

## SIMATIC

## S7-300 和 S7-400

## 编程语句表 (STL)

## 参考手册

2004 年 1 月版

### 前言，目录

位逻辑指令 1

比较指令 2

转换指令 3

计数器指令 4

数据块指令 5

逻辑控制指令 6

整数算术运算指令 7

浮点算术运算指令 8

装入和传送指令 9

程序控制指令 10

移位和循环移位指令 11

定时器指令 12

字逻辑指令 13

累加器操作指令 14

### 附录

所有语句表指令一览 A

编程举例 B

### 索引

## 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。

西门子公司版权所有©2004。保留所有权利。

未经明确的书面授权，禁止复制、传递或使用本手册或其中的内容。

违者必究。保留所有权利包括专利权、实用新型或外观设计专利权。

### 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。



# 前言

## 目的

本使用手册旨在提供指南，以使用语句表编程语言（STL）编制用户程序。  
本手册中还包含一个参考章节，阐述了 STL 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

STL 符合国际电工委员会标准 IEC 1131-3 中定义的“语句表”编程语言，然而考虑到操作仍有本质区别。关于详细信息，请参考 STEP 7 文件 NORM\_TBL.WRI 中的标准列表。

## 要求

为了有效地使用这本语句表手册，你要预先熟悉 STEP 7 在线帮助资料中 S7 编程理论。语言包也使用 STEP 7 标准软件，所以你要熟练使用这个软件并阅读相关的资料。

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：

资 料	用 途	订 货 号
STEP 7 基本信息 <ul style="list-style-type: none"><li>STEP 7 V5.3，《快速入门手册》</li><li>STEP 7 V5.3 编程</li><li>配置硬件和通讯连接，STEP 7 V5.3</li><li>《从 S5 到 S7 转换手册》</li></ul>	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 <ul style="list-style-type: none"><li>《S7-300/400 梯形逻辑 (LAD)/功能块图 (FBD)/语句表 (STL) 使用手册》</li><li>S7-300/400 标准和系统功能手册</li></ul>	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。	6ES7810-4CA06-8BW1

在线帮助	用 途	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
STL/LAD/FBD 参考帮助 系统功能块 / 系统功能 (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“语句表中的帮助信息”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>

## 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：

电话：+49 (911) 895-3200.

网址：<http://www.sitrain.com>

<http://www.ad.siemens.com.cn/training>

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

# A&D 技术支持

遍布全球，24小时服务：



<p>总部（纽伦堡）</p> <p>技术支持</p> <p>一天 24 小时，一年 365 天全天候服务</p> <p>电话：+49 (0) 180 5050-222</p> <p>传真：+49 (0) 180 5050-223</p> <p>E-Mail: adsupport@siemens.com</p> <p>GMT： +1:00</p>		
<p>欧洲/非洲（纽伦堡）</p> <p>授权</p> <p>当地时间：星期一到星期五 08:00:00 - 17:00</p> <p>电话：+49 (0) 180 5050-222</p> <p>传真：+49 (0) 180 5050-223</p> <p>E-Mail: adsupport@siemens.com</p> <p>GMT： +1:00</p>	<p>美国（约翰森城）</p> <p>技术支持和授权</p> <p>当地时间：星期一到星期五 08:00:00 - 17:00</p> <p>电话：+1 (0) 770 740 3505</p> <p>传真：+1 (0) 770 740 3699</p> <p>E-Mail: isd-callcenter@sea.siemens.com</p> <p>GMT： -5:00</p>	<p>亚洲/澳大利亚（北京）</p> <p>技术支持和授权</p> <p>当地时间：星期一到星期五 8:30 - 17:30</p> <p>电话：+86 10 64 75 75 75</p> <p>传真：+86 10 64 74 74 74</p> <p>E-Mail: adsupport.asia@siemens.com</p> <p>GMT： +8:00</p>
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> (英文网站)

<http://www.ad.siemens.com.cn/service> (中文网站)

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”。

北 京：(010) 6471 9990

大 连：(0411) 369 9760 - 40

上 海：(021) 5879 5255

广 州：(020) 8732 3967

成 都：(028) 6820 0939





# 目录

前言.....	iii
目录.....	ix
1 位逻辑指令.....	1-1
1.1 位逻辑指令概述 .....	1-1
1.2 A “与” .....	1-3
1.3 AN “与非” .....	1-4
1.4 O “或” .....	1-5
1.5 ON “或非” .....	1-6
1.6 X “异或” .....	1-7
1.7 XN “异或非” .....	1-8
1.8 O 先“与”后“或” .....	1-9
1.9 A( “与”操作嵌套开始 .....	1-10
1.10 AN( “与非”操作嵌套开始 .....	1-11
1.11 O( “或”操作嵌套开始 .....	1-11
1.12 ON( “或非”操作嵌套开始 .....	1-12
1.13 X( “异或”操作嵌套开始 .....	1-12
1.14 XN( “异或非”操作嵌套开始 .....	1-13
1.15 ) 嵌套闭合 .....	1-14
1.16 = 赋值 .....	1-15
1.17 R 复位 .....	1-16
1.18 S 置位 .....	1-17
1.19 NOT RLO 取反 .....	1-18
1.20 SET RLO 置位 (=1) .....	1-18
1.21 CLR RLO 清零 (=0) .....	1-19
1.22 SAVE 把 RLO 存入 BR 寄存器 .....	1-20
1.23 FN 下降沿 .....	1-21
1.24 FP 上升沿 .....	1-23
2 比较指令.....	2-1
2.1 比较指令概述 .....	2-1
2.2 ? I 比较两个整数 (16 位) .....	2-2
2.3 ? D 比较两个双整数 (32 位) .....	2-3
2.4 ? R 比较两个浮点数 (32 位) .....	2-4
3 转换指令.....	3-1
3.1 转换指令概述 .....	3-1
3.2 BTI BCD 转成整数 (16 位) .....	3-2
3.3 ITB 整数 (16 位) 转成 BCD .....	3-3
3.4 BTB BCD 转成整数 (32 位) .....	3-4

3.5	ITD 整数 (16 位) 转成双整数 (32 位)	3-5
3.6	DTB 双整数 (32 位) 转成 BCD	3-6
3.7	DTR 双整数 (32 位) 转成浮点数 (32 位, IEEE-FP)	3-7
3.8	INVI 对整数求反码 (16 位)	3-8
3.9	INVD 对双整数求反码 (32 位)	3-9
3.10	NEGI 对整数求补码 (16 位)	3-10
3.11	NEGD 对双整数求补码 (32 位)	3-11
3.12	NEGR 对浮点数求反 (32 位, IEEE-FP)	3-12
3.13	CAW 交换累加器 1 低字中的字节顺序 (16 位)	3-13
3.14	CAD 交换累加器 1 中的字节顺序 (32 位)	3-14
3.15	RND 取整	3-15
3.16	TRUNC 截尾取整	3-16
3.17	RND+ 取整为较大的双整数	3-17
3.18	RND- 取整为较小的双整数	3-18
4	计数器指令	4-1
4.1	计数器指令概述	4-1
4.2	FR 使能计数器 (任意)	4-2
4.3	L 将当前计数器值装入累加器 1	4-3
4.4	LC 将当前计数器值作为 BCD 码装入累加器 1	4-4
4.5	R 复位计数器	4-5
4.6	S 计数器置位	4-6
4.7	CU 加计数器	4-7
4.8	CD 减计数器	4-8
5	数据块指令	5-1
5.1	数据块指令概述	5-1
5.2	OPN 打开数据块	5-2
5.3	CDB 交换共享数据块和背景数据块	5-3
5.4	L DBLG 将共享数据块的长度装入累加器 1 中	5-3
5.5	L DBNO 将共享数据块的块号装入累加器 1 中	5-4
5.6	L DILG 将背景数据块的长度装入累加器 1 中	5-4
5.7	L DINO 将背景数据块的块号装入累加器 1 中	5-5
6	逻辑控制指令	6-1
6.1	逻辑控制指令概述	6-1
6.2	JU 无条件跳转	6-3
6.3	JL 跳转到标号	6-4
6.4	JC 若 RLO = 1, 则跳转	6-5
6.5	JCN 若 RLO = 0, 则跳转	6-6
6.6	JCB 若 RLO = 1, 则连同 BR 一起跳转	6-7
6.7	JNB 若 RLO = 0, 则连同 BR 一起跳转	6-8
6.8	JB 若 BR = 1, 则跳转	6-9
6.9	JNBI 若 BR = 0, 则跳转	6-10
6.10	JO 若 OV = 1, 则跳转	6-11

6.11	JOS 若 OS = 1, 则跳转 .....	6-12
6.12	JZ 若零, 则跳转 .....	6-13
6.13	JN 若非零, 则跳转 .....	6-14
6.14	JP 若正, 则跳转 .....	6-15
6.15	JM 若负, 则跳转 .....	6-16
6.16	JPZ 若正或零, 则跳转 .....	6-17
6.17	JMZ 若负或零, 则跳转 .....	6-18
6.18	JUO 若无效数, 则跳转 .....	6-19
6.19	LOOP 循环控制 .....	6-20
7	整数算术运算指令 .....	7-1
7.1	整数算术运算指令概述 .....	7-1
7.2	判断整数算术运算指令后状态字的位 .....	7-2
7.3	+I 作为整数(16 位), 将累加器 1 和累加器 2 中的内容相加 .....	7-3
7.4	-I 作为整数(16 位), 将累加器 2 的内容减累加器 1 的内容 .....	7-4
7.5	*I 作为整数(16 位), 将累加器 1 和累加器 2 中的内容相乘 .....	7-5
7.6	/I 作为整数(16 位), 将累加器 2 的内容除以累加器 1 的内容 .....	7-6
7.7	+ 加上一个整数常数 ( 16 位, 32 位 ) .....	7-7
7.8	+D 作为双整数(32 位), 将累加器 1 和累加器 2 的内容相加 .....	7-9
7.9	-D 作为双整数(32 位), 累加器 2 的内容减累加器 1 的内容 .....	7-10
7.10	*D 作为双整数(32 位), 将累加器 1 和累加器 2 的内容相乘 .....	7-11
7.11	/D 作为双整数(32 位), 累加器 2 的内容除以累加器 1 的内容 .....	7-12
7.12	MOD 双整数除法的余数 ( 32 位 ) .....	7-13
8	浮点算术运算指令 .....	8-1
8.1	浮点算术运算指令概述 .....	8-1
8.2	判断浮点算术运算指令后状态字的位 .....	8-2
8.3	浮点算术运算指令: 基本指令 .....	8-3
8.3.1	+R 作为浮点数(32 位, IEEE-FP), 将累加器 1 和累加器 2 中的内容相加 .....	8-3
8.3.2	-R 作为浮点数(32 位, IEEE-FP), 将累加器 2 中的内容减去累加器 1 中的内容 .....	8-4
8.3.3	*R 作为浮点数(32 位, IEEE-FP), 将累加器 1 和累加器 2 中的内容相乘 .....	8-5
8.3.4	/R 作为浮点数(32 位, IEEE-FP), 累加器 2 的内容除以累加器 1 的内容 .....	8-6
8.3.5	ABS 浮点数取绝对值 ( 32 位, IEEE-FP ) .....	8-7
8.4	浮点算术运算指令: 扩展指令 .....	8-8
8.4.1	SQR 浮点数平方运算 ( 32 位 ) .....	8-8
8.4.2	SQRT 浮点数开方运算 ( 32 位 ) .....	8-9
8.4.3	EXP 浮点数指数运算 ( 32 位 ) .....	8-10
8.4.4	LN 浮点数自然对数运算 ( 32 位 ) .....	8-11
8.4.5	SIN 浮点数正弦运算 ( 32 位 ) .....	8-12
8.4.6	COS 浮点数余弦运算 ( 32 位 ) .....	8-13
8.4.7	TAN 浮点数正切运算 ( 32 位 ) .....	8-14
8.4.8	ASIN 浮点数反正弦运算 ( 32 位 ) .....	8-15
8.4.9	ACOS 浮点数反余弦运算 ( 32 位 ) .....	8-16
8.4.10	ATAN 浮点数反正切运算 ( 32 位 ) .....	8-17

9	装入和传送指令	9-1
9.1	装入和传送指令概述	9-1
9.2	L 装入	9-2
9.3	L STW 将状态字装入累加器 1	9-3
9.4	LAR1 将累加器 1 中的内容装入地址寄存器 1	9-4
9.5	LAR1 <D> 将两个双整数(32 位指针)装入地址寄存器 1	9-5
9.6	LAR1 AR2 将地址寄存器 2 的内容装入地址寄存器 1	9-6
9.7	LAR2 将累加器 1 中的内容装入地址寄存器 2	9-6
9.8	LAR2 <D> 将两个双整数(32 位指针)装入地址寄存器 2	9-7
9.9	T 传送	9-8
9.10	T STW 将累加器 1 中的内容传送到状态字	9-9
9.11	CAR 交换地址寄存器 1 和地址寄存器 2 的内容	9-10
9.12	TAR1 将地址寄存器 1 中的内容传送到累加器 1	9-10
9.13	TAR1 <D> 将地址寄存器 1 的内容传送到目的地(32 位指针)	9-11
9.14	TAR1 AR2 将地址寄存器 1 的内容传送到地址寄存器 2	9-12
9.15	TAR2 将地址寄存器 2 中的内容传送到累加器 1	9-12
9.16	TAR2 <D> 将地址寄存器 2 的内容传送到目的地(32 位指针)	9-13
10	程序控制指令	10-1
10.1	程序控制指令概述	10-1
10.2	BE 块结束	10-2
10.3	BEC 条件块结束	10-3
10.4	BEU 无条件块结束	10-4
10.5	CALL 块调用	10-5
10.6	调用功能块	10-8
10.7	调用功能	10-10
10.8	调用系统功能块	10-12
10.9	调用系统功能	10-14
10.10	调用多背景块	10-15
10.11	从库中调用块	10-15
10.12	CC 条件调用	10-16
10.13	UC 无条件调用	10-17
10.14	MCR (主控继电器)	10-18
10.15	使用 MCR 功能的重要注意事项	10-20
10.16	MCR( 将 RLO 存入 MCR 堆栈, 开始 MCR	10-21
10.17	)MCR 结束 MCR	10-23
10.18	MCRA 激活 MCR 区域	10-24
10.19	MCRD 去活 MCR 区域	10-25
11	移位和循环移位指令	11-1
11.1	移位指令	11-1
11.1.1	移位指令概述	11-1
11.1.2	SSI 移位有符号整数 ( 16 位 )	11-2
11.1.3	SSD 移位有符号双整数 ( 32 位 )	11-3

11.1.4	SLW 字左移 (16 位)	11-5
11.1.5	SRW 字右移 (16 位)	11-6
11.1.6	SLD 双字左移 (32 位)	11-7
11.1.7	SRD 双字右移 (32 位)	11-8
11.2	循环移位指令	11-10
11.2.1	循环移位指令概述	11-10
11.2.2	RLD 双字循环左移 (32 位)	11-10
11.2.3	RRD 双字循环右移 (32 位)	11-12
11.2.4	RLDA 通过 CC 1 累加器 1 循环左移 (32 位)	11-13
11.2.5	RRDA 通过 CC 1 累加器 1 循环右移 (32 位)	11-14
12	定时器指令	12-1
12.1	定时器指令概述	12-1
12.2	存储区中定时器的存储单元和定时器的组成部分	12-2
12.3	FR 使能定时器 (任意)	12-5
12.4	L 将当前定时值作为整数装入累加器 1	12-7
12.5	LC 将当前定时器值作为 BCD 码装入累加器 1	12-8
12.6	R 复位定时器	12-9
12.7	SP 脉冲定时器	12-10
12.8	SE 延时脉冲定时器	12-11
12.9	SD 延时接通定时器	12-13
12.10	SS 保持型延时接通定时器	12-14
12.11	SF 延时断开定时器	12-16
13	字逻辑指令	13-1
13.1	字逻辑指令概述	13-1
13.2	AW 字“与”(16 位)	13-2
13.3	OW 字“或”(16 位)	13-3
13.4	XOW 字“异或”(16 位)	13-4
13.5	AD 双字“与”(32 位)	13-6
13.6	OD 双字“或”(32 位)	13-7
13.7	XOD 双字“异或”(32 位)	13-8
14	累加器操作指令	14-1
14.1	累加器和地址寄存器操作指令概述	14-1
14.2	TAK 累加器 1 与累加器 2 进行互换	14-2
14.3	POP 带有两个累加器的 CPU	14-3
14.4	POP 带有四个累加器的 CPU	14-4
14.5	PUSH 带有两个累加器的 CPU	14-5
14.6	PUSH 带有四个累加器的 CPU	14-6
14.7	ENT 进入累加器栈	14-7
14.8	LEAVE 离开累加器栈	14-7
14.9	INC 增加累加器 1 低字的低字节	14-8
14.10	DEC 减少累加器 1 低字的低字节	14-9
14.11	+AR1 加累加器 1 至地址寄存器 1	14-10

14.12	+AR2 加累加器 1 至地址寄存器 2.....	14-11
14.13	BLD 程序显示指令（空）.....	14-12
14.14	NOP 0 空操作指令.....	14-13
14.15	NOP 1 空操作指令.....	14-13
A	所有语句表指令一览.....	A-1
A.1	按德文助记符分类的语句表指令.....	A-1
A.2	按英文助记符分类的语句表指令（国际）.....	A-6
B	编程举例.....	B-1
B.1	编程举例概述.....	B-1
B.2	例如：位逻辑指令.....	B-2
B.3	例如：定时器指令.....	B-5
B.4	例如：计数器和比较指令.....	B-8
B.5	例如：整数算术运算指令.....	B-10
B.6	例如：字逻辑指令.....	B-11

# 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。在触点与线圈领域，“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）”。

布尔位逻辑应用于以下基本指令：

- A “与”
- AN “与非”
- O “或”
- ON “或非”
- X “异或”
- XN “异或非”
- O “先与后或”

你可用以下指令执行嵌套表达式：

- A( “与”操作嵌套开始
- AN( “与非”操作嵌套开始
- O( “或”操作嵌套开始
- ON( “或非”操作嵌套开始
- X( “异或”操作嵌套开始
- XN( “异或非”操作嵌套开始
- ) 嵌套闭合



使用以下指令，可以结束一个布尔位逻辑串：

- =        赋值
- R        复位
- S        置位

你可以使用下述指令之一，更改逻辑运算的结果（RLO）：

- NOT     RLO 取反
- SET     RLO 置位（=1）
- CLR     RLO 清零（=0）
- SAVE    把 RLO 存入 BR 寄存器

其它指令对上升沿和下降沿有反应：

- FN       下降沿
- FP       上升沿

1.2 A “与”

格式

A <位>

地 址	数据类型	存储区
<位>	BOOL	I , Q , M , L , D , T , C

说明

使用“与”指令可以检查被寻址位的信号状态是否为“1”，并将检查结果与逻辑运算结果（RLO）进行“与”运算。

使用“与”指令，也可通过使用以下地址，直接检查状态字：==0，<>0，>0，<0，>=0，<=0，OV，OS，UO，BR。

状态字

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

举例

语句表程序	继电器逻辑图	
		
A I 1.0	I 1.0 信号状态“1”	常开触点
A I 1.1	I 1.1 信号状态“1”	常闭触点
= Q 4.0	Q 4.0 信号状态“1”	线圈
显示为闭合的开关		

1.3 AN “与非”

格式

N <位>

地 址	数据类型	存储区
<位>	BOOL	I , Q , M , L , D , T , C

说明

使用“与非”指令可以检查被寻址位的信号状态是否为“0”，并将检查结果与逻辑运算结果（RLO）进行“与”运算。

使用“与非”指令，也可通过使用以下地址，直接检查状态字：==0，<>0，>0，<0，>=0，<=0，OV，OS，UO，BR。

状态字

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

举例

语句表程序	继电器逻辑图	
A I 1.0	I 1.0 信号状态“0”	常开触点
AN I 1.1	I 1.1 信号状态“1”	常闭触点
= Q 4.0	Q 4.0 信号状态“0”	线圈

1.4 O “或”

格式

O <位>

地 址	数据类型	存储区
<位>	BOOL	I , Q , M , L , D , T , C

说明

使用“或”指令可以检查被寻址位的信号状态是否为“1”，并将检查结果与逻辑运算结果（RLO）进行“或”运算。

使用“或”指令，也可通过使用以下地址，直接检查状态字：==0，<>0，>0，<0，>=0，<=0，OV，OS，UO，BR。

状态字

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

举例

语句表程序	继电器逻辑图
<b>O I 1.0</b>	
<b>O I 1.1</b>	
<b>= Q 4.0</b>	

1.5 ON “或非”

格式

ON <位>

地 址	数据类型	存储区
<位>	BOOL	I , Q , M , L , D , T , C

说明

使用“或非”指令可以检查被寻址位的信号状态是否为“0”，并将检查结果与逻辑运算结果（RLO）进行“或”运算。

使用“或非”指令，也可通过使用以下地址，直接检查状态字：==0，<>0，>0，<0，>=0，<=0，OV，OS，UO，BR。

状态字

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

举例

语句表程序	继电器逻辑图
<b>O I 1.0</b>	
O I 1.1	
<b>= Q 4.0</b>	

1.6 X “异或”

格式

X <位>

地 址	数据类型	存储区
<位>	BOOL	I , Q , M , L , D , T , C

说明

使用“异或”指令可以检查被寻址位的信号状态是否为“1”，并将检查结果与逻辑运算结果（RLO）进行“异或”运算。

你也可以连续几次使用“异或”指令。如果有不成对被检地址的信号状态为“1”，则逻辑运算的相互结果为“1”。

使用“异或”指令，也可通过使用以下地址，直接检查状态字：==0，<>0，>0，<0，>=0，<=0，OV，OS，UO，BR。

状态字

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

举例

语句表程序	继电器逻辑图
	电力线
X I 1.0	触点 I 1.0
X I 1.1	触点 I 1.1
= Q 4.0	Q 4.0 线圈

1.7 XN “异或非”

格式

XN <位>

地 址	数据类型	存储区
<位>	BOOL	I , Q , M , L , D , T , C

说明

使用“异或非”指令可以检查被寻址位的信号状态是否为“0”，并将检查结果与逻辑运算结果（RLO）进行“异或”运算。

使用“异或非”指令，也可通过使用以下地址，直接检查状态字：==0，<>0，>0，<0，>=0，<=0，OV，OS，UO，BR。

状态字

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

举例

语句表程序			继电器逻辑图
			电力线
X	I 1.0	触点 I 1.0	
XN	I 1.1	触点 I 1.1	
=	Q 4.0	Q 4.0 线圈	

1.8     O   先“与”后“或”

格式

O

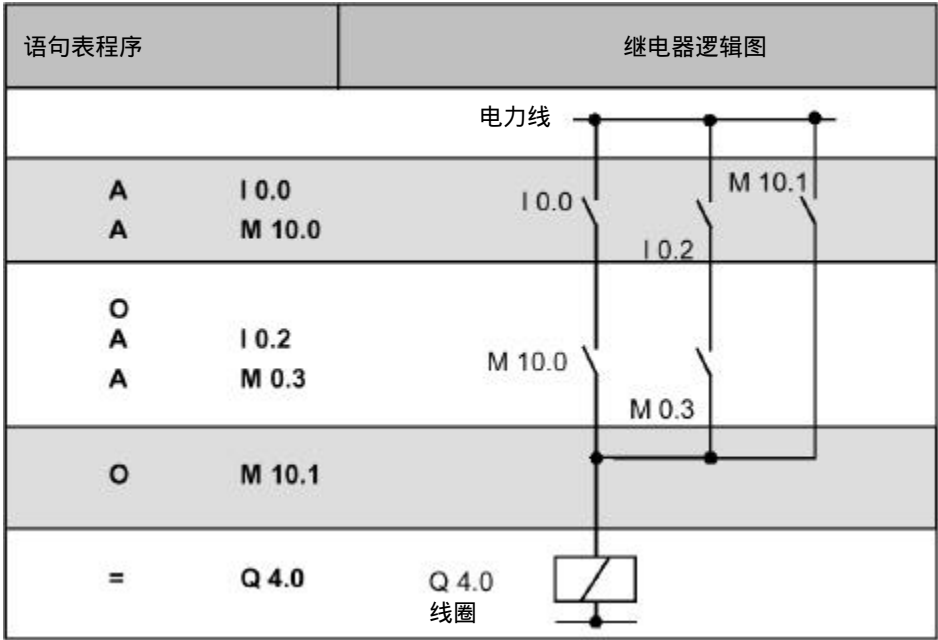
说明

“先与后或（O）”指令根据以下规则，对“与”运算执行逻辑“或”运算：在“OR（或）”之前“AND（与）”。

状态字

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

举例





1.9 A( “与”操作嵌套开始

格式

A(

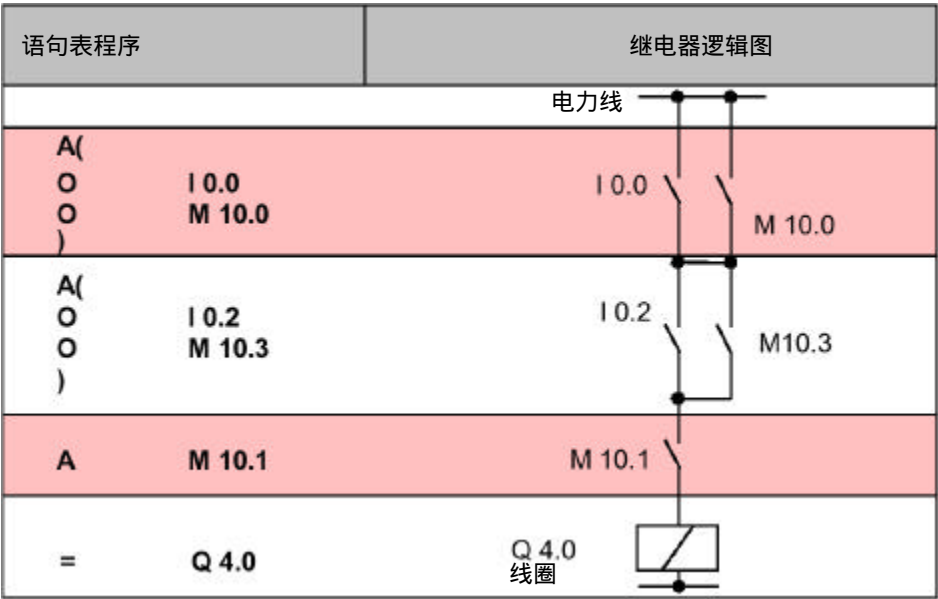
说明

A( (“与”操作嵌套开始) 可以将 RLO 和 OR 位以及一个指令代码保存在嵌套堆栈中。最多可有 7 个嵌套堆栈输入项。

状态字

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

举例



1.10 AN( “ 与非 ” 操作嵌套开始

格式

AN(

说明

AN( ( “ 与非 ” 操作嵌套开始 ) 可以将 RLO 和 OR 位 以及一个指令代码保存在嵌套堆栈中。最多可有 7 个嵌套堆栈输入项。

状态字

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

1.11 O( “ 或 ” 操作嵌套开始

格式

O(

说明

O( ( “ 或 ” 操作嵌套开始 ) 可以将 RLO 和 OR 位 以及一个指令代码保存在嵌套堆栈中。最多可有 7 个嵌套堆栈输入项。

状态字

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

## 1.12 ON( “或非”操作嵌套开始

格式

ON(

说明

ON( ( “或非”操作嵌套开始) 可以将 RLO 和 OR 位以及一个指令代码保存在嵌套堆栈中。最多可有 7 个嵌套堆栈输入项。

状态字

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

## 1.13 X( “异或”操作嵌套开始

格式

X(

说明

X( ( “异或”操作嵌套开始) 可以将 RLO 和 OR 位以及一个指令代码保存在嵌套堆栈中。最多可有 7 个嵌套堆栈输入项。

状态字

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

1.14 XN( “ 异或非 ” 操作嵌套开始

格式

XN(

说明

XN( ( “ 异或非 ” 操作嵌套开始 ) 可以将 RLO 和 OR 位 以及一个指令代码保存在嵌套堆栈中。最多可有 7 个嵌套堆栈输入项。

状态字

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

1.15    )   嵌套闭合

格式

)

说明

使用 ) ( 嵌套闭合 ) 指令，可以从嵌套堆栈中删除一个输入项，恢复 OR 位，根据指令代码，使堆栈输入项中所包含的 RLO 与当前 RLO 相关，以及赋值结果给 RLO。如果指令代码为 “ AND (与) ” 或 “ AND NOT (与非) ”，则也包括 OR 位。


使用括号组合的语句：

- U(       “ 与 ” 操作嵌套开始
- UN(      “ 与非 ” 操作嵌套开始
- O(       “ 或 ” 操作嵌套开始
- ON(      “ 或非 ” 操作嵌套开始
- X(       “ 异或 ” 操作嵌套开始
- XN(      “ 异或非 ” 操作嵌套开始

状态字

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

举例

语句表程序		继电器逻辑图		
		电力线		
A(	I 0.0	I 0.0	M 10.0	
O	M 10.0			
)				
A(	I 0.2	I 0.2	M 10.3	
O	M 10.3			
)				
A	M 10.1	M 10.1		
=	Q 4.0	Q 4.0 线圈		

1.16      =    赋值

格式

<位>

地 址	数据类型	存储区
<位>	BOOL	I , Q , M , L , D

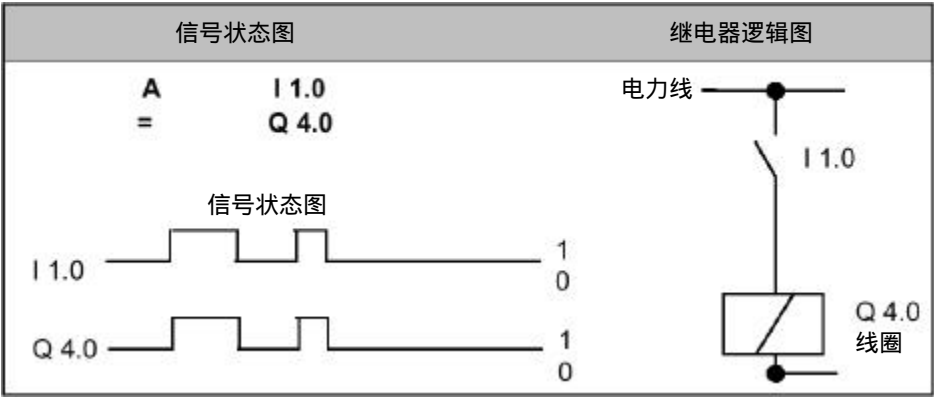
说明

如果 MCR = 1, 使用赋值指令 (= <位>), 可以将 RLO 写入寻址位, 以接通  
主控继电器。如果 MCR = 0, 则将数值 “0” 写入寻址位, 而不是 RLO。

状态字

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

举例



1.17 R 复位

格式

R <位>

地 址	数据类型	存储区
<位>	BOOL	I , Q , M , L , D

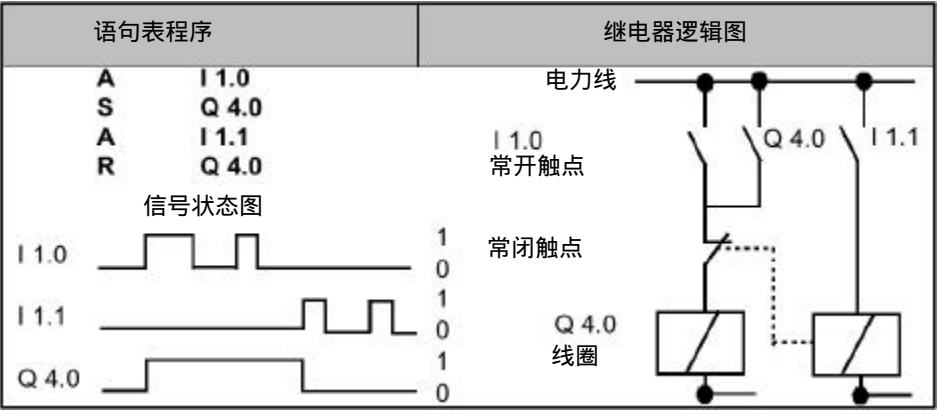
说明

如果 RLO = 1，并且主控继电器 MCR = 1，则使用复位指令（R），可以将寻址位复位为“0”。如果 MCR = 0，则寻址位没有改变。

状态字

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

举例



1.18 S 置位

格式

S <位>

地 址	数据类型	存储区
<位>	BOOL	I , Q , M , L , D

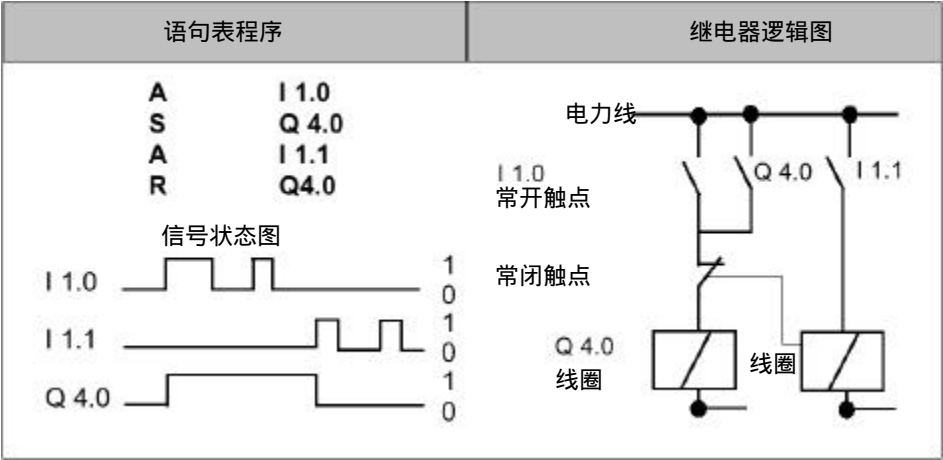
指令说明

如果 RLO = 1，并且主控继电器 MCR = 1，则使用置位指令（S），可以将寻址位置位为“1”。如果 MCR = 0，则寻址位没有改变。

状态字

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

举例





## 1.19 NOT RLO 取反

格式

NOT

说明

使用取反（NOT）指令，可以对 RLO 取反。

状态字

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

## 1.20 SET RLO置位（=1）

格式

SET

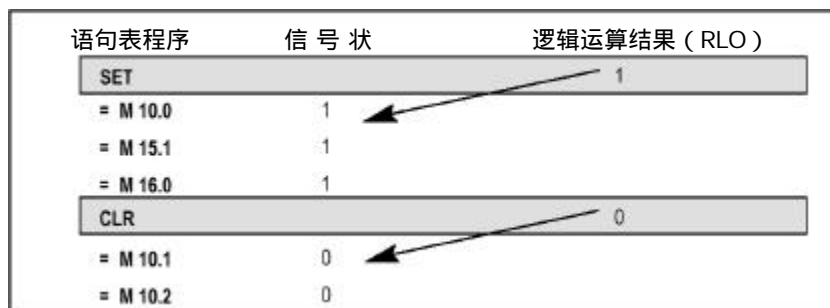
说明

使用 RLO 置位（SET）指令，可以将 RLO 的信号状态置为“1”。

状态字

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

举例



1.21 CLR RLO 清零 (=0)

格式

CLR

说明

使用 RLO 清零 (CLR) 指令，可以将 RLO 的信号状态置为“0”。

状态字

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

举例

语句表程序	信号状态	逻辑运算结果 (RLO)
SET		1
= M 10.0	1	←
= M 15.1	1	
= M 16.0	1	
CLR		0
= M 10.1	0	←
= M 10.2	0	

## 1.22 SAVE 把 RLO 存入 BR 寄存器

格式

SAVE

指令说明

使用 SAVE 指令，可以将 RLO 存入 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。

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，并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	-	-	-	-	-	-	-	-

1.23 FN 下降沿

格式

FN <位>

地址	数据类型	存储区	说 明
<位>	BOOL	I, Q, M, L, D	边沿标志, 存储 RLO 的前一信号状态。

说明

使用 RLO 下降沿检测指令 (FN <位>) 可以在 RLO 从 “1” 变为 “0” 时检测下降沿, 并以 RLO = 1 显示。

在每一个程序扫描周期过程中, RLO 位的信号状态都将与前一周期中获得的结果进行比较, 看信号状态是否有变化。前一 RLO 的信号状态必须保存在边沿标志地址 (<位>) 中, 以进行比较。如果在当前和先前的 RLO “1” 状态之间有变化 (检测到下降沿), 则在操作之后, RLO 位将为 “1”。

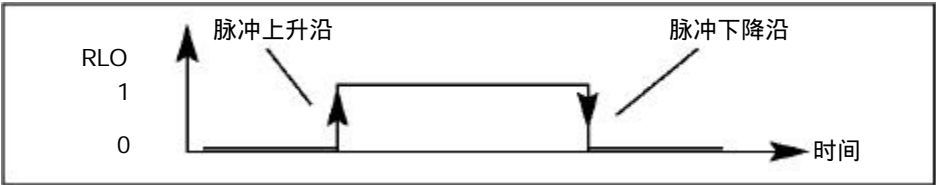
注意

由于一个块的本地数据只在块运行期间有效, 如果想要监视的位在过程映像中, 则该指令就不起作用。

状态字

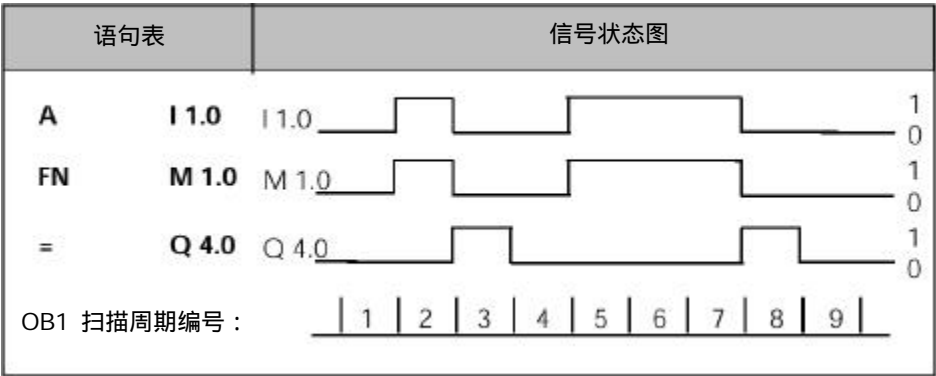
	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
写:	-	-	-	-	-	0	x	x	1

定义



举例

如果可编程控制器在触点 I 1.0 检测到一个下降沿，则它在一个 OB1 扫描周期使 Q 4.0 线圈得电。



1.24 FP 上升沿

格式

FP <位>

地址	数据类型	存储区	说 明
<位>	BOOL	I, Q, M, L, D	边沿标志, 存储 RLO 的前一信号状态。

说明

使用 RLO 上升沿检测指令 (FP <位>) 可以在 RLO 从 “0” 变为 “1” 时检测到一个上升沿, 并以 RLO = 1 显示。

在每一个程序扫描周期过程中, RLO 位的信号状态都将与前一周期中获得的结果进行比较, 看信号状态是否有变化。前一 RLO 的信号状态必须保存在边沿标志地址 (<位>) 中, 以进行比较。如果在当前和先前的 RLO “0” 状态之间有变化 (检测到上升沿), 则在操作之后, RLO 位将为 “1”。

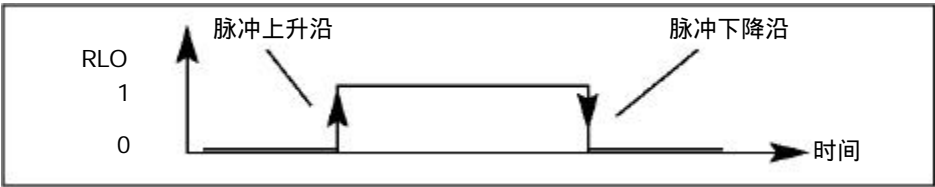
注意

由于一个块的本地数据只在块运行期间有效, 如果想要监视的位在过程映像中, 则该指令就不起作用。

状态字

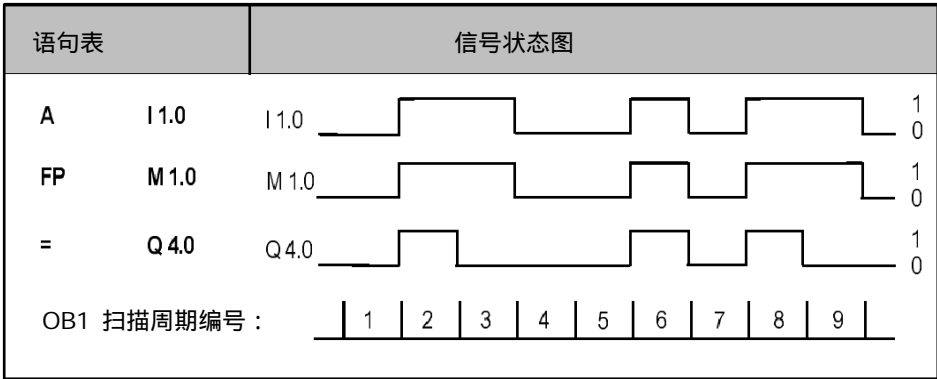
	BR	CC 1	CC 0	OV	OS	OR	STA	RLO	/FC
写:	-	-	-	-	-	0	x	x	1

定义



举例

如果可编程控制器在触点 I 1.0 检测到一个上升沿，则它在一个 OB1 扫描周期使 Q 4.0 线圈得电。



## 2 比较指令

### 2.1 比较指令概述

#### 说明

根据所选比较类型，对累加器 1 (ACCU1) 和累加器 2 (ACCU2) 进行比较：

- == 累加器 1 等于累加器 2
- <> 累加器 1 不等于累加器 2
- > 累加器 1 大于累加器 2
- < 累加器 1 小于累加器 2
- >= 累加器 1 大于等于累加器 2
- <= 累加器 1 小于等于累加器 2

如果比较结果为真，则指令的 RLO 为“1”。状态字位 CC 1 和 CC 0 表示“小于”、“等于”或“大于”关系。

可执行下列功能的比较指令：

- ?I 比较两个整数 (16位)
- ?D 比较两个双整数 (32位)
- ?R 比较两个浮点数 (32位)



2.2 ?I 比较两个整数（16位）

格式

==I , <>I , >I , <I , >=I , <=I

指令说明

使用比较整数指令（16 位），可以将累加器 2 中低字的内容与累加器 1 中低字的内容进行比较。累加器 2 和累加器 1 低字的内容都作为 16 位整数。比较的结果以 RLO 以及相关状态字位的设置来表示。RLO = 1 表示比较的结果为“真”；RLO = 0 表示比较的结果为“假”。状态字位 CC 1 和 CC 0 表示“小于”、“等于”或“大于”关系。

状态字

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

RLO 值

执行的比较指令	ACCU 2 > ACCU 1 时的 RLO 结果	ACCU 2 = ACCU 1 时的 RLO 结果	ACCU 2 < ACCU 1 时的 RLO 结果
==I	0	1	0
<>I	1	0	1
>I	1	0	0
<I	0	0	1
>=I	1	1	0
<=I	0	1	1

举例

STL	解 释
L MW10	// 装入存储字 MW10 的内容（16 位整数）。
L IW24	// 装入输入字 IW24 的内容（16 位整数）。
>I	// 比较累加器 2 低字中的内容（MW10）是否大于累加器 1 低字中的内容（IW24）。
= M2.0	// 如果 MW10 > IW24，则 RLO = 1。

2.3 ? D 比较两个双整数（32位）

格式

==D , <>D , >D , <D , >=D , <=D

指令说明

使用比较双整数指令（32 位），可以将累加器 2 中的内容与累加器 1 中的内容进行比较。累加器 2 和累加器 1 的内容都作为 32 位整数。比较的结果以 RLO 以及相关状态字位的设置来表示。RLO = 1 表示比较的结果为“真”；RLO = 0 表示比较的结果为“假”。状态字位 CC 1 和 CC 0 表示“小于”、“等于”或“大于”关系。

状态字

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

RLO 值

执行的比较指令	ACCU 2 > ACCU 1 时的 RLO 结果	ACCU 2 = ACCU 1 时的 RLO 结果	ACCU 2 < ACCU 1 时的 RLO 结果
==D	0	1	0
<>D	1	0	1
>D	1	0	0
<D	0	0	1
>=D	1	1	0
<=D	0	1	1

举例

STL	解 释
L MD10	// 装入存储双字 MD10 的内容（32 位双整数）。
L ID24	// 装入输入双字 ID24 的内容（32 位双整数）。
>D	// 比较累加器 2 中的内容（MD10）是否大于累加器 1 中的内容（ID24）。
= M 2.0	// 如果 MD10 > ID24，则 RLO = 1。

2.4 ? R 比较两个浮点数 ( 32 位 )

格式

==R , <>R , >R , <R , >=R , <=R

指令说明

使用比较浮点数指令 ( 32 位 , IEEE-FP ) , 可以将累加器 2 中的内容与累加器 1 中的内容进行比较。累加器 1 和累加器 2 的内容都作为 32 位浮点数 ( IEEE-FP ) 。比较的结果以 RLO 以及相关状态字位的设置来表示。RLO = 1 表示比较的结果为 “ 真 ” ; RLO = 0 表示比较的结果为 “ 假 ” 。状态字位 CC 1 和 CC 0 表示 “ 小于 ” 、 “ 等于 ” 或 “ 大于 ” 关系。

状态字

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

RLO 值

执行的比较指令	ACCU 2 > ACCU 1 时的 RLO 结果	ACCU 2 = ACCU 1 时的 RLO 结果	ACCU 2 < ACCU 1 时的 RLO 结果
==R	0	1	0
<>R	1	0	1
>R	1	0	0
<R	0	0	1
>=R	1	1	0
<=R	0	1	1

举例

STL	解 释
L MD10	// 装入存储双字 MD10 的内容 ( 浮点数 ) 。
L 1.359E+02	// 装入常数 1.359E+02。
>R	// 比较累加器 2 中的内容 ( MD10 ) 是否大于累加器 1 中的内容 ( 1.359-E+02 ) 。
= M 2.0	// 如果 MD10 > 1.359E+02 , 则 RLO = 1。

## 3 转换指令

### 3.1 转换指令概述

#### 说明

你可以使用以下指令将二进制编码十进制数（BCD）和整数转换为其它类型的数字：

- BTI      BCD 转成整数（16位）
- ITB      整数（16位）转成 BCD
- BTD      BCD 转成双整数（32位）
- ITD      整数（16 位）转成双整数（32 位）
- DTB      双整数（32位）转成 BCD
- DTR      双整数（32 位）转成浮点数（32 位，IEEE-FP）

你可以使用下述指令之一，形成一个整数的补码，或转换一个浮点数的符号：

- INVI      对整数求反码（16 位）
- INVD      对双整数求反码（32 位）
- NEGI      对整数求补码（16 位）
- NEGD      对双整数求补码（32 位）
- NEGR      对浮点数求反（32 位，IEEE-FP）

你可以使用以下“改变累加器 1 中的位顺序”指令，交换累加器 1 低字中或整个累加器中的字节顺序：

- CAW      交换累加器 1 低字中的字节顺序（16 位）
- CAD      交换累加器 1 中的字节顺序（32 位）

你可以使用以下任一指令，将累加器 1 中的 32 位 IEEE 浮点数转换成 32 位整数（双整数）。各条指令的取整方法略有不同：

- RND      取整
- TRUNC    截尾取整
- RND+    取整为较大的双整数
- RND-    取整为较小的双整数

3.2 BTI BCD 转成整数（16位）

格式

BTI

说明

使用 BTI 指令（3 位 BCD 数十进制 – 二进制转换），可以将累加器 1 低字中的内容作为一个 3 位二进制编码十进制数（BCD）进行编译，并转换为一个 16 位整数。转换结果保存在累加器 1 中。累加器 1 和累加器 2 的高字保持不变。

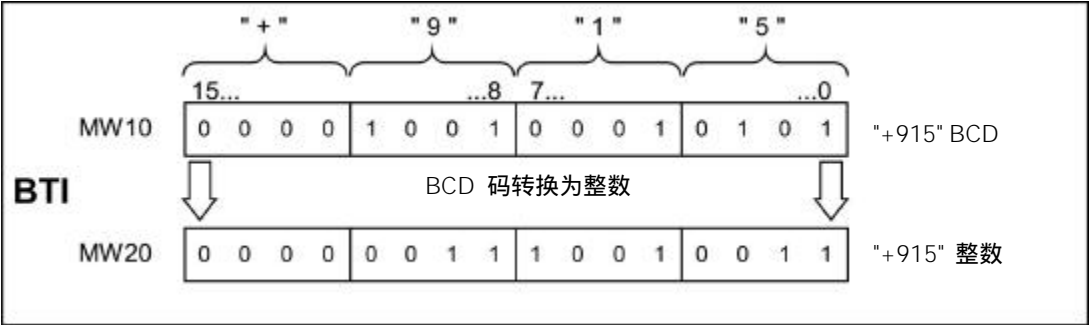
累加器 1 低字中的 BCD 数 :BCD 数的允许范围为 -999 - +999。位 0 到位 11 作为 BCD 数的数值进行编译，位 15 作为 BCD 数的符号位进行编译（0 = 正数，1= 负数）。位 12 到位 14 在转换时无用。如果 BCD 数的一个十进制数（4 位）在无效的 10 – 15 范围之间，则转换时会出现 BCDF 错误。一般地，CPU 会进入“STOP（停止）”状态。但是，通过编程 OB121，你可以设计其它的错误响应，来处理该同步编程错误。

状态字

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

举例

STL	解 释
L MW10	// 将 BCD 数装入累加器 1 低字中。
BTI	// 将 BCD 数转换为整数；结果保存到累加器 1 低字中。
T MW20	// 将结果（整数）传送到存储字 MW20。



3.3 ITB 整数（16位）转成 BCD

格式

ITB

说明

使用 ITB 指令（16 位整数的二进制 – 十进制转换），可以将累加器 1 低字中的内容作为一个 16 位整数进行编译，并转换为一个 3 位二进制编码十进制数（BCD）。转换结果保存在累加器 1 的低字中。位 0 – 11 包含 BCD 数的数值。位 12 – 15 设置为 BCD 数的符号位（0000 = 正数，1111= 负数）。累加器 1 和累加器 2 的高字保持不变。

BCD 数的范围在 –999 和 +999 之间。如果有数值超出这一范围，则状态位 OV（溢出位）和 OS（存储溢出位）被置为“1”。

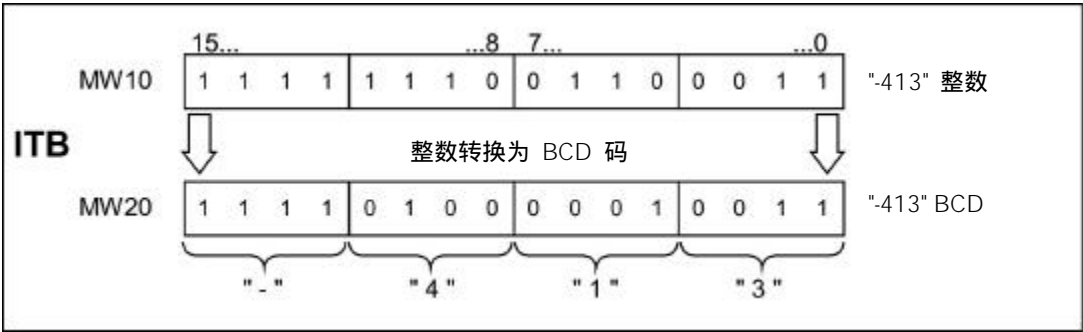
指令的执行与 RLO 无关，而且对 RLO 没有影响。

状态字

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

举例

STL	解 释
L MW10	// 将整数装入累加器 1 低字中。
ITB	// 将整数转换为 BCD 数（16 位）；结果保存到累加器 1 低字中。
T MW20	// 将结果（BCD 数）传送到存储字 MW20。



3.4      BTD   BCD 转成整数（32位）

格式

BTD

说明

使用 BTD 指令（7 位 BCD 数十进制 – 二进制转换），可以将累加器 1 中的内容作为一个 7 位二进制编码十进制数（BCD）进行编译，并转换为一个 32 位双整数。转换结果保存在累加器 1 中。累加器 2 保持不变。

累加器 1 中的 BCD 数：BCD 数的允许范围为 -9,999,999 - +9,999,999。位 0 到位 27 作为 BCD 数的数值进行编译，位 31 作为 BCD 数的符号位进行编译（0 = 正数，1= 负数）。位 28 到位 30 在转换时无用。

如果 BCD 数的任一十进制数（BCD 编码 4 位一组）在无效的 10 –15 范围之间，则转换时会出现 BCDF 错误。一般地，CPU 会进入“STOP（停止）”状态。但是，通过编程 OB121，你可以设计其它的错误响应，来处理该同步编程错误。

状态字

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

举例

STL	解 释
L   MD10	// 将 BCD 数装入累加器 1。
BTD	// 将 BCD 数转换为双整数；结果保存到累加器 1 中。
T   MD20	// 将结果（双整数）传送到存储双字 MD20。



3.5 ITD 整数（16 位）转成双整数（32 位）

格式

ITD

说明

使用 ITD 指令（16 位整数转换成为 32 位整数），可以将累加器 1 低字中的内容作为一个 16 位整数进行编译，并转换为一个 32 位双整数。转换结果保存在累加器 1 中。累加器 2 保持不变。

状态字

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

举例

STL	解 释
L MW12	// 将整数装入累加器 1。
ITD	// 将整数（16 位）转换为双整数（32 位）；结果保存到累加器 1 中。
T MD20	// 将结果（双整数）传送到存储双字 MD20。

例如：MW12 = “-10”（整数，16 位）

内 容	累加器 1 高字				累加器 1 低字			
位	31...	..	..	...16	15...	..	..	...0
ITD 执行之前	XXXX	XXXX	XXXX	XXXX	1111	1111	1111	0110
ITD 执行之后	1111	1111	1111	1111	1111	1111	1111	0110
(X = 0 或 1，该位不用于转换)								



3.6 DTB 双整数（32位）转成 BCD

格式

DTB

说明

使用 DTB 指令（32 位整数的二进制 – 十进制转换），可以将累加器 1 中的内容作为一个 32 位双整数进行编译，并转换为一个 7 位二进制编码十进制数（BCD）。转换结果保存在累加器 1 中。位 0 – 27 包含 BCD 数的数值。位 28 – 31 设置为 BCD 数的符号位（0000 = 正数，1111 = 负数）。累加器 2 保持不变。

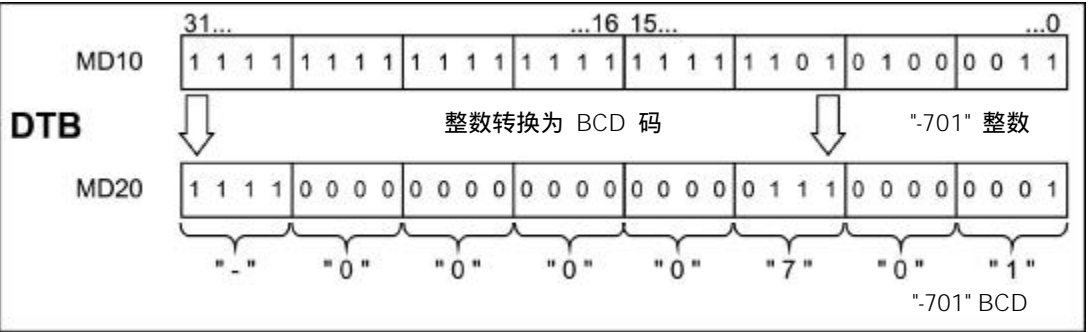
BCD 数的范围在 -9,999,999 和 +9,999,999 之间。如果有数值超出这一范围，则状态位 OV（溢出位）和 OS（存储溢出位）被置为“1”。

状态字

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

举例

STL	解 释
L MD10	// 将 32 位整数装入累加器 1。
DTB	// 将整数（32 位）转换为 BCD 数；结果保存到累加器 1 中。
T MD20	// 将结果（BCD 数）传送到存储双字 MD20。



3.7 DTR 双整数（32 位）转成浮点数（32 位，IEEE-FP）

格式

DTR

说明

使用 DTR 指令( 32 位整数转换为 32 位 IEEE 浮点数 ),可以将累加器 1 中的内容作为一个 32 位整数进行编译，并转换为一个 32 位 IEEE 浮点数。如果必要的话，该指令还可对结果进行取整。（一个 32 位整数要比一个 32 位浮点数精度高）。其结果保存在累加器 1 中。

状态字

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

举例

STL	解 释
L MD10	// 将 32 位整数装入累加器 1。
DTR	// 将双整数转换为浮点数（32 位，IEEE FP）；结果保存到累加器 1 中。
T MD20	// 将结果（BCD 数）传送到存储双字 MD20。

