Allen-Bradley

Logix5550 控制器

(Cat.No. 1756-L1)

指令集
参考手册

介绍

本手册包含很多新增的及修改的信息。为了有助于用户查找新的和修改的信息，我们在相关章节两侧添加了醒目条,如本段文字左侧所示。

更新信息

本手册已经全部更新。其最明显的改变是:

- 支持任意型MSG指令
- 修改了WALLCLOCKTIME对象
- 修改了PID指令
- 修改了运动控制指令和结构

注释:

使用本手册

介绍

本手册是 ControlLogix 系列出版物之一.

任务 / 目的:	参见资料:
安装控制器及其组件	Logix5550 控制器快速启动，出版号 1756-10.1 Logix5550 内存板安装指南，出版号 1756-5.33
控制器使用	Logix5550 控制器用户手册，出版号 1756-6.5.12 控制器编程
控制器编程 **既本手册** 	Logix5550 控制器指令集参考手册，出版号 1756-6.4.1
离散量 I/O 模块的配置与通讯	*离散量模块用户手册*，出版号 1756-6.5.8
配置模拟量 I/O 模块	*模拟量模块用户手册*，出版号 1756-6.5.9
框架的选择与安装	*ControlLogix* 框架安置指南，出版号 1756-5.69
电源的选择与安装	*ControlLogix* 电源安装指南，出版号 1756-5.1

本手册的使用对象

本手册为程序设计者提供关于Logix5550控制器每条指令的详细信息。用户首先应该已经熟悉了Logix5550 控制器是怎样存储和处理数据的。

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

本手册的用途

本手册提供Logix5550 控制器支持的每条指令的详细信息。每条指令都按下列格式说明。

在如下区域:	提供下列信息:
指令名称	标识指令 定义指令是输出指令或输入指令。
操作数	列出指令可以使用的所有操作数类型。
控制结构体	如果指令需要控制结构体，则列出控制状态位和数值。
说明指令使用说明	说明指令的使用 如果需要，说明指令使能和禁止时的区别。
执行	详细说明了指令在下列条件下是怎样操作的: • 预扫描 • 梯级输入条件为假 • 梯级输入条件为真
算术状态标志	确定指令是否影响算术状态标志位。参见附录 A
故障条件	确定指令是否产生次要或主要故障，如果产生，定义故障类型及故障码
举例	至少提供一个编程实例 每个实例都包含解释说明

相关项的约定

设置和清零

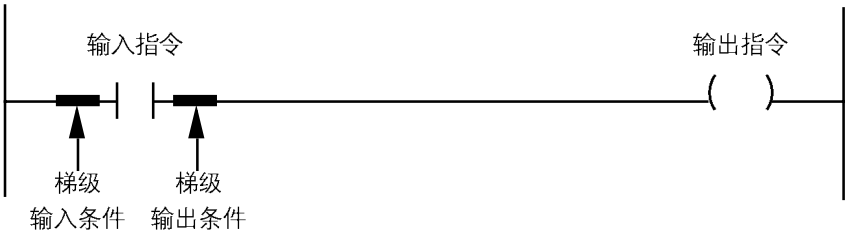
本手册用设置和清零来定义位(布尔型)和数值(非布尔型)的状态

项:	意义:
设置	设置一位为 1(ON) 设置一数值为非零值
清零	清零一位(OFF) 清零一个数值的所有位

在操作数部分，**黑体字**数据类型是最优数据类型。如果指令的所有操作数都使用同一最优数据类型，则指令执行速度快，且占用内存少，典型的最优数据类型是 DINT 或 REAL。

梯级条件

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。然后根据梯级输入条件和指令，设置指令后面的梯级条件(梯级输出条件)。这样交替影响后面的指令。



如果一条输入指令的梯级输入条件为真，则控制器评价该指令，并且根据指令执行的结果，设置梯级输出条件。如果指令被评价为真，则梯级输出条件为真；如果指令被评价为假，则梯级输出条件为假。

所有指令的通用信息

Logix5550 指令集有一些通用属性：

下列信息：	参见附录：
通用属性	在附录 A 详细说明： • 算术状态标志 • 数据类型 • 关键字
数组	在附录 B 详细说明了数组的概念，并解释了控制器是怎样处理数组的。
结构体	附录 C 阐述了控制器支持的控制结构体

注释:

目录

第五章 计算 / 算术指令

第六章 传送 / 逻辑指令

第七章 数组 (文件) / 综合指令

第八章 数组(文件)/ 位移指令

第九章 顺序器指令

第十章 程序控制指令

第十一章 循环 (For) / 终止循环 (Break) 指令

第十二章 专用指令

第十三章 三角函数指令

第十四章 高级算术指令

位指令

(XIC，XIO，OTE，OTL，OTU，ONS，OSR，OSF)

简介

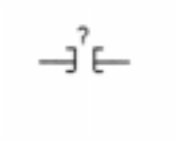
位(继电器 - 类型)指令用于监视和控制位的状态。

如果用户要:	使用下列指令:	参见页次:
当一位被置位时使能输出	XIC	1-2
当一位被清零时使能输出	XIO	1-4
使一位置位	OTE	1-6
使一位置位(保持)	OTL	1-7
使一位清零(保持)	OTU	1-8
每次梯级变为真时使输出使能一个扫描周期	ONS	1-9
每次梯级变为真时使一位置位一个扫描周期	OSR	1-11
每次梯级变为假时使一位置位一个扫描周期	OSF	1-14

检查是否闭合指令 (XIC)

XIC 指令是一条输入指令。

操作数:

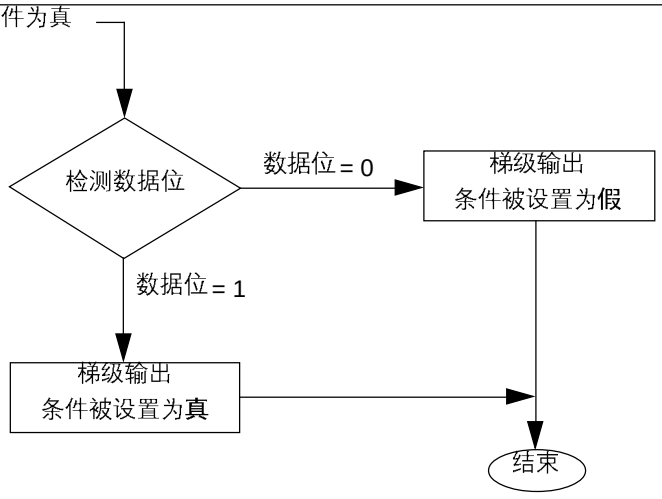


操作数:	数据类型:	格式:	说明:
数据位	BOOL	标签	被检测的位

说明: XIC 指令检查数据位看它是否是置位状态。

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	

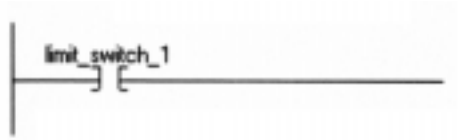


算术状态标志: 不影响

故障条件: 无

XIC 指令举例:

例 1



如果限位开关 1 *(limit_switch_1)* 被置位，则使能下一条指令 (梯级输出条件为真)

例 2

如果 *S:V* 被置位 (表明已经发生溢出)，则使能下一条指令 (梯级输出条件为真)

其他格式:

格式:	句法:
neutral 文本	`XIC` *`(data_bit);`*
ASCII 文本	`XIC` *`data_bit`*

相关指令: XIO

检查是否断开指令 (XIO)

XIO 指令是一条输入指令。

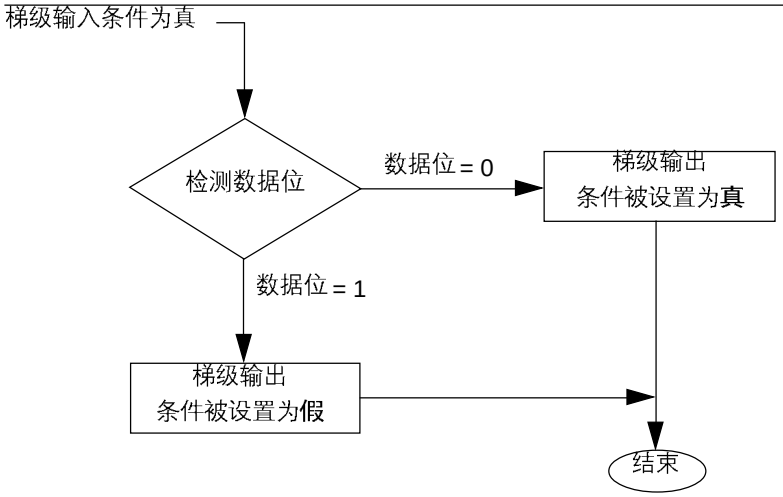
操作数:

操作数:	数据类型:	格式:	说明:
数据位	BOOL	标签	被检测的位

说明: XIO 指令检查数据位看它是否是清零状态。

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



算术状态标志: 不影响

故障条件: 无

XIO 指令举例:

例 1

如果限位开关 2 (limit_switch_2) 被清零，则使能下一条指令(梯级输出条件为真)

例 2

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

其他格式:

格式:	句法:
neutral 文本	`XIO (data_bit);`
ASCII 文本	`XIO data_bit`

相关指令: XIC

输出激励指令 (OTE)

OTE 指令是一条输出指令。

操作数:

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

说明: OTE 指令置位或清零数据位。

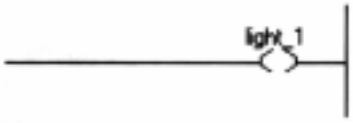
执行:

条件:	动作:
预扫描	数据位被清零 梯级输出条件被设置为假
梯级输入条件为假	数据位被清零 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

算术状态标志: 不影响

故障条件: 无

OTE 指令举例:



当指令被使能时，OTE 指令使指示灯 1 *(light_1)* 置位 (接通)。
当指令被禁止时，OTE 指令使指示灯 1 *(light_1)* 复位 (断开)。

其他格式:

格式:	句法:
neutral 文本	`OTE (data_bit);`
ASCII 文本	`OTE data_bit`

相关指令: OTL，OTU

输出锁存指令 (OTL)

OTL 指令是一条输出指令。

操作数:

操作数:	数据类型:	格式:	说明:
数据位	BOOL	标签	位被置位

说明: OTL 指令置位(锁存)数据位。

当使能时，OTL 指令置数据位位。数据位保持置位直到被清零，一般被一条 OTU 指令清零。当指令被禁止时，OTL 指令不改变数据位的状态。

执行:

条件:	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	置位数据位 梯级输出条件被设置为真

算术状态标志: 不影响

故障条件: 无

OTL 指令举例:

当指令被使能时，OTL 指令使指示灯 2 *(light_2.)* 置位(接通)。该位保持置位直到被清零，一般被一条 OTU 指令清零。

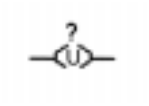
其他格式:

格式:	句法:
neutral 文本	*OTL (data_bit);*
ASCII 文本	OTL *data_bit*

相关指令: OTU，OTE

输出解锁存指令 (OTU)

OTU 指令是一条输出指令。



操作数:

操作数:	数据类型:	格式:	说明:
数据位	BOOL	标签	被清零的位

说明: OTU 指令 清零(解锁存)数据位。

当指令被使能时，OTU指令清零数据位。被禁止时OTU指令不改变数据位的状态。

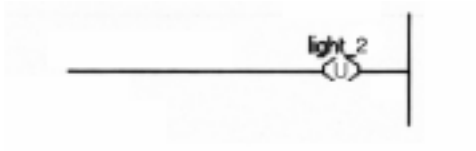
执行:

条件:	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

算术状态标志: 不影响

故障条件: 无

OTU 指令举例:



当指令被使能时，OTU 指令使指示灯 2 *(light_2)* 清零(断开)。

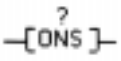
其他格式:

格式:	句法:
neutral 文本	`OTU (data_bit);`
ASCII 文本	`OTU data_bit`

相关指令: OTL，OTE

一次响应指令 (ONS)

指令是一条输入指令。



操作数:

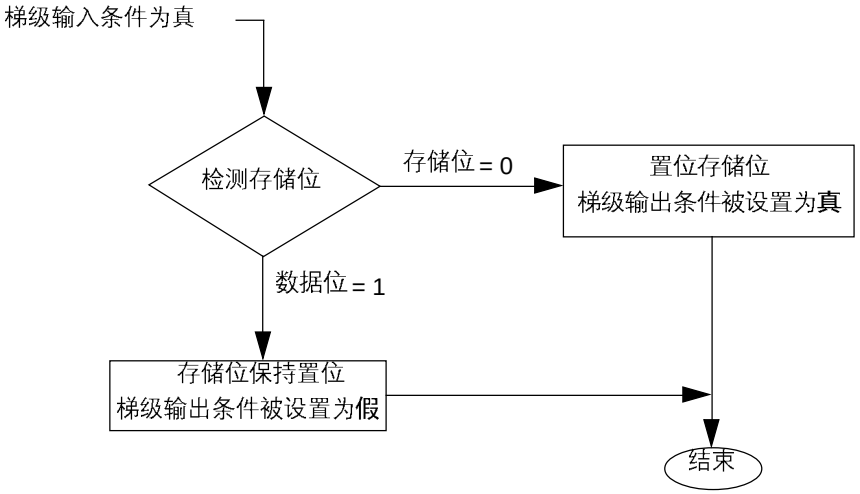
操作数:	数据类型:	格式:	说明:
存储位	BOOL	标签	内部存储位 存储指令最近一次执行的梯级输入条件

说明: ONS指令根据存储位的状态使能或禁止梯级的其余部分。

如果指令被使能时存储位清零，则 ONS 指令使能梯级的其余部分。如果被禁止或存储位置位，ONS指令禁止梯级的其余部分。

执行:

条件:	动作:
预扫描	置位存储位以防止在首次扫描期间发生无效的触发。梯级输出条件被设置为假
梯级输入为假	清零存储位 梯级输出条件被设置为假



算术状态标志: 不影响

故障条件: 无

ONS 指令举例：

一般在 ONS 指令前面用一条输入指令，因为扫描 ONS 指令的正常操作是它一使能之后就禁止。一旦 ONS 指令被使能，则只有梯级输入条件为假或存储位清零，ONS 指令才能再次被使能。

在扫描时，如果限位开关 1 是清零状态或存储位是置位状态，则不影响梯级。如果当扫描时限位开关 1 是置位状态且存储位是清零状态，则 ONS 指令置位存储位 1 且 ADD 指令的和加 1。只要限位开关 1 保持置位，ADD 指令的和数值就保持不变。必须在限位开关 1 再次从清零变为置位，和的值才再增加。

其他格式：

格式：	句法：
neutral 文本	`ONS (storage_bit);`
ASCII 文本	`ONS storage_bit`

相关指令： OSR，OSF

上升沿触发指令

OSR 指令是一条输出指令

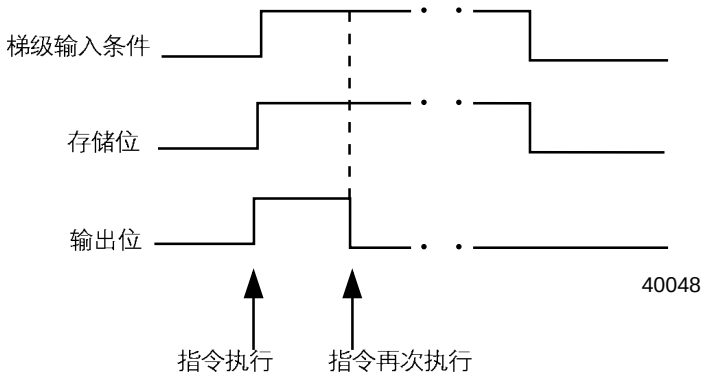
操作数:

操作数:	数据类型:	格式:	说明:
存储位	BOOL	标签	内部存储位 存储指令最近一次执行的梯级输入条件
输出位	BOOL	标签	被设置的位

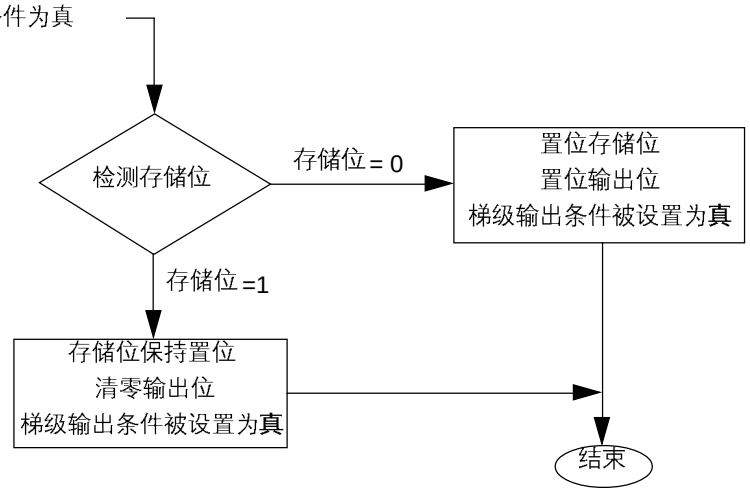
说明:

OSR 指令根据存储位的状态置位或清零输出位。

如果指令被使能时存储位清零，则 OSR 指令置位输出位。如果使能时存储位置位或禁止，则 OSR 指令清零输出位。



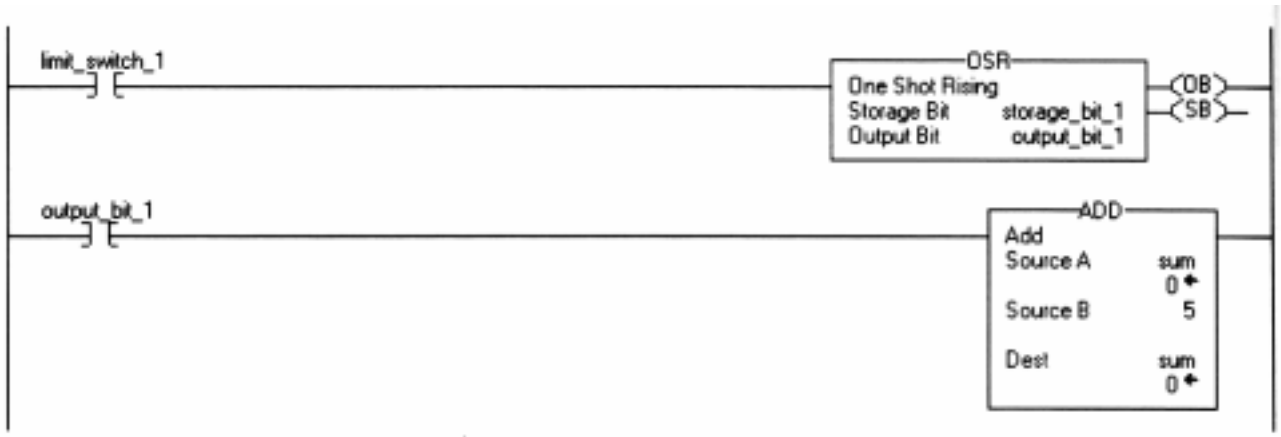
执行:

条件:	动作:
预扫描	置位存储位以防止在首次扫描期间发生无效的触发。 清零输出位 梯级输出条件被设置为假
梯级输入为假	存储位被清零 输出位不变 梯级输出条件被设置为假
梯级输入条件为真	

算术状态标志: 不影响

故障条件: 无

OSR 指令举例:



每次限位开关 1 从清零状态变为置位时，OSR 指令置位输出位 1 并且 ADD 指令的和加 5。只要限位开关 1 保持置位，和的值就保持不变。必须在限位开关 1 再次从清零变为置位，和的值才再增加。用户可以在多个梯级使用输出位 1 触发其他操作。

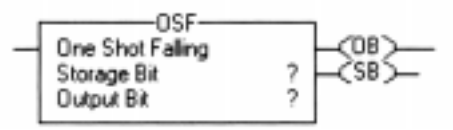
其他格式:

格式:	句法:
neutral 文本	`OSR (storage_bit,output_bit);`
ASCII 文本	`ONS storage_bit output_bit`

相关指令: OSF，ONS

下降沿触发指令 (OSF)

OSF 指令是一条输出指令。



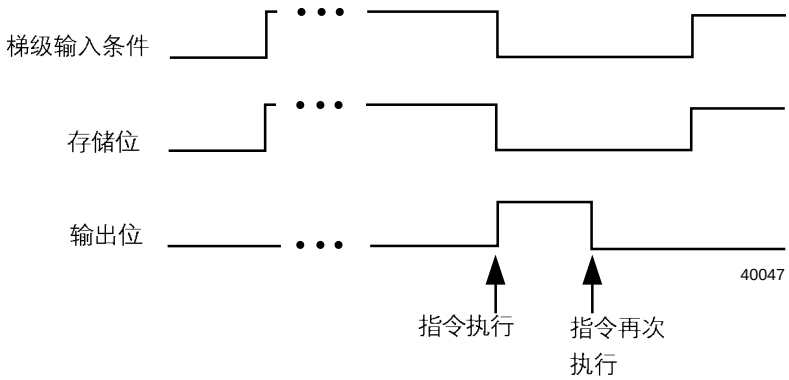
操作数:

操作数:	数据类型:	格式:	说明:
存储位	BOOL	标签	内部存储位 存储上次指令执行的梯级输入条件
输出位	BOOL	标签	位被置位

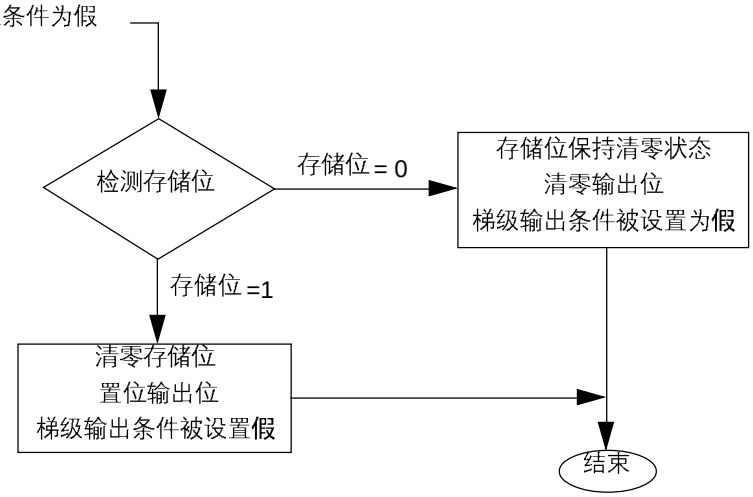
说明:

OSF 指令根据存储位的状态置位或清零输出位。

当指令被禁止时存储位置位，OSF 指令置位输出位。如果指令禁止或使能时存储位是清零状态，则OSF指令清零输出位。



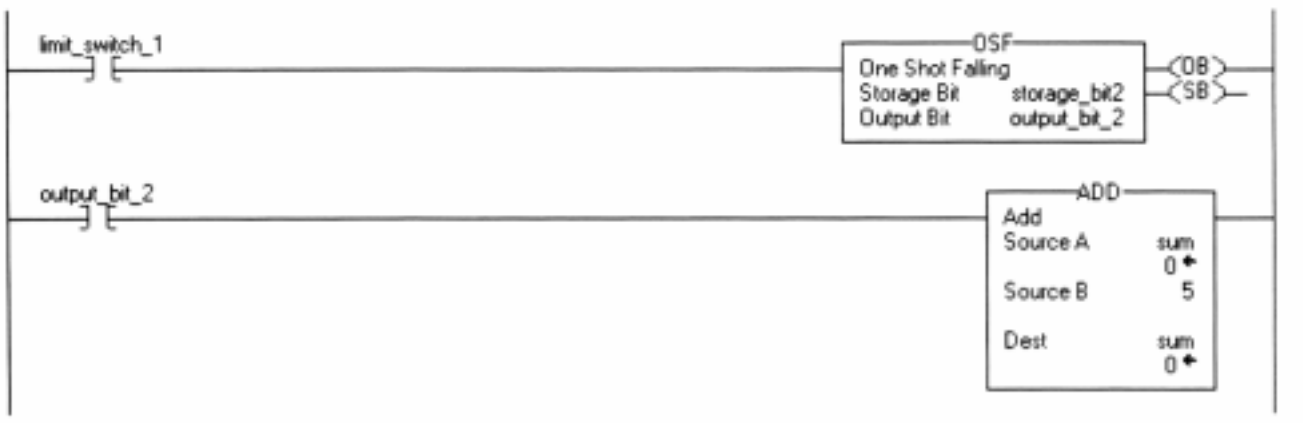
执行:

条件:	动作:
预扫描	存储位被清零以防止在首次扫描期间发生无效的触发。 输出位清零 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	存储位被置位 输出位被清零 梯级输出条件被设置为真

算术状态标志: 不影响

故障条件: 无

OSF 指令举例:



每次限位开关 1 从置位变为清零时，OSF 指令置位输出位 2 并且 ADD 指令的和加 5。只要限位开关 1 保持清零状态，和的值就保持不变。必须在限位开关 1 再次从置位变为清零状态，和的值才再增加。用户可以在多个梯级使用输出位 2 触发其他操作。

其他格式:

格式:	句法:
neutral 文本	OSF *(storage_bit,output_bit);*
ASCII 文本	OSF *storage_bit output_bit*

相关指令:

OSR，ONS

计时器和计数器指令

(TON，TOF，RTO，CTU，CTD，RES)

简介

计时器和计数器根据事件的时间和次数控制操作.

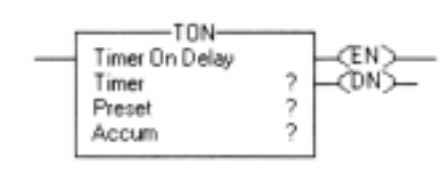
如果用户要:	使用下列指令:	参见页次:
计时计时器被使能的时间	TON	2-2
计时计时器被禁止的时间	TOF	2-5
计时时间的累加值	RTO	2-8
加计数	CTU	2-11
减计数	CTD	2-14
复位计时器或计数器	RES	2-18

计时器的时间基都是 1 毫秒。

延时导通计时器指令 (TON)

TON 指令是一条输出指令。

操作数:



操作数:	数据类型:	格式:	说明:
计时器	TIMER	标签	计时器结构
预置值	DINT	立即数	延时时间 (累积的时间值)
累加值	DINT	立即数	计时器已经计数的毫秒数，初始值一般为 0

计时器结构:

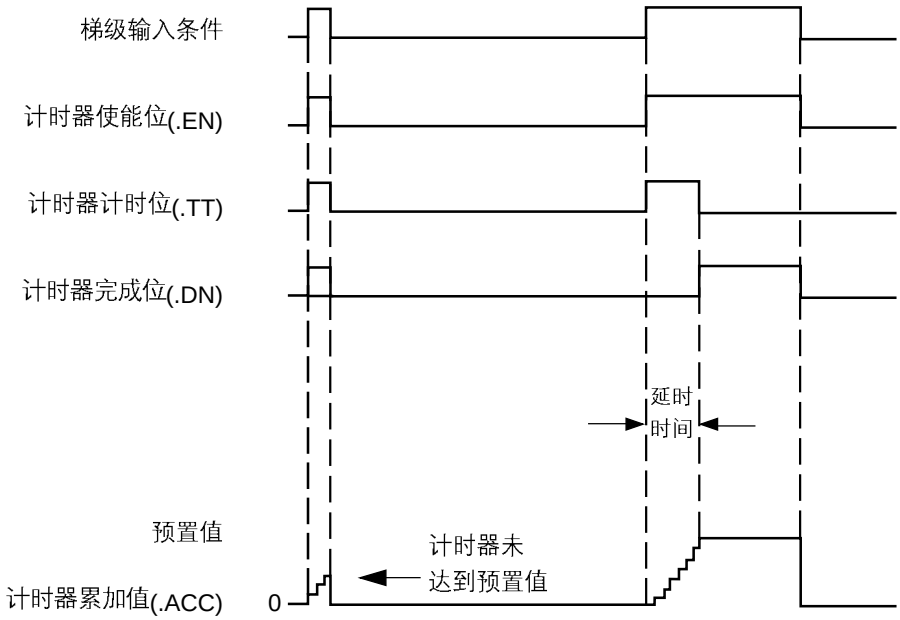
助记符:	数据类型:	说明:
.EN	BOOL	使能位—标识 TON 指令被使能。
.TT	BOOL	计时位—标识计时操作正在进行中。
.DN	BOOL	完成位—标识累加值 (.ACC) ≥ 预置值(.PRE)。
.PRE	DINT	预置值—指定在指令置位完成位 (.DN)时累加器所达到的值 (以 1 毫秒为单位)。
.ACC	DINT	累加值—表示从 TON 指令被使能开始已经经过的毫秒数。

说明: TON 是一条非保持的计时器指令，当该指令被使能时累计时间。计时器的时间基总是1毫秒。列如，对于一个2-秒的计时器，其预置值(.PRE)应该输入 2000。

当指令被使能时，TON 计时器指令累计时间直到发生下列事件:

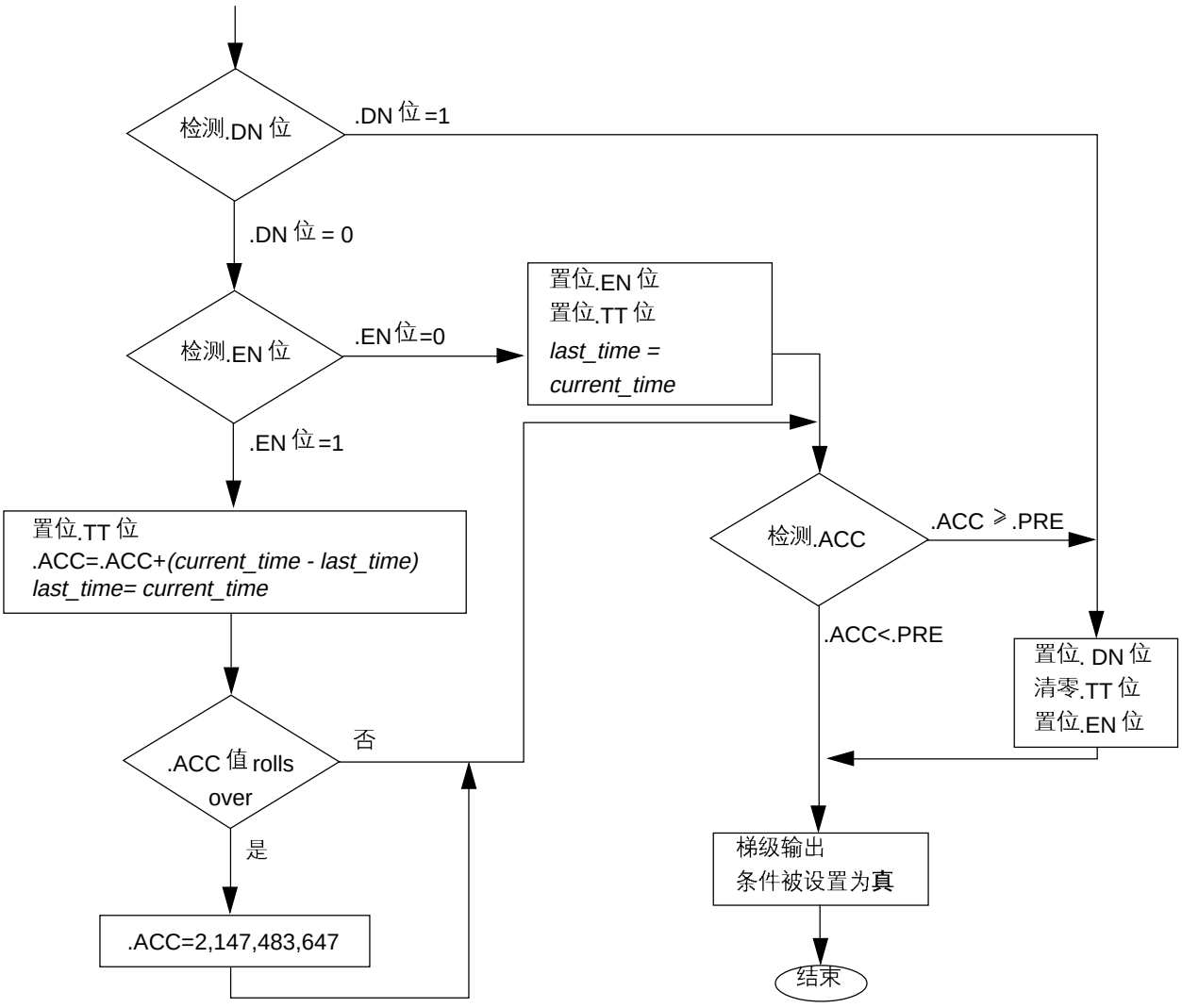
- TON 指令被禁止。
- 累加值 (.ACC) 预置值 (.PRE)

当 TON 指令被禁止时，清零累加值(.ACC)。



执行:

条件:	动作:
预扫描	清零使能位(.EN) 清零计时位(.TT) 清零完成位(.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) 清零计时位(.TT) 清零完成位(.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	

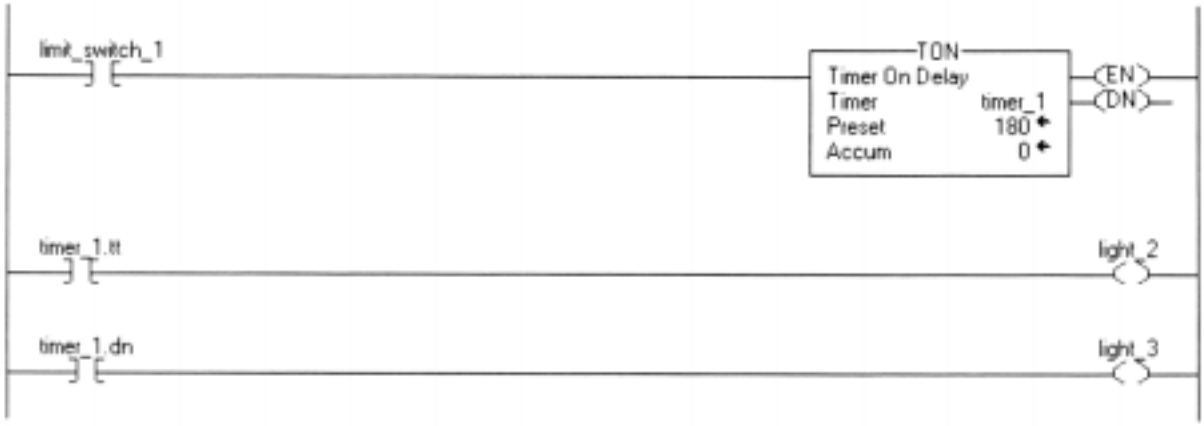


算术状态标志: 不影响

故障条件:

发生主要故障的条件:	故障类型:	故障代码:
预置值(.PRE)< 0	4	34
累加值(.ACC)< 0	4	34

TON 指令举例:



当限位开关 1 被置位时，指示灯 2 接通 180 毫秒(计时器 1 计时)。当计时器 1 的累加值.ACC 达到 180 时，指示灯 2 断开同时指示灯 3 接通。而且保持导通直到 TON 指令被禁止。如果在计时器正计时时限位开关 1 断开，则关断指示灯 2。

其他格式:

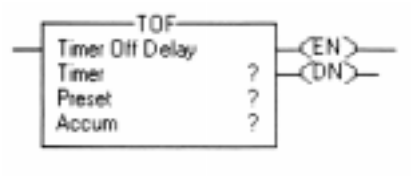
格式:	句法:
neutral 文本	`TON (timer,preset,accum);`
ASCII 文本	`TON timer preset accum`

相关指令: TOF，RTO

延时断开计时器指令 (TOF)

TOF 指令是一条输出指令。

操作数:



数据:	类型:	格式:	说明:
计时器	TIMER	标签	计时器结构
预置值	DINT	立即数	延时时间 (累积的时间值)
累加值	DINT	立即数	计时器已经计数的毫秒数，一般初始值为 0

计时器结构:

助记符:	数据类型:	说明:
.EN	BOOL	使能位—标识 TOF 指令被使能。
.TT	BOOL	计时位—标识计时操作正在进行中。
.DN	BOOL	完成位—标识累加值(.ACC) 预置值(.PRE)。
.PRE	DINT	预置值—指定在指令清零完成位(.DN)位时累加器所达到的值(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值。

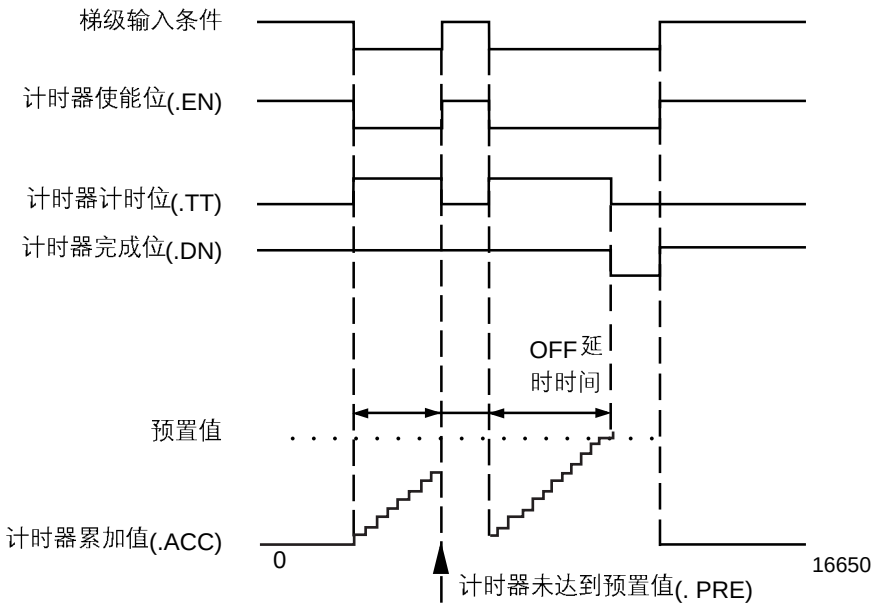
说明:

TOF是一条非保持计时器指令，当指令被使能时累计时间。计时器的时间基都是 1 毫秒。例如，对于一个 2-秒的计时器，其预置值(.PRE)应该输入 2000。

当指令被使能时，TOF计时器指令累计时间直到发生下列事件:

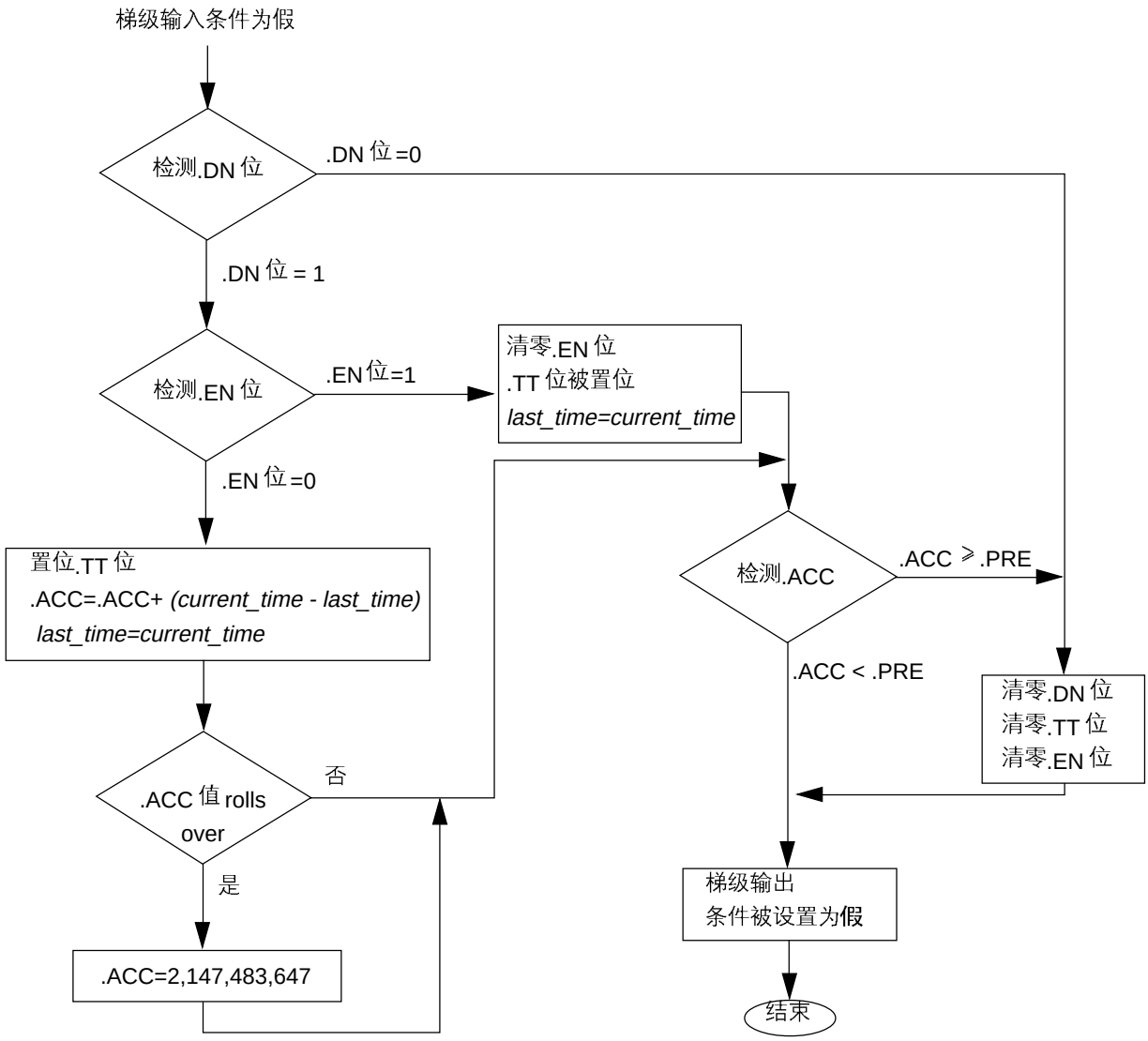
- TOF指令被禁止。
- 累加值(.ACC) 预置值(.PRE)

TOF 指令被禁止时，累加值(.ACC)被清零。



执行:

条件:	动作:
预扫描	清零使能位(.EN) 清零计时位(.TT) 清零完成位(.DN) 设置累加值(.ACC)等于预置值(.PRE)。 梯级输出条件设置为假。



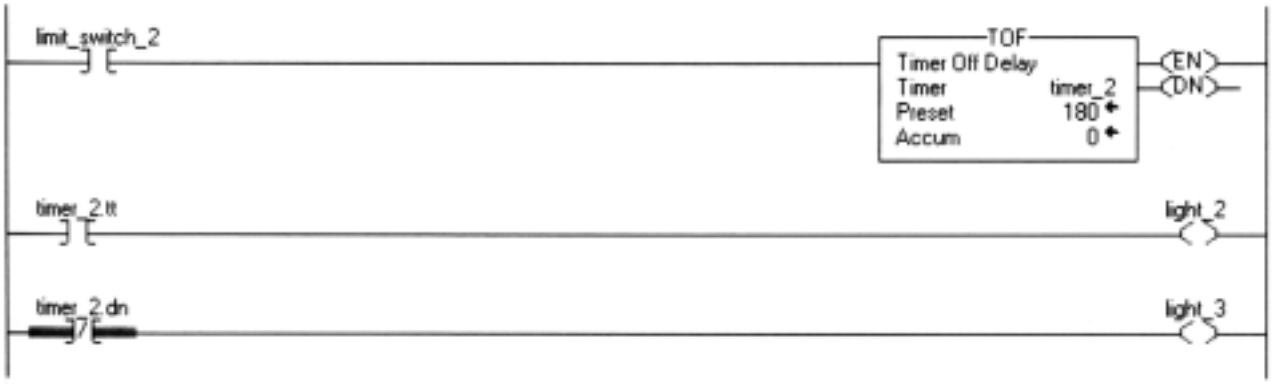
梯级输入条件为真	置位使能位(.EN)。 清零计时位(.TT)。 置位完成位(.DN)。 清零累加值值(.ACC)。 梯级输出条件被设置为真。

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:
预置值(.PRE)< 0	4	34
累加值(.ACC)< 0	4	34

TOF 指令举例:



当限位开关 2 被清零时，指示灯 2 导通 180 毫秒(计时器 2 计时)。当计时器 2 的累加值.ACC 达到 180 时，指示灯 2 断开同时接通指示灯 3。而且指示灯保持导通直到 TOF 指令被使能。如果在计时器 2 正计时时限位开关 2 被置位，则关断指示灯 2。

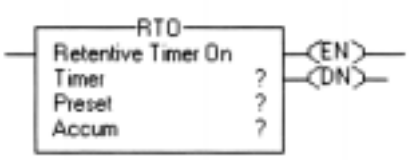
其他格式:

格式:	句法:
neutral 文本	`TOF (timer,preset,accum);`
ASCII 文本	`TOF timer preset accum`

相关指令: TON，RTO

保持型延时导通计时器指令 (RTO)

RTO 指令是一条输出指令。



操作数:

操作数:	数据类型:	格式:	说明:
计时器	TIMER	标签	计时器结构。
预置值	DINT	立即数	延时时间 (累积的时间值)。
累加值	DINT	立即数	计时器已经计数的毫秒数, 初始值一般为 0。

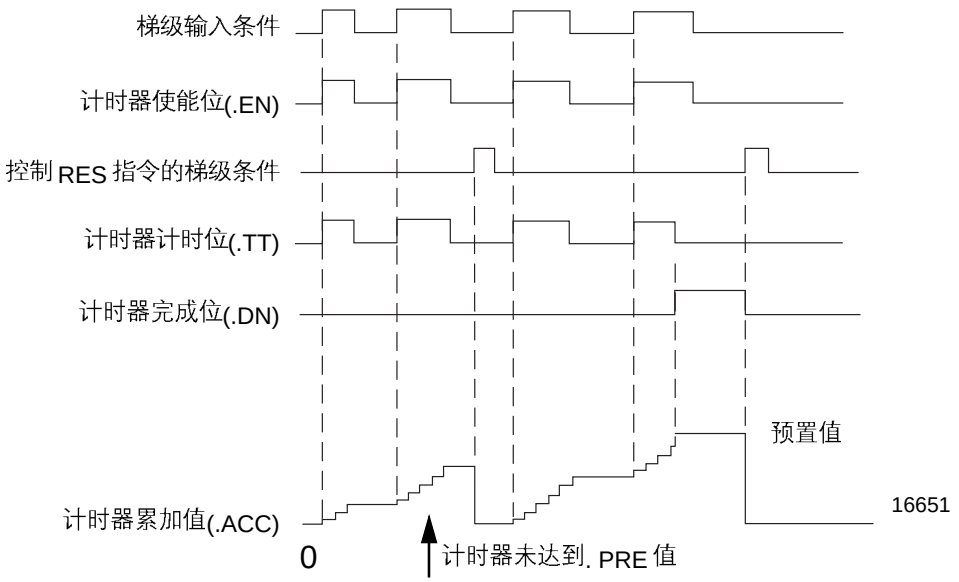
计时器结构:

助记符:	数据类型:	说明:
.EN	BOOL	使能位—标识 RTO 指令被使能。
.TT	BOOL	计时位—标识计时操作正在进行中。
.DN	BOOL	完成位—标识累加值(.ACC) ≥ 预置值 (.PRE)。
.PRE	DINT	预置值—指定在指令清零完成位(.DN)时累加器所达到的值(以 1 毫秒为单位)
.ACC	DINT	累加值—表示从 RTO 指令被使能时已经经过的毫秒值。

说明:

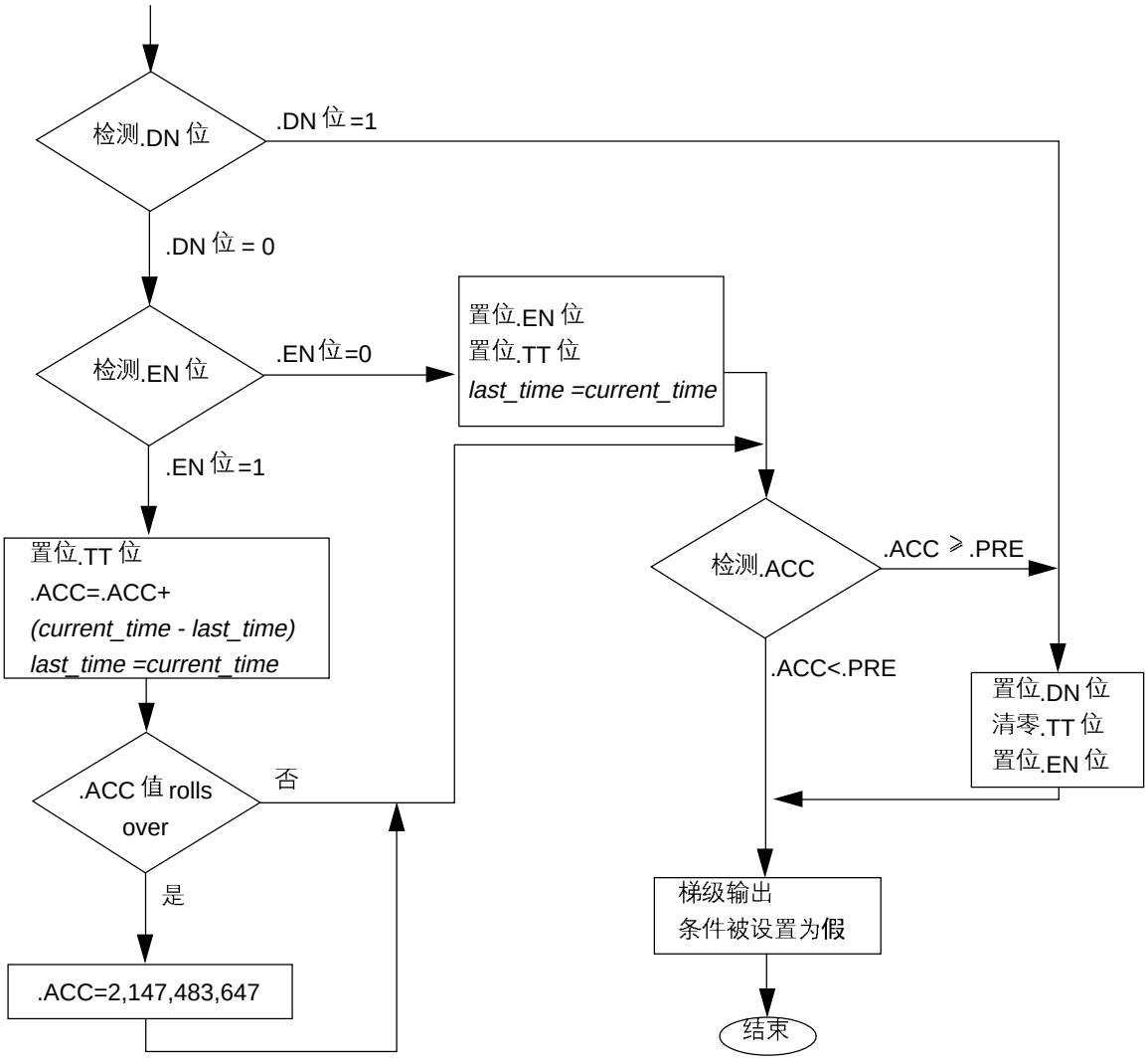
RTO是一条保持型的计时器指令, 当指令被使能时累计时间。计时器的时间基总是 1 毫秒。例如，对于一个 2- 秒的计时器，其预置值(.PRE)应该输入 2000。

当指令被使能时, RTO计时器指令累计时间直到它被禁止。如果 RTO 指令被禁止，则将保持累加值(.ACC)。用户必须清零累加值(.ACC)，一般用一条引用相同 TIMER 结构的 RES 指令清零计时器的累加值。



执行:

条件:	动作:
预扫描	清零使能位(.EN)。 清零计时位(.TT)。 清零完成位(.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)。 清零计时位(.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	

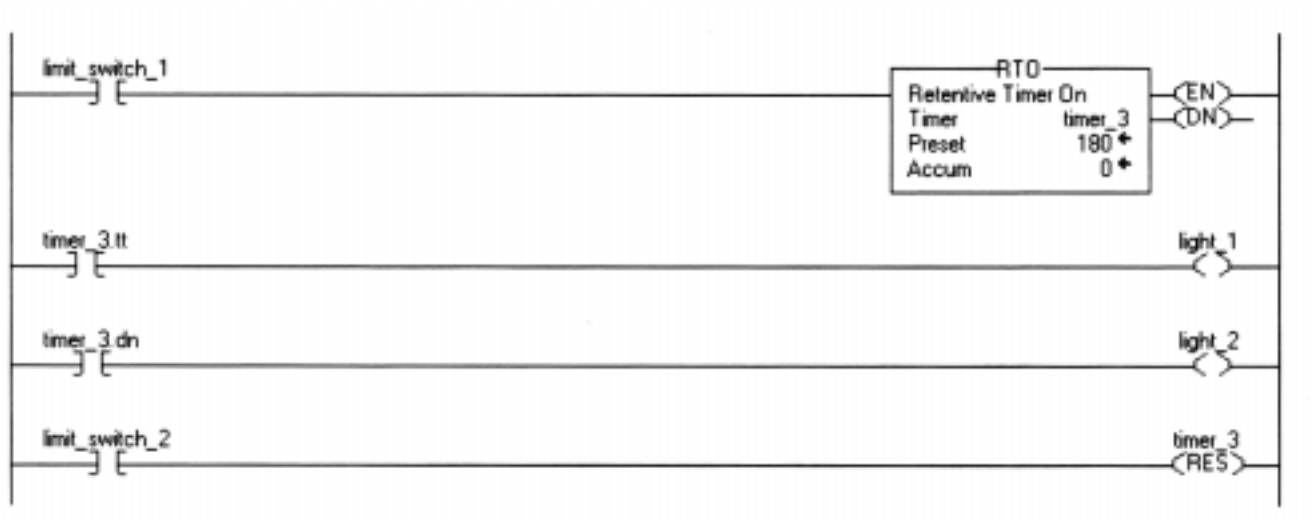


算术状态标志: 不影响

故障条件:

发生主要故障的条件:	故障类型:	故障代码:
预置值(.PRE)< 0	4	34
累加值(.ACC)<0	4	34

RTO 指令举例:



当限位开关 1 被置位时，指示灯 1 接通 180 毫秒(计时器 2 计时)。当计时器 3 的累加值.ACC 达到 180 时，指示灯 1 断开同时指示灯 2 接通。而且指示灯 2 保持导通直到计时器 3 被复位。如果在计时器 3 正计时时限位开关 2 被清零，则指示灯 1 保持导通。当限位开关 2 被置位时，RES 指令复位计时器 3(清零状态位和.ACC 值)。

其他格式:

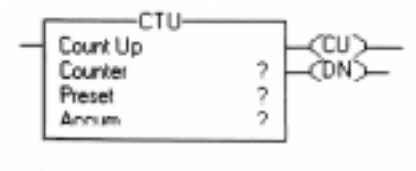
格式:	句法:
neutral 文本	`RTO (timer,preset,accum);`
ASCII 文本	`RTO timer preset accum`

相关指令: TON，TOF，RES

加计数指令 (CTU)

CTU 指令是一条输出指令。

操作数:



操作数:	数据类型:	格式:	说明:
计数器	COUNTER	标签	计数器结构。
预置值	DINT	立即数	计数的次数。
累加值	DINT	立即数	计数器已经计数的次数，初始值一般为 0

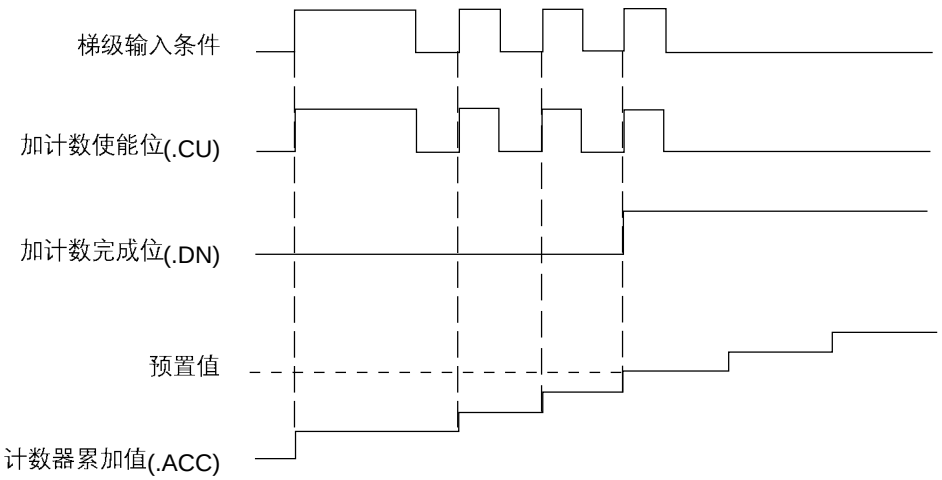
计时器结构:

助记符:	数据类型:	说明:
.CU	BOOL	加计数使能位—标识 CTU 指令被使能。
.DN	BOOL	完成位—标识累加值(.ACC) ≥ 预置值(.PRE)。
.OV	BOOL	溢出位—标识计数器超过上限值 2,147,483,647。然后计数器返回到 -2,147,483,648 并再开始加计数。
.PRE	DINT	预置值—指定在指令置位完成位(.DN)之前累加值所达到的值。
.ACC	DINT	累加值—表示指令已经计数的梯级转换的次数。

说明:

CTU 指令向上计数。

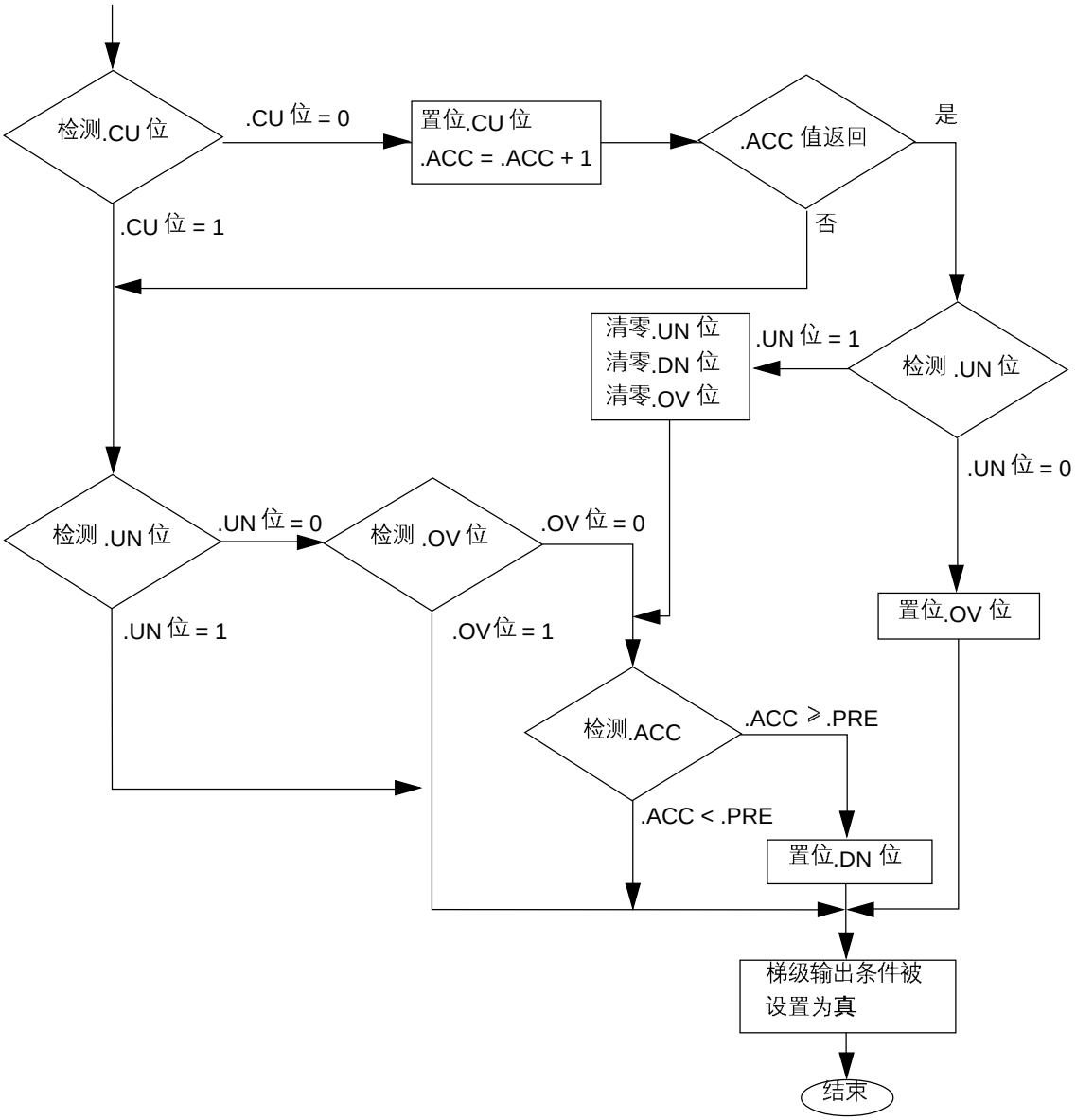
如果指令被使能时加计数使能位(.CU)是清零状态，则 CTU 指令使计数器加 1。如果指令被使能时加计数使能位(.CU)是置位状态，或指令被禁止，CTU 指令保持它的累加值(.ACC)。



即使完成位(.DN)被置位之后，累加值也继续增加。如果要清零累加值，可以用一条引用同一计数器结构的 RES 指令，或写 0 值到计数器的累加值。

执行:

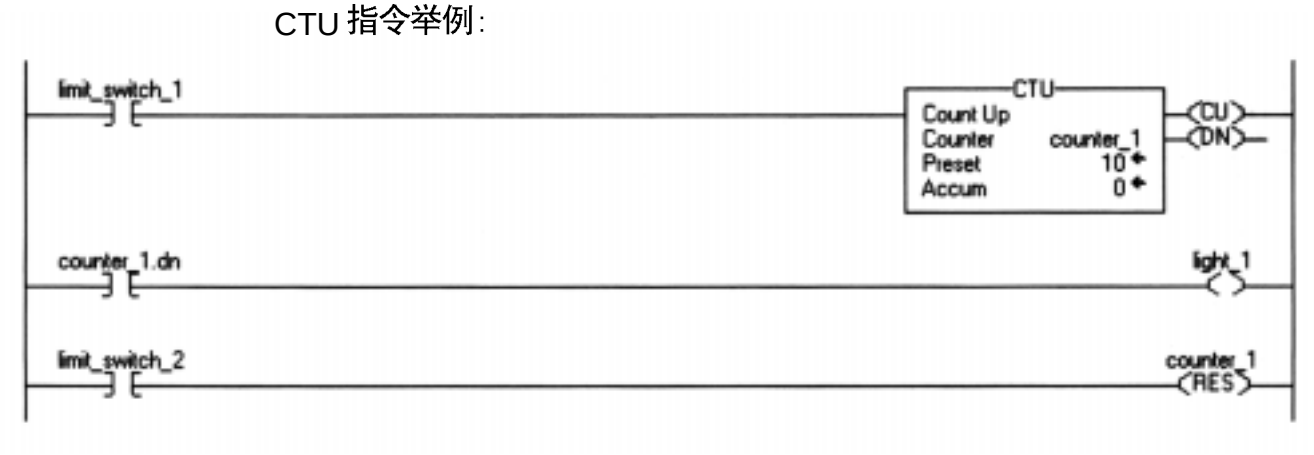
条件:	动作:
预扫描	置位加计数使能位(.CU)以防止在首次程序扫描期间计数值无效的增加。 梯级输出条件被设置为假。
梯级输入条件为假	清零置位加计数使能位(.CU)。 梯级输出条件被设置为假。
梯级输入条件为真	



算术状态标志: 不影响

故障条件: 无

CTU 指令举例:



限位开关 1 由禁止变为使能 10 次之后，完成位.DN 被置位，并且接通指示灯 1。如果限位开关 1 继续由禁止变为使能，则计数器 1 继续增加它的计数值，且完成位.DN 保持置位状态。当限位开关 2 被使能时，RES 指令复位计数器 1(清零状态位和.ACC 值)并且关断指示灯 1 。

其他格式:

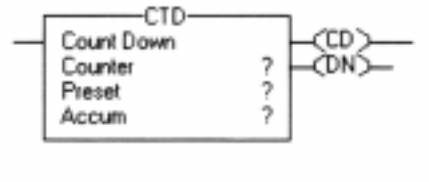
格式:	句法:
neutral 文本	`CTU(counter、preset、accum);`
ASCII 文本	`CTU counter preset accum`

相关指令: CTD，RES

减计数指令 (CTD)

CTD 指令是一条输出指令。

操作数:



操作数:	数据类型:	格式:	说明:
计数器	COUNTER	标签	计数器结构。
预置值	DINT	立即数	计数次数。
累加值	DINT	立即数	计数器已经计数的次数，一般初始值为 0。

计数器结构:

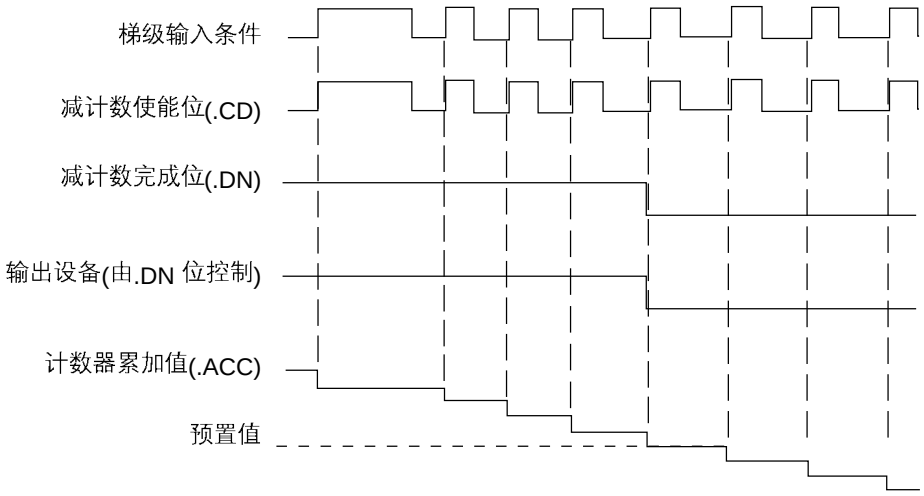
助记符:	数据类型:	说明:
.CD	BOOL	减计数使能位—标识 CTD 指令被使能。
.DN	BOOL	完成位—标识累加值(.ACC) (预置值(.PRE)。
.UN	BOOL	下溢出位—标识计数器超过下限值 -2,147,483,648。然后计数器返回到 2,147,483,647，再开始减计数。
.PRE	DINT	预置值—指定在指令置位完成位 (.DN)之前累加值所达到的值。
.ACC	DINT	累加值—表示指令已经计数的梯级转换的次数。

说明:

CTD 指令向下计数。

CTD指令比较有代表性的应用是与引用相同计数器结构的 CTU 指令一起使用。

如果指令被使能时减计数位 .CD 清零，则CTD 指令使计数值减 1。如果指令使能时减计数位 .CU 置位，或指令被禁止，则 CTD 指令保持它的 .ACC 值。

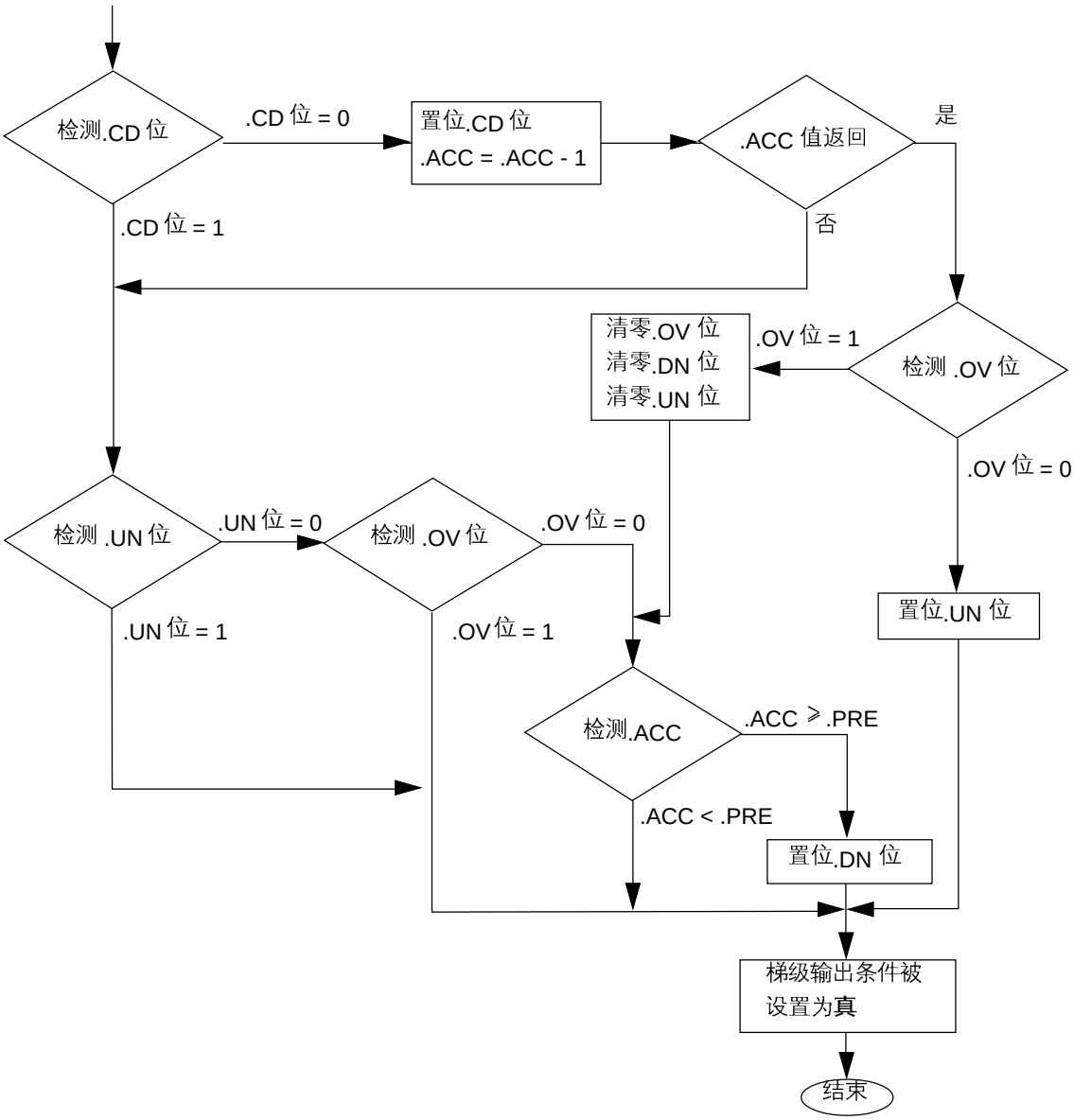


即使完成位(.DN)置位之后，累加值也继续减少。如果要清零累加值，可以用一条引用同一计数器结构的 RES 指令，或写 0 值到计数器的累加值。

执行:

条件:	动作:
预扫描	置位减计数使能位(.CD)以防止在首次扫描程序期间无效地减小计数值。梯级输出条件被设置为假。
梯级输入条件为假	清零减计数使能位(.CD)。 梯级输出条件被设置为假。

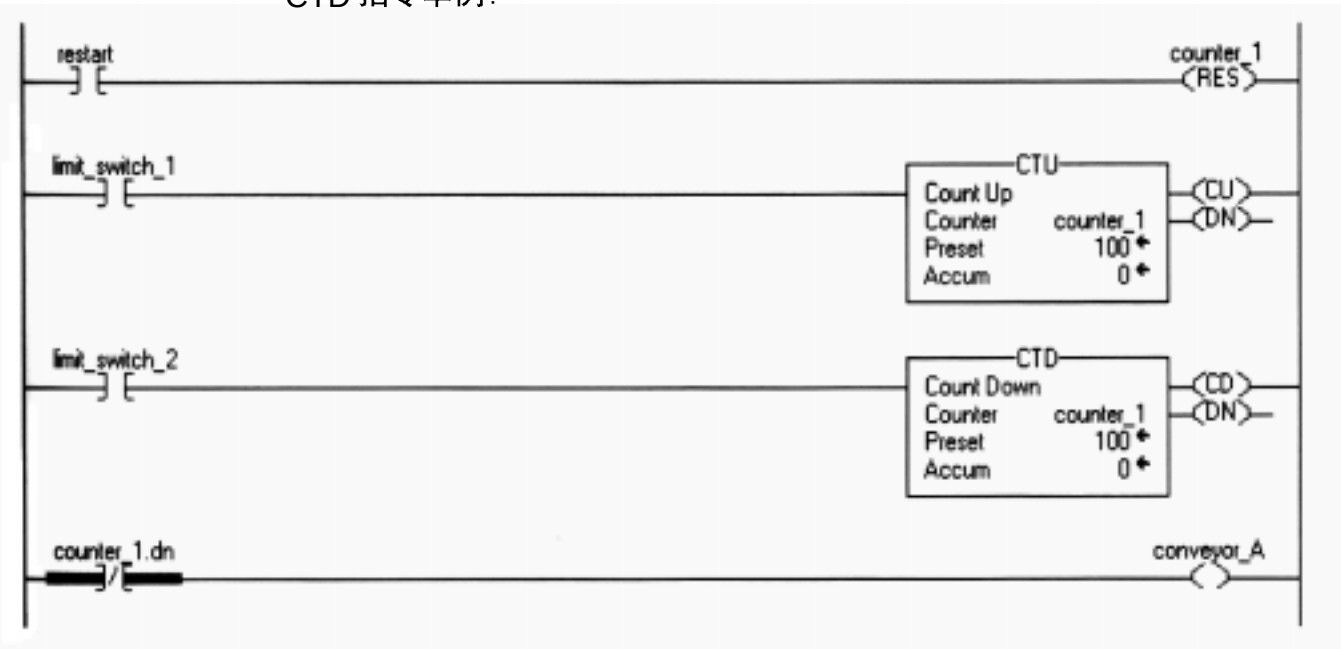
梯级输入条件为真



算术状态标志: 不影响

故障条件: 无

CTD 指令举例:



传送装置把零件带到缓存区。每进入一个零件，限位开关 1 被使能且计数器 1 的累加值加 1。每取出一个零件，限位开关 2 被使能且计数器 1 的累加值减 1。如果有 100 个零件进入缓存区(置位计数器 1 的完成位 .DN)，则关断传送装置 A，在缓存区有空间之前，不再传送零件进入缓存区。

其他格式:

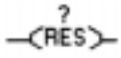
格式:	句法:
neutral 文本	`CTD (counter、preset、accum)`
ASCII 文本	`CTD counter preset accum`

相关指令: CTU，RES

复位指令 (RES)

RES 指令是一条输出指令。

操作数:



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

说明:

RES指令复位TIMER，COUNTER，或CONTROL 结构。

当 RES 指令被使能使能时清零下列元素:

当 RES 指令用于:	指令清零:
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：因为 RES 指令清零累加值(.ACC)，完成位(.DN)，和计时位(.TT)，所以不要用 RES 指令复位 TOF 计时器。

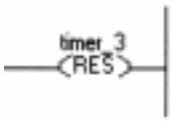
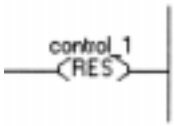
执行:

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	RES 指令复位指定的结构。 梯级输出条件被设置为真。

算术状态标志: 不影响

故障条件: 无

RES 指令举例：

举例：	说明：
	当使能时，复位*计时器 3*。
	当使能时，复位*计数器 1*。
	当使能时，复位*控制结构 1*。

其他格式：

格式：	句法：
neutral 文本	RES *(structure)*
ASCII 文本	RES *structure*

注释:

输入 / 输出指令

(MSG，GSV，SSV)

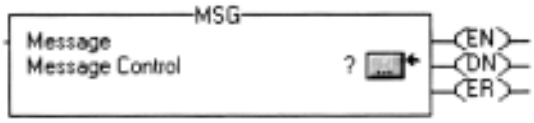
简介

输入/输出指令从控制器读数据，或写数据到控制器，或者从另一网络上的另一个模块读数据块，或写数据到该模块。

如果用户要：	使用下列指令：	参见页次：
与另一模块进行数据交换	MSG	3 - 2
获取控制器状态信息	GSV	3 - 22
设置控制器状态信息	SSV	3 - 22

通讯指令 (MSG)

MSG 指令是一条输出指令。



操作数:	类型:	格式:	说明:
信息控制	信息	标签	信息结构体

MSG 结构体:

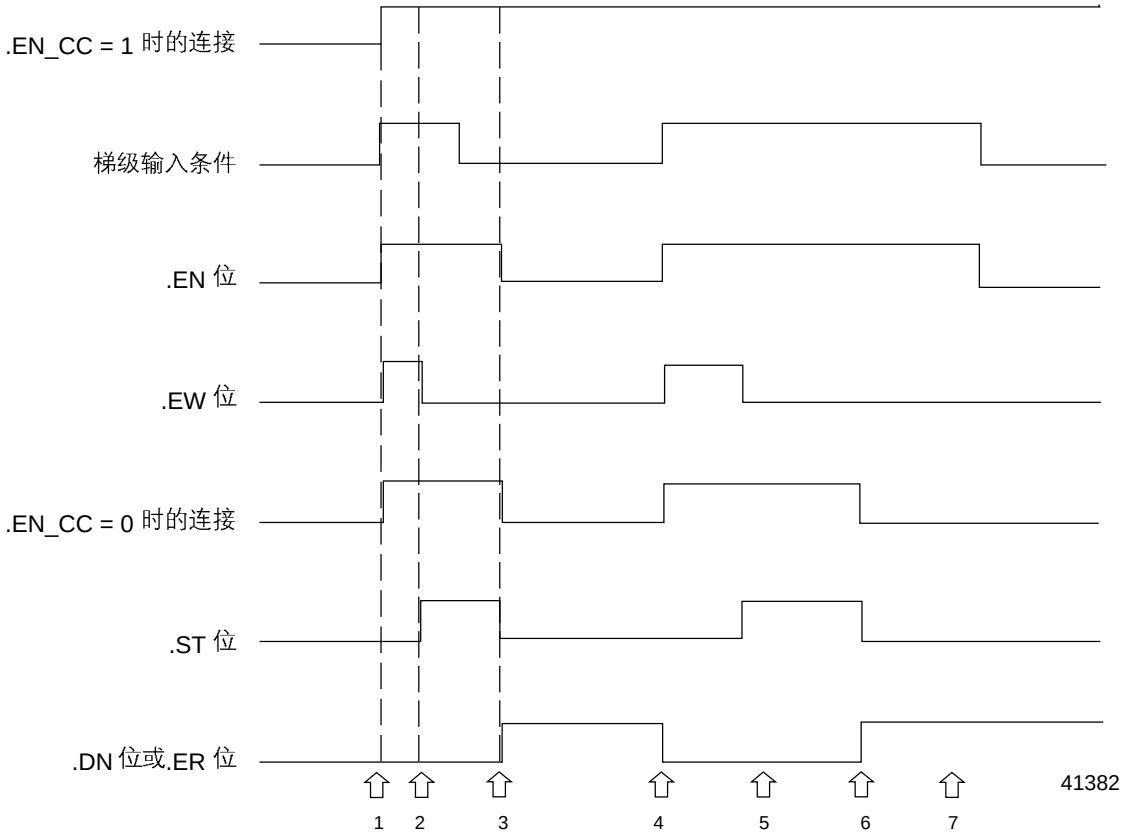
助记符:	数据类型:	说明:	
.FLAGS	INT	.FLAGS	成员在一个 16 位字内存储下列状态位。
		位:	数据类型: 说明:
		.EW	BOOL 当控制器检测到一个通讯请求已经进入队列时，置位使能等待位。当启动位(.ST)置位时，控制器复位使能等待位(.EW)。
		.ER	BOOL 当控制器检测到传送失败时，错误位被置位。下一次梯级输入条件由假变真时复位错误位(.ER)。
		.DN	BOOL 当最后一个信息数据包成功的传送之后，完成位被置位。下一次梯级输入条件由假变真时复位完成位(.DN)。
		.ST	BOOL 当控制器开始执行 MSG 指令时，起动位被置位。当完成位(.DN)或错误位(.ER)被置位时复位启动位(.ST)。
		.EN	BOOL 当梯级输入条件变为真时，使能位被置位。而且该位保持置位，直到完成位(.DN)或错误位(.ER)置位且梯级输入条件为假。
		.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 使能高速缓存位决定怎样管理 MSG 连接。如果用户希望控制器保持这种连接(例如用户要多次重复执行同一条 MSG 指令很多次)，置位.EN-CC 位。如果用户很少执行 MSG 指令而且对控制器的连接还有其它需要，则清零.EN-CC 位。(参见 3 - 3 页时序图) 即使.EN_CC 位被置位，经由串行接口的 MSG 指令连接也不被缓存。
.ERR	INT	如果错误位(.ER)被置位，则错误代码字表示 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代码。	
.REQ-LEN	INT	请求长度，用于指定通讯指令将要传送的字数。	
.DN-LEN	INT	完成的长度，用于表明实际传送的字数。	

注意：控制器处理启动位(.ST)和使能等待位(.EW)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，时间问题会使你的应用程序无效，这可能会造成设备的损坏或人身伤害。

说明: MSG 指令与网络上的另一模块进行数据块交换时，以异步的方式完成的。

MSG 指令传送数据元素。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。

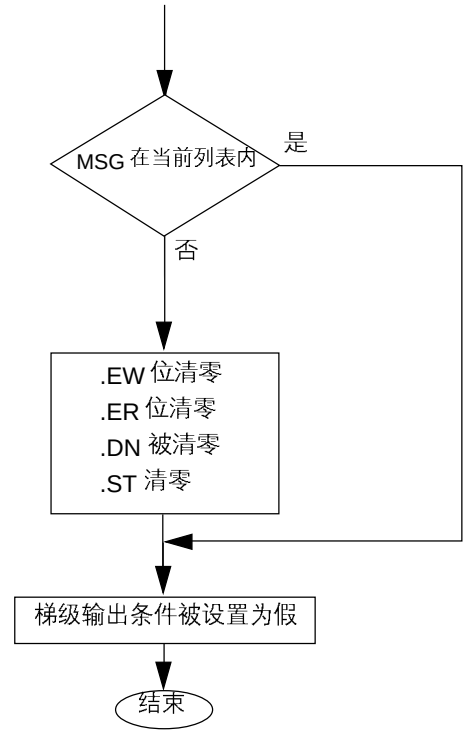
MSG 指令时序图



图中:	说明:
1	梯级输入条件为真 .EN 置位 .EW 置位 连接打开
2	发送信息 .ST 置位 .EW 被清零
3	通讯完成或出错而且梯级输入条件为假 .DN 或 .ER 置位 .ST 被清零 连接关闭 (如果 .EN_CC = 0) .EN 被清零 (因为梯级输入条件为假)

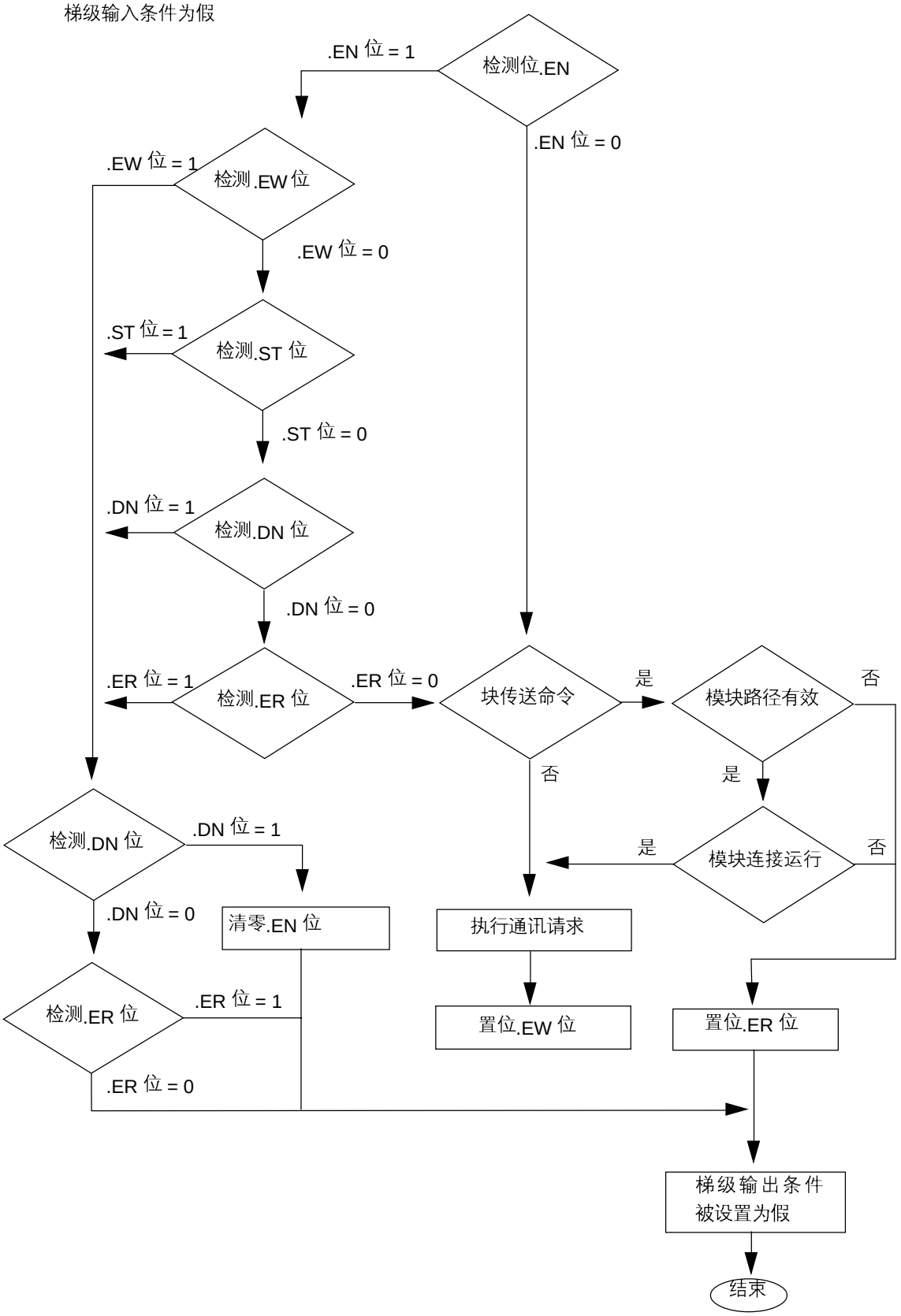
图中:	说明:
4	梯级输入条件为真并且.DN 或.ER 位先前是置位状态 .EN 置位 .EW 置位 连接打开 .DN 或 .ER 被清零
5	发送信息 .ST 置位 .EW 被清零
6	通讯完成或出错同时梯级输入条件为真 .DN 或 .ER 置位 .ST 被清零 连接被关闭 (如果 .EN_CC = 0)
7	梯级输入条件变为假而且 .DN 或 .ER 被置位 .EN 被清零

执行:

条件:	动作:
预扫描	

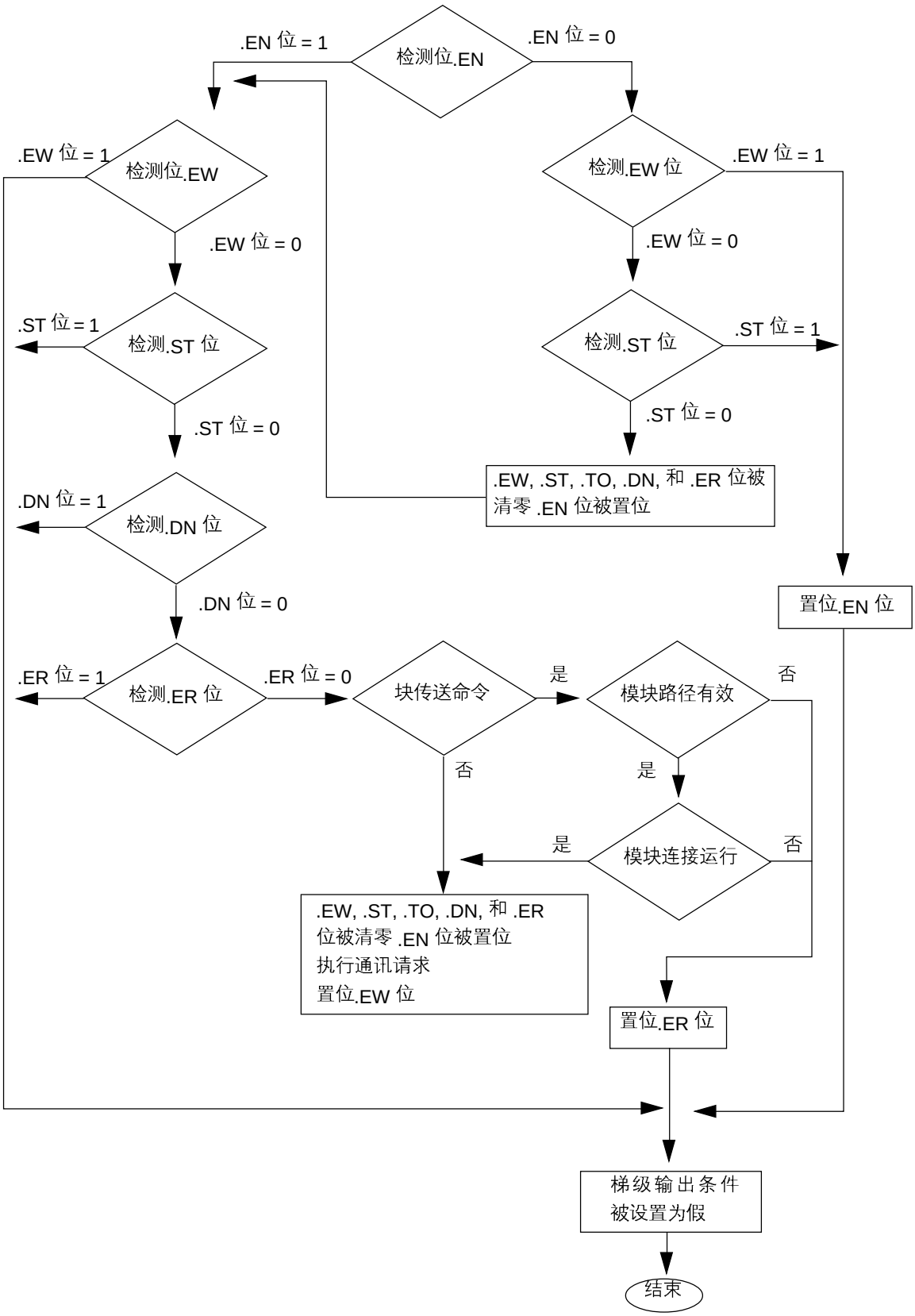
条件: **动作:**

梯级输入条件为假



条件: **动作:**

梯级输入条件为真



算术状态标志: 不影响

故障条件: 无

其它格式:

格式:	句法:
neutral 文本	`MSG (message_control);`
ASCII 文本	`MSG message_control`

MSG 错误代码

错误代码与 MSG 指令的类型有关。

ControlLogix (CIP) 错误代码

对于 ControlLogix (CIP) 错误代码，编程软件不能总是显示所有描述。

错误代码: (十六进制)	描述:	软件显示:
0001	连接失败(参见扩展错误代码)	*与描述相同*
0002	资源不足	*与描述相同*
0003	无效的数值	*与描述相同*
0004	IOI 语法错误(参见扩展错误代码)	*与描述相同*
0005	目标设备未知，某些要求未定义或元素结构未定义 (参见扩展错误代码)	*与描述相同*
0006	信息包空间不足	*与描述相同*
0007	连接丢失	*与描述相同*
0008	服务不支持	*与描述相同*
0009	数据段错误或数值属性无效	*与描述相同*
000A	属性列表错误	*与描述相同*
000B	状态已经存在	*与描述相同*
000C	目标模式冲突	*与描述相同*
000D	目标已经存在	*与描述相同*
000E	属性不可设置	*与描述相同*
000F	权限不允许	*与描述相同*
0010	设备状态冲突	*与描述相同*
0011	回答不适合	*与描述相同*
0012	访问未使用的存储器单元	*与描述相同*
0013	命令数据不适合	*与描述相同*
0014	属性不支持	*与描述相同*
0015	数据太多	*与描述相同*
001A	网桥请求太大	*与描述相同*
001B	网桥响应太大	*与描述相同*

错误代码: (十六进制)	描述:	软件显示:
001C	属性列表不足	*与描述相同*
001D	无效的属性列表	*与描述相同*
001E	嵌入服务错误	*与描述相同*
001F	与连接有关的错误(参见错误代码)	*与描述相同*
0022	收到无效的回答	*与描述相同*
0025	关键信息段错误	*与描述相同*
0026	无效 IOI 错误	*与描述相同*
0027	列表中有意外的属性	*与描述相同*
0028	DeviceNet 错误 - 无效的 ID 成员	*与描述相同*
0029	DeviceNet 错误 - 成员没有设置	*与描述相同*

ControlLogix 扩展错误代码

下面是 ControlLogix (CIP) 扩展错误代码。编程软件不显示扩展错误代码的任何文本。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):	描述
0100	连接正被使用
0103	不支持传送
0106	权限冲突
0107	未找到连接
0108	连接类型无效
0109	连接尺寸无效
0110	模块未配置
0111	EPR 不支持
0114	错误模块
0115	设备类型错误
0116	错误版本
0118	配置格式无效
011A	请求超出连接范围
0203	连接超时

扩展错误代码 (十六进制):	描述
0204	信息(未连接)超时
0205	发送参数(未连接)错误
0206	信息太大
0301	无缓冲器内存
0302	带宽无效
0303	无有效分级
0305	署名相同
0311	端口不可用
0312	链接地址无效
0315	无效信息段类型
0317	连接未预定

下表是错误代码是 **001F** 的扩展错误代码。

扩展错误代码 (十六进制):	描述:
0203	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):	描述
0000	扩展状态超出内存范围
0001	扩展状态超出要求

PLC 和 SLC 错误代码 (.ERR)

对于 PLC 和 SLC 错误代码，编程软件不显示全部描述。

错误代码(十六进制)	描述:	软件显示:
0010	来自本地处理器的非法命令或格式	设备状态冲突
0020	通讯模块不工作	未知错误
0030	远程节点丢失，断开，或关闭	未知错误
0040	处理器虽然连接但有故障(硬件)	未知错误
0050	站号错误	未知错误
0060	请求的功能无效	未知错误
0070	处理器在编程方式	未知错误
0080	处理器的兼容文件不存在	未知错误
0090	远程节点不能缓存命令	未知错误
00B0	处理器正在下载，所以不能访问	未知错误
00F0	PCCC 错误(参见扩展错误代码)	未知错误

PLC 与 SLC 扩展错误代码 (.EXERR)

对于扩展错误代码，软件不以任何文本的形式显示。下面是错误代码是 **00F0** 的扩展错误代码.

扩展错误代码 (十六进制):	描述:
0001	处理器不正确地转换地址
0002	地址不完整
0003	地址不正确
0004	非法的地址格式 - 不能找到符号
0005	非法的地址格式 - 符号包含 0 或大于设备支持的最大字符数
0006	目标处理器不存在指定的地址文件
0007	目的单元文件太小，不能满足请求的字数
0008	不能完成请求 在多组操作期间状况改变
0009	数据或文件太大，内存不够
000A	目标处理器不能把请求的信息放入信息包
000B	权限错误：拒绝访问
000C	请求的功能无效
000D	重复请求
000E	命令不能执行
000F	溢出：频率分布溢出
0010	无访问权

扩展错误代码 (十六进制):	描述:
0011	请求的数据类型与可用的数据类型不匹配
0012	不正确的命令参数
0013	引用地址存在被删除的区域
0014	因未知原因，命令执行失败 PLC-3 频率分布溢出
0015	数据转换错误
0016	扫描器不能与1771 框架适配器通讯
0017	适配器不能与指定的模块通讯
0018	1771 模块的响应不正确
0019	标号重复
001A	文件所有者正使用 - 文件正被使用
001B	程序所有者正使用 - 有人正在下载或正在在线编辑
001C	磁盘文件写保护或其他原因不能使用(只离线)
001D	另一个应用程序正使用磁盘文件不执行更新(只离线)

块传送错误代码

下面是块传送特有的错误代码。

错误代码 (十六进制):	描述:	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块传送响应	未知错误
00D1	来自读响应的校验和与数据串的校验和不匹配	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	块传送回答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回答，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	指定的逻辑槽不能用于给定的机架尺寸	未知错误
00EC	有一个块传送请求正在进行中,但是在另一个请求可以开始之前必须得到一个响应	未知错误
00ED	块传送请求的尺寸与有效的块传送请求的尺寸不一致	未知错误
00EE	块传送请求的类型与 BT_READ 或 BT_WRITE 要求的不一致	未知错误
00EF	扫描器不能在块传送表内找到一个有效的槽适应块传送请求	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	对于请求的机架或槽没有配置通道	未知错误
00F6	没有通讯通道被配置用于远程 I/O	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 - 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和错误	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	块传送的尺寸加上块传送数据表的索引尺寸大于块传送数据表文件的尺寸	未知错误

Logix5550 错误代码

下面是 Logix5550 错误代码。

错误代码 (十六进制):	描述:	软件显示:
00D0	映象列举未定义	未知错误
00D1	模块没有在运行状态	未知错误
00FB	通讯口不支持	未知错误
00FC	信息数据类型不支持	未知错误
00FD	信息未初始化	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

Logix5550 扩展错误代码

这些是 Logix5550 扩展错误代码。对于扩展错误代码，软件不以任何文本形式显示。

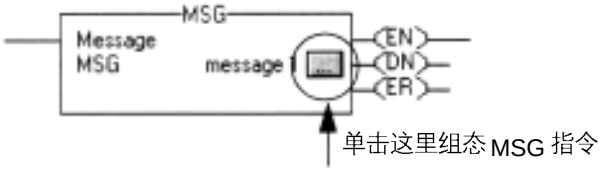
下面是错误代码 **00FF** 的扩展错误代码。

扩展错误代码 (十六进制):	描述:
2001	过多的 IOI
2002	Bad 参数值
2018	信号拒绝接收
201B	尺寸太小
201C	无效的尺寸
2100	特权失效
2101	钥匙开关在无效的位置
2102	密码无效
2103	没发出密码
2104	地址超出范围
2105	地址和数量超出范围
2106	数据正被使用

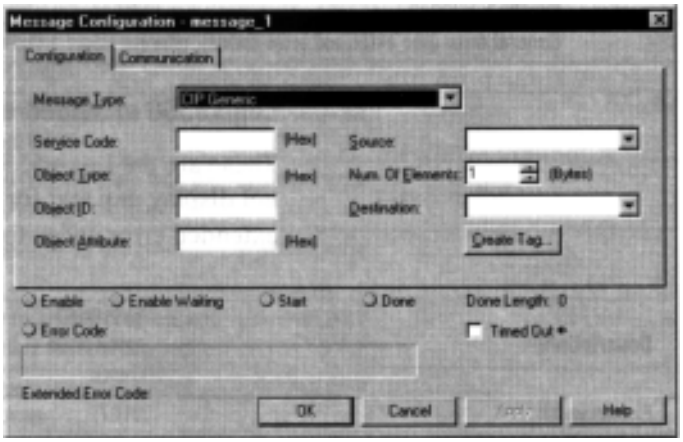
扩展错误代码 (十六进制):	描述:
2107	信息类型无效或不支持
2108	控制器正在下载或上载模式
2109	试图改变数组维数
210A	符号名称无效
210B	符号不存在
210E	搜索失败
210F	没起动任务
2110	不能写入
2111	不能读出
2112	共享例程未编辑
2113	控制器在故障模式
2114	运行模式被禁止

选择通讯类型 (组态列表Tab)

在用户输入MSG 指令而且指定了MESSAGE结构体之后，用编程软件的组态列表(tab)指定通讯的详细信息。



组态的详细信息与用户选择的通讯类型有关。



如果目标设备是:	选择下列信息类型之一:	参见页次:
Controllogix 设备或 1756I/O 模块	CIP 数据表读 CIP 数据表写 通用的 CIP	3-13
PLC-5 处理器	PLC5 类型读 PLC5 类型写 PLC5 字范围读 PLC5 字范围写	3-15
SLC 控制器	SLC 类型读 SLC 类型写	3-16
在通用远程 I/O 网络上的块传送模块	块传送读 块传送写	3-16
PLC-3 处理器	PLC3 类型读 PLC3 类型写 PLC3 字范围读 PLC3 字范围写	3-17
PLC-2 处理器	PLC2 无保护读 PLC2 无保护写	3-18

用户必须明确规定下列信息:

在下列区域:	详述:
源元素 / 标签	如果用户选择读通讯类型,则源元素是用户要读的目标设备的内的数据地址。用目标设备的地址句法。 如果用户选择写通讯类型,则源标签是在Logix5550控制器内要发送到目标设备的数据标签。
元素的数量	用户读 / 写元素的数量是由使用的数据类型决定的。一个元素的大小与其相关的数据“大块”有关。例如,标签 timer1 是由一个计时器控制结构体组成的元素。
目的元素 / 标签	如果用户选择读通讯类型,则目的标签是 Logix5550 控制器内存储数据(用户从目标设备读到的)的标签。 如果用户选择写通讯类型,则目标元素是目标设备内用户要写入数据的位置地址 。

如果用户在源或目的指定一个Logix5550 数组标签,则只需指定数组名称。不要包括括号或位置下标。

指定 CIP 通讯

CIP 通讯类型是设计与其它ControlLogix 设备交换数据的 。例如,从一个 Logix5550 控制器发送信息到另一个 Logix5550 控制器。

选择下列命令:	如果要完成:
CIP 数据表读	从另一个控制器读数据。 源和目的单元的数据类型必须匹配。
CIP 数据表写	写数据到另一个控制器。 源和目的单元的数据类型必须匹配。
通用 CIP	组态一自定义客户信息以发送组态数据到一个 I/O 模块。

用通用 CIP 信息复位 I/O 模块

用通用 CIP 信息类型指定下列信息以创建一个客户通讯.

如果用户要:	在此区域:	输入:
在离散量输出模块执行脉冲测试。	服务代码	4c
	对象 Type	1e
	对象 ID	1
	对象属性	*空白*
	源	*tag_name* 数据类型是 INT [5] 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(几百微妙，一般是 20) *tag_name[3]* 对于 ControlLogix I/O 是过零交叉延时(几百微妙，一般是 40) *tag_name[4]* 校验延时
	元素数量	10
	目的	*空白*
复位离散量输出模块上的电子保险丝	服务器代码	4d
	对象类型	1e
	对象属性	*空白*
	对象 ID	1
	源操作数	具有 DINT 类型的标签名称。 该标签代表被复位保险丝处的屏蔽位。
	元素数量	4
	目的操作数	*空白*
复位离散量 I/O 模块上的锁存诊断	服务器代码	4b
	对象类型	1e
	对象属性	*空白*
	对象 ID	1
	源操作数	具有 DINT 类型的标签名称。 该标签代表被复位诊断处的屏蔽位。
	元素数量	4
	目的操作数	*空白*

如果用户要:	在此区域:	输入:
复位模拟量模块上的锁存状态	服务器代码	4b
	对象类型	a
	对象属性	输入用户想要的属性代码
	对象 ID	0
	源操作数	*空白*
	元素数量	0
	目的操作数	*空白*

详述 PLC-5 信息

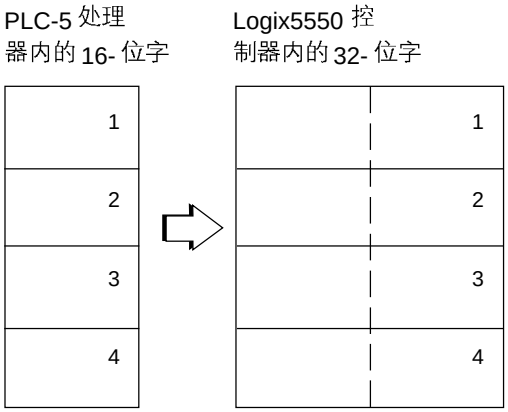
PLC-5 信息类型是为 PLC-5 处理器设计的。

选择下列命令:	如果用户要:
PLC5 定型读	读取整数或 REAL 类型数据。 对于整数，此命令从 PLC-5 处理器读取 16- 位整数 (文件类型 S, B, 和 N)，并且存储这些整数于 Logix5550 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-5 处理器读取浮点数 (F 文件类型) 并存入 Logix5550 控制器的 REAL 数据类型标签内。
PLC5 定型写	写入整数或 REAL 类型数据。 此命令写 SINT 或 INT 数据到 PLC-5 处理器 (文件类型 S, B, 和 N) 并且保持数据的完整性。只要不超出 INT 数据类型的范围 (-32,768 ⩾ 数据 ⩽ 32,767)，用户就可以写入 DINT 数据。 此命令也从 Logix5550 控制器写 REAL 类型数据到 PLC-5 的浮点数文件 (F 文件类型)。
PLC5 字域读	读 PLC-5 存储器内 16- 位字的相邻区 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单元标签指定的地址开始的区域内。
PLC5 字域写	从 Logix5550 存储器写 16- 位字相邻域到 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的地址开始的 PLC-5 处理器内。

读或写定型命令也适用于 SLC 5/03 处理器 (OS303 和更高), SLC 5/04 处理器 (OS402 和更高), 以及 SLC 5/05 处理器。

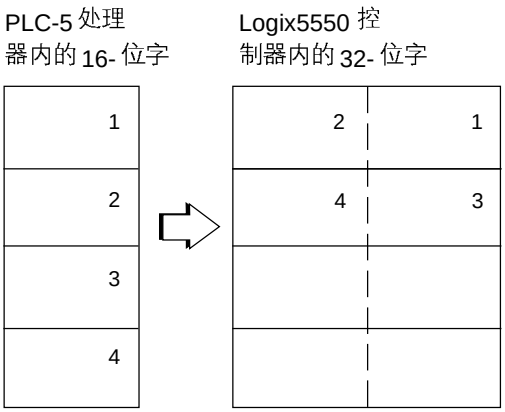
下图展示类型命令和字域命令的区别。此例从 PLC-5 处理器到一个 Logix5550 控制器使用读命令。

定型读命令



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。

详述 SLC 信息

SLC 信息类型是为 SLC 和 MicroLogix1000 控制器而设计的。

选择下列命令:	如果用户要:
SLC 定型读	读 INT 或 DINT 数据。
SLC 定型写	写 INT 或 DINT 类型数据。

Logix5550 标签类型必须与 SLC 数据类型匹配。用户可以只传送 DINT 数据 (对应 SLC 位数据类型) 或 INT 数据(对应 SLC 整数数据类型)。

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。

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

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，除 MESSAGE, AXIS, 和 MODULE 结构体以外。

用户必须指定发送或接收的 16- 位整数 (INT) 的数量. 可以指定整数的范围是从 0 到 64。如果对 BTR 信息指定 0，则块传送模块自己确定发送多少 16- 位整数。如果对 BTW 信息指定 0，则控制器发送 64 个整数。

接收块传送的 I/O 模块必须在控制器组织内定义。当用户选择块传送信息类型时，不要在配置列表 (tab) 选择通讯方法。CIP 和 DH+ 选项是灰的.

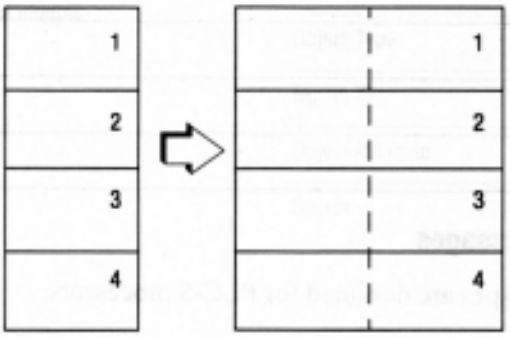
详述 PLC-3 信息

PLC-3 信息类型是专为 PLC-3 处理器设计的。

选择下列命令:	如果用户要:
PLC3 定型读	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 Logix5550 控制器的SINT, INT, 或DINT 数据数组内。而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 Logix5550 控制器的 REAL 数据类型标签内。
PLC3 定型写	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。 此命令也从 Logix5550 控制器到一个 PLC-3 浮点数文件写 REAL 类型数据。
PLC3 字域读	读 PLC-3 存储器内 16- 位字的相邻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的地址开始的区域内。
PLC3 字域写	从Logix5550 存储器写16- 位字相邻域到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从目的元素指定的地址开始的 PLC-3 处理器内。

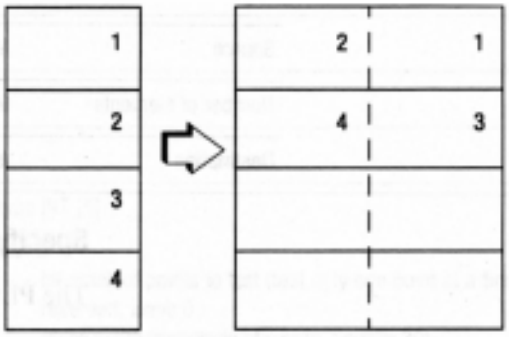
下图展示定型命令和字域命令的区别。本例用从PLC-3处理器到 Logix5550 控制器的读命令

定型读命令



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。

详述 PLC-2 信息

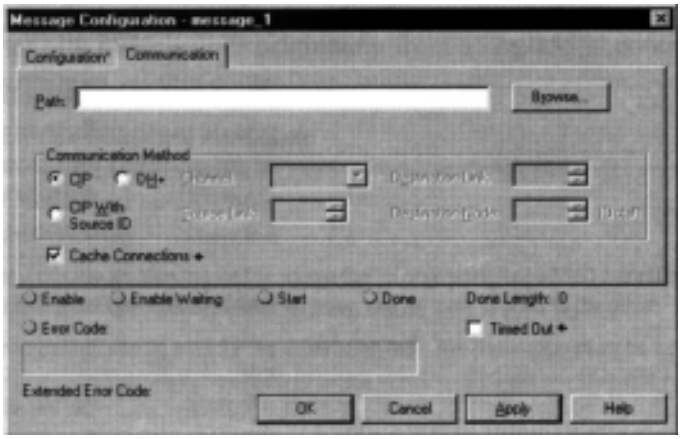
PLC-2 信息类型是为 PLC-2 处理器设计的。

选择下列命令:	如果用户要:
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 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 Logix5550 相应的存储传送数据的标签 (典型的是一个 INT 数组)。

指定详细通讯内容 (通讯列表 Tab)

当配置 MSG 指令时，用户可以在通讯列表(tab)处指定下列详细信息。



在此区域:	详述:
路径	输入到目标设备的连接路径 如果用户的目标设备是控制器的一个模块,则可以在编程软件的下拉菜单选择模块
通讯方法	如果信息是通过 ControlLogix 的背板传递的则选择 **CIP**，以太网，或 ControlNet 网 如果信息是通过DH+ 链路传递的自然选择 **DH+**。在这种情况下，路径指示到 DH/RIO 模块的路径。指定:
Channel:	连接到 DH+ 网络上的 1756-DHRIO 模块的通道 A 或 B
Source Link:	本地 DH+ 链的连接 ID
Destination Link:	目标设备所在的远程 DH+ 链路的链接 ID
Destination Node:	目标设备的站地址 如果只有一个 DH+ 链而且用户没有用 Gateway 软件组态用于远程链的 DH/RIO 模块，则源链接和目的链接都指定为 0。如果用户指定块传送信息，则 CIP 和 DH+ 选项是灰的
缓存连接	在复选框选择此项使能缓存连接。清除该复选框禁止缓存连接 如果一条 MSG 指令重复执行，应该保持连接在打开状态 (使能缓存连接) 以优化执行时间。如果每次打开连接都执行指令会增加执行时间 如果一条 MSG 指令很少执行则可以在完成时关闭它的连接(禁止缓存连接)以使连接可以被其它通讯利用。

指定连接路径

如果用户要从串行端口发出信息，在目标设备站地址之后的连接路径处输入 2。

如果为其它网络组合构造一个连接路径，可以输入一个或多个通向目标设备的路径段。每个路径段可以从一个模块到另一个模块：通过背板控制总线或通过 DH+, ControlNet, 或以太网。

每个路径段包含两个成员：

x,y

其中：

成员：	表示：	
x	用户用于从模块退出的端口类型号：	
	0	从 KT 卡的 DH+ 端口。
	1	从任意 1756 模块的背板。
	2	从 1756-L1 控制器的 DF1 端口。
	2	从 KTC 卡或 1756-CNB 模块的 ControlNet 端口。
	2	从 1756-ENET 模块的以太网端口。
	2	从 1756-DHRIO 模块通道 A 的 DH+ 端口。
	3	从 1756-DHRIO 模块通道 B 的 DH+ 端口。
,	分隔路径段的起始点和结束点。	
y	将要进入的模块的地址。	
	对于背板控制总线	**地址含义：**
	ControlBus backplane	槽号
	DF1 网络	站地址(0-254)
	ControlNet 网络	节点号(1-99 十进制)
	DH+ 网络	节点号(1-77 八进制)
	以太网	IP 地址(由句点分开的四个十进制数字)

如果用户有多个路径段，用逗号(,)分开每个路径段。

MSG 编程举例

下面举例展示对于不同控制器组合的源、目的标签和元素。

MSG 指令由 Logix5550 控制器启动并写到另一个控制器内。

通讯路径:	源和目的操作数举例:	
Logix5550 → Logix5550	源标签	*array_1*
	目的标签	*array_2*
	对于源标签可以使用别名标签 (在起动通讯的 Logix5550 控制器)。如果用户想从一数组内的偏移值开始，用一个别名指向偏移值。对于目的标签**不能**使用别名，目的操作数必须是基本标签。	
Logix5550 → PLC-5 Logix5550 → SLC	源标签	*array_1*
	目的元素	*N7: 10*
	对于源标签可以使用别名标签 (在起动通讯的 Logix5550 控制器)。如果用户想从一数组内的偏移值开始，用一个别名指向偏移值。	
Logix5550 → PLC-2	源标签	*array_1*
	目的元素	*010*

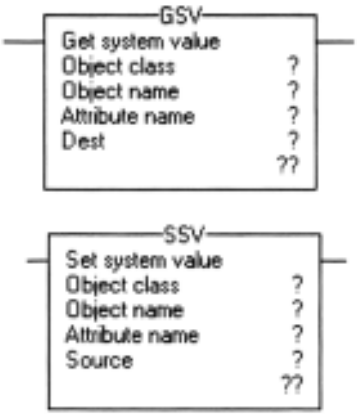
MSG 指令由 Logix5550 控制器启动并从另一个控制器读数据。

通讯路径:	源和目的操作数举例:	
Logix5550 → Logix5550	源标签	*array_1*
	目的标签	*array_2*
	对于源标签**不能**使用别名，源操作数必须是基本标签。对于目的标签可以使用别名标签 (在起动通讯的 Logix5550 控制器)。如果用户想从一数组内的偏移值开始，用一个别名指向偏移值。	
Logix5550 → PLC-5 Logix5550 → SLC	源元素	*N7: 10*
	目的标签	*array_1*
	对于目的标签可以使用别名标签 (在起动通讯的 Logix5550 控制器)。如果用户想从一数组内的偏移值开始，用一个别名指向偏移值。	
Logix5550 → PLC-2	源元素	*010*
	目的标签	*array_1*

获取系统数值 (GSV)和设置系统数值 (SSV)指令

GSV 指令是一条输出指令。

SSV 指令是一条输出指令。



操作数:

操作数:	类型:	格式:	说明:
GSV 指令			
对象类别		名称	对象类别名称
对象名称		名称	当对象要求名称时,指定的名称。
属性名称		名称	对象的属性， 数据类型与用户选择的属性有关。
目的	SINT INT DINT REAL	标签	用于属性数据的目的单元。
SSV 指令			
对象类别		name	对象类别名称
对象名称		name	当对象要求名称时,指定的名称。
属性名称		name	对象的属性
源	SINT INT DINT REAL	标签	包含要复制到属性的数据标签。

说明:

GSV/SSV 指令获取或设置存储在对象内的控制器系统数据。控制器存储系统数据于对象内。与 PLC-5 处理器一样，没有状态文件。

当指令被使能时，GSV 指令读取指定信息并把信息存放在目的单元内。当指令被使能时，SSV 指令用来自源操作数的数据设置指定的属性。

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，

用户可以从所有有效属性获取数值。对于SSV 指令，只有编程软件显示的属性允许设置(SSV)。

注意：使用 GSV/SSV 指令时要注意，对象的改变可能会导致不可预料的控制器操作或人身伤害。

如果源或目的单元的尺寸太小,则指令不执行并且记录一次次要错误。下节的 GSV/SSV 对象，定义每个对象的属性和与之相关的数据类型。例如，程序对象的主要错误记录属性要求一个 DINT[11]数据类型。

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	获取或设置指定的数值。 梯级输出条件被设置为真。

算术状态标志: 不影响

故障条件:

次要错误发生的条件:	故障类型:	故障代码:
无效的对象地址	4	5
指定了一个不支持 GSV/SSV 指令的对象。	4	6
无效属性。	4	6
没有为 SSV 指令提供足够的信息。	4	6
GSV 指令的目的单元太小不能保存请求的数据。	4	7

其它格式:

格式:	句法:
neutral 文本	GSV *(object_class,object_name,attribute_name,destination);* SSV *(object_class,object_name,attribute_name,destination);*
ASCII 文本	GSV *object_class object name attribute_name destination* SSV *object_class object name attribute_name destination*

GSV/SSV 对象

当用户输入一条 GSV/SSV 指令时，指定要访问的对象及其属性。有些情况下,对同一对象类型会有多种应用情况，所以用户可能也必须指定对象名称。例如，在用户的应用程序中可能有几个任务。用户可以通过任务名称访问每个任务自己的 TASK 对象 。

注意：对于 GSV 指令，只有指定的数据大小被复制到目的单元。例如，如果属性被指定为 SINT 而目的单元是 DINT，则只有 DINT 目的单元的低 8 位被更新，其余的 24 位不改变。

用户可以访问下列对象:

有关下列对象的信息:	参见页次:
AXIS	3-24
CONTROLLER	3-31
CONTROLLERDEVICE	3-31
CST	3-33
DF1	3-34
FAULTLOG	3-37
MESSAGE	3-37
MODULE	3-39
MOTIONGROUP	3-40
PROGRAM	3-40
ROUTINE	3-41
SERIALPORT	3-41
TASK	3-43
WALLCLOCKTIME	3-44

访问 AXIS 对象

AXIS 对象提供有关伺服模块传动轴的状态信息。指定传动轴的标签名称以确定用户选择的 AXIS 对象，详见 *Logix5550 运动控制模块用户手册*,出版号 1756-6.5.16. 中关于 AXIS 对象一节。

当属性用星号标记(*)时，意味着属性同时位于 ControlLogix 控制器和传动模块内。当用户用一条SSV 指令写一个数值时，控制器自动更新模块内的拷贝。然而，这一过程并不马上进行。要确认模块内的值已经更新，可以使用传动轴标签更新状态逻辑位的互锁机构。

例如, 如果用户对位置锁定误差执行一条SSV指令，则轴标签的位置误差状态将被置位直至模块更新成功。因此, 紧随SSV指令之后的逻辑程序可以在继续执行之前等待该位复位。

属性:	数据类型:	指令:	说明:
* 加速度前馈增益	REAL	GSV SSV	该值用于提供转矩控制输出，以实现操作加速。
实际位置	REAL	GSV	轴的实际位置
实际速率	REAL	GSV	轴的实际速率
平均速率	REAL	GSV	轴的平均速率
平均速率时基	REAL	GSV SSV	轴的平均速率的时基
轴配置状态	SINT	GSV	轴的配置状态
* 轴类型	INT	GSV SSV	用户正在使用的轴的类型 值: 意义: 0 未使用的轴 1 位置轴 2 伺服轴
控制位置	REAL	GSV	轴的控制位置
控制速率	REAL	GSV	轴的控制速率
换算常数	REAL	GSV SSV	该换算常数用于将用户单位转变为反馈计数。
衰减因数	REAL	GSV SSV	计算在执行运动运行轴调整指令期间最大位置伺服带宽需使用的的数值
* 驱动故障事件	SINT	GSV SSV	当产生驱动故障时，执行该操作 值: 意义: 0 轴停止运转 1 使驱动器不工作 2 停止控制运动 3 仅改变状态位
有效惯量	REAL	GSV	从在上一次运动运行轴调整(MRAT)指令期间控制器所进行的测量中计算出的轴的惯性值
编码器丢失故障事件	SINT	GSV SSV	当发生编码器丢失故障时，执行该操作 值: 意义: 0 轴停止运转 1 使驱动器不工作 2 停止控制运动 3 只改变状态位
* 编码器噪声故障事件	SINT	GSV SSV	当编码器噪声干扰故障产生时，执行该操作。 值: 意义: 0 轴停止运转 1 使驱动器不工作 2 停止控制运动 3 只改变状态位
* 摩擦补偿	REAL	GSV SSV	固定的输出电平用以补偿静摩擦。
组实例	DINT	GSV	包含用户的轴的运动组实例数

属性:	数据类型:	指令:	说明:
复位模式	SINT	GSV SSV	轴的复位模式 值: 意义: 0 被动复位 1 主动复位(缺省)
复位位置	REAL	GSV SSV	轴的复位位置
复位返回速度	REAL	GSV SSV	轴的复位返回速度
复位顺序类型	SINT	GSV SSV	轴的复位顺序类型 值: 意义: 0 立即复位 1 开关复位 2 标记复位 3 开关 - 标记复位(缺省)
复位速度	REAL	GSV SSV	轴复位速度
实例	DINT	GSV	轴的实例数量
映象表实例	DINT	GSV	伺服模块的 I/O 映象实例
最大加速度	REAL	GSV SSV	轴的最大加速度
最大减速度	REAL	GSV SSV	轴的最大减速度
* 最大反向行程	REAL	GSV SSV	最大反向行程限制
* 最大正向行程	REAL	GSV SSV	最大正向行程限制
最大速度	REAL	GSV SSV	轴的最大速度
模块通道	SINT	GSV	用户伺服模块的模块通道
运动配置位	SINT	GSV SSV	轴的运动配置位 位: 意义: 0 复位方向相反 1 复位开关正常闭合 2 复位信标边界负向
运动故障位	DINT	AXIS 结构体	轴运动故障位 位: 位名称: 意义: 0 ACAsyncConn Fault 异步连接故障 1 ACSyncConn Fault 同步连接故障
运动状态位	DINT	AXIS 结构体	轴运动状态位 位: 位名称 意义: 0 Accel Status 加速度 1 Decel Status 减速度 2 Move Status 转动 3 Jog Status 慢进给 4 Gearing Status 齿轮传动 5 Homing Status 复位 6 Clutch Status 啮合 7 Axis Homed Status 复位状态

属性:	数据类型:	指令:	说明:
电动机编码器测试增量	REAL	GSV SSV	启动运动运行连接诊断(MRHD)测试所必需的运动量
* 输出滤波器带宽	REAL	GSV SSV	伺服低通数字输出滤波器的带宽
* 输出限制	REAL	GSV SSV	轴的最大伺服输出电压值
* 输出偏差量	REAL	GSV SSV	该值用于补偿伺服模块 DAC 输出和伺服驱动输入的累积偏移影响。
* 输出定标	REAL	GSV SSV	该值用于将伺服系统的输出转变为驱动器的等效电压。
位置误差	REAL	GSV	轴的实际位置与控制位置间的差别
* 位置误差故障事件	SINT	GSV SSV	当产生位置误差故障时,执行该操作 值 意义: 0 轴停止运转 1 使驱动器不工作 2 停止控制运动 3 只改变状态位
* 位置容差	REAL	GSV SSV	在造成位置误差故障前,伺服模块能容许的位置误差量
* 位置积分增益	REAL	GSV SSV	在静摩擦及重力的干扰下,该值用于实现在含有静摩擦和重力影响下的精确的轴定位。
位置积分器误差	REAL	GSV	一个轴的位置误差总和
位置锁定容差	REAL	GSV SSV	当给出一锁定状态指示的实际位置时，伺服模块所能容许的位置误差量。
* 位置比例增益	REAL	GSV SSV	控制器复接位置误差以纠正位置误差的值(1/ 毫秒)
位置伺服带宽	REAL	GSV SSV	一致增益带宽，控制器用以对运动应用轴调整指令进行增益计算
* 位置松开	DINT	GSV SSV	该值用于执行旋转轴的自动松开。
编程停止模式	SINT	GSV SSV	停止轴操作的类型 值: 意义: 0 快速停止 1 快速关闭 2 突然停车
登录位置	REAL	GSV	轴的登录位置
* 伺服配置位	DINT	GSV SSV	用于伺服系统的伺服配置位 位: 意义: 0 旋转轴 1 外部速率伺服驱动 2 编码器极性负 3 伺服极性负 4 软超程检查 5 位置误差检查 6 编码器丢失故障检查 7 编码器噪声干扰故障检查 8 驱动故障检查 9 驱动故障正常闭合

属性:	数据类型:	指令:	说明:		
伺服配置更新位	DINT	AXIS	用户伺服系统的伺服配置状态位		
		结构体	位:	位名称:	意义:
			0	Axis Type Status	轴的类型
			1	PosUnwnd Status	位置松开
			2	MaxPTrvl Status	最大正向行程
			3	MaxNTrvl Status	最大反向行程
			4	PosErrorTol Status	位置错误容差
			5	PoslockTol Status	位置锁定容差
			6	PosPGain Status	位置比例增益
			7	PoslGain Status	位置积分增益
			8	VelF Gain Status	速率前馈增益
			9	AccFf Gain Status	加速度前馈增益
			10	VelP Gain Status	速率相对增益
			11	Vel Gain Status	速率整体增益
			12	Out Filt Bw Status	输出滤波器带宽
			13	Out Scale Status	输出定标
			14	Out Limit Status	输出限制
			15	Out Offset Status	输出偏移量
			16	FricComp Status	摩擦补偿
			17	POtrvl Fault Act Status	软超程故障事件
			18	Pos Error Fault Act Status	位置错误故障事件
			19	Enc Loss Fault Act Status	编码器丢失故障事件
			20	EncNs Fault Act Status	编码器噪声干扰故障事件
			21	Drive Fault Act Status	驱动故障事件
伺服事件位	DINT	AXIS	伺服回路中的伺服结果位		
		结构体	位:	位名称:	意义:
			0	WatchEvArm Status	提供监视事件
			1	WatchEv Status	监视事件
			2	RegEvStatus	提供登录事件
			3	RegEvArm status	登录事件
			4	HomeEvArm Status	提供复位事件
			5	HomeEv status	复位事件
伺服故障位	DINT	AXIS	伺服回路的伺服故障位		
		结构体	位:	位名称:	意义:
			0	POtrvl Fault	正向超程故障
			1	NOtrvl Fault	反向超程故障
			2	Pos Error Fault	位置错误故障
			3	EncCHA Loss Fault	编码器通道 A 丢失故障
			4	EncCHB Loss Fault	编码器通道 B 丢失故障
			5	EncCHZ Loss Fault	编码器通道 Z 丢失故障
			6	EncNs Fault	编码器噪声故障
			7	Drive Fault	驱动故障
			8	SyncConn Fault	同步连接故障
			9	Hard Fault	伺服硬件故障
伺服输出电平	REAL	GSV	在轴伺服系统中的输出电压等级。		

属性:	数据类型:	指令:	说明:		
伺服状态位	DINT	AXIS 结构体	伺服回路的状态位		
			位:	位名称:	意义:
			0	Servo Act Status	伺服作用
			1	Drive Enable Status	驱动使能
			2	OutLmt Status	输出限制
			3	PosLock Status	位置锁定
			13	Tune Status	调整过程
			14	Test Status	诊断测试
			15	Shutdown Status	轴停止运转
伺服状态更新位	DINT	GSV SSV	轴伺服状态更新位		
			位:	意义:	
			0	位置误差更新	
			1	位置积分器误差更新	
			2	速率误差更新	
			3	速率积分器误差更新	
			4	速率控制更新	
			5	速率反馈更新	
			6	伺服输出电平更新	
* 软超程故障事件	SINT	GSV SSV	当一个软超程故障发生时，执行该操作		
			值:	意义:	
			0	轴停止运转	
			1	使驱动器不工作	
			2	停止控制运动	
			3	只改变状态位	
实际起始位置	REAL	GSV	当对轴的新的控制运动开始时，轴的实际位置 (位置单位)		
控制起始位置	REAL	GSV	当对轴的新的控制运动开始时，轴的控制位置 (位置单位)		
选通实际位置	REAL	GSV	当执行运动组选通位置指令时，轴的实际位置		
选通控制位置	REAL	GSV	当运动组选通位置指令执行时，轴的控制位置 (位置单位)		
测试方向	BOOL	GSV	执行运动运行连接诊断指令 (MRHD) 期间，伺服模块鉴别出的轴的行程方向		
			值:	意义:	
			0	负向 (反) 方向	
			1	正向 (前) 方向	
测试状态	INT	GSV	最后的运动运行连接诊断指令的状态		
			值:	意义:	
			0	测试过程成功	
			1	测试正在进行中	
			2	测试过程被用户中断	
			3	超过 2 秒的超时测试	
			4	由于伺服故障导致测试过程失败	
			5	检验增量不足	
调整加速度	REAL	GSV	在最后的运动运行轴调整 (MRAT) 指令期间测量的加速度值		
调整加速度时间	REAL	GSV	在最后的运动运行轴调整 (MRAT) 指令期间测量的加速度时间		
调整减速度	REAL	GSV	在最后的运动运行轴调整 (MRAT) 指令期间测量的减速度值		

属性:	数据类型:	指令:	说明:
调整负加速度时间	REAL	GSV	在最后的运动运行轴调整(MRAT)指令期间测量的负速度时间
调整上升时间	REAL	GSV	在最后的运动运行轴调整 (MRAT)指令期间测量的轴上升时间
调整速度定标	REAL	GSV	在最后的运动运行轴调整 (MRAT)指令期间测量的轴驱动定标因子
调整状态	INT	GSV	最后的运动运行轴调整 (MRAT)指令的状态 值: 意义: 0 调整过程成功 1 调整正在进行中 2 调整过程被用户中断 3 超过 2 秒的超时调整 4 伺服故障导致调整过程失败 5 轴达到了调整行程界限 6 轴的极性设置不正确 7 调整速度过小以致不能进行测量
调整速率带宽	REAL	GSV	从在最后的运动运行轴调整 (MRAT)指令期间进行的测量中计算出的驱动器带宽。
调整配置位	DINT	GSV SSV	轴的调整配置位 位: 意义: 0 调整方向(0= 正向，1= 反向) 1 调整位置误差积分器 2 调整速率误差积分器 3 调整速率前馈位 4 加速度前馈 5 调整速率低通滤波器
调整速度	REAL	GSV SSV	由运动运行轴调整 (MRAT)指令初始化的最大速度。
调整行程限制	REAL	GSV SSV	运动运行轴调整 (MRAT)指令所使用的行程限制用以限制调整过程中的操作。
速率控制	REAL	GSV	当前速率参考轴的速率伺服回路。
速率误差	REAL	GSV	伺服轴的控制速率与实际速率间的差值。
速率反馈	REAL	GSV	伺服模块估计的轴实际速率。
* 速率前馈增益	REAL	GSV SSV	产生控制速率所必需的速率控制输出。
* 速率积分增益	REAL	GSV SSV	控制器乘以速率累积误差值以纠正速率误差的值。
速率累积误差	REAL	GSV	指定轴速率误差总数。
* 速率比例增益	REAL	GSV SSV	控制器乘以速率误差以纠正速率误差的值。
监视位置	REAL	GSV	轴的监视位置。

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:
时间段	INT	GSV	被分配用于通讯的 CPU 可利用的百分比。有效值是 10-90。
		SSV	当控制器钥匙开关在运行(run)位置时不能改变该值。

访问 CONTROLLERDEVICE 对象

CONTROLLERDEVICE 对象确定控制器的物理硬件。

属性:	数据类型:	指令:	说明:
设备名称	SINT[33]	GSV	控制器的 ASCII 字符串名称。 第一个字节包含数组串返回的 ASCII 字符的数字总数。
产品代码	INT	GSV	表明控制器的类型 Logix5550 = 3
产品型号	INT	GSV	表明当前产品型号。以十六进制显示。 低位字节包含主型号；高位字节包含辅助型号。
系列号	DINT	GSV	设备的系列号。 在设备出厂时就已经分配的系列号。

属性:	数据类型:	指令:	说明:
状态	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 正在进行闪烁内存更新 0010 保留 0011 保留 0100 闪烁内存损坏 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 **Logix5550 特有的状态位** **位 13-12: 意义:** 01 钥匙开关在运行 (run) 位置 10 钥匙开关在编程 (program) 位置 11 钥匙开关在远程 (remote) 位置 **位 15-14: 意义:** 01 控制器正在转换模式 10 调试模式，如果控制器在运行模式
类型	INT	GSV	表示设备是一个控制器。 控制器 = 14
制造商	INT	GSV	表示设备的制造商。 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:
当前状态	INT	GSV	协调系统时间的当前状态。数据位表示: 位: 含义: 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 **线性变化使能**: 定时器的低16位的当前值线性变化为需要的数值，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。 2 **主系统时间**: CST 对象是 ControlLogix 系统的主时间。 3 **同步**: CST对象的64位当前值由主机CST对象通过系统时间更新来同步 4 **本地网络主机**: CST 对象是本地网络的主时间源 5 **处于中继模式**: CST 对象工作于时间中继模式 6 **检测到多主机**: 检测到一个重复的本地网络主时间源。对于时间相关节点该位一直为零。 7 未使用 8-9 时间节点类型 00= 时间相关节点 01= 时间主节点 10= 时间中继节点 11= 未使用 10-15 未使用
当前值	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同步提供的数值相匹配。按照当前状态属性中指出的那样，这种调整或者是线性过渡到所要求的数值，或者是立即设定为所要求的数值。

访问 DF1 对象

DF1 对象提供一个到 DF1 通讯驱动程序的界面，可以用于配置串行端口。

属性:	数据类型:	指令:	说明:
ACK 超时	DINT	GSV	等待信息传送返回确认的时间量。(只用于点对点和主机通讯)。有效值是 0-32,767. 以 20 毫秒延迟周期计数。默认值是 50 (1 秒)。
诊断计数器	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 从机	主机
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 ACKed	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	丢失调治解调器计数	丢失调治解调器计数	丢失调治解调器计数
14	不使用	不使用	最大优先扫描时间
15	不使用	不使用	最后优先扫描时间
16	不使用	不使用	最大正常扫描时间
17	不使用	不使用	最后正常扫描时间
18	ENQs 发送	不使用	不使用

多重信息检测	SINT	GSV	使能多重信息检测。 值: 意义: 0 禁止检测重复信息 非零值 禁止检测重复信息
内置响应使能	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 (默认设置) 1 无条件使能
ENQ 发送限制	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. 默认设置是 3.
EOT 抑制	SINT	GSV	使能抑制 EOT 传送响应查询信息包(只适用于从机)。 值: 意义: 0 EOT 抑制禁止 (无效) 非零值 EOT 抑制使能
错误检测	SINT	GSV	指定错误检测配置。 值: 意义: 0 BCC (默认设置) 1 CRC

属性:	数据类型:	指令:	说明:
主机信息传送	SINT	GSV	主机信息传送的当前值(只适用于主机). 值: 意义: 0 在各站之间轮询 1 在查询序列(代替主机的站号) 默认值是 0.
NAKR 接收限制	SINT	GSV	在停止传送之前为响应一个通讯控制器所能接收的NAKs数量(只适用于点对点通讯)。 有效值是 0-127. 默认设置是 3.
正常轮询组大小	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(只适用于主机). 有效值 0-255. 默认值是 0.
轮询模式	SINT	GSV	当前轮询模式(只适用于主机)。 值: 意义: 0 基于信息,但不允许从机启动通讯。 1 基于信息, 但是允许从机启动通讯 (默认设置) 2 标准，每次扫描节点传送单个信息。 3 标准，每次扫描节点传送多条信息。 默认值设置为 1.
等待回答信息	DINT	GSV	在为了得到响应轮询从机之前，接收到一个 ACK 之后等待(作为主机)的时间(只适用于主机) 有效范围是 0-65,535. 以 20 毫秒延迟周期计数。默认值是 5 个周期 (100 毫秒).
站地址	INT	GSV	串口的当前站地址。 有效值是 0-254. 默认值是 0.
从机轮询超时	DINT	GSV	在从机声明其不能传送之前(因为主机在激活状态)，等待主机轮询的时间量(以毫秒为单位)(只适用于从机)。 有效值是 0-32,767. 以 20 毫秒延迟周期计数。默认值是 3000 个周期 (1 分钟).
重试传送	SINT	GSV	重新发送一条信息但没有得到确认的次数。(只适用于主机和从机)。 有效值是 0-127。默认值是 3.
挂起的应答超时	DINT	SSV	应答超时属性的挂起数值
挂起的二重检测	SINT	SSV	二重检测属性的挂起数值。
挂起的嵌入响应使能	SINT	SSV	嵌入响应属性的挂起数值。
挂起的 ENQ 发送制限	SINT	SSV	ENQ 发送限制属性的挂起数值
挂起的 EOT 抑制	SINT	SSV	EOT 抑制属性的挂起数值。
挂起的错误检测	SINT	SSV	错误检测属性的挂起数值。
挂起的正常查询组大小	INT	SSV	正常查询组大小属性的挂起数值。
挂起的主信息发送	SINT	SSV	主信息发送属性的挂起数值。
挂起的 NAK 接收限制	SINT	SSV	NAK 接收限制属性的挂起数值。
挂起的查询模式	SINT	SSV	查询模式属性的挂起数值。

属性:	数据类型:	指令:	说明:
挂起的应答信息等待	DINT	SSV	应答信息等待属性的挂起数值。
挂起的站点地址	INT	SSV	站点地址属性的挂起数值。
挂起的从查询超时	DINT	SSV	从查询超时属性的挂起数值。
挂起的发送重试	SINT	SSV	发送重试属性的挂起数值

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
2 使用 MSG 指令来应用该数值。配置 MSG 指令:

MSG 配置 Tab:	区域:	值:
配置	信息类型	一般的 CIP
	服务代码	0d 十六进制
	对象类型	a2
	对象 ID	1
	对象属性	空白
	源操作数	空白
	元素数量	0
	目的操作数	空白
通讯	路径	到自己的通讯路径 (1, s 这里 s = 控制器的槽号)

访问 FAULTLOG 对象

FAULTLOG 对象提供控制器的状态信息。

属性:	数据类型:	指令:	说明:
主要事件	INT	GSV SSV	从上次该计数器被复位以来，发生的主要错误的次数。
次要事件	INT	GSV SSV	从上次该计数器被复位以来，发生的次要错误的次数。
主要故障位	DINT	GSV SSV	各个位指示引起当前主要错误的原因。
次要故障位	DINT	GSV SSV	各个位指示引起当前次要错误的原因。

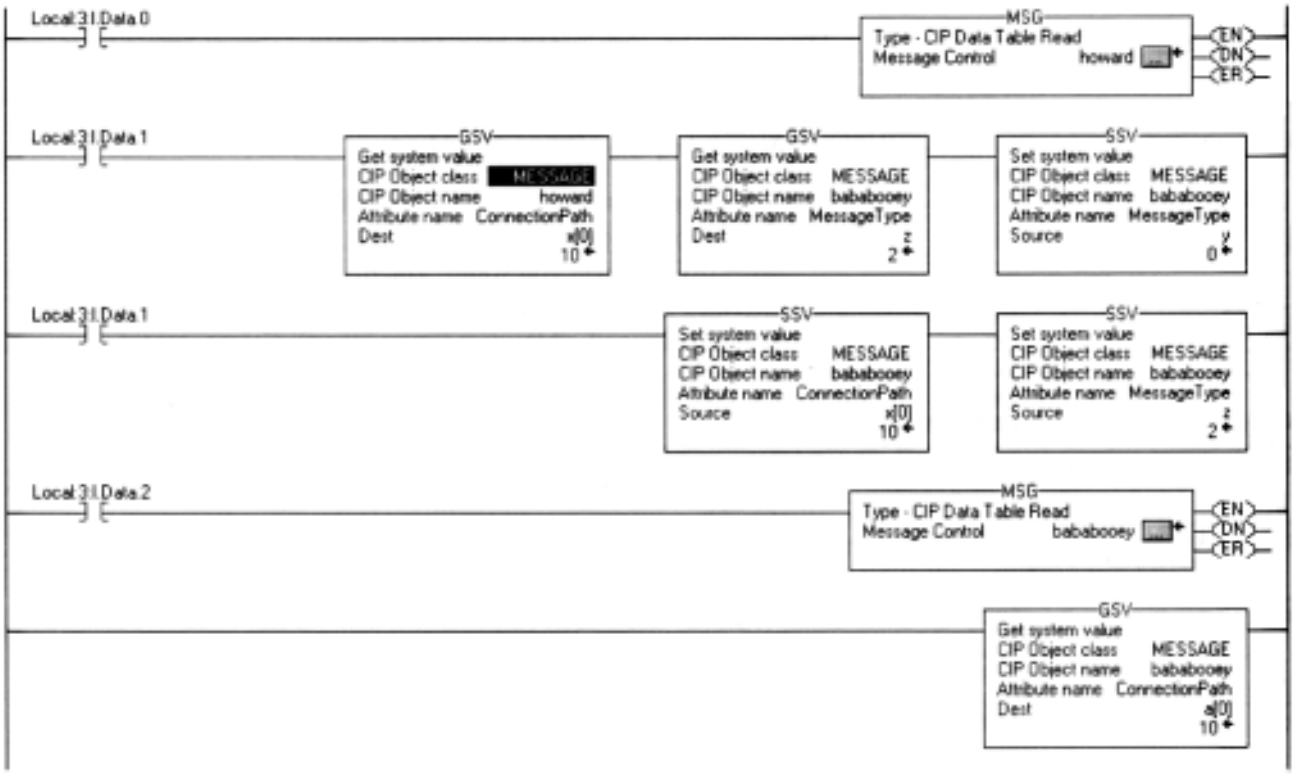
访问 MESSAGE 对象

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对象提供了一个建立和触发对等通讯的界面。该对象取代了 PLC-5 处理器的 MG 数据类型。

属性:	数据类型:	指令:	说明:
连接通路	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)是按字节表示的连接路径的长度。
连接速率	DINT	GSV SSV	请求信息包的连接速率。
信息类型	SINT	GSV SSV	指定信息的类型。 **数值:** **含义:** 0 未初始化
端口	SINT	GSV SSV	指明信息应被发送到的端口。 **数值:** **含义:** 1 背板， 2 串行接口
超时乘数	SINT	GSV SSV	确定一个连接应被作为超时并被关闭的时间。 **数值:** **含义:** 0 连接将以 4 倍的更新速率产生超时(默认值) 1 连接将以 8 倍的更新速率产生超时 2 连接将以 16 倍的更新速率产生超时
未连接超时	DINT	GSV SSV	用于所有未连接信息的以毫秒为单位的超时间隔。默认值是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把它存入一个标签。
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定回在步骤 1 得到的原有值。



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，被指定的模块必须存在于控制器编组器的 I/O 配置部份，且必须有一个设备名。

属性:	数据类型:	指令:	说明:
入口状态	INT	GSV	指明指定的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 12-15 位是有效的。 **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。不应该使用该数值来确定模块是否失效，因为当模块对象试图重新与模块连接时会周期性的脱离该状态。代之，测试运行状态 (16#4000)。检测故障码不等于零以确定是否模块出错。出错时，故障码和错误信息属性有效，错误状态被改正。 16#2000 **有效性验证**: 模块对象在建立与模块的连接之前验证模块对象的完整性。 16#3000 **连接**: 模块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成功的进行数据传送。 16#5000 **关闭**: 模块对象处于关闭所有与模块的连接的过程中。 16#6000 **禁止**: 模块对象被禁止(在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。
故障代码	INT	GSV	如果发生故障，用于鉴别模块故障的号码。
故障信息	DINT	GSV	提供关于模块对象故障代码的说明信息。
实例	IDINT	GSV	提供该模块对象的实例号码。

属性:	数据类型:	指令:	说明:
模式	INT	GSV SSV	指明模块对象的当前模式。 位: 含义: 0 置位时，当控制器处于运行状态时，如果任何模块对象连接失败，将导致产生一个主要错误。 2 置位时，在关闭与模块的所有连接后，导致模块对象进入禁止状态。
LED 状态	INT	GSV	标明在控制器前部的 I/O LED 的当前状态。 数值: 含义: 0 LED 关闭：对于控制器，没有配置模块对象(在控制器编组器的 I/O 配置部分没有模块。 1 红色闪烁：没有运行的模块对象。 2 绿色闪烁：至少有一个模块对象没有运行。 3 绿色稳定：所有的模块对象都在运行。 注释：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。

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:
实例	DINT	GSV	提供 MOTION_GROUP 对象的实例号码。

访问 PROGRAM 对象

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

属性:	数据类型:	指令:	说明:
禁止标志	SINT	GSV SSV	控制程序的执行。 数值: 含义: 0 使能执行 1 禁止执行
实例	DINT	GSV	提供该 PROGRAM 对象的实例号码。
最后一次扫描时间	DINT	GSV SSV	最后一次程序执行所用的时间，以毫秒为时间单位。

属性:	数据类型:	指令:	说明:
主要故障记录	DINT[11]	GSV SSV	记录该程序的主要故障 建议用户创建一个用户定义的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:
时间低位	DINT	十进制数	故障时间标志值的低 32 位。
时间高位	DINT	十进制数	故障时间标志值的高 32 位。
类型	INT	十进制数	故障类型(程序，I/O 等)
代码	INT	十进制数	对故障的唯一代码(与故障类型有关)
信息	DINT[8]	十六进制数	特定的故障信息(与故障类型和代码有关)

最大扫描时间	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。
次要故障记录	DINT[11]	GSV SSV	记录该程序的次要故障。建议用户创建一个用户定义的结构体以简化对次要故障记录属性的访问。

名称:	数据类型:	格式:	说明:
时间低位	DINT	十进制数	故障时间标志值的低 32 位。
时间高位	DINT	十进制数	故障时间标志值的高 32 位。
类型	INT	十进制数	故障类型(程序，I/O 等)
代码	INT	十进制数	对故障的唯一代码(与故障类型有关)
信息	DINT[8]	十六进制数	特定的故障信息(与故障类型和编码有关)

SFC 再启动	INT	GSV SSV	未使用 — 保留以备将来使用

访问 ROUTINE 对象

ROUTINE 对象提供关于一个进程的状态信息。指定进程名称以确定用户要访问的进程对象。

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

访问 SERIALPORT 对象

SERIALPORT 对象提供一个到串行通讯端口的界面。

属性:	数据类型:	指令:	说明:
波特率	DINT	GSV	指定波特率。有效的波特率为 110，300，600，1200，2400，4800，9600 和 19200(默认值)。
数据位数	SINT	GSV	指定每个字符的数据位数。 **数值:** **含义:** 7 7 个数据位(仅适用于 ASCII 码) 8 8 个数据位(默认值)

属性:	数据类型:	指令:	说明:
校验	SINT	GSV	设定奇偶校验。 数值: 含义: 0 无奇偶校验 1 奇校验 2 偶校验
RTS 关断延时	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。以 20 毫秒间隔为计数单位延时。默认值是 0 毫秒。
RST 发送延时	INT	GSV	开通 RTS 线后到发送信息的第一个字符间的延时时间。 有效值为 0-32，767。以 20 毫秒间隔为计数单位延时。默认值是 0 毫秒。
停止位	SINT	GSV	设定停止位数。 数值: 含义: 1 1 个停止位(默认值) 2 2 个停止位(只适用于 ASCII 码)
挂起波特率	DINT	SSV	波特率属性的挂起值。
挂起数据位	SINT	SSV	数据位属性的挂起值。
挂起奇偶校验	SINT	SSV	挂起的奇偶校验属性值。
挂起 RTS 关断延时	INT	SSV	挂起的 RTS 关断延时属性值。
挂起 RTS 发送延时	INT	SSV	挂起的 RTS 发送延时属性值。
挂起停止位	SINT	SSV	停止位属性的挂起值。

应用任何 SERIALPORT 挂起属性值:

1. 使用 SSV 指令设置挂起属性值。
2. 使用MSG指令应用这些值。按照如下配置MSG指令:

MSG 配置列表:	区域:	数值:
配置	信息类型	通用 CIP
	服务代码	0d 十六进制
	对象类型	6f 十六进制
	对象 ID	1
	对象属性	空白
	源操作数	空白
	元素个数	0
	目的操作数	空白
通讯	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 对象提供关于一个任务的状态信息。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:
实例	DINT	GSV	提供 TASK 对象的实例号。 有效值是 0-31。
最后一次扫描时间	DINT	GSV SSV	最后一次程序执行所用的时间，以毫秒为时间单位。
最大时间间隔	DINT[2]	GSV SSV	连续两次执行该任务间的最大时间间隔。DINT[0]包含该值的低 32 位，DINT[1]包含该值的高 32 位。数值为零表明该任务的执行次数等于或小于 1。
扫描时间最大值	DINT	GSV SSV	记录的该程序执行时间的最大值，以毫秒为时间单位。
最小时间间隔	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。
优先级	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。
速率	DINT	GSV SSV	执行该任务间的时间间隔，以毫秒为时间单位。
开始时间	DINT[2]	GSV SSV	当该任务的最近一次执行开始后的 WALLCLOCKTIME 的数值。DINT[0]包括该值的低 32 位，DINT[1]包括该值的高 32 位。
看门狗定时器	DINT	GSV SSV	用于执行与该任务相关的所有程序的时间限制。如果输入零，则进行如下赋值: **时间**: **任务类型**: 0.5 秒 周期性 5.0 秒 连续

访问 WALLCLOCKTIME 对象

WALLCLOCKTIME 对象提供一个控制器可用于排定时间计划的时间标志。

属性:	数据类型:	指令:	说明:
CST 偏置	DINT[2]	GSV SSV	相对于协调系统时间值的正的偏置值。DINT[0]包含该值的低 32 位，DINT[1]包含该值的高 32 位。数值以毫秒为单位。默认值为零。
当前值	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 1981 年 1 月 1 日 0000 时为参照。
时间数据	DINT[7]	GSV	数据和时间的可读格式是: 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)

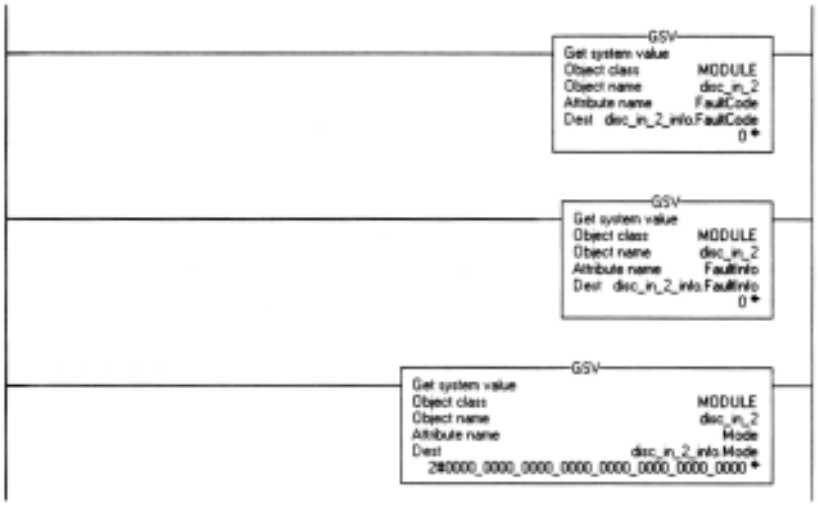
GSV/SSV 指令编程实例

获取故障信息

下面是使用 GSV 指令获取故障信息的实例。

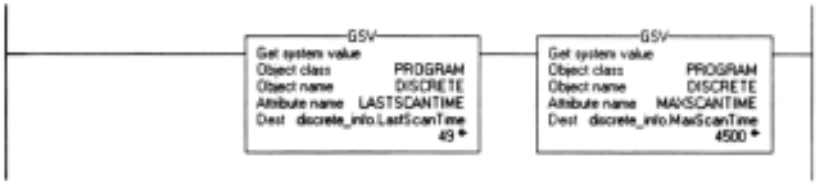
获取 I/O 故障信息

本例从 I/O 模块 *disc_in_2* 获取错误信息，并把数据存储于用户定义的结构体 *disc_in_2_info* 中。



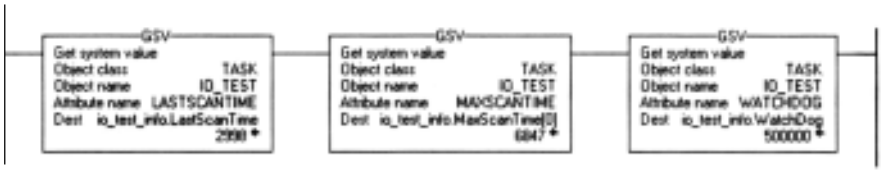
获取程序状态信息

本例获取关于程序 discrete 的状态信息并把数据存储于用户定义的结构体 *discrete_info* 中。



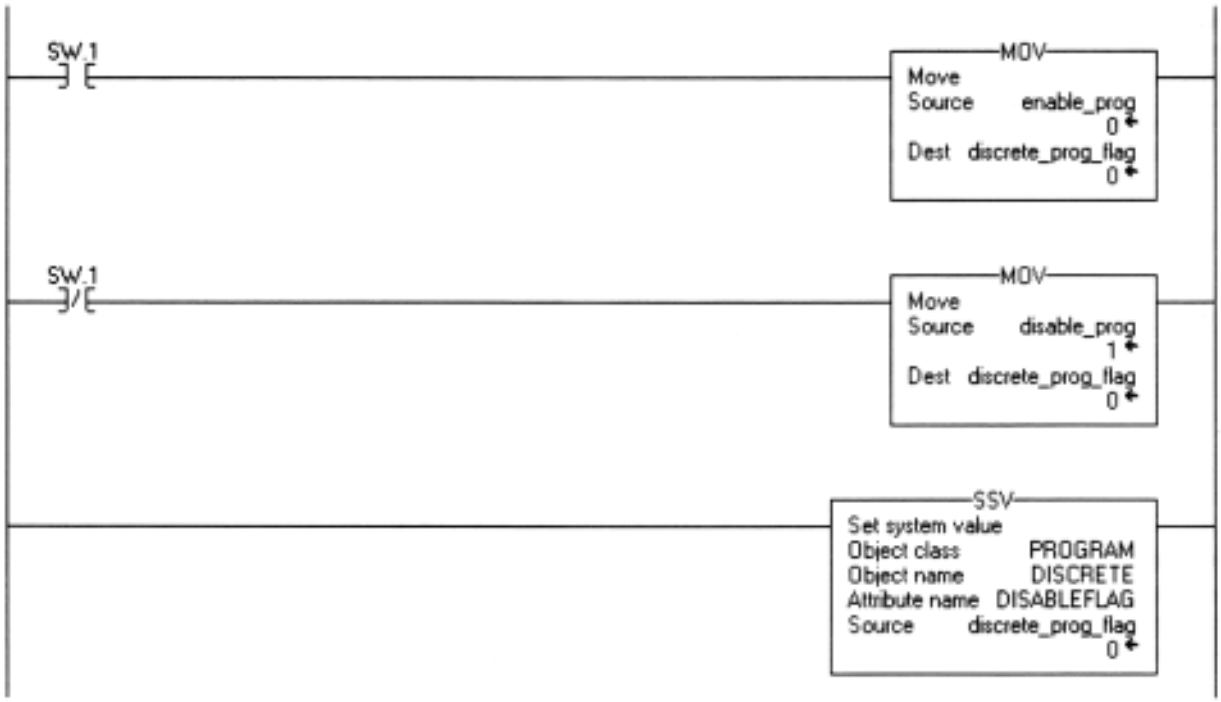
获取任务状态信息

本例获取关于任务 IO_test 的状态信息并把数据存储于用户定义的结构体 *io_test_info* 中。



设置使能和禁止标志

下列实例使用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，这是与 PLC-5 使用禁止位相似的程序处理方法。



基于状态 *SW.1*，将适当的值放入程序 discrete 的 *disableflag* 属性中。

比较指令

(CMP，EQU，GEQ，GRT，LEQ，LES，LIM，MEQ，NEQ)

简介

比较指令使用户通过用表达式或专用比较指令进行数值比较。

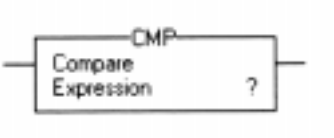
如果用户要:	使用下列指令:	参见页次:
根据表达式进行数值比较	CMP	4-2
测试二值是否相等	EQU	4-6
测试一个值是否大于或等于另一个值	GEQ	4-8
测试一个值是否大于另一个值	GRT	4-10
测试一个值是否小于或等于另一个值	LEQ	4-12
测试一个值是否小于另一个值	LES	4-14
测试一个值是否在另两个值范围之内	LIM	4-16
通过屏蔽测试二值是否相等	MEQ	4-19
测试一个值是否不等于另一个值	NEQ	4-22

可以比较不同数据类型的数值，例如浮点数与整数比较。

黑体字数据类型是最优的数据类型。如果指令的所有操作数都使用同一最优数据类型，则指令执行的速度快，而且占用内存少。典型最优数据类型是DINT 或 REAL。

比较指令 (CMP)

CMP 指令是一条输入指令。



操作数:

操作数:	数据类型:	格式:	说明:
表达式	SINT	立即数	表达式由被运算符分隔的标
	INT	标签	签与 / 或立即数组成
	DINT		
	REAL		

说明: CMP 指令执行表达式中指定的算术运算比较。

用户要执行的运算由表达式定义。用运算符，标签，和立即数定义表达式。表达式中的复杂部分用圆括号( ) 定义。

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	DINT，REAL
ACS	反余弦	REAL
AND	按位与	DINT
ASN	反正弦	REAL

运算符:	说明:	最优数据类型:
ATN	反正切	REAL
COS	余弦	REAL
DEG	弧度转换成角度	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
NOT	位补码	DINT
OR	按位 OR	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	DINT
XOR	按位异或	DINT

确定运算顺序

指令按预先规定的顺序，而不必按用户列出的顺序，执行写入表达式的运算。可以通过把分组项组合到圆括号内来改变运算顺序，强制指令在执行其他运算之前执行圆括号内的运算，来改变运算顺序。

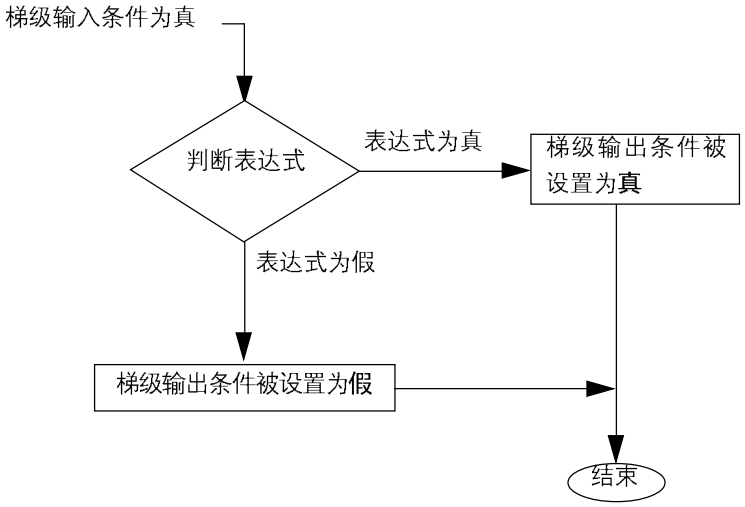
同级运算顺序从左向右执行。

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

与专用比较指令相比，执行一条 CMP 指令速度稍慢而且占用更多的内存。CMP 指令的优点是用户可以在一条指令内写入复杂的表达式。

执行:

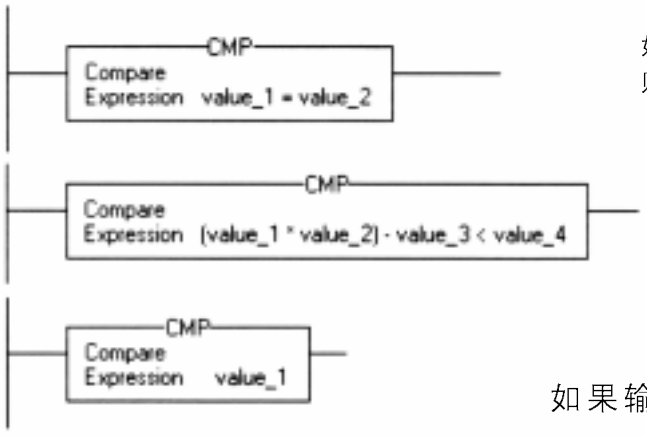
条件:	动作:
预扫描	梯级输出条件被设置为假。
梯级输入条件为假	梯级输出条件被设置为假。



算术状态标志: 不影响

故障条件: 无

CMP 指令举例:



如果 CMP 指令判断表达式为真，则梯级输出条件被设置为真。

如果输入一个没有比较运算符的表达时，例如，*Ivalue_1 + value_2*，或 *value_1*，则指令计算表达式的数值:

如果表达式的值是:	梯级输出条件被设置为:
非零值	真
零值	假

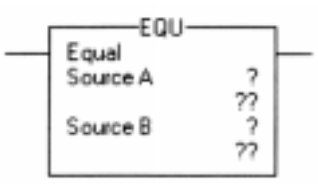
其他格式:

格式:	句法:
neutral 文本	`CMP` *`(expression);`*
ASCII 文本	`CMP` *`expression`*

相关指令: CPT

等于指令 (EQU)

EQU 指令是一条输入指令。



操作数:

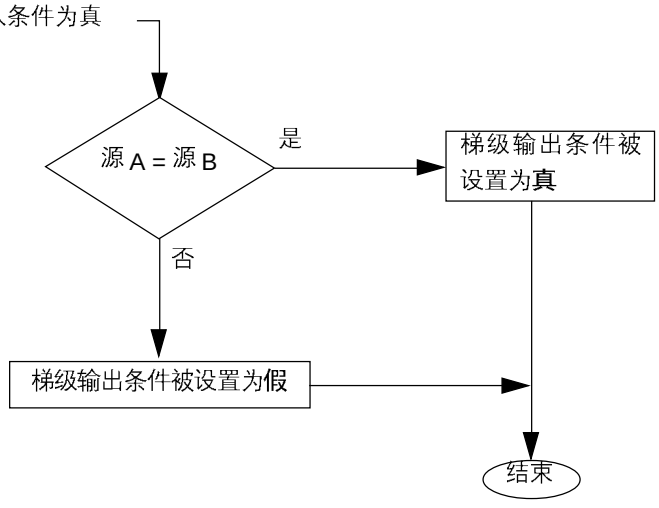
操作数:	数据类型:	格式:	说明:
源 A	SINT INT **DINT** **REAL**	立即数 标签	与源 B 比较的数值
源 B	SINT INT **DINT** **REAL**	立即数 标签	与源 A 比较的数值

说明:

EQU 指令测试源 A 的值与源 B 的值是否相等.

REAL 数据类型的数值很少绝对相等。如果必须确定两个 REAL 值是否相等，可以使用 LIM 指令。

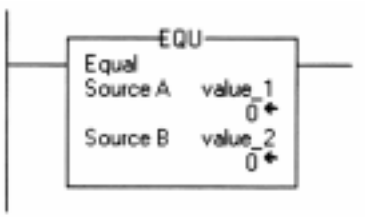
执行:

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	

算术状态标志: 不影响

故障条件: 无

EQU 指令举例:



如果 *value_1* 与 *value_2* 相等，则梯级输出条件被设置为真。

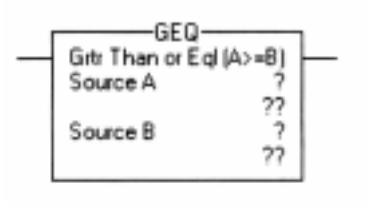
其它格式:

格式:	句法:
neutral 文本	`EQU (source_A,source_B);`
ASCII 文本	`EQU source_A source_B`

相关指令: CMP，GEQ，LEQ，MEQ，NEQ

大于或等于指令 (GEQ)

GEQ 指令是一条输入指令。

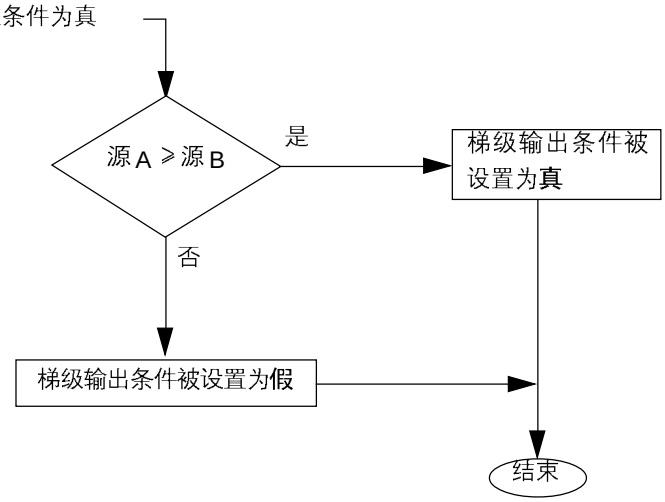


操作数：

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

说明：GEQ 指令测试源 A 的值是否大于或等于源 B 的值。

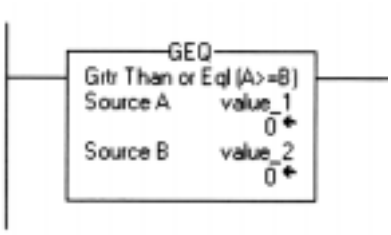
执行：

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	

算术状态标志: 不影响

故障条件: 无

GEQ 指令举例:



如果 *value_1* 大于或等于 *value_2*，则梯级输出条件被设置为真。

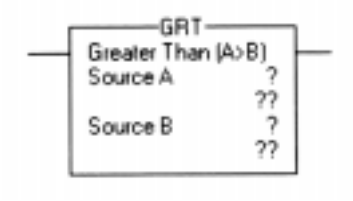
其它格式:

格式:	句法:
neutral 文本	`GEQ (source_A,source_B);`
ASCII 文本	`GEQ source_A source_B`

相关指令: CMP，EQU，LEQ，MEQ，NEQ

大于指令 (GRT)

GRT 指令是一条输入指令。



操作数:

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

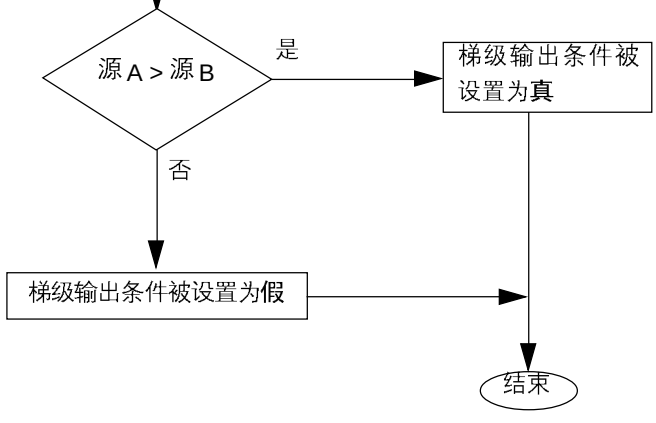
说明:

GRT 指令检测源 A 的值是否大于源 B 的值。

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。

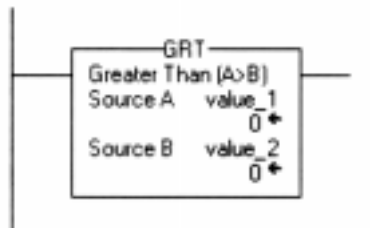
梯级输入条件为真



算术状态标志: 不影响

故障条件: 无

GRT 指令举例:



如果 *value_1* 大于 *value_2*，则梯级输出条件被设置为真。

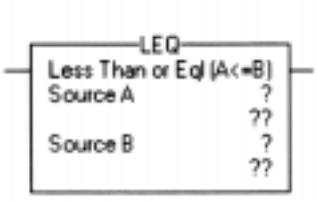
其它格式:

格式:	句法:
neutral 文本	`GRT (source_A,source_B);`
ASCII 文本	`GRT source_A source_B`

相关指令: CMP，LES

小于或等于指令 (LEQ)

LEQ 指令是一条输入指令。

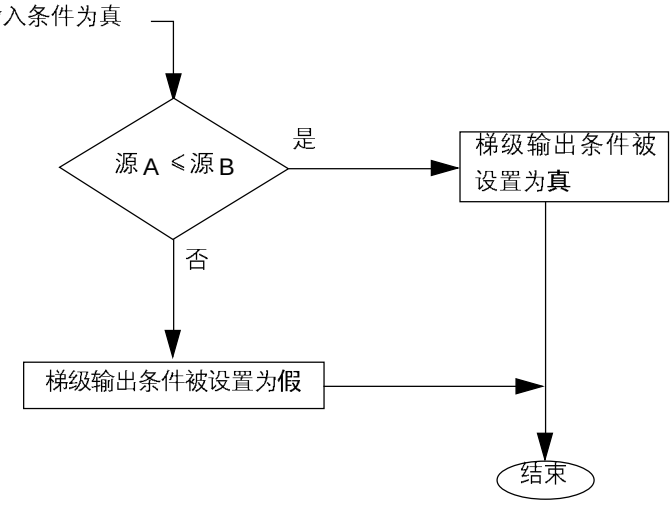


操作数:

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

说明: LEQ 指令测试源 A 的值是否小于或等于源 B 的值。

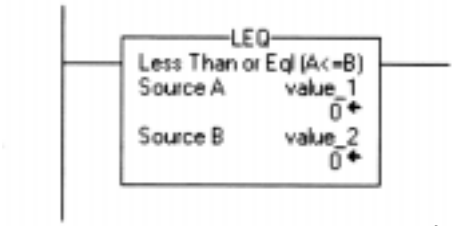
执行:

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	

算术状态标志: 不影响

故障条件: 无

LEQ 指令举例:



如果 *value_1* 小于或等于 *value_2*，则梯级输出条件被设置为真。

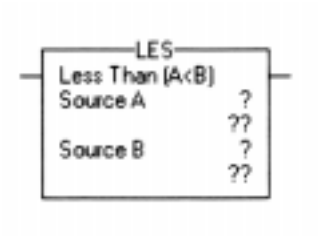
其它格式:

格式:	句法:
neutral 文本	`LEQ (source_A,source_B);`
ASCII 文本	`LEQ source_A source_B`

相关指令: CMP，EQU，GEQ，MEQ，NEQ

小于指令 (LES)

LES 指令是一条输入指令。



操作数:

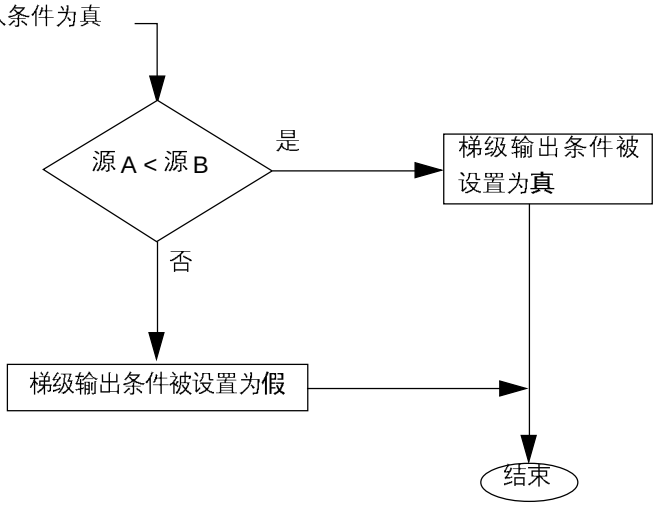
操作数:	**数据类型:**	**格式:**	**说明:**
源 A	SINT INT **DINT** **REAL**	立即数 标签	与源 B 比较的数值
源 B	SINT INT **DINT** **REAL**	立即 标签	与源 A 比较的数值

说明: LES 指令检测源 A 的值是否小于源 B 的值。

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。

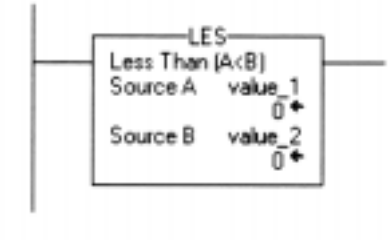
梯级输入条件为真



算术状态标志: 不影响

故障条件: 无

LES 指令举例:



如果 *value_1* 小于 *value_2*，则梯级输出条件被设置为真。

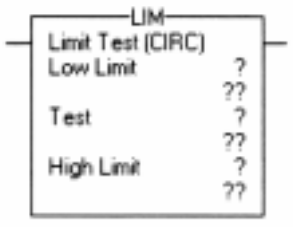
其它格式:

格式:	句法:
neutral 文本	LES *(source_A,source_B);*
ASCII 文本	LES *source_A source_B*

相关指令: CMP，GRT

极限比较指令 (LIM)

LIM 指令是一条输入指令。



操作数:

操作数:	数据类型:	格式:	说明:
下限	SINT	立即数	下限值
	INT	标签	
	DINT		
	REAL		
测试	SINT	立即数	测试值
	INT	标签	
	DINT		
	REAL		
上限	SINT	立即数	上限值
	INT	标签	
	DINT		
	REAL		

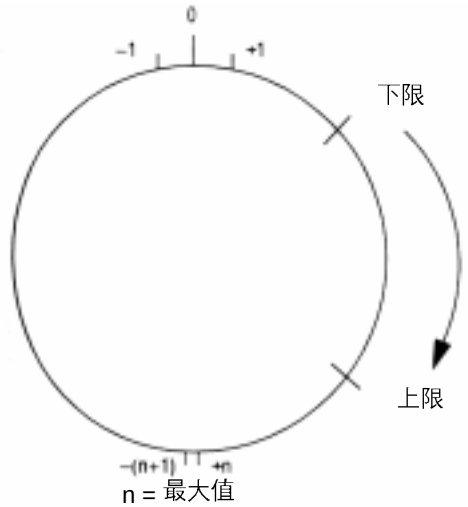
说明:

LIM 指令比较测试值是否在下限和上限值范围内。

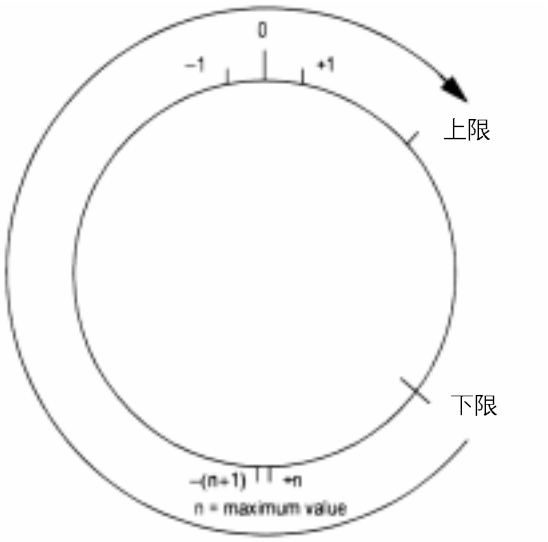
对于有符号整数，当其高位有效位被置位时，从最大正数“返回”到最大负数。例如，对于 16- 位整数(INT 数据类型)，最大正整数是 32,767，用十六进制表示位 16#7FFF(位 0 到 14 都置位)。如果用户对此数值加 1 则结果是 16#8000(位 15 置位)。对于有符号整数，十六进制 16#8000 等于十进制的 -32,768。从这一数值继续增加，直到全部 16 位都置位，结果是 16#FFFF，既等于十进制的 -1。

数值的这一变化过程可以用一循环线来说明(见下图)。LIM 指令从下限值开始按顺时针方向增加，直到达到上限值。只要测试值在下限值到上限值的顺时针方向范围内，梯级输出条件就被设置为真。如果测试值在上限值到下限值的顺时针方向范围内，则梯级输出条件被设置为假。

Low Limit ≤ High Limit
如果测试值等于或在下限值和上限值之间则指令为真。



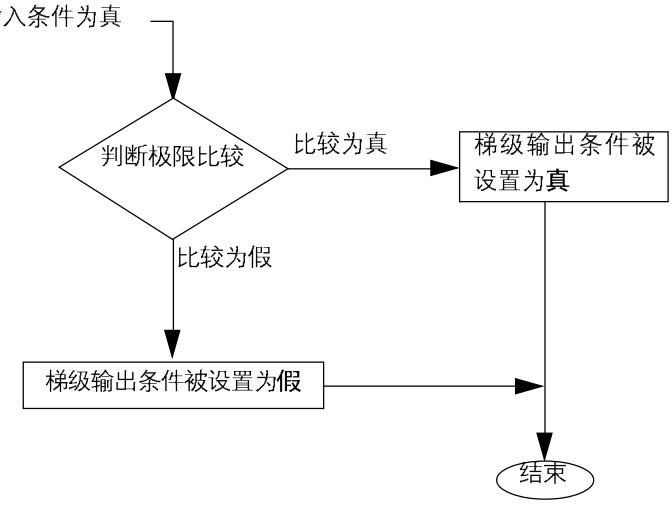
Low Limit ≥ High Limit
如果测试值等于或在下限值和上限值之间则指令为真。



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



If Low Limit ≤ High Limit:

如果测试值:	梯级输出条件为:
等于或在上下限值之间	真
不等于或在极限之外	假

If Low Limit ≥ High Limit:

如果测试值是:	梯级输出条件为:
等于或在极限值之外	真
不等于或在极限之内	假

算术状态标志: 不影响

故障条件: 无

LIM 指令举例:

例 1

Low Limit ⩾ High Limit:

当 0 ⩽ value ⩽ 100 时，接通指示灯 1。

例 2

Low Limit ⩾ High Limit:

当 *value* ⩾ 0 或 *value* ⩽ -100 时，接通指示灯 1。

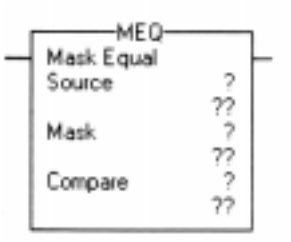
其它格式:

格式:	句法:
neutral 文本	`LIM (low_limit,test,high_limit);`
ASCII 文本	`LIM low_limit test high_limit`

相关指令: CMP

屏蔽等于指令 (MEQ)

MEQ 指令是一条输入指令。



操作数:

操作数：	数据类型:	格式:	说明:
源	SINT INT **DINT**	立即数 标签	与比较值比较的数值。
屏蔽	SINT INT **DINT**	立即数 标签	阻止或通过的位
比较	SINT INT **DINT**	立即数 标签	与源值比较的数值。

说明:

MEQ 指令比较通过屏蔽的源值和比较值的结果。

屏蔽位中的一个 1 意味着数据可以通过，在屏蔽位中的一个 0 意味着位数据被阻止。一般源值，屏蔽值，和比较值用相同数据类型。

如果用户用混合整型数据类型，则指令用 0 值来填充短整型数据的高位，以使它们与大整型数据类型的大小相同。

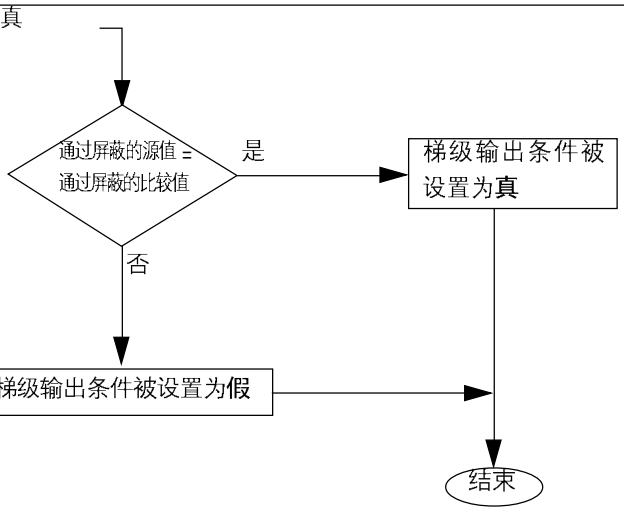
输入立即数作为屏蔽值

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。如果想用其它格式输入屏蔽值，则需要在数值之前用相应的前缀。

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

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	



算术状态标志: 不影响

故障条件: 无

MEQ 指令举例:

例 1

value_1	0	1	0	1	0	1	0	1	1	1	1	1	1	1	1
mask_1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0
通过屏蔽的 *value_1*	0	1	0	1	0	1	0	1	1	1	1	1	x	x	x	x

value_2	0	1	0	1	0	1	0	1	1	1	1	1	0	0	0	0
mask_1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0
通过屏蔽的 *value_2*	0	1	0	1	0	1	0	1	1	1	1	1	x	x	x	x

通过屏蔽的 *value_1* 等于通过屏蔽的，接通则指示灯 1。屏蔽位中的一个 0 值禁止指令比较相应的位(如本例所示的 x)。

例 2

value_1	0	1	0	1	0	1	0	1	1	1	1	1	1	1	1
mask_1	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1
通过屏蔽的 *value_1*	x	x	x	x	x	x	x	x	x	x	x	1	1	1	1

value_2	0	1	0	1	0	1	0	1	1	1	1	1	0	0	0	0
mask_1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0
通过屏蔽的 *value_2*	x	x	x	x	x	x	x	x	x	x	x	0	0	0	0

通过屏蔽的 value_1 不等于通过屏蔽的(value_2)，则关断指示灯 1(light_1)。屏蔽位中的一个 0 值禁止指令比较相应的位(如本例所示的 x)。

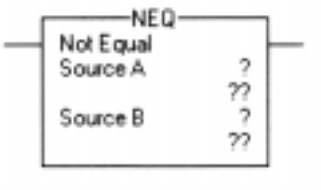
其它格式:

格式:	句法:
neutral 文本	`MEQ (source,mask,compare);`
ASCII 文本	`MEQ source mask compare`

相关指令: CMP，EQU，GEQ，LEQ，MEQ，NEQ

不等于指令(NEQ)

NEQ 指令是一条输入指令。



操作数:

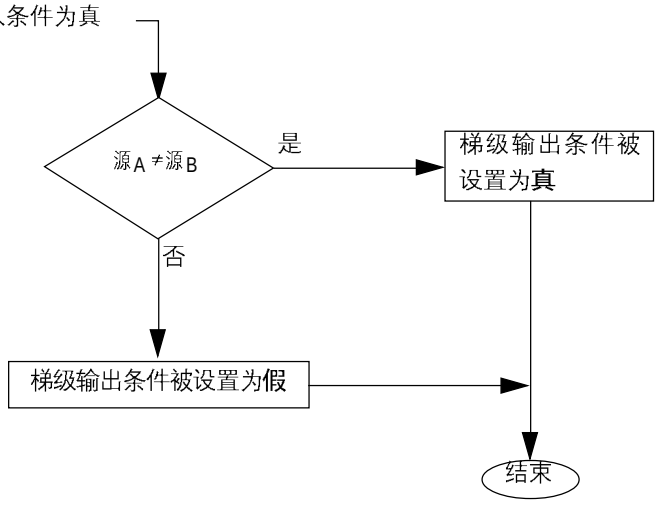
操作数:	数据类型:	格式:	说明:
源 A	SINT	立即数	与源 B 比较的数值
	INT	标签	
	DINT		
	REAL		
源 B	SINT	立即数	与源 A 比较的数值
	INT	标签	
	DINT		
	REAL		

说明:

NEQ 指令测试源 A 的值与源 B 的值是否相等。

执行:

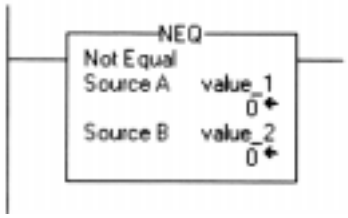
条件:	动作:
预扫描	梯级输出条件被设置为假。
梯级输入条件为假	梯级输出条件被设置为假。
梯级输入条件为真	



算术状态标志: 不影响

故障条件: 无

NEQ 指令举例:



如果 *value_1* 不等于 *value_2*，则梯级输出条件被设置为真。

其它格式:

格式:	句法:
neutral 文本	`NEQ (source_A,source_B);`
ASCII 文本	`NEQ source_A source_B`

相关指令: CMP，EQU，LEQ，GEQ，MEQ

注释:

计算 / 算术指令

(CPT，ADD，SUB，MUL，DIV，SQR，NEG)

简介

计算 / 算术指令用表达式或指定的算术指令计算算术运算的值。

如果用户要:	使用下列指令:	参见页次:
计算表达式	CPT	5-2
两个值相加	ADD	5-5
两个值相减	SUB	5-7
两个值相乘	MUL	5-9
两个值相除	DIV	5-11
计算一个数值的平方根	SQR	5-13
对一个数值取反	NEG	5-14

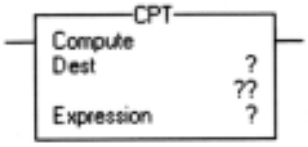
在计算中可以用混合数据类型，但是这样会损失精度，也可能发生取整误差。而且指令执行时间长。检测 S:V 位以确定结果是否被截断。

黑体数据类型表示是最优数据类型。如果指令的所有操作数都使用相同的最优数据类型，则指令执行速度快且占内存少。典型的最优数据类型是 DINT 或 REAL。

在每次梯级扫描时，只要梯级输入条件为真，计算 / 算术指令就执行一次。如果用户希望表达式只计算一次，则可以用一条一次响应指令来触发该指令。

计算指令 (CPT)

CPT 指令是一条输出指令。



操作数:

操作数:	数据类型:	格式:	说明:
目的单元	SINT	标签	存储结果的标签
	INT		
	DINT		
	REAL		
表达式	SINT	立即数	表达式由运算符分开的标签
	INT	标签	/ 立即数组成。
	DINT		
	REAL		

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 指令计算表达式的数值并且存放结果于目的单元内。

与其它算术指令运算相比，CPT 指令的运算速度稍慢而且占用更多的内存。CPT 指令的优点是它允许用户在一条指令内输入复杂的表达式。

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 to y)	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	弧度转换成角度	DINT，REAL

运算符:	说明:	最优数据类型:
FRD	BCD 码转换成整数	DINT
LN	自然对数	REAL
LOG	以 10 为底的对数	REAL
NOT	位补码	DINT
OR	按位 OR	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	DINT
XOR	按位异或	DINT

确定运算顺序

指令按预先规定的顺序，而不必按用户列出的顺序，执行写入表达式的运算。可以通过把分组项组合到圆括号内来改变运算顺序，强制指令在执行其他运算之前执行圆括号内的运算，来改变运算顺序。

同级的运算顺序是从左向右执行。

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

执行：

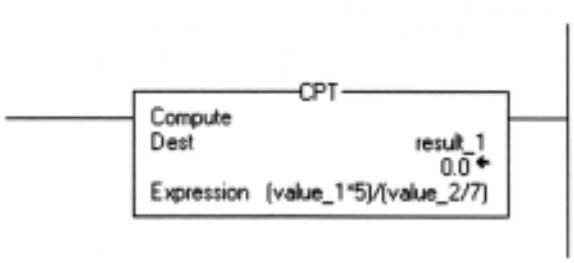
条件：	动作：
预扫描	梯级输出条件被设置为假。
梯级输入条件为假	梯级输出条件被设置为假。
梯级输入条件为真	指令计算表达式并存放结果于目的单元梯级输出条件被设置为真。

算术状态标志： 影响算术状态标志

故障条件： 无

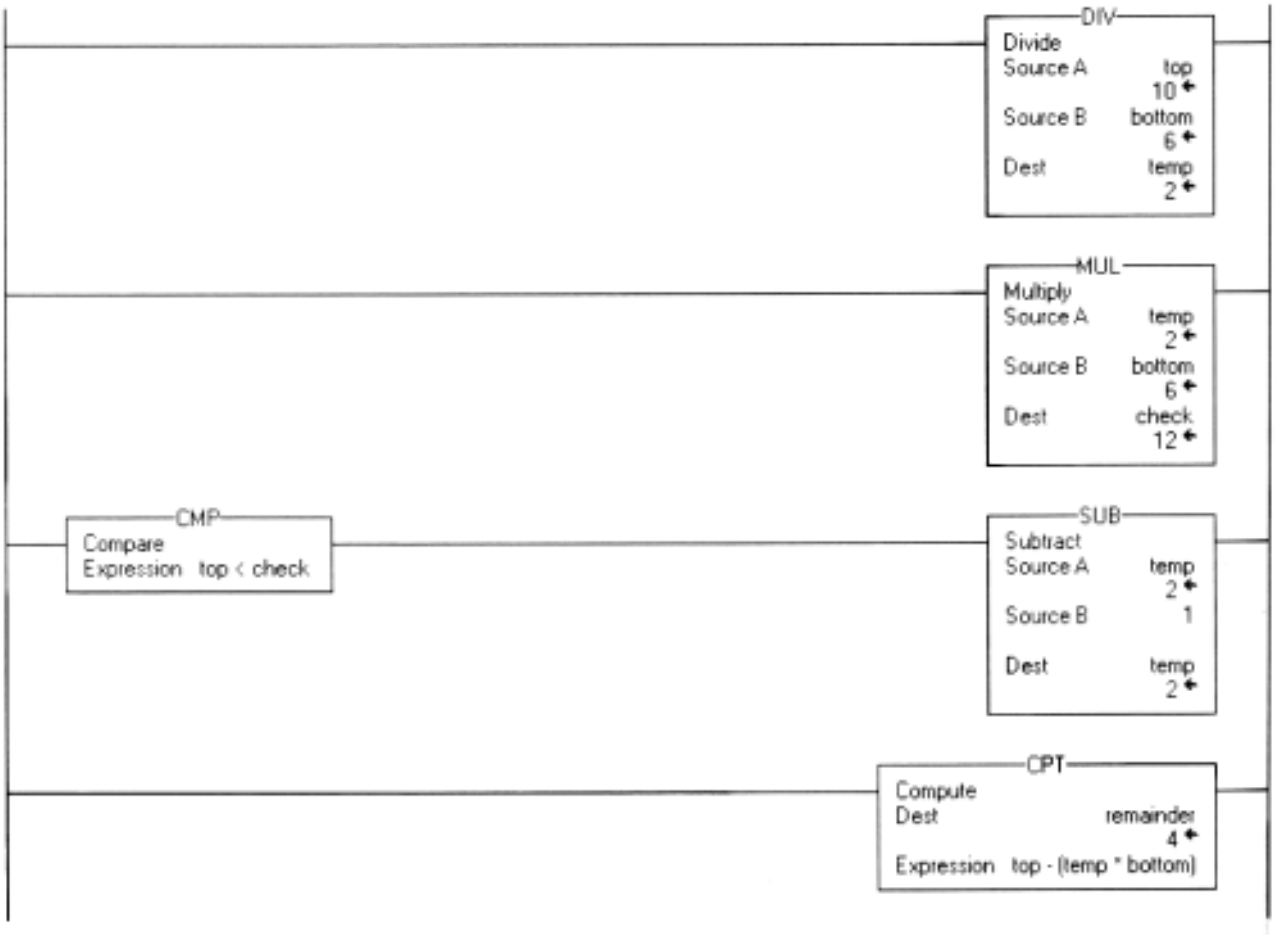
CPT 指令举例：

例 1



当指令被使能时，CPT 指令计算 *value_1* 乘以 5 的结果，
然后此结果被 *value_2* 除以 7 的结果除，并把最后结果存放在 *result_1* 内.

例 2 当指令被使能时，CPT 指令执行模数运算.



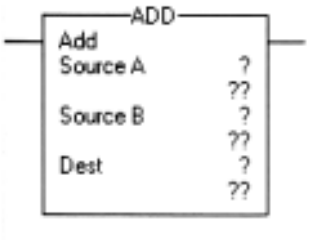
其它格式:

格式:	句法:
neutral 文本	`CPT (destination, expression);`
ASCII 文本	`CPT destination expression`

相关指令: CMP

加法指令 (ADD)

ADD 指令是一条输出指令。



操作数:

操作数:	数据类型:	格式:	说明:
源 A	SINT INT **DINT** **REAL**	立即数 标签	与源 B 操作数相加的值。
源 B	SINT INT **DINT** **REAL**	立即数 标签	与源 A 操作数相加的值。
目的单元	SINT INT **DINT** **REAL**	标签	存放计算结果的标签。

说明:

ADD 指令使源 A 操作数与源 B 操作数相加，并存放计算结果于目的单元内。

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	目的单元 = 源 A + 源 B 梯级输出条件被设置为真。

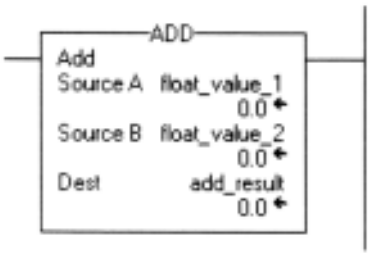
算术状态标志:

影响算术状态标志

故障条件:

无

ADD 指令举例:



当指令使能时，ADD 指令使 *float_value_1* 与 *float_value_2* 相加并存放结果于 *add_result* 内。

其它格式:

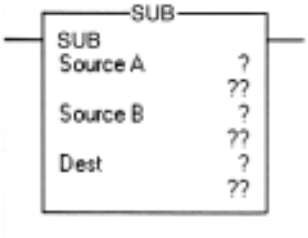
格式:	句法:
neutral 文本	`ADD (source_A,source_B,destination);`
ASCII 文本	`ADD source_A source_B destination`

相关指令:

CPT，DIV，MUL，SUB

减法指令(SUB)

SUB 指令是一条输出指令。



操作数:

操作数:	数据类型:	格式:	说明:
源 A	SINT INT **DINT** **REAL**	立即数 标签	减去源 B 操作数的数值。
源 B	SINT INT **DINT** **REAL**	立即数 标签	从源 A 操作数减去的数值。
目的单元	SINT INT **DINT** **REAL**	标签	存放计算结果的标签。

说明:

SUB 指令使源 A 操作数减去源 B 操作数，并存放结果于目的单元内。

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	目的单元 = 源 A - 源 B 梯级输出条件被设置为真。

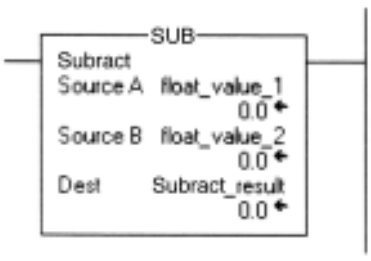
算术状态标志:

影响算术状态标志

故障条件:

无

SUB 指令举例:



当指令使能时，SUB 指令使 *float_value_1* 减去 *float_value_2* 并存放结果于 *subtract_result* 内。

其它格式:

格式:	句法:
neutral 文本	SUB *(source_A,source_B,destination);*
ASCII 文本	SUB *source_A source_B destination*

相关指令:

CPT，ADD，DIV，MUL

乘法指令 (MUL)

MUL 指令是一条输出指令。

操作数:

操作数:	数据类型:	格式:	说明:
源 A	SINT	立即数	被乘数
	INT	标签	
	DINT		
	REAL		
源 B	SINT	立即数	乘数
	INT	标签	
	DINT		
	REAL		
目的单元	SINT	标签	存放结果的标签
	INT		
	DINT		
	REAL		

说明: MUL 指令使源 A 操作数与源 B 操作数相乘，并存放计算结果于目的单元。

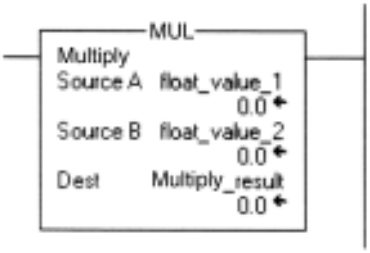
执行:

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	目的单元 = 源 A * 源 B 梯级输出条件被设置为真。

算术状态标志: 影响算术状态标志

故障条件: 无

MUL 指令举例:



当指令使能时，MUL 指令使 *float_value_1* 与 *float_value_2* 相乘并存放结果于 *multiply_result* 内。

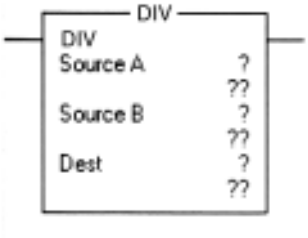
其它格式:

格式:	句法:
neutral 文本	`MUL (source_A,source_B,destination);`
ASCII 文本	`MUL source_A source_B destination`

相关指令: CPT，ADD，DIV，SUB

除法指令(DIV)

DIV 指令是一条输出指令。



操作数:

操作数:	数据类型:	格式:	说明:
源 A	SINT	立即数	被除数值。
	INT	标签	
	DINT		
	REAL		
源 B	SINT	立即数	除数值。
	INT	标签	
	DINT		
	REAL		
目的单元	SINT	标签	存放计算结果的标签。
	INT		
	DINT		
	REAL		

说明:

DIV 指令使源 A 操作数被源 B 操作数除并存放结果于目的单元。

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	目的单元 = 源 A / 源 B 梯级输出条件被设置为真。

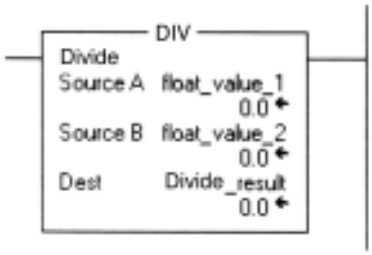
算术状态标志:

影响算术状态标志

故障条件:

无

DIV 指令举例:



当指令使能时，DIV 指令使 *float_value_1* 被 *float_value_2* 除并存放结果于 *divide_result* 内。

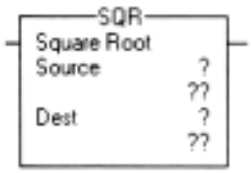
其它格式:

格式:	句法:
neutral 文本	`DIV (source_A,source_B,destination);`
ASCII 文本	`DIV source_A source_B destination`

相关指令: CPT，ADD，MUL，SUB

平方根指令(SQR)

SQR 指令是一条输出指令。



操作数:

操作数:	数据类型:	格式:	说明:
源	SINT INT **DINT** **REAL**	立即数 标签	计算该值的平方根。
目的单元	SINT INT **DINT** **REAL**	标签	存放计算结果的标签。

说明:

SQR 指令计算源操作数的平方根并存放计算结果于目的单元内。

如果源操作数是负数，则指令在计算源操作数的平方根之前先计算其绝对值。

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	$\text{Destination} = \sqrt{\text{Source}}$ 目的单元 = 梯级输出条件被设置为真。

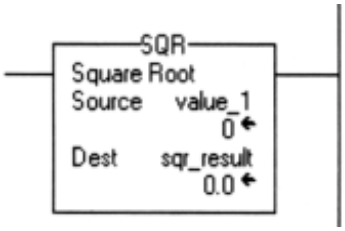
算术状态标志:

影响算术状态标志

故障条件:

无

SQR 指令举例:



当指令被使能时，SQR 指令计算*value_1* 的平方根并存放计算结果于*sqr_result* 内。

其它格式:

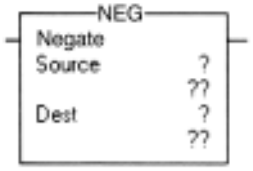
格式:	句法:
neutral 文本	SQR *(source,destination);*
ASCII 文本	SQR *source destination*

相关指令:

CPT，NEG

取反指令(NEG)

NEG 指令是一条输出指令。



操作数:

操作数:	数据类型:	格式:	说明:
源	SINT INT **DINT** **REAL**	立即数 标签	要取反的数值。
目的	SINT INT **DINT** **REAL**	标签	存放计算结果的标签。

说明:

NEG指令改变源操作数的符号并存放结果于目的单元。如果对一个负数取反则结果是正数。如果对正数取反则结果是负数。

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	目的单元 = 0 - 源操作数 梯级输出条件被设置为真。

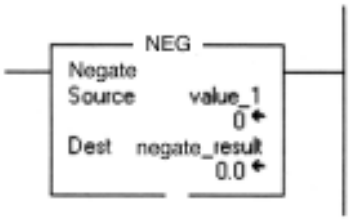
算术状态标志:

影响算术状态标志

故障条件:

无

NEG 指令举例:



当指令被使能时，NEG 指令改变 *value_1* 的符号并存放结果于 *negate_result* 内。

其它格式:

格式:	句法:
neutral 文本	`NEG (source,destination);`
ASCII 文本	`NEG source destination`

相关指令:

CPT，SQR

传送 / 逻辑指令

(MOV，MVM，BTD，CLR，AND，OR，XOR，NOT，)

简介

传送指令改变并且传送数据位。

如果用户要:	使用下列指令:	参见页次:
复制数值	MOV	6-2
复制一个整数的指定部分	MVM	6-3
在整数内或整数之间传送数据位	BTD	6-5
清零一个数值	CLR	6-8

逻辑指令执行位的逻辑运算。

如果用户要:	使用下列指令:	参见页次:
按位逻辑与运算	AND	6-9
按位逻辑或运算	OR	6-11
按位逻辑异或运算	XOR	6-13
按位逻辑非运算	NOT	6-15

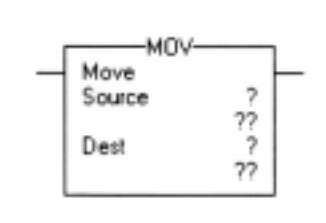
用户可以使用混合数据类型，但是这样会损失精度，也可能发生取整误差，并且会占用更多的时间执行指令。检测 S:V 位以确认结果是否被截断。

黑体字数据类型表示最优数据类型。如果一条指令的所有操作数都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。

只要梯级输入条件为真。则每次扫描传送 / 逻辑指令时指令都执行一次。如果用户希望表达式只计算一次，则可以用一条一次响应指令来触发传送 / 逻辑指令。

传送指令 (MOV)

MOV 指令是一条输出指令。



操作数:

操作数:	数据类型:	格式:	说明:
源	SINT	立即数	被传送(复制)的数值
	INT	标签	
	DINT		
	REAL		
目的单元	SINT	标签	存储结果的标签
	INT		
	DINT		
	REAL		

说明: MOV指令复制源操作数到目的单元。源操作数保持不变。

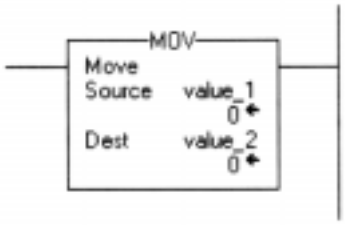
执行:

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	指令复制源操作数到目的单元。 梯级输出条件被设置为真。

算术状态标志: 不影响

故障条件: 无

MOV 指令举例:



当指令被使能时，MOV 指令复制在 *value_1* 内的数据到 *value_2.*

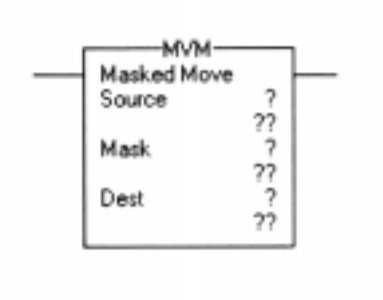
其它格式:

格式:	句法:
neutral 文本	`MOV (source, destination);`
ASCII 文本	`MOV source destination`

相关指令: BTD，CLR，MVM

屏蔽传送指令 (MVM)

MVM 指令是一条输出指令。



操作数:

操作数:	数据类型:	格式:	说明:
源	SINT	立即数	被传送的值
	INT	标签	
	DINT		
屏蔽	SINT	立即数	阻止或通过的位。
	INT	标签	
	DINT		
目的单元	SINT	标签	存储结果的标签
	INT		
	DINT		

说明:

MVM 指令复制源操作数数值到目的单元，并且允许部分数据被屏蔽。
源操作数保持不变。

当指令被使能时，MVM 指令通过屏蔽传送或阻止源数据位。屏蔽位的一个 1 值意味着位数据可以通过。屏蔽位的一个 0 值意味着位数据被阻止。

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

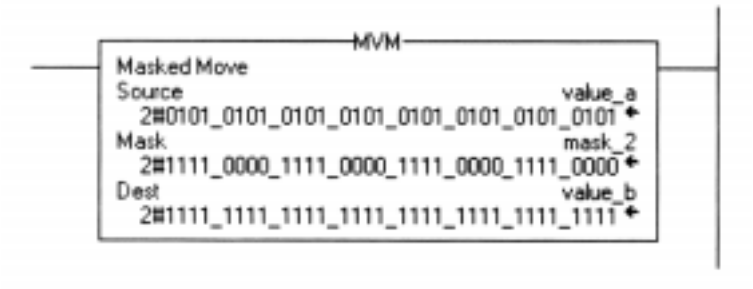
执行:

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	指令使源操作数通过屏蔽复制结果到目的单元。目的单元的被屏蔽位保持不变。 梯级输出条件被设置为真。

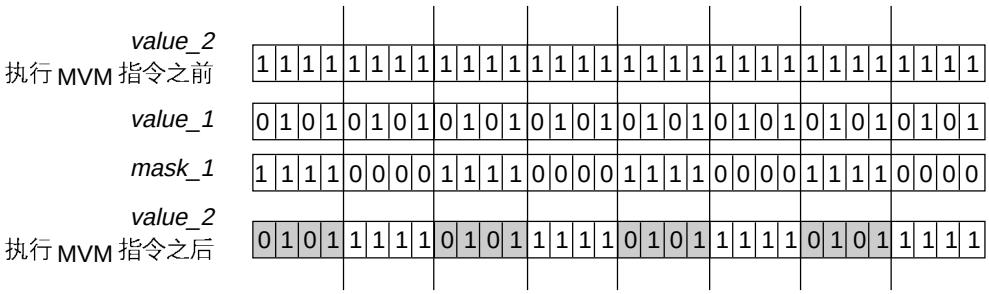
算术状态标志: 影响算术状态标志

故障条件: 无

MVM 指令举例:



当指令被使能时，MVM 指令从 *value_1* 复制数据到 *value_2*，同时允许数据被屏蔽 (屏蔽操作数内各位的一个 0 值屏蔽在 *value_1* 内的位数据)。



阴影部分表示 *value_2* 内的数值被改变。

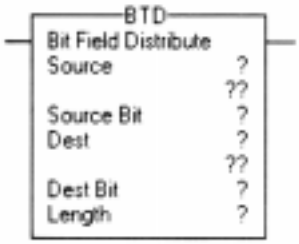
其它格式:

格式:	**句法**:
neutral 文本	`MVM (source,mask,destination);`
ASCII 文本	`MVM source mask destination`

相关指令: BTD，CLR，MOV

位域分配 (BTD)

BTD 指令是一条输出指令。



操作数:

操作数:	数据类型:	格式:	说明:
源	SINT INT **DINT**	立即数 标签	包含要传送数据位的标签。
源位	DINT	立即数 (0-31 DINT) (0-15 INT) (0-7 SINT)	开始传送位的位置号(低位号)必须在源数据类型的有效范围内。
目的	SINT INT **DINT**	标签	传送位的目的单元标签。
目的位	DINT	立即数 (0-31 DINT) (0-15 INT) (0-7 SINT)	从源操作数复制的位在目的单元的起始位号(低位号) 必须在目的操作数数据类型的有效范围内。
长度	DINT	立即数 (1-32)	被传送的位的数量。

说明:

BTD 指令复制源操作数的指定位,并传送这些位到适当的位置,并把这些位写到目的单元内。目的单元内的其余部分保持不变。当指令被使能时,BTD 指令复制来自源操作数的位组到目的单元内。该位组由源位 (位组的低位位号) 和长度 (要复制的位的数量)确定。目的位确定目的单元内开始的低位号。源操作数保持不变。

如果位字段扩展的长度超过目的单元的边界,则指令不保存超出的位。超出的位也不与下个字重叠。

如果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,以使它们与最大数据类型的大小相同。

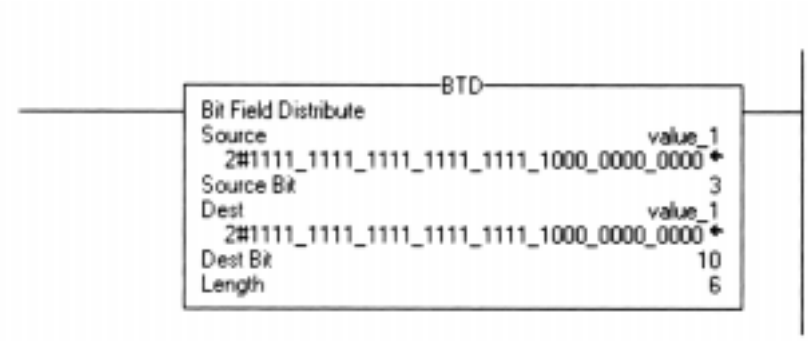
执行:	
条件:	**动作:**
预扫描	梯级输出条件被设置为假
梯级输入条件为假	梯级输出条件被设置为假
梯级输入条件为真	指令复制并且传送源数据位到目的单元。 梯级输出条件被设置为真。

算术状态标志: 影响算术状态标志

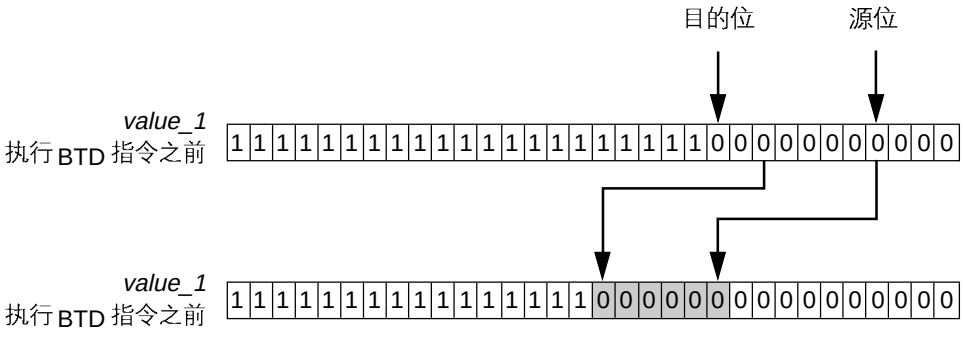
故障条件: 无

BTD 指令举例:

例 1

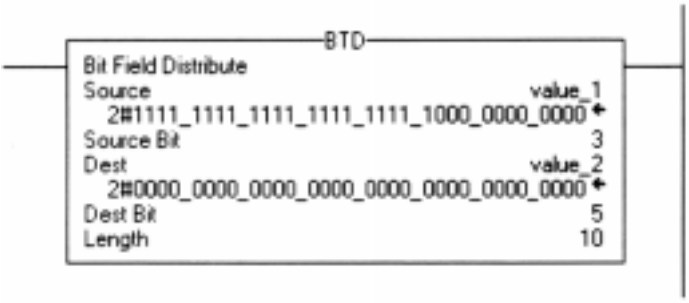


当指令被使能时，指令传送在 *value_1* 内的位。

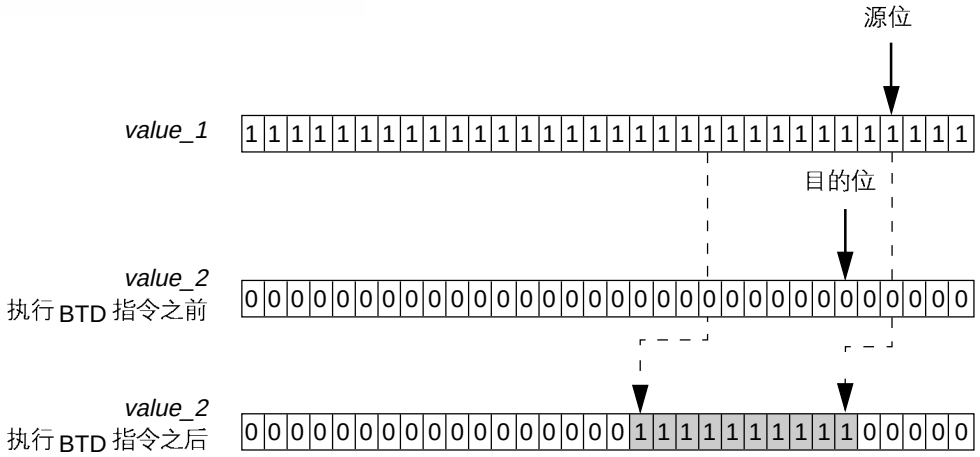


阴影部分表示 *value_1* 内发生变化的位

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 *value_2* 内发生变化的位

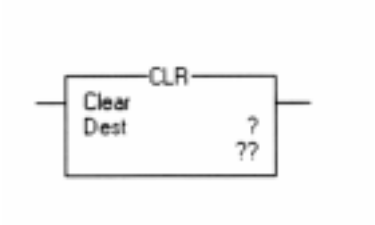
其它格式：

格式：	句法：
neutral 文本	`BTD (source,source_bit,destination,destination_bit,length);`
ASCII 文本	`BTD source source_bit destination destination_bit length`

相关指令： CLR，MOV，MVM

清零指令(CLR)

CLR 指令是一条输出指令。



操作数:

操作数:	数据类型:	格式:	说明:
目的	SINT	标签	被清零数据的标识符
	INT		
	DINT		
	REAL		

说明: CLR 指令清零目的单元的所有位。

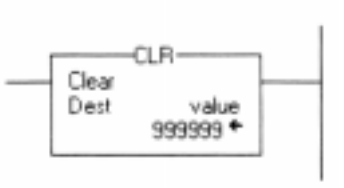
执行:

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	指令清零目的单元内的数据。 梯级输出被件被设置为真。

算术状态标志: 影响算术状态标志

故障条件: 无

CLR 指令举例:



当使能时，CLR 指令清零 *value_1* 内的所有位。

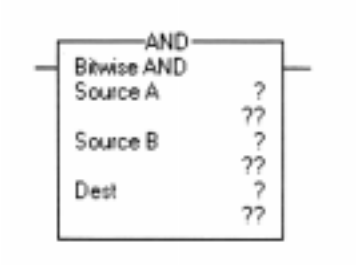
其他格式:

格式:	句法:
neutral 文本	`CLR(destination)`
ASCII 文本	`CLR(destination)`

相关指令: MOV

按位与指令(AND)

AND 指令是一条输出指令。



操作数:

操作数:	数据类型:	格式:	说明:
源 A	SINT INT **DINT**	立即数 标签	与源 B 操作数进行与运算的数值。
源 B	SINT INT **DINT**	立即数 标签	与源 A 操作数进行与运算的数值。
目的	SINT INT **DINT**	标签	存储运算结果的标签。

说明:

AND 指令执行源 A 与源 B 操作数的按位与运算，并存放结果于目的单元。

当指令被使能时，执行逻辑与运算:

如果源 A 的位:	源 B 的位:	目的单元的位是:
0	0	0
0	1	0
1	0	0
1	1	1

如果用户使用混合整型数据类型，则指令用 0 值填充小整数数据的高位，以使其与最大数据类型有相同的大小。

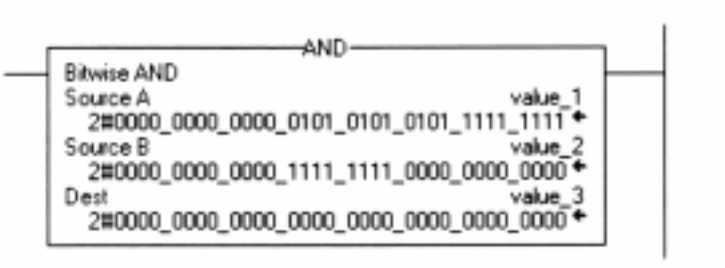
执行:

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	指令执行按位与运算。 梯级输出条件被设置为真。

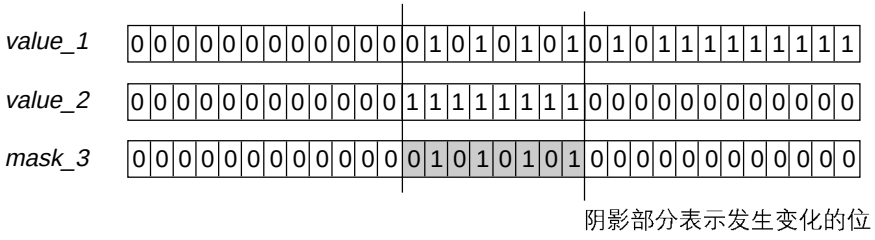
算术状态位: 影响算术状态位

故障条件: 无

AND 指令举例:



当指令被使能时，AND 指令执行 *value_1* 与 *value_2* 的按位与运算，并存放结果于 *value_3* 内。



阴影部分表示发生变化的位

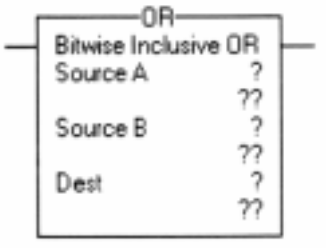
其他格式:

格式:	句法:
neutral 文本	`AND (source_A,source_B,destination);`
ASCII 文本	`AND source_A source_B destination`

相关指令: NOT，OR，XOR

按位或指令 (OR)

OR 指令是一条输出指令。



操作数:

操作数:	数据类型:	格式:	说明:
源 A	SINT INT **DINT**	立即数 标签	与源 B 进行逻辑或运算的数值。
源 B	SINT INT **DINT**	立即数 标签	与源 A 进行或运算的数值。
目的单元	SINT INT **DINT**	标签	存储运算结果的标签。

说明:

逻辑 OR 指令执行源 A 与源 B 操作数的按位或运算，并存放结果于目的单元。

当指令被使能时，执行逻辑或运算:

如果源 A 的位:	源 B 的位是:	目的单元内的位:
0	0	0
0	1	1
1	0	1
1	1	1

如果用户使用混合整型数据类型，则指令用 0 值填充小整数数据的高位，以使其与最大数据类型具有相同的大小。

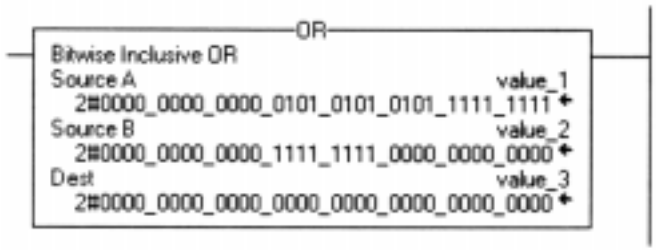
执行:

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	指令执行按位或运算。 梯级输出条件被设置为真。

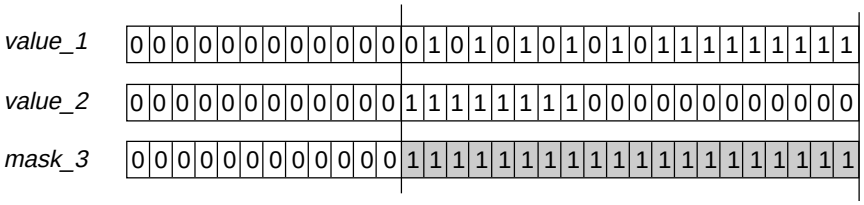
算术状态位: 影响算术状态位

故障条件: 无

OR 指令举例：



当指令被使能时，OR 指令执行 *value_1* 与 *value_2* 的按位或运算，并存放结果于 *value_3* 内。



阴影部分表示发生变化的位

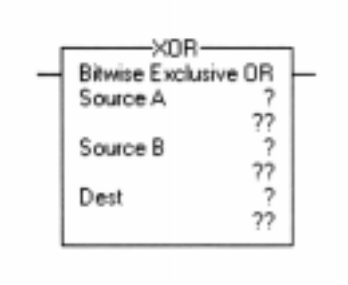
其他格式：

格式：	句法：
neutral 文本	`OR (source_A,source_B,destination);`
ASCII 文本	`OR source_A source_B destination`

相关指令： AND，OR，XOR

按位异或指令(XOR)

XOR 指令是一条输出指令。



操作数:

操作数:	数据类型:	格式:	说明:
源 A	SINT INT DINT	立即数 标签	与源 B 进行异或运算的数值。
源 B	SINT INT DINT	立即数 标签	与源 A 进行异或运算的数值。
目的	SINT INT DINT	标签	存储运算结果的标签。

说明:

XOR 指令执行源 A 与源 B 操作数的按位异或运算，并存放结果于目的单元。

当指令被使能时，执行逻辑异或运算:

如果源 A 的位:	源 B 的位:	则目的单元的位:
0	0	0
0	1	1
1	0	1
1	1	0

如果用户使用混合整型数据类型，则指令用 0 值填充小整数数据的高位，以使其与最大数据类型有相同的大小。

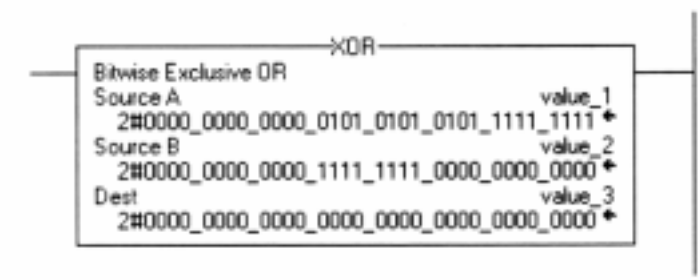
执行:

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	指令执行按位异或运算。 梯级输出条件被设置为真。

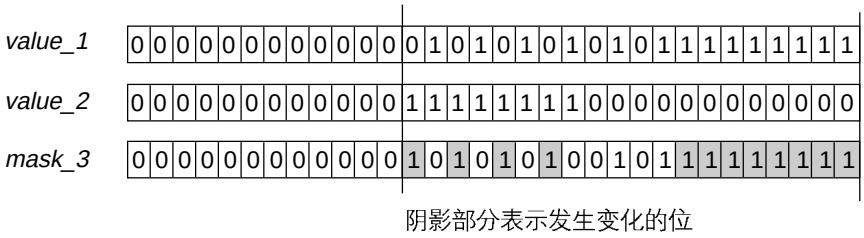
算术状态位: 影响算术状态位

故障条件: 无

XOR 指令举例：



当指令被使能时，XOR 指令执行 *value_1* 与 *value_2* 的按位异或运算，并存放结果于 *value_3* 内。



阴影部分表示发生变化的位

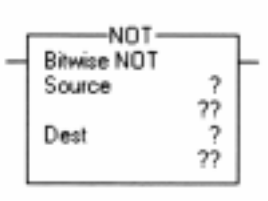
其他格式：

格式：	句法：
neutral 文本	`XOR (source_A,source_B,destination);`
ASCII 文本	`XOR source_A source_B destination`

相关指令： AND，NOT，OR

按位非指令(NOT)

NOT 指令是一条输出指令。



操作数:

操作数:	数据类型:	格式:	说明:
源	SINT	立即数	执行 NOT 运算的值。
	INT	标签	
	DINT		
目的	SINT	标签	存储运算结果的标签。
	INT		
	DINT		

说明:

NOT指令执行源操作数的按位非运算,并存放结果于目的单元内。

当指令被使能时，执行逻辑非运算:

如果源的位是:	目的单元的位是:
0	1
1	0

如果用户使用混合整型数据类型,则指令用 0 值填充小整数数据的高位,以使其与最大数据类型具有相同的大小。

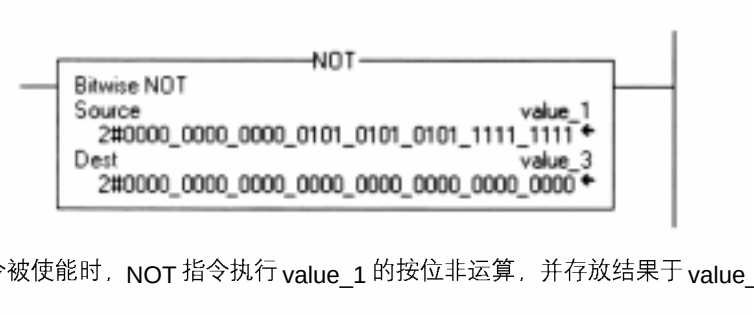
执行:

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	指令执行按位非运算。 梯级输出条件被设置为真。

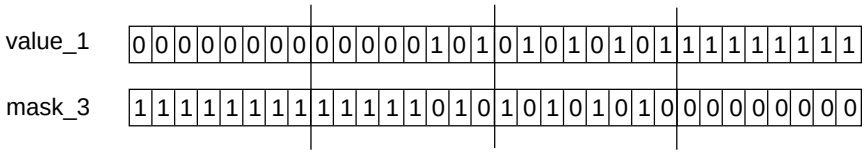
算术状态位: 影响算术状态位

故障条件: 无

NOT 指令举例：



当指令被使能时，NOT 指令执行 value_1 的按位非运算，并存放结果于 value_3 内。



其他格式：

格式：	句法：
neutral 文本	`NOT (source, destination);`
ASCII 文本	`NOT source destination`

相关指令： AND，OR，XOR

数组 (文件)/ 综合指令

(FAL，FSC，COP，FLL，AVE，SRT，STD)

简介

文件 / 综合指令对数据的数组进行操作。

如果用户要:	使用下列指令:	参见页次:
对数组内的数值进行算术，逻辑，移位，和函数运算	FAL	7 - 6
搜索并比较数组内的数值	FSC	7 - 16
复制一个数组内的内容到另一个数组	COP	7 - 25
用指定的数据填充数组	FLL	7 - 29
计算数组内一组数值的平均值	AVE	7 - 32
按上升的顺序排序数组内的一维数据	SRT	7 - 36
计算数组内一组数值的标准偏差	STD	7 - 39

可以使用混合数据类型，但是这样会损失精度，也可能会发生取整误差，而且指令执行时间长。检测 S:V 位以确定结果是否被截断。

黑体字数据类型是最优数据类型。如果指令的所有操作数都使用同一最优数据类型，则指令执行的速度快而且占用内存少。典型最优数据类型是DINT 或 REAL。

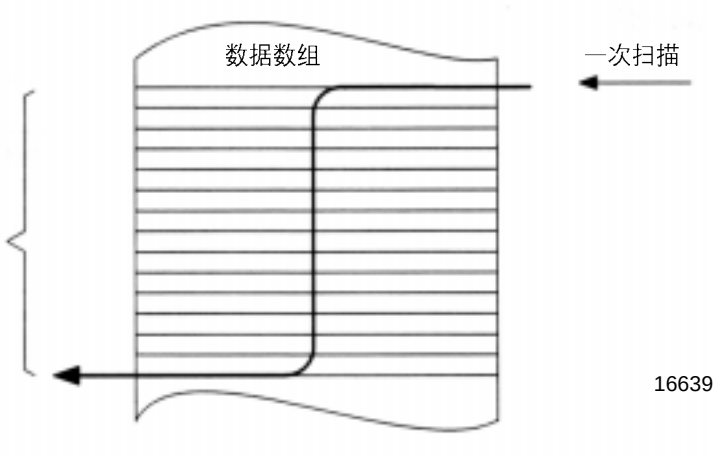
选择操作模式

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。

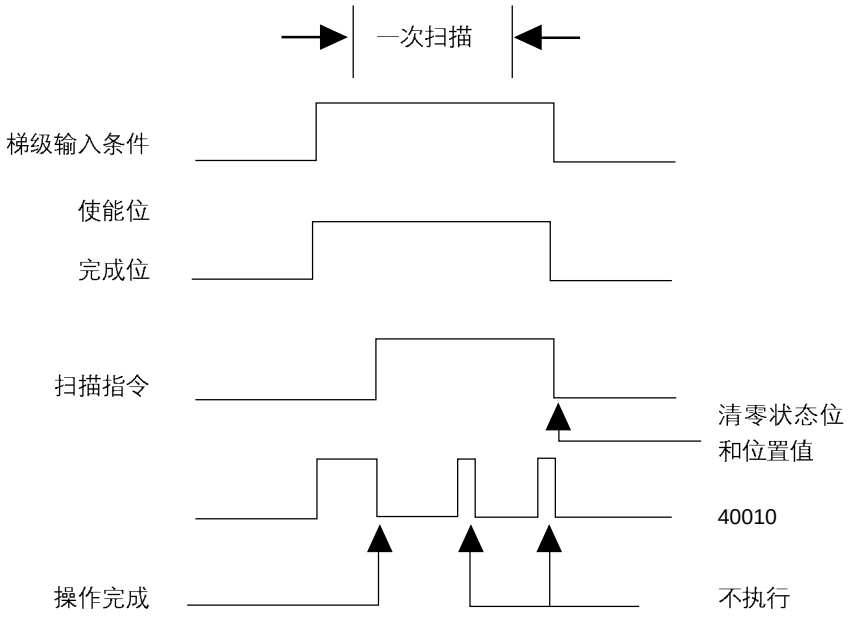
如果用户要:	选择下列模式:
在继续执行下一条指令之前对数组内指定的所有元素进行操作。	整体模式
把对数组内元素的操作分配给多个扫描周期，输入每次扫描要处理的元素数量(1-2147483647)	数值模式
梯级输入条件每由假到真转换一次，只对数组内的一个元素操作。	增量模式

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。



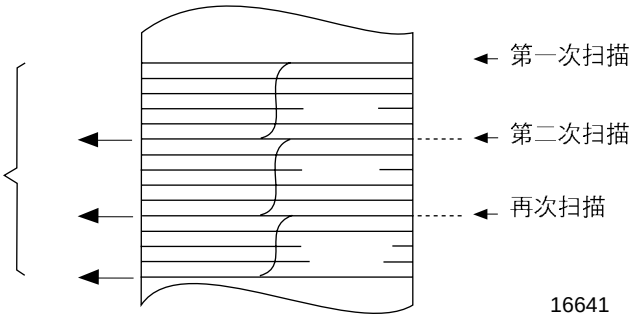
下列时序图说明状态位与指令操作之间的关系。当指令执行完成时，完成位 (.DN) 被置位。如果梯级输入条件为假，则完成位 (.DN) 、使能位 (.EN)、和位置值 (.POS) 被清零。只有梯级输入条件再次由假到真转换时，指令才能再次被触发执行。



数值模式

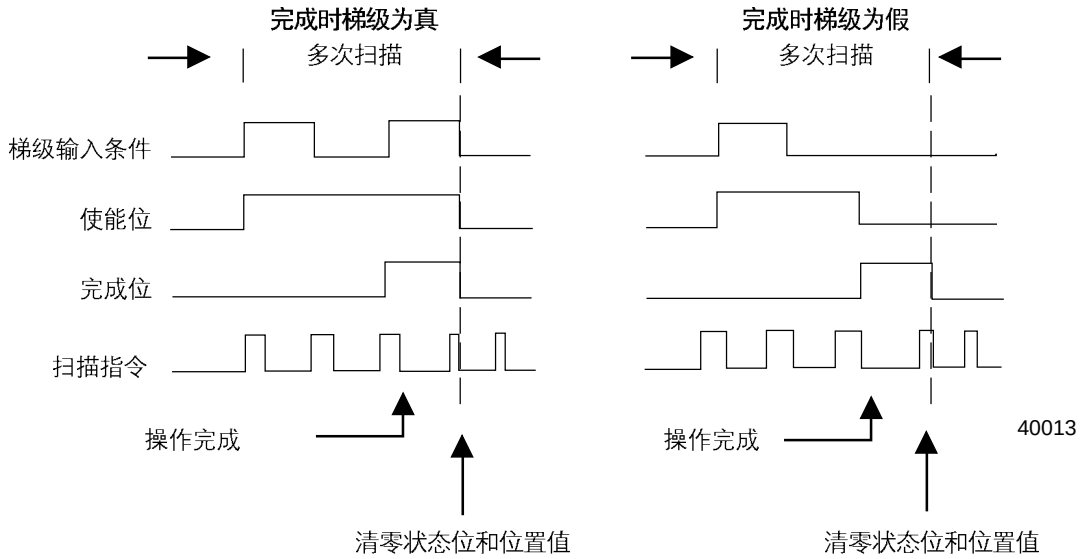
数值模式把对数组的操作分配给多个扫描周期。在对时间要求不很重要的数据或大数据量的应用中，这种模式是非常有用的。输入每次扫描需要处理的元素数量，这样可以缩短扫描时间。

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。一旦指令开始执行，则指令所在梯级输入条件可以反复变化而不中断指令的执行。



重要: 在完成位(.DN)被置位之前，不要使用以数值模式操作的指令文件的结果 。

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)。

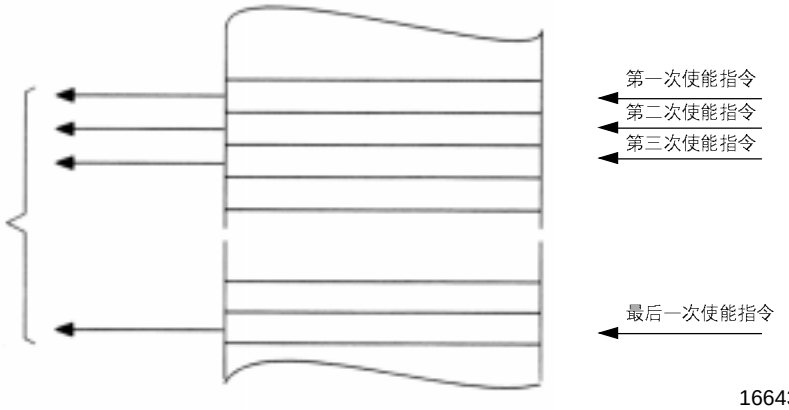


如果在指令完成时，梯级输入条件为真，则在梯级输入条件变为假之前，置位使能位 (.EN) 和完成位 (.DN)。当梯级输入条件变为假时，这些位被清零同时位置值 (.POS) 也被清零。

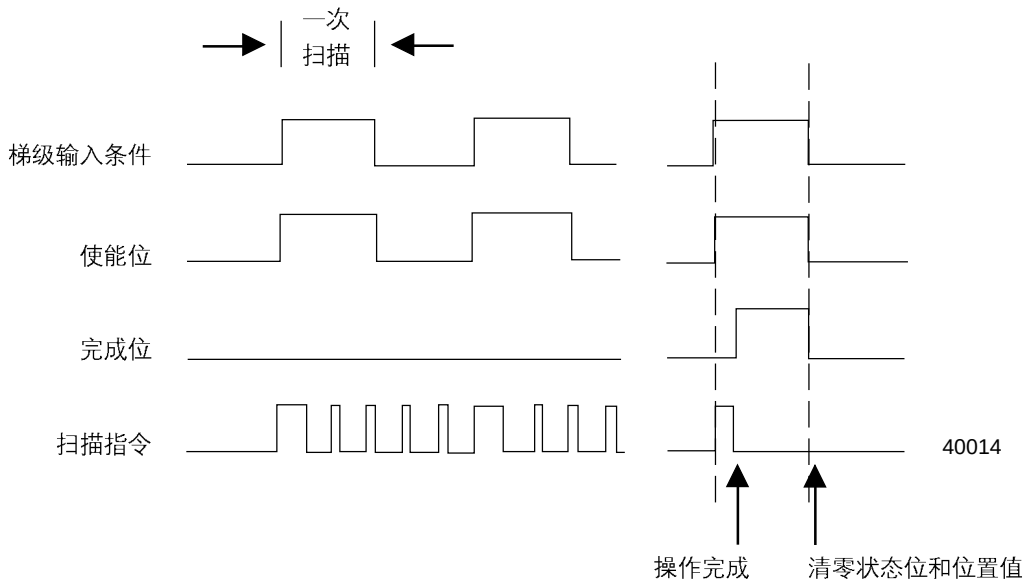
如果在指令完成时梯级输入条件为假，则立即清零使能位 (.EN)。使能位 (.EN) 被清零后的第一次扫描，完成位 (.DN) 和位置 (.POS) 值也被清零。

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。每次扫描时，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大于一个扫描周期，指令也只在第一次扫描期间执行一次。



当梯级输入条件为真时，使能位 (.EN) 被置位。如果处理完数组内的最后一个元素，则完成位 (.DN) 被置位。如果对数组内的最后一个元素的处理已经完成时，梯级输入条件变为假，则使能位 (.EN)，完成位 (.DN)，和位置 (.POS) 值都被清零。

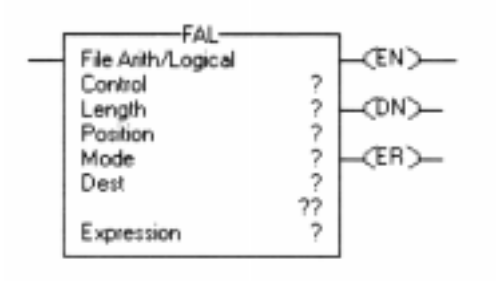
增量模式与数值模式的区别是每次扫描处理元素的数量不同:

- 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.

文件算术与逻辑指令 (FAL)

FAL 指令是一条输出指令。

操作数:



操作数:	数据类型:	格式:	说明:
控制	CONTROL	标签	运算的控制结构体
长度	DINT	立即数	要处理的数组元素数量
位置	DINT	立即数	数组内的当前元素 一般起始值为 0
模式	DINT	立即数	对数组内元素的操作模式, 选择增量(INC)、整体(ALL)、或输入一个数值
目的单元	SINT INT **DINT** **REAL**	标签	存储运算结果的标签
表达式	SINT INT **DINT** **REAL**	立即数 标签	由被运算符分开的标签与 / 或立即数组成的表达式。

控制结构体:

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(.ER) 位之前停止执行指令。位置值(.POS)包含引起溢出的元素的位置。
.LEN	DINT	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 指令执行存储在数组内数据的复制, 算术, 逻辑和函数运算。 FAL 指令执行的关于数组的运算与 CPT 指令执行的元素运算相同。

用位置值(.POS)提供整个数组的顺序级数。参见第 7-13 页开始的程序举例。也可以参见 B-1 页将数组看作元素的集合。

如果表达式中目的单元的下标超出范围, 则FAL指令发生主要故障 (故障类型 4, 代码 20).

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 to y)	DINT，REAL
ACS	反余弦	REAL
AND	按位 AND	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

运算符:	说明:	最优数据类型:
FRD	BCD 码转换成整数	DINT
LN	自然对数	REAL
LOG	以 10 为底的对数	REAL
NOT	取反	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	DINT
XOR	按位异或	DINT

确定运算的顺序

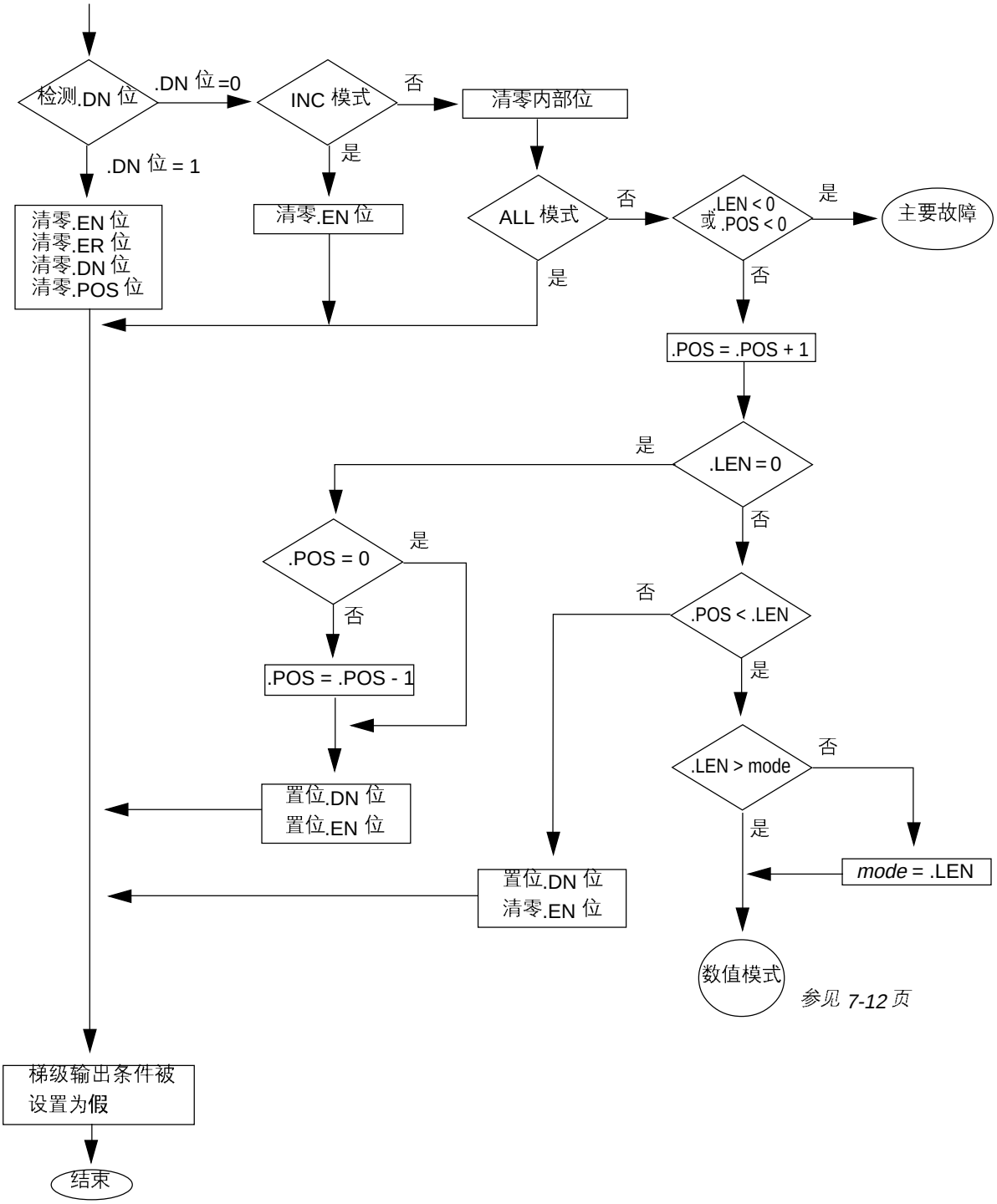
指令按预先规定的顺序，而不是按用户列出的顺序，执行写入表达式的操作。用户可以通过把一些分组项组合到圆括号内来改变运算顺序，强制指令在执行其它运算之前，执行圆括号内的运算，来改变运算顺序。

同级运算的顺序是从左向右执行。

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

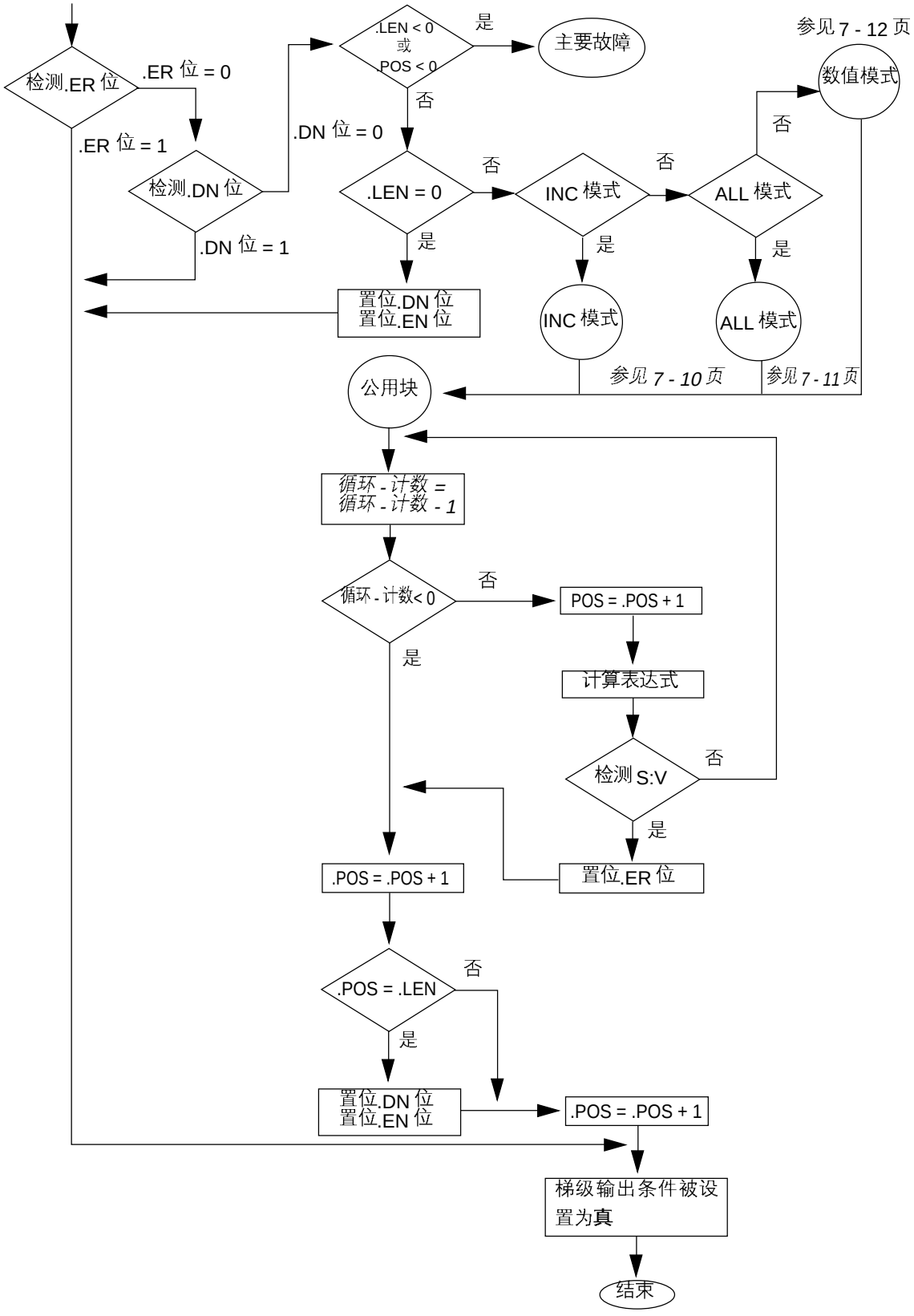
执行:

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	

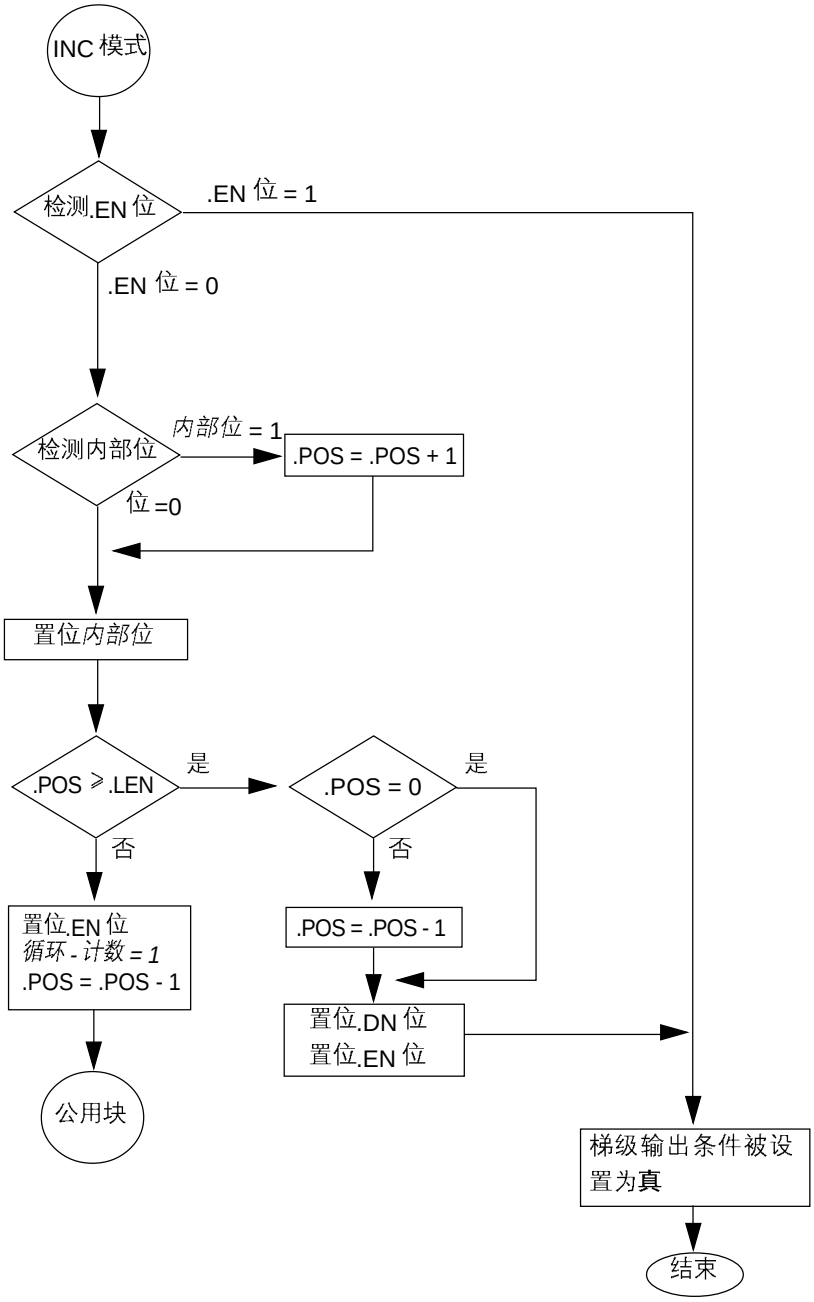


条件: **动作:**

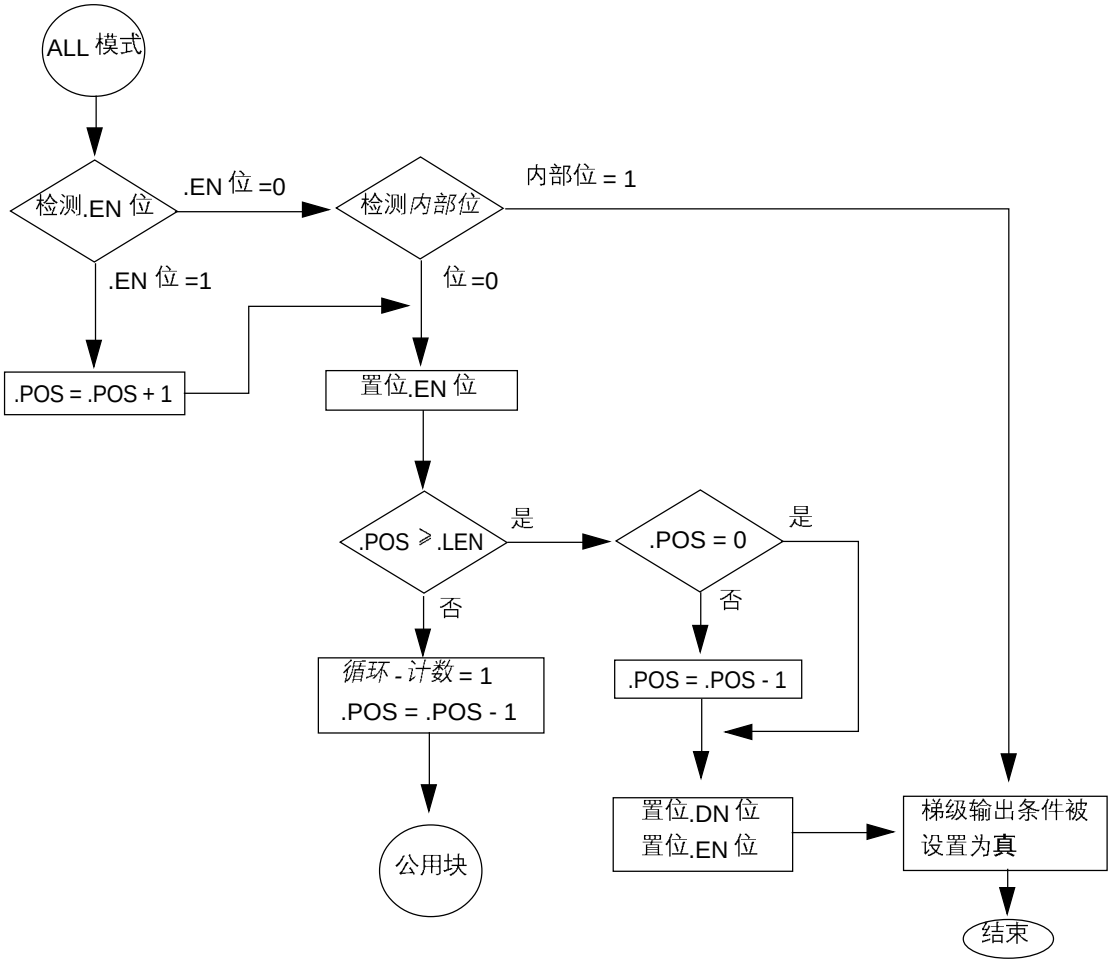
梯级输入条件为真



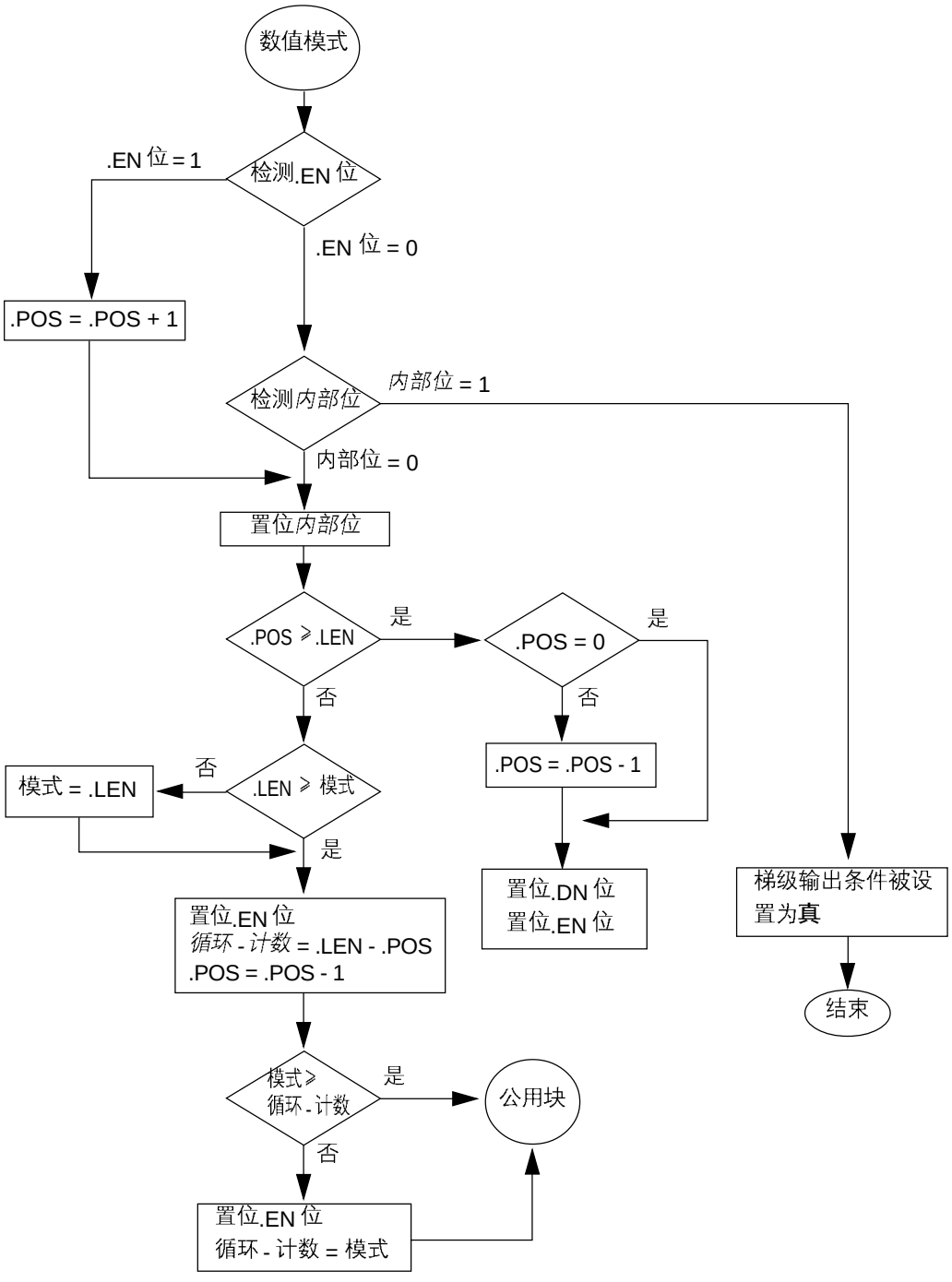
条件: 动作:


INC 模式
检测.EN 位
.EN 位 = 1
.EN 位 = 0
检测内部位
内部位 = 1
.POS = .POS + 1
位 =0
置位内部位
.POS ≥ .LEN
是
.POS = 0
是
否
否
置位.EN 位
循环 - 计数 = 1
.POS = .POS - 1
.POS = .POS - 1
置位.DN 位
置位.EN 位
公用块
梯级输出条件被设置为真
结束

条件: 动作:


ALL 模式
检测.EN 位
.EN 位 =0
检测内部位
内部位 = 1
.EN 位 =1
位 =0
.POS = .POS + 1
置位.EN 位
.POS ⩾ .LEN
是
.POS = 0
是
否
否
循环 - 计数 = 1
.POS = .POS - 1
.POS = .POS - 1
公用块
置位.DN 位
置位.EN 位
梯级输出条件被
设置为真
结束

条件: 动作:


数值模式
检测.EN 位
.EN 位 = 1
.EN 位 = 0
.POS = .POS + 1
检测内部位
内部位 = 1
内部位 = 0
置位内部位
.POS ≥ .LEN
是
.POS = 0
是
否
否
模式 = .LEN
否
.LEN ≥ 模式
.POS = .POS - 1
是
置位.DN 位
置位.EN 位
梯级输出条件被设
置为真
置位.EN 位
循环 - 计数 = .LEN - .POS
.POS = .POS - 1
结束
模式≥
循环 - 计数
是
公用块
否
置位.EN 位
循环 - 计数 = 模式

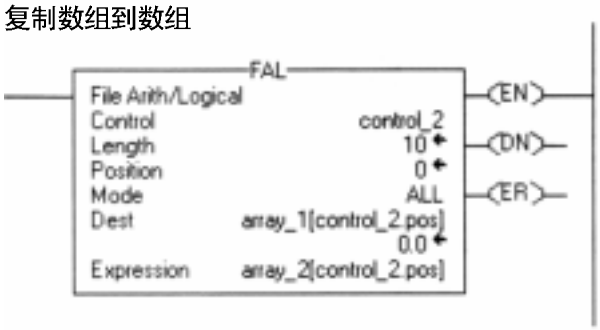
算术状态标志: 影响算术状态标志

故障条件:

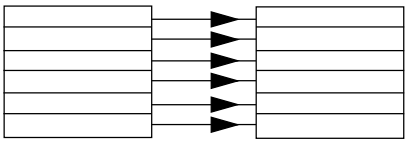
发生主要故障的条件:	故障类型:	故障代码:
下标超出范围	4	20
位置值 < 0 或 长度值 < 0	4	21

FAL 指令举例:

复制数组到数组



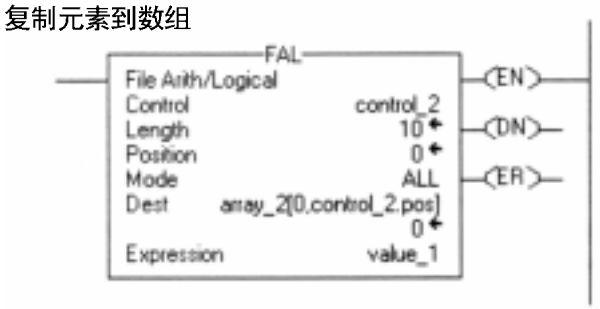
当指令被使能时，FAL指令复制 *array_2* 内的每个元素到 *array_1* 内的相同位置



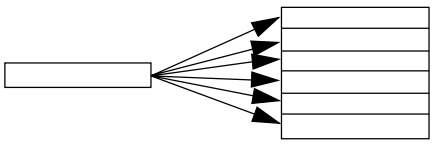
表达式:
array_2 [control_2.pos]

目的单元:
array_1 [control_2.pos]

复制元素到数组



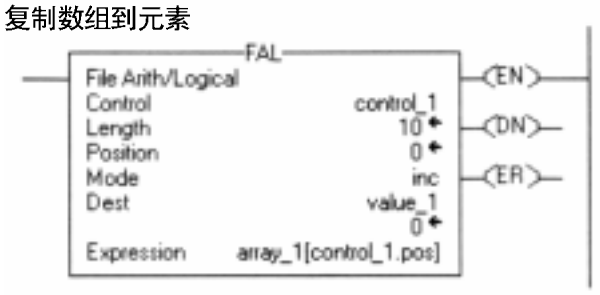
当指令被使能时，FAL指令复制 *value_1* 到 *array_2* 的第二维的前 10 个位置。



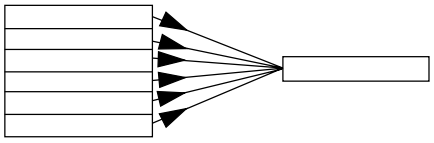
表达式:
value_1

目的单元:
array_2 [0，control_2.pos]

复制数组到元素



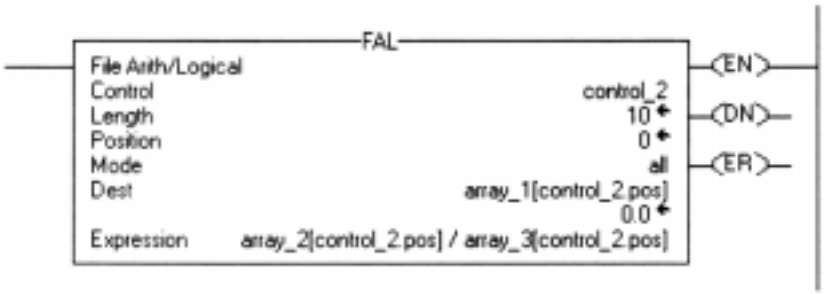
每次FAL指令被使能时，它都复制 *array_1* 的当前值到 *value_1* 内。因为FAL指令用增量模式，所以每次指令被使能时，只复制数组内的一个值，指令用 *array_1* 内的下一个数值覆盖 *value_1* 的当前值。



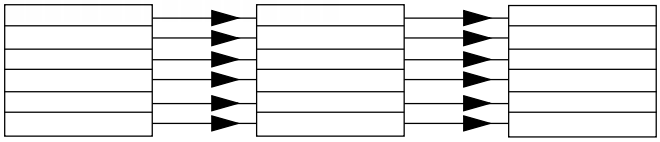
表达式:
array_1[control_1.pos]

目的单元:
value_1

算术运算: (数组 / 数组)到数组



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，并存放结果于 *array_1* 的当前位置。



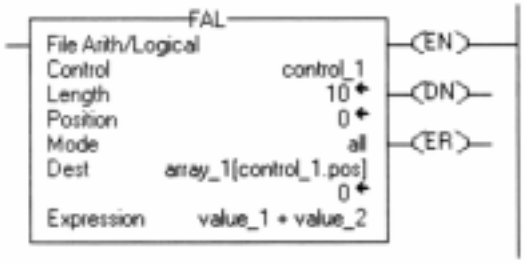
表达式:

array_2[control_2.pos] / array_3 [control_2.pos]

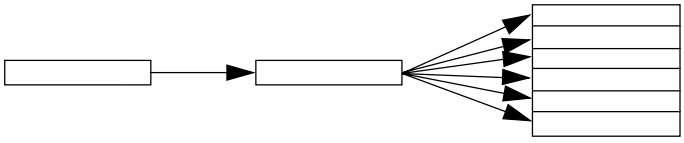
目的单元:

array_1[control_2.pos]

算术运算: (元素 + 元素)到数组



当指令被使能时，FAL 指令使 *value_1* 与 *value_2* 相加并存放结果于 array_1 的当前位置。



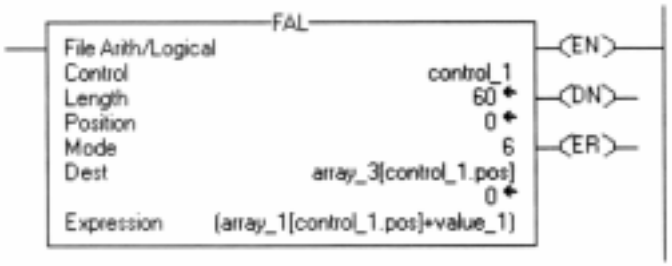
表达式:

value_1 + value_2

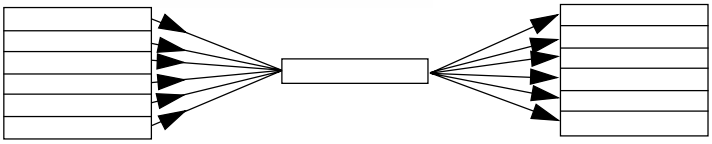
目的单元:

array_1[control_1.pos]

算术运算: (数组 + 元素)到数组



当指令被使能时，FAL 指令使 *array_1* 当前位置的数值与 *value_1* 相加并存放结果于 *array_3* 的当前位置。 如果要对 *array_1* 和 *array_3* 内的所有值进行操作，则指令必须执行 10 次。



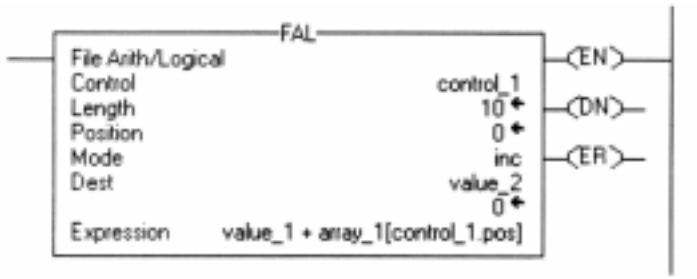
表达式:

array_1[control_1.pos] + value_1

目的单元:

array_3[control_1.pos]

算术运算: (元素 + 数组)到元素

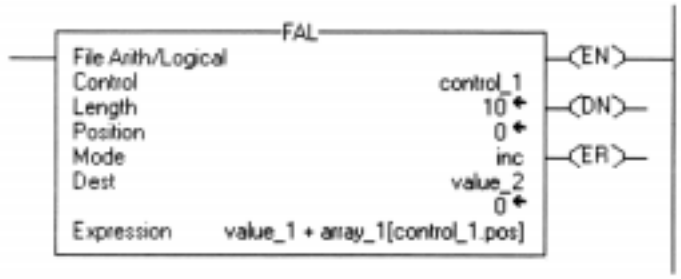


每次 FAL 指令被使能时，它使 *value_1* 与 *array_1* 的当前数值相加并存放结果于 *value_2*。FAL 指令使用增量模式，所以每次指令被使能，只有一个数组内的数值与 *value_1* 相加。下一次指令被使能时，其运算结果覆盖 *value_2*。

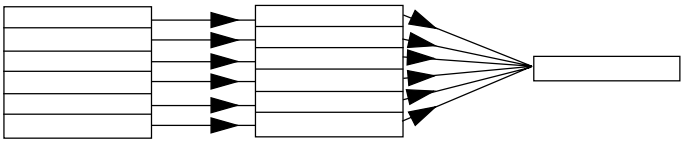
表达式:
value_1 + array_1[control_1.pos]

目的单元:
value_2

算术运算: (数组 * 数组)到元素



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，并存放结果于 *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*。



表达式:
*array_1[control_1.pos] * array_3 [control_1.pos]*

目的单元:
value_1

其它格式:

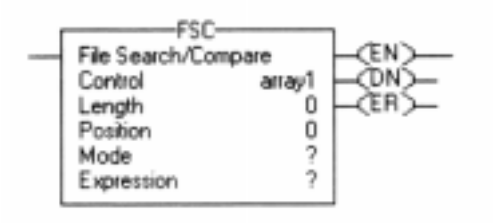
格式:	句法:
neutral 文本	FAL *(control,length,position,mode,destination,expression);*
ASCII 文本	FAL *control length position mode destination expression*

相关指令: CPT，CMP，FSC

文件搜索和比较指令(FSC)

FSC 指令是一条输出指令。

操作数:



操作数:	数据类型:	格式:	说明:
控制	CONTROL	标签	操作的控制结构体
长度	DINT	立即数	要处理的数组内元素的数量
位置	DINT	立即数	数组内的偏移量 一般初始值为 0
模式	DINT	立即数	对数组内元素的操作模式，选择增量(INC)、整体(ALL)、或输入一个数值
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。

控制结构体:

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	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值。

说明:

FSC 指令一个元素一个元素的比较数组内的值。对于逻辑运算，用户必须在表达式内说明。*参见* B-1 *页将数组看作元素的集合。*

如果 FSC 指令被使能，而且比较结果为真，则指令置位发现位 (.FD) 及其位置值 (.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。

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 to y)	DINT，REAL
ACS	反余弦	REAL
AND	按位 AND	DINT
ASN	反正弦	REAL

运算符:	说明:	最优数据类型:
ATN	反正切	REAL
COS	余弦	REAL
DEG	弧度转换成角度	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
NOT	取反	DINT
OR	按位 OR	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	转换成 BCD 码	DINT
XOR	按位异或	DINT

确定运算的顺序

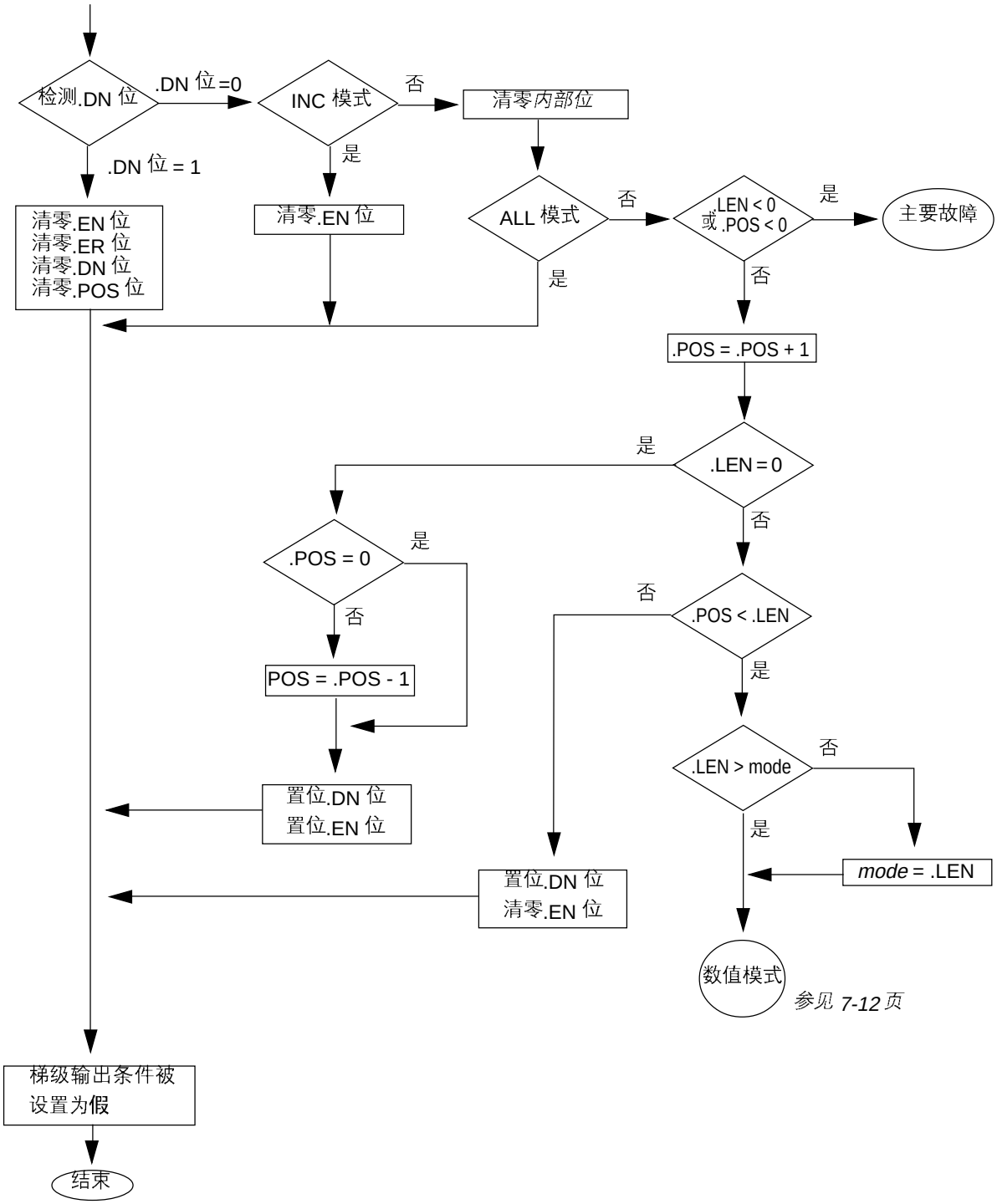
指令按预先规定的顺序，而不必按列出的顺序，执行用户写入表达式的操作。用户可以通过把一组项组合到括号中，来强制指令在执行其它运算之前，执行括号内的运算，来改变运算顺序。

同级运算的顺序是从左向右执行。

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

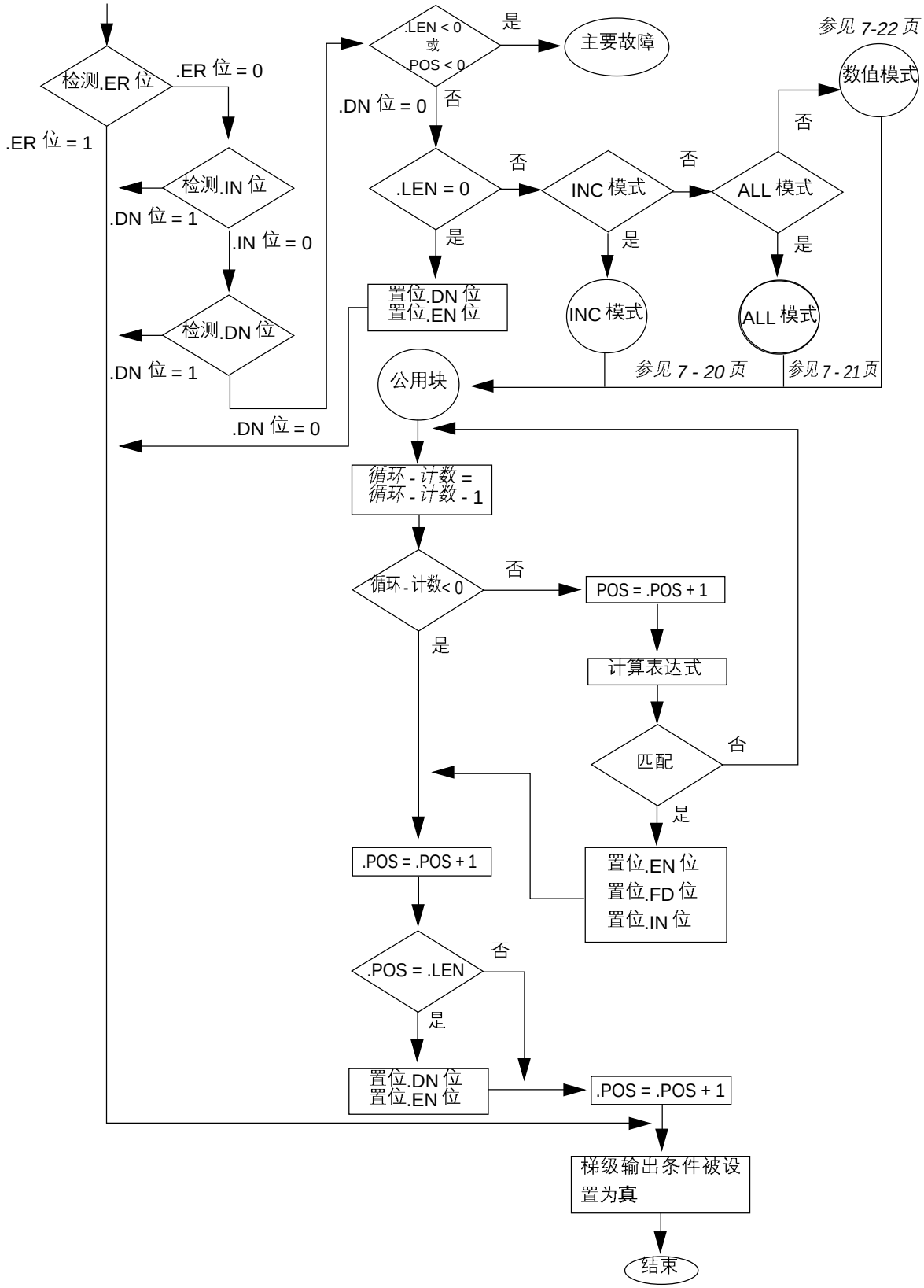
执行:

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	

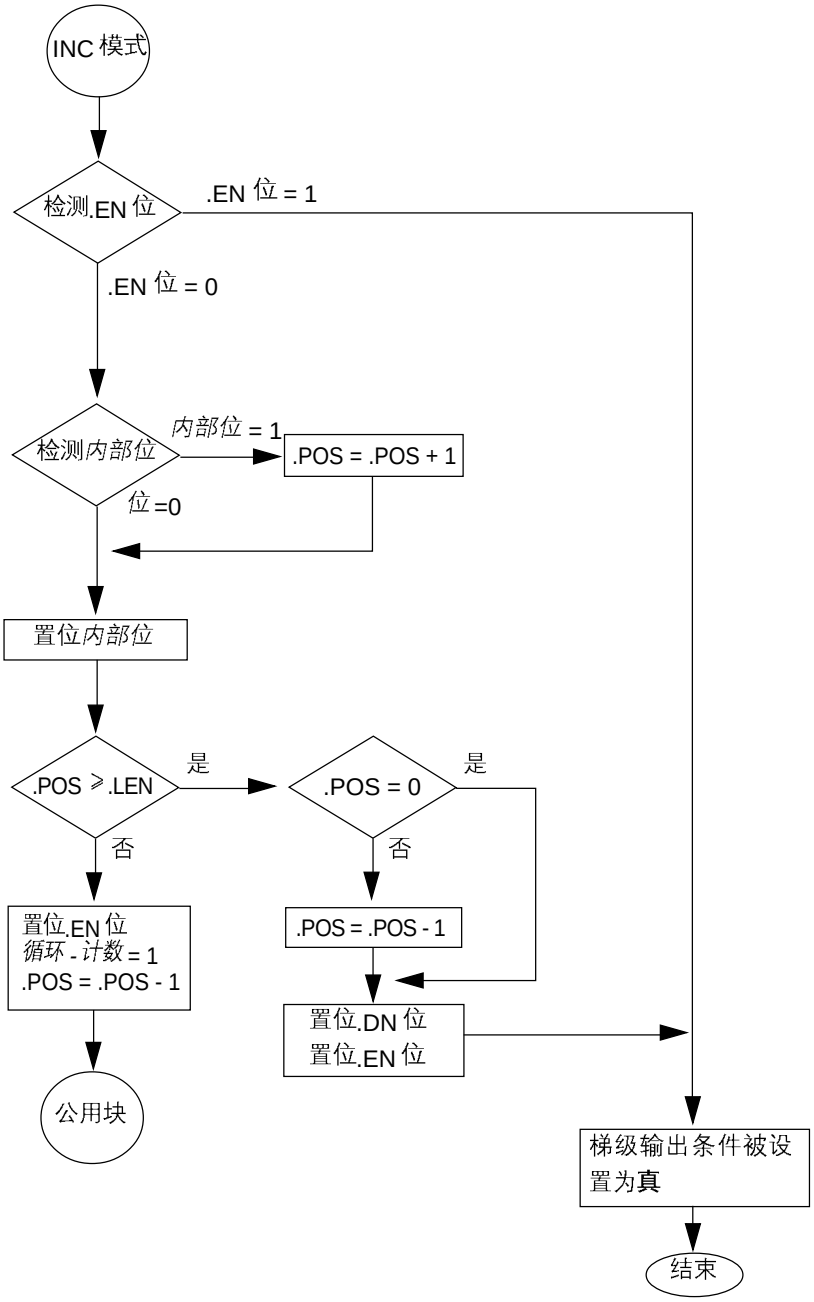


条件: **动作:**

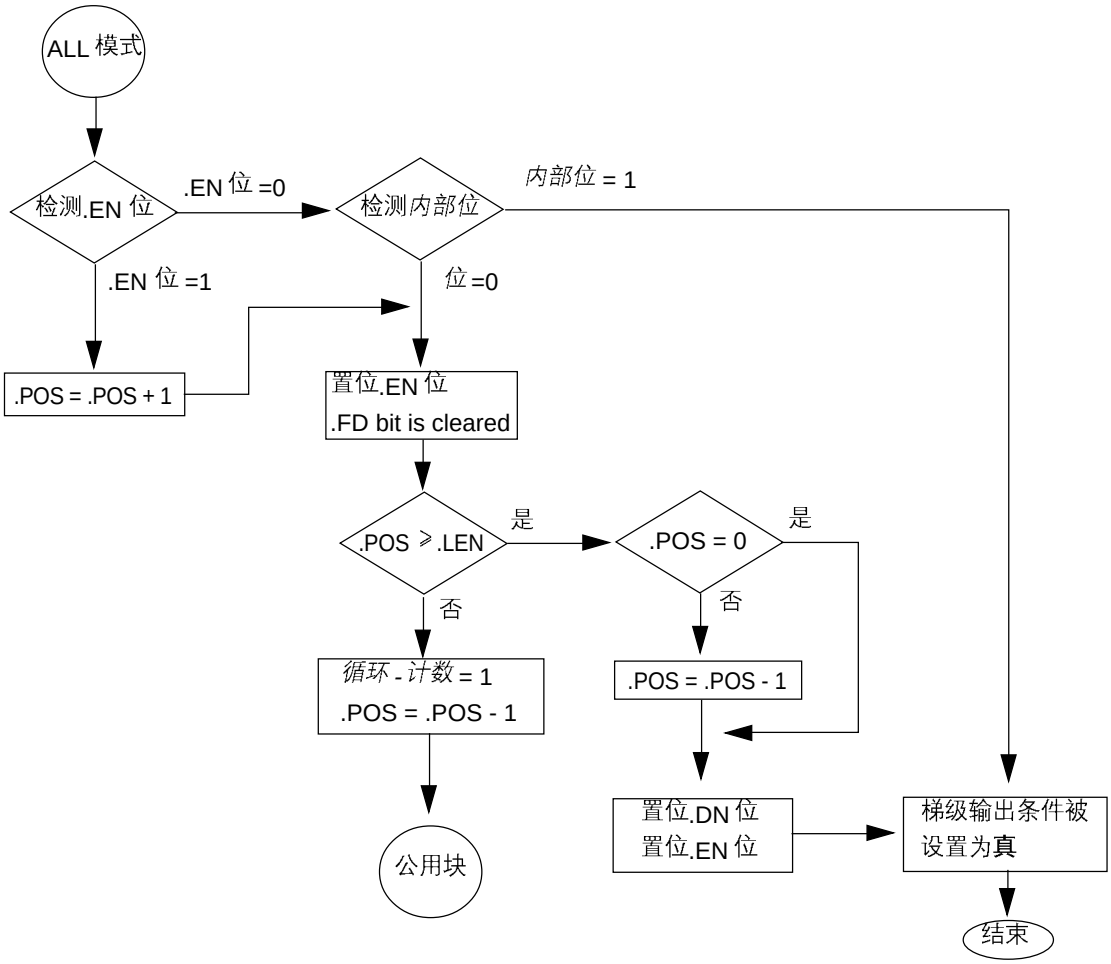
梯级输入条件为真



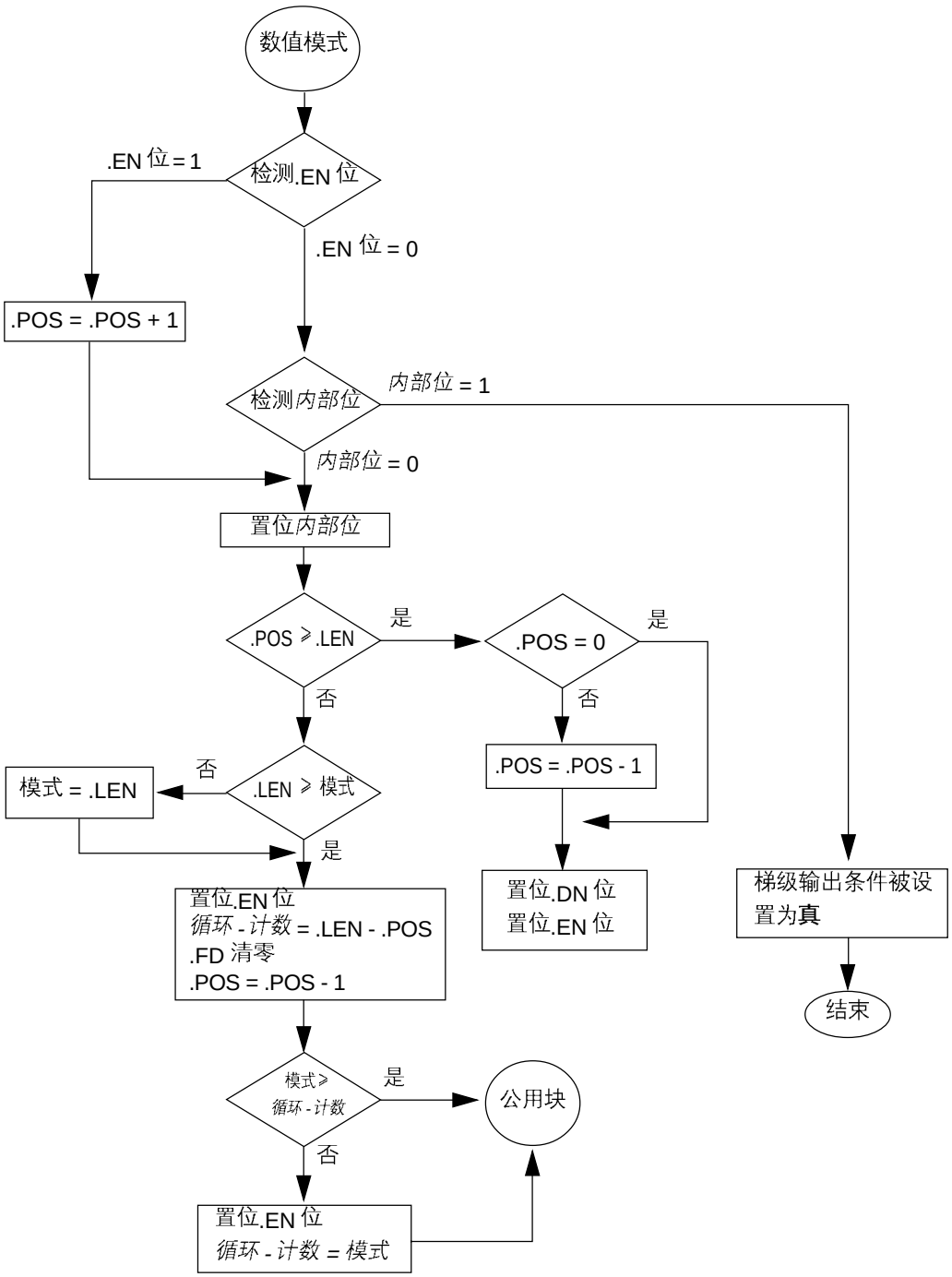
条件: 动作:


INC 模式
检测.EN 位
.EN 位 = 1
.EN 位 = 0
检测内部位
内部位 = 1
.POS = .POS + 1
位 =0
置位内部位
.POS ≥ .LEN
是
.POS = 0
是
否
否
置位.EN 位
循环 - 计数 = 1
.POS = .POS - 1
.POS = .POS - 1
置位.DN 位
置位.EN 位
公用块
梯级输出条件被设置为真
结束

条件: 动作:


ALL 模式
检测.EN 位
.EN 位 =0
检测内部位
内部位 = 1
.EN 位 =1
位 =0
.POS = .POS + 1
置位.EN 位
.FD bit is cleared
.POS ≥ .LEN
是
.POS = 0
是
否
否
循环 - 计数 = 1
.POS = .POS - 1
.POS = .POS - 1
公用块
置位.DN 位
置位.EN 位
梯级输出条件被
设置为真
结束

条件: 动作:


数值模式
检测.EN 位
.EN 位 = 1
.EN 位 = 0
.POS = .POS + 1
检测内部位
内部位 = 1
内部位 = 0
置位内部位
.POS ≥ .LEN
是
.POS = 0
是
否
否
模式 = .LEN
否
.LEN ≥ 模式
.POS = .POS - 1
是
置位.EN 位
循环 - 计数 = .LEN - .POS
.FD 清零
.POS = .POS - 1
置位.DN 位
置位.EN 位
梯级输出条件被设置为真
结束
模式≥
循环 - 计数
是
公用块
否
置位.EN 位
循环 - 计数 = 模式

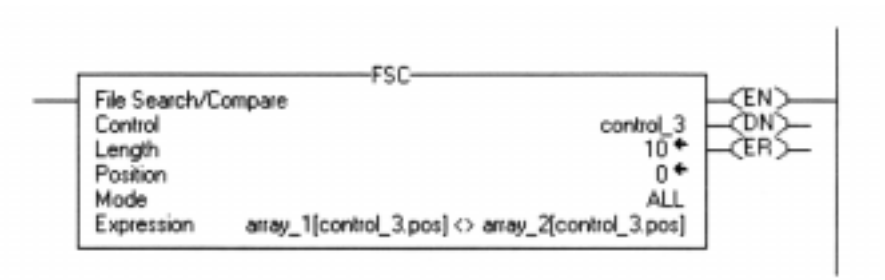
算术状态标志: 影响算术状态标志

故障条件:

发生主要故障条件:	故障类型:	故障代码:
位置值 < 0 或长度值 < 0	4	21

FSC 指令举例:

例 1
搜索两个数组内的匹配数据

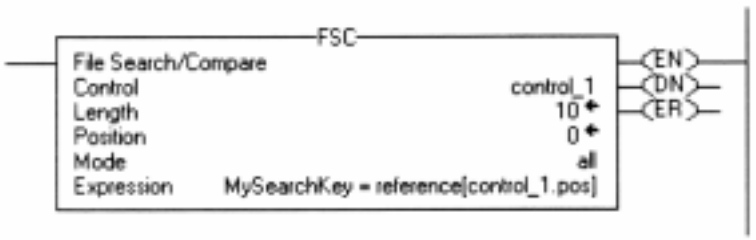


当指令被使能时，FSC 指令比较 *array_1* 内的前 10 个元素与 *array_2* 内的相应元素。

array_1:	*array_2:*	*control_3.pos*	
00000000000000000000000000000000	00000000000000000000000000000000	0	
00000000000000000000000000000000	00000000000000000000000000000000	1	
00000000000000000000000000000000	00000000000000000000000000000000	2	
00000000000000000000000000000000	00000000000000000000000000000000	3	→ 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。如果要继续比较其余数组，需要清零禁止位(.IN)。
00000000000000001111111111111111	11111111111111110000000000000000	4	
11111111111111111111111111111111	11111111111111111111111111111111	5	
11111111111111111111111111111111	11111111111111111111111111111111	6	
11111111111111111111111111111111	11111111111111111111111111111111	7	
11111111111111111111111111111111	11111111111111111111111111111111	8	
11111111111111111111111111111111	11111111111111111111111111111111	9	

例 2

搜索数组内的匹配数据



当指令被使能时，FSC 指令使 *MySearchKey* 的值分别与 *array_1* 内的 10 个元素比较。

MySearchKey :

11111111111111110000000000000000

reference :

00000000000000000000000000000000	0
00000000000000000000000000000000	1
00000000000000000000000000000000	2
00000000000000000000000000000000	3
11111111111111110000000000000000	4
11111111111111111111111111111111	5
11111111111111111111111111111111	6
11111111111111111111111111111111	7
11111111111111111111111111111111	8
11111111111111111111111111111111	9

control_3.pos

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 (.POS) (4) 值表明匹配元素所在的位置。如果要继续比较其余数组，需要清零 .IN 位。

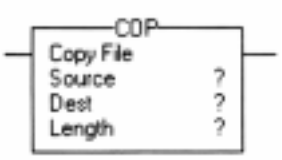
其他格式:

格式:	句法:
neutral 文本	FSC *(control,length,position,mode,expression);*
ASCII 文本	FSC *control length position mode expression*

相关指令: CMP，CPT，FAL

文件复制指令(COP)

COP 指令是一条输出指令。



操作数:

操作数:	数据类型:	格式:	说明:
源	SINT INT DINT REAL 结构体	标签	要复制的起始元素 重要: 源和目的操作数的数结构体据类型应该相同, 否则将发生不可预料的结果。
目的单元	SINT INT DINT REAL 结构体	标签	被源操作数覆盖的起始元素 重要 源和目的操作数的数据类型应该相同, 否则将发生不可预料的结果。
长度	DINT	立即数 标签	被复制到目的单元的元素数量

说明:

COP 指令复制源操作数的数值到目的单元。源操作数保持不变。复制数值的字节数量如下所示:

字节数量 = 长度 * (目的单元数据类型的字节数)

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

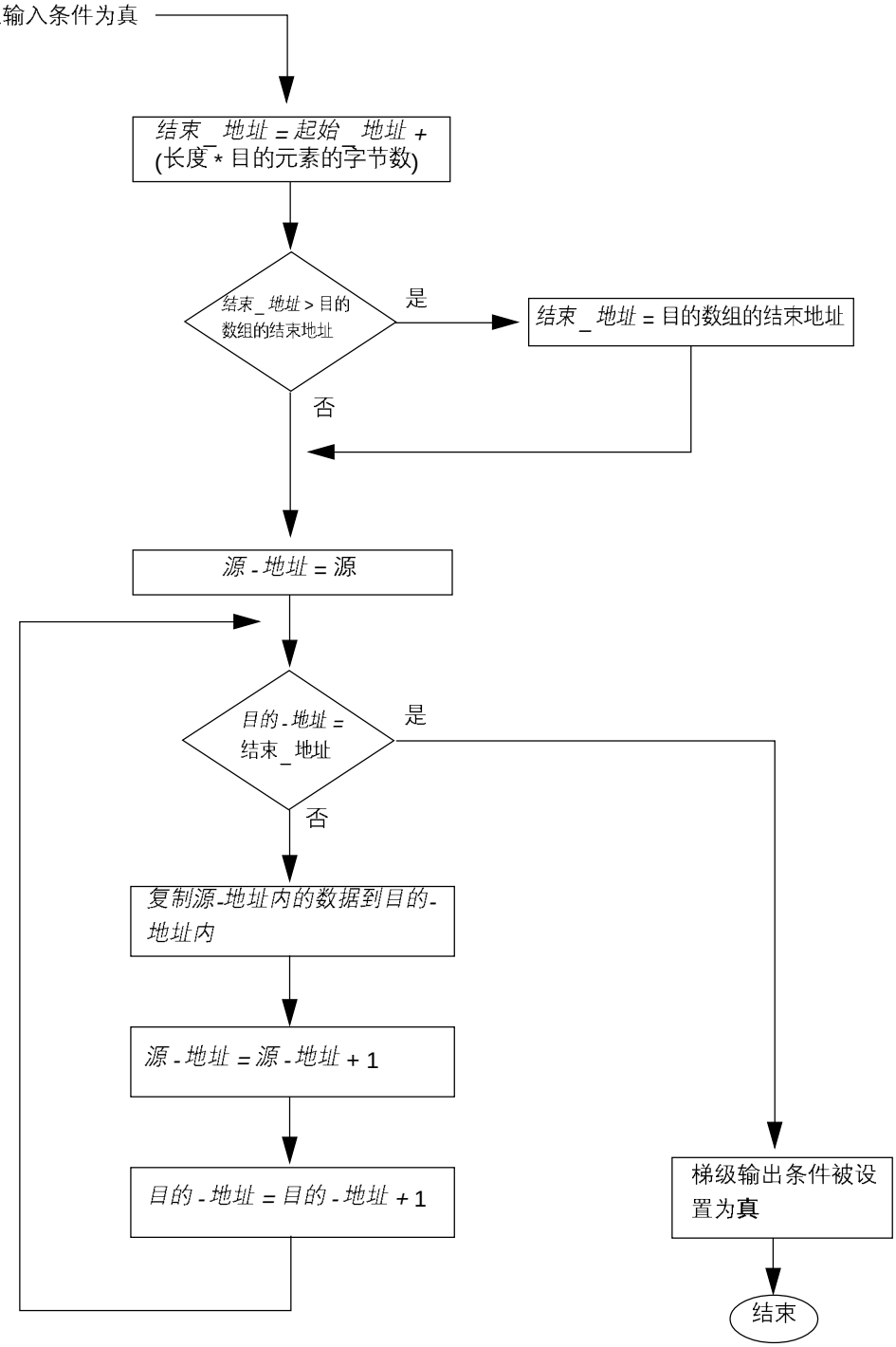
COP 指令对存储器内相邻的数据进行操作, 且执行存储器内字节到字节的直接复制,
这要求理解控制器的存储器结构。*详细信息参见B-3 页的将数组看作一存储块* 。

COP 指令不写出数组的末尾。如果长度大于目的数组元素的总数, 则 COP 指令在数组的末尾停止操作。而且不发生主要故障。

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

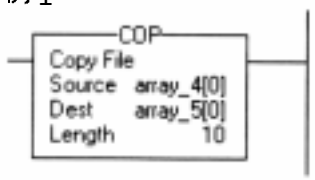


算术状态标志: 不影响

故障条件: 无

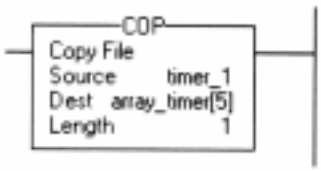
COP 指令举例:

例 1



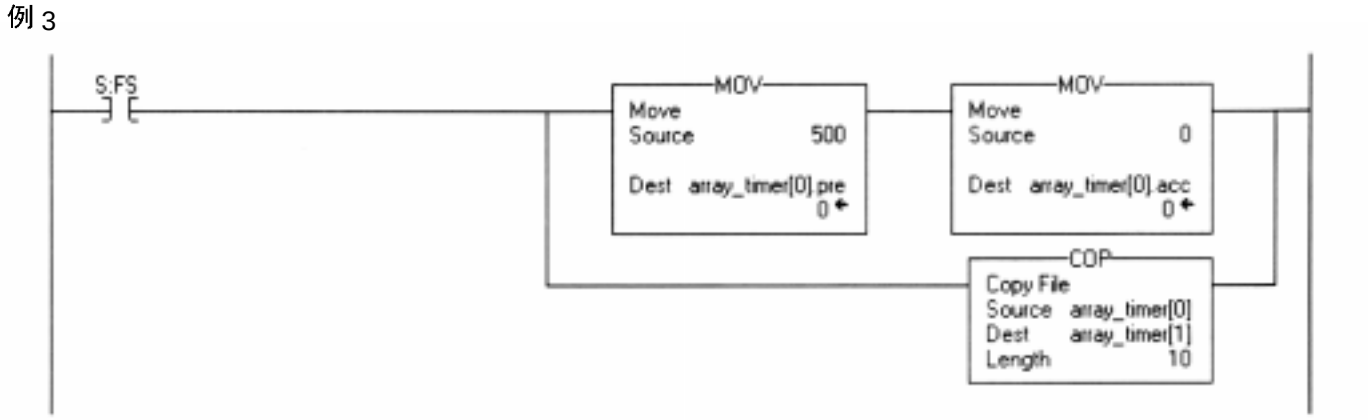
array_4 和 *array_5* 的数据类型相同。当指令被使能时，COP 指令复制*array_4* 内的前 10 个元素到 *array_5* 的前 10 个元素内

例 2

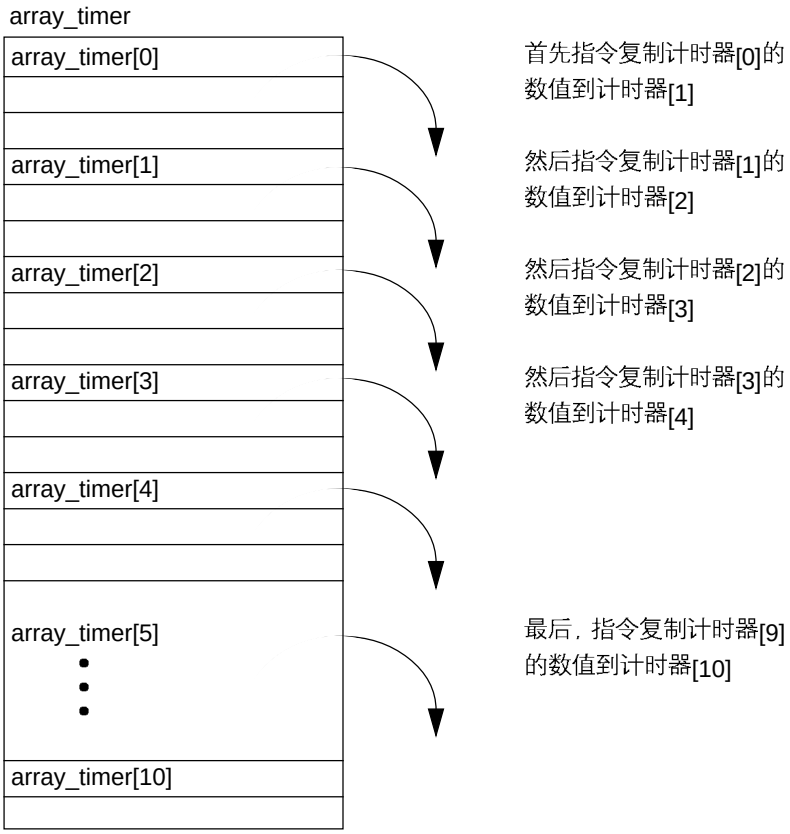


当指令被使能时，COP 指令复制 *timer_1* 结构体到 *array_timer* 的元素 5 。该指令只复制一个结构体到一数组元素内。

例 3



本实例初始化一个计时器结构体。当指令被使能时，MOV指令初始化第一个*array_timer* 元素的预置值(.PRE) 和累加值(.ACC)。当指令被使能时，COP 指令复制相邻的字节块，开始于 *array_timer* [0]。长度是 9 个计时器结构体。



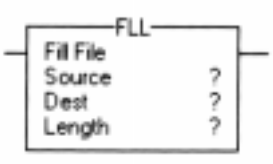
其它格式:

格式:	句法:
neutral 文本	`COP (source,destination,length);`
ASCII 文本	`COP source destination length`

相关指令: FAL，FLL，MOV

文件填充指令 (FLL)

FLL 指令是一条输出指令。



操作数:

操作数:	数据类型:	格式:	说明:
源	**SINT** **INT** **DINT** **REAL**	立即数 标签	要复制的元素(element to copy) **重要**: 源和目的操作数的数据类型应该相同，否则将发生不可预料的结果。
目的	**SINT** **INT** **DINT** **REAL** 结构体	标签	被源操作数覆盖的起始元素 **重要**: 源和目的单元操作数的数据类型应该相同，否则将发生不可预料的结果。 初始化一个结构体的首选方法是用 COP 指令.
长度	DINT	立即数	填充到目的单元的元素数量

说明:

FLL 指令用源值填充一个数组内的元素。源保持不变。填充字节的数量是:
字节数 = 长度 * (目的单元数据类型的字节数)

FLL 指令对存储器内的相邻数据进行操作。

详细信息参见 B-3 *页的将数组看作一存储块*。

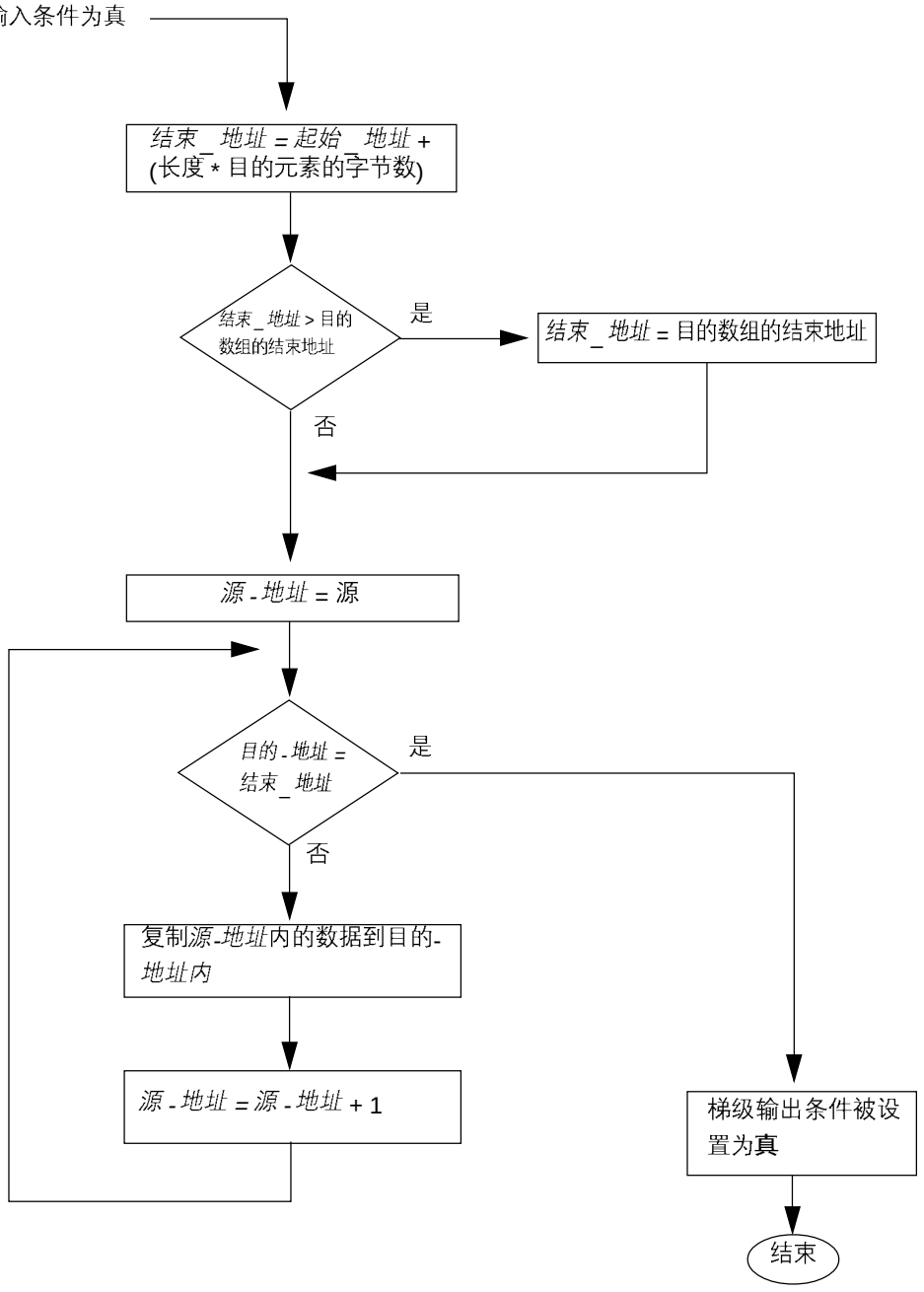
FLL 指令不写出数组的末尾。如果长度值大于目的数组元素的总数，则 FLL 指令在数组的末尾停止操作。不发生主要故障。要获得最好的结果，源和目的应该用相同的数据类型。如果用户要填充一个结构体，可以使用 COP 指令 (参见 7-28 页的例 3)。如果源和目的操作数使用混合数据类型，则目的元素用转换的源值填充。

如果源数据类型是:	目的数据类型是:	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)

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

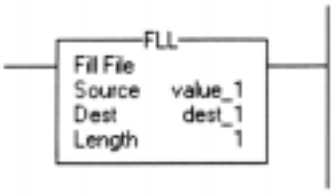
梯级输入条件为真



算术状态标志: 不影响

故障条件: 无

FLL 指令举例:



当指令被使能时，FLL 指令复制 value_1 到 dest_1 内

源(*value_1*) 数据类型:	源(*value_1*) 值:	目的(*dest_1*) 数据类型:	目的(*dest_1*) 值: FLL指令执行之后:
SINT	16#80 (-128)	DINT	16#FFFFFF80 (-128)
DINT	16#1234 5678	SINT	16#78
SINT	16#01	REAL	1.0
REAL	2.0	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

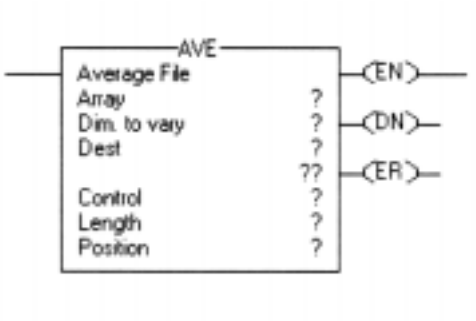
其他格式:

格式:	句法:
neutral 文本	`FLL (source,destination,length);`
ASCII 文本	`FLL source destination length`

相关指令: FAL，COP，MOV

文件平均值指令 (AVE)

AVE 指令是一条输出指令。



操作数:

操作数:	数据类型:	格式:	说明:
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
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: 数组[dim_0，dim_1，dim_2] 数组[dim_0，dim_1] 数组[dim_0]
目的	**DINT** **REAL**	标签	指令运算的结果 如果数组数据类型是 SINT，INT，或 DINT，则目的数据类型必须是DINT 如果数组数据类型是 REAL，则目的数据类型必须是 REAL。
控制	CONTROL	标签	指令运算的控制结构体
长度	DINT	立即数	求平均值的数组元素的数量
位置	DINT	立即数	数组内当前元素的位置 一般起始值是 0

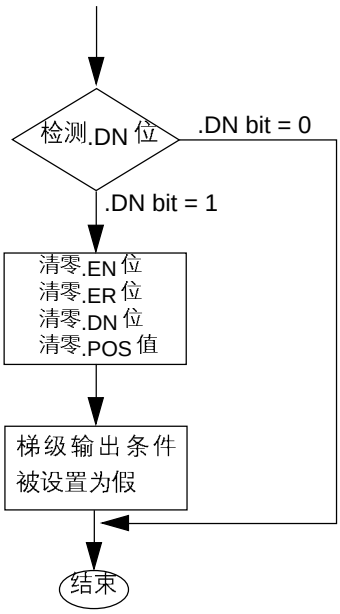
CONTROL 结构体:

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内的最后一个元素时(.POS = .LEN)，完成位被置位。
.ER	BOOL	如果指令执行时发生溢出，则错误位被置位。在程序清零错误位(.ER)之前指令停止执行。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。

说明: AVE 指令计算一组数值的平均值 。

重要: 编程时要确定长度值不能使指令超过指定的变换维数。如果发生此事件，则目的值会不正确。详细信息，
参见 B-3 *页将数组看作一存储块*。

执行:

条件:	**动作**:
预扫描	清零使能位 清零完成位 清零错误位 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	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 = ALL 关于 FAL 指令怎样执行详见 7-8 页。

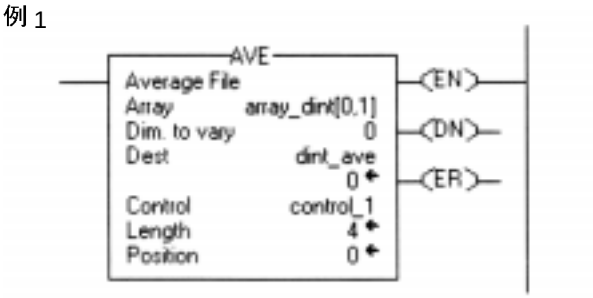
算术状态标志: 影响算术状态标志

故障条件:

发生主要故障的条件:	故障类型:	故障代码:
位置值 < 0 或 长度值 < 0	4	21
指定数组不存在的变换维数	4	20

AVE 指令举例:

例 1



array_dint 是 DINT[4，5]

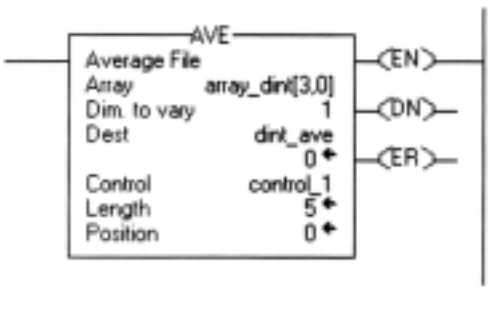
维数 1

不标		0	1	2	3	4
维数 0	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

例 2



array_dint 是 DINT[4,5]

维数 1

维数 0 \ 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

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

$dint\_ave = 3$

其他格式:

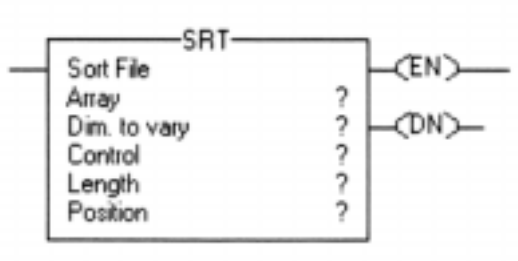
格式:	句法:
neutral 文本	AVE *(array,dim_to_vary,destination,control,length,position);*
ASCII 文本	AVE *array dim_to_vary destination control length position*

相关指令: SRT，STD

文件排序指令(SRT)

SRT 指令是一条输出指令。

操作数:



操作数:	数据类型:	格式:	说明:
数组	SINT INT **DINT** **REAL**	数组标签	要排序的数组 指定要排序的元素组的第一个元素，**不能**在此下标处用 CONTROL.POS
变换的维数	DINT	立即数 (0,1,2)	维数的使用与维数号有关，顺序是数组 [dim_0，dim_1，dim_2] 数组 [dim_0，dim_1] 数组[dim_0]
控制	CONTROL	标签	指令运算的控制结构体
长度	DINT	立即数	排序的数组元素的数量
位置	DINT	立即数	数组内当前元素的位置 一般初始值为 0

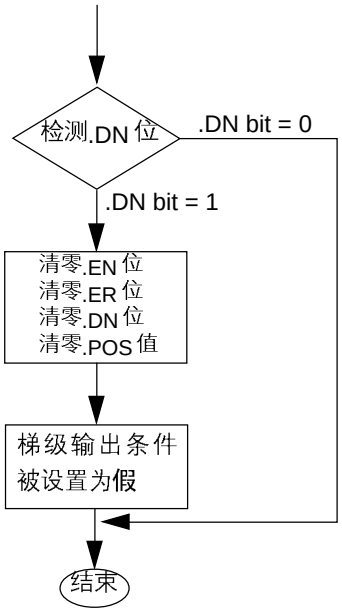
CONTROL 结构体:

助记符:	数据类型:	说明:
.EN	BOOL	使能位表明 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 则置位错误位。这两个故障条件的任何一个都会发生主要错误。
.LEN	DINT	长度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。

说明: SRT 指令以上升的顺序对数组内的一维数组(变换的维数)进行排序 。

重要: 编程时要确定长度值不能超过指令指定的变换维数。如果发生此事件，则会发生不可预料的结果。详细信息，*参见* B-3 *页将数组看作一存储块*。

执行:

条件:	动作:
预扫描	.EN 位清零 .DN 位清零 .ER 位清零 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	SRT 指令以上升的顺序对数组的指定元素进行排序。 实际上，指令用 FAL 指令计算平均值: 表达式 = 排序计算 模式 = ALL 关于 FAL 指令怎样执行详见 7-8 页。

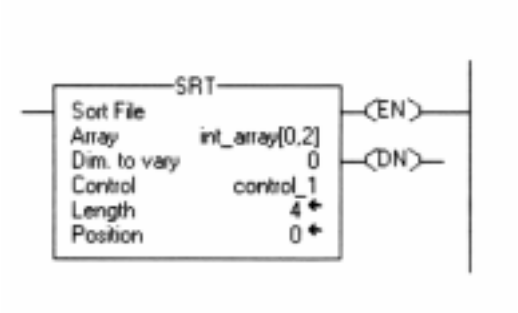
算术状态标志: 影响算术状态标志

故障条件:

发生主要故障的条件:	故障类型:	故障代码:
.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

SRT 指令举例:

例 1



array_dint 是 DINT[4，5]

指令执行之前

维数 1

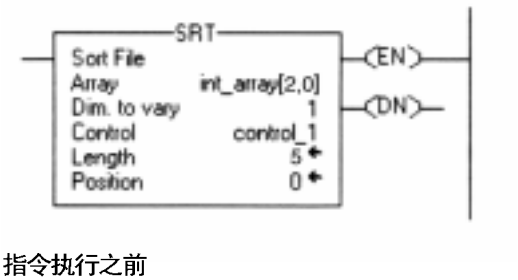
维数 0 下标	0	1	2	3	4
0	20	19	18	17	16
1	15	14	13	12	11
2	10	9	8	7	6
3	5	4	3	2	1

指令执行之后

维数 1

维数 0 下标	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

例 2



array_dint 是 DINT[4，5]

指令执行之前

维数 1

维数 0 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

指令执行之后

维数 1

维数 0 下标	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

其他格式:

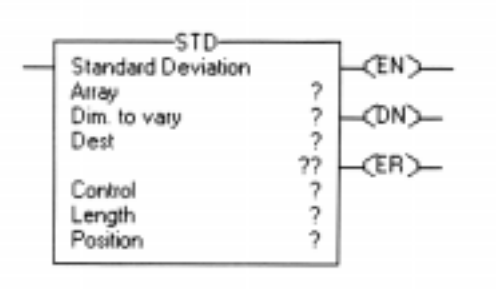
格式:	句法:
neutral 文本	SRT *(array,dim_to_vary,control,length,position);*
ASCII 文本	SRT *array dim_to_vary control length position*

相关指令: AVE，STD

文件标准偏差指令 (STD)

STD 指令是一条输出指令。

操作数:



操作数:	数据类型:	格式:	说明:
数组	SINT INT **DINT** **REAL**	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
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: 数组[dim_0，dim_1，dim_2] 数组[dim_0，dim_1] 数组[dim_0]
目的	REAL	标签	存储运算的结果
控制	CONTROL	标签	指令运算的控制结构体
长度	DINT	立即数	用于计算标准偏差的数组元素的数量
位置	DINT	立即数	数组内当前元素的位置 一般起始值为 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)之前指令停止执行。引起溢出的元素的位置存储在位置(.POS)值内。
.LEN	DINT	长度指定指令处理的数组元素的数量
.POS	DINT	位置值包含指令正访问的当前元素的位置。

说明: STD指令计算数组中一维数组内一组值的标准偏差，并存储结果于目的单元。

重要: 编程时要确定长度不能使指令超过指定的变换维数。如果发生此事件，则会发生不可预料的结果。详细信息，*参见* B-3 *页将数组看作一存储块*。

标准偏差按下列公式进行计算

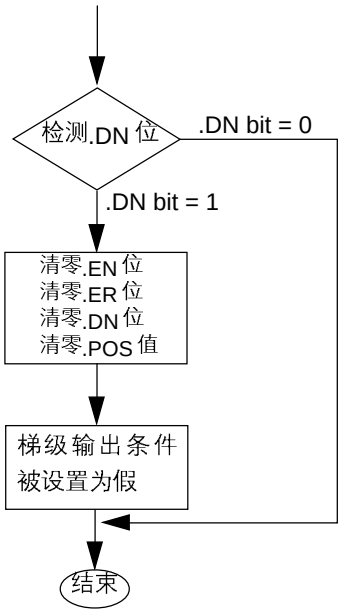
$$\text{标准偏差} = \sqrt{\frac{\left(\sum_{i=0}^{N} [<X_{(start+i)}-AVE>^2]\right)}{(N-1)}}$$

式中:

* 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
* AVE = $\frac{\left(\sum_{i=0}^{N} X_{(start+i)}\right)}{N}$

执行:

条件:	**动作**:
预扫描	清零使能位
	清零完成位
	清零错误位
	梯级输出条件被设置为假。

条件:	动作:
梯级输入条件为假	
梯级输入条件为真	SRT 指令计算指定元素的标准偏差。 实际上，指令用FAL 指令计算平均值: 表达式 = 标准偏差计算 模式 = ALL 对于 FAL 指令怎样执行参见 7-8 页。

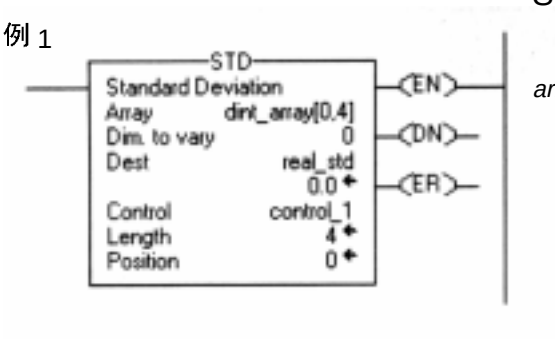
算术状态标志: 影响算术状态标志

故障条件:

发生主要故障的条件:	故障类型:	故障代码:
.POS < 0 或 .LEN < 0	4	21
指定数组不存在的变换维数	4	20

STD 指令举例:

例 1



array_dint 是 DINT[4，5]

维数 1

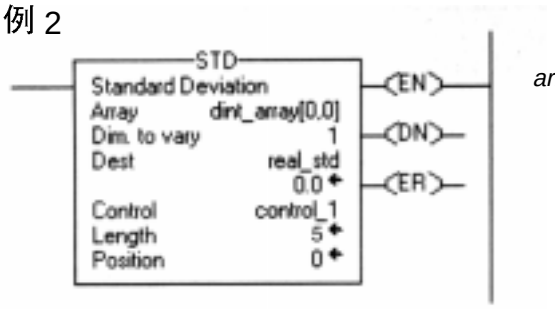
维数 0	0	1	2	3	4
0	20	19	18	17	16
1	15	14	13	12	11
2	10	9	8	7	6
3	5	4	3	2	1

$$AVE = \frac{16+11+6+1}{4} = \frac{34}{4} = 8.5$$

$$STD = \sqrt{\frac{<16-8.5>^2+<11-8.5>^2+<6-8.5>^2+<1-8.5>^2}{<4-1>}} = 6.454972$$

real_std = 6.454972

例 2



array_dint 是 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

$$AVE = \frac{20+19+18+17+16}{5} = \frac{90}{5} = 18$$

$$STD = \sqrt{\frac{<20-18>^2+<19-18>^2+<18-18>^2+<17-18>^2+<16-18>^2}{<5-1>}} = 1.581139$$

real_std=1.581139

其他格式:

格式:	句法:
neutral 文本	`STD (array,dim_to_vary,destination,control,length,position);`
ASCII 文本	`STD array dim_to_vary destination control length position`

相关指令: AVE，SRT

数组(文件)/ 移位指令

(BSL，BSR，FFL，FFU，LFL，LFU)

简介

用数组(文件)/ 移位指令修改数组内数据的位置。

如果用户要:	使用下列指令:	参见页次:
每次完成对位数组中一位的装载，移位，或卸载操作。	BSL	8 - 2
	BSR	8 - 5
按相同的顺序装载或卸载数据。	FFL	8 - 8
	FFU	8 - 14
按相反的顺序装载或卸载数据。	LFL	8 - 20
	LFU	8 - 26

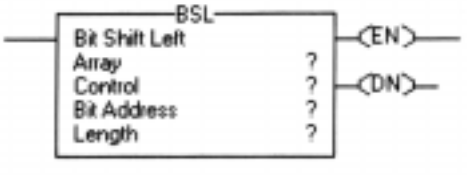
用户可以使用混合数据类型，但是这样会损失精度并且可能发生取整误差。

黑体字数据类型表示最优数据类型。如果一条指令的所有操作数都用同一种最优数据类型，则指令执行的速度快而且占用较少内存。典型的最优数据类型是 DINT 或 REAL。

位左移指令 (BSL)

BSL 指令是一条输出指令。

操作数:



操作数:	数据类型:	格式:	说明:
数组	DINT	数组标签	要改变的数组 指定元素组的第一个元 不要在下标处使用 CONTROL..POS
控制	CONTROL	标签	指令操作的控制结构体
位地址	BOOL	标签	移动的位
长度	DINT	立即数	数组内要移动的位的数量

CONTROL 结构体:

助记符:	数据类型:	说明:
.EN	BOOL	使能位 - 标识 BSL 指令被使能。
.DN	BOOL	完成位被置位表明数组内的指定位已经向左移动了一个位置。
.UL	BOOL	卸载位是指令的输出位。卸载位(.UL)存储被移动超出位范围的位状态。
.ER	BOOL	当长度(.LEN) < 0 时，置位错误位。
.LEN	DINT	由长度指定数组内被移动的位的数量。

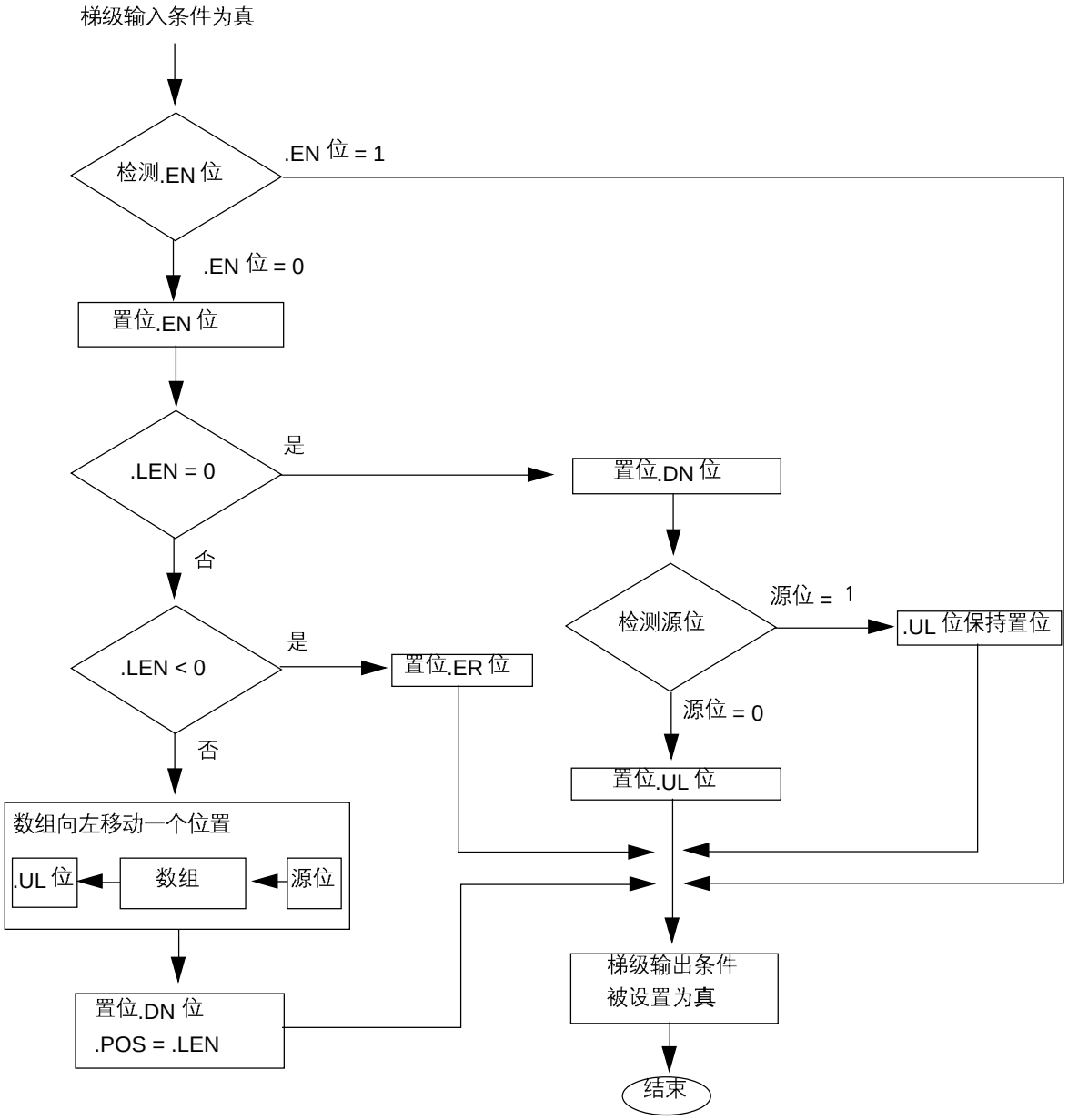
说明: BSL 指令使数组内的指定位向左移动一个位置。当指令被使能时，把指定位的最高位卸载到卸载位(.UL)，其余的位向左移动一个位置，并且装载位地址于数组的位 0 内。

BSL 指令对连续存储器单元进行操作。详细信息参见附录 B - 3 页的将数组看作一存储块。

执行:

条件:	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(.POS)值 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	清零使能位(.EN) 清零完成位(.DN) 清零错误位(.ER) 清零位置(.POS)值

条件: **动作:**

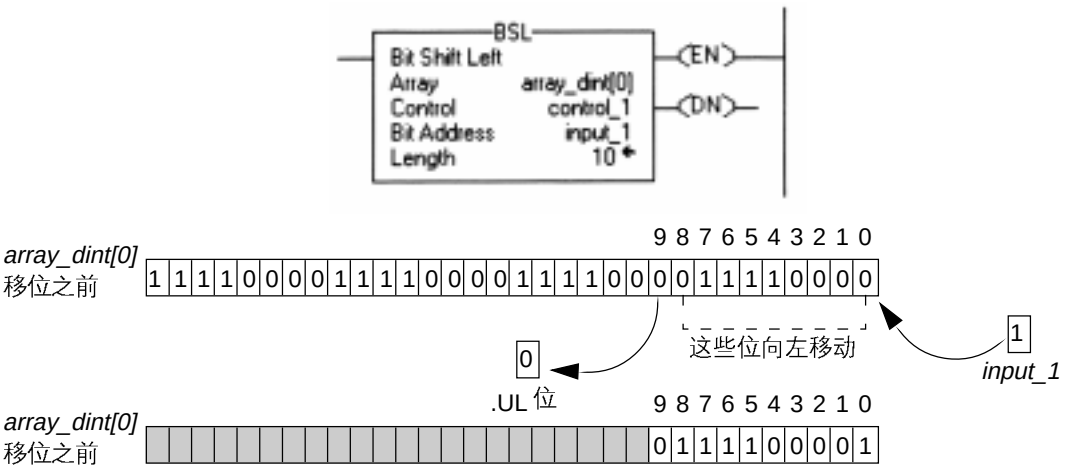


算术状态标志: 不影响

故障条件: 无

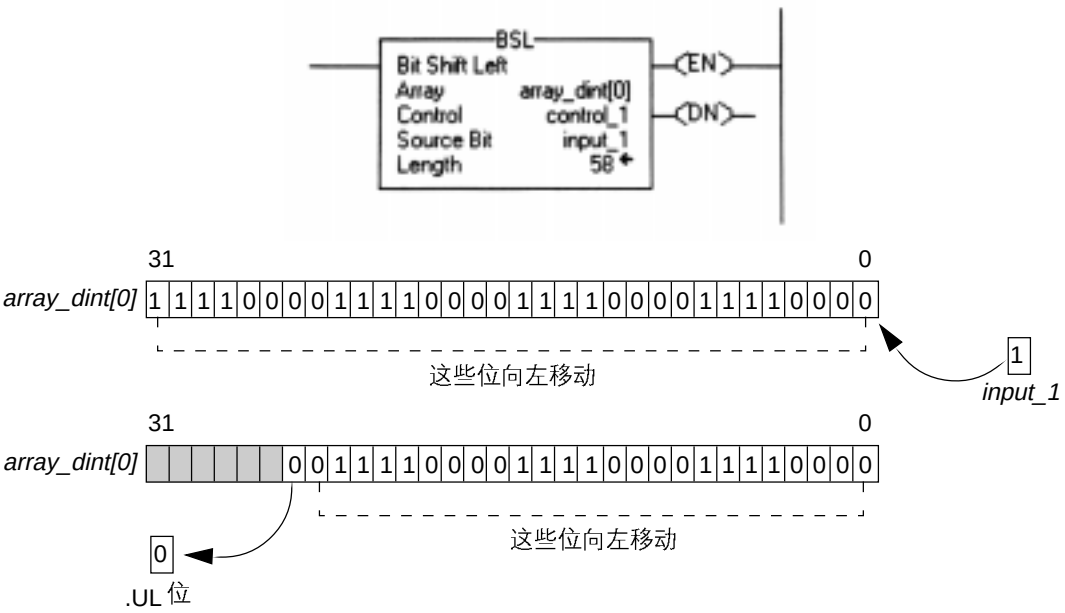
BSL 指令举例:

例 1



当指令被使能时，BSL 指令从 *array_dint[0]* 的位 0 开始执行，指令卸载 *array_dint[0]* 的位 9 进入卸载位(.UL)，同时移动其余的位，并装载 *input_1* 进入 *array_dint[0]* 的位 0。 其余位(10-31)无效。

例 2



当指令被使能时，BSL 指令从 *array_dint[0]* 的位 0 开始执行，指令卸载 *array_dint[1]* 的位 25 进入卸载(.UL)位，同时移动其余的位，并装载 *input_1* 进入 *array_dint[0]* 的位 0。 其余位

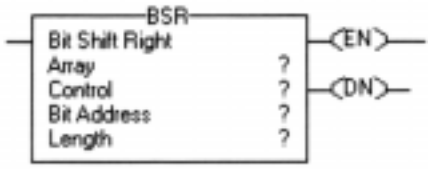
其他格式:

格式:	句法:
neutral 文本	BSL *(array,control,source_bit,length);*
ASCII 文本	BSL *array control source_bit length*

相关指令: BSR

位右移指令 (BSR)

BSR 指令是一条输出指令。



操作数:

操作数:	数据类型:	格式:	说明
数组	DINT	数组标签	要改变的数组 指定开始移动的元素 不要在下标处使用CONTROL.POS
控制	CONTROL	标签	指令操作的控制结构体
位地址	BOOL	标签	移动的位
长度	DINT	立即数	数组内移动的位的数量

CONTROL 结构体:

助记符:	数据类型:	说明:
.EN	BOOL	使能位 - 标识 BSR 指令被使能。
.DN	BOOL	完成位被置位表明指令指定的位已经向右移动了一个位置。
.UL	BOOL	卸载位是指令的输出位。卸载位(.UL)存储被移动超出位的范围的位状态。
.ER	BOOL	如果长度值(.LEN)< 0 则错误位被置位。
.LEN	DINT	指定数组内被移动的位的数量。

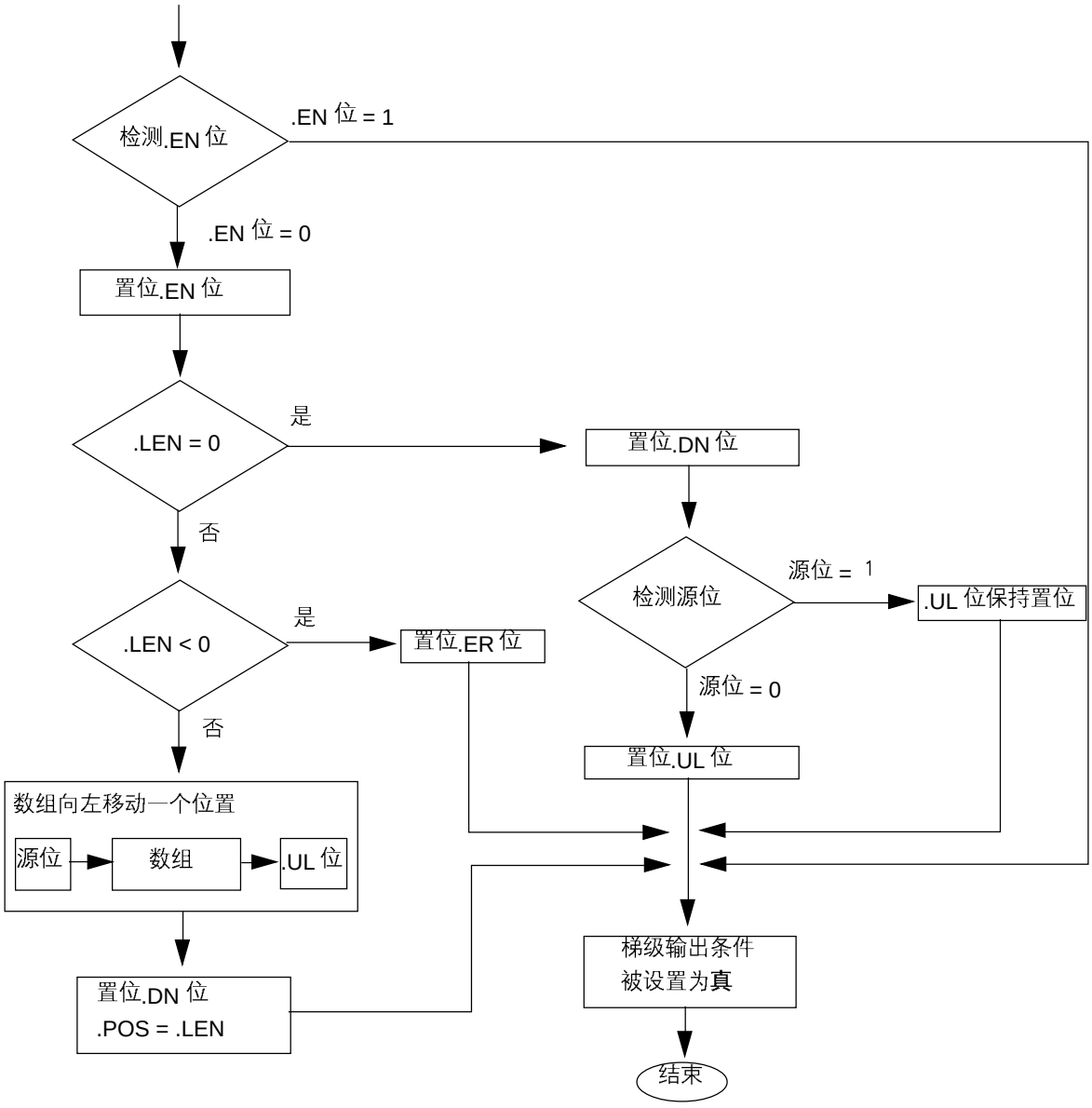
说明:

BSR 指令使数组内的指定位向右移动一个位置。当指令被使能时，指令把数组内位 0 的值卸载到卸载位(.UL)，其余的位向右移动一个位置，并且装载位地址于指定位的最高位 。

BSR 指令对连续存储器单元进行操作。详细信息参见附录 B - 3 页的将数组看作一存储块。

执行:

条件:	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(.POS)值
	梯级输出条件被设置为假

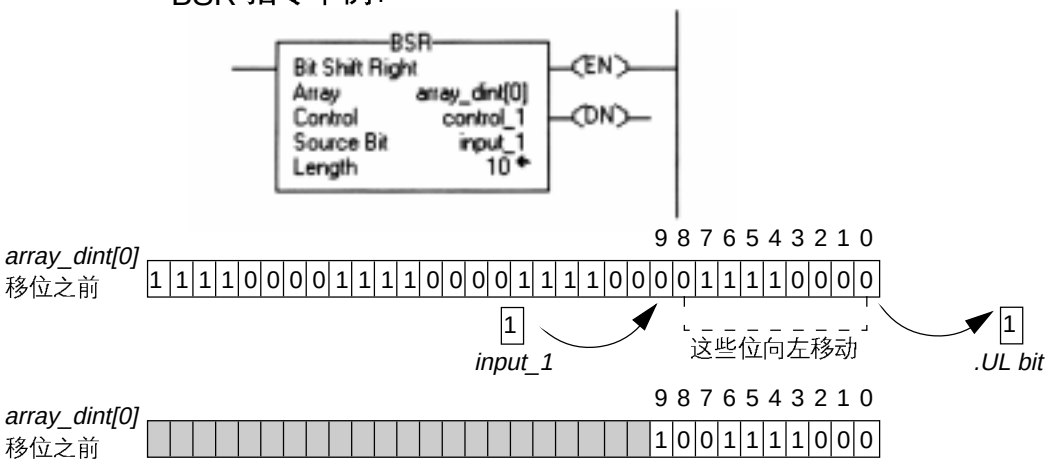
条件:	动作:
梯级输入条件为假	清零使能位(.EN) 清零错误位(.ER) 清零完成位(.DN) 清零位置(.POS)值 梯级输出条件被设置为假
梯级输入条件为真	

算术状态标志: 不影响

故障条件: 无

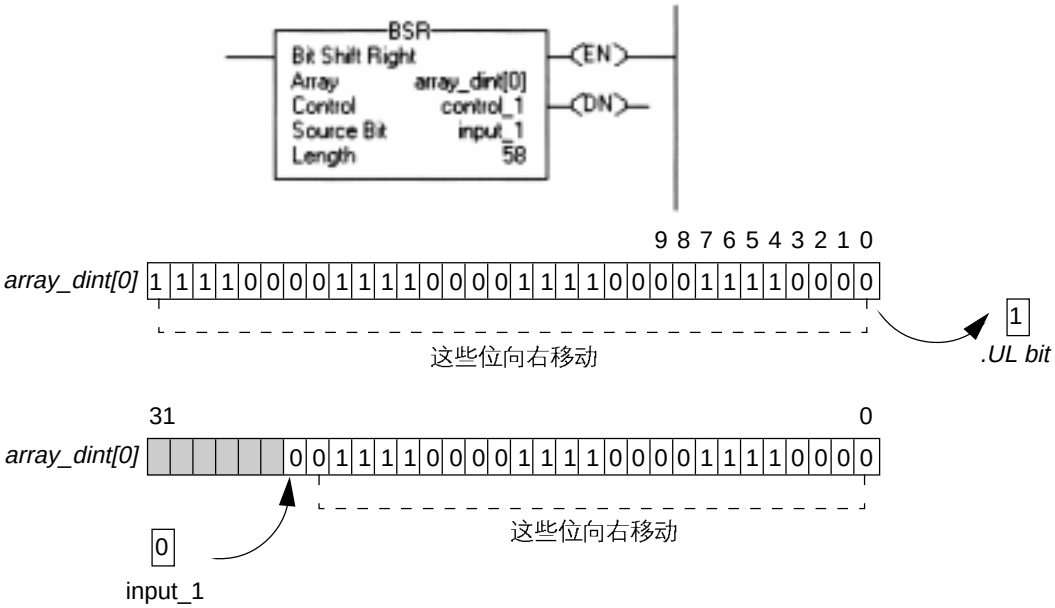
BSR 指令举例:

例 1



当指令被使能时，BSR 指令从 *array_dint[0]* 的位 9 开始执行，指令卸载 *array_dint[0]* 的位 0 进入.UL 位，右移其余的位，并装载 *input_1* 进入 *array_dint[0]* 的位 9。其余位(10-31)无效。

例 2



当指令被使能时，BSR 指令在 *array_dint[1]* 内的位 25 开始执行，指令卸载 *array_dint[0]* 的位 0 进入.UL 位，右移动其余的位，并装载 *input_1* 进入 *array_dint[1]* 的位 25。其余位 *(array_dint[1]* 内的 31-26) 无效。

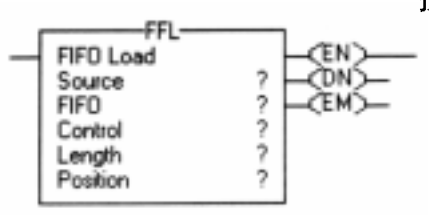
其他格式:

格式:	句法:
neutral 文本	BSR*(array,control,source_bit,length);*
ASCII 文本	BSR *array control source_bit length*

相关指令: BSL

FIFO 装载指令 (FFL)

FFL 是一条输出指令。



操作数:

操作数:	数据类型:	格式:	说明
源	SINT INT **DINT** **REAL**	立即数 标签	存入 FIFO 内的数据
FIFO	**DINT** **REAL**	数组标签	要改变的 FIFO 指定 FIFO 的第一个元素 **不能**在下标处使用 CONTROL.POS
控制	CONTROL	标签	指令操作的控制结构体 一般与其相关的 FFU 指令使用相同的控制(CONTROL)结构体.
长度	DINT	立即数	FIFO 可以同时支持的元素的最大数量
位置	DINT	立即数	在FIFO 内指令装载数据的下一个位置.一般起始值是 0

CONTROL 结构体:

助记符:	数据类型:	说明:
.EN	.BOOL	使能位表明 FFL 指令被使能.
.DN	BOOL	完成位被置位，表明FIFO 已满(.POS = .LEN)。在.POS < .LEN 之前，完成位(.DN)禁止数据装入 FIFO 。
.EM	BOOL	栈空位表明 FIFO 是空的。如果长度值(.LEN)≤ 0 或 位置值(.POS)< 0，则栈空位(.EM)和完成位(.DN)都被置位。
.LEN	DINT	长度指定 FIFO 可以同时支持的元素最大数量
.POS	DINT	位置表示指令将要装载的下一个值在 FIFO 内的位置

说明:

FFL 指令复制源值到FIFO 内。用FFL 指令和FFU 指令存储数据，并且可以按先进 / 先出的顺序取回数据。当使用该指令对时，FFL 和 FFU 指令建立了一个异步移位寄存器。

通常，源操作数和 FIFO 用相同的数据类型。

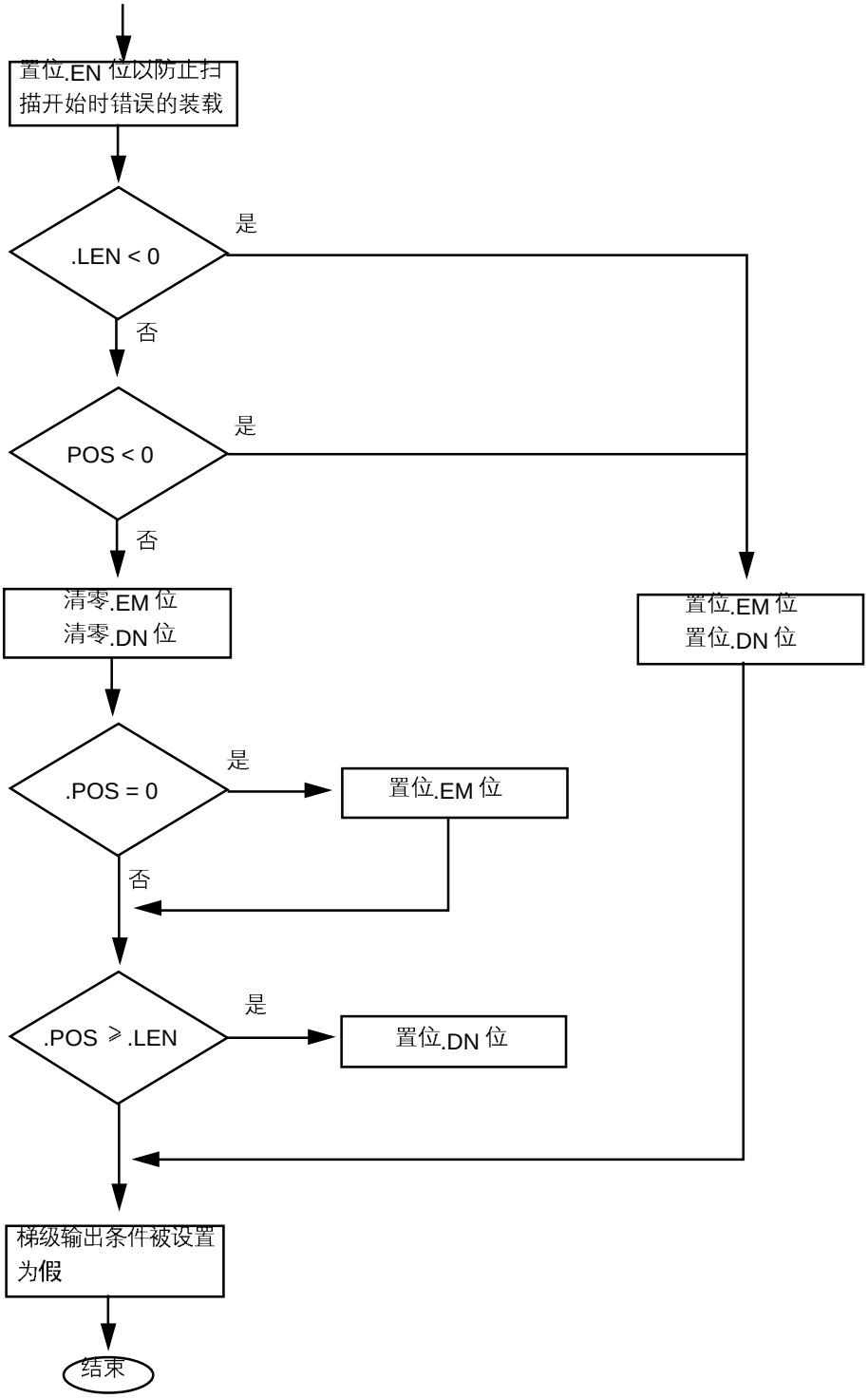
当指令被使能时，FFL 指令把源值装入由位置值(.POS)确定的FIFO 内的位置。每次指令被使能装载一个数值，直到 FIFO 满为止。

FFL 指令对连续存储器单元进行操作。*详细信息参见附录 B - 3 页的将数组看作一存储块*。

执行 :

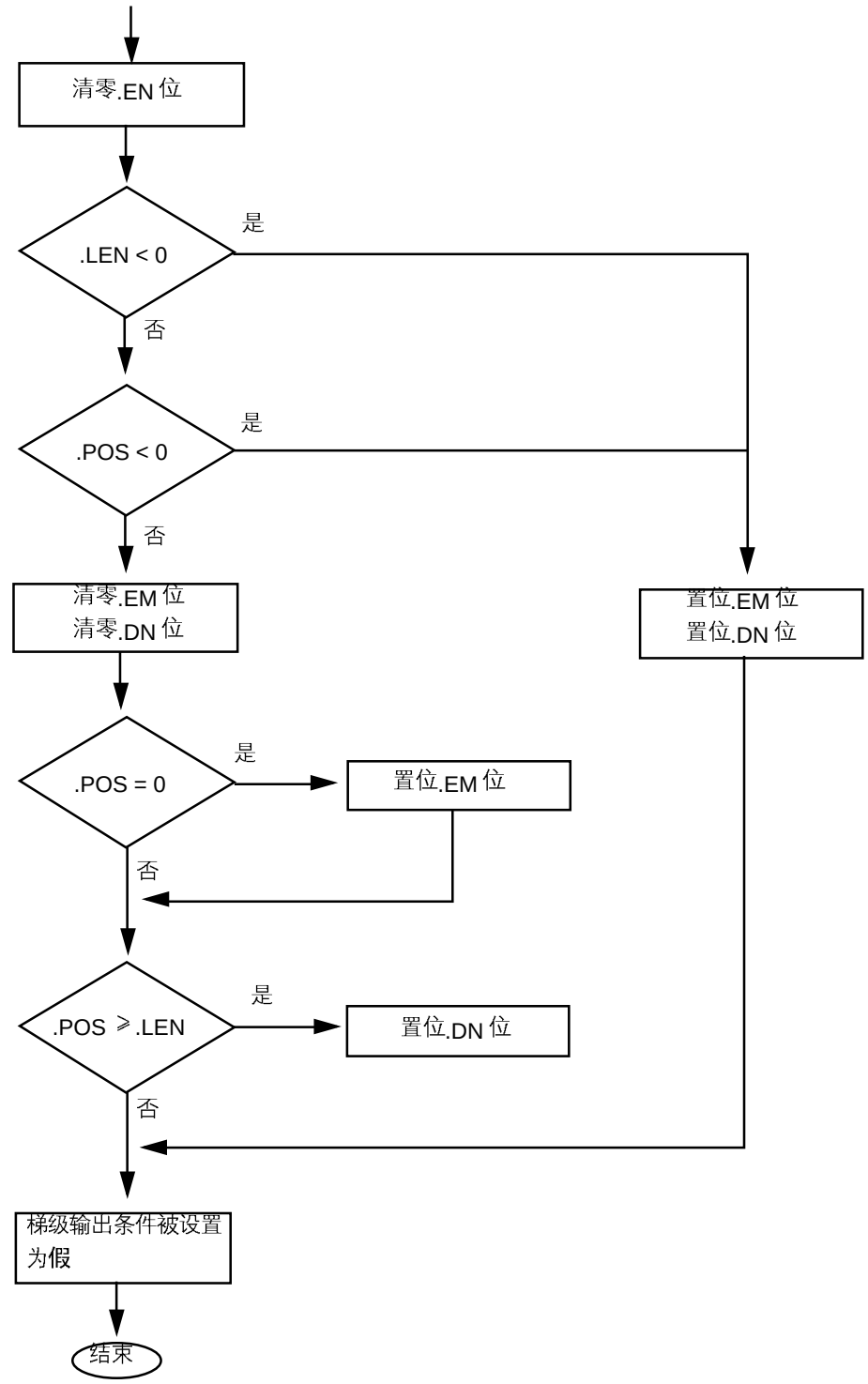
条件:	动作:

预扫描

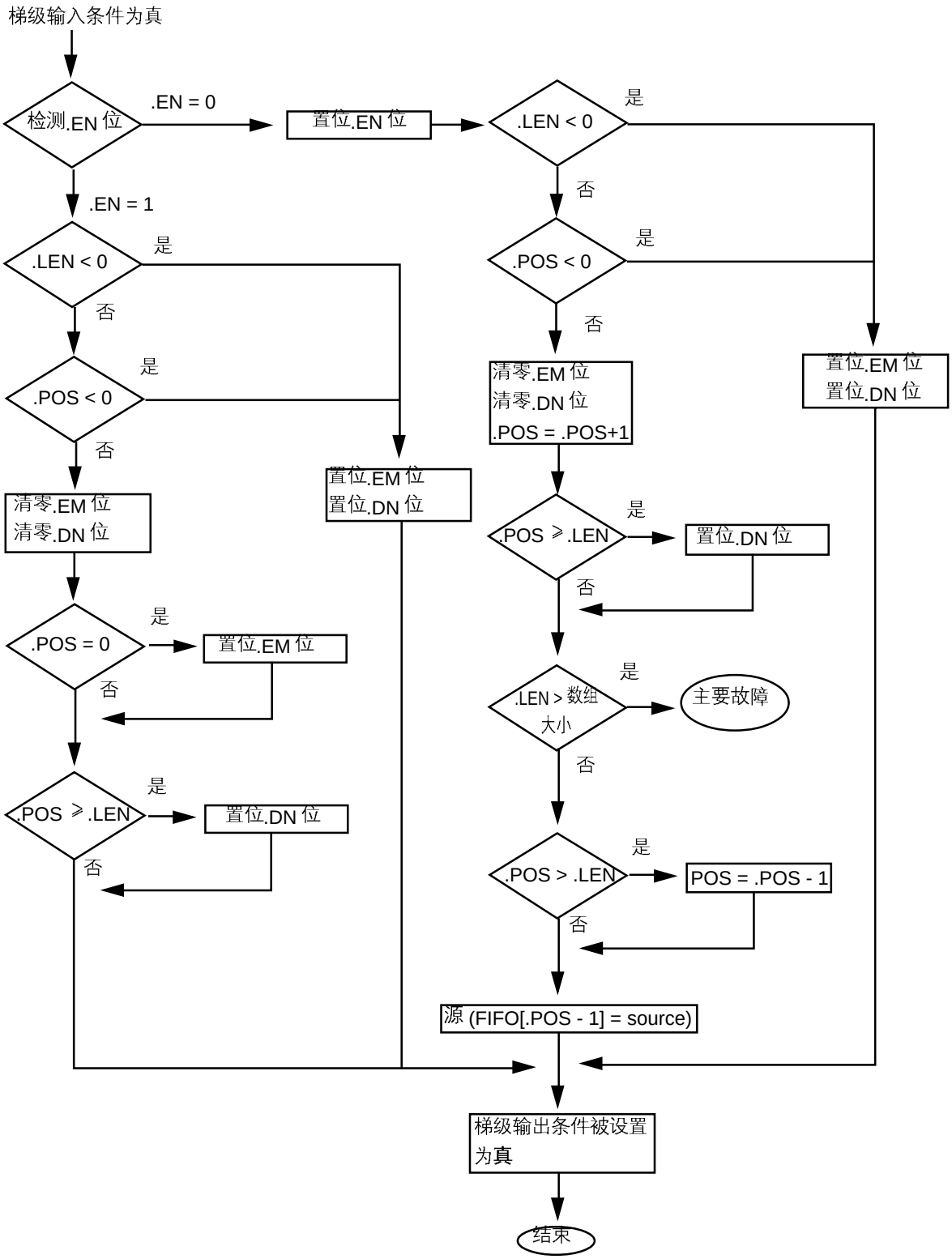

置位.EN 位以防止扫描开始时错误的装载
.LEN < 0
是
否
POS < 0
是
否
清零.EM 位
清零.DN 位
置位.EM 位
置位.DN 位
.POS = 0
是
置位.EM 位
否
.POS ≥ .LEN
是
置位.DN 位
梯级输出条件被设置为假
结束

条件: **动作:**

梯级输入条件为假



条件: **动作:**

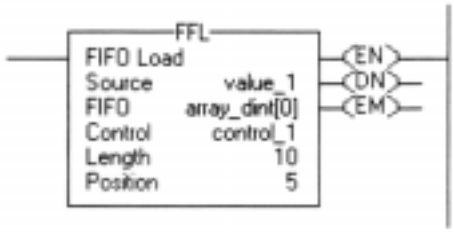

梯级输入条件为真
检测.EN 位
.EN = 0
置位.EN 位
.EN = 1
.LEN < 0
是
否
.POS < 0
是
否
清零.EM 位
清零.DN 位
.POS = 0
是
置位.EM 位
否
.POS ⩾ .LEN
是
置位.DN 位
否
置位.EM 位
置位.DN 位
.LEN < 0
是
否
.POS < 0
是
否
清零.EM 位
清零.DN 位
.POS = .POS+1
置位.EM 位
置位.DN 位
.POS ⩾ .LEN
是
置位.DN 位
否
.LEN > 数组大小
是
主要故障
否
.POS > .LEN
是
POS = .POS - 1
否
源 (FIFO[.POS - 1] = source)
梯级输出条件被设置为真
结束

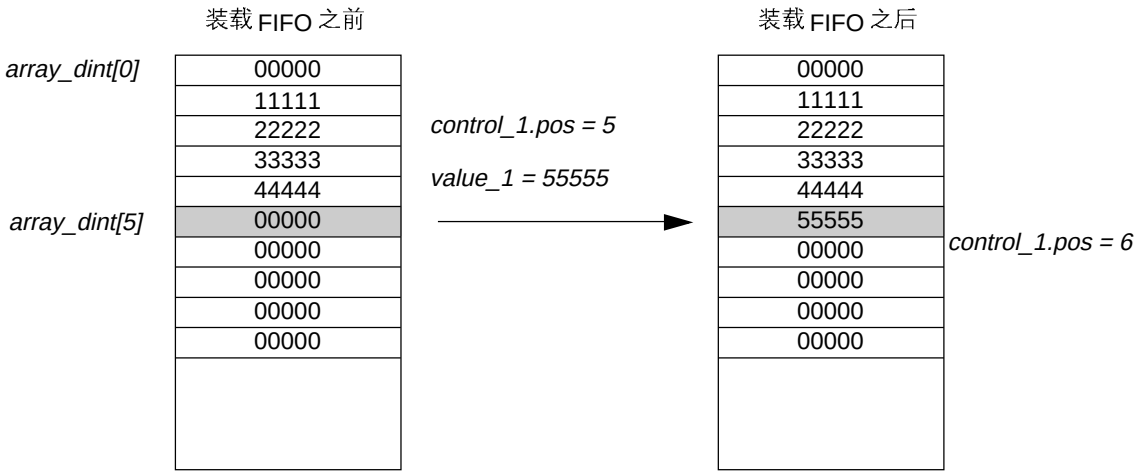
算术状态标志: 不影响

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 数组大小	4	20

FFL 指令举例:





当指令被使能时，FFL 指令把 *value_1* 装入 FIFO 的下一个位置，既本例的 *array_dint[5]* 。

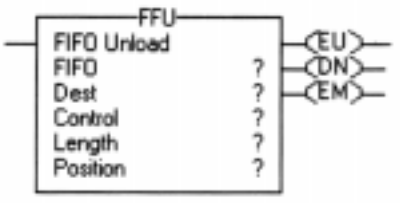
其他格式:

格式:	句法:
neutral 文本	`FFL(source,FIFO,control,length,position);`
ASCII 文本	`FFL source FIFO control length position`

相关指令: FFU，LFL，LFU

FIFO 卸载指令 (FFU)

FFU 是一条输出指令.



操作数:

操作数:	数据类型:	格式:	说明:
FIFO	**DINT** **REAL**	数组标签	要改变的 FIFO 指定 FIFO 的第一个元素 **不能**在下标处使用 CONTROL.POS
目的	**DINT** **REAL**	标签	从 FIFO 卸出的数值。
控制	CONTROL	标签	操作的控制结构体 一般与其相关的 FFL 指令使用相同的CONTROL结构体。
长度	DINT	立即数	FIFO 可以同时支持的元素的最大数量。
位置	DINT	立即数	在FIFO 内指令卸载数据的下一个位置。一般起始值是 0

CONTROL 结构体:

助记符:	数据类型:	说明:
.EN	BOOL	使能位 - 标识 FFU 指令被使能.
.DN	BOOL	完成位被置位表明 FIFO 已满(.POS = .LEN)。
.EM	BOOL	栈空位表明 FIFO 是空的。如果长度值(.LEN) ≤ 0 或位置值(.POS)< 0，则栈空位(.EM)和完成位(.DN)都被置位。
.LEN	DINT	长度指定 FIFO 内元素的最大数量
.POS	DINT	位置表示指令已经装入 FIFO 的最后数据

说明:

FFU 指令卸载 FIFO 的位置 0(第一个位置)的数值并存放该值于目的单元。FIFO内的其余数据向下移动一个位置。FFU 指令与 FFL 指令一起按先进 / 先出的顺序存储和返回数据。

如果 FIFO 是 DINT 数据类型，则目的单元必须也是 DINT 数据类型；如果 FIFO 是 REAL 数据类型，则目的单元必须也是 REAL 数据类型。

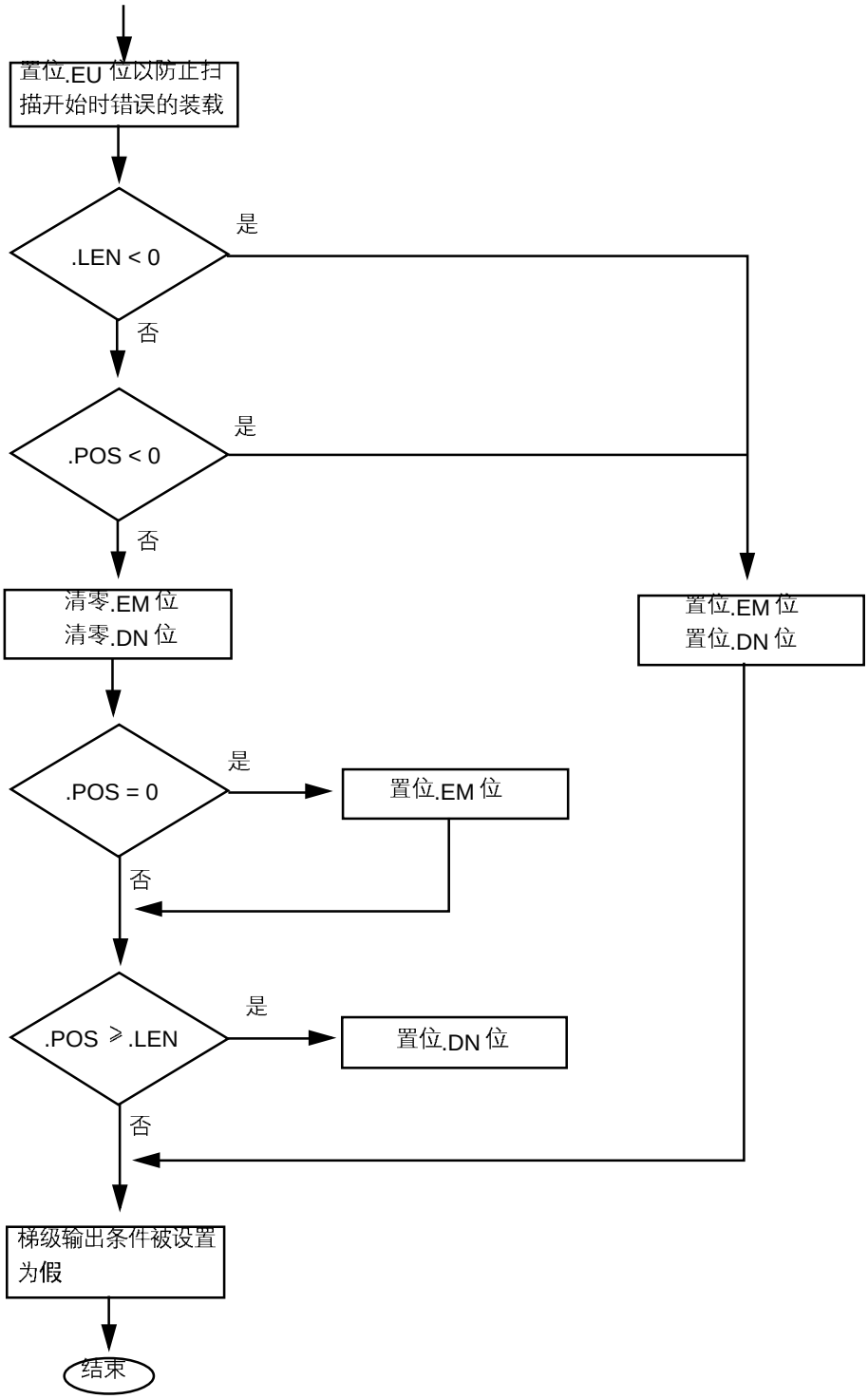
当指令被使能时，FFU 指令从FIFO的第一个元素卸载数据并存放该值于目的单元。指令每次被使能时卸载一个数值，直到 FIFO 栈空为止。如果 FIFO 为空，则 FFU 指令向目的单元返回 0 值。

FFU 指令对连续存储器单元进行操作。详细信息参见附录 B - 3 页的将数组看作一存储块。

执行 :

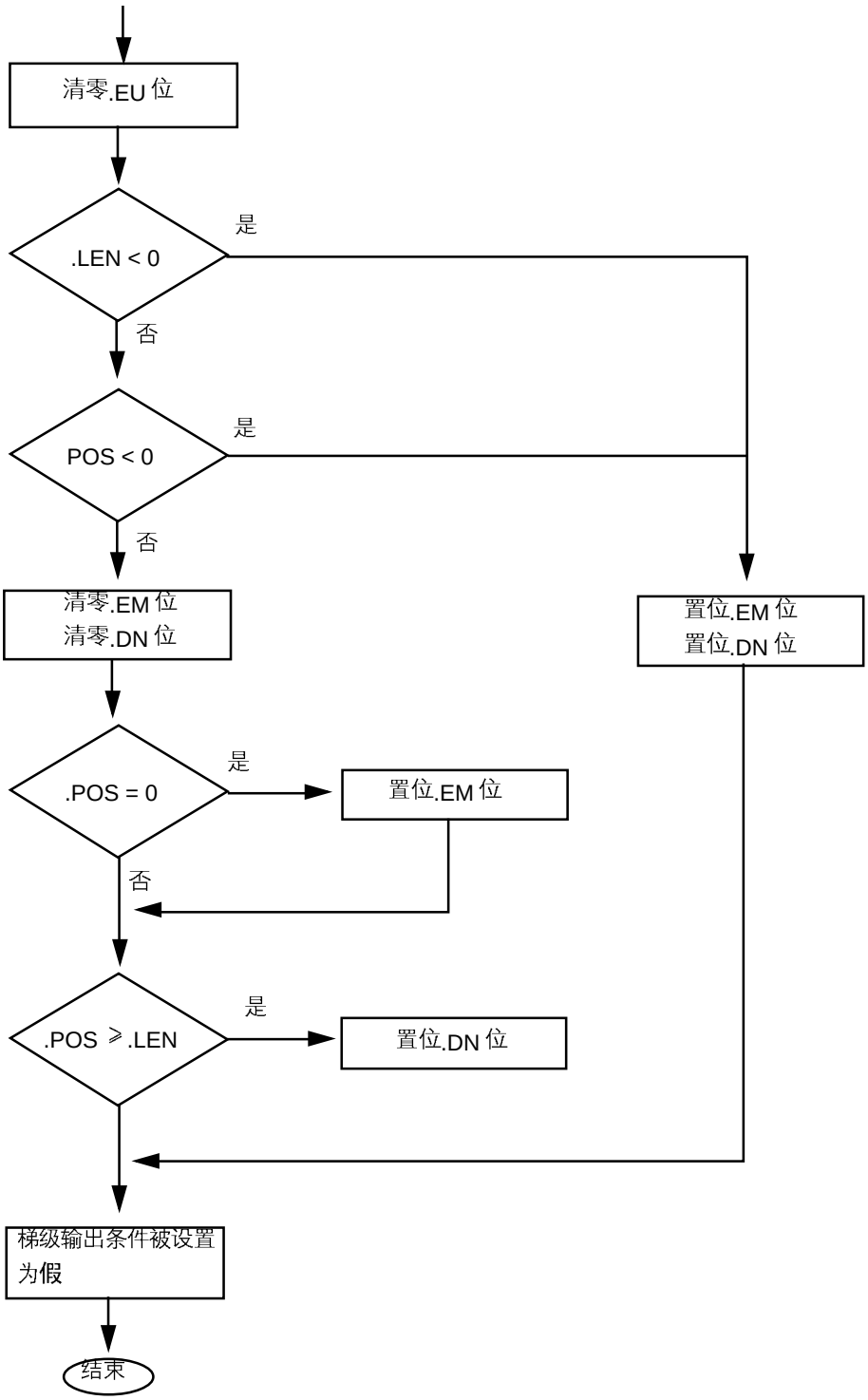
条件:	动作:

预扫描

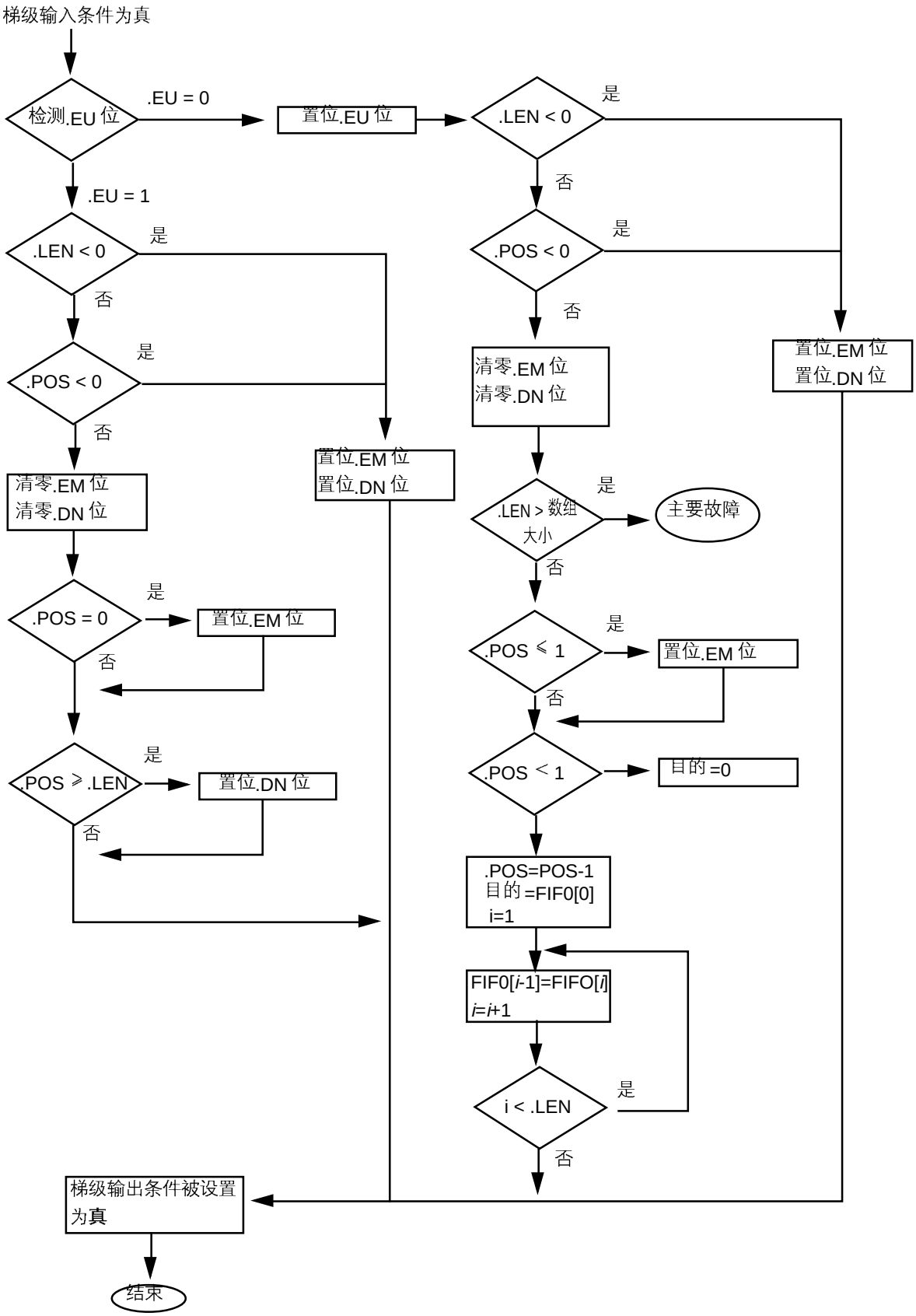

置位.EU 位以防止扫描开始时错误的装载
.LEN < 0
是
否
.POS < 0
是
否
清零.EM 位
清零.DN 位
置位.EM 位
置位.DN 位
.POS = 0
是
置位.EM 位
否
.POS ⩾ .LEN
是
置位.DN 位
否
梯级输出条件被设置为假
结束

条件: **动作:**

梯级输入条件为假



条件: **动作:**

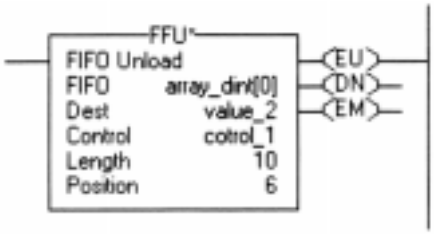

梯级输入条件为真
检测.EU 位
.EU = 0
置位.EU 位
.EU = 1
.LEN < 0
是
否
.POS < 0
是
否
清零.EM 位
清零.DN 位
置位.EM 位
置位.DN 位
.POS = 0
是
置位.EM 位
否
.POS ⩾ .LEN
是
置位.DN 位
否
.LEN < 0
是
否
.POS < 0
是
否
清零.EM 位
清零.DN 位
置位.EM 位
置位.DN 位
.LEN > 数组大小
是
主要故障
否
.POS ⩽ 1
是
置位.EM 位
否
.POS < 1
目的 =0
.POS=POS-1
目的 =FIF0[0]
i=1
FIF0[i-1]=FIFO[i]
i=i+1
i < .LEN
是
否
梯级输出条件被设置为真
结束

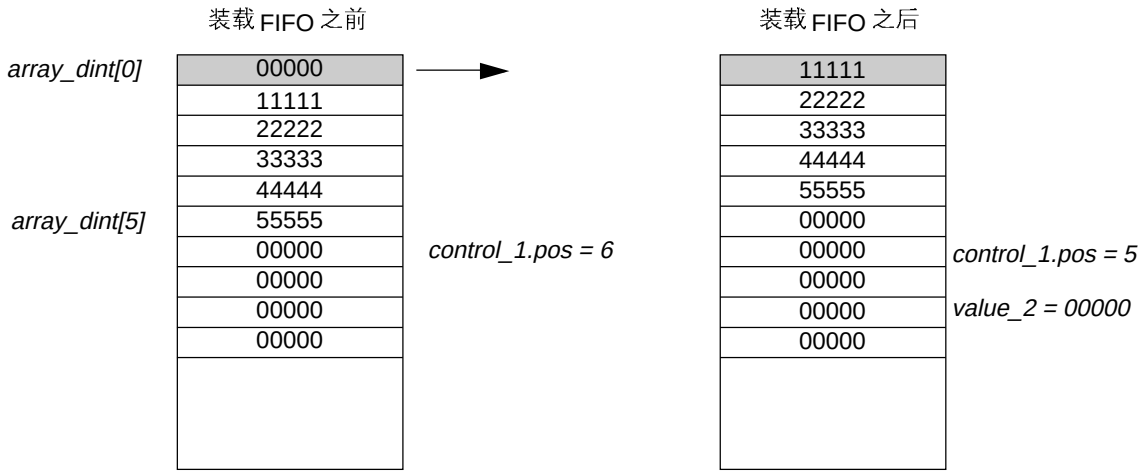
算术状态标志: 不影响

故障条件:

发生主要故障的条件:	故障类型:	故障代码:
长度 > FIFO 数组大小	4	20

FFU 指令举例:





当指令被使能时，FFU 指令卸载 *array_dint[0]* 进入 *value_2* 并且移动 *array_dint* 内的其余元素。

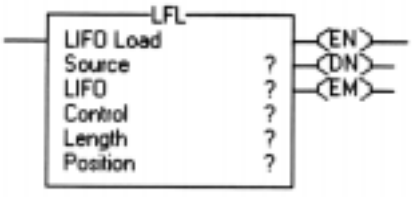
其他格式:

格式:	句法:
neutral 文本	`FFU(FIFO,destination,control,length,position);`
ASCII 文本	`FFU FIFO destination control length position`

相关指令: FFL，LFL，LFU

LIFO 装载指令 (LFL)

LFL 是一条输出指令。



操作数:

操作数:	数据类型:	格式:	说明:
源	SINT INT **DINT** **REAL**	立即数 标签	被存入 LIFO 内的数据
LIFO	**DINT** **REAL**	数组标签	指定 LIFO 的第一个元素 **不能**在标记处使用 CONTROL.POS
控制	CONTROL	标签	操作的控制结构体 一般与其相关的 LFU 指令使用相同的CONTROL结构体.
长度	DINT	立即数	LIFO 可以同时支持的元素最大数量
位置	DINT	立即数	在LIFO 内指令装载数据的下一个位置.典型的起始值是 0

CONTROL 结构体:

助记符:	数据类型:	说明:
.EN	BOOL	使能位表明 LFL 指令被使能.
.DN	BOOL	完成位被置位表明 LIFO 已满(.POS = .LEN)。在.POS < .LEN 之前完成(.DN)位禁止装入 LIFO 。
.EM	BOOL	栈空位表明LIFO 是空的。如果长度值(.LEN) ≤ 0 或位置值(POS)< 0 ，则栈空位(.EM)和完成位(.DN)都被置位。
.LEN	DINT	长度指定 LIFO 可以同时支持的元素最大数量
.POS	DINT	位置表示指令将要装载的下一个值在 LIFO 的位置

说明:

LFL 指令复制源值到 LIFO 内。用 LFL 指令和 LFU 指令存储数据，并且可以按后进 / 先出的顺序取回数据。当使用该指令对时，LFL 和 LFU 指令建立了一个异步移位寄存器。

一般源操作数和 LIFO 用相同的数据类型。

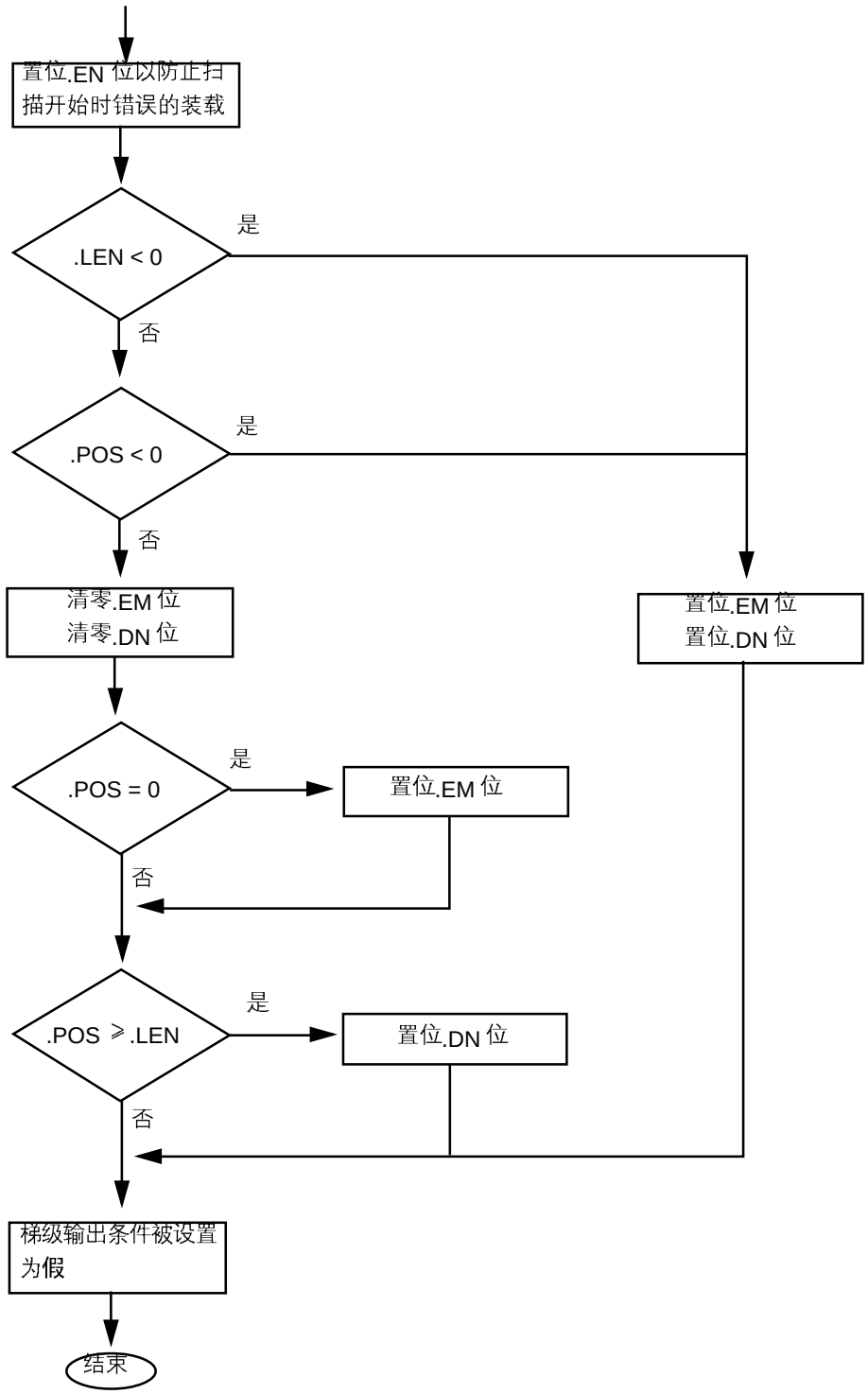
当指令被使能时，LFL 指令把源值装入由位置(.POS)值确定的 LIFO 内的位置内。每次指令被使能装载一个数值，直到 LIFO 栈满为止。

LFL FBC 指令对连续存储器单元进行操作。详细信息参见附录 B - 3 页的将数组看作一存储块。

执行 :

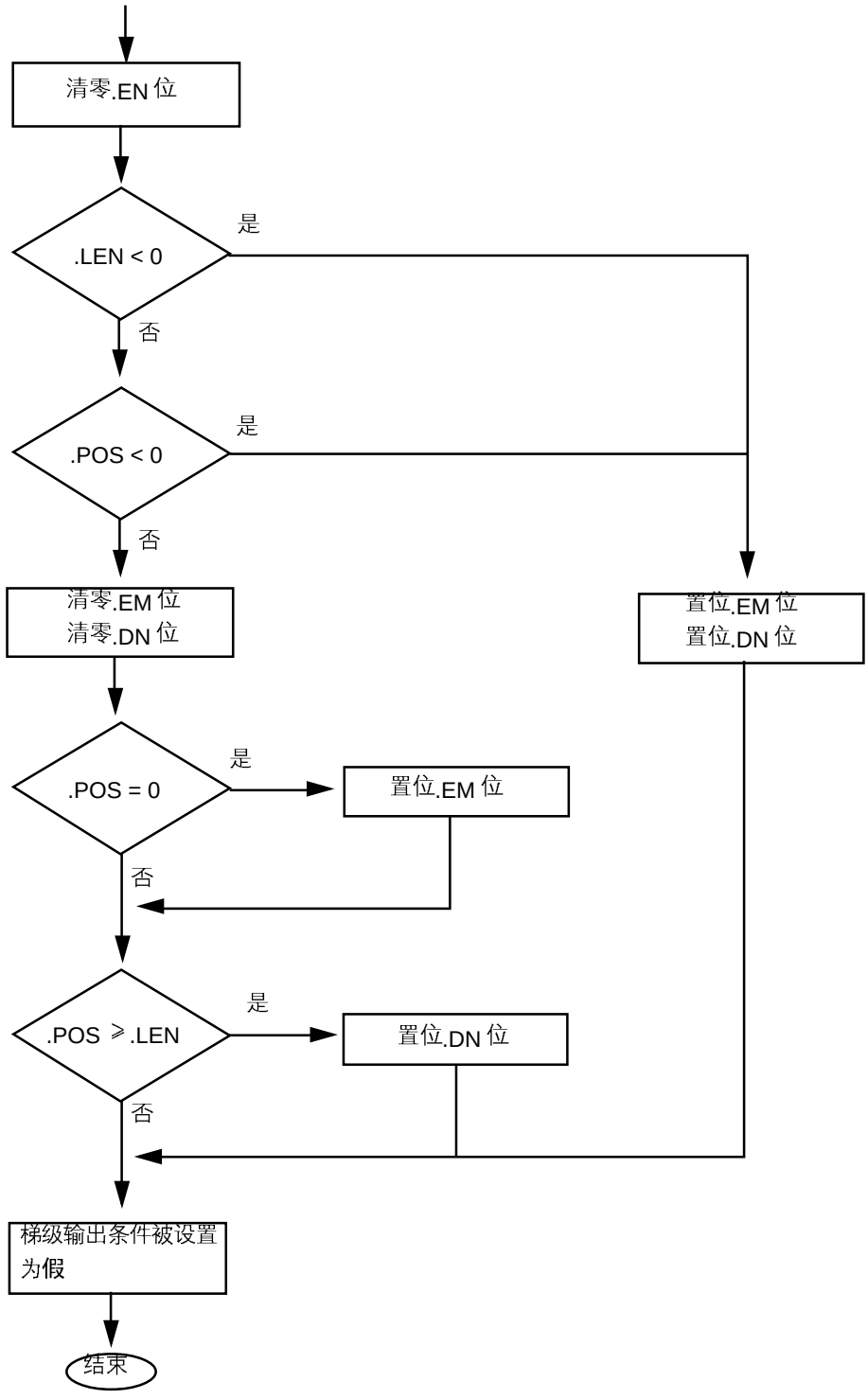
条件:	动作:

预扫描

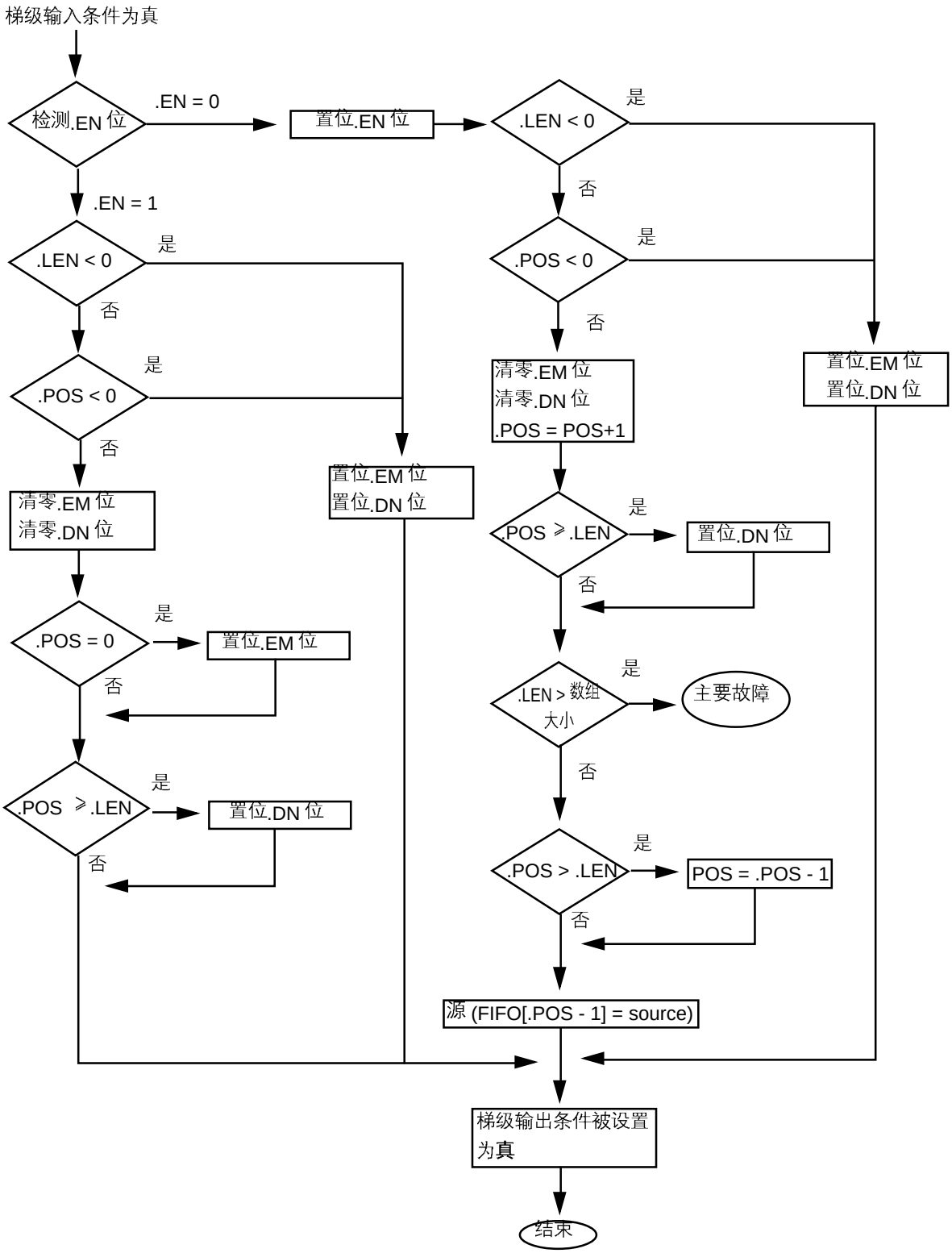

置位.EN 位以防止扫描开始时错误的装载
.LEN < 0
是
否
.POS < 0
是
否
清零.EM 位
清零.DN 位
置位.EM 位
置位.DN 位
.POS = 0
是
置位.EM 位
否
.POS ≥ .LEN
是
置位.DN 位
否
梯级输出条件被设置为假
结束

条件: **动作:**

梯级输入条件为假



条件: 动作:

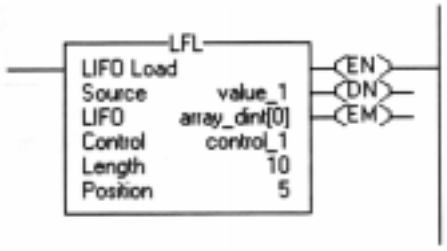

梯级输入条件为真
检测.EN 位
.EN = 0
置位.EN 位
.LEN < 0
是
否
.EN = 1
.LEN < 0
是
否
.POS < 0
是
否
清零.EM 位
清零.DN 位
置位.EM 位
置位.DN 位
.POS = 0
是
否
置位.EM 位
.POS ⩾.LEN
是
否
置位.DN 位
.POS < 0
是
否
清零.EM 位
清零.DN 位
.POS = POS+1
置位.EM 位
置位.DN 位
.POS ⩾.LEN
是
否
置位.DN 位
.LEN > 数组大小
是
否
主要故障
.POS > .LEN
是
否
POS = .POS - 1
源 (FIFO[.POS - 1] = source)
梯级输出条件被设置为真
结束

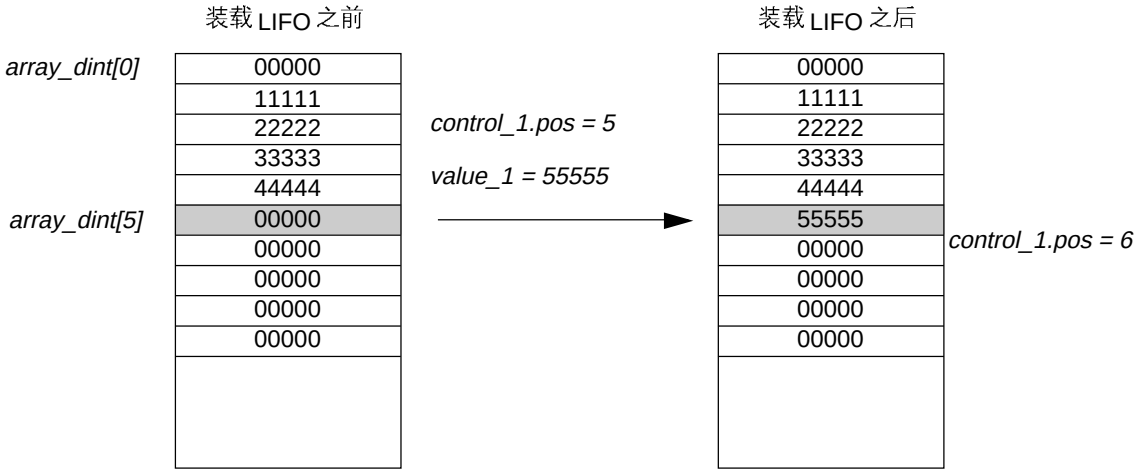
算术状态标志: 不影响

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 数组大小	4	20

LFL 指令举例:





当指令被使能时，FFL 指令把 *value_1* 装入 FIFO 的下一个位置，既本例的 *array_dint[5]* 内 。

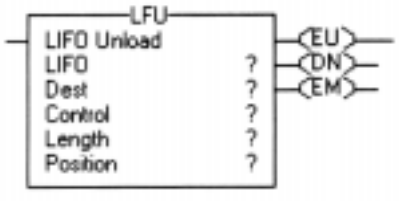
其他格式:

格式:	句法:
neutral 文本	`LFL(source,LIFO,control,length,position);`
ASCII 文本	`LFL source LIFO control length position`

相关指令: LFU，FFL，FFU

LIFO 卸载指令 (LFU)

LFU 是一条输出指令.



操作数:

操作数:	数据类型:	格式:	说明:
LIFO	DINT REAL	数组标签	要改变的 LIFO 指定 LIFO 的第一个元素 **不能**在标记处使用 CONTROL.POS
目的	**DINT** **REAL**	标签	从 LIFO 退出的值
控制	CONTROL	标签	用于指令操作的控制结构体 一般与其相关的 LFL 指令使 用相同的CONTROL结构体.
长度	DINT	立即数	LIFO 可以同时支持的元素的 最大数量
位置	DINT	立即数	在LIFO 内指令卸载数据的下 一个位置.一般起始值是 0

CONTROL 结构体:

助记符:	数据类型:	说明:
.EN	BOOL	使能位 - 标识 LFU 指令被使能.
.DN	BOOL	完成位被置位表明 LIFO 已满(.POS = .LEN)。
.EM	BOOL	栈空位表明 LIFO 是空的。如果.LEN ≤ 0 或 .POS < 0，则栈空位(.EM)和完成位(.DN)都被置位。
.LEN	DINT	长度指定 LIFO 内元素最大数量
.POS	DINT	位置表示指令已经装入 LIFO 的最后数据

说明:

LFU 指令卸载 LIFO 的位置值(.POS)内的数值并存 0 值于该位置。LFU 指令与 LFL 指令一起按后进 / 先出的顺序存储和返回数据。

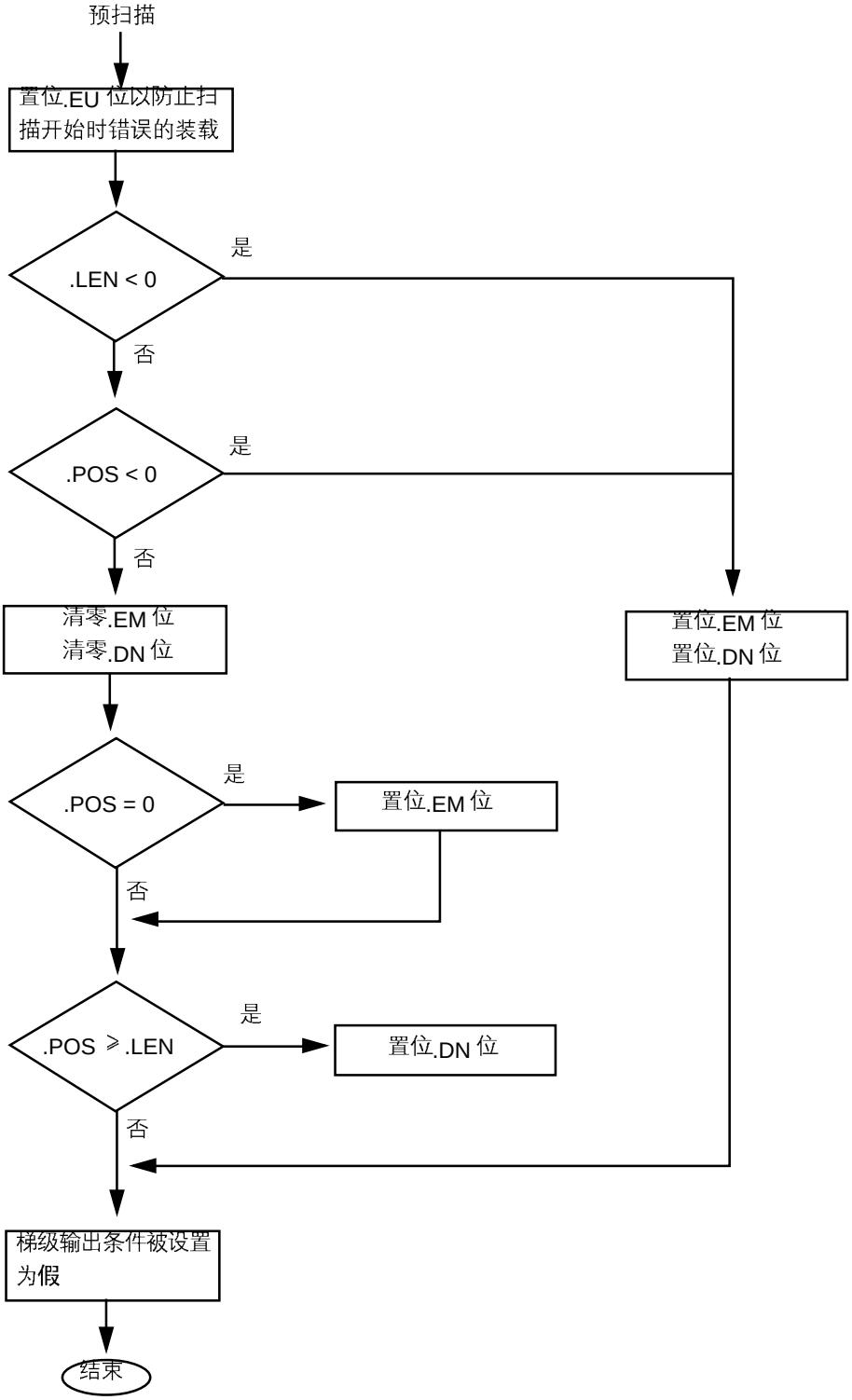
如果 LIFO 是 DINT 数据类型，则目的单元必须也是 DINT 数据类型；如果 LIFO 是 REAL 数据类型，则目的单元必须也是 REAL 数据类型。

当指令被使能时，LFU 指令卸载LIFO 的位置值(.POS)内的数值并存放该值于目的单元。指令每次被使能时都卸载一个数值并用0值替换该值，直到LIFO 栈空为止。如果 LIFO 是空的，则 LFU 指令返回 0 值到目的单元。

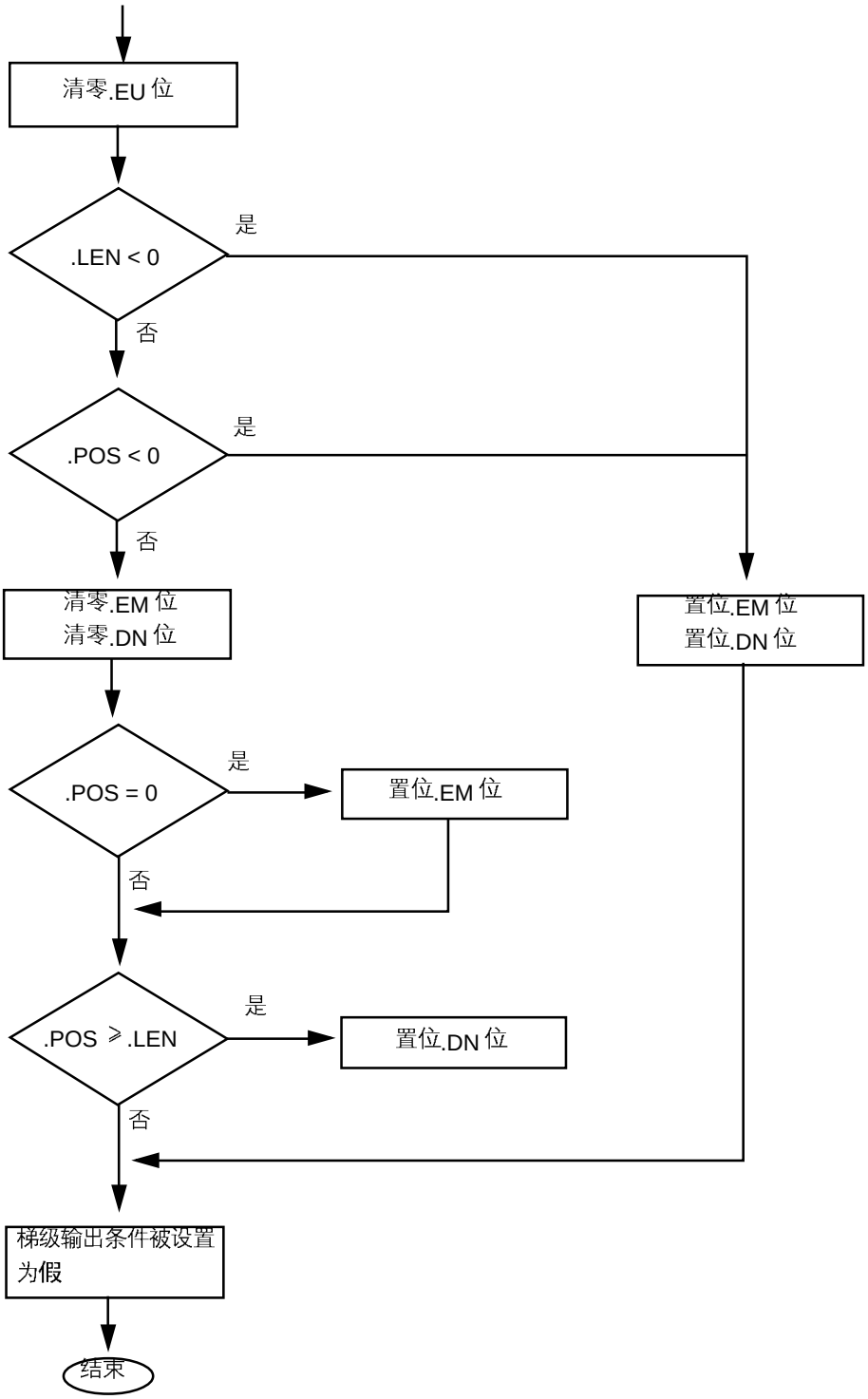
LFU 指令对连续存储器单元进行操作。详细信息参见附录 B - 3 页的将数组看作一存储块。

执行 :

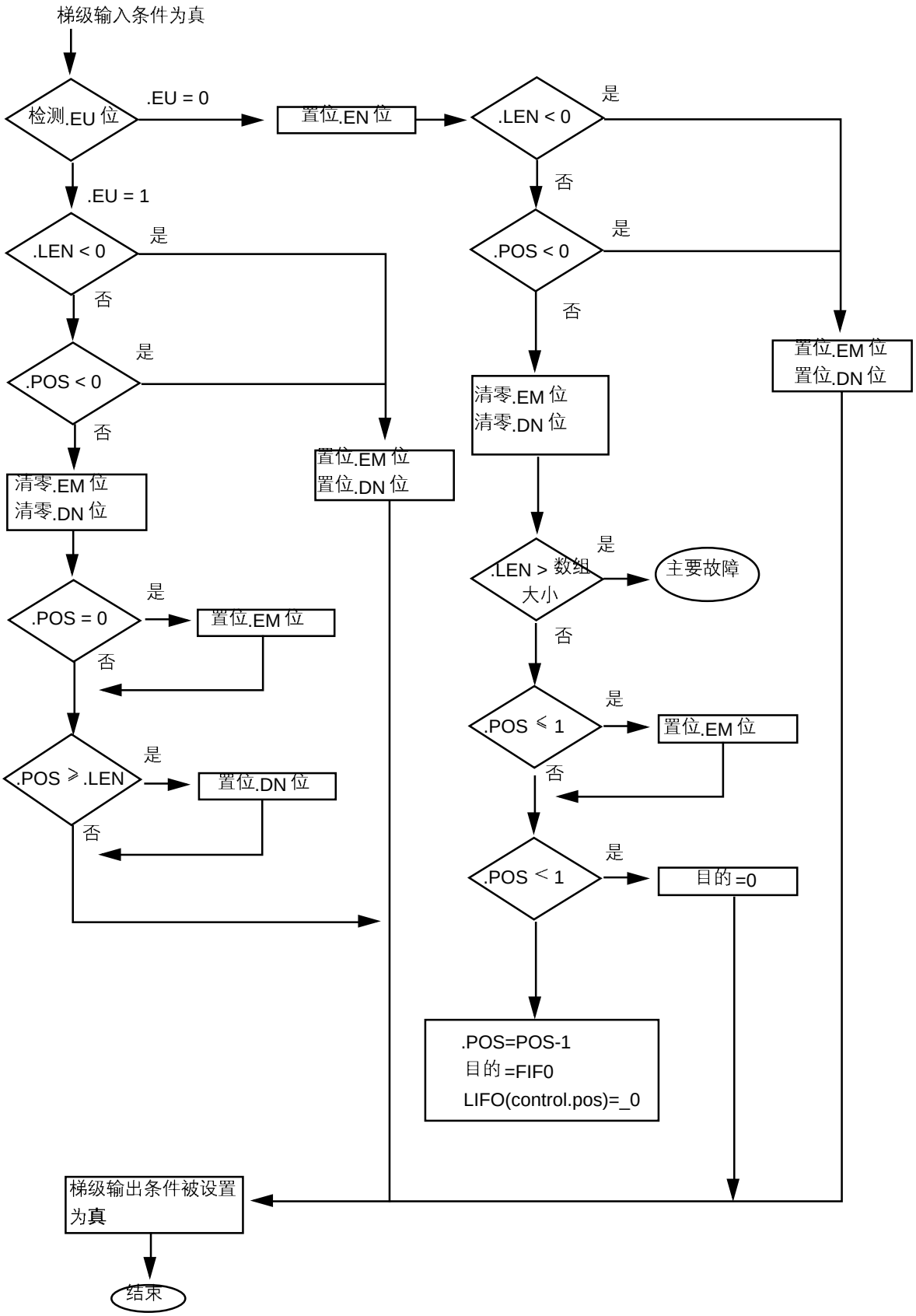
条件: 动作:


预扫描
置位.EU 位以防止扫描开始时错误的装载
.LEN < 0
是
否
.POS < 0
是
否
清零.EM 位
清零.DN 位
置位.EM 位
置位.DN 位
.POS = 0
是
置位.EM 位
否
.POS ⩾ .LEN
是
置位.DN 位
否
梯级输出条件被设置为假
结束

条件:	动作:
梯级输入条件为假	



条件: **动作:**

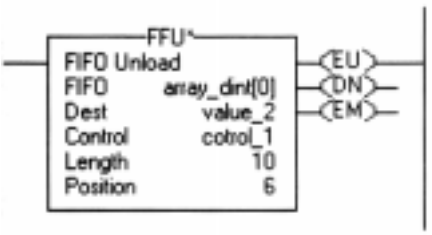

梯级输入条件为真
检测.EU 位
.EU = 0
置位.EN 位
.LEN < 0
是
否
.EU = 1
.LEN < 0
是
否
.POS < 0
是
否
.POS < 0
是
否
置位.EM 位
置位.DN 位
清零.EM 位
清零.DN 位
置位.EM 位
置位.DN 位
清零.EM 位
清零.DN 位
.LEN > 数组大小
是
主要故障
否
.POS = 0
是
置位.EM 位
否
.POS ≤ 1
是
置位.EM 位
否
.POS ≥ .LEN
是
置位.DN 位
否
.POS < 1
是
目的 =0
.POS=POS-1
目的 =FIF0
LIFO(control.pos)=_0
梯级输出条件被设置为真
结束

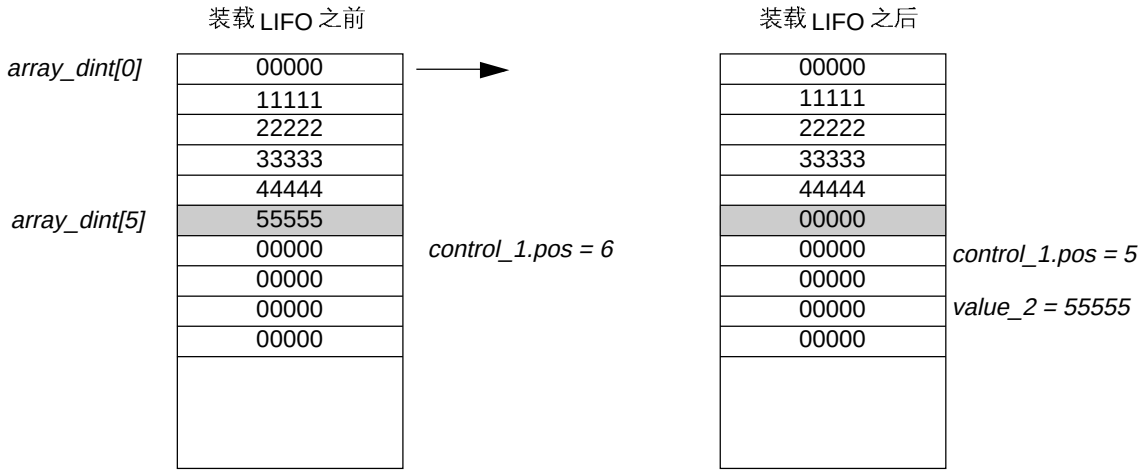
算术状态标志: 不影响

故障条件:

发生主要故障的条件:	故障类型:	故障代码:
长度 > LIFO 数组大小	4	20

LFU 指令举例:





当指令被使能时，LFU 指令卸载 *array_dint[5]* 进入 *value_2* 。

其他格式:

格式:	句法:
neutral 文本	`LFU (LIFO,destination,control,length,position);`
ASCII 文本	`LFU LIFO destination control length position`

相关指令: LFL，FFL，FFU

注释:

顺序器指令

(SQI，SQO，SQL)

简介

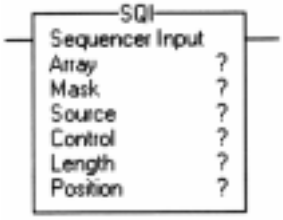
顺序器指令监控一致性且重复性的操作。

如果用户要:	使用下列指令:	参见页次:
当一步完成时检测	SQI	9 - 2
为下一步设置输出条件	SQO	9 - 6
装载参考条件于顺序器数组内	SQL	9 - 11

黑体字数据类型表示最优数据类型。如果指令的所有操作数都使用同一最优数据类型，则指令执行的速度快而且占用内存少。典型最优数据类型是 DINT 或 REAL。

顺序器输入指令 (SQI)

SQI 指令是一条输入指令。



操作数

操作数:	数据类型:	格式:	说明:
数组	DINT	数组标签	顺序器数组 指定顺序器数组的第一个元素 不能在此下标处用 CONTROL.POS
屏蔽	SINT INT **DINT**	标签 立即数	阻止或通过的位
源	DINT	标签	输入顺序器数组的数据
控制	CONTROL	标签	用于指令操作的控制结构体 一般用与 SQO 和 SQL 指令 相同的 CONTROL
长度	DINT	立即数	用于比较的数组(顺序器表) 内元素的数量
位置	DINT	立即数	在数组内的当前位置 一般初始值是 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，错误位被置位。
.LEN	DINT	长度指定顺序器数组的步数。
.POS	DINT	位置值表示指令当前正比较的元素位置

说明:

SQI 指令检测SQO/SQI顺序器指令对何时完成一步。当指令被使能时，SQI 指令通过屏蔽使源元素与一个数组元素作相等的比较。

一般与 SQO 和 SQL 指令用同一 CONTROL 结构体。

SQI 指令对连续存储器单元进行操作。详细信息参见附录 B - 3 页的将数组看作一存储块。

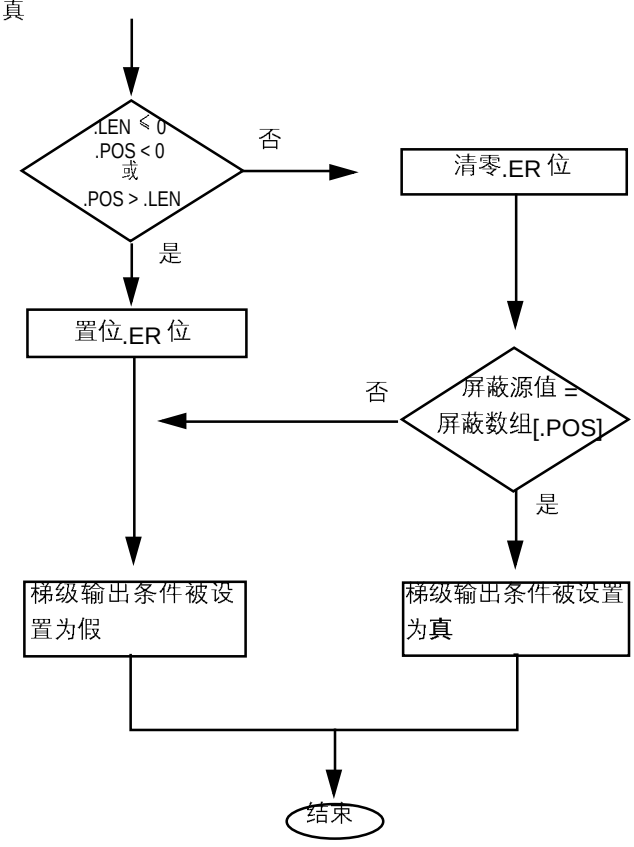
输入立即数作为屏蔽值

当输入屏蔽值时，编程软件默认为是十进制数值。如果想用其它格式输入屏蔽值，则需要在数值之前用相应的前缀。

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

执行

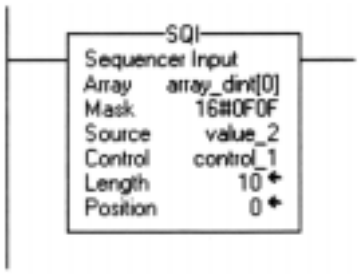
条件:	动作:
预扫描	梯级输出条件被设置为假。
梯级输入条件为假	梯级输出条件被设置为假
梯级输入条件为真	



算术状态标志: 不影响

故障条件: 无

SQI 指令举例:



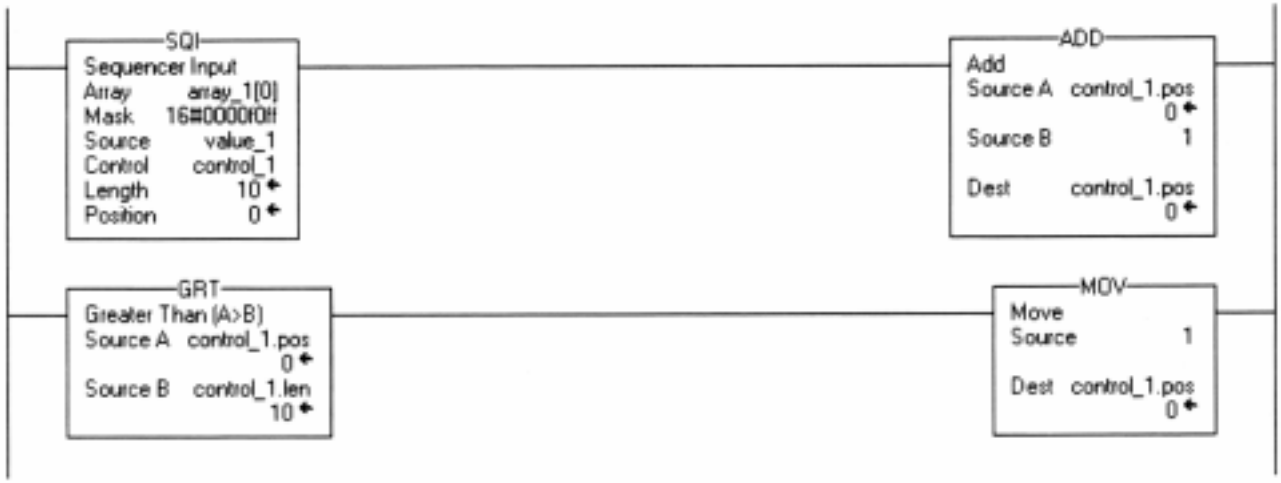
当指令被使能时，SQI 指令通过屏蔽使 value_2 与 array_dint 内的当前元素进行比较，确定结果是否为真。因为屏蔽比较为真，所以梯级输出条件变为真。

SQI 操作数 Z:	本例的数值(DINTs 以二进制显示):
源(Source)	xxxxxxxx xxxxxxxx xxxx0101 xxxx1010
屏蔽(Mask)	00000000 00000000 00001111 00001111
数组(Array)	xxxxxxxx xxxxxxxx xxxx0101 xxxx1010

屏蔽位的一个 0 值意味着处于此位置的值不被比较 (本例中用 xxxx 指明)。

只用 SQI 指令不用 SQO

如果用户只用 SQI 指令而不用SQO 指令，则用户必须在外部增加顺序器数组。



SQI 指令比较源数值

ADD 指令增加顺序器数组

GRT 确定顺序器数组的检测值是否有效。

MOV 指令在一次从头到尾完成顺序器数组的所有步之后，复位位置值。

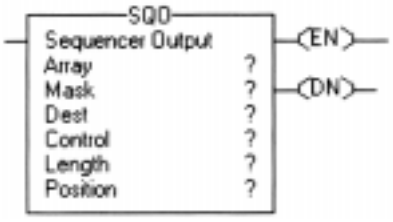
其他格式:

格式:	句法:
NEUTRAL 文本	`SQI(array,mask,source,control,length,position);`
ASCII 文本	`SQI array mask source control length position`

相关指令: SQO, SQL

顺序器输出指令 (SQO)

SQO 指令是一条输出指令。



操作数

操作数:	数据类型:	格式:	说明:
数组	DINT	数组标签	顺序器数组 指定顺序器数组的第一个元素 不能在此下标处用 CONTROL.POS
屏蔽	SINT INT **DINT**	标签	立即数 阻止或通过的位
目的	DINT	标签	从顺序器数组输出的数据
控制	CONTROL	标签	操作的控制结构体 一般用与 SQI 和 SQL 指令相同的 CONTROL
长度	DINT	立即数	数组(顺序器表)内输出元素的数量
位置	DINT	立即数	在数组内的当前位置 一般初始值是 0

CONTROL 结构体:

助记符:	数据类型:	说明:
.EN	BOOL	使能位标识 SQO 指令被使能。
.DN	BOOL	当所有指定的元素已经被移入目的单元后，置位完成位。
.ER	BOOL	当长度值(.LEN) ≤ 0，位置值(.POS)< 0，或 .POS > .LEN 时，错误位被置位。
.LEN	DINT	长度指定顺序器数组的步数。
.POS	DINT	位置值表示制器当前正处理的元素的位置。

说明:

SQO指令设置SQO/SQI 顺序器指令对的下一步输出条件。当指令被使能时，SQO 指令增加位置值，通过屏蔽传送该位置内的数据，并且把结果存入目的单元。如果位置值(.POS)> 长度值(.LEN)，则指令返回到顺序器数组的开始处，并从位置值(.POS)= 1 处继续执行。

SQO 指令一般与 SQI 和 SQL 指令使用相同的 CONTROL 结构体。

SQO 指令对连续存储器单元进行操作。详细信息参见附录 B - 3 页的将数组看作一存储块。

输入立即数作为屏蔽值

当输入屏蔽值时，编程软件默认为是十进制数值。如果想用其它格式输入屏蔽值，则需要在数值之前用相应的前缀。

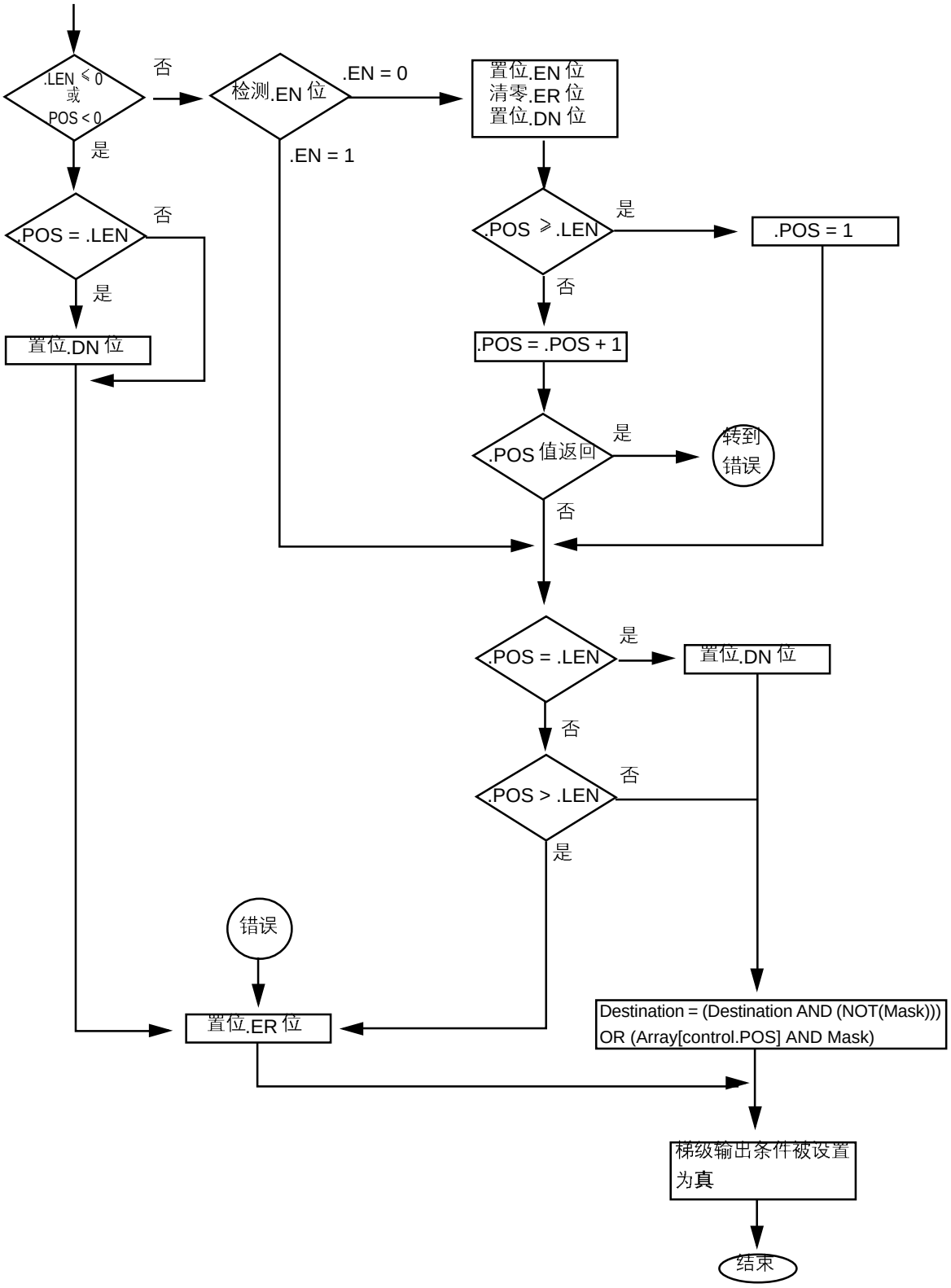
前缀:	说明:
16#	十六进制 例如: 16#0F0F
8#	八进制 例如: 8#16
2#	二进制 例如: 2#00110011

执行:

条件:	动作:
预扫描	置位使能位(.EN)以防止程序扫描开始时错误的装载。 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: **动作:**

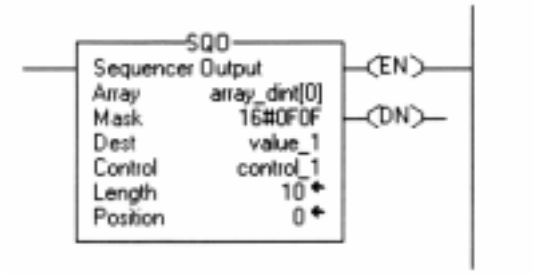
梯级输入条件为真



算术状态标志: 不影响

故障条件: 无

SQO 指令举例:



当指令被使能时，SQO 指令增加位置值，通过屏蔽传递 *array_dint* 位置值内的数据，并把结果存入 *value_1* 内 。

SQO 操作数	本例的数值(DINTs 以二进制显示)
源	xxxxxxxx xxxxxxxx xxxx0101 xxxx1010
屏蔽	00000000 00000000 00001111 00001111
目的	xxxxxxxx xxxxxxxx xxxx0101 xxxx1010

屏蔽位中的一个 0 值意味着该位置的位不改变 (本例中用 xxxx 指明)。

SQI 与 SQO 指令一起使用

如果用户成对使用 SQI 和 SQO 指令，则两条指令一定要使用同一控制，长度，和位置值。

复位 SQO 指令的位置值

每次控制器从编程方式进入运行方式，SQO 指令都清零(初始化)位置值(.POS)值。要复位位置值(.POS)到初始值 (.POS = 0)，可以用一条 RES 指令清零位置值来完成。本例用首次扫描位的位状态来清零位置值(.POS)值。

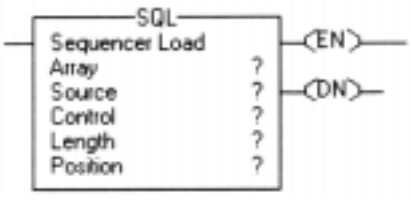
其他格式:

格式:	句法:
neutral 文本	SQO*(array,mask,destination,control,length,position);*
ASCII 文本	SQO *array mask detination control length position*

相关指令: SQI，SQL

顺序器装载指令 (SQL)

SQL 指令是一条输出指令。



操作数

操作数:	数据类型:	格式:	说明:
数组	DINT	数组标签	顺序器数组 指定顺序器数组的第一个元素 不能在此下标处用 CONTROL.POS
源	SINT INT **DINT**	标签	装入顺序器数组的输入数据
控制	CONTROL	标签	用于指令操作的控制结构体 一般用与 SQI 和 SQO 指令相同的 CONTROL
长度	DINT	立即数	装载到数组(顺序器表)内元素的数量
位置	DINT	立即数	数组内的当前位置 一般初始值是 0

CONTROL 结构体:

助记符:	数据类型:	说明:
.EN	BOOL	使能位 - 标识 SQL 指令被使能。
.DN	BOOL	当所有指定的元素已经被装入数组后，置位完成位。
.ER	BOOL	如果长度值(.LEN)≤ 0，位置值(.POS)< 0，或 .POS > .LEN，则错误位被置位。
.LEN	DINT	长度 - 指定顺序器数组的步数。
.POS	DINT	位置 - 值表示控制器当前正处理的元素的位置。

说明:

SQL指令把参考条件装入顺序器数组内，当指令被使能时，SQL指令增加位置值到顺序器数组的下一个位置，并装载源数值到该位置。

如果完成位被置位或位置值(.POS)> 度值(.LEN)，则指令设置位置值(.POS)= 1。

一般SQL指令与 SQI 和 SQO 指令用相同的CONTROL结构体。

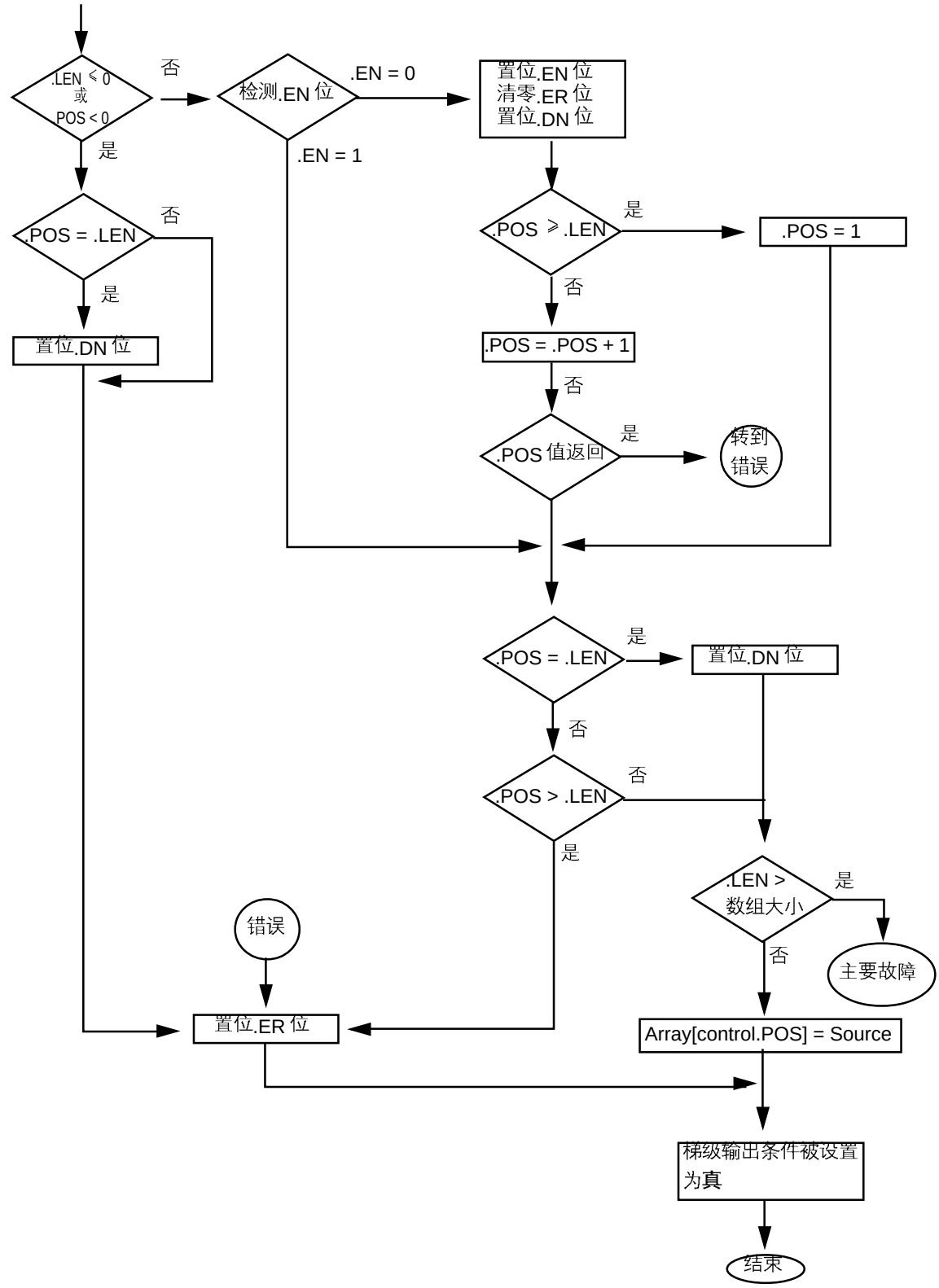
SQL 指令对连续存储器单元进行操作。详细信息参见附录 B - 3 页的将数组看作一存储块。

执行:

条件:	**动作**:
预扫描	置位使能位(.EN)以防止程序扫描开始时错误的装载。 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: **动作:**

梯级输入条件为真

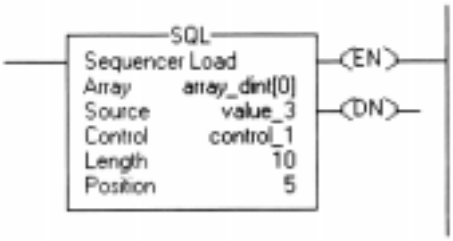


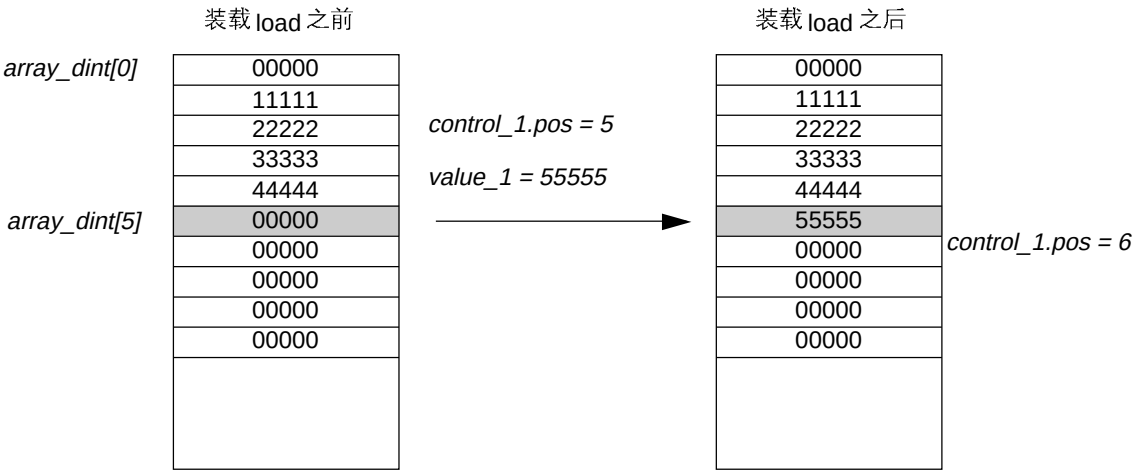
算术状态标志: 不影响

故障条件:

发生主要故障的条件:	故障类型:	故障代码:
长度 > 数组的大小	4	20

SQL 指令举例:





当指令被使能时，SQL 指令装载 *value_3* 到顺序器数组的下一个位置，在本例中是 *array_dint[5]*

其他格式:

格式:	句法:
neutral 文本	`SQL(array,source,control,length,position);`
ASCII 文本	`SQL array source control length position`

相关指令: SQI，SQO

程序控制指令

(JMP，LBL，JSR，RET，SBR，TND，MCR，UID，UIE，AFI，NOP)

简介

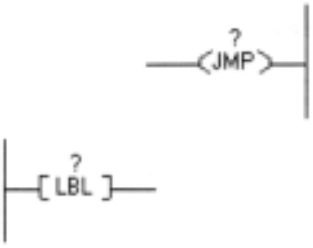
用程序控制指令改变程序的流程。

如果用户要：	使用下列指令：	参见页次：
跳过程序中不需要	JMP LBL	10 -2
一直执行的部分。 跳转到一独立的子程序， 向子程序传递数据，执行子 程序，并返回结果。	JSR SBR RET	10 - 4
标记暂时结束以停止程序执行。	TND	10 - 10
禁止一个程序区域内的所有梯级。	MCR	10 - 11
使用户任务无效。	UID	10 - 14
使用户任务使能。	UIE	10 - 15
使一梯级无效。	AFI	10 - 16
在程序内插入一个空操作。	NOP	10 - 17

跳转到标号指令(JMP)
标号 (LBL)

JMP 指令是一条输出指令。

LBL 指令是一条输入指令。



操作数

操作数:	数据类型:	格式:	说明:
JMP 指令			
标号名称		标号名称	输入相关的 LBL 指令名称
LBL 指令			
标号名称		标号名称	执行跳转到有相应参考标号名称的 LBL 指令

说明:

用 JMP 和 LBL 指令可以略过部分梯形图逻辑。当指令被使能时，JMP 指令跳到其引用的 LBL 指令，控制器从该处继续执行。当指令被禁止时，JMP 指令不影响梯形图程序的执行。

JMP 指令可以向前或向后跳转执行梯形图程序。向前跳转到标号可以通过略过部分梯形图逻辑直到需要的程序，从而节省程序扫描时间。向后跳转使控制器重复执行梯形图逻辑(逻辑迭代)。

注意向后跳转的次数不要太多。否则看门狗定时器可能会超时，因为控制器不能到达程序的末尾。这样会导致控制器发生故障。

注意：跳过的程序不被扫描，请把重要程序放在跳转区域之外

LBL 指令是具有同一标号名称的JMP 指令的跳转目标。**要确保 LBL 指令是其所在梯级的第一条指令。**

在一个程序内标号名称必须是唯一的，标号名称可以是

- 最多可以有 40 个字符
- 可以包含字母，数字，和下划线(_)

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真。
梯级输入条件为真	梯级输出条件被设置为真。 执行跳转到包含具有引用的标号名称的 LBL 指令的梯级。

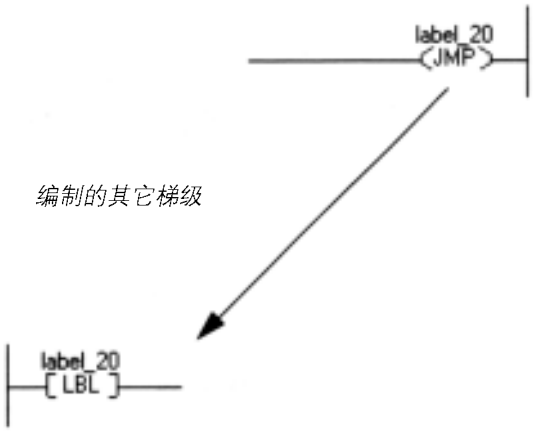
算术状态标志:

不影响

故障条件:

发生主要故障的条件:	故障类型:	故障代码:
标号不存在	4	42

JMP/LBL 指令举例:



当 JMP 指令被使能时，指令跳过其下面的逻辑梯级，执行包含名称是 label_20 的 LBL 指令梯级的以后梯级。

其他格式:

格式:	句法:
neutral 文本	`JMP(label_name);` `LBL(label_name);`
ASCII 文本	`JMP label_name` `LBL label_name`

相关指令:

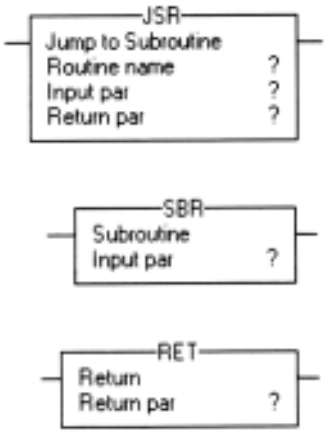
JSR，SBR，RET，FOR，BRK

跳转到子程序指令 (JSR)
子程序指令 (SBR)
返回 (RET)

JSR 指令是一条输出指令。

SBR 指令是一条输出指令.

RET 指令是一条输出指令.



操作数:

操作数:	数据类型:	格式:	说明:
JSR 指令			
子程序名称	ROUTINE	名称	要执行的子程序
输入参数	SINT INT DINT REAL 结构体	立即数 标签 数组标签	传递到子程序的参数
返回参数	SINT INT DINT REAL 结构体		从子程序接收的参数(0-n)
SBR 指令			
输入参数	SINT INT DINT REAL 结构体	标签 数组标签	从 JSR 指令接收的参数
RET 指令			
返回参数	SINT INT DINT REAL 结构体	立即数 标签 数组标签	返回到 JSR 指令的参数

注意: 输入参数和与之匹配的返回参数必须是相同的数据类型，否则会出现不可预知的数据或发生危险操作.

说明:

JSR，SBR 和 RET 指令使逻辑执行转到程序中的独立的子程序，对子程序进行一次扫描，然后返回到程序的转移点。

当指令被使能时，JSR指令使逻辑执行转到指定的子程序，如果需要，向子程序传递参数。如果没有输入参数，则控制经由 JSR 指令进入子程序的第一梯级。

当指令被使能时，如果有输入参数，JSR 指令传递它的输入参数，并使执行转到子程序的第一条梯级。SBR 指令接收输入参数并复制这些数值到指定的标签。JSR 指令输入参数的数量和数据类型需要与 SBR 指令相匹配。如果 JSR 指令输入参数的数量比相应的 SBR 指令的输入参数少，控制器出现主要错误。JSR 指令的输入参数数量可以多于相关的 SBR 指令的输入参数，而不会发生故障 。

如果要向子程序传递参数，则需要使用 SBR 指令。如果用户用了一条 SBR 指令，则它必须是子程序第一梯级的第一条指令。这一可选的 SBR 指令确定存储进入参数的标签。用户也可以使用 SBR 指令而不输入参数，用来表明其驻留的程序是子程序。

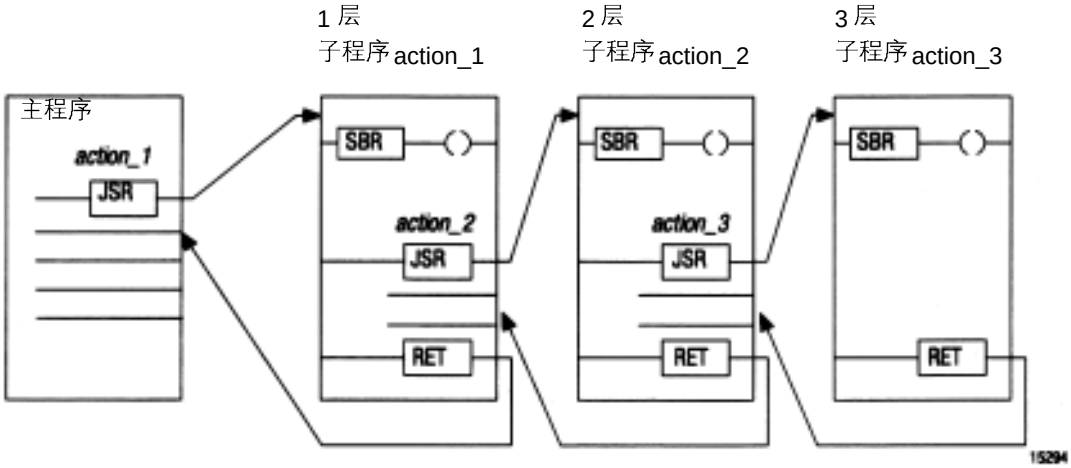
用户不能把 SBR 指令放入主程序内。

如果用户要返回参数到 JSR 指令，必须使用 RET 指令。 RET 指令结束子程序，而且根据需要向 JSR 指令返回参数。一个子程序可以有多条 RET 指令。

当指令被使能时，RET 指令传递它的参数，并且从相关的 JSR 指令的下一梯级继续执行梯形图程序。RET 指令返回参数的数量和数据类型应该与 JSR 指令的匹配。如果在 RET 指令返回参数的数量少于 JSR 指令返回参数的数量，控制器会发生主要故障。RET 指令的返回参数可以多于相关的 JSR 指令的返回参数，而不会引起故障。

当指令被禁止时，RET 指令不影响逻辑的执行。控制器继续执行当前的子程序。

对于用户可以使用的程序嵌套的层数，和传递或返回的参数数量，只要控制器内存容量允许，就没有限制。

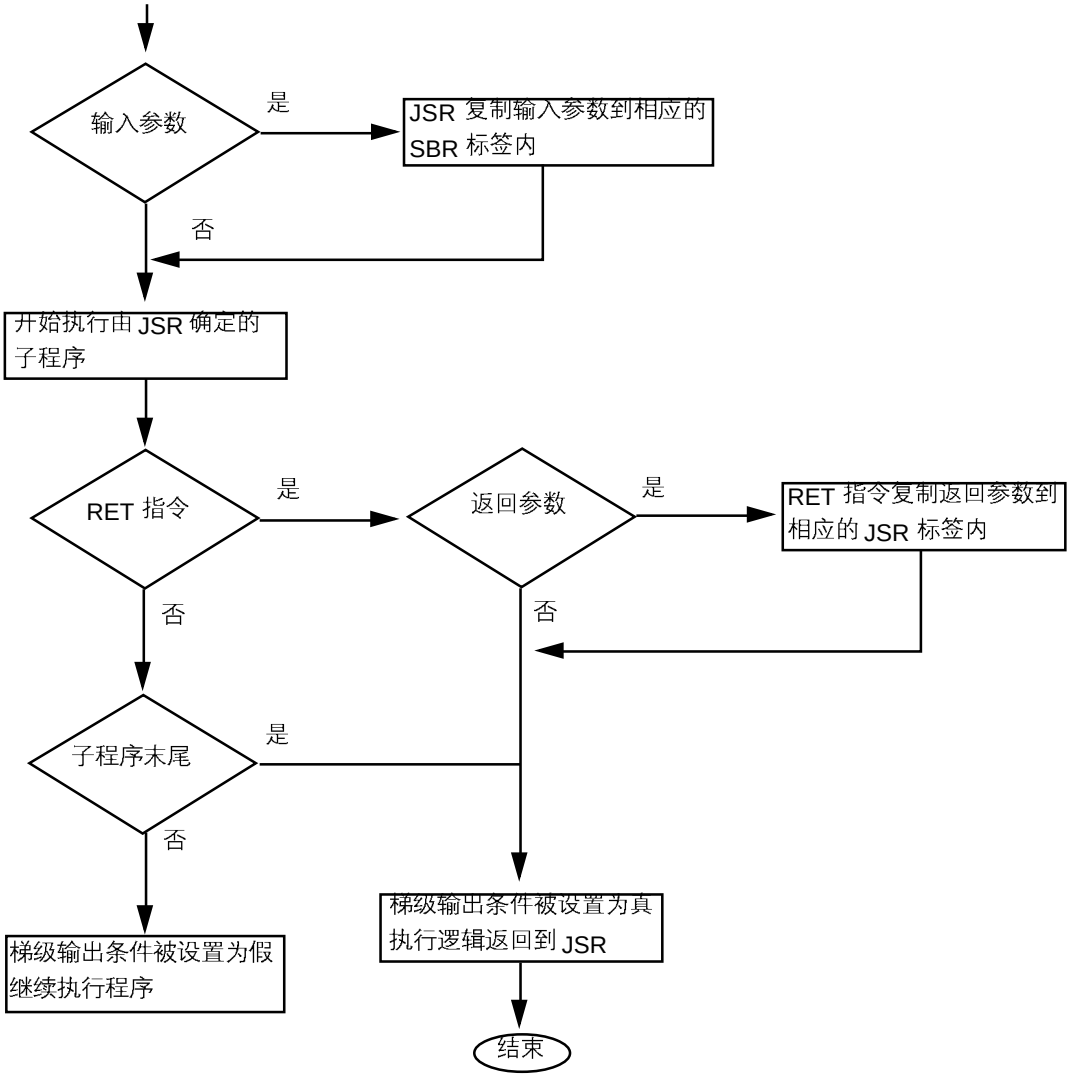


JSR，SBR 和 RET 指令通过数值把参数传入或传出子程序。也就是说指令使用额外的执行时间和存储器来复制数值。用户可以通过从子程序内直接访问程序范围内或控制器范围内的数据，而不是传递数值，以减少执行时间。

用户可以传递单个数组元素，整个数组，单个结构体元素，或整个结构体。数组和结构体的复制与使用COP指令拷贝数值方法相同。这里建议用户传递数组或结构体参数分别到其各自的相同类型的数组或结构体参数内。

执行:

条件:	动作:
预扫描	梯级输出条件被设置为假。 控制器执行所有子程序而与梯级条件无关，但是忽略RET指令。全部输入参数都被传递到子程序内。全部返回参数被传递，但是RET指令不退出子程序。这样子程序内的全部梯级都被预扫描。 如果对同一子程序存在递归调用，则子程序只有第一次被预扫描。如果对同一子程序存在多重调用(非-递归)，则子程序每次都被预扫描。
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	



算术状态标志: 不影响

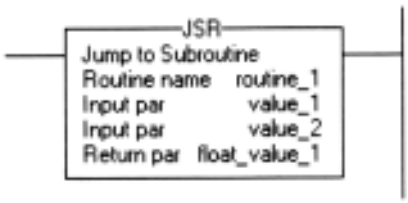
故障条件:

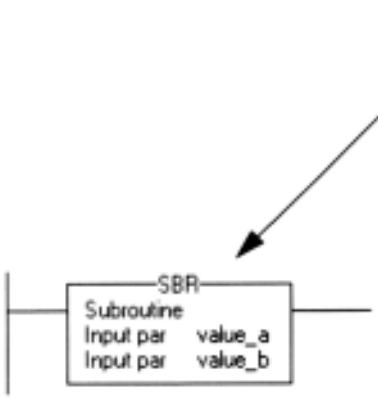
发生主要故障的条件:	故障类型:	故障代码:
JSR 指令的输入参数少于 SBR 指令的输入参数。	4	31
RET 的返回参数少于 JSR 指令的返回参数。	4	31

JSR/SBR/RET 指令举例:

例 1

当指令被使能时，JSR 指令传递 *value_1* 和 *value_2* 到 *routine_1* 内。

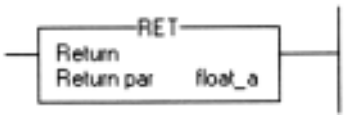




SBR 指令从 JSR 指令接收 *value_1* 和 *value_2* 并且分别复制这二值到 *value_a* 和 *value_b* 内。从该子程序继续执行程序。

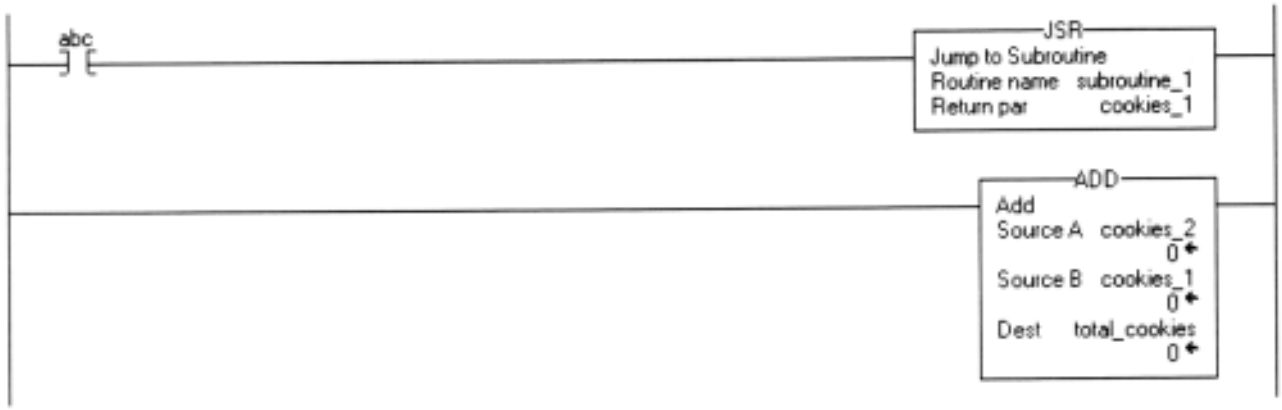
[编制的其它梯级]

当指令被使能时，RET 指令发送 *float_a* 到 JSR 指令。JSR 指令接收 *float_a* 并且复制该值到 *to float_value_1* 内.从 JSR 指令的下一条指令继续执行逻辑。

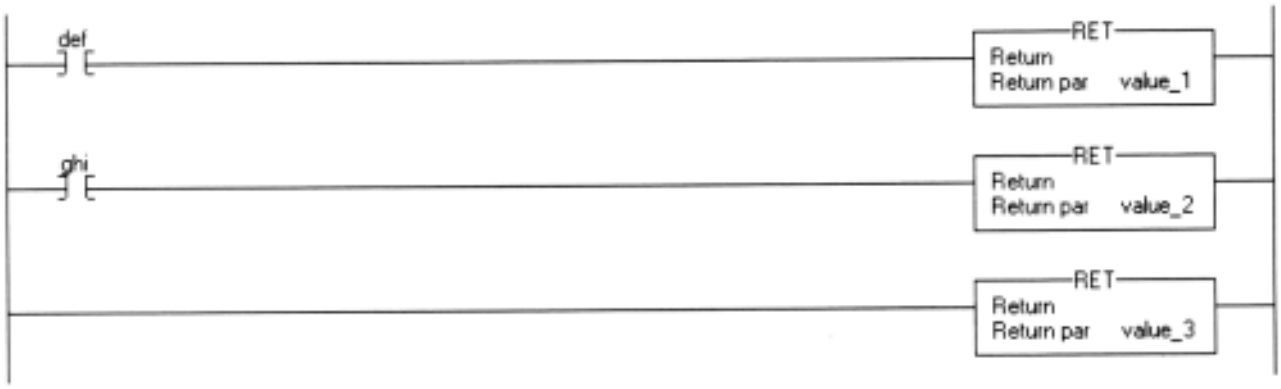


例 2

主程序



子程序 _1



如果abc 变为真，则JSR指令被使能，控制转移到*subroutine_1* 。如果def 被使能，则RET 指令返回*value_1* 到JSR指令的cookies_1 参数内，子程序的其余部分不被扫描。如果 ghi 被使能，则 RET 指令返回 *value_2* 到 JSR 指令的 cookies_1 参数内，子程序的其余部分不被扫描。如果 def 和 ghi 都不被使能，则 RET 指令返回 *value_3* 到 JSR 指令的 *cookies_1* 参数，子程序的其余部分不被扫描，然后 ADD 指令加 *cookies_1* 的值到 *cookies_2* 并存放结果于 *total_cookies* 内。

其他格式:

格式:	句法:
neutral 文本	`JSR (routine_name,input_1,...input_n,return_1,..return_n);`
	`SBR (routine_name,input_1,...input_n);`
	`RET (return_1,...return_n);`
ASCII 文本	`JSR routine_name input_1 ... input_n return_1 ... return_n`
	`SBR routine_name input_1 ... input_n`
	`RET return_1 ... return_n`

相关指令: JMP，LBL，FOR，BRK

暂停指令 (TND)

TND 指令是一条输出指令。

操作数: 无

说明: TND 指令可以作为一个分界线。

当指令被使能时，TND 指令使控制器程序只执行到该指令。

当指令被使能时，TND 指令担当程序的末尾。如果控制器扫描到一条 TND 指令，则控制器转移到当前程序的结束处。如果 TND 在一个子程序内，则控制返回到调用它的程序。如果 TND 指令在一个主程序内，则控制返回到当前任务的下一个程序。

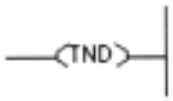
执行:

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。 终止执行当前程序。

算术状态标志: 不影响

故障条件: 无

TND 指令举例: 用户可以在调试或故障诊断时使用 TND 指令使程序执行到一确定点。然后在程序内进一步移动 TND 指令到用户调试程序的新部分。



当 TND 指令被使能时，控制器停止扫描当前程序。

其他格式:

格式:	句法:
neutral 文本	`TND();`
ASCII 文本	`TND`

相关指令: AFI，MCR，NOP

主控复位指令 (MCR)

MCR 指令是一条输出指令。

操作数: 无

说明: MCR 指令成对使用，用来创建一个程序区域，可以用 MCR 指令使该区域内的所有梯级无效。

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。当区域被禁止时，控制器仍扫描 MCR 区域内的梯级，但是因为在该区域内的所有输出被禁止，从而节省了扫描时间 。

当 MCR 区域被禁止时，所有在 MCR 区域内的指令梯级输入条件都为假。

用户编制 MCR 区域程序时，应注意:

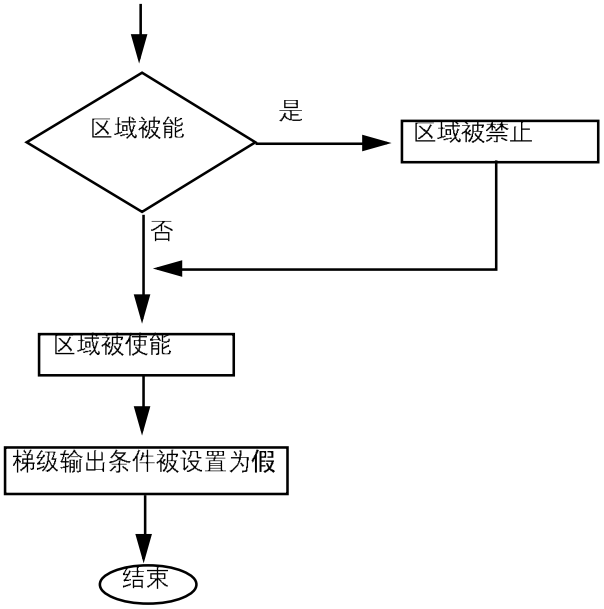
- 必须用一条无条件的 MCR 指令结束区域。
- 不能在 MCR 区域内嵌套另一个 MCR 区域。
- 不要跳转到MCR区域。如果该区域为假，则跳转到该区域会激活从跳入点到区域结束的部分。
- 如果MCR区域持续到程序的末尾，则没必要在区域的结束处编制一条 MCR 指令。

重要: 不能把 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 电源的能力。

注意: 不要重叠或嵌套 MCR 区域。每个 MCR 区域必须是独立的而且是完整的。如果它们被重叠或嵌套，则可能会发生不可预料的机器运转，这样会造成设备损坏或人身伤害。

应该把重要操作编程在 MCR 区域之外。如果在 MCR 区域内起动诸如计时器等类似指令，则当区域被禁止时，指令停止执行而且计时器被清零。

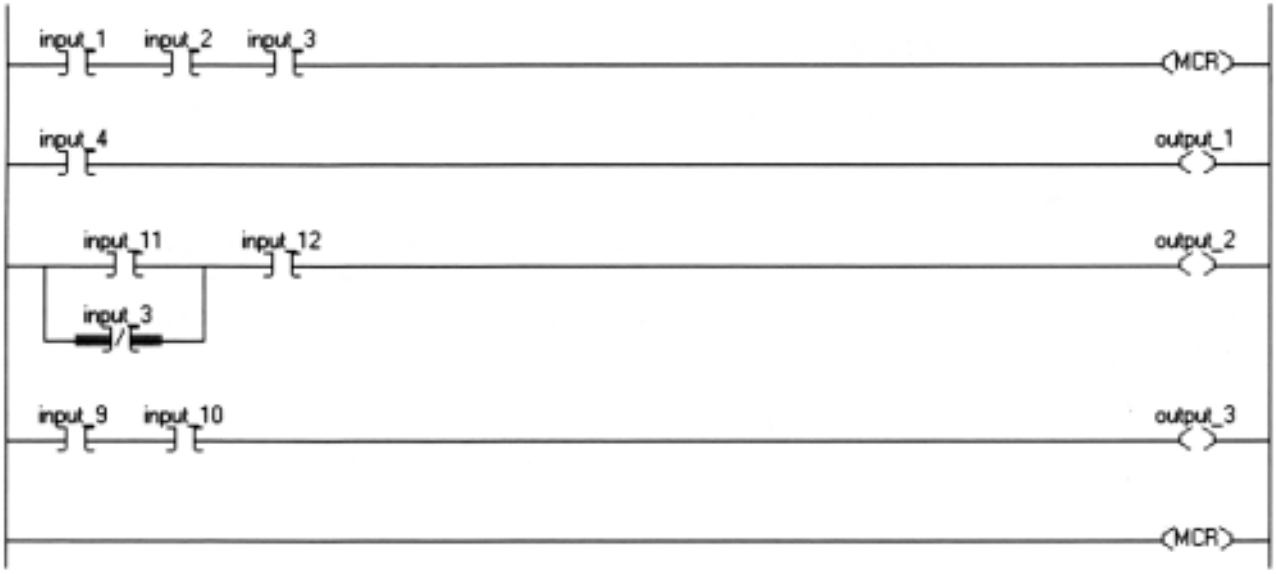
执行:

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

算术状态标志: 不影响

故障条件: 无

MCR 指令举例:



当第一条 MCR 指令被使能时 *(input_1，input_2，*和 *input_3* 被置位)，控制器执行 MCR 区域内的梯级(两条 MCR 指令之间的)，并且根据输入条件置位或复位输出。

当第一条 MCR 指令被禁止时 *(input_1，input_2，*和 *input_3* 不都被置位)，控制器执行 MCR 区域内的梯级(两条 MCR 指令之间的)，但是在 MCR 区域内的所有梯级，其输入条件都变为假，而与实际输入条件无关。

其他格式:

格式:	句法:
neutral 文本	`MCR();`
ASCII 文本	`MCR`

相关指令: AFI，NOP，TND

禁止用户中断指令 (UID)

UID 指令是一条输出指令。

操作数: 无

说明: UID 指令临时禁止切换换用户任务。

如果 UID 指令被使能时则继续执行当前任务，而不能被高优先级的任务中断，除非执行 UIE 指令或到达程序的末尾。UID 指令不能禁止一个故障子程序或故障任务。

当指令被使能时，UID 指令增加内部计数器的数值，只要计数器的值不为零，当前正执行的任务就不会被中断。用户可以嵌套使用 UID 指令达 65,535 层。

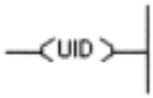
执行:

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	防止当前任务被高优先级的任务中断。 UID 内部计数器增加。 梯级输出条件被设置为真。

算术状态标志: 不影响。

障条件: 无

UID 指令举例:



当指令被使能时，UID 指令禁止切换用户任务。

其他格式:

格式:	句法:
neutral 文本	`UID();`
ASCII 文本	`UID`

相关指令: UIE

用户中断使能指令(UIE)

UIE 指令是一条输出指令。

操作数: 无

说明: UIE 指令再使能用户任务之间切换。

如果 UIE 指令被使能且内部计数器的值大于零，则计数器的值减少。当计数器的值等于零时，当前执行的任务可以再次被中断。此时执行任何先前被禁止中断的高优先级任务 。

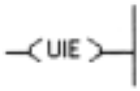
执行:

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	UID 内部计数器值减少。 如果内部计数器等于 0，则高优先级任务可以中断当前任务。 梯级输出条件被设置为真。

算术状态标志: 不影响。

障条件: 无

UIE 指令举例:



当指令被使能时，UIE 指令再使能用户任务之间切换

其他格式:

格式:	句法:
neutral 文本	`UIE();`
ASCII 文本	`UIE`

相关指令: UID

恒假指令 (AFI)

AFI 指令是一条输入指令。

操作数: 无

说明: AFI 指令设置它的梯级输出条件为假。

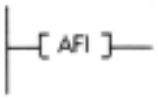
执行:

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假。

算术状态标志: 不影响

故障条件: 无

AFI 指令举例: 当用户调试程序时，AFI 指令临时禁止一个梯级。



当指令被使能时，AFI 指令禁止在该梯级的所有指令。

其他格式:

格式:	句法:
neutral 文本	`AFI();`
ASCII 文本	`AFI`

相关指令: MCR，NOP，TND

空操作指令 (NOP)

NOP 指令是一条输入和输出指令。

操作数: 无

说明: NOP 指令的功能相当于占位符.

编程时可以放置 NOP 指令于梯级的任何地方。当指令被使能时 NOP 指令执行空操作。当指令被禁止时，NOP 指令也执行空操作。

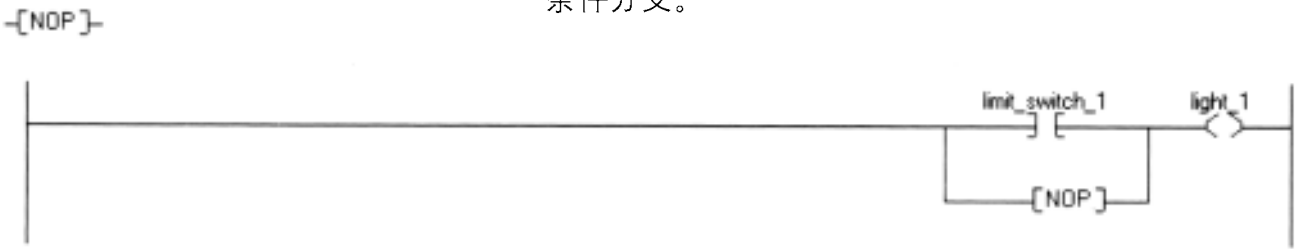
其他格式:

格式:	句法:
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。

算术状态标志: 不影响

故障条件: 无

NOP 指令举例: 当放置 NOP 指令于分支内时，该指令可以用于设置无条件分支。



NOP 指令旁路 XIC 指令使输出使能。

其他格式:

格式:	句法:
neutral 文本	`NOP();`
ASCII 文本	`NOP`

相关指令: AFI，MCR，TND

注释:

循环 / 终止循环指令

(FOR，BRK，RET)

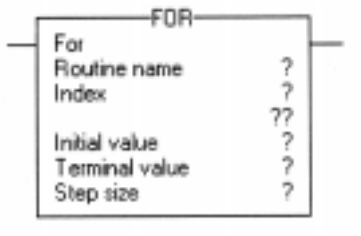
简介

用FOR指令重复调用子程序。用BRK指令中断一个子程序的执行。

如果用户要:	使用下列指令:	参见页次:
重复执行一个子程序	FOR	11 - 2
终止一个子程序的重复执行	BRK	11 - 5
返回到FOR指令	RET	11 - 6

For 指令

FOR 指令是一条输出指令。



操作数:

操作数:	数据类型:	格式:	说明:
子程序名称	ROUTINE	子程序名称	要执行的子程序
索引	DINT	标签	子程序已经执行的次数
初始值	SINT	立即数	索引的初始值
	INT	标签	
	DINT		
终止值	SINT	立即数	停止执行子程序的值
	INT	标签	
	DINT		
每步大小	SINT	立即数	每次 FOR 指令执行子程序时
	INT	标签	加到索引值的数量
	DINT		

说明:

FOR 指令重复执行子程序。

当指令被使能时，FOR指令重复执行子程序，直到索引值超过终止值。该指令不向子程序传递参数。

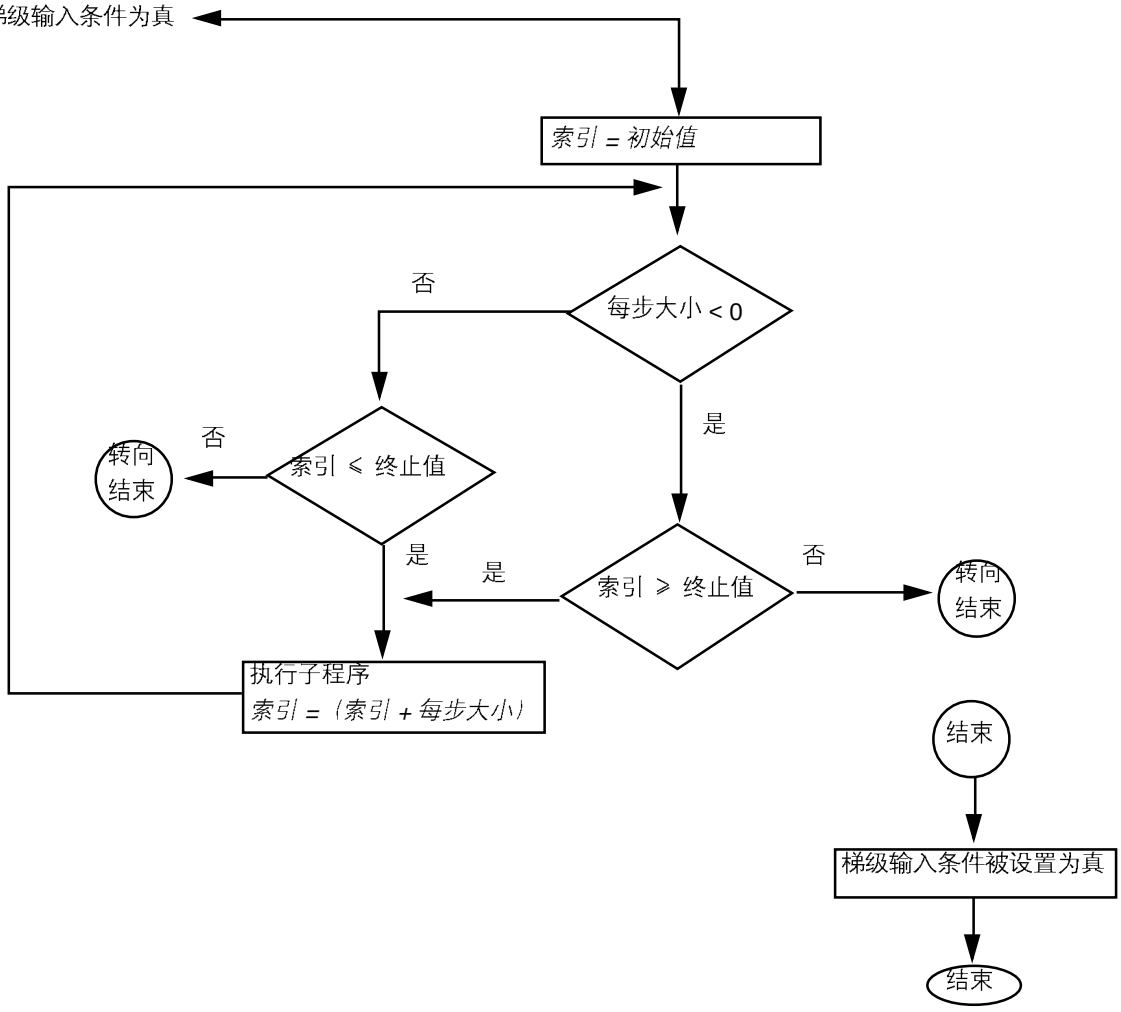
每次 FOR 指令执行子程序时，它都把每步大小加到索引值。

注意不要在单次扫描循环太多次。重复次数太多可能会引起控制器的看门狗超时，这会导致主要故障。

执行:

条件:	动作:
	预扫描梯级输出条件被设置为假。 控制器执行一次子程序 如果到同一子程序存在递归的FOR指令，则子程序只在第一次被预扫描。 如果到同一子程序存在多条 FOR 指令（非 - 递归），则子程序每次都被预扫描。
梯级输入条件为假	梯级输出条件被设置为假。

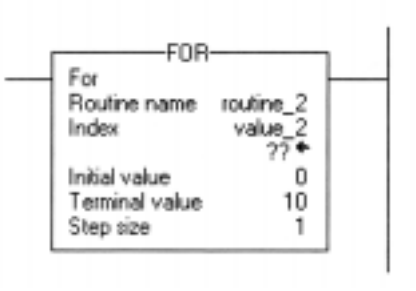
梯级输入条件为真



算术状态标志: 不影响。

故障条件: 无

FOR 指令举例:



当指令被使能时，FOR 指令重复执行 *routine_2*，而且每次 *value-2* 的值都加 1。

当 *value_2 > 10* 或一条 BRK 指令被使能时，FOR 指令不再执行 *routine_ 2*。

其他格式:

格式:	句法:
Neutral 文本	`FOR (routine-name,index,initial-value,terminal-value,step-size)`
ASCII 文本	`FOR routine-name,index,initial-valie,terminal-valie,step-size`

相关指令: BRK，JMP，LBL，JSR，SBR，RET

终止循环指令（BRK）

BRK 指令是一条输出指令

操作数: 无

说明: BRK 指令中断被 FOR 指令调用的子程序的执行。

当指令被使能时，BRK指令离开当前子程序并使控制器返回到 FOR 指令的下一条指令。

如果存在嵌套的 FOR 指令，则 BRK 指令使控制返回到 FOR 指令的最内层。

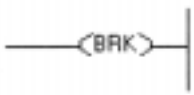
执行:

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。 执行返回到用于调用的 FOR 指令的下一条指令。

算术状态标志: 不影响

故障条件: 无

BRK 指令举例:



当指令被使能时，BRK 指令停止当前子程序的执行，并且返回到用于调用的FOR指令的下一条指令，

其他格式:

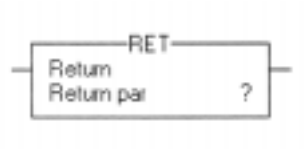
格式:	句法:
Neutral 文本	BRK ( ) ;
ASCII 文本	BRK

相关指令: FOR，JMP，LBL，JSR，SBR，RET

返回指令（RET）

RET 指令是一条输出指令

操作数: 无



说明:

RET 指令返回到调用的 FOR 指令。FOR 指令不使用操作数。FOR 指令忽略用户输入到 RET 指令的任何参数。

当指令被使能时，RET 指令返回到 FOR 指令。FOR 指令以每步大小为单位增加索引值，并再次执行子程序。如果索引值超过终止值，则 FOR 指令完成，而且程序执行转移到 FOR 指令的下一条指令。

用户也可以用 TND 指令结束子程序的执行。

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。

算术状态标志: 不影响

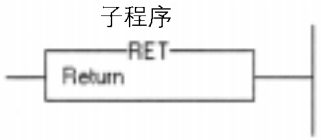
故障条件: 无

RET 指令举例:

调用子程序的程序

当指令被使能时，FOR 指令重复执行 *routine_2* 而且每次 *value_2* 的值加 1。如果 *value_2 > 10* 或 BRK 指令被使能，则 FOR 指令不再执行 *routine_2*。

子程序



当指令被使能时，RET 指令返回到用于调用的 FOR 指令。FOR指令再次执行子程序并且以每步大小为单位增加索引值，如果索引值超过终止值，则 FOR 指令完成，而且程序执行移动到 FOR 指令的下一条指令。

其他格式:

格式:	句法:
Neutral 文本	RET( );
ASCII 文本	RET

相关指令: BRK， JMP ，LBL，TND

专用指令

(FBC，DDT，DTR，PID)

简介

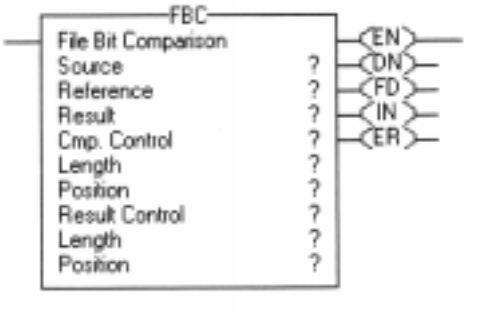
专用指令执行特殊应用操作。

如果用户要：	使用下列指令：	参见页次：
与一已知的参考数据比较，并记录所有不匹配位的位置.	FBC	12 - 2
与一已知的参考数据比较，记录所有不匹配位的位置.并且改参考数据使之与源操作数匹配.	DDT	12 - 9
通过屏蔽传递源操作数，并比较传递结果与参考数据.然后用源操作数覆盖参考数据以用于下一次比较.	DTR	12 - 16
控制一个 PID 回路。	PID	12 - 19

文件位比较 (FBC)

FBC 指令是一条输出指令。

操作数:



操作数:	数据类型:	格式:	说明:
源	DINT	数组标签	与参考数组比较的数组 不要在此下标处 CONTROL.POS
参考	DINT	数组标签	与源数组比较的参考数组 不要在此下标处 CONTROL.POS
结果	DINT	数组标签	存储比较结果的数组 不要在此下标处 CONTROL.POS
比较控制	CONTROL	结构体	比较运算的控制结构体
长度	DINT	立即数	要比较的位的数量
位置	DINT	立即数	源数组的当前位置 初始值一般为 0
结果控制	CONTROL	结构体	运算结果的控制结构体
长度	DINT	立即数	存储在结果数组内位的数量
位置	DINT	立即数	结果数组的当前位置 初始值一般为 0

注意: 比较控制结构体与结果控制结构体要用不同的标签。如果它们使用相同的控制结构体，会发生不可预料的操作，可能会导致设备的损坏和 / 或人身伤亡。

比较 CONTROL 结构体:

助记符:	数据类型:	说明:
.EN	BOOL	使能位 - 标识 FBC 指令被使能。
.DN	BOOL	如果FBC 指令完成对原数组与参考数组最后一位的比较，完成位被置位。
.FD	BOOL	每次 FBC 指令记录一个不匹配(每次完成一次操作模式)，或记录所有不匹配后(每次扫描完成整体操作模式)，发现位被置位。
.IN	BOOL	约束位 - 标识 FBC 指令搜索的模式。 0 = 整体模式 1 = 每次记录一个不匹配模式
.ER	BOOL	如果比较位置值(.POS)< 0；比较长度值(.LEN)< 0；结果位置值(.POS)< 0；或结果长度值(.LEN)< 0，则错误位被置位。在程序清零错误位(.ER)之前，指令停止执行。
.LEN	DINT	长度值 - 标识比较的位的数量。
.POS	DINT	位置值 - 标识当前位的位置。

结果 CONTROL 结构体:

助记符:	数据类型:	说明:
.DN	BOOL	当结果数组满时，置位完成位。
.LEN	DINT	长度值 - 标识结果数组内存储位置的数量。
.POS	DINT	位置值 - 标识结果数组的当前位置。

说明:

FBC 指令使源数组内的位与参考数组内的位进行比较。

当指令被使能时，FBC 指令比较源数组内的位与参考数组内的位，并且记录每个不匹配的位号于结果数组内。

FBC 指令对连续存储器单元进行操作。*详细信息参见附录 B - 3 页的将数组看作一存储块*。

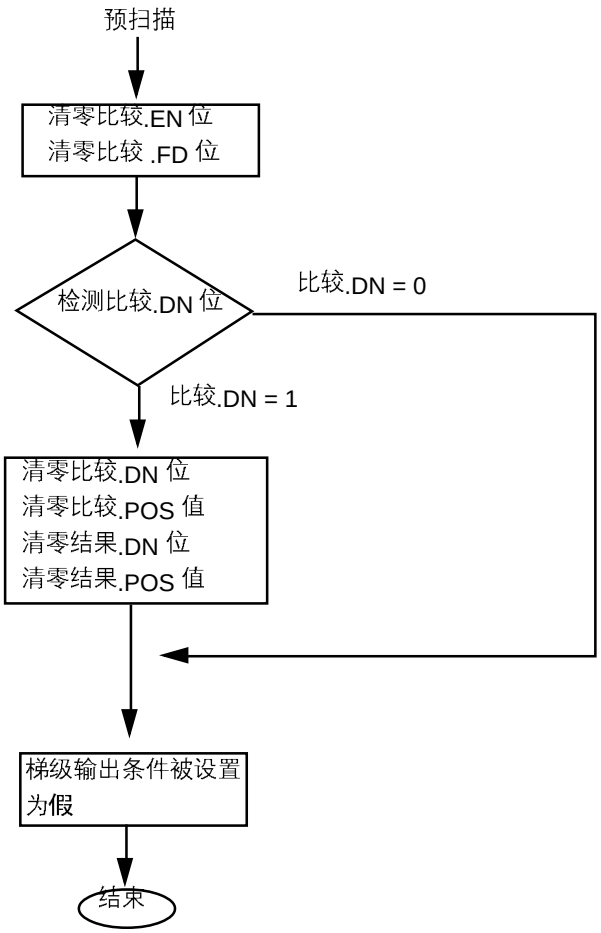
DDT 与 FBC 指令的区别是：每次 DDT 指令发现一个不匹配位，该指令改变参考位的值使之与原数组内的位匹配。而 FBC 指令不改变参考位的值。

选择搜索模式

如果要检测:	选择如下模式:
每次扫描记录一个不匹配	置位比较 CONTROL 结构体的 .IN 位。 每次梯级输入条件由假变真时，FSC 指令搜索源操作数与参考数组之间的下一个不匹配位。发现一个不匹配的位后，指令置位.FD 位，记录不匹配的位所在的位置，并停止执行。
每次扫描记录所有不匹配	清零比较 CONTROL 结构体的 .IN 位。 每次梯级输入条件由假变真时，FSC 指令搜索所有源操作数与参考数组之间的不匹配的位。

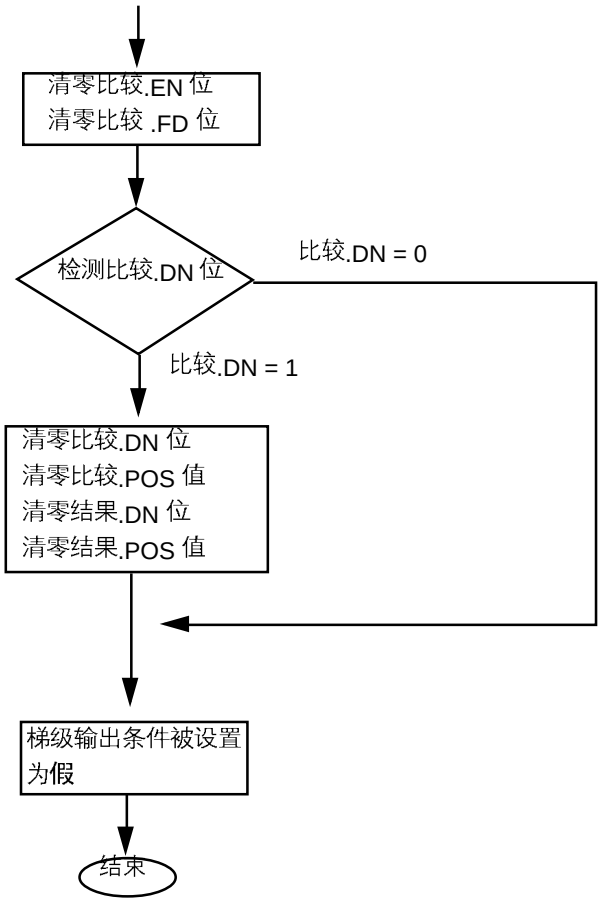
执行:

条件:	动作:

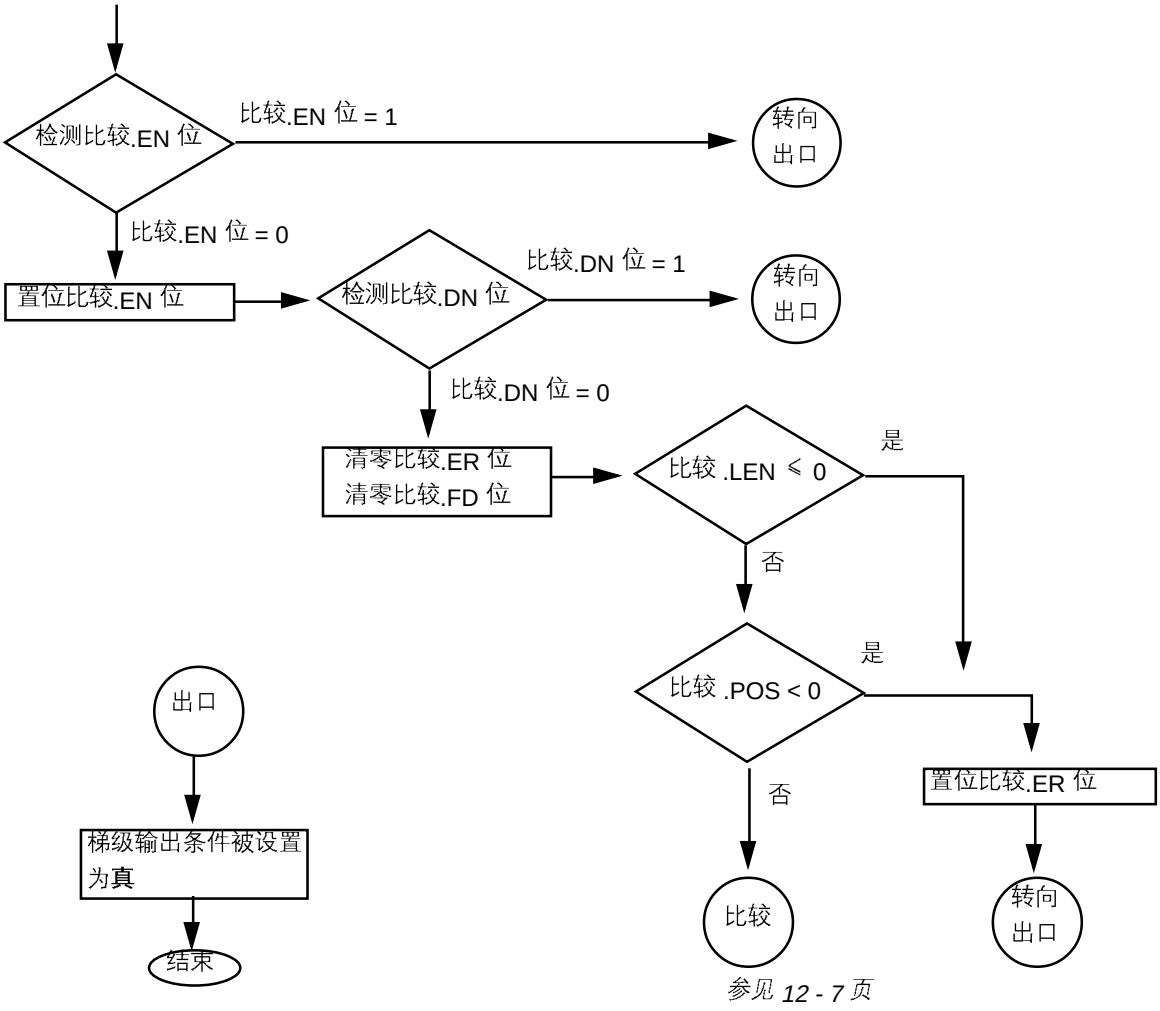


条件: **动作:**

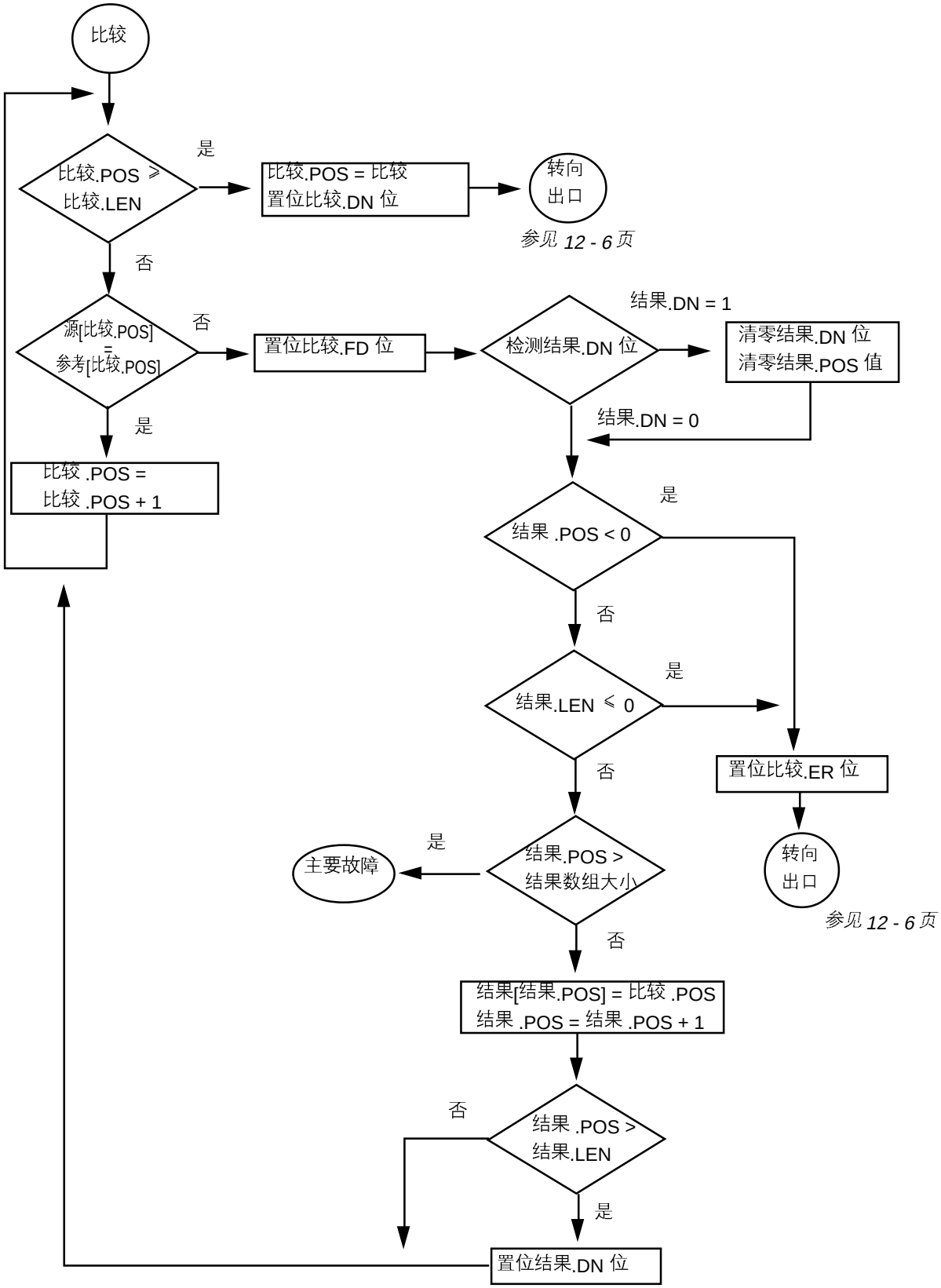
�梯级输入条件为假



条件:	动作:
梯级输入条件为真	



条件: 动作:

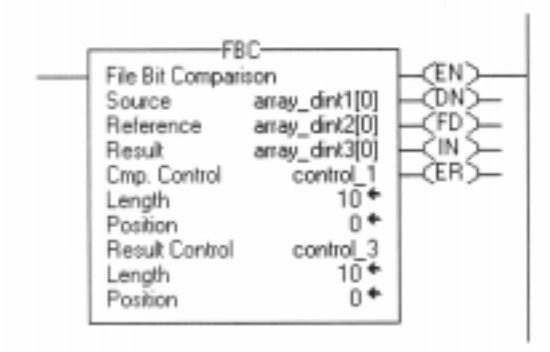

比较
比较.POS ≥ 比较.LEN
是
比较.POS = 比较
置位比较.DN 位
转向
出口
参见 12 - 6 页
否
源[比较.POS] = 参考[比较.POS]
否
置位比较.FD 位
检测结果.DN 位
结果.DN = 1
清零结果.DN 位
清零结果.POS 值
结果.DN = 0
是
比较 .POS = 比较 .POS + 1
结果 .POS < 0
是
否
结果.LEN ≤ 0
是
否
置位比较.ER 位
结果.POS > 结果数组大小
是
主要故障
转向
出口
参见 12 - 6 页
否
结果[结果.POS] = 比较 .POS
结果 .POS = 结果 .POS + 1
否
结果 .POS > 结果.LEN
是
置位结果.DN 位

算术状态标志: 不影响

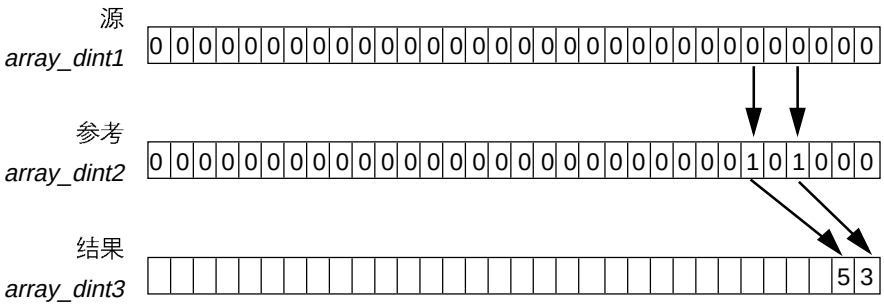
故障条件:

发生主要故障的条件:	故障类型:	故障代码:
结果 .POS > 结果数组的大小	4	20

FBC 指令举例:



当指令被使能时，FBC 指令比较源 array_dint1 与参考 *array_dint2* ，并存储所有不匹配的位所在位置于结果 *array_dint3* 内



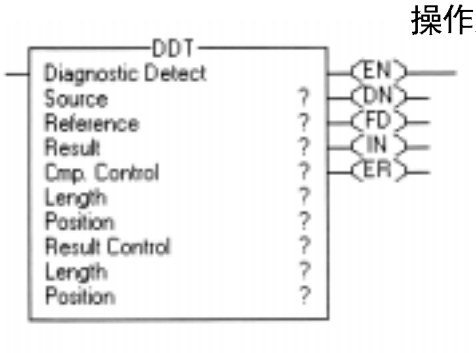
其它格式:

格式:	句法:
neutral 文本	FBC*(source,reference,result,cmp_control,length,position,result_control, length,position);*
ASCII 文本	FBC *source reference result cmp_control length position result_control length position*

相关指令: DDT，DTR

诊断检测(DDT)

DDT 指令是一条输出指令。



操作数:

操作数:	数据类型:	格式:	说明:
源	DINT	数组标签	与参考数组比较的数组 **不要**在此下标处用CONTROL.POS
参考	DINT	数组标签	与源数组比较的数组 **不要**在此下标处用CONTROL.POS
结果	DINT	数组标签	存储比较结果的数组 **不要**在此下标处用CONTROL.POS
比较控制	CONTROL	结构体	用于比较的控制结构体
长度	DINT	立即数	比较的位的数量
位置	DINT	立即数	源数组的当前位置 初始值一般为 0
结果控制	CONTROL	结构体	用于结果的控制结构体
长度	DINT	立即数	存储在结果数组内的位的数量
位置	DINT	立即数	结果数组的当前位置 初始值一般为 0

注意: 比较控制结构体与结果控制结构体要用不同的标签。如果它们使用相同的控制结构体，会发生不可预料的操作，可能会导致设备的损坏和 / 或人身伤亡。

比较 CONTROL 结构体:

助记符:	数据类型:	说明:
.EN	BOOL	使能位 - 标识 DDT 指令被使能。
.DN	BOOL	当 DDT 指令比较原数组与参考数组的最后一位时，完成位被置位。
.FD	BOOL	每次 DDT 指令记录一个不匹配(每次完成一次操作模式)，或记录所有不匹配后(每次扫描完成整体操作模式)，发现位被置位。
.IN	BOOL	约束位表示 DDT 指令搜索的模式。 0 = 整体模式 1 = 每次记录一个不匹配模式
.ER	BOOL	如果比较位置值(.POS)< 0；比较长度值(.LEN)< 0；结果位置值(.POS)<0；或结果长度值(.LEN)< 0，则错误位被置位。在程序清零错误位(.ER)之前，指令停止执行。
.LEN	DINT	长度值 - 表示比较的位的数量。
.POS	DINT	位置值 - 表示当前位置。

结果 CONTROL 结构体:

助记符:	数据类型:	说明:
.DN	BOOL	当结果数组满时，置位完成位。
.LEN	DINT	长度值表示结果数组内存储不匹配的位置的数量。
.POS	DINT	位置值表示结果数组的当前位置。

说明:

DDT 指令使源数组内的位与参考数组内的位进行比较，以确定状态的变化。

当指令被使能时，DDT 指令比较源数组内的位与参考数组内的位，并记录每个不匹配位的位号于结果数组内，而且改变参考数组内位的值，使其与相应的源数组内的位的数值匹配。

DDT 指令对连续存储器单元进行操作。详细信息参见附录 B - 3 页的将数组看作一存储块。

DDT 与 FBC 指令的区别是：每次 DDT 指令发现一个不匹配位，该指令都改变参考位数组内的位使之与原数组内的位匹配。而 FBC 指令不改变参考数组内位的值。

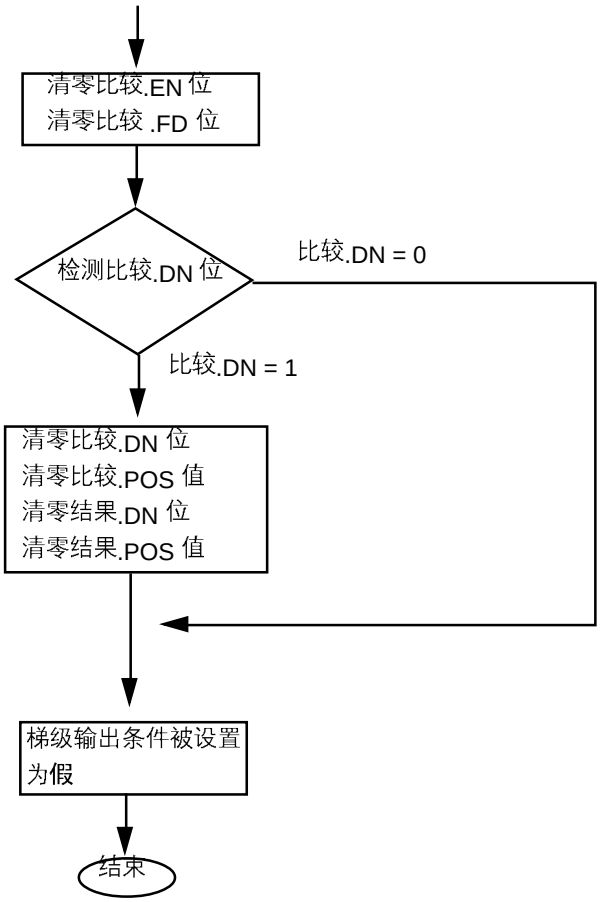
选择搜索模式

如果要检测:	选择如下模式:
每次扫描记录一个不匹配	置位比较 CONTROL 结构体的.IN 位。 每次梯级输入条件由假变真时，DDT 指令搜索源与参考数组之间的下一个不匹配位。发现一个不匹配的位后，指令置位.FD 位，记录不匹配的位所在的位置，并停止执行。
每次扫描记录所有不匹配	清零比较 CONTROL 结构体的.IN 位。 每次梯级输入条件由假变真时，DDT 指令搜索所有源数组与参考数组之间不匹配的位。

执行:

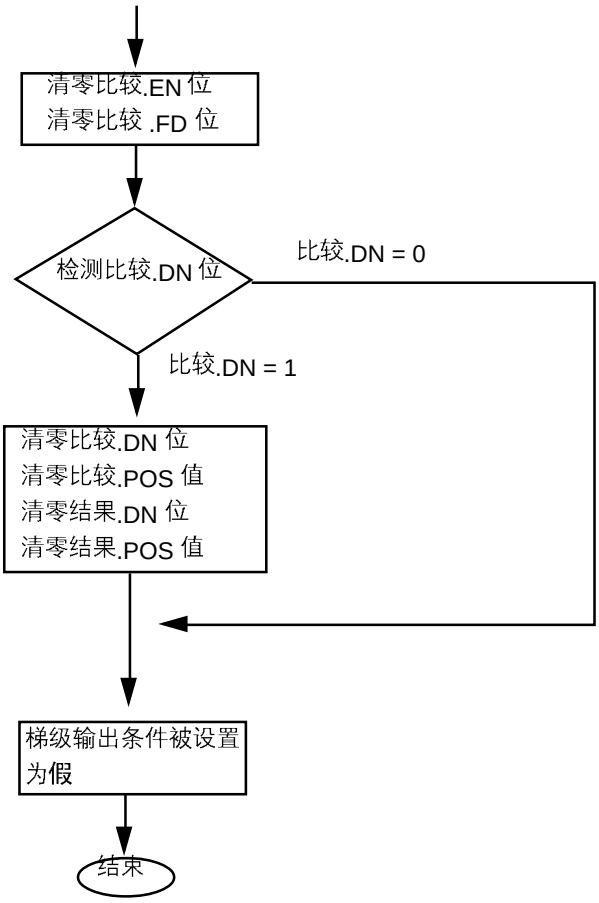
条件:	动作:

预扫描



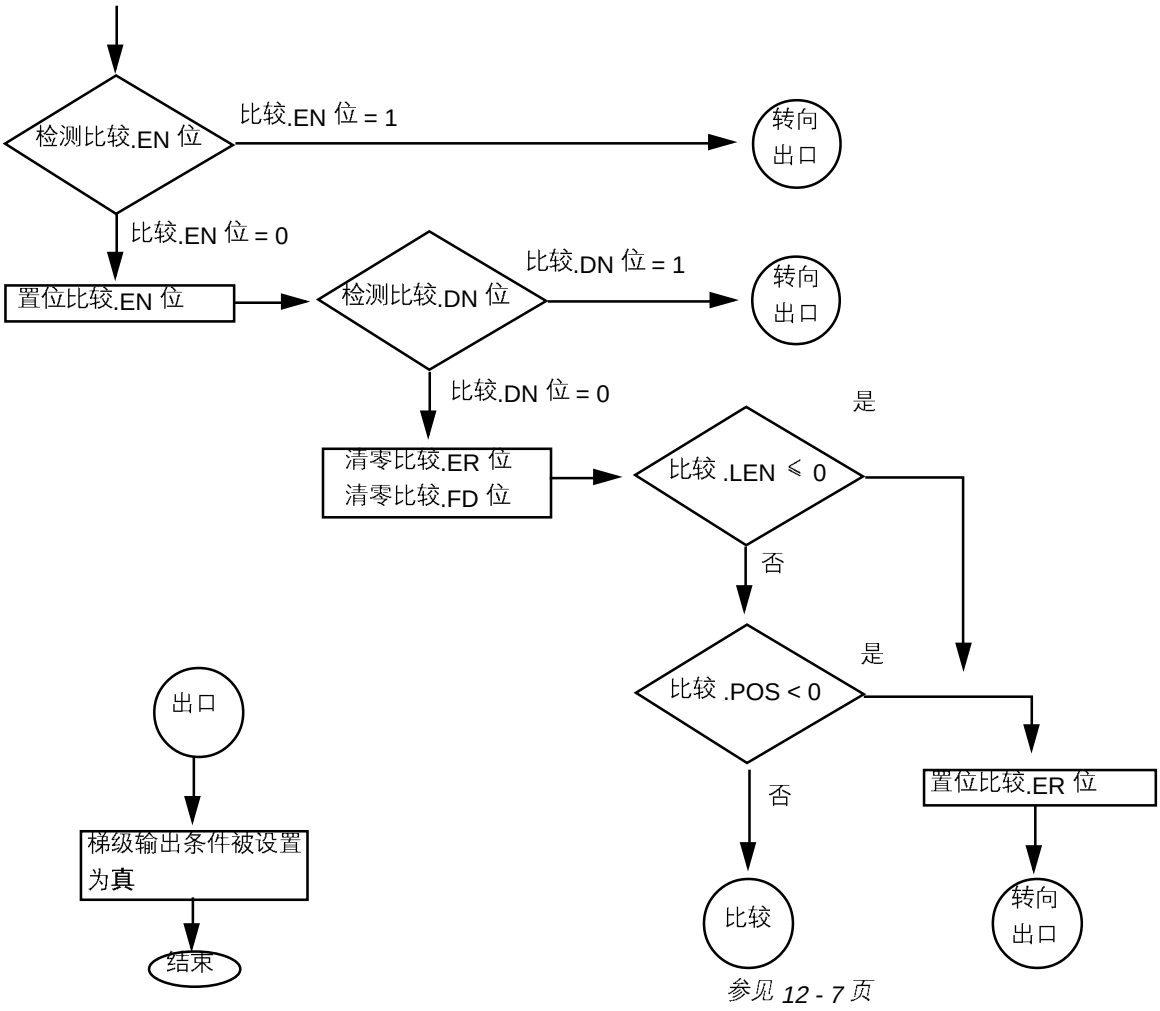
条件 **动作:**

�梯级输入条件为假

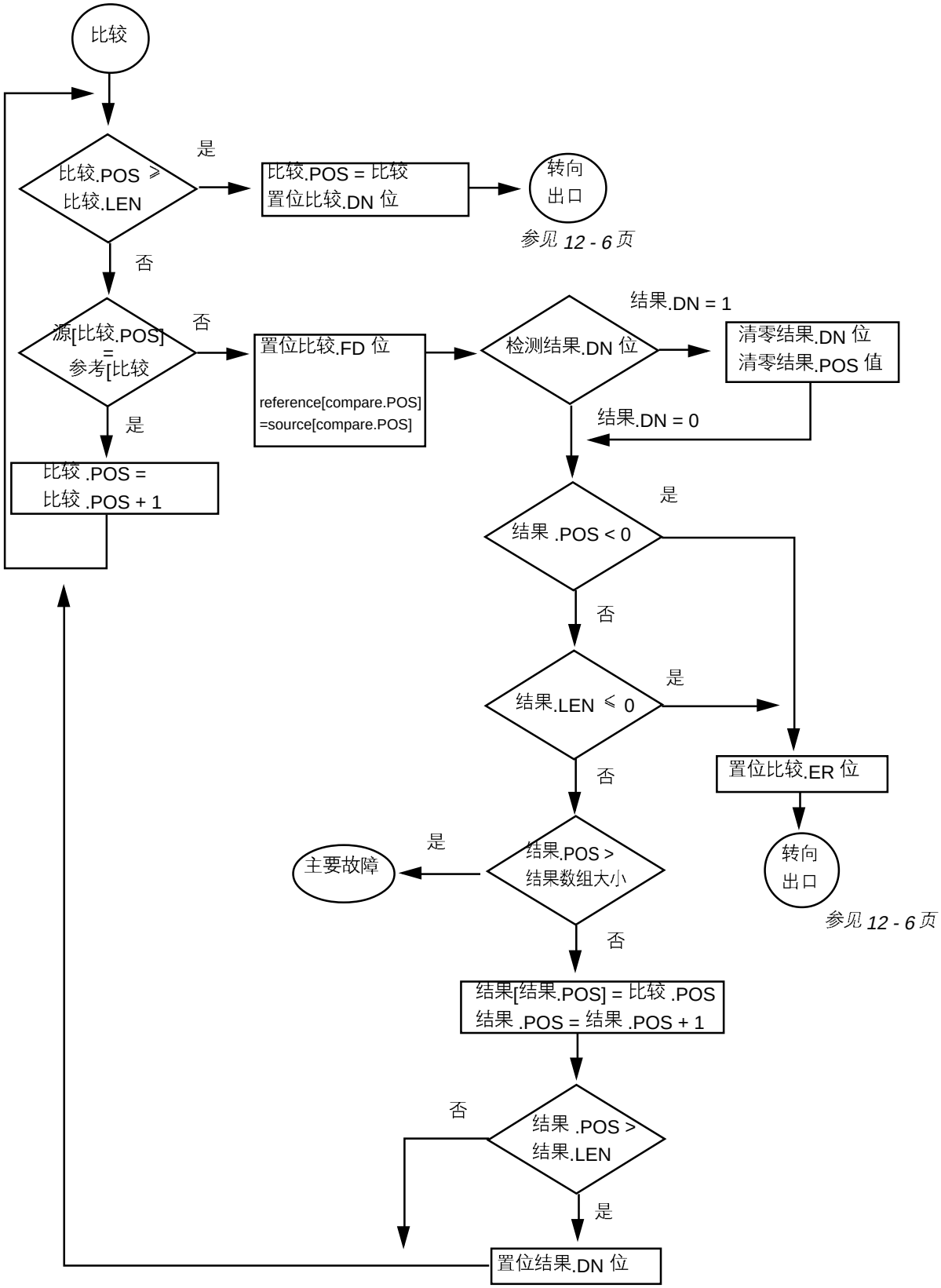


条件: **动作:**

梯级输入条件为真



条件: 动作:

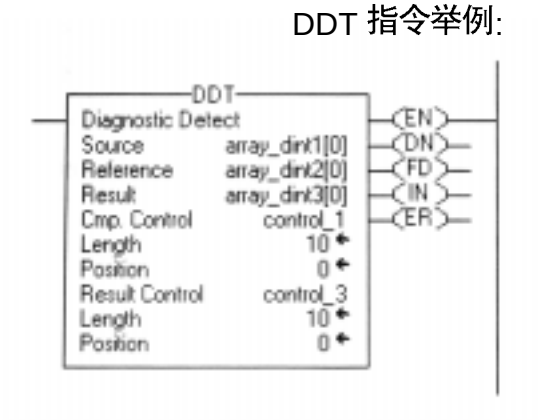

比较
比较.POS ≥ 比较.LEN
是
比较.POS = 比较
置位比较.DN 位
转向
出口
参见 12 - 6 页
否
源[比较.POS] = 参考[比较
否
置位比较.FD 位
reference[compare.POS] =source[compare.POS]
检测结果.DN 位
结果.DN = 1
清零结果.DN 位
清零结果.POS 值
结果.DN = 0
是
比较 .POS = 比较 .POS + 1
结果 .POS < 0
是
否
结果.LEN ≤ 0
是
否
置位比较.ER 位
转向
出口
参见 12 - 6 页
结果.POS > 结果数组大小
是
主要故障
否
结果[结果.POS] = 比较 .POS
结果 .POS = 结果 .POS + 1
否
结果 .POS > 结果.LEN
是
置位结果.DN 位

算术状态标志: 不影响

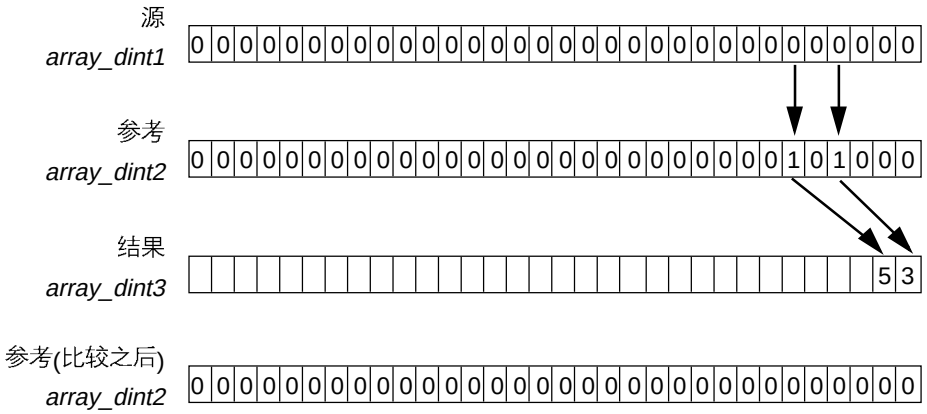
故障条件:

发生主要故障的条件:	故障类型:	故障代码:
结果 .POS > 结果数组的大小	4	20

DDT 指令举例:



当指令被使能时，DDT 指令比较源操作数 *array_dint1* 与参考操作数 *array_dint2* 并存储所有不匹配的位所在位置于结果操作数 *array_dint3* 内。控制器也改变参考操作数 *array_dint2* 内的不匹配位，使其与源操作数 *array_dint1* 匹配。



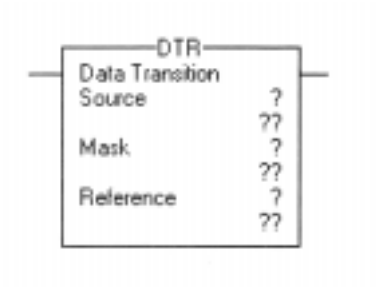
其它格式:

格式:	句法:
neutral 文本	`DDT( source,reference,result,cmp_control,length,position,result_control, length,position);`
ASCII 文本	`DDT source reference result cmp_control length position result_control length position`

相关指令: FBC，DTR

数据传送指令 (DTR)

DTR 指令是一条输出指令。



操作数:

操作数:	数据类型:	格式:	说明:
源	DINT	立即数 标签	与参考数组比较的数组。
屏蔽	DINT	立即数 标签	阻止或通过的位
参考	DINT	标签	与原数组比较的数组

说明:

DTR 指令通过屏蔽传递源值并且使其结果与参考值比较。DTR 指令同时也用被屏蔽的源值覆盖参考值，用于下一次比较。源值保持不变。

屏蔽操作数内的一个 ″1″ 意味着通过位数据。一个 ″0″ 意味着位数据被阻止。

当通过屏蔽后的源值与参考值不一致时，梯级输出条件为真一个扫描周期。如果屏蔽的源值与参考值相同，则梯级输出条件为假。

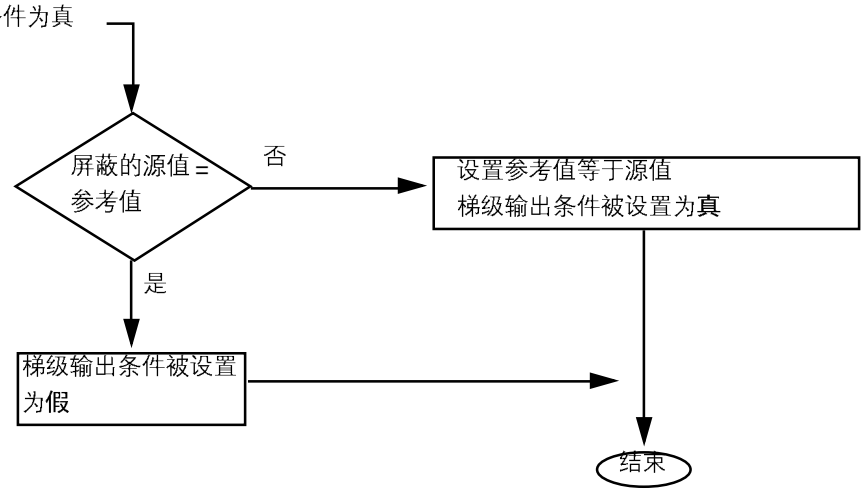
注意: 对该指令在线编程可能会有危险。如果参考值与源值不一致，则梯级输出条件变为真。当处理器正在运行或在远程运行方式时，插入该指令一定要特别小心。

输入立即数作为屏蔽值

当用户输入屏蔽值时，编程软件默认为是十进制的数值。如果用户要用其它格式输入屏蔽值，在该值前使用相应的前缀。

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

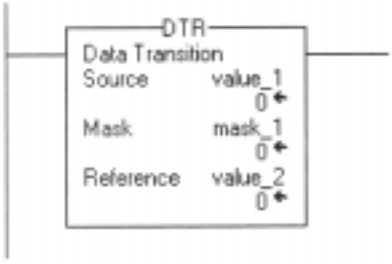
执行:

条件:	**动作**:
预扫描	参考值 = 源值 AND 屏蔽值。 梯级输出条件被设置为假。
梯级输入条件为假	参考值 = 源值 AND 屏蔽值。 梯级输出条件被设置为假。
梯级输入条件为真	

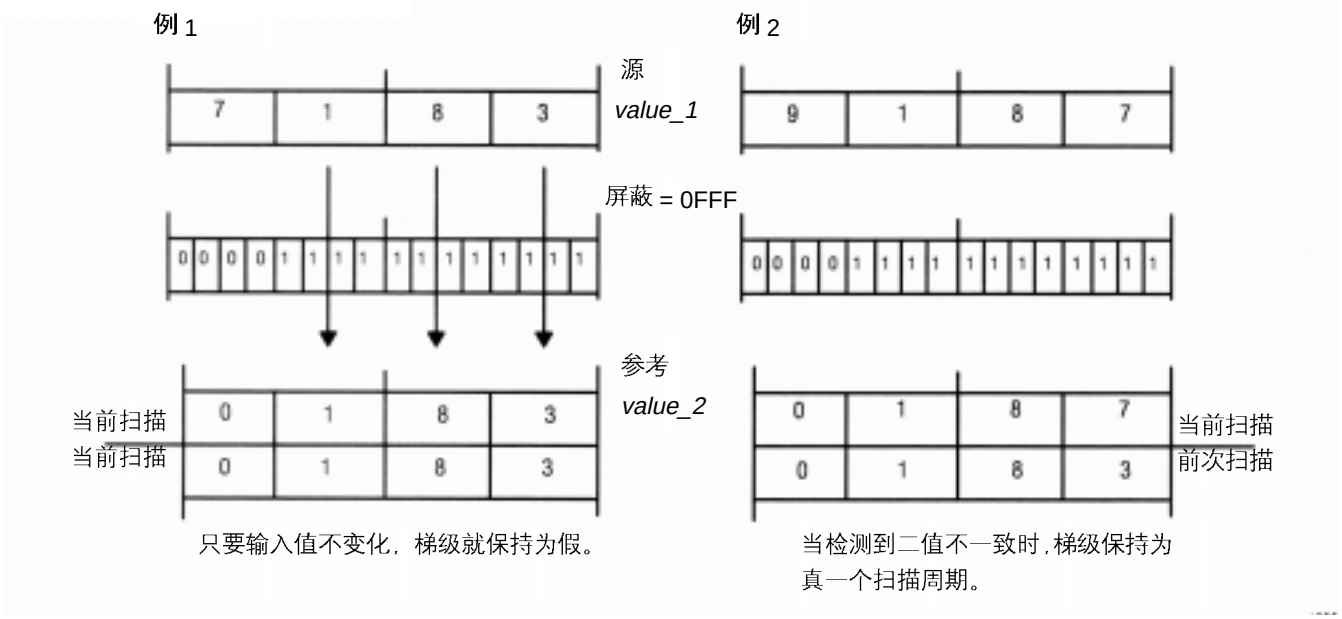
算术状态标志： 不影响

故障条件: 无

DTR 指令举例:



当指令被使能时，DTR 指令通过屏蔽比较 *value_1* 与 *value_2*，如果这二值不相同，则梯级输出条件被设置为真。



屏蔽操作数内的一个 0 值所对应的位不变。

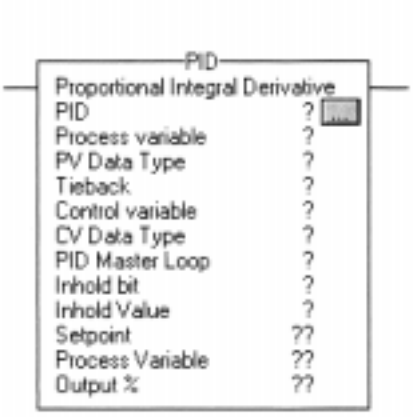
其它格式:

格式:	句法:
neutral 文本	`DTR(source,mask,reference);`
ASCII 文本	`DTR source mask reference`

相关指令: FBC，DDT

比例 积分 微分指令(PID)

PID 指令是一条输出指令。



操作数:

操作数:	数据类型:	格式:	说明:
PID	PID	结构体	PID 结构体
过程变量	SINT INT DINT REAL	标签	用户要控制的值
牵引信号	SINT INT DINT REAL	立即数 标签	*(可选的)* 硬件手动/自动工作站的输出，它旁路控制器的输出。 如果用户不想用此参数输入一个 0 值。
控制变量	SINT INT DINT REAL	标签	用于控制最终设备的数值(阀，气闸等) 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 主回路	PID	结构体	*可选择的* 用于主 PID 循环的 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。 如果用户不想用此参数输入一个 0 值。
初始化保持位	BOOL	标签	*可选择的* 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。 如果用户不想用此参数，输入一个 0 值。
初始化保持数据	SINT INT DINT REAL	标签	*可选择的* 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。 如果用户不想用此参数，输入一个 0 值。
设定点			*只用于显示* 显示设定点的当前值。
过程变量			*只用于显示* 显示整定的过程变量的当前值。
输出百分比			*只用于显示* 显示输出百分比的当前值。

PID 结构体: 为每条 PID 指令指定唯一的 PID 结构体。

助记符:	数据类型:	说明:
.CTL	DINT	.CTL 的各部分存储下列状态于一个 16- 位字内。 用户可以置位或清零下列状态位。

位:		数据类型:	说明:
.EN	31	BOOL	使能指令
.CT	30	BOOL	级联类型(0 = 从; 1 = 主)
.CL	29	BOOL	级联回路(0 = 否; 1 = 是)
.PVT	28	BOOL	跟踪过程变量(0 = 否; 1 = 是)
.DOE	27	BOOL	...的微分(0 = PV; 1 = 误差)
.SWM	26	BOOL	软件手动模式(0 = 否 - 自动) 1= 是 - 软件手动)
.CA	25	BOOL	控制作用 (0 = SP-PV; 1 = PV-SP)
.MO	24	BOOL	工作站模式 (0 = 自动; 1 = 手动)
.PE	23	BOOL	PID 方程(0 = 独立; 1 = 相关)
.NDF	22	BOOL	微分平滑处理(0 = 否; 1 = 是)
.NOBC	21	BOOL	反向偏置计算(0 = 否; 1 = 是)
.NOZC	20	BOOL	过零死区控制(0 = 否; 1 = 是)

PID 指令设置下列状态位。用户可以清零这些状态位。

位:		数据类型:	说明:
.INI	15	BOOL	PID 已初始化(0 = 否; 1 = 是)
.SPOR	14	BOOL	设置点超出范围(0 = 否; 1 = 是)
.OLL	13	BOOL	CV 低于输出限幅最小值(0 = 否; 1 = 是)
.OLH	12	BOOL	CV 高于输出限幅最大值(0 = 否; 1 = 是)
.EWD	11	BOOL	误差在死区范围内(0 = 否; 1 = 是)
.DVNA	10	BOOL	偏移下限报警(0 = 否; 1 = 是)
.DVPA	09	BOOL	偏移上限报警(0 = 否; 1 = 是)
.PVLA	08	BOOL	PV 下限报警 (0 = 否; 1 = 是)
.PVHA	07	BOOL	PV 上限报警 (0 = 否; 1 = 是)

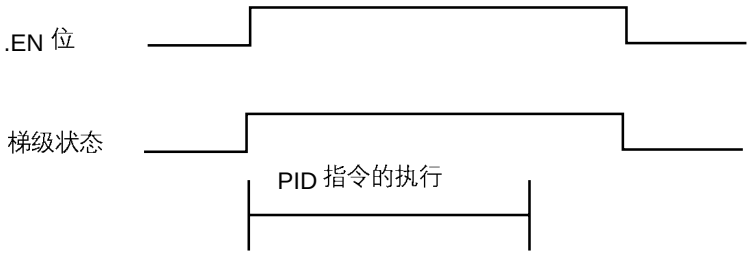
.SP	REAL	设定点
.KP	REAL	独立比例增益(无量纲) 相关控制器增益(无量纲)
.KI	REAL	独立积分增益(1 / 秒) 相关积分时间(分钟每循环)
.KD	REAL	独立微分增益(秒) 相关微分时间(分钟)
.BIAS	REAL	前馈或偏置百分比
.MAXS.	REAL	最大工程单位定标值
MINS	REAL	最小工程单位定标值
.DB	REAL	死区工程单位
.SO	REAL	设置输出百分比
.MAXO	REAL	最大输出限幅(输出百分比)
.MINO	REAL	最小输出限幅(输出百分比)
.UPD	REAL	回路更新时间(秒)
.PV	REAL	已定标的过程变量(PV)值
.ERR	REAL	已定标的误差值

助记符:	数据类型:	说明:
.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	正偏移报警极限
.DVN	REAL	负偏移报警极限
.PVDB	REAL	过程变量报警死区
.DVDB	REAL	偏移报警死区
.MAXI	REAL	最大过程变量(PV)值(未定标的输入)
.MINI	REAL	最小过程变量(PV)值(未定标的输入)
.TIE	REAL	手动控制的牵引信号
.MAXCV	REAL	最大控制变量(CV)值(对应于 100%)
.MINCV	REAL	最小控制变量(CV)值(对应于 0%)
.MINTIE	REAL	最小牵引值(对应于 100%)
.MAXTIE	REAL	最大牵引值(对应于 0%)
.DATA[17]	REAL	DATA[17] 中的各元素存储:

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]	百分比定标常数
.DATA[6]	.PV 的定标常数
.DATA[7]	微分定标常数
.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 的缩小定标常数
.DATA[16]	牵引值的缩小定标常数

说明: PID 指令控制诸如流速，压力，温度，或液位等过程变量。PID 指令通常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)，以保持过程变量在希望的设定点。

使能位(.EN)表示指令在执行状态。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)被复位。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。



执行:

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

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:
.UPD ≤ 0	4	35
设定点超出范围	4	36

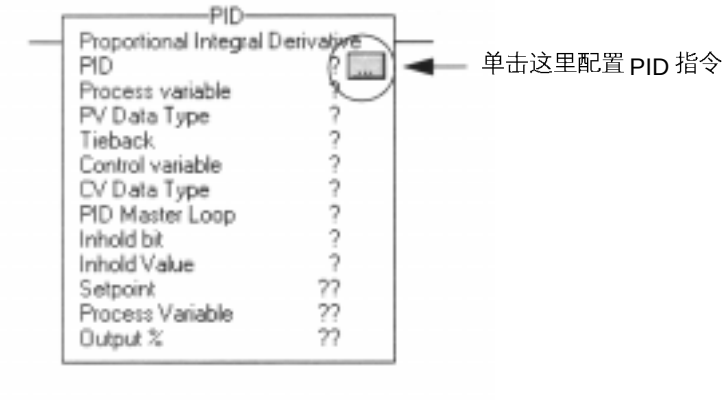
重要: 这些故障对于 PLC-5 处理器是主要故障

其他格式:

格式:	句法:
neutral 文本	`PID(pv,pv_type,tieback,cv,cv_type,master,inhold_bit, inhold_value);`
ASCII 文本	`PID pv pv_type tieback cv cv_type master inhold_bit inhold_value`

配置 PID 指令

当用户输入PID指令并指定PID结构体后，可以使用配置列表(tabs)来确定 PID 指令应如何发挥作用。



指定调节方式

选择调节列表(tab)。只要用户单击其它区域，然后单击OK，再单击应用(Apply)，或按回车(Enter)键，则改变就立刻有效。

在如下区域:	详述:
设定点(SP)	输入设定点值(.SP)。
设定输出百分比	设定输入输出百分比 (.SO)。 在软件手动模式，该值用于输出。 在自动模式，该值显示输出百分比。
输出偏置	输入输出偏置百分比(.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，它是控制增益(无量纲)。
积分增益(K i)	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，它是积分时间(分钟每循环)。
微分时间(K d)	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，它是微分时间(分钟)。
手动模式	在手动(.MO)或软件手动(.SWM)模式之间选择。 如果两者都被选择，则手动模式覆盖软件手动模式。

指定配置

选择配置列表(tab)。用户必须单击OK 或应用(Apply)才能使改动生效。

在如下区域:	详述:
PID 方程	选择独立增益或相关增益(.PE)。 当用户想要三项增益(P，I，和 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)(大于或等于 0.01 秒)
控制变量(CV)上限	输入控制变量上限值(.MAXO)。
控制变量(CV)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)。
无偏置计算	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，则选择是主回路或从回路(.CT)。

指定报警

选择报警列表(tab)。对于任何改动，用户必须单击 OK 或应用(Apply)才能使改变生效。

在如下区域:	详述:
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 死区报警值(.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)。

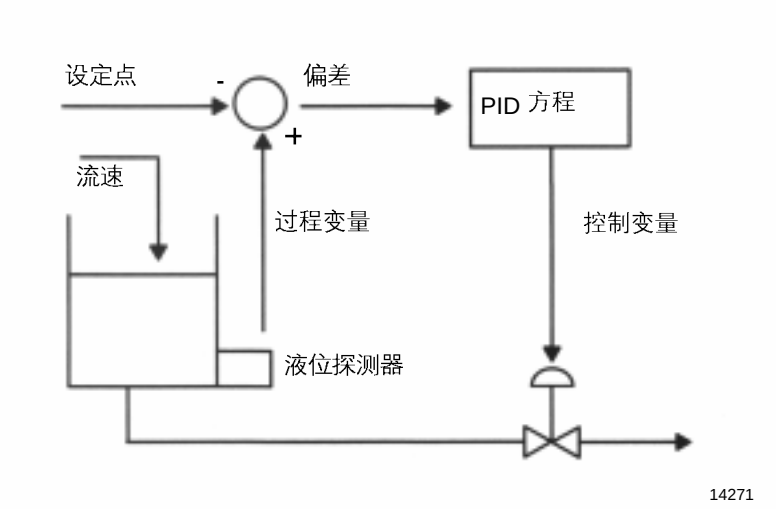
指定定标

选择定标列表(tab)。对于任何改变，用户必须单击 OK 或应用(Apply)才能使改变生效。

在如下区域:	指定:
未定标的 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定标的 PV 值。
未定标的 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定标的 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。
最大牵引值	输入最大牵引值(.MAXTIE)，它等于由模拟量输入通道获得的牵引值的最大未定标值。
最小牵引值	输入最小牵引值(.MINTIE)，它等于由模拟输入通道获得的牵引的最小未定标值。
PID 初始化	如果用户在运行模式中改变定标常数，则关闭该标志以重新初始化内部缩小比例值。

使用 PID 指令

PID 闭环控制使过程变量保持在希望的设定点。下图展示了一个流速 / 液位控制实例:



在上例中，比较槽内的液位与设定点。如果液位高于设定点，则 PID 方程增加控制变量并使槽的出口阀打开; 从而降低槽内的液位。

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特定的比例，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。

增益选项:	微分:	方程:
相关增益 (ISA 标准)	偏差(E)	$CV = K_C\left[E + \frac{1}{T_i}\int_o^t Edt + T_d\frac{dE}{dt}\right] + BIAS$
	过程变量(PV)	E = SP - PV $CV = K_C\left[E + \frac{1}{T_i}\int_o^t Edt - T_d\frac{dPV}{dt}\right] + BIAS$ E = PV - SP $CV = K_C\left[E + \frac{1}{T_i}\int_o^t Edt + T_d\frac{dPV}{dt}\right] + BIAS$
独立增益	偏差(E)	$CV = K_pE + K_i\int_o^t Edt + K_d\frac{dE}{dt} + BIAS$
	过程变量(PV)	E = SP - PV $CV = K_pE + K_i\int_o^t Edt + K_d\frac{dPV}{dt} + BIAS$ E = PV - SP $CV = K_pE + K_i\int_o^t Edt + K_d\frac{dPV}{dt} + BIAS$

式中:

变量:	说明:
K_P	比例增益 (无量纲) $K_p = K_c$ 无量纲
K_i	积分增益 (秒 $^{-1}$) 在 K_i (积分增益) 和 T_i (积分时间)之间进行转换，使用 $K_i = \frac{K_c}{60T_i}$
K_d	微分增益 (秒) 在K_d (微分增益)与T_d (微分时间)之间进行转换，使用公式: $K_d = K_c (T_d) 60$
K_C	控制器增益 (无量纲)
T_i	积分时间 (分钟 / 循环)
T_d	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

如果用户不想使用PID方程的某个特定项，仅须设置其增益为零即可。例如，如果用户不想采用微分作用，则设定Kd或Td等于零。

防止积分饱和及由手动向自动的无冲击转换

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饱和。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从新开始。

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。 让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接口设备的操作员输入。
手动(.MO)	把牵引(tieback)值作为一个输入，通过调节其中间变量以产生相同值的输出。 PID 指令的牵引信号输入根据.MINTIE 和.MAXTIE 值被定标到 0 至 100% 的范围，并被用作回路输出。牵引(tieback)输入一般来自于一个硬件的手控 / 自控工作站，使输出绕过控制器。 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指令反向计算使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(tieback)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(tieback)值开始，故输出值不会出现冲击。

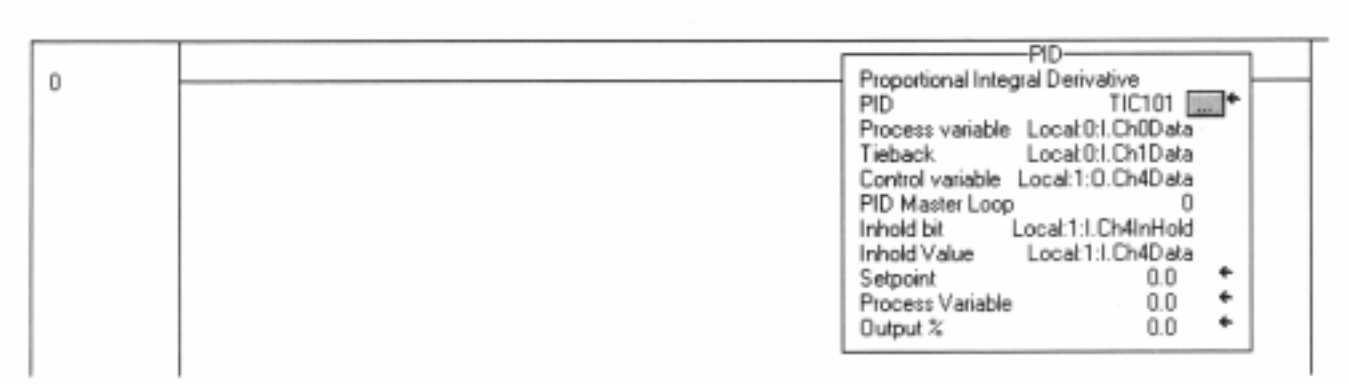
即使未使用积分控制(即 Ki = 0)，PID 指令也能自动地提供从手动到自动的无冲击转换。在这种情况下，指令修改.BIAS 项以使 CV 输出跟踪设定输出或牵引(tieback)值。当返回到自动控制时，.BIAS 项会保持其最后的值。用户可以通过在 PID 数据结构体中置位.NOBC位来禁止反向计算.BIAS项。注意如果用户设置.NOBC位为真，则PID指令在未使用积分控制时不再提供从手动到自动的无冲击转换。

PID 指令回路更新时间

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。对于各种缓慢回路，例如温度回路，采用每秒一次或更长的更新时间通常足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250毫秒一次的更新时间。只有很少的情况，(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。

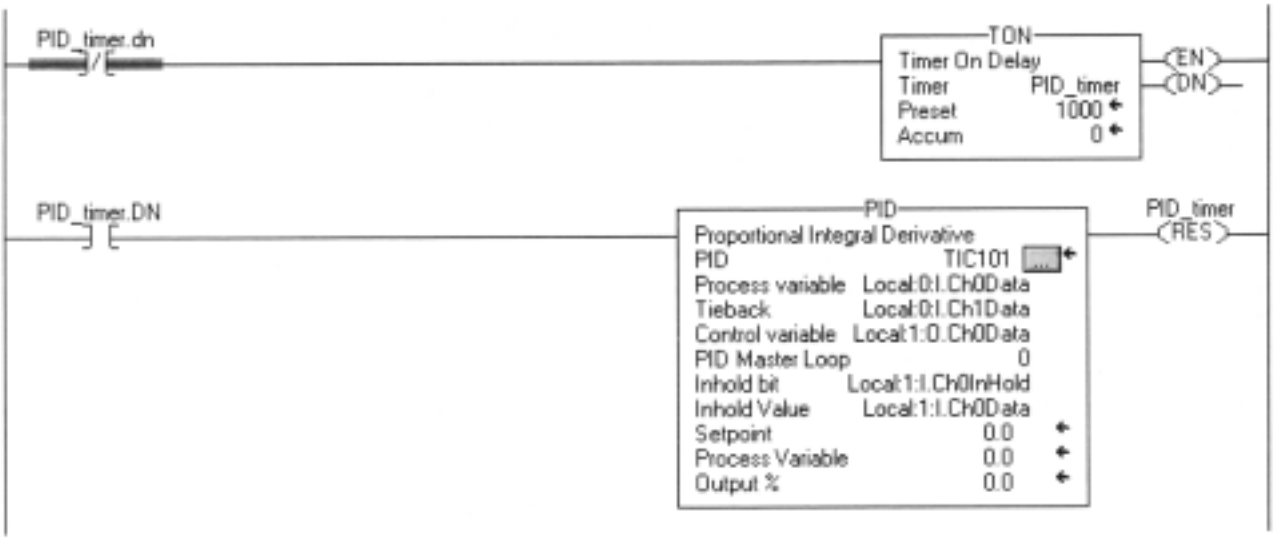
由于PID指令在其计算中使用时间基，所以用户需要把指令的执行同过程变量(PV)的采样相同步。

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得到执行。例如，使用一个无条件的梯段。



当使用周期性任务时，应确保以处理器更新过程变量所使用的模拟输入要比周期性任务的速率快得多。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。这能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，使用 250 毫秒的回路更新时间(.UPD =.25)，并配置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。但是精度稍差一些。

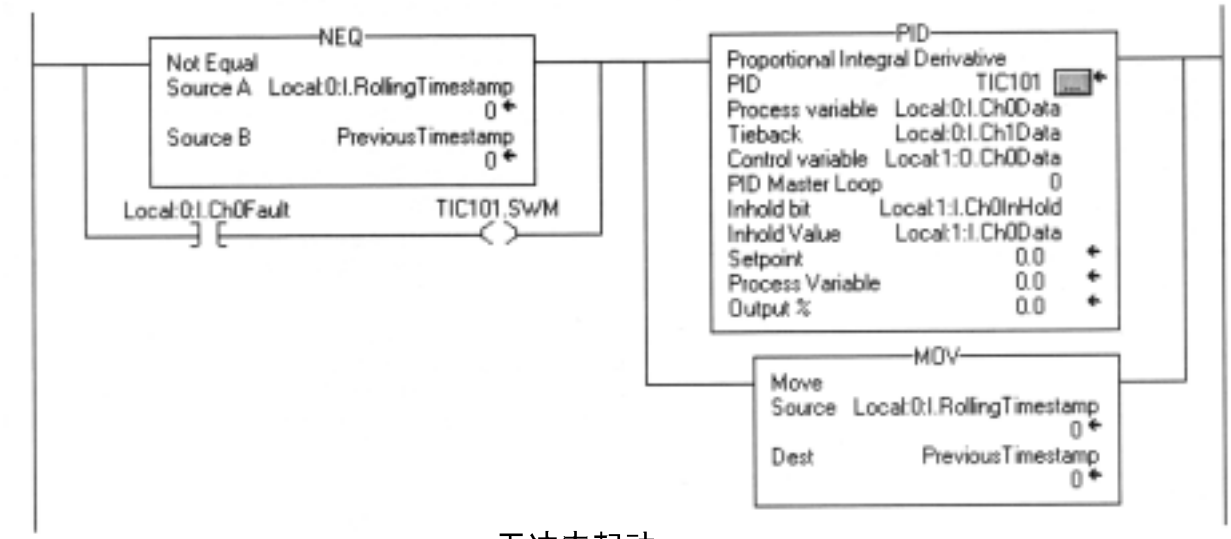


在这种方法中，需设定PID指令的回路更新时间等于定时器预置值。这与使用周期性任务时的情况相同，用户需要设置模拟量输入模块，使其生成过程变量的速率比回路更新时间快得多。用户只能在回路更新时间至少是最差情况下连续任务执行时间的若干倍长的回路中，使用定时器方式执行 PID。

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标记(由模拟量输入数据结构体的滚动时间标记{.Rolling Times stamp}的数值所表示)。时间标记范围为 0-32767 毫秒。监视时间标记，当它改变时，就已经收到一个新的过程变量采样。每次当时间标记发生改变时，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的，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指令放于连续任务中，只要确保程序逻辑能以比每 250 毫秒一次快的频率被更新即可。

下面是一个采用RTS方法执行的例子。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有故障或被拿掉，则控制器不再收到滚动时间标记，PID 回路就停止执行。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控制。这样使操作员仍能用手动方式改变 PID 回路的输出。



无冲击起动

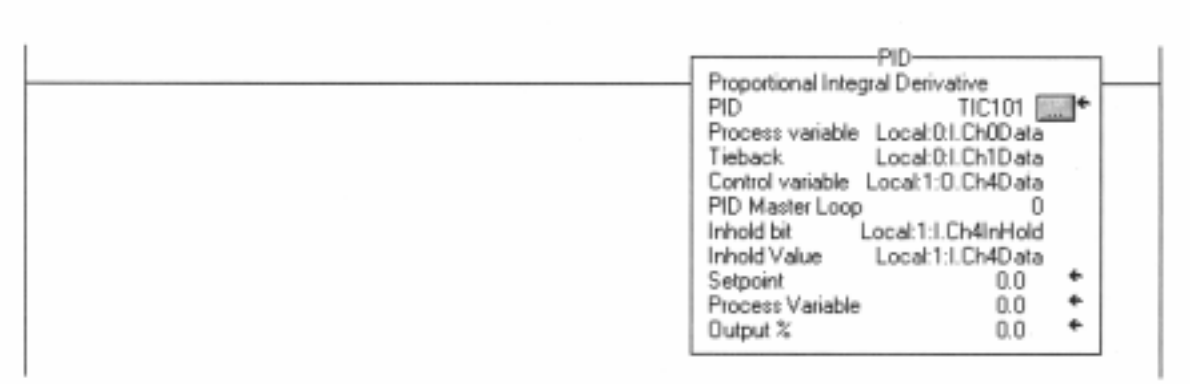
当控制器从编程模式转换为运行模式时，或控制器上电时，PID指令能与1756模拟量输出模块相配合，以支持无冲击再起动。

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。

设置无冲击起动

做如下工作:	详细说明:
配置从 PID 指令接收控制变量的 1756 模拟量输出模块的通道	在模块指定通道的参数页选取“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起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，直到从控制器送来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 项以一定斜率过渡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化保持数值标签	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，指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这个数值时，关断初始化保持状态位，从而允许 PID 指令开始正常的控制。

如下 PID 指令用到了初始化保持位和初始化保持数值:



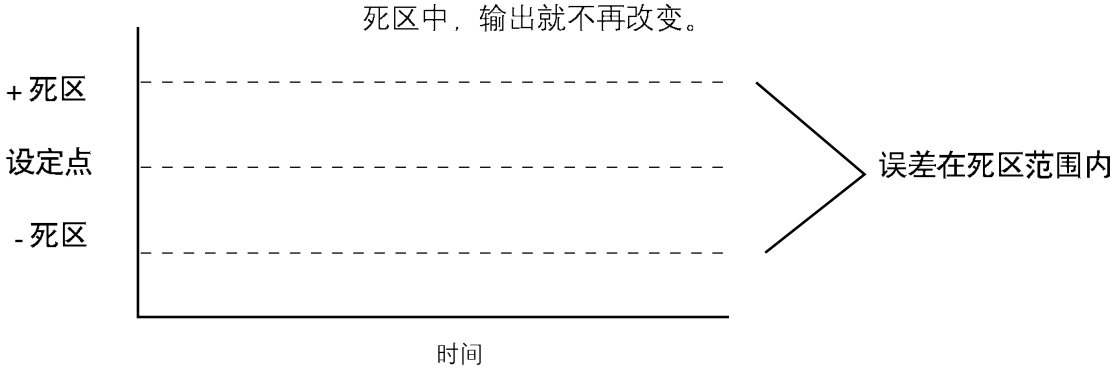
微分平滑

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于减小由 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。如果用户的过程需要非常大的微分增益数值时(例如 $K_d > 10$)，可以禁止微分平滑作用。要禁止微分平滑作用，可以在配置制表栏(tab)中选择 ″禁止微分平滑(No derivative smoothing)″ 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。这个死区使用户能够控制不改变输出的条件下，过程变量与设定点匹配的接近程度。死区还有助于减少用户的最终控制设备的磨损。

当过程变量进入死区后，死区控制使用过零检测使指令能把误差用于计算目的，直到过程变量穿过设定点。一旦过程变量经过设定点(误差过零并改变符号)，只要过程变量保持在死区中，输出就不再改变。



死区控制采用过零判别方式，当过程变量进入死区后，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)，只要它保持在死区范围内，指令在计算时就认为误差为零。

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定标单位。用户可以通过在配置制表栏(tab)中选择 ″禁止过零死区(no zero crossing for deadband)″ 选项，或在 PID 结构体中置位.NOZC 位，来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(输出的百分比)。当指令检测到输出达到限幅值时，它就置位一个报警位，并且防止输出超出上下限幅值。

前馈或输出偏置

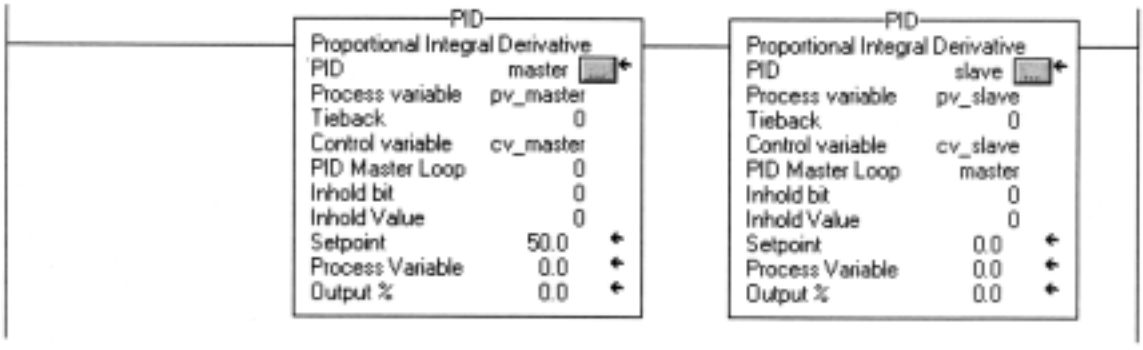
用户可以通过馈送.BIAS 值到 PID 指令的前馈 / 偏置数值中，以前馈补偿来自系统的扰动。

前馈值可以在馈入到PID指令的扰动有机会改变过程变量之前，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，一般使用偏置值。在这种情况下，可通过调节偏置值，以保持输出在需要的范围内，从而保持 PV 接近设定点。

级串回路

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。



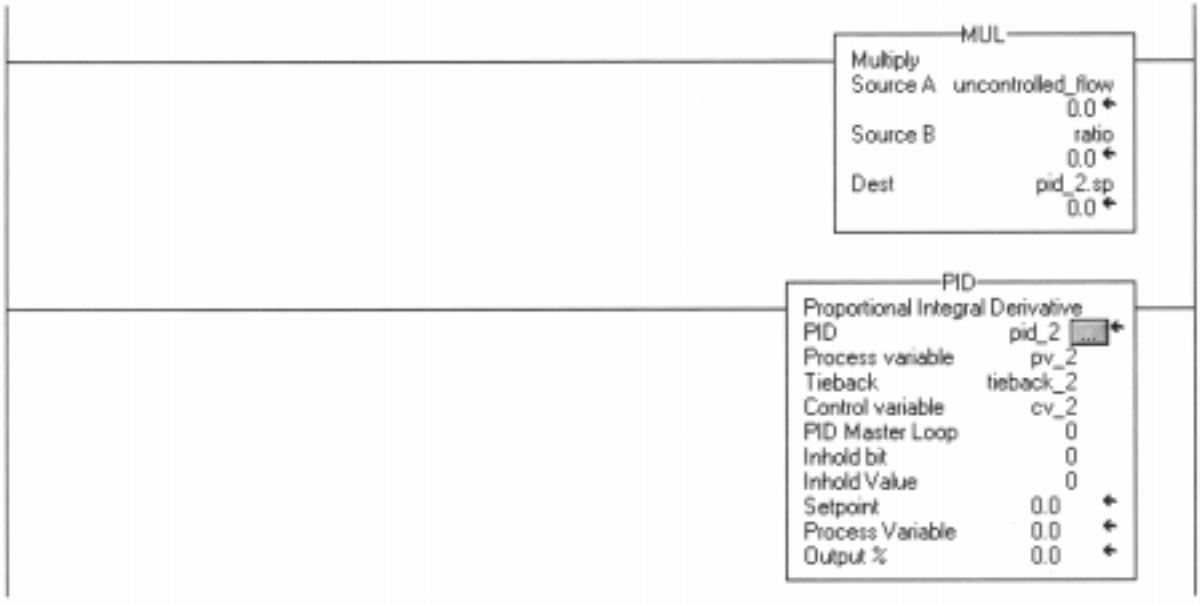
比率控制

用户可以通过使用具有下列参数的乘法 MUL 指令来保持两个量具有一定的比率:

- 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

在乘法 MUL 指令中输入:

对于如下乘法 MUL 参数:	输入下列数值:
目的数	控制量
源 A	非受控量
源 B	比率



注释:

三角函数指令

(SIN，COS，TAN，ASN，ACS，ATN)

简介

三角函数指令用三角函数来进行算术运算。

如果用户要:	使用下列指令:	参见页次:
计算一个数值的正弦值	SIN	13 - 2
计算一个数值的余弦值	COS	13 - 4
计算一个数值的正切值	TAN	13 - 6
计算一个数值的反正弦值	ASN	13 - 8
计算一个数值的反余弦值	ACS	13 - 10
计算一个数值的反正切值	ATN	13 - 12

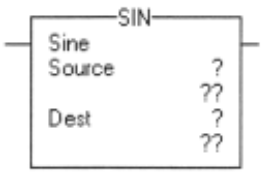
在运算时用户可以使用混合数据类型，但是这样会损失精度或发生取整误差，而且会增加指令的执行时间。检测 S:V 位以确认结果是否被截短。

黑体字数据类型是最优的数据类型。如果指令的所有操作数都使用同一最优数据类型，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。

只要梯级输入条件为真，则指令每次被扫描都执行一次。如果用户只希望指令计算一次，则可以用一条 ONS 指令来触发三角函数指令。

正弦指令 (SIN)

SIN 指令是一条输出指令。



操作数

操作数:	数据类型:	格式:	说明:
源	SINT INT DINT **REAL**	立即数 标签	计算该值的正弦值。
目的	SINT INT DINT **REAL**	标签	存储运算结果的标签。

说明: SIN 指令计算源操作数的正弦值 (以弧度表示)，并存储结果于目的单元内。

源值必须大于或等于 -205887.4 并且小于或等于 205887.4。目的单元内的结果数值总是大于或等于 -1 并且小于或等于 1。

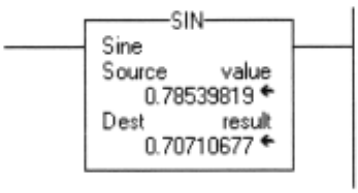
执行

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	控制器计算源数值的正弦值，并存放结果于目的单元内。 梯级输出条件被设置为真。

算术状态标志: 影响算术状态标志

故障条件: 无

SIN 指令举例:



当指令被使能时，SIN 指令计算 *value* 的正弦值并存放结果于

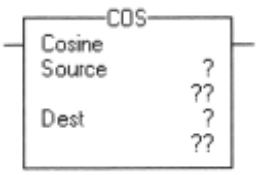
其他格式:

格式:	句法:
neutral 文本	`SIN(source,destination);`
ASCII 文本	`SIN source destination`

相关指令: CMP，CPT，COS，TAN，ASN，ACS，ATN，DEG，RAD

余弦指令 (COS)

COS 指令是一条输出指令。



操作数

操作数:	数据类型:	格式:	说明:
源	SINT INT DINT **REAL**	立即数 标签	计算该值的余弦值。
目的	SINT INT DINT **REAL**	标签	存储运算结果的标签。

说明: COS 指令计算源值的余弦值 (以弧度表示) 并存储结果于目的单元内。

源值必须大于或等于 -205887.4 并且小于或等于 205887.4。目的单元内的结果数值总是大于或等于 -1 并且小于或等于 1。

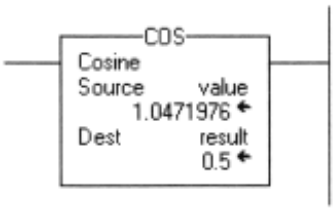
执行:

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	控制器计算源数值的余弦值并存放结果于目的单元。 梯级输出条件被设置为真。

算术状态标志: 影响算术状态标志

故障条件: 无

COS 指令举例



当指令被使能时，COS 指令计算 *value* 的余弦值并存放结果于 *result* 内。

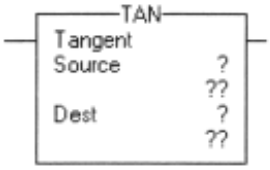
其他格式:

格式:	句法:
neutral 文本	`COS(source,destination);`
ASCII 文本	`COS source destination`

相关指令: CMP，CPT，SIN，TAN，ASN，ACS，ATN，DEG，RAD

正切指令 (TAN)

TAN 指令是一条输出指令。



操作数:

操作数:	数据类型:	格式:	说明:
源	SINT INT DINT **REAL**	立即数 标签	计算该值的正切值。
目的	SINT INT DINT **REAL**	标签	存储运算结果的标签。

说明:

TAN 指令计算源值的正切值 (以弧度表示) 并存储结果于目的单元内。

源值必须大于或等于 -102943.7 并且小于或等于 102943.7。

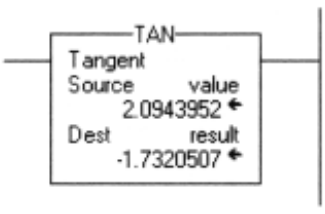
执行:

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	控制器计算源值的正切值并存放结果于目的单元。 梯级输出条件被设置为真。

算术状态标志: 影响算术状态标志

故障条件: 无

TAN 指令举例:



当指令被使能时，TAN 指令计算 *value* 的正切值并存放结果于 *result* 内。

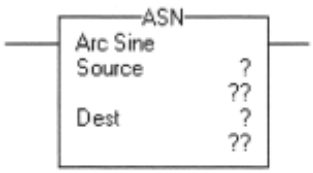
其他格式:

格式:	句法:
neutral 文本	`TAN(source,destination);`
ASCII 文本	`TAN source destination`

相关指令: CMP，CPT，COS，SIN，ASN，ACS，ATN，DEG，RAD

反正弦指令 (ASN)

ASN 指令是一条输出指令。



操作数:

操作数:	数据类型:	格式:	说明:
源	SINT INT DINT **REAL**	立即数 标签	计算该值的反正弦值。
目的	SINT INT DINT **REAL**	标签	存储运算结果的标签。

说明:

ASN 指令计算源值的反正弦值并存储运算结果于目的单元内(以弧度表示)。

源数值必须大于或等于-1 并且小于或等于1。目的单元内的结果数值总是大于或等于 $-\pi 2$ 并且小于或等于 $\pi 2$ (这里 π 3.141593)。

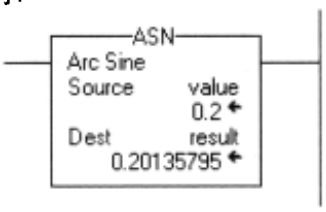
执行:

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	控制器计算源数值的反正弦值并存放结果于目的单元内。 梯级输出条件被设置为真。

算术状态标志: 影响算术状态标志

故障条件: 无

ASN 指令举例:



当指令被使能时，ASN 指令计算 *value* 的反正弦值并存放结果于 *result* 内。

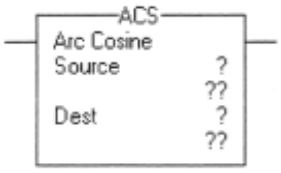
其他格式:

格式:	句法:
neutral 文本	`ASN(source,destination);`
ASCII 文本	`ASN source destination`

相关指令: CMP，CPT，ACS，ATN，SIN，COS，TAN，DEG，RAD

反余弦指令 (ACS)

ACS 指令是一条输出指令。



操作数:

操作数:	数据类型:	格式:	说明:
源	SINT INT DINT **REAL**	立即数 标签	计算该值的反余弦值。
目的	SINT INT DINT **REAL**	标签	存储运算结果的标签。

说明:

ACS 指令计算源数值的反余弦值并存储结果于目的单元(以弧度表示)。

源数值必须大于或等于-1 并且小于或等于1。目的单元内的结果值总是大于或等于 0 并且小于或等于 π (这里 π = 3.141593)。

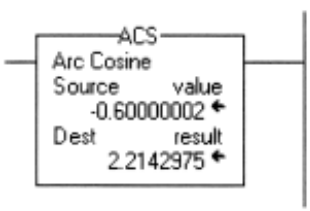
执行:

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	控制器计算源数值的反余弦值并存放结果于目的单元。 梯级输出条件被设置为真。

算术状态标志: 影响算术状态标志

故障条件: 无

ACS 指令举例:



当指令被使能时，ACS 指令计算 *value* 的反余弦值并存放结果于 *result* 内。

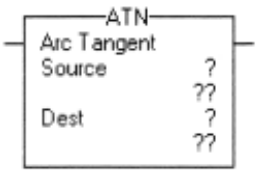
其他格式:

格式:	句法:
neutral 文本	`ACS(source,destination);`
ASCII 文本	`ACS source destination`

相关指令: CMP，CPT，ASN，ATN，SIN，COS，TAN，DEG，RAD

反正切指令 (ATN)

ATN 指令是一条输出指令。



操作数:

操作数:	数据类型:	格式:	说明:
源	SINT INT DINT **REAL**	立即数 标签	计算该数值的反正切值。
目的	SINT INT DINT **REAL**	标签	存储结果的标签。

说明:

ATN 指令计算源数值的反正切值并存储结果于目的单元内(以弧度表示)。

源数值必须大于或等于-1 并且小于或等于1。目的单元内的结果值总是大于或等于 -π/2 且小于或等于π/2π(这里 π = 3.141593)。

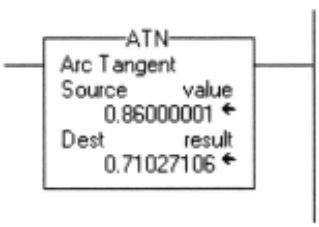
执行:

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	控制器计算源数值的反正切值并存放结果于目的单元内。 梯级输出条件被设置为真。

算术状态标志: 影响算术状态标志

故障条件: 无

ATN 指令举例:



当指令被使能时，ATN 指令计算 *value* 的反正切值并存放结果于 *result* 内。

其他格式:

格式:	句法:
neutral 文本	`ATN(source,destination);`
ASCII 文本	`ATN` *`source destination`*

相关指令: CMP，CPT，ACS，ASN，SIN，COS，TAN，DEG，RAD

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:

如果用户要:	使用下列指令:	参见页次:
计算一个数值的自然对数。	LN	14 - 2
计算一个数值的以 10 为底的对数。	LOG	14 - 4
对一个数值进行自乘(乘方)运算，自乘次数由另一个数值确定。	XPY	14 - 6

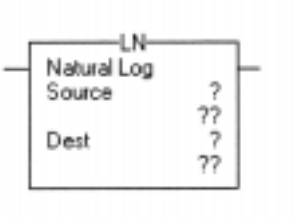
在运算时用户可以使用混合数据类型，但是这样会损失精度或发生取整误差，而且会增加指令的执行时间。检测 S:V 位以确认结果是否被截短。

黑体字数据类型是最优数据类型。如果指令的所有操作数都使用同一最优数据类型，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。

只要梯级输入条件为真，则指令每次被扫描都执行一次。 如果用户希望指令只计算一次，则可以使用一条 ONS 指令来触发算术指令。

自然对数指令 (LN)

LN 指令是一条输出指令



操作数:

操作数:	数据类型:	格式:	说明:
源	SINT INT DINT **REAL**	立即数 标签	计算该值的自然对数。
目的	SINT INT DINT **REAL**	标签	存储运算结果的标签。

说明: LN 指令计算源数操作数的自然对数并存储结果于目的单元内。

源数值必须大于零，否则 S:V 被置位。目的单元内的结果数值大于或等于-87.33655并且小于或等于88.72284。

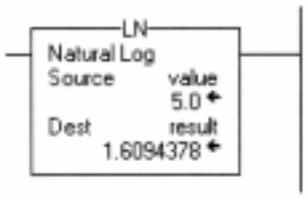
执行:

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	控制器计算源数值的自然对数并存放结果于目的单元。 梯级输出条件被设置为真。

算术状态标志: 影响算术状态标志

故障条件: 无

LN 指令举例:



当指令被使能时，LN 指令计算 value 的自然对数并存放结果于 result 内。

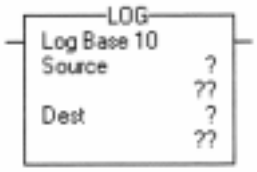
其他格式:

格式:	句法:
neutral 文本	`LN(source,destination);`
ASCII 文本	`LN source destination`

相关指令: CMP，CPT，LOG，XPY

计算以 10 为底的对数指令 (LOG)

LOG 指令是一条输出指令



操作数:

操作数:	数据类型:	格式:	说明:
源	SINT INT DINT **REAL**	立即数 标签	计算该值的对数。
目的	SINT INT DINT **REAL**	标签	存储运算结果的标签。

说明:

LOG 指令计算源操作数的以10为底的对数并存储结果于目的单元内。

源数值必须大于零，否则 S:V 被置位。目的单元内的结果数值大于或等于-37.92978并且小于或等于38.53184。

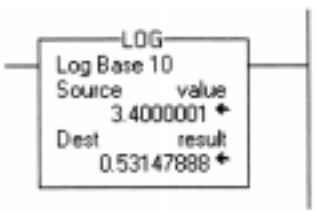
执行:

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	控制器计算源操作数的对数并存放结果于目的单元内。 梯级输出条件被设置为真。

算术状态标志: 影响算术状态标志

故障条件: 无

LOG 指令举例:



当指令被使能时，LOG 指令计算 value 的对数并存放结果于 result 内。

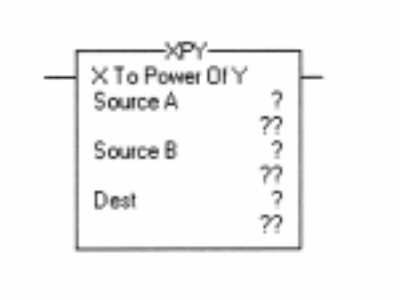
其他格式:

格式:	句法:
neutral 文本	`LOG(source,destination);`
ASCII 文本	`LOG source destination`

相关指令: CMP，CPT，LN，XPY

X 的 Y 次幂指令 (XPY)

XPY 指令是一条输出指令



操作数:

操作数:	数据类型:	格式:	说明:
源 A	SINT INT **DINT** **REAL**	立即数 标签	底数。
源 B	SINT INT **DINT** **REAL**	立即数 标签	指数。
目的	SINT INT **DINT** **REAL**	标签	存储运算结果的标签。

说明:

XPY 指令计算源 A(X)的源 B(Y)次幂并存储结果于目的单元。如果源 A 是负数，则源 B 必须是一个整数值， 否则将发生次要错误。

XPY 指令使用的运算法则是: Destination = X**Y

控制器计算 $x^0 = 1$； $0^x = 0$.

执行:

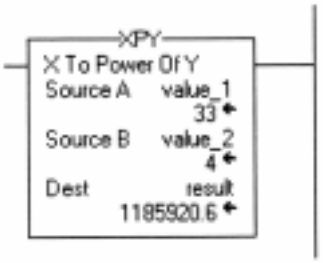
条件:	动作:
预扫描	梯级输出条件被设置为假。
梯级输入条件为假	梯级输出条件被设置为假。
梯级输入条件为真	控制器计算源 A(X)的源 B(Y)次幂并存放结果于目的单元。 梯级输出条件被设置为真。

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 A 的值是负数而源 B 的值是一个非整数值	4	4

XPY 指令举例:



当指令被使能时，xpy 指令计算 value_1 的 value_2 次幂且存放结果于

其他格式:

格式:	句法:
neutral 文本	XPY*(source_A,source_B,destination);*
ASCII 文本	XPY *source_A source_B destination*

相关指令: CMP，CPT，LN，LOG

注释:

数学转换指令

(DEG，RAD，TOD，FRD)

简介

数学转换指令可以转换数值。

如果用户要:	使用下列指令:	参见页次:
转换弧度为角度。	DEG	15 - 2
转换角度为弧度。	RAD	15 - 3
转换整数值为 BCD 码。	TOD	15 - 4
转换 BCD 码为整数值。	FRD	15 - 6

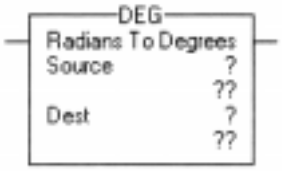
在运算时用户可以混合数据类型，但是这样会损失精度或发生取整误差，而且会增加指令的执行时间。检测 S:V 位以确认结果是否被截短。

黑体字数据类型是最优的数据类型。如果指令的所有操作数都使用同一最优数据类型，则指令执行的速度快而且占用内存少。典型最优数据类型是DINT 或 REAL。

只要梯级输入条件为真，指令每次被扫描都执行一次。如果用户只希望指令计算一次，则可以用一条 ONS 指令来触发转换指令。

转换为角度 (DEG)

DEG 指令是一条输出指令。



操作数:

操作数:	数据类型:	格式:	说明:
源	SINT INT DINT **REAL**	立即数 标签	转换该值为角度。
目的	SINT INT DINT **REAL**	标签	存储转换结果的标签。

说明:

DEG 指令转换源值(以弧度表示) 为角度并存储转换结果于目的单元内。

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	控制器转换源数值为角度值并存放结果于目的单元。 梯级输出条件被设置为真。

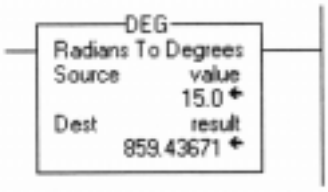
算术状态标志:

影响算术状态标志

故障条件:

无

DEG 指令举例:



当指令被使能时，DEG 指令转换 *value* 为角度并存放结果于 *result* 内。

其他格式:

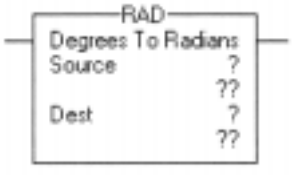
格式:	句法:
neutral 文本	`DEG(source,destination);`
ASCII 文本	`DEG source destination`

相关指令:

CMP，CPT，RAD，SIN，COS，TAN，ASN，ACS，ATN

转换为弧度(RAD)

RAD 指令是一条输出指令。



操作数

操作数:	数据类型:	格式:	说明:
源	SINT INT DINT **REAL**	立即数 标签	转换该数值为弧度。
目的	SINT INT DINT **REAL**	标签	存储转换结果的标签。

说明:

RAD 指令转换源值(以角度表示) 为弧度并存储结果于目的单元内。

RAD 指令使用的运算法则是: 源操作数 *π/180 (式中 π= 3.141593)

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	控制器转换源数值为弧度并存放结果于目的单元。 梯级输出条件被设置为真。

算术状态标志: 影响算术状态标志

故障条件: 无

RAD 指令举例:

当指令被使能时，RAD 指令转换 value 为弧度并存放结果于 result 内。

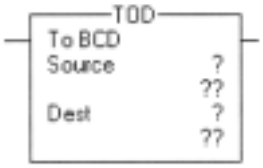
其他格式:

格式:	句法:
neutral 文本	*RAD(source,destination);*
ASCII 文本	*RAD source destination*

相关指令: CMP，CPT，DEG，SIN，COS，TAN，ASN，ACS，ATN

转换为 BCD 码指令 (TOD)

TOD 指令是一条输出指令。



操作数

操作数:	数据类型:	格式:	说明:
源	SINT INT **DINT**	立即数 标签	转换该值为 BCD 码。 (0 ≤源操作数 ≤99,999,999)
目的	SINT INT **DINT**	标签	存储转换结果的标签。

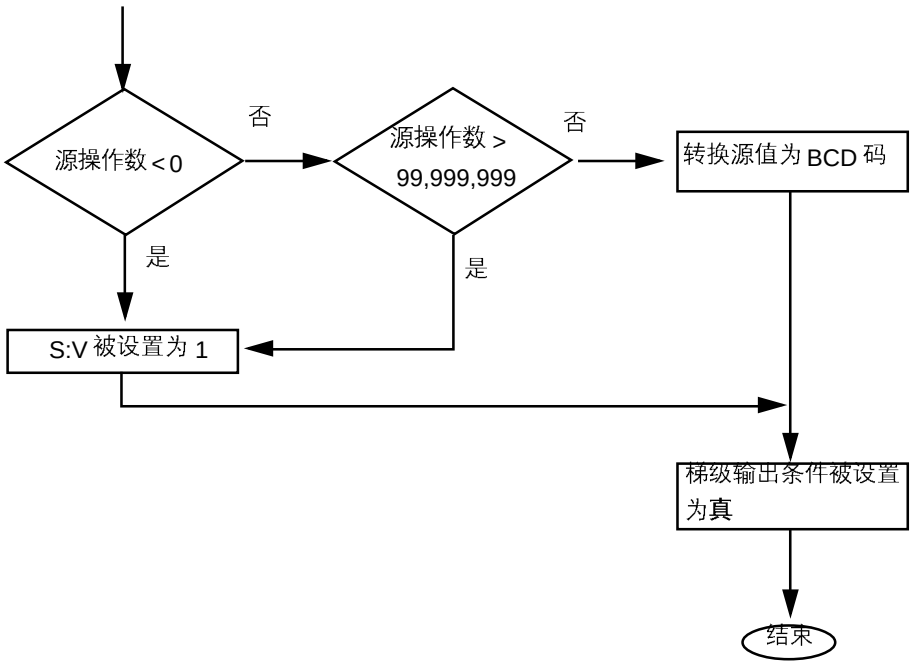
说明:

TOD 指令转换一个十进制的值(0 ≤源操作数 ≤ 99,999,999)为 BCD 码并存储转换结果于目的单元内。

如果输入源操作数一个负数，则指令发生次要故障并清零目的单元。

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	



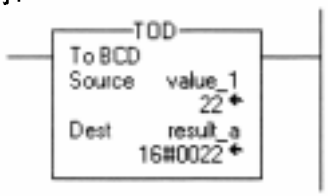
梯级输入条件为真	控制器转换源值为 BCD 码并存放结果于目的单元。 梯级输出条件被设置为真。

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:
源操作数 < 0	4	4

TOD 指令举例:



当指令被使能时，TOD 指令转换 *value_1* 为 BCD 码值并存放结果于 *result_a* 内。

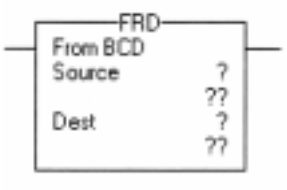
其他格式:

格式:	句法:
neutral 文本	`TOD(source,destination);`
ASCII 文本	`TOD source destination`

相关指令: CMP，CPT，FRD

转换为整数 (FRD)

FRD 指令是一条输出指令。



操作数

操作数:	数据类型:	格式:	说明:
源	SINT INT **DINT**	立即数 标签	转换该值为整数。
目的	SINT INT **DINT**	标签	存储转换结果的标签。

说明: FRD 指令转换一个 BCD 码值(源)为十进制值并存储转换结果于目的单元内。

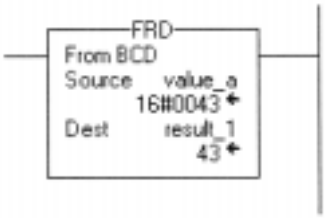
执行:

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	控制器转换源操作数为十进制值并存放结果于目的单元内。 梯级输出条件被设置为真。

算术状态标志: 影响算术状态标志

故障条件: 无

FRD 指令举例:



当指令被使能时，FRD 指令转换 value_ a 为十进制值并存放结果于 result_1 内。

其他格式:

格式:	句法:
neutral 文本	`FRD(source,destination);`
ASCII 文本	`FRD source destination`

相关指令: CMP，CPT，TOD

通用属性

简介

本附录讲述 Logix5550 指令的通用属性。

相关信息:	见页码:
算数状态标志	A - 1
数据类型	A - 4
关键字	A - 6

算数状态关键字

使用算数状态关键字检测算数状态标志的状态。

关键字:	状态标志:	说明:
S:V	溢出	如果要存储的值不能填入目的地，则溢出状态标志置位。或者该存储值大于目的的最大值或小于目的的最小值，则溢出状态标志置位。 **重点**: 每次 S:V 由清零变为置位，它都要产生一个次要故障(类型 4，代码 4)
S:Z	零	如果指令的目的值为零，则零状态标志置位。
S:N	符号(结果为负号)	如果指令的目的值为负，则符号状态标志置位。
S:C	进位	进位标志实际上不是数据类型的一部分。如果进位标志被存储到更大的数据类型，则进位标志代表位于数据类型内的位。 算数状态关键字无大小写区别。

因为算数状态标志变化非常迅速，所以在编程软件内的状态关键字不能动画显示实际的状态(不会显示不同的颜色)。

下表显示整型数据类型存储S:N和S:C状态标志的位置。

如果数据类型为 SINT

最大值 = +127
最小值 = -128

S:C	7 S:N	6	5	4	3	2	1	0

如果数据类型为 INT

最大值 = +32,767
最小值 = -32,768

S:C	15 S:N	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

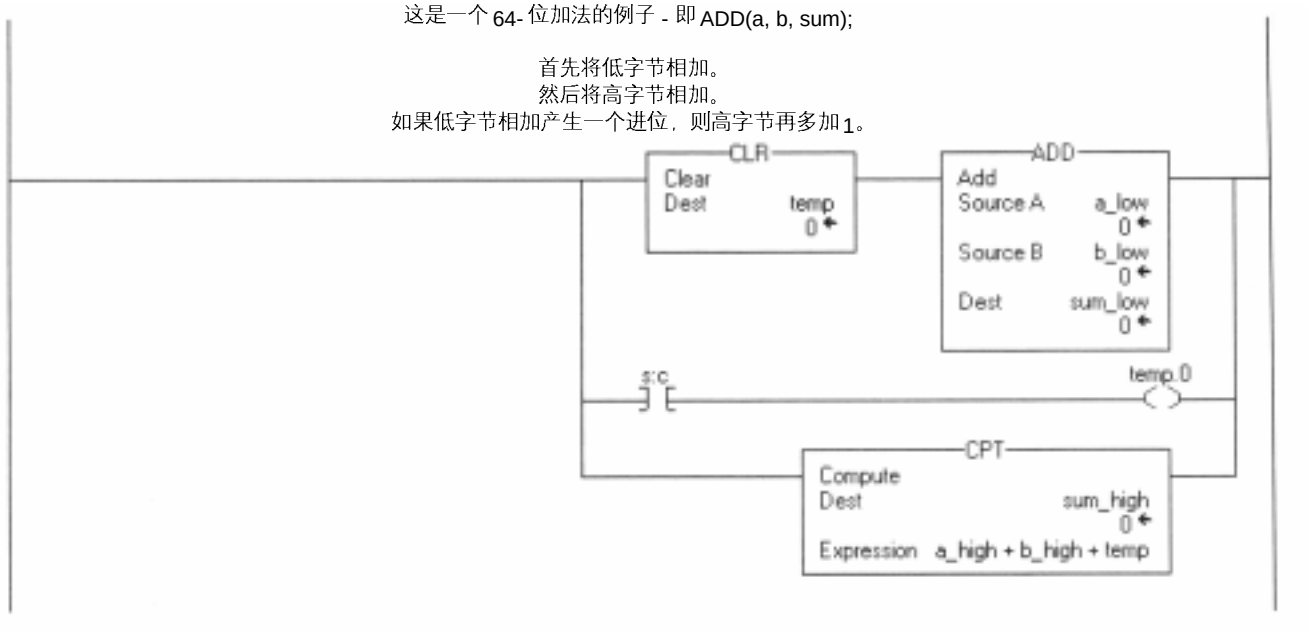
如果数据类型为 DINT

最大值 = +2,147,483,647
最小值 = -2,147,483,648

S:C	31 S:N	30	29	28	27	26	25	24	● ● ● ● ●	7	6	5	4	3	2	1

下面的编程示例显示怎样使用进位标志位。

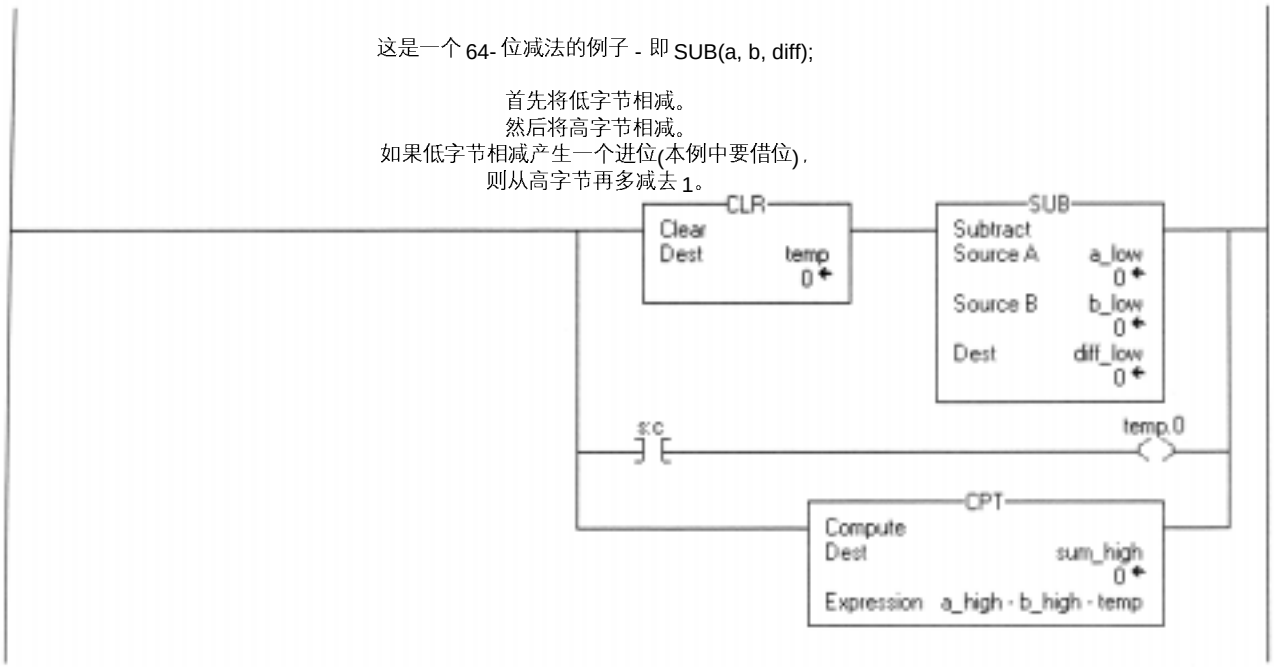
例 1



例 2

这是一个 64- 位减法的例子 - 即 SUB(a, b, diff);

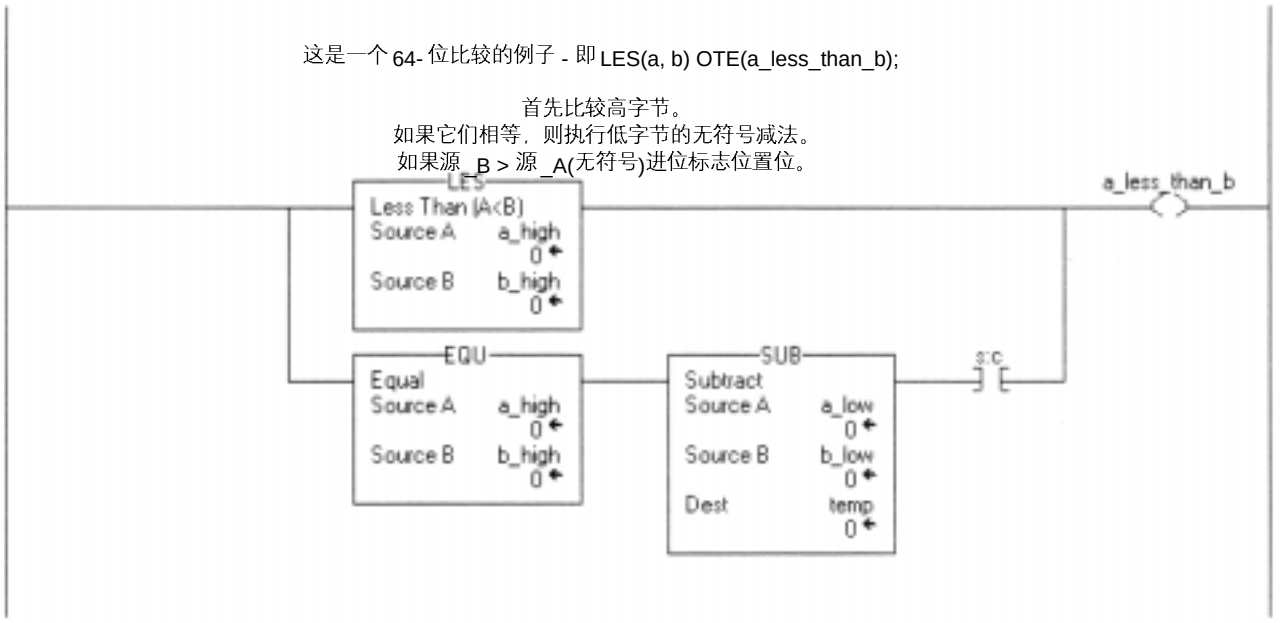
首先将低字节相减。
然后将高字节相减。
如果低字节相减产生一个进位(本例中要借位)，
则从高字节再多减去 1。



例 3

这是一个 64- 位比较的例子 - 即 LES(a, b) OTE(a_less_than_b);

首先比较高字节。
如果它们相等，则执行低字节的无符号减法。
如果源_B > 源_A(无符号)进位标志位置位。



数据类型

控制器数据类型仿照 IEC 1131-3 定义的数据类型。预定义的基本数据类型为:

数据类型:	说明:	范围:
BOOL	1- 位布尔值(逻辑值)	0 = 清零 1 = 置位
SINT	1- 字节整型数	-128 到 127
INT	2- 字节整型数	-32,768 到 32,767
DINT	4- 字节整型数	-2,147,483,648 到 2,147,483,647
REAL	4- 字节浮点数	$-3.402823E^{38}$ 到 $-1.1754944E^{-38}$(负值) 以及 0 以及 $1.1754944E^{-38}$ 到 $3.402823E^{38}$(正值)

控制器将所有的立即数处理为 DINT 数据类型。

REAL 数据类型也存储 ± 无穷和 ± NAN，但软件显示将根据显示格式的不同而不同。

显示格式:	等效于:
实数	+ 无穷 1.$
	- 无穷 -1.$
	+NAN 1.#QNAN
	-NAN -1.#QNAN
指数	+ 无穷 1.#INF000e+000
	- 无穷 -1. #INF000e+000
	+NAN 1.#QNAN00e+000
	-NAN -1.#QNAN00e+000

预定义的结构体有:

数据类型:	说明:
AXIS[1]	传动轴控制结构体
CONTROL	数组(文件)指令控制结构体
COUNTER	计数器指令控制结构体
MESSAGE[1]	MSG 指令控制结构体
MOTION_GROUP[1]	传动组控制结构体
MOTION_INSTRUCTION	运动指令控制结构体
PID	PID 指令控制结构体
TIMER	计时器指令控制结构体

1. 该结构体不支持数组,不能嵌入用户自定义结构体,并且不能通过JSR指令传送至其他程序。

数据类型转换

如果在指令内使用混合数据类型的操作数,有的指令会自动将数据转换为最佳数据类型。在某些情况下,控制器将数据转换为新的数据类型 有时控制器将数据转换为最佳数据类型。

转换:	结果:
大整型数转换为小整型数	控制器截去大整型数的高位部分并产生溢出。 例如: 十进制 二进制 DINT 65,665 0000_0000_0000_0001_0000_0000_1000_0001 INT 129 0000_0000_1000_0001 SINT -127 1000_0001
SINT 或 INT 转换为 REAL	不丢失数据精度
DINT 转换为 REAL	数据精度丢失。这两种数据类型存储数据都是 32 位,但 REAL 数据类型的 32 位中还包含指数值。如果丢失精度,控制器从 DINT 的最低有效位获取它。
REAL 转换为整型数	控制器对小数部分进行舍入并截取非-小数部分的高位。如果丢失数据,控制器置位溢出状态标志位。 舍入是指趋近于最近的偶数: 小于 .5 下舍入 等于 .5 舍去小数,等于最接近的偶数 大于 .5 上舍入 例如: REAL(源) DINT(结果) 1.6 2 -1.6 -2 1.5 2 -1.5 -2 1.4 1 -1.4 -1 2.5 2 -2.5 -2

不能对BOOL数据类型进行转换或将数据转换为BOOL数据类型。

要点: 算数状态标志位在所存储数值的基础上置位。在通常情况下，指令不会影响算数状态关键字，但如果由于指令参数中的混合数据类型而发生数据转换时，则表现出指令将影响算数状态关键字。实际上是类型转换过程对算数状态关键字进行了置位。

其他关键字

除了算数状态关键字外，控制器还支持以下关键字。

关键字:	访问:	说明:
S:FS	读	如果这是在当前程序内进行的第一次正常程序扫描，则首次扫描标志位置位。
S:MINOR	读 写	如果产生了至少一个最小故障时，最小故障标志位置位。只有在由于程序运行而发生最小故障时，控制器才置位该位。对于其他不是由于程序运行而导致的最小故障，如电池短缺，控制器不会置位该位。
THIS	Na	THIS 语句只在 GSV 和 SSV 指令访问 TASK、PROGRAM、或 ROUTINE 时有效。使用 THIS 指定当前的 TASK、PROGRAM、或 ROUTINE。

关键字不区分大小写。

由于上述状态位不能快速改变，所以它们不能在编程软件内动态显示实际的状态。

用户**不能**对关键字定义标签别名。

数组概念

数组为元素的集合

数组是把一组数据(具有相同数据类型)用同一名称组合在一起并通过下标来标识单个元素。数组中的每个元素可以是基本数据类型或结构体。

通过下标指定数组中的元素。下标在数组名的方括号内。下标规定了数组的维数。维数从零开始。

数组:	规定:
一维	*(数组名[下标_0])*
二维	*(数组名[下标_0，下标_1])*
三维	*(数组名[下标_0，下标_1，下标_2])*

数组最多到三维。数组内元素总数为每维大小的乘积。

数组: **存储数据如下图所示:** **举例:**

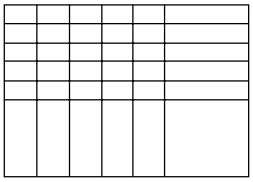
一维

标签名称:	类型	0 维	1 维	2 维
一维数组	DINT[7]	7	--	--

元素总数 = 7

DINT[x]的有效下标范围: x = 0-6

二维

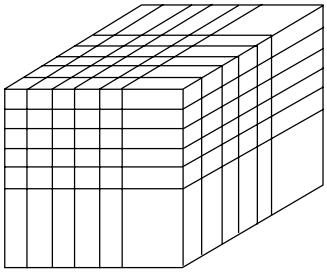


标签名称:	类型:	0 维	1 维	2 维
二维数组	DINT[4, 5]	4	5	--

元素总数 = 4 * 5 = 20

DINT[x, y]的有效下标范围: x = 0-3; y = 0-4

三维



标签名称:	类型:	0 维	1 维	2 维
一维数组	DINT[2,3,4]	2	3	4

元素总数 = 2 * 3 * 4 = 24

DINT[x,y,z]的有效下标范围: x = 0-1; y = 0-2; z = 0-3

通过数组变址寻址

要想动态改变逻辑引用的数组元素，可使用标签或表达式作为下标指向元素。这与PLC-5逻辑中的间接寻址类似。在作为数组下标的表达式中可使用下类运算符:

运算符:	说明:
+	加
-	减 / 取反
*	乘
/	除
AND	与
FRD	BCD 码转换为整型数
NOT	补码
OR	或
TOD	整型数转换为 BCD 码
SQR	平方根
XOR	异或

举例:

定义:		举例:	说明:
my_list	定义为 DINT[10]	*my_list[5]*	本例引用数组里的元素 5。因为下标值为一常数，所以该引用是静态引用。
my_list	定义为 DINT[10]	*将数值 5 移入 position*	本例引用数组里的元素5。因为通过改变position的值可以逻辑改变下标值，所以该引用是动态引用。
position	定义为 DINT	*my_list[position]*	
my_list	定义为 DINT[10]	*将数值 2 移入 position*	本例引用数组里的元素 7(2+5)。因为通过改变position或offset的值可以逻辑改变下标值，所以该引用是动态引用。
position	定义为 DINT	*将数值 5 移入 offset*	
offset	定义为 DINT	*my_list[position+offset]*	

要确保所输入数组下标在指定数组的范围内。如果下标超出其对应的维数，则表明数组为元素集合的指令将产生一个主要故障(类型 4，错误代码 20)。

指定数组内的位

用户可寻址数组元素内的位。例如:

定义:		举例:	说明:
array1	定义为 DINT[5]	*array1[1].2*	本例引用数组元素 1 内的第 2 位。
array2	定义为 INT[17, 36] 第一维有 17 个元素 第二维有 36 个元素	*array2[3,4].15*	本例引用元素 array2[3,4]内的第 15 位。
array3	定义为 SINT[2,4,6] 第一维有 2 个元素 第二维有 4 个元素 第三维有 6 个元素	*array3[1,3,2].4*	本例引用元素 array3[1,3,2]内的第 4 位。
MyArray *MyIndex*	定义为 SINT[100] 定义为 SINT	*MyArray[(MyIndex AND NOT 7) /8].[MyIndex AND 7]*	本例引用 SINT 数组内的一位。
MyArray *MyIndex*	定义为 INT[100] 定义为 INT	*MyArray[(MyIndex AND NOT 15) /16].[MyIndex AND 15]*	本例引用 INT 数组内的一位。
MyArray *MyIndex*	定义为 DINT[100] 定义为 DINT	*MyArray[(MyIndex AND NOT 31) /32].[MyIndex AND 31]*	本例引用 DINT 数组内的一位。

用户还可使用 B-2 页表中的运算符来指定位。

将数组看作一存储块

数组内的数据连续存储在内存里。文件指令一般都要求要有数组的起始地址和长度，这决定了指令要读、写多少元素以及哪些元素被读写。

要点: 如果指令读数据超出了数组范围，则无论那里的数据如何，指令都会读取它并进行处理，就象它是个有效数据一样(不发生错误)。如果指令写数据超出了数组的范围，则发生一主要错误(类型 4，错误代码 20)。

以下指令将数组数据作为连续存储块进行处理(其余指令将数组数据作为单个元素进行处理)

BSL	FLL
BSR	LFL
COP	LFU
DDT	SQI
FBC	SQL
FFL	SQO
FFU	

控制器如何存储数组数据

下表列出了 B-1 页示例中元素的顺序。

一维数组元素 (升序):	二维数组元素 (升序):	三维数组元素 (升序):
one_d_array[0]	*two_d_array[0,0]*	*three_d_array[0,0,0]*
one_d_array[1]	*two_d_array[0,1]*	*three_d_array[0,0,1]*
one_d_array[2]	*two_d_array[0,2]*	*three_d_array[0,0,2]*
one_d_array[3]	*two_d_array[0,3]*	*three_d_array[0,0,3]*
one_d_array[4]	*two_d_array[0,4]*	*three_d_array[0,1,0]*
one_d_array[5]	*two_d_array[1,0]*	*three_d_array[0,1,1]*
one_d_array[6]	*two_d_array[1,1]*	*three_d_array[0,1,2]*
	two_d_array[1,2]	*three_d_array[0,1,3]*
对于一维数组，	*two_d_array[1,3]*	*three_d_array[0,2,0]*
tag_name[subscript_0],	*two_d_array[1,4]*	*three_d_array[0,2,1]*
subscript_0 递增到其最大值。	*two_d_array[2,0]*	*three_d_array[0,2,2]*
	two_d_array[2,1]	*three_d_array[0,2,3]*
	two_d_array[2,2]	*three_d_array[1,0,0]*
	two_d_array[2,3]	*three_d_array[1,0,2]*
	two_d_array[2,4]	*three_d_array[1,0,3]*
	two_d_array[3,0]	*three_d_array[1,1,0]*
	two_d_array[3,1]	*three_d_array[1,1,1]*
	two_d_array[3,2]	*three_d_array[1,1,2]*
	two_d_array[3,3]	*three_d_array[1,1,3]*
	two_d_array[3,4]	*three_d_array[1,2,0]*
		three_d_array[1,2,1]
	对于二维数组，	*three_d_array[1,2,2]*
	tag_name[subscript_0,subscript_1], subscript_0 保持为 0，而 subscript_1 从 0 递增到其最大值。然后 subscript_0 增加 1(如果 0 维大于 1)并保持为 1，而 subscript_1 再次由 0 递增到其最大值。按照这种模式一直进行到两个下标都达到各自的最大值。	*three_d_array[1,2,3]*
		对于三维数组，
		tag_name[subscript_0,subscript_1, subscript_2], subscript_0 保持为 0，而 subscipt_1 和 subscript_2 按照二维数组递增。然后 subscript_0 增加 1(如果 0 维大于 1)并保持为 1 直到 subscript_1 和 subscript_2 达到各自的最大值为止。按照这种模式一直进行到三个下标都达到各自的最大值。

维数变换

AVE，SRT，及STD指令有维数变换操作数。指令使用该操作数计算确定要读、写数组元素的偏移量。

数组:	要变换的维数:	偏移量:
一维	0	1
二维	0	Dimension_1
	1	1
三维	0	(dimension_1)*(dimension_2)
	1	Dimension_2
	2	1

数组的内存分配

数组使用的内存量由创建数组的数据类型决定。控制器内的最小数据分配单元为四个字节，与32 BOOLS，4 SINTs，2 INTs，或1 DINT相同。

下例给出不同数组的内存分配:

bit_values为BOOL[32]

该例为具有32位元素的数组，每个数据类型为BOOL (每个元素1位)。

位:	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
数据分配单元1	[31]	[30]	[29]	[28]	[27]	[26]	[25]	[24]	[23]	[22]	[21]	[20]	[19]	[18]	[17]	[16]
位:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
数据分配单元1续	[15]	[14]	[13]	[12]	[11]	[10]	[9]	[8]	[7]	[6]	[5]	[4]	[3]	[2]	[1]	[0]

small_values为SINT[8]

该例为具有8个元素的数组，每个数据类型为SINT (每个元素1个字节)。

位:	31 24	23 16	15 8	7 0
数据分配单元1	small_values[3]	small_values[2]	small_values[1]	small_values[0]
数据分配单元2	small_values[7]	small_values[6]	small_values[5]	small_values[4]

small_values为SINT[3]

该例为具有3个元素的数组，每个数据类型为SINT (每个元素1个字节)。由于最小数据分配单元为4个字节，所以最后一个字节为零。

位:	31 24	23 16	15 8	7 0
数据分配单元1	0	small_values[2]	small_values[1]	small_values[0]

values 为 INT[4]

该例为具有 4 个元素的数组，每个数据类型为 INT(每个元素 2 个字节)。

位:	31 16	15 0
数据分配单元 1	values[1]	values[0]
数据分配单元 2	values[3]	values[2]

big_values 为 DINT[2]

该例为具有 2 个元素的数组，每个数据类型为 DINT(每个元素 4 个字节)。

位:	31 0
数据分配单元 1	big_values[0]
数据分配单元 2	big_values[1]

timer_list 为 TIMER[2]

该例为具有 2 个元素的数组，每个元素为 TIMER 结构体(每个结构体 12 个字节)。

位:	31 0
数据分配单元 1	timer_list[0]状态位
数据分配单元 2	timer_list[0].pre
数据分配单元 3	timer_list[0].acc
数据分配单元 4	timer_list[1]状态位
数据分配单元 5	timer_list[1].pre
数据分配单元 6	timer_list[1]acc

small_values 为 SINT[2,2,2]

该例为具有 8 个元素的三维数组，每个数据类型为 SINT。

位:	31 24	23 16	15 8	7 0
数据分配单元 1	small_values[0,1,1]	small_values[0,1,0]	Small_values[0,0,1]	small_values[0,0,0]
数据分配单元 2	small_values[1,1,1]	small_values[1,1,0]	Small_values[1,0,1]	small_values[1,0,0]

big_values 为 DINT[2,2,2]

该例为具有 8 个元素的三维数组，每个数据类型为 DINT。

位:	31 0
数据分配单元 1	big_values[0,0,0]
数据分配单元 2	big_values[0,0,1]
数据分配单元 3	big_values[0,1,0]
数据分配单元 4	big_values[0,1,1]
数据分配单元 5	big_values[1,0,0]
数据分配单元 6	big_values[1,0,1]
数据分配单元 7	big_values[1,1,0]
数据分配单元 8	big_values[1,1,1]

用户在离线编程时可修改数组维数而不会丢失标记数据。在线编程时不能修改数组维数。

结构体

简介

本附录列出了预定义结构体及用于寻址结构体内成员的助记符。

1. 这些标签仅仅是控制器标签。它们不支持数组，不能嵌入用户自定义结构体，并且不能通过 JSR 指令传送至其他程序。

有些更复杂的结构体除了本附录描述的属性外还有其他属性，它们只能从结构体的配置标签进入。

AXIS 结构体(传动轴结构体)

每个 AXIS 结构体包含传动轴的状态和组态信息。

助记符:	字节偏移量:	数据类型:	说明:			
.MotionFault	12	DINT	传动轴的运动故障标志位。			
			位:	**号码:**	**数据类型:**	**说明:**
			.ACAsyncConnFault	00	BOOL	异步连接故障
			.ACSyncConnFault	01	BOOL	同步连接故障
.MotionStatus	16	DINT	传动轴的运动状态位。			
			位:	**号码:**	**数据类型:**	**说明:**
			.AccelStatus	00	BOOL	加速状态
			.DecelStatus	01	BOOL	减速状态
			.MoveStatus	02	BOOL	传送状态
			.JogStatus	03	BOOL	慢进给状态
			.GearingStatus	04	BOOL	齿轮传动状态
			.HomingStatus	05	BOOL	复位状态
			.ClutchStatus	06	BOOL	啮合状态
			.AxisHomedStatus	07	BOOL	复位状态状态
.ServoFault	20	DINT	伺服环的伺服故障标志位。			
			位:	**号码:**	**数据类型:**	**说明:**
			.POtrvlFault	00	BOOL	正向超程故障
			.NOtrvlFault	01	BOOL	负向超程故障
			.PosErrorFault	02	BOOL	位置错误故障
			.EncCHALossFault	03	BOOL	编码器 A 通道信号丢失故障
			.EncCHBLossFault	04	BOOL	编码器 B 通道信号丢失故障
			.EncCHZLossFault	05	BOOL	编码器 Z 通道信号丢失故障
			.EncNsFault	06	BOOL	编码器噪声故障
			.DriveFault	07	BOOL	驱动故障
	21		**位:**	**号码:**	**数据类型:**	**说明:**
			.SyncConnFault	00	BOOL	同步连接故障
			.HardFault	01	BOOL	伺服硬件故障
.ServoStatus	24	DINT	伺服环的状态位。			
			位:	**号码:**	**数据类型:**	**说明:**
			.ServoActStatus	00	BOOL	伺服操作
			.DriveEnableStatus	01	BOOL	驱动使能
			.OutLmtStatus	02	BOOL	输出界限
			.PosLockStatus	03	BOOL	位置锁定
			位:	**号码:**	**数据类型:**	**说明:**
			.TuneStatus	05	BOOL	调整过程
			.TestStatus	06	BOOL	测试诊断
			.ShutdownStatus	07	BOOL	传动轴关机
.EventStatus	36	DINT	伺服环的伺服事件状态位。			
			位:	**号码:**	**数据类型:**	**说明:**
			.WatchEvArmStatus	00	BOOL	监视事件准备
			.WatchEvStatus	01	BOOL	监视事件
			.RegEvArmStatus	02	BOOL	登录事件准备
			.RegEvStatus	03	BOOL	登录事件
			.HomeEvArmStatus	04	BOOL	复位事件准备
			.HomeEvStatus	05	BOOL	复位事件

助记符:	字节偏移量:	数据类型:	说明:			
.UpdateStatus	40	DINT	传动轴的伺服状态更新位。			
			位:	号码:	数据类型:	说明:
			.AxisTypeStatus	00	BOOL	传动轴类型
			.PosUnwindStatus	01	BOOL	位置循环
			.MaxPTrvlStatus	02	BOOL	最大正向行程
			.MaxNTrvlStatus	03	BOOL	最大负向行程
			.PosErrorTolStatus	04	BOOL	位置错误容差
			.PosLockTolStatus	05	BOOL	位置锁定容差
			.PosPGainStatus	06	BOOL	位置比例增益
			.PosIGainStatus	07	BOOL	位置积分增益
			.VelFfGainStatus	08	BOOL	速度前馈增益
			.AccFfGainStatus	09	BOOL	加速度前馈增益
			.VelPGainStatus	10	BOOL	速度比例增益
			.VelIGainStatus	11	BOOL	速度积分增益
			.OutFiltBwStatus	12	BOOL	输出转换器带宽
			.OutScaleStatus	13	BOOL	伺服环输出转换为驱动等效电压
			.OutLimitStatus	14	BOOL	最大伺服输出电压
			.OutOffsetStatus	15	BOOL	伺服模块DAC输出和伺服驱动输入的累积偏移量的影响
			.FricCompStatus	16	BOOL	摩擦补偿
			.PotrvlFaultActStatus	17	BOOL	正向超程故障
			.PosErrorFaultActStatus	18	BOOL	位置故障
			.EncLossFaultActStatus	19	BOOL	编码器丢失故障
			.EncNsFaultActStatus	20	BOOL	编码器噪声故障
			.DriveFaultActStatus	21	BOOL	驱动故障
每个结构体的总内存量			44 个字节(包括上述各项加上附加组态信息)			

CONTROL 结构体(控制结构体)

CONTROL 结构体控制数组指令的操作。

CONTROL 结构体

位号

31	30	29	28	27	26	25	24	16	15	0	
EN	EU	DN	EM	ER	UL	IN	FD				DINT
长度值(.LEN)											DINT
位置值(.POS)											DINT

每个结构体的总内存量 12 字节

COUNTER 结构体(计数器结构体)

COUNTER结构体存储用于计数器指令的状态位及预置值和累加值。

COUNTER 结构体

位号

31	30	29	28	27	1615	0	
CU	CD	DN	OV	UN			DINT
预置值(.PRE)							DINT
累加值(.ACC)							DINT

每个结构体的总内存量 12 字节

MESSAGE 结构 (体信息结构体)

每条 MSG 指令都有一包含指令状态信息的 MESSAGE 结构体

助记符:	字节偏移量:	数据类型:	说明:			
.FLAGS	04	INT	标志字将下列状态位存储在一个 16- 位字内。			
			位:	号码:	数据类型:	说明:
			.EW	02	BOOL	当控制器检测到信息要求进入到队列时，使能等待标志位置位。当 .ST 位置位时，控制器复位 .EW 位。
			.ER	04	BOOL	当控制器检测到传送失败时，错误标志位置位。当下一次梯级 - 条件 - 入由假变为真时，.ER 位复位。
			.DN	05	BOOL	当成功地传送完最后一个信息包后，完成标志位置位。当下一次梯级 - 条件 - 入由假变为真时，.DN 位复位。
			.ST	06	BOOL	当控制器开始执行 MSG 指令时，起动标志位置位。当 .DN 位或 .ER 位置位时，.ST 位复位。
			.EN	07	BOOL	当梯级 - 条件 - 入变为真时使能标志位置位，该位保持置位直到 .DN 位或 .ER 位置位并且梯级 - 条件 - 入为假为止。如果梯级 - 条件 - 入变为假，但 .DN 位和 .ER 位都清零，则 .EN 保持置位。
	05		位:	号码:	数据类型:	说明:
			.TO	08	BOOL	如果用户人为地置位 .TO 位，则控制器停止处理信息并且置位 .ER 位。
			.EN_CC	09	BOOL	使能高速缓冲存储器位确定怎样管理 MSG 连接。如果用户想要控制器保持连接(例如重复同一 MSG 指令许多次时)，则置位 .EN_CC 位。如果用户很少运行 MSG 指令并且对于控制器连接有其他需要，则清零 .EN_CC 位。
.ERR	06	INT	如果 .ER 位置位，错误代码字标识为 MSG 指令的错误代码。			
.EXERR	10	INT	扩展错误代码字专门用于标识某些错误代码的附加错误代码信息。			
.REQ_LEN	12	INT	所要求的长度指定信息指令要传送多少字。			
.DN_LEN	14	INT	完成的长度标识出实际传送了多少字。			
	每个结构体的总内存量		328 个字节(包括上述各项加上附加组态信息)			

MOTION_GROUP (传动组结构体)

每个控制器都有一个 MOTION_GROUP 结构体。该结构体包含传动组的状态和组态信息。

助记符:	字节偏移量:	数据类型:	说明:			
.GroupStatus	48	DINT	传动组的状态位。			
			位:	号码:	数据类型:	说明:
			.InhibStatus	00	BOOL	禁止状态
			.GroupSynced	01	BOOL	同步状态
.MotionFault	56	DINT	传动组的运动故障标志位。			
			位:	号码:	数据类型:	说明:
			.ACAsyncConnFault	00	BOOL	异步连接故障。
			.ACSyncConnFault	01	BOOL	同步连接故障。
.ServoFault	60	DINT	传动组的伺服 - 模块故障标志位。			
			位:	号码:	数据类型:	说明:
			.POtrvlFault	00	BOOL	正向超程故障
			.NOtrvlFault	01	BOOL	负向超程故障
			.PosErrorFault	02	BOOL	位置错误故障
			.EncCHALossFault	03	BOOL	编码器 A 通道信号丢失故障
			.EncCHBLossFault	04	BOOL	编码器 B 通道信号丢失故障
			.EncCHZLossFault	05	BOOL	编码器 Z 通道信号丢失故障
			.EncNsFault	06	BOOL	编码器噪声故障
			.DriveFault	07	BOOL	驱动故障
	61		位:	号码:	数据类型:	说明: :
			.SyncConnFault	00	BOOL	同步连接故障
			.HardFault	01	BOOL	伺服硬件故障
.GroupFault	64	DINT	传动组的故障标志位。			
			位:	号码:	数据类型:	说明:
			.GroupOverlapFault	00	BOOL	传动组重叠故障。
	每个结构体的总内存量		68 个字节(包括上述各项加上附加组态信息)			

MOTION_INSTRUCTION 结构体 (运动 _ 指令结构体)

每个运动指令都有一个包含指令状态信息的MOTION_INSTRUCTION结构体。

助记符:	字节偏移量:	数据类型:	说明:
无	00	BOOL	指令状态位有:

位:	号码:	数据类型:	说明:
.PC	26	BOOL	过程完成标志位表明操作完成。指令执行完成后，.DN 位置位。当启动过程完成时，.PC 位置位。
.IP	27	BOOL	过程进行标志位表明正在执行一过程。
.ER	28	BOOL	错误标志位表明操作何时产生溢出。
.DN	29	BOOL	完成标志位表明操作完成。
.EN	31	BOOL	使能标志位表明指令被使能。

.STATE	04	SINT	对于运动指令当控制器置位.EN 位时，运行状态始终设置为 0，其他运行状态则依赖于运动指令。
.STATUS	05	SINT	信息状态值表明与运动功能相关的任何信息的状态条件。

数值:	说明:
0000	信息成功。
0001	模块正在处理另一信息。
0002	模块正在等待前一信息的响应。
0003	对信息的响应失败。
0004	模块没有准备好通讯。

助记符:	字节偏移量:	数据类型:	说明:
.ERR	06	INT	错误数值包含与运动功能相关的错误代码。
		数值:	说明:
		3	当指令的另一个请求正在执行时试图执行该指令。 在控制器没有检查前一条指令的 .DN 位而执行通讯指令时会发生此类错误。
		4	试图在具有闭环伺服的传动轴上执行指令。
		5	试图在具有开环伺服的传动轴上执行指令。
		6	传动轴驱动被使能。
		7	传动轴处于关机状态。
		8	没有将轴组态成伺服轴类型。
		9	试图执行使电流加速超出界限的指令。
		10	主轴参考位置与从轴参考位置相同。
		11	传动轴没有被组态。
		12	向伺服模块通讯失败。
		13	指令试图使用超出范围的参数。
		14	因为运行调整指令时有错误，所以不能调整参数。
		15	因为运行诊断测试指令时有错误，所以该操作不能诊断参数。
		16	在进行复位过程中试图执行指令。
		17	在没有组态成旋转操作的轴上试图执行指令以使其旋转传送。
		18	传动轴的类型被组态为没有使用过的类型。
		19	传动组不在同步状态。丢失伺服模块或错误组态可造成此类错误。
		20	传动轴处于故障状态。
		21	传动组处于故障状态。
		22	当传动轴处于运动过程中时，试图执行 MSO(运动伺服有效)或 MAH(传动轴复位) 指令。
		23	一条指令试图进行非法动态改变，例如合并一 S- 曲线或改变 S- 曲线的加速度。
	每个结构体的总内存量	8 个字节	

PID 结构体

每个PID指令都有一个包含指令状态信息的PID结构体。

助记符:	字节偏移量:	数据类型:	说明:
.CTL	00	DINT	.CTL 成员将以下这些状态位存储在一个 16- 位字内。 用户可置位并清零这些状态位。

位:	号码:	数据类型:	说明:
.EN	31	BOOL	指令被使能
.CT	30	BOOL	级联类型(0 = 从; 1 = 主)
.CL	29	BOOL	级联环路(0 = 否; 1 = 是)
.PVT	28	BOOL	跟踪过程变量(0 = 否; 1 = 是)
.DOE	27	BOOL	...的微分(0=PV; 1 = 误差)
.SWM	26	BOOL	软件手动模式(0 = 否 - 自动; 1= 是 - 软件手动)
.CA	25	BOOL	控制作用 (0 = 正向(SP-PV); 1 = 反向(PV-SP))
.MO	24	BOOL	工作站模式 (0 = 自动; 1 = 手动)
.PE	23	BOOL	PID 方程(0 = 独立; 1 = 相关)
.NDF	22	BOOL	微分平滑处理(0 = 否; 1 = 是)
.NOBC	21	BOOL	反向偏置计算(0 = 否; 1 = 是)
.NOZC	20	BOOL	死区过零控制(0 = 否; 1 = 是)

	02		PID 指令设置下列状态位。用户可以清零这些状态位。

位:	号码:	数据类型:	说明:
.INI	15	BOOL	PID 初始化(0 = 否; 1 = 是)
.SPOR	14	BOOL	设置点超出范围(0 = 否; 1 = 是)
.OLL	13	BOOL	CV 低于输出限幅最小值(0 = 否; 1 = 是)
.OLH	12	BOOL	CV 高于输出限幅最大值(0 = 否; 1 = 是)
.EWD	11	BOOL	误差在死区范围内(0 = 否; 1 = 是)
.DVNA	10	BOOL	偏移下限报警(0 = 否; 1 = 是)
.DVPA	09	BOOL	偏移上限报警(0 = 否; 1 = 是)
.PVLA	08	BOOL	PV 下限报警 (0 = 否; 1 = 是)
.PVHA	07	BOOL	PV 上限报警 (0 = 否; 1 = 是)

.SP	04	REAL	设定点	
.KP	08	REAL	独立	比例增益(无量纲)
			相关	控制器增益(无量纲)
.KI	12	REAL	独立	积分增益(1 / 秒)
			相关	积分时间(分钟每循环)
.KD	16	REAL	独立	微分增益(秒)
			相关	微分时间(分钟)
.BIAS	20	REAL	前馈或偏置百分比	
.MAXS	24	REAL	最大工程单位定标值	
.MINS	28	REAL	最小工程单位定标值	
.DB	32	REAL	死区工程单位	
.SO	36	REAL	设置输出百分比	
.MAXO	40	REAL	最大输出限幅(输出的百分比)	
.MINO	44	REAL	最小输出限幅(输出的百分比)	
.UPD	48	REAL	环路更新时间(秒)	

助记符:	字节偏移量:	数据类型:	说明:
.PV	52	REAL	已定标的过程变量(PV)值
.ERR	56	REAL	已定标的误差值
.OUT	60	REAL	输出百分比
.PVH	64	REAL	过程变量上限报警值
.PVL	68	REAL	过程变量下限报警值
.DVP	72	REAL	正偏移报警极限
.DVN	76	REAL	负偏移报警极限
.PVDB	80	REAL	过程变量报警死区
.DVDB	84	REAL	偏移报警死区
.MAXI	88	REAL	最大过程变量(PV)值(未定标的输入)
.MINI	92	REAL	最小过程变量(PV)值(未定标的输入)
.TIE	96	REAL	手动控制的牵引值
.MAXCV	100	REAL	最大控制变量(CV)值(对应于 100%)
.MINCV	104	REAL	最小控制变量(CV)值(对应于 0%)
.MINTIE	108	REAL	最小牵引值(对应于 100%)
.MAXTIE	112	REAL	最大牵引值(对应于 0%)
.DATA[17]	116	REAL	.DATA[17] 中的各元素存储:

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]	百分比定标常数
.DATA[6]	.PV 的定标常数
.DATA[7]	微分定标常数
.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 的缩小比例常数
.DATA[16]	牵引值的缩小比例常数

	每个结构体的总内存量	120 个字节

TIMER 结构体(计时器结构体)

TIMER结构体为计时器指令存储状态位以及预置值和累加值。

TIMER 结构体

位号	31	30	29	16	15	0	
	EN	TT	DN				DINT
	预置值(.PRE)						DINT
	累加值(.ACC)						DINT

每个结构体的总内存量 12 字节

注释:

本部分名词术语特定为 ControlLogix 的术语。对于全面而完整的术语表参见 Industrial Automation Glossary (工业自动化术语)，出版号 AG-7.1。

A

别名标签 它可引用其他标签。别名标签能够引用其他别名标签或基本标签。它还可通过引用结构体成员、数组元素或标签、成员内的位来引用另一个标签的组件。参见*基本标签*。

基本数据类型 该基本定义用于分配内存的位、字节或字，并且定义了其标签的含义，包括 BOOL、SINT、INT、DINT、以及 REAL 数据类型。参见*数组*，*结构体*。

数组 是大量的变址元素序列，每个元素具有同样的数据类型。在 Logix5550 控制器内，下标从 0 开始并且一直扩展到元素总个数减 1 (以零为基准)。一个数组最多可以有三维，但如果是结构体的成员，那么它只能有一维。数组标签占据控制器里一块连续的内存，每个元素按顺序排列。参见*基本数据类型*，*结构体*。

应用 是例程、程序、任务及用于定义单个控制器操作的 I/O 组态的组合。参见*工程*。

B

基本标签 该标签定义了存储数据元素的内存区域。参见*别名标签*。

双向连接 这种连接允许数据双向流动: 从发信方到接收器以及从接收器到发信方。参见*连接*，*单向连接*。

二进制 以整数值显示并以 2 为基数 (每个数字代表一位)。前缀为 2#。将填充至布尔值或整数的字长 (1，8，16，或 32 位)。显示时，每四个数为一组，为了方便辨认，每组间用下划线分开。参见*十进制*，*十六进制*，*八进制*。

位 二进制位。最小的存储单元。采用数字 0 (清除) 和 1 (置位) 代表。

布尔型 存储一位 (0 或 1) 状态的基本数据类型。

字节 一个 8 位的存储单元。

C

高速缓冲存储器连接 对于 MSG 指令，高速缓冲存储器连接命令控制器保持连接 (即使在 MSG 指令完成后)。这对于用户重复运行 MSG 指令非常有利，因为每次初始化连接都会增加扫描时间。参见*连接，非高速缓冲存储器连接*。

状态改变(COS) I/O 模块上一个 I/O 点或一组 I/O 点状态的任何变化。

CIP 参见*控制和信息协议*。

通讯方式 定义 I/O 模块如何与控制器通讯。
选择通讯方式定义下列各项:
- 通过编程软件确定可使用什么样的组态表格
- 标签结构体和组态方法

兼容模块 是一种电子锁保护模式，为了建立与该模块的连接，它要求物理模块的厂家、类型号、主要版本属性与软件内已组态的模块匹配。参见*禁止锁，精确匹配*。

连接 从控制器到控制系统内其他模块的通讯机构。单个控制器的连接数量是有限的。I/O 模块通讯、接收型标签、产生型标签以及 MSG 指令使用连接传送数据。

接收型标签 从其他控制器接收数据的标签。接收型标签一直在控制器作用域内。参见*产生型标签*。

连续任务 连续运行的任务，当最后一条指令完成时，再次开始运行该程序。尽管不必有任何连续任务，但只能有一个连续任务。参见*周期任务*。

控制和信息协议 由 Allen-Bradley 的 ControlLogix 系列使用的控制设备通讯协议。是在 ControlNet 网络上使用的本地协议。

ControlBus(控制总线) 1756 机架使用的背板。它作为一个网络来使用。

控制器作用域 在控制器内任何地方都可存取数据。控制器包含标签(可被例程及程序内的别名标签引用) 的集合以及控制器作用域内的其他别名标签。*参见程序作用域。*

协调系统时间(CST) 在单个 ControlBus 框架内所有模块的同步时间值。来自单个 ControlBus 框架模块的具有 CST 数据的时间信息数据能够安全比较以确定数据采样间的相对时间。

计数器(COUNTER) 包含用于计数器指令的状态和控制信息的结构体数据类型。

D

数据类型 定义了内存的大小以及创建数据类型标签时将要分配的内存的格式。数据类型可为基本单元，结构体，或数组。

十进制 以整数值显示并以 10 为基数。没有前缀。不填充到整数的长度。*参见二进制，十六进制，八进制。*

说明 标签的说明最长为 120 字符；其他对象的说明最长为 128 字符。任何印刷字符都可使用，包括回车、制表和空格。

维数 数组的指定大小。数组最多可有三维。

DINT 存储 32- 位带符号整数值的基本数据类型 (-2,147,483,648 到 +2,147,483,647)。

直接 是一种I/O连接，是控制器建立的和I/O模块的一种单独连接。*参见机架优化。*

禁止锁 是一种电子锁保护模式，这种模式要求没有物理模块的属性和软件内组态的模块匹配，但仍然建立与该模块的连接。*参见兼容模块，精确匹配。*

下载 在工作站上将工程内容传送到控制器里的过程。参见*上载。*

E

运行时间 运行单向任务内所有操作的总时间。如果控制器组态为运行多任务，则运行时间包括其他任务执行操作所使用 / 共享的时间。参见*执行时间*。

电子锁 是 1756 I/O 系列的一个特点，它要求模块执行一种电子检查以确保物理模块与软件所组态的模块一致。通过软件使用户能够防止不经意地使用错误模块或对模块的错误修改。参见*兼容模块*，*禁止锁*，*精确匹配*。

元素 是更大数据单元子单元数据的地址单元。是数组的一个单元。参见*数组*。

精确匹配 是一种电子锁保护模式，为了建立与该模块的连接，它要求物理模块的所有属性 (厂家、类型号、主要版本及次要版本) 都和软件内组态的模块匹配。

执行时间 执行单个程序的总时间。执行时间仅包括该单个程序使用的时间，不包括其他任务内的程序执行其他操作所共享 / 使用的时间。参见*运行时间*。

指数 实数表示为科学型或指数型格式。总是在十进制小数点的左边有一位，然后是十进制部分，最后是一个指数。参见*格式*。

F

故障模式 控制器产生主要故障，不能清除故障，并且关机。

浮点数 实数表示为浮点格式。十进制小数点左边的位数不定，由数的量值决定。参见*格式*。

H

十六进制

按整数显示并以 16 为基数 (每位数代表四位)。前缀为 16#。填充到布尔值或整数的长度 (1、8、16、或 32 位)。每四位数为一组，为便于识别，每组间用下划线分隔。参见*二进制，十进制，八进制*。

I

立即数

真实的 32-位带符号实数或整数。不是存储数值的标签。

下标

用于指定数组内元素的索引。

INT

是一种存储 16- 位整数 (-32,768 到 +32,767) 的基本数据类型。

接口模块(IFM)

布好线的 I/O 现场接线装备。

L

只 - 听连接

是一种其他控制器拥有 / 提供组态数据给 I/O 模块的 I/O 连接。使用只 - 听连接的控制器不能写组态数据，当拥有者控制器正在控制 I/O 模块时，它只能保持与该 I/O 模块的连接。参见*拥有者控制器*。

M

主要故障

误操作、或者硬件、指令使主要故障位置位并且运行故障逻辑以清除故障条件。如果故障逻辑不能清除故障，逻辑运行停止，控制器关机，输出回复到配置状态。参见*故障状态，次要故障*。

主要版本

模块的 1756 系列有主要和次要版本指示。只要模块的功能改变，主要版本就要更新。参见*电子锁，次要版本*。

主协调系统时间 (CST)　在一个框架内，只有一个控制器被称为主协调系统时间 (CST)。框架内的所有其他模块使各自的 CST 值与主 CST 值同步。

成员　是结构体的一个元素，具有自己的数据类型和名称。成员也可以是结构体，创建嵌入的结构体数据类型。结构体内每个成员的数据类型可以不一致。参见*结构体*。

内存　是建立在控制器内部、用来保存程序和数据的电子型存储媒体。

次要故障　误操作、或者硬件、指令使次要故障位置位，但是允许继续进行逻辑扫描。参见*主要故障*。

次要版本　模块的 1756 系列有主要和次要版本指示。模块有变化 (该变化不会影响其功能或接口) 时更新次要版本。参见*电子锁，主要版本*。

广播　是一种模块在网络上发送数据而同时被一个以上的接收器接收的过程。用它来说明 ControlLogix I/O 系列产品的一个特点: 支持多个控制器同时从同一 I/O 模块接收输入数据。

多宿主　这是一种组态设置: 具有相同组态信息的一个以上控制器同时拥有同样的输入模块。

N

名称　名称用来指定标签和模块。命名习惯符合 IEC-1131-3 的规定。一个名称:

- 必须以字母字符 (A-Z 或 a-z) 或下划线 (_) 开头
- 可以只包含字母字符、数字字符、和下划线
- 最多有 40 个字符
- 不必有连续的或后缀下划线字符 (_)

网络刷新时间(NUT)　是指重复时间周期，在此周期内，数据能够在 ControlNet 网络上发送。网络刷新时间范围是: 2ms-100ms。

O

对象 存储状态信息的数据结构。当输入 GSV/SSV 指令时，用户需指定对象及要访问的对象属性。有时有一个以上的相同类型对象请求，所以用户还需要指定对象名称。例如，用户应用中有几项任务，每项任务有各自的 TASK 对象，用户通过任务名称访问 TASKA 对象。

八进制 以整数显示并以8为基数(每位数代表三位)。前缀为8#。填充到布尔值或整数的长度(1、8、16、或32位)。每四位数为一组，为便于识别，每组间用下划线分隔。参见*二进制*，*十进制*，*十六进制*。

拥有者控制器 该控制器创建原始组态以及到模块的通讯连接。拥有者控制器写组态数据并建立与模块的连接。参见*只-听连接*。

P

路径 一个设备和其他设备之间设备与网络的描述。从一个设备到另一个设备按照规定的路径连接。参见*连接*。

周期任务 在特定时间周期触发的任务。时间周期一到，该任务触发，执行任务程序。控制器里最多可有 32 个周期任务。参见*连续任务*。

周期任务重叠 正在执行一个任务时同一任务又再次触发，这时就会发生周期任务重叠。任务的执行时间大于为该任务组态的周期率。参见*周期任务*。

预扫描 是控制器的一项功能：在运行前检查逻辑以初始化指令和数据。

当用户将控制器由编程模式改变为运行模式时，控制器执行预扫描。

优先级 任务执行的优先顺序。如果在同一时间触发两个任务，则首先执行具有更高优先级的任务。优先级范围为 1-15，1 是最高优先级。如果在同一时间触发两个具有同样优先级的任务，则控制器在两个任务间每毫秒切换一次。连续任务运行在一固定的优先级水平上，它的优先级要低于控制器内所有其他任务的优先级。

后期扫描 是控制器的一项功能：在禁止程序前检查程序内的逻辑以复位指令和数据。

产生型标签 由控制器创建的由其他控制器使用的标签。产生型标签一直在控制器的作用域内。参见接收型标签。

产品自定义结构体 由软件和控制器自动定义的一种结构体数据类型。通过组态 I/O 模块就为该模块加入了产品自定义结构体。

程序 一个程序包含一套相关例程及标签的集合。当任务执行一个程序时，逻辑从组态好的主例程处开始执行。按照顺序，可使用 JSR 指令执行子例程。如果发生程序故障，执行跳转到程序的故障例程部分。这些例程都可访问该程序标签，但其他程序的例程则不能访问该程序标签。参见*例程*，*任务*。

程序作用域 只能在当前程序内存取数据。每个程序包含标签的集合，这些标签只能被该程序内的例程和别名标签引用。参见*控制器作用域*。

工程 编程软件用来存储控制器逻辑和配置的文件。参见*应用*。

R

机架优化 是一种 I/O 连接：1756-CNB 模块将数字 I/O 字采集进机架映象内 (类似于 1771-ASB)。优化的机架连接保存 ControlNet 连接和带宽，但使用这种连接类型时，只可利用有限的状态和诊断信息。参见*直接连接*。

REAL 是一种存储 32- 位 IEEE 浮点数值的基本数据类型。

带电插拔 (RIUP) 是 ControlLogix 的一个特点：允许用户在框架上电时安装或拆除模块。

请求信息包周期 (RPI) 在网络上通讯时，这是连续生成输入数据之间的最大时间量。一般该周期组态为微秒。实际上生成数据的时间被限制在网络刷新时间 (小于所选的 RPI) 的最大倍数。所选择的 RPI 必须大于或等于网络刷新时间。

例程 例程是一套使用一种编程语言如梯形图的逻辑指令。在控制器内，例程为工程提供运行代码。例程类似于 PLC 或 SLC 处理器内的程序文件。*参见程序，任务。*

S

扫描时间 *参见运行时间，执行时间。*

作用域 定义用户能够访问特定标签的区域。*参见控制器作用域，程序作用域。*

SINT 是一种存储 8- 位带符号整数值 (-128 到 +127) 的基本数据类型。

结构体 一个结构体存储一组数据，每个数据的数据类型可以不同。控制器具有它自己的预定义结构体。用户为控制器组态的每个 I/O 模块都有自己的预定义结构体。用户能够使用专用标签的组合以及其他结构体创建专用的用户 - 自定义结构体。*参见成员，用户 - 自定义结构体。*

格式 是数值显示的格式。*参见二进制，十进制，十六进制，八进制，浮点数，指数。*

系统内务操作时间 控制器分配给执行通讯和后台功能的时间百分比。

T

标签 控制器内存的命名区域。标签是用于分配内存、由逻辑引用数据并监控数据的基本机制。*参见别名标签，基本标签，接收型标签。*

任务

用于运行程序的一种调度机制。当触发一项任务时，最多有 32 个程序能够被调度执行。一项任务可组态为以连续任务运行或以周期任务运行。最多能创建 32 项任务去调度程序。参见*连续任务*，*周期任务*。

时间片

是 ControlLogix 的一个过程：记录输入数据的变化，以发生改变时刻为相对时间基准。

U

非高速缓冲存储器连接

对于 MSG 指令，非高速缓冲存储器连接在完成 MSG 指令后命令控制器关闭连接。清除高速缓冲存储器连接使其他控制器能够利用此连接。参见*连接*，*高速缓冲存储器连接*。

单向连接

是指数据只流向一个方向的连接：从始发站到接收站。参见*连接*，*双向连接*。

上载

在工作站上将控制器的内容传送入工程文件里的过程。参见*下载*。

用户自定义结构体

用户自定义结构体将不同类型的数据组合在一单独命名的实体内，它包含一个或多个叫做成员的数据定义。在用户 - 自定义结构体内创建成员就象创建专用标签。每个成员的数据类型决定了分配给成员的内存量。每个成员的数据类型可以是：

- 基本数据类型
- 产品 - 自定义结构体
- 用户 - 自定义结构体
- 基本数据类型的单维数组
- 产品 - 自定义结构体的单维数组
- 用户 - 自定义结构体的单维数组

W

看门狗

指定在触发主要控制器故障前任务能运行多长时间。

欢迎访问我们的网址：

www.rockwellautomation.com.cn

www.rockwellautomation.com

www.theautomationbookstore.com

Rockwell Automation Headquarters 1201 South Second Street, Milwaukee, WI 53204, USA, Tel:(1)414 382-2000, Fax:(1)414 382-4444

Rockwell Automation Asia Pacific Headquarters, 27/F Citicorp Centre, 18 Whitfield Road, Causeway Bay, Hong Kong, Tel:(852)28874788, Fax:(852)25081846

请与当地罗克韦尔自动化办公室联系：

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 – 上海市娄山关路 85 号东方国际大厦 D 座 406-407 室 邮编：200335 电话：(8621)62701878 传真：(8621)62756217

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 – 沈阳市沈河区市府大路 262 号甲新基火炬大厦 2101 室 邮编：110013 电话：(8624)22791907 传真：(8624)22791908

武汉 – 武汉市青山区和平大道 939 号 11 层 邮编：430081 电话：(8627)86543885 传真：(8627)86545529

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 2506 室 邮编：400010 电话：(8623)63702668 传真：(8623)63702558

Rockwell Automation

Publication 1756-6.4.1 ZH-January 2000