SIEMENS

SIMATIC

S7-300和S7-400的梯形图(LAD)编程

参考手册

2004年1月版
A5E00446504-01

安全指南

本手册包括了保证人身安全与保护本产品及连接的设备所应遵守的注意事项。这些注意事项在手册中均以下列符号加以突出，并根据危险等级标明如下：

危险
表示如果不采取适当的预防措施，将导致死亡、严重的人身伤害或财产损失。

警告
表示如果不采取适当的预防措施，可能导致死亡、严重的人身伤害或财产损失。

当心
表示如果不采取适当的预防措施，可能导致轻微的人身伤害。

当心
表示如果不采取适当的预防措施，可能导致财产损失。

须知
提醒您注意有关产品、产品使用的特别重要的信息，或者是文档的特定部分。

合格人员

只有**合格人员**才允许安装和操作该设备。合格人员是指被授权按照既定安全惯例和标准，对线路、设备和系统进行调试、接地和标记的人员。

正确使用

请注意如下事项：

警告
该设备及其部件只能用于产品目录或技术说明书中所描述的范畴，并且只能与由西门子公司认可或推荐的第三方厂商提供的设备或部件一起使用。

只有正确地运输、保管、设置和安装本产品，并且按照推荐的方式操作和维护，产品才能正常、安全地运行。

商标

SIMATIC®、SIMATIC HMI®和SIMATIC NET®是SIEMENS AG的注册商标。

本文档中的其它一些标志也是注册商标，如果任何第三方出于个人目的而使用，都会侵犯商标所有者的权利。

免责声明

我们已检查过本手册中的内容与所描述的硬件和软件相符。由于差错在所难免，我们不能保证完全一致。我们会定期审查本手册中的内容，并在后续版本中进行必要的更正。欢迎提出改进意见。

Siemens AG
Bereich Automation and Drives
Geschaeftsgebiet Industrial Automation Systems
Postfach 4848, D- 90327 Nuernberg

前言

用途

本手册是以梯形图(LAD)程序语言创建用户程序的指南。

本手册同时也包含了描述梯形图语言元素的语法和函数的参考章节。

基础知识要求

本手册供S7程序员、操作员以及维护/维修人员使用。

要了解本手册，需要具有自动化技术的常规知识。

除此之外，还需要具有计算机应用能力和其它类似于PC(例如，编程设备)的、使用MS Windows 2000专业版或MS Windows XP专业版操作系统的工作设备的知识。

手册应用范围

本手册适用于STEP 7编程软件包5.3版本。

符合IEC 1131-3标准

LAD符合国际电工技术委员会标准IEC 1131-3中定义的“梯形图”语言。
欲知更多详细资料，请参见STEP 7文件NORM_TBL.WRI中的标准表。

要求

要有效使用本梯形图手册，应已熟悉S7程序理论，该理论归档在STEP 7在线帮助中。语言包也使用STEP 7标准软件，因此应熟悉如何使用该软件并已阅读了随附文档。

本手册是文档包“STEP 7参考书目”中的一部分。

下表显示了STEP 7文档的总览：

文档	用途	订货号
STEP 7基础信息 • STEP 7 V5.3，使用入门手册 • 使用STEP 7 V5.3编程 • 组态硬件和通讯连接，STEP 7 V5.3版本 • 从S5到S7，变频器手册	提供给技术人员的基础信息，说明了使用STEP 7和S7-300/400可编程控制器来实现控制任务的方法。	6ES7810-4CA07-8BW0
STEP 7参考书目 • 用于S7-300/400的梯形图(LAD)/功能块图(FDB)/语句表(STL)手册 • S7-300/400的标准函数及系统函数	提供了参考信息，并说明了编程语言LAD、FBD、STL、标准函数以及系统函数，扩充了STEP 7基础信息的范围。	6ES7810-4CA07-8BW1

在线帮助	用途	订货号
STEP 7帮助	以在线帮助的形式，提供了使用STEP 7进行编程和组态硬件的基础信息。	STEP 7标准软件中的一部分。
AWL/KOP/FUP帮助参考 SFB/SFC帮助参考 组织块帮助参考	上下文相关参考信息。	STEP 7标准软件中的一部分。

在线帮助

集成于软件中的在线帮助是对本手册的补充。提供在线帮助的目的是，在使用该软件时提供详细的支持。

该帮助系统通过一些界面集成于软件中：

- 上下文相关帮助提供关于当前语境的信息，例如，打开的对话框或激活的窗口。可以通过通过菜单命令**帮助 > 上下文相关的帮助**，或按下F1键或通过使用工具栏上的问号符来打开上下文相关的帮助。
- 可以通过使用菜单命令**帮助 > 目录**，或在上下文相关的帮助窗口中按“STEP 7帮助”按钮来调用STEP 7中的常规帮助。
- 可以通过按“词汇表”按钮，调用所有STEP 7应用程序的词汇表。

本手册是“梯形图帮助”的摘要。由于手册和在线帮助具有完全相同的结构，因此非常容易在手册和在线帮助之间切换。

更多支持

如果有任何技术问题，请联系西门子代表或代理商。

您可以在下列网页中查找联系人：

http://www.siemens.com/automation/partner

培训中心

西门子提供了很多培训教程，帮助您熟悉SIMATIC S7自动化系统。请联系当地的培训中心，或位于德国纽伦堡(D 90327)的培训总部，以获取详细信息。

电话： +49(911) 895-3200。

网址： http://www.sitrain.com

A&D技术支持

遍布世界各处，24小时服务：

全球(纽伦堡) **技术支持** 每年365天，每天24小时 电话： +49(180) 5050-222 传真： +49(180) 5050-223 电子邮件： adsupport@ siemens.com 格林威治标准时间： +1:00		
欧洲/非洲(纽伦堡) **许可证** 当地时间： 周一至周五，8:00 - 5:00 PM 电话： +49(180) 5050-222 传真： +49(180) 5050-223 电子邮件： adsupport@ siemens.com 格林威治标准时间： +1:00	**美国**(约翰逊城) **技术支持和授权** 当地时间： 周一至周五，8:00 - 5:00 PM 电话： +1(423) 262 2522 传真： +1(423) 262 2289 电子邮件： simatic.hotline@ sea.siemens.com 格林威治标准时间： -5:00	**亚洲/澳洲**(北京) **技术支持和授权** 当地时间： 周一至周五，8:00 - 5:00 PM 电话： +86 10 64 75 75 75 传真： +86 10 64 74 74 74 电子邮件： adsupport.asia@ siemens.com 格林威治标准时间： +8:00
SIMATIC热线以及授权热线所使用的语言通常为德语和英语。		

Internet服务和支持

除文档以外，还在Internet上在线提供了知识产权信息，网址如下：

http://www.siemens.com/automation/service&support

可在其中查找下列内容：

- 公司简讯，经常提供产品的最新信息。
- 相应文档资料，可通过“服务和支持”中的搜索功能查找。
- 论坛，世界各地的用户和专家可以在此交流经验。
- 当地自动化和驱动办事处。
- 在“服务”页面下提供了关于现场服务、维修、备件等信息。

目录

1 位逻辑指令

1.1 位逻辑指令概述

说明

位逻辑指令使用1和0两个数字。这两个数字组成了名为二进制数字系统基础。将1和0两个数字称作二进制数字或位。在触点和线圈领域中，1表示激活或激励状态，0表示未激活或未激励状态。

位逻辑指令对1和0信号状态加以解释，并按照布尔逻辑组合它们。这些组合会产生由1或0组成的结果，称作“逻辑运算结果”(RLO)。

由位逻辑指令触发的逻辑运算可以执行各种功能。
有可以执行下列功能的位逻辑指令：

- ---| |--- 常开触点(地址)
- ---| / |--- 常闭触点(地址)
- ---(SAVE) 将RLO的状态保存到BR
- XOR 逻辑“异或”
- —( ) 输出线圈
- ---(#)--- 中间输出
- ---| NOT |--- 能流取反

RLO为1时将触发下列指令：

- ---(S) 置位线圈
- ---(R) 重置线圈
- SR 复位优先型SR双稳态触发器
- RS 置位优先型RS双稳态触发器

其它指令将对上升沿或下降沿过渡做出反应，执行下列功能：

- ---(N)--- RLO负跳沿检测
- ---(P)--- RLO正跳沿检测
- NEG 地址下降沿检测
- POS 地址上升沿检测
- 立即读取
- 立即写入

1.2 ---| |--- 常开触点(地址)

符号

<address>

---| |---

参数	数据类型	内存区域	说明
<address>	BOOL	I、Q、M、L、D、T、C	选中的位

说明

---| |--- 存储在指定**<地址>**的位值为“1”时，(常开触点)处于闭合状态。触点闭合时，梯形图轨道能流流过触点，逻辑运算结果(RLO) =“1”。

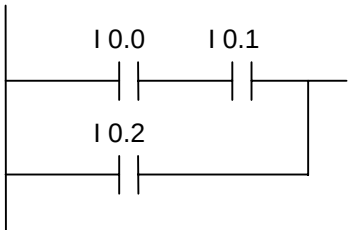
否则，如果指定**<地址>**的信号状态为“0”，触点将处于断开状态。触点断开时，能流不流过触点，逻辑运算结果(RLO) =“0”。

串联使用时，通过AND逻辑将 **---| |---** 与RLO位进行链接。并联使用时，通过OR逻辑将其与RLO位进行链接。

状态字

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
写:	-	-	-	-	-	X	X	X	1

实例



满足下列条件之一时，将会通过能流：

输入端I0.0和I0.1的信号状态为“1”时

或输入端I0.2的信号状态为“1”时

1.3 ---| / |--- 常闭触点(地址)

符号

<address>

---| / |---

参数	数据类型	内存区域	说明
<address>	BOOL	I、Q、M、L、D、T、C	选中的位

说明

---| / |--- 存储在指定**<地址>**的位值为“0”时，(常闭触点)处于闭合状态。触点闭合时，梯形图轨道能流流过触点，逻辑运算结果(RLO) =“1”。

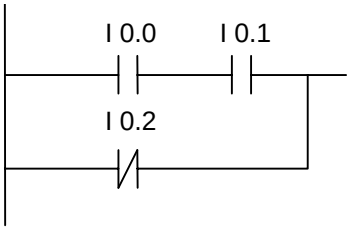
否则，如果指定**<地址>**的信号状态为“1”，将断开触点。触点断开时，能流不流过触点，逻辑运算结果(RLO) =“0”。

串联使用时，通过AND逻辑将, ---| / |--- 与RLO位进行链接。并联使用时，通过OR逻辑将其与RLO位进行链接。

状态字

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
写:	-	-	-	-	-	X	X	X	1

实例



满足下列条件之一时，将会通过能流：

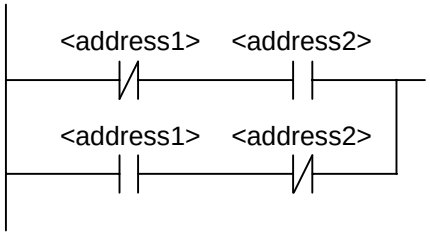
输入端I0.0和I0.1的信号状态为“1”时

或输入端I0.2的信号状态为“1”时

1.4 XOR 逻辑“异或”

对于XOR 函数，必须按以下所示创建由常开触点和常闭触点组成的程序段。

符号

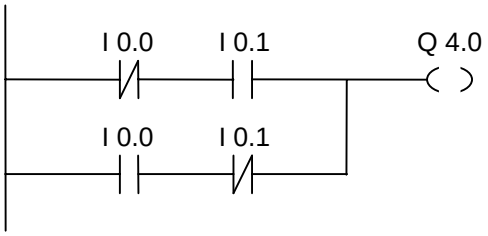


参数	数据类型	内存区域	说明
<address1>	BOOL	I、Q、M、L、D、T、C	扫描的位
<address2	BOOL	I、Q、M、L、D、T、C	扫描的位

说明

XOR(逻辑“异或”)如果两个指定位的信号状态不同，则创建状态为“1”的RLO。

实例



如果(I0.0 = “0”且I0.1 = “1”)或者(I0.0 = “1”且I0.1 = “0”)，输出Q4.0将是“1”。

1.5 --|NOT|-- 能流取反

符号

---| NOT |---

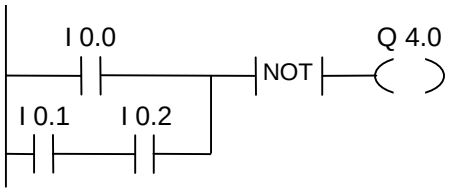
说明

---|NOT|--- (能流取反)取反RLO位。

状态字

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
写:	-	-	-	-	-	-	1	X	-

实例



满足下列条件之一时，输出端Q4.0的信号状态将是“0”：

输入端I0.0的信号状态为“1”时

或当输入端I0.1和I0.2的信号状态为“1”时。

1.6 ---( ) 输出线圈

符号

<address>

---( )

参数	数据类型	内存区域	说明
<address>	BOOL	I、Q、M、L、D	分配位

说明

---( ) (输出线圈)的工作方式与继电器逻辑图中线圈的工作方式类似。如果有能流通过线圈(RLO = 1)，将置位**<地址>**位置的位为“1”。如果没有能流通过线圈(RLO = 0)，将置位**<地址>**位置的位为“0”。只能将输出线圈置于梯级的右端。可以有多个(最多16个)输出单元(请参见实例)。使用---|NOT|---(能流取反)单元可以创建取反输出。

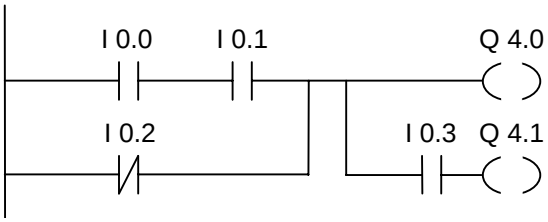
MCR (主控继电器)依存关系

只有在将输出线圈置于激活的MCR区内时，才会激活MCR依存关系。在激活的MCR区内，如果MCR处于接通状态并且输出线圈有能流通过，将把寻址位设置为能流的当前状态。如果MCR处于断开状态，则无论能流状态如何，都会将逻辑“0”写入指定地址。

状态字

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
写：	-	-	-	-	-	0	X	-	0

实例



满足下列条件之一时，输出端Q4.0的信号状态将是“1”：

输入端I0.0和I0.1的信号状态为“1”时

或输入端I0.2的信号状态为“0”时。

满足下列条件之一时，输出端Q4.1的信号状态将是“1”：

输入端I0.0和I0.1的信号状态为“1”时

或输入端I0.2的信号状态为“0”、输入端I0.3的信号状态为“1”时

如果实例梯级在激活的MCR区之内：

MCR处于接通状态时，将按照上述能流状态置位Q4.0和Q4.1。

MCR处于断开状态(=0)时，无论是否有能流通过，都将Q4.0和Q4.1复位为0。

1.7 ---(#)--- 中间输出

符号

<address>

---(#)---

参数	数据类型	内存区域	说明
<address>	BOOL	I、Q、M、*L、D	分配位

* 只有在逻辑块(FC、FB、OB)的变量声明表中将L区地址声明为TEMP时，才能使用L区地址。

说明

---(#)---
(中间输出)是中间分配单元，它将RLO位状态(能流状态)保存到指定**<地址>**。中间输出单元保存前面分支单元的逻辑结果。以串联方式与其它触点连接时，可以像插入触点那样插入---(#)---不能将---(#)---单元连接到电源轨道、直接连接在分支连接的后面或连接在分支的尾部。使用---|NOT|---(能流取反)单元可以创建取反---(#)---。

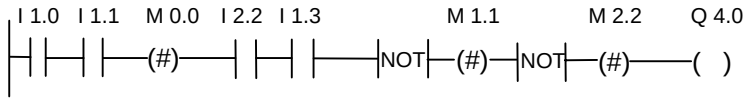
MCR (主控继电器)依存关系

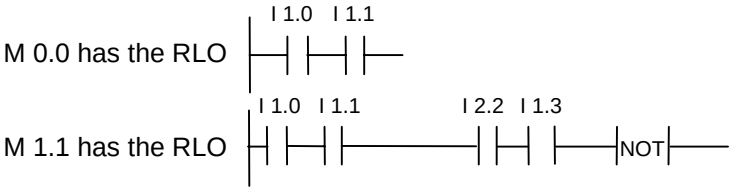
只有在将中间输出线圈置于激活的MCR区内时，才会激活MCR依存关系。在激活的MCR区内，如果MCR处于接通状态并且中间输出线圈有能流通过，将把寻址位设置为能流的当前状态。如果MCR处于断开状态，则无论能流状态如何，都会将逻辑“0”写入指定地址。

状态字

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
写：	-	-	-	-	-	0	X	-	1

实例





1.8 ---(R) 复位线圈

符号

<address>

---(R)

参数	数据类型	内存区域	说明
<address>	BOOL	I、Q、M、L、D、T、C	复位

说明

只有在前面指令的RLO为“1”(能流通过线圈)时，才会执行 **---(R)**(复位线圈)。如果能流通过线圈(RLO为“1”)，将把单元的指定**<地址>**复位为“0”。RLO为“0”(没有能流通过线圈)将不起作用，单元指定地址的状态将保持不变。**<地址>**也可以是值复位为“0”的定时器(T编号)或值复位为“0”的计数器(C编号)。

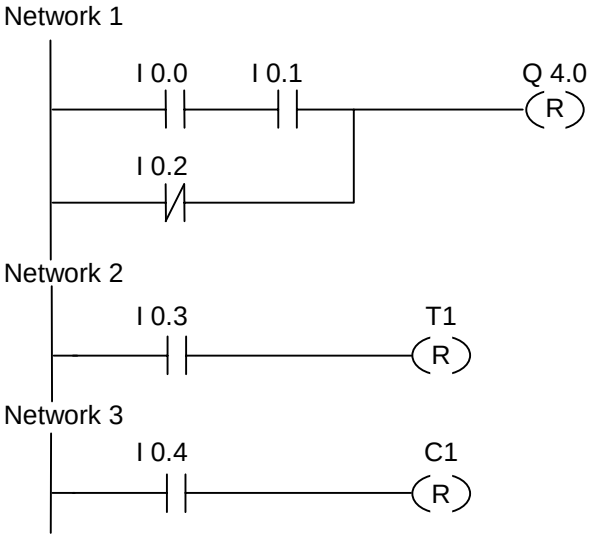
MCR (主控继电器)依存关系

只有将复位线圈置于激活的MCR区内时，才会激活MCR依存。在激活的MCR区内，如果MCR处于接通状态并且复位线圈有能流通过，将把寻址位状态复位为“0”。如果MCR处于断开状态，则无论能流状态如何，单元指定地址的当前状态均保持不变。

状态字

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
写：	-	-	-	-	-	0	X	-	0

实例



满足下列条件之一时，将把输出端Q4.0的信号状态复位为“0”：

输入端I0.0和I0.1的信号状态为“1”时

或输入端I0.2的信号状态为“0”时。

如果RLO为“0”，输出端Q4.0的信号状态将保持不变。

满足下列条件时才会复位定时器T1的信号状态：

输入端I0.3的信号状态为“1”时。

满足下列条件时才会复位计数器C1的信号状态：

输入端I0.4的信号状态为“1”时。

如果实例梯级在激活的MCR区之内：

MCR处于接通状态时，将按以上所述复位Q4.0、T1和C1。

MCR处于断开状态时，无论RLO的状态(能流状态)如何，Q4.0、T1和C1的状态均保持不变。

1.9 ---(S) 置位线圈

符号

<address>

---(S)

参数	数据类型	内存区域	说明
<address>	BOOL	I、Q、M、L、D	置位

说明

只有在前面指令的RLO为“1”(能流通过线圈)时，才会执行**---(S)(**置位线圈**)**。
如果RLO为“1”，将把单元的指定**<地址>**置位为“1”。

RLO = 0将不起作用，单元的指定地址的当前状态将保持不变。

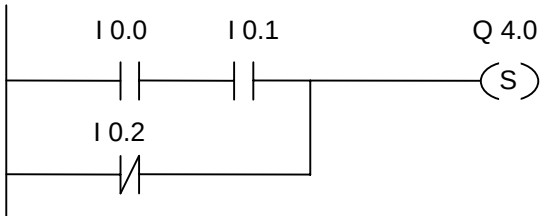
MCR (主控继电器)依存关系

只有将置位线圈置于激活的MCR区内时，才会激活MCR依存关系。在激活的MCR区内，如果MCR处于接通状态并且置位线圈有能流通过，将把寻址位的状态置位为“1”。如果MCR处于断开状态，则无论能流状态如何，单元指定地址的当前状态均保持不变。

状态字

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
写:	-	-	-	-	-	0	X	-	0

实例



满足下列条件之一时，输出端Q4.0的信号状态将是“1”：

输入端I0.0和I0.1的信号状态为“1”时

或输入端I0.2的信号状态为“0”时。

如果RLO为“0”，输出端Q4.0的信号状态将保持不变。

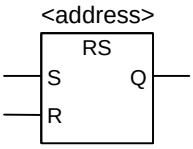
如果实例梯级在激活的MCR区之内：

MCR处于接通状态时，则按以上所述置位Q4.0。

MCR处于断开状态时，无论RLO状态(能流状态)如何，Q4.0状态均保持不变。

1.10 RS 置位优先型RS双稳态触发器

符号



参数	数据类型	内存区域	说明
<address>	BOOL	I、Q、M、L、D	置位或复位位
S	BOOL	I、Q、M、L、D	启用复位指令
R	BOOL	I、Q、M、L、D	启用复位指令
Q	BOOL	I、Q、M、L、D	<地址>的信号状态

说明

如果R输入端的信号状态为“1”，S输入端的信号状态为“0”，则复位**RS**(置位优先型RS双稳态触发器)。否则，如果R输入端的信号状态为“0”，S输入端的信号状态为“1”，则置位触发器。如果两个输入端的RLO均为“1”，则指令的执行顺序是最重要的。RS触发器先在指定**<地址>**执行复位指令，然后执行置位指令，以使该地址在执行余下的程序扫描过程中保持置位状态。

只有在RLO为“1”时，才会执行S(置位)和R(复位)指令。这些指令不受RLO“0”的影响，指令中指定的地址保持不变。

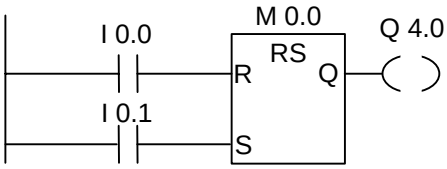
MCR (主控继电器)依存关系

只有将RS触发器置于激活的MCR区内时，才会激活MCR依存关系。在激活的MCR区内，如果MCR处于接通状态，则按以上所述将寻址位复位为“0”或置位为“1”。如果MCR处于关闭状态，则无论输入状态如何，指定地址的当前状态均保持不变。

状态字

	BR	**CC 1**	**CC 0**	**OV**	**OS**	**OR**	**STA**	**RLO**	**/FC**
写：	-	-	-	-	-	x	x	x	1

实例



如果输入端I0.0的信号状态为“1”，I0.1的信号状态为“0”，则置位存储器位M0.0，输出Q4.0将是“0”。否则，如果输入端I0.0的信号状态为“0”，I0.1的信号状态为“1”，则复位存储器位M0.0，输出Q4.0将是“1”。如果两个信号状态均为“0”，则不会发生任何变化。如果两个信号状态均为“1”，将因顺序关系执行置位指令；置位M0.0，Q4.0将是“1”。

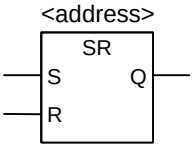
如果实例在激活的MCR区之内：

MCR处于打开状态时，将按以上所述复位或置位Q4.0。

MCR处于关闭状态时，无论输入状态如何，Q4.0均保持不变。

1.11 SR 复位优先型SR双稳态触发器

符号



参数	数据类型	内存区域	说明
<address>	BOOL	I、Q、M、L、D	置位或复位位
S	BOOL	I、Q、M、L、D	启用置位指令
R	BOOL	I、Q、M、L、D	启用复位指令
Q	BOOL	I、Q、M、L、D	<地址>的信号状态

说明

SR

如果S输入端的信号状态为“1”，R输入端的信号状态为“0”，则置位SR(复位优先型SR双稳态触发器)。否则，如果S输入端的信号状态为“0”，R输入端的信号状态为“1”，则复位触发器。如果两个输入端的RLO均为“1”，则指令的执行顺序是最重要的。SR触发器先在指定**<地址>**执行置位指令，然后执行复位指令，以使该地址在执行余下的程序扫描过程中保持复位状态。

只有在RLO为“1”时，才会执行S(置位)和R(复位)指令。这些指令不受RLO“0”的影响，指令中指定的地址保持不变。

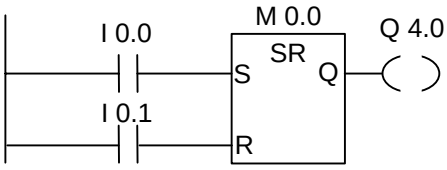
MCR (主控继电器)依存关系

只有将SR触发器置于激活的MCR区内时，才会激活MCR依存关系。在激活的MCR区内，如果MCR处于接通状态，则按以上所述将寻址位置位为“1”或复位为“0”。如果MCR处于关闭状态，则无论输入状态如何，指定地址的当前状态均保持不变。

状态字

	BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC
写：	-	-	-	-	-	x	x	x	1

实例



如果输入端I0.0的信号状态为“1”，I0.1的信号状态为“0”，则置位存储器位M0.0，输出Q4.0将是“1”。否则，如果输入端I0.0的信号状态为“0”，I0.1的信号状态为“1”，则复位存储器位M0.0，输出Q4.0将是“0”。如果两个信号状态均为“0”，则不会发生任何变化。如果两个信号状态均为“1”，将因顺序关系执行复位指令；复位M0.0，Q4.0将是“0”。

如果实例在激活的MCR区之内：

MCR处于打开状态时，将按以上所述置位或复位Q4.0。

MCR处于关闭状态时，无论输入状态如何，Q4.0均保持不变。

1.12 ---(N)--- RLO负跳沿检测

符号

<address>

---(N)

参数	数据类型	内存区域	说明
<address>	BOOL	I、Q、M、L、D	边沿存储位，存储RLO的上一信号状态

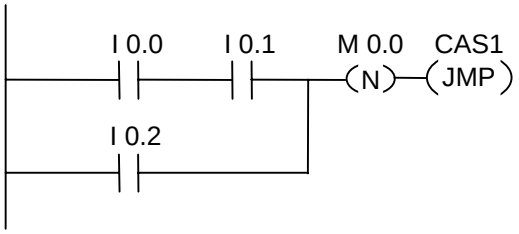
说明

---(N)---
(RLO负跳沿检测)检测地址中“1”到“0”的信号变化，并在指令后将其显示为RLO = “1”。将RLO中的当前信号状态与地址的信号状态(边沿存储位)进行比较。如果在执行指令前地址的信号状态为“1”，RLO为“0”，则在执行指令后RLO将是“1”(脉冲)，在所有其它情况下将是“0”。指令执行前的RLO状态存储在地址中。

状态字

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
写：	-	-	-	-	-	0	x	x	1

实例



边沿存储位M0.0保存RLO的先前状态。RLO的信号状态从“1”变为“0”时，程序将跳转到标号CAS1。

1.13 ---(P)--- RLO正跳沿检测

符号

<address>

---(P)---

参数	数据类型	内存区域	说明
<address>	BOOL	I、Q、M、L、D	边沿存储位，存储RLO的上一信号状态

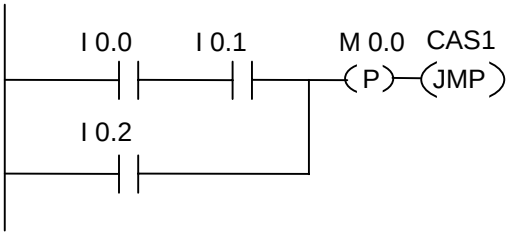
说明

---(P)---
(RLO正跳沿检测)检测地址中“0”到“1”的信号变化，并在指令后将其显示为RLO = “1”。将RLO中的当前信号状态与地址的信号状态(边沿存储位)进行比较。如果在执行指令前地址的信号状态为“0”，RLO为“1”，则在执行指令后RLO将是“1”(脉冲)，在所有其它情况下将是“0”。指令执行前的RLO状态存储在地址中。

状态字

	BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC
写：	-	-	-	-	-	0	X	X	1

实例



边沿存储位M0.0保存RLO的先前状态。RLO的信号状态从“0”变为“1”时，程序将跳转到标号CAS1。

1.14 ---(SAVE) 将RLO状态保存到BR

符号

---(SAVE)

说明

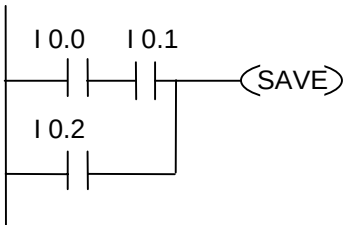
(SAVE)(将RLO状态保存到BR)将RLO保存到状态字的BR位。未复位第一个校验位/FC。因此，BR位的状态将包含在下一程序段的AND逻辑运算中。

指令“SAVE”(LAD、FBD、STL)适用下列规则，手册及在线帮助中提供的建议用法并不适用：
建议用户不要在使用SAVE后在同一块或从属块中校验BR位，因为这期间执行的指令中有许多会对BR位进行修改。建议用户在退出块前使用SAVE指令，因为ENO输出(= BR位)届时已设置为RLO位的值，所以可以检查块中是否有错误。

状态字

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
写：	X	-	-	-	-	-	-	-	-

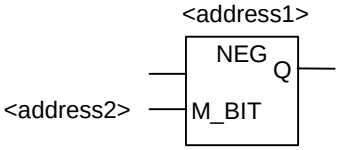
实例



将梯级(=RLO)的状态保存到BR位。

1.15 NEG 地址下降沿检测

符号



参数	数据类型	内存区域	说明
<address1>	BOOL	I、Q、M、L、D	已扫描信号
<address2>	BOOL	I、Q、M、L、D	M_BIT边沿存储位，存储<address1>的前一个信号状态
Q	BOOL	I、Q、M、L、D	单触发输出

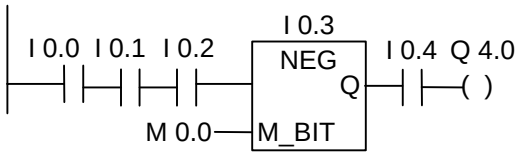
说明

NEG(地址下降沿检测)比较**<address1>**的信号状态与前一次扫描的信号状态(存储在**<address2>**中)。如果当前RLO状态为“1”且其前一状态为“0”(检测到上升沿)，执行此指令后RLO位将是“1”。

状态字

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
写:	x	-	-	-	-	x	1	x	1

实例

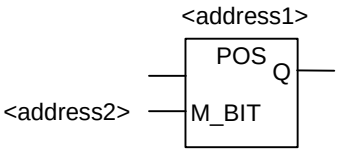


满足下列条件时，输出Q4.0的信号状态将是“1”：

- 输入I0.0、I0.1和I0.2的信号状态是“1”
- 输入I0.3有下降沿
- 输入I0.4的信号状态为“1”

1.16 POS 地址上升沿检测

符号



参数	数据类型	内存区域	说明
<address1>	BOOL	I、Q、M、L、D	已扫描信号
<address2>	BOOL	I、Q、M、L、D	M_BIT边沿存储位，存储<address1>的前一个信号状态
Q	BOOL	I、Q、M、L、D	单触发输出

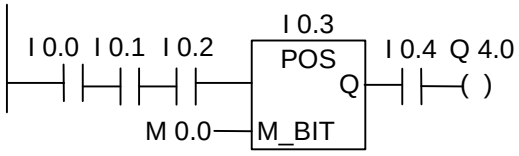
说明

POS(地址上升沿检测)比较**<address1>**的信号状态与前一次扫描的信号状态(存储在**<address2>**中)。如果当前RLO状态为“1”且其前一状态为“0”(检测到上升沿)，执行此指令后RLO位将是“1”。

状态字

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
写：	x	-	-	-	-	x	1	x	1

实例



满足下列条件时，输出Q4.0的信号状态将是“1”：

- 输入I0.0、I0.1和I0.2的信号状态是“1”
- 输入I0.3有上升沿
- 输入I0.4的信号状态为“1”

1.17 立即读取

说明

对于“立即读取”功能，必须按以下实例所示创建符号程序段。

对于对时间要求苛刻的应用程序，对数字输入的当前状态的读取可能要比正常情况下每OB1扫描周期一次的速度快。“立即读取”在扫描“立即读取”梯级时从输入模块中获取数字输入的状态。否则，必须等到下一OB1扫描周期结束，届时将以P存储器状态更新I存储区。

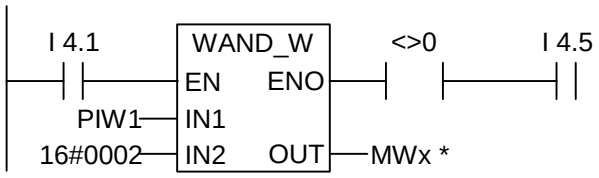
要从输入模块立即读取一个输入(或多个输入)，请使用外设输入(PI)存储区来代替输入(I)存储区。可以字节、字或双字形式读取外设输入存储区。因此，不能通过触点(位)元素读取单一数字输入。

根据立即输入的状态有条件地传递电压：

1. CPU读取包含相关输入数据的PI存储器的字。
2. 如果输入位处于接通状态(为“1”)，将对PI存储器的字与某个常数执行产生非零结果的AND运算。
3. 测试累加器的非零条件。

实例

可以立即读取外设输入I1.1的梯形图程序段



必须指定***** MWx，才能存储程序段。x可以是允许的任何数。

WAND_W指令说明：

PIW1	0000000000101010
W#16#0002	0000000000000010
结果	0000000000000010

在此实例中，立即输入I1.1与I4.1和I4.5串联。

字PIW1包含I1.1的立即状态。对PIW1与W#16#0002执行AND运算。如果PB1中的I1.1(第二位)为真(“1”)，则结果不等于零。如果WAND_W指令的结果不等于零，触点A<>0时将传递电压。

1.18 立即写入

说明

对于“立即写入”功能，必须按以下实例所示创建符号程序段。

对于对时间要求苛刻的应用程序，将数字输出的当前状态发送给输出模块的速度可能必须快于正常情况下在OB1扫描周期结束时发送一次的速度。“立即写入”将在扫描“立即写入”梯级时将数字输出写入输入模块。否则，必须等到下一OB1扫描周期结束，届时将以P存储器状态更新Q存储区。

要将一个输出(或多个输出)立即写入输出模块，请使用外设输出(PQ)存储区来代替输出(Q)存储区。可以字节、字或双字形式读取外设输出存储区。因此，不能通过线圈单元更新单一数字输出。要立即向输出模块写入数字输出的状态，将根据条件把包含相关位的Q存储器的字节、字或双字复制到相应的PQ存储器(直接输出模块地址)中。

当心

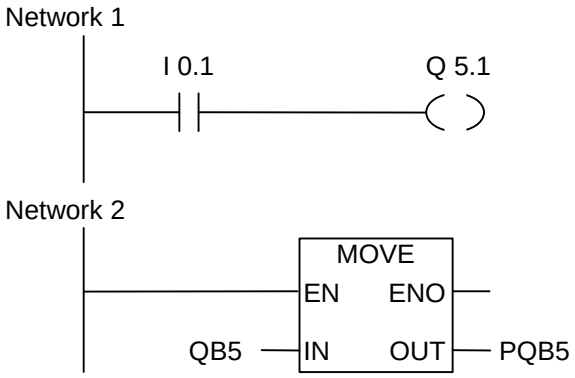
- 由于Q存储器的整个字节都写入了输出模块，因此在执行立即输出时，将更新该字节中的所有输出位。
- 如果输出位在程序各处产生了多个中间状态(1/0)，而这些状态不应发送给输出模块，则执行“立即写入”可能会导致危险情况(输出端产生瞬态脉冲)发生。
- 作为常规设计原则，在程序中只能以线圈形式对外部输出模块引用一次。如果用户遵循此设计原则，则可以避免使用立即输出时的大多数潜在问题。

实例

立即写入外设数字输出模块5通道1的等价梯形图程序段。

可以修改寻址输出Q字节(QB5)的状态位，也可以将其保持不变。程序段1中给Q5.1分配I0.1信号状态。将QB5复制到相应的直接外设输出存储区(PQB5)。

字PIW1包含I1.1的立即状态。对PIW1与W#16#0002执行AND运算。如果PB1中的I1.1(第二位)为真(“1”)，则结果不等于零。如果WAND_W指令的结果不等于零，触点A<>0时将传递电压。



在此实例中，Q5.1为所需的立即输出位。

字节PQB5包含Q5.1位的立即输出状态。

MOVE(复制)指令还会更新PQB5的其它7位。

2 比较指令

2.1 比较指令概述

说明

根据用户选择的比较类型比较IN1和IN2：

== IN1等于IN2
<> IN1不等于IN2
> IN1大于IN2
< IN1小于IN2
>= IN1大于或等于IN2
<= IN1小于或等于IN2

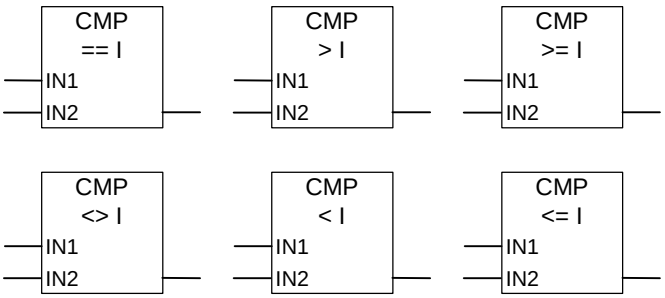
如果比较结果为“真”，则函数的RLO为“1”。如果以串联方式使用比较单元，则使用“与”运算将其链接至梯级程序段的RLO；如果以并联方式使用该框，则使用“或”运算将其链接至梯级程序段的RLO。

以下是可供使用的比较指令：

- CMP ?I 整数比较
- CMP ?D 比较双精度整数
- CMP ?R 比较实数

2.2 CMP ?I 比较整数

符号



参数	数据类型	内存区域	说明
输入框	BOOL	I、Q、M、L、D	上一逻辑运算的结果
输出框	BOOL	I、Q、M、L、D	比较的结果，仅在输入框的RLO = 1时才进一步处理
IN1	INT	I、Q、M、L、D 或常数	要比较的第一个值
IN2	INT	I、Q、M、L、D 或常数	要比较的第二个值

说明

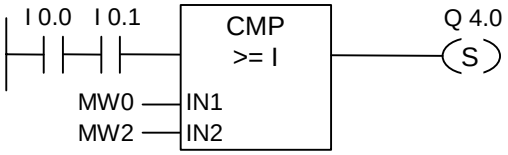
CMP ?I(整数比较)的使用方法与标准触点类似。它可位于任何可放置标准触点的位置。可根据用户选择的比较类型比较IN1和IN2。

如果比较结果为“真”，则函数的RLO为“1”。如果以串联方式使用该框，则使用“与”运算将其链接至整个梯级程序段的RLO；如果以并联方式使用该框，则使用“或”运算将其链接至整个梯级程序段的RLO。

状态字

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
写：	x	x	x	0	-	0	x	x	1

实例

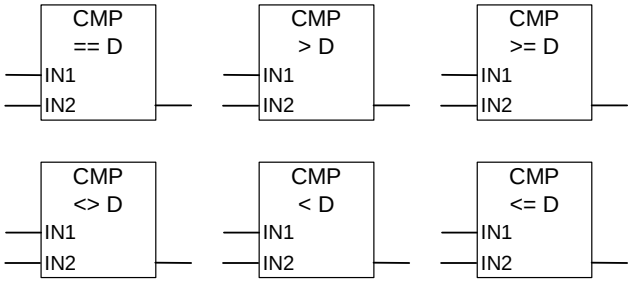


如果满足下列条件，则输出Q4.0置位：

- 输入I0.0和I0.1的信号状态为“1”
- 并且MW0 >= MW2

2.3 CMP ?D 比较双精度整数

符号



参数	数据类型	内存区域	说明
输入框	BOOL	I、Q、M、L、D	上一逻辑运算的结果
输出框	BOOL	I、Q、M、L、D	比较的结果，仅在输入框的 RLO = 1时才进一步处理
IN1	DINT	I、Q、M、L、D 或常数	要比较的第一个值
IN2	DINT	I、Q、M、L、D 或常数	要比较的第二个值

说明

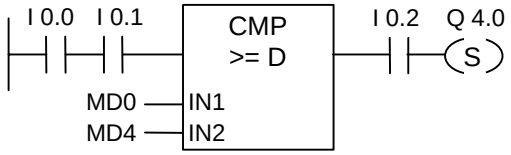
CMP ?D(比较双精度整数)的使用方法与标准触点类似。它可位于任何可放置标准触点的位置。可根据用户选择的比较类型比较IN1和IN2。

如果比较结果为“真”，则函数的RLO为“1”。如果以串联方式使用比较单元，则使用“与”运算将其链接至梯级程序段的RLO；如果以并联方式使用该框，则使用“或”运算将其链接至梯级程序段的RLO。

状态字

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
写：	x	x	x	0	-	0	x	x	1

实例

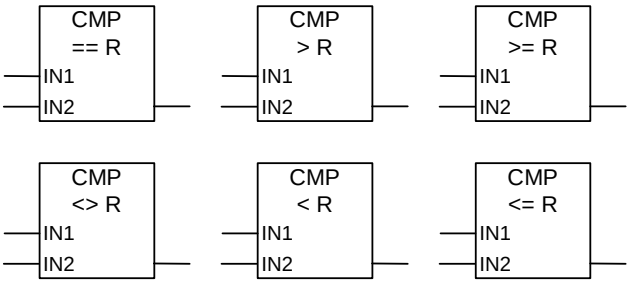


如果满足下列条件，则输出Q4.0置位：

- 输入I0.0和I0.1的信号状态为“1”
- 并且MD0 >= MD4
- 同时输入I0.2的信号状态为“1”

2.4 CMP ?R 比较实数

符号



参数	数据类型	内存区域	说明
输入框	BOOL	I、Q、M、L、D	上一逻辑运算的结果
输出框	BOOL	I、Q、M、L、D	比较的结果，仅在输入框的RLO = 1时才进一步处理
IN1	REAL	I、Q、M、L、D 或常数	要比较的第一个值
IN2	REAL	I、Q、M、L、D 或常数	要比较的第二个值

说明

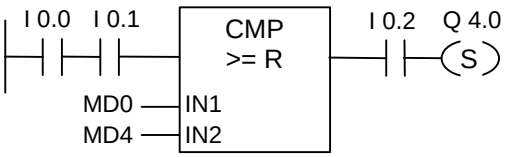
CMP ?R(整数比较)的使用方法类似标准触点。它可位于任何可放置标准触点的位置。可根据用户选择的比较类型比较IN1和IN2。

如果比较结果为“真”，则函数的RLO为“1”。如果以串联方式使用该框，则使用“与”运算将其链接至整个梯级程序段的RLO；如果以并联方式使用该框，则使用“或”运算将其链接至整个梯级程序段的RLO。

状态字

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
写：	x	x	x	x	x	0	x	x	1

实例



如果满足下列条件，则输出Q4.0置位：

- 输入I0.0和I0.1的信号状态为“1”
- 并且MD0 >= MD4
- 同时输入I0.2的信号状态为“1”

3 转换指令

3.1 转换指令概述

说明

转换指令读取参数IN的内容，然后进行转换或改变其符号。可通过参数OUT查询结果。

用户可使用下列转换指令：

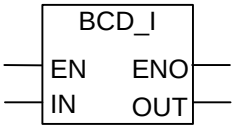
- BCD_I BCD码转换为整数
- I_BCD 整型转换为BCD码
- BCD_DI BCD码转换为双精度整数
- I_DINT 整型转换为长整型
- DI_BCD 长整型转换为BCD码
- DI_REAL 长整型转换为浮点型

- INV_I 二进制反码整型
- INV_DI 二进制反码长整型
- NEG_I 二进制补码整型
- NEG_DI 二进制补码长整型

- NEG_R 浮点数取反
- ROUND 取整为长整型
- TRUNC 截断长整型部分
- CEIL 上限
- FLOOR 向下取整

3.2 BCD_I BCD码转换为整数

符号



参数	数据类型	内存区域	说明
EN	BOOL	I、Q、M、L、D	使能输出
ENO	BOOL	I、Q、M、L、D	使能输出
IN	WORD	I、Q、M、L、D	BCD码数字
OUT	INT	I、Q、M、L、D	BCD码数字的整型值

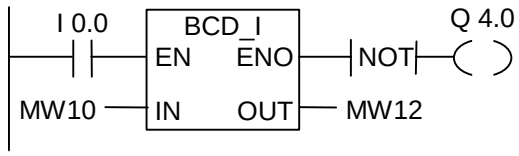
说明

BCD_I(BCD码转换为整数)将参数IN的内容以三位BCD码数字(+/- 999)读取，并将其转换为整型值(16位)。整型值的结果通过参数OUT输出。ENO始终与EN的信号状态相同。

状态字

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
写:	1	-	-	-	-	0	1	1	1

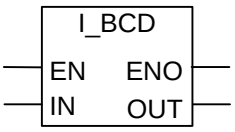
实例



如果输入I0.0的状态为“1”，则将MW10中的内容以三位BCD码数字读取，并将其转换为整型值。结果存储在MW12中。如果未执行转换(ENO = EN = 0)，则输出Q4.0的状态为“1”。

3.3 I_BCD 整型转换为BCD码

符号



参数	数据类型	内存区域	说明
EN	BOOL	I、Q、M、L、D	使能输出
ENO	BOOL	I、Q、M、L、D	使能输出
IN	INT	I、Q、M、L、D	整数
OUT	WORD	I、Q、M、L、D	整数的BCD码值

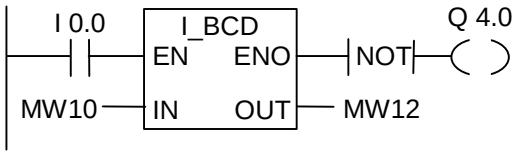
说明

I_BCD(整型转换为BCD码)将参数IN的内容以整型值(16位)读取，并将其转换为三位BCD码数字(+/- 999)。结果由参数OUT输出。如果产生溢出，ENO的状态为“0”。

状态字

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
写:	x	-	-	x	x	0	x	x	1

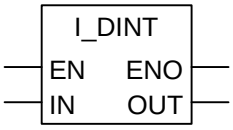
实例



如果I0.0的状态为“1”，则将MW10的内容以整型值读取，并将其转换为三位BCD码数字。结果存储在MW12中。如果产生溢出或未执行指令(I0.0 = 0)，则输出Q4.0的状态为“1”。

3.4 I_DINT 整型转换为长整型

符号



参数	数据类型	内存区域	说明
EN	BOOL	I、Q、M、L、D	使能输出
ENO	BOOL	I、Q、M、L、D	使能输出
IN	INT	I、Q、M、L、D	要转换的整型值
OUT	DINT	I、Q、M、L、D	长整型结果

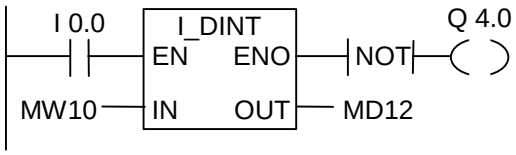
说明

I_DINT(整型转换为长整型)将参数IN的内容以整型(16位)读取，并将其转换为长整型(32位)。结果由参数OUT输出。ENO始终与EN的信号状态相同。

状态字

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
写:	1	-	-	-	-	0	1	1	1

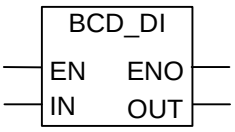
实例



如果I0.0为“1”，则MW10的内容以整型读取，并将其转换为长整型。结果存储在MD12中。如果未执行转换(ENO = EN = 0)，则输出Q4.0的状态为“1”。

3.5 BCD_DI BCD码转换为双精度整数

符号



参数	数据类型	内存区域	说明
EN	BOOL	I、Q、M、L、D	使能输出
ENO	BOOL	I、Q、M、L、D	使能输出
IN	DWORD	I、Q、M、L、D	BCD码数字
OUT	DINT	I、Q、M、L、D	BCD码数字的长整型值

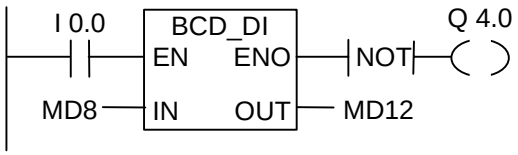
说明

BCD_DI(将BCD码转换为双精度整数)将参数IN的内容以七位BCD码(+/- 9999999)数字读取，并将其转换为长整型值(32位)。长整型值的结果通过参数OUT输出。ENO始终与EN的信号状态相同。

状态字

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
写:	1	-	-	-	-	0	1	1	1

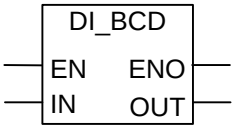
实例



如果I0.0的状态为“1”，则将MD8的内容以七位BCD码数字读取，并将其转换为长整型值。结果存储在MD12中。如果未执行转换(ENO = EN = 0)，则输出Q4.0的状态为“1”。

3.6 DI_BCD 长整型转换为BCD码

符号



参数	数据类型	内存区域	说明
EN	BOOL	I、Q、M、L、D	使能输出
ENO	BOOL	I、Q、M、L、D	使能输出
IN	DINT	I、Q、M、L、D	长整数
OUT	DWORD	I、Q、M、L、D	长整数的BCD码值

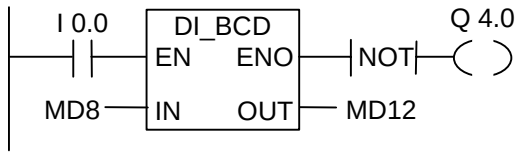
说明

DI_BCD(长整型转换为BCD码)将参数IN的内容以长整型值(32位)读取，并将其转换为七位BCD码数字(+/- 9999999)。结果由参数OUT输出。如果产生溢出，ENO的状态为“0”。

状态字

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
写：	x	-	-	x	x	0	x	x	1

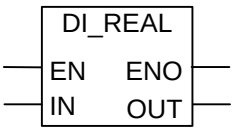
实例



如果I0.0的状态为“1”，则将MD8的内容以长整型读取，并将其转换为七位BCD码数字。结果存储在MD12中。如果产生溢出或未执行指令(I0.0 = 0)，则输出Q4.0的状态为“1”。

3.7 DI_REAL 长整型转换为浮点型

符号



参数	数据类型	内存区域	说明
EN	BOOL	I、Q、M、L、D	使能输出
ENO	BOOL	I、Q、M、L、D	使能输出
IN	DINT	I、Q、M、L、D	要转换的长整型值
OUT	REAL	I、Q、M、L、D	浮点数结果

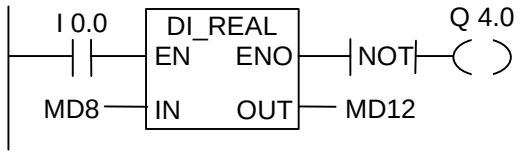
说明

DI_REAL(长整型转换为浮点型)将参数IN的内容以长整型读取，并将其转换为浮点数。结果由参数OUT输出。ENO始终与EN的信号状态相同。

状态字

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
写:	1	-	-	-	-	0	1	1	1

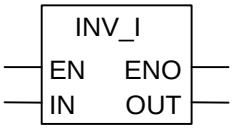
实例



如果I0.0的状态为“1”，则将MD8中的内容以长整型读取，并将其转换为浮点数。结果存储在MD12中。如果未执行转换(ENO = EN = 0)，则输出Q4.0的状态为“1”。

3.8 INV_I 对整数求反码

符号



参数	数据类型	内存区域	说明
EN	BOOL	I、Q、M、L、D	使能输出
ENO	BOOL	I、Q、M、L、D	使能输出
IN	INT	I、Q、M、L、D	整型输入值
OUT	INT	I、Q、M、L、D	整型IN的二进制反码

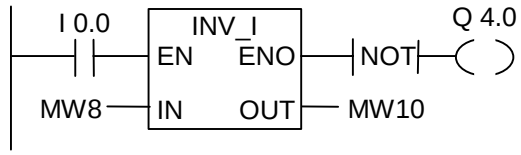
说明

INV_I(对整数求反码)读取IN参数的内容，并使用十六进制掩码W#16#FFFF执行布尔“异或”运算。此指令将每一位变成相反状态。ENO始终与EN的信号状态相同。

状态字

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
写：	1	-	-	-	-	0	1	1	1

实例



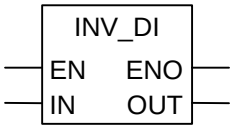
如果I0.0为“1”，则将MW8的每一位都取反，例如：

MW8 = 01000001 10000001取反结果为MW10 = 10111110 01111110。

如果未执行转换(ENO = EN = 0)，则输出Q4.0的状态为“1”。

3.9 INV_DI 对长整数求反码

符号



参数	数据类型	内存区域	说明
EN	BOOL	I、Q、M、L、D	使能输出
ENO	BOOL	I、Q、M、L、D	使能输出
IN	DINT	I、Q、M、L、D	长整型输入值
OUT	DINT	I、Q、M、L、D	长整型IN的二进制反码

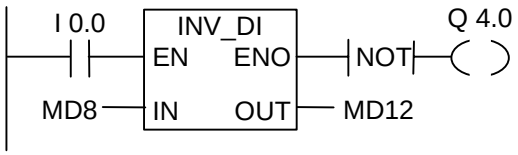
说明

INV_DI(对长整数求反码)读取IN参数的内容，并使用十六进制掩码W#16#FFFF FFFF执行布尔“异或”运算。此指令将每一位转换为相反状态。ENO始终与EN的信号状态相同。

状态字

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
写:	1	-	-	-	-	0	1	1	1

实例



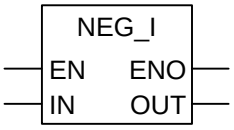
如果I0.0为“1”，则MD8的每一位都取反，例如：

MD8 = F0FF FFF0取反结果为MD12 = 0F00 000F。

如果未执行转换(ENO = EN = 0)，则输出Q4.0的状态为“1”。

3.10 NEG_I 对整数求补码

符号



参数	数据类型	内存区域	说明
EN	BOOL	I、Q、M、L、D	使能输出
ENO	BOOL	I、Q、M、L、D	使能输出
IN	INT	I、Q、M、L、D	整型输入值
OUT	INT	I、Q、M、L、D	整型IN的二进制补码

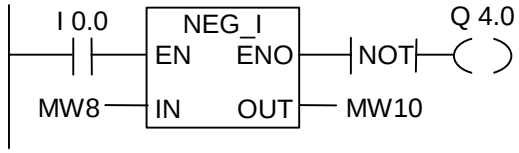
说明

NEG_I(对整数求补码)读取IN参数的内容并执行求二进制补码指令。二进制补码指令等同于乘以(-1)后改变符号(例如：从正值变为负值)。ENO始终与EN的信号状态相同，以下情况例外：如果EN的信号状态 = 1并产生溢出，则ENO的信号状态 = 0。

状态字

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
写：	x	x	x	x	x	0	x	x	1

实例



如果I0.0为“1”，则由OUT参数将MW8的值(符号相反)输出到MW10。

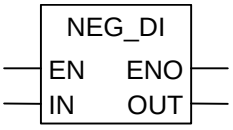
MW8 = + 10结果为MW10 = - 10。

如果未执行转换(ENO = EN = 0)，则输出Q4.0的状态为“1”。

如果EN的信号状态 = 1并产生溢出，则ENO的信号状态 = 0。

3.11 NEG_DI 对长整数求补码

符号



参数	数据类型	内存区域	说明
EN	BOOL	I、Q、M、L、D	使能输出
ENO	BOOL	I、Q、M、L、D	使能输出
IN	DINT	I、Q、M、L、D	长整型输入值
OUT	DINT	I、Q、M、L、D	IN值的二进制补码

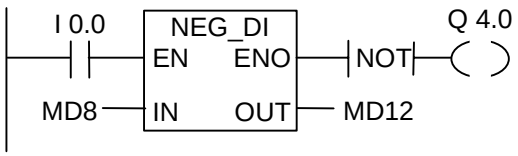
说明

NEG_DI(对长整数求补码)读取参数IN的内容并执行二进制补码指令。二进制补码指令等同于乘以(-1)后改变符号(例如：从正值变为负值)。ENO始终与EN的信号状态相同，以下情况例外：如果EN的信号状态 = 1并产生溢出，则ENO的信号状态 = 0。

状态字

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
写：	x	x	x	x	x	0	x	x	1

实例



如果I0.0为“1”，则由OUT参数将MD8的值(符号相反)输出到MD12。

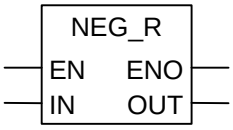
MD8 = + 1000结果为MD12 = - 1000。

如果未执行转换(ENO = EN = 0)，则输出Q4.0的状态为“1”。

如果EN的信号状态 = 1并产生溢出，则ENO的信号状态 = 0。

3.12 NEG_R 浮点数取反

符号



参数	数据类型	内存区域	说明
EN	BOOL	I、Q、M、L、D	使能输出
ENO	BOOL	I、Q、M、L、D	使能输出
IN	REAL	I、Q、M、L、D	浮点数输入值
OUT	REAL	I、Q、M、L、D	浮点数IN，带负号

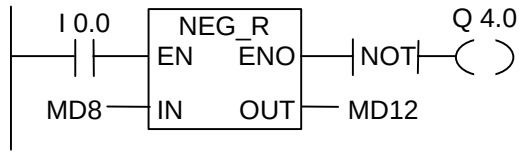
说明

NEG_R(取反浮点)读取参数IN的内容并改变符号。指令等同于乘以(-1)后改变符号(例如：从正值变为负值)。ENO始终与EN的信号状态相同。

状态字

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
写：	x	-	-	-	-	0	x	x	1

实例



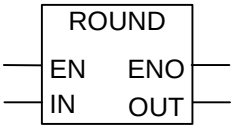
如果I0.0为“1”，则由OUT参数将MD8的值(符号相反)输出到MD12。

MD8 = + 6.234结果为MD12 = - 6.234。

如果未执行转换(ENO = EN = 0)，则输出Q4.0的状态为“1”。

3.13 ROUND 取整为长整型

符号



参数	数据类型	内存区域	说明
EN	BOOL	I、Q、M、L、D	使能输出
ENO	BOOL	I、Q、M、L、D	使能输出
IN	REAL	I、Q、M、L、D	要取整的值
OUT	DINT	I、Q、M、L、D	将IN取整至最接近的整数

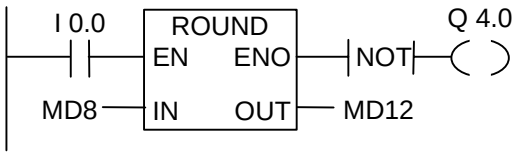
说明

ROUND(取整为长整型)将参数IN的内容以浮点数读取，并将其转换为长整型(32位)。结果为最接近的整数(“取整到最接近值”)。如果浮点数介于两个整数之间，则返回偶数。结果由参数OUT输出。如果产生溢出，ENO的状态为“0”。

状态字

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
写：	x	-	-	x	x	0	x	x	1

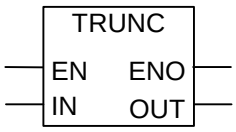
实例



如果I0.0的状态为“1”，则将MD8中的内容以浮点数读取，并将其转换为最接近的长整数。函数“取整为最接近值”的结果存储在MD12中。如果产生溢出或未执行指令(I0.0 = 0)，则输出Q4.0的状态为“1”。

3.14 TRUNC 截取长整数部分

符号



参数	数据类型	内存区域	说明
EN	BOOL	I、Q、M、L、D	使能输出
ENO	BOOL	I、Q、M、L、D	使能输出
IN	REAL	I、Q、M、L、D	要转换的浮点值
OUT	DINT	I、Q、M、L、D	IN值的所有数字部分

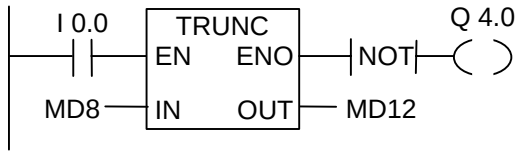
说明

TRUNC(截断长整型)将参数IN的内容以浮点数读取，并将其转换为长整型(32位)。("向零取整模式")的长整型结果由参数OUT输出。如果产生溢出，ENO的状态为"0"。

状态字

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
写:	x	-	-	x	x	0	x	x	1

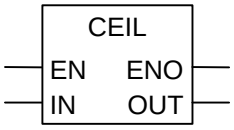
实例



如果I0.0的状态为"1"，则将MD8中的内容以实型数字读取，并将其转换为长整型值。结果为浮点数的整型部分，并存储在MD12中。如果产生溢出或未执行指令(I0.0 = 0)，则输出Q4.0的状态为"1"。

3.15 CEIL 向上取整

符号



参数	数据类型	内存区域	说明
EN	BOOL	I、Q、M、L、D	使能输出
ENO	BOOL	I、Q、M、L、D	使能输出
IN	REAL	I、Q、M、L、D	要转换的浮点值
OUT	DINT	I、Q、M、L、D	大于长整型的最小值

说明

CEIL(向上取整)将参数IN的内容以浮点数读取，并将其转换为长整型(32位)。结果为大于该浮点数的最小整数(“取整到 + 无穷大”)。如果产生溢出，ENO的状态为“0”。

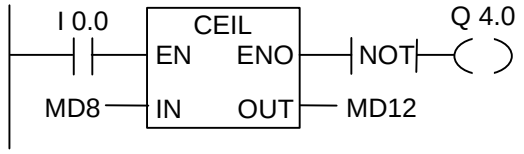
状态字

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
写*:	X	-	-	X	X	0	X	X	1
写**:	0	-	-	-	-	0	0	0	1

* 函数执行(EN = 1)

** 函数不执行(EN = 0)

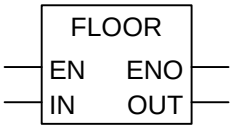
实例



如果I0.0为1，则将MD8的内容作为浮点数读取，并使用**取整**函数将其转换为长整型。结果存储在MD12中。如果出现溢出或未处理指令(I0.0 = 0)，则输出Q4.0的状态为“1”。

3.16 FLOOR 下取整

符号



参数	数据类型	内存区域	说明
EN	BOOL	I、Q、M、L、D	使能输出
ENO	BOOL	I、Q、M、L、D	使能输出
IN	REAL	I、Q、M、L、D	要转换的浮点值
OUT	DINT	I、Q、M、L、D	小于长整型的最大值

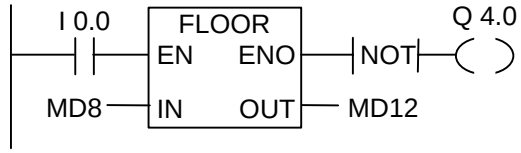
说明

FLOOR(下取整)将参数IN的内容以浮点数读取，并将其转换为长整型(32位)。结果为小于该浮点数的最大整数部分(“取整为 - 无穷大”)。如果产生溢出，ENO的状态为“0”。

状态字

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
写:	x	-	-	x	x	0	x	x	1

实例



如果I0.0为“1”，则将MD8的内容作为浮点数读取，并按取整到负无穷大模式将其转换为长整型。结果存储在MD12中。如果产生溢出或未执行指令(I0.0 = 0)，则输出Q4.0的状态为“1”。

4 计数器指令

4.1 计数器指令概述

存储器中的区域

在用户CPU的存储器中，有为计数器保留的存储区。此存储区为每个计数器地址保留一个16位字。梯形图指令集支持256个计数器。

计数器指令是仅有的可访问计数器存储区的函数。

计数值

计数器字中的0至9位包含二进制代码形式的计数值。当设置某个计数器时，计数值移至计数器字。计数值的范围为0至999。

用户可使用下列计数器指令在此范围内改变计数值：

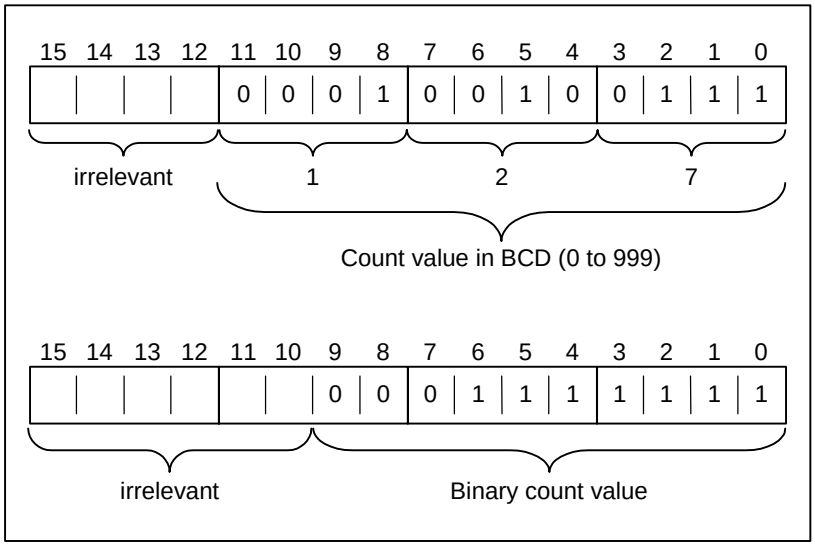
- S_CUD 双向计数器
- S_CD 降值计数器
- S_CU 升值计数器
- ---(SC) 设置计数器线圈
- ---(CU) 升值计数器线圈
- ---(CD) 降值计数器线圈

计数器中的位组态

输入从0至999的数字，用户可为计数器提供预设值，例如，使用下列格式输入127：C#127。其中C#代表二进制编码十进制格式(BCD格式：四位一组，包含一个用二进制编码的十进制值)。

计数器中的0至11位包含二进制编码十进制格式的计数值。

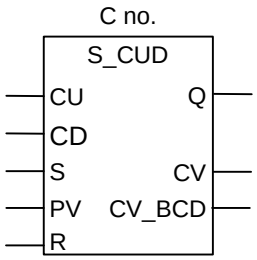
下图显示了加载计数值127之后计数器的内容，以及设置计数器之后计数器单元中的内容。



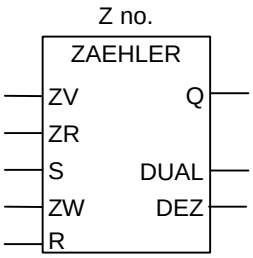
4.2 S_CUD 双向计数器

符号

English



German



参数 英语	参数 德语	数据类型	内存区域	说明
C编号	Z编号	COUNTER	C	计数器标识号，其范围依赖于CPU
CU	ZV	BOOL	I、Q、M、L、D	升值计数输入
CD	ZR	BOOL	I、Q、M、L、D	递减计数输入
S	S	BOOL	I、Q、M、L、D	为预设计数器设置输入
PV	ZW	WORD	I、Q、M、L、D 或常数	将计数器值以“C#<值>”的格式输入(范围0至999)
PV	ZW	WORD	I、Q、M、L、D	预置计数器的值
R	R	BOOL	I、Q、M、L、D	复位输入
CV	DUAL	WORD	I、Q、M、L、D	当前计数器值，十六进制数字
CV_BCD	DEZ	WORD	I、Q、M、L、D	当前计数器值，BCD码
Q	Q	BOOL	I、Q、M、L、D	计数器状态

说明

如果输入S有上升沿，**S_CUD**(双向计数器)预置为输入PV的值。如果输入R为1，则计数器复位，并将计数值设置为零。如果输入CU的信号状态从“0”切换为“1”，并且计数器的值小于“999”，则计数器的值增1。如果输入CD有上升沿，并且计数器的值大于“0”，则计数器的值减1。

如果两个计数输入都有上升沿，则执行两个指令，并且计数值保持不变。

如果已设置计数器，并且输入CU/CD的RLO = 1，则即使没有从上升沿到下降沿或下降沿到上升沿的切换，计数器也会在下一个扫描周期进行相应的计数。

如果计数值大于等于零(“0”)，则输出Q的信号状态为“1”。

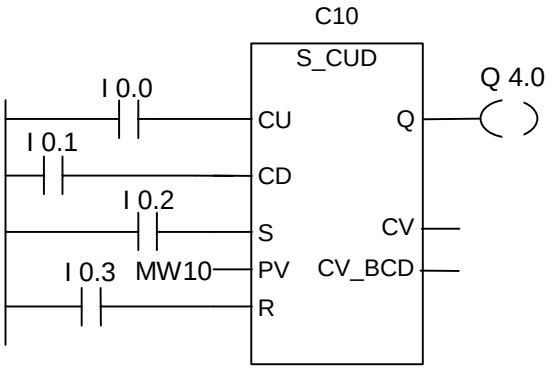
状态字

	BR	**CC 1**	**CC 0**	**OV**	**OS**	**OR**	**STA**	**RLO**	**/FC**
写：	-	-	-	-	-	x	x	x	1

注意

避免在多个程序点使用同一计数器(可能出现计数出错)。

实例

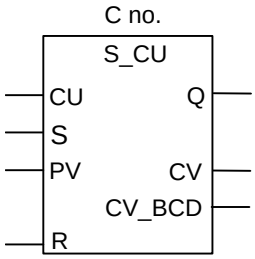


如果I0.2从“0”变为“1”，则计数器预设为MW10的值。如果I0.0的信号状态从“0”改变为“1”，则计数器C10的值将增加1 - 当C10的值等于“999”时除外。如果I0.1从“0”改变为“1”，则C10减少1 - 但当C10的值为“0”时除外。如果C10不等于零，则Q4.0为“1”。

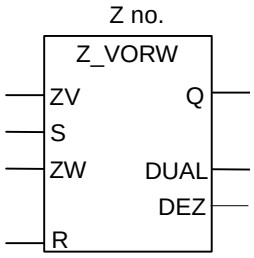
4.3 S_CU 升值计数器

符号

English



German



参数 英语	参数 德语	数据类型	内存区域	说明
C编号	Z编号	COUNTER	C	计数器标识号，其范围依赖于CPU
CU	ZV	BOOL	I、Q、M、L、D	升值计数输入
S	S	BOOL	I、Q、M、L、D	为预设计数器设置输入
PV	ZW	WORD	I、Q、M、L、D或常数	将计数器值以“C#<值>”的格式输入(范围0至999)
PV	ZW	WORD	I、Q、M、L、D	预置计数器的值
R	R	BOOL	I、Q、M、L、D	复位输入
CV	DUAL	WORD	I、Q、M、L、D	当前计数器值，十六进制数字
CV_BCD	DEZ	WORD	I、Q、M、L、D	当前计数器值，BCD码
Q	Q	BOOL	I、Q、M、L、D	计数器状态

说明

如果输入S有上升沿，则**S_CU**(升值计数器)预置为输入PV的值。

如果输入R为“1”，则计数器复位，并将计数值设置为零。

如果输入CU的信号状态从“0”切换为“1”，并且计数器的值小于“999”，则计数器的值增1。

如果已设置计数器，并且输入CU的RLO = 1，则即使没有从上升沿到下降沿或下降沿到上升沿的切换，计数器也会在下一个扫描周期进行相应的计数。

如果计数值大于等于零(“0”)，则输出Q的信号状态为“1”。

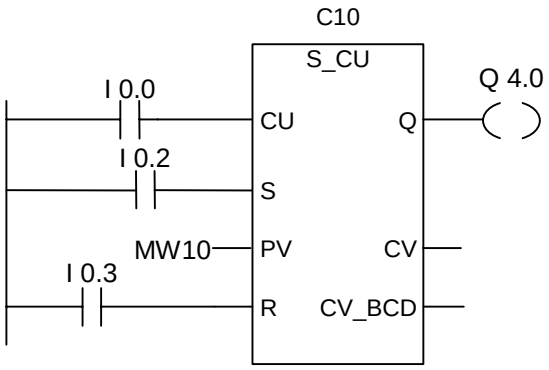
状态字

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
写:	-	-	-	-	-	x	x	x	1

注意

避免在多个程序点使用同一计数器(可能出现计数出错)。

实例

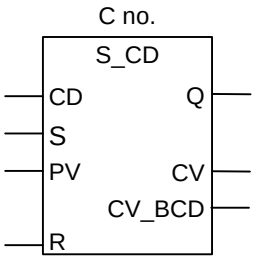


如果I0.2从“0”变为“1”，则计数器预设为MW10的值。如果I0.0的信号状态从“0”改变为“1”，则计数器C10的值将增加1 - 当C10的值等于“999”时除外。如果C10不等于零，则Q4.0为“1”。

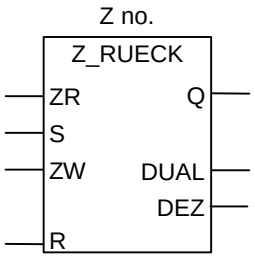
4.4 S_CD 降值计数器

符号

English



German



参数 英语	参数 德语	数据类型	内存区域	说明
C编号	Z编号	COUNTER	C	计数器标识号，其范围依赖于CPU
CD	ZR	BOOL	I、Q、M、L、D	递减计数输入
S	S	BOOL	I、Q、M、L、D	为预设计数器设置输入
PV	ZW	WORD	I、Q、M、L、D或常数	将计数器值以“C#<值>”的格式输入(范围0至999)
PV	ZW	WORD	I、Q、M、L、D	预置计数器的值
R	R	BOOL	I、Q、M、L、D	复位输入
CV	DUAL	WORD	I、Q、M、L、D	当前计数器值，十六进制数字
CV_BCD	DEZ	WORD	I、Q、M、L、D	当前计数器值，BCD码
Q	Q	BOOL	I、Q、M、L、D	状态计数器

说明

如果输入S有上升沿，则**S_CD**(降值计数器)设置为输入PV的值。

如果输入R为1，则计数器复位，并将计数值设置为零。

如果输入CD的信号状态从“0”切换为“1”，并且计数器的值大于零，则计数器的值减1。

如果已设置计数器，并且输入CD的RLO = 1，则即使没有从上升沿到下降沿或下降沿到上升沿的改变，计数器也会在下一个扫描周期进行相应的计数。

如果计数值大于等于零(“0”)，则输出Q的信号状态为“1”。

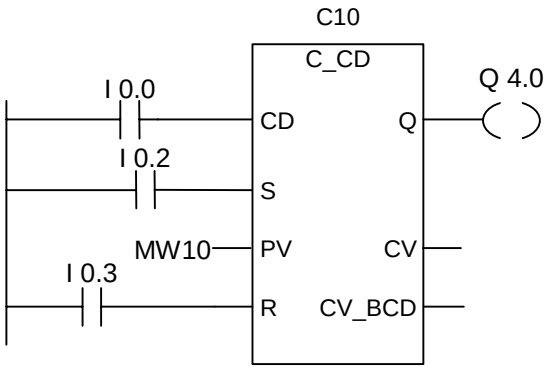
状态字

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
写:	-	-	-	-	-	x	x	x	1

注意

避免在多个程序点使用同一计数器(可能出现计数出错)。

实例



如果I0.2从“0”变为“1”，则计数器预设为MW10的值。如果I0.0的信号状态从“0”改变为“1”，则计数器C10的值将减1 - 当C10的值等于“0”时除外。如果C10不等于零，则Q4.0为“1”。

4.5 ---(SC) 设置计数器值

符号

英语	德语
<C编号>	<Z编号>
---(SC)	**---(SZ)**
<预设值>	<预设值>

参数 英语	参数 德语	数据类型	内存区域	说明
<C编号>	<Z编号>	COUNTER	C	要预置的计数器编号
<预设值>	<预设值>	WORD	I、Q、M、L、D 或常数	预置BCD的值(0至999)

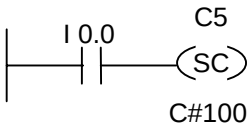
说明

仅在RLO中有上升沿时，**---(SC)(**设置计数器值)才会执行。此时，预设值被传送至指定的计数器。

状态字

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
写:	x	-	-	-	-	0	x	-	0

实例



如在I0.0有上升沿(从“0”改变为“1”)，则计数器C5预置为100。如果没有上升沿，则计数器C5的值保持不变。

4.6 ---(CU) 升值计数器线圈

符号

英语	德语
<C编号>	<Z编号>
---(CU)	**---(ZV)**

参数 英语	参数 德语	数据类型	内存区域	说明
<C编号>	<Z编号>	COUNTER	C	计数器标识号，其范围依赖于CPU

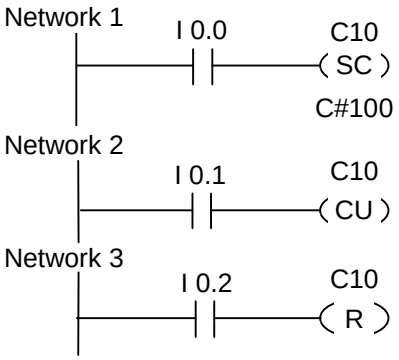
说明

如在RLO中有上升沿，并且计数器的值小于“999”，则**---(CU)**(升值计数器线圈)将指定计数器的值加1。如果RLO中没有上升沿，或者计数器的值已经是“999”，则计数器值不变。

状态字

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
写：	-	-	-	-	-	0	-	-	0

实例



如果输入I0.0的信号状态从“0”改变为“1”(RLO中有上升沿)，则将预设值100载入计数器C10。

如果输入I0.1的信号状态从“0”改变为“1”(在RLO中有上升沿)，则计数器C10的计数值将增加1，但当C10的值等于“999”时除外。如果RLO中没有上升沿，则C10的值保持不变。

如果I0.2的信号状态为“1”，则计数器C10复位为“0”。

4.7 ---(CD) 降值计数器线圈

符号

英语	德语
<C编号>	<Z编号>
---(CD)	**---(ZD)**

参数 英语	参数 德语	数据类型	内存区域	说明
<C编号>	<Z编号>	COUNTER	C	计数器标识号，其范围依赖于CPU

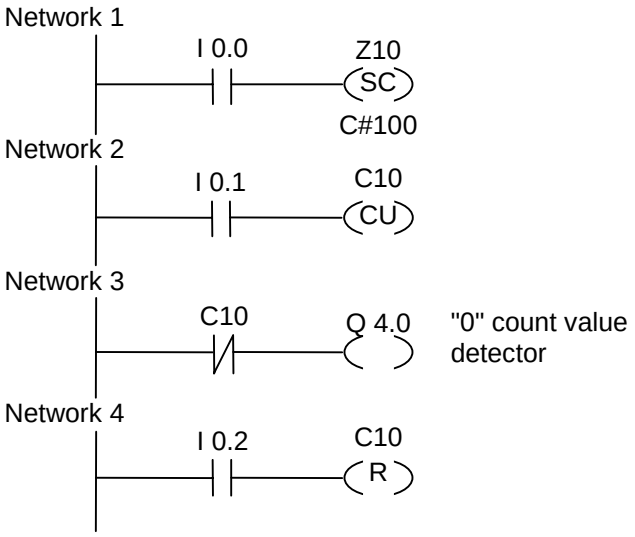
说明

如果RLO状态中有上升沿，并且计数器的值大于“0”，则**---(CD)(**降值计数器线圈)将指定计数器的值减1。如果RLO中没有上升沿，或者计数器的值已经是“0”，则计数器值不变。

状态字

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
写：	-	-	-	-	-	0	-	-	0

实例



如果输入I0.0的信号状态从“0”改变为“1”(RLO中有上升沿)，则将预设值100载入计数器C10。

如果输入I0.1的信号状态从“0”改变为“1”(在RLO中有上升沿)，则计数器C10的计数值将减1，但当C10的值等于“0”时除外。如果RLO中没有上升沿，则C10的值保持不变。

如果计数值 = 0，则接通Q4.0。

如果输入I0.2的信号状态为“1”，则计数器C10复位为“0”。

5 数据块指令

5.1 ---(OPN)打开数据块：DB或DI

符号

<DB编号> 或 <DI编号>

---(OPN)

参数	数据类型	内存区域	说明
<DB编号> <DI编号>	BLOCK_DB	DB、DI	DB/DI编号；编号范围取决于CPU

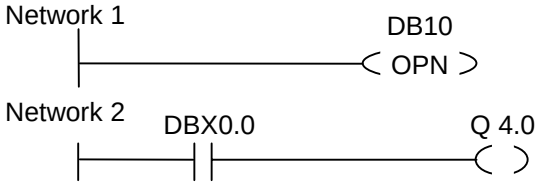
说明

---(OPN)(打开数据块)打开共享数据块(DB)或情景数据块(DI)。**---(OPN)**函数是一种对数据块的无条件调用。将数据块的编号传送到DB或DI寄存器中。后续的DB和DI命令根据寄存器内容访问相应的块。

状态字

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
写:	-	-	-	-	-	-	-	-	-

实例



打开数据块10(DB10)。触点地址(DBX0.0)引用包含在DB10中的当前数据记录的数据字节零的第零位。将此位的信号状态分配给输出Q4.0。

6 逻辑控制指令

6.1 逻辑控制指令概述

说明

可以在所有逻辑块(组织块(OB)、功能块(FB)和功能(FC))中使用逻辑控制指令。

有可以执行下列功能的逻辑控制指令：

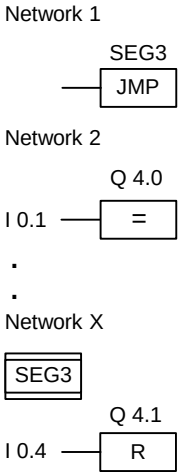
- ---(JMP)--- 无条件跳转
- ---(JMP)--- 有条件跳转
- ---(JMPN)--- 若“否”则跳转

标号作为地址

跳转指令的地址是标号。标号最多可以包含四个字符。第一个字符必须是字母表中的字母，其它字符可以是字母或数字(例如，SEG3)。跳转标号指示程序将要跳转到的目标。

标号作为目标

目标标号必须位于程序段的开头。可以通过从梯形图浏览器中选择LABEL，在程序段的开头输入目标标号。在显示的空框中，键入标号的名称。在框中键入标签的名称。



6.2 ---(JMP)--- 无条件跳转

符号

<标号名称>

---(JMP)

说明

---(JMP)(为1时在块内跳转)当左侧电源轨道与指令间没有其它梯形图元素时执行的是绝对跳转。(请参见示例)。

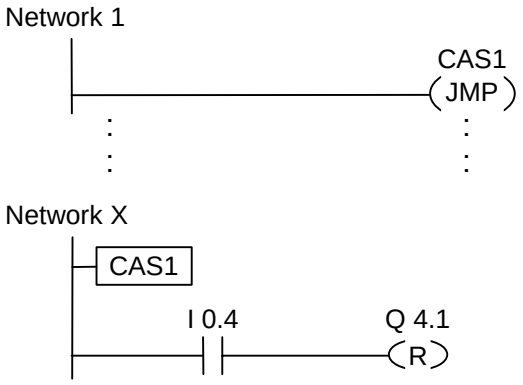
每个 ---(JMP)还都必须有与之对应的目标(LABEL)

跳转指令和标号间的所有指令都不予执行。

状态字

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
写:	-	-	-	-	-	-	-	-	-

实例



始终执行跳转，并忽略跳转指令和跳转标号间的指令。

6.3 ---(JMP)--- 有条件跳转

符号

<标号名称>

---(JMP)

说明

---(JMP)(为1时在块内跳转)当前 一逻辑运算的RLO为“1”时执行的是条件跳转。

每个 ---(JMP)还都必须有与之对应的目标(LABEL)

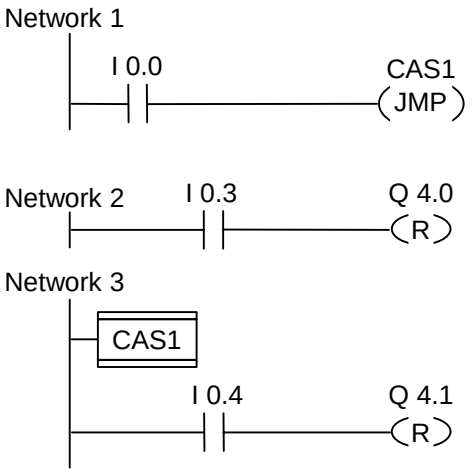
跳转指令和标号间的所有指令都不予执行。

如果未执行条件跳转，RLO将在跳转指令执行后变为“1”。

状态字

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
写：	-	-	-	-	-	0	1	1	0

实例



如果I0.0 = “1”，则执行跳转到标号CAS1。由于此跳转的存在，即使I0.3处有逻辑“1”，也不会执行复位输出Q4.0的指令。

6.4 ---(JMPN) 若“否”则跳转

符号

<标号名称>

---(JMPN)

说明

---(JMPN)(若“否”则跳转)相当于在RLO为“0”时执行的“转到标号”功能。

每个 ---(JMPN)还都必须有与之对应的目标(LABEL).

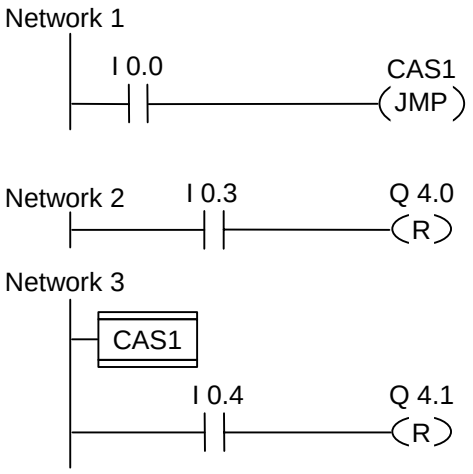
跳转指令和标号间的所有指令都不予执行。

如果未执行条件跳转，RLO将在执行跳转指令后变为“1”。

状态字

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
写:	-	-	-	-	-	0	1	1	0

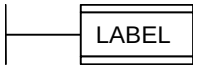
实例



如果I0.0 = “0”，则执行跳转到标号CAS1。由于此跳转的存在，即使I0.3处有逻辑“1”，也不会执行复位输出Q4.0的指令。

6.5 LABEL标号

符号



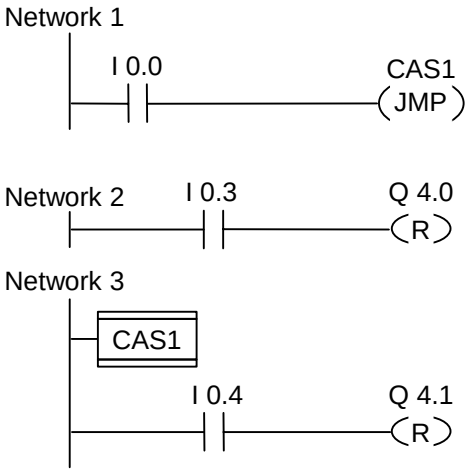
说明

LABEL是跳转指令目标的标识符。

第一个字符必须是字母表中的字母；其它字符可以是字母或数字(例如，CAS1)。

每个 ---(JMP)或 ---(JMPN)都还必须有与之对应的跳转标号 (**LABEL**)。

实例



在I0.0 = “1”,则执行跳转到标号CAS1。由于此跳转的存在，即使I0.3处有逻辑“1”，也不会执行复位输出Q4.0的指令。

7 整型数学运算指令

7.1 整型数学运算指令概述

说明

使用整数运算，您可以对**两个整数**(16和32位)执行以下运算：

- ADD_I 加整数
- SUB_I 减整型
- MUL_I 乘整型
- DIV_I 除整型
- ADD_DI 加双精度整数
- SUB_DI 减长整型
- MUL_DI 乘长整型
- DIV_DI 除长整型
- MOD_DI 返回分数长整型

7.2 使用整数算术指令计算状态字的位

说明

整数运算指令影响状态字中的以下位：CC1和CC0、OV和OS。

下表显示整数(16位和32位)运算指令运算结果的状态字中位的信号状态：

结果的有效范围	CC 1	CC 0	OV	OS
0(零)	0	0	0	*
16位：-32 768 <= 结果 < 0(负数) 32位：-2 147 483 648 <=结果 < 0(负数)	0	1	0	*
16位：32 767 >= 结果 > 0(正数) 32位：2 147 483 647 >= 结果 > 0(正数)	1	0	0	*

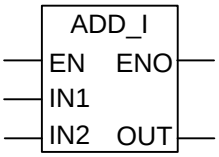
*OS位不受指令运算结果的影响。

结果的范围无效	A1	A0	OV	OS
下溢(加) 16位：结果 = -65536 32位：结果 = -4 294 967 296	0	0	1	1
下溢(乘) 16位：结果 < -32 768(负数) 32位：结果 < -2 147 483 648(负数)	0	1	1	1
溢出(加，减) 16位：结果 > 32 767(正数) 32位：结果 > 2 147 483 647(正数)	0	1	1	1
溢出(乘，除) 16位：结果 > 32 767(正数) 32位：结果 > 2 147 483 647(正数)	1	0	1	1
下溢(加，减) 16位：结果 < -32 768(负数) 32位：结果 < -2 147 483 648(负数)	1	0	1	1
0作除数	1	1	1	1

操作	A1	A0	OV	OS
+D: 结果 -4 294 967 296	0	0	1	1
/D或MOD：除以0	1	1	1	1

7.3 ADD_I 整数加

符号



参数	数据类型	内存区域	说明
EN	BOOL	I、Q、M、L、D	使能输出
ENO	BOOL	I、Q、M、L、D	使能输出
IN1	INT	I、Q、M、L、D 或常数	被加数
IN2	INT	I、Q、M、L、D 或常数	加数
OUT	INT	I、Q、M、L、D	加法结果

说明

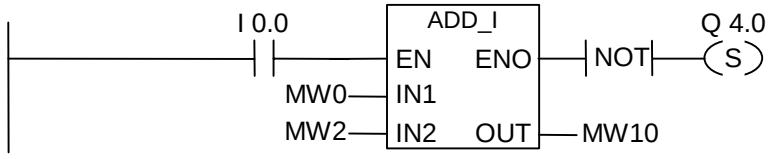
在启用(EN)输入端通过一个逻辑“1”来激活**ADD_I(**整数加)。IN1和IN2相加，结果通过OUT查看。如果该结果超出了整数(16位)允许的范围，OV位和OS位将为“1”并且ENO为逻辑“0”，这样便不执行此数学框后由ENO连接的其它函数(层叠排列)。

参见判断整数运算指令状态字的位。

状态字

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
写:	x	x	x	x	x	0	x	x	1

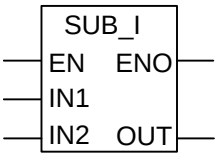
实例



如果I0.0 = “1”，则激活ADD_DI框。MW0 + MW2相加的结果输出到MW10。如果结果超出整数的允许范围，则设置输出Q4.0。

7.4 SUB_I 整数减

符号



参数	数据类型	内存区域	说明
EN	BOOL	I、Q、M、L、D	使能输出
ENO	BOOL	I、Q、M、L、D	使能输出
IN1	INT	I、Q、M、L、D 或常数	被减数
IN2	INT	I、Q、M、L、D 或常数	减数
OUT	INT	I、Q、M、L、D	减法结果

说明

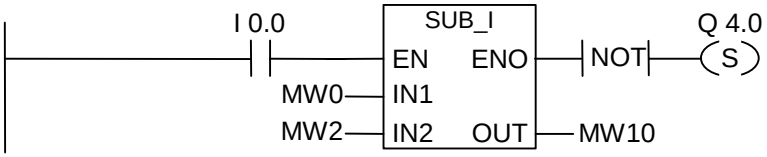
在启用(EN)输入端通过逻辑“1”激活**SUB_I**(整数减)。IN1减去IN2，结果可通过OUT查看。如果该结果超出了整数(16位)允许的范围，OV位和OS位将为“1”并且ENO为逻辑“0”，这样便不执行此数学框后由ENO连接的其它函数(层叠排列)。

参见判断整数运算指令状态字的位

状态字

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
写：	x	x	x	x	x	0	x	x	1

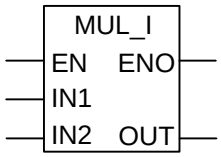
实例



如果I0.0 = “1”，则激活SUB_I框。MW0 - MW2相减的结果输出到MW10。如果结果超出整数允许范围，或者I0.0信号状态 = 0，则设置输出Q4.0。

7.5 MUL_I 整数乘

符号



参数	数据类型	内存区域	说明
EN	BOOL	I、Q、M、L、D	使能输出
ENO	BOOL	I、Q、M、L、D	使能输出
IN1	INT	I、Q、M、L、D 或常数	被乘数
IN2	INT	I、Q、M、L、D 或常数	第二个乘运算值
OUT	INT	I、Q、M、L、D	乘运算结果

说明

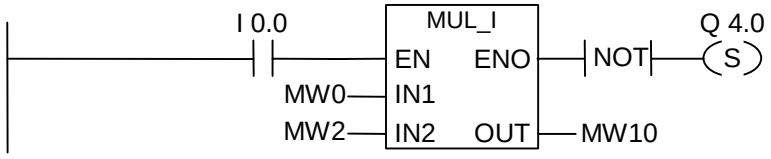
在启用(EN)输入端通过逻辑“1”激活**MUL_I**(整数乘)。IN1和IN2相乘，结果通过OUT查看。如果该结果超出了整数(16位)允许的范围，OV位和OS位将为“1”并且ENO为逻辑“0”，这样便不执行此数学框后由ENO连接的其它函数(层叠排列)。

参见判断整数运算指令状态字的位

状态字

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
写:	x	x	x	x	x	0	x	x	1

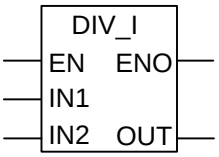
实例



如果I0.0 = “1”，则激活MUL_I框。MW0 x MW2相乘的结果输出到MD10。如果结果超出整数的允许范围，则设置输出Q4.0。

7.6 DIV_I 整数除

符号



参数	数据类型	内存区域	说明
EN	BOOL	I、Q、M、L、D	使能输出
ENO	BOOL	I、Q、M、L、D	使能输出
IN1	INT	I、Q、M、L、D 或常数	被除数
IN2	INT	I、Q、M、L、D 或常数	除数
OUT	INT	I、Q、M、L、D	除法结果

说明

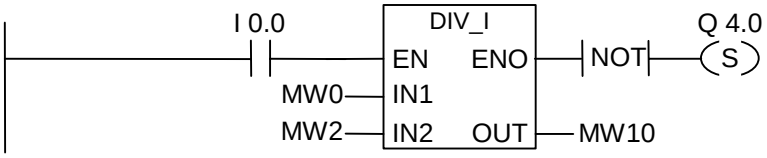
在启用(EN)输入端通过逻辑“1”激活**DIV_I(**整数除)。IN1除以IN2，结果可通过OUT查看。如果该结果超出了整数(16位)允许的范围，OV位和OS位将为“1”并且ENO为逻辑“0”，这样便不执行此数学框后由ENO连接的其它函数(层叠排列)。

参见判断整数运算指令状态字的位

状态字

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
写:	x	x	x	x	x	0	x	x	1

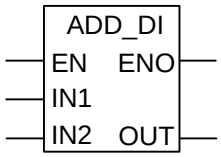
实例



如果I0.0 = “1”，则激活DIV_I框。MW0除以MW2的结果输出到MW10。如果结果超出整数的允许范围，则设置输出Q4.0。

7.7 ADD_DI 长整数加

符号



参数	数据类型	内存区域	说明
EN	BOOL	I、Q、M、L、D	使能输出
ENO	BOOL	I、Q、M、L、D	使能输出
IN1	DINT	I、Q、M、L、D 或常数	被加数
IN2	DINT	I、Q、M、L、D 或常数	加数
OUT	DINT	I、Q、M、L、D	加法结果

说明

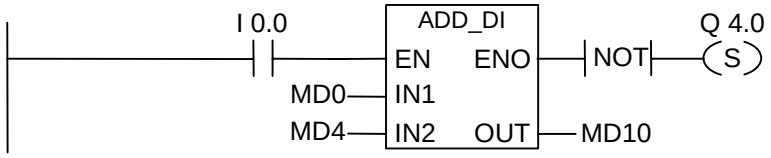
在启用(EN)输入端通过逻辑“1”激活**ADD_DI**(长整数加)。IN1和IN2相加，结果通过OUT查看。如果该结果超出了长整数(32位)允许的范围，OV位和OS位将为“1”并且ENO为逻辑“0”，这样便不执行此数学框后由ENO连接的其它函数(层叠排列)。

参见 判断整数运算指令状态字的位。

状态字

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
写:	x	x	x	x	x	0	x	x	1

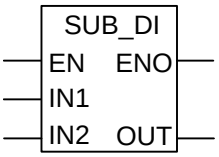
实例



如果I0.0 = “1”，则激活ADD_DI框。.MD0 + MD4相加的结果输出到MD10。如果结果超出长整数的允许范围，则设置输出Q4.0。

7.8 SUB_DI 长整数减

符号



参数	数据类型	内存区域	说明
EN	BOOL	I、Q、M、L、D	使能输出
ENO	BOOL	I、Q、M、L、D	使能输出
IN1	DINT	I、Q、M、L、D 或常数	被减数
IN2	DINT	I、Q、M、L、D 或常数	减数
OUT	DINT	I、Q、M、L、D	减法结果

说明

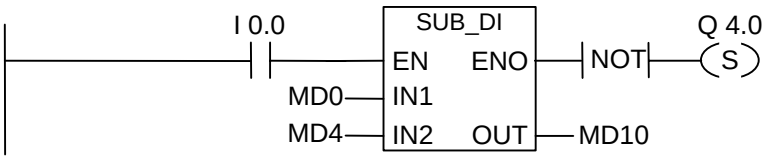
在启用(EN)输入端通过逻辑“1”激活**SUB_DI(**长整数减)。IN1减去IN2，结果可通过OUT查看。如果该结果超出了长整数(32位)允许的范围，OV位和OS位将为“1”并且ENO为逻辑“0”，这样便不执行此数学框后由ENO连接的其它函数(层叠排列)。

参见判断整数运算指令状态字的位。

状态字

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
写：	x	x	x	x	x	0	x	x	1

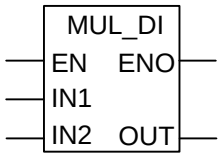
实例



如果I0.0 = “1”，则激活SUB_DI框。.MD0 - MD4相减的结果输出到MD10。如果结果超出长整数的允许范围，则设置输出Q4.0。

7.9 MUL_DI 长整数乘

符号



参数	数据类型	内存区域	说明
EN	BOOL	I、Q、M、L、D	使能输出
ENO	BOOL	I、Q、M、L、D	使能输出
IN1	DINT	I、Q、M、L、D 或常数	被乘数
IN2	DINT	I、Q、M、L、D 或常数	第二个乘运算值
OUT	DINT	I、Q、M、L、D	乘运算结果

说明

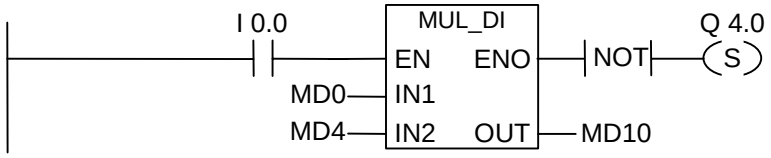
在启用(EN)输入端通过逻辑“1”激活**MUL_DI**(长整数乘)。IN1和IN2相乘，结果通过OUT查看。如果该结果超出了长整数(32位)允许的范围，OV位和OS位将为“1”并且ENO为逻辑“0”，这样便不执行此数学框后由ENO连接的其它函数(层叠排列)。

参见判断整数运算指令状态字的位。

状态字

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
写:	x	x	x	x	x	0	x	x	1

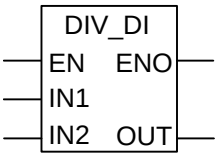
实例



如果I0.0 = “1”，则激活MUL_DI框。MD0 x MD4相乘的结果输出到MD10。如果结果超出长整数的允许范围，则设置输出Q4.0。

7.10 DIV_DI 长整数除

符号



参数	数据类型	内存区域	说明
EN	BOOL	I、Q、M、L、D	使能输出
ENO	BOOL	I、Q、M、L、D	使能输出
IN1	DINT	I、Q、M、L、D 或常数	被除数
IN2	DINT	I、Q、M、L、D 或常数	除数
OUT	DINT	I、Q、M、L、D	除法的整数结果

说明

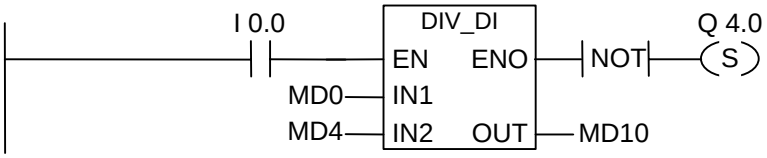
在启用(EN)输入端通过逻辑“1”激活**DIV_DI**(长整数除)。IN1除以IN2，结果可通过OUT查看。长整型除法不产生余数。如果该结果超出了长整数(32位)允许的范围，OV位和OS位将为“1”并且ENO为逻辑“0”，这样便不执行此数学框后由ENO连接的其它函数(层叠排列)。

参见判断整数运算指令状态字的位。

状态字

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
写:	x	x	x	x	x	0	x	x	1

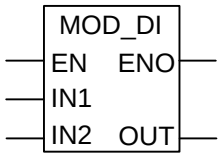
实例



如果I0.0 = “1”，则激活DIV_I框。MD0 : MD4的相除结果输出到MD10。如果结果超出长整数的允许范围，则设置输出Q4.0。

7.11 MOD_DI 返回长整数余数

符号



参数	数据类型	内存区域	说明
EN	BOOL	I、Q、M、L、D	使能输出
ENO	BOOL	I、Q、M、L、D	使能输出
IN1	DINT	I、Q、M、L、D 或常数	被除数
IN2	DINT	I、Q、M、L、D 或常数	除数
OUT	DINT	I、Q、M、L、D	除运算的余数

说明

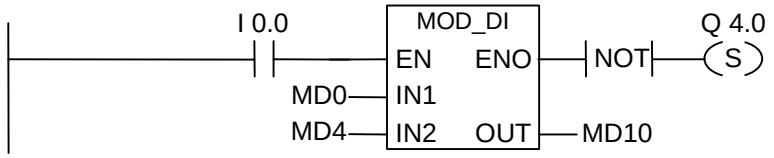
在启用(EN)输入端通过逻辑“1”激活**MOD_DI**(返回长整数余数)。IN1除以IN2，余数可通过OUT查看。如果该结果超出了长整数(32位)允许的范围，OV位和OS位将为“1”并且ENO为逻辑“0”，这样便不执行此数学框后由ENO连接的其它函数(层叠排列)。

参见判断整数运算指令状态字的位。

状态字

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
写:	x	x	x	x	x	0	x	x	1

实例



如果I0.0 = “1”，则激活DIV_DI框。MD0 : MD4相除的余数输出到MD10。如果余数超出长整数的允许范围，则设置输出Q4.0。

8 浮点型数学运算指令

8.1 浮点运算指令概述

说明

IEEE
32位浮点数属于REAL数据类型。可以使用浮点运算指令对**两个32位IEEE浮点数**执行下列运算指令：

- ADD_R 加实数
- SUB_R 实数减
- MUL_R 实数乘
- DIV_R 实数除

使用浮点运算指令，可对**一个 32位IEEE浮点数**执行下列操作：

- 求绝对值(ABS)
- 求平方(SQR)和平方根(SQRT)
- 求自然对数(LN)
- 求指数值(EXP)以e (= 2,71828)为底
- 求下列32位IEEE浮点数表示的角度的三角函数
  - 正弦(SIN)和反正弦(ASIN)
  - 余弦(COS)和反余弦(ACOS)
  - 正切(TAN)和反正切(ATAN)

参见计算状态字的位。

8.2 判断浮点运算指令状态字的位

说明

浮点型指令影响状态字中的下列位：CC 1和CC 0，OV和OS。

下表显示了浮点数(32位)指令的运算结果的状态字中位的信号状态：

结果的有效区域	CC 1	CC 0	OV	OS
+0，-0(零)	0	0	0	*
-3.402823E+38 < 结果 < -1.175494E-38(负值)	0	1	0	*
+1.175494E-38 < 结果 < 3.402824E+38(正值)	1	0	0	*

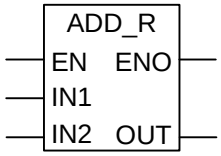
*OS位不受指令运算结果的影响。

结果的无效区域	CC 1	CC 0	OV	OS
下溢 -1.175494E-38 < 结果 < - 1.401298E-45(负值)	0	0	1	1
下溢 +1.401298E-45 < 结果 < +1.175494E-38(正值)	0	0	1	1
溢出 结果 < -3.402823E+38(负值)	0	1	1	1
溢出 结果 > 3.402823E+38(正值)	1	0	1	1
无效的浮点数或非法指令 (输入值超出了有效范围)	1	1	1	1

8.3 基本指令

8.3.1 ADD_R 实数加

符号



参数	数据类型	内存区域	说明
EN	BOOL	I、Q、M、L、D	使能输出
ENO	BOOL	I、Q、M、L、D	使能输出
IN1	REAL	I、Q、M、L、D 或常数	被加数
IN2	REAL	I、Q、M、L、D 或常数	加数
OUT	REAL	I、Q、M、L、D	加法结果

说明

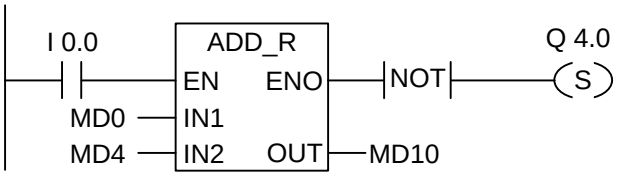
在启用(EN)输入端通过一个逻辑“1”来激活**ADD_R**(实数加)。IN1和IN2相加，结果通过OUT查看。如果结果超出了浮点数允许的范围(溢出或下溢)，OV位和OS位将为“1”并且ENO为“0”，这样便不执行此数学框后由ENO连接的其它功能(层叠排列)。

参见计算状态字的位。

状态字

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
写：	x	x	x	x	x	0	x	x	1

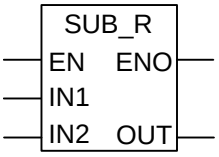
实例



由I0.0处的逻辑“1”激活ADD_R框。MD0 + MD4相加的结果输出到MD10。
如果结果超出了浮点数的允许范围，或者如果没有处理该程序语句(I0.0 = 0)，则设置输出Q4.0。

8.3.2 SUB_R 实数减

符号



参数	数据类型	内存区域	说明
EN	BOOL	I、Q、M、L、D	使能输出
ENO	BOOL	I、Q、M、L、D	使能输出
IN1	REAL	I、Q、M、L、D 或常数	被减数
IN2	REAL	I、Q、M、L、D 或常数	减数
OUT	REAL	I、Q、M、L、D	减法结果

说明

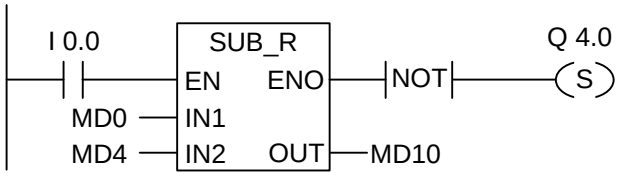
在启用(EN)输入端通过一个逻辑“1”来激活**SUB_R**(实数减)。IN1减去IN2，结果可通过OUT查看。如果该结果超出了浮点数允许的范围(溢出或下溢)，OV位和OS位将为“1”并且ENO为逻辑“0”，这样便不执行此数学框后由ENO连接的其它函数(层叠排列)。

参见计算状态字的位。

状态字

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
写:	x	x	x	x	x	0	x	x	1

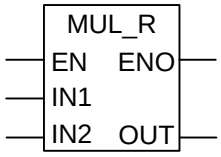
实例



在I0.0处由逻辑“1”激活SUB_R框。MD0 - MD4相减的结果输出到MD10。如果结果超出了浮点数的允许范围，或者如果没有处理该程序语句，则设置输出Q4.0。

8.3.3 MUL_R 实数乘

符号



参数	数据类型	内存区域	说明
EN	BOOL	I、Q、M、L、D	使能输出
ENO	BOOL	I、Q、M、L、D	使能输出
IN1	REAL	I、Q、M、L、D 或常数	被乘数
IN2	REAL	I、Q、M、L、D 或常数	第二个乘运算值
OUT	REAL	I、Q、M、L、D	乘运算结果

说明

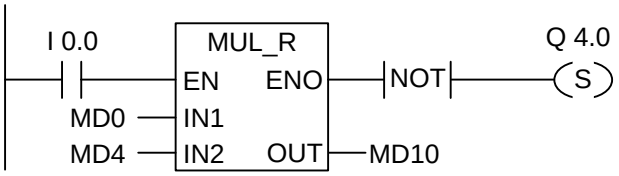
在启用(EN)输入端通过一个逻辑“1”来激活**MUL_R(**实数乘)。IN1和IN2相乘，结果通过OUT查看。如果该结果超出了浮点数允许的范围(溢出或下溢)，OV位和OS位将为“1”并且ENO为逻辑“0”，这样便不执行此数学框后由ENO连接的其它函数(层叠排列)。

参见计算状态字的位。

状态字

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
写：	x	x	x	x	x	0	x	x	1

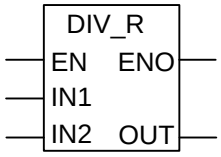
实例



在I0.0处由逻辑“1”激活MUL_R框。MD0 x MD4相乘的结果输出到MD0。如果结果超出了浮点数的允许范围，或者如果没有处理该程序语句，则设置输出Q4.0。

8.3.4 DIV_R 实数除

符号



参数	数据类型	内存区域	说明
EN	BOOL	I、Q、M、L、D	使能输出
ENO	BOOL	I、Q、M、L、D	使能输出
IN1	REAL	I、Q、M、L、D 或常数	被除数
IN2	REAL	I、Q、M、L、D 或常数	除数
OUT	REAL	I、Q、M、L、D	除法结果

说明

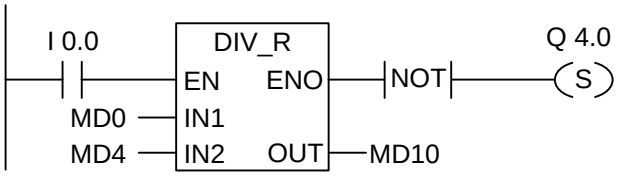
在启用(EN)输入端通过一个逻辑“1”来激活**DIV_R(**实数除)。IN1除以IN2，结果可通过OUT查看。如果该结果超出了浮点数允许的范围(溢出或下溢)，OV位和OS位将为“1”并且ENO为逻辑“0”，这样便不执行此数学框后由ENO连接的其它函数(层叠排列)。

参见计算状态字的位。

状态字

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
写：	x	x	x	x	x	0	x	x	1

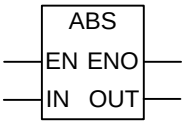
实例



由I0.0处的逻辑“1”激活DIV_R框。MD0除以MD4的结果输出到MD10。如果结果超出了浮点数的允许范围，或者如果没有处理该程序语句，则设置输出Q4.0。

8.3.5 ABS 得到浮点型数字的绝对值

符号



参数	数据类型	内存区域	说明
EN	BOOL	I、Q、M、L、D	使能输出
ENO	BOOL	I、Q、M、L、D	使能输出
IN	REAL	I、Q、M、L、D 或常数	输入值：浮点数
OUT	REAL	I、Q、M、L、D	输出值：浮点数的绝对值

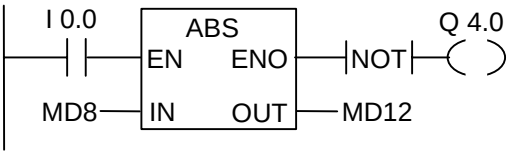
说明

ABS得到浮点型数字的绝对值。

状态字

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
写：	1	-	-	-	-	0	1	1	1

实例



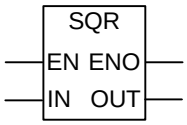
如果I0.0 =“1”，则MD8的绝对值在MD12输出。

MD8 = + 6.234得到MD12 = 6.234。如果未执行该转换(ENO = EN = 0)，则输出Q4.0为“1”。

8.4 扩展指令

8.4.1 SQR 求平方

符号



参数	数据类型	内存区域	说明
EN	BOOL	I、Q、M、L、D	使能输出
ENO	BOOL	I、Q、M、L、D	使能输出
IN	REAL	I、Q、M、L、D 或常数	输入值：浮点数
OUT	REAL	I、Q、M、L、D	输出值：浮点数的平方

说明

SQR求浮点数的平方

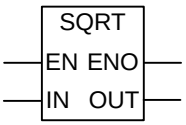
参见计算状态字的位。

状态字

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
写：	x	x	x	x	x	0	x	x	1

8.4.2 SQRT 求平方根

符号



参数	数据类型	内存区域	说明
EN	BOOL	I、Q、M、L、D	使能输出
ENO	BOOL	I、Q、M、L、D	使能输出
IN	REAL	I、Q、M、L、D 或常数	输入值：浮点数
OUT	REAL	I、Q、M、L、D	输出值：浮点数的平方根

说明

SQRT求浮点数的平方根。当地址大于“0”时，此指令得出一个正的结果。唯一例外的是：-0的平方根是 -0。

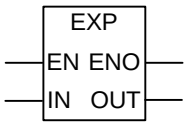
参见计算状态字的位。

状态字

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
写：	x	x	x	x	x	0	x	x	1

8.4.3 EXP 求指数值

符号



参数	数据类型	内存区域	说明
EN	BOOL	I、Q、M、L、D	使能输出
ENO	BOOL	I、Q、M、L、D	使能输出
IN	REAL	I、Q、M、L、D 或常数	输入值：浮点数
OUT	REAL	I、Q、M、L、D	输出值：浮点数的指数值

说明

EXP求浮点数的以e(=2,71828...)为底的指数值。

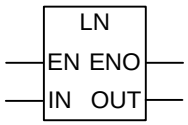
参见计算状态字的位。

状态字

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
写：	x	x	x	x	x	0	x	x	1

8.4.4 LN 求自然对数

符号



参数	数据类型	内存区域	说明
EN	BOOL	I、Q、M、L、D	使能输出
ENO	BOOL	I、Q、M、L、D	使能输出
IN	REAL	I、Q、M、L、D 或常数	输入值：浮点数
OUT	REAL	I、Q、M、L、D	输出值：浮点数的自然对数

说明

LN求浮点数的自然对数。

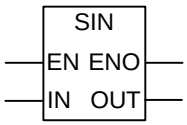
参见计算状态字的位。

状态字

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
写：	x	x	x	x	x	0	x	x	1

8.4.5 SIN 求正弦值

符号



参数	数据类型	内存区域	说明
EN	BOOL	I、Q、M、L、D	使能输出
ENO	BOOL	I、Q、M、L、D	使能输出
IN	REAL	I、Q、M、L、D 或常数	输入值：浮点数
OUT	REAL	I、Q、M、L、D	输出值：浮点数的正弦

说明

SIN求浮点数的正弦值。这里浮点数代表一个以弧度为单位的角度。

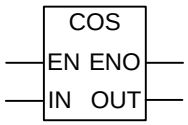
参见计算状态字的位。

状态字

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
写：	x	x	x	x	x	0	x	x	1

8.4.6 COS 求余弦值

符号



参数	数据类型	内存区域	说明
EN	BOOL	I、Q、M、L、D	使能输出
ENO	BOOL	I、Q、M、L、D	使能输出
IN	REAL	I、Q、M、L、D 或常数	输入值：浮点数
OUT	REAL	I、Q、M、L、D	输出值：浮点数的余弦

说明

COS求浮点数的余弦值。这里浮点数代表一个以弧度为单位的角度。

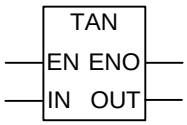
参见计算状态字的位。

状态字

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
写：	x	x	x	x	x	0	x	x	1

8.4.7 TAN 求正切值

符号



参数	数据类型	内存区域	说明
EN	BOOL	I、Q、M、L、D	使能输出
ENO	BOOL	I、Q、M、L、D	使能输出
IN	REAL	I、Q、M、L、D 或常数	输入值：浮点数
OUT	REAL	I、Q、M、L、D	输出值：浮点数的正切

说明

TAN求浮点数的正切值。这里浮点数代表一个以弧度为单位的角度。

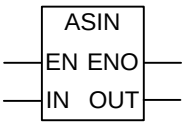
参见计算状态字的位。

状态字

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
写：	x	x	x	x	x	0	x	x	1

8.4.8 ASIN 得到反正弦值

符号



参数	数据类型	内存区域	说明
EN	BOOL	I、Q、M、L、D	使能输出
ENO	BOOL	I、Q、M、L、D	使能输出
IN	REAL	I、Q、M、L、D 或常数	输入值：浮点数
OUT	REAL	I、Q、M、L、D	输出值：浮点数的反正弦

说明

ASIN求一个定义在 - 1 <= 输入值 <= 1范围内的浮点数的反正弦值。结果代表下式范围内的一个以弧度为单位的角度

$$-\pi/2 \leq \text{output value} \leq +\pi/2$$

where π = 3.1415....

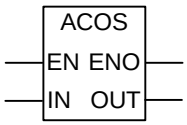
参见计算状态字的位。

状态字

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
写：	x	x	x	x	x	0	x	x	1

8.4.9 ACOS 得到反余弦值

符号



参数	数据类型	内存区域	说明
EN	BOOL	I、Q、M、L、D	使能输出
ENO	BOOL	I、Q、M、L、D	使能输出
IN	REAL	I、Q、M、L、D 或常数	输入值：浮点数
OUT	REAL	I、Q、M、L、D	输出值：浮点数的反余弦

说明

ACOS求一个定义在 -1 <= 输入值 <= 1范围内的浮点数的反余弦值。结果代表下式范围内的一个以弧度为单位的角度

$$0 \leq \text{output value} \leq +\pi$$

where π = 3.1415....

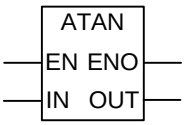
参见计算状态字的位。

状态字

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
写：	x	x	x	x	x	0	x	x	1

8.4.10 ATAN 得到反正切值

符号



参数	数据类型	内存区域	说明
EN	BOOL	I、Q、M、L、D	使能输出
ENO	BOOL	I、Q、M、L、D	使能输出
IN	REAL	I、Q、M、L、D 或常数	输入值：浮点数
OUT	REAL	I、Q、M、L、D	输出值：浮点数的反正切

说明

ATAN求浮点数的反正切值。结果代表下式范围内的一个以弧度为单位的角度

$$-\pi/2 \leq \text{output value} \leq +\pi/2$$

where $\pi = 3.1415....$

参见计算状态字的位。

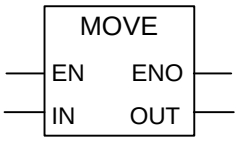
状态字

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
写：	x	x	x	x	x	0	x	x	1

9 传送指令

9.1 MOVE分配值

符号



参数	数据类型	内存区域	说明
EN	BOOL	I、Q、M、L、D	使能输出
ENO	BOOL	I、Q、M、L、D	使能输出
IN	所有长度为8、16或32位的基本数据类型	I、Q、M、L、D 或常数	源值
OUT	所有长度为8、16或32位的基本数据类型	I、Q、M、L、D	目标地址

说明

MOVE(分配值)通过启用EN输入来激活。在IN输入指定的值将复制到在OUT输出指定的地址。ENO与EN的逻辑状态相同。**MOVE**只能复制BYTE、WORD或DWORD数据对象。用户自定义数据类型(如数组或结构)必须使用系统功能“BLKMOVE”(SFC 20)来复制。

状态字

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
写:	1	-	-	-	-	0	1	1	1

MCR (主控继电器)依存关系

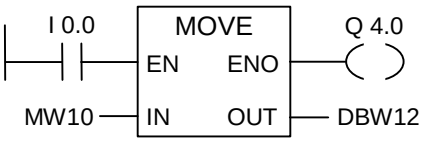
只有当“传送”框位于激活的MCR区内时，才会激活MCR依存。在激活的MCR区内，如果开启了MCR，同时有通往启用输入的电流，则按如上所述复制寻址的数据。如果MCR关闭，并执行了MOVE，则无论当前IN状态如何，均会将逻辑“0”写入到指定的OUT地址。

注意

将某个值传送给不同长度的数据类型时，会根据需要截断或以零填充高位字节：

实例：双字	**1111 1111**	**0000 1111**	**1111 0000**	**0101 0101**
Move	**结果**			
到双字：	1111 1111	0000 1111	1111 0000	0101 0101
到字节：				0101 0101
到字：			1111 0000	0101 0101
实例：字节				**1111 0000**
Move	**结果**			
到字节：				1111 0000
到字：			0000 0000	1111 0000
到双字：	0000 0000	0000 0000	0000 0000	1111 0000

实例



如果I0.0为“1”，则执行指令。把MW10的内容复制到当前打开DB的数据字12。

如果执行了指令，则Q4.0为“1”。

如果实例梯级在激活的MCR区之内：

- 如果MCR开启，则按如上所述将MW10数据复制到DBW12。
- 如果MCR关闭，则将“0”写入到DBW12。

10 程序控制指令

10.1 程序控制指令概述

说明

可使用下列程序控制指令：

- ---(CALL) 调用来自线圈的FC SFC(不带参数)
- CALL_FB 调用来自框的FB
- CALL_FC 调用来自框的FC
- CALL_SFB 调用来自框的系统FB
- CALL_SFC 调用来自框的系统FC
- 调用多重背景
- 调用来自库的块
- 使用MCR功能的重要注意事项
- ---(MCR<) 主控制继电器打开
- ---(MCR>) 主控制继电器关闭
- ---(MCRA) 主控制继电器激活
- ---(MCRD) 主控制继电器取消激活
- RET 返回

10.2 ---(Call) 调用来自线圈的FC SFC(不带参数)

符号

<FC/SFC编号>

---(CALL)

参数	数据类型	内存区域	说明
<FC/SFC编号>	BLOCK_FC BLOCK_SFC	-	FC/SFC编号，范围取决于CPU

说明

---(Call)(不带参数调用FC或SFC)用于调用没有传递参数的功能(FC)或系统功能(SFC)。只有在CALL线圈上RLO为“1”时，才执行调用。当执行**---(Call)**时，

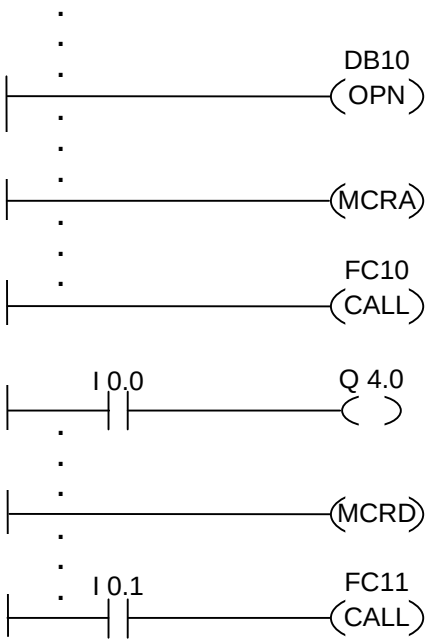
- 存储调用块的返回地址，
- 由当前的本地数据区代替以前的本地数据区，
- 将MA位(有效MCR位)移位到B堆栈中，
- 为被调用的功能创建一个新的本地数据区。

之后，在被调用的FC或SFC中继续进行程序处理。

状态字

		BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
无条件：	写：	-	-	-	-	0	0	1	-	0
有条件：	写：	-	-	-	-	0	0	1	1	0

实例



上面所示的梯形图的梯级是由用户编写的功能块中的程序段。在该FB中，打开DB10，并激活MCR功能。当执行无条件调用FC10时，发生下面情况：

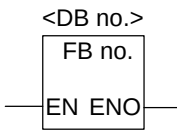
保存调用FB的返回地址，并保存DB10和调用FB的背景数据块的选择数据。
在MCRA指令中设置成“1”的MA位被推入到B堆栈中，然后为被调用块(FC10)将其设置成“0”。继续在FC10中进行程序处理。当FC10要求MCR功能时，必须在FC10内重新激活该功能。当完成FC10时，程序处理返回调用FB。不管FC10使用了哪个DB，都恢复MA位，DB10和用户编写FB的背景数据块重新成为当前DB。通过将I0.0的逻辑状态分配给输出Q4.0，程序继续处理下一个梯级。FC11的调用为条件调用。只有在I0.1为“1”时，才执行调用。执行调用后，将程序控制传递给FC11并从FC11返回程序控制的过程与已经描述过的FC10的过程相同。

注意

返回调用块后，不是总会再次打开以前打开的DB。请一定要阅读自述文件中的注意事项。

10.3 CALL_FB 调用来自框的FB

符号



该符号取决于FB(是否带参数以及带多少个参数)。它必须具有EN、ENO，以及FB的名称或编号。

参数	数据类型	内存区域	说明
EN	BOOL	I、Q、M、L、D	使能输出
ENO	BOOL	I、Q、M、L、D	使能输出
FB编号 DB号	BLOCK_FB BLOCK_DB	- -	FB/DB编号，范围取决于CPU

说明

当EN为“1”时，执行**CALL_FB**(调用来自框的功能块)。当执行CALL_FB时，

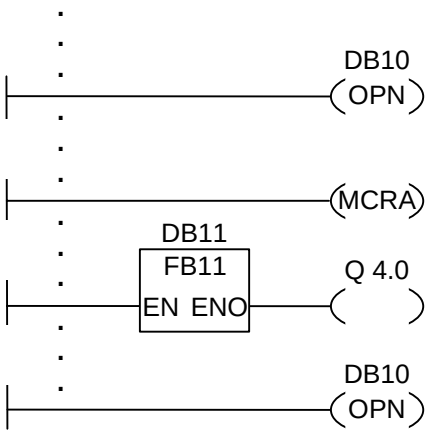
- 存储调用块的返回地址，
- 存储两个当前数据块(DB和背景DB)的选择数据，
- 由当前的本地数据区代替以前的本地数据区，
- 将MA位(有效MCR位)移位到B堆栈中，
- 为被调用的功能块创建一个新的本地数据区。

之后，在被调用的功能块中继续进行程序处理。扫描BR位，以查找ENO。用户必须使用---(SAVE)将所要求的状态(错误判断)分配给被调用块中的BR位。

状态字

		BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
无条件：	写：	x	-	-	-	0	0	x	x	x
有条件：	写：	-	-	-	-	0	0	x	x	x

实例



上面所示的梯形图的梯级是由用户编写的功能块中的程序段。在该FB中，打开DB10，并激活MCR功能。当执行无条件调用FB11时，发生下面情况：

保存调用FB的返回地址，并保存DB10和调用FB的背景数据块的选择数据。
在MCRA指令中设置成“1”的MA位被推入到B堆栈中，然后为被调用块(FB11)将MA位设置成“0”。继续在FB11中进行程序处理。当FB11要求MCR功能时，必须在FB11内重新激活该功能。必须使用指令---(SAVE)在BR位中保存RLO的状态，以便评估调用FB中的错误。当完成FB11时，程序处理返回调用FB。恢复MA位，并重新打开用户编写的FB的背景数据块。当正确处理FB11时，ENO = “1”，因此Q4.0 = “1”。

注意

打开FB或SFB时，会丢失以前打开的DB的编号。必须重新打开所要求的DB。

10.4 CALL_FC 调用来自框的FC

符号

该符号取决于FC(是否带参数以及带多少个参数)。它必须具有EN、ENO，以及FC的名称或编号。

参数	数据类型	内存区域	说明
EN	BOOL	I、Q、M、L、D	使能输出
ENO	BOOL	I、Q、M、L、D	使能输出
FC编号	BLOCK_FC	-	FCC编号，范围取决于CPU

说明

CALL_FC (调用来自框的功能)用于调用一个功能(FC)。当EN为“1”时，执行调用。如果执行了CALL_FC，

- 存储调用块的返回地址，
- 由当前的本地数据区代替以前的本地数据区，
- 将MA位(有效MCR位)移位到B堆栈中，
- 为被调用的功能创建一个新的本地数据区。

之后，在被调用的功能中继续进行程序处理。

扫描BR位，以查找ENO。用户必须使用**---(SAVE)**将所要求的状态(错误评估)分配给被调用块中的BR位。

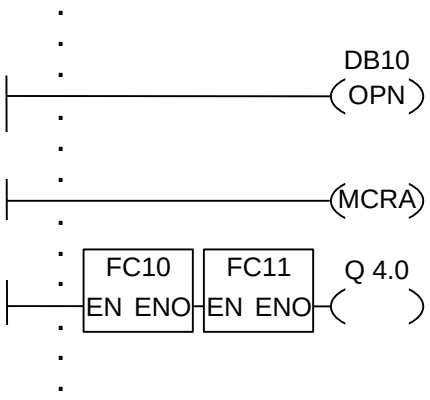
当调用一个功能，而被调用块的变量声明表中具有IN、OUT和IN_OUT声明时，这些变量以形式参数列表添加到调用块的程序中。

当调用功能时，**必须**在调用位置处将实际参数分配给形式参数。功能声明中的任何初始值都没有含义。

状态字

		BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
无条件：	写：	x	-	-	-	0	0	x	x	x
有条件：	写：	-	-	-	-	0	0	x	x	x

实例



上面所示的梯形图的梯级是由用户编写的功能块中的程序段。在该FB中，打开DB10，并激活MCR功能。当执行无条件调用FC10时，发生下面情况：

保存调用FB的返回地址，并保存DB10和调用FB的背景数据块的选择数据。在MCRA指令中设置成“1”的MA位被推入到B堆栈中，然后为被调用块(FC10)将其设置成“0”。继续在FC10中进行程序处理。当FC10要求MCR功能时，必须在FC10内重新激活该功能。必须使用指令---(SAVE)在BR位中保存RLO的状态，以便评估调用FB中的错误。当完成FC10时，程序处理返回调用FB。恢复MA位。执行FC10后，根据ENO，在调用FB中继续进行程序处理：

ENO = “1” FC11已处理

ENO = “0” 在下一个程序段中开始处理

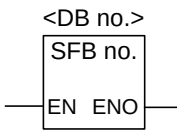
如果也正确处理了FC11，则ENO = “1”，因此Q**4.0 = “1”**。

注意

返回调用块后，不是总会再次打开以前打开的DB。请一定要阅读自述文件中的注意事项。

10.5 CALL_SFB 调用来自框的系统FB

符号



该符号取决于SFB(是否带参数以及带多少个参数)。它必须具有EN、ENO，以及SFB的名称或编号。

参数	数据类型	内存区域	说明
EN	BOOL	I、Q、M、L、D	使能输出
ENO	BOOL	I、Q、M、L、D	使能输出
SFB编号 DB号	BLOCK_SFB BLOCK_DB	- -	SFB编号，范围取决于CPU

说明

当EN为“1”时，执行**CALL_SFB**(调用来自框的系统功能块)。执行CALL_SFB时，

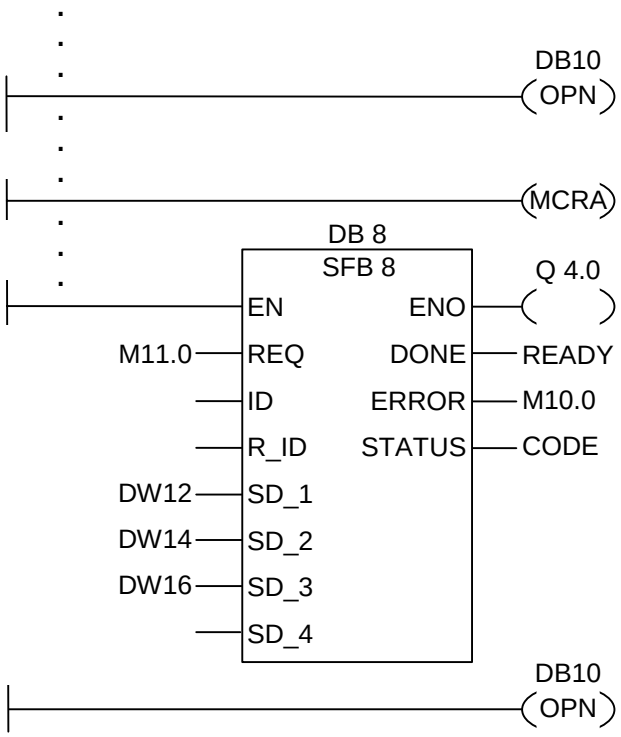
- 存储调用块的返回地址，
- 存储两个当前数据块(DB和背景DB)的选择数据，
- 由当前的本地数据区代替以前的本地数据区，
- 将MA位(有效MCR位)移位到B堆栈中，
- 为被调用的系统功能块创建一个新的本地数据区。

然后在被调用的SFB中继续进行程序处理。当调用SFB (EN = “1”)且没有发生错误时，ENO为“1”。

状态字

		BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
无条件：	写：	x	-	-	-	0	0	x	x	x
有条件：	写：	-	-	-	-	0	0	x	x	x

实例



上面所示的梯形图的梯级是由用户编写的功能块中的程序段。在该FB中，打开DB10，并激活MCR功能。当执行无条件调用SFB8时，发生下面情况：

保存调用FB的返回地址，并保存DB10和调用FB的背景数据块的选择数据。在MCRA指令中设置成“1”的MA位被推入到B堆栈中，然后为被调用块(SFB8)将MA位设置成“0”。继续在SFB8中进行程序处理。当完成SFB8时，程序处理返回调用FB。恢复MA位，且用户编写的FB的背景数据块成为当前背景DB。当正确处理SFB8时，ENO = “1”，因此Q4.0 = “1”。

注意

打开FB或SFB时，会丢失以前打开的DB的编号。必须重新打开所要求的DB。

10.6 CALL_SFC 调用来自框的系统FC

符号

该符号取决于SFC (是否带参数以及带多少个参数)。它必须具有EN、ENO，以及SFC的名称或编号。

参数	数据类型	内存区域	说明
EN	BOOL	-	使能输出
ENO	BOOL	-	使能输出
SFC编号	BLOCK_SFC	-	SFC编号；范围取决于CPU

说明

CALL_SFC(调用来自框的系统功能)用于调用一个SFC。当EN为“1”时，执行调用。当执行CALL_SFC时，

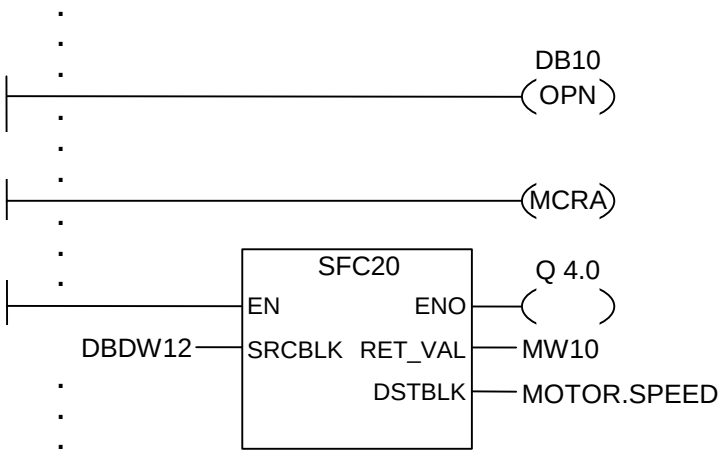
- 存储调用块的返回地址，
- 由当前的本地数据区代替以前的本地数据区，
- 将MA位(有效MCR位)移位到B堆栈中，
- 为被调用的系统功能创建一个新的本地数据区。

之后，在被调用的SFC中继续进行程序处理。当调用SFC (EN = “1”)且没有发生错误时，ENO为“1”。

状态字

		BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
无条件：	写：	x	-	-	-	0	0	x	x	x
有条件：	写：	-	-	-	-	0	0	x	x	x

实例



上面所示的梯形图的梯级是由用户编写的功能块中的程序段。在该FB中，打开DB10，并激活MCR功能。当执行无条件调用SFC20时，发生下面情况：

保存调用FB的返回地址，并保存DB10和调用FB的背景数据块的选择数据。在MCRA指令中设置成“1”的MA位被推入到B堆栈中，然后为被调用块(SFC20)将MA位设置成“0”。继续在SFC20中进行程序处理。当完成SFC20时，程序处理返回调用FB。恢复MA位。

处理SFC20后，根据ENO，在调用FB中继续进行程序处理：

ENO = “1” Q4.0 = “1”

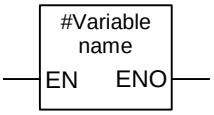
ENO = “0” Q4.0 = “0”

注意

返回调用块后，不是总会再次打开以前打开的DB。请一定要阅读自述文件中的注意事项。

10.7 调用多重背景

符号



参数	数据类型	内存区域	说明
EN	BOOL	I、Q、M、L、D	使能输出
ENO	BOOL	I、Q、M、L、D	使能输出
# 变量名	FB、SFB	-	多重背景的名称

说明

通过声明一个数据类型为功能块的静态变量，创建一个多重背景。只有已经声明的多重背景才会包括在程序元素目录中。多重背景的符号改变取决于是否带参数以及带多少个参数。始终标出EN、ENO和变量名。

状态字

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
写：	-	-	-	-	0	0	x	x	x

10.8 调用来自库的块

在此，可使用SIMATIC管理器中可供使用的库来选择下列块

- 集成在CPU操作系统中的块(对于V3版本的STEP 7项目，为“标准库”，对于V2版本的STEP 7项目，为“stdlibs (V2)”)
- 由于要多次使用而自行在库中保存的块。

10.9 使用MCR功能的重要注意事项

注意在其中使用MCRA激活主控制继电器的块：

- 取消激活MCR时，在MCR(和)MCR之间的程序段中的所有赋值都写入数值0。这对包含赋值的**所有**框都有效，包括传递到块的参数在内。
- 当**MCR<**指令之前的RLO = 0时，取消激活MCR。

危险：PLC处于STOP状态或未定义的运行特征！

编译器还对在VAR_TEMP中定义的临时变量之后的局部数据进行写访问，以计算地址。
这表示下列命令序列将把PLC设置成STOP状态，或导致未定义的运行特征：

形式参数访问

- 访问STRUCT、UDT、ARRAY、STRING类型的复杂FC参数的组件
- 访问来自具有多重背景能力的块(V2版本的块)的IN_OUT区域的STRUCT、UDT、ARRAY、STRING类型的复杂FB参数的组件。
- 当地址高于8180.0时，访问具有多重背景能力(V2版本的块)的功能块的参数。
- 在具有多重背景能力(V2版本的块)的功能块中访问类型为BLOCK_DB的参数，打开DB0。任何后继数据访问将CPU设置成STOP模式。T 0、C 0、FC0或FB0始终用于TIMER、COUNTER、BLOCK_FC和LOCK_FB。

参数传递

- 调用可传递参数的功能。

LAD/FBD

- 梯形图或FBD中的T分支和中线输出以RLO = 0开始。

补救方法

释放上述命令，使其与MCR不相关：

1. 在所述语句或程序段之前，使用**主控制继电器取消激活**指令，取消激活主控制继电器。
2. 在所述语句或程序段之后，使用**主控制继电器激活指**令，重新激活主控制继电器。

10.10 ---(MCR<) 主控制继电器打开

使用MCR功能的重要注意事项

符号

---(MCR<)

说明

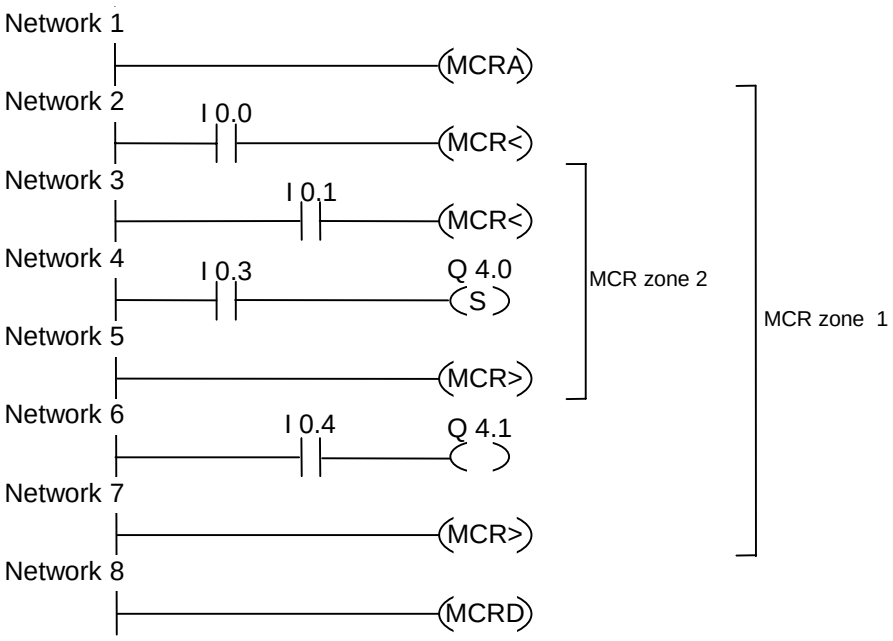
---(MCR<)(打开主控制继电器区域)在MCR堆栈中保存RLO。MCR嵌套堆栈为LIFO(后入先出)堆栈，且只能有8个堆栈条目(嵌套级别)。当堆栈已满时，---(MCR<)功能产生一个MCR堆栈故障(MCRF)。下列元素与MCR有关，并在打开MCR区域时，受保存在MCR堆栈中的RLO状态的影响：

- --(#) 中间输出
- —( ) 输出
- --(S) 设置输出
- --(R) 复位输出
- RS 复位触发器
- SR 置位触发器
- MOVE 分配值

状态字

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
写:	-	-	-	-	-	0	1	-	0

实例



MCRA梯级激活MCR功能。然后可以创建至多8个嵌套MCR区域。在此实例中，有两个MCR区域。按如下执行该功能：

I0.0 = “1”(区域1的MCR打开)：将I0.4的逻辑状态分配给Q4.1

I0.0 = “0”(区域1的MCR关闭)：无论输入I0.4的逻辑状态如何，Q4.1都为0。

I0.1 = “1”(区域2的MCR打开)：当I0.3为“1”时，将Q4.0设置成“1”

I0.1 = “0”(区域2的MCR关闭)：无论I0.3的逻辑状态如何，Q4.0都保持不变

10.11 ---(MCR>) 主控制继电器关闭

使用MCR功能的重要注意事项

符号

---(MCR>)

说明

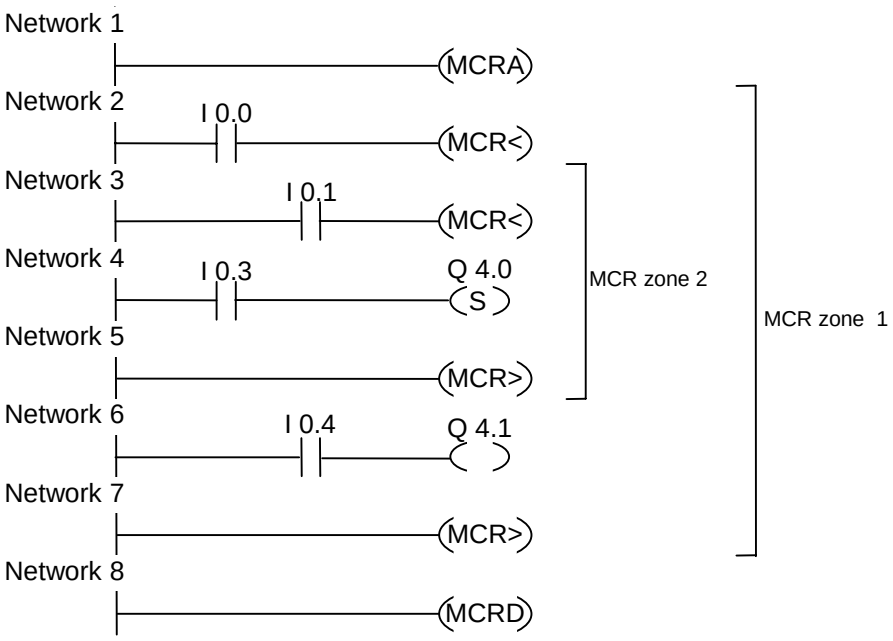
---(MCR>)(关闭最后打开的MCR区域)从MCR堆栈中删除一个RLO条目。MCR嵌套堆栈为LIFO(后入先出)堆栈，且只能有8个堆栈条目(嵌套级别)。当堆栈已空时，---(MCR>)产生一个MCR堆栈故障(MCRF)。下列元素与MCR有关，并在打开MCR区域时，受保存在MCR堆栈中的RLO状态的影响：

- --(#) 中间输出
- —( ) 输出
- --(S) 设置输出
- --(R) 复位输出
- RS 复位触发器
- SR 置位触发器
- MOVE 分配值

状态字

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
写:	-	-	-	-	-	0	1	-	0

实例



---(MCRA)梯级激活MCR功能。然后可以创建至多8个嵌套MCR区域。在此实例中，有两个MCR区域。第一个---(MCR>)(MCR关闭)梯级属于第二个(MCR打开)梯级。两者之间的所有梯级都属于MCR区域2。按如下执行这些功能：

I0.0 = “1”：将I0.4的逻辑状态分配给Q4.1

I0.0 = “0”：无论输入I0.4的逻辑状态如何，Q4.1都为0。

I0.1 = “1”：当I0.3为“1”时，将Q4.0设置成“1”

I0.1 = “0”：无论I0.3的逻辑状态如何，Q4.0都保持不变

10.12 ---(MCRA) 主控制继电器激活

使用MCR功能的重要注意事项

符号

---(MCRA)

说明

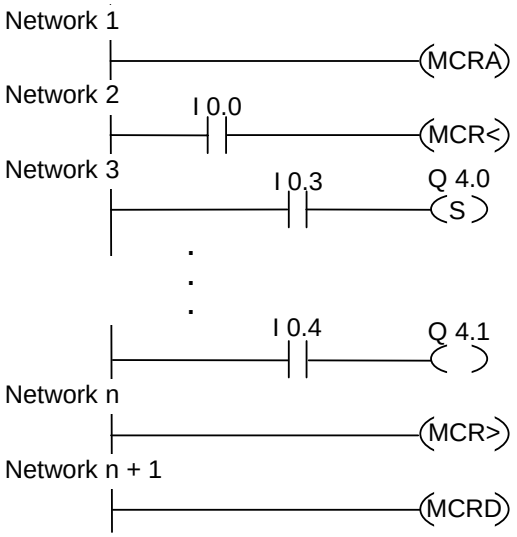
---(MCRA)(激活主控制继电器)激活主控制继电器功能。在该命令后，可以使用下列命令编程MCR区域：

- ---(MCR<)
- ---(MCR>)

状态字

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
写：	-	-	-	-	-	-	-	-	-

实例



MCRA梯级激活MCR功能。MCR<和MCR>(输出Q4.0、Q4.1)之间的梯级按如下执行：

I0.0 = “1”(MCR打开)：当I0.3为逻辑“1”时，将Q4.0设置成“1”，或当I0.3为“0”时，Q4.0保持不变，并将I0.4的逻辑状态分配给Q4.1

I0.0 = “0”(MCR关闭)：无论I0.3的逻辑状态如何，Q4.0保持不变，无论I0.4的逻辑状态如何，Q4.1为“0”

在下一个梯级中，指令---(MCRD)取消激活MCR。这表示不能再使用指令对---(MCR<)和--(MCR>)编程更多的MCR区域。

10.13 ---(MCRD) 主控制继电器取消激活

使用MCR功能的重要注意事项

符号

---(MCRD)

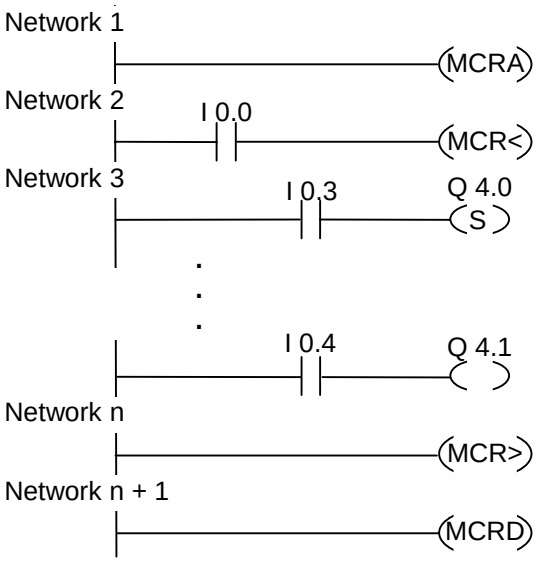
说明

---(MCRD)(取消激活主控制继电器)取消激活MCR功能。在该命令后，不能编程MCR区域。

状态字

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
写：	-	-	-	-	-	-	-	-	-

实例



MCRA梯级激活MCR功能。MCR<和MCR>(输出Q4.0、Q4.1)之间的梯级按如下执行：

I0.0 = “1”(MCR打开)：当I0.3为逻辑“1”时，将Q4.0设置成“1”，并将I0.4的逻辑状态分配给Q4.1。

I0.0 = “0”(MCR关闭)：无论I0.3的逻辑状态如何，Q4.0保持不变，无论I0.4的逻辑状态如何，Q4.1为“0”。

在下一个梯级中，指令---(MCRD)取消激活MCR。这表示不能再使用指令对---(MCR<)和--(MCR>)编程更多的MCR区域。

10.14 ---(RET) 返回

符号

---(RET)

说明

RET(返回)用于有条件地退出块。对于该输出，要求在前面使用一个逻辑运算。

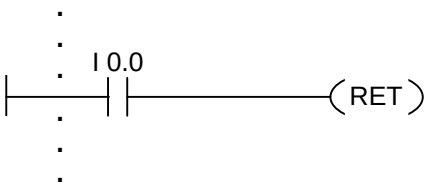
状态字

条件返回(当RLO = “1”时，返回)：

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
写：	*	-	-	-	0	0	1	1	0

* 在“SAVE; BEC,”序列中内部显示**RET**操作。这还影响BR位。

实例



当I0.0为“1”时，退出块。

11 移位和循环移位指令

11.1 移位指令

11.1.1 移位指令概述

说明

可使用移位指令逐位向左或向右移动输入端IN的内容(另请参见CPU寄存器)。向左移n位会将输入IN的内容乘以2的n次幂(2 n)；向右移n位则会将输入IN的内容除以2的n次幂(2 n)。例如，如果将十进制值3的等效二进制数向左移3位，则在累加器中将得到十进制值24的等效二进制数。如果将十进制值16的等效二进制数向右移2位，则在累加器中将得到十进制值4的等效二进制数。

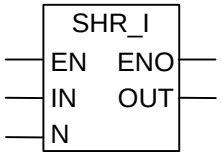
您为输入参数N提供的数值指示要移动的位数。由移位指令移空的位会用零或符号位的信号状态(0表示正，1表示负)补上。最后移动的位的信号状态会被载入状态字的CC 1位中。状态字的CC 0位和OV位会被复位为0。可以使用跳转指令来评估CC 1位。

可使用如下移位指令：

- SHR_I 整数右移
- SHR_DI 长整数右移
- SHL_W 左移字
- SHR_W 右移字
- SHL_DW 双字左移
- SHR_DW 右移双字

11.1.2 SHR_I 整数右移

符号

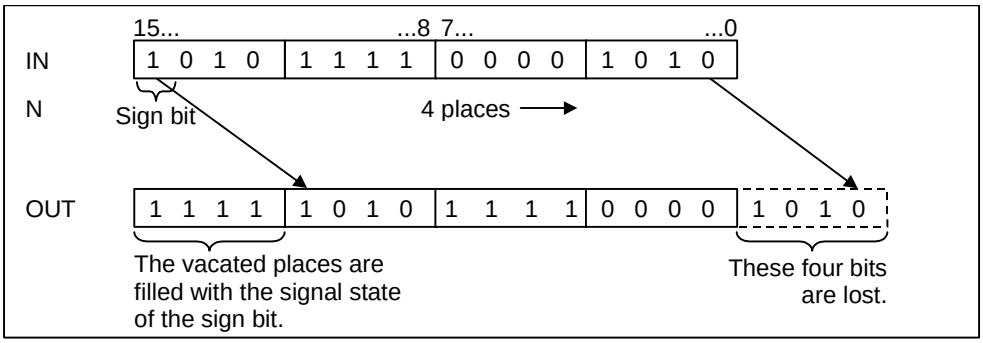


参数	数据类型	内存区域	说明
EN	BOOL	I、Q、M、L、D	使能输出
ENO	BOOL	I、Q、M、L、D	使能输出
IN	INT	I、Q、M、L、D	要移位的值
N	WORD	I、Q、M、L、D	要移动的位数
OUT	INT	I、Q、M、L、D	移位指令的结果

说明

SHR_I(整数右移)指令通过使能(EN)输入位置上的逻辑“1”来激活。SHR_I指令用于将输入IN的0至15位逐位向右移动。16到31位不受影响。输入N用于指定移位的位数。如果N大于16，命令将按照N等于16的情况执行。自左移入的、用于填补空出位的位位置将被赋予位15的逻辑状态(整数的符号位)。这意味着，当该整数为正时，这些位将被赋值“0”，而当该整数为负时，则被赋值为“1”。可在输出OUT位置扫描移位指令的结果。如果N不等于0，则SHR_I会将CC 0位和OV位设为“0”。

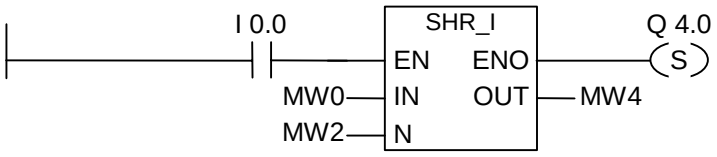
ENO与EN具有相同的信号状态。



状态字

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
写:	x	x	x	x	-	x	x	x	1

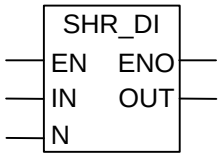
实例



SHR_I框由I0.0位置上的逻辑“1”激活。装载MW0并将其右移由MW2指定的位数。结果将被写入MW4。置位Q4.0。

11.1.3 SHR_DI 右移长整数

符号



参数	数据类型	内存区域	说明
EN	BOOL	I、Q、M、L、D	使能输出
ENO	BOOL	I、Q、M、L、D	使能输出
IN	DINT	I、Q、M、L、D	要移位的值
N	WORD	I、Q、M、L、D	要移动的位数
OUT	DINT	I、Q、M、L、D	移位指令的结果

说明

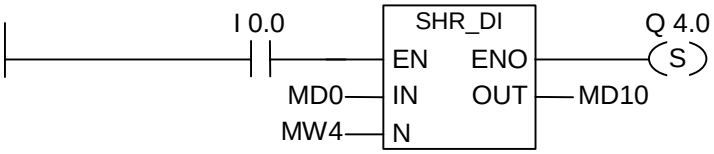
SHR_DI(右移长整数)指令通过使能(EN)输入位置上的逻辑“1”来激活。SHR_DI指令用于将输入IN的0至31位逐位向右移动。输入N用于指定移位的位数。如果N大于32，命令将按照N等于32的情况执行。自左移入的、用于填补空出位的位位置将被赋予位31的逻辑状态(整数的符号位)。这意味着，当该整数为正时，这些位将被赋值“0”，而当该整数为负时，则被赋值为“1”。可在输出OUT位置扫描移位指令的结果。如果N不等于0，则SHR_DI会将CC 0位和OV位设为“0”。

ENO与EN具有相同的信号状态。

状态字

	BR	**CC 1**	**CC 0**	**OV**	**OS**	**OR**	**STA**	**RLO**	**/FC**
写：	x	x	x	x	-	x	x	x	1

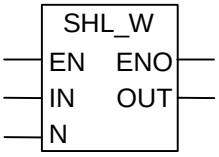
实例



SHR_DI框由I0.0位置上的逻辑“1”激活。装载MD0并将其右移由MW4指定的位数。结果将被写入MD10。置位Q4.0。

11.1.4 SHL_W 字左移

符号

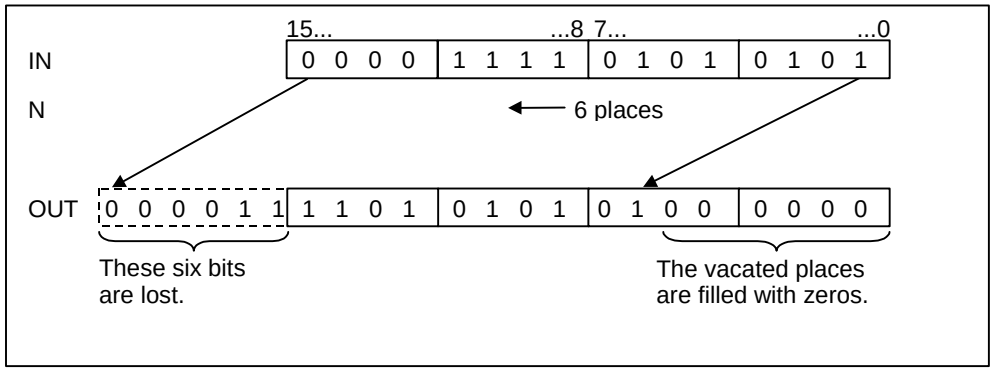


参数	数据类型	内存区域	说明
EN	BOOL	I、Q、M、L、D	使能输出
ENO	BOOL	I、Q、M、L、D	使能输出
IN	WORD	I、Q、M、L、D	要移位的值
N	WORD	I、Q、M、L、D	要移动的位数
OUT	WORD	I、Q、M、L、D	移位指令的结果

说明

SHL_W(字左移)指令通过使能(EN)输入位置上的逻辑“1”来激活。SHL_W指令用于将输入IN的0至15位逐位向左移动。16到31位不受影响。输入N用于指定移位的位数。若N大于16，此命令会在输出OUT位置上写入“0”，并将状态字中的CC 0位和OV位设置为“0”。将自右移入N个零，用以补上空出的位位置。可在输出OUT位置扫描移位指令的结果。如果N不等于0，则SHL_W会将CC 0位和OV位设为“0”。

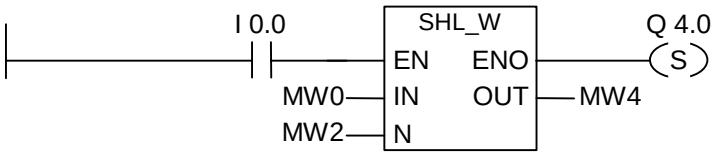
ENO与EN具有相同的信号状态。



状态字

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
写：	x	x	x	x	-	x	x	x	1

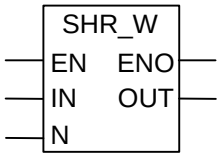
实例



SHL_W框由I0.0位置上的逻辑“1”激活。装载MW0并将其左移由MW2指定的位数。结果将被写入MW4。置位Q4.0。

11.1.5 SHR_W 字右移

符号



参数	数据类型	内存区域	说明
EN	BOOL	I、Q、M、L、D	使能输出
ENO	BOOL	I、Q、M、L、D	使能输出
IN	WORD	I、Q、M、L、D	要移位的值
N	WORD	I、Q、M、L、D	要移动的位数
OUT	WORD	I、Q、M、L、D	字移位指令的结果

说明

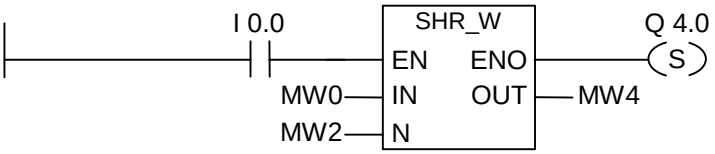
SHR_W(字右移)指令通过使能(EN)输入位置上的逻辑“1”来激活。SHR_W指令用于将输入IN的0至15位逐位向右移动。16到31位不受影响。输入N用于指定移位的位数。若N大于16，此命令会在输出OUT位置上写入“0”，并将状态字中的CC 0位和OV位设置为“0”。将自左移入N个零，用以补上空出的位位置。可在输出OUT位置扫描移位指令的结果。如果N不等于0，则SHR_W会将CC 0位和OV位设为“0”。

ENO与EN具有相同的信号状态。

状态字

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
写：	x	x	x	x	-	x	x	x	1

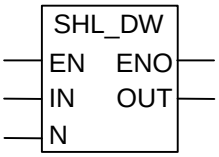
实例



SHR_W框由I0.0位置上的逻辑“1”激活。装载MW0并将其右移由MW2指定的位数。结果将被写入MW4。置位Q4.0。

11.1.6 SHL_DW 双字左移

符号



参数	数据类型	内存区域	说明
EN	BOOL	I、Q、M、L、D	使能输出
ENO	BOOL	I、Q、M、L、D	使能输出
IN	DWORD	I、Q、M、L、D	要移位的值
N	WORD	I、Q、M、L、D	要移动的位数
OUT	DWORD	I、Q、M、L、D	双字移位指令的结果

说明

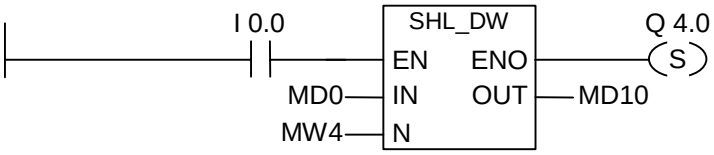
SHL_DW(双字左移)指令通过使能(EN)输入位置上的逻辑“1”来激活。SHL_DW指令用于将输入IN的0至31位逐位向左移动。输入N用于指定移位的位数。若N大于32，此命令会在输出OUT位置上写入“0”并将状态字中的CC 0和OV位设置为“0”。将自右移入N个零，用以补上空出的位位置。可在输出OUT位置扫描双字移位指令的结果。如果N不等于0，则SHL_DW会将CC 0位和OV位设为“0”。

ENO与EN具有相同的信号状态。

状态字

	BR	**CC 1**	**CC 0**	**OV**	**OS**	**OR**	**STA**	**RLO**	**/FC**
写：	x	x	x	x	-	x	x	x	1

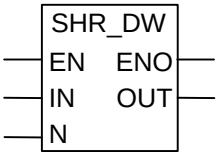
实例



SHL_DW框由I0.0位置上的逻辑“1”激活。装载MD0并将其左移由MW4指定的位数。结果将被写入MD10。置位Q4.0。

11.1.7 SHR_DW 双字右移

符号

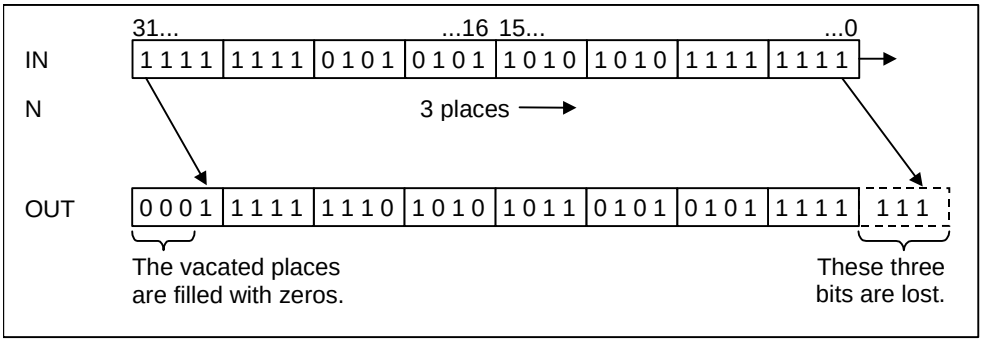


参数	数据类型	内存区域	说明
EN	BOOL	I、Q、M、L、D	使能输出
ENO	BOOL	I、Q、M、L、D	使能输出
IN	DWORD	I、Q、M、L、D	要移位的值
N	WORD	I、Q、M、L、D	要移动的位数
OUT	DWORD	I、Q、M、L、D	双字移位指令的结果

说明

SHR_DW(双字右移)指令通过使能(EN)输入位置上的逻辑“1”来激活。SHR_DW指令用于将输入IN的0至31位逐位向右移动。输入N用于指定移位的位数。若N大于32，此命令会在输出OUT位置上写入“0”并将状态字中的CC 0和OV位设置为“0”。将自左移入N个零，用以补上空出的位位置。可在输出OUT位置扫描双字移位指令的结果。如果N不等于0，则SHR_DW会将CC 0位和OV位设为”。

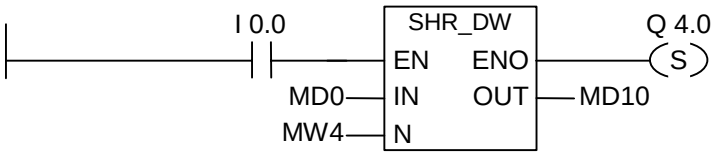
ENO与EN具有相同的信号状态。



状态字

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
写：	x	x	x	x	-	x	x	x	1

实例



SHR_DW框由I0.0位置上的逻辑“1”激活。装载MD0并将其右移由MW4指定的位数。结果将被写入MD10。置位Q4.0。

11.2 循环移位指令

11.2.1 循环移位指令概述

说明

可使用循环移位指令将输入IN的所有内容向左或向右逐位循环移位。移空的位将用被移出输入IN的位的信号状态补上。

您为输入参数N提供的数值指定要循环移位的位数。

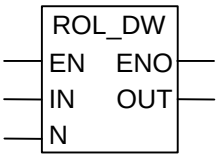
依据具体的指令，循环移位将通过状态字的CC 1位进行。状态字的CC 0位被复位为0。

可使用如下循环移位指令：

- ROL_DW　　循环左移双字
- ROR_DW　　循环右移双字

11.2.2 ROL_DW　双字循环左移

符号

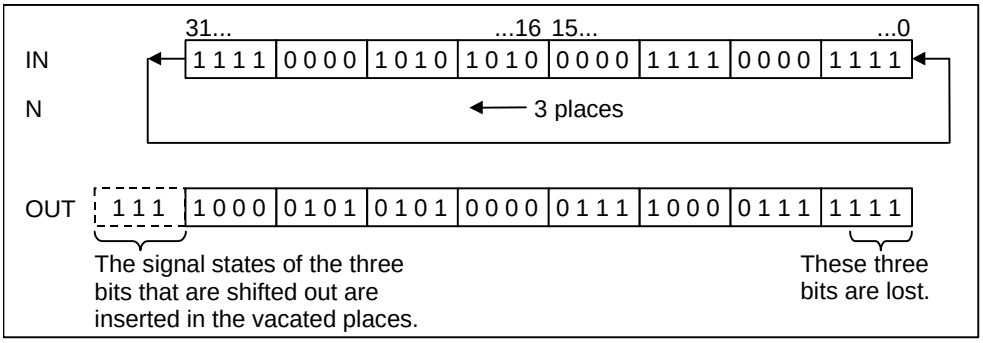


参数	数据类型	内存区域	说明
EN	BOOL	I、Q、M、L、D	使能输出
ENO	BOOL	I、Q、M、L、D	使能输出
IN	DWORD	I、Q、M、L、D	要循环移位的值
N	WORD	I、Q、M、L、D	要循环移位的位数
OUT	DWORD	I、Q、M、L、D	双字循环指令的结果

说明

ROL_DW(双字循环左移)指令通过使能(EN)输入位置上的逻辑“1”来激活。ROL_DW指令用于将输入IN的全部内容逐位向左循环移位。输入N用于指定循环移位的位数。如果N大于32，则双字IN将被循环移位((N-1)对32求模，所得的余数) +1位。自右移入的位位置将被赋予向左循环移出的各个位的逻辑状态。可在输出OUT位置扫描双字循环指令的结果。如果N不等于0，则ROL_DW会将CC 0位和OV位设为“0”。

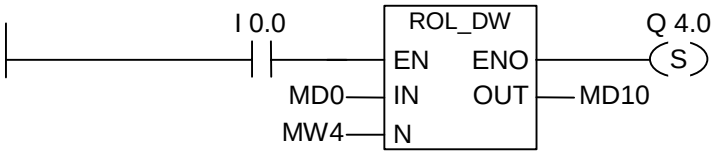
ENO与EN具有相同的信号状态。



状态字

	BR	**CC 1**	**CC 0**	**OV**	**OS**	**OR**	**STA**	**RLO**	**/FC**
写：	x	x	x	x	-	x	x	x	1

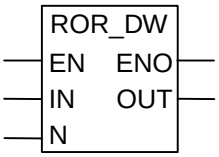
实例



ROL_DW 框由I0.0位置上的逻辑“1”激活。装载MD0并将其向左循环移位由MW4指定的位数。结果将被写入MD10。置位Q4.0。

11.2.3 ROR_DW 双字循环右移

符号

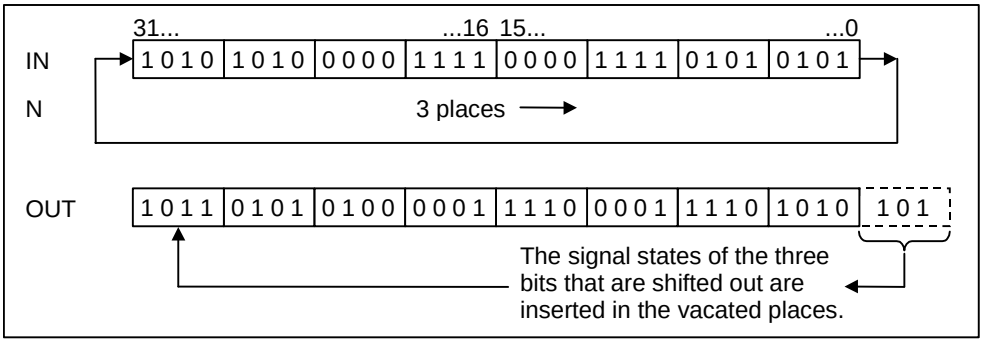


参数	数据类型	内存区域	说明
EN	BOOL	I、Q、M、L、D	使能输出
ENO	BOOL	I、Q、M、L、D	使能输出
IN	DWORD	I、Q、M、L、D	要循环移位的值
N	WORD	I、Q、M、L、D	要循环移位的位数
OUT	DWORD	I、Q、M、L、D	双字循环指令的结果

说明

ROR_DW(双字循环右移)指令通过使能(EN)输入位置上的逻辑“1”来激活。ROR_DW指令用于将输入IN的全部内容逐位向右循环移位。输入N用于指定循环移位的位数。如果N大于32，则双字IN将被循环移位((N-1)对32求模，所得的余数) +1位。自左移入的位位置将被赋予向右循环移出的各个位的逻辑状态。可在输出OUT位置扫描双字循环指令的结果。如果N不等于0，则ROR_DW会将CC 0位和OV位置为“0”。

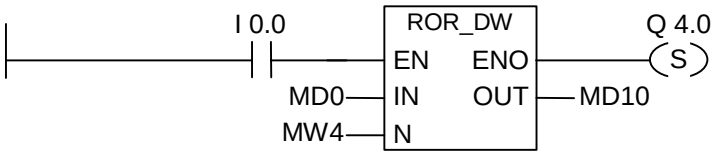
ENO与EN具有相同的信号状态。



状态字

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
写：	x	x	x	x	-	x	x	x	1

实例



ROR_DW框由I0.0位置上的逻辑“1”激活。装载MD0并将其向左循环移位由MW4指定的位数。结果将被写入MD10。置位Q4.0。

12 状态位指令

12.1 状态位指令概述

说明

状态位指令属于位逻辑指令，用于对状态字的位进行处理。各状态位指令分别对下列条件之一做出反应，其中每个条件以状态字的一个或多个位来表示：

- 二进制结果位(BR ---I I---) 被置位(即信号状态为1)。
- 数学运算函数发生溢出(OV ---I I---) 或存储溢出 (OS ---I I---)。
- 算术运算功能的结果无序(UO ---I I---)。
- 数学运算函数的结果与0的关系有： == 0、<> 0、> 0、< 0、>= 0、<= 0。

当状态位指令以串联方式连接时，该指令将根据“与”真值表将其信号状态校验的结果与前一逻辑运算结果合并。当状态位指令以并联方式连接时，该指令将根据“或”真值表将其结果与前一RLO合并。

状态字

状态字是CPU存储器中的一个寄存器，它所包含的位可通过位地址和字逻辑指令来参照。状态字的结构：

2^{15}... ...2^9	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
	BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC

可以通过下列函数求状态字位的值

- 整数数学运算函数
- 浮点数运算函数

12.2 OV ---| |--- 异常位溢出

符号

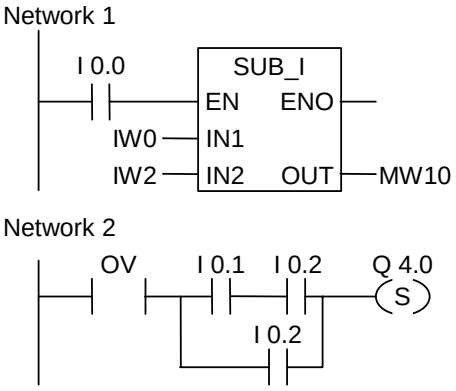
说明

OV ---| |---(溢出异常位)或**OV ---| / |---**(异常位溢出取反)触点符号用于识别上次执行数学运算函数时的溢出。也就是说，函数执行后指令的结果超出了允许的正、负范围。串联使用时，扫描的结果将通过AND与RLO链接；并联使用时，扫描结果通过OR与RLO链接。

状态字

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
写:	-	-	-	-	-	x	x	x	1

实例



I0.0的信号状态为“1”时将激活该框。如果数学运算函数“IW0 - IW2”的结果超出了允许的整数范围，则置位OV位。

OV的信号状态扫描为“1”。如果OV扫描的信号状态为“1”且程序段2的RLO为“1”，则置位Q4.0。

注意

只有在有两个独立的程序段时，才需要OV扫描。否则，如果结果超出了允许的范围，则可以提取为“0”的数学运算函数的ENO输出。

12.3 OS ---| |--- 存储的异常位溢出

符号

说明

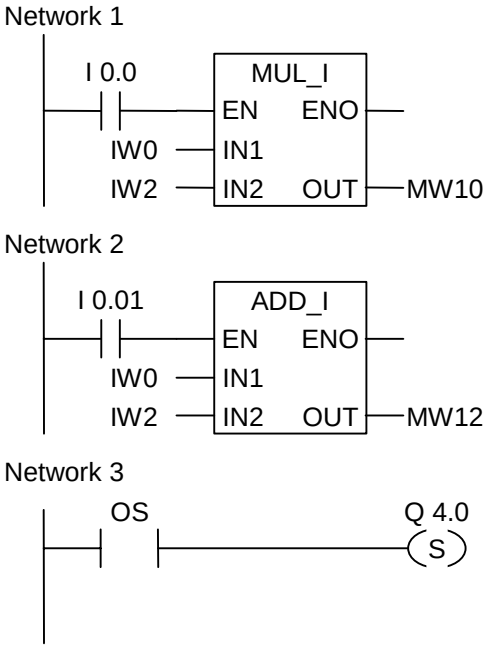
OS ---| |---(存储的异常位溢出)或**OS ---| / |---**(存储的异常位溢出取反)触点符号用于识别和存储数学运算函数中的锁存溢出。如果指令的结果超出了允许的负或正范围，则置位状态字中的OS位。与需要在执行后续数学运算函数前重写的OV位不同，OS位在溢出发生时存储。OS位将保持置位状态，直至离开该块。

串联使用时，扫描的结果将通过AND与RLO链接；并联使用时，扫描结果通过OR与RLO链接。

状态字

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
写:	-	-	-	-	-	x	x	x	1

实例



I0.0的信号状态为“1”时将激活MUL_I框。I0.1的逻辑为“1”时将激活ADD_I框。如果其中一个数学运算函数的结果超出了允许的整数范围，将把状态字中的OS位置位为“1”。如果OS扫描为逻辑“1”，则置位Q4.0。

注意

只有在有两个独立的程序段时，才需要OS扫描。否则，将可以提取第一个数学运算函数的ENO输出，并将其与第二个(层叠排列)数学运算函数的EN输入连接。

12.4 UO ---| |--- 无序异常位

符号

说明

UO ---| |---(无序异常位)或**UO ---| / |---**(无序异常位取反)触点符号用于识别含浮点数的数学运算函数是否无序(也就是说，数学运算函数中的值是否有无效浮点数)。

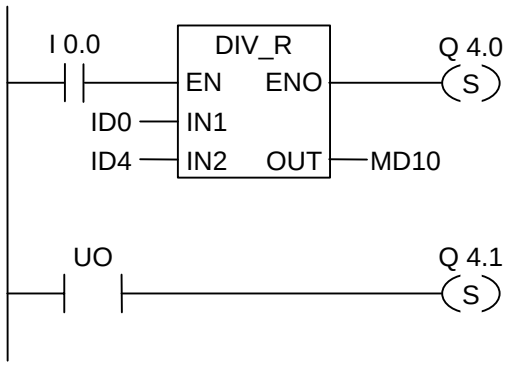
如果含浮点数(UO)的数学运算函数的结果无效，则信号状态扫描为“1”。如果CC 1和CC 0中的逻辑运算显示“无效”，信号状态扫描的结果将是“0”。

串联使用时，扫描的结果将通过AND与RLO链接；并联使用时，扫描结果通过OR与RLO链接。

状态字

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
写：	-	-	-	-	-	x	x	x	1

实例



I0.0的信号状态为“1”时将激活该框。如果ID0或ID4的值为无效浮点数，则数学运算函数无效。如果EN的信号状态 = 1(激活)且在处理函数DIV_R时出错，则ENO的信号状态 = 0。

执行函数DIV_R时如果其中一个值不是有效的**浮点数**，将置位输出Q4.1。

12.5 BR ---| |--- 异常位二进制结果

符号

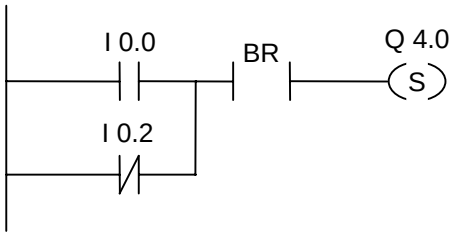
说明

BR ---| |---(异常位BR存储器)或**BR ---| / |---**(异常位BR存储器取反)触点符号用于测试状态字中BR位的逻辑状态。串联使用时，扫描的结果将通过AND与RLO链接；并联使用时，扫描结果通过OR与RLO链接。BR位用于字处理向位处理的转变。

状态字

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
写:	-	-	-	-	-	x	x	x	1

实例



如果I0.0为“1”或I0.2为“0”，且除此RLO外BR位的逻辑状态为“1”，则置位Q4.0。

12.6 ==0 ---| |--- 结果位等于0

符号

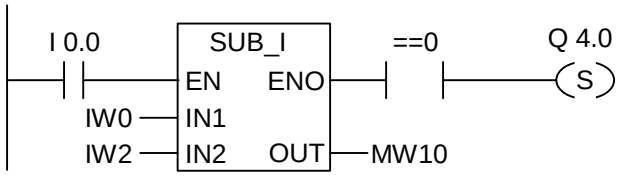
说明

==0 ---| |---(结果位等于0)或**==0 ---| / |---**(结果位取反后大于0)触点符号用于识别数学运算函数的结果是否等于“0”。指令扫描状态字的条件代码位CC 1和CC 0，以确定结果与“0”的关系。串联使用时，扫描的结果将通过AND与RLO链接；并联使用时，扫描结果通过OR与RLO链接。

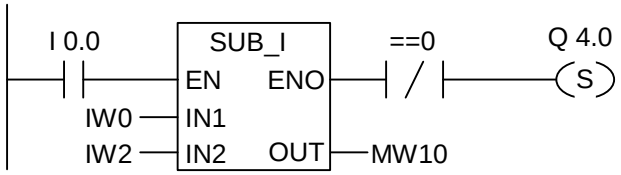
状态字

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
写：	-	-	-	-	-	x	x	x	1

实例



I0.0的信号状态为“1”时将激活该框。如果IW0的值等于IW2的值，数学运算函数“IW0 - IW2”的结果将等于“0”。如果函数得到正确执行且结果**等于**“0”，则置位Q4.0。



如果函数得到正确执行且结果**不等于**“0”，则置位Q4.0。

12.7 <>0 ---| |--- 结果位不等于0

符号

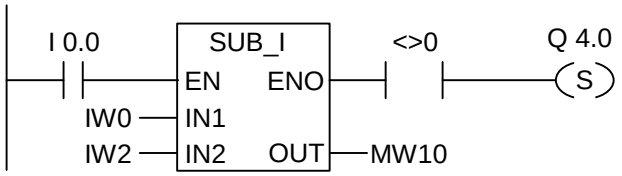
说明

<>0 ---| |---(结果位不等于0)或**<>0 ---| / |---**(结果位取反后小于0)触点符号用于识别数学运算函数的结果是否不等于“0”。指令扫描状态字的条件代码位CC 1和CC 0，以确定结果与“0”的关系。串联使用时，扫描的结果将通过AND与RLO链接；并联使用时，扫描结果通过OR与RLO链接。

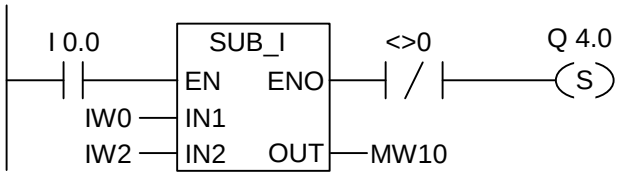
状态字

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
写:	-	-	-	-	-	x	x	x	1

实例



I0.0的信号状态为“1”时将激活该框。如果IW0的值与IW2的值不同，数学运算函数“IW0 - IW2”的结果将不等于“0”。如果函数得到正确执行且结果**不等于**“0”，则置位Q4.0。



如果函数得到正确执行且结果**等于**“0”，则置位Q4.0。

12.8 >0 ---| |--- 结果位大于0

符号

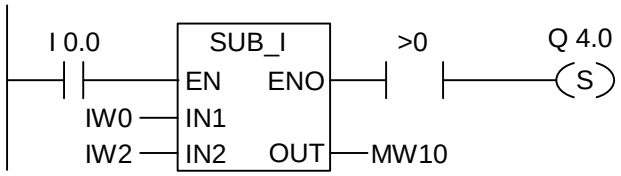
说明

>0 ---| |---(结果位大于0)或 **>0 ---| / |---(**取反结果位大于0)触点符号用于识别数学运算函数的结果是否大于“0”。指令扫描状态字的条件代码位CC 1和CC 0，以确定与“0”的关系。串联使用时，扫描的结果将通过AND与RLO链接；并联使用时，扫描结果通过OR与RLO链接。

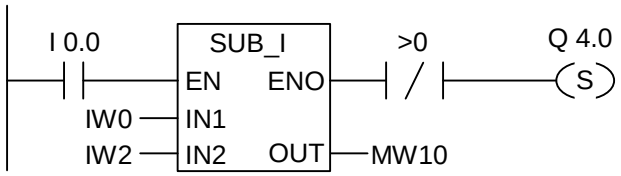
状态字

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
写:	-	-	-	-	-	x	x	x	1

实例



I0.0的信号状态为“1”时将激活该框。如果IW0的值大于IW2的值，数学运算函数“IW0 - IW2”的结果将大于“0”。如果函数得到正确执行且结果**大于**“0”，则置位Q4.0。



如果函数得到正确执行且结果**不**大于“0”，则置位Q4.0。

12.9 <0 ---| |--- 结果位小于0

符号

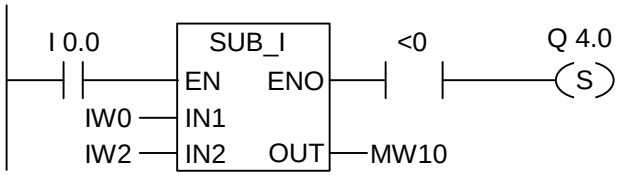
说明

<0 ---| |---(结果位小于0)或**<0 ---| / |---**(结果位取反后小于0)触点符号用于识别数学运算函数的结果是否小于“0”。指令扫描状态字的条件代码位CC 1和CC 0，以确定结果与“0”的关系。串联使用时，扫描的结果将通过AND与RLO链接；并联使用时，扫描结果通过OR与RLO链接。

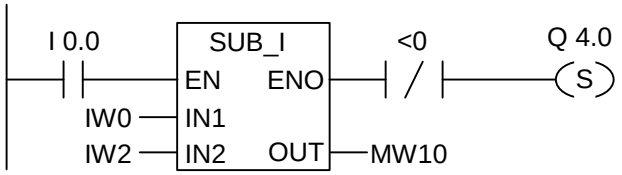
状态字

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
写：	-	-	-	-	-	x	x	x	1

实例



I0.0的信号状态为“1”时将激活该框。如果IW0的值小于IW2的值，数学运算函数“IW0 - IW2”的结果将小于“0”。如果函数得到正确执行且结果**小于**“0”，则置位Q4.0。



如果函数得到正确执行且结果**不**小于“0”，则置位Q4.0。

12.10 >=0 ---| |--- 结果位大于等于0

符号

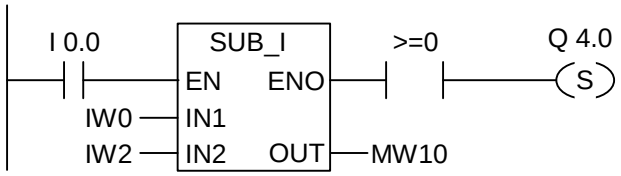
说明

>=0 ---| |---(结果位大于等于0)或**>=0 ---| / |---(**结果位取反后大于等于0)触点符号用于识别数学运算函数的结果是否大于或等于“0”。指令扫描状态字的条件代码位CC 1和CC 0，以确定与“0”的关系。串联使用时，扫描的结果将通过AND与RLO链接；并联使用时，扫描结果通过OR与RLO链接。

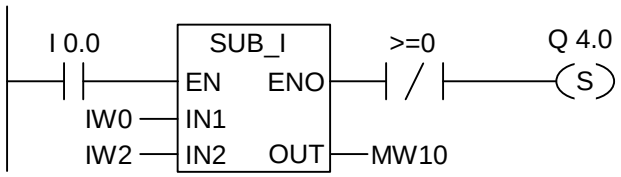
状态字

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
写:	-	-	-	-	-	x	x	x	1

实例



I0.0的信号状态为“1”时将激活该框。如果IW0的值大于或等于IW2的值，数学运算函数“IW0 - IW2”的结果将大于或等于“0”。如果函数得到正确执行且结果**大于**或等于“0”，则置位Q4.0。



如果函数得到正确执行且结果**不**大于或等于“0”，则置位Q4.0。

12.11 <=0 ---| |--- 结果位小于等于0

符号

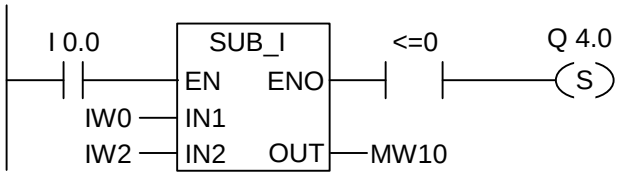
说明

<=0 ---| |---(结果位小于等于0)或**<=0 ---| / |---(**结果位取反后小于等于0)触点符号用于识别数学运算函数的结果是否小于或等于“0”。指令扫描状态字的条件代码位CC 1和CC 0，以确定结果与“0”的关系。串联使用时，扫描的结果将通过AND与RLO链接；并联使用时，扫描结果通过OR与RLO链接。

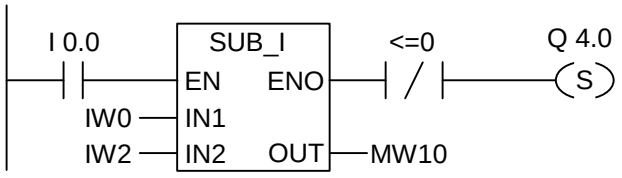
状态字

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
写：	-	-	-	-	-	x	x	x	1

实例



I0.0的信号状态为“1”时将激活该框。如果IW0的值小于或等于IW2的值，数学运算函数“IW0 - IW2”的结果将小于或等于“0”。如果函数得到正确执行且结果**小于**或等于“0”，则置位Q4.0。



如果函数得到正确执行且结果**不小于**或等于“0”，则置位Q4.0。

13 定时器指令

13.1 定时器指令概述

说明

有关设置和选择正确时间的信息，请参阅定时器在存储器中的位置与定时器组件描述。

下列定时器指令可用：

- S_PULSE 脉冲S5定时器
- S_PEXT 扩展脉冲S5定时器
- S_ODT 接通延时S5定时器
- S_ODTS 保持接通延时S5定时器
- S_OFFDT 断开延时S5定时器
- ---(SP) 脉冲定时器线圈
- ---(SE) 扩展脉冲定时器线圈
- ---(SD) 接通延时定时器线圈
- ---(SS) 保持接通延时定时器线圈
- ---(SA) 断开延时定时器线圈

13.2 存储器中定时器的位置和定时器的组件

存储器中的区域

在CPU的存储器中，有一个区域是专为定时器保留的。此存储区域为每个定时器地址保留一个16位字。梯形图逻辑指令集支持256个定时器。要确定可用的定时器字数，请参考CPU的技术信息。

下列功能可访问定时器存储区：

- 定时器指令
- 通过定时时钟更新定时器字。当CPU处于RUN模式时，此功能按以时间基准指定的时间间隔，将给定的时间值递减一个单位，直至时间值等于零。

时间值

定时器字的0到9位包含二进制编码的时间值。时间值指定单位数。时间更新操作按以时间基准指定的时间间隔，将时间值递减一个单位。递减至时间值等于零。可以用二进制、十六进制或以二进制编码的十进制(BCD)格式，将时间值装载到累加器1的低位字中。

可以使用以下任意一种格式预先装载时间值：

- W#16#wxyz
  - 其中，w = 时间基准(即时间间隔或分辨率)
  - 其中，xyz = 以二进制编码的十进制格式表示的时间值
- S5T#a**H**_b**M**_c**S**_d**MS**
  - 其中，H = 小时，M = 分钟，S = 秒，MS = 毫秒；
    a、b、c、d由用户定义。
  - 时间基准是自动选择的，数值会根据时间基准四舍五入到下一个较低数。

可以输入的最大时间值是9,990秒或2小时_46分钟_30秒。

S5TIME#4S = 4秒
s5t#2h_15m = 2小时15分钟
S5T#1H_12M_18S = 1小时12分钟18秒

时间基准

定时器字的12和13位包含二进制编码的时间基准。时间基准定义将时间值递减一个单位所用的时间间隔。最小的时间基准是10毫秒；最大的时间基准是10秒。

时间基准	时间基准的二进制编码
10毫秒	00
100毫秒	01
1秒	10
10秒	11

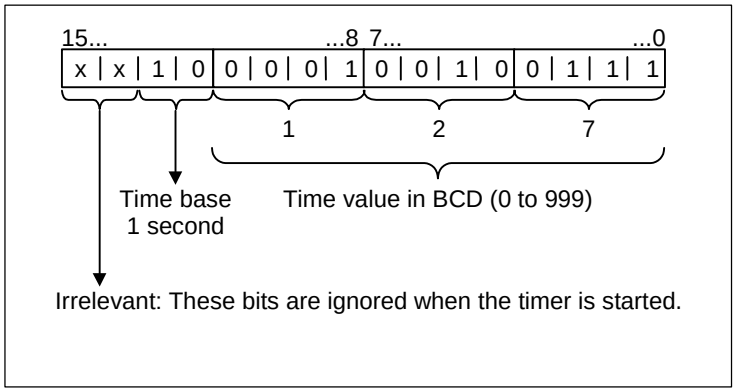
不接受超过2小时46分30秒的数值。其分辨率超出范围限制的值(例如2小时10毫秒)将被舍入到有效的分辨率。用于S5TIME的通用格式对范围和分辨率的限制如下：

分辨率	范围
0.01秒	10MS到9S_990MS
0.1秒	100MS到1M_39S_900MS
1秒	1S到16M_39S
10秒	10S到2H_46M_30S

时间单元中的位组态

定时器启动时，定时器单元的内容用作时间值。定时器单元的0到11位容纳二进制编码的十进制时间值(BCD格式：四位一组，包含一个用二进制编码的十进制值)。12和13位存储二进制编码的时间基准。

下图显示装载了时间值127，时间基准1秒的定时器单元的内容：

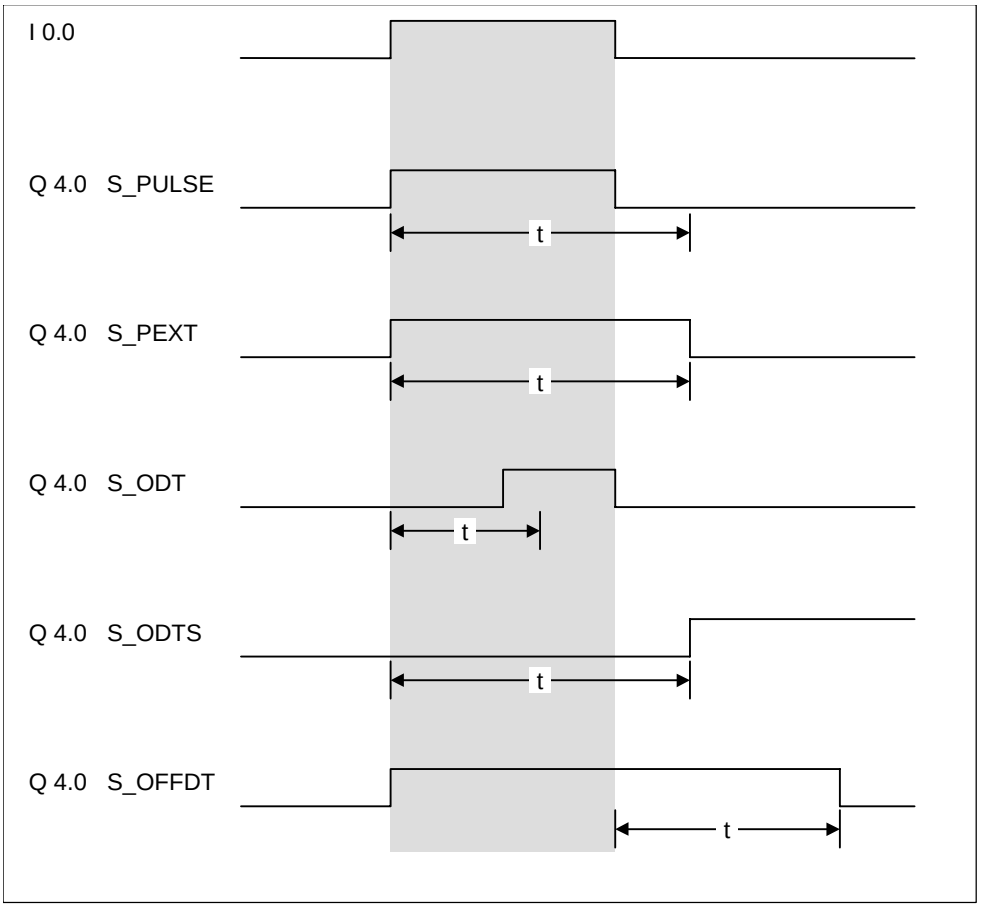


读取时间和时间基准

每个定时器逻辑框提供两种输出：BI和BCD，从中可指示一个字位置。BI输出提供二进制格式的时间值。BCD输出提供二进制编码的十进制(BCD)格式的时间基准和时间值。

选择合适的定时器

本概述用于帮助您为定时作业选择合适的定时器。

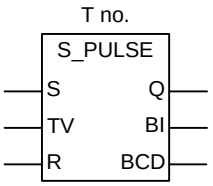


定时器	说明
S_PULSE 脉冲定时器	输出信号保持为1的最大时间与设定的时间值t相同。如果输入信号变为0，则输出信号在较短的时间内保持为1。
S_PEXT 扩展脉冲定时器	输出信号在设定的时间长度内保持为1，无论输入信号保持1多长时间。
S_ODT 接通延时定时器	只有在设定的时间已过且输入信号仍为1时，输出信号才变为1。
S_ODTS 保持接通延时定时器	只有在设定的时间已过时，输出信号才从0变为1，无论输入信号保持1多长时间。
S_OFFDT 断开延时定时器	输入信号变为1或定时器运行时，输出信号变为1。输入信号从1变为0时，时间启动。

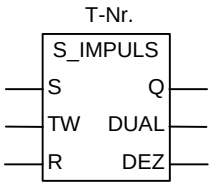
13.3 S_PULSE脉冲S5定时器

符号

English



German



参数 英语	参数 德语	数据类型	内存区域	说明
T 编号	T-Nr.	TIMER	表格	定时器标识号，范围取决于CPU
S	S	BOOL	I、Q、M、L、D	开始输入
TV	TW	S5TIME	I、Q、M、L、D	预设时间值
R	R	BOOL	I、Q、M、L、D	复位输入
BI	DUAL	WORD	I、Q、M、L、D	剩余时间值，整型格式
BCD	DEZ	WORD	I、Q、M、L、D	剩余时间值，BCD格式
Q	Q	BOOL	I、Q、M、L、D	定时器的状态

说明

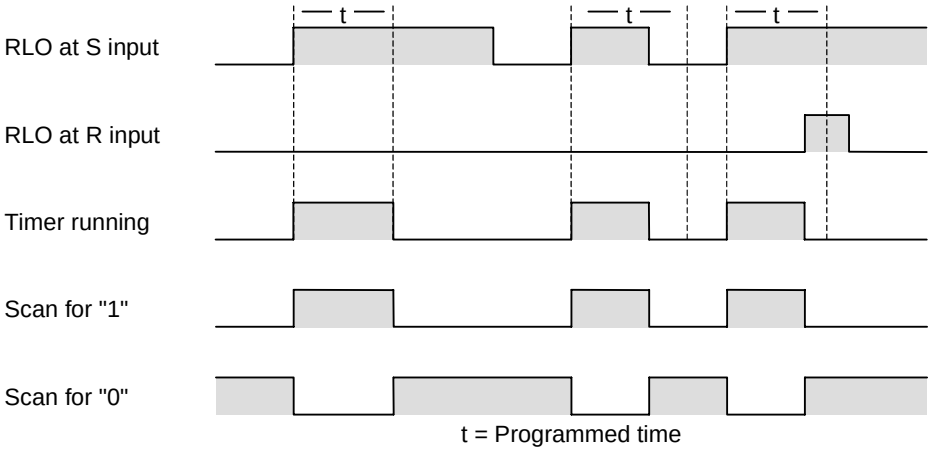
如果在启动(S)输入端有一个上升沿，**S_PULSE**(脉冲S5定时器)将启动指定的定时器。信号变化始终是启用定时器的必要条件。定时器在输入端S的信号状态为“1”时运行，但最长周期是由输入端TV指定的时间值。只要定时器运行，输出端Q的信号状态就为“1”。如果在时间间隔结束前，S输入端从“1”变为“0”，则定时器将停止。这种情况下，输出端Q的信号状态为“0”。

如果在定时器运行期间定时器复位(R)输入从“0”变为“1”时，则定时器将被复位。当前时间和时间基准也被设置为零。如果定时器不是正在运行，则定时器R输入端的逻辑“1”没有任何作用。

可在输出端BI和BCD上扫描当前时间值。时间值在BI端是二进制编码，在BCD端是BCD编码。当前时间值为初始TV值减去定时器启动后经过的时间。

时序图

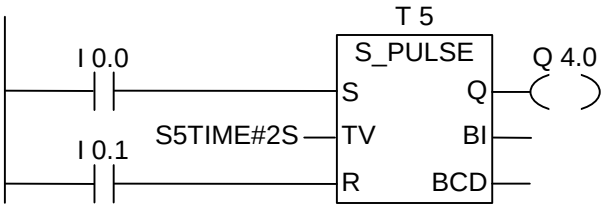
脉冲定时器特征：



状态字

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
写：	-	-	-	-	-	x	x	x	1

实例



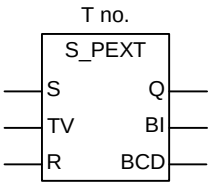
如果输入端I0.0的信号状态从“0”变为“1”(RLO中的上升沿)，则定时器T5将启动。只要I0.0为“1”，定时器就将继续运行指定的两秒(2s)时间。如果定时器达到预定时间前，I0.0的信号状态从“1”变为“0”，则定时器将停止。如果输入端I0.1的信号状态从“0”变为“1”，而定时器仍在运行，则时间复位。

只要定时器运行，输出端Q4.0就是逻辑“1”，如果定时器预设时间结束或复位，则输出端Q4.0变为“0”。

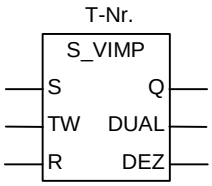
13.4 S_PEXT扩展脉冲S5定时器

符号

English



German



参数 英语	参数 德语	数据类型	内存区域	说明
T 编号	T-Nr.	TIMER	表格	定时器标识号，范围取决于CPU
S	S	BOOL	I、Q、M、L、D	开始输入
TV	TW	S5TIME	I、Q、M、L、D	预设时间值
R	R	BOOL	I、Q、M、L、D	复位输入
BI	DUAL	WORD	I、Q、M、L、D	剩余时间值，整型格式
BCD	DEZ	WORD	I、Q、M、L、D	剩余时间值，BCD格式
Q	Q	BOOL	I、Q、M、L、D	定时器的状态

说明

如果在启动(S)输入端有一个上升沿，**S_PEXT**(扩展脉冲S5定时器)将启动指定的定时器。信号变化始终是启用定时器的必要条件。定时器以在输入端TV指定的预设时间间隔运行，即使在时间间隔结束前，S输入端的信号状态变为“0”。只要定时器运行，输出端Q的信号状态就为“1”。如果在定时器运行期间输入端S的信号状态从“0”变为“1”，则将使用预设的时间值重新启动(“重新触发”)定时器。

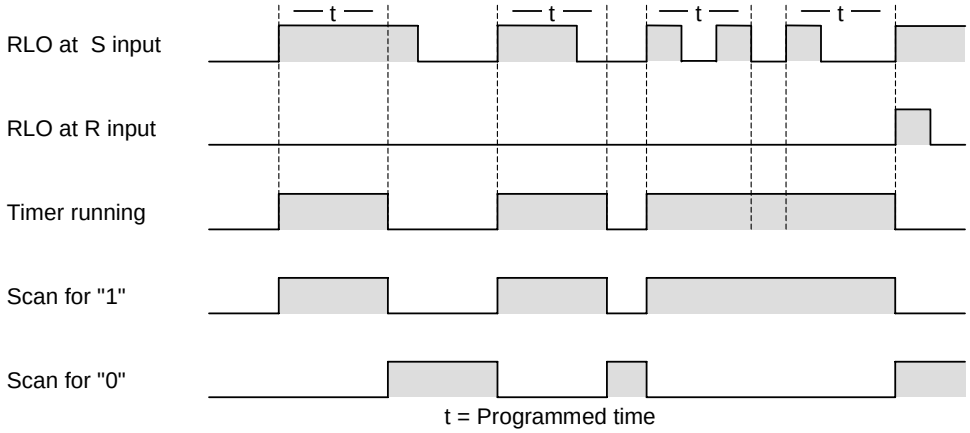
如果在定时器运行期间复位(R)输入从“0”变为“1”，则定时器复位。当前时间和时间基准被设置为零。

可在输出端BI和BCD上扫描当前时间值。时间值在BI处为二进制编码，在BCD处为BCD编码。当前时间值为初始TV值减去定时器启动后经过的时间。

参见定时器在存储器中的位置与定时器组件。

时序图

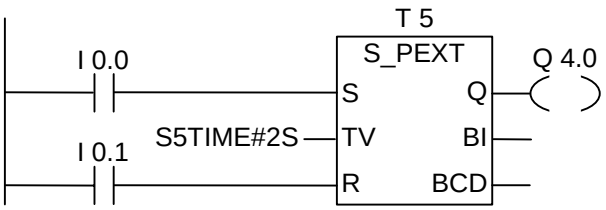
扩展脉冲定时器特征曲线：



状态字

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
写：	-	-	-	-	-	x	x	x	1

实例

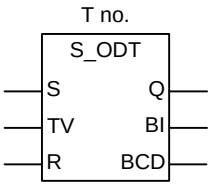


如果输入端I0.0的信号状态从“0”变为“1”(RLO中的上升沿)，则定时器T5将启动。定时器将继续运行指定的两秒(2秒)时间，而不会受到输入端S处下降沿的影响。如果在定时器达到预定时间前I0.0的信号状态从“0”变为“1”，则定时器将被重新触发。只要定时器运行，输出端Q4.0就为逻辑“1”。

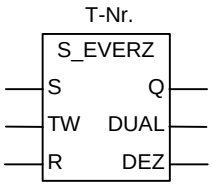
13.5 S_ODT接通延时S5定时器

符号

English



German



参数 英语	参数 德语	数据类型	内存区域	说明
T 编号	T-Nr.	TIMER	表格	定时器标识号，范围取决于CPU
S	S	BOOL	I、Q、M、L、D	开始输入
TV	TW	S5TIME	I、Q、M、L、D	预设时间值
R	R	BOOL	I、Q、M、L、D	复位输入
BI	DUAL	WORD	I、Q、M、L、D	剩余时间值，整型格式
BCD	DEZ	WORD	I、Q、M、L、D	剩余时间值，BCD格式
Q	Q	BOOL	I、Q、M、L、D	定时器的状态

说明

如果在启动(S)输入端有一个上升沿，**S_ODT**(接通延时S5定时器)将启动指定的定时器。信号变化始终是启用定时器的必要条件。只要输入端S的信号状态为正，定时器就以在输入端TV指定的时间间隔运行。定时器达到指定时间而没有出错，并且S输入端的信号状态仍为“1”时，输出端Q的信号状态为“1”。如果定时器运行期间输入端S的信号状态从“1”变为“0”，定时器将停止。这种情况下，输出端Q的信号状态为“0”。

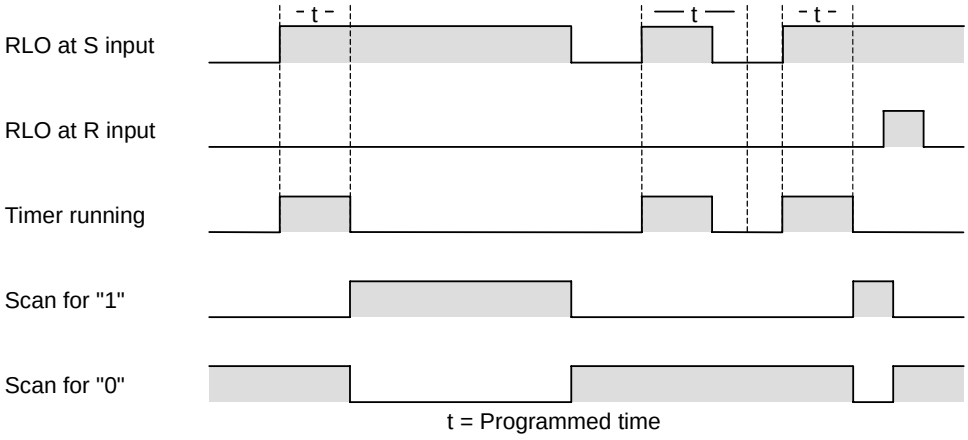
如果在定时器运行期间复位(R)输入从“0”变为“1”，则定时器复位。当前时间和时间基准被设置为零。然后，输出端Q的信号状态变为“0”。如果在定时器没有运行时R输入端有一个逻辑“1”，并且输入端S的RLO为“1”，则定时器也复位。

可在输出端BI和BCD上扫描当前时间值。时间值在BI处为二进制编码，在BCD处为BCD编码。当前时间值为初始TV值减去定时器启动后经过的时间。

参见定时器在存储器中的位置与定时器组件。

时序图

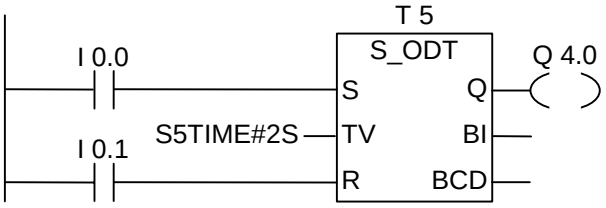
接通延时定时器特征曲线：



状态字

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
写：	-	-	-	-	-	x	x	x	1

实例

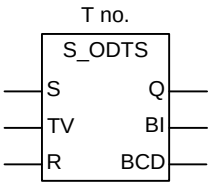


如果I0.0的信号状态从“0”变为“1”(RLO中的上升沿)，则定时器T5将启动。如果指定的两秒时间结束并且输入端I0.0的信号状态仍为“1”，则输出端Q4.0将为“1”。如果I0.0的信号状态从“1”变为“0”，则定时器停止，并且Q4.0将为“0”(如果I0.1的信号状态从“0”变为“1”，则无论定时器是否运行，时间都复位)。

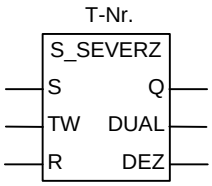
13.6 S_ODTS保持接通延时S5定时器

符号

English



German



参数 英语	参数 德语	数据类型	内存区域	说明
T 编号	T-Nr.	TIMER	表格	定时器标识号，范围取决于CPU
S	S	BOOL	I、Q、M、L、D	开始输入
TV	TW	S5TIME	I、Q、M、L、D	预设时间值
R	R	BOOL	I、Q、M、L、D	复位输入
BI	DUAL	WORD	I、Q、M、L、D	剩余时间值，整型格式
BCD	DEZ	WORD	I、Q、M、L、D	剩余时间值，BCD格式
Q	Q	BOOL	I、Q、M、L、D	定时器的状态

说明

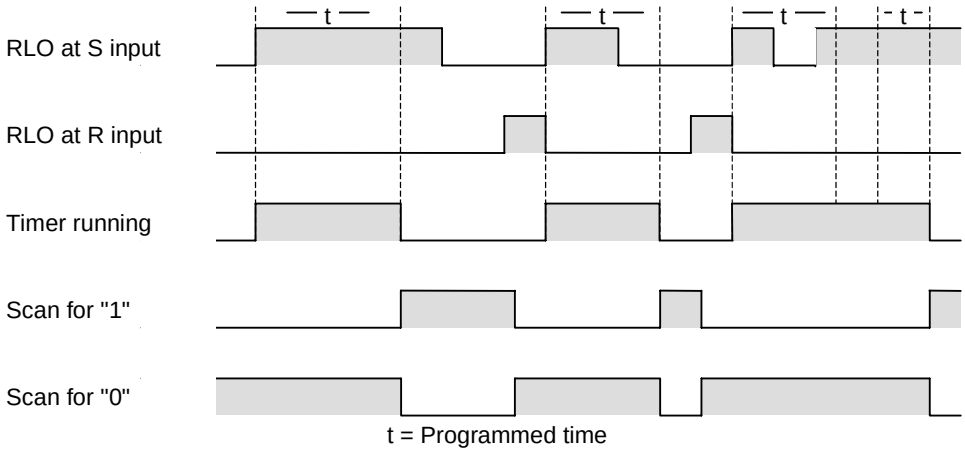
如果在启动(S)输入端有一个上升沿，**S_ODTS**(保持接通延时S5定时器)将启动指定的定时器。信号变化始终是启用定时器的必要条件。定时器以在输入端TV指定的时间间隔运行，即使在时间间隔结束前，输入端S的信号状态变为“0”。定时器预定时间结束时，输出端Q的信号状态为“1”，而无论输入端S的信号状态如何。如果在定时器运行时输入端S的信号状态从“0”变为“1”，则定时器将以指定的时间重新启动(重新触发)。

如果复位(R)输入从“0”变为“1”，则无论S输入端的RLO如何，定时器都将复位。然后，输出端Q的信号状态变为“0”。

可在输出端BI和BCD上扫描当前时间值。时间值在BI端是二进制编码，在BCD端是BCD编码。当前时间值为初始TV值减去定时器启动后经过的时间。

时序图

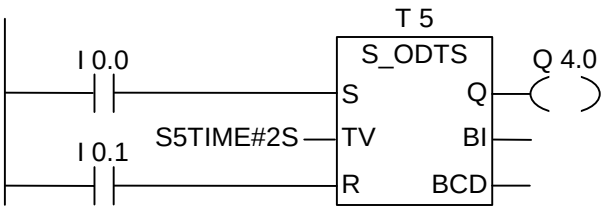
保持接通延时定时器特征曲线：



状态字

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
写：	-	-	-	-	-	x	x	x	1

实例

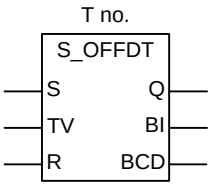


如果I0.0的信号状态从“0”变为“1”(RLO中的上升沿)，则定时器T5将启动。无论I0.0的信号是否从“1”变为“0”，定时器都将运行。如果在定时器达到指定时间前，I0.0的信号状态从“0”变为“1”，则定时器将重新触发。如果定时器达到指定时间，则输出端Q4.0将变为“1”。(如果输入端I0.1的信号状态从“0”变为“1”，则无论S处的RLO如何，时间都将复位。)

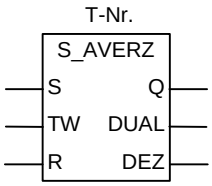
13.7 S_OFFDT断开延时S5定时器

符号

English



German



参数 英语	参数 德语	数据类型	内存区域	说明
T 编号	T-Nr.	TIMER	表格	定时器标识号，范围取决于CPU
S	S	BOOL	I、Q、M、L、D	开始输入
TV	TW	S5TIME	I、Q、M、L、D	预设时间值
R	R	BOOL	I、Q、M、L、D	复位输入
BI	DUAL	WORD	I、Q、M、L、D	剩余时间值，整型格式
BCD	DEZ	WORD	I、Q、M、L、D	剩余时间值，BCD格式
Q	Q	BOOL	I、Q、M、L、D	定时器的状态

说明

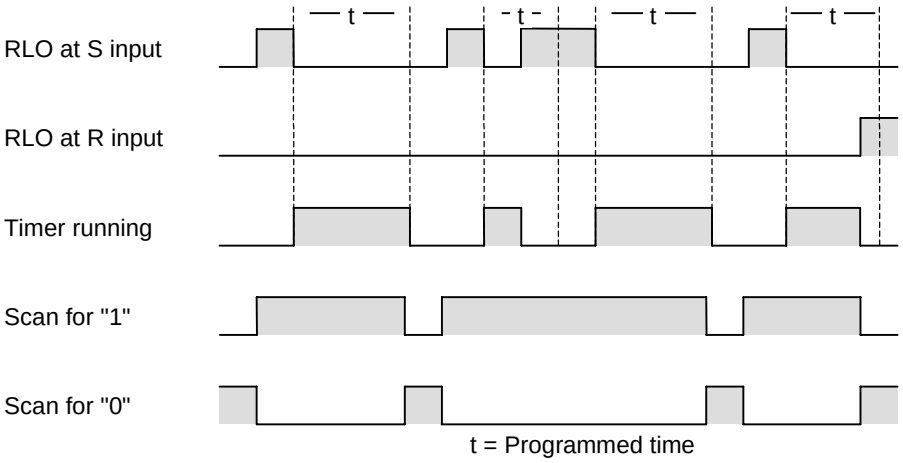
如果在启动(S)输入端有一个下降沿，**S_OFFDT**(断开延时S5定时器)将启动指定的定时器。信号变化始终是启用定时器的必要条件。如果S输入端的信号状态为“1”，或定时器正在运行，则输出端Q的信号状态为“1”。如果在定时器运行期间输入端S的信号状态从“0”变为“1”时，定时器将复位。输入端S的信号状态再次从“1”变为“0”后，定时器才能重新启动。

如果在定时器运行期间复位(R)输入从“0”变为“1”时，定时器将复位。

可在输出端BI和BCD上扫描当前时间值。时间值在BI端是二进制编码，在BCD端是BCD编码。当前时间值为初始TV值减去定时器启动后经过的时间。

时序图

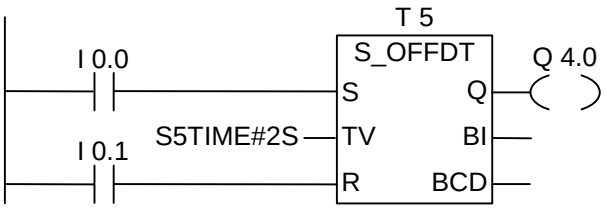
断开延时定时器特征曲线：



状态字

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
写：	-	-	-	-	-	x	x	x	1

实例



如果I0.0的信号状态从“1”变为“0”，则定时器启动。

I0.0为“1”或定时器运行时，Q4.0为“1”。(如果在定时器运行期间I0.1的信号状态从“0”变为“1”，则定时器复位)。

13.8 ---(SP)脉冲定时器线圈

符号

英语	**德语**
<T 编号.>	<T 编号>
---(SP)	**---(SI)**
<时间值>	<时间值>

参数	**数据类型**	**内存区域**	**说明**
<T 编号>	TIMER	表格	定时器标识号，范围取决于CPU
<时间值>	S5TIME	I、Q、M、L、D	预设时间值

说明

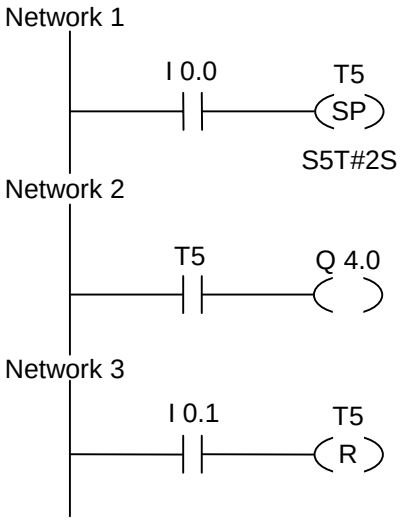
如果RLO状态有一个上升沿，**---(SP)(**脉冲定时器线圈)将以该**<时间值>**启动指定的定时器。只要RLO保持正值(“1”)，定时器就继续运行指定的时间间隔。只要定时器运行，计数器的信号状态就为“1”。如果在达到时间值前，RLO中的信号状态从“1”变为“0”，则定时器将停止。这种情况下，对于“1”的扫描始终产生结果“0”。

参见“存储器中定时器的位置和定时器的组件”和 S_PULSE(脉冲S5定时器)。

状态字

	BR	**CC 1**	**CC 0**	**OV**	**OS**	**OR**	**STA**	**RLO**	**/FC**
写:	-	-	-	-	-	0	-	-	0

实例



如果输入端I0.0的信号状态从“0”变为“1”(RLO中的上升沿)，则定时器T5启动。只要输入端I0.0的信号状态为“1”，定时器就继续运行指定的两秒时间。如果在指定的时间结束前输入端I0.0的信号状态从“1”变为“0”，则定时器停止。

只要定时器运行，输出端Q4.0的信号状态就为“1”。如果输入端I0.1的信号状态从“0”变为“1”，定时器T5将复位，定时器停止，并将时间值的剩余部分清为“0”。

13.9 ---(SE)扩展脉冲定时器线圈

符号

英语	德语
<T 编号>	<T no>
---(SE)	**---(SV)**
<时间值>	<时间值>

参数	数据类型	内存区域	说明
<T 编号>	TIMER	表格	定时器标识号，范围取决于CPU
<时间值>	S5TIME	I、Q、M、L、D	预设时间值

说明

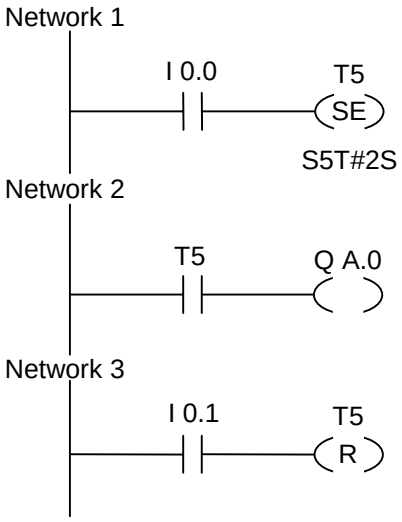
如果RLO状态有一个上升沿，**---(SE)(**扩展脉冲定时器线圈)将以指定的**<时间值>**启动指定的定时器。定时器继续运行指定的时间间隔，即使定时器达到指定时间前RLO变为“0”。只要定时器运行，计数器的信号状态就为“1”。如果在定时器运行期间RLO从“0”变为“1”，则将以指定的时间值重新启动定时器(重新触发)。

参见“存储器中定时器的位置和定时器的组件”和 S_PEXT(扩展脉冲S5定时器)。

状态字

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
写:	-	-	-	-	-	0	-	-	0

实例



如果输入端I0.0的信号状态从“0”变为“1”(RLO中的上升沿)，则定时器T5启动。定时器继续运行，而无论RLO是否出现下降沿。如果在定时器达到指定时间前I0.0的信号状态从“0”变为“1”，则定时器重新触发。

只要定时器运行，输出端Q4.0的信号状态就为“1”。如果输入端I0.1的信号状态从“0”变为“1”，定时器T5将复位，定时器停止，并将时间值的剩余部分清为“0”。

13.10 ---(SD)接通延时定时器线圈

符号

英语	**德语**
<T 编号>	<T 编号>
---(SD)	**---(SE)**
<时间值>	<时间值>

参数	**数据类型**	**内存区域**	**说明**
<T 编号>	TIMER	表格	定时器标识号，范围取决于CPU
<时间值>	S5TIME	I、Q、M、L、D	预设时间值

说明

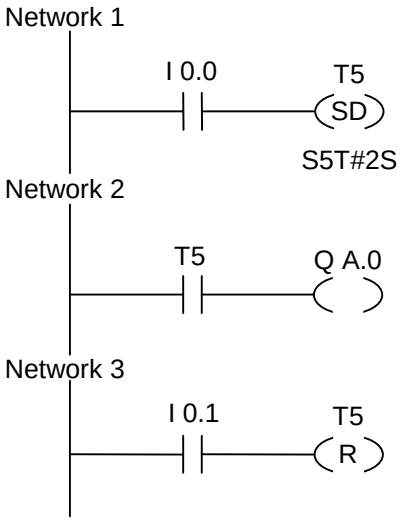
如果RLO状态有一个上升沿，**---(SD)(**接通延时定时器线圈)将以该**<时间值>**启动指定的定时器。如果达到该**<时间值>**而没有出错，且RLO仍为“1”，则定时器的信号状态为“1”。如果在定时器运行期间RLO从“1”变为“0”，则定时器复位。这种情况下，对于“1”的扫描始终产生结果“0”。

参见“存储器中定时器的位置和定时器的组件”和 S_ODT(接通延时S5定时器)。

状态字

	BR	**CC 1**	**CC 0**	**OV**	**OS**	**OR**	**STA**	**RLO**	**/FC**
写:	-	-	-	-	-	0	-	-	0

实例



如果输入端I0.0的信号状态从“0”变为“1”(RLO中的上升沿)，则定时器T5启动。如果指定时间结束而输入端I0.0的信号状态仍为“1”，则输出端Q4.0的信号状态将为“1”。

如果输入端I0.0的信号状态从“1”变为“0”，则定时器保持空闲，并且输出端Q4.0的信号状态将为“0”。如果输入端I0.1的信号状态从“0”变为“1”，定时器T5将复位，定时器停止，并将时间值的剩余部分清为“0”。

13.11 ---(SS)保持接通延时定时器线圈

符号

英语	**德语**
<T 编号>	<T 编号>
---(SS)	**---(SS)**
<时间值>	<时间值>

参数	**数据类型**	**内存区域**	**说明**
<T 编号>	TIMER	表格	定时器标识号，范围取决于CPU
<时间值>	S5TIME	I、Q、M、L、D	预设时间值

说明

如果RLO状态有一个上升沿，**---(SS)**(保持接通延时定时器线圈)将启动指定的定时器。如果达到时间值，定时器的信号状态为“1”。只有明确进行复位，定时器才可能重新启动。只有复位才能将定时器的信号状态设为“0”。

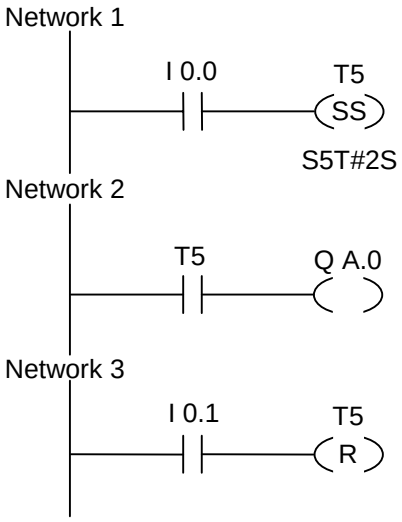
如果在定时器运行期间RLO从“0”变为“1”，则定时器以指定的时间值重新启动。

参见“存储器中定时器的位置和定时器的组件”和 S_ODTS(保持接通延时S5定时器)。

状态字

	BR	**CC 1**	**CC 0**	**OV**	**OS**	**OR**	**STA**	**RLO**	**/FC**
写:	-	-	-	-	-	0	-	-	0

实例



如果输入端I0.0的信号状态从“0”变为“1”(RLO中的上升沿)，则定时器T5启动。如果在定时器达到指定时间前输入端I0.0的信号状态从“0”变为“1”，则定时器将重新触发。如果定时器达到指定时间，则输出端Q4.0将变为“1”。输入端I0.1的信号状态“1”将复位定时器T5，使定时器停止，并将时间值的剩余部分清为“0”。

13.12 ---(SF)断开延时定时器线圈

符号

英语	**德语**
<T 编号>	<T 编号>
---(SF)	**---(SA)**
<时间值>	<时间值>

参数	**数据类型**	**内存区域**	**说明**
<T 编号>	TIMER	表格	定时器标识号，范围取决于CPU
<时间值>	S5TIME	I、Q、M、L、D	预设时间值

说明

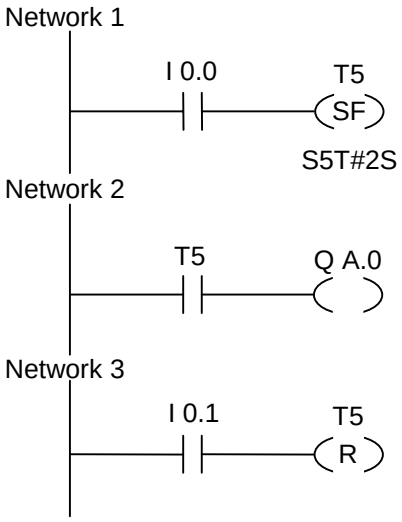
如果RLO状态有一个下降沿，**---(SF)**(断开延时定时器线圈)将启动指定的定时器。当RLO为“1”时或只要定时器在**<时间值>**时间间隔内运行，定时器就为“1”。如果在定时器运行期间RLO从“0”变为“1”，则定时器复位。只要RLO从“1”变为“0”，定时器即会重新启动。

参见“存储器中定时器的位置和定时器的组件”和 S_OFFDT(断开延时S5定时器)。

状态字

	BR	**CC 1**	**CC 0**	**OV**	**OS**	**OR**	**STA**	**RLO**	**/FC**
写:	-	-	-	-	-	0	-	-	0

实例



如果输入端I0.0的信号状态从“1”变为“0”，则定时器启动。

如果输入端I0.0为“1”或定时器正在运行，则输出端Q4.0的信号状态为“1”。如果输入端I0.1的信号状态从“0”变为“1”，定时器T5将复位，定时器停止，并将时间值的剩余部分清为“0”。

14 字逻辑指令

14.1 字逻辑指令概述

说明

字逻辑指令按照布尔逻辑按位比较字(16位)和双字(32位)对。

如果输出OUT的结果不等于0，将把状态字的CC 1位设置为“1”。

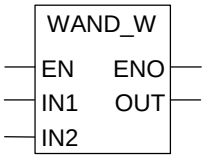
如果输出OUT的结果等于0，将把状态字的CC 1位设置为“0”。

可以使用下列字逻辑指令：

- WAND_W (字)单字与运算
- WOR_W (字)单字或运算
- WXOR_W (字)单字异或运算
- WAND_DW (字)双字与运算
- WOR_DW (字)双字或运算
- WXOR_DW (字)双字异或运算

14.2 WAND_W(字)单字与运算

符号



参数	数据类型	内存区域	说明
EN	BOOL	I、Q、M、L、D	使能输出
ENO	BOOL	I、Q、M、L、D	使能输出
IN1	WORD	I、Q、M、L、D	逻辑运算的第一个值
IN2	WORD	I、Q、M、L、D	逻辑运算的第二个值
OUT	WORD	I、Q、M、L、D	逻辑运算的结果字

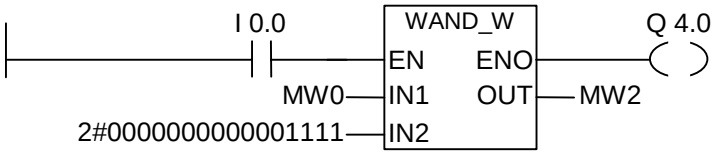
说明

使能(EN)输入的信号状态为“1”时将激活**WAND_W**(字与运算)，并逐位对IN1和IN2处的两个字值进行与运算。按纯位模式来解释这些值。可以在输出OUT处扫描结果。ENO与EN的逻辑状态相同。

状态字

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
写：	1	x	0	0	-	x	1	1	1

实例



如果I0.0为“1”，则执行指令。在MW0的位中，只有0至3位是相关的，其余位被IN2字位模式屏蔽：

MW0 = 01010101 01010101

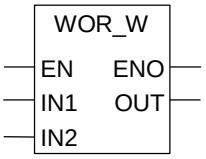
IN2 = 00000000 00001111

MW0 AND IN2 = MW2 = 00000000 00000101

如果执行了指令，则Q4.0为“1”。

14.3 WOR_W(字)单字或运算

符号



参数	数据类型	内存区域	说明
EN	BOOL	I、Q、M、L、D	使能输出
ENO	BOOL	I、Q、M、L、D	使能输出
IN1	WORD	I、Q、M、L、D	逻辑运算的第一个值
IN2	WORD	I、Q、M、L、D	逻辑运算的第二个值
OUT	WORD	I、Q、M、L、D	逻辑运算的结果字

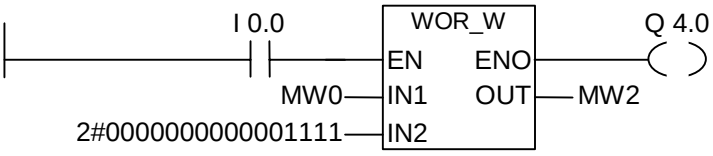
说明

使能(EN)输入的信号状态为“1”时将激活**WOR_W**(单字或运算)，并逐位对IN1和IN2处的两个字值进行或运算。按纯位模式来解释这些值。可以在输出OUT处扫描结果。ENO与EN的逻辑状态相同。

状态字

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
写：	1	x	0	0	-	x	1	1	1

实例



如果I0.0为“1”，则执行指令。将位0至3设置为“1”，不改变MW0的所有其它位。

MW0 = 01010101 01010101

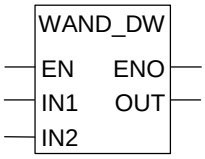
IN2 = 00000000 00001111

MW0 OR IN2=MW2 = 01010101 01011111

如果执行了指令，则Q4.0为“1”。

14.4 WAND_DW(字)双字与运算

符号



参数	数据类型	内存区域	说明
EN	BOOL	I、Q、M、L、D	使能输出
ENO	BOOL	I、Q、M、L、D	使能输出
IN1	DWORD	I、Q、M、L、D	逻辑运算的第一个值
IN2	DWORD	I、Q、M、L、D	逻辑运算的第二个值
OUT	DWORD	I、Q、M、L、D	逻辑运算的结果双字

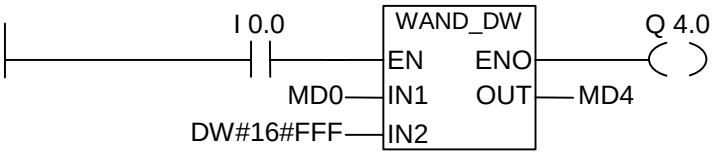
说明

使能(EN)输入的信号状态为“1”时将激活**WAND_DW(**双字与运算)，并逐位对IN1和IN2处的两个字值进行与运算。按纯位模式来解释这些值。可以在输出OUT处扫描结果。ENO与EN的逻辑状态相同。

状态字

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
写:	1	x	0	0	-	x	1	1	1

实例



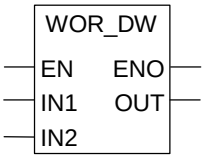
如果I0.0为“1”，则执行指令。在MD0的位中，只有0和11位是相关的，其余位被IN2位模式屏蔽：

MD0	=	01010101 01010101 01010101 01010101
IN2	=	00000000 00000000 00001111 11111111
MD0 AND IN2 = MD4	=	00000000 00000000 00000101 01010101

如果执行了指令，则Q4.0为“1”。

14.5 WOR_DW(字)双字或运算

符号



参数	数据类型	内存区域	说明
EN	BOOL	I、Q、M、L、D	使能输出
ENO	BOOL	I、Q、M、L、D	使能输出
IN1	DWORD	I、Q、M、L、D	逻辑运算的第一个值
IN2	DWORD	I、Q、M、L、D	逻辑运算的第二个值
OUT	DWORD	I、Q、M、L、D	逻辑运算的结果双字

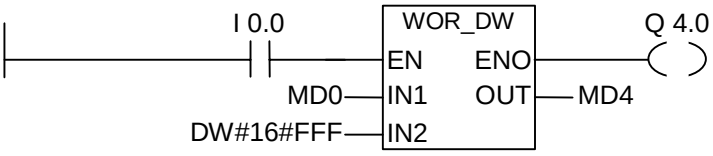
说明

使能(EN)输入的信号状态为“1”时将激活**WOR_DW**(双字或运算)，并逐位对IN1和IN2处的两个字值进行或运算。按纯位模式来解释这些值。可以在输出OUT处扫描结果。ENO与EN的逻辑状态相同。

状态字

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
写：	1	x	0	0	-	x	1	1	1

实例



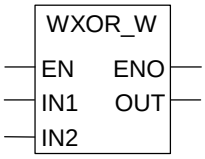
如果I0.0为“1”，则执行指令。将位0至11设置为“1”，不改变MD0的其余位：

MD0	=	01010101 01010101 01010101 01010101
IN2	=	00000000 00000000 00001111 11111111
MD0 OR IN2 = MD4	=	01010101 01010101 01011111 11111111

如果执行了指令，则Q4.0为“1”。

14.6 WXOR_W(字)单字异或运算

符号



参数	数据类型	内存区域	说明
EN	BOOL	I、Q、M、L、D	使能输出
ENO	BOOL	I、Q、M、L、D	使能输出
IN1	WORD	I、Q、M、L、D	逻辑运算的第一个值
IN2	WORD	I、Q、M、L、D	逻辑运算的第二个值
OUT	WORD	I、Q、M、L、D	逻辑运算的结果字

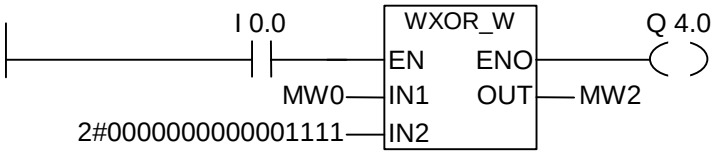
说明

使能(EN)输入的信号状态为“1”时将激活**WXOR_W**(单字异或运算)，并逐位对IN1和IN2处的两个字值进行异或运算。按纯位模式来解释这些值。可以在输出OUT处扫描结果。ENO与EN的逻辑状态相同。

状态字

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
写：	1	x	0	0	-	x	1	1	1

实例



如果I0.0为“1”，则执行指令：

MW0 = 01010101 01010101

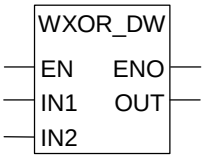
IN2 = 00000000 00001111

MW0 XOR IN2 = MW2 = 01010101 01011010

如果执行了指令，则Q4.0为“1”。

14.7 WXOR_DW(字)双字异或运算

符号



参数	数据类型	内存区域	说明
EN	BOOL	I、Q、M、L、D	使能输出
ENO	BOOL	I、Q、M、L、D	使能输出
IN1	DWORD	I、Q、M、L、D	逻辑运算的第一个值
IN2	DWORD	I、Q、M、L、D	逻辑运算的第二个值
OUT	DWORD	I、Q、M、L、D	逻辑运算的结果双字

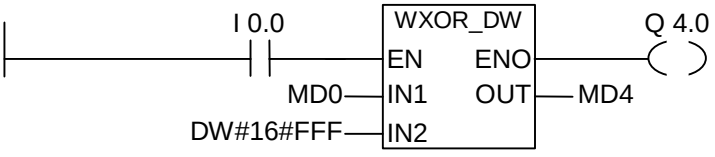
说明

使能(EN)输入的信号状态为“1”时将激活**WXOR_DW(**双字异或运算)，并逐位对IN1和IN2处的两个字值进行异或运算。按纯位模式来解释这些值。可以在输出OUT处扫描结果。ENO与EN的逻辑状态相同。

状态字

	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
写:	1	x	0	0	-	x	1	1	1

实例



如果I0.0为“1”，则执行指令：

MD0	=	01010101 01010101 01010101 01010101
IN2	=	00000000 00000000 00001111 11111111
MW2 = MD0 XOR IN2	=	01010101 01010101 01011010 10101010

如果执行了指令，则Q4.0为“1”。

A 所有LAD指令总览

A.1 按英语助记符(国际)排序的LAD指令

英语 助记符	德语 助记符	程序元素目录	
---| |---	---| |---	位逻辑指令	常开触点(地址)
---|/|---	---|/|---	位逻辑指令	常闭触点(地址)
---( )	---( )	位逻辑指令	输出线圈
---(#)---	---(#)---	位逻辑指令	中间输出
==0 ---| |---	==0 ---| |---	状态位	结果位等于0
>0 ---| |---	>0 ---| |---	状态位	结果位大于0
>=0 ---| |---	>=0 ---| |---	状态位	结果位大于等于0
<=0 ---| |---	<=0 ---| |---	状态位	结果位小于等于0
<0 ---| |---	<0 ---| |---	状态位	结果位小于0
<>0 ---| |---	<>0 ---| |---	状态位	结果位不等于0
ABS	ABS	浮点数指令	得到浮点数数字的绝对值
ACOS	ACOS	浮点数指令	得到反余弦值
ADD_DI	ADD_DI	整数数学运算指令	加双精度整数
ADD_I	ADD_I	整型数学运算指令	加整数
ADD_R	ADD_R	浮点数指令	加实数
ASIN	ASIN	浮点数指令	得到反正弦值
ATAN	ATAN	浮点数指令	得到反正切值
BCD_DI	BCD_DI	转换	BCD码转换为双精度整数
BCD_I	BCD_I	转换	BCD码转换为整数
BR ---| |---	BIE ---| |---	状态位	异常位二进制结果
----(CALL)	----(CALL)	程序控制	调用来自线圈的FC SFC(不带参数)
CALL_FB	CALL_FB	程序控制	从逻辑框中调用FB
CALL_FC	CALL_FC	程序控制	从逻辑框中调用FC
CALL_SFB	CALL_SFB	程序控制	从逻辑框中调用系统FB

英语 助记符	德语 助记符	程序元素目录	
CALL_SFC	CALL_SFC	程序控制	从逻辑框中调用系统FC
----(CD)	----(ZR)	计数器	减计数器线圈
CEIL	CEIL	转换	上限
CMP >=D	CMP >=D	比较	比较双精度整数（==、<>、>、<、>=、<=)
CMP >=I	CMP >=I	比较	比较整数（==、<>、>、<、>=、<=)
CMP >=R	CMP >=R	比较	比较实数（==、<>、>、<、>=、<=)
COS	COS	浮点数指令	得到余弦值
----(CU)	---(ZV)	计数器	升值计数器线圈
DI_BCD	DI_BCD	转换	双精度整数转换为BCD码
DI_R	DI_R	转换	双精度整数转换为浮点数
DIV_DI	DIV_DI	整数数学运算指令	除双精度整数
DIV_I	DIV_I	整数数学运算指令	除整数
DIV_R	DIV_R	浮点数指令	除实数
EXP	EXP	浮点数指令	得到指数值
FLOOR	FLOOR	转换	基数
I_BCD	I_BCD	转换	整数转换为BCD码
I_DI	I_DI	转换	整数转换为双精度整数
INV_I	INV_I	转换	二进制反码整数
INV_DI	INV_DI	转换	二进制反码双精度整数
---(JMP)	---(JMP)	跳转	无条件跳转
---(JMP)	---(JMP)	跳转	有条件跳转
---(JMPN)	---(JMPN)	跳转	如果非则跳转
LABEL	LABEL	跳转	标号
LN	LN	浮点数指令	得到自然对数
---(MCR>)	---(MCR>)	程序控制	主控制继电器关闭
---(MCR<)	---(MCR<)	程序控制	主控制继电器开启
---(MCRA)	---(MCRA)	程序控制	主控制继电器激活
---(MCRD)	---(MCRD)	程序控制	主控制继电器取消激活
MOD_DI	MOD_DI	整数数学运算指令	返回分数双精度整数
MOVE	MOVE	传送	分配值
MUL_DI	MUL_DI	整数数学运算指令	乘双精度整数
MUL_I	MUL_I	整数数学运算指令	乘整数
MUL_R	MUL_R	浮点数指令	乘实数
---(N)---	---(N)---	位逻辑指令	RLO负跳沿检测

英语 助记符	德语 助记符	程序元素目录	
NEG	NEG	位逻辑指令	地址下降沿检测
NEG_DI	NEG_DI	转换	二进制补码双精度整数
NEG_I	NEG_I	转换	二进制补码整数
NEG_R	NEG_R	转换	取反浮点数数字
---| NOT |---	---| NOT |---	位逻辑指令	取反能流
---(OPN)	---(OPN)	DB调用	打开数据块：DB或DI
OS ---| |---	OS ---| |---	状态位	存储的异常位溢出
OV ---| |---	OV ---| |---	状态位	异常位溢出
---(P)---	---(P)---	位逻辑指令	RLO正跳沿检测
POS	POS	位逻辑指令	地址上升沿检测
---(R)	---(R)	位逻辑指令	复位线圈
---(RET)	---(RET)	程序控制	返回
ROL_DW	ROL_DW	移位/循环	循环左移双字
ROR_DW	ROR_DW	移位/循环	循环右移双字
ROUND	ROUND	转换	取整为双精度整数
RS	RS	位逻辑指令	复位置位触发器
---(S)	---(S)	位逻辑指令	置位线圈
---(SAVE)	---(SAVE)	位逻辑指令	将RLO的状态保存到BR
---(SC)	---(SZ)	计数器	设置计数器值
S_CD	Z_RUECK	计数器	减计数器
S_CU	Z_VORW	计数器	增计数器
S_CUD	ZAEHLER	计数器	双向计数器
---(SD)	---(SE)	定时器	接通延时定时器线圈
---(SE)	---(SV)	定时器	扩展脉冲定时器线圈
---(SF)	---(SA)	定时器	断开延时定时器线圈
SHL_DW	SHL_DW	移位/循环	双字左移
SHL_W	SHL_W	移位/循环	左移字
SHR_DI	SHR_DI	移位/循环	右移双精度整数
SHR_DW	SHR_DW	移位/循环	右移双字
SHR_I	SHR_I	移位/循环	右移整数
SHR_W	SHR_W	移位/循环	右移字
SIN	SIN	浮点数指令	得到正弦值
S_ODT	S_EVERZ	定时器	接通延时S5定时器
S_ODTS	S_SEVERZ	定时器	保持接通延时S5定时器
S_OFFDT	S_AVERZ	定时器	断开延时S5定时器
---(SP)	---(SI)	定时器	脉冲定时器线圈
S_PEXT	S_VIMP	定时器	扩展脉冲S5定时器
S_PULSE	S_IMPULS	定时器	脉冲S5定时器
SQR	SQR	浮点数指令	得到平方

英语 助记符	德语 助记符	程序元素目录	
SQRT	SQRT	浮点数指令	得到平方根
SR	SR	位逻辑指令	置位复位触发器
---(SS)	---(SS)	定时器	保持接通延时定时器线圈
SUB_DI	SUB_DI	整数数学运算指令	减双精度整数
SUB_I	SUB_I	整数数学运算指令	减整数
SUB_R	SUB_R	浮点数指令	减实数
TAN	TAN	浮点数指令	得到正切值
TRUNC	TRUNC	转换	截断双精度整数部分
UO ---| |---	UO ---| |---	状态位	异常位无序
WAND_DW	WAND_DW	字逻辑指令	与运算双字
WAND_W	WAND_W	字逻辑指令	与运算字
WOR_DW	WOR_DW	字逻辑指令	或运算双字
WOR_W	WOR_W	字逻辑指令	或运算字
WXOR_DW	WXOR_DW	字逻辑指令	异或运算双字
WXOR_W	WXOR_W	字逻辑指令	异或运算字

A.2 按德语助记符(SIMATIC)排序的LAD指令

德语 助记符	英语 助记符	程序 元素目录	说明
---| |---	---| |---	位逻辑指令	常开触点(地址)
---|/|---	---|/|---	位逻辑指令	常闭触点(地址)
---( )	---( )	位逻辑指令	输出线圈
---(#)---	---(#)---	位逻辑指令	中间线输出
==0 ---| |---	==0 ---| |---	状态位	结果位等于0
>0 ---| |---	>0 ---| |---	状态位	结果位大于0
>=0 ---| |---	>=0 ---| |---	状态位	结果位大于等于0
<=0 ---| |---	<=0 ---| |---	状态位	结果位小于等于0
<0 ---| |---	<0 ---| |---	状态位	结果位小于0
<>0 ---| |---	<>0 ---| |---	状态位	结果位不等于0
ABS	ABS	浮点型指令	得到浮点型数字的绝对值
ACOS	ACOS	浮点型指令	得到反余弦值
ADD_DI	ADD_DI	整型数学运算指令	加双精度整数
ADD_I	ADD_I	整型数学运算指令	加整数
ADD_R	ADD_R	浮点型指令	加实数
ASIN	ASIN	浮点型指令	得到反正弦值
ATAN	ATAN	浮点型指令	得到反正切值
BCD_DI	BCD_DI	转换	BCD码转换为双精度整数
BCD_I	BCD_I	转换	BCD码转换为整数
BIE ---| |---	BR ---| |---	状态位	异常位二进制结果
----(CALL)	----(CALL)	程序控制	调用来自线圈的FC SFC(不带参数)
CALL_FB	CALL_FB	程序控制	调用来自框的FB
CALL_FC	CALL_FC	程序控制	调用来自框的FC
CALL_SFB	CALL_SFB	程序控制	调用来自框的系统FB
CALL_SFC	CALL_SFC	程序控制	调用来自框的系统FC
CEIL	CEIL	转换	上限
CMP >=D	CMP >=D	比较	比较长整数(==、<>、>、<、>=、<=)
CMP >=I	CMP >=I	比较	比较整数(==、<>、>、<、>=、<=)
CMP >=R	CMP >=R	比较	比较实数(==、<>、>、<、>=、<=)
COS	COS	浮点型指令	求余弦值

德语 助记符	英语 助记符	程序 元素目录	说明
DI_BCD	DI_BCD	转换	双精度整数转换为BCD码
DI_R	DI_R	转换	双精度整数转换为浮点数
DIV_DI	DIV_DI	整型数学运算指令	除双精度整数
DIV_I	DIV_I	整型数学运算指令	除整数
DIV_R	DIV_R	浮点数指令	除实数
EXP	EXP	浮点数指令	得到指数值
FLOOR	FLOOR	转换	基数
I_BCD	I_BCD	转换	整数转换为BCD码
I_DI	I_DI	转换	整数转换为双精度整数
INV_I	INV_I	转换	二进制反码整数
INV_DI	INV_DI	转换	二进制反码双精度整数
---(JMP)	---(JMP)	跳转	有条件跳转
---(JMP)	---(JMP)	跳转	无条件跳转
---(JMPN)	---(JMPN)	跳转	若非则跳转
LABEL	LABEL	跳转	标号
LN	LN	浮点数指令	得到自然对数
---(MCR>)	---(MCR>)	程序控制	主控制继电器关闭
---(MCR<)	---(MCR<)	程序控制	主控制继电器打开
---(MCRA)	---(MCRA)	程序控制	主控制继电器激活
---(MCRD)	---(MCRD)	程序控制	主控制继电器取消激活
MOD_DI	MOD_DI	整数数学运算指令	返回分数双精度整数
MOVE	MOVE	传送	分配值
MUL_DI	MUL_DI	整数数学运算指令	乘双精度整数
MUL_I	MUL_I	整数数学运算指令	乘整数
MUL_R	MUL_R	浮点数指令	乘实数
---(N)---	---(N)---	位逻辑指令	RLO负跳沿检测
NEG	NEG	位逻辑指令	地址下降沿检测
NEG_DI	NEG_DI	转换	二进制补码双精度整数
NEG_I	NEG_I	转换	二进制补码整数
NEG_R	NEG_R	转换	取反浮点数数字
---| NOT |---	---| NOT |---	位逻辑指令	取反能流
---(OPN)	---(OPN)	DB调用	打开数据块：DB或DI
OS ---| |---	OS ---| |---	状态位	存储的异常位溢出
OV ---| |---	OV ---| |---	状态位	异常位溢出
---(P)---	---(P)---	位逻辑指令	RLO正跳沿检测

德语 助记符	英语 助记符	程序 元素目录	说明
POS	POS	位逻辑指令	地址上升沿检测
---(R)	---(R)	位逻辑指令	复位线圈
---(RET)	---(RET)	程序控制	返回
ROL_DW	ROL_DW	移位/循环	循环左移双字
ROR_DW	ROR_DW	移位/循环	循环右移双字
ROUND	ROUND	转换	取整为双精度整数
RS	RS	位逻辑指令	复位置位触发器
---(S)	---(S)	位逻辑指令	置位线圈
---(SA)	---(SF)	定时器	断开延时定时器线圈
---(SAVE)	---(SAVE)	位逻辑指令	将RLO保存到BR存储器中
S_AVERZ	S_OFFDT	定时器	断开延时S5定时器
---(SE)	---(SD)	定时器	接通延时定时器线圈
S_EVERZ	S_ODT	定时器	接通延时S5定时器
SHL_DW	SHL_DW	移位/循环	双字左移
SHL_W	SHL_W	移位/循环	左移字
SHR_DI	SHR_DI	移位/循环	右移双精度整数
SHR_DW	SHR_DW	移位/循环	右移双字
SHR_I	SHR_I	移位/循环	右移整数
SHR_W	SHR_W	移位/循环	右移字
---(SI)	---(SP)	定时器	脉冲定时器线圈
S_IMPULS	S_PULSE	定时器	脉冲S5定时器
SIN	SIN	浮点数指令	得到正弦值
SQR	SQR	浮点数指令	得到平方
SQRT	SQRT	浮点数指令	得到平方根
SR	SR	位逻辑指令	置位复位触发器
---(SS)	---(SS)	定时器	保持接通延时定时器线圈
S_SEVERZ	S_ODTS	定时器	保持接通延时S5定时器
SUB_DI	SUB_DI	整数数学运算指令	减双精度整数
SUB_I	SUB_I	整数数学运算指令	减整数
SUB_R	SUB_R	浮点数指令	减实数
---(SV)	---(SE)	定时器	扩展脉冲定时器线圈
S_VIMP	S_PEXT	定时器	扩展脉冲S5定时器
---(SZ)	---(SC)	计数器	设置计数器值
TAN	TAN	浮点数指令	得到正切值
TRUNC	TRUNC	转换	截尾双精度整数部分
UO ---| |---	UO ---| |---	状态位	异常位无序

德语 助记符	英语 助记符	程序 元素目录	说明
WAND_DW	WAND_DW	字逻辑指令	与运算双字
WAND_W	WAND_W	字逻辑指令	与运算字
WOR_DW	WOR_DW	字逻辑指令	或运算双字
WOR_W	WOR_W	字逻辑指令	或运算字
WXOR_DW	WXOR_DW	字逻辑指令	异或运算双字
WXOR_W	WXOR_W	字逻辑指令	异或运算字
ZAEHLER	S_CUD	计数器	双向计数器
----(ZR)	----(CD)	计数器	减计数器线圈
Z_RUECK	S_CD	计数器	减计数器
---(ZV)	----(CU)	计数器	增计数器线圈
Z_VORW	S_CU	计数器	增计数器

B 编程实例

B.1 编程实例总览

实际应用

本手册中描述的每个梯形图指令都会触发一个特定操作。当将这些指令组合到程序中时，就可以实现各种各样的自动化任务。本章提供梯形图指令实际应用的以下实例：

- 控制传送带使用位逻辑指令
- 检测传送带的移动方向使用位逻辑指令使用位逻辑指令
- 生成时钟脉冲使用定时器指令
- 跟踪存储空间使用计数器和比较指令
- 使用整数数学运算指令解决问题
- 设置加热烘箱的时间长度

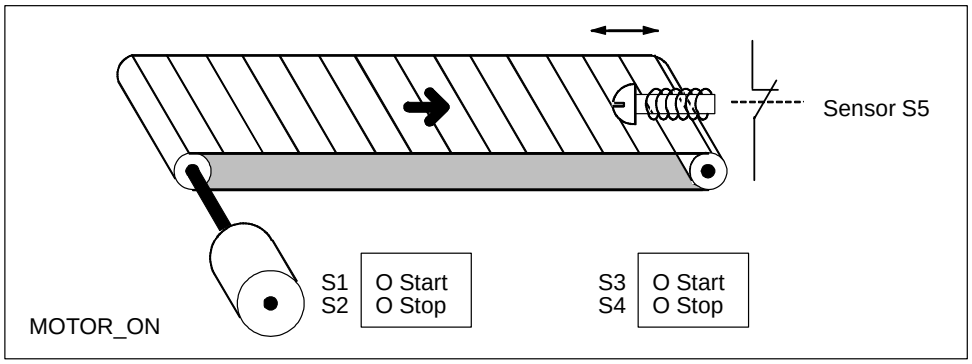
使用的指令

助记符	程序元素目录	说明
WAND_W	字逻辑指令	(字)与运算
WOR_W	字逻辑指令	(字)或运算
---(CD)	计数器	降值计数器线圈
---(CU)	计数器	升值计数器线圈
---(R)	位逻辑指令	重置线圈
---(S)	位逻辑指令	置位线圈
---(P)	位逻辑指令	RLO上升沿检测
ADD_I	浮点指令	整数加
DIV_I	浮点指令	整数除
MUL_I	浮点指令	整数乘
CMP <=I, CMP >=I	比较	比较整数
—| |—	位逻辑指令	常开触点
—| / |—	位逻辑指令	常闭触点
—( )	位逻辑指令	输出线圈
---(JMPN)	跳转	若非则跳转
---(RET)	程序控制	返回
MOVE	传送	分配值
---(SE)	定时器	扩展脉冲定时器线圈

B.2 实例：位逻辑指令

实例1：控制传送带

下图显示可以电气方式激活的传送带。在传送带的开始位置有两个按钮开关：用于启动的S1和用于停止的S2。在传送带末端也有两个按钮开关：用于启动的S3和用于停止的S4。从任何一端都可以启动或停止传送带。另外，当传送带上的物品到达尾部时传感器S5会使传送带停止。



绝对和符号编程

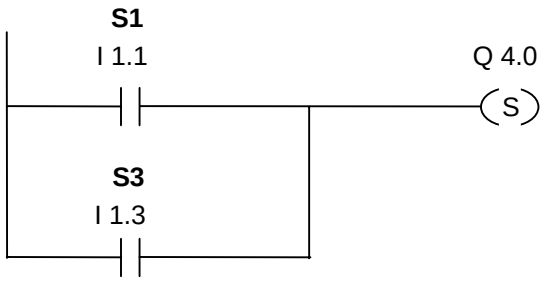
可使用代表传送系统各种组件的**绝对值**或**符号来**编写程序，以控制传送带。

需要制作一个符号表将您选择的符号与绝对值关联起来(参见STEP 7在线帮助)。

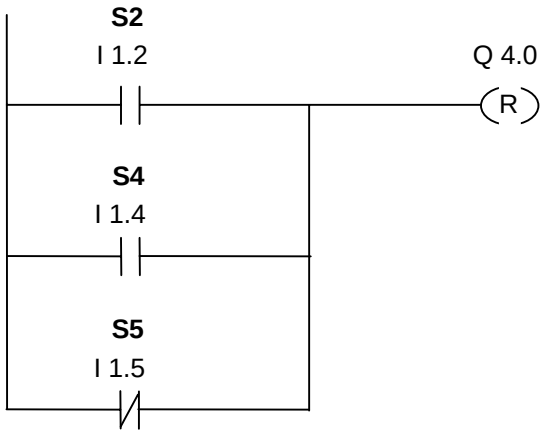
系统组件	绝对 地址	符号	符号表
按钮启动开关	I 1.1	S1	I 1.1 S1
按钮停止开关	I 1.2	S2	I 1.2 S2
按钮启动开关	I 1.3	S3	I 1.3 S3
按钮停止开关	I 1.4	S4	I 1.4 S4
传感器	I 1.5	S5	I 1.5 S5
电机	Q 4.0	MOTOR_ON	Q 4.0 MOTOR_ON

控制传送带的梯形图程序

程序段1：按下任一启动开关打开电机。

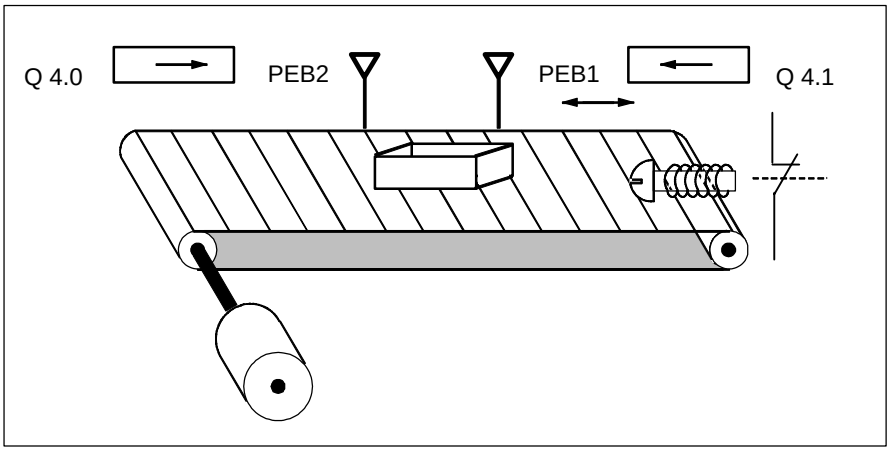


程序段2：按下任一停止开关或打开传送带尾部的常闭触点以关闭电机。



实例2：检测传送带方向

下图显示配备两个光电屏障(PEB1和PEB2)的传送带，这两个光电屏障专用于检测包裹在传送带上移动的方向。每个光电屏障都起到类似常开触点的作用。



绝对和符号编程

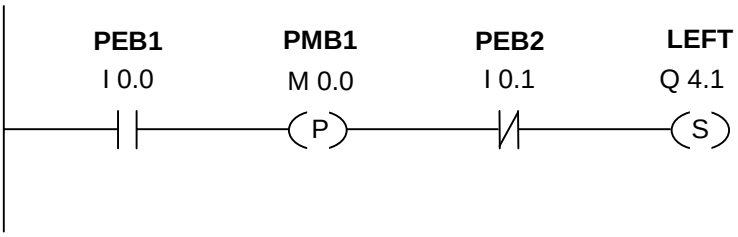
可使用代表传送系统各种组件的**绝对值**或**符号来**编写程序，以激活传送带系统的方向显示。

需要制作一个符号表将您选择的符号与绝对值关联起来(参见STEP 7在线帮助)。

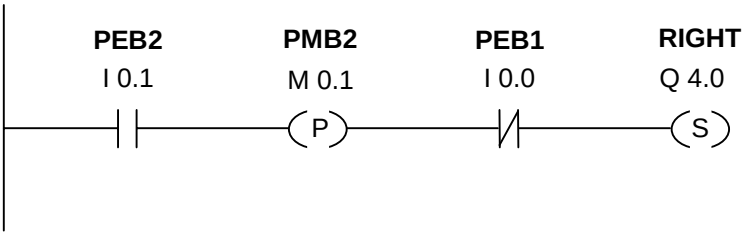
系统组件	绝对 地址	符号	符号表
光电屏障1	I 0.0	PEB1	I 0.0 PEB1
光电屏障2	I 0.1	PEB2	I 0.1 PEB2
右向移动显示	Q 4.0	右箭头	Q 4.0 RIGHT
左向移动显示	Q 4.1	左箭头	Q 4.1 LEFT
脉冲存储器位1	M 0.0	PMB1	M 0.0 PMB1
脉冲存储器位2	M 0.1	PMB2	M 0.1 PMB2

用于检测传送带方向的梯形图程序

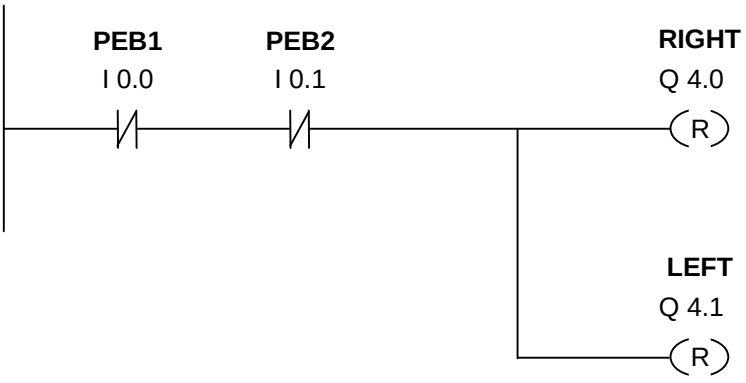
程序段1：如果输入I 0.0处信号状态从0过渡到1(上升沿)，同时输入I 0.1处信号状态为0，则传送带上的包裹会向左移动。



程序段2：如果输入I 0.1处信号状态从0过渡到1(上升沿)，同时输入I 0.0处信号状态为0，则传送带上的包裹会向右移动。如果光电屏障之一被中断，则表明屏障之间有包裹。



程序段3：如果两个光电屏障都未断开，则说明屏障之间无包裹。方向指针关闭。



B.3 实例：定时器指令

时钟脉冲发生器

需要产生周期性重复的信号时，可以使用时钟脉冲发生器或闪光继电器。时钟脉冲发生器在控制指示灯闪烁的信号系统中很常见。

使用S7-300时，可以通过在特殊组织块中使用时间驱动处理来实现时钟脉冲发生器功能。但下列梯形图程序中显示的实例说明的是使用定时器功能产生时钟脉冲。此示例程序显示如何使用定时器来实现任意的时钟脉冲发生器。

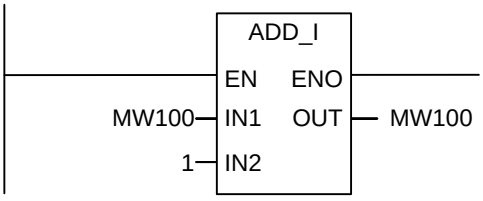
产生时钟脉冲(脉冲占空比1:1)的梯形图程序

程序段1：如果定时器T1的信号状态为0，将时间值250毫秒装入T1，并将T1作为扩展脉冲定时器启动。

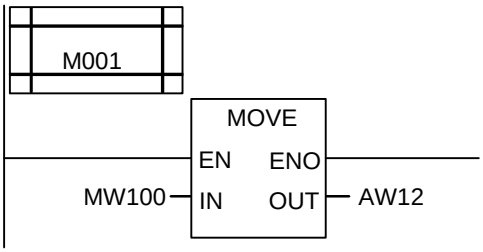
程序段2：该定时器的状态临时保存在一个辅助存储器符号中。

程序段3：如果定时器T1的信号状态为1，则跳转至跳转标签M001。

程序段4：定时器T1超时后，内存字100增加1。

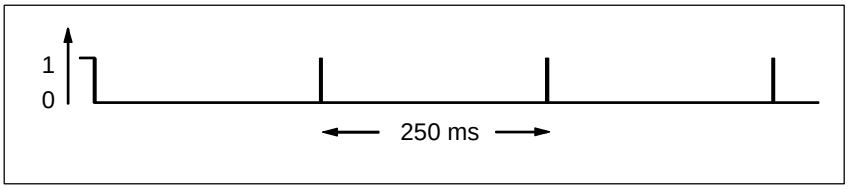


程序段5：**MOVE**指令允许在输出Q12.0到Q13.7处输出不同的时钟频率。



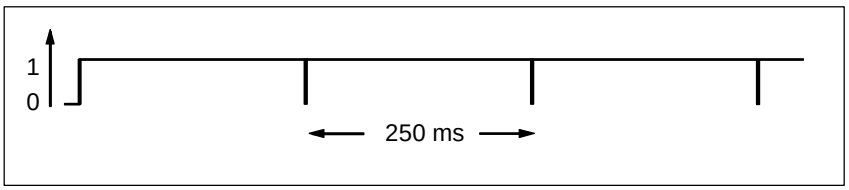
信号检查

定时器T1的信号检查为opener M0.2生成以下逻辑运算(RLO)结果。



一旦超时，定时器就重新启动。因此，由 ---| / |--- M0.2进行的信号检查只简单产生信号状态1。

取反(反向)RLO：



每隔250毫秒RLO位为0。忽略跳转且存储器字MW100的内容增加1。

实现特定频率

通过存储器字节MB101和MB100的单个位，可以实现下列频率：

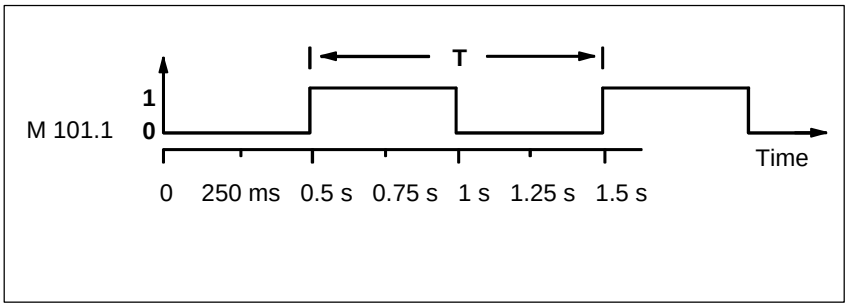
MB101/MB100的位	频率(赫兹)	持续时间
M 101.0	2.0	0.5s （250毫秒开/250毫秒关)
M 101.1	1.0	1秒 （0.5秒开/0.5秒关)
M 101.2	0.5	2秒 （1秒开/1秒关)
M 101.3	0.25	4s （2秒开/2秒关)
M 101.4	0.125	8s （4秒开/4秒关)
M 101.5	0.0625	16s （8秒开/8秒关)
M 101.6	0.03125	32s （16秒开/16秒关)
M 101.7	0.015625	64s （32秒开/32秒关)
M 100.0	0.0078125	128s （64秒开/64秒关)
M 100.1	0.0039062	256s （128秒开/128秒关)
M 100.2	0.0019531	512s （256秒开/256秒关)
M 100.3	0.0009765	1024s （512秒开/512秒关)
M 100.4	0.0004882	2048s （1024秒开/1024秒关)
M 100.5	0.0002441	4096s （2048秒开/2048秒关)
M 100.6	0.000122	8192s （4096秒开/4096秒关)
M 100.7	0.000061	16384s （8192秒开/8192秒关)

内存MB 101的位信号状态

扫描周期	位7	位6	位5	位4	位3	位2	位1	位0	时间值(毫秒)
0	0	0	0	0	0	0	**0**	0	250
1	0	0	0	0	0	0	**0**	1	250
2	0	0	0	0	0	0	1	0	250
3	0	0	0	0	0	0	1	1	250
4	0	0	0	0	0	1	**0**	0	250
5	0	0	0	0	0	1	**0**	1	250
6	0	0	0	0	0	1	1	0	250
7	0	0	0	0	0	1	1	1	250
8	0	0	0	0	1	0	**0**	0	250
9	0	0	0	0	1	0	**0**	1	250
10	0	0	0	0	1	0	1	0	250
11	0	0	0	0	1	0	1	1	250
12	0	0	0	0	1	1	**0**	0	250

MB 101的位1 (M 101.1)的信号状态

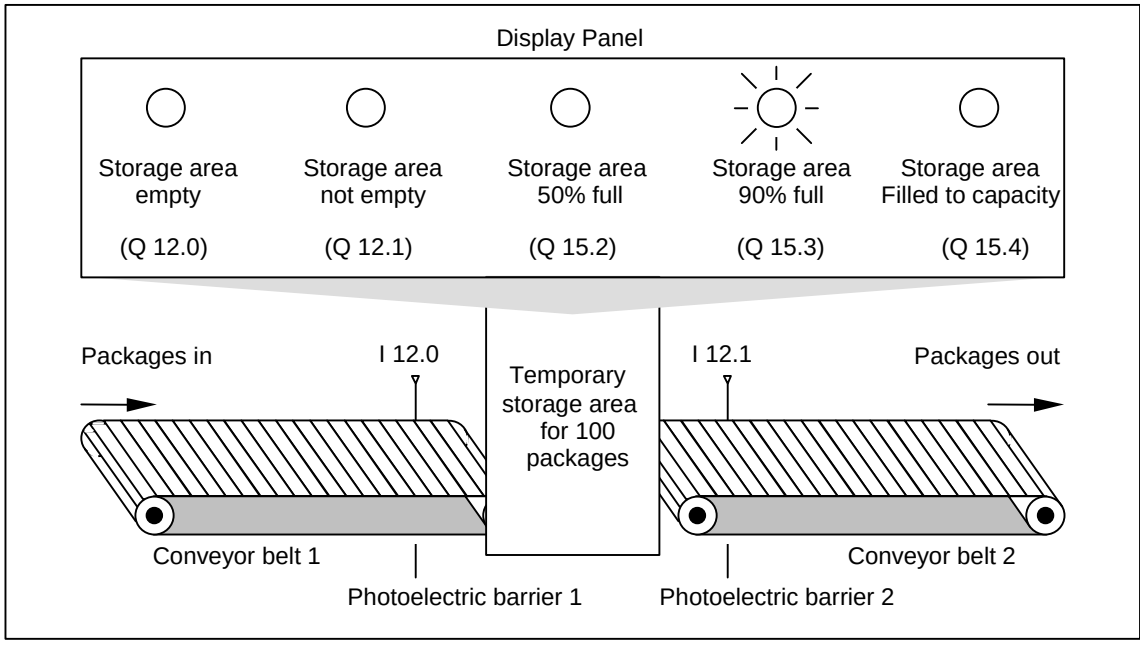
频率 = 1/T = 1/1 s = 1赫兹



B.4 实例：计数器和比较指令

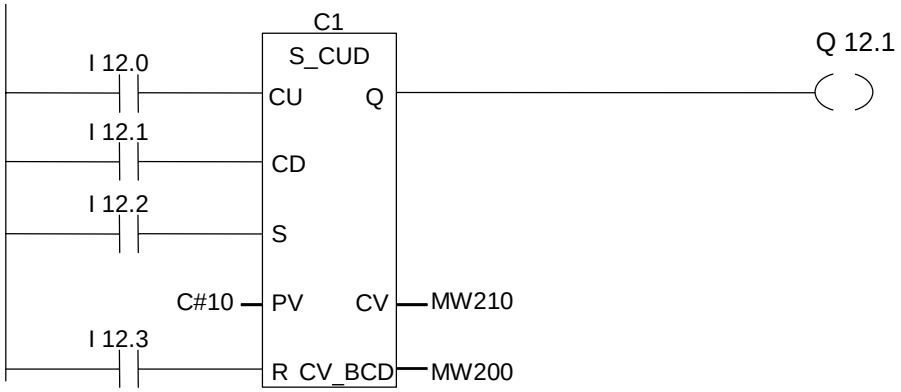
带计数器和比较器的存储区域

下图显示了带有两个传送带并且它们之间有一个临时存储区域的系统。传送带1将包裹传送到存储区域。传送带1的尾部靠近存储区域处有一个光电屏障，它确定向存储区域传送的包裹的数量。传送带2将包裹从临时存储区域传输到装载台，卡车从此处取走包裹并发送给用户。传送带2的尾部靠近存储区域处有一个光电屏障，它确定离开存储区域进入装载台的包裹的数量。带有五个灯的显示面板指示临时存储区域的填充量。



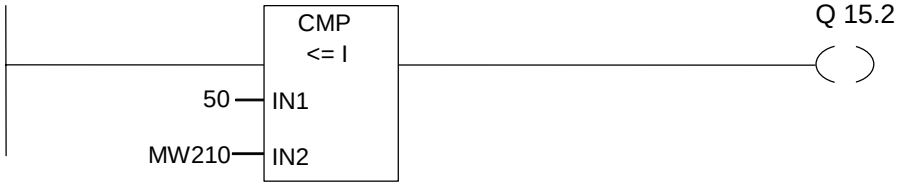
激活显示面板上的指示灯的梯形图程序

程序段1：计数器C1对输入CU处每次从“0”到“1”的信号改变都进行正计数，而对输入CD处每次从“0”到“1”的信号改变都进行倒计数。对于输入S处从“0”到“1”的信号改变，计数器值被设置为值PV。输入R处从“0”到“1”的信号改变将计数器值复位为“0”。MW200包含C1的当前计数器值。Q12.1指示“存储区域非空”。

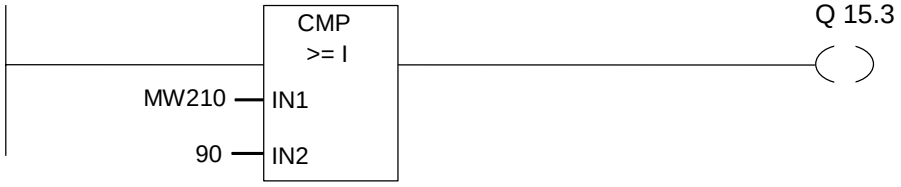


程序段2：Q12.0表明“存储区域为空”。

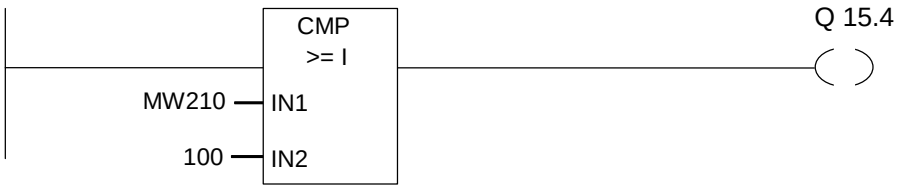
程序段3：如果50小于等于计数器值(换句话说，如果当前计数器值大于等于50)，则表示“存储区域50%满”的指示灯变亮。



程序段4：程序段4：如果计数器值大于等于90，“存储区域满90%”指示灯亮起。



程序段5：如果计数器值大于或等于100，则表示“存储区域满”的指示灯变亮。



B.5 实例：整型数学运算指令

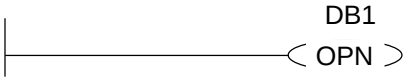
解决数学问题

此示例程序显示如何使用三个整数运算指令生成与下式相同的结果：

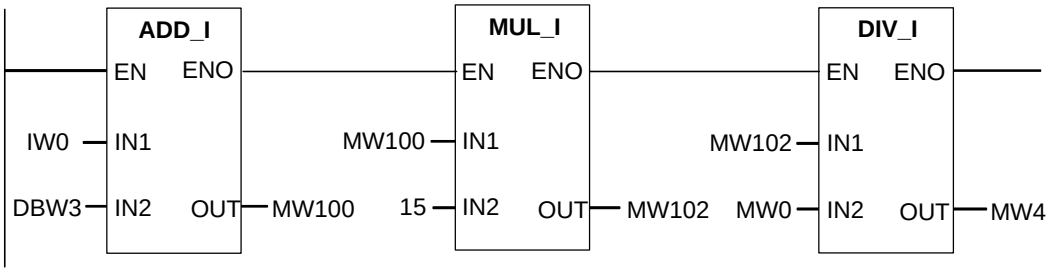
MW4 =((IW0 + DBW3) x 15) / MW0

梯形图程序

程序段1：打开数据块DB1。



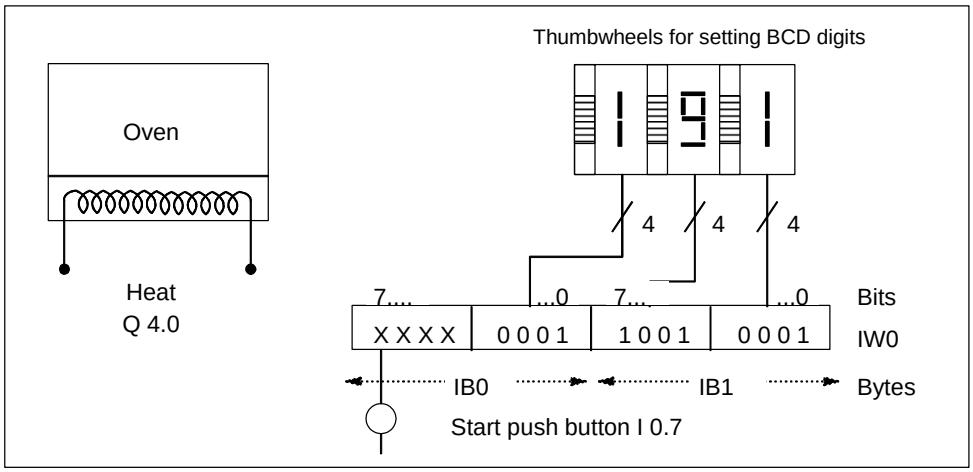
程序段2：输入字IW0与共享数据字DBW3相加(必须定义和打开数据块)，总和被载入存储器字MW100。然后，MW100乘以15，结果存储到存储器字MW102中。MW102除以MW0，结果存储到MW4中。



B.6 实例：字逻辑指令

加热烘箱

烘箱操作员通过按下启动按钮来启动烘箱加热。操作员可以使用图中所示的指轮开关设置加热时间。操作员设置的值以二进制编码十进制(BCD)格式显示秒数。



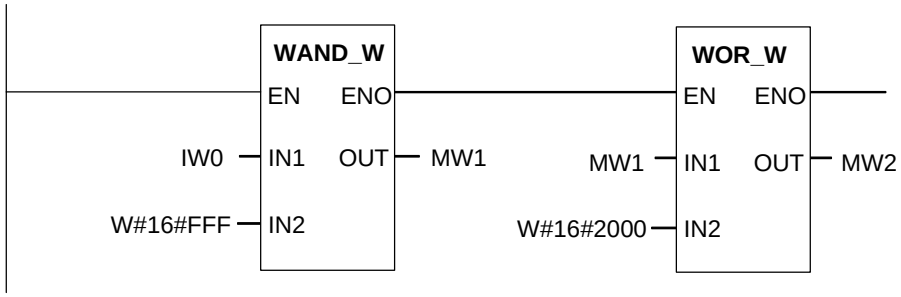
系统组件	绝对 地址
启动按钮	I 0.7
个位指轮开关	I 1.0到I 1.3
十位指轮开关	I 1.4到I 1.7
百位指轮开关	I 0.0到I 0.3
加热启动	Q 4.0

梯形图程序

程序段1：如果定时器正在运行，则打开加热器。

程序段2：如果定时器正在运行，**返回**指令结束此处的处理。

程序段3：屏蔽输入位I 0.4到I 0.7(即，将它们复位为0)。指轮开关输入的这些位未被使用。16位指轮开关输入根据**(字)与运算**指令与W#16#0FFF组合。结果载入存储器字MW1中。为了设置时间基准的秒数，预设值根据**(字)或运算**指令与W#16#2000组合，将位13设置为1，并将位12复位为0。



程序段4：如果按下启动按钮，则将定时器T 1作为扩展脉冲定时器启动，并作为预设值存储器字MW2装载(来自于上述逻辑)。

C 使用梯形图逻辑

C.1 EN/ENO机制

FBD/LAD框的使能(EN)和启用输出(ENO)通过BR位来获取。

如果连接了EN和ENO，则以下规则适用：

ENO = EN与非(框错误)

如果未出错(框错误 = 0)，ENO = EN。

EN/ENO机制用于：

- 数学运算指令、
- 传送和转换指令、
- 移位和循环移位指令、
- 块调用。

此机制**不**适合：

- 比较、
- 计数器、
- 定时器。

在框的实际指令周围，为EN/ENO机制生成附加的STL指令，这些指令依赖于现有的在此之前和之后的逻辑运算。以下是使用加法器实例说明的四种可能情况：

1. 加法器连接了EN和ENO
2. 加法器连接了EN但未连接ENO
3. 加法器未连接EN但连接了ENO
4. 加法器未连接EN和ENO

有关创建用户自己的块的说明

如果要编程在FBD或LAD中调用的块，那么必须确保退出块时，置位BR位。第四个例子说明这并不是一个自动处理过程。不能使用BR作为存储位，因为EN/ENO机制会不断重写BR位。.作为代替，可使用一个临时变量来保存发生的所有错误。请用0初始化此变量。在块中您认为由于不成功的指令导致整个块错误的各点，借助EN/ENO机制设置此变量。要做到这一点，只需要一个NOT和SET线圈即可。在块结尾编写以下程序段：

```
end:    AN error
        SAVE
```

确保在任何情况下都处理本程序段，这表示禁止在块内使用BEC，并禁止跳过本程序段。

C.1.1 加法器连接了EN和ENO

如果加法器连接了EN和ENO，则会触发以下STL指令：

```
1            A  I  0.0          //EN连接
2   JNB _001                    //将RLO移入BR，并在RLO = 0时跳转
3            L    in1  //框参数
4   L    in2            //框参数
5            +I                 //实际加法
6   T    out            //框参数
7            AN   OV//错误识别
8   SAVE                        //在BR中保存错误
9            CLR                //第一次校验
10 _001: A    BR       //将BR移位到RLO中
11  =    Q   4.0
```

在第1行之后，RLO包含先前逻辑运算的结果。JNB指令将RLO复制到BR位中，并设置首次检测位。

- 如果RLO = 0，则程序会跳到第10行，并继续执行A BR指令。
  不执行加法。在第10行中，再次将BR复制到RLO，从而将0赋给输出。
- 如果RLO = 1，则程序不跳转，即执行加法。在第7行，程序会计算执行加法期间是否出错，结果将存储在第8行的BR中。第9行设置首次检测位。此时在第10行即会将BR位复制回RLO中，从而输出结果将表明加法运算是否成功。
  BR位不会被第10行和11行改变，所以它也能够说明加法是否成功。

C.1.2 加法器连接了EN但未连接ENO

如果加法器连接了EN，但未连接ENO，则会触发以下STL指令：

```
1   A I 0.0             //EN连接
2             JNB _001          //将RLO移入BR，并在RLO = 0时跳转
3   L    in1            //框参数
4             L    in2  //框参数
5   +I                           //实际加法
6             T    out  //框参数
7   _001:     NOP  0
```

在第1行之后，RLO包含先前逻辑运算的结果。JNB指令将RLO复制到BR位中，并设置首次检测位。

- 如果RLO = 0，程序会跳转到第7行，且不执行加法。RLO和BR为0。
- 如果RLO为1，程序不会跳转，即执行加法。程序不判断执行加法期间是否出错。RLO和BR为1。

C.1.3 加法器未连接EN但连接了ENO

如果加法器未连接EN，但连接了ENO，则会触发以下STL指令：

```
1   L    in1            //框参数
2             L    in2  //框参数
3   +I                           //实际加法
4             T    out  //框参数
5   AN   OV             //错误识别
6             SAVE               //在BR中保存错误
7   CLR                          //第一次校验
8             A    BR   //将BR移位到RLO中
9             =  Q  4.0
```

在所有情况下都执行加法。在第5行，程序会判断执行加法期间是否出错，结果将存储在第6行的BR中。第7行设置首次检测位。此时在第8行即会将BR位复制回RLO中，从而输出结果将表明加法运算是否成功。

BR位不会被第8行和9行改变，所以它也能够说明加法是否成功。

C.1.4 加法器未连接EN和ENO

如果加法器未连接EN和ENO，则会触发以下STL指令：

```
1          L    in1  //框参数
2  L   in2           //框参数
3          +I              //实际加法
4  T   out           //框参数
5          NOP 0
```

执行加法。RLO和BR位保持不变。

C.2 参数传递

块的参数作为值传递。对于功能块，在被调用块中使用情景数据块中的实际参数值的副本。对于功能，实际值的副本存在于本地数据栈中。将不复制指针。在调用之前，将INPUT值复制到情景数据块或L堆栈中。调用以后，将OUTPUT值往回复制给变量。在被调用的块中，仅可使用副本。所需的STL指令存在于调用块中并且用户不可见。

注意

如果将位、输入、输出或外设I/O存储区用作功能的实际地址，则对它们的处理方法将不同于对其它地址的处理方法。在此，直接执行更新，而不是通过L堆栈更新。

例外：

如果相应的形式参数是BOOL数据类型的输入参数，则将通过L堆栈更新当前参数。

当心

编写被调用块时，请确保还写入了声明为OUTPUT的参数。否则，将输出随机值！
对于功能块，此值将是上一次调用记录的来自情景数据块的值；对于功能，此值将恰巧是L堆栈中的值。

请注意以下几点：

- 尽可能初始化所有OUTPUT参数。
- 尽量不要使用置位和复位指令。这些指令与RLO相关。如果RLO具有0值，则将保留随机值。
- 如果在块内跳转，请确保不要跳过任何编写了OUTPUT参数的位置。请勿忘记BEC和MCR指令的作用。

索引

字母

A

B

C

D

E

F

J

K

L

M

N

Q

R

S

T

W

X

Y

Z