SIEMENS

SIMATIC

S7-300 和 S7-400 梯形逻辑（LAD）编程

参考手册

2004年1月版

安全指南

本手册包括应该遵守的注意事项，以保证人身安全，保护产品和所连接的设备免受损坏。这些注意事项都使用符号明显警示，并根据严重程度使用下述文字分别说明：

危险

表示若不采取适当的预防措施，将造成死亡、严重的人身伤害或重大的财产损失。

警告

表示若不采取适当的预防措施，将可能造成死亡、严重的人身伤害或重大的财产损失。

小心

表示若不采取适当的预防措施，将可能造成轻微的人身伤害。

小心

表示若不采取适当的预防措施，将可能造成财产损失。

注意

引起你对产品的重要信息和处理产品或文件的特定部分的注意。

合格人员

只有合格人员才允许安装和操作这一设备。合格人员规定为根据既定的安全惯例和标准批准进行试运行、接地和为电路、设备和系统加装标签的人员。

正确使用

注意如下：

警告

本装置及其组件只能用于产品目录或技术说明书中阐述的应用，并且只能与西门子公司认可或推荐的其它生产厂的装置或组件相连接。

本产品只有在正确的运输、贮存、组装和安装的情况下，按建议方式进行运行和维护，才能正确而安全地发挥其功能。

商标

SIMATIC®、SIMATIC HMI®和 SIMATIC NET®为西门子公司的注册商标。

任何第三方为其自身目的使用与本手册中所及商标有关的其它名称，都将侵犯商标所有人的权益。

西门子股份有限公司
自动化与驱动集团
工业自动化系统部
纽伦堡邮政信箱4848，邮编D- 90327
Siemens Aktiengesellschaft

郑重声明
我们已核对过，本手册的内容与所述硬件和软件相符。但错误在所难免，不能保证完全的一致。本手册中的内容将定期审查，并在下一版中进行修正。欢迎提出改进意见。

A5E00171231-01

Excellence in Automation & Drives: Siemens

前言

目的

本使用手册旨在提供指南，以使用梯形逻辑（LAD）编程语言生成用户程序。

本手册中还包含一个参考章节，阐述了梯形逻辑语言元素的语法和功能。

所需基本知识

本手册旨在用于编程人员、操作人员以及维护和维修人员。

为了很好理解本手册，需要具有自动化技术的一般知识。

除此之外，还需要具备计算机知识以及操作系统 MS Windows 2000 Professional 或 MS Windows XP Professional 下类似于 PC 的其它工作设备知识。

本手册的应用范围

本手册适用于 STEP 7 编程软件包的 5.3 版。

符合标准 IEC 1131-3

LAD 是指国际电工委员会标准 IEC 1131-3 中定义的“梯形逻辑”编程语言。有关详细信息，请参见 STEP 7 文件 NORM_TBL.WRI 中的标准表。

要求

为了有效使用本《梯形逻辑手册》，用户应熟悉 S7 程序理论。关于 S7 程序，可参见 STEP 7 在线帮助。

编程语言软件包也使用 STEP 7 标准软件，因此，用户还应熟悉该软件的操作，并阅读随附的资料。

本手册是“STEP 7 参考资料”整套资料的一部分。

下表所示为 STEP 7 的整套资料：

文件	目的	订货号
STEP 7基本信息 • STEP 7 V5.3，《快速入门手册》 • STEP 7 V5.3 编程 • 配置硬件和通讯连接，STEP 7 V5.3 • 《从 S5 到 S7 转换手册》	向技术人员解释关于使用 STEP 7以及S7-300/400可编程控制器实现控制任务的方法的基本信息。	6ES7810-4CA07-8BW0
STEP 7 参考资料 • 《S7-300/400 梯形逻辑（LAD）/功能块图（FBD）/语句表（STL）使用手册》 • 《S7-300/400 标准和系统功能手册》	介绍一些参考信息以及编程语言 LAD、FBD 和 STL 以及 STEP 7 基本信息的扩展标准功能和系统功能。	6ES7810-4CA07-8BW1

在线帮助	目的	订货号
STEP 7 帮助	以在线帮助的形式提供关于使用STEP 7 编程和组态硬件的基本信息。	为 STEP 7 标准软件包的一部分
《AWL/KOP/FUP 参考帮助》 《SFB/SFC参考帮助》《组织块参考帮助》	上下文相关信息	为 STEP 7 标准软件包的一部分

在线帮助

集成在软件中的在线帮助是本手册的补充。在线帮助的目的是为你提供详细的软件使用帮助。

帮助系统通过多个界面集成在软件中：

- 上下文相关帮助可以提供关于当前的文本信息，例如，一个打开的对话框或一个激活的窗口。你可以按动 F1 或使用工具栏中的“？”，通过菜单命令 Help > Context-Sensitive Help，打开文本相关的帮助。
- 你可以使用菜单命令 Help > Contents 或文本相关帮助窗口中的“Help on STEP 7”按钮，调用 STEP 7 中的一般帮助信息。
- 你也可以通过“Glossary（术语）”按钮，调用所有 STEP 7 应用的术语。

本手册是“梯形逻辑中的帮助信息”摘选。由于手册和在线帮助的结构一样，所以能够很容易地在手册和在线帮助之间进行转换。

其它支持

如果你有任何技术问题，你可以与当地的西门子代表处或代理商联系。

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

http://www.ad.siemens.com.cn/service

培训中心

西门子公司还提供有许多培训课程，介绍 SIMATIC S7 自动化系统。详情请与您所在地区的培训中心联系，或与德国纽伦堡（邮编 D90327）的总部培训中心联系：

德　国：+49 (911) 895 - 3200

北　京：(010) 6439 2860

上　海：(021) 3220 0899 - 306

广　州：(020) 8732 0088 - 2279

武　汉：(027) 8548 6688 - 6601

哈尔滨：(0451) 239 3129

重　庆：(023) 6382 8919 - 3002

网址：http://www.sitrain.com

http://www.ad.siemens.com.cn

A&D 技术支持

遍布全球，24 小时服务：

面向全球（纽伦堡） 技术支持	欧洲/非洲（纽伦堡）授权	
一天24 小时，一年 365 天全天候 电话： +49 (0) 180 5050-222 传真： +49 (0) 180 5050-223 E-Mail： adsupport@ siemens.com GMT： +1:00	当地时间：星期一 — 星期五 08:00:00至17:00:00 电话： +49 (0) 180 5050-222 传真： +49 (0) 180 5050-223 E-Mail： adsupport@siemens.com GMT： +1:00	
美国（约翰逊市）技术支持和授权	**亚洲/澳大利亚（北京） 技术支持和授权**	**亚洲/中国（北京） 技术支持与服务热线**
当地时间：星期一 — 星期五 08:00:00至17:00:00 电话： +1 (0) 770 740 3505 传真： +1 (0) 770 740 3699 E-Mail： isd-callcenter@ sea.siemens.com GMT： -5:00	当地时间：星期一 — 星期五 08:30:00至17:30:00 电话： +86 10 64 75 75 75 传真： +86 10 64 74 74 74 E-Mail： adsupport.asia@ siemens.com GMT： +8:00	当地时间：星期一 — 星期五 08:30:00至17:30:00 电话： +86 10 64 75 75 75 传真： +86 10 64 74 74 74 E-Mail： adscs.china@siemens.com GMT： +8:00
SIMATIC 热线和授权热线的使用语言一般为德语和英语。		

网上服务和技术支持

除了纸文件资料以外，我们在网上还提供有在线资料：

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

http://www.ad.siemens.com.cn

在网上你可以找到：

- **新闻列表可以向你提供不断更新的最新产品信息。**
- **通过网上服务和技术支持部分的搜索功能，可以找到所需文件。**
- **在论坛部分，全世界的用户和专家都可交流其经验。**
- **通过我们在网上的代表处数据库，你可以找到当地的自动化与驱动集团代表处。**
- **有关现场服务、修理、备件等更多信息，可参见“ 服务 ”。**

目录

1 位逻辑指令

1.1 位逻辑指令概述

说明

位逻辑指令处理两个数字，“1”和“0”。这两个数字构成二进制数字系统的基础。这两个数字“1”和“0”称为二进制数字或二进制位。在接点与线圈领域，“1”表示动作或通电，“0”表示未动作或未通电。

位逻辑指令扫描信号状态 1 和 0，并根据布尔逻辑对它们进行组合。这些组合产生结果 1 或 0，称为“逻辑运算结果（RLO）”。

由位逻辑指令触发的逻辑操作可执行各种类型的功能。

可执行下列功能的位逻辑指令：

- ---| |--- **常开接点（地址）**
- ---| / |--- **常闭接点（地址）**
- ---(SAVE) **将 RLO 存入 BR 存储器**
- XOR **位异或**
- ---( ) **输出线圈**
- ---(#)--- **中间输出**
- ---|NOT|--- **信号流反向**

下列指令当 RLO 为 1 时起作用，执行下列功能：

- ---(S) **线圈置位**
- ---(R) **线圈复位**
- SR **置位复位触发器**
- RS **复位置位触发器**

其它指令对上升沿和下降沿有反应，执行下列功能：

- ---(N)--- RLO **下降沿检测**
- ---(P)--- RLO **上升沿检测**
- NEG **地址下降沿检测**
- POS **地址上升沿检测**
- **立即读操作**
- **立即写操作**

1.2 ---| |--- 常开接点（地址）

符号

<地址>

---| |---

参数	数据类型	存储区域	说明
<地址>	BOOL	I，Q，M，L，D，T，C	要检查的位

说明

当保存在指定<地址>中的位值等于“1”时，---| |---**（常开接点）闭合。当接点闭合时，梯形逻辑级中的信号流经接点，逻辑运算结果（RLO）=“1”。**

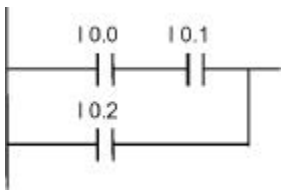
相反，如果指定<地址>的信号状态为“0”，接点打开。当接点打开时，没有信号流经接点，逻辑运算结果（RLO）=“0”。

串联使用时，---| |--- **通过“与（AND）”逻辑链接到** RLO **位。并联使用时，**---| |--- **通过“或（OR）”逻辑链接到** RLO **位。**

状态字

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

举例



如果下列条件之一成立，则电流流通：

在输入 I0.0 **和** I0.1 **的信号状态为“1”**

或在输入 I0.2 **的信号状态为“1”**

1.3 ---| / |--- 常闭接点（地址）

符号

<地址>

---| / |---

参数	数据类型	存储区域	说明
<地址>	BOOL	I，Q，M，L，D，T，C	要检查的位

说明

当保存在指定<地址>中的位值等于“ 0 ”时，---| / |---**（常闭接点）闭合。当接点闭合时，梯形逻辑级中的信号流经接点，逻辑运算结果（**RLO**）=“ 1 ”。**

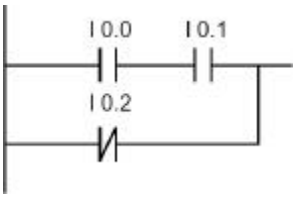
相反，如果指定<地址>的信号状态为“ 1 ”，接点打开。当接点打开时，没有信号流经接点，逻辑运算结果（RLO**）=“ 0 ”。**

串联使用时，---| / |--- **通过“ 与（**AND**） ”逻辑链接到** RLO **位。并联使用时，**---| / |--- **通过“ 或（**OR**） ”逻辑链接到** RLO **位。**

状态字

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

举例

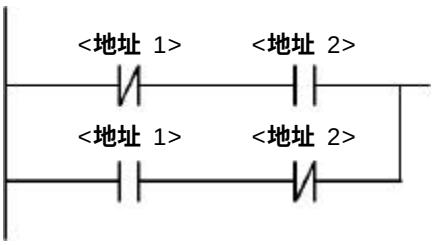


如果下列条件之一成立，则电流流通：
在输入 I0.0 **和** I0.1 **的信号状态为“** 1 **”**
或在输入 I0.2 **的信号状态为“** 1 **”**

1.4 XOR **位异或**

对于 XOR **功能，常开接点和常闭接点程序段必须如下生成。**

符号

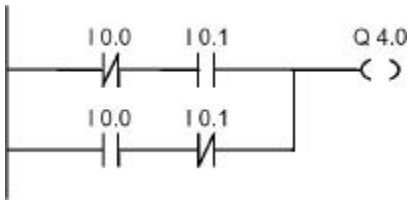


参数	数据类型	存储区域	说明
<地址 1>	BOOL	I，Q，M，L，D，T，C	扫描位
<地址 2>	BOOL	I，Q，M，L，D，T，C	扫描位

说明

如果两个指定位的信号状态不同，XOR**（位异或）将产生一个** RLO**“** 1 **”。**

举例



如果（I0.0 = “ 0 ” AND I0.1 = “ 1 ”）OR（I0.0 = “ 1 ” AND I0.1 = “ 0 ”），**则输出** Q4.0 **为** “ 1 ”。

1.5 --|NOT|-- 信号流反向

符号

---|NOT|---

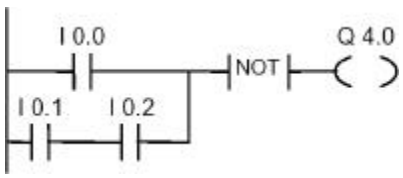
说明

--|NOT|---（**信号流反向指令**）**取** RLO **位的非值。**

状态字

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

举例



如果下列条件之一成立，则输出 Q4.0 **的信号状态为** “ 0 ” **：**

在输入 I0.0 **的信号状态为** “ 1 ”

或在输入 I0.1 **和** I0.2 **的信号状态为** “ 1 ”

1.6 ---( ) 输出线圈

符号

<地址>

---( )

参数	数据类型	存储区域	说明
<地址>	BOOL	I，Q，M，L，D	赋值位

说明

---(　　)（输出线圈指令）象继电器逻辑图中的线圈一样作用。如果有电流流过线圈（RLO = 1），位置<地址>处的位则被置为“1”。如果没有电流流过线圈（RLO = 0），位置<地址>处的位则被置为“0”。输出线圈只能放置在梯形逻辑级的右端。也可以有多个输出元素（最多 16 个）（见举例）。使用 ---|NOT|---（信号流反向）元素，可以生成求反输出。

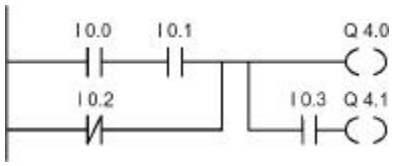
MCR（主控继电器）附属级

MCR 附属级只有在输出线圈放在激活的 MCR 区中时才能触发。在一个激活的 MCR 区内，如果 MCR 接通，并且有电流流经输出线圈，则被寻址的位将被置为信号流的当前状态。如果 MCR 断开，逻辑“0”则被写入指定地址，与信号流状态无关。

状态字

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

举例



如果下列条件之一成立，则输出 Q4.0 的信号状态为“1”：

在输入 I0.0 和 I0.1 的信号状态为“1”

或在输入 I0.2 的信号状态为“0”

如果下列条件之一成立，则输出 Q4.1 的信号状态为“1”：

在输入 I0.0 和 I0.1 的信号状态为“1”

或在输入 I0.2 的信号状态为“0”

并且在输入 I0.3 的信号状态为“1”

如果举例中的梯形逻辑级在激活的 MCR 区内：

当 MCR 接通时，Q4.0 和 Q4.1 将如上所述根据信号流状态置位。

当 MCR 断开时（=0），Q4.0 和 Q4.1 将被复位为“0”，与信号流状态无关。

1.7 ---(#)--- 中间输出

符号

<地址>

---(#)---

参数	数据类型	存储区域	说明
<地址>	BOOL	I，Q，M，*L，D	赋值位

* 只有 L 存储区中的地址在一个逻辑块（FC，FB，OB）的变量声明表中进行 TEMP 声明时才能被使用。

说明

---(#)---（中间输出指令）是一个中间赋值元素，可以将 RLO 位（信号流状态）保存到指定的 <地址>。这一中间输出元素可以保存前一分支元素的逻辑结果。与其它接点并联时，---(#)--- 可以象一个接点那样插入。---(#)--- 元素绝不能连接到电源线上或直接连接到一个分支连接的后面或一个分支的末尾。使用 --- |NOT|---（信号流反向）元素，可以生成求反 ---(#)---。

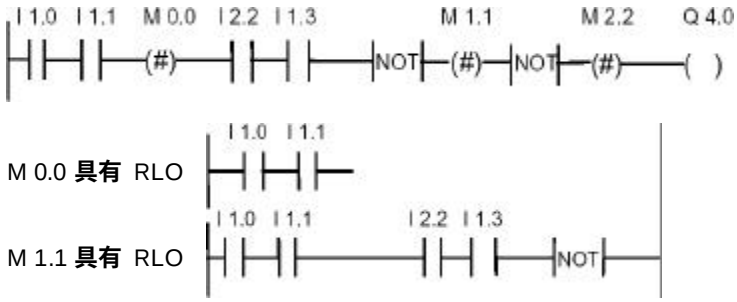
MCR（主控继电器）附属级

MCR 附属级只有在中间输出线圈放在激活的 MCR 区中时才能触发。在一个激活的 MCR 区内，如果 MCR 接通，并且有电流流经中间输出线圈，则被寻址的位将被置为信号流的当前状态。如果 MCR 断开，逻辑“ 0 ”则被写入指定地址，与信号流状态无关。

状态字

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

举例



M 2.2 具有全部位逻辑组合的 RLO

1.8 ---(R) 线圈复位

符号

<地址>

---(R)

参数	数据类型	存储区域	说明
<地址>	BOOL	I，Q，M，L，D，T，C	复位的位

说明

---(R)（线圈复位指令）只有在前一指令的 RLO 为“ 1 ”时（电流流经线圈），才能执行。如果有电流流过线圈（RLO 为“ 1 ”），元素的指定<地址>处的位则被复位为“ 0 ”。RLO 为“ 0 ”（没有电流流过线圈）没有任何作用，并且元素指定地址的状态保持不变。<地址>也可以是一个定时器值被复位为“ 0 ”的定时器（T no.）或一个计数器值被复位为“ 0 ”的计数器（C no.）。

MCR（主控继电器）附属级

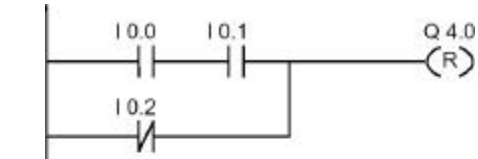
MCR 附属级只有在复位线圈放在激活的 MCR 区中时才能触发。在一个激活的 MCR 区内，如果 MCR 接通，并且有电流流经复位线圈，则被寻址的位将被复位为“0”状态。如果 MCR 断开，则元素指定地址的当前状态保持不变，与信号流状态无关。

状态字

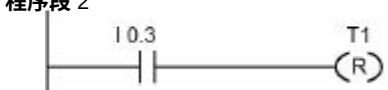
	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
写	-	-	-	-	-	0	X	-	0

举例

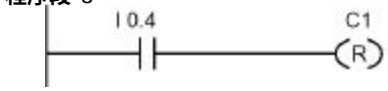
程序段 1



程序段 2



程序段 3



如果下列条件之一成立，则输出 Q4.0 的信号状态被复位为“0”：

在输入 I0.0 和 I0.1 的信号状态为“1”

或在输入 I0.2 的信号状态为“0”

如果 RLO 为“0”，则输出 Q4.0 的信号状态保持不变。

定时器 T1 的信号状态只有在以下情况下才被复位：

在输入 I0.3 的信号状态为“1”

计数器 C1 的信号状态只有在以下情况下才被复位：

在输入 I0.4 的信号状态为“1”

如果举例中的梯形逻辑级在激活的 MCR 区内：

当 MCR 接通时，Q4.0、T1 和 C1 将如上所述被复位。

当 MCR 断开时，Q4.0、T1 和 C1 将保持不变，与 RLO 的状态无关（信号流状态）。

1.9 ---(S) 线圈置位

符号

<地址>

---(S)

参数	数据类型	存储区域	说明
<地址>	BOOL	I，Q，M，L，D	被置位的位

说明

---(S)（线圈置位指令）只有在前一指令的 RLO 为“ 1 ”时（电流流经线圈），才能执行。如果 RLO 为“ 1 ”时，元素的指定<地址>将被置为“ 1 ”。

RLO = 0 没有任何作用，并且元素指定地址的状态保持不变。

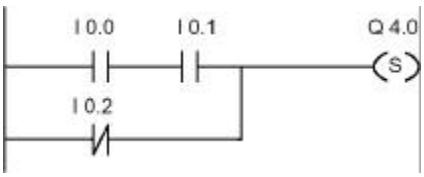
MCR（主控继电器）附属级

MCR 附属级只有在置位线圈放在激活的 MCR 区中时才能触发。在一个激活的 MCR 区内，如果 MCR 接通，并且有电流流经置位线圈，则被寻址的位将被置为“ 1 ”状态。如果 MCR 断开，则元素指定地址的当前状态保持不变，与信号流状态无关。

状态字

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

举例



如果下列条件之一成立，则输出 Q4.0 的信号状态为“ 1 ”：

在输入 I0.0 和 I0.1 的信号状态为“ 1 ”

或在输入 I0.2 的信号状态为“ 0 ”

如果 RLO 为“ 0 ”，则输出 Q4.0 的信号状态保持不变。

如果举例中的梯形逻辑级在激活的 MCR 区内：

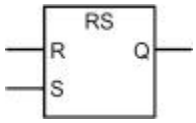
当 MCR 接通时，Q4.0 将如上所述被置位。

当 MCR 断开时，Q4.0 将保持不变，与 RLO 的状态无关（信号流状态）。

1.10 RS 复位置位触发器

符号

<地址>



参数	数据类型	存储区域	说明
<地址>	BOOL	I，Q，M，L，D	被置位或复位的位
S	BOOL	I，Q，M，L，D	使能置位指令
R	BOOL	I，Q，M，L，D	使能复位指令
Q	BOOL	I，Q，M，L，D	<地址>的信号状态

说明

如果在 R 端输入的信号状态为“1”，在 S 端输入的信号状态为“0”，则 RS（复位置位触发器）复位。相反，如果在 R 端输入的信号状态为“0”，在 S 端输入的信号状态为“1”，则 RS（复位置位触发器）置位。如果在两个输入端 RLO 均为“1”，则顺序优先，触发器置位。在指定<地址>，复位置位触发器首先执行复位指令，然后执行置位指令，以使该地址保持置位状态程序扫描剩余时间。

S（置位）和 R（复位）指令只有在 RLO 为“1”时才执行。RLO“0”对这些指令没有任何作用，并且指令中的指定地址保持不变。

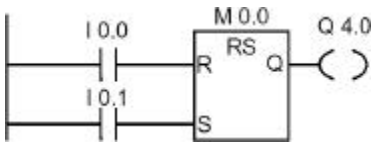
MCR（主控继电器）附属级

MCR 附属级只有在复位置位触发器放在激活的 MCR 区中时才能触发。在一个激活的 MCR 区内，如果 MCR 接通，则被寻址的位将如上所述被复位为“0”或置位为“1”。如果 MCR 断开，则指定地址的当前状态保持不变，与输入状态无关。

状态字

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

举例



如果输入 I0.0 的信号状态为“1”，输入 I0.1 的信号状态为“0”，则存储位 M0.0 将被复位，输出 Q4.0 为“0”。相反，如果输入 I0.0 的信号状态为“0”，输入 I0.1 的信号状态为“1”，则存储位 M0.0 将被复位，输出 Q4.0 为“1”。如果两个信号状态均为“0”，则无变化。如果两个信号状态均为“1”，则由于顺序之故，置位指令优先；M0.0 置位，Q4.0 为“1”。

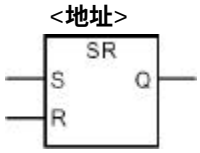
如果举例在激活的 MCR 区内：

当 MCR 接通时，Q4.0 将如上所述被复位或置位。

当 MCR 断开时，Q4.0 将保持不变，与输入状态无关。

1.11 SR 置位复位触发器

符号



参数	数据类型	存储区域	说明
<地址>	BOOL	I，Q，M，L，D	被置位或复位的位
S	BOOL	I，Q，M，L，D	使能置位指令
R	BOOL	I，Q，M，L，D	使能复位指令
Q	BOOL	I，Q，M，L，D	<地址>的信号状态

说明

如果在 S 端输入的信号状态为“1”，在 R 端输入的信号状态为“0”，则 SR（置位复位触发器）置位。相反，如果在 S 端输入的信号状态为“0”，在 R 端输入的信号状态为“1”，则 SR（置位复位触发器）复位。如果在两个输入端 RLO 均为“1”，则顺序优先，触发器置位。在指定<地址>，置位复位触发器首先执行置位指令，然后执行复位指令，以使该地址保持复位状态程序扫描剩余时间。

S（置位）和 R（复位）指令只有在 RLO 为“1”时才执行。RLO“0”对这些指令没有任何作用，并且指令中的指定地址保持不变。

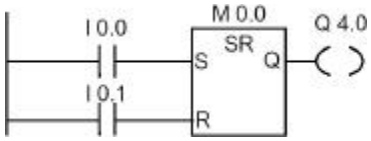
MCR（主控继电器）附属级

MCR 附属级只有在置位复位触发器放在激活的 MCR 区中时才能触发。在一个激活的 MCR 区内，如果 MCR 接通，则被寻址的位将如上所述被置位为“1”或复位为“0”。如果 MCR 断开，则指定地址的当前状态保持不变，与输入状态无关。

状态字

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

举例



如果输入 I0.0 的信号状态为“1”，输入 I0.1 的信号状态为“0”，则存储位 M0.0 将被置位，输出 Q4.0 为“1”。相反，如果输入 I0.0 的信号状态为“0”，输入 I0.1 的信号状态为“1”，则存储位 M0.0 将被复位，输出 Q4.0 为“0”。如果两个信号状态均为“0”，则无变化。如果两个信号状态均为“1”，则由于顺序之故，复位指令优先；M0.0 复位，Q4.0 为“0”。

如果举例在激活的 MCR 区内：

当 MCR 接通时，Q4.0 将如上所述被置位或复位。

当 MCR 断开时，Q4.0 将保持不变，与输入状态无关。

1.12 ---(N)--- RLO 下降沿检测

符号

<地址>

---(N)

参数	数据类型	存储区域	说明
<地址>	BOOL	I，Q，M，L，D	边沿存储位，存储RLO的前一信号状态

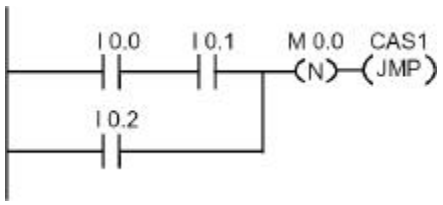
说明

---(N)--- （RLO 下降沿检测指令）可以检测地址从“1”到“0”的信号变化，并在操作之后以显示 RLO =“1”。将 RLO 的当前信号状态与“边沿存储位”地址的信号状态进行比较。如果操作之前地址的信号状态为“1”，并且 RLO 为“0”，则在操作之后，RLO 将为“1”（脉冲），所有其它的情况为“0”。操作之前的 RLO 存储在地址中。

状态字

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

举例



边沿存储位 M0.0 存储 RLO 的旧状态。如果 RLO 的信号从“1”变为“0”，则程序跳转至标号 CAS1 处。

1.13 ---(P)--- RLO 上升沿检测

符号

<地址>

---(P)---

参数	数据类型	存储区域	说明
<地址>	BOOL	I，Q，M，L，D	边沿存储位，存储RLO的前一信号状态

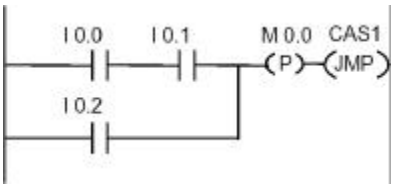
说明

---(P)---（RLO 上升沿检测指令）可以检测地址从“ 0 ”到“ 1 ”的信号变化，并在操作之后显示 RLO =“ 1 ”。将 RLO 的当前信号状态与“ 边沿存储位 ”地址的信号状态进行比较。如果操作之前地址的信号状态为“ 0 ”，并且 RLO 为“ 1 ”，则在操作之后，RLO 将为“ 1 ”（脉冲），所有其它的情况为“ 0 ”。操作之前的 RLO 存储在地址中。

状态字

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

举例



边沿存储位 M0.0 存储 RLO 的旧状态。如果 RLO 的信号从“ 0 ”变为“ 1 ”，则程序跳转至标号 CAS1 处。

1.14 ---(SAVE) 将 RLO 存入 BR 存储器

符号

---(SAVE)

说明

---(SAVE)（将 RLO 存入 BR 存储器指令）可以将 RLO 存储到状态字的 BR 位。首先检查位 /FC 是否复位。为此，BR 位的状态包括在下一程序段的与（AND）逻辑运算中。

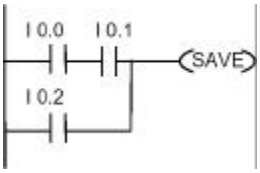
对于指令“ SAVE（保存）”（LAD，FBD，STL），适用以下所述，不适用手册和在线帮助中的建议使用：

我们不建议使用 SAVE，然后再检查相同块或附属块中的 BR 位，因为 BR 位可由在它们中间产生的许多指令进行修改。建议在退出块之前使用 SAVE 指令，这样 ENO 输出（= BR 位）就可设置为 RLO 位的值，可对块中是否有错误进行检查。

状态字

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

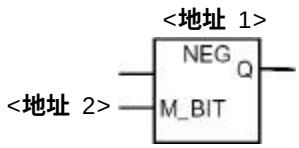
举例



梯形逻辑级的状态（=RLO**）被存储到** BR **位。**

1.15 NEG 地址下降沿检测

符号



参数	数据类型	存储区域	说明
<地址 1>	BOOL	I，Q，M，L，D	要扫描的信号
<地址 2>	BOOL	I，Q，M，L，D	M_BIT 边沿存储位，存储<地址 1>的前一信号状态
Q	BOOL	I，Q，M，L，D	输出为一个周期

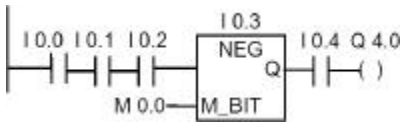
说明

NEG（地址下降沿检测指令）可以将<地址 1>的信号状态与存储在<地址 2>中的先前扫描的信号状态进行比较。如果当前的 RLO 状态为“1”，而先前的状态为“0”（上升沿检测），则在操作之后，RLO 位将为“1”。

状态字

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

举例

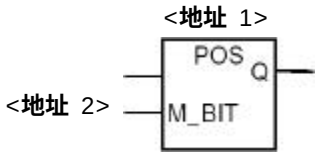


如果下列条件成立，则输出 Q4.0 **的信号状态为“**1**”：**

- **在输入** I0.0、I0.1 **和** I0.2 **的信号状态为“** 1 **”**
- **并且，在输入** I0.3 **有下降沿**
- **并且在输入** I0.4 **的信号状态为“** 1 **”**

1.16 POS **地址上升沿检测**

符号



参数	数据类型	存储区域	说明
<地址 1>	BOOL	I，Q，M，L，D	要扫描的信号
<地址 2>	BOOL	I，Q，M，L，D	M_BIT 边沿存储位，存储<地址 1>的前一信号状态
Q	BOOL	I，Q，M，L，D	输出为一个周期

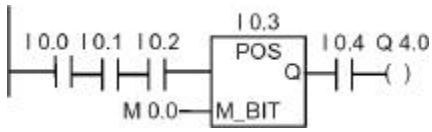
说明

POS（地址上升沿检测指令）可以将<地址 1>的信号状态与存储在<地址 2>中的先前扫描的信号状态进行比较。如果当前的 RLO 状态为“ 1 ”，而先前的状态为“ 0 ”（上升沿检测），则在操作之后，RLO 位将为“ 1 ”。

状态字

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

举例



如果下列条件成立，则输出 Q4.0 **的信号状态为“** 1 **”：**

- **在输入** I0.0、I0.1 **和** I0.2 **的信号状态为“** 1 **”**
- **并且，在输入** I0.3 **有上升沿**
- **并且在输入** I0.4 **的信号状态为“** 1 **”**

1.17 立即读操作

说明

对于立即读（Immediate Read）功能，必须如下面举例所示，生成符号程序段。

对于有时间限制的应用，可以以比每 OB1 扫描循环一次的正常情况快的速度，读取一个数字量输入的当前状态。立即读功能可以在扫描立即读逻辑程序级的同时，从输入模板获得一个数字量输入的状态。否则，当 I 存储区使用 P 存储状态更新时，必须等到下一 OB1 扫描循环结束。

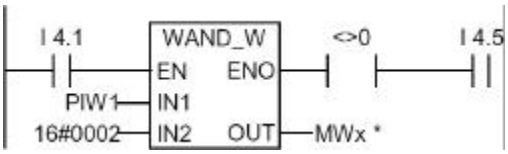
为了从输入模板立即读取一个输入，应使用外围输入（PI）存储区，而不使用输入（I）存储区。外围输入存储区可以作为一个字节、一个字或一个双字读取。因此，通过一个接点（位）元素，不能读取一个单独的数字量输入。

为了根据一个立即输入的状态，有条件地输送电压，应进行：

1. 由 CPU 读取包含相关输入数据的 PI 存储器中的字。
2. 然后将 PI 存储器中的字和一个常数进行与（AND）逻辑运算，如果输入位为“1”，则结果非“0”。
3. 检查累加器是否具有非“0”条件。

举例

带有外围输入 I1.1 立即读的梯形逻辑程序段



* 必须指定 MWx，以保存程序段。“x”可以是任何允许数值。

WAND_W 指令说明：

PIW1	0000000000101010
W#16#0002	0000000000000010
结果	0000000000000010

在该例中，立即输入 I1.1 与 I4.1 和 I4.5 串联。

字 PIW1 包含 I1.1 的立即状态。PIW1 与 W#16#0002 进行与（AND）逻辑运算。如果 PB1 中的 I1.1（第 2 位）为“1”，则结果非“0”。如果 WAND_W 指令的结果不等于“0”，则接点“A<>0”通过电压。

1.18 立即写操作

说明

对于立即写（Immediate Write）功能，必须如下面举例所示，生成符号程序段。

对于有时间限制的应用，可以以比每 OB1 扫描循环一次的正常情况快的速度，将一个数字量输出的当前状态发送到输出模板。立即写功能可以在扫描立即写逻辑程序级的同时，将一个数字量输出写入输出模板。否则，当 Q 存储区使用 P 存储状态更新时，必须等到下一 OB1 扫描循环结束。

为了将一个输出立即写入输出模板，应使用外围输出（PQ）存储区，而不使用输出（Q）存储区。外围输出存储区可以作为一个字节、一个字或一个双字读取。因此，通过一个线圈元素，不能更新一个单独的数字量输出。为了将一个数字量输出的状态立即写入输出模板，包含相关位的 Q 存储器的字节、字或双字可以有条件地复制到相应的 PQ 存储器中（直接输出模板地址）。

小心

- 由于 Q 存储器的整个字节被写入输出模板，当进行立即输出时，该字节中的所有输出位都将被更新。
- 如果一个输出位在不应发送到输出模板中的整个程序中出现中间状态（1/0），立即写功能会造成危险情况（输出瞬时脉冲）。
- 作为一般设计规则，在一个程序中，外部输出模板只能认为是一个线圈。如果遵守该设计规则，可以避免使用立即输出时的大多数潜在问题。

举例

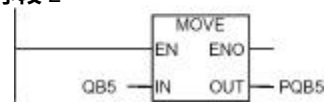
等效于立即写入外围数字量输出模板 5 通道 1 的梯形逻辑程序段。

寻址输出 Q 字节（QB5）的位状态可以修改，也可以保持不变。Q5.1 被赋给程序段 1 中 I0.1 的信号状态。QB5 被复制到相应的直接外围输出存储区（PQB5）。

字 PIW1 包含 I1.1 的立即状态。PIW1 与 W#16#0002 进行与（AND）逻辑运算。如果 PB1 中的 I1.1（第 2 位）为“ 1 ”，则结果非“ 0 ”。如果 WAND_W 指令的结果不等于“ 0 ”，则接点“ A<>0 ”通过电压。

程序段 1

程序段 2



在该例中，Q5.1 为所需立即输出位。

字节 PQB5 包含有位 Q5.1 的立即输出状态。

PQB5 中的其它 7 位也可以通过 MOVE（传送）指令更新。

2 比较指令

2.1 比较指令概述

说明

根据所选比较类型，对 IN1 **和** IN2 **进行比较：**

== IN1 **等于** IN2

<> IN1 **不等于** IN2

> IN1 **大于** IN2

< IN1 **小于** IN2

>= IN1 **大于等于** IN2

<= IN1 **小于等于** IN2

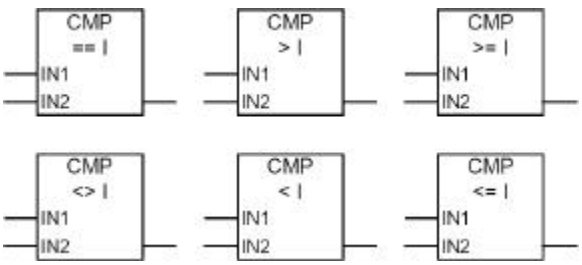
如果比较结果为真，则功能的 RLO **为“**1**”。如果串联使用比较元素可以通过与（**AND**）逻辑运算，或如果并联使用方块图可以通过或（**OR**）逻辑运算，将它与一个梯形逻辑级程序段的** RLO **链接。**

下述比较指令可供使用：

- CMP ?I **整数比较**
- CMP ?D **双整数比较**
- CMP ?R **实数比较**

2.2 CMP ? I 整数比较

符号



参数	数据类型	存储区域	说明
方块图输入	BOOL	I，Q，M，L，D	**先前逻辑运算的结果**
方块图输出	BOOL	I，Q，M，L，D	**只有在方块图输入的**RLO**为“**1**”时才能处理比较结果。**
IN1	INT	I，Q，M，L，D **或常数**	**第一个参与比较的数值**
IN2	INT	I，Q，M，L，D **或常数**	**第二个参与比较的数值**

说明

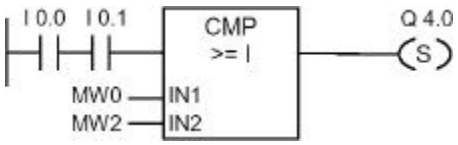
CMP ? I（整数比较指令）可以象一般的接点一样使用。它可以放在一般接点可以放的任何位置。根据所选比较类型，对 IN1 和 IN2 进行比较。

如果比较结果为真，则功能的 RLO 为“1”。如果串联使用方块图可以通过与（AND）逻辑运算，或如果并联使用方块图可以通过或（OR）逻辑运算，将它与整个梯形逻辑级的 RLO 链接。

状态字

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

举例

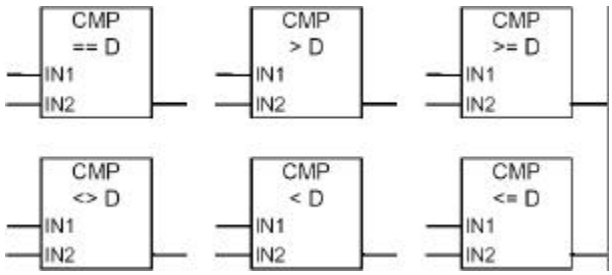


如果下列条件成立，则输出 Q4.0 置位：

- 在输入 I0.0 和 I0.1 的信号状态为“1”
- 并且 MW0 >= MW2

2.3 CMP ? D 双整数比较

符号



参数	数据类型	存储区域	说明
方块图输入	BOOL	I，Q，M，L，D	先前逻辑运算的结果
方块图输出	BOOL	I，Q，M，L，D	只有在方块图输入的RLO为“1”时才能处理比较结果。
IN1	DINT	I，Q，M，L，D 或常数	第一个参与比较的数值
IN2	DINT	I，Q，M，L，D 或常数	第二个参与比较的数值

说明

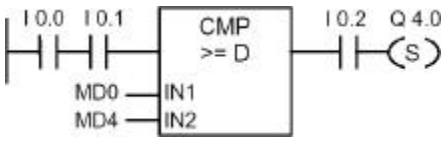
CMP ? D（双整数比较指令）可以象一般的接点一样使用。它可以放在一般接点可以放的任何位置。根据所选比较类型，对 IN1 和 IN2 进行比较。

如果比较结果为真，则功能的 RLO 为“1”。如果串联使用比较元素可以通过与（AND）逻辑运算，或如果并联使用方块图可以通过或（OR）逻辑运算，将它与一个梯形逻辑级程序段的 RLO 链接。

状态字

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

举例

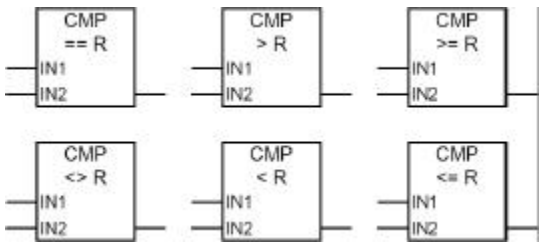


如果下列条件成立，则输出 Q4.0 置位：

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

2.4 CMP ? R 实数比较

符号



参数	数据类型	存储区域	说明
方块图输入	BOOL	I，Q，M，L，D	先前逻辑运算的结果
方块图输出	BOOL	I，Q，M，L，D	只有在方块图输入的RLO为“1”时才能处理比较结果。
IN1	INT	I，Q，M，L，D 或常数	第一个参与比较的数值
IN2	INT	I，Q，M，L，D 或常数	第二个参与比较的数值

说明

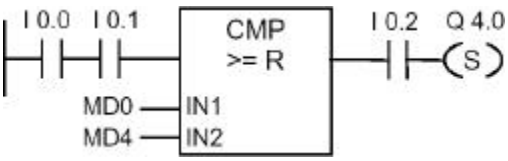
CMP ? R（实数比较指令）可以象一般的接点一样使用。它可以放在一般接点可以放的任何位置。根据所选比较类型，对 IN1 和 IN2 进行比较。

如果比较结果为真，则功能的 RLO 为“1”。如果串联使用方块图可以通过与（AND）逻辑运算，或如果并联使用方块图可以通过或（OR）逻辑运算，将它与整个梯形逻辑级的 RLO 链接。

状态字

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

举例



如果下列条件成立，则输出 Q4.0 置位：

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

3 转换指令

3.1 转换指令概述

说明

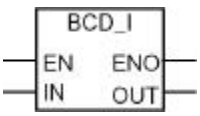
转换指令可以读取参数 IN **的内容，并进行转换或更改符号。其结果可以在参数** OUT **中查询。**

下述转换指令可供使用：

- BCD_I **BCD 码转换为整数**
- I_BCD **整数转换为** BCD **码**
- BCD_DI **BCD 码转换为双整数**
- I_DINT **整数转换为双整数**
- DI_BCD **双整数转换为** BCD **码**
- DI_REAL **双整数转换为浮点数**
- INV_I **整数的二进制反码**
- INV_DI **双整数的二进制反码**
- NEG_I **整数的二进制补码**
- NEG_DI **双整数的二进制补码**
- NEG_R **浮点数求反**
- ROUND **舍入为双整数**
- TRUNC **舍去小数取整为双整数**
- CEIL **上取整**
- FLOOR **下取整**

3.2 BCD_I BCD 码转换为整数

符号



参数	数据类型	存储区域	说明
EN	BOOL	I，Q，M，L，D	使能输入
ENO	BOOL	I，Q，M，L，D	使能输出
IN	WORD	I，Q，M，L，D	BCD 码
OUT	INT	I，Q，M，L，D	BCD 码转换的整数

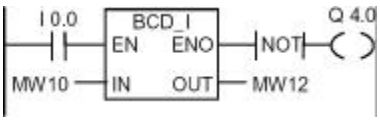
说明

BCD_I（BCD 码转换为整数指令）可以将输入参数 IN 的内容以三位数 BCD 代码（+/- 999）读入，并将这个数转换成整数（16 位）。其整数结果可以由参数 OUT 输出。ENO 和 EN 总是具有相同的信号状态。

状态字

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

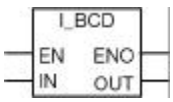
举例



如果输入 I0.0 为“1”，则 MW10 的内容作为三位 BCD 代码（+/- 999）读取，并转换成一个整数。其结果保存在 MW12 中。如果不执行转换（ENO =EN= 0），则输出 Q4.0 为“1”。

3.3 I_BCD 整数转换为 BCD 码

符号



参数	数据类型	存储区域	说明
EN	BOOL	I，Q，M，L，D	使能输入
ENO	BOOL	I，Q，M，L，D	使能输出
IN	INT	I，Q，M，L，D	整数
OUT	WORD	I，Q，M，L，D	整数的BCD码

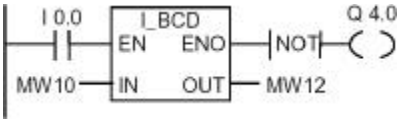
说明

I_BCD（整数转换为 BCD 码指令）可以将输入参数 IN 的内容以整数（16 位）读出，并转换为一个三位数 BCD 代码（+/- 999）。其结果可以由参数 OUT 输出。如果产生上溢，则 ENO 为“0”。

状态字

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

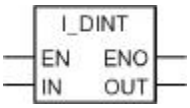
举例



如果输入端 I0.0 **为“** 1 **”，则** MW10 **的内容作为整数读入，并转换为一个三位** BCD **码。其结果保存在** MW12 **中。若产生上溢或没有执行指令（**I0.0 = 0**），则输出** Q4.0 **为“** 1 **”。**

3.4 I_DINT 整数转换为双整数

符号



参数	数据类型	存储区域	说明
EN	BOOL	I，Q，M，L，D	**使能输入**
ENO	BOOL	I，Q，M，L，D	**使能输出**
IN	INT	I，Q，M，L，D	**要转换的整数**
OUT	DINT	I，Q，M，L，D	**双整数结果**

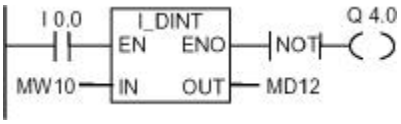
说明

I_DINT**（整数转换为双整数指令）可以将输入参数** IN **的内容以整数（**16 **位）读出，并转换为一个双整数（**32 **位）。其结果可以由参数** OUT **输出。**ENO **和** EN **总是具有相同的信号状态。**

状态字

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

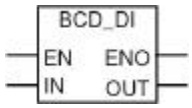
举例



如果 I0.0 **为“** 1 **”，则** MW10 **的内容作为整数读入，并转换为一个双整数。其结果保存在** MD12 **中。如果不执行转换（**ENO = EN = 0**），则输出** Q4.0 **为“** 1 **”。**

3.5 BCD_DI BCD 码转换为双整数

符号



参数	数据类型	存储区域	说明
EN	BOOL	I，Q，M，L，D	使能输入
ENO	BOOL	I，Q，M，L，D	使能输出
IN	DWORD	I，Q，M，L，D	BCD码
OUT	DINT	I，Q，M，L，D	由 BCD 码转换的双整数

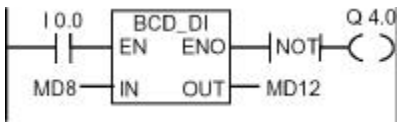
说明

BCD_DI （BCD 码转换为双整数指令）可以将输入参数 IN 的内容以七位数 BCD 代码（+/-9999999）读入，并将它转换成双整数（32 位）。其双整数结果可以由参数 OUT 输出。ENO 和 EN 总是具有相同的信号状态。

状态字

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

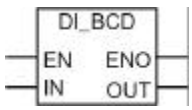
举例



如果 I0.0 为“1”，则 MD8 的内容作为七位 BCD 代码读取，并转换成一个双整数。其结果保存在 MD12 中。如果不执行转换（ENO = EN = 0），则输出 Q4.0 为“1”。

3.6 DI_BCD 双整数转换为 BCD 码

符号



参数	数据类型	存储区域	说明
EN	BOOL	I，Q，M，L，D	使能输入
ENO	BOOL	I，Q，M，L，D	使能输出
IN	DINT	I，Q，M，L，D	双整数
OUT	DWORD	I，Q，M，L，D	双整数的BCD码

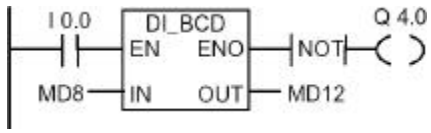
说明

DI_BCD（双整数转换为 BCD 码指令）可以将输入参数 IN 的内容以双整数（32 位）读出，并转换为一个七位数 BCD 代码（+/- 9999999）。其结果可以由参数 OUT 输出。如果产生上溢，则 ENO 为“ 0 ”。

状态字

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

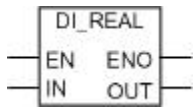
举例



如果 I0.0 为“ 1 ”，则 MD8 的内容作为双整数读取，并转换成一个七位 BCD 码。其结果保存在 MD12 中。若产生上溢或没有执行指令（I0.0 = 0），则输出 Q4.0 为“ 1 ”。

3.7 DI_REAL 双整数转换为浮点数

符号



参数	数据类型	存储区域	说明
EN	BOOL	I，Q，M，L，D	使能输入
ENO	BOOL	I，Q，M，L，D	使能输出
IN	DINT	I，Q，M，L，D	要转换的双整数
OUT	REAL	I，Q，M，L，D	浮点数结果

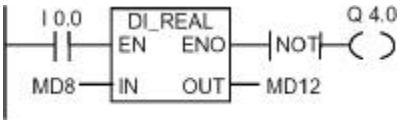
说明

DI_REAL（双整数转换为浮点数指令）可以将输入参数 IN 的内容以双整数读出，并将它转换为一个浮点数。其结果可以由参数 OUT 输出。ENO 和 EN 总是具有相同的信号状态。

状态字

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

举例



如果 I0.0 **为“** 1 **”，则** MD8 **的内容作为双整数读取，并转换成一个浮点数。其结果保存在** MD12 **中。如果不执行转换（**ENO = EN = 0**），则输出** Q4.0 **为“** 1 **”。**

3.8 INV_I 整数的二进制反码

符号

参数	数据类型	存储区域	说明
EN	BOOL	I，Q，M，L，D	**使能输入**
ENO	BOOL	I，Q，M，L，D	**使能输出**
IN	INT	I，Q，M，L，D	**整数输入值**
OUT	INT	I，Q，M，L，D	**整数**IN**的二进制反码**

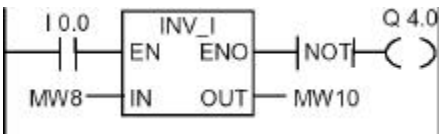
说明

INV_I**（整数的二进制反码指令）可以读取输入参数** IN **中的内容，并使用十六进制掩码** W#16#FFFF **执行布尔逻辑异或（**XOR**）功能。因此，该指令每一位均变为相反值。**ENO **和** EN **总是具有相同的信号状态。**

状态字

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

举例



如果 I0.0 **为“** 1 **”，则** MW8 **的每一位均被取反，例如：**

MW8 = 01000001 10000001 → MW10 = 10111110 01111110。

如果不执行转换（ENO = EN = 0**），则输出** Q4.0 **为“** 1 **”。**

3.9 INV_DI 双整数的二进制反码

符号

参数	数据类型	存储区域	说明
EN	BOOL	I，Q，M，L，D	使能输入
ENO	BOOL	I，Q，M，L，D	使能输出
IN	DINT	I，Q，M，L，D	双整数输入值
OUT	DINT	I，Q，M，L，D	双整数IN的二进制反码

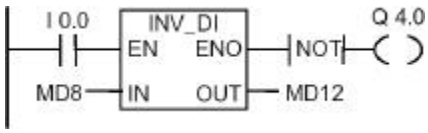
说明

INV_DI（双整数的二进制反码指令）可以读取输入参数 IN 中的内容，并使用十六进制掩码 W#16#FFFF 执行布尔逻辑异或（XOR）功能。因此，该指令每一位均变为相反值。ENO 和 EN 总是具有相同的信号状态。

状态字

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

举例



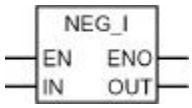
如果 I0.0 为“1”，则 MD8 的每一位均被取反，例如：

MD8 = F0FF FFF0 → MD12 = 0F00 000F。

如果不执行转换（ENO = EN = 0），则输出 Q4.0 为“1”。

3.10 NEG_I 整数的二进制补码

符号



参数	数据类型	存储区域	说明
EN	BOOL	I，Q，M，L，D	使能输入
ENO	BOOL	I，Q，M，L，D	使能输出
IN	INT	I，Q，M，L，D	整数输入值
OUT	INT	I，Q，M，L，D	整数IN的二进制补码

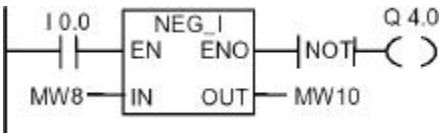
说明

NEG_I（整数的二进制补码指令）可以读取输入参数 IN 中的内容，并执行二进制补码操作。二进制补码指令相当于乘以（-1），并改变其符号（例如：从一个正值变为负值）。ENO 和 EN 总是具有相同的信号状态，以下例外：如果 EN 的信号状态为“1”，并发生上溢，则 ENO 的信号状态为“0”。

状态字

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

举例



如果 I0.0 为“1”，则 MW8 的数值连同相反符号由 OUT 参数输出到 MW10。

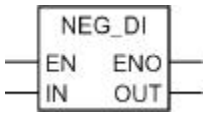
MW8 = + 10 → MW10 = - 10。

如果不执行转换（ENO = EN = 0），则输出 Q4.0 为“1”。

如果 EN 的信号状态为“1”，并发生上溢，则 ENO 的信号状态为“0”。

3.11 NEG_DI 双整数的二进制补码

符号



参数	数据类型	存储区域	说明
EN	BOOL	I，Q，M，L，D	使能输入
ENO	BOOL	I，Q，M，L，D	使能输出
IN	DINT	I，Q，M，L，D	双整数输入值
OUT	DINT	I，Q，M，L，D	双整数IN的二进制补码

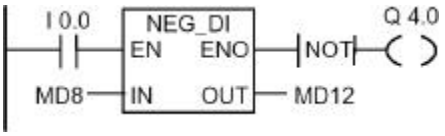
说明

NEG_DI（双整数的二进制补码指令）可以读取输入参数 IN 中的内容，并执行二进制补码操作。二进制补码指令相当于乘以（-1），并改变其符号（例如：从一个正值变为负值）。ENO 和 EN 总是具有相同的信号状态，以下例外：如果 EN 的信号状态为“1”，并发生上溢，则 ENO 的信号状态为“0”。

状态字

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

举例



如果 I0.0 **为“**1**”，则** MD8 **的数值连同相反符号由** OUT **参数输出到** MD12。

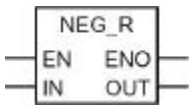
MD8 = + 1000 → MD12 = - 1000。

如果不执行转换（ENO = EN = 0），**则输出** Q4.0 **为“**1**”。**

如果 EN **的信号状态为“**1**”，并发生上溢，则** ENO **的信号状态为“**0**”。**

3.12 NEG_R 浮点数求反

符号



参数	数据类型	存储区域	说明
EN	BOOL	I，Q，M，L，D	**使能输入**
ENO	BOOL	I，Q，M，L，D	**使能输出**
IN	REAL	I，Q，M，L，D	**浮点数输入值**
OUT	REAL	I，Q，M，L，D	**带有负号的浮点数**IN

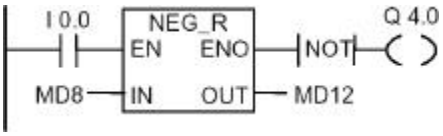
说明

NEG_R（浮点数求反指令）可以读取输入参数 IN 中的内容，并改变其符号。浮点数求反指令相当于乘以（-1），并改变其符号（例如：从一个正值变为负值）。ENO 和 EN 总是具有相同的信号状态。

状态字

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

举例



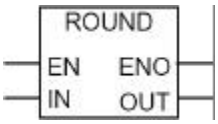
如果 I0.0 **为“** 1 **”，则** MD8 **的数值连同相反符号由** OUT **参数输出到** MD12。

MD8 = +6.234 →MD12 = -6.234。

如果不执行转换（ENO = EN = 0**），则输出** Q4.0 **为“** 1 **”。**

3.13 ROUND 舍入为双整数

符号



参数	数据类型	存储区域	说明
EN	BOOL	I，Q，M，L，D	**使能输入**
ENO	BOOL	I，Q，M，L，D	**使能输出**
IN	REAL	I，Q，M，L，D	**要舍入的值**
OUT	DINT	I，Q，M，L，D	**将**IN **舍入为最接近的整数**

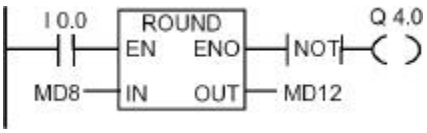
说明

ROUND**（舍入为双整数指令）可以将输入参数** IN **的内容以浮点数读入，并将它转换成一个双整数（**32 **位）。其结果为与输入数据最接近的整数（“最接近舍入”）。如果浮点数介于两个整数之间，则返回偶数。其结果可以由参数** OUT **输出。如果产生上溢，则** ENO **为“** 0 **”。**

状态字

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

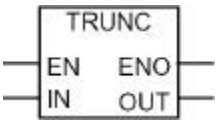
举例



如果 I0.0 **为“** 1 **”，则** MD8 **的内容作为浮点数读取，并转换成为最接近的双整数。该“最接近舍入”的结果保存在** MD12 **中。若产生上溢或没有执行指令（**I0.0 = 0**），则输出** Q4.0 **为“** 1 **”。**

3.14 TRUNC 舍去小数取整为双整数

符号



参数	数据类型	存储区域	说明
EN	BOOL	I，Q，M，L，D	使能输入
ENO	BOOL	I，Q，M，L，D	使能输出
IN	REAL	I，Q，M，L，D	要转换的浮点数
OUT	DINT	I，Q，M，L，D	IN 值的整数部分

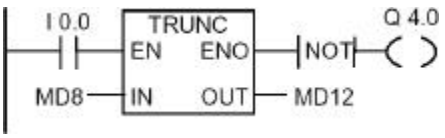
说明

TRUNC（舍去小数取整为双整数指令）可以将输入参数 IN 的内容以浮点数读入，并将它转换成一个双整数（32 位）。（“舍入到零方式”）其双整数结果可以由参数 OUT 输出。如果产生溢出，则 ENO 为“0”。

状态字

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

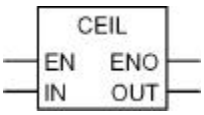
举例



如果 I0.0 为“1”，则 MD8 的内容作为实数读取，并转换成为一个双整数。结果是浮点数的整数部分，并保存在 MD12 中。若产生上溢或没有执行指令（I0.0 = 0），则输出 Q4.0 为“1”。

3.15 CEIL 上取整

符号



参数	数据类型	存储区域	说明
EN	BOOL	I，Q，M，L，D	使能输入
ENO	BOOL	I，Q，M，L，D	使能输出
IN	REAL	I，Q，M，L，D	要转换的浮点数
OUT	DINT	I，Q，M，L，D	最接近的较大双整数

说明

CEIL（上取整指令）可以将输入参数 IN 的内容以浮点数读入，并将它转换成一个双整数（32 位）。其结果为与输入数据最接近、大于浮点数的整数（“向正无穷大舍入”）。如果产生上溢，则 ENO 为“0”。

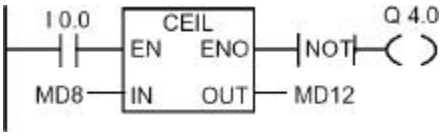
状态字

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

* **执行功能**（EN = 1）

** **不执行功能**（EN = 0）

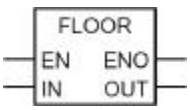
举例



如果 I0.0 为“1”，则 MD8 的内容作为浮点数读入，并使用 Round 功能转换成为一个双整数。其结果保存在 MD12 中。若产生上溢或没有执行指令（I0.0 = 0），则输出 Q4.0 为“1”。

3.16 FLOOR 下取整

符号



参数	数据类型	存储区域	说明
EN	BOOL	I，Q，M，L，D	使能输入
ENO	BOOL	I，Q，M，L，D	使能输出
IN	REAL	I，Q，M，L，D	要转换的浮点数
OUT	DINT	I，Q，M，L，D	最接近的较小双整数

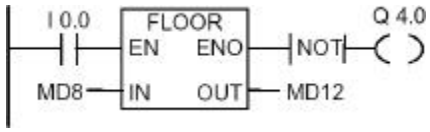
说明

FLOOR（下取整指令）可以将输入参数 IN 的内容以浮点数读入，并将它转换成一个双整数（32 位）。其结果为与输入数据的整数部分最接近、小于浮点数的整数（“向负无穷大舍入”）。如果产生上溢，则 ENO 为“0”。

状态字

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

举例



如果 I0.0 **为“** 1 **”，则** MD8 **的内容作为浮点数读取，并通过向负无穷大舍入方式转换成为一个双整数。其结果保存在** MD12 **中。若产生上溢或没有执行指令（** I0.0 = 0**），则输出** Q4.0 **为“** 1 **”。**

4 计数器指令

4.1 计数器指令概述

存储器区域

在 CPU 的存储器中，为计数器保留有存储区。该存储区为每一计数器地址保留一个 16 位的字。梯形逻辑指令集支持 256 个计数器。

计数器指令是访问计数器存储区的唯一功能。

计数值

计数器字的位 0 至位 9 包含二进制码的计数值。当计数器置位时，计数值传送至计数器字。计数值范围从 0 至 999。

通过使用以下计数器指令，可以在这一范围内改变计数值：

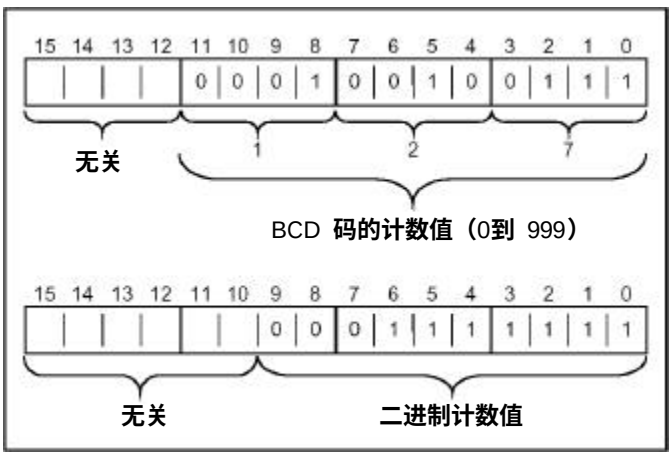
- S_CUD 加-减计数器
- S_CD 减计数器
- S_CU 加计数器
- ---(SC) 计数器线圈置位
- ---(CU) 加计数器线圈
- ---(CD) 减计数器线圈

计数器中的位组态

可用 0 至 999 范围内的数值，例如 127，为计数器设定初值，设定格式：C#127。C# 表示二 - 十进制格式（BCD 格式：四位一组表示一位十进制数值的二进制码）。

计数器字的位 0 至 11 位为二 - 十进制格式的计数值。

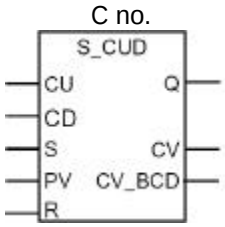
下图所示为在装载计数值 127 之后计数器的内容，以及计数器被设定后的计数器单元的内容。



4.2 S_CUD 加-减计数

符号

英文



德文

参数（英文）	参数（德文）	数据类型	存储区域	说明
C no.	Z no.	COUNTER	C	计数器标识号，范围与CPU有关
CU	ZV	BOOL	I，Q，M，L，D	加计数输入端
CD	ZR	BOOL	I，Q，M，L，D	减计数输入端
S	S	BOOL	I，Q，M，L，D	计数器预置输入端
PV	ZW	WORD	I，Q，M，L，D 或常数	计数器输入置的范围 0 –999，以C#<值> 形式表示。
PV	ZW	WORD	I，Q，M，L，D	计数器预置值
R	R	BOOL	I，Q，M，L，D	复位输入端
CV	DUAL	WORD	I，Q，M，L，D	当前计数器值，十六进制数值
CV_BCD	DEZ	WORD	I，Q，M，L，D	当前计数器值，BCD 码
Q	Q	BOOL	I，Q，M，L，D	计数器的状态

说明

S_CUD （加–减计数器）在 S 输入端出现上升沿时使用 PV 输入端的数值预置。如果 S 输入端为“1”，计数器则复位，计数值被置为“0”。如果输入端 CU 上的信号状态从“0”变为“1”，并且计数器的值小于“999”，则计数器加“1”。如果在输入端 CD 出现上升沿，并且计数器的值大于“0”，则计数器减“1”。

如果在两个计数输入端都有上升沿的话，则两种操作都执行，并且计数值保持不变。

如果计数器被置位，并且输入端 CU/CD 上的 RLO = 1，计数器将相应地在下一扫描循环计数，即使没有从上升沿到下降沿的变化或从下降沿到上升沿的变化。

如果计数值大于“0”，则输出 Q 上的信号状态为“1”；如果计数值等于“0”，则输出 Q 上的信号状态为“0”。

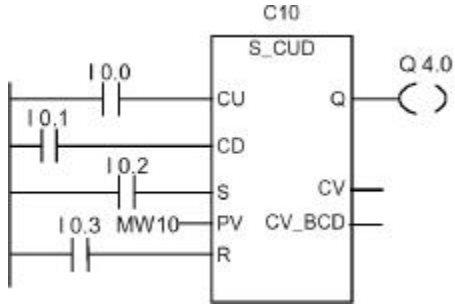
状态字

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

注意

应避免在几个程序中使用一个计数器（否则会出现计数错误）。

举例



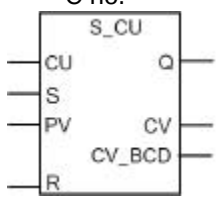
如果 I0.2 从“0”变为“1”，计数器使用 MW10 的值预置。如果 I0.0 的信号状态从“0”变为“1”，计数器 C10 的值将加“1”。C10 的值等于“999”除外。如果 I0.1 从“0”变为“1”，C10 将减“1”。C10 的值等于“0”除外。如果 C10 不等于“0”，则 Q4.0 为“1”。

4.3 S_CU 加计数器

符号

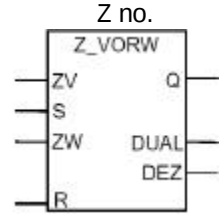
英文

C no.



德文

Z no.



参数（英文）	参数（德文）	数据类型	存储区域	说明
C no.	Z no.	COUNTER	C	计数器标识号，范围与CPU有关
CU	ZV	BOOL	I，Q，M，L，D	加计数输入端
S	S	BOOL	I，Q，M，L，D	计数器预置输入端
PV	ZW	WORD	I，Q，M，L，D 或常数	计数器输入置的范围 0 –999，以C#<值> 形式表示。
PV	ZW	WORD	I，Q，M，L，D	计数器预置值
R	R	BOOL	I，Q，M，L，D	复位输入端
CV	DUAL	WORD	I，Q，M，L，D	当前计数器值，十六进制数值
CV_BCD	DEZ	WORD	I，Q，M，L，D	当前计数器值，BCD 码
Q	Q	BOOL	I，Q，M，L，D	计数器的状态

说明

S_CU（加计数器）在输入端 S 出现上升沿时使用输入端 PV 上的数值预置。

如果在输入端 R 上的信号状态为“1”，则计数器复位，计数值被置为“0”。

如果输入端 CU 上的信号状态从“0”变为“1”，并且计数器的值小于“999”，则计数器加“1”。

如果计数器被置位，并且输入端 CU 上的 RLO = 1，计数器将相应地在下一扫描循环计数，即使没有从上升沿到下降沿的变化或从下降沿到上升沿的变化。

如果计数值大于“0”，则输出 Q 上的信号状态为“1”；如果计数值等于“0”，则输出 Q 上的信号状态为“0”。

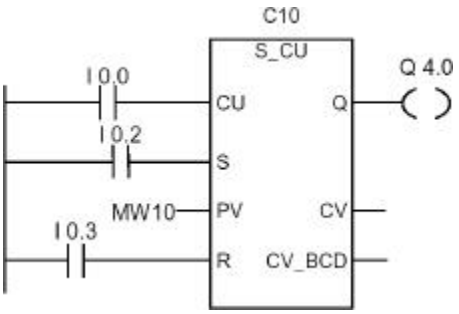
状态字

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

注意

应避免在几个程序点使用一个计数器（否则会出现计数错误）。

举例



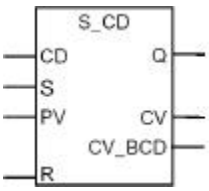
如果 I0.2 从“0”变为“1”，计数器使用 MW10 的值预置。如果 I0.0 的信号状态从“0”变为“1”，计数器 C10 的值将加“1”。C10 的值等于“999”除外。如果 C10 不等于“0”，则 Q4.0 为“1”。

4.4 S_CD 减计数器

符号

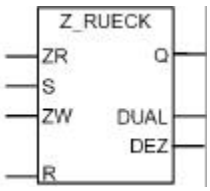
英文

C no.



德文

Z no.



参数（英文）	参数（德文）	数据类型	存储区域	说明
C no.	Z no.	COUNTER	C	计数器标识号，范围与CPU有关
CD	ZR	BOOL	I，Q，M，L，D	减计数输入端
S	S	BOOL	I，Q，M，L，D	计数器预置输入端
PV	ZW	WORD	I，Q，M，L，D 或常数	计数器输入置的范围 0 –999，以C#<值> 形式表示。
PV	ZW	WORD	I，Q，M，L，D	计数器预置值
R	R	BOOL	I，Q，M，L，D	复位输入端
CV	DUAL	WORD	I，Q，M，L，D	当前计数器值，十六进制数值
CV_BCD	DEZ	WORD	I，Q，M，L，D	当前计数器值，BCD 码
Q	Q	BOOL	I，Q，M，L，D	计数器的状态

说明

S_CD（减计数器）在输入端 S 出现上升沿时使用输入端 PV 上的数值预置。

如果在输入端 R 上的信号状态为“1”，则计数器复位，计数值被置为“0”。

如果输入端 CD 上的信号状态从“0”变为“1”，并且计数器的值大于“0”，则计数器减“1”。

如果计数器被置位，并且输入端 CD 上的 RLO = 1，计数器将相应地在下一扫描循环计数，即使没有从上升沿到下降沿的变化或从下降沿到上升沿的变化。

如果计数值大于“0”，则输出 Q 上的信号状态为“1”；如果计数值等于“0”，则输出 Q 上的信号状态为“0”。

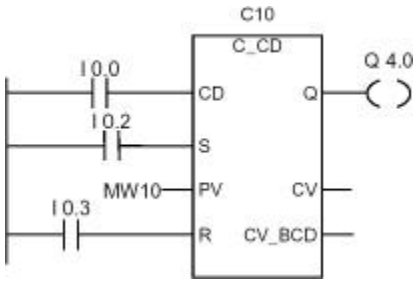
状态字

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

注意

应避免在几个程序点使用一个计数器（否则会出现计数错误）。

举例



如果 I0.2 从“0”变为“1”，计数器使用 MW10 的值预置。如果 I0.0 的信号状态从“0”变为“1”，计数器 C10 的值将减“1”。C10 的值等于“0”除外。如果 C10 不等于“0”，则 Q4.0 为“1”。

4.5 ---(SC) 计数器置初值

符号

英文	德文
<C no.>	<Z no.>
---(SC)	---(SZ)
<预置值>	<预置值>

参数（英文）	参数（德文）	数据类型	存储区域	说明
<C no.>	<Z no.>	COUNTER	C	要预置数值的计数器编号
<预置值>	<预置值>	WORD	I，Q，M，L，D 或常数	预置BCD 码值（0 - 999）

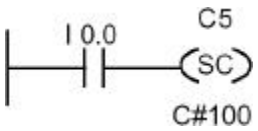
说明

---(SC)（计数器置初值指令）只有在 RLO 出现上升沿时才执行。同时，将预置值传送到指定的计数器。

状态字

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

举例



如果在输入端 I0.0（从 “ 0 ” 变为 “ 1 ”）出现上升沿，则计数器 C5 预置数值 “ 100 ”。如果没有出现上升沿，则计数器 C5 的值保持不变。

4.6 ---(CU) 加计数器线圈

符号

英文	德文
<C no.>	<Z no.>
---(CU)	---(ZV)

参数（英文）	参数（德文）	数据类型	存储区域	说明
<C no.>	<Z no.>	COUNTER	C	计数器标识号，范围与CPU有关

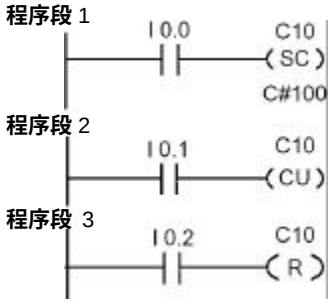
说明

---(CU)（加计数器线圈指令）在 RLO 出现上升沿并且计数器的值小于“ 999 ”时，则使指定计数器的值加“ 1 ”。如果在 RLO 没有出现上升沿，或计数器的值已经为“ 999 ”，则计数器的值保持不变。

状态字

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

举例



如果输入端 I0.0 的信号状态从“ 0 ”变为“ 1 ”（RLO 出现上升沿），则预置值“ 100 ”装入计数器 C10。

如果输入端 I0.1 的信号状态从“ 0 ”变为“ 1 ”(在 RLO 出现上升沿)，则计数器 C10 的值将加“ 1 ”。C10 的值等于“ 999 ”除外。如果在 RLO 没有出现上升沿，则计数器 C10 的值保持不变。

如果 I0.2 的信号状态为“ 1 ”，则计数器 C10 复位为“ 0 ”。

4.7 ---(CD) 减计数器线圈

符号

英文	德文
<C no.>	<Z no.>
---(CD)	---(ZD)

参数（英文）	参数（德文）	数据类型	存储区域	说明
<C no.>	<Z no.>	COUNTER	C	计数器标识号，范围与CPU有关

说明

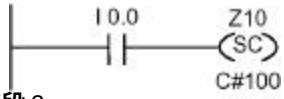
---(CD)（加计数器线圈指令）在 RLO 出现上升沿并且计数器的值大于“ 0 ”时，则使指定计数器的值减“ 1 ”。如果在 RLO 没有出现上升沿，或计数器的值已经为“ 0 ”，则计数器的值保持不变。

状态字

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

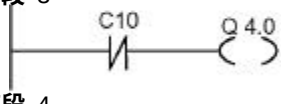
举例

程序段 1



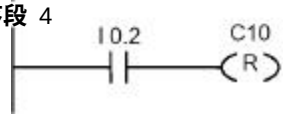
程序段 2

程序段 3



“ 0 ”计数值监测器

程序段 4



如果输入端 I0.0 的信号状态从“ 0 ”变为“ 1 ”（RLO 出现上升沿），则预置值“ 100 ”装入计数器 C10。

如果输入端 I0.1 的信号状态从“ 0 ”变为“ 1 ”（在 RLO 出现上升沿），则计数器 C10 的值将减“ 1 ”。C10 的值等于“ 0 ”除外。如果在 RLO 没有出现上升沿，则计数器 C10 的值保持不变。

如果计数值 = 0，则 Q4.0 接通。

如果 I0.2 的信号状态为“ 1 ”，则计数器 C10 复位为“ 0 ”。

5 数据块指令

5.1 ---(OPN) 打开数据块：DB 或 DI

符号

<DB 号> 或 <DI 号>

---(OPN)

参数	数据类型	存储区域	说明
<DB 号> <DI 号>	BLOCK_DB	DB，DI	DB/DI 的编号；范围与 CPU 有关

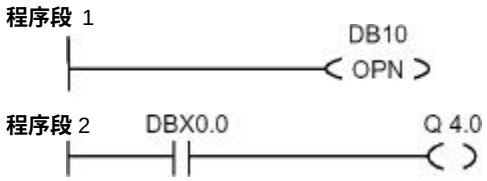
说明

---(OPN)（打开数据块指令）可以打开一个共享数据块（DB）或背景数据块（DI）。---(OPN)功能是一种数据块无条件调用功能。数据块的编号被传送到 DB 或 DI 寄存器。之后，DB 和 DI 指令根据寄存器的内容访问相应的数据块。

状态字

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

举例



数据块 10（DB10）被打开。接点地址（DBX0.0）指的是数据块 DB10 中包含的当前数据记录的数据字节 0 的 0 位。该位的信号状态被赋值给输出 Q4.0。

6 逻辑控制指令

6.1 逻辑控制指令概述

说明

逻辑控制指令可以用于所有逻辑块：组织块（OB），功能块（FB）和功能（FC）。

可执行下列功能的逻辑控制指令：

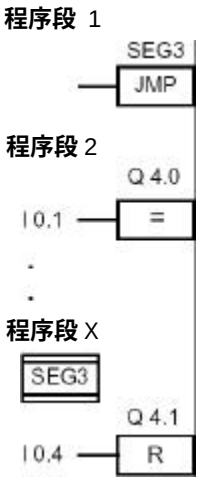
- ---(JMP)--- **无条件跳转**
- ---(JMP)--- **条件跳转**
- ---(JMPN)--- **若非则跳转**

地址标号

跳转指令的地址是一个标号。一个标号最多由四个字符组成。第一个字符必须是字母表中的一个字母，其它字符可以是字母，也可以是数字（例如 SEG3）。跳转标号指明想使程序跳转到的目的地。

目的地标号

目的地标号必须在一个程序段的开始。通过从梯形逻辑浏览器中选择 LABEL 的方法，在程序段开始处输入目的地标号。将出现一个空方块。在方块中，输入标号名。



6.2 ---(JMP)--- 无条件跳转

符号

<标号名>

---(JMP)

说明

---(JMP)（RLO = 1 时的块内跳转指令）在最左端电源线和该操作之间无其他的 LAD 元素时，用作绝对跳转功能。

对于每一个 ---(JMP)，必须有一个目的地（标号）。

不执行在跳转指令和标号间的任何指令。

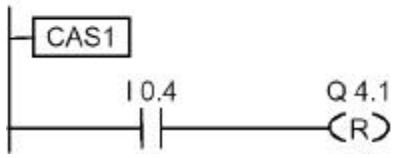
状态字

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

举例

程序段 1

程序段 X



每次都执行跳转，不执行跳转操作和跳转标号间的任何指令。

6.3 ---(JMP)--- 条件跳转

符号

<标号名>

---(JMP)

说明

---(JMP)（RLO = 1 **时的块内跳转指令）在前一逻辑操作的** RLO **为“** 1 **”时用作条件跳转功能。**

对于每一个 ---(JMP)**，必须有一个目的地（标号）。**

不执行在跳转指令和标号间的任何指令。

如果不执行条件跳转，在跳转操作后，RLO **将变为“** 1 **”。**

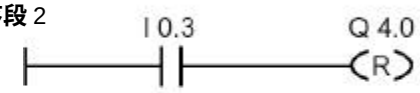
状态字

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

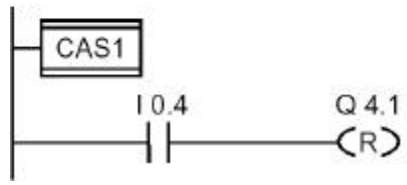
举例

程序段1

程序段 2



程序段 3



如果 I0.0 = **“** 1 **”，则跳转到标号** CAS1 **处执行。由于跳转，即使输入端** I0.3 **为逻辑“** 1 **”，也不执行输出** Q4.0 **复位指令。**

6.4 ---(JMPN) 若非则跳转

符号

<标号名>

---(JMPN)

说明

---(JMPN)（若非则跳转指令）相当于如果 RLO 为“ 0 ”，则执行“ 跳转至标号 ”指令。

对于每一个 ---(JMPN)，必须有一个目的地（标号）。

不执行在跳转指令和标号间的任何指令。

如果不执行条件跳转，在跳转操作后，RLO 将变为“ 1 ”。

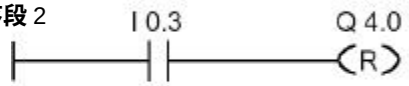
状态字

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

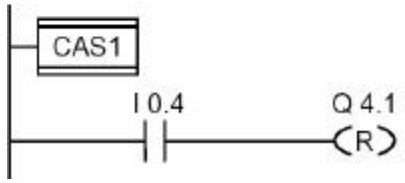
举例

程序段1

程序段 2



程序段 3



如果 I0.0 =“ 0 ”，则跳转到标号 CAS1 处执行。由于跳转，即使输入端 I0.3 为逻辑“ 1 ”，也不执行输出 Q4.0 复位指令。

6.5 LABEL 标号

符号

说明

LABEL **是一个跳转指令目的地的标识符。**

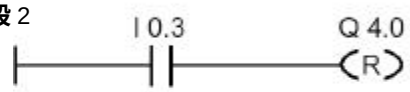
第一个字符必须是字母表中的一个字母,其它字符可以是字母,也可以是数字(例如 CAS1**)。**

对于每一个 ---(JMP) **或** ---(JMPN)**,必须有一个跳转标号**（LABEL）**。**

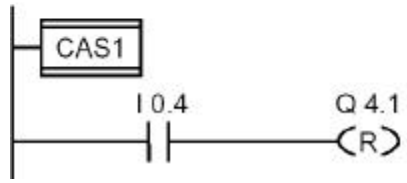
举例

程序段 1

程序段 2



程序段 3



如果 I0.0 =**“** 1 **”,执行跳转到标号** CAS1**。由于跳转,即使输入端** I0.3 **为逻辑“** 1 **”,也不执行输出** Q4.0 **复位指令。**

7 整数算术运算指令

7.1 整数算术运算指令概述

说明

使用整数算术运算指令，可以进行以下两个整数（16 位和 32 位）之间的运算：

- ADD_I **整数加法**
- SUB_I **整数减法**
- MUL_I **整数乘法**
- DIV_I **整数除法**
- ADD_DI **双整数加法**
- SUB_DI **双整数减法**
- MUL_DI **双整数乘法**
- DIV_DI **双整数除法**
- MOD_DI **回送余数的双整数**

7.2 判断整数算术运算指令后状态字的位

说明

整数算术运算指令可以影响以下状态字中的位：CC1 **和** CC0**，**OV **和** OS**。**

下表所示为使用了整数（16 位和 32 位）运算指令结果的状态字中各位的信号状态：

有效的结果范围	CC 1	CC 0	OV	OS
0**（零）**	0	0	0	*
16 **位：**-32 768 ≤ **结果** < 0**（负数）** 32 **位：**-2 147 483 648 ≤ **结果**< 0**（负数）**	0	1	0	*
16 **位：**32 767 ≥ **结果** > 0**（正数）** 32 **位：**2 147 483 647 ≥ **结果** > 0**（正数）**	1	0	0	*

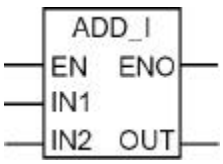
* OS **位不受指令结果的影响。**

无效的结果范围	A1	A0	OV	OS
上溢（加法） 16 位：结果 = -65536 32 位：结果 = -4 294 967 296	0	0	1	1
上溢（乘法） 16 位：结果 < -32 768（负数） 32 位：结果 < -2 147 483 648（负数）	0	1	1	1
上溢（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
上溢（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

运算	A1	A0	OV	OS
+D：结果 = -4 294 967 296	0	0	1	1
/D 或 MOD：被 0 除	1	1	1	1

7.3 ADD_I 整数加法

符号



参数	数据类型	存储区域	说明
EN	BOOL	I，Q，M，L，D	使能输入
ENO	BOOL	I，Q，M，L，D	使能输出
IN1	INT	I，Q，M，L，D 或常数	相加的第一个值
IN2	INT	I，Q，M，L，D 或常数	相加的第二个值
OUT	INT	I，Q，M，L，D	相加的结果

说明

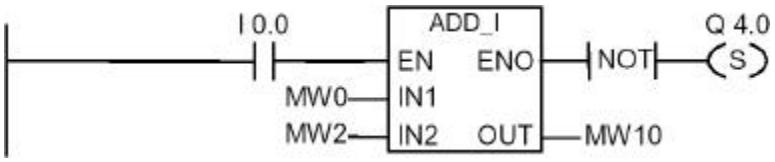
ADD_I（整数加法指令）可以由使能（EN）输入端的逻辑“1”信号激活。该指令可以使输入 IN1 和 IN2 相加，并在 OUT 扫描运算结果。如果结果在整数（16 位）的允许范围之外，则 OV 位和 OS 位为“1”，并且 ENO 为逻辑“0”，以防止执行通过 ENO 相连（级联布置）的该算术运算方块之后的其它功能。

请参见“判断整数算术运算指令后状态字的位”。

状态字

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

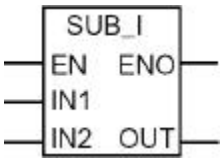
举例



如果 I0.0 = “1”**，则** ADD_I **方块激活。**MW0 + MW2 **相加的结果放入** MW10 **中。如果结果在整数的允许范围之外，则输出** Q4.0 **置位。**

7.4 SUB_I 整数减法

符号



参数	数据类型	存储区域	说明
EN	BOOL	I，Q，M，L，D	**使能输入**
ENO	BOOL	I，Q，M，L，D	**使能输出**
IN1	INT	I，Q，M，L，D **或常数**	**被减数**
IN2	INT	I，Q，M，L，D **或常数**	**减数**
OUT	INT	I，Q，M，L，D	**相减的结果**

说明

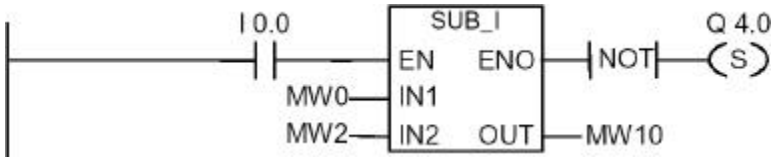
SUB_I（**整数减法指令**）**可以由使能（**EN**）输入端的逻辑“**1**”信号激活。该指令可以使输入** IN1 **减去** IN2**，并在** OUT **扫描运算结果。如果结果在整数（**16 **位）的允许范围之外，则** OV **位和** OS **位为“**1**”，并且** ENO **为逻辑“**0**”，以防止执行通过** ENO **相连（级联布置）的该算术运算方块之后的其它功能。**

请参见“判断整数算术运算指令后状态字的位”。

状态字

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

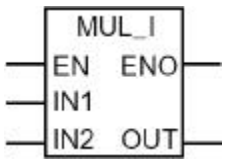
举例



如果 I0.0 = “ 1 ”，**则** SUB_I **方块激活。** MW0 - MW2 **相减的结果放入** MW10 **中。如果结果在整数的允许范围之外或输入** I0.0 = “ 0 ”，**则输出** Q4.0 **置位。**

7.5 MUL_I 整数乘法

符号



参数	数据类型	存储区域	说明
EN	BOOL	I，Q，M，L，D	使能输入
ENO	BOOL	I，Q，M，L，D	使能输出
IN1	INT	I，Q，M，L，D 或常数	被乘数
IN2	INT	I，Q，M，L，D 或常数	乘数
OUT	INT	I，Q，M，L，D	相乘的结果

说明

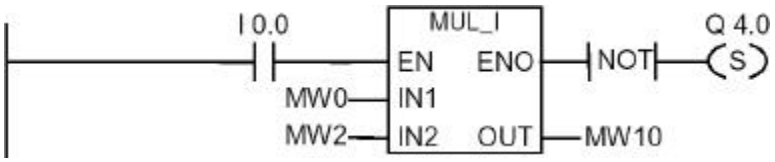
MUL_I（整数乘法指令）可以由使能（EN）输入端的逻辑 “ 1 ” 信号激活。该指令可以使输入 IN1 和 IN2 相乘，并在 OUT 扫描运算结果。如果结果在整数（16 位）的允许范围之外，则 OV 位和 OS 位为 “ 1 ”，并且 ENO 为逻辑 “ 0 ”，以防止执行通过 ENO 相连（级联布置）的该算术运算方块之后的其它功能。

请参见 “ 判断整数算术运算指令后状态字的位 ” 。

状态字

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

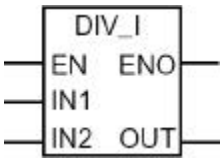
举例



如果 I0.0 = “1”，则 MUL_I 方块激活。MW0 x MW2 相乘的结果放入 MD10 中。如果结果在整数的允许范围之外，则输出 Q4.0 置位。

7.6 DIV_I 整数除法

符号



参数	数据类型	存储区域	说明
EN	BOOL	I，Q，M，L，D	使能输入
ENO	BOOL	I，Q，M，L，D	使能输出
IN1	INT	I，Q，M，L，D 或常数	被除数
IN2	INT	I，Q，M，L，D 或常数	除数
OUT	INT	I，Q，M，L，D	相除的结果

说明

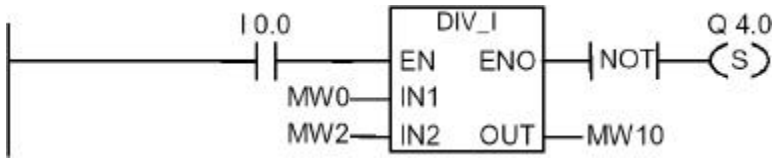
DIV_I（整数除法指令）可以由使能（EN）输入端的逻辑“1”信号激活。该指令可以使输入 IN1 被 IN2 除，并在 OUT 扫描运算结果。如果结果在整数（16 位）的允许范围之外，则 OV 位和 OS 位为“1”，并且 ENO 为逻辑“0”，以防止执行通过 ENO 相连（级联布置）的该算术运算方块之后的其它功能。

请参见“判断整数算术运算指令后状态字的位”。

状态字

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

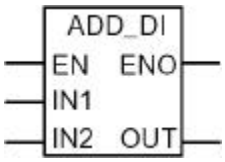
举例



如果 I0.0 = “1”，则 DIV_I 方块激活。MW0 被 MW2 除的结果放入 MW10 中。如果结果在整数的允许范围之外，则输出 Q4.0 置位。

7.7 ADD_DI 双整数加法

符号



参数	数据类型	存储区域	说明
EN	BOOL	I，Q，M，L，D	使能输入
ENO	BOOL	I，Q，M，L，D	使能输出
IN1	DINT	I，Q，M，L，D 或常数	相加的第一个值
IN2	DINT	I，Q，M，L，D 或常数	相加的第二个值
OUT	DINT	I，Q，M，L，D	相加的结果

说明

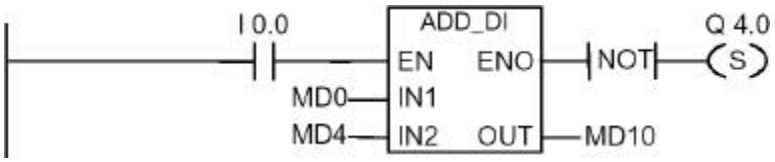
ADD_DI（双整数加法指令）可以由使能（EN）输入端的逻辑“1”信号激活。该指令可以使输入 IN1 和 IN2 相加，并在 OUT 扫描运算结果。如果结果在双整数（32 位）的允许范围之外，则 OV 位和 OS 位为“1”，并且 ENO 为逻辑“0”，以防止执行通过 ENO 相连（级联布置）的该算术运算方块之后的其它功能。

请参见“判断整数算术运算指令后状态字的位”。

状态字

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

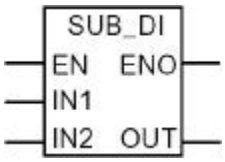
举例



如果 I0.0 = “1”，则 ADD_DI 方块激活。MD0 + MD4 相加的结果放入 MD10 中。如果结果在双整数的允许范围之外，则输出 Q4.0 置位。

7.8 SUB_DI 双整数减法

符号



参数	数据类型	存储区域	说明
EN	BOOL	I，Q，M，L，D	使能输入
ENO	BOOL	I，Q，M，L，D	使能输出
IN1	DINT	I，Q，M，L，D 或常数	被减数
IN2	DINT	I，Q，M，L，D 或常数	减数
OUT	DINT	I，Q，M，L，D	相减的结果

说明

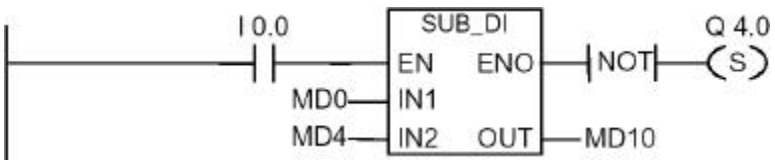
SUB_DI（双整数减法指令）可以由使能（EN）输入端的逻辑“1”信号激活。该指令可以使输入 IN1 减去 IN2，并在 OUT 扫描运算结果。如果结果在双整数（32 位）的允许范围之外，则 OV 位和 OS 位为“1”，并且 ENO 为逻辑“0”，以防止执行通过 ENO 相连（级联布置）的该算术运算方块之后的其它功能。

请参见“判断整数算术运算指令后状态字的位”。

状态字

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

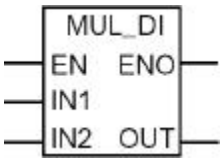
举例



如果 I0.0 = “1”，则 SUB_DI 方块激活。MD0 - MD4 相减的结果放入 MD10 中。如果结果在双整数的允许范围之外，则输出 Q4.0 置位。

7.9 MUL_DI 双整数乘法

符号



参数	数据类型	存储区域	说明
EN	BOOL	I，Q，M，L，D	使能输入
ENO	BOOL	I，Q，M，L，D	使能输出
IN1	DINT	I，Q，M，L，D 或常数	被乘数
IN2	DINT	I，Q，M，L，D 或常数	乘数
OUT	DINT	I，Q，M，L，D	相乘的结果

说明

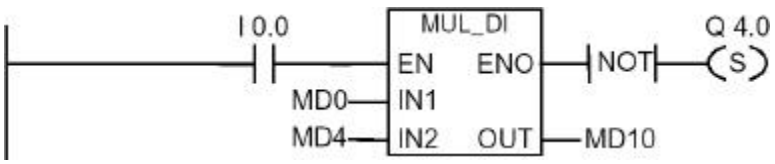
MUL_DI（双整数乘法指令）可以由使能（EN）输入端的逻辑“1”信号激活。该指令可以使输入 IN1 和 IN2 相乘，并在 OUT 扫描运算结果。如果结果在双整数（32 位）的允许范围之外，则 OV 位和 OS 位为“1”，并且 ENO 为逻辑“0”，以防止执行通过 ENO 相连（级联布置）的该算术运算方块之后的其它功能。

请参见“判断整数算术运算指令后状态字的位”。

状态字

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

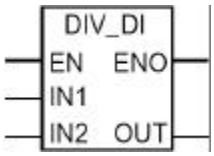
举例



如果 I0.0 = “1”，则 MUL_DI 方块激活。MD0 x MD4 相乘的结果放入 MD10 中。如果结果在双整数的允许范围之外，则输出 Q4.0 置位。

7.10 DIV_DI 双整数除法

符号



参数	数据类型	存储区域	说明
EN	BOOL	I，Q，M，L，D	使能输入
ENO	BOOL	I，Q，M，L，D	使能输出
IN1	DINT	I，Q，M，L，D 或常数	被除数
IN2	DINT	I，Q，M，L，D 或常数	除数
OUT	DINT	I，Q，M，L，D	相除的整数结果

说明

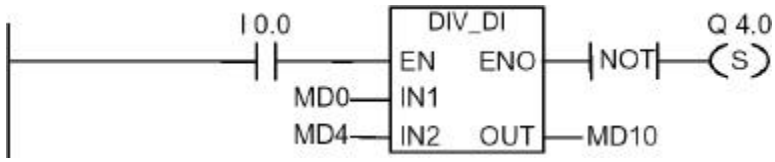
DIV_DI（双整数除法指令）可以由使能（EN）输入端的逻辑“1”信号激活。该指令可以使输入 IN1 被 IN2 除，并在 OUT 扫描运算结果。双整数除法元素不产生余数。如果结果在双整数（32 位）的允许范围之外，则 OV 位和 OS 位为“1”，并且 ENO 为逻辑“0”，以防止执行通过 ENO 相连（级联布置）的该算术运算方块之后的其它功能。

请参见“判断整数算术运算指令后状态字的位”。

状态字

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

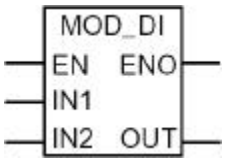
举例



如果 I0.0 = “1”，则 DIV_DI 方块激活。MD0 : MD4 的结果输出到 MD10 中。如果结果在双整数的允许范围之外，则输出 Q4.0 置位。

7.11 MOD_DI 回送余数的双整数

符号



参数	数据类型	存储区域	说明
EN	BOOL	I，Q，M，L，D	使能输入
ENO	BOOL	I，Q，M，L，D	使能输出
IN1	DINT	I，Q，M，L，D 或常数	被除数
IN2	DINT	I，Q，M，L，D 或常数	除数
OUT	DINT	I，Q，M，L，D	相除的余数

说明

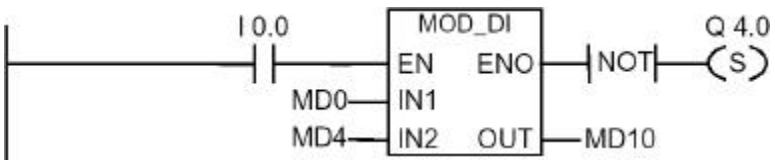
MOD_DI（回送余数的双整数指令）可以由使能（EN）输入端的逻辑“1”信号激活。该指令可以使输入 IN1 被 IN2 除，并在 OUT 扫描运算余数（小数）。如果结果在双整数（32 位）的允许范围之外，则 OV 位和 OS 位为“1”，并且 ENO 为逻辑“0”，以防止执行通过 ENO 相连（级联布置）的该算术运算方块之后的其它功能。

请参见“判断整数算术运算指令后状态字的位”。

状态字

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

举例



如果 I0.0 =“1”，则 MOD_DI 方块激活。MD0:MD4 相除的余数放入 MD10 中。如果余数在双整数的允许范围之外，则输出 Q4.0 置位。

8 浮点算术运算指令

8.1 浮点算术运算指令概述

说明

标准 IEEE 32 **位浮点数所属的数据类型称为** REAL**。应用浮点算术运算指令，可以对于两个** 32 **位标准** IEEE **浮点数完成以下算术运算：**

- ADD_R **实数加法**
- SUB_R **实数减法**
- MUL_R **实数乘法**
- DIV_R **实数除法**

应用浮点算术运算指令，可以对于一个 32 位标准 IEEE 浮点数完成以下算术运算：

- **完成一个浮点数的绝对值运算（**ABS**）**
- **完成一个浮点数的平方（**SQR**）和平方根（**SQRT**）运算**
- **完成一个浮点数的自然对数（**LN**）运算**
- **完成一个浮点数的基于** e **的指数运算（**EXP**），其中** e = 2.71828**。**
- **完成一个用** 32 **位标准** IEEE **浮点数表示的角度的以下三角函数运算：**
  - **正弦（**SIN**）和反正弦（**ASIN**）运算**
  - **余弦（**COS**）和反余弦（**ACOS**）运算**
  - **正切（**TAN**）和反正切（**ATAN**）运算**

请参见 " 判断浮点算术运算指令后状态字的位 "。

8.2 判断浮点算术运算指令后状态字的位

说明

浮点算术运算指令可以影响以下状态字中的位：CC 1 **和** CC 0**，**OV **和** OS**。**

下表所示为使用了浮点数（32 **位）运算指令结果的状态字中各位的信号状态：**

有效的结果范围	CC 1	CC 0	OV	OS
+0, -0	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	*

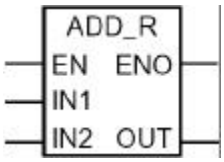
* OS **位不受指令结果的影响。**

无效的结果范围	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
上溢 **结果** > 3.402823E+38（**正数**）	1	0	1	1
不是有效的浮点数或者非法指令（输入值在有效范围之外）	1	1	1	1

8.3 基本指令

8.3.1 ADD_R 实数加法

符号



参数	数据类型	存储区域	说明
EN	BOOL	I，Q，M，L，D	**使能输入**
ENO	BOOL	I，Q，M，L，D	**使能输出**
IN1	REAL	I，Q，M，L，D **或常数**	**相加的第一个值**
IN2	REAL	I，Q，M，L，D **或常数**	**相加的第二个值**
OUT	REAL	I，Q，M，L，D	**相加的结果**

说明

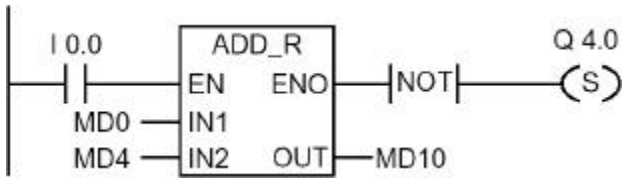
ADD_R（**实数加法指令**）**可以由使能**（EN）**输入端的逻辑“**1**”信号激活。该指令可以使输入** IN1 **和** IN2 **相加，并在** OUT **扫描运算结果。如果结果在浮点数的允许范围之外（上溢或下溢），则** OV **位和** OS **位为“**1**”，并且** ENO **为逻辑“**0**”，以防止执行通过** ENO **相连（级联布置）的该算术运算方块之后的其它功能。**

请参见“判断浮点算术运算指令后状态字的位”。

状态字

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

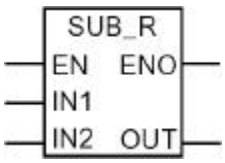
举例



如果 I0.0 = “ 1 ”，**则** ADD_R **方块激活。**MD0 + MD4 **相加的结果放入** MD10 **中。如果结果在浮点数的允许范围之外或程序语句没有执行（**I0.0 = “ 0 ”**），则输出** Q4.0 **置位。**

8.3.2 SUB_R 实数减法

符号



参数	数据类型	存储区域	说明
EN	BOOL	I，Q，M，L，D	**使能输入**
ENO	BOOL	I，Q，M，L，D	**使能输出**
IN1	REAL	I，Q，M，L，D **或常数**	**被减数**
IN2	REAL	I，Q，M，L，D **或常数**	**减数**
OUT	REAL	I，Q，M，L，D	**相减的结果**

说明

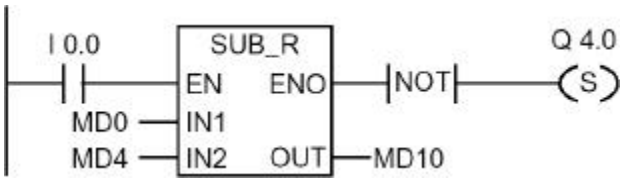
SUB_R**（实数减法指令）可以由使能（**EN**）输入端的逻辑“** 1 **”信号激活。该指令可以使输入** IN1 **减去** IN2**，并在** OUT **扫描运算结果。如果结果在浮点数的允许范围之外（上溢或下溢），则** OV **位和** OS **位为“** 1 **”，并且** ENO **为逻辑“** 0 **”，以防止执行通过** ENO **相连（级联布置）的该算术运算方块之后的其它功能。**

请参见“ 判断浮点算术运算指令后状态字的位 ”。

状态字

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

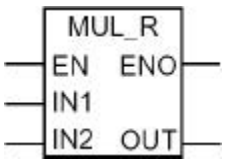
举例



如果 I0.0 = “1”，**则** SUB_R **方块激活。**MD0 - MD4 **相减的结果放入** MD10 **中。如果结果在浮点数的允许范围之外或程序语句没有执行，则输出** Q4.0 **置位。**

8.3.3 MUL_R 实数乘法

符号



参数	数据类型	存储区域	说明
EN	BOOL	I，Q，M，L，D	使能输入
ENO	BOOL	I，Q，M，L，D	使能输出
IN1	REAL	I，Q，M，L，D 或常数	被乘数
IN2	REAL	I，Q，M，L，D 或常数	乘数
OUT	REAL	I，Q，M，L，D	相乘的结果

说明

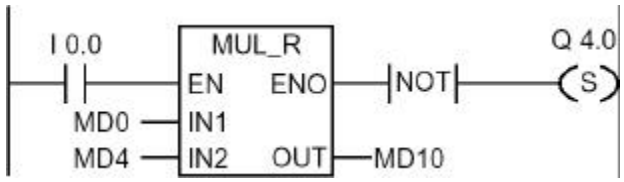
MUL_R（实数乘法指令）可以由使能（EN）输入端的逻辑“1”信号激活。该指令可以使输入 IN1 和 IN2 相乘，并在 OUT 扫描运算结果。如果结果在浮点数的允许范围之外（上溢或下溢），则 OV 位和 OS 位为“1”，并且 ENO 为逻辑“0”，以防止执行通过 ENO 相连（级联布置）的该算术运算方块之后的其它功能。

请参见“判断浮点算术运算指令后状态字的位”。

状态字

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

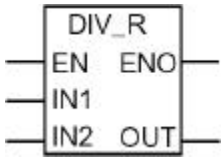
举例



如果 I0.0 = **“** 1 **”，则** MUL_R **方块激活。**MD0 x MD4 **相乘的结果放入** MD10 **中。如果结果在浮点数的允许范围之外或程序语句没有执行，则输出** Q4.0 **置位。**

8.3.4 DIV_R **实数除法**

符号



参数	数据类型	存储区域	说明
EN	BOOL	I，Q，M，L，D	**使能输入**
ENO	BOOL	I，Q，M，L，D	**使能输出**
IN1	REAL	I，Q，M，L，D **或常数**	**被除数**
IN2	REAL	I，Q，M，L，D **或常数**	**除数**
OUT	REAL	I，Q，M，L，D	**相除的结果**

说明

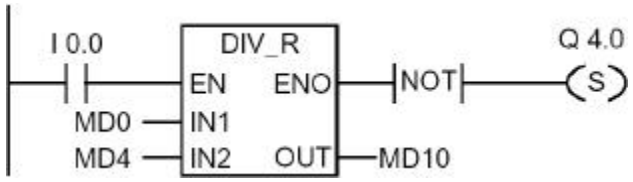
DIV_R**（实数除法指令）可以由使能（**EN**）输入端的逻辑“** 1 **”信号激活。该指令可以使输入** IN1 **被** IN2 **除，并在** OUT **扫描运算结果。如果结果在浮点数的允许范围之外（上溢或下溢），则** OV **位和** OS **位为“** 1 **”，并且** ENO **为逻辑“** 0 **”，以防止执行通过** ENO **相连（级联布置）的该算术运算方块之后的其它功能。**

请参见“判断浮点算术运算指令后状态字的位”。

状态字

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

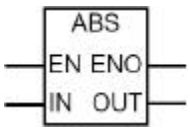
举例



如果 I0.0 = “1”**，则** DIV_R **方块激活。**MD0:MD4 **相除的结果放入** MD10 **中。如果结果在浮点数的允许范围之外或程序语句没有执行，则输出** Q4.0 **置位。**

8.3.5 ABS 浮点数绝对值运算

符号



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

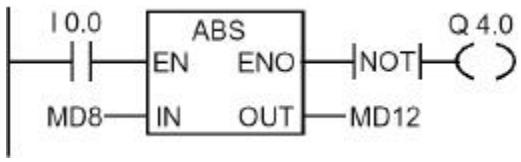
说明

ABS **可以完成一个浮点数的绝对值运算。**

状态字

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

举例



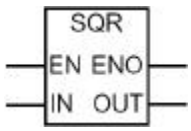
如果 I0.0 **为“**1**”，则** MD8 **的绝对值在** MD12 **端输出。**

MD8 = + 6.234 **的结果为** MD12 = 6.234**。如果不执行转换(**ENO = EN = 0**),则输出** Q4.0 **为“**1**”。**

8.4 扩展指令

8.4.1 SQR 浮点数平方

符号



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

说明

SQR 可以完成一个浮点数的平方运算。

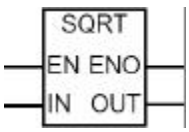
请参见“判断浮点算术运算指令后状态字的位”。

状态字

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

8.4.2 SQRT 浮点数平方根

符号



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

说明

SQRT 可以完成一个浮点数的平方根运算。当地址大于“0”时，结果是一个正数。例外情况：-0 的平方根为 -0。

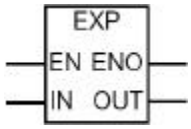
请参见“判断浮点算术运算指令后状态字的位”。

状态字

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

8.4.3 EXP 浮点数指数运算

符号



参数	数据类型	存储区域	说明
EN	BOOL	I，Q，M，L，D	**使能输入**
ENO	BOOL	I，Q，M，L，D	**使能输出**
IN	REAL	I，Q，M，L，D **或常数**	**输入值：浮点数**
OUT	REAL	I，Q，M，L，D	**输出值：浮点数的指数**

说明

EXP **可以完成一个浮点数基于** e =2.71828...**的指数运算。**

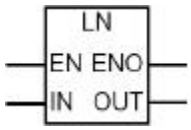
请参见“判断浮点算术运算指令后状态字的位”。

状态字

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

8.4.4 LN 浮点数自然对数运算

符号



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

说明

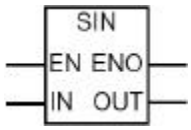
LN **可以完成一个浮点数的自然对数运算。**
请参见“判断浮点算术运算指令后状态字的位”。

状态字

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

8.4.5 SIN 浮点数正弦运算

符号



参数	数据类型	存储区域	说明
EN	BOOL	I，Q，M，L，D	**使能输入**
ENO	BOOL	I，Q，M，L，D	**使能输出**
IN	REAL	I，Q，M，L，D **或常数**	**输入值：浮点数**
OUT	REAL	I，Q，M，L，D	**输出值：浮点数的正弦值**

说明

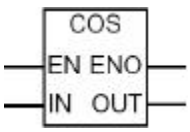
SIN **可以完成一个浮点数的正弦运算。在此，浮点数表示一个以弧度表示的角度。**
请参见“判断浮点算术运算指令后状态字的位”。

状态字

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

8.4.6 COS 浮点数余弦运算

符号



参数	数据类型	存储区域	说明
EN	BOOL	I，Q，M，L，D	**使能输入**
ENO	BOOL	I，Q，M，L，D	**使能输出**
IN	REAL	I，Q，M，L，D **或常数**	**输入值：浮点数**
OUT	REAL	I，Q，M，L，D	**输出值：浮点数的余弦值**

说明

COS **可以完成一个浮点数的余弦运算。在此，浮点数表示一个以弧度表示的角度。**

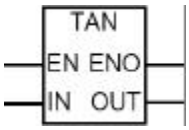
请参见“判断浮点算术运算指令后状态字的位”。

状态字

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

8.4.7 TAN 浮点数正切运算

符号



参数	数据类型	存储区域	说明
EN	BOOL	I，Q，M，L，D	**使能输入**
ENO	BOOL	I，Q，M，L，D	**使能输出**
IN	REAL	I，Q，M，L，D **或常数**	**输入值：浮点数**
OUT	REAL	I，Q，M，L，D	**输出值：浮点数的正切值**

说明

TAN **可以完成一个浮点数的正切运算。在此，浮点数表示一个以弧度表示的角度。**

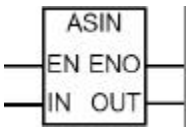
请参见“判断浮点算术运算指令后状态字的位”。

状态字

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

8.4.8 ASIN 浮点数反正弦运算

符号



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

说明

ASIN 可以完成一个在定义范围（-1≤输入值≤1）内的浮点数的反正弦运算。其结果是一个弧度表示的角度。其值在以下范围内：

-π/2 ≤ 输出值 ≤+π/2，

其中 π = 3.1415....

请参见“判断浮点算术运算指令后状态字的位”。

状态字

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

8.4.9 ACOS 浮点数反余弦运算

符号

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

说明

ACOS 可以完成一个在定义范围（-1≤输入值≤1）内的浮点数的反余弦运算。其结果是一个弧度表示的角度。其值在以下范围内：

0 ≤ 输出值≤+π，

其中 π = 3.1415....

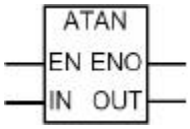
请参见“判断浮点算术运算指令后状态字的位”。

状态字

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

8.4.10 ATAN 浮点数反正切运算

符号



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

说明

ATAN 可以完成一个浮点数的反正切运算。其结果是一个以弧度表示的角度。其值在以下范围内：

$-\pi/2 \leq$ 输出值 $\leq +\pi/2$，

其中 π = 3.1415....

请参见“判断浮点算术运算指令后状态字的位”。

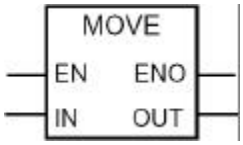
状态字

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

9 赋值指令

9.1 MOVE 赋值

符号



参数	数据类型	存储区域	说明
EN	BOOL	I，Q，M，L，D	使能输入
ENO	BOOL	I，Q，M，L，D	使能输出
IN	所有数据类型，长度可为 8 位、16 位或 32 位	I，Q，M，L，D 或常数	源数值
OUT	所有数据类型，长度可为 8 位、16 位或 32 位	I，Q，M，L，D	目的地址

说明

MOVE（赋值指令）可以由使能（EN）输入端的信号激活。将在输入端 IN 的特定值复制到输出端 OUT 上的特定地址中。ENO 和 EN 具有相同的逻辑状态。MOVE 只能复制 BYTE（字节）、WORD（字）或 DWORD（双字）数据对象。用户定义的数据类型（例如数组或结构）必须使用系统功能“ BLKMOVE ”（SFC 20）进行复制。

状态字

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

MCR（主控继电器）附属级

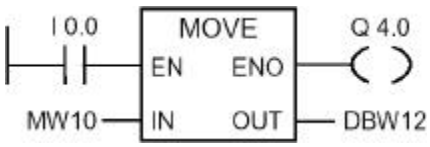
MCR 附属级只有在 Move 方块放在激活的 MCR 区中时才能触发。在一个激活的 MCR 区内，如果 MCR 接通，并且有信号经过使能输入，则被寻址的数据将如上所述进行复制。如果 MCR 断开，并执行了 MOVE 操作，逻辑“ 0 ”则被写入指定 OUT 地址，与当前 IN 的状态无关。

注意

如果将一个数值赋值给不同长度的数据类型，根据需要高位字节将被截去或填为“ 0 ”：

例如：双字	1111 1111	0000 1111	1111 0000	0101 0101
赋值	**结果**			
给一个双字：	1111 1111	0000 1111	1111 0000	0101 0101
给一个字节：				0101 0101
给一个字：			1111 0000	0101 0101
例如：字节				1111 0000
赋值	**结果**			
给一个字节：				1111 0000
给一个字：			0000 0000	1111 0000
给一个双字：	0000 0000	0000 0000	0000 0000	1111 0000

举例



如果 I0.0 = “1”，**则执行指令。**MW10 **的内容被复制到当前打开的数据块的数据字** 12 **中。**

如果执行指令，则 Q4.0 **为“**1**”。**

如果举例中的梯形逻辑级在激活的 MCR **区内：**

- **当** MCR **接通时，**MW10 **数据将如上所述被复制到** DBW12**。**
- **当** MCR **断开时，“**0**”被写入到** DBW12**。**

10 程序控制指令

10.1 程序控制指令概述

说明

下述程序控制指令可供使用：

- ---(CALL) **从线圈调用** FC/SFC**（无参数）**
- CALL_FB **从方块调用** FB
- CALL_FC **从方块调用** FC
- CALL_SFB **从方块调用** SFB
- CALL_SFC **从方块调用** SFC
- **调用多背景块**
- **从库中调用块**
- **使用** MCR **功能的重要注意事项**
- ---(MCR<) **主控继电器接通**
- ---(MCR>) **主控继电器断开**
- ---(MCRA) **主控继电器启动**
- ---(MCRD) **主控继电器停止**
- RET **返回**

10.2 ---(Call) **从线圈调用** FC/SFC**（无参数）**

符号

<FC/SFC **号**>

---(CALL)

参数	数据类型	存储区域	说明
<FC/SFC **号**>	BLOCK_FC BLOCK_SFC	-	FC/SFC **的编号；范围与** CPU **有关**

说明

---(Call)**（无参数调用** FC **或** SFC **指令）用来调用不带参数的功能（** FC**）或系统功能（** SFC**）。只有当** CALL **线圈的** RLO **为“** 1 **”时，才执行调用。如果执行** ---(Call)**，则**

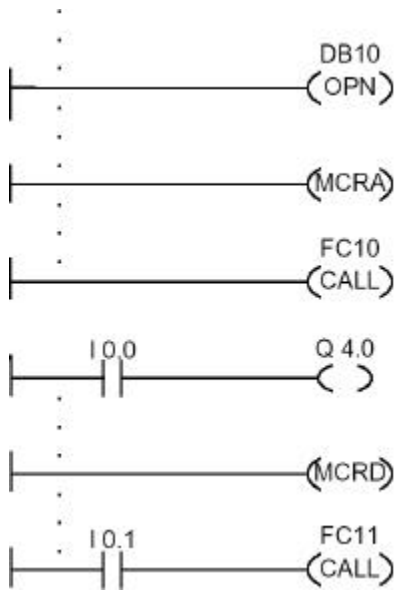
- **保存调用块所需要的返回地址**
- **将当前的本地数据范围变为以前的本地数据范围**
- **将** MA **位（**MCR **启动位）推至块堆栈中**
- **为被调用的功能生成新的本地数据范围**

在此之后，在被调用的 FC **或** SFC **中继续执行程序处理。**

状态字

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

举例



上图所示梯形逻辑级是由用户编写的一个功能块的程序段。在该功能块中，DB10 **被打开，**MCR **功能启动。如果执行** FC10 **的无条件调用，则执行以下功能：**

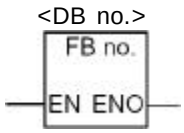
存储调用 FB **的返回地址以及用于** DB10 **和调用** FB **的背景数据块的选择数据。在** MCRA **指令中将** MA **位置为“**1**”，并将该位推入块堆栈中，然后为调用的块（**FC10**）将** MA **位复位为“**0**”。程序处理继续在** FC10 **中进行。如果** FC10 **需要** MCR **功能，必须在** FC10 **中重新启动它。当** FC10 **完成后，程序处理返回调用** FB。MA **位被保存，**DB10 **和用户编写的** FB **的背景数据块又成为当前的** DB，**与** FC10 **使用哪一个** DB **无关。通过将** I0.0 **的逻辑状态分配给输出** Q4.0，**程序继续下一梯形逻辑级。**FC11 **的调用是条件调用。只有在** I0.1 = **“**1**”时才执行调用。如果执行调用，传送程序控制的过程和从** FC11 **的返回过程与** FC10 **相同。**

注意

在返回调用块之后，以前打开的 DB **将不再总是打开。请仔细阅读** README **文件中的注意事项。**

10.3 CALL_FB 从方块调用 FB

符号



符号取决于 FB 的使用（是否有参数，以及有多少参数）。它必须具有 EN、ENO 以及 FB 的名称或编号。

参数	数据类型	存储区域	说明
EN	BOOL	I，Q，M，L，D	使能输入
ENO	BOOL	I，Q，M，L，D	使能输出
FB 号 DB 号	BLOCK_FB BLOCK_DB	-	FB/DB 的编号；范围与 CPU 有关

说明

CALL_FB（从方块调用功能块指令）在 EN 为“1”时执行。如果执行 CALL_FB，则

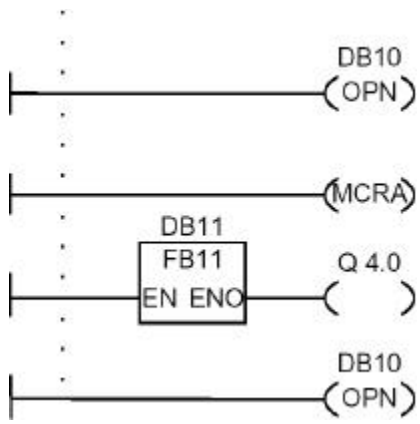
- 保存调用块所需要的返回地址
- 保存用于两个当前数据块（DB 和背景数据块）的选择数据
- 将当前的本地数据区变为以前的本地数据区
- 将 MA 位（MCR 启动位）推至块堆栈中
- 为被调用的功能块生成新的本地数据范围

在此之后，在被调用的功能块中继续执行程序处理。BR 位被扫描，以找到 ENO。用户必须使用 ---(SAVE) 将所需状态（错误评价）赋值给调用块中的 BR 位。

状态字

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

举例



上图所示梯形逻辑级是由用户编写的一个功能块的程序段。在该功能块中，DB10 被打开，MCR 功能启动。如果执行 FC11 的无条件调用，则执行以下功能：

存储调用 FB 的返回地址以及用于 DB10 和调用 FB 的背景数据块的选择数据。在 MCRA 指令中将 MA 位置为“1”，并将该位推入块堆栈中，然后为调用的块（FB11）将 MA 位复位为“0”。程序处理继续在 FB11 中进行。如果 FB11 需要 MCR 功能，必须在 FB11 中重新启动它。RLO 的状态必须使用指令 ---(SAVE) 保存在 BR 位中，以便能够在调用 FB 中评价错误。当 FB11 完成后，程序处理返回调用 FB。MA 位被重新恢复，用户编写 FB 的背景数据块又被打开。如果 FB11 能正确处理，则 ENO =“1”，并因此，Q4.0 =“1”。

注意

当打开一个 FB 或 SFB 时，以前打开的 DB 编号将丢失。所需 DB 必须重新打开。

10.4 CALL_FC 从方块调用 FC

符号

符号取决于 FC 的使用（是否有参数，以及有多少参数）。它必须具有 EN、ENO 以及 FC 的名称或编号。

参数	数据类型	存储区域	说明
EN	BOOL	I，Q，M，L，D	使能输入
ENO	BOOL	I，Q，M，L，D	使能输出
FB 号	BLOCK_FC	-	FC 的编号；范围与 CPU 有关

说明

CALL_FC（从方块调用功能指令）用于调用一个功能（FC）。如果 EN 为“1”，则执行调用。如果执行 CALL_FC，则

- 保存调用块所需要的返回地址
- 将当前的本地数据区变为以前的本地数据区
- 将 MA 位（MCR 启动位）推至块堆栈中
- 为被调用的功能生成新的本地数据区

在此之后，在被调用的功能中继续执行程序处理。

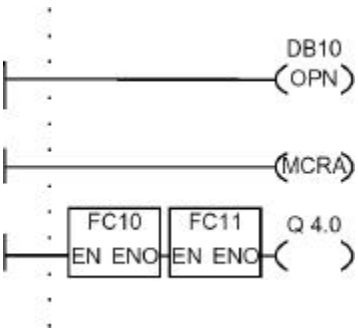
BR 位被扫描，以找到 ENO。用户必须使用 ---(SAVE) 将所需状态（错误评价）赋值给调用块中的 BR 位。如果调用一个功能，并且调用块的变量声明表中有 IN、OUT 和 IN_OUT 声明，则这些变量作为一个形式参数表被添加到用于调用块的程序中。

在调用功能时，必须将实际参数赋值给调用位置的形式参数。任何功能声明中的初始值都没有任何意义。

状态字

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

举例



上图所示梯形逻辑级是由用户编写的一个功能块的程序段。在该功能块中，DB10 **被打开，**MCR **功能启动。如果执行** FC10 **的无条件调用，则执行以下功能：**

存储调用 FB **的返回地址以及用于** DB10 **和调用** FB **的背景数据块的选择数据。在** MCRA **指令中将** MA **位置为“**1**”，并将该位推入块堆栈中，然后为调用的块（**FC10**）将** MA **位复位为“**0**”。程序处理继续在** FC10 **中进行。如果** FC10 **需要** MCR **功能，必须在** FC10 **中重新启动它。**RLO **的状态必须使用指令** ---(SAVE) **保存在** BR **位中，以便能够在调用** FB **中评价错误。当** FC10 **完成后，程序处理返回调用** FB**。**MA **位被恢复。在** FC10 **执行之后，根据** ENO**，在被调用的** FB **中继续执行程序处理：**

ENO = “1” **执行** FC11

ENO = “0” **从下一程序段开始执行**

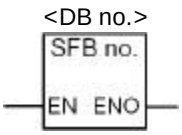
如果 FC11 **也能正确处理，则** ENO = “1”**，并因此，**Q4.0 = “1”**。**

注意

在返回调用块之后，以前打开的 DB **将不再总是打开。请仔细阅读** README **文件中的注意事项。**

10.5 CALL_SFB **从方块调用** SFB

符号



符号取决于 SFB **的使用（是否有参数，以及有多少参数）。它必须具有** EN、ENO **以及** SFB **的名称或编号。**

参数	数据类型	存储区域	说明
EN	BOOL	I，Q，M，L，D	使能输入
ENO	BOOL	I，Q，M，L，D	使能输出
SFB 号 DB 号	BLOCK_SFB BLOCK_DB	- -	SFB 的编号；范围与 CPU 有关

说明

CALL_SFB（从方块调用系统功能块指令）在 EN 为“1”时执行。如果执行 CALL_SFB，则

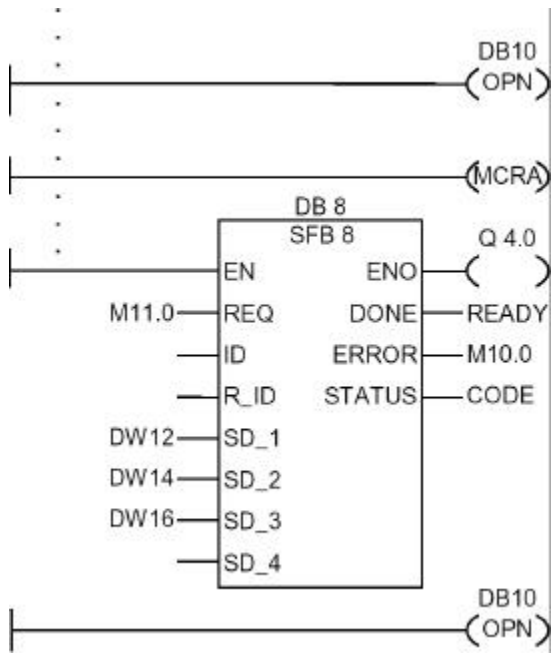
- 保存调用块所需要的返回地址
- 保存用于两个当前数据块（DB 和背景数据块）的选择数据
- 将当前的本地数据区变为以前的本地数据区
- 将 MA 位（MCR 启动位）推至块堆栈中
- 为被调用的系统功能块生成新的本地数据区之后，在被调用的系统功能块中继续执行程序处理。

如果调用了 SFB（EN =“1”），并且没有出现错误，则 ENO 为“1”。

状态字

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

举例



上图所示梯形逻辑级是由用户编写的一个功能块的程序段。在该功能块中，DB10 被打开，MCR 功能启动。如果执行 SFB8 的无条件调用，则执行以下功能：

存储调用 FB 的返回地址以及用于 DB10 和调用 FB 的背景数据块的选择数据。在 MCRA 指令中将 MA 位置为“1”，并将该位推入块堆栈中，然后为调用的块（SFB8）将 MA 位复位为“0”。程序处理继续在 SFB8 中进行。当 SFB8 完成后，程序处理返回调用 FB。MA 位被恢复，用户编写 FB 的背景数据块又成为当前的背景数据块。如果 SFB8 能正确处理，则 ENO =“1”，并因此，Q4.0 =“1”。

注意

当打开一个 FB 或 SFB 时，以前打开的 DB 编号将丢失。所需 DB 必须重新打开。

10.6 CALL_SFC 从方块调用 SFC

符号

符号取决于 SFC 的使用（是否有参数，以及有多少参数）。它必须具有 EN、ENO 以及 SFC 的名称或编号。

参数	数据类型	存储区域	说明
EN	BOOL	I，Q，M，L，D	使能输入
ENO	BOOL	I，Q，M，L，D	使能输出
SFB 号	BLOCK_SFC	-	SFC 的编号；范围与 CPU 有关

说明

CALL_SFC（从方块调用系统功能指令）用于调用一个系统功能（SFC）。如果 EN 为“1”，则执行调用。如果执行 CALL_SFC，则

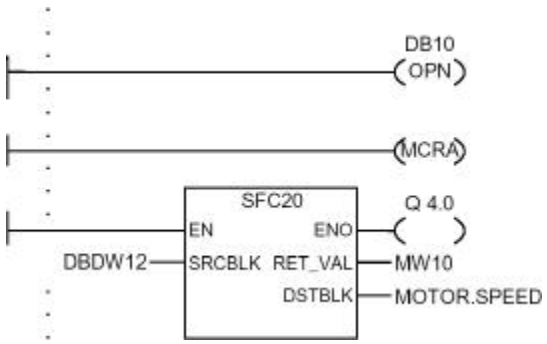
- 保存调用块所需要的返回地址
- 将当前的本地数据区变为以前的本地数据区
- 将 MA 位（MCR 启动位）推至块堆栈中
- 为被调用的系统功能生成新的本地数据区

在此之后，在被调用的 SFC 中继续执行程序处理。如果调用了 SFC（EN =“1”），并且没有出现错误，则 ENO 为“1”。

状态字

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

举例



上图所示梯形逻辑级是由用户编写的一个功能块的程序段。在该功能块中，DB10 **被打开，**MCR **功能启动。如果执行** SFC20 **的无条件调用，则执行以下功能：**

存储调用 FB **的返回地址以及用于** DB10 **和调用** FB **的背景数据块的选择数据。在** MCRA **指令中将** MA **位置为“** 1 **”，并将该位推入块堆栈中，然后为调用的块（**SFC20**）将** MA **位复位为“** 0 **”。程序处理继续在** SFC20 **中进行。当** SFC20 **完成后，程序处理返回调用** FB**。**MA **位被恢复。**

在 SFC20 **执行之后，根据** ENO**，在被调用的** FB **中继续执行程序处理：**
ENO = “ 1 ” Q4.0 = “ 1 ”
ENO = “ 0 ” Q4.0 = “ 0 ”

注意

在返回调用块之后，以前打开的 DB **将不再总是打开。请仔细阅读** README **文件中的注意事项。**

10.7 调用多背景块

符号

参数	数据类型	存储区域	说明
EN	BOOL	I，Q，M，L，D	使能输入
ENO	BOOL	I，Q，M，L，D	使能输出
#变量名	FB，SFB	-	多背景块名称

说明

通过使用一个功能块的数据类型声明一个静态变量，可以生成一个多背景块。在程序元素目录中只包含已声明的多背景块。根据是否有参数以及有多少参数，多背景块的符号会有不同。EN、ENO **和变量名总是存在。**

状态字

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

10.8 从库中调用块

SIMATIC 管理器中的库变量可以用于选择以下程序块：

- 集成在你的 CPU 操作系统中的块（对于版本 3 的 STEP 7 项目，为“ Standard Library（标准库）”；对于版本 2 的 STEP 7 项目，为“ stdlibs (V2)”。
- 由于想多次使用你自己保存在库中的块。

10.9 使用 MCR 功能的重要注意事项

在使用主控继电器以 MCRA 方式启动的块时应注意：

- 如果 MCR 停止，则 ---(MCR<) 和 ---(MCR>) 之间的程序段的所有赋值写入“ 0 ”值。这适用于包含赋值的所有方块，包括块参数传送。
- 在 MCR< 指令之前，如果 RLO = “ 0 ”，则 MCR 停止。

危险：PLC 处于 STOP 状态，或未定义运行时间特性！

在 VAR_TEMP 中定义临时变量之后，编译器也可以使用写指令访问本地数据，以便计算地址。这就意味着以下指令顺序会将 PLC 置为 STOP 状态，或造成未定义的运行时间特性：

形式参数存取

- 存取类型为 STRUCT、UDT、ARRAY、STRING 的复杂的 FC 参数部分
- 从版本 2 块的 IN_OUT 区域中存取类型为 STRUCT、UDT、ARRAY、STRING 的复杂的 FB 参数部分
- 如果版本 2 功能块的地址大于 8180.0，则存取它的参数。
- 在版本 2 功能块中存取类型为 BLOCK_DB 的参数打开 DB0。任何后续的数据存取将 CPU 置为 STOP 状态。T 0、C 0、FC0 或 FB0 经常用于 TIMER、COUNTER、BLOCK_FC 和 BLOCK_FB。

参数传送

- 参数在调用中传送。

LAD/FBD

- 在梯形逻辑或 FBD 中 T 分支和中间输出从 RLO = 0 开始。

补救

从它们对 MCR 的相关性，来解决以上问题：

1. 在语句或程序段出现问题之前，利用 Master Control Relay Deactivate（停止主控继电器）指令停止主控继电器。
2. 在语句或程序段出现问题之后，利用 Master Control Relay Activate（启动主控继电器）指令启动主控继电器。

10.10 ---(MCR<) 主控继电器接通

使用 MCR 功能的重要注意事项

符号

---(MCR<)

说明

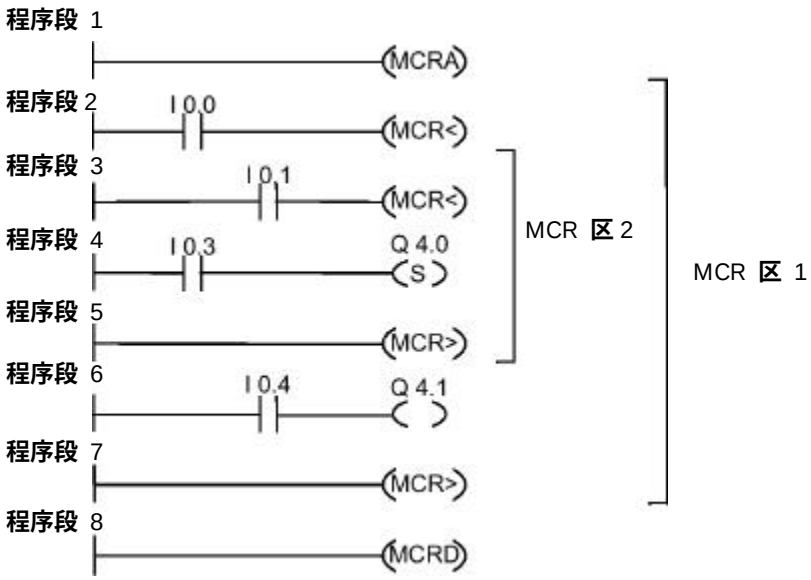
---(MCR<)（**主控继电器区打开指令**）**用于将** RLO **保存在** MCR **堆栈中。**MCR **嵌套堆栈是一个** LIFO **堆栈（后进先出），只能有** 8 **个堆栈输入（嵌套深度）。如果堆栈已满,** ---(MCR<) **功能将产生一个** MCR **堆栈错误（**MCRF**）。以下元素与** MCR **有关，并在打开一个** MCR **区时受保存在** MCR **堆栈中的** RLO **状态的影响：**

- --(#) **中间输出**
- --( ) **输出**
- --(S) **输出置位**
- --(R) **输出复位**
- RS **复位触发器**
- SR **置位触发器**
- MOVE **赋值**

状态字

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

举例



MCR 功能由 MCRA 梯形逻辑级启动。然后，它可以最多生成 8 个嵌套的 MCR 区。在该举例中，有两个 MCR 区。如下执行功能：

I0.0 = “1”（对于区 1，MCR 接通）：I0.4 的逻辑状态被赋值给 Q4.1

I0.0 = “0”（对于区 1，MCR 断开）：Q4.1 为“0”，与 I0.4 的逻辑状态无关

I0.1 = “1”（对于区 2，MCR 接通）：如果 I0.3 为“1”，则 Q4.0 置为“1”。

I0.1 = “0”（对于区 2，MCR 断开）：Q4.0 保持不变，与 I0.3 的逻辑状态无关

10.11 ---(MCR>)主控继电器断开

使用 MCR 功能的重要注意事项

符号

---(MCR>)

说明

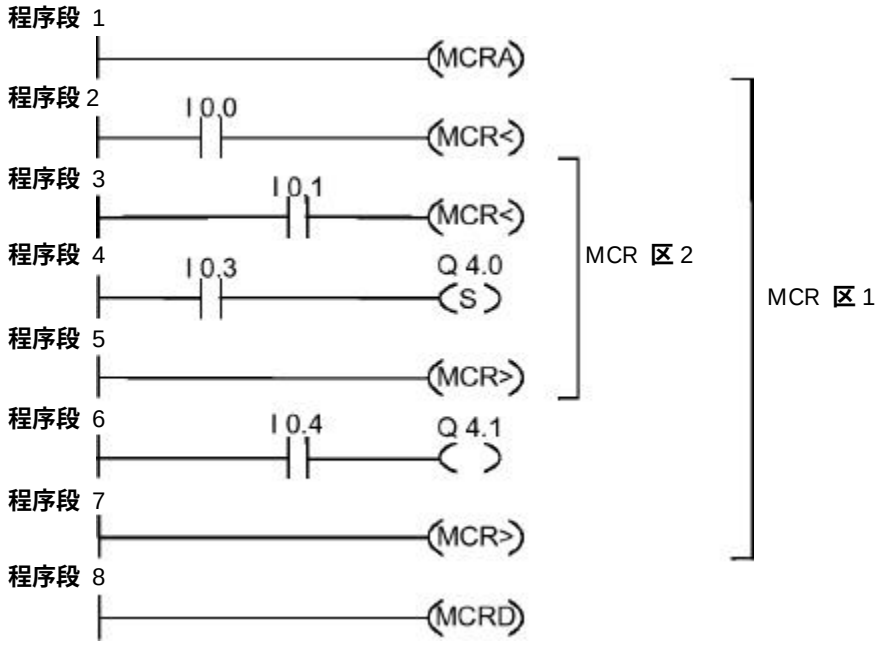
---(MCR>)（主控继电器区关闭指令）用于将 RLO 输入从 MCR 堆栈中删除。MCR 嵌套堆栈是一个 LIFO 堆栈（后进先出），只能有 8 个堆栈输入（嵌套深度）。如果堆栈已空，---(MCR>) 功能将产生一个 MCR 堆栈错误（MCRF）。以下元素与 MCR 有关，并在打开一个 MCR 区时受保存在 MCR 堆栈中的 RLO 状态的影响：

- --(#) 中间输出
- --( ) 输出
- --(S) 输出置位
- --(R) 输出复位
- RS 复位触发器
- SR 置位触发器
- MOVE 赋值

状态字

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

举例



MCR 功能由 ---(MCRA) 梯形逻辑级启动。然后，它可以最多生成 8 个嵌套的 MCR 区。在该举例中，有两个 MCR 区。第一个 ---(MCR>)（MCR 断开）梯形逻辑级属于第二个 ---(MCR<)（MCR 接通）梯形逻辑级。其之间的所有梯形逻辑级都属于 MCR 区 2。如下执行功能：

I0.0 = “ 1 ”：I0.4 的逻辑状态被赋值给 Q4.1

I0.0 = “ 0 ”：Q4.1 为 “ 0 ”，与 I0.4 的逻辑状态无关

I0.1 = “ 1 ”：如果 I0.3 为 “ 1 ”，则 Q4.0 置为 “ 1 ”。

I0.1 = “ 0 ”：Q4.0 保持不变，与 I0.3 的逻辑状态无关

10.12 ---(MCRA) 主控继电器启动

使用 MCR 功能的重要注意事项

符号

---(MCRA)

说明

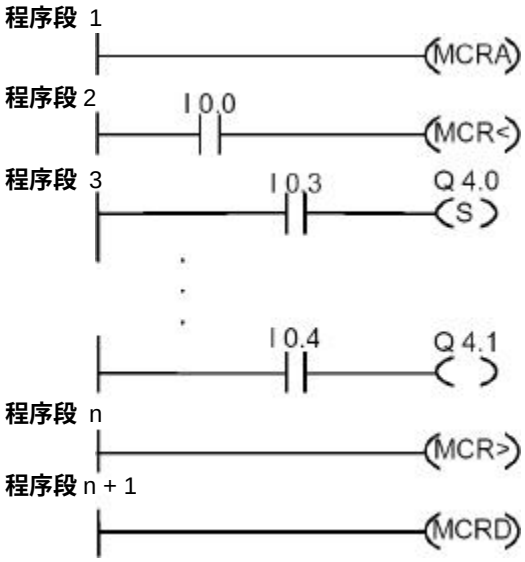
---(MCRA)（启动主控继电器指令）用于启动主控继电器的功能。在该指令之后，它可以使用指令编程 MCR 区：

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

状态字

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

举例



MCR **功能由** MCRA **梯形逻辑级启动。** MCR< **和** MCR>**（输出** Q4.0，Q4.1**）之间的梯形逻辑级如下执行：**

I0.0 =“ 1 ”（MCR **接通）：如果** I0.3 **为逻辑“** 1 **”，则** Q4.0 **置为“** 1 **”，或如果** I0.3 **为“** 0 **”，则** Q4.0 **保持不变，并且** I0.4 **的逻辑状态被赋值给** Q4.1。

I0.0 =“ 0 ”（MCR **断开）：**Q4.0 **保持不变，与** I0.3 **的逻辑状态无关，并且** Q4.1 **为“** 0 **”，与** I0.4 **的逻辑状态无关。**

在下一梯形逻辑级，指令 ---(MCRD) **将停止** MCR。**这就意味着你不能再使用指令对** ---(MCR<) **和** ---(MCR>) **编程** MCR **区。**

10.13 ---(MCRD) 主控继电器停止

使用 MCR **功能的重要注意事项**

符号

---(MCRD)

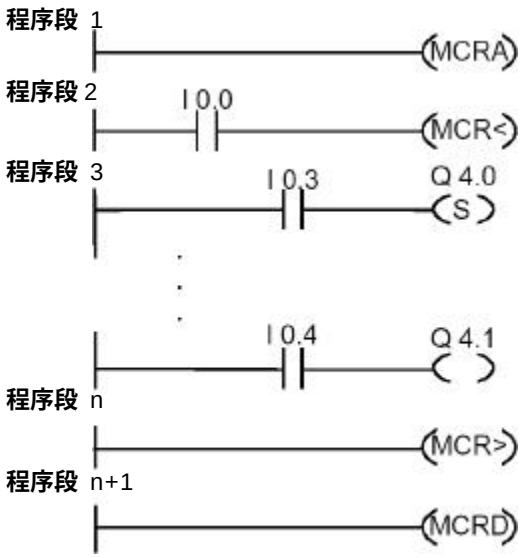
说明

---(MCRD)**（停止主控继电器指令）可以停止** MCR **功能。在该指令之后，你不能编程** MCR **区。**

状态字

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

举例



MCR **功能由** MCRA **梯形逻辑级启动。** MCR< **和** MCR>**（输出** Q4.0，Q4.1**）之间的梯形逻辑级如下执行：**

I0.0 =“ 1 ”（MCR **接通）：如果** I0.3 **为逻辑“** 1 **”，则** Q4.0 **置为“** 1 **”，并且** I0.4 **的逻辑状态被赋值给** Q4.1。

I0.0 =“ 0 ”（MCR **断开）：**Q4.0 **保持不变，与** I0.3 **的逻辑状态无关，并且** Q4.1 **为“** 0 **”，与** I0.4 **的逻辑状态无关。**

在下一梯形逻辑级，指令 ---(MCRD) **将停止** MCR。**这就意味着你不能再使用指令对** ---(MCR<) **和** ---(MCR>) **编程** MCR **区。**

10.14 ---(RET) 返回

符号

---(RET)

说明

RET**（返回指令）用于有条件地放弃一个块。对于该输出，需要前一逻辑操作。**

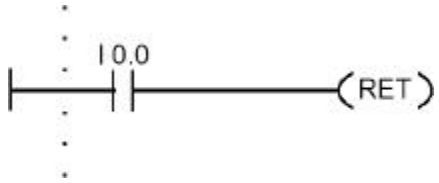
状态字

条件返回（若 RLO = “ 1 ”**，则返回）：**

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

* **操作** RET **内部显示在顺序“** SAVE **；** BEC**，”中。这也会影响** BR **位。**

举例



如果 I0.0 = **“** 1 **”，则块被放弃。**

11 移位和循环指令

11.1 移位指令

11.1.1 移位指令概述

说明

使用移位指令，可以将输入 IN 中的内容向左或向右逐位移动（请参见“ CPU 寄存器 ”）。将输入 IN 中的内容左移相当于完成乘 2 加权；将输入 IN 中的内容右移相当于完成除 2 加权的运算。例如，如果将十进制数值“ 3 ”的等效二进制数左移 3 位，则累加器中的结果是十进制数“ 24 ”的二进制数。如果将十进制数值“ 16 ”的等效二进制数右移 2 位，则累加器中的结果是十进制数“ 4 ”的二进制数。

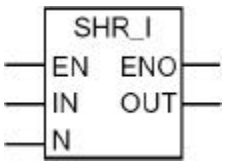
输入参数 N 提供的数值表示移动的位数。执行移位指令所空出的位既可以用零填入，也可以用符号位的信号状态填入（“ 0 ”代表“ 正 ”，“ 1 ”代表“ 负 ”）。最后移出位的信号状态装入状态字的 CC 1 位。状态字的 CC 0 和 OV 位清零。可用跳转指令判断 CC 1 位的状态。

下述移位指令可供使用：

- SHR_I **整数右移**
- SHR_DI **双整数右移**
- 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

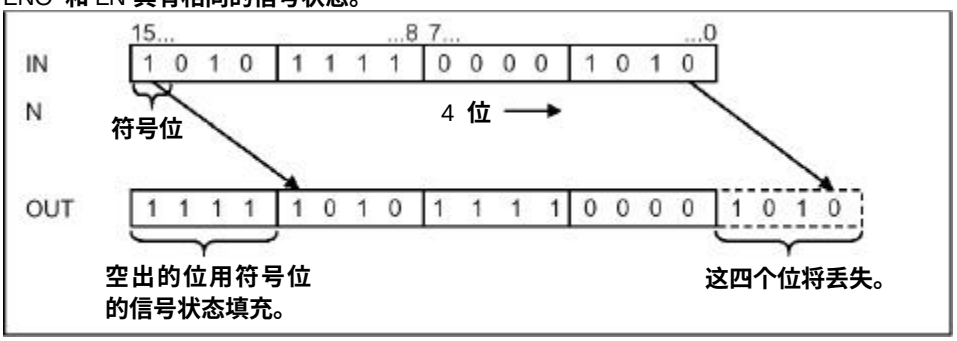


参数	数据类型	存储区域	说明
EN	BOOL	I，Q，M，L，D	**使能输入**
ENO	BOOL	I，Q，M，L，D	**使能输出**
IN	INT	I，Q，M，L，D	**要移位的值**
N	WORD	I，Q，M，L，D	**要移位的位数**
OUT	INT	I，Q，M，L，D	**移位操作的结果**

说明

SHR_I（整数右移指令）可以由使能（EN）输入端的逻辑“1”信号激活。SHR_I 指令用于将输入 IN 位的位 0 到位 15 逐位右移。位 16 到位 31 不受影响。输入 N 指定移位的位数。如果 N 大于 16，则该命令的作用和 N 等于 16 时一样。从左边到需填充空出位的所有移位都根据位 15 的信号状态填充（这是一个整数的符号位）。这就意味着，如果整数为正值，则这些位被赋值“0”；如果整数为负值，则这些位被赋值“1”。移位操作的结果可以在输出 OUT 中扫描。如果 N 不等于“0”，则通过 SHR_I 指令将 CC 0位和 OV 位清零。

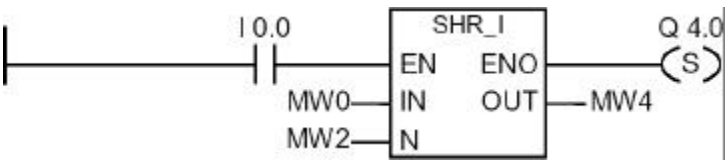
ENO 和 EN 具有相同的信号状态。



状态字

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

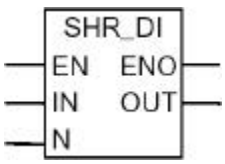
举例



如果 I0.0 = “1”，则 SHR_I 方块激活。MW0 装入，并右移使用 MW2 指定的位数。其结果被写入 MW4 中。Q4.0 置位。

11.1.3 SHR_DI 双整数右移

符号



参数	数据类型	存储区域	说明
EN	BOOL	I，Q，M，L，D	使能输入
ENO	BOOL	I，Q，M，L，D	使能输出
IN	DINT	I，Q，M，L，D	要移位的值
N	WORD	I，Q，M，L，D	要移位的位数
OUT	DINT	I，Q，M，L，D	移位操作的结果

说明

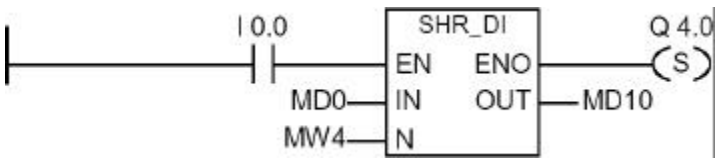
SHR_DI（双整数右移指令）可以由使能（EN）输入端的逻辑“1”信号激活。SHR_DI 指令用于将输入 IN 位的位 0 到位 31 逐位右移。输入 N 指定移位的位数。如果 N 大于 32，则该命令的作用和 N 等于 32 时一样。从左边到需填充空出位的所有移位都根据位 31 的信号状态填充（这是一个整数的符号位）。这就意味着，如果整数为正值，则这些位被赋值“0”；如果整数为负值，则这些位被赋值“1”。移位操作的结果可以在输出 OUT 中扫描。如果 N 不等于“0”，则通过 SHR_DI 指令将 CC 0 位和 OV 位清零。

ENO 和 EN 具有相同的信号状态。

状态字

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

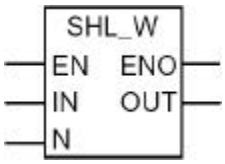
举例



如果 I0.0 为逻辑“1”，则 SHR_DI 方块激活。MD0 装入，并右移使用 MW4 指定的位数。其结果被写入 MD10 中。Q4.0 置位。

11.1.4 SHL_W 字左移

符号

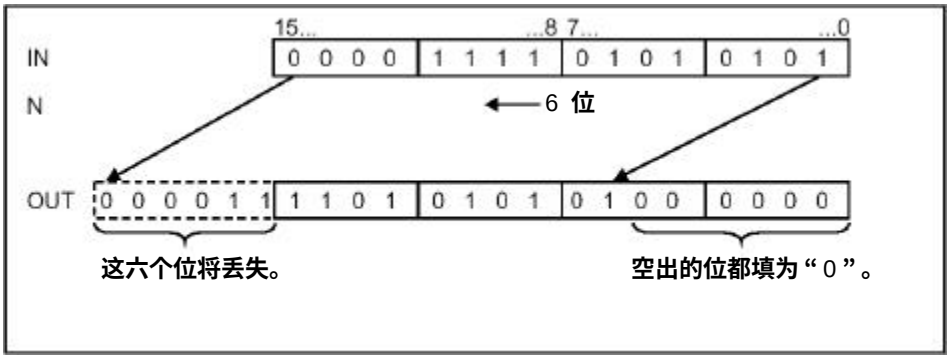


参数	数据类型	存储区域	说明
EN	BOOL	I，Q，M，L，D	使能输入
ENO	BOOL	I，Q，M，L，D	使能输出
IN	DINT	I，Q，M，L，D	要移位的值
N	WORD	I，Q，M，L，D	要移位的位数
OUT	DINT	I，Q，M，L，D	移位操作的结果

说明

SHL_W（字左移指令）可以由使能（EN）输入端的逻辑“ 1 ”信号激活。SHL_W 指令用于将输入 IN 位的位 0 到位 15 逐位左移。位 16 到位 31 不受影响。输入 N 指定移位的位数。如果 N 大于 16，该命令将“ 0 ”写入输出 OUT，并将状态字中的位 CC 0 和 OV 清零。从右边到需填充空出位的所有位将填入 N 个零。移位操作的结果可以在输出 OUT 中扫描。如果 N 不等于“ 0 ”，则通过 SHL_W 指令将 CC 0 位和 OV 位清零。

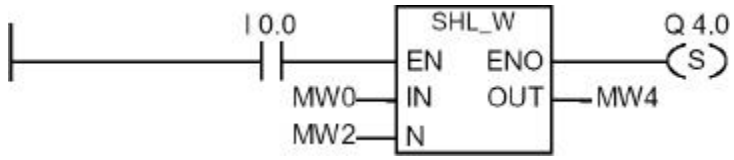
ENO 和 EN 具有相同的信号状态。



状态字

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

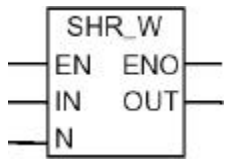
举例



如果 I0.0 为逻辑“ 1 ”，则 SHL_W 方块激活。MW0 装入，并左移使用 MW2 指定的位数。其结果被写入 MW4 中。Q4.0 置位。

11.1.5 SHR_W 字右移

符号



参数	数据类型	存储区域	说明
EN	BOOL	I，Q，M，L，D	使能输入
ENO	BOOL	I，Q，M，L，D	使能输出
IN	WORD	I，Q，M，L，D	要移位的值
N	WORD	I，Q，M，L，D	要移位的位数
OUT	WORD	I，Q，M，L，D	移位操作的结果

说明

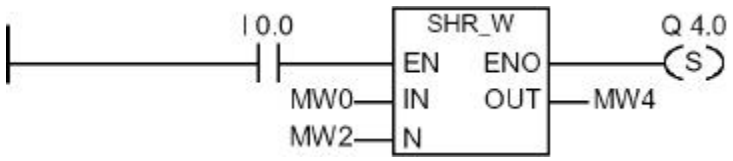
SHR_W（字右移指令）可以由使能（EN）输入端的逻辑“1”信号激活。SHR_W 指令用于将输入 IN 位的位 0 到位 15 逐位右移。位 16 到位 31 不受影响。输入 N 指定移位的位数。如果 N 大于 16，该命令将“0”写入输出 OUT，并将状态字中的位 CC 0 和 OV 清零。从左边到需填充空出位的所有位将填入 N 个零。移位操作的结果可以在输出 OUT 中扫描。如果 N 不等于“0”，则通过 SHR_W 指令将 CC 0 位和 OV 位清零。

ENO 和 EN 具有相同的信号状态。

状态字

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

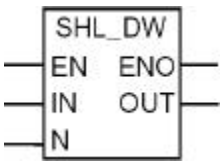
举例



如果 I0.0 为逻辑“1”，则 SHR_W 方块激活。MW0 装入，并右移使用 MW2 指定的位数。其结果被写入 MW4 中。Q4.0 置位。

11.1.6 SHL_DW 双字左移

符号



参数	数据类型	存储区域	说明
EN	BOOL	I，Q，M，L，D	使能输入
ENO	BOOL	I，Q，M，L，D	使能输出
IN	DWORD	I，Q，M，L，D	要移位的值
N	WORD	I，Q，M，L，D	要移位的位数
OUT	DWORD	I，Q，M，L，D	双字移位操作的结果

说明

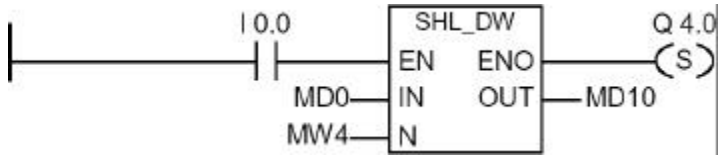
SHL_DW（双字左移指令）可以由使能（EN）输入端的逻辑“1”信号激活。SHL_DW 指令用于将输入 IN 位的位 0 到位 31 逐位左移。输入 N 指定移位的位数。如果 N 大于 32，该命令将“0”写入输出 OUT，并将状态字中的位 CC 0 和 OV 清零。从右边到需填充空出位的所有位将填入 N 个零。双字移位操作的结果可以在输出 OUT 中扫描。如果 N 不等于“0”，则通过 SHL_DW 指令将 CC 0 位和 OV 位清零。

ENO 和 EN 具有相同的信号状态。

状态字

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

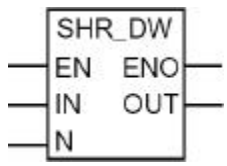
举例



如果 I0.0 为逻辑“1”，则 SHL_DW 方块激活。MD0 装入，并左移使用 MW4 指定的位数。其结果被写入 MD10 中。Q4.0 置位。

11.1.7 SHR_DW 双字右移

符号

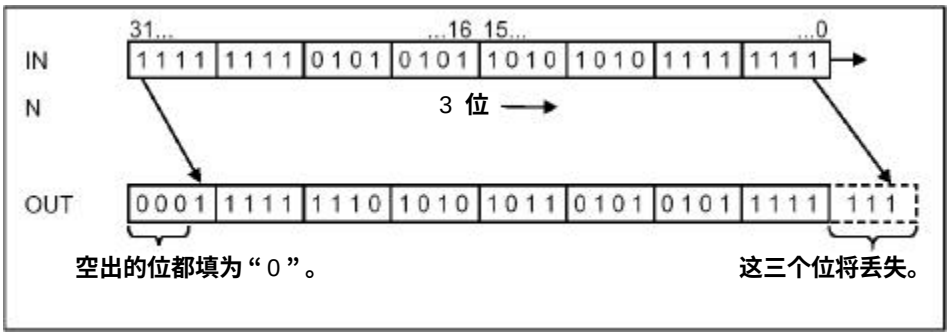


参数	数据类型	存储区域	说明
EN	BOOL	I，Q，M，L，D	使能输入
ENO	BOOL	I，Q，M，L，D	使能输出
IN	DWORD	I，Q，M，L，D	要移位的值
N	WORD	I，Q，M，L，D	要移位的位数
OUT	DWORD	I，Q，M，L，D	双字移位操作的结果

说明

SHR_DW（双字右移指令）可以由使能（EN）输入端的逻辑“1”信号激活。SHR_DW 指令用于将输入 IN 位的位 0 到位 31 逐位右移。输入 N 指定移位的位数。如果 N 大于 32，该命令将“0”写入输出 OUT，并将状态字中的位 CC 0 和 OV 清零。从左边到需填充空出位的所有位将填入 N 个零。双字移位操作的结果可以在输出 OUT 中扫描。如果 N 不等于“0”，则通过 SHR_DW 指令将 CC 0 位和 OV 位清零。

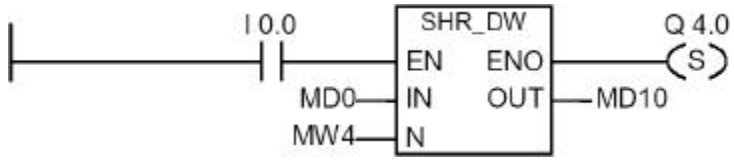
ENO 和 EN 具有相同的信号状态。



状态字

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

举例



如果 I0.0 为逻辑“1”，则 SHR_DW 方块激活。MD0 装入，并右移使用 MW4 指定的位数。其结果被写入 MD10 中。Q4.0 置位。

11.2 循环指令

11.2.1 循环指令概述

说明

使用循环指令，可以将输入 IN 中的全部内容循环地逐位左移或右移。空出的位用输入 IN 移出位的信号状态填充。

输入参数 N 提供的数值表示循环的位数。

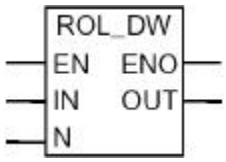
根据指令，通过状态字的 CC 1 位执行循环。状态字的 CC 0 位复位为“0”。

下述循环指令可供使用：

- ROL_DW **双字左循环**
- ROR_DW **双字右循环**

11.2.2 ROL_DW 双字左循环

符号

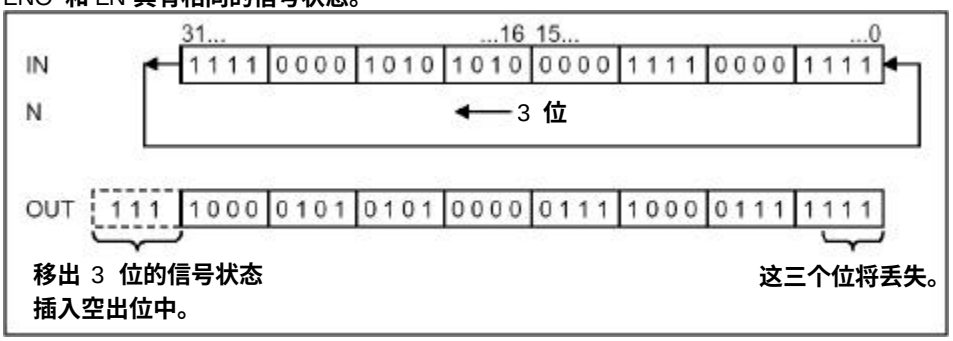


参数	数据类型	存储区域	说明
EN	BOOL	I，Q，M，L，D	使能输入
ENO	BOOL	I，Q，M，L，D	使能输出
IN	DWORD	I，Q，M，L，D	要循环的值
N	WORD	I，Q，M，L，D	要循环的位数
OUT	DWORD	I，Q，M，L，D	双字循环操作的结果

说明

ROL_DW（双字左循环指令）可以由使能（EN）输入端的逻辑“1”信号激活。ROL_DW 指令用于将输入 IN 位的全部内容逐位循环左移。输入 N 指定循环的位数。如果 N 大于 32，则双字 IN 循环 ((N-1) 乘 32)+1 位。右边的位以循环位状态填充。双字循环操作的结果可以在输出 OUT 中扫描。如果 N 不等于“0”，则通过 ROL_DW 指令将 CC 0 位和 OV 位清零。

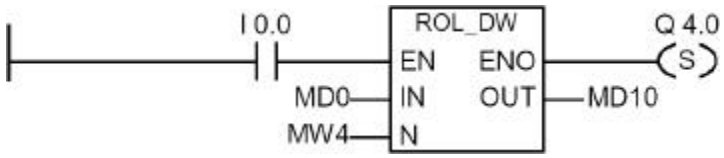
ENO 和 EN 具有相同的信号状态。



状态字

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

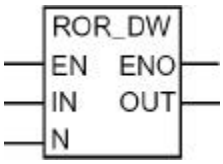
举例



如果 I0.0 为逻辑“1”，则 ROL_DW 方块激活。MD0 装入，并左循环使用 MW4 指定的位数。其结果被写入 MD10 中。Q4.0 置位。

11.2.3 ROR_DW 双字右循环

符号



参数	数据类型	存储区域	说明
EN	BOOL	I，Q，M，L，D	使能输入
ENO	BOOL	I，Q，M，L，D	使能输出
IN	DWORD	I，Q，M，L，D	要循环的值
N	WORD	I，Q，M，L，D	要循环的位数
OUT	DWORD	I，Q，M，L，D	双字循环操作的结果

说明

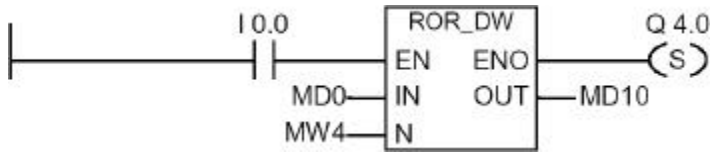
ROR_DW（双字右循环指令）可以由使能（EN）输入端的逻辑“1”信号激活。ROR_DW 指令用于将输入 IN 位的全部内容逐位循环右移。输入 N 指定循环的位数。如果 N 大于 32，则双字 IN 循环 ((N-1) 乘 32)+1 位。左边的位以循环位状态填充。双字循环操作的结果可以在输出 OUT 中扫描。如果 N 不等于“0”，则通过 ROR_DW 指令将 CC 0 位和 OV 位清零。

ENO 和 EN 具有相同的信号状态。

状态字

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

举例



如果 I0.0 为逻辑“1”，则 ROR_DW 方块激活。MD0 装入，并右循环使用 MW4 指定的位数。其结果被写入 MD10 中。Q4.0 被置位。

12 状态位指令

12.1 状态位指令概述

说明

状态位指令是位逻辑指令，针对状态字的各位进行操作。这些指令的每一条对由一个或多个状态字位表明的下列条件之一起作用。

- **二进制结果位** (BR ---| |--) **置位（即有一个信号状态为“** 1 **”）。**
- **算术运算功能有上溢** (OV ---| |--) **或存储的上溢** (OS ---| |---)。
- **算术运算功能的结果是无序的** (UO ---| |--)。
- **算术运算功能的结果以下列方式之一与“** 0 **”相关：** == 0，<> 0，> 0，< 0，>= 0，<= 0。

当状态位指令以串联方式连接时，根据“与”真值表，指令将其信号状态的检验结果与前面的逻辑运算结果相结合。当状态位指令以并联方式连接时，根据“或”真值表，指令将其信号状态的检验结果与前面的逻辑运算结果（RLO**）相结合。**

状态字

状态字是 CPU **中存储区中的一个寄存器，包含有为位地址和字逻辑指令提供参考的位。状态字的结构：**

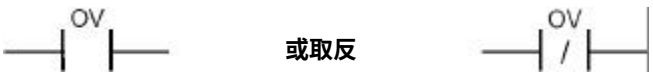
2^{15}...	...2^9	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
		BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC

你可以判断状态字中的各位：

- **通过整数算术运算功能**
- **通过浮点功能。**

12.2 OV ---| |--- 溢出异常位

符号



说明

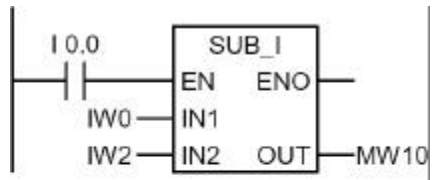
OV ---| |---**（溢出异常位指令）或** OV ---| / |--- **（取反溢出异常位指令）接点符号用于识别上一次算术运算功能中的溢出。这也就是说，在功能执行完后，操作的结果在允许的负范围或正范围之外。当在串联中使用时，扫描的结果通过与（**AND**）逻辑运算与** RLO **相结合；当在并联中使用时，扫描的结果通过或（**OR**）逻辑运算与** RLO **相结合。**

状态字

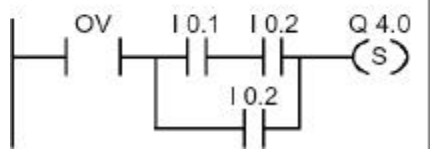
	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
写	-	-	-	-	-	x	x	x	1

举例

程序段 1



程序段 2



方块通过 I0.0 **的信号状态“** 1 **”启动。如果算术运算功能“** IW0- IW2 **”的结果在整数的允许范围之外，则** OV **位被置位。**OV **的信号状态为“** 1 **”。如果** OV **的扫描结果为信号状态“** 1 **”，并且程序段** 2 **的** RLO **位“** 1 **”，则** Q4.0 **被置位。**

注意

因为两个程序段是独立的，必须进行 OV **的扫描。否则，可以采用算术运算功能的** ENO **输出，当运算结果超出允许的范围时，**ENO **为“** 0 **”。**

12.3 OS ---| |--- 存储溢出异常位

符号

说明

OS ---| |---**（存储溢出异常位指令）或** OS ---| / |---**（存储取反溢出异常位指令）接点符号用于识别和存储上一次算术运算功能中的锁存溢出。如果运算结果在允许的负范围或正范围之外，则状态字的** OS **位置位。不象** OV **位，对于后续算术运算功能需重新写入，**OS **位可保存发生时的溢出。**OS **位可保持置位，直到离开块。**

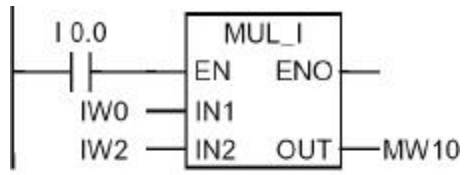
当在串联中使用时，扫描的结果通过与（AND**）逻辑运算与** RLO **相结合；当在并联中使用时，扫描的结果通过或（**OR**）逻辑运算与** RLO **相结合。**

状态字

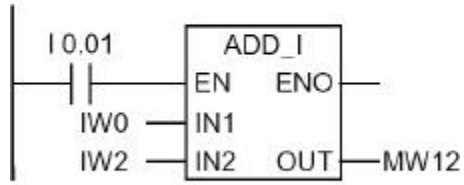
	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
写	-	-	-	-	-	x	x	x	1

举例

程序段1



程序段 2



程序段 2

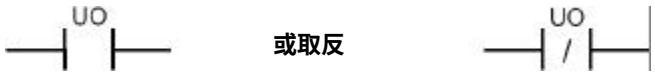
MUL_I **方块通过** I0.0 **的信号状态“** 1 **”启动。**ADD_I **方块通过** I0.1 **的逻辑“** 1 **”启动。如果算术运算功能的结果在整数的允许范围之外，则状态字的** OS **位被置为“** 1 **”。如果** OS **的扫描结果为逻辑“** 1 **”，则** Q4.0 **置位。**

注意

因为两个程序段是独立的，必须进行 OS **的扫描。否则，可以采用第一次算术运算功能的** ENO **输出，并将与第二次算术运算功能的** EN **输入连接（级联布置）。**

12.4 UO ---| |--- 无序异常位

符号



说明

UO ---| |--**（无序异常位指令）或** UO ---| / |---**（取反无序异常位指令）接点符号用于识别浮点算术运算功能的结果是否无序（即，算术运算功能中的数值中是否有无效的浮点数）。**

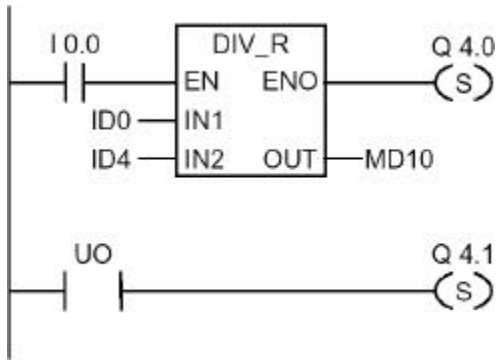
如果浮点算术运算功能的结果（UO）无效，则信号状态扫描的结果为“1”。如果 CC 1 和 CC 0 中的逻辑运算显示“非无效”，则信号状态扫描的结果为“0”。

当在串联中使用时，扫描的结果通过与（AND）逻辑运算与 RLO 相结合；当在并联中使用时，扫描的结果通过或（OR）逻辑运算与 RLO 相结合。

状态字

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

举例

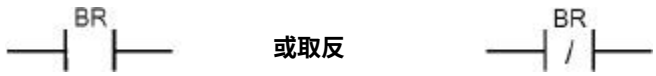


方块通过 I0.0 的信号状态“1”启动。如果 ID0 或 ID4 的值为无效浮点数，则算术运算功能无效。如果 EN 的信号状态为“1”（启动），并且如果在功能 DIV_R 执行过程中出现错误，则 ENO 的信号状态为“0”。

如果执行功能 DIV_R，但有一个值是无效的浮点数，则输出 Q4.1 置位。

12.5 BR ---| |--- 异常位二进制结果

符号



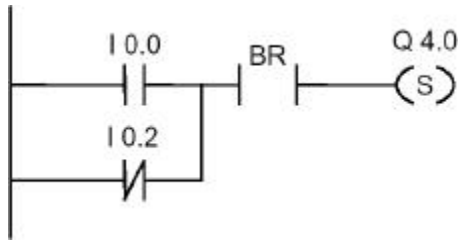
说明

BR ---| |---（异常位 BR 存储器指令）或 BR ---| / |---（取反异常位 BR 存储器指令）接点符号用于检查状态字中的 BR 位的逻辑状态。当在串联中使用时，扫描的结果通过与（AND）逻辑运算与 RLO 相结合；当在并联中使用时，扫描的结果通过或（OR）逻辑运算与 RLO 相结合。BR 位用于从字到位的处理传送过程中。

状态字

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

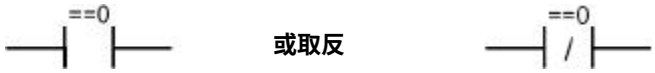
举例



如果 I0.0 **为“** 1 **”或** I0.2 **为“** 0 **”，则** Q4.0 **置位，除该** RLO **以外，**BR **位的逻辑状态为“** 1 **”。**

12.6 ==0 ---| |--- 结果位等于“ 0 ”

符号



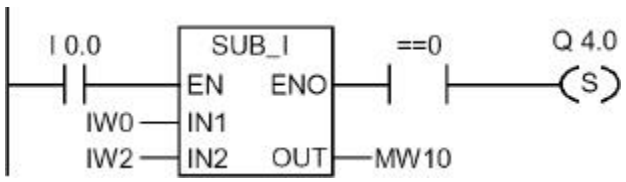
说明

==0 ---| |--**（结果位等于“** 0 **”指令）或** ==0 ---| / |---**（取反结果位等于“** 0 **”指令）接点符号用于识别算术运算功能的结果是否等于“** 0 **”。该指令可以扫描状态字中的条件代码位** CC 1 **和** CC 0**，以决定结果与“** 0 **”的关系。当在串联中使用时，扫描的结果通过与（**AND**）逻辑运算与** RLO **相结合；当在并联中使用时，扫描的结果通过或（**OR**）逻辑运算与** RLO **相结合。**

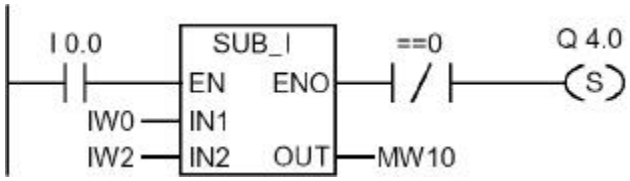
状态字

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

举例



方块通过 I0.0 **的信号状态“** 1 **”启动。如果** IW0 **的值等于** IW2 **的值，则算术运算功能“** IW0 - IW2 **”的结果为“** 0 **”。如果功能正确执行，并且结果等于“** 0 **”，则** Q4.0 **置位。**



如果功能正确执行，并且结果不等于“ 0 ”，则 Q4.0 置位。

12.7 <>0 ---| |--- 结果位不等于“ 0 ”

符号

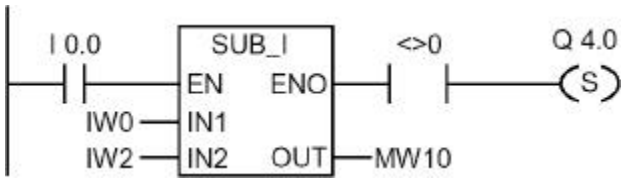
说明

<>0 ---| |--（结果位不等于“ 0 ”指令）或 <>0 ---| / |---（取反结果位不等于“ 0 ”指令）接点符号用于识别算术运算功能的结果是否不等于“ 0 ”。该指令可以扫描状态字中的条件代码位 CC 1 和 CC 0，以决定结果与“ 0 ”的关系。当在串联中使用时，扫描的结果通过与（AND）逻辑运算与 RLO 相结合；当在并联中使用时，扫描的结果通过或（OR）逻辑运算与 RLO 相结合。

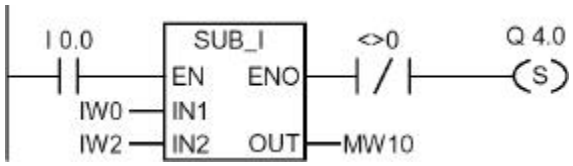
状态字

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

举例



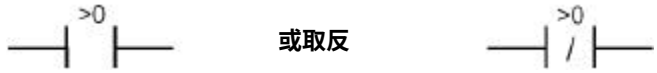
方块通过 I0.0 的信号状态“ 1 ”启动。如果 IW0 的值不等于 IW2 的值，则算术运算功能“ IW0 - IW2 ”的结果不等于“ 0 ”。如果功能正确执行，并且结果不等于“ 0 ”，则 Q4.0 置位。



如果功能正确执行，并且结果等于“ 0 ”，则 Q4.0 置位。

12.8 >0 ---| |--- 结果位大于“0”

符号



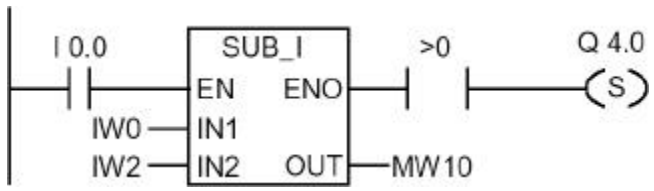
说明

>0 ---| |--（结果位大于“0”指令）或 >0 ---| / |---（取反结果位大于“0”指令）接点符号用于识别算术运算功能的结果是否大于“0”。该指令可以扫描状态字中的条件代码位 CC 1 和 CC 0，以决定结果与“0”的关系。当在串联中使用时，扫描的结果通过与（AND）逻辑运算与 RLO 相结合；当在并联中使用时，扫描的结果通过或（OR）逻辑运算与 RLO 相结合。

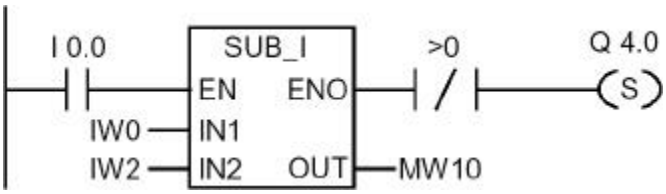
状态字

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

举例



方块通过 I0.0 的信号状态“1”启动。如果 IW0 的值大于 IW2 的值，则算术运算功能“IW0-IW2”的结果大于“0”。如果功能正确执行，并且结果大于“0”，则 Q4.0 置位。



如果功能正确执行，并且结果不大于“0”，则 Q4.0 置位。

12.9 <0 ---| |--- 结果位小于“0”

符号

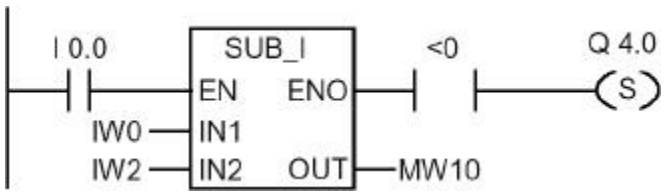
说明

<0 ---| |--- **（结果位小于“0”指令）或** <0 ---| / |--- **（取反结果位小于“0”指令）接点符号用于识别算术运算功能的结果是否小于“0”。该指令可以扫描状态字中的条件代码位** CC 1 **和** CC 0**，以决定结果与“0”的关系。当在串联中使用时，扫描的结果通过与（**AND**）逻辑运算与** RLO **相结合；当在并联中使用时，扫描的结果通过或（**OR**）逻辑运算与** RLO **相结合。**

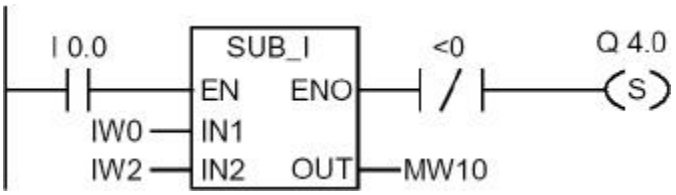
状态字

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

举例



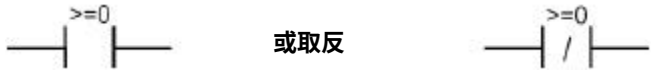
方块通过 I0.0 **的信号状态“**1**”启动。如果** IW0 **的值小于** IW2 **的值，则算术运算功能“**IW0 - IW2**”的结果小于“**0**”。如果功能正确执行，并且结果小于“**0**”，则** Q4.0 **置位。**



如果功能正确执行，并且结果不小于“0**”，则** Q4.0 **置位。**

12.10 >=0 ---| |--- 结果位大于等于“0”

符号



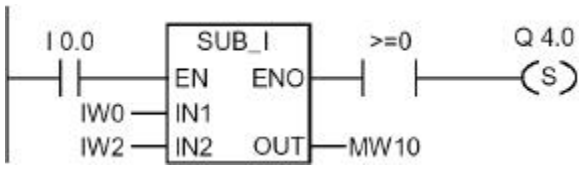
说明

>=0 ---| |--（结果位大于等于“0”指令）或 >=0 ---| / |---（取反结果位大于等于“0”指令）接点符号用于识别算术运算功能的结果是否大于等于“0”。该指令可以扫描状态字中的条件代码位 CC 1 和 CC 0，以决定结果与“0”的关系。当在串联中使用时，扫描的结果通过与（AND）逻辑运算与 RLO 相结合；当在并联中使用时，扫描的结果通过或（OR）逻辑运算与 RLO 相结合。

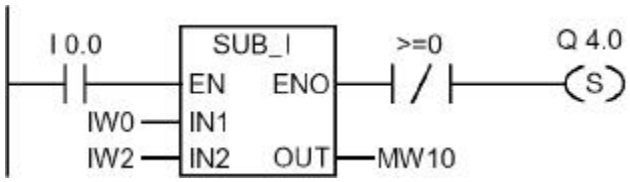
状态字

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

举例



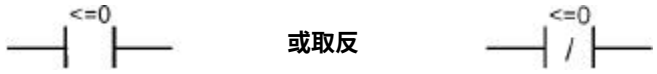
方块通过 I0.0 的信号状态“1”启动。如果 IW0 的值大于或等于 IW2 的值，则算术运算功能“IW0 - IW2”的结果大于或等于“0”。如果功能正确执行，并且结果大于或等于“0”，则 Q4.0 置位。



如果功能正确执行，并且结果不大于或不等于“0”，则 Q4.0 置位。

12.11 <=0 ---| |--- 结果位小于等于“ 0 ”

符号



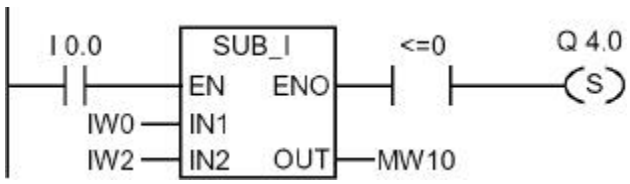
说明

<=0 ---| |--（结果位小于等于“ 0 ”指令）或 <=0 ---| / |---（取反结果位小于等于“ 0 ”指令）接点符号用于识别算术运算功能的结果是否小于等于“ 0 ”。该指令可以扫描状态字中的条件代码位 CC 1 和 CC 0，以决定结果与“ 0 ”的关系。当在串联中使用时，扫描的结果通过与（AND）逻辑运算与 RLO 相结合；当在并联中使用时，扫描的结果通过或（OR）逻辑运算与 RLO 相结合。

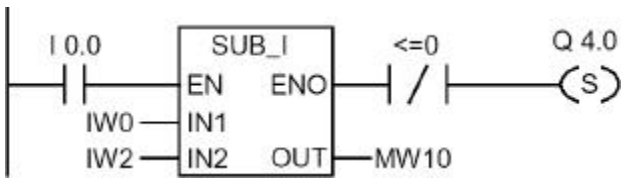
状态字

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

举例



方块通过 I0.0 的信号状态“ 1 ”启动。如果 IW0 的值小于或等于 IW2 的值，则算术运算功能“ IW0 - IW2 ”的结果小于或等于“ 0 ”。如果功能正确执行，并且结果小于或等于“ 0 ”，则 Q4.0 置位。



如果功能正确执行，并且结果不小于或不等于“ 0 ”，则 Q4.0 置位。

13 定时器指令

13.1 定时器指令概述

说明

关于正确时间的设定和选择信息，请参见“存储区中定时器的存储单元和定时器的组成部分”。

下述定时器指令可供使用：

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

13.2 存储区中定时器的存储单元和定时器的组成部分

存储器区域

在 CPU 的存储器中，为定时器保留有存储区。该存储区为每一定时器地址保留一个 16 位的字。梯形逻辑指令集支持 256 个定时器。请参考有关 CPU 的技术资料，以建立有效数量的定时器字。

下列功能可以访问定时器存储区：

- 定时器指令
- 利用时钟计时刷新定时器字。这是 CPU 在 CPU 模式下的功能，按时基规定的时间间隔为单位减少给定时间值，一直到时间值等于“0”。

时间值

定时器字的位 0 至位 9 包含二进制码的时间值。时间值按单位个数给出。时间刷新按时基规定的时间间隔为单位减少时间值。时间值逐渐连续减少，一直到等于“0”。时间值可以以二进制、十六进制和二-十进制（BCD）格式输入累加器 1 的低位字。

你可以使用下列格式预装一个时间值：

- W#16#wxyz
  - 其中，W = 时基（即时间间隔或分辨率）
  - 其中，xyz = 二-十进制格式的时间值

- S5T#aH_bM_cS_dMS
  - 其中，H = 小时，M = 分钟，S = 秒，MS = 毫秒；a、b、c、d 由用户定义。
  - 时基自动选择，时间值按其所取时基取整为下一个较小的数。

你可以输入的最大时间值是 9,990 秒，或 2H_46M_30S。

S5TIME#4S = 4 秒

s5t#2h_15m = 2 小时和 15 分钟

S5T#1H_12M_18S = 1 小时 12 分钟 18 秒

时基

定时器字的位 12 和位 13 包含二进制码的时基。时基定义时间值递减的单位时间间隔。最小时基为 10ms；最大时基为 10s。

时基	时基的二进制码
10 ms	00
100 ms	01
1s	10
10s	11

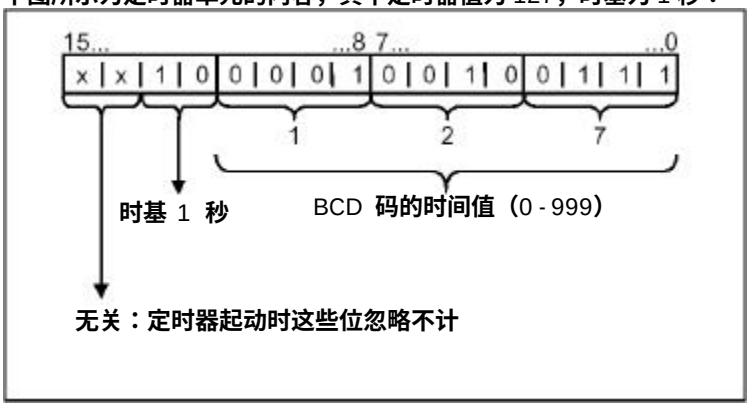
数值不允许超过 2h46m30s。对于范围极限分辨率太高的时间值（例如，2h10ms），将向下舍入为一个有效的分辨率。S5TIME 的一般格式具有如下所示的范围和分辨率：

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

定时器单元中的位组态

当定时器启动时，定时器单元的内容用作时间值。定时器单元的位 0 至位 11 为二 - 十进制格式的时间值（BCD 格式：四位一组表示一位十进制数值的二进制码）。位 12 和位 13 包含二进制码的时基。

下图所示为定时器单元的内容，其中定时器值为 127，时基为 1 秒：

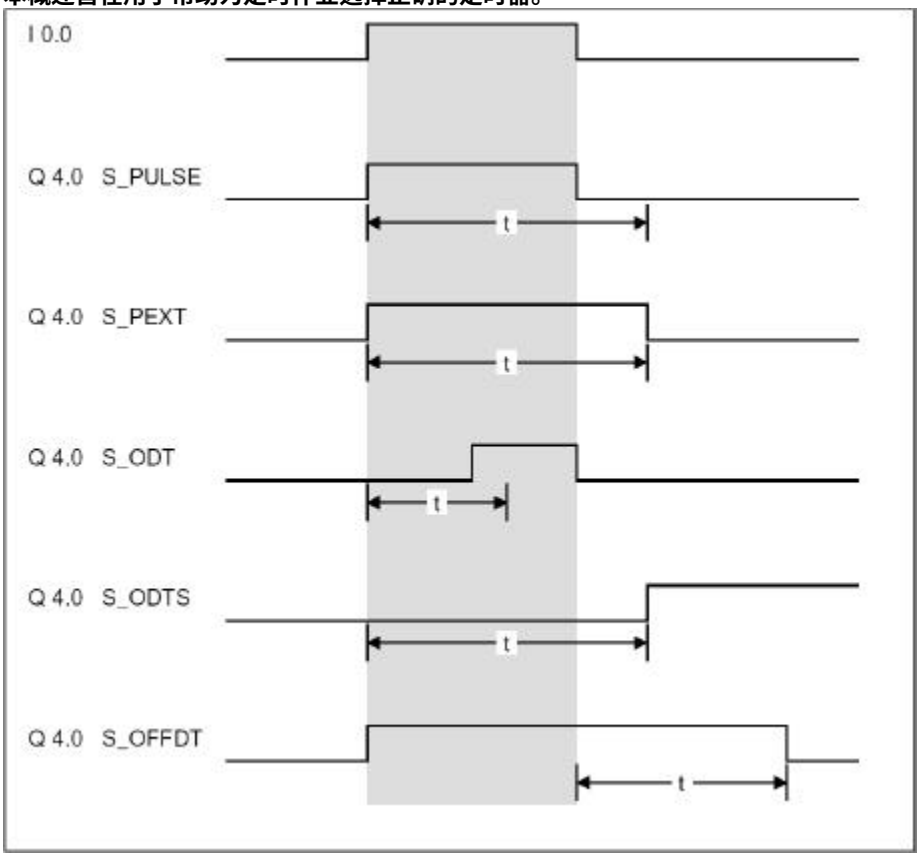


读时间值和时基

每一定时器方块提供两个输出，BI 和 BCD，以字存储单元来表示。BI 输出提供二进制格式的时间值。BCD 输出提供时基和二-十进制（BCD）格式的时间值。

正确选择定时器

本概述旨在用于帮助为定时作业选择正确的定时器。

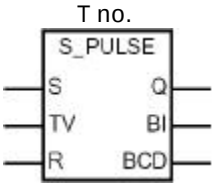


定时器	说明
S_PULSE 脉冲定时器	输出信号为“1”的最大时间等于设定的时间值 t。如果输入信号变为“0”，则输出信号为“1”的时间较短。
S_PEXT 扩展脉冲定时器	不管输入信号为“1”的时间有多长，输出信号为“1”的时间长度等于设定的时间值。
S_ODT 接通延时定时器	只有当设定的时间已经结束并且输入信号仍为“1”时，输出信号才从“0”变为“1”。
S_ODTS 保持型接通延时定时器	只有当设定的时间已经结束时，输出信号才从“0”变为“1”，而不管输入信号为“1”的时间有多长。
S_OFFDT 断电延时定时器	当输入信号变为“1”或定时器在运行时，输出信号变为“1”。当输入信号从“1”变为“0”时，定时器启动。

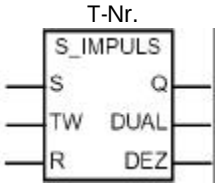
13.3 S_PULSE 脉冲 S5 定时器

符号

英文



德文



参数（英文）	参数（德文）	数据类型	存储区域	说明
T no.	T-Nr.	TIMER	T	定时器标识号，范围与 CPU 有关
S	S	BOOL	I，Q，M，L，D	起动输入端
TV	TW	S5TIME	I，Q，M，L，D	预置时间值
R	R	BOOL	I，Q，M，L，D	复位输入端
BI	DUAL	WORD	I，Q，M，L，D	剩余时间值，整数格式
BCD	DEZ	WORD	I，Q，M，L，D	剩余时间值，BCD 格式
Q	Q	BOOL	I，Q，M，L，D	定时器的状态

说明

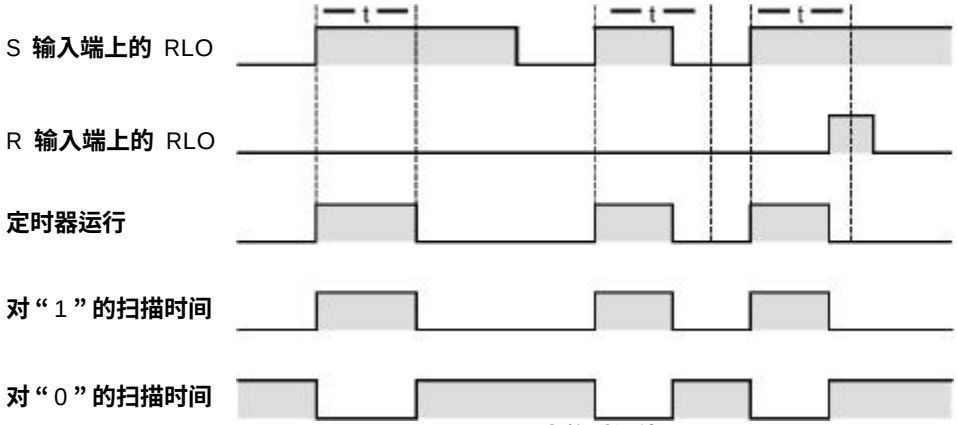
S_PULSE（脉冲 S5 定时器指令）用于在起动（S）输入端上出现上升沿时，起动指定的定时器。为了起动定时器，信号变化总是必要的。只要 S 输入端的信号状态为“1”，则定时器就连续地以 TV 输入端上设定的时间值运行。只要定时器一运行，输出 Q 上的信号状态就为“1”。如果在时间间隔结束之前，在 S 输入端出现从“1”到“0”的变化，则定时器停止运行。此时，输出 Q 的信号状态为“0”

当定时器运行时，如果定时器复位（R）输入端从“0”变为“1”，则定时器复位。同时当前时间和时基清零。如果定时器未运行，则定时器的 R 输入端为逻辑“1”对定时器没有影响。

当前的时间值可以在输出 BI 和 BCD 扫描出来。BI 上的时间值为二进制值，BCD 上的时间值为 BCD 码。当前的时间值等于初始 TV 值减去定时器起动以来的历时时间。

时序图

脉冲定时器特性：

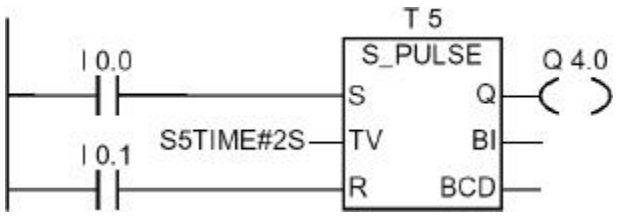


t = 设定的时间值

状态字

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

举例



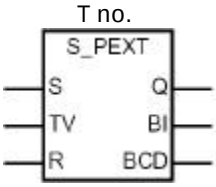
如果输入端 I0.0 的信号状态从 “ 0 ” 变为 “ 1 ” （RLO 出现上升沿），则起动定时器 T5。只要 I0.0 为 “ 1 ”，则定时器连续运行 2 秒钟的设定时间。如果在定时器结束之前，I0.0 的信号状态从 “ 1 ” 变为 “ 0 ”，则定时器停止运行。当定时器正在运行时，如果 I0.1 的信号状态从 “ 0 ” 变为 “ 1 ”，则定时器复位。

只要定时器在运行，则输出 Q4.0 为逻辑 “ 1 ”；如果时间结束或定时器复位，则输出 Q4.0 为逻辑 “ 0 ”。

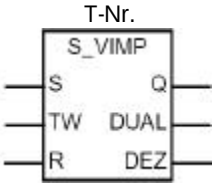
13.4 S_PEXT 扩展脉冲 S5 定时器

符号

英文



德文



参数（英文）	参数（德文）	数据类型	存储区域	说明
T no.	T-Nr.	TIMER	T	定时器标识号，范围与 CPU 有关
S	S	BOOL	I，Q，M，L，D	起动输入端
TV	TW	S5TIME	I，Q，M，L，D	预置时间值
R	R	BOOL	I，Q，M，L，D	复位输入端
BI	DUAL	WORD	I，Q，M，L，D	剩余时间值，整数格式
BCD	DEZ	WORD	I，Q，M，L，D	剩余时间值，BCD 格式
Q	Q	BOOL	I，Q，M，L，D	定时器的状态

说明

S_PEXT（扩展脉冲 S5 定时器指令）用于在起动（S）输入端上出现上升沿时，起动指定的定时器。为了起动定时器，信号变化总是必要的。即使在时间结束之前在 S 输入端的信号状态为“0”，定时器还是按 TV 输入端上设定的时间间隔继续运行。只要定时器一运行，输出 Q 上的信号状态就为“1”。当定时器正在运行时，如果输入端 S 的信号状态从“0”变为“1”，则定时器以预置时间值重新启动（“重新触发”）。

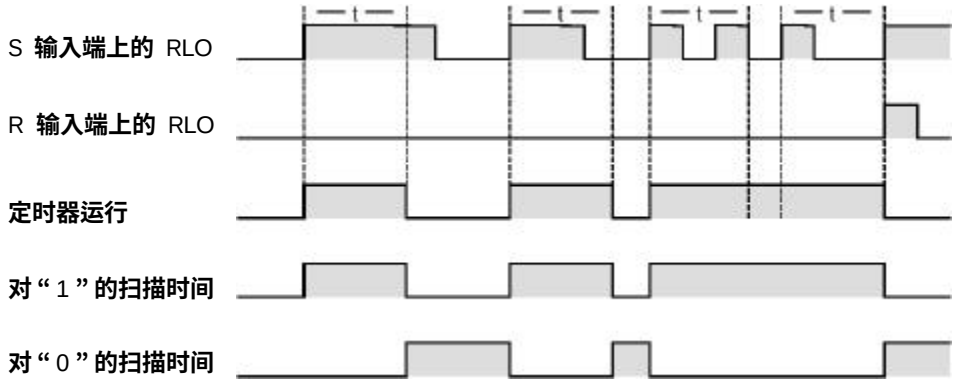
当定时器运行时，如果复位（R）输入端从“0”变为“1”，则定时器复位。同时当前时间和时基清零。

当前的时间值可以在输出 BI 和 BCD 扫描出来。BI 上的时间值为二进制值，BCD 上的时间值为 BCD 码。当前的时间值等于初始 TV 值减去定时器起动以来的历时时间。

请参见“存储区中定时器的存储单元和定时器的组成部分”。

时序图

扩展脉冲定时器特性：

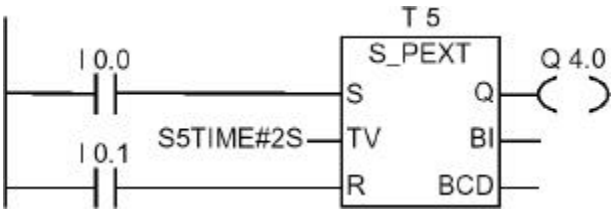


t = 设定的时间值

状态字

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

举例

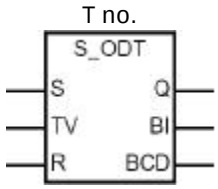


如果输入端 I0.0 的信号状态从 “ 0 ” 变为 “ 1 ” (RLO 出现上升沿)，则起动定时器 T5。定时器按规定的 2 秒（2s）继续运行，而不管输入端 S 上是否出现下降沿。如果在定时器结束之前，I0.0 的信号状态从 “ 0 ” 变为 “ 1 ”，则定时器重新启动。只要定时器一运行，则输出 Q4.0 上的信号状态就为逻辑 “ 1 ”。

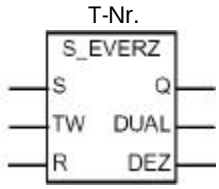
13.5 S_ODT 接通延时 S5 定时器

符号

英文



德文



参数（英文）	参数（德文）	数据类型	存储区域	说明
T no.	T-Nr.	TIMER	T	定时器标识号，范围与 CPU 有关
S	S	BOOL	I，Q，M，L，D	起动输入端
TV	TW	S5TIME	I，Q，M，L，D	预置时间值
R	R	BOOL	I，Q，M，L，D	复位输入端
BI	DUAL	WORD	I，Q，M，L，D	剩余时间值，整数格式
BCD	DEZ	WORD	I，Q，M，L，D	剩余时间值，BCD 格式
Q	Q	BOOL	I，Q，M，L，D	定时器的状态

说明

S_ODT（接通延时 S5 定时器指令）用于在起动（S）输入端上出现上升沿时，起动指定的定时器。为了起动定时器，信号变化总是必要的。只要 S 输入端的信号状态为“ 1 ”，则定时器就按输入端 TV 上设定的时间间隔继续运行。当时间已经结束，未出现错误并且 S 输入端上的信号状态仍为“ 1 ”，则输出 Q 的信号状态为“ 1 ”。当定时器正在运行时，如果 S 输入端的信号状态从“ 1 ”变为“ 0 ”，则定时器停止运行。此时，输出 Q 的信号状态为“ 0 ”。

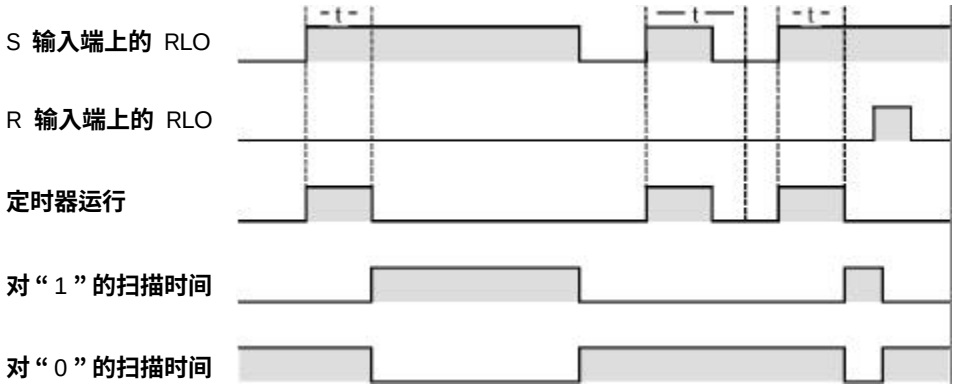
当定时器运行时，如果复位（R）输入端从“ 0 ”变为“ 1 ”，则定时器复位。同时当前时间和时基清零。此时，输出 Q 的信号状态为“ 0 ”。如果在输入端 R 的信号状态为逻辑“ 1 ”，同时定时器没有运行，输入端 S 为“ 1 ”，则定时器复位。

当前的时间值可以在输出 BI 和 BCD 扫描出来。BI 上的时间值为二进制值，BCD 上的时间值为 BCD 码。当前的时间值等于初始 TV 值减去定时器起动以来的历时时间。

请参见“ 存储区中定时器的存储单元和定时器的组成部分 ”。

时序图

接通延时定时器特性：

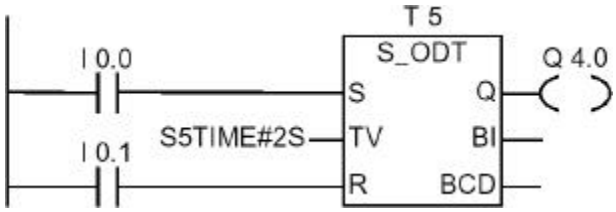


t = 设定的时间值

状态字

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

举例

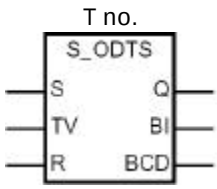


如果输入端 I0.0 的信号状态从“0”变为“1”(RLO 出现上升沿)，则起动定时器 T5。如果规定的 2 秒时间已结束，输入 I0.0 的信号状态仍为“1”，则输出 Q4.0 为“1”。如果输入 I0.0 的信号状态从“1”变为“0”，则定时器停止运行，Q4.0 为“0”(如果 I0.1 的信号状态从“0”变为“1”，则定时器复位，而不管定时器是否正在运行)。

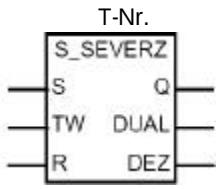
13.6 S_ODTS 保持型接通延时 S5 定时器

符号

英文



德文



参数(英文)	参数(德文)	数据类型	存储区域	说明
T no.	T-Nr.	TIMER	T	定时器标识号，范围与 CPU 有关
S	S	BOOL	I，Q，M，L，D	起动输入端
TV	TW	S5TIME	I，Q，M，L，D	预置时间值
R	R	BOOL	I，Q，M，L，D	复位输入端
BI	DUAL	WORD	I，Q，M，L，D	剩余时间值，整数格式
BCD	DEZ	WORD	I，Q，M，L，D	剩余时间值，BCD 格式
Q	Q	BOOL	I，Q，M，L，D	定时器的状态

说明

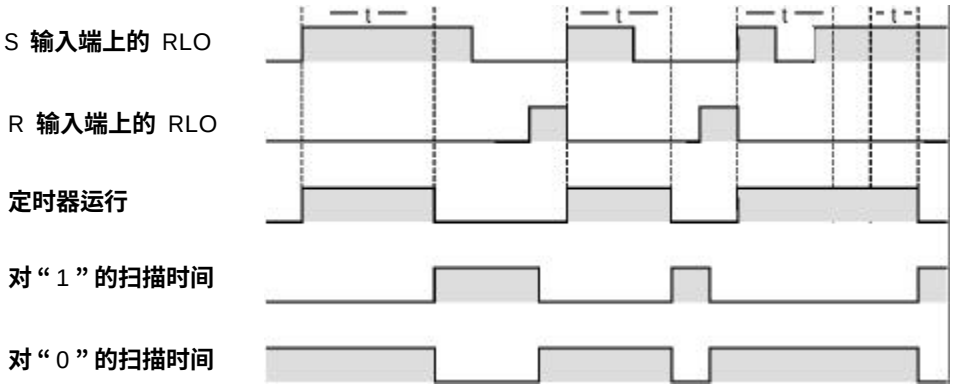
S_ODTS（保持型接通延时 S5 定时器指令）用于在起动（S）输入端上出现上升沿时，起动指定的定时器。为了起动定时器，信号变化总是必要的。即使在时间结束之前在 S 输入端的信号状态变为“0”，定时器还是按 TV 输入端上设定的时间间隔继续运行。当时间已经结束，不管 S 输入端上的信号状态如何，则输出 Q 的信号状态为“1”。当定时器正在运行时，如果输入端 S 的信号状态从“0”变为“1”，则定时器以预置时间值重新启动（“重新触发”）。

如果复位（R）输入端从“0”变为“1”，则定时器复位，而不管在 S 输入端上的 RLO 状态。此时，输出 Q 的信号状态为“0”。

当前的时间值可以在输出 BI 和 BCD 扫描出来。BI 上的时间值为二进制值，BCD 上的时间值为 BCD 码。当前的时间值等于初始 TV 值减去定时器起动以来的历时时间。

时序图

保持型接通延时定时器特性：

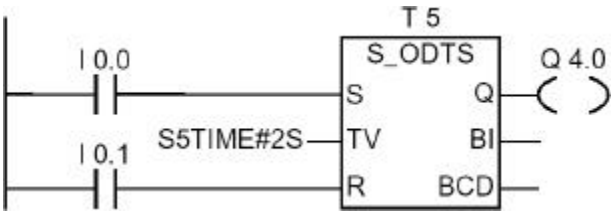


t = 设定的时间值

状态字

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

举例

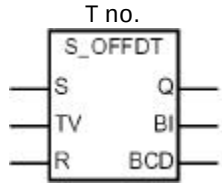


如果输入端 I0.0 的信号状态从 “ 0 ” 变为 “ 1 ” (RLO 出现上升沿)，则起动定时器 T5。定时器继续运行，而不管 I0.0 的信号状态是否从 “ 1 ” 变为 “ 0 ”。如果在定时器结束之前，I0.0 的信号状态从 “ 0 ” 变为 “ 1 ”，则定时器重新启动。如果时间已结束，则输出 Q4.0 为 “ 1 ”。（如果输入端 I0.1 的信号状态从 “ 0 ” 变为 “ 1 ”，则定时器复位，而不管 S 端 RLO 的状态如何。）

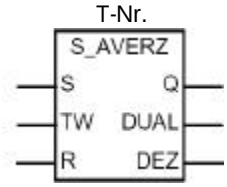
13.7 S_OFFDT 断电延时 S5 定时器

符号

英文



德文



参数（英文）	参数（德文）	数据类型	存储区域	说明
T no.	T-Nr.	TIMER	T	定时器标识号，范围与 CPU 有关
S	S	BOOL	I，Q，M，L，D	起动输入端
TV	TW	S5TIME	I，Q，M，L，D	预置时间值
R	R	BOOL	I，Q，M，L，D	复位输入端
BI	DUAL	WORD	I，Q，M，L，D	剩余时间值，整数格式
BCD	DEZ	WORD	I，Q，M，L，D	剩余时间值，BCD 格式
Q	Q	BOOL	I，Q，M，L，D	定时器的状态

说明

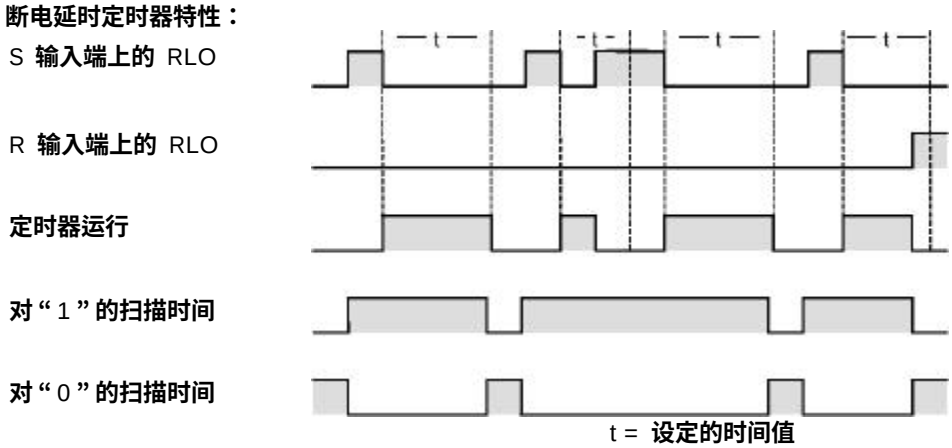
S_OFFDT（断电延时 S5 定时器指令）用于在起动（S）输入端上出现下降沿时，起动指定的定时器。为了起动定时器，信号变化总是必要的。如果 S 输入端的信号状态为“1”，或当定时器运行时，则输出 Q 上的信号状态为“1”。当定时器运行时，如果 S 输入端的信号状态从“0”变为“1”，则定时器复位。一直到 S 输入端的信号状态从“1”变为“0”，定时器才重新启动。

当定时器运行时，如果复位（R）输入端从“0”变为“1”，则定时器复位。

当前的时间值可以在输出 BI 和 BCD 扫描出来。BI 上的时间值为二进制值，BCD 上的时间值为 BCD 码。当前的时间值等于初始 TV 值减去定时器起动以来的历时时间。

时序图

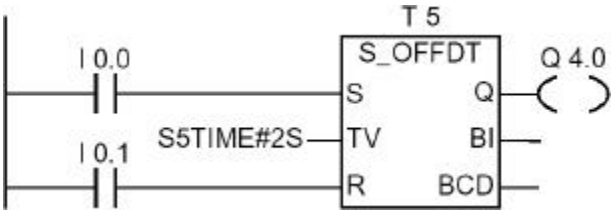
断电延时定时器特性：



状态字

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

举例



如果 I0.0 的信号状态从“1”变为“0”，则起动定时器。当 I0.0 为“1”或定时器在运行时，则输出 Q4.0 为“1”。（当定时器正在运行时，如果 I0.1 的信号状态从“0”变为“1”，则定时器复位）。

13.8 ---(SP) 脉冲定时器线圈

符号

英文	**德文**
<T no..>	<T no.>
---(SP)	---(SI)
<时间值>	**<时间值>**

参数	**数据类型**	**存储区域**	**说明**
<T no.>	TIMER	T	**定时器标识号，范围与 CPU 有关**
<时间值>	S5TIME	I，Q，M，L，D	**预置时间值**

说明

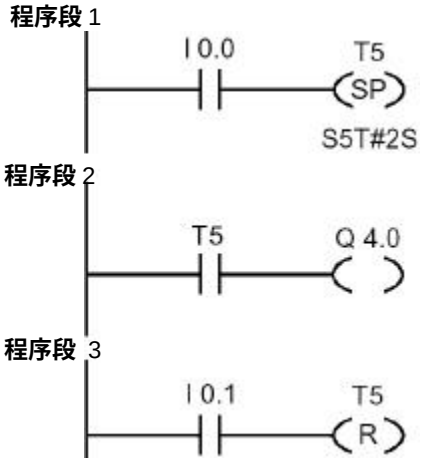
---(SP)**（脉冲定时器线圈指令）用于在 RLO 状态出现上升沿时，起动指定的具有给定时间值（<时间值>）的定时器。只要 RLO 为正（“1”），则定时器就按设定的时间运行。只要定时器一运行，则该定时器的信号状态就为“1”。如果在规定时间值过去之前，RLO 从“1”变为“0”，则定时器停止运行。在这种情况下，“1”信号扫描产生结果“0”。**

请参见“存储区中定时器的存储单元和定时器的组成部分”和“S_PULSE（脉冲 S5 定时器）”。

状态字

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

举例



如果输入端 I0.0 的信号状态从“ 0 ”变为“ 1 ”(RLO 出现上升沿)，则起动定时器 T5。只要输入 I0.0 的信号状态为“ 1 ”，则定时器连续运行 2 秒钟的设定时间。如果在规定的时间过去之前，输入 I0.0 的信号状态从“ 1 ”变为“ 0 ”，则定时器停止运行。

只要定时器一运行，输出 Q4.0 上的信号状态就为“ 1 ”。如果 I0.1 的信号状态从“ 0 ”变为“ 1 ”，则复位定时器 T5，定时器停止，并清零剩余时间值。

13.9 ---(SE) 扩展脉冲定时器线圈

符号

英文	德文
<T no.>	<T no.>
---(SE)	---(SV)
<时间值>	<时间值>

参数	数据类型	存储区域	说明
<T no.>	TIMER	T	定时器标识号，范围与 CPU 有关
<时间值>	S5TIME	I，Q，M，L，D	预置时间值

说明

---(SE)**(扩展脉冲定时器线圈指令）用于在 RLO 状态出现上升沿时，起动指定的具有给定时间值（<时间值>）的定时器。即使在时间过去之前 RLO 变为“ 0 ”，定时器仍按设定的时间运行。只要定时器一运行，则该定时器的信号状态就为“ 1 ”。当定时器正在运行时，如果 RLO 从“ 0 ”变为“ 1 ”，则定时器以预置时间值重新启动（“ 重新触发 ”）。**

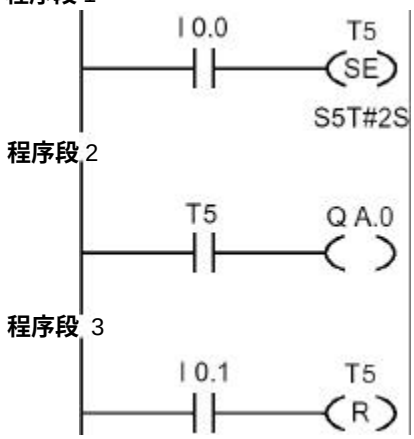
请参见“ 存储区中定时器的存储单元和定时器的组成部分 ”和“ S_PEXT（扩展脉冲 S5 定时器） ”。

状态字

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

举例

程序段 1



如果输入端 I0.0 **的信号状态从“** 0 **”变为“** 1 **”（**RLO **出现上升沿），则起动定时器** T5**。定时器继续运行，而不管** RLO **是否出现下降沿。如果在定时器结束之前，**I0.0 **的信号状态从“** 0 **”变为“** 1 **”，则定时器重新启动。只要定时器一运行，输出** Q4.0 **上的信号状态就为“** 1 **”。如果** I0.1 **的信号状态从“** 0 **”变为“** 1 **”，则复位定时器** T5**，定时器停止，并清零剩余时间值。**

13.10 ---(SD) 接通延时定时器线圈

符号

英文	**德文**
<T no..>	<T no.>
---(SD)	---(SE)
<时间值>	**<时间值>**

参数	数据类型	存储区域	说明
<T no.>	TIMER	T	**定时器标识号，范围与** CPU **有关**
<时间值>	S5TIME	I，Q，M，L，D	**预置时间值**

说明

---(SD)**（接通延时定时器线圈指令）用于在** RLO **状态出现上升沿时，起动指定的具有给定时间值（<时间值>）的定时器。当 <时间值> 已经结束，未出现错误并且** RLO **仍为“** 1 **”，则该定时器的信号状态为“** 1 **”。当定时器运行时，如果** RLO **从“** 1 **”变为“** 0 **”，则定时器复位。在这种情况下，“** 1 **”信号扫描产生结果“** 0 **”。**

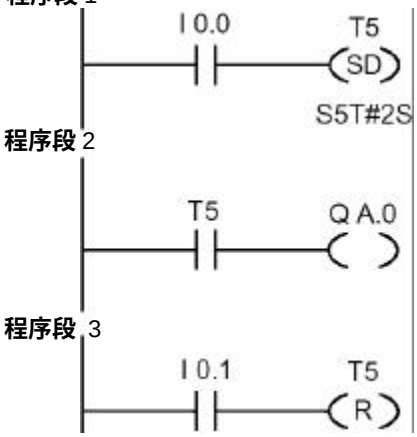
请参见“存储区中定时器的存储单元和定时器的组成部分”和“S_ODT（接通延时 S5 定时器）”。

状态字

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

举例

程序段 1



程序段 2

程序段 3

如果输入端 I0.0 的信号状态从“0”变为“1”（RLO 出现上升沿），则起动定时器 T5。如果时间已结束，输入 I0.0 的信号状态仍为“1”，则输出 Q4.0 的信号状态为“1”。如果输入 I0.0 的信号状态从“1”变为“0”，则定时器保持停止，输出 Q4.0 的信号状态为“0”。如果 I0.1 的信号状态从“0”变为“1”，则复位定时器 T5，定时器停止，并清零剩余时间值。

13.11 ---(SS) 保持型接通延时定时器线圈

符号

英文	德文
<T no..>	<T no.>
---(SS)	---(SS)
<时间值>	<时间值>

参数	数据类型	存储区域	说明
<T no.>	TIMER	T	定时器标识号，范围与 CPU 有关
<时间值>	S5TIME	I，Q，M，L，D	预置时间值

说明

---(SS)（保持型接通延时定时器线圈指令）用于在 RLO 状态出现上升沿时，起动指定的定时器。如果时间值已经过去，则该定时器的信号状态就为“ 1 ”。只有在定时器复位后，定时器才能重新启动。只有通过复位才能使定时器的信号状态置为“ 0 ”。

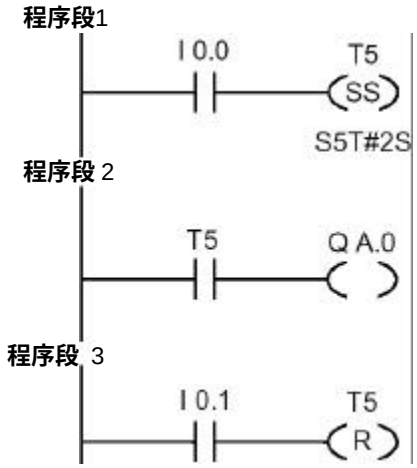
当定时器正在运行时，如果 RLO 从“ 0 ”变为“ 1 ”，则定时器以预置时间值重新启动。

请参见“ 存储区中定时器的存储单元和定时器的组成部分 ”和“ S_ODTS（保持型接通延时 S5 定时器） ”。

状态字

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

举例



如果输入端 I0.0 的信号状态从“ 0 ”变为“ 1 ”（RLO 出现上升沿），则起动定时器 T5。如果在定时器结束之前，I0.0 的信号状态从“ 0 ”变为“ 1 ”，则定时器重新启动。如果时间已结束，则输出 Q4.0 为“ 1 ”。如果 I0.1 的信号状态为“ 1 ”，则复位定时器 T5，定时器停止，并清零剩余时间值。

13.12 ---(SF) 断开延时定时器线圈

符号

英文	德文
<T no..>	<T no.>
---(SF)	---(SA)
<时间值>	<时间值>

参数	数据类型	存储区域	说明
<T no.>	TIMER	T	定时器标识号，范围与 CPU 有关
<时间值>	S5TIME	I，Q，M，L，D	预置时间值

说明

---(SF)（断开延时定时器线圈指令）用于在 RLO 状态出现下降沿时，起动指定的定时器。当 RLO 为“ 1 ”时，或在 <时间值> 间隔内只要定时器运行，该定时器就为“ 1 ”。当定时器运行时，如果 RLO 从“ 0 ”变为“ 1 ”，则定时器复位。如果 RLO 从“ 1 ”变为“ 0 ”，则总是重新起动定时器。

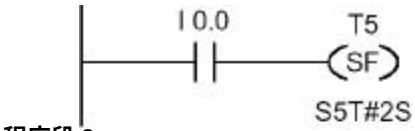
请参见“ 存储区中定时器的存储单元和定时器的组成部分 ”和“ S_OFFDT（断电延时 S5 定时器） ”。

状态字

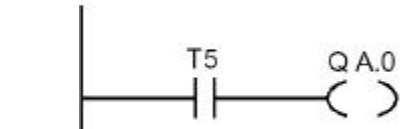
	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
写	-	-	-	-	-	0	-	-	0

举例

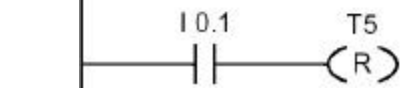
程序段1



程序段 2



程序段 3



如果 I0.0 的信号状态从“ 1 ”变为“ 0 ”，则起动定时器。

当 I0.0 为“ 1 ”或定时器在运行时，则输出 Q4.0 为“ 1 ”。如果 I0.1 的信号状态从“ 0 ”变为“ 1 ”，则复位定时器 T5，定时器停止，并清零剩余时间值。

14 字逻辑指令

14.1 字逻辑指令概述

说明

字逻辑指令按照布尔逻辑将成对的字（16 位）和双字（32 位）逐位进行比较。

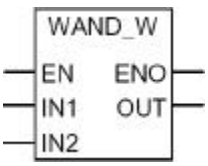
如果输出 OUT **的结果不等于“** 0 **”，则状态字的位** CC 1 **被置为“** 1 **”。**

如果输出 OUT **的结果等于“** 0 **”，则状态字的位** CC 1 **被置为“** 0 **”。下述字逻辑指令可供使用：**

- WAND_W **字和字相“与（**AND**）”**
- WOR_W **字和字相“或（**OR**）”**
- WXOR_W **字和字相“异或（**XOR**）”**
- WAND_DW **双字和双字相“与（**AND**）”**
- WOR_DW **双字和双字相“或（**OR**）”**
- WXOR_DW **双字和双字相“异或（**XOR**）”**

14.2 WAND_W 字和字相“与”

符号



参数	数据类型	存储区域	说明
EN	BOOL	I，Q，M，L，D	使能输入
ENO	BOOL	I，Q，M，L，D	使能输出
IN1	WORD	I，Q，M，L，D	第一个逻辑运算值
IN2	WORD	I，Q，M，L，D	第二个逻辑运算值
OUT	WORD	I，Q，M，L，D	逻辑运算的结果字

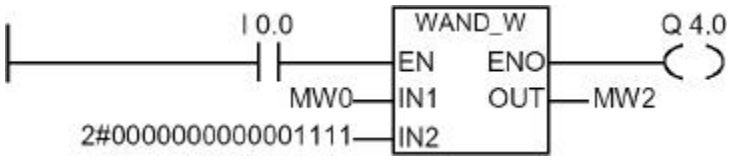
说明

WAND_W**（字和字相“与”指令）通过使能输入（**EN**）的信号状态“**1**”激活，并将输入** IN1 **和** IN2 **表示的两个字值逐位进行“与（**AND**）”运算。数值用纯二进制位的形式表示。其结果可以在输出** OUT **中扫描。**ENO **和** EN **具有相同的逻辑状态。**

状态字

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

举例



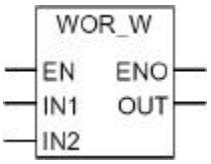
如果 I0.0 = “1”，**则执行指令。只有** MW0 **的位** 0 **到位** 3 **与之相关，**MW0 **的其余位被** IN2 **字位屏蔽。**

MW0	=	01010101 01010101
IN2	=	00000000 00001111
MW0 AND IN2 = MW2	=	00000000 00000101

如果执行指令，则 Q4.0 **为“**1**”。**

14.3 WOR_W 字和字相“或”

符号



参数	数据类型	存储区域	说明
EN	BOOL	I，Q，M，L，D	使能输入
ENO	BOOL	I，Q，M，L，D	使能输出
IN1	WORD	I，Q，M，L，D	第一个逻辑运算值
IN2	WORD	I，Q，M，L，D	第二个逻辑运算值
OUT	WORD	I，Q，M，L，D	逻辑运算的结果字

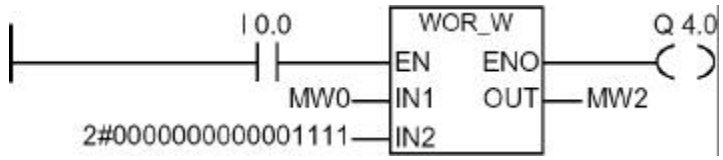
说明

WOR_W**（字和字相“或”指令）通过使能输入（**EN**）的信号状态“**1**”激活，并将输入** IN1 **和** IN2 **表示的两个字值逐位进行“或（**OR**）”运算。数值用纯二进制位的形式表示。其结果可以在输出** OUT **中扫描。**ENO **和** EN **具有相同的逻辑状态。**

状态字

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

举例



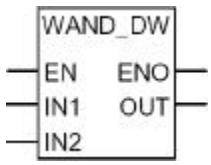
如果 I0.0 = **“** 1 **”，则执行指令。位** 0 **到位** 3 **被置为“** 1 **”，所有其它** MW0 **位不变。**

MW0	=	01010101 01010101
IN2	=	00000000 00001111
MW0 OR IN2=MW2	=	01010101 01011111

如果执行指令，则 Q4.0 **为“** 1 **”。**

14.4 WAND_DW **双字和双字相“与”**

符号



参数	数据类型	存储区域	说明
EN	BOOL	I，Q，M，L，D	使能输入
ENO	BOOL	I，Q，M，L，D	使能输出
IN1	DWORD	I，Q，M，L，D	第一个逻辑运算值
IN2	DWORD	I，Q，M，L，D	第二个逻辑运算值
OUT	DWORD	I，Q，M，L，D	逻辑运算的结果双字

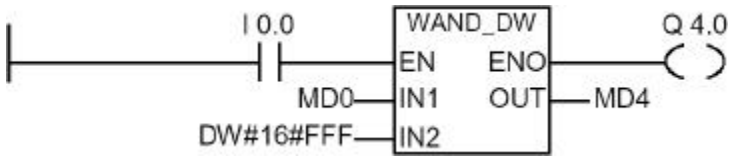
说明

WAND_DW**（双字和双字相“与”指令）通过使能输入（**EN**）的信号状态“** 1 **”激活，并将输入** IN1 **和** IN2 **表示的两个字值逐位进行“与（**AND**）”运算。数值用纯二进制位的形式表示。其结果可以在输出** OUT **中扫描。**ENO **和** EN **具有相同的逻辑状态。**

状态字

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

举例



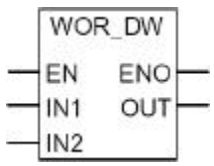
如果 I0.0 = “1”，**则执行指令。只有** MD0 **的位** 0 **到位** 11 **与之相关，**MD0 **的其余位被** IN2 **位屏蔽。**

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

如果执行指令，则 Q4.0 **为“**1**”。**

14.5 WOR_DW 双字和双字相“或”

符号



参数	数据类型	存储区域	说明
EN	BOOL	I，Q，M，L，D	使能输入
ENO	BOOL	I，Q，M，L，D	使能输出
IN1	DWORD	I，Q，M，L，D	第一个逻辑运算值
IN2	DWORD	I，Q，M，L，D	第二个逻辑运算值
OUT	DWORD	I，Q，M，L，D	逻辑运算的结果双字

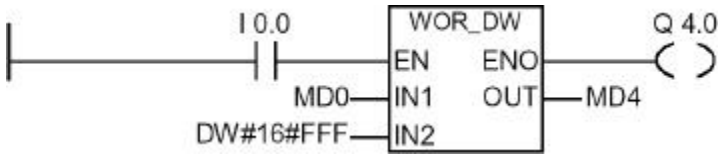
说明

WOR_DW（双字和双字相“或”指令）通过使能输入（EN）的信号状态“1”激活，并将输入 IN1 和 IN2 表示的两个字值逐位进行“或（OR）”运算。数值用纯二进制位的形式表示。其结果可以在输出 OUT 中扫描。ENO 和 EN 具有相同的逻辑状态。

状态字

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

举例



如果 I0.0 = **“** 1 **”，则执行指令。位** 0 **到位** 11 **被置为“** 1 **”，所有其它** MD0 **位不变。**

MD0 = 01010101 01010101 01010101 01010101

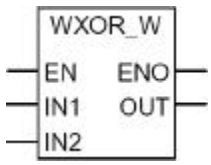
IN2 = 00000000 00000000 00001111 11111111

MD0 OR IN2 = MD4 = 01010101 01010101 01011111 11111111

如果执行指令，则 Q4.0 **为“** 1 **”。**

14.6 WXOR_W 字和字相“异或”

符号



参数	数据类型	存储区域	说明
EN	BOOL	I，Q，M，L，D	使能输入
ENO	BOOL	I，Q，M，L，D	使能输出
IN1	WORD	I，Q，M，L，D	第一个逻辑运算值
IN2	WORD	I，Q，M，L，D	第二个逻辑运算值
OUT	WORD	I，Q，M，L，D	逻辑运算的结果字

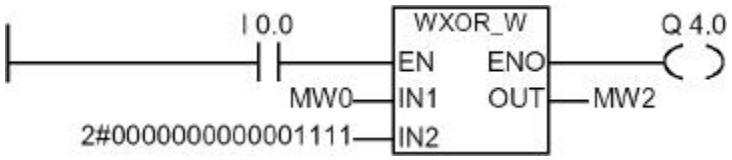
说明

WXOR_W（字和字相“异或”指令）通过使能输入（EN）的信号状态“1”激活，并将输入 IN1 和 IN2 表示的两个字值逐位进行“异或（XOR）”运算。数值用纯二进制位的形式表示。其结果可以在输出 OUT 中扫描。ENO 和 EN 具有相同的逻辑状态。

状态字

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

举例



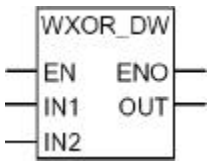
如果 I0.0 = “1”**，则执行指令：**

MW0	=	01010101 01010101
IN2	=	00000000 00001111
MW0 XOR IN2 = MW2	=	01010101 01011010

如果执行指令，则 Q4.0**为“**1**”。**

14.7 WXOR_DW 双字和双字相“异或”

符号



参数	数据类型	存储区域	说明
EN	BOOL	I，Q，M，L，D	使能输入
ENO	BOOL	I，Q，M，L，D	使能输出
IN1	DWORD	I，Q，M，L，D	第一个逻辑运算值
IN2	DWORD	I，Q，M，L，D	第二个逻辑运算值
OUT	DWORD	I，Q，M，L，D	逻辑运算的结果双字

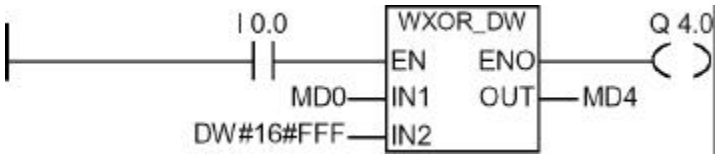
说明

WXOR_DW（双字和双字相“异或”指令）通过使能输入（EN）的信号状态“1”激活，并将输入 IN1 和 IN2 表示的两个字值逐位进行“异或（XOR）”运算。数值用纯二进制位的形式表示。其结果可以在输出 OUT 中扫描。ENO 和 EN 具有相同的逻辑状态。

状态字

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

举例



如果 I0.0 = **“** 1 **”，则执行指令：**

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

如果执行指令，则 Q4.0 **为“** 1 **”**

A 所有梯形逻辑指令一览

A.1 按英文助记符分类的LAD指令（国际）

英文助记符	德文助记符	程序元素分类	说明
---||---	---||---	位逻辑指令	常开接点（地址）
---|/|---	---|/|---	位逻辑指令	常闭接点（地址）
---( )	---( )	位逻辑指令	输出线圈
---(#)---	---(#)---	位逻辑指令	中间输出
==0 ---||---	==0 ---||---	状态位指令	结果位等于“0”
>0 ---||---	>0 ---||---	状态位指令	结果位大于“0”
>=0 ---||---	>=0 ---||---	状态位指令	结果位大于等于“0”
<=0 ---||---	<=0 ---||---	状态位指令	结果位小于等于“0”
<0 ---||---	<0 ---||---	状态位指令	结果位小于“0”
<>0 ---||---	<>0 ---||---	状态位指令	结果位不等于“0”
ABS	ABS	浮点算术运算指令	浮点数绝对值运算
ACOS	ACOS	浮点算术运算指令	浮点数反余弦运算
ADD_DI	ADD_DI	整数算术运算指令	双整数加法
ADD_I	ADD_I	整数算术运算指令	整数加法
ADD_R	ADD_R	浮点算术运算指令	实数加法
ASIN	ASIN	浮点算术运算指令	浮点数反正弦运算
ATAN	ATAN	浮点算术运算指令	浮点数反正切运算
BCD_DI	BCD_DI	转换指令	BCD 码转换为双整数
BCD_I	BCD_I	转换指令	BCD 码转换为整数
BR ---||---	BIE ---||---	状态位指令	异常位二进制结果
----(CALL)	----(CALL)	程序控制指令	从线圈调用 FC/SFC（无参数）
CALL_FB	CALL_FB	程序控制指令	从方块调用 FB
CALL_FC	CALL_FC	程序控制指令	从方块调用 FC
CALL_SFB	CALL_SFB	程序控制指令	从方块调用 SFB
CALL_SFC	CALL_SFC	程序控制指令	从方块调用 SFC
----(CD)	----(ZR)	计数器指令	减计数器线圈
CEIL	CEIL	转换指令	上取整
CMP >=D	CMP >=D	比较指令	双整数比较（==，<>，>，<，>=，<=）
CMP >=I	CMP >=I	比较指令	整数比较（==，<>，>，<，>=，<=）
CMP >=R	CMP >=R	比较指令	实数比较（==，<>，>，<，>=，<=）
COS	COS	浮点算术运算指令	浮点数余弦运算
----(CU)	---(ZV)	计数器指令	加计数器线圈
DI_BCD	DI_BCD	转换指令	双整数转换为 BCD 码
DI_R	DI_R	转换指令	双整数转换为浮点数
DIV_DI	DIV_DI	整数算术运算指令	双整数除法

英文助记符	德文助记符	程序元素分类	说明
DIV_I	DIV_I	整数算术运算指令	整数除法
DIV_R	DIV_R	浮点算术运算指令	实数除法
EXP	EXP	浮点算术运算指令	浮点数指数运算
FLOOR	FLOOR	转换指令	下取整
I_BCD	I_BCD	转换指令	整数转换为 BCD 码
I_DI	I_DI	转换指令	整数转换为双整数
INV_I	INV_I	转换指令	整数的二进制反码
INV_DI	INV_DI	转换指令	双整数的二进制反码
---(JMP)	---(JMP)	跳转指令	无条件跳转
---(JMP)	---(JMP)	跳转指令	条件跳转
---(JMPN)	---(JMPN)	跳转指令	若非则跳转
LABEL	LABEL	跳转指令	标号
LN	LN	浮点算术运算指令	浮点数自然对数运算
---(MCR>)	---(MCR>)	程序控制指令	主控继电器断开
---(MCR<)	---(MCR<)	程序控制指令	主控继电器接通
---(MCRA)	---(MCRA)	程序控制指令	主控继电器启动
---(MCRD)	---(MCRD)	程序控制指令	主控继电器停止
MOD_DI	MOD_DI	整数算术运算指令	回送余数的双整数
MOVE	MOVE	赋值指令	赋值
MUL_DI	MUL_DI	整数算术运算指令	双整数乘法
MUL_I	MUL_I	整数算术运算指令	整数乘法
MUL_R	MUL_R	浮点算术运算指令	实数乘法
---(N)---	---(N)---	位逻辑指令	RLO 下降沿检测
NEG	NEG	位逻辑指令	地址下降沿检测
NEG_DI	NEG_DI	转换指令	双整数的二进制补码
NEG_I	NEG_I	转换指令	整数的二进制补码
NEG_R	NEG_R	转换指令	浮点数求反
---|NOT|---	---|NOT|---	位逻辑指令	信号流反向
---(OPN)	---(OPN)	数据块调用指令	打开数据块：DB 或 DI
OS ---| |---	OS ---| |---	状态位指令	存储溢出异常位
OV ---| |---	OV ---| |---	状态位指令	溢出异常位
---(P)---	---(P)---	位逻辑指令	RLO 上升沿检测
POS	POS	位逻辑指令	地址上升沿检测
---(R)	---(R)	位逻辑指令	线圈复位
---(RET)	---(RET)	程序控制指令	返回
ROL_DW	ROL_DW	移位和循环指令	双字左循环
ROL_DW	ROL_DW	移位和循环指令	双字右循环
ROUND	ROUND	转换指令	舍入为双整数
RS	RS	位逻辑指令	复位置位触发器
---(S)	---(S)	位逻辑指令	线圈置位
---(SAVE)	---(SAVE)	位逻辑指令	将 RLO 存入 BR 存储器
---(SC)	---(SZ)	计数器指令	设置计数器值
S_CD	Z_RUECK	计数器指令	减计数器
S_CU	Z_VORW	计数器指令	加计数器

英文助记符	德文助记符	程序元素分类	说明
S_CUD	ZAEHLER	计数器指令	加-减计数器
---(SD)	---(SE)	定时器指令	接通延时定时器线圈
---(SE)	---(SV)	定时器指令	扩展脉冲定时器线圈
---(SF)	---(SA)	定时器指令	断开延时定时器线圈
SHL_DW	SHL_DW	移位和循环指令	双字左移
SHL_W	SHL_W	移位和循环指令	字左移
SHR_DI	SHR_DI	移位和循环指令	双整数右移
SHR_DW	SHR_DW	移位和循环指令	双字右移
SHR_I	SHR_I	移位和循环指令	整数右移
SHR_W	SHR_W	移位和循环指令	字右移
SIN	SIN	浮点算术运算指令	浮点数正弦运算
S_ODT	S_EVERZ	定时器指令	接通延时 S5 定时器
S_ODTS	S_SEVERZ	定时器指令	保持型接通延时 S5 定时器
S_OFFDT	S_AVERZ	定时器指令	断电延时 S5 定时器
---(SP)	---(SI)	定时器指令	脉冲定时器线圈
S_PEXT	S_VIMP	定时器指令	扩展脉冲 S5 定时器
S_PULSE	S_IMPULS	定时器指令	脉冲 S5 定时器
SQR	SQR	浮点算术运算指令	浮点数平方
SQRT	SQRT	浮点算术运算指令	浮点数平方根
SR	SR	位逻辑指令	置位复位触发器
---(SS)	---(SS)	定时器指令	保持型接通延时定时器线圈
SUB_DI	SUB_DI	整数算术运算指令	双整数减法
SUB_I	SUB_I	整数算术运算指令	整数减法
SUB_R	SUB_R	浮点算术运算指令	实数减法
TAN	TAN	浮点算术运算指令	浮点数正切运算
TRUNC	TRUNC	转换指令	舍去小数取整为双整数
UO ---||---	UO ---||---	状态位指令	无序异常位
WAND_DW	WAND_DW	字逻辑指令	双字和双字相“与”
WAND_W	WAND_W	字逻辑指令	字和字相“与”
WOR_DW	WOR_DW	字逻辑指令	双字和双字相“或”
WOR_W	WOR_W	字逻辑指令	字和字相“或”
WXOR_DW	WXOR_DW	字逻辑指令	双字和双字相“异或”
WXOR_W	WXOR_W	字逻辑指令	字和字相“异或”

A.2 按德文助记符分类的LAD指令（SIMATIC）

英文助记符	德文助记符	程序元素分类	说明
---| |---	---| |---	位逻辑指令	常开接点（地址）
---|/|---	---|/|---	位逻辑指令	常闭接点（地址）
---( )	---( )	位逻辑指令	输出线圈
---(#)---	---(#)---	位逻辑指令	中间输出
==0 ---| |---	==0 ---| |---	状态位指令	结果位等于“0”
>0 ---| |---	>0 ---| |---	状态位指令	结果位大于“0”
>=0 ---| |---	>=0 ---| |---	状态位指令	结果位大于等于“0”
<=0 ---| |---	<=0 ---| |---	状态位指令	结果位小于等于“0”
<0 ---| |---	<0 ---| |---	状态位指令	结果位小于“0”
<>0 ---| |---	<>0 ---| |---	状态位指令	结果位不等于“0”
ABS	ABS	浮点算术运算指令	浮点数绝对值运算
ACOS	ACOS	浮点算术运算指令	浮点数反余弦运算
ADD_DI	ADD_DI	整数算术运算指令	双整数加法
ADD_I	ADD_I	整数算术运算指令	整数加法
ADD_R	ADD_R	浮点算术运算指令	实数加法
ASIN	ASIN	浮点算术运算指令	浮点数反正弦运算
ATAN	ATAN	浮点算术运算指令	浮点数反正切运算
BCD_DI	BCD_DI	转换指令	BCD 码转换为双整数
BCD_I	BCD_I	转换指令	BCD 码转换为整数
BIE ---| |---	BR ---| |---	状态位指令	异常位二进制结果
----(CALL)	----(CALL)	程序控制指令	从线圈调用 FC/SFC（无参数）
CALL_FB	CALL_FB	程序控制指令	从方块调用 FB
CALL_FC	CALL_FC	程序控制指令	从方块调用 FC
CALL_SFB	CALL_SFB	程序控制指令	从方块调用 SFB
CALL_SFC	CALL_SFC	程序控制指令	从方块调用 SFC
CEIL	CEIL	转换指令	上取整
CMP >=D	CMP >=D	比较指令	双整数比较（==，<>，>，<，>=，<=）
CMP >=I	CMP >=I	比较指令	整数比较（==，<>，>，<，>=，<=）
CMP >=R	CMP >=R	比较指令	实数比较（==，<>，>，<，>=，<=）
COS	COS	浮点算术运算指令	浮点数余弦运算
DI_BCD	DI_BCD	转换指令	双整数转换为 BCD 码
DI_R	DI_R	转换指令	双整数转换为浮点数
DIV_DI	DIV_DI	整数算术运算指令	双整数除法
DIV_I	DIV_I	整数算术运算指令	整数除法
DIV_R	DIV_R	浮点算术运算指令	实数除法

英文助记符	德文助记符	程序元素分类	说明
EXP	EXP	浮点算术运算指令	浮点数指数运算
FLOOR	FLOOR	转换指令	下取整
I_BCD	I_BCD	转换指令	整数转换为 BCD 码
I_DI	I_DI	转换指令	整数转换为双整数
INV_I	INV_I	转换指令	整数的二进制反码
INV_DI	INV_DI	转换指令	双整数的二进制反码
---(JMP)	---(JMP)	跳转指令	条件跳转
---(JMP)	---(JMP)	跳转指令	无条件跳转
---(JMPN)	---(JMPN)	跳转指令	若非则跳转
LABEL	LABEL	跳转指令	标号
LN	LN	浮点算术运算指令	浮点数自然对数运算
---(MCR>)	---(MCR>)	程序控制指令	主控继电器断开
---(MCR<)	---(MCR<)	程序控制指令	主控继电器接通
---(MCRA)	---(MCRA)	程序控制指令	主控继电器启动
---(MCRD)	---(MCRD)	程序控制指令	主控继电器停止
MOD_DI	MOD_DI	整数算术运算指令	回送余数的双整数
MOVE	MOVE	赋值指令	赋值
MUL_DI	MUL_DI	整数算术运算指令	双整数乘法
MUL_I	MUL_I	整数算术运算指令	整数乘法
MUL_R	MUL_R	浮点算术运算指令	实数乘法
---(N)---	---(N)---	位逻辑指令	RLO 下降沿检测
NEG	NEG	位逻辑指令	地址下降沿检测
NEG_DI	NEG_DI	转换指令	双整数的二进制补码
NEG_I	NEG_I	转换指令	整数的二进制补码
NEG_R	NEG_R	转换指令	浮点数求反
---|NOT|---	---|NOT|---	位逻辑指令	信号流反向
---(OPN)	---(OPN)	数据块调用指令	打开数据块：DB 或 DI
OS ---| |---	OS ---| |---	状态位指令	存储溢出异常位
OV ---| |---	OV ---| |---	状态位指令	溢出异常位
---(P)---	---(P)---	位逻辑指令	RLO 上升沿检测
POS	POS	位逻辑指令	地址上升沿检测
---(R)	---(R)	位逻辑指令	线圈复位
---(RET)	---(RET)	程序控制指令	返回
ROL_DW	ROL_DW	移位和循环指令	双字左循环
ROR_DW	ROR_DW	移位和循环指令	双字右循环
ROUND	ROUND	转换指令	舍入为双整数
RS	RS	位逻辑指令	复位置位触发器
---(S)	---(S)	位逻辑指令	线圈置位
---(SA)	---(SF)	定时器指令	断开延时定时器线圈
---(SAVE)	---(SAVE)	位逻辑指令	将 RLO 存入 BR 存储器

英文助记符	德文助记符	程序元素分类	说明
S_AVERZ	S_OFFDT	定时器指令	断电延时 S5 定时器
---(SE)	---(SD)	定时器指令	扩展接通定时器线圈
S_EVERZ	S_ODT	定时器指令	接通延时 S5 定时器
SHL_DW	SHL_DW	移位和循环指令	双字左移
SHL_W	SHL_W	移位和循环指令	字左移
SHR_DI	SHR_DI	移位和循环指令	双整数右移
SHR_DW	SHR_DW	移位和循环指令	双字右移
SHR_I	SHR_I	移位和循环指令	整数右移
SHR_W	SHR_W	移位和循环指令	字右移
---(SI)	---(SP)	定时器指令	脉冲定时器线圈
S_IMPULS	S_PULSE	定时器指令	脉冲 S5 定时器
SIN	SIN	浮点算术运算指令	浮点数正弦运算
SQR	SQR	浮点算术运算指令	浮点数平方
SQRT	SQRT	浮点算术运算指令	浮点数平方根
SR	SR	位逻辑指令	置位复位触发器
---(SS)	---(SS)	定时器指令	保持型接通延时定时器线圈
S_SEVERZ	S_ODTS	定时器指令	保持型接通延时 S5 定时器
SUB_DI	SUB_DI	整数算术运算指令	双整数减法
SUB_I	SUB_I	整数算术运算指令	整数减法
SUB_R	SUB_R	浮点算术运算指令	实数减法
---(SV)	---(SE)	定时器指令	扩展脉冲定时器线圈
S_VIMP	S_PEXT	定时器指令	扩展脉冲 S5 定时器
---(SZ)	---(SC)	计数器指令	设置计数器值
TAN	TAN	浮点算术运算指令	浮点数正切运算
TRUNC	TRUNC	转换指令	舍去小数取整为双整数
UO ---||---	UO ---||---	状态位指令	无序异常位
WAND_DW	WAND_DW	字逻辑指令	双字和双字相“与”
WAND_W	WAND_W	字逻辑指令	字和字相“与”
WOR_DW	WOR_DW	字逻辑指令	双字和双字相“或”
WOR_W	WOR_W	字逻辑指令	字和字相“或”
WXOR_DW	WXOR_DW	字逻辑指令	双字和双字相“异或”
WXOR_W	WXOR_W	字逻辑指令	字和字相“异或”
ZAEHLER	S_CUD	计数器指令	加-减计数器
----(ZR)	----(CD)	计数器指令	减计数器线圈
Z_RUECK	S_CD	计数器指令	减计数器
---(ZV)	----(CU)	计数器指令	加计数器线圈
Z_VORW	S_CU	计数器指令	加计数器

B 编程举例

B.1 编程举例概述

实际应用

本手册中描述的每条梯形逻辑指令都触发一个专门的操作。当你将这些指令组合成程序时，就能够完成很多种类的自动化任务。本章提供梯形逻辑指令实际应用的如下例子：

- **使用位逻辑指令控制传送带**
- **使用位逻辑指令检测传送带的运动方向**
- **使用定时器指令生成时钟脉冲**
- **使用计数器和比较指令监视存储空间**
- **使用整数算术运算指令解题**
- **设定加热炉的加热时间**

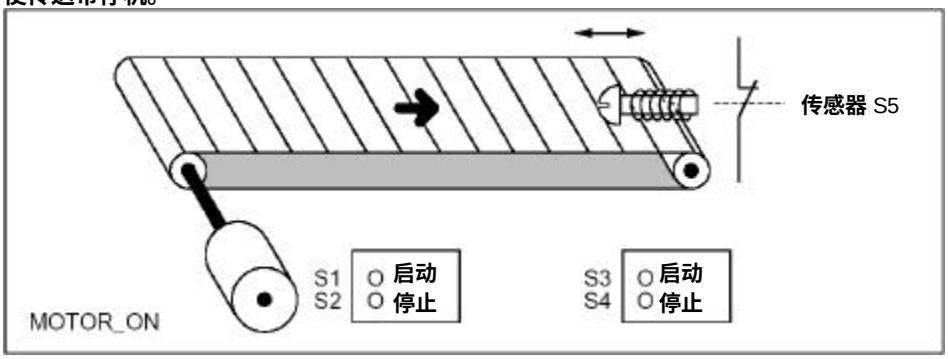
使用的指令

助记符	程序元素分类	说明
WAND_W	字逻辑指令	字和字相“与”
WOR_W	字逻辑指令	字和字相“或”
---(CD)	计数器指令	减计数器线圈
---(CU)	计数器指令	加计数器线圈
---(R)	位逻辑指令	线圈复位
---(S)	位逻辑指令	线圈置位
---(P)	位逻辑指令	RLO 上升沿检测
ADD_I	浮点算术运算指令	整数加法
DIV_I	浮点算术运算指令	整数除法
MUL_I	浮点算术运算指令	整数乘法
CMP <=I, CMP >=I	比较指令	整数比较
--| |--	位逻辑指令	常开接点
--| / |--	位逻辑指令	常闭接点
--( )	位逻辑指令	输出线圈
---(JMPN)	跳转指令	若非则跳转
---(RET)	程序控制指令	返回
MOVE	赋值指令	赋值
---(SE)	定时器指令	扩展脉冲定时器线圈

B.2 举例：位逻辑指令

举例 1：控制传送带

下图所示为一个能够电气启动的传送带。在传送带的起点有两个按钮开关：用于 START 的 S1 和用于 STOP 的 S2。在传送带的尾端也有两个按钮开关：用于 START 的 S3 和用于 STOP 的 S4。可以从任一端起动或停止传送带。另外，当传送带上的物件到达末端时，传感器 S5 使传送带停机。



绝对编程和符号编程

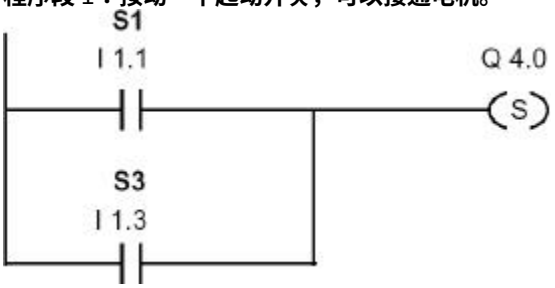
你可以使用代表传送带系统不同部件的绝对值或符号编写传送带控制程序。

你需要作一个符号表，使选择的符号与绝对值相对应（参见“ STEP 7 在线帮助 ”）。

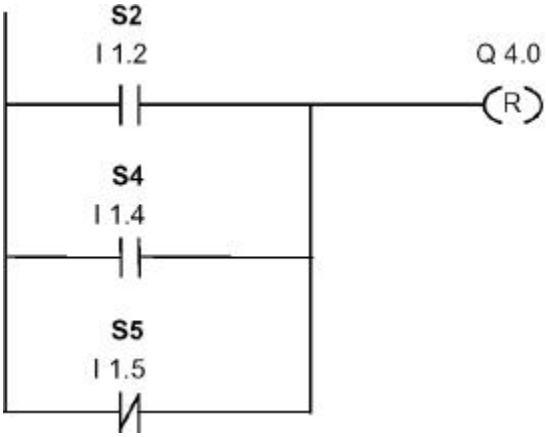
系统部件	绝对地址	符号	符号表
起动按钮开关	I 1.1	S1	I 1.1 S1
停止按钮开关	I 1.2	S2	I 1.2 S2
起动按钮开关	I 1.3	S3	I 1.3 S3
停止按钮开关	I 1.4	S4	I 1.4 S4
传感器	I 1.5	S5	I 1.5 S5
电机	Q 4.0	MOTOR_ON	Q 4.0 MOTOR_ON

控制传送带的梯形逻辑程序

程序段 1：按动一个起动开关，可以接通电机。

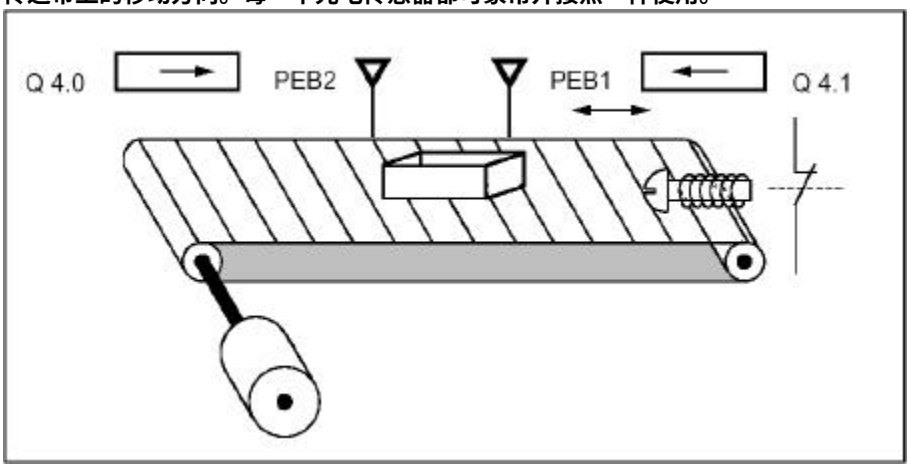


程序段 2：按动一个停止开关或打开传送带端部的常闭接点，可以切断电机。



举例 2：检测传送带的运动方向

下图所示为一个装配有两个光电传感器（PEB1 和 PEB2）的传送带，设计用于检测包裹在传送带上的移动方向。每一个光电传感器都可象常开接点一样使用。



绝对编程和符号编程

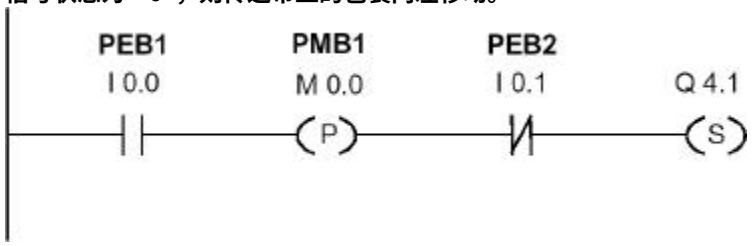
你可以使用代表传送带系统不同部件的绝对值或符号编写传送带系统方向显示功能启动程序。

你需要作一个符号表，使选择的符号与绝对值相对应（参见“ STEP 7 在线帮助 ”）。

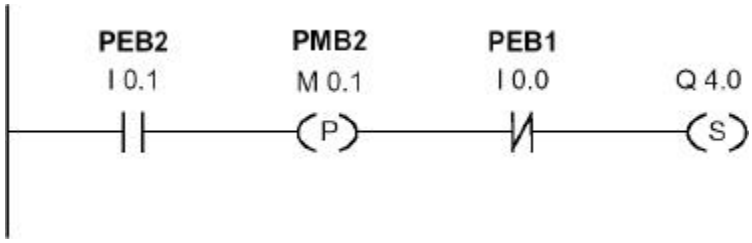
系统部件	绝对地址	符号	符号表
光电传感器 1	I 0.0	PEB1	I 0.0 PEB1
光电传感器 2	I .1	PEB2	I 0.1 PEB2
向右运动显示	Q 4.0	RIGHT	Q 4.0 RIGHT
向左运动显示	Q 4.1	LEFT	Q 4.1 LEFT
脉冲存储位 1	M .0	PMB1	M 0.0 PMB1
脉冲存储位 2	M .1	PMB2	M 0.1 PMB2

检测传送带运动方向的梯形逻辑程序

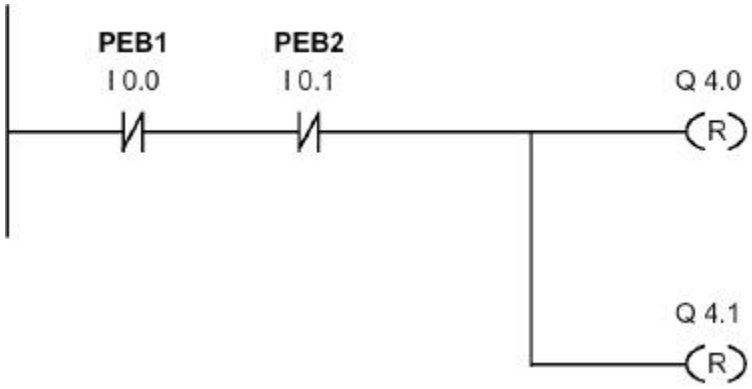
程序段 1：如果在输入 I0.0 上出现信号状态从“ 0 ”变为“ 1 ”（上升沿），同时输入 I0.1 的信号状态为“ 0 ”，则传送带上的包裹向左移动。



程序段 2：如果在输入 I0.1 上出现信号状态从“ 0 ”变为“ 1 ”（上升沿），同时输入 I0.0 的信号状态为“ 0 ”，则传送带上的包裹向右移动。如果有一个光电传感器被遮挡，这就意味着在光电传感器间有一个包裹。



程序段 3：如果没有一个光电传感器被遮挡，则在光电传感器之间没有包裹。方向指示灯熄灭。



B.3 举例：定时器指令

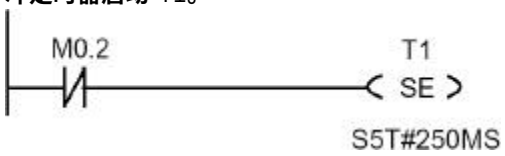
时钟脉冲发生器

当需要产生周期重复的信号时，可以使用时钟脉冲发生器或闪烁继电器。时钟脉冲发生器在信号系统是公用的，它可控制指示灯的闪烁。

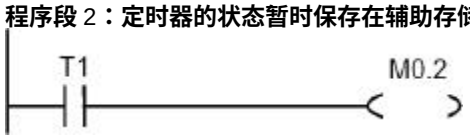
当使用 S7-300 **时，能够使用专用组织块中的时间驱动处理来实现时钟脉冲发生器功能。下面以梯形逻辑程序表示的例子，来说明使用定时器功能来产生时钟脉冲。示例程序表示如何使用定时器实现自由设定时钟脉冲发生器功能。**

产生时钟脉冲的梯形逻辑程序（脉冲占空系数 1:1）

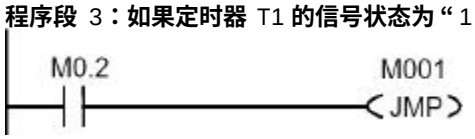
程序段 1：**如果定时器** T1 **的信号状态为“** 0 **”，则将时间值** 250 ms **装入** T1、**并按扩展脉冲定时器启动** T1。



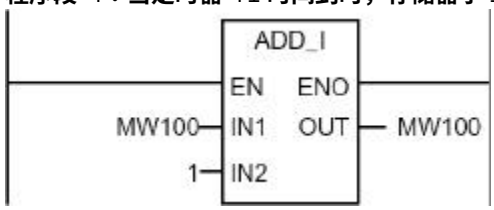
程序段 2：**定时器的状态暂时保存在辅助存储器标志中。**



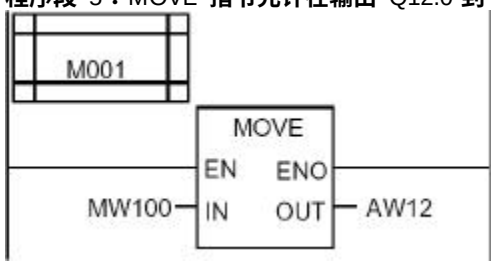
程序段 3：**如果定时器** T1 **的信号状态为“** 1 **”，则跳转到标号** M001。



程序段 4：**当定时器** T1 **时间到时，存储器字** 100 **加“** 1 **”。**

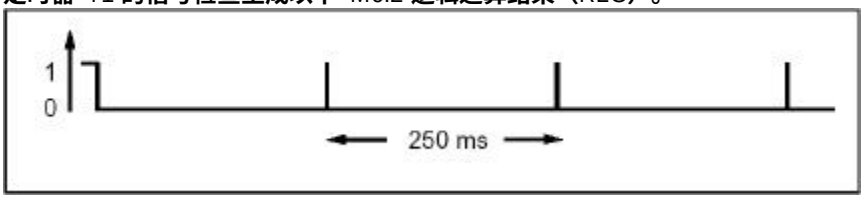


程序段 5：MOVE **指令允许在输出** Q12.0 **到** Q13.7 **获得各种不同的时钟频率。**



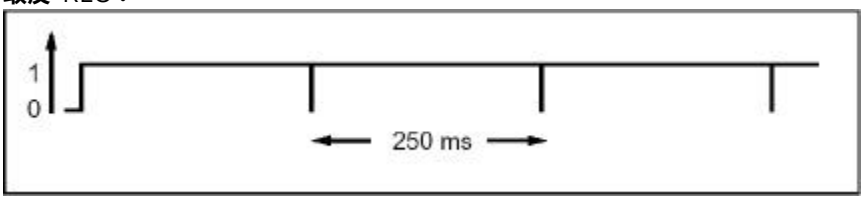
信号检查

定时器 T1 **的信号检查生成以下** M0.2 **逻辑运算结果（**RLO**）。**



只要时间一结束，定时器就重新启动。因此，由 --| / |-- M0.2 **进行的信号检查只短暂地生成信号状态“** 1 **”。**

取反 RLO**：**



每隔 250 ms**，**RLO **位就出现一次“** 0 **”。不跳转，存储器字** MW100 **的内容加“** 1 **”。**

获得专用频率

从存储器字节 MB101 **和** MB100 **的每个位，可以获得以下频率：**

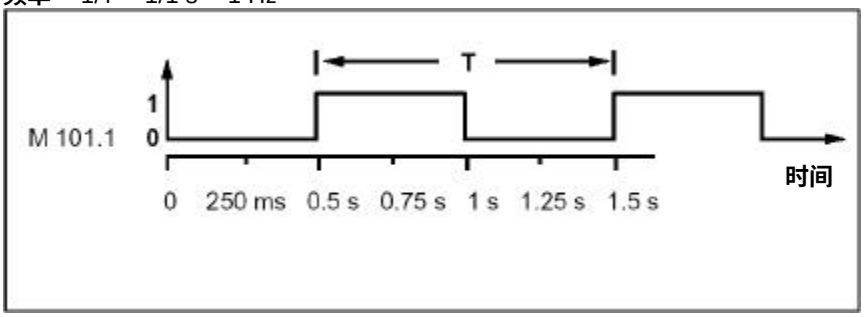
MB101/MB100 **位**	**频率，**[Hz]	**持续时间**	
M 101.0	2.0	0.5 s	(250 ms **通** / 250 ms **断**)
M 101.1	1.0	1s	(0.5 s **通** / 0.5 s **断**)
M 101.2	0.5	2s	(1 s **通** / 1 s **断**)
M 101.3	0.25	4s	(2 s **通** / 2 s **断**)
M 101.4	0.125	8s	(4 s **通** / 4 s **断**)
M 101.5	0.0625	16s	(8 s **通** / 8 s **断**)
M 101.6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

存储器字节 MB 101 **的各位信号状态**

扫描周期	**位** 7	**位** 6	**位** 5	**位** 4	**位** 3	**位** 2	**位** 1	**位** 0	**时间值，** [ms]
0	0	0	0	0	0	0	0	0	250
1	0	0	0	0	0	0	0	1	250
2	0	0	0	0	0	0	1	0	250
3	0	0	0	0	0	0	1	1	250
4	0	0	0	0	0	1	0	0	250
5	0	0	0	0	0	1	0	1	250
6	0	0	0	0	0	1	1	0	250
7	0	0	0	0	0	1	1	1	250
8	0	0	0	0	1	0	0	0	250
9	0	0	0	0	1	0	0	1	250
10	0	0	0	0	1	0	1	0	250
11	0	0	0	0	1	0	1	1	250
12	0	0	0	0	1	1	0	0	250

MB 101 **位** 1（M 101.1）**的信号状态**

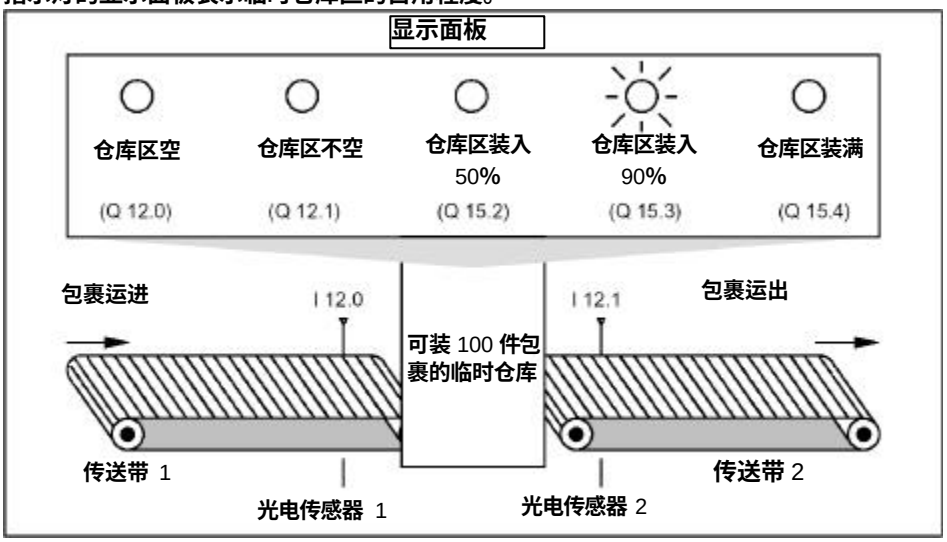
频率 = 1/T = 1/1 s = 1 Hz



B.4 举例：计数器和比较指令

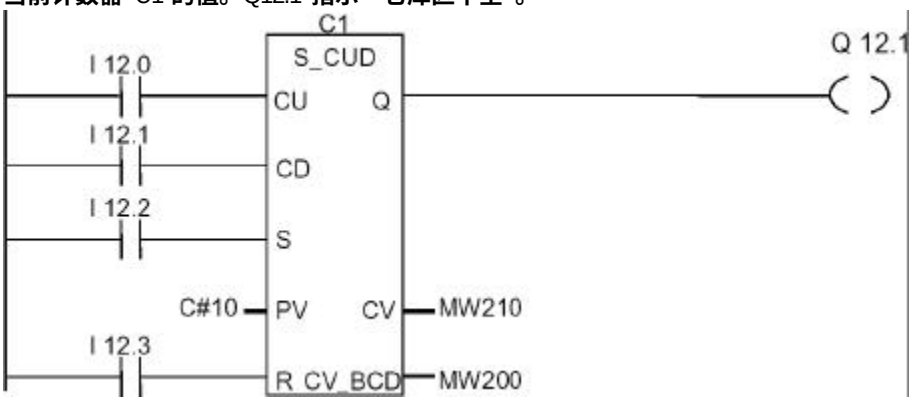
装有计数器和比较器的仓库区

下图所示为包括两台传送带的系统，在两台传送带之间有一个临时仓库区。传送带 1 将包裹运送至仓库区。传送带 1 靠近仓库区一端安装的光电传感器确定已有多少包裹运送至仓库区。传送带 2 将临时库区中的包裹运送至装货场，在这里货物由卡车运送至顾客。传送带 2 靠近仓库区一端安装的光电传感器确定已有多少包裹从仓库区运送至装货场。含 5 个指示灯的显示面板表示临时仓库区的占用程度。



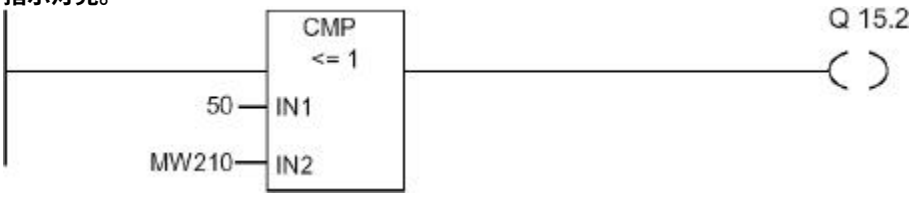
启动显示面板上指示灯的梯形逻辑程序

程序段 1：输入 CU 端的信号每次从“ 0 ”变为“ 1 ”时，计数器 C1 加 1，输入 CD 端的信号每次从“ 0 ”变为“ 1 ”时，计数器 C1 减 1。输入 S 端的信号从“ 0 ”变为“ 1 ”时，计数器值置为 PV。输入 R 端的信号从“ 0 ”变为“ 1 ”时，计数器值清零。MW200 中保存当前计数器 C1 的值。Q12.1 指示“ 仓库区不空 ”。



程序段 2:Q12.0 **指示“仓库区空”。**

程序段 3:**如果** 50 **小于等于计数器值(即如果计数器值大于等于** 50),**“仓库区装入** 50% **”指示灯亮。**



程序段 4:**如果计数器值大于等于** 90,**“仓库区装入** 90% **”指示灯亮。**

程序段 5：**如果计数器值大于等于** 100,**“仓库区装满”指示灯亮。**

B.5 举例：整数算术运算指令

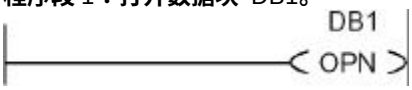
解决算术问题

以下示例程序告诉你如何使用 3 种整数算术运算指令产生如下列方程一样的结果：

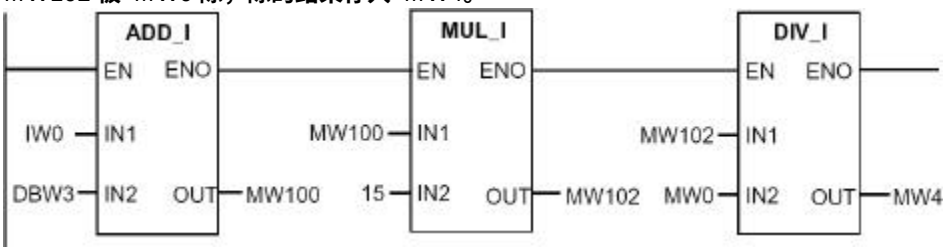
MW4 = ((IW0 + DBW3) x 15) / MW0

梯形逻辑程序

程序段 1：打开数据块 DB1。



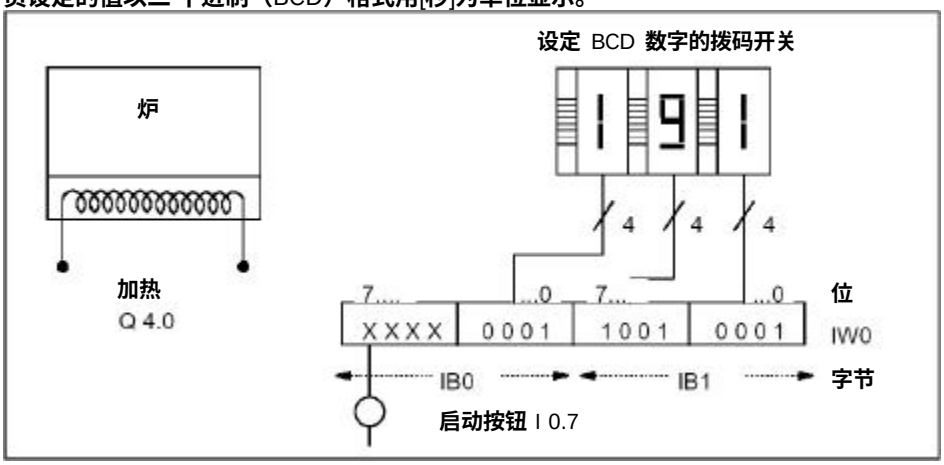
程序段 2：输入字 IW0 **与共享数据字** DBW3 **相加（数据块必须已定义好并打开），相加之和装入存储器字** MW100。**然后，**MW100 **与** 15 **相乘，相乘的结果存储器字** MW102。MW102 **被** MW0 **除，除的结果存入** MW4。



B.6 举例：字逻辑指令

加热炉

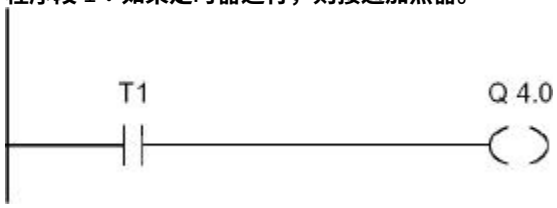
操作员按启动按钮开始加热炉。操作员能够使用如图所示的拨码开关设定加热时间。操作员设定的值以二-十进制（BCD**）格式用[秒]为单位显示。**



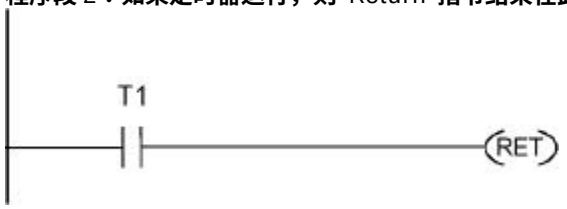
系统部件	绝对地址
启动按钮	I .7
个位数拨码开关	I 1.0 到 I 1.3
十位数拨码开关	I 1.4 到 I 1.7
百位数拨码开关	I 0.0 到 I 0.3
开始加热	Q 4.0

梯形逻辑程序

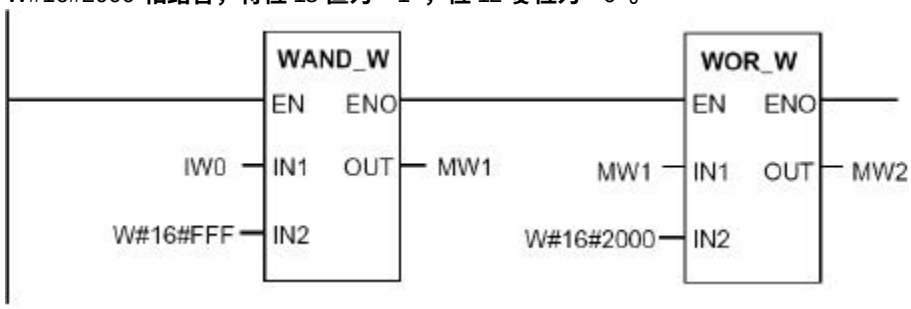
程序段 1：如果定时器运行，则接通加热器。



程序段 2：如果定时器运行，则 Return 指令结束在此的处理。



程序段 3：屏蔽输入位 I 0.4 至 I 0.7（即将其复位为“0”）。不使用拨码开关输入这些位。按照字和字相“与”指令，将拨码开关输入的 16 位与 W#16#0FFF 相结合。其结果装入存储器字 MW1 中。为了设定以[秒]为单位的时基，按照字和字相“或”指令将预置值和 W#16#2000 相结合，将位 13 置为“1”，位 12 复位为“0”。



程序段 4：如果按动启动按钮，则启动定时器 T1 为扩展脉冲定时器，以存储器字 MW2 装入（取自上述逻辑）预置时间值。