LISTEN.
THINK.
SOLVE.℠

Logix5000™控制器基本指令集

1756 ControlLogix®, 1769 CompactLogix™, 1789 SoftLogix™,
1794 FlexLogix™, 带有DriveLogix的PowerFlex 700S
参考手册

ALLEN-BRADLEY • ROCKWELL SOFTWARE • DODGE • RELIANCE ELECTRIC

Rockwell Automation

重要用户信息

由于本出版物中所述产品使用方法不同，为了对产品的应用负责，在使用时必须按照所有必要步骤进行操作，以确保每个实际应用都满足性能指标且符合安全要求，包括应用准则、规则、规范和标准。罗克韦尔自动化对使用此产品所造成的间接或因此而产生的损坏不负任何责任。

本手册中的图解、表格、程序实例和电路设计举例，完全是作为范例给出。因为各个应用不尽相同，且对于每个应用涉及到具体安装时又有许多不同要求，罗克韦尔自动化对以本出版物的范例为基础的实际应用所造成的后果不负任何责任(包括智能化产品的可靠性)。

艾伦—布拉德利出版物SGI-1.1，《固态控制的应用、安装和维护安全性指南》(可从当地罗克韦尔自动化办事处获取)，说明了固态设备与机电设备的重要差别，在应用产品时(如本手册所述的产品)必须考虑这些差异。

在本书中，我们使用以下的注释符号提醒您安全方面的注意事项。其中的注释和相关说明有助于用户去识别和避免潜在危险，并意识到潜在危险的后果：

注意 指明可导致人身伤害或死亡，财产毁坏或经济损失的行为和环境。

重要事项 指明成功应用和理解产品的关键信息。

介 绍

该版本的文档包括了新增和更改的信息。为了帮助尽快找到这些内容，特将有变化的项目列于下表。

更新信息

该文档包括如下更新信息：

更新：	查阅 章节/附录：
指令索引中包括每条指令的名称	指令索引
列表显示从R9.x及更低版本到R10.x及更高版本中PLC/SLC错误代码的变化	3
更新如何为信息选择缓存选项。更新包括用户可缓存的连接数量。	3
在CONTROLLERDEVICE对象的ProductCode属性中附加产品代码。	3
TASK对象的新增属性： • DisableUpdateOutputs（禁止输出更新) • EnableTimeOut（使能超时) • InhibitTask（禁止任务) • OverlapCount（重叠计数) • Status（状态)	3
设置系统数据（SSV)指令现在允许用户编程更改任务的优先级和周期型任务的频率。	3
对于事件型任务，使用TASK对象的Rate（周期)属性获取或设置事件型任务的超时时间值。	3
立即输出（IOT)指令。使用IOT指令立即更新输出数据或触发另一控制器中的事件任务。	3
SFC复位（SFR)指令如何影响存储动作的说明。	10
触发事件任务（EVENT)指令。使用EVENT指令触发事件任务的执行。	10
关于如何选择功能块元素，包括IREFs、OREFs、ICONs和OCONs	B
周期型定时方式下使用功能块指令的更新信息	B
说明结构文本CASE结构体与C/C++转换语句的区别	C

注释：

快速查找指令

使用此索引可查找到Logix 指令的详细内容(灰色的指令可在其它手册查到)。该索引同时列出了该指令可使用的编程语言。

如果索引列出：	指令所在文档：
页码	本手册中
过程控制	Logix5000 控制器过程控制和驱动指令集参考手册（Logix5000 Controllers Process Control and Drives Instruction Set Reference Manual)，出版号1756-RM006
运动控制	Logix5000控制器运动指令集参考手册（Logix5000 Controllers Motion Instruction Set Reference Manual)，出版号1756-RM007

指令：	位置：	编程语言：
ABL ASCII测试缓冲区行	5-16	梯形图 结构文本
ABS 绝对值	5-29	梯形图 结构文本 功能块
ACB ASCII计数缓冲区中字符数	8-16	梯形图 结构文本
ACL ASCII清空缓冲区	10-16	梯形图 结构文本
ACOS 余弦	13-14	结构文本
ACS 反余弦	13-14	梯形图 功能块
ADD 加	5-6	梯形图 结构文本 功能块
AFI 恒假指令	10-23	梯形图
AHL ASCII 握手信号线	12-16	梯形图 结构文本
ALM Alarm	过程控制	结构文本 功能块
AND 按位与	6-23	梯形图 结构文本 功能块
ARD ASCII读	16-16	梯形图 结构文本
ARL ASCII读行	16-19	梯形图 结构文本
ASIN 正弦	11-13	结构文本

指令：	位置：	编程语言：
ASN 反正弦	11-13	梯形图 功能块
ATAN 正切	13-17	结构文本
ATN 反正切	13-17	梯形图 功能块
AVE 文件平均值	7-38	梯形图
AWA ASCII写附加	16-23	梯形图 结构文本
AWT ASCII写	16-28	梯形图 结构文本
BAND 逻辑与	6-35	结构文本 功能块
BNOT 逻辑非	6-44	结构文本 功能块
BOR 逻辑或	6-38	结构文本 功能块
BRK 中断	11-5	梯形图
BSL 位左移	8-2	梯形图
BSR 位右移	8-5	梯形图
BTD 位域分配	6-11	梯形图
BTDT 带目标的位域分配	6-14	结构文本 功能块
BTR 信息	3-2	梯形图 结构文本

指令:	位置:	编程语言:
HLL 上/下限	过程控制	结构文本 功能块
HPF 高通滤波	过程控制	结构文本 功能块
ICON 输入线连接器	B-1	功能块
INSERT 插入字符串	9-17	梯形图 结构文本
INTG 积分器	过程控制	结构文本 功能块
IOT 立即输出	3-57	梯形图 结构文本
IREF 输入参考	B-1	功能块
JKFF JK触发器	过程控制	结构文本 功能块
JMP 跳转到标号	10-2	梯形图
JSR 跳转到子例程	10-4	梯形图 结构文本 功能块
JXR 跳转到外部例程	10-14	梯形图
LBL 标号	10-2	梯形图
LDL2 二阶超前-滞后	过程控制	结构文本 功能块
LDLG 超前-滞后	过程控制	结构文本 功能块
LEQ 小于或等于	4-19	梯形图 结构文本 功能块
LES 小于	4-23	梯形图 结构文本 功能块
LFL LIFO 装载	8-20	梯形图
LFU LIFO 卸载	8-26	梯形图
LIM 极限比较	4-27	梯形图 功能块
LN 自然对数	2-14	梯形图 结构文本 功能块

指令:	位置:	编程语言:
LOG 以10为底的对数	4-14	梯形图 结构文本 功能块
LOWER 转换成小写	18-14	梯形图 结构文本
LPF 低通滤波	过程控制	结构文本 功能块
MAAT 运动轴整定应用	运动控制	梯形图 结构文本
MAFR 运动轴故障复位	运动控制	梯形图 结构文本
MAG 运动轴电子齿轮传动	运动控制	梯形图 结构文本
MAH 运动轴回零	运动控制	梯形图 结构文本
MAHD 运动接线诊断应用	运动控制	r梯形图 结构文本
MAJ 运动轴按速度给定移动	运动控制	梯形图 结构文本
MAM 运动轴按位置给定移动	运动控制	梯形图 结构文本
MAOC 运动输出凸轮启动	运动控制	梯形图 结构文本
MAPC 运动轴位置凸轮	运动控制	梯形图 结构文本
MAR 运动套准输入使能	运动控制	梯形图 结构文本
MAS 运动轴停止	运动控制	梯形图 结构文本
MASD 运动轴关断	运动控制	梯形图 结构文本
MASR 运动轴关断复位	运动控制	梯形图 结构文本
MATC 运动轴时间凸轮	运动控制	梯形图 结构文本
MAVE 移动均值运算	过程控制	结构文本 功能块
MAW 运动置位监视	运动控制	梯形图 结构文本
MAXC 捕捉最大值	过程控制	结构文本 功能块

简介

本手册是Logix5000 系列指令资料之一。

任务/目标：	参见资料：
控制器顺序控制应用编程 本手册 **You are here** 	Logix5000控制器基本指令参考手册（Logix5000 Controllers General Instructions Reference Manual) 出版号：1756-RM003
控制器过程控制驱动应用编程	Logix5000控制器过程控制及驱动指令参考手册（Logix5000 Controllers Process Control and Drives Instructions Reference Manual) 出版号：1756-RM006
运动系统应用编程	Logix5000控制器运动指令集参考手册（Logix5000 Controllers Motion Instruction Set Reference Manual) 出版号：1756-RM007
导入文本文件或标签到一个项目 导出一个项目或标签到文本文件	Logix5000控制器导入/导出文件参考手册（Logix5000 Controllers Import/Export Reference Manual) 出版号：1756-RM084
将PLC5或SLC500应用程序转换成Logix5000应用程序	Logix5550控制器转换PLC-5或SLC500程序逻辑到Logix5550应用程序参考手册（Logix5550 Controller Converting PLC-5 or SLC 500 Logic to Logix5550 Logic Reference Manual) 出版号：1756-6.8.5

本手册使用对象

本手册为程序设计者提供Logix系列控制器所用指令的详细信息。用户首先应该已经熟悉Logix系列控制器存储并处理数据的原理。

初学者在使用一条指令之前，应该首先阅读与指令有关的详细资料。有经验的程序设计者可以只查阅有关信息来确认指令的详细内容。

本手册的用途

本手册以如下格式为每条指令提供说明：

该区域：	提供以下信息：
指令名称	标识指令 确定指令是输入指令或输出指令
操作数	列出指令的所有操作数类型  在梯形图中能够使用的操作数类型  在结构文本中能够使用的操作数类型  在功能块中能够使用的操作数类型 功能块所显示的引脚均为缺省引脚。操作数列表显示功能块所有可能引脚的操作数。
指令结构体	列出指令的控制状态位及其数值
说明	说明指令的用法 如果需要，说明指令使能时和禁止时的区别
算术状态标志	指示指令是否影响算术状态标志 参阅附录通用属性
故障条件	“确定指令是否产生次要或主要故障，如果产生，指明故障类型及故障代码”
执行	详细描述该指令在特定条件下如何操作的
实例	每种可用编程语言至少提供一个实例，每个实例都包含解释说明

以下图标用于帮助定义编程语言的特定信息：

图标：	表示以下编程语言：
	梯形图
	结构文本
	功能块

所有指令通用信息

Logix5000指令集具有以下通用属性

信息：	参考附录：
通用属性	在通用属性附录中说明了： • 算术状态标志 • 数据类型 • 关键字
功能块属性	在功能块属性附录中说明了： • 编程和操作员控制 • 定时模式

约定和相关术语

置位和清零

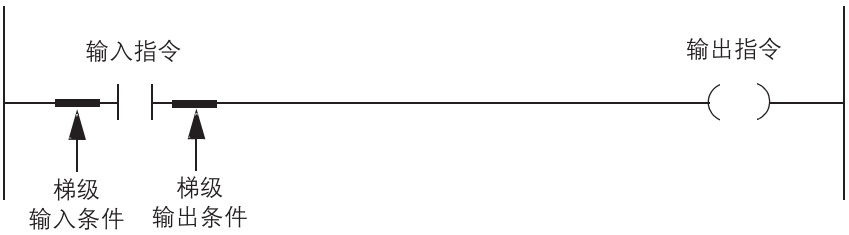
本手册使用置位/设置和清零来定义位(布尔型)和数值(非布尔型)的状态：

术语：	含义：
置位	将某一位设置为1 (ON) 将某个数值设置为非零数值
清零	将某一位设置为0 (OFF) 将某个数值的所有位设置为零

假如操作数或参数支持多种数据类型，则黑体字数据类型为最优数据类型。指令的所有操作数的使用同种最优数据类型，则指令执行速度最快且占用内存最小，典型的最优数据类型是DINT和REAL。

梯级运行条件

控制器根据指令前面的梯级条件(梯级输入条件)判断梯形图指令。然后根据梯级条件和指令，控制器设置下一指令的梯级条件(梯级输出条件)，这样交替影响后面的指令。



如果一个输入指令的梯级输入条件为真，则控制器执行该指令，并根据指令执行的结果，设置梯级输出条件。如果指令执行结果为真，则输出梯级条件为真；如果该指令执行结果为假，则输出梯级条件为假。

控制器同时预扫描梯级。预扫描是控制器对所有梯级的一种特殊扫描方式，控制器在预扫描期间扫描所有主例程和子例程，但会忽略跳过执行的跳转指令。控制器执行所有FOR循环和子例程调用。如果一个子例程被多次调用，则每次调用都会执行。控制器通过预扫描梯形图指令来重置非保持型I/O和内部数据。

在预扫描期间，输入数据不是当前数据，输出数据也不写入。以下条件产生预扫描：

- 将程序从编程状态转换到运行状态；
- 从上电状态自动转换到运行状态。

当程序处于下列条件时不进行预扫描：

- 当控制器运行时，程序为预定型；
- 当控制器进入运行模式时，程序为非预定型。

功能块状态

重要事项

由于其内部浮点数计算使用单精度浮点数，在功能块编程时，工程量范围严格限制在+/-10+/-15。如果工程单位超出此范围但接近单精度浮点数极限(+/-10+/-38)会导致计算结果精度降低。

控制器根据不同的状态条件执行功能块指令。

可能的条件：	描述：
预扫描	功能块例程的预扫描与梯形图的预扫描相同。唯一的不同就是每个功能块使能参数在预扫描期间都被清零。
指令首次扫描	指令首次扫描是指指令在预扫描后首次执行。控制器使用首次指令扫描来读取当前输入并确定输入的状态。
指令首次执行	指令首次执行是指指令在新的数据结构体的情况下的首次运行。控制器使用首次运行来产生系数和其它数据存储，但不改变初始下载的功能块

每个功能块指令包括EnableIn(输入使能)和EnableOut(输出使能)参数：

- 当输入使能位置位时功能块指令正常执行；
- 当输入使能位清零时功能块指令只执行预扫描逻辑、后期扫描逻辑或跳过正常算法执行。
- 输出使能位为输入使能位映象，但是当功能块执行时检测到一个溢出条件，则输出使能位也被清零。
- 当输入使能位从清零变为置位时，功能块从断点继续执行。然而有些功能块指令包含特殊功能函数，如重新初始化，当使能位由清零转换到置位状态，则重新初始化。对于有时间基参数的功能块指令且定时模式为过采样方式，当输入使能位从清零转换为置位时，功能块总是从断点继续执行。

当输入使能位不连接时，指令总是将输入使能位置位条件正常执行。如果用户清零输入使能位，则下次指令执行时置位。

注释：

目录

比较指令
(CMP，EQU，GEQ，
GRT，LEQ，LES，LIM，
MEQ，NEQ)

第四章

计算/算术指令 (CPT，ADD，SUB，MUL，DIV，MOD，SQR，SQRT，NEG，ABS)

第五章

传送/逻辑指令 (MOV，MVM，BTD，MVMT，BTDT，CLR，SWPB，AND，OR，XOR，NOT，BAND，BOR，BXOR，BNOT)

第六章

数组(文件)/综合指令 (FAL，FSC，COP，CPS，FLL，AVE，SRT，STD，SIZE)

第七章

数组(文件)/移位指令 (BSL，BSR，FFL，FFU，LFL，LFU)

顺序器指令 (SQI, SQO, SQL)

程序控制指令 (JMP，LBL，JSR，RET，SBR，JXR，TND，MCR，UID，UIE，AFI，NOP，EOT，SFP，SFR，EVENT)

循环/终止循环指令（FOR，FOR…DO，BRK，EXIT，RET）

第十一章

专用指令（FBC，DDT，DTR，PID）

第十二章

三角函数指令
(SIN，COS，TAN，
ASN，ASIN，ACS，
ACOS，ATN ，ATAN)

第十三章

高级算术指令
(LN，LOG，XPY)

第十四章

算术转换指令
(DEG，RAD，TOD，
FRD，TRN ，TRUNC)

第十五章

ASCII 串行口指令
(ABL，ACB，ACL，
AHL，ARD，ARL，
AWA，AWT)

第十六章

ASCII 字符串指令（CONCAT，DELETE，FIND，INSERT，MID）

第十七章

ASCII 转换指令（STOD，STOR，DTOS，RTOS，UPPER，LOWER）

第十八章

通用属性

附录A

功能块属性

附录B

结构文本编程

附录C

位指令
(XIC，XIO，OTE，OTL，OTU，ONS，OSR，OSF，OSRI，OSFI)

简介

位(继电器类型)指令用于监视和控制位的状态。

如果用户需要：	所用指令：	可用语言：	参见页码：
当被寻址的位是置位状态时，使能梯级输出	XIC	梯形图 结构文本(1)	1-2
当被寻址的位是清零状态时，使能梯级输出	XIO	梯形图 结构文本(1)	1-4
将被寻址的位设置成置位状态	OTE	梯形图 结构文本(1)	1-6
将被寻址的位设置成置位状态(保持)	OTL	梯形图 结构文本(1)	1-8
将被寻址的位设置成清零状态(保持)	OTU	梯形图 结构文本(1)	1-10
每次梯级输入条件变为真时，在一个扫描周期内使能梯级输出	ONS	梯形图 结构文本(1)	1-12
梯级输入条件变为真时，将被寻址的位设置成置位状态一个扫描周期	OSR	梯形图	1-15
梯级输入条件变为假时，在一个扫描周期内将被寻址的位置1	OSF	梯形图	1-17
当功能块中的输入位是置位状态时，在一个扫描周期内将被寻址的位置位	OSRI	结构文本 功能块	1-19
当功能块中的输入位是置位状态时，在一个扫描周期内将被寻址位清零	OSFI	结构文本 功能块	1-22

(1)无等价的结构文本指令。可使用其它结构文本编程来完成相同功能。参见指令说明。

检查位是否闭合 (XIC)

XIC 指令查看数据位是否为置位状态。

操作数：

梯形图

操作数：	数据类型：	格式：	说明：
数据位	BOOL	标签	该位被检测

结构文本

结构文本中不存在XIC指令，但用户可使用IF…THEN结构体完成相同功能。

```
IF data_bit THEN
        <statement>;
END_IF;
```

关于结构文本中结构体语法信息，请参见附录C。

说明： XIC 指令查看数据位是否为置位状态。

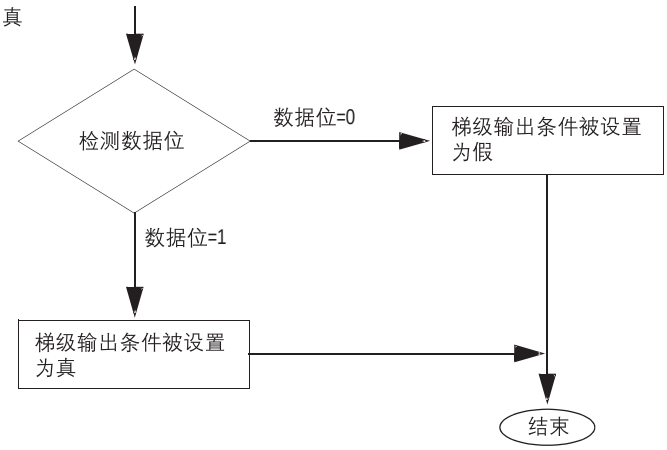
算术状态标志： 不影响

故障条件： 无

执行：

梯级条件：	梯形图动作：
预扫描	梯级输出条件被设置为假。
梯级输入条件为假	梯级输出条件被设置为假。

梯级输入条件为真



后期扫描	梯级输出条件被设置为假。

例 1：如果 limit_switch_1 是置位状态，则使能下一条指令 (梯级输出条件为真)。

梯形图

结构文本

```
IF limit_switch THEN
        <statement>;
END_IF;
```

例 2：如果S:V为置位状态 (表示已经发生溢出)，则使能下一条指令 (梯级输出条件为真)。

梯形图

结构文本

```
IF S:V THEN
        <statement>;
END_IF;
```

检查位是否断开 (XIO)

XIO 指令查看数据位是否为清零状态。

操作数：

梯形图

操作数：	数据类型：	格式：	说明：
数据位	BOOL	标签	被检测的位

结构文本

结构文本中不存在XIO指令，但用户可使用IF…THEN结构体完成相同功能。

```
IF NOT data_bit THEN
        <statement>;
END_IF;
```

关于结构文本中结构体语法信息，请参见附录C。

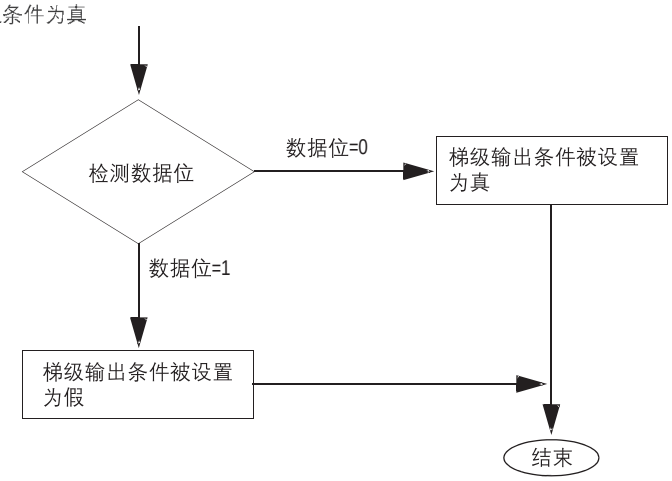
说明： XIO 指令查看数据位是否为清零状态。

算术状态标志： 不影响

故障条件： 无

执行：

梯级条件：	梯形图动作：
预扫描	梯级输出条件被设置为假。
梯级输入条件为假	梯级输出条件被设置为假。
梯级输入条件为真	



后期扫描	梯级输出条件被设置为假。

例 1：如果 limit_switch_2 是清零状态，则使能下一条指令 (梯级输出条件为真)。

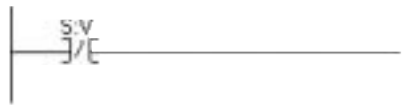
梯形图

结构文本

```
IF NOT limit_switch_2 THEN
    <statement>;
END_IF;
```

例 2：如果S:V 是清零状态 (表示没有发生溢出)，则使能下一条指令 (梯级输出条件为真)。

梯形图



结构文本

```
IF NOT S:V THEN
    <statement>;
END_IF;
```

使能输出 (OTE)

OTE 指令置位或清零数据位。

操作数：


梯形图

操作数：	数据类型：	格式：	说明：
数据位	BOOL	标签	被置位或清零的位

结构文本

结构文本中不存在OTE指令，但用户可使用非保持赋值语句完成相同功能。
data_bit [:=] BOOL_expression;

关于结构文本中结构体语法信息，请查阅附录C。

说明： 如果OTE 指令被使能，则控制器置位寻址数据位。如果OTE指令被禁止，则控制器清零寻址的数据位。

算术状态标志： 不影响

故障条件： 无

执行：

梯级条件：	梯形图动作：
预扫描	清零数据位。 梯级输出条件被设置为假。
梯级输入条件为假	清零数据位。 梯级输出条件被设置为假。
梯级输入条件为真	置位数据位。 梯级输出条件被设置为真。
后期扫描	清零数据位。 梯级输出条件被设置为假。

举例：如果置位switch，则OTE指令置位(打开)light_1。如果清零switch，则OTE指令清零(关断)light_1。

梯形图

结构文本

light_1 [:=] switch;

输出锁存 (OTL)

OTL 指令置位(锁存)数据位。

操作数：


梯形图

操作数：	数据类型：	格式：	说明：
数据位	BOOL	标签	将置位的位

结构文本

结构文本中不存在OTL指令，但用户可使用IF…THEN结构体和赋值语句完成相同功能。

```
IF BOOL_expression THEN
        data_bit := 1;
END_IF;
```

关于结构文本中结构体语法、表达式和赋值语句信息，请参见附录C。

说明： 如果OTL 指令被使能，则控制器置位寻址的数据位。数据位会保持置位状态直到被清零，通常用OTU指令来完成。如果OTL指令被禁止，则控制器不改变数据位的状态。

算术状态标志： 不影响

故障条件： 无

执行：

梯级条件：	梯形图动作：
预扫描	不改变数据位状态。 梯级输出条件被设置为假。
梯级输入条件为假	不改变数据位。 梯级输出条件被设置为假。
梯级输入条件为真	置位数据位。 梯级输出条件被设置为真。
后期扫描	不改变数据位状态。 梯级输出条件被设置为假。

举例：如果OTL 指被使能，则置位(打开)light_2，该数据位会保持置位状态直到被清零，通常用OTU指令来完成。

梯形图

结构文本

```
IF BOOL_expression THEN
        light_2 := 1;
END_IF;
```

输出解锁存 (OTU)

OTU指令清零(解锁存)数据位。

操作数：

梯形图


操作数:	数据类型:	格式:	说明:
数据位	BOOL	标签	被清零的位

结构文本

结构文本中不存在OTU指令，但用户可使用IF…THEN结构体和赋值语句完成相同功能。

```
IF BOOL_expression THEN
    data_bit := 0;
END_IF;
```

关于结构文本中结构体语法、表达式和赋值语句信息，请参见附录C。

说明： 如果OTU 指令被使能，则控制器清零寻址的数据位。如果OTU指令被禁止，则控制器不改变数据位的状态。

算术状态标志： 不影响

故障条件： 无

执行：

梯级条件:	梯形图动作:
预扫描	不改变数据位。 梯级输出条件被设置为假。
梯级输入条件为假	不改变数据位。 梯级输出条件被设置为假。
梯级输入条件为真	清零数据位。 梯级输出条件被设置为真。
后期扫描	不改变数据位。 梯级输出条件被设置为假。

举例：如果OTU 指令被使能，则清零(关断)light_2。

梯形图

结构文本

```
IF BOOL_expression THEN
    light_2 := 0;
END_IF;
```

单脉冲触发 (ONS)

ONS指令根据存储位的状态使能或禁止梯级的其余部分。

操作数：

梯形图

操作数：	数据类型：	格式：	说明：
存储位	BOOL	标签	内部存储位 存储指令最后一次执行的梯级输入条件

结构文本

结构文本不存在ONS指令，但用户可使用IF…THEN结构体完成相同功能。

```
IF BOOL_expression  AND NOT storage_bit THEN
        <statement>;
END_IF;
storage_bit := BOOL_expression;
```

关于结构文本中结构体语法，表达式和赋值语句信息，请参见附录C。

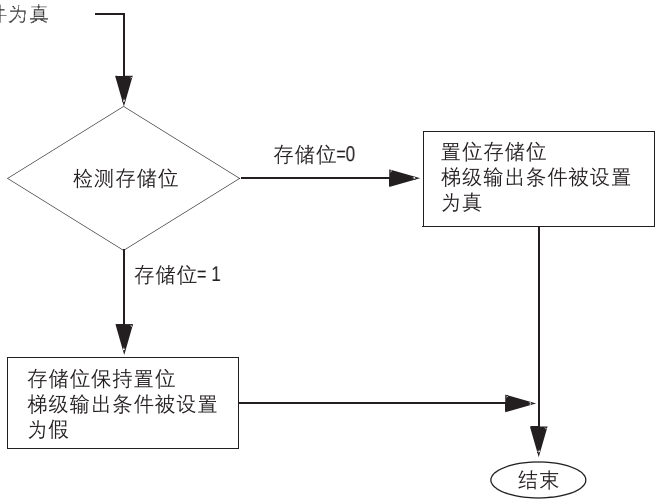
说明： 如果指令被使能时存储位是清零状态，则ONS指令使能梯级的其余部分。如果指令被禁止或存储位是置位状态，则ONS指令禁止梯级的其余部分。

算术状态标志： 不影响

故障条件： 无

执行：

执行：

梯级条件：	**梯形图动作：**
预扫描	置位存储位以防止首次扫描时误触发。 梯级输出条件被设置为假。
梯级输入条件为假	清零存储位。 梯级输出条件被设置为假。
梯级输入条件为真	
后期扫描	清零存储位。 梯级输出条件被设置为假。

举例：通常在ONS指令前面都用一条输入指令，因为扫描ONS指令的正确操作是使其使能后就禁止。一旦ONS指令被使能，则只有梯级输入条件或存储位必须被清零才能再次使能ONS指令。

如果在某次扫描时limit_switch_1是清零状态或 storage_1是置位状态，则不影响梯级。如果在某次扫描时limit_switch_1是置位状态或 storage_1 是清零状态，则ONS指令置位 storage_1同时ADD指令的和加1。只要limit_switch_1保持置位状态，ADD 指令的和就保持不变。只有limit_switch_1 再次从清零状态变为置位状态时，和值才会再增加。

梯形图

结构文本

```
IF limit_switch_1 AND NOT storage_1 THEN
        sum := sum + 1;
END_IF;
storage_1 := limit_switch_1;
```

上升沿触发 (OSR)

OSR指令根据存储位的状态置位或清零输出位。

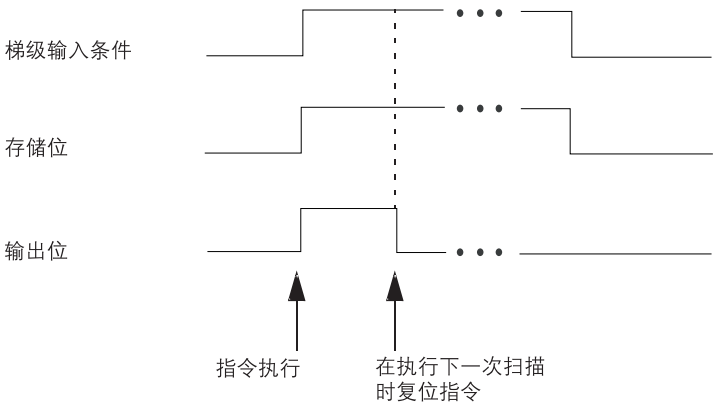
该指令相当于结构文本和功能块编程中的OSRI指令，参见1-19页。

操作数：

梯形图

操作数：	数据类型：	格式：	说明：
storage bit	BOOL	标签	内部存储位 存储指令最后一次执行的梯级输入条件
output bit	BOOL	标签	将设置的位

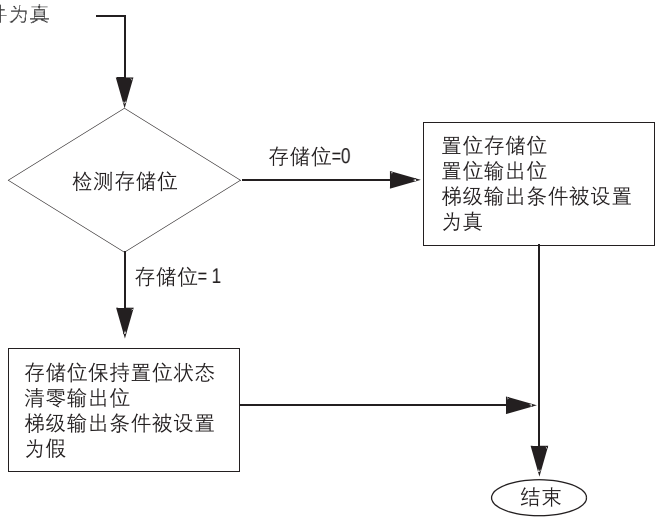
说明： 如果指令被使能且存储位是清零状态，则OSR 指令置位输出位。如果指令被使能且存储位是置位状态或指令被禁止，则OSR 指令清零输出位。



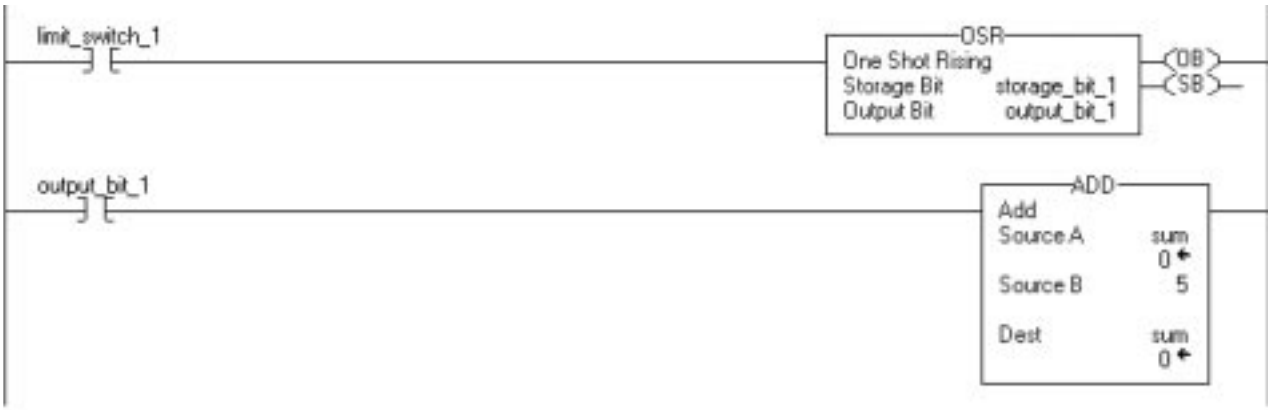
算术状态标志： 不影响

故障条件： 无

执行：

梯级条件：	梯形图动作：
预扫描	置位存储位以防止首次扫描时误触发。 清零输出位。 梯级输出条件被设置为假。
梯级输入条件为假	清零存储位。 不改变输出位。 梯级输出条件被设置为假。
梯级输入条件为真	
后期扫描	清零存储位。 不改变输出位。 梯级输出条件被设置为假。

举例： 每次limit_switch_1从清零状态变为置位状态时，OSR指令都置位output_bit_1同时ADD指令的和加5。只要limit_switch_1保持置位状态，和值就始终不变。只有limit_switch_1再次从清零状态变为置位状态时和值才再增加。用户可以在多个梯级使用output_bit_1来触发其它操作。



下降沿触发 (OSF)

OSF指令根据存储位的状态置位或清零输出位。

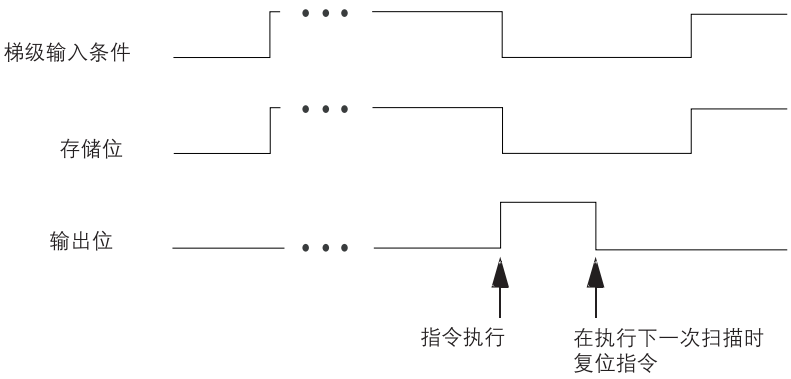
该指令相当于结构文本和功能块编程中的OSFI指令，参见1-22页。

操作数：

梯形图操作数

操作数：	数据类型：	格式：	说明：
storage bit	BOOL	标签	内部存储位 存储指令最后一次执行的梯级输入条件
output bit	BOOL	标签	将设置的位

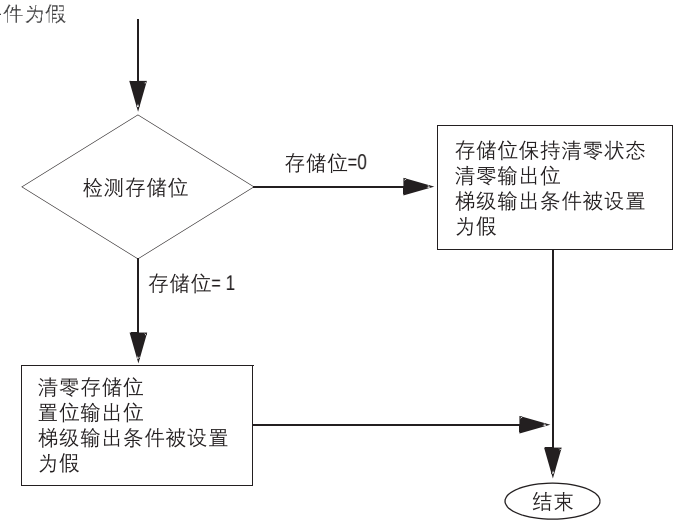
说明： 如果指令被禁止时存储位是置位状态，则OSF 指令置位输出位。如果指令被禁止时存储位是清零状态或指令被使能，则OSF 指令清零输出位。



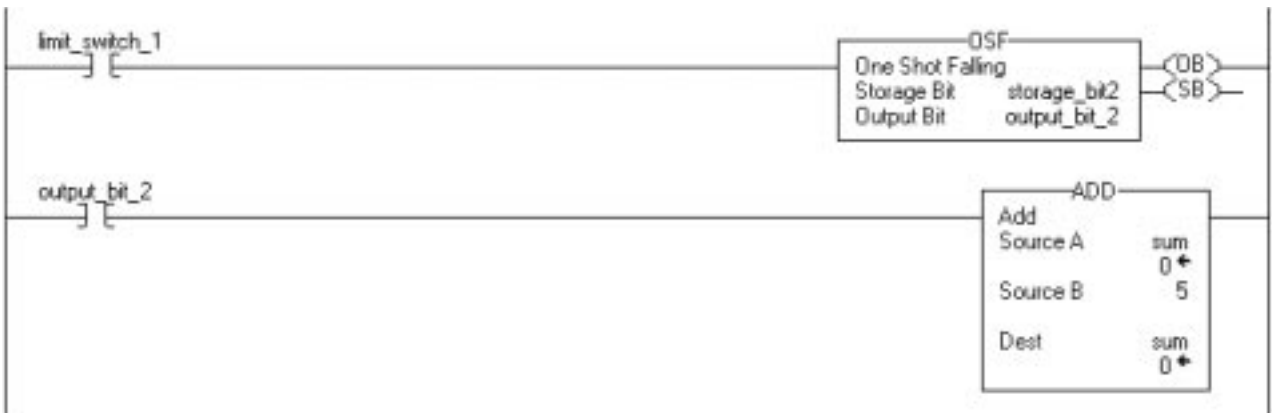
算术状态标志： 不影响

故障条件： 无

执行：

梯级条件：	梯形图动作：
预扫描	清零存储位以防止在首次扫描期间误触发。 清零输出位。 梯级输出条件被设置为假。
梯级输入条件为假	
梯级输入条件为真	置位存储位。 清零输出位。 梯级输出条件被设置为真。
后期扫描	参见上述梯级输入条件为假。

举例： 每次limit_switch_1从置位状态变为清零状态时，OSF指令都置位output_bit_2同时ADD指令的和加5。只要limit_switch_1保持清零状态，和值就始终不变。只有limit_switch_1再次从置位状态变为清零状态时和才再增加。用户可以在多个梯级使用output_bit_2来触发其它操作。



带输入的上升沿触发(OSRI)

当输入位从清零状态变为置位状态触发时，OSRI指令置位输出位一个执行周期。

该指令相当于梯形图编程中的OSR指令，参见1-15页。

操作数：

```
OSRI(OSRI_tag);
```

结构文本

操作数：	数据类型：	格式：	说明：
OSRI标签	FBD_ONESHOT	结构体	OSRI 结构体

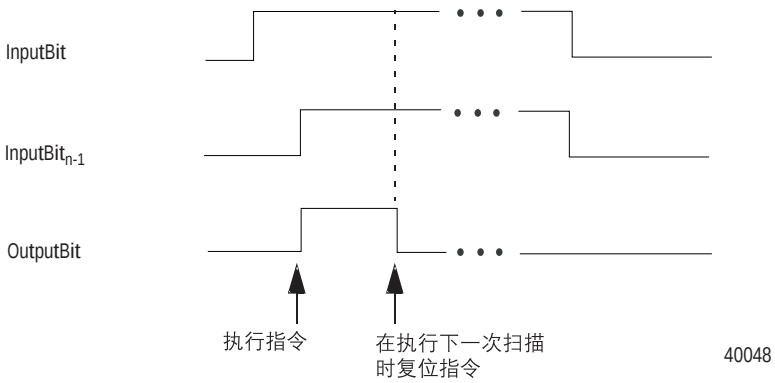
功能块

操作数：	数据类型：	格式：	说明：
OSRI标签	FBD_ONESHOT	结构体	OSRI 结构体

FBD_ONESHOT结构体

输入参数：	数据类型：	说明：
EnableIn	BOOL	**功能块：** 如果被清零，则不执行指令也不更新输出。 如果被置位，执行指令。 缺省状态是置位。 **结构文本：** 无影响。执行指令。
InputBit	BOOL	输入位。与用梯形图编程的OSR指令的梯级条件相同。 缺省状态是清零。
输出参数：	**数据类型：**	**说明：**
EnableOut	BOOL	指令产生一有效结果。
OutputBit	BOOL	输出位

说明：当InputBit是置位状态且InputBit $_{n-1}$ 是清零状态时，OSRI指令置位OutputBit。
当InputBit $_{n-1}$ 是置位状态或InputBit是清零状态时，OSRI指令清零OutputBit。



不影响

算术状态标志：

无

故障条件：

执行：

条件：	**功能块动作**：	**结构文本动作**：
预扫描	不动作。	不动作。
首次扫描指令	置位InputBit $_{n-1}$。	置位InputBit $_{n-1}$。
首次运行指令	置位InputBit $_{n-1}$。	置位InputBit $_{n-1}$。
EnableIn是清零状态	清零EnableOut，指令不执行，输出不更新。	无影响
EnableIn是置位状态	在EnableIn由清零状态转换为置位状态时，指令置位InputBit $_{n-1}$。 执行指令。 置位EnableOut。	在EnableIn由清零状态转换为置位状态时，指令置位InputBit $_{n-1}$。 EnableIn总是置位状态。 执行指令。
后期扫描	不动作。	不动作。

举例：当limit_switch1由清零状态变为置位状态时，OSRI指令置位OutputBit一个扫描周期。

结构文本

```
OSRI_01.InputBit := limit_switch1;

OSRI(OSRI_01);

State := OSRI_01.OutputBit;
```

功能块

带输入的下降沿触发 (OSFI)

当输入位从置位状态变为清零状态时，OSFI指令置位输出位一个执行周期。

该指令相当于梯形图编程中的OSF指令，参见1-17页。

操作数：

结构文本

操作数：	数据类型：	格式：	说明：
OSFI标签	FBD_ONESHOT	结构体	OSFI结构体

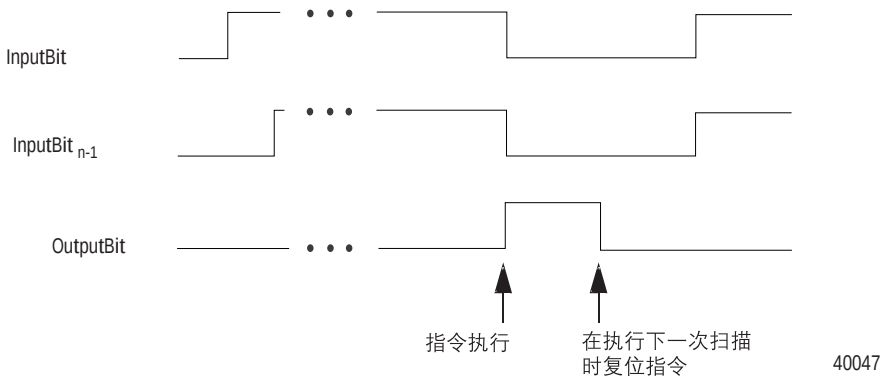
功能块

操作数：	数据类型：	格式：	说明：
OSFI标签	FBD_ONESHOT	结构体	OSFI结构体

FBD_ONESHOT结构体

输入参数：	数据类型：	说明：
EnableIn	BOOL	**功能块**： 如果被清零，则不执行指令而且也不更新输出。 如果被置位，则执行指令。 缺省状态是置位。 **结构文本**： 无影响。执行指令。
InputBit	BOOL	输入位。与用梯形图编程的OSF指令的梯级条件相同。 缺省状态是清零。
输出参数：	**数据类型：**	**说明：**
EnableOut	BOOL	指令产生一有效结果。
OutputBit	BOOL	输出位。

说明：当InputBit是清零状态而InputBit $_{n-1}$ 是置位状态时，OSFI指令置位OutputBit。
当InputBit $_{n-1}$ 是清零状态或InputBit是置位状态，OSFI指令清零OutputBit。



算术状态标志：不影响

故障条件：无

执行：

条件：	**功能块动作**：	**结构文本动作**：
预扫描	不动作。	不动作。
首次扫描指令	清零InputBit $_{n-1}$。	清零InputBit $_{n-1}$。
首次运行指令	清零InputBit $_{n-1}$。	清零InputBit $_{n-1}$。
EnableIn是清零状态	清零EnableOut，指令不执行，输出不更新。	无影响
EnableIn是置位状态	在InputBit由清零状态转换为置位状态时， 指令清零InputBit $_{n-1}$。 执行指令。 置位EnableOut。	在InputBit由清零状态转换为置位状态时， 指令清零InputBit $_{n-1}$。 总是置位EnableIn。 执行指令。
后期扫描	不动作。	不动作。

举例：当limit_switch1由置位状态变为清零状态时，OSFI指令置位OutputBit一个扫描周期。

结构文本

```
OSFI_01.InputBit := limit_switch1;

OSFI(OSFI_01);

Output_state := OSFI_01.OutputBit;
```

功能块

计时器和计数器指令
(TON，TOF，RTO，TONR，TOFR，RTOR，CTU，CTD，CTUD，RES)

简介

计时器和计数器指令根据触发事件发生的时间或次数来控制操作。

如果用户需要:	所用指令:	有效语言:	参见页码:
记录使能计时器指令的时间	TON	梯形图	2-2
记录禁止计时器指令的时间	TOF	梯形图	2-6
记录时间的累加值	RTO	梯形图	2-10
记录使能计时器指令的时间且带有内置复位按钮的功能块	TONR	结构文本 功能块	2-14
记录禁止计时器指令的时间且带有内置复位按钮的功能块	TOFR	结构文本 功能块	2-17
记录时间累加值且带有内置复位按钮的功能块	RTOR	结构文本 功能块	2-20
加计数	CTU	梯形图	2-24
减计数	CTD	梯形图	2-28
在功能块中加计数和减计数	CTUD	结构文本 功能块	2-32
复位计时器或计数器	RES	梯形图	2-36

所有计时器的时间基都是1 毫秒。

延时导通计时器 (TON)

TON 指令是非保持型计时器指令，从指令被使能(梯级输入条件为真)时开始累计时间。

该指令相当于结构文本和功能块中的TONR指令，参见2-14页。

操作数:

梯形图

操作数:	数据类型:	格式:	说明:
Timer	TIMER	标签	计时器—计时器结构体
Preset	DINT	立即数	预置值—需延迟的时间(累计的时间)
Accum	DINT	立即数	累加值—计时器以毫秒为单位已经计数的时间值，典型初始值为0

计时器结构体

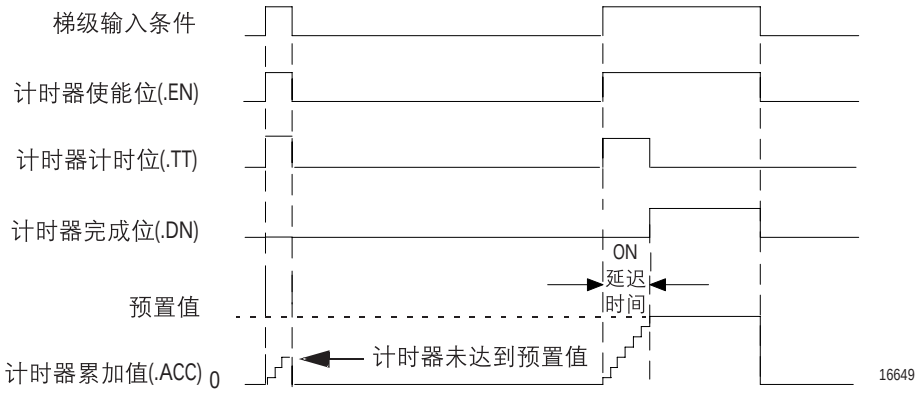
助记符:	数据类型:	说明:
.EN	BOOL	使能位—表明TON指令被使能。
.TT	BOOL	计时位—表明计时器正在计时。
.DN	BOOL	完成位—当累加值.ACC≥ 预置值.PRE.时置位该位。
.PRE	DINT	预置值—指定在指令置位完成位.DN之前累加值要达到的数值(以1毫秒为单位)。
.ACC	DINT	累加值—表示从TON指令被使能开始已经经过的毫秒数。

说明: TON 指令开始计时直到:

- TON 指令被禁止
- 累加值 .ACC ≥预置值.PRE

计时器指令的时间基总是 1 毫秒。例如，一个 2秒的计时器，预置值.PRE 要输入2000。

如果TON 指令被禁止，则清零累加值(.ACC)。



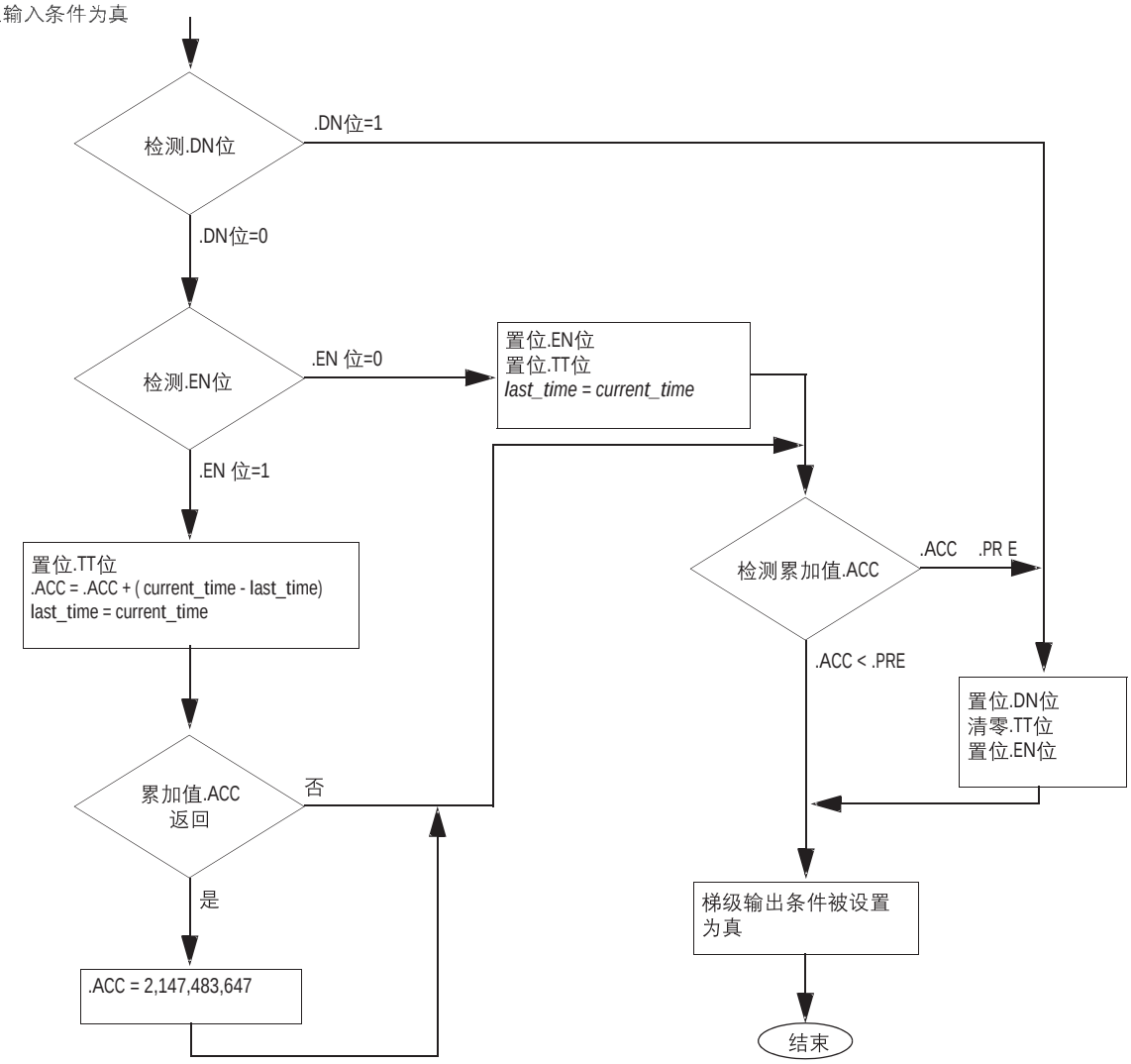
算术状态标志：不影响

故障条件：

发生主要故障的条件：	故障类型：	故障代码：
.PRE < 0	4	34
.ACC < 0	4	34

执行:

条件:	梯形图动作:
预扫描	清零使能位 .EN, 计时位 .TT, 完成位 .DN 。 清零累加值 .ACC 。 梯级输出条件被设置为假。
梯级输入条件为假	清零使能位 .EN, 计时位 .TT, 完成位 .DN 。 清零累加值 .ACC 。 梯级输出条件被设置为假。
梯级输入条件为真	



后期扫描	梯级输出条件被设置为假。

举例：如果limit_switch_1 是置位状态， 则light_2 打开180毫秒(timer_1计时)。如果timer_1.acc达到180，则light_2 关断同时接通light_3。在TON指令被禁止之前Light_3 一直保持接通状态。如果在timer_1计时期间 limit_switch_1 被清零，则关断light_2 。

延时断开计时器 (TOF)

TOF指令是非保持型计时器指令，当指令被使能时(梯级输入条件为假)开始累加时间。

该指令相当于结构文本和功能块编程中的TOFR指令，参见2-17页。

操作数:

梯形图

操作数:	数据类型:	格式:	说明:
Timer	TIMER	标签	计时器—计时器结构体
Preset	DINT	立即数	预置值—需延迟的时间(要累计的时间)
Accum	DINT	立即数	累加值—计时器以毫秒为单位已经计数的时间值，典型初始值为0

计时器结构体

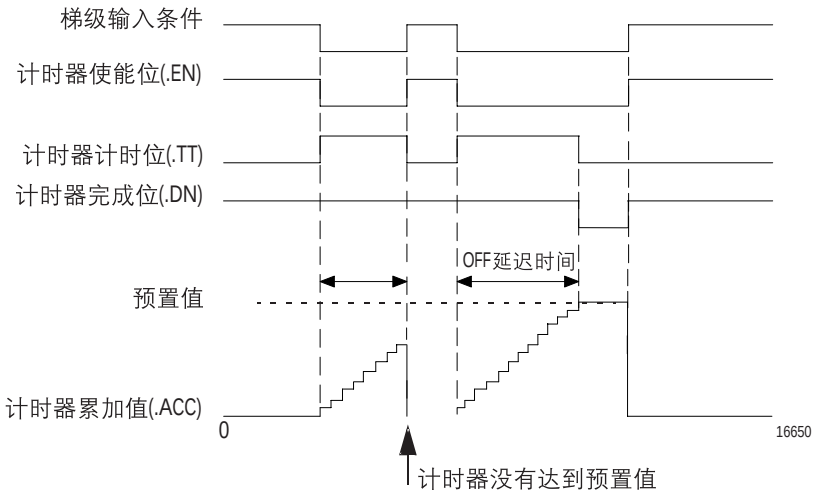
助记符:	数据类型:	说明:
.EN	BOOL	使能位—表明TOF指令被使能。
.TT	BOOL	计时位——表明计时器正在计时。
.DN	BOOL	完成位—当累加值.ACC ⩾预置值.PRE时置位该位。
.PRE	DINT	预置值—指定在指令置位完成位.DN之前累加值要达到的数值(以1毫秒为单位)。
.ACC	DINT	累加值—表示从TOF指令被使能开始已经经过的毫秒数。

说明: TOF 指令开始计时直到：

- TOF指令被禁止
- 累加值.ACC ⩾预置值.PRE

该指令的时间基总是1毫秒。例如，一个2秒的计时器，预置值.PRE 要输入2000。

如果TOF 指令被禁止，则清零累加值(.ACC)。

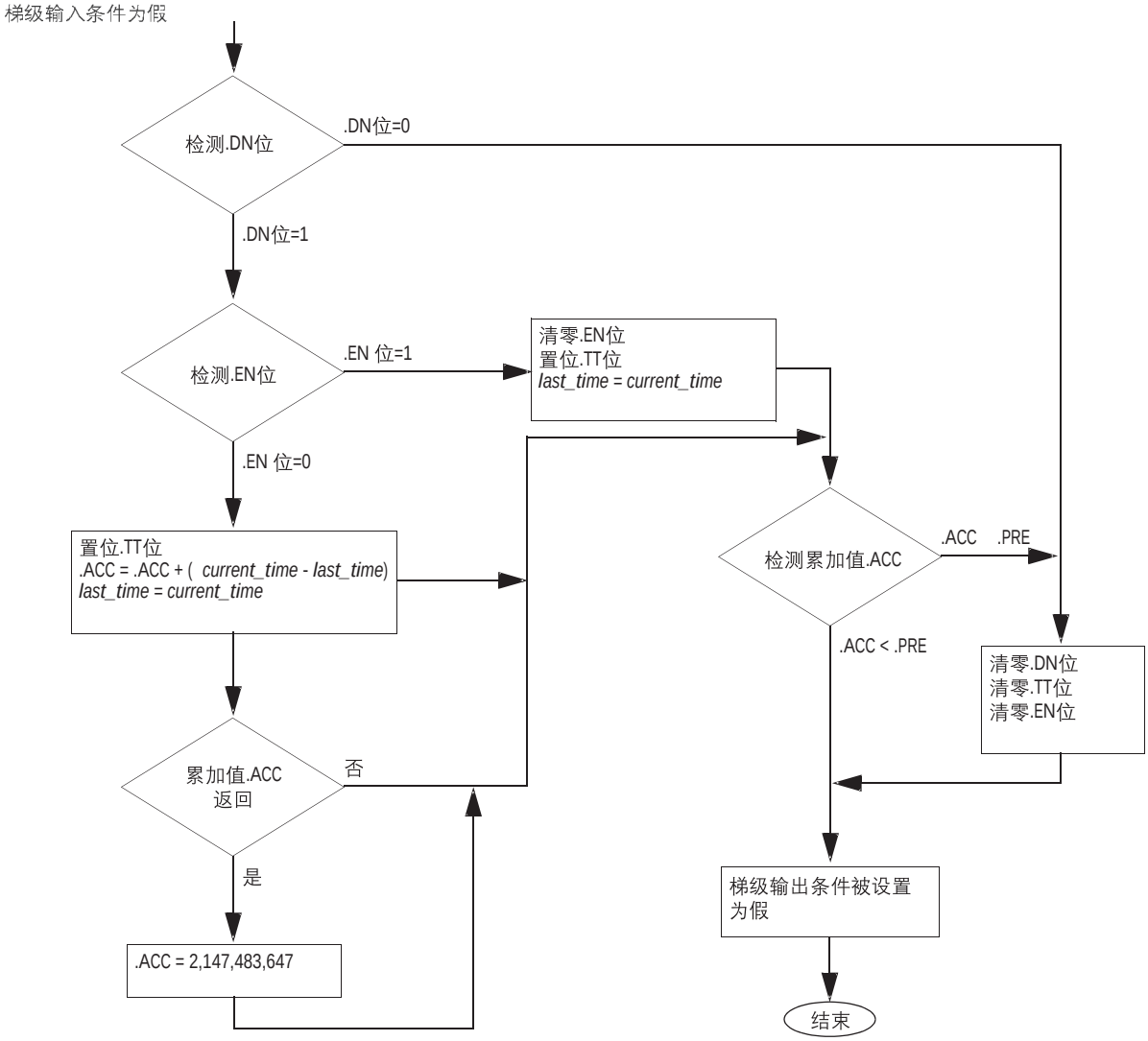


算术状态标志：不影响

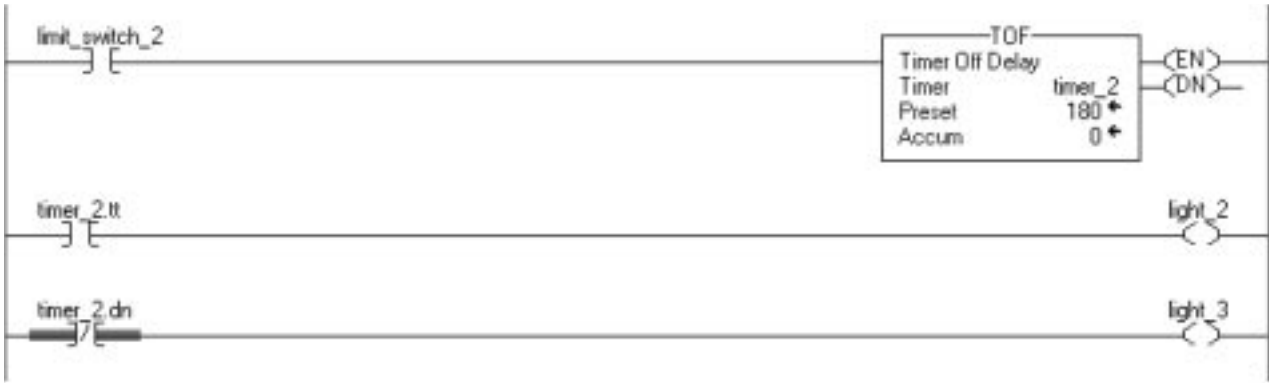
故障条件：

发生主要故障的条件：	故障类型：	故障代码：
.PRE < 0	4	34
.ACC < 0	4	34

执行:

条件:	梯形图动作:
预扫描	清零使能位.EN，计时位.TT，完成位.DN。 设置累加值.ACC等于预置值.PRE。 梯级输出条件被设置为假。
梯级输入条件为假	
梯级输入条件为真	置位使能位.EN，计时位.TT，完成位.DN。 清零累加值.ACC 。 梯级输出条件被设置为真。
后期扫描	梯级输出条件被设置为假。

举例：如果limit_switch_2 被清零， 则light_2 接通180 毫秒(timer_2 计时)。当timer_2.acc达到180时，关断light_2同时接通light_3。在TOF指令被使能之前，light_3 一直保持接通状态。如果在timer_2计时期间 limit_switch_2 被置位，则关断light_2 。



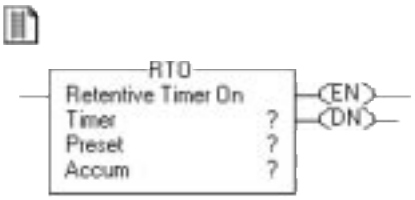
保持型延时导通计时器 (RTO)

RTO 指令时一条保持型计时器指令，从指令被使能时开始累加时间。

该指令相当于结构文本和功能块编程中的RTOR指令，参见2-20页。

操作数：

梯形图



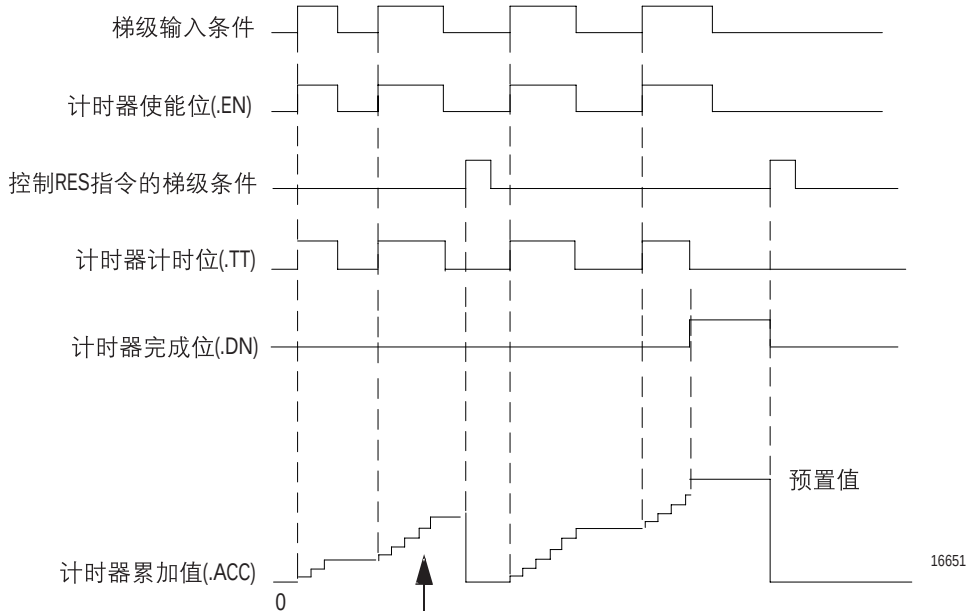
操作数：	数据类型：	格式：	说明：
Timer	TIMER	标签	计时器—计时器结构体
Preset	DINT	立即数	预置值—需延迟的时间(要累计的时间)
Accum	DINT	立即数	累加值—计时器以毫秒为单位已经计数的时间值，典型初始值输入0

计时器结构体

助记符：	数据类型：	说明：
.EN	BOOL	使能位—表明RTO指令被使能。
.TT	BOOL	计时位—表明计时器正在计时。
.DN	BOOL	完成位—当累加值.ACC ≥ 预置值.PRE.时置位该位。
.PRE	DINT	预置值—指定在指令置位完成位.DN之前累加值要达到的数值(以1毫秒为单位)。
.ACC	DINT	累加值—表示从RTO指令被使能开始已经经过的毫秒数。

说明: RTO 指令开始计时直至被禁止。当RTO指令被禁止时仍保持它的累加值.ACC。用户必须在程序的其它位置清零累加值.ACC，一般用RES指令来完成。该指令所用的结构体与TIMER相同。

计时器指令的时间基总是 1 毫秒。例如，一个2秒的计时器，预置值.PRE 要输入2000。



计时器没有达到预置值

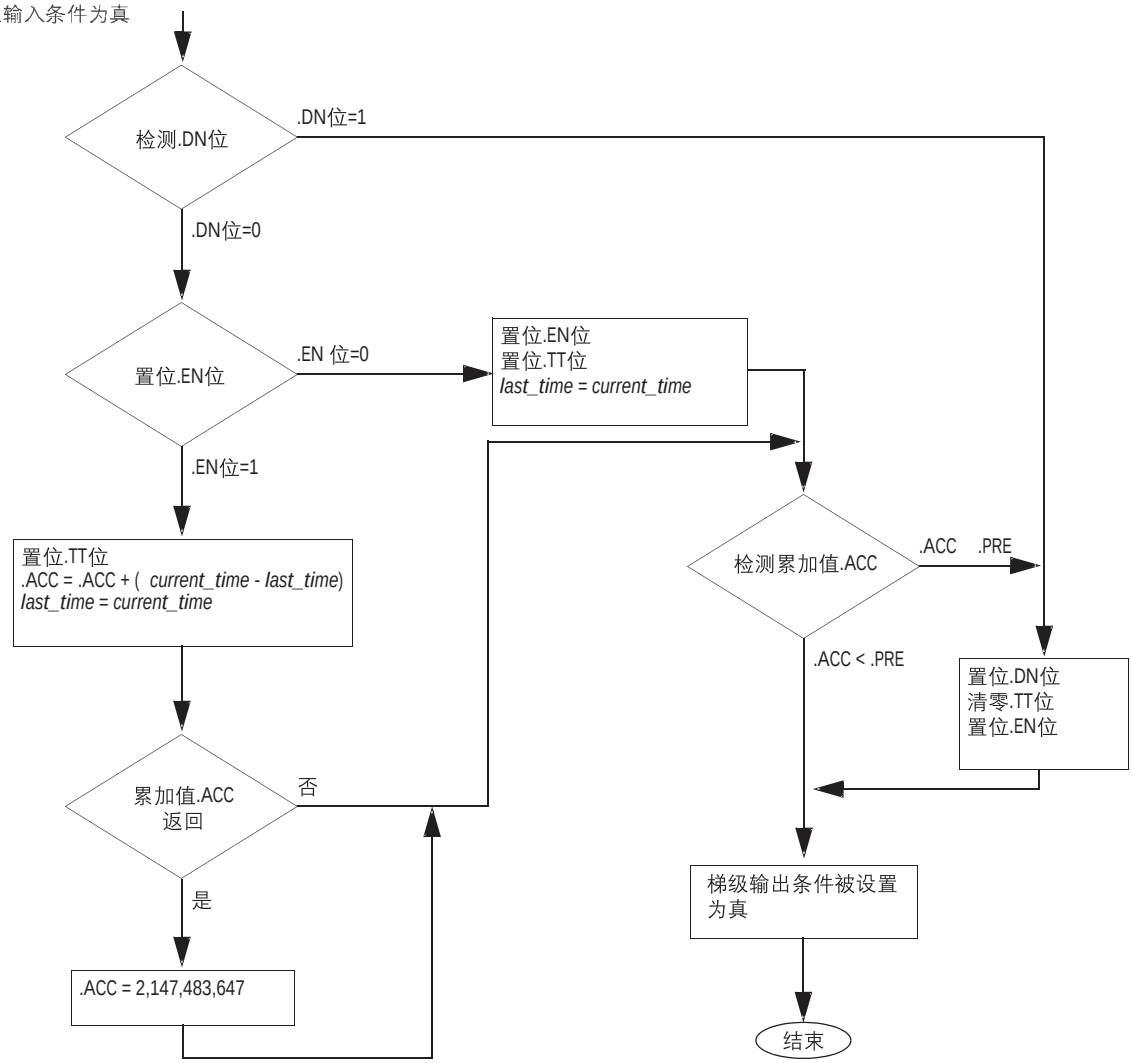
算术状态标志：不影响

故障条件：

发生主要故障的条件：	故障类型：	故障代码：
.PRE < 0	4	34
.ACC < 0	4	34

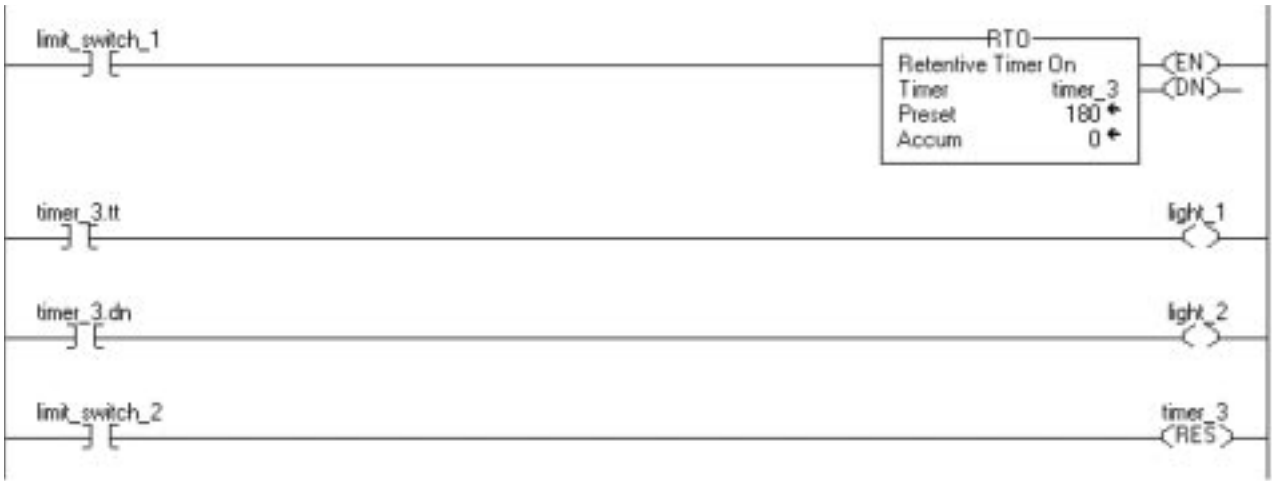
执行:

条件:	梯形图动作:
预扫描	清零使能位.EN，计时位.TT，完成位.DN。 累加值.ACC保持不变。 梯级输出条件被设置为假。
梯级输入条件为假	清零使能位.EN，计时位.TT。 完成位.DN保持不变。 累加值.ACC保持不变 。 梯级输出条件被设置为假。
梯级输入条件为真	



后期扫描	梯级输出条件被设置为假。

举例：如果limit_switch_1 是置位状态，则light_1 打开180毫秒(timer_2计时)。如果timer_3.acc达到180，则关断light_1同时接通light_2。Light_2 将保持接通直到timer_3 被复位。如果在timer_3计时期间 limit_switch_2 是清零状态，light_1 保持接通状态。在limit_switch_2置位时， RES指令复位timer_3 (清零状态位和累加值.ACC)。



带复位的延时导通计时器 (TONR)

TONR 指令是一条非保持型计时器指令，当TimerEnable被置位时开始累计时间。

该指令相当于梯形图编程中两条独立指令：TON (参见 2-2页)和RES (参见 2-36页)。

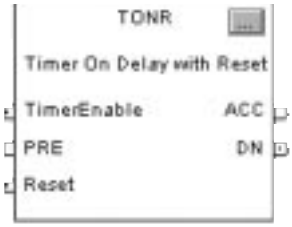
操作数：

```
TONR(TONR_tag);
```

结构文本

变量：	数据类型：	格式：	说明：
TONR 标签	FBD_TIMER	结构体	TONR结构体

功能块



操作数：	数据类型：	格式：	说明：
TONR 标签	FBD_TIMER	结构体	TONR结构体

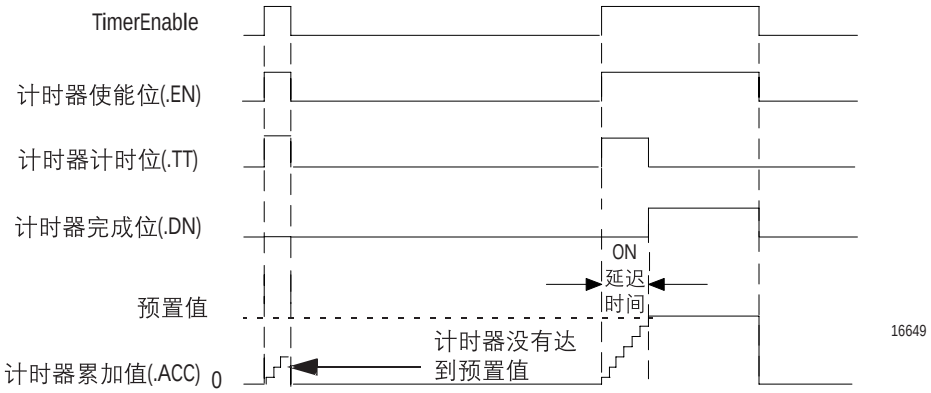
FBD_TIMER结构体

输入参数：	数据类型：	说明：
EnableIn	BOOL	**功能块**： 如果被清零，则指令不执行输出也不更新。 如果被置位，指令执行。 缺省状态是置位。 **结构文本**： 无影响。指令执行。
TimerEnable	BOOL	如果置位，则使计时器进入运行状态并累计时间。 缺省状态是清零。
PRE	DINT	计时器预置值。该值以1 毫秒为单位，在计时完成之前累加值ACC 必须达到这个值。如果为无效值，则指令置位状态字中的相应位，同时计时器不工作。 有效值= 0 到最大正整数
Reset	BOOL	请求复位计时器。当置位时，计时器被复位。 缺省状态是清零。
输出参数：	**数据类型：**	**说明：**
EnableOut	BOOL	指令产生一有效结果。
ACC	BOOL	以毫秒为单位的时间累加值。
EN	BOOL	计时器使能位输出。表明计时器指令被使能。
TT	BOOL	计时器计时位输出。当置位时，表明计时器正在计时。
DN	BOOL	计时器完成位输出。表示累加值大于或等于预置值。
Status	DINT	功能块的状态。
InstructFault (Status.0)	BOOL	指令检测下列运行错误之一。 这里不是指控制器的次要或主要错误。而是检查其余状态位以确定发生错误的原因。
PresetInv (Status.1)	BOOL	预置值无效。

说明: TONR指令计时直到：

- TONR 指令被禁止
- 累加值ACC ≥ 预置值PRE

计时器指令的时间基总是 1 毫秒。例如，一个 2秒的计时器，预置值.PRE要输入2000。



可以通过置位复位输入参数来复位指令。如果在复位位被置位时TimerEnable也是置位状态，则TONR指令在复位位被清零时重新开始计时。

算术状态标志： 不影响

故障条件： 无

执行：

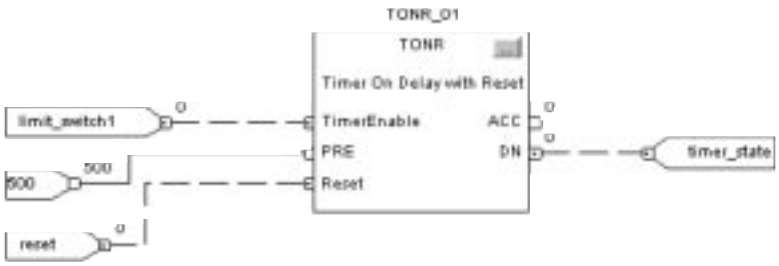
条件：	功能块动作：	结构文本动作：
预扫描	不动作。	不动作。
首次扫描指令	清零使能位.EN，计时位.TT 和完成位.DN。 累加值ACC被设置为0。	清零使能位.EN，计时位.TT 和完成位.DN。 累加值ACC被设置为0。
首次运行指令	清零使能位.EN，计时位.TT 和完成位.DN。 累加值ACC 值被设置为0。	清零使能位.EN，计时位.TT 和完成位.DN。ACC 值被设置为0。 累加值ACC 值被设置为0。
EnableIn 被清零	清零EnableOut，指令不执行，输出不更新。	无影响
EnableIn被置位	当EnableIn 从清零变为置位状态时，指令按照首次扫描指令的描述来初始化指令。 执行指令。 置位EnableOut。	EnableIn一直置位。 执行指令。
复位	当Reset输入参数置位时，指令清零使能位.EN，计时位.TT 和完成.DN 。 累加值ACC =0。	当Reset输入参数置位时，指令清零使能位.EN，计时位.TT 和完成位.DN 。 累加值ACC =0。
后期扫描	不动作。	不动作。

举例：每次扫描limit_switch1是置位状态时，TONR指令一直通过增加累加值ACC来记录经过的时间，直到累加值达到预置值PRE。当累加值 ACC ≥预置值≥ PRE时，置位完成位.DN和timer_state。

结构文本

```
TONR_01.Preset := 500;
TONR_01.Reset : = reset;
TONR_O1.TimerEnable := limit_switch1;

TONR(TONR_01);

timer_state := TONR_01.DN;
```

功能块举例：



带复位的延时断开计时器 (TOFR)

TOFR 指令是一条非保持计时器指令，当TimerEnable被清零时开始累计时间。

该指令相当于梯形图编程中两条独立指令：TOF(参见 2-6 页)和RES (参见 2-36页)。

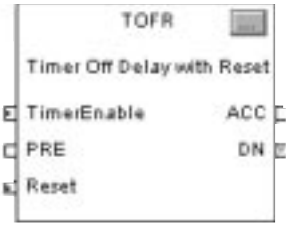
操作数：

TOFR(TOFR_tag);

结构文本

变量：	数据类型：	格式：	说明：
TOFR标签	FBD_TIMER	结构体	TOFR结构体

功能块操作数



操作数：	数据类型：	格式：	说明：
TOFR标签	FBD_TIMER	结构体	TOFR结构体

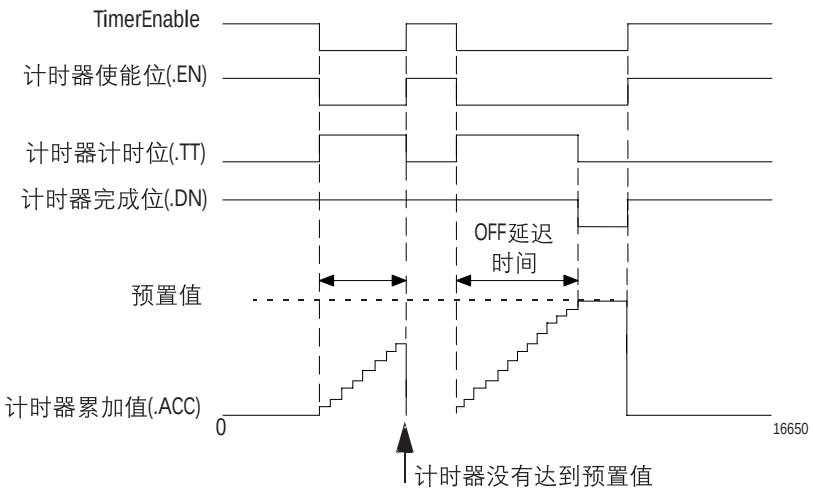
FBD_TIMER结构体

输入参数：	数据类型：	说明：
EnableIn	BOOL	**功能块：** 如果被清零，则指令不执行而且输出也不更新。 如果被置位，指令执行。 缺省状态是置位。 **结构文本：** 无影响。指令执行。
TimerEnable	BOOL	如果是清零状态，则使计时器进入运行状态并累计时间。 缺省状态是清零。
PRE	DINT	计时器预置值。该值以1毫秒为单位，在计时完成前累加值ACC 必须达到该值。如果该值无效，则指令置位状态字中的相应位，同时计时器不执行。 有效值= 0到最大正整数
Reset	BOOL	请求复位计时器。当置位时，计时器被复位。 缺省状态是清零。
输出参数：	**数据类型：**	**说明：**
EnableOut	BOOL	指令产生一有效结果。
ACC	BOOL	以毫秒为单位的时间累加值。
EN	BOOL	计时器使能位输出。表明计时器指令被使能。
TT	BOOL	计时器计时位输出。当置位时，表示计时器正在计时。
DN	BOOL	计时器完成位输出。表明累加值大于或等于预置值。
Status	DINT	功能块的状态。
InstructFault (Status.0)	BOOL	指令检测下列运行错误之一，这里不是指控制器的次要或主要错误。而是检查其余状态位以确定发生错误的原因。
PresetInv (Status.1)	BOOL	预置值无效。

说明: TOFR指令开始计时直到：

- TOFR 指令被禁止
- 累加值ACC ⩾预置值PRE

计时器指令的时间基总是 1 毫秒。例如，一个2秒的计时器，预置值PRE 要输入2000。



可以通过置位复位输入参数来复位指令。如果在复位位被置位时TimerEnable也是清零状态，则TOFR指令在复位位被清零时才重新开始计时。

算术状态标志：不影响

故障条件：无

执行：

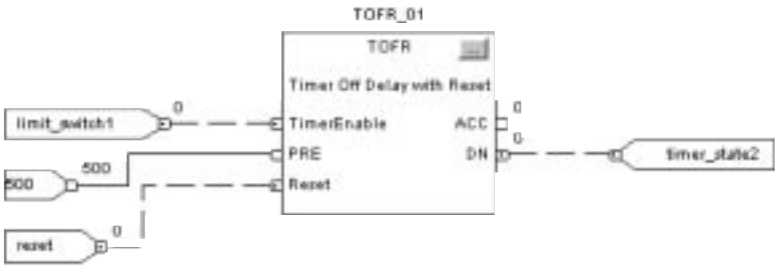
条件：	功能块动作：	结构文本动作：
预扫描	不动作。	不动作。
首次扫描指令	清零使能位.EN，计时位.TT 和 完成位.DN 。 设置累加值ACC等于预置值PRE。	清零使能位.EN，计时位.TT 和 完成位.DN 。 设置累加值ACC等于预置值PRE。
首次运行指令	清零使能位.EN，计时位.TT 和 完成位.DN 。 设置累加值ACC等于预置值PRE。	清零使能位.EN，计时位.TT 和 完成位.DN 。 设置累加值ACC等于预置值PRE。
EnableIn 被清零	清零EnableOut。	无影响
EnableIn被置位	当EnableIn 从清零变为置位时，指令按照首次扫描指令的描述来初始化指令。 执行指令。 置位EnableOut。	EnableIn总是被置位。 执行指令。
复位	当复位输入参数置位时，指令清零使能位.EN，计时位.TT 和完成位.DN 。并设置ACC = PRE。 注意这与用RES复位TOF指令不一样。	当复位输入参数置位时，指令清零使能位.EN，计时位.TT 和完成位.DN 。并设置ACC = PRE。 注意这与用RES复位TOF指令不一样。
后期扫描	不动作。	不动作。

举例：每次扫描limit_switch1是清零状态时，TOFR指令一直通过增加累加值ACC来记录经过的时间，直到累加值ACC达到预置值PRE 。当累加值 ACC≥预置值≥PRE时，清零完成位DN，置位timer_state2。

结构文本

```
TOFR_01.Preset := 500
TOFR_01.Reset := reset;
TOFR_O1.TimerEnable := limit_switch1;

TOFR(TOFR_01);

timer_state2 := TOFR_01.DN
```

功能块



带复位的保持型延时导通计时器(RTOR)

RTOR 指令是一条保持型计时器指令，当TimerEnable被置位时开始累计时间。

该指令相当于梯形图编程中两条独立的指令：RTO (参见 2-10页)和RES (参见 2-36页)。

操作数：

```
RTOR(RTOR_tag);
```

结构文本

变量：	数据类型：	格式：	说明：
RTOR 标签	FBD_TIMER	结构体	RTOR结构体

功能块操作数

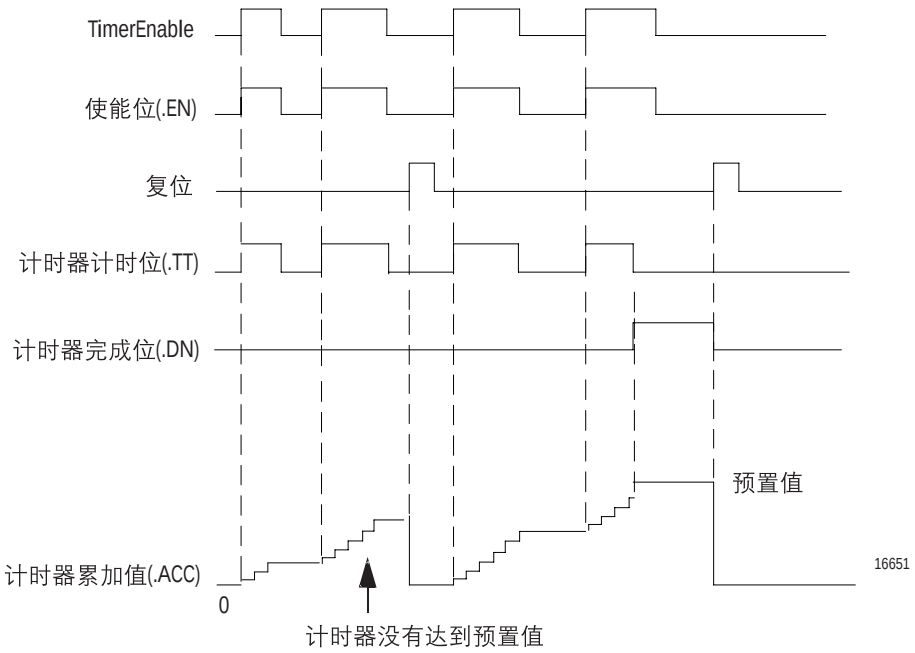
操作数：	数据类型：	格式：	说明：
RTOR 标签	FBD_TIMER	结构体	RTOR结构体

FBD_TIMER结构体

输入参数：	数据类型：	说明：
EnableIn	BOOL	**功能块：** 如果被清零，则指令不执行而且输出也不更新。 如果被置位，则指令执行。 缺省状态是置位。 **结构文本：** 无影响。指令执行。
TimerEnable	BOOL	如果置位，则使计时器进入运行状态并累计时间。 缺省状态是清零。
PRE	DINT	计时器预置值。该值以1毫秒位单位，在计时完成之前累加值ACC 必须达到这个值。如果是个无效值，则指令置位状态字中的相应位，同时计时器不工作。 有效值= 0 到最大正整数
Reset	BOOL	请求复位计时器。当置位时，计时器被复位。
输出参数：	**数据类型：**	**说明：**
EnableOut	BOOL	指令产生一有效结果。
ACC	BOOL	以毫秒为单位的时间累加值。即使TimerEnable输入被清零，该值仍保持不变。这是该功能块与TONR功能块的区别。
EN	BOOL	计时器使能位输出。表明计时器指令被使能。
TT	BOOL	计时器计时位输出。当置位时，表明计时器正在计时。
DN	BOOL	计时器完成位输出。表示累加值大于或等于预置值。
Status	DINT	功能块的状态。
InstructFault (Status.0)	BOOL	指令检测下列运行错误之一。 这里不是指控制器的次要或主要错误。而是检查其余状态位以确定发生错误的原因。
PresetInv (Status.1)	BOOL	预置值无效。

说明: RTOR 指令开始计时直至被禁止。RTOR指令被禁止时仍保持它的累加值.ACC。用户必须用复位输入来清零累加值.ACC。

计时器指令的时间基总是 1 毫秒。例如，一个2秒的计时器，预置值PRE 要输入2000。



可以通过置位复位输入参数来复位指令。如果在复位位被置位时TimerEnable也是置位状态，则RTOR指令在复位参数被清零时才重新开始计时。

算术状态标志： 不影响

故障条件： 无

执行：

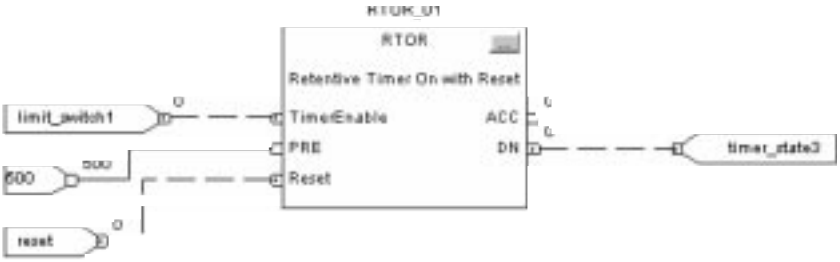
条件：	功能块动作：	结构文本动作：
预扫描	不动作。	不动作。
首次扫描指令	清零使能位.EN，计时位.TT 和完成位.DN 。 累加值ACC不变。	清零使能位.EN，计时位.TT 和完成位.DN 。 累加值ACC不变。
首次运行指令	清零使能位.EN，计时位.TT 和完成位.DN 。 累加值ACC不变。	清零使能位.EN，计时位.TT 和完成位.DN 。 累加值ACC不变。
EnableIn 被清零	清零EnableOut，指令不执行，输出不更新。	无影响
EnableIn被置位	**功能块**： 当EnableIn 从清零变为置位状态时，指令按照首次扫描指令的描述来初始化指令。 执行指令。 置位EnableOut。	EnableIn一直置位。 执行指令。
复位	当复位输入参数置位时，指令清零.EN，.TT 和.DN 位。并设置 ACC = 0。	当复位输入参数置位时，指令清零.EN，.TT 和.DN 位。 并设置 ACC = 0。
后期扫描	不动作。	不动作。

举例：每次扫描limit_switch1是置位状态时，RTOR指令通过增加累加值ACC来记录经过的时间，直到累加值ACC达到预置值PRE 。当累加值 ACC≥预置值≥PRE时，置位完成位DN和timer_state3。

结构文本

```
RTOR_01.Preset := 500
RTOR_01.Reset := reset;
RTOR_O1.TimerEnable := limit_switch1;

RTOR(RTOR_01);

timer_state3 := RTOR_01.DN;
```

功能块

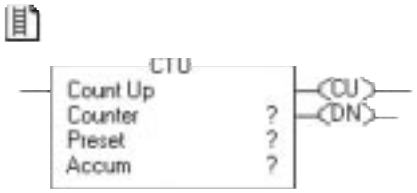


加计数 (CTU)

CTU指令用于向上计数。

该指令相当于结构文本和功能块编程中的CTUD指令，参见2-32页。

操作数：



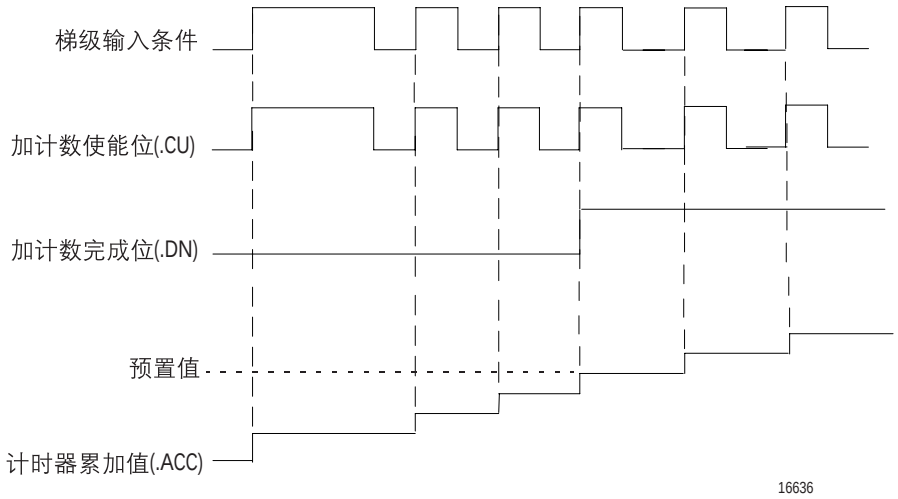
梯形图

操作数：	数据类型：	格式：	说明：
Counter	COUNTER	标签	计数器—计数器结构体
Preset	DINT	立即数	预置值—需要计数的数值
Accum	DINT	立即数	累加值—计数器已经计数的次数 初始值是0

计数器结构体

助记符：	数据类型：	说明：
.CU	BOOL	加计数使能位—表示CTU指令被使能。
.DN	BOOL	完成位—该位置位表示累加值.ACC ≥预置值.PRE。
.OV	BOOL	上溢出位—表示计数值超过上限值2，147，483，647。然后计数值返回到-2，147，483，648并重新开始向上计数。
.UN	BOOL	下溢出位—表示计数值超出下限值-2，147，483，648。计数值自动返回到2，147，483，647并重新开始向下计数。
.PRE	DINT	预置值—指定在指令置位完成位.DN之前累加值必须达到的数值。
.ACC	DINT	累加值—表示指令已经发生计数转换的次数。

说明: 如果指令被使能时，.CU位是清零状态，则CTU 指令使计数累加值加1。如果指令被使能时，使能位.CU是置位状态，或指令被禁止，则CTU指令保持累加值.ACC。



即使完成位.DN已经置位，累加值还继续上升。可以用计数器结构体的RES指令来清零累加值，或向累加值写入一个0值。

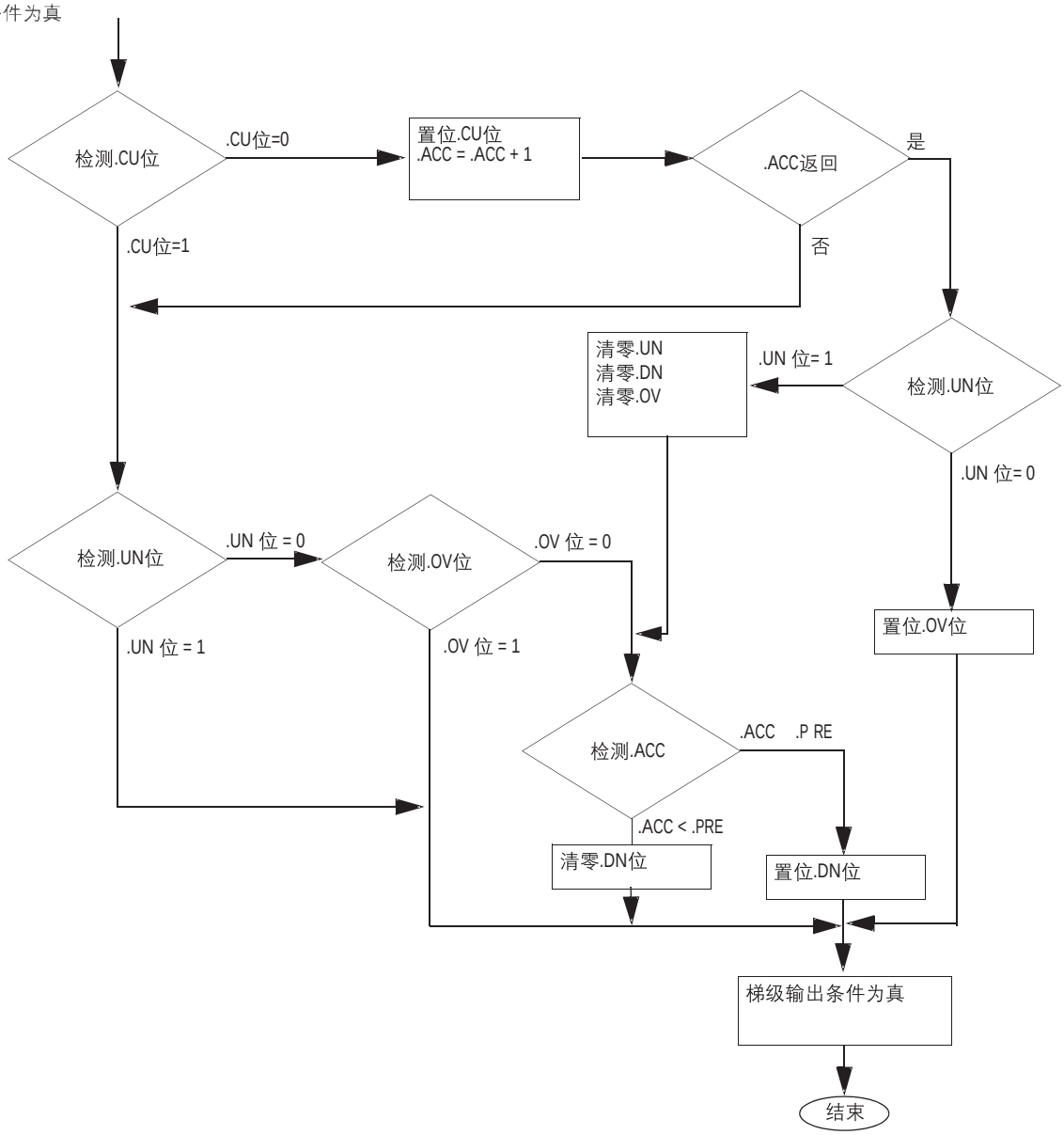
算术状态标志： 不影响

故障条件： 无

执行：

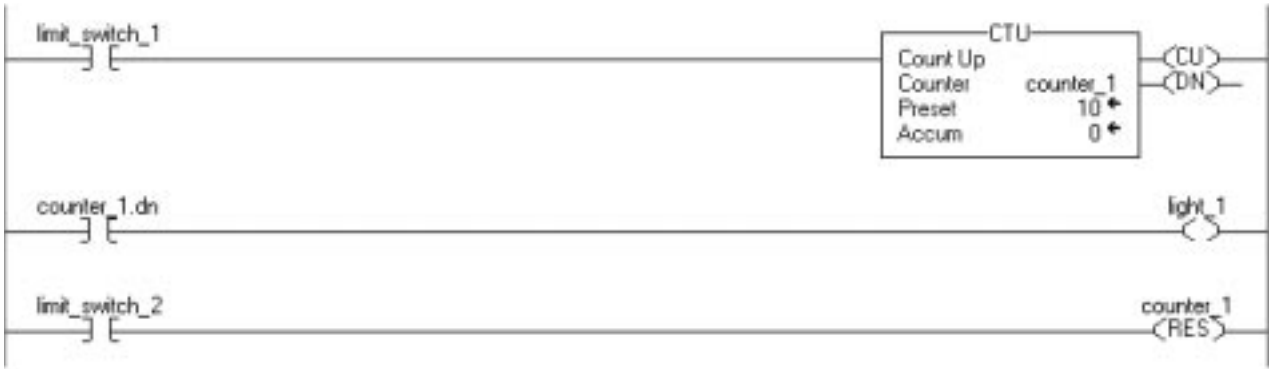
条件：	梯形图动作：
预扫描	置位使能位.CU以防止在首次扫描程序期间计数器计数无效。 梯级输出条件设置为假。
梯级输入条件为假	清零使能位.CU。 梯级输出条件设置为假。

梯级输入条件为真



后期扫描	梯级输出条件设置为假。

举例：在limit_switch_1从禁止变为使能10次后，置位完成位.DN 同时接通light_1。如果limit_switch_1继续由禁止变为使能，则counter_1继续增加计数值且完成位.DN保持置位状态。当limit_switch_2被使能时，RES指令复位counter_1(清零状态位和累加值.ACC)同时关断light_1。

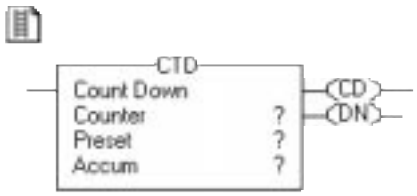


减计数指令 (CTD)

CTD指令用于向下计数。

该指令相当于结构文本和功能块编程中的CTUD指令，参见2-32页。

操作数：



梯形图

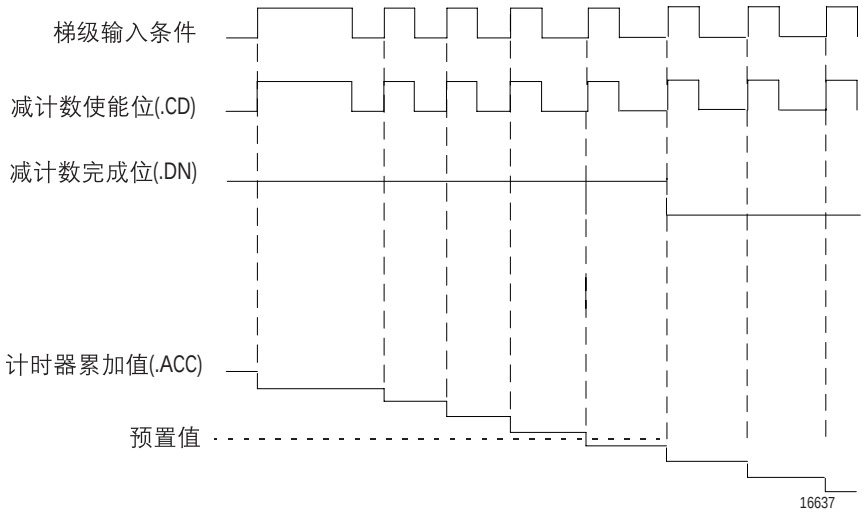
操作数：	数据类型：	格式：	说明：
Counter	COUNTER	标签	计数器—计数器结构体
Preset	DINT	立即数	预置值—需要计数的数值
Accum	DINT	立即数	累加值—计数器已经计数的次数 初始值是0

计数器结构体

助记符：	数据类型：	说明：
.CD	BOOL	减计数使能位—表示CTU指令被使能。
.DN	BOOL	完成位—表示累加值.ACC ≥预置值.PRE。
.OV	BOOL	上溢出位—表示计数值超过上限值2，147，483，647。然后计数值返回到-2，147，483，648并重新开始向上计数。
.UN	BOOL	下溢出位—表示计数值超出下限值-2，147，483，648。计数值自动返回到2，147，483，647并重新开始向下计数。
.PRE	DINT	预置值指定在指令置位完成位.DN之前累加值必须达到的数值。
.ACC	DINT	累加值表示指令已经发生的计数转换次数。

说明: CTD 指令通常与CTU指令一起使用，这时两条指令使用同一个计数器结构体。

如果指令被使能时，使能位.CD是清零状态，则CTD 指令使计数值减1。如果指令被使能时，使能位.CD是置位状态，或指令被禁止，则CTD指令保持累加值.ACC不变。

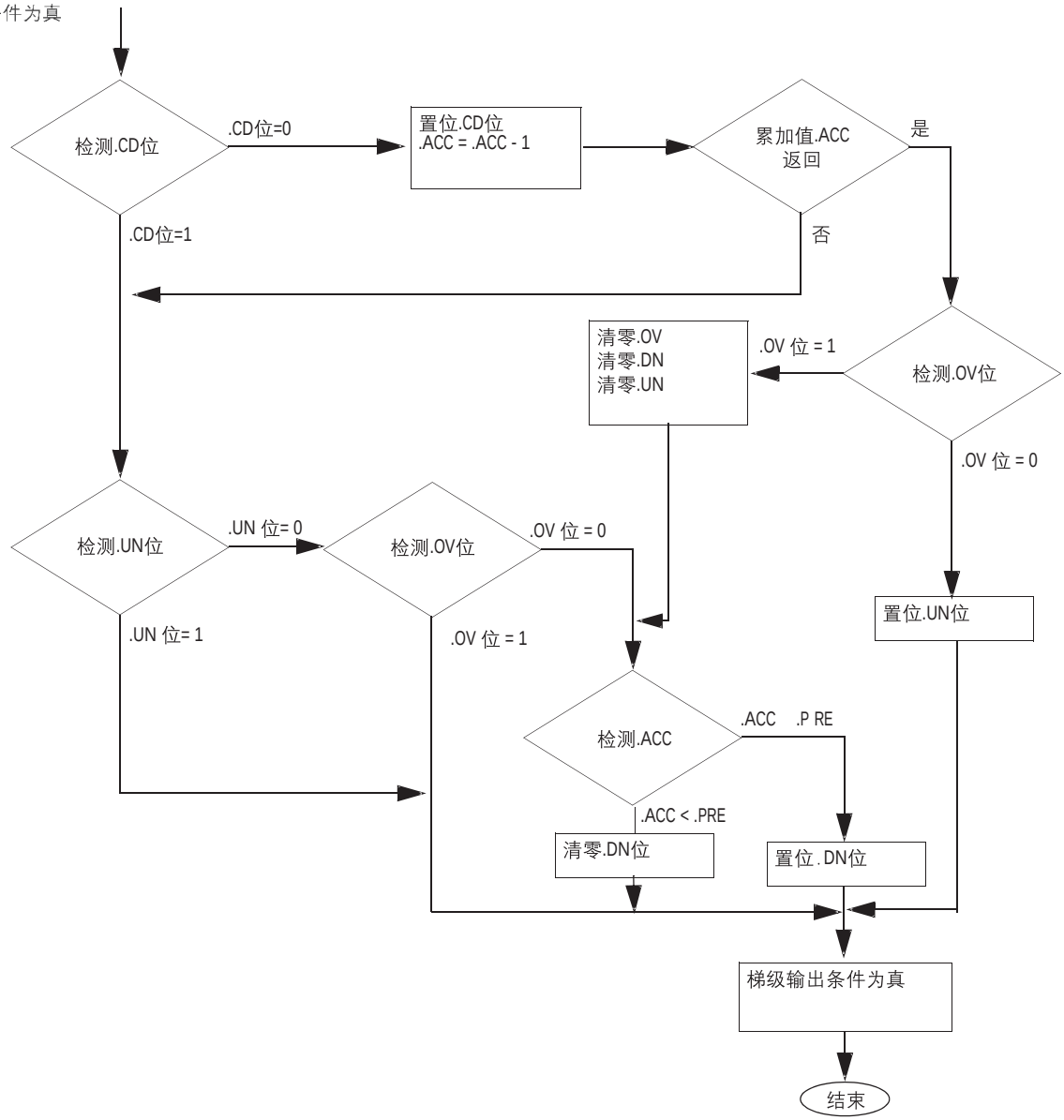


即使完成位.DN已经置位，累加值还继续减小。可以用使用同一计数器结构体的RES指令来清零累加值，或向累加值写入一个0值来清零累加值。

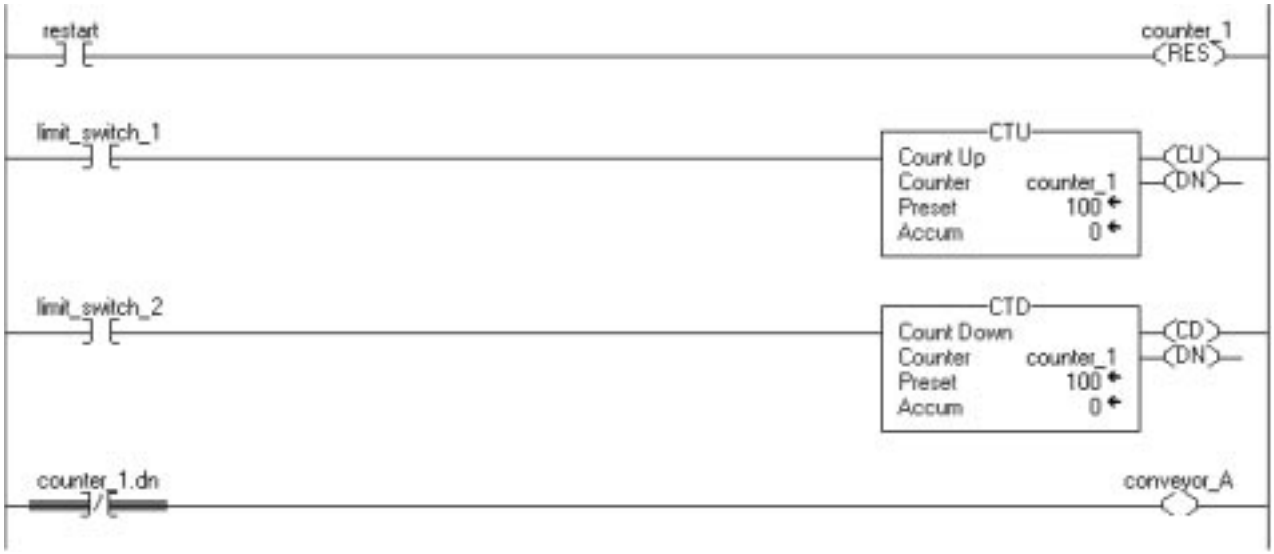
算术状态标志： 不影响

故障条件： 无

执行：

条件：	梯形图动作：
预扫描	置位.CD位以防止在首次扫描程序期间计数器计数无效。 梯级输出条件设置为假。
梯级输入条件为假	清零.CD位。 梯级输出条件设置为假。
梯级输入条件为真	
后期扫描	梯级输出条件设置为假。

举例：一条传送带将零件传送到缓存区，每传进一个零件，limit_switch_1就接通一次，计数器counter_1的累加值加1，每传出一个零件limit_switch_2 接通，计数器counter_1 的累加值减1。如果有100个零件进入缓存区（ counter_1.dn被置位)，则conveyor_a接通，同时使传送带停止传送更多的零件，除非缓存区再有空间存储零件。



加/减计数 (CTUD)

当CUEnable由清零状态变为置位状态时，CTUD指令累加值加1，当CDEnable由清零状态变为置位状态时，CTUD指令累加值减1。

该指令相当于梯形图编程中的三条独立指令：CTU (2-24页)， CTD (2-28页)，和RES (2-36页)。

操作数：

```
CTUD(CTUD_tag);
```

结构文本

变量：	数据类型：	格式：	说明：
CTUD标签	FBD_COUNTER	结构体	CTUD结构体

功能块

操作数：	数据类型：	格式：	说明：
CTUD标签	FBD_COUNTER	结构体	CTUD结构体

FBD_COUNTER结构体

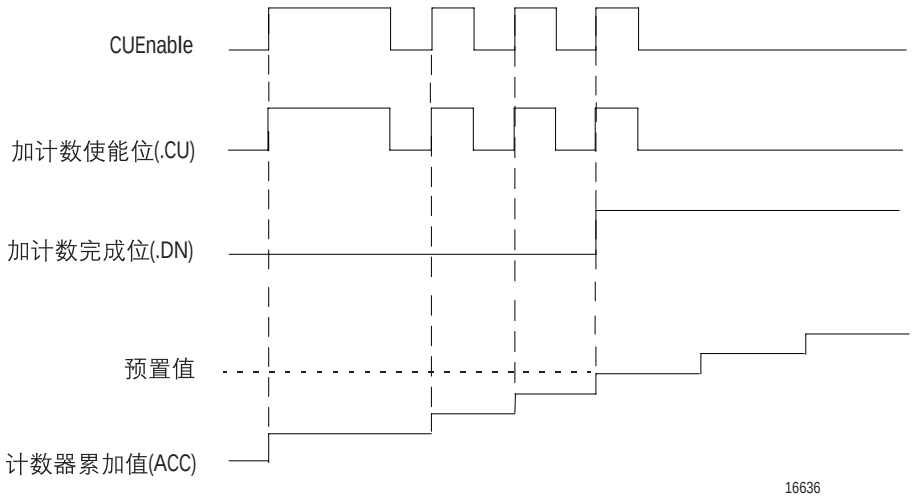
输入参数：	数据类型：	说明：
EnableIn	BOOL	功能块： 如果被清零，则指令不执行而且输出也不更新。 如果被置位，则指令执行。 缺省状态是置位。 结构文本： 无影响。指令执行。
CUEnable	BOOL	使能加计数功能。如果此输入由清零变为置位状态，则计数器累加值加1。 缺省状态是清零。
CDEnable	BOOL	使能减计数功能，如果此输入由清零变为置位状态，则计数器累加值减1。 缺省状态是清零。
PRE	DINT	计数器预置值，在计数器置位完成位DN之前，累加值必须达到该值。 有效值= 任意整数 缺省值是0
Reset	BOOL	请求复位计数器。当置位时，计数器被复位。 缺省状态是清零。

输出参数:	数据类型:	说明:
EnableOut	BOOL	指令产生有效结果。
ACC	DINT	累加值。
CU	BOOL	使能加计数功能。
CD	BOOL	使能减计数功能。
DN	BOOL	计数器完成位。当累加值大于或等于预置值时该位置位。
OV	BOOL	计数器上溢出位。表示累加值已经超出上限值2，147，483，647.， 计数器返回到-2，147，483，648重新开始向上计数。
UN	BOOL	计数器下溢出位。表示累加值已经超出下限值-2，147，483，648， 计数器返回到2，147，483，647重新开始向下计数。

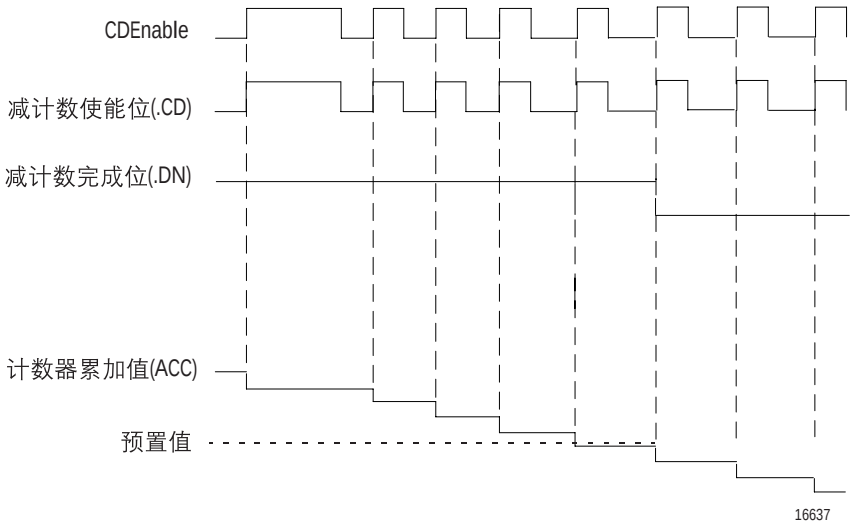
说明: 当指令被使能而且CUEnable是置位状态时， CTUD指令的累加值加1；当指令被使能而且CDEnable是置位状态时，CTUD指令累加值减1。

在同一个扫描周期内，输入参数CUEnable 和CDEnable可同时被触发。指令优先执行加计数。

加计数



减计数



当指令被禁止时，CTUD指令的累加值保持不变。可以通过置位FBD_COUNTER结构体的复位输入参数来复位指令。

算术状态标志：不影响

故障条件：无

执行：

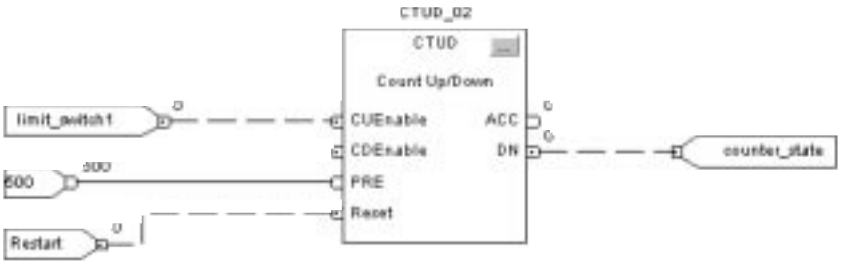
条件：	梯形图动作：	结构文本动作：
预扫描	无需初始化。	无需初始化。
首次扫描指令	置位CUEnable$_{n-1}$和CDEnable $_{n-1}$。	置位CUEnable$_{n-1}$和CDEnable $_{n-1}$。
首次运行指令	置位CUEnable$_{n-1}$和CDEnable $_{n-1}$。	置位CUEnable$_{n-1}$和CDEnable $_{n-1}$。
EnableIn被清零	清零EnableOut，指令不执行，输出不更新。	无影响
EnableIn被置位	指令置位CUEnable$_{n-1}$和CDEnable $_{n-1}$ 位。 在EnableIn位由清零变为置位状态时： • 执行指令。 • 置位EnableOut 。	指令置位CUEnable$_{n-1}$和 CDEnable $_{n-1}$ 位。 EnableIn—直置位。 执行指令。
复位	该位置位时，指令清零 CUEnable$_{n-1}$，CDEnable$_{n-1}$，CU，CD，DN， OV和UN位，并设置ACC = 0。	该位置位时，指令清零 CUEnable$_{n-1}$，CDEnable$_{n-1}$，CU，CD，DN， OV，和UN位，并设置ACC = 0。
后期扫描	不动作。	不动作。

举例：当limit_switch1 从清零状态转换到置位状态时，CUEnable 置位一个扫描周期而且CTUD指令的累加值ACC加1。当ACC ≥PRE时，置位完成位.DN，将使能CTUD指令后面的功能块指令。

结构文本

```
CTUD_01.Preset := 500;
CTUD_01.Reset := Restart;
CTUD_O1.CUEnable := limit_switch1;

CTUD(CTUD_01);

counter_state := CTUD_01.DN;
```

功能块



复位 (RES)

RES指令复位TIMER(计时器)， COUNTER(计数器)指令或CONTROL结构体。

操作数：

梯形图

操作数：	数据类型：	格式：	说明：
结构体	TIMER CONTROL COUNTER	标签	要复位的结构体

说明： 当使能RES指令时清零下列元素：

RES指令用于：	指令将清零：
TIMER	累加值.ACC 控制状态位
COUNTER	累加值.ACC 控制状态位
CONTROL	位置值.POS 控制状态位

注意：由于RES指令清零累加值.ACC，完成位.DN和计时位.TT，所以不能用RES指令复位TOF计时器指令。

算术状态标志： 不影响

故障条件： 无

执行：

条件：	梯形图动作：
预扫描	梯级输出条件被设置为假。
条件输入条件为假	梯级输出条件被设置为假。
条件输入条件为真	RES指令复位指定结构体。 梯级输出条件被设置为真。
后期扫描	梯级输出条件被设置为假。

举例：

举例：	说明：
	当指令被使能时复位timer_3。
	当指令被使能时复位counter_1。
	当指令被使能时复位control_1。

输入/输出指令
(MSG，GSV，SSV，IOT)

简介

输入/输出指令用于从控制器读数据，或向控制器写数据，或从其它网络上的模块读数据，或向该模块写数据。

如果用户需要：	所用指令：	可用语言：	参见页码：
与其它模块进行数据交换	MSG	梯形图 结构文本	3-2
获取控制器状态信息	GSV	梯形图 结构文本	3-34
设置控制器状态信息	SSV	梯形图 结构文本	3-34
• 在用户程序中指定位置发送输出值到I/O模块或消费者控制器 • 触发另一控制器的事件型任务	IOT	梯形图 结构文本	3-57

通讯指令(MSG)

MSG指令用于控制器与网络上的另一模块以异步读写数据块。

操作数：

梯形图

操作数：	数据类型：	格式：	说明：
Message control	MESSAGE	标签	message结构体

结构文本

```
MSG(MessageControl);
```

该操作数与梯形图中MSG指令的操作数相同。

MESSAGE结构体

助记符：	数据类型：	说明：
.FLAGS	INT	.FLAGS 项在一个16位字内存储下列状态位。 位： 状态项： 2 .EW 4 .ER 5 .DN 6 .ST 7 .EN 8 .TO 9 .EN_CC **重要事项**：如果在MSG指令使能期间复位任一MSG状态位都可能会导致通讯中断。
.ERR	INT	如果错误位.ER被置位，则此字表示MSG指令的错误代码。
.EXERR	INT	扩展错误代码字为一些错误代码指定附加错误信息。
.REQ_LEN	INT	请求数据长度，指定通讯指令尝试传送的信息字数。
.DN_LEN	INT	完成数据长度，表明实际传送的信息字数。
.EW	BOOL	当控制器检测到一个通讯请求已经进入队列时，置位使能等待位。当启动位.ST置位时，控制器复位使能等待位.EW。
.ER	BOOL	当控制器检测到传送失败时，错误位被置位。下次梯级输入条件由假变真时复位错误位.ER。
.DN	BOOL	当最后一个信息数据包成功传送之后，完成位被置位。下次梯级输入条件由假变真时复位完成位.DN。
.ST	BOOL	当控制器开始执行MSG指令时，置位启动位。在完成位.DN或错误位.ER被置位时复位启动位.ST。

助记符:	数据类型:	说明:
.EN	BOOL	当梯级输入条件变为真时，使能位被置位,且该位保持置位，直到完成位.DN或错误位.ER置位且梯级输入条件为假。如果在梯级输入条件为假时，完成位.DN 和错误位.ER 是清零状态，则使能位.EN保持置位状态。
.TO	BOOL	如果用户手动置位.TO 位，则控制器停止通讯过程并且置位错误位.ER。
.EN_CC	BOOL	使能缓存位确定如何管理MSG连接。参见3-31的 “选择缓存选项”。 即使.EN_CC位是置位状态，MSG 指令的连接不在串行口队列中也不被缓存。
.ERR_SRC	SINT	RSLogix 5000 编程软件在通讯配置对话框中显示错误路径
.DestinationLink	INT	用Source ID信息改变DH+ 或CIP 的目标连接，将该项设置成所需值。
.DestinationNote	INT	用Source ID信息改变DH+ 或CIP 的目标节点，将该项设置成所需值。
.SourceLink	INT	用Source ID信息改变DH+ 或CIP 的源连接，将该项设置成所需值。
.Class	INT	将该项设置成所需值，可以改变CIP通用信息的类参数。
.Attribute	INT	将该项设置成所需值，可以改变CIP通用信息的属性参数。
.Instance	DINT	将该项设置成所需值，可以改变CIP通用信息的实例参数。
.LocalIndex	DINT	如果用星号 [*] 来指明本地数组的元素号，则LocalIndex 提供单元号。可通过将该项设置成所需值来改变元素号。 **如果是:** / **则本地数组指明:** 读数据 / 目标单元 写数据 / 源单元
.Channel	SINT	如果要从1756-DHRIO模块的不同通道发送信息，则将该项设置成所需值。选择ASCII 字符A 或B。
.Rack	SINT	如果要改变块传送信息的机架号，则将该项设置成所需的机架号(八进制)。
.Group	SINT	如果要改变块传送信息的组号，则将该项设置成所需的组号(八进制)。
.Slot	SINT	如果要改变块传送信息的槽号，则将该项设置成所需的槽号。 **如果信息通过以下网络:** / **指定槽号的格式:** 通用远程I/O / 八进制 ControlNet / 十进制 (0-15)
.Path	STRING	如果要将信息发送到不同的控制器，则将该项设置为新路径。 • 用十六进制数值输入路径。 • 忽略逗号 [,] 例如，如果路径是1, 0, 2, 42, 1, 3, 则输入 $01$00$02$2A$01$03。 要浏览设备并自动创建部分或全部新字符串，右击字符串标签并选择转到信息路径编辑器。

助记符：	数据类型：	说明：	
.RemoteIndex	DINT	如果用星号 [*] 来指明远程数组的元素号，那么RemoteIndex 提供元素号。可通过将该项设置成所需值来改变元素号。	
		如果是：	则远程数组为：
		读数据	源元素
		写数据	目标元素
.RemoteElement	STRING	指定要发送的信息在控制器中的标签或地址，将该项设置成所需值。用ASCII 字符输入标签或地址。	
		如果是：	则远程数组为：
		读数据	源单元
		写数据	目标单元
.UnconnnectedTimeout	DINT	未连接通讯的超时时间。缺省值是30秒。	
.ConnectionRate	DINT	连接速率乘以超时因子产生通讯连接的超时时间 。 • 连接速率缺省值是7.5秒。 • 超时因子缺省值是 0 (等于一个乘法因子4)。 • 通讯连接超时的缺省时间是30秒 (7.5秒 x 4 = 30 秒)。 • 要改变超时时间，可以改变连接速率值或超时因子不为缺省值。	
.TimeoutMultiplier	SINT		

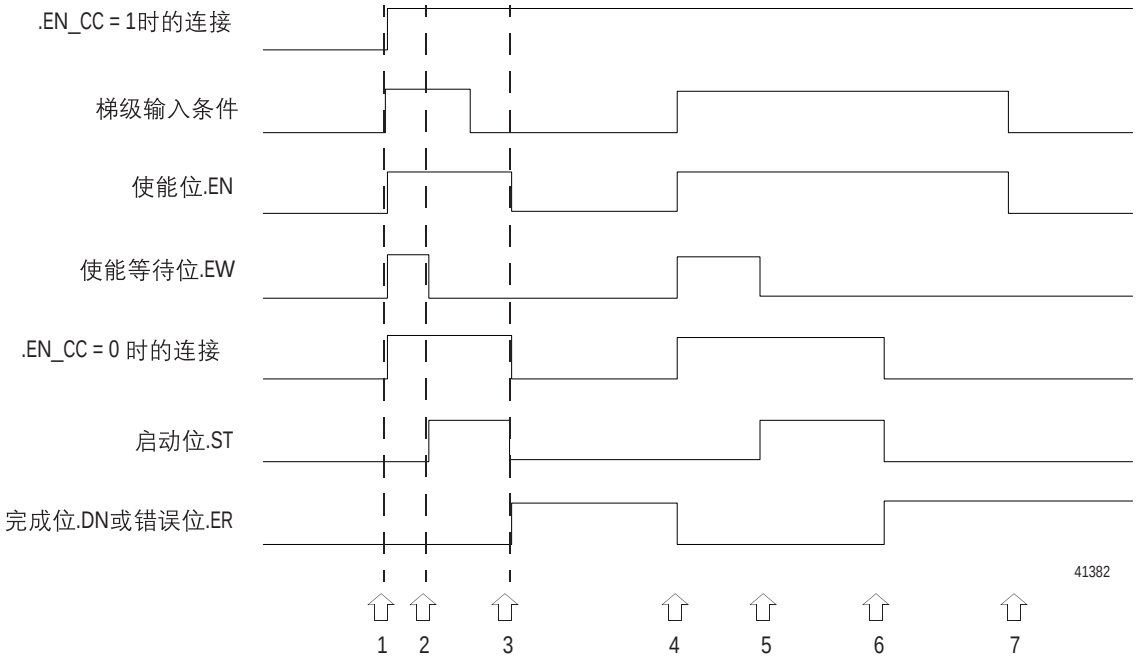
注 意: 控制器处理启动位.ST，使能等待位.EW，完成位.DN和错误位.ER异步于程序扫描。如果要在梯形图逻辑检测这些位，可以复制 .FLAGS字到一个INT标签，然后在该处检测这些位。否则，时间问题会使用户应用程序无效，可能会造成设备的损坏或人员伤亡。

说明： MSG指令传送数据元素。

该指令为边沿触发指令：

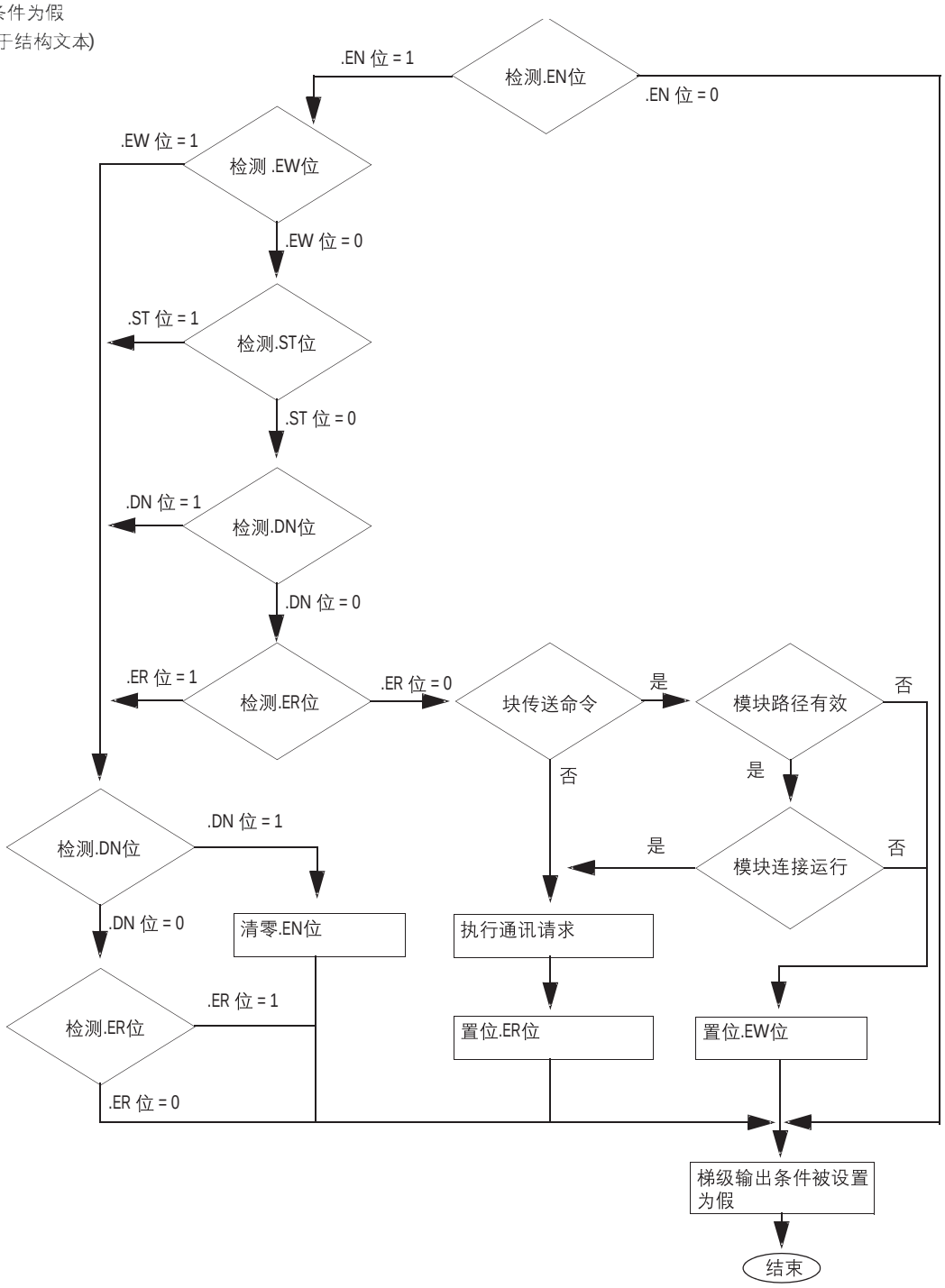
- 用梯形图编程，每次梯级输入条件由清零变为置位时，指令执行。
- 用结构文本编程，只在边沿触发时执行指令。参见附录C。

每个元素的大小与用户使用的数据类型和通讯命令的类型有关。



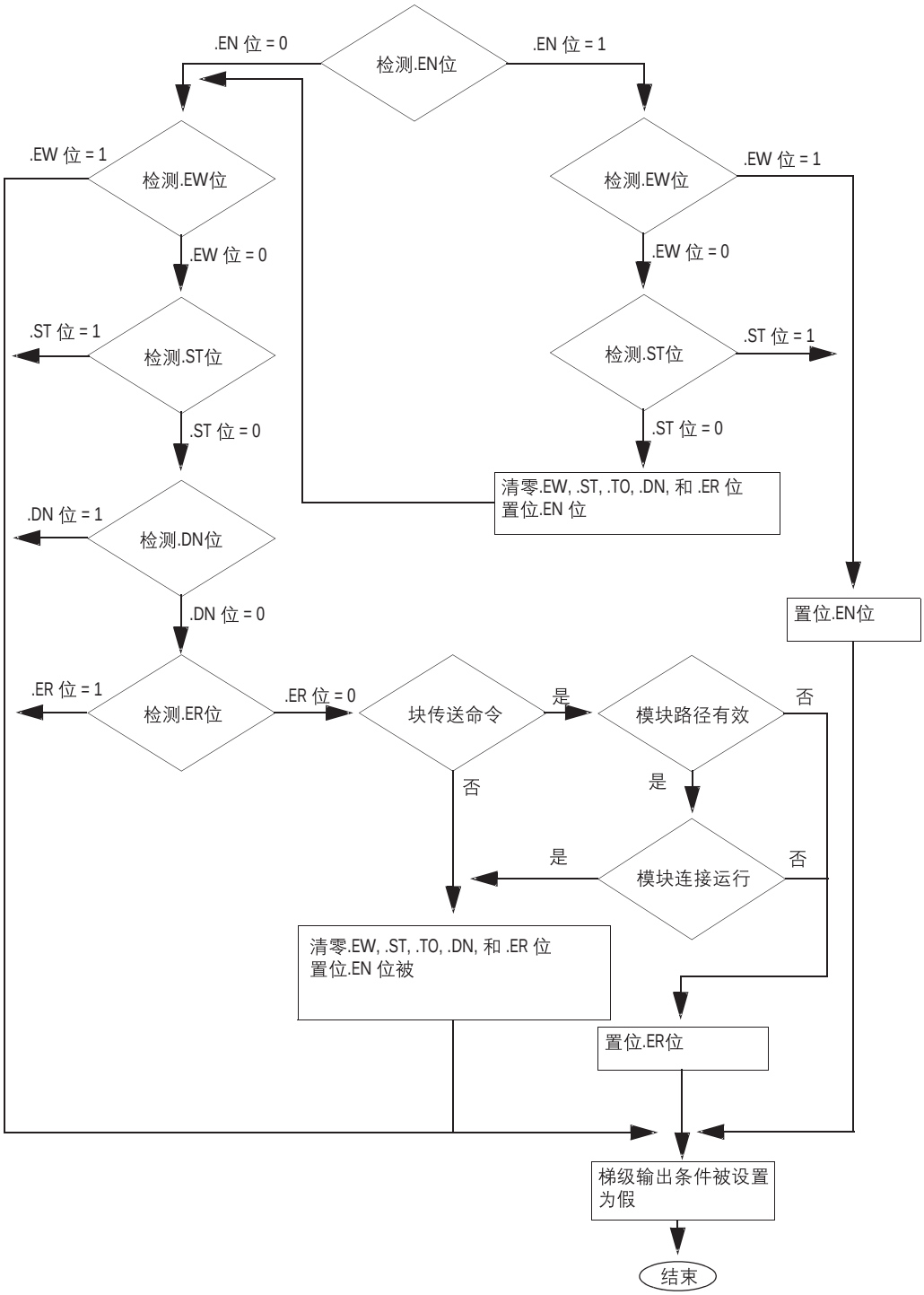
图中:	说明:	图中:	说明:
1	梯级输入条件为真 置位使能位.EN 置位使能等待位.EW 打开连接	5	发送信息 置位起动位.ST 清零使能等待位.EW
2	发送信息 置位启动位.ST 清零使能等待位.EW	6	通讯完成或出错 梯级输入条件仍为真 置位完成位.DN 或错误位.ER 清零起动位.ST 关闭连接(如果.EN_CC = 0)
3	通讯完成或出错 梯级输入条件为假 置位完成位.DN 或错误位.ER 起动位.ST 被清零 连接关闭(如果.EN_CC = 0) 清零使能位.EN(梯级输入条件为假)	7	梯级输入条件变为假而且置位完成位.DN 或错误位.ER 清零使能位.EN
4	梯级输入条件为真 完成位.DN 或错误位.ER 先前是置位状态 置位使能位.EN 置位使能等待位.EW 打开连接 清零完成位.DN 或错误位 .ER		

执行

条件：	梯形图动作：	结构文本动作：
预扫描	梯级输出条件设置为假。	不动作。
梯级输入条件为假 (不能应用于结构文本)		
梯级输入条件为真	指令执行。 梯级输出条件设置为假。	无影响

条件：	梯形图动作：	结构文本动作：
EnableIn被置位	无影响	EnableIn—直置位。 指令执行。

指令执行



后期扫描	梯级输出条件设置为假。	不动作。

算术状态标志: 不影响

故障条件: 无

MSG 错误代码

错误代码取决于MSG指令类型。

错误代码

RSLogix 5000编程软件无法显示所有说明。

错误代码: (十六进制)	说明:	软件显示:
1	连接失败(参见扩展错误代码)	与说明相同
2	资源不足	与说明相同
3	无效数值	与说明相同
4	IOI 语法错误(参见扩展错误代码)	与说明相同
5	未知目标、无效类、未定义实例或结构元素(参见扩展错误代码)	与说明相同
6	信息包空间不足	与说明相同
7	丢失连接	与说明相同
8	不支持该服务	与说明相同
9	数据段错误或数值属性无效	与说明相同
000A	属性列表错误	与说明相同
000B	状态已经存在	与说明相同
000C	对象模型冲突	与说明相同
000D	对象已经存在	与说明相同
000E	属性不可设置	与说明相同
000F	权限不够	与说明相同
10	设备状态冲突	与说明相同
11	应答不适合	与说明相同
12	未使用的存储器单元	与说明相同
13	命令数据不足	与说明相同
14	属性不支持	与说明相同
15	数据太多	与说明相同
001A	网桥请求数据太大	与说明相同
001B	网桥响应数据太大	与说明相同
001C	缺少属性列表	与说明相同

错误代码: (十六进制)	说明:	软件显示:
001D	无效的属性列表	与说明相同
001E	嵌入服务错误	与说明相同
001F	连接相关错误(参见扩展错误代码)	与说明相同
22	接收无效应答	与说明相同
25	关键信息段错误	与说明相同
26	无效 IOI 错误	与说明相同
27	列表中未预期属性	与说明相同
28	DeviceNet 错误 ®C 无效ID 项	与说明相同
29	DeviceNet 错误 ®C 项未设置	与说明相同
00D1	模块不在运行状态	未知错误
00FB	通讯口不支持	未知错误
00FC	数据类型不支持	未知错误
00FD	通讯未初始化	未知错误
00FE	通讯超时	未知错误
00FF	一般错误(参见扩展错误代码)	未知错误

扩展错误代码

RSLogix 5000编程软件不显示扩展错误代码的任何文本信息。
下表是错误代码是 0001 的扩展错误代码。

扩展错误代码(十六进制):	说明:
100	连接正被使用
103	不支持传送
106	权限冲突
107	未找到连接
108	无效连接类型
109	无效连接长度
110	未配置模块
111	EPR 不支持
114	模块错误
115	设备类型错误
116	版本错误
118	配置格式无效
011A	应用超出连接范围

扩展错误代码(十六进制):	说明:
203	连接超时
204	未连接信息超时
205	未连接发送参数错误
206	信息太大
301	无缓冲器内存
302	带宽不可用
303	无效筛选装置
305	署名匹配
311	端口不可用
312	链接地址无效
315	无效信息段类型
317	连接未预定

下面时错误代码 001F 的扩展错误代码。

扩展错误代码(十六进制):	说明:
203	连接超时

下表是错误代码是 0004 和 0005 的扩展错误代码。

扩展错误代码(十六进制):	说明:
0	扩展状态超出内存范围
1	扩展状态超出实例

下面是错误代码00FF的扩展错误代码。

扩展错误代码(十六进制)：	说明：
2001	IOI过多
2002	参数值出错
2018	拒绝接收信号
201B	长度太小
201C	无效长度
2100	授权失效
2101	无效钥匙开关位置
2102	密码无效
2103	未发布密码
2104	地址超出范围
2105	地址和数量超出范围
2106	数据正被使用
2107	信息类型无效或不支持

扩展错误代码(十六进制)：	说明：
2108	控制器正在下载或上载模式
2109	试图改变数组维数
210A	符号名称无效
210B	符号不存在
210E	搜索失败
210F	未起动任务
2110	不能写入
2111	不能读出
2112	共享例程不可编辑
2113	控制器处于故障模式
2114	运行模式被禁止

PLC 与SLC 错误代码 (.ERR)

对于PLC和SLC信息类型(PCCC信息)关联错误，Logix控制器固件版本10.x及更高提供了新的错误代码。

- 此更新使得RSLogix5000软件为许多错误给出更有意义的说明。此前RSLogix5000无法给出与00F0错误代码关联的任何错误说明。
- 此更新同时使得Logix控制器的错误代码与由其它控制器(例如PLC-5控制器)返回的错误更加一致。

下表给出了从R9.x及更低版本到R10.x及更高版本错误代码的更新。更新后，对应每个PCCC错误，.ERR项均返回唯一值。这些错误不再需要.EXERR。

表3.1 PLC和SLC错误代码(十六进制)

R9.x及更早版本		R10.x及更高版本		说明:
.ERR	.EXERR	.ERR	.EXERR	
10		1000		来自本地处理器的命令或格式不规范
20		2000		通讯模板不工作
30		3000		远程节点丢失，断开或关闭
40		4000		处理器连接但是有故障(硬件)
50		5000		错误的站点号
60		6000		请求功能不可用
70		7000		处理器处在编程模式
80		8000		处理器的兼容文件不存在
90		9000		远程节点不能缓存命令
00B0		B000		处理器正在下载，所以不能访问
00F0	1	F001		处理器转换地址不正确
00F0	2	F002		地址不完整
00F0	3	F003		地址不正确
00F0	4	F004		地址格式无效—未发现符号
00F0	5	F005		地址格式无效—符号是0或大于设备支持的最大字符长度
00F0	6	F006		寻址的文件在目标处理器中不存在
00F0	7	F007		目标文件中用于请求字数太小
00F0	8	F008		不能完成请求 处理多数据包操作期间，情况发生变化

表3.1 PLC和SLC错误代码(十六进制)(续)

R9.x及更早版本		R10.x及更高版本		说明:
.ERR	.EXERR	.ERR	.EXERR	
00F0	9	F009		数据或文件太大 内存不可用
00F0	000A	F00A		目标处理器不能将请求的信息放入信息包中
00F0	000B	F00B		权限错误，拒绝访问
00F0	000C	F00C		请求功能不可用
00F0	000D	F00D		请求多余
00F0	000E	F00E		不能执行命令
00F0	000F	F00F		溢出；超出频率范围
00F0	10	F010		不可用
00F0	11	F011		请求数据与可用数据的类型不匹配
00F0	12	F012		命令参数不正确
00F0	13	F013		相关地址在删除区
00F0	14	F014		由于未知原因导致命令执行失败 PLC-3超出频率范围
00F0	15	F015		数据转换失败
00F0	16	F016		扫描器不适合与1771机架适配器通讯
00F0	17	F017		适配器不适合与模块通讯
00F0	18	F018		1771模块响应无效
00F0	19	F019		标签重复
00F0	001A	F01A		文件宿主激活－文件正被所有者使用
00F0	001B	F01B		程序宿主激活－程序正被下载或在线编辑。
00F0	001C	F01C		磁盘文件写保护或不能访问(只离线)
00F0	001D	F01D		磁盘文件正被另一个应用程序使用，不能更新(只离线)

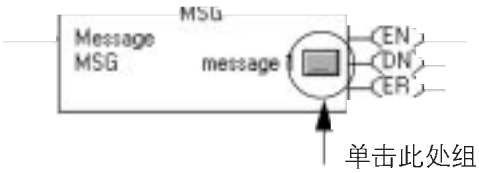
块传送错误代码

下面是Logix5000块传送特定的错误代码。

错误代码 (十六进制):	说明:	软件显示:
00D0	扫描器在发出请求3.5 秒内没有收到来自块传送模块的响应	未知错误
00D1	来自读响应的校验与数据流的校验不匹配	未知错误
00D2	扫描器发出读或写请求，但是块传送模块的响应与请求相反	未知错误
00D3	块传送相应的长度与扫描器请求的长度不一致	未知错误
00D6	扫描器从块传送模块收到响应，表明写请求失败	未知错误
00EA	扫描器未被配置与包含块传送模块的机架通讯	未知错误
00EB	指定的逻辑槽不可用于给定的机架尺寸	未知错误
00EC	″当前块传送请求正在进行中,但是在另一请求开始前必须得到响应″	未知错误
00ED	块传送请求尺寸与有效块传送请求尺寸不一致	未知错误
00EE	块传送请求的类型与BT_READ 或 BT_WRITE 要求不一致	未知错误
00EF	扫描器不能在块传送表内找到一个可用槽来完成块传送请求	未知错误
00F0	在块传送执行未完成时，扫描器收到一个复位远程I/O通道的请求	未知错误
00F3	用于远程块传送的队列已满	未知错误
00F5	没有为请求的机架或槽配置通讯通道	未知错误
00F6	没有为远程I/O组态通讯通道	未知错误
00F7	块传送超时，在完成块传送之前达到指令设置的超时时间	未知错误
00F8	块传送协议错误 ®C 未经请求的块传送	未知错误
00F9	由于通讯通道损坏，块传送数据丢失	未知错误
00FA	块传送模块请求的长度与相应的块传送指令不一致	未知错误
00FB	块传送读数据的校验错误	未知错误
00FC	在适配器与块传送模块之间存在无效的块传送写数据	未知错误
00FD	块传送的长度加上块传送数据表的索引长度大于块传送数据表文件的长度	未知错误

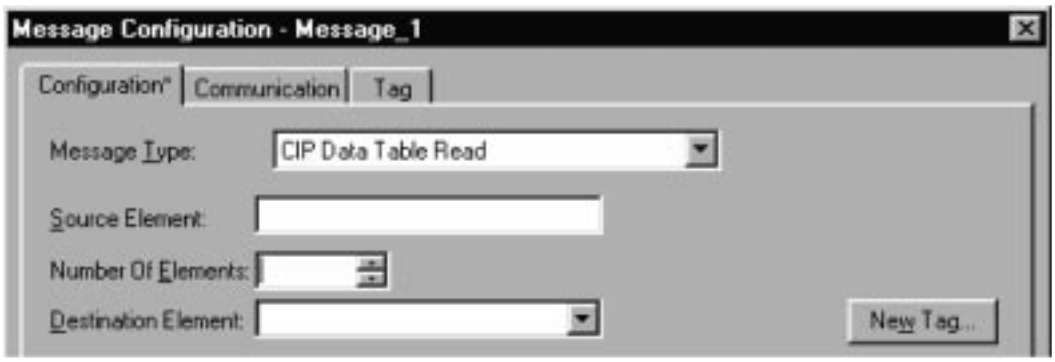
指定详细组态信息

在用户输入MSG 指令而且指定了MESSAGE结构体之后，使用通讯组态对话框指定通讯的详细信息。



单击此处组态MSG指令

通讯组态的细节取决于与用户选择的通讯类型。



如果目标设备是：	选择以下信息类型之一：	参见页码：
Logix5000 控制器	CIP 数据表读	3-16
	CIP 数据表写	
用RSLogix 5000 软件配置的I/O模块	重新组态模块	3-17
	CIP 通用信息	3-18
PLC-5 控制器	PLC5 类型读	3-19
	PLC5 类型写	
	PLC5 字域读	
	PLC5 字域写	
SLC 控制器 MicroLogix 控制器	SLC 类型读	3-20
	SLC 类型写	
块传送模块	块传送读	3-21
	块传送写	
PLC-3 处理器	PLC3 类型读	3-22
	PLC3 类型写	
	PLC3 字域读	
	PLC3 字域写	
PLC-2 处理器	PLC2 无保护读	3-23
	PLC2 无保护写	

用户必须明确指定以下信息：

在下列属性：	指定：
Source Element 源元素	• 如果用户选择读信息类型，则源元素是用户要读的目标设备内的数据地址。采用目标设备的寻址语法。 • 如果用户选择写信息类型，则源标签是要发送到目标设备的数据标签的第一个元素。
Number of Elements 元素数	用户读/写元素的数量由使用的数据类型决定。一个元素的大小与其相关的数据“组块”有关。例如，标签timer1是由一个计时器控制结构体组成的元素。
Destination Element 目标元素	• 如果用户选择读信息类型，则目标元素是存储从目标设备读取的数据的Logix5000 控制器内的第一个元素。 • 如果用户选择写信息类型，则目标元素是目标设备内用户要写入数据的位置地址 。

指定 CIP 数据表读或写信息

CIP 数据表读或写信息类型用于在Logix5000 控制器之间传送数据。

选择下列命令：	如果用户需要：
CIP Data Table Read CIP数据表读	从另一个控制器读数据。 源和目的数据类型必须匹配。
CIP Data Table Write CIP数据表写	写数据到另一个控制器。 源和目的数据类型必须匹配。

重新组态 I/O 模块

使用模块重组态信息向I/O模块发送新的配置信息。在重新组态模块过程中：

- 输入模块继续向控制器发送输入数据。
- 输出模块继续控制输出设备。

重新组态模块信息需要配置下列属性：

属性：	选择：
Message Type	模块重新组态

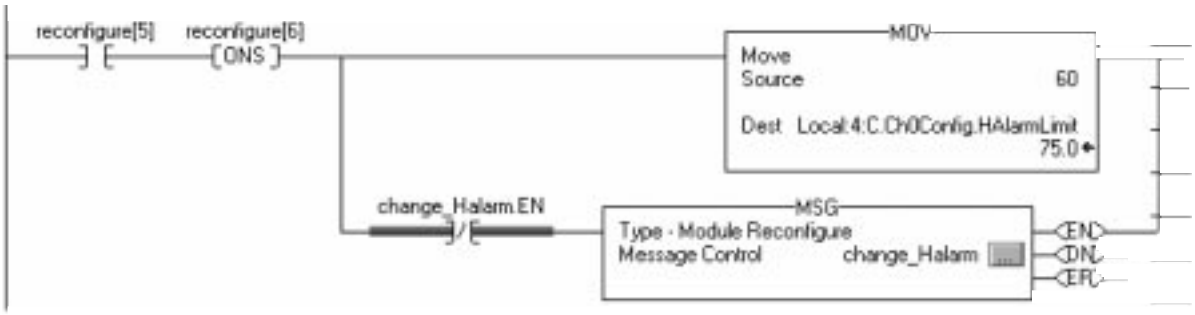
举例：重新组态I/O 模块：

1. 将模块配置标签需要的项组态成新的值。

2. 将模块重新组态信息发送到模块。

如果 reconfigure[5] 被置位，则将本地4 号槽模块的上限报警设置 为60。然后重新组态模块信息 将新报警值发送到模块。槽指令防止在reconfigure[5] 是接通状态期间梯级向模块发送多条信息。

梯形图



结构文本

```
IF reconfigure[5] AND NOT reconfigure[6]THEN
Local:4:C.Ch0Config.HAlarmLimit := 60;
IF NOT change_Halarm.EN THEN
MSG(change_Halarm);
END_IF;
END_IF;
reconfigure[6] := reconfigure[5]
```

指定CIP 通用信息

CIP 通用类型信息在I/O模块上执行特定动作。

如果用户需要：	在此属性：	类型或选择：	
在离散量输出模块执行脉冲测试	Message Type	CIP通用信息	
	Service Type	脉冲测试	
	Source	数据类型INT[5]的tag_name	
		该数组包含：	
		tag_name[0]	测试点的位屏蔽(同时只测试一个点)
		tag_name[1]	保留，可以是 0
		tag_name[2]	脉冲宽度(几百微秒，一般是20)
		tag_name[3]	对于ControlLogix I/O是过零交叉延时(几百微秒，一般是40)
		tag_name[4]	校验延时
	Destination	保留空白	
复位离散量输出模块上的电子保险	Message Type	CIP通用信息	
	Service Type	复位电子保险	
	Source	DINT类型的tag_name 该标签代表被复位保险丝处的位屏蔽	
	Destination	空白	
复位离散量输入模块上的锁存诊断	Message Type	CIP通用信息	
	Service Type	复位锁存诊断(I)	
	Source	DINT类型的tag_name 该标签代表被复位保险丝处的位屏蔽	
复位离散量输出模块上的锁存状态	Message Type	CIP通用信息	
	Service Type	复位锁存诊断 (O)	
	Source	DINT类型的tag_name 该标签代表被复位保险丝处的位屏蔽	
解锁存模拟量输入模块报警	Message Type	CIP通用信息	
	Service Type	选择要解锁存报警： • 解锁存所有报警(I) • 解锁存模拟量高报警(I) • 解锁存模拟量高高报警(I) • 解锁存模拟量低报警(I) • 解锁存模拟量低低报警(I) • 解锁存速率报警(I)	
	Instance	要解锁存的报警通道	

如果用户需要：	在此属性：	类型或选择：
解锁存模拟量输出模块报警	Message Type	CIP通用信息
	Service Type	选择解锁存报警： • 解锁存所有报警(O) • 解锁存高报警(O) • 解锁存低报警(O) • 解锁存斜率报警(O)
	Instance	要解锁存的报警通道

指定 PLC-5 信息

使用PLC-5 信息类型与 PLC-5 控制器通讯。

选择以下命令：	如果用户需要：
PLC5 Type Read PLC5类型读	读取16位整数，浮点数或字符串数据类型并保持数据的完整性。参见3-19页的表3.2。
PLC5 Type Write PLC5类型写	写入16位整数，浮点数或字符串数据类型并保持数据的完整性。参见3-19页的表3.2。
PLC5 Word Range Read PLC5字范围读	读PLC-5 存储器内16-位字的连续区域而与数据类型无关。 此命令从源元素指定的地址开始依次读取请求数量的16-位字。 来自源元素的数据存储在从目的标签指定地址开始的区域内。
PLC5 Word Range Write PLC5字范围写	从Logix5000 存储器写16-位字连续区域到PLC-5存储器，而与数据类型无关。 此命令从源标签指定的地址开始依次读取请求数量的16-位字。 来自源标签的数据被存储在从目的元素指定地址开始的PLC-5处理器内。

下表给出了PLC5 类型读与PLC5 类型写信息所用的数据类型。

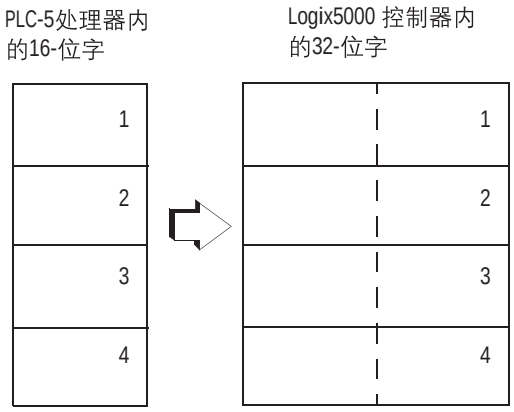
表3.2 PLC5 类型读和类型写的信息数据类型

PLC-5数据类型：	使用Logix5000数据类型：
B	INT
F	REAL
N	INT
器	DINT(如果数值是≥-32,768 并 ≤32,767，只有写DINT值到PLC-5控制时用)
S	INT
ST	STRING

类型读和类型写命令也适用于SLC 5/03处理器 (OS303及更高, SLC 5/04 处理器(OS402及更高))及SLC 5/05处理器。

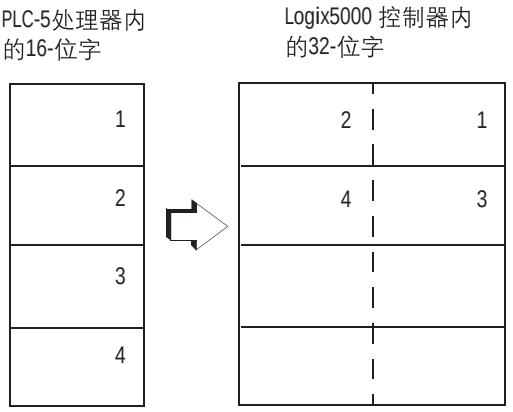
下图给出类型命令和字范围命令的区别。此例使用读命令实现Logix5000控制器从PLC-5处理器中读取数据。

类型读命令



定型命令保持数据结构和数值。

字域读命令



字域命令连续填充目的单元标签。
数据结构和数值根据目的单元的数据类型变化。

指定 SLC 信息

SLC 信息类型用于与SLC 和 MicroLogix控制器通讯。下表列出了允许访问的数据类型。同时也列出相应的Logix5000数据类型。

对于SLC 或 MicroLogix 数据类型:	使用该Logix5000 数据类型:
F	REAL
L (MicroLogix 1200和 1500控制器)	DINT
N	INT

指定块传送信息

块传送信息类型用于与通用远程I/O 网络上的块传送模块进行通讯。

如果用户需要：	选择下列命令：
从块传送模块读数据。 此信息类型可取代BTR 指令。	块传送读
写数据到块传送模块 此信息类型取代BTW 指令。	块传送写

组态块传送信息应按照如下规则：

- 源(对于 BTW 指令) 和目的(对于 BTR 指令) 标签必须足够大以能够接受请求的数据， 除MESSAGE, AXIS和 MODULE 结构体外。
- 用户必须指定发送或接收的16-位整数(INT)的长度。 可指定整数范围是从0 到 64。

如果用户需要：	指定：
块传送模块确定发送多少16-位整数(BTR)。 控制器发送 64 个整数(BTW)。	单元数量为0

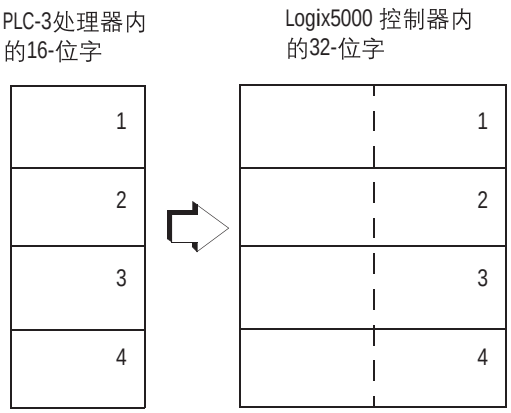
指定PLC-3 信息

PLC-3 信息类型专为PLC-3 处理器设计。

选择下列命令：	如果用户需要：
PLC3 Typed Read PLC3类型读	读取整数或REAL 类型数据。 对于整数，此命令从PLC-3处理器读取16-位整数，并存储它们到Logix5000 控制器的SINT, INT, 或DINT 数据数组内且保持数据的完整性。 此命令也从PLC-3 处理器读取浮点数并存入Logix5000 控制器的REAL数据类型标签内。
PLC3 Typed Write PLC3类型写	写整数或 REAL 类型数据。 此命令写SINT 或 INT 数据到PLC-3 整数文件内，并且保持数据的完整性。只要在INT数据类型范围内(-32,768≥ 数据≤32,767) 用户就可以写DINT 数据。 此命令也可从Logix5000 控制器写REAL类型数据到一个PLC-3浮点数文件。
PLC3 Word Range Read PLC3字域读	读PLC-3 存储器内16-位字的连续域而与数据类型无关。 此命令从源单元指定的地址开始依次读取请求数量的16-位字。 来自源单元的数据存储在从目的标签指定的地址开始的区域内。
PLC3 Word Range Write PLC3字域写	从Logix5000 存储器写16-位字连续域到PLC-3存储器，而与数据类型无关。 此命令从源标签指定的地址开始依次读取请求数量的16-位字。 来自源标签的数据被存储在PLC-3处理器内从目的元素指定的地址起始范围内。

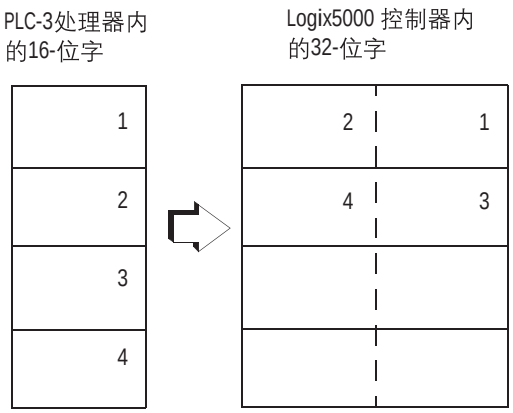
下图表明类型命令和字域命令的区别。本例中使用读命令从PLC-3 处理器读取数据到Logix5000 控制器。

类型读命令



定型命令保持数据结构和数值。

字域读命令



字域命令连续填充目的单元标签。
数据结构和数值根据目的单元的数据类型变化。

指定 PLC-2 信息

PLC-2 信息类型专为PLC-2 处理器设计。

选择下列命令：	如果用户需要：
PLC2 Unprotected Read PLC2无保护读	从PLC-2数据表的任何区域或另一个处理器的PLC-2兼容文件读取16-位字。
PLC2 Unprotected Write PLC2无保护写	写16-位字到PLC-2数据表的任何区域或另一个处理器的PLC-2兼容文件。

因为信息传送使用16-位字，所以要确定 Logix5000 相应的存储传送数据的标签 (通常是一个INT 数组)。

MSG指令组态举例

下面实例给出对于不同控制器组合所使用的源、目的标签和元素。

MSG 指令由Logix5000 控制器启动并写数据到另一个控制器内：

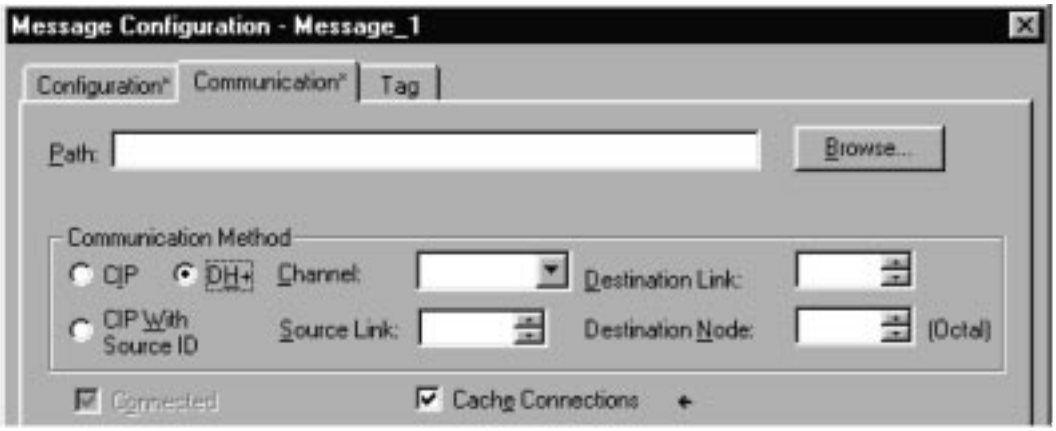
通讯路径：	源和目的标签举例：	
Logix5000→Logix5000	源标签	array_1 [0]
	目的标签	array_2 [0]
	可以使用别名标签作为源标签 (在启动通讯的 Logix5000 控制器)。	
	对于目的标签不能使用别名。目的标签必须是基本标签。	
Logix5000 →PLC-5	源标签	array_1[0]
Logix5000 →SLC	目的元素	N7:10
	可以使用别名标签作为源标签 (在起动通讯的 Logix5000 控制器)。	
Logix5000 →PLC-2	源标签	array_1[0]
	目的单元	010

MSG 指令由Logix5000 控制器起动，从另一个控制器读数据。

通讯路径：	源和目的标签举例：	
Logix5000→Logix5000	源标签	array_1[0]
	目的标签	array_2[0]
	对于源标签不能使用别名。源操作数必须是基本标签。	
	可以使用别名标签作为目的标签(在初始化通讯的 Logix5000 控制器)。	
Logix5000→PLC-5 Logix5000 →SLC	源元素	N7:10
	目的标签	array_1[0]
	可以使用别名标签作为目的标签(在起动通讯的 Logix5000 控制器)。	
Logix5000 -PLC-2	源元素	010
	目的标签	array_1[0]

指定通讯详细信息

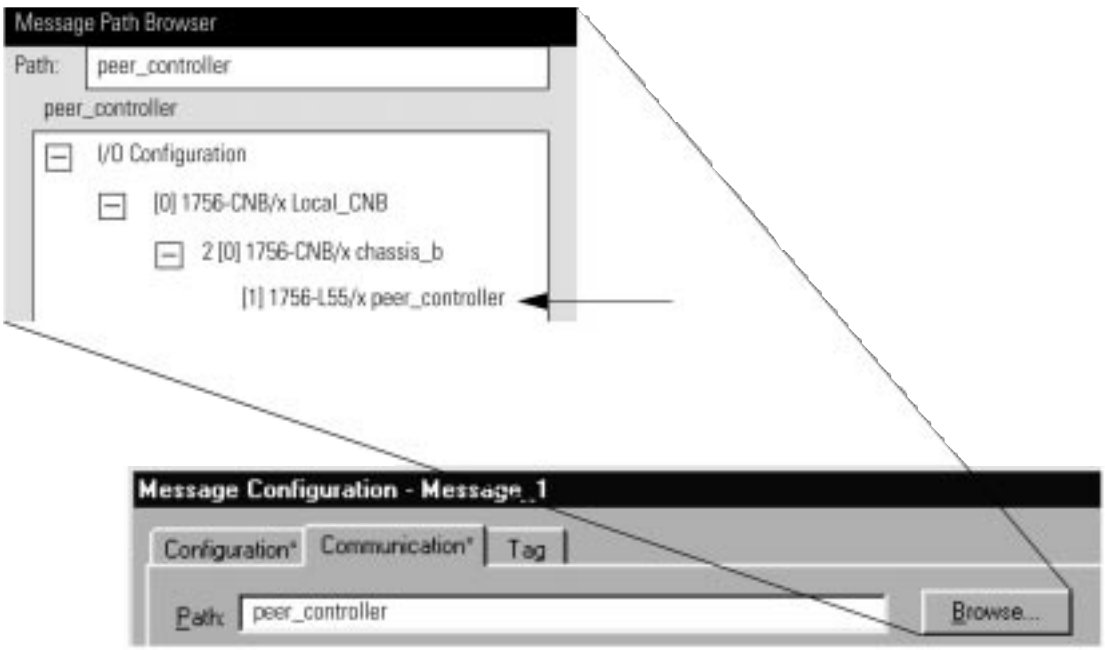
当组态 MSG 指令时，用户可在通讯选项卡中指定详细信息。



指定路径

路径指明使信息到达目的地所经由的路线。

- 如果要将本地通讯模块，远程通讯模块，目标控制器或设备添加到控制器的I/O组态中，则可用浏览按钮来选择目的地。



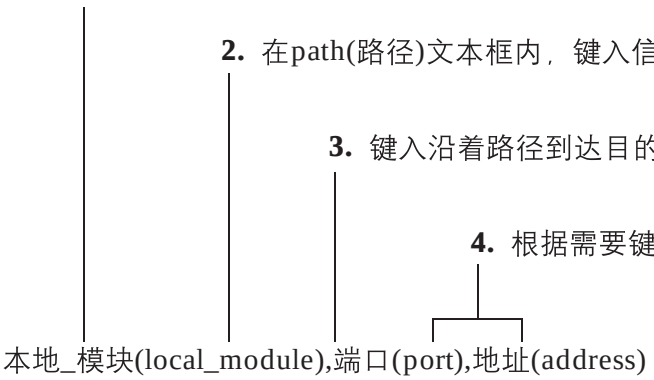
- 某些远程通讯模块或设备在控制器的I/O组态中不可用。对于这种情况按照下列方式选择路径：

1. 用 Browse (扫描)按钮选择本地通讯模块。

2. 在path(路径)文本框内，键入信息离开模块的端口。

3. 键入沿着路径到达目的地经由的下个模块的地址。

4. 根据需要键入另一个端口或地址的组合。



此处:	通讯对象:	填入:
Port 端口	任何1756控制器或模块的背板。	1
	Logix5000控制器的DF1 端口	2
	1756-CNB模块的ControlNet端口	
	1756-ENBx 或 ®CENET模块的以太网端口	
	1756-DHRIO 模块的DH+口经由通道A	
	1756-DHRIO 模块的DH+口，经由通道B	3
Address 地址	ControlLogix 背板	槽号
	DF1网络	站点地址(0-254)
	ControlNet网络	节点号(1-99 十进制)
	DH+网络	键入8# 接下来是节点号(1-77 八进制) 例如，要指定八进制节点地址37，则键入8#37。
	EtherNet/IP 网络	可以用以下任一格式指定EtherNet/IP 网络上模块： IP 地址 (例如, 130.130.130.5) IP 地址：端口 (例如130.130.130.5:24) DNS 名称 (例如, 罐) DNS 名称：端口 (例如, 罐:24)

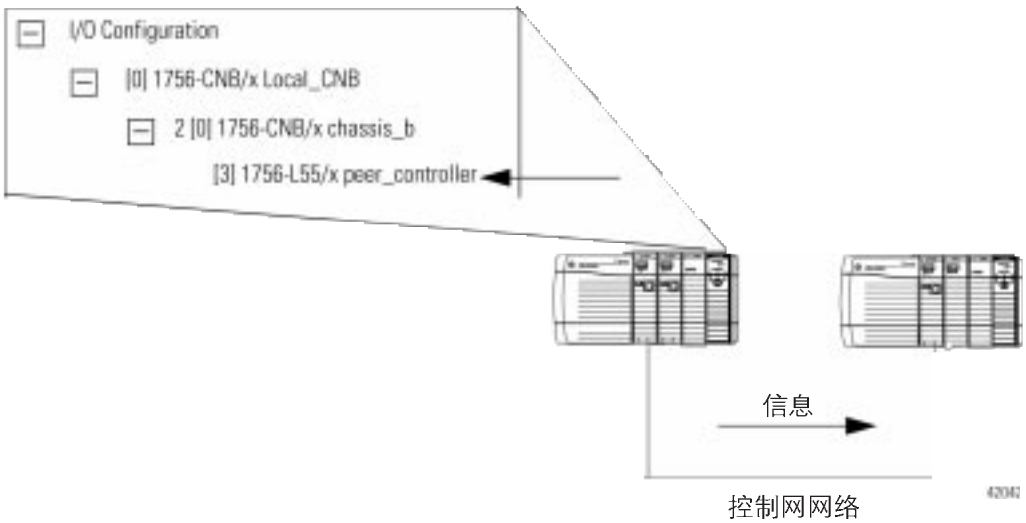
- 对于块传送信息，需要将下列模块加到控制器的I/O组态中：

块传送通过的网络：	添加到I/O组态中的模块：
ControlNet	• 本地通讯模块 (例如, 1756-CNB 模块) • 远程适配器模块(例如, 1771-ACN 模块)
通用远程 I/O	• 本地通讯模块(例如, 1756-DHRIO 模块) • 一个远程适配器模块 (例如, 1771-ASB 模块)对应框架内一个机架或机架的一部分 • 块传送模块 (可选)

路径选择的实例参见下列页：

- 通过ControlNet, 参见 3-27 页
- 通过 EtherNet/IP, 参见3-28页
- DH+信息, 参见 3-29页

举　例　指定经由ControlNet的路径



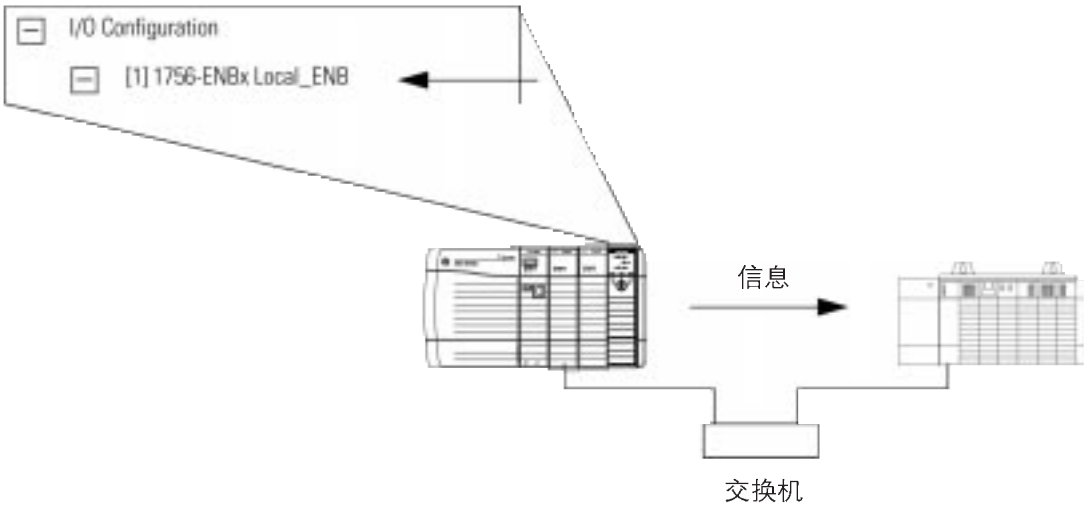
路径：peer_controller

此处：

peer_controller 是指接收信息的控制器名称。

举 例

指定一经由EtherNet/IP的路径

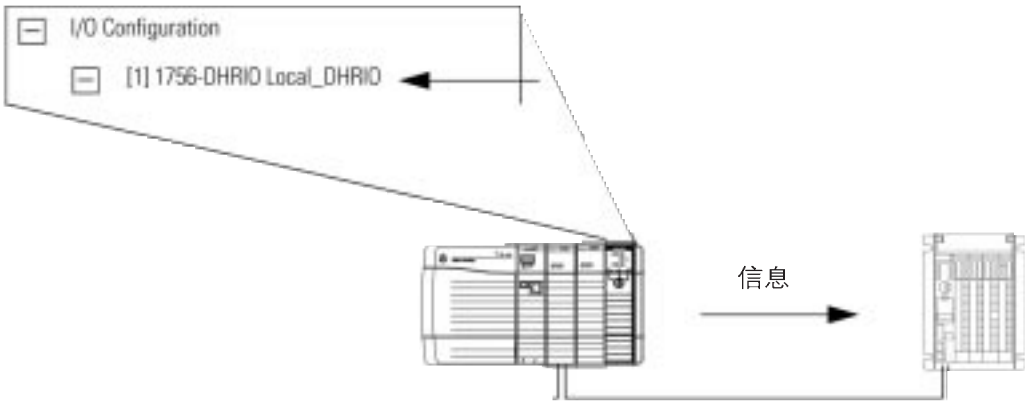


路径：Local_ENB,2,127.127.127.12

此处：	是指：
Local_ENB	本地框架中1756-ENBx模块的名称
2	本地框架中1756-ENBx模块的以太网口
127.127.127.12	SLC 5/05控制器的IP地址

举　例

指定一经由DH+网络的路径



DH+网络

路径： Local_DHRIO

此处：

Local_DHRIO 是指与发送信息的控制器在同一个机架中的1756-DHRIO模块的名称。

指定通讯方法或模块地址

下表是选择信息的通讯方式或模块地址。

如果目标设备是:	则选择:	还需指定:	
Logix5000 控制器	CIP	没有其它特殊要求	
EtherNet/IP网上的PLC-5控制器			
ControlNet上的PLC-5控制器			
SLC 5/05控制器			
DH+网上的PLC-5 控制器	DH+	通道	连接到DH+网上的1756-DHRIO模块的通道A或B 。
DH+网上的SLC 控制器		源链路	在1756-DHRIO模块路线表中控制器所在背板的链路ID。(路线表中的源节点自动给定控制器的槽号。)
PLC-3 处理器		目地链路	目标设备驻留的远程DH+链路的链路 ID
PLC-2处理器		目的节点	目标设备的站点地址，八进制表示。
		如果只有一个 DH+ 链路而且不用RSLinx软件配置DH/RIO模块为远程链路，则源链路和目的链路都设定为0。	
在工作站上应用程序使用RSLinx通过EtherNet/IP或 ControlNet 接收主动提供的信息	CIP with Source ID (应用程序可从控制器接收数据。)	源链路	RSLinx 软件中主题的远程ID 。
		目的链路	RSLinx 软件设置的虚拟链路ID (0-65535)
		目的节点	目的ID (0-77八进制) 由应用程序提供给RSLinx 软件。对于RSLinx 中的DDE 用77。
		ControlLogix 控制器的槽号被用作源节点。	
通过通用远程I/O网的块传送模块	RIO	通道	连接到RIO网上的1756-DHRIO模块的通道A或B 。
		机架	模块的机架号 (八进制)
		组	模块的组号
		槽号	模块所在的槽号
通过ControlNet的块传送模块	ControlNet	槽号	模块所在的槽号

选择缓存选项：

根据用户对MSG指令的组态方式不同，可能使用一个连接来发送或接收数据。

信息类型：	通讯方式：	是否使用连接：
CIP数据表读或写	→	✓
PLC2, PLC3, PLC5, 或SLC (所有类型)	CIP 带有源ID的CIP	
	DH+	✓
CIP generic CIP通用	→	用户可选(1)
块传送读或写	→	✓

(1)用户可连接CIP通用信息。但对大多数应用，我们建议用户保留CIP通用信息未连接。

如果MSG指令占用连接，在信息传输完成时，用户可选择保留连接打开(缓存)或关闭连接。

如果用户：	则：
缓存连接	MSG指令执行完毕后仍然打开连接。这会优化执行时间。每次打开连接，信息传送延长执行时间。
未缓存连接	MSG指令执行完毕后关闭连接。使得该连接空闲并可用于其它应用。

控制器对用户可缓存的连接数有以下限制：

如果用户使用如下固件版本的软件：	则用户可缓存：
11.x或更低	• 块传送信息，最多16个连接 • 其它类型信息，最多16个连接
12.x或更高	最多32个连接

如果多条信息发送至同一设备，则信息可共享一个连接。
如果MSG指令发送到：

如果MSG指令发送到：	且该指令：	则：
不同设备	→	每条MSG指令占用1个连接。
同一设备	在同一时间使能	每条MSG指令占用1个连接。
	不在同一时间使能	MSG指令共享连接。 （也就是说，共计占用一个连接。）

举　例

共享连接

如果控制器将块传送读信息和块传送写信息交替发送到同一模块，则这两个信息将占用一个连接。在缓存列表中这两个信息按一个计算。

规则

如果用户规划和编程MSG指令，遵守下列规则：

规则：	详细内容：
1. 为每条MSG指令创建一个控制标签。	每条MSG指令需要其自己的控制标签。 • 数据类型＝MESSAGE • 作用域＝控制器 • 标签不能为数组的一部分或用户定义的数据类型。
2．保证源和(或)目的数据位于控制器作用域内。	MSG指令只能访问在控制器标签文件夹(控制器作用域)内的标签。
3．如果要将信息MSG到16位整型数据格式的设备中，需为工程中MSG 指令的INTs型和DINTs型数据设置缓冲区。	如果用户将信息发送到16位整型格式的设备，例如PLC-5®或SLC 500™控制器，且传送整数(而非实数)，需为工程中MSG指令的INTs型和DINTs型数据设置缓冲区。 这样可提高工程的执行效率，因为以32位整数(DINTs)工作时，Logix控制器的执行效率更高且占用内存更少。 INTs和DINTs之间的转换，参阅Logix5000 Controllers Common Procedures，版本号1756-PM001。
4．缓存执行最频繁的已连接MSGs	将执行频率最高的MSG指令缓存，最多可达到用户控制器所允许的最大值。 由于控制器不必在每次信息执行时都打开连接，所以执行时间得到优化。
5．如果用户同时使能的MSGs多于16条，需使用不同类型的管理策略。	如果用户同时使能的MSGs多于16条，MSG指令可能延迟进入信息队列。要保证每条信息均执行，请使用以下选项之一： • 顺次使能每条信息。 • 按组使能信息。 • 编写MSG指令与多个设备通讯。更多信息，请参见Logix5000控制器通用程序，版本号1756-PM001。 • 编写逻辑调整信息执行的。更多信息，请参见Logix5000控制器通用程序，版本号1756-PM001。
6．保证未连接和未缓存的MSGs数量少于未连接缓冲区的数量。	控制器有10-40个未连接的缓冲器。缺省值为10。 • 当指令离开信息队列时，如果全部未连接的缓冲器都在使用，则指令错误且不传送数据。 • 用户可增加未连接的缓冲器数量(最多40)，但仍需遵循规则5。 • 要增加未连接的缓冲器数量，参阅Logix5000控制器通用程序，版本号1756-PM001。

获取系统值 (GSV) 和设置系统值 (SSV)

GSV/SSV 指令用于读取或设置存储于对象内的控制器系统值。

操作数：

梯形图

操作数：	类型：	格式：	说明：
Class name 类名称		名称	对象名称
Instance name 实例名称		名称	特定对象的名称，可选
Attribute name 属性名称		名称	对象属性 数据类型取决于用户所选属性
Destination 目的(GSV)	SINT INT DINT REAL	标签	属性数据的目的标签
Source 源(SSV)	SINT INT DINT REAL	标签	包含要复制到属性项的数据标签

结构文本

```
GSV(ClassName,InstanceName,AttributeName,Dest);
SSV(ClassName,InstanceName,AttributeName,Source);
```

说明：结构文本中GSV和SSV指令的操作数与梯形图中GSV和SSV指令的操作数相同。

GSV/SSV 指令读取或设置存储在对象内的控制器系统数据。Logix控制器将系统数据存储于对象内。而区别于PLC-5处理器，不存在状态文件。

当指令被使能时，GSV 指令获取指定信息并把信息存放在目的标签中。当指令被使能时，SSV 指令用来自源操作数的数据设置指定属性。

当用户输入一条 GSV/SSV 指令时，对于每条指令编程软件显示其有效的对象类别，对象名称,和属性名称。对于GSV 指令，用户可获取其所有可用属性值。对于SSV 指令，编程软件只显示允许设置的属性值。

注意：使用 GSV/SSV 指令时要注意，将对象属性值改变可能会导致不可预料的控制器运行或人身伤害。

如果源或目的标签的长度太小，则指令不执行且记录为次要故障。下节中 GSV/SSV 对象定义了每个对象属性和与之相关的数据类型。例如，程序对象的MajorFaultRecord(主要故障记录)属性需要DINT[11]数据类型。

算术状态标志：不影响

故障条件：

发生次要故障的条件：	故障类型：	故障代码：
对象地址无效	4	5
指定了不支持GSV/SSV 指令的对象	4	6
属性无效	4	6
没有为SSV指令提供足够的信息	4	6
GSV 指令的目的标签太小不能保持请求的数据	4	7

执行：

条件：	梯形图动作：	结构文本动作：
预扫描	梯级输出条件被设置为假。	不动作
梯级输入条件为假	梯级输出条件被设置为假。	无影响
梯级输入条件为真	指令执行。 梯级输出条件被设置为真。	无影响
输入使能被置位	无影响	输入使能一直被置位。 指令执行。
指令执行	获取或设置指定值。	获取或设置指定值。
后期扫描	梯级输出条件被设置为假。	不动作。

GSV/SSV 对象

当用户输入一条 GSV/SSV 指令时，指定访问对象及其属性。有些情况下,对同一对象类型会有多个实例，所以用户也必须指定对象名称。例如，在用户应用程序中可能有多个任务。用户可以通过任务名称访问其对应的TASK 对象 。

注 意：对于 GSV 指令，只有指定长度的数据被复制到目的标签。例如，如果属性被指定为SINT 而目的标签是DINT，则只有DINT目的标签的低 8 位被更新，其余的 24 位不改变。

用户可以访问下列对象：

有关对象的信息：	参见页码次或出版物：
AXIS	ControlLogix运动模块安装和组态手册，出版号 1756-UM006
CONTROLLER	3-37
CONTROLLERDEVICE	3-37
CST	3-39
DF1	3-40
FAULTLOG	3-43
MESSAGE	3-44
MODULE	3-46
MOTIONGROUP	3-47
PROGRAM	3-48
ROUTINE	3-49
SERIALPORT	3-49
TASK	3-51
WALLCLOCKTIME	3-53

访问 CONTROLLER 对象

CONTROLLER 对象提供有关控制器执行的状态信息。

属性:	数据类型:	指令:	说明:
TimeSlice	INT	GSV	指定CPU用于通讯的比率。
		SSV	有效值是10-90。 当控制器钥匙开关处于运行位置时不能改变该值。

访问 CONTROLLERDEVICE 对象

CONTROLLERDEVICE 对象标识控制器的物理硬件。

属性:	数据类型:	指令:	说明:	
DeviceName	SINT[33]	GSV	ASCII字符串表明控制器目录号和所用内存模板。 第一个字节包含字符串数组返回的ASCII字符的数量。	
ProductCode	INT	GSV	标识控制器类型	
			Logix控制器:	产品代码:
			CompactLogix5320	43
			CompactLogix5330	44
			ControlLogix5335E	65
			ControlLogix5550	3
			ControlLogix5553	50
			ControlLogix5555	51
			ControlLogix5561	54
			ControlLogix5562	55
			ControlLogix5563	56
			DriveLogix5720	48
			FlexLogix5433	41
			FlexLogix5434	42
			SoftLogix5860	15
ProductRev	INT	GSV	标识当前产品版本号。以十六进制显示。 低位字节包含主要版本号；高位字节包含次要版本号。	

属性:	数据类型:	指令:	说明:
SerialNumber	DINT	GSV	设备的序列号。 在设备出厂时就已分配的产品序列号。
Status	INT	GSV	各位表示的状态: 位 3-0 保留 设备状态位 位 7-4: 意义: 0000 保留 0001 正在更新闪存 0010 保留 0011 保留 0100 闪存损坏 0101 有故障 0110 运行 0111 编程 故障状态位: 位 11-8: 意义: 0001 可恢复的次要故障 0010 不可恢复的次要故障 0100 可恢复的主要故障 1000 不可恢复的主要故障 Logix5000 特定状态位 位13-12 意义: 01 钥匙开关处于运行位置 10 钥匙开关处于编程位置 11 钥匙开关处于远程位置 位 15-14 意义: 01 控制器正在切换模式 10 调试模式，如果控制器在运行模式
Type	INT	GSV	标识设备是一个控制器。 控制器 = 14
Vendor	INT	GSV	标识设备的制造商。 Allen-Bradley = 0001

访问CST对象

CST(协调系统时间)对象为同一机架上的设备提供协调的系统时间。

属性:	数据类型:	指令:	说明:	
CurrentStatus	INT	GSV	协调系统时间的当前状态。数据位标识：	
			位：	含义：
			1-0	定时器硬件故障：设备的内部定时器硬件处于故障状态
			1-1	斜率变化使能：定时器的低16位的当前值线性变化为请求数值，而不是跳变到低位值。这些位由网络指定的时钟同步方法来处理。
			1-2	主系统时间：CST对象是ControlLogix系统的主时间。
			1-3	同步：CST对象的64位当前值(CurrentValue)由主站CST对象通过系统时间更新来同步
			1-4	主本地网络：CST对象是本地网络的主时间源
			1-5	处于中继模式 ：CST对象工作于时间中继模式
			1-6	检测到多主站：检测到一个重复的本地网络主时间源。对于时间关联节点，该位一直为零。
			1-7	未使用
			8-9	00=时间关联节点 01=时间主节点 10=时间中继节点 11=未使用
			10-15	未使用
CurrentValue	DINT[2]	GSV	定时器的当前值，DINT[0]包含低32位；DINT[1]包含高32位。 在更新服务中，定时器的源操作数被调整为与由本地通讯网络同步提供的数据相匹配。如CurrentStatus(当前状态)属性所示，该调整或者是线性过渡到请求值，或者是立即设定为请求值。	

访问 DF1 对象

DF1 对象提供一个到DF1通讯驱动程序的接口，可以用于组态串行口。

属性:	数据类型:	指令:	说明:		
ACKTimeout	DINT	GSV	等待应答信息传送的时间值。(只用于点对点和主站通讯)。 有效值是 0-32,767。 延迟以20毫秒为周期计数。缺省值是 50 (1 秒)。		
DiagnosticCounters	INT[19]	GSV	用于DF1 通讯驱动程序的诊断计数器数组。		
字偏移量	DF1 点对点		DF1 从站	主站	
0	标识 (0x0043)		标识 (0x0042)	标识 (0x0044)	
1	调制解调器位		调制解调器位	调制解调器位	
2	信息包发送		信息包发送	信息包发送	
3	信息包接收		信息包接收	信息包接收	
4	未送达的信息包		未送达的信息包	未送达的信息包	
5	不使用		重发信息	重发信息	
6	NAKs 接收		NAKs 接收	不使用	
7	ENQs 接收		接收查询信息包	不使用	
8	不正确的信息包NAKed		不正确的信息包不能应答	不正确的信息包不能应答	
9	无内存发送 NAK		无内存发送 ACKed	不使用	
10	接收重复信息包		接收重复信息包	接收重复信息包	
11	收到不正确的字符		不使用	不使用	
12	DCD 恢复计数		DCD 恢复计数	DCD 恢复计数	
13	丢失调制解调器计数		丢失调制解调器计数	丢失调制解调器计数	
14	不使用		不使用	最大优先扫描时间	
15	不使用		不使用	上次优先扫描时间	
16	不使用		不使用	最大正常扫描时间	
17	不使用		不使用	上次正常扫描时间	
18	ENQs 发送		不使用	不使用	
DuplicateDetection	SINT	GSV	使能重复信息检测。 值: 意义: 0 禁止检测重复信息 非零值 禁止检测重复信息		
EmbeddedResponse Enable	SINT	GSV	使能内置响应功能 (只适用于点对点)。 值: 意义: 0 只在接收到一个内置响应后才启动 (缺省设置) 1 无条件使能		
ENQTransmitLimit	SINT	GSV	ACK 超时后请求(ENQs) 发送的数量(只适用于点对点)。 有效值是 0-127. 缺省设置是3.		
EOTSuppression	SINT	GSV	使能抑制EOT 传送响应查询信息包(只适用于从站)。 值: 意义: 0 EOT抑制禁止 (禁止) 非零值 EOT抑制使能		
ErrorDetection	SINT	GSV	指定错误检测配置。 值: 意义: 0 BCC (缺省设置) 1 CRC		

属性:	数据类型:	指令:	说明:
MasterMessageTransmit	SINT	GSV	主站信息传送的当前值(只适用于主站). 值: 意义: 0 在各站之间轮询 1 在查询序列(代替主站的站号) 缺省值是 0。
NAKReceiveLimit	SINT	GSV	在停止传送前为响应信息，控制器所接收的NAKs数量(只适用于点对点通讯)。 有效值是 0-127. 缺省设置是3。
NormalPollGroupSize	INT	GSV	在轮询所有优先轮询节点组之后，轮询正常轮询节点组的站的数量(只适用于主站)。 有效值 0-255。缺省值是 0。
PollingMode	SINT	GSV	当前轮询模式(只适用于主站)。 值: 意义: 0 基于信息, 但不允许从站起动通讯。 0 基于信息, 但是允许从站起动通讯(缺省设置) 2 标准，每次扫描节点传送单条信息。 3 标准，每次扫描节点传送多条信息。 缺省值设置为1。
ReplyMessageWait	DINT	GSV	在为了得到响应轮询从站之前，接收到一个ACK 之后等待(作为主站)的时间(只适用于主站)。 有效范围是 0-65,535。延迟以20毫秒为周期计数。缺省值是 5 个周期 (100 毫秒)。
StationAddress	INT	GSV	串行口的当前站地址。 有效值是0-254。缺省值是 0。
SlavePollTimeout	DINT	GSV	在从站声明其不能传送之前(因为主站处于未激活状态)，等待主站轮询的时间量(以毫秒为单位)(只适用于从站)。 有效值是0-32,767。延迟以20毫秒为周期计数。缺省值是3000个周期 (1 分钟)。
TransmitRetries	SINT	GSV	无应答的重发信息次数。(只适用于主站和从站)。 有效值是 0-127。缺省值是 3。
PendingACKTimeout	DINT	SSV	应答超时属性的挂起值。
PendingDuplicateDetection	SINT	SSV	重复检测属性的挂起值。
PendingEmbeddedResponse Enable	SINT	SSV	嵌入响应属性的挂起值。
PendingENQTransmitLimit	SINT	SSV	ENQ发送限制属性的挂起值。
PendingEOTSuppression	SINT	SSV	EOT抑制属性的挂起值。
PendingErrorDetection	SINT	SSV	错误检测属性的挂起值。
PendingNormalPollGroupSize	INT	SSV	正常查询组大小属性的挂起值。
PendingMasterMessage Transmit	SINT	SSV	主信息发送属性的挂起值。
PendingNAKReceiveLimit	SINT	SSV	NAK接收限制属性的挂起值。
PendingPollingMode	SINT	SSV	查询模式属性的挂起值。

属性:	数据类型:	指令:	说明:
PendingReplyMessageWait	DINT	SSV	应答信息等待属性的挂起值。
PendingStationAddress	INT	SSV	站地址属性的挂起值。
PendingSlavePollTimeout	DINT	SSV	从站轮询超时属性的挂起值。
PendingTransmitRetries	SINT	SSV	重试发送属性的挂起值。

要应用DF1挂起属性的任何数值时：

1 使用SSV指令设置挂起属性的数值。

用户可以根据需要设定多个挂起属性，每个挂起属性的设置均使用SSV指令。

2 使用MSG指令来应用该数值。MSG指令可以应用用户设置的每个挂起属性。将MSG指令组态为：

MSG 组态选项卡:	区域:	数值:
Configuration 组态	Message Type 信息类型	CIP通用信息
	Service Code 服务代码	0d十六进制
	Object Type 对象类型	a2
	Object ID 对象ID	1
	Object Attribute 对象属性	空白
	Source 源	空白
	Number of Elements 元素数量	0
	Destination 目的地址	空白
Communication 通讯	Path 路径	至自身的通讯路径 (1, s 这里 s = 控制器的槽号)

访问 FAULTLOG 对象

FAULTLOG 对象提供控制器的故障信息。

属性：	数据类型：	指令：	说明：
MajorEvents	INT	GSV SSV	自上次该计数器被复位后，发生主要故障的次数。
MinorEvents	INT	GSV SSV	自上次该计数器被复位后，发生次要故障的次数。
MajorFaultBits	DINT	GSV SSV	各位标识引起当前主要故障的原因。 位：　意义： 1　掉电 3　I/O 4　执行指令(编程) 5　故障处理 6　看门狗 7　堆栈 8　模式改变 11　运动控制
MinorFaultBits	DINT	GSV SSV	各位标识引起当前次要故障的原因。 位：　意义： 4　执行指令(编程) 6　看门狗 9　串行口 10　电池

访问 MESSAGE 对象

用户可以通过GSV/SSV指令来访问MESSAGE对象。指定信息标签名称来确定用户要访问的MESSAGE对象。MESSAGE对象提供了建立和触发对等通讯的界面。该对象取代了PLC-5处理器中MG数据类型。

属性：	数据类型：	指令：	说明：
ConnectionPath	SINT[130]	GSV SSV	用于设置连接路径的数据。前两个字节(低字节和高字节)为按字节表示的连接路径长度。
ConnectionRate	DINT	GSV SSV	请求信息包的连接速率。
MessageType	SINT	GSV SSV	指定信息类型。 **数值：** **含义：** 0 未初始化
Port	SINT	GSV SSV	指明信息应被发送到的端口。 **数值：** **含义：** 1 背板 2 串行口
Timeout Multiplier	SINT	GSV SSV	确定连接可考虑的超时和关闭时间。 **数值：** **含义：** 0 连接超时时间为4倍的更新速率(缺省值) 1 连接超时时间为8倍的更新速率 2 连接超时时间为16倍的更新速率
UnconnectedTimeout	DINT	GSV SSV	用于所有未连接信息的以毫秒为单位的超时周期。缺省值是30，000，000毫秒(30秒)。

改变MESSAGE 属性的步骤：

1 使用一条GSV指令来获取信息类型属性并存储在一个标签中。

2 使用一条SSV指令来设定信息类型为零。

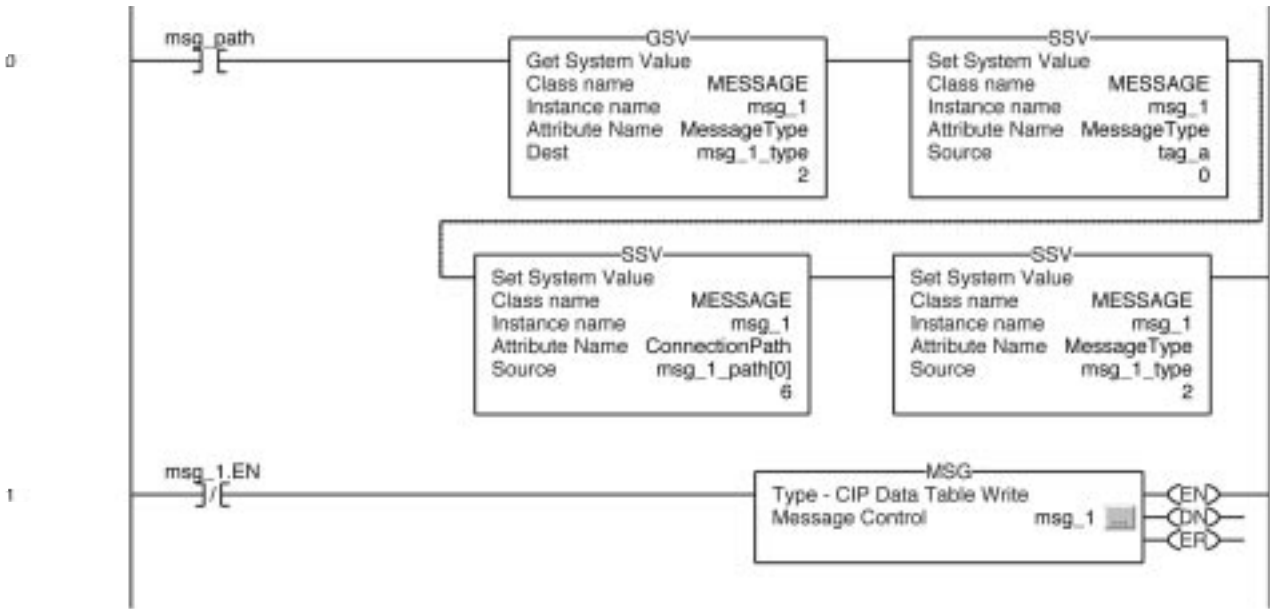
3 使用一条SSV指令来设定用户要改变的信息属性。

4 使用一条SSV指令把信息类型属性设置为步骤1中该属性原值。

举例：下例通过改变 ConnectionPath的属性，将信息传送到不同的控制器中。当msg_path是导通状态时，将msg_1信息的路径值设置成msg_1_path。这样可以将信息发送到不同控制器中。

此处:	为:
msg_1	用户需要改变其属性的信息
msg_1_type	存储信息类型属性值的标签
tag_a	存储一个0值的标签
msg_1_path	存储信息新连接路径的数组标签

梯形图



结构文本

```
IF msg_path THEN
    GSV(MESSAGE,msg_1,MessageType,msg_1_type);
    SSV(MESSAGE,msg_1,MessageType,tag_a);
    SSV(MESSAGE,msg_1,ConnectionPath,msg_1_path[0]);
    SSV(MESSAGE,msg_1,MessageType,msg_1_type);
END_IF;
IF NOT msg_1.EN THEN
    MSG(msg_1);
END_IF;
```

访问 MODULE 对象

MODULE 对象提供关于模块的状态信息。选择一个特定的模块对象，设置GSV/SSV指令的对象名称操作数为模块名称，被指定的模块必须存在于控制器管理器的I/O组态文件夹中，且必须有一个设备名。

属性：	数据类型：	指令：	说明：
EntryStatus	INT	GSV	表明指定映射入口的当前状态。当执行比较操作时，应对其低12位进行屏蔽。只有位12-15有效。 **数值**：　**含义**： 16#0000　**待机**：控制器正在上电。 16#1000　**故障**：任何模块对象与相关模块的连接失败。不能使用该数值来确定模块是否失效，因为当MODULE对象试图重新与模块连接时会周期性脱离该状态。并以测试运行状态(16#4000)代替。检测故障代码是否零以确定模块是否出错。发生故障时，故障代码和故障信息属性有效，直到故障条件被更正。 16#2000　**有效性验证**：MODULE对象在建立与模块的连接之前验证MODULE对象的完整性。 16#3000　**连接**：MODULE对象初始化与模块的连接。 16#4000　**运行**：建立起与模块的所有连接并在成功的进行数据传送。 16#5000　**关闭**：MODULE对象处于关闭与模块的所有连接过程。 16#6000　**禁止**：MODULE对象被禁止(模式属性中的禁止位被置位)。 16#7000　**等待**：该模块对象所依存的父模块对象未运行。
FaultCode	INT	GSV	如果发生故障，用于鉴别模块故障的数字。
FaultInfo	DINT	GSV	提供关于模块对象故障代码的特定信息。
ForceStatus	INT	GSV	指明强制状态。 **位**：　**意义**： 0　强制安装 (1=是, 0=否) 1　强制使能 (1=是, 0=否) 2-15　不使用

属性：	数据类型：	指令：	说明：
Instance	DINT	GSV	提供该模块对象的实例号。
LEDStatus	INT	GSV	指定在控制器前面的I/O LED指示灯的当前状态。 数值： 含义： 0 LED 灭：没有组态控制器中模块对象(在控制器管理器的I/O组态文件夹中没有模块)。 1 红色闪烁：没有MODULE对象运行。 2 绿色闪烁：至少有一个MODULE对象未运行。 3 绿色稳定：所有的MODULE对象都在运行。 注意：用户不要对该属性输入对象名，因为该属性适用于整个模块集合。
Mode	INT	GSV SSV	指明MODULE(模块)对象的当前模式。 位： 含义： 0 置位时，当控制器处于运行模式时，如果任何MODULE对象连接失败，将导致主要故障。 2 置位时，在关闭与模块的全部连接后，将导致MODULE对象进入禁止状态。

访问 MOTIONGROUP 对象

MOTIONGROUP 对象提供关于伺服模块轴的状态信息。指定运动组标签名称以确定用户需要的MOTIONGROUP。

属性:	数据类型：	指令：	说明:
Instance	DINT	GSV	提供MOTION_GROUP对象的实例号。

访问 PROGRAM 对象

PROGRAM 对象提供关于程序的状态信息。指定程序名称以确定用户要访问的PROGRAM对象。

属性:	数据类型:	指令:	说明:
DisableFlag	SINT	GSV SSV	控制程序的执行。 数值: 含义: 0 使能程序执行 1 禁止程序执行
Instance	DINT	GSV	提供该PROGRAM对象的实例号。
LastScanTime	DINT	GSV SSV	上一次程序执行所用的时间。时间以毫秒为单位。
MajorFaultRecord	DINT[11]	GSV SSV	记录该程序的主要故障 推荐用户创建自定义结构体以简化对主要故障记录属性的访问:

名称:	数据类型:	格式:	说明:
TimeLow	DINT	十进制数	故障时间戳值的低32位。
TimeHigh	DINT	十进制数	故障时间戳值的高32位。
Type	INT	十进制数	故障类型(程序，I/O等)
Code	INT	十进制数	故障的唯一代码(与故障类型有关)
Info	DINT[8]	十六进制数	故障特定信息(与故障类型和代码有关)

MaxScanTime	DINT	GSV SSV	记录的该程序执行时间的最大值。时间以毫秒为单位。
MinorFaultRecord	DINT「11」	GSV SSV	记录该程序的次要故障。 建议用户创建自定义结构体以简化对MinorFaultRecord(次要故障记录)属性的访问。

名称:	数据类型:	格式:	说明:
TimeLow	DINT	十进制数	故障时间戳值的低32位。
TimeHigh	DINT	十进制数	故障时间戳值的高32位。
Type	INT	十进制数	故障类型(程序，I/O等)
Code	INT	十进制数	故障的唯一代码(与故障类型有关)
Info	DINT「8」	十六进制数	故障特定信息(与故障类型和代码有关)
SFCRestart	INT	GSV SSV	未使用 — 保留以备将来使用

访问 ROUTINE 对象

ROUTINE 对象提供关于例程的状态信息。指定例程名称以确定用户要访问的ROUTINE对象。

属性:	数据类型:	指令:	说明:
Instance	DINT	GSV	提供该例程对象的实例号。 有效值为0-65,535。

访问 SERIALPORT 对象

SERIALPORT 对象提供一个与串行通讯端口的接口。

属性:	数据类型:	指令:	说明:
BaudRate	DINT	GSV	指定波特率。 有效值为110，300，600，1200，2400，4800，9600和19200(缺省值)。
DataBits	SINT	GSV	指定每个字符的数据位数。 数值: 含义: 7 7位数据位(仅ASCII) 8 8位数据位(缺省值)
Parity	SINT	GSV	指定校验位。 数值: 含义: 0 无奇偶校验(无缺省值) 1 奇校验(仅ASCII) 2 偶校验
RTSOffDelay	INT	GSV	在最后一个字符被传送后到关断RTS线的延时时间。 有效值为0-32,767。延时以20毫秒周期计算。缺省值是0毫秒。
RTSSendDelay	INT	GSV	接通RST线后到发送信息的第一个字符的延时时间。 有效值为0-32,767。延时以20毫秒周期计算。缺省值是0毫秒。
StopBits	SINT	GSV	设定停止位数。 数值: 含义: 1 1个停止位(缺省值) 2 2个停止位(只适用于ASCII)
PendingBaudRate	DINT	SSV	波特率属性的挂起值。
PendingDataBits	SINT	SSV	数据位属性的挂起值。
PendingParity	SINT	SSV	奇偶校验属性的挂起值。

属性:	数据类型:	指令:	说明:
PendingRTSOffDelay	INT	SSV	RST关断延时属性的挂起值。
PendingRTSSendDelay	INT	SSV	RST发送延时属性的挂起值。
PendingStopBits	SINT	SSV	停止位属性的挂起值。

应用任何SERIALPORT挂起属性值都要:

1 .用SSV指令设置挂起属性值。

如果需要可以用SSV 指令设置很多挂起属性，每条挂起属性用一条SSV指令。

2 .使用MSG指令应用此值。MSG指令可以使用用户设置的每个挂起属性。按照以下方法组态MSG指令:

MSG 组态选项卡:	区域:	数值:
Configuration 组态	Message Type 信息类型	CIP通用信息
	Service Code 服务代码	0d 十六进制
	Object Type 对象类型	6f 十六进制
	Object ID 对象ID	1
	Object Attribute 对象属性	空白
	Source源	空白
	Number of Elements 元素数量	0
	Destination 目的	空白
Communication 通讯	Path 路径	到其自己的通讯路径 (1, s 其中s = 控制器槽号)

访问 TASK 对象

TASK 对象提供关于任务的状态信息。指定任务名称以确定用户需要访问的 TASK 对象。

属性：	数据类型：	指令：	说明：	
DisableUpdateputs	DINT	GSV SSV	在任务结束时使能或禁止输出处理	
			目的：	设置属性为：
			在任务结束时使能输出处理	0
			在任务结束时禁止输出处理	1(或任一非零值)
EnableTimeOut	DINT	GSV SSV	使能或禁止事件任务的超时功能。	
			目的：	设置属性为：
			禁止超时功能	0
			使能超时功能	1(或任一非零值)
InhibitTask	DINT	GSV SSV	防止任务执行。如果任务被禁止，当控制器由编程模式转换为运行或测试模式时，控制器仍预扫描任务。	
			目的：	设置属性为：
			使能任务	0(缺省值)
			禁止任务	1(或任一非零值)
Instance	DINT	GSV	提供TASK对象的实例号。 有效值是0-31 。	
LastScanTime	DINT	GSV SSV	上次程序执行所用的时间。以毫秒为时间单位。	
MaxInterval	DINT[2]	GSV	连续执行该任务的最大时间间隔。DINT[0]包含该值的低32位；DINT[1]包含该值的高32位。	
		SSV	数值为零表明该任务的执行次数等于或小于1 。	
MaxScanTime	DINT	GSV SSV	记录的该程序执行周期的最大值，以毫秒为时间单位。	
MinInterval	DINT[2]	GSV SSV	连续两次执行该任务间的最小时间间隔。DINT[0]包括该值的低32位，DINT[1]包括该值的高32位。数值为零表明该任务的执行次数等于或小于1 。	
OverlapCount	DINT	GSV SSV	任务还在执行时被触发的次数。对事件型任务或周期型任务有效。 要清除计数，将属性设置为0 。	
Priority	INT	GSV SSV	该任务与其它任务相比的相对优先级。 有效值为0-15 。	

属性：	数据类型：	指令：	说明：	
Rate	DINT	GSV SSV	如果任务类型为：	则Rate属性指定：
			周期型	任务周期。时间以微秒为单位。
			事件型	任务的超时值。 时间以微秒为单位。
StartTime	DINT[2]	GSV SSV	该任务上次执行开始后的WALLCLOCKTIME的数值。DINT[0]包含该值的低32位，DINT[1]包含该值的高32位。	
Status	DINT	GSV SSV	提供任务的状态信息。一旦控制器设置这些位中的一位，用户必须手动清除该位。	
			决定是否：	检查该位：
			EVENT指令触发任务(仅事件型任务)。	0
			超时触发任务(仅事件型任务)。	1
			发生与该任务重叠	2
Watchdog	DINT	GSV SSV	用于执行与该任务相关的所有程序的时间限值。 时间以微秒为单位。 如果输入零，则进行如下赋值：	
			时间:	任务类型:
			0.5 秒	周期型或事件型
			5.0 秒	连续

访问 WALLCLOCKTIME 对象

WALLCLOCKTIME 对象向控制器提供一个可用于预订的时间戳。

属性：	数据类型：	指令：	说明：
CSTOffset	DINT[2]	GSV SSV	CST对象当前值的正偏移值(协调系统时间，参见页码3-39)。DINT[0]包含该值的低32位；DINT[1]包含该值的高32位。 数值以 Ãsecs为单位。缺省值为零。
CurrentValue	DINT[2]	GSV SSV	WALLCLOCKTIME的当前值。DINT[0]包含该值的低32位；DINT[1]包含该值的高32位。 该值以1972年1月1日0000时为参考。 CST和WALLCLOCKTIME对象的计算关系存储于控制器中。例如，如果将CST CurrentValue 和WALLCLOCKTIME CTSOffset相加，则结果是WALLCLOCKTIME CurrentValue。
DataTime	DINT[7]	GSV	数据和时间的可读格式为： DINT[0] 年 DINT[1] 月的整数形式(1-12) DINT[2] 日期的整数形式(1-31) DINT[3] 小时(0-23) DINT[4] 分钟(0-59) DINT[5] 秒(0-59) DINT[6] 毫秒(0-999,999)

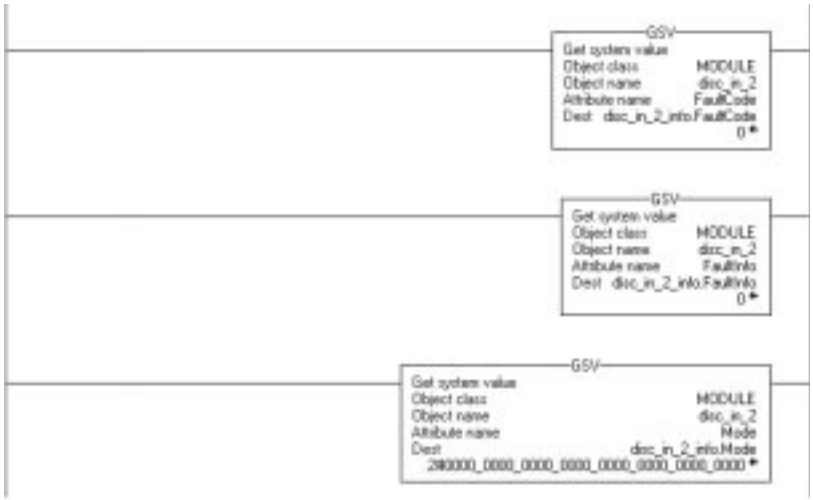
GSV/SSV指令编程实例

获取故障信息

下面是使用GSV指令获取故障信息的实例。

例1：本例从I/O模块disc_in_2 获取故障信息，并把数据存储于用户自定义结构体disc_in_2_info中。

梯形图

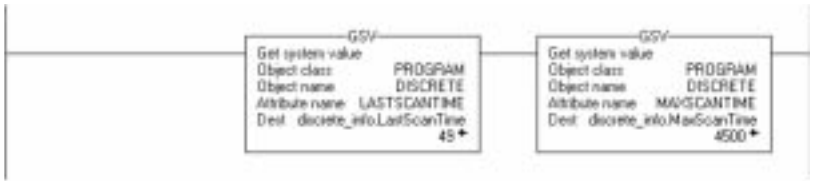


结构文本

```
GSV(MODULE,disc_in_2,FaultCode,disc_in_2_info.FaultCode);

GSV(MODULE,disc_in_2,FaultInfo,disc_in_2_info.FaultInfo);

GSV(MODULE,disc_in_2,Mode,disc_in_2info.Mode);
```

例2：本例获取关于程序discrete的状态信息并把数据存储于用户自定义结构体discrete_info中。

梯形图

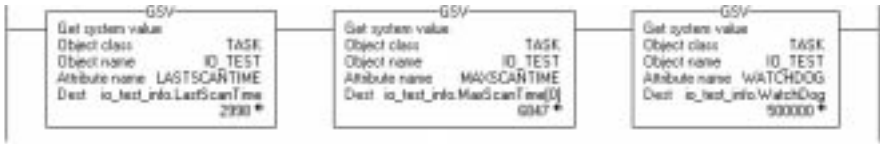


结构文本

```
GSV(PROGRAM,DISCRETE,LASTSCANTIME,
        discrete_info.LastScanTime);

GSV(PROGRAM,DISCRETE,MAXSCANTIME,discrete_info.MaxScanTime);
```

例3：本例获取关于任务IO_test的状态信息并把数据存储于用户自定义结构体io_test_info中。

梯形图



结构文本

GSV(TASK,IO_TEST,LASTSCANTIME,io_test_info.LastScanTime);

GSV(TASK,IO_TEST,MAXSCANTIME,io_test_info.MaxScanTime);

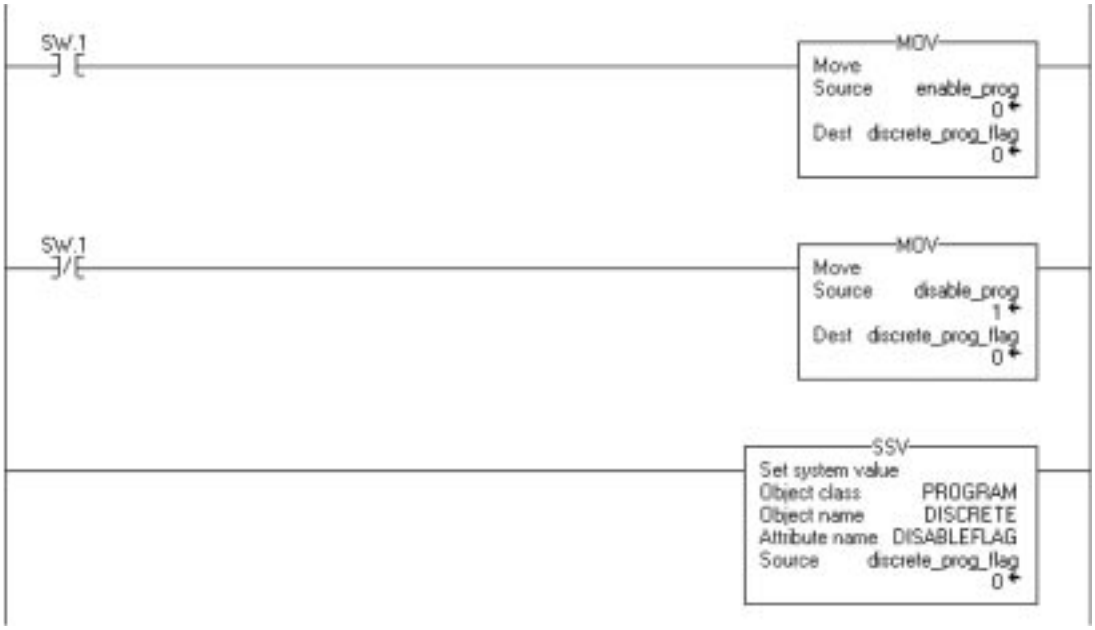
GSV(TASK,IO_TEST,WATCHDOG,io_test_info.WatchDog);

设置使能和禁止标志

下列实例用SSV指令来使能或禁止一个程序。用户还可以使用该方法来使能或禁止一个I/O模块，与使用PLC-5处理器的禁止位相似。

举例：根据SW.1的状态，将合适值放入程序discrete的disableflag属性中。

梯形图



结构文本

```
IF SW.1 THEN
        discrete_prog_flag := enable_prog;
ELSE
        discrete_prog_flag := disable_prog;
END_IF;
SSV(PROGRAM,DISCRETE,DISABLEFLAG,discrete_prog_flag);
```

立即输出(IOT)

IOT指令用于立即更新特定的输出数据(输出标签或生产者标签)。

操作数：

梯形图：

操作数：	数据类型：	格式：	说明：
Update Tag 更新标签		标签	用户需更新的标签，下列之一： • I/O模块的输出标签 • 生产者标签 不能选择标签中项或元素。 例如，Local：5：0可选，但Local：5：0.Data不可选。

结构文本

```
IOT(output_tag);
```

该操作数与梯形图中IOT指令的操作数相同。

说明：IOT指令越过输出连接的请求信息包间隔时间(RPI)并通过该连接发送最新数据。

- 输出连接是与I/O模块的输出标签或生产者标签相关联的连接。
- 如果是关于生产者标签的连接，IOT指令同时发送事件触发到消费者控制器。这样允许IOT指令触发消费者控制器中的事件任务。

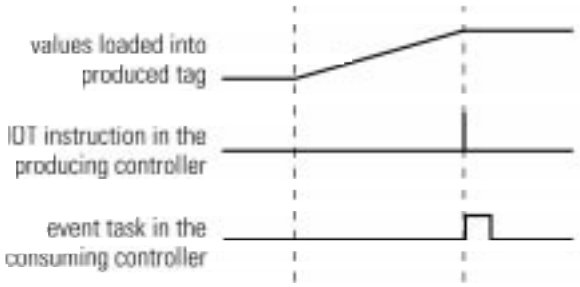
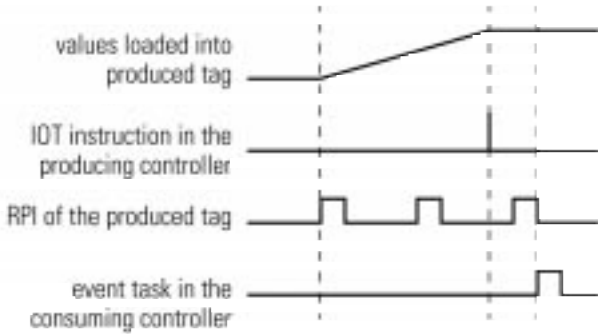
要使用IOT指令和生产者标签触发消费者控制器中事件任务，按如下组态生产者标签：

选中该复选框
组态标签通过IOT指更新事件触发

控制器之间的网络类型决定消费者控制器何时经由IOT指令接收新数据和事件触发。

控制器:	通过以下网络:	接收数据和事件触发的消费者设备:
ControlLogix	背板	立即
	EtherNet/IP 网络	立即
	ControlNet 网络	消费者标签的实际数据包间隔时间(API)内。(连接)
SoftLogix5800	用户仅可在Control Net网络上生产和消费标签	消费者标签的实际数据包间隔时间(API)内。(连接)

下表就EtherNet/IP和ControlNet网络通过IOT指令接收数据进行了比较。

EtherNet/IP网络	ControlNet网络
	

算术状态标志: 不影响

故障条件: 无

执行：

条件：	梯形图动作：	结构文本动作：
预扫描	梯级输出条件被设置为假。	不动作。
梯级输入条件为假	梯级输出条件被设置为假。	无影响
梯级输入条件为真	指令执行。 梯级输出条件被设置为真。	无影响
输入使能被置位	无影响	输入使能一直被置位。 指令执行。
指令执行	指令： • 更新特定标签连接。 • 复位连接的RPI计时器	
后期扫描	梯级输出条件被设置为假。	不动作。

例1：IOT指令执行时，该指令立即将Local:5:0标签值发送到输出模块。

梯形图

结构文本

IOT (Local:5:O);

例2：该处理器控制24号站并为下一个站(站25)生产数据。使用IOT指令发送信号并传输新数据，生产者标签按下图组态：

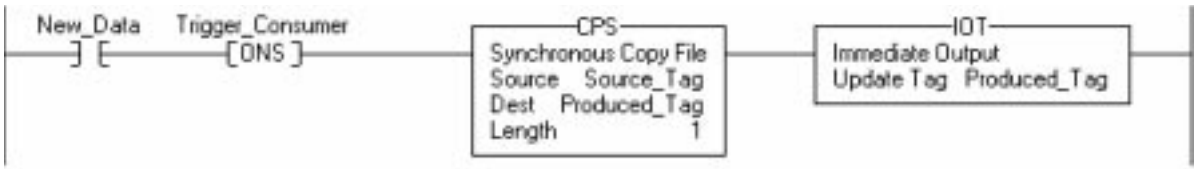
Produced_Tag被组态为经由IOT指令更新事件触发。

梯形图

如果New_Data=on, 则在一个扫描周期内将发生：

CPS指令设置Produced_Tag=Source_Tag。

IOT指令更新Produced_Tag且该将更新发送到消费者控制器(站25)。当消费型控制器接收到该更新时，它触发控制器内关联的事件任务。



结构文本

```
IF New_Data  AND NOT Trigger_Consumer THEN
      CPS (Source_Tag,Produced_Tag,1);
      IOT (Produced_Tag);
END_IF;
Trigger_Consumer := New_Data;
```

比较指令
(CMP，EQU，GEQ，GRT，LEQ，LES，LIM，MEQ，NEQ)

简介

比较指令用于用户通过表达式或专用比较指令来比较数值。

如果用户需要:	所用指令:	可用语言:	参见页码:
根据表达式比较数值	CMP	梯形图 结构文本(1)	4-2
测试两个值是否相等	EQU	梯形图 结构文本(2) 功能块	4-7
测试一个值是否大于或等于另一个值	GEQ	梯形图 结构文本(1) 功能块	4-11
测试一个值是否大于另一个值	GRT	梯形图 结构文本(1) 功能块	4-15
测试一个值是否小于或等于另一个值	LEQ	梯形图 结构文本(1) 功能块	4-19
测试一个值是否小于另一个值	LES	梯形图 结构文本(1) 功能块	4-23
测试一个值是否在另两个值之间	LIM	梯形图 结构文本(1) 功能块	4-27
屏蔽两个数值并测试这两个值是否相等	MEQ	梯形图 结构文本(1) 功能块	4-33
测试两个值是否不相等	NEQ	梯形图 结构文本(1) 功能块	4-38

(1)不存在等价的结构文本指令。使用其它结构文本编程来完成相同功能。参见该指令说明。

(2)不存在等价的结构文本指令。使用表达式中的运算符。

可以比较不同数据类型的数值，例如浮点数和整数进行比较。

如果用梯形图语言编程，则黑体的数据类型是最优数据类型。如果指令的所有操作数都使用相同的优化数据类型，则指令的执行速度快且占用内存少。典型的最优数据类型是DINT或REAL数据类型。

比较(CMP)

CMP 指令执行表达式中指定的算术运算比较。

操作数:

梯形图

操作数:	数据类型:	格式:	说明:
Expression	SINT	立即数	表达式由被运算符分开的标签和(或)立即数组成
表达式	INT **DINT** **REAL** 字符串	标签	
	SINT 或INT 数据类型的标签可以通过符号-扩充法转换成DINT 数据类型。		

结构文本

结构文本中无CMP指令，但可使用IF…THEN结构体和表达式完成相同功能。

```
IF BOOL_expression THEN
<statement>;
END_IF;
```

关于结构文本中结构体和表达式的语法信息，请参见附录C。

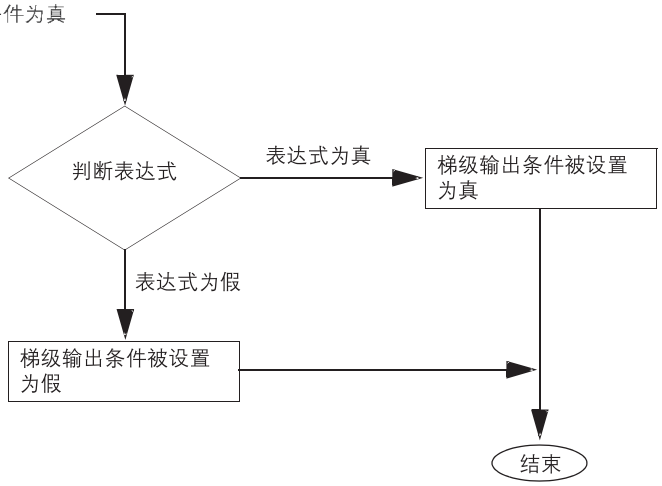
说明： 用运算符、标签和立即数定义CMP指令的表达式。如果表达式比较复杂，可以用圆括号( )将表达式分段。

与其它比较指令相比，执行CMP指令的速度稍慢而且占用更多内存。使用CMP指令的好处在于可以在一条指令中完成较复杂的比较。

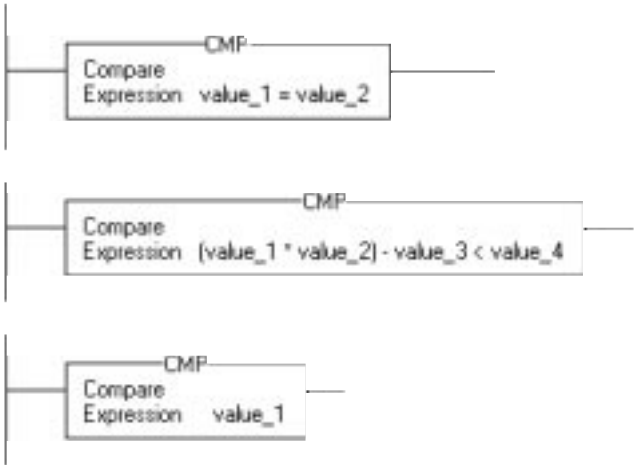
算术状态标志： 如果CMP指令的表达式中包含影响算术状态标志的运算符(例如，+，-，*，/)，该指令影响算术状态标志。

故障条件： 无

执行：

条件：	动作：
预扫描	梯级输出条件被设置为假。
梯级输入条件为假	梯级输出条件被设置为假。
梯级输入条件为真	
后期扫描	梯级输出条件被设置为假。

举例：如果CMP指令表达式的执行结果为真，则梯级输出条件被设置为真。



如果用户输入一个没有比较运算符的表达式，如value_1 + value_2，或者value_1，则指令按下列原则来判断表达式：

如果表达式的执行结果：	梯级输出条件被设置为：
非零值	真
零	假

CMP 指令表达式

CMP指令中编程表达式与FSC 指令相同。下面所描述的有效运算符、格式和运算顺序对两条指令都适用。

有效运算符

运算符:	说明:	最优数据类型:
+	加	DINT, REAL
-	减/取负	DINT, REAL
*	乘	DINT, REAL
/	除	DINT, REAL
=	等于	DINT, REAL
<	小于	DINT, REAL
<=	小于或等于	DINT, REAL
>	大于	DINT, REAL
>=	大于或等于	DINT, REAL
<>	不等于	DINT, REAL
**	指数(x 的y 次幂)	DINT, REAL
ABS	绝对值	DINT, REAL
ACS	反余弦	REAL
AND	按位与	DINT
ASN	反正弦	REAL
ATN	反正切	REAL
COS	余弦	REAL

运算符:	说明:	最优数据类型:
DEG	弧度转换成度	DINT, REAL
FRD	BCD码转换成整数	DINT
LN	自然对数	REAL
LOG	以10为底的对数	REAL
MOD	取除法模数	DINT, REAL
NOT	按位取补	DINT
OR	按位或	DINT
RAD	度转换成弧度	DINT, REAL
SIN	正弦	REAL
SQR	平方根	DINT, REAL
TAN	正切	REAL
TOD	整数转换成BCD码	DINT
TRN	截断	DINT, REAL
XOR	按位异或	DINT

格式化表达式

在表达式中的每个运算符都必须有一个或两个操作数 (标签或立即数)。按照下表确定表达式中的运算符和操作数的格式：

运算符：	格式：	举例：
一个操作数	运算符(操作数)	ABS(tag_a)
两个操作数	操作数_a 运算符 操作数_b	• tag_b + 5 • tag_c AND tag_d • (tag_e ** 2) MOD (tag_f / tag_g)

确定运算顺序

指令按规定的顺序，而不是按写入的顺序，执行写入表达式的运算符和操作数。通过用圆括号分组来重新安排运算顺序，强制指令在执行其它运算前先执行括号内的运算。

同级运算按从左到右的顺序执行。

次序：	运算符：
1	( )
2	ABS, ACS, ASN, ATN, COS, DEG, FRD, LN, LOG, RAD, SIN, SQR, TAN, TOD, TRN
3	**
4	″- (取负), NOT ″
5.	″*, /, MOD″
6.	″<, <=, >, >=, =″
7.	″- (减), +″
8.	AND
9.	XOR
10.	OR

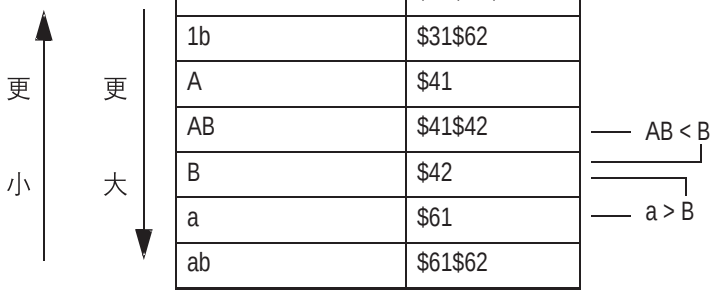
在表达式中使用字符串

可使用梯形图或结构文本表达式比较字符串数据类型。要在表达式中使用字符串，应遵循下列规则：

- 可通过一个表达式比较两个字符串标签。
- 不能在表达式中直接输入ASCII字符。
- 只允许使用下列运算符

运算符：	说明：
=	等于
<	小于
<=	小于或等于
>	大于
>=	大于或等于
<>	不等于

- 如果字符串标签中字符匹配，则字符串相等。
- ASCII 字符区分大写和小写。如大写的 ″A″ ($41) 与小写的 ″a″ ($61)不相等。
- 用字符的十六进制值来确定一个字符串是否小于或大于另一个字符串。对于字符的十六进制代码参见本手册封底。
- 如果两个字符串保存在电话号码簿中，字符串的顺序来确定哪个更大。



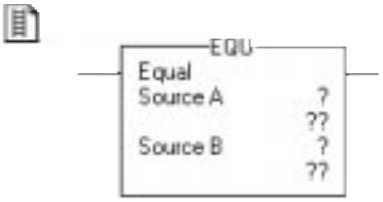
ASCII字符	十六进制代码
1ab	$31$61$62
1b	$31$62
A	$41
AB	$41$42
B	$42
a	$61
ab	$61$62

等于 (EQU)

EQU 指令检测源A的值是否等于源B的值。

操作数：

梯形图



操作数：	数据类型：	格式：	说明：
Source A 源A	SINT INT **DINT** **REAL** 字符串	立即数 标签	要与源B比较的数值
Source B 源B	SINT INT **DINT** **REAL** 字符串	立即数 标签	要与源A比较的数值

- 如果输入的是SINT或INT数据类型的标签，则按照符号-扩充的方法将其转换成DINT数据类型。
- REAL数据类型的数值几乎不能完全相等。因此如果需要确定两个REAL数据类型的数值是否相等，可以用LIM指令。
- 字符串数据类型是指：
  —缺省值STRING数据类型
  —任何由用户自定义的新字符串数据类型。
- 如果要比较字符串中的字符，则源A和源B中都要输入字符串数据类型标签。

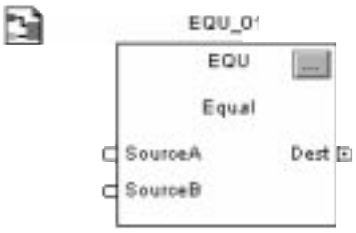
结构文本

```
IF sourceA = sourceB THEN
    <statements>;
```

使用等号 ″=″ 作为表达式中的运算符。该表达式判断sourceA是否等于sourceB。

关于结构文本中表达式的语法信息，请参见附录C。

功能块



操作数：	数据类型：	格式：	说明：
EQU 标签	FBD_COMPARE	结构体	EQU 结构体

FBD_COMPARE结构体

输入参数:	数据类型:	说明:
EnableIn	BOOL	使能输入。如果处于清零状态，则指令不执行而且输出不更新。 缺省状态是置位。
SourceA	REAL	要与源B 进行比较的数值。 有效值=任一浮点数
SourceB	REAL	要与源A进行比较的数值。 有效值=任一浮点数
输出参数:	数据类型:	说明:
EnableOut	BOOL	指令产生一有效结果。
Dest	BOOL	指令的执行结果。等同于用梯形图语言编程EQU指令的梯级输出条件。

说明：可以用EQU 指令比较两个数值或两个ASCII字符串。如果比较的是字符串，则：

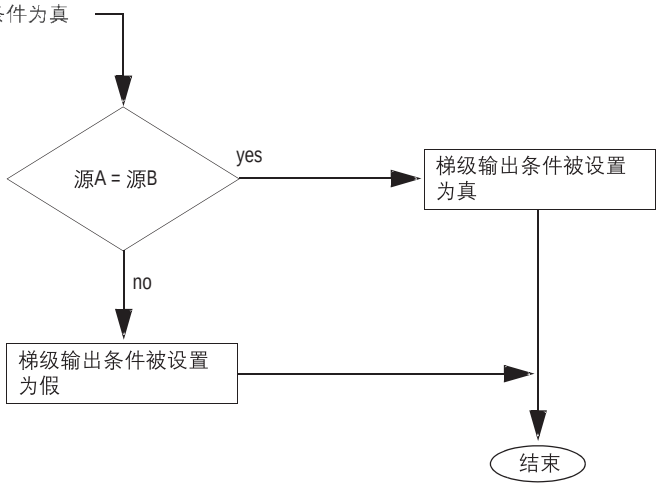
- 如果字符串标签中字符匹配，则字符串相等。
- ASCII 字符区分大写和小写，如大写的 ″A″ ($41) 与小写的 ″a″ ($61)是不相等。

算术状态标志：不影响

故障条件：无

执行：

梯形图

条件：	动作：
预扫描	梯级输出条件设置为假。
梯级输入条件为假	梯级输出条件设置为假。
梯级输入条件为真	
后期扫描	梯级输出条件被设置为假。

功能块

条件：	动作：
预扫描	不动作。
首次扫描指令	不动作。
首次执行指令	不动作。
EnableIn被清零	清零EnableOut。
EnableIn被置位	执行指令。 置位EnableOut。
后期扫描	不动作。

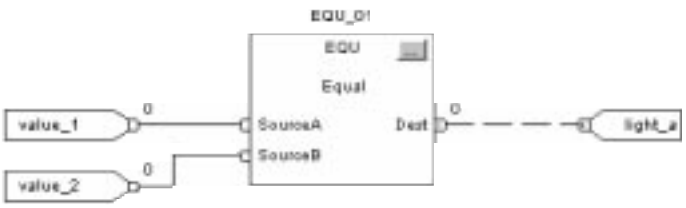
举例：如果value_1等于value_2，则置位light_a。如果value_1不等于value_2，则清零light_a。

梯形图

结构文本

```
light_a := (value_1 = value_2);
```

功能块

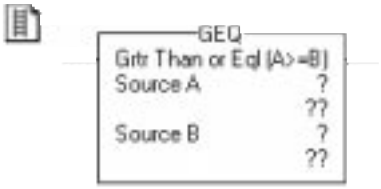


大于或等于(GEQ)

GEQ 指令测试源A的值是否大于或等于源B的值。

操作数：

梯形图



操作数：	数据类型：	格式：	说明：
Source A 源A	SINT INT **DINT** **REAL** 字符串	立即数 标签	要与源B比较的数值
Source B 源B	SINT INT **DINT** **REAL** 字符串	立即数 标签	要与源A比较的数值

- 如果输入的是SINT或INT数据类型的标签，则按照符号-扩充的方法将其转换成DINT数据类型。
- 字符串数据类型为：
  —缺省字符串数据类型
  —任何由用户自定义的字符串类型
- 如果比较的是字符串中的字符，则源A和源B 中都要输入字符串数据类型标签。

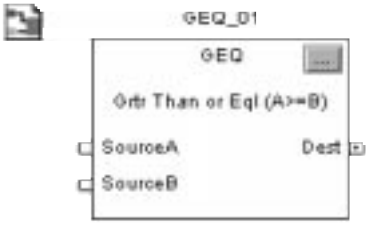
结构文本

```
IF sourceA >= sourceB THEN
    <statements>;
```

用大于等于号 ″>=″ 作为表达式中的运算符。该表达式判断sourceA的值是否大于或等于sourceB的值。

关于结构文本中表达式的语法信息，参见附录C。

功能块



操作数：	数据类型：	格式：	说明：
GEQ标签	FBD_COMPARE	结构体	GEQ结构体

FBD_COMPARE结构体

输入参数:	数据类型:	说明:
EnableIn	BOOL	使能输入。如果是清零状态，则指令不执行而且输出不更新。 缺省状态是置位。
SourceA 源A	REAL	要与源B进行比较的数值。 有效值=任一浮点数
SourceB 源B	REAL	要与源A进行比较的数值。 有效值=任一浮点数
输出参数:	**数据类型:**	**说明:**
EnableOut	BOOL	指令产生一有效结果。
Dest	BOOL	指令的执行结果。等同于用梯形图语言编程的GEQ指令的梯级输出条件。

说明: GEQ 指令测试源A的值是否大于或等于源B的值。

如果比较字符串，则：

- 用字符的十六进制值来确定一个字符串是否小于或大于另一个字符串。对于字符的十六进制代码，参见本手册封底。
- 如果两个字符串存储于电话号码簿，则以字符串的顺序来确定哪个更大。

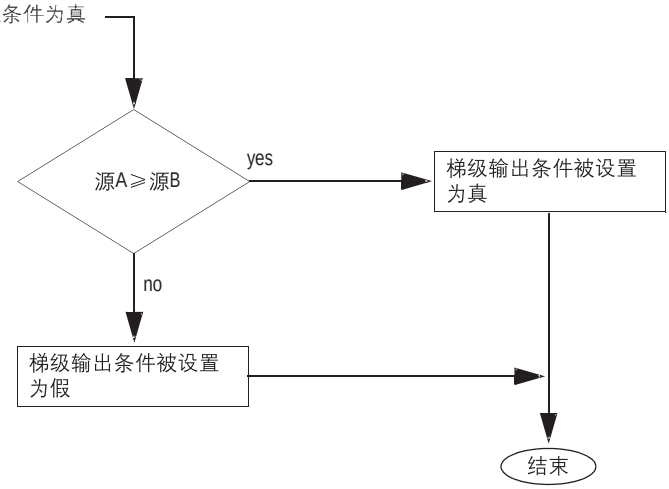
ASCII字符	十六进制代码	
1ab	$31$61$62	
1b	$31$62	
A	$41	
AB	$41$42	AB < B
B	$42	
a	$61	a > B
ab	$61$62	

算术状态标志: 算术状态标志： 不影响

故障条件: 故障条件： 无

执行：

梯形图

条件：	**动作：**
预扫描	梯级输出条件被设置为假。
梯级输入条件为假	梯级输出条件被设置为假。
梯级输入条件为真 	
后期扫描	梯级输出条件被设置为假。

功能块

条件：	**动作：**
预扫描	不动作。
首次扫描指令	不动作。
首次执行指令	不动作。
EnableIn 被清零	清零EnableOut。
EnableIn 被置位	执行指令。 置位EnableOut。
后期扫描	不动作。

举例：如果value_1大于或等于value_2 ，则置位light_b。如果value_1小于value_2 ，则清零light_b。

梯形图

结构文本

```
light_b := (value_1 >= value_2);
```

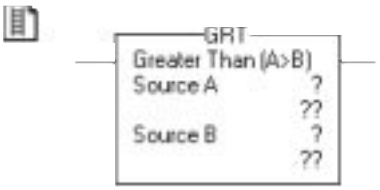
功能块

大于(GRT)

GRT 指令测试源A的值是否大于源B的值。

操作数：

梯形图



操作数：	数据类型：	格式：	说明：
Source A 源A	SINT INT **DINT** **REAL** 字符串	立即数 标签	要与源B进行比较的数值
Source B 源B	SINT INT **DINT** **REAL** 字符串	立即数 标签	要与源A进行比较的数值

- 如果输入的是SINT或INT数据类型的标签，则按照符号-扩充的方法将其转换成DINT数据类型。
- 字符串数据类型是指：
  —缺省字符串数据类型
  —任何由用户自定义的字符串类型
- 如果要测试字符串中字符，则源A和源B中都要输入字符串标签。

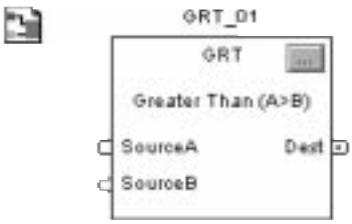
结构文本

```
IF sourceA > sourceB THEN
    <statements>;
```

使用大于号“>”作为表达式内的运算符。该表达式判断sourceA的值是否大于sourceB的值。

关于结构文本中表达式的语法信息，参见附录C。

功能块



操作数：	数据类型：	格式：	说明：
GRT标签	FBD_COMPARE	结构体	GRT 结构体

FBD_COMPARE结构体

输入参数:	数据类型:	说明:
EnableIn	BOOL	使能输入。如果是清零状态，则指令不执行而且输出不更新。 缺省状态是置位。
SourceA 源A	REAL	要与源B进行比较的数值。 有效值=任一浮点数
SourceB 源B	REAL	要与源A进行比较的数值。 有效值=任一浮点数
输出参数:	**数据类型:**	**说明:**
EnableOut	BOOL	指令产生一有效结果。
Dest	BOOL	指令的执行结果。等同于用梯形图语言编程GRT指令的梯级输出条件。

说明: GRT 指令测试源A的值是否大于源B的值。

如果比较的是字符串，则:

- 用字符的十六进制值来确定一个字符串是否小于或大于另一个字符串。对于字符的十六进制代码，参见本手册封底。
- 如果两个字符串存储于电话号码簿，则用字符串的顺序来确定哪个更大。

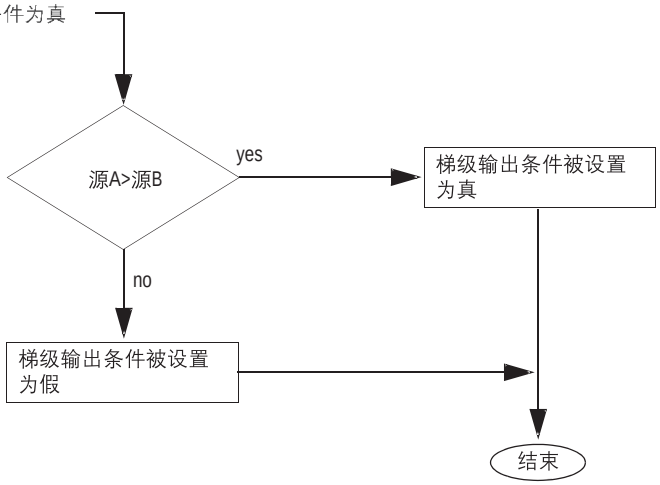
ASCII字符	十六进制代码	
1ab	$31$61$62	
1b	$31$62	
A	$41	
AB	$41$42	AB < B
B	$42	
a	$61	a > B
ab	$61$62	

算术状态标志: 不影响

故障条件: 无

执行：

梯形图

条件：	动作：
预扫描	梯级输出条件被设置为假。
梯级输入条件为假	梯级输出条件被设置为假。
梯级输入条件为真	
后期扫描	梯级输出条件被设置为假。

功能块

条件：	动作：
预扫描	不动作。
首次扫描指令	不动作。
首次执行指令	不动作。
EnableIn被清零	清零EnableOut。
EnableIn被置位	执行指令。 置位EnableOut。
后期扫描	不动作。

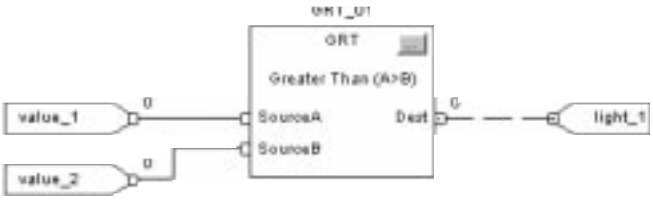
举例：如果value_1大于value_2 ，则置位light_1。如果value_1小于或等于value_2，则清除light_1。

梯形图

结构文本

```
light_1 := (value_1 > value_2);
```

功能块

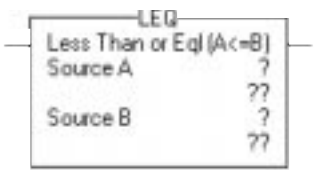


小于或等于(LEQ)

LEQ 指令测试源A的值是否小于或等于源B的值。

操作数：

梯形图



操作数：	数据类型：	格式：	说明：
Source A	SINT INT **DINT** **REAL** 字符串	立即数 标签	源A—要与源B进行比较的数值
Source B	SINT INT **DINT** **REAL** 字符串	立即数 标签	源B—要与源A进行比较的数值

- 如果输入的是SINT或INT数据类型的标签，则按照符号-扩充方法将其转换成DINT数据类型。
- 字符串数据类型是指：
  —缺省字符串数据类型
  —任何由用户自定义的字符串类型
- 如果要测试字符串中的字符，则源A和源B中都要输入字符串数据类型标签。

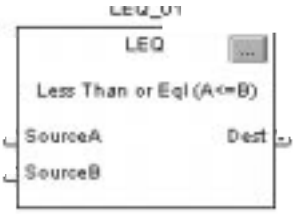
结构文本

```
IF sourceA <= sourceB THEN
    <statements>;
```

使用小于等于号 ″<=″ 号作为表达式内的运算符。该表达式判断sourceA的值是否小于或等于sourceB的值。

关于结构文本中表达式的语法信息，参见附录C。

功能块



操作数：	数据类型：	格式：	说明：
LEQ标签	FBD_COMPARE	结构体	LEQ结构体

FBD_COMPARE结构体

输入参数:	数据类型:	说明:
EnableIn	BOOL	使能输入。如果是清零状态，则指令不执行而且输出不更新。 缺省状态是置位。
SourceA 源A	REAL	要与源B进行比较的数值。 有效值=任一浮点数
SourceB 源B	REAL	要与源A进行比较的数值。 有效值=任一浮点数
输出参数:	**数据类型:**	**说明:**
EnableOut	BOOL	指令产生一有效结果。
Dest	BOOL	指令的执行结果。等同于用梯形图语言编程的LEQ指令的梯级输出条件。

说明: LEQ指令测试源A的值是否小于或等于源B的值。

如果比较字符串，则:

- 用字符的十六进制值来确定一个字符串是否小于或大于另一个字符串。对于字符的十六进制代码参见本手册封底。
- 如果两个字符串存储于电话号码簿，则用字符串的顺序来确定哪个更大。

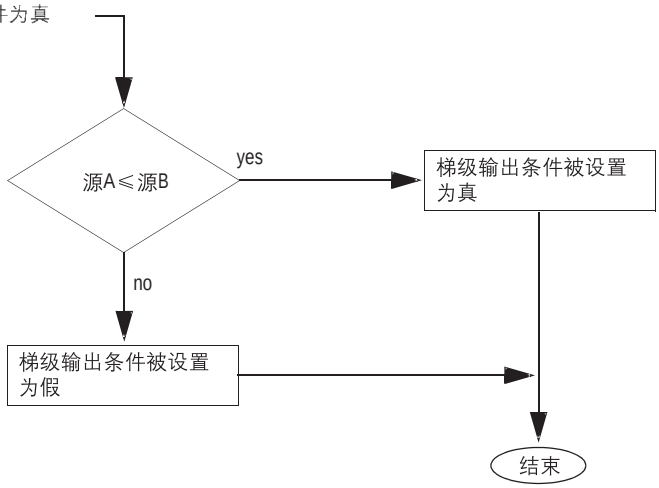
ASCII字符	十六进制代码	
1ab	$31$61$62	
1b	$31$62	
A	$41	
AB	$41$42	AB < B
B	$42	
a	$61	a > B
ab	$61$62	

算术状态标志: 不影响

故障条件: 无

执行：

梯形图

条件：	**动作：**
预扫描	梯级输出条件被设置为假。
梯级输入条件为假	梯级输出条件被设置为假。
梯级输入条件为真	
后期扫描	梯级输出条件被设置为假。

功能块

条件：	**动作：**
预扫描	不动作。
首次扫描指令	不动作。
首次执行指令	不动作。
EnableIn被清零	清零EnableOut。
EnableIn被置位	执行指令。 置位EnableOut。
后期扫描	不动作。

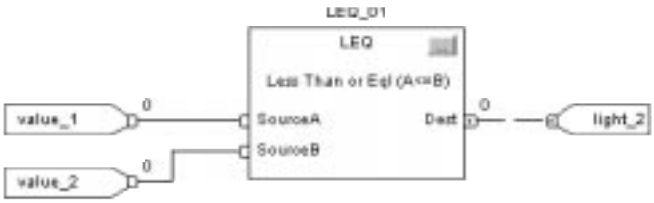
举例：如果value_1小于或等于value_2 ，则置位light_2。如果value_1大于value_2 ，则清除light_2。

梯形图

结构文本

```
light_2 := (value_1 <= value_2);
```

功能块

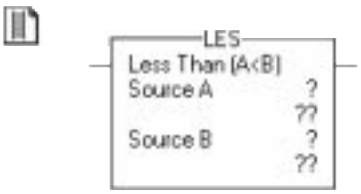


小于(LES)

LES 指令测试源A的值是否小于源B的值。

操作数：

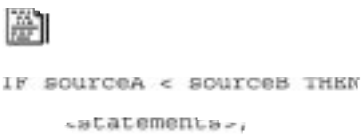
梯形图



操作数：	数据类型：	格式：	说明：
Source A 源A	SINT INT **DINT** **REAL** 字符串	立即数 标签	要与源B进行比较的数值
Source B 源B	SINT INT **DINT** **REAL** 字符串	立即数 标签	要与源A进行比较的数值

- 如果输入的是SINT或INT数据类型的标签，则按照符号-扩充方法将其转换成DINT数据类型。
- 字符串数据类型是指：
  —缺省字符串数据类型
  —任何由用户自定义的字符串类型
- 如果要测试字符串标签中字符，则源A和源B中都要输入字符串类型标签。

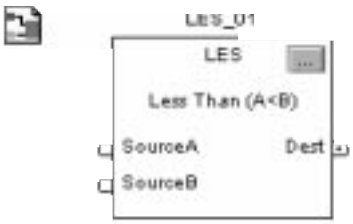
结构文本



用小于号 "<" 作为表达式内的运算符。该表达式判断sourceA的值是否小于sourceB的值。

关于结构文本中表达式的语法信息，参见附录C。

功能块



操作数：	数据类型：	格式：	说明：
LES标签	FBD_COMPARE	结构体	LES 结构体

FBD_COMPARE结构体

输入参数:	数据类型:	说明:
EnableIn	BOOL	使能输入。如果是清零状态，则指令不执行而且输出不更新。 缺省状态是置位。
SourceA 源A	REAL	要与源B 进行比较的数值。 有效值=任一浮点数
SourceB 源B	REAL	要与源A进行比较的数值。 有效值=任一浮点数
输出参数:	**数据类型:**	**说明:**
EnableOut	BOOL	指令产生一有效结果。
Dest	BOOL	指令的执行结果。等同于用梯形图语言编程的LES指令的梯级输出条件。

说明: LES 指令测试源A的值是否小于源B的值。

如果比较字符串，则:

- 用字符的十六进制值来确定一个字符串是否小于或大于另一个字符串。对于字符的十六进制代码参见本手册的封底。
- 如果两个字符串存储于电话号码簿，则根据字符串的顺序来确定哪个更大。

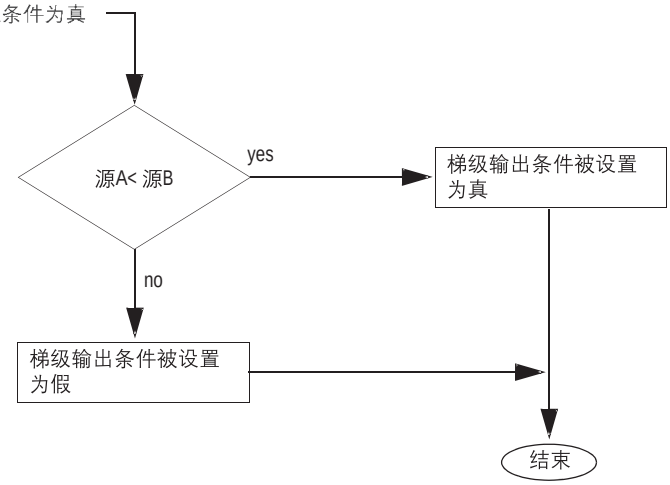
ASCII字符	十六进制代码	
1ab	$31$61$62	
1b	$31$62	
A	$41	
AB	$41$42	AB < B
B	$42	
a	$61	a > B
ab	$61$62	

算术状态标志: 不影响

故障条件: 无

执行：

梯形图

条件：	动作：
预扫描	梯级输出条件被设置为假。
梯级输入条件为假	梯级输出条件被设置为假。
梯级输入条件为真	
后期扫描	梯级输出条件被设置为假。

功能块

条件：	动作：
预扫描	不动作。
首次扫描指令	不动作。
首次执行指令	不动作。
EnableIn 为假	清零EnableOut。
EnableIn 为真	执行指令。 置位EnableOut。
后期扫描	不动作。

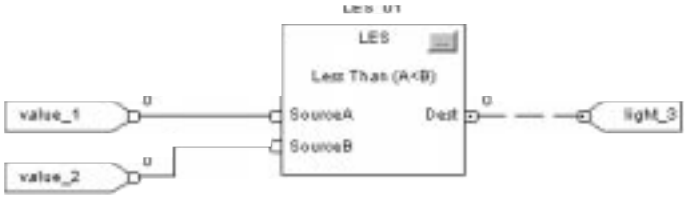
举例：如果value_1小于value_2，置位light_3。如果value_1大于或等于value_2，清零light_3。

梯形图

结构文本

```
light_3 := (value_1 < value_2);
```

功能块

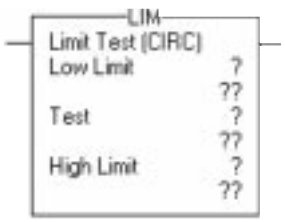


极限比较(LIM)

LIM指令检测被测值是否在上限值与下限值之间。

操作数：

梯形图



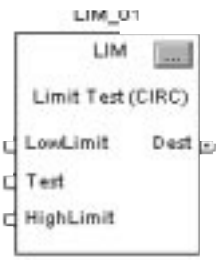
操作数：	数据类型：	格式：	说明：
Low limit	SINT	立即数	下限值
	INT	标签	
	DINT		
	REAL		
	SINT或INT数据类型的标签按照符号-扩充的方法将其转换成DINT数据类型。		
Test	SINT	立即数	被测值
	INT	标签	
	DINT		
	REAL		
	SINT或INT数据类型的标签按照符号-扩充的方法将其转换成DINT数据类型。		
High limit	SINT	立即数	上限值
	INT	标签	
	DINT		
	REAL		
	SINT或INT数据类型的标签按照符号-扩充的方法将其转换成DINT数据类型。		

结构文本

结构文本中无LIM指令，但用户可使用结构文本完成相同功能。

```
IF (LowLimit <= HighLimit AND
        (Test >= LowLimit AND Test <= HighLimit)) OR
        (LowLimit >= HighLimit AND
        (Test <= LowLimit OR Test >= HighLimit)) THEN

            <statement>;

END_IF;
```

功能块



操作数：	数据类型：	格式：	说明：
LIM标签	FBD_LIMIT	结构体	LIM结构体

FBD_LIMIT结构体

输入参数:	数据类型:	说明:
EnableIn	BOOL	如果是清零状态，则指令不执行而且输出不更新。 如果置位，指令按操作说明执行。 缺省状态是置位。
LowLimit	REAL	下限值。 有效值=任一浮点数
Test	REAL	被测值。 有效值=任一浮点数
输出参数:	**数据类型:**	**说明:**
EnableOut	BOOL	指令产生一有效结果。
Dest	BOOL	指令执行结果。等同于梯形图语言中LIM指令的梯级输出条件。
HighLimit	REAL	上限值。 有效值=任一浮点数

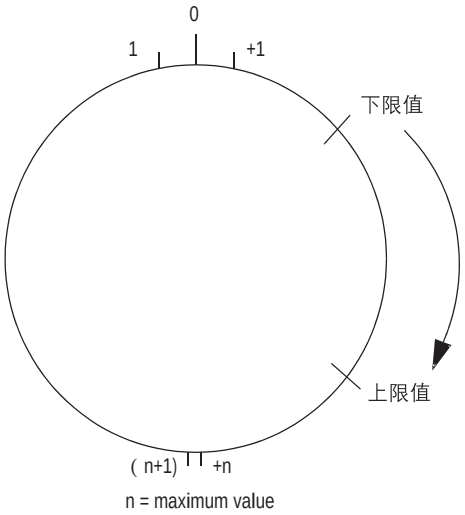
说明: LIM 指令检测被测值是否在上限值和下限值之间。

如果下限值:	被测值:	梯级输出条件:
小于等于High Limit	等于或在上限值与下限值之间	真
	不等于或在上限值和下限值之外	假
大于等于High Limit	等于或在上限值和下限值之外	真
	不等于或在上限值和下限值之间	假

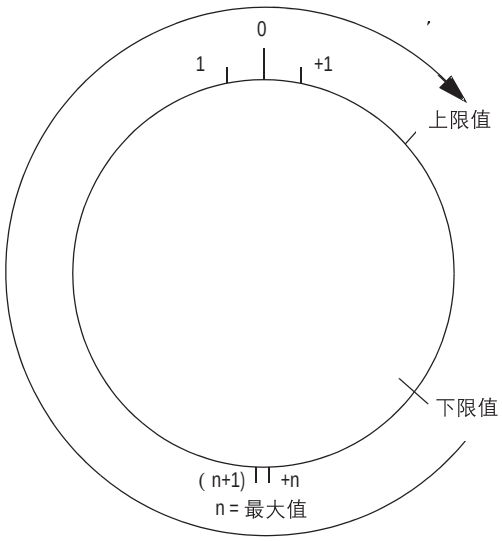
对于带符号整数，当最高有效位被置位时，就从最大正数值“返回”到最小负数值。例如，16位整数 (INT数据类型)，最大正整数是32,767，用十六进制表示成16#7FFF
(从0到14位都被置位)。如果对此数值加1，则计算结果是16#8000 (位15被置位)。有符号整数的十六进制16#8000与十进制的-32,768相等。从该数值继续增加直到全部16位都被置位为止，即16#FFFF，等于十进制的-1。

带符号数的这一变化可用循环数字线表示(参见下图)。LIM 指令从下限开始按顺时针方向增加直到达到上限值。在顺时针方向的任何下限到上限范围内的值都会将梯级输出条件设置为真。在顺时针方向的任何上限到下限范围内的测试值都会将梯级输出条件设置为假。

如果被测值等于或在下限值与上限值之间，则指令为真。



如果被测值等于或在下限与上限值之外，则指令为真。

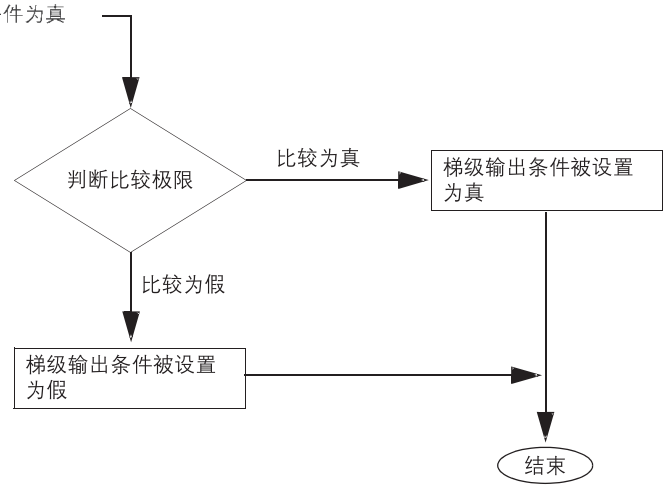


算术状态标志：不影响

故障条件：无

执行：

梯形图

条件：	动作：
预扫描	梯级输出条件被设置为假。
梯级输入条件为假	梯级输出条件被设置为假。
梯级输入条件为真	
后期扫描	梯级输出条件被设置为假。

功能块

条件：	动作：
预扫描	不动作。
首次扫描指令	不动作。
首次执行指令	不动作。
EnableIn被清零	清零EnableOut，指令不执行，输出不更新。
EnableIn被置位	执行指令。 置位EnableOut。
后期扫描	不动作。

例1：下限值≤上限值：

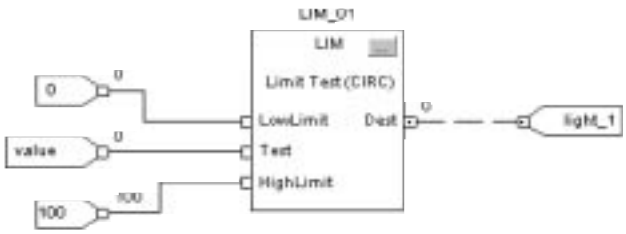
如果0 ≤value ≥100，则置位 light_1。如果value <0或value>100，则清零 light_1。

梯形图

结构文本

```
IF (value <= 100 AND(value >= 0 AND value <= 100)) OR
        (value >= 100 AND value <= 0 OR value >= 100)) THEN

          light_1 := 1;

ELSE

          light_1 := 0;

END_IF;
```

功能块



例2：下限值≥上限值:

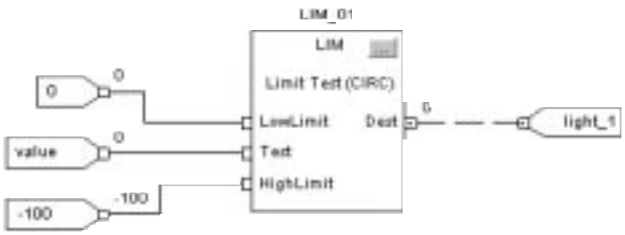
如果 value ≥0 或 value ≤-100，则置位light_1。如果 value<0 或 value>-100，则清零light_1。

梯形图

结构文本

```
IF (0 <= -100 AND value >= 0 AND value <= -100)) OR
     (0 >= -100 AND(value <= 0 OR value >= -100)) THEN

          light_1 := 1;

ELSE

          light_1 := 0;

END_IF;
```

功能块

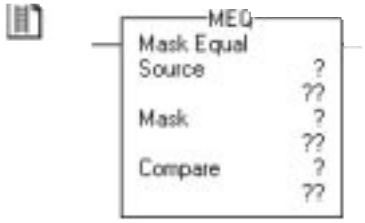


屏蔽等于 (MEQ)

MEQ指令通过屏蔽字使源值和参考值通过并进行比较。

操作数：

梯形图



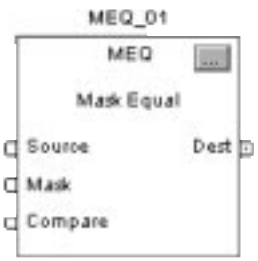
操作数：	数据类型：	格式：	说明：
Source 源值	SINT INT DINT SINT或INT数据类型的标签按照零-填充法转换成DINT数据类型。	立即数 标签	需要与参考值进行比较的数值
Mask 屏蔽值	SINT INT DINT SINT或INT数据类型的标签按照零-填充法转换成DINT数据类型。	立即数 标签	定义需要屏蔽或通过的位
Compare 参考值	SINT INT DINT SINT或INT数据类型的标签按照零-填充法转换成DINT数据类型。	立即数 标签	要与源值进行比较的数值

结构文本

结构文本中无 MEQ 指令，但用户可使用结构文本完成相同功能。

```
IF (Source AND Mask) = (Compare AND Mask) THEN

        <statement>;

END_IF;
```

功能块



操作数：	数据类型：	格式：	说明：
MEQ标签	FBD_MASK_EQUAL	结构体	MEQ 结构体

FBD_MASK_EQUAL结构体

输入参数:	数据类型:	说明:
EnableIn	BOOL	如果是清零状态，则不执行指令而且不更新输出。 如果是置位状态，指令按操作说明执行。 缺省值是置位。
Source 源值	DINT	需要与参考值进行比较的数值。 有效值=任一整数
Mask 屏蔽值	DINT	定义需要屏蔽的位。 有效值=任一整数
Compare 参考值	DINT	要与源值进行比较的数值。 有效值=任一整数
输出参数:	**数据类型:**	**说明:**
EnableOut	BOOL	指令产生一有效结果。
Dest	BOOL	指令的执行结果。等同于用梯形图语言编程的MEQ指令的梯级输出条件。

说明: 如果屏蔽位是 ″1″ 表示数据可以通过。 ″0″ 表示数据被阻滞。通常源、屏蔽位和参考值是同一数据类型。

如果用混合整数类型，则指令用0值填充较小整数类型的高位，以使它们与最大数据类型有相同的长度。

输入立即数作为屏蔽值

如果以立即数做屏蔽值，则编程软件缺省输入的数值为十进制。如果希望用其它格式输入屏蔽值，则需要在数值之前按下列方式加相应的前缀。

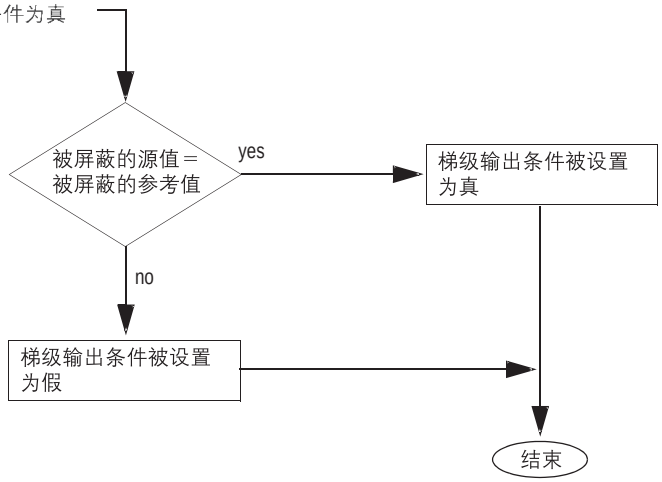
前缀:	说明:
16#	十六进制 例如：16#0F0F
8#	八进制 例如：8#16
2#	二进制 例如：2#00110011

算术状态标志: 不影响

故障条件: 无

执行：

梯形图

条件：	动作：
预扫描	梯级输出条件被设置为假。
梯级输入条件为假	梯级输出条件被设置为假。
梯级输入条件为真 	
后期扫描	梯级输出条件被设置为假。

功能块

条件：	动作：
预扫描	不动作。
首次扫描指令	不动作。
首次执行指令	不动作。
EnableIn被清零	清零EnableOut，指令不执行，输出不更新。
EnableIn被置位	执行指令。 置位EnableOut。
后期扫描	不动作。

例1：如果通过屏蔽的value_1等于通过屏蔽的value_2，则置位light_1。如果通过屏蔽的value_1不等于通过屏蔽的value_2，则清零light_1。该实例表明通过屏蔽的值相等。屏蔽值中0的禁止指令比较该位(在本例中用x表示)。

值_1	0	1	0	1	0	1	0	1	1	1	1	1	1	1	1	1
屏蔽位_1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0
通过屏蔽的值_1	0	1	0	1	0	1	0	1	1	1	1	1	x	x	x	x

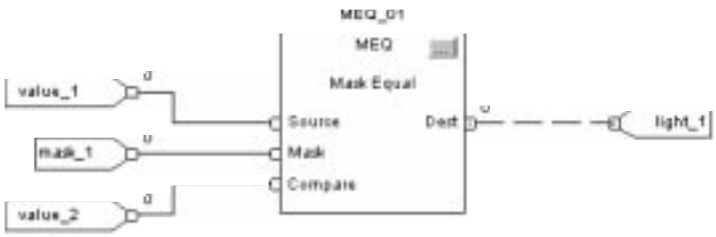
值_2	0	1	0	1	0	1	0	1	1	1	1	1	0	0	0	0
屏蔽位_1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0
通过屏蔽的值_2	0	1	0	1	0	1	0	1	1	1	1	1	x	x	x	x

梯形图

结构文本

```
light_1 := ((value_1 AND mask_1)=(value_2 AND mask_2));
```

功能块



例1：如果通过屏蔽的value_1等于通过屏蔽的value_2，则置位light_1。如果通过屏蔽的value_1不等于通过屏蔽的value_2，则清零light_1。该实例表明通过屏蔽的值不相等。屏蔽值中0的禁止指令比较该位 (在本例中用 x 表示)。

值_1	0	1	0	1	0	1	0	1	1	1	1	1	1	1	1	1
屏蔽位_1	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1
通过屏蔽的值_1	x	x	x	x	x	x	x	x	x	x	x	x	1	1	1	1

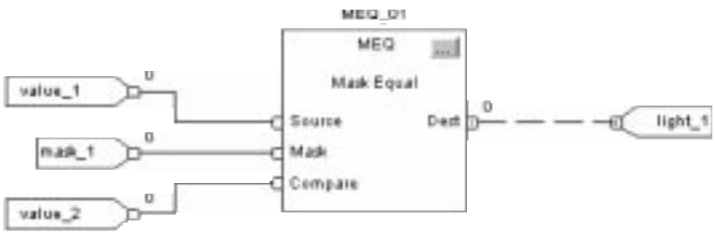
值_2	0	1	0	1	0	1	0	1	1	1	1	1	0	0	0	0
屏蔽位_1	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1
通过屏蔽的值_2	x	x	x	x	x	x	x	x	x	x	x	x	0	0	0	0

梯形图

结构文本

```
light_1 := ((value_1 AND mask_1)=(value_2 AND mask_2));
```

功能块

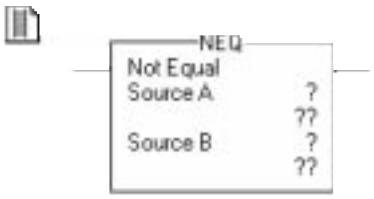


不等于(NEQ)

NEQ 指令测试源A的值是否不等于源B的值。

操作数：

梯形图



操作数：	数据类型：	格式：	说明：
Source A	SINT INT **DINT** **REAL** 字符串	立即数 标签	源A—要与源B进行比较的数值
Source B	SINT INT **DINT** **REAL** 字符串	立即数 标签	源B—要与源A进行比较的数值

- 如果输入的是SINT或INT数据类型的标签，则按照符号-扩充的方法将其转换成DINT数据类型。
- 字符串数据类型是指：
  —缺省字符串数据类型
  —任何由用户自定义的字符串类型
- 如果要测试字符串中的字符，则源A和源B中都要输入字符串数据类型标签。

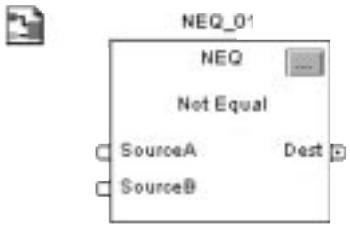
结构文本

```
IF sourceA <> sourceB THEN
    <statements>;
```

同时使用小于和大于号“<>”作为表达式内的运算符。该表达式判断sourceA是否不等于sourceB。

关于结构文本中表达式的语法信息，参见附录C。

功能块



操作数：	数据类型：	格式：	说明：
NEQ标签	FBD_COMPARE	结构体	NEQ 结构体

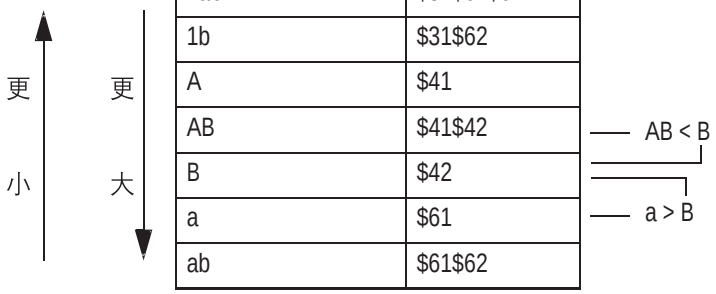
FBD_COMPARE结构体

输入参数:	数据类型:	说明:
EnableIn	BOOL	使能输入。如果是清零状态，则不执行指令而且不更新输出。 缺省状态是置位。
SourceA 源A	REAL	要与源B进行比较的数值 有效值=任一浮点数
SourceB 源B	REAL	要与源A进行比较的数值 有效值=任一浮点数
输出参数:	**数据类型:**	**说明:**
EnableOut	BOOL	指令产生一有效结果。
Dest	BOOL	指令的执行结果。等同于用梯形图语言编程的NEQ指令的梯级输出条件。

说明：NEQ 指令测试源A的值是否不等于源B的值。

如果比较的是字符串，则：

- 如果两个字符串标签内的字符相不匹配，则字符串不相等。
- ASCII 字符区分大写和小写，如大写 ″A″ ($41) 与小写 ″a″ ($61)不相等。



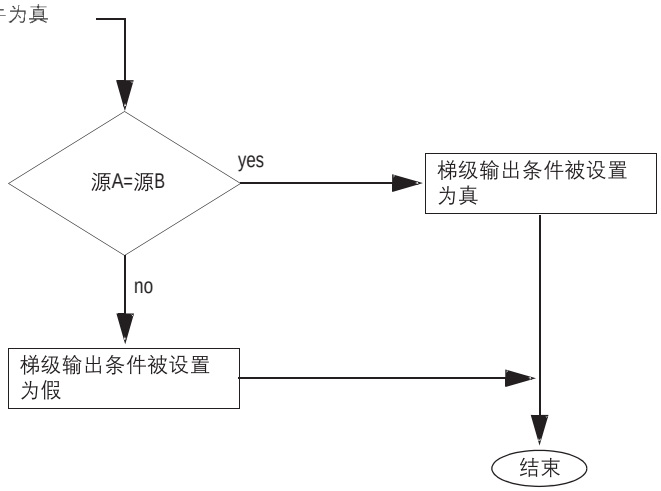
ASCII字符	十六进制代码
1ab	$31$61$62
1b	$31$62
A	$41
AB	$41$42
B	$42
a	$61
ab	$61$62

算术状态标志：不影响

故障条件：无

执行:

梯形图

条件:	动作:
预扫描	梯级输出条件被设置为假。
梯级输入条件为假	梯级输出条件被设置为假。
梯级输入条件为真	
后期扫描	梯级输出条件被设置为假。

功能块

条件:	动作:
预扫描	不动作。
首次扫描指令	不动作。
首次执行指令	不动作。
EnableIn被清零	清零EnableOut。
EnableIn被置位	执行指令。 置位EnableOut。
后期扫描	不动作。

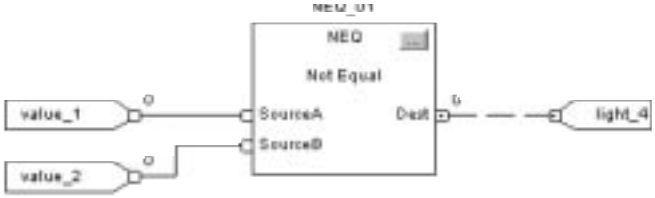
举例：如果value_1不等于value_2 ，则置位light_4。如果value_1等于value_2 ，则清零light_4。

梯形图

结构文本

```
light_4 := (value_1 <> value_2);
```

功能块



注释：

计算/算术指令 (CPT， ADD，SUB，MUL，DIV，MOD，SQR，SQRT，NEG，ABS)

简介

计算指令用表达式或专用计算指令来求算术运算的值。

如果用户需要：	所用指令：	可用语言：	参见页码：
计算表达式的值	CPT	梯形图 结构文本(1)	5-2
使两个数相加	ADD	梯形图 结构文本(2) 功能块	5-6
使两个数相减	SUB	梯形图 结构文本(2) 功能块	5-9
使两个数相乘	MUL	梯形图 结构文本(2) 功能块	5-12
使两个数相除	DIV	梯形图 结构文本(2) 功能块	5-15
在一个数被另一个数除后取余数	MOD	梯形图 结构文本(2) 功能块	5-19
计算一个数的平方根	SQR SQRT(3)	梯形图 结构文本 功能块	5-23
对一个数取负	NEG	梯形图 结构文本(2) 功能块	5-26
对一个数取绝对值	ABS	梯形图 结构文本 功能块	5-29

(1)不存在等价结构文本指令。使用其它结构文本编程来完成相同功能。参见该指令说明。
(2)不存在等价结构文本指令。使用表达式中的运算符。
(3)仅结构文本。

用户在计算时可混用不同数据类型，但会损失精度也可能会发生取整误差，并且会延长指令执行时间。检测S:V位以确定结果是否被截断。

如果用梯形图语言编程，则黑体字数据类型是最优数据类型。如果指令的全部操作数都用同一优化数据类型，则指令的执行速度快且占用较少内存，典型最优数据类型是DINT或REAL数据类型。

综合计算 (CPT)

CPT 指令执行表达式中定义的算术运算。

操作数：

梯形图

操作数：	数据类型：	格式：	说明：
Destination 目的	SINT INT DINT REAL	标签	目的—存储运算结果的标签
Expression 表达式	SINT INT DINT REAL	立即数 标签	表达式—由运算符分开的由标签和(或)立即数组成的表达式
	SINT 或INT 数据类型标签用符号-填充法转换成DINT 数据类型的数值。		

结构文本

结构文本中不存在CPT指令，但用户可使用赋值和表达式完成相同功能。

destination := numeric_expresion;

关于结构文本中赋值和表达式的语法信息，参见附录C。

说明： CPT 指令执行表达式中定义的算术运算。当指令被使能时，CPT指令计算表达式的值并将结果存放在目的标签中。

与其它计算指令相比，执行CPT指令速度稍慢，而且占用更多内存。CPT指令的优点是允许用户在一条指令中输入较复杂的表达式。

提　示　表达式的长度没有限制 。

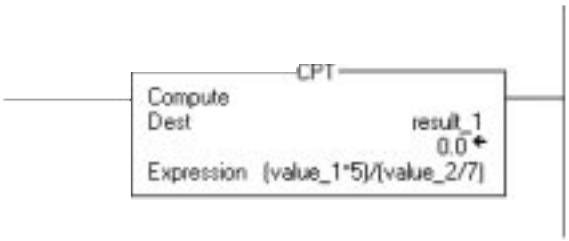
算术状态标志： 该指令影响算术状态标志

故障条件： 无

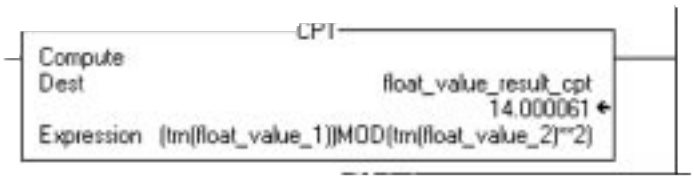
执行：

条件：	梯形图动作：
预扫描	梯级输出条件被设置为假。
梯级输入条件为假	梯级输出条件被设置为假。
梯级输入条件为真	指令计算表达式的值，并将结果存入目的标签中。 梯级输出条件被设置为真。
后期扫描	梯级输出条件被设置为假。

例1：当指令被使能时，CPT指令计算value_1乘以5的值，计算结果被value_2除以7的结果除，结果存放在result_1标签中。



例2：当指令被使能时，CPT指令截断float_value_1和float_value_2，再用float_value_2的平方的结果去除float_value_1，并将余数存储在float_value_result_cpt标签中。



有效运算符

运算符：	说明：	最优数据类型：
+	加	DINT，REAL
-	减/非	DINT，REAL
*	乘	DINT，REAL
/	除	DINT，REAL
**	指数(x的y次幂)	DINT，REAL
ABS	绝对值	DINT，REAL
ACS	反余弦	REAL
AND	按位与	DINT
ASN	反正弦	REAL
ATN	反正切	REAL
COS	余弦	REAL
DEG	弧度转换成度	DINT，REAL
FRD	BCD码转换成整数	DINT
LN	自然对数	REAL

运算符：	说明：	最优数据类型：
LOG	以10为底的对数	REAL
MOD	模数-除	DINT，REAL
NOT	按位取补码	DINT
OR	按位或	DINT
RAD	度转换成弧度	DINT，REAL
SIN	正弦	REAL
SQR	平方根	DINT，REAL
TAN	正切	REAL
TOD	整数转换成BCD码	DINT
TRN	截断	DINT，REAL
XOR	按位异或	DINT

格式化表达式

对于表达式中使用的每个运算符，用户必须提供一个或两个操作数(标签或立即数)。表达式中运算符和操作数的格式必须符合下表的规定：

运算符要求：	所用格式：	举例：
一个操作数	运算符(操作数)	ABS(tag_a)
两个操作数	操作数_a 运算符 操作数_b	• tag_b + 5 • tag_c AND tag_d • (tag_e ** 2) MOD (tag_f / tag_g)

确定运算顺序

指令按规定的顺序而非写入表达式的顺序，来执行写入表达式的运算。可以通过用圆括号分组的方法来重新安排运算顺序，以强制指令在执行其它运算之前先执行括号内的运算。

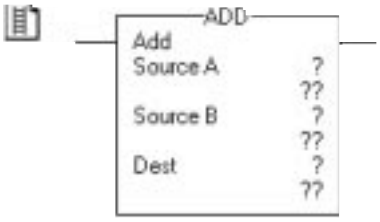
同级运算按从左到右的顺序执行。

顺序：	运算符：
1	( )
2	ABS，ACS，ASN，ATN，COS，DEG，FRD，LN，LOG，RAD，SIN，SQR，TAN，TOD，TRN
3	**
4	- (取负)，NOT
5.	*，/，MOD
6.	- (减)，+
7.	AND
8.	XOR
9.	OR

加法 (ADD)

ADD指令计算源A与源B的和，并将结果存入目的标签中。

操作数：



梯形图

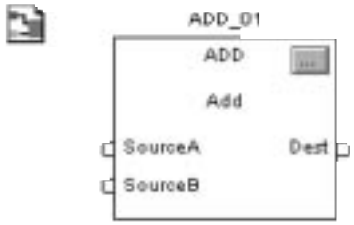
操作数：	数据类型：	格式：	说明：
Source A 源A	SINT INT DINT REAL SINT 或 INT数据类型标签用符号-扩充法转换成DINT数据类型的数值。	立即数 标签	要与源B的值相加的数值
Source B 源B	SINT INT DINT REAL SINT 或 INT数据类型标签用符号-扩充法转换成DINT数据类型的数值。	立即数 标签	要与源A的值相加的数值
Destination 目的	SINT INT DINT REAL	标签	存储计算结果的标签

结构文本

```
dest := sourceA + sourceb;
```

使用加号 ″+″ 作为表达式内的运算符。该表达式计算sourceA与sourceB的和，并将结果保存在dest中。

关于结构文本中表达式的语法信息，参见附录C。



功能块

操作数：	数据类型：	格式：	说明：
ADD标签	FBD_MATH	结构体	ADD结构体

FBD_MATH结构体

输入参数:	数据类型:	说明:
EnableIn	BOOL	使能输入。如果清零，则不执行指令也不更新输出。 缺省状态是置位。
SourceA 源A	REAL	要与源B 中的数值相加的数值。 有效值=任一浮点数
SourceB 源B	REAL	要与源A 中的数值相加的数值。 有效值=任一浮点数
输出参数:	**数据类型:**	**说明:**
EnableOut	BOOL	指令产生一有效结果。
Dest	REAL	算术运算的结果。该输出影响算术状态标志。

说明： ADD指令计算源A与源B的和，并将结果存入目的标签。

算术状态标志： 影响算术状态标志。

故障条件: 无

执行：

梯形图

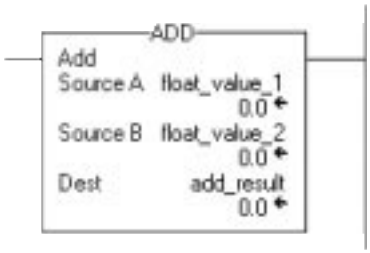
条件:	动作:
预扫描	梯级输出条件被设置为假。
梯级输入条件为假	梯级输出条件被设置为假。
梯级输入条件为真	Destination = Source A + Source B 梯级输出条件被设置为真。
后期扫描	梯级输出条件被设置为假。

功能块

条件:	动作:
预扫描	不动作。
首次扫描指令	不动作。
首次执行指令	不动作。
EnableIn 是清零状态	清零EnableOut。
EnableIn 是置位状态	执行指令。 置位EnableOut。
后期扫描	不动作。

举例：ADD指令计算float_value_1 与float_value_2 的和，并将结果存入add_result标签中。

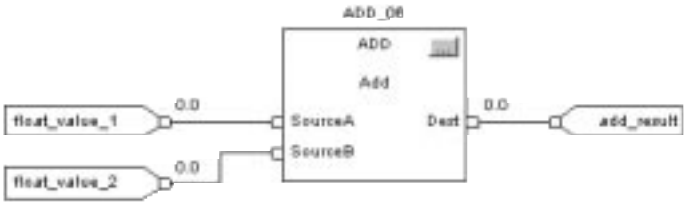
梯形图



结构文本

```
add_result := float_value_1 + float_value_2;
```

功能块

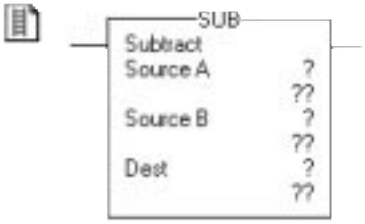


减法 (SUB)

SUB指令将源B的数值从源A的数值中减去，并将结果存入目的标签。

操作数：

梯形图



操作数：	数据类型：	格式：	说明：
Source A 源A	SINT INT DINT REAL SINT 或 INT数据类型标签用符号-扩充法转换成DINT 数据类型数值。.	立即数 标签	从该值中减去源B的值。
Source B 源B	SINT INT DINT REAL SINT 或 INT数据类型标签用符号-扩充法转换成DINT 数据类型数值。.	立即数 标签	从源A中减去该值
Destination 目的	SINT INT DINT REAL	标签	存储计算结果的标签

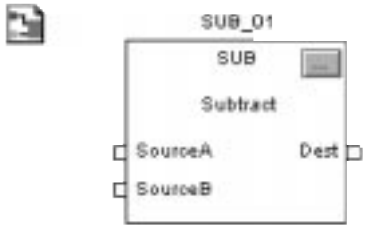
结构文本

```
dest := sourceA - sourceB;
```

使用减号作为表达式内的运算符。该表达式将sourceB从sourceA中减去，并将结果保存在dest中。

关于结构文本中表达式的语法信息，参见附录C。

功能块



操作数：	数据类型：	格式：	说明：
SUB标签	FBD_MATH	结构体	SUB结构体

FBD_MATH结构体

输入参数:	数据类型:	说明:
EnableIn	BOOL	使能输入。如果清零，则不执行指令也不更新输出。 缺省状态是置位。
SourceA 源A	REAL	从该值中减去源B的值。 有效值=任一浮点数
SourceB 源B	REAL	从源A中减去该值。 有效值=任一浮点数
输出参数:	**数据类型:**	**说明:**
EnableOut	BOOL	指令产生一有效结果。
Dest	REAL	算术运算的结果。该输出影响算术状态标志。

说明: SUB指令将源B从源A减去，并将结果存入目的标签中。

算术状态标志: 影响算术状态标志。

故障条件: 无

执行:

梯形图

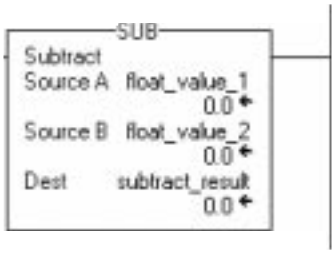
条件:	动作:
预扫描	梯级输出条件被设置为假。
梯级输入条件为假	梯级输出条件被设置为假。
梯级输入条件为真	Destination(目的标签) = Source A(源A) - Source B(源B) 梯级输出条件被设置为真。
后期扫描	梯级输出条件被设置为假。

功能块

条件:	动作:
预扫描	不动作。
首次扫描指令	不动作。
首次执行指令	不动作。
EnableIn是清零状态	清零EnableOut。
EnableIn是置位状态	执行指令。 置位EnableOut。
后期扫描	不动作。

举例：计算float_value_1 减去float_value_2 的差，并将结果存入subtract_result标签中。

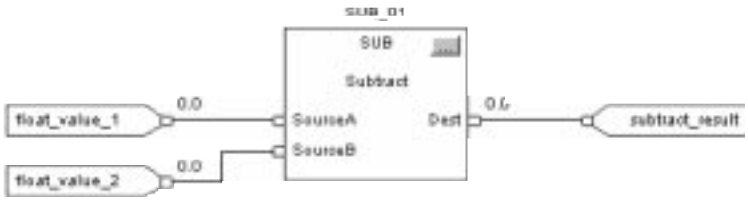
梯形图



结构文本

```
subtract_result := float_value_1 - float_value_2;
```

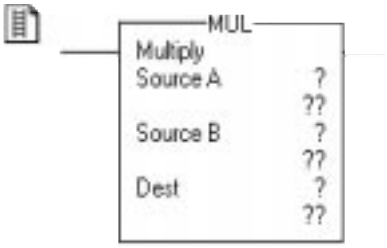
功能块



乘法 (MUL)

MUL指令计算源A与源B的乘积，并将计算结果存入目的标签。

操作数：



梯形图

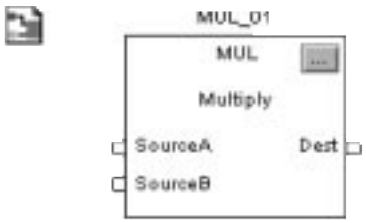
操作数：	数据类型：	格式：	说明：
Source A 源A	SINT INT **DINT** **REAL** SINT 或 INT 数据类型标签用符号-扩充法转换成DINT数据类型数值。	立即数 标签	被乘数
Source B 源B	SINT INT **DINT** **REAL** SINT 或 INT 数据类型标签用符号-扩充法转换成DINT数据类型数值。	立即数 标签	乘数
Destination 目的	SINT INT **DINT** **REAL**	标签	存储计算结果的标签

结构文本

```
dest := sourceA * sourceB;
```

使用乘号 "*" 作为表达式内的运算符。该表达式计算sourceA与sourceB的乘积，并将结果保存在dest中。

关于结构文本中表达式的语法信息，参见附录C。



功能块

操作数：	数据类型：	格式：	说明：
MUL标签	FBD_MATH	结构体	MUL结构体

FBD_MATH结构体

输入参数:	数据类型:	说明:
EnableIn	BOOL	使能输入。如果清零，则不执行指令也不更新输出。 缺省状态是置位。
SourceA 源A	REAL	被乘数。 有效值=任一浮点数
SourceB 源B	REAL	乘数。 有效值=任一浮点数
输出参数:	**数据类型:**	**说明:**
EnableOut	BOOL	指令产生一有效结果。
Dest	REAL	算术运算的结果。该输出影响算术状态标志。

说明：MUL指令计算源A与源B的乘积，并将计算结果存入目的标签。

算术状态标志：影响算术状态标志。

故障条件: 无

执行：

梯形图

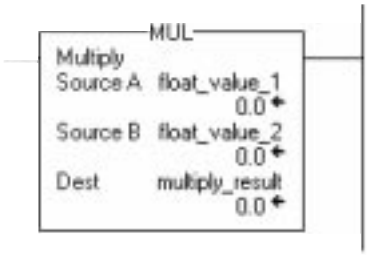
条件:	动作:
预扫描	梯级输出条件被设置为假。
梯级输入条件为假	梯级输出条件被设置为假。
梯级输入条件为真	Destination(目的标签) = Source B(源B) × Source A(源A) 梯级输出条件被设置为真。
后期扫描	梯级输出条件被设置为假。

功能块

条件:	动作:
预扫描	不动作。
首次扫描指令	不动作。
首次执行指令	不动作。
EnableIn是清零状态	清零EnableOut。
EnableIn是置位状态	执行指令。 置位EnableOut。
后期扫描	不动作。

举例：计算float_value_1 与float_value_2 的乘积，并将结果存入multiply_result标签中。

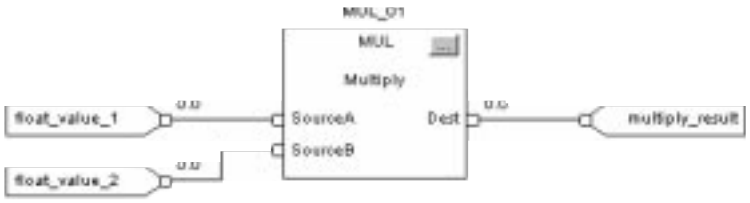
梯形图



结构文本

```
multiply_result := float_value_1 *  float_value_2;
```

功能块

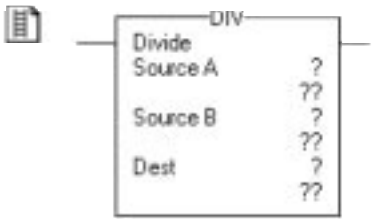


除法 (DIV)

DIV指令计算源A除以源B除的商，并将计算结果存入目的标签。

操作数：

梯形图



操作数：	数据类型：	格式：	说明：
Source A 源A	SINT INT **DINT** **REAL** SINT或INT数据类型标签用符号-扩充法转换成DINT数据类型数值。	立即数 标签	被除数
Source B 源B	SINT INT **DINT** **REAL** SINT 或 INT数据类型标签用符号填充法转换成DINT数据类型数值。	立即数 标签	除数
Destination 目的	SINT INT **DINT** **REAL**	标签	存储计算结果的标签

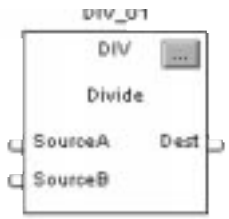
结构文本

```
dest := sourceA / sourceB;
```

使用除号 ″/″ 作为表达式内的运算符。该表达式计算sourceA除以sourceB的商，并将结果保存在dest中。

关于结构文本中表达式的语法信息，参见附录C。

功能块



操作数：	数据类型：	格式：	说明：
DIV标签	FBD_MATH	结构体	DIV结构体

FBD_MATH结构体

输入参数：	数据类型：	说明：
EnableIn	BOOL	使能输入。如果清零，则指令不执行且输出不更新。 缺省状态是置位。
SourceA 源A	REAL	被除数。 有效值=任一浮点数
SourceB 源B	REAL	除数。 有效值=任一浮点数
输出参数：	**数据类型：**	**说明：**
EnableOut	BOOL	指令产生一有效结果。
Dest	REAL	算术运算的结果。该输出影响算术状态标志。

说明： 如果目的标签不是REAL数据类型，则DIV指令按下列规则处理运算结果的小数部分：

如果源A：	结果的小数部分：	举例：		
和源B的数据类型不是REAL	截断	Source A	DINT	5
		Source B	DINT	3
		Destination	DINT	1
或源B的数据类型是REAL	取整	Source A	REAL	5
		Source B	DINT	3
		Destination	DINT	2

如果源B (除数)是0，则：

- 发生次要故障：
  —故障类型4：编程错误。
  —故障代码4：算术溢出。
- 按下列方式设置目的标签：

如果Source B是0且	目的标签是：	结果是：	目的标签被设置为：
所有操作数都是整数(SINT， INT，或 DINT)	→	→	Source A
至少有一个操作数是REAL	SINT， INT或 DINT	正数	-1
		负数	0
	REAL	正数	1.$ (正无穷大)
		负数	-1.$ (负无穷大)

可以通过检测次要故障位(S:MINOR)，来查看指令的除数是不是0值。参见Logix5000 Controllers Common Procedures一书，出版号1756-PM001。

算术状态标志：影响算术状态标志。

故障条件:

发生次要故障如果：	故障类型：	故障代码：
除数是0	4	4

执行：

梯形图

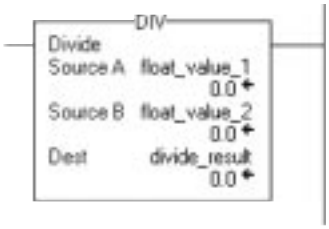
条件：	动作：
预扫描	梯级输出条件被设置为假。
梯级输入条件为假	梯级输出条件被设置为假。
梯级输入条件为真	Destination = Source A/ Source B 梯级输出条件被设置为真。
后期扫描	梯级输出条件被设置为假。

功能块

条件：	动作：
预扫描	不动作。
首次扫描指令	不动作。
首次执行指令	不动作。
EnableIn是清零状态	清零EnableOut。
EnableIn是置位状态	执行指令。 置位EnableOut。
后期扫描	不动作。

例1：计算float_value_1 被float_value_2 除的商，并将结果存入 divide_result标签中。

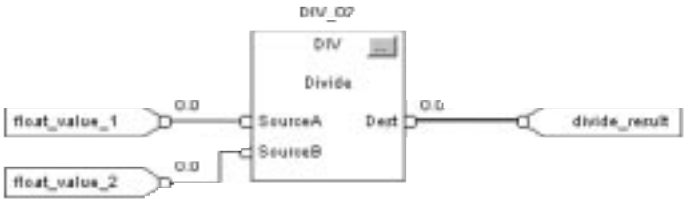
梯形图



结构文本

```
divide_result := float_value_1 / float_value_2;
```

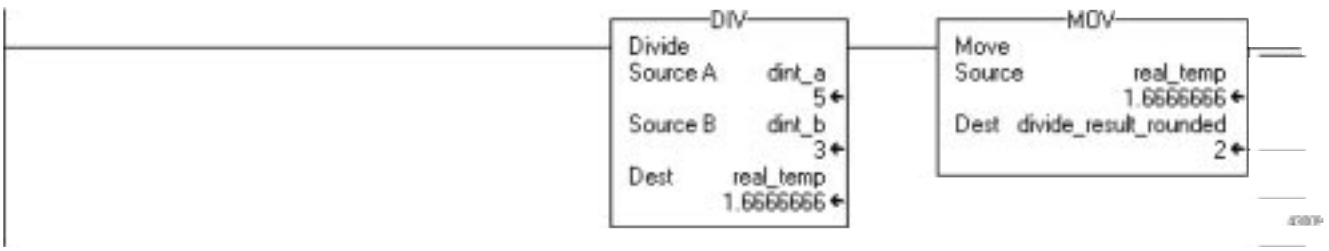
功能块



例2： 本例一起使用DIV和MOV两条指令去除两个整数，并将取整的结果存入整数标签中：

- DIV指令用dint_b去除dint_a。
- 对结果取整，目的标签是REAL类型。(如果目的标签是整型标签(SINT， INT，或DINT)，则指令会将结果截断。)
- MOV指令将DIV指令的取整结果(real_temp)传送到divide_result_rounded标签中。
- 因为 divide_result_rounded 是一个DINT型标签，来自real_temp 的数值被取整并存放到DINT类型的目的标签中。

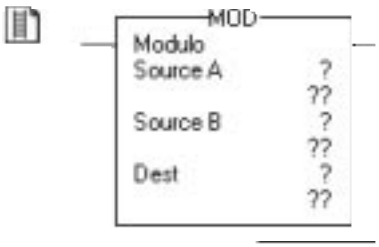
梯形图



求模(MOD)

MOD指令计算源A被源B除的余数，并将计算结果的余数存入目的标签。

操作数：



梯形图

操作数：	数据类型：	格式：	说明：
Source A 源A	SINT INT DINT REAL	立即数 标签	被除数
	SINT 或 INT数据类型标签用符号-扩充法转换成DINT 数据类型数值。		
Source B 源B	SINT INT DINT REAL	立即数 标签	除数
	SINT 或 INT数据类型标签用符号-扩充法转换成DINT 数据类型数值。		
Destination 目的地址	SINT INT DINT REAL	标签	存储计算结果的标签

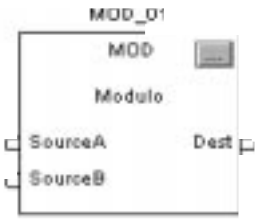
结构文本

```
dest := sourceA MOD sourceB;
```

使用MOD作为表达式内的运算符。该表达式用sourceB除sourceA，并将余数保存在dest中。

关于结构文本中表达式的语法信息，参见附录C。

功能块



操作数：	数据类型：	格式：	说明：
MOD标签	FBD_MATH	结构体	MOD 结构体

FBD_MATH结构体

输入参数：	数据类型：	说明：
EnableIn	BOOL	使能输入。如果清零，则不执行指令也不更新输出。 缺省状态是置位。
SourceA 源A	REAL	被除数。 有效值=任一浮点数
SourceB 源B	REAL	除数。 有效值=任一浮点数
输出参数：	**数据类型：**	**说明：**
EnableOut	BOOL	指令产生一有效结果。
Dest	REAL	算术运算的结果。该输出影响算术状态标志。

说明： 如果源B(除数)是0，则：

- 发生次要故障：
  —故障类型4：编程错误
  —故障代码4：算术溢出
- 按下列方式设置目的标签：

如果Source B是0且	目的标签是：	结果是：	目的标签被设置为：
全部操作数都是整数(SINT， INT，或 DINT)	→	→	Source A
至少有一个操作数是REAL	SINT， INT或 DINT	正数	-1
		负数	0
	REAL	正数	1.$ (正无穷大)
		负数	-1.$ (负无穷大)

可以通过检测次要故障位(S:MINOR)，来查看除数是否为0值。参见Logix5000 Controllers Common Procedures一书，出版号1756-PM001。

算术状态标志：算术状态标志： 影响算术状态标志位。

故障条件: 故障条件:

发生次要故障如果：	故障类型：	故障代码：
除数是0	4	4

执行：执行：

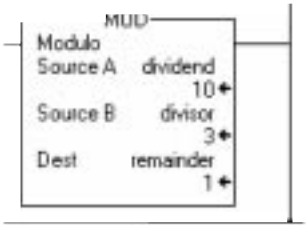
梯形图

条件：	动作：
预扫描	梯级输出条件被设置为假。
梯级输入条件为假	梯级输出条件被设置为假。
梯级输入条件为真	Destination = Source A ®C (TRN (Source A / Source B) * Source B)梯级输出条件被设置为真。
后期扫描	梯级输出条件被设置为假。

功能块

条件：	动作：
预扫描	不动作。
首次扫描指令	不动作。
首次执行指令	不动作。
EnableIn是清零状态	清零EnableOut。
后期扫描	不动作。

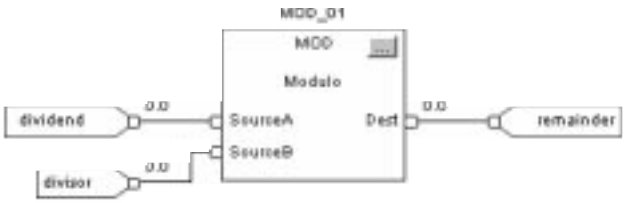
举例：执行dividend 被divisor除的运算，并将余数存入remainder中。在此例中，10除以3余数是1。



结构文本

```
remainder := dividend MOD divisor;
```

功能块

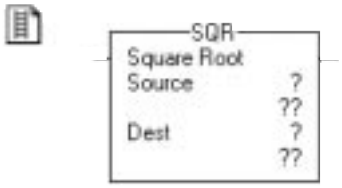


平方根 (SQR)

SQR指令计算源值的平方根，并将计算结果存入目的标签。

操作数：

梯形图



操作数：	数据类型：	格式：	说明：
Source 源	SINT INT **DINT** **REAL** SINT 或 INT数据类型标签用符号-扩充法转换成DINT 数据类型数值。	立即数 标签	计算该值的平方根
Destination 目的	SINT INT **DINT** **REAL**	标签	存储计算结果的标签

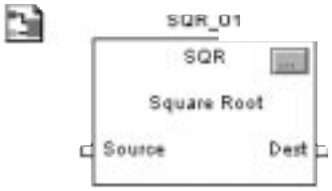
结构文本

```
dest := SQRT(source);
```

使用SQRT作为一个函数。该表达式计算源值的平方根，并将结果保存在dest中。

关于结构文本中表达式的语法信息，参见附录C。

功能块



操作数：	数据类型：	格式：	说明：
SQR 标签	FBD_MATH_ADVANCED	结构体	SQR结构体

FBD_MATH_ADVANCED结构体

输入参数：	数据类型：	说明：
EnableIn	BOOL	使能输入。如果清零，则不执行指令也不更新输出。 缺省状态是置位。
Source 源	REAL	计算该值的平方根。 有效值=任一浮点数
输出参数：	**数据类型：**	**说明：**
EnableOut	BOOL	指令产生一有效结果。
Dest	REAL	算术运算的结果。该输出影响算术状态标志。

说明： 如果目的标签不是REAL数据类型，则DIV指令按下列规则处理运算结果的小数部分：

如果Source 是：	结果的小数部分：	举例：		
非REAL	截断	Source	DINT	3
		Destination	DINT	1
REAL	取整	Source	REAL	3
		Destination	DINT	2

如果源值是负数，则指令在计算平方根前先求其绝对值。

算术状态标志： 影响算术状态标志。

故障条件： 无

执行：

梯形图

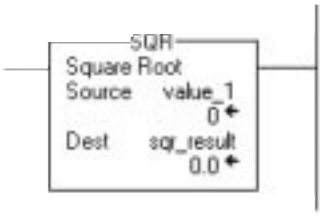
条件：	动作：
预扫描	梯级输出条件被设置为假。
梯级输入条件为假	梯级输出条件被设置为假。
梯级输入条件为真	梯级输出条件被设置为真。
后期扫描	梯级输出条件被设置为假。

功能块

条件：	动作：
预扫描	不动作。
首次扫描指令	不动作。
首次执行指令	不动作。
EnableIn 是清零状态	清零EnableOut。
EnableIn 是置位状态	执行指令。 置位EnableOut。
后期扫描	不动作。

举例：计算value_1的平方根，并将计算结果存入sqr_result中。

梯形图



结构文本

```
sqr_result := SQRT(value_1);
```

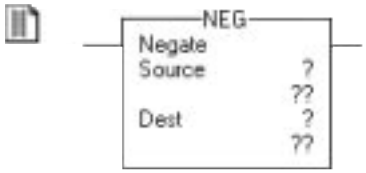
功能块

取负 (NEG)

NEG指令改变源值的符号，并将结果存入目的标签。

操作数：

梯形图



操作数：	数据类型：	格式：	说明：
Source 源	SINT INT **DINT** **REAL** SINT 或 INT数据类型标签用符号-扩充法转换成DINT数据类型数值。	立即数 标签	需要取负的值
Destination 目的	SINT INT **DINT** **REAL**	标签	存储计算结果的标签

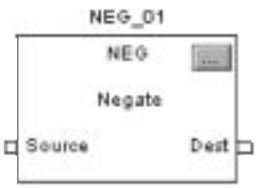
结构文本

```
dest := -source;
```

使用负号“–”作为表达式内的运算符。该表达式改变源值的符号，并将结果保存在dest中。

关于结构文本中表达式的语法信息，参见附录C。

功能块



操作数：	数据类型：	格式：	说明：
NEG标签	FBD_MATH_ADVANCED	结构体	NEG结构体

FBD_MATH结构体

输入参数：	数据类型：	说明：
EnableIn	BOOL	使能输入。如果清零，则指令不执行且输出不更新。 缺省状态是置位。
Source	REAL	需要取负的值。 有效值=任一浮点数
输出参数：	数据类型：	说明：
EnableOut	BOOL	指令产生一有效结果。
Dest	REAL	算术运算的结果。该输出影响算术状态标志。

说明：说明：如果对负数取负，结果为正数。如果对正数取负，结果为负数。

算术状态标志：算术状态标志： 影响算术状态标志位。

故障条件:

执行：

梯形图

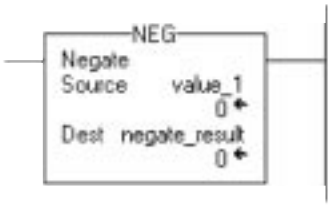
条件：	动作：
预扫描	梯级输出条件被设置为假。
梯级输入条件为假	梯级输出条件被设置为假。
梯级输入条件为真	Destination = 0 - Source 梯级输出条件被设置为真。
后期扫描	梯级输出条件被设置为假。

功能块

条件：	动作：
预扫描	不动作。
首次扫描指令	不动作。
首次执行指令	不动作。
EnableIn 是清零状态	清零EnableOut 。
EnableIn 是置位状态	执行指令。 置位EnableOut 。
后期扫描	不动作。

举例：改变value_1的符号，并将计算结果存入negate_result中。

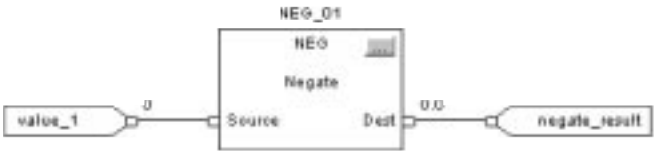
梯形图



结构文本

```
negate_result := -value_1;
```

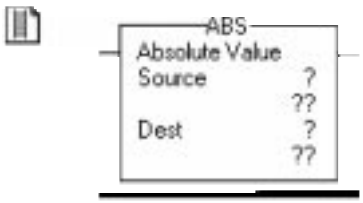
功能块



绝对值 (ABS)

ABS指令计算源值的绝对值，并将计算结果存入目的标签。

操作数：



梯形图

操作数：	数据类型：	格式：	说明：
Source 源	SINT INT DINT REAL	立即数 标签	需要取绝对值的值
	SINT 或 INT数据类型标签用符号-扩充法转换成DINT 数据类型数值。		
Destination 目的	SINT INT DINT REAL	标签	存储计算结果的标签

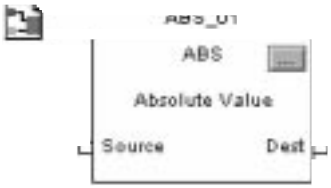
结构文本

```
dest := ABS(source);
```

ABS指令作为一个函数。该表达式计算source的绝对值，并将结果保存在dest中。

关于结构文本中表达式的语法信息，参见附录C。

功能块



操作数：	数据类型：	格式：	说明：
ABS标签	FBD_MATH_ADVANCED	结构体	ABS结构体

FBD_MATH_ADVANCED结构体

输入参数：	数据类型：	说明：
EnableIn	BOOL	使能输入。如果清零，则不执行指令也不更新输出。 缺省状态是置位。
Source 源	REAL	需要取绝对值的值。 有效值=任一浮点数
输出参数：	**数据类型：**	**说明：**
EnableOut	BOOL	指令产生一有效结果。
Dest	REAL	算术运算的结果。该输出影响算术状态标志。

说明： ABS指令计算源值的绝对值，并将计算结果存入目的标签。

算术状态标志： 影响算术状态标志。

故障条件: 无

执行：

梯形图

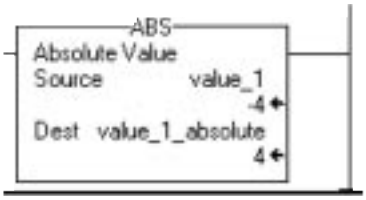
条件：	动作：
预扫描	梯级输出条件被设置为假。
梯级输入条件为假	梯级输出条件被设置为假。
梯级输入条件为真	梯级输出条件被设置为真。
后期扫描	梯级输出条件被设置为假。

功能块

条件：	动作：
预扫描	不动作。
首次扫描指令	不动作。
首次执行指令	不动作。
EnableIn 是清零状态	清零EnableOut。
EnableIn 是置位状态	执行指令。 置位EnableOut。
后期扫描	不动作。

实例：将value_1的绝对值存入value_1_absolute中。在本例中- 4的绝对值是4。

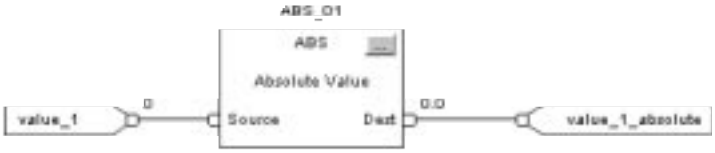
梯形图



结构文本

```
value_1_absolute := ABS(value_1);
```

功能块



注释：

传送/逻辑指令 (MOV，MVM，BTD，MVMT，BTDT，CLR，SWPB，AND，OR，XOR，NOT，BAND，BOR，BXOR，BNOT)

简介

用户可以混用数据类型，但是会损失精度，也可能发生取整误差，并且会占用更多的时间执行指令。检测S:V位以确认结果是否被截断。

对于梯形图指令，黑体字数据类型表示最优数据类型。如果一条指令的全部操作数都使用同一种最优数据类型，则指令执行的速度更快而且占用内存较少。典型的最优数据类型是DINT或REAL。

传送指令用于修改和传送数据位。

如果用户需要：	所用指令：	可用语言：	参见页码：
复制数值	MOV	梯形图 结构文本(1)	6-3
复制整数的特定部分	MVM	梯形图	6-5
复制功能块内整数的特定部分	MVMT	结构文本 功能块	6-8
在整数内或整数间传送数据位	BTD	梯形图	6-11
在功能块内的整数内或整数间传送数据位	BTDT	结构文本 功能块	6-14
清零数值	CLR	结构文本(1) 梯形图	6-17
调整INT，DINT，或REAL标签的字节顺序	SWPB	梯形图 结构文本	6-19

(1)不存在等价的结构文本指令。使用其它结构文本编程完成相同功能。参见指令说明。

逻辑指令执行位的逻辑运算。

如果用户需要：	所用指令：	可用语言：	参见页码：
按位逻辑与运算	Bitwise AND &(1)	梯形图 结构文本(2) 功能块	6-23
按位逻辑或运算	Bitwise OR	梯形图 结构文本(2) 功能块	6-26
按位逻辑异或运算	Bitwise XOR	梯形图 结构文本(2) 功能块	6-29
按位逻辑非运算	Bitwise NOT	梯形图 结构文本(2) 功能块	6 - 32
最多与8个布尔型输入量做逻辑与操作。	Boolean AND(BAND)	结构文本(2) 功能块	6 - 35
最多与8个布尔型输入量做逻辑或操作。	Boolean OR(BOR)	结构文本(2) 功能块	6 - 38
对两个布尔型输入量做逻辑异或操作。	Boolean Exclusive OR (BXOR)	结构文本(2) 功能块	6 - 41
对布尔型输入量取反。	Boolean NOT(BNOT)	结构文本(2) 功能块	6 - 44

(1) 仅结构文本。

(2) 结构文本中，AND、OR、XOR及NOT可做按位运算或逻辑运算。

传送(MOV)

MOV指令复制源值数值到目的标签。源值保持不变。

操作数：

梯形图

操作数：	数据类型：	格式：	说明：
Source 源	SINT INT **DINT** **REAL**	立即数 标签	被传送(复制)的数值
	SINT或INT数据类型标签通过符号-扩充转换为DINT数据类型数值。		
Destination 目的标签	SINT INT **DINT** **REAL**	标签	存储传送结果

结构文本

使用带有赋值符号“：＝”的表达式。该赋值符号将source传送到dest中。

关于结构文本中表达式和赋值符号的语法信息，参见结构文本编程。

说明：MOV指令复制源值到目的标签。源值保持不变。

算术状态标志：影响算术状态标志。

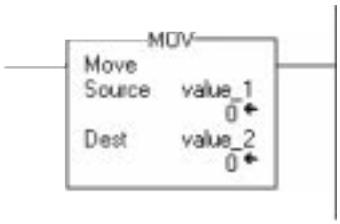
故障条件：无

执行：

条件：	动作：
预扫描	梯级输出条件被设置为假。
梯级输入条件为假	梯级输出条件被设置为假。
梯级输入条件为真	指令复制源值到目的标签。 梯级输出条件被设置为真。
后期扫描	梯级输出条件被设置为假。

举例：将value_1中的数据传送到value_2。

梯形图



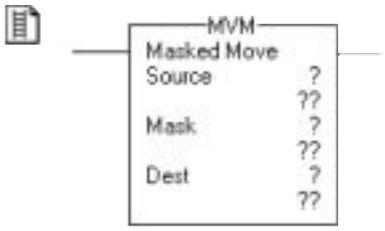
结构文本

value_2 := value _1;

屏蔽传送(MVM)

MVM指令复制源值数据到目的标签，并且允许部分数据被屏蔽。
该指令相当于结构文本和功能块编程中MVMT指令，参见6 - 8页。

操作数：



梯形图

操作数：	数据类型：	格式：	说明：
Source 源值	SINT INT **DINT** SINT或INT数据类型标签通过零-填充转换为DINT数据类型数值。	立即数 标签	被传送的数值
Mask 屏蔽值	SINT INT **DINT** SINT或INT数据类型标签通过零-填充转换为DINT数据类型数值。	立即数 标签	阻滞或通过的位
Destination 目的标签	SINT INT **DINT**	标签	存储传送结果

结构文本

```
dest := (Dest AND NOT (Mask))
     OR (Source AND Mask);
```

该指令相当于结构文本编程中的MVMT指令。或者用户可将按位逻辑和表达式相结合，并将结果赋值给目的标签。该表达式实现源值的屏蔽传送。

关于结构文本的表达式和赋值语句的语法信息，请参见结构文本编程。

说明：MVM指令通过屏蔽传送或阻滞源数据位。屏蔽位的1值意味着位数据可通过。屏蔽位的0值意味着位数据被阻滞。

如果用混合整型数据类型，则指令用0值来填充小整数数据类型的高位，以使它们与最大整型数据类型的大小相同。

输入立即数作为屏蔽值

当输入立即数作为屏蔽值时，编程软件默认为是十进制数值。如果要输入一个其它格式的屏蔽值，可以在数值之前加相应的前缀，如下表所示：

前缀：	说明：
16#	十六进制 例如：16#0F0F
8#	八进制 例如：8#16
2#	二进制 例如：2#00110011

算术状态标志：影响算术状态标志。

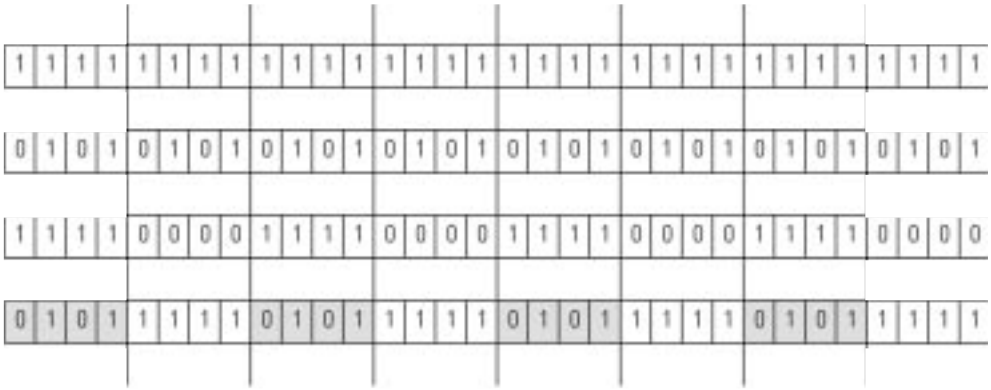
故障条件：无

执行：

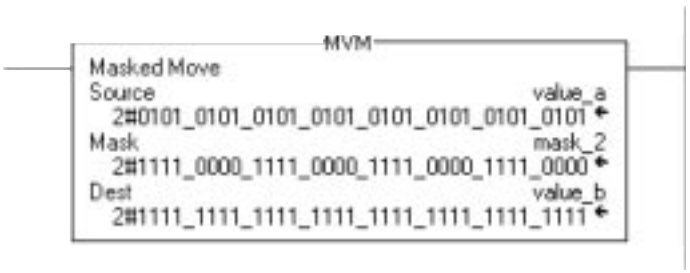
条件：	动作：
预扫描	梯级输出条件被设置为假。
梯级输入条件为假	梯级输出条件被设置为假。
梯级输入条件为真	指令使源值值通过屏蔽并复制结果到目的标签。目的标签的未屏蔽位保持不变。 梯级输出条件被设置为真。
后期扫描	梯级输出条件被设置为假。

举例：MVM指令从value_a复制数据到value_b，同时允许数据被屏蔽(屏蔽操作数内各位的0值屏蔽value_a内的数据)。

The shaded boxes show the bits that changed in value_b.
阴影部分表示value_b内被改变的位。



梯形图



结构文本

```
value_b := (value_b AND NOT (mask_2)) OR
(value_a AND mask_2);
```

带目标的屏蔽传送 (MVMT)

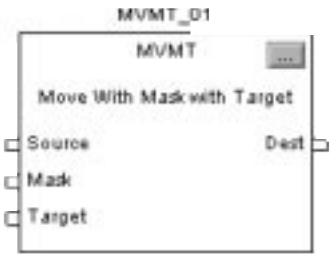
MVMT指令首先将目标复制到目的标签。然后比较被屏蔽的源值和目的标签操作数，改变目的标签内数值。目标值和源值保持不变。
该指令相当于梯形图编程中MVM指令，参见6 - 5页。

操作数：

```
MVMT(MVMT_tag),
```

结构文本

变量：	数据类型：	格式：	说明：
MVMT标签	FBD_MASKED_MOVE	结构体	MVMT结构体



功能块

操作数：	数据类型：	格式：	说明：
MVMT标签	FBD_MASKED_MOVE	结构体	MVMT结构体

FBD_MASKED_MOVE结构体

输入参数：	数据类型：	说明：
EnableIn	BOOL	功能块： 如果被清零，该指令不执行也输出不更新。 如果被置位，指令执行。 缺省状态为置位。 结构文本： 无影响。执行指令。
Source 源值	DINT	经屏蔽的传送到目的标签的输入值。 有效值 = 任一整数
Mask 屏蔽值	DINT	屏蔽由源传送到目的标签的位。全部位设置为1，则对应位由源传送到目的标签。全部位设置为0，则对应的位不能由源传送到目的标签。 有效值 = 任一整数
Target 目标值	DINT	在通过屏蔽传送源值传送位之前传送到目的标签的输入值。 有效值 = 任一整型数
输出参数：	**数据类型：**	**说明：**
EnableOut	BOOL	指令产生一有效结果。
Dest	DINT	屏蔽传送指令的结果。该输出影响算术状态标志。

说明：当指令被使能时，MVMT指令通过屏蔽传送或阻滞源数据位。屏蔽位的1值意味着位数据可以通过。屏蔽位的0值意味着位数据被阻滞。

如果用户使用混合整型，则指令用0值来填充小整数数据类型的高位，以使它们与最大整型的大小相同。

使用输入参考值输入立即数作为屏蔽值

当输入立即数作为屏蔽值时，编程软件默认为是十进制数值。如果要输入一个其它格式的屏蔽值，可以在数值之前加相应的前缀，如下表所示：

前缀：	说明：
16#	十六进制 例如：16#0F0F
8#	八进制 例如：8#16
2#	二进制 例如：2#00110011

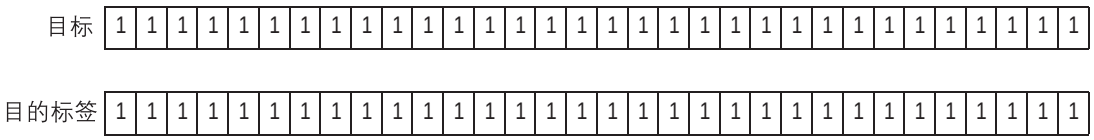
算术状态标志：影响算术状态标志。

故障条件：无

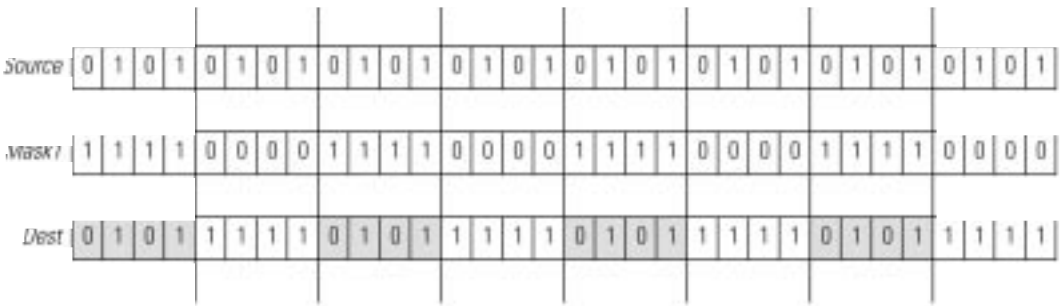
执行：

条件：	功能块动作：	结构文本动作：
预扫描	不动作。	不动作。
指令首次扫描	不动作。	不动作。
指令首次运行	不动作。	不动作。
EnableIn被清零	EnableOut被清零，指令不执行也输出不更新。	无影响。
EnableIn被置位	执行指令。 EnableOut被置位。	EnableIn一直被置位。 执行指令。
后期扫描	不动作。	不动作。

举例：1 控制器将目标数值复制到目的标签。



2 屏蔽源值，并与目的标签内的数值比较。使目的标签内的数值作相应改变。源和目标保持不变。屏蔽值内的0禁止指令比较该位(本例中示为x)。

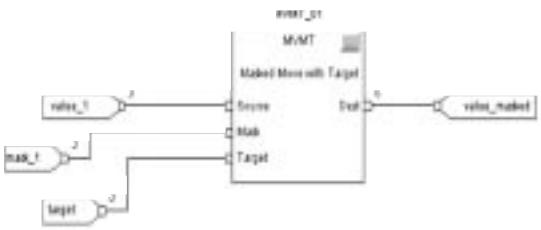


阴影部分表示被改变的位。

结构文本

```
MVMT_01.Source := value_1;
MVMT_01.Mask := mask1;
MVMT_01.Target := target;

MVMT(MVMT_01);

value_masked := MVMT_01.Dest;
```

功能块



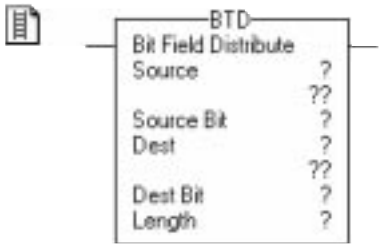
位域分配(BTD)

BTD指令复制源值中指定位，移动这些位到适当位置，并将其写入目的标签。

该指令相当于结构文本和功能块编程中的BTDT指令，参见6 - 14页。

操作数：

梯形图



操作数：	数据类型：	格式：	说明：
Source 源值	SINT INT **DINT** SINT或INT数据类型标签通过零-填充转换为DINT类型数值。	立即数 标签	包含要传送数据位的标签
Source bit 源位	DINT	立即数 (0-31 DINT) (0-15 INT) (0-7 SINT)	开始传送位的位置号(最低位号)必须在源数据类型的有效范围内
Destination	SINT INT **DINT**	标签	传送位的目的标签
Destination bit	DINT	立即数 (0-31 DINT) (0-15 INT) (0-7 SINT)	目的位—从源值复制的位在目的标签的起始位号(最低位号)必须在目的操作数数据类型的有效范围内
Length	DINT	立即数 (1-32)	长度—被传送位的数量

说明： 当指令被使能时，BTD指令从源值复制一组位到目的标签。这组位用源位(该组的最低位号)和长度(要复制位的数量)标识。目的位标识目的标签内开始的最低位号。源值保持不变。

如果位字段长度超出了目的标签长度，则指令不保存超出位。超出位不与下个字重叠。

如果使用混合整数数据类型，则指令用0值来填充小整数数据类型的高位，以使它们与最大整型的大小相同。

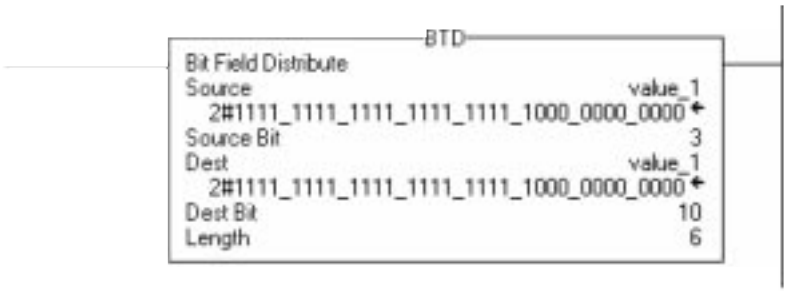
算术状态标志： 影响算术状态标志。

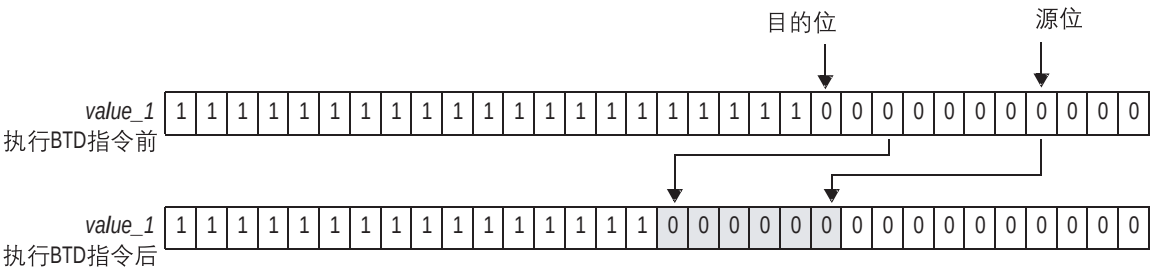
故障条件： 无

执行：

条件：	梯形图动作：
预扫描	梯级输出条件被设置为假。
梯级输入条件为假	梯级输出条件被设置为假。
梯级输入条件为真	指令复制并且传送源数据位到目的标签。 梯级输出条件被设置为真。
后期扫描	梯级输出条件被设置为假。

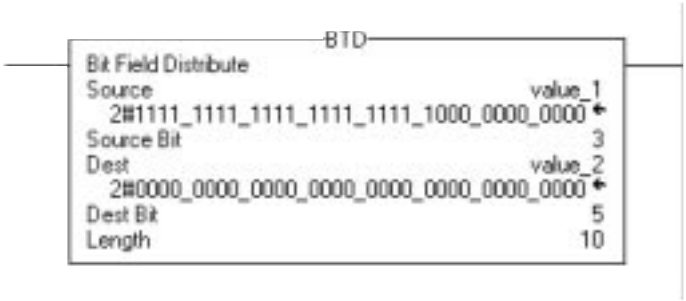
例1 指令被使能时，BTD指令传送value_1内的位。

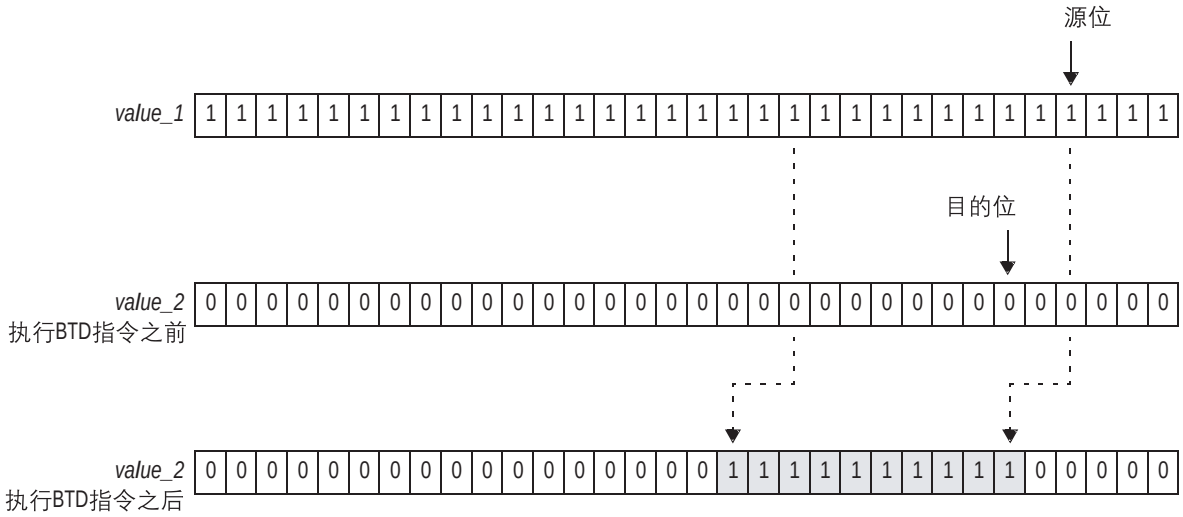




阴影部分表示在value_1内被改变的位。

例2：当指令被使能时，BTD指令传送来自value_1的10个位到value_2。





阴影部分表示在value_2内被改变的位。

带目标的位域分配 (BTDT)

BTDT指令首先将目标数值复制到目的标签。然后指令复制源值的指定位，移动这些位到适当位置，并将其写入目的标签。目标和源值保持不变。

该指令相当于梯形图编程中的BTD指令，参见6 - 11页。

操作数：

```
BTDT(BTDT_tag);
```

结构文本

变量：	数据类型：	格式：	说明：
BTDT标签	FBD_BIT_FIELD_DISTRIBUTE	结构体	BTDT结构体

功能块

操作数：	数据类型：	格式：	说明：
BTDT标签	FBD_BIT_FIELD_DISTRIBUTE	结构体	BTDT结构体

FBD_BIT_FIELD_DISTRIBUTE结构体

输入参数：	数据类型：	说明：
EnableIn	BOOL	**功能块**： 如果被清零，则该指令不执行且输出不更新。 如果被置位，则执行指令。 缺省状态为置位。 **结构文本**： 无影响。执行指令。
Source	DINT	含有传送到目的标签数据位的输入值。 有效值 = 任一整数
SourceBit	DINT	源值内各位位置(开始传送的最低位号)。 有效值 = 0-31
Length	DINT	要传送的位数 有效值 = 1-32
DestBit	DINT	目的标签内各位位置(开始复制位的最低位号)。 有效值 = 0-31
Target	DINT	在来自源值的传送位之前，输入到目的标签的数值。 有效值 = 任一整型数
输出参数：	**数据类型：**	**说明：**
EnableOut	BOOL	指令产生一有效结果。
Dest	DINT	位传送操作结果。该输出影响算术状态标志。

说明：当指令被使能时，BTD指令从源值复制一组位到目的标签。这组位用源位(该组的最低位号)和长度(要复制位的数量)标识。目的位标识目的标签内开始的最低位号。源值保持不变。

如果位字段的长度超出了目的标签，则指令不保存超出位。超出位不会与下个字相互重叠。

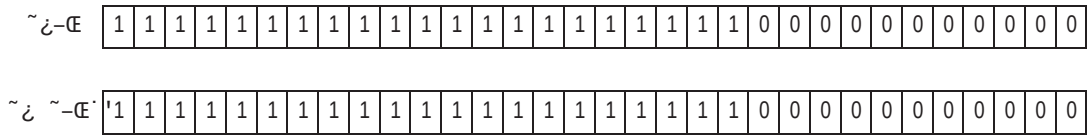
算术状态标志：影响算术状态标志。

故障条件：无

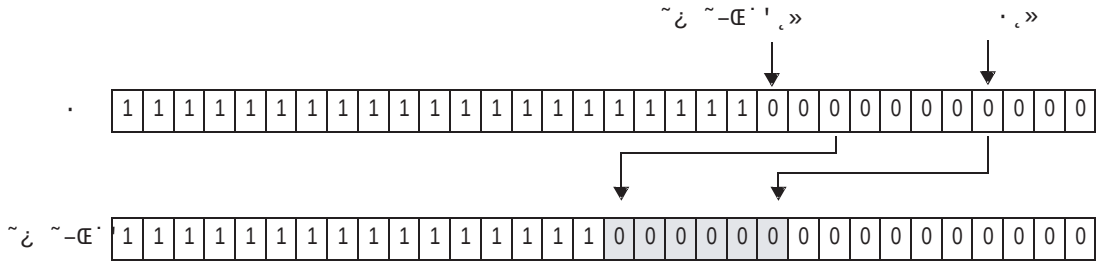
执行：

条件：	功能块动作：	结构文本动作：
预扫描	不动作。	不动作。
指令首次扫描	不动作。	不动作。
指令首次运行	不动作。	不动作。
EnableIn被清零	EnableOut被清零，指令不执行，且输出不更新。	无影响。
EnableIn被置位	指令执行。 EnableOut被置位。	EnableIn一直被置位。 指令执行。
后期扫描	不动作。	不动作。

举例：1. 控制器将目标值复制到目的标签。



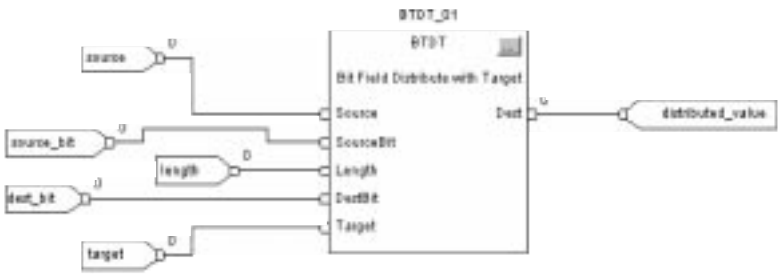
2. 源位和长度指定源值内哪些位要复制到目的标签，起始于目的标签位。源值和目标值保持不变。



结构文本

```
BTDT_01.Source := source;
BTDT_01.SourceBit := source_bit;
BTDT_01.Length := length;
BTDT_01.DestBit := dest_bit;
BTDT_01.Target := target;

BTDT(BTDT_01);

distributed_value := BTDT_01.Dest;
```

功能块



清零(CLR)

CLR指令清零目的标签的全部位。

操作数：

梯形图

操作数：	数据类型：	格式：	说明：
Destination 目的	SINT INT **DINT** **REAL**	标签	要清零的标签

结构文本

```
dest := 0;
```

结构文本中无CLR指令。赋0值给需清零的标签。该赋值语句清零dest。

关于结构文本中表达式和赋值语句的语法信息，参见结构文本编程。

说明： CLR指令清零目的标签内的全部位。

算术状态标志： 影响算术状态标志。

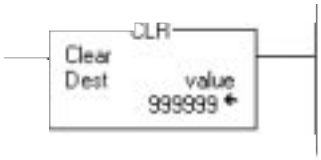
故障条件： 无

执行：

条件：	梯形图动作：
预扫描	梯级输出条件被设置为假。
梯级输入条件为假	梯级输出条件被设置为假。
梯级输入条件为真	指令清零目的标签。 梯级输出条件被设置为真。
后期扫描	梯级输出条件被设置为假。

举例：清零value内的全部位。

梯形图



结构文本

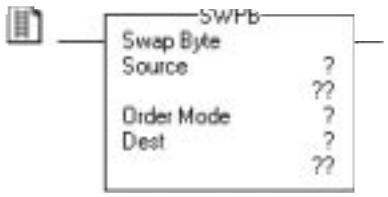
```
value := 0;
```

交换字节(SWPB)

SWPB指令重新安排数值中字节顺序。

操作数：

梯形图



操作数：	数据类型：	格式：	输入：		
Source 源值	INT DINT REAL	标签	包含要重新安排字节顺序的标签		
Order Mode 排序方式			如果源是：	要将字节改变为以下方式 (每个字母代表不同的字节)：	则选择：
			INT	n/a	任何选项
			DINT	ABCD DCBA	REVERSE(或输入0)
			REAL	ABCD CDAB	WORD(或输入1)
				ABCD BADC	HIGH/LOW(或输入2)
Destination 目的	INT DINT REAL	标签	存储以新顺序排序后字节的标签		
			如果源是：	则目的标签必须是：	
			INT	INT	
				DINT	
			DINT	DINT	
			REAL	REAL	

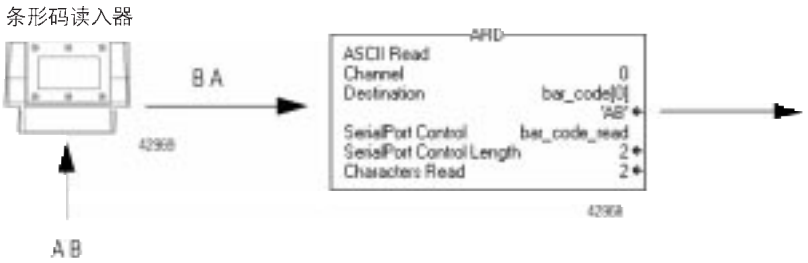
结构文本

```
SWPB(Source,OrderMode,Dest);
```

该指令操作数与梯形图SWPB指令的操作数相同。如果用户选择HIGH/LOW排序方式，则输入HIGHLOW或HIGH_LOW(没有斜线)。

说明：SWPB指令重新调整源值字节的顺序。结果存放在目的标签。

当用户读或写ASCII字符时，不必交换字符。ASCII读和写指令(ARD，ARL，AWA，AWT)自动交换字符，如下图所示。



标签名称	值	显示字体	数据类型
bar_code[0]	AB	ASCII	INT

算术状态标志位：不影响

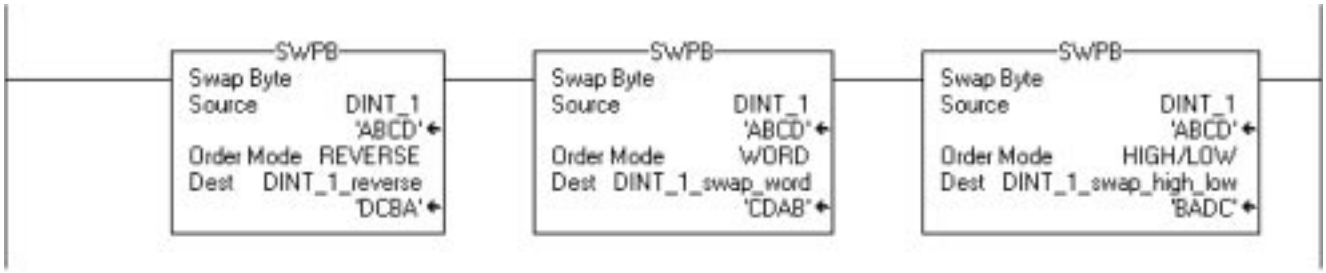
故障条件：无

执行：

条件：	功能块动作：	结构文本动作：
预扫描	梯级输出条件被设置为假。	不动作
梯级输入条件为假	梯级输出条件被设置为假。	无影响
梯级输入条件为真	梯级输出条件被设置为真。	无影响
EnableIn被置位	无影响	EnableIn一直被置位。 指令执行。
指令执行	指令重新调整指定字节顺序。	指令重新调整指定字节顺序。
后期扫描	梯级输出条件被设置为假。	不动作。

例1：三条SWPB指令按照不同的排序方式将DINT_1的字节重新排序。显示字体为ASCII，每个字符代表一个字节。每条指令将字节按照新的顺序放入不同的目的标签内。

梯形图



结构文本

```
SWPB(DINT_1,REVERSE,DINT_1_reverse);
SWPB(DINT_1,WORD,DINT_1_swap_word);
SWPB(DINT_1,HIGHLOW,DINT_1_swap_high_low);
```

例2：下面的实例将数组内每个元素的字节顺序翻转。RSLogix 5000的一个工程文件中包含本例，请打开RSLogix 5000\Projects\Samples文件夹下的Swap_Bytes_in_Array.ACD文件。

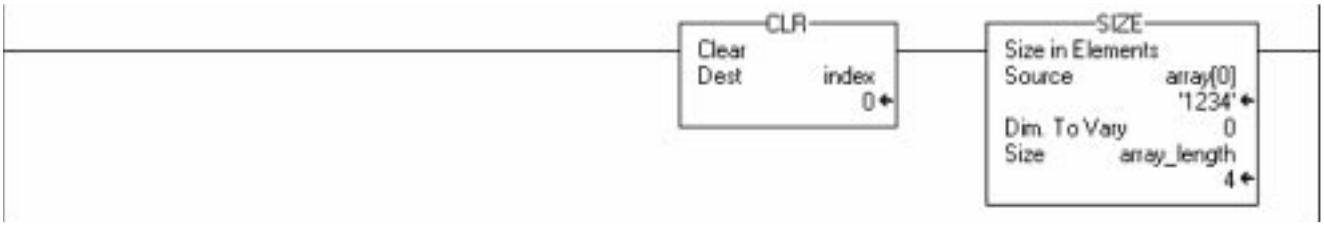
1. 初始化标签。SIZE指令查找array内的元素号，并将其存储在array_length内。下面的指令使用该值确定例程何时对数组内的全部元素进行操作。

2. 将array中一个元素的字节顺序翻转。
   - SWPB指令翻转由index值指定的元素号的字节。例如，当index等于0时，SWPB指令对array[0]进行操作。
   - ADD指令增加index的值。下次执行本指令时，SWPB指令将对数组的下一个元素进行操作。

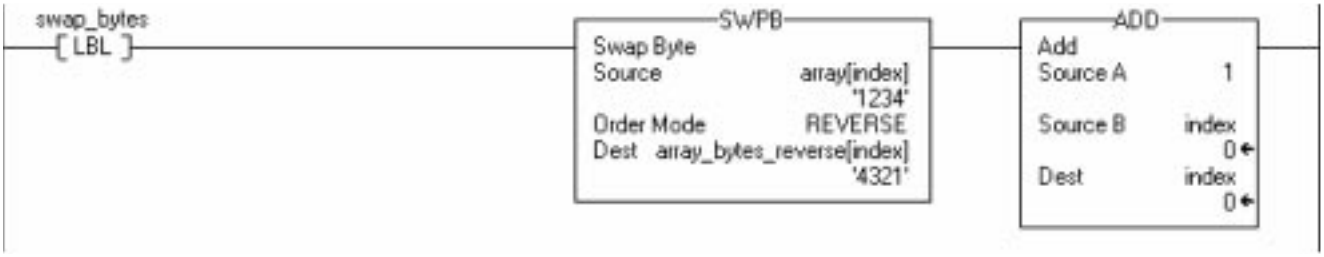
3. 确定SWPB指令何时对数组内的全部元素进行操作。
   - 如果index小于数组内的元素数量(array_length)，则继续对数组的下一个元素进行操作。
   - 如果index等于array_length，则SWPB已经对数组内全部元素都进行了操作。

梯形图

初始化标签



翻转字节顺序



确定SWPB指令是否对数组内的全部元素进行操作

结构文本

```
index := 0;
SIZE(array[0],0,array_length);
REPEAT
   SWPB(array[index],REVERSE,array_bytes_reverse[index]);
   index := index + 1;
UNTIL(index >= array_length)END_REPEAT;
```

按位与(AND)

AND指令执行源A值与源B值的按位与运算，并存放结果于目的标签。

若要执行逻辑与，参见6-35页。

操作数：

梯形图

操作数：	数据类型：	格式：	说明：
Source A 源A	SINT INT **DINT** SINT或INT数据类型标签通过零-填充转换为DINT类型数值。	立即数 标签	与源B操作数进行与运算的数值
Source B 源B	SINT INT **DINT** SINT或INT数据类型标签通过零-填充转换为DINT类型数值。	立即数 标签	与源A操作数进行与运算的数值
Destination 目的	SINT INT **DINT**	标签	存储结果的标签

```
dest := sourceA AND sourceB
```

结构文本

使用AND或“&”符号作为表达式内的运算符。该表达式计算sourceA AND sourceB。

关于结构文本中表达式的语法信息，参见结构文本编程。

功能块

操作数：	数据类型：	格式：	说明：
AND标签	FBD_LOGICAL	结构体	AND结构体

FBD_LOGICAL结构体

输入参数:	数据类型:	说明:
EnableIn	BOOL	输入使能。如果被清零，该指令不执行也且输出不更新。 缺省状态为置位。
SourceA	DINT	与源B进行与运算的值。 有效值 = 任一整数
SourceB	DINT	与源A进行与运算的值。 有效值 = 任一整数
输出参数:	**数据类型:**	**说明:**
EnableOut	BOOL	指令产生一有效结果。
Dest	DINT	指令执行结果。该输出影响算术状态标志。

说明: 当指令被使能时，指令进行与运算:

如果源A内的位是:	如果源B内的位是:	目的标签内的位是:
0	0	0
0	1	0
1	0	0
1	1	1

如果用混合整型数据类型，则指令用0值来填充小整数数据类型的高位，以使它们与最大整型数据类型的大小相同。

算术状态标志位: 影响算术状态标志。

故障条件: 无

执行:

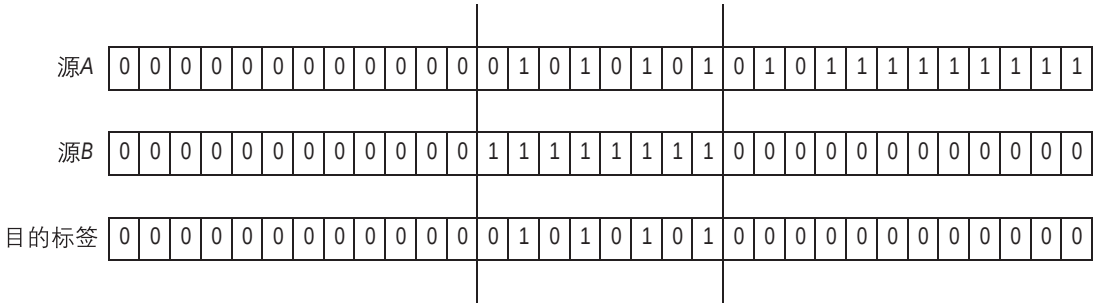
梯形图

条件:	动作:
预扫描	梯级输出条件被设置为假。
梯级输入条件为假	梯级输出条件被设置为假。
梯级输入条件为真	指令执行按位与运算。 梯级输出条件被设置为真。
后期扫描	梯级输出条件被设置为假。

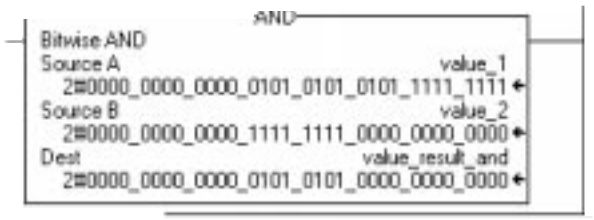
功能块

条件：	动作：
预扫描	不动作。
指令首次扫描	不动作。
指令首次运行	不动作。
EnableIn被清零	EnableOut被清零。
EnableIn被置位	指令执行。 EnableOut被置位。
后期扫描	不动作。

举例：当指令被使能时，AND指令执行源A操作数与源B值的按位与运算，并将结果存放在Dest内。



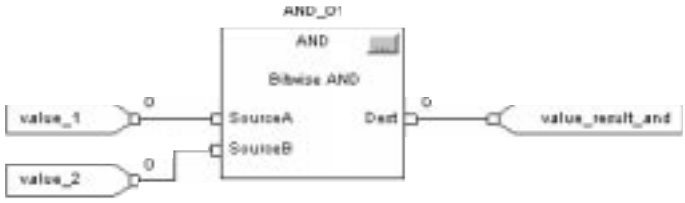
梯形图



结构文本

```
value_result_and := value_1 AND value_2;
```

功能块



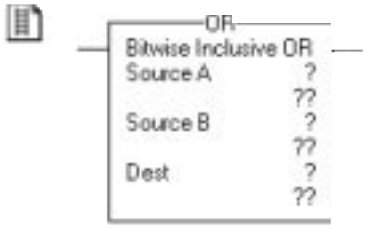
按位或(OR)

OR指令执行源A与源B操作数的按位或运算，并将结果存放在目的标签。

要执行逻辑或指令，参见页码6-38。

操作数：

梯形图



操作数：	数据类型：	格式：	说明：
Source A 源A	SINT INT **DINT** SINT或INT数据类型标签通过零-填充转换为DINT类型数值。	立即数 标签	与源B操作数进行或运算的数值
Source B 源B	SINT INT **DINT** SINT或INT数据类型标签通过零-填充转换为DINT类型数值。	立即数 标签	与源A操作数进行或运算的数值
Destination 目的标签	SINT INT **DINT**	标签	存储运算结果

结构文本

```
dest := sourceA OR sourceB
```

使用OR作为表达式中运算符。该表达式计算sourceA OR sourceB。

关于结构文本中表达式的语法信息，参见结构文本编程。

功能块

操作数：	数据类型：	格式：	说明：
OR标签	FBD_LOGICAL	结构体	OR结构体

FBD_LOGICAL结构体

输入参数：	数据类型：	说明：
EnableIn	BOOL	输入使能。如果被清零，该指令不执行且输出不更新。 缺省状态为置位。
SourceA	DINT	与源B进行或运算的值。 有效值 = 任一整数
SourceB	DINT	与源A进行或运算的值。 有效值 = 任一整数
输出参数：	**数据类型：**	**说明：**
EnableOut	BOOL	指令产生一有效结果。
Dest	DINT	指令执行结果。该输出影响算术状态标志。

说明：当指令被使能时，指令进行或运算：

如果源A内的位是：	如果源B内的位是：	目的标签内的位是：
0	0	0
0	1	1
1	0	1
1	1	1

如果用混合整型数据类型，则指令用0值来填充小整数数据类型的高位，以使它们与最大整型数据类型的大小相同。

算术状态标志位：影响算术状态标志。

故障条件：无

执行：

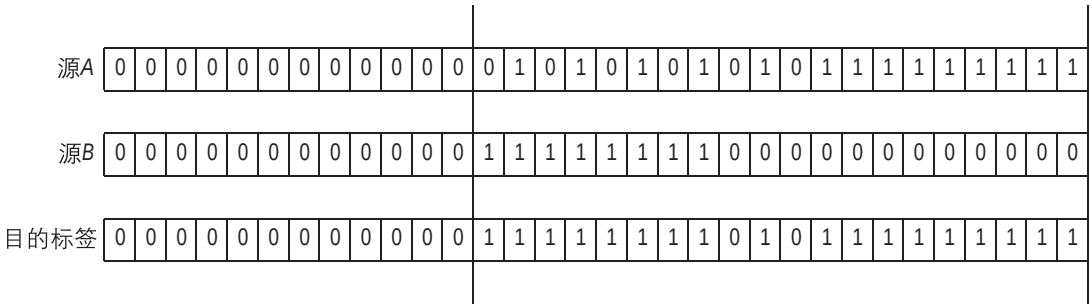
梯形图

条件：	动作：
预扫描	梯级输出条件被设置为假。
梯级输入条件为假	梯级输出条件被设置为假。
梯级输入条件为真	指令执行按位或运算。 梯级输出条件被设置为真。
后期扫描	梯级输出条件被设置为假。

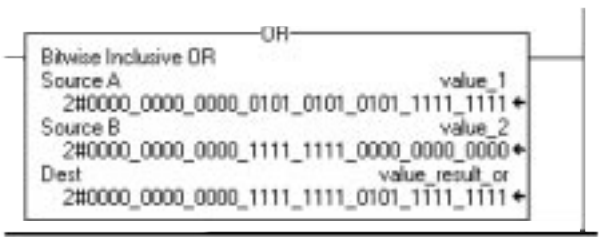
功能块

条件：	动作：
预扫描	不动作。
指令首次扫描	不动作。
指令首次运行	不动作。
EnableIn被清零	EnableOut被清零。
EnableIn被置位	指令执行。 EnableOut被置位。
后期扫描	不动作。

举例：当指令被使能时，OR指令执行SourceA与SourceB的按位或运算，并存放结果于目的标签内。



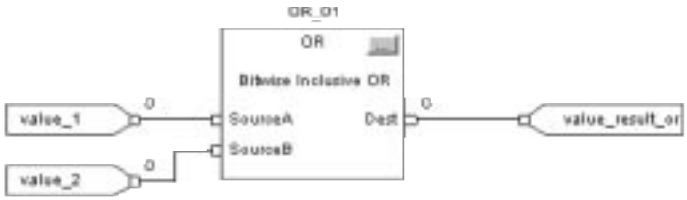
梯形图



结构文本

```
value_result_or := value_1 OR value_2;
```

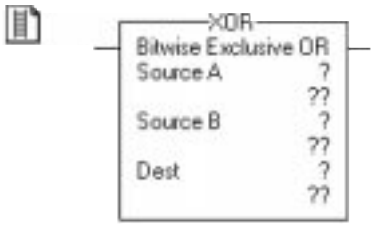
功能块



按位异或(XOR)

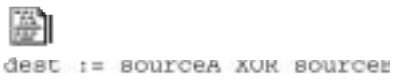
XOR指令执行源A与源B操作数的按位异或运算，并将结果保存在目的标签。要执行逻辑异或指令，参见6-41页。

操作数：



梯形图

操作数：	数据类型：	格式：	说明：
Source A 源A	SINT INT DINT	立即数 标签	与源B操作数进行异或运算的数值
	SINT或INT数据类型标签通过零-填充转换为DINT类型数值。		
Source B 源B	SINT INT DINT	立即数 标签	与源A操作数进行异或运算的数值
	SINT或INT数据类型标签通过零-填充转换为DINT类型数值		
Destination 目的标签	SINT INT DINT	标签	存储结果的标签

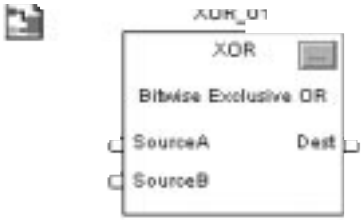


结构文本

使用XOR指令作为表达式中运算符。该表达式计算sourceA XOR sourceB。

关于结构文本中表达式的语法信息，参见结构文本编程。

功能块



操作数：	数据类型：	格式：	说明：
XOR标签	FBD_LOGICAL	结构体	XOR结构体

FBD_LOGICAL结构体

输入参数:	数据类型:	说明:
EnableIn	BOOL	输入使能。如果被清零，该指令不执行输出也不更新。 缺省状态为置位。
SourceA	DINT	与源B进行异或运算的值。 有效值 = 任一整数
SourceB	DINT	与源A进行异或运算的值。 有效值 = 任一整数
输出参数:	**数据类型:**	**说明:**
EnableOut	BOOL	指令产生一有效结果。
Dest	DINT	指令执行结果。该输出影响算术状态标志。

说明: 当指令被使能时，指令进行异或运算：

如果源A内的位是:	如果源B内的位是:	目的标签内的位是:
0	0	0
0	1	1
1	0	1
1	1	0

如果用混合整型数据类型，则指令用0值来填充小整数数据类型的高位，以使它们与最大整型数据类型的大小相同。

算术状态标志位: 影响算术状态标志。

故障条件: 无

执行:

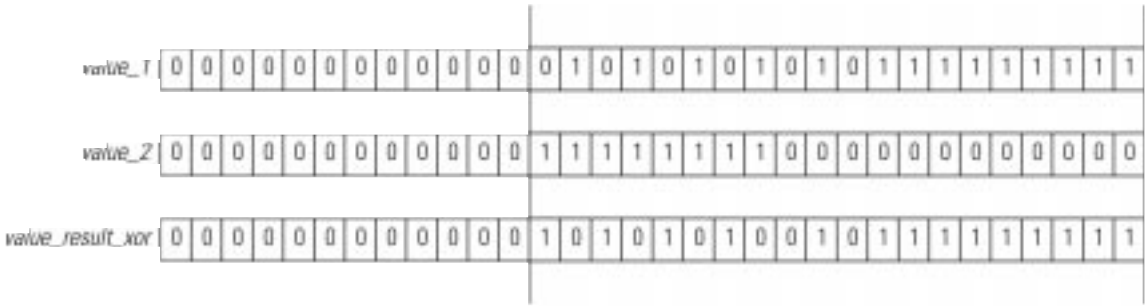
梯形图

条件:	动作:
预扫描	梯级输出条件被设置为假。
梯级输入条件为假	梯级输出条件被设置为假。
梯级输入条件为真	指令执行按位异或运算。 梯级输出条件被设置为真。
后期扫描	梯级输出条件被设置为假。

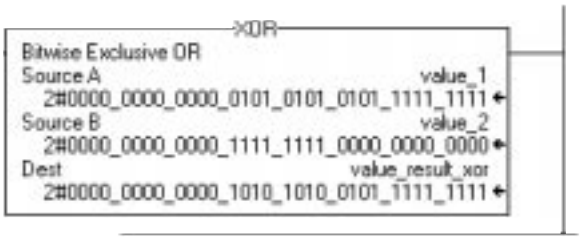
功能块

条件：	动作：
预扫描	不动作。
指令首次扫描	不动作。
指令首次运行	不动作。
EnableIn被清零	EnableOut清零。
EnableIn被置位	指令执行。 EnableOut被置位。
后期扫描	不动作。

举例：当指令被使能时，XOR指令执行SourceA与SourceB的按位异或运算，并存放结果于目的标签内。



梯形图



结构文本

```
value_result_xor := value_1 XOR value_2;
```

功能块

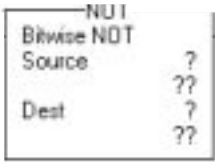
按位非(NOT)

NOT指令执行源值的按位非运算，并将结果存放在目的标签。

要执行逻辑非运算，参见6-44页。

操作数：

梯形图



操作数：	数据类型：	格式：	说明：
Source 源	SINT INT DINT	立即数 标签	执行非运算的数值
	SINT或INT数据类型标签通过零-填充转换为DINT类型数值。		
Destination 目的	SINT INT DINT	标签	存储结果的标签

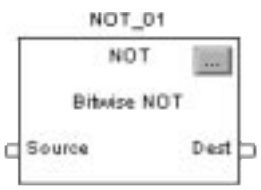
结构文本

```
dest := NOT source
```

使用NOT指令作为表达式中运算符。该表达式计算NOT source。

关于结构文本中表达式的语法信息，参见结构文本编程。

功能块



操作数：	数据类型：	格式：	说明：
NOT标签	FBD_LOGICAL	结构体	NOT结构体

FBD_LOGICAL结构体

输入参数：	数据类型：	说明：
EnableIn	BOOL	使能输入。如果被清零，该指令不执行且输出也不更新。 NOT结构体 缺省状态为置位。
Source	DINT	进行非运算的值。 有效值 = 任一整型数
输出参数：	**数据类型：**	**说明：**
EnableOut	BOOL	指令产生一有效结果。
Dest	DINT	指令执行结果。该输出影响算术状态标志。

说明：当指令被使能时，指令进行非运算：

如果源值中位是：	目的标签中位是：
0	1
1	0

如果用混合整型时，则该指令用0值来填充较小整数数据类型的高位，以使它们与最大整型数据类型的大小相同。

算术状态标志位：影响算术状态标志。

故障条件：无

执行：

梯形图

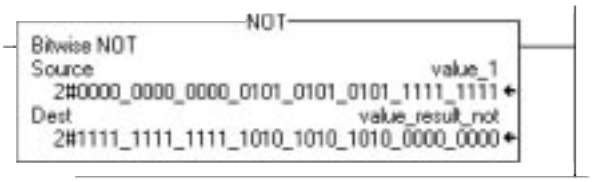
条件：	动作：
预扫描	梯级输出条件被设置为假。
梯级输入条件为假	梯级输出条件被设置为假。
梯级输入条件为真	指令执行按位非运算。 梯级输出条件被设置为真。
后期扫描	梯级输出条件被设置为假。

功能块

条件：	动作：
预扫描	不动作。
指令首次扫描	不动作。
指令首次运行	不动作。
EnableIn被清零	EnableOut被清零。
EnableIn被置位	指令执行。 EnableOut被置位。
后期扫描	不动作。

举例：当指令被使能时，NOT指令执行源值的按位非运算，并存放结果于目的标签内。

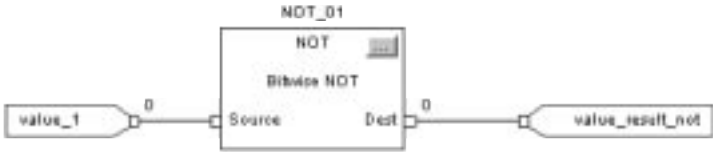
梯形图



结构文本

```
value_result_not := NOT value_1;
```

功能块



逻辑与 (BAND)

BAND指令最多可将8个布尔型输入做逻辑与运算。

要执行按位与运算，参见6-23页。

操作数：

结构文本

```
IF operandA AND operandB THEN
    <statement>;
END IF;
```

使用AND或符号 ″&″ 作为表达式内运算符。该操作数必须为BOOL型数值或计算BOOL型数值的表达式。该表达式判断operandA 与operandB后是否为置位(真)。

关于结构文本中表达式的语法信息，参见结构文本编程。

功能块

操作数：	类型：	格式：	说明：
BAND 标签	FBD-BOOLEAN-AND	结构体	BAND结构体

FBD_BOOLEAN_AND结构体

输入参数：	数据类型：	说明：
EnableIn	BOOL	使能输入。如果被清零，指令不执行且输出不更新。 缺省状态为置位。
In1	BOOL	第一个布尔量输入。 缺省状态为置位。
In2	BOOL	第二个布尔量输入。 缺省状态为置位。
In3	BOOL	第三个布尔量输入。 缺省状态为置位。
In4	BOOL	第四个布尔量输入。 缺省状态为置位。
In5	BOOL	第五个布尔量输入。 缺省状态为置位。
In6	BOOL	第六个布尔量输入。 缺省状态为置位。
In7	BOOL	第七个布尔量输入。 缺省状态为置位。
In8	BOOL	第八个布尔量输入。 缺省状态为置位。
输出参数：	**数据类型：**	**说明：**
EnableOut	BOOL	使能输出。
Out	BOOL	指令输出。

说明：BAND指令可对8个布尔输入量进行与运算。未用的输入量缺省设置为(1)。

Out=In1 AND In2 AND In3 AND In4 AND In5 AND In6 AND In7 AND In8

算术状态标志位：不影响

故障条件：无

执行：

条件：	功能块动作：
预扫描	不动作。
指令首次扫描	不动作。
指令首次执行	不动作。
EnableIn被清零	EnableOut被清零。
EnableIn被置位	执行指令。 EnableOut被置位。
后期扫描	不动作。

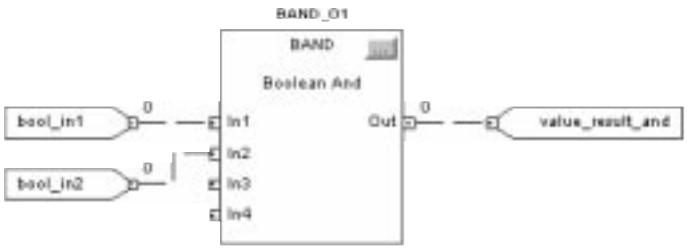
例1：此例对bool_in1和bool_in2进行与运算，并将结果保存在value_result_and中。

如果bool_in1为:	如果bool_in2为:	则value_result_and为:
0	0	0
0	1	0
1	0	0
1	1	1

结构文本

```
value_result_and := bool_in1 AND bool_in2;
```

功能块



例2：如果bool_in1 和bool_in2 都被置位(真)，则ligbt1被置位(接通)。否则，ligbt1被清零(关断)。

结构文本

```
IF bool_in1 AND bool_in2 THEN
        light1 := 1;
ELSE
        light1 := 0;
END_IF;
```

逻辑或(BOR)

BOR指令最多可对8个布尔输入量进行逻辑或运算。

要执行按位或运算，参见6-26。

操作数：

结构文本

```
IF operandA OR operandB THEN
    <statement>;
END IF;
```

使用OR作为表达式内的运算符。其操作数必须为BOOL型数值或计算出BOOL型数值的表达式。该表达式判断operandA 或operandB或二者是否被置位(真)。

关于结构文本中表达式的语法信息，参见结构文本编程。

功能块

操作数：	类型：	格式：	说明：
BOR标签	FBD_BOOLEAN_OR	结构体	BOR结构体

FBD_BOOLEAN_OR结构体

输入参数：	数据类型：	说明：
EnableIn	BOOL	使能输入。如果被清零，则指令不执行且输出不更新。 缺省状态为置位。
In1	BOOL	第一个布尔量输入。 缺省状态为清零。
In2	BOOL	第二个布尔量输入。 缺省状态为清零。
In3	BOOL	第三个布尔量输入。 缺省状态为清零。
In4	BOOL	第四个布尔量输入。 缺省状态为清零。
In5	BOOL	第五个布尔量输入。 缺省状态为清零。
In6	BOOL	第六个布尔量输入。 缺省状态为清零。
In7	BOOL	第七个布尔量输入。 缺省状态为清零。
In8	BOOL	第八个布尔量输入。 缺省状态为清零。
输出参数：	**数据类型：**	**说明：**
EnableOut	BOOL	使能输出。
Out	BOOL	指令输出。

说明：BOR指令最多可对8个布尔型输入量进行或运算。未用输入量的输入缺省值为清零状态(0)。

Out=In1 OR In2 OR In3 OR In4 OR In5 OR In6 OR In7 OR In8

算术状态标志位：不影响

故障条件：无

执行：

条件：	功能块动作：
预扫描	不动作。
指令首次扫描	不动作。
指令首次执行	不动作。
EnableIn被清零	EnableOut被清零。
EnableIn被置位	执行指令。 EnableOut被置位。
后期扫描	不动作。

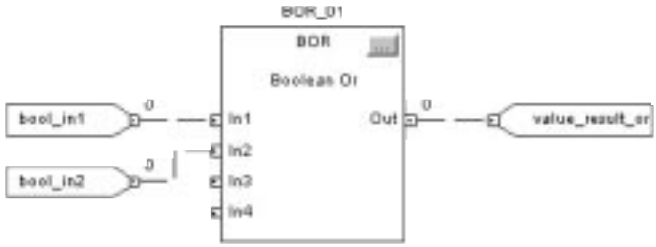
例1：此例中OR指令对bool_in1和bool_in2进行或运算，并将结果保存在value_result_or中。

如果bool_in1为:	如果bool_in2为:	则value_result_or为:
0	0	0
0	1	1
1	0	1
1	1	1

结构文本

```
value_result_or := bool_in1 OR bool_in2;
```

功能块



例2：在该例中，ligbt_1被置位(接通)，如果：

- 仅bool_in1被置位(真)。
- 仅bool_in2被置位(真)。
- bool_in1和bool_in2都被置位(真)。

否则，ligbt_1被清零(关断)。

结构文本

```
IF bool_in1 OR bool_in2 THEN
        light1 := 1;
ELSE
        light1 := 0;
END_IF;
```

逻辑异或(BXOR)

BXOR指令可对两个布尔型输入量进行异或运算。

要执行按位异或运算，参见6-29页。

操作数：

结构文本

```
IF operandA XOR operandB THEN
    <statement>;
END_IF;
```

使用XOR指令作为表达式中运算符。该操作数必须为BOOL型数值或计算出BOOL型数值的表达式。该表达式判断仅operandA或仅operandB是否被置位(真)。

关于结构文本中表达式的语法信息，参见结构文本编程。

功能块

操作数：	类型：	格式：	说明：
BXOR标签	FBD_BOOLEAN_XOR	结构体	BXOR结构体

FBD_BOOLEAN_XOR结构体

输入参数：	数据类型：	说明：
EnableIn	BOOL	使能输入。如果被清零，则指令不执行且输出不更新。 缺省状态为置位。
In1	BOOL	第一个布尔量输入。 缺省状态为置位。
In2	BOOL	第二个布尔量输入。 缺省状态为置位。
输出参数：	**数据类型：**	**说明：**
EnableOut	BOOL	使能输出。
Out	BOOL	指令输出。

说明： BXOR指令对两个布尔输入量做异或运算。

Out= In1 XOR In2

算术状态标志位： 无影响

故障条件： 无

执行：

条件：	功能块动作：
预扫描	不动作。
指令首次扫描	不动作。
指令首次执行	不动作。
EnableIn被清零	EnableOut被清零。
EnableIn被置位	执行指令。 EnableOut被置位。
后期扫描	不动作。

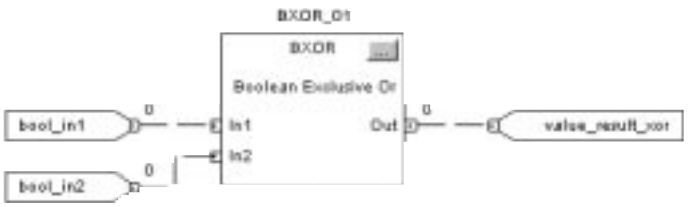
例1：本例中对bool_in1和bool_in2的布尔输入做异或运算，并将结果保存在value_result_xor中。

如果bool_in1为：	如果bool_in2为：	则value_result_xor为：
0	0	0
0	1	1
1	0	1
1	1	0

结构文本

```
value_result_xor := bool_in1 XOR bool_in2;
```

功能块



例2：本例中，ligbt1被置位，如果：

- 仅bool_in1被置位(真)。
- 仅bool_in1被置位(真)。

否则，ligbt1被清零(关断)。

结构文本

```
IF bool_in1 XOR bool_in2 THEN
    light1 := 1;
ELSE
    light1 := 0;
END_IF;
```

逻辑非(BNOT)

BNOT指令可对布尔型输入量进行取反运算。

要执行按位非运算，参见6-32页。

操作数：

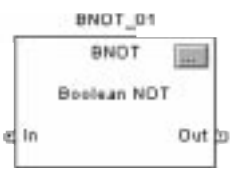
结构文本

```
IF NOT operand THEN
    <statement>;
END IF;
```

使用NOT指令作为表达式内运算符。该操作数必须为BOOL型数值或计算出BOOL型数值的表达式。该表达式判断operandA是否被清零(假)。

关于结构文本中表达式的语法信息，参见结构文本编程。

功能块



操作数：	类型：	格式：	说明：
BNOT标签	FBD_BOOLEAN_NOT	结构体	BNOT结构体

FBD_BOOLEAN_NOT结构体

输入参数：	数据类型：	说明：
EnableIn	BOOL	使能输入。如果被清零，指令不执行，且输出不更新。 缺省状态为置位。
In	BOOL	输入至指令。 缺省状态为置位。
输出参数：	**数据类型：**	**说明：**
EnableOut	BOOL	使能输出。
Out	BOOL	指令的输出。

说明： BNOT指令对布尔输入做取反运算。
Out=NOT In

算术状态标志位： 不影响

故障条件： 无

执行：

条件：	功能块动作：
预扫描	不动作。
指令首次扫描	不动作。
指令首次执行	不动作。
EnableIn 被清零	EnableOut 被清零。
EnableIn 被置位	执行指令。 EnableOut 被置位。
后期扫描	不动作

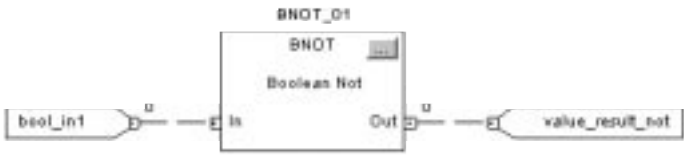
例1：本例中，指令对bool_in1的布尔输入做取反运算，并将结果保存在value_result_not中。

如果bool_in1为：	则value_result_not为：
0	1
1	0

结构文本

```
value_result_not := NOT bool_in1;
```

功能块



例2：如果bool_in1被清零，则ligbt1被清零(关断)。否则，ligbt1被置位(接通)。

结构文本

```
IF NOT bool_in1 THEN
    light1 := 0;
ELSE
    light1 := 1;
END_IF;
```

注释：

数组(文件)/综合指令 (FAL，FSC，COP，CPS，FLL，AVE，SRT，STD，SIZE)

简介

文件/综合指令对数组进行操作。

如果用户需要：	所用指令：	可用语言：	参见页码：
对数组内的数值进行算术，逻辑，移位和函数运算	FAL	梯形图 结构文本(1)	7-7
搜索并比较数组内的数值	FSC	梯形图	7-19
复制一个数组的内容到另一个数组	COP	梯形图 结构文本	7-28
无中断复制一个数组的内容到另一个数组	CPS	梯形图 结构文本	7-28
用指定的数据填充数组	FLL	梯形图 结构文本(1)	7 - 34
计算数组内一组数值平均值	AVE	梯形图 结构文本(1)	7 - 38
按升序排序数组内的一维数据	SRT	梯形图 结构文本	7 - 43
计算数组内一组数值的标准偏差	STD	梯形图 结构文本(1)	7 - 48
查找数组一维元素的数目	SIZE	梯形图 结构文本	7 - 53

(1)不存在等价结构文本指令。使用其它结构文本编程完成相同功能。参见该指令说明。

用户可以混用数据类型，但这样会损失精度，也可能发生取整误差，并且会占用更多时间执行指令。检测S:V位以确认结果是否被截短。

对于梯形图指令，黑体字数据类型是最优数据类型。如果一条指令的全部操作数都使用同一种最优数据类型，则指令执行的速度快而且占用内存少。典型的最优数据类型是DINT或REAL。

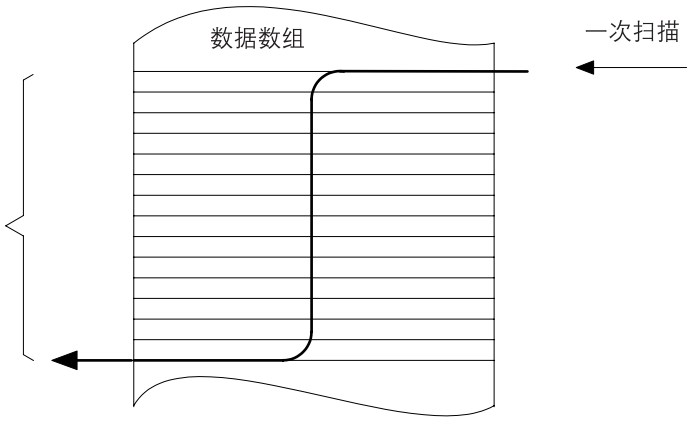
选择操作模式

对于FAL和FSC指令，操作模式表明控制器怎样对数组进行操作。

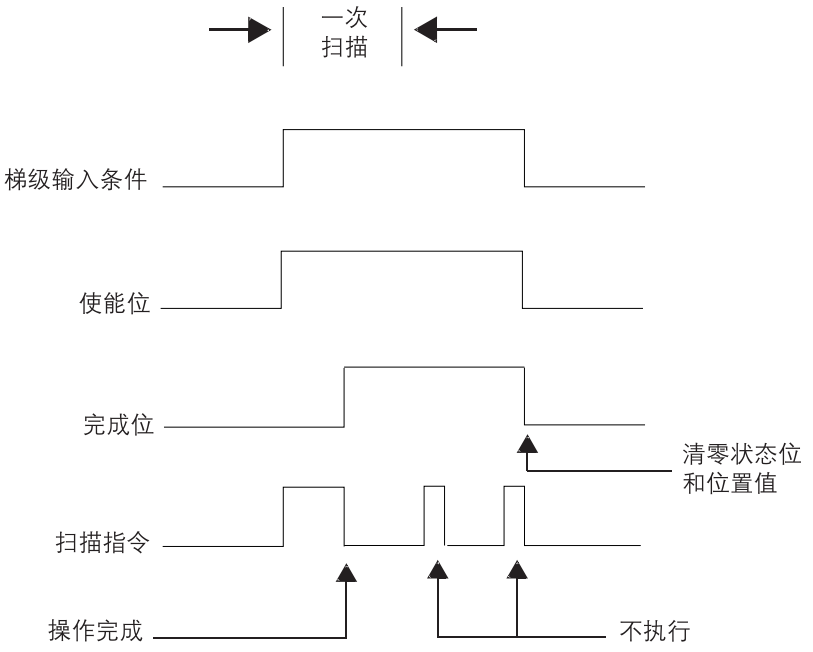
如果用户需要：	选择下列模式：
在继续执行下一条指令前对数组内指定的所有元素进行操作。	整体模式
把对数组内元素的操作分配给多个扫描周期，输入每次扫描要处理的元素数量(1-2147483647)	数值模式
梯级输入条件每由假到真转换一次，只对数组内的一个元素操作	增量模式

整体模式

如果选择整体模式，则在继续执行下一条指令之前对数组内所有指定元素进行操作。这一操作从指令所在梯级输入条件由假到真转换时开始。控制结构体内的位置值.POS指向指令当前正对其操作的数组元素。当位置值(.POS)等于长度值(.LEN)时，指令停止操作。



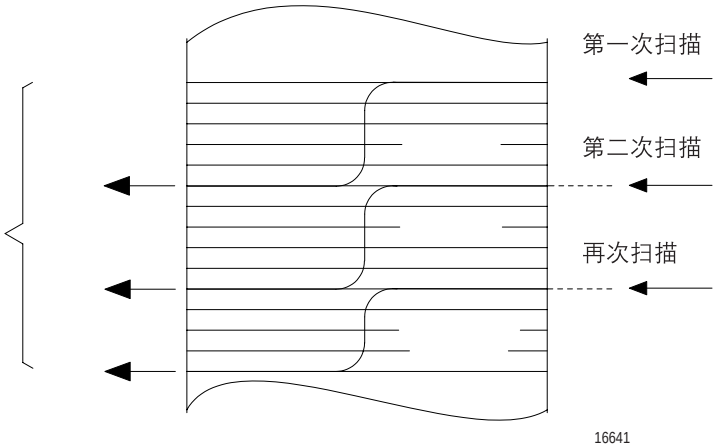
下列时序图说明状态位与指令操作之间的关系。当指令执行完成时，完成位(.DN)被置位。如果梯级输入条件为假，则完成位(.DN)、使能位(.EN)、和位置值(.POS)被清零。只有梯级输入条件再次由假到真转换时，指令才能再次被触发执行。



数值模式

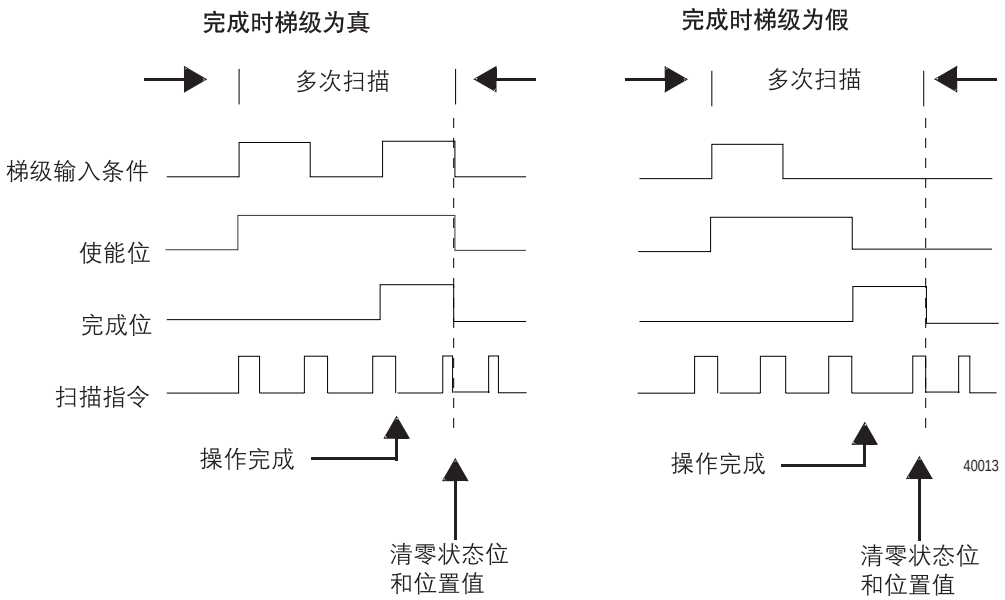
数值模式把对数组的操作分配给多个扫描周期。在对时间要求不很苛刻的数据或大数据量的应用中，这种模式非常有用。输入每次扫描需要处理的元素数量，这样可以缩短扫描时间。

当梯级输入条件由假到真变化时，指令开始执行。一旦被触发，则指令每次被扫描都执行。指令需要被扫描多次才能完成对整个数组的操作。一旦被触发，则梯级输入条件可以反复变化而不中断指令的执行。



重要事项 在完成位.DN被置位之前，不要使用以数值模式操作的文件指令的结果。

下列时序图说明状态位与指令操作的关系。当指令执行完成时，置位完成位.DN。

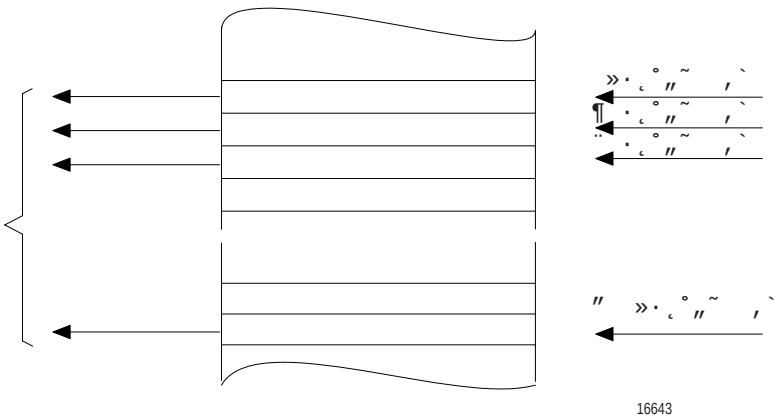


如果在指令完成时，梯级输入条件为真，则在梯级输入条件变为假之前，置位使能位(.EN)和完成位(.DN)。当梯级输入条件变为假时，这些位被清零同时位置值(.POS)也被清零。

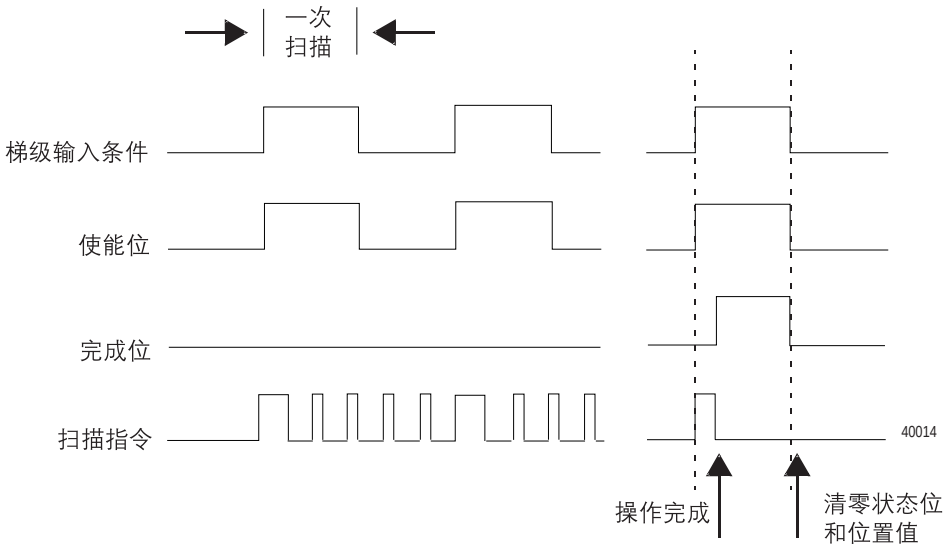
如果在指令完成时梯级输入条件为假，则立即清零使能位(.EN)。使能位(.EN)被清零后的第一次扫描，完成位(.DN)和位置值(.POS)也被清零。

增量模式

增量模式是在每次指令所在梯级输入条件由假到真变化时，只处理数组内的一个元素。



下列时序图说明状态位与指令操作之间的关系。只有在梯级输入条件由假到真转换的一次扫描中，指令执行一次。每次执行时，只处理数组内的一个元素。如果梯级输入条件保持为真的时间多于一个扫描周期，指令也只在第一次扫描期间执行。



当梯级输入条件为真时，使能位(.EN)被置位。当处理完数完组内的最后一个元素时，完成位(.DN)被置位。当数组内的最后一个元素已经处理完时，梯级输入条件变为假时，使能位(.EN)、完成位(.DN)和位置值(.POS)都被清零。

增量模式与数值模式的区别是每次扫描处理元素的数量不同：

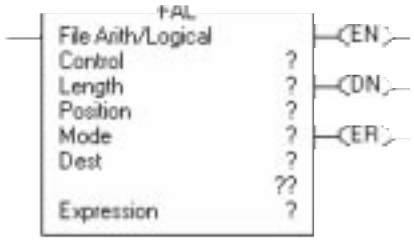
- 数值模式每次指令被扫描时处理多个元素，而且只需梯级输入条件由假到真转换一次即可起动指令执行。并且每次扫描指令都连续处理指定数量的元素，而与梯级输入条件的状态无关。
- 增量模式每次梯级输入条件由假到真转换只处理数组内的一个元素。

文件算术与逻辑(FAL)

FAL指令执行存储在数组内数据的复制、算数、逻辑和函数运算。

操作数：

梯形图



操作数：	数据类型：	格式：	说明：
Control 控制	CONTROL	标签	运算的控制结构体
Length 长度	DINT	立即数	要处理的数组元素数量
Position 位置	DINT	立即数	数组内的当前元素 一般初始值为0
Mode 模式	DINT	立即数	如何对数组内的元素进行操作 选择增量模式(INC)、整体模式(ALL)或者输入一个数值
Destination 目的单元	SINT INT **DINT** **REAL**	标签	存储结果的标签
Expression 表达式	SINT INT **DINT** **REAL**	立即数 标签	由被运算符分开的标签和(或)立即数组成的表达式
	SINT或INT标签通过符号-扩充转换为DINT值。		

结构文本

结构文本中无FAL指令，但用户可使用SIZE指令和FOR…DO或其它循环结构体完成相同功能。

```
SIZE(destination,0,length-1);
FOR position = 0 TO length DO
    destination[position] := numeric_expression;
END_FOR
```

关于结构文本中结构体的语法信息，参加附录C。

CONTROL结构体

助记符：	数据类型：	说明：
.EN	BOOL	使能位标识FAL指令被使能。
.DN	BOOL	当指令已经处理完最后一个元素时(.POS = .LEN)，完成位被置位。
.ER	BOOL	如果在计算表达式时发生溢出(S:V被置位)，则错误位被置位。在程序清零错误位ER之前停止执行指令。位置值.POS包含引起溢出的元素的位置。
.LEN	DINT	长度值指定FAL指令操作的数组内元素的数量。
.POS	DINT	位置值包含指令正访问的当前元素的位置。

说明： FAL指令对数组执行的操作与CPT指令对元素执行的操作相同。

如何用位置值.POS提供整个数组的顺序级数，参见7 - 14页开始的示例。如果表达式中目的标签的下标超出范围，则FAL指令发生主要故障(故障类型4，代码20)。

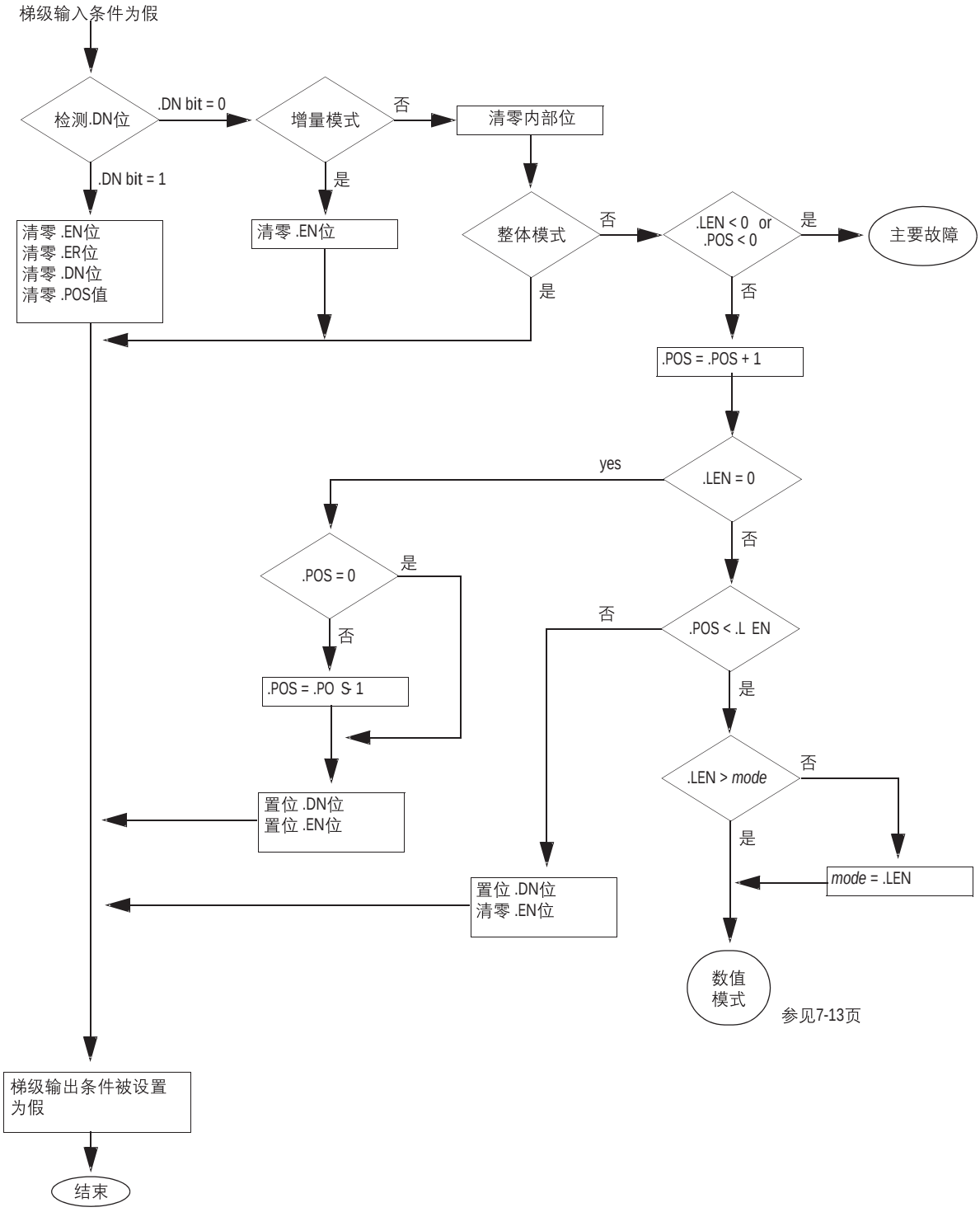
算术状态标志： 影响算术状态标志。

故障条件：

在下列情况将发生主要故障：	故障类型：	故障代码：
下标超出范围	4	20
.POS < 0 或 .LEN < 0	4	21

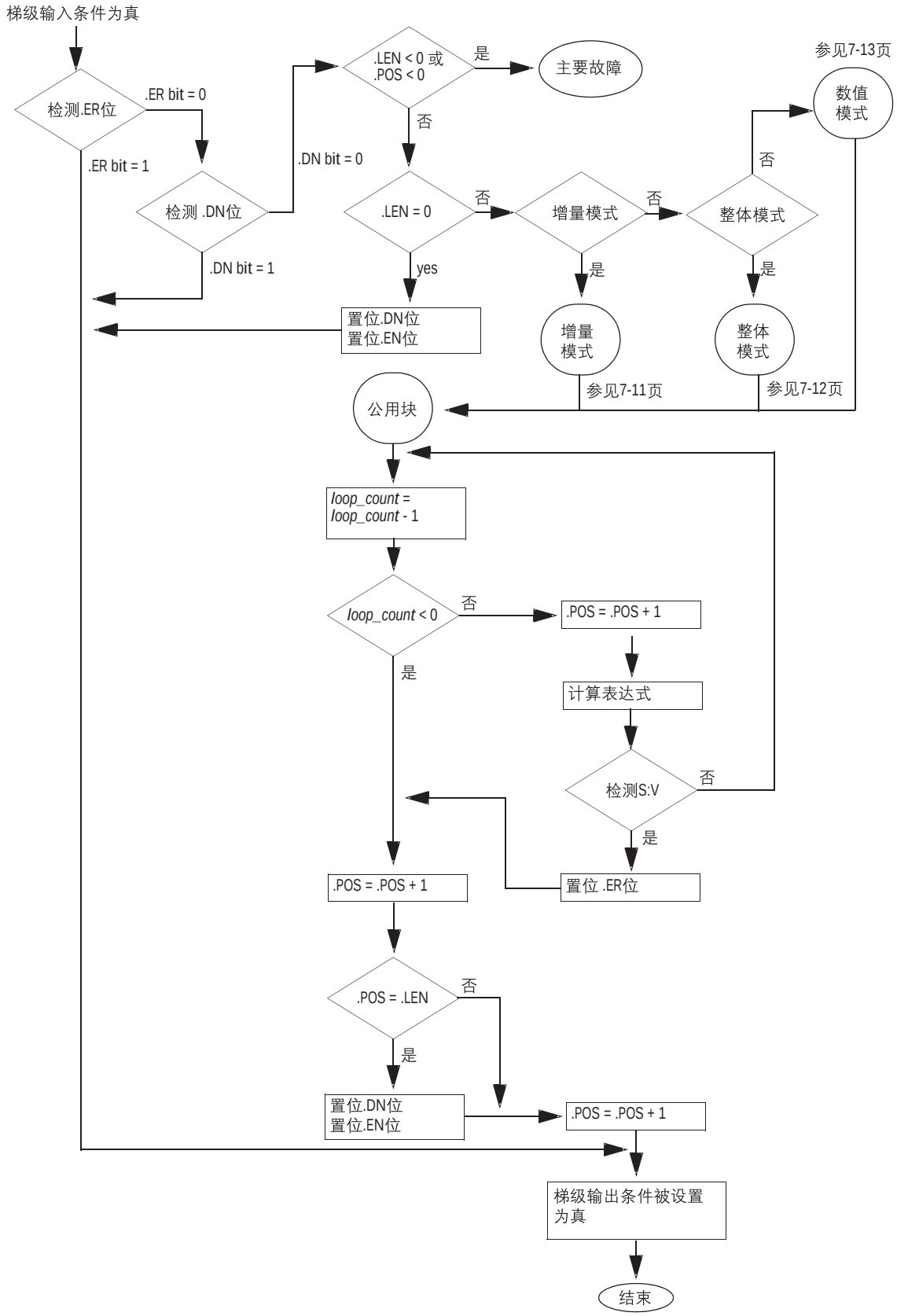
执行：执行：

条件：	**梯形图动作**：
预扫描	梯级输出条件被设置为假。

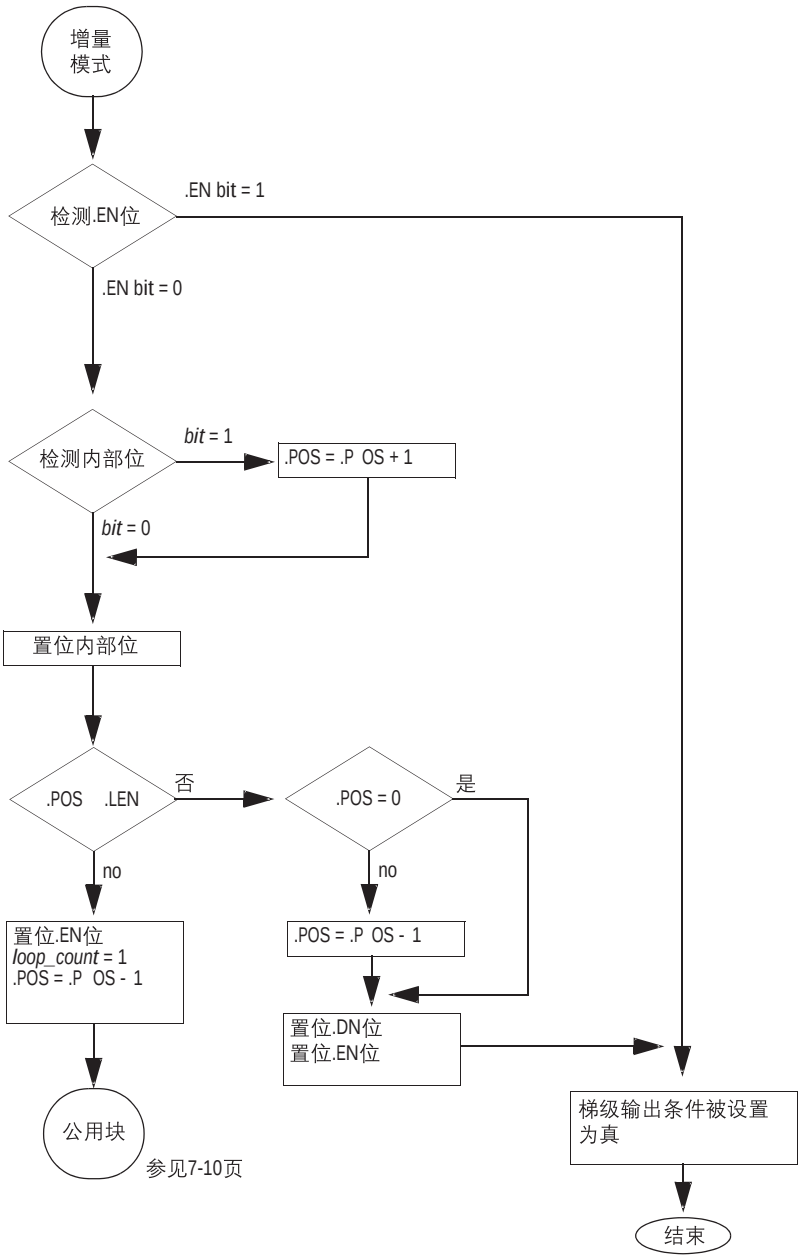


条件：

梯形图动作：

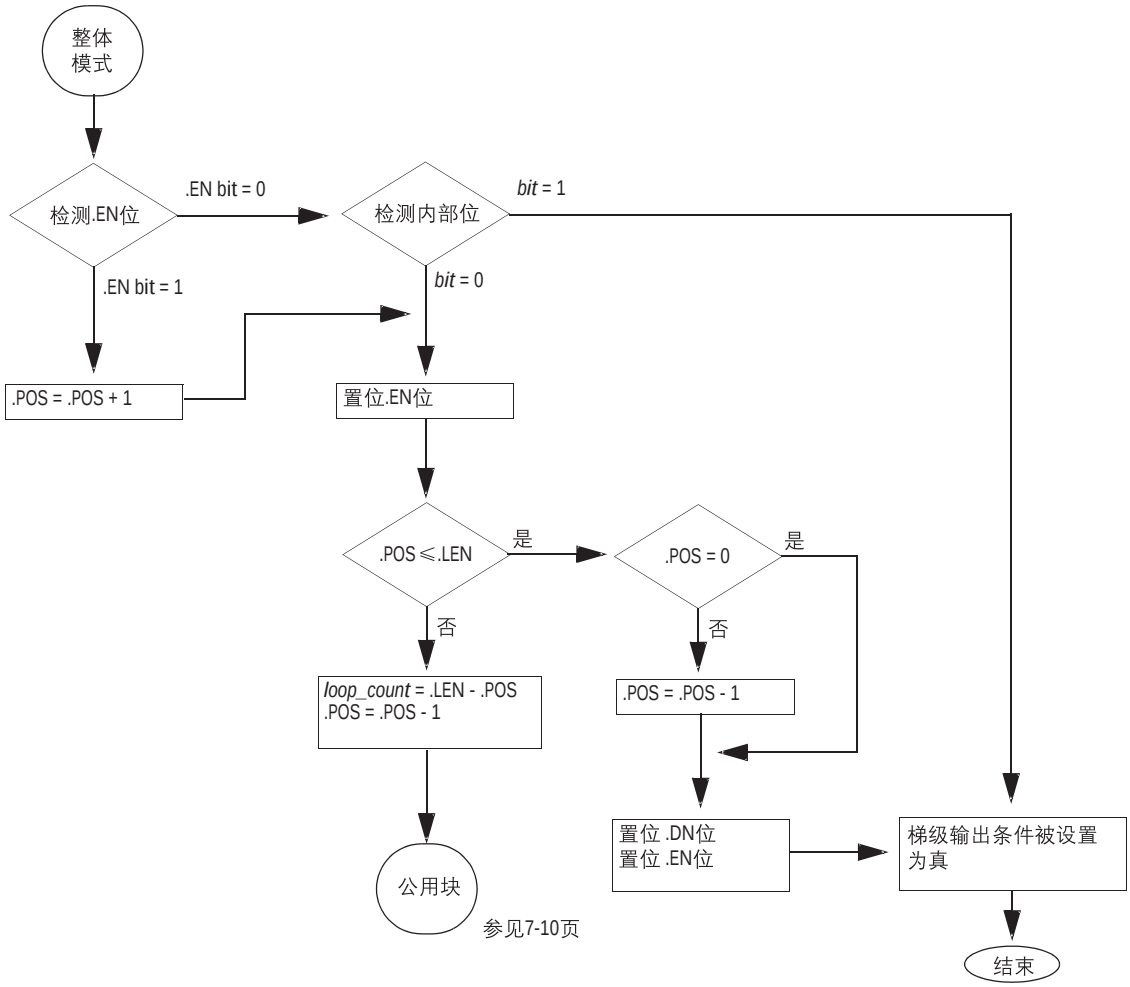

梯级输入条件为真
检测.ER位
.ER bit = 0
.ER bit = 1
检测 .DN位
.DN bit = 0
.DN bit = 1
.LEN < 0 或
.POS < 0
是
主要故障
否
.LEN = 0
否
yes
增量模式
否
是
整体模式
否
是
数值
模式
参见7-13页
置位.DN位
置位.EN位
增量
模式
整体
模式
参见7-11页
参见7-12页
公用块
loop_count =
loop_count - 1
loop_count < 0
否
是
.POS = .POS + 1
计算表达式
检测S:V
否
是
.POS = .POS + 1
置位 .ER位
.POS = .LEN
否
是
置位.DN位
置位.EN位
.POS = .POS + 1
梯级输出条件被设置
为真
结束

条件：　　梯形图动作：


增量
模式
检测.EN位
.EN bit = 1
.EN bit = 0
检测内部位
bit = 1
.POS = .P OS + 1
bit = 0
置位内部位
.POS .LEN
否
.POS = 0
是
no
no
置位.EN位
loop_count = 1
.POS = .P OS - 1
.POS = .P OS - 1
置位.DN位
置位.EN位
公用块
参见7-10页
梯级输出条件被设置
为真
结束

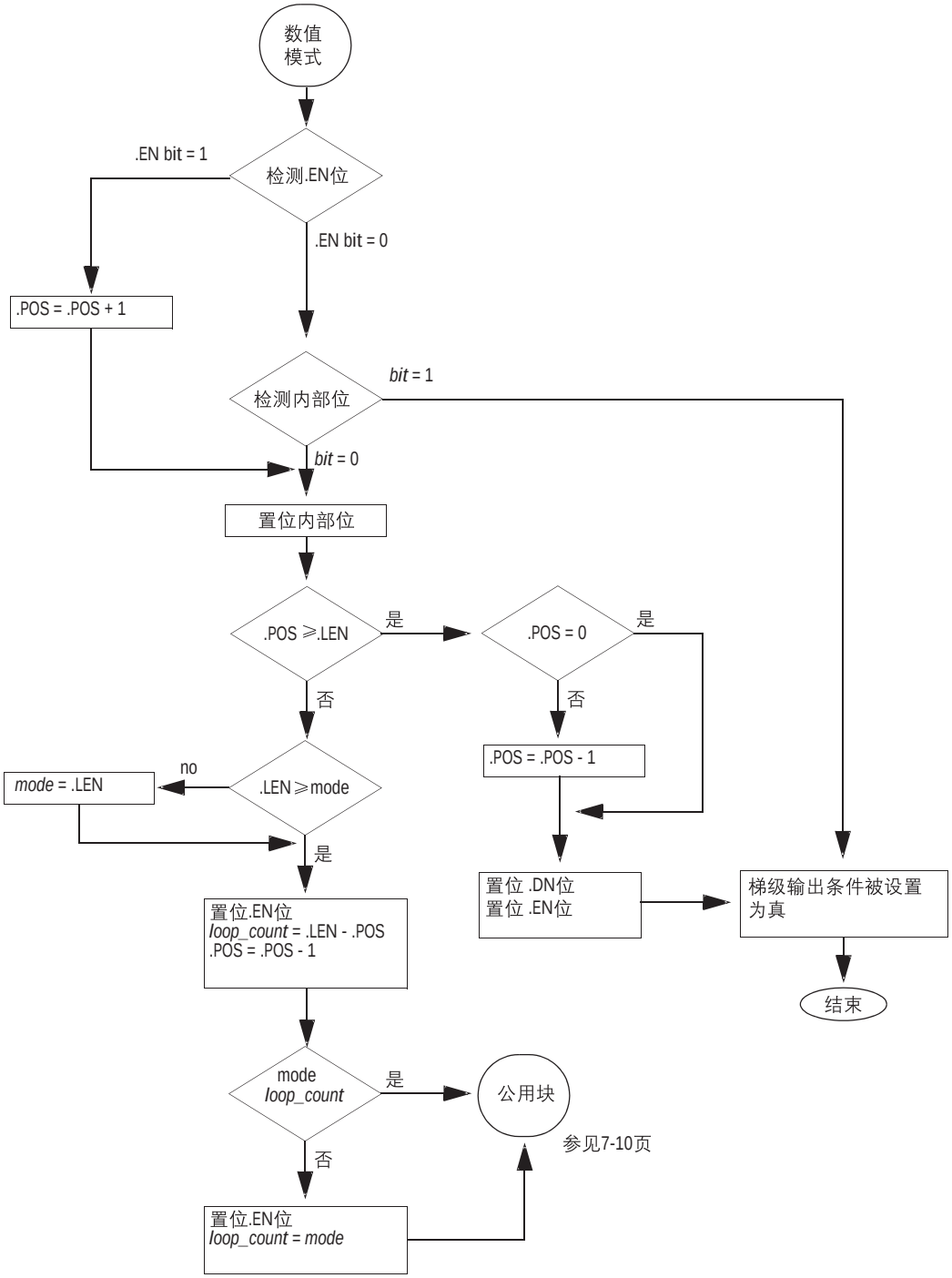
条件：

梯形图动作：


整体
模式
检测.EN位
.EN bit = 0
检测内部位
bit = 1
.EN bit = 1
bit = 0
.POS = .POS + 1
置位.EN位
.POS ⩽ .LEN
是
.POS = 0
是
否
否
loop_count = .LEN - .POS
.POS = .POS - 1
.POS = .POS - 1
置位 .DN位
置位 .EN位
梯级输出条件被设置
为真
公用块
参见7-10页
结束

条件：

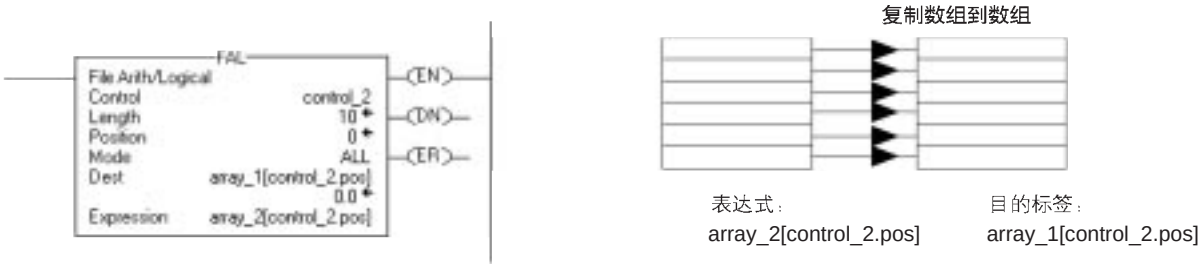
梯形图动作：



后期扫描

梯级输出条件被设置为假。

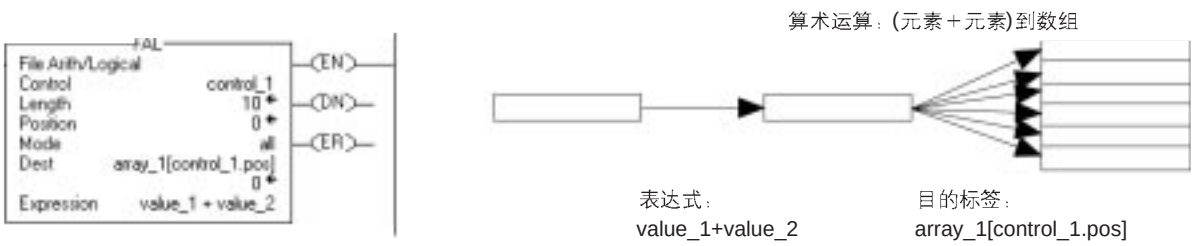
例1：当指令被使能时，FAL指令复制array_2内的每个元素到array_1内的相同位置。



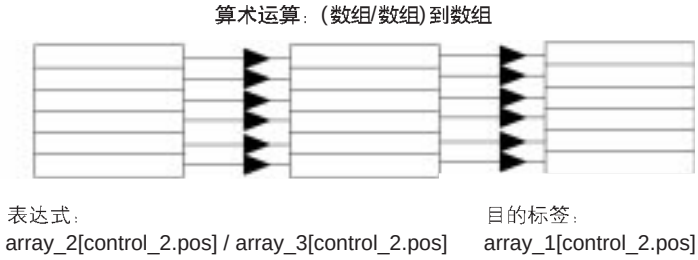
例2：当指令被使能时，FAL指令复制value_1到array_2 的第二维的前10个位置。

例3：每次FAL指令被使能时，均复制array_1的当前值到value_1内。因为FAL指令用增量模式，所以每次指令被使能时，只复制数组内的一个值。该指令下次被使能时，用array_1内的下一个数值覆盖value_1的当前值。

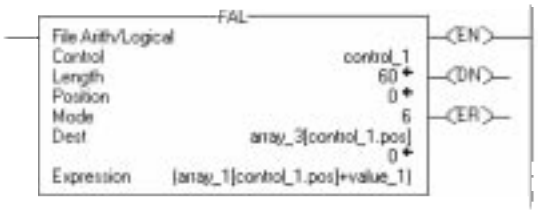
例4：当指令被使能时，FAL指令将value_1和value_2内的数值相加，并将结果保存在array_1的当前位置。

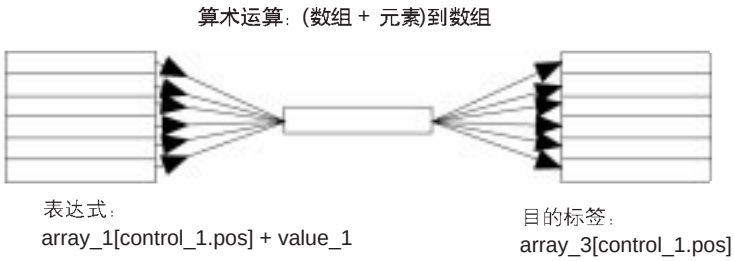


例5：当指令被使能时，FAL指令用array_3内当前位置的数值除array_2内当前位置的数值，并将结果保存在array_1的当前位置。

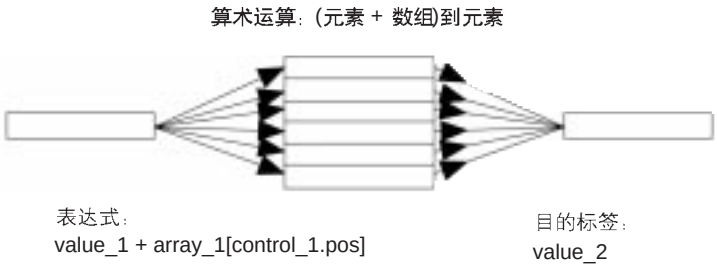


例6：当指令被使能时，FAL指令使array_1当前位置的数值与value_1相加并将结果保存在array_3的当前位置。如果要对array_1和array_3内的全部值进行操作，则指令必须执行10次。

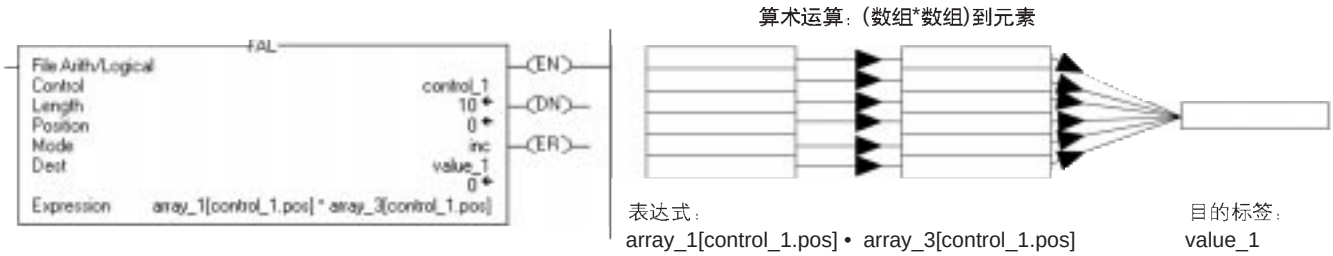




例7：每次FAL指令被使能时，它使value_1与array_1的当前数值相加并将结果保存在value_2中。FAL指令使用增量模式，所以每次指令被使能时，只有数组内的一个数值与value_1相加。下一次指令被使能时，其运算结果覆盖value_2。



例8：当指令被使能时，FAL指令使array_1的当前数值与array_3的当前数值相乘，并将结果保存value_1。FAL指令使用增量模式，所以每次指令被使能，数组内只有一对数值相乘。下一次指令被使能时，其运算结果覆盖value_1。



FAL表达式

FAL指令内的表达式与CPT指令内的表达式编程方法相同。以下部分是关于有效运算符，格式，以及运算顺序的信息，这部分内容两条指令通用。

有效运算符

运算符：	说明：	最优数据类型：
+	加	DINT，REAL
-	减 / 非	DINT，REAL
*	乘	DINT，REAL
/	除	DINT，REAL
**	指数(x 的 y次幂)	DINT，REAL
ABS	绝对值	DINT，REAL
ACS	反余弦	REAL
AND	按位与	DINT
ASN	反正弦	REAL
ATN	反正切	REAL
COS	余弦	REAL
DEG	弧度转换为角度	DINT，REAL
FRD	BCD码转换成整数	DINT
LN	自然对数	REAL

运算符：	说明：	最优数据类型：
LOG	以10为底的对数	REAL
MOD	模数-除	DINT，REAL
NOT	按位非	DINT
OR	按位或	DINT
RAD	角度转换成弧度	DINT，REAL
SIN	正弦	REAL
SQR	平方根	DINT，REAL
TAN	正切	REAL
TOD	整数转换成BCD码	DINT
TRN	截短	DINT，REAL
XOR	按位异或	DINT

表达式格式

对于表达式内的每项运算，必须提供一个或两个操作数(标签或者立即数)。按照下表所示格式书写表达式内的运算符和操作数：

运算符作用于：	使用下述格式：	举例：
一个操作数	运算符(操作数)	ABS(tag_a)
两个操作数	operand_a 运算符 operand_b	• tag_b + 5 • tag_c AND tag_d • (tag_e ** 2) MOD (tag_f / tag_g)

确定运算的顺序

指令按预先规定的顺序而不是按用户写入的顺序，执行写入表达式的操作。用户可以通过圆括号分组的方法来改变运算顺序，强制指令在执行其它运算之前，执行圆括号内的运算。

同级运算的顺序是从左向右执行。

顺序：	运算：
1	( )
2	ABS，ACS，ASN，ATN，COS，DEG，FRD，LN，LOG，RAD，SIN，SQR，TAN，TOD，TRN
3	**
4	-（取反），NOT
5.	*，/，MOD
6.	-（减），+
7.	AND
8.	XOR
9.	OR

文件搜索和比较(FSC)

FSC指令用于逐个元素地比较数组内的值。

操作数：

梯形图

操作数：	数据类型：	格式：	说明：
Control 控制	CONTROL	标签	操作的控制结构体
Length 长度	DINT	立即数	要处理的数组元素数量
Position 位置	DINT	立即数	数组内的偏移量 典型的初始值为0

CONTROL_结构体

助记符：	数据类型：	说明：
.EN	BOOL	使能位标识FSC指令被使能。
.DN	BOOL	当指令已经处理完最后一个元素时(.POS = .LEN)，完成位被置位。
.ER	BOOL	不改变错误位。
.IN	BOOL	禁止位标识FSC指令检测到一个为真的比较。用户必须清零该位才能继续搜索操作。
.FD	BOOL	发现位标识FSC指令检测到一个为真的比较。
.LEN	DINT	长度值指定FSC指令操作的数组内元素的数量。
.POS	DINT	位置值包含指令正访问的当前元素的位置。

说明： 当FSC指令被使能，而且比较结果为真时，指令置位发现位(.FD)，位置值(.POS)表明指令发现的比较为真的数组位置。指令置位禁止位(.IN)以防止进一步搜索。

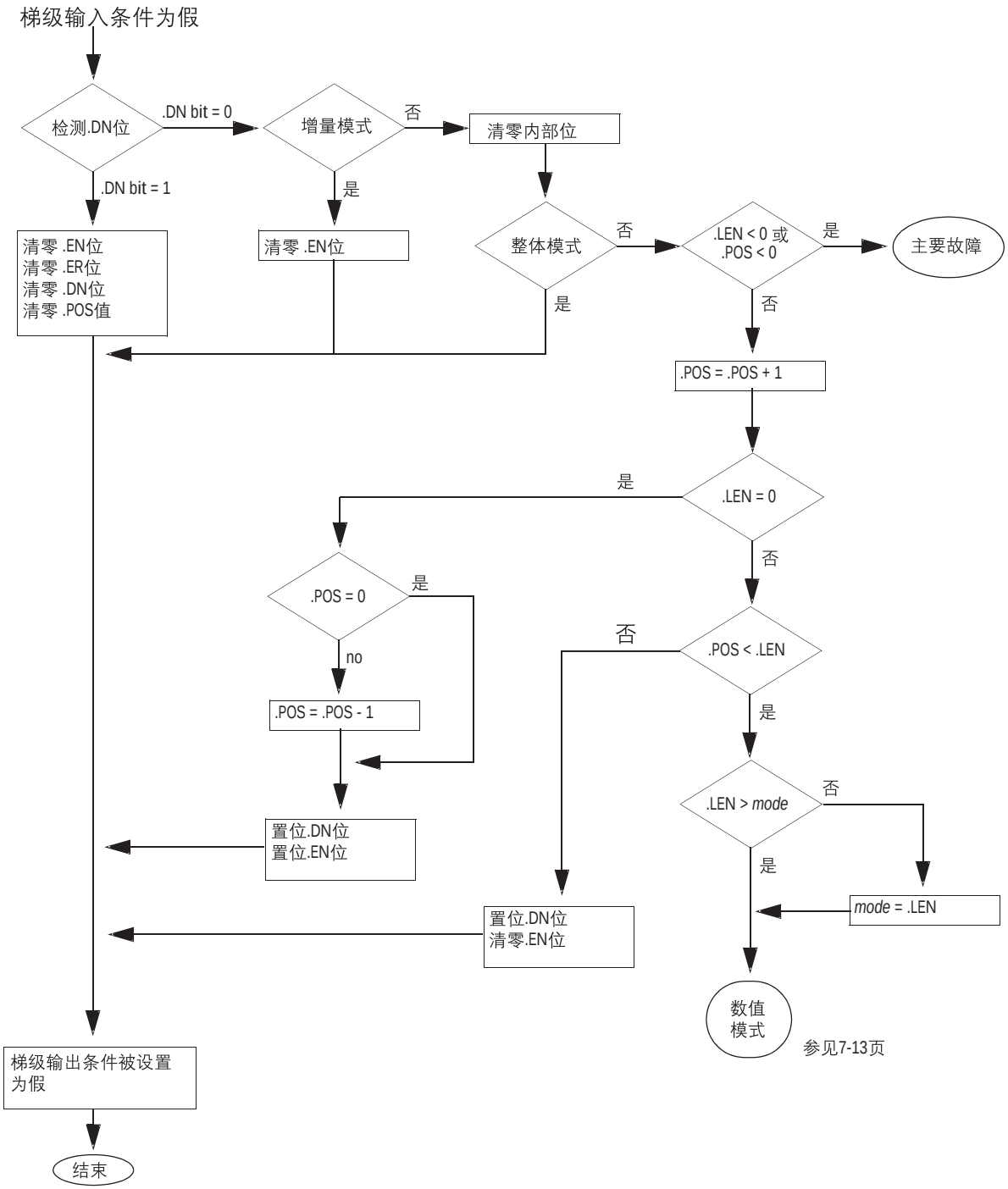
算术状态标志： 影响算术状态标志。

故障条件：

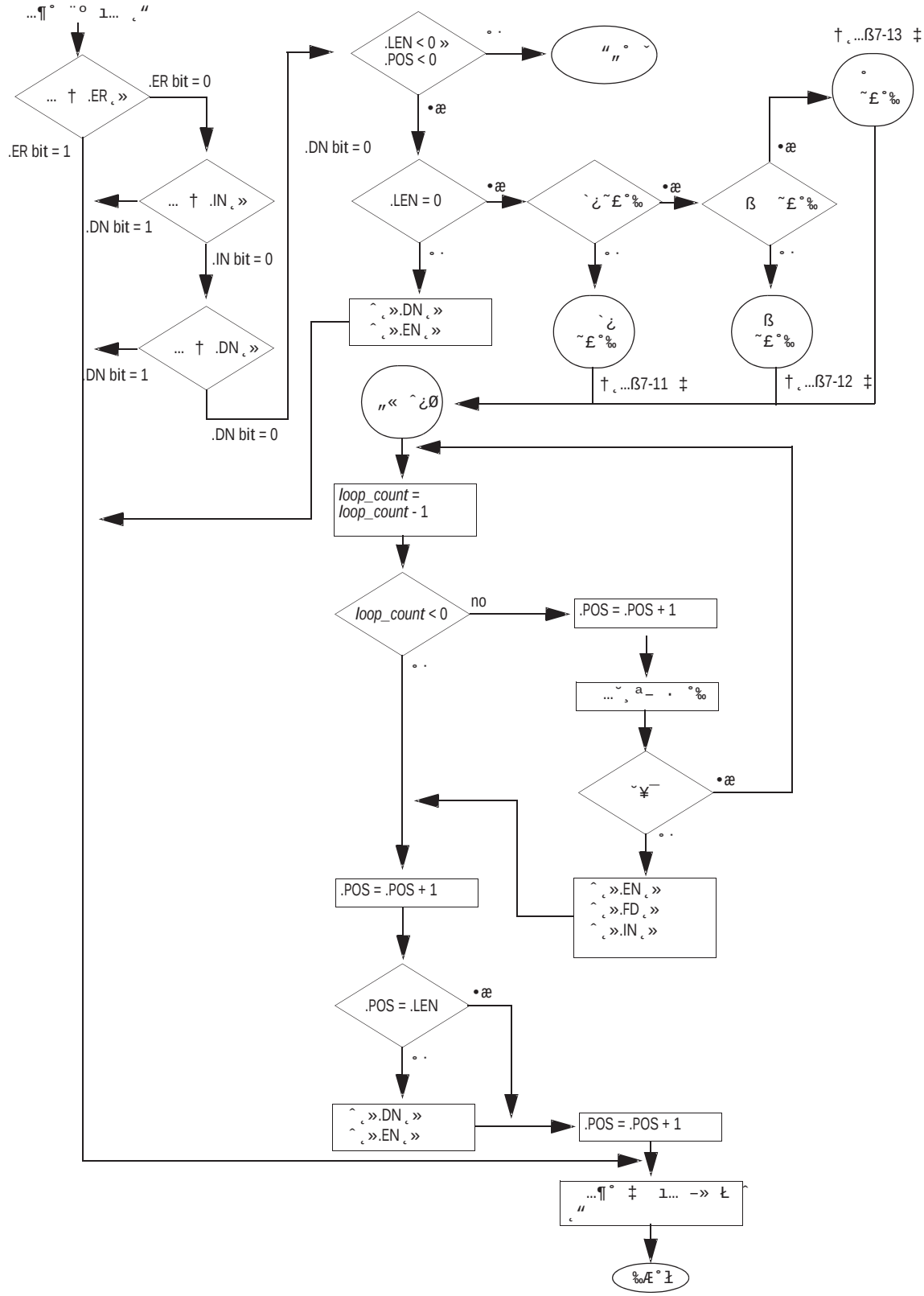
在下列情况将发生主要故障：	故障类型：	故障代码：
.POS < 0 或 .LEN < 0	4	21

执行：

条件：	梯形图动作：
预扫描	梯级输出条件被设置为假。

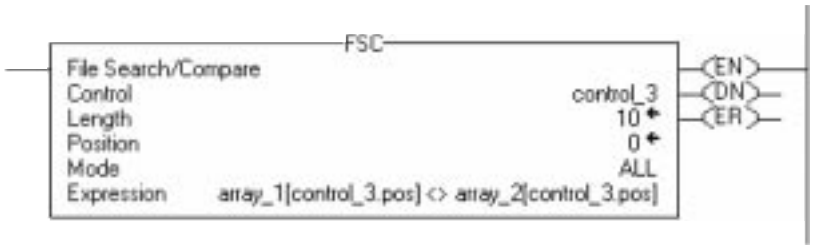


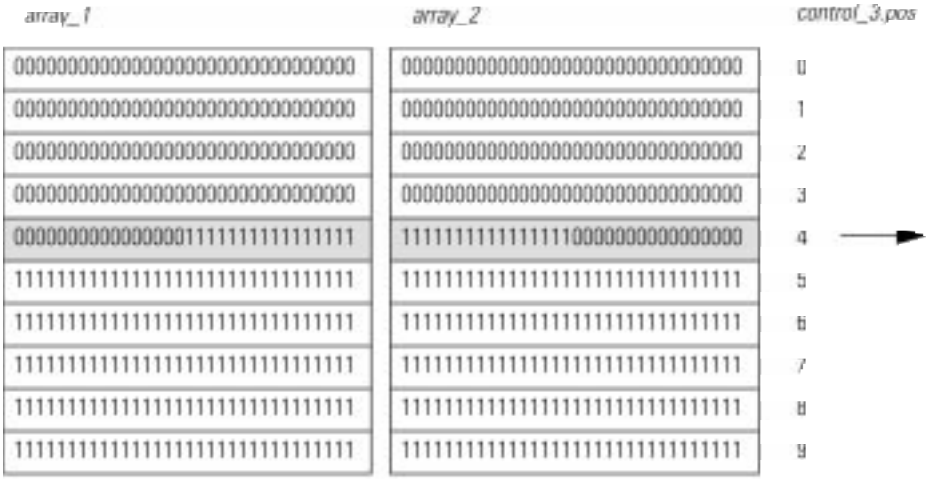
条件： 梯形图动作：



后期扫描 梯级输出条件被设置为假。

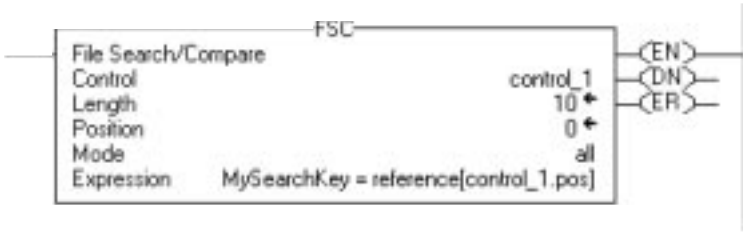
例1: 搜索两个数组内的匹配数据。当指令被使能时，FSC指令比较array_1内的前10个元素与array_2内的相应元素。

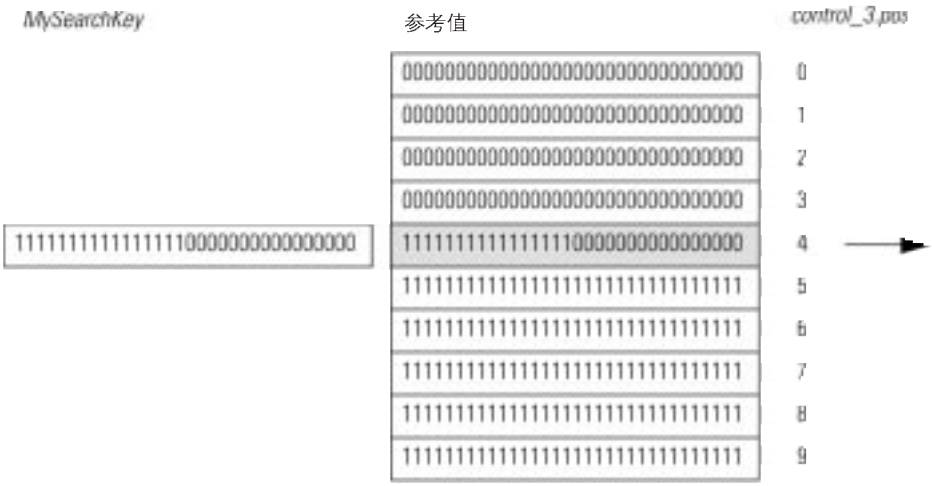




FSC指令发现这些元素不相等。指令置位发现位(.FD)和禁止位(.IN)。位置值(.POS)(4)表明不相等元素的位置。如果要继续比较数组的其余元素，需要清零禁止位(.IN)。

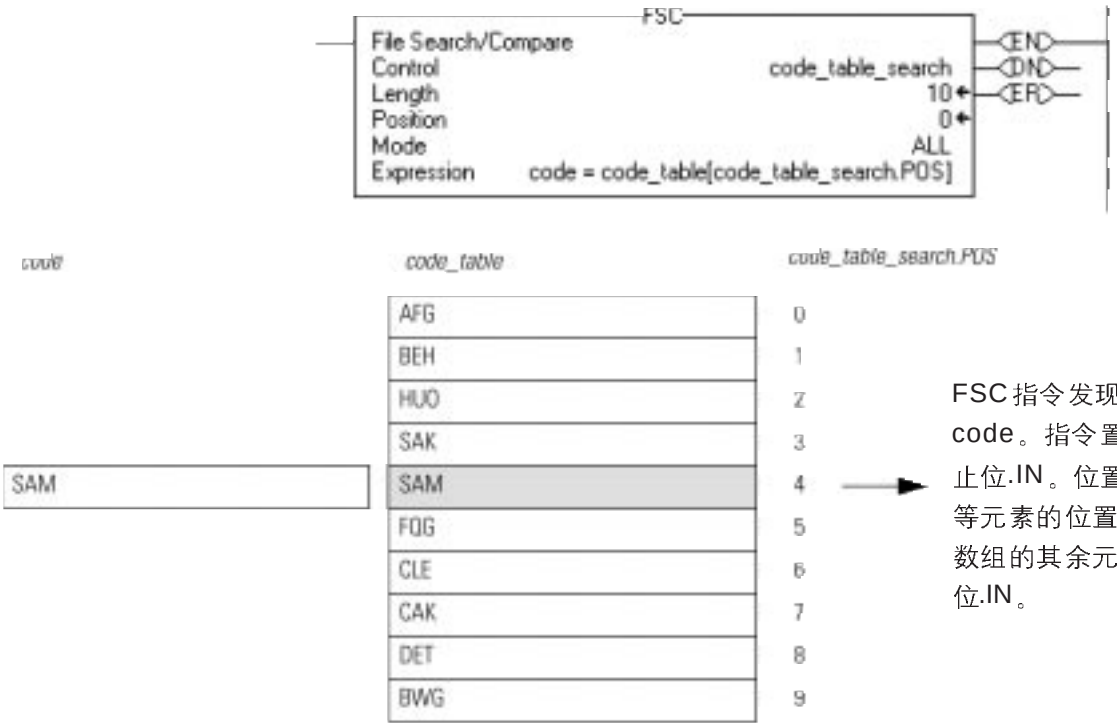
例2：搜索数组内的匹配数据。当指令被使能时，FSC指令使MySearchKey值分别与array_1内的10个元素比较。





FSC指令发现这一数组的元素等于MySearchKey。指令置位发现位(.FD)和禁止位(.IN)。位置值(.POS)(4)表明匹配元素所在的位置。如果要继续比较数组的其余元素，需要清零禁止位(.IN)。

例3：在字符串数组内搜索字符串。当指令被使能时，FSC指令使code内的字符分别与code_table内的10个元素进行比较。



FSC指令发现到该数组元素等于code。指令置位发现位.FD和禁止位.IN。位置值.POS(4)表明相等元素的位置。如果要继续比较数组的其余元素，需要清零禁止位.IN。

FSC表达式

FSC指令内的表达式与CMP指令内的表达式编程方法相同。以下部分是关于有效运算符，格式，以及运算顺序的信息，这部分内容两条指令通用。

有效运算符

运算符：	说明：	最优数据类型：
+	加	DINT，REAL
-	减 / 非	DINT，REAL
*	乘	DINT，REAL
/	除	DINT，REAL
=	等于	DINT，REAL
<	小于	DINT，REAL
<=	小于或等于	DINT，REAL
>	大于	DINT，REAL
>=	大于或等于	DINT，REAL
<>	不等于	DINT，REAL
**	指数(x 的 y次幂)	DINT，REAL
ABS	绝对值	DINT，REAL
ACS	反余弦	REAL
AND	按位与	DINT
ASN	反正弦	REAL
ATN	反正切	REAL
COS	余弦	REAL

运算符：	说明：	最优数据类型：
DEG	弧度转换为角度	DINT，REAL
FRD	BCD码转换成整数	DINT
LN	自然对数	REAL
LOG	以10为底的对数	REAL
MOD	模数-除	DINT，REAL
NOT	按位非	DINT
OR	按位或	DINT
RAD	角度转换成弧度	DINT，REAL
SIN	正弦	REAL
SQR	平方根	DINT，REAL
TAN	正切	REAL
TOD	整数转换成BCD码	DINT
TRN	截短	DINT，REAL
XOR	按位异或	DINT

表达式格式

对于表达式内的每项运算，必须提供一个或两个操作数(标签或者立即数)。按照下表的格式书写表达式内的运算符和操作数：

运算符作用于：	使用下述格式：	举例：
一个操作数	运算符(操作数)	ABS(tag_a)
两个操作数	operand_a 运算符 operand_b	• tag_b + 5 • tag_c AND tag_d • (tag_e ** 2) MOD (tag_f / tag_g)

确定运算顺序

指令按预先规定的顺序，而不是按用户写入的顺序，执行写入表达式的操作。用户可以通过圆括号分组的方法来改变运算顺序，强制指令在执行其它运算之前，执行圆括号内的运算。

同级运算的顺序是从左向右执行。

顺序：	运算：
1	( )
2	ABS，ACS，ASN，ATN，COS，DEG，FRD，LN，LOG，RAD，SIN，SQR，TAN，TOD，TRN
3	**
4	-（取负），NOT
5.	*，/，MOD
6.	<，<=，>，>=，=
7.	-（减），+
8.	AND
9.	XOR
10.	OR

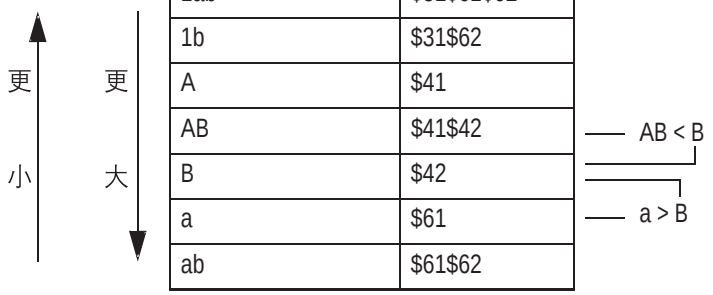
在表达式内使用字符串

要在表达式内使用ASCII字符串，请遵循下述原则：

- 一个表达式允许用户比较两个字符串标签。
- 用户不能在表达式内直接输入ASCII字符。
- 只有下面的运算符允许输入ASCII

运算符：	说明：
=	等于
<	小于
<=	小于或等于
>	大于
>=	大于或等于
<>	不等于

- 如果ASCII码匹配则字符串相等。
- ASCII字符区分大小写。大写 ″A″ ($41)不等于小写 ″a″ ($61)。
- 字符的十六进制值决定一个字符串是否小于或者大于另一个字符串。字符的十六进制代码参见本手册封底。
- 如果两个字符串是在电话号码簿中，则大小由字符串的顺序决定。



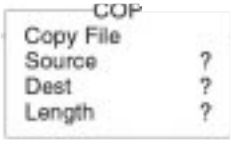
ASCII字符	十六进制代码
1ab	$31$61$62
1b	$31$62
A	$41
AB	$41$42
B	$42
a	$61
ab	$61$62

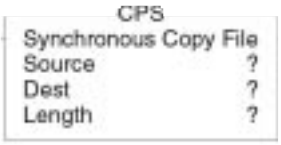
文件复制(COP)
文件同步复制(CPS)

COP和CPS指令复制源值到目的标签。源值保持不变。

操作数：

梯形图





操作数：	数据类型：	格式：	说明：
Source 源值	**SINT** **INT** **DINT** **REAL** 字符串 结构体	标签	要复制的初始元素 **注意**：源和目的操作数必须是相同的数据类型，否则将发生不可预料的结果
Destination 目的标签	**SINT** **INT** **DINT** **REAL** 字符串 结构体	标签	被源操作数覆盖的初始元素 **注意**：源和目的操作数必须是相同的数据类型，否则将发生不可预料的结果
Length 长度	DINT	立即数 标签	被复制到目的标签的元素数量

结构文本

```
COP(Source,Dest,Length);
CPS(Source,Dest,Length);
```

该指令操作数与梯形图COP和CPS指令的操作数相同。

说明：在COP和CPS指令执行期间，控制器的其它操作可能要中断复制操作并改变源或目的标签内的数据：

如果源或目的是：	并且用户要：	则选择：	注释：
• 生产者标签 • 消费者标签 • I/O数据 • 另一任务能够覆盖的数据	防止数据在复制操作期间发生改变	CPS	• 试图中断CPS指令的任务被延迟，直到指令执行结束。 • 要估算CPS指令的执行时间，参见ControlLogix系统用户手册，出版号1756-UM001。
	允许数据在复制操作期间发生改变	COP	
没有上述要求	——————→	COP	

复数值制的字节数量如下所示：

字节数量 = 长度 *（目的标签数据类型的字节数）

注意：如果字节数量大于源操作数的长度，则对于其余元素将复制一些不可预知的数据。

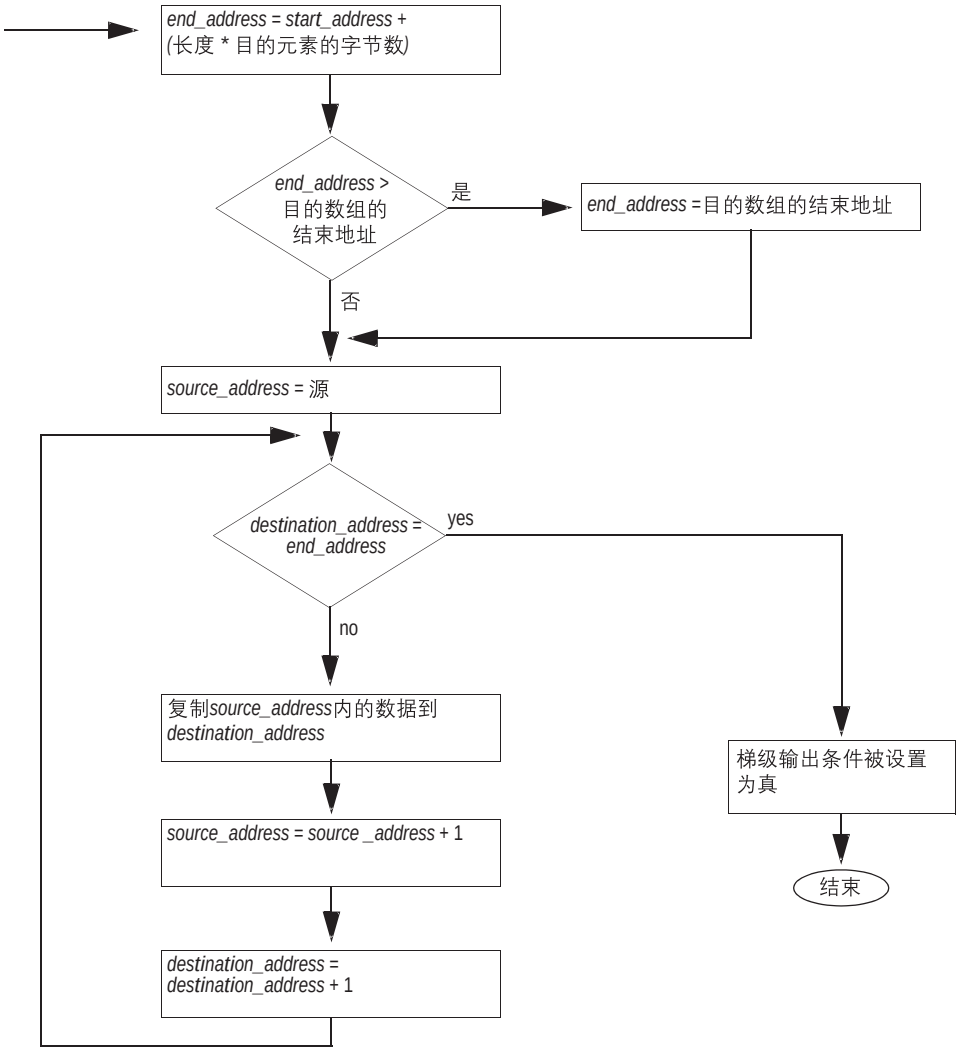
COP 和CPS 指令用于对存储器内的连续数据进行操作，执行存储器内字节到字节的直接复制，这要求理解控制器的存储器结构。

COP 和CPS 指令的写入不超出数组的末尾。如果长度大于目的数组元素的总数，则COP 和CPS 指令在数组的末尾停止操作。而且不发生主要故障。

算术状态标志：不影响

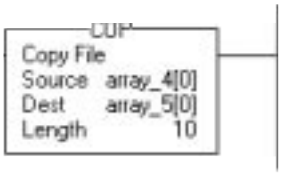
故障条件：无

执行：

条件：	梯形图动作：	结构文本动作：
预扫描	梯级输出条件被设置为假。	无动作发生。
梯级输入条件为假	梯级输出条件被设置为假。	无影响。
梯级输入条件为真	指令执行。 梯级输出条件被设置为真。	无影响。
EnableIn 被置位	无影响。	EnableIn 一直被置位。 指令执行。
指令执行		
后期扫描	梯级输出条件被设置为假。	无动作。

例1：array_4和array_5的数据类型相同。当指令被使能时，COP指令复制array_4内的前10个元素到array_5的前10个元素。

梯形图

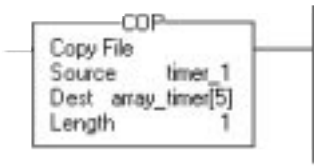


结构文本

```
COP(array_4[0],array_5[0],10);
```

例2：当指令被使能时，COP指令复制结构体timer_1到array_timer的元素5。该指令只复制一个结构体到一数组元素内。

梯形图



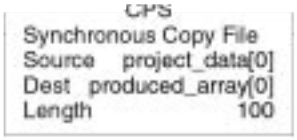
结构体

```
COP(timer_1,array_timer[5],1);
```

例3：project_data数组(100个元素)存储了一组数值，这组数值在应用程序的不同时期发生变化。为了及时将project_data的整个映像传送给另外一个控制器，CPS指令将project_data复制到produced_array。

- 当CPS指令复制数据时，I/O刷新或者其它任务不能改变这些数据。
- produced_array标签在ControlNet网络上生产数据，用于其它控制器的消费。
- 为了使用相同的数据映像(也就是说，数据的同步复制)，消费者控制器使用CPS指令将数据从消费者标签复制到应用程序中使用的另一个标签内。

梯形图



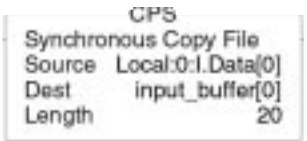
结构文本

```
CPS(project_data[0],produced_array[0],100);
```

例4：Local:0:I.Data存储用于DeviceNet网络的输入数据，该DeviceNet网络连接到0号槽内的1756-DNB模块。为了使输入数据和应用程序同步，CPS指令将输入数据复制到input_buffer。

- 当CPS指令复制数据时，I/O刷新不能改变这些数据。
- 当应用程序执行时，使用input_buffer内的输入数据。

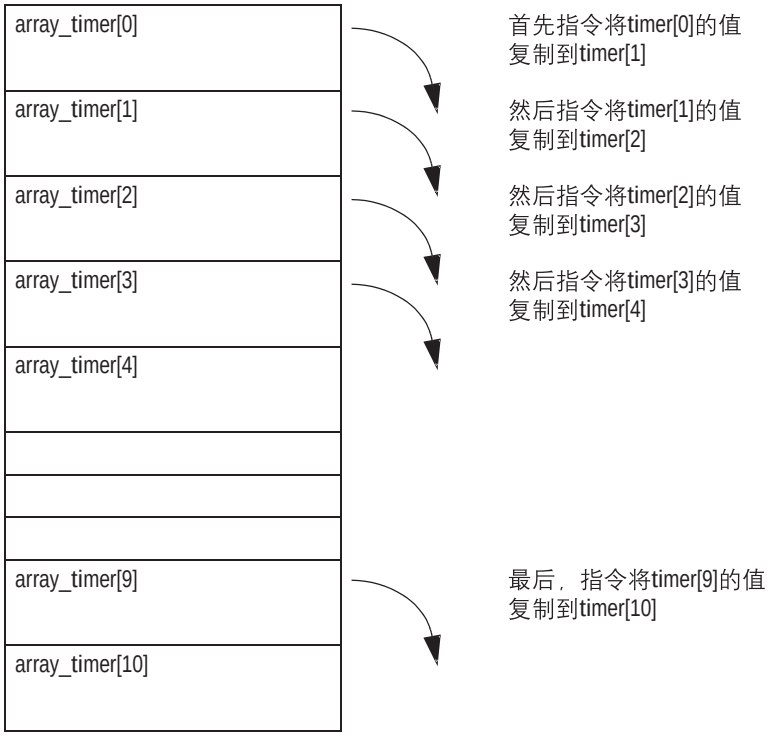
梯形图



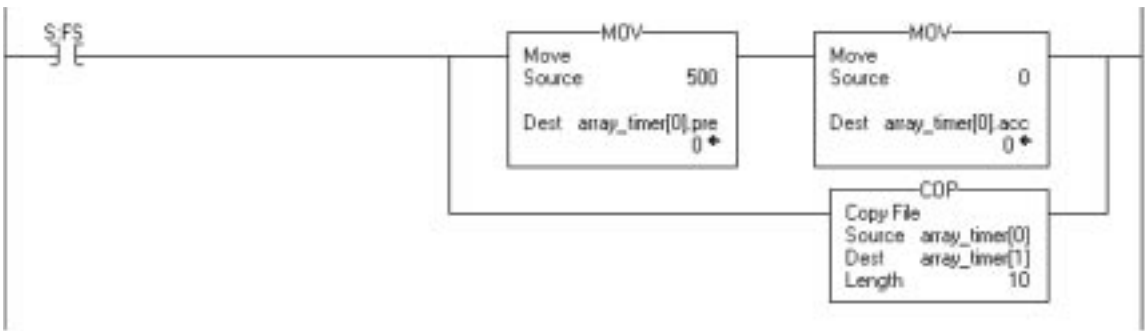
结构文本

```
CPS(Local:0:I.Data[0],input_buffer[0],20);
```

例5：本例初始化一个计时器结构体数组。当指令被使能时，MOV指令初始化array_timer第一个元素的.PRE和.ACC值。当指令被使能时，COP指令复制连续的起始于array_timer[0]的字节块。长度为9个计时器结构体。



梯形图



结构文本

```
IF S:FS THEN
        array_timer[0].pre := 500;
        array_timer[0].acc :=0;
        COP(array_timer[0],array_timer[1],10);
END_IF;
```

文件填充(FLL)

FLL指令用源值填充一个数组内的元素。源值保持不变。

操作数：

梯形图

操作数：	数据类型：	格式：	说明：
Source 源值	**SINT** **INT** **DINT** **REAL**	立即数 标签	要复制的元素 注意：源和目的操作数必须是相同的数据类型，否则将发生不可预料的结果
Destination 目的标签	**SINT** **INT** **DINT** **REAL** 结构体	标签	被源操作数值覆盖的初始元素 注意：源和目的操作数必须是相同的数据类型，否则将发生不可预料的结果 初始化结构体的首选方法是使用COP指令
Length 长度	DINT	立即数	填充到目的标签的元素数量

结构文本

结构文本中无FLL指令，但用户可使用SIZE指令和FOR…DO或其它循环结构体完成相同功能。

```
SIZE(destination,0,length);
FOR position = 0 TO length-1 DO
        destination[position] := source;
END_FOR;
```

关于结构文本中结构体的语法信息，参见附录C。

说明： 填充字节的数量是：

字节数 = 长度 *（目的标签数据类型的字节数）

FLL指令对存储器内的连续数据进行操作。

FLL指令的写入范围不超出数组的末尾。如果长度大于目的数组元素的总数，则FLL指令在数组的末尾停止操作。而且不发生主要故障。

要获得最好结果，源值和目的标签应该用相同的数据类型。如果用户要填充一个结构体，可以使用COP指令(参见7-32页的例3)。如果源值和目的标混用数据类型，则目的标签元素用转换后的源值填充。

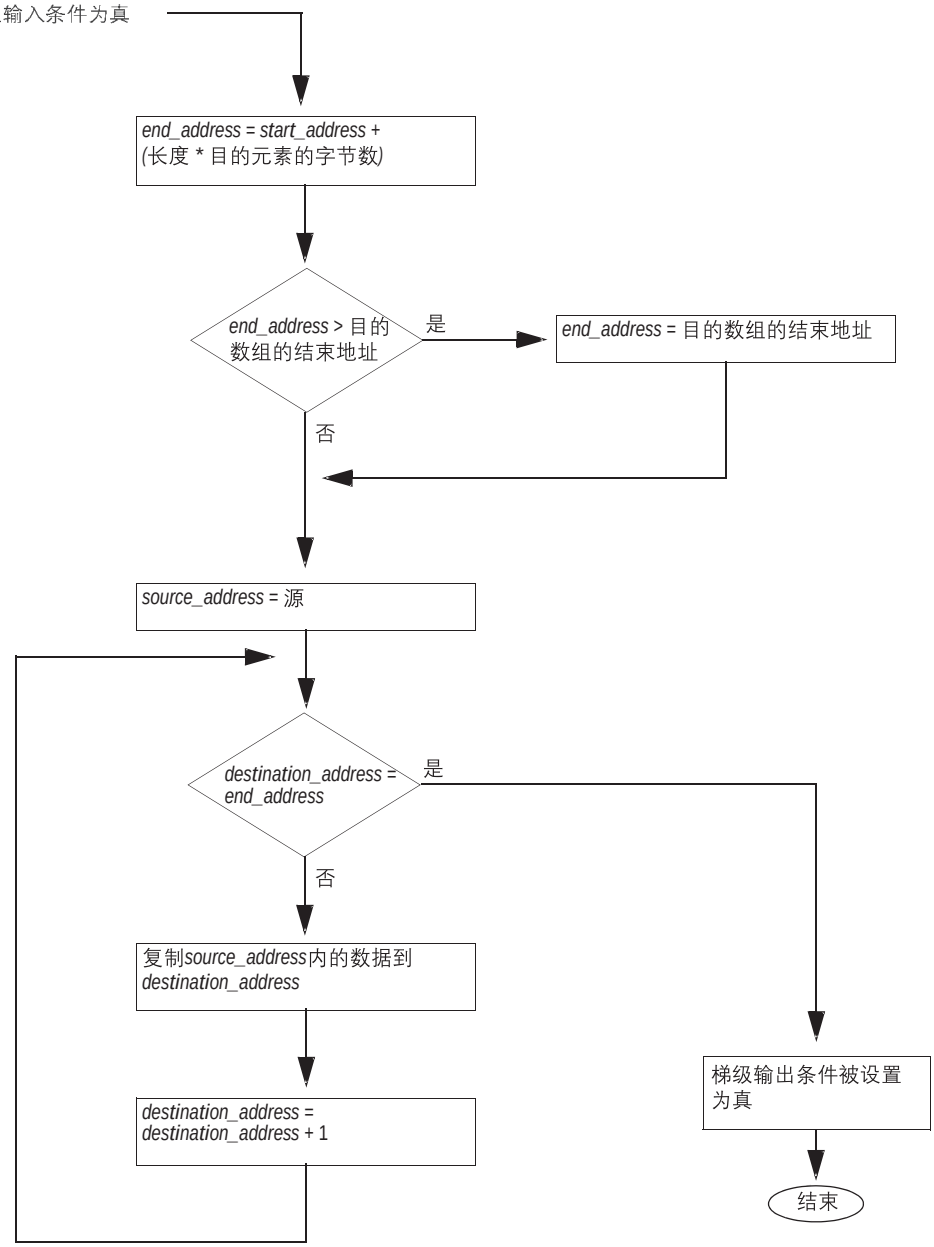
如果源值数据类型是：	目的标签数据类型是：	源值被转换为：
SINT，INT，DINT，或REAL	SINT	SINT
SINT，INT，DINT，或REAL	INT	INT
SINT，INT，DINT，或REAL	DINT	DINT
SINT，INT，DINT，或REAL	REAL	REAL
SINT	结构体	SINT(不转换)
INT	结构体	INT(不转换)
DINT	结构体	DINT(不转换)
REAL	结构体	REAL(不转换)

算术状态标志：不影响

故障条件：无

执行：

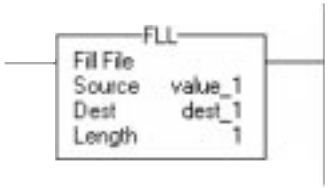
条件：	梯形图动作：
预扫描	梯级输出条件被设置为假。
梯级输入条件为假	梯级输出条件被设置为假。
梯级输入条件为真	



后期扫描	梯级输出条件被设置为假。

举例：FLL指令将value_1内的值复制到dest_1。

梯形图



源(value_1)数据类型：	源(value_1)值：	目的(dest_1)数据类型：	FLL指令执行之后的目的(dest_1)值：
SINT	16#80 (-128)	DINT	16#FFFF FF80 (-128)
DINT	16#1234 5678	SINT	16#78
SINT	16#01	REAL	1
REAL	2	INT	16#0002
SINT	16#01	TIMER	16#0101 0101 16#0101 0101 16#0101 0101
INT	16#0001	TIMER	16#0001 0001 16#0001 0001 16#0001 0001
DINT	16#0000 0001	TIMER	16#0000 0001 16#0000 0001 16#0000 0001

结构文本

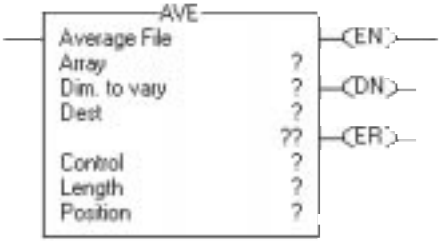
```
dest_1 := value_1;
```

文件平均值(AVE)

AVE指令计算一组数值的平均值。

操作数：

梯形图



操作数：	数据类型：	格式：	说明：
Array 数组	SINT INT DINT **REAL**	数组标签	计算该数组内数值的平均值 指定要计算平均值的元素组的第一个元素 不能在此下标处用CONTROL.POS
Dimension to vary 变换的维数	DINT	立即数 (0，1，2)	维数的使用与维数号有关，顺序是 array[dim_0,dim_1,dim_2] array[dim_0,dim_1] array[dim_0]
Destination 目的标签	SINT INT DINT **REAL**	标签	指令运算的结果
Control 控制	CONTROL	标签	指令运算的控制结构体
Length 长度	DINT	立即数	求平均值的数组元素的数量
Position 位置	DINT	立即数	数组内当前元素的位置，典型的初始值为0

结构体

结构体文本中不存在AVE指令，但用户可使用SIZE指令和FOR…DO或其它循环指令完成相同功能。

```
SIZE(array,0,length);
sum := 0;
FOR position = 0 TO length-1 DO
        sum := sum + array[position];
END_FOR;
destination := sum / length;
```

关于结构文本中结构体的语法信息，参见附录C。

CONTROL结构体

助记符：	数据类型：	说明：
.EN	BOOL	使能位标识AVE指令被使能。
.DN	BOOL	当指令已经处理完数组中最后一个元素时(.POS = .LEN)，完成位被置位。
.ER	BOOL	如果指令执行时发生溢出，则该位被置位。在程序清零错误位(.ER)之前停止执行指令。引起溢出的元素的位置存储在位置值(.POS)内。
.LEN	DINT	长度值指定指令处理的数组中元素的数量。
.POS	DINT	位置值包含指令正访问的当前元素的位置。

说明： AVE指令计算一组数值的平均值。

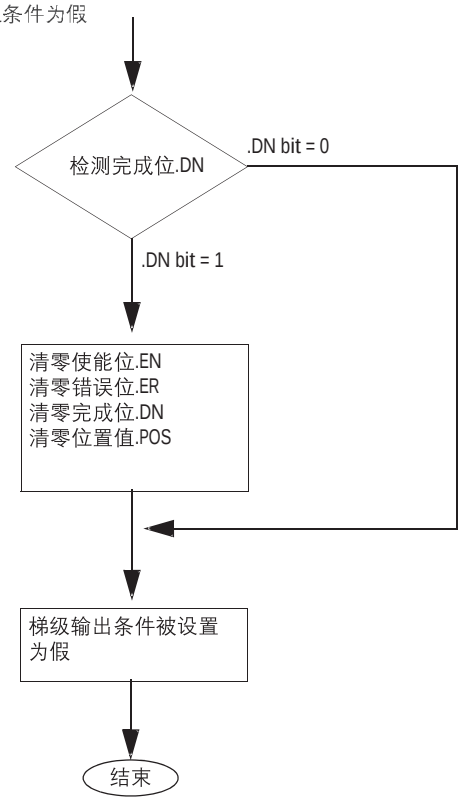
重要事项 编程时要确定长度值不能使指令超出指定的变换维数。如果发生这种情况，则目的值不正确。

算术状态标志： 影响算术状态标志。

故障条件：

在下列情况将发生主要故障：	故障类型：	故障代码：
.POS < 0 或 .LEN < 0	4	21
指定数组不存在的变换维数	4	20

执行：

条件：	梯形图动作：
预扫描	清零使能位.EN。 清零完成位.DN。 清零错误位.ER。 梯级输出条件被设置为假。
梯级输入条件为假	
梯级输入条件为真	AVE指令通过使数组内的所有指定元素相加，再被元素数量除来求平均值。 实际上，该指令用FAL指令计算平均值： 表达式 = 平均值计算 模式 = 整体模式 关于FAL指令如何执行，参见7-9页。
后期扫描	梯级输出条件被设置为假。

例1：计算array_dint平均值，其中array_dint为DINT[4,5]。

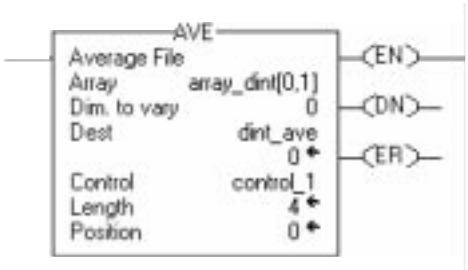
维数1

维数0	0	1	2	3	4
0	20	19	18	17	16
1	15	14	13	12	11
2	10	9	8	7	6
3	5	4	3	2	1

$$AVE = \frac{19+14+9+4}{4} = \frac{46}{4} = 11.5$$

dint_ave = 12

梯形图



结构文本

```
SIZE(array_dint,0,length);
sum := 0;
FOR position = 0 TO (length-1) DO
    sum := sum + array_dint[position];
END_FOR;
dint_ave := sum / length;
```

例2：计算array_dint平均值，array_dint是DINT[4,5]。

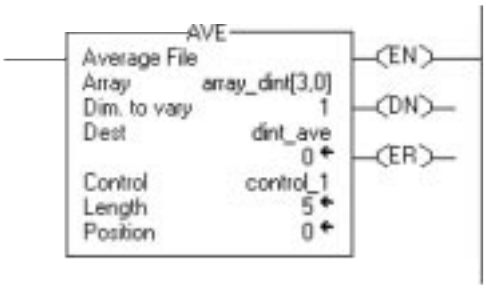
维数1

维数0 / 索引	0	1	2	3	4
0	20	19	18	17	16
1	15	14	13	12	11
2	10	9	8	7	6
3	5	4	3	2	1

$$AVE = \frac{5+4+3+2+1}{5} = \frac{15}{5} = 3$$

dint_ave = 3

梯形图



结构文本

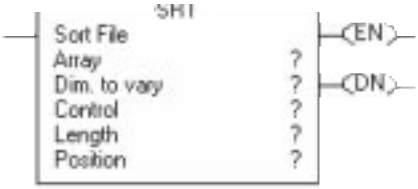
```
SIZE(array_dint,1,length);
sum := 0;
FOR position = 0 TO (length-1) DO
        sum := sum + array_dint[position];
END_FOR;
dint_ave := sum / length;
```

文件排序(SRT)

SRT指令以升序对数组内的一维数组(变换的维数)进行排序。

操作数：

梯形图



操作数：	数据类型：	格式：	说明：
Array 数组	**SINT** **INT** **DINT** **REAL**	数组标签	要排序的数组 指定要排序元素组的第一个元素 不能在此下标处用CONTROL.POS
Dimension to vary 变换的维数	DINT	立即数 (0，1，2)	维数的使用与维数号有关，顺序是 array[dim_0,dim_1,dim_2] array[dim_0,dim_1] array[dim_0]
Control 控制	CONTROL	标签	指令运算的控制结构体
Length 长度	DINT	立即数	排序的数组元素的数量
Position 位置	DINT	立即数	数组内当前元素的位置 典型的初始值为0

结构文本

```
SRT(Array,DimtoVary,
    Control);
```

该指令操作数与梯形图SRT指令操作数相同。然而，用户通过访问CONTROL结构体中长度(.LEN)和位置(.POS)项来指定长度和位置值，而不是通过包含在操作数列表中的值。

CONTROL结构体

助记符：	数据类型：	说明：
.EN	BOOL	使能位标识SRT指令被使能。
.DN	BOOL	当指定的元素已经被排序后，置位完成位。
.ER	BOOL	如果.LEN < 0 或.POS < 0，则置位该位。这两个故障条件的任何一个都会发生主要故障。
.LEN	DINT	长度值表明指令处理的数组内元素的数量。
.POS	DINT	位置值包含指令正访问的当前元素的位置。

说明：SRT指令以升序对数组内的一维数组(变换的维数)进行排序。

重要事项 编程时要确定长度值不能使指令超出指定的变换维数。如果发生此事件，则会发生不可预料的结果。

该指令由边沿触发：

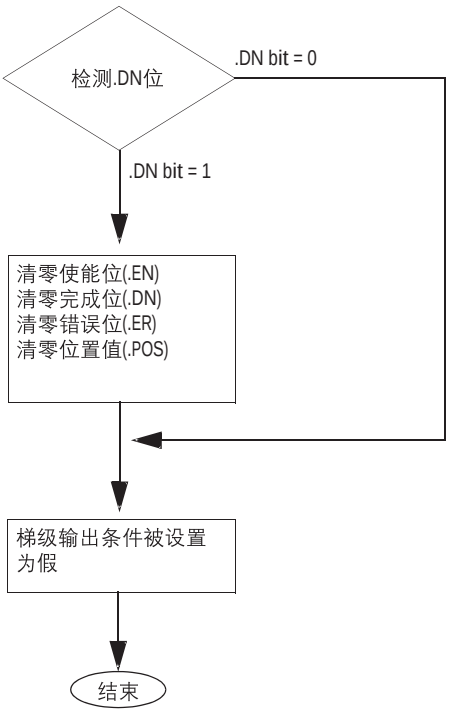
- 用梯形图语言编程，每次梯级输入条件由清零到置位被触发时，该指令都应该执行。
- 用结构文本编程，只有条件为边沿触发时指令才执行。参见附录C。

算术状态标志：影响算术状态标志。

故障条件：

在下列情况将发生主要故障：	故障类型：	故障代码：
.POS < 0 或 .LEN < 0	4	21
指定数组不存在的变换维数	4	20
指令试图访问超出数组边界的数据	4	20

执行：

条件：	梯形图动作：	结构文本动作：
预扫描	清零使能位.EN。 清零完成位.DN。 清零错误位.ER。 梯级输出条件被设置为假。	清零使能位.EN。 清零完成位.DN。 清零错误位.ER。
梯级输入条件为假		不影响。
梯级输入条件为真	指令执行。 梯级输出条件被设置为真。	不影响
EnableIn被置位	不影响	EnagleIn一直被置位。 指令执行。
指令执行	指令以升序对数组的指定元素进行排序。	指令以升序对数组的指定元素进行排序。
后期扫描	梯级输出条件被设置为假。	无动作。

例1：对int _ array排序，即DINT[4,5]。

指令执行之前

维数1

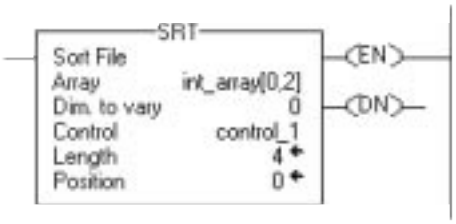
维数0 \ 维数1	0	1	2	3	4
0	20	19	18	17	16
1	15	14	13	12	11
2	10	9	8	7	6
3	5	4	3	2	1

指令执行之后

维数1

维数0 \ 维数1	0	1	2	3	4
0	20	19	3	17	16
1	15	14	8	12	11
2	10	9	13	7	6
3	5	4	18	2	1

梯形图



结构文本

```
control_1.LEN := 4;

control_1.POS := 0;

SRT(int_array[0,2],0,control_1);
```

例2：对int _ array排序，即DINT[4,5]。

指令执行之前

维数1

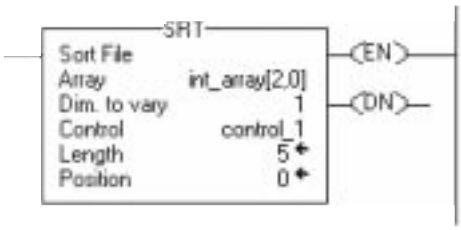
维数0	0	1	2	3	4
0	20	19	18	17	16
1	15	14	13	12	11
2	10	9	8	7	6
3	5	4	3	2	1

指令执行之后

维数1

维数0	0	1	2	3	4
0	20	19	18	17	16
1	15	14	13	12	11
2	6	7	8	9	10
3	5	4	3	2	1

梯形图



结构文本

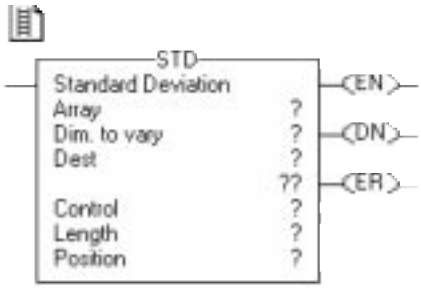
```
control_1.LEN := 5;

control_1.POS := 0;

SRT(int_array[2,0],1,control_1);
```

文件标准偏差(STD)

STD指令计算数组中一维数组内一组值的标准偏差，并将结果存储于目的标签。

操作数：

梯形图



操作数：	数据类型：	格式：	说明：
Array 数组	SINT INT **DINT** **REAL** SINT或INT标签通过符号-扩充转换为DINT值。	数组标签	计算数组内数值的标准偏差 指定用于计算标准偏差的元素组的第一个元素 不能在此下标处用CONTROL.POS
Dimension to vary 变换的维数	DINT	立即数 (0，1，2)	维数的使用与维数号有关，顺序是 array[dim_0,dim_1,dim_2] array[dim_0,dim_1] array[dim_0]
Destination 目的标签	REAL	标签	指令运算的结果
Control 控制	CONTROL	标签	指令运算的控制结构体
Length 长度	DINT	立即数	用于计算标准偏差的数组元素的数量
Position 位置	DINT	立即数	数组内当前元素的位置 典型的初始值为0

CONTROL结构体

助记符：	数据类型：	说明：
.EN	BOOL	使能位标识STD指令被使能。
.DN	BOOL	当完成计算时，置位完成位。
.ER	BOOL	如果指令执行时发生溢出，则错误位被置位。在程序清零错误位.ER之前停止执行指令。引起溢出的元素的位置存储在位置值.POS内。
.LEN	DINT	长度值指定指令处理的数组内元素的数量。
.POS	DINT	位置值包含指令正访问的当前元素的位置。

结构文本

结构文本中无STD指令，但用户可使用SIZE指令和FOR…DO或其它循环结构体完成相同功能。

```
SIZE(array,0,length);
sum := 0;
FOR position = 0 TO length-1 DO
        sum := sum + array[position];
END_FOR;
average := sum / length;
sum := 0;
FOR position = 0 TO length-1 DO
        sum := sum + ((array[position] - average)**2);
END_FOR;
destination := SQRT(sum /(length-1));
```

说明：标准偏差按照下列公式进行计算：

$$\text{标准偏差} = \sqrt{\frac{\left(\sum_{i=1}^{N}\left[\left(X_{(start+i)}-AVE\right)^2\right]\right)}{(N-1)}}$$

其中：

- start = 数组操作数的变换维数下标
- xi = 数组内的变量元素
- N = 数组内指定元素的数量
-

$$\frac{\left(\sum_{i=1}^{N} x_{(start+i)}\right)}{N}$$

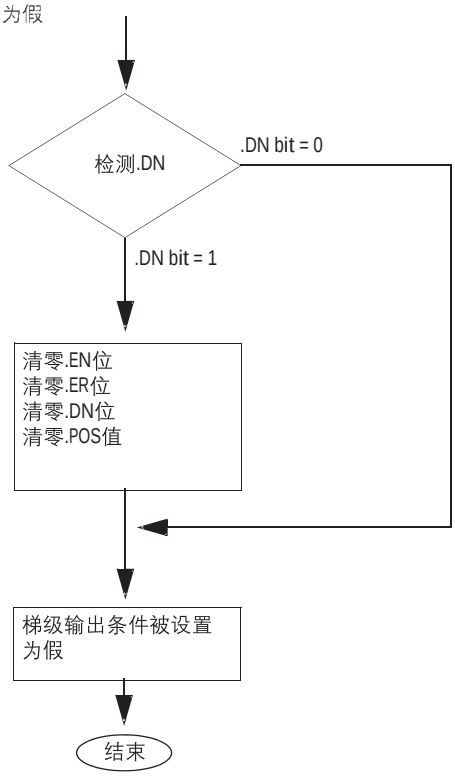
重要事项 编程时要确定长度值，不能使指令超出指定的变换维数。如果该情况发生，则目的标签值不正确。

算术状态标志：影响算术状态标志。

故障条件：

在下列情况将发生主要故障：	故障类型：	故障代码：
.POS < 0 或 .LEN < 0	4	21
指定数组不存在的变换维数	4	20

执行：

条件：	梯形图动作：
预扫描	清零使能位.EN。 清零完成位.DN。 清零错误位.ER。 梯级输出条件被设置为假。
梯级输入条件为假	
梯级输入条件为真	STD指令计算指定元素的标准偏差。实际上，该指令使用FAL指令计算平均值： 表达式 = 标准偏差计算 模式 =整体模式 关于FAL指令如何执行，参见7-9页。
后期扫描	梯级输出条件被设置为假。

例1：计算dint_array的标准偏差，其中dint_array为DINT[4,5]。

$$AVE = \frac{16+11+6+1}{4} = \frac{34}{4} = 8.5$$

维数0

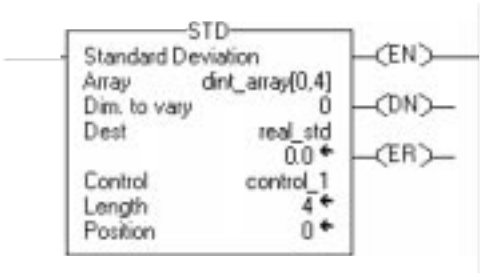
维数1

下标	0	1	2	3	4
0	20	19	18	17	16
1	15	14	13	12	11
2	10	9	8	7	6
3	5	4	3	2	1

$$STD = \sqrt{\frac{(16-8.5)^2+(11-8.5)^2+(6-8.5)^2+(1-8.5)^2}{(4-1)}} = 6.454972$$

real_std = 6.454972

梯形图



结构文本

```
SIZE(dint_array,0,length);
sum := 0;
FOR position = 0 TO (length-1) DO
        sum := sum + dint_array[position];
END_FOR;
average := sum / length;
sum := 0;
FOR position = 0 TO (length-1) DO
        sum := sum +((dint_array[position]-average)**2);
END_FOR;
real_std := SQRT(sum / (length-1));
```

例2：计算dint_array的标准偏差，其中dint_array为DINT[4,5]。

维数1

个数	0	1	2	3	4
0	20	19	18	17	16
1	15	14	13	12	11
2	10	9	8	7	6
3	5	4	3	2	1

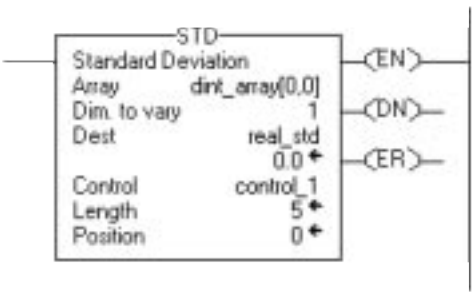
维数0

$$AVE = \frac{20+19+18+17+16}{5} = \frac{90}{5} = 18$$

$$STD = \sqrt{\frac{\langle 20-18\rangle^2 + \langle 19-18\rangle^2 + \langle 18-18\rangle^2 + \langle 17-18\rangle^2 + \langle 16-18\rangle^2}{\langle 5-1\rangle}} = 1.581139$$

$real\_std = 1.581139$

梯形图



结构文本

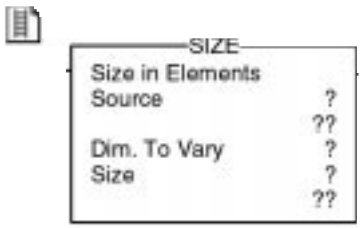
```
SIZE(dint_array,1,length);
sum := 0;
FOR position = 0 TO (length-1) DO
        sum := sum + dint_array[position];
END_FOR;
average := sum / length;
sum := 0;
FOR position = 0 TO (length-1) DO
        sum := sum +((dint_array[position]-average)**2);
END_FOR;
real_std := SQRT(sum / (length-1));
```

元素尺寸(SIZE)

SIZE指令用于查找数组中一维元素的数量。

操作数：

梯形图



操作数：	数据类型：	格式：	说明：
Source 源值	SINT INT DINT REAL 结构体 字符串	数组标签	指令要进行操作的数组
Dimension to Vary 变换的维数	DINT	立即数 (0，1，2)	使用的维数： 对于：　输入： 第一维　0 第二维　1 第三维　2
Size 大小	SINT INT DINT REAL	标签	用于存储数组中指定维数内元素数量的标签

结构文本

```
SIZE(Source,Dimtovary,Size);
```

该指令操作数与梯形图中SIZE指令的操作数相同。

说明： SIZE指令查找源数组的指定维数内的元素数量(大小)，并将结果放置在Size操作数内。

- 指令查找数组的一维的大小。
- 指令作用于：
  —数组
  —结构体中数组
  —更大型数组内的一部分

算术状态标志： 不影响

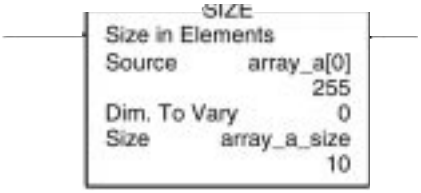
故障条件： 无

执行：

条件：	梯形图动作：	结构文本动作：
预扫描	梯级输出条件被设置为假。	无动作。
梯级输入条件为假	梯级输出条件被设置为假。	无影响
梯级输入条件为真	指令执行。 梯级输出条件被设置为真。	无影响
EnableIn被置位	无影响	EnableIn一直被置位。 指令执行。
指令执行	指令查找维数大小。	指令查找维数大小。
后期扫描	梯级输出条件被设置为假。	无动作。

例1：查找array_a 的0维(第一维)内的元素数量，将结果存储在array_a_size内。本例中，array_a 的0维有10个元素。

梯形图

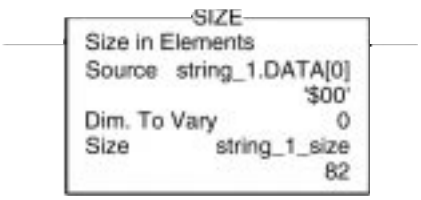


结构文本

```
SIZE(array_a,0,array_a_size);
```

例2：查找字符串string_1的数据项内的元素数量，将结果存储在string_1_size内。本例中，string_1的数据项有82个元素。(字符串使用缺省字符串数据类型。)由于每个元素保存一个字符，所以string_1最多可包含82个字符。

梯形图

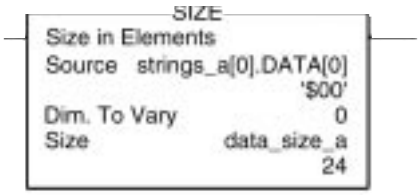


结构文本

```
SIZE(string_1.DATA[0],0,string_1_size);
```

例3：Strings_a是字符串结构体的一个数组。SIZE指令查找该字符串结构体的数据项内的元素数量，并将结果存储在data_size_a内。本例中，数据项有24个元素。（该字符串结构体具有用户指定的长度24。）

梯形图



结构文本

```
SIZE(strings_a[0].DATA[0],0,data_size_a);
```

注释：

数组 (文件)/移位指令 (BSL， BSR， FFL， FFU， LFL， LFU)

简介

数组(文件)/移位指令用来改变数组内数据的位置。

如果用户需要：	所用指令：	可用语言：	参见页码：
完成数组中数据位装载、移动或卸载操作	BSL	梯形图	8-2
	BSR	梯形图	8-5
按相同顺序装载和卸载数值	FFL	梯形图	8-8
	FFU	梯形图	8-14
按相反的顺序装载和卸载数值	LFL	梯形图	8-20
	LFU	梯形图	8-26

可以混合使用数据类型，但是会损失精度或发生取整误差。

对于梯形图语言编程，则黑体字数据类型是最优数据类型。如果指令的全部操作数都用同一优化数据类型，则指令的执行速度快且占用内存少。典型最优数据类型是DINT或REAL数据类型。

位左移指令 (BSL)

BSL指令将数组中的指定位向左移动一个位置。

操作数：

梯形图

操作数：	数据类型：	输入格式：	说明：
Array 数组	DINT	数组标签	需要修改的数组 指定元素组的第一个元素 不能在下标处使用CONTROL.POS
Control 控制	CONTROL	标签	指令操作的控制结构体
Source bit 源位	BOOL	标签	要移入的位
Length 长度	DINT	立即数	数组中需要移动的位数

CONTROL结构体

助记符：	数据类型：	说明：
.EN	BOOL	使能位表明BSL指令被使能。
.DN	BOOL	完成位置位表示数组中的位已经向左移动了一个位置。
.UL	BOOL	卸载位是指令的输出。卸载位.UL存储移出数组位的状态。
.ER	BOOL	当长度值.LEN < 0.时置位错误位。
.LEN	DINT	长度指定数组中被移动位的数量。

说明： 当指令被使能时，BSL指令将指定位的最高位卸载到.UL位中，其余的位顺序向左移动一个位置，并将源位装载到数组中的位0内。

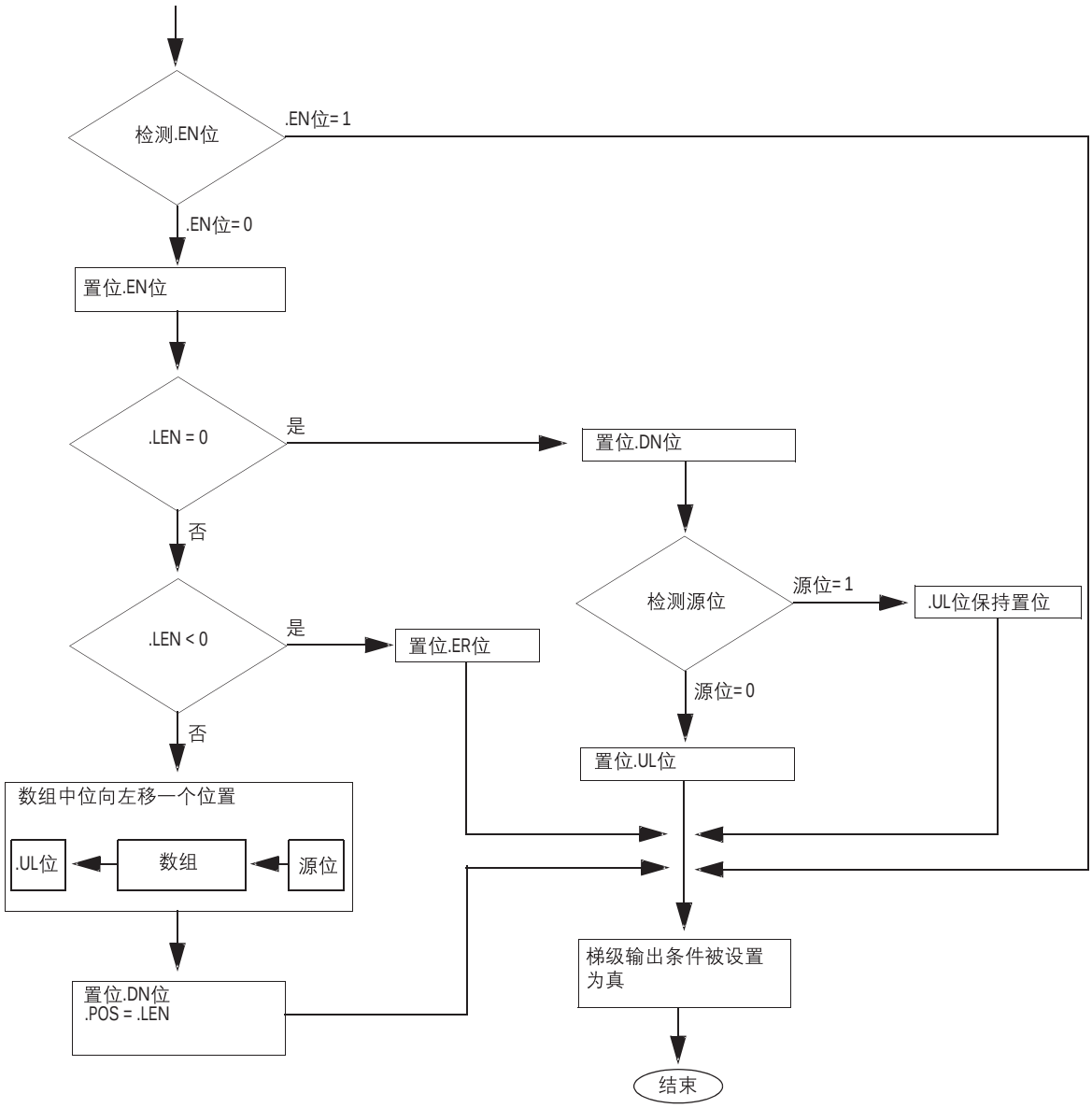
BSL指令对内存中的一个连续区域进行操作。

算术状态标志：不影响

故障条件：无

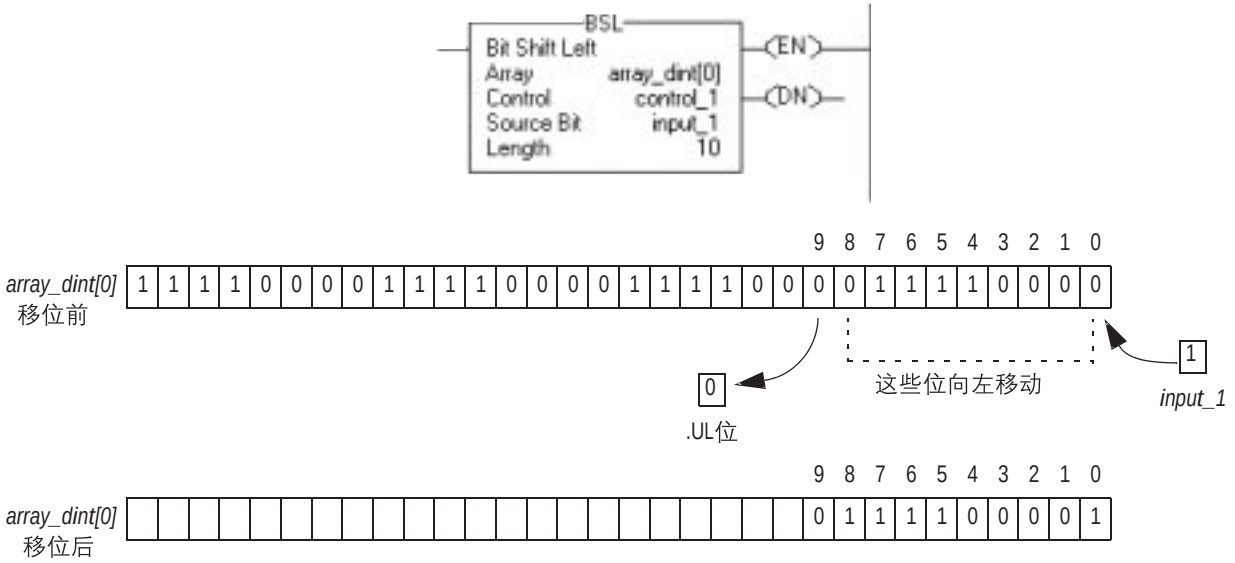
执行:

条件:	梯形图动作:
预扫描	清零使能位.EN。 清零完成位.DN。 清零错误位.ER。 清零位置值.POS。 梯级输出条件被设置为假。
梯级输入条件为假	清零使能位.EN。 清零完成位.DN。 清零错误位.ER。 清零位置值.POS。 梯级输出条件被设置为假。
梯级输入条件为真	

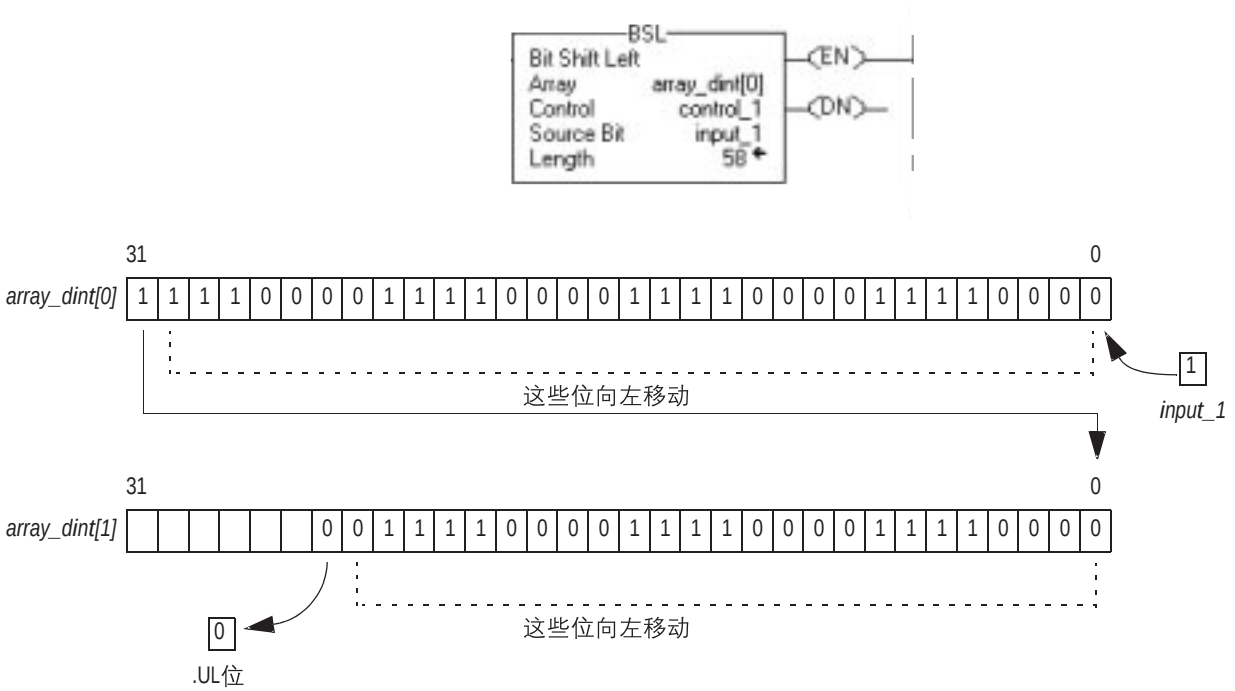


后期扫描	梯级输出条件被设置为假。

例1：当指令被使能时，BSL指令从array_dint[0] 中的位0开始执行。指令将array_dint[0].9卸载到.UL 位中，移动其余的位并将input_1 装载到array_dint[0].0中。其余位的值 (10-31)无效。



例2：当指令被使能时，BSL指令从array_dint[0] 中的位0开始执行。指令将array_dint[1].25卸载到.UL 位中，移动其余的位并将input_1 装载到array_dint[0].0 中。其余位的值 (array_dint[1]中的31-26)无效。注意array_dint[0].31如何移动到array_dint[1]. 0。

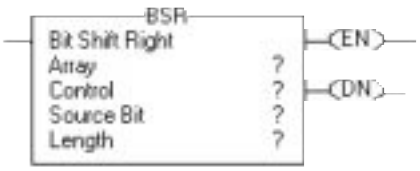


位右移指令 (BSR)

BSR指令将数组中的指定位向右移动一个位置。

操作数：

梯形图



操作数：	数据类型：	输入格式：	说明：
Array 数组	DINT	数组标签	需要修改的数组 指定移动元素的起始位 不能在下标处用CONTROL.POS
Control 控制	CONTROL	标签	指令操作的控制结构体
Source bit 源位	BOOL	标签	要移入的位
Length 长度	DINT	立即数	数组中需要移动的位数

CONTROL结构体

助记符：	数据类型：	说明：
.EN	BOOL	使能位表明BSR指令被使能。
.DN	BOOL	完成位置位表示数组中的位已经向右移动了一个位置。
.UL	BOOL	卸载位是指令的输出。卸载位.UL存储移出数组位的状态。
.ER	BOOL	当长度值.LEN < 0.时置位错误位。
.LEN	DINT	长度指定数组中要移动位的数量。

说明： 当指令被使能时，BSR指令将数组中的位0卸载到.UL位中，其余的位顺序向右移动一个位置，并将源位装载到指定位的最高位。

BSR指令对内存中的一个连续区域进行操作。

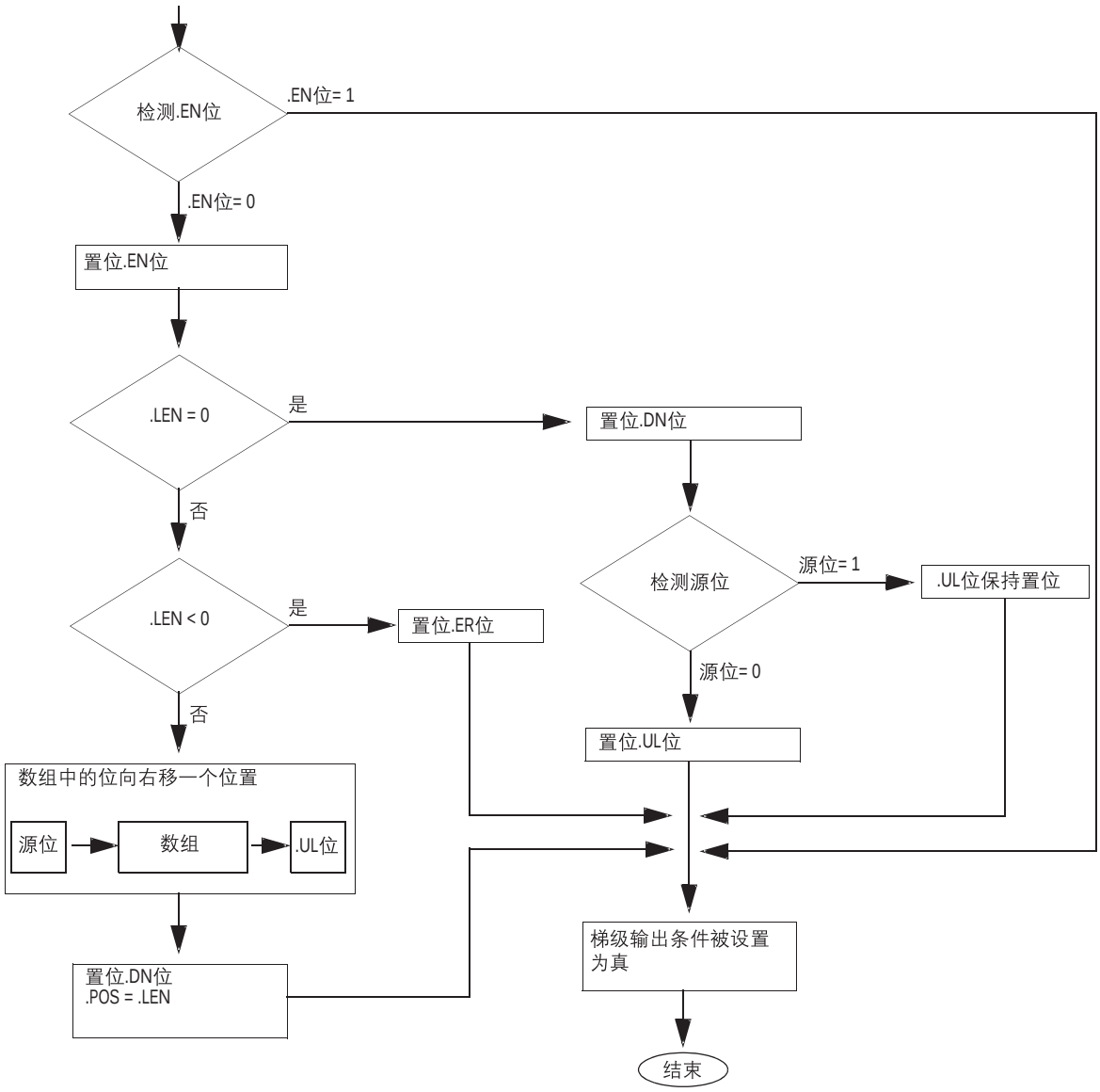
算术状态标志： 不影响

故障条件： 无

执行:

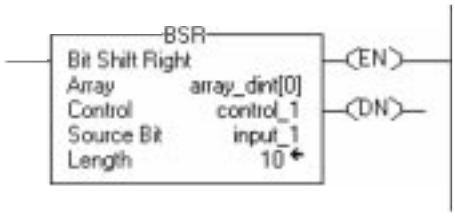
条件:	梯形图动作:
预扫描	清零使能位.EN。 清零完成位.DN。 清零错误位.ER。 清零位置值.POS。 梯级输出条件被设置为假。
梯级输入条件为假	清零使能位.EN。 清零完成位.DN。 清零错误位.ER。 清零位置值.POS。 梯级输出条件被设置为假。

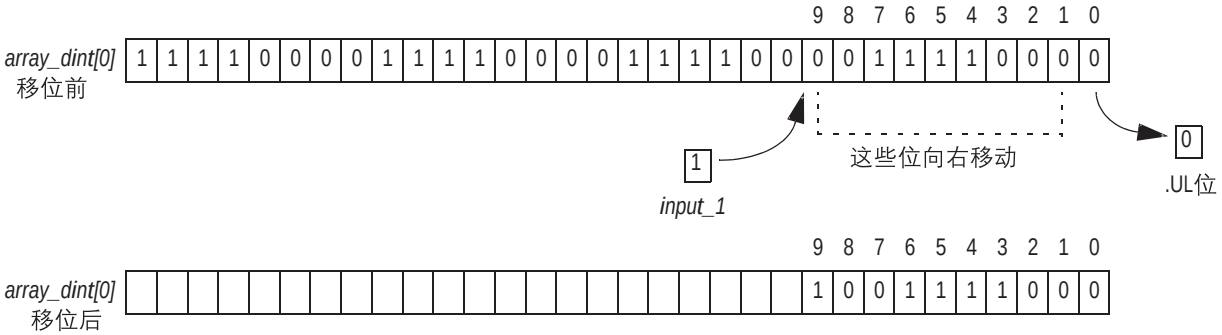
梯级输入条件为真:



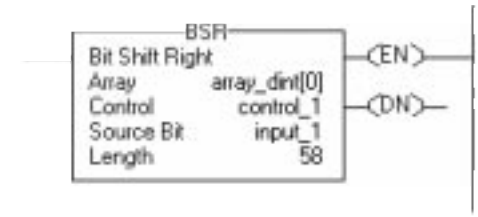
后期扫描	梯级输出条件被设置为假。

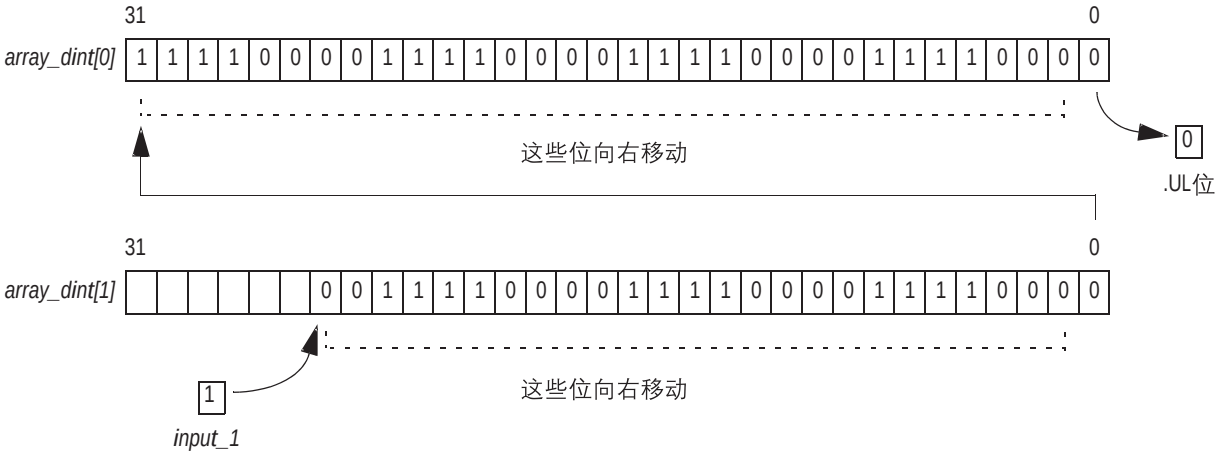
例1：当指令被使能时，BSR指令从array_dint[0] 中的位9开始执行。指令将array_dint[0].0卸载到.UL 位中，向右移动其余位，并将input_1 装载到array_dint[0].9中。其余位(10-31)的值无效。





例2：当指令被使能时，BSR指令从array_dint[1] 中的位25开始执行。指令将array_dint[0].0卸载到.UL 位中，向右移动其余位，并将input_1 装载到array_dint[1].25中。其余位(dint_array[1]中的31-26) 的值无效。注意array_dint[1].0如何移动到array_dint[0].31。



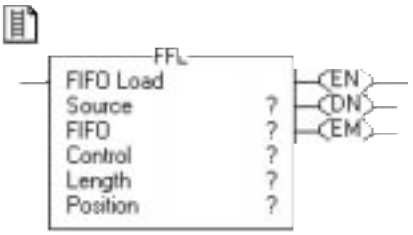


FIFO装载(FFL)

FFL指令将源值复制到FIFO中。

操作数：

梯形图



操作数：	数据类型：	输入格式：	说明：
Source 源值	**SINT** **INT** **DINT** **REAL** 字符串 结构体 源值转换成数组标签中的数据类型。	立即数 标签	要保存到FIFO的数据 较小整数用符号-填充法转换成较大整数。
FIFO 堆栈	**SINT** **INT** **DINT** **REAL** 字符串 结构体	数组标签	要更改的FIFO 指定FIFO中的第一个元素 不能在下标处使用CONTROL.POS
Control 控制	CONTROL	标签	指令操作的控制结构体 一般与相应的FFU指令用同一个CONTROL
Length 长度	DINT	立即数	FIFO中一次可容纳的最多元素数量
Position 位置	DINT	立即数	指令将数据装载在FIFO缓冲区中下一个区域 典型初始值是0

如果以用户自定义的结构体作为源值或FIFO操作数的数据类型，则两个操作数要用同一个结构体。

CONTROL结构体

助记符：	数据类型：	说明：
.EN	BOOL	使能位表示FFL指令被使能。
.DN	BOOL	完成位置位表示FIFO已满(.POS = .LEN)。完成位.DN禁止向FIFO装载数据，直到.POS < .LEN。
.EM	BOOL	栈空位表示FIFO为空。如果LEN ≤0 或 .POS < 0，则.EM位和.DN 位均被置位。
.LEN	DINT	长度值指示FIFO中一次可容纳的最多元素数量。
.POS	DINT	位置值指示了指令将下一个数值装载到FIFO区域位置。

说明：FFL指令和FFU指令按先进/先出的顺序存取数据。当成对使用这两条指令时，FFL与FFU指令的作用相当于一个异步移位寄存器。

一般源值和FIFO具有相同数据类型。

当指令被使能时，FFL 指令将源值装载到FIFO由.POS值指定的位置中。每次指令被使能时向FIFO装载一个值，直到栈满为止。

FFL指令对内存中的一个连续区域进行操作。

算术状态标志：不影响

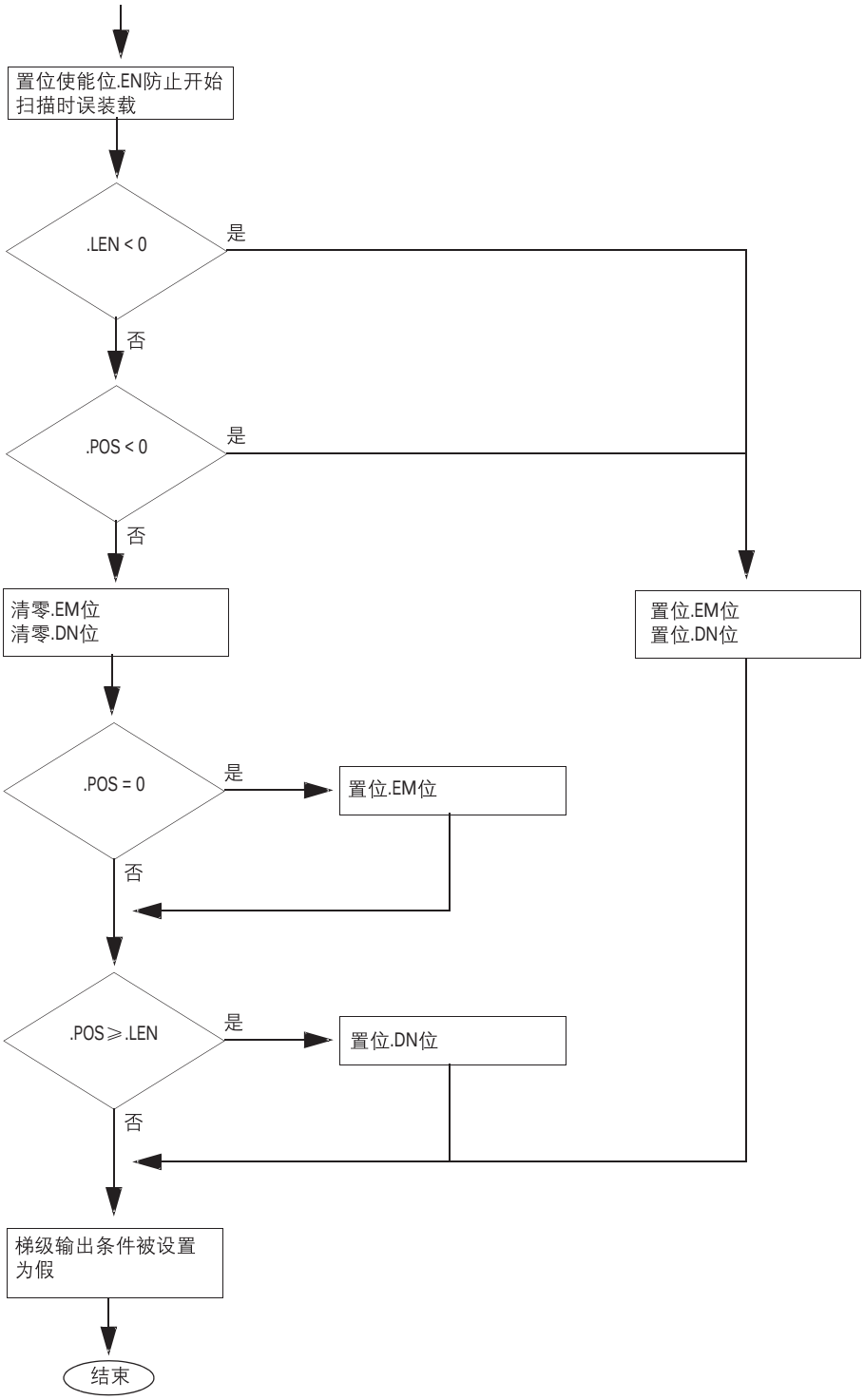
故障条件：

发生主要故障的条件：	故障类型：	故障代码：
(起始元素 + .POS) > FIFO 数组长度	4	20

执行:

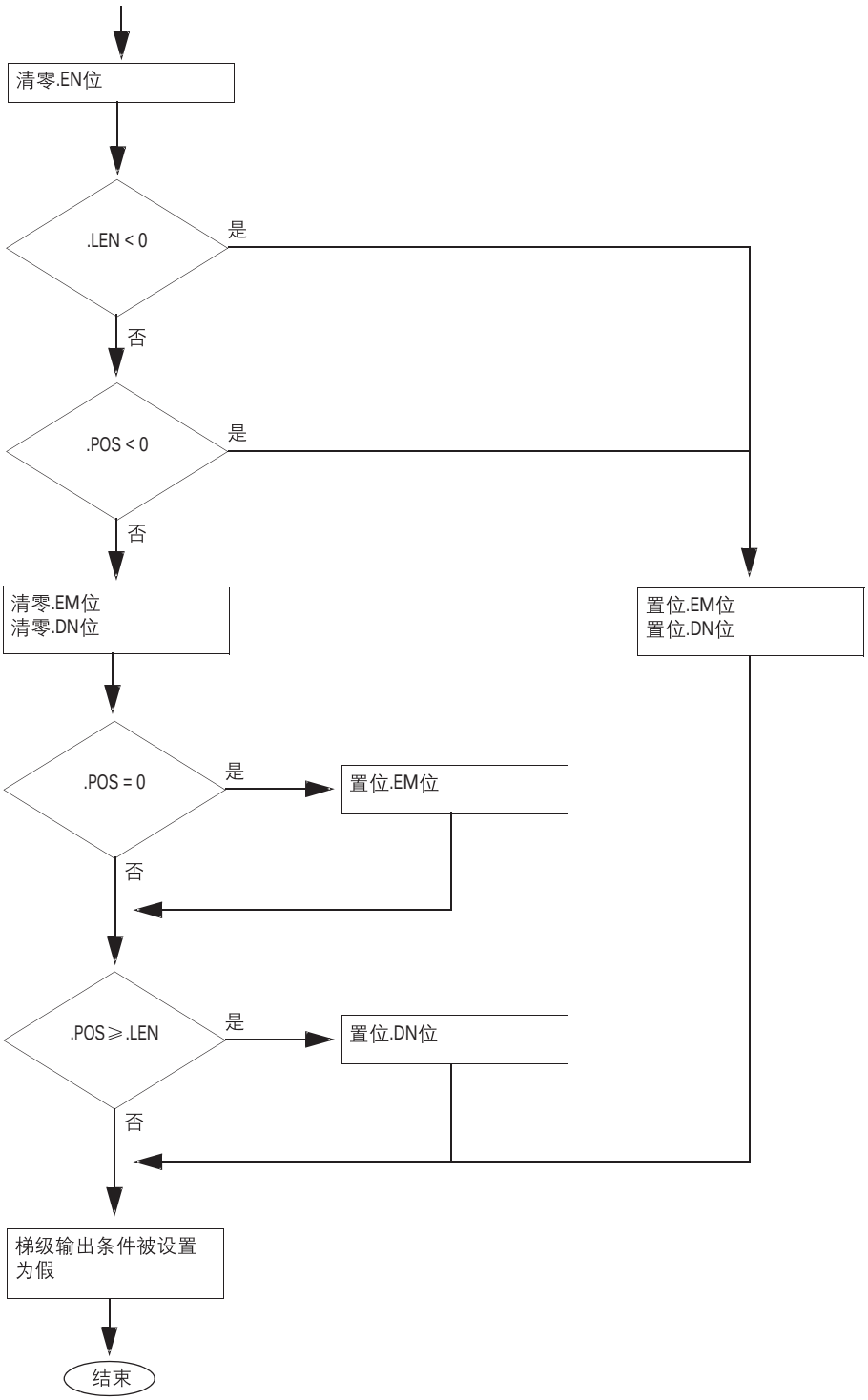
条件: 梯形图动作:

预扫描

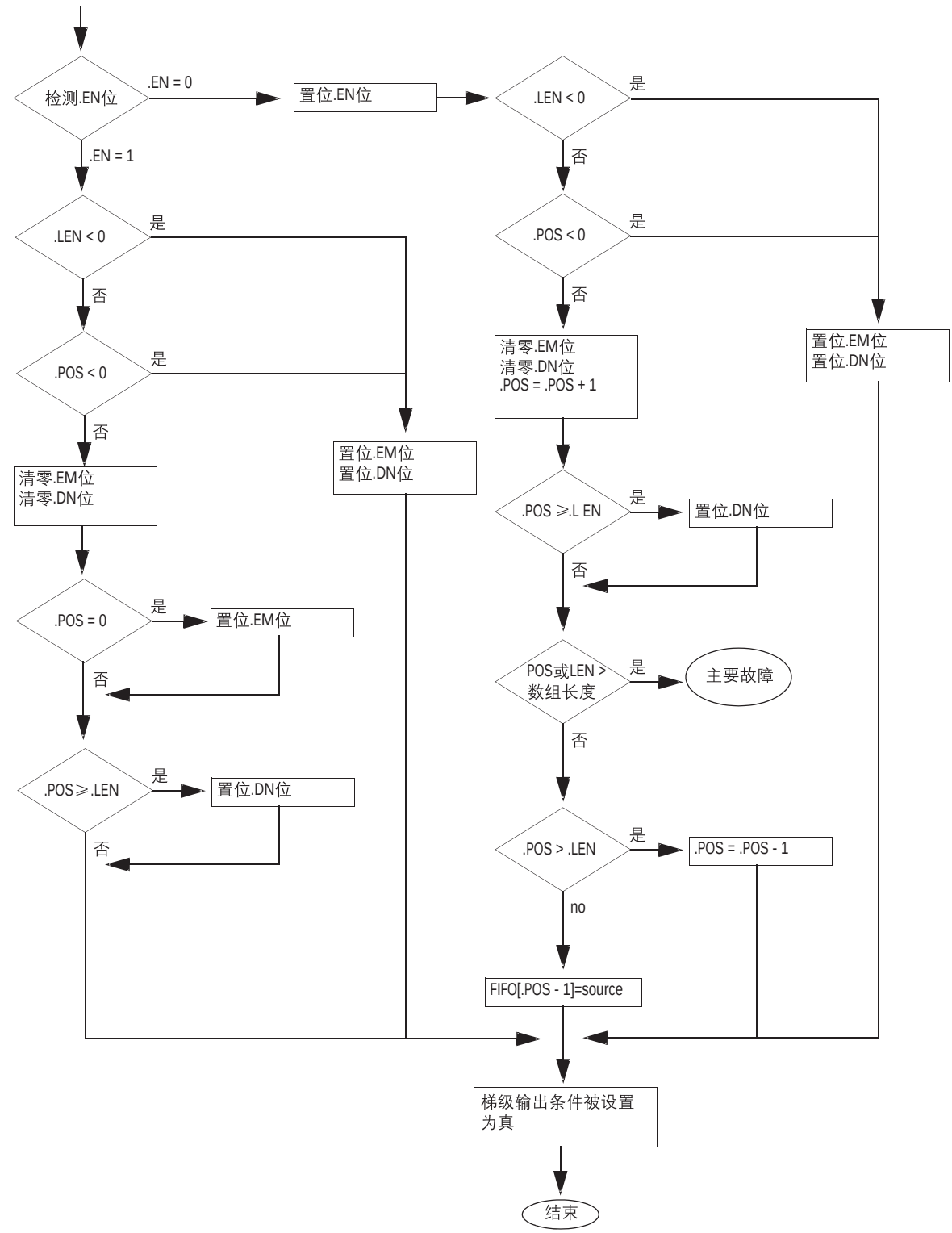


条件： **梯形图动作：**

梯级输入条件为假

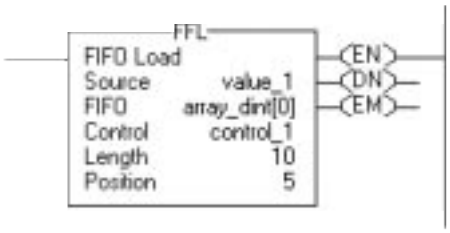


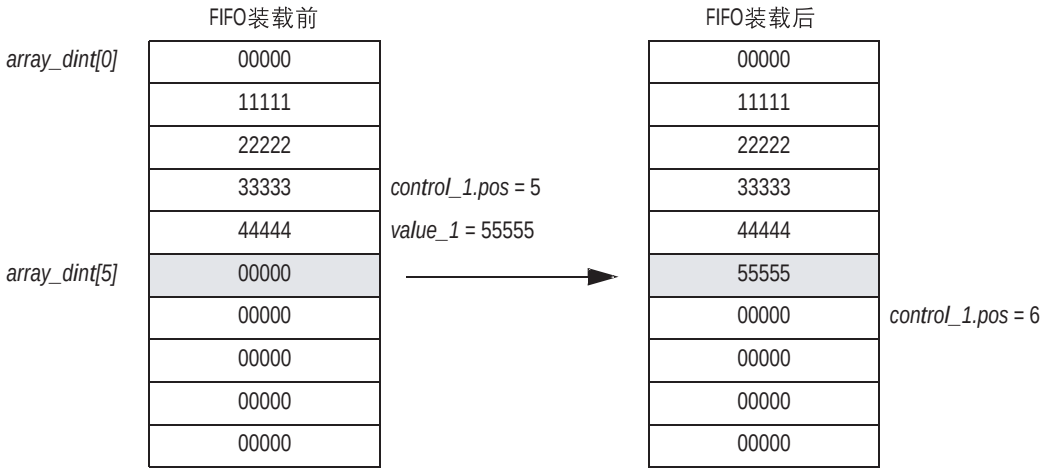
条件:	梯形图动作:
梯级输入条件为真	



后期扫描	梯级输出条件被设置为假。

举例：当指令被使能时，FFL 指令将value_1值装载到FIFO中的下一位置， 在本例中是array_dint[5] 。





FIFO卸载指令(FFU)

FFU指令从FIFO的位置0(第一个位置) 中卸载数值并将其存入目的标签中。FIFO中的其余数据向下移动一个位置。

操作数：

梯形图

操作数：	数据类型：	格式：	说明：
FIFO 栈	SINT INT DINT REAL 字符串 结构体	数组标签	要更改的FIFO 指定FIFO中的第一个元素 不能在下标处使用CONTROL.POS。
Destination 目的	SINT INT DINT REAL 字符串 结构体	标签	从FIFO中取出的数值 目的标签数值转换成目的标签的数据类型。较小整数用符号-填充法转换成较大整数。
Control 控制	CONTROL	标签	指令操作的控制结构体 一般与相应的FFL指令用同一个CONTROL
Length 长度	DINT	立即数	FIFO中一次可容纳的最多元素数量
Position 位置	DINT	立即数	FIFO指令将数据卸载在缓冲区中下一个区域 典型初始值是0

如果以用户自定义的结构体作为FIFO或目的标签的数据类型，则两个操作数要使用同一个结构体。

CONTROL结构体

助记符：	数据类型：	说明：
.EU	BOOL	使能卸载位表明FFU指令被使能。置位.EU 位以防止程序扫描开始时错误地卸载。
.DN	BOOL	完成位置位表示FIFO已满(.POS = .LEN)。
.EM	BOOL	栈空位表示FIFO为空。如果LEN ≤ 0 或 .POS < 0，则.EM位和.DN 位都置位。
.LEN	DINT	长度值表明FIFO中一次可容纳的最多元素数量
.POS	DINT	位置值表示已经装入FIFO中数据的末尾位置。

说明：FFU指令和FFL指令按先进/先出的顺序存取数据。

当指令被使能时，FFU 指令从FIFO的第一个元素卸载数据并将其存入目的标签中。每次指令被使能时都卸载一个值，直到栈空为止。如果FIFO为空，则FFU返回0值到目的标签。

FFU指令对内存中的一个连续区域进行操作。

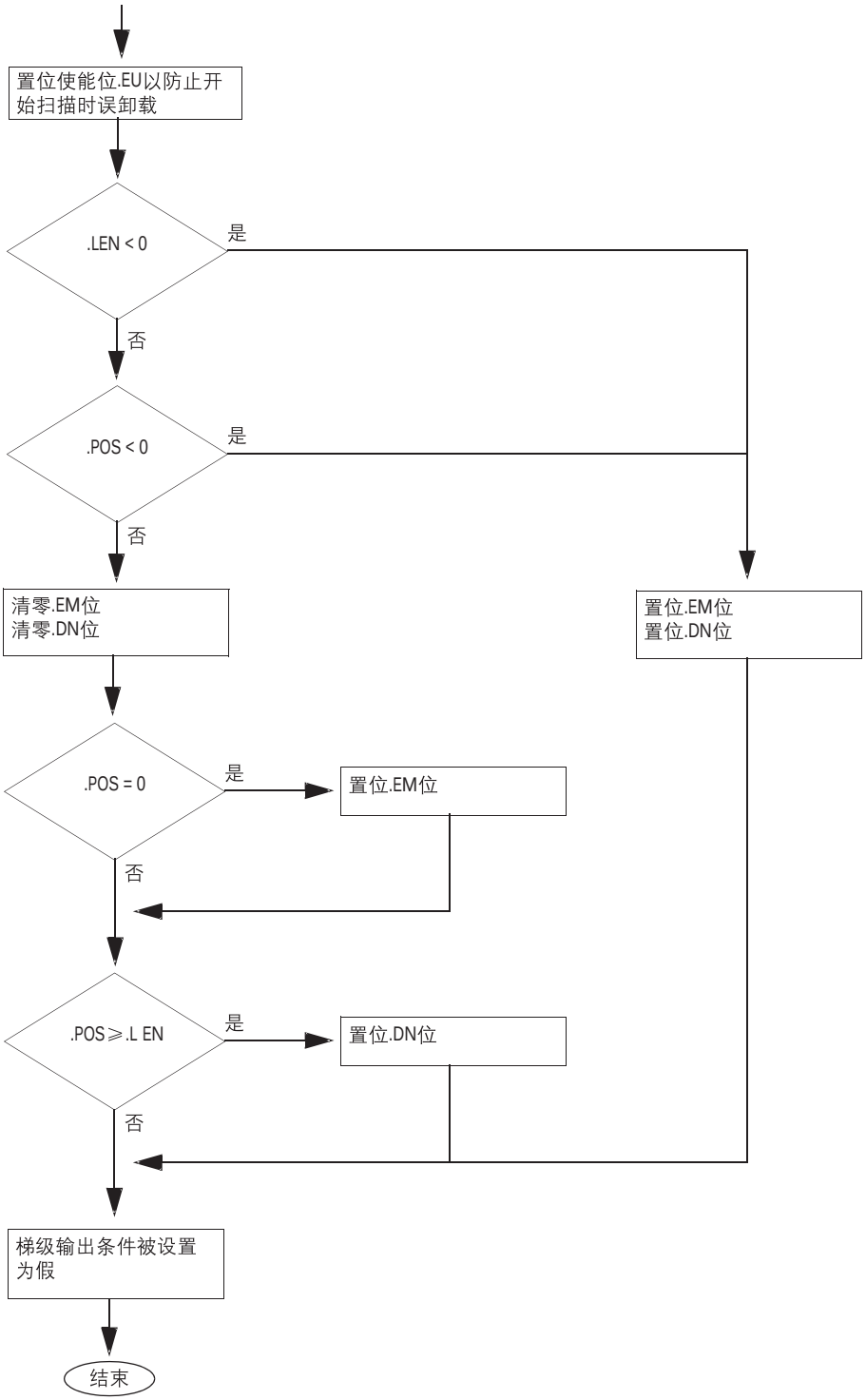
算术状态标志：不影响

故障条件：

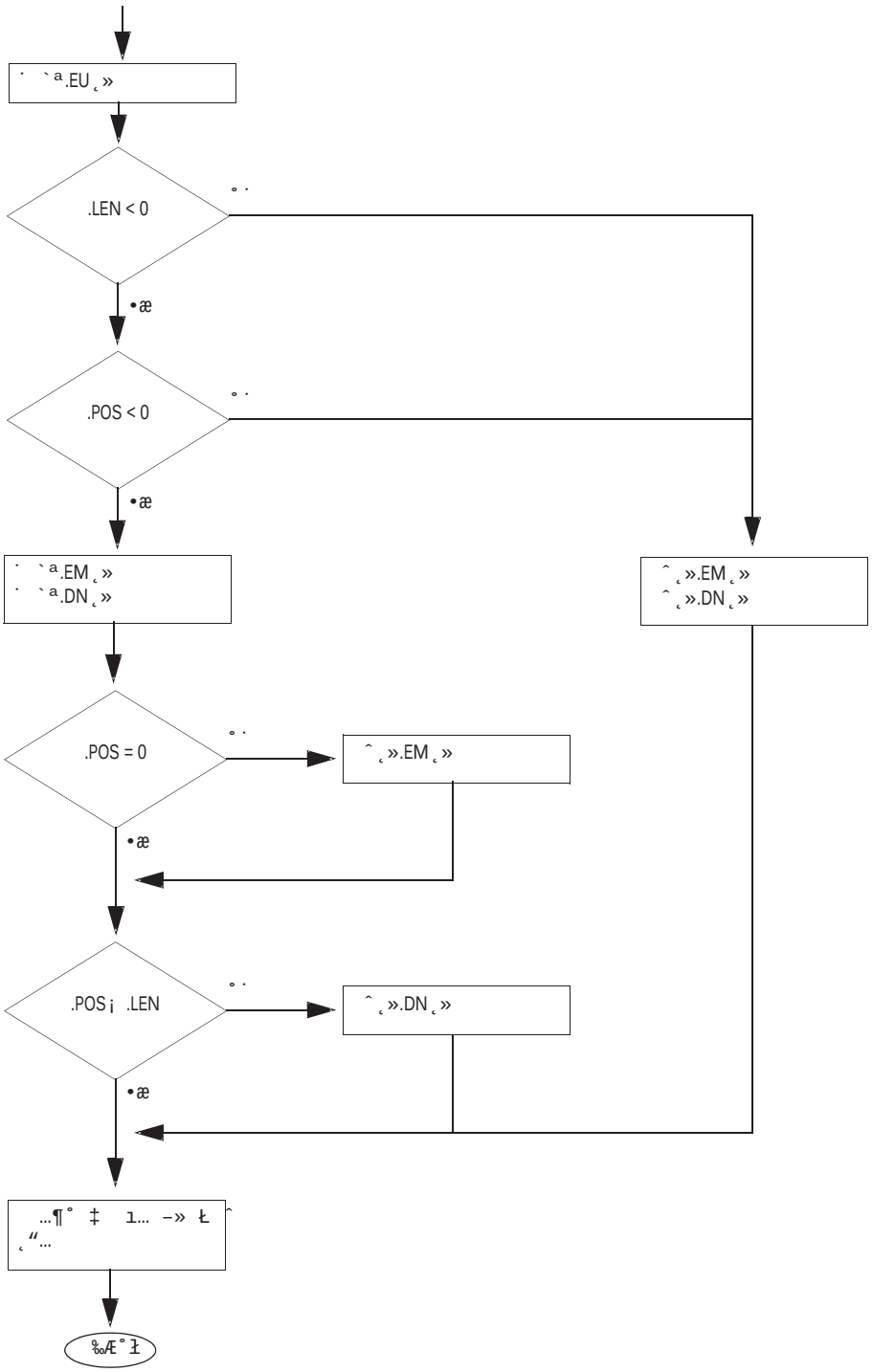
发生主要故障的条件：	故障类型：	故障代码：
长度> FIFO 数组长度	4	20

执行:

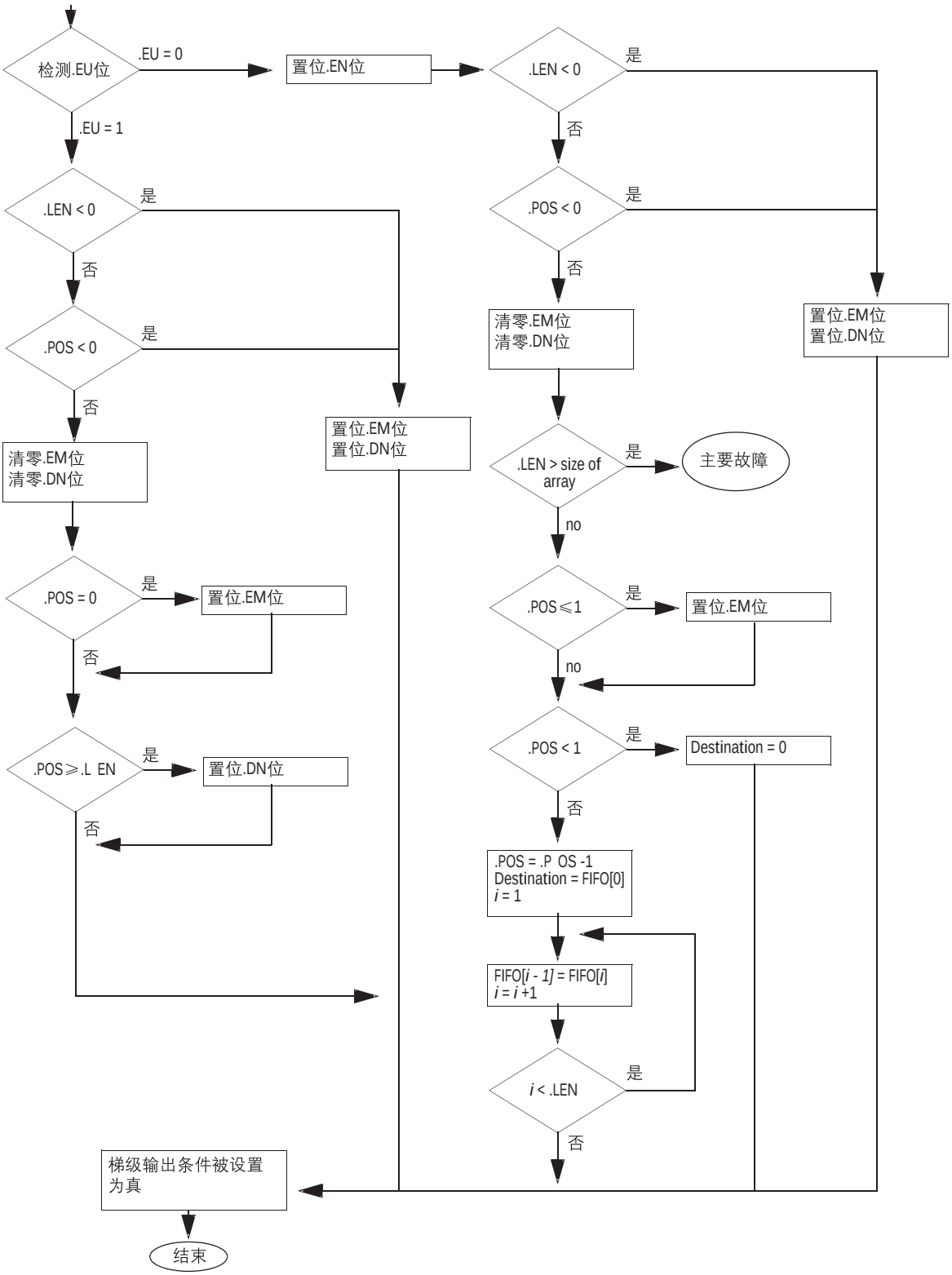
条件:	梯形图动作:
预扫描	



条件：	梯形图动作：
梯级輸入条件为假	

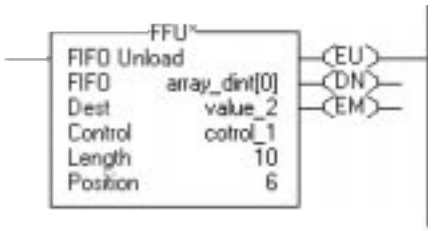


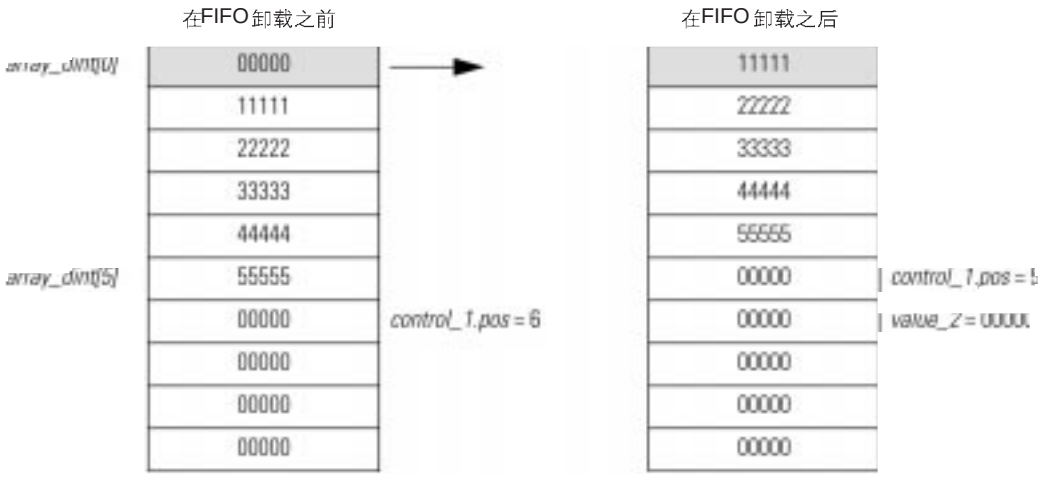
条件:	梯形图动作:
梯级输入条件为真	



后期扫描	梯级输出条件被设置为假。

举例：当指令被使能时， FFU指令将array_dint[0] 卸载到value_2中， 并移动 array_dint 中的其余元素。

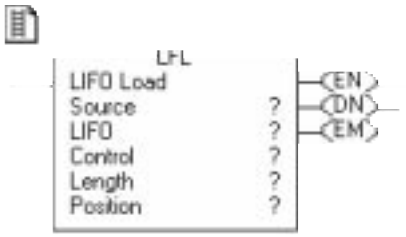




LIFO装载 (LFL)

LFL指令将源值复制到LIFO中。

操作数：



梯形图

操作数：	数据类型：	格式：	说明：
Source 源值	**SINT** **INT** **DINT** **REAL** 字符串 结构体 源值转换成数组标签的数据类型时，较小整数用符号-填充法转换成较大整数。	立即数 标签	要存入LIFO的数据
LIFO 栈	**SINT** **INT** **DINT** **REAL** 字符串 结构体	数组标签	要更改的LIFO 指定 LIFO 中的第一个元素 下标处不能使用CONTROL.POS
Control 控制	CONTROL	标签	指令操作的控制结构体 一般与相应的LFU指令用同一个CONTROL
Length 长度	DINT	立即数	LIFO中一次可容纳的最多元素数量
Position 位置	DINT	立即数	指令将数据装载在LIFO缓冲区中下一个位置 初始值是0

如果用用户自定义的结构作为源或LIFO的数据类型，则两个操作数要用同一个结构体。

CONTROL结构体

助记符：	数据类型：	说明：
.EN	BOOL	使能位—表示LFL指令被使能。
.DN	BOOL	完成位—该位置位表示LIFO已满(.POS = .LEN)。在.POS < .LEN之前完成位.DN禁止向LIFO装载数据。
.EM	BOOL	栈空位—表示LIFO为空。如果LEN ≤ 0 或 .POS < 0，则.EM位和.DN 位都被置位。
.LEN	DINT	长度值—LIFO中一次可容纳的最多元素数量。
.POS	DINT	位置值—指示了LIFO指令将下一个数值装载到缓冲区域。

说明：LFL 指令和LFU 指令按后进/先出的顺序存取数据。当成对使用这两条指令时，LFL 与LFU指令的作用相当于一个异步移位寄存器。

一般源值和LIFO 用相同数据类型。

当指令被使能时，LFL 指令将源值装载到LIFO由.POS值指定的位置中。每次指令被使能时向LIFO 装载一个值，直到栈满为止。

LFL 指令对内存中的连续区域进行操作。

算术状态标志：不影响

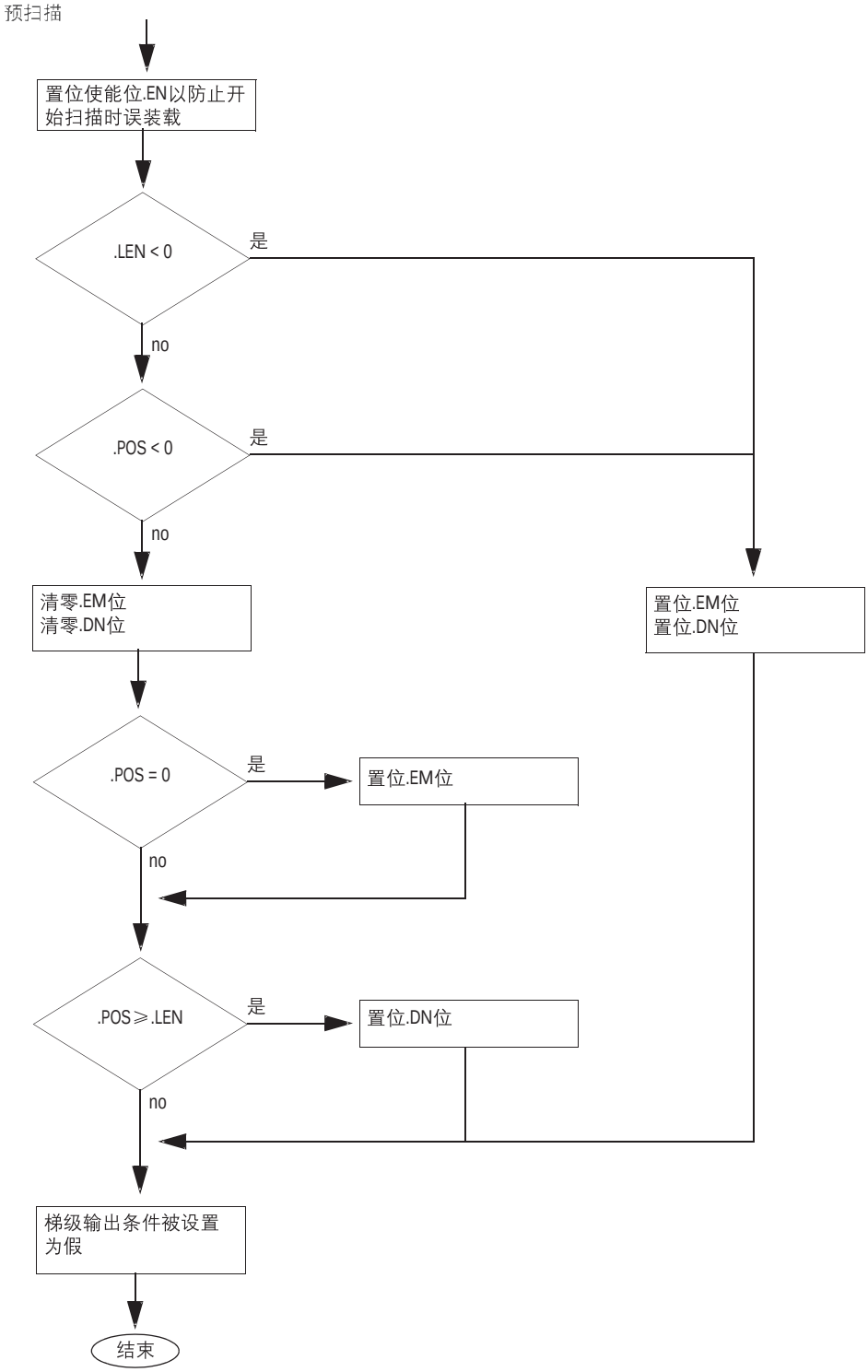
故障条件：

发生主要故障的条件：	故障类型：	故障代码：
(起始元素 + .POS) > LIFO 数组长度	4	20

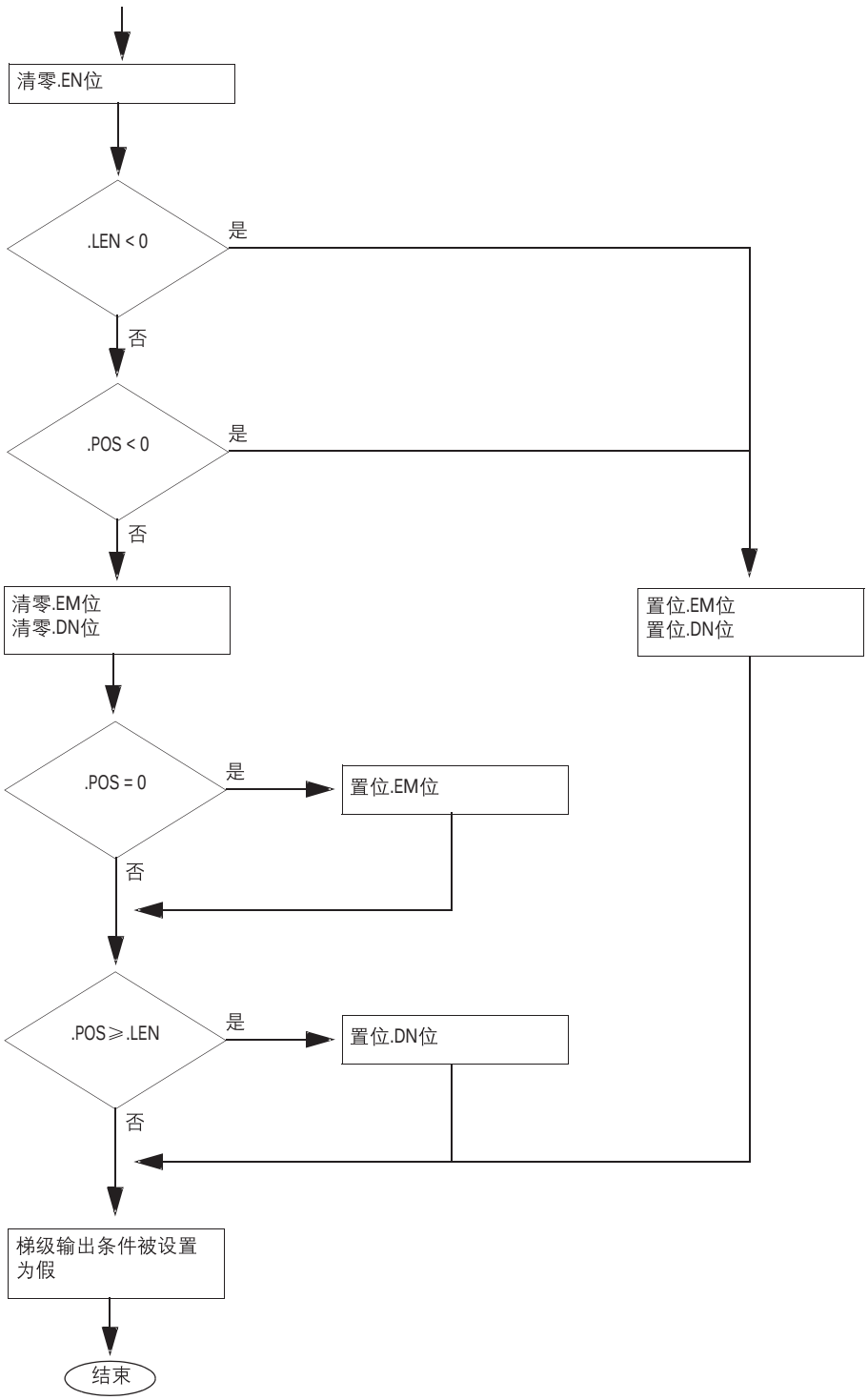
执行:

条件: 梯形图动作:

预扫描

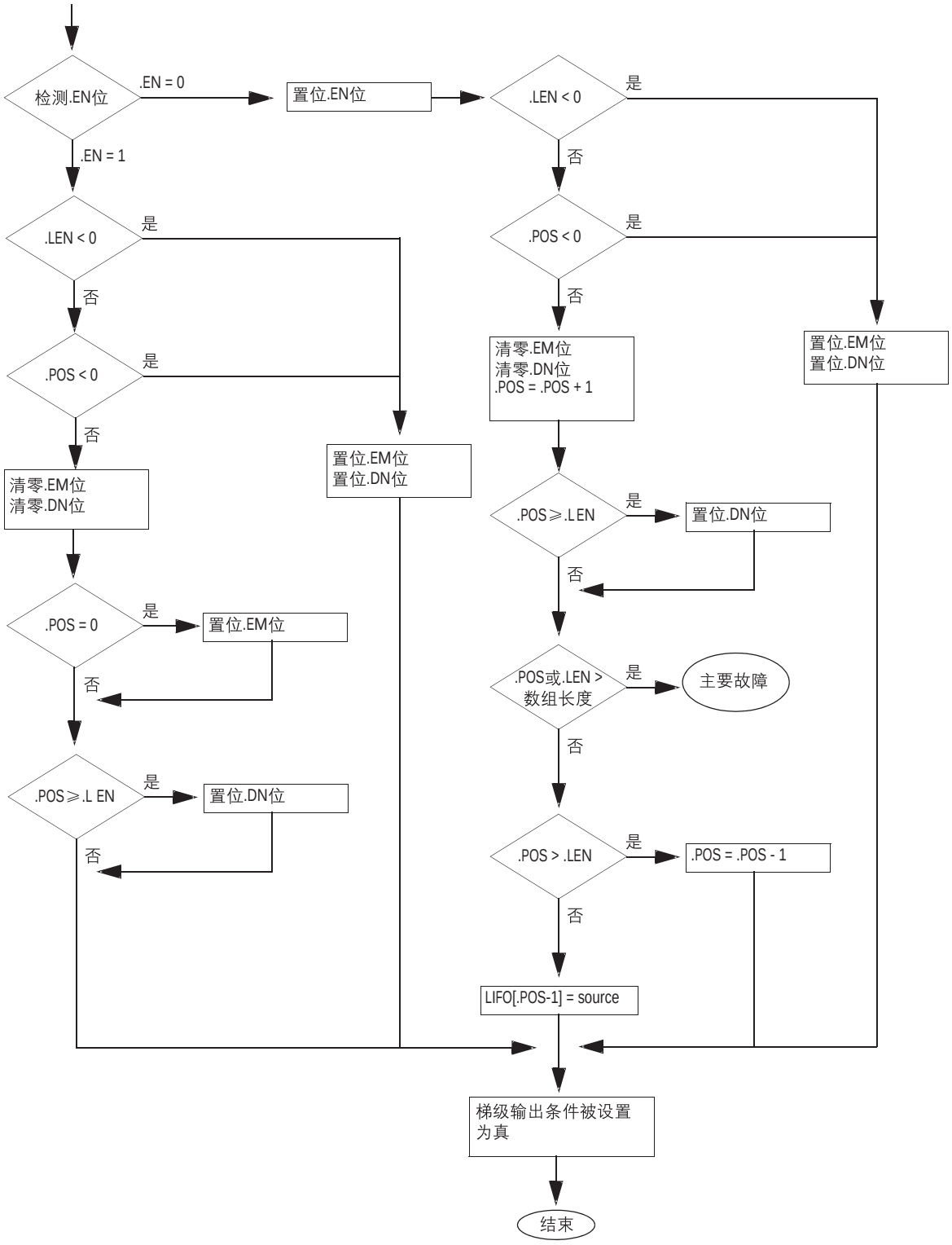


条件：	梯形图动作：
梯级输入条件为假	



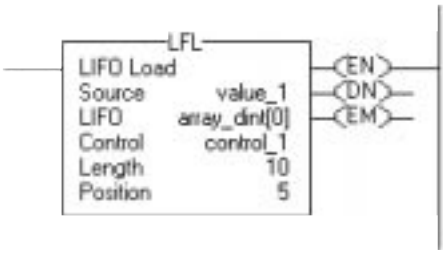
条件:	梯形图动作:

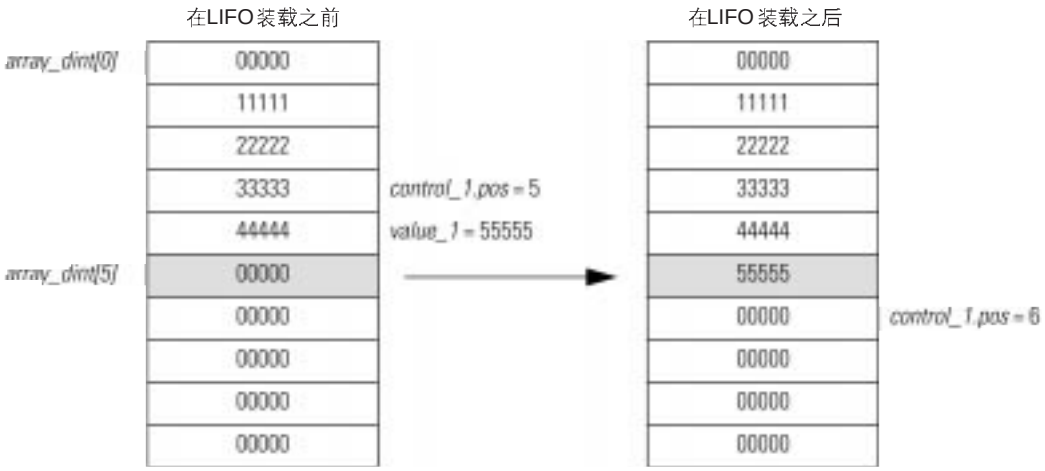
梯级输入条件为真



后期扫描	梯级输出条件被设置为假。

举例：当指令被使能时，LFL 指令将value_1 值装载到LIFO 中的下个位置，在本例中是array_dint[5] 。

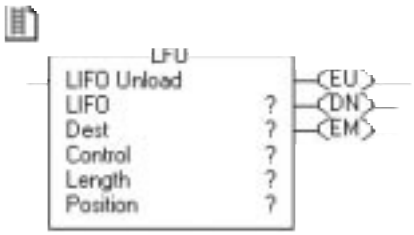




LIFO卸载 (LFU)

LFU指令从LIFO的.POS中卸载数值并在该位置存入0值。

操作数：



梯形图

操作数：	数据类型：	输入格式：	说明：
LIFO 栈	**SINT** **INT** **DINT** **REAL** 字符串 结构体	数组标签	要更改的LIFO 指定LIFO中的第一个元素 不能在此下标处使用CONTROL.POS
Destination 目的单元	**SINT** **INT** **DINT** **REAL** 字符串 结构体	标签	从LIFO中取出的数据 目的标签值转换成目的标签数据类型时，较小整数用符号-填充法转换成较大整数。
Control 控制	CONTROL	标签	指令操作的控制结构体 一般与相应的LFL指令用同一个CONTROL
Length 长度	DINT	立即数	LIFO中一次可容纳的最多元素数量
Position 位置	DINT	立即数	LIFO指令将数据卸载在缓冲区中下一个区域 初始值是0

如果以用户自定义的结构作为LIFO或目的标签的数据类型，则两个操作数要用同一个结构体。

CONTROL结构体

助记符：	数据类型：	说明：
.EU	BOOL	使能卸载位表明LFU指令被使能。置位.EU 位以防止程序扫描开始时错误地卸载。
.DN	BOOL	完成位置位表示LIFO已满(.POS = .LEN)。
.EM	BOOL	栈空位表示LIFO为空。如果LEN ≤ 0 或 .POS < 0，则.EM位和.DN 位都被置位。
.LEN	DINT	长度值表示LIFO中一次可容纳的最多元素数量
.POS	DINT	位置值表示已经装入LIFO中数据的末尾。

说明：LFU指令和LFL指令按后进/先出的顺序存取数据。

当指令被使能时，LFU 指令从LIFO的.POS中卸载数据并将其存入目的标签中。每次指令被使能时，卸载一个值并以0代替，直到栈空为止。如果LIFO为空，则LFU向目的标签返回0值。

LFU指令对内存中的一个连续区域进行操作。

算术状态标志：不影响

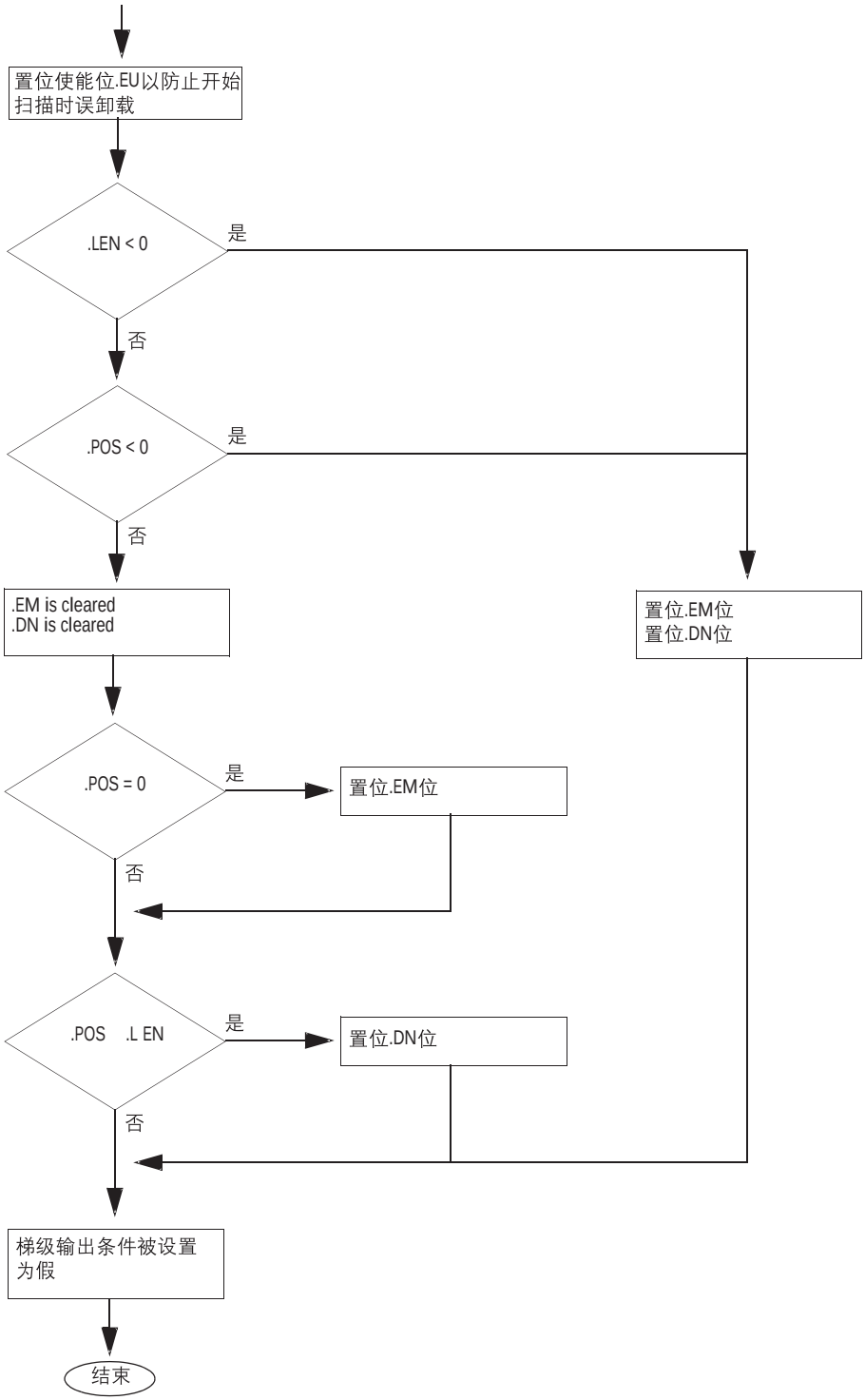
故障条件：

发生主要故障的条件：	故障类型：	故障代码：
长度> LIFO 数组长度	4	20

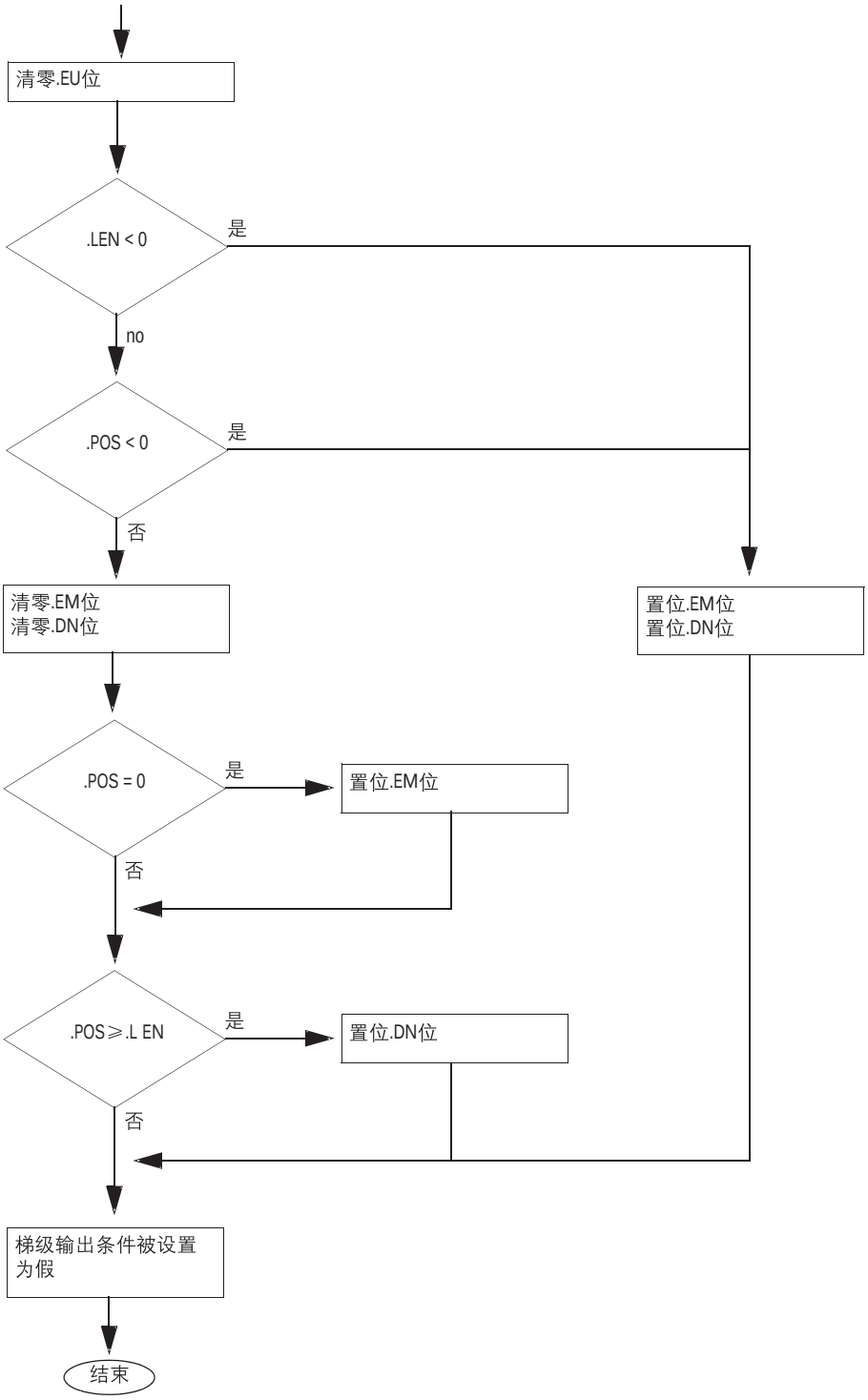
执行:

条件:	梯形图动作:

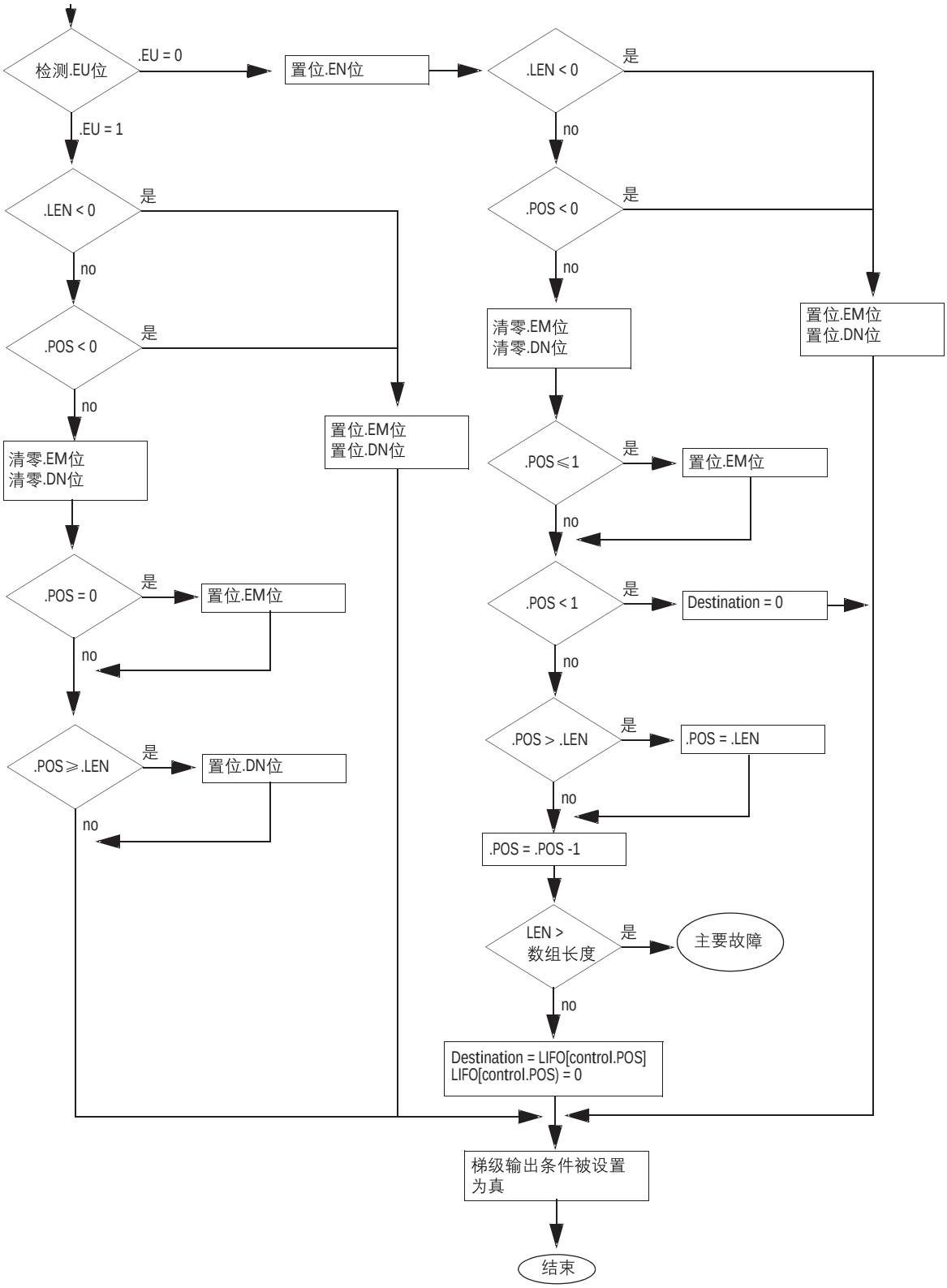
预扫描



条件：	梯形图动作：
梯级输入条件为假	

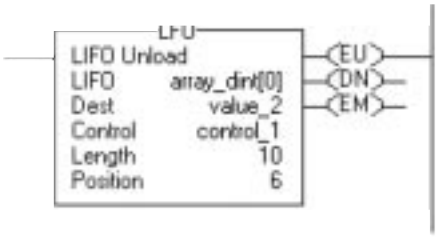


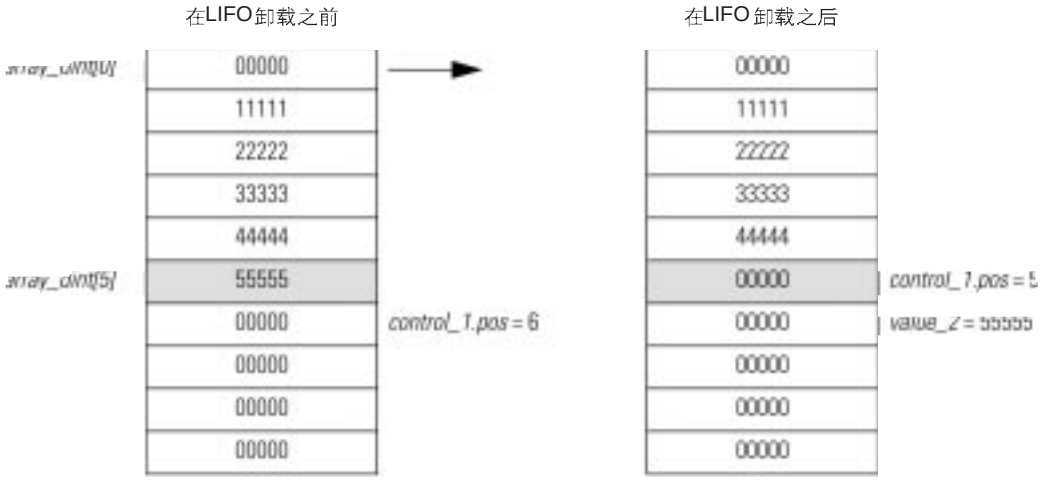
条件:	梯形图动作:
梯级输入条件为真	



后期扫描	梯级输出条件被设置为假。

举例：当指令被使能时， LFU指令将array_dint[5] 卸载到value_2中。





注释：

顺序器指令
(SQI, SQO, SQL)

简介

无动作。顺序器指令用来监控具有规律性和重复性的操作。

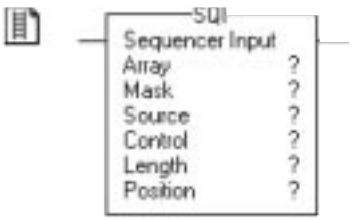
如果用户需要:	所用指令:	可用语言:	参见页码:
检测某步骤何时完成。	SQI	梯形图	9-2
设置下一步的输出条件。	SQO	梯形图	9-6
将参考条件装载至顺序器数组。	SQL	梯形图	9-10

对于梯形图语言编程，则黑体字的数据类型表示最优数据类型。如果指令的操作数全部使用同一优化数据类型，则指令的执行速度快且占用内存少。典型最优数据类型是DINT或REAL数据类型。

顺序器输入 (SQI)

SQI指令检测在SQO/SQI指令序列对中某一步何时完成。

操作数：



梯形图

操作数:	数据类型:	输入格式:	说明:
Array 数组	DINT	数组标签	顺序器数组 指定顺序器数组中的第一个元素 不能在下标处用CONTROL.POS
Mask 屏蔽	SINT INT **DINT** SINT或INT数据类型标签用符号-扩充法转换成DINT数据类型。	标签 立即数	要阻滞或通过的位
Source 源	SINT INT DINT SINT或INT数据类型标签用符号-扩充法转换成DINT数据类型。	标签	顺序器数组的输入数据
Control 控制	CONTROL	标签	指令操作的控制结构体 一般成对使用的SQO与SQL指令使用同一个 CONTROL
Length 长度	DINT	立即数	数组(顺序器表) 中要比较的元素数量
Position 位置	DINT	立即数	数组中的当前位置 典型初始值为0

CONTROL结构体

助记符:	数据类型:	说明:
.ER	BOOL	当长度值.LEN < 0， 位置值.POS < 0, 或位置值.POS > 长度值.LEN.时置位错误位。
.LEN	DINT	长度值指定顺序器数组中要执行的步数。
.POS	DINT	位置值表明指令当前比较元素的位置。

说明：当指令被使能时，SQI指令比较通过屏蔽的源中元素与数组元素是否相等。

成对使用的SQO与SQL指令一般用同一个 CONTROL结构体。

SQI指令对内存中的连续区域进行操作。

输入立即数作为屏蔽值

用户以立即数作屏蔽值时，编程软件缺省输入的数值为十进制。如果需要用其它格式输入屏蔽值，需在数值前按下列方法加相应的前缀。

前缀：	说明：
16#	十六进制 例如：16#0F0F
8#	八进制 例如：8#16
2#	二进制 例如：2#00110011

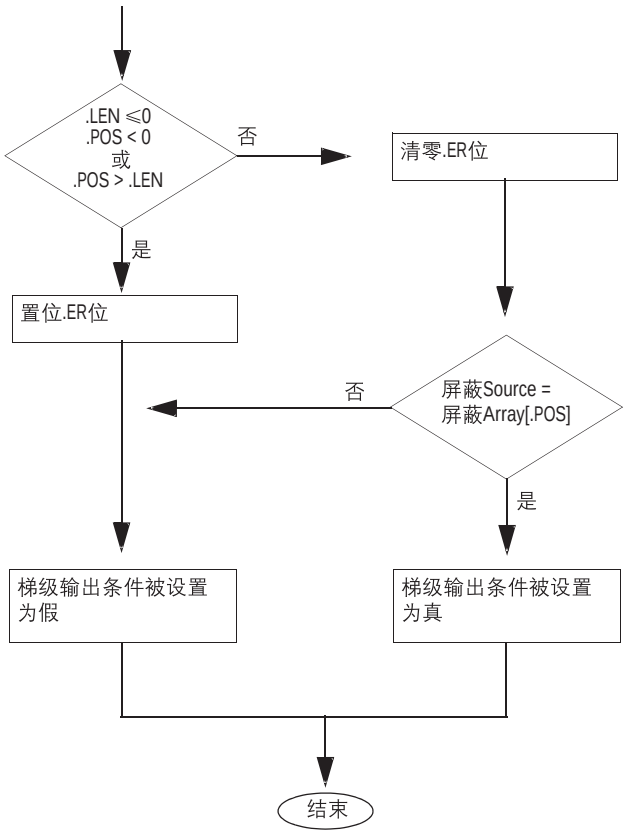
算术状态标志：不影响

故障条件：无

执行：

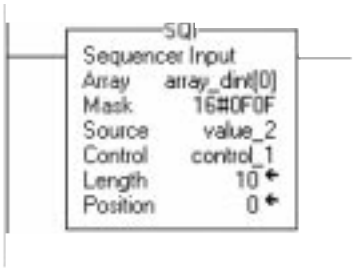
条件：	梯形图动作：
预扫描	梯级输出条件被设置为假。
梯级输入条件为假	梯级输出条件被设置为假。

梯级输入条件为真



后期扫描	梯级输出条件被设置为假。

举例：当指令被使能时，SQI指令使value_2通过屏蔽后与array_dint中的当前元素进行比较，结果二者是否相等。此例中的通过屏蔽的比较结果为真，所以梯级输出条件被设置为真。



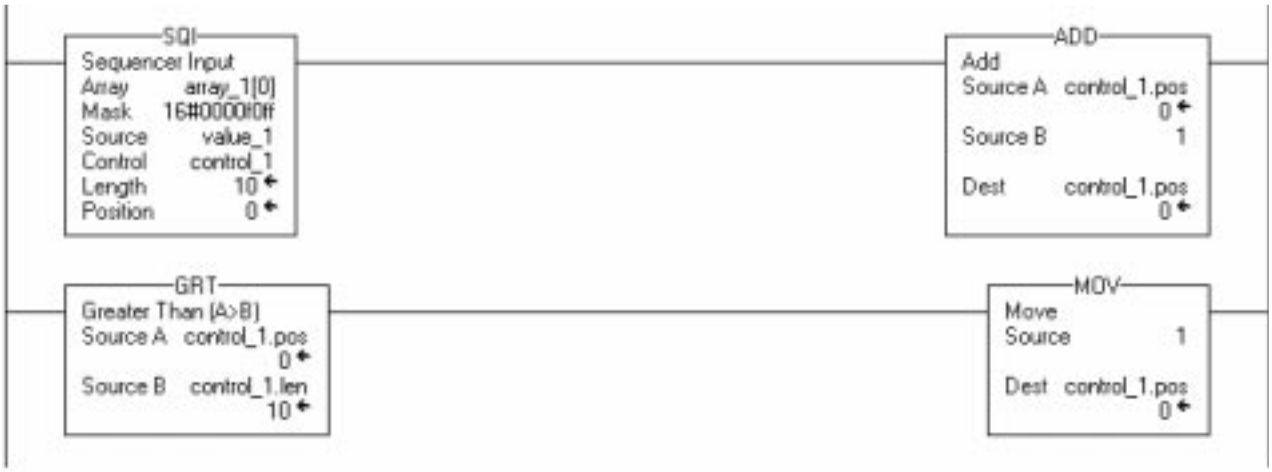
SQI指令操作数：	数值举例(DINTs 用二进制显示)
Source	xxxxxxxx xxxxxxxx xxxx0101 xxxx1010
Mask	00000000 00000000 00001111 00001111
Array	xxxxxxxx xxxxxxxx xxxx0101 xxxx1010

屏蔽字中位的0值表示该位不参与比较(此例中用xxxx表示)。

只用 SQI 而不用SQO指令

如果用户只用SQI指令而不用对应的SQO指令，则必须用外部指令来增加顺序器数组的位置值。

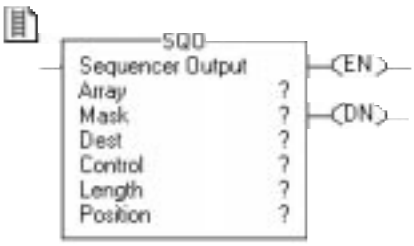
SQI 指令比较源值。ADD 指令增加顺序器数组的位置值。GRT 指令确定在顺序器数组中检测到的值是否有效。 MOV 指令在顺序器数组的全部操作都一次完成后复位位置值。



顺序器输出 (SQO)

SQO指令设置SQO/SQI指令序列对的下一步输出条件。

操作数: 操作数:



梯形图

操作数:	数据类型:	输入格式:	说明:
Array 数组	DINT	数组标签	顺序器数组 指定顺序器数组中的第一个元素 不能在下标处用CONTROL.POS
Mask 屏蔽值	SINT INT DINT SINT或INT数据类型标签用符号-扩充法转换成DINT数据类型。	标签 立即数	要阻滞或通过的位
Destination 目的标签	DINT	标签	顺序器数组的输出数据
Control 控制	CONTROL	标签	指令操作的控制结构体 一般成对使用的SQI与SQL指令用同一个CONTROL
Length 长度值	DINT	立即数	数组(顺序器表)中要输出的元素数量
Position 位置	DINT	立即数	数组中的当前位置 典型初始值为0

CONTROL结构体

助记符:	数据类型:	说明:
.EN	BOOL	使能位表明SQO指令被使能。
.DN	BOOL	当所有指定元素都已经移动到目的标签中时置位完成位。
.ER	BOOL	"当长度值.LEN ≤0，位置值.POS < 0, 或位置值.POS >长度值.LEN时置位错误位。"
.LEN	DINT	长度值指定顺序器数组中要执行的步数。
.POS	DINT	位置值表示控制器当前正处理的元素在数组中的位置。

说明: 当指令被使能时，SQO指令增加位置值，将指定位置的数据通过屏蔽后移出，并存储到目的标签中。如果位置值.POS >长度值.LEN, 则指令返回到顺序器数组的起始位置并从.POS = 1处继续执行。

成对使用的SQI与SQL指令一般用同一个 CONTROL。

SQO指令对内存中的连续区域进行操作。

输入立即数作为屏蔽值

当以立即数作为屏蔽值时，编程软件缺省输入的数值是十进制。如果想用其它格式输入屏蔽值，则需要在数值之前按下列方法加相应的前缀。

前缀：	说明：
16#	十六进制
	例如：16#0F0F
8#	八进制
	例如：8#16
2#	二进制
	例如：2#00110011

算术状态标志：不影响

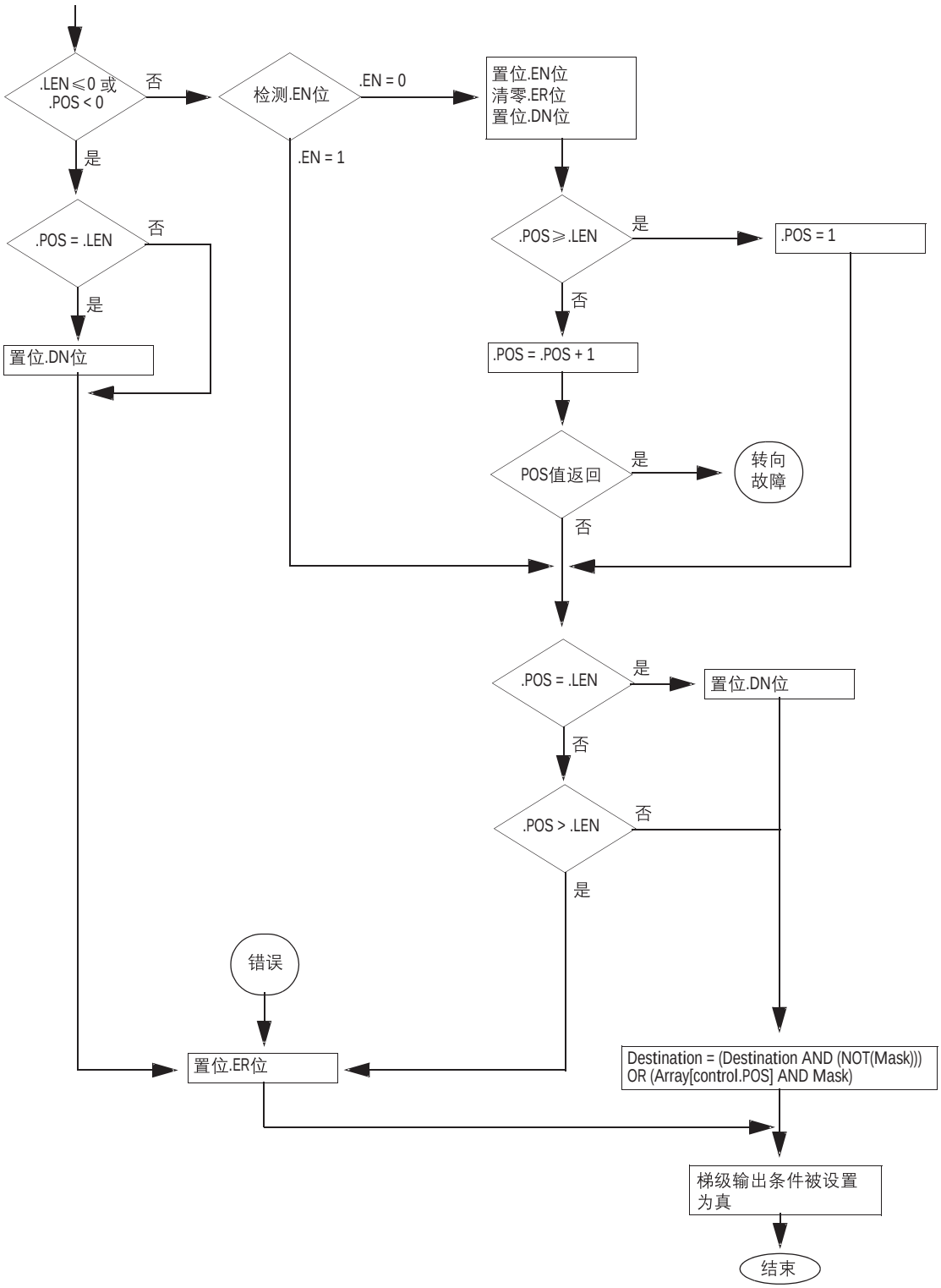
故障条件：无

执行:

条件：	梯形图动作：
预扫描	使能位.EN被置位以防止程序扫描开始时错误装载。 梯级输出条件被设置为假。
梯级输入条件为假	使能位.EN被清零。 梯级输出条件被设置为假。

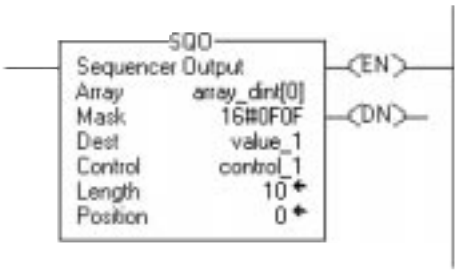
条件:	梯形图动作:

梯级输入条件为真



后期扫描	梯级输出条件被设置为假。

举例：当指令被使能时，SQO指令增加位置值，array_dint中当前位置的数据经过屏蔽后，将结果存储在value_1中。



SQO指令操作数：	数值举例(使用INTs 以二进制显示)
Array	xxxxxxxx xxxxxxxx xxxx0101 xxxx1010
Mask	00000000 00000000 00001111 00001111
Destination	xxxxxxxx xxxxxxxx xxxx0101 xxxx1010

屏蔽字中位的0值表示该位不参与比较(此例中用xxxx表示)

成对使用SQI与SQO指令

如果用户成对使用SQI和SQO指令，则一定要确定两条指令使用同样的控制结构体、长度和位置值。

复位SQO指令的位置值

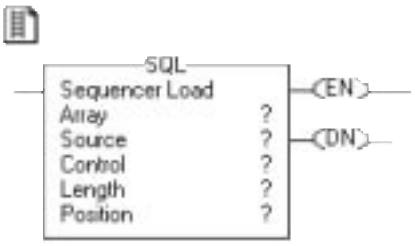
每次控制器由编程模式转换到运行模式， SQO指令都清零(初始化).POS值。可以用RES指令清零位置值，将位置值.POS复位到初始值(位置值.POS = 0)。本例用首次扫描位的状态清零.POS值。

顺序器装载 (SQL)

SQL指令将参考条件装载到顺序器数组中。

操作数：

梯形图



操作数：	数据类型：	输入格式：	说明：
Array 数组	DINT	数组标签	顺序器数组 指定顺序器数组中的第一个元素 不能在下标处用CONTROL.POS
Source 源	SINT INT DINT SINT或INT数据类型标签用符号-扩充法转换成DINT数据类型。	标签 立即数	要装载到顺序器数组中的输入数据
Control 控制	CONTROL	标签	指令操作的控制结构体 一般成对使用的SQI与SQO指令用同一个CONTROL结构体
Length 长度	DINT	立即数	要装载到数组(顺序器表)中的元素数量
Position 位置	DINT	立即数	数组中的当前位置 典型初始值为0

CONTROL结构体

助记符：	数据类型：	说明：
.EN	BOOL	使能位表明SQL指令被使能。
.DN	BOOL	当全部指定元素都已经装载到数组中时置位完成位。
.ER	BOOL	"当长度值.LEN ≤0， 位置值.POS < 0, 或位置值.POS >长度值.LEN时，置位错误位。"
.LEN	DINT	长度值指定顺序器数组中要执行的步数。
.POS	DINT	位置值表示控制器当前正处理的元素在队列中的位置。

说明： 当指令被使能时，SQL指令将顺序器数组位置值增加到下一个位置，并将源值装载到数组中的该位置。如果完成位.DN位被置位或位置值.POS ≥长度值.LEN，则指令设置位置值.POS = 1。

成对使用的SQI与SQO指令一般用同一个CONTROL结构体。

SQL指令对内存中的连续区域进行操作。

算术状态标志：不影响

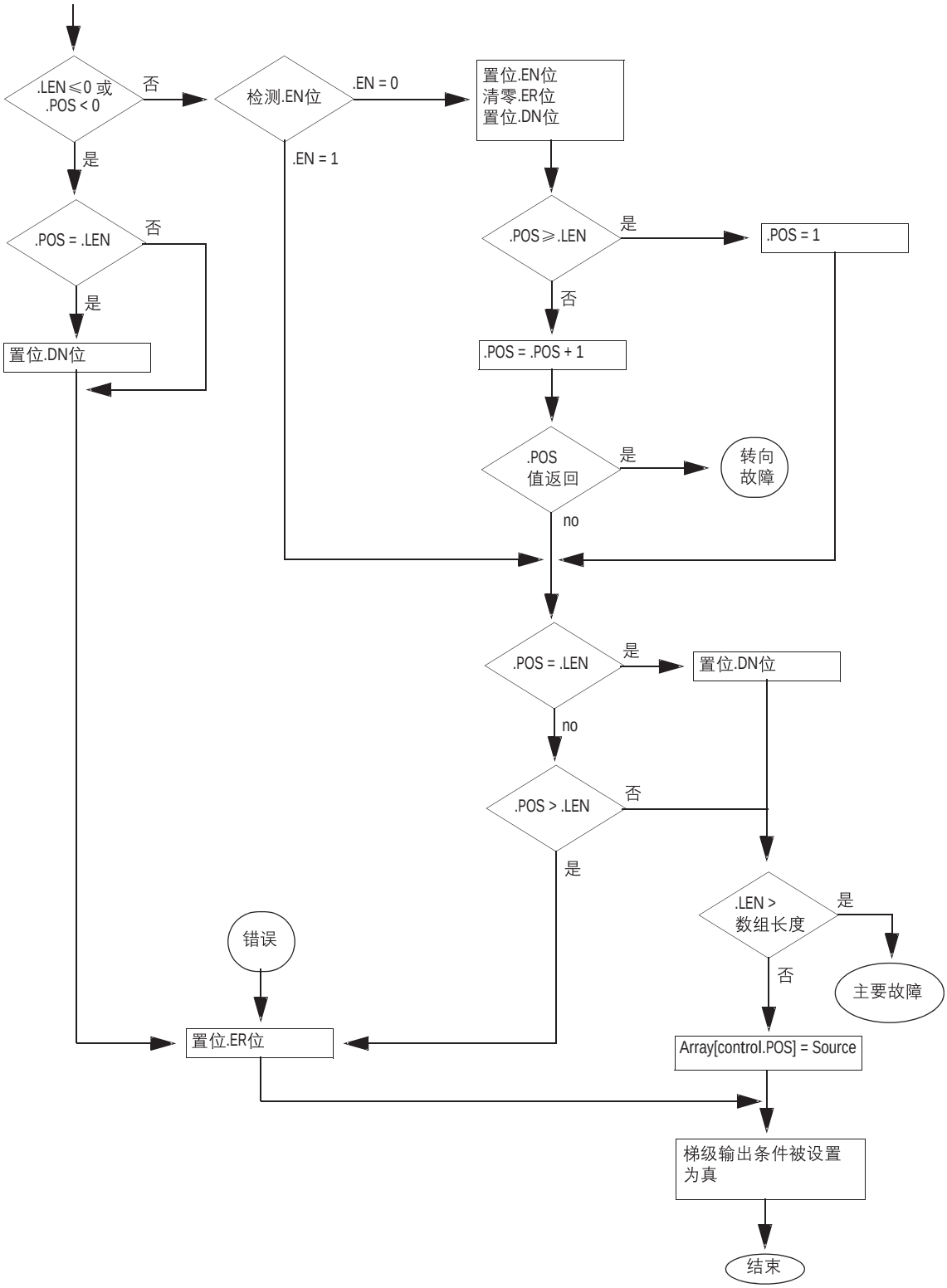
故障条件：

发生主要故障的条件：	故障类型：	故障代码：
长度> 数组长度	4	20

执行：

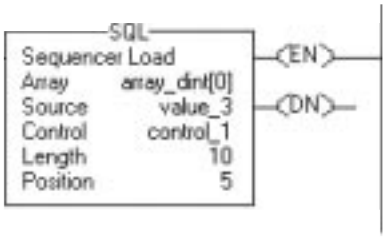
条件：	梯形图动作：
预扫描	置位.EN位以防止程序扫描开始时误装载。 梯级输出条件被设置为假。
梯级输入条件为假	清零使能位.EN。 梯级输出条件被设置为假。

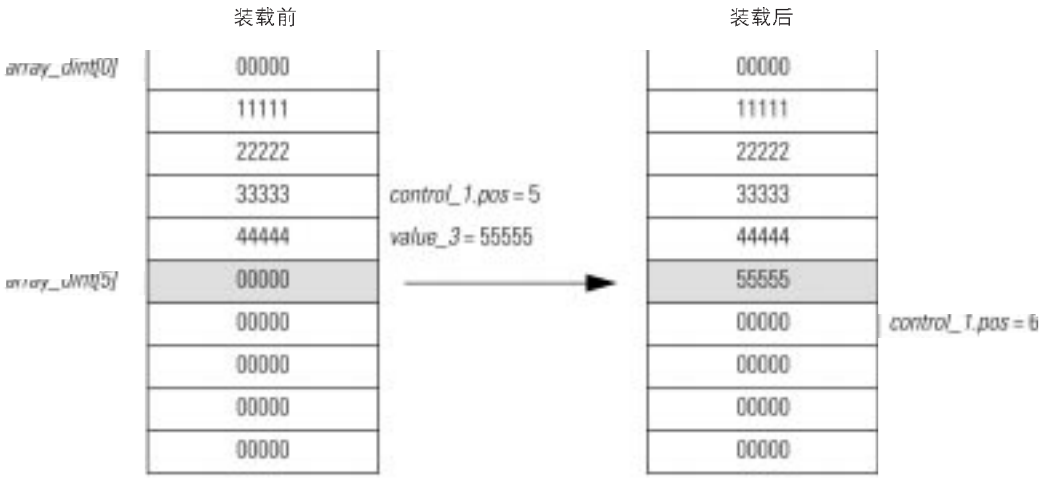
条件:	梯形图动作:
梯级输入条件为真	



后期扫描	梯级输出条件被设置为假。

举例：当指令被使能时，SQL指令将value_3 装载到顺序器数组的下一位置，在本例中是array_dint[5] 。





注释：

程序控制指令
(JMP，LBL，JSR，RET，SBR，JXR，TND，MCR，UID，UIE，AFI， NOP，EOT，SFP，SFR，EVENT)

简介

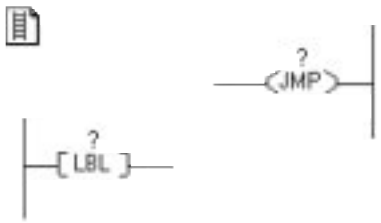
可以用程序控制指令改变例程逻辑执行的流程。

如果用户需要：	所用指令：	可用语言：	参见页码：
跳过逻辑中无需一直执行的部分。	JMP LBL	梯形图	10-2
跳转到一独立的子例程，向子例程传送数据，执行子例程，并返回结果。	JSR SBR RET	梯形图 功能块 结构文本	10-4
跳转到外部子例程(仅SoftLogix5800控制器)	JXR	梯形图	10-14
标记一个暂停以挂起程序的执行。	TND	梯形图 结构文本	10-17
禁止执行逻辑中一个区域的全部梯级。	MCR	梯形图	10-19
禁止用户任务。	UID	梯形图 结构文本	10-21
使能用户任务。	UIE	梯形图 结构文本	10-21
禁止一个梯级。	AFI	梯形图	10-23
在逻辑中插入一个占位符。	NOP	梯形图	10-24
结束顺序功能图的传送	EOT	梯形图 结构文本	10-25
暂停顺序功能图执行	SFP	梯形图 结构文本	10-27
复位顺序功能图	SFR	梯形图 结构文本	10-29
触发执行事件型任务	EVENT	梯形图 结构文本	10-31

跳转到标签 (JMP)
标签 (LBL)

JMP 和LBL指令跳过部分梯形图例程逻辑。

操作数：



梯形图

操作数：	数据类型：	格式：	说明：
JMP指令			
标签名称		标签名称	输入关联的LBL指令名称
LBL指令			
标签名称		标签名称	执行跳转到引用标签名称的LBL指令

说明：当指令被使能时，JMP 指令跳转到其引用的LBL指令，控制器从该处开始继续执行例程。当指令被禁止时，JMP指令不影响梯形图执行。

JMP指令可以向前或向后跳转执行梯形图。向前跳转到标签可以通过略过一些例程段来节省例程扫描时间。向后跳转会使控制器重复执行梯形图逻辑。

要注意向后跳转的次数不要过多，否则可能会因为控制器总也不能到达逻辑末尾而造成看门狗时间超时，这样会使控制器发生主要故障。

指令跳过的逻辑不被扫描。重要例程逻辑要放在跳转区域外。

LBL指令是有相同标签名称的JMP指令的目的地。**一定要确保LBL指令是梯级中第一条指令。**

同一个例程中的标签名称必须唯一。标签名称可为：

- 最多有40个字符
- 可包含字母，数字和下划线 (_)

算术状态标志：不影响

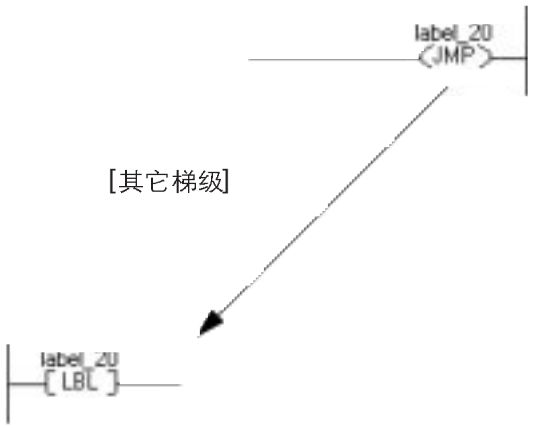
故障条件：

发生主要故障的条件：	故障类型：	故障代码：
标签不存在	4	42

执行：

条件：	动作：
预扫描	梯级输出条件被设置为假。
梯级输入条件为假	梯级输出条件被设置为假。
梯级输入条件为真	梯级输出条件被设置为真。 执行跳转到包含LBL指令的梯级，该LBL指令名称为JMP指令所引用。
后期扫描	梯级输出条件被设置为假。

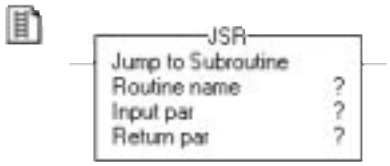
举例：当JMP 指令被使能时，例程执行跳过其接下来的逻辑梯级直达名称为label_20的LBL指令的梯级。



跳转到子例程 (JSR)
子例程(SBR)
返回(RET)

JSR指令 跳转去执行其它例程。SBR和RET指令是可选指令，可以与JSR指令交换数据。

JSR操作数：



梯形图

操作数：	数据类型：	格式：	说明：
Routine name 例程名称	ROUTINE	名称	要执行的例程(例如：子例程)
Input parameter 输入参数	BOOL SINT INT DINT REAL 结构体	立即数 标签 数组标签	本例程中的数据，用户希望将该数据复制到子例程的标签中 • 输入参数可选 • 如果需要，可以输入多个输入参数
Return Parameter 返回参数	BOOL SINT INT DINT REAL 结构体	标签 数组标签	本例程中的标签，要将子例程的结果复制到该标签中 • 返回参数可选 • 如果需要，可以输入多个返回参数

```
JSR(RoutineName,InputCount,
InputPar,ReturnPar);
```

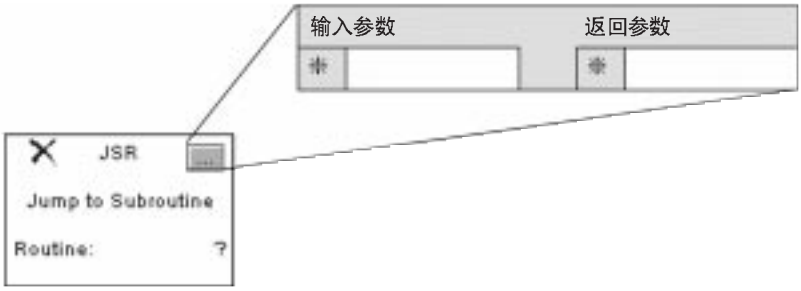
结构文本

操作数：	数据类型：	格式：	说明：
Routine name 例程名称	ROUTINE	名称	要执行的例程(例如：子例程)
Input count输入个数	INT DINT REAL	SINT	立即数　　输入参数的数量
Input parameter 输入参数	BOOL SINT INT DINT REAL 结构体	立即数 标签 数组标签	本例程中的数据，要将该数据复制到子例程的标签中 • 输入参数可选 • 如果需要，可以输入多个输入参数
Return Parameter 返回参数	BOOL SINT INT DINT REAL 结构体	标签 数组标签	本例程中的标签，要将子例程的结果复制到该标签中 • 返回参数可选 • 如果需要，可以输入多个返回参数

(JSR指令操作数继续见下页)

JSR操作数(继续)

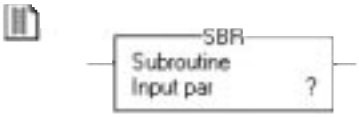
功能块



该指令操作数与梯形图的JSR指令操作数相同。

对于SBR 或RET指令中每个参数，都要与JSR指令中相应参数的数据类 型相同(包括任意维数 组)。如果 使用不同的数据类型，可能会导致不可预料的结果。

SBR操作数：在梯形图或结构文本例程中，SBR指令必须是首条指令。



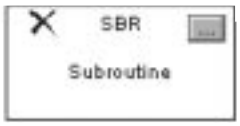
梯形图

操作数：	数据类型：	格式：	说明：
Input parameter 输入参数	BOOL SINT INT DINT REAL 结构体	标签 数组标签	本例程的标签，将JSR指令中对应的输入参数复制到该标签中

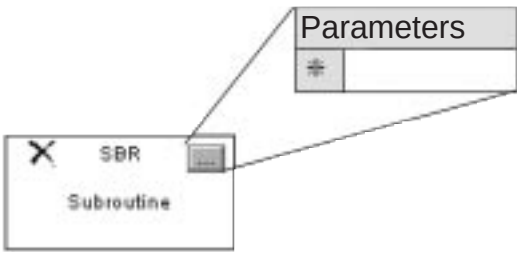
```
SBR(InputPar);
```

结构文本

该指令操作数与梯形图SBR指令的操作数相同。



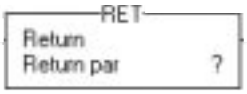
功能块



该指令操作数与梯形图SBR指令的操作数相同。

RET操作数：

梯形图



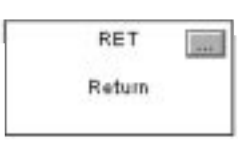
操作数：	数据类型：	格式：	说明：
Return	BOOL	立即数	本例程中的数据，将该数据复制到相应的
Parameter	SINT	标签	JSR指令的返回参数中
返回参数	INT	数组标签	
	DINT		
	REAL		
	结构体		

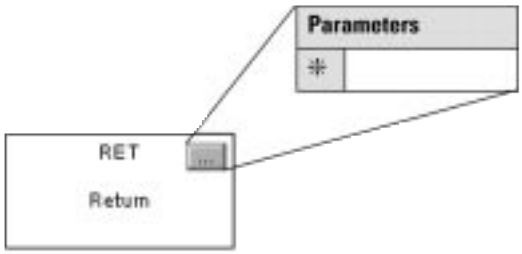
结构文本

```
RET(ReturnPar);
```

该指令操作数与梯形图RET指令的操作数相同。

功能块





该指令操作数与梯形图RET指令的操作数相同。

说明： JSR指令启动执行指定子例程：

- 子例程执行一次
- 子例程执行后，例程逻辑返回到包含调用子例程的JSR指令的例程中。

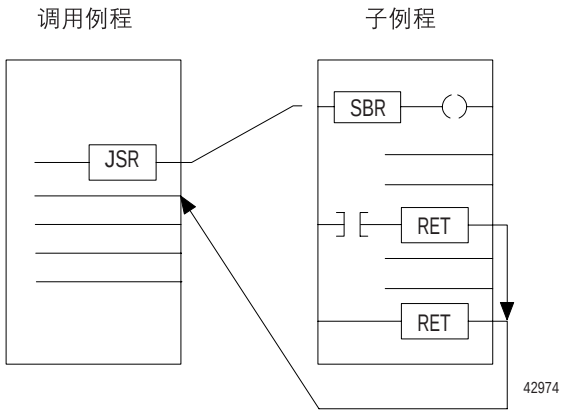
要编程跳转到子例程中，按照如下规则：

重要事项

不能使用JSR指令调用(执行)主例程。

- JSR指令可以放置在主例程中，也可以在其它任何例程中。
- 如果用户使用JSR指令调用主例程，然后再将RET指令放入主例程，则会发生主要故障(故障类型4，故障代码31)。

下表描述了指令如何执行。



JSR

1. 如果要将数据复制到子例程的标签中，则键入一个输入参数。
2. 如果要将子例程的结果复制到该例程的标签中，则键入一个返回参数。
3. 可以根据需要输入多个输入和返回参数。

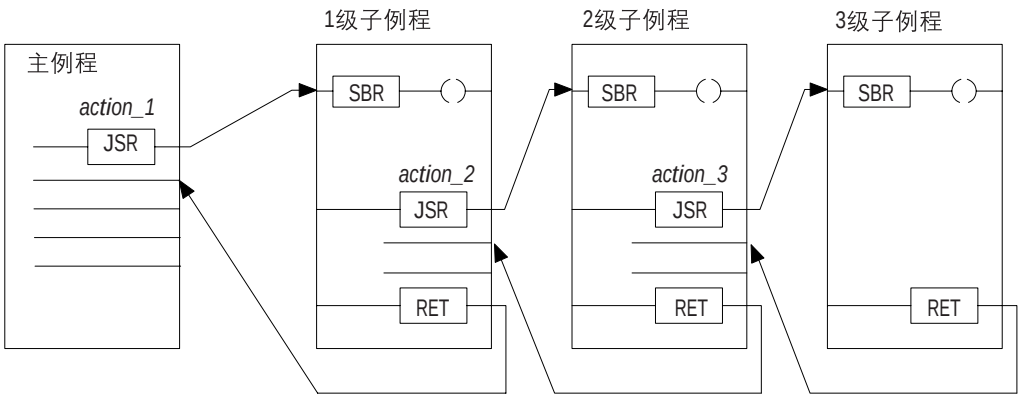
SBR

1. 如果JSR指令有输入参数，则需要键入一条SBR指令。
2. SBR指令必须是例程中的首条指令。
3. 对于JSR指令中的每个输入参数，都要键入一个要复制数据的标签。

RET

1. 如果JSR指令有返回参数，则需要键入一条RET指令。
2. RET指令必须是例程中最后一条指令。
3. 对于JSR指令的每个返回参数，都要输入一个返回参数将数据发送到JSR指令。
4. 在梯形图例程中，如果需要可以输入多个RET指令以根据不用的输入条件退出子例程。(功能块编程只允许用一条RET指令。)

用户可以嵌套的子例程级数或可传送或返回的参数量没有任何限值，只与控制器的内存容量有关。



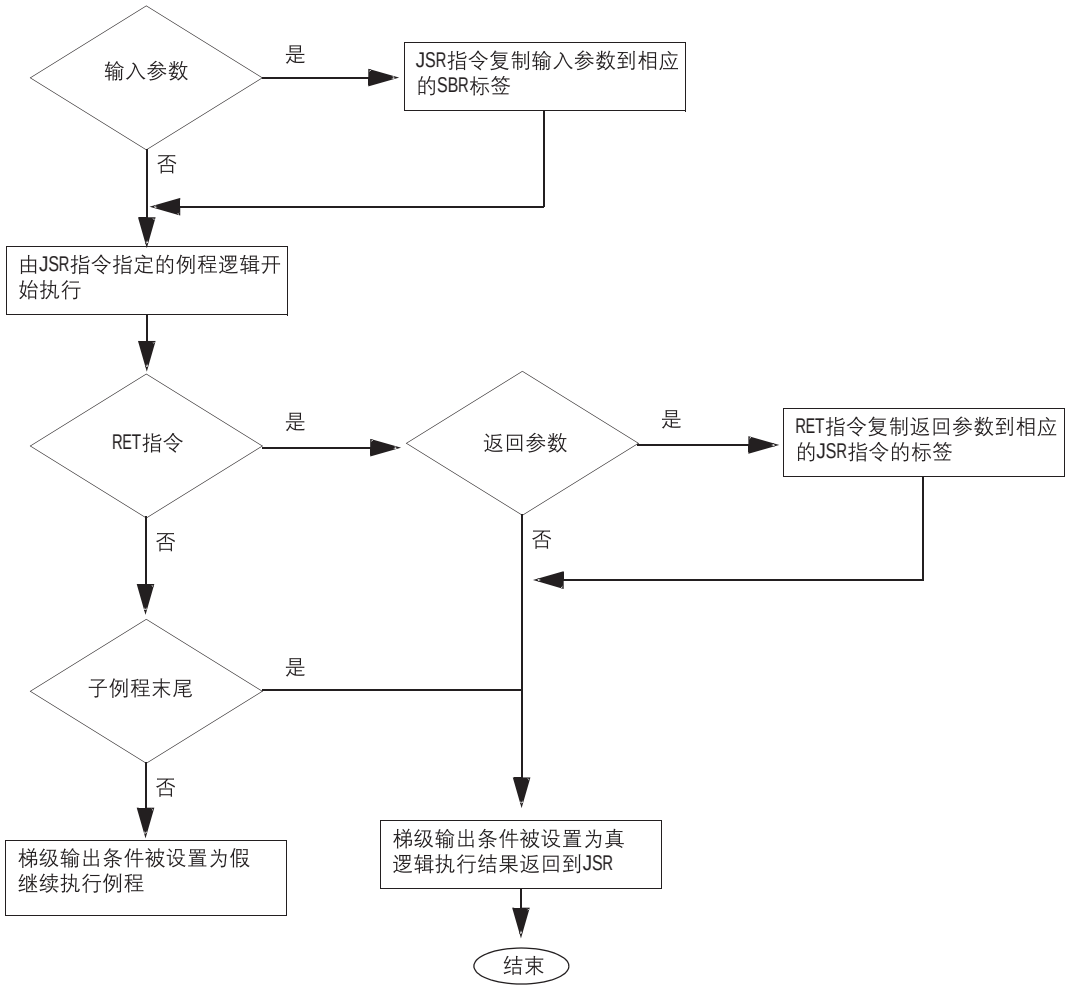
算术状态标志：影响算术状态标志

故障条件：

发生主要故障的条件：	故障类型：	故障代码：
JSR指令的输入参数数量少于SBR指令	4	31
JSR指令跳转到故障例程	4或用户提供	0或用户提供
RET指令的返回参数数量少于JSR指令	4	31
主例程中包含一条RET指令	4	31

执行：


条件：	梯形图动作：	结构文本动作：
预扫描	控制器忽略梯级条件执行全部子例程。为了保证子例程中的全部梯级都能被预扫描，控制器忽略RET指令。(也就是说，遇到RET指令不退出子例程。) • 6.x 或更早版本，传递输入和返回参数。 • 7.x 及以后版本，不传递输入和返回参数。 对于相同子例程，如果存在递归调用，子例程只在第一次调用时被预扫描。如果对于相同子例程，存在多次调用(非递归)，则子例程每次都被预扫描。 梯级输出条件被设置为假(仅梯形图)。	
JSR的梯级输入条件为假	子例程不执行。 子例程的输出保持其上次状态。 梯级输出条件被设置为假。	不影响
梯级输入条件为真	指令执行。 梯级输出条件被设置为真。	不影响
EnableIn置位	不影响	EnableIn一直被置位。 指令执行。

条件：	梯形图动作：	结构文本动作：
指令执行		

后期扫描	与上面所述预扫描动作相同。	与上面所述预扫描动作相同。

功能块

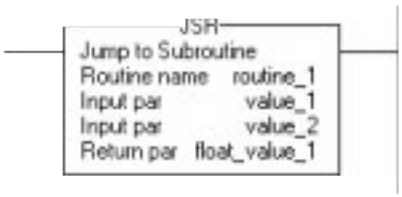
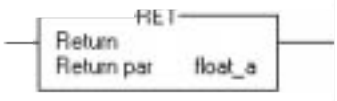
条件：	动作：
预扫描	无动作。
首次执行指令	无动作。
首次运行指令	无动作。
正常执行	1、 如果例程中包含SBR指令，控制器首先执行SBR指令。 2、 控制器锁定IREFs中所有数据。 3、 控制器按照其接线所确定的顺序执行其它功能块。这其中包含其它JSR指令。 4、 控制器将输出字写入OREFs。 5、 如果例程中包含RET指令，控制器最后执行RET指令。
后期扫描	调用子程序。 如果例程类型为SFC，该例程被初始化，与预扫描动作相同。

例1：JSR指令将value_1 与 value_2 中的数据传递到routine_1中。

SBR指令从JSR指令接收value_1 和value_2 并将这些值分别复制到value_a和value_b。然后继续执行该例程中的其它逻辑。

RET指令将float_a发送到JSR指令。JSR指令接收float_a 并将数据复制到float_value_1中。例程继续执行JSR后面的指令。

梯形图

例程：	编程：
主例程	
子例程	
	其它梯级
	

结构文本

例程：	编程：
主例程	JSR(routine_1,2,value_1,value_2,float_value_1);
子例程	SBR(value_a,value_b); <statements>; RET(float_a);

例 2：

梯形图

主例程

当abc处于导通状态时，执行subroutine_1 ，计算cookies数量，并在cookies_1中存入数值。

将cookies_1的数值加到cookies_2 并将计算结果存入total_cookies中。

Subroutine_1

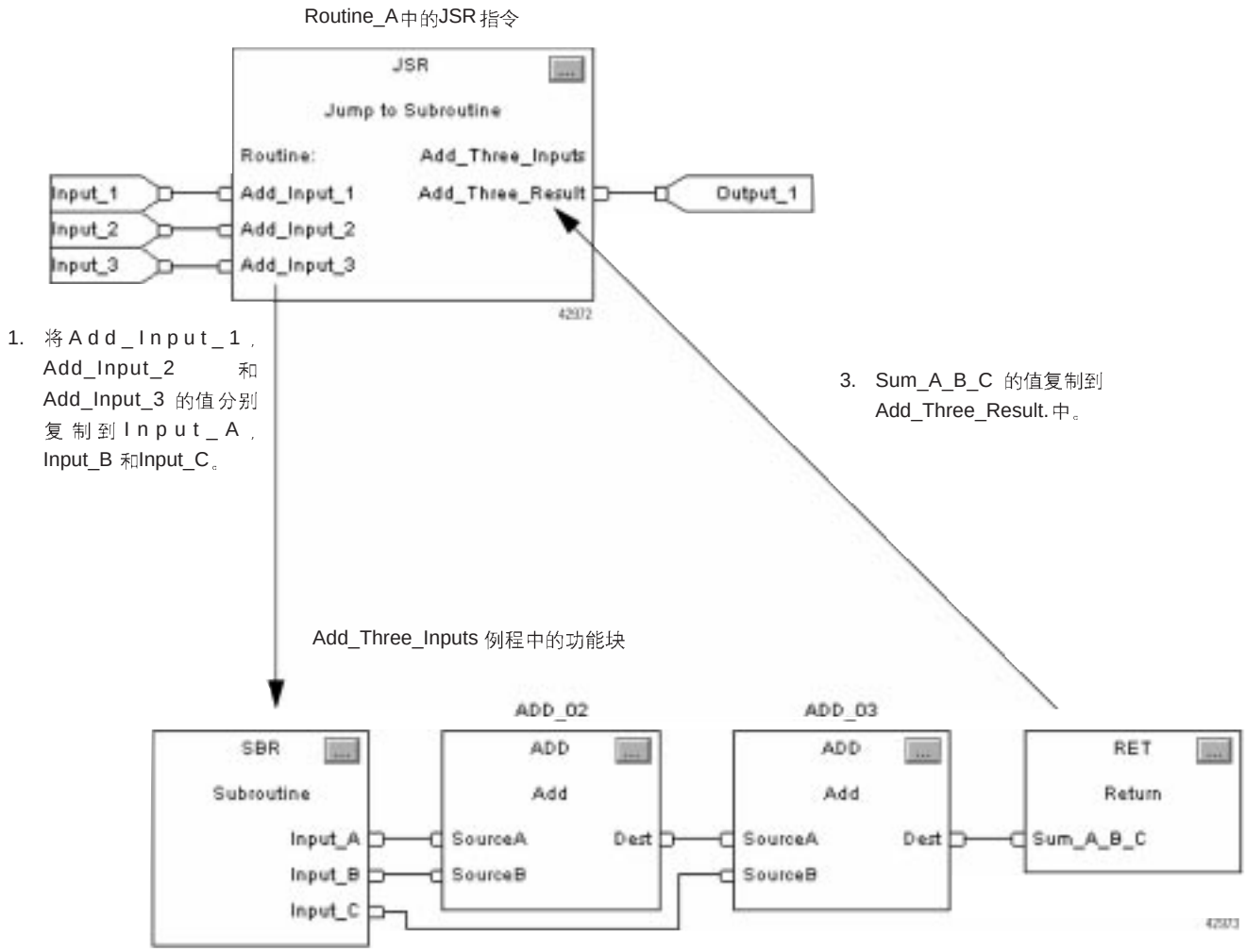
如果def处于导通状态，则RET指令将value_1返回到JSR的cookies_1 参数中，而且不再扫描其余子例程。

如果def处于断开状态 (在前面梯级) 且ghi 处于导通状态，则RET指令将value_2返回到JSR指令cookies_1参数中，且不再扫描其余子例程。

如果def和ghi均处于断开状态 (在前面梯级)，则RET将value_3 返回到JSR指令cookies_1参数中。

例3：

功能块



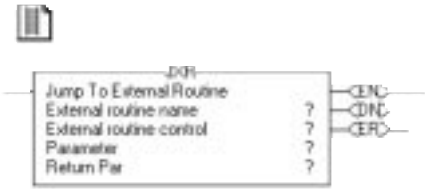
2. ADD指令将Input_A， Input_B，与 Input_C 相加并将结果存入Sum_A_B_C. 中

跳转到外部例程(JXR)

JXR指令用于执行外部例程。只有SoftLogix5800控制器支持该指令。

操作数：

梯形图



操作数：	数据类型：	格式：	说明：
External routine name外部例程名称	ROUTINE	名称	要执行的外部例程
External routine control外部例程控制	EXT_ROUNTINE_CONTROL	标签	控制结构体(查看下一页)
Parameter参数	BOOL SINT INT DINT REAL 结构体	立即数 标签 数组标签	该例程中数据，用户需将该数据复制到外部例程变量中 • 参数可选 • 如果需要，输入多个参数 • 用户最多有10个参数
Return parameter返回参数	BOOL SINT INT DINT REAL	标签	该例程中标签，用户将外部例程结果复制到该标签中 • 返回参数可选 • 用户仅有一个返回参数

EXT_ROUTINE_CONTROL 结构体

助记符	数据类型：	说明：	执行：
ErroCode 供错误代码。	SINT	如果错误产生，该值标识错误。有效值为0-255。	无预定义错误代码。外部例程的开发人员必须提
NumParams	SINT	该值指示与此指令关联的参数数量。	仅显示－该信息由指令输入衍生。
ParameterDefs	EXT_ROUTINE_ PARAMETERS[10]	该数组包含对进入外部例程参数的定义。该指令最多执行10个参数。	仅显示－该信息由指令输入衍生。
ReturnParamDef	EXT_ROUTIN_ PARAMETERS	该值包含从外部例程返回参数的定义。 仅有1个返回参数。	仅显示－该信息由指令输入衍生。
EN	BOOL	该位置1时，使能位指示JXR指令被使能。	外部例程置位该位。
ReturnValue	BOOL	如果置位，该位指示已为指令输入一个返回参数。如果清零，该位指示无返回参数为指令输入。	仅显示－该信息由指令输入衍生。
DN	BOOL	外部例程完成一次执行时，完成位置位。	外部例程置位该位。
ER	BOOL	如果错误出现，错误位置位。指令停止执行直至例程清除错误位。	外部例程置位该位。
FirstScan	BOOL	该位标识是否为控制器切换到运行模式后的首次扫描。根据需要使用FirstScan来初始化外部例程。	控制器设置该位反映扫描状态。
EnableOut	BOOL	使能输出。	外部例程置位该位。
EnableIn	BOOL	使能输入。	控制器置位该位来反映梯级输入条件。不管梯级条件为何，该指令执行。外部例程开发人员必须监视该状态并执行相应动作。
User1 User0	BOOL BOOL	该位用户可用。控制器无法初始化该位。	外部例程或用户例程可以置位该位。
ScanType1 ScanType0	BOOL BOOL	该位指示当前扫描类型： 位值： 扫描类型： 00 正常 01 预扫描 10 后期扫描 (不能用于延迟梯形图例程)	控制器置位该位来反映扫描状态。

说明： 在用户项目的梯形图例程中使用跳转到外部例程(JXR)指令来调用外部程序。JXR指令支持多个参数，以便用户在梯形图和外部程序间调用。

JXR指令与跳转到子例程(JSR)指令相同。JXR指令开始执行指定外部例程：

- 外部例程执行一次。
- 外部例程执行结束后，逻辑执行返回到包含JXR指令的梯级。

算术状态符： 算术状态符：算术状态符不受影响。

故障条件： 故障条件：

出现主要故障，如果：	故障类型：	故障代码：
• 外部梯级DLL出现异常 • DLL无法加载 • DLL中无法找到输入点	4	88

执行： 根据DLL的执行情况，JXR可能同步或异步执行。DLL中代码也决定了如何响应扫描状态、梯级条件输入状态和梯级条件输出状态。

关于使用JXR指令和创建外部例程，查阅SoftLogix5800系统用户手册，出版号1789-UM002。

暂停 (TND)

TND 可以作为一个分界线。

操作数：

梯形图操作数

无

结构文本

无

用户必须在指令助记符后面输入圆括号()，即使该指令没有操作数。

说明： 当指令被使能时，TND指令使程序逻辑只执行到该指令。

当指令被使能时，TND指令作为例程的结尾。控制器扫描到TND指令时，控制器就转到当前例程的末尾处。如果TND指令在子例程中，则控制返回到调用它的例程。如果TND指令在主例程中，则控制返回到当前任务中下个程序。

算术状态标志： 不影响

故障条件： 无

执行：

条件：	梯形图动作：	结构文本动作：
预扫描	梯级输出条件被设置为假。	无动作。
梯级输入条件为假	梯级输出条件被设置为假。	无影响
梯级输入条件为真	指令执行。 梯级输出条件被设置为真。	无影响
EnableIn置位	不影响	EnableIn一直置位。 指令执行。
指令执行	当前例程结束。	当前例程结束。
后期扫描	梯级输出条件被设置为假。	无动作发生。

举例：在调试例程或处理故障时，用户可以用TND指令使例程逻辑执行到某一特定点。然后在程序内进一步移动到TND指令到用户调试程序的新部分。

如果TND 指令被使能，则控制器停止扫描当前例程。

梯形图

结构文本

TND()；

主控复位指令 (MCR)

MCR指令成对使用，用来创建一个区域，可以使用MCR指令使该区域内所有梯级无效。

操作数：

梯形图

无

说明： 当MCR区域被使能时，在MCR区域内的梯级的为真或为假条件被正常扫描。当MCR区域禁止，控制器仍然扫描MCR区域中的梯级，但是由于此区域中非保持性输出均被禁止，减少了扫描时间。禁止的MCR区域中所有指令的梯级输入条件为假。

用户在MCR区域内编程时，应注意：

- 必须以一条无条件MCR指令结束该区域。
- MCR区域不能嵌套。
- 不要跳转到MCR区域。如果该区域为假，则跳转到该区域会激活从跳入点到区域结束部分的逻辑程序。
- 如果MCR区域持续到例程的末尾，就不必在区域的末尾再用MCR指令。

不能将MCR指令用作提供急停功能的硬件主控继电器的代用品。用户仍然需要安装硬件主控继电器，以提供紧急断开I/O电源功能。

注意：一定不能重叠或嵌套MCR区域。每个MCR区域必须是完整的而且是独立的。 如果重叠或嵌套MCR区域会导致不可预料的机器运行，因此可能会造成设备损坏或人身伤害。

应将关键操作放在MCR区域的外面，如果在MCR区域起动诸如计时器之类的指令，则当该区域被禁止时指令停止执行而且计时器被清零。

算术状态标志： 不影响

故障条件： 无

执行：

条件：	梯形图动作：
预扫描	梯级输出条件被设置为假。
梯级输入条件为假	梯级输出条件被设置为假。 仍然扫描该区域中的指令，但是梯级输入条件被设置为假，而且该区域中的非保持输出被禁止。
梯级输入条件为真	梯级输出条件被设置为真。 正常扫描该区域中的指令。
后期扫描	梯级输出条件被设置为假。

举例：当第一条MCR指令被使能时(input_1， input_2 和 input_3 是置位状态)，控制器执行MCR区域内的梯级(在两条MCR指令之间的)并根据输入条件置位或清零输出。

当第一条MCR指令被禁止时(input_1， input_2 和 input_3 是清零状态)，控制器执行MCR区域的梯级(在两条MCR指令之间)，但是此区域的所有梯级输入条件都变为假，与输入条件无关。

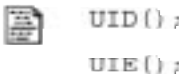
禁止用户中断 (UID)
使能用户中断(UIE)

一起使用UID和UIE指令可以防止一些关键梯级被其它任务中断。

操作数：

梯形图

无



结构文本

无

用户必须在指令助记符后输入圆括号()，即使该指令无操作数。

说明： 当梯级输入条件为真时：

- UID 指令可以防止更高优先级的任务中断当前任务，但是不禁止故障例程或控制器故障处理任务的执行。
- UIE 指令允许其它任务中断当前任务。

为了防止一连串梯级被中断：

1. 尽量减少不希望被中断的梯级数。如果禁止中断的时间过长可能会导致通讯丢失。
2. 在用户不希望中断的首个梯级前添加带有UID指令的梯级。
3. 在用户不希望中断的一系列梯级的最后梯级后面添加带有UIE指令的梯级。
4. 如果需要，可以嵌套几对UID/UIE指令。

算术状态标志： 不影响

故障条件： 无

执行：

条件：	梯形图动作：	结构文本动作：
预扫描	梯级输出条件被设置为假。	无动作
梯级输入条件为假	梯级输出条件被设置为假。	无影响
梯级输入条件为真	UID指令防止例程被高优先级的任务中断。 UIE指令允许例程被高优先级的任务中断。 梯级输出条件被设置为真。	无影响
输入使能置位	无影响	输入使能一直置位。 执行指令。
执行指令	UID指令防止更高优先级任务中断。 UIE指令使能更高优先级任务中断。	
后期扫描	梯级输出条件设置为假。	无动作执行。

举例：错误产生时 (error_bit为导通状态)，FSC 指令根据严重错误列表检测错误代码。如果FSC指令发现了错误(error_check.FD是导通状态)，则发出报警指示。UID和UIE指令防止其它任何任务中断错误检测和报警任务的执行。

梯形图

结构文本

```
UID();
<statements>
UIE();
```

恒假指令 (AFI)

AFI指令将梯级输出条件设为假。

操作数：

梯形图

无

说明：AFI指令设置所在梯级的输出条件为假。

算术状态标志：不影响

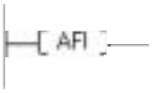
故障条件：无

执行：

条件：	梯形图动作：
预扫描：	梯级输出条件被设置为假。
梯级输入条件为假	梯级输出条件被设置为假。
梯级输入条件为真	梯级输出条件被设置为假。
后期扫描	梯级输出条件被设置为假。

举例：在调试例程时，可以用AFI指令暂时禁止梯级。

当指令被使能时，AFI指令禁止其所在梯级上的所有指令。



空操作 (NOP)

AFI指令的功能相当于占位符。

操作数：

梯形图

无

说明：用户可以将NOP指令放置在梯级的任何地方。当使能NOP指令时，执行空操作。禁止NOP指令时，也执行空操作。

算术状态标志： 不影响

故障条件： 无

执行：

条件：	动作：
预扫描：	梯级输出条件被设置为假。
梯级输入条件为假	梯级输出条件被设置为假。
梯级输入条件为真	梯级输出条件被设置为真。
后期扫描	梯级输出条件被设置为假。

举例： 当用户将NOP指令置于分支上，它的作用是定位一个无条件分支。

NOP指令旁路XIC指令使输出使能。

传送结束(EOT)

EOT指令将一个布尔型状态值返回至SFC传送。

操作数：

梯形图

操作数：	类型：	格式：	说明：
数据位	BOOL	标签	转换状态 (0表示执行，1表示完成)

结构文本

```
EOT(data_bit);
```

该指令操作数与梯形图中EOT指令相同。

说明： 因为EOT指令返回一个布尔型状态值，多个SFC例程可以共享同一带有EOT指令的梯级。如果所调用例程并未传送，EOT指令作为一个TND指令(查看页10-17)。

Logix控制器中执行EOT指令与PLC-5控制器有所区别。在PLC-5控制器中，EOT指令没有参数。与此对应，PLC-5中EOT指令返回梯级条件作为其状态。在Logix控制器中，返回参数回送转换状态，而Logix编程语言中梯级条件无法获取。

算术状态标志： 不影响

故障条件： 无

执行：

条件：	梯形图动作：	结构文本动作：
预扫描	梯级输出条件被设置为假。	无动作发生。
梯级输入条件为假	梯级输出条件被设置为假。	无影响
梯级输入条件为真	梯级输出条件被设置为真。	无影响
输入使能置位	无影响	输入使能一直置位。 该指令执行。
指令执行	指令将数据位值返回到调用例程。	
后期扫描	梯级输出条件被设置为假。	无动作发生。

当limit_switch1和interlock_1均被置位，设置状态。在timer_1完成后，EOT将状态值返回到调用例程中。

梯形图

结构文本

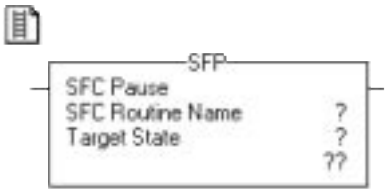
```
state := limit_switch1 AND interlock_1;


IF timer_1.DN THEN

    EOT(state);

END_IF;
```

SFC暂停(SFP)

SFP指令用于暂停一个SFC例程。

操作数：

梯形图



操作数：	类型：	格式：	说明：
SFC Routine Name例程名称	ROUTINE	名称	要暂停的SFC例程
TargetState目标状态	DINT	立即数 标签	选择其中一个： 执行(或输入0) 暂停(或输入1)

结构文本

该指令操作数与梯形图SFP指令操作数相同。

说明： SFP指令允许用户暂停一个执行的SFC例程。如果一个SFC例程处于暂停状态，再次使用SFP指令改变状态并恢复执行例程。

同时，在使用一条SFR指令(查看页10-29)来复位SFC例程后，使用SFP指令恢复SFC执行。

算术状态标志：不影响

故障状态：

将产生主要故障，如果：	故障类型：	故障代码：
例程类型不是SFC例程	4	85

执行：

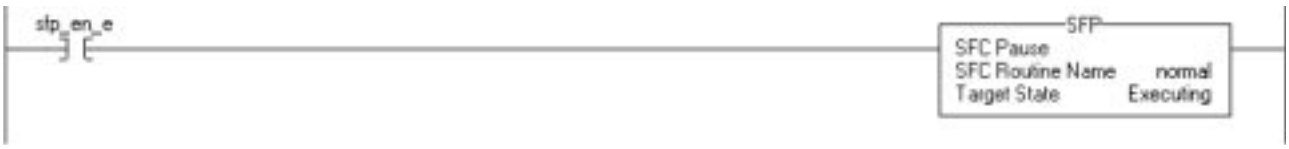
条件：	梯形图动作：	结构文本动作：
预扫描	梯级输出条件被设置为假。	无动作发生。
梯级输入条件为假	梯级输出条件被设置为假。	无影响
梯级输入条件为真	梯级输出条件被设置为真。	无影响
输入使能置位	无影响	输入使能一直置位。 该指令执行。
指令执行	指令暂停或返回执行指定的SFC例程。	
后期扫描	梯级输出条件被设置为假。	无动作发生。

举例：如果sfc_en_p被置位，暂停名称为normal的SFC例程。当sfc_en_p被置位重启SFC。

梯形图

暂停SFC例程。

恢复执行SFC例程。



结构文本

暂停SFC例程：

```
IF (sfp_en_p) THEN
    SFP(normal,paused);
    sfp_en_p := 0;
END_IF;
```

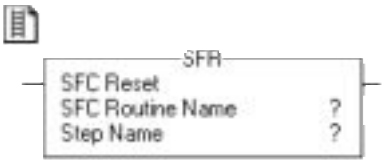
继续执行SFC例程：

```
IF (sfp_en_e) THEN
    SFP(normal,executing);
    sfp_en_e := 0;
END_IF;
```

SFC 复位(SFR)

SFR指令将SFC例程复位到特定步执行。

操作数：



梯形图操作数

操作数：	类型：	格式：	说明：
SFC Routine Name例程名称	ROUTINE	名称	要复位的SFC例程
Step Name步名	SFC_STEP	标签	恢复操作的目标步

结构文本

```
SFR(SFCRoutineName,StepName);
```

该指令操作数与梯形图SFR指令操作数相同。

说明：当SFR指令被使能：

- 在指定的SFC例程中，所有存储的动作停止执行(复位)。
- SFC在特定步开始执行。

如果目标步为0，则顺序功能表被复位至其初始步。

SFR指令在Logix控制中的执行方式与PLC-5控制器略有不同。在PLC-5控制器中，当梯级条件为真时，SFR指令执行。复位后，SFC仍暂停直至包含SFR指令的梯级为假。这会造成在复位后执行的动作出现延时。PLC-5的SFR指令的暂停/解锁暂停的特性从梯级条件上被减弱，并移入SFP指令。

算术状态标志：不影响

故障条件：

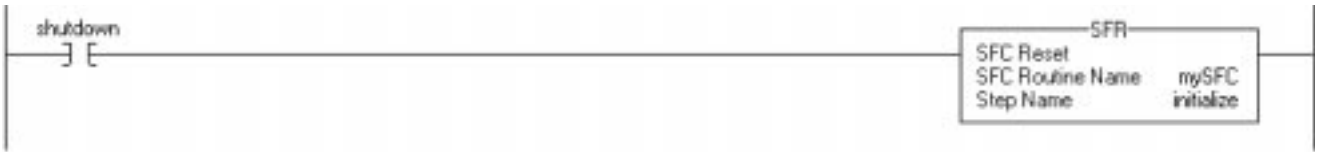
将产生主要故障，如果：	故障类型：	故障代码：
例程类型不是SFC例程	4	85
指定的目标步在SFC例程中不存在	4	89

执行：

条件：	梯形图动作：	结构文本动作：
预扫描	梯级输出条件被设置为假。	无动作发生。
梯级输入条件为假	梯级输出条件被设置为假。	无影响
梯级输入条件为真	指令执行。 梯级输出条件被设置为真。	无影响
输入使能置位	无影响	输入使能一直置位。 该指令执行。
指令执行	指令复位指定的SFC例程。	指令复位指定的SFC例程。
后期扫描	梯级输出条件被设置为假。	无动作发生。

举例：如果指定条件发生(shutdown被置位)，在initialize步重启SFC。

梯形图



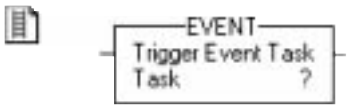
结构文本

```
IF shutdown THEN
        SFR(mySFC,initialize);
END_IF;
```

触发事件型任务(EVENT)

EVENT指令触发执行一个事件型任务。

操作数：



梯形图

操作数：	类型：	格式：	说明：
Task任务	TASK	名称	要执行的事件型任务 该指令允许用户选择其它类型的任务，但无法执行。

结构文本

该指令操作数与梯形图的EVENT指令操作数相同。

说明：使用EVENT指令来编程执行一个事件型任务：

- 该指令每次执行时触发指定的事件型任务。
- 用户再次触发事件型任务前，确保为执行完该任务留有足够的时间。否则，将发生执行重叠故障。
- 如果用户在执行一条EVENT指令时，另一个事件型任务已经执行，控制器将重叠计数器的值增加，但不触发事件型任务。

编程确定EVENT指令是否触发任务

要确定EVENT指令是否触发一个事件型任务，可使用获取系统值(GSV)指令来监视任务的状态属性值。

表10.1 TASK对象的状态属性

属性：	数据类型：	指令：	说明：	
状态	DINT	GSV SSV	提供该任务的状态信息。一旦控制器将该位置位，用户必须手动清除该位以确定是否产生另一该类型的故障。	
			要确定是否：	检查该位为：
			一个EVENT指令触发执行任务(仅事件型任务)。	0
			超时触发任务(仅事件型任务)。	1
			该任务出现重叠	2

一旦状态属性位被置1，控制器则无法将其清零。

- 要使用某位获取新的状态信息，用户必须手动清除该位。
- 使用设置系统值(SSV)将属性值设为不同值。

算术状态标志：不影响

故障条件：无

执行：

条件：	梯形图动作：	结构文本动作：
预扫描	梯级输出条件被设置为假。	无动作发生。
梯级输入条件为假	梯级输出条件被设置为假。	无影响
梯级输入条件为真	指令执行。 梯级输出条件被设置为真。	无影响
输入使能置位	无影响	输入使能一直置位。 该指令执行。
指令执行	指令触发执行指定的事件型任务。	
后期扫描	梯级输出条件被设置为假。	无动作发生。

举例1：控制器使用多个程序，但关闭流程相同。当程序检测到某一需要关闭程序的条件，便使用一个名为Shut_Down_Line的程序作用域标签启动关闭任务。每个程序中执行的逻辑如下所示：

如果Shut_Down_Line = 0(需要关闭的条件)那么

执行一次Shut_Down任务

梯形图

程序A

程序B

结构文本

程序A

```
IF Shut_Down_Line AND NOT Shut_Down_Line_One_Shot THEN
        EVENT (Shut_Down);
END_IF;
Shut_Down_Line_One_Shot := Shut_Down_Line;
```

程序B

```
IF Shut_Down_Line AND NOT Shut_Down_Line_One_Shot THEN
        EVENT (Shut_Down);
END_IF;
Shut_Down_Line_One_Shot := Shut_Down_Line;
```

举例2：下例中使用一条EVENT指令来启动事件型任务。(其它类型的事件触发事件型任务。)

连续型任务

如果 Initialize_Task_1 = 1 那么

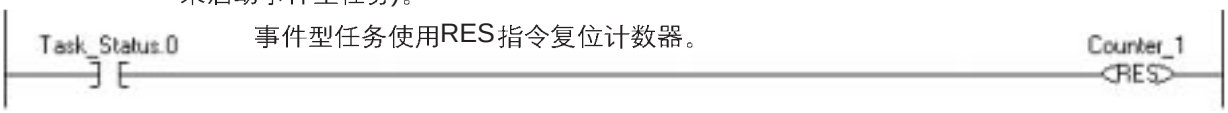
ONS指令限制执行EVENT指令为一次扫描。

EVENT指令触发执行Task_1任务(事件型任务)。

Task_1(事件型任务)

GSV指令设置Task_Status(DINT标签)为事件型任务状态属性。在实例名称(Instance Name)属性项中，THIS表示任务目的对象是指令所在任务。

如果Task_Status.0 = 1，那么EVENT指令触发事件型任务(例如，当连续型任务执行EVENT指令来启动事件型任务)。



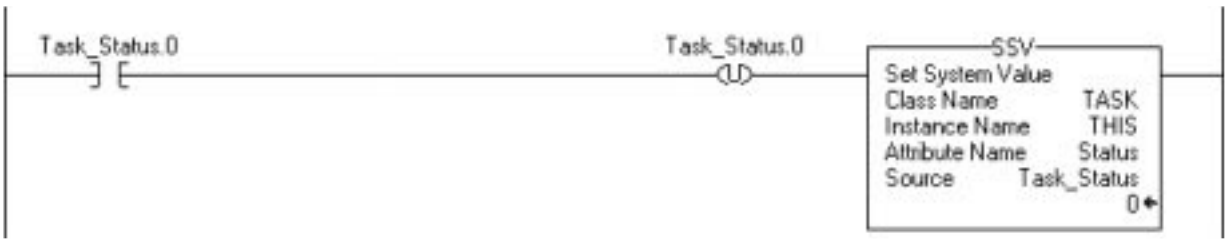
事件型任务使用RES指令复位计数器。

一旦状态属性位置位，控制器无法自动清除该位。若将某位用于新的状态信息，用户必须手动清除该位。

如果Task_Status.0=1，那么清除该位。

使用OTU指令将Task_Status.0=0.

使用SSV指令将THIS任务(Task_1)=Task_Status.此时可清除该位。



循环/中断指令
（FOR，FOR…DO，BRK，EXIT，RET）

简介

FOR指令用于重复调用子例程。BRK指令用于中断子例程的执行。

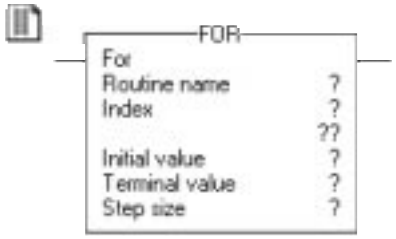
如果用户需要：	所用指令：	可用语言：	参见页码：
重复执行某一例程。	FOR	梯形图	11-2
	FOR…DO(1)	结构文本	
中止例程的重复执行。	BRK	梯形图	11-5
	EXIT(1)	结构文本	
返回到FOR指令。	RET	梯形图	11-6

循环(FOR)

FOR指令用于重复执行某一例程。

操作数：

梯形图：



操作数：	数据类型：	格式：	说明：
Routine name 例程名称	ROUTINE	例程名称	要执行的例程
Index 索引	DINT	标签	计数例程被执行的次数
Initial value 初始值	SINT INT DINT	立即数 标签	索引的起始值
Terminal value 中止值	SINT INT DINT	立即数 标签	使例程停止执行的值
Step size 步长	SINT INT DINT	立即数 标签	每次FOR 指令执行例程时加到索引值中的数

结构文本

```
FOR count:= initial value TO
final_value BY increment DO
    <statement>;
END_FOR;
```

说明：

重要事项

不能使用FOR指令调用(执行)主例程。

- FOR指令可以在主例程中，也可以在任何其它例程中。
- 如果用户使用FOR指令调用主例程，然后再将RET指令放入主例程中，将发生主要故障(类型4，代码31)

当指令被使能时，FOR 指令重复执行例程，直到索引值超过中止值。该指令不向例程传递参数。

每次FOR指令执行例程时，都将步长值加到索引值中。

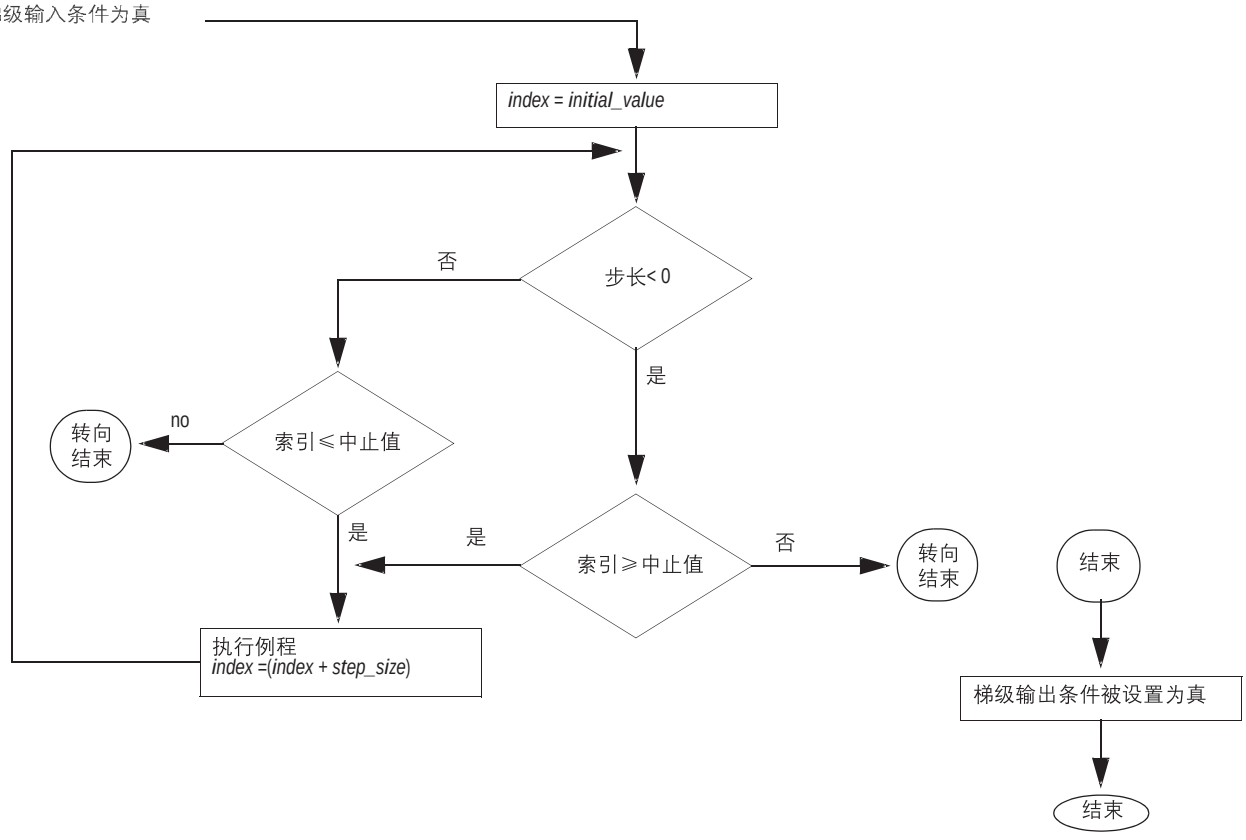
注意 在一次扫描中循环的次数不要太多。重复的次数过多可能会导致控制器的看门狗时间超时，这将导致主要故障。

算术状态标志：不影响

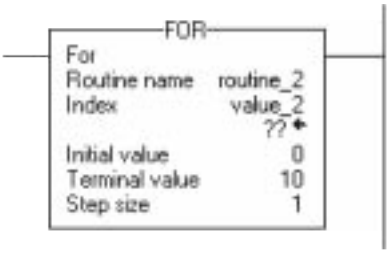
故障条件：

发生主要故障的条件：	故障类型：	故障代码：
主例程中有RET指令	4	31

执行：

条件：	梯形图动作：
预扫描	梯级输出条件被设置为假。 控制器执行一次子例程。 如果同一子例程中存在递归FOR指令，则只在第一次时预扫描子例程。如果同一子例程中存在多条FOR指令(非递归)，则每次都预扫描子例程。
梯级输入条件为假	梯级输出条件被设置为假。
梯级输入条件为真	
后期扫描	梯级输出条件被设置为假。

举例: 当指令被使能时，FOR指令重复执行routine_2 ，并且每次都使value_2 中的值加1。如果value_2 中的值>10 或BRK 指令被使能，则FOR 指令不再执行routine_2。



中断(BRK)

BRK指令用于中断FOR指令调用例程的执行。

操作数：

梯形图

无

结构文本

在循环结构体中使用EXIT语句。关于结构文本编程中结构体的信息，参见结构文本编程。

说明：当指令被使能时，BRK指令使控制器退出正在执行的例程并返回执行FOR指令下一条指令。

如果是嵌套的FOR指令，则BRK指令返回到最里层的FOR指令。

算术状态标志：不影响

故障条件：无

执行：

条件：	梯形图动作：
预扫描	梯级输出条件被设置为假。
梯级输入条件为假	梯级输出条件被设置为假。
梯级输入条件为真	梯级输出条件被设置为真。 程序执行返回到FOR指令下面的指令。
后期扫描	梯级输出条件被设置为假。

举例：当指令被使能时，BRK指令使控制器停止执行当前例程并返回到执行调用FOR指令的下一条指令。

返回(RET)

RET指令返回到调用的FOR指令。

操作数：

梯形图

无

说明：

重要事项 不要将RET指令放入主例程中。如果将RET指令放入主例程中，将会发生主要故障(类型4，代码31)。

当指令被使能时，RET指令返回到FOR指令。FOR指令以步长为单位增加索引值并再次执行子例程。如果索引值超过中止值，则完成FOR指令的执行，转去执行FOR指令后面的指令。

因为FOR指令不用参数，FOR指令忽略在RET指令中输入的任何参数。

用户也可以用一条TND指令来结束子例程的执行。

算术状态标志： 不影响

故障条件：

发生主要故障的条件：	故障类型：	故障代码：
主例程中有RET指令	4	31

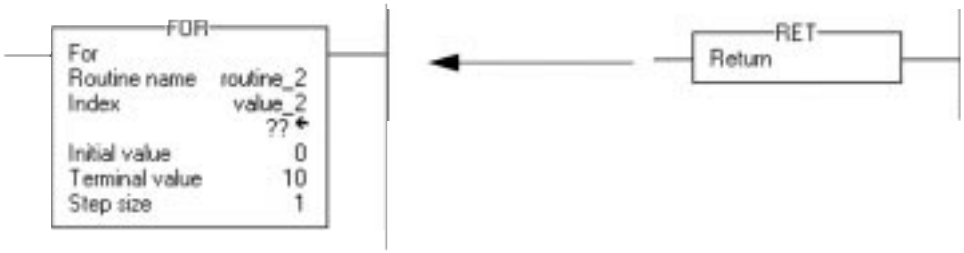
执行：

条件：	梯形图动作：
预扫描	梯级输出条件被设置为假。
梯级输入条件为假	梯级输出条件被设置为假。
梯级输入条件为真	返回指定参数到调用例程。 梯级输出条件被设置为真。
后期扫描	梯级输出条件被设置为假。

举例：FOR指令重复执行routine_2 并在每次都使value_2 中的值加1。如果value_2 中的值>10 或BRK 指令被使能，则FOR 指令不再执行routine_2。

RET指令返回到调用的FOR指令。FOR指令或以步长为单位增加索引值并再次执行子例程或如果索引值超过中止值，则完成FOR指令的执行，转去执行FOR指令下一条指令。

调用例程　　子例程



注释：

专用指令
(FBC，DDT，DTR，PID)

简介

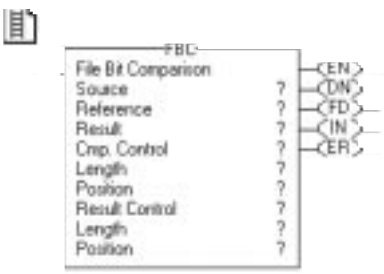
专用指令用来完成一些特殊应用。

如果用户需要：	所用指令：	可用语言：	参见页码：
将数据与一已知参考数据进行比较并记录不匹配的位置	FBC	梯形图	12-2
将数据与一已知参考数据进行比较，记录不匹配的位置，并更新参考数据使之与源数据匹配。	DDT	梯形图	12-10
将通过屏蔽字的源数据与参考数据比较。然后用源数据覆盖参考数据用于下一次比较。	DTR	梯形图	12-18
控制PID回路。	PID	梯形图 结构文本	12-21

文件位比较 (FBC)

FBC指令用于比较源数组各位与参考数组中各位。

操作数：操作数：



梯形图

操作数：	数据类型：	格式：	说明：
Source 源数据	DINT	数组标签	要与参考数组进行比较的数组 不能在下标处用CONTROL.POS
Reference 参考数据	DINT	数组标签	要与源数组进行比较的数组 不能在下标处用CONTROL.POS
Result 结果	DINT	数组标签	存储比较结果的数组 不能在下标处用CONTROL.POS
Cmp control 比较控制结构体	CONTROL	结构体	比较操作的控制结构体
Length 长度	DINT	立即数	参与比较的位数量
Position 位置	DINT	立即数	源数组中的当前位置 初始值一般是0
Result control 结果控制结构体	CONTROL	结构体	运算结果的控制结构体
Length 长度	DINT	立即数	存储在结果数组中不匹配位置的数量
Position 位置	DINT	立即数	结果数组的当前位置 初始值一般是0

注意：比较控制结构体和结果控制结构体要用不同的标签。如果二者使用相同的标签会发生不可预料的操作，可能会导致设备损坏和(或)人员伤亡。

COMPARE结构体

助记符:	数据类型:	说明
.EN	BOOL	使能位表明FBC指令被使能。
.DN	BOOL	当FBC指令完成源与参考数组最后一位的比较时置位完成位。
.FD	BOOL	每次FBC指令记录一次不匹配(一次比较一位操作模式)或记录全部不匹配(每次扫描比较全部位操作模式)后置位发现位。
.IN	BOOL	禁止位表示FBC指令的搜索模式 0=整体模式 1=一次比较一个不匹配模式
.ER	BOOL	如果出现比较位置值.POS < 0、比较长度值.LEN < 0、结果位置值.POS < 0或长度值.LEN < 0，则置位错误位。在程序清零错误位.ER位之前不执行指令。
.LEN	DINT	长度值表示参与比较的位的数量。
.POS	DINT	位置值表示当前位的位置。

RESULT结构体

助记符:	数据类型:	说明:
.DN	BOOL	当结果数组已满时，置位完成位。
.LEN	DINT	长度值表示存储在结果数组中位的数量。
.POS	DINT	位置值表示结果数组中当前位的位置。

说明: 如果FBC指令被使能，就将源数组中的位与参考数组中的位进行比较，在结果数组中记录每个不匹配位的位号。

FBC 指令对存储器内连续区域进行操作。

DDT与FBC指令的区别是DDT指令每次查找一个不匹配位，该指令改变参考位的值使其与源中的位匹配。而FBC指令不改变参考位。

选择搜索模式

如果要检测：	选择如下模式：
一次扫描记录一个不匹配位	置位比较控制结构体的禁止位.IN。 每次梯级输入条件由假变真时，FBC指令搜索源和参考数组中的下一个不匹配位。 发现不匹配后，置位发现位.FD，记录不匹配位的位置并停止执行。
一次扫描记录全部不匹配位	清零比较控制结构体的禁止位.IN。 每次梯级输入条件由假变真时，FBC指令搜索源和参考数组中全部不匹配位。

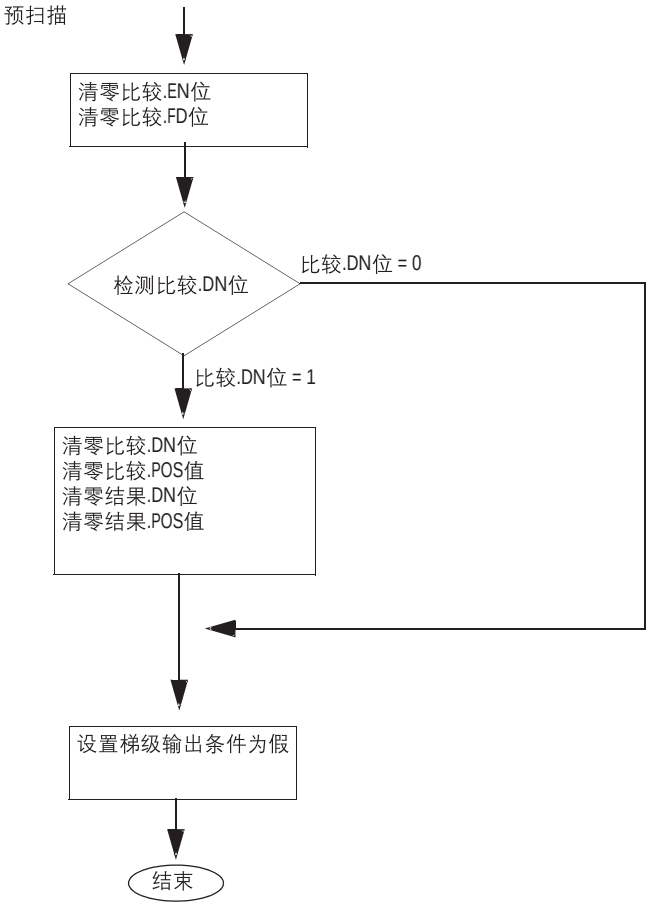
算术状态标志：不影响

故障条件：

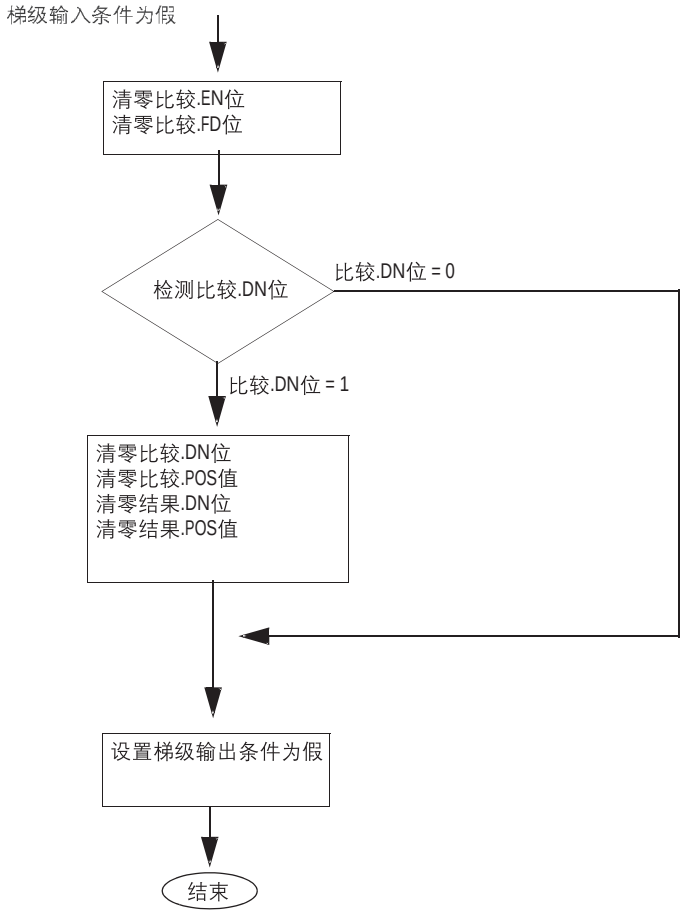
发生主要故障的条件：	故障类型：	故障代码：
结果.POS >结果数组的长度	4	20

执行：

条件：	梯形图动作：


预扫描
清零比较.EN位
清零比较.FD位
检测比较.DN位
比较.DN位 = 0
比较.DN位 = 1
清零比较.DN位
清零比较.POS值
清零结果.DN位
清零结果.POS值
设置梯级输出条件为假
结束

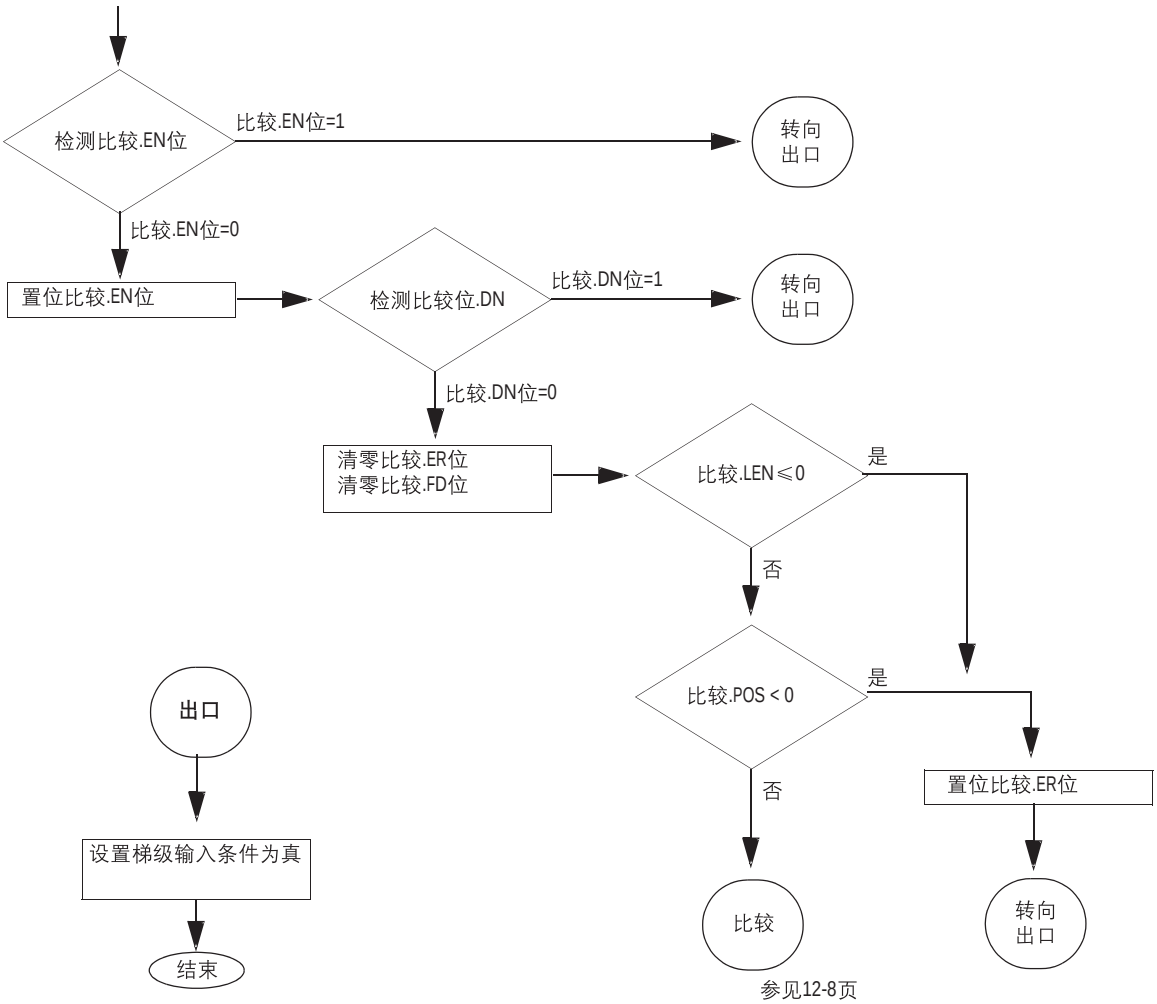
条件：	梯形图动作：


梯级输入条件为假
清零比较.EN位
清零比较.FD位
检测比较.DN位
比较.DN位 = 0
比较.DN位 = 1
清零比较.DN位
清零比较.POS值
清零结果.DN位
清零结果.POS值
设置梯级输出条件为假
结束

条件：

梯级输入条件为真

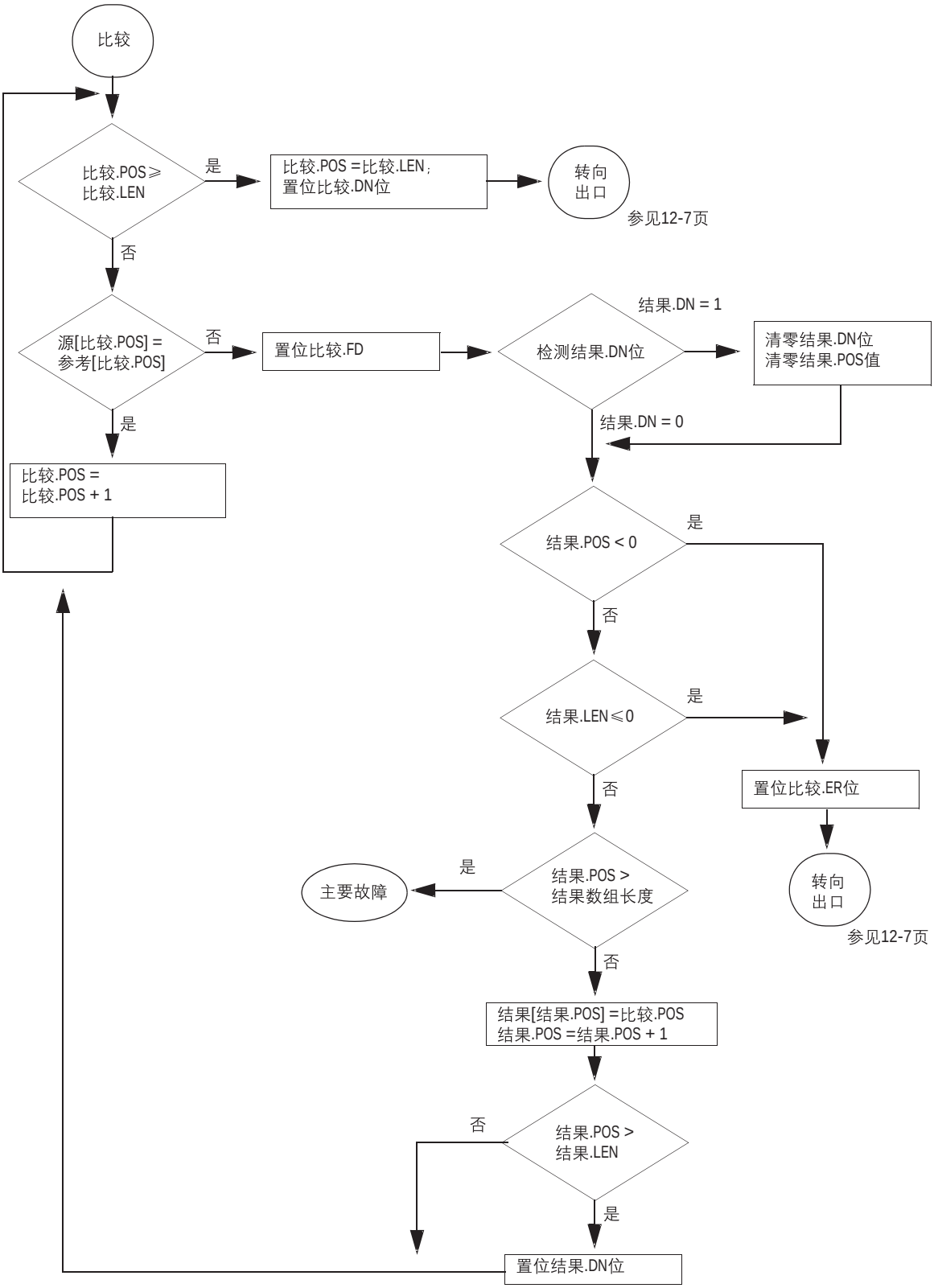
梯形图动作：



参见12-8页

条件：

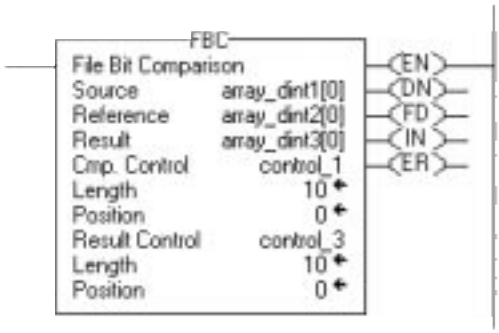
梯形图动作：



后期扫描

梯级输出条件被设置为假。

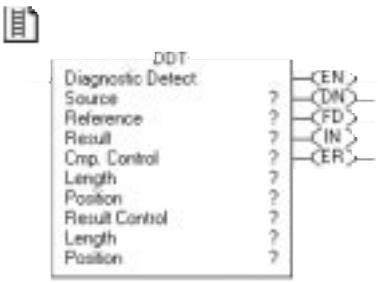
举例：如果FBC指令被使能，则对源array_dint1 与参考 array_dint2 进行比较并将全部不匹配的位置存入结果array_dint3中。



诊断检测 (DDT)

DDT指令使源数组的位与参考数组中的对应位进行比较，以检测状态变化。

操作数：



梯形图

操作数：	数据类型：	格式：	说明：
Source 源数组	DINT	数组标签	要与参考数组进行比较的数组 不要在下标处用CONTROL.POS
Reference 参考数组	DINT	数组标签	要与源数组进行比较的数组 不要在下标处用CONTROL.POS
Result 结果数组	DINT	数组标签	存储比较结果的数组 不要在下标处用CONTROL.POS
Cmp control 比较控制结构体	CONTROL	结构体	比较操作的控制结构体
Length 长度	DINT	立即数	参与比较的位的数量
Position 位置	DINT	立即数	源数组中参与比较数值的当前位置 初始值一般是0
Result Control 结果控制结构体	CONTROL	结构体	运算结果的控制结构体
Length 长度	DINT	立即数	存储在结果数组中不匹配位置的数量
Position 位置	DINT	立即数	结果数组的当前位置，初始值一般是0

注 意:比较控制结构体和结果控制结构体要用不同的标签。如果二者使用相同的标签会发生不可预料的操作，可能会导致设备的损坏和(或)人员伤亡。

COMPARE结构体

助记符:	数据类型:	说明:
.EN	BOOL	使能位表明DDT指令被使能。
.DN	BOOL	当DDT指令完成源与参考数组最后一位的比较时，置位完成位。
.FD	BOOL	每次DDT指令记录一次不匹配(一次比较一个操作模式)或记录全部不匹配(每次扫描比较全部位操作模式)后置位发现位。
.IN	BOOL	禁止位表明DDT指令的搜索模式 0=整体模式 1=一次比较一个不匹配模式
.ER	BOOL	如果出现比较.POS < 0、比较.LEN < 0、结果.POS < 0或.LEN < 0则置位错误位。在程序清零错误位.ER之前不执行指令。
.LEN	DINT	长度值表示参与比较的位的数量。
.POS	DINT	位置值表示当前位的位置。

RESULT结构体

助记符:	数据类型:	说明:
.DN	BOOL	当结果数组已满时，置位完成位。
.LEN	DINT	长度值表示结果数组中用于存储的位置数量。
.POS	DINT	位置值表示结果数组中当前位的位置。

说明: 当DDT指令被使能时，将源数组中的位与参考数组中的位进行比较，在结果数组中记录每个不匹配位的位号，并改变参考位的值以使它与对应的源中的位匹配。

DDT 指令对内存中的连续区域进行操作。

DDT与FBC指令的区别是DDT指令每次查找一个不匹配位，而且改变参考位的值使其与源中的位匹配。而FBC指令不改变参考位。

选择搜索模式

如果要检测：	选择如下模式：
一次扫描记录一个不匹配	置位比较CONTROL结构体的禁止位.IN。 每次梯级输入条件由假变真时，DDT指令搜索源和参考数组中的下一个不匹配位。发现不匹配后置位发现位.FD，记录不匹配位的位置并停止执行。
一次扫描记录全部不匹配位	清零比较CONTROL结构体的禁止位.IN。 每次梯级输入条件由假变真时，DDT指令搜索源和参考数组中的全部不匹配位。

算术状态标志：不影响

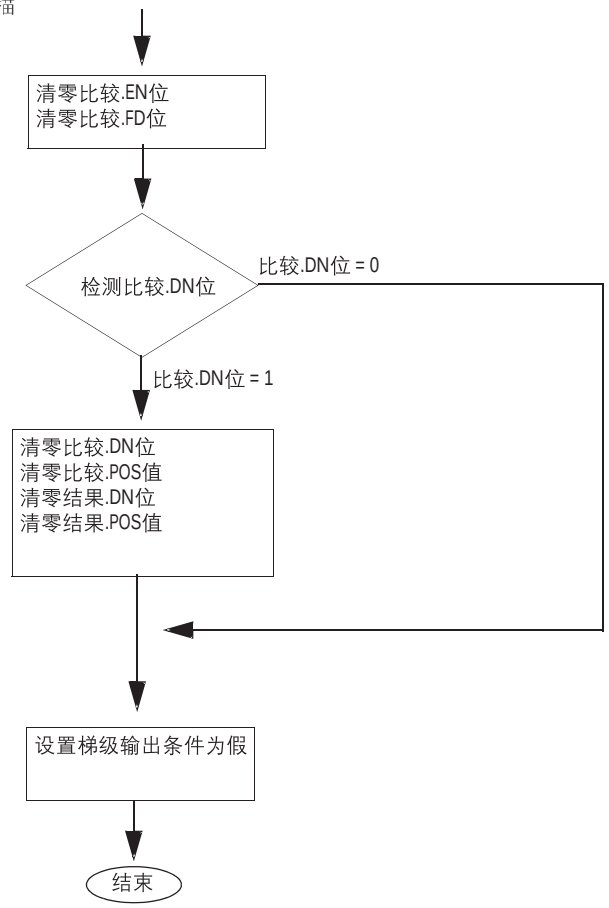
故障条件：

发生重要故障的条件：	故障类型：	故障代码：
结果.POS >数组的长度	4	20

执行：

条件：　**梯形图动作：**

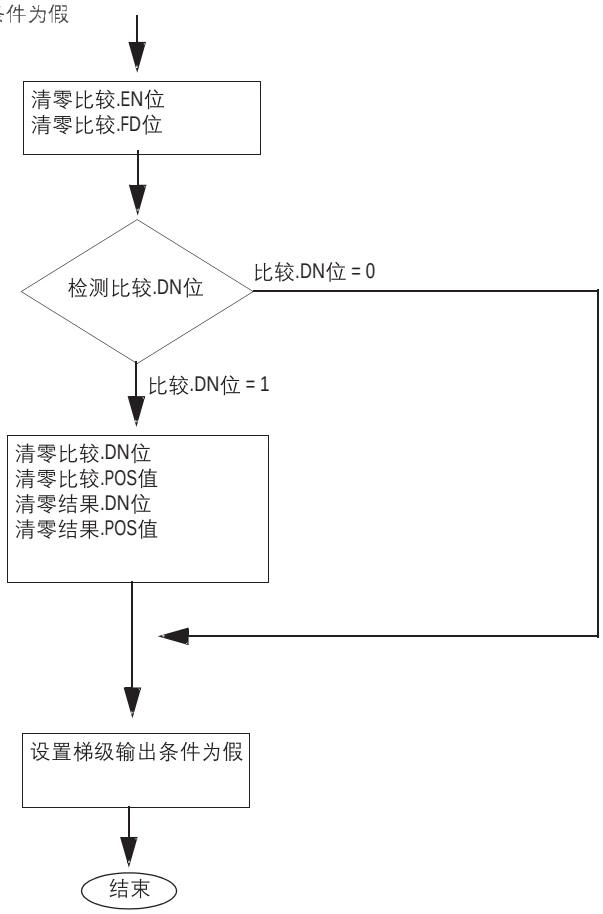
预扫描



条件：

梯级输入条件为假

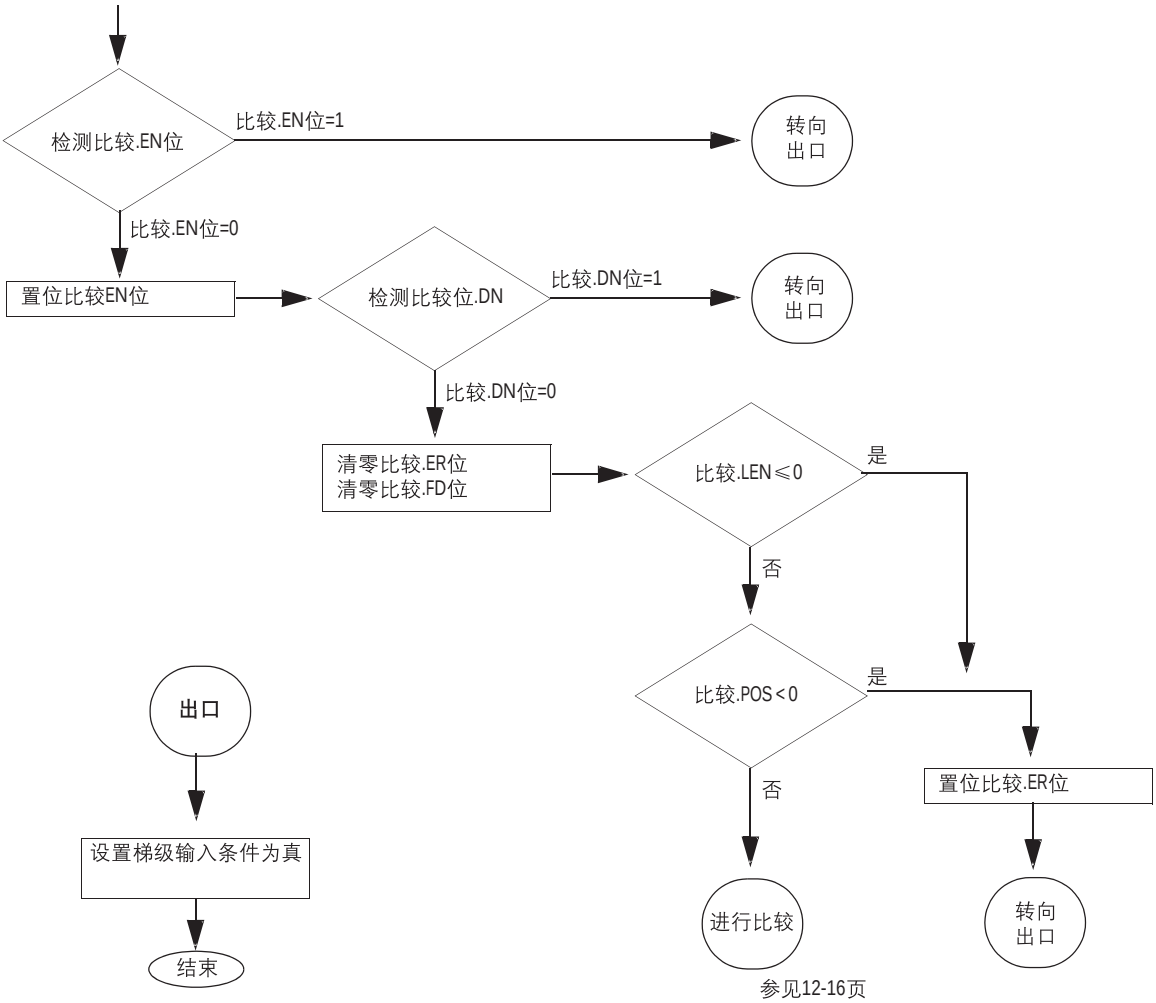
梯形图动作：



条件：

梯级输入条件为真

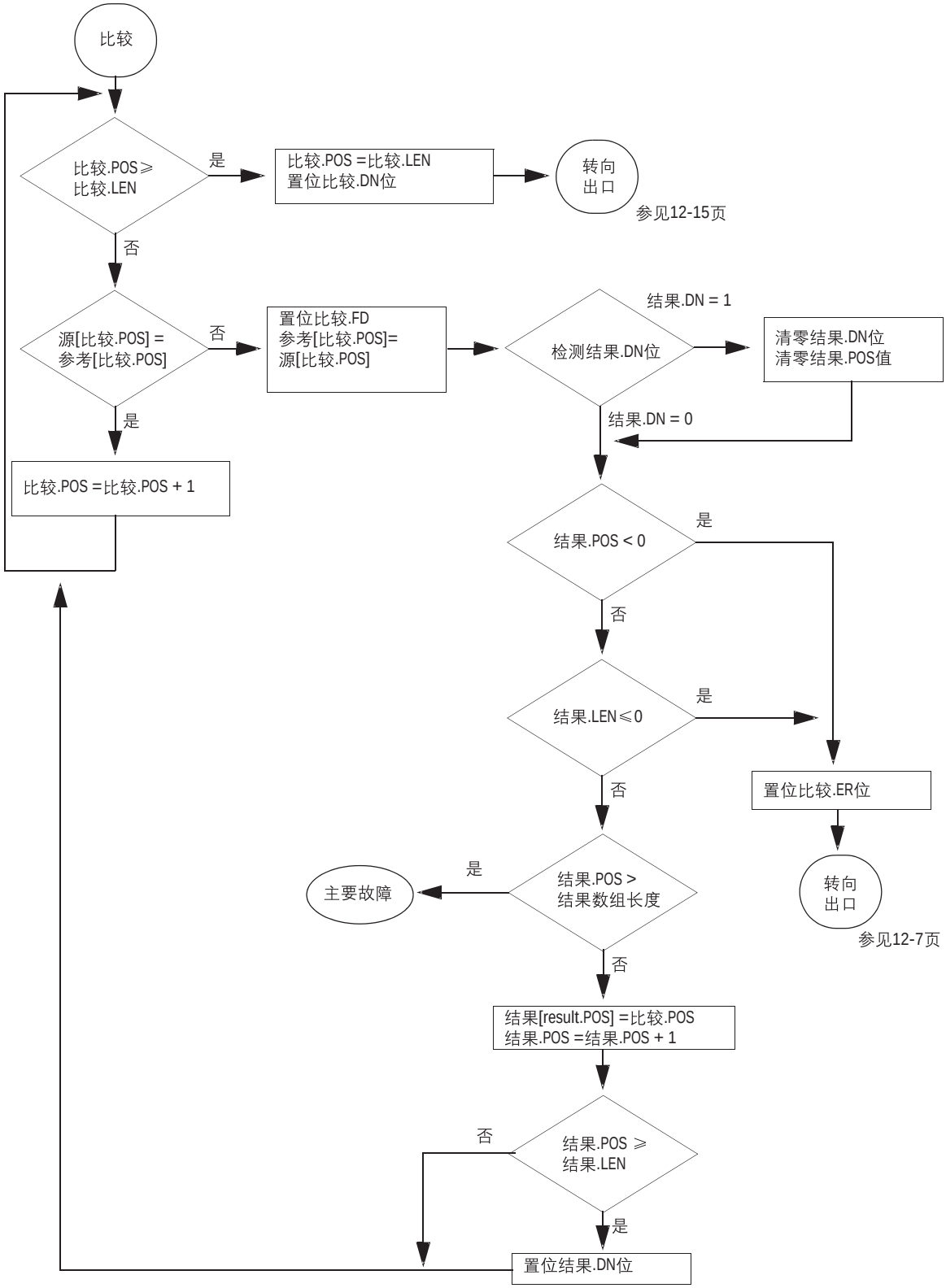
梯形图动作：



参见12-16页

条件：

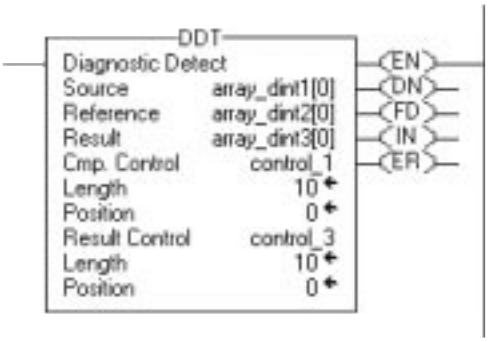
梯形图动作：

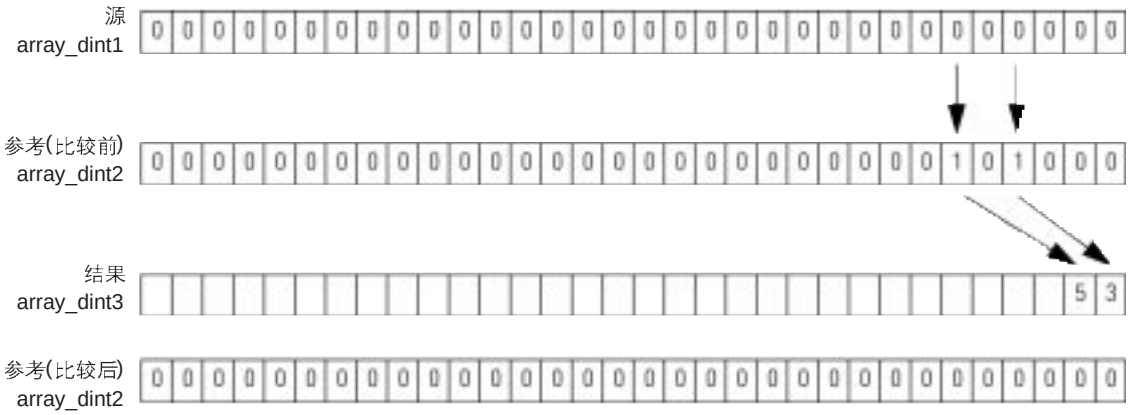


后期扫描

梯级输出条件被设置为假。

举例: 如果DDT 指令被使能 则对源数组array_dint1 与参考数组 array_dint2 进行比较并 将全 部不 匹配 位的 位置 存入 结果array_dint3 中。控制器同时改变参考array_dint2中的不匹配位使之与源array_dint1匹配。



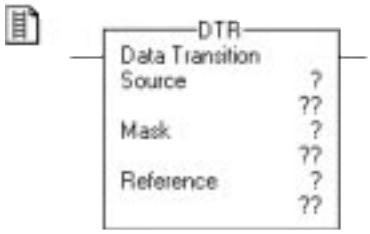


数据传送(DTR)

DTR 指令通过屏蔽传送源中的值，并将传送的结果与参考值进行比较。

操作数：

梯形图



操作数：	数据类型：	格式：	说明：
Source 源值	DINT	立即数 标签	要与参考数组进行比较的数组
Mask 屏蔽值	DINT	立即数 标签	定义需要阻滞或通过的位
Reference 参考值	DINT	标签	要与源数组进行比较的数组

说明： DTR 指令通过屏蔽传送源中的值，并将传送的结果与参考值进行比较。DTR 指令同时将屏蔽的源值写入参考值中以用于下次比较。源值保持不变。

屏蔽位中的 ″1″ 意味着数据可以通过， ″0″ 意味着数据将被阻滞。

如果通过屏蔽的源值与参考值不一致，则梯级输出条件为真一个扫描周期。如果通过屏蔽的源值与参考值一致，则梯级输出条件变为假。

注 意:对该指令在线编程可能会有危险。如果参考值与源值不一致，则梯级输入条件变为真。所以在处理器处在运行或远程运行状态模式时插入该指令一定要小心。

用立即数做屏蔽值

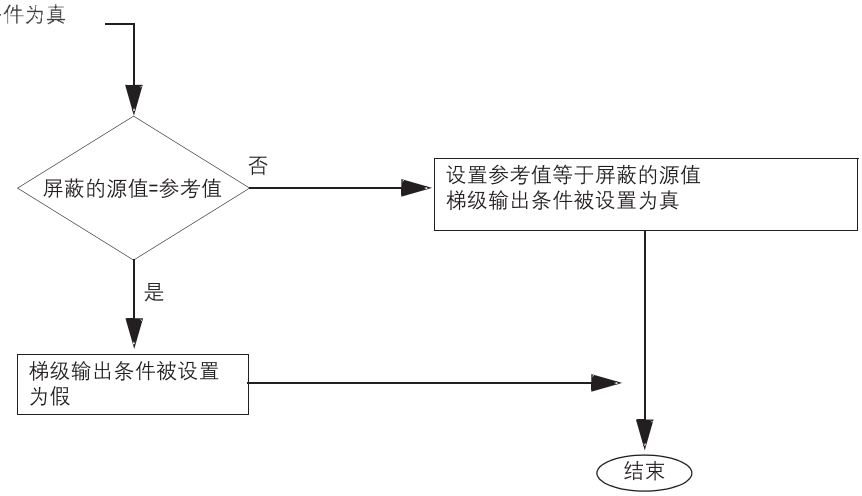
当用户输入屏蔽值时，编程软件缺省为输入十进制值。如果用户需要用其它数据格式输入屏蔽值，则应该按下列方法在输入值前加相应前缀。

前缀：	说明：
16#	十六进制 例如16#0F0F
8#	八进制 例如8#16
2#	二进制 例如2#00110011

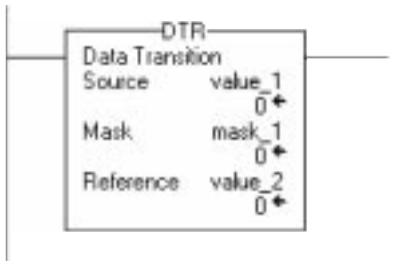
算术状态标志：不影响

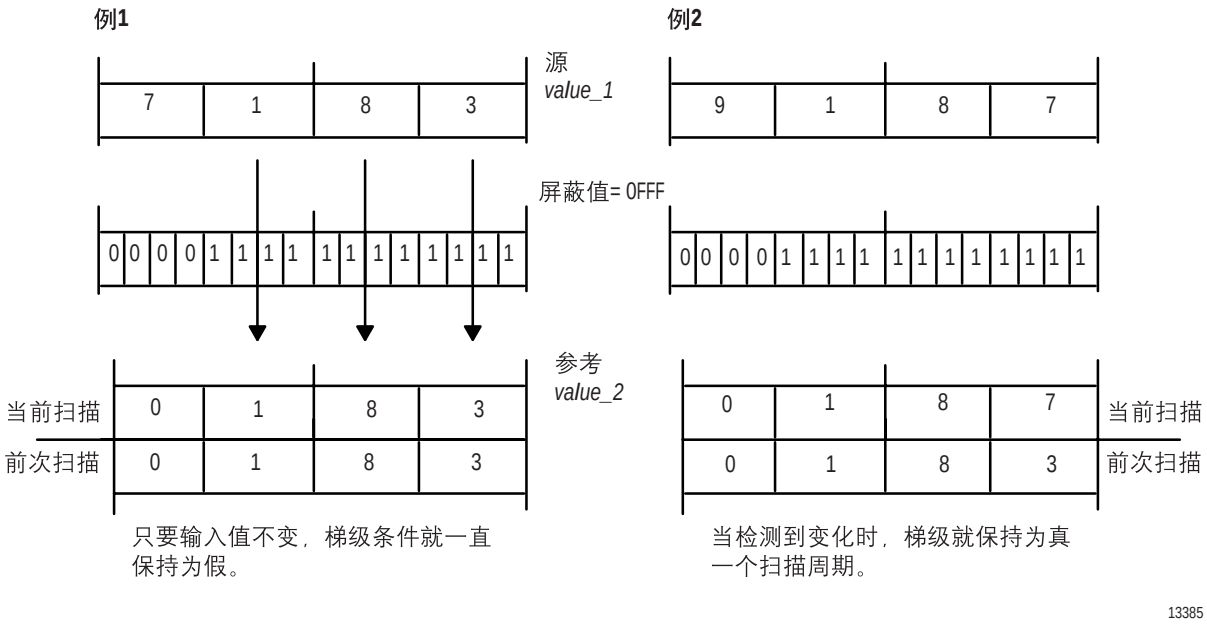
故障条件：无

执行：

条件：	梯形图动作：
预扫描	Reference = Source AND Mask。 梯级输出条件被设置为假。
梯级输入条件为假	Reference = Source AND Mask。 梯级输出条件被设置为假。
梯级输入条件为真	
后期扫描	梯级输出条件被设置为假。

举例：当DTR指令被使能时，指令屏蔽value_1。如果两个值不同，则设置梯级输出条件为真。





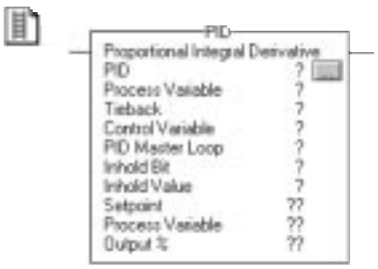
13385

屏蔽位中的0值不改变相应的位。

比例 积分 微分指令(PID)

PID指令可用于控制以下过程变量，如：流量、压力、温度或液位。

操作数：



梯形图

操作数：	数据类型：	格式：	说明：
PID	PID	结构体	PID 结构体
Process variable 过程变量	SINT INT DINT REAL	标签	用户需要控制的数值
Tieback	SINT INT DINT REAL	立即数 标签	(可选)硬件手动/自动工作站的输出，它旁路控制器的输出 如果用户不需要使用此参数，可输入0值。
Control variable 控制变量	SINT INT DINT REAL	标签	用于控制最终设备的数值(阀、气泵等) 如果用户用死区控制，则控制变量的数据类型必须是REAL，否则当误差在死区范围内时，该值会被强制为0。
PID master Loop(主PID回路)	PID	结构体	(可选)用于主PID回路的PID标签 如果用户执行级联控制，而且此PID是从回路，则输入主PID回路的名称。如果用户不需要使用此参数，可输入0值。
Inhold bit	BOOL	标签	(可选)来自1756模拟量输出通道的初始化保持位的当前状态，用于支持无冲击再起动 如果用户不需要使用此参数，可输入0值。
Inhold value	SINT INT DINT REAL	标签	(可选)来自1756模拟量输出通道的数据读出值，用于支持无冲击再起动 如果用户不需要使用此参数，可输入0值。
Setpoint 设定点			显示设定点的当前值
Process Variable 过程变量			显示标定过程变量的当前值
Output % 输出百分比			显示输出百分比的当前值

结构文本

```
PID(PID,ProcessVariable,
Tieback,ControlVariable,
PIDMasterLoop,InholdBit,
InHoldValue);
```

该指令中操作数与梯形图中PID指令的操作数相同。用户通过访问PID结构体中的.SP、.PV和.OUT指定设定点、过程变量和输出百分比，而非通过包含在操作数列表中的数值。

PID结构体

助记符：	数据类型：	说明：	
.CTL	DINT	.CTL项允许单独访问32-位状态字的各项。PID指令置位第7到15位。	
		位：	项：
		31	.EN
		30	.CT
		29	.CL
		28	.PVT
		27	.DOE
		26	.SWM
		25	.CA
		24	.MO
		23	.PE
		22	.NDF
		21	.NOBC
		20	.NOZC
		位：	PID指令置位的项：
		15	.INI
		14	.SPOR
		13	.OLL
		12	.OLH
		11	.EWD
		10	.DVNA
		9	.DVPA
		8	.PVLA
		7	.PVHA
.SP	REAL	设定点	
.KP	REAL	独立	比例增益(无量纲)
		相关	控制器增益(无量纲)
.KI	REAL	独立	积分增益(1 / 秒)
		相关	复位时间(分钟每循环)
.KD	REAL	独立	微分增益(秒)
		相关	加速时间(分钟)
.BIAS	REAL	前馈或偏差百分比	
.MAXS	REAL	最大工程单位标定值	
.MINS	REAL	最小工程单位标定值	

助记符:	数据类型:	说明:
.DB	REAL	死区工程单位
.SO	REAL	设置输出百分比
.MAXO	REAL	最大输出限值(输出百分比)
.MINO	REAL	最小输出限值(输出百分比)
.UPD	REAL	回路更新时间(秒)
.PV	REAL	已标定的过程变量值
.ERR	REAL	已标定的误差值
.OUT	REAL	输出百分比
.PVH	REAL	过程变量上限报警值
.PVL	REAL	过程变量下限报警值
.DVP	REAL	正偏移报警限值
.DVN	REAL	负偏移报警限值
.PVDB	REAL	过程变量报警死区
.DVDB	REAL	偏移报警死区
.MAXI	REAL	过程变量最大值(未标定的输入)
.MINI	REAL	过程变量最小PV值(未标定的输入)
.TIE	REAL	手动控制的牵引信号
.MAXCV	REAL	控制变量最大值(对应于100%)
.MINCV	REAL	控制变量最小值(对应于0%)
.MINTIE	REAL	牵引最小值(对应于100%)
.MAXTIE	REAL	牵引最大值(对应于0%)

助记符：	数据类型：	说明：	
.DATA	REAL[17]	.DATA 中的各项存储:	
		元素：	说明：
		.DATA[0]	积分累加值
		.DATA[1]	微分平滑临时数据
		.DATA[2]	前一次.PV 值
		.DATA[3]	前一次.ERR 值
		.DATA[4]	前一次有效.SP 值
		.DATA[5]	百分比标定常数
		.DATA[6]	.PV 的标定常数
		.DATA[7]	微分标定常数
		.DATA[8]	前一次.KP 值
		.DATA[9]	前一次.KI 值
		.DATA[10]	前一次.KD 值
		.DATA[11]	相关增益.KP
		.DATA[12]	相关增益.KI
		.DATA[13]	相关增益.KD
		.DATA[14]	前一次.CV 值
		.DATA[15]	.CV 的缩小标定常数
		.DATA[16]	牵引值的缩小标定常数
.EN	BOOL	使能	
.CT	BOOL	级联类型(0＝从；1＝主)	
.CL	BOOL	级联回路(0＝否；1＝是)	
.PVT	BOOL	过程变量跟踪(0＝否；1＝是)	
.DOE	BOOL	(0＝过程变量；1＝误差) 的微分	
.SWM	BOOL	软件手动模式(0＝否－自动；1＝是－SW 手动)	
.CA	BOOL	控制动作(0 表示E=SP-PV;1表示E=PV-SP)	
.MO	BOOL	工作站模式(0＝自动；1＝手动)	
.PE	BOOL	PID 方程(0＝独立；1＝非独立)	
.NDF	BOOL	无微分平滑 (0＝微分平滑滤波器被使能；1＝微分平滑滤波器被禁止)	
.NOBC	BOOL	无偏移量计算 (0＝偏移量计算被使能；1＝偏移量计算被禁止)	
.NOZC	BOOL	无死区过零 (0＝死去过零；1＝死区不过零)	
.INI	BOOL	PID 初始化(0＝否；1＝是)	
.SPOR	BOOL	设定点超限(0＝否；1＝是)	
.OLL	BOOL	CV 小于最小输出限值(0＝否；1＝是)	
.OLH	BOOL	CV 大于最大输出限值(0＝否；1＝是)	

助记符：	数据类型：	说明：
.EWD	BOOL	误差在死区范围内(0＝否；1＝是)
.DVNA	BOOL	偏差下限报警(0＝否；1＝是)
.DVPA	BOOL	偏差上限报警(0＝否；1＝是)
.PVLA	BOOL	PV下限报警(＝否；1＝是)
.PVHA	BOOL	PV上限报警(＝否；1＝是)

说明： 典型地，PID指令接收来自模拟量输入模块的过程变量(PV)并且通过模拟量输出模块调节控制变量输出(CV)，以保持过程变量在期望的设定点。

使能位.EN表示指令所在执行状态。当梯级输入条件由假转换为真时，使能位.EN被置位。当梯级输入条件变为假时，使能位.EN被清零。PID指令不使用完成位.DN。只要PID指令所在梯级输入条件为真，则每次扫描都执行PID指令。

算术状态标志： 不影响

故障条件：

重要事项 这些故障对于PLC-5处理器是主要故障。

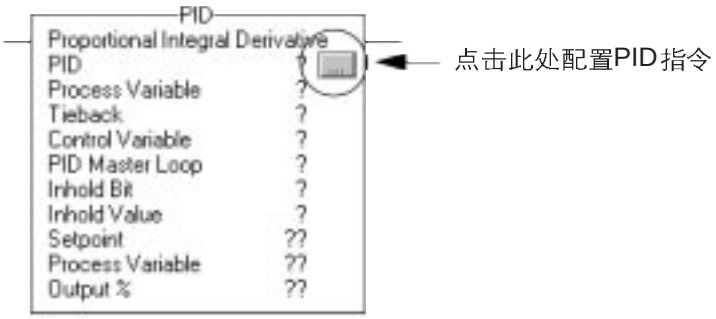
发生次要故障的条件：	故障类型：	故障代码：
.UPD ≤ 0	4	35
设定点超出范围	4	36

执行：

条件：	动作：	动作：
预扫描	梯级输出条件被设置为假。	不动作。
梯级输入条件为假	梯级输出条件被设置为假。	无影响
梯级输入条件为真	指令执行。 梯级输出条件被设置为真。	无影响
EnableIn被置位	不影响	EnableIn一直被置位。 指令执行。
指令执行	指令执行PID回路。	指令执行PID回路。
后期扫描	梯级输出条件被设置为假。	不动作。

组态PID指令

当用户输入PID指令并指定PID结构体后，用户可以使用组态选项卡来指定PID指令如何发挥作用。



指定调节方式

选择调节选项卡。只要用户单击其它区域，然后单击OK，再单击应用，或按回车键，则更改立刻有效。

在如下区域：	指定：
设定点(SP)	输入设定点值(.SP)。
设定输出百分比	输入给定的输出百分比(.SO)。 在软件手动模式中，该值用于输出。 在自动模式中，该值显示输出百分比。
输出偏差	输入输出偏差百分比(.BIAS)。
比例增益(Kp)	输入比例增益(.KP)。 对于独立增益，它是比例增益(无量纲)。 对于相关增益，它是控制器增益(无量纲)。
积分增益(Ki)	输入积分增益(.KI)。 对于独立增益，它是积分增益(1 / 秒)。 对于相关增益，它是复位时间(分钟每循环)。
微分增益(Kd)	输入微分增益(.KD)。 对于独立增益，它是微分增益(秒)。 对于相关增益，它是加速时间(分钟)。
手动模式	选择或手动(.MO)或软件手动(.SWM)模式。 如果两者都被选择，则手动模式优先于软件手动模式。

指定组态

选择组态选项卡。用户必须单击确定或应用才能使更改生效。

在如下区域：	指定：
PID方程	选择独立增益或相关增益(.PE)。 当用户想要需要三项增益(P、I、D)独立地运作时，使用独立增益。 当用户需要一个总的控制器增益对所有三项(P、I和D)起作用时，使用相关增益。
控制动作	选择控制动作(.CA)为E=PV-SP或E=SP-PV。
微分对象	选择对PV或误差进行微分(.DOE)。 用对PV进行微分以消除由设定点变动导致的输出尖峰。 当算法容许有超调时，使用对误差进行微分以获得对设定点变动的快速响应。
回路更新时间	输入指令更新时间(.UPD)
控制变量上限	输入控制变量上限值(.MAXO)。
控制变量下限	输入控制变量下限值(.MINO)。
死区值	输入死区值(.DB)。
无微分平滑作用	使能或禁止该项选择(.NDF)。

在如下区域：	指定：
无偏差计算	使能或禁止该项选择(.NOBC)。
无过零死区	使能或禁止该项选择(.NOZC)。
PV 跟踪	使能或禁止该项选择(.PVT)。
回路级联	使能或禁止该项选择(. CL)。
级联类型	如果回路级联被使能，则选择是从回路或主回路(.CT)

指定报警

选择报警选项卡。对于任何改动，用户必须单击OK或应用才能使更改生效。

在如下区域：	指定：
PV上限	输入PV上限报警值(.PVH)。
PV下限	输入PV下限报警值(.PVL)。
PV死区	输入PV报警死区值(.PVDB)。
正偏移	输入一个正偏移值(.DVP)。
负偏移	输入一个负偏移值(.DVN)。
偏移死区	输入偏移报警死区值(.DVDB)。

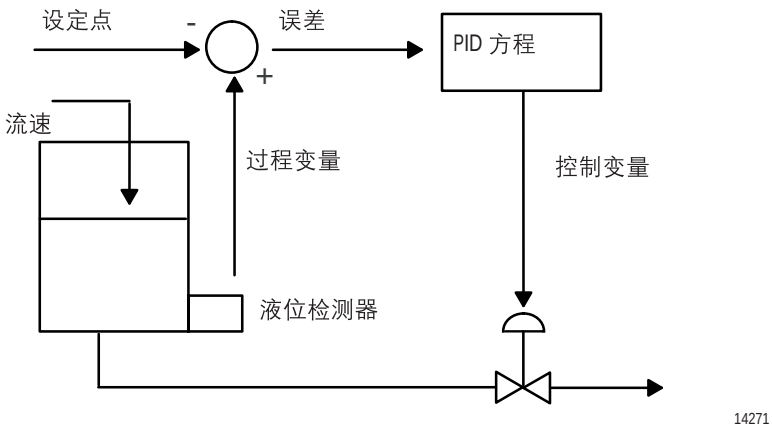
指定标定

选择标定选项卡。对于任何更改，用户必须单击OK或应用才能使更改生效。

在如下区域：	指定：
未标定PV最大值	输入PV的最大值(.MAXI)，它等于由模拟量输入通道获得的最大未标定PV值。
未标定PV最小值	输入PV的最小值(.MINI)，它等于由模拟量输入通道获得的最小未标定PV值。
PV的工程单位最大值	输入对应于.MAXI(.MAXS)的工程单位最大值。
PV的工程单位最小值	输入对应于.MINI(.MINS)的工程单位最小值。
CV最大值	输入对应于100%(.MAXCV)的CV最大值。
CV最小值	输入对应于0%(.MINCV)的CV最小值。
牵引最大值	输入牵引最大值(.MAXTIE),它等于由模拟量输入通道获得的牵引值的最大未标定值。
牵引最小值	输入牵引最小值(.MINTIE),它等于由模拟量输入通道获得的牵引值的最小未标定值。
PID 初始化	如果用户在运行模式中改变标定常数，则关闭该标志以重新初始化内部缩小比例值(.INI)。

使用PID指令

PID闭环控制使过程变量保持在期望的设定点。下图展示了一个流速/液位控制实例：



在上例中，比较罐内的液位与设定点。如果液位高于设定点，则PID方程增加控制变量并使罐的出口阀打开；从而降低罐内的液位。

用于PID指令的PID方程是一个可选择使用独立增益或相关增益的位置形式方程。当使用独立增益时，比例、积分和微分增益只分别影响它们各自专用的比例、积分和微分项。当使用相关增益时，比例增益被控制器增益所代替，它影响所有的三项。用户可以用任何一种形式的方程来执行同一类型的控制。提供两种方程形式仅仅是为了让用户使用其最熟悉的方程形式。

增益：	微分选项：	方程：
相关增益 (ISA 标准)	偏差(E)	$CV = K_C\left[E + \frac{1}{T_i}\int_0^t Edt + T_d\frac{dE}{dt}\right] + BIAS$
	过程变量(PV)	E = SP - PV $CV = K_C\left[E + \frac{1}{T_i}\int_0^t Edt - T_d\frac{dPV}{dt}\right] + BIAS$ E = PV - SP $CV = K_C\left[E + \frac{1}{T_i}\int_0^t Edt + T_d\frac{dPV}{dt}\right] + BIAS$
独立增益	偏差(E)	$CV = K_pE + K_i\int_0^t Edt + K_d\frac{dE}{dt} + BIAS$
	过程变量(PV)	E = SP - PV $CV = K_pE + K_i\int_0^t Edt - K_d\frac{dPV}{dt} + BIAS$ E = PV - SP $CV = K_pE + K_i\int_0^t Edt + K_d\frac{dPV}{dt} + BIAS$

式中：

变量：	说明：
Kp	比例增益(无量纲) Kp=Kc 无量纲
Ki	积分增益(1 / 秒) 在Ki(积分增益)和Ti(复位时间) 之间进行转换，使用公式： $K_i = \frac{K_c}{60T_i}$
Kd	微分增益(秒) 在K_d (微分增益)和Td(加速度时间) 之间进行转换，使用公式： $K_d=K_c(T_d)60$
Kc	控制器增益(无量纲)
Ti	积分时间(分钟/循环)
Td	微分时间(分钟)
SP	设定点
PV	过程变量
E	偏差[(SP-PV)或(PV-SP)]
BIAS	前馈或偏置
CV	控制变量
dt	回路更新时间

如果用户不需要使用PID方程的某个特定项，只须设置其增益为零即可。例如，如果用户不需要采用微分动作，则设定Kd或Td等于零。

防止积分超调及由手动转自动的无冲击转换

PID指令通过防止CV输出达到其由MAXO和MINO设定的最大或最小值时，积分项继续累加来自动避免积分超调。累加的积分项保持不变，直到CV输出降到其最大上限值以下，或升到其最小下限值以上，然后正常的积分累加值自动重新开始。

PID指令支持两种手动控制模式：

手动控制模式：	说明：
软件手动(.SWM)	也称为设定输出模式 允许用户通过软件设定输出百分比 设定输出(.SO)值被用作回路输出。设定输出值一般来自操作员界面设备的操作员输入
手动(.MO)	牵引值作为输入，通过调节其中间变量产生相同值的输出 PID指令牵引信号的输入根据.MINTIE和.MAXTIE值被标定到0至100%的范围，并被用作回路输出。牵引输入一般来自于硬件的手动/自动工作站，使输出旁路控制器输出 注意：如果两种手动模式选择位均被置位，则手动模式覆盖软件手动模式。

PID指令还自动提供由软件手动模式到自动模式，或手动模式到自动模式的无冲击转换。PID指令反向计算CV输出跟踪软件手动模式时的设定输出(.SO)，或手动模式时的牵引输入所需要的积分累加项的值。采用这种方式，当回路切换到自动模式时，CV输出从设定输出或牵引值开始，故输出值不会出现冲击。

即使未使用积分控制(即Ki=0)，PID指令也能自动提供从手动到自动的无冲击转换。在这种情况下，指令修改.BIAS项以使CV输出跟踪设定输出或牵引值。当自动控制重新开始时，.BIAS项保持上次的数值。用户可以通过在PID数据结构体中置位.NOBC位来禁止反向计算.BIAS项。注意如果用户设置.NOBC位为真，则PID指令在未使用积分控制时不再提供从手动到自动的无冲击转换。

PID指令回路更新

PID指令和对过程变量的采样需要以一定周期进行更新。更新时间与用户所控制的物理过程有关。对于非常缓慢的回路，例如温度回路，采用每秒一次或更长的更新时间能够获得好的控制效果。而对于快速回路，例如压力或流量回路，可能需要如每250毫秒一次的更新时间。极少数情况下，(如开卷机的张力控制)需要每10毫秒一次或更快的回路更新时间。

由于PID指令在其计算中使用时间基，所以用户需要把指令的执行同过程变量(PV)的采样同步。

执行PID指令的最简单方法是把PID指令放于周期性任务中。设定回路更新时间(.UPD)等于周期性任务的速率，并确保PID指令在每次扫描周期性任务时都执行。

梯形图

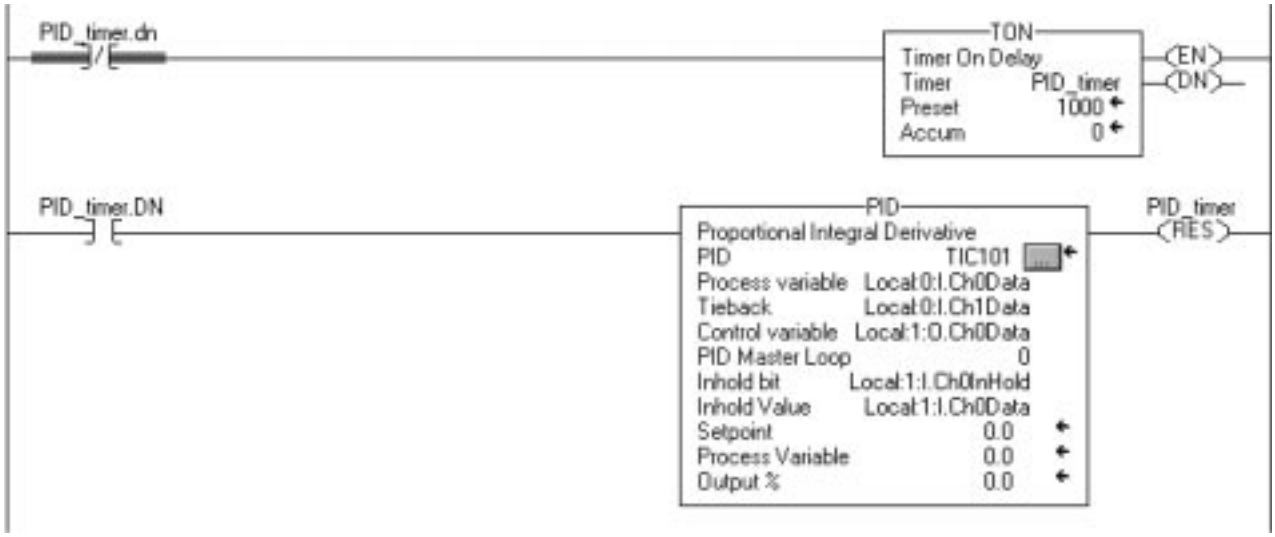
结构文本

```
PID(TIC101,Local:0:I.Ch0Data,Local:0:I.Ch1Data,
      Local:1:O.Ch4Data,0,Local:1:I.Ch4InHold,
      Local:1:I.Ch4Data);
```

当使用周期性任务时，应确保处理器中用于更新过程变量的模拟量输入的速率要比周期性任务速率快的多。理想情况下，需要以至少比周期性任务的速率快5-10倍的速率向处理器传送过程变量。这使过程变量的实际采样和PID回路的执行间的时间差最小。例如，如果PID回路处于一个250毫秒的周期性任务中，设置回路更新时间为250毫秒 (.UPD=.25)，并组态模拟量输入模块至少每25到50毫秒产生一次数据。

执行PID指令的另外一种方法是把指令放于连续任务中，并使用定时器完成位来触发PID指令的执行。但是精度稍差些。

梯形图



结构文本

```
PID_timer.pre := 1000

TONR(PID_timer);

IF PID_timer.DN THEN

    PID(TIC101,Local:0:I.Ch0Data,Local:0:I.Ch1Data,
    Local:1:O.Ch0Data,0,Local:1:I.Ch0InHold,
    Local:1:I.Ch0Data);

END_IF;
```

在这种方法中，需设定PID指令的回路更新时间等于定时器预置值。如使用周期性任务的情况，用户需要设置模拟量输入模块，使其生成过程变量的速率比回路更新时间快得多。用户对于回路更新时间至少是最差情况下连续任务执行时间几倍的回路中使用定时器方式执行PID。

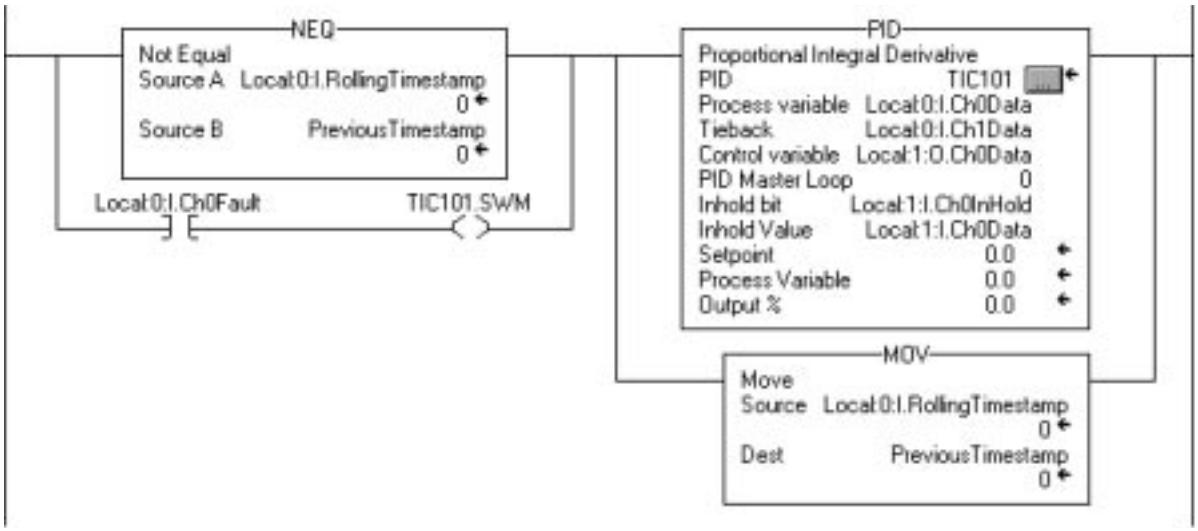
执行PID指令的最精确方法是利用1756模拟量输入模块的实时采样特性。模拟量输入模块以用户设置模块时配置的实时采样速率对其输入进行采样。当每经过一个模块实时采样周期时，它就更新其输入，并更新由模块产生的滚动时间戳(由模拟量输入数据结构体的滚动时间戳的数值所表示)。

时间戳范围为0-32767毫秒。监视时间戳。当时间戳改变时，已经收到一个新的过程变量采样。每次当时间戳发生改变，执行一次PID指令。这是因为过程变量采样是由模拟量输入模块驱动，输入采样时间非常准确，故PID指令使用的回路更新时间应设置为等于模拟量输入模块的RTS时间。

为确保用户不错过过程变量的采样，应以比RTS时间快的速率执行用户的程序逻辑。例如，如果RTS时间是250毫秒，则用户可以把PID逻辑放于每100毫秒执行一次的周期性任务中，以确保不错过每次采样。用户甚至可以把PID逻辑放于连续任务中，只要确保程序逻辑能比每250毫秒一次快的频率被更新即可。

下面是一个采用RTS方法执行的实例。PID指令的执行与是否获得新的模拟量输入数据有关。如果模拟量输入模块故障或被移出，则控制器不再收到滚动时间戳，PID回路停止执行。用户应该监视PV模拟量输入的状态位，如果它显示错误状态，则强制回路进入软件手动模式，并在每次扫描时执行回路。这样使操作员仍能用手动方式改变PID回路的输出。

梯形图



结构文本

```
IF (Local:0:I.Ch0Fault) THEN
        TIC101.SWM [:=] 1;
ELSE
        TIC101.SWM := 0;
END_IF;

IF (Local:0:I.RollingTimestamp<>PreviousTimestamp) OR
        (Local:0:I.Ch0Fault) THEN

        PreviousTimestamp :=Local:0:I.RollingTimestamp;

        PID(TIC101,Local:0:I.Ch0Data,Local:0:I.Ch1Data,
        Local:1:O.Ch0Data,0,Local:1:I.Ch0InHold,
        Local:1:I.Ch0Data);

END_IF;
```

无冲击再起动

当控制器从编程模式转换为运行模式时，或控制器上电时，PID指令能与1756模拟量输出模块相配合，以支持无冲击再起动。

当1756模拟量输出模块失去与控制器的通讯联系，或检测到控制器处于编程模式时，模拟量输出模块将其输出设置为用户配置模块时指定的出错状态值。如果控制器返回到运行模式，或重新建立起与模拟量输出模块的通讯联系，用户可以通过使用PID指令参数的初始化保持位和初始化保持值，使PID指令自动重新设置，使其控制变量输出等于模拟量输出。

设置无冲击重启动：

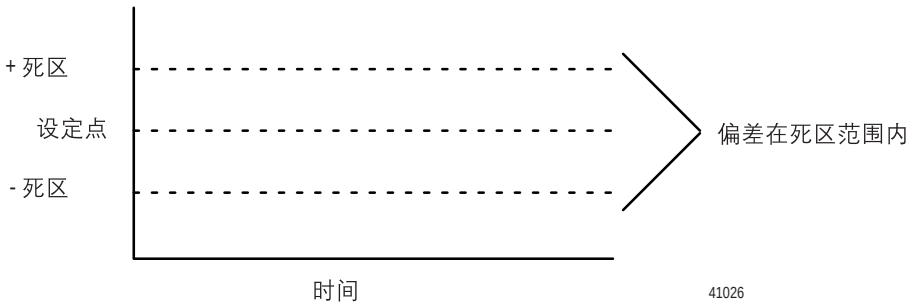
做如下工作：	详细说明：
组态从PID指令接收控制变量的1756模拟量输出模块的通道	在模块专用通道的属性页选取“初始化保持”选择框。 从而通知模拟量输出模块，当控制器返回到运行模式或重新建立与模块的通讯联系时，模块应将其模拟量输出保持在现有的数值上，直到从控制器发送的数值与输出通道使用的现有数据相匹配(在0.1%的幅度内)。控制器的输出通过使用.BIAS项以一定斜率过度到保持的当前输出值。这种过渡方式类似于自动无冲击转换。
在PID指令中输入初始化保持位标签和初始	1756模拟量输出模块在其输入数据结构体中为每个通道返回两个数值。当初始化保持状态位(例如：通道2的初始化保持位.CH2InHold)为真时，表明模拟量输出通道保持其数值。其数据读出值(例如：.Ch2Data)按工程单位显示现有的输出值。 输入初始化保持位状态标签作为PID指令的初始化保持位参数。输入数据读出数值标签作为初始化保持数值参数。 当初始化保持位变为真时，PID指令把初始化保持数值移入控制变量输出，并使用该数值重新初始化，以支持无冲击再起动。当模拟量输出模块从控制器接收到该数值时，关断初始化保持状态位，从而允许PID指令开始正常的控制。

微分平滑

通过微分平滑滤波器来增强微分计算。一阶低通数字滤波器有助有减小由PV的噪声引起的较大的微分项冲击。这种平滑作用影响采用较大的微分增益数值。如果用户过程中需要非常大的微分增益数值时(例如Kd>10)，可以禁止微分平滑作用。要禁止微分平滑作用，可以在组态选项卡中选择“禁止微分平滑”选项或在PID结构体中置位.NDF位。

设置死区

可调整的死区使用户能在设定点的上下选择一个误差范围，只要误差保持在这个范围内，输出就不发生变化。该死区使用户能够控制在不改变输出的条件下，过程变量与设定点匹配的接近程度。死区还有助有减少用户的最终控制设备的磨损。



死区控制采用过零判别方式，当过程变量进入死区后，允许指令使用误差进行计算，直到过程变量经过设定点。一旦过程变量经过设定点(误差过零，且改变符号)，只要保持在死区范围内，输出就不改变。

过零死区是按用户指定的数值在设定点上下形成一个区域。输入零值可以使死区无效。死区与设定点具有相同的标定单位。用户可以通过在组态选项卡中选择“禁止过零死区”选项，或在PID结构体中置位.NOZC位，使用不具有过零特性的死区。

如果用户使用死区，则控制变量必须是REAL，否则当误差位于死区中时它会被强制为零。

使用输出限幅

用户可以对控制输出设置输出限幅(输出百分比)。当指令检测到输出达到限幅值时，置位一个报警位，并且防止输出超出上下限幅值。

前馈或输出偏置

用户可以通过反馈.BIAS值到PID指令的前馈/偏置数值中，以前馈补偿来自系统的扰动。

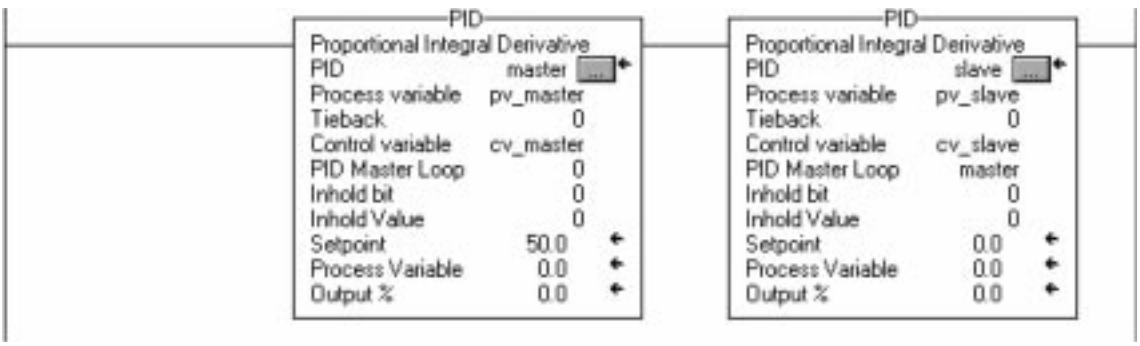
前馈值可以在反馈到PID指令的扰动有可能改变过程变量之前，补偿扰动的作用。在存在传输滞后的控制过程中经常采用前馈补偿。例如：代表注入混合器中冷水的前馈量能加速输出值，从而快于等待因混合结果引起的过程变量变化。

当未采用积分控制时，一般使用偏差值。在这种情况下，可通过调节偏差值，以保持输出在需要的范围内，从而保持PV接近设定点。

级联回路

PID指令通过把主回路的百分比输出赋值给从回路的设定点，来级联两个回路。从回路基于其回路的.MAXS和.MINS值，自动把主回路的输出转换为正确的工程单位，用来作为从回路的设定点。

梯形图



结构文本

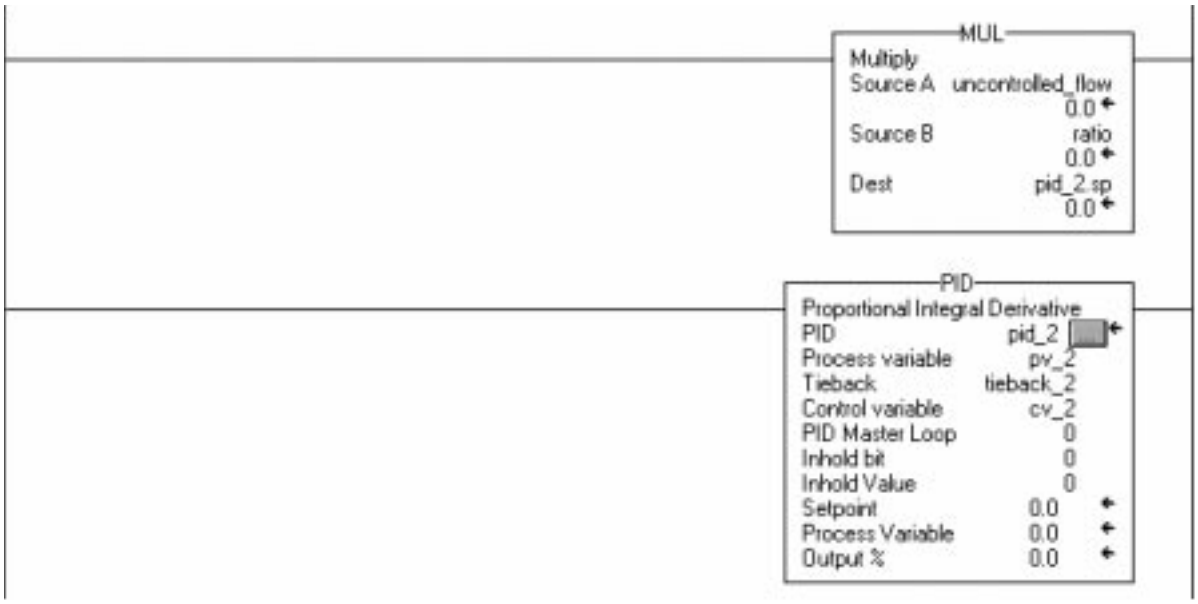
```
PID(master,pv_master,0,cv_master,0,0,0);

PID (slave,pv_slave,0,cv_slave,master,0,0);
```

比率控制

用户可以通过使用下列参数来保持两个数值具有一定的比率：

- 非受控量
- 控制量(PID指令使用的结果设定点)
- 两个量间的比率

梯形图



结构文本

pid_2.sp :=uncontrolled_flow*ratio

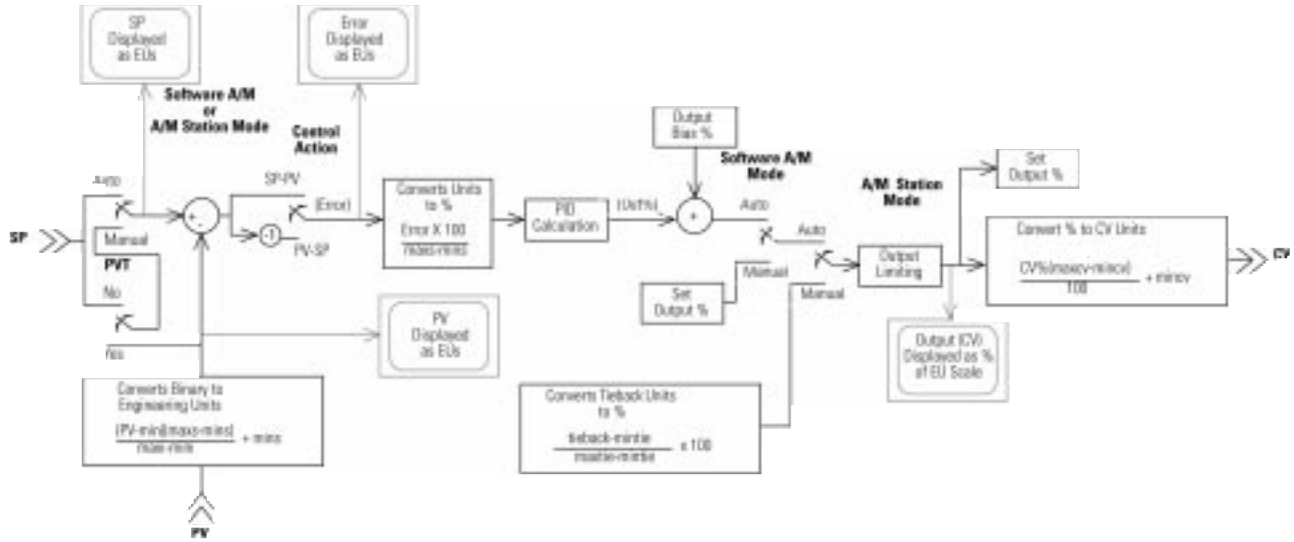
PID(pid_2,pv_2,tieback_2,cv_2,0,0,0);

对于如下乘法参数：	输入下列数值：
目的标签	控制量
源A	非受控量
源B	比率

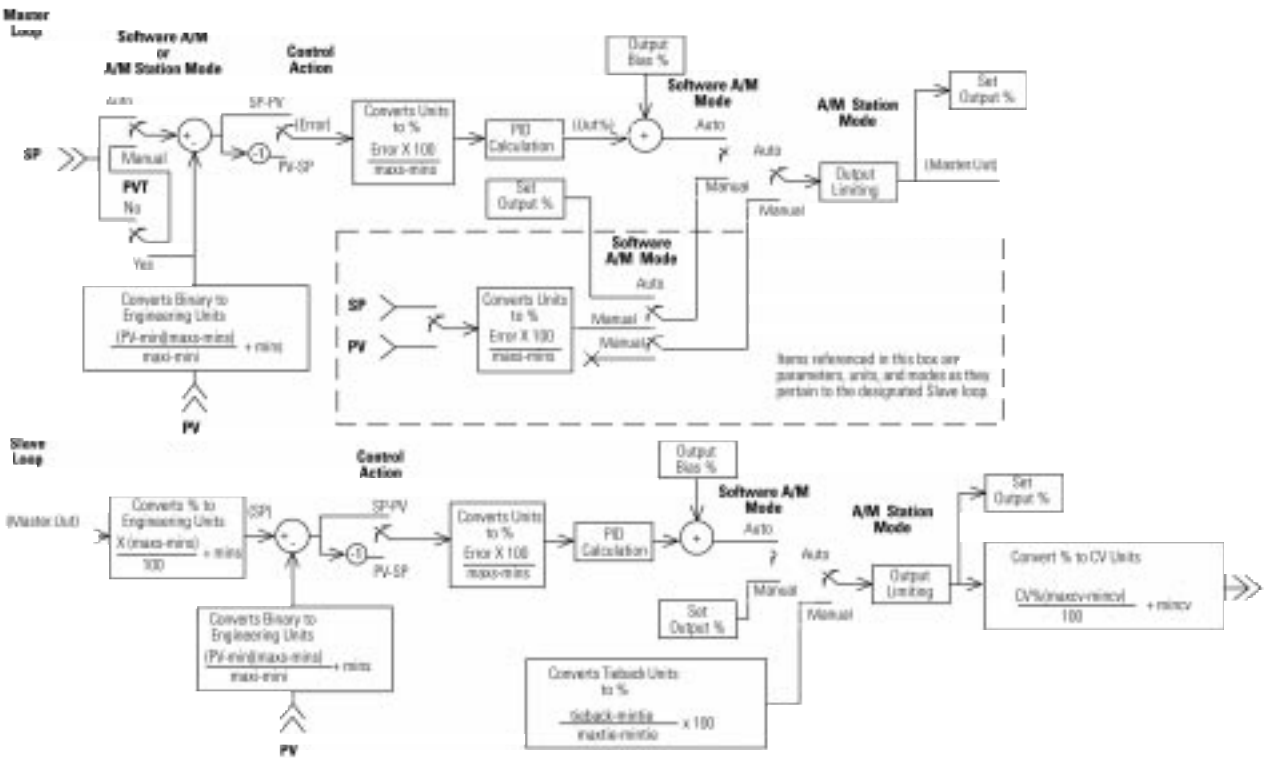
PID原理

下图演示了PID指令流程。

PID流程



带主/从回路的PID过程



注释：

三角函数指令
(SIN，COS，TAN，ASN，ASIN，ACS，ACOS，ATN ，ATAN)

简介

三角函数指令用三角函数来进行算术运算。

如果用户需要：	所用指令：	可用语言：	参见页码：
计算数值的正弦值	SIN	梯形图 结构文本 功能块	1-2
计算数值的余弦值	COS	梯形图 结构文本 功能块	13-5
计算数值的正切值	TAN	梯形图 结构文本 功能块	13-8
计算数值的反正弦值	ASN ASIN(1)	梯形图 结构文本 功能块	13-11
计算数值的反余弦值	ACS ACOS(1)	梯形图 结构文本 功能块	13-14
计算数值的反正切值	ATN ATAN(1)	梯形图 结构文本 功能块	13-17

(1)仅结构文本。

在运算时用户可以混用数据类型，但是这样会损失精度和发生取整误差，并且会延长指令的执行时间。检测溢出状态位(S:V)以确定结果是否被截短。

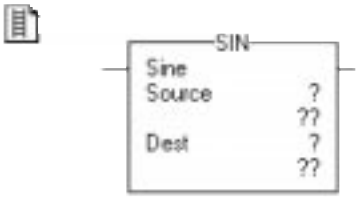
对于梯形图编程，黑体字数据类型是最优数据类型。如果指令的操作数都用同一最优数据类型，则指令的执行速度快且占用内存少。典型最优数据类型是DINT或REAL。

正弦 (SIN)

SIN指令计算源值(弧度)的正弦值，并将结果存入目的标签中。

操作数：

梯形图



操作数：	数据类型：	格式：	说明：
Source 源	SINT INT DINT **REAL**	立即数 标签	计算该值的正弦值
Destination 目的	SINT INT DINT **REAL**	标签	存储计算结果的标签

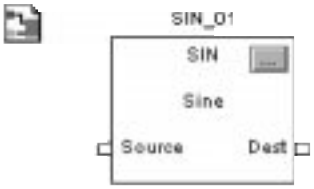
结构文本

```
dest := SIN(source);
```

将SIN用作函数。该函数计算source的正弦值，并将结果存入dest中。

关于结构文本中表达式的语法信息，参见附录C。

功能块



操作数：	数据类型：	格式：	说明：
SIN标签	FBD_MATH_ADVANCED	结构体	SIN结构体

FBD_MATH_ADVANCED结构体

输入参数：	数据类型：	说明：
EnableIn	BOOL	使能输入。如果清零，则指令不执行且输出不更新。 缺省状态是置位。
Source	REAL	输入到算术指令的值(计算该值的正弦值)。 有效值=任一浮点数
输出参数：	**数据类型：**	**说明：**
EnableOut	BOOL	指令产生一有效结果。
Dest	REAL	算术运算的结果。输出影响算术状态标志。

说明：源值必须大于或等于 -205887.4 ($-2\pi x2^{15}$) 且小于或等于205887.4 ($2\pi x2^{15}$)。目的标签中的结果总是大于或等于-1且小于或等于1。

算术状态标志：影响算术状态标志。

故障条件: 无

执行：

梯形图

条件：	动作：
预扫描	梯级输出条件被设置为假。
梯级输入条件为假	梯级输出条件被设置为假。
梯级输入条件为真	控制器计算源值的正弦值，并将结果存入目的标签。 梯级输出条件被设置为真。
后期扫描	梯级输出条件被设置为假。

功能块

条件：	动作：
预扫描	不动作。
首次扫描	不动作。
首次执行	不动作。
EnableIn是清零状态	清零EnableOut位。
EnableIn是置位状态	执行指令。 置位EnableOut位。
后期扫描	不动作。

举例：计算value的正弦值并将计算结果存入result标签中。

梯形图

结构文本

```
result := SIN(value);
```

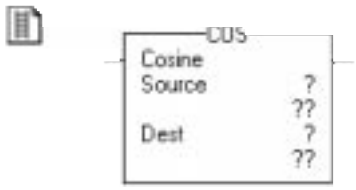
功能块

余弦 (COS)

COS指令计算源值(弧度)的余弦值，并将结果存入目的标签中。

操作数：

梯形图



操作数：	数据类型：	格式：	说明：
Source 源	SINT INT DINT **REAL**	立即数 标签	计算该值的余弦值
Destination 目的	SINT INT DINT **REAL**	标签	存储计算结果的标签

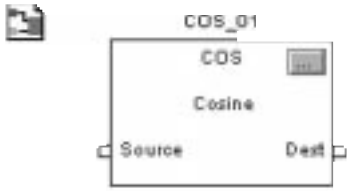
结构文本

```
dest := COS(source);
```

将COS用作函数。该函数计算source的余弦值，并将结果存入dest中。

关于结构文本中表达式的语法信息，参见附录C。

功能块



操作数：	数据类型：	格式：	说明：
COS标签	FBD_MATH_ADVANCED	结构体	COS结构体

FBD_MATH_ADVANCED结构体

输入参数：	数据类型：	说明：
EnableIn	BOOL	使能输入。如果清零，则指令不执行且输出不更新。 缺省状态是置位。
Source	REAL	输入到算术指令的值(计算该值的余弦值)。 有效值=任一浮点数
输出参数：	**数据类型：**	**说明：**
EnableOut	BOOL	指令产生一有效结果。
Dest	REAL	算术运算的结果。输出影响算术状态标志。

说明： 源必须大于或等于 -205887.4 ($-2\pi x2^{15}$) 且小于或等于205887.4 ($2\pi x2^{15}$)。目的标签中的结果总是大于或等于-1并小于或等于1。

算术状态标志：影响算术状态标志。

故障条件: 无

执行：

梯形图

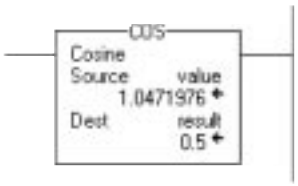
条件：	动作：
预扫描	梯级输出条件被设置为假。
梯级输入条件为假	梯级输出条件被设置为假。
梯级输入条件为真	控制器计算源值的余弦值，并将结果存入目的标签中。 梯级输出条件被设置为真。
后期扫描	梯级输出条件被设置为假。

功能块

条件：	动作：
预扫描	不动作。
首次扫描	不动作。
首次执行	不动作。
EnableIn 是清零状态	清零EnableOut位。
EnableIn 是置位状态	执行指令。 置位EnableOut位。
后期扫描	不动作。

举例：计算 value的余弦值，并将结果存入result中。

梯形图



结构文本

```
result := COS(value);
```

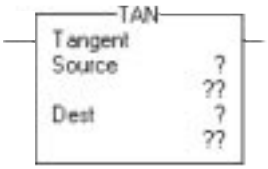
功能块

正切 (TAN)

TAN指令计算源值(弧度)的正切值，并将结果存入目的标签。

操作数：

梯形图



操作数：	数据类型：	格式：	说明：
Source 源	SINT INT DINT **REAL**	立即数 标签	计算该值的正切值
Destination 目的	SINT INT DINT **REAL**	标签	存储计算结果的标签

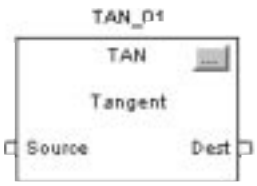
结构文本

```
dest := TAN(source);
```

使用TAN作为一个函数。该函数计算source的正切值，并将结果存入dest中。

关于结构文本中的表达式的语法信息，参见附录C。

功能块



操作数：	数据类型：	格式：	说明：
TAN 标签	FBD_MATH_ADVANCED	结构体	TAN结构体

FBD_MATH_ADVANCED结构体

输入参数：	数据类型：	说明：
EnableIn	BOOL	使能输入。如果清零，则指令不执行且输出不更新。 缺省状态是置位。
Source	REAL	输入到算术指令的值(计算该值的正切值)。 有效值=任一浮点数
输出参数：	**数据类型：**	**说明：**
EnableOut	BOOL	指令产生一有效结果。
Dest	REAL	算术运算的结果。输出影响算术状态标志。

说明： 源必须大于或等于 -205887.4 ($-2\pi x2^{14}$) 且小于或等于205887.4 ($-2\pi x2^{14}$)。

算术状态标志： 影响算术状态标志。

故障条件: 无

执行：

梯形图

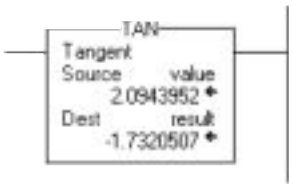
条件：	动作：
预扫描	梯级输出条件被设置为假。
梯级输入条件为假	梯级输出条件被设置为假。
梯级输入条件为真	控制器计算源值的正切值，并将结果存入目的标签。 梯级输出条件被设置为真。
后期扫描	梯级输出条件被设置为假。

功能块

条件：	动作：
预扫描	不动作。
首次扫描	不动作。
首次执行	不动作。
EnableIn是清零状态	清零EnableOut位。
EnableIn是置位状态	执行指令。 置位EnableOut位。
后期扫描	不动作。

举例：指令计算value的正切值并将计算结果存入result标签中。

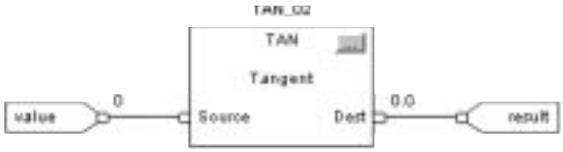
梯形图



结构文本

```
result := TAN(value);
```

功能块

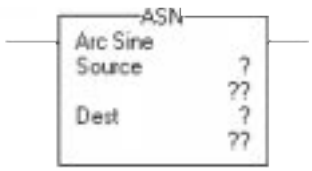


反正弦 (ASN)

ASN指令计算源值的反正弦值，并将结果存入目的标签(弧度)中。

操作数：

梯形图



操作数：	数据类型：	格式：	说明：
Source 源	SINT INT DINT **REAL**	立即数 标签	计算该值的反正弦值
Destination 目的	SINT INT DINT **REAL**	标签	存储计算结果的标签

结构文本

```
dest := ASIN(source);
```

将ASIN用作函数。该函数计算source的反正弦值，并将该值保存在dest中。

关于结构文本中表达式的语法信息，参见附录C。

功能块

操作数：	数据类型：	格式：	说明：
ASN标签	FBD_MATH_ADVANCED	结构体	ASN结构体

FBD_MATH_ADVANCED结构体

输入参数：	数据类型：	说明：
EnableIn	BOOL	使能输入。如果清零，则指令不执行且输出不更新。 缺省状态是置位。
Source	REAL	输入到算术指令的值(计算该值的反正弦值)。 有效值=任一浮点数
输出参数：	**数据类型：**	**说明：**
EnableOut	BOOL	指令产生一有效结果。。
Dest	REAL	算术运算的结果。输出影响算术状态标志。

说明： 说明：源值必须大于或等于-1且小于或等于1。目的标签的结果总是大于或等于-π/2且小于或等于π/2(此处 π= 3.141593)。

算术状态标志： 算术状态标志： 影响算术状态标志。

故障条件: 无

执行：

梯形图

条件：	动作：
预扫描	梯级输出条件被设置为假。
梯级输入条件为假	梯级输出条件被设置为假。
梯级输入条件为真	控制器计算源值的反正弦值，并将结果存入目的标签中。 梯级输出条件被设置为真。
后期扫描	梯级输出条件被设置为假。

功能块

条件：	动作：
预扫描	不动作。
首次扫描	不动作。
首次执行	不动作。
EnableIn 是清零状态	清零EnableOut 位。
EnableIn 是置位状态	执行指令。 置位EnableOut 位。
后期扫描	不动作。

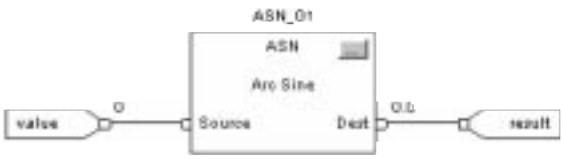
举例：计算value的反正弦值并将计算结果存入result标签中。

梯形图

结构文本

```
result := ASIN(value);
```

功能块

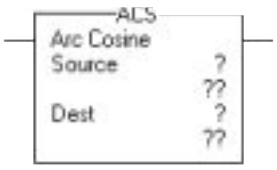


反余弦 (ACS)

ACS指令计算源值的反余弦值，并将结果存入目的标签(弧度)。

操作数：

梯形图



操作数：	数据类型：	格式：	说明：
Source 源	SINT INT DINT **REAL**	立即数 标签	计算该值的反余弦值
Destination 目的	SINT INT DINT **REAL**	标签	存储计算结果的标签

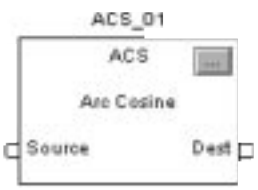
结构文本

```
dest := ACOS(source);
```

将ACOS用作函数。该函数计算source的反余弦值，并将结果保存在dest中。

关于结构文本中表达式的语法信息，参见附录C。

功能块



操作数：	数据类型：	格式：	说明：
ACS标签	FBD_MATH_ADVANCED	结构体	ACS结构体

FBD_MATH_ADVANCED结构体

输入参数：	数据类型：	说明：
EnableIn	BOOL	使能输入。如果清零，则不执行指令且不更新输出。 缺省状态是置位。
Source	REAL	输入到算术指令的值(计算该值的反余弦值)。 有效值=任一浮点数
输出参数：	**数据类型：**	**说明：**
EnableOut	BOOL	指令产生一有效结果。
Dest	REAL	算术运算的结果。输出影响算术状态标志。

说明： 源值必须大于或等于-1且小于或等于1。目的标签的结果总是大于或等于π/2且小于或等于π/2(此处π= 3.141593)。

算术状态标志： 影响算术状态标志。

故障条件: 无

执行：

梯形图

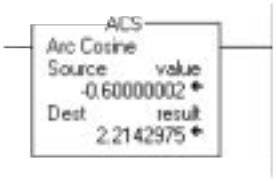
条件：	动作：
预扫描	梯级输出条件被设置为假。
梯级输入条件为假	梯级输出条件被设置为假。
梯级输入条件为真	控制器计算源值的反余弦值，并将结果存入目的标签。 梯级输出条件被设置为真。
后期扫描	梯级输出条件被设置为假。

功能块

条件：	动作：
预扫描	不动作。
首次扫描	不动作。
首次执行	不动作。
EnableIn 是清零状态	清零EnableOut位。
EnableIn 是置位状态	执行指令。 置位EnableOut位。
后期扫描	不动作。

举例：计算value的反余弦值并将计算结果存入result标签中。

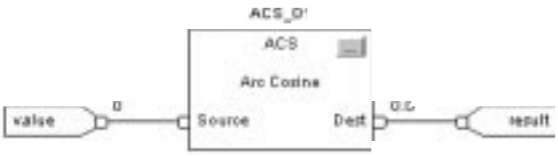
梯形图



结构文本

```
result := ACOS(value);
```

功能块

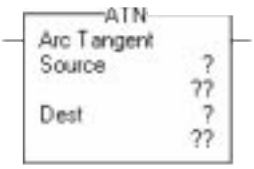


反正切 (ATN)

ATN指令计算源值的反正切值，并将结果存入目的标签(弧度)。

操作数：

梯形图



操作数：	数据类型：	格式：	说明：
Source 源	SINT INT DINT **REAL**	立即数 标签	计算该值的反正切值
Destination 目的	SINT INT DINT **REAL**	标签	存储计算结果的标签

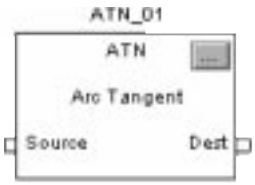
结构文本

```
dest := ATAN(source);
```

将ATAN用作函数。该函数计算source的反正切值，并将结果存入dest中。

关于结构文本中表达式的语法信息，参见附录C。

功能块



操作数：	数据类型：	格式：	说明：
ATN标签	FBD_MATH_ADVANCED	结构体	ATN结构体

FBD_MATH_ADVANCED结构体

输入参数：	数据类型：	说明：
EnableIn	BOOL	使能输入。如果清零，则指令不执行且输出不更新。 缺省状态是置位。
Source	REAL	输入到算术指令的值(计算该值的反正切值)。 有效值=任一浮点数
输出参数：	**数据类型：**	**说明：**
EnableOut	BOOL	指令产生一有效结果。
Dest	REAL	算术运算的结果。输出影响算术状态标志。

说明： 目的标签的结果总是大于或等于π/2且小于或等于π/2(此处π= 3.141593)。

算术状态标志： 影响算术状态标志。

故障条件: 无

执行：

梯形图

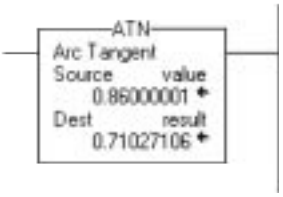
条件：	动作：
预扫描	梯级输出条件被设置为假。
梯级输入条件为假	梯级输出条件被设置为假。
梯级输入条件为真	控制器计算源值的反正切值，并将结果存入目的标签中。 梯级输出条件被设置为真。
后期扫描	梯级输出条件被设置为假。

功能块

条件：	动作：
预扫描	不动作。
首次扫描	不动作。
首次执行	不动作。
EnableIn 是清零状态	清零EnableOut 位。
EnableIn 是置位状态	执行指令。 置位EnableOut 位。
后期扫描	不动作。

举例：计算value的反正切值并将结果存入result 标签中。

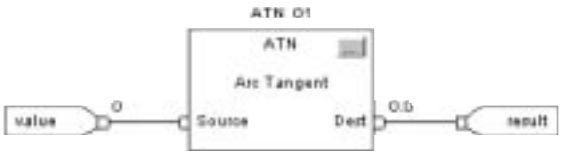
梯形图



结构文本

```
result := ATAN(value);
```

功能块



注释：

高级算术指令
(LN，LOG，XPY)

简介

高级算术指令包括下列指令：

如果用户需要：	所用指令：	可用语言：	参见页码：
计算数值的自然对数	LN	梯形图 结构文本 功能块	14-2
计算数值的以10为底的对数	LOG	梯形图 结构文本 功能块	14-4
对数值进行自乘(乘方)运算，自乘次数由另一个数值确定	XPY	梯形图 结构文本(1) 功能块	14-6

(1)无等价的结构文本指令。使用表达式内的运算符。

计算时可以混合使用不同数据类型，但是这样会损失精度也可能会发生取整误差，并且会延长指令的执行时间。检测S:V位，以确定结果是否被截短。

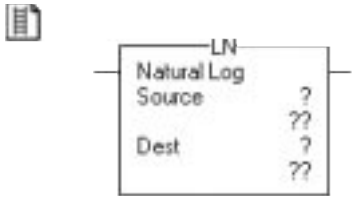
对于梯形图编程，黑体字数据类型是最优数据类型。如果指令的操作数都用同一最优数据类型，则指令的执行速度快且占用内存少。典型最优数据类型是DINT或REAL数据类型。

自然对数 (LN)

LN指令计算源值的自然对数，并将结果存入目的标签中。

操作数：

梯形图



操作数：	数据类型：	格式：	说明：
Source 源值	SINT INT DINT **REAL**	立即数 标签	计算该值的自然对数值
Destination 目的	SINT INT DINT **REAL**	标签	存储计算结果的标签

结构文本

```
dest := LN(source);
```

将LN用作函数。该函数计算source的自然对数，并将结果存入dest中。

关于结构文本中表达式的语法信息，参见附录C。

功能块

操作数：	数据类型：	格式：	说明：
LN标签	FBD_MATH_ADVANCED	结构体	LN结构体

FBD_MATH_ADVANCED结构体

输入参数：	数据类型：	说明：
EnableIn	BOOL	使能输入。如果清零，则不执行指令且不更新输出。 缺省状态是置位。
Source	REAL	输入到算术指令的值(计算该值的自然对数值)。 有效值=任意浮点数
输出参数：	**数据类型：**	**说明：**
EnableOut	BOOL	指令产生一有效结果。

说明： 源值必须大于0，否则会置位溢出状态位(S:V)。目的标签中的结果必须大于或等于-87.33655并小于或等于88.72284。

算术状态标志： 影响算术状态标志。

故障条件: 无

执行：

梯形图

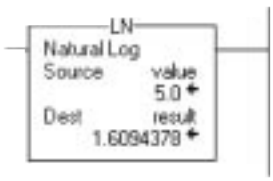
条件：	动作：
预扫描	梯级输出条件被设置为假。
梯级输入条件为假	梯级输出条件被设置为假。
梯级输入条件为真	控制器计算源值的自然对数值，并将结果存入目的标签中。 梯级输出条件被设置为真。
后期扫描	梯级输出条件被设置为假。

功能块

条件：	动作：
预扫描	不动作。
首次扫描指令	不动作。
首次执行指令	不动作。
EnableIn 是清零状态	清零EnableOut 位。
EnableIn 是置位状态	执行指令。 置位EnableOut 位。
后期扫描	不动作。

举例: 计算value的自然对数值并将计算结果存入result标签中。

梯形图



结构文本

```
result := LN(value);
```

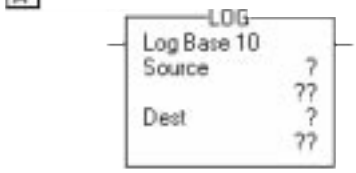
功能块

以10为底的对数 (LOG)

LOG指令计算源值以10为底的对数，并将结果存入目的标签中。

操作数：

梯形图



操作数：	数据类型：	格式：	说明：
Source 源	SINT INT DINT **REAL**	立即数 标签	计算该值以10为底的对数值
Destination 目的	SINT INT DINT **REAL**	标签	存储计算结果的标签

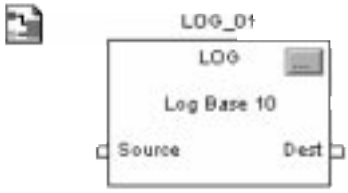
结构文本

```
dest := LOG(source);
```

将LOG用作函数。该函数计算source以10为底的对数，并将结果存入dest中。

关于结构文本中表达式的语法信息，参见附录C。

功能块



操作数：	数据类型：	格式：	说明：
LOG标签	FBD_MATH_ADVANCED	结构体	LOG结构体

FBD_MATH_ADVANCED结构体

输入参数：	数据类型：	说明：
EnableIn	BOOL	使能输入。如果清零，则指令不执行且输出不更新。 缺省状态是置位。
Source	REAL	输入到算术指令的值(计算该值的以10为底的对数值)。 有效值=任一浮点数
输出参数：	**数据类型：**	**说明：**
EnableOut	BOOL	指令产生一有效结果。
Dest	REAL	算术运算的结果。输出影响算术状态标志。

说明： 源值必须大于0，否则会置位溢出状态位(S:V)。目的标签中的结果必须大于或等于-37.92978并小于或等于38.53184。

算术状态标志： 影响算术状态标志。

故障条件: 无

执行：

梯形图

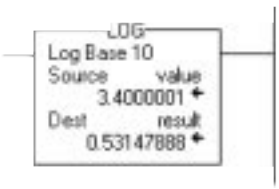
条件：	动作：
预扫描	梯级输出条件被设置为假。
梯级输入条件为假	梯级输出条件被设置为假。
梯级输入条件为真	控制器计算源值以10为底的对数值，并将结果存入目的标签中。 梯级输出条件被设置为真。
后期扫描	梯级输出条件被设置为假。

功能块

条件：	动作：
预扫描	不动作。
首次扫描指令	不动作。
首次执行指令	不动作。
EnableIn是清零状态	清零EnableOut位。
EnableIn是置位状态	执行指令。 置位EnableOut位。
后期扫描	不动作。

举例： 计算value的以10为底的对数值，并将计算结果存入result标签中。

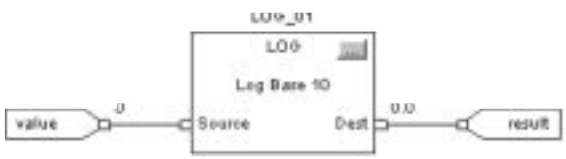
梯形图



结构文本

result := LOG(value);

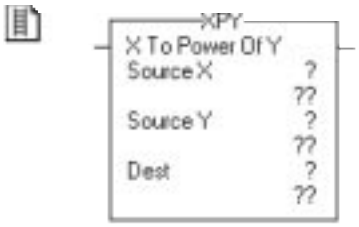
功能块



X的Y次幂 (XPY)

XPY指令计算源值A(X)的源值B(Y)次幂，并将结果存入目的标签中。

操作数：



梯形图

操作数：	数据类型：	格式：	说明：
Source x 源值	SINT INT DINT **REAL**	立即数 标签	计算该值的自然对数值
Source y 源值	SINT INT DINT **REAL**	立即数 标签	计算该值的自然对数值
Destination 目的	SINT INT DINT **REAL**	标签	存储计算结果的标签

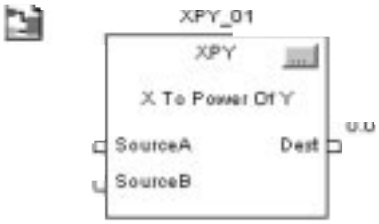
结构文本

```
dest := sourceX ** sourceY;
```

连续使用两个乘号“**”作为表达式内运算符。该表达式计算sourceX的sourceY次幂，并将结果存入dest中。

关于结构文本中表达式的语法信息，参见附录C。

功能块



操作数：	数据类型：	格式：	说明：
XPY标签	FBD_MATH	结构体	XPY结构体

FBD_MATH结构体

输入参数：	数据类型：	说明：
EnableIn	BOOL	使能输入。如果清零，则指令不执行且输出不更新。 缺省状态是置位。
SourceX	REAL	底数。 有效值=任一浮点数
SourceY	REAL	指数。 有效值=任一浮点数
输出参数：	**数据类型：**	**说明：**
EnableOut	BOOL	指令产生一有效结果。
Dest	REAL	算术运算的结果。输出影响算术状态标志。

说明：如果源X是负数，则源Y必须是整数值，否则会发生次要故障。

XPY 指令的运算法则：Destination = X**Y

控制器认为：x 0 =1 和 0 x =0。

算术状态标志：影响算术状态标志。

故障条件:

发生次要故障的条件：	故障类型：	故障代码：
源X是负数而源Y不是整数值	4	4

执行：执行：

梯形图

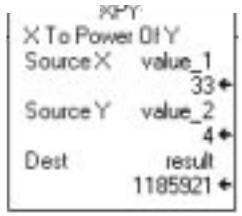
条件：	动作：
预扫描	梯级输出条件被设置为假。
梯级输入条件为假	梯级输出条件被设置为假。
梯级输入条件为真	控制器计算Source X的Source Y次幂，并将结果存入目的标签中。 梯级输出条件被设置为真。
后期扫描	梯级输出条件被设置为假。

功能块

条件：	动作：
预扫描	不动作。
首次扫描指令	不动作。
首次执行指令	不动作。
EnableIn 是清零状态	清零EnableOut位。
EnableIn 是置位状态	执行指令。 置位EnableOut位。
后期扫描	不动作。

举例：XPY指令计算value_1的value_2次幂，并将计算结果存入result 标签中。

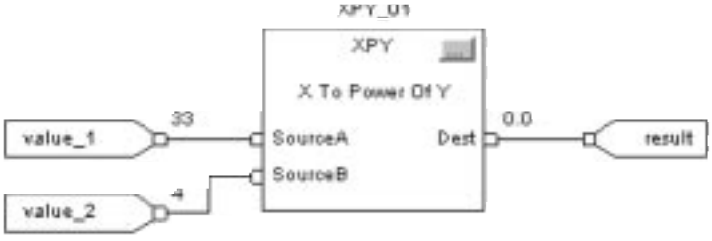
梯形图



结构文本

```
result := (value_1 ** value_2);
```

功能块



算术转换指令
(DEG，RAD，TOD，FRD，TRN ，TRUNC)

简介

数学转换指令可以转换数值。

如果用户需要：	所用指令：	可用语言：	参见页码：
将弧度转换成角度。	DEG	梯形图 结构文本 功能块	15-2
将角度转换成弧度。	RAD	梯形图 结构文本 功能块	15-4
将整数值转换成BCD码值。	TOD	梯形图 功能块	15-6
将BCD码值转换成整数值。	FRD	梯形图 功能块	15-9
舍去数值的小数部分	TRN TRUNC(1)	梯形图 结构文本 功能块	15-11

(1)仅结构文本

运算时用户可以混用数据类型，但是这样会损失精度也可能会发生取整误差，长指令的执行时间。检测S:V位，以确定结果是否被截短。

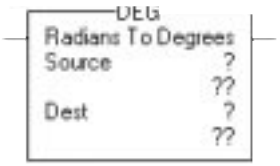
对于梯形图语言编程，则黑体字数据类型是最优数据类型。如果指令的操作数都用同一最优数据类型，则指令的执行速度快且占用内存少。典型最优数据类型是DINT或REAL数据类型。

转换成角度 (DEG)

DEG指令将源值(弧度)转换成角度，并将结果存入目的标签中。

操作数：

梯形图



操作数：	数据类型：	格式：	说明：
Source 源	SINT INT DINT **REAL**	立即数 标签	将该值转换成角度值
Destination 目的	SINT INT DINT REAL	标签	存储转换结果的标签

结构文本

```
dest := DEG(source);
```

将DEG用作函数。该函数将source转换为角度，并将结果存入dest中。

关于结构文本中表达式的语法信息，参见附录C。

功能块

操作数：	数据类型：	格式：	说明：
DEG标签	FBD_MATH_ADVANCED	结构体	DEG结构体

FBD_MATH_ADVANCED结构体

输入参数：	数据类型：	说明：
EnableIn	BOOL	使能输入。如果清零，则指令不执行且输出不更新。 缺省状态是置位。
Source	REAL	输入到转换指令的值(将该值转换成角度值)。 有效值=任一浮点数
输出参数：	数据类型：	说明：
EnableOut	BOOL	指令产生一有效运算结果
Dest	REAL	算术运算的结果。输出影响算术状态标志。

说明： DEG指令所用运算法则是：
Source*180/π(此处π= 3.141593)

算术状态标志： 影响算术状态标志。

故障条件: 无

执行：

梯形图

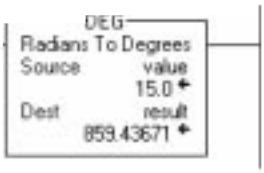
条件：	动作：
预扫描	梯级输出条件被设置为假。
梯级输入条件为假	梯级输出条件被设置为假。
梯级输入条件为真	控制器将源值转换成角度值，并将结果存入目的标签。 梯级输出条件被设置为真。
后期扫描	梯级输出条件被设置为假。

功能块

条件：	动作：
预扫描	不动作。
首次扫描指令	不动作。
首次执行指令	不动作。
EnableIn 是清零状态	清零EnableOut位。
EnableIn 是置位状态	执行指令。 置位EnableOut位。
后期扫描	不动作。

举例： 将value转换成角度值并将转换结果存入result 标签中。

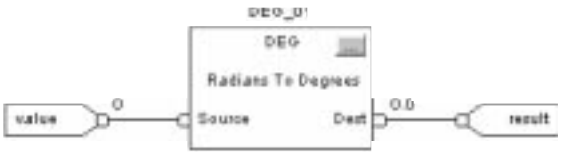
梯形图



结构文本

result := DEG(value);

功能块

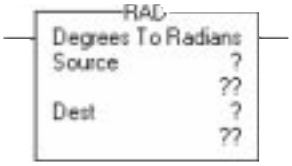


转换成弧度 (RAD)

RAD指令将源值(角度)转换成弧度，并将结果存入目的标签中。

操作数：

梯形图



操作数：	数据类型：	格式：	说明：
Source 源	SINT INT DINT **REAL**	立即数 标签	将该值转换成弧度值
Destination 目的	SINT INT DINT REAL	标签	存储转换结果的标签

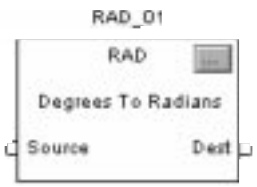
结构文本

```
dest := RAD(source);
```

将RAD用作函数。该函数将source转换为弧度，并将结果存入dest中。

关于结构文本中表达式的语法信息，参见附录C。

功能块



操作数：	数据类型：	格式：	说明：
RAD标签	FBD_MATH_ADVANCED	结构体	RAD结构体

FBD_MATH_ADVANCED结构体

输入参数：	数据类型：	说明：
EnableIn	BOOL	使能输入。如果清零，则指令不执行且输出不更新。 缺省状态是置位。
Source	REAL	输入到转换指令的值(将该值转换成弧度值)。 有效值=任一浮点数
输出参数：	**数据类型：**	**说明：**
EnableOut	BOOL	指令运算产生一有效结果。
Dest	REAL	算术运算的结果。输出影响算术状态标志。

说明：RAD指令所用运算法则：

Source• π /180 (此处π= 3.141593)

算术状态标志：影响算术状态标志。

故障条件: 无

执行：

梯形图

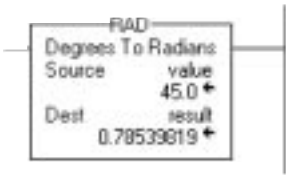
条件：	动作：
预扫描	梯级输出条件被设置为假。
梯级输入条件为假	梯级输出条件被设置为假。
梯级输入条件为真	控制器将源值转换成弧度值，并将结果存入目的标签。 梯级输出条件被设置为真。
后期扫描	梯级输出条件被设置为假。

功能块

条件：	动作：
预扫描	不动作。
首次扫描指令	不动作。
首次执行指令	不动作。
EnableIn是清零状态	清零EnableOut位。
EnableIn是置位状态	执行指令。 置位EnableOut位。
后期扫描	不动作。

举例: 将value转换成弧度值并将转换结果存入result标签中。

梯形图



结构文本

result := RAD(value);

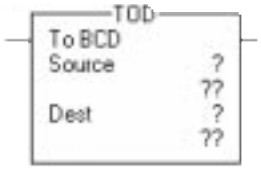
功能块

转换成BCD码 (TOD)

TOD指令将十进制值(0≤源值≤99,999,999)转换成BCD码，并将结果存入目的标签中。

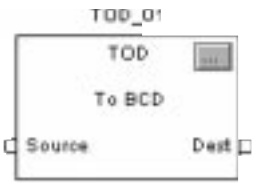
操作数：

梯形图



操作数：	数据类型：	格式：	说明：
Source 源	SINT INT DINT SINT或INT数据类型标签用零-填充法转换成DINT类型的值。	立即数 标签	将该十进制值转换成BCD码值
Destination 目的	SINT INT DINT	标签	存储转换结果的标签

功能块



操作数：	数据类型：	格式：	说明：
TOD标签	FBD_CONVERT	结构体	TOD结构体

FBD_CONVERT结构体

输入参数：	数据类型：	说明：
EnableIn	BOOL	使能输入。如果清零，则指令不执行且输出不更新。 缺省状态是置位。
Source	DINT	输入到转换指令的值(将该值转换成BCD码值)。 有效值=任一浮点数
输出参数：	**数据类型：**	**说明：**
EnableOut	BOOL	指令运算产生一有效结果。
Dest	DINT	算术运算的结果。输出影响算术状态标志。

说明： BCD 是Binary Coded Decimal 的缩写，用4位二进制符号表示十进制数值的一位(0-9)。

如果源值是负数，则指令会产生次要故障并清零目的标签中的值。

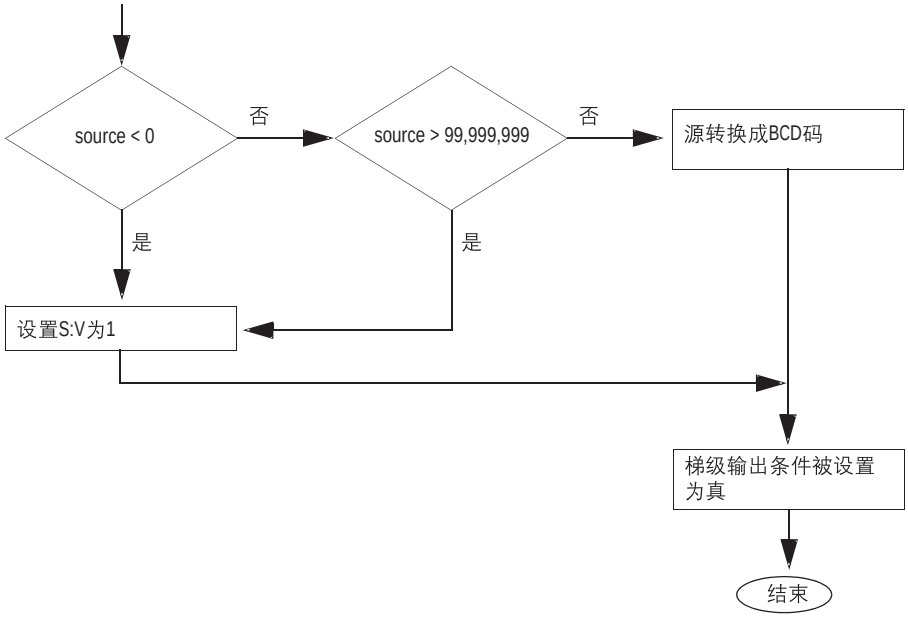
算术状态标志： 影响算术状态标志。

故障条件：

产生次要故障的条件：	故障类型：	故障代码：
源值< 0	4	4

执行：

梯形图

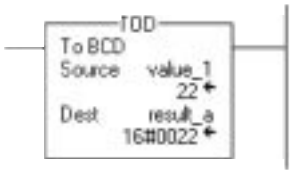
条件：	动作：
预扫描	梯级输出条件被设置为假。
梯级输入条件为假	梯级输出条件被设置为假。
梯级输入条件为真	
梯级输入条件为真	控制器将源值转换成BCD码值，并将结果存入目的标签。 梯级输出条件被设置为真。
后期扫描	梯级输出条件被设置为假。

功能块

条件：	动作：
预扫描	不动作。
首次扫描指令	不动作。
首次执行指令	不动作。
EnableIn是清零状态	清零EnableOut位。
EnableIn是置位状态	执行指令。 置位EnableOut位。
后期扫描	不动作。

执行：将value_1的值转换成BCD码值并将转换结果存入result_a标签中。

梯形图

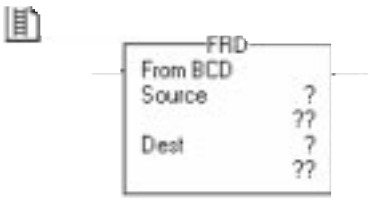


功能块

转换成整数 (FRD)

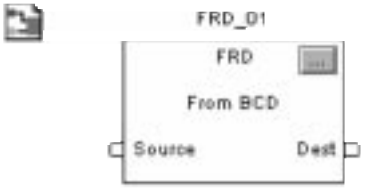
FRD指令将BCD码值(源值)转换成十进制整数值，并将结果存入目的标签中。

操作数：



梯形图

操作数：	数据类型：	格式：	说明：
Source 源	SINT INT DINT	立即数 标签	将该值转换成十进制整数值
	SINT或INT数据类型标签用零-填充法转换成DINT类型的值。		
Destination 目的	SINT INT DINT	标签	存储转换结果的标签



功能块

操作数：	数据类型：	格式：	说明：
FRD标签	FBD_CONVERT	结构体	FRD结构体

FBD_CONVERT结构体

输入参数：	数据类型：	说明：
EnableIn	BOOL	使能输入。如果清零，则指令不执行且输出不更新。 缺省状态是置位。
Source	REAL	输入到转换指令的值(将该值转换成整数值)。 有效值= 任一浮点数
输出参数：	数据类型：	说明：
EnableOut	BOOL	指令运算产生一有效结果
Dest	REAL	算术运算的结果。输出影响算术状态标志。

说明：FRD指令将BCD码值(源值)转换成十进制整数值，并将结果存入目的标签。

算术状态标志：影响算术状态标志。

故障条件: 无 无

举例：

梯形图

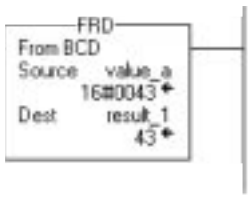
条件：	动作：
预扫描	梯级输出条件被设置为假。
梯级输入条件为假	梯级输出条件被设置为假。
梯级输入条件为真	控制器将源值转换成十进制值，并将结果存入目的标签。 梯级输出条件被设置为真。
后期扫描	梯级输出条件被设置为假。

功能块

条件：	动作：
预扫描	不动作。
首次扫描指令	不动作。
首次执行指令	不动作。
EnableIn是清零状态	清零EnableOut位。
EnableIn是置位状态	执行指令。 置位EnableOut位。
后期扫描	不动作。

举例：将value_a的值转换成十进制整数值并将转换结果存入result_1标签中。

梯形图



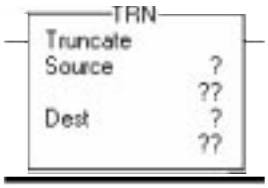
功能块

截短 (TRN)

TRN指令删除(截短)源值的小数部分，并将结果存入目的标签。

操作数：

梯形图



操作数：	数据类型：	格式：	说明：
Source 源	**REAL**	立即数 标签	要截短的值
Destination 目的	SINT INT **DINT** **REAL**	标签	存储转换结果的标签

结构文本

```
dest := TRUNC(source);
```

将TRUNC用作函数。该函数截短source值，并将结果存入dest中。

关于结构文本中表达式的语法信息，参见附录C。

功能块

操作数：	数据类型：	格式：	说明：
TRN标签	FBD_TRUNCATE	结构体	TRN结构体

FBD_TRUNCATE结构体

输入参数：	数据类型：	说明：
EnableIn	BOOL	使能输入。如果清零，则指令不执行且输出不更新。 缺省状态是置位。
Source	REAL	输入到转换指令的值(将该值转换成整数值)。 有效值=任一浮点数
输出参数：	数据类型：	说明：
EnableOut	BOOL	指令运算产生一有效结果。
Dest	DINT	算术运算的结果。输出影响算术状态标志。

说明： 截短并不是对数值取整；该指令只保留非小数部分而与小数部分的值没有关系。

算术状态标志： 影响算术状态标志。

故障条件： 无

执行：

梯形图

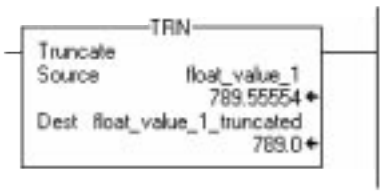
条件：	动作：
预扫描	梯级输出条件被设置为假。
梯级输入条件为假	梯级输出条件被设置为假。
梯级输入条件为真	控制器删除源值的小数部分，并将结果存入目的标签。 梯级输出条件被设置为真。
后期扫描	梯级输出条件被设置为假。

功能块

条件：	动作：
预扫描	不动作。
首次扫描指令	不动作。
首次执行指令	不动作。
EnableIn 是清零状态	清零EnableOut 位。
EnableIn 是置位状态	执行指令。 置位EnableOut 位。
后期扫描	不动作。

举例： 删除float_value_1 的小数部分，保留非小数部分，并将结果存入float_value_1_truncated 中。

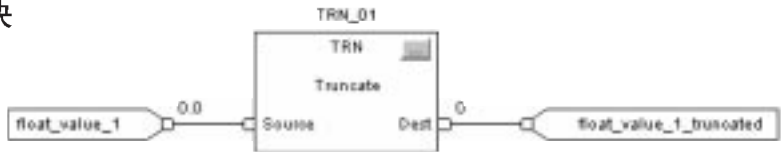
梯形图



结构文本

```
float_value_1_truncated := TRUNC(float_value_1);
```

功能块



ASCII 串行口指令 (ABL，ACB，ACL，AHL，ARD，ARL，AWA，AWT)

简介

用ASCII串行口指令读/写ASCII字符。

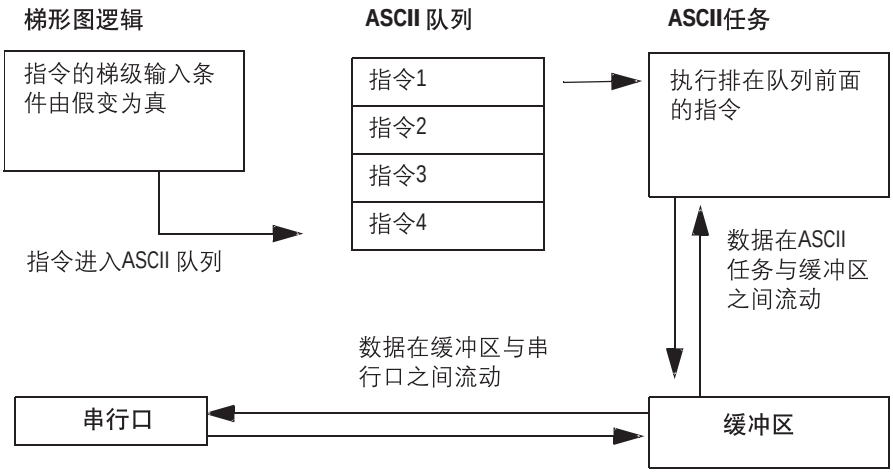
重要事项

要使用ASCII串行口指令时，用户必须先组态控制器的串行口。详见 Logix5000 Controllers Common Procedures一书，出版号1756-PM001。

如果用户需要：	例如：	所用指令：	可用语言：	参见页码：
确定缓存区何时包含终止符	检查包含终止符的数据	ABL	梯形图 结构文本	16-5
计算缓冲区中字符的数量	在读缓冲区之前检测所需的字符数量	ACB	梯形图 结构文本	16-8
清空缓冲区 清除正在执行或在队列中的ASCII串行口指令	• 在启动时将原有数据从缓冲区中移除 • 使缓冲区与设备同步	ACL	梯形图 结构文本	16-10
获取串行口控制信号线的状态 接通或关断DTR信号 接通或关断RTS信号	使调制解调器挂起	AHL	梯形图 结构文本	16-12
读取固定数量的字符	每次传送时，从发送字符的设备读取相同数量的字符	ARD	梯形图 结构文本	16-16
读取数量变化的字符，直到找到第一个结束符为止。	每次传送时，从发送字符的设备读取不同数量的字符	ARL	梯形图 结构文本	16-19
发送字符并且自动添加一个或两个附加字符作为数据结束的标志	发送的信息始终用相同的终止符(1或多个)	AWA	梯形图 结构文本	16-23
发送字符	发送的信息使用变化的终止符	AWT	梯形图 结构文本	16-28

指令执行

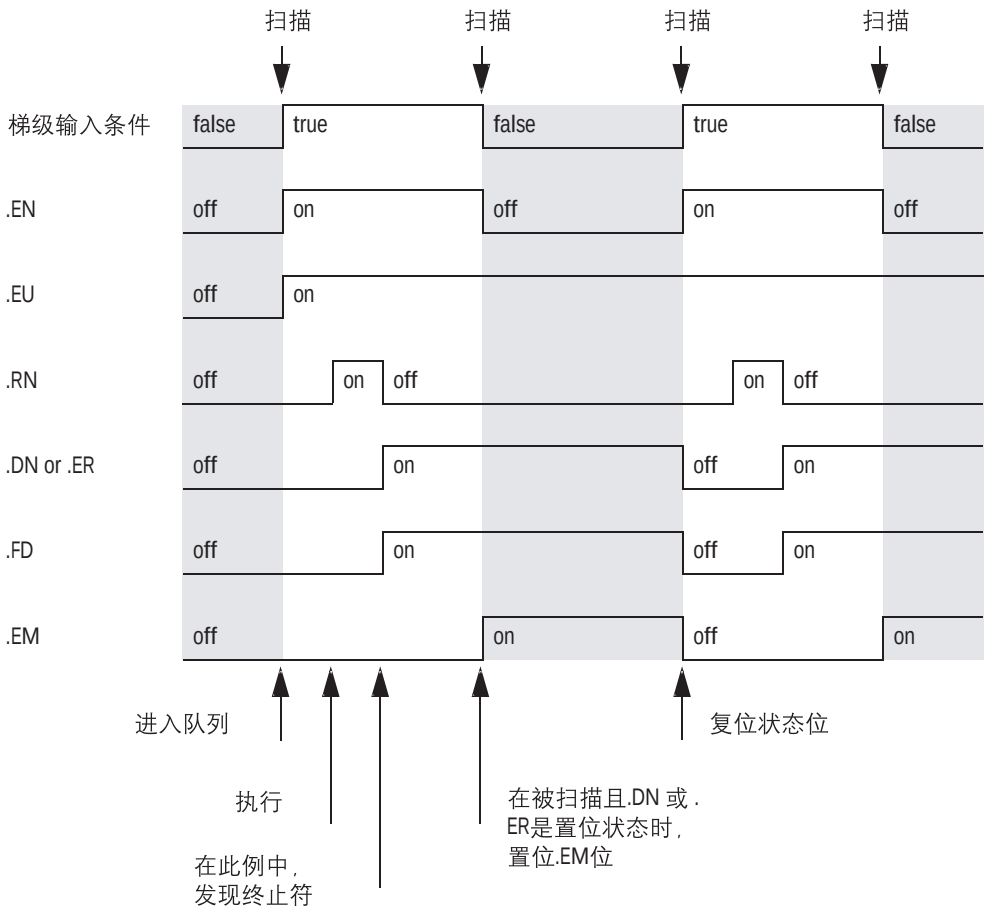
ASCII串行口指令的执行与逻辑扫描异步：



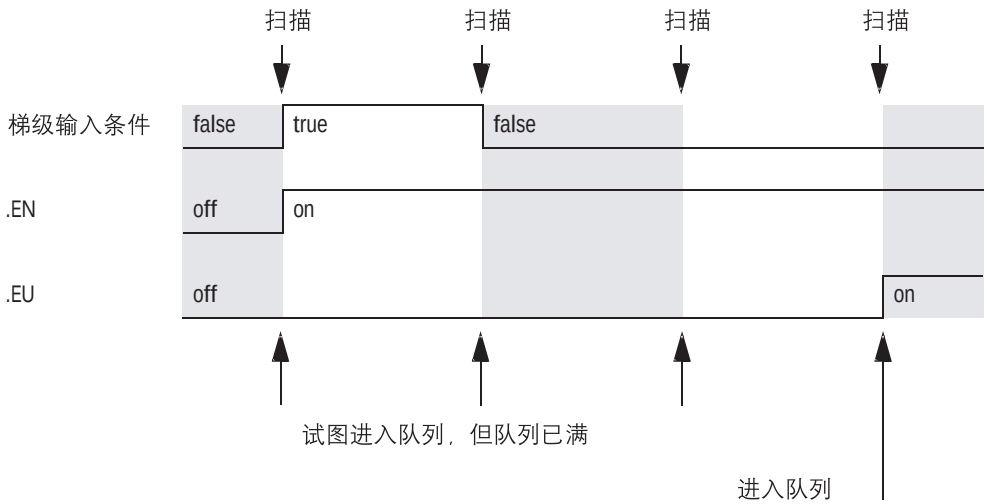
每条ASCII串行口指令(ACL指令除外) 用SERIAL_PORT_CONTROL 结构体实现下列功能：

- 控制指令的执行
- 提供与指令有关的状态信息

下列时序图描述了ABL指令在检测缓冲区中终止符时状态位的变化情况。



ASCII队列中最多有16条指令。如果队列已满，而接下来的扫描中仍有指令要进入队列，会出现如下情形：

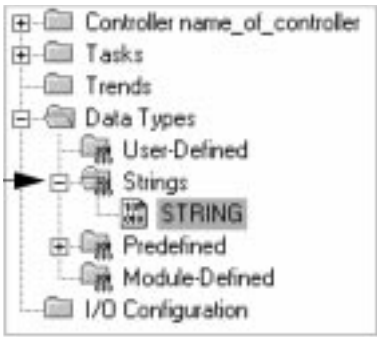


ASCII指令错误代码

如果ASCII串行口指令执行失败，则 SERIAL_PORT_CONTROL 结构体的 ERROR 项会包含下列十六进制的错误代码之一：

十六进制代码：	含义：
16#2	调制解调器离线。
16#3	在通讯期间CTS信号丢失。
16#4	串行口处于系统工作模式。
16#A	在指令执行之前，.UL已经置位。使指令不能执行。
16#C	控制器从运行模式变为编程模式。使ASCII串行口指令停止执行并且清零队列。
16#D	控制器属性对话框中，用户协议选项卡，缓冲区长度或响应模式参数被更改并应用。这使得ASCII串行口指令停止执行并且清零队列。
16#E	执行ACL指令。
16#F	组态串行口从用户模式变为系统模式。这样使ASCII串行口指令停止执行并且清零串行口指令队列。
16#51	字符串标签的长度值是负数或大于字符串标签的数据长度 。
16#54	Serial Port Control Length(串行口控制长度)大于缓冲区的长度。
16#55	Serial Port Control Length(串行口控制长度)是负数或大于源或目的标签长度。

字符串数据类型



用户要将ASCII字符存储在字符串数据类型的标签内。

- 可以直接使用缺省的字符串数据类型。该数据类型最多可以存储82个字符。.
- 用户可以创建新的字符串数据类型，这样可以自定义存储字符个数。

如果要创建新字符串数据类型，参见Logix5000 Controllers Common Procedures一书，出版号是1756-PM001。

每个字符串数据类型都包含下列项：

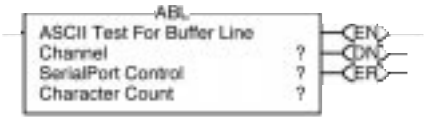
名称：	数据类型：	描述：	注释：
LEN	DINT	字符串中字符数量	只要有下列情况发生，LEN值就自动更新为新的字符数量： • 用字符串浏览器对话框输入字符 • 用指令读取、转换或处理字符串 LEN指示了当前字符串的长度。DATA项可能包含附加的、旧的字符，这些字符不包含在LEN计数的字符内。
DATA	SINT数组	字符串中ASCII字符	• 如果要访问字符串中的字符，则要寻址标签名称。 例如，要访问string_1标签中的字符，则输入string_1。 • DATA 数组每个元素包含一个字符。 • 用户也可以创建新的字符串数据类型，所存储的字符个数可以自己定义。

ASCII 测试缓冲区行(ABL)

ABL指令对缓冲区中的字符进行计数，直至且包含第一个终止符。

操作数：

梯形图



操作数：	数据类型：	格式：	说明：
Channel 通道	DINT	立即数 标签	0
Serial Port Control 串行口控制	SERIAL_PORT_ CONTROL	标签	指令操作的控制标签
Character Count 字符数	DINT	立即数	0 在指令执行期间，显示缓冲区中字符的个数，其中包括第一个终止符

结构文本

```
ABL(Channel,
    serialPortControl);
```

该指令操作数与梯形图ABL指令的操作数相同。用户通过SERIAL_PORT_CONTROL结构体的.POS项访问字符值。

SERIAL_PORT_CONTROL 结构体

助记符：	数据类型：	说明：
.EN	BOOL	使能位表明指令被使能。
.EU	BOOL	队列位表明指令进入ASCII 队列。
.DN	BOOL	完成位表明指令何时完成，但与梯形图扫描异步。
.RN	BOOL	运行位表明正在执行指令。
.EM	BOOL	空位表明指令已经完成，但与逻辑扫描同步。
.ER	BOOL	错误位表明指令执行时失败(发生错误)。
.FD	BOOL	发现位表明指令发现终止符或字符。
.POS	DINT	位置值确定缓冲区中字符的数量，包括第一个终止符集合。指令在发现终止符或字符后返回这个数值。
.ERROR	DINT	错误码包含指示引起错误原因的十六进制代码。

说明：ABL指令搜索缓冲区中第一个终止符集合。如果该指令发现终止符，它将：

- 置位发现位.FD
- 计数缓冲区中的字符个数，包括第一个终止符集

在控制器属性对话框中，用户协议选项卡，定义指令用来做终止符的ASCII字符。

要编程ABL 指令，应遵循如下规则：

1. 将控制器的串行口组态成用户模式并定义用作终止符的字符。

2. 该指令为边沿触发指令：
   - 用梯形图语言编程，每次ABL指令所在梯级由假变真都对缓冲区中字符进行计数。
   - 用结构文本编程，以指令为条件，因此只有边沿触发时才执行。参见附录C。

算术状态标志：不影响

故障条件：无

执行：

条件：	梯形图动作：	结构文本动作：
预扫描	梯级输出条件被设置为假。	不动作。
梯级输入条件为假	梯级输出条件被设置为假。	不影响
梯级输入条件为真	当梯级输入由清零到置位触发时，指令执行。 梯级输出条件被设置为假。	不影响
EnableIn被置位	不影响	EnableIn一直被置位。 指令执行。
指令执行	指令计数缓冲区内的字符。 使能位.EN被置位。 其余状态位，除.UL位，均被清零。 指令尝试进入ASCII队列。	
后期扫描	梯级输出条件被设置为假。	不动作。

举例：连续检测缓冲区中的终止符。

梯形图

结构文本

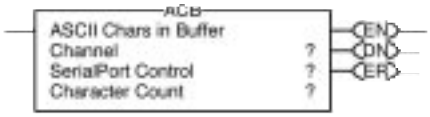
```
ABL(0,MV_line);
```

计数缓冲区中ASCII字符数(ACB)

ACB 指令计数存储在缓冲区中的字符数。

操作数：操作数：

梯形图



操作数：	数据类型：	格式：	说明：
Channel 通道	DINT	立即数 标签	0
Serial Port Control 串行口控制	SERIAL_PORT_CONTROL	标签	指令操作的控制标签
Character Count 字符数	DINT	立即数	0 在指令执行期间，显示缓冲区中字符的数量。

结构文本

```
ACB(Channel
    SerialPortControl);
```

该指令的操作数与梯形图中ACB指令的操作数相同。可是，用户通过访问SERIAL_PORT _CONTROL结构体的位置值.POS项可指定字符数量值，而不是通过将该值包含在操作数列表中。

SERIAL_PORT_CONTROL结构体

助记符：	数据类型：	说明：
.EN	BOOL	使能位表明指令被使能。
.EU	BOOL	队列位表明指令进入ASCII队列。
.DN	BOOL	完成位指明指令何时完成，但与逻辑扫描异步。
.RN	BOOL	运行位表明指令正在执行。
.EM	BOOL	空位表明指令已经完成，但与逻辑扫描同步。
.ER	BOOL	错误位表示指令执行失败(发生错误)。
.FD	BOOL	发现位表示指令发现了一个字符。
.POS	DINT	位置值确定缓冲区中字符的数量，包括终止符的第一个字符集合在内。
.ERROR	DINT	错误码包含表明引起错误原因的十六进制代码。

说明：ACB指令计数缓冲区中的字符数量。

在程序中使用ACB 指令，应遵循如下规则：

1. 将控制器的串行口组态成用户模式并定义用作终止符的字符。

2. 该指令为边沿触发指令：
   - 用梯形图语言编程，每次ABL指令所在梯级由假变真都对缓冲区中字符进行计数。
   - 用结构文本编程，以指令为条件，因此只有边沿触发时才执行。参见附录C。

算术状态标志：不影响

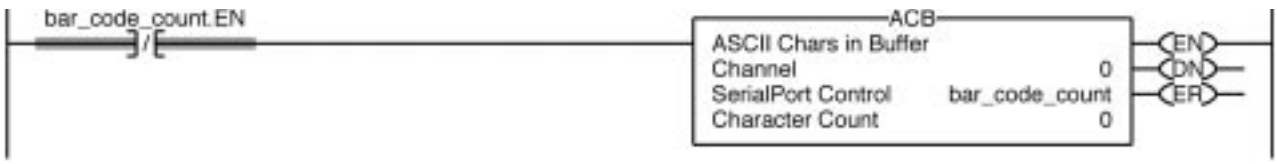
故障条件：无

执行：

条件：	梯形图动作：	结构文本动作：
预扫描	梯级输出条件被设置为假。	不动作。
梯级输入条件为假	梯级输出条件被设置为假。	不影响
梯级输入条件为真	当梯级输入由清零到置位触发时，指令执行。 梯级输出条件被设置为真。	不影响
EnableIn被置位	不影响	EnableIn一直被置位。 指令执行。
指令执行	指令计数缓冲区内的字符。 使能位.EN被置位。 其余状态位，除.UL位，均被清零。 指令尝试进入ASCII队列。	
后期扫描	梯级输出条件被设置为假。	不动作。

举例：连续检测缓冲区中的字符数

梯形图：



结构文本

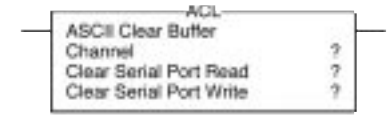
ACB(0,bar_code_count);

ASCII 清空缓冲区(ACL)

ACL 指令立即清空ASCII缓冲区和ASCII指令队列。

操作数：

梯形图



操作数：	数据类型：	格式：	说明：
Channel 通道	DINT	立即数 标签	0
Clear Serial Port Read 清空串行口读缓冲区	BOOL	立即数 标签	如果要清空缓冲区并将ARD和ARL指令从队列中移出，则输入“是”
Clear Serial Port Write 清空串行口写缓冲区	BOOL	立即数 标签	如果要将AWA和AWT指令从队列中移出，则输入“是”。

结构文本

```
ACL(Channel,
    ClearSerialPortRead,
    ClearSerialPortWrite);
```

操作数与梯形图ACL指令的操作数相同。

说明： ACL指令立即执行下列动作中的一项或两项：

- 清空缓冲区中的字符并将读指令从ASCII 队列中移出。(清空缓冲区内的字符，并清除读指令的ASCII队列。)
- 将写指令从 ASCII 队列中清除。(清除写指令的ASCII队列。)

在程序中使用ACL 指令，应遵循如下规则：

1. 配置控制器的串行口：

如果有下列应用：	则：
使用 ARD 或 ARL 指令	选择用户模式
未使用 ARD 或 ARL 指令	选择系统模式或用户模式均可

2. 如果要确定指令是否从队列中移出或中止，可以查验下列相应的指示：
   - 错误位.ER被置位
   - 错误代码.ERROR项为16#E

算术状态标志： 不影响

故障条件： 无

执行：

条件：	梯形图动作：	结构文本动作：
预扫描	梯级输出条件被设置为假。	不动作。
梯级输入条件为假	梯级输出条件被设置为假。	无影响
梯级输入条件为真	指令执行。 梯级输出条件被设置为真。	无影响。
EnableIn 被置位	无影响	EnableIn 一直被置位。 指令执行。
指令执行	指令清除指定指令和缓冲区。	
后期扫描	梯级输出条件被设置为假。	不动作。

举例：当控制器进入运行模式时，清空缓冲区和ASCII队列。

梯形图

结构文本

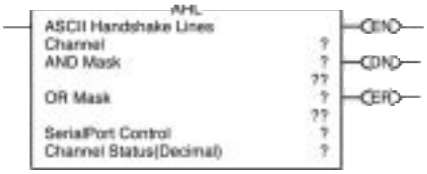
```
osri_1.InputBit := S:FS;
OSRI(osri_1);

IF (osri_1.OutputBit) THEN
        ACL(0,0,1);
END_IF;
```

ASCII 握手信号线(AHL)

AHL 指令获取控制控制信号线的状态并接通或关断DTR和RTS信号。

操作数：

梯形图



操作数：	数据类型：	格式：	输入值：
Channel 通道	DINT	立即数 标签	通道—0
AND Mask 与屏蔽	DINT	立即数 标签	参见说明
OR Mask 或屏蔽	DINT	立即数 标签	
Serial Port Control 串行口控制	SERIAL_PORT_CONTROL	标签	串行口控制—指令操作的控制标签
Channel Status (Decimal) 通道状态(十进制)	DINT	立即数	通道状态(十进制)—0 在执行期间，显示控制信号线状态。

控制信号线状态：	检测下列位：
CTS	0
RTS	1
DSR	2
DCD	3
DTR	4
接收XOFF字符	5

结构文本

```
AHL(Channel,ANDMask,ORMask,
    SerialPortControl);
```

操作数与梯形图AHL指令的操作数相同。可是，用户可通过访问SERIAL_PORT_CONTROL结构体的位置值.POS项指定通道状态值，而不是通过将该值包含在操作数列表中。

SERIAL_PORT_CONTROL 结构体

助记符:	数据类型:	说明:
.EN	BOOL	使能位—表明指令被使能。
.EU	BOOL	队列位—表明指令进入ASCII队列。
.DN	BOOL	完成位—表明指令已经完成时，与逻辑扫描异步。
.RN	BOOL	运行位—表明指令正在执行。
.EM	BOOL	空位—表明指令已经完成，但与逻辑扫描同步。
.ER	BOOL	错误位—表示指令执行失败(发生错误)。
.FD	BOOL	发现位—此指令不使用该位。
.POS	DINT	位置值—存储控制信号线状态。
.ERROR	DINT	错误码—包含表明引起错误原因的十六进制代码。

说明: AHL指令可完成下列工作:

- 获得串行口控制信号线的状态
- 接通或关断数据终端准备(DTR)信号
- 接通或关断发送信号请求(RTS)

在程序中使用AHL指令，应遵循如下规则:

1. 配置控制器的串行口:

如果有下列应用:	则:
使用ARD 或ARL指令	选择用户模式
未使用ARD 或ARL指令	选择系统模式或用户模式均可

2. 用下表选择ANDMask 和 ORMask操作数的合适值:

DTR:	RTS:	输入ANDMask值:	输入ORMask值:
关断	关断	3	0
	接通	1	2
	不变	1	0
接通	关断	2	1
	接通	0	3
	不变	0	1
不变	关断	2	0
	接通	0	2
	不变	0	0

3．指令为边沿触发指令：

- 用梯形图语言编程，每次梯级输入条件由清零变为置位触发时，指令都应该执行。
- 用结构文本编程，以指令为条件，因此只有边沿触发时，指令才执行。参见附录C。

算术状态标志：不影响

故障条件：

故障类型：	故障代码：	引起故障原因：	恢复方法：
4	57	由于串行口设置成无握手信号导致AHL指令执行失败	任选其一： • 改变串行口控制信号线的设置 • 删除AHL 指令

执行：

条件：	梯形图动作：	结构文本动作：
预扫描	梯级输出条件被设置为假。	不动作。
梯级输入条件为假	梯级输出条件被设置为假。	无影响
梯级输入条件为真	当梯级输入条件由清零变为置位被触发时，指令执行。 梯级输出条件被设置为真。	无影响
EnableIn被置位	无影响	EnableIn一直被置位。 指令执行。
指令执行	指令获取控制信号线状态，并接通或关断DTR和RTS信号。 使能位.EN被置位。 其余状态位，除.UL外，均被清零。 指令尝试进入ASCII队列。	
后期扫描	梯级输出条件被设置为假。	不动作。

举例：当get_control_line_status 被置位时，获取串行口控制信号线的状态并保存在通道状态操作数中。要观察指定控制信号线的状态，监控串行口控制标签并详述POS 项。

梯形图

结构文本

```
osri_1.InputBit := get_control_line_status;
OSRI(osri_1);

IF (osri_1.OutputBit) THEN
        AHL(0,0,0,serial_port);
END_IF;
```

ASCII读 (ARD)

ARD 指令将字符从缓冲区中移出并存储在目的标签中。

操作数：

梯形图

操作数：	数据类型：	格式：	输入值：	注释：
Channel通道	DINT	立即数 标签	0	通道
Destination目的	String SINT INT DINT	标签	存储被移出(读)的字符的标签： • 对于字符串数据类型，输入标签名称。 • 对于SINT, INT, 或DINT数组，输入数组中的第一个元素。	目的标签 • 如果用户要对字符进行比较，转换或处理，则要用字符串数据类型。 • 字符串数据类型是指： • 缺省STRING 数据类型 • 由用户自定义的任何新字符串数据类型
Serial Port Control 串行口控制	SERIAL_PORT_ CONTROL	标签	指令操作的控制标签	串行口控制
Serial Port Control Length 串行口控制长度	DINT	立即数	要移送(读)到目的标签的字符个数。	串行口控制长度 • 串行口控制长度必须小于或等于目的标签长度。 • 如果用户要设置串行口控制长度等于目的标签长度，则输入0值。
Characters Read字符读	DINT	立即数	0	在执行期间，显示要读入的字符数量。

结构文本

```
ARD(Channel,Destination,
    SerialPortControl);
```

操作数与梯形图ARD指令的操作数相同。可是，用户可通过访问SERIAL_PORT_CONTROL结构体的长度值.LEN项和位置值.POS项指定串行口长度和读入的字符数量，而不是通过操作数列表中的数值。

SERIAL_PORT_CONTROL 结构体

助记符:	数据类型:	说明:
.EN	BOOL	使能位—表明指令被使能。
.EU	BOOL	队列位—表明指令进入ASCII队列。
.DN	BOOL	完成位—表明指令已经完成时，与逻辑扫描异步。
.RN	BOOL	运行位—表明指令正在执行。
.EM	BOOL	空位—表明指令已经完成，但与逻辑扫描同步。
.ER	BOOL	错误位—表明指令执行失败(发生错误)。
.FD	BOOL	发现位—此指令不使用该位。
.LEN	DINT	长度值—表明要移动(读)到目的标签的字符数量。
.POS	DINT	位置值—显示读到目的标签的字符位置(数量)。
.ERROR	DINT	错误码—包含表明引起错误原因的十六进制代码。

说明：ARD指令将指定数量的字符从缓冲区中移出并存储到目的标签中。

- ARD指令会连续执行直到指定的字符(串行口控制长度)都已经被移出。
- 在ARD指令执行期间，不执行其它的ASCII串行口指令 。

在程序中使用AHL指令，应遵循如下规则：

1. 将控制器的串行口配置成用户模式。

2. 用一条 ACB 指令的执行结果来触发ARD指令。这样可以防止在等待请求字符期间，ARD指令在ASCII队列中挂起 。

3. 该指令为边沿触发指令：
   - 用梯形图语言编程，每次梯级输入条件由清零变为置位触发时，指令都应该执行。
   - 用结构文本编程，以指令为条件，因此只在边沿触发时执行。参见附录C。

4. 如果当该指令完成后要触发后续事件，则检测.EM 位。

算术状态标志：不影响

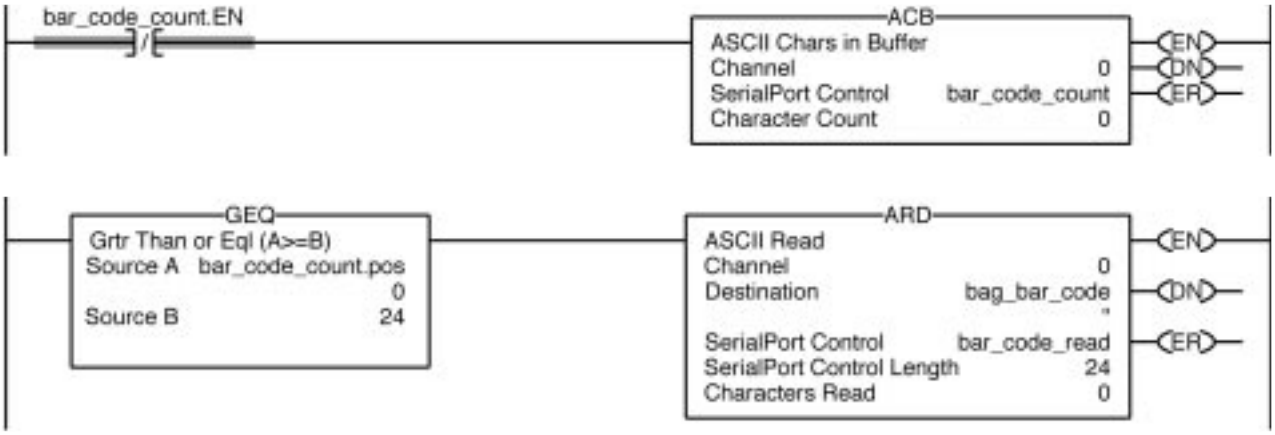
故障条件：无

执行：

条件：	梯形图动作：	结构文本动作：
预扫描	梯级输出条件被设置为假。	不动作。
梯级输入条件为假	梯级输出条件被设置为假。	无影响
梯级输入条件为真	当梯级输入条件由清零变为置位时，指令执行。 梯级输出条件被设置为真。	无影响
EnableIn 被置位	无影响	EnableIn 一直被置位。 指令执行。
指令执行	指令从缓冲区中移出字符并存储在目的标签中。 使能位.EN 被置位。 其余状态位，除.UL 外，均被清零。 指令尝试进入ASCII队列。	
后期扫描	梯级输出条件被设置为假。	不动作。

举例： 一个条形码读入器将条形码发送到控制器的串行口(通道0) 。每个条形码包含24个字符。为了确定控制器已正确接受了条形码， ACB 指令连续对缓冲区中的字符进行计数。当缓冲区中至少有24个字符时，表明控制器已经接收到了条形码。 ARD 指令将条形码移送到bag_bar_code标签的DATA项，它是一个字符串。

梯形图



结构文本

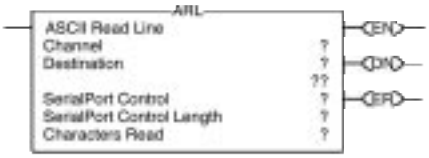
```
ACB(0,bar_code_count);
IF bar_code_count.POS >= 24 THEN
    bar_code_read.LEN := 24;
    ARD(0,bag_bar_code,bar_code_read);
END_IF;
```

ASCII 读行 (ARL)

ARL 指令将指定字符从缓冲区中移出并将其存储在目的标签中。

操作数：

梯形图



操作数：	数据类型：	格式：	输入值：	注释：
Channel通道	DINT	立即数 标签	0	通道
Destination目的	STRING SINT INT DINT	标签	用来存储被移出(读)的字符的标签： • 对于字符串数据类型，输入标签名称。 • 对于 SINT, INT, 或DINT 数组，输入数组中的第一个元素。	目的标签 • 如果用户要对字符进行比较，转换或处理，则要用字符串数据类型。 • 字符串数据类型是指： • 缺省的STRING 数据类型 • 由用户自定义的任何新字符串数据类型。
Serial Port Control 串行口控制	SERIAL_PORT_ CONTROL	标签	指令操作的控制标签	串行口控制
Serial Port Control Length 串行口控制长度	DINT	立即数	如果没有发现终止符，则读取字符数量的最大值	串行口控制长度 • 输入字符数量最大值，它包括任何将要接受的信息(也就是指如果没有发现终止符，何时停止读取数据)。 例如，如果信息的长度范围是3到6个字符，则输入6。 • 串行口控制长度必须小于或等于目的标签长度。 • 如果用户要设置串行口控制长度等于目的标签长度，则输入0值。
Characters Read 字符读	DINT	立即数	0	读字符 在指令执行期间，显示要读入的字符数量。

结构文本

```
ARL(Channel,Destination,
    SerialPortControl);
```

操作数与梯形图ARL指令的操作数相同。可是，用户可通过访问SERIAL_PORT_CONTROL结构体的长度值.LEN项和位置值.POS项来指定读字符值，而不是通过将该值包含在操作数列表中。

SERIAL_PORT_CONTROL结构体

助记符：	数据类型：	说明：
.EN	BOOL	使能位—表明指令被使能。
.EU	BOOL	队列位—表明指令进入ASCII队列。
.DN	BOOL	完成位—表明指令已经完成时，与逻辑扫描异步。
.RN	BOOL	运行位—表明指令正在执行。
.EM	BOOL	空位—表明指令已经完成，但与逻辑扫描同步。
.ER	BOOL	错误位—表明指令执行失败(发生错误)。
.FD	BOOL	发现位—此指令不使用该位。
.LEN	DINT	长度值—表明要移动(读)到目的标签的字符数量。
.POS	DINT	位置值—显示读取的字符数量。
.ERROR	DINT	错误码—包含表明引起错误原因的十六进制代码。

说明： ARL指令将字符从缓冲区中移出并存储到目的标签中：

- ARL指令会连续执行直到出现下列任一情况：
  —终止符的第一个字符集
  —完成(串行口控制长度)中指定数量的字符
- 在ARL指令执行期间，不执行其它的ASCII串行口指令。

在程序中使用ARL指令，应遵循如下规则：

1. 配置控制器的串行口：
   a. 选择用户模式
   b. 定义用做终止符的字符。

2. 用一条 ABL 指令的执行结果触发ARL指令。这样可以防止在等待终止符期间，ARL指令在ASCII队列中挂起 。

3. 该指令为边沿触发指令：
   - 用梯形图语言编程，每次梯级输入条件由清零变为置位被触发时，指令都应该执行。
   - 用结构文本编程，以指令为条件，因此只有边沿触发时才执行。参见附录C。

4. 如果要在该指令完成后触发发后续事件，则检测.EM 位。

算术状态标志：不影响

故障条件：无

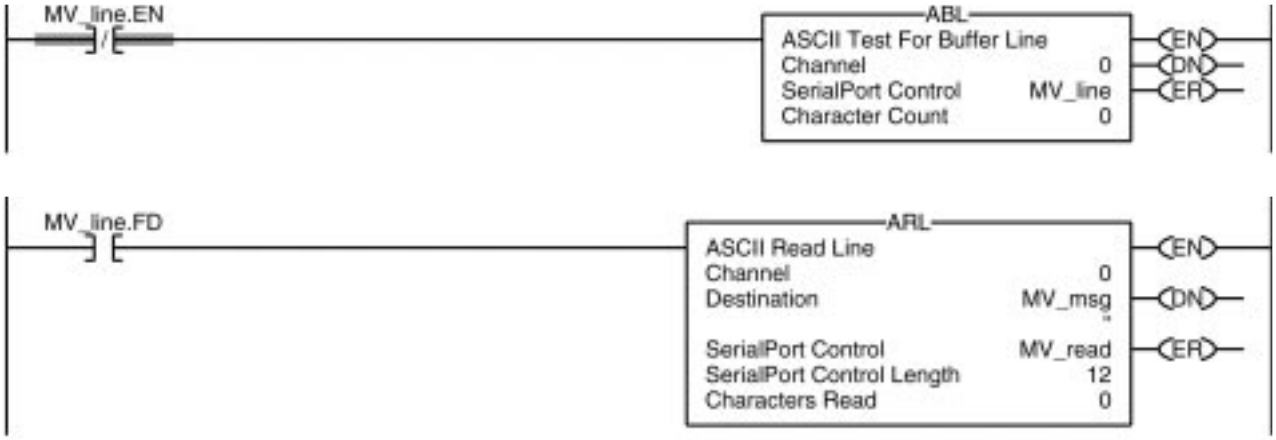
执行：

条件：	梯形图动作：	结构文本动作：
预扫描	梯级输出条件被设置为假。	不动作。
梯级输入条件为假	梯级输出条件被设置为假。	无影响
梯级输入条件为真	当梯级输入条件由清零变为置位时，指令执行。 梯级输出条件被设置为真。	无影响
EnableIn被置位	无影响	EnableIn一直被置位。 指令执行。
指令执行	指令从缓冲区中移出字符并存储在目的标签中。 使能位.EN被置位。 剩余状态位，除.UL外，均被清零。 指令尝试进入ASCII队列。	
后期扫描	梯级输出条件被设置为假。	不动作。

举例：连续测试缓冲区中来自信息显示终端的信息。因为每个信息都以回车符($r)结束， 在控制器属性对话框，用户协议对话框中回车符被设置为终止符。如果ABL指令发现一个回车符，则置位发现位.FD。

如果在ABL指令发现一个回车符 (MV_line.FD 置位)时，表示控制器已经收到一个完整的信息。ARL指令就将字符从缓冲区中移出，直到并且包括这个回车符，并将这些字符存储在MV_msg标签的DATA项中。

梯形图



结构文本

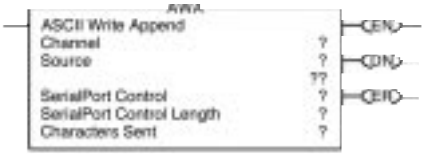
```
ABL(0,MV_line);

osri_1.InputBit := MVLine.FD;
OSRI(osri_1);

IF (osri_1.OutputBit) THEN
        mv_read.LEN := 12;
        ARL(0,MV_msg,MV_read);
END_IF;
```

ASCII写附加(AWA)

AWA指令将源标签中的指定数量的字符发送到一连接到串行口的设备中，并填加1或2个预定义的字符。

操作数：

梯形图



操作数：	数据类型：	格式：	输入值：	注释：
Channel通道	DINT	立即数 标签	0	通道
Source 源	STRING SINT INT DINT	标签	包含要发送字符的标签： • 对于字符串数据类型，输入标签名称。 • 对于 SINT, INT, 或DINT 数组，输入数组中的第一个元素。	源 • 如果用户要比较，转换或复制字符，则要用字符串数据类型。 • 字符串数据类型是指： • 缺省STRING 数据类型 • 由用户自定义的任何新字符串数据类型
Serial Port Control 串行口控制	SERIAL_PORT_ CONTROL	标签	指令操作的控制标签	串行口控制
Serial Port Control Length 串行口控制长度	DINT	立即数	要发送的字符数量	串行口控制长度 • 串行口控制长度必须小于或等于源标签长度。 • 如果用户想让串行口控制长度与源标签中的字符数相等，则输入0值。
Characters Send字符发送	DINT	立即数	0	发送字符 在执行期间，显示发送的字符数量。

结构文本

```
AWA(Channel,Source,
    SerialPortControl);
```

操作数与梯形图AWA指令的操作数相同。可是，用户可通过访问SERIAL_PORT_CONTROL结构体的长度值.LEN项和位置值.POS项来指定串行口控制长度值和发送字符值，而不是通过将该值包含在操作数列表中。

SERIAL_PORT_CONTROL 结构体

助记符：	数据类型：	说明：
.EN	BOOL	使能位—表明指令被使能。
.EU	BOOL	队列位—表明指令进入ASCII队列。
.DN	BOOL	完成位—表明指令已经完成时，与逻辑扫描异步。
.RN	BOOL	运行位—表明指令正在执行。
.EM	BOOL	空位—表明指令已经完成，但与逻辑扫描同步。
.ER	BOOL	错误位—表明指令执行失败(发生错误)。
.FD	BOOL	发现位—此指令不使用该位。
.LEN	DINT	长度值—表明要移动(读)到目的标签的字符数量。
.POS	DINT	位置值—显示读取的字符数量。
.ERROR	DINT	错误码—包含表明引起错误原因的十六进制代码。

说明： AWA指令将：

- 将源标签中由指定数量(串行口控制长度)的字符发送到与控制器串行口连接的目标设备中。
- 在字符(附加)的末尾添加一个或两个字符，该字符是在控制器属性对话框中用户协议选项卡定义。

在程序中使用AWA指令，应遵循如下规则：

1．配置控制器的串行口：

a. 在用户应用程序中包括ARD或ARL 指令吗?

如果：	则：
是	选择用户模式
不是	选择或系统模式或用户模式

b. 定义添加到数据末尾的字符。

2．该指令为边沿触发指令：

- 用梯形图语言编程，每次梯级输入条件由清零变为置位触发时，指令都应该执行。
- 用结构文本编程，以指令为条件，因此只边沿触发。参见附录C。

3. 每次指令执行时希望发送相同数量的字符吗？

如果：	则：
是	在串行口控制长度参数中输入要发送的字符数量。
不是	在指令执行之前，将源标签中LEN 项移送到串行口控制标签的LEN 项中。

算术状态标志：不影响

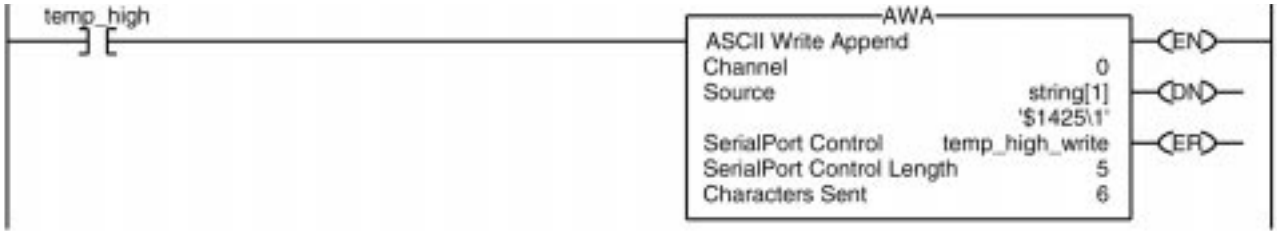
故障条件：无

执行：

条件：	**梯形图动作**：	**结构文本动作**：
预扫描	梯级输出条件被设置为假。	不动作。
梯级输入条件为假	梯级输出条件被设置为假。	无影响
梯级输入条件为真	当梯级输入条件由清零变为置位时，指令执行。 梯级输出条件被设置为真。	无影响
EnableIn 被置位	无影响	EnableIn 一直被置位。 指令执行。
指令执行	指令发送指定数量的字符并添加一或二个预定义的字符。 使能位.EN 被置位。 其余状态位，除.UL 外，均被清零。 指令尝试进入ASCII 队列。	
后期扫描	梯级输出条件被设置为假。	不动作。

例1：当温度值超过上限时 (temp_high是置位状态)，AWA指令向连接到控制器串行口的显示终端发送一条信息。该信息存储在string[1]标签的DATA项中包含5个字符(该信息包含string[1]标签DATA项中的5个字符)，作为一个字符串。 ($14算做一个字符，它是Ctrl-T 字符的十六进制编码。) 此指令也将在控制器属性中定义的(附加)字符发送出去(该指令同时发送(附加)控制器属性 中定义的字符)。在该例中，AWA 指令发送一个回车符 ($0D)，用来作为信息结束的标志。

梯形图

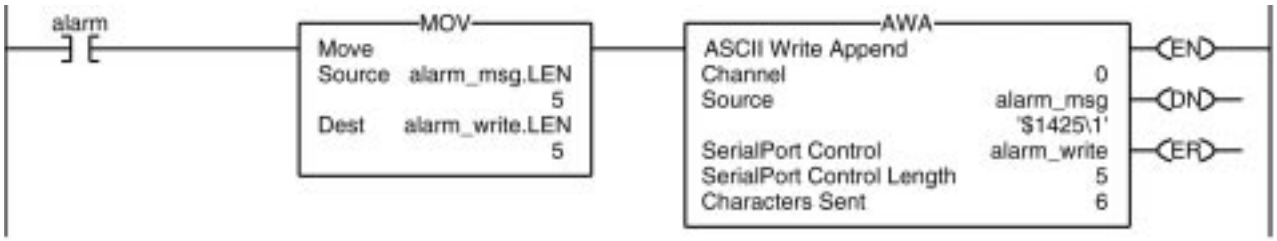


结构文本

```
IF temp_high THEN

    temp_high_write.LEN :=5;

    AWA(0,string[1],temp_high_write);

    temp_high := 0;

END_IF;
```

例2：当alarm信号被置位时，AWA 指令发送 alarm_msg 标签中指定的字符连同附加的终止符。因为alarm_msg 标签中的字符数量变化，所以该梯级首先将字符串的长度值(alarm_msg.LEN) 传送到 AWA 指令的串行口控制长度 (alarm_write.LEN)中。在alarm_msg中，$14 算做一个字符。它是 Ctrl-T字符的十六进制代码。

梯形图



结构文本

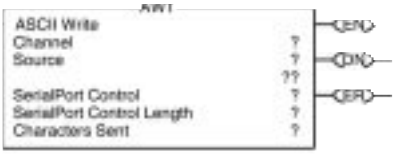
```
osri_1.InputBit := alarm;
OSRI(osri_1);

IF (osri_1.OutputBit) THEN
        alarm_write.LEN :=alarm_msg.LEN;
        AWA(0,alarm_msg,alarm_write);
END_IF;
```

ASCII写(AWT)

AWT指令将源标签中指定数量的字符发送到串行口设备中 。

操作数：

梯形图



操作数：	数据类型：	格式：	输入值：	注释：
Channel通道	DINT	立即数 标签	0	通道
Source 源	STRING SINT INT DINT	标签	包含要发送的字符的标签： • 对于字符串数据类型，输入标签名称。 • 对于SINT, INT, 或DINT 数组，输入数组中的第一个元素。	源 • 如果用户要比较，转换或操作字符，则要用字符串数据类型。 • 字符串数据类型是指： • 缺省STRING数据类型 • 由用户自定义的任何新字符串 数据类型
Serial Port Control 串行口控制	SERIAL_PORT_ CONTROL	标签	指令操作的控制标签	串行口控制
Serial Port Control Length 串行口控制长度	DINT	立即数	要发送的字符数量	串行口控制长度 • 串行口控制长度必须小于或等于源标签长度。 • 如果用户需要设置串行口控制长度与源标签中的字符数相等，则输入0值。
Characters Sent字符发送	DINT	立即数	0	发送字符 在执行期间，显示发送的字符数量。

结构文本

```
AWT(Channel,Source,
    SerialPortControl);
```

操作数与梯形图AWA指令的操作数相同。可是，用户可通过访问SERIAL_PORT_CONTROL结构体的长度值.LEN项和位置值.POS项来指定串行口控制长度值和发送字符值，而不是通过将该值包含在操作数列表中。

SERIAL_PORT_CONTROL 结构体

助记符：	数据类型：	说明：
.EN	BOOL	使能位—表明指令被使能。
.EU	BOOL	队列位—表明指令进入ASCII队列。
.DN	BOOL	完成位—表明指令已经完成时，与逻辑扫描异步。
.RN	BOOL	运行位—表明指令正在执行。
.EM	BOOL	空位—表明指令已经完成，但与逻辑扫描同步。
.ER	BOOL	错误位—表明指令执行失败(发生错误)。
.FD	BOOL	发现位—此指令不使用该位。
.LEN	DINT	长度值—表示要发送的字符数量。
.POS	DINT	位置值—显示已经发送字符数量。
.ERROR	DINT	错误码—包含表明引起错误原因的十六进制代码。

说明： AWT指令将源标签中指定数量的字符(串行口控制长度)发送到与控制器串行口相连接的设备中。

在程序中使用AWT指令，应遵循如下规则：

1. 配置控制器的串行口：

如果用户程序：	则：
使用ARD或ARL指令	选择用户模式
未使用ARD或ARL指令	选择系统模式和用户模式均可

2. 该指令为边沿触发指令：
   - 用梯形图语言编程，每次梯级输入条件由清零变为置位触发时，指令都应该执行。
   - 用结构文本编程，以指令为条件，因此只有边沿触发才，指令才执行。参见附录C。

3. 用户希望每次执行指令时都发送相同数量的字符吗？

如果：	则：
是	在串行口控制长度中输入要发送的字符数量。
不是	在指令执行之前，将源标签中的LEN 项移送到串行口控制标签的LEN 项中。

算术状态标志：不影响

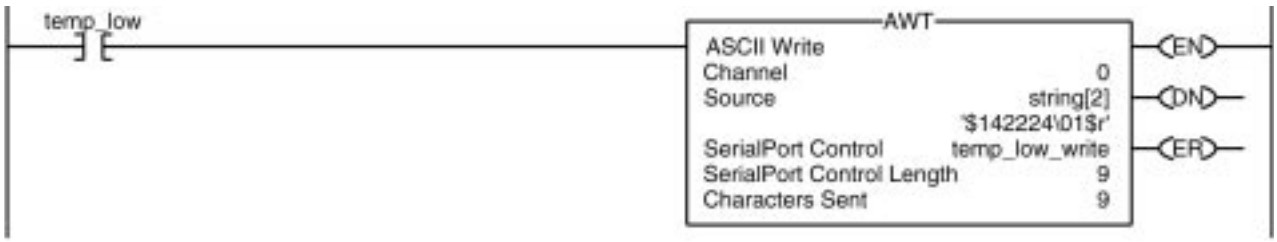
故障条件：无

执行：

条件：	梯形图动作：	结构文本动作：
预扫描	梯级输出条件被设置为假。	不动作。
梯级输入条件为假	梯级输出条件被设置为假。	无影响
梯级输入条件为真	当梯级输入条件由清零变为置位时，指令执行。 梯级输出条件被设置为真。	无影响
EnableIn被置位	无影响	EnableIn一直被置位。 指令执行。
指令执行	指令发送指定数量的字符。 使能位.EN被置位。 其余状态位，除.UL外，均被清零。 指令尝试进入ASCII队列。	
后期扫描	梯级输出条件被设置为假。	不动作。

例1：当温度值达到下限时 (temp_ low被置位)， AWT指令向连接到控制器串行口的显示终端发送一个信息。该信息以字符串的形式存储在string[2]标签的DATA中包含9个字符(该信息包含string[2]标签DATA项的九个字符)。 ($14 算做一个字符，它是Ctrl-T 字符的十六进制编码。) 最后字符是一个回车符($0D)，用来作为信息结束的标志。

梯形图

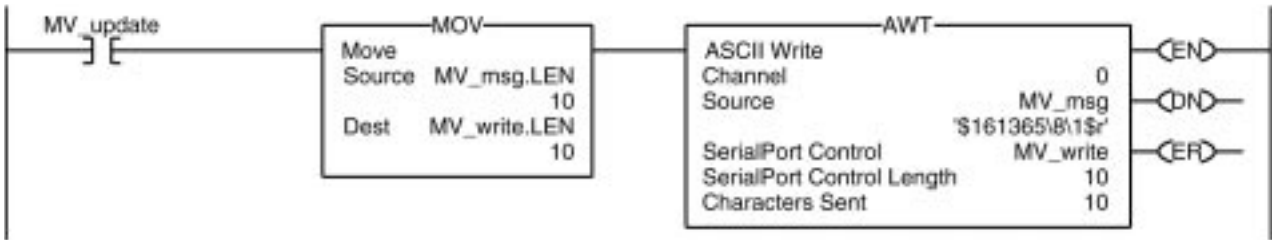


结构文本

```
osri_1.InputBit := temp_low;
OSRI(osri_1);
IF (osri_1.OutputBit) THEN
        temp_low_write.LEN := 9;
        AWT(0,string[2],temp_low_write);
END_IF;
```

例2：当MV_update被置位时，AWT 指令发送 MV_msg 标签中的指定字符。因为 MV_msg 标签中的字符数变化，所以该梯级首先将字符串的长度值(MV_msg.LEN) 传送到 AWA 指令的MV_write.LEN项中。在MV_msg中，$16 算做一个字符。它是 Ctrl-V字符的十六进制代码。

梯形图



结构文本

```
osri_1.InputBit := MV_update;
OSRI(osri_1);

IF (osri_1.OutputBit) THEN
        MV_write.LEN := Mv_msg.LEN;
        AWT(0,MV_msg,MV_write);
END_IF;
```

注释：

ASCII 字符串指令
(CONCAT，DELETE，FIND，INSERT，MID)

简介

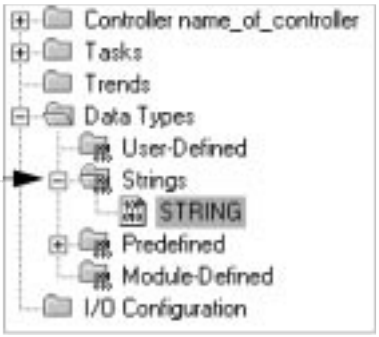
ASCII字符串指令用于修改和创建ASCII字符串。

如果用户需要：	例如：	所用指令：	可用语言：	参见页码：
在字符串末尾添加字符	向一字符串添加终止符或分隔符	CONCAT	梯形图 结构文本	17-3
从字符串中删除字符	从一字符串中删除头字符或控制符	DELETE	梯形图 结构文本	17-5
确定一个子字符串的起始字符	定位字符串中的一组字符	FIND	梯形图 结构文本	17-7
在字符串中插入字符	用变量创建一字符串	INSERT	梯形图 结构文本	17-9
从字符串中抽取字符	从条形码中抽取信息	MID	梯形图 结构文本	17-11

用户也可以用下列指令来比较或转换ASCII字符：

如果用户需要：	所用指令：	参见页码：
比较两组字符串是否一致	CMP	4-2
查看一组字符是否与指定字符相等	EQU	4-7
查看一组字符是否与指定字符不等	NEQ	4-38
查看一组字符是否等于或大于指定字符	GEQ	4-11
查看一组字符是否大于指定字符	GRT	4-15
查看一组字符是否等于或小于指定字符	LEQ	4-19
查看一组字符是否小于指定字符	LES	4-23
重新排列INT, DINT, 或 REAL 标签中的字节	SWPB	6-19
在一字符串数组中查找字符串	FSC	7-19
将字符转换成SINT, INT, DINT 或 REAL 值	STOD	4-18
将字符转换成REAL值	STOR	6-18
将SINT, INT, DINT 或 REAL 值转换成ASCII字符串	DTOS	8-18
将REAL值转换成ASCII字符串	RTOS	10-18

字符串数据类型



将ASCII字符存储在字符串数据类型的标签内。

- 可以直接使用缺省的字符串数据类型。该数据类型最多可以存储82个字符。
- 用户也可以创建新字符串数据类型，这样所存储的字符个数可以自定义。

如果要创建新字符串数据类型，参见《Logix5000 控制器通用步骤(Logix5000 Controllers Common Procedures)》一书，出版号是1756-PM001。

每个字符串数据类型都包含如下项：

名称：	数据类型：	说明：	注释：
LEN	DINT	字符串中字符的数量	只要有下列情况发生，LEN值就自动更新为新的字符数量： • 用字符串浏览器对话框输入字符 • 用指令读取、转换或处理字符串 LEN 指示的是当前字符串的长度。DATA项可能包含额外的，原有字符，这些字符不包含在LEN所计数的字符内。
DATA	SINT数组	字符串的ASCII字符	• 如果要访问字符串中的字符，则要寻址标签名称。例如，要访问string_1标签中的字符，则输入string_1 。 • 每个DATA 数组元素包含一个字符 • 用户也可以创建新的字符串数据类型，所存储的字符个数可以自定义。

连接字符串 (CONCAT)

CONCAT 指令在字符串的末尾添加ASCII字符。

操作数：

梯形图

操作数：	数据类型：	格式：	输入：	注释：
Source A源A	STRING	标签	包含初始字符的标签	字符串数据类型是指： • 缺省字符串数据类型 • 任何用户自定义的新字符串数据类型
Source B源B	STRING	标签	包含结尾字符的标签	
Destination目的	STRING	标签	存储指令执行结果	

结构文本

该指令操作数与梯形图中CONCAT指令操作数相同。

说明： CONCAT 指令将源A和源B中的字符结合在一起，结果存放在目的标签中。

- 源A中的字符在前，接下来是源B中的字符
- 源A中的字符保持不变，除非源A标签与目的标签相同。

算术状态标志： 不影响

故障条件：

类型：	代码：	原因：	恢复方法：
4	51	字符串标签LEN值大于字符串标签DATA的长度。	1. 检查是否有其它指令正在向字符串标签LEN项写入数值。 2. 在LEN值中输入字符串所包含的字符数量。

执行：

条件：	梯形图动作：	结构文本动作：
预扫描	梯级输出条件被设置为假。	无动作。
梯级输入条件为假	梯级输出条件被设置为假。	无影响
梯级输入条件为真	执行指令。 梯级输出条件被设置为真。	无影响
EnableIn被置位	不影响	EnableIn一直被置位。 执行指令。
指令执行	指令连接字符串。	
后期扫描	梯级输出条件被设置为假。	无动作。

举例：要触发一个MessageView信息显示终端的信息，控制器必须发送一个包含信息号、节点号的ASCII字符串。String_1 中包含信息号。当 add_node 被置位时，CONCAT 指令将node_num_ascii(节点号)中的字符添加到string_1 中字符的末尾并将结果存储到msg标签中。

梯形图

结构文本

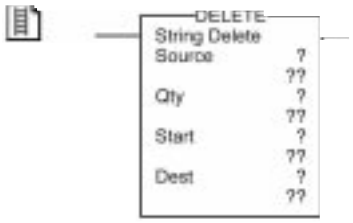
```
IF add_node THEN
    CONCAT(string_1,node_num_ascii,msg);
    add_node := 0;
END_IF;
```

删除字符串指令 (DELETE)

DELETE 指令将ASCII字符从字符串中删除。

操作数：

梯形图



操作数：	数据类型：	格式：	输入：	注释：
Source 源	STRING	标签	包含要删除字符的字符串标签	字符串数据类型是指： • 缺省的字符串数据类型 • 任何用户自己创建的新字符串数据类型
Quantity 数量	SINT INT DINT	立即数 标签	要删除的字符数量	起始字符位置加上字符数量必须小于或等于源中DATA 项的长度。
Start 起始位置	SINT INT DINT	立即数 标签	要删除的第一个字符的位置。	输入一个范围在1到源中数据项长度之间的一个数值。
Destination 目的	STRING	标签	存储操作结果的标签	

结构文本

```
DELETE(Source,Qty,Start,
    Dest);
```

说明：该指令操作数与梯形图中DELETE指令操作数相同。

DELETE指令将源值中的一组字符删除(或移出)并将剩余的字符存放在目的标签中。

- 移出的字符由起始位置和数量决定。
- 除非源与目的使用相同的标签，否则源中的字符保持不变。

算术状态标志：不影响

故障条件：

类型：	代码：	原因：	恢复方法：
4	51	字符串标签的长度值大于其数据项长度。	1. 检查是否有其它指令正在向字符串标签的长度项写入数值。 2. 长度项输入字符串实际所包含的字符数量
4	56	起始值或数量值无效。	1. 检查起始值是否是1到源中数据长度之间的一个数值。 2. 检查起始字符位置加上字符数量是否小于或等于源中数据项长度。

执行：

条件：	梯形图动作：	结构文本动作：
预扫描	梯级输出条件被设置为假。	无动作。
梯级输入条件为假	梯级输出条件被设置为假。	无影响
梯级输入条件为真	执行指令。 梯级输出条件被设置为真。	无影响
EnableIn被置位	无影响	EnableIn一直被置位。 执行指令。
指令执行	指令删除指定字符。	
后期扫描	梯级输出条件被设置为假。	无动作。

举例：来自终端的ASCII信息包含一个头字符。控制器读入数据后 (term_read.EM 被置位)，DELETE指令将头字符删除。

梯形图

结构文本

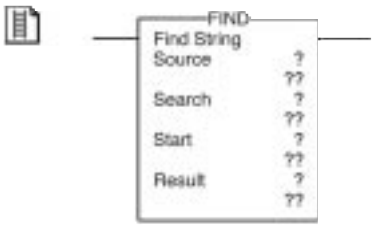
```
IF term_read.EM THEN
        DELETE(term_input,1,1,term_text);
        term_read.EM := 0;
END_IF;
```

查找字符串 (FIND)

FIND 指令定位一指定字符串在另一个字符串中的起始位置。

操作数：

梯形图



操作数：	数据类型：	格式：	输入：	注释：
Source 源	STRING	标签	要在其中搜索的字符串	字符串数据类型是指： • 缺省的字符串数据类型 • 任何用户自己创建的新字符串数据类型
Search 搜索项	STRING	标签	要查找的字符串	
Start 起始位置	SINT INT DINT	立即数 标签	在源标签中开始搜索的位置。	输入一个范围在1到源中数据长度之间的一个数值。
Result 结果	SINT INT DINT	标签	用于存储要查找的字符串的起始位置值。	

结构文本

```
FIND(Source,Search,Star_,
     Result);
```

该指令操作数与梯形图中FIND指令的操作数相同。

说明： FIND指令在源字符串中搜索要查找的字符串。如果指令发现了要查找的字符串，则在结果中给出所要查找的字符串在源字符串中的开始位置。

算术状态标志： 不影响

故障条件：

类型：	代码：	原因：	恢复方法：
4	51	字符串标签的长度值大于其数据项长度。	1. 检查是否有其它指令正在向字符串标签的长度项写入数值。 2. 在长度项输入字符串实际所包含的字符数量。
4	56	起始值无效。	检查起始值是否是1到源中数据长度之间的一个数值。

执行：

条件：	梯形图动作：	结构文本动作：
预扫描	梯级输出条件被设置为假。	无动作。
梯级输入条件为假	梯级输出条件被设置为假。	无影响
梯级输入条件为真	执行指令。 梯级输出条件被设置为真。	无影响
EnableIn 被置位	无影响	EnableIn 一直被置位。 执行指令。
指令执行	指令搜索指定字符。	
后期扫描	梯级输出条件被设置为假。	无动作。

举例：从信息显示终端传送来的信息包含几个信息段。用反斜线符号[\ ] 将每段信息分开。为了定位一段信息，用FIND指令搜索反斜线符号并将其位置记录在find_pos标签中。

梯形图

结构文本

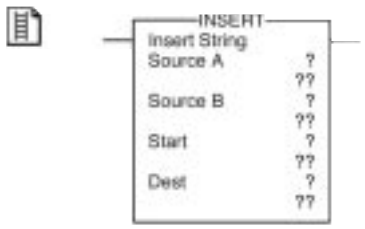
```
IF MV_read.EM THEN
    FIND(MV_msg,find,1,find_pos);
    MV_read.EM := 0;
END_IF;
```

插入字符串(INSERT)

FIND 指令将ASCII字符添加到一字符串中的指定位置。

操作数：

梯形图



操作数：	数据类型：	格式：	输入：	注释：
Source A 源A	STRING	标签	在该字符串中插入字符	字符串数据类型是指： • 缺省的 STRING 数据类型 • 任何用户自己创建的新字符串数据类型
Source B 源B	STRING	标签	将该字符串插入源A	
Start 起始位置	SINT INT DINT	立即数 标签	在源A中插入字符的位置	输入一个范围在1到源中DATA长度之间的一个数值。
Result 结果标签	STRING	标签	存储结果字符串	

结构文本

```
INSERT(SourceA,SourceB,
    Start,Dest);
```

该指令操作数与梯形图中INSERT指令操作数相同。

说明： INSERT指令将源B中的字符插入到源A中指定的位置，并将结果存放到目的标签中：

- 起始位置确定源B要插入到源A的位置。
- 除非源A与目的标签相同，否则源A保持不变。

算术状态标志： 不影响

故障条件：

类型：	代码：	原因：	恢复方法：
4	51	字符串标签的长度值大于其数据长度。	1．检查是否有其它指令正在向字符串标签的长度项写入数值。 2．在长度值中输入字符串实际所包含的字符数量。
4	56	起始值无效。	查看起始值是否是1到源中数据项长度之间的一个数值。

执行：

条件：	梯形图动作：	结构文本动作：
预扫描	梯级输出条件被设置为假。	无动作。
梯级输入条件为假	梯级输出条件被设置为假。	无影响
梯级输入条件为真	执行指令。 梯级输出条件被设置为真。	无影响
EnableIn 被置位	无影响	EnableIn 一直被置位。 执行指令。
指令执行	指令插入指定字符。	
后期扫描	梯级输出条件被设置为假。	无动作。

举例：当 temp_high 被置位时， INSERT 指令将string_2中的字符插入到string_1中位置2起始位置， 并将结果存储在string_3中。

梯形图

结构文本

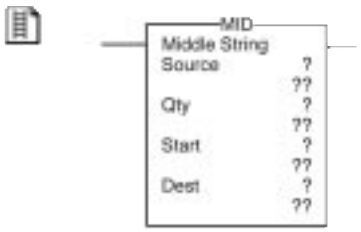
```
IF temp_high THEN
        INSERT(string_1,string_2,2,string_3);
        temp_high := 0;
END_IF;
```

抽取字符串(MID)

MID 指令从ASCII字符串中复制指定数量的字符并将其存储在另一个字符串中。

操作数：

梯形图



操作数：	数据类型：	格式：	输入：	注释：
Source 源	STRING	标签	要从中复制字符的字符串	字符串数据类型是指： • 缺省的字符串数据类型 • 任何用户自己创建的新字符串数据类型
Quantity 数量	SINT INT DINT	立即数 标签	需复制的字符数量	起始值加上数量值必须小于或等于源中数据项长度。.
Start 起始位置	SINT INT DINT	立即数 标签	要复制的第一个字符位置	输入范围在1到源中数据项长度之间的一个数值。
Destination 目的	STRING	标签	存储复制结果的字符串	

结构文本

```
MID(Source,Qty,Start,
    Dest);
```

该指令操作数与梯形图中MID指令的操作数相同。

说明：MID 指令从源中复制一组字符并将结果存储在目的标签中。

- 起始位置和数量确定要复制的字符。
- 除非源与目的使用相同标签，否则源值保持不变。

算术状态标志：不影响

故障条件：

类型：	代码：	原因：	恢复方法：
4	51	字符串标签的长度值大于其数据项长度。	1． 检查是否有其它指令正在向字符串标签的长度项写入数值。 2． 在长度项输入字符串实际所包含的字符数量。
4	56	起始值或数量值无效。	1. 查看起始值是否是1到源中数据长度之间的一个数值。 2. 查看起始值加上数量值是否小于或等于源中数据长度

执行：

条件：	梯形图动作：	结构文本动作：
预扫描	梯级输出条件被设置为假。	无动作。
梯级输入条件为假	梯级输出条件被设置为假。	无影响
梯级输入条件为真	执行指令。 梯级输出条件被设置为真。	无影响
EnableIn被置位	无影响	EnableIn一直被置位。 执行指令。
指令执行	指令从字符串中复制指定字符，并保存在另一个字符串中。	
后期扫描	梯级输出条件被设置为假。	无动作。

举例： 在机场行李传送带上，每个行李有一个条形码标签。其中条形码的 9 - 17 号字符表示航班号和行李的目的地机场号。在读过条形码之后(bag_read.EM 被置位），MID指令将航班号和目的地机场号复制到bag_flt_and_dest 字符串标签中。

梯形图

结构文本

```
IF bag_read.EM THEN
        MID(bar_barcode,9,9,bag_flt_and_dest);
        bag_read.EM := 0;
END_IF;
```

ASCII 转换指令 (STOD，STOR，DTOS，RTOS，UPPER，LOWER)

简介

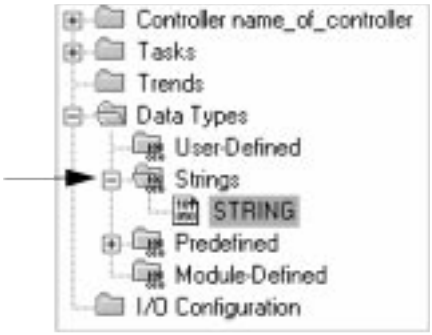
ASCII转换指令用来改变数据格式。

如果用户需要：	例如：	所用指令：	可用语言：	参见页码：
将用ASCII码表示的整数值转换成SINT, INT, DINT, 或REAL值	将来自称重秤或其它ASCII设备的值转换成用户程序逻辑中可以使用的整数	STOD	梯形图 结构文本	18-4
将用ASCII表示的浮点数值转换成一个REAL值	将来自称重秤或其它ASCII设备的值转换成用户程序逻辑中可以使用的REAL值。	STOR	梯形图 结构文本	18-6
将SINT, INT, DINT, 或REAL值转换成ASCII字符	将变量转换成ASCII字符，以便将数据发送到显示终端	DTOS	梯形图 结构文本	18-8
将REAL值转换成ASCII字符串	将变量转换成ASCII字符串，以便将数据发送到显示终端	RTOS	梯形图 结构文本	18-10
将ASCII字符串中的字母转换成大写形式	将操作员输入的字符转换成大写形式，以便在数组中查找	UPPER	梯形图 结构文本	18-12
将ASCII字符串中的字母转换成小写形式	将操作员输入的字符转换成小写形式，以便在数组中查找	LOWER	梯形图 结构文本	18-14

用户也可以用下列指令来比较或处理 ASCII字符：

如果用户需要：	使用如下指令：	参见页码：
在字符串末尾添加字符	CONCAT	17-3
从字符串中删除字符	DELETE	17-5
检测一个子-字符串的起始字符	FIND	17-7
在字符串中插入字符	INSERT	17-9
从字符串中抽取字符	MID	17-11
"重新排列INT, DINT,或REAL标签中的字节"	SWPB	6-19
比较两个字符串是否一致	CMP	4-2
查看一组字符是否与指定字符相等	EQU	4-7
查看一组字符是否与指定字符不等	NEQ	4-38
查看一组字符是否等于或大于指定字符	GEQ	4-11
查看一组字符是否大于指定字符	GRT	4-15
查看一组字符是否等于或小于指定字符	LEQ	4-19
查看一组字符是否小于指定字符	LES	4-23
在一字符串数组中查找字符串	FSC	7-19

字符串数据类型



用户可将ASCII字符存储在字符串数据类型的标签内。

- 可以直接使用缺省的字符串数据类型。该数据类型最多可存储82个字符。.
- 用户也可以创建新字符串数据类型，这样所存储的字符个数可以自定义。

要创建新字符串数据类型，参见《Logix5000控制器通用步骤(Logix5000 Controllers Common Procedures)》一书，出版号是1756-PM001。

每个字符串数据类型都包含如下项：

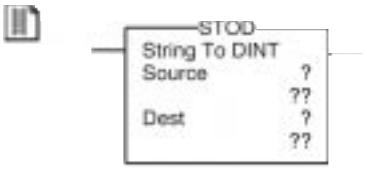
名称：	数据类型：	说明：	注释：
LEN	DINT	字符串中字符数	只要有下列情况发生，LEN值就自动更新为新字符数： • 用字符串浏览器对话框输入字符 • 用指令读取、转换或处理字符串 LEN 指示当前字符串长度。DATA项可能包含附加的，原有字符，这些字符不包含在LEN所计数的字符内。
DATA	SINT数组	字符串中的ASCII字符	• 如果要访问字符串中的字符，则要寻址标签名称。例如，要访问string_1标签中的字符，则输入string_1。 • 每个DATA 数组元素包含一个字符。 • 用户也可以创建新字符串数据类型，所存储的字符个数可自定义。

字符串转换成 DINT (STOD)

STOD 指令将用ASCII表示的整数转换成一个整数或REAL值。

操作数：

梯形图



操作数：	数据类型：	格式：	输入：	注释：
Source 源	STRING	标签	包含用ASCII表示的数值的标签	字符串数据类型是指： • 缺省的字符串数据类型 • 任何用户自己创建的新字符串数据类型
Destination 目的	SINT INT DINT REAL	标签	存储转换后的整数值	如果源值是一个浮点数，则指令只转换该数的整数部分，（与目的标签的数据类型无关）。

结构文本

```
STOD(Source,Dest);
```

该指令操作数与梯形图中STOD指令的操作数相同。

说明: STOD 指令把源标签中的值转换成整数并将结果存储于目的标签中。

- 指令可以转换正数也可以转换负数。
- 如果源字符串包含非数字字符，则STOD指令转换第一个与之临近的数：
  —指令将略过任何初始控制符或非数字字符（但是数字前面的负号除外）。
  —如果字符串中包含多组由分隔符(例如, /)分开的数字，则该指令只转换第一组数。

算术状态标志： 影响算术状态标志。

故障条件：

类型：	代码：	原因：	恢复方法：
4	51	字符串标签的长度值大于其DATA长度。	1． 检查是否有其它指令向字符串标签的长度项写入数值。 2． 在长度项输入字符串实际所包含的字符数量。
4	53	输出值超过目标数据类型的限制。	选择其一： • 或缩小ASCII值的长度。 • 或在目的标签中用较大的数据类型。

执行：

条件：	梯形图动作：	结构文本动作：
预扫描	梯级输出条件被设置为假。	无动作。
梯级输入条件为假	梯级输出条件被设置为假。	无影响
梯级输入条件为真	执行指令。 梯级输出条件被设置为真。	无影响
EnableIn被置位	无影响	EnableIn一直被置位。 执行指令。
指令执行	置位S:C位。 清零目的标签。 指令转换源值。 如果结果是0，则置位S:Z位。	无动作。
后期扫描	梯级输出条件被设置为假。	无动作。

举例：当MV_read.EM被置位时，STOD指令将MV_msg 中的第一组数字字符转换成整数值。指令将略过初始控制字符($06)，在分隔符处(\ )结束。

梯形图

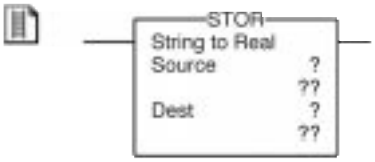
结构文本

```
IF MV_read.EM THEN
    STOD(MV_msg,MV_msg_nmbr);
    MV_read.EM := 0;
END_IF;
```

字符串转换成实数(STOR)

STOR 指令将用ASCII表示的浮点数转换成REAL型数值。

操作数：



梯形图操作数

操作数：	数据类型：	格式：	输入：	注释：
Source 源	STRING	标签	包含用ASCII表示的数值的标签	字符串数据类型是指： • 缺省的字符串数据类型 • 任何用户自己创建的新字符串数据类型
Destination 目的	REAL	标签	存储转换的REAL型数值	

```
STOR(Source,Dest);
```

结构文本

该指令操作数与梯形图中STOR指令的操作数相同。

说明： STOR指令把源值转换成实数并将结果存储于目的标签。

- 该指令既可以转换正数也可以转换负数。
- 如果源字符串包含非数字字符，则STOR指令将转换第一个与之临近的数字，包括小数点[.]：
  —指令将略过任何初始控制字符或非数字字符(但是数字前面的负号除外)。
  —如果字符串中包含多组由分隔符(例如.,/)分开的数字，则该指令只转换第一组数字。

算术状态标志： 影响算术状态标志

故障条件：

类型：	代码：	原因：	恢复方法：
4	51	字符串标签的长度值(LEN)大于其数据项长度。	1. 检查是否有其它指令正在向字符串标签的长度项写入数值。 2. 在长度项输入字符串实际所包含的字符数量。
4	53	输出值超过目标数据类型的限制。	选择其一： • 或缩小ASCII值长度。 • 或在目的标签中用比较大的数据类型。

执行：

条件：	梯形图动作：	结构文本动作：
预扫描	梯级输出条件被设置为假。	无动作。
梯级输入条件为假	梯级输出条件被设置为假。	无影响
梯级输入条件为真	执行指令。 梯级输出条件被设置为真。	无影响
EnableIn被置位	无影响	EnableIn一直被置位。 执行指令。
指令执行	置位S:C位。 清零目的标签。 指令转换源值。 如果结果是0，则置位S:Z位。	无动作。
后期扫描	梯级输出条件被设置为假。	无动作。

举例：读完称重秤的重量后（weight_read.EM被置位)，STOR 指令将存储在 weight_ascii 中的数字字符转换成REAL型数值。

用户可能会发现源值和目的标签值的小数部分有微小误差。

梯形图

结构文本

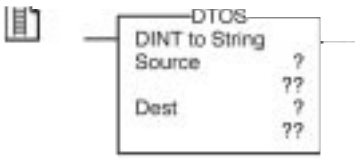
```
IF weight_read.EM THEN
    STOR(weight_ascii,weight);
    weight_read.EM := 0;
END_IF;
```

DINT 转换成字符串 (DTOS)

DTOS 指令用ASCII码表示一个数值。

操作数：

梯形图



操作数：	数据类型：	格式：	输入：	注释：
Source 源	SINT INT DINT REAL	标签	包含数值的标签	如果源值是REAL数据类型，则指令先将其转换成DINT数据类型。参见A-6页REAL转换成整数。
Destination 目的	STRING	标签	存储转换后的ASCII值的标签。	字符串数据类型是指： • 缺省的字符串数据类型 • 任何用户自己创建的新字符串数据类型

结构文本

```
DTOS(Source,Dest);
```

该指令操作数与梯形图DTOS指令的操作数相同。

说明: DTOS 指令把源值转换成 ASCII字符，并将结果存储于目的标签。

算术状态标志: 不影响

故障条件：

类型：	代码：	原因：	恢复方法：
4	51	字符串标签的长度值(LEN)大于其数据项长度。	1．检查是否有其它指令正在向字符串标签的长度项写入数值。 2．在长度项输入字符串实际所包含的字符数量。
4	52	输出的字符串比目的标签容量大。	为输出字符串创建一个足够大的新字符串数据类型。用新的字符串数据类型做为目的标签的数据类型。

执行：

条件：	梯形图动作：	结构文本动作：
预扫描	梯级输出条件被设置为假。	无动作。
梯级输入条件为假	梯级输出条件被设置为假。	无影响
梯级输入条件为真	执行指令。 梯级输出条件被设置为真。	无影响
EnableIn被置位	无影响	EnableIn一直被置位。 执行指令。
指令执行	指令转换源值。	
后期扫描	梯级输出条件被设置为假。	无动作。

举例：如果 temp_high 被置位，则DTOS 指令将 msg_num中的值转换成ASCII字符串，并将结果存放到msg_num_ascii中。后续梯形图程序会使msg_num_ascii标签中的字符插入或连接其它字符串，以产生一个可以显示的完整信息。

梯形图

结构文本

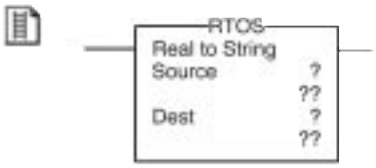
```
IF temp_high THEN
        DTOS(msg_num,msg_num_ascii);
        temp_high := 0;
END_IF;
```

REAL 转换成字符串 (RTOS)

RTOS 指令用ASCII字符串表示一个实型数值。

操作数：

梯形图



操作数：	数据类型：	格式：	输入：	注释：
Source 源	REAL	标签	包含REAL 型数值的标签	
Destination 目的	STRING	标签	存储ASCII 数值的标签	字符串数据类型是指： • 缺省的字符串数据类型 • 任何用户自己创建的新字符串数据类型

结构文本

```
RTOS(Source,Dest);
```

该指令操作数与梯形图RTOS 指令的操作数相同。

说明: RTOS 指令把源值转换成ASCII字符，并将结果存储于目的标签。

算术状态标志: 不影响

故障条件：

类型：	代码：	原因：	恢复方法：
4	51	字符串标签的长度值(LEN)大于其数据项长度。	1． 检查是否有其它指令正在向字符串标签的长度项写入数值。 2． 在长度项输入字符串实际所包含的字符数量。
4	52	输出的字符串比目的标签的容量大。	为输出字符串创建一个足够大的新字符串数据类型。用新字符串数据类型做为目的标签的数据类型。

执行：

条件：	梯形图动作：	结构文本动作：
预扫描	梯级输出条件被设置为假。	无动作。
梯级输入条件为假	梯级输出条件被设置为假。	无影响
梯级输入条件为真	执行指令。 梯级输出条件被设置为真。	无影响
EnableIn被置位	无影响	EnableIn一直被置位。 执行指令。
指令执行	指令转换源值。	
后期扫描	梯级输出条件被设置为假。	无动作。

举例： 如果send_data被置位，则RTOS 指令将data_1 中的数值转换成ASCII字符并将转换结果存储在data_1_ascii中。后续梯形图程序会将data_1_ascii标签中的字符插入或连接其它字符串，以产生一个可以显示的完整信息。

用户可能会发现源值和目的标签值的小数部分会有微小误差。

梯形图

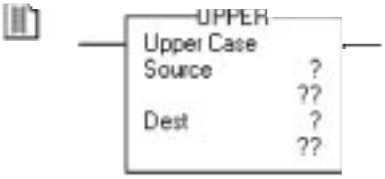
结构文本

```
IF send_data THEN
    RTOS(data_1,data_1_ascii);
    send_data := 0;
END_IF;
```

转换成大写(UPPER)

UPPER 指令将字符串中小写字母都转换成大写字母。

操作数：



梯形图

操作数：	数据类型：	格式：	说明：
Source 源	STRING	标签	包含要转换成大写字母的字符
Destination 目的	STRING	标签	存储转换后的大写字母

结构文本

```
UPPER(Source,Dest);
```

该指令操作数与梯形图UPPER指令的操作数相同。

说明: UPPER 指令把源标签中的全部字母都转换成大写形式，并将结果存储于目的标签。

- ASCII字符区分大小写，大写形式的 ″A″ ($41) 和小写形式的 ″a″ ($61)不一样。.
- 如果操作员直接输入ASCII字符，在比较这些字符前首先要将全部字母转换成大写或小写形式。，

在源标签中的所有非字母字符都保持不变。

算术状态标志： 不影响。

故障条件： 无

执行：

条件：	梯形图动作：	结构文本动作：
预扫描	梯级输出条件被设置为假。	无动作。
梯级输入条件为假	梯级输出条件被设置为假。	无影响
梯级输入条件为真	执行指令。 梯级输出条件被设置为真。	无影响
EnableIn被置位	无影响	EnableIn一直被置位。 执行指令。
指令执行	指令将源值转换成大写字母。	
后期扫描	梯级输出条件被设置为假。	无动作。

举例：要找到一指定项目信息，操作员在ASCII操作终端输入该项目的目录号。控制器从操作员终端读取输入信息后(terminal_read.EM 被置位)，UPPER 指令将catalog_number 中的全部字符转换成大写形式，并将结果存储于catalog_number_upper_case标签中。后续梯形图程序就会搜索与catalog_number_upper_case 中字符匹配的字符数组。

梯形图

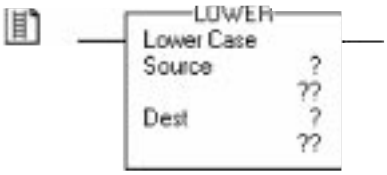
结构文本

```
IF terminal_read.EM THEN
        UPPER(catalog_number,catalog_number_upper_case);
        terminal_read.EM := 0;
END_IF;
```

转换成小写 (LOWER)

LOWER指令将字符串中字母都转换成小写形式。

操作数：



梯形图

操作数：	数据类型：	格式：	说明：
Source 源	STRING	标签	包含要转换成小写字母的字符
Destination 目的	STRING	标签	存储转换后的小写字母字符

结构文本

```
LOWER(Source,Dest);
```

该指令操作数与梯形图LOWER指令的操作数相同。

说明：LOWER指令把源标签中所有字母都转换成小写的形式，并将结果存储于目的标签中。

- ASCII 码字符区分大小写，大写形式的 ″A″ ($41) 和小写形式的 ″a″ ($61)不一样。
- 如果操作员直接输入ASCII字符，在比较这些字符前首先要将所有字母都转换成大写或小写形式。

在源标签中的所有非字母字符都保持不变。

算术状态标志：不影响

故障条件：无

条件：	梯形图动作：	结构文本动作：
预扫描	梯级输出条件被设置为假。	无动作。
梯级输入条件为假	梯级输出条件被设置为假。	无影响
梯级输入条件为真	执行指令。 梯级输出条件被设置为真。	无影响
EnableIn被置位	无影响	EnableIn一直被置位。 执行指令。
指令执行	指令将源值转换称小写形式。	
后期扫描	梯级输出条件被设置为假。	无动作。

举例：要找到一指定项目的信息，操作员将该项目号输入到ASCII码操作终端。控制器从终端读取输入信息后(terminal_read.EM被置位)，LOWER 指令将item _number 中的全部字符转换成小写形式，并将结果存储到item _number_ lower _case 标签中。后续的梯形图程序搜索与item _number_ lower _case 标签中字符匹配的字符数组。

梯形图

结构文本

```
IF terminal_read.EM THEN
        LOWER(item_number,item_number_lower_case);
        terminal_read.EM := 0;
END_IF;
```

注释：

通用属性

简介

本附录描述了Logix指令的通用属性。

相关信息：	参见页码码：
立即数	A - 1
数据转换	A- 1

立即数

无论用户何时输入十进制格式的立即数(常数)(如 -2，3)，控制器都使用32位存储该数值。如果用户输入的值为基数形式而不是十进制，例如二进制或十六进制，并且没有指定所有的32位，则控制器将零放入没有指定的位(零-填充方式)。

举　例　立即数的零-填充。

若输入：	控制器存储：
-1	16#ffff ffff (-1)
16#ffff (-1)	16#0000 ffff (65535)
8#1234 (668)	16#0000 029c (668)
2#1010 (10)	16#0000 000a (10)

数据转换

当用户在程序内混合使用数据类型时，将进行数据转换。

在…内编程时：	下述情况发生数据转换：
梯形图	在一条指令内参数为混合数据类型
功能块	连接两个具有不同数据类型的参数

如果指令的所有操作数使用下述类型，则指令的执行时间更短、需求的内存更小：

- 同一数据类型
- 最优数据类型：

  —本手册每条指令的“操作数”部分，黑体字数据类型代表最优数据类型。
  —DINT和REAL数据类型是典型的最优数据类型。
  —大部分功能块指令的操作数只支持一种数据类型(最优数据类型)。

如果用户混用数据类型并且使用非最优数据类型的标签，控制器按照下述原则转换数据：

- 其中一个操作数是REAL值吗？

如果：	那么输入操作数(例如，源，表达式中标签，极限值)转换为：
Yes	REALs
No	DINTs

- 指令执行后，如有必要，结果(DINT或REAL值)将转换为目标数据类型。

用户不能在一条使用整数或REAL数据类型的指令内指定BOOL型标签。

由于数据转换占用额外的时间和内存，用户可通过下述方法提高程序的效率：

- 整条指令使用同样的数据类型
- 尽量不使用SINT或者INT数据类型

也就是说，在指令内，包括立即数，使用全部DINT标签或者REAL标签。

下面解释了用户使用SINT或者INT标签时或者混合数据类型时，数据如何转换。

SINT或INT转换为DINT

对于将SINT或INT值转换到DINT值的指令，本手册的“操作数”部分标出了转换方法。

转换方法：	数据转换过程：
符号-扩充	将最左边位(数值符号)的值放入现有位以左的每一位内，直到满32位。
零-填充	将零放入现有位以左的每一位内，直到满32位。

下例给出了使用符号-扩充和零-填充方法转换数值的结果。

数值	2#1111_1111_1111_1111	(-1)
通过符号-扩充转换数值	2#1111_1111_1111_1111_1111_1111_1111_1111	(-1)
通过零-填充转换数值	2#0000_0000_0000_0000_1111_1111_1111_1111	-65535

由于立即数一直按零-填充存储，所以SINT或INT值的转换可能会产生非预期结果。在下例中，比较结果为假，这是因为源A是一个INT值，通过符号-扩充转换；而源B是一个立即数，按零-填充存储。

如果用户在一条通过符号-扩充转换数据的指令内使用SINT或INT标签和立即数，那么使用下述方法对处理立即数：

- 以十进制基数指定每一个立即数。
- 如果用户输入基数而非十进制的值，则指定立即数的全部32位。也就是将最左边位的值输入它以左的每位内，直到具有32位。
- 为每个操作数创建标签并且整条指令使用相同的数据类型。为了赋值一个常数，可采取下列方法之一：

  —将其输入到多个标签之一
  —增加一条MOV指令，将该值移到标签里。

- 使用MEQ指令仅检查所要求的位

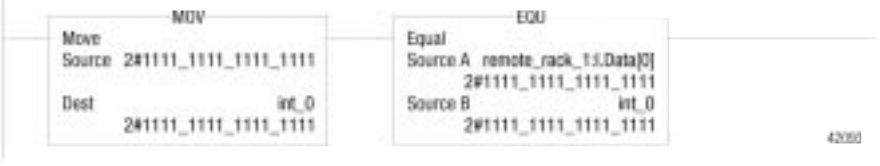
下例给出将立即数与INT标签混合使用的两种方法。两个例子均检查1771 I/O模块的各位，从而确定所有位是否接通。由于1771 I/O模块的输入数据字是INT标签，易于使用16-位的常数。

举　例

混合使用INT标签与立即数
由于remote_rack_1:I.Data[0]是一个INT标签，故检测它的值也以一个INT标签输入。

举　例

混合使用INT标签与立即数
由于remote_rack_1:I.Data[0]是一个INT标签，故检测它的值首先移到INT标签的int_0内，然后EQU指令比较这两个标签。



整数转换为REAL

控制器以IEEE单精度、浮点数格式存储REAL数值。其中一位用作数值符号，23位用于基值，8位用于指数(总共32位)。如果在同一条指令内将混合的整数标签(SINT， INT或 DINT)和REAL标签作为输入，那么在指令执行前控制器将整数转换为REAL值。

- SINT或者INT值总是转换为相同的REAL值。
- DINT值可能不会转换为相同的REAL值：

  ——一个REAL值的基值占用24位(23个存储位加一个“隐藏”位)。
  ——一个DINT值占用32位(1位用于符号，31位用于数值)。
  —如果DINT值要求24位以上的有效位，那么它不可能转换为相同的REAL值。控制器将其四舍五入为最接近的使用24个有效位的REAL值。

DINT转换为SINT或INT

为了将DINT值转换到SINT或INT值，如有必要，控制器将截去DINT的上面部分并置位溢出状态标志位。下例给出了DINT转换到SINT或INT的结果。

举 例 DINT转换为SINT以及INT

DINT值：	转换为更小的值：	
16#0001_0081 (65,665)	INT:	16#0081 (129)
	SINT:	16#81 (-127)

REAL转换为整数

为了转换REAL值为整数值，控制器将小数部分四舍五入并对非小数部分向上取整。如果数据丢失，控制器置位溢出状态标志位。
按下列原则取整数据：

- 除了x.5以外，其它数都四舍五入为最接近的整数。
- x.5取整为最接近的偶数。

下例给出了转换REAL值为DINT值的结果。

举　例　转换REAL值为DINT值

REAL值：	转换为DINT值：
-2.5	-2
-1.6	-2
-1.5	-2
-1.4	-1
1.4	1
1.5	2
1.6	2
2.5	2

重要事项　算术状态标志位由所存储的值决定。一般不影响算术状态关键字的指令可能也会对其产生影响，这是因为指令参数的混合数据类型如果发生类型转换，则类型转换过程将设置算术状态关键字。

功能块属性

简介

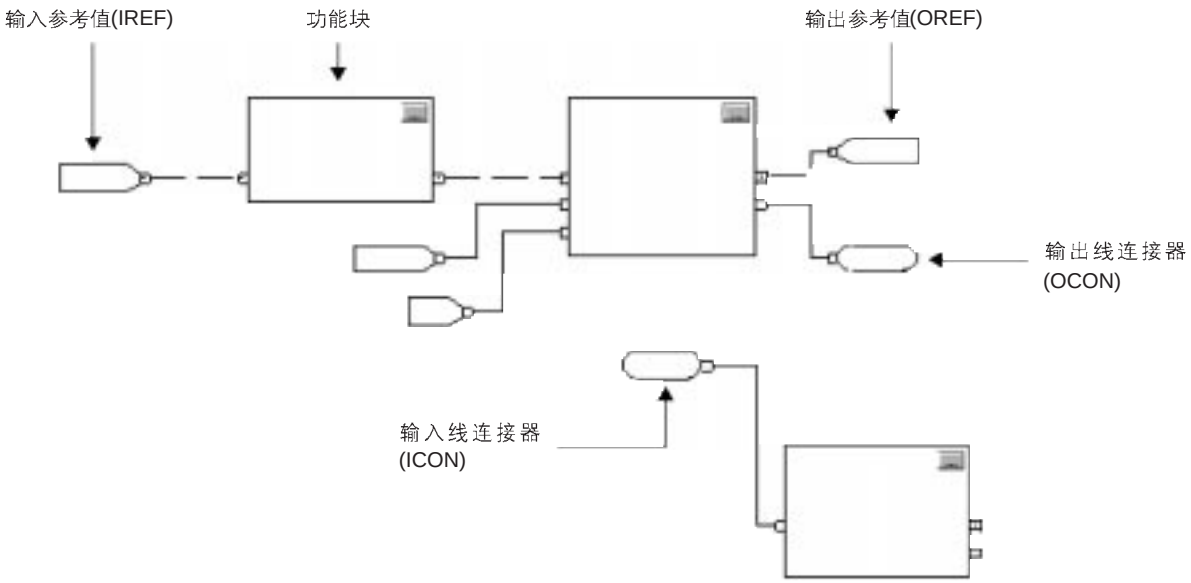
本附录描述了功能块指令的特点。阅读这部分内容有助于用户理解功能块例程如何执行。

重要事项

在功能块编程时，限定工程单位在+/-10 +/-15范围内，这是因为内部浮点运算采用的是单精度浮点数。若工程单位超出该范围，则当计算结果接近单精度浮点数的极限值(+/-10 +/-38)，可能导致精度降低。

选择功能块单元

使用以下单元实现对设备的控制：

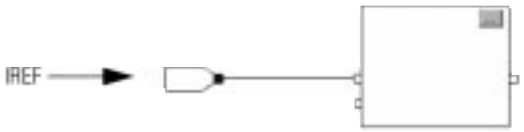


使用下表选择用户的功能块单元：

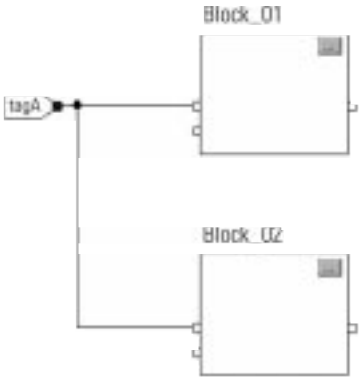
如果用户考虑：	则选用：
由输入设备或标签提供数据	输入参考(IREF)
向输出设备或标签发送数据	输出参考(OREF)
对一个或多个输入执行操作并产生一个或多个输出数据	功能块
在两个功能块间传送数据，当它们： • 位于同一表内但距离较远 • 位于同一例程不同表内	输出线连接器(OCON)和输入线连接器(ICON)
分散数据到例程中多个点	单个输出线连接器(OCON)和多个输入线连接器(ICON)

数据锁存

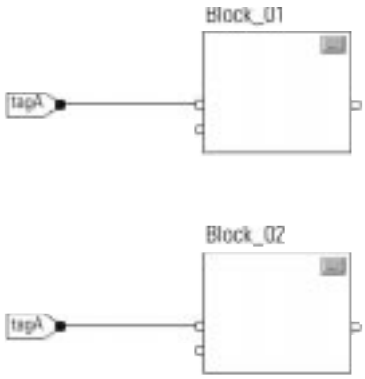
如果用户使用IREF指定功能块指令的输入数据，则在功能块例程扫描期间，IREF内的数据被锁存。IREF锁存来自程序作用域和控制器作用域的标签数据。控制器在每次扫描开始时更新所有的IREF数据。



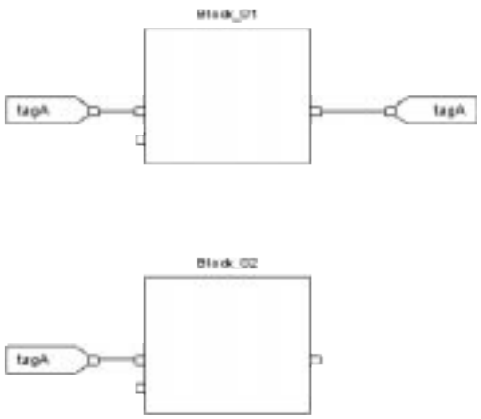
本例中，TagA的数值在开始执行例程时被存储。当Block_01执行时使用该存储值。当Block_02执行时使用同一存储值。如果在例程执行期间TagA存储值改变，在IREF内的TagA的存储值直到下一次执行例程时才改变。



本例与上例相同。tagA的值只有在例程开始执行时才被存储。在整个例程内都使用该存储值。



RSLogix5000版本11开始，用户可以在一个例程中多个IREFs和一个OREF中使用同一标签。IREFs中标签的数值在程序扫描过程中被锁定，因此在例程执行过程中，即使OREF中有多个标签值，所有的IREFs仍为相同值。该例中，如果例程开始执行扫描时，tagA值为25.4，Block_01将tagA 的数值变为50.9，但Block_02执行此次扫描时，连接到Block_02的第二个IREF仍然使用25.4。直至下次扫描开始，tagA的新值50.9才能被IREFs使用。



执行顺序

当用户进行如下操作时，RSLogix 5000编程软件自动确定例程内功能块的执行顺序：

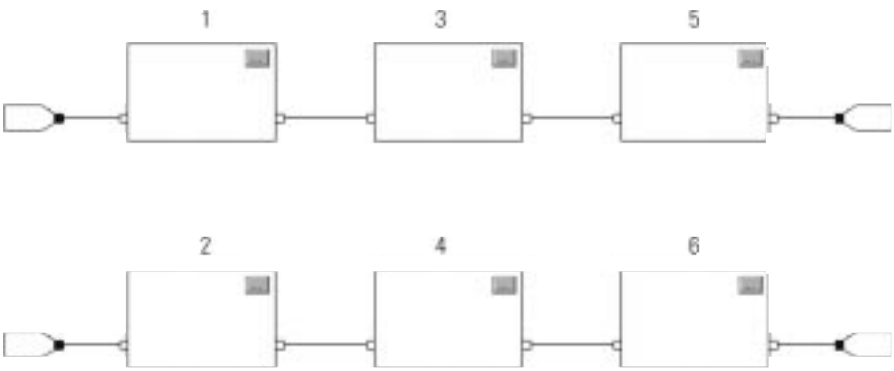
- 校验功能块例程
- 校验含有功能块例程的项目
- 下载含有功能块例程的项目

如有必要，用户可通过连接功能块并指明反馈线数据流来指定执行顺序。

如果功能块没有连接在一起，哪个功能块首先执行没有影响。此时，功能块间无数据流。

如果用户顺序连接功能块，则执行顺序为从输入到输出。功能块的输入必须在控制器执行该功能块前可用。例如，功能块2必须在功能块3前执行，因为功能块2的输出是功能块3的输入。

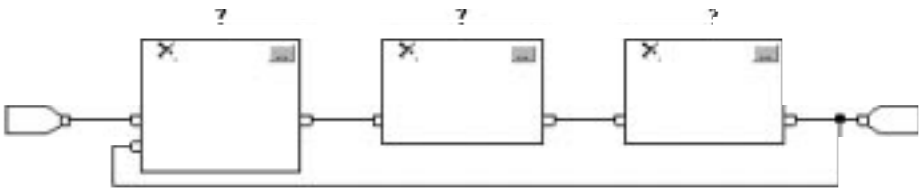
执行顺序只与连接的功能块相关。下例中的两个功能块组没有连接在一起。在特定组内的功能块执行顺序只与该组内的功能块有关。



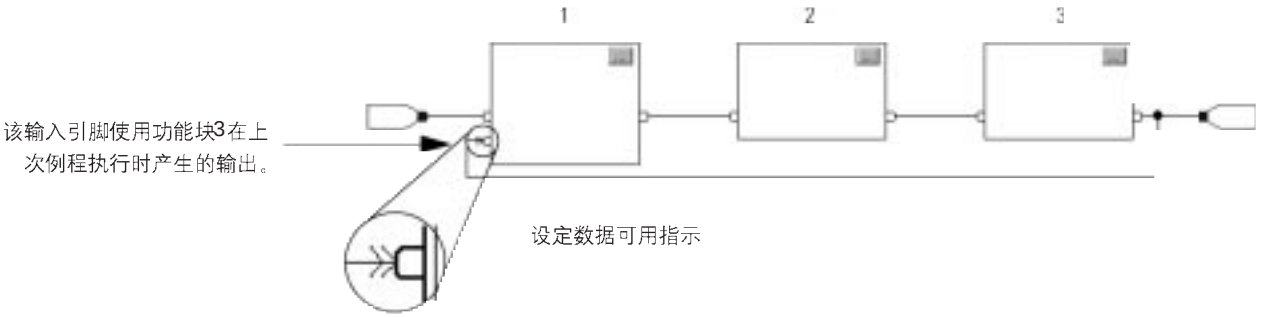
解析回路

创建功能块反馈回路需将块输出引脚连接到同一功能块的输入引脚上。下图中已经实现。该回路仅包含单个功能块，因此不受执行顺序的影响。

如果一组功能块在同一闭环内，则控制器无法确定哪个功能块最先执行。也就是说，它无法解析回路。

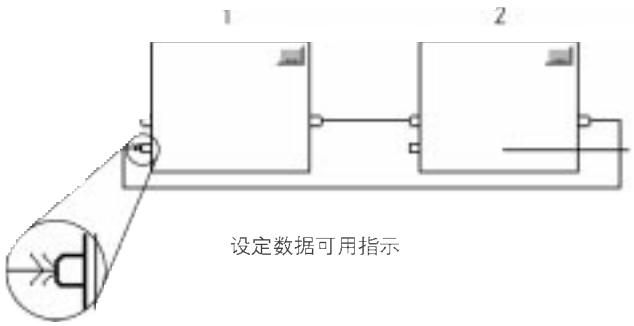
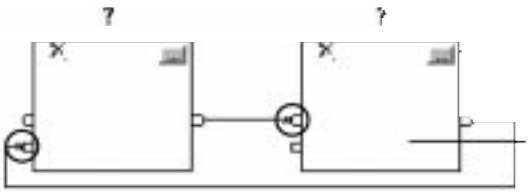


为确定功能块的执行顺序，可使用设定数据可用指示来标记用于创建回路(反馈线)的输入线。本例中，功能块1使用功能块3在上次例程执行时产生的输出。



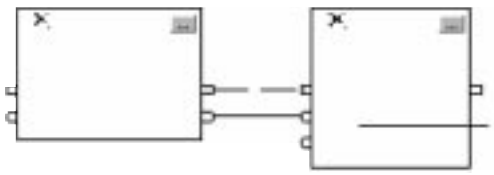
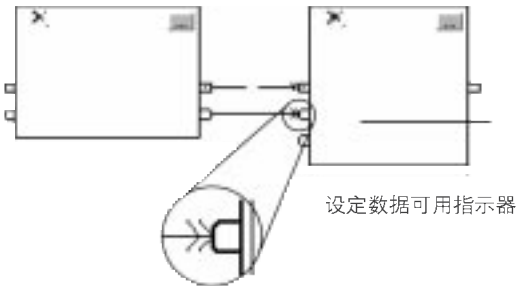
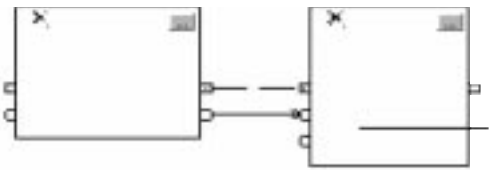
设定数据可用指示确定了回路内数据流向。箭头指示了数据输入到回路的首个功能块。

切忌使用设定数据可用指示来标记回路中所有连接线。

正确接法	错误接法
 设定数据可用指示 设定数据可用指示确定了回路内的数据流。	 由于所有连接线均使用设定数据可用指示器，控制器无法解析回路。

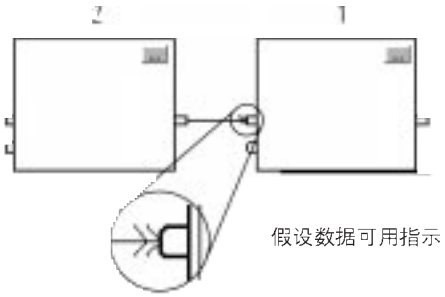
解析两功能块间数据流

如果用户使用两条或更多连接线连接两个功能块，则所有连接线需使用相同的数据流指示。

正确接法	错误接法
 所有连接线均不使用设定数据可用指示。  设定数据可用指示器 两条连接线均使用设定数据可用指示器。	 一条连接线使用设定数据可用指示，而另一条不使用。

创建一个扫描周期的延迟

使用设定数据可用标记在两个功能块间产生一个扫描周期延迟。在下例中，功能块1首先执行。它使用例程上次扫描时功能块2的输出数据。



总结

总之，功能块例程按以下顺序执行：

1. 控制器锁存IREFs内的所有数据。

2. 控制器按顺序(由功能块接线方式决定)执行其它功能块。

3. 控制器将输出写入OREFs。

功能块对溢出条件的响应

通常，发生溢出时功能块指令会保持历史记录而不使用±NAN或 ±INF值来更新记录。每条指令都有相对于溢出条件的响应：

响应1：		响应2：	响应3：	
功能块执行其算法并检查结果是否为±NAN或±INF。如果是±NAN或±INF，则功能块输出±NAN或±INF。		带输出极限的功能块执行算法并检查结果是否为±NAN或±INF。输出极限通过HighLimit(上限)和LowLimit(下限)输入参数来定义。如果是±INF，则功能块输出一极限结果。如果是±NAN，则不使用输出极限，功能块输出±NAN。	不接受溢出条件。这些指令通常具有布尔型输出。	
ALM	NTCH	HLL	BAND	OSRI
DEDT	PMUL	INTG	BNOT	RESD
DERV	POSP	PI	BOR	RTOR
ESEL	RLIM	PIDE	BXOR	SETD
FGEN	RMPS	SCL	CUTD	TOFR
HPF	SCRV	SOC	D2SD	TONR
LDL2	SEL		D3SD	
LDLG	SNEG		DFF	
LPF	SRTP		JKFF	
MAVE	SSUM		OSFI	
MAXC	TOT			
MINC	UPDN			
MSTD				
MUX				

定时方式

以下过程控制和驱动指令支持不同的定时方式。

DEDT	LDLG	RLIM
DERV	LPF	SCRV
HPF	NTCH	SOC
INTG	PI	TOT
LDL2	PIDE	

有三种不同的定时方式：

定时方式：	说明：	
Periodic周期	周期方式是缺省方式，适用于大多数的控制应用系统。如果用户例程在周期性任务中执行，我们推荐其中指令采用此方式。指令使用的时间增量(DeltaT)由下列条件确定：	
	如果指令执行在：	**那么DeltaT等于：**
	周期任务	任务的执行周期 设置DeltaT为任务执行周期的整数部分，单位为毫秒。控制器截掉任务执行周期的小数部分。例如，任务执行周期为10.5ms，控制器设置DeltaT等于10ms。 如果用户需要使用任务执行周期的小数，选用过采样方式。用户可以在过采样方式下设置OversampleDT参数等于任务执行周期，包含任一小数。
	事件或连续任务	自上次任务执行后的运行时间 控制器截取运行时间的整数部分(毫秒)。例如，如果运行时间等于10.5ms，控制器设置DeltaT等于10ms。
	过程输入的刷新需要与任务执行同步，或采样速度比任务执行快5-10倍，以达到输入与指令之间的采样误差最小。	
Oversample过采样	在过采样方式下，指令使用的时间增量(DeltaT)为写入该指令OversampleDT参数内的值。在事件或连续任务内执行该指令，且过程量输入不具有与其更新相关联的时间戳时使用这种方式。如果过程量输入具有时间戳值，则使用实时采样方式。 在用户程序中增加逻辑来控制何时执行该指令。例如，用户使用定时器设置OversampleDeltaT值，通过EnableIn输入来控制指令执行。 为了使输入与指令之间的采样误差最小，过程输入需要的采样时间比指令的执行时间快5-10倍。	

定时方式：	说明：
Real time sampling 实时采样	在实时采样方式下，指令使用的时间增量(DeltaT)是两个对应的过程输入的更新时间戳的差值。在事件或连续任务内执行该指令，且工程输入具有与其更新相关联的时间戳值时使用这种方式。 时间戳值读自指令的RTSTimeStamp参数中输入的标签名。通常该标签名是与过程输入相关联的输入模块上的一个参数。 指令将组态的RTSTime值(预定刷新周期)与计算的DeltaT进行比较，从而确定指令是否读取过程输入的每次刷新值。如果DeltaT与组态时间相差超过1毫秒，则指令置位RTSMissed状态位来表示读取模块输入更新存在问题。

基于时间的指令要求DeltaT为常数，这是为了使控制算法能够正确计算过程输出。如果DeltaT变化，则过程输出的出现不连续。不连续的严重程度取决于指令和DeltaT的变化范围。在下述情况下发生输出不连续：

- 该指令在扫描期间不执行。
- 该指令在任务期间多次执行。
- 任务运行时任务扫描速率或者过程输入的采样时间改变。
- 运行任务时用户改变时间基方式。
- 任务运行时过滤器功能块上的阶数参数改变。阶数参数的改变将在指令内选择不同的控制算法。

用于定时方式的通用指令参数

支持时间基方式的指令具有下述输入与输出参数：

输入参数

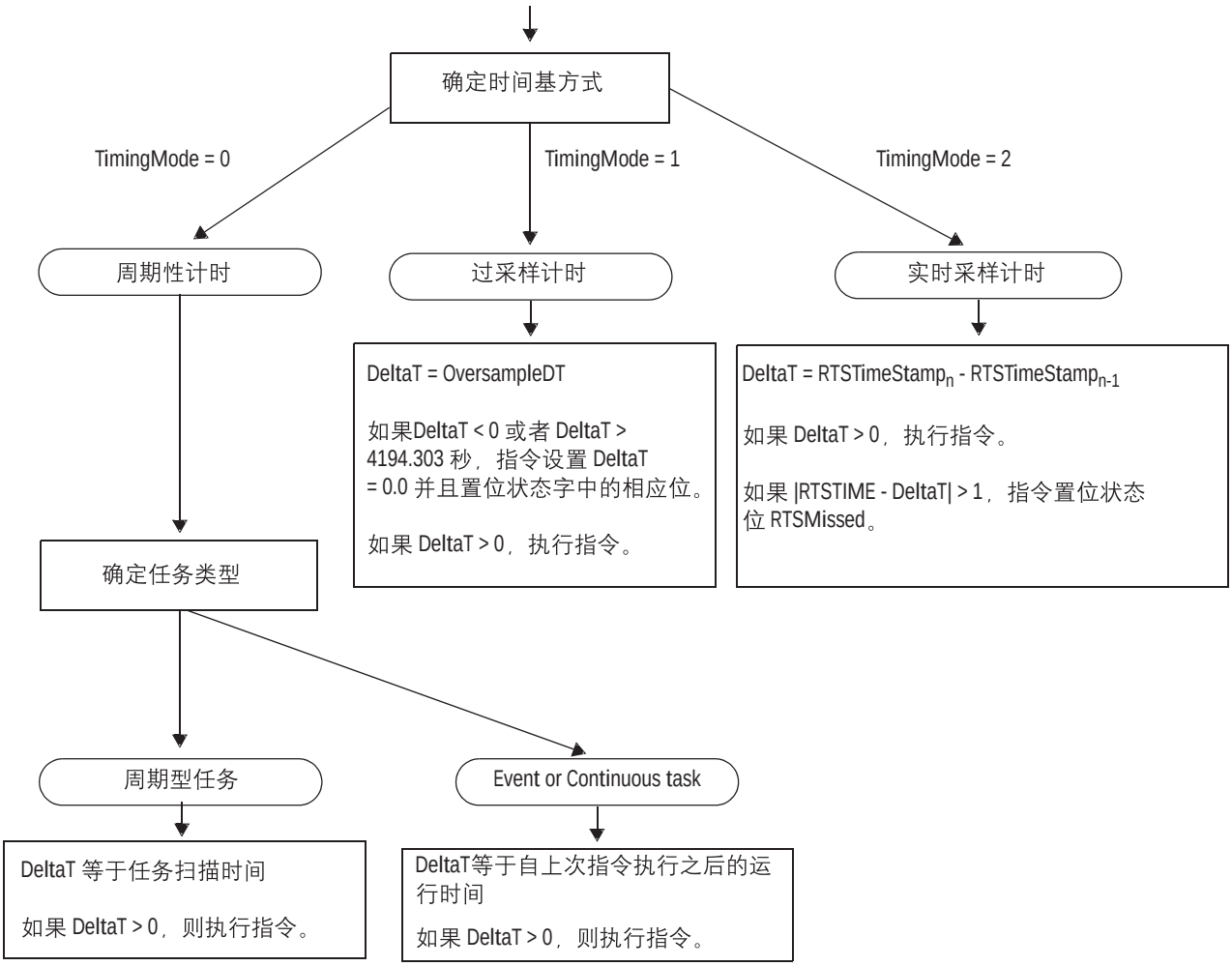
输入参数：	数据类型：	说明：
TimingMode	DINT	选择定时执行方式。 数值： 说明： 0 周期方式 1 过采样方式 2 实时采样方式 有效值 = 0到2 缺省值 = 0 当TimingMode = 0且为周期性任务时，周期定时使能，DeltaT设置为任务扫描速率。当TimingMode = 0并且为事件或连续性任务时，周期定时使能，DeltaT等于自上次指令执行之后的运行时间。 当TimingMode = 1时，过采样定时使能，DeltaT设置为OversampleDT参数值。 当TimingMode = 2时，实时采样定时使能，DeltaT为当前与上一次时间戳值(读自与输入相关联的模块)之差。 如果TimingMode无效，指令设置状态字中相应位。
OversampleDT	REAL	用于过采样方式的执行时间。DeltaT以秒为单位。如果TimingMode = 1，则OversampleDT = 0.0将禁止控制算法的执行。如果无效，指令设置DeltaT = 0.0并且置位状态字中的相应位。 有效值 = 0 到4194.303秒 缺省值 = 0.0
RTSTime	DINT	实时采样定时方式的模块刷新周期。期望的DeltaT刷新周期以毫秒为单位。更新周期通常为用于配置模块更新时间的值。如果无效，指令置位状态字中的相应位并禁止RTSMissed的检查。 有效值 = 1 到32,767 ms 缺省值 = 1
RTSTimeStamp	DINT	实时采样定时的模块时间戳。时间戳对应于输入信号的上次更新。该值用于计算DeltaT。如果无效，指令置位状态字中的相应位并禁止控制算法的执行以及RTSMissed的检测。 有效值 = 1 到32,767 ms(覆盖从32767到0) 1个计数= 1毫秒 缺省值 = 0

输出参数

输出参数：	数据类型：	说明：
DeltaT	REAL	两次刷新之间的运行时间，以秒为单位。控制算法用该时间值来计算过程输出。 周期方式：DeltaT=任务扫描速率(如果是周期任务)，DeltaT=自上次指令执行之后的运行时间(如果是事件或连续任务) 过采样方式：DeltaT = OversampleDT 实时采样方式：DeltaT = (RTSTimeStamp n - RTSTimeStamp n-1)
Status	DINT	功能块的状态字。
TimingModeInv (Status.27)	BOOL	TimingMode值无效。
RTSMissed (Status.28)	BOOL	只用于实时采样方式。当ABS | DeltaT - RTSTime | > 1 (.001 second)时置位。
RTSTimeInv (Status.29)	BOOL	RTSTime值无效。
RTSTimeStampInv (Status.30)	BOOL	RTSTimeStamp值无效。
DeltaTInv (Status.31)	BOOL	DeltaT值无效。

定时方式概述

下面的图表给出一条指令确定合适的定时方式的方法。



程序/操作员控制

一些指令支持程序/操作员(Program/Operator)控制。这些指令有：

- 增强选择 (ESEL)
- 累加器 (TOT)
- 增强型PID (PIDE)
- 斜率/渗透 (RMPS)
- 离散型2-态设备 (D2SD)
- 离散型3-态设备(D3SD)

程序/操作员控制允许用户由程序和操作员界面设备同时控制这些指令。在程序控制方式下，指令由程序输入控制；在操作员控制方式下，指令由操作员输入控制。

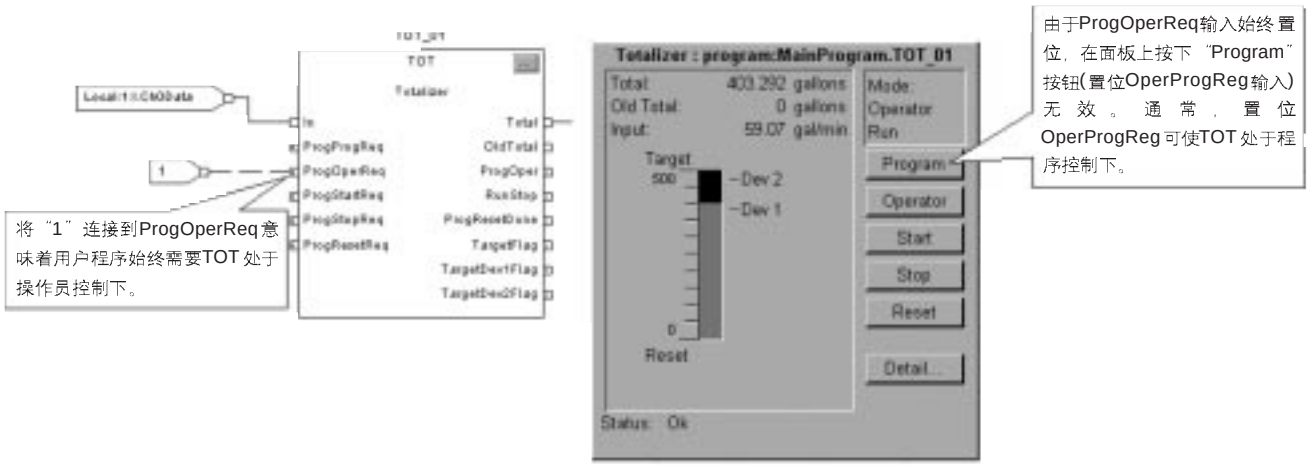
通过以下输入状态确定由程序还是操作员来控制：

输入：	说明：
.ProgProgReq	程序请求进入程序控制方式。
.ProgOperReq	程序请求进入操作员控制方式。
.OperProgReq	操作员请求进入程序控制方式。
.OperOperReq	操作员请求进入操作员控制方式。

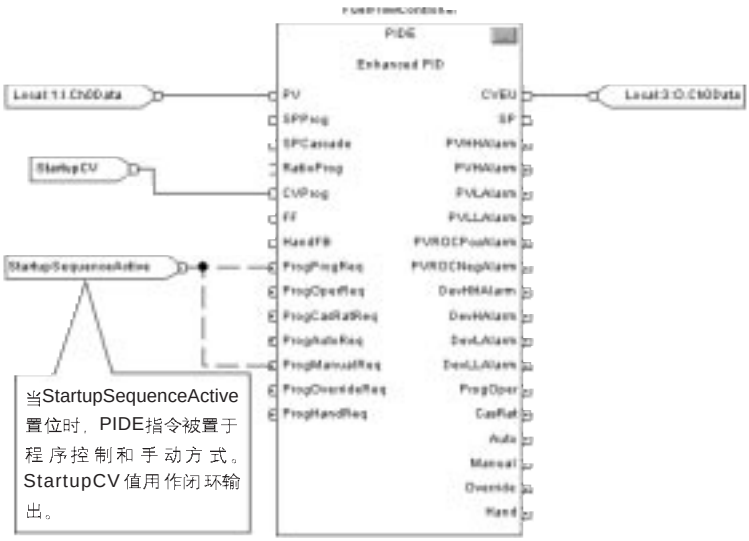
为了确定指令处于程序或操作员控制方式，可检查ProgOper输出。如果ProgOper置位，则该指令处于程序控制方式；如果ProgOper清零，则该指令处于操作员控制方式。

如果两个输入请求位都置位，则操作员控制优先于程序控制。例如，如果ProgProgReq 和ProgOperReq都置位，则指令进入操作员控制。

程序请求输入优先于操作员请求输入。因此可使用ProgProgReq和ProgOperReq输入“锁定”指令在期望的控制方式下。例如，假定一条累加器指令始终处于操作员控制下，并且用户程序始终不能控制该累加器的运行或停止；这种情况下，用户可以将数值1连接到ProgOperReq。这就防止了通过操作员界面设备置位OperProgReq将累加器置于程序控制方式。



同样，始终置位ProgProgReq能够“锁定”指令处于程序控制。这在自动设置起动顺序时很有用，由程序来控制指令的操作不用担心操作员疏忽对指令进行控制。例如，在起动期间由程序置位ProgProgReq输入，然后在起动完成时清零ProgProgReq输入。一旦ProgProgReq输入清零，指令保持为程序控制直到接到改变的请求。例如，操作员可以从面板置位OperOperReq输入，从而接管该指令的控制。下例给出如何锁定指令处于程序控制。



当指令执行时，该指令的操作员请求输入始终被清零。这样就通过设置期望方式对应的请求位来实现操作员界面对指令的操作。用户不必编程操作员界面复位请求位。例如，若操作员界面置位PIDE指令的OperAutoReq输入，当PIDE指令执行时，它决定适当的响应并清零OperAutoReq。

程序请求输入通常不由指令清零，因为它们通常连接到指令的输入。如果指令清零这些输入，那么输入刚好再次被接入的输入置位。用户可能会遇到这种情形：在希望程序请求被本指令清零同时，还希望使用其它逻辑置位程序请求。在这种情况下，用户可以置位ProgValueReset输入，且在指令执行时，指令始终清零程序方式请求输入。

本例中，另一梯形图例程中的一条梯级用于在按下按钮时单步锁存PIDE指令的ProgAutoReq。由于PIDE指令自动清零程序方式请求，用户不必编程梯形图逻辑在例程执行完后清零ProgAutoReq，并且每次按下按钮时PIDE指令只收到一个转到Auto的请求。

当按下TIC101AutoReq按钮时，单步锁存PIDE指令TIC101的ProgAutoReq。TIC101已被组态为ProgValueReset输入置位，所以PIDE指令执行时，指令自动清零ProgAutoReq。

注释：

结构文本编程

简介

本附录说明了结构文本编程的特点。阅读这部分内容有助于用户理解结构文本程序如何执行。

关于：	参见：
结构文本语法	C-1
赋值	C-2
表达式	C-4
指令	C-11
结构体	C-12
注释	C-28

结构文本语法

结构文本是一种使用语句来定义执行动作的文本化编程语言。结构文本不区分字母大小写。它包含以下要素：

术语：	定义：		实例：
赋值(参见页码C-2)	使用赋值语句指定标签的数值。 赋值运算符是:= 。 赋值语句以半角“;”结束。		tag:=expression;
表达式(参见页码C-4)	表达式是完整的赋值或结构语句的一部分。 表达式计算出数值(数值表达式)或真假状态(布尔表达式)。		
	一个表达式包括：		
	标签	用于存储数据的命名内存区域 (BOOL,SINT,INT,DINT,REAL,string)	value1 4
	立即数	常数	tag1+tag2
	运算符	在表达式中特指一定操作的符号或助记符	tag1>=value1
	函数	一个函数执行后产生一个数值，括号内是其操作数。 函数与指令的语法相似，区别在于函数仅能用在表达式中，而指令不能用在表达式中。	function(tag1)

术语：	定义：	实例：

指令
(参见页码C-11)
	每条指令是独立的语句。
	每条指令使用括号包括其操作数。
	根据指令不同，数可以为0、1或多个操作数。
	当指令执行时，产生数据结构的部分值。
	赋值语句以半角“;”结束。
	
	指令与功能的语法相似，区别在于指令不能用在表达式中。
	而功能仅用在表达式中。
		instruction();
		instruction(operand);
		instruction(operand1,
		operand2,operand3);

结构体 (参见页码C-12)	用于触发结构文本代码的条件语句(例如，other语句)。 赋值语句以半角“;”结束。	IF…THEN CASE FOR…DO WHILE…DO REPEAT…UNTIL EXIT
注释 (参见页码C-28)	用于解释或阐明一段结构文本代码功能的文本。 • 使用注释提高结构文本程序的可读性 • 注释不影响结构文本执行 • 注释可以写在结构文本程序的任意位置	//comment (*start of comment… end of comment*) /*start of comment…end of comment*/

赋值语句

使用赋值语句改变标签内存储的数值。赋值语句的语法是：

tag:=expression;

其中：

元素：	描述：	
tag	取得新值的标签名称 其数据类型必须为BOOL,SINT,INT,DINT或REAL	
:=	赋值符号	
expression	赋给标签的新值	
	如果tag的数据类型为：	使用以下类型表达式：
	BOOL	布尔表达式
	SINT INT DINT REAL	数值表达式
;	赋值语句结束符	

该标签将数值保持至其它赋值语句改变该值。

表达式可以很简单，如立即数或其它标签名；也可以很复杂，包括多个运算符和/或功能。详细信息请参见页码C-4 ″表达式″一节。

指定非保持赋值语句

非保持型赋值语句与上面所描述的赋值语句不同，每次控制器进入下的状态，非保持型赋值语句中的标签复位为0：

- 进入RUN模式
- 如果用户组态SFC为自动复位，离开SFC的步时(仅适用于用户在步动作间嵌入赋值语句或使用JSR指令调用结构文本例程的情况)

一个非保持型赋值语句的语法：

tag[:=]expression;

其中：

元素：	描述：	
tag	取得新值的标签名称 其数据类型必须为BOOL,SINT,INT,DINT或REAL	
[:=]	非保持型赋值符号	
expression	赋给标签的新值	
	如果tag的数据类型为：	使用以下类型表达式：
	BOOL	布尔表达式
	SINT INT DINT REAL	数值表达式
;	赋值语句结束符	

为字符串赋值ASCII

通过赋值运算符可将ASCII赋给字符串标签中DATA成员的元素。通过指定字符值或指定标签名称、DATA成员和字符元素来为字符串赋值。例如：

正确语法：	错误语法：
string1.DATA[0]:= 65;	string1.DATA[0] := A;
string1.DATA[0]:=string2.DATA[0];	string1 := string2;

ASCII字符串指令用于增加或插入ASCII字符串。

目的：	所用指令：
在字符串末尾添加字符	CONCAT
在字符串中插入字符	INSERT

表达式

一个表达式包括标签名、相等或比较关系。使用以下元素写出表达式：

- 存储数值的标签名(变量)
- 用户直接输入到表达式的数值(立即数)
- 功能，如：ABS,TRUNC
- 运算符，如：+,-,<,>,And,Or

用户写表达式时必须遵守的规则：

- 表达式中可混合使用大小写字母。例如，三种变形的 ″AND″：AND，And和and均可使用。
- 对于较复杂的要求，可使用括号将多个表达式集合到一个表达式内。这样可以提高整个表达式的可读性，并使其按照期望的顺序执行。参见页码C-10 ″确定执行顺序″ 一节。

在结构文本中，用户可使用以下两种类型的表达式：

布尔表达式：产生1(真)或0(假)布尔值的表达式。

- 布尔表达式中使用布尔标签，关系运算符和逻辑运算符来比较或检查条件是否为真或假。例如，tag1>65。
- 简单的布尔表达式可以是单独BOOL标签。
- 用户使用布尔表达式作为其它逻辑执行的条件。

数值表达式：计算出整数或浮点数值的表达式。

- 数值表达式使用算术运算符、算术功能和位运算符。例如，tag1+5。
- 通常，用户会将数值表达式嵌套到布尔表达式中。例如，(tag1+5)>65。

使用下表为用户表达式选择运算符：

如果用户希望：	那么：
计算算术值	“使用算术运算符和功能”，页B-7。
比较两数值或字符串	“使用关系运算符”，页B-8。
检查条件为真或假	“使用逻辑运算符”，页B-10。
比较数值内各位	“使用位运算符”，页B-11。

使用算术运算符和功能

用户可以在同一算术表达式内使用多个运算符和功能。

使用下列算术运算符计算数值：

目的：	运算符：	最优数据类型：
加	+	DINT，REAL
减 / 非	-	DINT，REAL
乘	*	DINT，REAL
指数(x 是y 的幂)	**	DINT，REAL
除	/	DINT，REAL
模数-除	MOD	DINT，REAL

算术功能执行数学操作。在功能中需指定常数、非布尔型标签或表达式。

目的：	运算符：	最优数据类型：
绝对值	ABS(numeric_expression)	DINT，REAL
反余弦	ACOS(numeric_expression)	REAL
反正弦	ASIN(numeric_expression)	REAL
反正切	ATAN(numeric_expression)	REAL
余弦	COS(numeric_expression)	REAL
弧度转换为角度	DEG(numeric_expression)	DINT，REAL
自然对数	LN(numeric_expression)	REAL
以10为底的对数	LOG(numeric_expression)	REAL
角度转换成弧度	RAD(numeric_expression)	DINT，REAL
正弦	SIN(numeric_expression)	REAL
平方根	SQR(numeric_expression)	DINT，REAL
正切	TAN(numeric_expression)	REAL
截断	TRN(numeric_expression)	DINT，REAL

例如：

使用该格式：	实例： 此情形下：	 用户需编程：
value1 operator value2	如果gain_4和gain_4_adj均为DINT型标签且要求："gain_4加上15并将结果存到gain_4_adj。"	gain_4_adj := gain_4+15;
operator value1	如果alarm和high_alarm均为DINT型标签且要求："将high_alarm取负并将结果存到alarm。"	alarm:= -high_alarm;
function(numeric_expression)	如果overtravel和overtravel _POS均为DINT标签且要求"计算overtravel的绝对值并将结果保存到overtravel_POS。"	overtravel_POS := ABS(overtravel);
value1 operator (function((value2+value3)/2)	如果adjustment和position均为DINT型标签且sensor1和sensor2均为REAL型标签且要求："计算sensor1和sensor2平均值的绝对值再加上adjustment，并将结果保存到position。"	position := adjustment + ABS((sensor1 + sensor2)/2);

使用关系运算符

关系运算符将两数值或字符串进行比较，产生一个真或假的结果。关系运算的结果为BOOL值：

比较关系为：	结果：
真	1
假	0

使用下例关系运算符：

比较关系：	使用运算符：	最优数据类型：
等于	=	DINT，REAL，string
小于	<	DINT，REAL，string
小于或等于	<=	DINT，REAL，string
大于	>	DINT，REAL，string
大于或等于	>=	DINT，REAL，string
不等于	<>	DINT，REAL，string

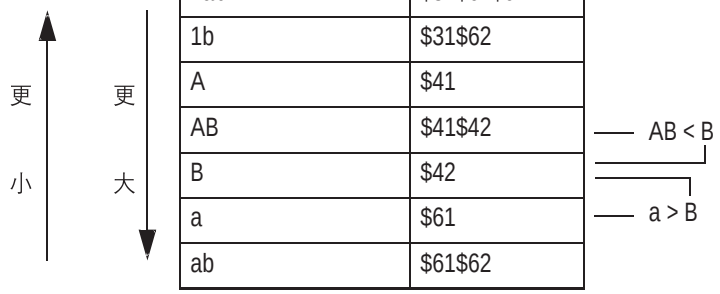
例如：

使用该格式：	实例： 此情形下：	 用户需编程：
value1 operator value2	如果temp是DINT标签且要求：“如果temp小于100º，则…。”	IF temp<100 THEN...
stringtag1 operator stringtag2	如果bar_code和dest均为字符串标签且要求：“如果bar_code等于dest，那么…”	IF bar_code=dest THEN...
char1 operator char2 将ASCII对应的十进制数值直接输入表达式。	如果bar_code是字符串标签且要求：“如果bar_code.DATA[0]等于‘A’则”	IF bar_code.DATA[0]=65 THEN...
bool_tag := bool_expressions	如果count和length均为DINT型标签，done是BOOL型标签，且要求：“如果count大于或等于length，计算完成。”	done := (count >= length);

如何计算字符串

ASCII字符对应的十六进制值决定了一个字符串小于或者大于另一个字符串。

- 两个字符串按照电话号码簿方式分类时，它们的大小由字符串的顺序决定。



ASCII字符	十六进制代码
1ab	$31$61$62
1b	$31$62
A	$41
AB	$41$42
B	$42
a	$61
ab	$61$62

- 如果字符匹配则字符串相等。
- ASCII字符区分大小写。大写“A”($41)不等于小写“a”($61)。

字符的十进制数值和十六进制代码，参见本手册的封底。

使用逻辑运算符

用户可使用逻辑运算符检查多个条件为真或假。逻辑操作的结果是BOOL值：

如果比较关系为：	结果是：
真	1
假	0

使用以下逻辑运算符：

目的：	运算符：	数据类型：
按位与	&,AND	BOOL
按位非	NOT	BOOL
按位或	OR	BOOL
按位异或	XOR	BOOL

例如：

使用该格式：	实例：	
	此情形下：	用户需编程：
BOOLtag	如果photoeye是BOOL型标签且要求："如果photoeye_1为真，那么…"	IF photoeye THEN...
NOT BOOLtag	如果photoeye是BOOL型标签且要求："如果photoeye_1为假，那么…"	IF NOT photoeye THEN...
expression1 & expression2	如果photoeye是BOOL型标签,temp是DINT标签且要求："如果photoeye是真且temp小于100º，那么…"	IF photoeye & (temp<100) THEN...
expression1 OR expression2	如果photoeye是BOOL型标签, temp是DINT标签且要求："如果photoeye是真或temp小于100º那么…"	IF photoeye OR (temp<100) THEN...
expression1 XOR expression2	如果photoeye1和photoeye2是BOOL型标签且要求："如果： • photoeye1为真同时photoeye2为假 或 • photoeye1为假同时photoeye2为真" 那么…	IF photoeye1 XOR photoeye2 THEN...
BOOLtag := expression1 & expression2	如果photoeye1和photoeye2是BOOL型标签，open是BOOL型标签且要求："如果photoeye1和photoeye2同为真，置位open为真。"	open := photoeye1 & photoeye2;

使用位运算符

位运算符基于两个数值处理一个数值中各位。

目的：	所用运算符：	最优数据类型：
位与	&,AND	DINT
位或	OR	DINT
位异或	XOR	DINT
位补码	NOT	DINT

例如：

所用格式：	实例： 此情形下：	 用户需编程：
value1 operator value2	如果input1,input2 和result1 均为DINT 型标签且要求：“计算input1 和input2 的位结果，并将其保存到result1 。”	result1 := input1 AND input2;

确定运算顺序

指令按预先规定的顺序，而非一定是从左到右来执行写入表达式的操作。

- 同级运算的顺序是从左向右执行。
- 如果表达式中包含多个运算符或功能，可通过圆括号 “()” 将条件分组。这样可以确保执行顺序正确并使得表达式更具可读性。

顺序：	运算符：
1	( )
2	function(…)
3	**
4	- (取负)
5.	NOT
6.	*，/，MOD
7.	+，- (减)
8	<，<=，>，>=，
9	=,<>
10	&,AND
11	XOR
12	OR

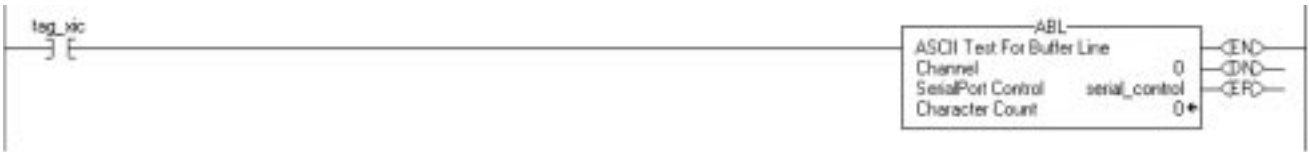
指令

结构文本编程语句也可以是指令。请参见本手册开始的指令列表获取结构文本语言可用指令。每扫描一次结构文本程序，指令执行一遍。如果结构体条件为真，则结构体中结构文本指令执行。如果结构体条件为假，则结构体中的语句不被扫描。此时没有梯级条件或状态转换触发指令执行。

与功能块的区别在于，功能块指令通过输入使能(EnableIn)触发其执行，而结构文本指令执行时相当于输入使能(EnableIn)一直置位。

与梯形图的区别在于，梯形图指令通过输入梯级条件触发执行，一些梯形图指令仅当输入梯级条件触发上升沿时执行。这些是边沿触发型梯形图指令。在结构文本程序中，除非用户预先规定结构文本指令的执行顺序，否则指令将会在其被扫描时执行。

例如，ABL指令是边沿触发梯形图指令。该例中，当tag_xic由清零到置位跃变一次，ABL指令执行一次。当tag_xic一直置位或清零时，ABL指令不执行。



在结构文本语言中，如果用户将该例写为：

```
IF tag_xic THEN ABL(0,serial_control);
END_IF;
```

那么，当tag_xic置位而不仅是从清零到置位跃变时， ABL指令每次扫描时均执行。

如果用户希望仅当tag_xic为上升沿时ABL指令执行，必须编写结构文本条件指令。使用上升沿指令触发执行。

```
osri_1.InputBit := tag_xic;
OSRI(osri_1);

IF (osri_1.OutputBit) THENABL(0,serial_control);
END_IF;
```

结构体

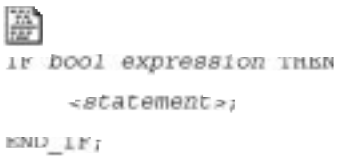
结构体可以单独或与其它结构体嵌套编程。

如果用户希望：	使用此结构体：	可用编程语言：	参见页码码：
当特定条件发生时，执行某项操作	IF…THEN	结构文本	C-13
基于数值选择执行的操作	CASE…OF	结构文本	C-16
根据指定的次数重复执行某项操作	FOR…DO	结构文本	C-19
一旦特定条件为真，重复执行某项操作	WHILE…DO	结构文本	C-22
直到条件为真，否则重复执行某项操作	REPEAT…UNTIL	结构文本	C-25

IF…THEN

当特定条件发生时，使用IF…THEN语句执行某项操作。

操作数：



结构文本

操作数：	类型：	格式：	输入：
bool_expression 布尔表达式	BOOL	标签表达式	计算出布尔值(布尔表达式)的布尔型标签或表达式

说明：语法是：

```
IF bool_expression1 THEN
      <statement >;          ← 当bool_expression1 为真时，执行语句。
      .
      .
      .
可选  ELSIF bool_expression2 THEN
      <statement>;           ← 当bool_expression2 为真时，执行语句。
      .
      .
      .
可选  ELSE
      <statement>;           ← 当两个表达式均为假时，执行语句。
      .
      .
      .
END_IF;
```

根据以下准则使用ELSIF或ELSE：

1. 如果从多个语句组中选出执行条件，需添加一个或多个ELSIF语句。

   - 每个ELSIF表示一个选择路径。
   - 根据用户需要指定的ELSIF路径数。
   - 控制器执行首个条件为真的IF或ELSIF语句，并跳过其余的ELSIFs和ELSE语句。

2. 如果要求当所有IF或ELSIF条件为假时执行语句，需添加一条ELSE语句。

下表总结了IF,WHEN,ELSIF和ELSE语句的多种组合。

如果用户要求：	并且：	那么使用此结构体：
当条件为真时，执行语句	如果条件为假，不执行语句	IF…THEN
	如果条件为假，执行其它操作	IF…THEN…ESLE
根据输入条件，从多个语句(语句组)中选择	如果条件为假，不执行语句	IF…THEN…ELSIF
	如果所有条件为假，分配缺省语句	IF…THEN…ELSIF…ELSE

实例1：IF…THEN

如果用户希望：	输入结构文本程序：

如果 rejects > 3 那么
conveyor = off (0)
alarm = on (1)
	IF rejects > 3 THEN
	  conveyor := 0;
	  alarm := 1;
	END_IF;

实例2：IF…THEN...ELSE

如果用户希望：	输入结构文本程序：

如果传送带方向触点为前进 (1)那么
指示灯灭
否则 指示灯亮
	IF conveyor_direction THEN
	  light := 0;
	ELSE
	  light [:=] 1;
	END_IF;

当控制器进入以下状态，[:=]会将light清零：

- 进入RUN模式
- 如果用户组态SFC为自动复位，离开SFC的步时(仅适用于用户在步动作中嵌入赋值语句或使用JSR指令调用结构文本的例程的情况)

实例3：IF...THEN…ELSIF

如果用户希望：	输入结构文本：

如果糖料上下限位开关均导通，那么
进给阀打开(导通)
直至糖料上限位开关关闭
	IF Sugar.Low & Sugar.High THEN
	  Sugar.Inlet [:=] 1;
	ELSIF NOT(Sugar.High) THEN
	  Sugar.Inlet := 0;
	END_IF;

当控制器进入以下状态，[:=]将Sugar.Inlet清零：

- 进入RUN模式
- 如果用户组态SFC为自动复位，离开SFC的步时(仅适用于用户在步动作中嵌入赋值语句或使用JSR指令调用结构文本的例程的情况)

实例4：IF…THEN...ELSIF...ELSE

如果用户希望：	输入结构文本程序：

如果罐温度>100
那么设置泵缓慢运行
如果罐温度>200
那么设置泵快速运行
其它情况关闭泵
	IF tank.temp > 200 THEN
	  pump.fast :=1; pump.slow :=0; pump.off :=0;
	ELSIF tank.temp > 100 THEN
	  pump.fast :=0; pump.slow :=1; pump.off :=0;
	ELSE
	  pump.fast :=0; pump.slow :=0; pump.off :=1;
	END_IF;

CASE…OF

使用CASE语句根据数值选择执行的操作。

操作数：

结构文本

```
CASE numeric_expression OF
     selector1: statement;
     selectorN: statement;
ELSE
     statement;
```

操作数：	类型：	格式：	输入：
numeric_expression 数值表达式	SINT INT DINT REAL	标签表达式	计算出数值(数值表达式)的标签或表达式
selector 选择器	SINT INT DINT REAL	立即数	与numeric_expression类型相同

重要事项

REAL型数值极有可能在一定的数据范围而非精确匹配某个特定数值，因此如果用户使用REAL数值，必须为选择器设定一个数值范围。

说明：语法：

```
CASE numeric_expression OF
    selector1:  <statement>;
                .
                .
                .
    selector2:  <statement>;
                .
                .
                .
    selector3:  <statement>;
                .
                .
                .
    ELSE
                <statement>;
                .
                .
                .
END_CASE;
```

根据用户需要指定的选择器的值(路径数)。

当numeric_expression = 选择器1，执行的语句

当numeric_expression = 选择器2，执行的语句

当numeric_expression =选择器3，执行的语句

可选

当numeric_expression ≠ 任一选择器，执行的语句

获取有效的选择器数值，请参见下页表格。

输入的选择器数值的语法为：

当选择器为：	输入：
值	数值: 语句
多个，分散的数值	数值1,数值2,数值N:<语句> 使用逗号(,)将每个值分开。
数值范围	数值1..数值N:<语句> 使用两个句点(..)标记范围。
分散得数值加上一个数值范围	数值a,数值b,数值1..数值N:<语句>

CASE结构体与C或C＋＋编程语言中的转换语句类似。可是，用CASE结构体，控制仅执行与第一个匹配选择器数值相结合的语句。在这个选择器的语句后，执行总是被中断，并转向END_CASE语句。

实例：

如果用户希望：	输入结构文本程序：
如果配方号＝1，那么	CASE recipe_number OF
成分A出口1＝打开(1)	1: Ingredient_A.Outlet_1 :=1;
成分B出口4＝打开(1)	Ingredient_B.Outlet_4 :=1;
如果配方号＝2或3，那么	2,3: Ingredient_A.Outlet_4 :=1;
成分A出口4＝打开(1)	Ingredient_B.Outlet_2 :=1;
成分B出口2＝打开(1)	
如果配方号＝4,5,6或7，那么	4..7: Ingredient_A.Outlet_4 :=1;
成分A出口4＝打开(1)	Ingredient_B.Outlet_2 :=1;
成分B出口2＝打开(1)	8,11..13 Ingredient_A.Outlet_1 :=1;
如果配方号＝8,11,12或13，那么	Ingredient_B.Outlet_4 :=1;
成分A出口1＝打开(1)	
成分B出口4＝打开(1)	ELSE
其它所有的出口＝关闭(0)	Ingredient_A.Outlet_1 [:=]0;
	Ingredient_A.Outlet_4 [:=]0;
	Ingredient_B.Outlet_2 [:=]0;
	Ingredient_B.Outlet_4 [:=]0;
	END_CASE;

当控制器进入以下状态，[:=]会将出口的标签清零：

- 进入RUN模式
- 如果用户组态SFC为自动复位，离开SFC的步时(仅适用于用户在步动作中嵌入赋值语句或使用JSR指令调用结构文本的例程的情况)

FOR…DO

使用FOR…DO循环在完成其它工作前，执行指定次数的某工作。

操作数：

```
FOR count:= initial_value TO
final_value BY increment DO
     <statement>;
END_FOR;
```

结构文本

操作数：	数据类型：	格式：	说明：
Count计数值	SINT INT DINT	标签	FOR…DO结构体执行时存储计数值的位置
initial_value初始值	SINT INT DINT	标签 表达式 立即数	必须指定计数初始值
final_value结束值	SINT INT DINT	标签 表达式 立即数	指定计数结束值，确定何时退出循环
increment增量	SINT INT DINT	标签 表达式 立即数	(可选)循环一次时计数值的增量 如果用户不指定增量，计数值增量默认为1。

重要事项

确定在一个扫描周期内重复循环执行的次数不能过多。

- 控制器在完成循环前无法执行例程中其它语句。
- 如果循环完成所用的时间超出了任务看门狗定时器的值，产生一个主要故障。
- 考虑使用不同结构体，如IF…THEN。

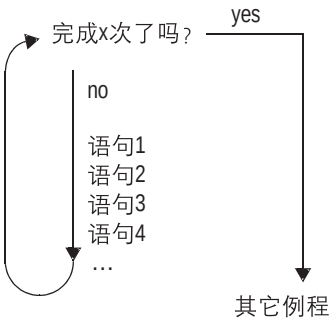
说明：语法是：

```
FOR count := initial_value
   TO final_value
   BY increment
   DO
   <statement>;
   IF bool_expression THEN
      EXIT;
   END_IF;
END_FOR;
```

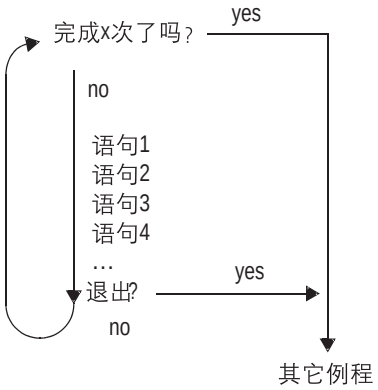
可选（BY increment）：如果用户没有指定增量值，循环增量值为1。

可选（IF bool_expression THEN … END_IF;）：如果需要提前退出循环，可使用其它语句，例如采用IF…THEN结构体作为EXIT语句。

以下图示了FOR…DO循环如何执行以及如何使用EXIT语句提前退出循环。



FOR…DO循环结构体执行特定次数。



使用EXIT语句在计数值达到最终值前终止循环。

实例1：

如果用户希望：	输入结构文本：

清除布尔型数组的0-31位：
1. 初始化subscript标签为0。
2. 清除array[subscript]。例如，当subscript＝5，清除array[5]。
3.subscript加1。
4. 如果subscript<＝31，重复步骤2和3。否则退出循环。
	For subscript:=0 to 31 by 1 do
	array[subscript] := 0;
	End_for;

实例2：

如果用户希望：	输入结构文本：

通过用户自定义数据类型(结构)存储用户的货物库存信息：
• 货物的条形码ID号(字符串数据类型)
• 货物的采购数量(DINT数据类型)
上面的自定义结构体数组包括了库存货物的各个元素。用户希望使用条形码ID在数组中搜索特定产品，并确定采购数量。
1. 设置Inventory数组的大小(货物数)并将结果存储到Inventory_Items(DINT标签)。
2. 初始化position标签值为0。
3. 如果Barcode与数组中某条目的ID一致，那么：
a. 设置Quantity标签=Inventory[position].Qty，生成该货物库存量。
b. 结束。
Barcode是存储货物条形码ID号的字符串标签，用户搜索该项。例如，当position=5，比较Barcode和Inventory[5].ID。
4. 将position值加1。
5. 如果position<=(Inventory_Items-1)，重复步骤3和4。
元素编号起始于0，且最后一个元素编号比数组的元素号少1。否则，停止。
	SIZE(Inventory,0,Inventory_Items);
	For position:=0 to Inventory_Items - 1 do
	If Barcode = Inventory[position].ID then
	Quantity := Inventory[position].Qty;
	
	Exit;
	End_if;
	End_for;

WHILE…DO

只要条件为真，则使用WHILE…DO循环连续工作。

操作数：

```
WHILE bool_expression DO
    <statement>;
END_WHILE;
```

结构文本

操作数：	类型：	格式：	输入：
bool_expression布尔表达式	BOOL	标签表达式	计算出布尔值的BOOL标签或表达式

重要事项

确定在一个扫描周期内重复循环执行的次数不能太多。

- 控制器在完成循环前无法执行例程中其它语句。
- 如果循环完成所用的时间超出了任务看门狗定时器的值，产生一个主要故障。
- 考虑使用不同的结构体，如IF…THEN。

说明：语法为：

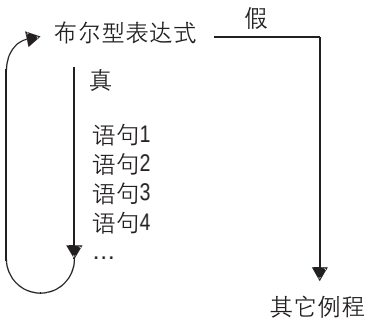
```
WHILE bool expression1 DO
   <statement>;
   IF bool_expression2 THEN
     EXIT;
   END_IF;
END_WHILE;
```

当bool_expression1为真时，执行语句。

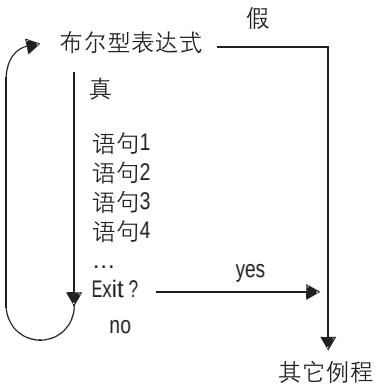
可选

如果需要提前退出循环，可使用其它语句，例如采用IF…THEN结构体作为EXIT语句退出条件。

以下图示了FOR…DO循环如何执行以及如何使用EXIT语句提前退出循环。



当bool_expression表达式值为真，控制器仅执行WHILE…DO循环内的语句。



在条件为真前，使用EXIT语句退出循环。

实例1：

如果用户希望：	输入结构文本：

WHILE…DO循环首先判断其条件。如果条件为真，然后控制器执行循环内的语句。
 	
与REPEAT…UNTIL循环的区别在于REPEAT…UNTIL循环执行结构体内语句，然后再次执行语句前确定条件是否为真。这样，REPEAT…UNTIL循环至少执行一次。而WHILE…DO循环中的语句可能从不执行。
	pos := 0;
	While ((pos <= 100) & structarray[pos].value <> targetvalue)) do
	  pos := pos + 2;
	  String_tag.DATA[pos] :=SINT_array[pos];
	end_while;

实例2：

如果用户希望：	输入结构文本：

将SINT数组中ASCII字符移入字符串标签。(在SINT数组中，每个元素保存一个字符。)当到达结束符时停止。
1．初始化Element_number值为0。
2．计算SINT_array中的元素数(数组中包含ASCII字符)并将结果存入SINT_array_size(DINT标签)。
3．如果SINT_array[element_number]的字符为13(结束符对应的十进制数值)，则停止循环。
4．设置String_tag[element_number]= SINT_array[element_number]。
5．element_number加1。控制器检查SINT_array中下一个字符。
6．设置String_tag的长度值为element_number。(该值记录当前String_tag中的字符号。)
7．如果element_number＝SINT_array_size，则停止循环。(已执行到数组末尾但无结束符。)
8．跳到步骤3。
	element_number := 0;
	
	SIZE(SINT_array, 0, SINT_array_size);
	While SINT_array[element_number] <> 13 do
	String_tag.DATA[element_number] :=
	SINT_array[element_number];
	element_number := element_number + 1;
	String_tag.LEN := element_number;
	If element_number = SINT_array_size then
	exit;
	end_if;
	end_while;

REPEAT…UNTIL

REPEAT…UNTIL循环连续执行语句，直至条件为真。

操作数：

```
REPEAT
    <statement>;
UNTIL bool_expression
END_REPEAT;
```

结构文本

操作数：	类型：	格式：	输入：
bool_expression 布尔表达式	BOOL	标签表达式	计算出布尔值(布尔型表达式)的BOOL标签或表达式

重要事项

确定在一个扫描周期内重复循环执行的次数不能过多。

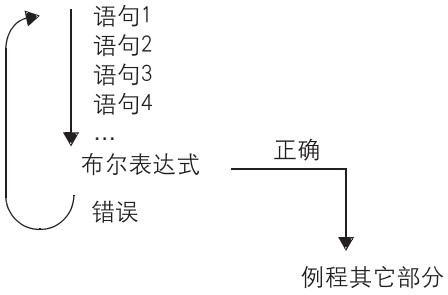
- 控制器在完成循环前无法执行例程中其它语句。
- 如果循环完成所用的时间超出了任务看门狗定时器的值，产生一个主要故障。
- 考虑使用不同的结构体，如IF…THEN。

说明：语法是：

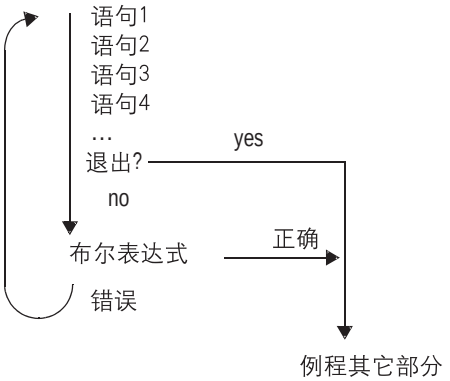
```
REPEAT
   <statement>;          <-- 当bool_expression1为假时，执行语句
   IF bool_expression2 THEN
     EXIT;               <-- 如果需要提前退出循环，可使用其它语句，例如采用IF…THEN结构体作为EXIT语句。
   END_IF;
UNTIL bool_expression1
END_REPEAT;
```

可选

以下图示了FOR…DO循环如何执行以及如何使用EXIT语句提前退出循环。



当bool_expression条件为假，控制器仅执行REPEAT…UNTIL循环内的语句。



在条件为假前退出循环，使用EXIT语句。

实例1:

如果用户希望:	输入结构文本:

REPEAT…UNTIL 首先执行循环内的语句，在再次执行循环内语句前，确定条件是否为真。
 	
REPEAT…UNTIL 与WHILE…DO循环的区别在于WHILE…DO循环首先判断条件。如果条件为真，处理器执行循环内语句。REPEAT…UNTIL 循环内语句至少执行一次。而WHILE…DO循环内的语句可能从不执行。
	pos := -1;
	REPEAT
	  pos := pos + 2;
	UNTIL ((pos = 101) OR
	(structarray[pos].value =targetvalue))
	end_repeat;

实例2：

如果用户希望：	输入此结构文本：

将SINT数组中ASCII字符移入字符串标签。(在SINT数组中，每个元素保存一个字符。)到达结束符时停止循环。
1. 初始化Element_number为0。
2. 计算SINT_array中的元素数(数组中包含ASCII字符)并将结果存入SINT_array_size (DINT标签)。
3. 设置String_tag[element_number]= SINT_array[element_number]中字符。
4. element_number加1。使得控制器检查SINT_array中下一个字符。
5. 设置String_tag的长度值为element_number。(该值记录当前String_tag中的字符号。)
6. 如果element_number＝SINT_array_size，则停止循环。(已执行到数组末尾但无结束符。)跳到步骤3。
7. 如果SINT_array[element_number]的字符为13(结束符的十进制数值)，则停止循环。否则，跳转到步骤3。
	element_number := 0;
	SIZE(SINT_array, 0,SINT_array_size);
	Repeat
	String_tag.DATA[element_number] := SINT_array[element_number];
	element_number := element_number + 1;
	String_tag.LEN := element_number;
	If element_number = SINT_array_size then
	exit;
	end_if;
	Until SINT_array[element_number] = 13
	end_repeat;

注释

为使用户编写的结构文本程序更具可读性，需添加注释。

- 用户通过注释可以使用清晰的语言来描述结构文本程序的工作流程
- 注释不影响结构文本程序执行

添加注释：	使用以下格式：
单独一行	//comment (*comment*)
结构文本程序的最后一行 一行结构文本程序内	/*comment*/ (*comment*) /*comment*/
范围超过一行	(*注释起始…注释结束*) /*注释起始…注释结束*/

实例：

格式：	实例：

//comment	行起始
	//Check conveyor belt direction IF conveyor_direction THEN...
	行尾
	″ELSE //If conveyor isn.t moving, set alarm light″
	light := 1;
	END_IF;

(*comment*)	Sugar.Inlet[:=]1;(*open the inlet*) IF Sugar.Low (*low level LS*)& Sugar.High (*high level LS*)THEN... (*Controls the speed of the recirculation pump. The speed depends on the temperature in the tank.*)IF tank.temp > 200 THEN...
/*comment*/	Sugar.Inlet:=0;/*close the inlet*/ IF bar_code=65 /*A*/ THEN... ″/*Gets the number of elements in the Inventory array and stores the value in the Inventory_Items tag*/SIZE(Inventory,0,Inventory_Items);″

T

注释:

www.rockwellautomation.com.cn

动力、控制与信息解决方案

Americas: Rockwell Automation, 1201 South Second Street, Milwaukee, WI 53204-2496 USA, Tel: (1)414 382.2000, Fax: (1)414 382.4444

亚太地区 – 香港数码港道100号数码港3座F区14楼 电话：(852)28874788 传真：(852)25109436

北京 – 北京市建国门内大街18号恒基中心办公楼1座4层 邮编：100005 电话：(8610)65182535 传真：(8610)65182536
上海 – 上海市仙霞路319号远东国际广场A幢7楼 邮编：200051 电话：(8621)61206007 传真：(8621)62351099
厦门 – 厦门市湖里工业区悦华路38号 邮编：361006 电话：(86592)6022084 传真：(86592)6021832
沈阳 – 沈阳市沈河区青年大街219号华新国际大厦15-F单元 邮编：110015 电话：(8624)23961518 传真：(8624)23963539
武汉 – 武汉市建设大道568号新世界国贸大厦I座2202室 邮编：430022 电话：(8627)68850233 传真：(8627)68850232
广州 – 广州市环市东路362号好世界广场2703-04室 邮编：510060 电话：(8620)83849977 传真：(8620)83849989
重庆 – 重庆市渝中区邹容路68号大都会商厦3112-13室 邮编：400010 电话：(8623)63702668 传真：(8623)63702558
大连 – 大连市西岗区中山路147号森茂大厦2305层 邮编：116011 电话：(86411)83687799 传真：(86411)83679970
西安 – 西安市南大街30号中大国际大厦712室 邮编：710002 电话：(8629)87203577 传真：(8629)87203123
深圳 – 深圳市深南东路5047号深圳发展银行大厦15L 邮编：518001 电话：(86755)25847099 传真：(86755)25870900
南京 – 南京市中山南路49号商茂世纪广场44楼A3-A4座 邮编：210005 电话：(8625)86890445 传真：(8625)86890142
青岛 – 青岛市香港中路36号新世界数码港招银大厦1006室 邮编：266071 电话：(86532)86678338 传真：(86532)86678339
成都 – 成都市总府路2号时代广场A座906室 邮编：610016 电话：(8628)86726886 传真：(8628)68726887

Publication 1756-RM003G-ZH March 2006