技术资料

文档编号：AD220002

AC500

可升级的控制器
灵活的自动化控制系统

CodeSys 指令手册

声明

(1) 在实际的系统构建时，请先确认系统组成设备、装置，如使用过程中对额定值、性能留有余量，以及万一发生故障时将危险降到最低的安全电路等。

(2) 为了安全使用系统，请获取组成系统的各个设备、装置的指南及安装说明书，在确认好包括“安全注意事项”、“安全要点”等内容后再使用。

(3) 本文件所属的产品/系统只允许由具备相关工作经验要求的合格人员进行操作，确认适合系统的规格、法规及规定。

(4) 未经ABB公司许可，严禁擅自对本资料的一部分或全部内容进行篡改及散发。

(5) 本资料的记录内容为测试指导，在应用过程，请根据现场实际情况适当调整，或者咨询ABB技术支持人员。

(6) 我们已对文档中所述内容与硬件和软件的一致性作过检查。然而不排除存在偏差的可能性，因此我们不保证文档中所述内容与硬件和软件完全一致。文档中的数据都按规定经过检测，必要的修正值包含在下一版本中。

(7) 本资料的记录内容若有改版，恕不另行通知。

目 录

第1章　CoDeSys 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

CoDeSys 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。

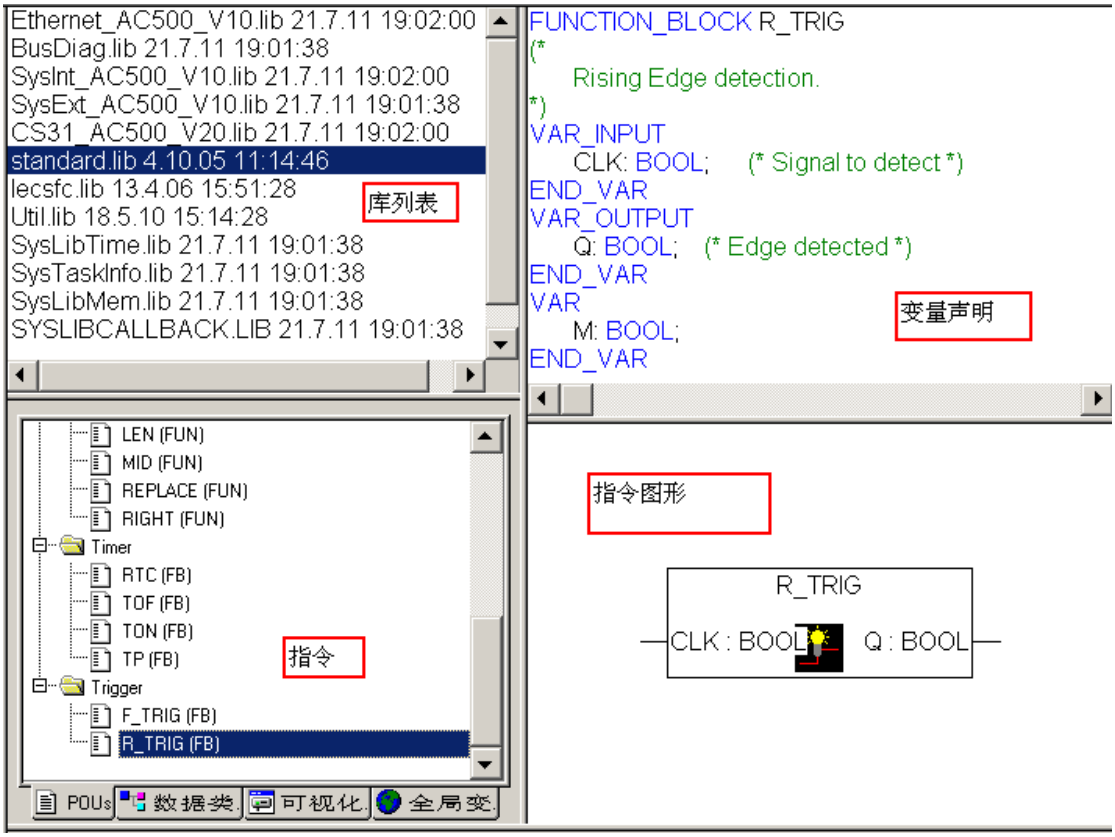
常用功能指令	算术运算指令（如 ADD 加法指令）
	赋值指令（MOVE 赋值指令）
	逻辑运算指令（如 AND 与指令）
	移位指令（如 SHL 左移指令）
	选择指令（如 MAX 取最大值指令）
	比较指令（如 GT 大于指令）
	类型转换指令（如 REAL_TO_<TYPE>实数类型转换指令）
	初等数学运算指令（如 ABS 绝对值指令）
	地址运算指令（如 ADR 取地址指令）
常用功能块指令	双稳态指令（如 SR 置位优先双稳态器）
	触发器(如 R_TRIG 上升沿检测触发器)
	计数器（如 CTU 递增计数器）
	定时器（如 TON 通电延时定时器）
	信号发生器指令（如 BLINK 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，库文件管理器窗口如下图所示。

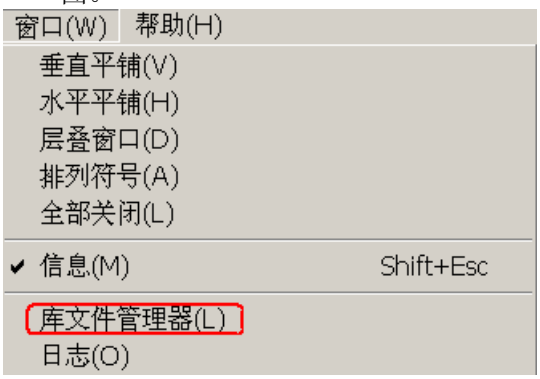


库文件管理器窗口分为“库列表”、“指令”、“变量声明”和“指令图形”这 4 部分。

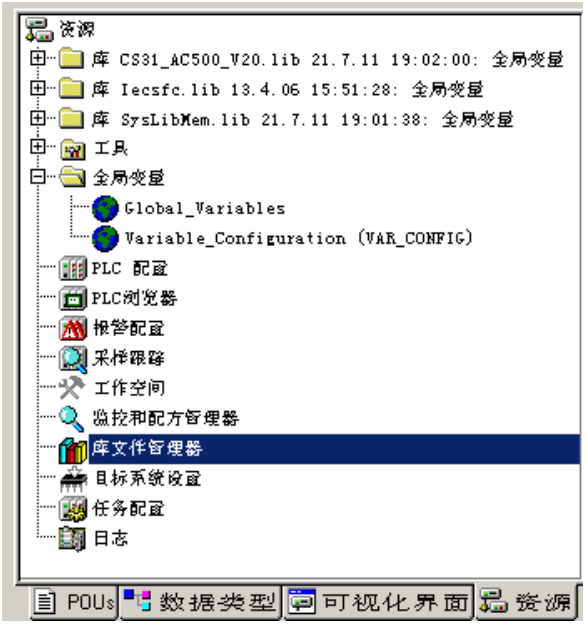
- **库列表** 列出当前工程中添加的所有库文件名。
- **指令** 列出该库文件中包含的指令并按类分别列出。
- **变量声明** 列出所选指令的变量声明。
- **指令图形** 显示所选指令的图形，左边为输入端，右边为输出端。

1. 打开库管理器

- 点击编程界面菜单拦的“窗口(W)”，鼠标左键点击“**库文件管理器(L)**”，如下图。



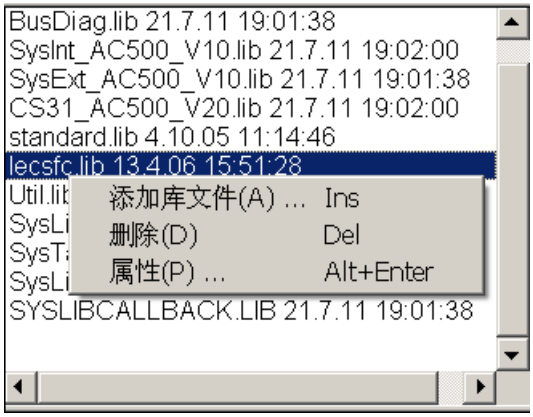
- 点击对象管理器中资源”，打开资源窗口，再双击“库文件管理器”，如下图。



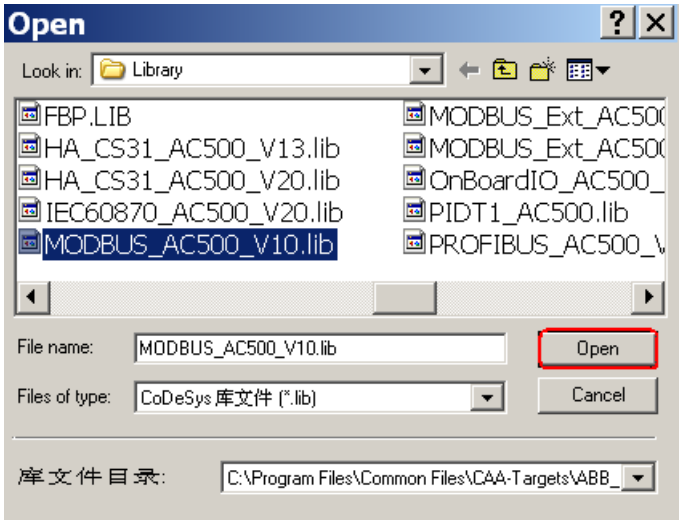
2. 指令库的添加

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“Open”。 不论哪种指令库，只需要打开对应的*.lib 文件即可。



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。

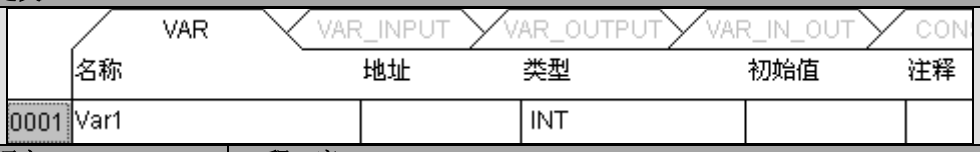
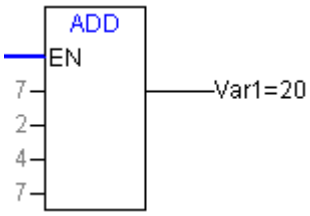
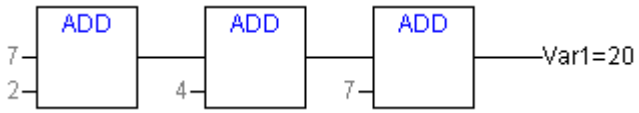
第2章　基本指令

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 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:=7+2+4+7;　（*结果 Var1 为 20*）</td></tr>
<tr><td>指令列表（IL）</td><td>LD　7
ADD　2,4,7
ST　Var1　（*结果 Var1 为 20*）</td></tr>
<tr><td>功能块（FBD）</td><td> 
</td></tr>
</table>

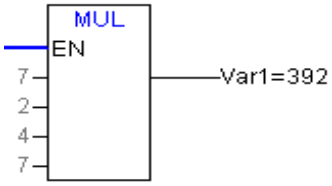
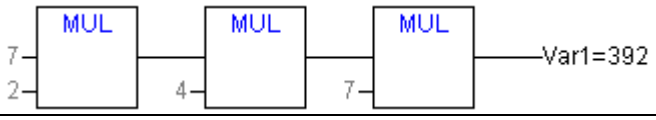
提示：

- 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。
- 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。

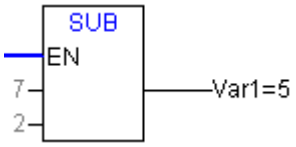
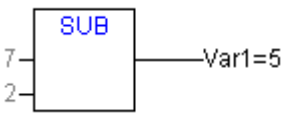
指令使用举例

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	
编程语言	**程　序**
梯形图（LD）	
结构化文本（ST）	Var1:=7*2*4*7;　（*结果 Var1 为 392*）
指令列表（IL）	LD　7 MUL　2,4,7 ST　Var1　（*结果 Var1 为 392*）
功能块（FBD）	

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

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	
编程语言	**程　序**
梯形图（LD）	
结构化文本（ST）	Var1:=7-2;　（*结果 Var1 为 5*）
指令列表（IL）	LD　7 SUB　2 ST　Var1　（*结果 Var1 为 5*）
功能块（FBD）	

提示：

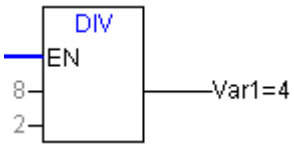
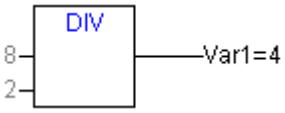
- TIME 型量也可以使用减法功能，两个 TIME 型量相减得到一个新的时间量。例如：t#1m35s - t#50s = t#45s ，但时间结果不能有负值。
- TOD 型量也可以使用减法功能，两个 TOD 型量相减得到一个新的 TIME 型数据，例如：TOD#23:40:30- TOD#00:30:20=T#1390m10s0ms ,但时间结果不能有负值。

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

变量定义

编程语言	程　序
梯形图（LD）	
结构化文本（ST）	Var1:=8/2;　　（*结果 Var1 为 4*）
指令列表（IL）	LD　8 DIV　2 ST　Var1　　（*结果 Var1 为 4*）
功能块（FBD）	

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

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	
编程语言	程　序
梯形图（LD）	MOD EN 9 2 Var1=1
结构化文本（ST）	Var1:=9 MOD 2;　（*结果 Var1 为 1*）
指令列表（IL）	LD　9 MOD　2 ST　Var1　（*结果 Var1 为 1*）
功能块（FBD）	MOD 9 2 Var1=1

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

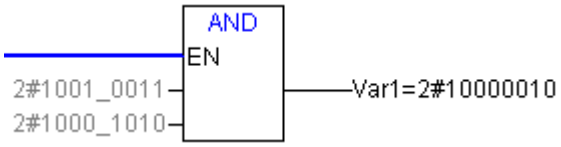
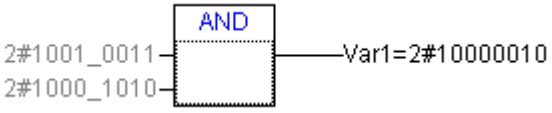
变量定义	
VAR　VAR_INPUT　VAR_OUTPUT　VAR_IN_OUT　CON 名称　地址　类型　初始值　注释 0001　Var1　　INT	
编程语言	程　序
梯形图（LD）	MOVE EN 100 Var1=100
结构化文本（ST）	Var1:=100;　（*结果 Var1 为 100*）
指令列表（IL）	LD　100 MOVE ST　Var1　（*结果 Var1 为 100*）
功能块（FBD）	MOVE 100 Var1=100

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。

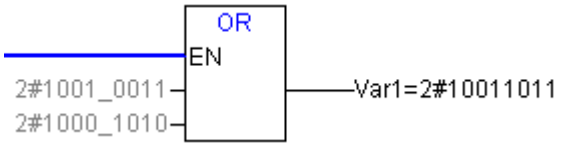
指令使用举例

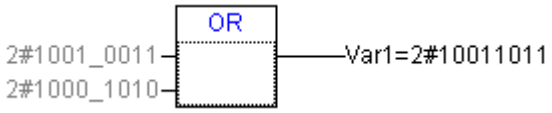
<table>
<tr><td colspan="2">变量定义</td></tr>
<tr><td colspan="2"> 
</td></tr>
<tr><td>编程语言</td><td>程　序</td></tr>
<tr><td>梯形图（LD）</td><td> 
</td></tr>
<tr><td>结构化文本（ST）</td><td>Var1:=2#1001_0011 AND 2#1000_1010;
（*结果 Var1 为 2#10000010*）</td></tr>
<tr><td>指令列表（IL）</td><td>LD　　2#1001_0011
AND　2#1000_1010
ST　　Var1　　（*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

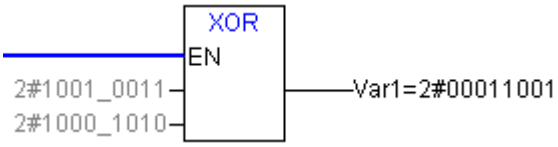
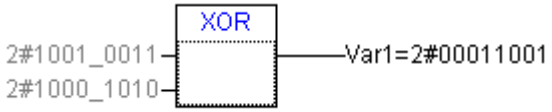
<table>
<tr><td colspan="2">变量定义</td></tr>
<tr><td colspan="2"> 
</td></tr>
<tr><td>编程语言</td><td>程　序</td></tr>
<tr><td>梯形图（LD）</td><td> 
</td></tr>
<tr><td>结构化文本（ST）</td><td>Var1:=2#1001_0011 OR 2#1000_1010 ;
（*结果 Var1 为 2#10011011*）</td></tr>
</table>

指令列表（IL）	LD 2#1001_0011 OR 2#1000_1010 ST Var1 （*结果 Var1 为 2#10011011*）
功能块（FBD）	

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

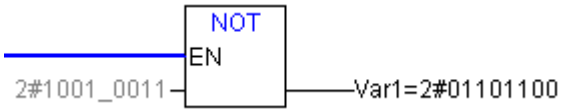
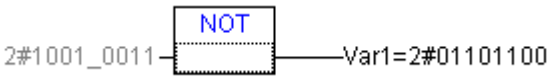
变量定义	
	
编程语言	**程 序**
梯形图（LD）	
结构化文本（ST）	Var1:=2#1001_0011 XOR 2#1000_1010 ; （*结果 Var1 为 2#00011001*）
指令列表（IL）	LD 2#1001_0011 XOR 2#1000_1010 ST Var1 （*结果 Var1 为 2#00011001*）
功能块（FBD）	

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

变量定义	
	
编程语言	**程 序**

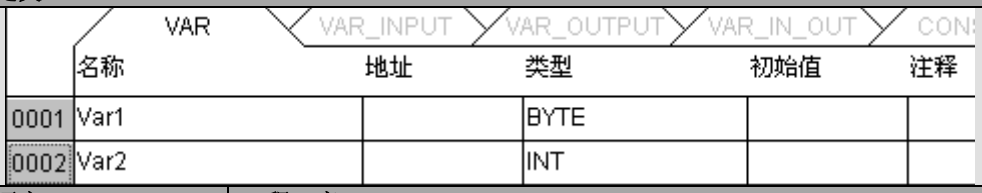
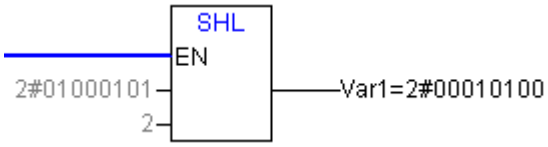
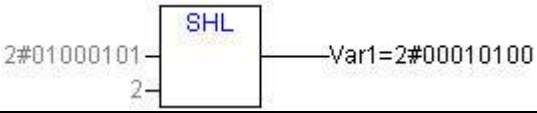
梯形图（LD）	
结构化文本（ST）	Var1:= NOT 2#1001_0011 ; （*结果 Var1 为 2#01101100*）
指令列表（IL）	LD 2#1001_0011 NOT ST Var1 （*结果 Var1 为 2#01101100*）
功能块（FBD）	

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

变量定义	
	
编程语言	**程　序**
梯形图（LD）	
结构化文本（ST）	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。
指令列表（IL）	LD 16#45 SHL 2 ST Var1 （*结果 Var1 为 16#14*）
功能块（FBD）	

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。

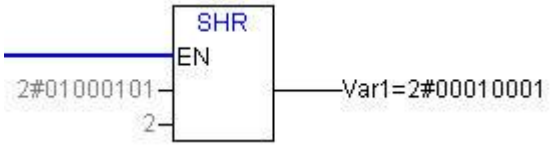
- 输入/输出数据类型：BYTE、INT、WORD、DWORD、SINT、UINT。

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*）
指令列表（IL）	LD　16#45 SHR　2 ST　Var1　（*结果 Var1 为 16#11*）
功能块（FBD）	

2.4.3　ROL——循环左移指令

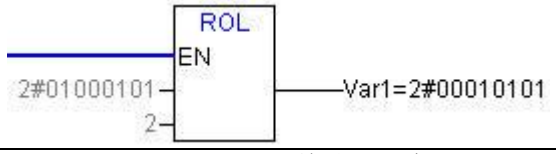
- 功能：对操作数进行按位循环左移，左边移出的位直接补充到右边最低位。
- 输入/输出数据类型：BYTE、INT、WORD、DWORD、SINT、UINT。

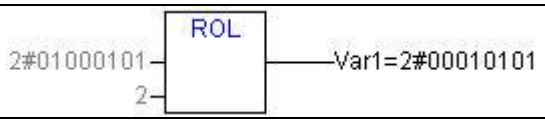
指令使用举例

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。
指令列表（IL）	LD　16#45 ROL　2

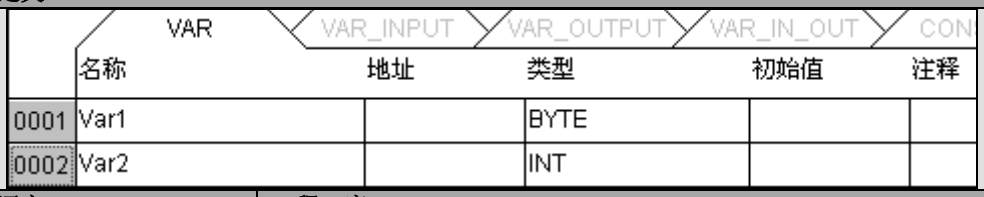
	ST　　Var1　　（*结果 Var1 为 16#15*）
功能块（FBD）	

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



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。
指令列表（IL）	LD　　16#45 ROR　　2 ST　　Var1　　（*结果 Var1 为 16#51*）
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：

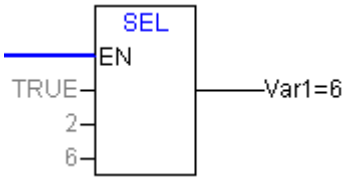
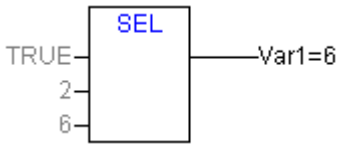
- G 必须是 BOOL 类型，IN0、IN1 和输出数据可以是任意数据类型。

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:=SEL(TRUE,3,4); (*结果 Var1 为 4 *)
指令列表（IL）	LD　TRUE SEL　2,6 ST　Var1　(*结果 Var1 为 6*) LD　FALSE SEL　2,6 ST　Var2　(*结果 Var2 为 2*)
功能块（FBD）	

2.5.2　MAX——取最大值指令

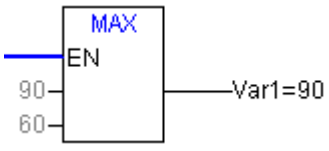
- 功能：在两个输入数据中选择最大值作为输出。
- 指令格式：OUT:=MAX(IN0, IN1)，其中 IN0 和 IN1 分别为第 1 个输入数据和第 2 个输入数据，OUT 是输出数据。
- 输入/输出数据类型：IN0, IN1 和 OUT 可以是任意数据类型。

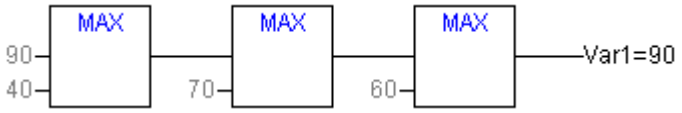
指令使用举例

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		

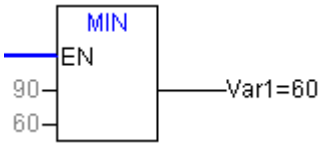
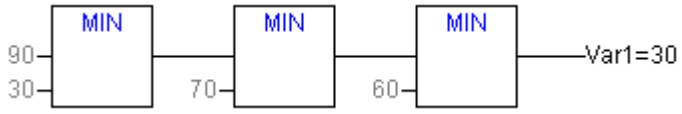
编程语言	程　序
梯形图（LD）	

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 *)
指令列表（IL）	LD 90 MAX 40 MAX 70 MAX 60 ST Var1 （*结果 Var1 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	
	
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 *)
指令列表（IL）	LD 90 MIN 30 MIN 40 MIN 70 ST Var1 （*结果 Var1 为 30*）
功能块（FBD）	

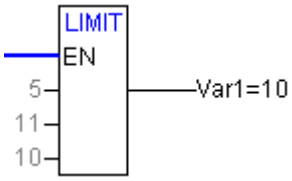
2.5.4 LIMIT——极限值指令

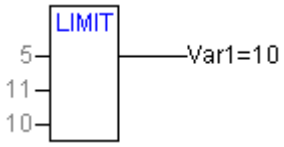
- 功能：判断输入数据是否在最小值和最大值之间，若输入数据在二者之间，则直接把

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。若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 colspan="2">VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CON
名称 | 地址 | 类型 | 初始值 | 注释
0001 | Var1 | | INT | | </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>指令列表（IL）</td><td>LD 90
LIMIT 30，80
ST Var1 (*结果 Var1 为 80*)</td></tr>
<tr><td>功能块（FBD）</td><td> 

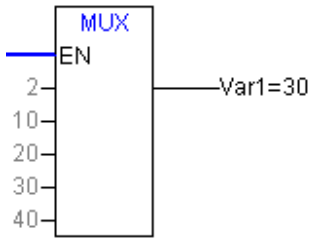
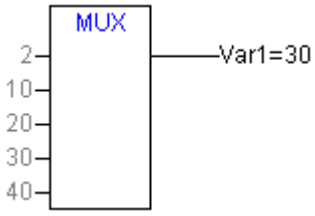
（*11 为输入数据，5 为最小值，10 为最大值*）</td></tr>
</table>

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 colspan="2">VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CON
名称 | 地址 | 类型 | 初始值 | 注释
0001 | Var1 | | INT | | </td></tr>
<tr><td>编程语言</td><td>程 序</td></tr>
</table>

梯形图（LD）	 （*2 为控制数据，对应于 30，所以输出 30 *）
结构化文本（ST）	Var1:=MUX(0,30,40,50,60,70,80);（*结果 Var1 为 30*）
指令列表（IL）	LD 0 MUX 30，40，50，60，70，80 ST Var1 （*结果 Var1 为 30*）
功能块（FBD）	 （*2 为控制数据，对应于 30，所以结果为 30*）

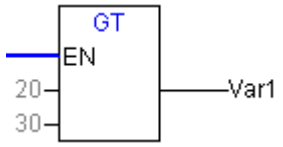
2.6 比较指令

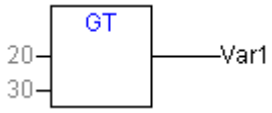
所有的比较指令在执行时均可以带有变量。为了能够更加清楚地说明问题，以下各例只使用常量。

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/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

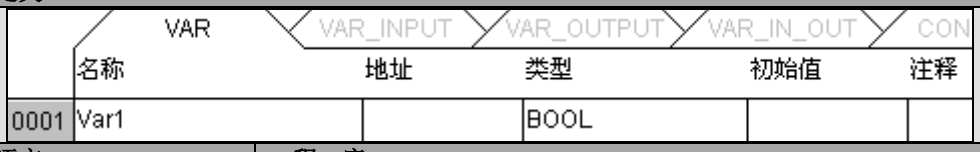
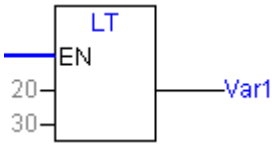
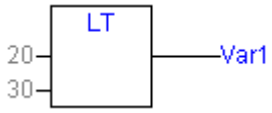
变量定义					
	VAR / VAR_INPUT / VAR_OUTPUT / VAR_IN_OUT / CON				
	名称	地址	类型	初始值	注释
0001	Var1		BOOL		
编程语言	**程　序**				
梯形图（LD）	 （*结果 Var1 为 FALSE*）				
结构化文本（ST）	Var1:=20>30;				
指令列表（IL）	LD 20				

	GT 30 ST Var1 （*结果 Var1 为 FALSE*）
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/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。

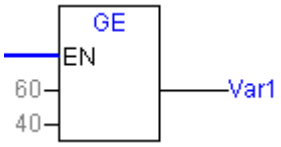
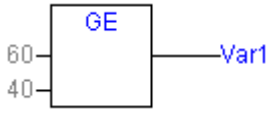
指令使用举例

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*）
指令列表（IL）	LD 20 LT 30 ST Var1　　（*结果 Var1 为 TRUE*）
功能块（FBD）	 （*结果 Var1 为 TRUE*）

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。

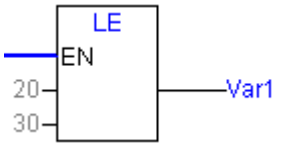
指令使用举例

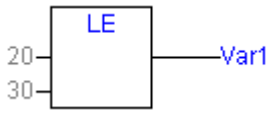
变量定义	
VAR / VAR_INPUT / VAR_OUTPUT / VAR_IN_OUT / CON 名称 地址 类型 初始值 注释 0001 Var1 BOOL	
编程语言	**程　序**
梯形图（LD）	 （*结果 Var1 为 TRUE*）
结构化文本（ST）	VAR1:=60>=40;　　（*结果 Var1 为 TRUE*）
指令列表（IL）	LD　60 GE　40 ST　Var1　　（*结果 Var1 为 TRUE*）
功能块（FBD）	 （*结果 Var1 为 TRUE*）

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/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

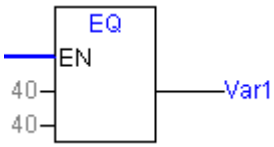
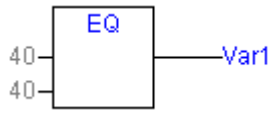
变量定义	
VAR / VAR_INPUT / VAR_OUTPUT / VAR_IN_OUT / CON 名称 地址 类型 初始值 注释 0001 Var1 BOOL	
编程语言	**程　序**
梯形图（LD）	 （*结果 Var1 为 TRUE*）
结构化文本（ST）	VAR1:=20<=30;　　（*结果 Var1 为 TRUE*）
指令列表（IL）	LD　20 LE　30 ST　Var1　　（*结果 Var1 为 TRUE*）

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/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。

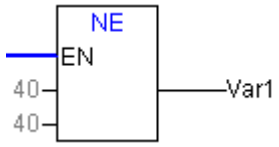
指令使用举例

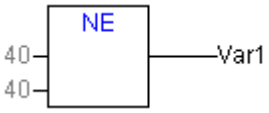
变量定义					
	VAR	VAR_INPUT	VAR_OUTPUT	VAR_IN_OUT	CON
	名称	地址	类型	初始值	注释
	0001 Var1		BOOL		
编程语言	**程　序**				
梯形图（LD）	 （*结果 Var1 为 TRUE*）				
结构化文本（ST）	VAR1:=40=40; （*结果 Var1 为 TRUE*）				
指令列表（IL）	LD 40 EQ 40 ST Var1 （*结果 Var1 为 TRUE*）				
功能块（FBD）	 （*结果 Var1 为 TRUE*）				

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 colspan="2">
<table>
<tr><td></td><td>VAR</td><td>VAR_INPUT</td><td>VAR_OUTPUT</td><td>VAR_IN_OUT</td><td>CON</td></tr>
<tr><td></td><td>名称</td><td>地址</td><td>类型</td><td>初始值</td><td>注释</td></tr>
<tr><td>0001</td><td>Var1</td><td></td><td>BOOL</td><td></td><td></td></tr>
</table>
</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>指令列表（IL）</td><td>LD　40
NE　40
ST　Var1　　（*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。

数据类型转换指令列表

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。

表 4-7-1

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

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

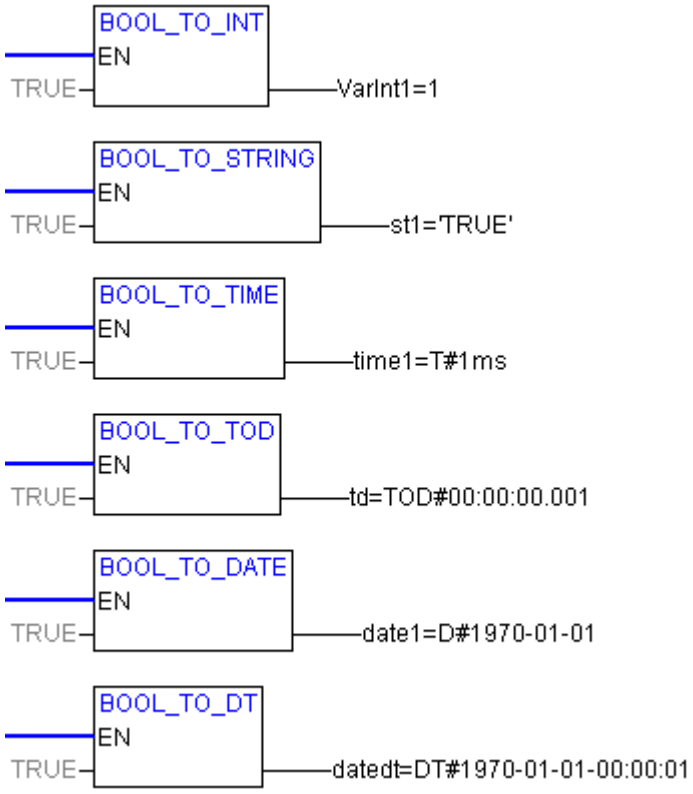
变量定义

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		

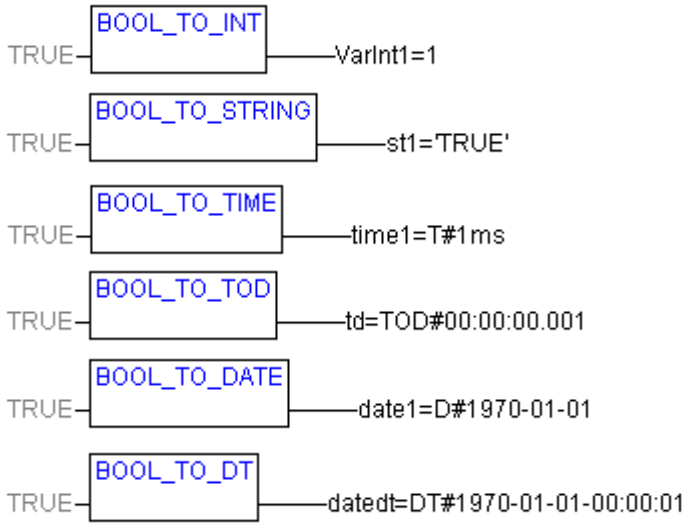
编程语言 | **程 序（部 分）**

梯形图（LD）



结构化文本（ST）

```
VarInt1:=BOOL_TO_INT(TRUE);      (*结果为 1*)
st1:=BOOL_TO_STRING(TRUE);       (*结果为'TRUE'*)
time1:=BOOL_TO_TIME(TRUE);       (*结果为 T#1ms*)
td:=BOOL_TO_TOD(TRUE);     (*结果为 TOD#00:00:00.001*)
```

	date1:=BOOL_TO_DATE(TRUE); (*结果为 D#1970-01-01*) datedt :=BOOL_TO_DT(TRUE); (*结果为 DT#1970-01-01-00:00:01*)
指令列表（IL）	LD TRUE BOOL_TO_INT ST VarInt1 (*结果为 1*) LD TRUE BOOL_TO_STRING ST st1 (*结果为'TRUE'*) LD TRUE BOOL_TO_TIME ST time1 (*结果为 T#1ms*) LD TRUE BOOL_TO_TOD ST td (*结果为 TOD#00:00:00.001*) LD TRUE BOOL_TO_DATE ST date1 (*结果为 D#1970-01-01*) LD TRUE BOOL_TO_DT ST datedt (*结果为 DT#1970-01-01-00:00:01*)
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

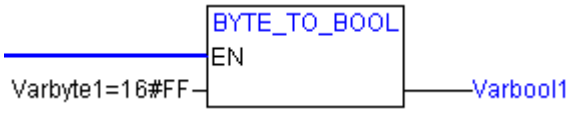
变量定义

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CONSTANT

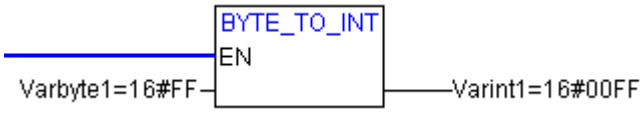
	名称	地址	类型	初始值	注释
0001	Varbool1		BOOL		
0002	Varbyte1		BYTE		
0003	Varint1		INT		
0004	Vartime1		TIME		
0005	Vardt1		DT		
0006	Varreal1		REAL		
0007	Varstring1		STRING		

编程语言 | **程 序（部 分）**

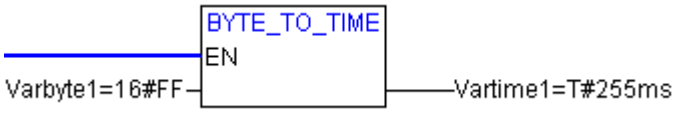
梯形图（LD）



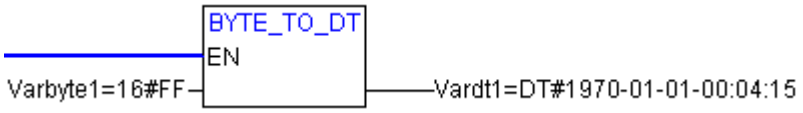
(*结果为 TRUE *)



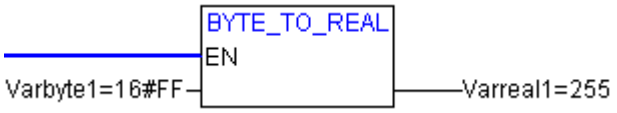
(*结果为 16# 00FF *)



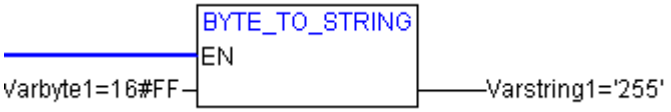
(*结果为 T#255ms *)



(*结果为 DT#1970-01-01-00:04:15 *)



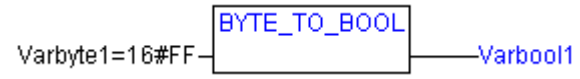
(*结果为 255 *)

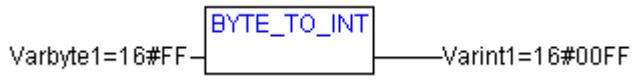


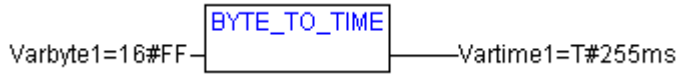
(*结果为字符串'255'*)

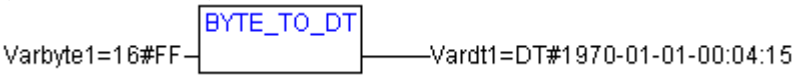
结 构 化 文 本（ST）

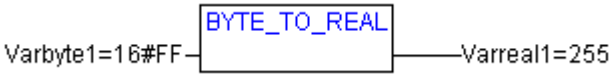
```
Varbyte1:=16#FF                                    (*Varbyte1 取值*)
Varbool1:=BYTE_TO_BOOL(Varbyte1);                  (*结果为 TRUE *)
Varint1:=BYTE_TO_INT(Varbyte1);                    (*结果为 16# FF *)
Vartime1:=BYTE_TO_TIME(Varbyte1);                  (*结果为 T#255ms *)
Vardt1:=BYTE_TO_DT(Varbyte1);
(*结果为 DT#1970-01-01-00:04:15 *)
Varreal1:=BYTE_TO_REAL(Varbyte1);                  (*结果为 255 *)
Varstring1:=BYTE_TO_STRING(Varbyte1); (*结果为字符串'255'*)
```

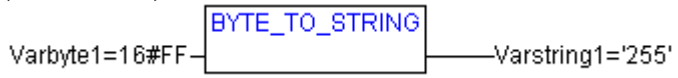
<table>
<tr><td>指令列表（IL）</td><td>
LD 16#FF
ST Varbyte1 (*Varbyte1 取值*)
LD Varbyte1
BYTE_TO_BOOL
ST Varbool1 (*结果为 TRUE *)
LD Varbyte1
BYTE_TO_INT
ST Varint1 (*结果为 16# FF *)
LD Varbyte1
BYTE_TO_TIME
ST Vartime1 (*结果为 T#255ms *)
LD Varbyte1
BYTE_TO_DT
ST Vardt1 (*结果为 DT#1970-01-01-00:04:15 *)
LD Varbyte1
BYTE_TO_REAL
ST Varreal1 (*结果为 255 *)
LD Varbyte1
BYTE_TO_STRING
ST Varstring1 (*结果为字符串'255'*)
</td></tr>
<tr><td>功能块（FBD）</td><td>


(*结果为 TRUE *)


(*结果为 16# 00FF *)


(*结果为 T#255ms *)


(*结果为 DT#1970-01-01-00:04:15 *)


(*结果为 255 *)


(*结果为字符串'255'*)
</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

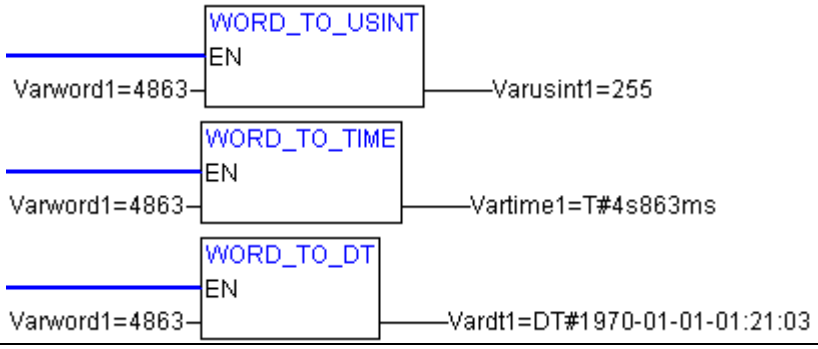
变量定义

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 *)
```

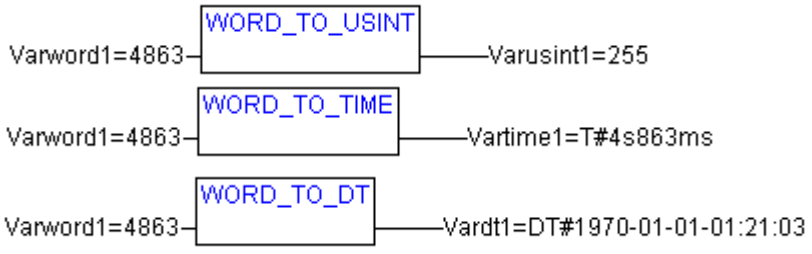
说明：如果将整数 4863（十六进制为 16#12FF）保存为 USINT 型变量，则会丢失高位数据，只显示低位数据 255（十六进制为 16#FF）。

```
Vartime1:=WORD_TO_TIME(Varword1);          (*结果 T#4s863ms*)
Vardt1:=WORD_TO_DT(Varword1);
(*结果 DT#1970-01-01-01:21:03 *)
```

指令列表（IL）

```
LD      4863
ST      Varword1                           (*Varword1 取值*)
LD      Varword1
WORD_TO_USINT
ST      Varusint1                          (*结果 255 *)
LD      Varword1
WORD_TO_TIME
ST      Vartime1                           (*结果 T#4s863ms*)
LD      Varword1
WORD_TO_DT
ST      Vardt1                 (*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

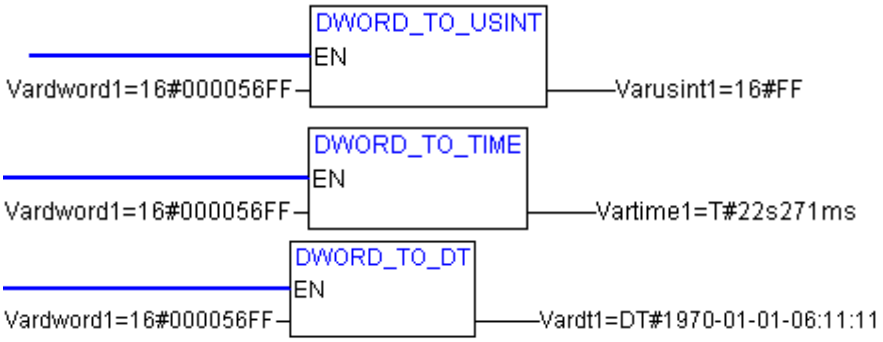
变量定义

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）

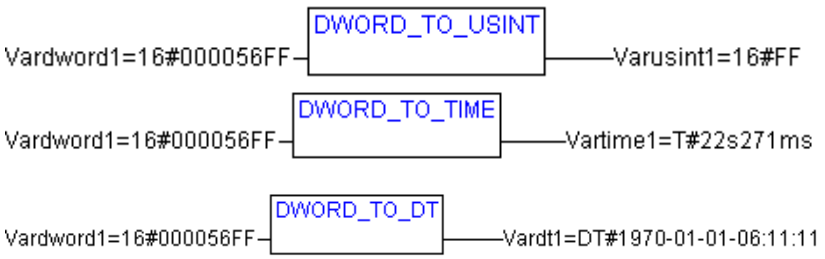
```
Varword1:=16#56FF;                          (*Varword1 取值*)
Varusint1:=DWORD_TO_USINT(Vardword1);       (*结果 255 *)
```

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 *)
```

指令列表（IL）

```
LD      16#56FF
ST      Vardword1                   (*Vardword1 取值*)
LD      Vardword1
DWORD_TO_USINT
ST      Varusint1                   (*结果 255 *)
LD      Vardword1
DWORD_TO_TIME
ST      Vartime1                    (*结果 T#22s271ms*)
LD      Vardword1
DWORD_TO_DT
```

	ST Vardt1 (*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

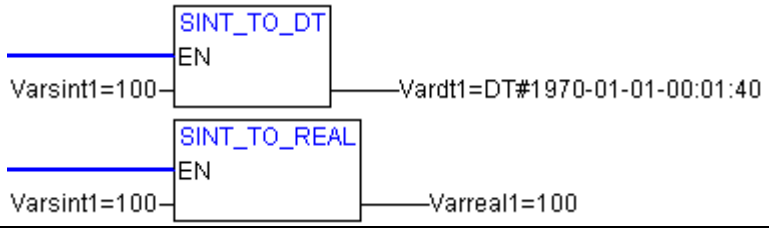
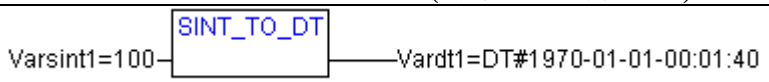
- 功能：把短整型转换为其它数据类型。
- 输入/输出数据类型（参见表 4-7-1）：
- 当 SINT_TO_BOOL 时，输入不等于 0 时输出为 TRUE，当输入等于 0 时输出为 FALSE；当 SINT_TO_TIME、SINT _TO_TOD 时，输入将以毫秒值进行转换；
- 当 SINT_TO_DATE、SINT _TO_DT 时，输入将以秒值进行转换。

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*)
指令列表（IL）	LD 100 ST Varsint1 (*Varsint1 取值*) LD Varsint1 SINT_TO_DT ST Vardt1 (*结果 DT#1970-01-01-00:01:40 *) LD Varsint1 SINT_TO_REAL ST Varreal1 (*结果 Varrea1 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

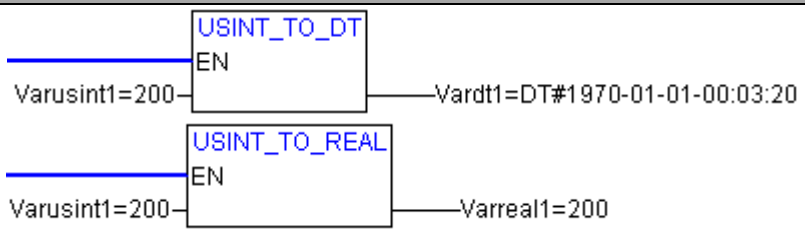
变量定义

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）



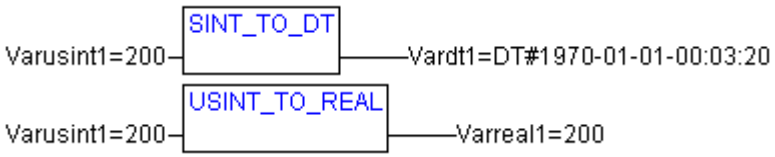
结构化文本（ST）

```
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*)
```

指令列表（IL）

```
LD      200
ST      Varusint1                          (*Varusint1 取值*)
LD      Varusint1
USINT_TO_DT
ST      Vardt1                  (*结果 DT#1970-01-01-00:03:20 *)
LD      Varusint1
USINT_TO_REAL
ST      Varreal1                  (*结果 Varreal1 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）：

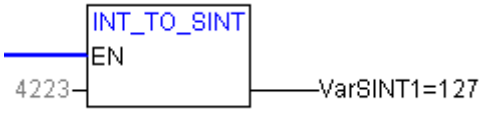
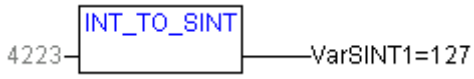
- 当 INT_TO_BOOL 时，输入不等于 0 时输出为 TRUE，当输入等于 0 时输出为 FALSE；
  当 INT_TO_TIME、INT _TO_TOD 时，输入将以毫秒值进行转换；
- 当 INT_TO_DATE、INT _TO_DT 时，输入将以秒值进行转换。

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）。
指令列表（IL）	LD　2 INT_TO_REAL ST　VarREAL1　(*结果 VarREAL1 为 2.0*)
功能块（FBD）	

2.7.8　UINT_TO_<TYPE>——无符号整数类型转换指令

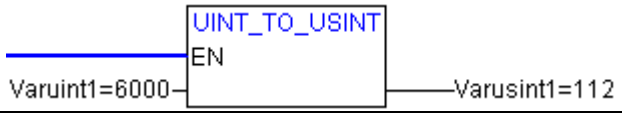
- 功能：无符号整数类型转换为其它数据类型。
- 输入/输出数据类型（参见表 4-7-1）：
- 当UINT_TO_BOOL时，输入不等于0时输出为TRUE，当输入等于0时输出为FALSE；
  当 UINT_TO_TIME、UINT _TO_TOD 时，输入将以毫秒值进行转换；
- 当 UINT_TO_DATE、UINT _TO_DT 时，输入将以秒值进行转换。

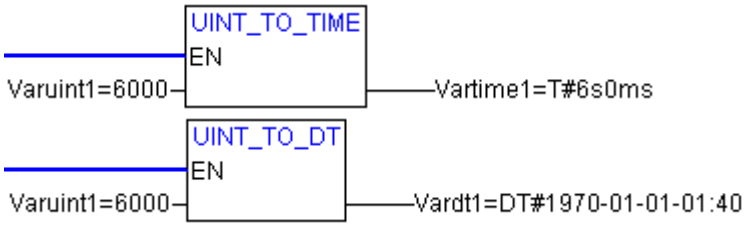
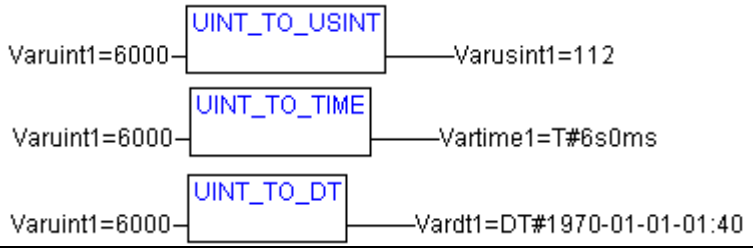
指令使用举例

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);
指令列表（IL）	LD 6000 ST Varuint1 (*Varuint1 取值*) LD Varuint1 UINT_TO_USINT ST Varusint1 (*结果 112*) LD Varuint1 UINT_TO_TIME ST Vartime1 (*结果 T#6s0ms*) LD Varuint1 UINT_TO_DT ST Vardt1 (*结果 DT#1970-01-01-01:40:00 *)
功能块（FBD）	

2.7.9 DINT_TO_<TYPE>——双整数类型转换指令

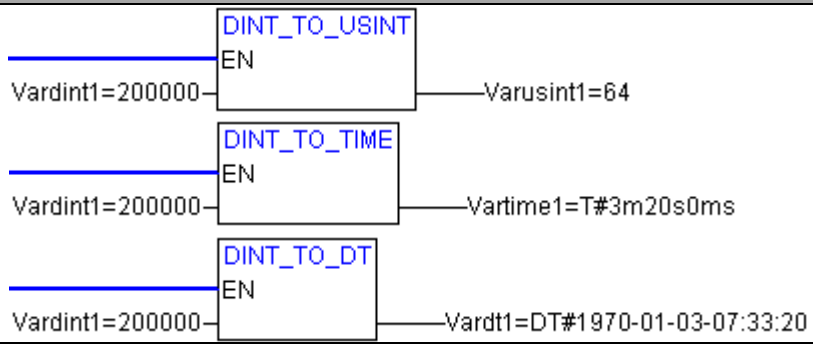
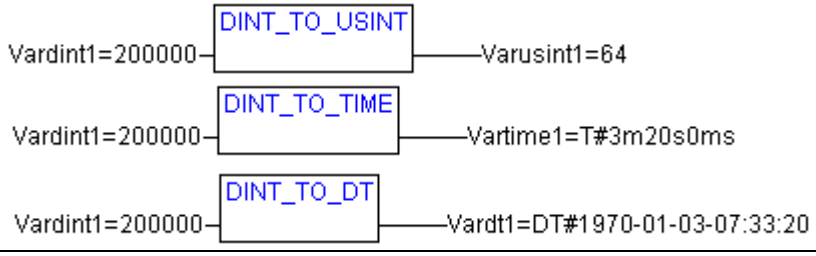
- 功能：双整数类型类型转换为其它数据类型。
- 输入/输出数据类型（参见表 4-7-1）：
- 当DINT_TO_BOOL时，输入不等于0时输出为TRUE，当输入等于0时输出为FALSE；当 DINT_TO_TIME、DINT _TO_TOD 时，输入将以毫秒值进行转换；
- 当 DINT_TO_DATE、DINT _TO_DT 时，输入将以秒值进行转换。

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	Vardint1		DINT		
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 *)
指令列表（IL）	LD 200000 ST Vardint1 (*Vardint1 取值*) LD Vardint1 DINT_TO_USINT ST Varusint1 (*结果 64*) LD Vardint1 DINT_TO_TIME ST Vartime1 (*结果 T#3m20s0ms*) LD Vardint1 DINT_TO_DT ST Vardt1 (*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

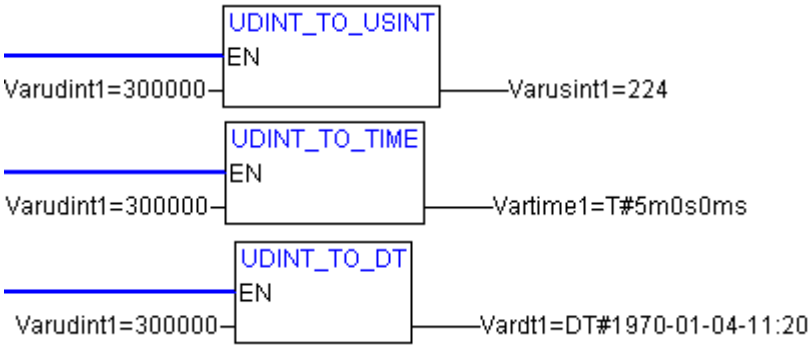
变量定义

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*)
```

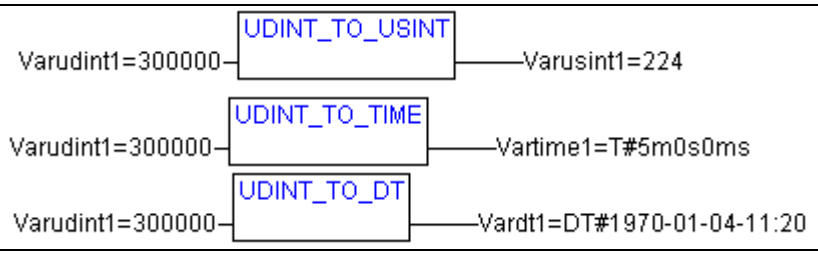
说明：如果将整数 300000（十六进制为 16#493E0）保存为 USINT 型变量，则会丢失高位数据，只显示低位数据 224（十六进制为 16#E0）。

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

指令列表（IL）

```
LD      300000
ST      Varudint1                    (*Varudint1 取值*)
LD      Varudint1
UDINT_TO_USINT
ST      Varusint1                    (*结果 224*)
LD      Varudint1
UDINT_TO_TIME
ST      Vartime1                     (*结果 T#5m0s0ms*)
LD      Varudint1
UDINT_TO_DT
ST      Vardt1          (*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

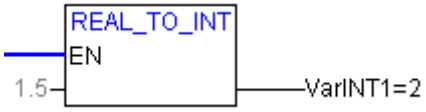
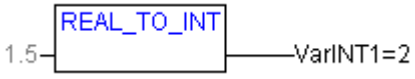
- 功能：把浮点数转换为其它类型数据。把浮点数转换为其它类型数据时，先将值四舍五入成整数值，然后转成新的变量类型。
- 输入/输出数据类型（参见表 4-7-1）：
- 当 REAL_TO_BOOL 时，输入不等于 0 时输出为 TRUE，当输入等于 0 时输出为 FALSE；当 REAL_TO_TIME、REAL_TO_TOD 时，输入将以毫秒值进行转换；
- 当 REAL_TO_DATE、REAL_TO_DT 时，输入将以秒值进行转换。

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*)
指令列表（IL）	LD　2.7 REAL_TO_INT ST　VarINT1　(*结果 VarINT1 为 3*)
功能块（FBD）	

2.7.12 TIME_TO_<TYPE>——时间类型转换指令

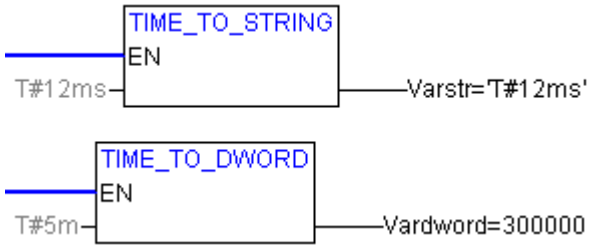
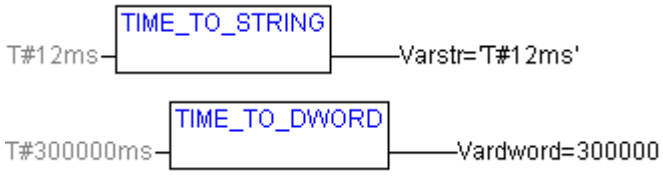
- 功能：把时间型数据转换为其它类型数据，时间在内部以毫秒为单位存储成 DWORD 类型（对于 TIME_OF_DAY 变量从凌晨 00：00 开始）。
- 输入/输出数据类型（参见表 4-7-1）：
- 当 TIME_TO_BOOL 时，输入不等于 0 时输出为 TRUE，当输入等于 0 时输出为 FALSE。

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*)
指令列表（IL）	LD　T#12ms TIME_TO_STRING ST　Varstr　(*结果为‘T#12ms’*) LD　T#300000ms TIME_TO_DWORD ST　Vardword　(*结果为 300000*)
功能块（FBD）	

2.7.13　DATE_TO_<TYPE>——日期类型转换指令

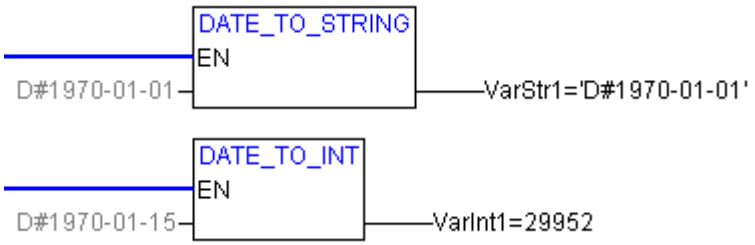
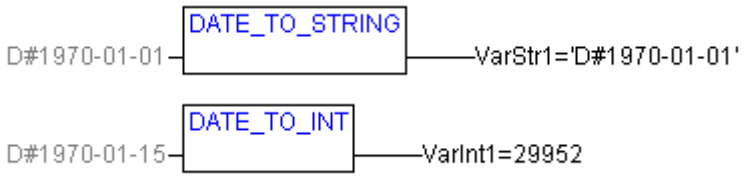
- 功能：把日期型数据转换为其它类型数据，日期在内部以秒为单位存储，时间从 1970 年 1 月 1 日开始。
- 输入/输出数据类型（参见表 4-7-1）：
- 当 DATE_TO_BOOL 时，输入不等于 0 时输出为 TRUE，当输入等于 0 时输出为 FALSE。

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。
指令列表（IL）	LD　　D#1970-01-01 DATE_TO_STRING ST　　VarStr1　　(*结果为‘TRUE’*) LD　　D#1970-01-15 DATE_TO_INT ST　　VarInt1　　(*结果为 29952*)
功能块（FBD）	

2.7.14　DT_TO_<TYPE>——日期时间类型转换指令

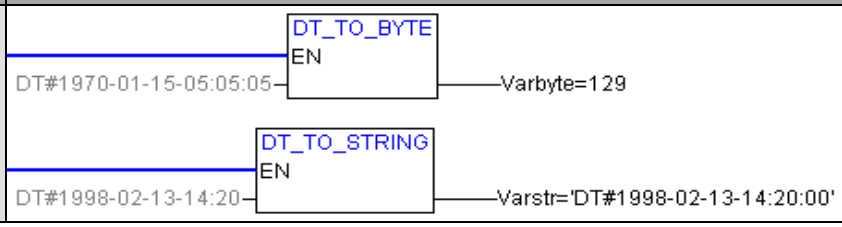
- 功能：把日期时间型数据转换为其它类型数据，日期在内部以秒为单位存储，时间从 1970 年 1 月 1 日开始。
- 输入/输出数据类型（参见表 4-7-1）：
- 当 DT_TO_BOOL 时，输入不等于 0 时输出为 TRUE，当输入等于 0 时输出为 FALSE。

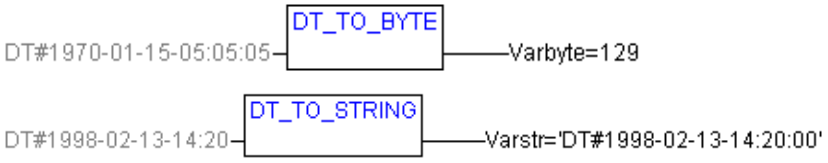
指令使用举例

变量定义

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CONS

	名称	地址	类型	初始值	注释
0001	Varbyte		BYTE		
0002	VarStr		STRING		

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’)
指令列表（IL）	LD DT#1970-01-15-05:05:05 DT_TO_BYTE ST Varbyte (*结果 Varbyte 为 129*) LD DT#1998-02-13-14:20 DT_TO_STRING ST Varstr (*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。

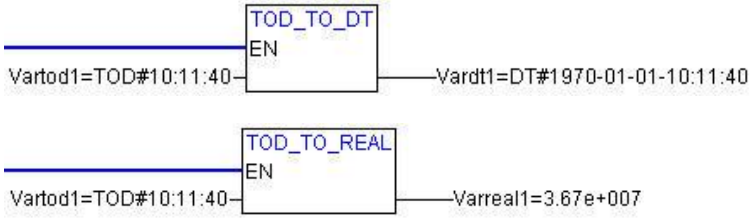
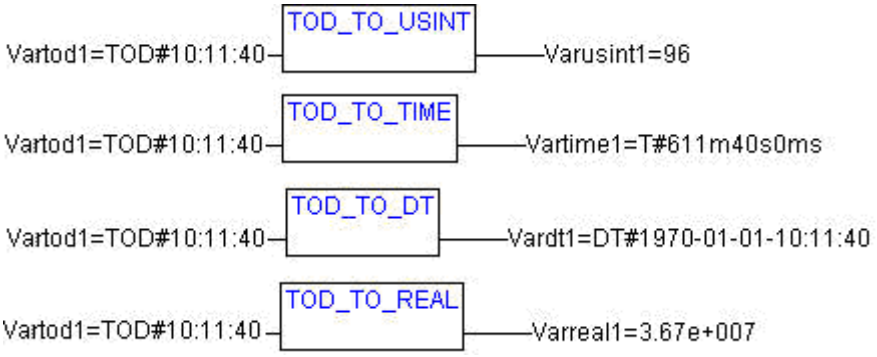
指令使用举例

变量定义

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CONSTANT

	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*)
指令列表（IL）	LD TOD#10:11:40 ST Vartod1 (*Vartod1 取值*) LD Vartod1 TOD_TO_USINT ST Varusint1 (*结果为 96*) LD Vartod1 TOD_TO_TIME ST Vartime1 (*结果为 T#611m40s0ms*) LD Vartod1 TOD_TO_DT ST Vardt1 (*结果为 DT#1970-01-01-10:11:40*) LD Vartod1 TOD_TO_REAL ST Varreal1 (*结果为 3.67e+007*)
功能块（FBD）	

2.7.16 STRING_TO_<TYPE>——字符类型转换指令

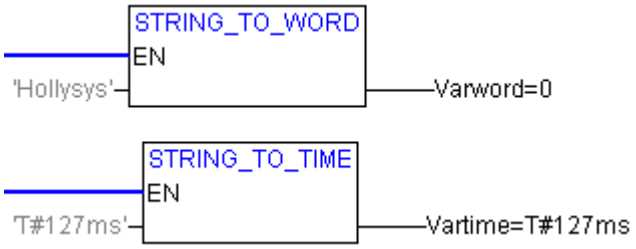
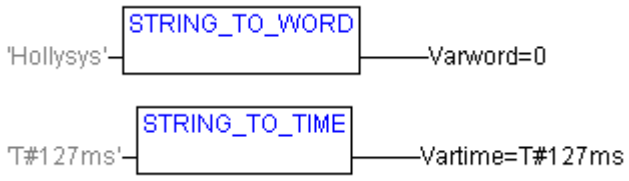
- 功能：把字符串转换为其它类型数据，字符串型变量必须包含一个有效的目标变量值，否则转换结果为 0。
- 输入/输出数据类型：（参见表 4-7-1）

指令使用举例

变量定义

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CON

	名称	地址	类型	初始值	注释
0001	Varword		WORD		
0002	Vartime		TIME		

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*)
指令列表（IL）	LD　'Hollysys' STRING_TO_WORD ST　Varword　(*结果 Varword 为 0*) LD　'T#127ms' STRING_TO_TIME ST　Vartime　(*结果 Vartime 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。

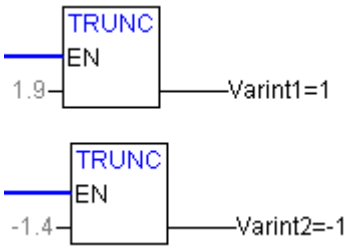
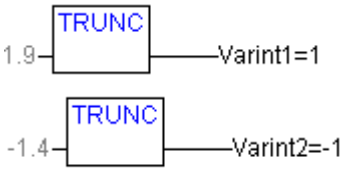
指令使用举例

变量定义

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CON

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

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*)
指令列表（IL）	LD　　1.9 TRUNC ST　　Varint1　　(*结果 Varint1 为 1*) LD　　-1.4 TRUNC ST　　Varint2　　(*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

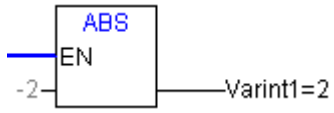
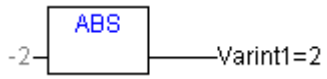
2.8.1 ABS——绝对值指令

- 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

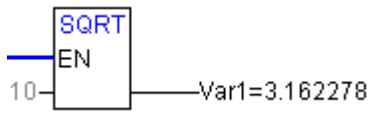
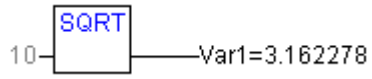
编程语言	程　序
梯形图（LD）	
结构化文本（ST）	i:=ABS(-2);　　(*结果 Varint1 为 2*)
指令列表（IL）	LD　-2 ABS ST　Varint1　　(*结果 Varint1 为 2*)
功能块（FBD）	

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

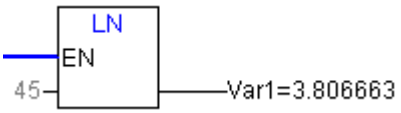
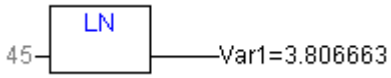
变量定义

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*)
指令列表（IL）	LD　10 SQRT ST　Var1　　(*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。

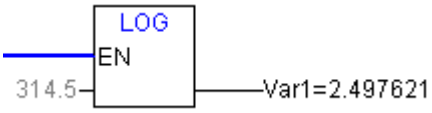
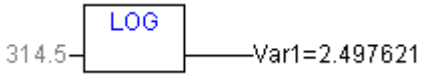
指令使用举例

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*)
指令列表（IL）	LD　45 LN ST　Var1　(*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

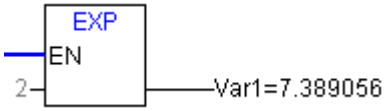
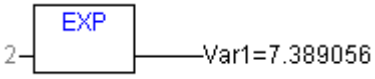
变量定义	
	
编程语言	**程　序**
梯形图（LD）	
结构化文本（ST）	Var1:=LOG(314.5);　(*结果 Var1 为 2.497621 *)
指令列表（IL）	LD　314.5 LOG ST　Var1　(*结果 Var1 为 2.497621 *)
功能块（FBD）	

2.8.5　EXP——指数指令

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

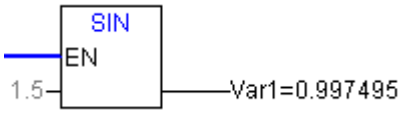
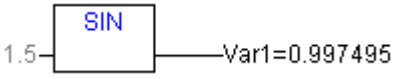
指令使用举例

变量定义	
	
编程语言	**程　序**
梯形图（LD）	
结构化文本（ST）	Var1:=EXP(2);　　（*结果 Var1 为 7.389056*）
指令列表（IL）	LD　2 EXP ST　Var1　　（*结果 Var1 为 7.389056*）
功能块（FBD）	

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

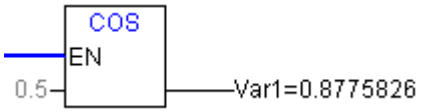
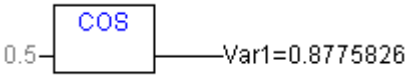
变量定义	
	
编程语言	**程　序**
梯形图（LD）	
结构化文本（ST）	Var1:=SIN(1.5);　　(*结果 Var1 为 0.997495 *)
指令列表（IL）	LD　1.5 SIN ST　Var1　　(*结果 Var1 为 0.997495 *)
功能块（FBD）	

2.8.7　COS——余弦指令

- 功能：求输入数据的余弦值，输入数据以弧度表示。$\text{弧度(rad)} = \text{角度} * \frac{\pi}{180^o}$
- 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。

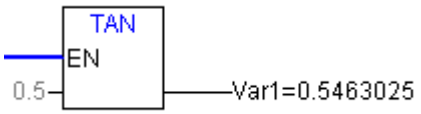
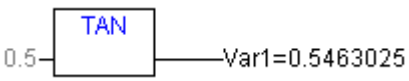
指令使用举例

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 *)
指令列表（IL）	LD　0.5 COS ST　Var1　　(*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，输入数据以弧度表示。$\text{弧度(rad)} = \text{角度} * \frac{\pi}{180^{o}}$
- 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

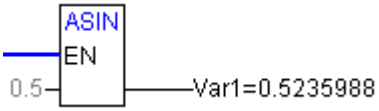
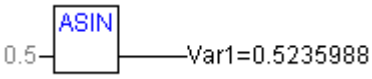
变量定义	
	
编程语言	**程　序**
梯形图（LD）	
结构化文本（ST）	Var1:=TAN(0.5);　　(*结果 Var1 为 0.5463025 *)
指令列表（IL）	LD　0.5 TAN ST　Var1　　(*结果 Var1 为 0.5463025 *)
功能块（FBD）	

2.8.9　ASIN——反正弦指令

- 功能：求输入数据的反正弦值。
- 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。

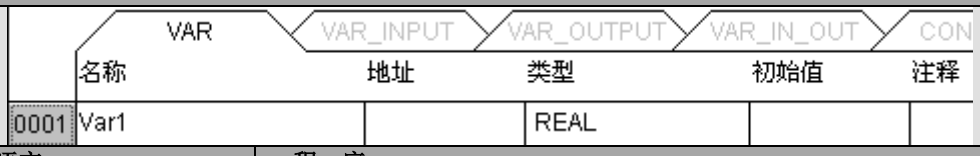
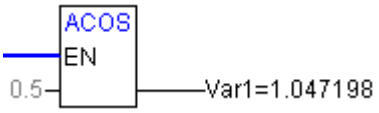
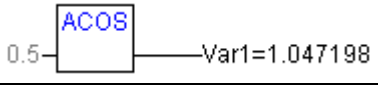
指令使用举例

变量定义	
	
编程语言	**程　序**
梯形图（LD）	
结构化文本（ST）	Var1:=ASIN(0.5);　(*结果 Var1 为 0.5235988 *)
指令列表（IL）	LD　0.5 ASIN ST　Var1　(*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/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

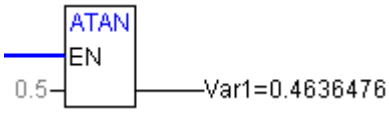
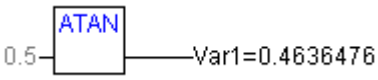
变量定义	
	
编程语言	**程　序**
梯形图（LD）	
结构化文本（ST）	Var1:=ACOS(0.5);　(*结果 Var1 为 1.047198 *)
指令列表（IL）	LD　0.5 ACOS ST　Var1　(*结果 Var1 为 1.047198 *)
功能块（FBD）	

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。

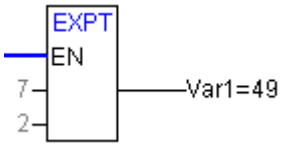
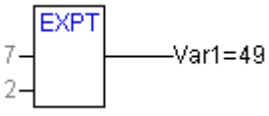
指令使用举例

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 *)
指令列表（IL）	LD　0.5 ATAN ST　Var1　(*结果 Var1 为 0.4636476 *)
功能块（FBD）	

2.8.12　EXPT——幂指令

- 功能：对输入数据求幂，输入数据 1 为幂底数，输入数据 2 为幂指数。
- 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 *)
指令列表（IL）	LD　7 EXPT　2 ST　Var1　(*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

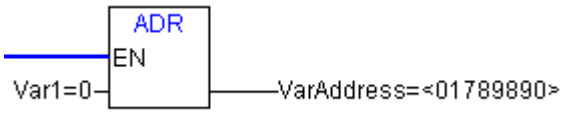
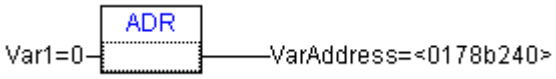
- 功能：取得输入变量的内存地址，并输出。该地址可以在程序内当作指针使用，也可以作为指针传送给函数。

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);
指令列表（IL）	LD　Var1 ADR ST　VarAddress
功能块（FBD）	

2.9.2 ^——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。

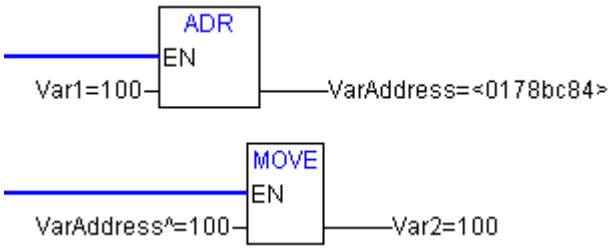
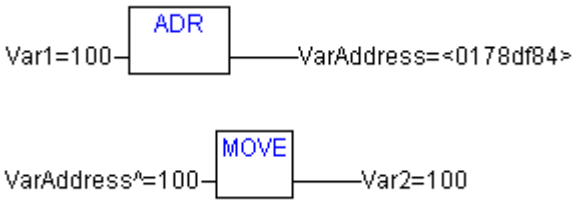
指令使用举例

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 *)
指令列表（IL）	LD Var1 ADR ST VarAddress LD VarAddress^ ST Var2
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

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	Varbool1	%MX300.7	BOOL		
0002	Bitadr1		DWORD		

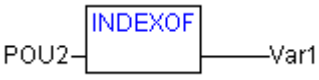
编程语言	**程　序**
梯形图（LD）	
结构化文本（ST）	Bitadr1:=BITADR(Varbool1)；
指令列表（IL）	LD Varbool1 BITADR ST Bitadr1
功能块（FBD）	

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> 

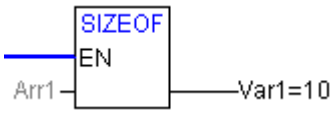
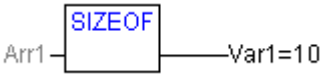
（POU2 可以是程序、功能块、函数）</td></tr>
<tr><td>结构化文本（ST）</td><td>Var1:=INDEXOF(POU2);　（*结果 Var1 为 38*）</td></tr>
<tr><td>指令列表（IL）</td><td>LD　　POU2
INDEXOF
ST　　Var1　　（*结果 Var1 为 38*）</td></tr>
<tr><td>功能块（FBD）</td><td> 

（*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 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:=SIZEOF(arr1);　（*结果 Var1 为 10*）</td></tr>
<tr><td>指令列表（IL）</td><td>LD　　arr1
SIZEOF
ST　　Var1　　（*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

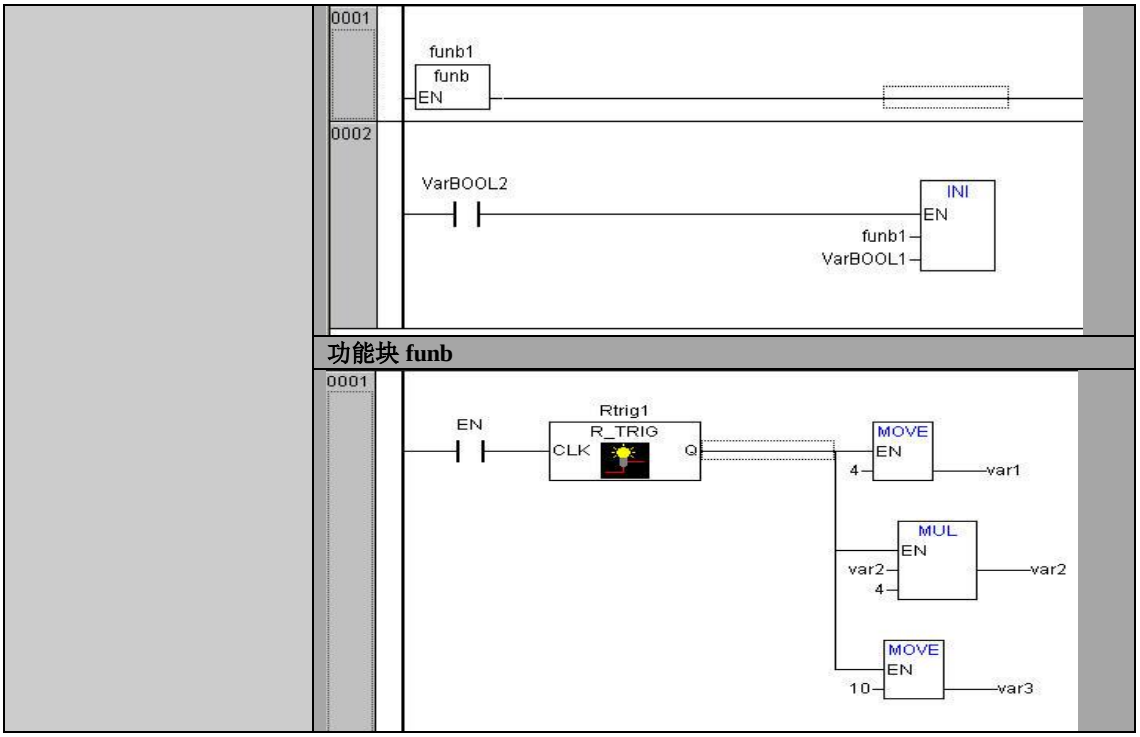
	名称	地址	类型	初始值	注释
0001	VarBOOL2		BOOL		
0002	VarBOOL1		BOOL		
0003	funb1		funb		

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



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。

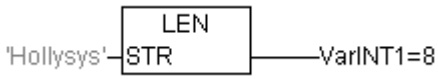
2.12 字符串处理指令（Standard.lib）

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

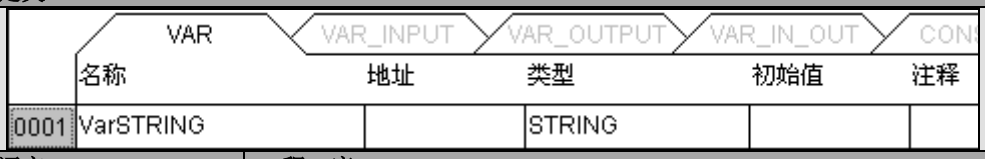
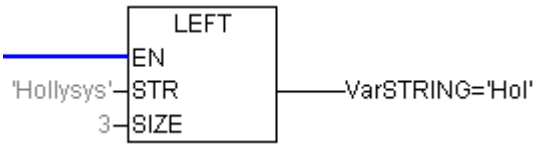
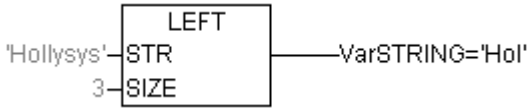
<table>
<tr><td colspan="2">变量定义</td></tr>
<tr><td colspan="2">VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CON
名称 | 地址 | 类型 | 初始值 | 注释
0001 | VarINT1 | | INT | | </td></tr>
<tr><td>编程语言</td><td>程　序</td></tr>
<tr><td>梯形图（LD）</td><td>LEN
EN
'Hollysys'—STR
VarINT1=8</td></tr>
<tr><td>结构化文本（ST）</td><td>Var1INT1 := LEN (‘Hollysys’); (*结果 Var1INT1 为 8*)</td></tr>
<tr><td>指令列表（IL）</td><td>LD　‘Hollysys’
LEN
ST　VarINT1　(*结果 Var1INT1 为 8*)</td></tr>
</table>

功能块（FBD）	

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。

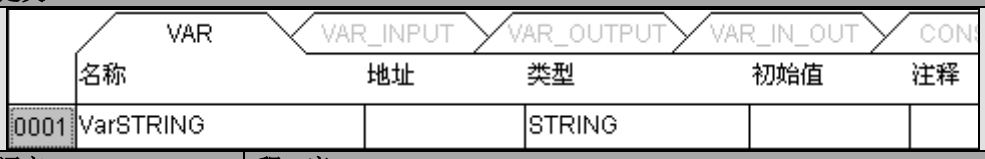
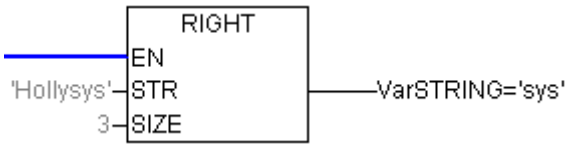
指令使用举例

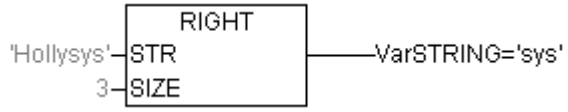
变量定义	
	
编程语言	**程　序**
梯形图（LD）	
结构化文本（ST）	VarSTRING := LEFT (‘Hollysys’,3); (*结果为' Hol '*)
指令列表（IL）	LD　　' Hollysys ' LEFT　 3 ST　　VarSTRING　　(*结果为' Hol '*)
功能块（FBD）	

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

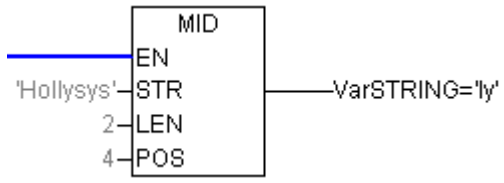
变量定义	
	
编程语言	**程　序**
梯形图（LD）	
结构化文本（ST）	VarSTRING := RIGHT(‘Hollysys’,3);　(*结果为' sys '*)

指令列表（IL）	LD ' Hollysys ' RIGHT 3 ST VarSTRING (*结果为' sys '*)
功能块（FBD）	

2.12.4 MID——中间取字符串指令

- 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

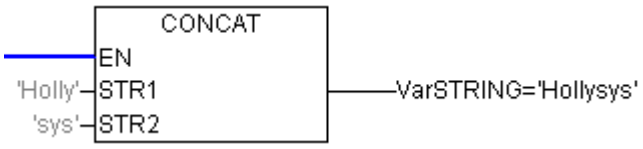
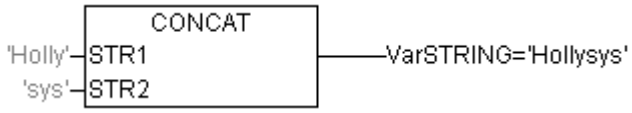
变量定义	
	
编程语言	**程 序**
梯形图（LD）	
结构化文本（ST）	VarSTRING:= MID('Hollysys',2,4); (*结果为'ly' *)
指令列表（IL）	LD 'Hollysys' MID 2,4 ST VarSTRING (*结果为'ly' *)
功能块（FBD）	

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**

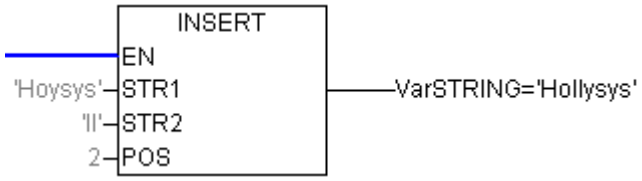
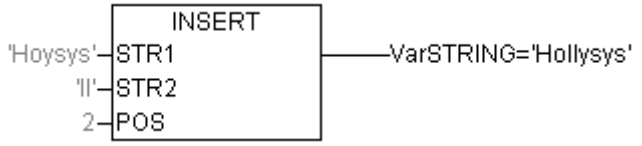
梯形图（LD）	
结构化文本（ST）	VarSTRING := CONCAT ('Holly', 'sys'); (*结果为'Hollysys'*)
指令列表（IL）	LD 'Holly' CONCAT ‘sys’ ST VarSTRING (*结果为'Hollysys'*)
功能块（FBD）	

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

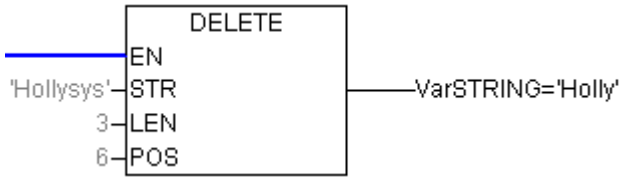
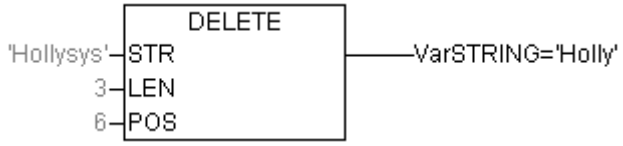
变量定义					
	VAR	VAR_INPUT	VAR_OUTPUT	VAR_IN_OUT	CONS
	名称	地址	类型	初始值	注释
0001	VarSTRING		STRING		

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'*)
指令列表（IL）	LD 'Hoysys' INSERT ‘ll’,2 ST VarSTRING (*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。

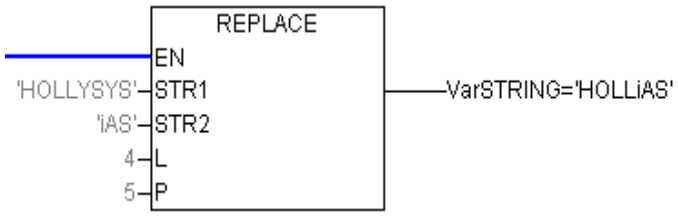
指令使用举例

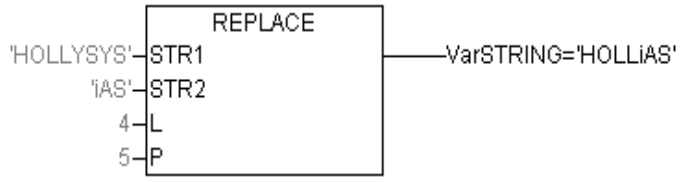
变量定义	
	
编程语言	**程　序**
梯形图（LD）	
结构化文本（ST）	VarSTRING := DELETE ('Hollysys',3,6);　(*结果为'Holly'*)
指令列表（IL）	LD　　'Hollysys' DELETE　3,6 ST　　VarSTRING　　(*结果为'Holly'*)
功能块（FBD）	

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

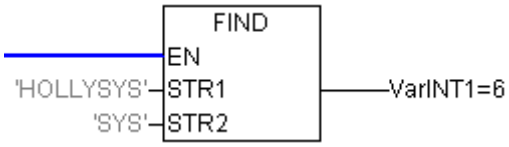
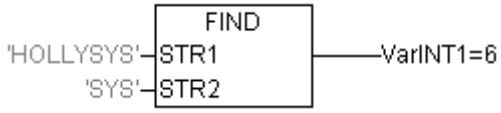
变量定义	
	
编程语言	**程　序**
梯形图（LD）	
结构化文本（ST）	VarSTRING:= REPLACE ('HOLLYSYS', 'iAS',4,5); (*结果为'HOLLiAS'*)
指令列表（IL）	LD　　'HOLLYSYS' REPLACE　'iAS', 4,5 ST　　VarSTRING　　(*结果为'HOLLiAS'*)

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*)
指令列表（IL）	LD　　'HOLLYSYS' FIND　　'SYS' ST　　VarINT1　　(*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

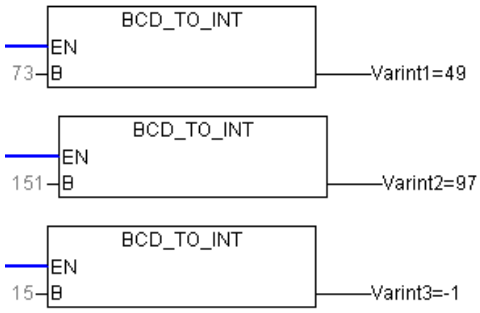
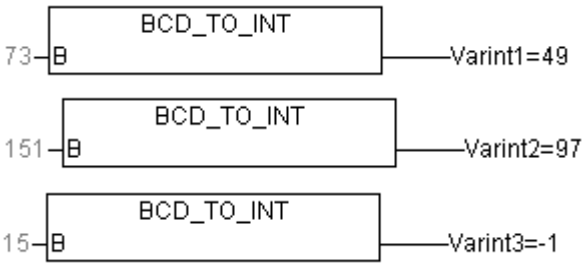
- 功能：该指令将 BCD 码转为 INT 值。
- 输入/输出数据类型：
- 输入 B：BYTE 型，输入 BCD 码的二进制形式（或者该二进制对应的十进制和十六进制）。比如 BCD 码 49，表示为 2#100 1001（或者对应 10#73、16#49），则此处输入 2#100 1001（或者 10#73，16#49）；
- 输出：INT 型，该 BCD 码所代表的实际值，如果输入的字节不是 BCD 码，输出是-1。

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*)
指令列表（IL）	LD　73 BCD_TO_INT ST　Varint1　(*结果为 49 *)
	LD　151 BCD_TO_INT ST　Varint2　(*结果为 97*)
	LD　15 BCD_TO_INT 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/输出数据类型：
- 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

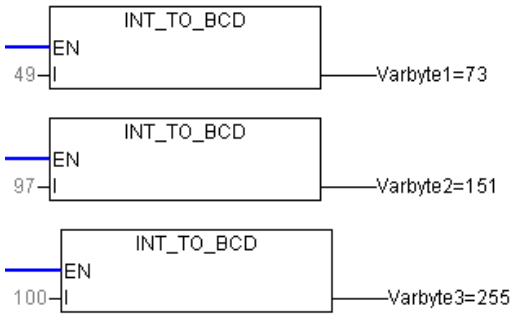
变量定义

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*)
```

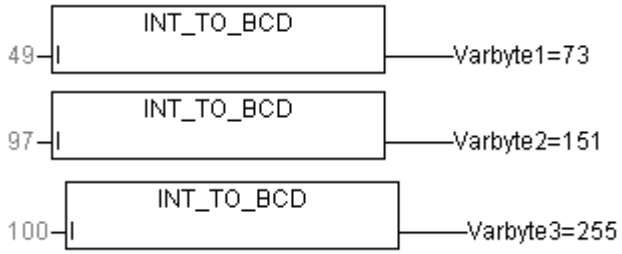
指令列表（IL）

```
LD      49
INT_TO_BCD
ST      Varbyte1                   (*结果为 73*)
```

```
LD      97
INT_TO_BCD
ST      Varbyte2                   (*结果为 151*)
```

```
LD      100
INT_TO_BCD
ST      Varbyte3                   (*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

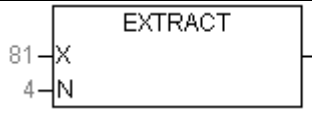
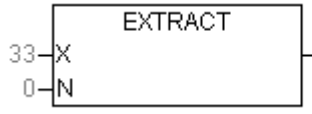
- 功能：提取输入变量 X 二进制数的第 N 位（N=0,1...）并输出该位数值。
- 输入/输出数据类型：
- 输入变量 X 是 DWORD 类型，N 是 BYTE 型；输出变量是 BOOL 类型。

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*)
指令列表（IL）	LD　81 EXTRACT　4 ST　FLAG1 (*结果：TRUE，因为 81 的二进制数是 1010001，所以第四位是 1*)
	LD　33 EXTRACT　0 ST　FLAG2 (*结果：TRUE，因为 33 的二进制数是 100001，所以第 0 位是 1*)
梯形图（LD）	 (*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。

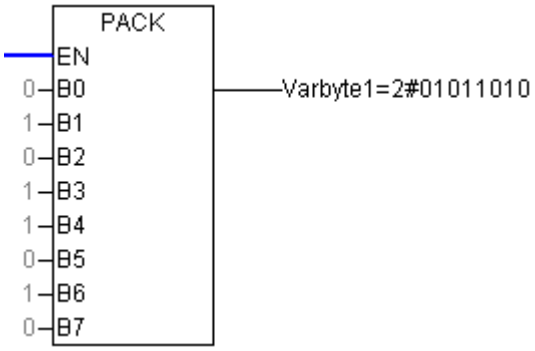
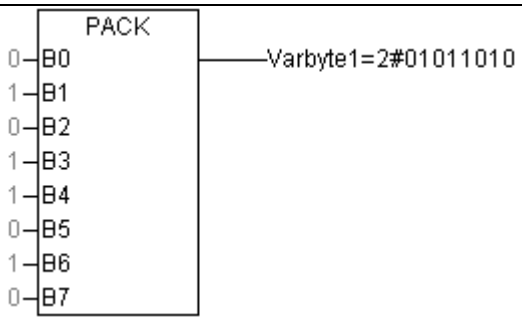
- 输入/输出数据类型：
- 输入 B0、B1、……、B7 均为 BOOL 类型；输出数据为 BYTE 型。

指令使用举例

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		

编程语言	程　序
梯形图（LD）	
结构化文本（ST）	Varbyte1:= PACK(0,1,0,1,1,0,1,0); (*结果为 2#01011010*)
指令列表（IL）	LD FALSE PACK TRUE,FALSE,TRUE,TRUE,FALSE,TRUE,FALSE ST Varbyte1 (*结果为 2#01011010*)
功能块（FBD）	

2.14.3 PUTBIT——位赋值指令

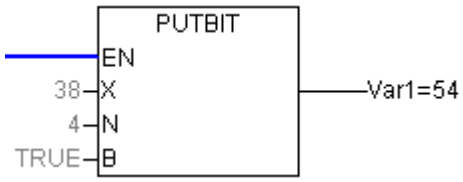
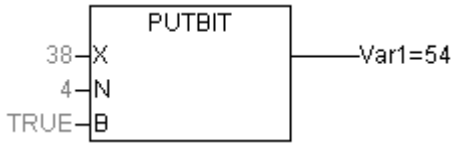
- 功能：将输入变量 X 值的第 N 位（N=0,1...）赋值为 B，并输出 X 转变后的值。
- 输入/输出数据类型：
- 输入变量 X 为 DWORD 型、N 为 BYTE 型和 B 为 BOOL 型；输出为 DWORD 型。

指令使用举例

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		DWORD		
0002	Var2		DWORD		

编程语言	程　序
梯形图（LD）	
结构化文本（ST）	Var2:=38;　　(*二进制 100110*) Var1:=PUTBIT(Var2,4,TRUE);　(*结果：54 = 2#110110*)
指令列表（IL）	LD　38　　(*二进制 100110*) PUTBIT　4,TRUE ST　Var1　　(*结果：54 = 2#110110*)
功能块（FBD）	

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。

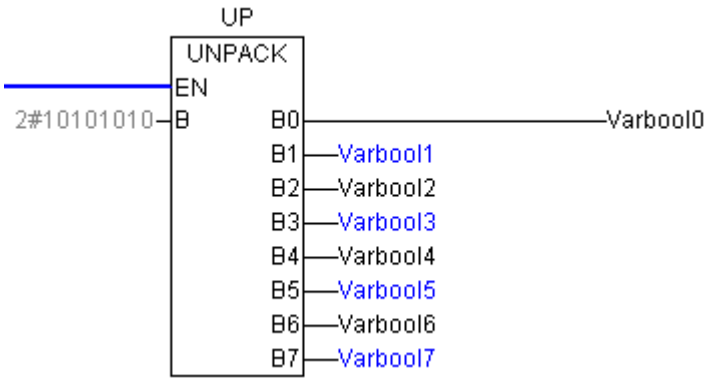
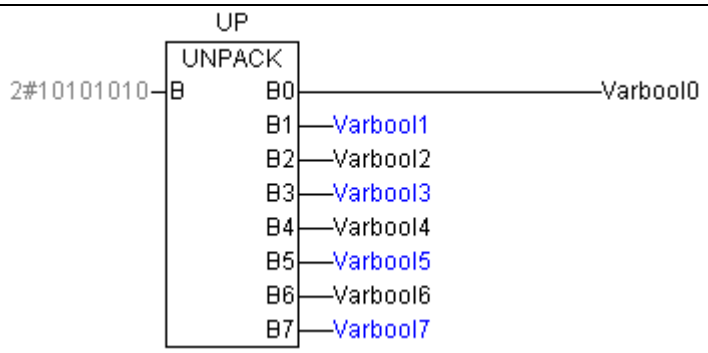
指令使用举例

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;
指令列表（IL）	CAL UP(B := 2#10101010) LD UP.B1 ST Varbool1 LD UP.B2 ST Varbool2 LD UP.B3 ST Varbool3 LD UP.B4 ST Varbool4 LD UP.B5 ST Varbool5 LD UP.B6 ST Varbool6 LD UP.B7 ST Varbool7 LD UP.B0 ST Varbool0
功能块（FBD）	

2.15 高等数学运算指令（Util.lib）

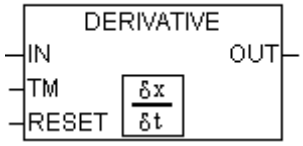
2.15.1 DERIVATIVE——微分

- 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。
- 微分迭代公式：

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

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

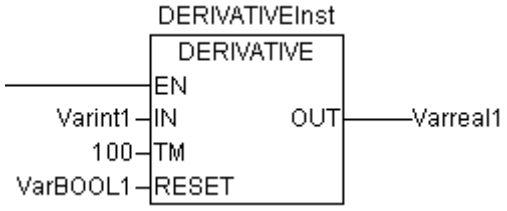
输入参数	数据类型	功能描述	参数值说明
IN	REAL	连续输入的变量	
TM	DWORD	微分时间	毫秒
RESET	BOOL	复位信号	值是 TRUE 时，重新启动指令
输出参数	**数据类型**	**功能描述**	**参数值说明**
OUT	REAL	微分结果输出	

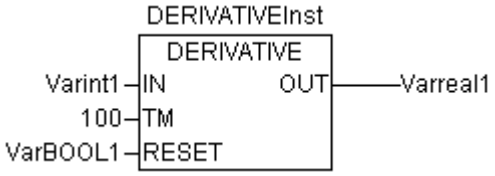
指令使用举例

变量定义

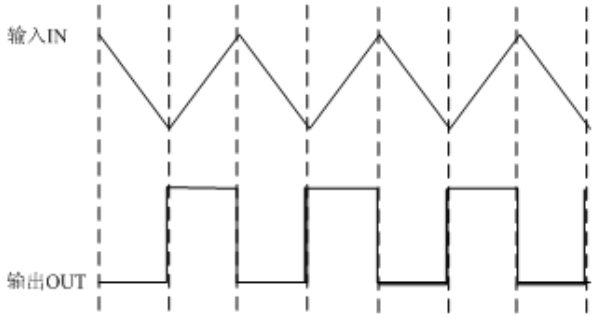
VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CONSTANT

	名称	地址	类型	初始值	注释
0001	DERIVATIVEInst		DERIVATIVE		
0002	Varreal1		REAL		
0003	Varint1		INT		
0004	VarBOOL1		BOOL		

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;

指令列表（IL）	CAL DERIVATIVEInst(IN := Varint1, TM := 100, RESET := VarBOOL1) LD DERIVATIVEInst.OUT ST Varreal1
功能块（FBD）	

输入输出对应关系如下图所示。



2.15.2 INTEGRAL——积分

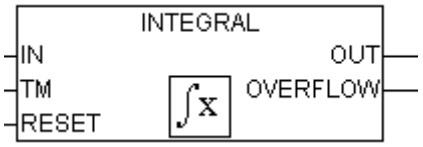
- 功能：该指令对连续输入的变量进行积分运算。
- 积分迭代公式： $A(k) = A(k-1) + TM * IN(k-1)$

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

$$OUT(k) = \frac{A(k) + B(k)}{2}$$

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

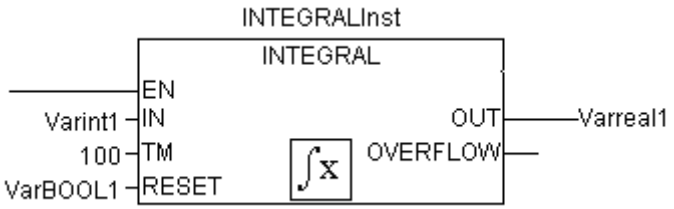
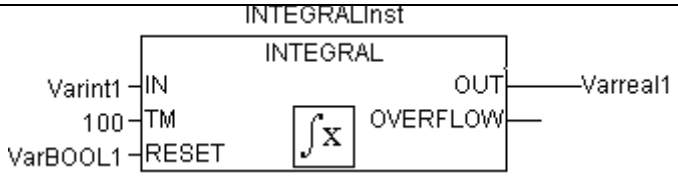
输入参数	**数据类型**	**功能描述**	**参数值说明**
IN	REAL	连续输入的变量	
TM	DWORD	积分时间	
RESET	BOOL	复位信号	其值是 TRUE 时，重新启动指令
输出参数	**数据类型**	**功能描述**	**参数值说明**
OUT	REAL	积分结果输出	
OVERFLOW	BOOL	溢出标志	其值是 TRUE 时，说明计算溢出

指令使用举例

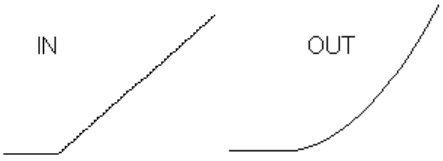
变量定义

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CONSTANT

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

编程语言	程　序
梯形图（LD）	
结构化文本（ST）	INTEGRALInst(IN := Varint1, TM := 100, RESET := VarBOOL1); Varreal1:=INTEGRALInst.OUT;
指令列表（IL）	CAL INTEGRALInst(IN := Varint1, TM := 100, RESET := VarBOOL1) LD　　INTEGRALInst.OUT ST　　Varreal1
功能块（FBD）	

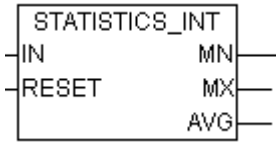
输入输出对应关系如下图所示。



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

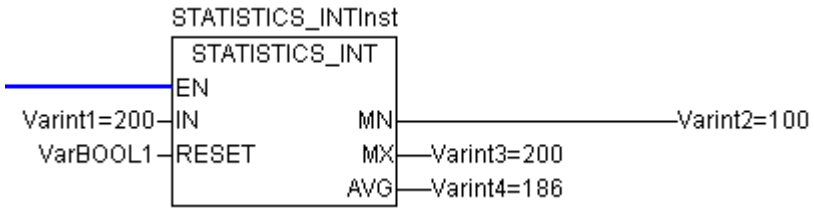
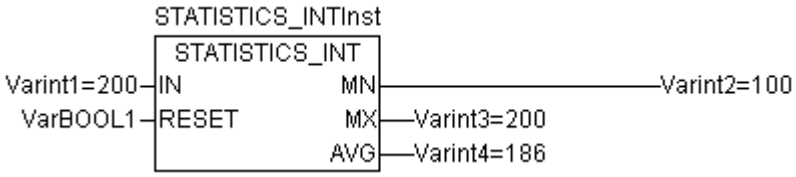
输入参数	数据类型	功能描述	参数值说明
IN	INT	输入	
RESET	BOOL	初始化	其值是 TRUE 时，重新初始化
输出参数	**数据类型**	**功能描述**	**参数值说明**
MN	INT	最小值	
MX	INT	最大值	
AVG	INT	平均值	

指令使用举例

变量定义

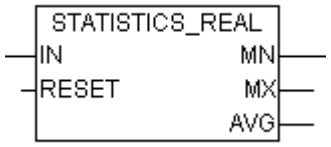
VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CONSTANT

	名称	地址	类型	初始值	注释
0001	STATISTICS_INTInst		STATISTICS_INT		
0002	VarBOOL1		BOOL		
0003	Varint1		INT		
0004	Varint2		INT		
0005	Varint3		INT		
0006	Varint4		INT		

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;
指令列表（IL）	CAL STATISTICS_INTInst(IN := Varint1, RESET := VarBOOL1) LD　　STATISTICS_INTInst.MX ST　　Varint3 LD　　STATISTICS_INTInst.AVG ST　　Varint4 LD　　STATISTICS_INTInst.MN ST　　Varint2
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

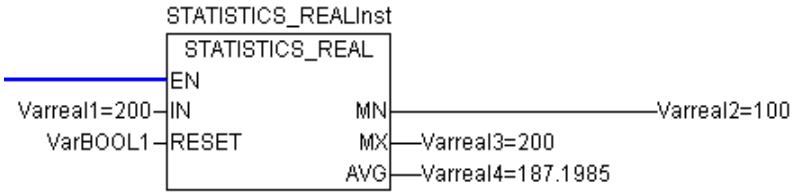
输入参数	数据类型	功能描述	参数值说明
IN	REAL	输入	
RESET	BOOL	初始化	其值是 TRUE 时，重新初始化
输出参数	**数据类型**	**功能描述**	**参数值说明**
MN	REAL	最小值	
MX	REAL	最大值	
AVG	REAL	平均值	

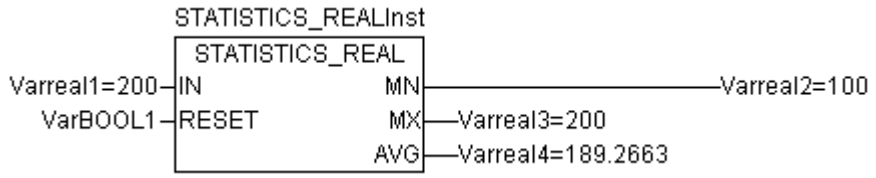
指令使用举例

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;
指令列表（IL）	CAL STATISTICS_REALInst(IN := Varreal1, RESET := VarBOOL1) LD　　STATISTICS_REALInst.MX ST　　Varreal3 LD　　STATISTICS_REALInst.AVG ST　　Varreal4 LD　　STATISTICS_REALInst.MN ST　　Varreal2

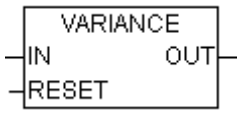
功 能 块（FBD）	

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

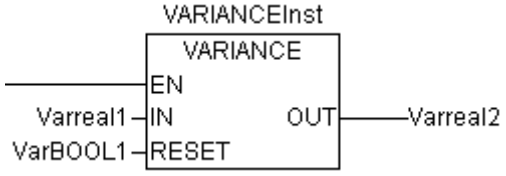
输入参数	数据类型	功能描述	参数值说明
IN	REAL	输入	
RESET	BOOL	复位	其值是 TRUE 时，指令复位
输出参数	**数据类型**	**功能描述**	**参数值说明**
OUT	REAL	平方偏差	

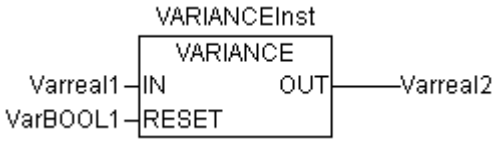
指令使用举例

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		

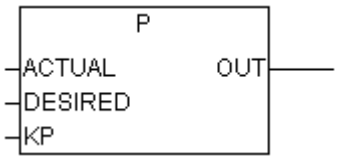
编程语言	程 序
梯形图（LD）	
结 构 化 文 本（ST）	VARIANCEInst(IN := Varreal1, RESET := VarBOOL1); Varreal2:=VARIANCEInst.OUT;

指令列表（IL）	CAL VARIANCEInst(IN := Varreal1, RESET := VarBOOL1) LD VARIANCEInst.OUT ST Varreal2
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	

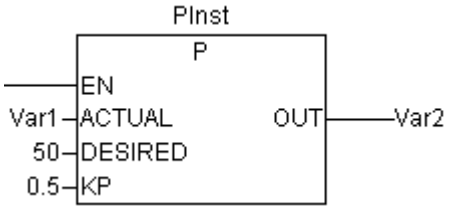
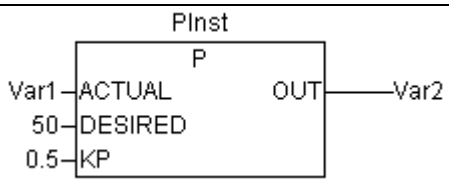
指令使用举例

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**

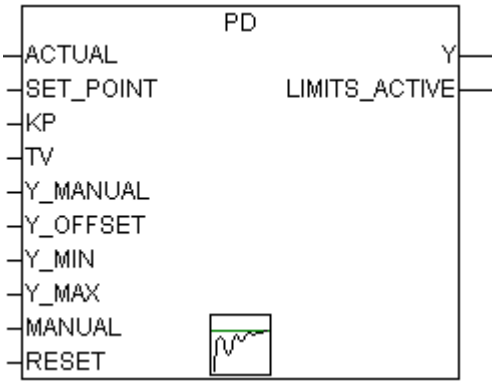
梯形图（LD）	
结构化文本（ST）	PInst(ACTUAL := Var1, DESIRED := 50, KP := 0.5); Var2:=PInst.OUT;
指令列表（IL）	CAL PInst(ACTUAL := Var1, DESIRED := 50, KP := 0.5) LD PInst.OUT ST Var2
功能块（FBD）	

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：$\Delta = \mathrm{SET\_POINT} - ACTUAL$
- $Y = KP * (\Delta + TV * \frac{\sigma\Delta}{\sigma t}) + Y\_OFFSET$

  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

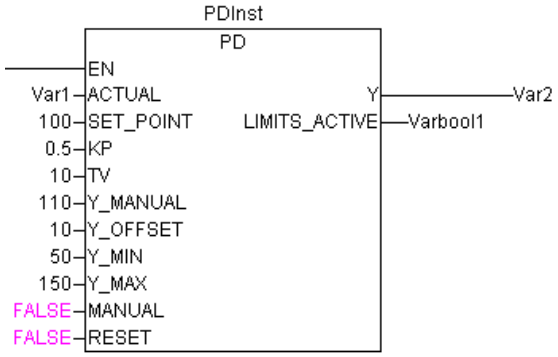
变量定义

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）

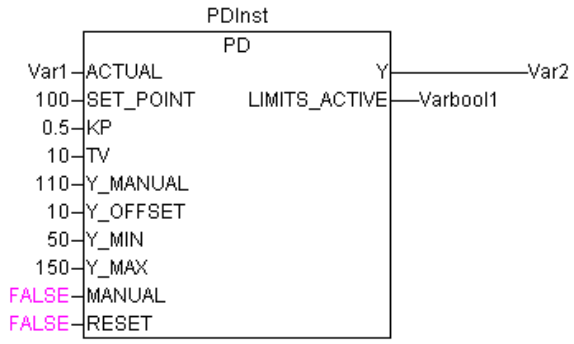


结构化文本（ST）

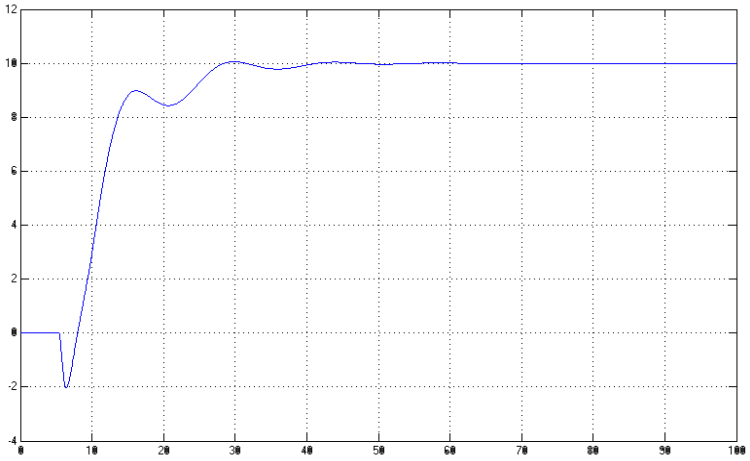
```
PDInst(ACTUAL := Var1, SET_POINT := 100, KP := 0.5, TV := 10,
Y_MANUAL := 110, Y_OFFSET := 10, Y_MIN := 50,
Y_MAX := 150, MANUAL := FALSE, RESET := FALSE);
Varbool1:=PDInst.LIMITS_ACTIVE;
Var2:=PDInst.Y;
```

指令列表（IL）

```
CAL 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)
LD      PDInst.LIMITS_ACTIVE
ST      Varbool1
LD      PDInst.Y
ST      Var2
```

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

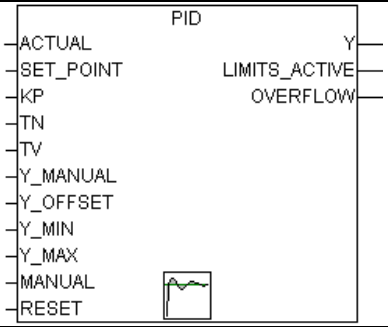
- 功能：该指令为比例积分微分控制器。当 TV=0 时，PID 控制器作为 PI 控制器。
- 控制方程：

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

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

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

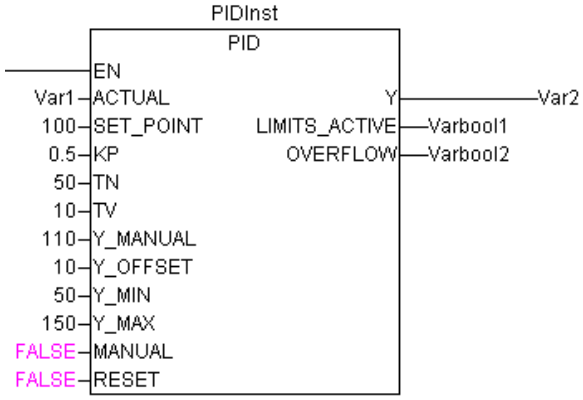
变量定义

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		

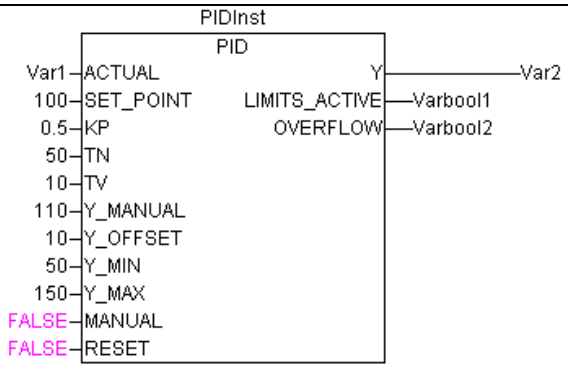
编程语言 | **程　序**

梯形图（LD）

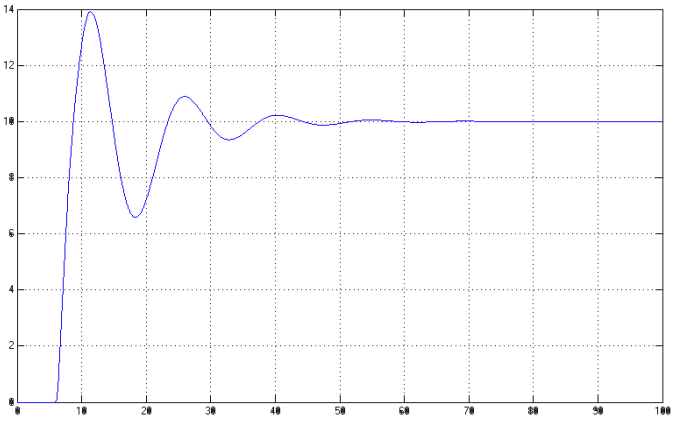


结构化文本（ST）

```
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;
```

指令列表（IL）	CAL 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) LD PIDInst.LIMITS_ACTIVE ST Varbool1 LD PIDInst.OVERFLOW ST Varbool2 LD PIDInst.Y ST Var2
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

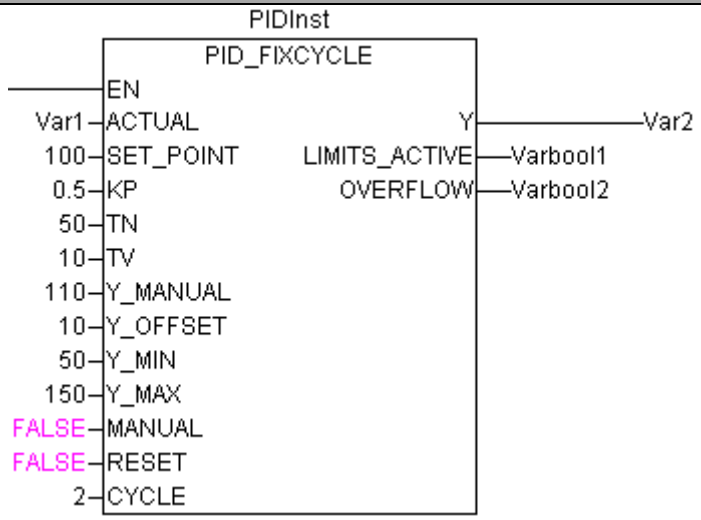
变量定义

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）

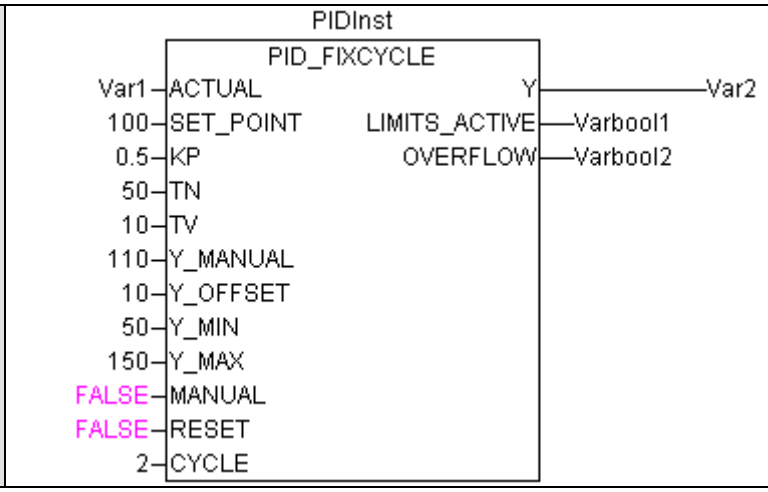


结构化文本（ST）

```
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;
```

指令列表（IL）

```
CAL 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)
LD      PIDInst.LIMITS_ACTIVE
ST      Varbool1
LD      PIDInst.OVERFLOW
ST      Varbool2
LD      PIDInst.Y
ST      Var2
```

功能块（FBD）	

调整波形如前面 PID 指令所示。

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

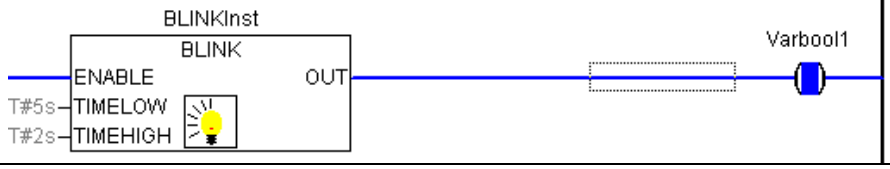
输入参数	数据类型	功能描述	参数值说明
ENABLE	BOOL	使能	TRUE 时，指令开始工作
TIMELOW	TIME	输出低电平时间	
TIMEHIGH	TIME	输出高电平时间	
输出参数	**数据类型**	**功能描述**	**参数值说明**
OUT	BOOL	脉冲信号输出值	

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;

指令列表（IL）	CAL BLINKInst(TIMELOW := T#5s,TIMEHIGH := T#2s) LD BLINKInst.OUT ST Varbool1
功能块（FBD）	

当指令执行时，OUT 如图 4-20-1 所示波形输出。

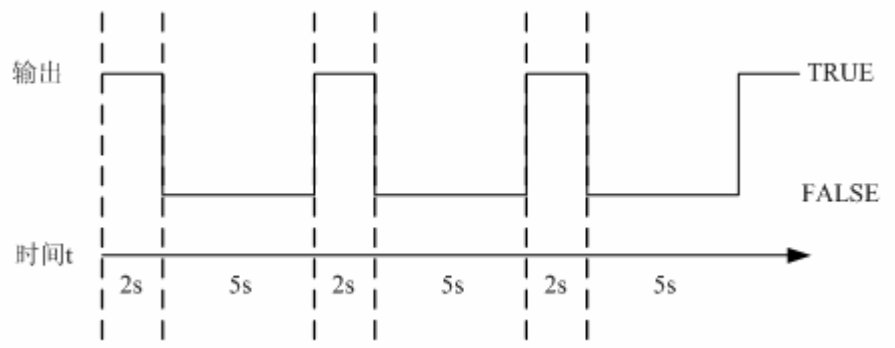


图 4-20-1

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。
- 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

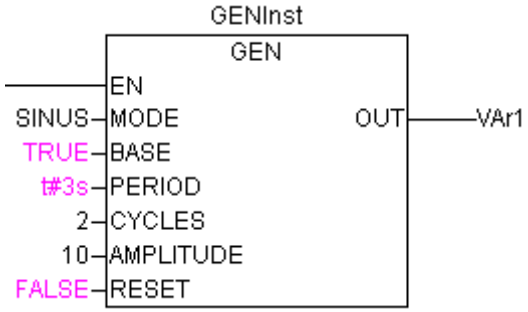
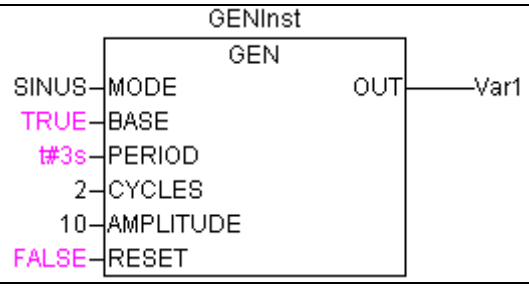
			时，信号发生器与特定的发生的个数有关
PERIOD	TIME	循环周期	
CYCLES	INT	发生的个数	
AMPLITUDE	INT	信号的振幅	
RESET	BOOL	初始化	当 RESET=TRUE 时，信号发生器被重新设置为 0
输出参数	**数据类型**	**功能描述**	**参数值说明**
OUT	INT	波形信号输出值	

指令使用举例

变量定义

VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | CONSTANT

	名称	地址	类型	初始值	注释
0001	GENInst		GEN		
0002	Var1		INT		

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;
指令列表（IL）	CAL　GENInst(MODE:=SINUS, BASE:=TRUE, PERIOD:=T#3s, CYCLES:=2, AMPLITUDE:=10, RESET:=FALSE) LD　　GENInst.OUT ST　　Var1
功能块（FBD）	

根据 MODE 处输入不同，产生的波形如图 4-20-2 所示。

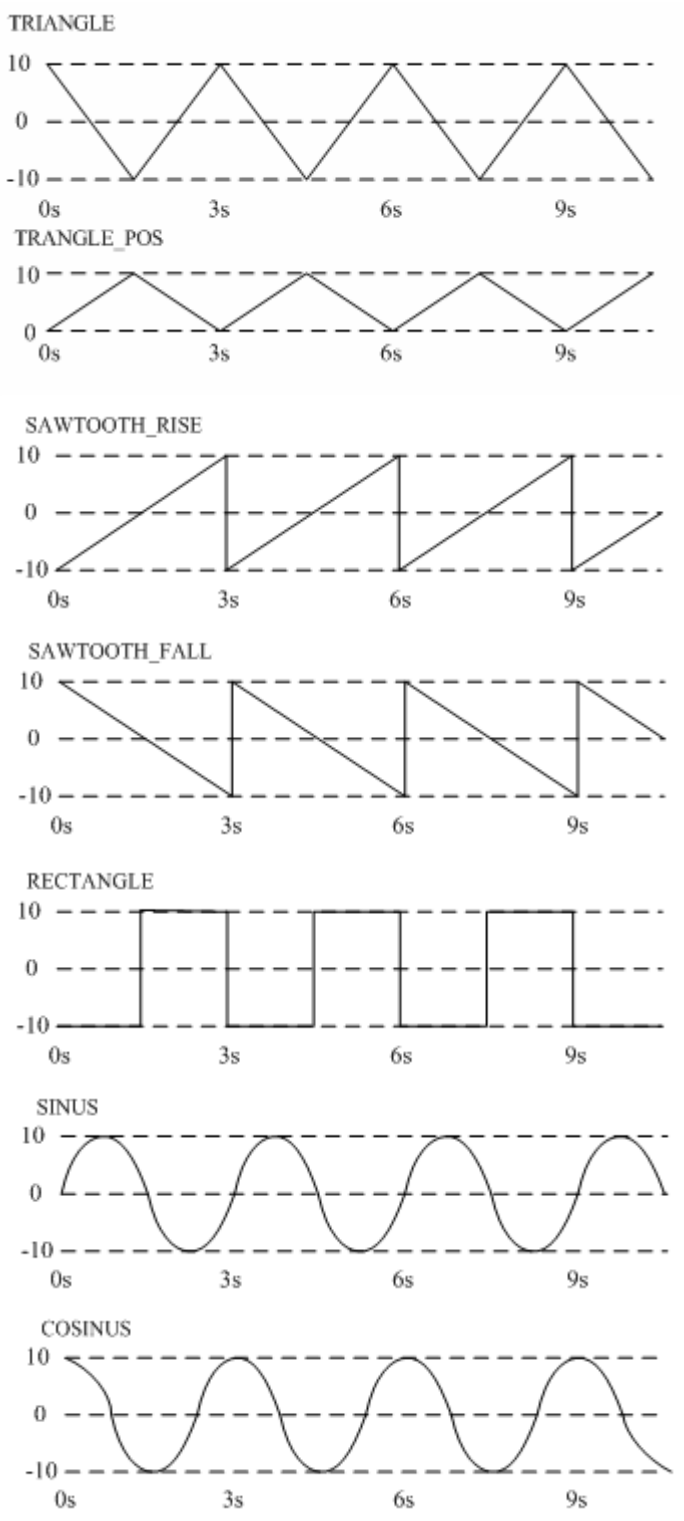

TRIANGLE
10
0
-10
0s
3s
6s
9s
TRANGLE_POS
10
0
0s
3s
6s
9s
SAWTOOTH_RISE
10
0
-10
0s
3s
6s
9s
SAWTOOTH_FALL
10
0
-10
0s
3s
6s
9s
RECTANGLE
10
0
-10
0s
3s
6s
9s
SINUS
10
0
-10
0s
3s
6s
9s
COSINUS
10
0
-10
0s
3s
6s
9s

图 4-20-2

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≤N≤11）
P	ARRAY[0..10] OF POINT		用来在 XY 坐标上定义曲线

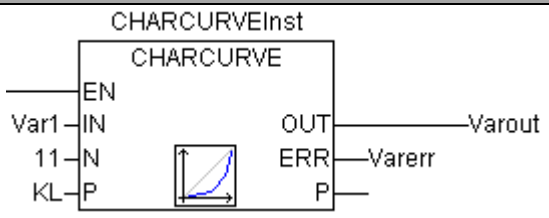
输出参数	数据类型	功能描述	参数值说明
OUT	INT	输出值	坐标图上曲线对应的 Y 轴的值
ERR	BYTE	显示错误类型	ERR=1：数组中的点 P[0]..P[N-1]中的 X 值有错误。 ERR=2：输入值 IN 不在 P[0].X 和 P[N-1].X 之间，即超出了数组定义曲线的 X 轴的范围。此时 OUT 输出 IN 包含在限制值 P[0].Y 和 P[N-1].Y 之间所对应的数据。 ERR=4：输入 N 小于 2，或者大于 11。
P	ARRAY[0..10] OF POINT	N 输出值	

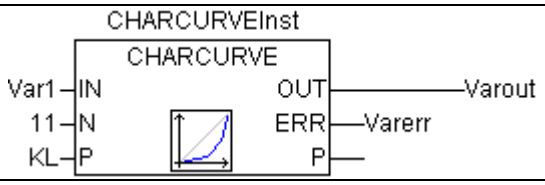
指令使用举例

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;
指令列表（IL）	CAL CHARCURVEInst(IN := Var1, N := 11, P := KL) LD CHARCURVEInst.ERR ST Varerr LD CHARCURVEInst.OUT ST Varout
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。

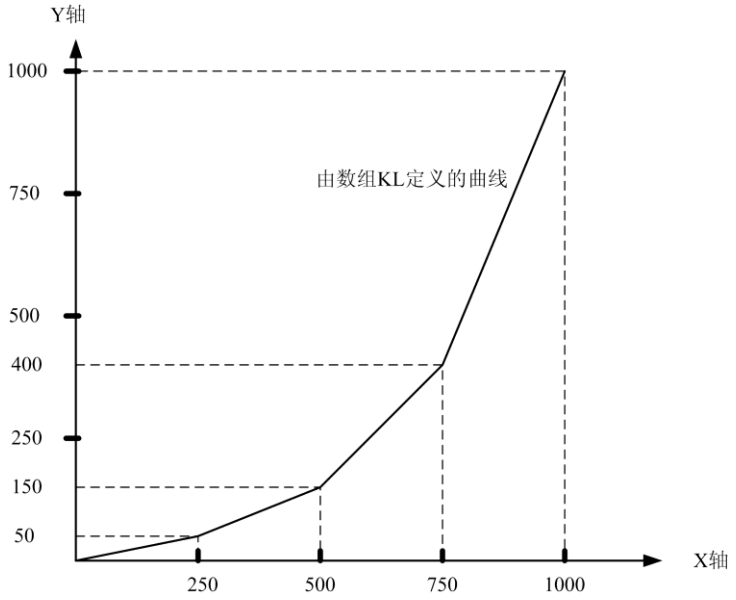


图 4-21-1

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

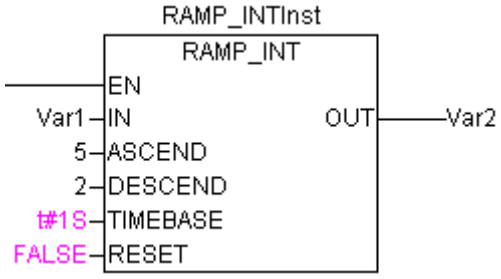
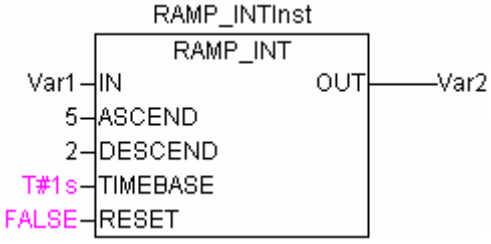
TIMEBASE	TIME	时间基数	规定上升或者下降的速率
RESET	BOOL	初始化	设置为 TRUE 时，RAMP_INT 被重新初始化
输出参数	**数据类型**	**功能描述**	**参数值说明**
OUT	INT	即时数据输出	

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;
指令列表（IL）	CAL　RAMP_INTInst(IN := Var1, ASCEND := 5, DESCEND := 2, TIMEBASE := T#1000ms, RESET := FALSE) LD　RAMP_INTInst.OUT ST　Var2
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。

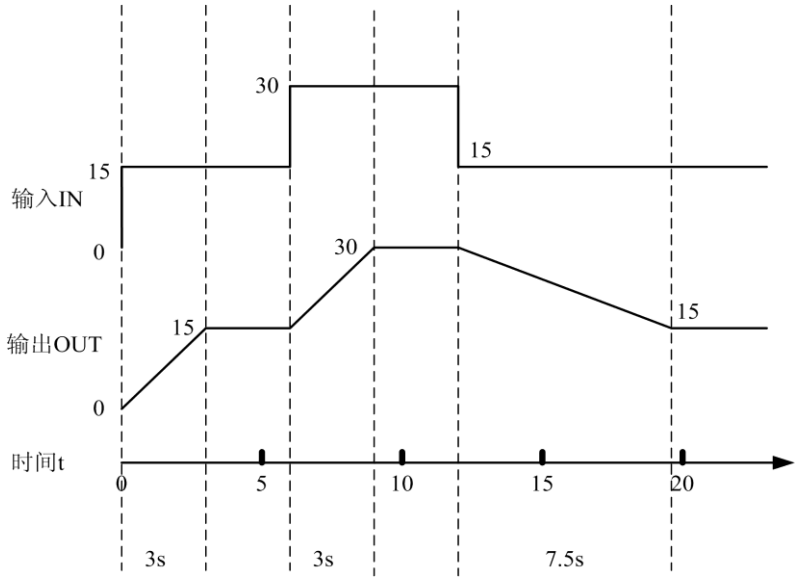


图 4-21-2

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

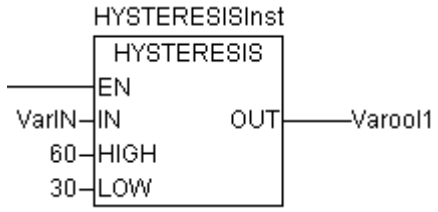
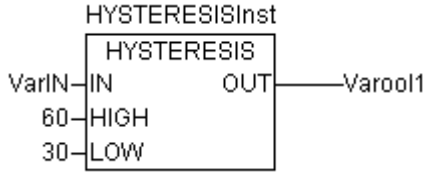
输入参数	数据类型	功能描述	参数值说明
IN	INT	输入值	
HIGH	INT	上限值	
LOW	INT	下限值	
输出参数	**数据类型**	**功能描述**	**参数值说明**
OUT	BOOL	输出值	低下限值后为 TRUE，直到上限值后为恢复 FALSE

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;
指令列表（IL）	CAL　HYSTERESISInst(IN := VarIN, HIGH := 60, LOW := 30) LD　HYSTERESISInst.OUT ST　Varool1
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。

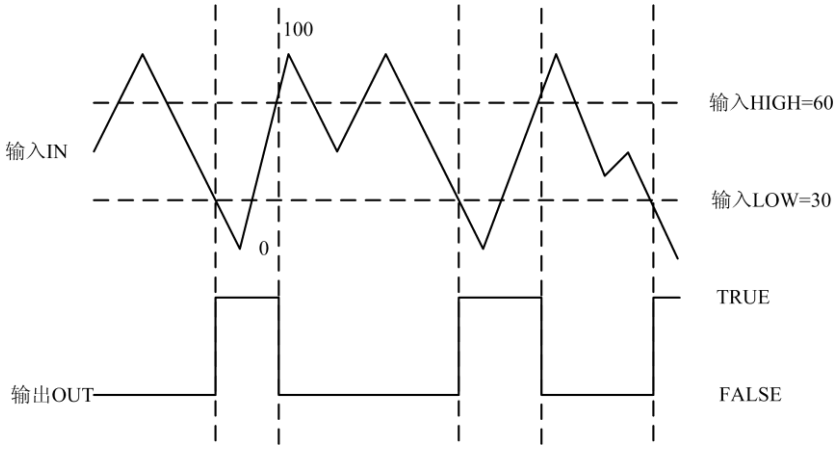


图 4-22-1

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	

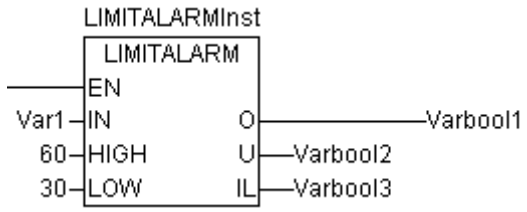
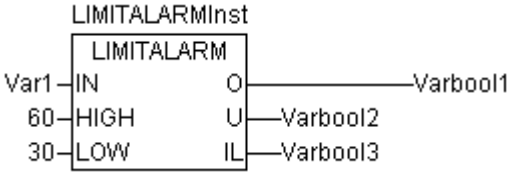
指令使用举例

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;
指令列表（IL）	CAL LIMITALARMInst(IN := Var1, HIGH := 60, LOW := 30) LD LIMITALARMInst.U ST Varbool2 LD LIMITALARMInst.IL ST Varbool3 LD LIMITALARMInst.O ST Varbool1
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。

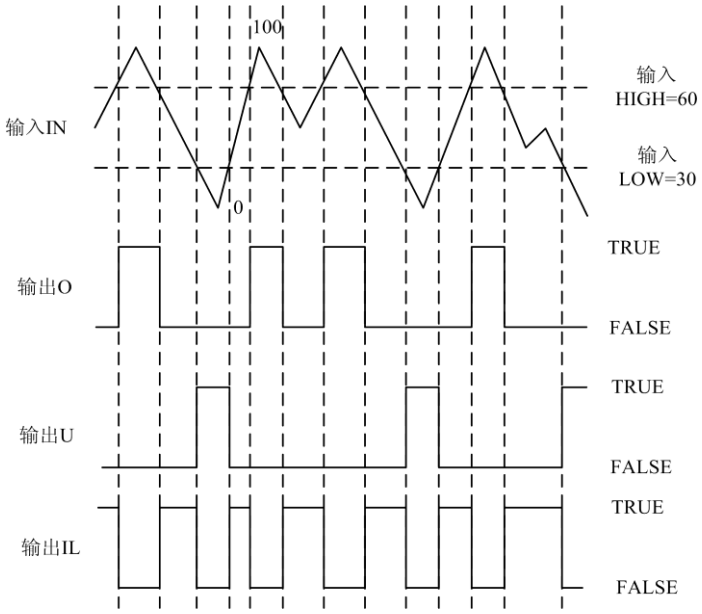


图 4-22-2

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

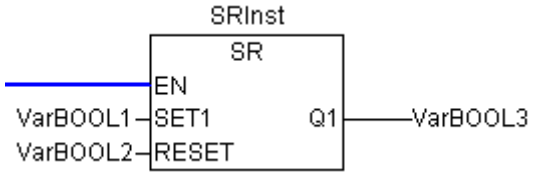
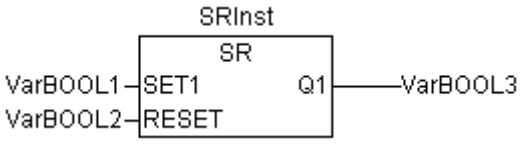
  其中 SET1 为置位信号，RESET 为复位信号。
- 输入/输出数据类型：均为 BOOL 型。

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 ;
指令列表（IL）	CALSRInst(SET1:= VarBOOL1, RESET:= VarBOOL2) LD　　SRInst.Q1 ST　　VarBOOL3
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)

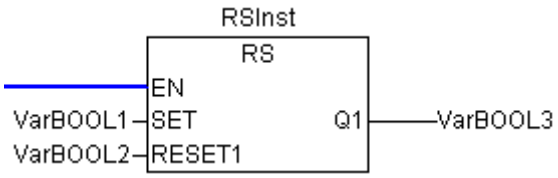
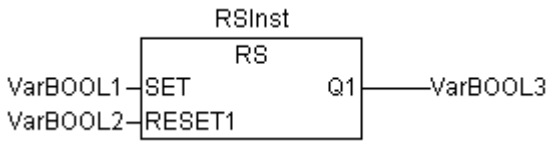
  其中 SET 为置位信号，RESET1 为复位信号。
- 输入/输出数据类型：均为 BOOL。

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 ;
指令列表（IL）	CAL RSInst(SET := VarBOOL1, RESET1 := VarBOOL2) LD RSInst.Q1 ST VarBOOL3
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;
指令列表（IL）	CAL RTRIGInst(CLK := VarBOOL1) LD RTRIGInst.Q ST VarBOOL2
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。

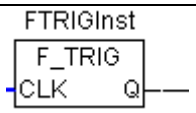
指令使用举例

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;
指令列表（IL）	CAL FTRIGInst(CLK := VarBOOL1) LD FTRIGInst.Q ST VarBOOL2
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	

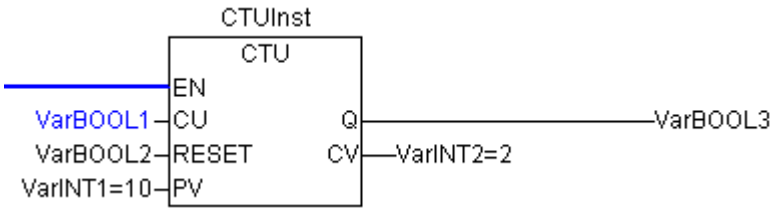
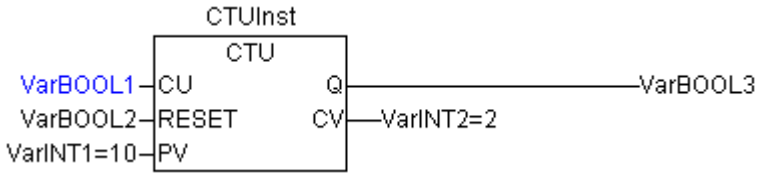
指令使用举例

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;
指令列表（IL）	CAL CTUInst(CU := VarBOOL1, RESET := VarBOOL2, PV :=VarINT1) LD CTUInst.Q ST VarBOOL3 LD CTUInst.CV ST VarINT2
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	

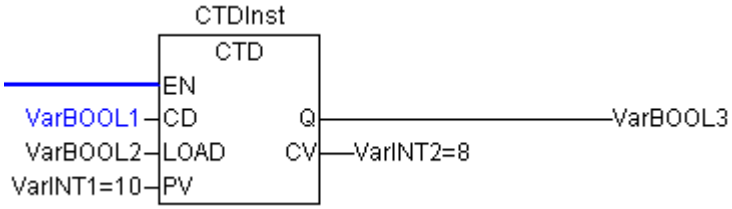
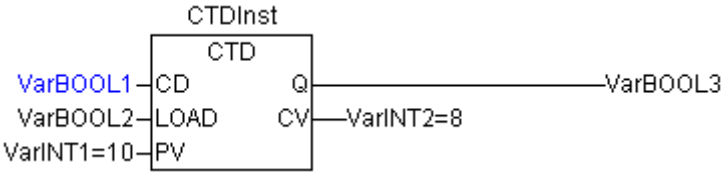
指令使用举例

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;
指令列表（IL）	CAL CTDInst(CD := VarBOOL1, LOAD := VarBOOL2, PV :=VarINT1) LD CTDInst.Q ST VarBOOL3 LD CTDInst.CV ST VarINT2
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

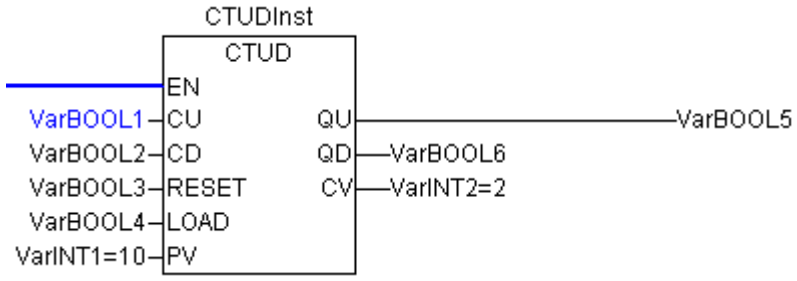
变量定义

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）



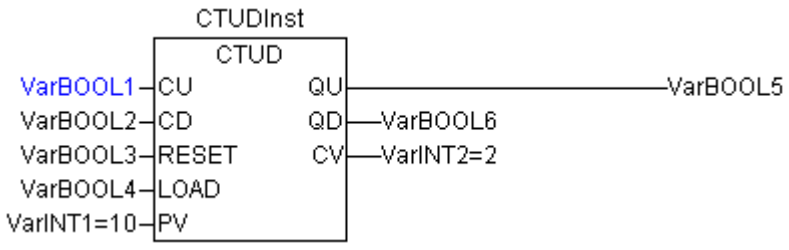
结构化文本（ST）

```
CTUDInst(CU:=VarBOOL1,CD:=VarBOOL2,RESET:=VarBOOL3,
LOAD:=VarBOOL4,PV:=VarINT1)
VarBOOL5 := CTUDInst.QU
VarBOOL6 := CTUDInst.QD
VarINT2 := CTUDInst.CV
```

指令列表（IL）

```
CAL     CTUDInst(CU:=VarBOOL1,CD:=VarBOOL2,
RESET:=VarBOOL3,LOAD:=VarBOOL4,PV:=VarINT1)
LD      CTUDInst.QU
ST      VarBOOL5
LD      CTUDInst.QD
ST      VarBOOL6
LD      CTUDInst.CV
ST      VarINT2
```

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，

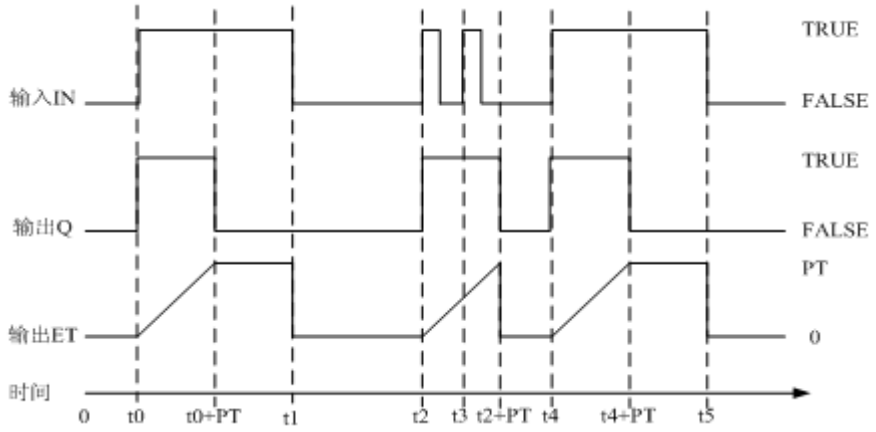
下面分别描述。

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

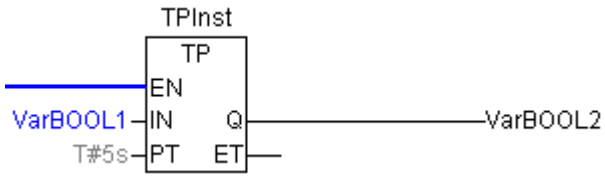
- TP 时间顺序图

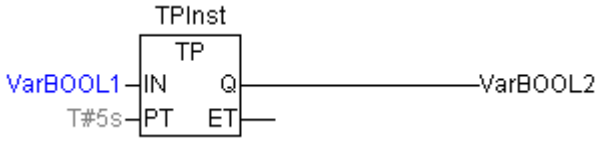


指令使用举例

变量定义

	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）	

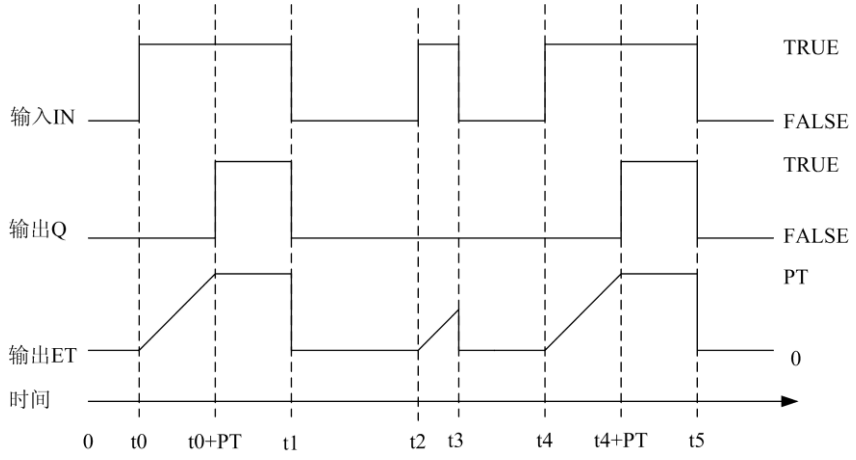
结构化文本（ST）	TPInst(IN:= VarBOOL1,PT:= T#5s); VarBOOL2:=TPInst.Q;
指令列表（IL）	CAL TPInst(IN:= VarBOOL1,PT:=T#5s) LD TPInst.Q ST VarBOOL2
功能块（FBD）	

2.23.2 TON——通电延时定时器

- 功能：通电延时定时器。
- 参数说明

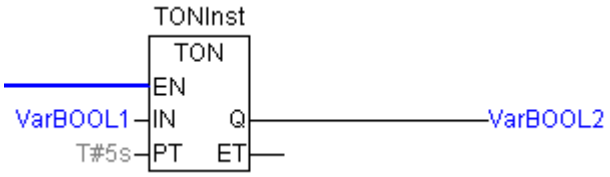
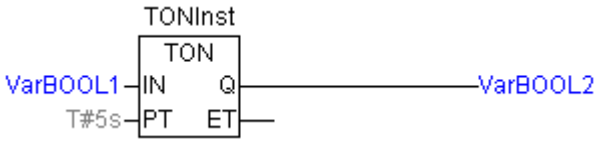
输入参数	**数据类型**	**功能描述**	**参数值说明**
IN	BOOL	定时器初始化信号	当 IN 变成 TRUE 时，ET 以毫秒计时直到 ET 等于 PT，然后 ET 保持常数
PT	TIME	定时时间值	
输出参数	**数据类型**	**功能描述**	**参数值说明**
Q	BOOL	定时器输出	当 IN 为 FALSE 时，Q 为 FALSE，ET 为 0。当 IN 为 TRUE 时，定时器开始工作，ET 等于 PT 时，Q 为 TRUE
ET	TIME	当前时间值	当 IN 变成 TRUE 时，ET 以毫秒计时直到 ET 等于 PT，然后 ET 保持常数。无论何时，当 IN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

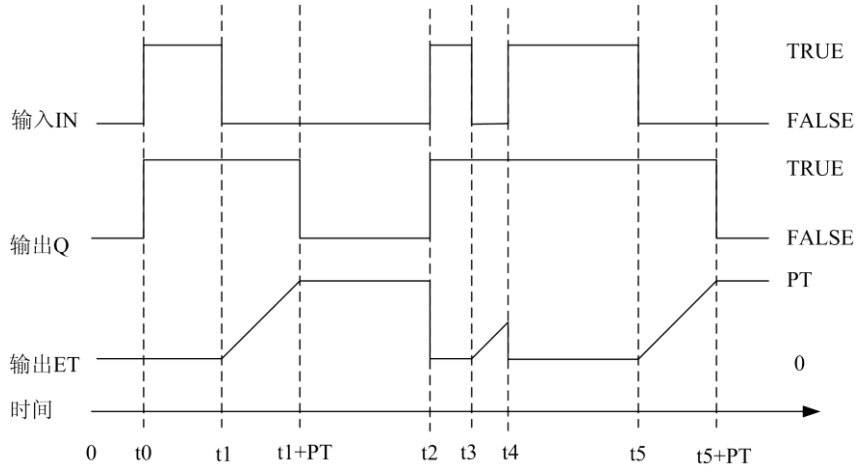
编程语言	程　序
梯形图（LD）	
结构化文本（ST）	TONInst(IN:=VarBOOL1,PT:= T#5s);
指令列表（IL）	CAL　TONInst(IN:= VarBOOL1,PT:=T#5s) LD　TONInst.Q ST　VarBOOL2
功能块（FBD）	

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）	TOFInst TOF EN VarBOOL1 IN Q VarBOOL2 T#5s PT ET
结构化文本（ST）	TOFInst(IN:= VarBOOL1,PT:=T#5s); VarBOOL2 :=TOFInst.Q;
指令列表（IL）	CAL　TOFInst(IN:= VarBOOL1,PT:= T#5s) LD　TOFInst.Q ST　VarBOOL2
功能块（FBD）	TOFInst TOF VarBOOL1 IN Q VarBOOL2 T#5s PT ET

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

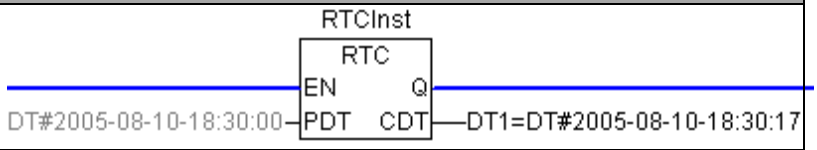
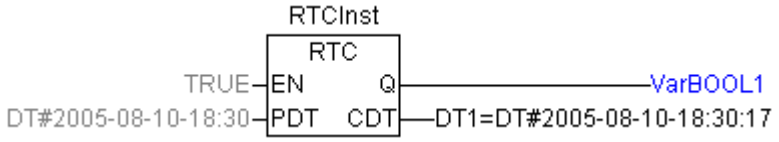
CDT	DT	当前时间值	EN 为 FALSE 时，CDT 为 1970-01-01-00:00:00，EN 为 TRUE 时，CDT 从 PDT 中的时间开始计时

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
梯形图（LD）	
结构化文本（ST）	RTCInst(PDT:=DT#2005-08-10-18:30:31); DT1:=RTCInst.CDT; VarBOOL1:=RTCInst.Q;
指令列表（IL）	CAL　RTCInst(PDT := DT#2005-08-10-18:30:31) LD　RTCInst.CDT ST　DT1 LD　RTCInst.Q ST　VarBOOL1
功能块（FBD）	

附录 A

➢ 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
	幂指令	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

联系我们

北京 ABB 电气传动系统有限公司
地址：北京市朝阳区酒仙桥北路甲 10 号 D 区 1 号
邮编：100015
总机：(86-10)58217788
传真：(86-10)58217518
服务热线： 400 810 8885
E-Mail：plc.service@cn.abb.com
网址： http://www.abb.com.cn/plc