第四章　PLC的指令系统

S7-300PLC的编程指令包括简单逻辑处理指令与功能指令两类。简单逻辑处理指令用于实现开关量信号(二进制位信号)的“读入”、“输出”及“与”、“或”、“非”等逻辑操作。在梯形图编程时,简单逻辑处理指令可用触点、线圈、连线等基本符号表示。功能指令用于实现定时、计数、逻辑处理、数据比较、代码转换、数学运算、传送、移位等功能。功能指令由指令代码、控制输入、操作数、执行状态输出、结果存储器等部分组成。在梯形图编程时，功能指令需要用功能指令框或功能触点来表示。指令系统是 PLC的核心内容，只有认真掌握了指令的使用,才能够应用指令完成各种功能，体现 PLC的先进性和实用性。

4.1PLC的指令概述

4.1.1 指令的组成

1.语句指令

一条指令是由一个操作码和一个操作数组成，操作数由标识符和参数组成。操作码定义要执行的功能，操作数为执行该操作所需要的信息。

例如，A　I1.5是一条位逻辑操作指令，其中:“A”是操作码，它表示执行“与”操作；“I1.5”是操作数，表示对输入继电I1.5进行的操作。

有些语句指令不带操作数，它们操作的对象是唯一的。例如，NOT指令，它是对逻辑操作结果(RLO)取反。

2.梯形逻辑指令

梯形逻辑指令用图形元素表示 PLC要完成操作。在梯形逻辑指令中。其操作码是用图素表示的。该图素形象地表明 PLC做什么，其操作数的表示方法与语句指令相同如：

Q4.0
——( )

该指令中：——( ) 可认为是操作码,表示一个二进制赋值操作；Q4.0是操作数，表示赋值的对象。

梯形逻辑指令也可不带操作数。例如，——| NOT |—— 指令是对逻辑操作结果取反的操作。

4.1.2 操作数

1.标识符及表示参数

一般情况下，指令的操作数存放在PLC的存储器中，此时操作数由操作数标识符和参数组成操作数标识符由主标识符和辅助标识符组成，主标识符表示操作数所在的存储区，辅助标识符则进一步说明操作数的位数长度。如果没有辅助标识符，则操作数的位数是一位。

（1）主标识符I(输入过程映像存储区)，Q(输出过程映象存储区)，M(位存储区)，PI(外部输入)，PQ(外部输入)，T(定时器)，C(计数器)，DB(数据块)，L(本地数据)。

（2）辅助标识符 X(位),B(字节),W(字, 2 字节),D(双字, 4 字节)。

PLC各存储区的功能见表4-1。

表4-1　　存储器及其功能

存储区	功　能	标识符	寻址范围
定时器(T)	访问本区域可得到定时剩余时间	T	0～255
计数器(C)	访问本区域可得到当前计数器值	C	0～255
数据块(DB)	本区域包含所有数据块的数据	DBX	0～65 535.7
		DBB	0～65 535
		DBW	0～65 534
		DBD	0～65 532
		DIX	0～65 535.7
		DIB	0～65 535
		DIW	0～65 534
		DID	0～65 532
本地数据(L)	本区域存放逻辑块(OB,FB或FC)中使用的临时数据。当逻辑块结束时,数据丢失	L	0～65 535.7
		LB	0～65 535
		LW	0～65 534
		LD	0～65 532

PLC物理存储器是以字节为单位的，所以存储单元规定为字节单元。 位地址参数用一个点与字节地址分开，如：

M　 I0.1

当操作数长度是字或双字时，标识符后给出的标识参数是字或双字内的最低字节单元号。图4-1给出了字节、字、双字的相互关系及表示方法。当使用宽度为字或双字的地址时，应保证没有生成任何重叠的字节分配，以免造成数据读写错误。

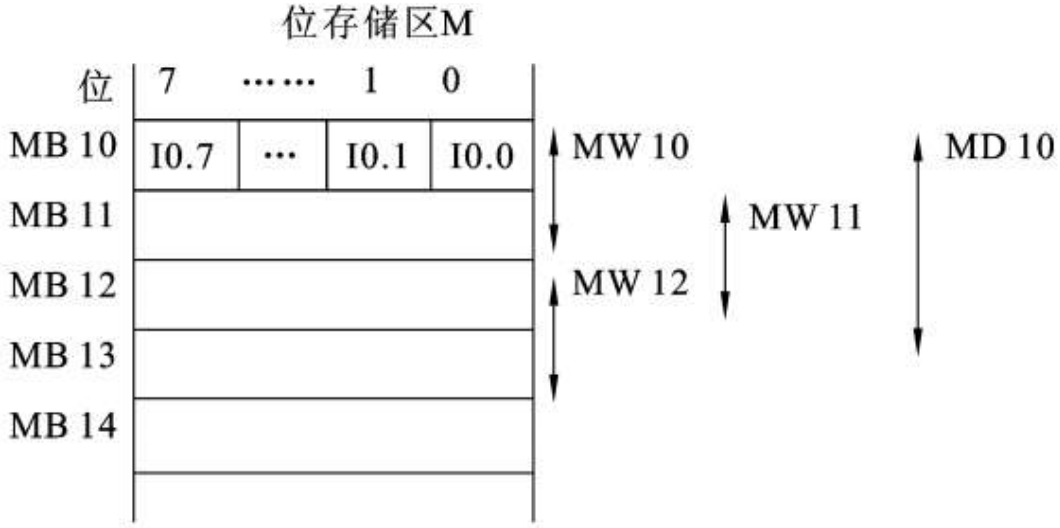


图4-1　　以字节单元为基准标记存储器存储单元

2.操作数的表示法

在STEP7中,操作数有两种表示方法:一种是物理地址(绝对地址)表示法,另一种是符号地址表示法.

使用物理地址表示操作数时,要明确指出操作数所在的存储区,以及该操作数的位数具体位置。例如：Q4.0。

STEP7允许用符号地址表示操作数，例如：Q4.0可用符号名MOTOR_ON替代表示。需要注意的是。符号名必须先定义后使用，而且符号名必须是唯一的，不能重名。定义符号时，需要指明操作数所在的存储区，以及操作数的位数、具体位置及数据类型。

4.1.3 寻址方式

1.立即寻址

立即寻址是对常数或常量的寻址方式，操作数本身直接包含在指令中,具体实例如下所示。

```
SET                 //把 RLO 置 1
OW   W#16#A320      //将常量 W#16#A320与累加器1进行“或”运算
L    27             //把整数 27 装入累加器 1
L    ‘ABCD’         //把 ASCII 码字符 ABCD 装入累加器 1
L    C#0100         //把 BCD 码常数 0100 装入累加器 1
```

2.直接寻址

直接寻址包括对寄存器和存储器的直接寻址。在直接寻址的指令中，直接给出操作数的存储单元地址。例如：

```
A    I0.0      //对输入位I0.0 进行“与”逻辑操作
S    L20.0     //把本地数据位L20.0置1
=    M115.4    //使存储区位M115.4的内容等于RLO的内容
L    IB10      //把输入字节IB10的内容装入累加器1
T    DBD12     //把累加器1中的内容传送给数据双字DBD12中
```

3.存储器间接寻址

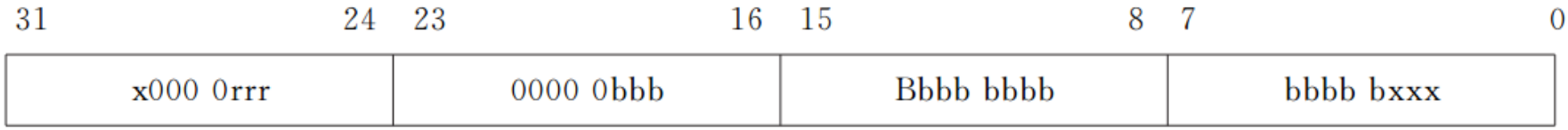
在存储器间接寻址的指令中，给出一个存储器(必须是表4-1中的存储器)，该存储器的内容是操作数所在存储单元的地址，故该地址又称为地址指针。存储器间接寻址方式的优点是，当程序执行时，能改变操作数的存储器地址，这对程序中的循环尤为重要。例如：

```
A   I[MD 2]    //对由MD 2指出的输入位进行“与”逻辑操作。
               //如:当MD 2的值为
               //2#0000 0000 0000 0000 0000 0000 0101 0110
               //则是对I 10.6进行“与”操作
```

4. 寄存器间接寻址

在S7中有两个地址寄存器，分别是 AR1和 AR2。通过地址寄存器，可以对各存储区的存储器内容实现寄存器间接寻址。地址寄存器的内容加上偏移量形成地址指针，该指针指向数值所在的存储单元。

地址寄存器存储的地址指针有两种格式，即区内寄存器间接寻址和区域间寄存器间接寻址，二者的长度均为双字。图4-2给出了这两种格式的细节及其差别，区域标识位的组合状态见表4-2。



位31的x＝ 0 表明是区域内寄存器间接寻址；x＝ 1 表明是区域间寄存器间接寻址。
位24、25和26(rrr)为被寻址地址的区域标识(见表4-1)
位3至18为被寻址位的字节编号(0~65535)
位 0 至 2 (xxx)为被寻址地址的中位编号(0~7)

图4-2　　寄存器间接寻址的指针格式

表4-2 地址指针区域标识位含义

区域标识符	存储区	位 26、25 和 24 的二进制值
P	I/O，外设 I/O	000
I	输入过程暂存区	001
Q	输出过程暂存区	010
M	位存储区	011
DBX	共享数据块	100
DIX	背景数据块	101
L	本地数据	111

4.1.4 状态字

状态字用于表示CPU执行指令时所具有的状态。一些指令是否执行或以何种方式执行，可能取决于状态字中的某些位，执行指令时也可能改变状态字中的某些位，并且也能在位逻辑指令或字逻辑指令中访问并检测它们。图4-3显示了状态字的结构，每一位的功能分别介绍如下。

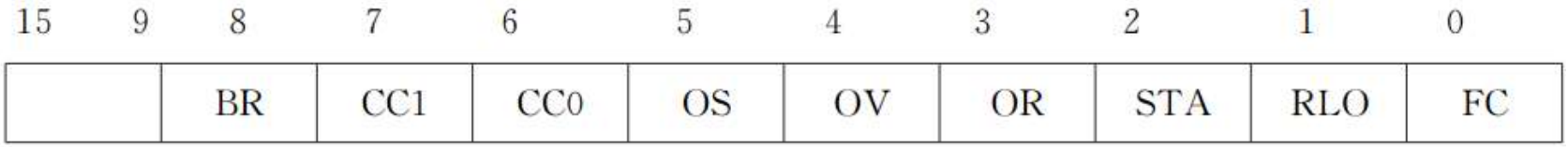


图4-3　　状态字的结构

（1）首次检测位(FC)状态字的位0称为首次检测位。若 FC位的状态为0，则表明一个梯形逻辑网络的开始，或者该指令为逻辑串第一条指令。

（2）逻辑操作结果(RLO)状态字的位1称为逻辑操作结果 RLO(result of logic operation)。该位用于存储位逻辑指令或算术比较指令的结果。

（3）状态位(STA)状态字的位2称为状态位。状态位不能用指令检测，它只是在程序测试中被 CPU 解释并使用。

（4）或位(OR)状态字的位3称为或位。在先逻辑“与”后逻辑“或”的逻辑串中，或位暂存逻辑“与”的操作结果,以便进行后面的逻辑“或”运算，其他指令将或位清0。

（5）溢出位(OV)状态字的位 4 称为溢出位。溢出位被置1,则表明一个算术运算或浮点数比较指令执行时出现错误(包括溢出、非法操作、不规范格式等)。

（6）溢出状态保持位(OS)状态字的位5称为溢出状态保持位。OV 被置1时OS也被置1,OV被清0时OS仍保持。只有以下指令才能使OS位复位：JOS(OS＝ 1 时跳转)，块调用和块结束指令.

（7）条件码1(CC1)和条件码0(CC0)状态字的位7和位6称为条件码1和条件码0。这两位结合起来用于表示在累加器 1 中产生的算术运算或逻辑运算结果与 0 的大小关系，以及表示比较指令的执行结果或移位指令的移出位状态。详见表4-3和表4-4。

表4-3　算术运算后的CC1和CC0

CC1	CC0	算术运算无溢出	整数算术运算有溢出	浮点数算术运算有溢出
0	0	结果＝0	整数加运算时产生负范围溢出	平缓下溢
0	1	结果＜0	乘运算时负溢出；加、减、取负运算时正溢出	负范围溢出
1	0	结果＞0	乘、除运算时正溢出；加、减运算时负溢出	正范围溢出
1	1	—	在除运算时除数为 0	非法操作

表4-4　比较、移位和循环移位、字逻辑指令后的CC1和CC0

CC1	CC0	比较指令	移位和循环移位指令	字逻辑指令
0	0	累加器 2＝累加器 1	移出位＝0	结果＝0
0	1	累加器 2＜累加器 1	—	—
1	0	累加器 2＞累加器 1	—	结果≠0
1	1	不规范(只用于浮点数比较)	移出位＝1	—

（8）二进制结果位(BR)状态字的位8称为二进制结果位。该位将字处理程序与位处理联系起来，用于表示字操作结果是否正确(异常)。将 BR 位加入程序后，无论字操作结果如何，都不会造成二进制逻辑链中断。 在LAD的方块指令中，BR位与ENO有对应关系，用于表明方块指令是否被正确执行。如果执行出现了误,BR 位为0,ENO也为0；如果功能被正确执行，则BR位为1，ENO也为1。

在用户编写的FB和FC程序中，必须对BR位进行管理,当功能块正确运行后使BR位为1，否则使其为0。 使用STL指令SAVE或LAD指令—(SAVE)，可将RLO存入BR中，从而达到管理BR位的目的。当FB或FC执行无错误时，使RLO为1并存入BR，否则，在BR中存入0。

4.2 位逻辑指令

4.2.1 位逻辑运算指令

位逻辑运算指令包括“与”(AND)、“或”(OR)、“异或”(XOR)指令及其组合。位逻辑运算令对“0”或“1”这些布尔操作数扫描，经逻辑运算后将逻辑操作结果送入状态字的RLO位。

1.“与”和“与非”（A,AN）指令

逻辑“与”在梯形图里是用串联的触点回路表示的，如果串联回路中的所有触点皆闭合，那么该回路就通“电”了,如图4-4所示。

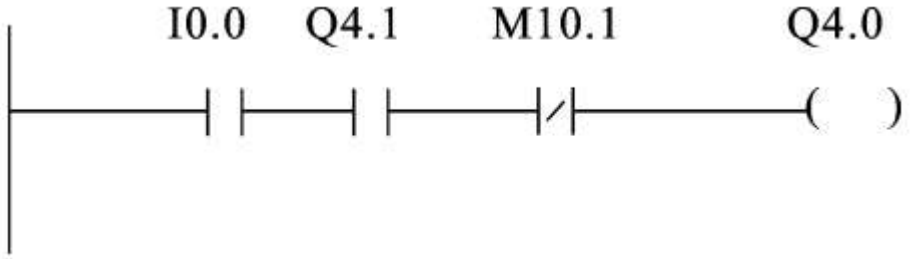


图4-4 “与”逻辑梯形图

左侧梯形逻辑图可以用语句表指令完全表示,对应的语句表如下。

```
A    I0.0
A    Q4.1
AN   M10.1
=    Q4.0
```

2.“或”和“或非”(O,ON)指令

逻辑“或”在梯形图里是用并联的触点回路表示的，被扫描的操作数标在触点上方。在图4-5中，只要有一个触点闭合，输出4.1的信号状态就为“ 1 ”。

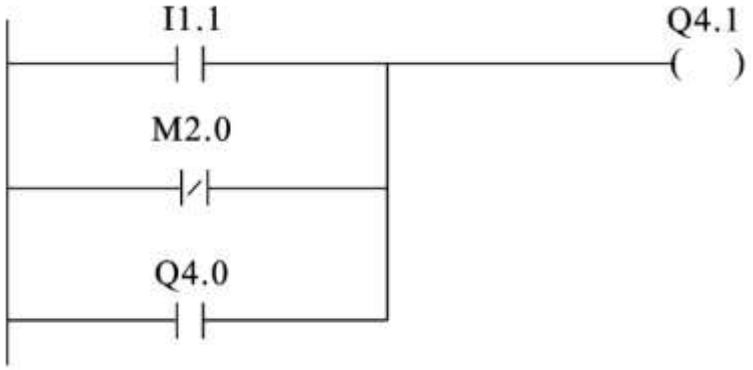


图4-5 “或”逻辑梯形图及语句表

它对应的语句表如下。

```
O    I1.1
ON   M2.0
O    Q4.0
=    Q4.1
```

3.“异或”和“异或非”(X,XN)指令

如图4-6所示是“异或”逻辑梯形图，图的下面是与梯形图对应的语句表。在梯形图对应的语句表中，使用了“异或”和“异或非”指令，分别使用助记符“X”和“XN”来标识。这两个指令类似于“或”和“或非”指令，用于扫描并联回路能否“通电”。

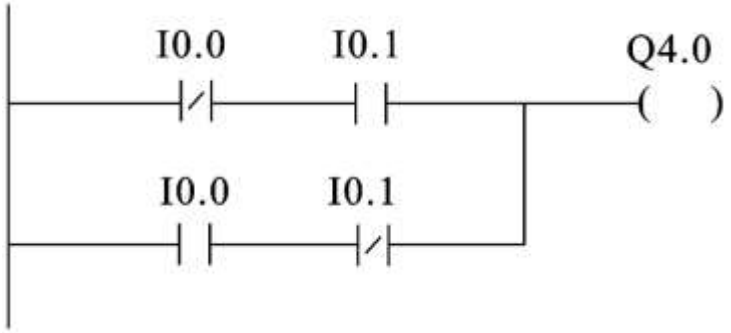


图4-6“异或”逻辑梯形图

它对应的语句表如下。

```
X    I0.0
X    I0.1
=    Q4.0
```

4.2.2 位操作指令

1.输出指令

逻辑串输出指令又称为赋值操作指令，该操作把状态字中RLO的值赋给指定的操作数(位地址)。表4-5中列出了操作数的数据类型和所在的存储区。

表4-5　输出指令

LAD指令	STL指令	功能	操作数	类型	存储区
<位地址> ——(　)	=<位地址>	逻辑串赋值输出	<位地址>	BOOL	Q、M、D、L
<位地址> ——(#)—	—	中间结果赋值输出	<位地址>	BOOL	Q、M、D、L

一个RLO可被用来驱动几个输出元件。在LAD中，输出线圈是上下依次排列的。在STL中，与输出信号有关的指令被一个接一个地连续编程，这些输出具有相同的优先级。图4-7是多重输出梯形图。

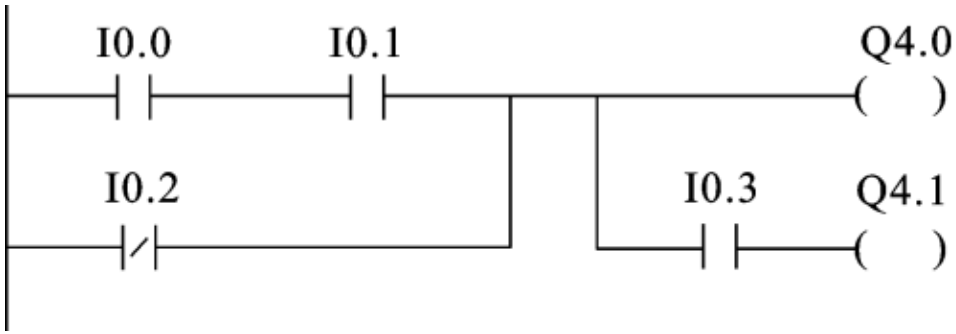


图4-7　多重输出梯形图

与之对应的语句表如下。

```
A    I0.0
A    I0.1
ON   I0.2
=    Q4.0
A    I0.3
=    Q4.1
```

2. 置位/复位指令

置位/复位指令根据RLO的值来决定被寻址位的信号状态是否需要改变。若RLO的值为1，被寻址位的信号状态被置1或清0，若RLO的值为0，则被寻址位的信号保持原状态不变。置位/复位指令的相关内容见表4-6。

表4-6　置位/复位指令

LAD 指令	STL 指令	功能	操作数	数据类型	存储区
＜位地址＞ ——(R)	R＜位地址＞	复位输出	＜位地址＞	BOOL TIMER COUNTER	Q、M、D、L T C
＜位地址＞ ——(S)—	S＜位地址＞	置位输出	＜位地址＞	BOOL	Q、M、D、L

置位/复位指令的梯形图如图4-8所示。

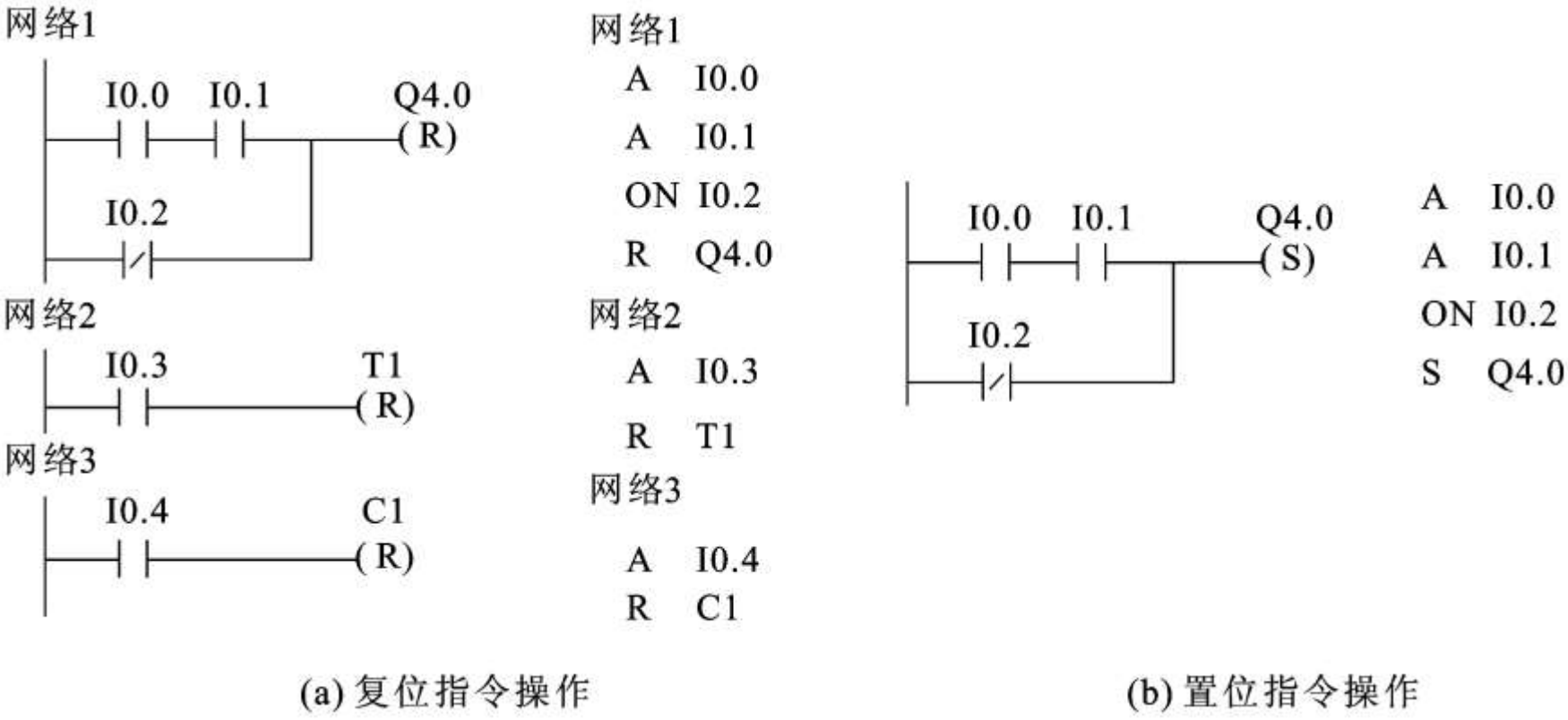
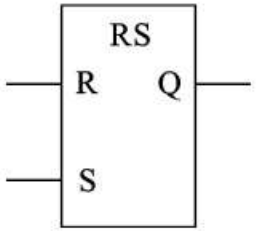
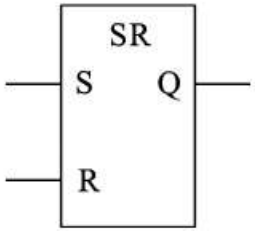


图4-8　　置位/复位指令

3. RS触发器

RS触发器的梯形图方块指令表示见表4-7。方块指令中有三个端口,分别为置位输入端(S)、复位输入端(R)及输出端(Q)。触发器可以用在逻辑串最右端，用于结束一个逻辑串；也可用在逻辑串中，从而影响右边的逻辑操作结果。

表4-7　RS触发器

置位优先 RS	复位优先型 RS	参　数	数据类型	存储区
<位地址>  	<位地址>  	＜位地址＞　需要置位、复位的位	BOOL	Q、M、D、L
		S　允许置位输入		
		R　允许复位输入		
		Q　＜地址＞的状态		

RS触发器分为置位优先型和复位优先型两种。置位优先型RS触发器的R端在S端之上，当两个输入端都为1时，下面的置位输入最终有效。即置位输入优先，触发器或被复位或保持复位不变，如图4-9所示。

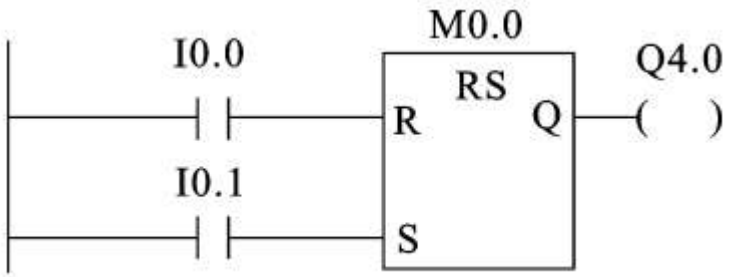


图4-9　　置位优先型 RS触发器

与之对应的语句表如下。

```
A   I0.0
R   M0.0
A   I0.1
S   M0.0
A   M0.0
=   Q4.0
```

4. 对 RLO 的直接操作指令

这一类指令直接对逻辑操作结果RLO进行操作，从而改变状态字中RLO位的状态。有关内容见表4-8。

表4-8　对RLO的直接操作指令

LAD 指令	STL 指令	功　　能	说　　明
——|NOT|——	NOT	取反 RLO	在逻辑串中，对当前的 RLO 取反
—	SET	置位 RLO	把 RLO 无条件置 1 并结束逻辑串； 使 STA 置 1，OR、FC 清 0
—	CLR	复位 RLO	把 RLO 无条件清 0 并结束逻辑串； 使 STA、OR、FC 清 0
——(SAVE)	SAVE	保存 RLO	把 RLO 存入状态字的 BR 位， 该指令不影响其他状态位

4.2.3 位测试指令

当信号状态变化时就会产生跳变沿。当信号从0变到1时，产生一个上升沿(或正跳沿)；若信号从1变到0，则产生一个下降沿(或负跳沿)。S7-300中有两类跳变沿检测指令，一种是对RLO的跳变沿检测的指令，另一种是对触点跳变沿直接检测的梯形图方块指令。具体内容见表4-9。

表4-9　　跳变沿检测指令

LAD 指令	STL 指令	功　　能	操作数	数据类型	存储区
<位地址> ——(P)——	FP<位地址>	RLO 正跳沿检测	<位地址>	BOOL	I、Q、M、D、L
<位地址> ——(N)——	FN<位地址>	RLO 负跳沿检测	<位地址>	BOOL	I、Q、M、D、L
触点正跳沿检测	触点负跳沿检测		参数	数据类型	存储区
<位地址1> POS Q <地址2> M_BIT	<位地址1> NEG Q <地址2> M_BIT		<位地址 1> 被检测的位(触点)	BOOL	I、Q、M、D、L
			M_BIT 存储被检测位上一个扫描周期的状态	BOOL	Q、M、D
			Q 单稳输出	BOOL	I、Q、M、D、L

图4-10是使用RLO正跳沿检测指令的例子。这个例子中，若CPU检测到输入I1.0一个正跳沿，将使得输出Q4.0的线圈在一个扫描周期内通电。将对输入I1.0常开触点进行扫描的RLO值存放在存储位M1.0中。

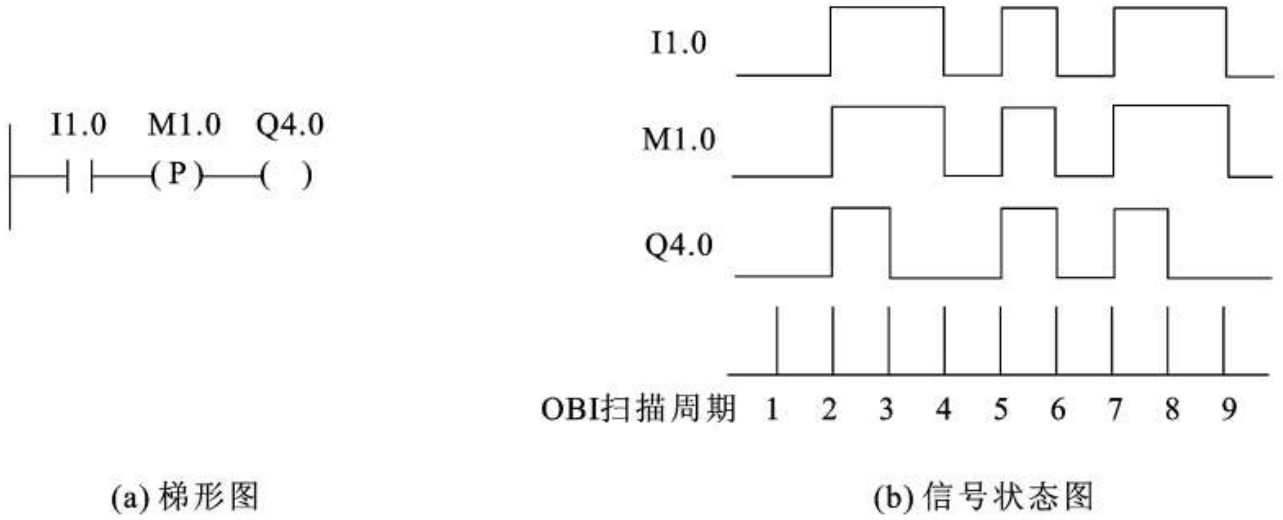


图4-10　RLO正跳沿检测

其对应的语句表如下。

```
A    I1.0
FP   M1.0
=    Q4.0
```

图4-11所示是使用触点负跳沿检测指令的例子。在图4-11中，由＜位地址1＞给出需要检测的触点编号(I0.3)，＜地址2＞(M0.0)用于存放该触点在前一个扫描周期的状态。

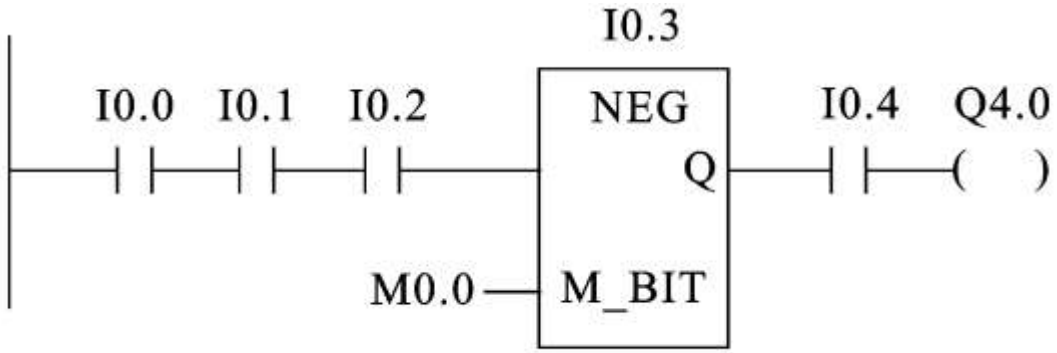


图4-11　触点负跳沿检测

在图4-11中，如果下列条件同时成立，则输出Q4.0。

（1）输入I0.0、I0.1和I0.2的信号状态为1。

（2）输入I0.3有负跳沿。

（3）输入I0.4的信号状态为1。

4.2.4 位逻辑指令编程举例

1.传送带控制

图4-12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：用于启动(Start)的S1和用于停止(Stop)的S2。在传送带的尾部也有两个按钮开关：S3用于启动(Start)，S4用于停止(Stop)。可以从任一端启动或停止传送带。另外，当传送带上的物件到达末端时，传感器S5可以使传送带停机。

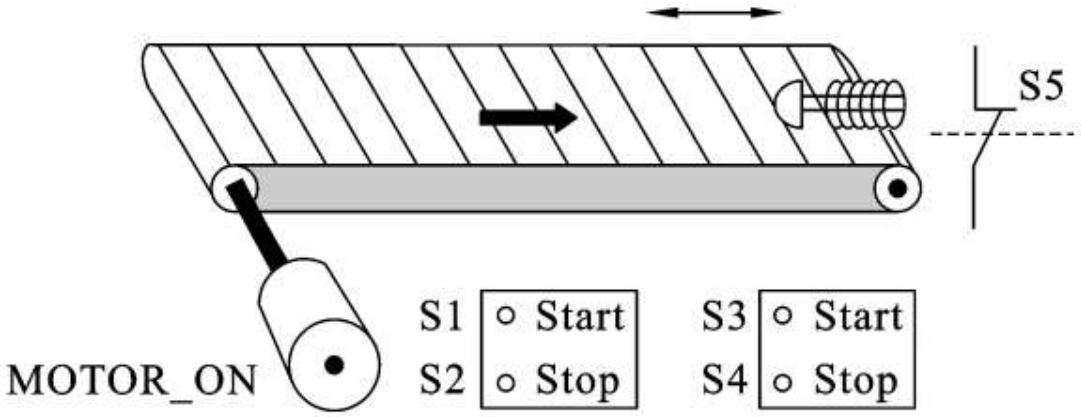


图4-12　传送带示意图

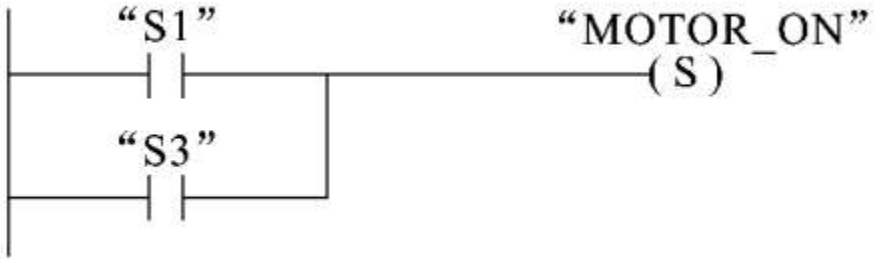
系统中各部件对应的编程元素见表4-10。

表4-10　用于传送带系统符号编程的元素

系统部件	绝对地址	符　　号	符号表
启动按钮开关	I1.1	S1	I1.1　S1
停止按钮开关	I1.2	S2	I1.2　S2
启动按钮开关	I1.3	S3	I1.3　S3
停止按钮开关	I1.4	S4	I1.4　S4
传感器	I1.5	S5	I1.5　S5
电　机	Q4.0	MOTOR_ON	Q4.0　MOTOR_ON

传送带控制的具体程序如图4-13所示

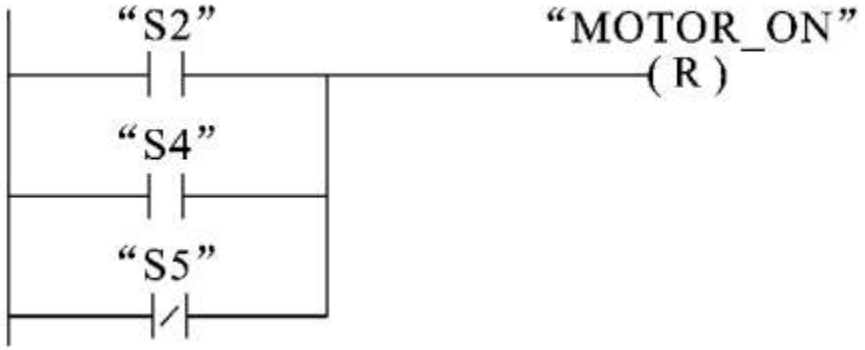
网络1：按任何一个启动开关，接通电机



网络1：

```
O   S1
O   S3
S   MOTOR_ON
```

网络2：按任何一个停止或打开传送带末端的常闭接点，停止电机



网络2：

```
O    S2
O    S4
ON   S5
R    MOTOR_ON
```

图4-13　　控制传送带程序

2. 串、并联组合表示法

当逻辑串是复杂组合时，CPU 的扫描顺序是先“与”后“或”。图4-14(a)所示的梯形逻辑是触点先并联后串联的例子，与其对应的语句表如下。

```
A(
O   I0.0
O   I0.2
)
A(
O   M10.0
O   M0.3
)
A   M10.1
=   Q4.0
```

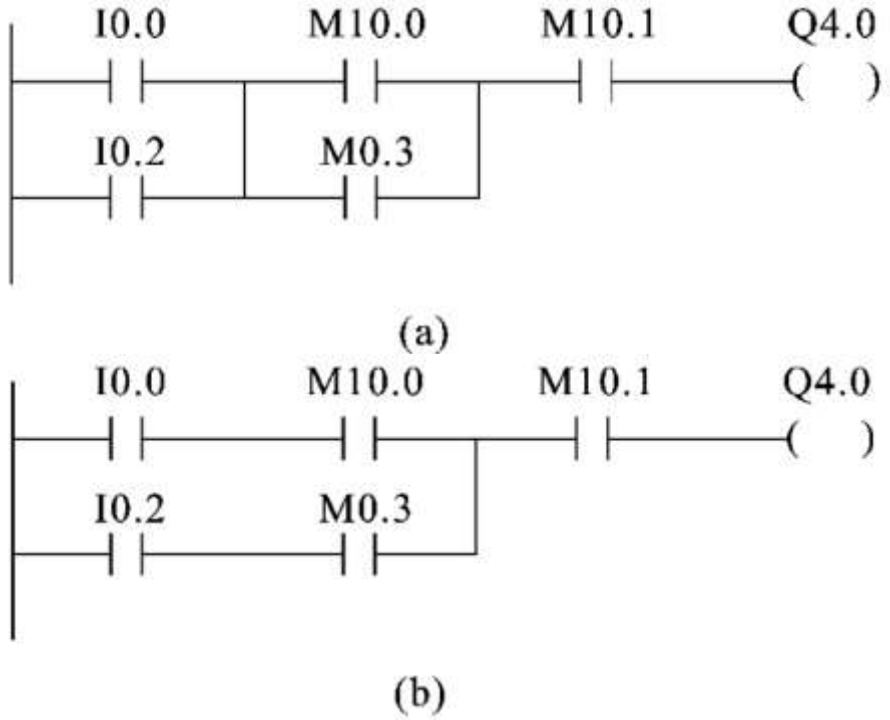


图4-14　串、并联组合逻辑梯形图

3. 风机监控程序

某设备有三台风机，当设备处于运行状态时，如果风机至少有两台以上转动，则指示灯常亮；如果仅有一台风机转动，则指示灯以0.5Hz的频率闪烁；如果没有任何风机转动，则指示灯以2Hz的频率闪烁。当设备不运行时，指示灯不亮。实现上述功能的梯形图程序如图4-15所示。

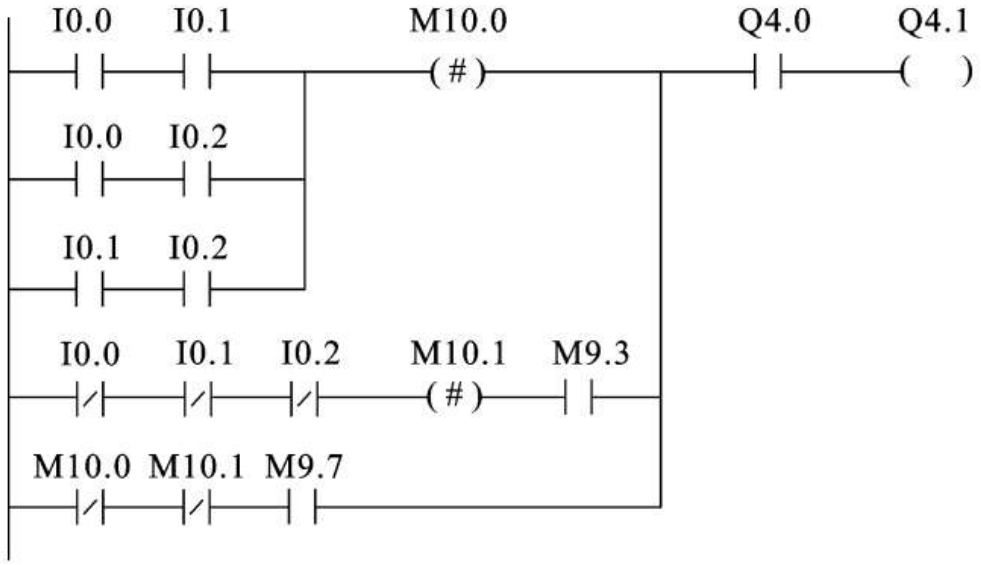


图4-15　风机监控程序

在图4-15中,输入位I0.0,I0.1,I0.2分别为风机1,2,3的反馈输入端。M9.3 为2Hz频率CPU中的时钟信号，M9.7为0.5Hz频率信号。

下面给出实现风机状态检测的语句表程序,从中可看出中间输出指令的用法.

```
A(
A(
A    I0.0
A    I0.1
O
A    I0.0
A    I0.2
O
A    I0.1
A    I0.2
)
= M10.0
A   M10.0
O(
AN  I0.0
AN  I0.1
AN  I0.2
=   M10.1
A   M10.1
A   M9.3
)
O
AN  M10.0
AN  M10.1
A   M9.7
)
A   Q4.0
=   Q4.1
```

4.3 定时器与计数器指令

4.3.1 定时器指令

定时器是PLC中的重要部件，它用于实现或监控时间序列。定时器是一种由位和字组成的复合单元，定时器的触点由位表示，其定时时间值存储在字存储器中。

S7-300/400提供的定时器有脉冲定时器(SP),扩展定时器(SE)，接通延时定时器(SD)，带保持的接通延时定时器(SS)和断电延时定时器(SF)等。

1.定时器的组成

在CPU的存储器中留出了定时器区域，用于存储定时器的定时时间值。每个定时器为2B，称为定时字。在S7-300中，最多允许使用256个定时器。

S7-200中定时时间由时基和定时值两部分组成，定时时间等于时基与定时值的乘积，采用减计时。定时时间到后会引起定时器触点的动作。

定时器的第0位到第11位存放二进制格式的定时值，第12、13位存放二进制格式的时基，如图4-16所示。表4-11给出了时基与定时范围。

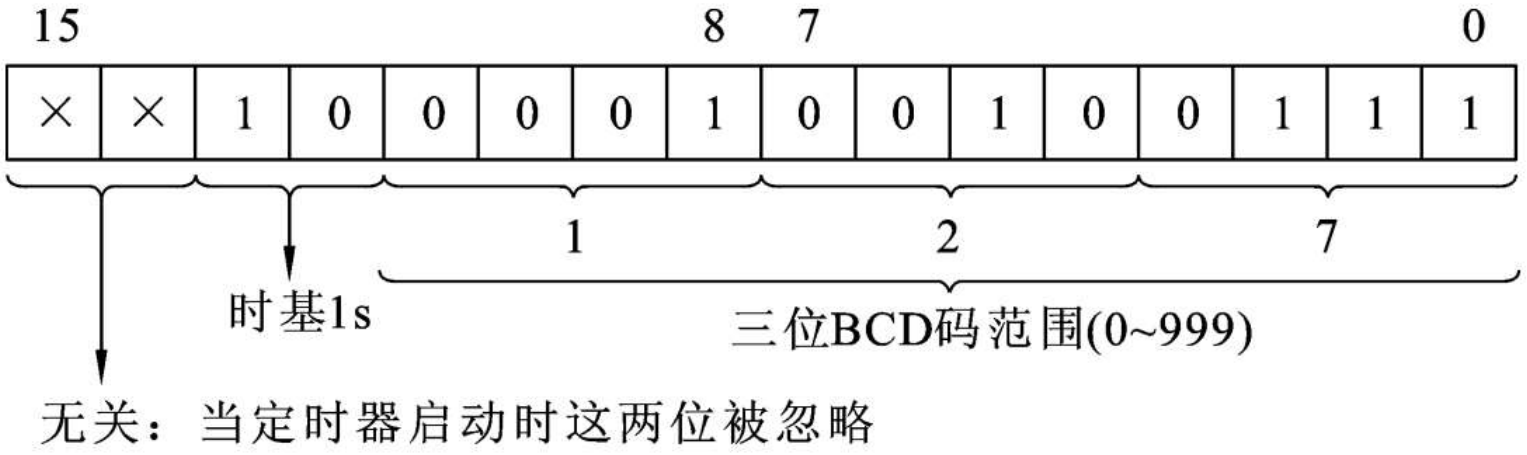


图4-16　累加器1低字的内容(定时值127，时基1s)

表4-11 时基与定时范围

时　基/ms	时基的二进制代码	分辨率/s	定 时 范 围
10	0　0	0.01	10MS 至 9S_990MS
100	0　1	0.1	100MS 至 1M_39S_900MS
10	1　0	1	1S 至 16M_39S
10	1　1	10	10S 至 2H_46M_30S

设置定时时间时，用户需给累加器1装入需要的数值。为了避免格式错误，推荐采用下述直观的句法。

L W#16#wxyz

其中,w为时基,取值可为0、1、2或3，分别表示时基为10ms、100ms、1s或10s；xyz为定时值，取值范围为1~999。

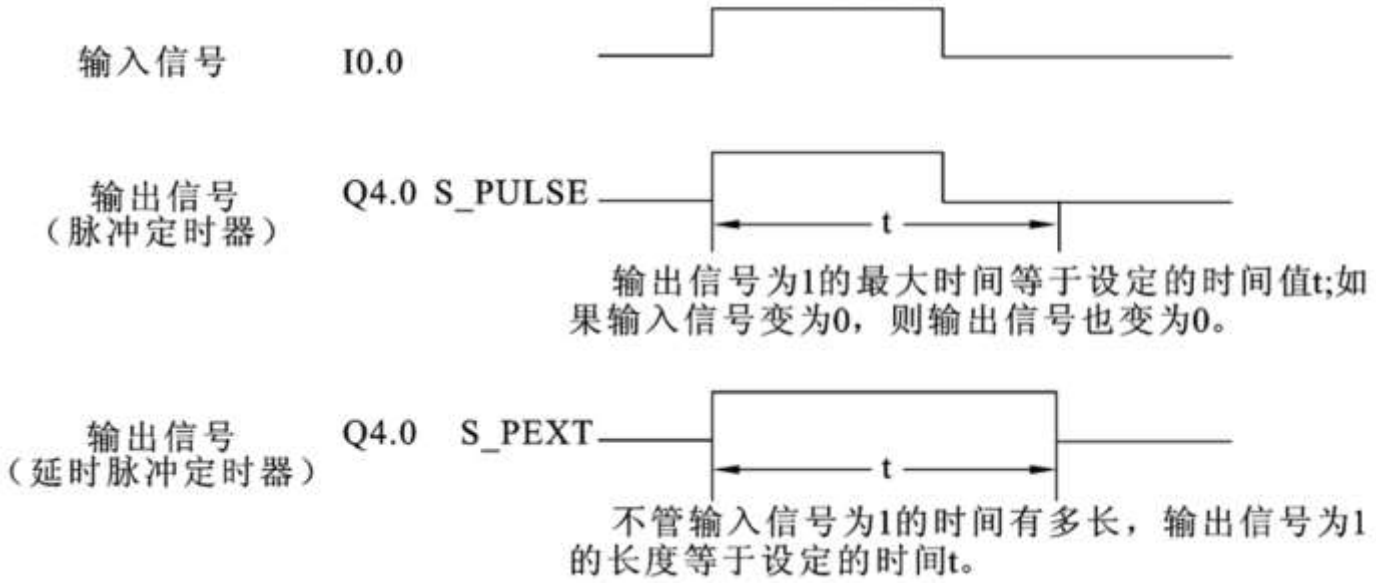
也可直接使用S5中的时间表示法装入定时数值，例如：

L S5T#aH_bbM_ccS_dddMS

其中，a表示小时，bb表示分钟，cc表示秒，ddd表示毫秒。时基是自动选择的，原则是能满足定时范围要求的最小时基。

2. 定时器的启动与运行

S7-300中的定时器与时间继电器的工作特点相似，对定时器同样应设置定时时间，也要启动定时器(使定时器线圈通电)。除此之外，定时器还增加了一些功能，如随时复位定时器、随时重置定时时间(定时器再启动)、查看当前剩余定时时间等。S7-300中的定时器不仅功能强，而且类型多。图4-17所示为定时作业如何正确选择定时器的示意图。下面将以LAD方块图为主详细介绍定时器的运行原理及使用方法。



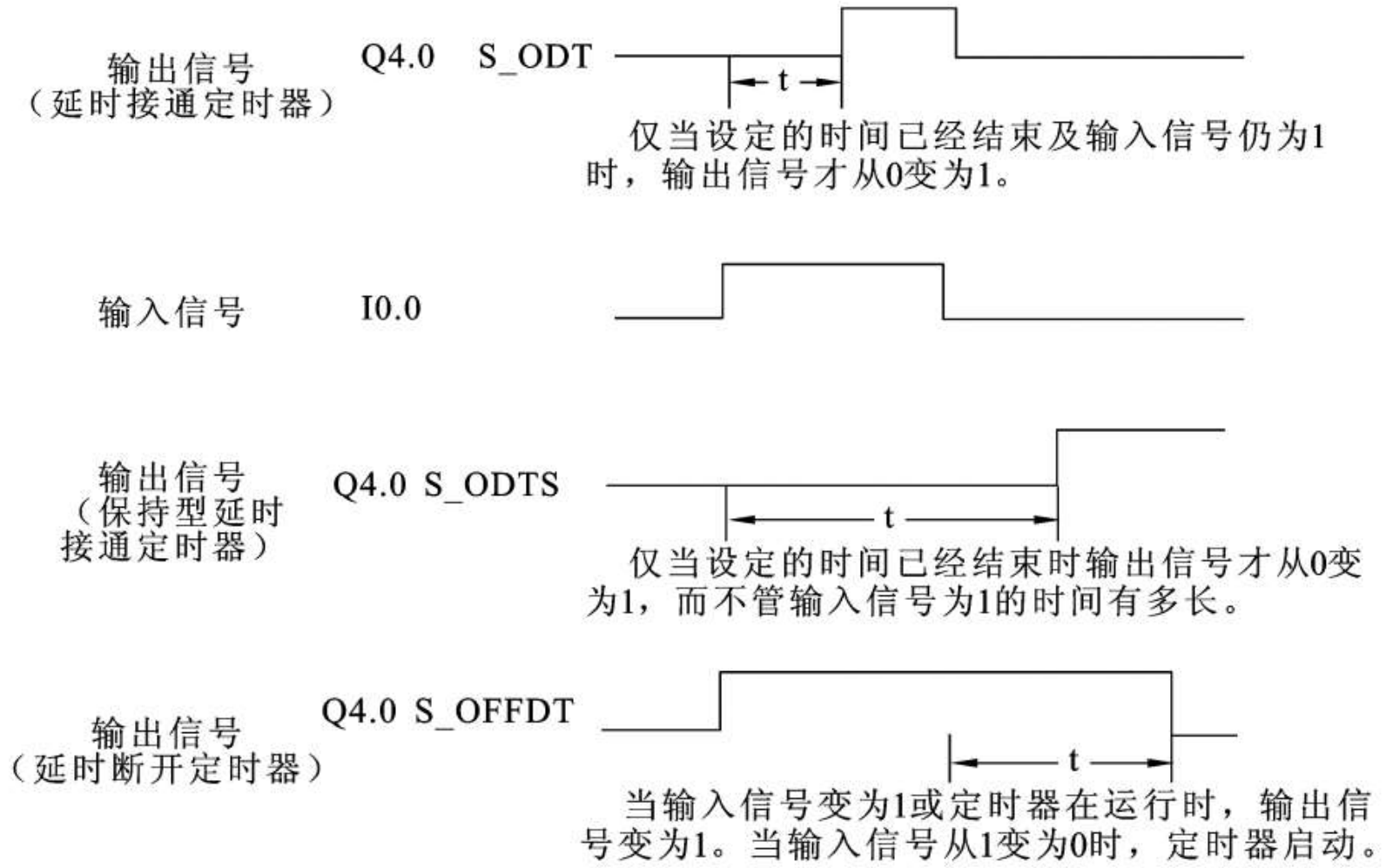


图4-17　5种类型定时器

3. 定时器梯形图方块指令

1）脉冲定时器

脉冲定时器梯形图如图4-18所示。

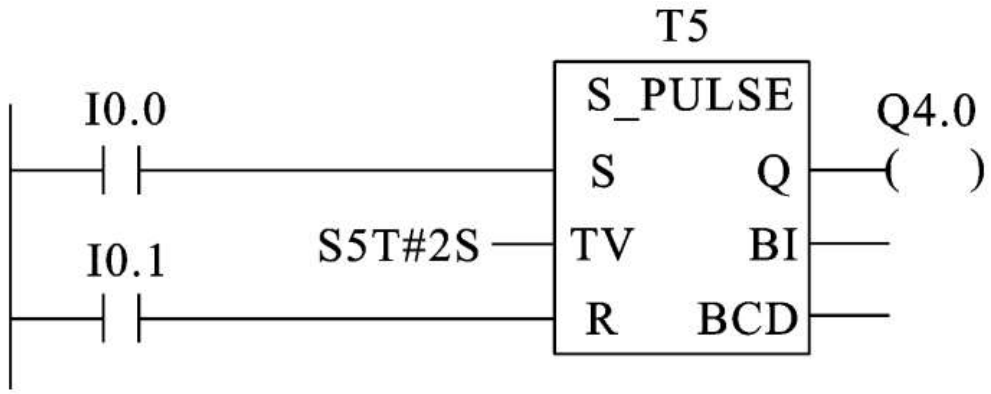


图4-18　脉冲定时器梯形图

图4-18所示的梯形图对应的指令表如下。

```
A   I0.0
L   S5T#2S
SP  T5
A   I0.1
R   T5
A   T5
=   Q4.0
```

脉冲定时器时序如图4-19所示。

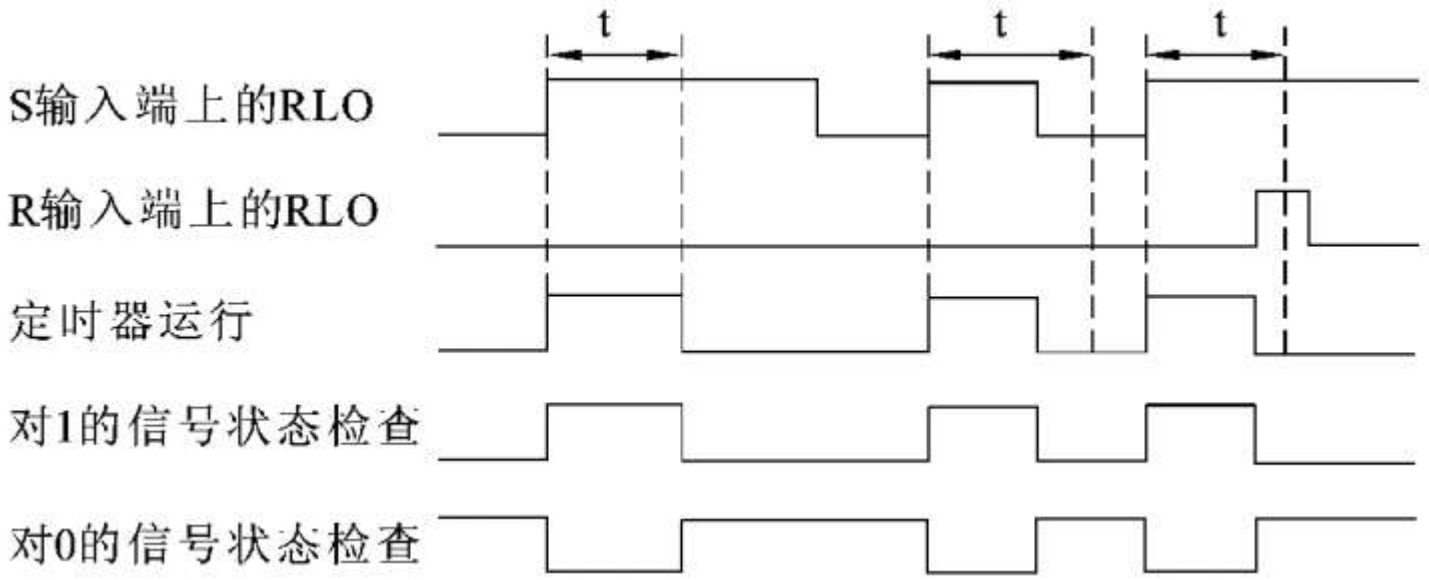


图4-19　脉冲定时器时序

2）接通延时定时器

接通延时定时器梯形图如图4-20所示。

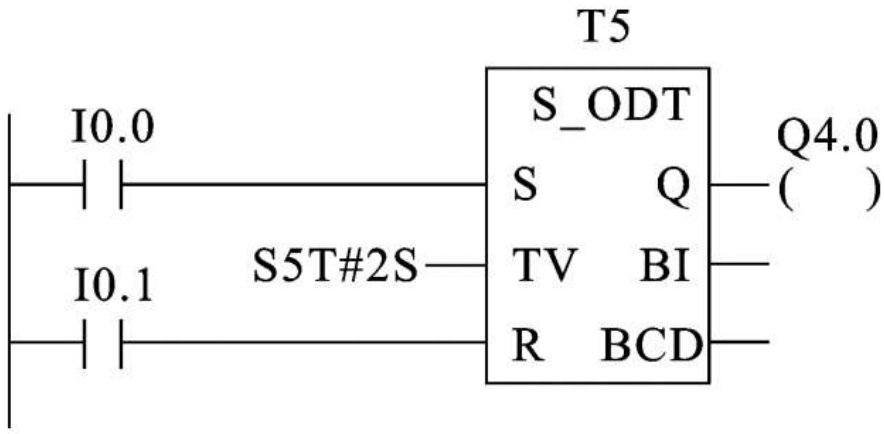


图4-20　接通延时定时器梯形图

图4-20所示梯形图对应的指令表如下。

```
A   I0.0
L   S5T#2S
SD  T5
A   I0.1
R   T5
A   T5
=   Q4.0
```

接通延时定时器时序如图4-21所示。

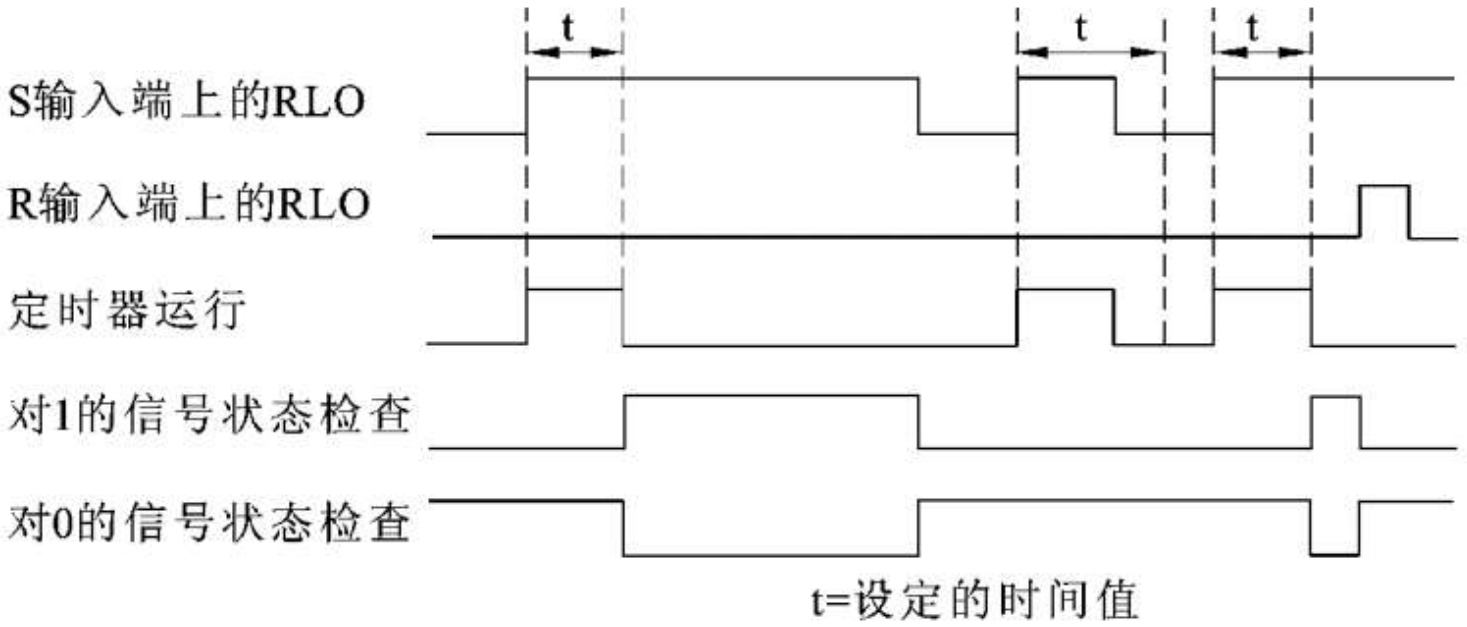


图4-21　接通延时定时器时序

4. 定时器线圈指令

定时器线圈指令见表4-12。

表4-12　定时器线圈指令

LAD 指令	STL 指令	功　能
T　no. ——(SP) 时间值	SP　T no.	启动脉冲定时器时间值的数据类型为：S5TIME
T　no. ——(SE) 时间值	SE　T no.	启动扩展脉冲定时器
T　no. ——(SD) 时间值	SD　T no.	启动接通延时定时器
T　no. ——(SS) 时间值	SS　T no.	启动保持型接通延时定时器
T　no. ——(SF) 时间值	SF　T no.	启动关断延时定时器

扩展脉冲定时器线圈指令梯形图如图4-22所示。

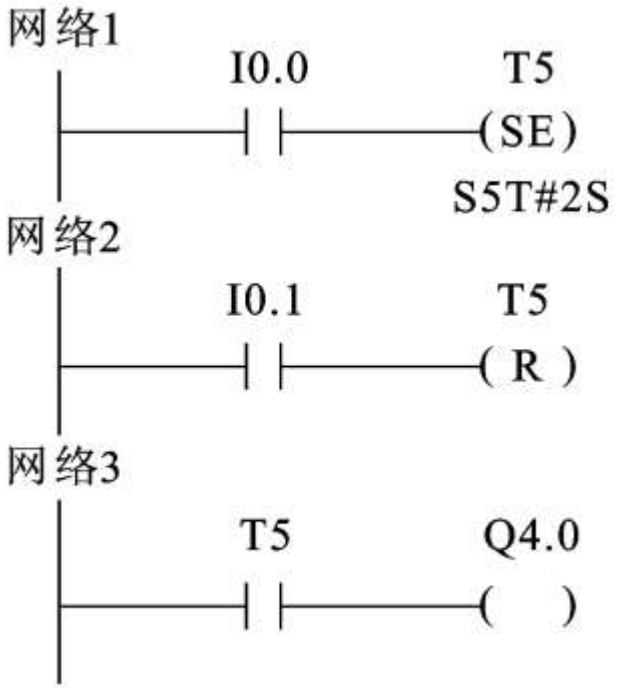


图4-22　扩展脉冲定时器线圈指令梯形图

图4-22所示梯形图对应的指令表如下。

```
A   I0.0
L   S5T#2S
SE  T5
A   I0.1
R   T5
A   T5
=   Q4.0
```

4.3.2 计数器指令

S7-300中的计数器用于对RLO正跳沿进行计数。计数器是由表示当前计数值的字及状态的位组成。S7-300中有三种计数器，即加计数器(S_CU)、减计数器 (S_CD)和可逆计数器(S_CUD)。

1.计数器组成

在CPU中保留了一块存储区作为计数器计数值的存储区，每个计数器占用两个字节，计数器字中的第0~11位表示计数值(二进制格式)，计数范围为0~999图4-23所示的是累加器1低字的内容计数值。

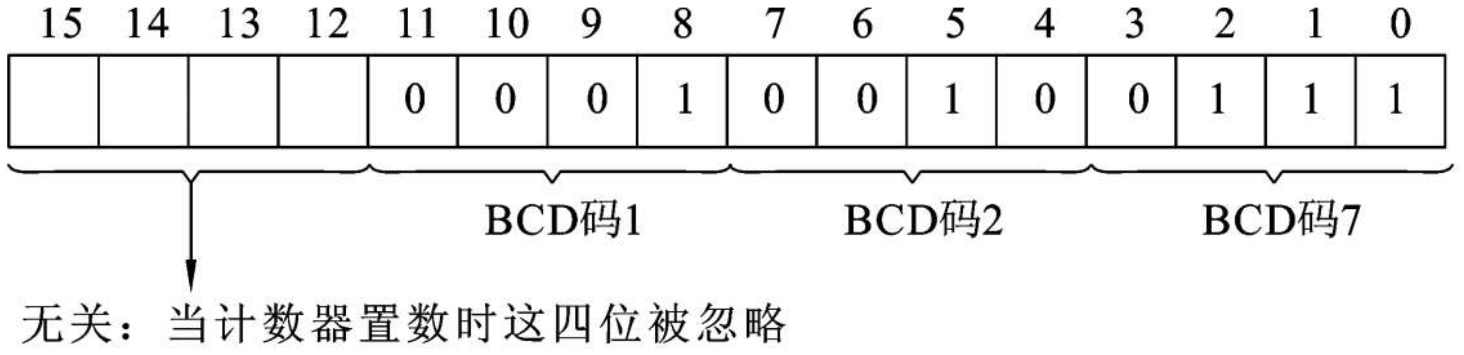
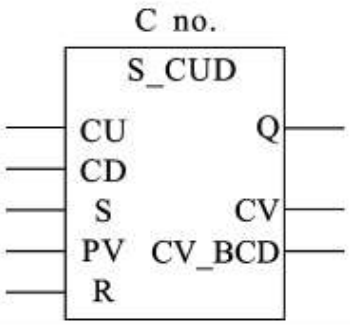
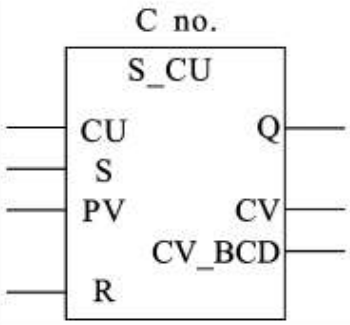
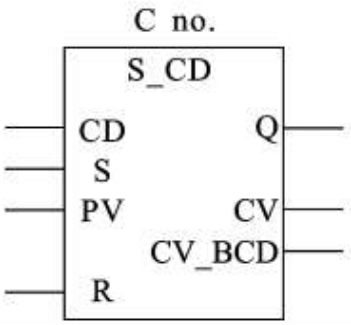


图4-23 累加器1低字的内容计数值（127）

2. 计数器梯形图方块指令

计数器梯形图方块指令见表4-13。

表4-13　计数器梯形图方块指令

可逆计数器		加计数器	减计数器
			
参数	数据类型	存储区	说明
no.	COUNTER	C	计数器标识号
CU	BOOL	I、Q、M、D、L	加计数输入
CD	BOOL	I、Q、M、D、L	减计数输入
S	BOOL	I、Q、M、D、L	计数器预置输入
PV	WORD	I、Q、M、D、L	计数初始值输入(BCD码)
R	BOOL	I、Q、M、D、L	复位输入端

续表

可逆计数器		加计数器	减计数器
Q	BOOL	Q、M、D、L	计数器状态输出
CV	WORD	Q、M、D、L	当前计数值输出(整数格式)
CV_BCD	WORD	Q、M、D、L	当前计数值输出(BCD 格式)

可逆计数器的梯形图方块应用如图4-24所示。

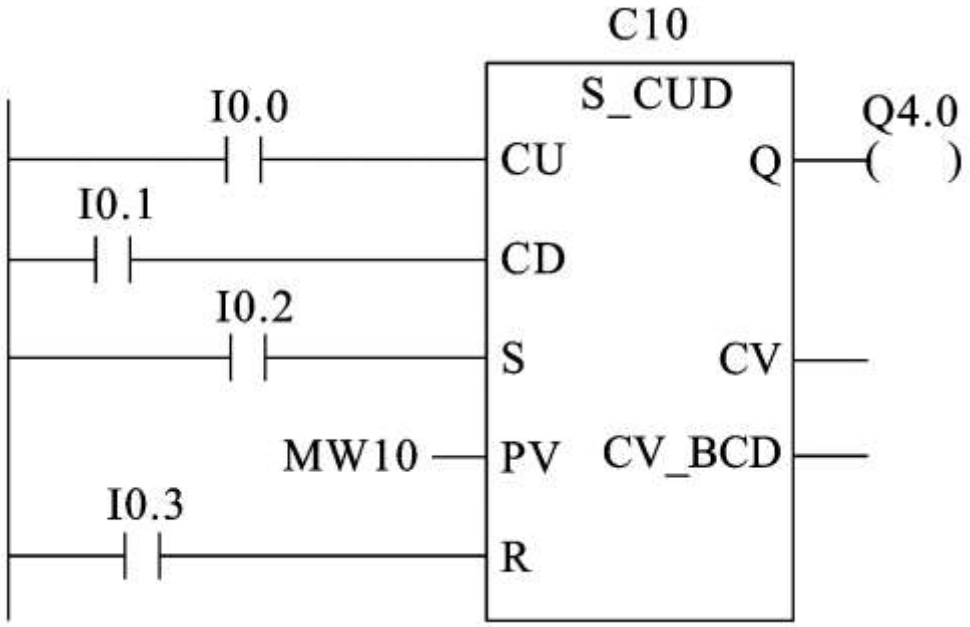


图4-24　可逆计数器梯形图方块应用

图4-24所示梯形图对应的指令表如下。

```
A    I0.0
CU   C10
A    I0.1
CD   C10
A    I0.2
L    MW10
S    C10
A    I0.3
R    C10
A    C10
=    Q4.0
```

3. 计数器线圈指令

计数线圈指令如表4-14所示。

表4-14　计数器线圈指令

LAD 指令	STL 指令	功　　能
C　no. ——(SC) <预置值>	S　C　no.	计数器置初始值
C　no. ——(CU)	CU　C　no.	加记数
C　no. ——(CD)	CD　C　no.	减记数
	FR　C　no.	允许计数器再启动

4.3.3 应用举例

1.脉冲发生器

用定时器构成一个脉冲发生器，使其产生如图4-25所示的脉冲时序，脉冲信号的周期为3s，脉冲宽度为1s. 图4-26所示为其梯形图及语句表程序。

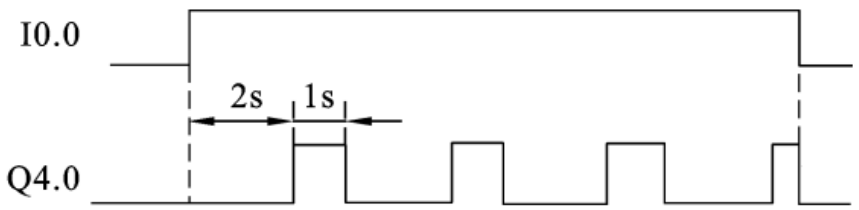


图4-25 脉冲发生器时序

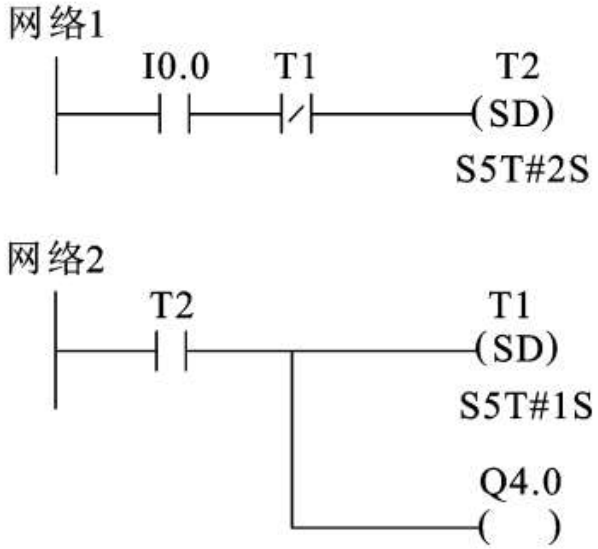


图4-26 脉冲发生器梯形图

图4-26所示梯形图对应的指令表如下。

```
网络 1
A    I0.0
AN   T1
L    S5T#2S
SD   T2
网络 2
A    T2
L    S5T#1S
SD   T1
A    T2
=    Q4.0
```

2. 频率监测器

频率监测器用于监测脉冲信号的频率，若其低于下限值，则指示灯亮，“确认”按键能使指示灯复位。为此，可以使用一个扩展脉冲定时器，该定时器当频率信号有一个上升沿时就启动一次定时器。如果超过了定时时间没有启动定时器，则表明两个脉冲之间的时间间隔太长，即频率太低了。图4-27所示为频率监测器时序。

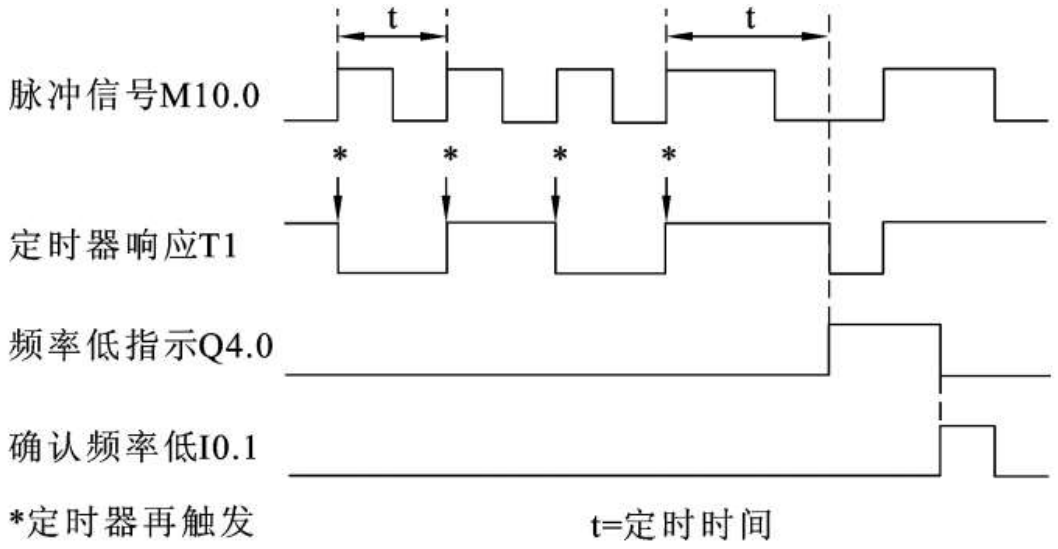


图4-27　频率监测器时序

在频率监测程序中，输入I0.0用于关闭监测器，I0.1用于确认频率低；输出Q4.0用于控制指示灯。定时器 T1的定时时间为2s ，即设置脉冲信号M10.0的频率监测下限为0.5Hz。图4-28所示为频率监测梯形图程序。

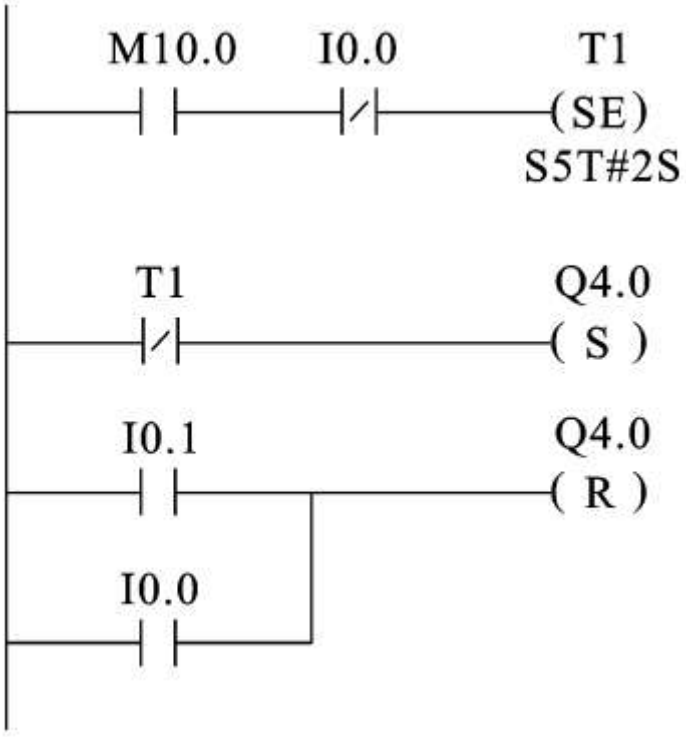


图4-28　频率监测梯形图程序

图4-28所示的梯形图对应的语句表如下。

```
A    M10.0
AN   I0.0
L    S5T#2S
SE   T1
AN   T1
S    Q4.0
O    I0.1
O    I0.0
R    Q4.0
```

4.4 数据处理指令

装入(L)和传送(T)指令可以在存储区之间，或者存储区与过程输入、输出之间交换数据。CPU执行这些指令不受逻辑操作结果RLO的影响。

L指令将源操作数装入累加器1中，而累加器原有的数据移入累加器2中，累加器2中原有的内容被覆盖。

T指令将累加器1中的内容写入目的存储区中，累加器的内容保持不变。

1.对累加器1的装入和传送指令

该指令的用法参见下面的例子。

```
L +5                  //将立即数+5 装入累加器1中
L MW 10               //将MW10中的值装入累加器1中
L IB[DID 8]           //将由数据双字DID8指出的输入字节装入累加器1中
T MW 20               //将累加器1中的内容传送给存储字MW20
T MW[AR1,P#10.0]      //将累加器1中的内容传送给由地址寄存器1加偏移量
                      //确定的存储字
```

2.读取或传送状态字

该指令的用法参见下面的例子。

```
L STW    //将状态字中0~8位装入累加器1中，累加器中9~31位被清0
T STW    //将累加器1中的内容传送到状态字中
```

3.装入时间值或计数值

该指令的用法参见下面的例子。

```
 L   T1    //将定时器T1中二进制格式的时间值直接装入累加器1的低字中
 LC  T1    //将定时器T1的时间值和时基以BCD码装入累加器1的低字中
 L   C1    //将计数器C1中二进制格式的计数值直接装入累加器1的低字中
 LC  C1    //将计数器C1中的计数值以BCD码格式装入累加器1的低字中
```

4.地址寄存器的装入和传送

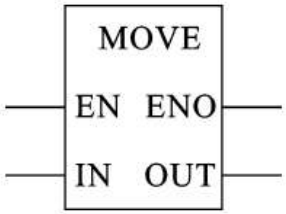
对于地址寄存器,可以不经过累加器1而直接将操作数装入或传送,或者将两个地址寄存器的内容直接交换. 该指令的用法参见下面的例子。

```
LAR1   P#I0.0     //将输入位10.0的地址指针装入AR1
LAR2   P#0.0      //将二进制数 2#0000 0000 0000 0000 0000
                  //0000 0000 0000 装入AR2
LAR1   P#Start    //将符号名为Start的存储器的地址指针装入AR1
LAR1   AR2        //将AR2的内容装入AR1
LAR1   DBD 20     //将数据双字DBD20的内容装入AR1
TAR1   AR2        //将AR1的内容传送至AR2
TAR2              //将AR2的内容传送至累加器1
TAR1   MD 20      //将AR1的内容传送至存储器双字MD20
CAR               //交换AR1和AR2的内容
```

5. 梯形图方块传送指令

梯形方块传送指令的具体参数见表4-15。

表4-15　梯形图方块传送指令

LAD方块	参　数	数据类型	存储区	说　明
	EN	BOOL	I、Q、M、D、L	允许输入
	ENO	BOOL	Q、M、D、L	允许输出
	IN	8、16、32位长的所有数据类型	I、Q、M、D、L	源数值　（可为常数）
	OUT	8、16、32位长的所有数据类型	Q、M、D、L	目的操作数

MOVE方块指令的实际应用如图4-29所示。

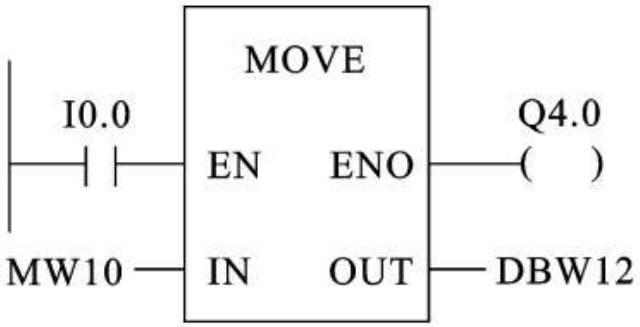


图4-29　使用 MOVE方块指令

图4-29所示梯形图对应的语句表如下。

```
A     I0.0
JNB   _0001
L     MW10
T     DBW12
SET           //使RLO为1
SAVE          //使BR为1
CLR
_0001: A BR
=     Q4.0
```

4.4.2 转换指令

转换指令首先将源数据按照规定的格式读入累加器，然后在累加器中对数据进行类型转换，最后再将转换的结果传送到目的地址。能够实现的转换操作有以下几种。

（1）BCD码和整数及长整数间的转换(BTI, ITB, ITD, BTD, DTB, DTR)。

（2）实数和长整数间的转换(RND, TRUNC, RND＋, RND-)。

（3）数的取反、取负等(INVI, INVD, NEGI, NEGD, NEGR)。

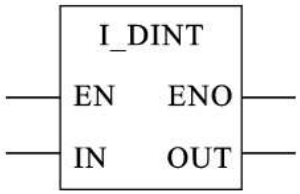
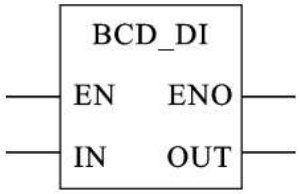
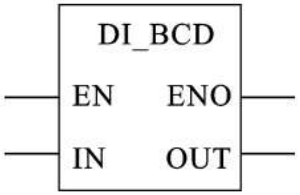
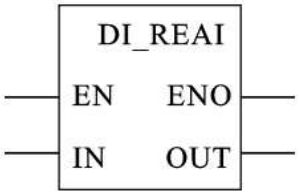
1. BCD和整数间的转换

BCD和整数间的转换指令见表4-16。

表4-16　BCD和整数间的转换

LAD 方块	STL 指令	参数	数据类型	存　储　区	说　明
BCD_I EN　ENO IN　OUT	BTI 将 3 位 BCD 码转换为 16 位整数	EN	BOOL	I、Q、M、D、L	使能输入
		ENO	BOOL	Q、M、D、L	使能输出
		IN	WORD	I、Q、M、D、L	BCD 码
		OUT	INT	Q、M、D、L	整数
I_BCD EN　ENO IN　OUT	ITB 将 16 位整数转换为 3 位 BCD 码	EN	BOOL	I、Q、M、D、L	使能输入
		ENO	BOOL	Q、M、D、L	使能输出
		IN	INT	I、Q、M、D、L	整数
		OUT	WORD	Q、M、D、L	BCD 码

续表

LAD 方块	STL 指令	参数	数据类型	存　储　区	说　明
	ITD 将 16 位整数转换为 32 位整数	EN	BOOL	I、Q、M、D、L	使能输入
		ENO	BOOL	Q、M、D、L	使能输出
		IN	INT	I、Q、M、D、L	要转换的值
		OUT	DINT	Q、M、D、L	转换的结果值
	BTD 将 7 位 BCD 码数转换为 32 位整数	EN	BOOL	I、Q、M、D、L	使能输入
		ENO	BOOL	Q、M、D、L	使能输出
		IN	DWORD	I、Q、M、D、L	BCD 码
		OUT	DINT	Q、M、D、L	由 BCD 码转换成的双整数
	DTB 将 32 位整数转换为 7 位 BCD 码数	EN	BOOL	I、Q、M、D、L	使能输入
		ENO	BOOL	Q、M、D、L	使能输出
		IN	DINT	I、Q、M、D、L	双整数
		OUT	DWORD	Q、M、D、L	BCD 码的结果
	DTR 将 32 位整数转换为 32 位实数	EN	BOOL	I、Q、M、D、L	使能输入
		ENO	BOOL	Q、M、D、L	使能输出
		IN	DINT	I、Q、M、D、L	要转换的值
		OUT	REAL	Q、M、D、L	转换的结果值

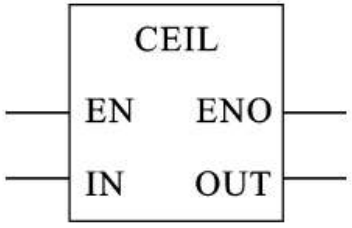
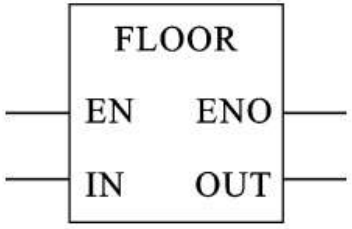
2. 实数和长整数间的转换

实数和长整数间的转换见表4-17。因为实数的数值范围远大于32位整数，所以有的实数不能成功地转换为32位整数。如果被转换的实数格式非法或超出了32位整数的表示范围，则得不到有效结果，而且状态字中的OV位和OS位被置1。

表4-17　实数和长整数间的转换

LAD方块	STL指令	参数	数据类型	存储区	说明
ROUND EN　ENO IN　OUT	RND 将实数化整为最接近的整数	EN	BOOL	I、Q、M、D、L	使能输入
		ENO	BOOL	Q、M、D、L	使能输出
		IN	REAL	I、Q、M、D、L	要舍入的值
		OUT	DINT	Q、M、D、L	舍入后的结果
TRUNC EN　ENO IN　OUT	TRUNC 取实数的整数部分(截尾取整)	EN	BOOL	I、Q、M、D、L	使能输入
		ENO	BOOL	Q、M、D、L	使能输出
		IN	REAL	I、Q、M、D、L	要取整的值
		OUT	DINT	Q、M、D、L	IN的整数部分

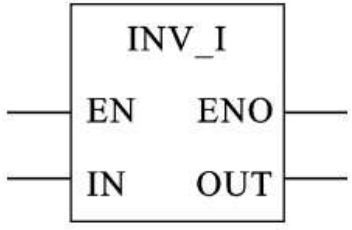
续表

LAD 方块	STL 指令	参数	数据类型	存 储 区	说　　明
	RND+ 将实数化整为大于或等于该实数的最小整数	EN	BOOL	I、Q、M、D、L	使能输入
		ENO	BOOL	Q、M、D、L	使能输出
		IN	REAL	I、Q、M、D、L	要取整的值
		OUT	DINT	Q、M、D、L	上取整后的结果
	RND− 将实数化整为小于或等于该实数的最大整数	EN	BOOL	I、Q、M、D、L	使能输入
		ENO	BOOL	Q、M、D、L	使能输出
		IN	REAL	I、Q、M、D、L	要取整的值
		OUT	DINT	Q、M、D、L	下取整后的结果

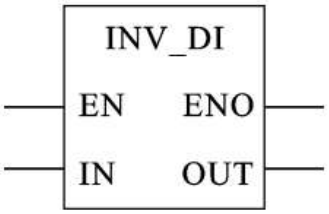
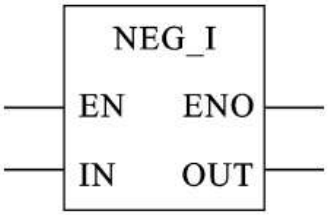
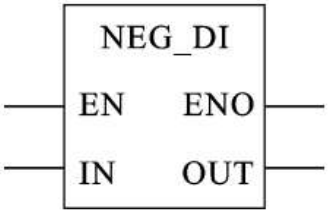
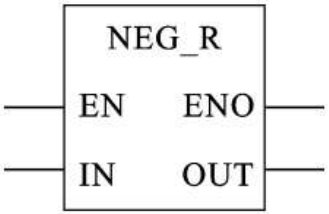
3. 数的取反、取负

数的取反、取负见表4-18。

表4-18 数的取反、取负

LAD 方块	STL 指令	参数	数据类型	存 储 区	说　　明
	INVI 对 16 位整数求反码	EN	BOOL	I、Q、M、D、L	使能输入
		ENO	BOOL	Q、M、D、L	使能输出
		IN	INT	I、Q、M、D、L	输入值
		OUT	INT	Q、M、D、L	整数的二进制反码

续表

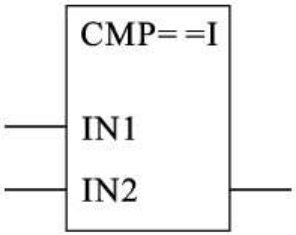
LAD 方块	STL 指令	参数	数据类型	存 储 区	说　　明
	INVD 对 32 位整数求反码	EN	BOOL	I、Q、M、D、L	使能输入
		ENO	BOOL	Q、M、D、L	使能输出
		IN	DINT	I、Q、M、D、L	输入值
		OUT	DINT	Q、M、D、L	双整数的二进制反码
	NEGI 对 16 位整数求补码(取反码再加 1),相当于乘 −1	EN	BOOL	I、Q、M、D、L	使能输入
		ENO	BOOL	Q、M、D、L	使能输出
		IN	INT	I、Q、M、D、L	输入值
		OUT	INT	Q、M、D、L	整数的二进制补码
	NEGD 对 32 位整数求补码	EN	BOOL	I、Q、M、D、L	使能输入
		ENO	BOOL	Q、M、D、L	使能输出
		IN	DINT	I、Q、M、D、L	输入值
		OUT	DINT	Q、M、D、L	双整数的二进制补码
	NEGR 对 32 位实数的符号位求反码	EN	BOOL	I、Q、M、D、L	使能输入
		ENO	BOOL	Q、M、D、L	使能输出
		IN	REAL	I、Q、M、D、L	输入值
		OUT	REAL	Q、M、D、L	对输入值求反的结果

4.4.3 比较指令

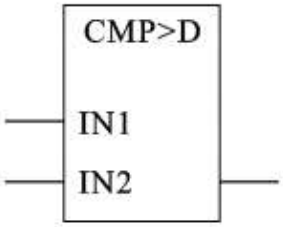
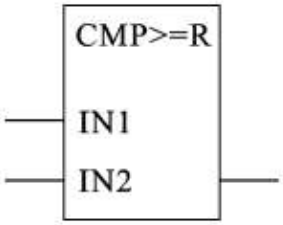
1. 比较指令

比较指令见表4-19。

表4-19　比较指令

LAD 方块	STL 指令	方块上部的符号	比较类型
	==I	==	IN1 等于 IN2
	<>I	<>	IN1 不等于 IN2
	>I	>	IN1 大于 IN2
	<I	<	IN1 小于 IN2
	>=I	>=	IN1 大于等于 IN2
	<=I	<=	IN1 小于等于 IN2

续表

LAD 方块	STL 指令	方块上部的符号	比较类型
	==D	==	IN1 等于 IN2
	<>D	<>	IN1 不等于 IN2
	>D	>	IN1 大于 IN2
	<D	<	IN1 小于 IN2
	>=D	>=	IN1 大于等于 IN2
	<=D	<=	IN1 小于等于 IN2
	==R	==	IN1 等于 IN2
	<>R	<>	IN1 不等于 IN2
	>R	>	IN1 大于 IN2
	<R	<	IN1 小于 IN2
	>=R	>=	IN1 大于等于 IN2
	<=R	<=	IN1 小于等于 IN2

比较指令的参数见表4-20。

表4-20　比较指令的参数

参数	数据类型	存储区	说　　明
IN1	INT DINT REAL	I、Q、M、D、L	参与比较的数可以是整数、长整数、实数，但数据类型必须一致
IN2	INT DINT REAL	I、Q、M、D、L	

整数比较的梯形图如图4-30所示。

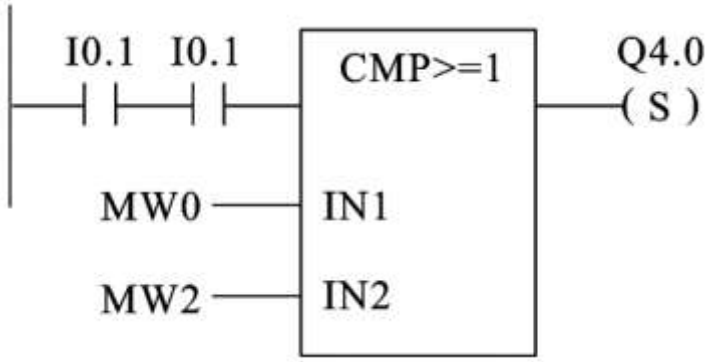
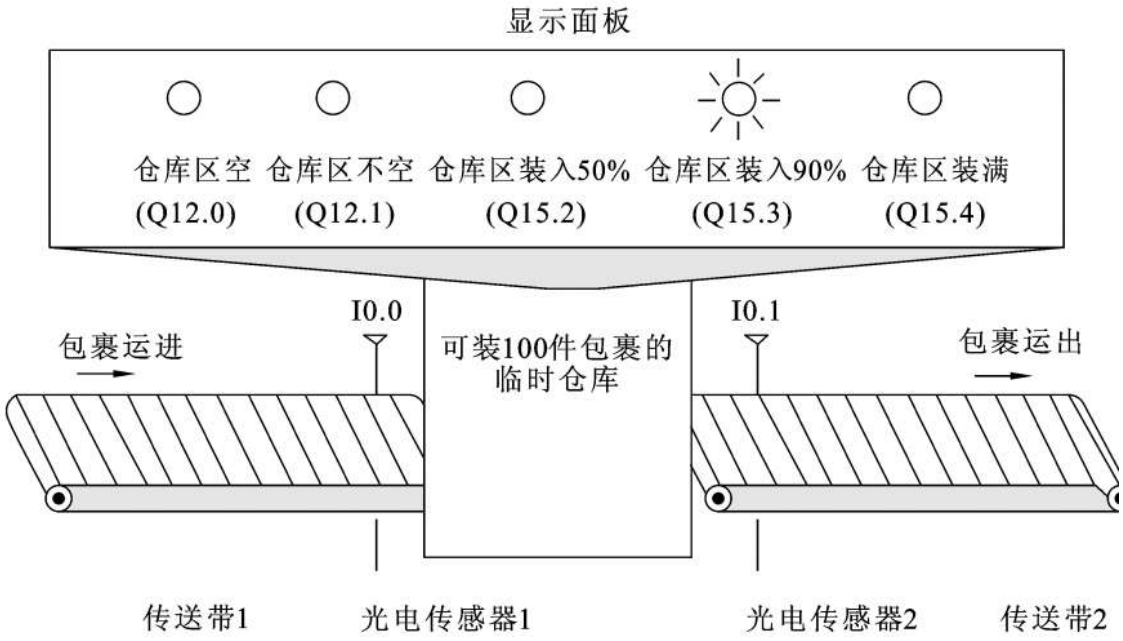


图4-30所示梯形图对应的指令表如下。

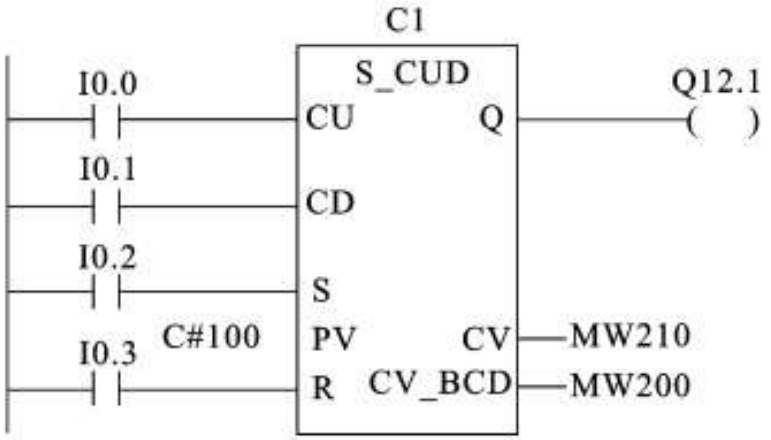
```
A   I0.0
A   I0.1
A(
L   MW0
L   MW2
=>I
)
S   Q4.0
```

2. 比较指令的应用实例

图4-31所示为包括两台传送带的系统，在两台传送带之间有一个仓库区。传送带1将包裹运送至临时仓库区。传送带1在靠近仓库区的一端安装的光电传感器用于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在靠近库区的一端安装的光电传感器用于确定已有多少包裹从库区运送至装货场。含5个指示灯的显示盘则用于显示临时仓库区的占用程度。图4-32给出了启动显示盘上指示灯的梯形逻辑程序。



程序段1：MW200中保存计数器C1当前值的BCD码，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），则“仓库区装入50%”

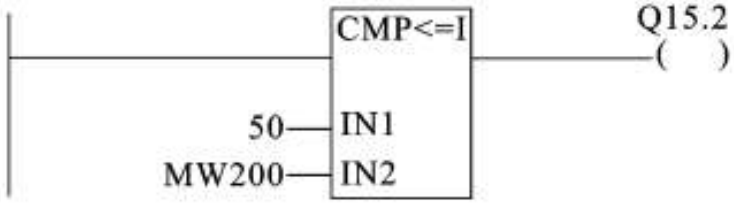
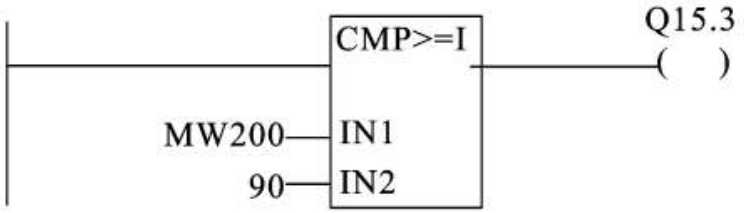
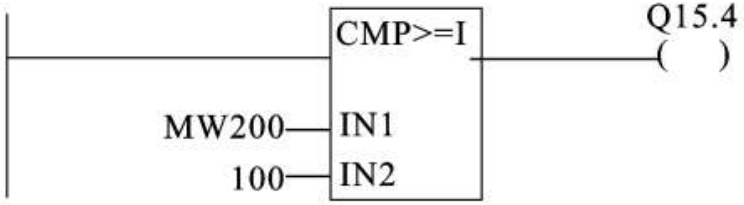


图4-32　启动显示盘上指示灯的梯形逻辑

程序段4：如果计数器值大于等于90，则“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，则“仓库区装满”指示灯亮，用输出Q15.4将传送带1联锁



续图4-32

与图4-32对应的语句表程序如下。

```
A    I12.0
CU   C1
A    I12.1
CD   C1
AN   C1
=    Q12.0
A    C1
=    Q12.1
L    +50
L    C1
<=I
=    Q15.2
L    +90
>=I
=    Q15.3
L    C1
L    +100
>=I
=    Q15.4
```

4.4.4 移位和循环移位指令

移位指令将输入IN中的内容向左或向右逐位移动，移动次数由输入值N提供的数值来确定。移位后空出的位用0或符号位(0代表正，1代表负)填充，被移动的最后一位保存在状态字中的CC1里，CC0和OV被复位为0，可以使用条件跳转指令对CC1进行判断。循环移位指令与一般移位指令的差别是：循环移位指令的空位是用从IN中移出的位来填充的。

1.无符号数移位指令

无符号数的移位指令见表4-21，无符号数移位指令的具体操作见图4-33。

表4-21　无符号数移位指令

LAD 方块	STL 指令	LAD 方块	STL 指令
SHL_W EN ENO IN OUT N	SLW 将 IN 中的字逐位 左移，空出位填以 0	SHL_DW EN ENO IN OUT N	SLD 将 IN 中的双字逐位 左移，空出位填以 0
SHR_W EN ENO IN OUT N	SRW 将 IN 中的字逐位 右移，空出位填以 0	SHR_DW EN ENO IN OUT N	SRD 将 IN 中的双字逐位 右移，空出位填以 0

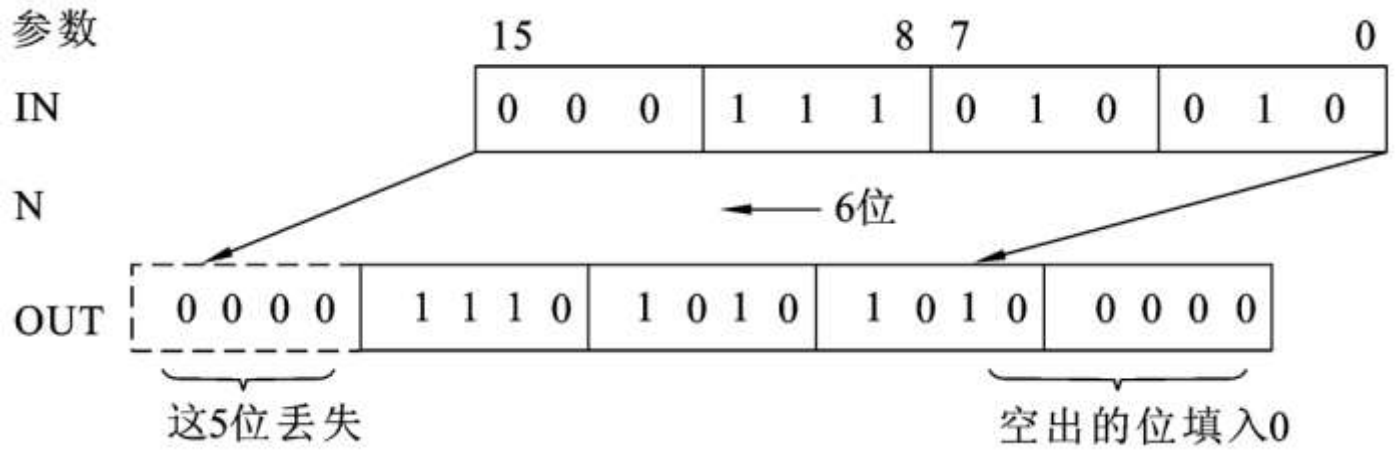
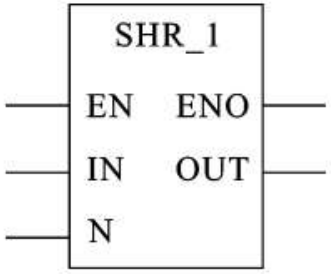
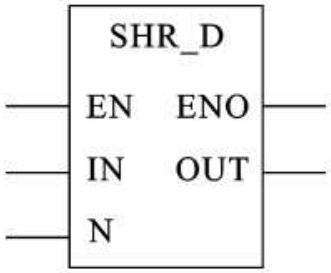


图4-33　输入I N 左移5位

2. 有符号数移位指令

有符号数的移位指令见表4-22，其具体操作见图4-34。

表4-22　有符号数移位指令

LAD 方块	STL 指令	参数	数据类型	存储区	说　明
	SSI 将 IN 中的字逐位右移，空出位填以符号位	EN	BOOL	I、Q、M、D、L	使能输入
		ENO	BOOL	Q、M、D、L	使能输出
		IN	WORD	I、Q、M、D、L	要移位值
		N	WORD	Q、M、D、L	要移位数
		OUT	WORD	I、Q、M、D、L	操作结果
	SSD 将 IN 中的双字逐位右移，空出位填以符号位	EN	BOOL	I、Q、M、D、L	使能输入
		ENO	BOOL	Q、M、D、L	使能输出
		IN	DINT	I、Q、M、D、L	要移的值
		N	WORD	I、Q、M、D、L	要移位数
		OUT	DINT	Q、M、D、L	操作结果

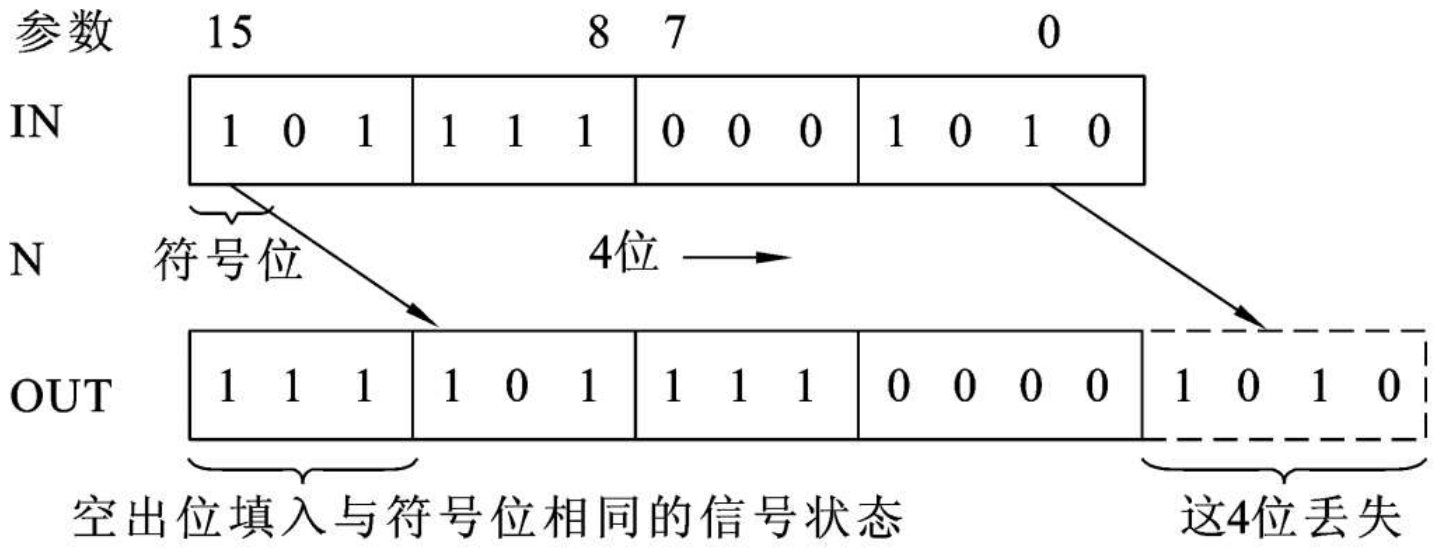
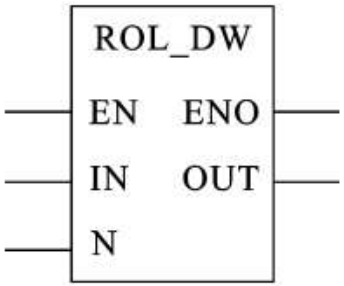
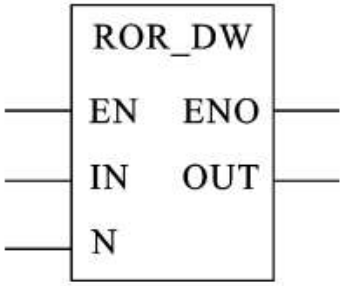


图4-34　输入IN带符号右移４位

3. 循环移位指令

循环移位指令见表4-23，其具体操作见图4-35与图4-36。

表4-23　循环移位指令

LAD 方块	STL 指令	参数	数据类型	存储区	说　明
	RLD 将 IN 中的双字逐位左移，空出位填以移出的位	EN	BOOL	I、Q、M、D、L	双字循环左移（ROL_DW）：有效移位位数是 0～32
		ENO	BOOL	Q、M、D、L	
		IN	WORD	I、Q、M、D、L	
		N	WORD	I、Q、M、D、L	
		OUT	WORD	Q、M、D、L	
	RRD 将 IN 中的双字逐位右移，空出位填以移出的位	EN	BOOL	I、Q、M、D、L	双字循环右移（ROR_DW）：有效移位位数是 0～32
		ENO	BOOL	Q、M、D、L	
		IN	WORD	I、Q、M、D、L	
		N	WORD	I、Q、M、D、L	
		OUT	WORD	Q、M、D、L	

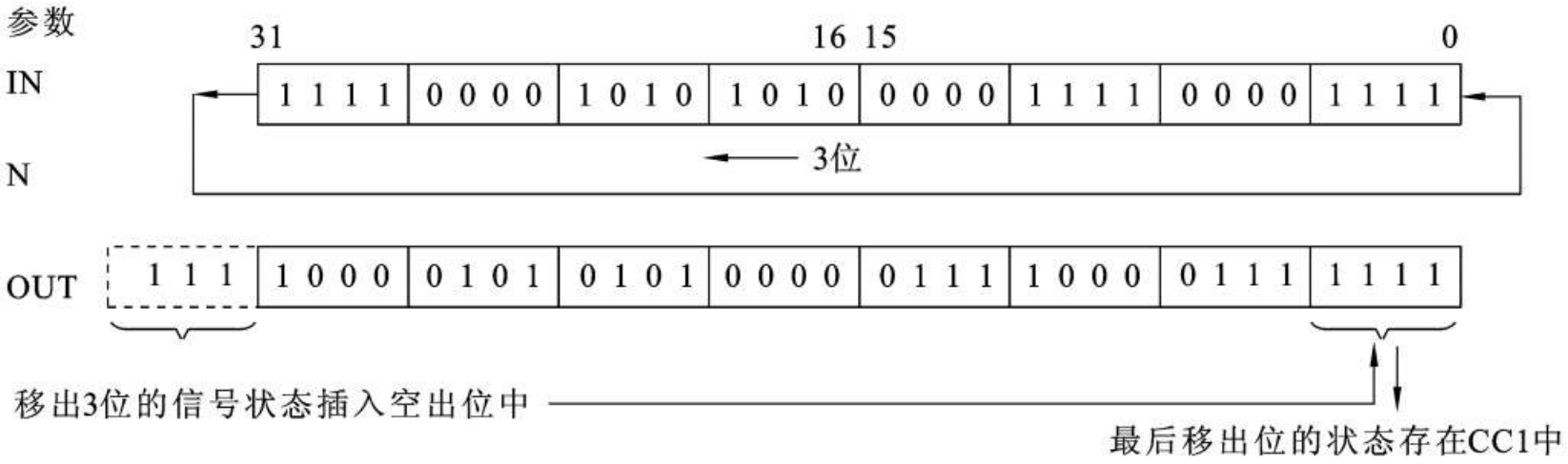


图4-35　输入IN循环左移３位

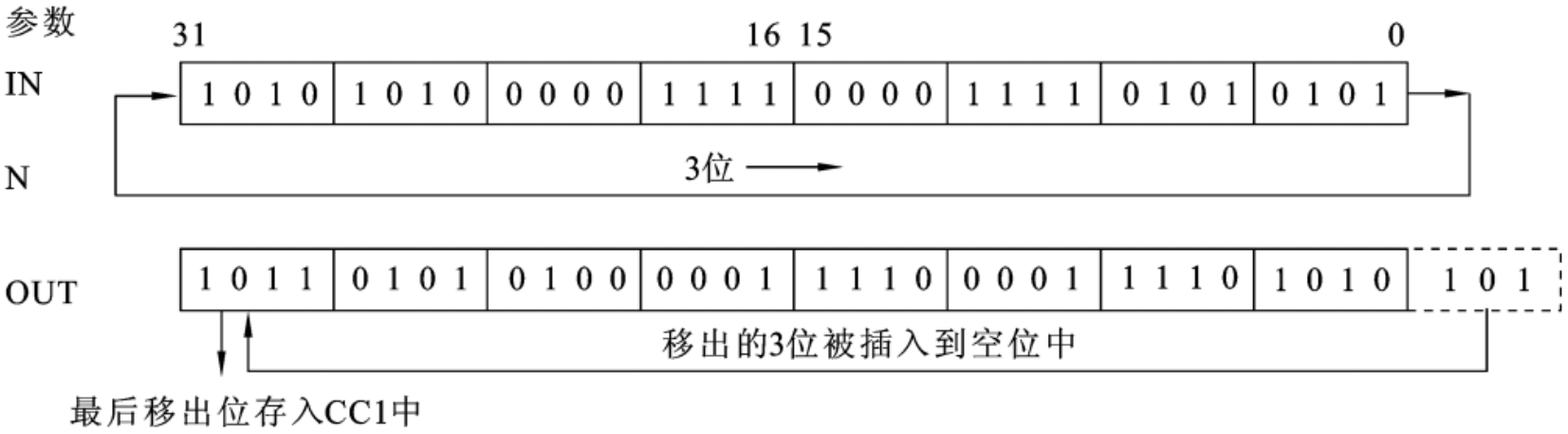


图4-36　输入IN循环右移３位

4.4.5 累加器操作和地址寄存器指令

1. 累加器操作指令

累加器操作指令见表4-24，其具体操作见图4-37。

表4-24　累加器操作指令

指　　令	指 令 功 能
TAK	累加器 1 和累加器 2 的内容互换
PUSH	把累加器 1 的内容移入累加器 2，累加器 2 中原内容被删除
POP	把累加器 2 的内容移入累加器 1，累加器 1 中原内容被删除
INC	将累加器 1 中低字的低字节内容加上指令中给出的常数，常数范围为 0～255；指令的执行是无条件的，其结果不影响状态字
DEC	将累加器 1 中低字的低字节内容减去指令中给出的常数，常数范围为 0～255；指令的执行是无条件的，其结果不影响状态字
CAW	交换累加器 1 低字中的字节顺序
CAD	交换累加器 1 中的字节顺序

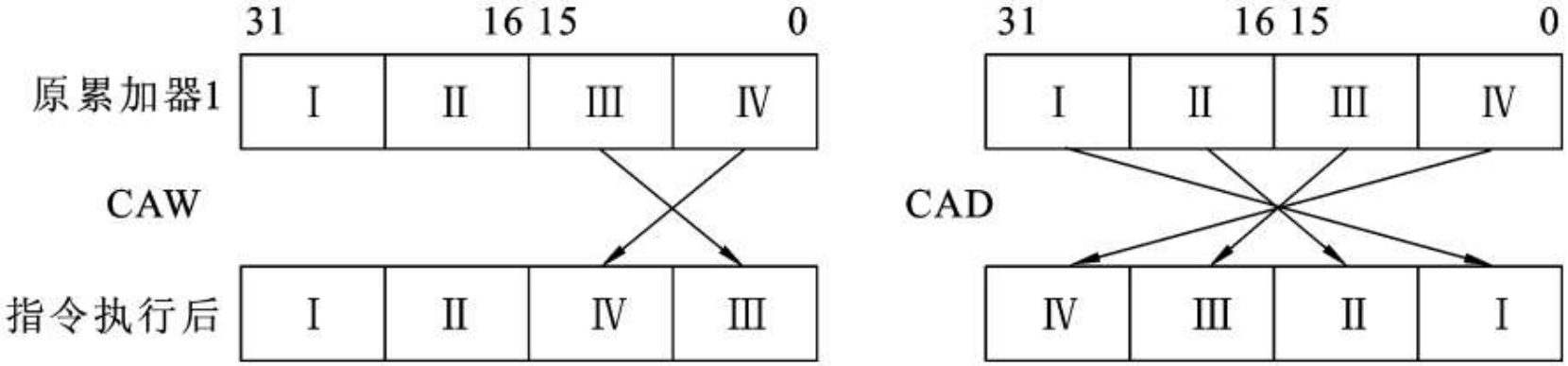


图4-37　CAW、CAD指令执行时累加器1的变化

下面的例子说明了在有条件触发的程序中INC指令是如何工作的。本例以存储字节MB10作为循环次数计数暂存器，使用INC指令修正循环次数，循环体中的程序连续执行5次。

```
LOOP:L MB10          //循环体开始，装载存储字节至累加器1
INC    1             //循环计数器加1
T      MB10          //保存循环次数
L      B#16#5
<＝I
JC     LOOP          //如果循环次数小于等于5次，则继续循环
L      1             //循环体外的操作，为下次循环做准备
T      MB10
```

2. 地址寄存器指令

地址寄存器指令见表4-25。

表4-25　地址寄存器指令

指　令	操作数	指 令 功 能
＋AR1		将累加器 1 低字的内容加至地址寄存器 1
＋AR2		将累加器 1 低字的内容加至地址寄存器 2
＋AR1	P＃Byte. Bit	将一个指针常数加至地址寄存器 1
＋AR2	P＃Byte. Bit	将一个指针常数加至地址寄存器 2

在使用地址寄存器加指令时，应保证累加器1或指针常数的正确格式。下面的例子说明了把累加器1的内容加至地址寄存器指令的用法。在加入累加器的内容之前应先为累加器1装入一个指针常数。

```
L   P#250.7     //装载指针常数(250.7)至累加器1
+   AR1         //把250.7加至地址寄存器1
+   AR2         //把250.7加至地址寄存器2
```

3. 数据块指令

数据块指令见表4-26。

表4-26　数据块指令

LAD 指令	STL 指令	说　　明
——(OPEN)	OPEN	打开一个数据块作为共享数据块或背景数据块
	CAD	交换共享数据块和背景数据块
	DBLG	将共享数据块的长度(字节数)装入累加器 1
	CBNO	将共享数据块的块号装入累加器 1
	DILG	将背景数据块的长度(字节数)装入累加器 1
	DINO	将背景数据块的块号装入累加器 1

使用以上指令必须先打开一个数据块，才能使用其他的数据块指令。下面的例子说明，如何使用共享数据块装入指令。当数据块的长度大于50B时,程序跳转到ERR标号处。ERR标号可以调用功能块FC10做适当的处理。

```
OPEN DB40          //打开共享数据块DB40
L    DBLG          //将共享数据块的长度装入累加器1
L    +50           //将整数50装入累加器1，累加器1原内容移入累加器2
> =I               //比较
JC   ERR           //如果数据块长度大于50，则跳转至ERR标号处
A    I0.0
BEU                //无条件结束当前块
ERR: CALL FC10     //对于块长度大于50的情况，调用FC10做相应的处理
```

4. 显示和空操作指令

显示和空操作指令见表4-27。

表4-27　显示和空操作指令

指　　令	指 令 功 能
BLD	该指令控制编程器显示程序的形式，执行程序时不产生任何影响
NOP 0	空操作 0：不进行任何操作
NOP 1	空操作 1：不进行任何操作

4.5 数学运算指令

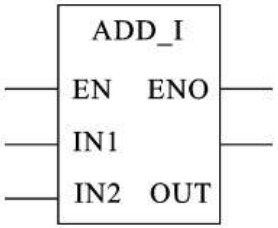
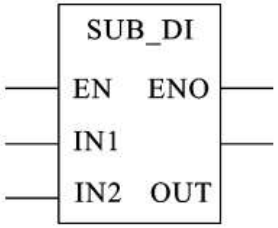
4.5.1 算术运算指令

在STEP7中可以对整数、长整数和实数进行加、减、乘、除等算术运算。算术运算指令在累加器1和2中进行，累加器2的值作为被减数或被除数。算术运算的结果保存在累加器1中，累加器1中原有的值被运算结果覆盖，累加器2中的值保持不变。算术运算指令对状态字的某些位将产生影响，如C C1、CC0、OV和OS等。可以利用位操作指令或条件跳转指令对状态字中的标志位进行判断操作。

1. 整数算术运算

整数算术运算见表4-28，其具体操作方法见图4-38。

表4-28　整数算术运算

LAD 方块	STL 指令	方块上部的符号	说　　明
	+I	ADD_I	将 IN1 和 IN2 中的 16 位整数相加，结果保存在 OUT 中
	−I	SUB_I	IN1 整数减去 IN2，结果保存在 OUT 中
	* I	MUL_I	IN1 和 IN2 整数相乘，结果以 32 位整数保存在 OUT 中
	/I	DIV_I	将 IN1 中 16 位整数除以 IN2 中的 16 位整数，商保存在 OUT 中
	+D	ADD_DI	IN1 和 IN2 的双字相加，保存在 OUT 中
	−D	SUB_DI	IN1 减去 IN2，结果保存在 OUT 中
	* D	MUL_DI	将 IN1 和 IN2 中的 32 位整数相乘，结果保存在 OUT 中
	/D	DIV_DI	将 IN1 中 32 位整数除以 IN2 中的 32 位整数，商保存在 OUT 中
	MOD	MOD	将 IN1 中 32 位整数除以 IN2 中的 32 位整数，余数保存在 OUT 中

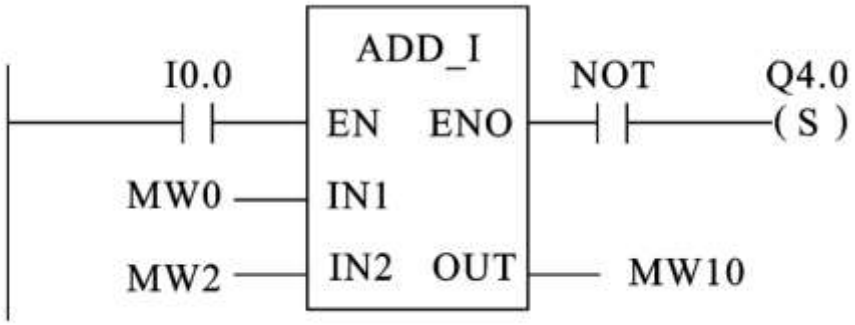


图4-38　使用整数加法方块指令

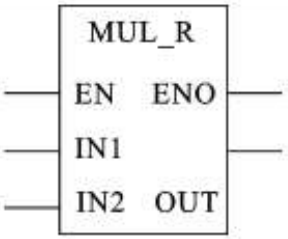
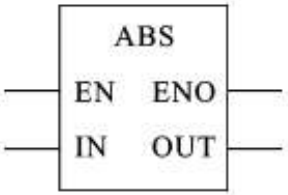
下面是与图4-38所示梯形图完全对应的语句表程序。

```
A(
A       I0.0
JNB     _0001
L       MW0
L       MW2
+ I
T       MW10
AN      OV      //若OV为1，则RLO为0，否则RLO为1
SAVE            //使BR＝RLO
CLR
_0001: A BR
)
NOT
S       Q4.0
```

2. 实数算术运算

实数算术运算见表4-29。

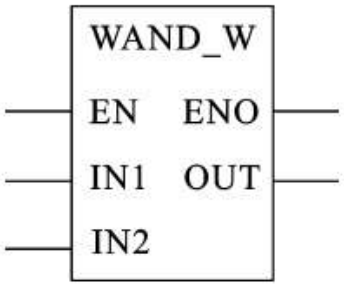
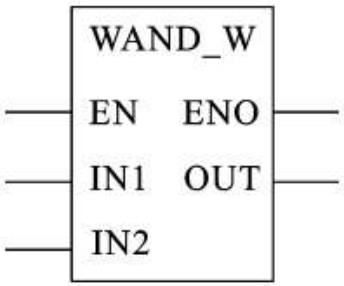
表4-29　实数算术运算

LAD方块	STL指令	方块上部的符号
	+R	ADD_R
	−R	SUB_R
	* R	MUL_R
	/R	DIV_R
	ABS	ABS
	SQR	SQR
	SQRT	SQRT
	LN	LN
	EXP	EXP
	SIN	SIN
	COS	COS
	ASIN	ASIN
	ACOS	ACOS
	TAN	TAN
	ATAN	ATAN

4.5.2 字逻辑运算指令

字逻辑运算指令见表4-30。

表4-30　字逻辑运算指令

LAD方块	STL指令	方块上部的符号
	AW	WAND_W
	OW	WOR_W
	XOW	WXOR_W
	AD	WAND_DW
	OD	WOR_DW
	XOD	WXOR_DW

4.5.3 数据运算指令应用举例

1.运用算术运算指令完成方程式运算

运用算术运算指令完成下面的方程式运算，其梯形逻辑图如图4-39所示。

MW4＝((IW0＋DBW3)×15)/MW0

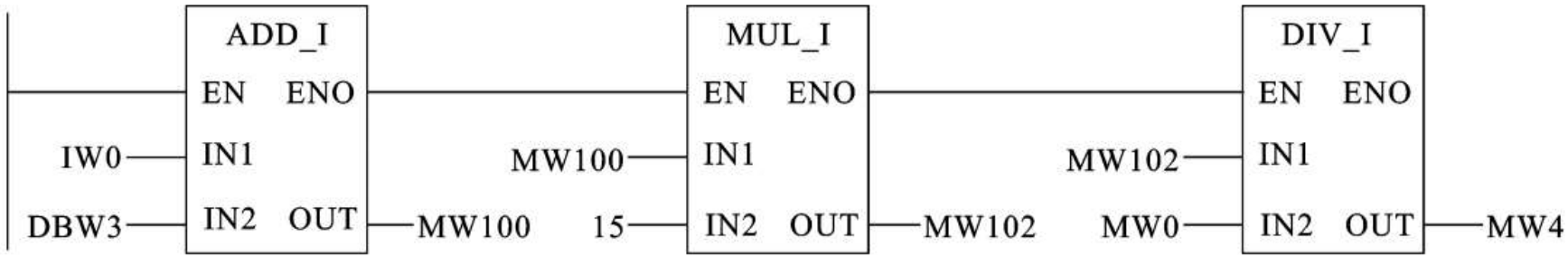


图4-39　算术运算指令梯形逻辑图

实现相同运算的语句表程序如下。

```
L   IW0   //将输入字IW0的内容装入累加器1
L   DBW3  //将DBW3的内容装入累加器1,累加器1原内容装入累加器2
+I        //累加器2与累加器1相加,结果存在累加器1中
L   +15   //将常数15装入累加器1,累加器1原内容(和)装入累加器2
*I        //累加器2与累加器1相乘,结果为长整数放在累加器1中
L   MW0   //将存储双字MD0的内容装入累加器1,累加器1原内容装入累加器2
/I        //累加器2除累加器1,结果的整数部分存在累加器1中
T   MW4   //将运算结果传送至存储双字MD4
```

完成相同运算功能的梯形图程序和语句表程序各有优缺点：梯形图程序直观易读；语句表程序简洁，而且使用中间结果存储器较少。

2.使用位逻辑运算指令实现对信号的跳变沿检测

对输入位I12.0~I13.7进行跳变沿检测，并将正跳沿的检测结果存入存储位M14.0~M15.7的对应位中(1表示有跳变，0表示无跳变),负跳沿的结果存入M16.0~M17.7中。为此，在检测正跳沿时，使用存储位M10.0~M11.7存储对应输入位在前一个扫描周期时的状态;在检测负跳沿时使用存储位M12.0~M13.7。相应的语句表程序如下。

```
网络10:正跳沿检测
L  MW10   //将输入位的上一个周期状态装入累加器1的低字中
L  IW12   //将输入位的当前状态装入累加器1的低字中,上一个周期状态
          //被移入累加器2中
T  MW10   //保存当前状态,供下一个扫描周期使用
XOW       //异或运算后,当前状态与以前不同的位在累加器1的低字中被
          //置为1
L  IW12   //重新装入当前状态,将累加器1中原内容移入累加器2
AW        //与运算后,当前状态为0的位被清0(负跳变被屏蔽)
T  MW14   //将正跳变检测结果送入MW14中
```

网络 11:负跳沿检测

```
L   MW12   //将输入位的上一个周期状态写入累加器1的低字中
L   IW12   //将输入位的当前状态装入累加器1的低字中,上一个周
           //期状态被移入累加器2
T   MW12   //保存当前状态,供下一个扫描周期使用
XOW        //异或运算后,当前状态与以前不同的位在累加器1的低
           //字中被置为1
L   IW12   //重新装入当前状态,将累加器1中原内容移入累加器2
INVI       //将当前状态取反
AW         //与运算后,当前状态为1的位(上条指令中已被取反)被
           //清0(正跳变被屏蔽)
T   MW16   //将负跳变检测结果送入MW16
```

4.6 控制指令

4.6.1 逻辑控制指令

逻辑控制指令是指逻辑块内的跳转和循环指令，这些指令可以中止程序原有的线性逻辑流,跳转到另一处执行程序。跳转或循环指令的操作数是地址标号，该地址标号指出程序要跳往何处，标号最多为4个字符，第一个字符必须是字母，其余字符可为字母或数字。

1. 无条件跳转指令

无条件跳转指令(JU)将无条件中断正常的程序逻辑流，使程序跳转到目标处继续执行，见图4-40。

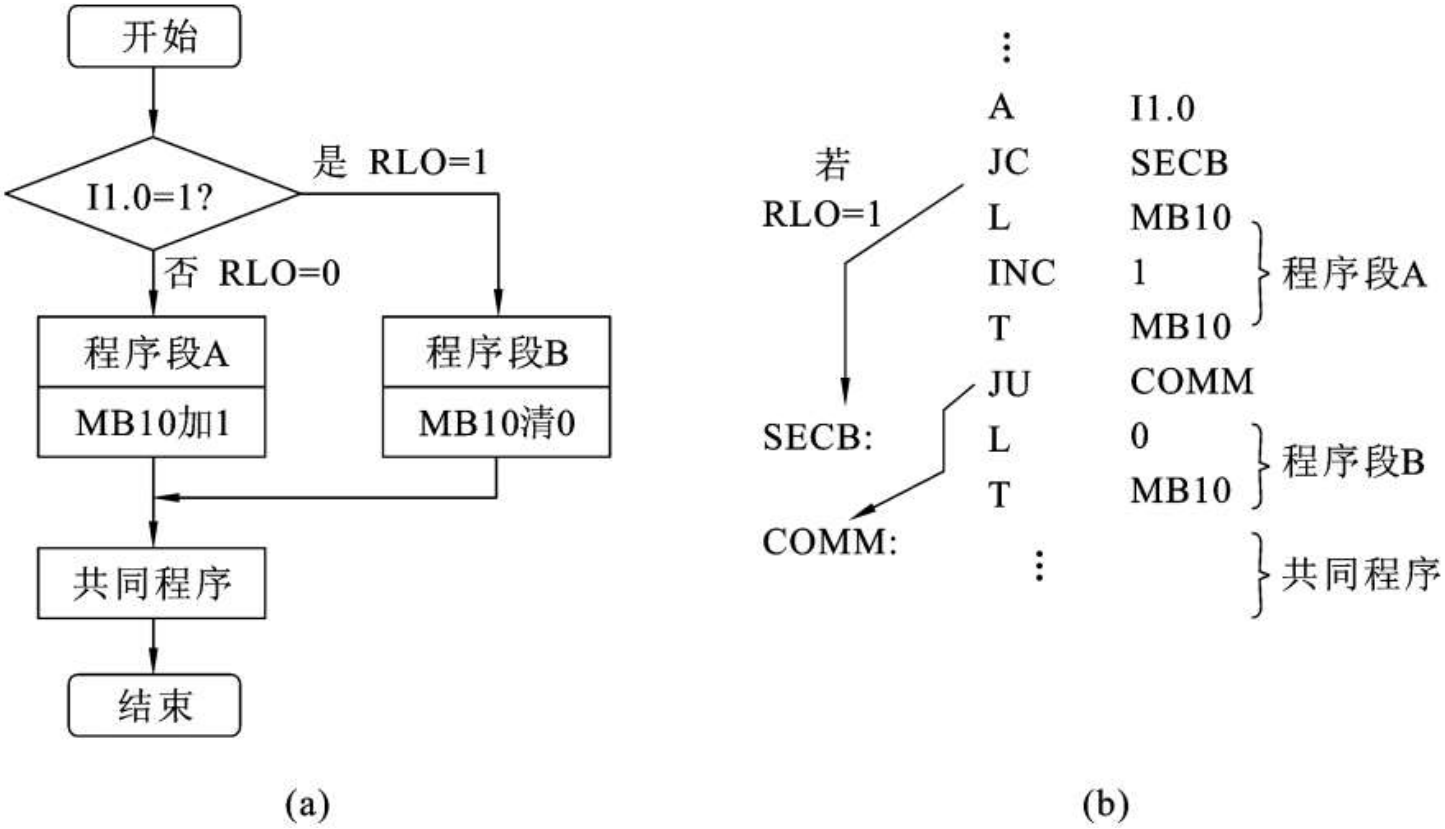


图4-40　　使用跳转指令控制程序流

2. 条件跳转指令

条件跳转指令见表4-31。

表4-31　条件跳转指令

指　令	说　明
JC	当 RLO=1 时跳转
JCN	当 RLO=0 时跳转
JCB	当 RLO=1 且 BR=1 时跳转，将 RLO 保存在 BR 中
JNB	当 RLO=0 且 BR=0 时跳转，将 RLO 保存在 BR 中
JBI	当 BR=1 时跳转，指令执行时，OR、FC 清 0，STA 置 1
JNBI	当 BR=0 时跳转，指令执行时，OR、FC 清 0，STA 置 1
JO	当 OV=1 时跳转
JOS	当 OS=1 时跳转，指令执行时，OS 清 0
JZ	累加器 1 中的计算结果为零跳转
JN	累加器 1 中的计算结果为非零跳转
JP	累加器 1 中的计算结果为正跳转
JM	累加器 1 中的计算结果为负跳转
JMZ	累加器 1 中的计算结果小于等于零(非正)跳转
JPZ	累加器 1 中的计算结果大于等于零(非负)跳转
JUO	实数溢出跳转

条件跳转指令与CC0位、CC1位的关系见表4-32。

表4-32　条件跳转指令与CC0位、CC1位的关系

状　态		计 算 结 果	触发的跳转指令 CC1
CC1	CC0		
0	0	=0	0
1 或 0	0 或 1	≠0	1 或 0
1	0	>0	1
0	1	<0	0
0 或 1	0	≤0	0 或 1
0	1 或 0	≥0	0
1	1	UO(溢出)	1

条件跳转指令控制程序的流程图如图4-41所示。

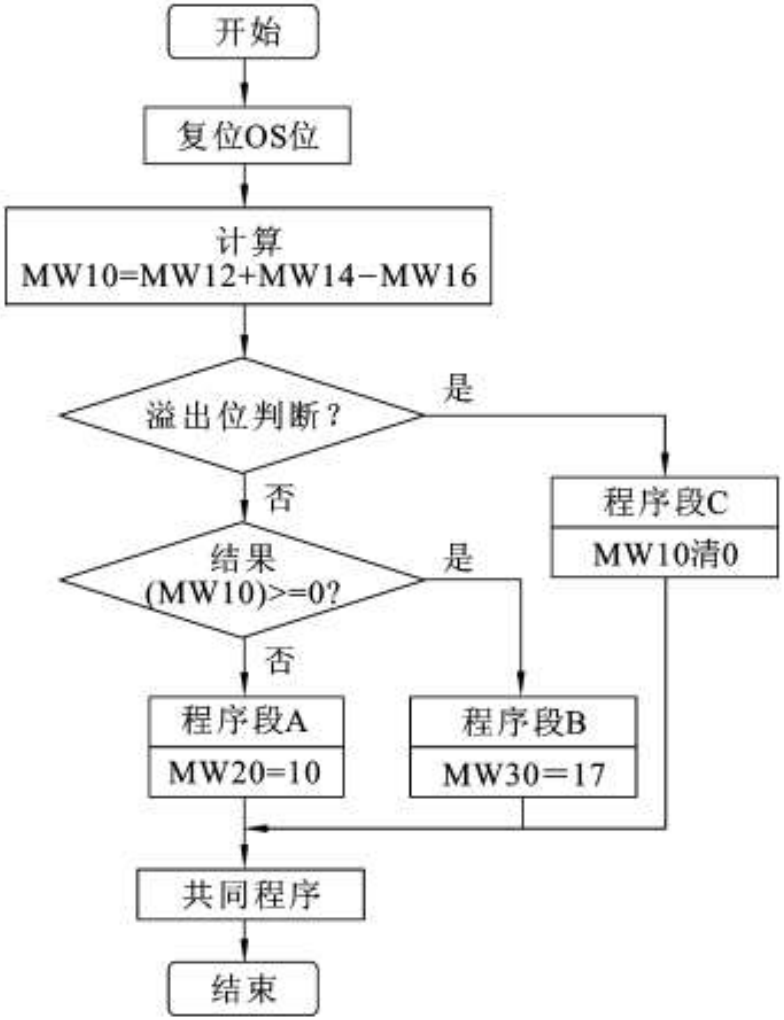


图4-41　　条件跳转指令控制程序流程

条件跳转指令控制程序如下。

```
      JOS ODEL      //OS位清0
ODEL:L    MW12      //计算
      L    MW14
      +I
      L    MW16
      -I
      T    MW10
      JOS SECC
      JPZ SECB
      L    +10      //程序段 A
      T    MW20
      JU   COMM
SECB:L    +17      //程序段 B
      T    MW30
      JU   COMM
SECC:L    0        //程序段 C
      T    MW10
COMM:                //共同程序
```

本例中未使用JO指令，因为JO指令只能判断前面一I指令是否溢出。

3. 循环指令

使用循环指令(LOOP)可以多次重复执行特定的程序段，重复执行的次数存在累加器1中，即以累加器1为循环计数器。LOOP指令执行时，将累加器1低字中的值减1，如果减1后其值不为0，则回到循环体开始处继续循环过程，否则，执行LOOP指令后面的指令。循环体是指循环标号和LOOP指令间的程序段。

由于循环次数不能是负数，因此程序应保证循环计数器中的数为正整数(数值范围为0~32767)或字型数据(数值范围为W#16#0000~W#16#FFFF)。图4-24所示是使用LOOP指令的例子。在下面的例子中，考虑到循环体(程序段A)中可能用到累加器1，特设置了循环计数暂存器MB10。

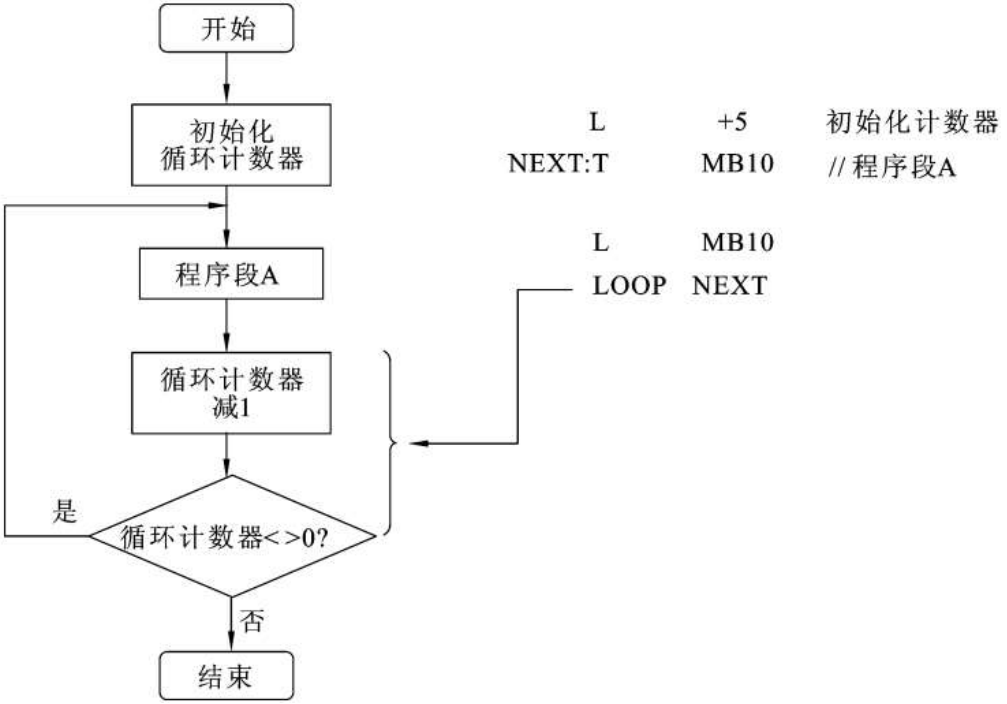


图4-42 使用LOOP指令

4. 梯形图逻辑控制指令

梯形图逻辑控制指令只有两条，可用于无条件跳转指令或条件跳转指令。由于无条件跳转时对应STL指令JU，因此不影响状态字。而由于在梯形图中目的标号只能在梯形网络的开始处，因此条件跳转指令会影响到状态字。在图4-43和图4-44中给出了梯形跳转指令的用法及其对应的语句表.

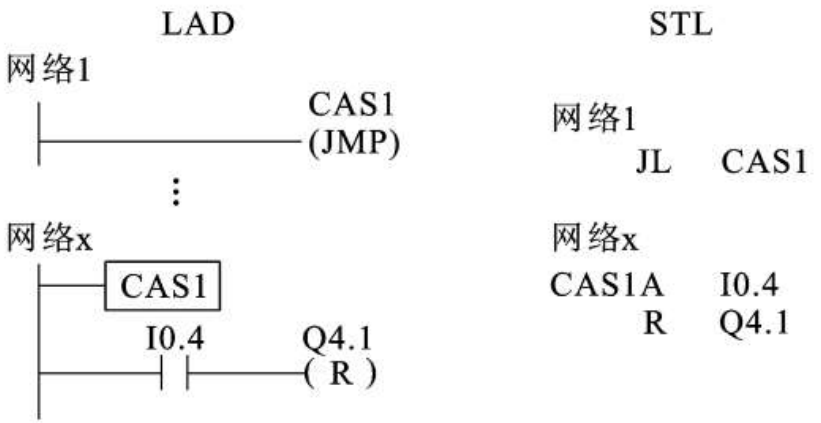


图4-43 无条件跳转指令

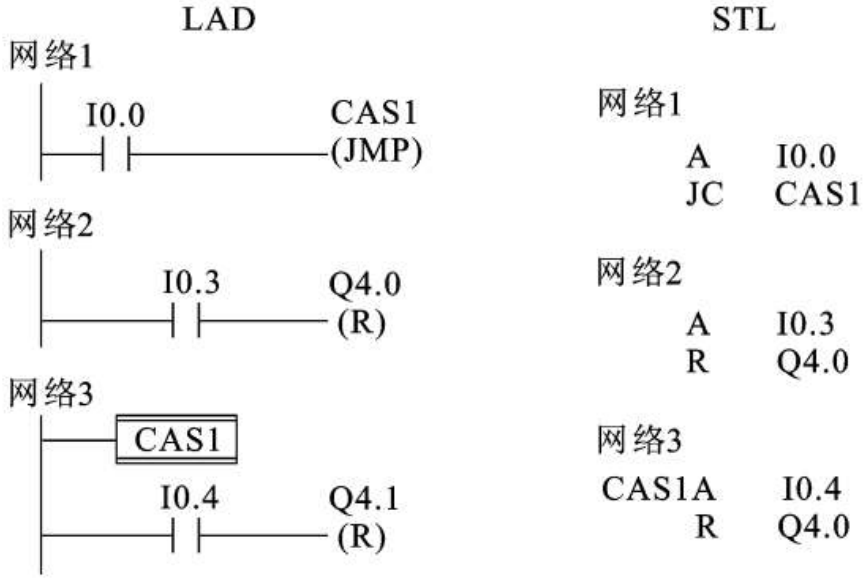
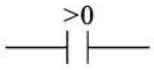
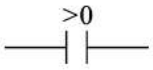
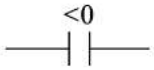
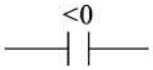
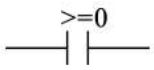
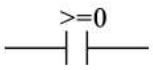
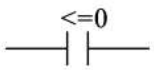
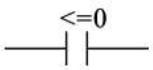
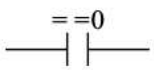
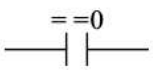
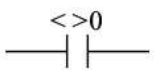
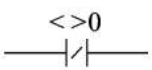
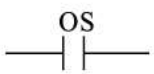
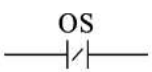
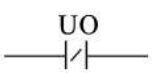


图4-44 条件跳转指令

状态位的常开/常闭触点的详细介绍见表4-33

表4-33　状态位常开/常闭触点

LAD 单元		LAD 单元
		算术运算结果大于 0，则常开触点闭合、常闭触点断开
		算术运算结果小于 0，则常开触点闭合、常闭触点断开
		算术运算结果大于等于 0，则常开触点闭合、常闭触点断开
		算术运算结果小于等于 0，则常开触点闭合、常闭触点断开
		算术运算结果等于 0，则常开触点闭合、常闭触点断开
		算术运算结果不等于 0，则常开触点闭合、常闭触点断开
		若状态字的 OV 位（溢出位）为 1，则常开触点闭合、常闭触点断开
		若状态字的 OS 位（存储溢出位）为 1，则常开触点闭合、常闭触点断开
		浮点算术运算结果溢出，则常开触点闭合、常闭触点断开
		若状态字的 BR 位（二进制结果位）为 1，则常开触点闭合、常闭触点断开

这些LAD单元可以用在梯形图程序中，影响逻辑运算结果(RLO)，最终形成以状态位为条件的跳转操作。图4-45所示的是使用状态位的一个例子。

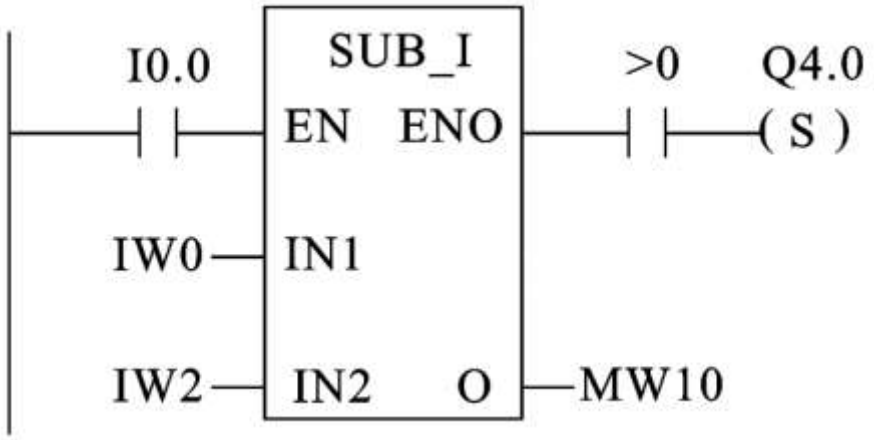


图4-45　使用状态位指令

图4-45中，如果输入位I0.0为1，则执行整数减操作方法；如果输入字IW0大于输入字IW2，(IW0)－(IW2)大于0，则输出端Q4.0被置位。

相应的语句表程序如下。

```
A(
A10.0
JNB     _0001
L       IW0
L       IW2
—I
T       MW10
AN      OV          //若OV为1,则RLO为0,否则RLO为1
SAVE                //使BR＝RLO
CLR
_0001: A BR
)
A       >0
S       Q4.0
```

4.6.2 程序控制指令

程序控制指令是指功能块(FB、FC、SFB、SFC)调用指令和逻辑块(OB、FB、FC) 结束指令。调用块或结束块可以是有条件的或是无条件的，STEP7中的功能块实质上就是子程序。

1.STL程序控制指令

STL程序控制指令见表4-34。

表4-34 STL程序控制指令

指　令	说　　明
CALL	该指令在程序中无条件执行，用于调用 FB、FC、SFB、SFC
UC	该指令可在程序中无条件调用功能块(一般是 FC 或 SFC)，但不能传递参数
CC	RLO=1，调用功能块(一般是 FC)，但不能传递参数
BEU	该指令无条件结束当前块的扫描，将控制返回给调用块
BEC	若 RLO=1，则结束当前块的扫描，将控制返回给调用块；若 RLO=0，则将 RLO 置 1，程序继续在当前块内扫描

2. 梯形图程序控制指令

梯形图程序控制指令见表4-35。

表4-35　梯形图程序控制指令

LAD指令	参　数		数据类型	存储区	说　明
<FC/SFC no.> ——(CALL)	FC/SFC　no.		BLOCK_FC	—	no.为被调用的不带参数的FC或SFC号数
<DB no.> FB no. EN　ENO	方块上部符号	参　数			
	FB no.	DB no.	BLOCK_DB	—	调用FB时背景数据块号
<DB no.> FB no. EN　ENO	FC no.	Block no.	BLOCK_FB/ BLOCK_FC	—	被调用的功能块号
	SFB no.	EN	BOOL	I、Q、M、D、L	允许输入
	SFC no.	ENO	BOOL	Q、M、D、L	允许输出
——(RET)	—		—	—	块结束

4.6.3 主控继电器指令

主控继电器(MCR)是一种美国梯形图逻辑主控开关，用于控制信号流(电流路径)的通断。其内部结构如图4-46所示。

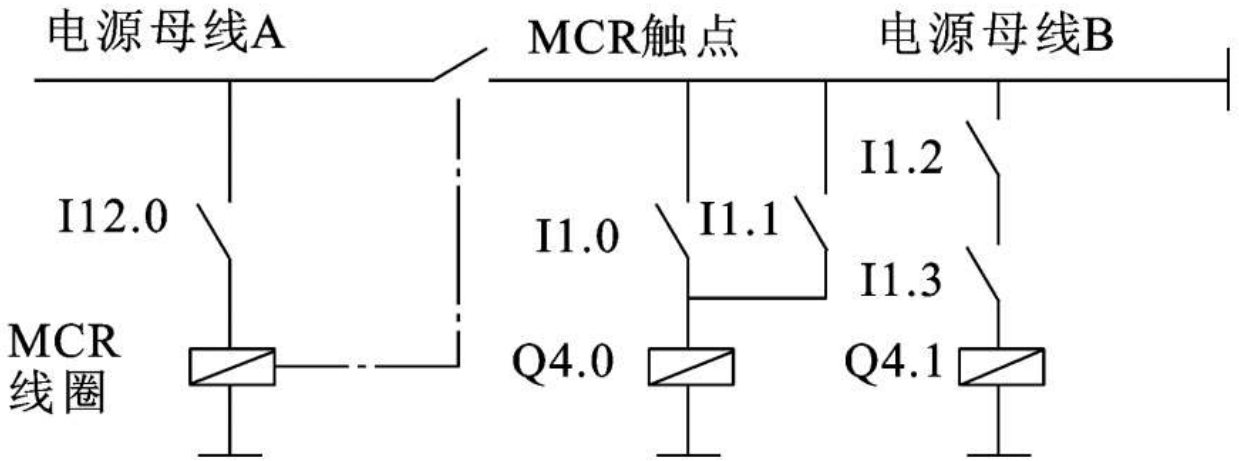


图4-46　主控继电器

主控继电器对应的语句表如下。

```
MCRA
A   I2.0
MCR(
O   I1.0
O   I1.1
=  Q4.0
A   I1.2
A   I1.3
=  Q4.1
)MCR
MCRD
```

STEP7中与主控继电器相关的指令见表4-36。

表4-36　STEP7中与主控继电器相关的指令

STL 指令	LAD 指令	说　　明
MCRA	——(MCRA)	激活 MCR 区，表明一个 MCR 区域的开始
MCRD	——(MCRD)	表明一个按 MCR 方式操作区域的结束
MCR(	——(MCR<)	主控继电器，并产生一条母线(子母线)
)MCR	——(MCRA>)	恢复 RLO，结束子母线，返回主母线