SIEMENS

SIMATIC

用于 S7-300 和 S7-400 的功能块图(FBD)编程

参考手册

2004 年 1 月版
A5E00446507-01

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

Siemens Aktiengesellschaft

A5E00446507-01

前言

目的

本手册是以功能块图(FBD)编程语言创建用户程序的指南。

本手册也包含了描述功能块图中语言要素的语法和函数的参考部分。

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。

符合的标准

FBD 即“功能块图”语言，由国际电工技术委员会标准 IEC 1131-3 定义。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

要有效地使用功能块图手册，应先熟悉 STEP 7 在线帮助中提供的 S7 程序原理。该语言包也使用了 STEP 7 标准软件，所以还应熟悉该软件的操作，并阅读了相关的文档。

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	目的	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，描述了使用 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(FBD)/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，并描述了编程语言 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。
STL/LAD/FBD 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 STEP7 应用程序的词汇表。

本手册是“功能块图帮助”的摘录。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

描述

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系统的数系的基础。将 1 和 0 两个数字称作二进制数字或位。在“与”运算、“或”运算、“异或”运算和输出连用时，1 代表逻辑“是”，0 代表逻辑“否”。

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。这些组合生成的结果 1 或 0 称为“逻辑操作的结果”(RLO)。

有可以执行下列功能的位逻辑指令：

- 与运算或运算和异或运算：这些指令检查信号状态并产生一个结果，然后将结果复制到 RLO 位或与其组合。
- 先“与”后“或”逻辑操作和先“或”后“与”逻辑操作
- 赋值和中间输出。这些指令用于设置 RLO 或临时存储它。

RLO 为 1 时将触发下列指令：

- S：设置输出
- R：复位输出
- SR：置位复位触发器
- RS：复位置位触发器

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：

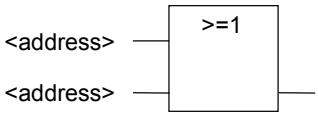
- N：RLO 负跳沿检测
- P：RLO 正跳沿检测
- NEG：地址负跳沿检测
- POS：地址正跳沿检测

其余指令直接以下列方式影响 RLO：

- 插入数字输入
- 数字输入取反
- SAVE：将 RLO 存入 BR 存储区

1.2 >=1：“或”逻辑操作

符号



参数	数据类型	内存区域	描述
<地址>	BOOL	I、Q、M、T、C、D、L	地址表示将检查其信号状态的位。

描述

使用“**或**”运算指令，可以检查在“或”运算框输入处两个或更多个指定地址的信号状态。

如果其中一个地址的信号状态为 1，则满足条件，此指令产生结果 1。如果所有地址的信号状态都为 0，则不满足条件，此指令产生结果 0。

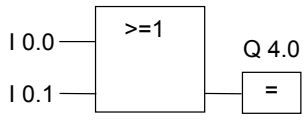
如果“**或**”运算指令是系列逻辑操作的第一个指令，则它会将其信号状态的检查结果存入 RLO 位。

如果“**或**”运算指令不是系列逻辑操作的第一个指令，则它会组合其信号状态的检查结果与 RLO 位中存储的值。这些值将根据“或”真值表进行组合。

状态字

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

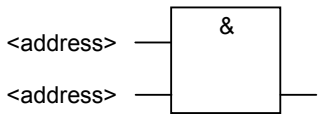
举例



如果输入 I0.0“或”输入 I0.1 的信号状态为 1，则输出 Q4.0 被置位。

1.3 &：“与”逻辑操作

符号



参数	数据类型	内存区域	描述
<地址>	BOOL	I、Q、M、T、C、D、L	地址表示将检查其信号状态的位。

描述

使用“与”运算指令，可以检查在“与”运算框输入处两个或更多个指定地址的信号状态。

如果所有操作数的信号状态都为 1，则满足条件，并且此指令的结果为 1。如果有一个地址的信号状态为 0，则不满足条件，并且该指令生成结果 0。

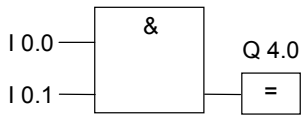
如果“与”运算指令是系列逻辑操作的第一个指令，则它会将其信号状态的检查结果存入 RLO 位。

如果“与”运算指令不是系列逻辑操作的首个指令，则它会组合其信号状态的检查结果与 RLO 位中存储的值。这些值将根据“与”运算真值表进行组合。

状态字

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

举例



如果输入 I0.0“与”I0.1 的信号状态为 1，则输出 Q4.0 被置位。

1.4 先“与”后“或”逻辑操作和先“或”后“与”逻辑操作

描述

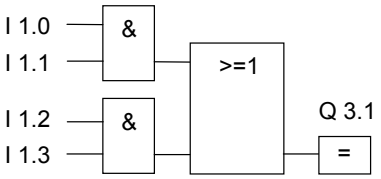
使用**先“与”后“或”**指令，可以根据“或”运算真值表检查信号状态的结果。

对于先“与”后“或”逻辑操作，至少有一个“与”逻辑操作得到满足时，信号状态才为 1。

状态字

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

举例



如果至少有一个“与”逻辑操作得到满足，输出 Q3.1 的信号状态为 1。

如果全部“与”逻辑操作均不满足，则输出 Q3.1 的信号状态为 0。

描述

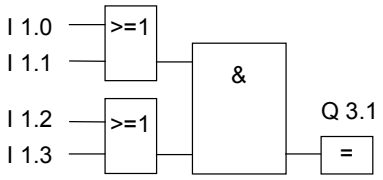
使用**先“或”后“与”**指令，可以根据“与”真值表检查信号状态的结果。

对于**先“或”后“与”**逻辑操作，必须满足全部“或”逻辑操作，信号状态才为 1。

状态字

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

举例

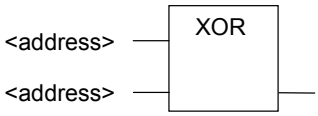


如果两个或逻辑操作都满足，输出 Q3.1 的信号状态为 1。

如果至少有一个“或”逻辑操作不满足，输出 Q3.1 的信号状态为 0。

1.5 XOR：“异或”逻辑操作

符号



参数	数据类型	内存区域	描述
<地址>	BOOL	I、Q、M、T、C、D、L	地址表示将检查其信号状态的位。

描述

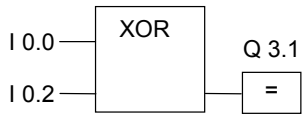
使用“**异或**”运算指令，可以根据“异或”运算真值表检查信号状态的结果。

对于“**异或**”逻辑操作，两个指定地址之一的信号状态为 1 时，其信号状态为 1。也可以重复使用“异或”运算功能。因此，如果有奇数个被检查地址为“1”，则逻辑操作的交互结果为“1”。

状态字

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

举例



如果输入 I0.0“或”输入 I0.2 的信号状态为 1 (互斥，换言之不同时为 1)，输出 Q3.1 的信号状态为 1。

1.6 插入数字输入

符号

<地址>

参数	数据类型	内存区域	描述
<地址>	BOOL	I、Q、M、T、C、D、L	地址表示将检查其信号状态的位。

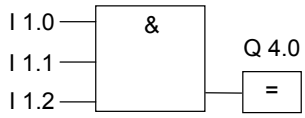
描述

插入数字输入指令在“与”运算、“或”运算或“异或”运算框中再插入一个二进制输入。

状态字

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

举例



如果 I1.0“与”I1.1“与”I1.2 的信号状态均为 1，输出 Q4.0 为 1。

1.7 数字输入取反

符号

描述

数字输入取反指令对 RLO 取反。

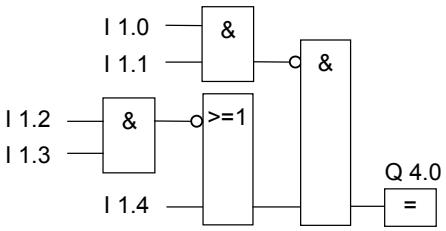
如果要对逻辑操作的结果取反，务必谨记以下规则：

- 如果是对“与”运算框或者“或”运算框的第一个输入处的逻辑操作结果取反，则不进行嵌套。
- 如果被取反的对象不是“或”运算框的第一个输入处的逻辑操作结果，则在此“或”逻辑操作中将包括在此输入之前的全部二进制逻辑操作。
- 如果被取反的对象不是“与”运算框的第一个输入处的逻辑操作结果，则在此“与”逻辑操作中将包括在此输入之前的全部二进制逻辑操作。

状态字

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

举例

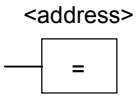


如果满足以下条件，则输出 Q4.0 为 1：

- I1.0“与”I1.1 运算的信号状态为 0
- 并且 I1.2“与”I1.3 运算的信号状态为 0
- 或者 I1.4 的信号状态为 0。

1.8 =：赋值

符号



参数	数据类型	内存区域	描述
<地址>	BOOL	I、Q、M、D、L	地址将指定要为其分配系列逻辑操作的信号状态值的位。

描述

赋值指令生成逻辑操作的结果。根据下列标准，逻辑操作结束后框中的信号为 1 或 0：

- 满足该输出框之前的逻辑操作条件时，输出信号为 1。
- 不满足该输出框之前的逻辑操作条件时，输出信号为 0。

FBD 逻辑操作将信号状态赋给此由指令寻址的输出(为了达到同样的效果，也可以将 RLO 位的信号状态赋给该地址)。如果 FBD 逻辑操作的条件得到满足，则输出框中的信号状态为 1。否则，信号状态为 0。**赋值**指令受主控继电器(MCR)的影响。

关于 MCR 函数的更详细信息，请参考 MCR 开/关。

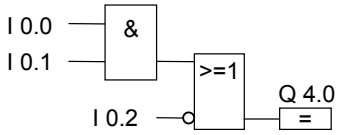
只能将**赋值**框置于系列逻辑操作的右端。然而，可以使用多个**赋值**框。

可以使用**取反输入**指令创建取反的赋值。

状态字

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

举例

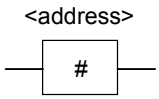


如果满足以下条件，输出 Q4.0 的信号状态为 1：

- 输入 I0.0“与”I0.1 的信号状态为 1
- 或者 I0.2 为 0

1.9 #：中间输出

符号



参数	数据类型	内存区域	描述
<地址>	BOOL	I、Q、M、D、*L	地址指定要为其设置 RLO 的位。

* 如果地址是在代码块(FC、FB、OB)的 TEMP 区域的变量声明表中声明，则只能使用本地数据栈中的地址。

描述

中间输出指令是缓存 RLO 的一个中间元素。更准确地说，此元素缓存在执行“中间输出”前要打开的上一个分支的位逻辑操作。

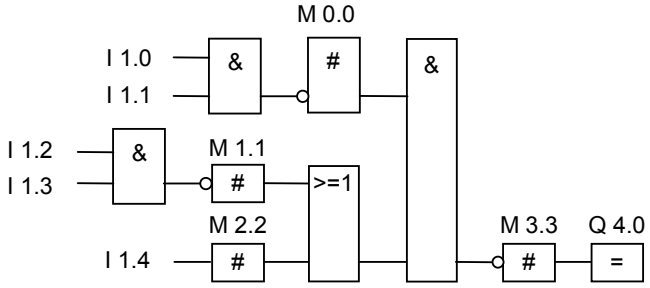
中间输出指令受主控继电器(MCR)的影响。关于 MCR 如何起作用的更详细信息，请参考 MCR 开/关。

可以通过取反“中间输出”的输入来生成取反的“中间输出”。

状态字

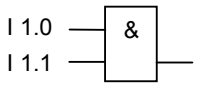
	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	FC
写	-	-	-	-	-	0	X	-	1

举例

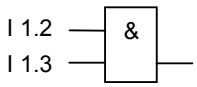


“中间输出”会缓存下列逻辑操作的结果：

M0.0 缓存以下逻辑操作的取反 RLO：



M1.1 保存以下逻辑操作的取反 RLO：

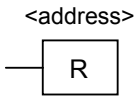


M2.2 保存 I1.4 的 RLO

M3.3 保存整个位逻辑操作的取反 RLO。

1.10 R：复位输出

符号



参数	数据类型	内存区域	描述
<地址>	BOOL TIMER COUNTER	I、Q、M、T、C、 D、L	地址指定将要复位哪一位。

描述

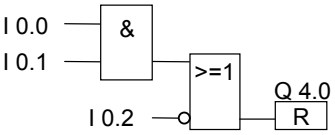
只有在 RLO 为 1 时，才执行**复位输出**指令。如果 RLO 为 1，此指令将指定地址复位为 0。如果 RLO 为 0，此指令不影响指定地址，该地址中的内容将保持不变。

复位输出指令受主控继电器(MCR)的影响。关于 MCR 如何起作用的更多详细信息，请参考 MCR 开/关。

状态字

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

举例



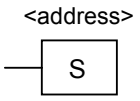
仅当满足下列条件时，输出 Q4.0 的信号状态才复位为 0：

- 输入 I0.0“与”I0.1 的信号状态为 1
- 或者输入 I0.2 的信号状态为 0。

如果分支的 RLO 为 0，则输出 Q4.0 的信号状态不变。

1.11 S：设置输出

符号



参数	数据类型	内存区域	描述
<地址>	BOOL	I、Q、M、D、L	地址指定将要置位的位。

描述

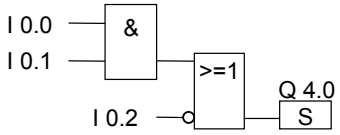
只有在 RLO 为 1 时，才执行**置位输出**指令。如果 RLO 为 1，此指令将指定地址置 1。如果 RLO 为 0，此指令不影响指定地址，该地址中的内容将保持不变。

置位输出指令受主控继电器(MCR)的影响。关于 MCR 如何起作用的更多详细信息，请参考 MCR 开/关。

状态字

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

举例



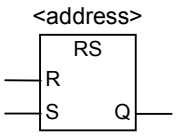
仅当满足下列条件时，才将输出 Q4.0 的信号状态置 1：

- 输入 I0.0“与”I0.1 的信号状态为 1
- 或者输入 I0.2 的信号状态为 0。

如果分支的 RLO 为 0，则 Q4.0 的信号状态不变。

1.12 RS：复位置位触发器

符号



参数	数据类型	内存区域	描述
<地址>	BOOL	I、Q、M、D、L	地址指定将要置位或复位的位。
S	BOOL	I、Q、M、D、L、T、C	启用了复位指令
R	BOOL	I、Q、M、D、L、T、C	启用了置位指令
Q	BOOL	I、Q、M、D、L	<地址>的信号状态

描述

复位置位触发器指令仅在 RLO 为 1 时执行“置位”(S)或“复位”(R)等指令。RLO 为 0 时不影响这些指令，在指令中指定的地址不变。

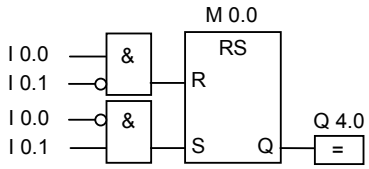
在输入端 R 的信号状态为 1，而输入端 S 的信号状态为 0 时，**复位置位触发器**被复位。如果输入端 R 为 0，而输入端 S 为 1，则此触发器被置位。如果两个输入的 RLO 均为 1，此触发器被置位。

复位置位触发器指令受主控继电器(MCR)的影响。关于 MCR 如何起作用的更多详细信息，请参考 MCR 开/关。

状态字

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

举例

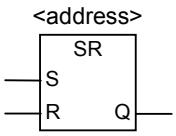


如果 I0.0 为 1 而 I0.1 为 0，则存储位 M0.0 被复位且输出 Q4.0 为 0。如果 I0.0 为 0 而 I0.1 为 1，则存储位 M0.0 被置位且输出 Q4.0 为 1。

如果两个信号状态均为 0，则没有变化。如果两个信号状态均为 1，则置位指令起作用，因为指令次序如此。M 0.0 被置位且 Q4.0 为 1。

1.13 SR：置位复位触发器

符号



参数	数据类型	内存区域	描述
<地址>	BOOL	I、Q、M、D、L	地址指定将要置位或复位的位。
S	BOOL	I、Q、M、D、L、T、C	启用了置位指令
R	BOOL	I、Q、M、D、L、T、C	启用了复位指令
Q	BOOL	I、Q、M、D、L	<地址>的信号状态

描述

置位复位触发器指令仅在 RLO 为 1 时执行“置位”(S)或“复位”(R)指令。RLO 为 0 时对这些指令没有影响，在指令中指定的地址保持不变。

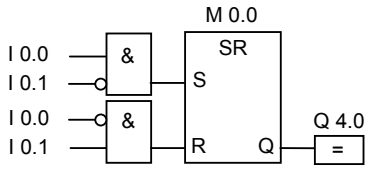
在输入端 S 的信号状态为 1，而输入端 R 的信号状态为 0 时，**置位复位触发器**被置位。如果输入端 S 为 0，而输入端 R 为 1，则触发器被复位。如果两个输入的 RLO 均为 1，则该触发器被复位。

置位复位触发器指令受主控继电器(MCR)的影响。关于 MCR 如何起作用的更多详细信息，请参考 MCR 开/关。

状态字

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

举例

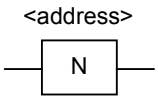


如果 I0.0 为 1 而 I0.1 为 0，则存储位 M0.0 被置位且 Q4.0 为 1。如果 I0.0 为 0 而 I0.1 为 1，则存储位 M0.0 被复位且 Q4.0 为 0。

如果两个信号状态均为 0，则没有变化。如果两个信号状态均为 1，复位指令起作用，因为指令次序如此。M0.0 被复位且 Q4.0 为 0。

1.14 N：RLO 负跳沿检测

符号



参数	数据类型	内存区域	描述
<地址>	BOOL	I、Q、M、D、L	地址指定哪个沿存储位将存储前一个 RLO。

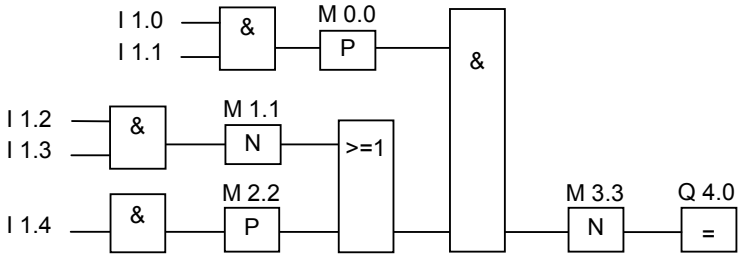
描述

RLO 负跳沿检测指令检测指定地址从 1 到 0 的变化(下降沿)，并在执行指令后以 RLO 为 1 表示此变化。系统会将 RLO 的当前信号状态与相应地址(沿存储位)的信号状态进行比较。如果地址的信号状态为 1，而在执行指令前 RLO 为 0，则执行指令后 RLO 将为 1 (脉冲)，对于其它情况则 RLO 为 0。执行指令之前的 RLO 将存储到该地址中。

状态字

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

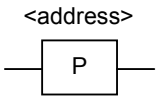
举例



沿存储位 M3.3 存储先前 RLO 的信号状态。

1.15 P：RLO 正跳沿检测

符号



参数	数据类型	内存区域	描述
<地址>	BOOL	I、Q、M、D、L	地址指定哪个沿存储位将存储前一个 RLO。

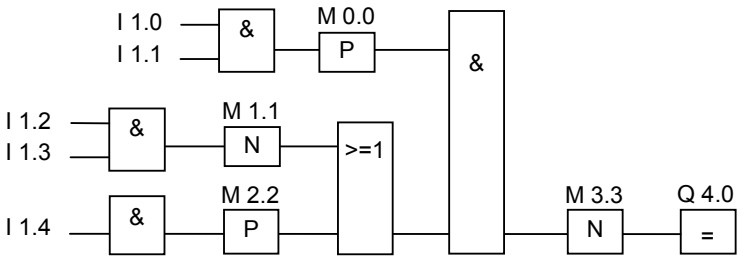
描述

RLO 正跳沿检测指令检测指定地址从 0 到 1 的变化(上升沿)，并在执行该指令后以 RLO 为 1 表示这种变化。系统会将 RLO 的当前信号状态与相应地址(沿存储位)的信号状态进行比较。如果地址的信号状态为 0，而在执行指令前 RLO 为 1，则执行指令后 RLO 将为 1 (脉冲)，对于其它情况则 RLO 为 0。执行指令之前的 RLO 将存储到该地址中。

状态字

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

举例



沿存储位 M3.3 存储先前 RLO 的信号状态。

1.16 SAVE：将 RLO 存入 BR 存储区

符号

描述

将 RLO 存入 BR 存储区指令将 RLO 存入状态字的 BR 位。不会使第一个检查位 FC 复位。

因此，如果在下一个程序段中有“与”逻辑操作，在该逻辑操作中将包含 BR 位的状态。

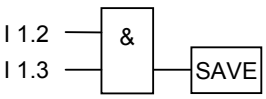
对于指令 **SAVE** (LAD、FBD、STL)，适用下列原则而不是在手册和在线帮助中建议的用途：
我们建议您不要在使用 SAVE 后在同一块或下位块中校验 BR 位，因为这期间执行的指令中有许多会对 BR 位进行修改。建议在退出块前使用 SAVE 指令，因为 ENO 输出(= BR 位)届时已置位为 RLO 位的值，从而用户可以检查该块的出错。

使用**将 RLO 存入 BR 存储区**指令后，程序段的 RLO 可以是从属块中的逻辑操作的一部分。调用块中的 CALL 指令会复位第一个检查位。

状态字

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

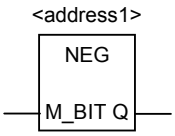
举例



逻辑操作的结果(RLO)将写入 BR 位。

1.17 NEG：地址负跳沿检测

符号



参数	数据类型	内存区域	描述
<地址 1>	BOOL	I、Q、M、D、L	待检查负跳(下降)沿改变的信号。
M_BIT	BOOL	Q、M、D	M_BIT 地址指定 NEG 的前一个信号状态所在的沿存储位。应当仅在尚无输入模块占用过程映像输入区域 I 时，将其用作 M_BIT。
Q	BOOL	I、Q、M、D、L	单触发输出。

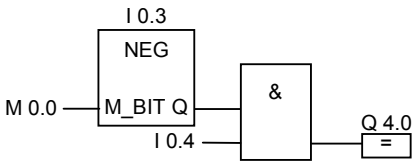
描述

地址负跳沿检测指令比较 <地址 1> 的信号状态与存储在 M_BIT 参数中的前一次检查的信号状态。如果发生了从 1 到 0 的变化，则输出 Q 值为 1，而对于其它情况下则为 0。

状态字

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

举例

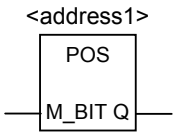


如果满足以下条件，则输出 Q4.0 为 1：

- 输入 I0.3 处存在下降沿
- 并且输入 I0.4 处的信号状态为 1。

1.18 POS：地址正跳沿检测

符号



参数	数据类型	内存区域	描述
<地址 1>	BOOL	I、Q、M、D、L	待检查正跳(上升)沿的信号。
M_BIT	BOOL	Q、M、D	M_BIT 地址指定用于存储 POS 的前一个信号状态的沿存储位。 应当仅在尚无输入模块占用过程映像输入区域 I 时，将其用作 M_BIT。
Q	BOOL	I、Q、M、D、L	单触发输出。

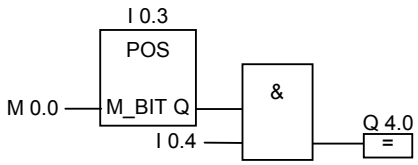
描述

地址正跳沿检测指令比较 <地址 1> 的信号状态与存储在参数 M_BIT 中的前一次信号检查的信号状态。如果发生了从 0 到 1 的变化，则输出 Q 值为 1，而对于其它情况则为 0。

状态字

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

举例



如果满足以下条件，则输出 Q4.0 为 1：

- 输入 I0.3 有上升沿
- 并且输入 I0.4 的信号状态为 1。

2 比较指令

2.1 比较指令概述

描述

根据您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

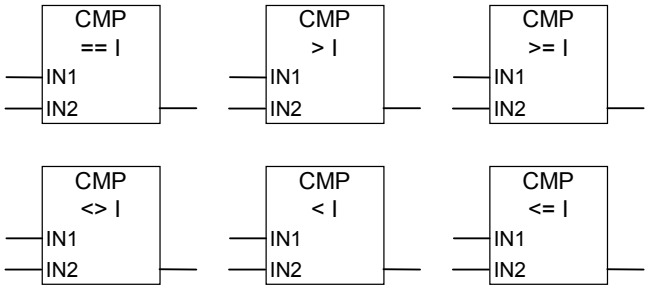
如果比较结果为 true，则函数的 RLO 为“1”。否则，RLO 为 0。您不能对比较结果本身进行取反，但可通过使用相反的比较指令获得与取反同样的效果。

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：整数比较

符号



参数	数据类型	内存区域	描述
IN1	INT	I、Q、M、D、L 或常数	要比较的第一个值
IN2	INT	I、Q、M、D、L 或常数	要比较的第二个值
框输出	BOOL	I、Q、M、D、L	比较结果

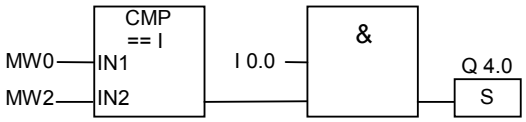
描述

整数比较指令用于在 16 位浮点数的基础上比较两个数值。此指令将根据您从列表框中选定的比较类型比较输入 IN1 和 IN2。

状态字

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

举例

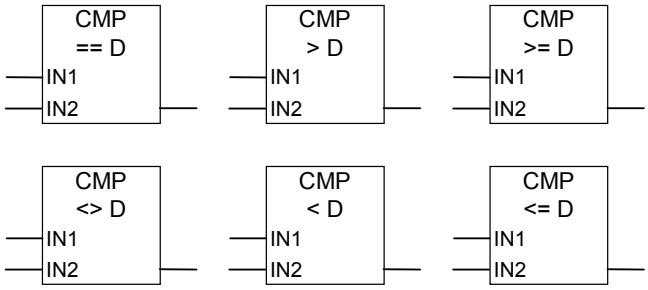


在满足以下条件时会将 Q 4.0 置 1：

- MW0 等于 MW2
- 并且输入 I0.0 的信号状态为 1。

2.3 CMP ?D：比较双精度整数

符号



参数	数据类型	内存区域	描述
IN1	DINT	I、Q、M、D、L 或常数	要比较的第一个值
IN2	DINT	I、Q、M、D 或常数 L	要比较的第二个值
框输出	BOOL	I、Q、M、D、L	比较结果

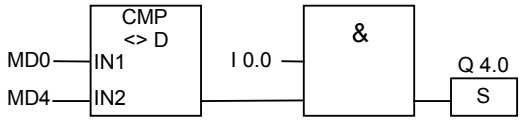
描述

比较双精度整数指令用于在 32 位浮点数的基础上比较两个数值。此指令将根据您从列表框中选定的比较类型比较输入 IN1 和 IN2。

状态字

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

举例

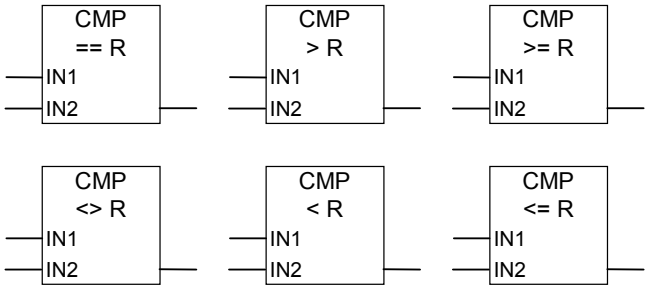


在满足以下条件时会将 Q4.0 置 1：

- MD0 不等于 MD4
- 并且输入 I0.0 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	描述
IN1	REAL	I、Q、M、D、L 或常数	要比较的第一个值
IN2	REAL	I、Q、M、D、L 或常数	要比较的第二个值
框输出	BOOL	I、Q、M、D、L	比较结果

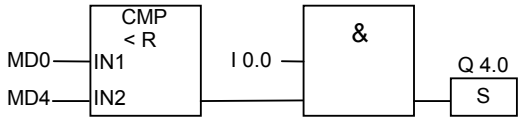
描述

比较实数指令用于比较两个实数的值。此指令将根据您从列表框中选定的比较类型比较输入 IN1 和 IN2。

状态字

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

举例



在满足以下条件时会将 Q 4.0 置 1：

- MD0 小于 MD4
- 并且输入 I0.0 的信号状态为 1。

3 转换指令

3.1 转换指令概述

描述

可使用下列指令将二进制编码的十进制数和整数转换成其它类型的数：

- BCD_I： BCD 转换为整数
- I_BCD： 整型数转换为 BCD
- BCD_DI： BCD 转换为长整数
- I_DI： 整型转换为双精度整型
- DI_BCD： 双精度整数转换为 BCD
- DI_R： 双精度整型转换为实型

可使用下列指令之一求取整数的补码，或改变浮点数的符号：

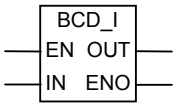
- INV_I： 对整型数求反码
- INV_DI： 二进制反码双精度整型
- NEG_I： 二进制补码整型
- NEG_DI： 二进制补码双精度整型
- NEG_R： 实数取反

可使用下列任一指令将累加器 1 中的 32 位 IEEE 浮点数转换为 32 位整数(双精度整数)。每个指令的取整方法有所不同：

- ROUND： 取整为双精度整型
- TRUNC： 截尾取整数部分
- CEIL： 上限
- FLOOR： 下取整

3.2 BCD_I：BCD 码转换为整型

符号



参数	数据类型	内存区域	描述
EN	BOOL	I、Q、M、D、L、T、C	使能输入
IN	WORD	I、Q、M、D、L 或常数	BCD 格式的数
OUT	INT	I、Q、M、D、L	BCD 数的整数值
ENO	BOOL	I、Q、M、D、L	使能输出

描述

BCD 码转换为整型指令将输入参数 IN 的内容作为二进制编码十进制格式(BCD，“999”)的三位数读取，并将该数转换为整数值。输出参数 OUT 包含转换结果。

ENO 始终与 EN 的信号状态相同。

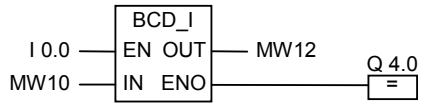
如果在该 BCD 数中任何一个十进制数字处于无效范围 10 和 15 之间，则在尝试转换时会出现 BCD 错误，从而引发下列反应：

- CPU 切换到 STOP 模式。在诊断缓冲区中将输入“BCD 转换错误”，且事件 ID 号为 2521。
- 如果编写了 OB121，则会调用该程序。

状态字

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

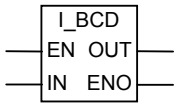
举例



在 I0.0 的信号状态为 1 时执行转换。指令会将存储器字 MW10 的内容作为 BCD 格式的三位数读取，然后转换成一个整数。结果将存入存储器字 MW12 中。执行转换后，输出 Q4.0 的信号状态为 1 (ENO = EN)。

3.3 I_BCD：整型转换为 BCD 码

符号



参数	数据类型	内存区域	描述
EN	BOOL	I、Q、M、D、L、T、C	使能输入
IN	INT	I、Q、M、D、L 或常数	整型
OUT	WORD	I、Q、M、D、L	整数的 BCD 值
ENO	BOOL	I、Q、M、D、L	使能输出

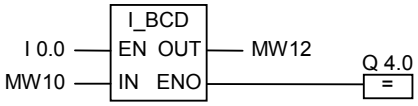
描述

整型数转换为 BCD 指令将输入参数 IN 的内容作为整数值读取并将该值转换为二进制编码十进制格式(BCD，“999”)的三位数。输出参数 OUT 包含转换结果。如果出现溢出，则会将 ENO 置 0。

状态字

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

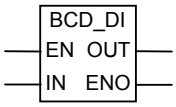
举例



在 I0.0 的信号状态为 1 时执行转换。指令会将存储器字 MW10 的内容作为一个整数，读取然后转换成一个 BCD 格式的三位数。结果将存入存储器字 MW12 中。如果出现溢出，则输出 Q4.0 的信号状态为 0。如果输入 EN 的信号状态为 0 (表示不执行转换)，则输出 Q4.0 的信号状态也为 0。

3.4 BCD_DI：BCD 码转换为双精度整型

符号



参数	数据类型	内存区域	描述
EN	BOOL	I、Q、M、D、L、T、C	使能输入
IN	DWORD	I、Q、M、D、L 或常数	BCD 格式的数
OUT	DINT	I、Q、M、D、L	BCD 数的双精度整型值
ENO	BOOL	I、Q、M、D、L	使能输出

描述

BCD 转换为长整数指令将输入参数 IN 的内容作为二进制编码十进制格式(BCD，“9,999,999”)的七位数读取，并将其转换为双精度整型值。输出参数 OUT 包含转换结果。

ENO 始终与 EN 的信号状态相同。

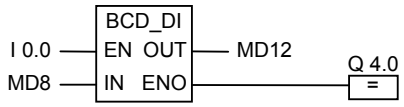
如果在该 BCD 数中任何一个十进制数字处于无效范围 10 和 15 之间，则在尝试转换时会出现 BCD 错误，从而引发下列反应：

- CPU 切换到 STOP 模式。在诊断缓冲区中将输入“BCD 转换错误”，且事件 ID 号为 2521。
- 如果编写了 OB121，则会调用该程序。

状态字

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

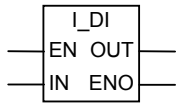
举例



在 I0.0 的信号状态为 1 时执行转换。指令会将存储器双字 MD8 的内容作为 BCD 格式的七位数读取，然后转换成一个双精度整数。结果存储在 MD12 中。执行转换后，输出 Q4.0 的信号状态为 1 (ENO = EN)。

3.5 I_DI：整型转换为双精度整型

符号



参数	数据类型	内存区域	描述
EN	BOOL	I、Q、M、D、L、T、C	使能输入
IN	INT	I、Q、M、D、L 或常数	要转换的值
OUT	DINT	I、Q、M、D、L	结果
ENO	BOOL	I、Q、M、D、L	使能输出

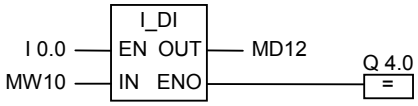
描述

整型转换为双精度整型整型指令将输入参数 IN 的内容作为一个整数读取，并将其转换成一个双精度整数。输出参数 OUT 包含转换结果。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

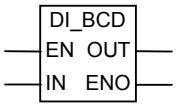
举例



在 I0.0 的信号状态为 1 时执行转换。指令会将存储器字 MW10 的内容作为一个整数读取，然后转换成一个双精度整数。结果将存入存储器双字 MD12 中。执行转换后，输出 Q4.0 的信号状态为 1 (ENO = EN)。

3.6 DI_BCD：双精度整型转换为 BCD 码

符号



参数	数据类型	内存区域	描述
EN	BOOL	I、Q、M、D、L、T、C	使能输入
IN	DINT	I、Q、M、D、L 或常数	双精度整数
OUT	DWORD	I、Q、M、D、L	双精度整数的 BCD 值
ENO	BOOL	I、Q、M、D、L	使能输出

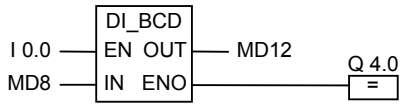
描述

双精度整型转换为 BCD 码指令将输入参数 IN 的内容作为双精度整型值读取，并将此值转换成一个 BCD 格式(“9 999 999”)的七位数。输出参数 OUT 包含转换结果。如果出现溢出，则会将 ENO 置 0。

状态字

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

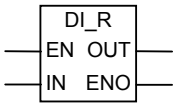
举例



在 I0.0 的信号状态为 1 时执行转换。指令会将存储器双字 MD8 的内容作为一个双精度整数读取，然后转换成一个 BCD 格式的七位数。结果存储在 MD12 中。如果出现溢出，则输出 Q4.0 的信号状态为 0。如果输入 EN 的信号状态为 0 (表示不执行转换)，则输出 Q4.0 的信号状态也为 0。

3.7 DI_R：双精度整型转换为实型

符号



参数	数据类型	内存区域	描述
EN	BOOL	I、Q、M、D、L、T、C	使能输入
IN	DINT	I、Q、M、D、L 或常数	要转换的值
OUT	REAL	I、Q、M、D、L	结果
ENO	BOOL	I、Q、M、D、L	使能输出

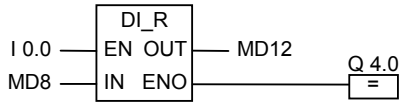
描述

双精度整型转换为实型指令将输入参数 IN 的内容作为双精度整型值读取，并将该值转换成一个实数。输出参数 OUT 包含转换结果。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

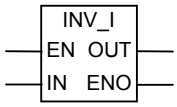
举例



在 I0.0 的信号状态为 1 时执行转换。指令会将存储器双字 MD8 的内容作为一个整数实数读取，然后转换成一个。结果将存入存储器双字 MD12 中。如果不执行转换，则输出 Q4.0 的信号状态为 0 (ENO = EN)。

3.8 INV_I：对整型数求反码

符号



参数	数据类型	内存区域	描述
EN	BOOL	I、Q、M、D、L、T、C	使能输入
IN	INT	I、Q、M、D、L 或常数	输入值
OUT	INT	I、Q、M、D、L	整数的反码
ENO	BOOL	I、Q、M、D、L	使能输出

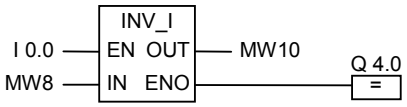
描述

对整型数求反码指令会读取输入参数 IN 的内容，并与字 FFFFH 按位执行一个布尔逻辑异或运算，从而将每一位的值取反。输出参数 OUT 包含转换结果。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

举例



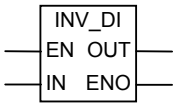
在 I0.0 的信号状态为 1 时执行该转换。对 MW8 中的每一位的值取反：

MW8 = 01000001 10000001 → MW10 = 10111110 01111110

在 I0.0 的信号状态为 0 时不执行该转换，并且 Q4.0 为 0 (ENO = EN)。

3.9 INV_DI：二进制反码双精度整型

符号



参数	数据类型	内存区域	描述
EN	BOOL	I、Q、M、D、L、T、C	使能输入
IN	DINT	I、Q、M、D、L 或常数	输入值
OUT	DINT	I、Q、M、D、L	双精度整数的反码
ENO	BOOL	I、Q、M、D、L	使能输出

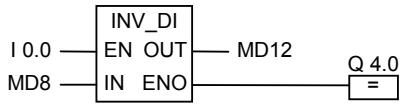
描述

对双精度整数求反码指令会读取输入参数 IN 的内容，并与字 FFFF FFFFH 按位执行一个布尔逻辑异或运算，从而将每一位的值取反。输出参数 OUT 包含转换结果。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

举例



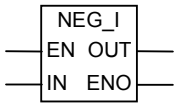
在 I0.0 的信号状态为 1 时执行该转换。对存储器双字 MD8 中的每一位的值取反：

MD8 = F0FF FFF0 → MD12 = 0F00 000F

如果不执行转换，则 Q4.0 为 0 (ENO = EN)。

3.10 NEG_I：二进制补码整型

符号



参数	数据类型	内存区域	描述
EN	BOOL	I、Q、M、D、L、T、C	使能输入
IN	INT	I、Q、M、D、L 或常数	输入值
OUT	INT	I、Q、M、D、L	整数的补码
ENO	BOOL	I、Q、M、D、L	使能输出

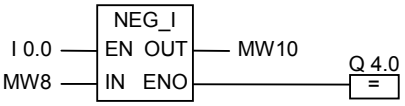
描述

对双精度整数求补码指令会读取输入参数 IN 的内容并改变符号(例如，从正值改变为负值)。输出参数 OUT 包含转换结果。EN 和 ENO 的信号状态始终相同，但在 EN 的信号状态为 1 且产生溢出时除外。这种情况下，ENO 的信号状态为 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

举例



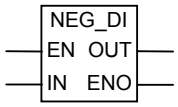
在 I0.0 的信号状态为 1 时执行转换。存储器字 MW8 的值将在 O 位置输出到存储器字 MW10 中，并具有相反的符号：

MW8 = +10 → MW10 = -10

如果 EN 的信号状态为 1 并产生溢出，则 ENO 为 0 并且 Q4.0 的信号状态为 0。如果不执行转换，则 Q4.0 为 0 (ENO = EN)。

3.11 NEG_DI：二进制补码双精度整型

符号



参数	数据类型	内存区域	描述
EN	BOOL	I、Q、M、D、L、T、C	使能输入
IN	DINT	I、Q、M、D、L 或常数	输入值
OUT	DINT	I、Q、M、D、L	双精度整数的补码
ENO	BOOL	I、Q、M、D、L	使能输出

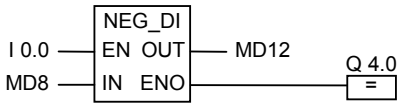
描述

对双精度整数求补码指令会读取输入参数 IN 的内容并改变符号(例如，从正值改变为负值)。输出参数 OUT 包含转换结果。EN 和 ENO 的信号状态始终相同，但在 EN 的信号状态为 1 且产生溢出时除外。这种情况下，ENO 的信号状态为 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

举例



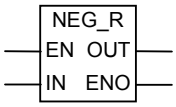
在 I0.0 的信号状态为 1 时执行转换。存储器双字 MD8 的值将在 O 位置输出到存储器双字 MD10 中，并具有相反的符号：

MW8 = +10 → MW10 = -10

如果 EN 的信号状态为 1 并产生溢出，则 ENO 为 0 并且 Q4.0 的信号状态为 0。如果不执行转换，则 Q4.0 为 0 (ENO = EN)。

3.12 NEG_R：实数取反

符号



参数	数据类型	内存区域	描述
EN	BOOL	I、Q、M、D、L、T、C	使能输入
IN	REAL	I、Q、M、D、L 或常数	输入值
OUT	REAL	I、Q、M、D、L	结果是经过取反的输入值。
ENO	BOOL	I、Q、M、D、L	使能输出

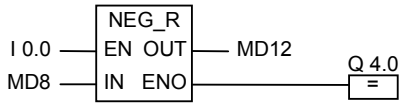
描述

实数取反指令会读取输入参数 IN 的内容并改变符号位(该指令会改变数字的符号。例如，从表示正号的 0 改变为表示负号的 1)。指数位和尾数位保持不变。输出参数 OUT 提供结果。ENO 和 EN 的信号状态始终相同，但在 EN 的信号状态为 1 并且产生溢出时除外。这种情况下，ENO 的信号状态为 0。

状态字

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

举例



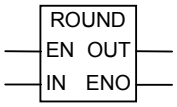
在 I0.0 的信号状态为 1 时执行转换。存储器双字 MD8 的值将在 O 位置输出到存储器双字 MD12 中且符号相反，结果如下例所示：

MD8 = + 6.234 → MD12 = - 6.234

如果不执行转换，则输出 Q4.0 的信号状态为 0 (ENO = EN)。

3.13 ROUND：取整为双精度整型

符号



参数	数据类型	内存区域	描述
EN	BOOL	I、Q、M、D、L、T、C	使能输入
IN	REAL	I、Q、M、D、L 或常数	要取整的值
OUT	DINT	I、Q、M、D、L	将 IN 的值取整为最接近的双精度整数
ENO	BOOL	I、Q、M、D、L	使能输出

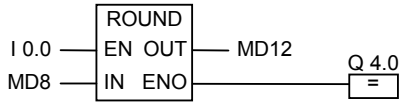
描述

取整为双精度整型指令将输入参数 IN 的内容作为一个实数读取，并将其转换为一个双精度整型数。结果得到最接近的整数，并包含在输出参数 OUT 中。如果小数是 x.5，则会将其取为最接近的偶数(例如：2.5 -> 2，1.5 -> 2)。 如果产生溢出，则会将 ENO 置 0。如果输入值不是实数，则 OV 位和 OS 位的值均为 1，并且 ENO 值为 0。

状态字

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

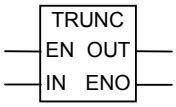
举例



在 I0.0 为 1 时执行转换。指令会将存储器双字 MD8 的内容作为一个实数读取，然后将其转换成一个双精度整数。该“取整为最接近整数”功能的结果将存入存储器双字 MD12 中。如果出现溢出，则输出 Q4.0 的信号状态为 0。如果输入 EN 的信号状态为 0 (表示不执行转换)，则输出 Q4.0 的信号状态也为 0。

3.14 TRUNC：截尾取整数部分

符号



参数	数据类型	内存区域	描述
EN	BOOL	I、Q、M、D、L、T、C	使能输入
IN	REAL	I、Q、M、D、L 或常数	要截尾的值
OUT	DINT	I、Q、M、D、L	IN 的整数部分
ENO	BOOL	I、Q、M、D、L	使能输出

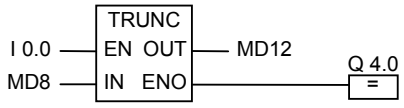
描述

截尾取整数部分指令将输入参数 IN 的内容作为一个实数读取，并将此数转换成一个双精度整型数(例如 1.5 转成 1)。结果得到实数的整数部分。输出参数 OUT 包含转换结果。如果产生溢出，则会将 ENO 置 0。如果输入值不是实数，则 OV 位和 OS 位的值均为 1，并且 ENO 值为 0。

状态字

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

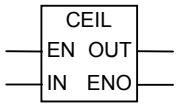
举例



在 I0.0 的信号状态为 1 时执行转换。指令会将存储器双字 MD8 的内容作为实数读取，并根据“向零取整的原则”将该实数转换成双精度整型数。结果得到整数部分，并存入存储器双字 MD12 中。如果出现溢出，则输出 Q4.0 的信号状态为 0。如果输入 EN 的信号状态为 0 (表示不执行转换)，则输出 Q4.0 的信号状态也为 0。

3.15 CEIL：上限

符号



参数	数据类型	内存区域	描述
EN	BOOL	I、Q、M、D、L、T、C	使能输入
IN	REAL	I、Q、M、D、L 或常数	要转换的值
OUT	DINT	I、Q、M、D、L	结果
ENO	BOOL	I、Q、M、D、L	使能输出

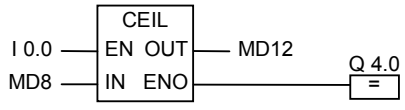
描述

上取整指令将输入参数 IN 的内容作为一个实数读取，并将其转换为一个双精度整型数(例如：+1.2 -> +2；-1.5 -> -1)。结果是大于或等于所指定的实数的最小整数。输出参数 OUT 包含转换结果。如果产生溢出，则 ENO 为 0。如果输入值不是实数，则 OV 位和 OS 位的值均为 1，并且 ENO 值为 0。

状态字

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

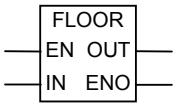
举例



在 I0.0 为 1 时执行转换。指令将存储器双字 MD8 的内容作为一个实数读取，并按照取整为临近的较大(或相等)整数的方式将其转换为一个双精度整型数。结果将存入存储器双字 MD12 中。如果出现溢出，则输出 Q4.0 的信号状态为 0。如果输入 EN 的信号状态为 0 (表示不执行转换)，则输出 Q4.0 的信号状态也为 0。

3.16 FLOOR：下取整

符号



参数	数据类型	内存区域	描述
EN	BOOL	I、Q、M、D、L、T、C	使能输入
IN	REAL	I、Q、M、D、L 或常数	要转换的值
OUT	DINT	I、Q、M、D、L	结果
ENO	BOOL	I、Q、M、D、L	使能输出

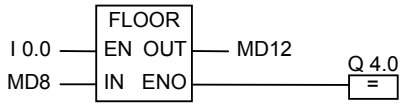
描述

下取整指令将输入参数 IN 的内容作为一个实数读取，并将其转换成一个双精度整数。结果是小于或等于所指定的实数的最大整数。输出参数 OUT 包含转换结果。如果产生溢出，则会将 ENO 置 0。如果输入值不是实数，则 OV 位和 OS 位的值均为 1，并且 ENO 值为 0。

状态字

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

举例



在 I0.0 为 1 时执行转换。指令将存储器双字 MD8 的内容作为一个实数读取，并按照取整为临近的较小(或相等)整数的方式将其转换为一个双精度整型数。结果将存入存储器双字 MD12 中。如果出现溢出，则输出 Q4.0 的信号状态为 0。如果输入 EN 的信号状态为 0 (表示不执行转换)，则输出 Q4.0 的信号状态也为 0。

4 计数器指令

4.1 计数器指令概述

存储器中的区域

在您的 CPU 的存储器中，有为计数器保留的区域。此存储区为每个计数器地址保留一个 16 位字。若是使用 FBD 编程，共支持 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。

可使用下列计数器指令在此范围内改变计数值：

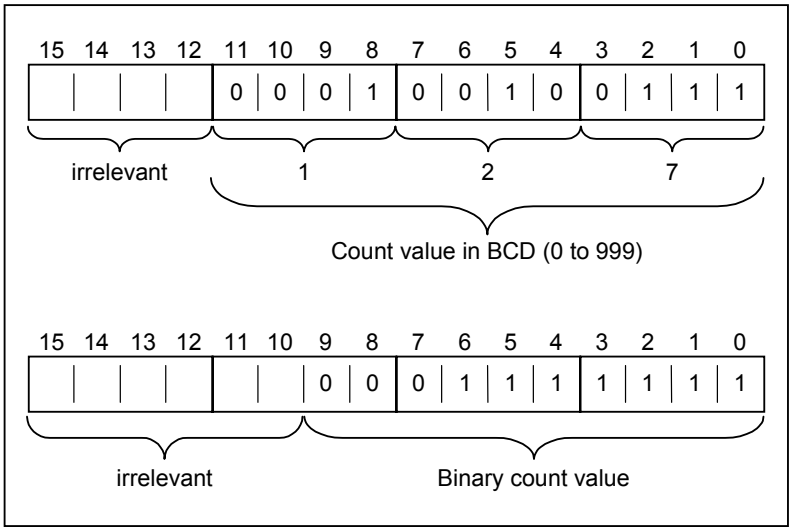
- S_CUD： 分配参数和递增/递减计数
- S_CU： 分配参数和递增计数
- S_CD： 分配参数和递减计数
- SC： 设置计数器值
- CU： 值加计数器
- CD： 减计数器

计数器中的位组态

输入从 0 至 999 的数字，可为计数器提供预设值，例如，使用下列格式提供 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，包含一个十进制值的二进制代码)。

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。

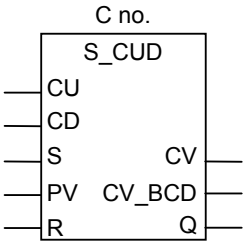
下图显示了装入计数值 127 之后计数器的内容，以及设置计数器之后计数器单元中的内容。



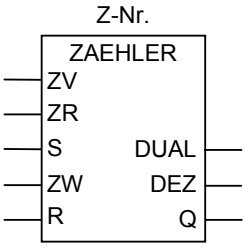
4.2 S_CUD：分配参数和递增/递减计数

符号

English



German



参数 英语	参数 德语	数据类型	内存区域	描述
no.	Nr.	COUNTER	C	计数器标识号。 范围取决于 CPU。
CU	ZV	BOOL	I、Q、M、D、L	ZV 输入端：值加计数器
CD	ZR	BOOL	I、Q、M、D、L	ZR 输入端：值减计数器
S	S	BOOL	I、Q、M、D、L、T、C	用于预置计数器的输入端
PV	ZW	WORD	I、Q、M、D、L 或常数	在 0 和 999 范围之间的计数值 或 以 BCD 格式输入为 C#<值> 形式的计数值
R	R	BOOL	I、Q、M、D、L、T、C	复位输入端
CV	DUAL	WORD	I、Q、M、D、L	当前计数值(十六进制数)
CV_BCD	DEZ	WORD	I、Q、M、D、L	当前计数值(BCD 格式)
Q	Q	BOOL	I、Q、M、D、L	计数器状态

描述

在输入端 S 为上升沿(信号状态从 0 变化为 1)时，**分配参数和递增/递减计数**指令将使用“预设值”(PV)输入端的值设置计数器。如果输入端 CU 的信号状态从 0 变化到 1 (上升沿)，并且计数器值小于 999，则计数器加 1。如果输入端 CD 的信号状态从 0 变化到 1 (上升沿)，并且计数器值大于 0，则计数器减 1。如果此两个计数输入端都存在一个上升沿，则会执行相应的两个操作，从而计数值保持不变。

如果设置了计数器，并且输入端 CU/CD 的 RLO = 1，则即使没有发生从正跳沿到负跳沿的变化或与之相反的变化，计数器也因此将在下一个扫描周期计数。

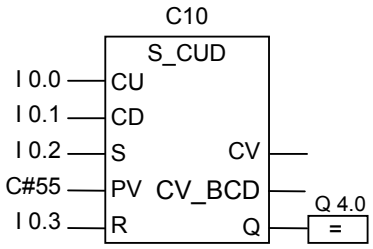
在输入端 R 的信号为“1”时会复位计数器。复位计数器时会将计数值设置为 0。

对于判断输出端 Q 是否为 1 的信号状态检查，如果计数值大于 0，则检查结果为 1；但如果计数值等于 0，检查结果则为 0。

状态字

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

举例



在输入 I0.2 的信号状态从 0 变化为 1 时会将计数器 C10 的值设置为 55。如果输入 I0.0 的信号状态从 0 变化为 1，除非计数器 C10 的值已经是 999，否则计数器 C10 的值加 1。如果输入 I0.1 从 0 变化为 1，除非计数器 C10 的值已经是 0，否则计数器 C10 减 1。如果 I0.3 从 0 变化为 1，则会将 C10 的值设置为 0。当 C10 不等于 0 时，Q4.0 为 1。

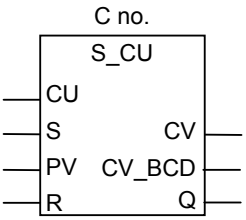
注意

避免在多个程序点上使用同一个计数器(否则会出现计数错误)。

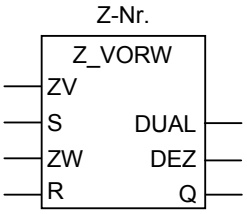
4.3 S_CU：分配参数和递增计数

符号

English



German



参数 英语	参数 德语	数据类型	内存区域	描述
no.	Nr.	COUNTER	C	计数器标识号。 范围取决于 CPU。
CU	ZV	BOOL	I、Q、M、 D、L	ZV 输入端：值加计数器
S	S	BOOL	I、Q、M、 D、L、T、C	用于预置计数器的输入端
PV	ZW	WORD	I、Q、M、 D、L 或常数	在 0 和 999 范围之间的计数值 或 以 BCD 格式输入为 C#<值> 形式的计数值
R	R	BOOL	I、Q、M、 D、L、T、C	复位输入端
CV	DUAL	WORD	I、Q、M、 D、L	当前计数值(十六进制数)
CV_BCD	DEZ	WORD	I、Q、M、 D、L	当前计数值(BCD 格式)
Q	Q	BOOL	I、Q、M、 D、L	计数器状态

描述

在输入端 S 为上升沿(信号状态从 0 变化为 1)时，**分配参数和递增计数**指令将使用“预设值”(PV)输入端的值设置计数器。如果输入端 CU 是一个上升沿，则可在计数值小于 999 的情况下使计数值加 1。

如果设置了计数器，并且输入端 CU 的 RLO = 1，则即使没有发生从正跳沿到负跳沿的变化或与之相反的变化，计数器也因此将在下一个扫描周期计数。

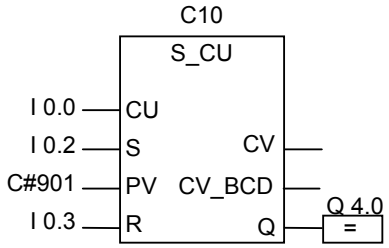
在输入 R 的信号为“1”时会复位计数器。复位计数器时会将计数值设置为 0。

对于判断输出端 Q 是否为 1 的信号状态检查，如果计数值大于 0，则检查结果为 1；但如果计数值等于 0，检查结果则为 0。

状态字

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

举例



在输入 I0.2 的信号状态从 0 变化为 1 时会将计数器 C10 的值设置为 901。 如果输入 I0.0 的信号状态从 0 变化为 1，除非计数器 C10 的值已经是 999，否则计数器 C10 的值加 1。如果 I0.3 从 0 变化为 1，则会将 C10 的值设置为 0。当 C10 不等于 0 时，输出 Q4.0 的信号状态为 1。

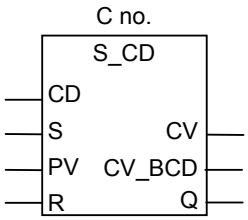
注意

避免在多个程序点使用同一个计数器(否则会出现计数错误)。

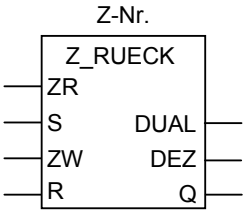
4.4 S_CD：分配参数和递减计数

符号

English



German



参数 英语	参数 德语	数据类型	内存区域	描述
no.	Nr.	COUNTER	C	计数器标识号。 范围取决于 CPU。
CD	ZR	BOOL	I、Q、M、D、L	CD 输入：值减计数器
S	S	BOOL	I、Q、M、D、L、T、C	用于预置计数器的输入端
PV	ZW	WORD	I、Q、M、D、L 或常数	在 0 和 999 范围之间的计数值 或 以 BCD 格式输入为 C#<值> 形式的计数值
R	R	BOOL	I、Q、M、D、L、T、C	复位输入端
CV	DUAL	WORD	I、Q、M、D、L	当前计数值(十六进制数)
CV_BCD	DEZ	WORD	I、Q、M、D、L	当前计数值(BCD 格式)
Q	Q	BOOL	I、Q、M、D、L	计数器状态

描述

在输入端 S 为上升沿(信号状态从 0 变化为 1)时，**分配参数和递减计数**指令将使用“预设值”(PV)输入端的值设置计数器。如果输入端 CD 是一个上升沿，则可在计数值大于 0 的情况下使计数值减 1。

如果设置了计数器，并且输入端 CD 的 RLO = 1，则即使没有发生从正跳沿到负跳沿的变化或与之相反的变化，计数器也因此将在下一个扫描周期计数。

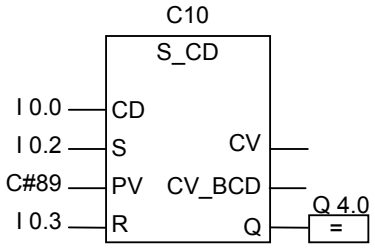
在输入 R 的信号为“1”时会复位计数器。复位计数器时会将计数值设置为 0。

对于判断输出端 Q 是否为 1 的信号状态检查，如果计数值大于 0，则检查结果为 1；但如果计数值等于 0，检查结果则为 0。

状态字

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

举例



在输入 I0.2 的信号状态从 0 变化为 1 时会将计数器 C10 的值设置为 89。如果输入 I0.0 的信号状态从 0 变化为 1，除非计数器 C10 的值等于 0，否则计数器 C10 的值减 1。如果 I0.3 从 0 变化为 1，则会将 C10 的值设置为 0。

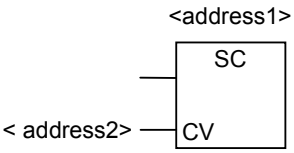
注意

避免在多个程序点上使用同一个计数器(否则会出现计数错误)。

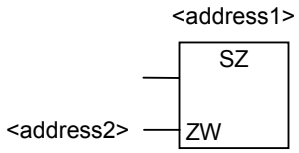
4.5 SC：设置计数器值

符号

English



German



参数 英语	参数 德语	数据类型	内存区域	描述
计数器编号	计数器编号	COUNTER	C	地址 1 指定要预设值的计数器的编号。
CV	ZW	WORD	I、Q、 M、D、L 或常数	预设的值(地址 2)可介于 0 到 999 之间。在输入常数时，C#必须位于指定为 BCD 格式的值之前，例如 C#100。

描述

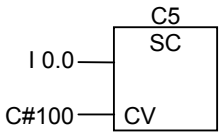
设置计数器值指令将为指定的计数器分配一个预设值。仅当在 RLO 中存在一个上升沿(信号状态从 0 变化为 1)时，才会执行此指令。

只能将设置计数器值框放在一个逻辑指令的右端。可以使用多个**设置计数器值**框。

状态字

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

举例



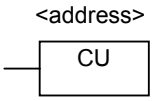
在输入 I0.0 的信号状态从 0 变化为 1 (RLO 中的上升沿)时将计数器 C5 的值预设为 100。C#表示您将输入一个 BCD 格式的值。

如果不存在上升沿，则 C5 的计数器值不变。

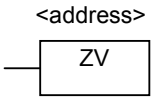
4.6 CU：值加计数器

符号

English



German



参数	数据类型	内存区域	描述
计数器编号	COUNTER	C	该地址指定将增加计数的计数器的编号。

描述

如果 RLO 中存在一个上升沿(信号状态从 0 变化为 1)，并且计数器的值小于 999，则**值加计数器**指令将使指定计数器的值加 1。如果 RLO 中没有上升沿，或计数器的值已经是 999，则不增加值。

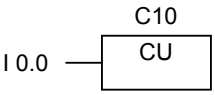
设置计数器值指令可设置计数器的值。

只能在逻辑指令串的右端放置**值加计数器**框。可以使用许多**值加计数器**框。

状态字

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

举例



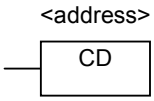
如果输入 I0.0 的信号状态从 0 变化为 1 (RLO 中存在上升沿)，则计数器 C10 的值增加 1 (在 C10 值已经是 999 时例外)。

如果不存在上升沿，则 C10 的值保持不变。

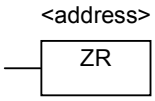
4.7 CD：值减计数器

符号

English



German



参数	数据类型	内存区域	描述
计数器编号	COUNTER	C	该地址指定将递减的计数器的编号。

描述

如果 RLO 中存在一个上升沿(信号状态从 0 变化为 1)，并且计数器的值大于 0，则**值减计数器**指令使指定计数器的值减 1。如果 RLO 中没有上升沿或计数器的值已经是 0，则不递减值。

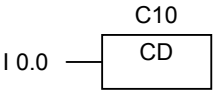
设置计数器值指令可设置计数器的值。

只能在逻辑指令串的右端放置**值减计数器**框。可以使用许多**值减计数器**框。

状态字

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

举例



如果输入 I0.0 的信号状态从 0 变化为 1 (RLO 中存在上升沿)，则计数器 C10 的值减 1 (在 C10 值已经是 0 时例外)。

如果不存在上升沿，则 C10 的值保持不变。

5 数据块指令

5.1 OPN：打开数据块

符号

<DB-Number> or
<DI-Number>

OPN

参数	数据类型	内存区域	描述
DB 或 DI 编号	BLOCK_DB	-	DB 或 DI 编号；范围取决于 CPU。

描述

可以使用**打开数据块**指令将现有数据块作为共享数据块(DB)或背景数据块(DI)打开。数据块的编号将传送至 DB 或 DI 寄存器。后续的 DB 和 DI 命令将根据寄存器内容访问相应的块。

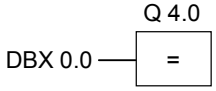
状态字

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

举例

程序段 1

程序段 2



DB10 是当前打开的数据块。因此 DBX0.0 的扫描引用数据块 DB10 的数据字节 0 的 0 位。此位的信号状态分配给输出 Q 4.0。

6 跳转指令

6.1 跳转指令概述

描述

可以在所有逻辑块中使用本指令，例如在组织块(OB)、功能块(FB)和功能(FC)中。

以下是可用的跳转指令：

- JMP 块中无条件跳转
- JMP 块中有条件跳转
- JMPN 若非则跳转

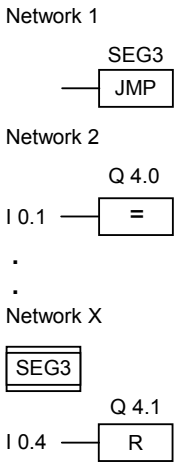
跳转标签作为地址

跳转指令的地址是标签。跳转标签指示想让程序跳转到的目标。

在 JMP 框上方输入标签。标签最多可以包含四个字符。首字符必须为字母；其它字符可以是字母或数字(例如，SEG3)。

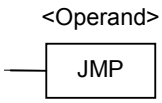
跳转标签作为目标

目标标签必须位于程序段的开头。要在程序段开头输入目标标签，可以通过从 FBD 列表框中选择 LABEL 一个空框将会出现。.一个空框将会出现。在框中键入标签的名称。



6.2 JMP：块中无条件跳转

符号



参数	数据类型	内存区域	描述
跳转标签的名称	-	-	该地址指定程序将无条件跳转到的位置的标签。

描述

块中无条件跳转指令相当于“跳转到标签”指令。不执行跳转指令和标签之间的所有指令。

可以在所有逻辑块中使用本指令，例如在组织块(OB)、功能块(FB)和功能(FC)中。

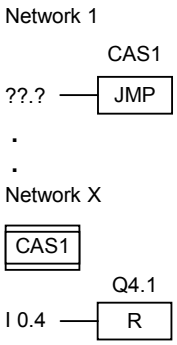
在**块中无条件跳转**框之前，不能有任何逻辑运算。

状态字

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

该指令不改变状态字的位。

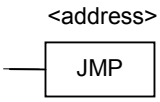
举例



跳转无条件执行。不执行跳转指令和标签之间的所有指令。

6.3 JMP：块中有条件跳转

符号



参数	数据类型	内存区域	描述
跳转标签的名称	-	-	地址指定在 RLO 为 1 时程序将跳转到的位置的位置的标签。

描述

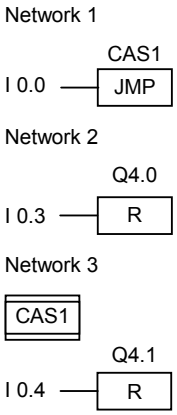
如果 ROL 为 1，则**块中有条件跳转**指令相当于“跳转到标签”指令。FBD 元素“无条件跳转”也可用于此操作，但本指令需要基于之前的逻辑运算来执行。仅当此逻辑运算的结果为 1 时才执行该有条件跳转指令。不执行该跳转操作和标签之间的所有指令。

可以在所有逻辑块中使用本指令，例如在组织块(OB)、功能块(FB)和功能(FC)中。

状态字

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

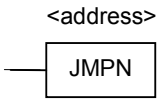
举例



在输入 I0.0 的信号状态为 1 时执行跳转到标签 CAS1。即使输入 I0.3 的信号状态为 1，也不会执行用于复位输出端 Q4.0 的指令。

6.4 JMPN：若非则跳转

符号



参数	数据类型	内存区域	描述
跳转标签的名称	-	-	地址指定在 RLO 为 0 时程序将跳转到的标签。

描述

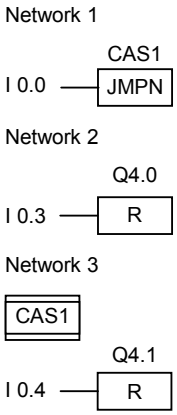
若非则跳转指令相当于在 RLO 为 0 时执行的“跳转到标签”指令

可以在所有逻辑块中使用本指令，例如在组织块(OB)、功能块(FB)和功能(FC)中。

状态字

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

举例



在输入 I0.0 的信号状态为 0 时执行跳转到标签 CAS1。即使输入 I0.3 的信号状态为 1，也不会执行用于复位输出端 Q4.0 的指令。

不执行该跳转指令及其标签间的所有指令。

6.5 LABEL：跳转标签

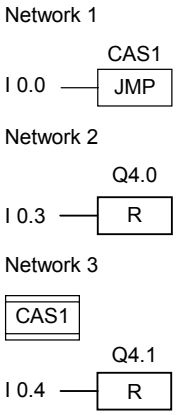
符号

描述

跳转标签是跳转指令目标的标识符。标签最多可以包含四个字符。首字符必须为字母；其它字符可以是字母或数字(例如，CAS1)。

跳转或若非则跳转指令(**JMP** 或 **JMPN**)必须有跳转标签。

举例



在 I0.0 = 1 时执行跳转到标签 CAS1。

由于跳转的原因，即便 I0.3 的信号状态为 1，也不会执行 Q4.0 处的“复位输出”操作。

7 整数算术运算指令

7.1 整数算术运算指令概述

描述

使用整数算术运算，您可以对**两个整数**(16 和 32 位)执行以下运算：

- ADD_I： 加整型
- SUB_I： 减整型
- MUL_I： 乘整型
- DIV_I： 除整型
- ADD_DI： 加双精度整型
- SUB_DI： 减双精度整型
- MUL_DI： 乘双精度整型
- DIV_DI： 除双精度整型
- MOD_DI：返回分数双精度整型

参见使用整数算术运算指令计算状态字的位

7.2 使用整数算术运算指令计算状态字的位

描述

整数算术运算指令影响状态字中的下列位：CC 1 和 CC 0，OV 和 OS。

下表显示整数(16 位和 32 位)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	*

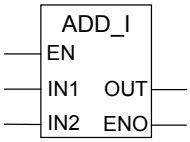
* 指令的结果不影响 OS 位。

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：加整型

符号



参数	数据类型	内存区域	描述
EN	BOOL	I、Q、M、D、L、T、C	使能输入
IN1	INT	I、Q、M、D、L 或常数	加法的第一个值
IN2	INT	I、Q、M、D、L 或常数	加法的第二个值
OUT	INT	I、Q、M、D、L	相加的结果
ENO	BOOL	I、Q、M、D、L	使能输出

描述

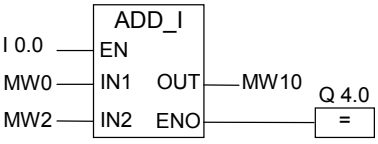
在使能(EN)输入的信号状态为 1 时会激活**整数加法**指令。该指令将输入 IN1 与 IN2 相加。可在 OUT 处扫描结果。若结果超出了整型数的允许范围，则状态字的 OV 和 OS 位会被置 1，且 ENO 被置 0。

参见使用整数算术运算指令计算状态字的位

状态字

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

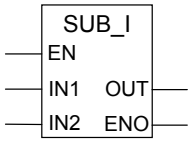
举例



在输入 I0.0 的信号状态为 1 时会激活 ADD_I 框。将 MW0 + MW2 的结果输入到存储器字 MW10 中。若结果超出整型数的允许范围或输入 I0.0 的信号状态为 0，则会将输出 Q4.0 置 0，并且不执行该指令。

7.4 SUB_I：减整型

符号



参数	数据类型	内存区域	描述
EN	BOOL	I、Q、M、D、L、T、C	使能输入
IN1	INT	I、Q、M、D、L 或常数	被减数(将从中减去第二个值的值)
IN2	INT	I、Q、M、D、L 或常数	减数(从第一个值中减去的值)
OUT	INT	I、Q、M、D、L	相减的结果
ENO	BOOL	I、Q、M、D、L	使能输出

描述

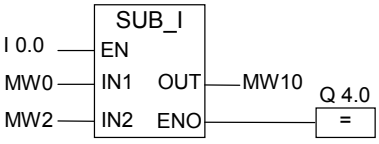
使能(EN)输入的信号状态为 1 时会激活**整数减法**指令。该指令从输入 IN1 减去输入 IN2。可在 OUT 处扫描结果。若结果超出整型数的允许范围，则状态字的 OV 和 OS 位会被置 1，且 ENO 被置 0。

参见使用整数算术运算指令计算状态字的位

状态字

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

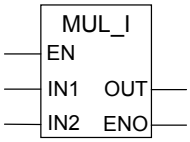
举例



在输入 I0.0 的信号状态为 1 时会激活 SUB_I 框。减法 MW0 - MW2 的结果将输入到存储器字 MW10 中。若结果超出整型数的允许范围或输入 I0.0 的信号状态为 0，则会将输出 Q4.0 置 0，并且不执行该指令。

7.5 MUL_I：乘整型

符号



参数	数据类型	内存区域	描述
EN	BOOL	I、Q、M、D、L、T、C	使能输入
IN1	INT	I、Q、M、D、L 或常数	被乘数(被第二个值乘的值)
IN2	INT	I、Q、M、D、L 或常数	乘数(乘第一个值的值)
OUT	INT	I、Q、M、D、L	相乘的结果
ENO	BOOL	I、Q、M、D、L	使能输出

描述

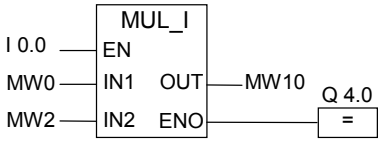
使能(EN)输入的信号状态为 1 时会激活**整数乘法**指令。该指令用输入 IN2 乘输入 IN1。其结果是可在 OUT 处扫描的 32 位整型数。若结果超出 16 位整型数的允许范围，则状态字的 OV 和 OS 位会被置 1，且 ENO 被置 0。

参见使用整数算术运算指令计算状态字的位

状态字

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

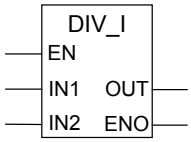
举例



在输入 I0.0 的信号状态为 1 时会激活 MUL_I 框。乘法 MW0 x MW2 的结果将输入到存储器字 MW10 中。若结果超出 16 位整型数的允许范围或输入 I0.0 的信号状态为 0，则会将输出 Q4.0 置 0，并且不执行该指令。

7.6 DIV_I：除整型

符号



参数	数据类型	内存区域	描述
EN	BOOL	I、Q、M、D、L、T、C	使能输入
IN1	INT	I、Q、M、D、L 或常数	被除数
IN2	INT	I、Q、M、D、L 或常数	除数
OUT	INT	I、Q、M、D、L	相除的结果
ENO	BOOL	I、Q、M、D、L	使能输出

描述

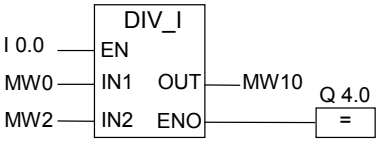
在使能(EN)输入的信号状态为 1 时会激活**整数除法**指令。该指令用输入 IN2 除输入 IN1。可在 OUT 处扫描整数商值(整数部分的结果)。但不能扫描余数。如果商超出整型数的允许范围，则状态字的 OV 和 OS 位会被置 1，且 ENO 被置 0。

参见使用整数算术运算指令计算状态字的位

状态字

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

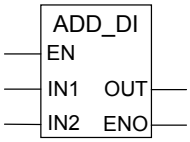
举例



在输入 I0.0 的信号状态为 1 时会激活 DIV_I 框。MW0 除以 MW2 的商将输入到存储器字 MW10 中。如果商超出整型数的允许范围或输入 I0.0 的信号状态为 0，则会将输出 Q4.0 置 0，并且不执行该指令。

7.7 ADD_DI：加双精度整型

符号



参数	数据类型	内存区域	描述
EN	BOOL	I、Q、M、D、L、T、C	使能输入
IN1	DINT	I、Q、M、D、L 或常数	加法的第一个值
IN2	DINT	I、Q、M、D、L 或常数	加法的第二个值
OUT	DINT	I、Q、M、D、L	相加的结果
ENO	BOOL	I、Q、M、D、L	使能输出

描述

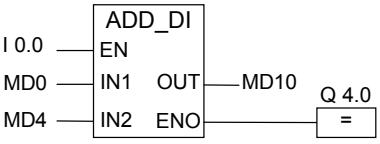
使能(EN)输入的信号状态为 1 时会激活**双精度整数加法**指令。该指令将输入 IN1 与 IN2 相加。可在 OUT 处扫描结果。若结果超出双精度整型数的允许范围，则状态字的 OV 和 OS 位会被置 1，且 ENO 被置 0。

参见使用整数算术运算指令计算状态字的位

状态字

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

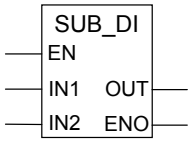
举例



在输入 I0.0 的信号状态为 1 时会激活 ADD_DI 框。将 MD0 + MD4 的结果存入存储器双字 MD10 中。若结果超出双精度整型数的允许范围或输入 I0.0 的信号状态为 0，则会将输出 Q4.0 置 0，并且不执行该指令。

7.8 SUB_DI：减双精度整型

符号



参数	数据类型	内存区域	描述
EN	BOOL	I、Q、M、D、L、T、C	使能输入
IN1	DINT	I、Q、M、D、L 或常数	被减数(将从中减去第二个值的值)
IN2	DINT	I、Q、M、D、L 或常数	减数(从第一个值中减去的值)
OUT	DINT	I、Q、M、D、L	相减的结果
ENO	BOOL	I、Q、M、D、L	使能输出

描述

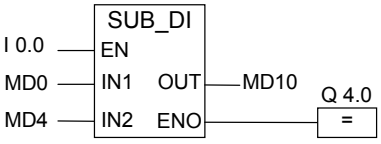
使能(EN)输入的信号状态为 1 时会激活**双精度整数减法**指令。该指令从输入 IN1 减去输入 IN2。可在 OUT 处扫描结果。若结果超出双精度整型数的允许范围，则状态字的 OV 和 OS 位会被置 1，且 ENO 被置 0。

参见使用整数算术运算指令计算状态字的位

状态字

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

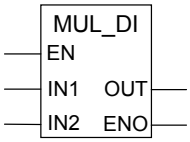
举例



在输入 I0.0 的信号状态为 1 时会激活 SUB_DI 框。将 MD0 - MD4 的结果存入存储器双字 MD10 中。若结果超出双精度整型数的允许范围或输入 I0.0 的信号状态为 0，则会将输出 Q4.0 置 0，并且不执行该指令。

7.9 MUL_DI：乘双精度整型

符号



参数	数据类型	内存区域	描述
EN	BOOL	I、Q、M、D、L、T、C	使能输入
IN1	DINT	I、Q、M、D、L 或常数	被乘数(被第二个值乘的值)
IN2	DINT	I、Q、M、D、L 或常数	乘数(乘第一个值的值)
OUT	DINT	I、Q、M、D、L	相乘的结果
ENO	BOOL	I、Q、M、D、L	使能输出

描述

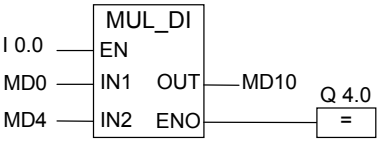
在使能(EN)输入的信号状态为 1 时会激活**双精度整数乘法**指令。此指令将输入 IN1 与 IN2 相乘。可在 OUT 处扫描结果。若结果超出双精度整型数的允许范围，则状态字的 OV 和 OS 位会被置 1，且 ENO 被置 0。

参见使用整数算术运算指令计算状态字的位

状态字

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

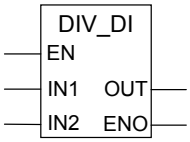
举例



在输入 I0.0 的信号状态为 1 时会激活 MUL_DI 框。将 MD0 x MD4 的结果存入存储器双字 MD10 中。若结果超出双精度整型数的允许范围或输入 I0.0 的信号状态为 0，则会将输出 Q4.0 置 0，并且不执行该指令。

7.10 DIV_DI：除双精度整型

符号



参数	数据类型	内存区域	描述
EN	BOOL	I、Q、M、D、L、T、C	使能输入
IN1	DINT	I、Q、M、D、L 或常数	被除数
IN2	DINT	I、Q、M、D、L 或常数	除数
OUT	DINT	I、Q、M、D、L	相除的结果
ENO	BOOL	I、Q、M、D、L	使能输出

描述

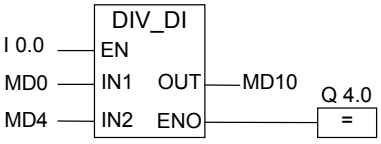
在使能(EN)输入的信号状态为 1 时会激活**双精度整数除法**指令。该指令用输入 IN2 除输入 IN1。可在 OUT 处扫描商(整数部分的结果)。双精度整数除法指令以 DINT 格式将商存储为 32 位单精度值。此指令不生成余数。如果商超出双精度整型数的允许范围，则状态字的 OV 和 OS 位会被置 1，且 ENO 被置 0。

参见使用整数算术运算指令计算状态字的位

状态字

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

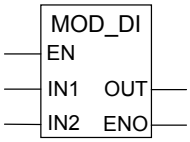
举例



在输入 I0.0 的信号状态为 1 时会激活 DIV_DI 框。MD0 除以 MD4 的商将输入到存储器双字 MD10 中。如果商超出双精度整型数的允许范围或输入 I0.0 的信号状态为 0，则会将输出 Q4.0 置 0，并且不执行该指令。

7.11 MOD_DI：返回分数双精度整型

符号



参数	数据类型	内存区域	描述
EN	BOOL	I、Q、M、D、L、T、C	使能输入
IN1	DINT	I、Q、M、D、L 或常数	被除数
IN2	DINT	I、Q、M、D、L 或常数	除数
OUT	DINT	I、Q、M、D、L	余数
ENO	BOOL	I、Q、M、D、L	使能输出

描述

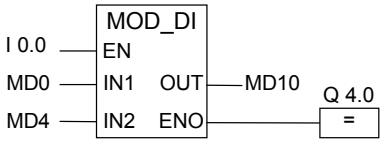
在使能(EN)输入的信号状态为 1 时会激活**返回双精度整数余数**指令。该指令用输入 IN2 除输入 IN1。可在 OUT 处扫描余数(分数)。若结果超出双精度整型数的允许范围，则状态字的 OV 和 OS 位会被置 1，且 ENO 被置 0。

参见使用整数算术运算指令计算状态字的位

状态字

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

举例



在输入 I0.0 的信号状态为 1 时会激活 MOD_DI 框。MD0 除以 MD4 的余数(分数)将存储在存储器双字 MD10 中。若结果超出双精度整型数的允许范围或输入 I0.0 的信号状态为 0，则会将输出 Q4.0 置 0，并且不执行该指令。

8 浮点算术运算指令

8.1 浮点数数学运算概述

描述

IEEE 32 位浮点数属于称为 REAL 的数据类型。可以使用浮点算术运算指令对两个 32 位 IEEE 浮点数执行下列数学指令：

- ADD_R： 加实型
- SUB_R： 减实型
- MUL_R： 乘实型
- DIV_R： 除实型

您可对一个 32 位 IEEE 浮点数执行下列运算：

- 计算浮点数的绝对值(ABS)
- 计算浮点数的平方(SQR)或平方根(SQRT)
- 计算浮点数的自然对数(LN)
- 计算底数为 e (= 2.71828...)的浮点数指数值 (EXP)
- 计算以 32 位浮点数表示的角的下列三角函数：
  - 正弦(SIN)和反正弦(ASIN)
  - 余弦(COS)和反余弦(ACOS)
  - 正切(TAN)和反正切(ATAN)

参见判断浮点算术运算指令结果状态字的位

8.2 判断浮点算术运算指令结果状态字的位

描述

浮点算术运算指令影响状态字中的下列位：CC 1 和 CC 0，OV 和 OS。

下表显示了使用浮点数(32 位)指令进行运算后得到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、-0 (零)	0	0	0	*
-3.402823E+38 < 结果 < -1.175494E-38 (负数)	0	1	0	*
+1.175494E-38 < 结果 < 3.402824E+38 (正数)	1	0	0	*

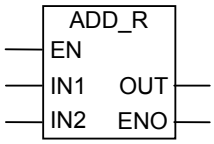
* 指令的结果不影响 OS 位。

结果的范围无效	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的指令(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：加实型

符号



参数	数据类型	内存区域	描述
EN	BOOL	I、Q、M、D、L、T、C	使能输入
IN1	REAL	I、Q、M、D、L 或常数	要相加的第一个数
IN2	REAL	I、Q、M、D、L 或常数	要相加的第二个数
OUT	REAL	I、Q、M、D、L	相加的结果
ENO	BOOL	I、Q、M、D、L	使能输出

描述

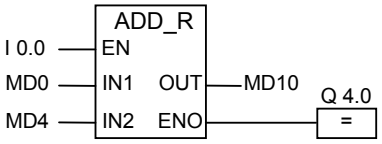
在使能输入(EN)的信号状态为 1 时激活**实数加法**指令。该指令将输入 IN1 与 IN2 相加。结果可在输出 OUT 处扫描。如果其中一个输入或结果不是浮点数，则将 OV 位和 OS 位置 1，并将 ENO 置 0。

参见判断浮点算术运算指令结果状态字的位。

状态字

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

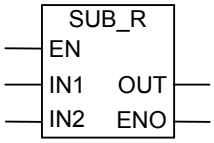
举例



在输入 I0.0 的信号状态为 1 时激活 ADD_R 框。将 MD0 + MD4 的结果存入存储器双字 MD10 中。如果其中一个输入或结果不是浮点数，或者如果 I0.0 的信号状态为 0，则会将输出 Q4.0 置 0，并且不执行该指令。

8.3.2 SUB_R：减实型

符号



参数	数据类型	内存区域	描述
EN	BOOL	I、Q、M、D、L、T、C	使能输入
IN1	REAL	I、Q、M、D、L 或常数	被减数(将从中减去第二个值)
IN2	REAL	I、Q、M、D、L 或常数	减数(从第一个值中减去的数)
OUT	REAL	I、Q、M、D、L	相减的结果
ENO	BOOL	I、Q、M、D、L	使能输出

描述

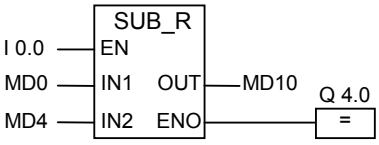
在使能输入(EN)的信号状态为 1 时激活**实数减法**指令。该指令从输入 IN1 减去输入 IN2。结果可在输出 OUT 处扫描。如果其中一个输入或结果不是浮点数，则会将 OV 位和 OS 位置 1，并将 ENO 置 0。

参见判断浮点算术运算指令结果状态字的位。

状态字

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

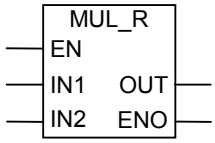
举例



在输入 I0.0 的信号状态为 1 时会激活 SUB_R 框。将 MD0 - MD4 的结果存入存储器双字 MD10 中。如果其中一个输入或结果不是浮点数，或者如果 I0.0 的信号状态为 0，则会将输出 Q4.0 置 0，并且不执行该指令。

8.3.3 MUL_R：乘实型

符号



参数	数据类型	内存区域	描述
EN	BOOL	I、Q、M、D、L、T、C	使能输入
IN1	REAL	I、Q、M、D、L 或常数	被乘数(被乘的值)
IN2	REAL	I、Q、M、D、L 或常数	乘数(用来乘第一个值的值)
OUT	REAL	I、Q、M、D、L	乘运算结果
ENO	BOOL	I、Q、M、D、L	使能输出

描述

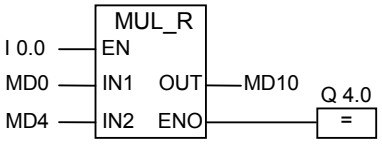
在使能输入(EN)的信号状态为 1 时激活**实数乘法**指令。该指令用输入 IN2 乘输入 IN1。结果可在输出 OUT 处扫描。如果其中一个输入或结果不是浮点数，则会将 OV 位和 OS 位置 1，并将 ENO 置 0。

参见判断浮点算术运算指令结果状态字的位。

状态字

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

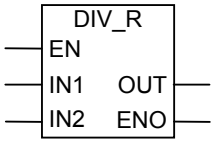
举例



在输入 I0.0 的信号状态为 1 时激活 MUL_R 框。将 MD0 x MD4 的结果存入存储器双字 MD10 中。如果其中一个输入或结果不是浮点数，或者如果 I0.0 的信号状态为 0，则会将输出 Q4.0 置 0，并且不执行该指令。

8.3.4 DIV_R：除实型

符号



参数	数据类型	内存区域	描述
EN	BOOL	I、Q、M、D、L、T、C	使能输入
IN1	REAL	I、Q、M、D、L 或常数	被除数(被第二个值除的值)
IN2	REAL	I、Q、M、D、L 或常数	除数(用来除第一个值的值)
OUT	REAL	I、Q、M、D、L	相除的结果
ENO	BOOL	I、Q、M、D、L	使能输出

描述

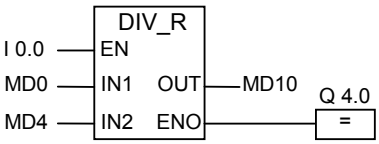
在使能输入(EN)的信号状态为 1 时会激活**实数除法**指令。该指令用输入 IN2 除输入 IN1。结果可在输出 OUT 处扫描。如果其中一个输入或结果不是浮点数，则会将 OV 位和 OS 位置 1，并将 ENO 置 0。

参见判断浮点算术运算指令结果状态字的位。

状态字

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

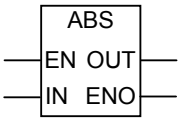
举例



在输入 I0.0 的信号状态为 1 时激活 DIV_R 框。MD4 除 MD0 的结果将存入存储器双字 MD10 中。如果其中一个输入或结果不是浮点数，或者如果 I0.0 的信号状态为 0，则会将输出 Q4.0 置 0，并且不执行该指令。

8.3.5 ABS：浮点数的绝对值运算

符号



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

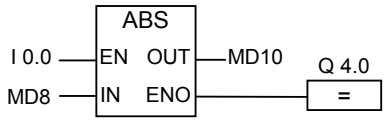
描述

使用**计算浮点数的绝对值**指令，可生成浮点数的绝对值。

状态字

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

举例



如果 I0.0 = 1，则在 MD12 输出 MD8 的绝对值。

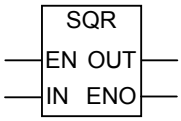
MD8 = +6.234 将导致 MD12 = 6.234 x 1。

如果未执行转换，则输出 Q4.0 为 0 (ENO = EN = 0)。

8.4 扩充指令

8.4.1 SQR：计算浮点数的平方

符号



参数	数据类型	内存区域	描述
EN	BOOL	I、Q、M、D、L、T、C	使能输入
IN	REAL	I、Q、M、D、L 或常数	编号
OUT	REAL	I、Q、M、D、L	该数字的平方
ENO	BOOL	I、Q、M、D、L	使能输出

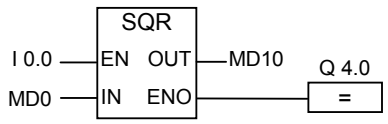
描述

使用**计算浮点数的平方**指令，可生成浮点数的平方。如果其中一个输入或结果不是浮点数，则将 OV 位和 OS 位置 1，并将 ENO 置 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

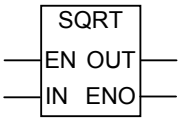
举例



在输入 I0.0 的信号状态为 1 时会激活 SQR 框。SQR(MD0)的结果将存入存储器双字 MD10 中。如果 MD0 小于 0 或有一个输入或结果不是浮点数，或者如果 I0.0 的信号状态为 0，则会将输出 Q4.0 置 0。

8.4.2 SQRT：计算浮点数的平方根

符号



参数	数据类型	内存区域	描述
EN	BOOL	I、Q、M、D、L、T、C	使能输入
IN	REAL	I、Q、M、D、L 或常数	编号
OUT	REAL	I、Q、M、D、L	数字的平方根
ENO	BOOL	I、Q、M、D、L	使能输出

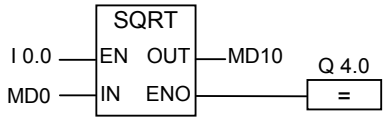
描述

使用**计算浮点数的平方根**指令，可计算浮点数的平方根。如果在地址处的值大于“0”，该指令将返回正值结果。如果其中一个输入或结果不是浮点数，则将 OV 位和 OS 位置 1，并将 ENO 置 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

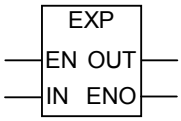
举例



在输入 I0.0 的信号状态为 1 时激活 SQRT 框。SQRT (MD0)的结果将存入存储器双字 MD10 中。如果 MD0 小于 0 或有一个输入或结果不是浮点数，或者如果 I0.0 的信号状态为 0，则会将输出 Q4.0 置 0。

8.4.3 EXP：计算浮点数的指数值

符号



参数	数据类型	内存区域	描述
EN	BOOL	I、Q、M、D、L、T、C	使能输入
IN	REAL	I、Q、M、D、L 或常数	编号
OUT	REAL	I、Q、M、D、L	数字的指数
ENO	BOOL	I、Q、M、D、L	使能输出

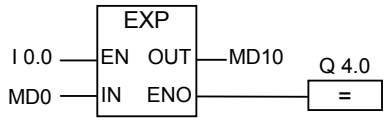
描述

使用**计算浮点数的指数值**指令，可生成底数为 e (= 2.71828...)的浮点数的指数值。如果其中一个输入或结果不是浮点数，则将 OV 位和 OS 位置 1，并将 ENO 置 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

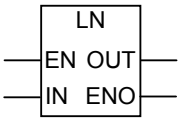
举例



在输入 I0.0 的信号状态为 1 时会激活 EXP 框。EXP (MD0)的结果将存入存储器双字 MD10 中。如果其中一个输入或结果不是浮点数，或者如果 I0.0 的信号状态为 0，则会将输出 Q4.0 置 0。

8.4.4 LN：浮点数自然对数运算

符号



参数	数据类型	内存区域	描述
EN	BOOL	I、Q、M、D、L、T、C	使能输入
IN	REAL	I、Q、M、D、L 或常数	编号
OUT	REAL	I、Q、M、D、L	数字的自然对数
ENO	BOOL	I、Q、M、D、L	使能输出

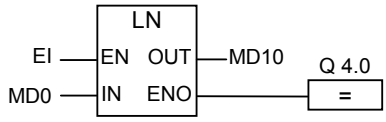
描述

使用**计算浮点数的自然对数**指令，可生成浮点数的自然对数。如果其中一个输入或结果不是浮点数，则将 OV 位和 OS 位置 1，并将 ENO 置 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

举例



在输入 I0.0 的信号状态为 1 时会激活 LN 框。LN(MD0) 的结果将存入存储器双字 MD10 中。如果 MD0 小于 0 或有一个输入或结果不是浮点数，或者如果 I0.0 的信号状态为 0，则会将输出 Q4.0 置 0。

8.4.5 计算以浮点数表示的角的三角函数

描述

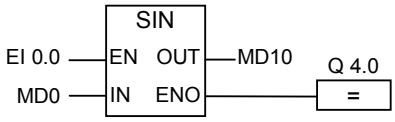
使用下列指令，可计算以 32 位 IEEE 浮点数表示的角的三角函数值。

指令	含义
SIN	计算浮点数(以弧度表示的角度)的正弦值。
COS	计算浮点数(以弧度表示的角度)的余弦。
TAN	计算浮点数(以弧度表示的角度)的正切。
ASIN	计算浮点数的反正弦。结果是用弧度表示的角度。结果的值处于下列范围： -p / 2 <= arc sine <= + p / 2，式中 p = 3.14...
ACOS	计算浮点数的反余弦。结果是用弧度表示的角度。结果的值处于下列范围： 0 v arc cosine <= + p，式中 p = 3.14...
ATAN	计算浮点数的反正切。结果是用弧度表示的角度。结果的值处于下列范围： -p / 2 <= arc tangent <= + p / 2，式中 p = 3.14...

状态字

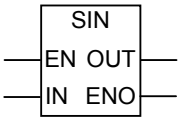
	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	FC
写	X	X	X	X	X	0	X	X	1

举例



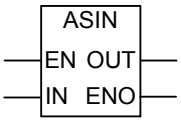
在输入 I0.0 的信号状态为 1 时会激活 SIN 框。SIN (MD0)的结果将存入存储器双字 MD10 中。如果其中一个输入或结果不是浮点数，或者如果 I0.0 的信号状态为 0，则会将输出 Q4.0 置 0。

符号



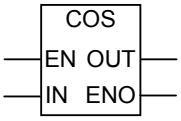
参数	数据类型	内存区域	描述
EN	BOOL	I、Q、M、D、L、T、C	使能输入
IN	REAL	I、Q、M、D、L 或常数	编号
OUT	REAL	I、Q、M、D、L	该数的正弦
ENO	BOOL	I、Q、M、D、L	使能输出

符号



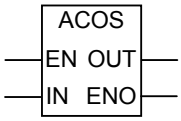
参数	数据类型	内存区域	描述
EN	BOOL	I、Q、M、D、L、T、C	使能输入
IN	REAL	I、Q、M、D、L 或常数	编号
OUT	REAL	I、Q、M、D、L	该数的反正弦
ENO	BOOL	I、Q、M、D、L	使能输出

符号



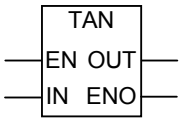
参数	数据类型	内存区域	描述
EN	BOOL	I、Q、M、D、L、T、C	使能输入
IN	REAL	I、Q、M、D、L 或常数	编号
OUT	REAL	I、Q、M、D、L	该数的余弦
ENO	BOOL	I、Q、M、D、L	使能输出

符号



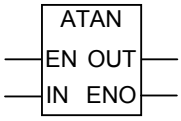
参数	数据类型	内存区域	描述
EN	BOOL	I、Q、M、D、L、T、C	使能输入
IN	REAL	I、Q、M、D、L 或常数	编号
OUT	REAL	I、Q、M、D、L	该数的反余弦
ENO	BOOL	I、Q、M、D、L	使能输出

符号



参数	数据类型	内存区域	描述
EN	BOOL	I、Q、M、D、L、T、C	使能输入
IN	REAL	I、Q、M、D、L 或常数	编号
OUT	REAL	I、Q、M、D、L	该数的正切
ENO	BOOL	I、Q、M、D、L	使能输出

符号

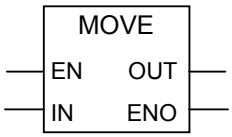


参数	数据类型	内存区域	描述
EN	BOOL	I、Q、M、D、L、T、C	使能输入
IN	REAL	I、Q、M、D、L 或常数	编号
OUT	REAL	I、Q、M、D、L	该数的反正切
ENO	BOOL	I、Q、M、D、L	使能输出

9 传送指令

9.1 MOVE：赋值

符号



参数	数据类型	内存区域	描述
EN	BOOL	I、Q、M、D、L、T、C	使能输入
IN	长度为 8、16 或 32 位的所有基本数据类型	I、Q、M、D、L 或常数	来源值
OUT	长度为 8、16 或 32 位的所有基本数据类型	I、Q、M、D、L	目标地址
ENO	BOOL	I、Q、M、D、L	使能输出

描述

使用**赋值**指令，可将特定的值赋给变量。

在 IN 输入端指定的值将复制到在 OUT 输出端指定的地址。ENO 具有与 EN 相同的信号状态。

使用 MOVE 框，赋值指令可复制长度为 8、16 或 32 位的所有基本数据类型。用户自定义的数据类型(例如数组或结构)则必须使用系统功能 SFC 20“BLKMOV”复制。

赋值指令受主控继电器(MCR)的影响。关于 MCR 如何起作用的更详细信息，请参考主控继电器开/关。

状态字

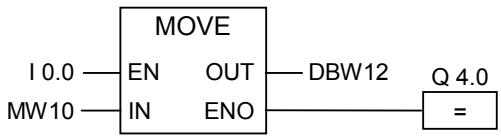
	BR	**CC 1**	**CC 0**	**OV**	**OS**	**OR**	**STA**	**RLO**	**FC**
写	1	-	-	-	-	0	1	1	1

注意

将某个值传送给另一长度的数据类型时，将根据需要截断或以零填充高位字节：

举例：双字	**1111 1111**	**0000 1111**	**1111 0000**	**0101 0101**
传送	**结果**			
到双字：	1111 1111	0000 1111	1111 0000	0101 0101
到字节：				0101 0101
到字：			1111 0000	0101 0101
举例：字节				**1111 0000**
传送	**结果**			
到字节：				1111 0000
到字：			0000 0000	1111 0000
到双字：	0000 0000	0000 0000	0000 0000	1111 0000

举例



在输入 I0.0 为 1 时执行该指令。结果将 MW10 的内容复制到打开的 DB 的数据字 12 中。

执行该指令后，会将 Q4.0 置 1。

10 程序控制指令

10.1 程序控制指令概述

描述

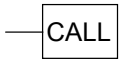
下列指令可用于执行程序控制操作：

- CALL： 调用无参数的 FC/SFC
- CALL_FB： 以框方式调用 FB
- CALL_FC： 以框方式调用 FC
- CALL_SFB： 以框方式调用系统 FB
- CALL_SFC： 以框方式调用系统 FC
- 调用多重背景
- 从库中调用块
- 主控继电器指令
- 使用 MCR 函数的重要注意事项
- MCR< 主控继电器打开
- MCR> 主控继电器关闭
- MCRA 主控继电器激活
- MCRD 主控继电器去活
- RET 返回

10.2 CALL：调用无参数的 FC/SFC

符号

<FC-/SFC 编号>



参数	数据类型	内存区域	描述
编号	BLOCK_FC	-	FC 或 SFC 的编号(例如，FC10 或 SFC59)。可用的 SFC 取决于 CPU。 只能在 FB 而不能在 FC 中进行数据类型为 BLOCK_FC 的参数作为地址的条件调用。

描述

通过**调用不带参数的 FC/SFC** 指令，可以调用没有参数的功能(FC)或系统功能(SFC)。这种调用是条件调用还是无条件调用取决于调用前面的逻辑运算(参见举例)。

在功能(FC)的代码段，不能为条件调用将类型为 BLOCK_FC 的任何参数指定为地址。然而，可以在功能块(FB)中将 BLOCK_FC 类型的参数指定为地址。

只有当 RLO 为 1 时，才执行条件调用。当没有执行条件调用时，调用指令后的 RLO 为 0。当执行了该指令后，执行下列功能：

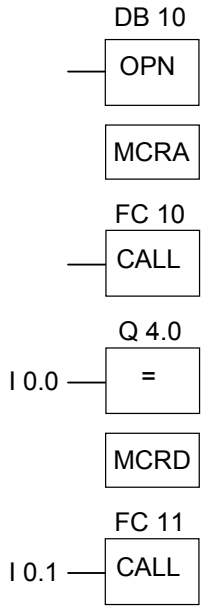
- 保存要求返回调用块的地址。
- 保存数据块寄存器(数据块和背景数据块)。
- 将 MA 位(有效 MCR 位)写入到块堆栈(BSTACK)中。
- 为被调用的 FC 或 SFC 创建新的本地数据范围。

然后在被调用块中继续执行程序。

状态字

		BR	CC 1	CC 0	OV	OS	OR	STA	RLO	FC
条件调用	写	-	-	-	-	0	0	1	1	0
无条件调用	写	-	-	-	-	0	0	1	-	0

举例



当对 FC10 执行无条件调用时，CALL 指令执行下列功能：

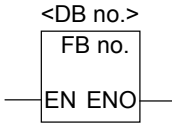
- 保存要求返回当前 FB 的地址。
- 为 DB10 和 FB 的背景数据块保存选择器。
- 将在 MCRA 指令中设置成 1 的 MA 位压入块堆栈(BSTACK)，然后为被调用的 FC10 将该位复位成 0

继续在 FC10 中执行程序。如果要在 FC10 中使用 MCR 函数，那么必须在此重新激活该功能。当完成 FC10 时，程序执行返回到调用 FB。恢复 MA 位。DB10 和用户自定义的 FB 的背景数据块重新成为当前 DB，而与 FC10 使用了哪个 DB 无关。

从 FC10 返回跳转后，将输入 I0.0 的信号状态分配给输出 Q4.0。FC11 的调用为条件调用。只有在输入 I0.1 的信号状态为 1 时才执行调用。如果执行该调用，那么该功能与调用 FC10 相同。

10.3 CALL_FB：以框方式调用 FB

符号



该符号取决于功能块(是否存在参数以及存在多少个参数)。必须存在 EN、ENO 和 FB 的名称或编号。

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

描述

当 EN 的信号状态为 1 时，执行 **CALL_FB** (以框方式调用 FB)。当执行 CALL_FB 指令调用时：

- 保存调用块的返回地址，
- 保存当前打开的两个数据块(DB 和背景 DB)的选择数据，
- 为被调用的功能块创建一个新的本地数据范围。
- 将 MA 位(有效 MCR 位)压入到 BSTACK 中。

状态字

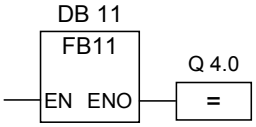
		BR	CC 1	CC 0	OV	OS	OR	STA	RLO	FC
条件调用	写	X	-	-	-	0	0	X	X	X
无条件调用	写	-	-	-	-	0	0	X	X	X

举例

程序段 1

程序段 2

程序段 3



程序段 4

上面所示的程序段为由用户创建的功能块中的程序段。在该块中打开 DB10，并激活 MCR。当执行无条件 FB11 调用时，发生下面情况：

保存调用块的返回地址和 DB10 以及调用功能块的背景数据块的选择数据。由 MCRA 指令设置成 1 的 MA 位被压入到 BSTACK 中，然后为被调用的功能块 FB11 将其设置成 0。继续在 FB11 中执行程序。当 FB11 需要 MCR 时，必须在功能块中重新激活 MCR。必须由[SAVE]指令在 BR 位中保存 RLO 的信号状态，以便评估调用 FB 中的错误。当已经执行了 FB11 时，程序返回调用功能块。恢复 MA 位，且由用户编写的功能块的背景数据块重新成为打开的数据块。当正确执行 FB11 时，ENO 的信号状态为 1，因此，Q4.0 的状态也为 1。

注意

通过 FB/SFB 调用，丢失上一个打开数据块的编号。必须重新打开所要求的 DB。

10.4 CALL_FC (以框方式调用 FC)

符号

该符号取决于功能(是否存在参数以及存在多少个参数)。必须存在 EN、ENO 和 FC 的名称或编号。

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

描述

CALL_FC (以框方式调用 FC)调用一个功能(FC)。当 EN 的信号状态为 1 时，执行该调用。当执行 CALL_FC 指令时：

- 保存调用块的返回地址，
- 为被调用的功能创建一个新的本地数据范围。
- 将 MA 位(有效 MCR 位)压入到 BSTACK 中。

最后，在被调用功能中继续执行程序。

检查 BR 位，以确定 ENO。用户必须在被调用块[SAVE]中将所要求的信号状态(错误评估)赋值给 BR 位。

当调用一个功能，而被调用块的变量声明表中含有 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赋值给形式参数。功能声明中的任何初始值都没有意义。

状态字

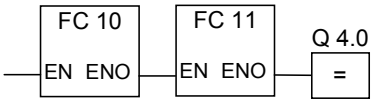
		BR	CC 1	CC 0	OV	OS	OR	STA	RLO	FC
条件调用	写	X	-	-	-	0	0	X	X	X
无条件调用	写	-	-	-	-	0	0	X	X	X

举例

程序段 1

程序段 2

程序段 3



上面所示的程序段为由用户创建的功能块中的程序段。在该块中打开 DB10，并激活 MCR。当执行无条件 FC1 调用时，发生下面情况：

保存调用功能块的返回地址和 DB10 以及调用功能块的背景数据块的选择数据。由 MCRA 指令设置成 1 的 MA 位被压入到 BSTACK 中，然后为被调用块 FC10 将其设置成 0。继续在 FC10 中执行程序。当 FC10 要求 MCR 时，必须在 FC10 中重新激活 MCR。必须由[SAVE]指令在 BR 位中保存 RLO 的信号状态，以便评估调用 FB 中的错误。当已经执行了 FC10 时，程序返回调用功能块。恢复 MA 位。已经执行 FC10 后，根据 ENO 的信号状态，在调用 FB 中继续执行程序，执行情况如下：

ENO = 1　执行 FC11

ENO = 0　在下一个程序段中启动程序

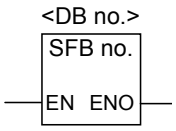
正确执行了 FC11 时，将 ENO 设置成 1，因此也将 Q4.0 设置成。

注意

程序返回调用块后，不能始终确保重新打开上一次打开的 DB。请参见自述文件中的注意事项。

10.5 CALL_SFB：以框方式调用系统 FB

符号



该符号取决于系统功能块(是否存在参数以及存在多少个参数)。必须存在 EN、ENO 和 SFB 的名称或编号。

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

描述

当 EN 的信号状态为 1 时，执行 **CALL_SFB** (以框方式调用 SFB)。当执行 CALL_SFB 指令调用时：

- 保存调用块的返回地址，
- 保存当前打开的两个数据块(DB 和背景 DB)的选择数据，
- 为被调用的系统功能块创建一个新的本地数据范围。
- 将 MA 位(有效 MCR 位)压入到 BSTACK 中。

最后，在被调用的系统功能块中继续执行程序。当调用该功能且没有发生错误时，ENO 为 1。

状态字

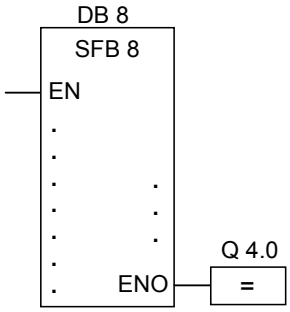
		BR	**CC 1**	**CC 0**	**OV**	**OS**	**OR**	**STA**	**RLO**	**FC**
条件调用	写	X	-	-	-	0	0	X	X	X
无条件调用	写	-	-	-	-	0	0	X	X	X

举例

程序段 1

程序段 2

程序段 3



程序段 4

上面所示的程序段为由用户创建的功能块中的程序段。在该块中打开 DB10，并激活 MCR。当执行无条件 SFB8 调用时，发生下面情况：

保存调用块的返回地址和 DB10 以及调用功能块的背景数据块的选择数据。由 MCRA 指令设置成 1 的 MA 位被压入到 BSTACK 中，然后为被调用的系统功能块 SFB8 将其设置成 0。继续在 SFB8 中执行程序。当已经执行了 SFB8 时，程序返回调用功能块。恢复 MA 位，且由用户编写的功能块的背景数据块重新成为当前数据块。当正确执行 SFB8 时，ENO 的信号状态为 1，因此，Q4.0 的状态也为 1。

注意

通过 FB/SFB 调用，丢失了上一个打开数据块的编号。必须重新打开所要求的 DB。

10.6 CALL_SFC (以框方式调用系统 FC)

符号

该符号取决于是否存在参数以及存在多少个参数。必须存在 EN、ENO 和 SFC 的名称或编号。

参数	数据类型	内存区域	描述
EN	BOOL	I、Q、M、L、D	使能输入
ENO	BOOL	I、Q、M、L、D	使能输出
SFC 编号	BLOCK_SFC	-	SFC 编号，范围取决于 CPU

描述

CALL_SFC (以框方式调用系统 FC)调用一个系统功能。当 EN 的信号状态为 1 时，执行该调用。当执行 CALL_SFC 指令时：

- 保存调用块的返回地址，
- 为被调用的系统功能创建一个新的本地数据范围。
- 将 MA 位(有效 MCR 位)压入到 BSTACK 中。

最后，在被调用的系统功能中继续执行程序。当调用该功能且没有发生错误时，ENO 为 1。

状态字

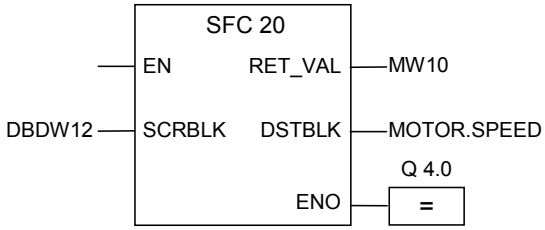
		BR	CC 1	CC 0	OV	OS	OR	STA	RLO	FC
条件调用	写	X	-	-	-	0	0	X	X	X
无条件调用	写	-	-	-	-	0	0	X	X	X

举例

程序段 1

程序段 2

程序段 3



上面所示的程序段为由用户创建的功能块中的程序段。在该块中打开 DB10，并激活 MCR。当执行无条件 SFC20 调用时，发生下面情况：

保存调用块的返回地址和 DB10 以及调用功能块的背景数据块的选择数据。由 MCRA 指令设置成 1 的 MA 位被压入到 BSTACK 中，然后为被调用块 SFC20 将其设置成 0。继续在 SFC20 中执行程序。当已经执行了 SFC20 时，程序返回调用功能块。恢复 MA 位。

已经执行 SFC20 后，根据 ENO 的信号状态，在调用 FB 中继续执行程序，执行情况如下：

ENO = 1　Q4.0 = 1

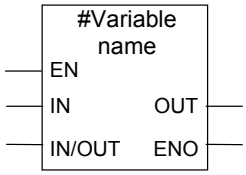
ENO = 0　Q4.0 = 0

注意

程序返回调用块后，不能始终确保重新打开上一次打开的 DB。请参见自述文件中的注意事项。

10.7 调用多重调用多重背景

符号



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

描述

通过声明一个功能块类型的静态变量可以创建一个多重背景。在程序元素分类中只列出了已经声明的多重背景。多重背景的符号改变取决于是否存在参数以及存在多少个参数。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 从库中调用块

在此，可使用 SIMATIC 管理器中可用的库来选择一个块，该块：

- 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)”)
- 保存在库中以便再次使用的程序块

10.9 主控继电器指令

使用 MCR 函数的重要注意事项

主控继电器的定义

主控继电器(MCR，参见主控继电器开/关)用于激活和去活信号流。.去活的信号流相当于一个写入零数值而不是已计算数值的指令序列，或相当于一个保持现有存储值不变的指令序列。由下面所示的指令触发的操作取决于 MCR。

赋值和**中间输出**指令在 MCR 为 0 时，将 0 写入到存储器中。**置位输出**和**复位输出**指令保持现有数值不变。

受 MCR 区域影响的指令如下：

- #： 中间输出
- =： 赋值
- R： 复位输出
- S： 设置输出
- SR : 置位复位触发器
- RS： 复位置位触发器
- MOVE： 赋值

程序按 **MCR** 的指令执行及其如何对 **MCR** 信号状态作出反应

MCR 的信号状态	赋值、 中间输出	置位或 复位输出	移动
0 (“关闭”**)**	写入 0。 (掉电时，模仿继电器的静止状态。)	不写入。 (掉电时，模仿仍然保持其当前状态的继电器。)	写入 0。 (掉电时，模仿生成数值“0”的元件。)
1 (“打开”**)**	正常执行	正常执行	正常执行

10.10 使用 MCR 函数的重要注意事项

在使用主控继电器以 MCRA 方式激活块时应注意

- 去活 MCR 时，为主控继电器打开和主控继电器关闭之间的程序段中的所有赋值写入数值 0。这对包含一个赋值的**所有**框都有效，包括传送到块的参数在内。
- 当 RLO = 0 时(位于主控继电器打开指令前)，去活 MCR。

危险：PLC 处于 STOP 状态，或未定义的运行时间特性！

编译器还对在 VAR_TEMP 中定义的临时变量后的本地数据进行写访问，以便计算地址。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时间特性：

形式参数存取

- 存取 STRUCT、UDT、ARRAY、STRING 类型的复杂 FC 参数的组件
- 存取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)的功能块访问类型为 BLOCK_DB 的参数，打开 DB0。任何后继数据存取都会将 CPU 设置成 STOP 状态。T 0、C 0、FC0 或 FB0 也经常用于 TIMER、COUNTER、BLOCK_FC 和 LOCK_FB。

参数传递

- 通过调用可传递参数。

LAD/FBD

- 梯形图或 FBD 中的 T 分支和中间输出从 RLO = 0 开始。

纠正方法

从上述命令对 MCR 的依赖性，来解题：

1. 在语句或程序段出现问题**之前**利用主控继电器去激活指令，可以**去活**主控继电器。
2. 在语句或程序段出现问题之后，利用主控继电器激活指令，可以**再次激活**主控继电器。

10.11 MCR</MCR>：主控继电器开/关

使用 MCR 函数的重要注意事项

符号

MCR 打开

主控继电器打开(MCR<)指令将触发一个在 MCR 堆栈中保存 RLO 并打开一个 MCR 区域的操作。打开 MCR 区域时，保存在 MCR 堆栈中的 RLO 将会对 MCR 指令中所示的指令产生影响。

MCR 堆栈的工作形式如同 LIFO (后入先出)缓冲区。只能使用 8 个输入项。当堆栈已满时，主控继电器打开指令产生一个 MCR 堆栈故障(MCRF)。

符号

MCR 关闭

主控继电器关闭(MCR>)指令关闭最后打开的 MCR 区域。该指令通过从 MCR 堆栈中移除 RLO 条目完成该操作。RLO 由主控继电器打开指令保存在栈内。在 LIFO(后入先出)MCR 堆栈的另一端释放的输入项被设置成 1。

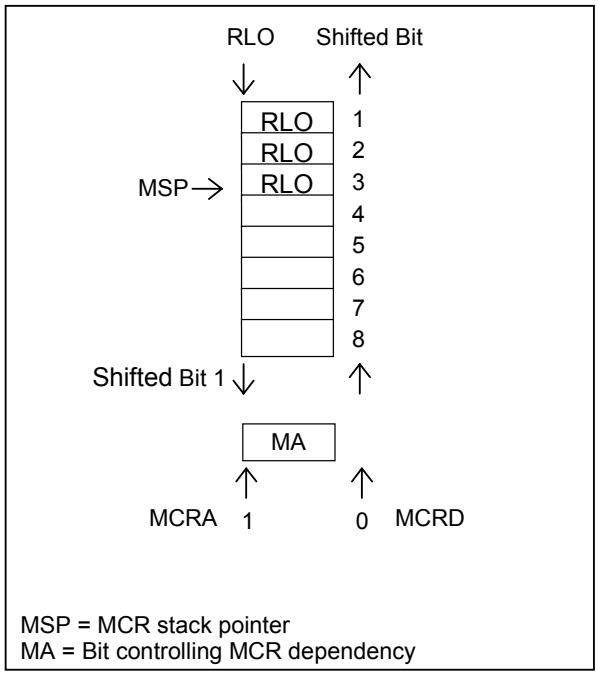
当堆栈已空时，主控继电器关闭指令产生一个 MCR 堆栈故障(MCRF)。

MCR 堆栈

MCR 由 1 位宽和 8 位深的堆栈控制。只要堆栈中的所有 8 个输入项都等于 1，MCR 就被激活。MCR<指令将 RLO 位复制到 MCR 堆栈中。MCR>指令从堆栈中删除最后一个输入项，然后将已释放的堆栈地址设置成 1。

如果发生错误，例如，当连续出现多于 8 个 MCR>指令，或在堆栈为空时尝试执行 MCR>指令时，将触发 MCRF 出错消息。

MCR 堆栈的监视基于堆栈指针(MSP：0 = 空；1 = 1 个输入项；2 = 2 个输入项； ...、8 = 8 个输入项)。



MCR<指令采用 RLO 的信号状态，并将它复制到 MCR 位中。

MCR>指令无条件将 MCR 位设置成 1。由于该特性，MCRA 和 MCRD 指令之间的每个其它指令都独立于 MCR 位操作。

嵌套指令 MCR<和 MCR>

可以嵌套 **MCR<**和 **MCR>**指令。最大嵌套深度为 8 个，也就是说，在插入一个 MCR>指令之前，最多可以连续写入 8 个 MCR<指令。必须编写相同数目的 MCR< 和 MCR>指令。

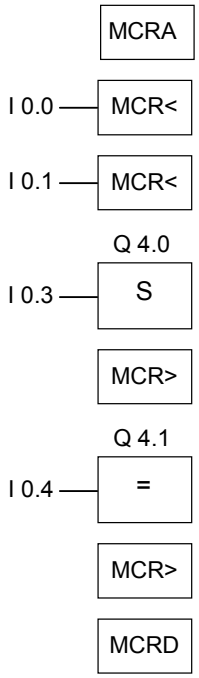
当嵌套了 MCR<指令时，形成较低嵌套级别的 MCR 位。然后，MCR<指令根据与运算真值表组合当前 RLO 与当前的 MCR 位。

当 MCR>指令完成一个嵌套级别时，它从一个较高的级别读取 MCR 位。

状态字

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

举例



当 **MCRA** 指令激活 MCR 函数时，可以最多创建 8 个嵌套的 MCR 区域。在该例中，有两个 MCR 区域。第一个 MCR>指令与第二个 MCR<指令一起使用。第二组 MCR 括号(MCR<MCR>)之间的所有指令都属于第二个 MCR 区域。按如下执行操作

- 当 I0.0 = 1 时：将输入 I0.4 的信号状态赋值给输出 Q4.1。
- 当 I0.0 = 0 时：输出 Q4.1 的信号状态为 0，与输入 I0.4 的信号状态无关。输出 Q4.0 保持不变，与输入 I0.3 的信号状态无关。
- 当 I0.0 = 1 且 I0.1 = 1 时：当 I0.3 = 1 且 Q4.1 =I0.4 时，将输出 Q4.0 设置成 1。
- 当 I0.1 = 0 时：输出 Q4.0 保持不变，与输入 I0.3 和输入 I0.0 的信号状态无关。

10.12 MCRA/MCRD：主控继电器激活/去活

使用 MCR 函数的重要注意事项

符号

激活 MCR

通过**激活主控继电器指令**，可以生成依赖于 MCR 的后继命令。输入该命令后，可以使用这些指令对 MCR 区域进行编程(参见主控继电器开/关)。当程序激活一个 MCR 区域时，所有 MCR 操作都取决于 MCR 堆栈的内容。

符号

去活 **MCR**

通过**去活主控继电器指令**，后继命令不再与 MCR 有关。执行该指令后，不能再对任何 MCR 区域进行编程。当程序去活一个 MCR 区域时，MCR 始终处于通电状态，这与 MCR 堆栈中的条目无关。

MCR 堆栈和控制其依赖性的位(MA 位)与单个级别有关，必须在每次改变顺序级别时保存并读取它们。在每个顺序级别开始处预置它们(将 MCR 输入位 1 - 8 设置成 1，将 MCR 堆栈指针设置成 0，将 MA 位设置成 0)。

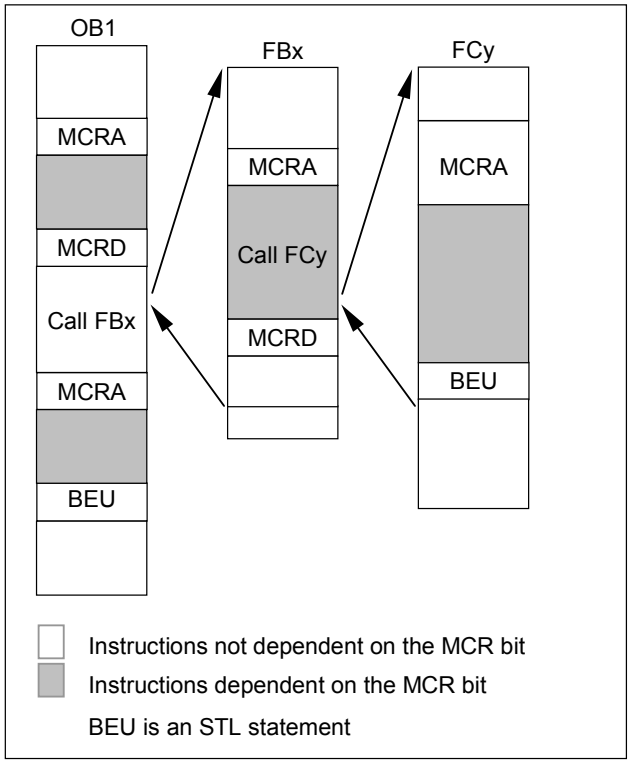
在块与块之间传送 MCR 堆栈，且在每次调用块时，保存 MA 位并将其设置成 0。在块结束处重新读取 MA 位。

MCR 可以按优化生成代码的 CPU 的运行时间的方式执行。这是因为，块不传递 MCR 的依赖性；必须由一个 MCR 指令显式激活。生成代码的 CPU 识别该指令，并生成估算 MCR 堆栈所需的附加代码，直到它识别 MCR 指令或到达块结束处。当指令超出 MCRA/MCRD 范围时，不会增加运行时间。

MCRA 和 **MCRD** 指令必须始终在程序内成对使用。

激活和去活 **MCR** 区域

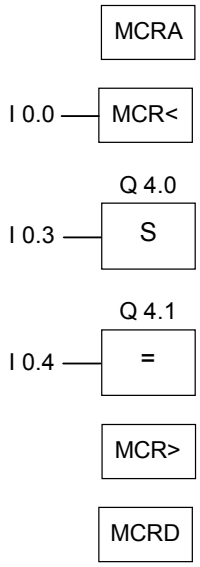
在 MCRA 和 MCRD 之间编程的操作取决于 MCR 位的信号状态。在 MCRA-MCRD 序列外进行编程的操作不依赖于 MCR 位的信号状态。如果丢失 MCRD 指令，那么在 MCRA 和 BEU 指令之间编程的操作取决于 MCR 位。



状态字

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

举例



MCRA 指令激活 MCR 函数，直到下一个 MCRD 指令执行。根据 MA 位(在此为 I0.0)处理 MCR<和 MCR>之间的指令：

- 当输入 I0.0 的信号状态为 1 时：
  - 当输入 I0.3 的信号状态为 1 时，将输出 Q4.0 设置成 1。
  - 当输入 I0.3 的信号状态为 0 时，输出 Q4.0 保持不变。
  - 将输入 I0.4 的信号状态赋值给输出 Q4.1。
- 当输入 I0.0 的信号状态为 0 时：
  - 输出 Q4.0 保持不变，与输入 I0.3 的信号状态无关。
  - 输出 Q4.1 为 0，与输入 I0.4 的信号状态无关。

必须在块中自行编写功能(FC)和功能块(FB)的依赖性。当从一个 MCRA/MCR 序列调用该功能或功能块时，不是该序列内的所有指令都自动与 MCR 位相关。为此，使用被调用块的 MCRA 指令。

10.13 RET：返回

符号

描述

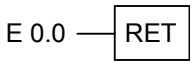
可使用**返回**指令来退出块。可以有条件地退出块。

状态字

	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。这还将影响到 BR 位

举例



当输入 I0.0 的信号状态为 1 时，会退出块。

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

描述

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)。向左移动 n 位相当于将输入端 IN 的内容乘以 2 的 n 次幂(2^n)；向右移动 n 位则相当于将输入端 IN 的内容除以 2 的 n 次幂(2^n)。例如，如果将等价于十进制值 3 的二进制数左移 3 位，将得到等价于十进制值 24 的二进制数。如果将等价于十进制值 16 的二进制数右移 2 位，则会得到等价于十进制值 4 的二进制数。

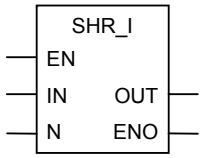
您提供给输入参数 N 的数值决定了移动相应值的位数。移位指令产生的空位将用零或符号位的信号状态(0 表示正，1 表示负)来填补。最后移动的位的信号状态将装入状态字的 CC 1 位中(请参见“CPU 寄存器”)。状态字的 CC 0 和 OV 位会被复位为 0。可以使用跳转指令估算 CC 1 位。

以下是可用的移位指令：

- SHR_I： 右移整数
- SHR_DI： 右移双精度整型
- 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：右移整数

符号



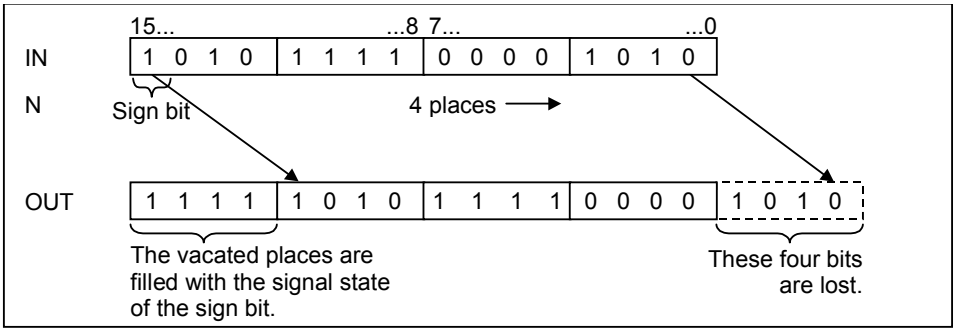
参数	数据类型	内存区域	描述
EN	BOOL	I、Q、M、L、D、T、C	使能输入
IN	INT	I、Q、M、L、D	要移位的值
N	WORD	I、Q、M、L、D	值将移位的位数
OUT	INT	I、Q、M、L、D	移位指令的结果
ENO	BOOL	I、Q、M、L、D	使能输出

描述

在使能输入(EN)端的信号状态为 1 时会激活**整数右移**指令。此指令将输入端 IN 的 0 到 15 位逐位右移。输入 N 指定值将移位的位数。如果 N 大于 16，则该命令将 N 视为 16 进行处理。左侧的空位将用第 15 位的信号状态(整数的符号)填补。正数填补 0，负数则填补 1。移位操作的结果可在 OUT 输出端扫描。

如果 N 不等于 0，则此指令触发的操作总是将状态字的 CC 0 和 OV 位复位为 0。

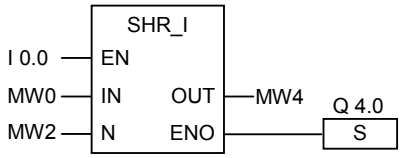
ENO 具有与 EN 相同的信号状态。



状态字

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

举例

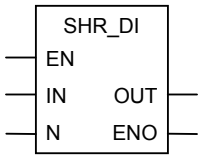


在 I0.0 的信号状态为 1 时激活此指令。存储器字 MW0 将按在存储器字 MW2 中指定的位数右移。

结果将存入存储器字 MW4 中。输出 Q4.0 置 1。

11.1.3 SHR_DI：右移双精度整型

符号



参数	数据类型	内存区域	描述
EN	BOOL	I、Q、M、L、D、T、C	使能输入
IN	DINT	I、Q、M、L、D	要移位的值
N	WORD	I、Q、M、L、D	值将移位的位数
OUT	DINT	I、Q、M、L、D	移位指令的结果
ENO	BOOL	I、Q、M、L、D	使能输出

描述

在使能输入(EN)端的信号状态为 1 时会激活**整数右移**指令。此指令将输入端 IN 的全部内容逐位右移。输入 N 指定值将移位的位数。如果 N 大于 32，则该命令将 N 视为 32 进行处理。左侧的位将用第 31 位的信号状态(双精度整数的符号)填补。正数填补 0，负数则填补 1。移位操作的结果可在 OUT 输出端扫描。

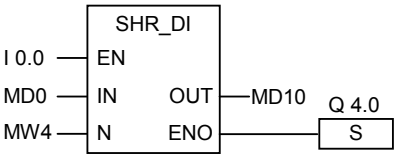
如果 N 不等于 0，则此指令触发的操作总是将状态字的 CC 0 和 OV 位复位为 0。

ENO 具有与 EN 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

举例

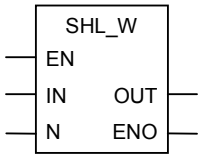


在 I0.0 的信号状态为 1 时会激活此指令。存储器双字 MD0 将按在存储器字 MW4 中指定的位数右移。

结果将存入存储器双字 MD10 中。输出 Q4.0 置 1。

11.1.4 SHL_W：左移字

符号



参数	数据类型	内存区域	描述
EN	BOOL	I、Q、M、L、D、T、C	使能输入
IN	WORD	I、Q、M、L、D	要移位的值
N	WORD	I、Q、M、L、D	值将移位的位数
OUT	WORD	I、Q、M、L、D	移位指令的结果
ENO	BOOL	I、Q、M、L、D	使能输出

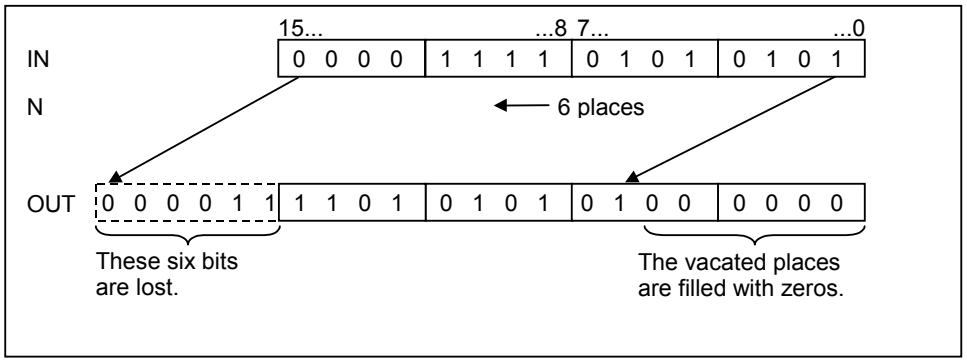
描述

在使能输入(EN)的信号状态为 1 时会激活**字左移**指令。此指令将输入端 IN 的 0 到 15 位逐位左移。

输入 N 指定要使值移位的位数。如果 N 大于 16，则此命令在 OUT 输出端写入 0 并将状态字的 CC 0 和 OV 位置 0。将右侧的各位补零。移位操作的结果可在 OUT 输出端扫描。

如果 N 的值不等于 0，则此指令触发的操作总是将状态字的 CC 0 和 OV 位复位为 0。

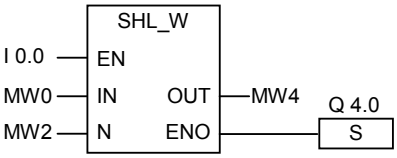
ENO 具有与 EN 相同的信号状态。



状态字

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

举例



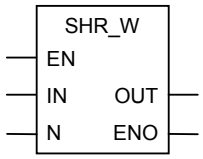
在 I0.0 的信号状态为 1 时会激活该指令。

存储器字 MW0 将按在存储器字 MW2 中指定的位数左移。

结果将存入存储器字 MW4 中。输出 Q4.0 置 1。

11.1.5 SHR_W：右移字

符号



参数	数据类型	内存区域	描述
EN	BOOL	I、Q、M、L、D、T、C	使能输入
IN	WORD	I、Q、M、L、D	要移位的值
N	WORD	I、Q、M、L、D	值将移位的位数
OUT	WORD	I、Q、M、L、D	移位指令的结果
ENO	BOOL	I、Q、M、L、D	使能输出

描述

在使能输入(EN)端的信号状态为 1 时会激活**字右移**指令。此指令将输入端 IN 的 0 到 15 位逐位右移。16 到 31 位不受影响。输入 N 指定值将移位的位数。如果 N 大于 16，则此命令在 OUT 输出端写入 0 并将状态字的 CC 0 和 OV 位置 0。将左侧移空的位补零。移位操作的结果可在 OUT 输出端扫描。

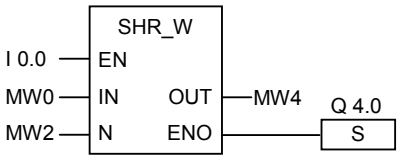
如果 N 不等于 0，则此指令触发的操作总是将状态字的 CC 0 和 OV 位复位为 0。

ENO 具有与 EN 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

举例

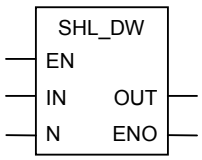


在 I0.0 的信号状态为 1 时激活此指令。存储器字 MW0 将按在存储器字 MW2 中指定的位数右移。

结果将存入存储器字 MW4 中。输出 Q4.0 置 1。

11.1.6 SHL_DW：双字左移

符号



参数	数据类型	内存区域	描述
EN	BOOL	I、Q、M、L、D、T、C	使能输入
IN	DWORD	I、Q、M、L、D	要移位的值
N	WORD	I、Q、M、L、D	值将移位的位数
OUT	DWORD	I、Q、M、L、D	移位指令的结果
ENO	BOOL	I、Q、M、L、D	使能输出

描述

在使能输入(EN)端的信号状态为 1 时会激活**双字左移**指令。此指令将输入端 IN 的 0 到 31 位逐位左移。输入 N 指定值将移位的位数。如果 N 大于 32，则此命令在 OUT 输出端写入 0 并将状态字的 CC 0 和 OV 位置 0。将右侧移空的各位补零。移位操作的结果可在 OUT 输出端扫描。

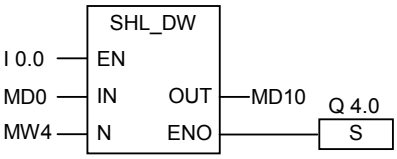
如果 N 的值不等于 0，则此指令触发的操作总是将状态字的 CC 0 和 OV 位复位为 0。

ENO 具有与 EN 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

举例



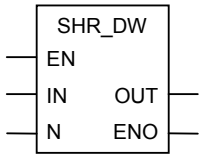
在 I0.0 的信号状态为 1 时会激活该指令。

存储器双字 MD0 将按在存储器字 MW4 中指定的位数左移。

结果将存入存储器双字 MD10 中。输出 Q4.0 置 1。

11.1.7 SHR_DW：右移双字

符号



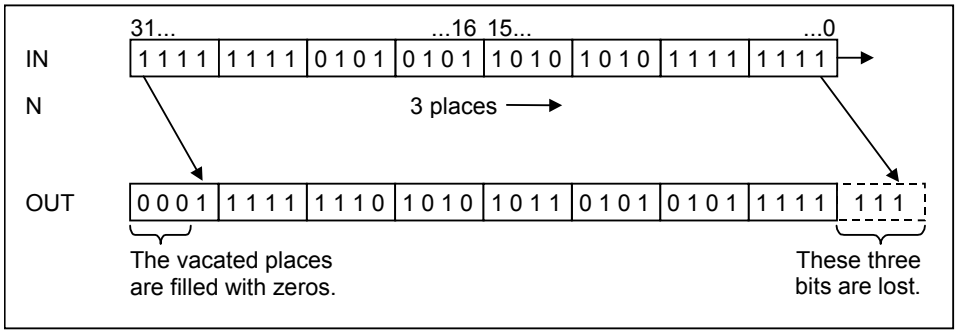
参数	数据类型	内存区域	描述
EN	BOOL	I、Q、M、L、D、T、C	使能输入
IN	DWORD	I、Q、M、L、D	要移位的值
N	WORD	I、Q、M、L、D	值将移位的位数
OUT	DWORD	I、Q、M、L、D	移位指令的结果
ENO	BOOL	I、Q、M、L、D	使能输出

描述

在使能输入(EN)端的信号状态为 1 时会激活**双字右移**指令。此指令将输入端 IN 的 0 到 31 位逐位右移。输入 N 指定值将移位的位数。如果 N 大于 32，则此命令在 OUT 输出端写入 0 并将状态字的 CC 0 和 OV 位置 0。将左侧移空的位补零。移位操作的结果可在 OUT 输出端扫描。

如果 N 不等于，则此指令触发的操作总是将状态字的 CC 1 和 OV 位复位为 0。

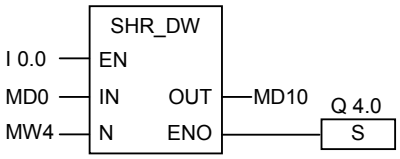
ENO 具有与 EN 相同的信号状态。



状态字

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

举例



在 I0.0 的信号状态为 1 时会激活此指令。存储器双字 MD0 将按在存储器字 MW4 中指定的位数右移。

结果将存入 MD10 中。输出 Q4.0 置 1。

11.2 循环移位指令

11.2.1 循环移位指令概述

描述

可使用循环移位指令将输入端 IN 的全部内容逐位向左或向右循环移动(请参见 CPU 寄存器)。移空的位将用从输入端 IN 移出的位的信号状态填补。

为输入参数 N 指定的值即是要循环移位的位数。

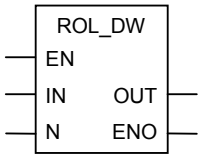
根据指令不同，循环移位将使用状态字的 CC 1 位。状态字的 CC 0 位被复位为 0。

以下是可用的循环移位指令：

- ROL_DW： 循环左移双字
- ROR_DW： 循环右移双字

11.2.2 ROL_DW：循环左移双字

符号



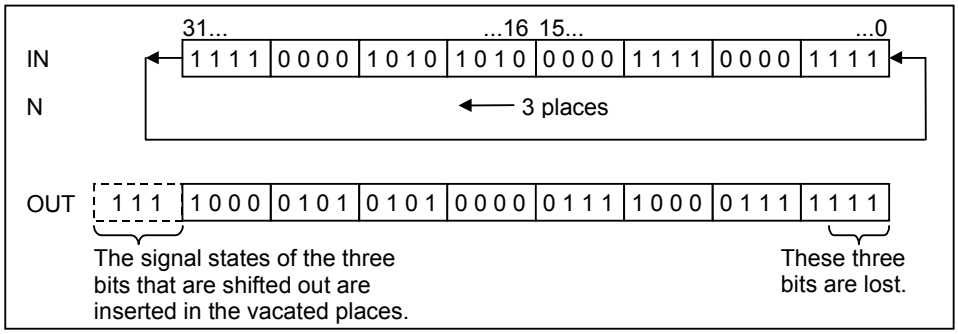
参数	数据类型	内存区域	描述
EN	BOOL	I、Q、M、L、D、T、C	使能输入
IN	DWORD	I、Q、M、L、D	要循环移位的值
N	WORD	I、Q、M、L、D	值将循环移动的位数
OUT	DWORD	I、Q、M、L、D	循环移位指令的结果
ENO	BOOL	I、Q、M、L、D	使能输出

描述

在使能输入(EN)端的信号状态为 1 时激活**双字循环左移**指令。此指令将输入端 IN 的全部内容逐位循环左移。输入 N 指定要循环移动的位数。如果 N 大于 32，则双字循环移动((N-1)以 32 为模) +1)位。右侧的位用循环位的信号状态填补。循环运算的结果可在 OUT 输出端扫描。

如果 N 不等于 0，则此指令触发的操作总是将状态字的 CC 0 和 OV 位复位为 0。

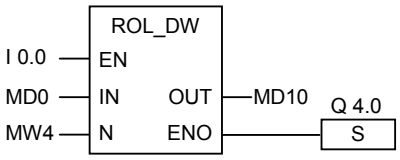
ENO 具有与 EN 相同的信号状态。



状态字

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

举例

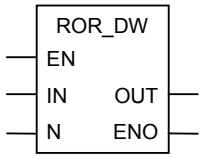


在 I0.0 的信号状态为 1 时激活此指令。存储器双字 MD0 将按在存储器字 MW4 中指定的位数循环右移。

结果将存入存储器双字 MD10 中。输出 Q4.0 置 1。

11.2.3 ROR_DW：循环右移双字

符号



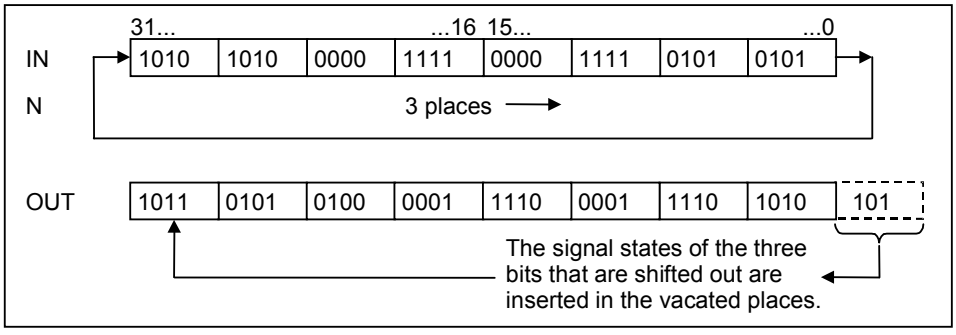
参数	数据类型	内存区域	描述
EN	BOOL	I、Q、M、L、D、T、C	使能输入
IN	DWORD	I、Q、M、L、D	要循环移位的值
N	WORD	I、Q、M、L、D	值将循环移动的位数
OUT	DWORD	I、Q、M、L、D	循环移位指令的结果
ENO	BOOL	I、Q、M、L、D	使能输出

描述

在使能输入(EN)端的信号状态为 1 时激活**双字循环右移**指令。此指令将输入端 IN 的全部内容逐位循环右移。输入 N 指定值将循环移动的位数。如果 N 大于 32，则双字循环移动((N-1)以 32 为模) +1)位。N 值可介于 0 和 31 之间。左侧的位将用循环位的信号状态填补。循环运算的结果可在 OUT 输出端扫描。

如果 N 不等于 0，则此指令触发的操作总是将状态字的 CC 0 和 OV 位复位为 0。

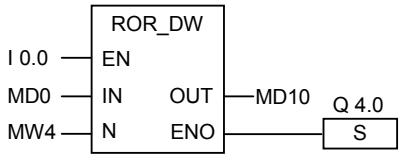
ENO 具有与 EN 相同的信号状态。



状态字

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

举例



在 I0.0 的信号状态为 1 时会激活此指令。存储器双字 MD0 将按在存储器字 MW4 中指定的位数循环右移。

结果将存入存储器双字 MD10 中。输出 Q4.0 置 1。

12 状态位指令

12.1 状态位指令概述

描述

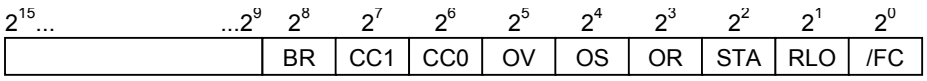
状态位指令是对状态字的位进行操作的逻辑指令(请参见 CPU 寄存器)。其中的各条指令分别对下列条件之一做出反应，每条指令以状态字的一个或多个位来表示：

- 二进制结果位(BR)被置位(信号状态为 1)。
- 运算的结果通过下列方式之一与 0 相关： == 0、<> 0、> 0、< 0、>= 0、<= 0。
- 运算发生溢出(UO)。
- 运算发生溢出(OV)或存储溢出(OS)。

当状态位指令以串联方式连接时，该指令将根据 And 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 Or 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：

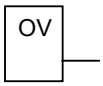


可以通过下列函数求状态字位的值

- 通过整数算术运算
- 通过浮点数算术运算

12.2 OV：溢出异常位

符号



描述

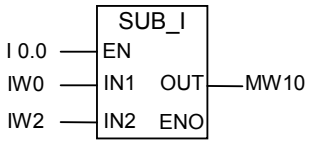
可使用**溢出异常位**指令来检测最后一个算术运算是否有溢出(OV)。如果在系统执行算术运算后，结果超出了许可的负数范围或许可的正数范围，则会设置状态字(参见 CPU 寄存器)的 OV 位。指令将检查此位的信号状态。如果算术运算未出错，则复位该位。

状态字

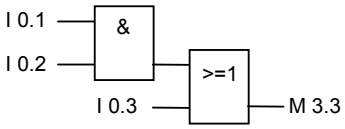
	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	FC
写	-	-	-	-	-	X	X	X	1

举例

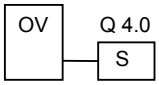
程序段 1



程序段 2



程序段 3

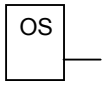


在输入 I0.0 的信号状态为 1 时会激活 SUB_I 框。如果算术运算(输入字 IW0 减输入字 IW2)的结果超出了整数值的许可范围，则会置位状态字的 OV 位。OV 位的信号检查结果为 1。如果 OV 位的检查结果为 1 且程序段 2 的 RLO 为 1 (如果输出 Q4.0 之前的 RLO 为 1)，则会置位输出 Q4.0。

如果输入 I0.0 的信号状态为 0 (未激活)，则 EN 和 ENO 的信号状态均为 0。如果 EN 的信号状态为 1 (已激活)且算术运算的结果超出范围，则 ENO 的信号状态为 0。

12.3 OS：存储的溢出异常位

符号



描述

在与运算中，本指令将按照与运算真值表组合自身的检查结果和先前的逻辑运算结果。在或运算运算中，则将按照或运算真值表组合。

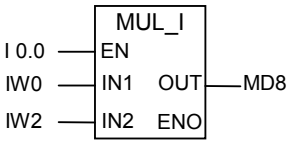
可使用**存储溢出异常位**指令判断先前算术运算结果是否有溢出(存储溢出，OS)。如果在系统执行算术运算后，结果超出了许可的负数范围或许可的负数值范围或者正数值范围，则会设置状态字(参见 CPU 寄存器)的 OS 位。指令将检查此位的信号状态。与 OV(溢出)位不同，即使在执行随后的算术运算时未出现错误，OS 位仍将保留设置(参见溢出异常位)。

状态字

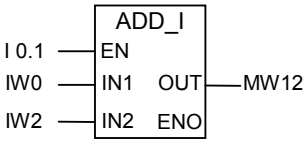
	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	FC
写	-	-	-	-	-	X	X	X	1

举例

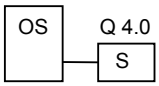
程序段 1



程序段 2



程序段 3



在输入 I0.0 的信号状态为 1 时会激活 MUL_I 框。在输入 I0.1 的信号状态为 1 时会激活 ADD_I 框。如果其中一个算术运算的结果超出整数的许可范围，则会设置状态字的 OS 位。

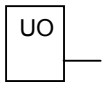
OS 位的信号状态检查的结果为 1 且置位输出 Q4.0。

程序段 1：如果输入 I0.0 的信号状态为 0 (未激活)，则 EN 和 ENO 的信号状态均为 0。如果 EN 的信号状态为 1 (已激活)且算术运算的结果超出范围，则 ENO 的信号状态为 0。

程序段 2：如果输入 I0.1 的信号状态为 0 (未激活)，则 EN 和 ENO 的信号状态均为 0。如果 EN 的信号状态为 1 (已激活)且算术运算的结果超出范围，则 ENO 的信号状态为 0。

12.4 UO：例外位无序

符号



描述

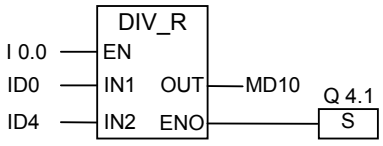
可使用**无序异常位**指令检查浮点数运算的结果是否无序(换言之，该运算中是否存在一个无效的浮点数)。指令将判断状态字的条件码位(CC 1 和 CC 0，参见 CPU 寄存器)。如果运算的结果是无序(UO)，信号状态检查的结果将是 1。如果在 CC 1 和 CC 0 中的组合结果表明不是无序的，则信号状态检查的结果将是 0。

状态字

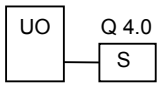
	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	FC
写	-	-	-	-	-	X	X	X	1

举例

程序段 1



程序段 2



在输入 I0.0 的信号状态为 1 时会激活 DIV_R 框。如果输入的双字 ID0 或 ID4 中有一个是无效浮点数，则浮点数运算的结果将是无序。
如果 EN 的信号状态为 1 (已激活)，但在指令执行期间出错，则 ENO 的信号状态为 0。

如果执行了函数 DIV_R，但该运算有一个值是无效的浮点数，则置位输出 Q4.0。如果输入 I0.0 的信号状态为 0 (未激活)，则 EN 和 ENO 的信号状态均为 0。

12.5 BR：BR 存取区异常位

符号

English	German
BR	BIE

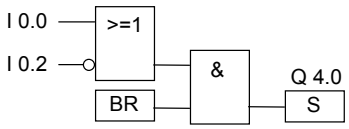
描述

可以使用 **BR 存取区异常位**指令来检查状态字 BR 位(二进制结果)的信号状态(请参见 CPU 寄存器)。

状态字

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

举例



如果输入 I0.0 的信号状态为 1，或者输入 I0.2 的信号状态为 0，将会置位输出端 Q4.0；除了此逻辑运算结果外，BR 位的信号状态为 1 时也会如此。

12.6 <> 0：结果位

符号

== 0 ──	结果位指令**等于 0** 用于确定算术运算指令的结果是否等于 0。
<> 0 ──	结果位指令**不等于 0** 用于确定算术运算指令的结果是否不等于 0。
> 0 ──	结果位指令**大于 0** 用于确定表明算术运算指令的结果是否大于 0。
< 0 ──	结果位指令**小于 0** 用于确定算术运算指令的结果是否小于 0。
>= 0 ──	结果位指令**大于或等于 0** 用于确定算术运算指令的结果是否大于或等于 0。
<= 0 ──	结果位指令**小于或等于 0** 用于确定算术运算指令的结果是否小于或等于 0。

描述

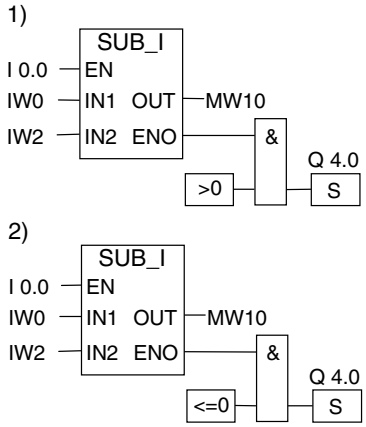
您可以使用**结果位**指令来确定运算的结果与 0 之间的关系，换言之，确定结果是否是 ==0、<>0、>0、<0、>=0 或 <=0。 指令将判断状态字的条件码位(CC 1 和 CC 0，请参见 CPU 寄存器)。如果满足地址中指示的比较条件，则信号状态检查的结果为 1。

在与运算中，本指令将按照与运算真值表组合自身的检查结果和先前的逻辑运算结果(RLO)。在或运算中，本指令将按照或运算真值表组合自身的检查结果和先前的 RLO。

状态字

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

举例



在输入 I0.0 的信号状态为 1 时会激活 SUB_I 框。如果输入字 IW0 的值大于输入字 IW2 的值，则数学运算 IW0 - IW2 的结果大于 0。如果 EN 的信号状态为 1 (已激活)，而在执行指令期间发生错误，则 ENO 的信号状态为 0。

2. 如果正确执行了该运算且结果小于或等于 0，则会将输出端 Q4.0 置位。如果信号输入 I0.0 的信号状态为 0 (未激活)，则 EN 和 ENO 的信号状态为 0。
3. 如果正确执行了该运算且结果小于或等于 0，则会将输出端 Q4.0 置位。如果信号输入 I0.0 的信号状态为 0 (未激活)，则 EN 和 ENO 的信号状态为 0。

13 定时器指令

13.1 定时器指令概述

描述

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

下列定时器指令可用：

- S_PULSE： 设置脉冲定时器参数并启动
- S_PEXT： 设置延时脉冲定时器参数并启动
- S_ODT： 设置接通延时定时器参数并启动
- S_ODTS： 设置掉电保护接通延时定时器参数并启动
- S_OFFDT： 设置断开延时定时器参数并启动
- SP： 启动脉冲定时器
- SE： 启动延时脉冲定时器
- SD： 启动接通延时定时器
- SS： 启动掉电保护接通延时定时器
- SF： 启动延时断开定时器

13.2 定时器的存储区和组件

内存区域

在 CPU 的存储器中，为定时器保留有存储区。此存储区为每个定时器地址保留一个 16 位字。在 FBD 中编程时，可支持 256 个定时器。请参考 CPU 的技术信息来确定可用的定时器字数目。

下列功能访问定时器存储区：

- 定时器指令
- 按时钟定时更新定时器字。在 RUN 模式下，此 CPU 功能按时间基准所指定的间隔将给定的时间值减少一个单位，直到该时间值等于零。

时间值

定时器字的 0 到 9 位包含二进制代码的时间值。时间值指定单位数。更新定时器时，时间值按时间基准所指定的间隔递减一个单位。时间值递减到等于零为止。

可通过下列任一种格式预加载时间值：

- 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，值会根据时间基准四舍五入到下一个更低的数值。

可以输入的最大时间值是 9,990S 或 2H_46M_30S。

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 位包含二进制代码的时间基准。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。

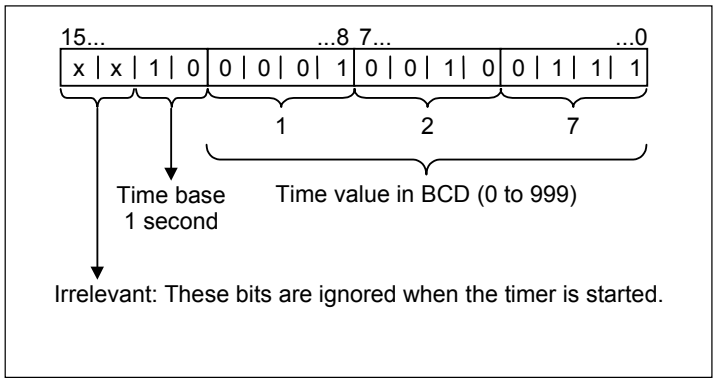
时间基准	时间基准的二进制代码
10 毫秒	00
100 毫秒	01
1 秒	10
10 秒	11

由于时间值仅按一个时间间隔保存，因此不是时间间隔整倍数的值将被截断。对于所需范围，分辨率过高的值将舍入到所需的范围，而不是期望的分辨率。下表显示了可能的分辨率和相应的范围。

分辨率	时间基准
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 2HR_46M_30S

定时器单元格中的位组态

当启动定时器时，定时器单元格的内容将用作时间值。定时器单元格的 0 到 11 位所包含的时间值显示为二进制编码的十进制格式(BCD 格式：每四位为一组包含对应一个十进制值的二进制代码)。12 和 13 位包含二进制代码格式的时间基准。下图显示了以定时器值 127 (时间基准为 1 秒)加载的定时器单元格的内容。

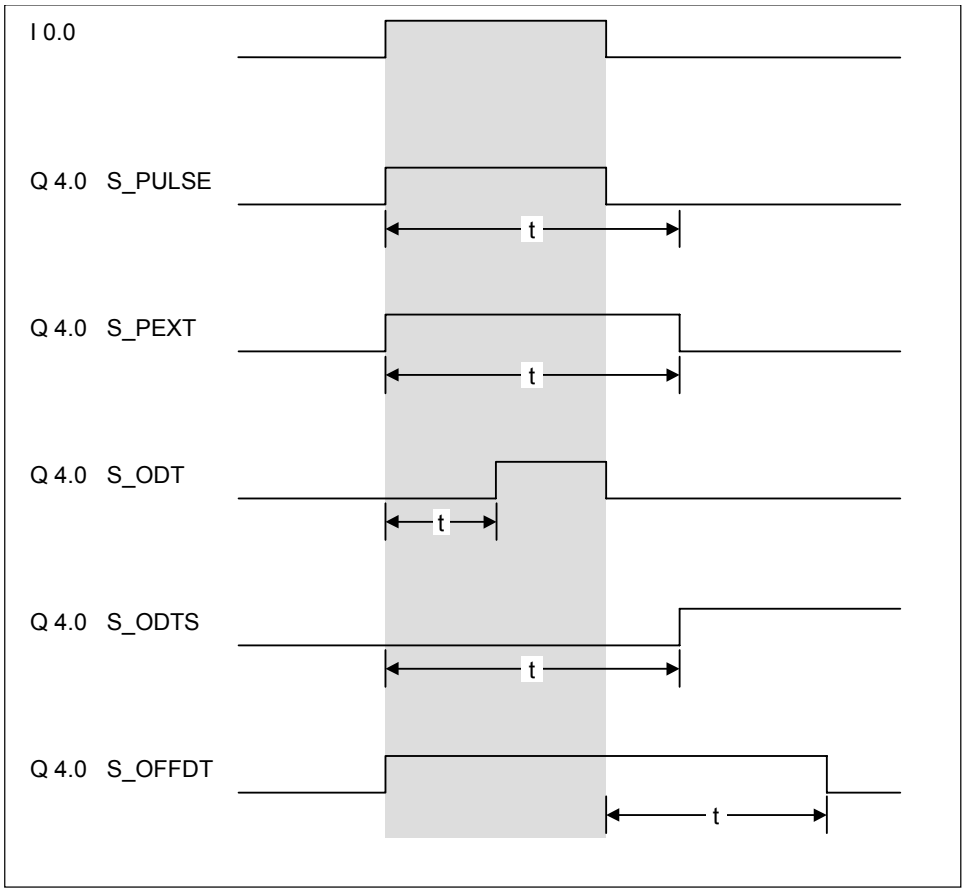


读取时间和时间基准

每个定时器框都提供了两个输出(BI 和 BCD)，用户可为其指定字位置。BI 输出提供二进制格式的时间值，但不显示时间基准。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

此概述旨在帮助用户正确选择定时作业的定时器。

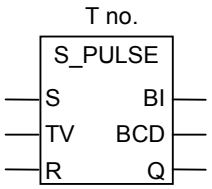


定时器	描述
S_PULSE 脉冲定时器	输出信号保持为 1 的最大时间与设定的时间值 t 相同。如果输入信号变为 0，则输出信号在较短的时间内保持为 1。
S_PEXT 延时脉冲定时器	输出信号在设定的时长内保持为 1，无论输入信号保持 1 多长时间。
S_ODT 接通延时定时器	只有在设定的时间已过且输入信号仍为 1 时，输出信号才变为 1。
S_ODTS 保持性接通延时定时器	只有在设定的时间已过时，输出信号才从 0 变为 1，无论输入信号保持 1 多长时间。
S_OFFDT 断开延时定时器	输入信号变为 1 或定时器运行时，输出信号变为 1。输入信号从 1 变为 0 时，时间启动。

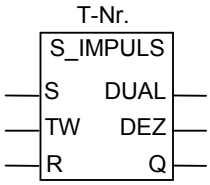
13.3 S_PULSE：设置脉冲定时器参数并启动

符号

English



German



参数 英语	参数 德语	数据类型	内存区域	描述
no.	Nr.	TIMER	T	定时器标识号。 范围取决于 CPU。
S	S	BOOL	I、Q、M、D、 L、T、C	使能输入
TV	TW	S5TIME	I、Q、M、D、L 或常数	预设时间值(范围 0-9999)
R	R	BOOL	I、Q、M、D、 L、T、C	复位输入端
BI	DUAL	WORD	I、Q、M、D、L	剩余时间 (值为整型格式)
BCD	DEZ	WORD	I、Q、M、D、L	剩余时间 (值为 BCD 格式)
Q	Q	BOOL	I、Q、M、D、L	定时器的状态

描述

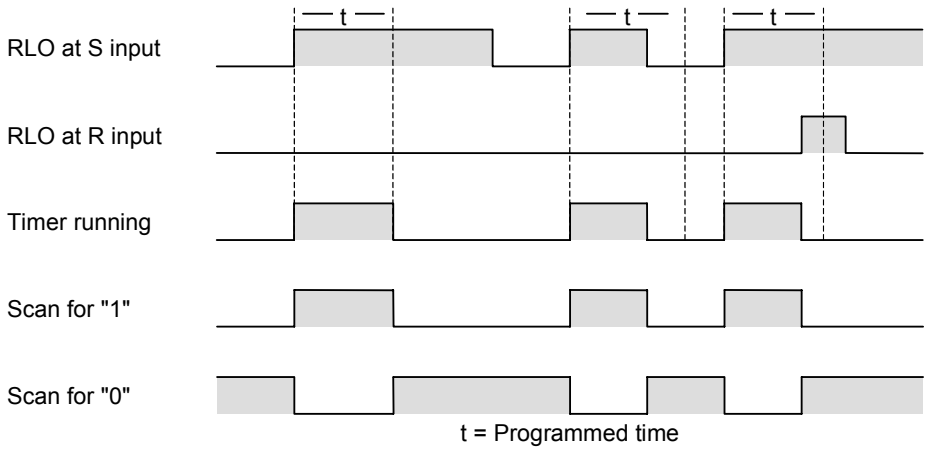
如果“启动(S)”输入有上升沿(信号状态从 0 改变为 1)，**设置脉冲定时器参数并启动**指令将启动指定的定时器。要启动定时器，必须有信号改变。只要输入 S 的信号状态为 1，定时器将继续运行“时间值(TV)”输入处指定的时间。当定时器运行时，对输出 Q 的信号状态 1 的检查产生结果 1。如果在时间用完前，S 输入存在从 1 到 0 的改变，则定时器停止。随后对输出 Q 的信号状态 1 的检查产生结果 0。

定时器正运行时，如果“复位(R)”输入从 0 改变为 1，则定时器复位。此改变还会将时间和时间基准复位到零。如果定时器未运行，则定时器 R 输入处的信号状态 1 将不起作用。

可在输出 BI 和 BCD 处扫描当前时间值。BI 处的时间值为二进制格式；BCD 处的时间值为二进制编码的十进制格式。

时序图

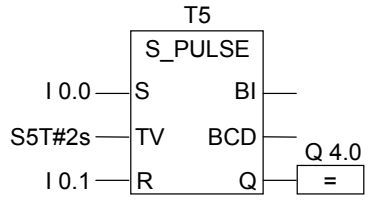
脉冲定时器特性：



状态字

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

举例

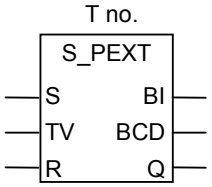


如果输入 I0.0 的信号状态从 0 改变为 1 (如果 RLO 中有上升沿)，则定时器 T5 启动。只要输入 I0.0 为 1，定时器将继续运行指定的两秒(2s)时间。如果在时间用尽前输入 I0.0 的信号状态从 1 改变为 0，则定时器停止。如果定时器运行时，输入 I0.1 的信号状态从 0 改变为 1，则定时器复位。只要定时器运行，输出 Q4.0 的信号状态就为 1。

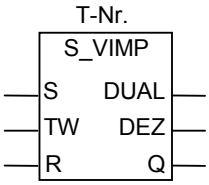
13.4 S_PEXT：设置延时脉冲定时器参数并启动

符号

English



German



参数 英语	参数 德语	数据类型	内存区域	描述
no.	Nr.	TIMER	T	定时器标识号。 范围取决于 CPU。
S	S	BOOL	I、Q、M、D、L、T、C	使能输入
TV	TW	S5TIME	I、Q、M、D、L 或常数	预设时间值 (范围 0-9999)
R	R	BOOL	I、Q、M、D、L、T、C	复位输入端
BI	DUAL	WORD	I、Q、M、D、L	剩余时间 (值为整型格式)
BCD	DEZ	WORD	I、Q、M、D、L	剩余时间 (值为 BCD 格式)
Q	Q	BOOL	I、Q、M、D、L	定时器的状态

描述

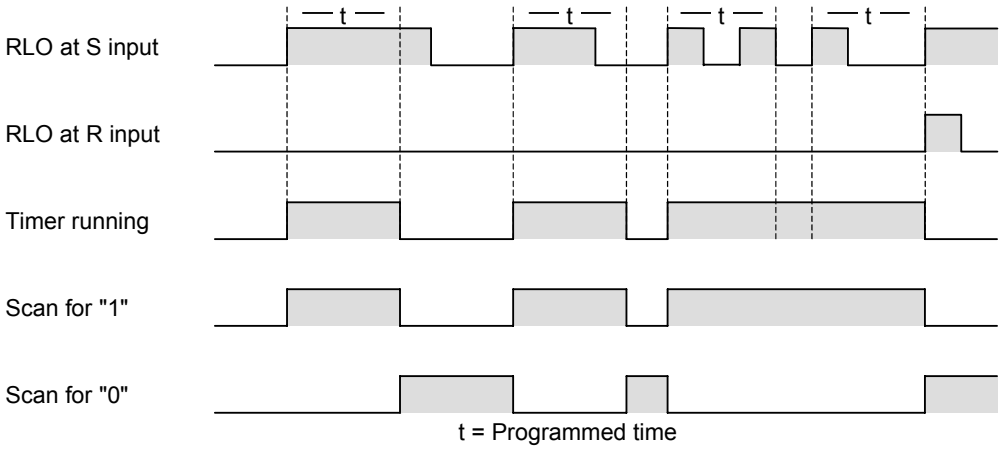
如果“启动(S)”输入处有上升沿(信号状态从 0 改变为 1)，**设置延时脉冲定时器参数并启动**指令将启动指定的定时器。要启动定时器，必须有信号改变。定定时器继续运行“时间值(TV)”输入处指定的时间，即使 S 输入处的信号状态在时间用尽前改变为 0。只要定时器运行，对输出 Q 信号状态 1 的检查就产生结果 1。定时器运行时，如果输入 S 处的信号状态从 0 改变为 1，则定时器以指定的时间重新启动。

定时器运行时，如果“复位(R)”输入从 0 改变为 1，则定时器复位。此改变还会将时间和时间基准复位到零。此改变还会将时间和时间基准复位到零。

可在输出端 BI 和 BCD 扫描当前时间值。BI 处的时间值为二进制格式；BCD 处的时间值为二进制编码的十进制格式。

时序图

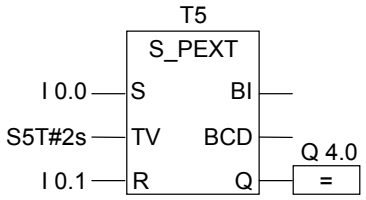
扩充的脉冲定时器特性：



状态字

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

举例

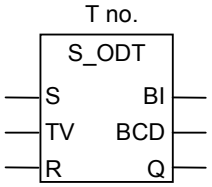


如果输入 I0.0 的信号状态从 0 改变为 1 (RLO 中有上升沿)，则定时器 T5 启动。定时器将继续运行指定的两秒(2s)时间，而不考虑输入 S 的下降沿。如果在时间用尽前输入 I0.0 的信号状态从 0 改变为 1，则定时器重新启动。如果定时器运行时，输入 I0.1 的信号状态从 0 改变为 1，则定时器复位。只要定时器运行，输出 Q4.0 的信号状态就为 1。

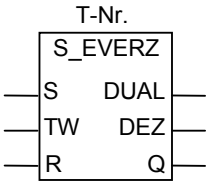
13.5 S_ODT：设置接通延时定时器参数并启动

符号

English



German



参数 英语	参数 德语	数据类型	内存区域	描述
no.	Nr.	TIMER	T	定时器标识号。 范围取决于 CPU。
S	S	BOOL	I、Q、M、D、L、 T、C	使能输入
TV	TW	S5TIME	I、Q、M、D、L 或 常数	预设时间值 (范围 0-9999)
R	R	BOOL	I、Q、M、D、L、 T、C	复位输入端
BI	DUAL	WORD	I、Q、M、D、L	剩余时间 (值为整型格式)
BCD	DEZ	WORD	I、Q、M、D、L	剩余时间 (值为 BCD 格式)
Q	Q	BOOL	I、Q、M、D、L	定时器的状态

描述

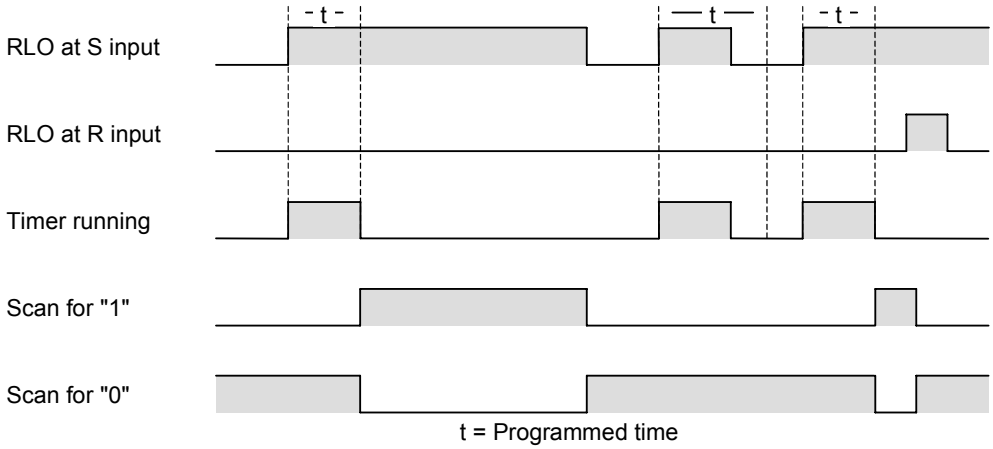
如果“启动(S)”输入有上升沿(信号状态从 0 改变为 1)，**设置接通延时定时器参数并启动**指令将启动指定的定时器。要启动定时器，必须有信号改变。只要输入 S 的信号状态为 1，定时器将继续运行“时间值(TV)”输入处指定的时间。当时间已用尽且未出错，而输入 S 的信号状态仍为 1 时，则对输出 Q 的信号状态 1 的检查将产生结果 1。定时器运行时，如果输入 S 的信号状态从 1 改变为 0，定时器将停止。这种情况下，对输出 Q 的信号状态 1 的检查将始终产生结果 0。

定时器正运行时，如果“复位(R)”输入从 0 改变为 1，则定时器复位。此改变还会将时间和时间基准复位到零。定时器未运行时，如果 R 输入的信号状态为 1，定时器也会复位。

可在输出端 BI 和 BCD 扫描当前时间值。BI 处的时间值为二进制格式；BCD 处的时间值为二进制编码的十进制格式。

时序图

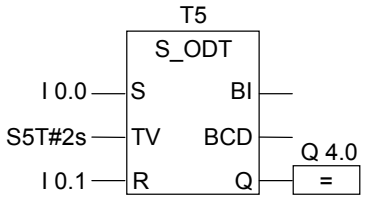
接通延时定时器特征：



状态字

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

举例

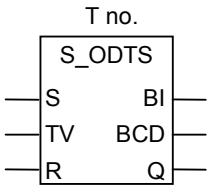


如果输入 I0.0 的信号状态从 0 改变为 1 (RLO 中有上升沿)，则定时器 T5 启动。如果指定的时间两秒(2s)已过，且输入 I0.0 的信号状态仍为 1，则输出 Q4.0 的信号状态为 1。如果输入 I0.0 的信号状态从 1 改变为 0，则定时器停止并且输出 Q4.0 为 0。定时器运行时，如果输入 I0.1 的信号状态从 0 改变为 1，定时器将重新启动。

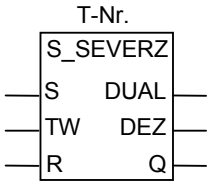
13.6 S_ODTS：设置掉电保护接通延时定时器参数并启动

符号

English



German



参数 英语	参数 德语	数据类型	内存区域	描述
no.	Nr.	TIMER	T	定时器标识号。 范围取决于 CPU。
S	S	BOOL	I、Q、M、D、L、 T、C	使能输入
TV	TW	S5TIME	I、Q、M、D、L 或 常数	预设时间值 (范围 0-9999)
R	R	BOOL	I、Q、M、D、L、 T、C	复位输入端
BI	DUAL	WORD	I、Q、M、D、L	剩余时间 (值为整型格式)
BCD	DEZ	WORD	I、Q、M、D、L	剩余时间 (值为 BCD 格式)
Q	Q	BOOL	I、Q、M、D、L	定时器的状态

描述

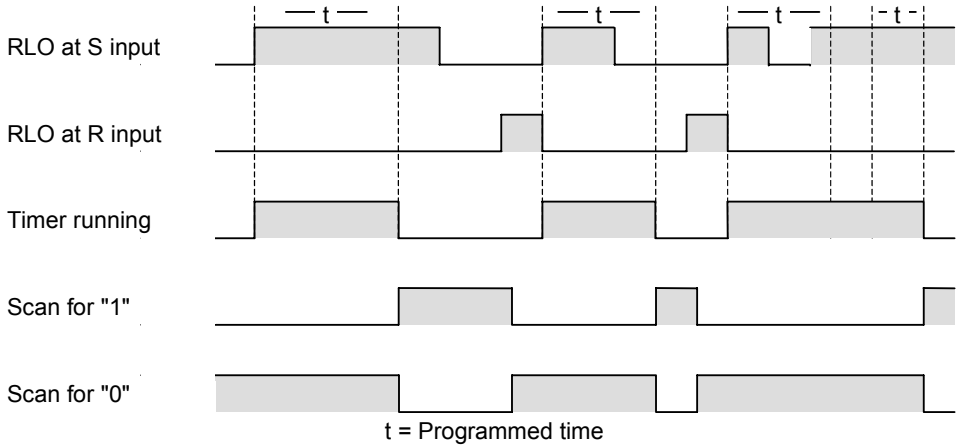
如果“启动(S)”输入有上升沿(信号状态从 0 改变为 1)，**设置掉电保护接通延时定时器参数并启动**指令将启动指定的定时器。要启动定时器，必须有信号改变。定时器继续运行“时间值(TV)”输入处指定的时间，即使输入 S 的信号状态在定时器时间用尽前改变为 0。定时器时间用尽时，对输出 Q 的信号状态 1 的检查将产生结果 1，而不考虑复位输入端(R)保持为 0 时输入 S 的信号状态。定时器运行时，如果输入 S 的信号状态从 0 改变 1，则定时器以指定的时间重新启动。

定时器“复位(R)”输入从 0 改变为 1 时，定时器将复位，而不考虑 S 输入的 RLO。

可在输出端 BI 和 BCD 扫描当前时间值。BI 处的时间值为二进制格式；BCD 处的时间值为二进制编码的十进制格式。

时序图

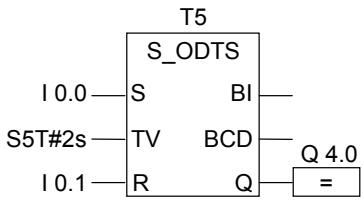
掉电保护的接通延时定时器特性：



状态字

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

举例

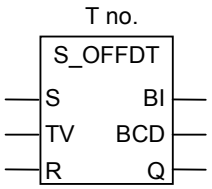


如果输入 I0.0 的信号状态从 0 改变为 1 (RLO 中有上升沿)，则定时器 T5 启动。定时器将继续运行，而不考虑输入 I0.0 处的信号状态从 1 改变到 0。如果在时间用尽前输入 I0.0 的信号状态从 0 改变为 1，定时器将重新启动。如果定时器运行时，输入 I0.1 的信号状态从 0 改变为 1，定时器将重新启动。如果时间已用尽，且 I0.1 的信号状态仍为 0，则输出 Q4.0 的信号状态为 1。

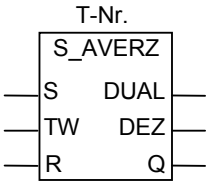
13.7 S_OFFDT：设置断开延时定时器参数并启动

符号

English



German



参数 英语	参数 德语	数据类型	内存区域	描述
no.	Nr.	TIMER	T	定时器标识号。 范围取决于 CPU。
S	S	BOOL	I、Q、M、D、 L、T、C	使能输入
TV	TW	S5TIME	I、Q、M、D、L 或常数	预设时间值(范围 0-9999)
R	R	BOOL	I、Q、M、D、 L、T、C	复位输入端
BI	DUAL	WORD	I、Q、M、D、L	剩余时间 (值为整型格式)
BCD	DEZ	WORD	I、Q、M、D、L	剩余时间 (值为 BCD 格式)
Q	Q	BOOL	I、Q、M、D、L	定时器的状态

描述

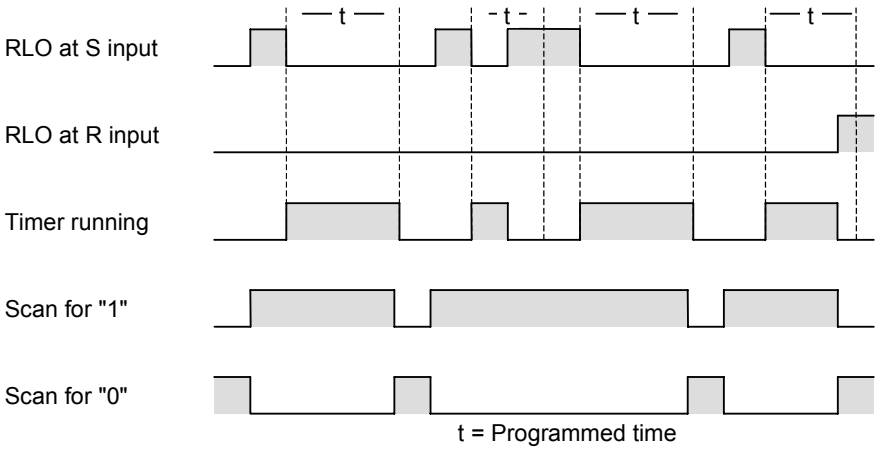
如果“启动(S)”输入有下降沿(信号状态从 1 改变为 0)，**设置断开延时定时器参数并启动**指令将启动指定的定时器。要启动定时器，必须有信号改变。当 S 输入的信号状态为 1，或定时器运行时，对输出 Q 处的信号状态 1 的检查结果为 1。定时器运行时，当输入 S 的信号状态从 0 改变为 1 时，定时器复位。直到输入 S 的信号状态再次从 1 改变为 0 时，定时器才重新启动。

定时器运行时，如果“复位(R)”输入从 0 改变为 1，定时器将复位。

可在输出 BI 和 BCD 处扫描实际时间值。BI 处的时间值为二进制格式；BCD 处的时间值为二进制编码的十进制格式。

时序图

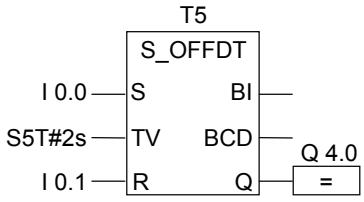
断开延时定时器特性：



状态字

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

举例

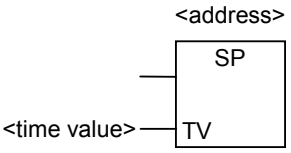


如果输入 I0.0 的信号状态从 1 改变为 0，则定时器启动。当 I0.0 为 1 或定时器运行时，输出 Q4.0 为 1。如果定时器运行时，I0.1 的信号状态从 0 改变为 1，定时器将复位。

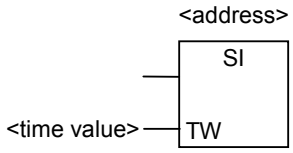
13.8 SP：启动脉冲定时器

符号

English



German



参数 英语	参数 德语	数据类型	内存区域	描述
Timer no.	Timer no.	TIMER	T	此地址指定要启动的定时器的编号。
TV	TW	S5TIME	E、A、M、D、L 或常数	定时器值(S5TIME 格式)

描述

如果 RLO 中有上升沿(信号状态从 0 改变为 1)，**启动脉冲定时器**指令将启动指定的定时器。只要 RLO 为高电平，定时器就以指定的值继续运行。定时器运行时，对信号状态 1 的检查结果为 1。如果在定时器时间用尽前 RLO 从 1 改变为 0，定时器将停止。这种情况下，对信号状态 1 的检查结果为 0。

有关定时器的存储区和组件的详细信息，请参见存储区。

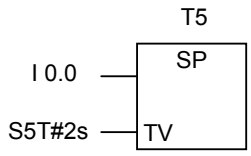
只能将**启动脉冲定时器**框放在逻辑串的右端。可以使用若干个**启动脉冲定时器**框。

状态字

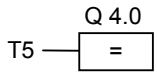
	BR	**CC 1**	**CC 0**	**OV**	**OS**	**OR**	**STA**	**RLO**	**FC**
写	-	-	-	-	-	0	-	-	0

举例

程序段 1



程序段 2

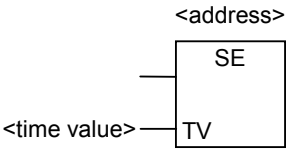


如果输入 I0.0 的信号状态从 0 改变为 1 (RLO 中有上升沿)，则定时器 T5 启动。只要信号状态为 1，定时器就以指定的值(2 秒)继续运行。如果定时器运行时，I0.0 的信号状态从 1 改变为 0，定时器将停止。
定时器运行时，输出 Q4.0 为 1。

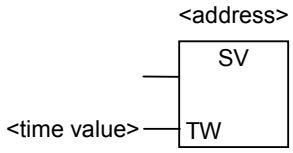
13.9 SE：启动延时脉冲定时器

符号

English



German



参数 英语	参数 德语	数据类型	内存区域	描述
Timer no.	Timer no.	TIMER	T	此地址指定要启动的定时器的编号。
TV	TW	S5TIME	E、A、M、D、L 或常数	定时器值(S5TIME 格式)

描述

如果 RLO 中有上升沿(信号状态从 0 改变为 1)，**启动延时脉冲定时器**指令将启动指定的定时器。如果在定时器时间用尽前 RLO 改变为 0，定时器将继续以指定的值运行。定时器运行时，对信号状态 1 的检查结果为 1。定时器正运行时，如果 RLO 从 0 改变为 1，定时器将以指定的时间重新启动。

有关定时器的存储区和组件的详细信息，请参见存储区。

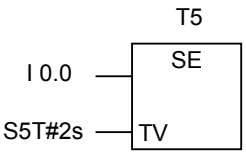
只能将**启动延时脉冲定时器**框放在逻辑串的右端。可以使用若干个**启动延时脉冲定时器**框。

状态字

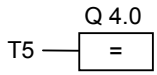
	BR	**CC 1**	**CC 0**	**OV**	**OS**	**OR**	**STA**	**RLO**	**FC**
写	-	-	-	-	-	0	-	-	0

举例

程序段 1



程序段 2

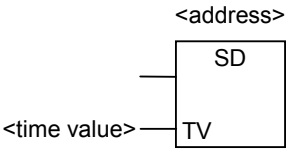


如果输入 I0.0 的信号状态从 0 改变为 1 (RLO 中有上升沿)，则定时器 T5 启动。定时器继续运行而不受 RLO 中下降沿的影响。如果在指定的时间用尽前 I0.0 的信号状态从 0 改变为 1，定时器将重新启动。
定时器运行时，输出 Q4.0 为 1。

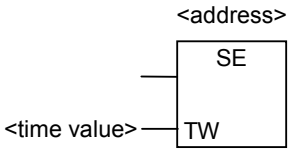
13.10 SD：启动接通延时定时器

符号

English



German



参数 英语	参数 德语	数据类型	内存区域	描述
Timer no.	Timer no.	TIMER	T	此地址指定要启动的定时器的编号。
TV	TW	S5TIME	E、A、M、 D、L 或常数	定时器值(S5TIME 格式)

描述

如果 RLO 中有上升沿(信号状态从 0 改变为 1)，**启动接通延时定时器**指令将启动指定的定时器。如果指定的时间已用尽且未出错，而 RLO 的值仍为 1，对信号状态 1 的检查结果为 1。定时器运行时，如果 RLO 从 1 改变为 0，定时器将停止。这种情况下，对信号状态 1 的检查结果始终为 0。

有关定时器的存储区和组件的详细信息，请参见存储区。

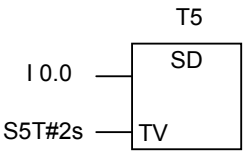
只能将**启动接通延时定时器**框放在逻辑串的右端。可以使用若干个**启动接通延时定时器**框。

状态字

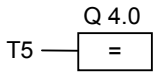
	BR	**CC 1**	**CC 0**	**OV**	**OS**	**OR**	**STA**	**RLO**	**FC**
写	-	-	-	-	-	0	-	-	0

举例

程序段 1



程序段 2

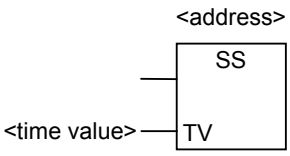


如果输入 I0.0 的信号状态从 0 改变为 1 (RLO 中有上升沿)，则定时器 T5 启动。当定时器时间用尽时，如果 I0.0 的信号状态仍为 1，则输出 Q4.0 为 1。如果信号状态从 1 改变为 0，定时器将停止。

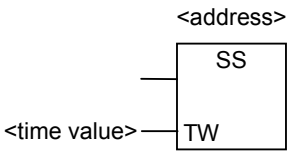
13.11 SS：启动掉电保护接通延时定时器

符号

English



German



参数 英语	参数 德语	数据类型	内存区域	描述
Timer no.	Timer no.	TIMER	T	此地址指定要启动的定时器的编号。
TV	TW	S5TIME	E、A、M、D、L 或常数	定时器值(S5TIME 格式)

描述

如果 RLO 中有上升沿(信号状态从 0 改变为 1)，**启动掉电保护接通延时定时器**指令可启动指定的定时器。如果在定时器时间用尽前 RLO 改变为 0，定时器将继续以指定的值运行。如果定时器时间已用尽，则对信号状态 1 的检查结果为 1，而与 RLO 无关。定时器正运行时，如果 RLO 从 0 改变为 1，定时器将以指定的时间重新启动。

有关定时器的存储区和组件的详细信息，请参见存储区。

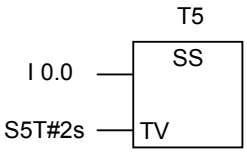
只能将**启动掉电保护接通延时定时器**框放在逻辑串的右端。可以使用若干个**启动掉电保护接通延时定时器**框。

状态字

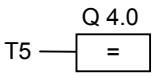
	BR	**CC 1**	**CC 0**	**OV**	**OS**	**OR**	**STA**	**RLO**	**FC**
写	-	-	-	-	-	0	-	-	0

举例

程序段 1



程序段 2

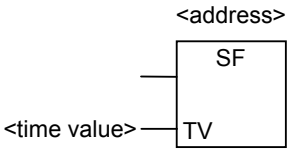


如果输入 I0.0 的信号状态从 0 改变为 1 (RLO 中有上升沿)，则定时器 T5 启动。定时器将继续运行，而与输入 I0.0 上信号状态是否从 1 改变为 0 无关。如果在指定的时间用尽前信号状态从 0 改变为 1，定时器将重新启动。
定时器时间用尽后，输出 Q4.0 为 1。

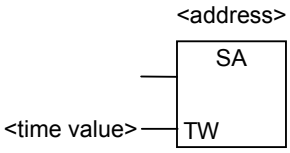
13.12 SF 启动断开延迟定时器

符号

English



German



参数 英语	参数 德语	数据类型	内存区域	描述
Timer no.	Timer no.	TIMER	T	此地址指定要启动的定时器的编号。
TV	TW	S5TIME	E、A、M、 D、L 或常数	定时器值(S5TIME 格式)

描述

如果 RLO 中有下降沿(信号状态从 1 改变为 0)，**启动延时断开定时器**指令将启动指定的定时器。如果 RLO 为 1，或定时器在运行，则对信号状态 1 的检查结果为 1。定时器运行时，如果 RLO 从 0 改变为 1，定时器将复位。当 RLO 从 1 改变为 0 后，定时器将重新启动。

有关定时器的存储区和组件的详细信息，请参见存储区。

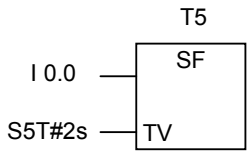
只能将**启动延时断开定时器**框放在逻辑串的右端。可以使用若干个**启动延时断开定时器**框。

状态字

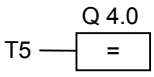
	BR	**CC 1**	**CC 0**	**OV**	**OS**	**OR**	**STA**	**RLO**	**FC**
写	-	-	-	-	-	0	-	-	0

举例

程序段 1



程序段 2



如果输入 I0.0 的信号状态从 1 改变为 0，则定时器启动。

如果信号状态从 0 改变为 1，定时器将复位。

如果输入 I0.0 的信号状态为 1，或定时器正运行，则输出 Q4.0 的信号状态为 1。

14 字逻辑指令

14.1 字逻辑指令概述

描述

字逻辑指令按照布尔逻辑按位比较字(16 位)和双字(32 位)对。

相对于 0 的输出 OUT 的结果值对状态字中的位具有以下影响：

如果输出 OUT 的结果不等于 0，则状态字中的 CC 1 位会被设置为 1。

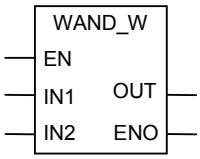
如果输出 OUT 的结果为 0，则状态字中的 CC 1 位会被设置为 0。

要执行字逻辑运算，可使用下列指令：

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

14.2 WAND_W：单字与运算(字)

符号



参数	数据类型	内存区域	描述
EN	BOOL	I、Q、M、D、L、T、C	使能输入
IN1	WORD	I、Q、M、D、L 或常数	逻辑运算的第一个值
IN2	WORD	I、Q、M、D、L 或常数	逻辑运算的第二个值
OUT	WORD	I、Q、M、D、L	逻辑运算结果
ENO	BOOL	I、Q、M、D、L	使能输出

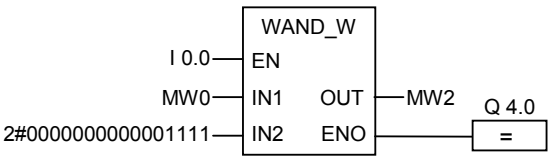
描述

(字)单字与运算指令由位于使能输入(EN)的信号状态 1 激活，并根据与运算真值表逐位组合位于输入 IN1 和 IN2 的两个数值。这些值被解释为纯位模式。结果可在输出 OUT 处扫描。ENO 具有与 EN 相同的信号状态。

状态字

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

举例



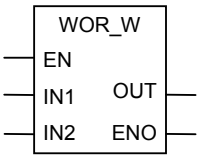
当 I0.0 的信号状态为 1 时会激活该指令。只有 0 到 3 之间的位是相关的，所有其它 MW0 位都被屏蔽。

IN1 = 0101010101010101
IN2 = 0000000000001111
OUT = 0000000000000101

如果执行了该指令，则 Q4.0 为 1。

14.3 WOR_W：单字或运算(字)

符号



参数	数据类型	内存区域	描述
EN	BOOL	I、Q、M、D、L、T、C	使能输入
IN1	WORD	I、Q、M、D、L 或常数	逻辑运算的第一个值
IN2	WORD	I、Q、M、D、L 或常数	逻辑运算的第二个值
OUT	WORD	I、Q、M、D、L	逻辑运算结果
ENO	BOOL	I、Q、M、D、L	使能输出

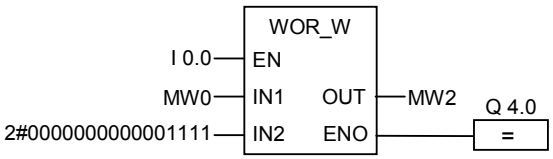
描述

(字)单字或运算指令由位于使能输入(EN)的信号状态 1 激活，并根据或运算真值表逐位组合位于输入 IN1 和 IN2 的两个数值。这些值被解释为纯位模式。结果可在输出 OUT 处扫描。ENO 具有与 EN 相同的信号状态。

状态字

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

举例



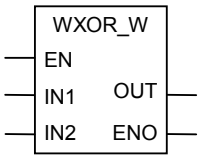
当 I0.0 为 1 时将激活该指令。对 MW0 中的位和常数中的位执行或运算，0 到 3 之间的位被设置为 1，MW0 的所有其它位均以不变形式被输入 MW2 中

IN1 = 0101010101010101
IN2 = 0000000000001111
OUT = 0101010101011111

如果执行了该指令，则 Q4.0 为 1。

14.4 WXOR_W：单字异或运算(字)

符号



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

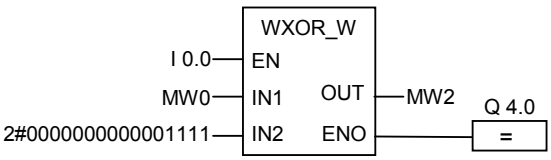
描述

(字)单字或运算指令由位于使能输入(EN)的信号状态 1 激活，并根据或运算真值表逐位组合位于输入 IN1 和 IN2 的两个数值。这些值被解释为纯位模式。结果可在输出 OUT 处扫描。ENO 具有与 EN 相同的信号状态。

状态字

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

举例



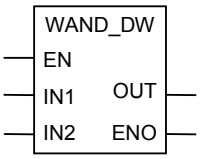
当输入 I0.0 为 1 时，会激活该指令。

IN1 = 0101010101010101
IN2 = 0000000000001111
OUT = 0101010101011010

如果执行了该指令，则 Q4.0 为 1。

14.5 WAND_DW：双字与运算(字)

符号



参数	数据类型	内存区域	描述
EN	BOOL	I、Q、M、D、L、T、C	使能输入
IN1	DWORD	I、Q、M、D 或常数 L	逻辑运算的第一个值
IN2	DWORD	I、Q、M、D、L 或常数	逻辑运算的第二个值
OUT	DWORD	I、Q、M、D、L	逻辑运算结果
ENO	BOOL	I、Q、M、D、L	使能输出

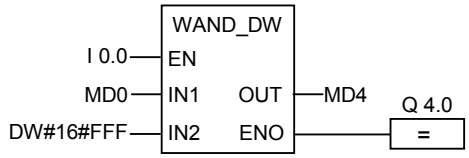
描述

(字)双字与运算指令由位于使能输入(EN)的信号状态 1 激活，并根据与运算真值表逐位组合位于输入 IN1 和 IN2 的两个数值。这些值被解释为纯位模式。结果可在输出 OUT 处扫描。ENO 具有与 EN 相同的信号状态。

状态字

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

举例



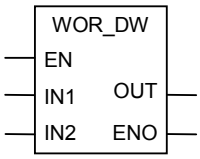
当 I0.0 为 1 时会激活该指令。只有 0 到 11 之间的位是相关的，所有其它 MD4 位都被屏蔽。

IN1	=	0101010101010101	0101010101010101
IN2	=	0000000000000000	0000111111111111
OUT	=	0000000000000000	0000010101010101

如果执行了该指令，则 Q4.0 为 1。

14.6 WOR_DW：双字或运算(字)

符号



参数	数据类型	内存区域	描述
EN	BOOL	I、Q、M、D、L、T、C	使能输入
IN1	DWORD	I、Q、M、D、L 或常数	逻辑运算的第一个值
IN2	DWORD	I、Q、M、D、L 或常数	逻辑运算的第二个值
OUT	DWORD	I、Q、M、D、L	逻辑运算结果
ENO	BOOL	I、Q、M、D、L	使能输出

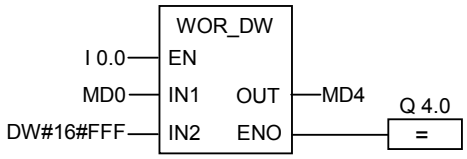
描述

(字)双字或运算指令由位于使能输入(EN)的信号状态 1 激活，并根据或运算真值表逐位组合位于输入 IN1 和 IN2 的两个数值。这些值被解释为纯位模式。结果可在输出 OUT 处扫描。ENO 具有与 EN 相同的信号状态。

状态字

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

举例



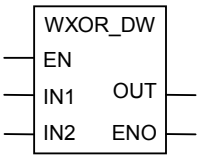
当 I0.0 为 1 时将激活该指令。对 MD0 中的位和常数中的位执行或运算，0 到 11 之间的位被设置为 1，MD0 的所有其它位均以不变形式被输入 MD4 中。

IN1	=	0101010101010101	0101010101010101
IN2	=	0000000000000000	0000111111111111
OUT	=	0101010101010101	0101111111111111

如果执行了该指令，则 Q4.0 为 1。

14.7 WXOR_DW：双字异或运算(字)

符号



参数	数据类型	内存区域	描述
EN	BOOL	I、Q、M、D、L、T、C	使能输入
IN1	DWORD	I、Q、M、D、L 或常数	逻辑运算的第一个值
IN2	DWORD	I、Q、M、D、L 或常数	逻辑运算的第二个值
OUT	DWORD	I、Q、M、D、L	逻辑运算结果
ENO	BOOL	I、Q、M、D、L	使能输出

描述

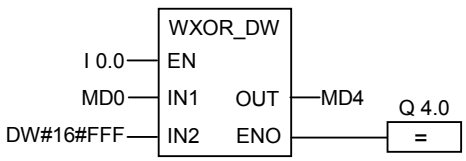
(字)双字异或运算指令由位于使能输入(EN)的信号状态 1 激活，并根据异或运算真值表逐位组合位于输入 IN1 和 IN2 的两个数值。这些值被解释为纯位模式。结果可在输出 OUT 处扫描。

ENO 具有与 EN 相同的信号状态。

状态字

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

举例



当输入 I0.0 为 1 时，会激活该指令。

IN1	=	0101010101010101	0101010101010101
IN2	=	0000000000000000	0000111111111111
OUT	=	0101010101010101	0101101010101010

如果执行了该指令，则 Q4.0 为 1。

A 全部 FBD 指令概述

A.1 根据德语助记符(SIMATIC)排序的 FBD 指令

德语助记符	英语助记符	程序元素分类	描述
&	&	位逻辑指令	“与”逻辑操作
>=1	>=1	位逻辑指令	“或”逻辑操作
=	=	位逻辑指令	赋值
#	#	位逻辑指令	中间输出
---I	---I	位逻辑指令	插入数字输入
---oI	---oI	位逻辑指令	数字输入取反
==0	==0	状态位	结果位
>0	>0	状态位	结果位
>=0	>=0	状态位	结果位
<0	<0	状态位	结果位
<=0	<=0	状态位	结果位
<>0	<>0	状态位	结果位
ABS	ABS	浮点算术运算指令	浮点数的绝对值运算
ACOS	ACOS	浮点算术运算指令	计算以浮点数表示的角的三角函数
ADD_DI	ADD_DI	整数算术运算指令	加双精度整型
ADD_I	ADD_I	整数算术运算指令	加整型
ADD_R	ADD_R	浮点算术运算指令	加实型
ASIN	ASIN	浮点算术运算指令	计算以浮点数表示的角的三角函数
ATAN	ATAN	浮点算术运算指令	计算以浮点数表示的角的三角函数
BCD_DI	BCD_DI	转换	BCD 码转换为双精度整型
BCD_I	BCD_I	转换	BCD 码转换为整型
BIE	BR	状态位	BR 存取区异常位
CALL	CALL	程序控制指令	调用无参数的 FC/SFC
CALL_FB	CALL_FB	程序控制指令	CALL_FB (以框方式调用 FB)
CALL_FC	CALL_FC	程序控制指令	CALL_FC (以框方式调用 FC)
CALL_SFB	CALL_SFB	程序控制指令	CALL_SFB (以框方式调用系统 FB)
CALL_SFC	CALL_SFC	程序控制指令	CALL_SFC (以框方式调用系统 FC)
CEIL	CEIL	转换指令	上限
CMP ?D	CMP ?D	比较指令	比较双精度整数

德语助记符	英语助记符	程序元素分类	描述
CMP ?I	CMP ?I	比较指令	整数比较
CMP ?R	CMP ?R	比较指令	比较实数
COS	COS	浮点算术运算指令	计算以浮点数表示的角的三角函数
DI_BCD	DI_BCD	转换指令	双精度整型转换为 BCD 码
DI_R	DI_R	转换指令	双精度整型转换为实型
DIV_DI	DIV_DI	整数算术运算指令	除双精度整型
DIV_I	DIV_I	整数算术运算指令	除整型
DIV_R	DIV_R	浮点算术运算指令	除实型
EXP	EXP	浮点算术运算指令	计算浮点数的指数值
FLOOR	FLOOR	转换指令	基数
I_BCD	I_BCD	转换指令	整型转换为 BCD 码
I_DI	I_DI	转换指令	整型转换为双精度整型
INV_I	INV_I	转换指令	对整型数求反码
INV_DI	INV_DI	转换指令	二进制反码双精度整型
JMP	JMP	跳转	块中无条件跳转
JMP	JMP	跳转	块中有条件跳转
JMPN	JMPN	跳转	若非则跳转
LABEL	LABEL	跳转	跳转标签
LN	LN	浮点算术运算指令	计算浮点数的自然对数
MCR>	MCR>	程序控制	主控继电器开/关
MCR<	MCR<	程序控制	主控继电器开/关
MCRA	MCRA	程序控制	主控继电器激活/去活
MCRD	MCRD	程序控制	主控继电器激活/去活
MOD_DI	MOD_DI	整数算术运算指令	返回分数双精度整型
MOVE	MOVE	传送	赋值
MUL_DI	MUL_DI	整数算术运算指令	乘双精度整型
MUL_I	MUL_I	整数算术运算指令	乘整型
MUL_R	MUL_R	浮点算术运算指令	乘实型
N	N	位逻辑指令	RLO 负跳沿检测
NEG	NEG	位逻辑指令	地址负跳沿检测
NEG_DI	NEG_DI	转换指令	二进制补码双精度整型
NEG_I	NEG_I	转换指令	二进制补码整型
NEG_R	NEG_R	转换指令	实数取反
OPN	OPN	DB 调用指令	打开数据块
OS	OS	状态位指令	存储的溢出异常位
OV	OV	状态位指令	溢出异常位
P	P	位逻辑指令	RLO 正跳沿检测
POS	POS	位逻辑指令	地址正跳沿检测
R	R	位逻辑指令	复位输出
RET	RET	程序控制	返回
ROL_DW	ROL_DW	移位/循环	循环左移双字

德语助记符	英语助记符	程序元素分类	描述
ROUND	ROUND	转换指令	取整为双精度整型
ROR_DW	ROR_DW	移位/循环	循环右移双字
RS	RS	位逻辑指令	复位置位触发器
S	S	位逻辑指令	置位输出
SA	SF	定时器	启动断开延时定时器
SAVE	SAVE	位逻辑指令	将 RLO 存入 BR 存储区
S_AVERZ	S_OFFDT	定时器	设置断开延时定时器参数并启动
SE	SD	定时器	设置接通延时定时器参数并启动
S_EVERZ	S_ODT	定时器	设置接通延时定时器参数并启动
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	右移双精度整型
SHR_DW	SHR_DW	移位/循环	右移双字
SHR_I	SHR_I	移位/循环	右移整型
SHR_W	SHR_W	移位/循环	右移字
SI	SP	定时器	启动脉冲定时器
S_IMPULS	S_PULSE	定时器	设置脉冲定时器参数并启动
SIN	SIN	浮点算术运算指令	计算以浮点数表示的角的三角函数
SQR	SQR	浮点算术运算指令	浮点数平方(SQR)运算
SQRT	SQRT	浮点算术运算指令	计算浮点数的平方根(SQRT)
SR	SR	位逻辑指令	置位复位触发器
SS	SS	定时器	启动掉电保护接通延时定时器
S_SEVERZ	S_ODTS	定时器	设置掉电保护接通延时定时器参数并启动
SUB_DI	SUB_DI	整数算术运算指令	减双精度整型
SUB_I	SUB_I	整数算术运算指令	减整型
SUB_R	SUB_R	浮点算术运算指令	减实型
SV	SE	定时器	启动延时脉冲定时器
S_VIMP	S_PEXT	定时器	设置延时脉冲定时器参数并启动
SZ	SC	计数器	设置计数器值
TAN	TAN	浮点算术运算指令	计算以浮点数表示的角的三角函数
TRUNC	TRUNC	转换指令	截尾取整数部分
UO	UO	状态位	例外位无序
WAND_DW	WAND_DW	字逻辑指令	双字与运算(字)
WAND_W	WAND_W	字逻辑指令	单字与运算(字)
WOR_DW	WOR_DW	字逻辑指令	双字或运算(字)
WOR_W	WOR_W	字逻辑指令	单字或运算(字)
WXOR_DW	WXOR_DW	字逻辑指令	双字异或运算(字)
WXOR_W	WXOR_W	字逻辑指令	单字异或运算(字)
XOR	XOR	位逻辑指令	“异或”逻辑操作

德语助记符	英语助记符	程序元素分类	描述
ZAEHLER	S_CUD	计数器	分配参数和递增/递减计数
ZR	CD	计数器	值减计数器
Z_RUECK	S_CD	计数器	分配参数和递减计数
ZV	CU	计数器	值加计数器
Z_VORW	S_CU	计数器	分配参数和递增计数

A.2 根据英语助记符(国际)排序的 FBD 指令

英语助记符	德语助记符	程序元素分类	描述
&	&	位逻辑指令	“与”逻辑操作
>=1	>=1	位逻辑指令	“或”逻辑操作
=	=	位逻辑指令	赋值
#	#	位逻辑指令	中间输出
---|	---|	位逻辑指令	插入数字输入
---o|	---o|	位逻辑指令	数字输入取反
==0	==0	状态位	结果位
>0	>0	状态位	结果位
>=0	>=0	状态位	结果位
<0	<0	状态位	结果位
<=0	<=0	状态位	结果位
<>0	<>0	状态位	结果位
ABS	ABS	浮点算术运算指令	浮点数的绝对值运算
ACOS	ACOS	浮点算术运算指令	计算以浮点数表示的角的三角函数
ADD_DI	ADD_DI	整数算术运算指令	加双精度整型
ADD_I	ADD_I	整数算术运算指令	加双精度整型
ADD_R	ADD_R	浮点算术运算指令	加实型
ASIN	ASIN	浮点算术运算指令	计算以浮点数表示的角的三角函数
ATAN	ATAN	浮点算术运算指令	计算以浮点数表示的角的三角函数
BCD_DI	BCD_DI	转换指令	BCD 码转换为双精度整型
BCD_I	BCD_I	转换指令	BCD 码转换为整型
BR	BIE	状态位指令	BR 存取区异常位
CALL	CALL	程序控制指令	调用无参数的 FC/SFC
CALL_FB	CALL_FB	程序控制指令	CALL_FB (以框方式调用 FB)
CALL_FC	CALL_FC	程序控制指令	CALL_FC (以框方式调用 FC)
CALL_SFB	CALL_SFB	程序控制指令	CALL_SFB (以框方式调用系统 FB)
CALL_SFC	CALL_SFC	程序控制指令	CALL_SFC (以框方式调用系统 FC)
CD	ZR	计数器指令	值减计数器
CEIL	CEIL	转换指令	上限
CMP ?D	CMP ?D	比较指令	比较双精度整数
CMP ?I	CMP ?I	比较指令	整数比较
CMP ?R	CMP ?R	比较指令	比较实数
COS	COS	浮点算术运算指令	计算以浮点数表示的角的三角函数
CU	ZV	计数器指令	值加计数器
DI_BCD	DI_BCD	转换指令	双精度整型转换为 BCD 码
DI_R	DI_R	转换指令	双精度整型转换为实型
DIV_DI	DIV_DI	整数算术运算指令	除双精度整型
DIV_I	DIV_I	整数算术运算指令	除整型
DIV_R	DIV_R	浮点算术运算指令	除实型

英语助记符	德语助记符	程序元素分类	描述
EXP	EXP	浮点算术运算指令	计算浮点数的指数值
FLOOR	FLOOR	转换指令	基数
I_BCD	I_BCD	转换指令	整型转换为 BCD 码
I_DI	I_DI	转换指令	整型转换为双精度整型
INV_I	INV_I	转换指令	对整型数求反码
INV_DI	INV_DI	转换指令	二进制反码双精度整型
JMP	JMP	跳转	块中无条件跳转
JMP	JMP	跳转	块中有条件跳转
JMPN	JMPN	跳转	若非则跳转
LABEL	LABEL	跳转	跳转标签
LN	LN	浮点算术运算指令	计算浮点数的自然对数
MCR>	MCR>	程序控制	主控继电器开/关
MCR<	MCR<	程序控制	主控继电器开/关
MCRA	MCRA	程序控制	主控继电器激活/去活
MCRD	MCRD	程序控制	主控继电器激活/去活
MOD_DI	MOD_DI	整数算术运算指令	返回分数双精度整型
MOVE	MOVE	传送	赋值
MUL_DI	MUL_DI	整数算术运算指令	乘双精度整型
MUL_I	MUL_I	整数算术运算指令	乘整型
MUL_R	MUL_R	浮点算术运算指令	乘实型
N	N	位逻辑指令	RLO 负跳沿检测
NEG	NEG	位逻辑指令	地址负跳沿检测
NEG_DI	NEG_DI	转换指令	二进制补码双精度整型
NEG_I	NEG_I	转换指令	二进制补码整型
NEG_R	NEG_R	转换指令	实数取反
OPN	OPN	DB 调用	打开数据块
OS	OS	状态位	存储的溢出异常位
OV	OV	状态位	溢出异常位
P	P	位逻辑指令	RLO 正跳沿检测
POS	POS	位逻辑指令	地址正跳沿检测
R	R	位逻辑指令	复位输出
RET	RET	程序 控制	返回
ROL_DW	ROL_DW	移位/循环指令	循环左移双字
ROUND	ROUND	转换指令	取整为双精度整型
ROR_DW	ROR_DW	移位/循环指令	循环右移双字
RS	RS	位逻辑指令	复位置位触发器
S	S	位逻辑指令	置位输出
SAVE	SAVE	位逻辑指令	将 RLO 存入 BR 存储区

英语助记符	德语助记符	程序元素分类	描述
SC	SZ	计数器指令	设置计数器值
S_CD	Z_RUECK	计数器指令	分配参数和递减计数
S_CU	Z_VORW	计数器指令	分配参数和递增计数
S_CUD	ZAEHLER	计数器	分配参数和递增/递减计数
SD	SE	定时器指令	启动接通延时定时器
SE	SV	定时器指令	启动延时脉冲定时器
SF	SA	定时器指令	启动断开延时定时器
SHL_DW	SHL_DW	移位/循环指令	双字左移
SHL_W	SHL_W	移位/循环指令	左移字
SHR_DI	SHR_DI	移位/循环指令	右移双精度整型
SHR_DW	SHR_DW	移位/循环指令	右移双字
SHR_I	SHR_I	移位/循环指令	右移整型
SHR_W	SHR_W	移位/循环指令	右移字
SIN	SIN	浮点算术运算指令	计算以浮点数表示的角的三角函数
S_ODT	S_EVERZ	定时器指令	设置接通延时定时器参数并启动
S_ODTS	S_SEVERZ	定时器指令	设置掉电保护接通延时定时器参数并启动
S_OFFDT	S_AVERZ	定时器指令	设置断开延时定时器参数并启动
SP	SI	定时器指令	启动脉冲定时器
S_PEXT	S_VIMP	定时器指令	设置延时脉冲定时器参数并启动
S_PULSE	S_IMPULS	定时器指令	设置脉冲定时器参数并启动
SQR	SQR	浮点算术运算指令	浮点数平方(SQR)运算
SQRT	SQRT	浮点算术运算指令	计算浮点数的平方根(SQRT)
SR	SR	位逻辑指令	置位复位触发器
SS	SS	定时器指令	启动掉电保护接通延时定时器
SUB_DI	SUB_DI	整数算术运算指令	减双精度整型
SUB_I	SUB_I	整数算术运算指令	减整型
SUB_R	SUB_R	浮点算术运算指令	减实型
TAN	TAN	浮点算术运算指令	计算以浮点数表示的角的三角函数
TRUNC	TRUNC	转换指令	截尾取整数部分
UO	UO	状态位指令	例外位无序
WAND_DW	WAND_DW	字逻辑指令	双字与运算(字)
WAND_W	WAND_W	字逻辑指令	单字与运算(字)
WOR_DW	WOR_DW	字逻辑指令	双字或运算(字)
WOR_W	WOR_W	字逻辑指令	单字或运算(字)
WXOR_DW	WXOR_DW	字逻辑指令	双字异或运算(字)
WXOR_W	WXOR_W	字逻辑指令	单字异或运算(字)
XOR	XOR	位逻辑指令	“异或”逻辑操作

B 编程实例

B.1 编程举例概述

实际应用

每一个语句表指令都触发一个特定的操作。将这些指令组合到程序中后，可以完成很多种类的自动化任务。本章提供下列实际应用语句表指令的实例：

- 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
- 监视存储空间使用计数器和比较指令
- 使用整数算术运算指令解题
- 设置加热烘炉的时长

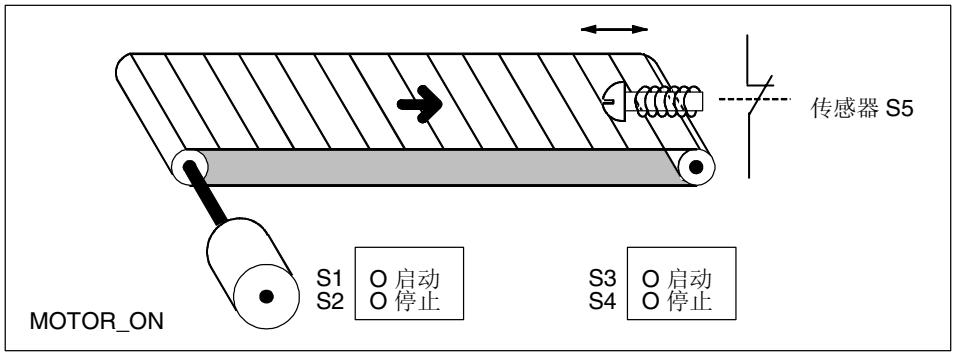
使用的指令

助记符	程序元素分类	描述
WAND_W	字逻辑指令	(字)单字与运算
WOR_W	字逻辑指令	(字)单字或运算
S_CD	计数器指令	值减计数器
S_CU	计数器指令	值加计数器
R	位逻辑指令	复位输出
S	位逻辑指令	设置输出
P	位逻辑指令	RLO 正跳沿检测
ADD_I	浮点算术运算指令	加整型
DIV_I	浮点指令	除整型
MUL_I	浮点指令	乘整型
CMP >=I、CMP <=I	比较	整数比较
&	位逻辑指令	与运算
>=1	位逻辑指令	或运算
=	位逻辑指令	赋值
JMPN	跳转指令	若非则跳转
RET	程序控制指令	返回
MOVE	传送指令	赋值
SE	定时器指令	延时脉冲定时器

B.2 举例：位逻辑指令

举例 1：控制传送带

下图显示可以电气启动的传送带。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。在传送带尾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末端时传感器 S5 会使传送带停止。



绝对编程和符号编程

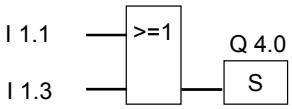
您可以使用代表传送机系统各种组件的**绝对值**或**符号**编写程序，借以激活传送带系统的方向显示。

您需要制定一个符号表，使您所选择的符号与绝对值建立关联(参见 STEP 7 在线帮助)。

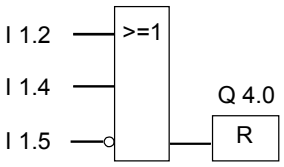
系统组件	绝对地址	符号	符号表
启动按钮开关	I 1.1	S1	I 1.1 S1
停止按钮开关	I 1.2	S2	I 1.2 S2
启动按钮开关	I 1.3	S3	I 1.3 S3
停止按钮开关	I 1.4	S4	I 1.4 S4
传感器	I 1.5	S5	I 1.5 S5
电机	Q 4.0	MOTOR_ON	Q 4.0 MOTOR_ON

传送带控制功能块图

程序段 1：按任一个启动开关接通电机。

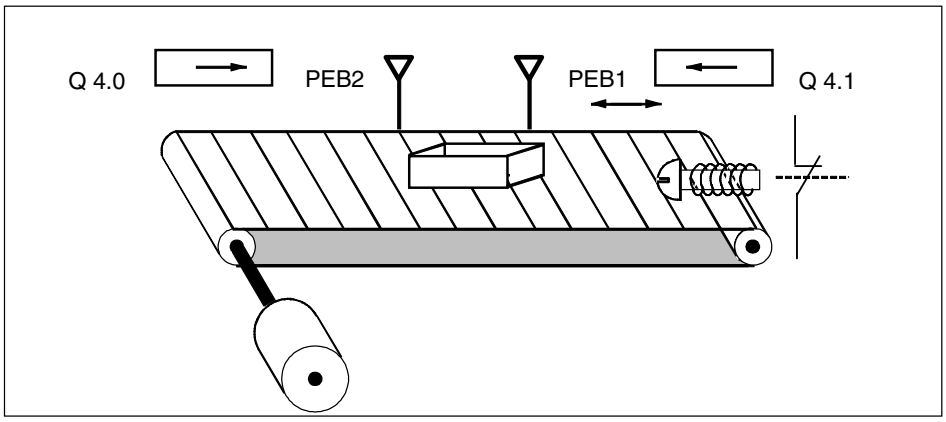


程序段 2：按任一停止开关或打开传送带末端的传感器的常闭触点，可以关闭电机。



举例 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，它们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。



绝对编程和符号编程

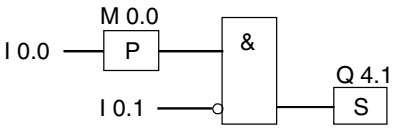
可以使用代表传送系统各种组件的**绝对值**或**符号**编写激活传送带系统方向显示的程序。

您需要制定一个符号表，使您所选择的符号与绝对值建立关联(参见 STEP 7 在线帮助)。

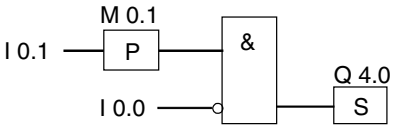
系统组件	绝对 地址	符号	符号表
光电屏障 1	I 0.0	PEB1	I 0.0 PEB1
光电屏障 2	I 0.1	PEB2	I 0.1 PEB2
右向移动显示	Q 4.0	RIGHT	Q 4.0 RIGHT
左向移动显示	Q 4.1	LEFT	Q 4.1 LEFT
脉冲存储位 1	M 0.0	PMB1	M 0.0 PMB1
脉冲存储位 2	M 0.1	PMB2	M 0.1 PMB2

用于检测传送带方向的功能块图

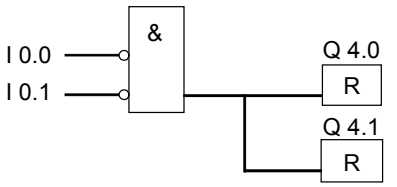
程序段 1：如果输入 I 0.0 处信号状态从 0 变到 1 (上升沿)，同时输入 I 0.1 处信号状态为 0，则传送带上的包裹会正在向左移动。



程序段 2：如果输入 I 0.1 处信号状态从 0 变到 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正在向右移动。



程序段 3：如果光电屏障之一被中断，则表明屏障之间有包裹。方向指针关闭。



B.3 举例：定时器指令

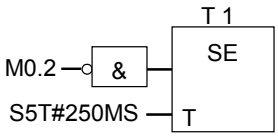
时钟脉冲发生器

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烁继电器。时钟脉冲发生器在控制指示灯闪烁的信号系统中是公用的。

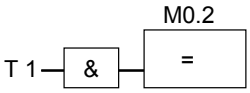
使用 S7-300 时，可以通过在专用组织块中使用时间驱动处理来实现时钟脉冲发生器功能。然而，下列 FBD 程序中显示的实例说明了使用定时器功能生成时钟脉冲。此示例程序显示如何使用定时器实现自由设定时钟脉冲发生器。

产生时钟脉冲(脉冲占空比 1:1)的功能块图

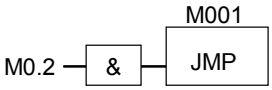
程序段 1：如果定时器 T1 的信号状态为 0，将时间值 250 毫秒装入 T1，并将 T1 作为延时脉冲定时器启动。



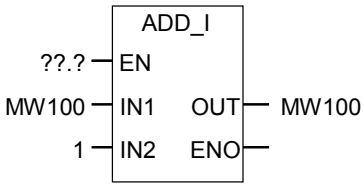
程序段 2：该定时器的状态临时保存在一个辅助的内存标记中。



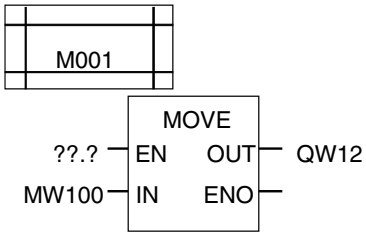
程序段 3：如果定时器 T1 的信号状态为 1，则跳转至跳转标签 M001。



程序段 4：定时器 T1 到时后，存储字节 100 增加 1。

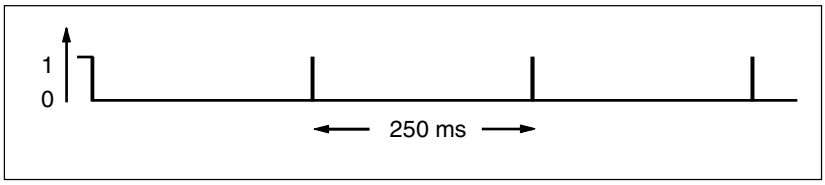


程序段 5：**MOVE** 指令允许在输出 Q12.0 到 Q13.7 处输出不同的时钟频率。



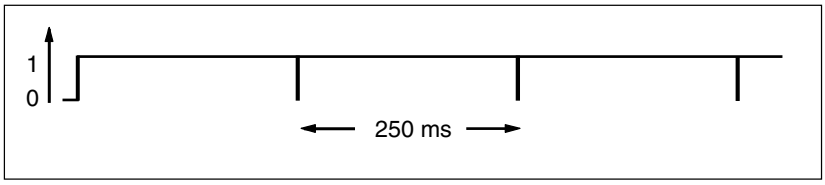
信号检查

在以下时钟脉冲实例中，对于与逻辑操作取反的输入参数(M0.2)来说，定时器 T1 的信号检查将产生下列逻辑运算结果(RLO)：



只要时间一结束，定时器就将重新启动。因此，该信号只是很短暂地产生一个信号状态 1。

取反(反向) RLO：



每隔 250msRLO 位都为 0 一次。跳转被忽略，而存储字节 MW100 的内容增加 1。

获得专用频率

通过存储字节 MB101 和 MB100 的每个位可以获得下列频率：

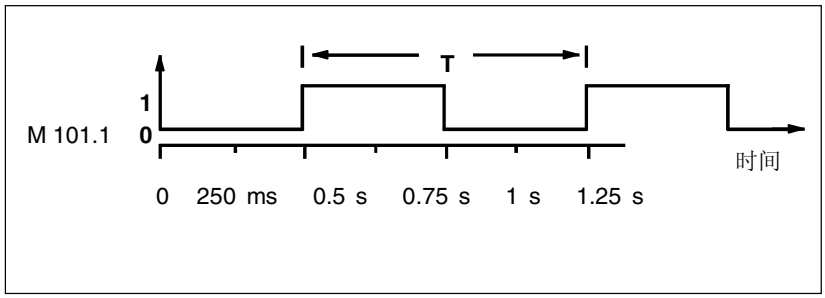
MB101/MB100 的位	频率**(**赫兹**)**	持续时间
M 101.0	2.0	0.5 秒 (250 毫秒开/250 毫秒关)
M 101.1	1.0	1 秒 (0.5 秒开/0.5 秒关)
M 101.2	0.5	2 秒 (1 秒开/1 秒关)
M 101.3	0.25	4 秒 (2 秒开/2 秒关)
M 101.4	0.125	8 秒 (4 秒开/4 秒关)
M 101.5	0.0625	16 秒 (8 秒开/8 秒关)
M 101.6	0.03125	32 秒 (16 秒开/16 秒关)
M 101.7	0.015625	64 秒 (32 秒开/32 秒关)
M 100.0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

存储字节 **MB 101** 的各位信号状态

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 位 1　(M 101.1)的信号状态

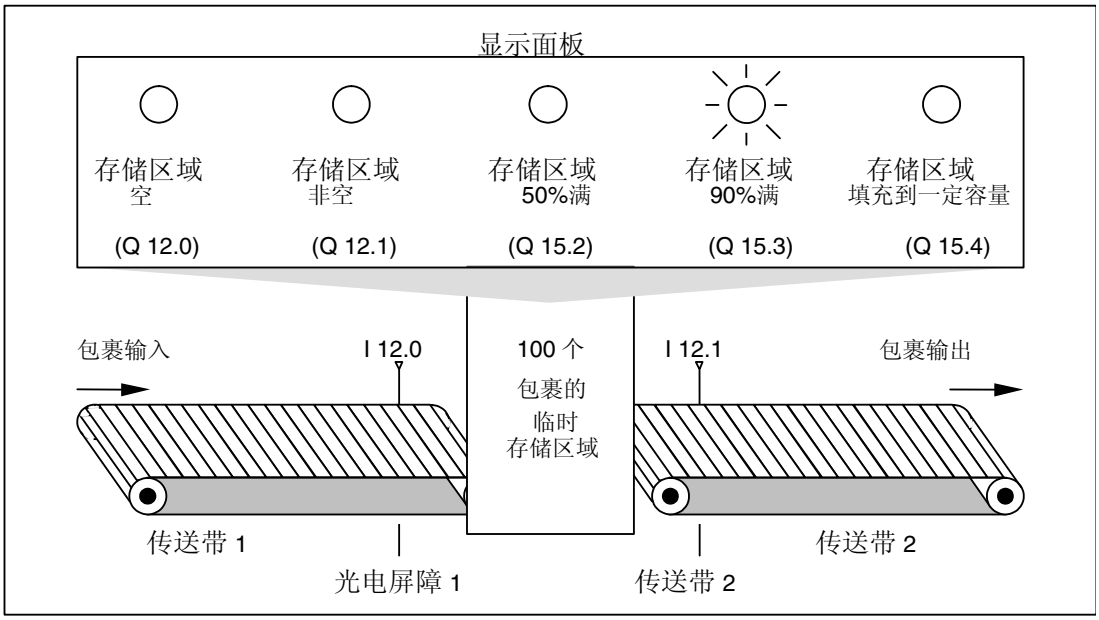
频率 = 1/T = 1/1 秒 = 1 赫兹



B.4 举例：计数器和比较指令

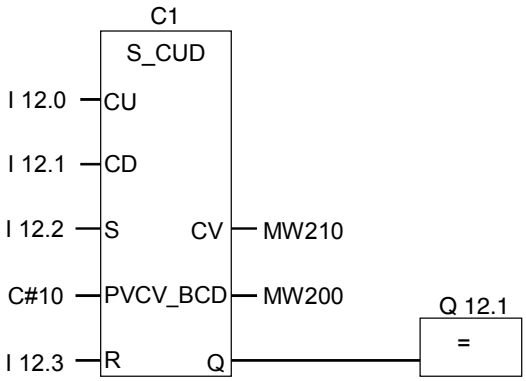
带计数器和比较器的存储区

下图显示带有两个传送带并且两个带之间有一个临时存储区的系统。传送带 1 将包裹传送到存储区。传送带 1 的末端靠近存储区处有一个光电屏障，它确定已向存储区传送的包裹的数量。传送带 2 将包裹从临时存储区传输到装载货场，卡车从此处取走包裹发送给用户。传送带 2 的末端靠近存储区处有一个光电屏障，它确定离开存储区进入装载码头的包裹的数量。带有 5 个指示灯的显示面板指示临时存储区的占用程度。

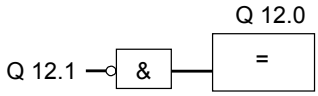


启动显示面板上的指示灯的功能块图

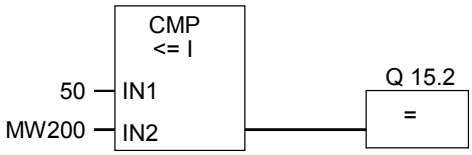
程序段 1：计数器 C1 对输入端 CU 处从“0”到“1”的每次信号切换都进行正计数，而对输入端 CD 处从“0”到“1”的每次信号切换都进行倒计数。对于输入端 S 处从“0”到“1”的信号切换，计数器值被设置为值 PV。输入端 R 处从“0”到“1”的信号切换将计数器值复位为“0”。MW200 包含 C1 的当前计数器值。Q12.1 指示“存储区不空”。



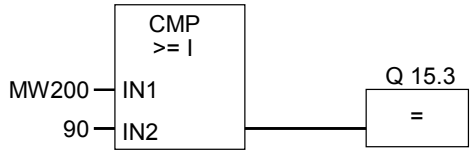
程序段 2：Q12.0 表明“存储区为空”。



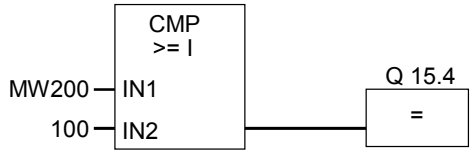
程序段 3：如果 50 小于等于计数器值(换句话说，如果当前计数器值大于等于 50)，则表示“存储区 50%满”的指示灯变亮。



程序段 4：如果计数器值大于等于 90，“存储区满 90%”指示灯变亮。



程序段 5：如果计数器值大于或等于 100，则表示“存储区满”的指示灯变亮。



B.5 举例：整数算术运算指令

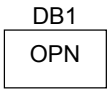
解决数学问题

此示例程序显示如何使用三个整数算术运算指令产生与下列等式相同的结果：

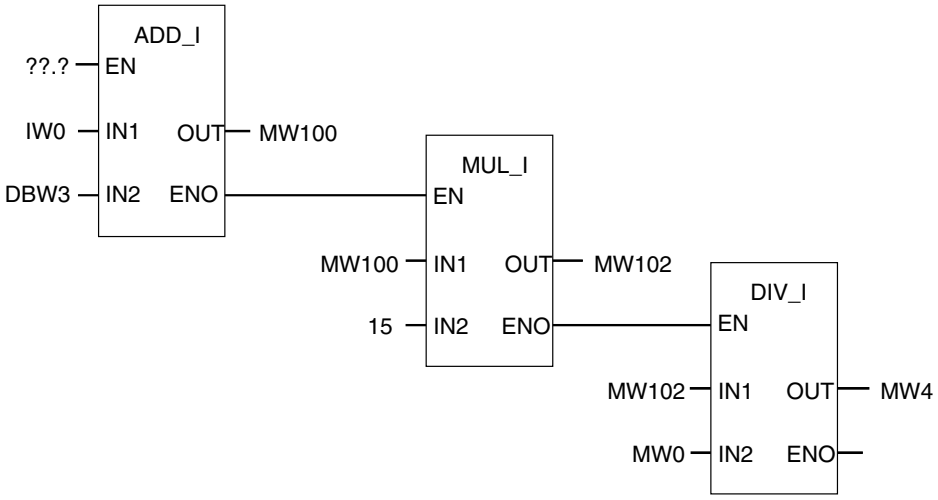
MW4 = ((IW0 + DBW3) x 15) / MW0

功能块图

程序段 1：打开数据块 DB1。



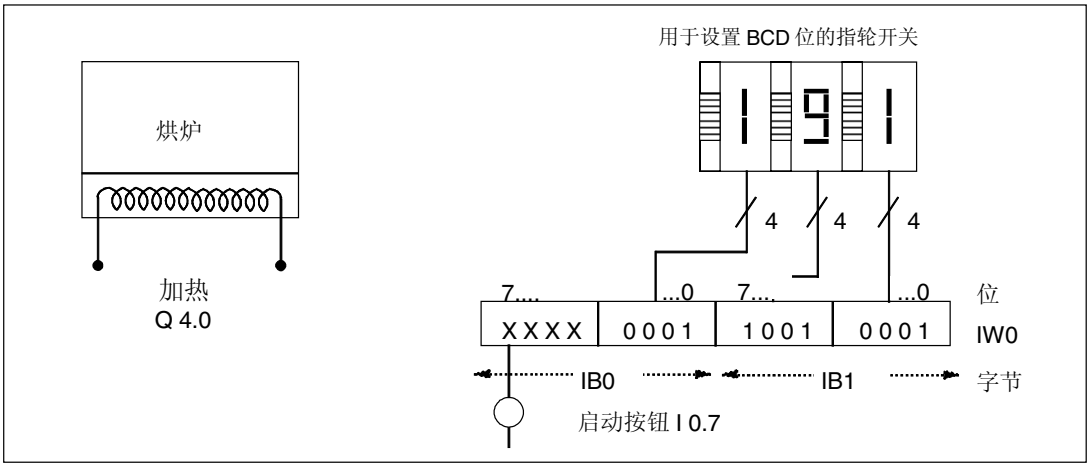
程序段 2：输入字 IW0 与共享数据字 DBW3 相加(必须定义和打开数据块)，然后总和被载入存储字节 MW100。然后，MW100 乘以 15，结果存储到存储字节 MW102 中。MW102 除以 MW0，结果存储到 MW4 中。



B.6 举例：字逻辑指令

加热烘炉

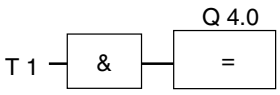
烘炉操作员按启动按钮开始烘炉加热。操作员可以使用图中所示的拨码开关设置加热时长。操作员设置的值以二进制编码十进制(BCD)格式用秒为单位显示。



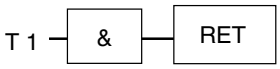
系统组件	绝对 地址
启动按钮	I 0.7
个位拨码开关	I 1.0 到 I 1.3
十位拨码开关	I 1.4 到 I 1.7
百位拨码开关	I 0.0 到 I 0.3
加热开始	Q 4.0

功能块图

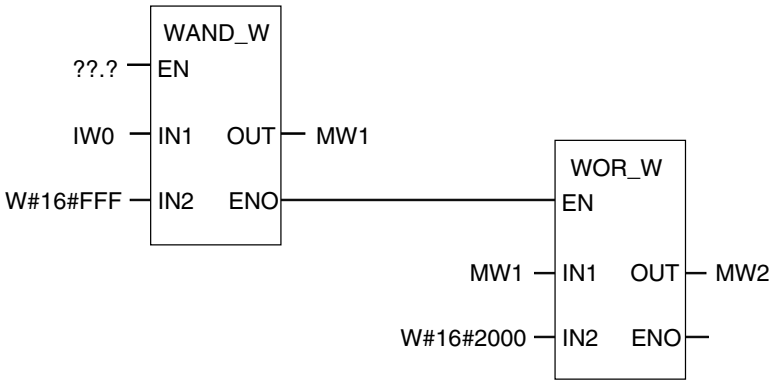
程序段 1：如果定时器正在运行，则打开加热装置。



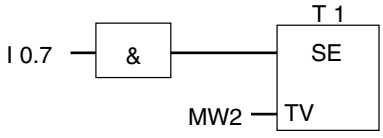
程序段 2：如果定时器正在计时，**返回**指令结束此处的处理。



程序段 3：屏蔽输入位 I 0.4 到 I 0.7 (即，将它们复位为 0)。不使用拨码开关输入的这些位。16 位拨码开关输入根据**(字)单字与运算**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)单字或运算**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 T1 按延时脉冲定时器方式启动，并作为预设值存储字节 MW2 装载(来自上述逻辑)。



C 使用功能块图

C.1 EN/ENO 机制

FBD/LAD 框的使能(EN)和使能输出(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。

对于依赖先前已存在的和后续的逻辑运算的 EN/ENO 机制，会在框中实际指令周围生成附加的 STL 指令。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

如果要编写希望在 FBD 或 LAD 中调用的程序块，必须确保在退出块时设置 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。而应当使用一个临时变量，以在其中保存所发生的任何错误。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
```

确保在任何情况下均能处理此程序段，即不可在块中使用 BEC 和跳过此程序段。

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 参数传送

块参数以值的形式传送。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，请确保同时还编写了声明为 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

B

C

D

E

F

H

J

K

L

P

Q

R

S

T

W

X

Y

Z