SIEMENS

SIMATIC

S7-300 和 S7-400 编程的功能块图 (FBD)

参考手册

05/2017
A5E41525025-AA

法律资讯

警告提示系统

为了您的人身安全以及避免财产损失，必须注意本手册中的提示。人身安全的提示用一个警告三角表示，仅与财产损失有关的提示不带警告三角。警告提示根据危险等级由高到低如下表示。

⚠危险

表示如果不采取适当的预防措施，**可能**导致死亡或严重的人身伤害。

⚠警告

表示如果不采取适当的预防措施，**可能**导致死亡或严重的人身伤害。

⚠注意

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。

注意

表示如果不采取适当的预防措施，可能造成财产损失。

当出现多个危险等级的情况下，每次总是使用最高等级的警告提示。如果在某个警告提示中带有警告可能导致人身伤害的警告三角，则可能在该警告提示中另外还附带有可能导致财产损失的警告。

合格的专业人员

仅允许安装和驱动与本文件相关的附属设备或系统。设备或系统的调试和运行仅允许由合格的**专业人员**进行。本文件安全技术提示中的合格专业人员是指根据安全技术标准具有从事进行设备、系统和电路的运行，接地和标识资格的人员。

按规定使用 Siemens 产品

注意下列各项：

⚠警告

Siemens 产品只允许用于目录和相关技术文件中规定的使用情况。如果要使用其他公司的产品和组件，必须得到 Siemens 推荐和允许。正确的运输、储存、组装、装配、安装、调试、操作和维护是产品安全、正常运行的前提。必须遵守允许的环境条件。必须注意相关文件中的提示。

商标

所有带有标记符号 ® 的都2017的注册商标。标签中的其他符号可能是一些其他商标，这是出于保护所有者权利的目地由第三方使用而特别标示的。

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信息，并在后续版本中进行必要的更正。

Siemens AG
Division Digital Factory
Postfach 48 48
90026 NÜRNBERG
GERMANY

A5E41525025-AA
℗ 03/2017 恕不提前通知

前言

用途

本手册是在功能块图(FBD)编程语言中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。

所需要的基础知识

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 XP、MS Windows Server 2003 或 MS Windows 7 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 5.6 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.RTF 中的标准表。

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(例如，打开的对话框或激活的窗口)的信息。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"FBD 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

可以在下列网址上找到单个 SIAMTIC 产品和系统的技术文档指南：

http://www.siemens.com/simatictechdokuportal

可以在下列网址上获得在线目录和订货系统：

http://mall.automation.siemens.com/

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 90026)的培训总部，以获取详细信息。

Internet：http://sitrain.automation.siemens.com/sitrainworld/

技术支持

您可访问"技术支持"来了解所有的工业自动化和驱动技术产品

- 通过网站请求支持
  http://www.siemens.com/automation/support-request

关于技术支持的更多信息请参见 Internet 网页：

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

Internet 服务和支持

除文档以外，还在 Internet 上在线提供了专业技术信息，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。
- 您当地的关于工业自动化和驱动技术的销售代表。
- 关于现场服务、维修、备件和查阅等信息。

安全提示：

西门子在工业安全功能方面提供产品和解决方案，旨在支持工厂、系统、机器和网络的安全运行。

为保护工厂、系统、机器和网络免受网络攻击威胁，必须实施并不断保持全方位的先进工业安全理念。西门子产品和解决方案仅仅是其中的一个方面。

客户应负责保护其工厂、系统、机器和网络免受未经授权的访问。系统、机器和组件仅可连接企业网络，且只能在必要时且相应安全措施(例如，使用防火墙和网络分段)到位的情况下连接互联网。

此外，还应考虑西门子在相应安全措施方面的指导。有关工业安全的更多信息，请访问

http://www.siemens.com/industrialsecurity。

西门子产品和解决方案经过不断发展，安全性日趋完善。西门子强烈建议您尽快应用产品更新，并始终使用最新的产品版本。如果使用不再受支持的产品版本，并且未能应用最新的更新，则会增加客户受到网络攻击的危险。

要时刻了解产品更新，请订阅西门子工业安全 RSS 信息源

http://www.siemens.com/industrialsecurity。

目录

1 位逻辑指令

1.1 位逻辑指令概述

描述

位逻辑指令使用两个数字 1 和 0。这两个数字构成二进制系统的基础。这两个数字 1 和 0 称为二进制数字或位。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"。

位逻辑指令解释信号状态 1 和 0，并根据布尔逻辑将其组合。这些组合产生称为"逻辑运算结果"(RLO)的结果 1 或 0。

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：置位输出
- 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：

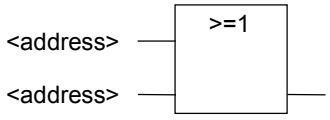
- N ：RLO 负跳沿检测
- P ：正 RLO 边沿检测
- NEG：地址下降沿检测
- POS: 地址上升沿检测

其余指令直接以下列方式影响 RLO：

- 插入二进制输入
- 二进制取反输入
- SAVE ：将 RLO 存入 BR 存储区

1.2 >=1 ："或"逻辑运算

符号



参数	数据类型	存储器区	描述
<地址>	BOOL	I、Q、M、T、C、D、L	地址表示要检查信号状态的位。

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。

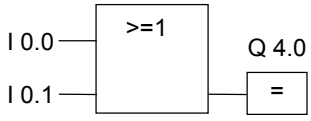
如果"**或**"运算指令是系列逻辑操作的第一个指令，则它会将其信号状态的检查结果存入 RLO 位。

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	CC1	CC0	OV	OS	OR	STA	RLO	FC
写	-	-	-	-	-	X	X	X	1

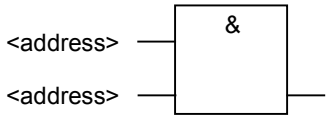
实例



如果输入端 I0.0"或"输入端 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	存储器区	描述
<地址>	BOOL	I、Q、M、T、C、D、L	地址表示要检查信号状态的位。

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。

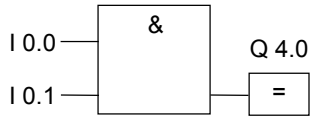
如果"**与**"运算指令是系列逻辑操作的第一个指令，则它会将其信号状态的检查结果存入 RLO 位。

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	CC1	CC0	OV	OS	OR	STA	RLO	FC
写	-	-	-	-	-	X	X	X	1

实例



如果输入端 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

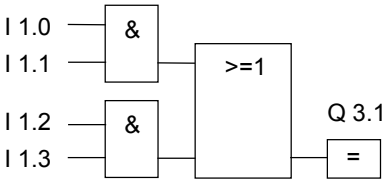
使用**先"与"后"或"**指令，可以根据"或"运算真值表检查信号状态的结果。

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	CC1	CC0	OV	OS	OR	STA	RLO	FC
写	-	-	-	-	-	X	X	X	1

实例



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

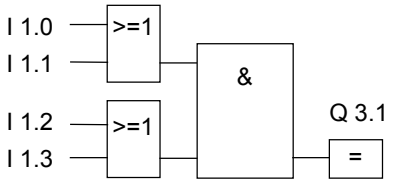
使用**先"或"后"与"**指令，可以根据"与"真值表检查信号状态的结果。

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	CC1	CC0	OV	OS	OR	STA	RLO	FC
写	-	-	-	-	-	X	X	X	1

实例

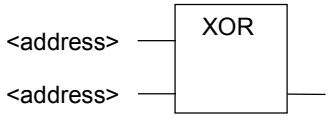


如果两个"或"逻辑操作都满足，输出 Q3.1 的信号状态为 1。

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"逻辑运算

符号



参数	数据类型	存储器区	描述
<地址>	BOOL	I、Q、M、T、C、D、L	地址表示要检查信号状态的位。

描述

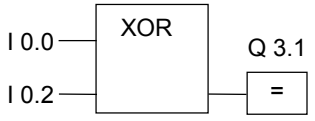
使用"**异或**"运算指令，可以根据"异或"运算真值表检查信号状态的结果。

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 Exclusive OR 函数。这样，如果有奇数个被检查地址状态为"1"，则逻辑运算的最终结果为"1"。

状态字

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

实例



如果输入端 I0.0"或"输入端 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 插入二进制输入

符号

<地址>

参数	数据类型	存储器区	描述
<地址>	BOOL	I、Q、M、T、C、D、L	地址表示要检查信号状态的位。

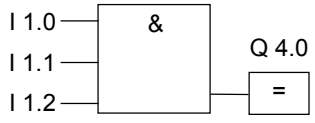
描述

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	CC1	CC0	OV	OS	OR	STA	RLO	FC
写	-	-	-	-	-	-	1	X	-

实例



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 二进制取反输入

符号

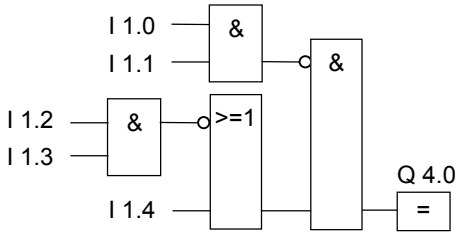
描述

数字输入取反指令对 RLO 取反。

状态字

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

实例

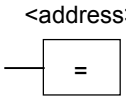


如果满足以下条件，则输出 Q4.0 为 1：

- 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	存储器区	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 功能的更详细信息，请参考 MCR 开/关。

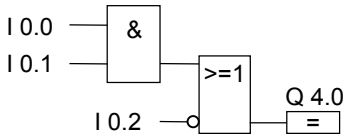
只能将**赋值**框置于系列逻辑操作的右端。然而，可以使用多个**赋值**框。

可以使用**取反输入**指令创建取反的赋值。

状态字

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

实例

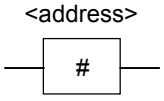


如果满足以下条件，输出 Q4.0 的信号状态为 1：

- 输入端 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	存储器区	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。

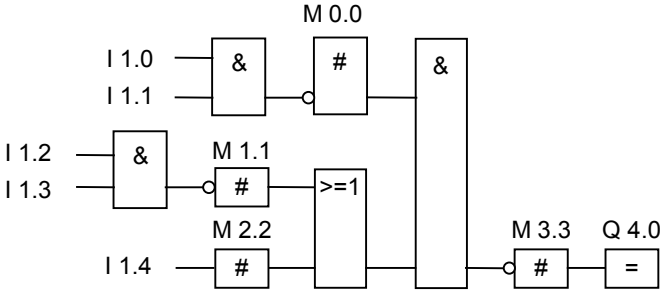
可以通过取反"中间输出"的输入来生成取反的"中间输出"。

状态字

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

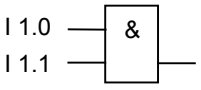
1.9 # ：中间输出

实例

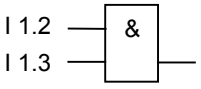


"中间输出"会缓存下列逻辑操作的结果：

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：

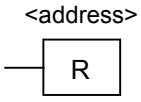


M2.2 保存 I1.4 的 RLO

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	存储器区	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

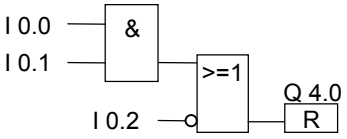
只有在 RLO 为 1 时，才执行**复位输出**指令。如果 RLO 为 1，此指令将指定地址复位为 0。如果 RLO 为 0，此指令不影响指定地址，该地址中的内容将保持不变。

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

状态字

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

实例



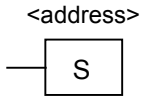
仅当满足下列条件时，输出 Q4.0 的信号状态才复位为 0：

- 输入端 I0.0"与"I0.1 的信号状态为 1
- 或者输入端 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：置位输出

符号



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

描述

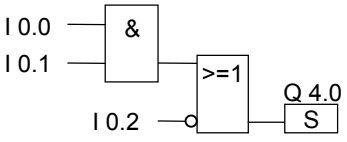
只有在 RLO 为 1 时，才执行**置位输出**指令。如果 RLO 为 1，此指令将指定地址置 1。如果 RLO 为 0，此指令不影响指定地址，该地址中的内容将保持不变。

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

状态字

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

实例



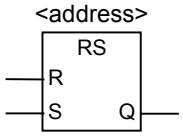
仅当满足下列条件时，才将输出 Q4.0 的信号状态置 1：

- 输入端 I0.0"与"I0.1 的信号状态为 1
- 或者输入端 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	存储器区	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。

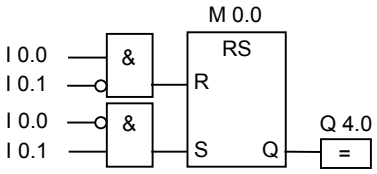
在输入端 R 的信号状态为 1，而输入端 S 的信号状态为 0 时，**复位置位触发器**被复位。如果输入端 R 为 0，而输入端 S 为 1，则此触发器被置位。如果两个输入的 RLO 均为 1，此触发器被置位。

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

状态字

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

实例

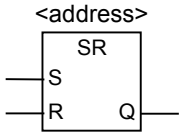


如果 I0.0 为 1 而 I0.1 为 0，则存储位 M0.0 被复位且输出 Q4.0 为 0。如果 I0.0 为 0 而 I0.1 为 1，则存储位 M0.0 被置位且输出 Q4.0 为 1。

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	存储器区	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。

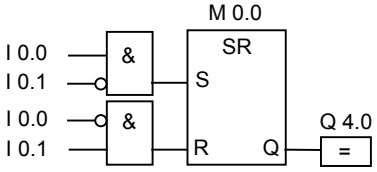
在输入端 S 的信号状态为 1，而输入端 R 的信号状态为 0 时，**置位复位触发器**被置位。如果输入端 S 为 0，而输入端 R 为 1，则触发器被复位。如果两个输入的 RLO 均为 1，则该触发器被复位。

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

状态字

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

实例

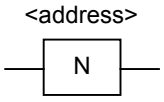


如果 I0.0 为 1 而 I0.1 为 0，则存储位 M0.0 被置位且 Q4.0 为 1。如果 I0.0 为 0 而 I0.1 为 1，则存储位 M0.0 被复位且 Q4.0 为 0。

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 Q 4.0 为 0。

1.14 N：RLO 负跳沿检测

符号



参数	数据类型	存储器区	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。

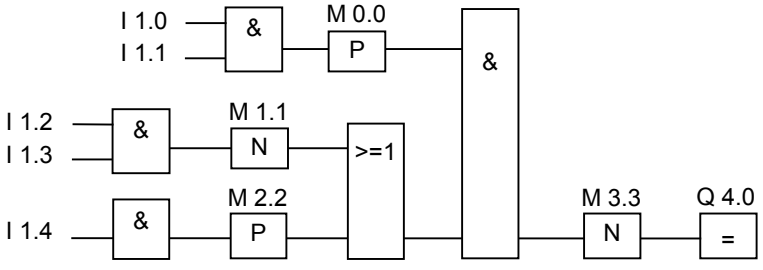
描述

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。指令执行前的 RLO 状态存储在地址中。

状态字

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

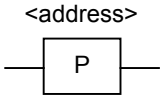
实例



沿存储位 M3.3 存储先前 RLO 的信号状态。

1.15 P：正 RLO 边沿检测

符号



参数	数据类型	存储器区	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。

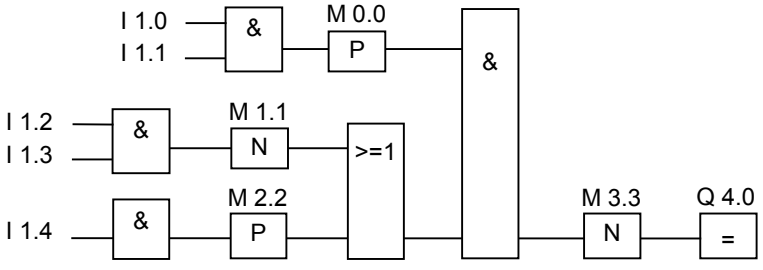
描述

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。指令执行前的 RLO 状态存储在地址中。

状态字

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

实例



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。

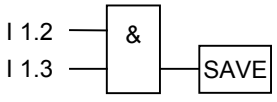
对于指令 **SAVE** (LAD、FBD、STL)，适用下列原则而不是在手册和在线帮助中建议的用途：
建议用户不要在使用 SAVE 后在同一块或从属块中校验 BR 位，因为这期间执行的指令中有许多会对 BR 位进行修改。建议用户在退出块前使用 SAVE 指令，因为 ENO 输出(= BR 位)届时已设置为 RLO 位的值，所以可以检查块中是否有错误。

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	CC1	CC0	OV	OS	OR	STA	RLO	FC
写	-	-	-	-	-	-	-	-	-

实例

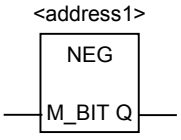


逻辑操作的结果(RLO)将写入 BR 位。

BR 二进制结果位(状态字，位 8)

1.17 NEG：地址下降沿检测

符号



参数	数据类型	存储器区	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.

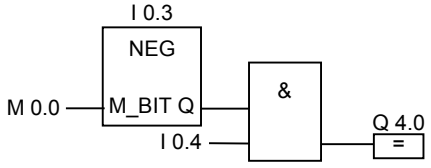
描述

地址下降沿检测指令比较 <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	CC1	CC0	OV	OS	OR	STA	RLO	FC
写	-	-	-	-	-	0	1	X	1

实例

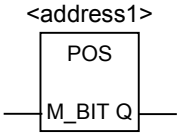


如果满足以下条件，则输出 Q4.0 为 1：

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

1.18 POS：地址上升沿检测

符号



参数	数据类型	存储器区	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.

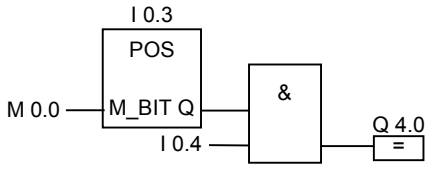
描述

地址上升沿检测指令比较 <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	CC1	CC0	OV	OS	OR	STA	RLO	FC
写	-	-	-	-	-	0	1	X	1

实例



如果满足以下条件，则输出 Q4.0 为 1：

输入端 I0.3 有上升沿

并且输入端 I0.4 的信号状态为 1。

1.18 POS：地址上升沿检测

2 比较指令

2.1 比较指令概述

描述

根据用户选择的比较类型比较 IN1 和 IN2：

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

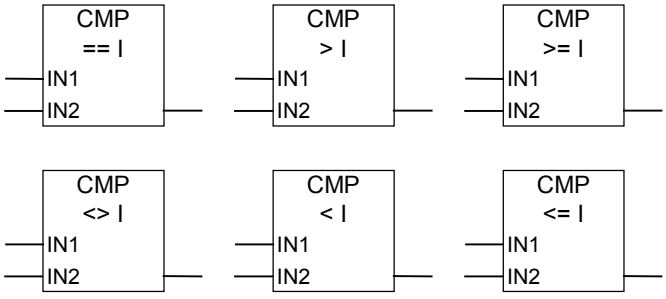
如果比较结果为 true，则此函数的 RLO 为"1"。否则，RLO 为 0。您不能对比较结果本身进行取反，但可通过使用相反的比较指令获得与取反同样的效果。

可使用下列比较指令：

- CMP ? I ：比较整数
- CMP ? D ：比较长整数
- CMP ? R：比较实数

2.2 CMP ? I ：比较整数

符号



参数	数据类型	存储器区	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

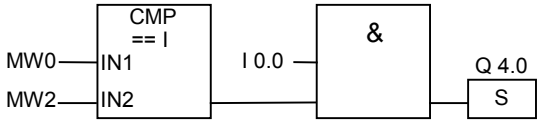
描述

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

实例

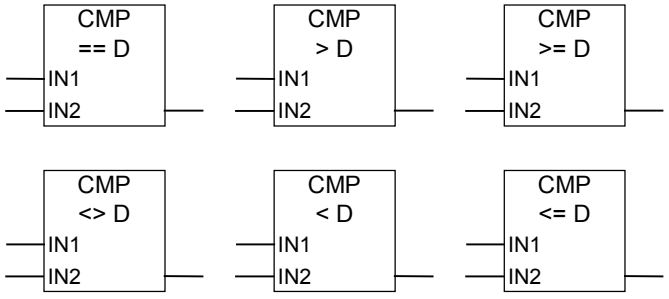


在满足以下条件时会将 Q 4.0 置 1：

- MD0 小于 MD4
- 并且输入端 I0.0 的信号状态为 1。

2.3 CMP ? D ：比较长整数

符号



参数	数据类型	存储器区	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

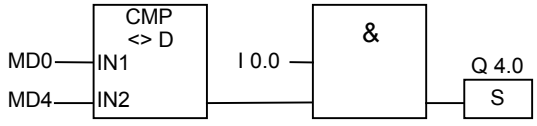
描述

比较长整数指令用于在 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	CC1	CC0	OV	OS	OR	STA	RLO	FC
写	-	X	X	0	-	0	X	X	1

实例

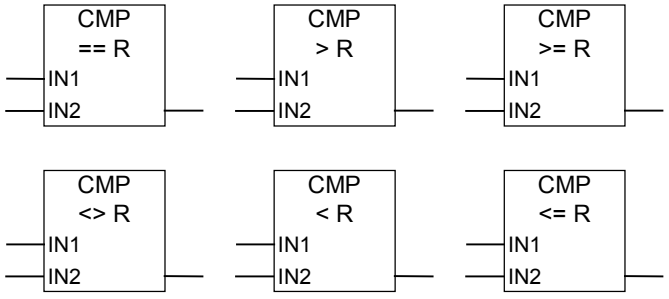


在满足以下条件时会将 Q 4.0 置 1：

- MD0 不等于 MD4
- 并且输入端 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	存储器区	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

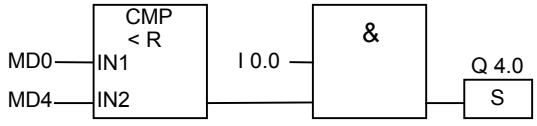
描述

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

实例



在满足以下条件时会将 Q 4.0 置 1：

- MD0 小于 MD4
- 并且输入端 I0.0 的信号状态为 1。

2.4 CMP ? R：比较实数

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为其它类型的数字：

- BCD_I : 将 BCD 码转换为整型
- I_BCD : 将整型转换为 BCD 码
- BCD_DI : BCD 码转换为长整型
- I_DI : 整型转换为长整型
- DI_BCD : 长整型转换为 BCD 码
- DI_R : 长整型转换为实数

可使用下列指令计算整数的补(反)码，或将浮点数的符号取反：

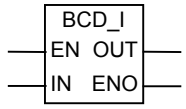
- INV_I : 二进制反码整数
- INV_DI : 二进制反码双精度整数
- NEG_I : 二进制补码整数
- NEG_DI : 二进制补码双精度整数
- NEG_R : 取反实数

可使用下列任何指令将累加器 1 中的 32 位 IEEE 浮点数转换为 32 位整型(长整型)。各个指令的取整方法有所不同：

- ROUND : 取整到最接近的双精度整数
- TRUNC : 截取双精度整数部分
- CEIL : 向上取整
- FLOOR : 向下取整

3.2 BCD_I：将 BCD 码转换为整型

符号



参数	数据类型	存储器区	描述
EN	BOOL	I、Q、M、D、L、T、C	启用输入
IN	WORD	I、Q、M、D、L 或常数	BCD 格式的数字
OUT	INT	I、Q、M、D、L	BCD 数字的整数值
ENO	BOOL	I、Q、M、D、L	启用输出

描述

将 BCD 码转换为整型指令读取输入参数 IN 中的内容，一个 BCD (二进制编码的十进制，± 999)格式的三位数字，然后将该数字转换为整数值。结果由输出参数 OUT 给出。

ENO 的信号状态始终与 EN 相同。

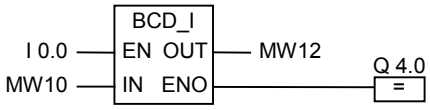
如果任意一个 BCD 格式的十进制数字位于无效范围 10 至 15 之间，则将在转换时发生 BCD 错误，导致以下反应：

- CPU 切换到 STOP 模式。"BCD 转换出错"被输入到诊断缓冲区中，事件 ID 号为 2521。
- 如果已编程 OB121，则将调用该块。

状态字

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

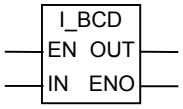
实例



如果 I0.0 的信号状态为 1，则执行转换。读取存储器字 MW10 中 BCD 格式的三位数字，将其转换为整数。结果存储在存储器字 MW12 中。如果转换已执行 ,则输出 Q4.0 的信号状态将为 1 (ENO = EN)。

3.3 I_BCD：将整型转换为 BCD 码

符号



参数	数据类型	存储器区	描述
EN	BOOL	I、Q、M、D、L、T、C	启用输入
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	启用输出

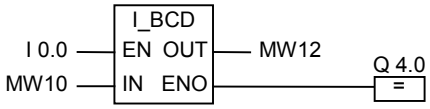
描述

将整型转换为 BCD 码指令以整型格式读取输入参数 IN 中的内容，然后将该值转换 BCD (二进制编码的十进制，± 999)格式的三位数字。结果由输出参数 OUT 给出。如果发生溢出，则 ENO 将被设为 0。

状态字

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

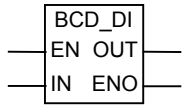
实例



如果 I0.0 的信号状态为 1，则执行转换。读取存储器字 MW10 中的整数，将其转换为 BCD 格式的三位数字。结果存储在存储器字 MW12 中。如果发生溢出，则输出 Q4.0 的信号状态将被设为 0。如果输入 EN 的信号状态为 0 (即转换未执行)，则输出 Q4.0 的信号状态也将为 0。

3.4 BCD_DI：BCD 码转换为长整型

符号



参数	数据类型	存储器区	描述
EN	BOOL	I、Q、M、D、L、T、C	启用输入
IN	DWORD	I、Q、M、D、L 或常数	BCD 格式的数字
OUT	DINT	I、Q、M、D、L	BCD 数字的长整型值
ENO	BOOL	I、Q、M、D、L	启用输出

描述

BCD 码转换为长整型指令读取输入参数 IN 中的内容，一个 BCD (二进制编码的十进制，± 9,999,999) 格式的七位数字，然后将该数字转换为长整型值。结果由输出参数 OUT 给出。

ENO 的信号状态始终与 EN 相同。

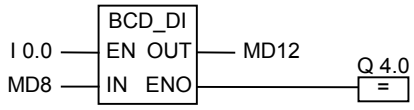
如果任意一个 BCD 格式的十进制数字位于无效范围 10 至 15 之间，则将在转换时发生 BCD 错误，导致以下反应：

- CPU 切换到 STOP 模式。"BCD 转换出错"被输入到诊断缓冲区中，事件 ID 号为 2521。
- 如果已编程 OB121，则将调用该块。

状态字

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

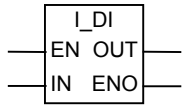
实例



如果 I0.0 的信号状态为 1，则执行转换。读取存储器双字 MD8 中的 BCD 格式的七位数字，将其转换为长整型数。结果存储在 MD12 中。如果转换已执行，则输出 Q4.0 的信号状态将为 1 (ENO = EN)。

3.5 I_DI：整型转换为长整型

符号



参数	数据类型	存储器区	描述
EN	BOOL	I、Q、M、D、L、T、C	启用输入
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	启用输出

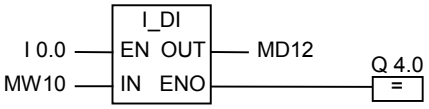
描述

整型转换为长整型指令以整型格式读取输入参数 IN 中的内容，将整型转换成长整型。结果由输出参数 OUT 给出。ENO 的信号状态始终与 EN 相同。

状态字

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

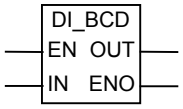
实例



如果 I0.0 的信号状态为 1，则执行转换。读取存储器字 MW10 中的整型数，将其转换为长整型。结果存储在存储器双字 MD12 中。如果转换已执行，则输出 Q4.0 的信号状态将为 1 (ENO = EN)。

3.6 DI_BCD：长整型转换为 BCD 码

符号



参数	数据类型	存储器区	描述
EN	BOOL	I、Q、M、D、L、T、C	启用输入
IN	DINT	I、Q、M、D、L 或常数	长整型
OUT	DWORD	I、Q、M、D、L	长整型数的 BCD 值
ENO	BOOL	I、Q、M、D、L	启用输出

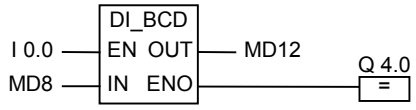
描述

长整型转换为 BCD 码指令读取输入参数 IN 中的内容，一个长整型值，将该值转换为 BCD 格式(± 9 999 999)的七位数字。结果由输出参数 OUT 给出。如果发生溢出，则 ENO 将被设为 0。

状态字

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

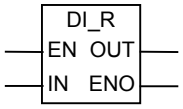
实例



如果 I0.0 的信号状态为 1，则执行转换。读取存储器双字 MD8 中的长整型数，将其转换为 BCD 格式的七位数字。结果存储在 MD12 中。如果发生溢出，则输出 Q4.0 的信号状态将被设为 0。如果输入 EN 的信号状态为 0 (即转换未执行)，则输出 Q4.0 的信号状态也将为 0。

3.7 DI_R：长整型转换为实数

符号



参数	数据类型	存储器区	描述
EN	BOOL	I、Q、M、D、L、T、C	启用输入
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	启用输出

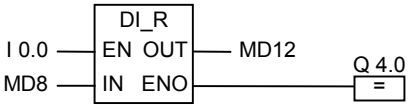
描述

长整型转换为实数指令读取输入参数 IN 中的内容，一个长整型值，并将该值转换为实数。结果由输出参数 OUT 给出。ENO 的信号状态始终与 EN 相同。

状态字

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

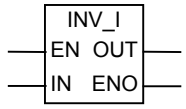
实例



如果 I0.0 的信号状态为 1，则执行转换。读取存储器双字 MD8 中的整型数，将其转换为实数。结果存储在存储器双字 MD12 中。如果转换未执行，则输出 Q4.0 的信号状态将为 0 (ENO = EN)。

3.8 INV_I：二进制反码整数

符号



参数	数据类型	存储器区	描述
EN	BOOL	I、Q、M、D、L、T、C	启用输入
IN	INT	I、Q、M、D、L 或常数	输入值
OUT	INT	I、Q、M、D、L	整数按位取反
ENO	BOOL	I、Q、M、D、L	启用输出

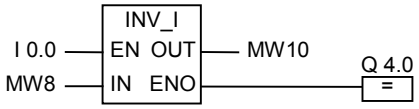
描述

对整数求反码指令读取输入参数 IN 中的内容，以掩码 FFFFH 执行布尔型字逻辑指令异或字，这样每位的值都被反转。结果由输出参数 OUT 给出。ENO 的信号状态始终与 EN 相同。

状态字

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

实例



如果 I0.0 的信号状态为 1，则执行转换。MW8 中每位的值都被反转：

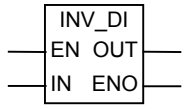
MW8 = 01000001 10000001 →

MW10 = 10111110 01111110

当 I0.0 的信号状态为 0 且 Q4.0 的信号状态为 0 (ENO = EN)时，不执行转换。

3.9 INV_DI：二进制反码双精度整数

符号



参数	数据类型	存储器区	描述
EN	BOOL	I、Q、M、D、L、T、C	启用输入
IN	DINT	I、Q、M、D、L 或常数	输入值
OUT	DINT	I、Q、M、D、L	长整型数按位取反
ENO	BOOL	I、Q、M、D、L	启用输出

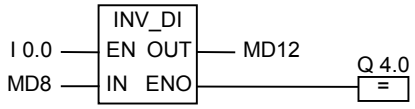
描述

二进制反码双精度整数指令读取输入参数 IN 中的内容，以掩码 FFFF FFFFH 执行布尔型字逻辑运算异或字，这样每位的值都被反转。结果由输出参数 OUT 给出。ENO 的信号状态始终与 EN 相同。

状态字

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

实例



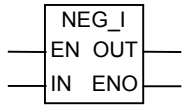
如果 I0.0 的信号状态为 1，则执行转换。存储器双字 MD8 中每个位的值都被反转：

MD8 = F0FF FFF0 → MD12 = 0F00 000F

如果未执行转换，则 Q4.0 为 0 (ENO = EN)。

3.10 NEG_I：二进制补码整数

符号



参数	数据类型	存储器区	描述
EN	BOOL	I、Q、M、D、L、T、C	启用输入
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	启用输出

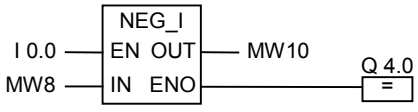
描述

对整数求补码指令读取输入参数 IN 中的内容，更改符号(例如，从正值变为负值)。结果由输出参数 OUT 给出。EN 和 ENO 的信号状态始终相同，除了当 EN 的信号状态为 1，发生溢出时。在这种情况下，ENO 的信号状态为 0。

状态字

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

实例



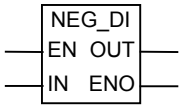
如果 I0.0 的信号状态为 1，则执行转换。存储器字 MW8 的值在 O 处输出到存储器字 MW10，并更改为相反的符号：

MW8 = +10 → MW10 = -10

如果 EN 的信号状态为 1，发生溢出，则 ENO 为 0，Q4.0 的信号状态为 0。如果未执行转换，则 Q4.0 为 0 (ENO = EN)。

3.11 NEG_DI：二进制补码双精度整数

符号



参数	数据类型	存储器区	描述
EN	BOOL	I、Q、M、D、L、T、C	启用输入
IN	DINT	I、Q、M、D、L 或常数	输入值
OUT	DINT	I、Q、M、D、L	长整数的补码
ENO	BOOL	I、Q、M、D、L	启用输出

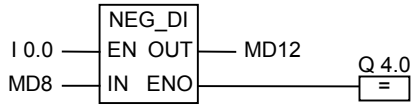
描述

二进制补码双精度整数指令读取输入参数 IN 中的内容，更改符号(例如，从正值变为负值)。结果由输出参数 OUT 给出。EN 和 ENO 的信号状态始终相同，除了当 EN 的信号状态为 1，发生溢出时。在这种情况下，ENO 的信号状态为 0。

状态字

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

实例



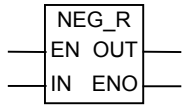
如果 I0.0 的信号状态为 1，则执行转换。存储器双字 MD8 的值在 O 处输出到存储器字 MD10，并更改为相反的符号：

MW8 = +10 → MW10 = -10

如果 EN 的信号状态为 1，发生溢出，则 ENO 为 0，Q4.0 的信号状态为 0。如果未执行转换，则 Q4.0 为 0 (ENO = EN)。

3.12 NEG_R：取反实数

符号



参数	数据类型	存储器区	描述
EN	BOOL	I、Q、M、D、L、T、C	启用输入
IN	REAL	I、Q、M、D、L 或常数	输入值
OUT	REAL	I、Q、M、D、L	结果为输入值取反。
ENO	BOOL	I、Q、M、D、L	启用输出

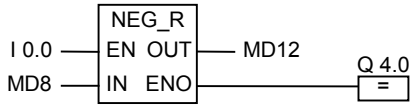
描述

取反实数指令读取输入参数 IN 中的内容，转换符号位(指令更改数字的符号，例如，从代表正号的 0 变为代表负号的 1)。指数和尾数的位保持不变。结果由输出参数 OUT 给出。ENO 和 EN 的信号状态始终相同，除了当 EN 的信号状态为 1，发生溢出时。在这种情况下，ENO 的信号状态为 0。

状态字

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

实例



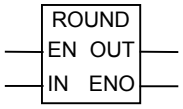
如果 I0.0 的信号状态为 1，则执行转换。存储器双字 MD8 的值改为相反的符号，在 O 处输出到存储器双字 MD12，如下例所示：

MD8 = + 6.234 → MD12 = - 6.234

如果转换未执行，则输出 Q4.0 的信号状态将为 0 (ENO = EN)。

3.13 ROUND：取整到最接近的双精度整数

符号



参数	数据类型	存储器区	描述
EN	BOOL	I、Q、M、D、L、T、C	启用输入
IN	REAL	I、Q、M、D、L 或常数	要取整的值
OUT	DINT	I、Q、M、D、L	IN 被取整到下一个双精度整数
ENO	BOOL	I、Q、M、D、L	启用输出

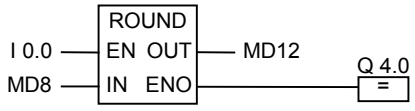
描述

取整到最接近的双精度整数指令读取输入参数 IN 中的内容，一个实数，并将该数字转换为双精度整数。结果为最接近的整数，保存在输出参数 OUT 中。如果小数部分为 x.5，数字将被取整到偶数(例如：2.5 -> 2，1.5 -> 2)。如果发生溢出，则 ENO 将被设为 0。如果输入值不是实数，则 OV 位和 OS 位将设为值 1，ENO 将设为值 0。

状态字

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

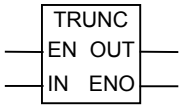
实例



如果 I0.0 为 1，则执行转换。读取存储器双字 MD8 中的实数，将其转换为双精度整数。该取整到最接近的双精度整数功能的结果保存在存储器双字 MD12 中。如果发生溢出，则输出 Q4.0 的信号状态将被设为 0。如果输入 EN 的信号状态为 0 (即转换未执行)，则输出 Q4.0 的信号状态也将为 0。

3.14 TRUNC：截取双精度整数部分

符号



参数	数据类型	存储器区	描述
EN	BOOL	I、Q、M、D、L、T、C	启用输入
IN	REAL	I、Q、M、D、L 或常数	要截取的值
OUT	DINT	I、Q、M、D、L	IN 的整数部分
ENO	BOOL	I、Q、M、D、L	启用输出

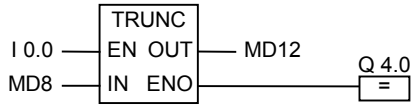
描述

截取双精度整数部分指令读取输入参数 IN 中的内容，一个实数，将该数字转换为双精度整数(例如，1.5 变为 1)。结果为实数的整数部分。结果由输出参数 OUT 给出。如果发生溢出，则 ENO 将被设为 0。如果输入值不是实数，则 OV 位和 OS 位将设为值 1，ENO 将设为值 0。

状态字

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

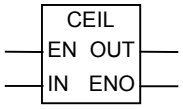
实例



如果 I0.0 的信号状态为 1，则执行转换。读取存储器双字 MD8 中的内容，一个实数，根据"向零方向取整的原则"，将其转换为双精度整数。结果为整数部分，保存在存储器双字 MD12 中。如果发生溢出，则输出 Q4.0 的信号状态将被设为 0。如果输入 EN 的信号状态为 0 (即转换未执行)，则输出 Q4.0 的信号状态也将为 0。

3.15 CEIL：向上取整

符号



参数	数据类型	存储器区	描述
EN	BOOL	I、Q、M、D、L、T、C	启用输入
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	启用输出

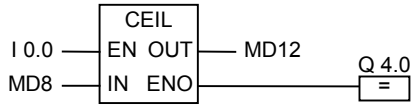
描述

向上取整指令读取输入参数 IN 中的内容，一个实数，并将该数字转换为双精度整数(例如 :+1.2 -> +2；-1.5 -> -1)。结果是大于或等于指定实数最小整数。结果由输出参数 OUT 给出。如果发生溢出，则 ENO 将被设为 0。如果输入值不是实数，则 OV 位和 OS 位将设为值 1，ENO 将设为值 0。

状态字

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

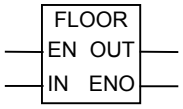
实例



如果 I0.0 为 1，则执行转换。读取存储器双字 MD8 中的内容，一个实数，通过取整到下一个大于(或等于)整个数字，将其转换为双精度整数，结果存储在存储器双字 MD12 中。如果发生溢出，则输出 Q4.0 的信号状态将被设为 0。如果输入 EN 的信号状态为 0 (即转换未执行)，则输出 Q4.0 的信号状态也将为 0。

3.16 FLOOR：向下取整

符号



参数	数据类型	存储器区	描述
EN	BOOL	I、Q、M、D、L、T、C	启用输入
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	启用输出

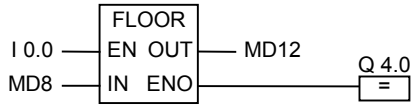
描述

向下取整指令读取输入参数 IN 中的内容，一个实数，并将该数字转换为双精度整数。结果是小于或等于指定实数最大整数。结果由输出参数 OUT 给出。如果发生溢出，则 ENO 将被设为 0。如果输入值不是实数，则 OV 位和 OS 位将设为值 1，ENO 将设为值 0。

状态字

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

实例



如果 I0.0 为 1，则执行转换。读取存储器双字 MD8 中的内容，一个实数，通过取整到下一个小于(或等于)整个数字，将其转换为双精度整数。结果存储在存储器双字 MD12 中。如果发生溢出，则输出 Q4.0 的信号状态将被设为 0。如果输入 EN 的信号状态为 0 (即转换未执行)，则输出 Q4.0 的信号状态也将为 0。

4 计数器指令

4.1 计数器指令概述

存储器中的区域

在 CPU 存储器中，有为计数器保留的区域。该存储区域为每个计数器地址保留一个 16 位字空间。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：

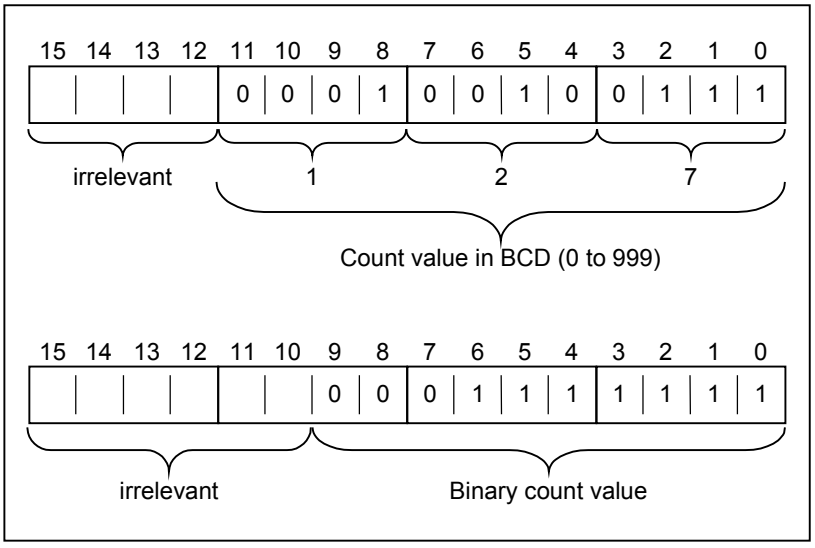
- S_CUD：分配参数和递增/递减计数
- S_CU：分配参数和递增计数
- S_CD：分配参数和递减计数
- SC：置位计数器数值
- CU：升值计数器
- CD：降值计数器

计数器中的位组态

输入从 0 至 999 的数字，您可为计数器提供预设值，例如，使用下列格式输入 127：C#127。其中 C#代表二进制编码十进制格式(BCD 格式：每个四位元组包含的二进制码代表一个十进制值)。

计数器中的 0 至 11 位包含二进制编码十进制格式的计数值。

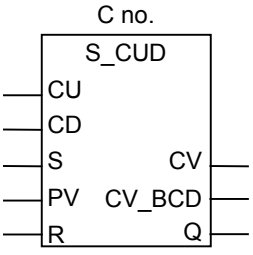
下图显示了加载计数值 127 之后计数器的内容，以及设置计数器之后计数器单元中的内容。



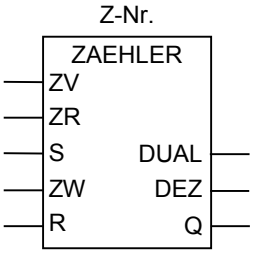
4.2 S_CUD：分配参数和递增/递减计数

符号

English



German



参数 英语	参数 德语	数据类型	存储器区	描述
编号	编号	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：升值计数器
CD	ZR	BOOL	I、Q、M、D、L	ZR 输入端：降值计数器
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
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	计数器的状态

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。

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

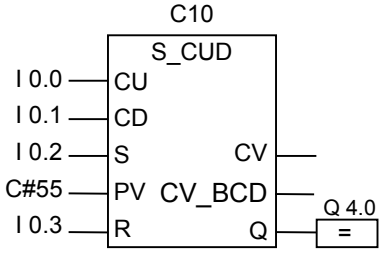
在输入端 R 的信号为"1"时会复位计数器。复位计数器时会将计数值设置为 0。

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	CC1	CC0	OV	OS	OR	STA	RLO	FC
写	-	-	-	-	-	X	X	X	1

实例



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。如果输入端 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。

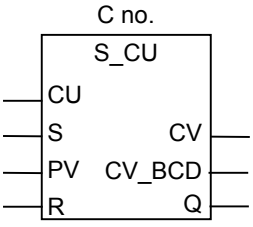
注意

避免在多个程序点上使用同一个计数器(存在计数错误的风险)。

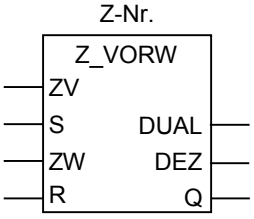
4.3 S_CU：分配参数和递增计数

符号

English



German



参数 英语	参数 德语	数据类型	存储器区	描述
编号	编号	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：升值计数器
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
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	计数器的状态

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。

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

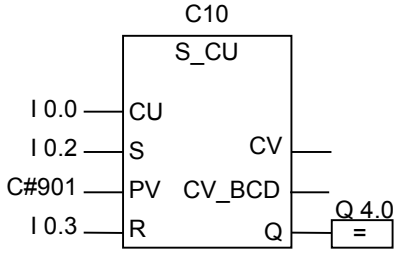
在输入端 R 的信号为"1"时会复位计数器。复位计数器时会将计数值设置为 0。

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	CC1	CC0	OV	OS	OR	STA	RLO	FC
写	-	-	-	-	-	X	X	X	1

实例



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。如果输入端 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。

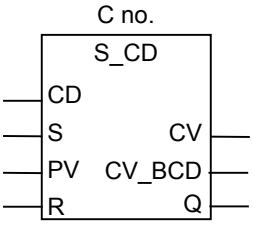
注意

避免在多个程序点上使用同一个计数器(存在计数错误的风险)。

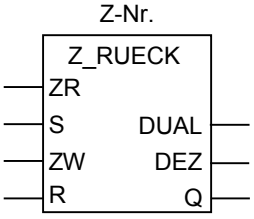
4.4 S_CD：分配参数和递减计数

符号

English



German



参数 英语	参数 德语	数据类型	存储器区	描述
编号	编号	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：降值计数器
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
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	计数器的状态

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。

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

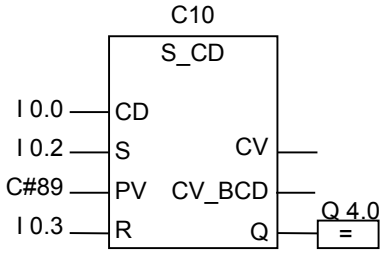
在输入端 R 的信号为"1"时会复位计数器。复位计数器时会将计数值设置为 0。

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	CC1	CC0	OV	OS	OR	STA	RLO	FC
写	-	-	-	-	-	X	X	X	1

实例



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。如果输入端 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。当计数器 C10 不等于 0 时，输出 Q4.0 的信号状态为 1。如果 I0.3 从 0 变化为 1，则会将 C10 的值设置为 0。

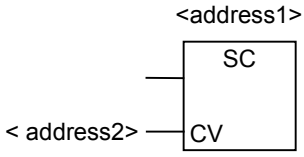
注意

避免在多个程序点上使用同一个计数器(存在计数错误的风险)。

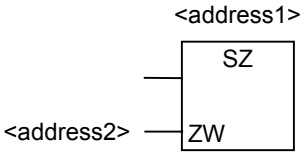
4.5 SC：置位计数器数值

符号

English



German



参数 英语	参数 德语	数据类型	存储器区	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

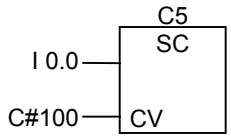
设置计数器值指令将为指定的计数器分配一个预设值。仅当在 RLO 中存在一个上升沿(信号状态从 0 变化为 1)时，才会执行此指令。

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	CC1	CC0	OV	OS	OR	STA	RLO	FC
写	-	-	-	-	-	0	-	-	0

实例



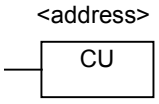
在输入 I0.0 的信号状态从 0 变化为 1(RLO 中的上升沿)时将计数器 C5 的值预设为 100。C#表示您将输入一个 BCD 格式的值。

如果不存在上升沿，则 C5 的计数器值不变。

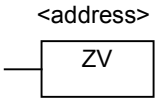
4.6 CU：升值计数器

符号

English



German



参数	数据类型	存储器区	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**递增计数器**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。

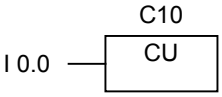
设置计数器值指令用于设置计数器的值。

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

状态字

	BR	CC1	CC0	OV	OS	OR	STA	RLO	FC
写	-	-	-	-	-	0	-	-	0

实例



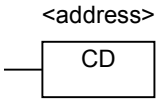
如果输入端 I0.0 的信号状态从 0 变化为 1(RLO 中存在上升沿)，则计数器 C10 的值增加 1(在 C10 值已经是 999 时例外)。

如果不存在上升沿，则 C10 的值保持不变。

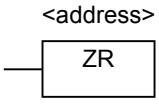
4.7 CD：降值计数器

符号

English



German



参数	数据类型	存储器区	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**递减计数器**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。

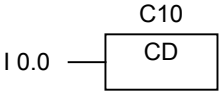
设置计数器值指令用于设置计数器的值。

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

状态字

	BR	CC1	CC0	OV	OS	OR	STA	RLO	FC
写	-	-	-	-	-	0	-	-	0

实例



如果输入端 I0.0 的信号状态从 0 变化为 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	存储器区	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)或实例数据块(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。

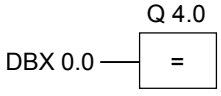
状态字

	BR	CC1	CC0	OV	OS	OR	STA	RLO	FC
写	-	-	-	-	-	-	-	-	-

实例

程序段 1

程序段 2



DB10 是当前打开的数据块。因此 DBX0.0 的扫描参考数据块 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

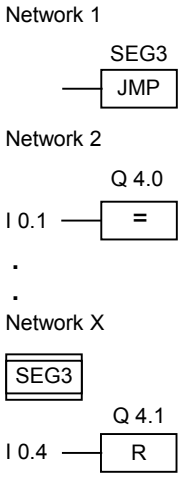
跳转标签作为地址

跳转指令的地址是标号。跳转标号指示程序将要跳转到的目标。

在 JMP 框上方输入标签。标号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)。

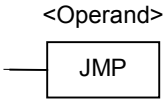
跳转标签作为目标

目标标号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。在显示的空框中，键入标号的名称。



6.2 JMP：块中无条件跳转

符号



参数	数据类型	存储器区	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。

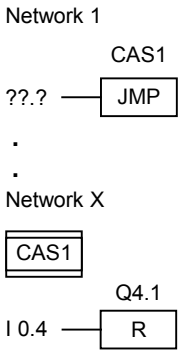
在**块中无条件跳转**框之前，不能有任何逻辑运算。

状态字

	BR	CC1	CC0	OV	OS	OR	STA	RLO	FC
写	-	-	-	-	-	-	-	-	-

该指令不改变状态字的位。

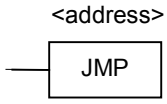
实例



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	存储器区	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

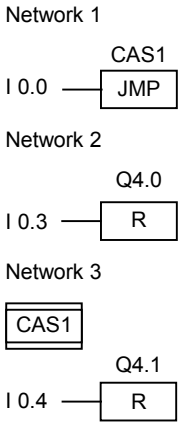
如果 ROL 为 1，则**块中有条件跳转**指令相当于"跳转到标签"指令。FBD 元素"无条件跳转"也可用于此操作，但本指令需要基于之前的逻辑运算来执行。仅当此逻辑运算的结果为 1 时才执行该有条件跳转指令。不执行跳转指令和标签之间的所有指令。

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。

状态字

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

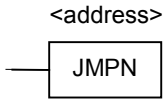
实例



在输入端 I0.0 的信号状态为 1 时执行跳转到标签 CAS1。即使输入端 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	存储器区	描述
跳转标签的名称	-	-	地址指定在 RLO 为 0 时程序将跳转到的位置的标签。

描述

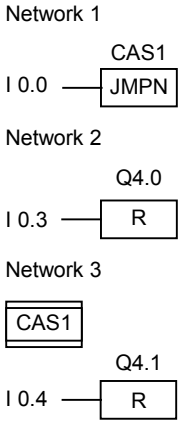
若非则跳转指令相当于在 RLO 为 0 时执行的"跳转到标签"指令

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	CC1	CC0	OV	OS	OR	STA	RLO	FC
写	-	-	-	-	-	0	1	1	0

实例



在输入端 I0.0 的信号状态为 0 时执行跳转到标签 CAS1。即使输入端 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。

不执行该跳转操作及其标签间的所有指令。

6.5 LABEL：跳转标签

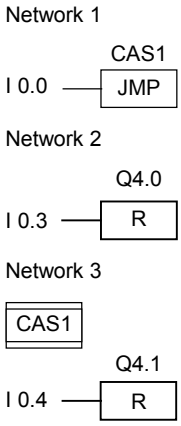
符号

描述

跳转标签是跳转指令目标的标识符。标号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。

实例



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 整型数学运算指令

7.1 整数算术指令概述

描述

使用整数运算，您可以对**两个整数**(16 和 32 位)执行以下运算：

- ADD_I：加上整数
- SUB_I：减去整数
- MUL_I：乘以整数
- DIV_I：除以整数
- ADD_DI：加上双精度整数
- SUB_DI：减去双精度整数
- MUL_DI：乘以双精度整数
- DIV_DI：除以双精度整数
- MOD_DI：返回双精度除法的余数

参见使用整数算术指令时得出状态字的位数值。

7.2 使用整数算术指令时得出状态字的位数值

描述

整数运算指令影响状态字中的以下位：CC1 和 CC0、OV 和 OS。

下表显示指令结果为整数(16 位和 32 位)时状态字中各位的信号状态：

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	*

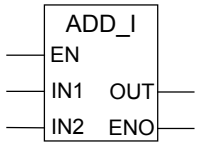
* 指令结果不影响 OS 位。

结果的无效范围	A1	A0	OV	OS
下溢(加法) 16 位：结果 = -65536 32 位：结果 = -4 294 967 296	0	0	1	1
下溢(乘法) 16 位：结果 < -32 768 (负数) 32 位：结果 < -2 147 483 648 (负数)	0	1	1	1
溢出(加法、减法) 16 位：结果 > 32 767 (正数) 32 位：结果 > 2 147 483 647 (正数)	0	1	1	1
溢出(乘法、除法) 16 位：结果 > 32 767 (正数) 32 位：结果 > 2 147 483 647 (正数)	1	0	1	1
下溢(加法、减法) 16 位：结果 < -32. 768 (负数) 32 位：结果 < -2 147 483 648 (负数)	1	0	1	1
被 0 除	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：加上整数

符号



参数	数据类型	存储器区	描述
EN	BOOL	I、Q、M、D、L、T、C	启用输入
IN1	INT	I、Q、M、D、L 或常数	被加数
IN2	INT	I、Q、M、D、L 或常数	加数
OUT	INT	I、Q、M、D、L	相加结果
ENO	BOOL	I、Q、M、D、L	启用输出

描述

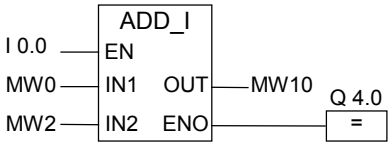
启用输入(EN)的信号状态 1 激活**加上整数**指令。该指令将输入 IN1 和 IN2 相加。结果可从输出 OUT 处扫描得到。如果输入或结果都不是浮点数，OV 位和 OS 位将被设为 1，ENO 被设为 0。

参见使用整数算术指令时得出状态字的位数值。

状态字

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

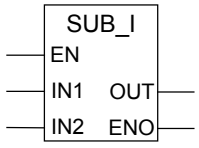
实例



在输入端 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。如结果不在整型数的允许范围内或输入端 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：减去整数

符号



参数	数据类型	存储器区	描述
EN	BOOL	I、Q、M、D、L、T、C	启用输入
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	相减结果
ENO	BOOL	I、Q、M、D、L	启用输出

描述

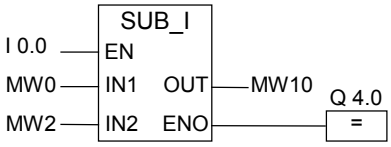
使能(EN)输入的信号状态为 1 时会激活**减去整数**指令。该指令将输入 IN1 减去 IN2。结果可从输出 OUT 处扫描得到。若结果超出整型数的允许范围，则状态字的 OV 和 OS 位会被置 1，且 ENO 被置 0。

参见使用整数算术指令时得出状态字的位数值。

状态字

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

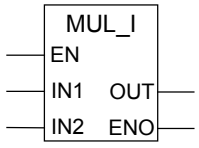
实例



在输入端 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。如结果不在整型数的允许范围内或输入端 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：乘以整数

符号



参数	数据类型	存储器区	描述
EN	BOOL	I、Q、M、D、L、T、C	启用输入
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	相乘结果
ENO	BOOL	I、Q、M、D、L	启用输出

描述

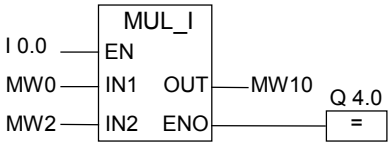
启用输入(EN)的信号状态 1 激活**乘以整数**指令。该指令将输入 IN1 乘以 IN2。其结果是可在 OUT 处扫描的 32 位整型数。若结果超出 16 位整型数的允许范围 ,则状态字的 OV 和 OS 位会被置 1 ,且 ENO 被置 0。

参见使用整数算术指令时得出状态字的位数值。

状态字

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

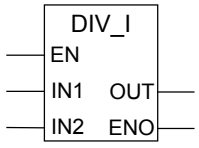
实例



在输入端 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端 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：除以整数

符号



参数	数据类型	存储器区	描述
EN	BOOL	I、Q、M、D、L、T、C	启用输入
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	除法结果
ENO	BOOL	I、Q、M、D、L	启用输出

描述

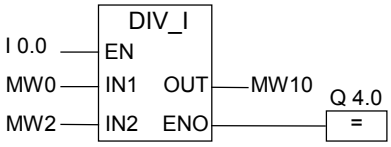
使能(EN)输入的信号状态为 1 时会激活**除以整数**指令。该指令将输入 IN1 除以 IN2。可在 OUT 处扫描整数商值(整数部分的结果)。但不能扫描余数。如果商超出整型数的允许范围，则状态字的 OV 和 OS 位会被置 1，且 ENO 被置 0。

参见使用整数算术指令时得出状态字的位数值。

状态字

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

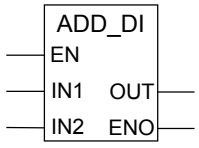
实例



在输入端 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端 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：加上双精度整数

符号



参数	数据类型	存储器区	描述
EN	BOOL	I、Q、M、D、L、T、C	启用输入
IN1	DINT	I、Q、M、D、L 或常数	被加数
IN2	DINT	I、Q、M、D、L 或常数	加数
OUT	DINT	I、Q、M、D、L	相加结果
ENO	BOOL	I、Q、M、D、L	启用输出

描述

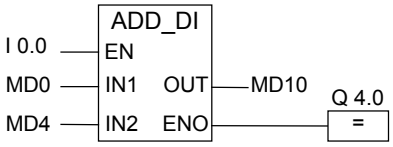
使能(EN)输入的信号状态为 1 时会激活**双精度整数加法**指令。该指令将输入 IN1 和 IN2 相加。结果可从输出 OUT 处扫描得到。若结果超出长整型数的允许范围，则状态字的 OV 和 OS 位会被置 1，且 ENO 被置 0。

参见使用整数算术指令时得出状态字的位数值。

状态字

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

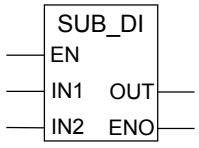
实例



在输入端 I0.0 的信号状态为 1 时会激活 ADD_DI 框。MD0 + MD4 相加的结果被输入到存储器双字 MD10 中。若结果超出长整型数的允许范围或输入端 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：减去双精度整数

符号



参数	数据类型	存储器区	描述
EN	BOOL	I、Q、M、D、L、T、C	启用输入
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	相减结果
ENO	BOOL	I、Q、M、D、L	启用输出

描述

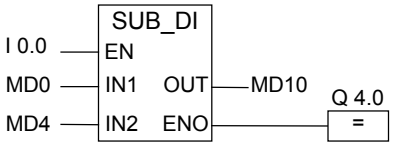
使能(EN)输入的信号状态为 1 时会激活**减去双精度整数**指令。该指令将输入 IN1 减去 IN2。结果可从输出 OUT 处扫描得到。若结果超出长整型数的允许范围 ,则状态字的 OV 和 OS 位会被置 1 ,且 ENO 被置 0。

参见使用整数算术指令时得出状态字的位数值。

状态字

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

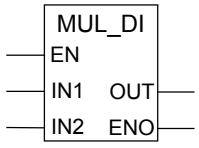
实例



输入端 I0.0 的信号状态为 1 时会激活 SUB_DI 框。MD0 - MD4 相减的结果被输入到存储器双字 MD10 中。若结果超出长整型数的允许范围或输入端 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：乘以双精度整数

符号



参数	数据类型	存储器区	描述
EN	BOOL	I、Q、M、D、L、T、C	启用输入
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	相乘结果
ENO	BOOL	I、Q、M、D、L	启用输出

描述

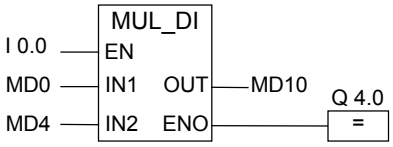
使能(EN)输入的信号状态为 1 时会激活**乘以双精度整数**指令。此指令将输入 IN1 与 IN2 相乘。结果可从输出 OUT 处扫描得到。若结果超出长整型数的允许范围，则状态字的 OV 和 OS 位会被置 1，且 ENO 被置 0。

参见使用整数算术指令时得出状态字的位数值。

状态字

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

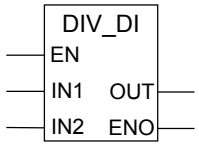
实例



在输入端 I0.0 的信号状态为 1 时会激活 MUL_DI 框。MD0 x MD4 相乘的结果被输入到存储器双字 MD10 中。若结果超出长整型数的允许范围或输入端 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：除以双精度整数

符号



参数	数据类型	存储器区	描述
EN	BOOL	I、Q、M、D、L、T、C	启用输入
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	除法结果
ENO	BOOL	I、Q、M、D、L	启用输出

描述

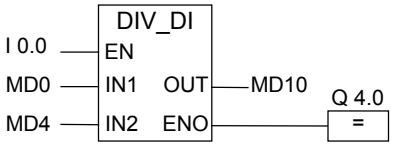
使能(EN)输入的信号状态为 1 时会激活**除以双精度整数**指令。该指令将输入 IN1 除以 IN2。可在 OUT 处扫描商(整数部分的结果)。除以双精度整数指令以 DINT 格式将商存储为 32 位单精度值。此指令不生成余数。如果商超出长整型数的允许范围，则状态字的 OV 和 OS 位会被置 1，且 ENO 被置 0。

参见使用整数算术指令时得出状态字的位数值。

状态字

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

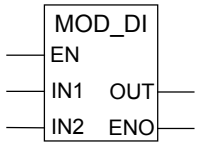
实例



在输入端 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。如果商超出长整型数的允许范围或输入端 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：返回双精度除法的余数

符号



参数	数据类型	存储器区	描述
EN	BOOL	I、Q、M、D、L、T、C	启用输入
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	启用输出

描述

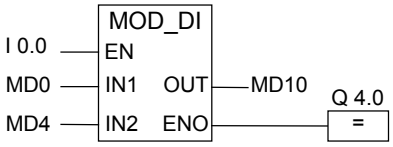
使能(EN)输入的信号状态为 1 时会激活**返回双精度除法的余数**指令。该指令将输入 IN1 除以 IN2。可在 OUT 处扫描余数(分数)。若结果超出长整型数的允许范围，则状态字的 OV 和 OS 位会被置 1，且 ENO 被置 0。

参见使用整数算术指令时得出状态字的位数值。

状态字

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

实例



在输入端 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。若结果超出长整型数的允许范围或输入端 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 浮点运算指令

8.1 浮点型数学运算总览

描述

IEEE 32 位浮点数属于被称作实数(REAL)的数据类型。您可使用浮点运算指令通过**两个** 32 位 IEEE 浮点数来执行下列数学运算指令：

- ADD_R：加上实数
- SUB_R：减去实数
- MUL_R：乘以实数
- DIV_R：除以实数

可对**一个** 32 位 IEEE 浮点数执行下列运算：

- 生成浮点数的绝对值(ABS)
- 生成浮点数的平方(SQR)或平方根(SQRT)
- 生成浮点数的自然对数(LN)
- 生成浮点数以 e (= 2.71828...)为底的指数值 (EXP)
- 生成以 32 位浮点数表示的角度的下列三角函数。
  - 正弦(SIN)和反正弦(ASIN)
  - 余弦(COS)和反余弦(ACOS)
  - 正切(TAN)和反正切(ATAN)

参见评估浮点数指令状态字的位。

8.2 评估浮点数指令状态字的位

描述

浮点指令影响状态字中的下列位：CC 1 和 CC 0、OV 和 OS。

下表说明了指令结果为浮点数(32 位)时状态字中各位的信号状态：

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	*

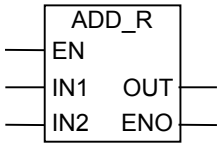
* 指令结果不影响 OS 位。

结果的无效范围	CC 1	CC 0	OV	OS
下溢 -1.175494E-38 < 结果 < - 1.401298E-45 (负数)	0	0	1	1
下溢 +1.401298E-45 < 结果 < +1.175494E-38 (正数)	0	0	1	1
上溢 结果 < -3.402823E+38 (负数)	0	1	1	1
溢出 结果 > 3.402823E+38 (正数)	1	0	1	1
无效浮点数或非法指令 (输入值不在有效范围内)	1	1	1	1

8.3 基本指令

8.3.1 ADD_R：加上实数

符号



参数	数据类型	存储器区	描述
EN	BOOL	I、Q、M、D、L、T、C	启用输入
IN1	REAL	I、Q、M、D、L 或常数	要加的第一个数
IN2	REAL	I、Q、M、D、L 或常数	要加的第二个数
OUT	REAL	I、Q、M、D、L	相加结果
ENO	BOOL	I、Q、M、D、L	启用输出

描述

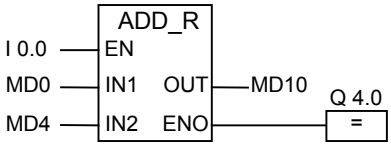
启用输入(EN)的信号状态 1 激活**加上实数**指令。该指令将输入 IN1 和 IN2 相加。结果可从输出 OUT 处扫描得到。如果输入或结果都不是浮点数，OV 位和 OS 位将被设为 1，ENO 被设为 0。

参见评估浮点数指令状态字的位。

状态字

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

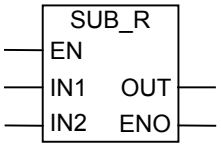
实例



输入 I0.0 的信号状态 1 激活 ADD_R 框。MD0 + MD4 相加的结果被输入到存储器双字 MD10 中。如果输入或结果都不是浮点数，且 I0.0 的信号状态为 0，则输出 Q4.0 将被设为 0，不执行指令。

8.3.2 SUB_R：减去实数

符号



参数	数据类型	存储器区	描述
EN	BOOL	I、Q、M、D、L、T、C	启用输入
IN1	REAL	I、Q、M、D、L 或常数	被减数(从其中减去第二个值)
IN2	REAL	I、Q、M、D、L 或常数	减数(要从第一个值中减去的数)
OUT	REAL	I、Q、M、D、L	相减结果
ENO	BOOL	I、Q、M、D、L	启用输出

描述

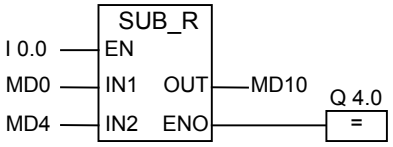
启用输入(EN)的信号状态 1 激活**减去实数**指令。该指令将输入 IN1 减去 IN2。结果可从输出 OUT 处扫描得到。如果输入或结果都不是浮点数，OV 位和 OS 位将被设为 1，ENO 被设为 0。

参见评估浮点数指令状态字的位。

状态字

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

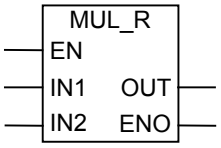
实例



输入 I0.0 的信号状态 1 激活 SUB_R 框。MD0 - MD4 相减的结果被输入到存储器双字 MD10 中。如果输入或结果都不是浮点数，且 I0.0 的信号状态为 0，则输出 Q4.0 将被设为 0，不执行指令。

8.3.3 MUL_R：乘以实数

符号



参数	数据类型	存储器区	描述
EN	BOOL	I、Q、M、D、L、T、C	启用输入
IN1	REAL	I、Q、M、D、L 或常数	被乘数(要被乘的值)
IN2	REAL	I、Q、M、D、L 或常数	乘数(乘以第一个值的值)
OUT	REAL	I、Q、M、D、L	乘运算结果
ENO	BOOL	I、Q、M、D、L	启用输出

描述

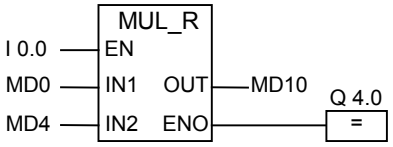
启用输入(EN)的信号状态 1 激活**乘以实数**指令。该指令将输入 IN1 乘以 IN2。结果可从输出 OUT 处扫描得到。如果输入或结果都不是浮点数，OV 位和 OS 位将被设为 1，ENO 被设为 0。

参见评估浮点数指令状态字的位。

状态字

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

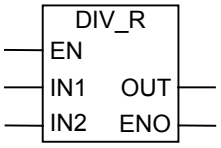
实例



输入 I0.0 的信号状态 1 激活 MUL_R 框。MD0 x MD4 相乘的结果被输入到存储器双字 MD10 中。如果输入或结果都不是浮点数，且 I0.0 的信号状态为 0，则输出 Q4.0 将被设为 0，不执行指令。

8.3.4 DIV_R：除以实数

符号



参数	数据类型	存储器区	描述
EN	BOOL	I、Q、M、D、L、T、C	启用输入
IN1	REAL	I、Q、M、D、L 或常数	被除数(要除以第二个值的值)
IN2	REAL	I、Q、M、D、L 或常数	除数(除第一个值的值)
OUT	REAL	I、Q、M、D、L	除法结果
ENO	BOOL	I、Q、M、D、L	启用输出

描述

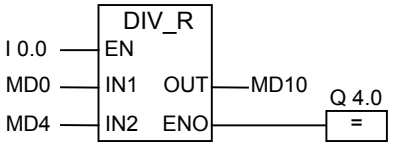
启用输入(EN)的信号状态 1 激活**除以实数**指令。该指令将输入 IN1 除以 IN2。结果可从输出 OUT 处扫描得到。如果输入或结果都不是浮点数，OV 位和 OS 位将被设为 1，ENO 被设为 0。

参见评估浮点数指令状态字的位。

状态字

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

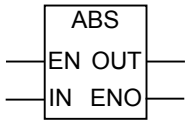
实例



输入 I0.0 的信号状态 1 激活 DIV_R 框。MD0 除以 MD4 的结果被输入到存储器双字 MD10 中。如果输入或结果都不是浮点数，且 I0.0 的信号状态为 0，则输出 Q4.0 将被设为 0，不执行指令。

8.3.5 ABS：生成浮点数的绝对值

符号



参数	数据类型	存储器区	描述
EN	BOOL	I、Q、M、D、L、T、C	启用输入
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	启用输出

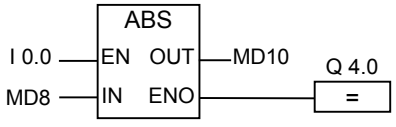
描述

使用**生成浮点数的绝对值**指令，可生成浮点数的绝对值。

状态字

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

实例



如果 I0.0 = 1，MD8 的绝对值被输出到 MD12。

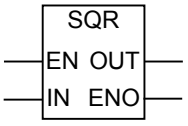
MD8 = +6.234，MD12 中的结果 = 6.234 x 1。

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 : 生成浮点数的平方

符号



参数	数据类型	存储器区	描述
EN	BOOL	I、Q、M、D、L、T、C	启用输入
IN	REAL	I、Q、M、D、L 或常数	编号
OUT	REAL	I、Q、M、D、L	数字的平方
ENO	BOOL	I、Q、M、D、L	启用输出

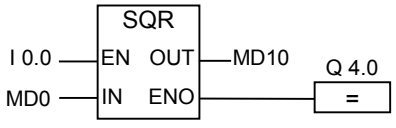
描述

使用**生成浮点数的平方**指令，可得到浮点数的平方值。如果输入或结果都不是浮点数，OV 位和 OS 位将被设为 1，ENO 被设为 0。

状态字

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

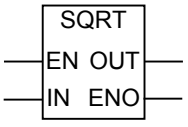
实例



输入 I0.0 的信号状态 1 激活 SQR 框。SQR 的结果(MD0)被输入到存储器双字 MD10 中。如果 MD0 小于 0，或者输入或结果都不是浮点数，且 I0.0 的信号状态为 0，则输出 Q4.0 将被设为 0。

8.4.2 SQRT：生成浮点数的平方根

符号



参数	数据类型	存储器区	描述
EN	BOOL	I、Q、M、D、L、T、C	启用输入
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	启用输出

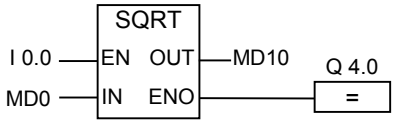
描述

使用**生成浮点数的平方根**指令，可得到浮点数的平方根。如果地址中的值大于"0"，该指令返回一个正数结果。如果输入或结果都不是浮点数，OV 位和 OS 位将被设为 1，ENO 被设为 0。

状态字

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

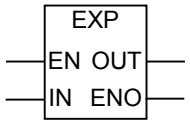
实例



输入 I0.0 的信号状态 1 激活 SQRT 框。SQR 的结果(MD0)被输入到存储器双字 MD10 中。如果 MD0 小于 0，或者输入或结果都不是浮点数，且 I0.0 的信号状态为 0，则输出 Q4.0 将被设为 0。

8.4.3 EXP：生成浮点数的指数值

符号



参数	数据类型	存储器区	描述
EN	BOOL	I、Q、M、D、L、T、C	启用输入
IN	REAL	I、Q、M、D、L 或常数	编号
OUT	REAL	I、Q、M、D、L	数字的指数值
ENO	BOOL	I、Q、M、D、L	启用输出

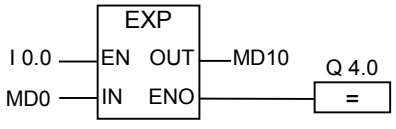
描述

使用**生成浮点数的指数值**指令，可生成浮点数以 e (= 2.71828...)为底的指数值。如果输入或结果都不是浮点数，OV 位和 OS 位将被设为 1，ENO 被设为 0。

状态字

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

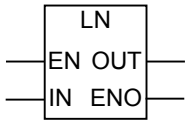
实例



输入 I0.0 的信号状态 1 激活 EXP 框。EXP 的结果(MD0)被输入到存储器双字 MD10 中。如果输入或结果都不是浮点数，且 I 0.0 的信号状态为 0，则输出 Q4.0 将被设为 0。

8.4.4 LN：生成浮点数的自然对数

符号



参数	数据类型	存储器区	描述
EN	BOOL	I、Q、M、D、L、T、C	启用输入
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	启用输出

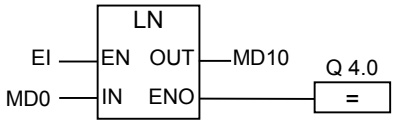
描述

使用**生成浮点数的自然对数**指令，可生成浮点数的自然对数。如果输入或结果都不是浮点数，OV 位和 OS 位将被设为 1，ENO 被设为 0。

状态字

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

实例



输入 I0.0 的信号状态 1 激活 LN 框。LN 的结果(MD0)被输入到存储器双字 MD10 中。如果 MD0 小于 0，或者输入或结果都不是浮点数，且 I0.0 的信号状态为 0，则输出 Q4.0 将被设为 0。

8.4.5 生成浮点值角度的三角函数

描述

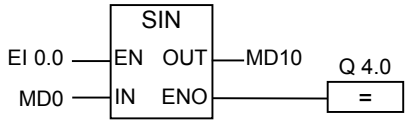
使用下列指令，可生成以 32 位 IEEE 浮点数表示的角度的三角函数值。

指令	含义
SIN	生成以弧度指定的角度浮点数的正弦值。
COS	生成以弧度指定的角度浮点数的余弦值。
TAN	生成以弧度指定的角度浮点数的正切值。
ASIN	生成浮点数的反正弦值。结果是用弧度表示的角度。值位于下列范围： -π / 2 <= 反正弦 <= + π / 2，其中，π = 3.14...
ACOS	生成浮点数的反余弦值。结果是用弧度表示的角度。值位于下列范围： 0 v 反余弦 <= + π，其中，π = 3.14...
ATAN	生成浮点数的反正切值。结果是用弧度表示的角度。值位于下列范围： -π / 2 <= 反正切 <= + π / 2，其中，π = 3.14...

状态字

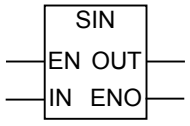
	BR	CC1	CC0	OV	OS	OR	STA	RLO	FC
写	X	X	X	X	X	0	X	X	1

实例



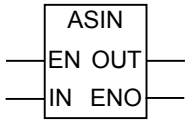
输入 I0.0 的信号状态 1 激活 SIN 框。SIN 的结果(MD0)被输入到存储器双字 MD10 中。如果输入或结果都不是浮点数，且 I0.0 的信号状态为 0，则输出 Q4.0 将被设为 0。

符号



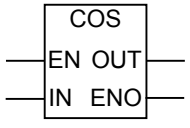
参数	数据类型	存储器区	描述
EN	BOOL	I、Q、M、D、L、T、C	启用输入
IN	REAL	I、Q、M、D、L 或常数	编号
OUT	REAL	I、Q、M、D、L	数字的正弦值
ENO	BOOL	I、Q、M、D、L	启用输出

符号



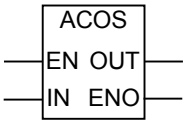
参数	数据类型	存储器区	描述
EN	BOOL	I、Q、M、D、L、T、C	启用输入
IN	REAL	I、Q、M、D、L 或常数	编号
OUT	REAL	I、Q、M、D、L	数字的反正弦值
ENO	BOOL	I、Q、M、D、L	启用输出

符号



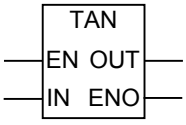
参数	数据类型	存储器区	描述
EN	BOOL	I、Q、M、D、L、T、C	启用输入
IN	REAL	I、Q、M、D、L 或常数	编号
OUT	REAL	I、Q、M、D、L	数字的余弦值
ENO	BOOL	I、Q、M、D、L	启用输出

符号



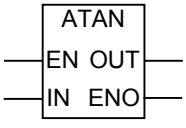
参数	数据类型	存储器区	描述
EN	BOOL	I、Q、M、D、L、T、C	启用输入
IN	REAL	I、Q、M、D、L 或常数	编号
OUT	REAL	I、Q、M、D、L	数字的反余弦值
ENO	BOOL	I、Q、M、D、L	启用输出

符号



参数	数据类型	存储器区	描述
EN	BOOL	I、Q、M、D、L、T、C	启用输入
IN	REAL	I、Q、M、D、L 或常数	编号
OUT	REAL	I、Q、M、D、L	数字的正切值
ENO	BOOL	I、Q、M、D、L	启用输出

符号

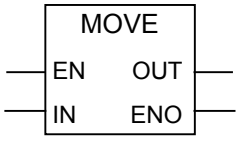


参数	数据类型	存储器区	描述
EN	BOOL	I、Q、M、D、L、T、C	启用输入
IN	REAL	I、Q、M、D、L 或常数	编号
OUT	REAL	I、Q、M、D、L	数字的反正切值
ENO	BOOL	I、Q、M、D、L	启用输出

9 传送指令

9.1 MOVE : 分配值

符号



参数	数据类型	存储器区	描述
EN	BOOL	I、Q、M、D、L、T、C	启用输入
IN	所有 8、16 或 32 位长度的基本数据类型	I、Q、M、D、L 或常数	源值
OUT	所有 8、16 或 32 位长度的基本数据类型	I、Q、M、D、L	目标地址
ENO	BOOL	I、Q、M、D、L	启用输出

描述

使用**分配值**指令，可将特定值分配给变量。

在 IN 输入中指定的值被复制到 OUT 输出中指定的地址中。ENO 与 EN 具有相同的信号状态。

使用 MOVE 方框，"分配值"指令可复制所有 8、16 或 32 位长度的基本数据类型。用户自定义数据类型，如阵列或结构，必须使用系统功能 SFC 20 "BLKMOV"来复制。

分配值指令受主控继电器(MCR)影响。关于 MCR 功能的更多信息，请参见主控制继电器打开/关。

状态字

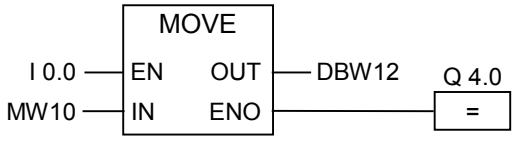
	BR	CC1	CC0	OV	OS	OR	STA	RLO	FC
写	1	-	-	-	-	0	1	1	1

注意

在将值移动到不同长度的数据类型中时，将根据需要截去或以零填充高值字节：

实例：双字	**1111 1111**	**0000 1111**	**1111 0000**	**0101 0101**
移动	**结果**			
至双字：	1111 1111	0000 1111	1111 0000	0101 0101
至字节：				0101 0101
至字：			1111 0000	0101 0101
实例：Byte:				**1111 0000**
移动	**结果**			
至字节：				1111 0000
至字：			0000 0000	1111 0000
至双字：	0000 0000	0000 0000	0000 0000	1111 0000

实例



当输入 I0.0 为 1 时，执行指令。MW10 中的内容被复制到所打开的 DB 的数据字 12 中。

如果已执行指令，则 Q4.0 将被设为 1。

10 程序控制指令

10.1 程序控制指令总览

描述

下列指令可用于执行程序控制操作：

- CALL：调用不带参数的 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
- 调用多重实例
- 从库中调用块

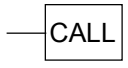
- 主控继电器指令
- 关于使用 MCR 功能的重要注意事项
- MCR< 主控继电器打开
- MCR> 主控继电器关闭
- MCRA 主控继电器激活
- MCRD 主控制继电器去活

- RET 返回

10.2 CALL：调用不带参数的 FC/SFC

符号

<FC-/SFC 编号>



参数	数据类型	存储器区	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(参见实例)。

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：

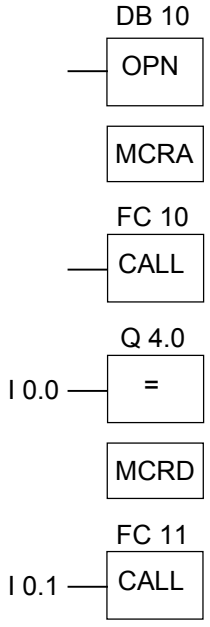
- 保存要求返回调用块的地址。
- 保存数据块寄存器(数据块和实例数据块)。
- 将 MA 位(有效 MCR 位)写入到块堆栈(BSTACK)中。
- 为被调用的 FC 或 SFC 创建新的本地数据区。

然后在被调用块中继续执行程序。

状态字

		BR	CC1	CC0	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

实例



当对 FC10 执行无条件调用时，CALL 指令执行下列功能：

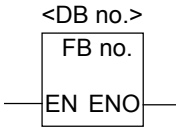
- 保存要求返回当前 FB 的地址。
- 为 DB10 和 FB 的实例数据块保存选择器。
- 将在 MCRA 指令中设置成 1 的 MA 位压入块堆栈(BSTACK)，然后为被调用的 FC10 将该位复位成 0

继续在 FC10 中执行程序。如果要在 FC10 中使用 MCR 功能，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 的实例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	存储器区	描述
EN	BOOL	I、Q、M、L、D	启用输入
ENO	BOOL	I、Q、M、L、D	启用输出
FB 编号 DB 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，
- 保存两个当前打开的数据块(DB 和实例 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

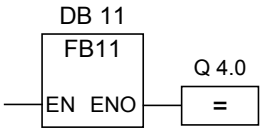
		BR	CC1	CC0	OV	OS	OR	STA	RLO	FC
条件调用	写	X	-	-	-	0	0	X	X	X
无条件调用	写	-	-	-	-	0	0	X	X	X

实例

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：

保存调用功能块的返回地址和 DB10 以及调用功能块的实例数据块的选择数据。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实例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，上一个打开的数据块的编号将会丢失。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	存储器区	描述
EN	BOOL	I、Q、M、L、D	启用输入
ENO	BOOL	I、Q、M、L、D	启用输出
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。

如果调用功能，且被调用块的变量声明表具有 IN、OUT 和 IN_OUT 声明，这些变量将作为形式参数表被添加到调用块的程序中。

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

状态字

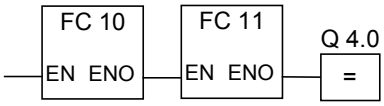
		BR	CC1	CC0	OV	OS	OR	STA	RLO	FC
条件调用	写	X	-	-	-	0	0	X	X	X
无条件调用	写	-	-	-	-	0	0	X	X	X

实例

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 以及调用功能块的实例数据块的选择数据。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

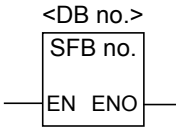
正确执行了 FC11 时，将 ENO 设置成 1，因此也将 Q4.0 设置成 1。

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	存储器区	描述
EN	BOOL	I、Q、M、L、D	启用输入
ENO	BOOL	I、Q、M、L、D	启用输出
SFB 编号 DB 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 指令时：

- 保存调用块的返回地址，
- 保存两个当前打开的数据块(DB 和实例 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

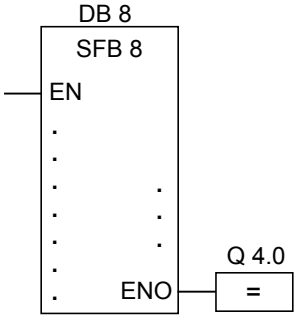
		BR	CC1	CC0	OV	OS	OR	STA	RLO	FC
条件调用	写	X	-	-	-	0	0	X	X	X
无条件调用	写	-	-	-	-	0	0	X	X	X

实例

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：

保存调用功能块的返回地址和 DB10 以及调用功能块的实例数据块的选择数据。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实例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，上一个打开的数据块的编号将会丢失。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	存储器区	描述
EN	BOOL	I、Q、M、L、D	启用输入
ENO	BOOL	I、Q、M、L、D	启用输出
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

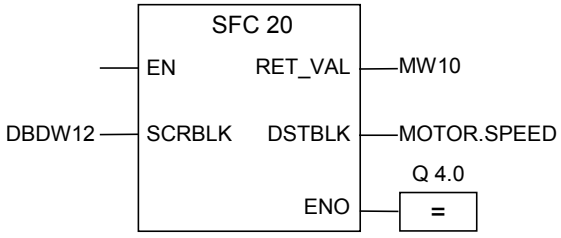
		BR	CC1	CC0	OV	OS	OR	STA	RLO	FC
条件调用	写	X	-	-	-	0	0	X	X	X
无条件调用	写	-	-	-	-	0	0	X	X	X

实例

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：

保存调用功能块的返回地址和 DB10 以及调用功能块的实例数据块的选择数据。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

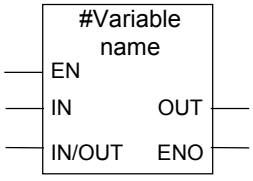
ENO = 0　Q4.0 = 0

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 调用多重实例

符号



参数	数据类型	存储器区	描述
EN	BOOL	I、Q、M、D、L、T、C	启用输入
ENO	BOOL	I、Q、M、D、L	启用输出
# 变量名	FB/SFB	-	多重实例的名称

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实例。在程序目录中只列出了已经声明的多重实例。多重实例的符号改变取决于是否存在参数以及存在多少个参数。始终显示 EN、ENO 和变量名。

状态字

	BR	CC1	CC0	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 Manager 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)")
- 由于要多次使用而自行在库中保存的块。

10.9 主控继电器指令

关于使用 MCR 功能的重要注意事项

主控继电器的定义

主控继电器(MCR，参见主控制继电器开/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 : 置位输出
- SR ：置位复位触发器
- RS ：复位置位触发器
- MOVE : 分配值

取决于 MCR 的指令和它们对 MCR 信号状态的响应

MCR 的信号状态	赋值、 中间输出	置位或复位 输出	移动
0 ("OFF")	写入 0。 (模拟继电器在移除电压后进入其静止状态。)	不写入。 (模拟继电器在移除电压后保持其当前状态。)	写入 0。 (模拟组件在移除电压后生成一个 0 值。)
1 ("ON")	正常处理	正常处理	正常处理

10.10 关于使用 MCR 功能的重要注意事项

请小心使用那些已用 MCRA 激活主控继电器的块

- 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。这对包含赋值的**所有**框都有效，包括传送到块的参数在内。
- 当主控继电器打开指令前的 RLO= 0 时，则 MCR 去活。

危险：PLC 处于 STOP 状态或未定义的运行特征！

编译器也使用在 VAR_TEMP 中为计算地址而定义的临时变量，对本地数据进行写访问。这意味着，下列命令序列将把 PLC 设为 STOP 模式，或导致未定义的运行特性：

形式参数访问

- 访问 STRUCT、UDT、ARRAY、STRING 类型的复杂 FC 参数的构成成分。
- 访问具有多重背景能力的块(版本 2 型块)中 IN_OUT 区域的 STRUCT、UDT、ARRAY、STRING 类型的复杂 FC 参数的构成成分。
- 如果其地址大于 8180.0，访问具有多重背景功能的功能块(版本 2 型块)的参数。
- 访问具有多重背景功能的功能块(版本 2 型块)的 BLOCK_DB 类型的参数，打开 DB0。其后任何数据访问都将会把 CPU 设为 STOP 模式。T 0、C 0、FC0 或 FB0 也将始终用于 TIMER、COUNTER、BLOCK_FC 和 BLOCK_FB。

参数传递

- 调用可传递参数的功能。

LAD/FBD

- 梯形图中的 T 分支和中线输出或 FBD 以 RLO = 0 开始。

纠正方法

解除上述命令的 MCR 依存关系：

1st 在所述语句或程序段**之前**，使用主控制继电器取消激活指令，**取消激活**主控制继电器。

2nd 在所述语句或程序段之后，使用主控制继电器激活指令，**重新激活**主控制继电器。

10.11 MCR</MCR>：主控制继电器打开/关

关于使用 MCR 功能的重要注意事项

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

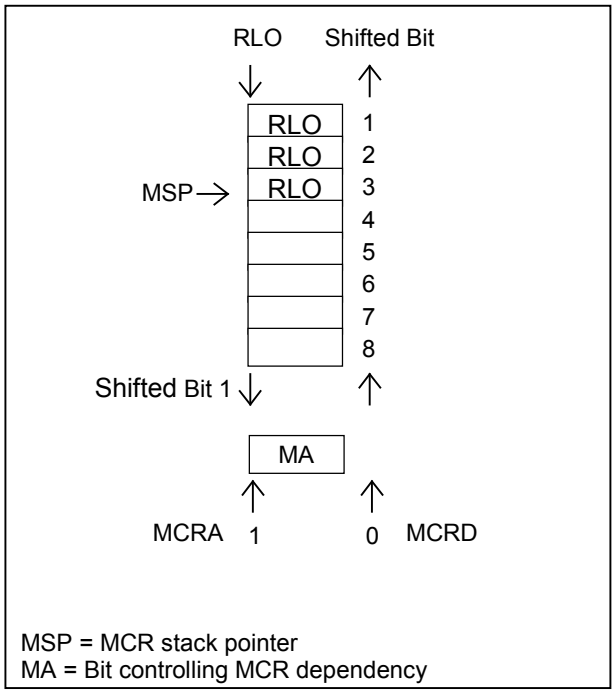
MCR 关闭

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。

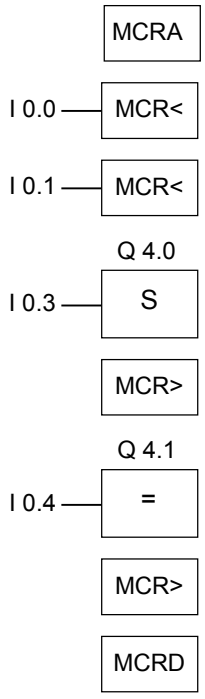
当嵌套了 MCR<指令时，形成较低嵌套级别的 MCR 位。然后，MCR<指令根据与运算真值表组合当前 RLO 与当前的 MCR 位。

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	CC1	CC0	OV	OS	OR	STA	RLO	FC
写	-	-	-	-	-	0	1	-	0

实例



当 **MCRA** 指令激活 MCR 功能时，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：主控制继电器激活/取消激活

关于使用 MCR 功能的重要注意事项

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(参见主控制继电器开/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)。

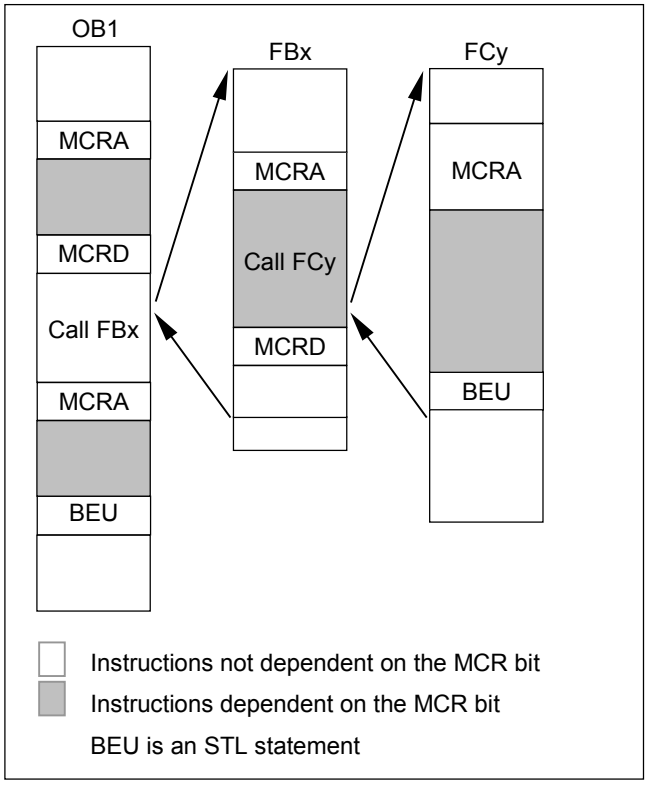
在块与块之间传送 MCR 堆栈，且在每次调用块时，保存 MA 位并将其设置成 0。在块结束处重新读取 MA 位。

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

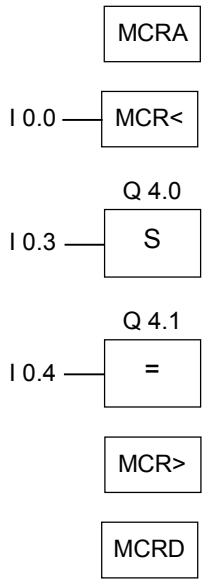
在 MCRA 和 MCRD 之间编程的操作取决于 MCR 位的信号状态。在 MCRA-MCRD 序列外进行编程的操作不依赖于 MCR 位的信号状态。如果丢失 MCRD 指令，那么在 MCRA 和 BEU 指令之间编程的操作取决于 MCR 位。



状态字

	BR	CC1	CC0	OV	OS	OR	STA	RLO	FC
写	-	-	-	-	-	-	-	-	-

实例



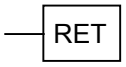
MCRA 指令激活 MCR 功能，直到下一个 MCRD 指令执行。根据 MA 位(在此为 I0.0)处理 MCR<和 MCR>之间的指令：

- 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

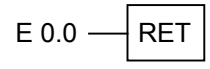
可使用**返回**指令来退出块。可以有条件地退出块。

状态字

	BR	CC1	CC0	OV	OS	OR	STA	RLO	FC
写	*	-	-	-	0	0	1	1	0

* 操作 **RET** 在内部以"SAVE; BEC,"序列显示。这还影响 BR 位。

实例



当输入 I0.0 的信号状态为 1 时，会退出块。

11 移位和循环指令

11.1 移位指令

11.1.1 移位指令概述

描述

可使用移位指令向左或向右逐位移动输入 IN 的内容(另请参阅 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。

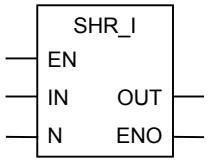
您提供给输入参数 N 的数值决定了移动相应值的位数。移位指令产生的空位将用零或符号位的信号状态(0 表示正，1 表示负)来填补。最后移动的位的信号状态将装入状态字的 CC1 位中(请参见"CPU 寄存器")。状态字的 CC0 和 OV 位将复位为 0。您可以使用跳转指令判断 CC1 位。

下列移位指令可用：

- SHR_I：整数右移
- SHR_DI：长整数右移
- SHL_W：字左移
- SHR_W：字右移
- SHL_DW：双字左移
- SHR_DW：双字右移

11.1.2 SHR_I：整数右移

符号



参数	数据类型	存储器区	描述
EN	BOOL	I、Q、M、L、D、T、C	启用输入
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	启用输出

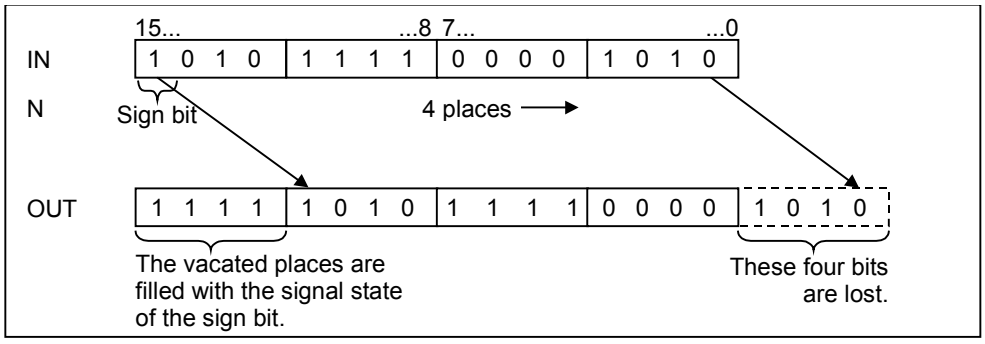
描述

在启用输入(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 CC0 和 OV 位复位为 0。

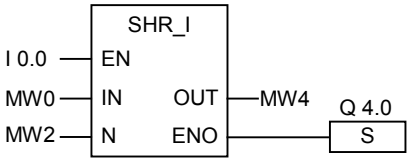
ENO 与 EN 具有相同的信号状态。

状态字



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

实例

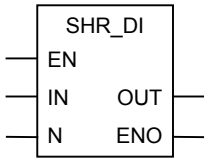


在 I0.0 的信号状态为 1 时会激活该指令。存储器字 MW0 将按在存储器字 MW2 中指定的位数右移。

结果将存入存储器字 MW4 中。输出 Q4.0 置 1。

11.1.3 SHR_DI：长整数右移

符号



参数	数据类型	存储器区	描述
EN	BOOL	I、Q、M、L、D、T、C	启用输入
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	启用输出

描述

在启用输入(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。

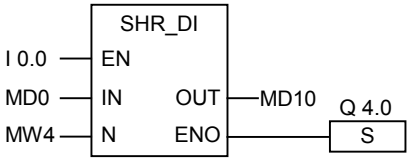
如果 N 不等于 0，则此指令触发的操作总是将状态字的 CC0 和 OV 位复位为 0。

ENO 与 EN 具有相同的信号状态。

状态字

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

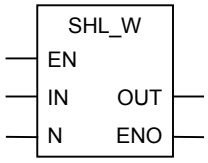
实例



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：字左移

符号



参数	数据类型	存储器区	描述
EN	BOOL	I、Q、M、L、D、T、C	启用输入
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	启用输出

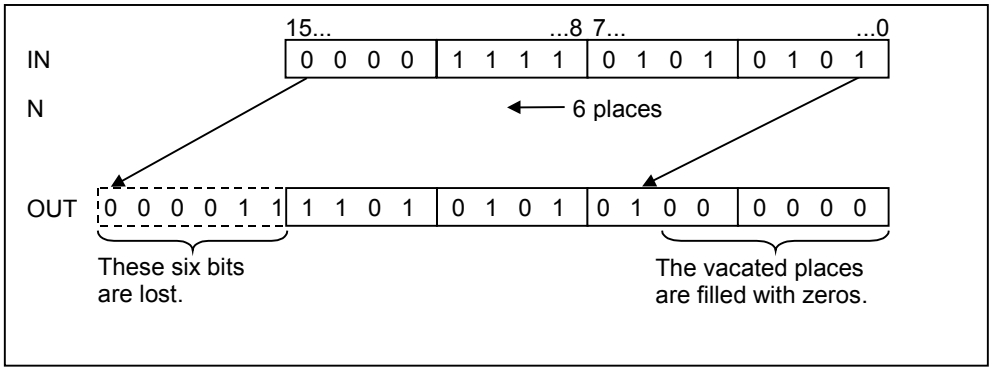
描述

在启用输入(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 CC0 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 CC0 和 OV 位复位为 0。

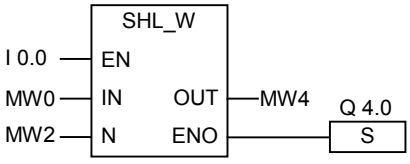
ENO 与 EN 具有相同的信号状态。



状态字

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

实例



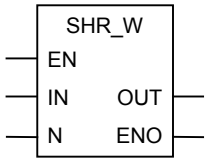
在 I0.0 的信号状态为 1 时会激活该指令。

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：字右移

符号



参数	数据类型	存储器区	描述
EN	BOOL	I、Q、M、L、D、T、C	启用输入
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	启用输出

描述

在启用输入(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 CC0 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。

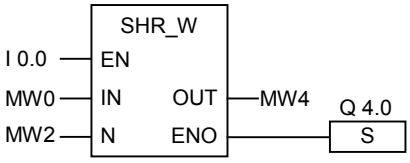
如果 N 不等于 0，则此指令触发的操作总是将状态字的 CC0 和 OV 位复位为 0。

ENO 与 EN 具有相同的信号状态。

状态字

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

实例

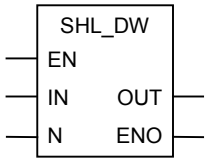


在 I0.0 的信号状态为 1 时会激活该指令。存储器字 MW0 将按在存储器字 MW2 中指定的位数右移。

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	存储器区	描述
EN	BOOL	I、Q、M、L、D、T、C	启用输入
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	启用输出

描述

在启用输入(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 CC0 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。

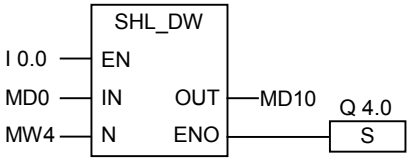
如果 N 的值不等于 0，则此指令触发的操作总是将状态字的 CC0 和 OV 位复位为 0。

ENO 与 EN 具有相同的信号状态。

状态字

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

实例



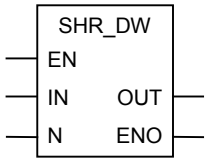
在 I0.0 的信号状态为 1 时会激活该指令。

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：双字右移

符号



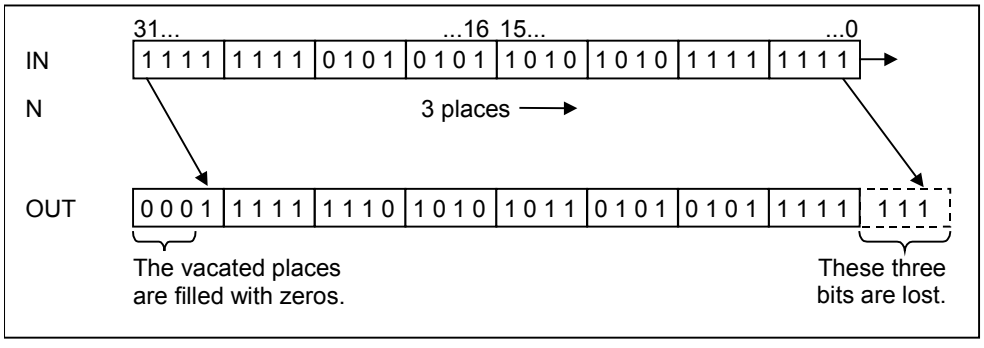
参数	数据类型	存储器区	描述
EN	BOOL	I、Q、M、L、D、T、C	启用输入
IN	DWORD	I、Q、M、L、D	要移位的值
N	WORD	I、Q、M、L、D	值将移位的位数
OUT	DWORD	I、Q、M、L、D	移位指令的结果
ENO	BOOL	I、Q、M、L、D	启用输出

描述

在启用输入(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 CC0 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 CC0 和 OV 位复位为 0。

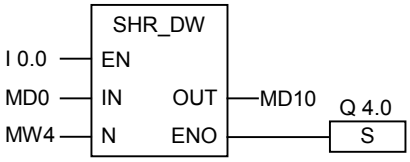
ENO 与 EN 具有相同的信号状态。



状态字

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

实例



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。

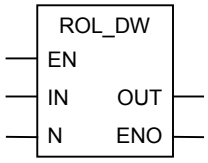
根据指令不同，循环移位将使用状态字的 CC1 位。将状态字的 CC0 位复位为 0。

下列循环移位指令可用：

- ROL_DW：双字循环左移
- ROR_DW：双字循环右移

11.2.2 ROL_DW：双字循环左移

符号



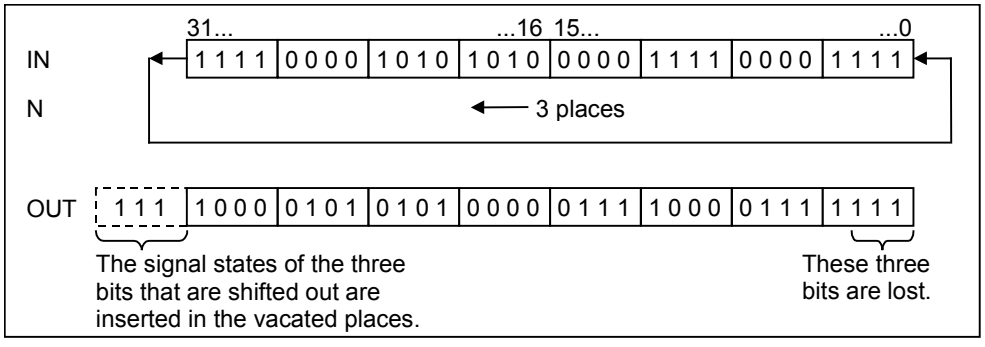
参数	数据类型	存储器区	描述
EN	BOOL	I、Q、M、L、D、T、C	启用输入
IN	DWORD	I、Q、M、L、D	要循环移位的值
N	WORD	I、Q、M、L、D	值将循环移动的位数
OUT	DWORD	I、Q、M、L、D	循环指令的结果
ENO	BOOL	I、Q、M、L、D	启用输出

描述

在启用输入(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 CC0 和 OV 位复位为 0。

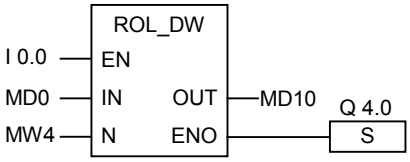
ENO 与 EN 具有相同的信号状态。



状态字

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

实例

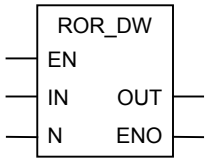


在 I0.0 的信号状态为 1 时激活此指令。存储器双字 MD0 将按在存储器字 MW4 中指定的位数循环右移。

结果将存入存储器双字 MD10 中。输出 Q4.0 置 1。

11.2.3 ROR_DW：双字循环右移

符号



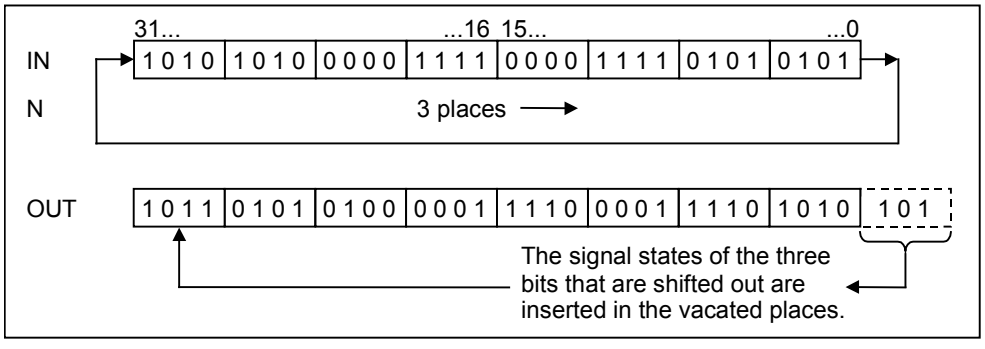
参数	数据类型	存储器区	描述
EN	BOOL	I、Q、M、L、D、T、C	启用输入
IN	DWORD	I、Q、M、L、D	要循环移位的值
N	WORD	I、Q、M、L、D	值将循环移动的位数
OUT	DWORD	I、Q、M、L、D	循环指令的结果
ENO	BOOL	I、Q、M、L、D	启用输出

描述

在启用输入(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 CC0 和 OV 位复位为 0。

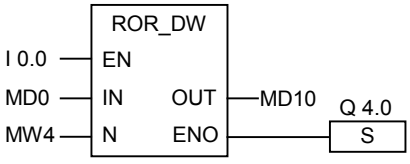
ENO 与 EN 具有相同的信号状态。



状态字

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

实例



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(参见 CPU 寄存器)。各状态位指令分别对下列条件之一做出反应，其中每个条件以状态字的一个或多个位来表示：

- 二进制结果位(BR)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。
- 运算的结果无序(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"与"真值表将其信号状态校验的结果与前一逻辑运算结果合并。当状态位指令以并联方式连接时，该指令将根据"或"真值表将其结果与前一 RLO 合并。

状态字

状态字是 CPU 存储器中的一个寄存器，它包含可以在位逻辑指令和字逻辑指令的地址中引用的位。状态字的结构：

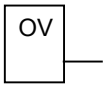
2^{15}... ...2^9	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
	BR	CC1	CC0	OV	OS	OR	STA	RLO	/FC

可以通过下列函数求状态字位的值

- 整数数学运算函数
- 浮点数运算函数

12.2 OV：异常位溢出

符号



描述

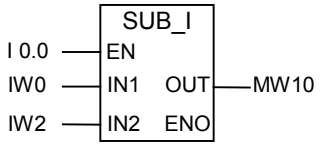
可使用**溢出异常位**指令来检测最后一个算术运算是否有溢出(OV)。如果在系统执行算术运算后，结果超出了许可的负数范围或许可的正数范围，则会设置状态字(参见 CPU 寄存器)的 OV 位。指令将检查此位的信号状态。如果算术运算未出错，则复位该位。

状态字

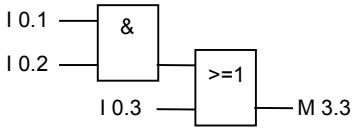
	BR	CC1	CC0	OV	OS	OR	STA	RLO	FC
写	-	-	-	-	-	X	X	X	1

实例

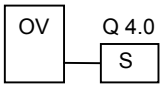
程序段 1



程序段 2



程序段 3

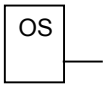


在输入 I0.0 的信号状态为 1 时会激活 SUB_I 框。如果算术运算 (输入字 IW0 减输入字 IW2)的结果超出了整数值的许可范围，则会置位状态字的 OV 位。OV 位的信号检查结果为 1。如果 OV 位的检查结果为 1 且程序段 2 的 RLO 为 1(如果输出 Q4.0 之前的 RLO 为 1)，则会置位输出 Q4.0。

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：存储的异常位溢出

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。

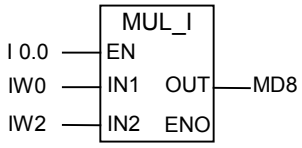
可使用**存储溢出异常位**指令判断先前算术运算结果是否有溢出(存储溢出，OS)。如果在系统执行数学运算后，结果超出了许可的负数值范围或者正数值范围，将会设置状态字的 OS 位(参见 CPU 寄存器)。指令将检查此位的信号状态。与 OV(溢出)位不同，即使在执行随后的算术运算时未出现错误，OS 位仍将保留设置(参见溢出异常位)。

状态字

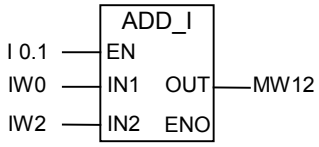
	BR	CC1	CC0	OV	OS	OR	STA	RLO	FC
写	-	-	-	-	-	X	X	X	1

实例

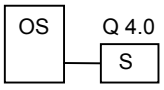
程序段 1



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。在输入端 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。

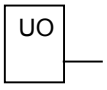
OS 位的信号状态检查的结果为 1 且置位输出 Q4.0。

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：无序的异常位

符号



描述

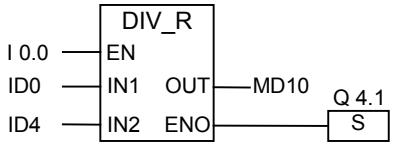
可使用**无序异常位**指令检查浮点数运算的结果是否无序(换言之，该运算中是否存在一个无效的浮点数)。指令将判断状态字的条件码位(CC 1 和 CC 0，参见 CPU 寄存器)。如果运算的结果是无序(UO)，信号状态检查的结果将是 1。如果在 CC 1 和 CC 0 中的组合结果表明不是无序的，则信号状态检查的结果将是 0。

状态字

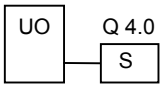
	BR	CC1	CC0	OV	OS	OR	STA	RLO	FC
写	-	-	-	-	-	X	X	X	1

实例

程序段 1



程序段 2



在输入端 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

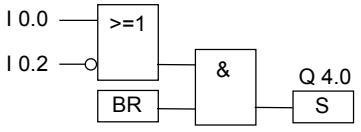
描述

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(参见 CPU 寄存器)。

状态字

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

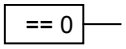
实例



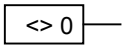
如果输入 I0.0 的信号状态为 1，或输入 I0.2 的信号状态为 0，将会置位输出 Q4.0；除了此逻辑运算结果外，BR 位的信号状态为 1 时也会如此。

12.6 <> 0：结果位

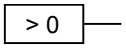
符号



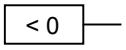
结果位指令**等于 0** 用于确定算术运算指令的结果是否等于 0。



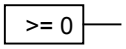
结果位指令**不等于 0** 用于确定算术运算指令的结果是否不等于 0。



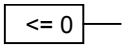
结果位指令**大于 0** 用于确定算术运算指令的结果是否大于 0。



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。



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。



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

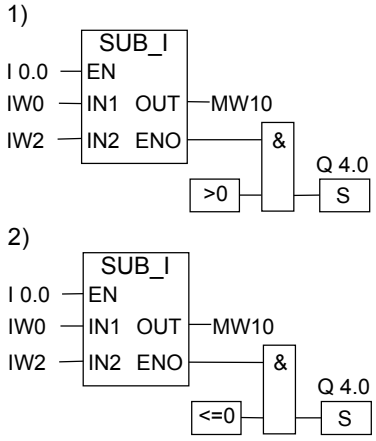
你可以使用**结果位**指令来确定运算的结果与 0 之间的关系，换言之，确定结果是否是 ==0、<>0、>0、<0、>=0 或 <=0。指令将判断状态字的条件码位(CC 1 和 CC 0，参见 CPU 寄存器)。如果满足地址中指示的比较条件，则信号状态检查的结果为 1。

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	CC1	CC0	OV	OS	OR	STA	RLO	FC
写	-	-	-	-	-	X	X	X	1

实例



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)，但在指令执行期间出错，则 ENO 的信号状态为 0。

- 如果正确执行了该运算且结果小于或等于 0，则会将输出 Q4.0 置位。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。
- 如果正确执行了该运算且结果小于或等于 0，则会将输出 Q4.0 置位。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。

13 定时器指令

13.1 定时器指令总览

描述

可在"定时器在存储器中的位置和定时器组件"中找到关于设置和选择正确时间的信息。

提供以下定时器指令：

- S_PULSE : 分配脉冲定时器参数和启动
- S_PEXT： 分配延长脉冲定时器参数和启动
- S_ODT： 分配接通延迟定时器参数和启动
- S_ODTS : 分配保持接通延迟定时器参数和启动
- S_OFFDT : 分配关闭延迟定时器参数和启动
- SP : 启动脉冲定时器
- SE : 启动延长脉冲定时器
- SD : 启动接通延迟定时器
- SS : 启动保持接通延迟定时器
- SF : 启动关闭延迟定时器

13.2 定时器的存储区和组件

存储器区

CPU 的存储器中为时间保留了一个区域。该存储器区为每个定时器地址保留一个 16 位的字。当在 FBD 中编程时，支持 256 个定时器。请参考 CPU 的技术信息，查看可用的定时器字的数目。

下列功能可访问定时器存储区：

- 定时器指令
- 通过时钟定时更新定时器字。在 RUN 模式中，该 CPU 功能以由时间基准指定的间隔，将给定的时间值减小一个单位，直到时间值变为零。

时间值

定时器字的位 0 到 9 包含二进制编码的时间值。时间值指定多个单位。当定时器更新时，时间值以由时间基准指定的间隔，减小一个单位。时间值将一直减小到零。

可使用以下格式之一预先加载时间值：

- S5T#a**H**_b**M**_c**S**_d**MS**
  - 其中，H = 小时，M = 分钟，S = 秒钟，MS = 毫秒；
    a、b、c、d 由用户定义。
  - 自动选择时间基准，其值舍入为具有该时间基准的下一个较小的数字。

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
S5T#1H_12M_18S = 1 小时，12 分，18 秒

时间基准

定时器字的位 12 和 13 包含二进制编码的时间基准。时间基准定义时间值减小一个单位的间隔。最小时间基准为 10 ms；最大为 10 s。

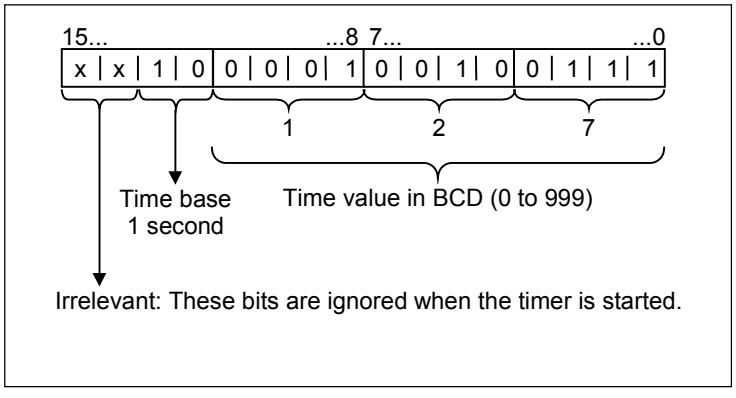
时间基准	时间基准的二进制编码
10ms	00
100ms	01
1 s	10
10 s	11

由于时间值仅以一个时间间隔保存，不能被时间间隔整除的余数值将被丢弃。如果对于所需的范围，值的分辨率太高，则会根据所需的范围进行舍入，而不是根据所需的分辨率进行舍入。下表给出了可能的分辨率和相应的范围。

分辨率	时间基准
0.01 秒钟	10MS 至 9S_990MS
0.1 秒钟	100MS 至 1M_39S_900MS
1 秒	1S 至 16M_39S
10 秒	10S 至 2HR_46M_30S

定时器单元的位组态

当定时器启动后，定时器单元中的内容将作为时间值。定时器单元的位 0 到 11 包含二进制编码的十进制格式(BCD 格式：每组四位，包含一个十进制值的二进制代码)的时间值。位 12 和 13 包含二进制编码的时间基准。下图给出了装载了时间值 127，时间基准为 1 秒的定时器单元的内容。

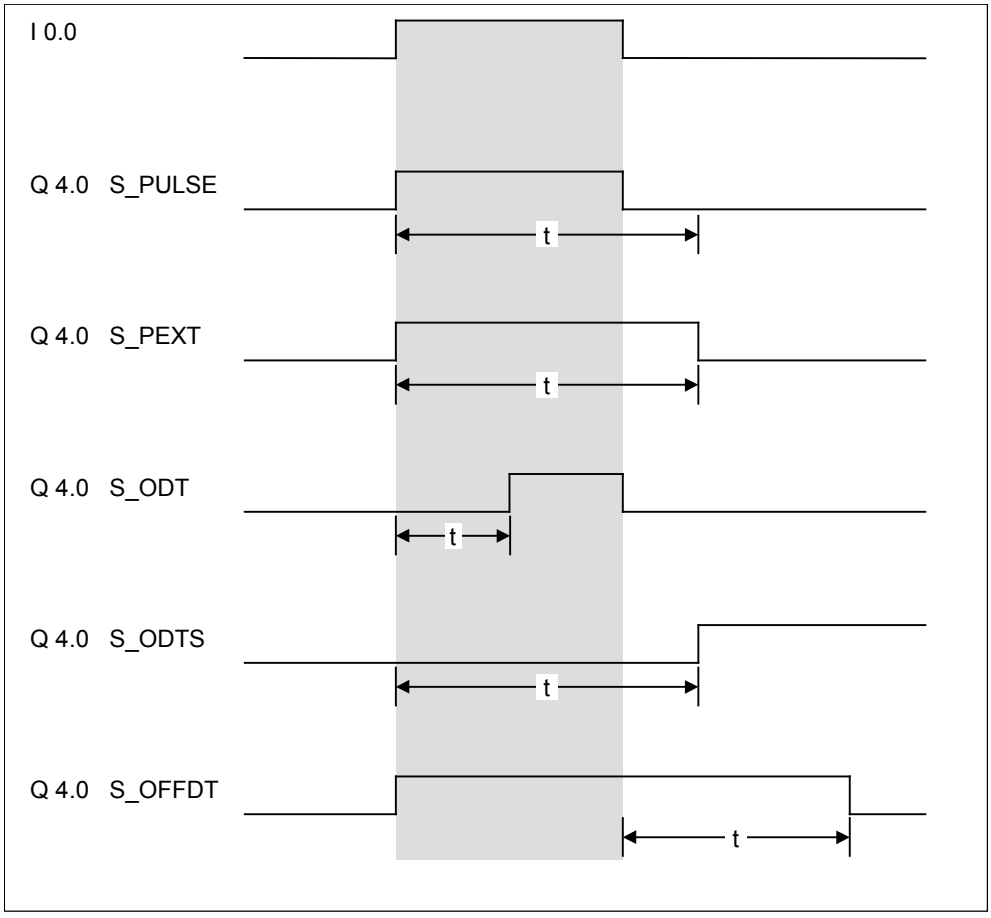


读取时间和时间基准

每个定时器方框提供两个输出，分别为 BI 和 BCD，可为其指定字位置。BI 输出提供二进制格式的时间值，时间基准则未显示。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。

选择正确的定时器

该总览旨在帮您为定时任务选择正确的定时器。

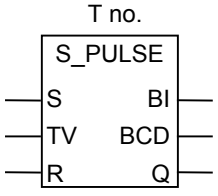


定时器	描述
S_PULSE 脉冲定时器	输出信号保持在 1 的最长时间与编程时间值 t 相同。如果输入信号变为 0，则输出信号停留在 1 的时间会很短。
S_PEXT 延长脉冲定时器	输出信号在编程时间长度内始终保持在 1，而与输入信号停留在 1 的时间长短无关。
S_ODT 接通延迟定时器	仅在编程时间到期，且输入信号仍为 1 时，输出信号变为 1。
S_ODTS 保持接通延迟定时器	输出信号仅在编程时间到期时才从 0 变为 1，而与输入信号停留在 1 的时间长短无关。
S_OFFDT 关闭延迟定时器	输出信号在输入信号变为 1 时或当定时器在运行时变为 1。当输入信号从 1 变为 0 是启动时间。

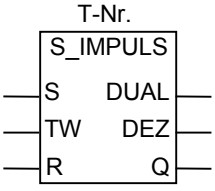
13.3 S_PULSE：分配脉冲定时器参数和启动

符号

English



German



参数 英语	参数 德语	数据类型	存储器区	描述
编号	编号	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
BI	DUAL	WORD	I、Q、M、D、L	剩余时间 (值为整数格式)
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

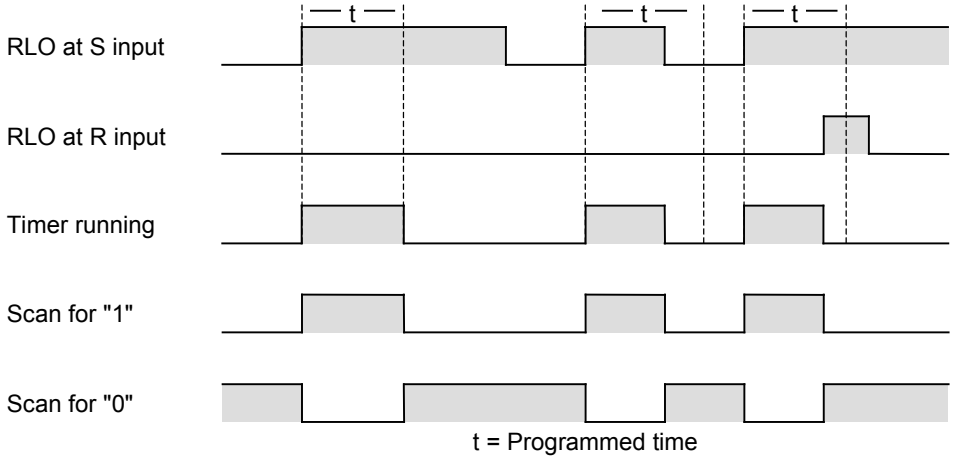
分配脉冲定时器参数和启动指令在启动(S)输入发生上升沿(信号状态从 0 变为 1)时启动指定的定时器。信号改变始终是启动定时器的必要条件。只要输入 TV 信号状态为 1，定时器就将继续按照时间值(TV)输入指定的时间运行，直到达到编定的时间。在定时器运行期间，输出 Q 的检查信号状态 1 将产生结果 1。如果在时间结束前，S 输入的信号状态从 1 变为 0，则将停止定时器。然后输出 Q 的检查信号状态 1 将产生结果 0。

在定时器运行期间，定时器复位(R)输入的从 0 变为 1，则将复位定时器。这一改变还将把时间和时间基准复位为零。如果定时器未运行，则定时器 R 输入的信号状态 1 无效。

当前时间值可从输出 BI 和 BCD 扫描得到。BI 时间值为二进制格式；BCD 时间值则为二进制编码的十进制格式。

时序图

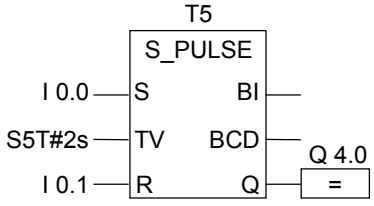
脉冲定时器特性：



状态字

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

实例

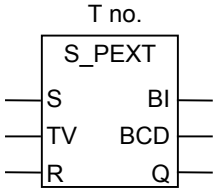


如果输入 I0.0 的信号状态从 0 变为 1 (RLO 的上升沿)，则将启动定时器 T5。只要输入 I0.0 为 1，定时器将继续运行指定时间两秒(2s)。如果在该时间之内，输入 I0.0 的信号状态从 1 变为 0，则将停止定时器。如果在定时器运行期间，输入 I0.1 的信号状态从 0 变为 1，则将复位定时器。在定时器运行期间，输出 Q4.0 的信号状态始终为 1。

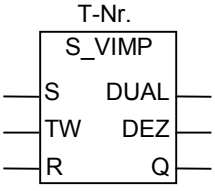
13.4 S_PEXT：分配延长脉冲定时器参数和启动

符号

English



German



参数 英语	参数 德语	数据类型	存储器区	描述
编号	编号	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
BI	DUAL	WORD	I、Q、M、D、L	剩余时间 (值为整数格式)
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

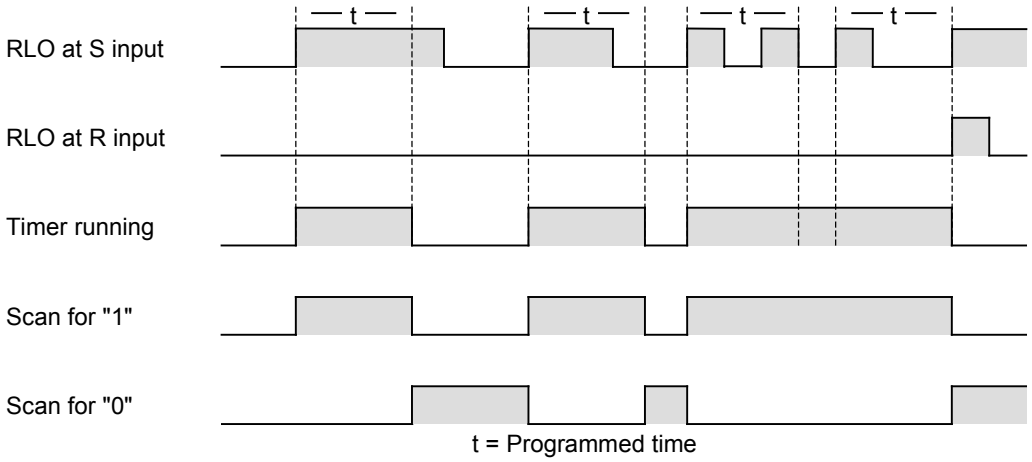
分配延长脉冲定时器参数和启动指令在启动(S)输入发生上升沿(信号状态从 0 变为 1)时启动指定的定时器。信号改变始终是启动定时器的必要条件。定时器将继续按照时间值(TV)输入指定的时间运行，即使在该时间结束之前，S 输入的信号状态已变为 0。在定时器运行期间，输出 Q 的检查信号状态 1 将产生结果 1。如果在定时器运行期间，输入 S 的信号状态从 0 变为 1，则将在指定时间内重新启动定时器。

在定时器运行期间，定时器复位(R)输入的信号状态从 0 变为 1，则将复位定时器。这一改变还将把时间和时间基准复位为零。

当前时间值可从输出 BI 和 BCD 扫描得到。BI 时间值为二进制格式；BCD 时间值则为二进制编码的十进制格式。

时序图

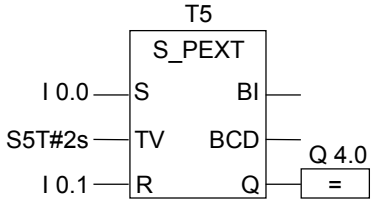
延长脉冲定时器特性：



状态字

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

实例

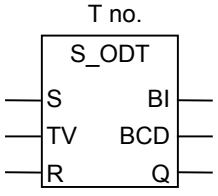


如果输入 I0.0 的信号状态从 0 变为 1 (RLO 的上升沿)，则将启动定时器 T5。定时器继续运行指定时间两秒(2s)，而无论输入 S 是否出现下降沿。如果在该时间之内，输入 I0.0 的信号状态从 0 变为 1，则将重新启动定时器。如果在定时器运行期间，输入 I0.1 的信号状态从 0 变为 1，则将复位定时器。在定时器运行期间，输出 Q4.0 的信号状态始终为 1。

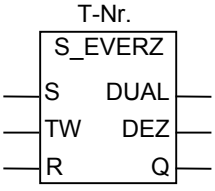
13.5 S_ODT：分配接通延迟定时器参数和启动

符号

English



German



参数 英语	参数 德语	数据类型	存储器区	描述
编号	编号	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
BI	DUAL	WORD	I、Q、M、D、L	剩余时间 (值为整数格式)
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

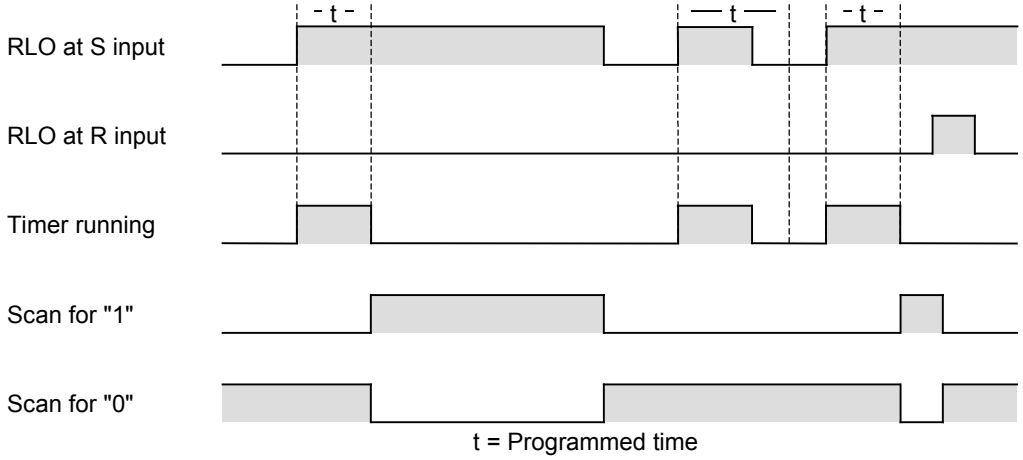
分配接通延迟定时器参数和启动指令在启动(S)输入发生上升沿(信号状态从 0 变为 1)时启动指定的定时器。信号改变始终是启动定时器的必要条件。只要输入 S 的信号状态为 1，定时器就将继续按照时间值(TV)输入指定的时间运行。当时间到期，未发生错误，而输入 S 的信号状态仍为 1 时，输出 Q 的检查信号状态 1 将产生结果 1。如果在定时器运行期间，输入 S 的信号状态从 1 变为 0，则将停止定时器。在这种情况下，输出 Q 的检查信号状态 1 将始终产生结果 0。

在定时器运行期间，定时器复位(R)输入的信号状态从 0 变为 1，则将复位定时器。这一改变还将把时间和时间基准复位为零。如果定时器未运行，而定时器 R 输入的信号状态为 1，则定时器也将复位。

当前时间值可从输出 BI 和 BCD 扫描得到。BI 时间值为二进制格式；BCD 时间值则为二进制编码的十进制格式。

时序图

接通延迟定时器特性：

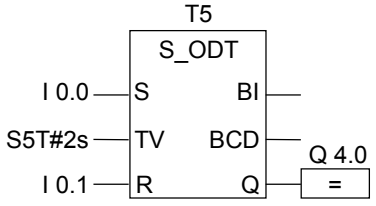


t = Programmed time

状态字

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

实例

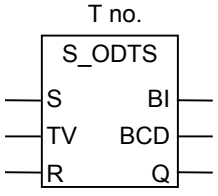


如果输入 I0.0 的信号状态从 0 变为 1 (RLO 的上升沿)，则将启动定时器 T5。如果指定时间两秒(2s)结束后，输入 I0.0 的信号状态仍旧为 1，则输出 Q4.0 的信号状态将为 1。如果输入 I0.0 的信号状态从 1 变为 0，则将停止定时器，输出 Q4.0 变为 0。如果在定时器运行期间，输入 I0.1 的信号状态从 0 变为 1，则将重新启动定时器。

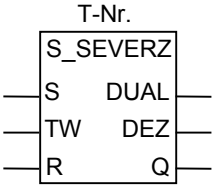
13.6 S_ODTS：分配保持接通延迟定时器参数和启动

符号

English



German



参数 英语	参数 德语	数据类型	存储器区	描述
编号	编号	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
BI	DUAL	WORD	I、Q、M、D、L	剩余时间 (值为整数格式)
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

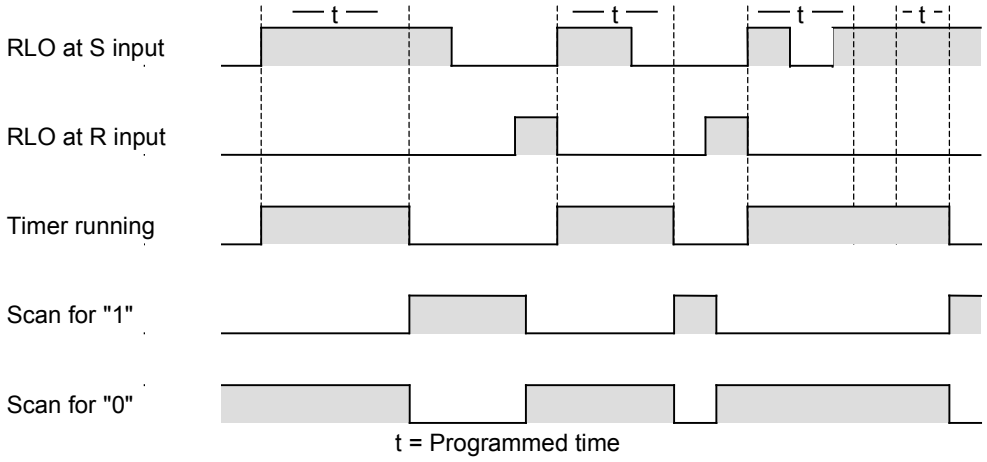
分配保持接通延迟定时器参数和启动指令在启动(S)输入发生上升沿(信号状态从 0 变为 1)时启动指定的定时器。信号改变始终是启动定时器的必要条件。定时器将继续按照时间值(TV)输入指定的时间运行，即使在该时间结束之前，S 输入的信号状态已变为 0。当时间到期，输出 Q 的检查信号状态 1 将产生结果 1，无论在复位输入(R)保持为 0 时，输入 S 的信号状态如何。如果在定时器运行期间，输入 S 的信号状态从 0 变为 1，则将在指定时间内重新启动定时器。

定时器复位(R)输入从 0 变为 1，将复位定时器，无论 S 输入的 RLO 为何值。

当前时间值可从输出 BI 和 BCD 扫描得到。BI 时间值为二进制格式；BCD 时间值则为二进制编码的十进制格式。

时序图

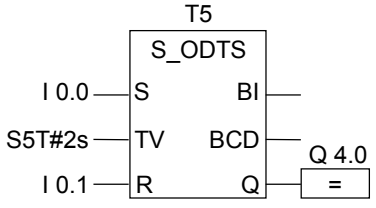
保持接通延迟定时器特性：



状态字

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

实例

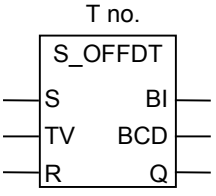


如果输入 I0.0 的信号状态从 0 变为 1 (RLO 的上升沿)，则将启动定时器 T5。定时器继续运行，无论输入 I0.0 的信号是否从 1 变为 0。如果在该时间之内，输入 I0.0 的信号状态从 0 变为 1，则将重新启动定时器。如果在定时器运行期间，输入 I0.1 的信号状态从 0 变为 1，则将重新启动定时器。如果时间到期时，I0.1 的信号状态保持为 0，则输出 Q4.0 的信号状态为 1。

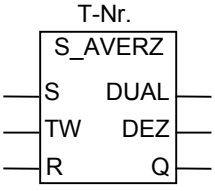
13.7 S_OFFDT：分配关闭延迟定时器参数和启动

符号

English



German



参数 英语	参数 德语	数据类型	存储器区	描述
编号	编号	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
BI	DUAL	WORD	I、Q、M、D、L	剩余时间 (值为整数格式)
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

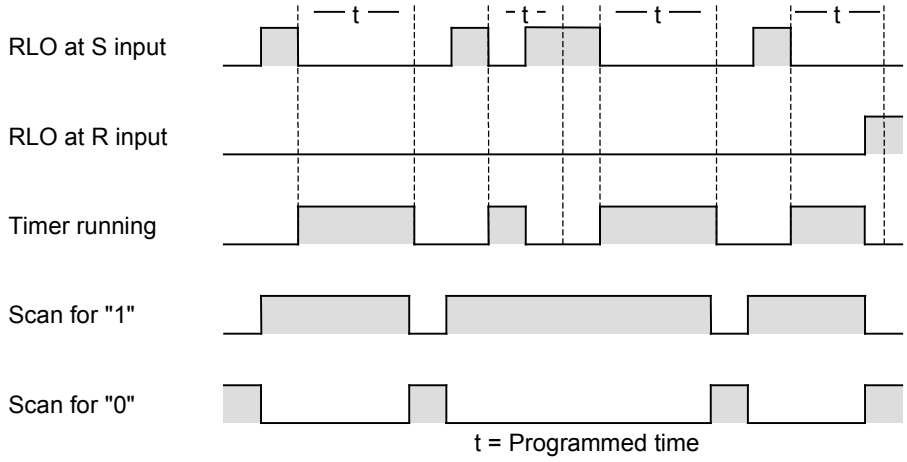
分配关闭延迟定时器参数和启动指令在启动(S)输入发生下降沿(信号状态从 1 变为 0)时启动指定的定时器。信号改变始终是启动定时器的必要条件。当输入 S 的信号状态为 1 时或当定时器正在运行时，输出 Q 的检查信号状态 1 将产生结果 1。如果在定时器运行期间，输入 S 的信号状态从 0 变为 1，则将复位定时器。定时器将一直等到输入 S 的信号状态从 1 变为 0 之后才会重新启动。

在定时器运行期间，定时器复位(R)输入的信号状态从 0 变为 1，则将复位定时器。

实际时间值可从输出 BI 和 BCD 扫描得到。BI 时间值为二进制格式；BCD 时间值则为二进制编码的十进制格式。

时序图

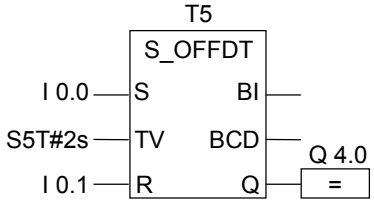
关闭延迟定时器特性：



状态字

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

实例

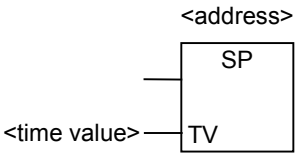


如果输入 I0.0 的信号状态从 1 变为 0，则将启动定时器。当 I0.0 为 1 或定时器正在运行时，输出 Q4.0 为 1。如果在定时器运行期间，I0.1 的信号状态从 0 变为 1，则将复位定时器。

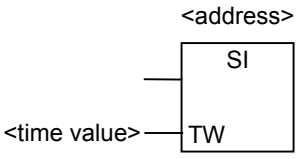
13.8 SP：启动脉冲定时器

符号

English



German



参数 英语	参数 德语	数据类型	存储器区	描述
定时器编号	定时器编号	TIMER	T	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	时间值(S5TIME 格式)

描述

启动脉冲定时器指令在 RLO 发生上升沿(信号状态从 0 变为 1)时启动指定的定时器。只要 RLO 为正，定时器就将继续以指定值运行。在定时器运行期间，检查信号状态 1 的结果为 1。如果在定时器时间到期之前，RLO 从 1 变为 0，则将停止定时器。在这种情况下，检查信号状态 1 的结果为 0。

可在存储区中找到关于定时器的存储区和组件的更多信息。

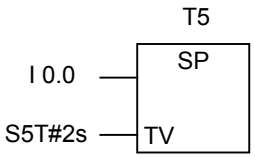
只能将方框**启动脉冲定时器**放置在逻辑串的右侧末端。可使用多个**启动脉冲定时器**方框。

状态字

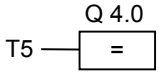
	BR	CC1	CC0	OV	OS	OR	STA	RLO	FC
写	-	-	-	-	-	0	-	-	0

实例

程序段 1



程序段 2

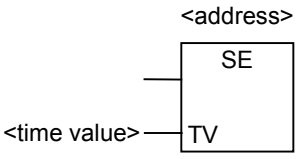


如果输入 I0.0 的信号状态从 0 变为 1 (RLO 的上升沿)，则将启动定时器 T5。只要信号状态为 1，定时器将继续运行指定时间两秒(2s)。如果在定时器运行期间，I0.0 的信号状态从 1 变为 0，则将停止定时器。
在定时器运行期间，输出 Q4.0 为 1。

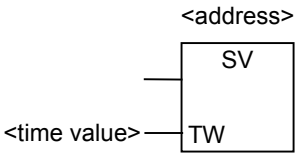
13.9 SE：启动延长脉冲定时器

符号

English



German



参数 英语	参数 德语	数据类型	存储器区	描述
定时器编号	定时器编号	TIMER	T	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	时间值(S5TIME 格式)

描述

启动延长脉冲定时器指令在 RLO 发生上升沿(信号状态从 0 变为 1)时启动指定的定时器。如果在定时器时间结束之前，RLO 变为 0，则定时器将以指定值继续运行。在定时器运行期间，检查信号状态 1 的结果为 1。如果在定时器运行期间，RLO 从 0 变为 1，则将在指定时间内重新启动定时器。

可在存储区中找到关于定时器的存储区和组件的更多信息。

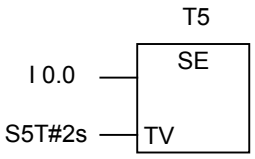
只能将方框**启动延长脉冲定时器**放置在逻辑串的右侧末端。可使用多个**启动延长脉冲定时器**方框。

状态字

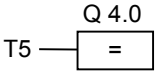
	BR	CC1	CC0	OV	OS	OR	STA	RLO	FC
写	-	-	-	-	-	0	-	-	0

实例

程序段 1



程序段 2

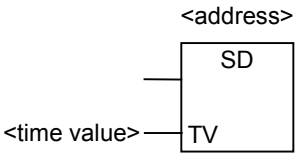


如果输入 I0.0 的信号状态从 0 变为 1 (RLO 的上升沿)，则将启动定时器 T5。定时器将继续运行，不受 RLO 下降沿的影响。如果在指定时间之内，I0.0 的信号状态从 0 变为 1，则将重新启动定时器。在定时器运行期间，输出 Q4.0 为 1。

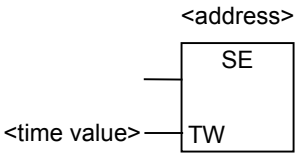
13.10 SD：启动接通延迟定时器

符号

English



German



参数 英语	参数 德语	数据类型	存储器区	描述
定时器编号	定时器编号	TIMER	T	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	时间值(S5TIME 格式)

描述

启动接通延迟定时器指令在 RLO 发生上升沿(信号状态从 0 变为 1)时启动指定的定时器。如果指定时间到期，未发生错误，而 RLO 值仍旧为 1，则检查信号状态 1 的结果为 1。如果在定时器运行期间，RLO 从 1 变为 0，则将停止定时器。在这种情况下，检查信号状态 1 的结果将始终为 0。

可在存储区中找到关于定时器的存储区和组件的更多信息。

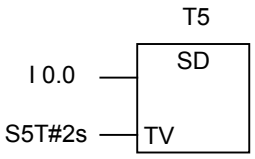
只能将方框**启动接通延迟定时器**放置在逻辑串的右侧末端。可使用多个**启动接通延迟定时器**方框。

状态字

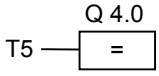
	BR	CC1	CC0	OV	OS	OR	STA	RLO	FC
写	-	-	-	-	-	0	-	-	0

实例

程序段 1



程序段 2

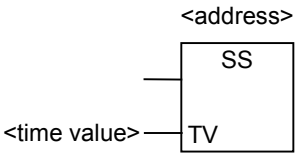


如果输入 I0.0 的信号状态从 0 变为 1 (RLO 的上升沿)，则将启动定时器 T5。如果当定时器时间到期后，I0.0 的信号状态仍旧为 1，则输出 Q4.0 为 1。如果信号状态从 1 变为 0，则将停止定时器。

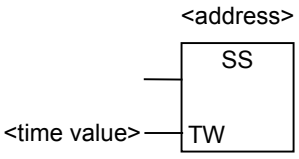
13.11 SS：启动保持接通延迟定时器

符号

English



German



参数 英语	参数 德语	数据类型	存储器区	描述
定时器编号	定时器编号	TIMER	T	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	时间值(S5TIME 格式)

描述

启动保持接通延迟定时器指令在 RLO 发生上升沿(信号状态从 0 变为 1)时启动指定的定时器。如果在定时器时间结束之前，RLO 变为 0，则定时器将以指定值继续运行。如果定时器时间到期，无论 RLO 结果如何，检查信号状态 1 的结果为 1。如果在定时器运行期间，RLO 从 0 变为 1，则将在指定时间内重新启动定时器。

可在存储区中找到关于定时器的存储区和组件的更多信息。

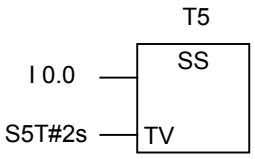
只能将方框**启动保持接通延迟定时器**放置在逻辑串的右侧末端。可使用多个**启动保持接通延迟定时器**方框。

状态字

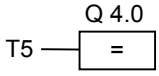
	BR	CC1	CC0	OV	OS	OR	STA	RLO	FC
写	-	-	-	-	-	0	-	-	0

实例

程序段 1



程序段 2

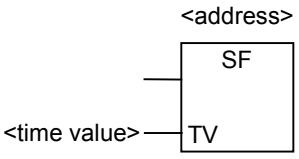


如果输入 I0.0 的信号状态从 0 变为 1 (RLO 的上升沿)，则将启动定时器 T5。定时器继续运行，无论输入 I0.0 的信号状态是否从 1 变为 0。如果在指定时间之内，信号状态从 0 变为 1，则将重新启动定时器。
在定时器时间到期后，输出 Q4.0 为 1。

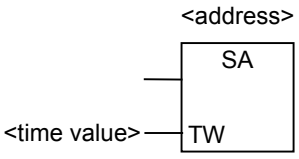
13.12　SF　启动关闭延迟定时器

符号

English



German



参数 英语	参数 德语	数据类型	存储器区	描述
定时器编号	定时器编号	TIMER	T	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	时间值(S5TIME 格式)

描述

启动关闭延迟定时器指令在 RLO 发生下降沿(信号状态从 1 变为 0)时启动指定的定时器。如果 RLO 为 1，或定时器正在运行，则检查信号状态 1 的结果为 1。如果在定时器运行期间，RLO 从 0 变为 1，则将复位定时器。定时器将在 RLO 从 1 变为 0 后重新启动。

可在存储区中找到关于定时器的存储区和组件的更多信息。

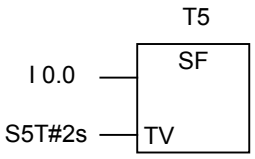
只能将方框**启动关闭延迟定时器**放置在逻辑串的右侧末端。可使用多个**启动关闭延迟定时器**方框。

状态字

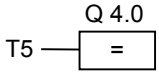
	BR	CC1	CC0	OV	OS	OR	STA	RLO	FC
写	-	-	-	-	-	0	-	-	0

实例

程序段 1



程序段 2



如果输入 I0.0 的信号状态从 1 变为 0，则将启动定时器。

如果信号状态从 0 变为 1，则将复位定时器。

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

13.12 SF　启动关闭延迟定时器

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逐位比较字(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：

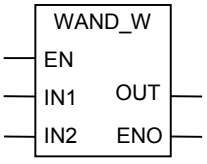
- 如果输出 OUT 的结果不等于 0，则状态字中的 CC1 位会被设置为 1。
- 如果输出 OUT 的结果为 0，则状态字中的 CC1 位会被设置为 0。

要执行字逻辑运算，可使用下列指令：

- WAND_W ：字与(字)
- WOR_W ：字或(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：字与(字)

符号



参数	数据类型	存储器区	描述
EN	BOOL	I、Q、M、D、L、T、C	启用输入
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	启用输出

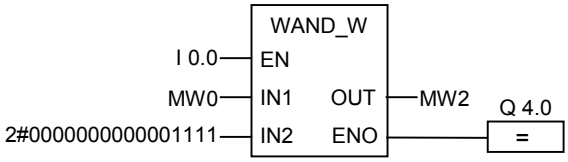
描述

(字)单字与运算指令由位于启用输入(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。按纯位模式来解释这些值。结果可从输出 OUT 处扫描得到。ENO 与 EN 具有相同的信号状态。

状态字

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

实例



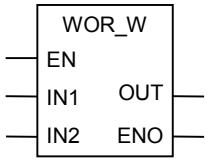
当 I0.0 的信号状态为 1 时会激活该指令。只有 0 到 3 之间的位是相关的 ,所有其它 MW0 位都被屏蔽。

IN1	=	0101010101010101
IN2	=	0000000000001111
OUT	=	0000000000000101

如果执行了该指令，则 Q4.0 为 1。

14.3 WOR_W：字或(字)

符号



参数	数据类型	存储器区	描述
EN	BOOL	I、Q、M、D、L、T、C	启用输入
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	启用输出

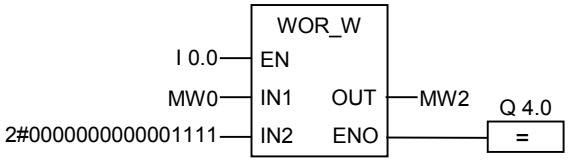
描述

(字)单字或运算指令由位于启用输入(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。按纯位模式来解释这些值。结果可从输出 OUT 处扫描得到。ENO 与 EN 具有相同的信号状态。

状态字

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

实例



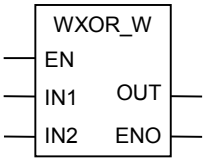
当 I0.0 为 1 时会激活该指令。对 MW0 中的位和常数中的位执行或运算，0 到 3 之间的位被设置为 1，MW0 的所有其它位均以不变形式被输入 MW2 中

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	存储器区	描述
EN	BOOL	I、Q、M、D、L	启用输入
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	启用输出

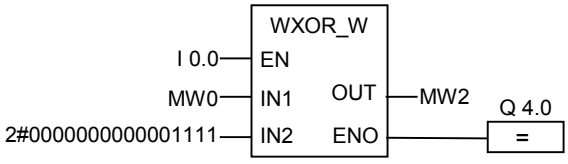
描述

(字)单字或运算指令由位于启用输入(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。按纯位模式来解释这些值。结果可从输出 OUT 处扫描得到。ENO 与 EN 具有相同的信号状态。

状态字

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

实例



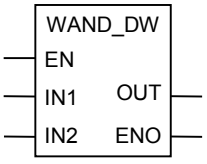
当输入 I0.0 为 1 时，会激活该指令。

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	存储器区	描述
EN	BOOL	I、Q、M、D、L、T、C	启用输入
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	启用输出

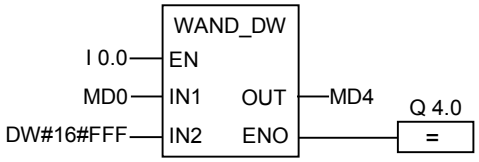
描述

(字)双字与运算指令由位于启用输入(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。按纯位模式来解释这些值。结果可从输出 OUT 处扫描得到。ENO 与 EN 具有相同的信号状态。

状态字

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

实例



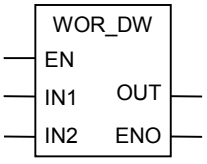
当 I0.0 为 1 时会激活该指令。只有 0 到 11 之间的位是相关的，所有其它 MD4 位都被屏蔽。

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	存储器区	描述
EN	BOOL	I、Q、M、D、L、T、C	启用输入
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	启用输出

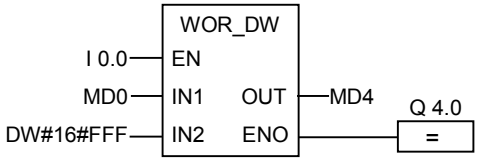
描述

(字)双字或运算指令由位于启用输入(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。按纯位模式来解释这些值。结果可从输出 OUT 处扫描得到。ENO 与 EN 具有相同的信号状态。

状态字

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

实例



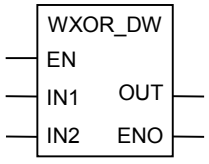
当 I0.0 为 1 时会激活该指令。对 MD0 中的位和常数中的位执行或运算 ,0 到 11 之间的位被设置为 1，MD0 的所有其它位均以不变形式被输入 MD4 中。

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	存储器区	描述
EN	BOOL	I、Q、M、D、L、T、C	启用输入
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	启用输出

描述

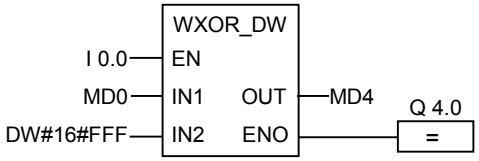
(字)双字异或运算指令由位于启用输入(EN)的信号状态 1 激活，并根据异或运算真值表逐位组合位于输入 IN1 和 IN2 的两个数值。按纯位模式来解释这些值。结果可从输出 OUT 处扫描得到。

ENO 与 EN 具有相同的信号状态。

状态字

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

实例



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。

14.7 WXOR_DW：双字异或运算(字)

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 类别	描述
&	&	位逻辑指令	“与”逻辑运算
>=1	>=1	位逻辑指令	"或"逻辑运算
=	=	位逻辑指令	赋值
#	#	位逻辑指令	中间输出
---|	---|	位逻辑指令	插入二进制输入
---o|	---o|	位逻辑指令	二进制取反输入
==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	生成浮点数的绝对值
ACOS	ACOS	浮点算术运算指令	生成浮点值角度的三角函数
ADD_DI	ADD_DI	整数算术运算指令	加上双精度整数
ADD_I	ADD_I	整数算术运算指令	加上整数
ADD_R	ADD_R	浮点算术运算指令	加上实数
ASIN	ASIN	浮点算术运算指令	生成浮点值角度的三角函数
ATAN	ATAN	浮点算术运算指令	生成浮点值角度的三角函数
BCD_DI	BCD_DI	转换	BCD 码转换为长整型
BCD_I	BCD_I	转换	将 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	调用不带参数的 FC/SFC
CALL_FB	CALL_FB	程序控制	CALL_FB (以框方式调用 FB)
CALL_FC	CALL_FC	程序控制	CALL_FC(以框方式调用 FC)
CALL_SFB	CALL_SFB	程序控制	CALL_SFB (以框方式调用系统 FB)
CALL_SFC	CALL_SFC	程序控制	CALL_SFC (以框方式调用系统 FC)
CEIL	CEIL	转换	向上取整

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 类别	描述
CMP ? D	CMP ? D	比较	比较长整数
CMP ? I	CMP ? I	比较	比较整数
CMP ? R	CMP ? R	比较	比较实数
COS	COS	浮点算术运算指令	生成浮点值角度的三角函数
DI_BCD	DI_BCD	转换	长整型转换为 BCD 码
DI_R	DI_R	转换	长整型转换为实数
DIV_DI	DIV_DI	整数算术运算指令	除以双精度整数
DIV_I	DIV_I	整数算术运算指令	除以整数
DIV_R	DIV_R	浮点算术运算指令	除以实数
EXP	EXP	浮点算术运算指令	生成浮点数的指数值
FLOOR	FLOOR	转换	向下取整
I_BCD	I_BCD	转换	将整型转换为 BCD 码
I_DI	I_DI	转换	整型转换为长整型
INV_I	INV_I	转换	二进制反码整数
INV_DI	INV_DI	转换	二进制反码双精度整数
JMP	JMP	跳转	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	生成浮点数的自然对数
MCR>	MCR>	程序控制	主控制继电器打开/关
MCR<	MCR<	程序控制	主控制继电器打开/关
MCRA	MCRA	程序控制	主控制继电器激活/取消激活
MCRD	MCRD	程序控制	主控制继电器激活/取消激活
MOD_DI	MOD_DI	整数算术运算指令	返回双精度除法的余数
MOVE	MOVE	移动	分配值
MUL_DI	MUL_DI	整数算术运算指令	乘以双精度整数
MUL_I	MUL_I	整数算术运算指令	乘以整数
MUL_R	MUL_R	浮点算术运算指令	乘以实数
N	N	位逻辑指令	RLO 负跳沿检测
NEG	NEG	位逻辑指令	地址下降沿检测
NEG_DI	NEG_DI	转换	二进制补码双精度整数
NEG_I	NEG_I	转换	二进制补码整数
NEG_R	NEG_R	转换	取反实数
OPN	OPN	DB 调用	打开数据块
OS	OS	状态位	存储的异常位溢出
OV	OV	状态位	异常位溢出
P	P	位逻辑指令	正 RLO 边沿检测

德语 助记符	英语 助记符	程序元素 类别	描述
POS	POS	位逻辑指令	地址上升沿检测
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	双字循环左移
ROUND	ROUND	转换	取整到最接近的双精度整数
ROR_DW	ROR_DW	移位/循环	双字循环右移
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	启动关闭延迟定时器
SAVE	SAVE	位逻辑指令	将 RLO 存入 BR 存储区
S_AVERZ	S_OFFDT	定时器	分配关闭延迟定时器参数和启动
SE	SD	定时器	分配接通延迟定时器参数和启动
S_EVERZ	S_ODT	定时器	分配接通延迟定时器参数和启动
SHL_DW	SHL_DW	移位/循环	双字左移
SHL_W	SHL_W	移位/循环	字左移
SHR_DI	SHR_DI	移位/循环	长整数右移
SHR_DW	SHR_DW	移位/循环	双字右移
SHR_I	SHR_I	移位/循环	整数右移
SHR_W	SHR_W	移位/循环	字右移
SI	SP	定时器	启动脉冲定时器
S_IMPULS	S_PULSE	定时器	分配脉冲定时器参数和启动
SIN	SIN	浮点算术运算指令	生成浮点值角度的三角函数
SQR	SQR	浮点算术运算指令	生成浮点数的平方(SQR)
SQRT	SQRT	浮点算术运算指令	生成浮点数的平方根(SQRT)
SR	SR	位逻辑指令	置位复位触发器
SS	SS	定时器	启动保持接通延迟定时器
S_SEVERZ	S_ODTS	定时器	分配保持接通延迟定时器参数和启动
SUB_DI	SUB_DI	整数数学运算指令	减去双精度整数
SUB_I	SUB_I	整数数学运算指令	减去整数
SUB_R	SUB_R	浮点算术运算指令	减去实数
SV	SE	定时器	启动延长脉冲定时器
S_VIMP	S_PEXT	定时器	分配延长脉冲定时器参数和启动
SZ	SC	计数器	置位计数器数值
TAN	TAN	浮点算术运算指令	生成浮点值角度的三角函数
TRUNC	TRUNC	转换	截取双精度整数部分

德语 助记符	英语 助记符	程序元素 类别	描述
UO	UO	状态位	无序的异常位
WAND_DW	WAND_DW	字逻辑指令	双字与运算(字)
WAND_W	WAND_W	字逻辑指令	字与(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	字或(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"逻辑运算
ZAEHLER	S_CUD	计数器	分配参数和递增/递减计数
ZR	CD	计数器	降值计数器
Z_RUECK	S_CD	计数器	分配参数和递减计数
ZV	CU	计数器	升值计数器
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 类别	描述
&	&	位逻辑指令	“与”逻辑运算
>=1	>=1	位逻辑指令	"或"逻辑运算
=	=	位逻辑指令	赋值
#	#	位逻辑指令	中间输出
---|	---|	位逻辑指令	插入二进制输入
---o|	---o|	位逻辑指令	二进制取反输入
==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	生成浮点数的绝对值
ACOS	ACOS	浮点算术运算指令	生成浮点值角度的三角函数
ADD_DI	ADD_DI	整数算术运算指令	加上双精度整数
ADD_I	ADD_I	整数算术运算指令	加上双精度整数
ADD_R	ADD_R	浮点算术运算指令	加上实数
ASIN	ASIN	浮点算术运算指令	生成浮点值角度的三角函数
ATAN	ATAN	浮点算术运算指令	生成浮点值角度的三角函数
BCD_DI	BCD_DI	转换	BCD 码转换为长整型
BCD_I	BCD_I	转换	将 BCD 码转换为整型
BR	BIE	状态位	BR 存取区异常位
CALL	CALL	程序控制	调用不带参数的 FC/SFC
CALL_FB	CALL_FB	程序控制	CALL_FB (以框方式调用 FB)
CALL_FC	CALL_FC	程序控制	CALL_FC(以框方式调用 FC)
CALL_SFB	CALL_SFB	程序控制	CALL_SFB (以框方式调用系统 FB)
CALL_SFC	CALL_SFC	程序控制	CALL_SFC (以框方式调用系统 FC)
CD	ZR	计数器	降值计数器
CEIL	CEIL	转换	向上取整
CMP ? D	CMP ? D	比较	比较长整数

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 类别	描述
CMP ? I	CMP ? I	比较	比较整数
CMP ? R	CMP ? R	比较	比较实数
COS	COS	浮点算术运算指令	生成浮点值角度的三角函数
CU	ZV	计数器	升值计数器
DI_BCD	DI_BCD	转换	长整型转换为 BCD 码
DI_R	DI_R	转换	长整型转换为实数
DIV_DI	DIV_DI	整数算术运算指令	除以双精度整数
DIV_I	DIV_I	整数算术运算指令	除以整数
DIV_R	DIV_R	浮点算术运算指令	除以实数
EXP	EXP	浮点算术运算指令	生成浮点数的指数值
FLOOR	FLOOR	转换	向下取整
I_BCD	I_BCD	转换	将整型转换为 BCD 码
I_DI	I_DI	转换	整型转换为长整型
INV_I	INV_I	转换	二进制反码整数
INV_DI	INV_DI	转换	二进制反码双精度整数
JMP	JMP	跳转	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	生成浮点数的自然对数
MCR>	MCR>	程序控制	主控制继电器打开/关
MCR<	MCR<	程序控制	主控制继电器打开/关
MCRA	MCRA	程序控制	主控制继电器激活/取消激活
MCRD	MCRD	程序控制	主控制继电器激活/取消激活
MOD_DI	MOD_DI	整数算术运算指令	返回双精度除法的余数
MOVE	MOVE	移动	分配值
MUL_DI	MUL_DI	整数算术运算指令	乘以双精度整数
MUL_I	MUL_I	整数算术运算指令	乘以整数
MUL_R	MUL_R	浮点算术运算指令	乘以实数
N	N	位逻辑指令	RLO 负跳沿检测
NEG	NEG	位逻辑指令	地址下降沿检测
NEG_DI	NEG_DI	转换	二进制补码双精度整数
NEG_I	NEG_I	转换	二进制补码整数
NEG_R	NEG_R	转换	取反实数
OPN	OPN	DB 调用	打开数据块
OS	OS	状态位	存储的异常位溢出
OV	OV	状态位	异常位溢出

英语 助记符	德语 助记符	程序元素 类别	描述
P	P	位逻辑指令	正 RLO 边沿检测
POS	POS	位逻辑指令	地址上升沿检测
R	R	位逻辑指令	复位输出
RET	RET	程序 控制	返回
ROL_DW	ROL_DW	移位/循环	双字循环左移
ROUND	ROUND	转换	取整到最接近的双精度整数
ROR_DW	ROR_DW	移位/循环	双字循环右移
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
SC	SZ	计数器	置位计数器数值
S_CD	Z_RUECK	计数器	分配参数和递减计数
S_CU	Z_VORW	计数器	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	启动接通延迟定时器
SE	SV	定时器	启动延长脉冲定时器
SF	SA	定时器	启动关闭延迟定时器
SHL_DW	SHL_DW	移位/循环	双字左移
SHL_W	SHL_W	移位/循环	字左移
SHR_DI	SHR_DI	移位/循环	长整数右移
SHR_DW	SHR_DW	移位/循环	双字右移
SHR_I	SHR_I	移位/循环	整数右移
SHR_W	SHR_W	移位/循环	字右移
SIN	SIN	浮点算术运算指令	生成浮点值角度的三角函数
S_ODT	S_EVERZ	定时器	分配接通延迟定时器参数和启动
S_ODTS	S_SEVERZ	定时器	分配保持接通延迟定时器参数和启动
S_OFFDT	S_AVERZ	定时器	分配关闭延迟定时器参数和启动
SP	SI	定时器	启动脉冲定时器
S_PEXT	S_VIMP	定时器	分配延长脉冲定时器参数和启动
S_PULSE	S_IMPULS	定时器	分配脉冲定时器参数和启动
SQR	SQR	浮点算术运算指令	生成浮点数的平方(SQR)
SQRT	SQRT	浮点算术运算指令	生成浮点数的平方根(SQRT)
SR	SR	位逻辑指令	置位复位触发器
SS	SS	定时器	启动保持接通延迟定时器
SUB_DI	SUB_DI	整数数学运算指令	减去双精度整数

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 类别	描述
SUB_I	SUB_I	整数数学运算指令	减去整数
SUB_R	SUB_R	浮点算术运算指令	减去实数
TAN	TAN	浮点算术运算指令	生成浮点值角度的三角函数
TRUNC	TRUNC	转换	截取双精度整数部分
UO	UO	状态位	无序的异常位
WAND_DW	WAND_DW	字逻辑指令	双字与运算(字)
WAND_W	WAND_W	字逻辑指令	字与(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	字或(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"逻辑运算

B 编程实例

B.1 编程实例总览

实际应用

每个 FBD 指令触发一个特定的操作。将这些指令组合到一个程序中时，便可完成多种自动化任务。本章说明了下列 FBD 指令的实际应用实例：

- 使用位逻辑指令控制传送带
- 使用为逻辑指令检测传送带上的移动方向
- 使用定时器指令生成一个时钟脉冲
- 使用计数器和比较指令跟踪存储空间
- 使用整数数学指令解决问题
- 设置加热烘炉的时间长度

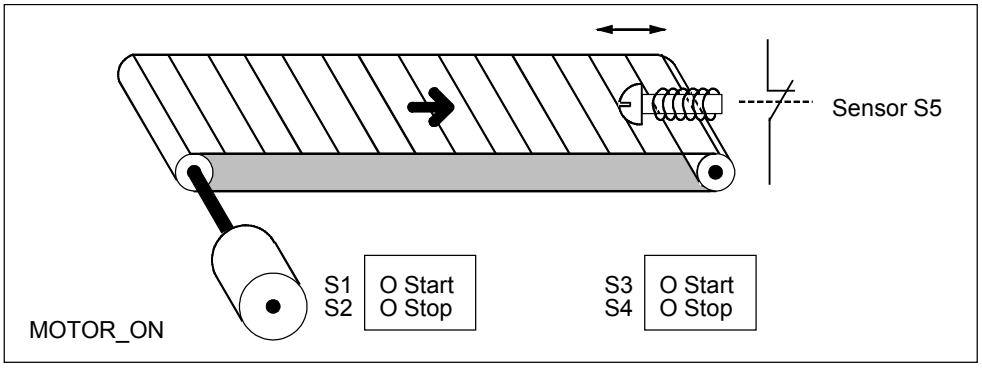
使用的指令

助记符	程序元素目录	描述
WAND_W	字逻辑指令	(字)与字
WOR_W	字逻辑指令	(字)或字
S_CD	计数器	降值计数器
S_CU	计数器	升值计数器
R	位逻辑指令	复位输出
S	位逻辑指令	置位输出
P	位逻辑指令	正 RLO 边沿检测
ADD_I	浮点指令	加上整数
DIV_I	浮点指令	除以整数
MUL_I	浮点指令	乘以整数
CMP >=I，CMP <=I	比较	比较整数
&	位逻辑指令	AND
>=1	位逻辑指令	OR
=	位逻辑指令	赋值
JMPN	跳转	若非则跳转
RET	程序控制	返回
MOVE	移动	赋值
SE	定时器	扩展脉冲定时器

B.2 实例：位逻辑指令

实例 1：控制传送带

下图显示可用电动方式激活的传送带。在传送带的开始位置有两个按钮开关：用于启动的 S1 和用于停止的 S2。在传送带末端也有两个按钮开关：用于启动的 S3 和用于停止的 S4。可从任何一端启动或停止传送带。此外，当传送带上的部件到达终点时，传感器 S5 将停止传送带。



绝对地址和符号编程

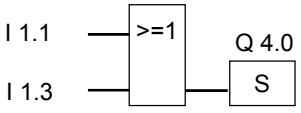
您可编写程序使用**绝对地址**或代表传送带系统各种组件的**符号**来控制传送带。

需要制定一个符号表，以建立所选择的符号与绝对地址的联系(参见 STEP 7 在线帮助)。

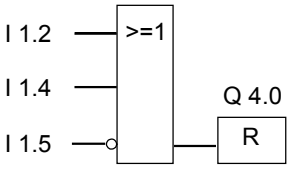
系统组件	绝对地址	符号	符号表
按钮启动开关	I 1.1	S1	I 1.1 S1
按钮停止开关	I 1.2	S2	I 1.2 S2
按钮启动开关	I 1.3	S3	I 1.3 S3
按钮停止开关	I 1.4	S4	I 1.4 S4
传感器	I 1.5	S5	I 1.5 S5
马达	Q 4.0	MOTOR_ON	Q 4.0 MOTOR_ON

控制传送带的功能块图

程序段 1：按下任意一个启动开关，启动电机。

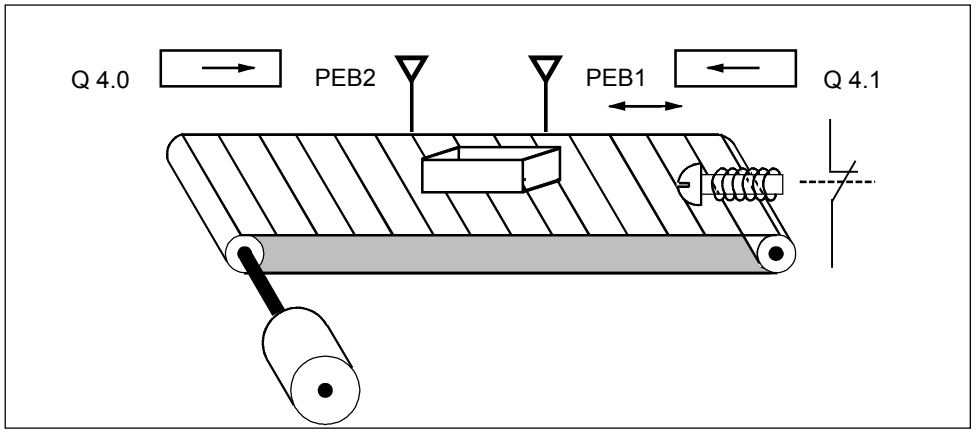


程序段 2：按下任意一个停止开关，或传送带末端的传感器，关闭电机。



实例 2：检测传送带方向

下图显示配备有两个光电屏障(PEB1 和 PEB2)的传送带 ,这两个光电屏障用于检测包裹在传送带上的移动方向。每个光电屏障的功能类似常开触点。



绝对地址和符号编程

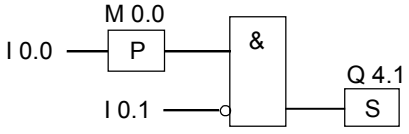
您可编写程序以使用**绝对地址**或代表传送带系统各种组件的**符号**来激活传送带系统的方向显示。

需要制定一个符号表，以建立所选择的符号与绝对地址的联系(参见 STEP 7 在线帮助)。

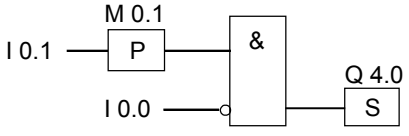
系统组件	绝对地址	符号	符号表
光电屏障 1	I 0.0	PEB1	I 0.0 PEB1
光电屏障 2	I 0.1	PEB2	I 0.1 PEB2
显示向右移动	Q 4.0	RIGHT	Q 4.0 RIGHT
显示向左移动	Q 4.1	LEFT	Q 4.1 LEFT
脉冲存储器位 1	M 0.0	PMB1	M 0.0 PMB1
脉冲存储器位 2	M 0.1	PMB2	M 0.1 PMB2

检测传送带方向的功能块图

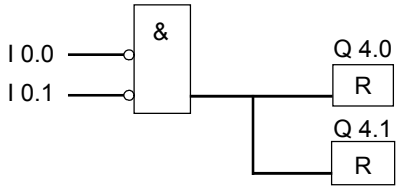
程序段 1：如果输入 I 0.0 的信号状态从 0 跳变为 1 (上升沿)，同时，输入 I 0.1 的信号状态为 0，则传送带上的包裹正在向左移动。



程序段 2：如果输入 I 0.1 的信号状态从 0 跳变为 1 (上升沿)，同时，输入 I 0.0 的信号状态为 0，则传送带上的包裹正在向右移动。



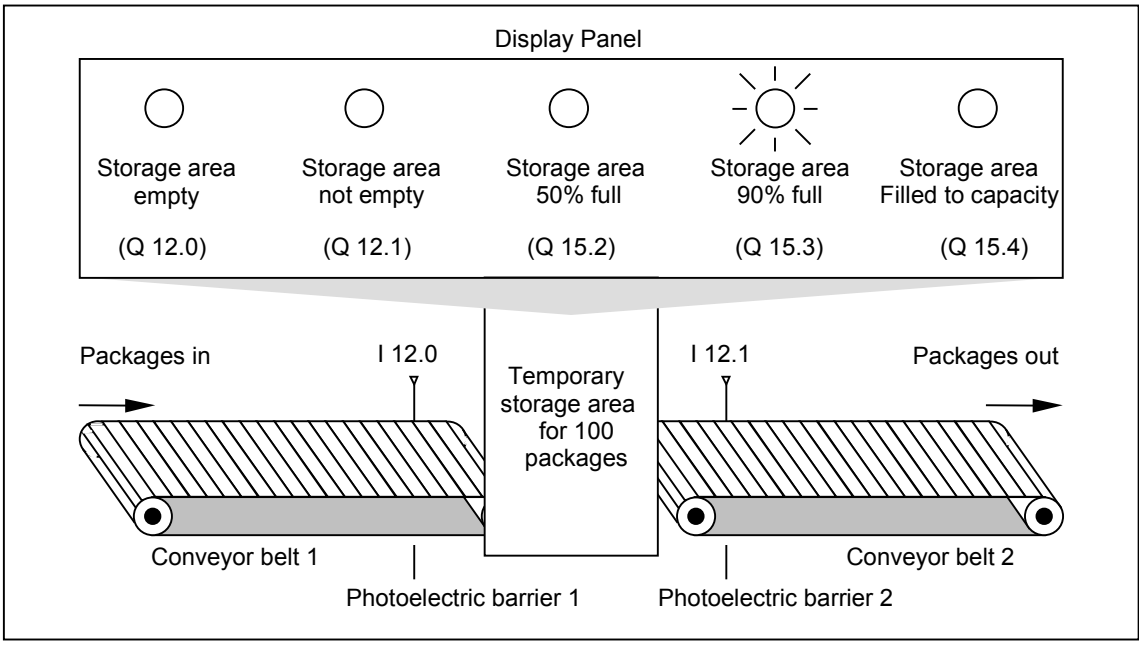
程序段 3：如果其中一个光电屏障被中断，则意味着在两个屏障之间有一个包裹。方向指针关闭。



B.3 实例：计数器和比较指令

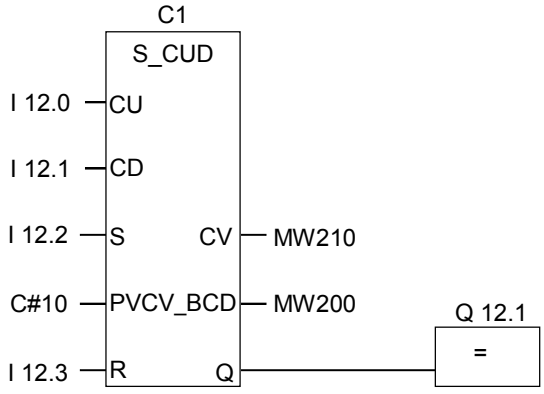
带计数器和比较器的存储区域

下图显示了具有两个传送带且在传送带之间有临时存储区域的系统。传送带 1 将包裹传送到存储区域。存储区域附近的传送带 1 末端的光电屏障确定向存储区域传送的包裹数量。传送带 2 会将包裹从临时存储区域传输到装载码头，而卡车在此将包裹发送给客户。存储区域附近的传送带 2 末端的光电屏障确定离开存储区域而转向装载码头的包裹数量。带五个指示灯的显示面板将指示临时存储区域的填充量。

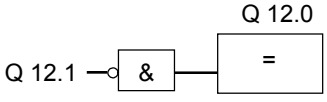


激活显示面板上的指示灯的功能块图

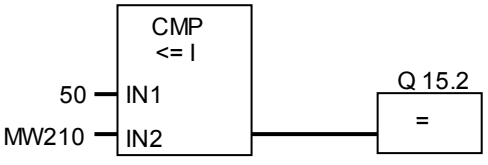
程序段 1：计数器 C1 对输入 CU 处每次从"0"到"1"的信号改变都进行正计数，而对输入 CD 处每次从"0"到"1"的信号改变都进行倒计数。对于输入 S 处从"0"到"1"的信号改变，计数器值被设置为值 PV。输入 R 处从"0"到"1"的信号改变将计数器值复位为"0"。MW200 包含 C1 的当前计数值。Q12.1 指示"存储区域非空"。



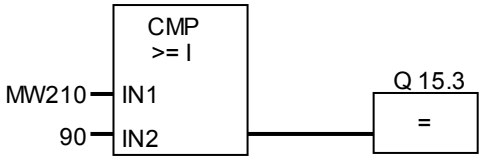
程序段 2：Q12.0 表明"存储区域为空"。



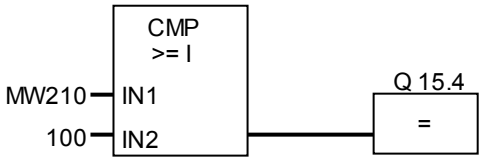
程序段 3：如果 50 小于等于计数器值(换句话说，如果当前计数器值大于等于 50)，则表示"存储区域 50%满"的指示灯变亮。



程序段 4：如果计数器值大于或等于 90，则表示"存储区域 90%满"的指示灯变亮。



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



B.4 实例：定时器指令

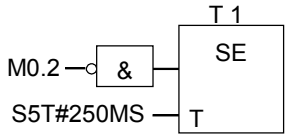
时钟脉冲发生器

当需要生成定期重复的信号时，可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很常见。

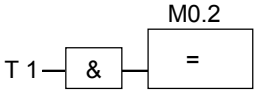
当使用 S7-300 时，您可用特殊组织块中的时间处理功能来执行时钟脉冲发生器功能。然而，以下 FBD 程序所示的实例，展示了如何使用定时器功能来生成时钟脉冲。实例程序显示如何通过使用定时器实现任意的时钟脉冲发生器。

生成时钟脉冲的功能块图(脉冲占空比 1:1)

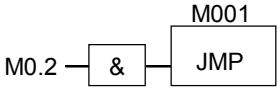
程序段 1：如果定时器 T1 的信号状态为 0，则将时间值 250 ms 加载到 T1 中，并启动 T1 作为延时脉冲定时器。



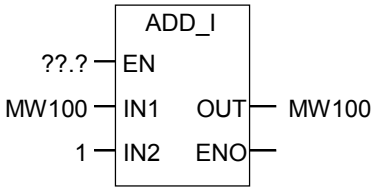
程序段 2：定时器的状态临时保存在辅助存储器标识器中。



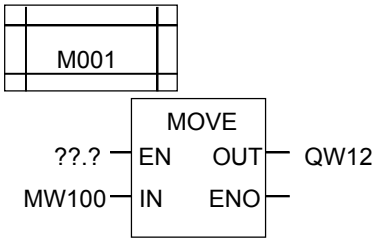
程序段 3：如果定时器 T1 的信号状态为 1，则跳转到跳转标签 M001。



程序段 4：当定时器 T1 时间到期后，存储器字 100 将增加 1。

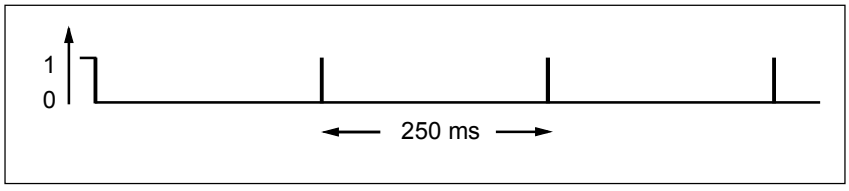


程序段 5：**MOVE** 指令允许在输出 Q12.0 到 Q13.7 上输出不同的时钟频率。



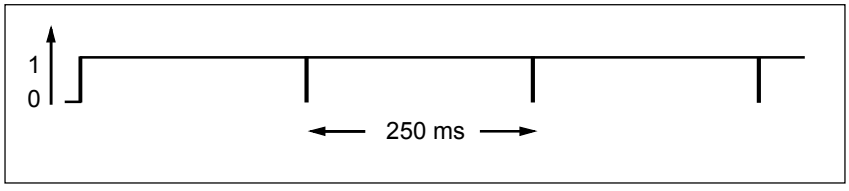
信号检测

对于时钟脉冲实例中 AND 逻辑运算(M0.2)取反的输入参数，定时器 T1 的信号检测生成下列逻辑运算结果(RLO)。



一旦定时时间到，就会重新启动定时器。因此，信号仅暂时生成信号状态 1。

RLO 取反(反向)：



每隔 250 ms，RLO 位变为 0。跳转被忽略，存储器字 MW100 中的内容加 1。

获得指定频率

从存储器字节 MB101 和 MB100 的各个位中，可以获得下列频率：

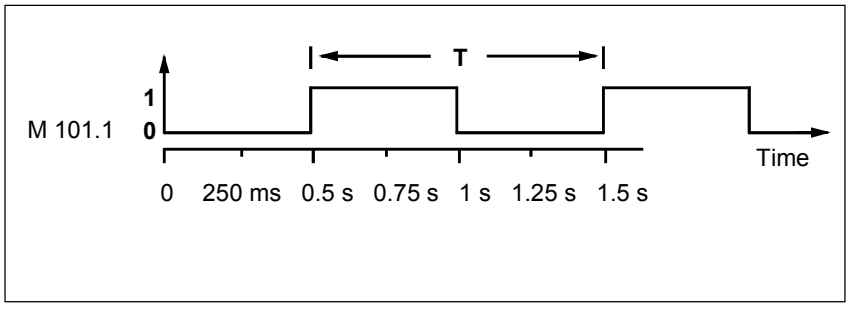
MB101/MB100 的位	频率(赫兹)	持续时间
M 101.0	2.0	0.5 s (250ms 开/ 250ms 关)
M 101.1	1.0	1 s (0.5s 开/0.5s 关)
M 101.2	0.5	2 s (1s 开/1s 关)
M 101.3	0.25	4 s (2s 开/2s 关)
M 101.4	0.125	8 s (4s 开/4s 关)
M 101.5	0.0625	16 s (8s 开/8s 关)
M 101.6	0.03125	32 s (16s 开/16s 关)
M 101.7	0.015625	64 s (32s 开/32s 关)
M 100.0	0.0078125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

存储器 MB 101 的位信号状态

扫描周期	第 7 位	第 6 位	第 5 位	第 4 位	第 3 位	第 2 位	第 1 位	第 0 位	时间值 (单位: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 (M 101.1)第 1 位的信号状态

频率 = 1/T = 1/1 s = 1 赫兹



B.5 实例：整型数学运算指令

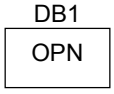
解决数学问题

实例程序显示了如何使用三个整数数学运算指令来产生与下列方程式相同的结果：

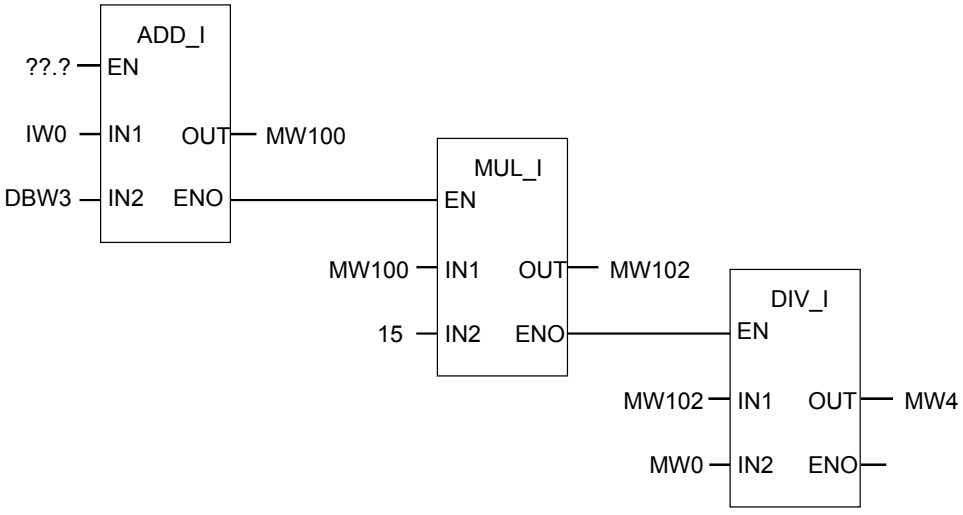
MW4 = ((IW0 + DBW3) x 15) / MW0

功能块图

程序段 1：打开数据块 DB1。



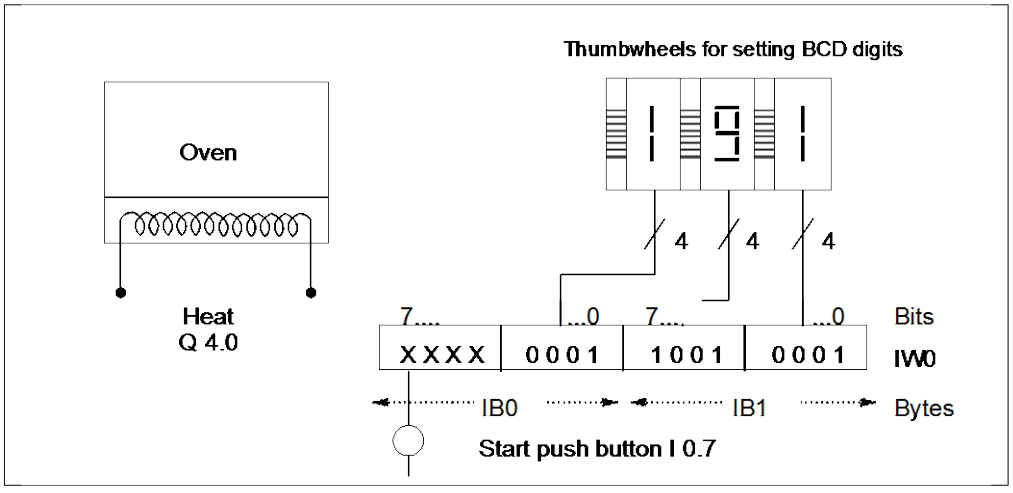
程序段 2：输入字 IW0 被加到共享数据字 DBW3 (必须先定义并打开数据块)，和被加载到存储器字 MW100 中。然后，将 MW100 乘以 15，结果保存在存储器字 MW102 中。再将 MW102 除以 MW0，结果保存在 MW4 中。



B.6 实例：字逻辑指令

加热烘炉

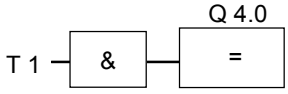
烘炉操作员通过按启动按钮来启动烘炉加热。操作员可用图中所示的码盘开关来设置加热的时间。操作员设置的值以二进制编码的十进制(BCD)格式显示，单位为秒。



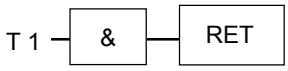
系统组件	绝对地址
启动按钮	I 0.7
个位码盘	I 1.0 到 I 1.3
十位码盘	I 1.4 到 I 1.7
百位码盘	I 0.0 到 I 0.3
加热启动	Q 4.0

功能块图

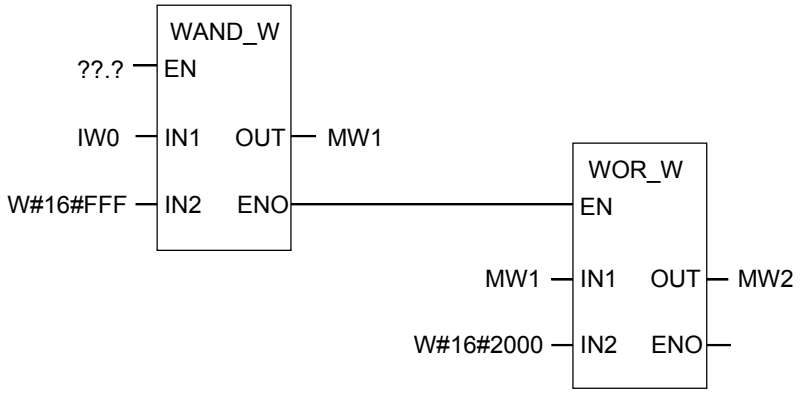
程序段 1：如果定时器正在运行，则启动加热器。



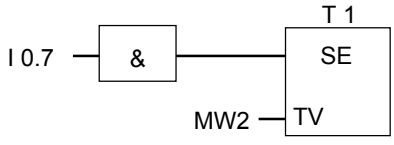
程序段 2：如果定时器正在运行，则 **Return** 指令结束此处的处理。



程序段 3：屏蔽输入位 I 0.4 到 I 0.7 (即将其复位为 0)。码盘输入的这些位不使用。码盘输入的 16 个位，根据**(字)与字**指令，与 W#16#0FFF 相组合。结果装载在存储器字 MW12 中。为了设置秒数的时间基准，预设值将根据**(字)或字**指令，与 W#16#2000 相组合，将位 13 设为 1，将位 12 复位为 0。



程序段 4：如果按下启动按钮，则将启动定时器 T1 作为延时脉冲定时器，装载存储器字 MW2 (从上面的逻辑得到)作为预设值。



C 使用功能块图

C.1 块类型

各种块类型

用户程序由各种可用于构造程序的块组成。块是程序的独立部分，用于执行特定的功能。在 STEP 7 中，对逻辑块和数据块进行了区分。逻辑块(OB、FB、SFB、FC 和 SFC)包含用于处理信号的指令，而数据块(DB、DI)用于存储数据。

逻辑块

逻辑块用作程序和子程序的软件模块。基础块为 OB1 (组织块 1)。若要从另一个块中调用一个块，可对块的变量声明表进行编程，以便它们能够彼此交换参数。例如，可对被调用块进行编程，以便它接受来自调用块的各种输入/输出参数。被调用块对输入/输出参数的输入值进行处理，生成一个结果。然后将结果和对程序的控制权返回给调用块。

指令

指令 CALL FC1 (调用一个功能)和 CALL FB1, DB2 (调用一个功能块)的区别是：调用 FB 时会分配一个数据块(实例 DB)。此数据块根据相应功能块的变量声明表存储本地块变量。对于这些功能，只要调用另一个块，就会删除执行该功能期间在本地存储器中存储的调用参数和赋值。

数据块

有两种类型的数据块：

- 包含已创建的结构中的用户数据的数据块。这些数据块可由逻辑块打开，以进行读、写操作。
- 存储用于执行功能块实例(实例 DB)的调用参数和静态局部数据的数据块。

可在梯形图、FBD 或 STL 编辑器中创建数据块和逻辑块。西门子(公司)在 STEP 7 数据包中为用户提供了各种标准块。

C.2 EN/ENO 机制

FBD/LAD 框的启用(EN)和启用输出(ENO)通过 BR 位来获取。

如果连接了 EN 和 ENO，则以下规则适用：

ENO = EN AND NOT (框错误)

当没有发生错误(框错误 = 0)时，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。

该机制**不**用于：

- 比较、
- 计数器、
- 定时器。

在框的实际指令周围，为 EN/ENO 机制生成附加的 STL 指令，这些指令依赖于现有的在此之前和之后的逻辑运算。使用一个加法器实例，显示下列四种可能的情况：

1. 连接了 EN 和 ENO 的加法器
2. 连接 EN 但未连接 ENO 的加法器
3. 连接 EN 但未连接 ENO 的加法器
4. 没有连接 EN 和 ENO 的加法器

创建块的注意事项

如果要编程在 FBD 或 LAD 中调用的块，那么必须确保退出块时，置位 BR 位。第四个实例显示这种结果并不会自动出现。不能将 BR 作为存储位，因为 EN/ENO 机制不断改写 BR。作为代替，可使用一个临时变量来保存发生的所有错误。用 0 初始化该变量。在块中任何一个您认为指令不成功即表示整个块出错的地方 ,借助 EN/ENO 机制来设置该变量。一个 NOT 和一个 SET 线圈足以完成这项工作。在块结束处，编程下列程序段：

```
end:   AN error
       SAVE
```

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

C.2.1 连接了 EN 和 ENO 的加法器

当加法器连接了 EN 和 ENO 时，触发下列 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                    // 将错误存入 BR
9      CLR                    // 首次检查
10 _001:   A      BR          //将 BR 移位到 RLO 中
11     =    Q    4.0
```

在第 1 行后，RLO 包含在此之前的逻辑运算的结果。JNB 指令将 RLO 复制到 BR 位，并设置第一个校验位。

- 当 RLO = 0 时，程序跳转到第 10 行，继续执行 A BR。不执行加法指令。在第 10 行，重新将 BR 复制到 RLO 中，然后给输出赋值 0。
- 当 RLO = 1 时，程序不跳转，表示执行加法指令。在第 7 行中，程序判断是否在执行加法指令期间发生了错误，然后在第 8 行存储到 BR 中。第 9 行设置第一个校验位。现在，在第 10 行中将 BR 位复制回 RLO，因此，输出显示是否成功执行了加法指令。
  第 10 行和第 11 行不改变 BR 位，因此，也显示是否成功执行了加法指令。

C.2.2 连接 EN 但未连接 ENO 的加法器

当加法器连接了 EN 但未连接 ENO 时，触发下列 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 行后，RLO 包含在此之前的逻辑运算的结果。JNB 指令将 RLO 复制到 BR 位，并设置第一个校验位。

- 当 RLO = 0 时，程序跳转到第 7 行，不执行加法指令。RLO 和 BR 都为 0。
- 当 RLO 为 1 时 ,程序不跳转 ,表示执行加法指令。程序不判断在执行加法指令期间是否发生错误。RLO 和 BR 都为 1。

C.2.3 连接 EN 但未连接 ENO 的加法器

当加法器没有连接 EN，但连接了 ENO 时，触发下列 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               // 将错误存入 BR
7     CLR                        // 首次检查
8             A      BR           // 将 BR 移位到 RLO 中
9     = Q 4.0
```

在每种情况下都执行加法指令。在第 5 行中，程序判断在执行加法指令期间是否发生了错误，然后在第 6 行存储到 BR 中。第 7 行设置第一个校验位。现在，在第 8 行中将 BR 位复制回 RLO，因此，输出显示是否成功执行了加法指令。

第 8 行和第 9 行不改变 BR 位，因此，也显示是否成功执行了加法指令。

C.2.4 没有连接 EN 和 ENO 的加法器

当加法器没有连接 EN 和 ENO 时，触发下列 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
```

执行加法指令。RLO 和 BR 位保持不变。

C.3 参数传送

块的参数作为数值传送。对于功能块，在被调用块中使用背景数据块中的实际参数值副本。对于功能，实际值的副本存在于本地数据堆栈中。不复制指针。调用前，将 INPUT 数值复制到背景 DB 或 L 堆栈中。调用后，将 OUTPUT 数值复制回变量中。在被调用块中，只能使用副本。所需的 STL 指令位于调用块中，并且对于用户来说是不可见。

注意

如果存储位、输入、输出或外围设备 I/O 作为功能的实际地址使用，那么它们以与其它地址不同的方式进行处理。这种情况下，将直接进行更新，而不是通过 L 堆栈。

例外：

如果相应的形式参数为 BOOL 数据类型的一个输入参数，那么通过 L 堆栈更新当前参数。

注意

在编程调用块时，请确保写入声明为 OUTPUT 的参数。否则，输出值为随机值！对于功能块，该值为由最后一个调用指定的背景 DB 中的数值；对于功能，该值为位于 L 堆栈的数值。

请注意以下几点：

- 尽可能初始化所有 OUTPUT 参数。
- 尽量不要使用任何置位和复位指令。这些指令取决于 RLO。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

字母数字

B

C

D

E

F

G

H

J

K

L

M

P

Q

R

S

T

W

X

Y

Z