤波
	FirstOrderFilter	FB	一阶滤波器

3.10.1 ArithmeticAverageFilter 一维算数平均滤波

①指令格式

指令	功能	FB/FC	LD 表现	ST表现
ArithmeticAverageFilter	一维算数平均滤波	FB		ArithmeticAverageFilter_0 (xEnable:= , fSampleIn:= , uiSampleNum:= , uiSampleCycle:= , fValue=> , xBusy=> , xValid=> , xError=> , eErrorID=>);

②相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
xEnable	滤波使能	BOOL	[FALSE, TRUE]	FALSE	滤波使能
fSampleIn	输入采样值	REAL	-	0	输入采样值
uiSampleNum	输入采样个数N	UINT	1-1000	0	输入采样个数N
uiSampleCycle	输入采样周期	UINT	1-1000	0	输入采样周期

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
fValue	滤波输出有效值	REAL	0	0	滤波输出有效值
xBusy	指令正在执行	BOOL	[FALSE, TRUE]	FALSE	指令正在执行
xValid	输出有效	BOOL	[FALSE, TRUE]	FALSE	输出有效
XError	错误	BOOL	[FALSE, TRUE]	FALSE	错误

数据类型

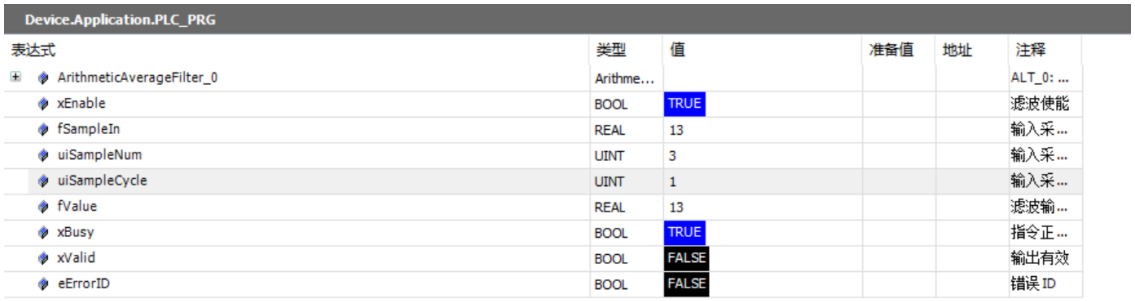
	布尔	位串				整数							实数		时刻、持续时间 日期、字符串					
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
xEnable	√																			
fSampleIn														√						
uiSampleNum							√													
uiSampleCycle							√													
fValue														√						
xBusy	√																			
xValid	√																			
XError	√																			

③功能说明

采样计算的样本个数为3（1~3000，无符号整型）。 采样周期为1（给定运行周期数执行一次数据采样，默认值为1）。 输入xFilter 为TRUE时，功能块能流有效开始采样滤波，设定采样计算的样本个数为3（即每3个采样值进行滤波计算一次），将3个采样值求算术平均值作为滤波值输出，滤波执行有效标志位xValid置TRUE。

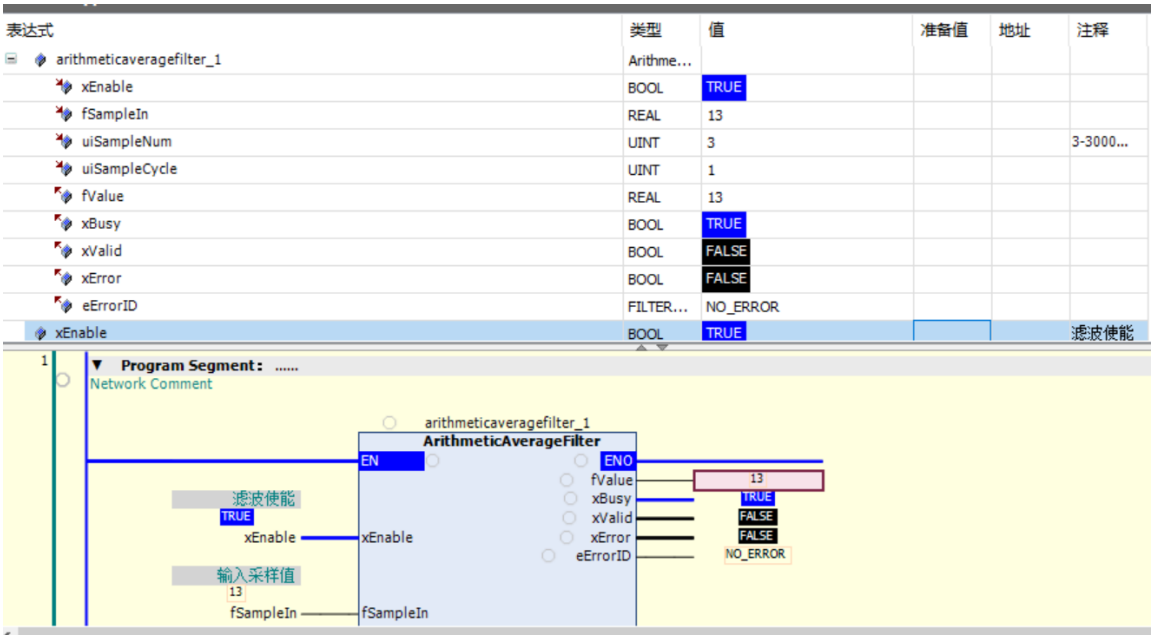
④程序示例

ST语言

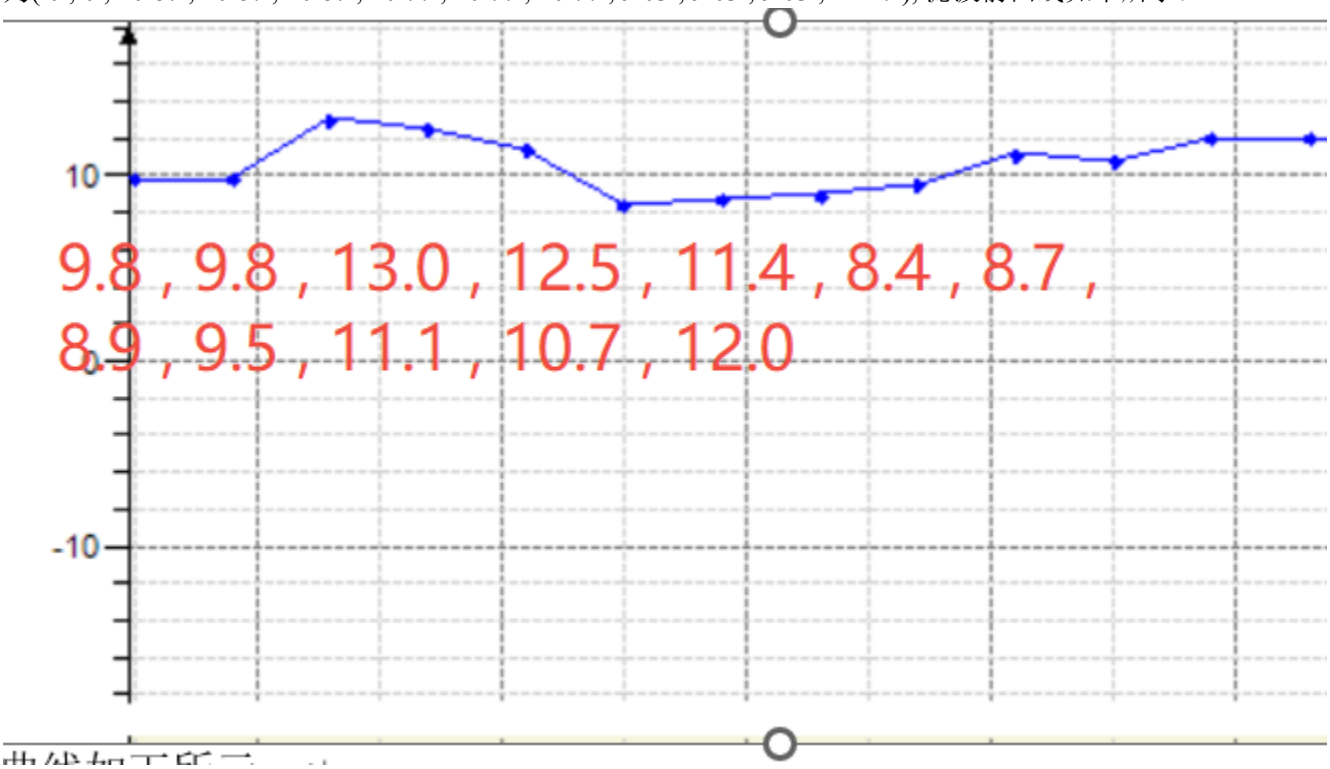



```
uiSampleNum 3 :=uiSampleNum 3 ,
uiSampleCycle 1 := uiSampleCycle 1 ,
fValue 13 => fValue 13 ,
xBusy True => xBusy True ,
xValid False => xValid False ,
xError=> ,
eErrorID=> );
```

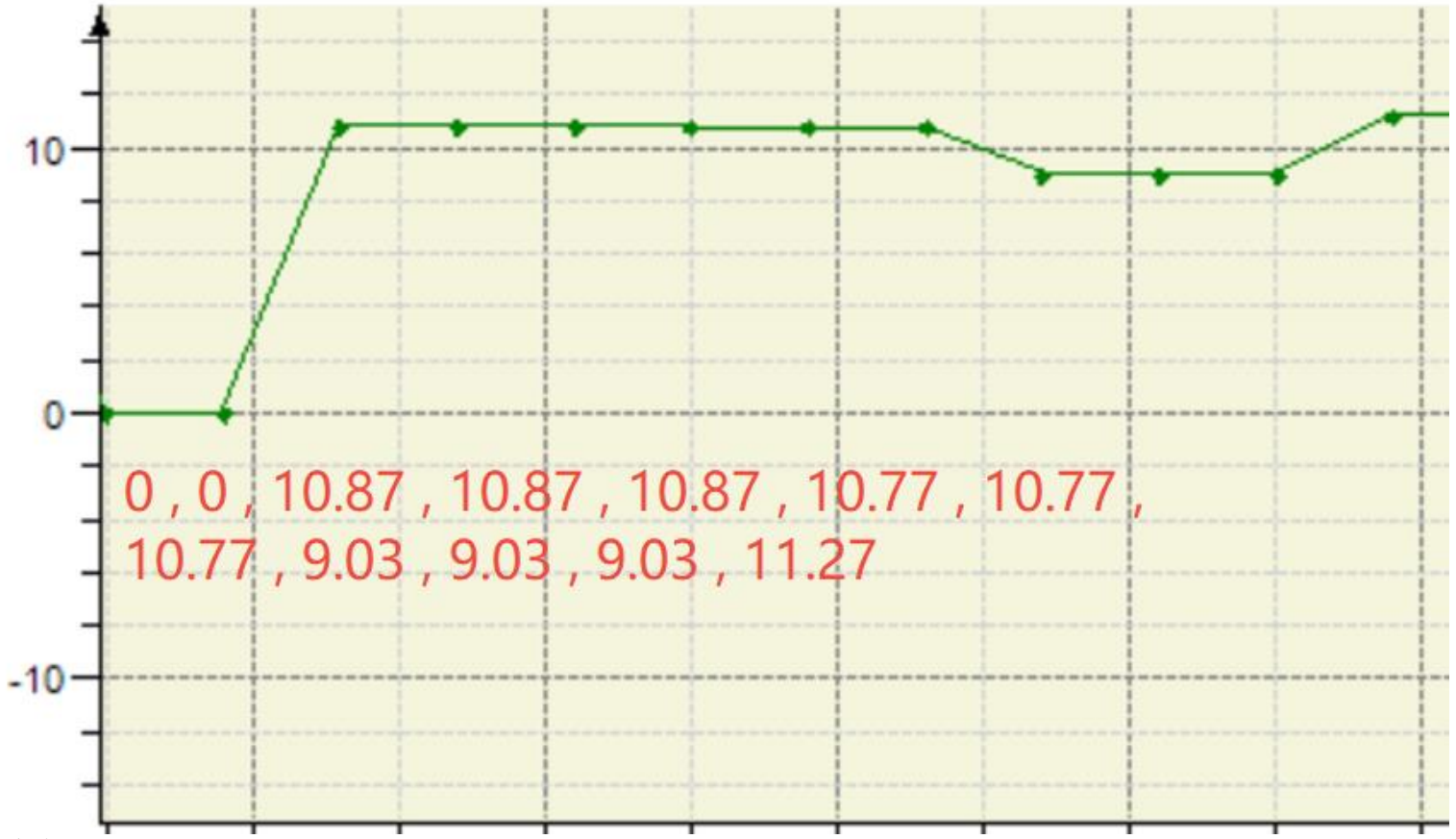
LD语言



连续12个周期，采样值分别为(9.8 , 9.8 , 13.0 , 12.5 , 11.4 , 8.4 , 8.7 , 8.9 , 9.5 , 11.1 , 10.7 , 12.0), 经过滤波后，结果分别为(0 , 0 , 10.87 , 10.87 , 10.87 , 10.77 , 10.77 , 10.77 , 9.03 , 9.03 , 9.03 , 11.27), 滤波前曲线如下所示：



滤波后曲线如下所示：



注意事项

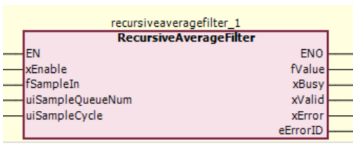
1） 滤波功能块中输入参数uiSampleCycle为采样周期，若uiSampleCycle=1时表示每个周期均进行采样 操作；若uiSampleCycle=3表示每3个周期进行采样一次。功能块的标志位ArithmeticAverageFilter. xValid 输出TRUE则表示滤波有效。滤波输出值fValue在滤波输出有效时会更新输出。

2） 若采样数据为存放于数组的数据，需将数组中的数据逐个输入滤波功能块，需用户自行处理数据。

3） 输入采样个数uiSampleNum参数不在有效范围内，xError置TRUE，并且功能块故障码eErrorID提示SAMPLENUM_PARAMETER_SET_ERROR错误信息，需重新设置该参数在有效范围内并且能流重新触发， 复位功能块的故障和xError信号。

3.10.2 RecursiveAverageFilter 递推平均滤波

①指令格式

指令	功能	FB/FC	LD 表现	ST表现
RecursiveAverageFilter	递推平均滤波	FB		RecursiveAverageFilter_0(xEnable:= , fSampleIn:= , uiSampleQueueNum:= , uiSampleCycle:= , fValue=> , xBusy=> , xValid=> , xError=> , eErrorID=>);

②相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
xEnable	滤波使能	BOOL	[FALSE, TRUE]	FALSE	滤波使能
fSampleIn	输入采样值	REAL	-	0	输入采样值
uiSampleQueueNum	输入队列长度N	UINT	1-3000	0	输入采样个数N
uiSampleCycle	输入采样周期	UINT	1-3000	0	输入采样周期

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
fValue	滤波输出有效值	REAL	0	0	滤波输出有效值
xBusy	指令正在执行	BOOL	[FALSE, TRUE]	FALSE	指令正在执行
xValid	输出有效	BOOL	[FALSE, TRUE]	FALSE	输出有效
XError	错误	BOOL	[FALSE, TRUE]	FALSE	错误

数据类型

	布尔	位串				整数							实数		时刻、持续时间 日期、字符串					
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
xEnable	√																			
fSampleIn														√						
uiSampleQueueNum							√													
uiSampleCycle							√													
fValue														√						
xBusy	√																			
xValid	√																			
XError	√																			

③功能说明

1） 采样类别：连续采样，采样原始数据来源于连续更新的程序变量。

2） 采样计算的样本个数为3（1~3000，无符号整型）。

3） 采样周期为1（给定运行周期数执行一次数据采样，默认值为1）。

4） 输入xFilter为TRUE时，功能块能流有效开始采样滤波，设定采样计算的样本队列个数为3，每次把采 样到一个新数据放入队尾，并扔掉原来队首的一个数据（先进先出原则），将队列中的3个采样值求算术 平均值作为滤波值输出，滤波执行有效标志位xValid置TRUE。

④程序示例

ST语言

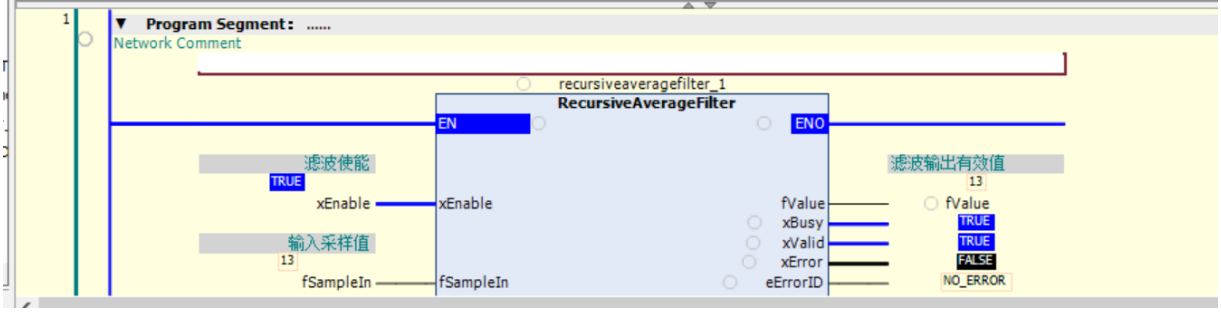
Device.Application.PLC_PRG

表达式	类型	值	准备值	地址	注释
ArithmeticAverageFilter_0	Arithme...				ALT_0: ...
xEnable	BOOL	TRUE			滤波使能
fSampleIn	REAL	13			输入采...
uiSampleNum	UINT	3			输入采...
uiSampleCycle	UINT	1			输入采...
fValue	REAL	13			滤波输...
xBusy	BOOL	TRUE			指令正...
xValid	BOOL	FALSE			输出有效
eErrorID	BOOL	FALSE			错误 ID

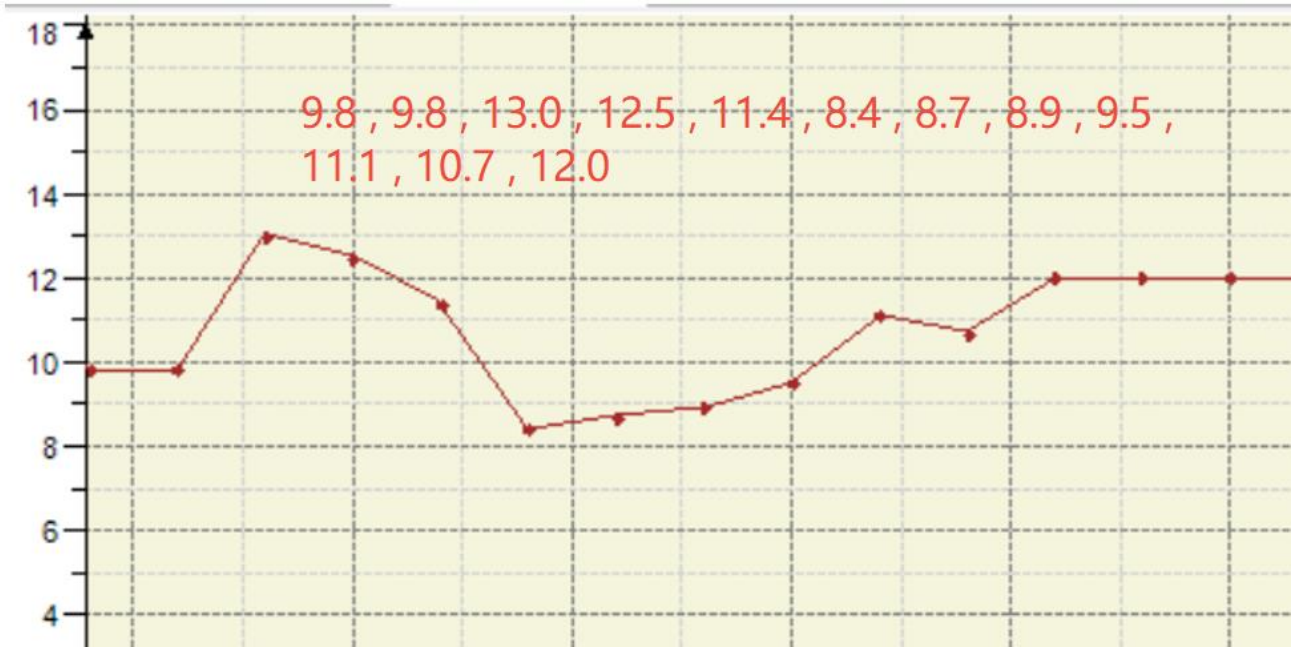
```
4       uiSampleNum 3 :=uiSampleNum 3 ,
5       uiSampleCycle 1 := uiSampleCycle 1 ,
6       fValue 13 => fValue 13 ,
7       xBusy True =>xBusy True ,
8       xValid False =>xValid False ,
9       xError=> ,
10      eErrorID=> );
```

LD语言

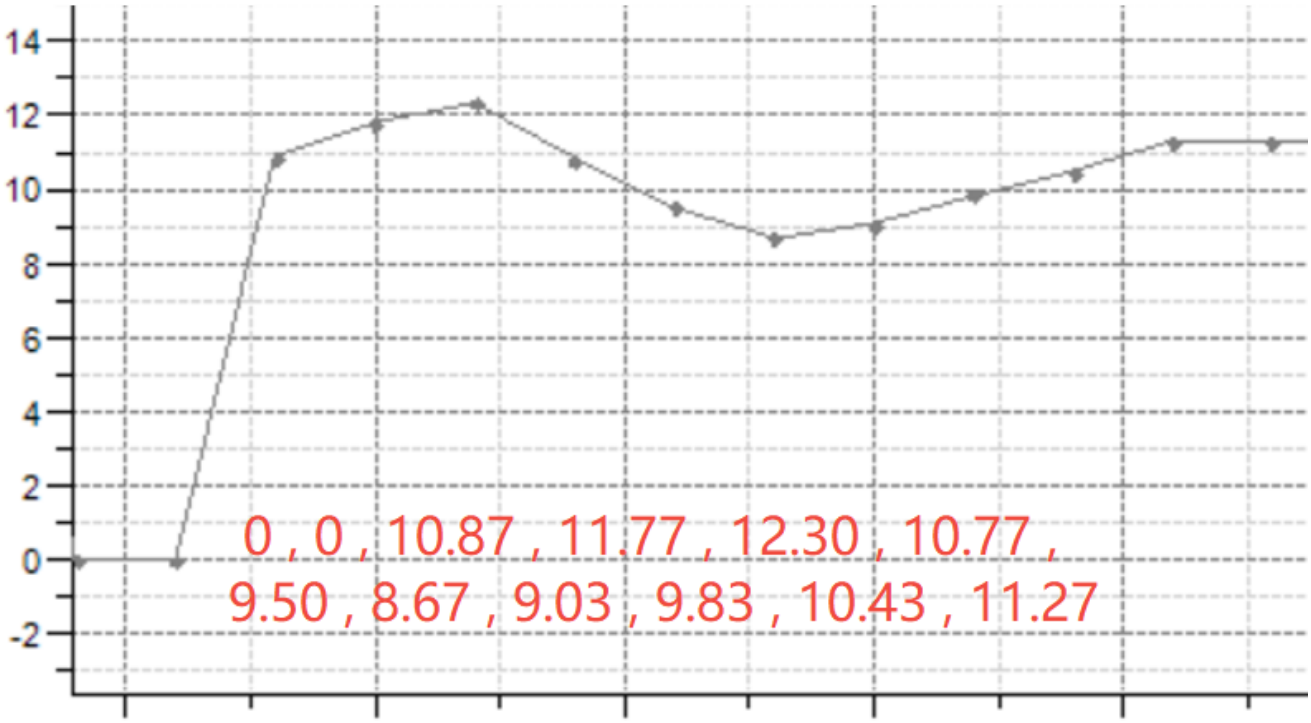
表达式	类型	值	准备值	地址	注释
recursiveaveragefilter_1	Recursi...				
xEnable	BOOL	TRUE			
fSampleIn	REAL	13			
uiSampleQueueNum	UINT	3			3-3000...
uiSampleCycle	UINT	1			
fValue	REAL	13			
xBusy	BOOL	TRUE			
xValid	BOOL	TRUE			
xError	BOOL	FALSE			
eErrorID	FILTER...	NO_ERROR			



连续12个周期，采样值分别为(9.8 , 9.8 , 13.0 , 12.5 , 11.4 , 8.4 , 8.7 , 8.9 , 9.5 , 11.1 , 10.7 , 12.0)。 经过滤波后，结果分别为(0 , 0 , 10.87 , 11.77 , 12.30 , 10.77 , 9.50 , 8.67 , 9.03 , 9.83 , 10.43 , 11.27)。 滤波前曲线如下所示：



滤波后曲线如下所示：



主意事项 ↵

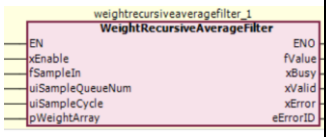
注意事项

1） 滤波功能块中输入参数uiSampleCycle为采样周期，若uiSampleCycle=1时表示每个周期均进行采样操 作，若uiSampleCycle=3表示每3个周期进行采样一次。RecursiveAverageFilter.xValid输出TRUE则 表示滤波有效。滤波输出值fValue在滤波输出有效时会更新输出。

2） 若采样数据为存放于数组的数据，需将数组中的数据逐个输入滤波功能块，需用户自行处理数据。

3.10.3 WeightRecursiveAverageFilter 加权递推平均滤波

①指令格式

指令	功能	FB/FC	LD 表现	ST表现
WeightRecursiveAverageFilter	加权递推平均滤波	FB		WeightRecursiveAverageFilter_0(xEnable:= , fSampleIn:= , uiSampleQueueNum:= , uiSampleCycle:= , pWeightArray:= , fValue=> , xBusy=> , xValid=> , xError=> , eErrorID=>);

②相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
xEnable	滤波使能	BOOL	[FALSE, TRUE]	FALSE	滤波使能

fSampleIn	输入采样值	REAL	-	0	输入采样值
uiSampleNum	输入采样个数N	UINT	1-1000	0	输入采样个数N
uiSampleCycle	输入采样周期	UINT	1-1000	0	输入采样周期
pWeightArray	输入加权系数，存放于数组中	POINTER to REAL	0	0	输入加权系数，存放于数组中

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
fValue	滤波输出有效值	REAL	0	0	滤波输出有效值
xBusy	指令正在执行	BOOL	[FALSE, TRUE]	FALSE	指令正在执行
xValid	输出有效	BOOL	[FALSE, TRUE]	FALSE	输出有效
XError	错误	BOOL	[FALSE, TRUE]	FALSE	错误

数据类型

	布尔	位串				整数							实数		时刻、持续时间 日期、字符串					
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
xEnable	√																			
fSampleIn														√						
uiSampleNum							√													
uiSampleCycle							√													
pWeightArray														√						
fValue														√						
xBusy	√																			
xValid	√																			
XError	√																			

③功能说明

连续采样，采样原始数据来源于连续更新的程序变量。 加权系数为(1.0 , 1.2 , 1.3)（采样数据对应的权值，浮点型，加权系数个数m与队列长度n相等，若加权系 数个数m大于队列长度n，取加权系数前n个），采样计算的样本队列为3（1~3000，无符号整型）。 滤波计算周期为1（给定运行周期数执行一次采样滤波，默认值为1）。 输入xFilter 为TRUE时，功能块能流有效开始采样滤波，设定采样计算的样本队列个数为3，每次把采样到 一个新数据放入队尾，并扔掉原来队首的一个数据（先进先出原则），将队列中的3个采样值分别乘以相应的 加权系数之和，再除以加权系数之和，得到采样滤波值，功能块输出滤波值，滤波执行有效标志位xValid置 TRUE

④程序示例

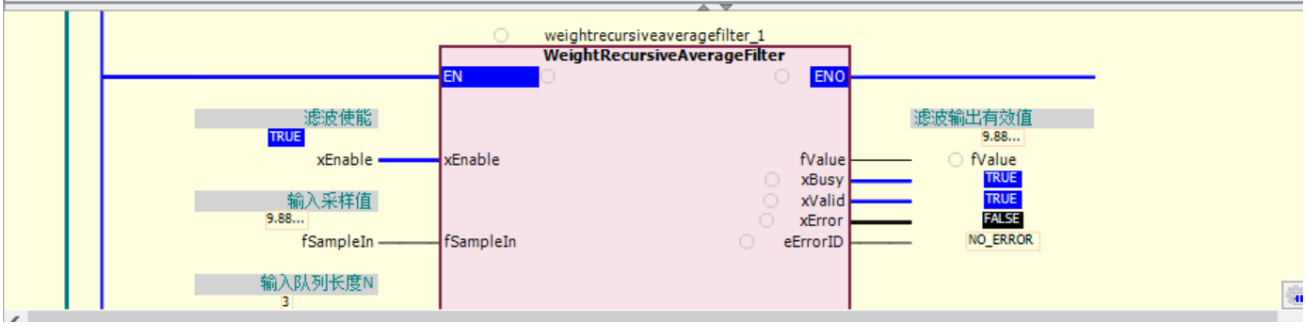
ST语言

表达式	类型	值	准备值	地址	注释
WeightRecursiveAverageFilter_0	Weight...				ALT_0: ...
xEnable	BOOL	TRUE			
fSampleIn	REAL	9.84546			
uiSampleQueueNum	UINT	3			3-3000...
uiSampleCycle	UINT	1			
pWeightArray	POINTE...	16#B6772CC4			
pWeightArray^	REAL	1.1			
fValue	REAL	9.84546			
xBusy	BOOL	TRUE			
xValid	BOOL	TRUE			

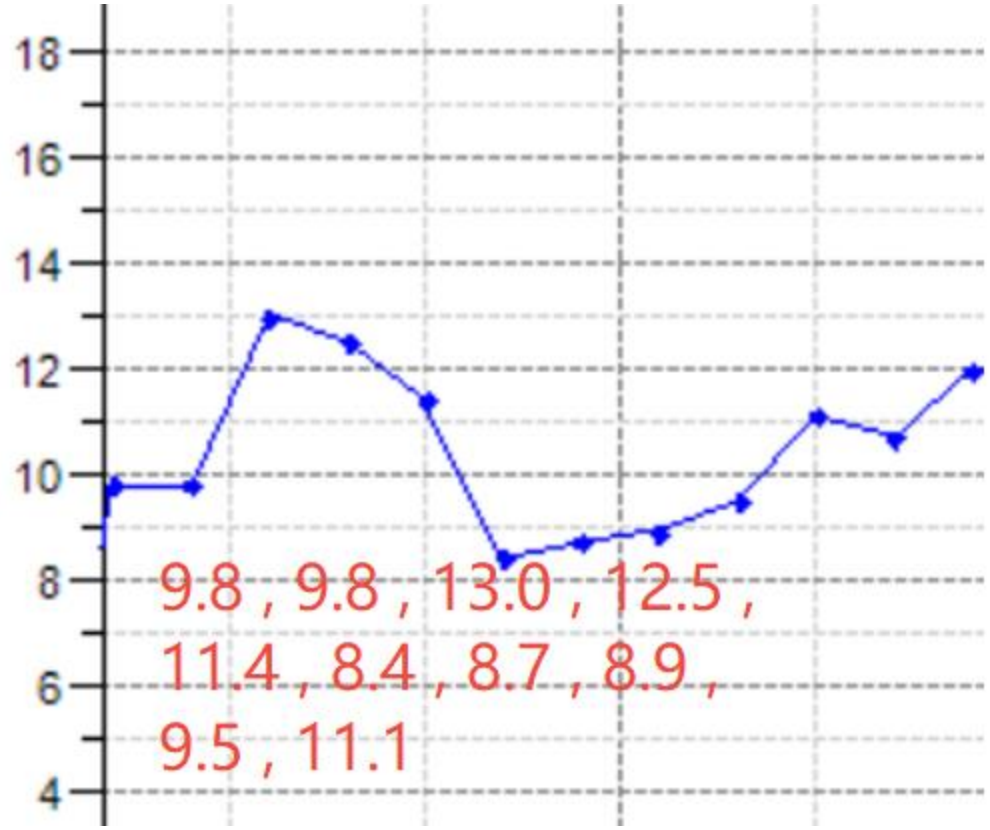
```
1  WeightRecursiveAverageFilter_0(
2      xEnable True :=xEnable True ,
3      fSampleIn 9.85 :=fSampleIn 9.85 ,
4      uiSampleQueueNum 3 := uiSampleQueueNum 3 ,
5      uiSampleCycle 1 :=uiSampleCycle 1 ,
6      pWeightArray 16#B6772CC4 :=ADR(aWeightArray 1.1 ) ,
7      fValue 9.85 =>fValue 9.85 ,
8      xBusy=> ,
9      xValid=> ,
10     xError=> ,
11     eErrorID=> );
```

LD语言

表达式	类型	值	准备值	地址	注释
xEnable	BOOL	TRUE			
fSampleIn	REAL	9.8765			
uiSampleQueueNum	UINT	3			3-3000...
uiSampleCycle	UINT	1			
pWeightArray	POINTE...	16#B6772CC8			
fValue	REAL	9.8765			
xBusy	BOOL	TRUE			
xValid	BOOL	TRUE			
xError	BOOL	FALSE			
eErrorID	FILTER...	NO_ERROR			



连续10个周期，权系数为(1.0 , 1.1 , 1.2)， 采样值分别为(9.8 , 9.8 , 13.0 , 12.5 , 11.4 , 8.4 , 8.7 , 8.9 , 9.5 , 11.1)， 经过滤波后，结果分别为(0 , 0 , 10.96 , 11.85 , 12.25 , 10.64 , 9.42 , 8.68 , 9.06 , 9.9) 滤波前曲线如下所示：



滤波后曲线如下所示：

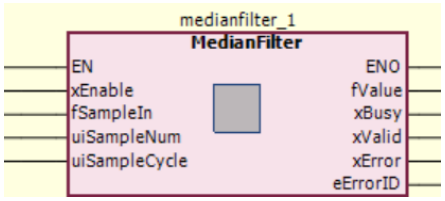
■ 注意事项 1） 滤波功能块中输入参数uiSampleCycle为采样周期，若uiSampleCycle=1时表示每个周期均进行采样操 作；若uiSampleCycle=3表示每3个周期进行采样一次。WeightRecursiveAverageFilter.xValid输出 TRUE则表示滤波有效。滤波输出值fValue在滤波输出有效时会更新输出。

2） 若采样数据为存放于数组的数据，需将数组中的数据逐个输入滤波功能块，需用户自行处理数据。

3） 加权系数不能全为0。

3.10.4 MedianFilter 中位值滤波

①指令格式

指令	功能	FB/FC	LD 表现	ST表现
MedianFilter	中位值滤波	FB		MedianFilter_0(xEnable:= , fSampleIn:= , uiSampleNum:= , uiSampleCycle:= , fValue=> , xBusy=> , xValid=> , xError=> , eErrorID=>);

②相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
xEnable	滤波使能	BOOL	[FALSE, TRUE]	FALSE	滤波使能
fSampleIn	输入采样值	REAL	-	0	输入采样值
uiSampleNum	输入采样个数N	UINT	1-1000	0	输入采样个数N
uiSampleCycle	输入采样周期	UINT	1-1000	0	输入采样周期

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
fValue	滤波输出有效值	REAL	0	0	滤波输出有效值
xBusy	指令正在执行	BOOL	[FALSE, TRUE]	FALSE	指令正在执行
xValid	输出有效	BOOL	[FALSE, TRUE]	FALSE	输出有效
XError	错误	BOOL	[FALSE, TRUE]	FALSE	错误

数据类型

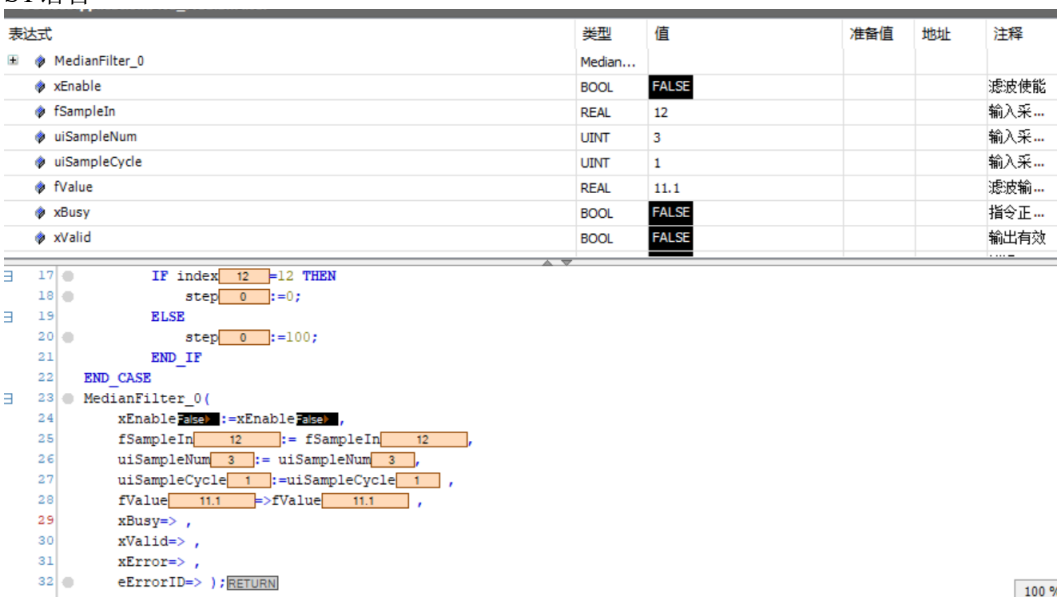
	布尔	位串				整数							实数		时刻、持续时间 日期、字符串					
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
xEnable	√																			
fSampleIn														√						
uiSampleNum							√													
uiSampleCycle							√													
fValue														√						
xBusy	√																			
xValid	√																			
XError	√																			

③功能说明

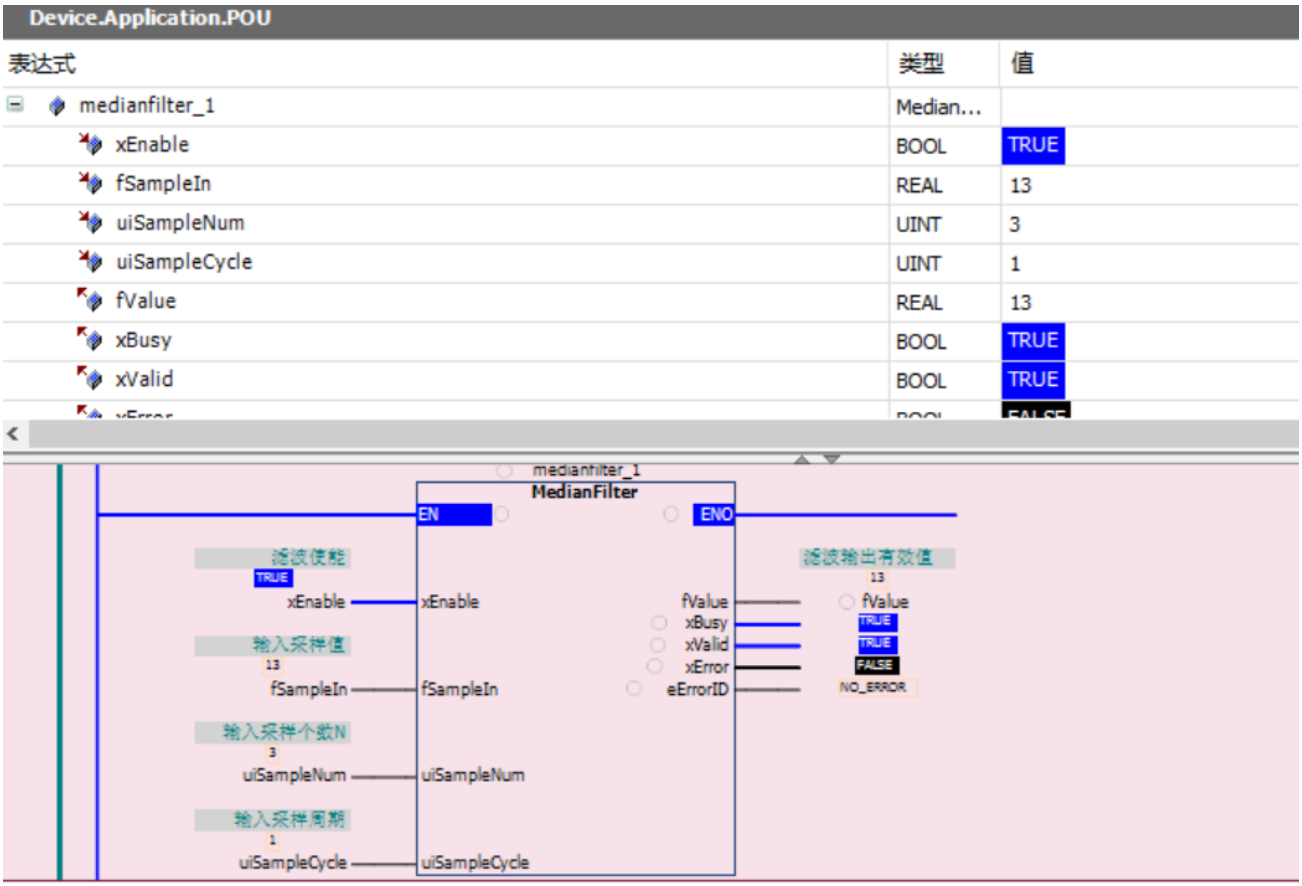
采样类别：连续采样，采样原始数据来源于连续更新的程序变量。 采样计算的样本个数为3（1~3000，无符号整型）。 采样周期为1（给定运行周期数执行一次数据采样，默认值为1）。 输入xFilter 为TRUE时，功能块能流有效开始采样滤波，设定采样计算的样本个数为3（即每3个采样值进 行滤波计算一次）。将3个采样值按大小进行排序，再取中间值作为滤波值输出，滤波执行有效标志位xValid 置TRUE。

④程序示例

ST语言

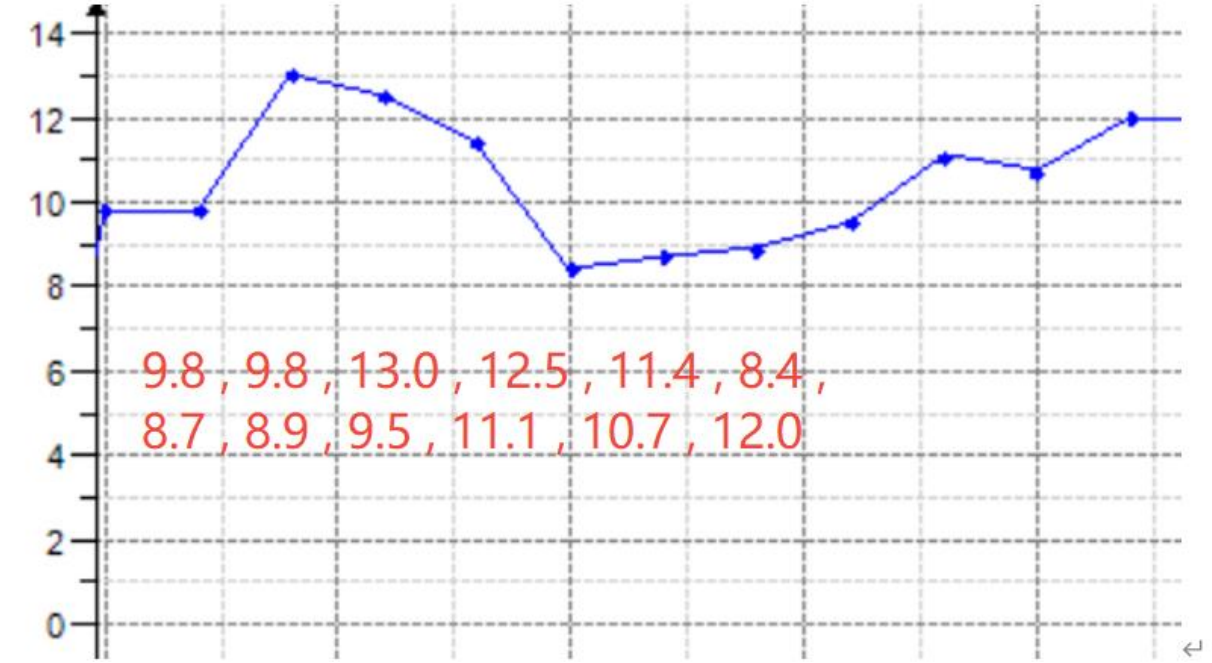


LD语言

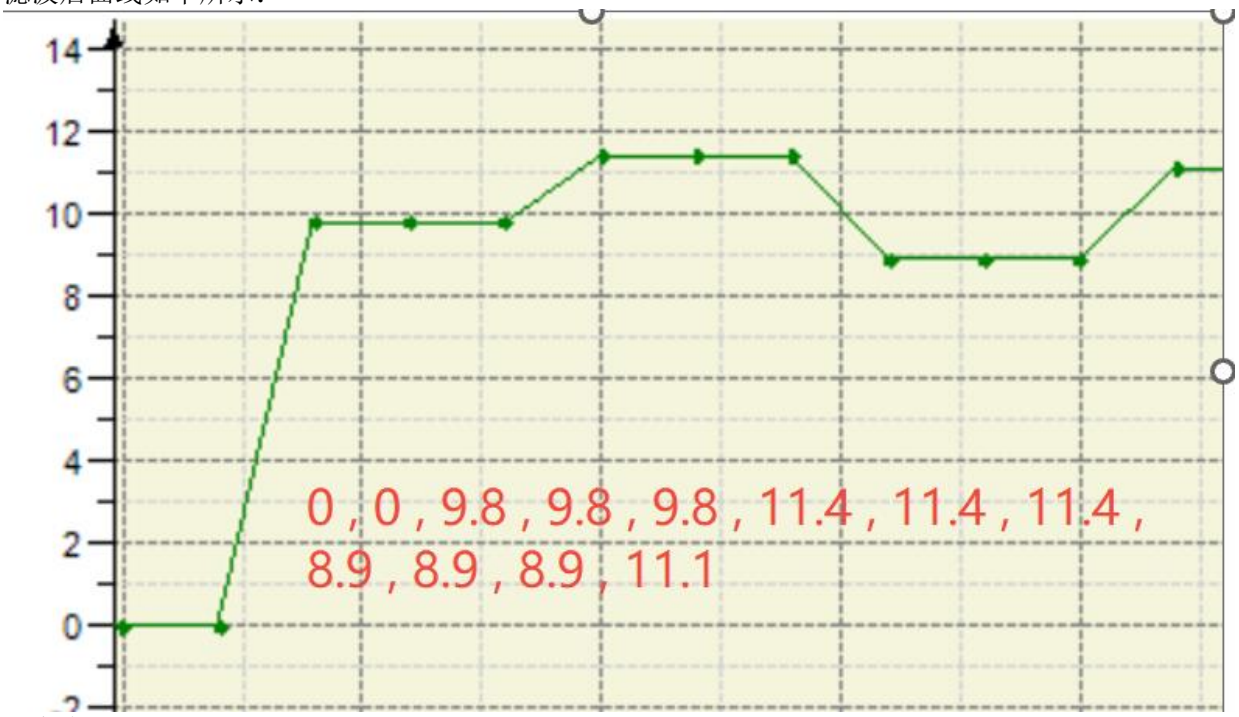


连续12个周期，采样值分别为(9.8 , 9.8 , 13.0 , 12.5 , 11.4 , 8.4 , 8.7 , 8.9 , 9.5 , 11.1 , 10.7 , 12.0)， 经过滤波后，结果分别为(0 , 0 , 9.8 , 9.8 , 9.8 , 11.4 , 11.4 , 11.4 , 8.9 , 8.9 , 8.9 , 11.1)，

滤波前曲线如下所示：



滤波后曲线如下所示：



注意事项

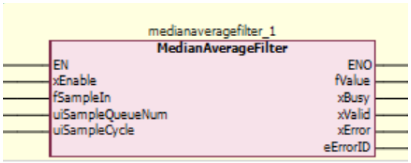
1） 滤波功能块中输入参数uiSampleCycle为采样周期，若uiSampleCycle=1时表示每个周期均进行采样操 作；若uiSampleCycle=3表示每3个周期进行采样一次。MedianFilter.xValid输出TRUE则表示滤波有效。 滤波输出值fValue在滤波输出有效时会更新输出。

2） 若采样数据为存放于数组的数据，需将数组中的数据逐个输入滤波功能块，需用户自行处理数据。

3.10.5 MedianAverageFilter 中位值平均滤波

①指令格式

指令	功能	FB/FC	LD 表现	ST表现

MedianAverageFilter	中位值平均滤波	FB		MedianAverageFilter_0(xEnable:= , fSampleIn:= , uiSampleQueueNum:= , uiSampleCycle:= , fValue=> , xBusy=> , xValid=> , xError=> , eErrorID=>);

②相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
xEnable	滤波使能	BOOL	[FALSE, TRUE]	FALSE	滤波使能
fSampleIn	输入采样值	REAL	-	0	输入采样值
uiSampleQueueNum	输入采样个数N	UINT	3-3000	0	输入采样个数N
uiSampleCycle	输入采样周期	UINT	1-1000	0	输入采样周期

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
fValue	滤波输出有效值	REAL	0	0	滤波输出有效值
xBusy	指令正在执行	BOOL	[FALSE, TRUE]	FALSE	指令正在执行
xValid	输出有效	BOOL	[FALSE, TRUE]	FALSE	输出有效
XError	错误	BOOL	[FALSE, TRUE]	FALSE	错误

数据类型

	布尔	位串				整数							实数		时刻、持续时间 日期、字符串					
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
xEnable	√																			
fSampleIn														√						
uiSampleNum							√													
uiSampleCycle							√													
fValue														√						
xBusy	√																			
xValid	√																			
XError	√																			

③功能说明

采样类别：连续采样，采样原始数据来源于连续更新的程序变量。 采样计算的样本个数为4（3~3000，无符号整

型）。 采样周期为1（给定运行周期数执行一次数据采样，默认值为1）。 输入xFilter 为TRUE时，功能块能流有效开始采样滤波，设定采样计算的样本个数为4（即每4个采样值进 行滤波计算一次）。将4个采样值按大小进行排序，再取中间值作为滤波值输出，滤波执行有效标志位xValid 置TRUE。

④程序示例

ST语言

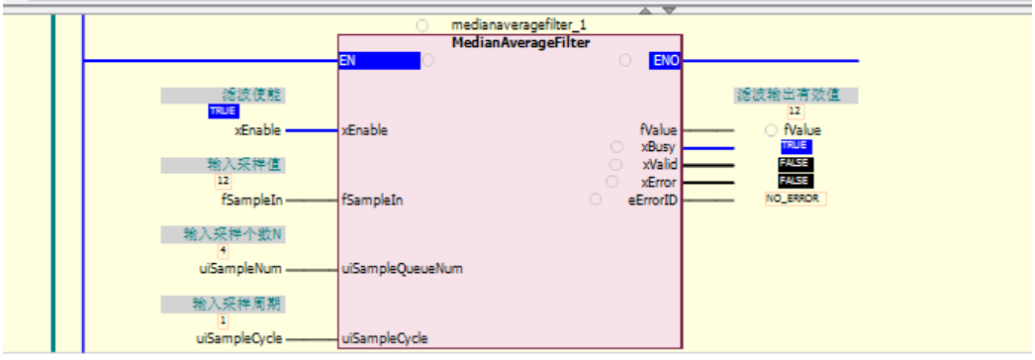
表达式	类型	值
MedianAverageFilter_0	Median...	
xEnable	BOOL	TRUE
fSampleIn	REAL	12
uiSampleQueueNum	UINT	4
uiSampleCycle	UINT	1
fValue	REAL	10.9
xBusy	BOOL	FALSE
xValid	BOOL	FALSE
xError	BOOL	FALSE
eErrorID	FILTER...	NO_ERROR
xEnable	BOOL	FALSE

```
        END_IF
END_CASE
MedianAverageFilter_0(
    xEnable:=xEnable ,
    fSampleIn:=fSampleIn ,
    uiSampleQueueNum:=uiSampleQueueNum ,
    uiSampleCycle:=uiSampleCycle ,
    fValue=>fValue ,
    xBusy=> ,
    xValid=> ,
    xError=> ,
    eErrorID=> );RETURN
```

LD语言

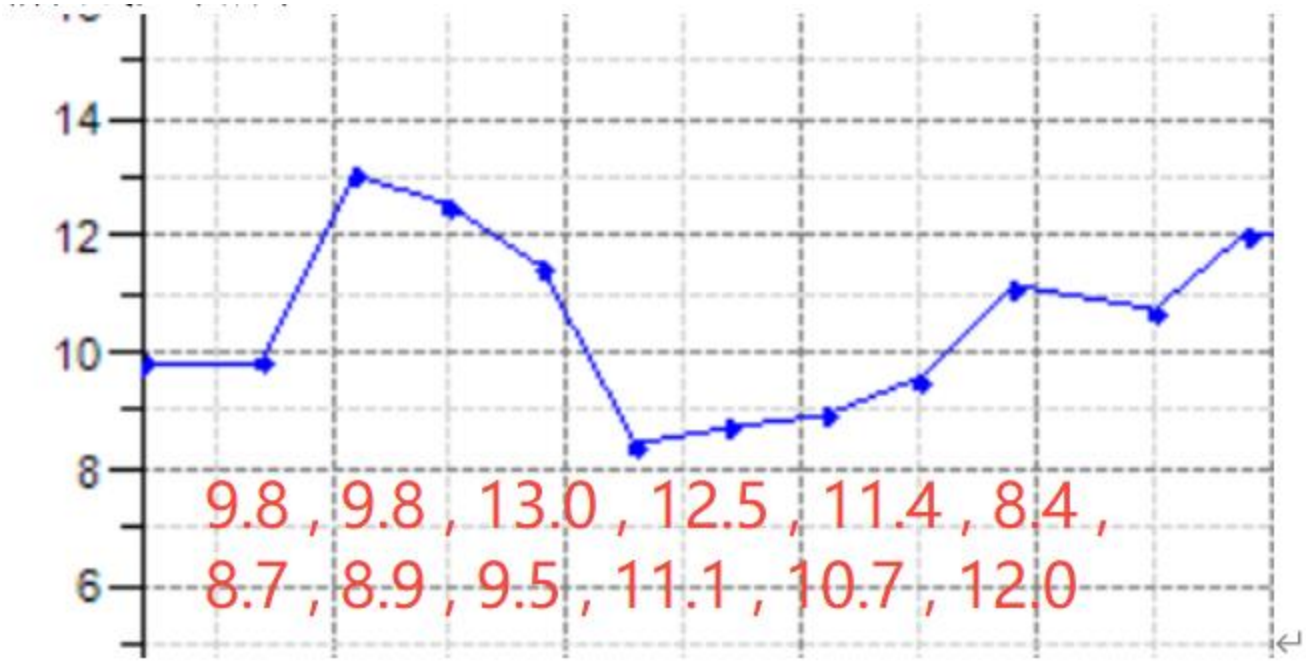
Device.Application.POU

表达式	类型	值
xEnable	BOOL	TRUE
fSampleIn	REAL	12
uiSampleNum	UINT	4
uiSampleCycle	UINT	1
fValue	REAL	12
xBusy	BOOL	FALSE
xValid	BOOL	FALSE
eErrorID	BOOL	FALSE

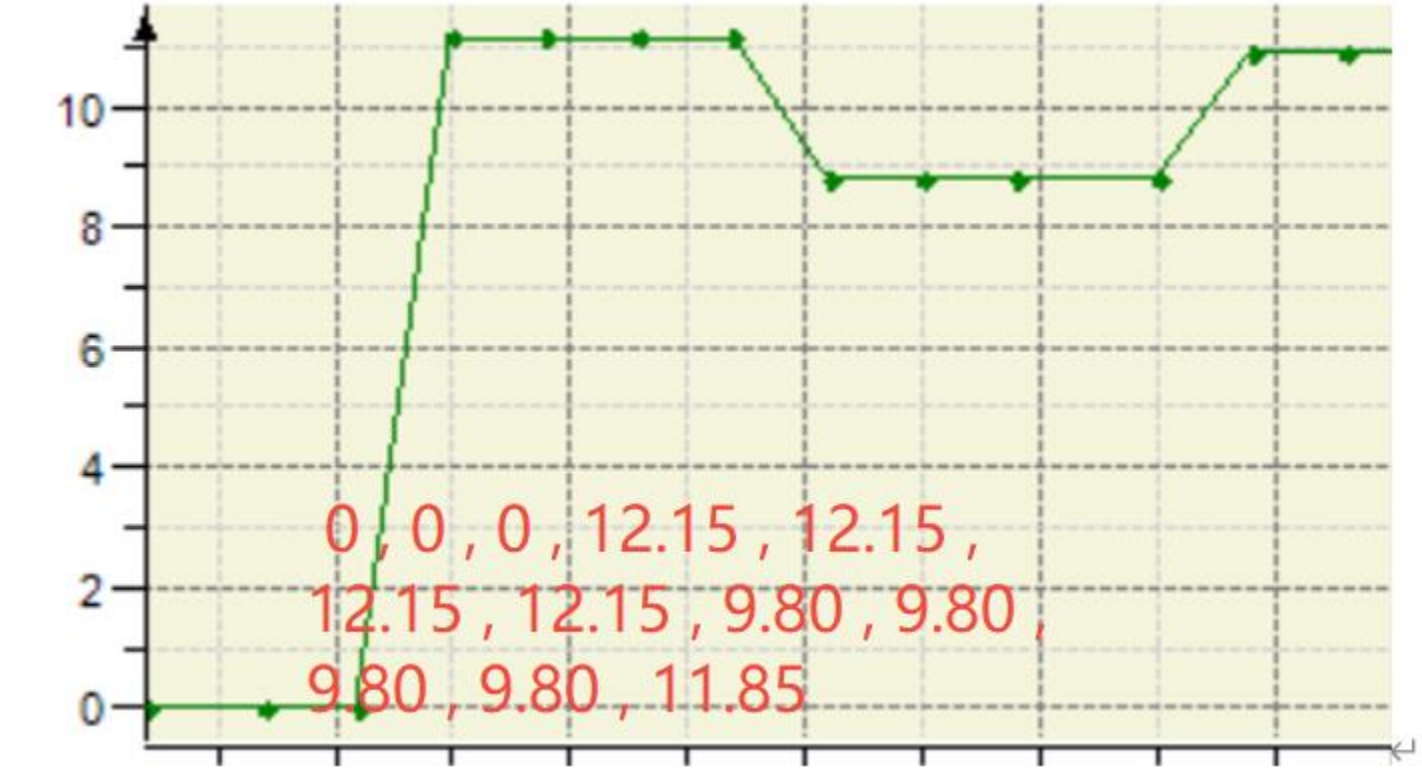


连续12个周期，采样值分别为(9.8 , 9.8 , 13.0 , 12.5 , 11.4 , 8.4 , 8.7 , 8.9 , 9.5 , 11.1 , 10.7 , 12.0)， 经过滤波后，结果分别为(0 , 0 , 0 , 12.15 , 12.15 , 12.15 , 12.15 , 9.80 , 9.80 , 9.80 , 9.80 , 11.85)，

滤波前曲线如下所示：



滤波后曲线如下所示：



注意事项

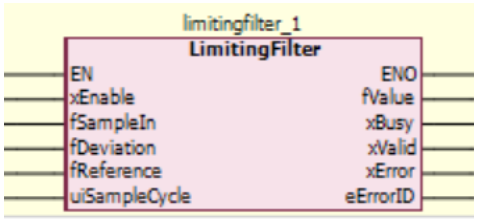
1） 滤波功能块中输入参数uiSampleCycle为采样周期，若uiSampleCycle=1时表示每个周期均进行采样操 作；若uiSampleCycle=3表示每3个周期进行采样一次。MedianAverageFilter.xValid输出TRUE则表示滤波有效。 滤波输出值fValue在滤波输出有效时会更新输出。

2） 若采样数据为存放于数组的数据，需将数组中的数据逐个输入滤波功能块，需用户自行处理数据。

3.10.6 LimitingFilter 限幅滤波

①指令格式

指令	功能	FB/FC	LD 表现	ST表现

LimitingFilter	限幅滤波	FB		LimitingFilter_0(xEnable:= , fSampleIn:= , fDeviation:= , fReference:= , uiSampleCycle:= , fValue=> , xBusy=> , xValid=> , xError=> , eErrorID=>);

②相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
xEnable	滤波使能	BOOL	[FALSE, TRUE]	FALSE	滤波使能
fSampleIn	输入采样值	REAL	-	0	输入采样值
fDeviation	输入两次采样允许的最大偏差值	REAL	-	0	输入两次采样允许的最大偏差值
fReference	输入采样参考值	REAL	-	0	输入采样参考值
uiSampleCycle	输入采样周期	UINT	1-1000	0	输入采样周期

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
fValue	滤波输出有效值	REAL	0	0	滤波输出有效值
xBusy	指令正在执行	BOOL	[FALSE, TRUE]	FALSE	指令正在执行
xValid	输出有效	BOOL	[FALSE, TRUE]	FALSE	输出有效
XError	错误	BOOL	[FALSE, TRUE]	FALSE	错误

数据类型

	布尔	位串				整数							实数		时刻、持续时间 日期、字符串					
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
xEnable	√																			
fSampleIn														√						
fDeviation														√						
fReference														√						
uiSampleCycle							√													
fValue														√						

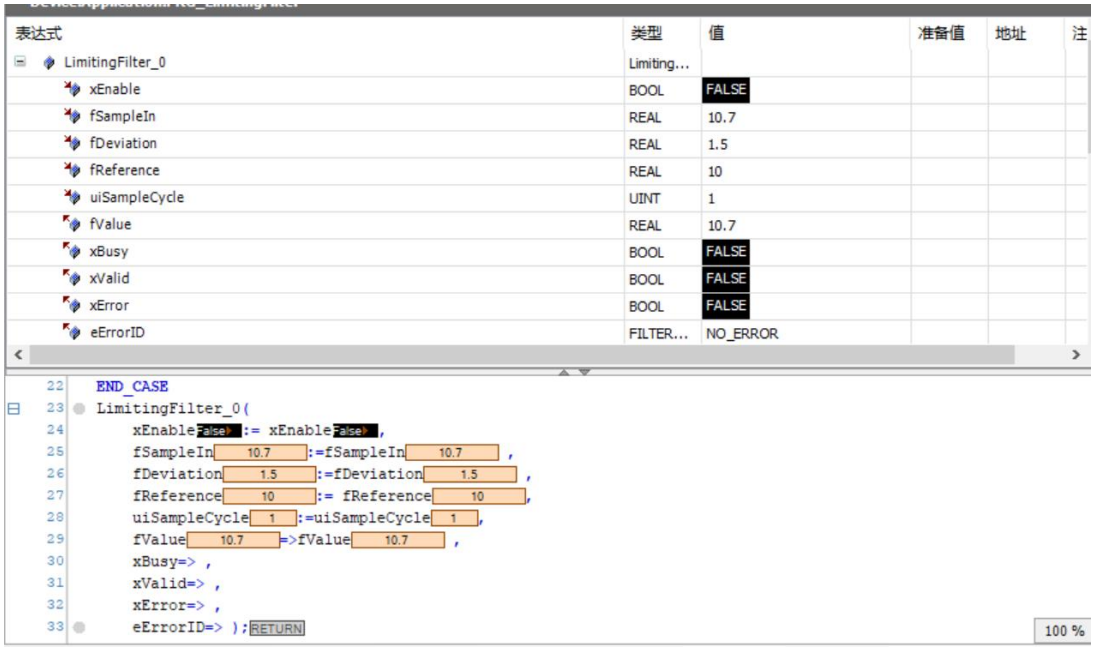
xBusy	√																			
xValid	√																			
XError	√																			

③功能说明

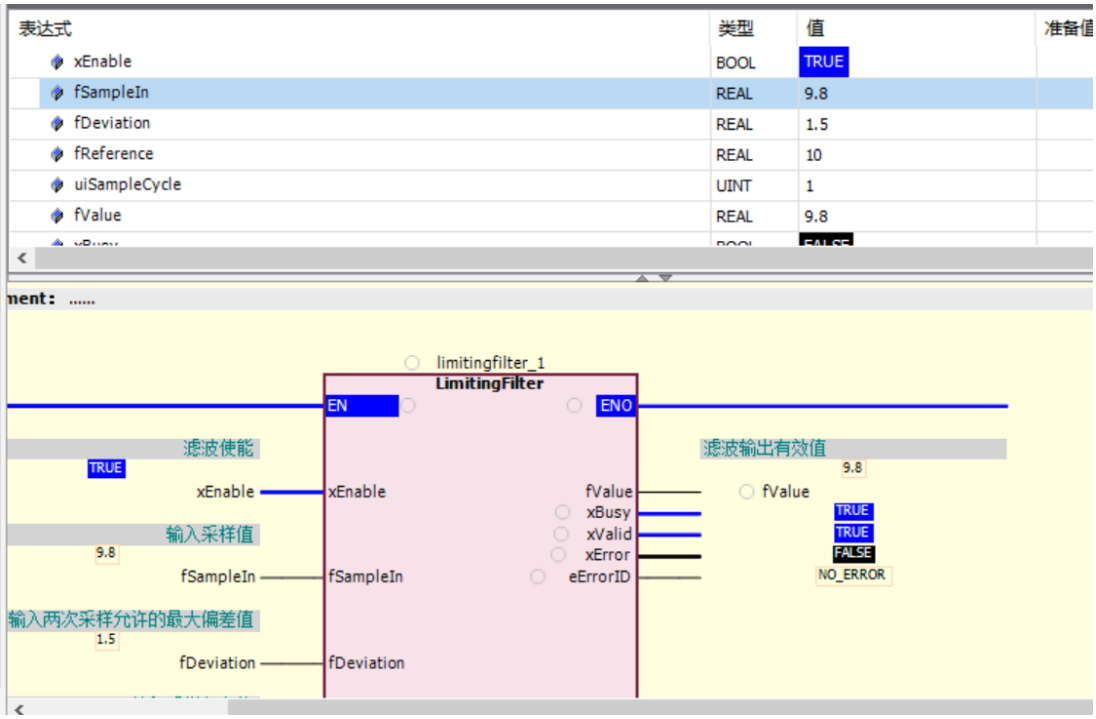
采样类别：连续采样，采样原始数据来源于连续更新的程序变量。 幅度参考值为10.0（用户给定的参考值，浮点型）。 幅度允许最大偏差值为1.5（偏差范围，浮点型）。 滤波计算周期为1（给定运行周期数执行一次采样滤波，默认值为1）。 输入xFilter 能流有效时限幅滤波功能块开始采样滤波，设定初次参考值为10.0作为有效值。将采样值与上次 有效值比较，若两者之差的绝对值大于设定的最大偏差值1.5，则上次有效值即为本次有效值；若小于等于最 大偏差值则本次值为有效值，将有效值作为滤波值输出，滤波执行有效标志位xValid置TRUE。

④程序示例

ST语言

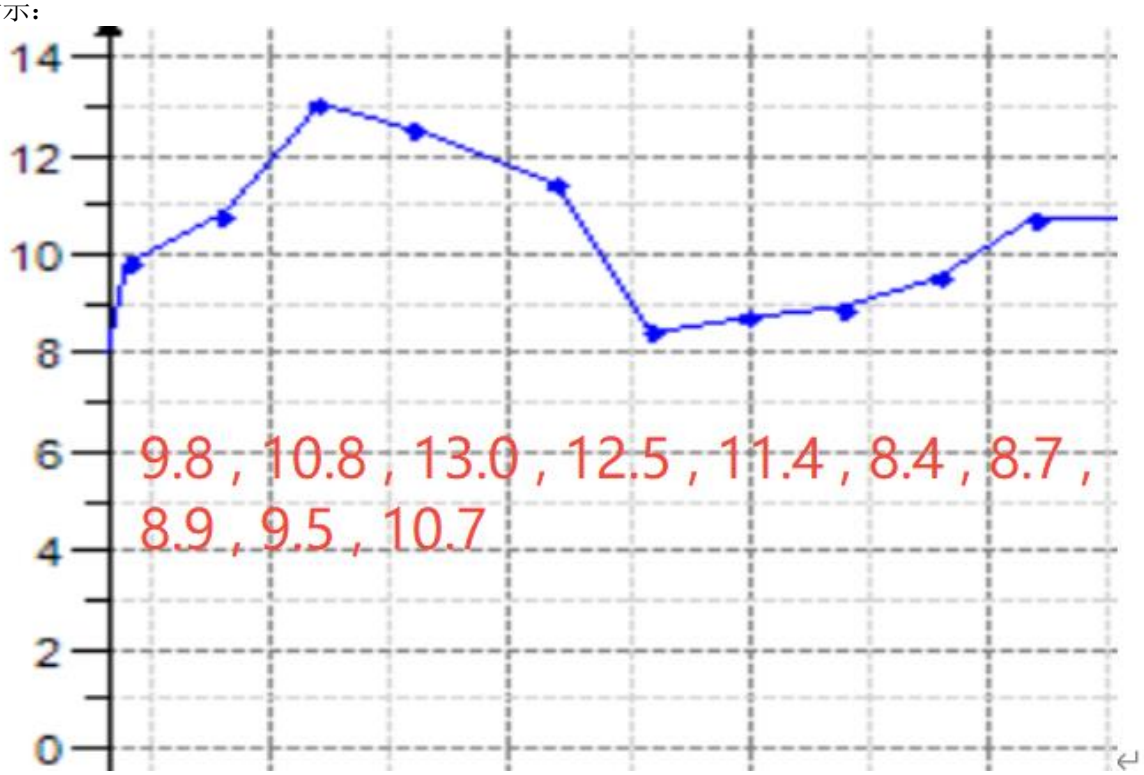


LD语言

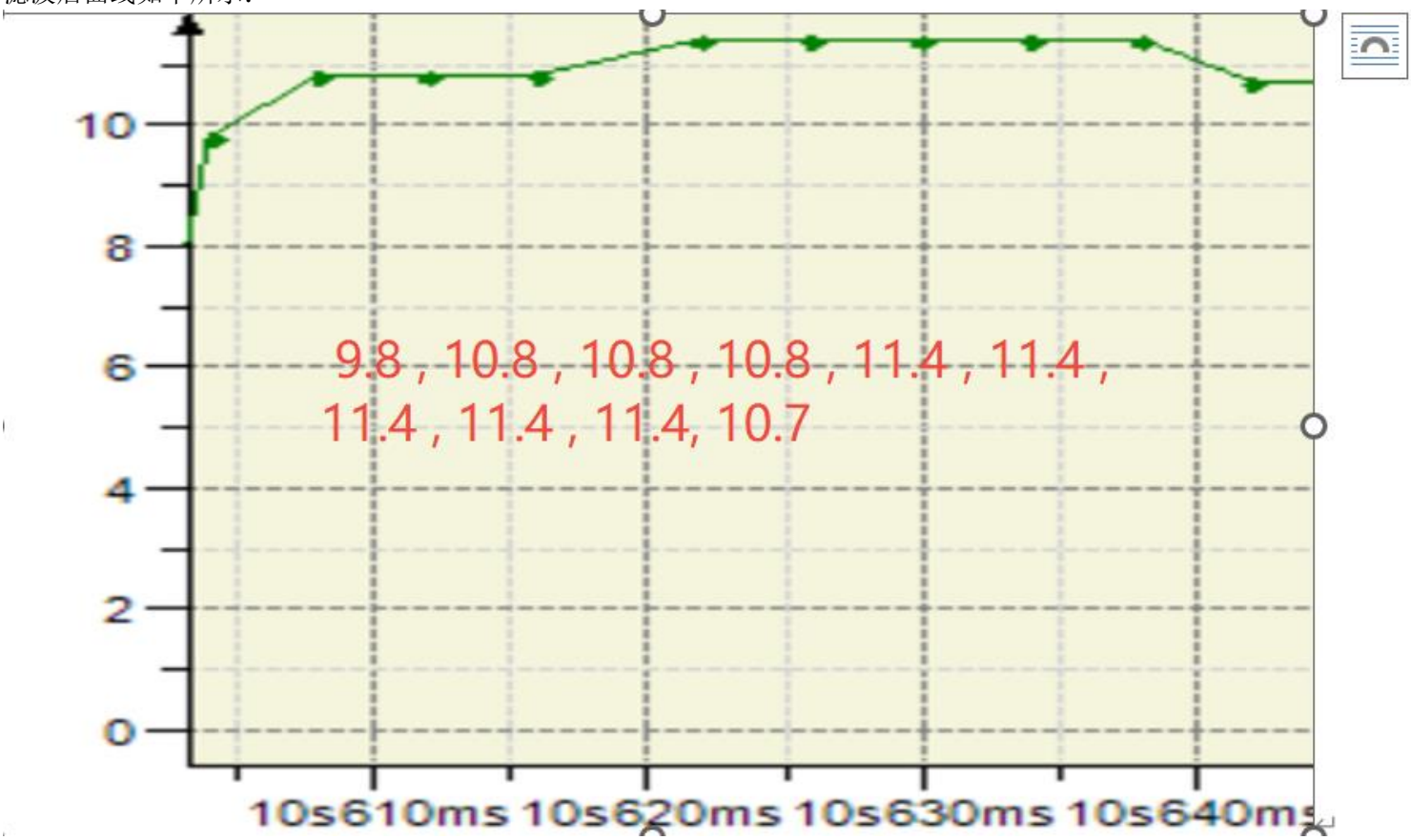


连续运行10个周期，采样值分别为(9.8 , 10.8 , 13.0 , 12.5 , 11.4 , 8.4 , 8.7 , 8.9 , 9.5 , 10.7)， 滤波指令 经过滤波后，结果分别为(9.8 , 10.8 , 10.8 , 10.8 , 11.4 , 11.4 , 11.4 , 11.4 , 11.4, 10.7)

滤波前曲线如下所示：



滤波后曲线如下所示：



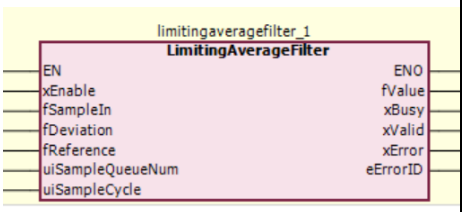
注意事项

1）滤波功能块中输入参数uiSampleCycle为采样周期，若uiSampleCycle=1时表示每个周期均进行采样 操作；若uiSampleCycle=3表示每3个周期进行采样一次。功能块的标志位LimitingFilter.xValid输出 TRUE则表示滤波有效。滤波输出值fValue在滤波输出有效时会更新输出。

2） 若采样数据为存放于数组的数据，需将数组中的数据逐个输入滤波功能块，需用户自行处理数据。

3.10.7 LimitingAverageFilter 限幅平均滤波

①指令格式

指令	功能	FB/FC	LD 表现	ST表现
LimitingAverageFilter	限幅平均滤波	FB		LimitingAverageFilter_0(xEnable:= , fSampleIn:= , fDeviation:= , fReference:= , uiSampleQueueNum := , uiSampleCycle:= , fValue=> , xBusy=> , xValid=> , xError=> , eErrorID=>);

②相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
xEnable	滤波使能	BOOL	[FALSE, TRUE]	FALSE	滤波使能
fSampleIn	输入采样值	REAL	-	0	输入采样值
fDeviation	输入两次采样允许的最大偏差值	REAL	-	0	输入两次采样允许的最大偏差值
fReference	输入采样参考值	REAL	-	0	输入采样参考值
uiSampleQueueNum	输入采样个数N	UINT	3-3000	0	输入采样个数N
uiSampleCycle	输入采样周期	UINT	1-1000	0	输入采样周期

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
fValue	滤波输出有效值	REAL	0	0	滤波输出有效值
xBusy	指令正在执行	BOOL	[FALSE, TRUE]	FALSE	指令正在执行
xValid	输出有效	BOOL	[FALSE, TRUE]	FALSE	输出有效
XError	错误	BOOL	[FALSE, TRUE]	FALSE	错误

数据类型

	布尔	位串	整数	实数	时刻、持续时间 日期、字符串

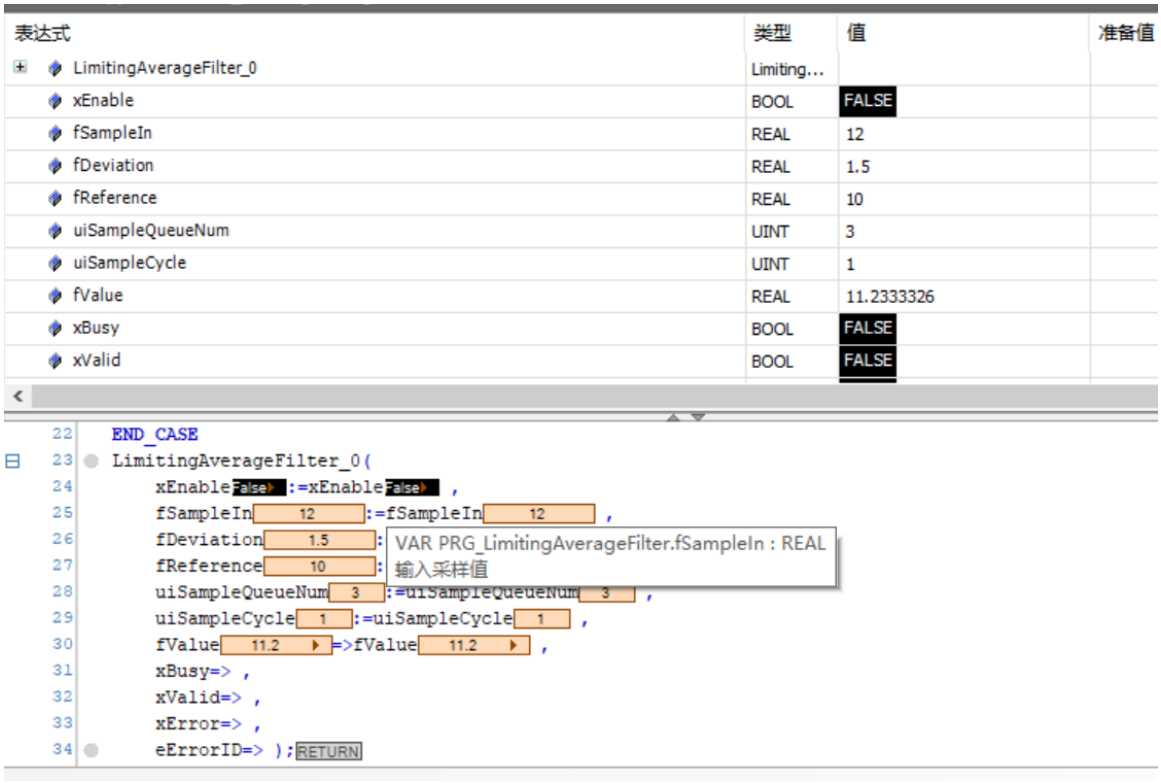
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
xEnable	√																			
fSampleIn														√						
fDeviation														√						
fReference														√						
uiSampleQueueNum							√													
uiSampleCycle							√													
fValue														√						
xBusy	√																			
xValid	√																			
XError	√																			

③功能说明

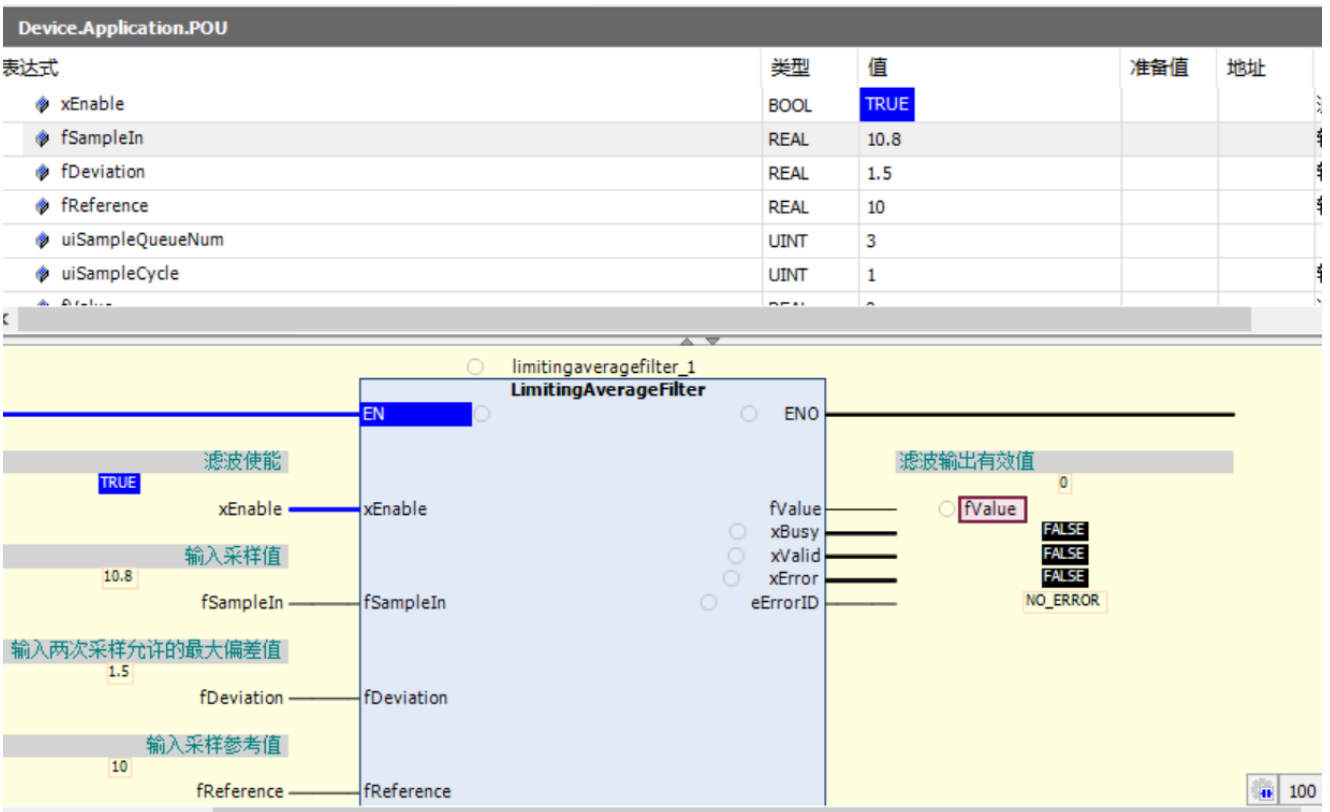
采样类别：连续采样，采样原始数据来源于连续更新的程序变量。 幅度允许最大偏差值为1.5（偏差范围，浮点型）。 幅度参考值为10.0（用户给定的参考值，浮点型）。 采样计算的样本个数为3（1~3000，无符号整型）。 采样周期为1（给定运行周期数执行一次数据采样，默认值为1）。 输入xFilter 为TRUE时，功能块能流有效开始采样滤波，设定初次参考值为10.0作为有效值，将采样值与上 次有效值比较。若两者之差的绝对值大于设定的最大偏差值1.5，则上次有效值即为本次有效值；若小于等于 最大偏差值则本次值为有效值。最后将有效值作为滤波值输出，滤波执行有效标志位xValid置TRUE。

④程序示例

ST语言

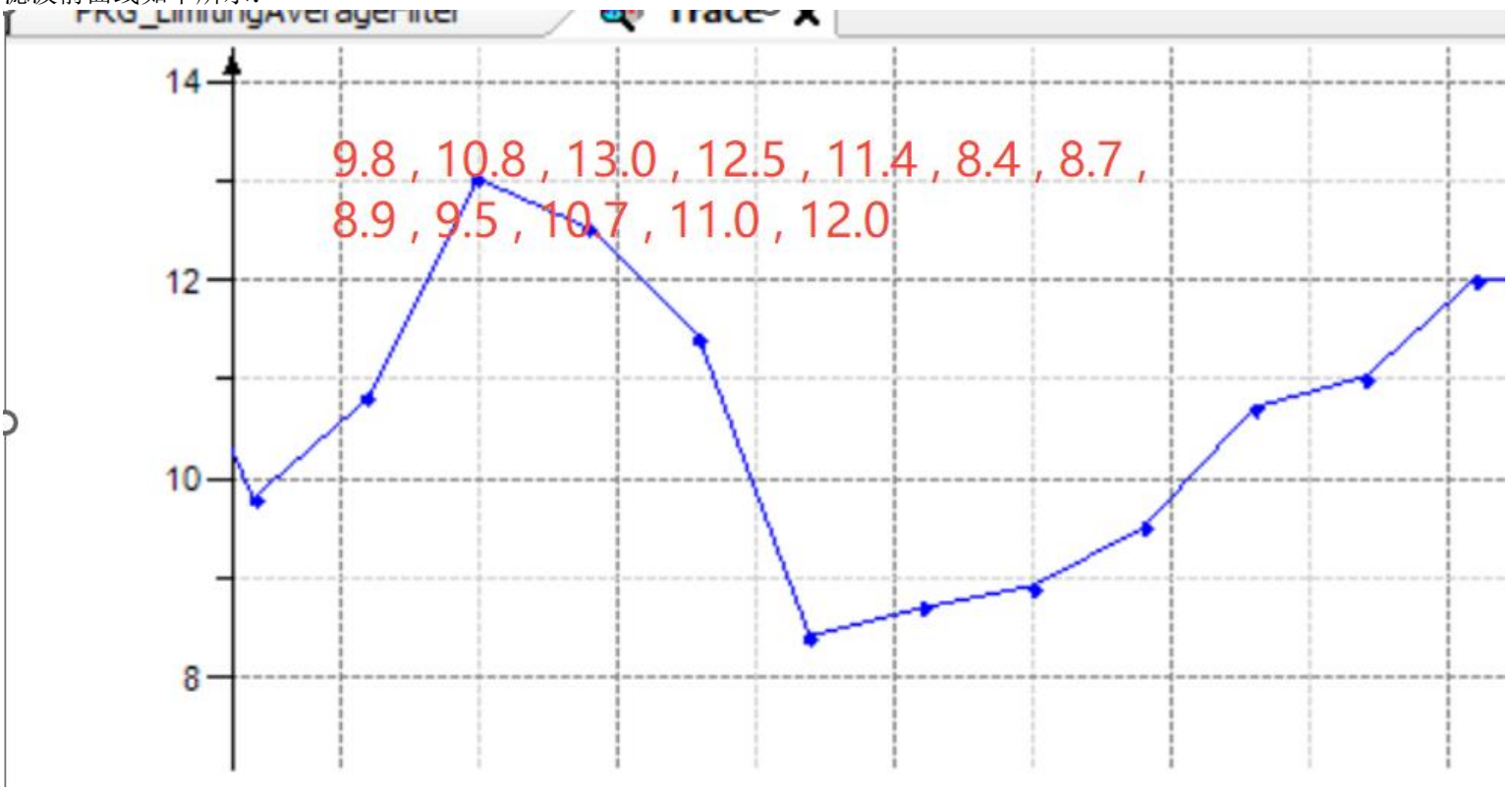


LD语言

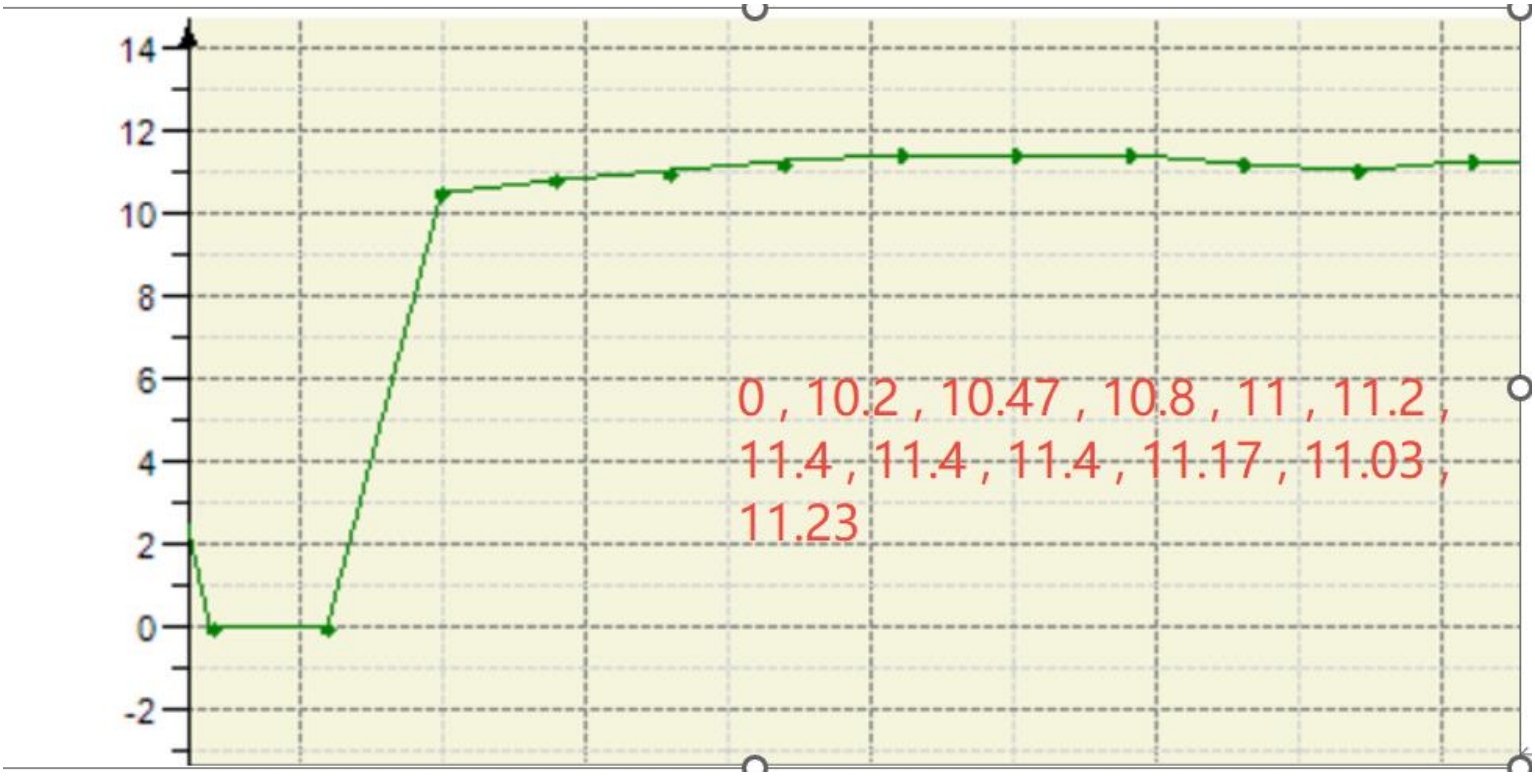


连续12个周期，采样值分别为(9.8 , 10.8 , 13.0 , 12.5 , 11.4 , 8.4 , 8.7 , 8.9 , 9.5 , 10.7 , 11.0 , 12.0)。 经过滤波后，结果分别为(0 , 10.2 , 10.47 , 10.8 , 11 , 11.2 , 11.4 , 11.4 , 11.4 , 11.17 , 11.03 , 11.23)。

滤波前曲线如下所示：



滤波后曲线如下所示：



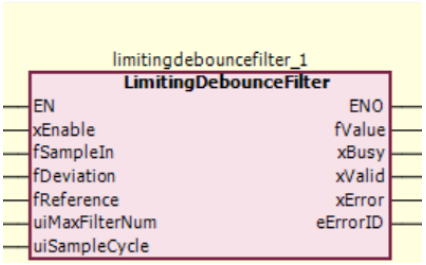
注意事项

1） 滤波功能块中输入参数uiSampleCycle为采样周期，若uiSampleCycle=1时表示每个周期均进行采样操 作，若uiSampleCycle=3表示每3个周期进行采样一次。功能块的标志位LimitingAverageFilter.xValid 输出TRUE则表示滤波有效。滤波输出值fValue在滤波输出有效时会更新输出。

2） 若采样数据为存放于数组的数据，需将数组中的数据逐个输入滤波功能块，需用户自行处理数据。

3.10.8 LimitingDebounceFilter 限幅消抖滤波

①指令格式

指令	功能	FB/FC	LD 表现	ST表现
LimitingDebounceFilter	限幅消抖滤波	FB		LimitingDebounceFilter_0 (xEnable:= , fSampleIn:= , fDeviation:= , fReference:= , uiMaxFilterNum:= , uiSampleCycle:= , fValue=> , xBusy=> , xValid=> , xError=> , eErrorID=>);

②相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
xEnable	滤波使能	BOOL	[FALSE, TRUE]	FALSE	滤波使能

fSampleIn	输入采样值	REAL	-	0	输入采样值
fDeviation	输入两次采样允许的最大偏差值	REAL	-	0	输入两次采样允许的最大偏差值
fReference	输入采样参考值	REAL	-	0	输入采样参考值
uiMaxFilterNum	最大计数值	UINT	3-3000	0	最大计数值
uiSampleCycle	输入采样周期	UINT	1-1000	0	输入采样周期

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
fValue	滤波输出有效值	REAL	0	0	滤波输出有效值
xBusy	指令正在执行	BOOL	[FALSE, TRUE]	FALSE	指令正在执行
xValid	输出有效	BOOL	[FALSE, TRUE]	FALSE	输出有效
XError	错误	BOOL	[FALSE, TRUE]	FALSE	错误

数据类型

	布尔	位串				整数							实数		时刻、持续时间 日期、字符串					
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
xEnable	√																			
fSampleIn														√						
fDeviation														√						
fReference														√						
uiMaxFilterNum							√													
uiSampleCycle							√													
fValue														√						
xBusy	√																			
xValid	√																			
XError	√																			

③功能说明

采样类别：连续采样，采样原始数据来源于连续更新的程序变量。 样本的参考有效值为10.0（用户给定的参考值，浮点型）。 样本值抖动允许最大计数器上限为3（抖动值允许个数，无符号号整型）。 采样周期为1（给定运行周期数执行一次数据采样，默认值为1）。 输入xFilter 为TRUE时，功能块能流有效开始采样滤波，设定参考值为10.0作为有效值，将采样值与当前有 效值比较，若相等则样本值抖动计数值清零；若采样值不等于当前有效值，则计数值加1，若计数器值大于最 大计数值上限，则将本次采样值替换当前有效值作为滤波有效值，将滤波值输出，滤波执行有效标志位xValid 置TRUE。

④程序示例

ST语言

表达式	类型	值	准备值	地址	注
LimitingDebounceFilter_0	Limiting...				
xEnable	BOOL	FALSE			滤
fSampleIn	REAL	10.7			输
fDeviation	REAL	1.5			输
fReference	REAL	10			输
uiMaxFilterNum	UINT	3			
uiSampleCycle	UINT	1			输
fValue	REAL	11.1			滤
xBusy	BOOL	FALSE			指
xValid	BOOL	FALSE			输

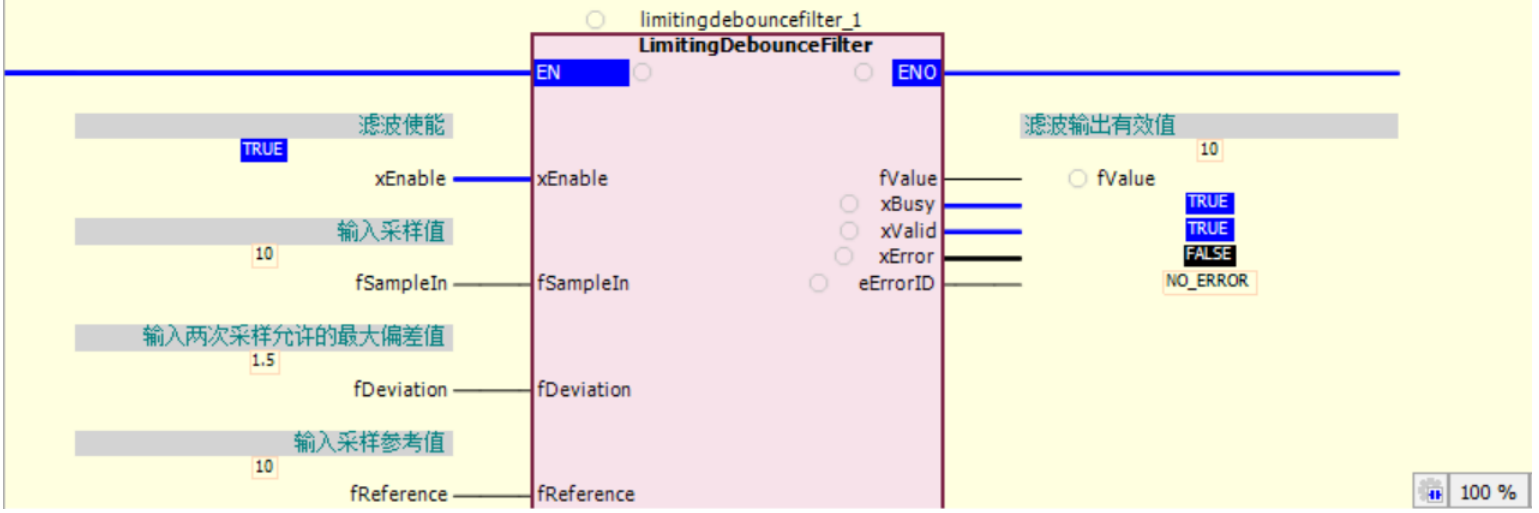
```
22  END_CASE
23  LimitingDebounceFilter_0(
24      xEnable(False) :=xEnable(False) ,
25      fSampleIn(10.7) :=fSampleIn(10.7) ,
26      fDeviation(1.5) :=fDeviation(1.5) ,
27      fReference(10) := fReference(10),
28      uiMaxFilterNum(3) := uiMaxFilterNum(3),
29      uiSampleCycle(1) := uiSampleCycle(1),
30      fValue(11.1) => fValue(11.1),
31      xBusy=> ,
32      xValid=> ,
33      xError=> ,
34      eErrorID=> );RETURN
```

100 %

LD语言

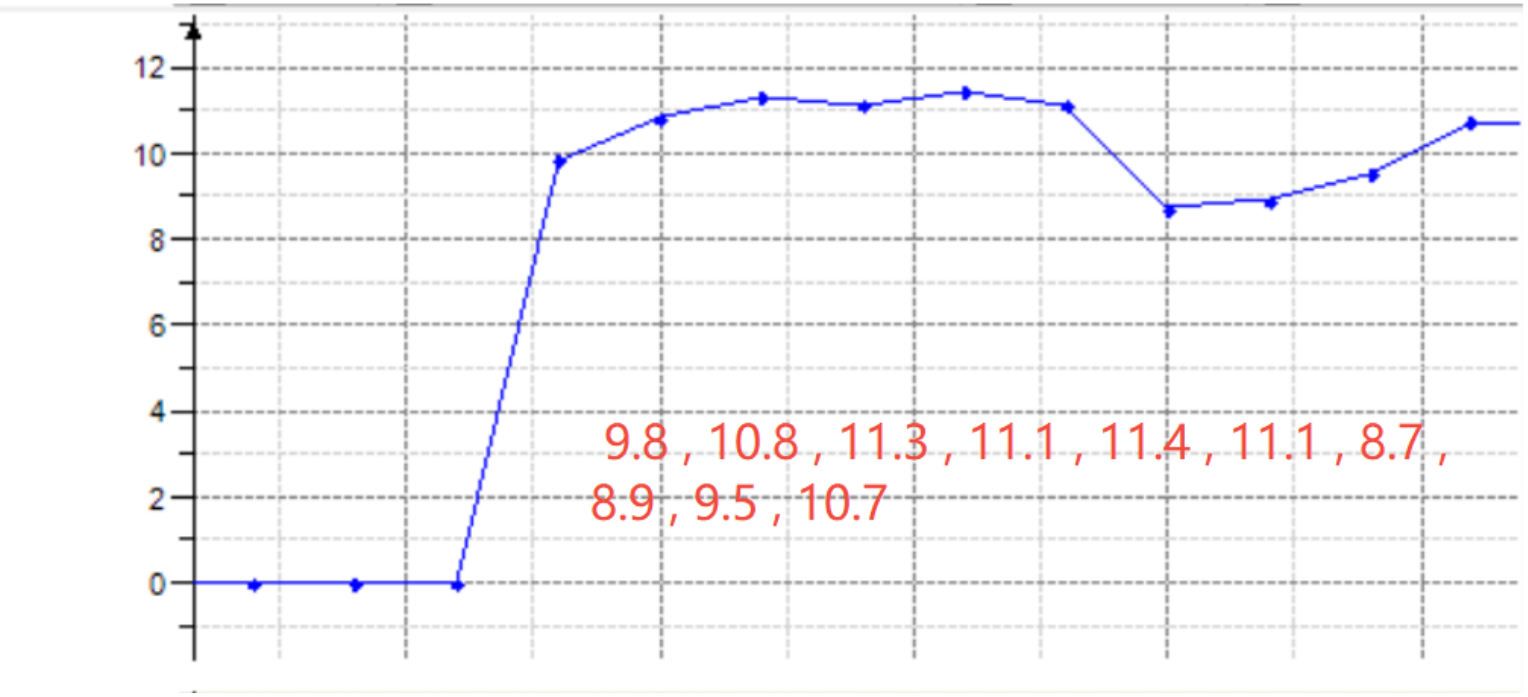
Device.Application.POU

表达式	类型	值	准备值	地址	注
xEnable	BOOL	TRUE			滤
fSampleIn	REAL	10			输
fDeviation	REAL	1.5			输
fReference	REAL	10			输
uiMaxFilterNum	UINT	3			
uiSampleCycle	UINT	1			输

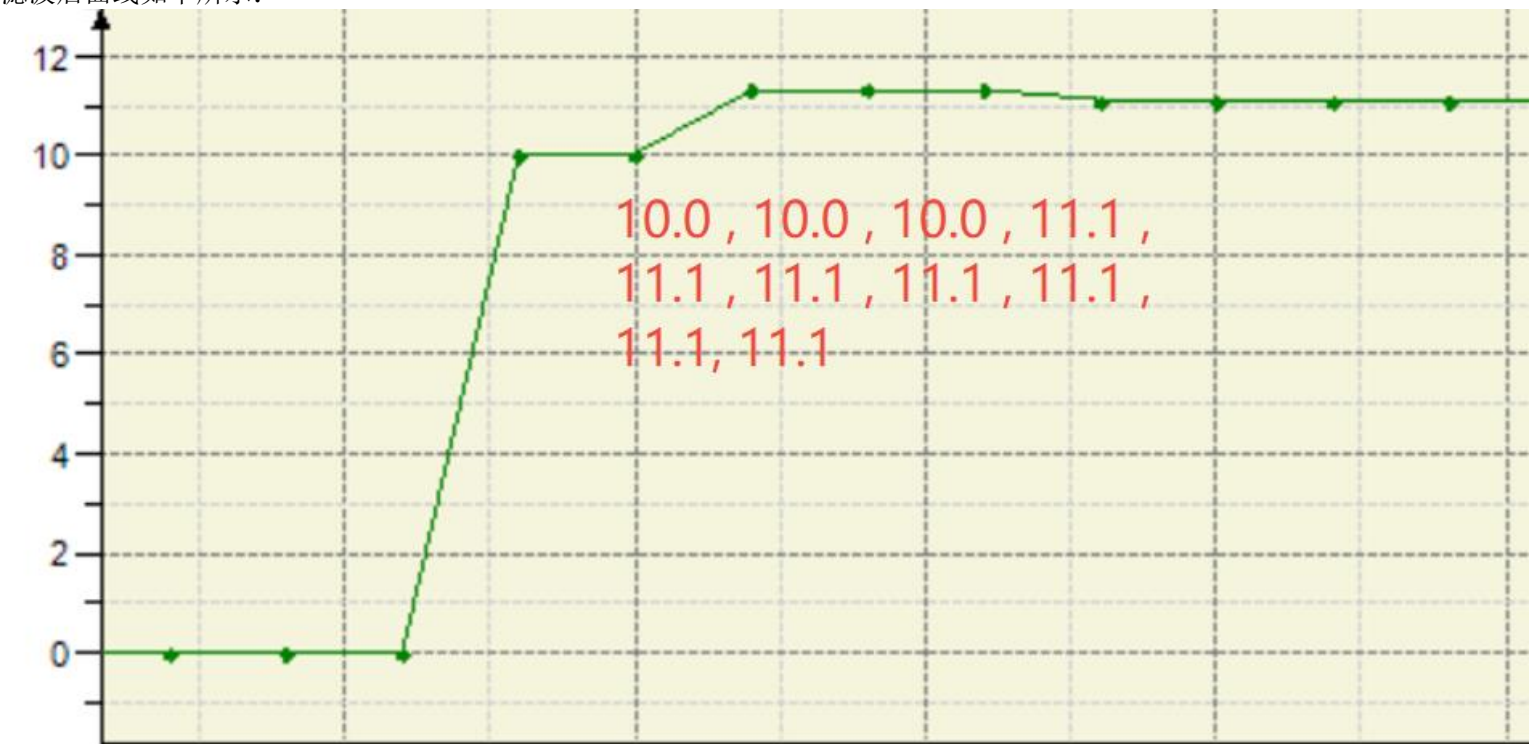


连续10个周期，采样值分别为(9.8 , 10.8 , 11.3 , 11.1 , 11.4 , 11.1 , 8.7 , 8.9 , 9.5 , 10.7)， 经过滤波后，结果分别为(10.0 , 10.0 , 10.0 , 11.1 , 11.1 , 11.1 , 11.1 , 11.1 , 11.1, 11.1)。

滤波前曲线如下所示：



滤波后曲线如下所示：



注意事项

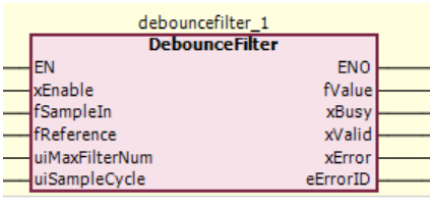
1） 滤波功能块中输入参数uiSampleCycle为采样周期，若uiSampleCycle=1时表示每个周期均进行采样 操作；若uiSampleCycle=3表示每3个周期进行采样一次。功能块的标志位LimitingDebounceFilter. xValid 输出TRUE则表示滤波有效。滤波输出值fValue在滤波输出有效时会更新输出。

2） 若采样数据为存放于数组的数据，需将数组中的数据逐个输入滤波功能块，需用户自行处理数据。

3.10.9 DebounceFilter 消抖滤波

①指令格式

指令	功能	FB/FC	LD 表现	ST表现

DebounceFilter	消抖滤波	FB		DebounceFilter_0(xEnable:= , fSampleIn:= , fReference:= , uiMaxFilterNum:= , uiSampleCycle:= , fValue=> , xBusy=> , xValid=> , xError=> , eErrorID=>);

②相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
xEnable	滤波使能	BOOL	[FALSE, TRUE]	FALSE	滤波使能
fSampleIn	输入采样值	REAL	-	0	输入采样值
fReference	输入采样参考值	REAL	-	0	输入采样参考值
uiMaxFilterNum	最大计数值	UINT	3-3000	0	最大计数值
uiSampleCycle	输入采样周期	UINT	1-1000	0	输入采样周期

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
fValue	滤波输出有效值	REAL	0	0	滤波输出有效值
xBusy	指令正在执行	BOOL	[FALSE, TRUE]	FALSE	指令正在执行
xValid	输出有效	BOOL	[FALSE, TRUE]	FALSE	输出有效
XError	错误	BOOL	[FALSE, TRUE]	FALSE	错误

数据类型

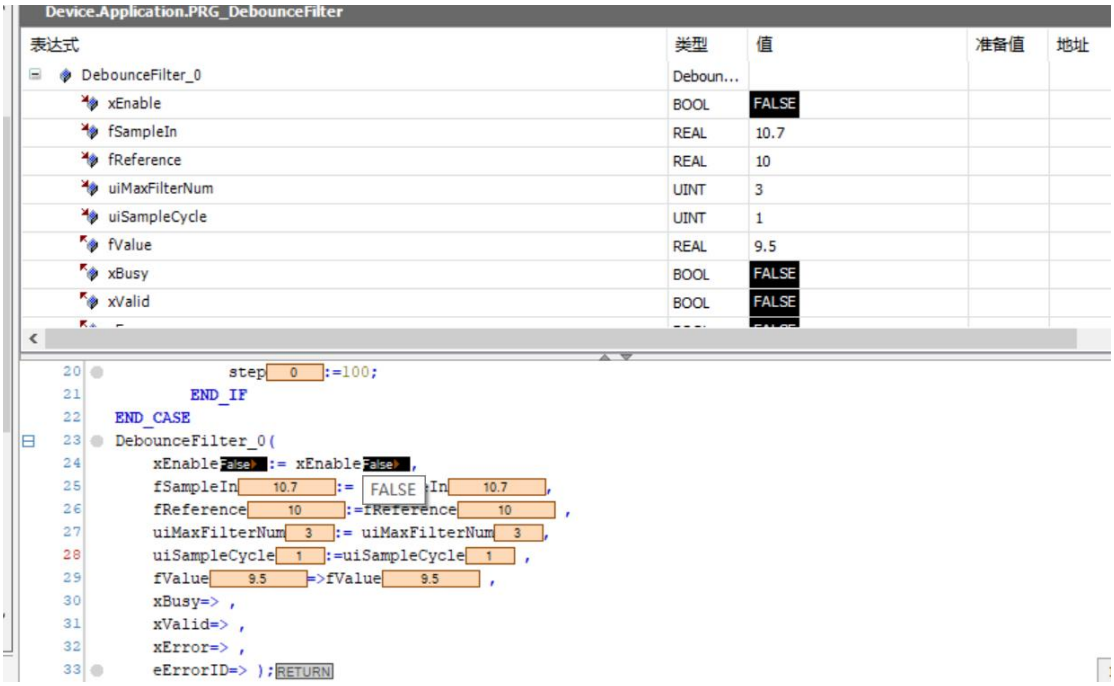
	布尔	位串				整数							实数		时刻、持续时间 日期、字符串					
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
xEnable	√																			
fSampleIn														√						
fReference														√						
uiMaxFilterNum							√													
uiSampleCycle							√													
fValue														√						
xBusy	√																			
xValid	√																			
XError	√																			

③功能说明

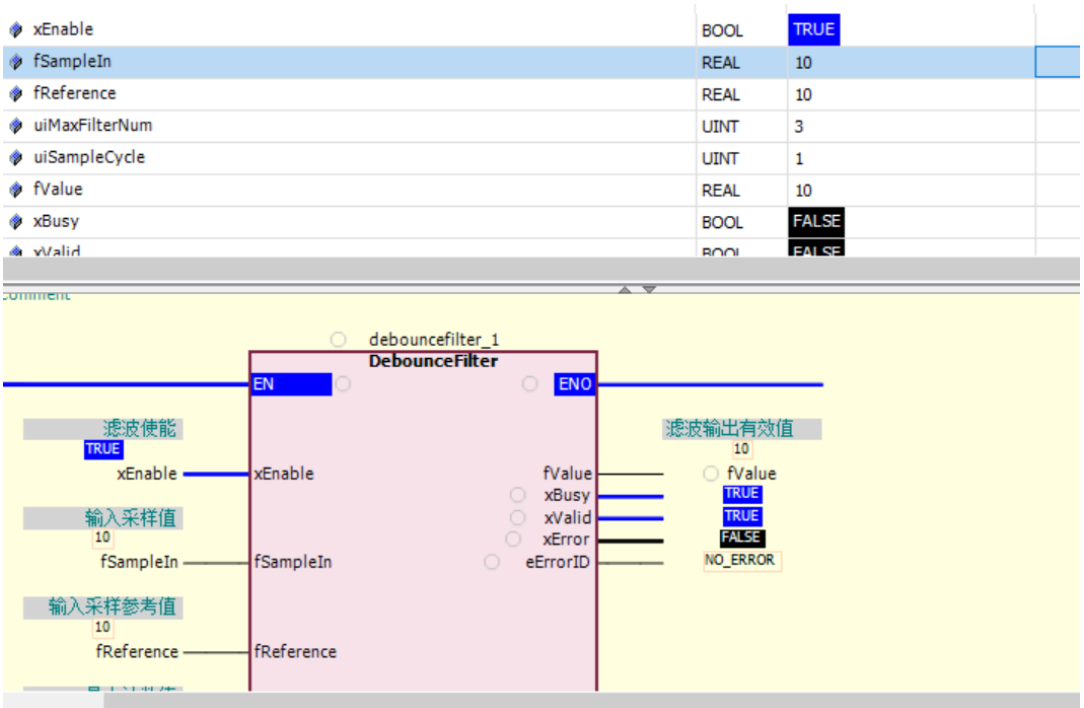
采样类别：连续采样，采样原始数据来源于连续更新的程序变量。 样本的参考有效值为10.0（用户给定的参考值，浮点型）。 样本值抖动允许最大计数器上限为3（抖动值允许个数，无符号号整型） 采样周期为1（给定运行周期数执行一次数据采样，默认值为1）。 输入xFilter 为TRUE时，功能块能流有效开始采样滤波，设定参考值为10.0作为有效值，将采样值与当前有 效值比较，若相等则样本值抖动计数器清零；若采样值不等于当前有效值，则计数器加1，若计数器值大于最 大计数器上限，则将本次采样值替换当前有效值作为滤波有效值，将滤波值输出，滤波执行有效标志位xValid 置TRUE。

④程序示例

ST语言



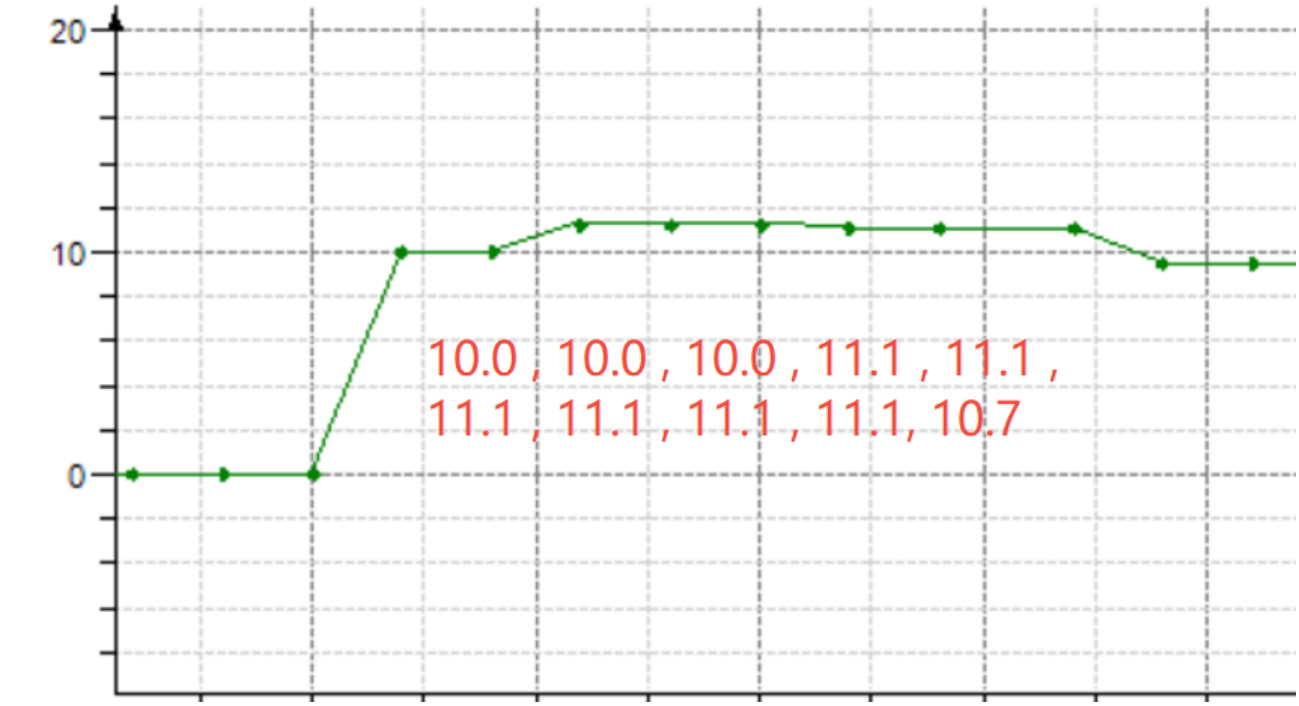
LD语言



连续10个周期，采样值分别为(9.8 , 10.8 , 11.3 , 11.1 , 11.4 , 11.1 , 8.7 , 8.9 , 9.5 , 10.7), 经过滤波后，结果分别为(10.0 , 10.0 , 10.0 , 11.1 , 11.1 , 11.1 , 11.1 , 11.1 , 11.1, 10.7)。

滤波前曲线如下所示：

滤波后曲线如下所示：



注意事项

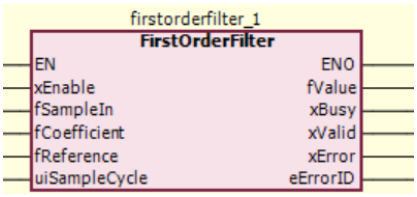
1） 滤波功能块中输入参数uiSampleCycle为采样周期，若uiSampleCycle=1时表示每个周期均进行采样操 作；若uiSampleCycle=3表示每3个周期进行采样一次。功能块的标志位DebounceFilter.xValid输出 TRUE则表示滤波有效。滤波输出值fValue在滤波输出有效时会更新输出

2） 若采样数据为存放于数组的数据，需将数组中的数据逐个输入滤波功能块，需用户自行处理数据。。

3.10.10 FirstOrderFilter 一阶滤波器

①指令格式

指令	功能	FB/FC	LD 表现	ST表现

FirstOrderFilter	一阶滤波器	FB		FirstOrderFilter_0(xEnable:= , fSampleIn:= , fCoefficient:= , fReference:= , uiSampleCycle:= , fValue=> , xBusy=> , xValid=> , xError=> , eErrorID=>);

②相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
xEnable	滤波使能	BOOL	[FALSE, TRUE]	FALSE	滤波使能
fSampleIn	输入采样值	REAL	-	0	输入采样值
fCoefficient	输入一阶低通滤波系数a=0~1	REAL	0-1	0	输入一阶低通滤波系数a=0~1
fReference	输入采样参考值	REAL	-	0	输入采样参考值
uiSampleCycle	输入采样周期	UINT	1-1000	0	输入采样周期

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
fValue	滤波输出有效值	REAL	0	0	滤波输出有效值
xBusy	指令正在执行	BOOL	[FALSE, TRUE]	FALSE	指令正在执行
xValid	输出有效	BOOL	[FALSE, TRUE]	FALSE	输出有效
XError	错误	BOOL	[FALSE, TRUE]	FALSE	错误

数据类型

	布尔	位串				整数							实数		时刻、持续时间 日期、字符串					
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
xEnable	√																			
fSampleIn														√						
fCoefficient														√						
fReference														√						
uiSampleCycle							√													
fValue														√						
xBusy	√																			

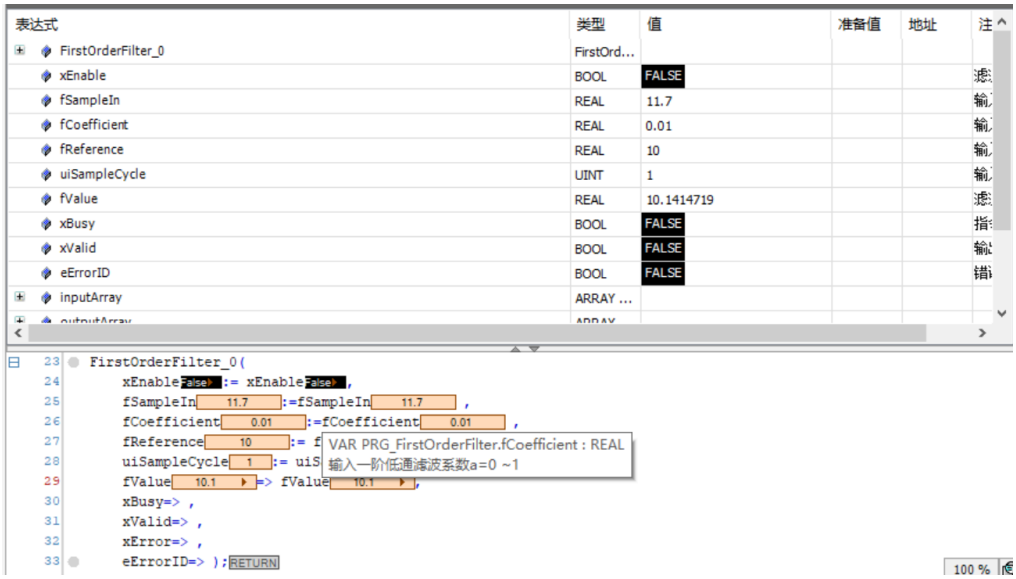
xValid	√																			
XError	√																			

③功能说明

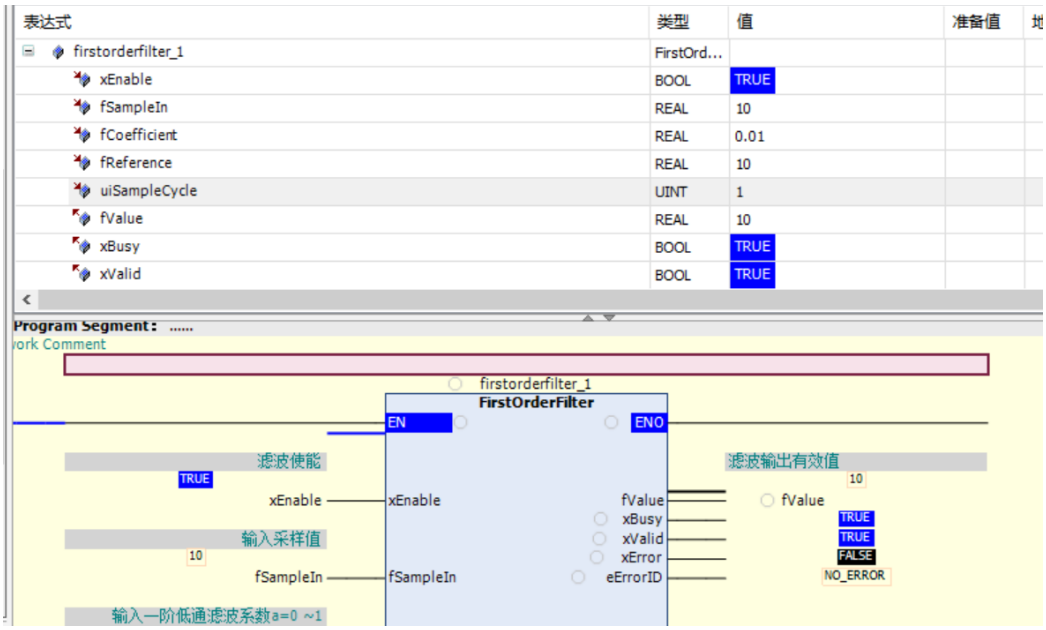
采样类别：连续采样，采样原始数据来源于连续更新的程序变量。 一阶低通滤波系数为0.01（浮点型）。 滤波参考值为10.0， 采样周期为1（即每周期数据采样一次）。 输入xFilter 为TRUE时，功能块能流有效开始采样滤波，设定初次参考值为10.0作为有效值，一阶低通滤波 系数为0.01，将滤滤波值=fcoefficient*本次采样值+（1-fcoefficient)*上次滤波结果，将滤波值输出,滤波 执行有效标志位xValid置TRUE。

④程序示例

ST语言

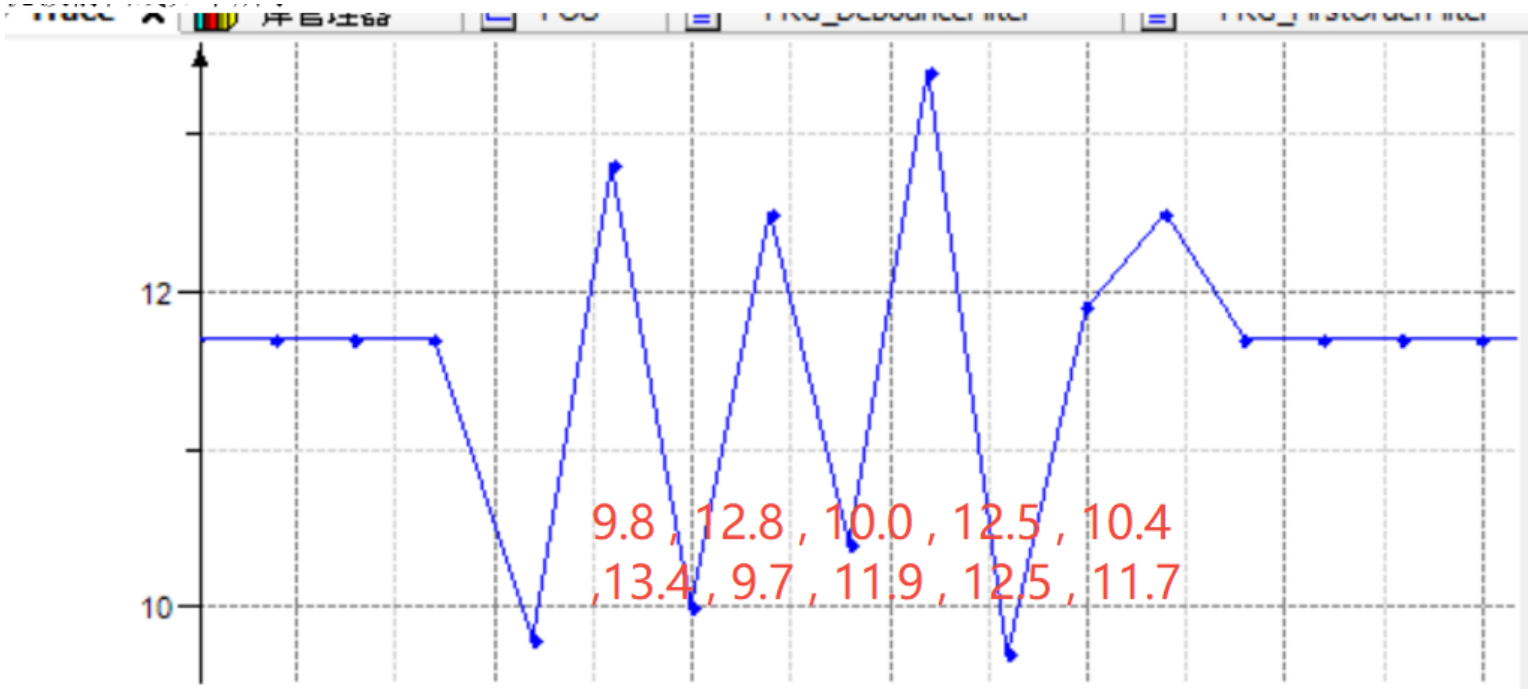


LD语言

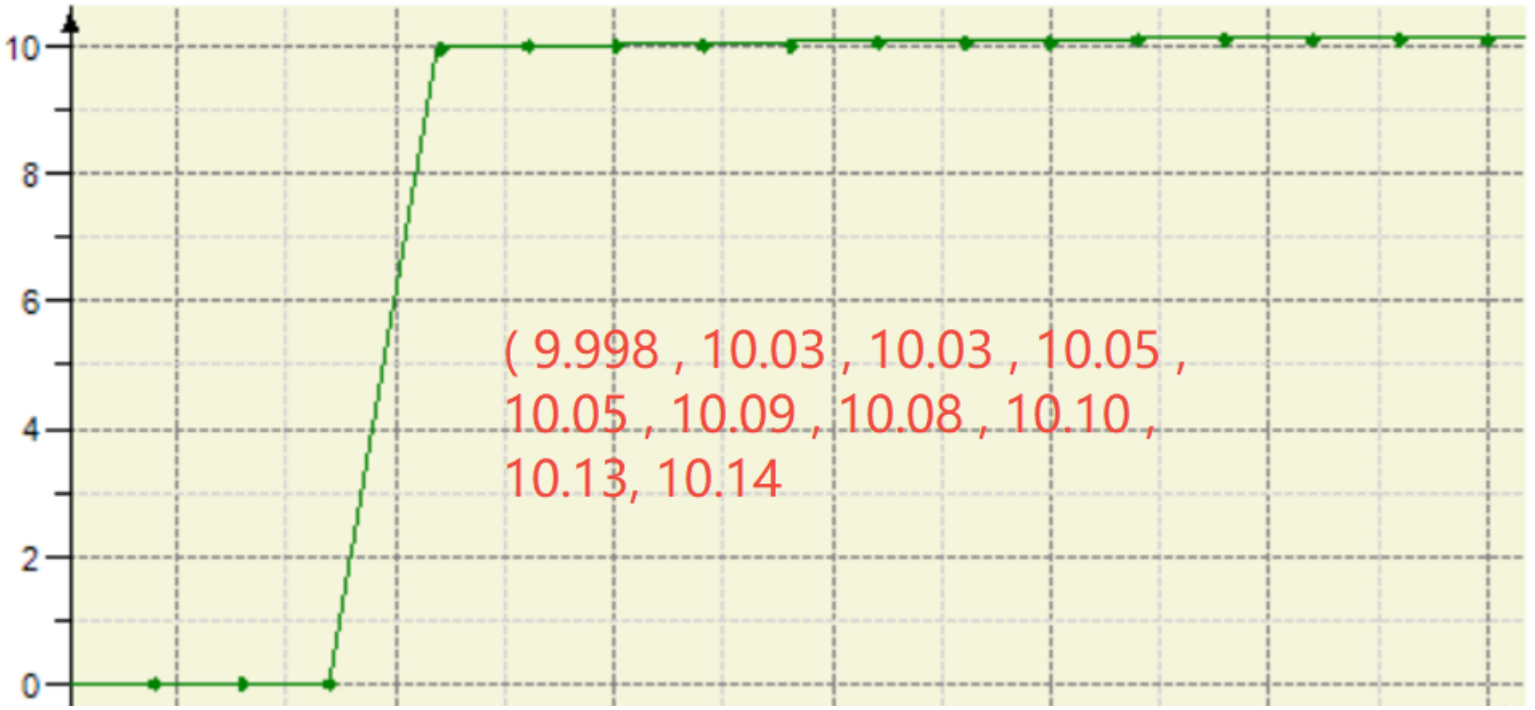


连续10个周期，采样值分别为(9.8 , 12.8 , 10.0 , 12.5 , 10.4 ,13.4 , 9.7 , 11.9 , 12.5 , 11.7), 经过滤波后，结果分别为(9.998 , 10.03 , 10.03 , 10.05 , 10.05 , 10.09 , 10.08 , 10.10 , 10.13, 10.14)。

滤波前曲线如下所示：



滤波后曲线如下所示：



注意事项

1） 滤波功能块中输入参数uiSampleCycle为采样周期，若uiSampleCycle=1时表示每个周期均进行采样操 作；若uiSampleCycle=3表示每3个周期进行采样一次。功能块的标志位FirstOrderLagFilter.xValid输 出TRUE则表示滤波有效。滤波输出值fValue在滤波输出有效时会更新输出。

2） 若采样数据为存放于数组的数据，需将数组中的数据逐个输入滤波功能块，需用户自行处理数据。

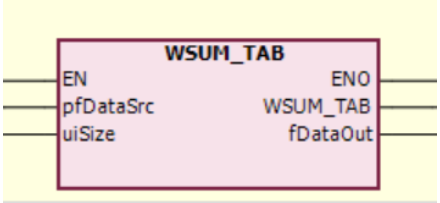
3.11 表格和区间

指令列表

指令类别	名称	FB/FC	功能
表格和区间	WSUM_TAB	FC	数据总和计算
	MEAN_TAB	FC	平均值计算
	BZAND_TAB	FC	死区控制
	ZONE_TAB	FC	区域控制
	SCL_TAB	FC	定坐标（不同点坐标数据）
	ZRST_TAB	FC	全部数据复位
	SORT_TAB	FC	数据排序
	RAMP_TAB	FB	斜坡指令
	RecSearch	FB	记录检索
	RecRangeSearch	FB	范围指定记录检索
	RecSort	FB	记录分类
	RecNum	FC	记录数获取
	RecMax	FC	记录最大值检索
	RecMin	FC	记录最小值检索

3.11.1 WSUM_TAB 数据总和计算

①指令格式

指令	功能	FB/FC	LD 表现	ST表现
WSUM_TAB	数据总和计算	FC		WSUM_TAB(WSUM_TAB=> , pfDataSrc:= , uiSize:= , fDataOut=>);

②相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
pfDataSrc	源数据	POINTER TO REAL	–	-	源数据
uiSize	数据长度	UINT	1- 1024	0	数据长度

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
fDataOut	目标数据	REAL	-	-	总和求值

数据类型

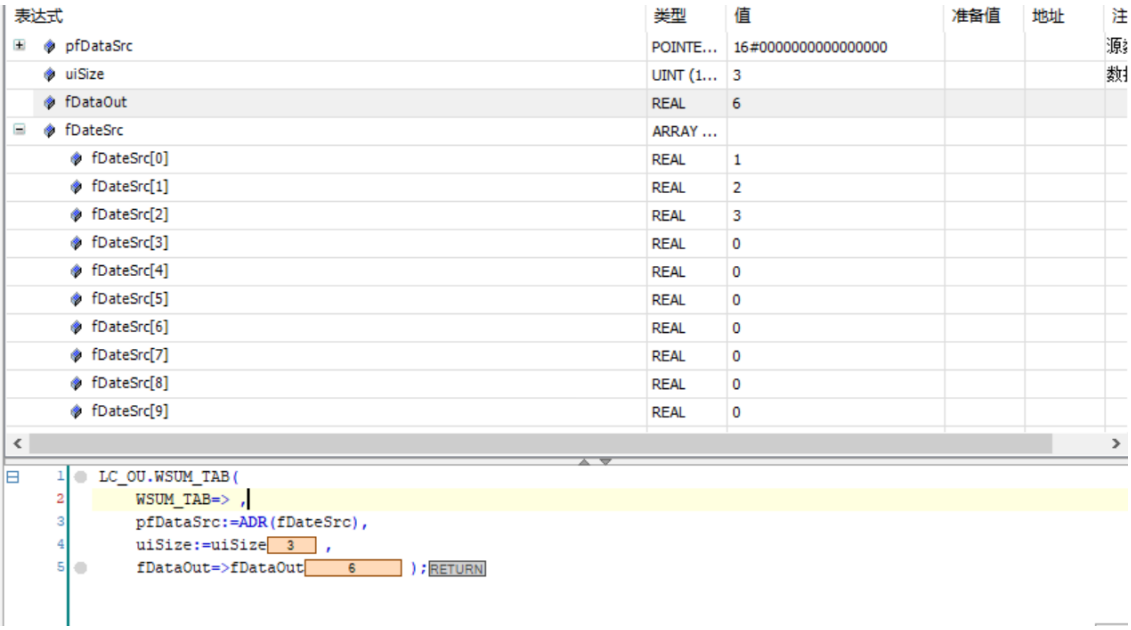
	布尔	位串				整数							实数		时刻、持续时间 日期、字符串					
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
pfDataSrc	POINTER TO REAL																			
uiSize							√													
fDataOut														√						

③功能说明

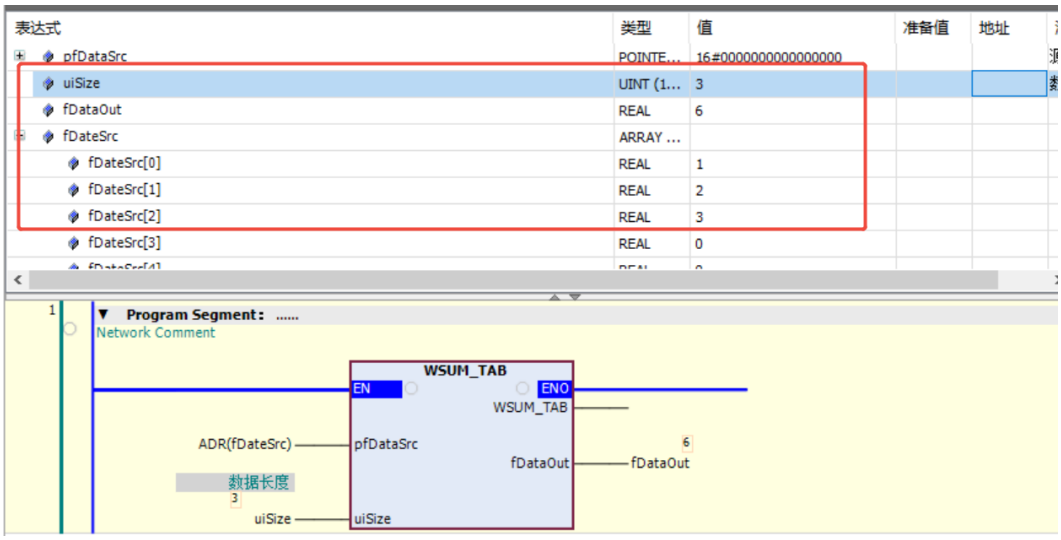
计算指定数量的表格数据的总和值。

④程序示例

ST语言

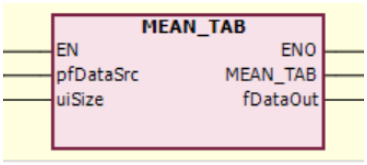


LD语言



3.11.2 MEAN_TAB 平均值计算

①指令格式

指令	功能	FB/FC	LD 表现	ST表现
WSUM_TAB	平均值计算	FC		MEAN_TAB(MEAN_TAB=> , pfDataSrc:= , uiSize:= , fDataOut=>);

②相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
pfDataSrc:	源数据	POINTER TO REAL	-	-	源数据
uiSize	数据长度	UINT	1- 1024	0	数据长度

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
fDataOut	目标数据	REAL	-	-	求平均值

数据类型

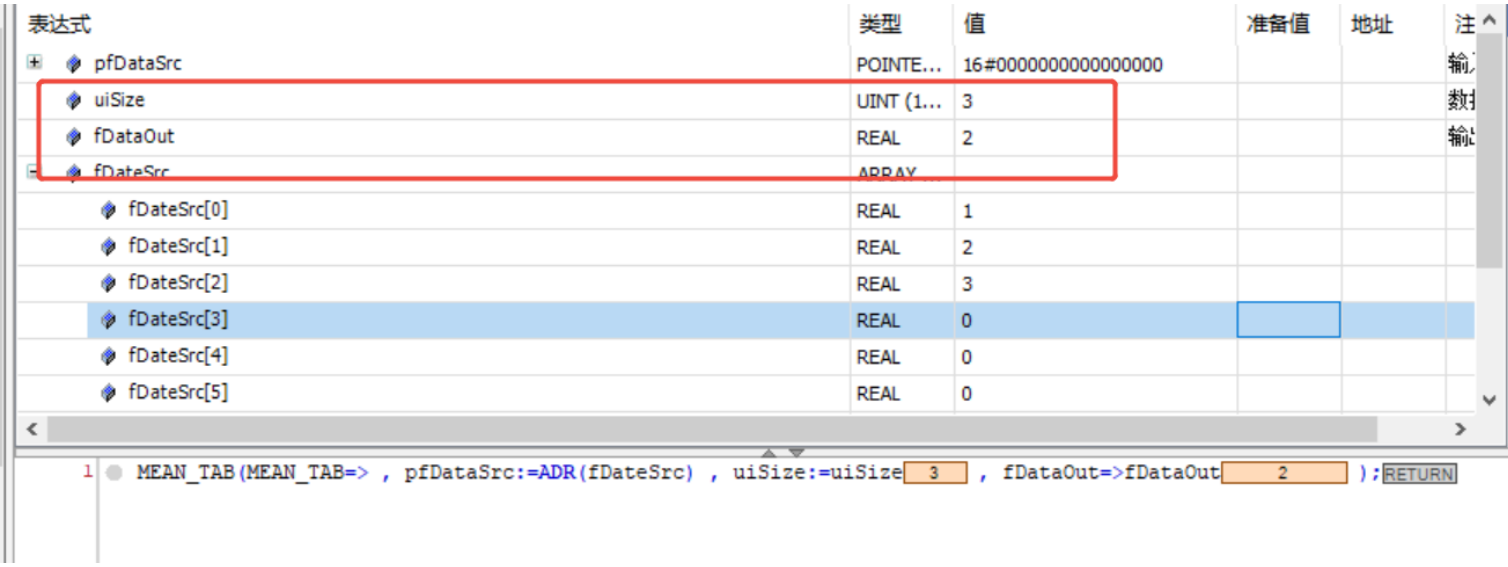
	布尔	位串				整数							实数		时刻、持续时间 日期、字符串					
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
pfDataSrc	POINTER TO REAL																			
uiSize							√													
fDataOut														√						

③功能说明

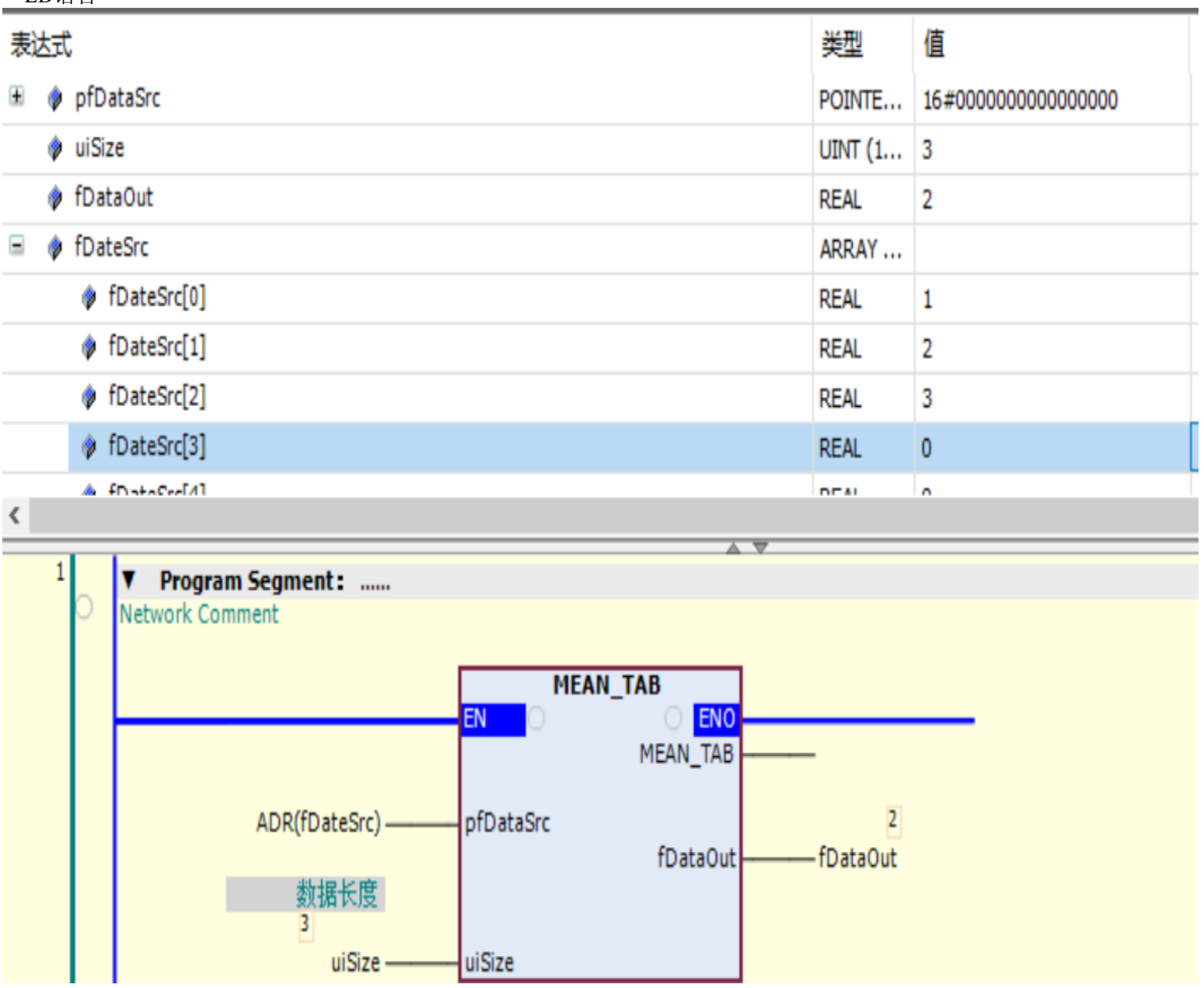
计算指定数据的平均值(对 pfDataSrc开始的uiSize个数进行求平均值操作，结果存储于fDataOut)。

④程序示例

ST语言

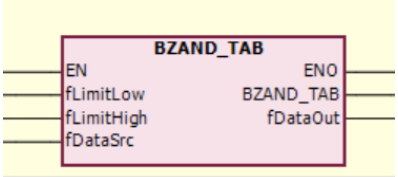

表达式
类型
值
准备值
地址
pfDataSrc
POINTE...
16#0000000000000000
uiSize
UINT (1...
3
fDataOut
REAL
2
fDateSrc
ARRAY ...
fDateSrc[0]
REAL
1
fDateSrc[1]
REAL
2
fDateSrc[2]
REAL
3
fDateSrc[3]
REAL
0
fDateSrc[4]
REAL
0
fDateSrc[5]
REAL
0
1
MEAN_TAB(MEAN_TAB=> , pfDataSrc:=ADR(fDateSrc) , uiSize:=uiSize 3 , fDataOut=>fDataOut 2);RETURN

LD语言


表达式
类型
值
pfDataSrc
POINTE...
16#0000000000000000
uiSize
UINT (1...
3
fDataOut
REAL
2
fDateSrc
ARRAY ...
fDateSrc[0]
REAL
1
fDateSrc[1]
REAL
2
fDateSrc[2]
REAL
3
fDateSrc[3]
REAL
0
1
Program Segment: .....
Network Comment
MEAN_TAB
EN
ENO
MEAN_TAB
ADR(fDateSrc)
pfDataSrc
2
fDataOut
fDataOut
数据长度
3
uiSize
uiSize

3.11.3 BZAND_TAB 死区控制

①指令格式

指令	功能	FB/FC	LD 表现	ST表现
BZAND_TAB	死区控制	FC		BZAND_TAB(BZAND_TAB=> , fLimitLow:= , fLimitHigh:= , fDataSrc:= , fDataOut=>);

②相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
fLimitHigh	正偏差	REAL	-	0	正偏差
fLimitLow	负偏差	REAL	-	0	负偏差
fDataSrc	输入源数据	REAL	-	0	输入源数据

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
fDataOut	输出值	REAL	-	-	输出值

数据类型

	布尔	位串				整数							实数		时刻、持续时间 日期、字符串					
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
FLimitLow														√						
FLimitHigh														√						
fDataSrc														√						
fDataOut														√						

③功能说明

通过判断输入值是否在指定的死区的上下限范围内，从而来控制输出值;通过在 fLimitLow、 fLimitHigh中设 定死区范围，使输入值 fDataSrc 在死区范围外输出至fDataOut 中。

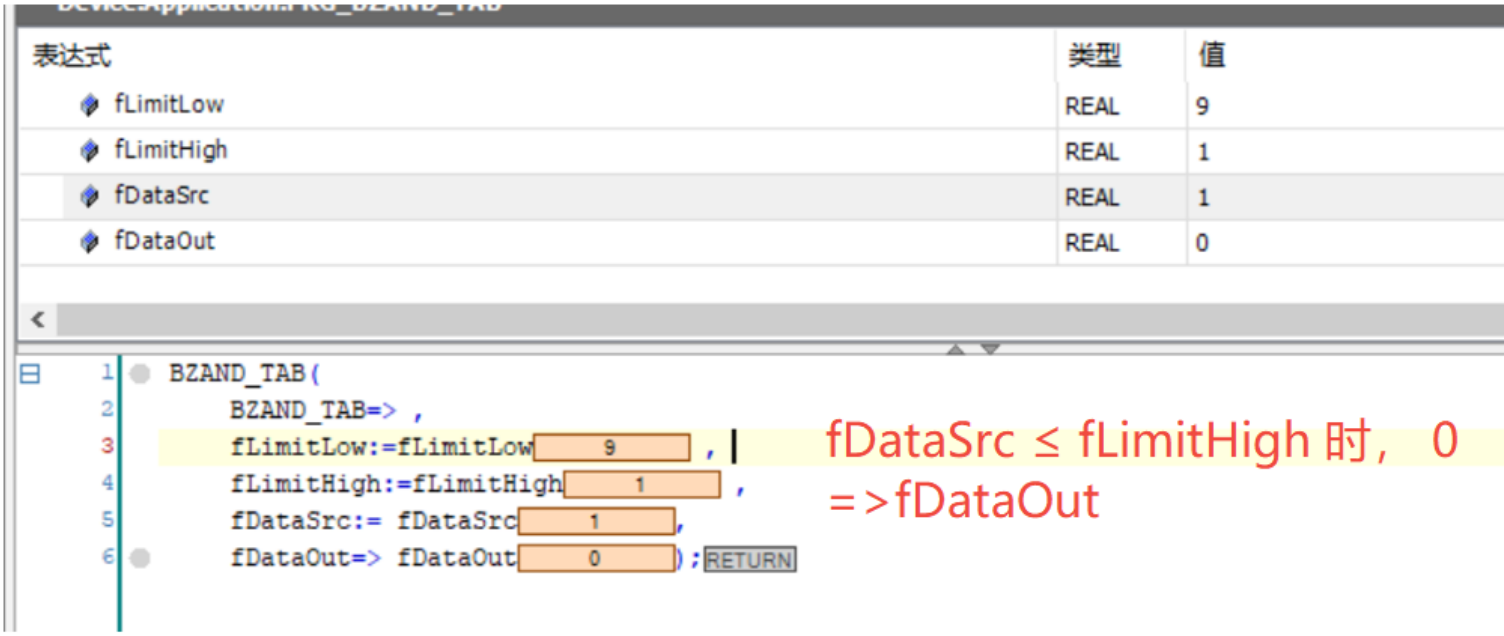
当 fLimitLow ＞ fDataSrc 时， fDataSrc - fLimitLow => fDataOut；

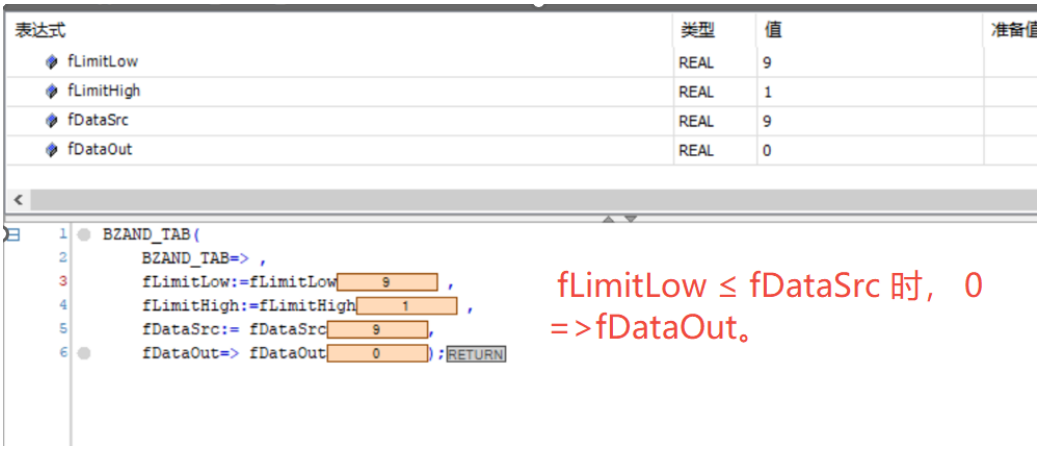
当 fLimitHigh ＜ fDataSrc 时， fDataSrc - fLimitHigh =>fDataOut；

当 fLimitLow ≤ fDataSrc ≤ fLimitHigh 时， 0 =>fDataOut。

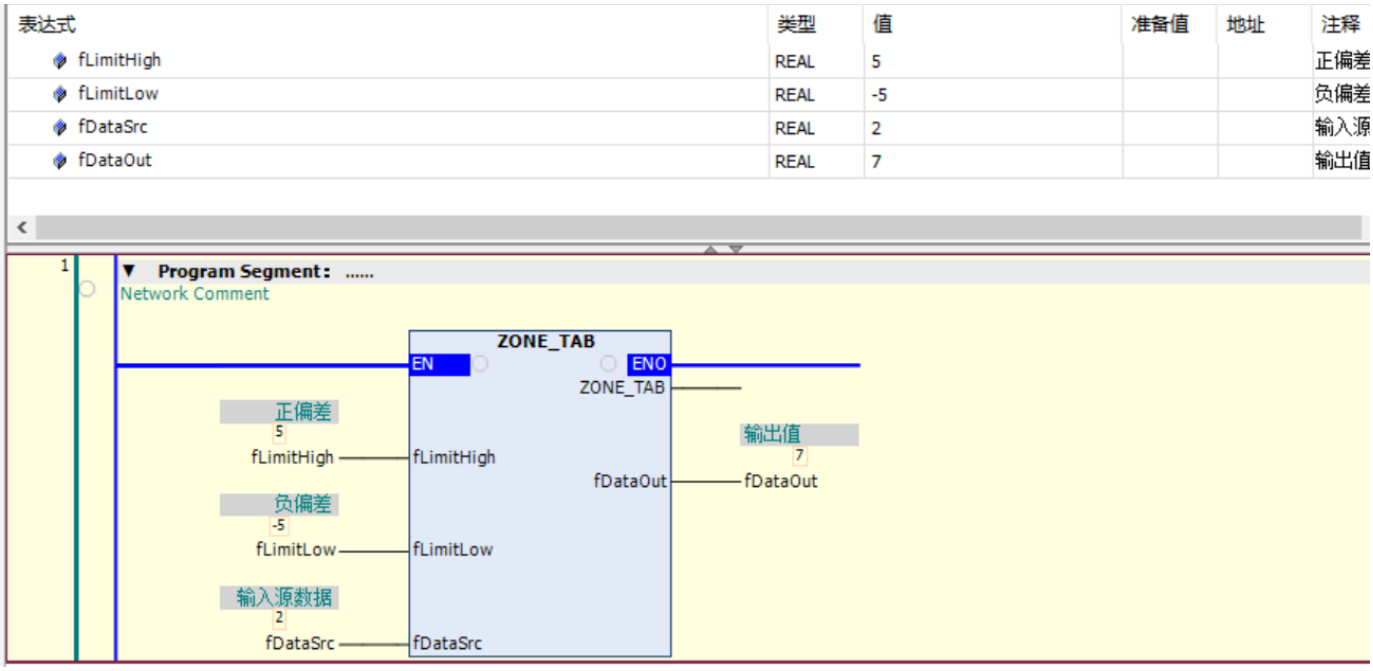
④程序示例

ST语言


表达式
类型
值
fLimitLow
REAL
9
fLimitHigh
REAL
1
fDataSrc
REAL
1
fDataOut
REAL
0
BZAND_TAB(
BZAND_TAB=> ,
fLimitLow:=fLimitLow 9 ,
fLimitHigh:=fLimitHigh 1 ,
fDataSrc:= fDataSrc 1 ,
fDataOut=> fDataOut 0);RETURN
fDataSrc ≤ fLimitHigh 时， 0 =>fDataOut

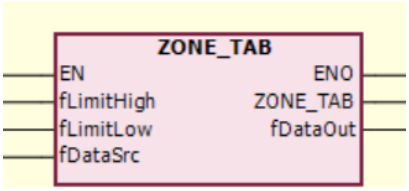

表达式
类型
值
fLimitLow
REAL
9
fLimitHigh
REAL
1
fDataSrc
REAL
9
fDataOut
REAL
0
BZAND_TAB(
BZAND_TAB=> ,
fLimitLow:=fLimitLow 9 ,
fLimitHigh:=fLimitHigh 1 ,
fDataSrc:= fDataSrc 9 ,
fDataOut=> fDataOut 0);RETURN
fLimitLow ≤ fDataSrc 时， 0 =>fDataOut。

LD语言


表达式
类型
值
准备值
地址
注释
fLimitHigh
REAL
5
正偏差
fLimitLow
REAL
-5
负偏差
fDataSrc
REAL
2
输入源
fDataOut
REAL
7
输出值
Program Segment: .....
Network Comment
ZONE_TAB
EN
ENO
ZONE_TAB
正偏差
5
fLimitHigh
fLimitHigh
输出值
7
fDataOut
fDataOut
负偏差
-5
fLimitLow
fLimitLow
输入源数据
2
fDataSrc
fDataSrc

3.11.4 ZONE_TAB 区域控制

①指令格式

指令	功能	FB/FC	LD 表现	ST表现
ZONE_TAB	区域控制	FC		ZONE_TAB(ZONE_TAB=> , fLimitHigh:= , fLimitLow:= , fDataSrc:= , fDataOut=>);

②相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
fLimitHigh	正偏差	REAL	-	0	正偏差
fLimitLow	负偏差	REAL	-	0	负偏差
fDataSrc	输入源数据	REAL	-	0	输入源数据

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
fDataOut	输出值	REAL	-	-	输出值

数据类型

	布尔	位串				整数							实数		时刻、持续时间 日期、字符串					
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
FLimitLow														√						
FLimitHigh														√						
fDataSrc														√						
fDataOut														√						

③功能说明

根据输入值是正数还是负数，用指定的偏差值来控制输出值（将偏差值加在输入值上）。 依据输入值fDataSrc的符号确定，加上fLimitHigh或fLimitLow，执行结果保存至fDataOut中。

当 fDataSrc ＜ 0 时， fDataSrc + fLimitLow => fDataOut；

当 fDataSrc ＞ 0 时， fDataSrc + fLimitHigh => fDataOut；

当 fDataSrc =0 时， 0 => fDataOut

④程序示例

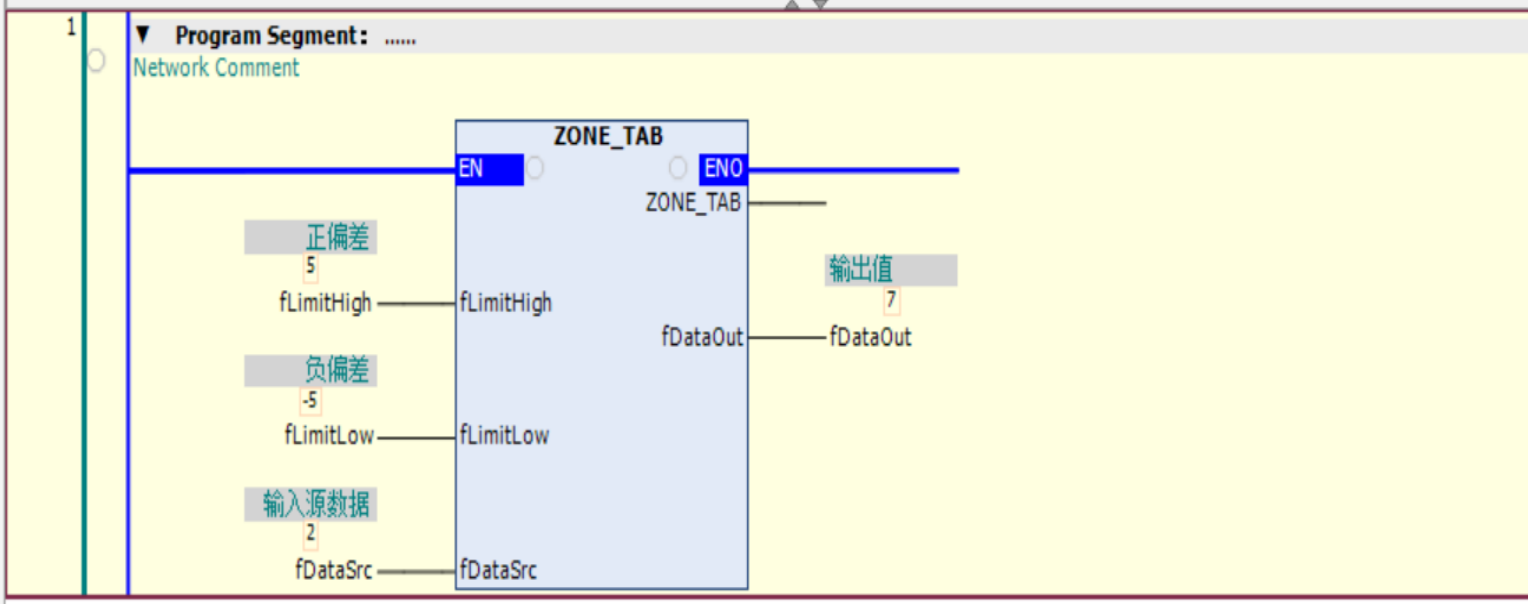
ST语言

表达式	类型	值	准备值	地址	注释
fLimitHigh	REAL	5			正偏差
fLimitLow	REAL	-5			负偏差
fDataSrc	REAL	2			输入源...
fDataOut	REAL	7			输出值

```
1  ZONE_TAB(
2      ZONE_TAB=> ,
3      fLimitHigh:= fLimitHigh 5 ,
4      fLimitLow:=fLimitLow -5 ,
5      fDataSrc:=fDataSrc 2 ,
6      fDataOut=>fDataOut 7 );RETURN
```

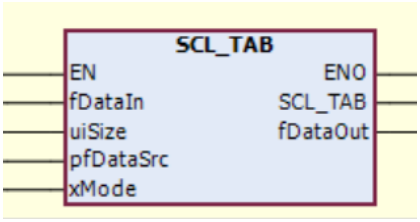
LD语言

表达式	类型	值	准备值	地址	注释
fLimitHigh	REAL	5			正偏差
fLimitLow	REAL	-5			负偏差
fDataSrc	REAL	2			输入源
fDataOut	REAL	7			输出值



3.11.5 SCL_TAB 定坐标（不同点坐标数据）

①指令格式

指令	功能	FB/FC	LD 表现	ST表现
SCL_TAB	定坐标（不同点坐标数据）	FC		SCL_TAB(SCL_TAB=> , fDataIn:= , uiSize:= , pfDataSrc:= , xMode:= , fDataOut=>);

②相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
fDataIn	输入数据	REAL	-	0	输入数据
uiSize	表格长度	UINT	-	0	表格长度
pfDataSrc	表格数据	POINTER TO REAL	-	0	表格数据
xMode	(X,Y)坐标模式， False: x1,y1;x2,y2;...,xn,yn; True: x1,x2,...,xn;y1,y2,...,yn	BOOL	[FALSE, TRUE]	FALSE	(X,Y)坐标模式 False: x1,y1;x2,y2;...,xn,yn; True: x1,x2,...,xn;y1,y2,...,yn

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
fDataOut	输出数据	REAL	-	-	输出结果

数据类型

	布尔	位串				整数							实数		时刻、持续时间 日期、字符串					
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
pfDataSrc	POINTER TO REAL																			
uiSize							√													
fDataIn														√						
xMode	√																			
fDataOut														√						

③功能说明

根据输入值是正数还是负数，用指定的偏差值来控制输出值（将偏差值加在输入值上）。

④程序示例

ST语言

Device.Application.表格坐标获取ST

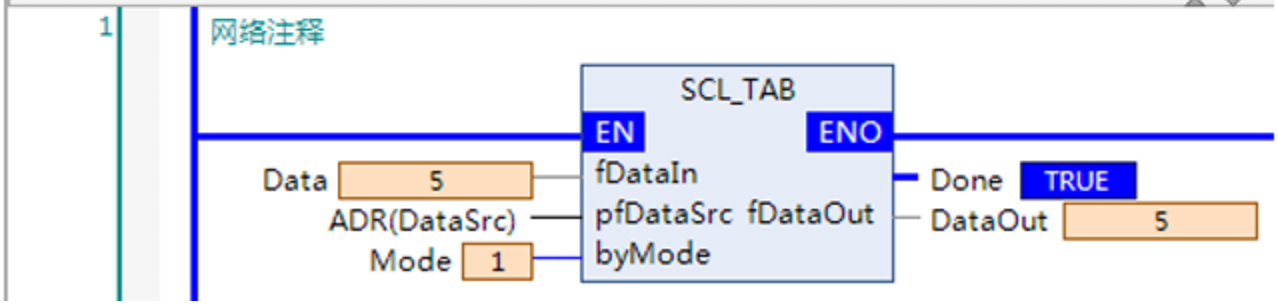
表达式	类型	值
Data	REAL	5
DataSrc	ARRAY [0..10] OF R...	
DataSrc[0]	REAL	3
DataSrc[1]	REAL	0
DataSrc[2]	REAL	0
DataSrc[3]	REAL	10
DataSrc[4]	REAL	10
DataSrc[5]	REAL	20
DataSrc[6]	REAL	0

```
1  SCL_TAB(
2  fDataIn:= Data 5 ,
3  pfDataSrc:=ADR(DataSrc) ,
4  byMode:= Mode 1 ,
5  fDataOut=>DataOu 5 );
6  RETURN
```

LD语言

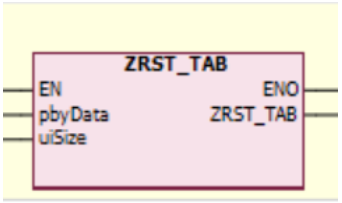
Device.Application.表格坐标获取LD

表达式	类型	值	准
Data	REAL	5	
DataSrc	ARRAY [0..10] OF R...		
DataSrc[0]	REAL	3	
DataSrc[1]	REAL	0	
DataSrc[2]	REAL	0	
DataSrc[3]	REAL	10	
DataSrc[4]	REAL	10	
DataSrc[5]	REAL	20	
DataSrc[6]	REAL	0	



3.11.6 ZRST_TAB 全部数据复位

①指令格式

指令	功能	FB/FC	LD 表现	ST表现
ZRST_TAB	全部数据复位	FC		ZRST_TAB(ZRST_TAB=> , pbyData:= , uiSize:=);

②相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
pbyData	复位数据首地址	POINTER TO BYTE	–	-	复位数据首地址
uiSize	复位数据长度，以字节为单位	UINT	1- 1024	0	数据长度

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
ZRST_TAB	输出结果	Bool	[FALSE,TRUE]	FALSE	输出结果

数据类型

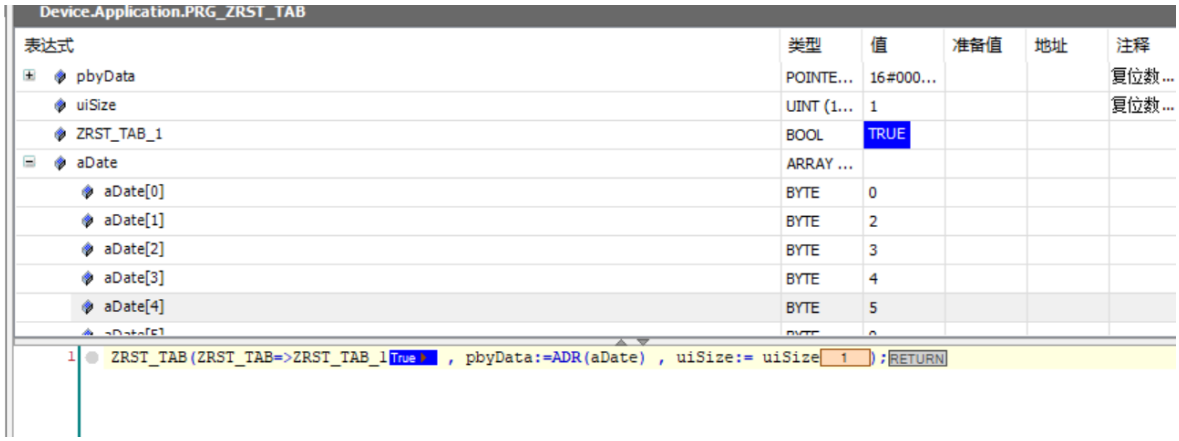
	布尔	位串				整数							实数		时刻、持续时间 日期、字符串					
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
pbfData	POINTER TO BYTE																			
uiSize							√													
ZRST_TAB	√																			

③功能说明

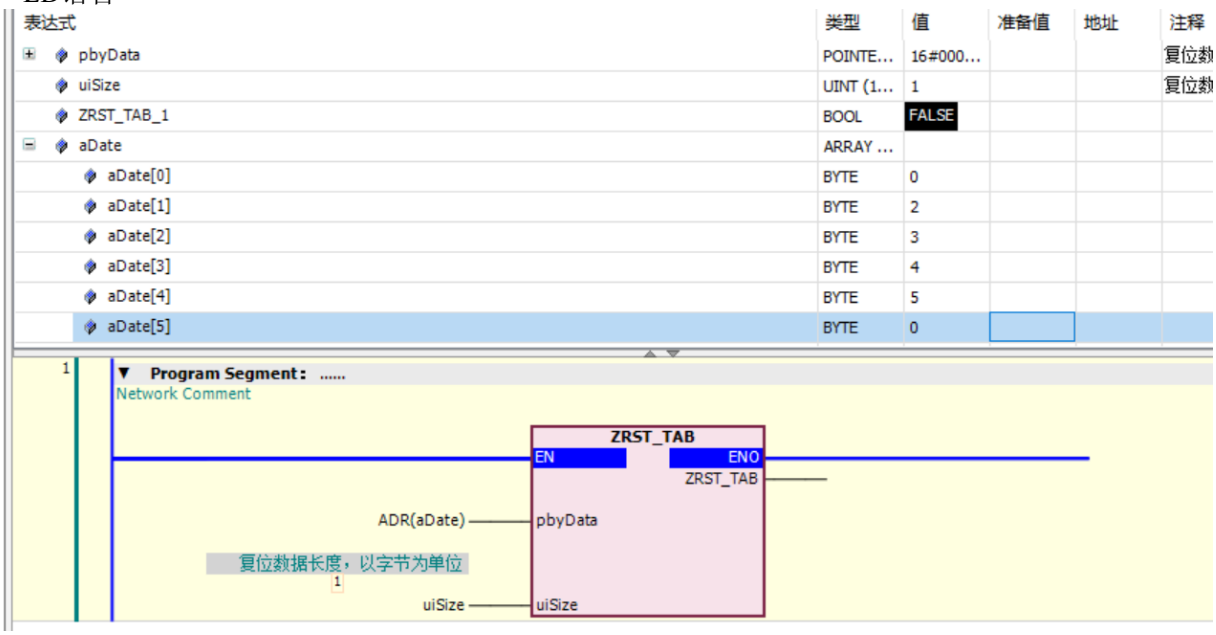
批量复位表格内的数据

④程序示例

ST语言

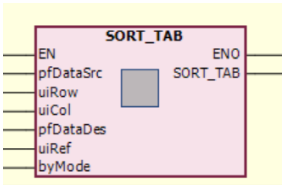


LD语言



3.11.7 SORT_TAB 数据排序

①指令格式

指令	功能	FB/FC	LD 表现	ST表现
SORT_TAB	数据排序	FC		SORT_TAB(SORT_TAB=>SORT_TAB_1 , pfDataSrc:= , uiRow:=, uiCol:= , pfDataDes:= , uiRef:= , byMode:=);

②相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
pfDataSrc	源数据	POINTER TO REAL	–	-	源数据
uiRow	行	UINT	1- 32	0	数据长度
uiCol	列	UINT	1-32	0	
pfDataDes	数据输出	POINTER TO REAL	–	-	
uiRef	依据行或者列	UINT	1-32	0	
byMode	排序方式	BYTE	1-4	0	

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
SORT_TAB_1	返回值	BOOL	[FALSE, TRUE]	FALSE	TRUE：函数执行成功 FALSE：函数执行错误

数据类型

	布尔	位串				整数							实数		时刻、持续时间 日期、字符串					
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
pfDataSrc	POINTER TO REAL																			
pfDataDes	POINTER TO REAL																			
uiRef							√													
uiRow							√													
uiCol							√													
byMode		√																		
SORT_TAB_1	√																			

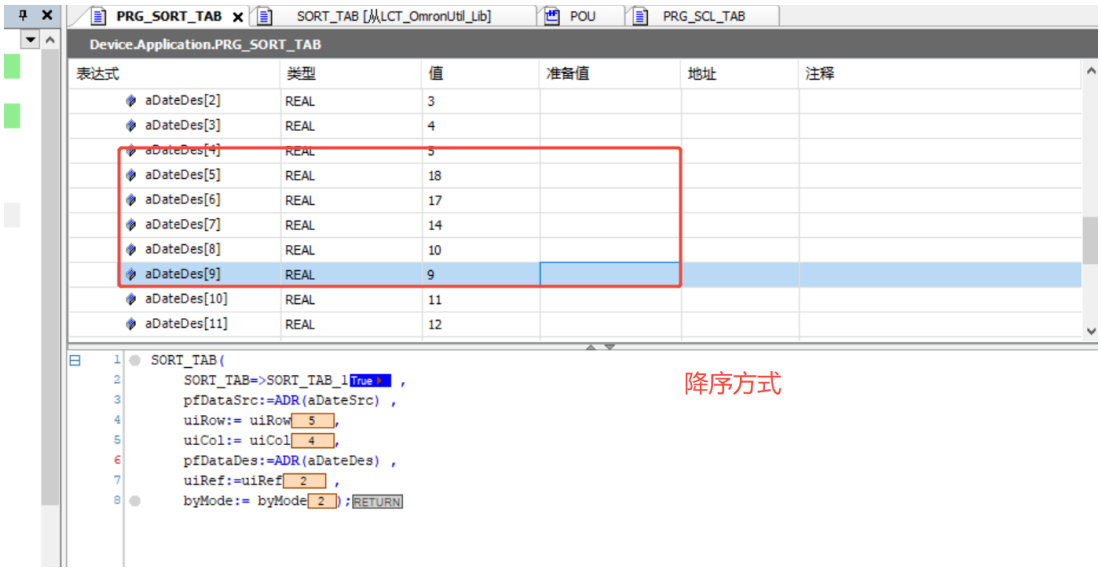
③功能说明

将 iRow* 行 *× **iCol 列的数组，以第 uiRef 列参数排序后，排序结果存放于以 pfDataDes 单元起始的变量区 域。

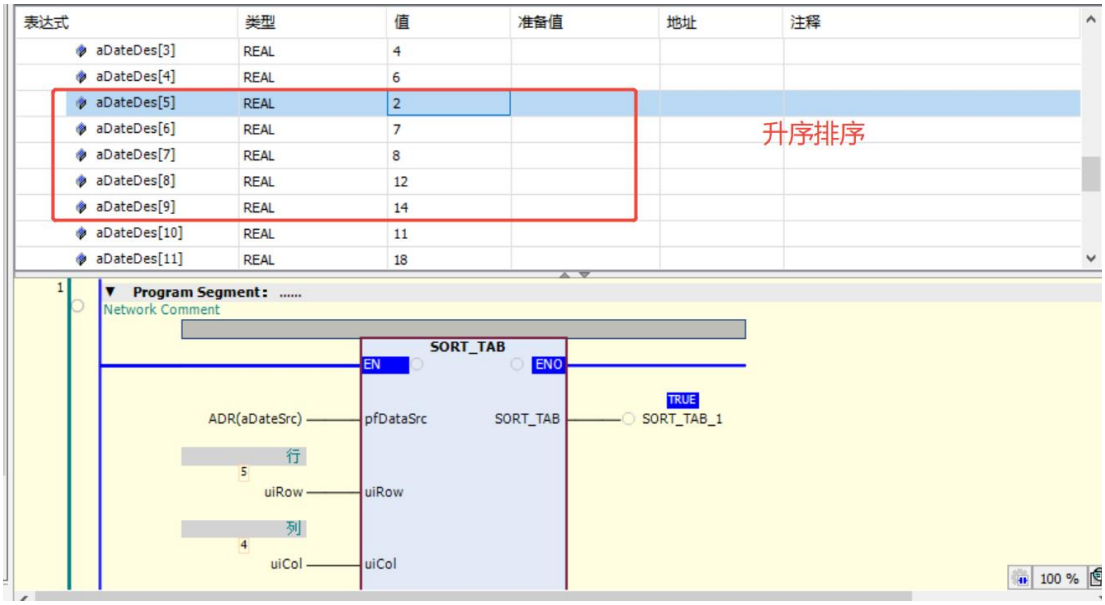
pfDataSrc 为第 1 行（或称第 1 条记录）的首个变量的起始单元； uiRow 数组的行数，或称记录数； uiCol 数组的列数，或称每条记录的栏目数； pfDataDes 排序后存放的起始单元，占用随后的变量单元数目与排序前的数组变量数目相同； uiRef 为以排序为依据行/列号。 byMode 排序模式1：列参考升序，2：列参考降序，3：行参考升序，4：行参考降序。

④程序示例

ST语言

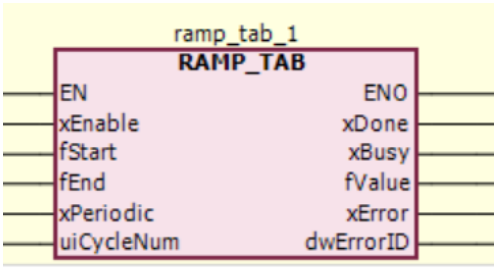


LD语言



3.11.8 RAMP_TAB 斜坡指令

①指令格式

指令	功能	FB/FC	LD 表现	ST表现
RAMP_TAB	斜坡指令	FB		RAMP_TAB_0(xEnable:= , fStart:= , fEnd:= , xPeriodic:= , uiCycleNum:= , xDone=> , xBusy=> , fValue=> , xError=> , dwErrorID=>);

②相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
xEnable	启用信号，高电平有效	BOOL	[FALSE, TRUE]	FALSE	启用信号，高电平有效
fStart	起始值	REAL	-	0	起始值
fEnd	终止值	REAL	-	0	终止值
xPeriodic	循环输出，FALSE:执行一次，TRUE:循环执行	BOOL	[FALSE, TRUE]	FALSE	循环输出，FALSE:执行一次，TRUE:循环执行
uiCycleNum	完成周期数	UINT	-	0	完成周期数

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
xDone	执行完成	BOOL	[FALSE, TRUE]	FALSE	TRUE：函数执行成功 FALSE：函数执行错误
fValue	当前值	REAL	-	0	当前值
xError	错误	BOOL	[FALSE, TRUE]	FALSE	错误
xBusy	执行中	BOOL	[FALSE, TRUE]	FALSE	执行中
dwErrorID	错误ID	DWORD	-	0	错误ID

数据类型

	布尔	位串				整数							实数		时刻、持续时间 日期、字符串					
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
xEnable	√																			
fStart														√						
fEnd														√						
xPeriodic	√																			
uiCycleNum							√													
xDone	√																			
fValue														√						
xError	√																			
xBusy	√																			
dwErrorID			√																	

③功能说明

按照指定间隔，输出值从最低值变化到最高值，或者从最高值变化到最低值；该指令的功能是在给定的两个数 据中间，在指定的时间区间，进行线性插值，按扫描执行的时间依次输出过程值，直到区间末端的终点值为止。

④程序示例

ST语言

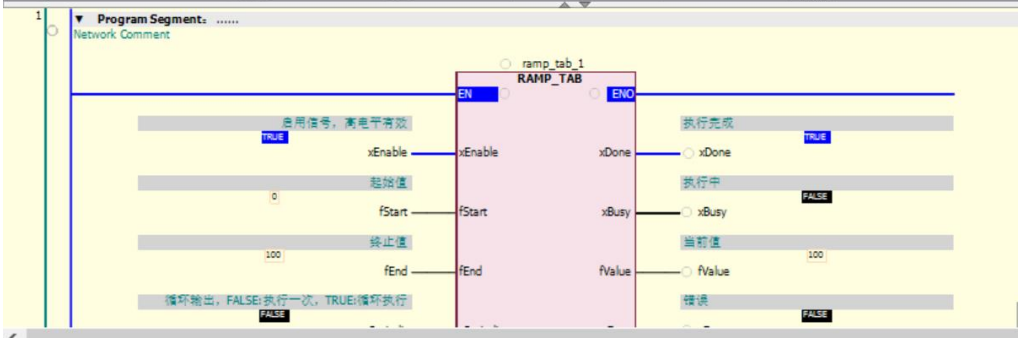
Device.Application.PRG_RAMP_TAB

表达式	类型	值	准备值	地址	注释
RAMP_TAB_0	RAMP_TAB				
xEnable	BOOL	TRUE			启用信号，高电平有效
fStart	REAL	0			起始值
fEnd	REAL	100			终止值
xPeriodic	BOOL	FALSE			循环输出，FALSE:...次，TRUE:循环
uiCycleNum	UINT (UINT#1..65535)	1			完成周期数
xDone	BOOL	TRUE			执行完成
xBusy	BOOL	FALSE			执行中
fValue	REAL	100			当前值
xError	BOOL	FALSE			错误
dwErrorID	DWORD	0			错误ID

```
1  RAMP_TAB_0(
2      xEnable:= xEnable,
3      fStart:=fStart ,
4      fEnd:= fEnd,
5      xPeriodic:= xPeriodic,
6      uiCycleNum:=uiCycleNum ,
7      xDone=>xDone ,
8      xBusy=>xBusy ,
9      fValue=> fValue,
10     xError=> xError,
11     dwErrorID=>dwErrorID );RETURN
```

LD语言

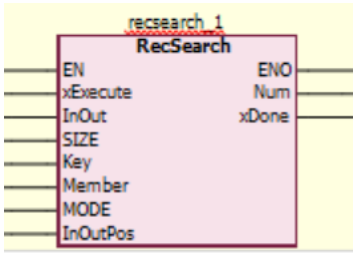
ramp_tab_1	RAMP_TAB				
xEnable	BOOL	TRUE			启用信号，高电平有效
fStart	REAL	0			起始值
fEnd	REAL	100			终止值
xPeriodic	BOOL	FALSE			循环输出，FALSE:...次，TRUE:循
uiCycleNum	UINT (UINT#1..65535)	1			完成周期数
xDone	BOOL	TRUE			执行完成
xBusy	BOOL	FALSE			执行中
fValue	REAL	100			当前值



3.11.9 RecSearch 记录检索

①指令格式

指令	功能	FB/FC	LD 表现	ST表现

RecSearch	记录检索	FB		RecSearch_0(xExecute:= , InOut:= , SIZE:= , Key:= , Member:= , MODE:= , Num=> , xDone=> , InOutPos:=);

②相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
xExecute	信号源	BOOL	[true,false]	-	信号源
InOut	检索对象排列	-	遵从数据类型	-	检索对象排列
SIZE	检索对象要素数	UINT	遵从数据类型	1	检索对象要素数
Key	检索关键字	-	遵从数据类型	-	检索关键字
Member	检索对象成员	-	遵从数据类型	-	检索对象成员
MODE	检索方法	eSearch_MODE	1(_LINEAR), 2(_BIN_ASC), 3(_BIN_DESC)	1(_LINEAR)	检索方法

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Num	匹配要素的个数	UINT	遵从数据类型	0	匹配要素的个数
xDone	检索结果	BOOL	[FALSE, TRUE]	FALSE	检索结果

数据类型

	布尔	位串				整数							实数		时刻、持续时间 日期、字符串					
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
xExecute	√																			
InOut	指定将结构体作为要素的排列																			
SIZE							√													
Key						√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
Member						√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
MODE	列举型_eSEARCH_MODE列举值参考功能说明																			
Num							√													
xDone	√																			

③功能说明

从以结构体为要素的排列In[]的“Size”个要素中，即In[0]到In[“Size”-1]中，检索结构体的检索对象成 员“Member”值与检索关键字“Key”匹配的要素。

将In[]中任一要素的检索对象成员作为参数传递到“Member”。

若有匹配的要素，检索结果“Out”的值为TRUE。然后，分别将匹配要素的要素编号带入InOutPos[0]中， 将匹配要素的数量带入“Num”中。匹配要素有2个以上是，将In[]中最低位的匹配要素的要素编号带入 InOutPos[0]中。

若无匹配要素，则“Out”的值为FALSE，InOutPos[0]和“Num”为0

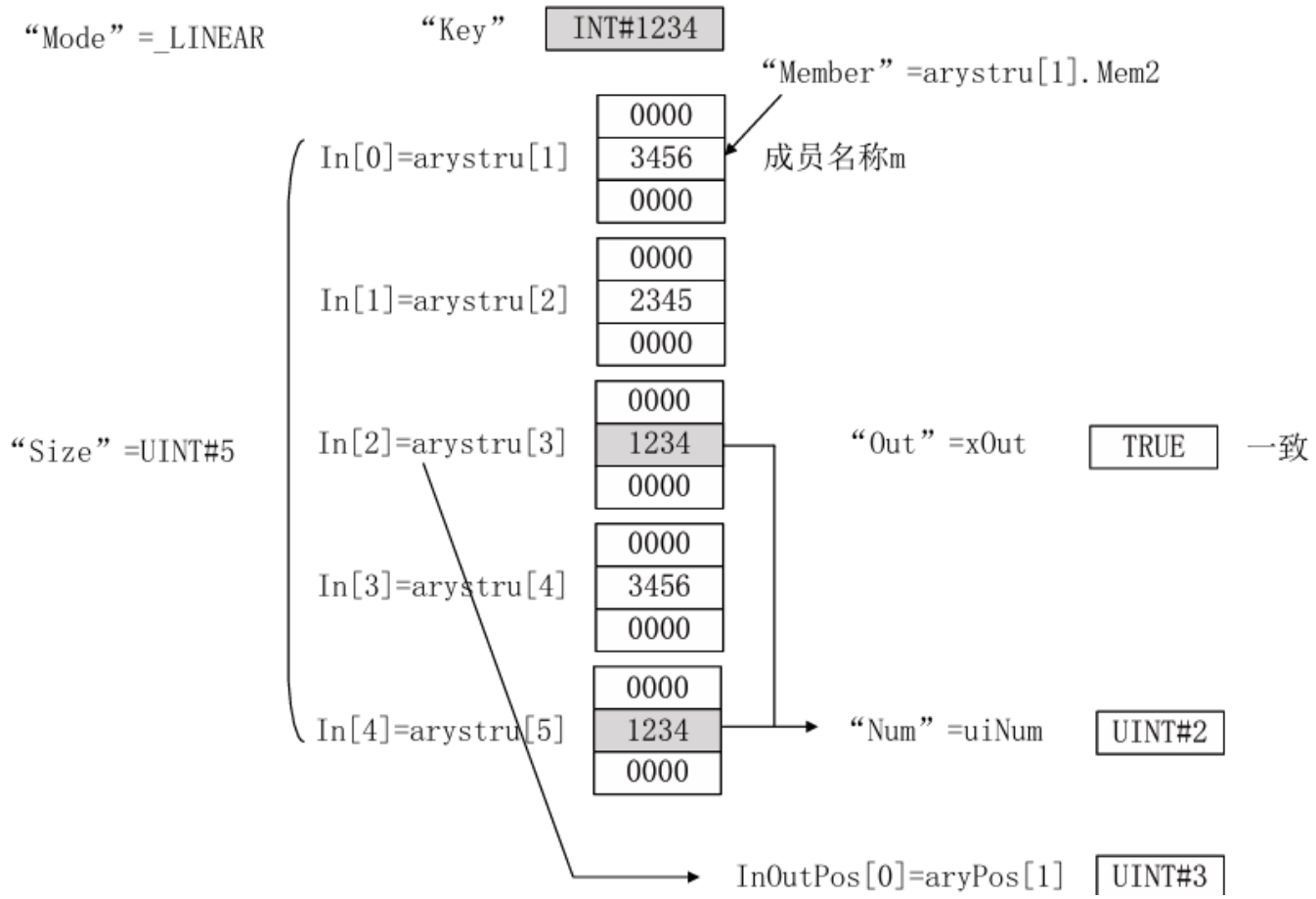
传递到In[]的输入参数应连同要素编号一起指定，如array[3]。

检索方法“Mode”的数据类型为列举型_eSEARCH_MODE。列举值的含义如下所示。

列举值含义

列举值	含义
1（_LINEAR）	线性查找
2（_BIN_ASC）	升序二分查找
3（_BIN_DESC）	降序二分查找

图示操作:



④程序示例

ST语言

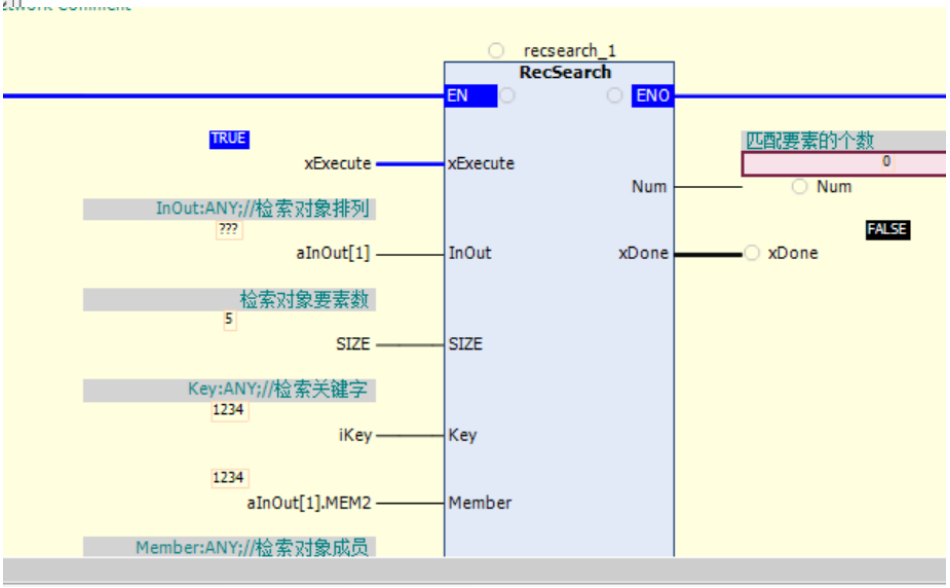
表达式	类型	值	准备值	地址	注释
RecSearch_0	RecSearch				
xExecute	BOOL	TRUE			
aInOut	ARRAY [1..5] OF DUT				InOut:ANY;//检索对象排列
aInOut[1]	DUT				
MEM1	BOOL	FALSE			
MEM2	INT	3456			
MEM4	REAL	0			
aInOut[2]	DUT				
MEM1	BOOL	FALSE			
MEM2	INT	2345			
MEM4	REAL	0			
aInOut[3]	DUT				
MEM1	BOOL	FALSE			
MEM2	INT	1234			
MEM4	REAL	0			
aInOut[4]	DUT				
MEM1	BOOL	FALSE			
MEM2	INT	1234			
MEM4	REAL	0			
aInOut[5]	DUT				
MEM1	BOOL	FALSE			
MEM2	INT	4567			
MEM4	REAL	0			
SIZE	UINT	5			检索对象要素数
iKey	INT	1234			Key:ANY;//检索关键字
MODE	ESEARCH_MODE	_LINER			Member:ANY;//检索对象成员
Num	UINT	2			匹配要素的个数
xDone	BOOL	FALSE			
aInOutPos	ARRAY [1..5] OF UINT				

```
1  RecSearch_0(
2      xExecute True := xExecute True ,
3      InOut:=aInOut[1] ,
4      SIZE 5 :=SIZE 5 ,
5      Key:= iKey 1234 ,
6      Member:= aInOut[1].MEM2 3456 ,
7      MODE _LINER :=MODE _LINER ,
8      Num 2 =>Num 2 ,
9      xDone False => xDone False ,
10     InOutPos 2 :=aInOutPos[1] 2 );RETURN
```

LD语言

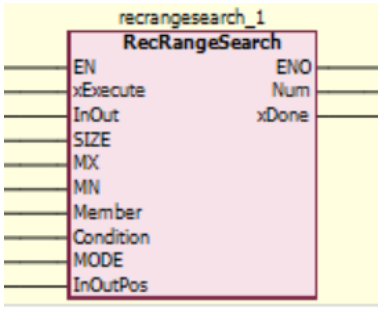
表达式	类型	值	准备值	地址	注释
RecSearch_0	RecSearch				
xExecute	BOOL	TRUE			
aInOut	ARRAY [1..5] OF DUT				InOut:ANY;//检索对象排列
aInOut[1]	DUT				
MEM1	BOOL	FALSE			
MEM2	INT	3456			
MEM4	REAL	0			
aInOut[2]	DUT				
MEM1	BOOL	FALSE			
MEM2	INT	2345			
MEM4	REAL	0			
aInOut[3]	DUT				
MEM1	BOOL	FALSE			
MEM2	INT	1234			
MEM4	REAL	0			
aInOut[4]	DUT				
MEM1	BOOL	FALSE			
MEM2	INT	1234			
MEM4	REAL	0			
aInOut[5]	DUT				
MEM1	BOOL	FALSE			

aInOut[5]	DUT				
MEM1	BOOL	FALSE			
MEM2	INT	4567			
MEM4	REAL	0			
SIZE	UINT	5			检索对象要素数
iKey	INT	1234			Key:ANY;//检索关键字
MODE	ESEARCH_MODE	_LINER			Member:ANY;//检索对象成员
Num	UINT	2			匹配要素的个数
xDone	BOOL	FALSE			
aInOutPos	ARRAY [1..5] OF UINT				



3.11.10 RecRangeSearch 范围指定记录检索

①指令格式

指令	功能	FB/FC	LD 表现	ST表现
RecRangeSearch	范围指定记录检索	FB		RecRangeSearch_0(xExecute:= , InOut:= , SIZE:= , MX:= , MN:= , Member:= , Condition:= , MODE:= , Num=> , xDone=> , InOutPos:=);

②相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
xExecute	信号源	BOOL	[true,false]	-	信号源
InOut	检索对象排列	-	遵从数据类型	-	检索对象排列
SIZE	检索对象要素数	UINT	遵从数据类型	1	检索对象要素数

Key	检索关键字	-	遵从数据类型	-	检索关键字
Member	检索对象成员	-	遵从数据类型	-	检索对象成员
MX	检索条件下限	ANY	遵从数据类型	-	检索条件下限
MN	检索条件上限	ANY	遵从数据类型	-	检索条件上限
Condition	检索条件	_eSEARCH_CONDITION	1(_EQ_BOTH), 2(_EQ_MIN), 3(_EQ_MAX), 4(_NE_BOTH)	1(_EQ_BOTH)	检索条件
MODE	检索方法	eSearch_MODE	1(_LINEAR), 2(_BIN_ASC), 3(_BIN_DESC)	1 (_ LINEAR)	检索方法

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Num	匹配要素的个数	UINT	遵从数据类型	0	匹配要素的个数
xDone	检索结果	BOOL	[FALSE,TRUE]	FALSE	检索结果

数据类型

	布尔	位串				整数							实数		时刻、持续时间 日期、字符串					
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
xExecute	√																			
InOut	指定将结构体作为要素的排列																			
SIZE							√													
Key						√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
Member						√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
MX						√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
MN						√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
Condition	列举型_eSEARCH_CONDITION 列举值参考功能说明																			
MODE	列举型_eSEARCH_MODE列举值参考功能说明																			
Num							√													
xDone	√																			

③功能说明

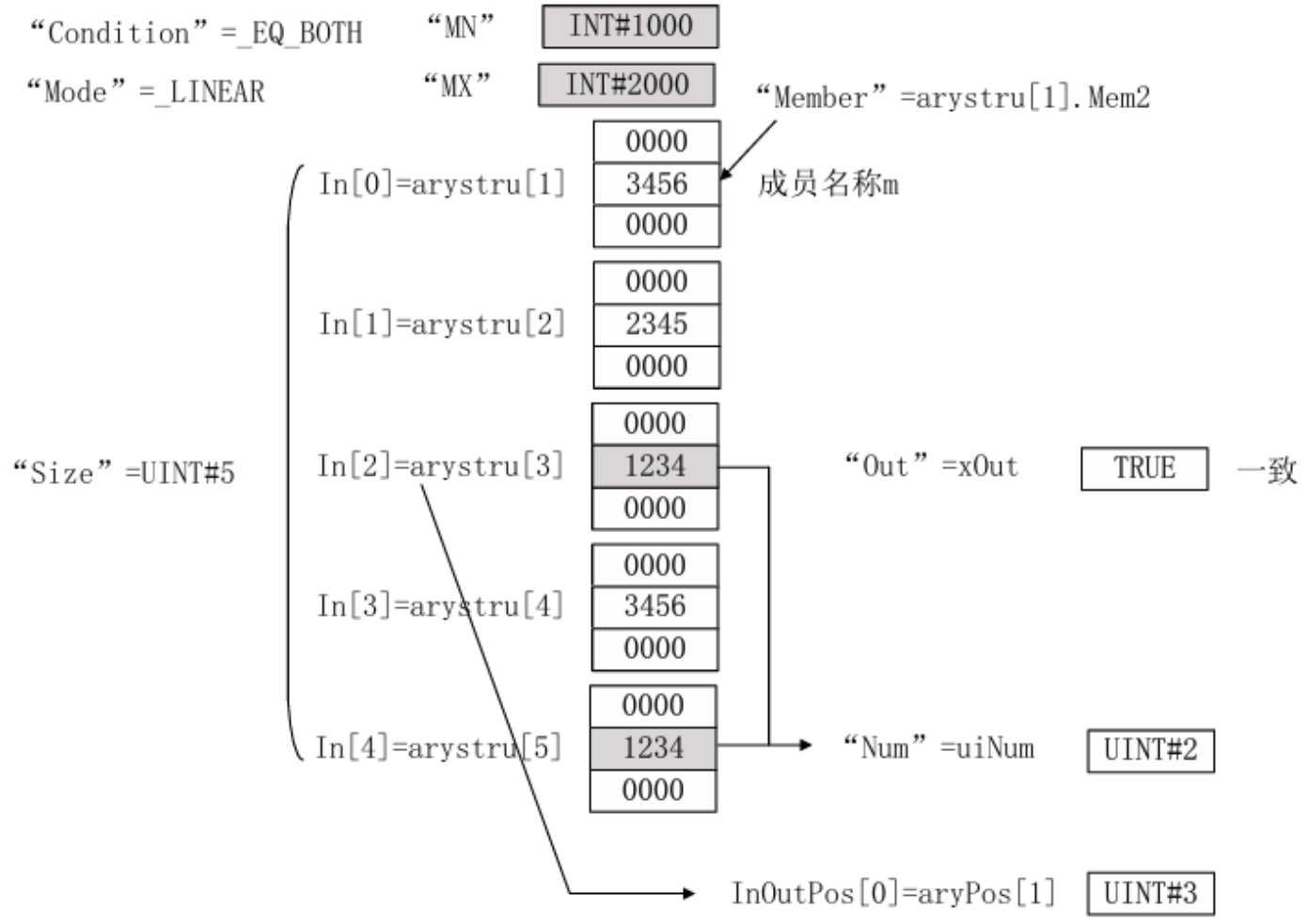
从以结构体为要素的排列In[]的“Size”个要素中，即In[0]到In[“Size”-1]中，检索结构体的检索对象成 员“Member”值与检索条件匹配的要素。 检索条件和检索方法在“Condition”和“Mode”中指定。详细信息后面再介绍 将In[]中任一要素的检索对象成员作为参数传递到“Member”。 若有与条件匹配的要素，检索结果“Out”的值为TRUE。然后，分别将匹配要素的要素编号带入InOutPos[0]中， 将匹配要素的数量带入“Num”中。匹配要素有2个以上是，将In[]中最低位的匹配要素的要素编号带入 InOutPos[0]中。 若无匹配要素，则“Out”的值为FALSE，InOutPos[0]和“Num”为0. 传递到In[]的输入参数应连同要素编号一起指定，如array[3]。 检索条件“Condition”的数据类型为列举型_eSEARCH_CONDITION。列举值的含义如下。

列举值	含义
1（EQ_BOTH）	“MN”≤“Member”≤“MX”
2（_EQ_MIN）	“MN”≤“Member”<“MX”
3（_EQ_MAX）	“MN”<“Member”≤“MX”

检索方法“Mode”的数据类型为列举型_eSEARCH_MODE。列举值的含义如下所示。

列举值	含义
1（_LINEAR）	线性查找
2（_BIN_ASC）	升序二分查找
3（_BIN_DESC）	降序二分查找

图例解析：



④程序示例

ST语言

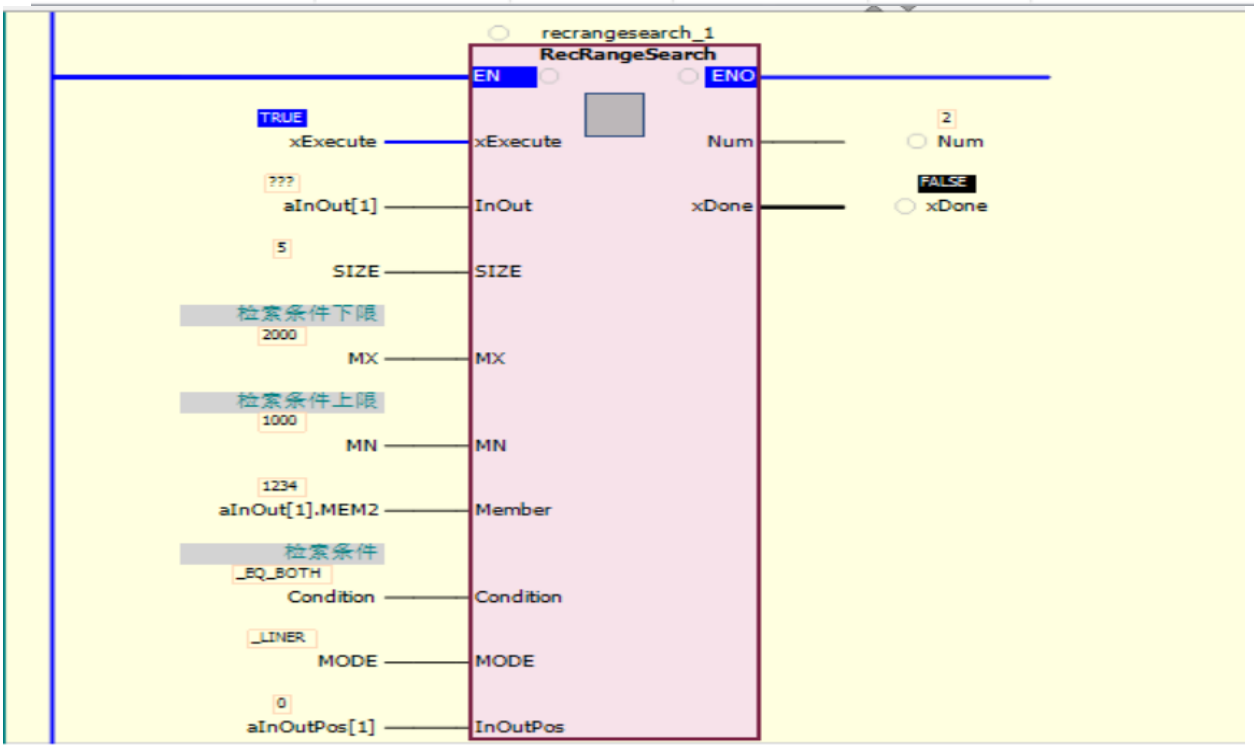
表达式	类型	值	准备值	地址	注释
RecRangeSearch_0	RecRangeSearch				
xExecute	BOOL	TRUE			
aInOut	ARRAY [1..5] OF DUT				
aInOut[1]	DUT				
MEM1	BOOL	FALSE			
MEM2	INT	1234			
MEM4	REAL	0			
aInOut[2]	DUT				
MEM1	BOOL	FALSE			
MEM2	INT	2345			
MEM4	REAL	0			
aInOut[3]	DUT				
MEM1	BOOL	FALSE			
MEM2	INT	3456			
MEM4	REAL	0			
aInOut[4]	DUT				
MEM1	BOOL	FALSE			
MEM2	INT	1234			
MEM4	REAL	0			
aInOut[5]	DUT				
MEM1	BOOL	FALSE			
MEM2	INT	5678			
MEM4	REAL	0			
SIZE	UINT	5			
MX	INT	2000			检索条件下限
MN	INT	1000			检索条件上限
Condition	ESEARCH_CONDITION	_EQ_BOTH			检索条件
MODE	ESEARCH_MODE	_LINER			
Num	UINT	2			
xDone	BOOL	FALSE			
aInOutPos	ARRAY [1..5] OF UINT				

```
1  RecRangeSearch_0(
2      xExecute:=xExecute ,
3      InOut:=aInOut[1] ,
4      SIZE:= SIZE ,
5      MX:= MX ,
6      MN:=MN ,
7      Member:= aInOut[1].MEM2 ,
8      Condition:= Condition ,
9      MODE:=MODE ,
10     Num=>Num ,
11     xDone=> xDone ,
12     InOutPos:=aInOutPos[1] );RETURN
```

LD语言

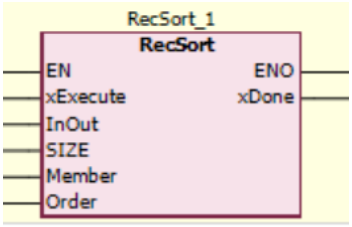
表达式	类型	值	准备值	地址	注释
RecRangeSearch_0	RecRangeSearch				
xExecute	BOOL	TRUE			
aInOut	ARRAY [1..5] OF DUT				
aInOut[1]	DUT				
MEM1	BOOL	FALSE			
MEM2	INT	1234			
MEM4	REAL	0			
aInOut[2]	DUT				
MEM1	BOOL	FALSE			
MEM2	INT	2345			
MEM4	REAL	0			
aInOut[3]	DUT				
MEM1	BOOL	FALSE			
MEM2	INT	3456			
MEM4	REAL	0			
aInOut[4]	DUT				
MEM1	BOOL	FALSE			
MEM2	INT	1234			
MEM4	REAL	0			

表达式	类型	值	准备值	地址	注释
aInOut[5]	DUT				
MEM1	BOOL	FALSE			
MEM2	INT	5678			
MEM4	REAL	0			
SIZE	UINT	5			
MX	INT	2000			检索条件下限
MN	INT	1000			检索条件上限
Condition	ESEARCH_CONDITION	_EQ_BOTH			检索条件
MODE	ESEARCH_MODE	_LINER			
Num	UINT	2			
xDone	BOOL	FALSE			
aInOutPos	ARRAY [1..5] OF UINT				



3.11.11 RecSort 记录分类

①指令格式

指令	功能	FB/FC	LD 表现	ST表现
RecSort	记录分类	FB		RecSort_0(xExecute:= , InOut:= , SIZE:= , Member:= , Order:= , xDone=>);

②相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
xExecute	信号源	BOOL	[true,false]	-	信号源
InOut	检索对象排列	-	遵从数据类型	-	检索对象排列
SIZE	检索对象要素数	UINT	遵从数据类型	1	检索对象要素数
Member	检索对象成员	-	遵从数据类型	-	检索对象成员
Order	排列顺序	eSORT_ORDER	1(_ASC), 2(_DESC)	1(_ASC	排列顺序

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
xDone	检索结果	BOOL	[FALSE,TRUE]	FALSE	检索结果

数据类型

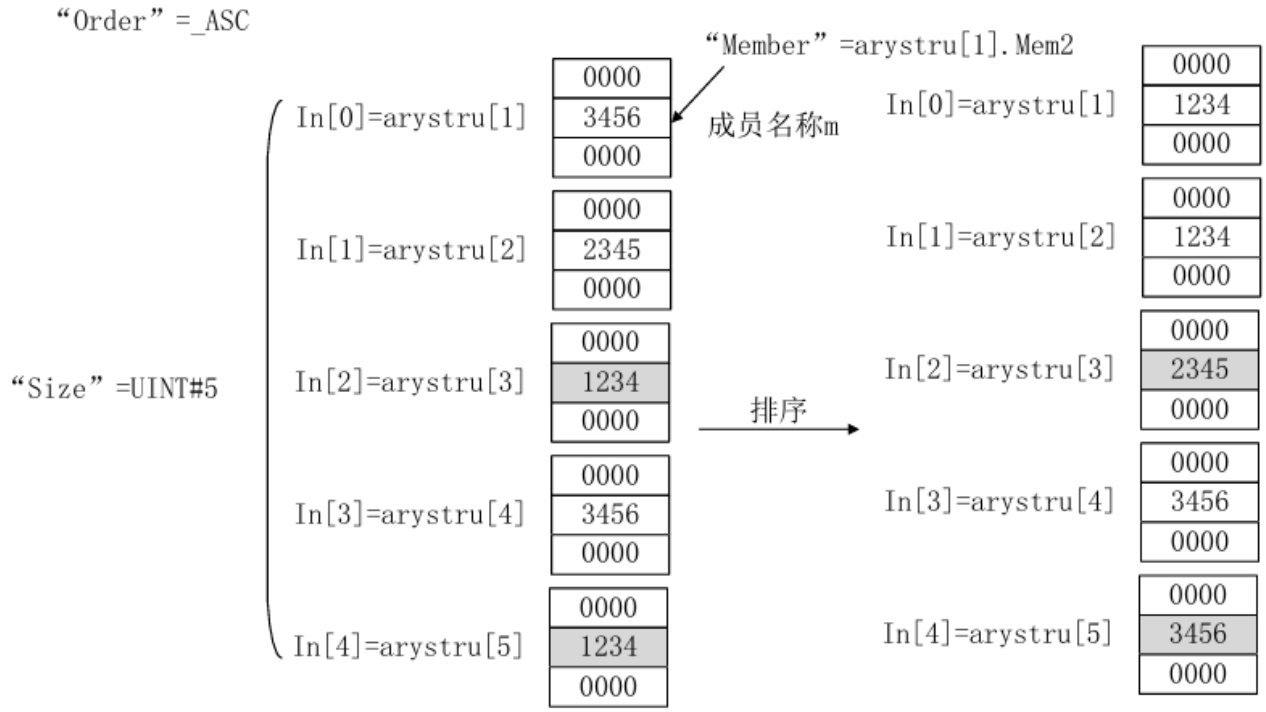
	布尔	位串				整数							实数		时刻、持续时间 日期、字符串					
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
xExecute	√																			
InOut	指定将结构体作为要素的排列																			
SIZE							√													
Member						√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
Order	eSORT_ORDER																			
xDone	√																			

③功能说明

“Execute”的值为TRUE时，对以结构体为要素的排列InOut[]的“Size”个要素，即从InOut[0]到 InOut[“Size”-1]，按结构体的排序对象成员“Member”的值进行排序。排序顺序在“Order”中指定。详 细内容后面再介绍。 将InOut[]中任一要素的排序对象成员作为参数传递到“Member”。 传递到InOut[]输入输出参数应连同要素编号一起指定，如array[3]。 排序顺序“Order”的数据为列举型_eSORT_ORDER。列举值的含义如下所示。

列举值	含义
1 (_ASC)	升序
2 (_DESC)	降序

图例说明：



④程序示例

ST语言

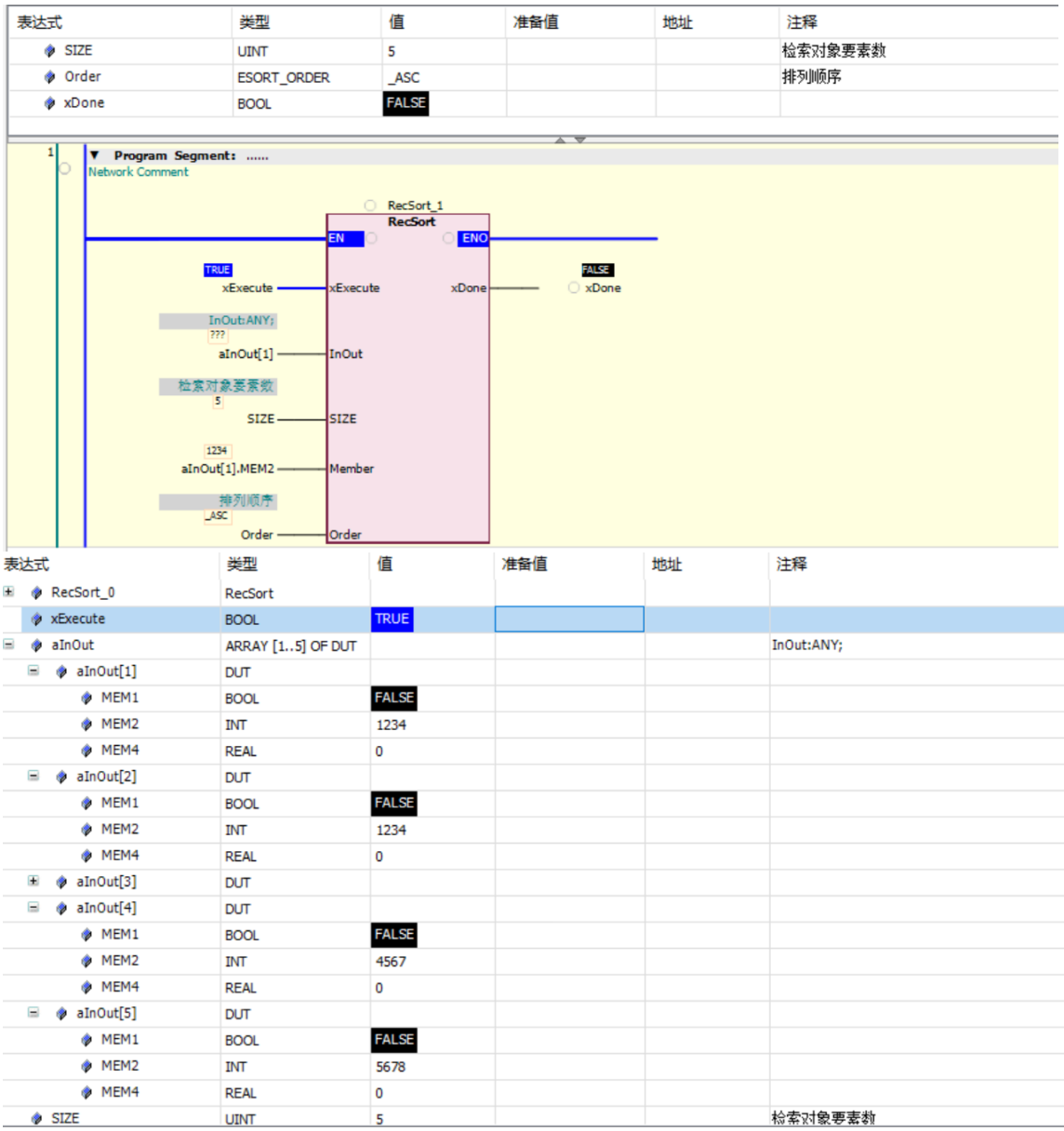
表达式	类型	值	准备值	地址	注释
RecSort_0	RecSort				
xExecute	BOOL	TRUE			
aInOut	ARRAY [1..5] OF DUT				InOut:ANY;
aInOut[1]	DUT				
MEM1	BOOL	FALSE			
MEM2	INT	1234			
MEM4	REAL	0			
aInOut[2]	DUT				
MEM1	BOOL	FALSE			
MEM2	INT	1234			
MEM4	REAL	0			
aInOut[3]	DUT				
aInOut[4]	DUT				
MEM1	BOOL	FALSE			
MEM2	INT	4567			
MEM4	REAL	0			
aInOut[5]	DUT				
MEM1	BOOL	FALSE			
MEM2	INT	5678			
MEM4	REAL	0			
SIZE	UINT	5			检索对象要素数

Device.Application.PRG_RecSort

表达式	类型	值	准备值	地址	注释
MEM4	REAL	0			
SIZE	UINT	5			检索对象要素数
Order	ESORT_ORDER	_ASC			排列顺序
xDone	BOOL	FALSE			

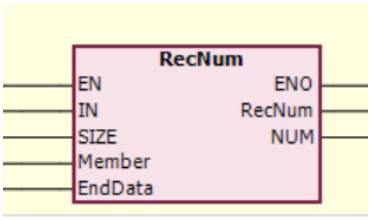
```
RecSort_0(
    xExecute:=xExecute,
    InOut:=aInOut[1] ,
    SIZE:= SIZE,
    Member:=aInOut[1].MEM2 ,
    Order:=Order ,
    xDone=>xDone);RETURN
```

LD语言



3.11.12 RecNum 记录数获取

①指令格式

指令	功能	FB/FC	LD 表现	ST表现
RecNum	记录数获取	FC		RecNum(RecNum=> , IN:= , SIZE:= , Member:= , EndData:= , NUM=>);

②相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
In	检索对象排列	-	遵从数据类型	-	检索对象排列
SIZE	检索对象要素数	UINT	遵从数据类型	1	检索对象要素数
Member	检索对象成员	-	遵从数据类型	-	检索对象成员
EndData	结束数据		遵从数据类型		结束数据

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
NUM	记录数	UINT	遵从数据类型	–	记录数

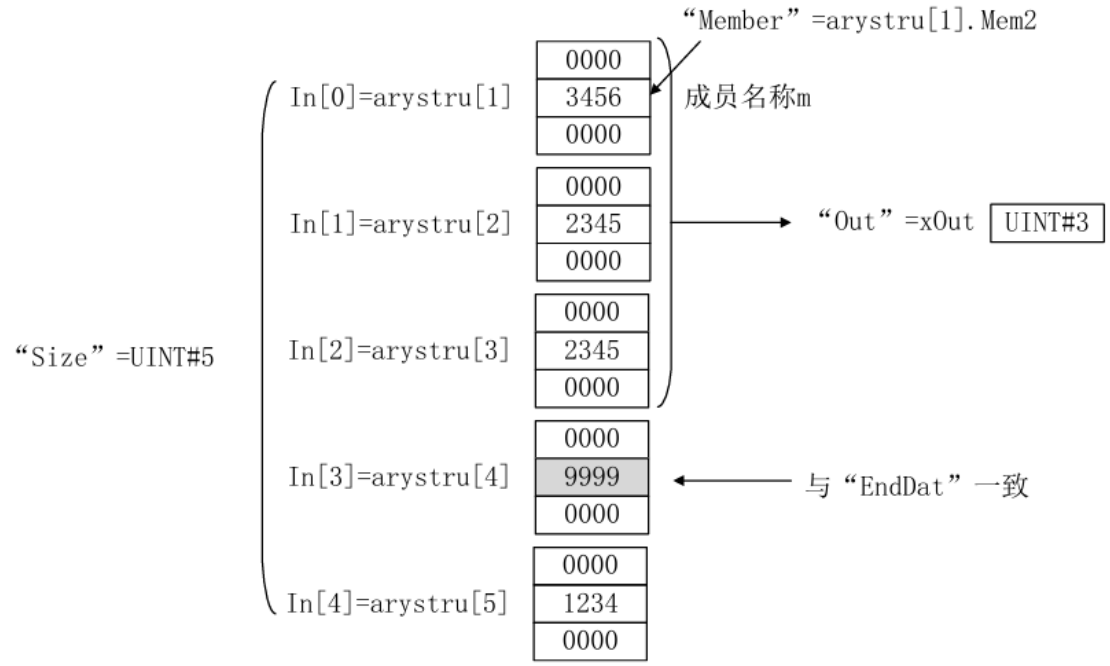
数据类型

	布尔	位串				整数							实数		时刻、持续时间 日期、字符串					
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In	指定将结构体作为要素的排列																			
SIZE							√													
Member						√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
EndData						√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
NUM							√													

③功能说明

从以结构体为要素的排列In[]的“Size”个元素即In[0]~In[“Size”-1]中，对对象成员“Member”的值与 结束数据“EndDat”匹配的要素进行检索。然后，将与“EndDat”匹配的要素数（记录数）带入“Out”中。 将In[]中任一要素的对象成员作为参数传递到“Member”。 传递到In[]的输入参数应连同要素编号一起指定，如array[3]。

图例说明：



④程序示例

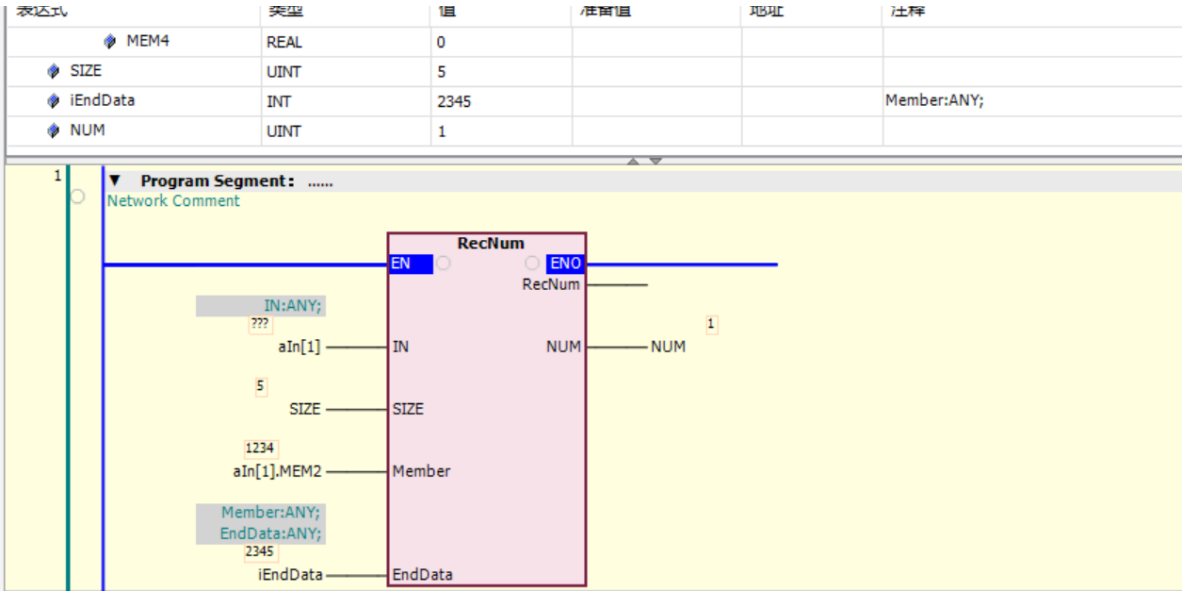
ST语言

表达式	类型	值	准备值	地址	注释
aIn	ARRAY [1..5] OF DUT				IN:ANY;
aIn[1]	DUT				
MEM1	BOOL	FALSE			
MEM2	INT	1234			
MEM4	REAL	0			
aIn[2]	DUT				
MEM1	BOOL	FALSE			
MEM2	INT	2345			
MEM4	REAL	0			
aIn[3]	DUT				
MEM1	BOOL	FALSE			
MEM2	INT	3456			
MEM4	REAL	0			
aIn[4]	DUT				
MEM1	BOOL	FALSE			
MEM2	INT	1234			
MEM4	REAL	0			
aIn[5]	DUT				
MEM1	BOOL	FALSE			
MEM2	INT	4567			
MEM4	REAL	0			
SIZE	UINT	5			
iEndData	INT	2345			Member:ANY;
NUM	UINT	1			

```
1  RecNum(
2      RecNum=> ,
3      IN:= aIn[1],
4      SIZE:= SIZE 5 ,
5      Member:=aIn[1].MEM2 1234 ,
6      EndData:=iEndData 2345 ,
7      NUM=>NUM 1 );RETURN
```

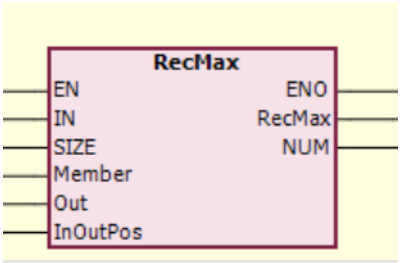
LD语言

表达式	类型	值	准备值	地址	注释
aIn	ARRAY [1..5] OF DUT				IN:ANY;
aIn[1]	DUT				
MEM1	BOOL	FALSE			
MEM2	INT	1234			
MEM4	REAL	0			
aIn[2]	DUT				
MEM1	BOOL	FALSE			
MEM2	INT	2345			
MEM4	REAL	0			
aIn[3]	DUT				
MEM1	BOOL	FALSE			
MEM2	INT	3456			
MEM4	REAL	0			
aIn[4]	DUT				
MEM1	BOOL	FALSE			
MEM2	INT	1234			



3.11.13 RecMax 记录最大值检索

①指令格式

指令	功能	FB/FC	LD 表现	ST表现
RecMax	记录最大值检索	FC		RecMax(RecMax=> , IN:= , SIZE:= , Member:= , Out:= , InOutPos:= , NUM=>);

②相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
In	检索对象排列	-	遵从数据类型	-	检索对象排列
SIZE	检索对象要素数	UINT	遵从数据类型	1	检索对象要素数
Member	检索对象成员	-	遵从数据类型	-	检索对象成员
Out	检索结果	-	遵从数据类型	-	检索结果

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Num	检索个数	UINT	遵从数据类型	-	检索个数
InOutPos[]	检索结果的要素编 号	UINT	遵从数据类型	-	检索结果

数据类型

	布尔	位串				整数							实数		时刻、持续时间 日期、字符串					
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In	指定将结构体作为要素的排列																			
SIZE							√													
Member						√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
Num							√													
Out						√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
InOutPos[]							√													

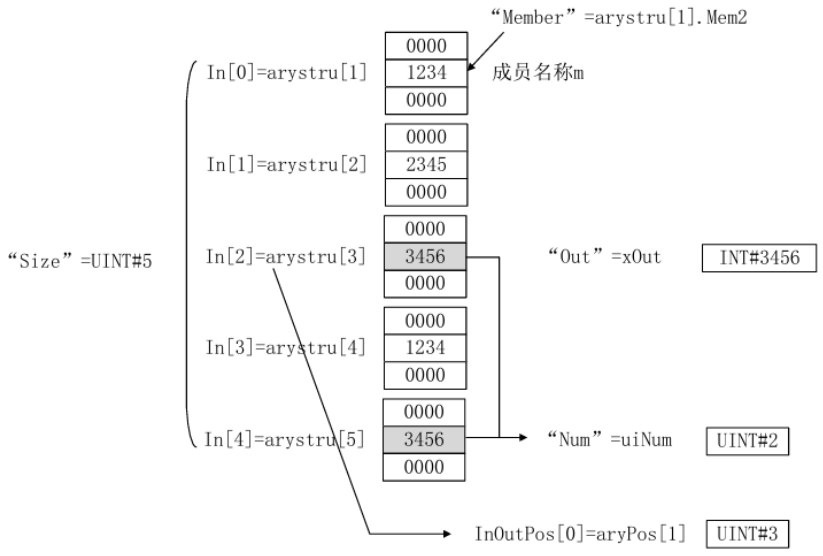
③功能说明

“从以结构体为要素的排列In[]的“Size”个要素中，即从In[0]到In[“Size”-1]中，检索结构体的检索对象 成员“Member”的值。 将In[]中任一要素的检索对象成员作为参数传递到“Member”。 分别将检索结果的要素编号带入InOutPos[0]中，检索要素的数量带入“Num”中。检索结果有2个以上时， 将In[]中最低位的检索结果的要素编号带入InOutPos[0]中。 传递到In[]的输入参数连同要素编号一起指定，如array[3]。

整数、实数类型以外值的大小关系判断如下。

数据类型	大小关系
TIME	值较大者判断为大
DATA、TOD、DT	日期或时刻较后面者判断为大
STRING	大小比较时使用字符编码。比较步骤如下所示： 首先，比较各字符串的第 1 个字符的字符编码。若字符编码不同，则字符编码的大小就是字符串的大小。 若第 1 个字符的字符编码相同，则继续比较第 2 个字符、第 3 个字符，直至发现不同的字符编码。 若各字符串的长度不同，将根据较长的字符串，在较短的字符串末尾补上 NULL 字符 (16#00)，再进行比较。

图例解析：



RecMax 检索最大值。检索结果“Out”中将带入检索对象成员的最大值。

④程序示例

ST语言

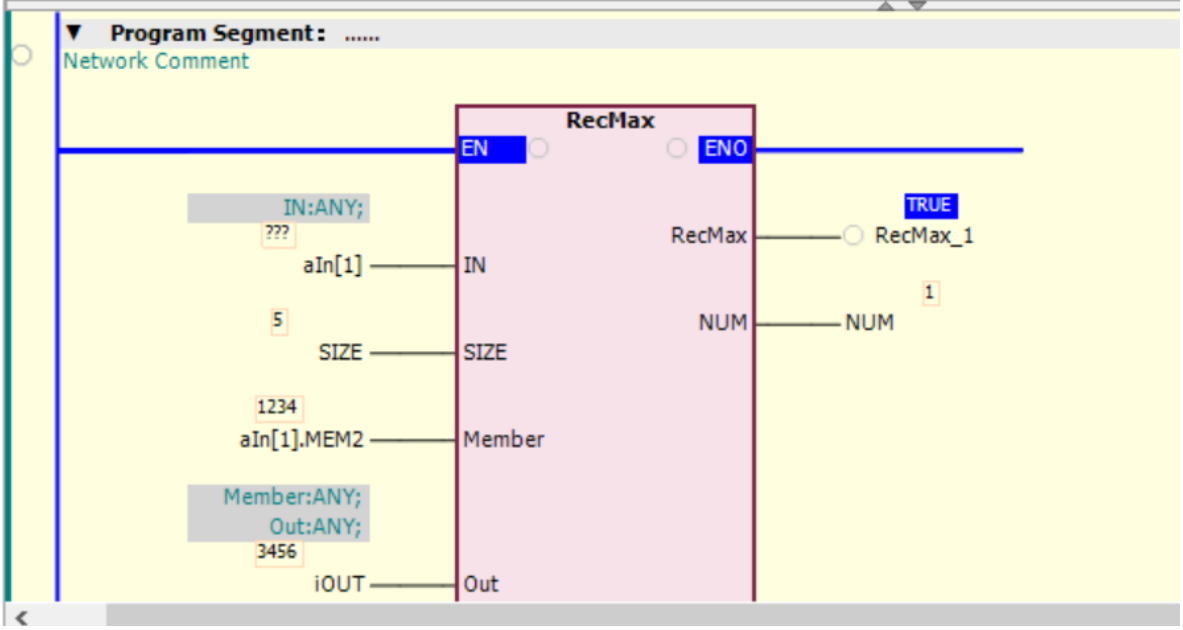
表达式	类型	值	准备值	地址	注释
aIn	ARRAY [1..5] OF DUT				IN:ANY;
aIn[1]	DUT				
MEM1	BOOL	FALSE			
MEM2	INT	1234			
MEM4	REAL	0			
aIn[2]	DUT				
MEM1	BOOL	FALSE			
MEM2	INT	2345			
MEM4	REAL	0			
aIn[3]	DUT				
MEM1	BOOL	FALSE			
MEM2	INT	3456			
MEM4	REAL	0			
aIn[4]	DUT				
MEM1	BOOL	FALSE			
MEM2	INT	1234			
MEM4	REAL	0			
aIn[5]	DUT				
MEM1	BOOL	FALSE			
MEM2	INT	3456			
MEM4	REAL	0			
SIZE	UINT	5			
iOUT	INT	3456			Member:ANY;
InOutPos	UINT	0			
aryPos	ARRAY [1..5] OF UINT				
aryPos[1]	UINT	2			
aryPos[2]	UINT	0			
aryPos[3]	UINT	0			
aryPos[4]	UINT	0			
aryPos[5]	UINT	0			
RecMax_1	BOOL	TRUE			
NUM	UINT	2			

```
1  RecMax(
2      RecMax=>RecMax_1 True ,
3      IN:=aIn[1] ,
4      SIZE:=SIZE 5 ,
5      Member:=aIn[1].MEM2 1234 ,
6      Out:=iOUT 3456 ,
7      InOutPos:=aryPos[1] 2 ,
8      NUM=>NUM 2 );RETURN
```

LD语言

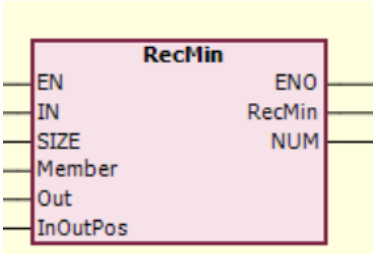
表达式	类型	值	准备值	地址	注释
aIn	ARRAY [1..5] OF DUT				IN:ANY;
aIn[1]	DUT				
MEM1	BOOL	FALSE			
MEM2	INT	1234			
MEM4	REAL	0			
aIn[2]	DUT				
MEM1	BOOL	FALSE			
MEM2	INT	2345			
MEM4	REAL	0			
aIn[3]	DUT				
MEM1	BOOL	FALSE			
MEM2	INT	3456			
MEM4	REAL	0			
aIn[4]	DUT				
MEM1	BOOL	FALSE			
MEM2	INT	1234			
MEM4	REAL	0			
aIn[5]	DUT				
MEM1	BOOL	FALSE			
MEM2	INT	3456			
MEM4	REAL	0			

表达式	类型	值	准备值	地址
aryPos[3]	UINT	0		
aryPos[4]	UINT	0		
aryPos[5]	UINT	0		
c	BOOL	FALSE		
RecMax_1	BOOL	TRUE		
NUM	UINT	1		



3.11.14 RecMin 记录最小值检索

①指令格式

指令	功能	FB/FC	LD 表现	ST表现
RecMin	记录最小值检索	FC		RecMin(RecMin=> , IN:= , SIZE:= , Member:= , Out:= , InOutPos:= , NUM=>);

②相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
In	检索对象排列	-	遵从数据类型	-	检索对象排列
SIZE	检索对象要素数	UINT	遵从数据类型	1	检索对象要素数
Member	检索对象成员	-	遵从数据类型	-	检索对象成员
Out	检索结果	-	遵从数据类型	-	检索结果

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Num	检索个数	UINT	遵从数据类型	–	检索个数
InOutPos[]	检索结果的要素编 号	UINT	遵从数据类型	–	检索结果

数据类型

	布尔	位串				整数							实数		时刻、持续时间 日期、字符串					
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In	指定将结构体作为要素的排列																			
SIZE							√													
Member						√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
Num							√													
Out						√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
InOutPos[]							√													

③功能说明

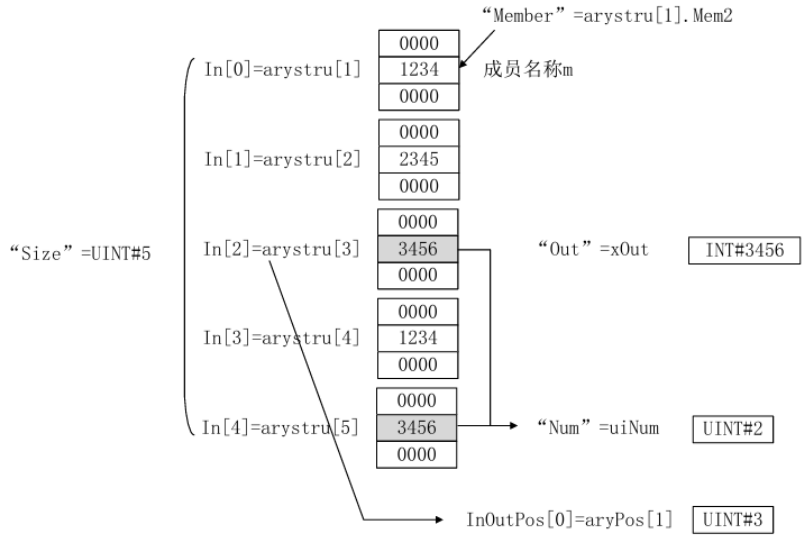
“从以结构体为要素的排列In[]的“Size”个要素中，即从In[0]到In[“Size”-1]中，检索结构体的检索对象 成员

“Member”的值。 将In[]中任一要素的检索对象成员作为参数传递到“Member”。 分别将检索结果的要素编号带入InOutPos[0]中，检索要素的数量带入“Num”中。检索结果有2个以上时， 将In[]中最低位的检索结果的要素编号带入InOutPos[0]中。 传递到In[]的输入参数连同要素编号一起指定，如array[3]。

整数、实数类型以外值的大小关系判断如下。

数据类型	大小关系
TIME	值较大者判断为大
DATA、TOD、DT	日期或时刻较后面者判断为大
STRING	大小比较时使用字符编码。比较步骤如下所示： 首先，比较各字符串的第 1 个字符的字符编码。若字符编码不同，则字符编码的大小就是字符串的大小。 若第 1 个字符的字符编码相同，则继续比较第 2 个字符、第 3 个字符，直至发现不同的字符编码。 若各字符串的长度不同，将根据较长的字符串，在较短的字符串末尾补上 NULL 字符 (16#00)，再进行比较。

图例解析：



RecMin 检索最小值。检索结果“Out”中将带入检索对象成员的最小值。

④程序示例

ST语言

表达式	类型	值	准备值	地址	注释
RecMin_1	BOOL	TRUE			in:any;
aIn	ARRAY [1..5] OF DUT				
aIn[1]	DUT				
MEM1	BOOL	FALSE			
MEM2	INT	1234			
MEM4	REAL	0			
aIn[2]	DUT				
MEM1	BOOL	FALSE			
MEM2	INT	2345			
MEM4	REAL	0			
aIn[3]	DUT				
MEM1	BOOL	FALSE			
MEM2	INT	3456			
MEM4	REAL	0			
aIn[4]	DUT				
MEM1	BOOL	FALSE			
MEM2	INT	1234			
MEM4	REAL	0			
aIn[5]	DUT				

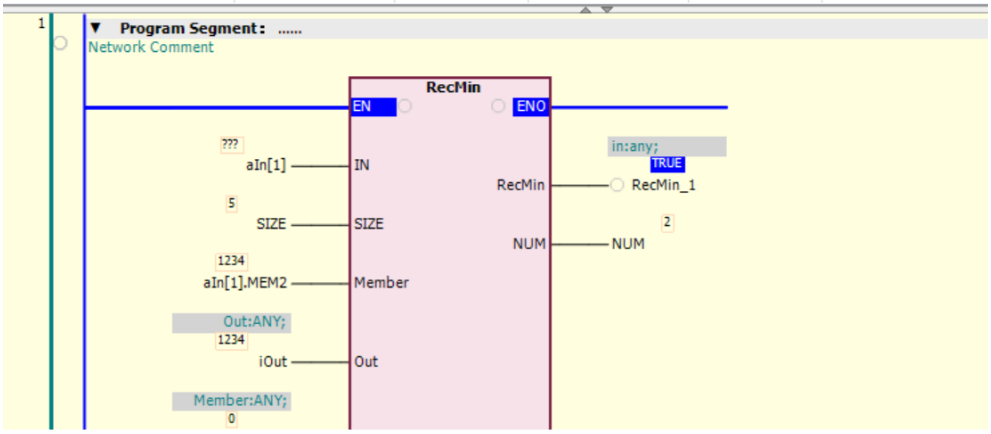
表达式	类型	值	准备值	地址	注释
MEM4	REAL	0			
aIn[5]	DUT				
MEM1	BOOL	FALSE			
MEM2	INT	3456			
MEM4	REAL	0			
SIZE	UINT	5			
iOut	INT	1234			Out:ANY;
InOutPos	UINT	0			
NUM	UINT	2			
aryPos	ARRAY [1..5] OF UINT				Member:ANY;

```
1  RecMin(
2      RecMin=>RecMin_1 True ,
3      IN:=aIn[1] ,
4      SIZE:=SIZE 5 ,
5      Member:=aIn[1].MEM2 1234 ,
6      Out:=iOut 1234 ,
7      InOutPos:=aryPos[1] 0 ,
8      NUM=> NUM 2 );RETURN
```

LD语言

表达式	类型	值	准备值	地址	注释
RecMin_1	BOOL	TRUE			in:any;
aIn	ARRAY [1..5] OF DUT				
aIn[1]	DUT				
MEM1	BOOL	FALSE			
MEM2	INT	1234			
MEM4	REAL	0			
aIn[2]	DUT				
MEM1	BOOL	FALSE			
MEM2	INT	2345			
MEM4	REAL	0			
aIn[3]	DUT				
MEM1	BOOL	FALSE			
MEM2	INT	3456			
MEM4	REAL	0			
aIn[4]	DUT				
MEM1	BOOL	FALSE			
MEM2	INT	1234			
MEM4	REAL	0			
aIn[5]	DUT				

表达式	类型	值	准备值	地址	注释
MEM4	REAL	0			
SIZE	UINT	5			
iOut	INT	1234			Out:ANY;
InOutPos	UINT	0			
NUM	UINT	2			
aryPos	ARRAY [1..5] OF UINT				Member:ANY;



3.12 保持数据指令

指令列表

指令类别	名称	FB/FC	功能
保持数据处理	ManualLoadPerVars	FB	手动加载持久化变量
	ManualSavePerVars	FB	手动保存持久化变量

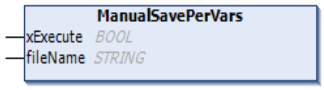
3.12.1 ManualLoadPerVars　手动加载持久化变量

①指令格式

指令	功能	FB/FC	LD 表现	ST表现
ManualLoadPerVars	手动加载持久化变量	FB		ManualLoadPerVars_0(xExecute:=IN3 , fileName:=IN4);

3.12.2 ManualSavePerVars　手动保存持久化变量

①指令格式

指令	功能	FB/FC	LD 表现	ST表现
ManualSavePerVars	手动保存持久化变量	FB		ManualSavePerVars_0(xExecute:=IN1 , fileName:=IN2);

②相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
xExecute	触发引脚	BOOL	-	0	加载触发
fileName	名称	STRING	-	‘’	加载文件名称

输入变量	名称	数据类型	有效范围	初始值	描述
xExecute	触发引脚	BOOL	-	0	保存触发
fileName	名称	STRING	-	‘’	保存文件名称

数据类型	布尔	位串				整数								实数		时刻、持续时间 日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATA	TOD	DT	STRING
xExecute	√																			
fileName																				√

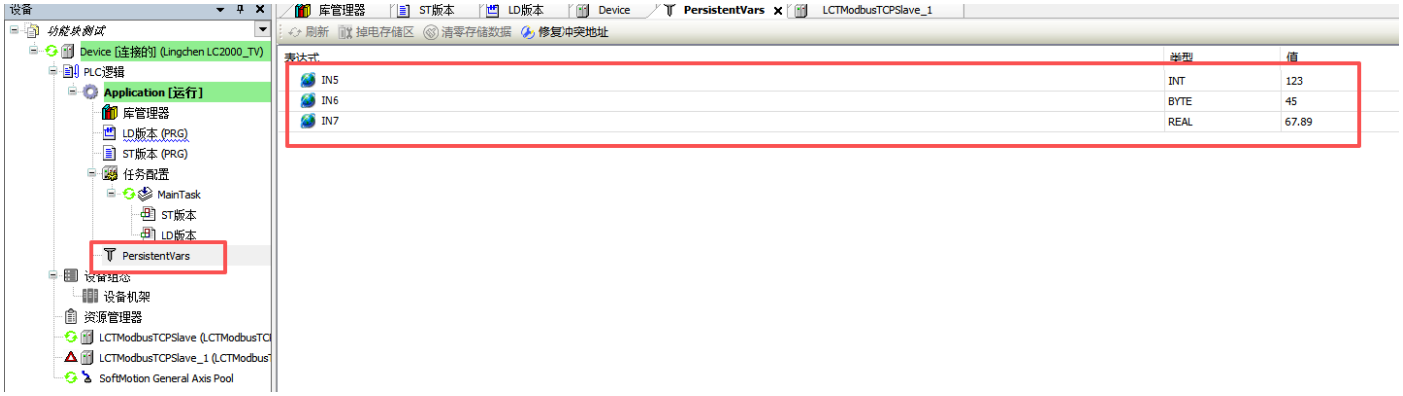
数据类型	布尔	位串				整数								实数		时刻、持续时间 日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATA	TOD	DT	STRING
xExecute	√																			
fileName																				√

③程序示例

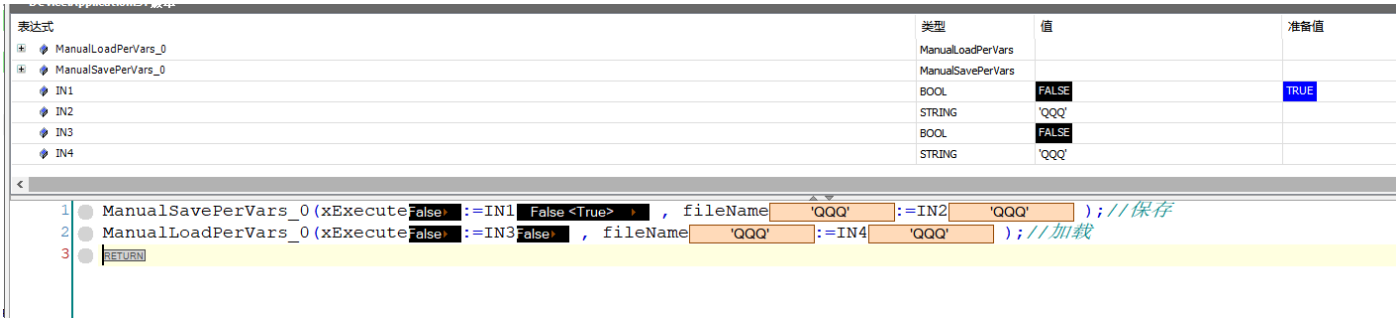
1.通过输入ManualSavePerVars中的fileName想要保存的文件名称触发xExecute引脚将整个保持变量区域保存起来。

2.输入想要查询已经经过ManualSavePerVars处理保存过的MAXLimit文件名字，断电后触发ManualLoadPerVars中的MinLimit引脚将保存过的数据载入到对应的变量中。、

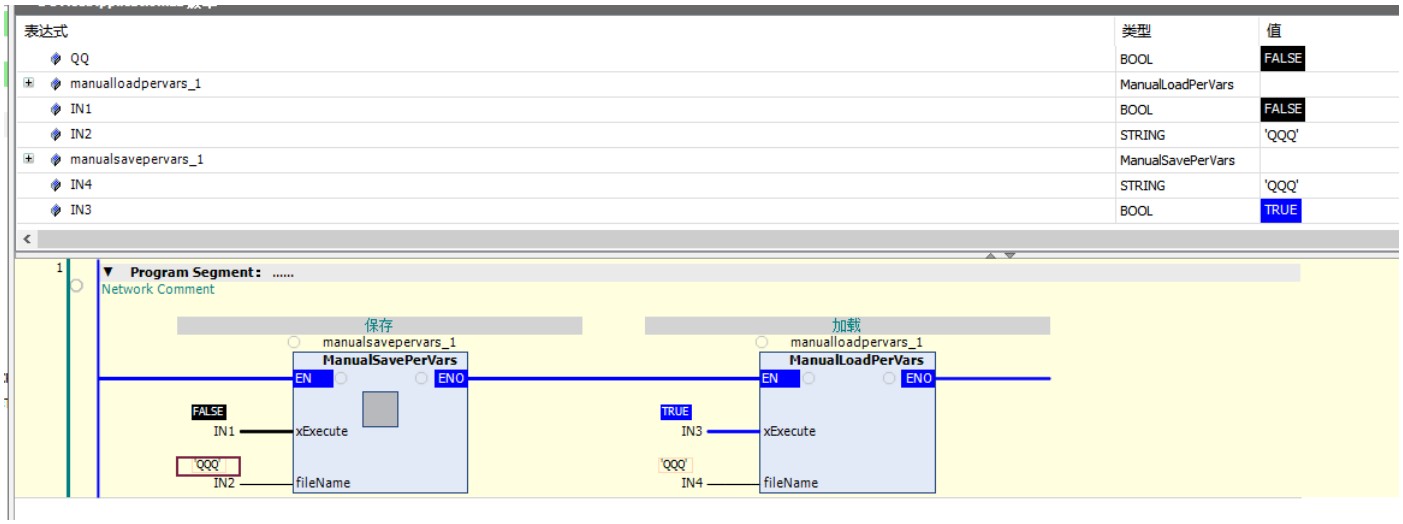
3.读取/保存的变量要放在保持区域中。



ST:



LD:



※注意事项

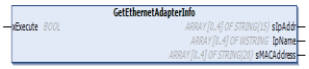
3.13 系统指令

指令列表

指令类别	名称	FB/FC	功能
系统状态	GetEthernetAdapterInfo	FB	获取PLC硬件网口信息
	ManualSavePerVars	FC	获取设备是否属于凌臣

3.13.1 GetEthernetAdapterInfo 获取PLC硬件网口信息

①指令格式

指令	功能	FB/FC	LD 表现	ST表现
GetEthernetAdapterInfo	获取PLC硬件网口信息	FB		tEthernetAdapterInfo_0(xExecute:=IN1 , sIpAddr=>OUT1 , IpName=>OUT2 , sMACAddress=>OUT3);

②相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
xExecute	触发引脚	BOOL	-	0	触发条件

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
sIpAddr	网口地址	ARRAY [0..4] OF STRING(15);	-	-	得到的PLC网口地址
IpName	网口名称	ARRAY [0..4] OF WSTRING;	-	-	得到的PLC网口名称
sMACAddress	MAC地址	ARRAY [0..4] OF STRING(20);	-	-	得到的PLC网口MAC地址

数据类型	布尔	位串				整数								实数		时刻、持续时间 日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATA	TOD	DT	STRING
xExecute	√																			
sIpAddr																				√
IpName																				√
sMACAddress																				√

③程序示例

1.通过触发xExecute引脚来获取当前PLC的IP地址.IP名称.MAC地址给到对应的变量数组中。

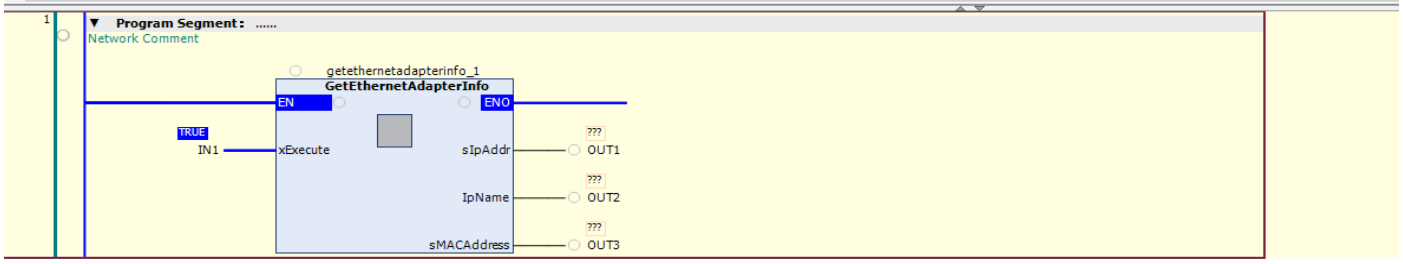
ST:

表达式	类型	值
IN1	BOOL	TRUE
OUT1	ARRAY [0..4] OF ST...	
OUT1[0]	STRING(15)	'192.168.0.99'
OUT1[1]	STRING(15)	'0.0.0.0'
OUT1[2]	STRING(15)	'0.0.0.0'
OUT1[3]	STRING(15)	'192.168.9.99'
OUT1[4]	STRING(15)	'192.168.8.99'
OUT2	ARRAY [0..4] OF W...	
OUT2[0]	WSTRING	"enp0s31f6"
OUT2[1]	WSTRING	"sit0"
OUT2[2]	WSTRING	"ip6tnl0"
OUT2[3]	WSTRING	"enp2s0"
OUT2[4]	WSTRING	"enp3s0"
OUT3	ARRAY [0..4] OF ST...	
OUT3[0]	STRING(20)	'88:88:88:88:87:88'
OUT3[1]	STRING(20)	''
OUT3[2]	STRING(20)	''
OUT3[3]	STRING(20)	''
OUT3[4]	STRING(20)	''
GetEthernetAdapterInfo_0	GetEthernetAdapter...	

```
1  GetEthernetAdapterInfo_0(xExecute:=IN1, sIpAddr=>OUT1 , IpName=>OUT2 , sMACAddress=>OUT3 );RETURN
```

LD:

表达式	类型	值
getethernetadapterinfo_1	GetEthernetAdapter...	
IN1	BOOL	TRUE
OUT1	ARRAY [0..4] OF ST...	
OUT1[0]	STRING(15)	'192.168.0.99'
OUT1[1]	STRING(15)	'0.0.0.0'
OUT1[2]	STRING(15)	'0.0.0.0'
OUT1[3]	STRING(15)	'192.168.9.99'
OUT1[4]	STRING(15)	'192.168.8.99'
OUT2	ARRAY [0..4] OF W...	
OUT2[0]	WSTRING	"enp0s31f6"
OUT2[1]	WSTRING	"sit0"
OUT2[2]	WSTRING	"ip6tnl0"
OUT2[3]	WSTRING	"enp2s0"
OUT2[4]	WSTRING	"enp3s0"
OUT3	ARRAY [0..4] OF ST...	
OUT3[0]	STRING(20)	'88:88:88:88:87:88'
OUT3[1]	STRING(20)	''
OUT3[2]	STRING(20)	''
OUT3[3]	STRING(20)	''
OUT3[4]	STRING(20)	''



※注意事项

3.13.2 ManualSavePerVars　获取设备是否属于凌臣

①指令格式

指令	功能	FB/FC	LD 表现	ST表现
ManualSavePerVars	获取设备是否属于凌臣	FB		GetDeviceIsLCT(GetDeviceIsLCT=> OUT1);

②相关变量

输出变量

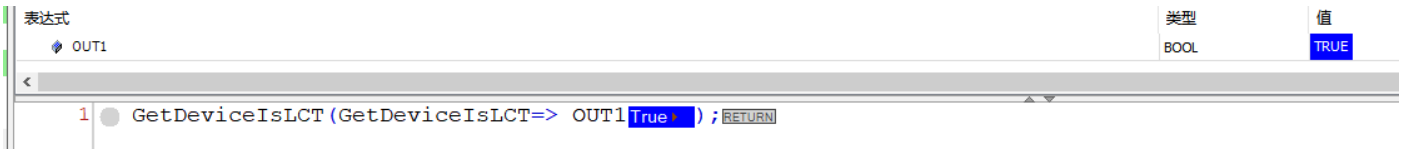
输出变量	名称	数据类型	有效范围	初始值	描述
GetDeviceIsLCT	获取结果	BOOL	-	-	设备是否属于凌臣

数据类型	布尔	位串				整数								实数		时刻、持续时间 日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATA	TOD	DT	STRING
GetDeviceIsLCT	√																			

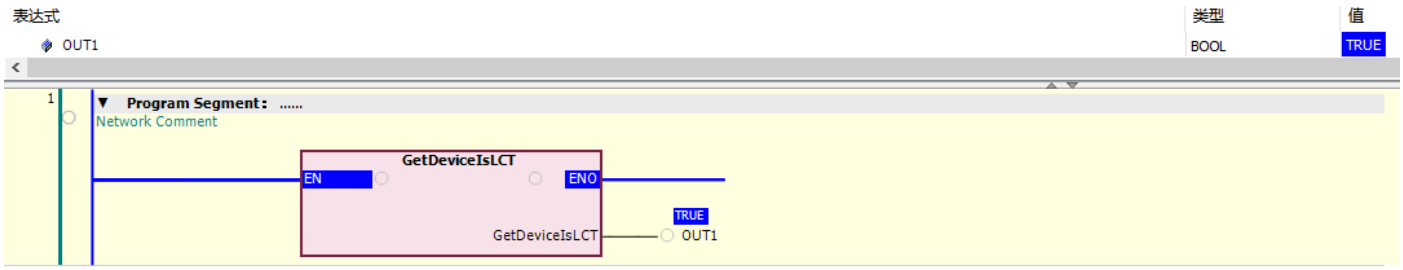
③程序示例

1.判断设备是否属于凌臣

ST:



LD:


表达式
类型
值
OUT1
BOOL
TRUE
1
Program Segment: .....
Network Comment
GetDeviceIsLCT
EN
ENO
TRUE
GetDeviceIsLCT
OUT1

第四章： 串口通讯指令

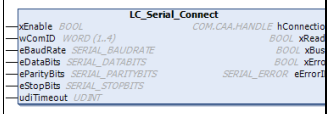
4.1 自由串口指令

指令列表

指令类别	名称	FB/FC	功能
自由串口通信	LC_Serial_Connect	FB	串口自由协议配置指令
	LC_Serial_Send	FB	串口自由协议发送指令
	LC_Serial_Recv	FB	串口自由协议接收指令

4.1.1 LC_Serial_Connect 串口自由协议配置指令。

①指令格式

指令	功能	FB/FC	LD 表现	ST表现
LC_Serial_Connect	串口自由协议配置指令	FB		LC_Serial_Connect(xEnable:=, wComID:=, eBaudRate:=, eDataBits:=, eParityBits:=, eStopBits:=, udiTimeout:=, hConnection=>, xReady=>, xBusy=>, xError=>, eErrorID=>);

②相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
xEnable	功能块使能	BOOL	TRUE/FALSE	FALSE	电平触发：TRUE_打开串口，FALSE_关闭串口
wComID	COM端口号	WORD	(1..4)	1	COM端口id号，与PLC型号相关
eBaudRate	波特率	ENUM	-	-	波特率
eDataBits	数据位	ENUM	-	-	数据位
eParityBits	奇偶校验	ENUM	-	-	奇偶校验
eStopBits	停止位	ENUM	-	-	停止位
udiTimeout	连接超时时间	UDINT	遵照数据类型	-	连接超时时间

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
hConnection	串口句柄	HANDLE	-	-	串口句柄
xReady	功能块有效标志	BOOL	TRUE/FALSE	-	功能块有效标志
xBusy	功能块执行标志	BOOL	TRUE/FALSE	-	功能块执行标志
xError	功能块异常标志	BOOL	TRUE/FALSE	-	功能块异常标志
eErrorID	功能块异常错误码	ENUM	-	-	功能块异常错误码

枚举类型

- SERIAL_BAUDRATE 波特率

枚举参数	值	描述
B4800	4800	4800 bps
B9600	9600	9600 bps
B19200	19200	19200 bps
B38400	38400	38400 bps
B57600	57600	57600 bps
B115200	115200	115200 bps

- SERIAL_PARITYBITS 奇偶校验

枚举参数	值	描述
EVEN	0	偶校验
ODD	1	奇校验
NONE	2	无校验

- SERIAL_DATABITS 数据位

枚举参数	值	描述
BIT5	5	5 数据位
BIT6	6	6 数据位
BIT7	7	7 数据位

BIT8	8	8 数据位

- SERIAL_STOPBITS 停止位

枚举参数	值	描述
BIT1	1	1 停止位
BIT2	2	2 停止位

- SERIAL_ERROR 错误码

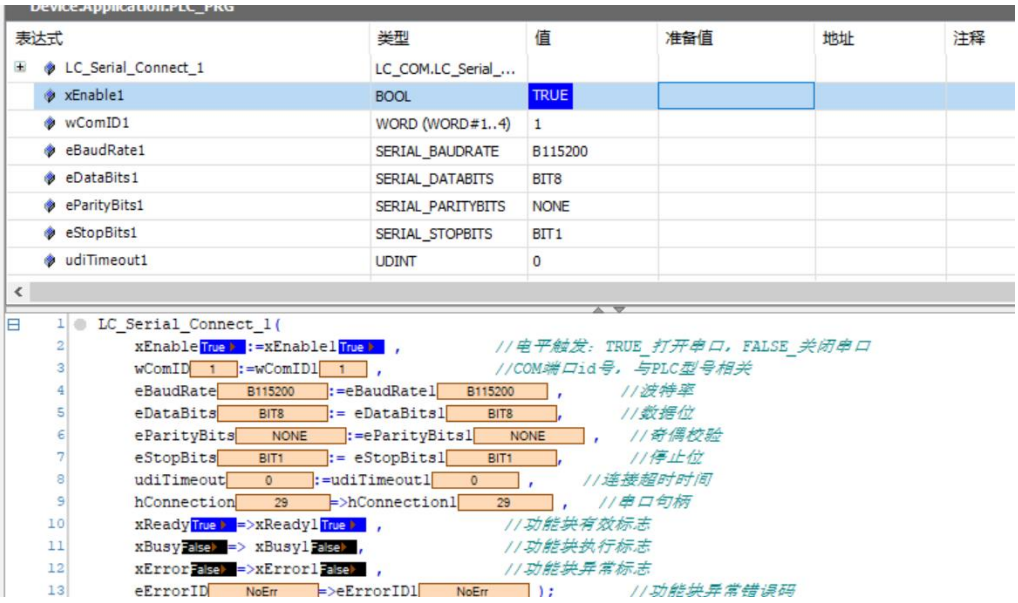
枚举参数	值	描述
NoErr	0	无报错
WrongComIdex	1	端口号错误
ComAlreadyOpen	2	端口已打开
ComAlreadyClose	3	端口已关闭
ComOpenFail	4	端口打开失败
ComCloseFail	5	端口关闭失败
NotAllowedOpenAndClose	6	端口不能同时打开和关闭
FirstError	5000	第一个库特定错误
TimeOut	5001	功能块超时
Abort	5002	打断
INVALID_HANDLE	5003	句柄无效
ErrUnknown	5004	错误位置
WrongParameter	5005	参数错误
WriteIncomplete	5006	写入不完整
FirstMF	5050	首次制造特定错误
LastErr	5099	最后的库特定错误

数据类型	布尔	位串				整数								实数		时刻、持续时间 日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATA	TOD	DT	STRING
xEnable	√																			
wComID			√																	
eBaudRate	ENUM																			
eDataBits	ENUM																			
eParityBits	ENUM																			
eStopBits	ENUM																			
udiTimeout								√												
hConnection	HANDLE																			
xReady	√																			
xBusy	√																			
xError	√																			
eErrorID	ENUM																			

功能说明:

用户可更改运行期间数据传输速率、校验位等参数。

程序示例:



注意事项:

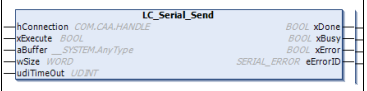
1）Serial_Connect 的连接句柄只能被Serial_Recv或Serial_Send其中一种指令进行使用；

2）请确保相同连接句柄的多条指令，在同一用户任务内使用，请勿多任务使用！ 3）在线修改参数无法立即生效，需要重新使能指令。

4.1.2 LC_Serial_Send 串口自由协议发送指令。

①指令格式

指令	功能	FB/FC	LD 表现	ST表现

LC_Serial_Send	串口自由协议发送指令	FB		LC_Serial_Send(hConnection:=, xExecute:=, aBuffer:= , wSize:= , udiTimeOut:= , xDone=>, xBusy=>, xError=>, eErrorID=>);

②相关变量

输入输出变量

输入输出变量	名称	数据类型	有效范围	初始值	描述
hConnection	初始化	HANDLE	-	-	串口句柄

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
xExecute	发送	BOOL	TRUE/FALSE	-	上升沿触发发送
aBuffer	发送数据缓冲区	ANY	-	-	发送数据类型见注释
wSize	发送数据长度	WORD	遵照数据类型	-	发送数据长度
udiTimeOut	发送超时	UDINT	遵照数据类型	-	发送超时

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
xDone	功能块完成标志	BOOL	TRUE/FALSE	-	发送完成
xBusy	功能块执行标志	BOOL	TRUE/FALSE	-	发送中
xError	功能块异常标志	BOOL	TRUE/FALSE	-	功能块异常标志
eErrorID	功能块异常错误码	ENUM	-	-	功能块异常错误码

数据类型	布尔	位串				整数								实数		时刻、持续时间 日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATA	TOD	DT	STRING
hConnection	HANDLE																			
xExecute	√																			
aBuffer	ANY																			√
wSize		√																		
udiTimeOut								√												
xDone	√																			
xBusy	√																			
xError	√																			
eErrorID	ENUM																			

注释：缓冲区可支持数据类型：BYTE、WORD、DWORD、SINT、USINT、INT、UINT、DINT、UDINT、REAL、LWORD、LINT、ULINT、LREAL、STRING以及一维数组；

缓冲区不支持数据类型：

二维及以上多维数组

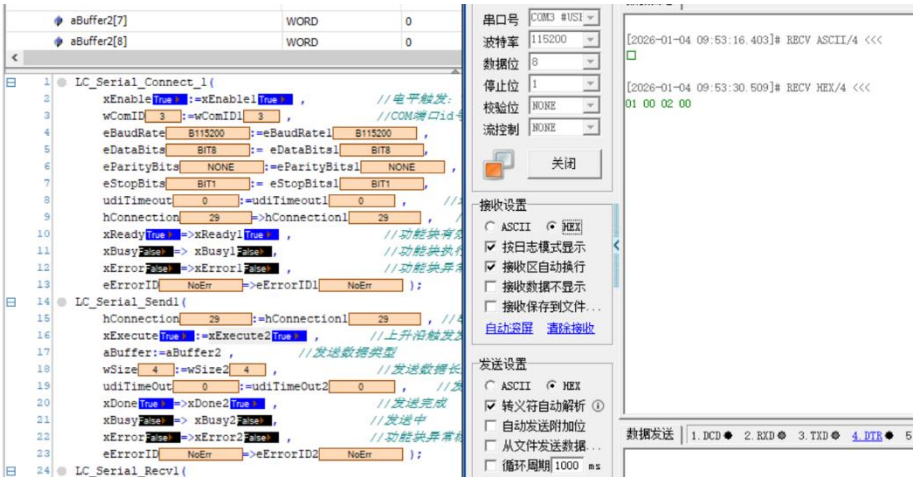
特殊类型：TIME、TIME OF DAY(TOD)、DATA、DATA OF TIME(DT)、POINTER、STRUCT。

功能说明:

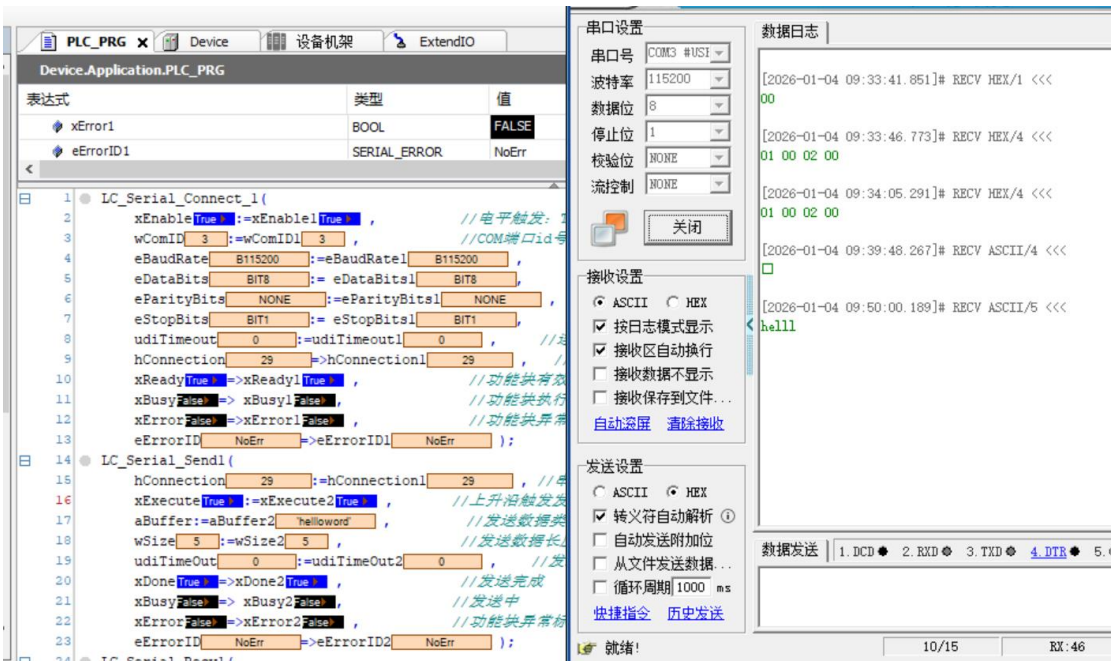
启动数据传输，将发送缓存区数据通过串口总线发送

程序示例:

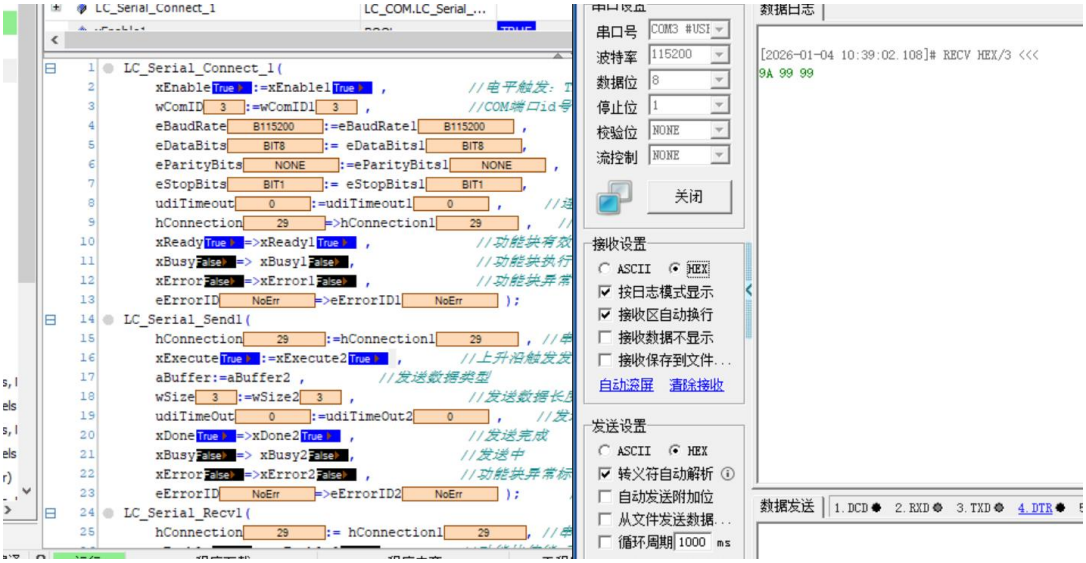
(1)aBuffer传入数据类型为WORD



(2)aBuffer传入数据类型为STRING



(3) aBuffer传入数据类型为REAL



注意事项

1）需要配合Serial_Connect指令进行使用。

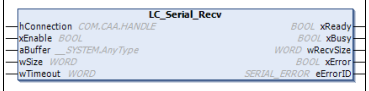
2）在线修改参数无法立即生效，需要重新使能指令。

3）aBuffer 不可直接使用I/Q/M直接变量，需要通过AT映射方式到上述支持的变量/数组数据

4.1.3 LC_Serial_RECV 串口自由协议接收指令。

①指令格式

指令	功能	FB/FC	LD 表现	ST表现

LC_Serial_Recv	串口自由协议接收指令	FB		LC_Serial_Recv(hConnection:=, xEnable:=, aBuffer:= , wSize:= , udiTimeout:=, Ready=> , xBusy=> , wRecvSize=> , xError=> , eErrorID=>);

②相关变量

输入输出变量

输入输出变量	名称	数据类型	有效范围	初始值	描述
hConnection	初始化	HANDLE	-	-	串口句柄

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
xEnable	功能块使能	BOOL	TRUE/FALSE	-	功能块使能 高电平有效
aBuffer	接收数据缓冲区	ANY	-	-	接收数据类型见注释
wSize	接收数据长度	WORD	遵照数据类型	-	接收数据长度
udiTimeout	超时时间	UDINT	遵照数据类型	-	超时时间

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
xReady	功能块完成标志	BOOL	TRUE/FALSE	-	可以接收
xBusy	功能块执行标志	BOOL	TRUE/FALSE	-	接收中
wRecvSize	接收数据字节数	WORD	遵照数据类型	-	串口接收数据字节数
xError	功能块异常标志	BOOL	TRUE/FALSE	-	功能块异常标志
eErrorID	功能块异常错误码	ENUM	-	-	功能块异常错误码

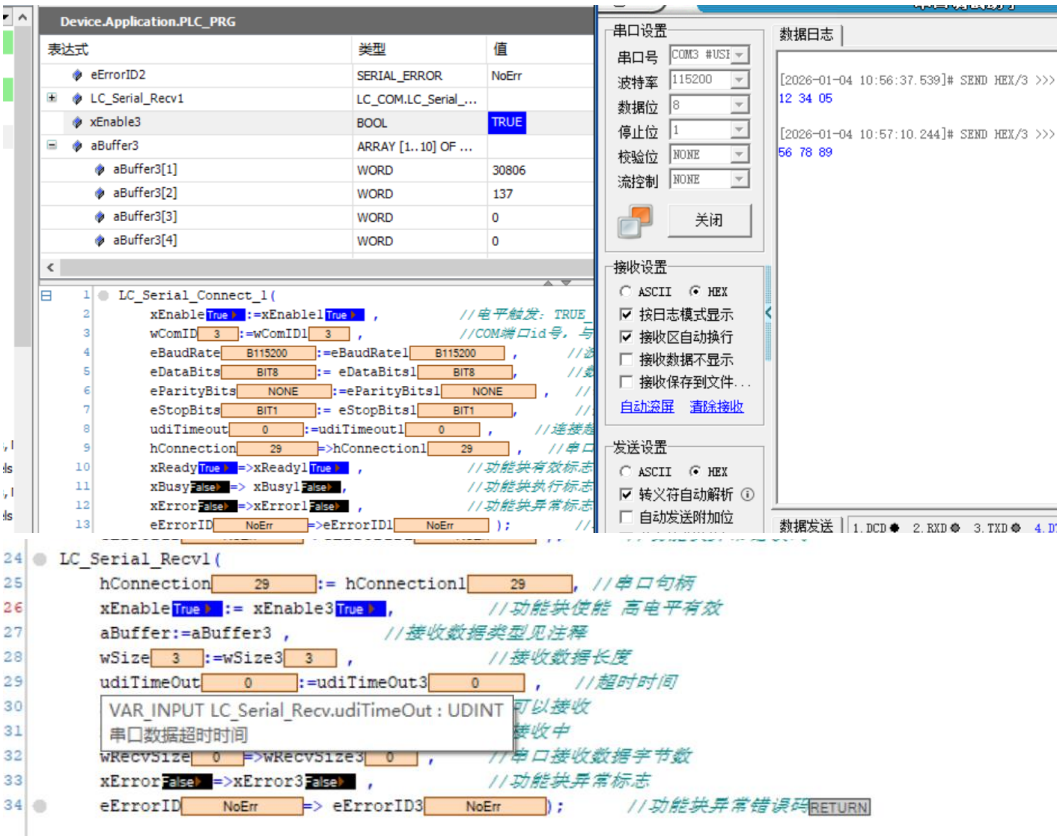
数据类型	布尔	位串				整数								实数		时刻、持续时间 日期、字符串				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATA	TOD	DT	STRING
hConnection	HANDLE																			
xExecute	√																			
aBuffer	√																			
wSize		√																		
udiTimeOut								√												
xDone	√																			
xBusy	√																			
xError	√																			
eErrorID	ENUM																			

功能说明:

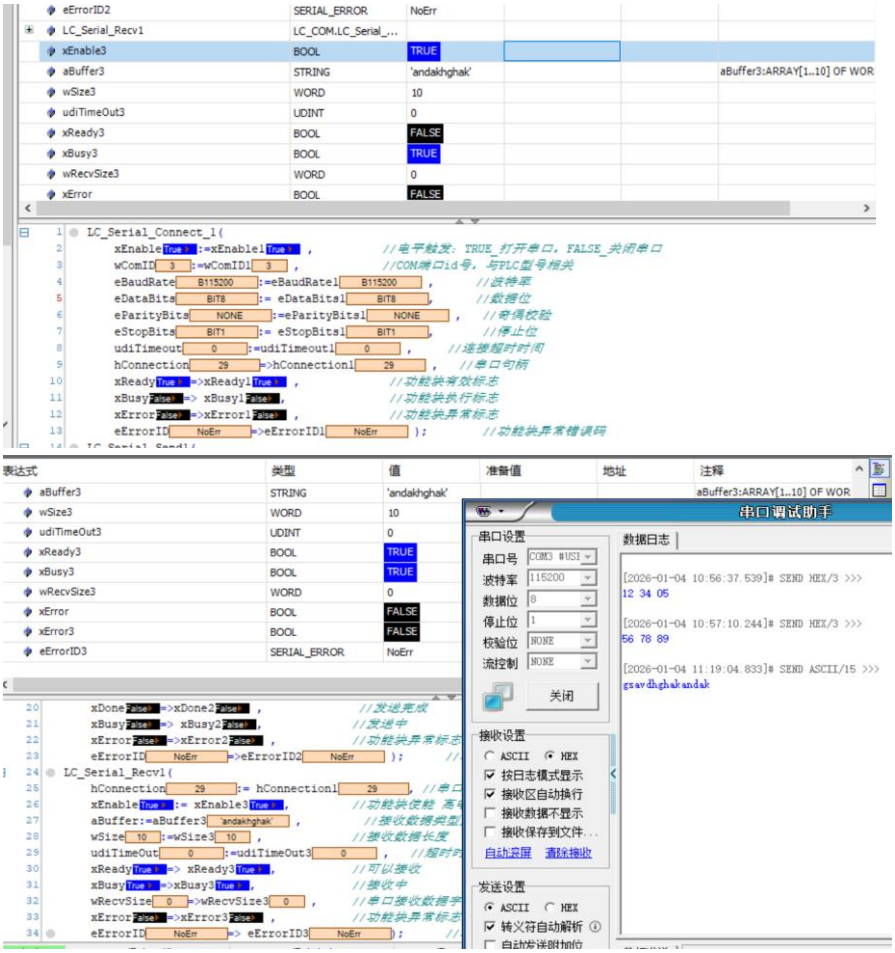
启动数据接收，将串口总线数据存储到接收缓存区。

程序示例:

(1) aBuffer接收数据类型为WORD



(2) aBuffer接收数据类型为STRING



注意事项:

1) 需要配合Serial_Connect指令一起使用

2) 如果设置了超时时间参数，当接收串口数据失败或串口无数据时，开始计算超时时间。如果超时时间达到仍无数据，则报接收超时错误；如果超时时间未到达，同时串口接收到数据，则超时时间重置。

3) 当wRecvSize = wSize，表明数据接收完成，xReady将置位一个周期，下一个周期复位，同时wRecvSize计数清零。

4) aBuffer支持的数据类型：BYTE、WORD、DWORD、SINT、USINT、INT、UINT、DINT、UDINT、REAL及对应的一维数组。

5) aBuffer不可直接使用I/Q/M直接变量，需要通过AT映射方式到上述支持的变量/数组数据。

6) 在线修改参数后，需要重新使能指令才生效

4.2 ModbusRTU读写操作指令

4.2.1 OpenSerial 打开通讯串口

①指令格式

指令	功能	FB/FC	LD 表现	ST表现
OpenSerial	打开通讯串口	FB		_OpenSerial(xExecute:=, bOpen:=, comIndex:=, baudrate:=, stopbit:=, PARITY:=, status=>, xDone=>, xError=>, errorCode=>);

②相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
xExecute	触发执行	BOOL	[TRUE, FALSE]	FALSE	触发执行
bOpen	串口开关	BOOL	[TRUE, FALSE]	FALSE	串口开关
comIndex	串口号	BYTE	遵从数据类型	-	串口号
baudrate	波特率	ENUM	遵从数据类型	-	波特率
stopbit	停止位	ENUM	遵从数据类型	-	停止位
PARITY	校验位	ENUM	遵从数据类型	-	校验位

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
status	串口状态	BOOL	[TRUE, FALSE]	FALSE	串口状态
xDone	执行到位	BOOL	[TRUE, FALSE]	FALSE	执行到位
xError	功能块错误	BOOL	-	-	功能块错误
errorCode	错误码	ModbusRtu_ErrorCode	遵从数据类型	-	错误码

数据类型

	布尔	位串				整数							实数		时刻、持续时间 日期、字符串					
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
xExecute	√																			
bOpen	√																			
comIndex		√																		
baudrate	ENUM																			
stopbit	ENUM																			
PARITY	ENUM																			

status	√																			
xDone	√																			
xError	√																			
errorCode	ModbusRtu ErrorCode																			

③功能说明

bOpen引脚用于打开串口，xExecute引脚用来触发指令，

同一个串口不能重复打开，如果重复触发报错，要先触发一次关闭串口的指令

④程序示例

ST语言

```
10:
    // 单口功能测试
    _OpenSerial(
        xExecute:=x_触发执行,
        bOpen:=x_串口开关,
        comIndex:=by_串口号,
        baudrate:=e_LC波特率,
        stopbit:=e_LC停止位,
        PARITY:=e_LC校验位,
        status=>x_串口状态,
        xDone=>x_执行到位,
        xError=>xArr_功能块错误[1],
        errorCode=>eArr_LC错误码[1]);
    IF _OpenSerial.xDone THEN
```

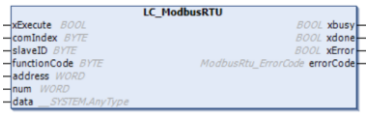
```
CASE 测试选择 OF
10:
    // 单口功能测试
    _OpenSerial(
        xExecute:=x_触发执行,
        bOpen:=x_串口开关,
        comIndex:=by_串口号,
        baudrate:=e_LC波特率,
        stopbit:=e_LC停止位,
        PARITY:=e_LC校验位,
        status=>x_串口状态,
        xDone=>x_执行到位,
        xError=>xArr_功能块错误[1],
        errorCode=>eArr_LC错误码[1]);
    IF _OpenSerial.xDone THEN
        x_触发执行:=FALSE;
    END_IF
```

'ModbusRtu_ErrorCode.NoErr'表示原始值'0'

4.2.2 LC_ModbusRTU RTU主站通讯

①指令格式

指令	功能	FB/FC	LD 表现	ST表现

LC_ModbusRTU	RTU主站通讯	FB		__ModbusRTUMaster(xExecute:=, comIndex:=, slaveID:=, functionCode:=, address:=, num:=, data:=, xbusy=>, xdone=>, xError=>, errorCode=>)

②相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
xExecute	执行通信	BOOL	[TRUE, FALSE]	FALSE	执行通信
comIndex	串口号	BYTE	遵从数据类型	-	串口号
slaveID	从站ID	BYTE	遵从数据类型	-	从站ID
functionCode	功能码	BYTE	遵从数据类型	-	功能码
address	偏移地址	WORD	遵从数据类型	-	偏移地址
num	数量,	WORD	遵从数据类型	-	数量,
data	数据	WORD	遵从数据类型	-	数据

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
xbusy	执行中	BOOL	[TRUE, FALSE]	FALSE	执行中
xDone	执行完成	BOOL	[TRUE, FALSE]	FALSE	执行到位
xError	功能块错误	BOOL	-	-	功能块错误
errorCode	错误码	ModbusRtu_ErrorCode	遵从数据类型	-	错误码

数据类型

	布尔	位串				整数							实数		时刻、持续时间 日期、字符串					
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
xExecute	√																			
comIndex		√																		
slaveID		√																		
functionCode		√																		
address			√																	
num			√																	
data			√																	

status	√																			
xDone	√																			
xError	√																			
errorCode	ModbusRtu ErrorCode																			

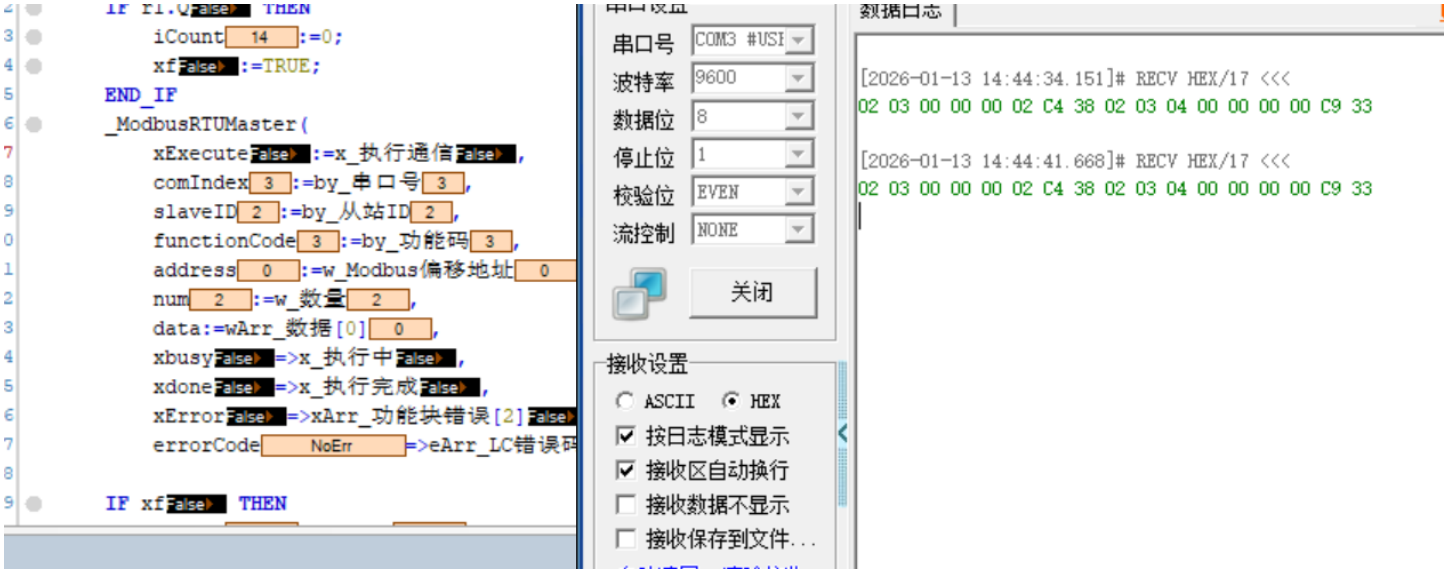
③功能说明

主站连接从站进行通讯

④程序示例

ST语言

```
IF f1.Q THEN
    iCount 14 :=0;
    xf :=TRUE;
END_IF
_ModbusRTUMaster(
    xExecute :=x_执行通信,
    comIndex 3 :=by_串口号 3 ,
    slaveID 2 :=by_从站ID 2 ,
    functionCode 3 :=by_功能码 3 ,
    address 0 :=w_Modbus偏移地址 0
    num 2 :=w_数量 2 ,
    data:=wArr_数据[0] 0 ,
    xbusy =>x_执行中 ,
    xdone =>x_执行完成 ,
    xError =>xArr_功能块错误[2]
    errorCode NoErr =>eArr_LC错误码
IF xf THEN
```



附录

A.1 指令速查表

基 本 指 令			
指令类型	指令说明	指令名	所在库
算术运算指令	加法指令	ADD（FUN）	无
	乘法指令	MUL（FUN）	
	减法指令	SUB（FUN）	
	除法指令	DIV（FUN）	
	取余指令	MOD（FUN）	
赋值指令	赋值指令	MOVE（FUN）	无
逻辑运算指令	与指令	AND（FUN）	无
	或指令	OR（FUN）	
	异或指令	XOR（FUN）	
	取非指令	NOT（FUN）	
移位指令	左移指令	SHL（FUN）	无
	右移指令	SHR（FUN）	
	循环左移指令	ROL（FUN）	
	循环右移指令	ROR（FUN）	
选择指令	二选一指令	SEL（FUN）	无
	取最大值指令	MAX（FUN）	
	取最小值指令	MIN（FUN）	
	极限值指令	LIMIT（FUN）	
	多选一指令	MUX（FUN）	
比较指令	大于指令	GT（FUN）	无
	小于指令	LT（FUN）	
	大于等于指令	GE（FUN）	
	小于等于指令	LE（FUN）	
	等于指令	EQ（FUN）	
	不等于指令	NE（FUN）	
数据类型 转换指令	布尔类型转换指令	BOOL_TO_<TYPE>(FUN)	无
	字节类型转换指令	BYTE_TO_<TYPE>(FUN)	
	日期类型转换指令	DATE_TO_<TYPE>(FUN)	
	长整型转换指令	DINT_TO_<TYPE>(FUN)	
	日期时间类型转换指令	DT_TO_<TYPE>(FUN)	
	双字类型转换指令	DWORD_TO_<TYPE>(FUN)	
	整数类型转换指令	INT_TO_<TYPE>(FUN)	

指令类型	指令说明	指令名	所在库
数据类型转换指令	字类型转换指令	WORD_TO_<TYPE>(FUN)	无
	实数类型转换指令	REAL_TO_<TYPE>(FUN)	
	短整型转换指令	SINT_TO_<TYPE>(FUN)	
	字符类型转换指令	STRING_TO_<TYPE>(FUN)	
	时钟类型转换指令	TIME_TO_<TYPE>(FUN)	
	时间类型转换指令	TOD_TO_<TYPE>(FUN)	
	无符号长整型转换指令	UDINT_TO_<TYPE>(FUN)	
	无符号整型转换指令	UINT_TO_<TYPE>(FUN)	
	无符号短整型转换指令	USINT_TO_<TYPE>(FUN)	
	截短转换指令	TRUNC(FUN)	
初等数学运算指令	绝对值指令	ABS(FUN)	无
	平方根指令	SQRT(FUN)	
	自然对数指令	LN(FUN)	
	常用对数指令	LOG(FUN)	
	指数指令	EXP(FUN)	
	正弦指令	SIN(FUN)	
	余弦指令	COS(FUN)	
	正切指令	TAN(FUN)	
	反正弦指令	ASIN(FUN)	
	反余弦指令	ACOS(FUN)	
	反正切指令	ATAN(FUN)	
	幂指令	EXPT(FUN)	
地址运算指令	取地址指令	ADR(FUN)	无
	取地址内容指令	^(FUN)	
	位地址指令	BITADR(FUN)	
	索引指令	INDEXOF （FUN）	
	数据类型大小指令	SIZEOF （FUN）	
调用指令	调用运算指令	CAL(FUN)	无
初始化操作指令	初始化操作指令	INI(FUN)	无
字符串指令	结合字符串指令	CONCAT(FUN)	Standard.lib
	删除字符指令	DELETE(FUN)	
	查找字符串指令	FIND(FUN)	
	插入字符串指令	INSERT(FUN)	
	左边取字符串指令	LEFT(FUN)	
	字符串长度指令	LEN(FUN)	
	中间取字符串指令	MID(FUN)	
	替换字符串指令	REPLACE(FUN)	
	右边取字符串指令	RIGHT(FUN)	
库版本信息指令	读取库版本查看	Version_Util(FUN)	Util.lib

指令类型	指令说明	指令名	所在库
BCD 转换指令	BCD 码转整型指令	BCD_TO_INT(FUN)	Util.lib
	整型转 BCD 码指令	INT_TO_BCD(FUN)	
位/字节操作指令	位提取指令	EXTRACT(FUN)	Util.lib
	位整合指令	PACK(FUN)	
	位赋值指令	PUTBIT(FUN)	
	位拆分指令	UNPACK(FB)	
高等数学运算指令	微分	DERIVATIVE(FB)	Util.lib
	积分	INTEGRAL(FB)	
	整型统计	STATISTICS_INT(FB)	
	实型统计	STATISTICS_REAL(FB)	
	平方偏差	VARIANCE(FB)	
控制器指令	比例控制器	P(FB)	Util.lib
	比例微分控制器	PD(FB)	
	比例积分微分控制器	PID(FB)	
	比例积分微分控制器	PID_FIXCYCLE(FB)	
信号发生器指令	脉冲信号发生器	BLINK(FB)	Util.lib
	典型周期信号发生器	GEN(FB)	
SFC 动作控制指令	SFC 动作控制	SFCActionControl(FB)	Iecsfc.lib
函数操纵器指令	特征曲线	CHARCURVE(FB)	Util.lib
	整型限速	RAMP_INT(FB)	
	实型限速	RAMP_REAL(FB)	
模拟量处理指令	滞后	HYSTERESIS(FB)	Util.lib
	上下限报警	LIMITALARM(FB)	
双稳态指令	置位优先双稳态器	SR(FB)	Standard.lib
	复位优先双稳态器	RS(FB)	
触发器	上升沿检测触发器	R_TRIG(FB)	Standard.lib
	下降沿检测触发器	F_TRIG(FB)	
计数器	递增计数器	CTU(FB)	Standard.lib
	递减计数器	CTD(FB)	
	递增递减计数器	CTUD(FB)	
定时器	普通定时器	TP(FB)	Standard.lib
	通电延时定时器	TON(FB)	
	断电延时定时器	TOF(FB)	
	实时时钟	RTC(FB)	

A.2 IEC 标准指令表

CodeSys 遵循国际电工技术委员会（IEC） 的 IEC61131-3 标准，在这个标准中明确规定了作为可编程控制器必须具有的指令，即标准指令。标准指令包括类型转换指令、数字运算指令、位移位指令、比较指令、 类型转换指令、定时器、计数器等， 所有 IEC61131-3 标准规定指令， 见下表。

IEC 标 准 指 令		
指令类型	指令说明	指令名
算术运算指令	加法指令	ADD
	乘法指令	MUL
	减法指令	SUB
	除法指令	DIV
	取余指令	MOD
逻辑指令	与指令	AND
	或指令	OR
	异或指令	XOR
	取非指令	NOT
移位指令	左移指令	SHL
	右移指令	SHR
	循环左移指令	ROL
	循环右移指令	ROR
选择指令	二选一指令	SEL
	取最大值指令	MAX
	取最小值指令	MIN
	极限值指令	LIMIT
	多选一指令	MUX
比较指令	大于指令	GT
	小于指令	LT
	大于等于指令	GE
	小于等于指令	LE
	等于指令	EQ
	不等于指令	NE
类型转换指令	布尔类型转换指令	BOOL_TO_<TYPE>
	字节类型转换指令	BYTE_TO_<TYPE>
	日期类型转换指令	DATE_TO_<TYPE>
	长整型转换指令	DINT_TO_<TYPE>
	日期时间类型转换指令	DT_TO_<TYPE>
	双字类型转换指令	DWORD_TO_<TYPE>
	整数类型转换指令	INT_TO_<TYPE>
	字类型转换指令	WORD_TO_<TYPE>
	实数类型转换指令	REAL_TO_<TYPE>
	短整型转换指令	SINT_TO_<TYPE>
	字符类型转换指令	STRING_TO_<TYPE>
	时钟类型转换指令	TIME_TO_<TYPE>
	时间类型转换指令	TOD_TO_<TYPE>
	无符号长整型转换指令	UDINT_TO_<TYPE>

	无符号整型转换指令	UINT_TO_<TYPE>
	无符号短整型转换指令	USINT_TO_<TYPE>
	截短转换指令	TRUNC
初等数学运算指令	绝对值指令	ABS
	平方根指令	SQRT
	自然对数指令	LN
	常用对数指令	LOG
	指数指令	EXP
	正弦指令	SIN
	余弦指令	COS
	正切指令	TAN
	反正弦指令	ASIN
	反余弦指令	ACOS
	反正切指令	ATAN
	幕指令	EXPT
地址运算指令	取地址指令	ADR
	取地址内容指令	^
	位地址指令	BITADR
	索引指令	INDEXOF
	数据类型大小指令	SIZEOF
调用指令	调用运算指令	CAL
初始化操作指令	初始化操作指令	INI
字符串指令	结合字符串指令	CONCAT
	删除字符指令	DELETE
	查找字符串指令	FIND
	插入字符串指令	INSERT
	左边取字符串指令	LEFT
	字符串长度指令	LEN
	中间取字符串指令	MID
	替换字符串指令	REPLACE
	右边取字符串指令	RIGHT
BCD 转换指令	BCD 码转整型指令	BCD_TO_INT
	整型转 BCD 码指令	INT_TO_BCD
双稳态指令	置位优先双稳态器	SR
	复位优先双稳态器	RS
触发器	上升沿检测触发器	R_TRIG
	下降沿检测触发器	F_TRIG
计数器	递增计数器	CTU
	递减计数器	CTD
	递增递减计数器	CTUD
定时器	普通定时器	TP
	通电延时定时器	TON
	断电延时定时器	TOF
	实时时钟	RTC