第 4 章　基本指令

S7-200 系列 PLC 主机中有两类基本指令集：SIMATIC 指令集和 IEC 1131-3 指令集。程序员可以任选一种，两者都提供了许多类型的指令以完成广泛的自动化任务。SIMATIC 指令集是为 S7-200 系列 PLC 设计的，该指令执行时间短，而且可以供 LAD、STL 和 FBD 三种编程语言使用。

本章概括了 SIMATIC 指令集中的基本指令及使用方法，所涉及的指令包括基本逻辑指令、复杂逻辑指令、定时器指令、计数器指令比较指令、数学运算指令、逻辑运算指令、表功能指令和数据转换指令等；同时介绍了编程元件的有效编程范围和梯形图基本绘制规则。

- 位操作类指令，主要是位操作相关运算指令，也包含定时器和计数器指令等
- 运算指令，包括常用的算术运算和逻辑运算指令
- 其他数据处理指令，包括数据的传送、移位、填充和交换等指令
- 表功能指令，包括对表的存取和查找指令
- 转换指令，包括数据类型转换、码转换和字符转换指令

- SIMATIC 基本指令的应用

4.1　位操作类指令

本节位操作类指令都是常用的与位操作和位运算有关的指令，也包括定时器和计数器指令等。利用这些指令基本可以实现继电接触控制系统的控制任务。

为顺利介绍本章各指令及其应用情况，首先介绍指令、程序及图形的一些基本格式及说明方式的约定，这些约定也适用于本书后面其他章节。

4.1.1　指令使用概述

可编程序控制器的指令系统依赖机器硬件，同时又是编写控制程序的基础。所以学习指令系统及编程时要将程序和硬件相结合，同时遵循一定的编程规则。

1．主机的有效编程范围

存储器的存储容量及各编程元件的有效编程范围如表 4.1 所示。

表 4.1　存储器编程范围

编程元件		CPU221	CPU222	CPU224	CPU226
输入映像寄存器		I0.0-I15.7	I0.0-I15.7	I0.0-I15.7	I0.0-I15.7
输出映像寄存器		Q0.0-Q15.7	Q0.0-Q15.7	Q0.0-Q15.7	Q0.0-Q15.7
模拟输入，只读		无	AIW0-AIW30	AIW0-AIW62	AIW0-AIW62
模拟输出，只写		无	AQW0-AQW30	AQW0-AQW62	AQW0-AQW62
变量存储器		VB0.0-VB2047.7	VB0.0-VB2047.7	VB0.0-VB5119.7	VB0.0-VB5119.7
局部变量存储器		LB0.0-LB63.7	LB0.0-LB63.7	LB0.0-LB63.7	LB0.0-LB63.7
位存储器		M0.0-M31.7	M0.0-M31.7	M0.0-M31.7	M0.0-M31.7
顺序控制继电器		S0.0-S31.7	S0.0-S31.7	S0.0-S31.7	S0.0-S31.7
特殊存储器		SM0.0-SM179.7	SM0.0-SM179.7	SM0.0-SM179.7	SM0.0-SM179.7
只读型		SM0.0-SM29.7	SM0.0-SM29.7	SM0.0-SM29.7	SM0.0-SM29.7
定时器	编号	256 (T0-T255)	256 (T0-T255)	256 (T0-T255)	256 (T0-T255)
	TONR 1ms	T0,T64	T0,T64	T0,T64	T0,T64
	TONR 10ms	T1-T4, T65-T68	T1-T4, T65-T68	T1-T4, T65-T68	T1-T4, T65-T68
	TONR 100ms	T5-T31, T69-T95	T5-T31, T69-T95	T5-T31, T69-T95	T5-T31, T69-T95
	TON/TOFF 1ms	T32,T96	T32,T96	T32,T96	T32,T96
	TON/TOFF 10ms	T33-T36, T97-T100	T33-T36, T97-T100	T33-T36, T97-T100	T33-T36, T97-T100
	TON/TOFF 100ms	T37-T63, T101-T255	T37-T63, T101-T255	T37-T63, T101-T255	T37-T63, T101-T255
计数器		C0-C255	C0-C255	C0-C255	C0-C255
高速计数器		HSC0,HSC3, HSC4,HSC5	HSC0,HSC3, HSC4,HSC5	HSC0-HSC5	HSC0-HSC5
累加器		AC0-AC3	AC0-AC3	AC0-AC3	AC0-AC3
跳转及其标号		0-255	0-255	0-255	0-255
子程序编号及调用		0-63	0-63	0-63	0-63
中断时间		0-127	0-127	0-127	0-127
PID 回路		0-7	0-7	0-7	0-7
通信口		通信口 0	通信口 0	通信口 0	通信口 0，1

许多指令中含有操作数，操作数的有效编址范围如表 4.2 所示。

表 4.2　操作数编程范围

操作数类型	CPU221	CPU222	CPU224CPU226
位	I0.0-15.7 ，M0.0-31.7 Q0.0-15.7， S0.0-31.7 SM0.0-179.7， T0-255 C0-255，V0.0-2047.7 L0.0-63.7	I0.0-15.7 ，M0.0-31.7 Q0.0-15.7， S0.0-31.7 SM0.0-179.7， T0-255 C0-255，V0.0-2047.7 L0.0-63.7	I0.0-15.7 ，M0.0-31.7 Q0.0-15.7， S0.0-31.7 SM0.0-179.7， T0-255 C0-255，V0.0-5119.7 L0.0-63.7
字节	IB0-15，QB0-15 MB0-31，SMB0-179 SB0-31，VB0-2047 LB0-63，AC0-3 常数	IB0-15，QB0-15 MB0-31，SMB0-179 SB0-31，VB0-2047 LB0-63，AC0-3 常数	IB0-15，QB0-15 MB0-31，SMB0-179 SB0-31，VB0-5119 LB0-63，AC0-3 常数
字	IW0-14，QW0-14 MW0-30，SMW0-178 SW0-30，VW0-2046 LW0-62，AC0-3 T0-255，C0-255 常数	IW0-14，QW0-14 MW0-30，SMW0-178 SW0-30，VW0-2046 LW0-62，AC0-3 T0-255，C0-255 AIW0-30，AQW0-30 常数	IW0-14，QW0-14 MW0-30，SMW0-178 SW0-30，VW0-5118 LW0-62，AC0-3 T0-255，C0-255 AIW0-30，AQW0-30 常数
双字	ID0-12，QD0-12 MD0-28，SMD0-176 SD0-28，VD0-2044 LD0-60，AC0-3 HC0，3，4，5 常数	ID0-12，QD0-12 MD0-28，SMD0-176 SD0-28，VD0-2044 LD0-60，AC0-3 HC0，3，4，5 常数	ID0-12，QD0-12 MD0-28，SMD0-176 SD0-28，VD0-5116 LD0-60，AC0-3 HC0-5 常数

2．梯形图的基本绘制规则

（1）能流。

梯形图是在继电器电路图基础之上演变过来的，其结构和继电器电路十分相似。在梯形图中以左母线和右母线（通常不画出）代替电源线，而用“能流”（能量流）的概念来代替继电器电路中的电流。和真实电流一样，梯形图中的能流只能单向流动（从左到右，从上到下）。

梯形图中的触点闭合时，能流就可以从左母线通过触点（如图 4.1 中的 I0.0）向后传送；而功能指令（如图 4.1 中 ADD_I 和 MOV_W）只有 EN 端（能流输入端）有能流流入，且功能指令被正确执行后，其 ENO 端（能流输出端）才把能流传到下一个单元，否则能流在此终止（相当于实际电路中的断路）。在图 4.1 所示梯形图中，触点 I0.0 闭合后，能流从左母线传送至功能指令 ADD_I；ADD_I 正确执行后，其 ENO 端为 1，能流向后传送到功能指令 MOV_W，MOV_W 指令被执行。

（2）网络（Network）。

在梯形图中，程序被划分成称为网络（Network）的一些段。一个网络是触点、线圈和功能块的有序排列组成的完整电路。STEP 7-Micro/WIN 32 允许以网络为单位给梯形图程序

建立注释；语句表程序不使用网络，但可以使用 Network 关键词对程序分段，以便通过编程软件自动在 STL、LAD 和 FBD 程序之间进行转换。

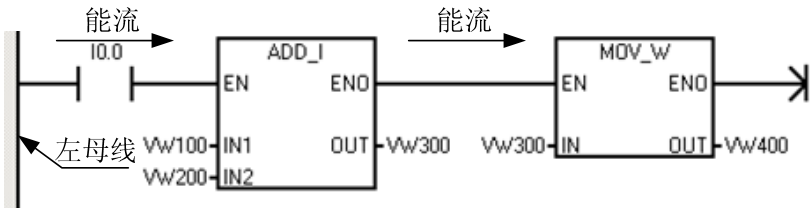


图 4.1　梯形图中的能流

（3）编程顺序。

梯形图按照从上到下，从左到右的顺序绘制。每个逻辑行开始于左母线，一般来说，触点要放在左侧，线圈和指令盒放在右侧，线圈和指令盒的右边不能再有触点，整个梯形图呈阶梯形结构。

（4）编号分配。

对外接电路各元件分配编号，编号的分配必须是主机或扩展模块本身实际提供的，而且是可以用来进行编程的（无论是输入设备还是输出设备）。每个元件都必须分配不同的输入和输出点。两个设备不能共用一个输入输出点。

（5）内、外触点的配合。

可编程序控制器无法识别输入设备用的是常开还是常闭触点，只能识别输入电路是接通还是断开。梯形图中的常开、常闭触点只是反映对应输入、输出映象寄存器中的相应位的状态，而不是现场物理开关触点的实际状态。因此设计 PLC 程序时必须注意物理开关触点（外触点）和梯形图中使用的触点（内触点）之间的配合。

通常情况下，内外触点的选择没有特殊的要求，但在某些特殊情况下要求必须使用特定的外触点时（比如停止开关、热继电器登必须使用常闭触点），就必须根据需要选择对应的内触点。内触点的选择决定于外触点的类型和控制电路的要求两方面。内、外触点的配合关系如表 4.3 所示。

表 4.3　内外触点的配合关系

外触点类型	控制电路的要求	内触点类型
常开	启动	常开
常开	停止	常闭
常闭	启动	常闭
常闭	停止	常开

当起动按钮用常开触点时，在梯形图中输入触点应用常开触点；当起动按钮用常闭触点时，在梯形图中输入触点应用常闭触点。

当停止按钮用常开触点时，在梯形图中输入触点应用常闭触点；当停止按钮用常闭触点时，在梯形图中输入触点则用常开触点。

（6）触点的使用次数。

梯形图中的触点并不是真正的物理触点，而是保存了外部触点状态的寄存器，梯形图中对某一触点的使用，实际是对寄存器中数据的逻辑操作，这种逻辑操作是没有次数限制的。因此在梯形图中，同一编程元件（如输入输出继电器、通用辅助继电器、定时器和计数器等元件）的常开、常闭触点可以任意多次重复使用，不受限制。

（7）线圈的使用次数。

若在同一梯形图中，同一组件的线圈使用两次或两次以上，称为双线圈输出。PLC 的扫描特性决定了在双线圈输出时，只有最后一次的线圈操作才是有效的，而前面的线圈操作是无效的。因此在绘制梯形图时，最好不要使用双线圈。

（8）线圈的连接。

用一个使能条件驱动多个线圈性质的指令或指令盒（如定时器指令），连接时必须用并联，不能出现串联形式。非线圈性质的指令盒（如上图的 ADD_I 指令），其 ENO 输出可以连接其他指令盒或线圈，但习惯上仍采用并联结构。

4.1.2 基本逻辑指令

基本逻辑指令在语句表语言中是指对位存储单元的简单逻辑运算，在梯形图中是指对触点的简单连接和对标准线圈的输出。

语句表编程语言用指令助记符创建控制程序，它是一种面向具体机器的语言，可被 PLC 直接执行，一般来说，语句表语言更适合于熟悉可编程序控制器和逻辑编程方面有经验的编程人员。用这种语言可以编写出用梯形图或功能框图无法实现的程序，但利用语句表时进行位运算时需要考虑主机的内部存储结构。

S7-200 可编程序控制器使用一个逻辑堆栈来分析控制逻辑，用语句表编程时要根据这一堆栈逻辑进行组织程序，用相关指令来实现堆栈操作。用梯形图和功能框图时，程序员不必考虑主机的这一逻辑，编程软件会自动地插入必要的指令来处理各种堆栈逻辑操作。

可编程序控制器中的堆栈是一组能够存储和取出数据的暂时存储单元。堆栈的存取特点是“后进先出”，S7-200 可编程序控制器的主机逻辑堆栈结构如表 4.4 所示。

表 4.4 逻辑堆栈结构

堆栈结构	名称	说明
S0	STACK 0	第一个堆栈（即栈顶）
S1	STACK 1	第二个堆栈
S2	STACK 2	第三个堆栈
S3	STACK 3	第四个堆栈
S4	STACK 4	第五个堆栈
S5	STACK 5	第六个堆栈
S6	STACK 6	第七个堆栈
S7	STACK 7	第八个堆栈
S8	STACK 8	第九个堆栈

这种逻辑堆栈结构是由九个堆栈存储器位组成的串联堆栈，栈顶是布尔型数据进出堆栈的必由之路。进栈时，数据由栈顶压入，堆栈中原来所存的数据被串行下移一格，如果原来STACK 8 中存有数据，则这一数据将被推出堆栈而自动丢失。出栈时，数据从栈顶被取出，所有数据串行上移一格，STACK 8 中随机地装入一个数值，用语句表编程时程序员应该注意这一特点。

栈顶 STACK 0 在此逻辑堆栈的位运算中兼有累加器的作用，存放第一操作数。对于简单逻辑指令，通常是进栈操作和一些最简单的位运算，这些运算是栈顶与第二个堆栈的内容进行与、或、非等逻辑运算。对于复杂指令，可以是堆栈中的其他数据位直接进行运算，结果经栈顶弹出。

基本逻辑指令主要包括标准触点指令、正负跳变指令、置位和复位指令、立即指令。

1．标准触点指令

标准触点指令有 LD、LDN、A、AN、O、ON、NOT、=指令（语句表），如表 4.5 所示。这些指令在逻辑堆栈中对寄存器位进行操作。标准触点指令中如果有操作数，则为 BOOL 型，操作数的编址范围可以是 I、Q、M、SM、T、C、S、VL。

表 4.5　标准触点指令

指令	语句	描述
LD（Load）	LD　bit	装载，电路开始的常开触点
LDN（Load Not）	LDNbit	装载非，电路开始的常闭触点
A（And）	A　bit	与，串联一个常开触点
AN（And Not）	AN　bit	非与，串联一个常闭触点
O（Or）	O　bit	并联一个常开触点
ON（Or Not）	ON　bit	并联一个常闭触点
NOT	NOT　bit	取反，（无操作数）
=	=　bit	输出指令，将逻辑运算结果输出到指定存储器位或输出继电器对应的映像寄存器位，以驱动本位线圈

应用举例：图 4.2 所示程序段用于介绍标准触点指令在梯形图和语句表语言编程中的应用，程序执行的时序图如图 4.3 所示。

梯形图绘制技巧提示：

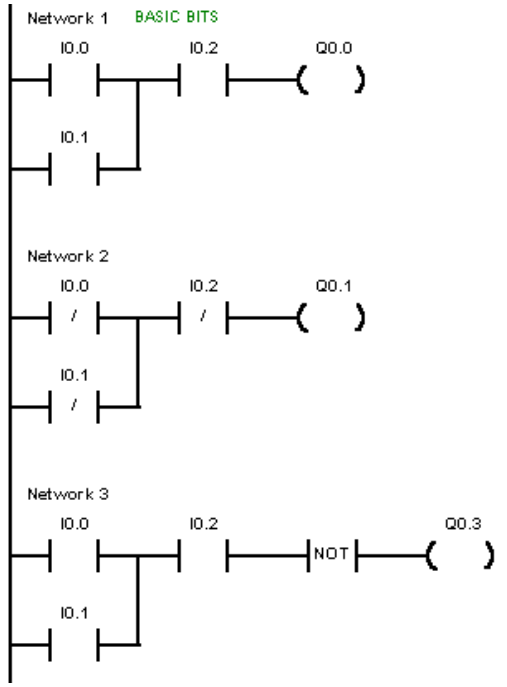
（1）梯形图每一行都是从左母线开始，而且输出线圈接在最右边，输入触点不能放在输出线圈的右边。

（2）输出线圈不能直接与左母线连接。

（3）遵循“上重下轻，左重右轻，避免混联”，即梯形图应把串联触点较多的电路放在梯形图上方；把并联触点较多的电路放在梯形图最左边。

2．正负跳变指令

正负跳变指令在梯形图中以触点形式使用。用于检测脉冲的正跳变（上升沿）或负跳变（下降沿），利用跳变让能流接通一个扫描周期，即可以产生一个扫描周期长度的微分脉冲，常用此脉冲触发内部继电器线圈。




```
LD      I0.0    //装入常开触点
O       I0.1    //或常开触点
A       I0.2    //与常开触点
=       Q0.0    //输出触点
//如果本梯级中将 I0.1 的触点改
//为 Q0.0 的常开触点，则成为电
//机起动停止控制环节的梯形图。
LDN     I0.0    //装入常闭触点
ON      I0.1    //或常闭触点
AN      I0.2    //与常闭触点
=       Q0.1    //
```

```
LD      I0.0    //
O       I0.1    //
A       I0.2    //
NOT     //取非，即输出反相
=       Q0.3    //
```

图 4.2　标准触点 LAD 和 STL 例

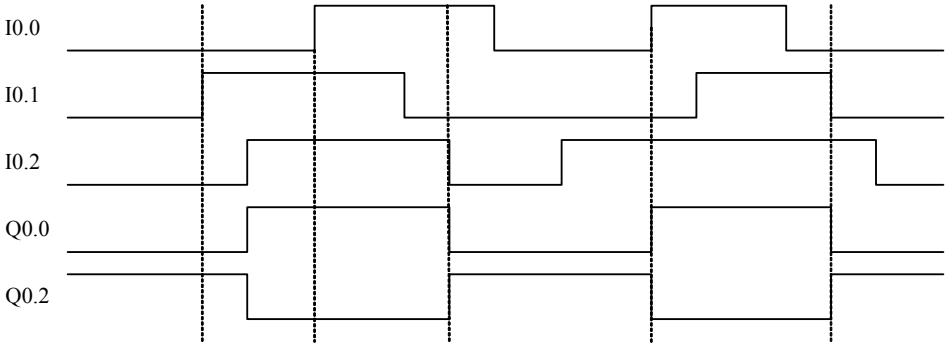


图 4.3　时序图

（1）EU，正跳变指令。

正跳变触点检测到脉冲的每一次正跳变后，产生一个微分脉冲，脉冲持续时间为一个扫描周期。

指令格式：EU（无操作数）

（2）ED，负跳变指令。

负跳变触点检测到脉冲的每一次负跳变后，产生一个微分脉冲，脉冲持续时间为一个扫描周期。

指令格式：ED（无操作数）

应用举例：图 4.4 所示是跳变指令的程序片断，图 4.5 所示是图 4.4 指令执行的时序。

3．置位和复位指令

置位即置 1，复位即置 0。置位和复位指令可以将位存储区的某一位开始的一个或多个

（最多可达 255 个）同类存储器位置 1 或置 0。这两条指令在使用时需指明 3 点：操作性质、开始位和位的数量。各操作数类型及范围如表 4.6 所示。

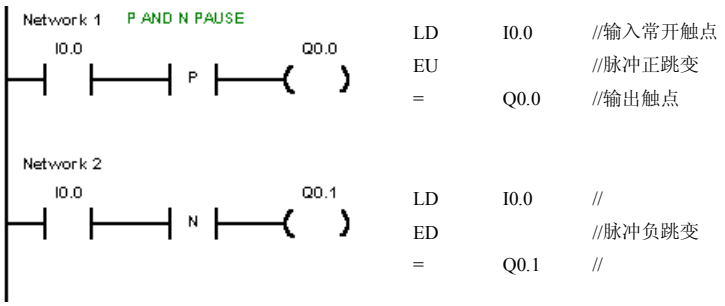


图 4.4　跳变应用

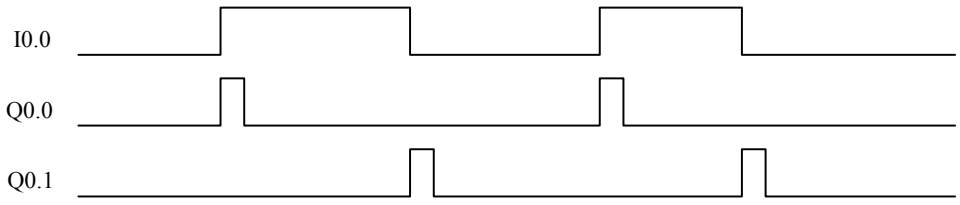


图 4.5　时序

表 4.6　操作数

操作数	范围	类型
位 bit	I, Q, M, SM, TC, V, S, L	BOOL 型
数量 N	VB , IB , QB , MB , SMB , LB , SB , AC , *VD , *AC , *LD	BYTE 型

（1）S，置位指令。

将位存储区的指定位（位 bit）开始的 N 个同类存储器位置位。

用法：　S　　bit,　N

例：　　S　　Q0.0, 1

（2）R，复位指令。

将位存储区的指定位（位 bit）开始的 N 个同类存储器位复位。当用复位指令时，如果是对定时器 T 位或计数器 C 位进行复位，则定时器位或计数器位被复位，同时，定时器或计数器的当前值被清零。

用法：　R　　bit,　N

例：　　R　　Q0.2, 3

应用举例：图 4.6 为置位和复位指令应用程序片断，图 4.7 为指令执行的时序图。

注意

在存储区的一位或多位被置位后，不能自己恢复，必须用复位指令由 1 跳回到 0。

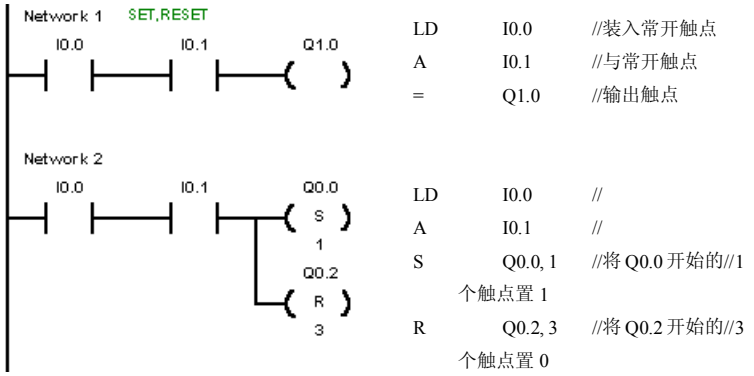


图 4.6　置位复位

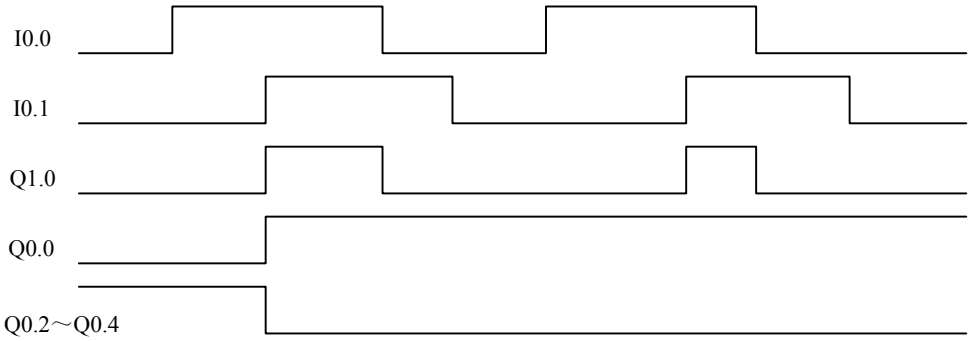


图 4.7　时序图

4．立即指令

立即指令允许直接访问物理输入和输出点。当用立即指令读取输入点的状态时，相应输入映像寄存器中的值不会同时更新；用立即指令访问输出点时，输出点和相应的输出寄存器的内容同时被更新。

注意

只有输入继电器 I 和输出继电器 Q 可以使用立即指令。

（1）立即触点指令。

在每个标准触点指令的后面加“I”。指令执行时，立即读取物理输入点的值，但是不刷新对应映像寄存器的值。

这类指令包括：LDI、LDNI、AI、ANI、OI 和 ONI。下面以 LDI 指令为例。

用法：　LDI　bit

例：　　LDI　I0.2

注意

bit 只能是 I 类型。

（2）=I，立即输出指令。

用立即指令访问输出点时，把栈顶值立即复制到指令所指出的物理输出点，同时，相应

的输出映像寄存器的内容也被刷新。

用法： =I bit

例： =I Q0.2

注意

bit 只能是 Q 类型。

（3）SI，立即置位指令。

用立即置位指令访问输出点时，从指令所指出的位（bit）开始的 N 个（最多为 128 个）物理输出点被立即置位，同时，相应的输出映像寄存器的内容也被刷新。

用法： SI bit, N

例： SI Q0.0, 2

注意

bit 只能是 Q 类型。SI 和 RI 指令的操作数类型及范围如表 4.7 所示。

表 4.7 操作数

操作数	范围	类型
位 bit	Q	BOOL
数量 N	VB，IB，QB，MB，SMB，LB，SB，AC，*VD，*AC，*LD，常数	BYTE

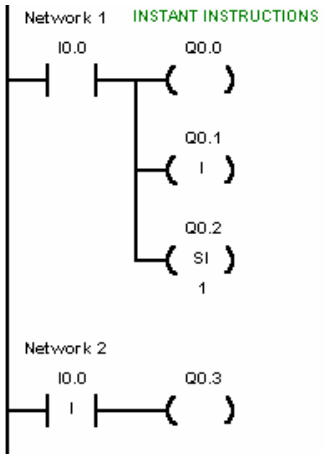
（4）RI，立即复位指令。

用立即复位指令访问输出点时，从指令所指出的位（bit）开始的 N 个（最多为 128 个）物理输出点被立即复位，同时，相应的输出映像寄存器的内容也被刷新。

用法： RI bit, N

例： RI Q0.0, 1

应用举例：图 4.8 所示为立即指令应用中的一段程序，图 4.9 所示是程序对应的时序图。




```
LD      I0.0      //装入常开触点
=       Q0.0      //输出触点，非立即
=I      Q0.1      //立即输出触点
SI      Q0.2, 1   //从 Q0.2 开始的 1 个
                  //触点被立即置 1

LDI     I0.0      //立即输入触点指令
=       Q0.3      //
```

图 4.8 立即指令程序

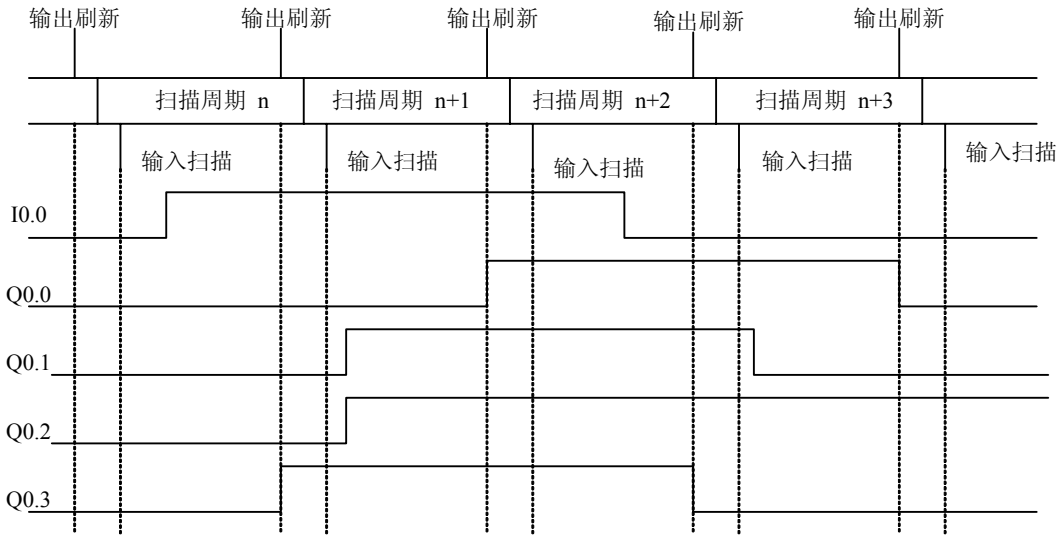


图 4.9 时序图

时序图中的 Q0.1 和 Q0.2 的跳变与扫描周期的输入扫描时刻不同步，这是由于两者的跳变发生在程序执行阶段，立即输出和立即置位指令执行完成的一刻。

4.1.3 复杂逻辑指令

基本逻辑指令涉及可编程元件的触点和线圈的简单连接，不能表达在梯形图中触点的复杂连接结构，而复杂逻辑指令则可以实现对触点复杂连接的描述。

本类指令包括 ALD、OLD、LPS、LRD、LPP 和 LDS，这些指令中除 LDS 外，其余指令都无操作数。

1．栈装载与指令

ALD，栈装载与指令（块与）。在梯形图中用于将并联电路块进行串联连接。

在语句表中指令 ALD 执行情况如表 4.8 所示。

表 4.8 指令 ALD

名称	执行前	执行后	说明
STACK 0	1	0	假设执行前，S0=1，S1=0
STACK 1	0	S2	本指令对堆栈中的第一层 S0 和第二层 S1
STACK 2	S2	S3	的值进行逻辑与运算，结果放回栈顶。即：
STACK 3	S3	S4	S0=S0×S1
STACK 4	S4	S5	=1×0
STACK 5	S5	S6	=0
STACK 6	S6	S7	执行完本指令后堆栈串行上移 1 格，深度减 1
STACK 7	S7	S8	
STACK 8	S8	X	

2．栈装载或指令

OLD，栈装载或指令（或块）。在梯形图中用于将串联电路块进行并联连接。

在语句表中指令 OLD 执行情况如表 4.9 所示。

表 4.9　指令 OLD

名称	执行前	执行后	说明
STACK 0	1	1	假设执行前，S0=1，S1=0
STACK 1	0	S2	本指令对堆栈中的第一层 S0 和第二层 S1 的值进
STACK 2	S2	S3	行逻辑或运算，结果放回栈顶。即：
STACK 3	S3	S4	S0=S0+S1
STACK 4	S4	S5	=1+0
STACK 5	S5	S6	=1
STACK 6	S6	S7	执行完本指令后堆栈串行上移 1 个单元，深度减 1
STACK 7	S7	S8	
STACK 8	S8	X	

3．逻辑入栈指令

LPS，逻辑入栈指令（分支或主控指令）。在梯形图中的分支结构中，用于生成一条新的母线，左侧为主控逻辑块时，第一个完整的从逻辑行从此处开始。

注意　使用 LPS 指令时，本指令为分支的开始，以后必须有分支结束指令 LPP。即 LPS 与 LPP 指令必须成对出现。

在语句表中指令 LPS 执行情况如表 4.10 所示。

表 4.10　指令 LPS

名称	执行前	执行后	说明
STACK 0	1	1	假设执行前，S0=1
STACK 1	S1	1	本指令对堆栈中的栈顶 S0 进行复制，并将这个
STACK 2	S2	S1	复制值由栈顶压入堆栈。即：
STACK 3	S3	S2	S0=S0
STACK 4	S4	S3	=1
STACK 5	S5	S4	执行完本指令后堆栈串行下移 1 格，深度加 1
STACK 6	S6	S5	原来的栈底 S8 内容将自动丢失
STACK 7	S7	S6	
STACK 8	S8	S7	

4．逻辑出栈指令

LPP，逻辑出栈指令（分支结束或主控复位指令）。在梯形图中的分支结构中，用于将 LPS 指令生成一条新的母线进行恢复。

注意：使用 LPP 指令时，必须出现在 LPS 的后面，与 LPS 成对出现。

在语句表中指令 LPP 执行情况如表 4.11 所示。

表 4.11　指令 LPP

名称	执行前	执行后	说明
STACK 0	1	1	假设执行前，S0=1，S1=1。
STACK 1	1	S1	本指令将堆栈的栈顶 S0 弹出，则第二层 S1 的值
STACK 2	S1	S2	上升进入栈顶，用以进行本指令之后的操作。即：
STACK 3	S2	S3	S0=S1
STACK 4	S3	S4	=1
STACK 5	S4	S5	执行完本指令后堆栈串行上移 1 格，深度减 1
STACK 6	S5	S6	栈底 S8 内容将生成一个随机值 X
STACK 7	S6	S7	
STACK 8	S7	X	

5．逻辑读栈指令

LRD，逻辑读栈指令。在梯形图中的分支结构中，当左侧为主控逻辑块时，开始第二个和后边更多的从逻辑块。

在语句表中指令 LRD 执行情况如表 4.12 所示。

表 4.12　指令 LRD

名称	执行前	执行后	说明
STACK 0	1	0	假设执行前，S0=1，S1=0
STACK 1	0	0	本指令将堆栈中的第二层 S1 中的值进行复制，然后
STACK 2	S2	S2	将这个复制值放入栈顶 S0，本指令不对堆栈进行压入
STACK 3	S3	S3	和弹出操作。即：
STACK 4	S4	S4	S0=S1
STACK 5	S5	S5	=0
STACK 6	S6	S6	执行完本指令后堆栈不串行上移或下移，除栈顶值之外，其他部分的值不变
STACK 7	S7	S7	
STACK 8	S8	S8	

注意　LPS 后第一个和最后一个从逻辑块不用本指令。

6．装入堆栈指令

LDS，装入堆栈指令。本指令编程时较少使用。

指令格式：　LDS　　n（n 为 0～8 的整数）

例：　　LDS　　4

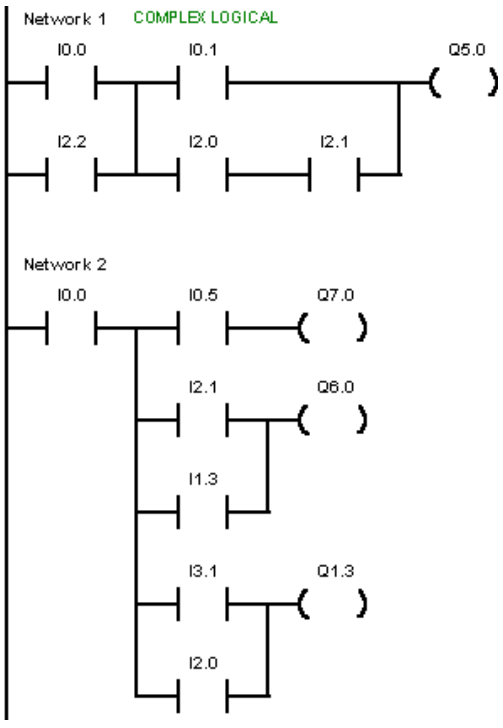
指令　　LDS　　4

在语句表中执行情况如表 4.13 所示。

表 4.13　指令 LDS

名称	执行前	执行后	说明
STACK 0	1	0	假设执行前，S0=1，S4=0
STACK 1	S1	1	本指令对堆栈中的第五层 S4 进行复制，并将这个
STACK 2	S2	S1	复制值由栈顶压入堆栈。即：
STACK 3	S3	S2	S0=S4
STACK 4	0	S3	=0
STACK 5	S5	0	执行完本指令后堆栈串行下移 1 格，深度加 1
STACK 6	S6	S5	原来的栈底 S8 内容将自动丢失
STACK 7	S7	S6	
STACK 8	S8	S7	

应用举例：图 4.10 所示是复杂逻辑指令在实际应用中的一段程序的梯形图。




```
LD      I0.0      //装入常开触点
O       I2.2      //或常开触点
LD      I0.1      //被串的块开始
LD      I2.0      //被并路开始
A       I2.1      //
OLD     //栈装载或，并路结束
ALD     //栈装载与，串路结束
LD      I0.0      //
LPS       //逻辑推入栈，主控
A       I0.5      //
=       Q7.0      //
LRD     //逻辑读栈，新母线
LD      I2.1      //
O       I1.3      //
ALD     //栈装载与
=       Q6.0      //
LPP       //逻辑弹出栈，母线复原
LD      I3.1      //
O       I2.0      //
ALD     //
=       Q1.3      //
```

图 4.10　复杂逻辑指令的应用

4.1.4 定时器指令

定时器是 PLC 中的重要编程元件，主要用于和时间相关的操作。定时器编程时需要提前输入时间预设值，在运行时当定时器的输入条件满足时开始计时，当前值从 0 开始按一定的时间单位增加，当定时器的当前值达到预设值时，定时器发生动作（相应的常开触点闭合，常闭触点断开），以便 PLC 响应而作出相应的动作。利用定时器的输入与输出触点就可以得到控制所需的延时时间。

S7-200 PLC 系统提供了 256 个定时器，定时器可以使用的定时指令分为 TON、TONR 和 TOF 3 种。每种定时指令都有 3 个精度等级（时间增量/时间单位/分辨率）：1ms、10ms 和 100ms，定时器类型、精度等级和定时器号关系如表 4.14 所示。

表 4.14 定时精度与编号

定时器类型	精度等级（ms）	最大当前值（s）	定时器编号
TON	1	32.767	T32,T96
	10	327.67	T33-T36,T97-T100
TOF	100	3276.7	T37-T63,T101-T255
	1	32.767	T0,T64
TONR	10	327.67	T1-T4,T65-T68
	100	3276.7	T5-T31,T69-T95

定时时间的计算：

T=PT×S （T 为实际定时时间，PT 为预设值，S 为精度等级）

例如：TON 指令用定时器 T33，预设值为 125，则实际定时时间

T=125×10=1250ms

指令操作数有 3 个：编号、预设值和使能输入。

（1）编号：用定时器的名称和它的常数编号（最大 255）来表示，即 Txxx，如 T4。

T4 不仅仅是定时器的编号，它还包含两方面的变量信息：定时器位和定时器当前值。

定时器位：定时器位与时间继电器的输出相似，当定时器的当前值达到预设值 PT 时，该位被置为“1”。

定时器当前值：存储定时器当前所累计的时间，它用 16 位符号整数来表示，故最大计数值为 32767。

（2）预设值 PT：数据类型为 INT 型。寻址范围可以是 VW、IW、QW、MW、SW、SMW、LW、AIW、T、C、AC、*VD、*AC、*LD 和常数。

（3）使能输入（只对 LAD 和 FBD）：BOOL 型，可以是 I、Q、M、SM、T、C、V、S、L 和能流。

1．接通延时定时器指令

TON，接通延时定时器指令，用于单一间隔的定时。首次扫描时定时器位 OFF，当前值为 0；使能输入接通时，定时器位为 OFF，当前值从 0 开始计数；当前值达到预设值时，定时器位 ON，当前值连续计数到 32767。使能输入断开，定时器自动复位，即定时器位

OFF，当前值为 0。

指令格式： TON Txxx,PT

例： TON T120,8

2．保持型接通延时定时器指令

TONR，保持型接通延时定时器指令，用于对许多间隔的累计定时。首次扫描时定时器位 OFF，当前值保持；使能输入接通时，定时器位为 OFF，当前值从 0 开始计数；使能输入断开，定时器位和当前值保持最后状态；使能输入再次接通时，当前值从上次的保持值继续计数，当累计当前值达到预设值时，定时器位 ON，当前值连续计数到 32767。

TONR 定时器只能用复位指令进行复位操作。

指令格式： TONR Txxx,PT

例： TONR T20,63

3．断开延时定时器指令

TOF，断开延时定时器指令。用于断开后的单一间隔定时。首次扫描时定时器位 OFF，当前值为 0；使能输入接通时，定时器位为 ON，当前值为 0；当使能输入由接通到断开时，定时器开始计数；当前值达到预设值时，定时器位 OFF，定时器停止计数。

TOF 复位后，如果使能输入再有从 ON 到 OFF 的负跳变，则可实现定时器再次启动。

指令格式： TOF Txxx,PT

例： TOF T35,6

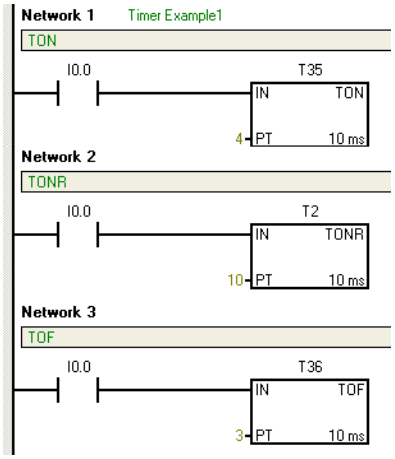
注意

（1）TON 定时器和 TOF 定时器不能使用相同的地址，比如不能同时使用 TON T32 和 TOF T32。

（2）3 种定时器都可以用复位指令复位，复位指令的执行结果是：定时器位变为 OFF；定时器当前值变为 0。

4．应用举例

例 1：图 4.11 所示是介绍 3 种定时器工作特性的程序片断，其中 T35 为通电延时定时器，T2 为保持型通电延时定时器，T36 为断电延时定时器。




```
LD      I0.0     //使能输入
TON     T35, 4   //通电延时定时
                 //延时时间为 40ms

LD      I0.0     //
TONR    T2, 10   //有记忆通电
                 //延时时间累计
                 //为 1000ms

LD      I0.0     //
TOF     T36, 3   //断电延时定时
                 //延时时间为 30ms
```

图 4.11 定时器特性

本梯形图程序中输入输出执行时序关系如图 4.12 所示。

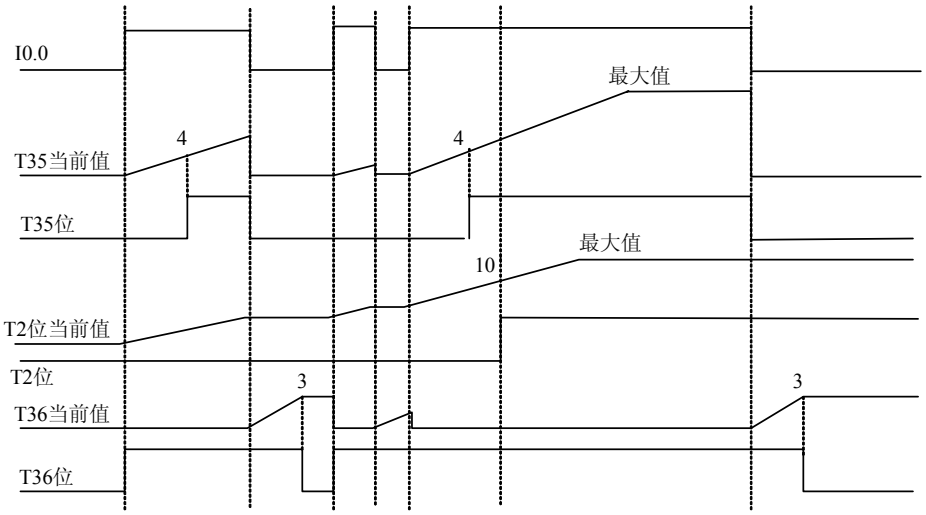
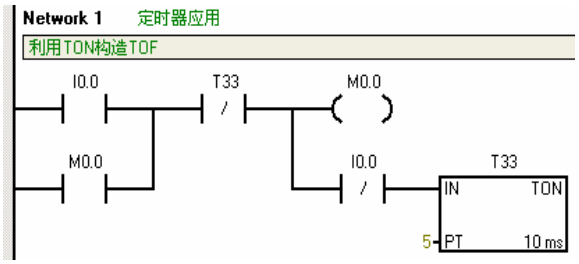


图 4.12　定时器时序

例 2：用 TON 类型定时器构造其他类型定时器。

某些 PLC 只有 TON 定时器，如果程序设计时需要其他类型的定时器，则可以利用 TON 来构造。

图 4.13 所示是利用 TON 构造的 TOF 类型定时器，其时序图与 TOF 的时序完全相同。



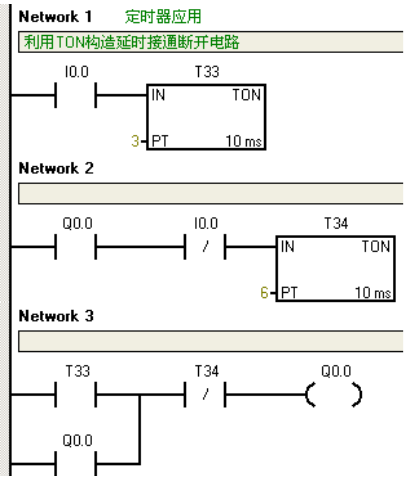

```
LD      I0.0     //启动 M0.0
O       M0.0     //自保
AN      T33      //断开 M0.0
=       M0.0     //瞬时闭合
                 //延时 50ms 断开
AN      I0.0     //连续输出
TON     T33, 5
```

图 4.13　定时器的应用 1

例 3：利用 TON 实现通电和断电都延时的触点作用。

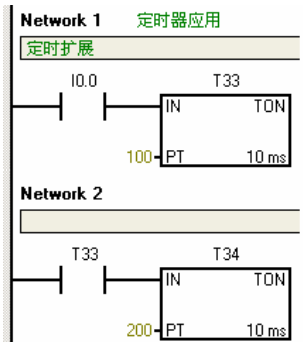
图 4.14 所示的梯形图程序中，当 I0.0 闭合后，启动 T33 定时器。30ms 后 T33 的常开触点闭合，由于 T34 尚未启动，其常闭触点闭合，线圈 Q0.0“通电”从而达到延时通电的效果。在触点 I0.0 闭合期间，其对应的常闭触点始终处于断开状态，定时器 T34 无法启动；一旦触点 I0.0 断开，其对应的常开触点断开（导致定时器 T33 复位，但由于 Q0.0 常开触点的自锁作用，使得线圈 Q0.0 仍处于“通电”状态），常闭触点闭合（导致定时器 T34 的启动）。在 60ms 后，定时器 T34 的常闭触点断开，线圈 Q0.0“断电”，从而达到延时断电的效果。

例 4：扩大延时范围，梯形图如图 4.15 所示。




```
LD      I0.0      //
TON T33, +3       //常开通电延时
                  //用作通电延时定时
LD      Q0.0      //断电延时的基础
AN      I0.0      //断电延时开始
TON T34, +6       //通电延时定时器
                  //用作断电延时计时
LD      T33       //
O       Q0.0      //
AN      T34       //
=       Q0.0      //用作通、断电延时
```

图 4.14　定时器的应用 2




```
LD      I0.0      //启动定时
TON T33, +100     //一级通电
                  //延时时间
                  //为 1000ms
                  //
LD      T33       //启动定时
TON T34, +200     //二级通电
                  //延时时间
                  //为 2000ms
```

图 4.15　定时器的应用 3

例 5：电机顺序起动

控制要求：3 台电机按顺序起动。电机 M1 先起动，运行 20 秒后，M2 起动，再经 30 秒后，M3 起动。

程序如图 4.16 所示。图中 3 台电机 M1、M2、M3 分别由 Q0.0、Q0.1 和 Q0.2 控制。

4.1.5　计数器指令

计数器用于累计输入脉冲的次数，是应用非常广泛的编程元件，经常用来对产品进行计数。计数器与定时器的使用基本相似，编程时输入它的预设值 PV（计数的次数），计数器累计它的脉冲输入端电位上升沿（正跳变）个数，当计数器达到预设值 PV 时，计数器动作以便 PLC 作出相应的处理。

计数器指令有 3 种：增计数 CTU、增减计数 CTUD 和减计数 CTD。

指令操作数包含 4 项：编号、预设值、脉冲输入和复位输入。

（1）编号：用计数器名称和它的常数编号（最大 255）来表示，即 Cxxx，如 C6。

C6 不仅仅是计数器的编号，它还包含两方面的变量信息：计数器位和计数器当前值。

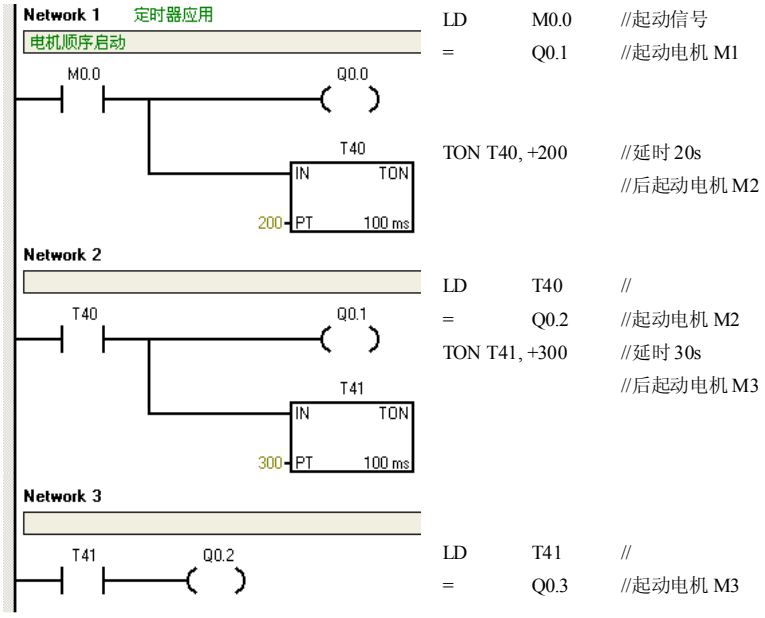


图 4.16　电机顺序起动

计数器位：表示计数器是否发生动作的状态，当计数器的当前值达到预设值 PV 时，该位被置为“1”。

计数器当前值：存储计数器当前所累计的脉冲个数，它用 16 位符号整数来表示，故最大计数值为 32767。

（2）预设值 PV：数据类型为 INT 型。寻址范围可以是 VW、IW、QW、MW、SW、SMW、LW、AIW、T、C、AC、*VD、*AC、*LD 和常数。

（3）复位输入、脉冲输入：BOOL 型，可以是 I、Q、M、SM、T、C、V、S、L 和能流。

1．增计数器

CTU，增计数器指令。首次扫描时定时器位 OFF，当前值为 0。脉冲输入的每个上升沿，计数器计数 1 次，当前值增加 1 个单位，当前值达到预设值时，计数器位 ON，当前值继续计数到 32767 停止计数。复位输入有效或执行复位指令，计数器自动复位，即计数器位 OFF，当前值为 0。

指令格式：　CTU　Cxxx,PV

例：　　CTU　C20,3

2．增减计数器

CTUD，增减计数器指令。有两个脉冲输入端：CU 输入端用于递增计数，CD 输入端用于递减计数。首次扫描时定时器位 OFF，当前值为 0。CU 输入的每个上升沿，计数器当前值增加 1 个单位，CD 输入的每个上升沿，计数器当前值减小 1 个单位，当前值达到预设值时，计数器位 ON。

增减计数器计数到 32767（最大值）后，下一个 CU 输入的上升沿将使当前值跳变为最小值（-32768）；反之，当前值达到最小值（-32768）时，下一个 CD 输入的上升沿将使当前值跳变为最大值（32767）。复位输入有效或执行复位指令，计数器自动复位，即计数器位

OFF，当前值为 0。

指令格式： CTUD Cxxx,PV

例： CTUD C30,5

3．减计数器

CTD，减计数器指令。脉冲输入端 CD 用于递减计数。首次扫描时定时器位 OFF，当前值为等于预设值 PV。计数器检测到 CD 输入的每个上升沿时，计数器当前值减小一个单位，当前值减到 0 时，计数器位 ON。复位输入有效或执行复位指令，计数器自动复位，即计数器位 OFF，当前值复位为预设值，而不是 0。

指令格式： CTD Cxxx,PV

例： CTD C40,4

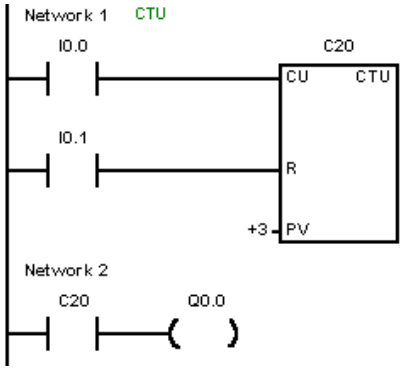
注意

（1）可以用复位指令来对 3 种计数器复位，复位指令的执行结果是：计数器位变为 OFF；计数器当前值变为 0（CTD 除外）。

（2）同一个计数器编号线圈只能使用一次。

4．应用举例

例 1：图 4.17 所示为增计数器的程序片断和时序图。




```
LD      I0.0     //计数脉冲信号输入端
LD      I0.1     //复位信号输入端
CTU C20, +3      //增计数，计数设定值
                 //为 3 个脉冲

LD      C20      //计数值达到 3，则
=       Q0.0     //将输入位置 1
```

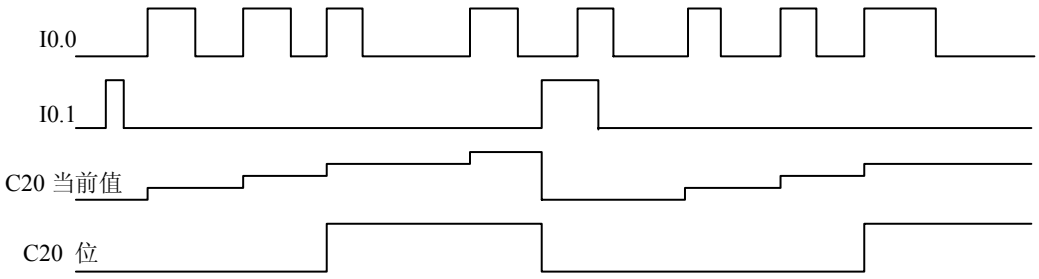
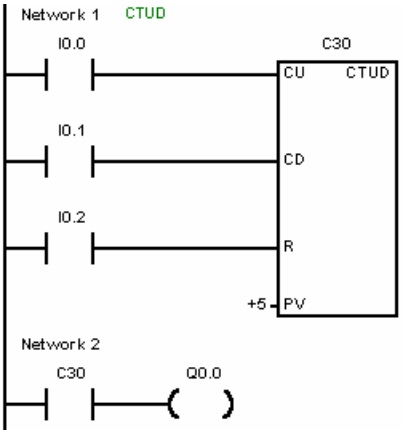


图 4.17 增计数程序及时序

例 2：图 4.18 所示为增减计数器的程序片断和时序图。

例 3：图 4.19 所示为减计数器的程序片断和时序图。




```
LD      I0.0    //增计数输入端
LD      I0.1    //减计数输入端
LD      I0.2    //复位输入端
CTUD    C30, +5 //增减计数，设定
                //脉冲数为 5
LD      C30     //计数器触点
=       Q0.0    //输出触点
```

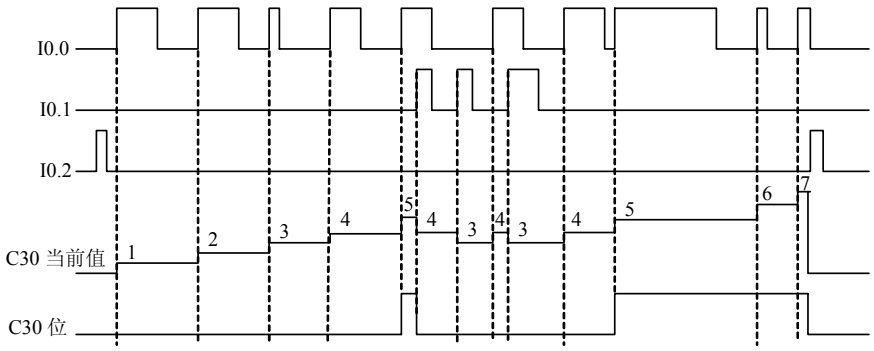
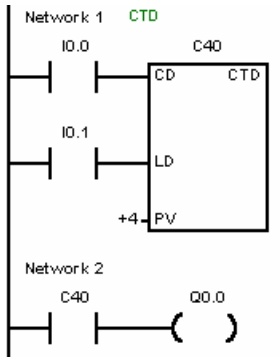


图 4.18　增减计数程序及时序




```
LD      I0.0    //减计数脉冲输入端
LD      I0.1    //复位输入端
CTD C40, +4     //减计数器，设定计数
                //脉冲数为 4
LD      C40     //装入计数器触点
=       Q0.0    //输出触点
```

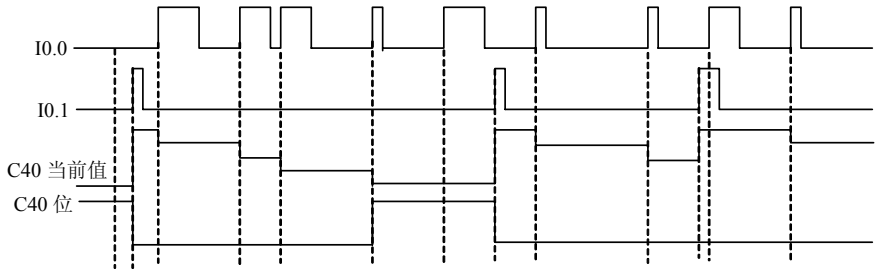
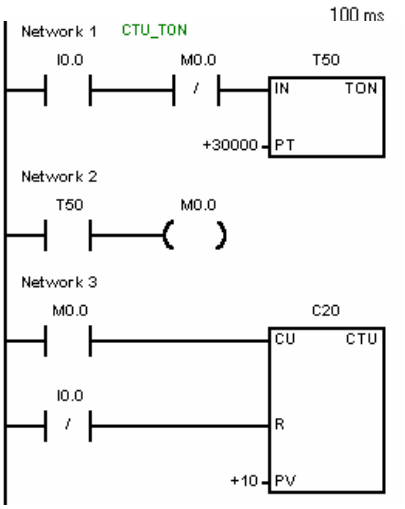


图 4.19　减计数程序及时序

例 4：循环计数。

以上三种类型的计数器如果在使用时，将计数器位的常开触点作为复位输入信号，就可以实现循环计数。

例 5：用计数器和定时器配合增加延时时间，如图 4.20 所示。试分析以下程序中实际延时为多长时间。




```
LD      I0.0        //启动通电延时
AN      M0.0        //重新启动延时
TON T50, +30000     //通电延时定时器
                    //时间设定为 3000s

LD      T50         //延时时间到
=       M0.0        //关定时器，产生一个
                    //脉冲

LD      M0.0        //每隔 3000s 输入一个
                    //脉冲
LDN I0.0            //复位输入
CTU C20, +10        //增计数器，累计脉冲
                    //的总数
```

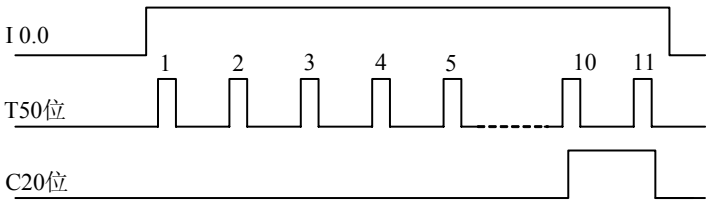


图 4.20　计数器应用例

4.1.6　比较指令

比较指令用于比较两个符号数或无符号数。

在梯形图中以带参数和运算符号的触点的形式编程，当这两个数比较式的结果为真时，该触点闭合。

在功能框图中以指令盒的形式编程，当比较式结果为真时，输出接通。

在语句表中使用 LD、A 和 O 指令进行编程，当比较式为真时，主机将栈顶置 1。

比较指令的类型有：字节比较、整数比较、双字整数比较和实数比较。

比较运算符有：=、>=、<=、>、<和< >6 种。

1．字节比较

字节比较用于比较两个字节型整数值 IN1 和 IN2 的大小，字节比较是无符号的。比较式可以是 LDB、AB 或 OB 后直接加比较运算符构成。

如 LDB=、AB< >、OB>= 等。

整数 IN1 和 IN2 的寻址范围：VB、IB、QB、MB、SB、SMB、LB、*VD、*AC、*LD 和常数。

指令格式例：

LDB=　VB10,　VB12

AB<>　MB0,　MB1

OB<=　AC1,　116

2．整数比较

整数比较用于比较两个一字长整数值 IN1 和 IN2 的大小，整数比较是有符号的（整数范围为 16#8000 和 16#7FFF 之间）。比较式可以是 LDW、AW 或 OW 后直接加比较运算符构成。

如 LDW=、AW< >、OW>= 等。

整数 IN1 和 IN2 的寻址范围：VW、IW、QW、MW、SW、SMW、LW、AIW、T、C、AC、*VD、*AC、*LD 和常数。

指令格式例：

LDW=　VW10,　VW12

AW<>　MW0,　MW4

OW<=　AC2,　1160

3．双字整数比较

双字整数比较用于比较两个双字长整数值 IN1 和 IN2 的大小，双字整数比较是有符号的（双字整数范围为 16#80000000 和 16#7FFFFFFF 之间）。比较式可以是 LDD、AD 或 OD 后直接加比较运算符构成。

如 LDD=、AD< >、OD>= 等。

双字整数 IN1 和 IN2 的寻址范围：VD、ID、QD、MD、SD、SMD、LD、HC、AC、*VD、*AC、*LD 和常数。

指令格式例：

LDD=　VD10,　VD14

AD< >　MD0,　MD8

OD<=　AC0,　1160000

LDD>=　HC0,　*AC0

4．实数比较

实数比较用于比较两个双字长实数值 IN1 和 IN2 的大小，实数比较是有符号的（负实数范围为-1.175495E-38 和-3.402823E+38，正实数范围为+1.175495E-38 和+3.402823E+38）。比较式可以是 LDR、AR 或 OR 后直接加比较运算符构成。

如：LDR=、AR< >、OR>= 等。

整数 IN1 和 IN2 的寻址范围：VD、ID、QD、MD、SD、SMD、LD、AC、*VD、*AC、*LD 和常数。

指令格式例：

LDR=　VD10,　VD18

AR<>　MD0,　MD12

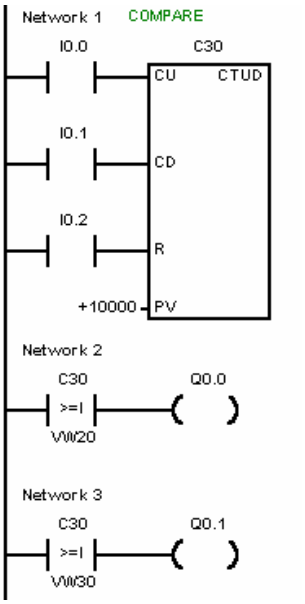
OR<=　AC1,　1160.478

AR>　　*AC1,　VD100

5．应用举例

控制要求：一自动仓库存放某种货物，最多 6000 箱，需对所存的货物进出计数。货物多于 1000 箱，灯 L1 亮；货物多于 5000 箱，灯 L2 亮。其中，L1 和 L2 分别受 Q0.0 和 Q0.1 控制，数值 1000 和 5000 分别存储在 VW20 和 VW30 字存储单元中。

本控制系统的程序如图 4.21 所示。




```
LD      I0.0          //增计数输入端
LD      I0.1          //减计数输入端
LD      I0.2          //复位输入端
CTUD    C30, +10000   //增减计数
                      //设定脉冲数
                      //为 10000
LDW>=   C30, VW20     //比较计数器
                      //当前值是否大于
                      //VW20 中的值
=       Q0.0          //输出触点
LDW>=   C30, VW30     //比较计数器
                      //当前值是否大于
                      //VW30 中的值
=       Q0.1          //输出触点
```

图 4.21　程序举例

4.2　运算指令

运算功能的加入是现代可编程序控制器与早期可编程逻辑控制器的最大区别，目前各厂商生产的各种型号 PLC 普遍具备较强的运算功能。运算包括算术运算和逻辑运算。

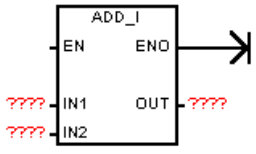
4.2.1～4.2.6 节讲述的都属于算术运算，包括加、减、乘、除和一些常用的数学函数；其余为逻辑运算，如逻辑与、逻辑或和取反等。

4.2.1　加法

加法指令是对有符号数进行相加操作。包括：整数加法、双整数加法和实数加法。

1．整数加法

+I，整数加法指令。使能输入有效时，将两个单字长（16 位）的符号整数 IN1 和 IN2 相加，产生一个 16 位整数结果 OUT。



在 LAD 和 FBD 中，以指令盒形式编程，执行结果：IN1+IN2=OUT。

在 STL 中，执行结果：IN1+OUT=OUT。

IN1 和 IN2 的寻址范围：VW、IW、QW、MW、SW、SMW、LW、AIW、T、C、AC、*VD、*AC、*LD 和常数。

OUT 的寻址范围：VW、IW、QW、MW、SW、SMW、LW、T、C、AC、*VD、*AC 和*LD。

本指令影响的特殊存储器位：SM1.0（零）、SM1.1（溢出）、SM1.2（负）。

使能流输出 ENO 断开的出错条件：SM1.1（溢出）、SM4.3（运行时间）、0006（间接寻址）。

指令格式： +I IN1,OUT

例： +I VW0,VW4

操作数	地址单元	单元长度（字节）	运算前值	运算结果值
IN1	VW0	2	2000	2000
IN2	VW4	2	3028	5028
OUT	VW4	2	3028	5028

程序实例：梯形图如图 4.22 所示。

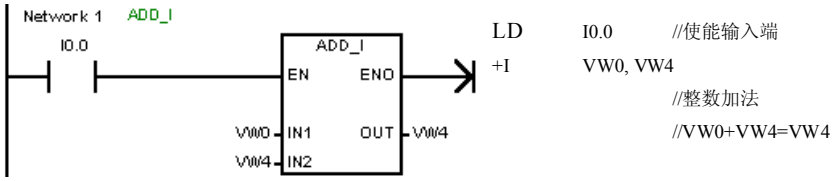
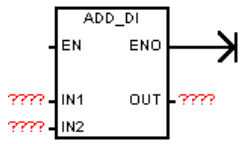


图 4.22 整数加法例

2．双整数加法

+D，双整数加法指令。使能输入有效时，将两个双字长（32 位）的符号双整数 IN1 和 IN2 相加，产生一个 32 位双整数结果 OUT。



在 LAD 和 FBD 中，以指令盒形式编程，执行结果：IN1+IN2=OUT。

在 STL 中，执行结果：IN1+OUT=OUT。

IN1 和 IN2 的寻址范围：VD、ID、QD、MD、SD、SMD、LD、HC、AC、*VD、*AC、*LD 和常数。

OUT 的寻址范围：VD、ID、QD、MD、SD、SMD、LD、AC、*VD、*AC 和*LD。

本指令影响的特殊存储器位：SM1.0（零）、SM1.1（溢出）、SM1.2（负）。

使能流输出 ENO 断开的出错条件：SM1.1（溢出）、SM4.3（运行时间）、0006（间接寻址）。

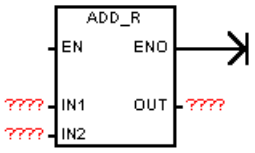
操作数	地址单元	单元长度（字节）	运算前值	运算结果值
IN1	VD0	4	120000	120000
IN2	VD4	4	30281	150281
OUT	VD4	4	30281	150281

指令格式：　+D　IN1,OUT

例：　+D　VD0,VD4

3．实数加法

+R，实数加法指令。使能输入有效时，将两个双字长（32 位）的实数 IN1 和 IN2 相加，产生一个 32 位实数结果 OUT。



在 LAD 和 FBD 中，以指令盒形式编程，执行结果：IN1+IN2=OUT。

在 STL 中，执行结果：IN1+OUT=OUT。

IN1 和 IN2 的寻址范围：VD、ID、QD、MD、SD、SMD、LD、AC、*VD、*AC、*LD 和常数。

OUT 的寻址范围：VD、ID、QD、MD、SD、SMD、LD、AC、*VD、*AC 和*LD。

本指令影响的特殊存储器位：SM1.0（零）、SM1.1（溢出）、SM1.2（负）。

使能流输出 ENO 断开的出错条件：SM1.1（溢出）、SM4.3（运行时间）、0006（间接寻址）。

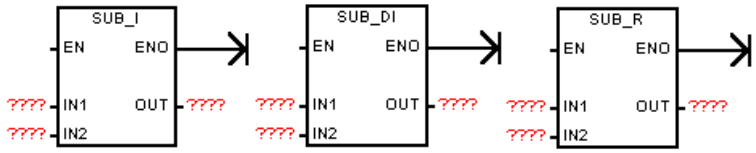
操作数	地址单元	单元长度（字节）	运算前值	运算结果值
IN1	VD0	4	200.03	200.03
IN2	VD4	4	302.815	502.845
OUT	VD4	4	302.815	502.815

指令格式：+R　IN1,OUT

例：　+R　VD0,VD4

4.2.2　减法

减法指令是对有符号数进行相减操作，包括整数减法、双整数减法和实数减法。这 3 种减法指令与所对应的加法指令除运算法则不同之外，其他方面基本相同。



在 LAD 和 FBD 中，以指令盒形式编程，执行结果：IN1-IN2=OUT。

在 STL 中，执行结果：　OUT- IN2=OUT。

指令格式：　-I　IN2，OUT　（整数减法）

　　-D　IN2，OUT　（双整数减法）

　　-R　IN2，OUT　（实数减法）

例：　-I　AC0，VW4

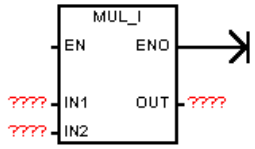
操作数	地址单元	单元长度（字节）	运算前值	运算结果值
IN1	VW4	2	3000	1000
IN2	AC0	2	2000	2000
OUT	VW4	2	3000	1000

4.2.3 乘法

乘法指令是对有符号数进行相乘运算，包括整数乘法、完全整数乘法、双整数乘法和实数乘法。

1．整数乘法

*I，整数乘法指令。使能输入有效时，将两个单字长（16 位）的符号整数 IN1 和 IN2 相乘，产生一个 16 位整数结果 OUT。



运算结果如果大于 16 位二进制表示的范围，则产生溢出。

在 LAD 和 FBD 中，以指令盒形式编程，执行结果：IN1*IN2=OUT。

在 STL 中，执行结果：IN1*OUT=OUT。

IN1 和 IN2 的寻址范围：VW、IW、QW、MW、SW、SMW、LW、AIW、T、C、AC、*VD、*AC、*LD 和常数。

OUT 的寻址范围：VW、IW、QW、MW、SW、SMW、LW、T、C、AC、*VD、*AC 和*LD。

本指令影响的特殊存储器位：SM1.0（零）、SM1.1（溢出）、SM1.2（负）、SM1.3（被 0 除）。

使能流输出 ENO 断开的出错条件：SM1.1（溢出）；SM4.3（运行时间）；0006（间接寻址）。

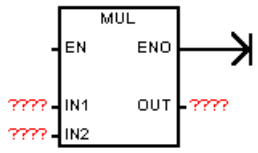
指令格式：*I　　IN1,OUT

例：　　*I　　VW0,AC0

操作数	地址单元	单元长度（字节）	运算前值	运算结果值
IN1	VW0	2	20	20
IN2	AC0	2	400	8000
OUT	AC0	2	400	8000

2．完全整数乘法

MUL，完全整数乘法指令。使能输入有效时，将两个单字长（16 位）的符号整数 IN1 和 IN2 相乘，产生一个 32 位双整数结果 OUT。



在 LAD 和 FBD 中，以指令盒形式编程，执行结果：IN1*IN2=OUT。

在 STL 中，执行结果：IN1*OUT=OUT（32 位结果的低 16 位曾被用作乘数）。

IN1 和 IN2 的寻址范围：VW、IW、QW、MW、SW、SMW、LW、AIW、T、C、

AC、*VD、*AC、*LD 和常数。

OUT 的寻址范围：VD、ID、QD、MD、SD、SMD、LD、AC、*VD、*AC 和*LD。

本指令影响的特殊存储器位：SM1.0（零）、SM1.1（溢出）、SM1.2（负）、SM1.3（被 0 除）。

使能流输出 ENO 断开的出错条件：SM1.1（溢出）、SM4.3（运行时间）、0006（间接寻址）。

指令格式：MUL　IN1,OUT

例：　MUL　AC0,VD10

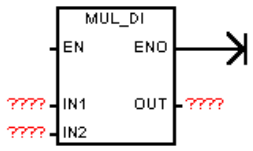
操作数	地址单元	单元长度（字节）	运算前值	运算结果值
IN1	AC0	2	20	20
IN2	VW12	2	400	8000
OUT	VD10	4	400	8000

注意

在梯形图中用本指令编程时，如果 OUT 为存储器单元，则必须考虑到存储器的编址特点。输入数据 IN2 与 OUT 的低 16 位用的是同一单元。

3．双整数乘法

*D，双整数乘法指令。使能输入有效时，将两个双字长（32 位）的符号整数 IN1 和 IN2 相乘，产生一个 32 位双整数结果 OUT。



运算结果如果大于 32 位二进制表示的范围，则产生溢出。

在 LAD 和 FBD 中，以指令盒形式编程，执行结果：IN1*IN2=OUT。

在 STL 中，执行结果：IN1*OUT=OUT。

IN1 和 IN2 的寻址范围：VD、ID、QD、MD、SD、SMD、LD、HC、AC、*VD、*AC、*LD 和常数。

OUT 的寻址范围：VD、ID、QD、MD、SD、SMD、LD、AC、*VD、*AC 和*LD。

本指令影响的特殊存储器位：SM1.0（零）、SM1.1（溢出）、SM1.2（负）、SM1.3（被 0 除）。

使能流输出 ENO 断开的出错条件：SM1.1（溢出）、SM4.3（运行时间）、0006（间接寻址）。

指令格式：*D　IN1,OUT

例：　*D　VD0,AC0

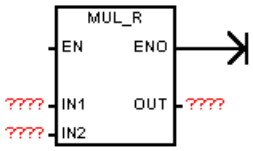
操作数	地址单元	单元长度（字节）	运算前值	运算结果值
IN1	VD0	4	200	200
IN2	AC0	4	400	80000
OUT	AC0	4	400	80000

4．实数乘法

*R，实数乘法指令。使能输入有效时，将两个双字长（32 位）的实数 IN1 和 IN2 相

乘，产生一个 32 位实数结果 OUT。

运算结果如果大于 32 位二进制表示的范围，则产生溢出。溢出以及输入非法参数，或运算中产生非法值，都会使特殊标志位 SM1.1 置位。



在 LAD 和 FBD 中，以指令盒形式编程，执行结果：IN1*IN2=OUT。

在 STL 中，执行结果：IN1*OUT=OUT。

IN1 和 IN2 的寻址范围：VD、ID、QD、MD、SD、SMD、LD、AC、*VD、*AC、*LD 和常数。

OUT 的寻址范围：VD、ID、QD、MD、SD、SMD、LD、AC、*VD、*AC 和*LD。

本指令影响的特殊存储器位：SM1.0（零）、SM1.1（溢出）、SM1.2（负）、SM1.3（被 0 除）。

使能流输出 ENO 断开的出错条件：SM1.1（溢出）；SM4.3（运行时间）；0006（间接寻址）。

指令格式：*R　　IN1,OUT

例：　　*R　　VD0,AC0

操作数	地址单元	单元长度（字节）	运算前值	运算结果值
IN1	VD0	4	20.2	20.2
IN2	AC0	4	0.8	16.16
OUT	AC0	4	0.8	16.16

4.2.4 除法

除法指令是对有符号数进行相除操作。包括：整数除法、整数完全除法、双整数除法和实数除法。这 3 种除法指令与所对应的乘法指令除运算法则不同之外，其他方面基本相同。

在 LAD 和 FBD 中，以指令盒形式编程，执行结果：IN1/IN2=OUT。

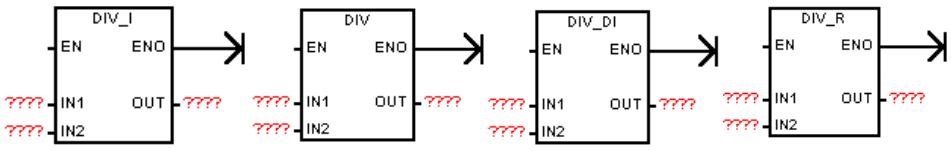
在 STL 中，执行结果： OUT/ IN2=OUT。

指令格式：/I　　IN2,　OUT　（整数除法）

DIV　IN2,　OUT　（整数完全除法）

/D　　IN2,　OUT　（双整数除法）

/R　　IN2,　OUT　（实数除法）

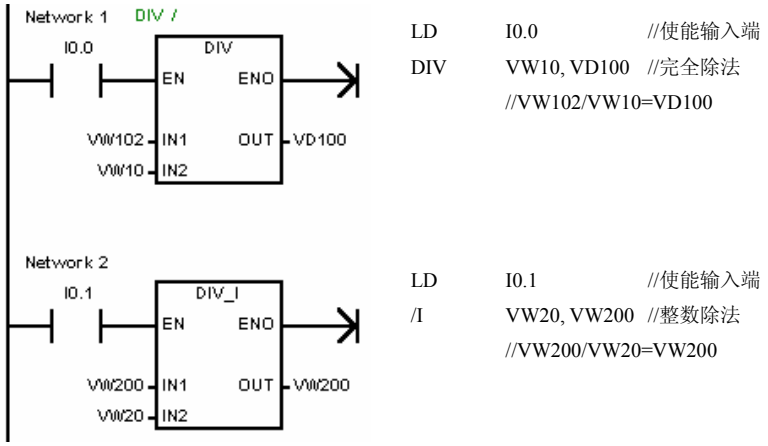


在整数除法中，两个 16 位的整数相除，产生一个 16 位的整数商，不保留余数。双整数除法也是同样的过程，只是位数变为 32 位。

在整数完全除法中，两个 16 位的符号整数相除，产生一个 32 位结果，其中，低 16 位为商，高 16 位为除数。32 位结果的低 16 位在运算前被兼用存放被除数。

例：DIV　　VW10,　VD100
　　/I　　VW20,　VW200

两条指令的编程及执行情况比较如图 4.23 所示。




```
LD      I0.0          //使能输入端
DIV     VW10, VD100   //完全除法
        //VW102/VW10=VD100

LD      I0.1          //使能输入端
/I      VW20, VW200   //整数除法
        //VW200/VW20=VW200
```

图 4.23　除法指令应用

对于完全除法指令：

操作数	地址单元	单元长度（字节）	运算前值	运算结果值	
IN1	VW102	2	2003	50	
IN2	VW10	2	40	40	
OUT	VD100	4	203	VW100	3
				VW102	50

对于除法指令：

操作数	地址单元	单元长度（字节）	运算前值	运算结果值
IN1	VW200	2	2003	50
IN2	VW20	2	40	40
OUT	VW200	2	400	50

4.2.5　数学函数指令

本小节介绍几个常用的数学函数指令（也称数学功能指令）：平方根、自然对数、指数、正弦、余弦和正切。

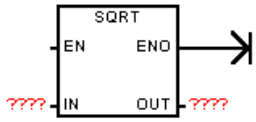
这几条指令中的 IN 寻址范围：VD、ID、QD、MD、SD、SMD、LD、AC、*VD、*AC、*LD 和常数。

OUT 的寻址范围：VD、ID、QD、MD、SD、SMD、LD、AC、*VD、*AC 和*LD。

运算输入输出数据都为实数。结果如果大于 32 位二进制表示的范围，则产生溢出。

1．平方根

SQRT，平方根指令。把一个双字长（32 位）的实数 IN 开平方，得到 32 位的实数结果。



在 LAD 和 FBD 中，以指令盒形式编程，执行结果：SQRT(IN)=OUT。

在 STL 中，执行结果：SQRT(IN)=OUT。

本指令影响的特殊存储器位：SM1.0（零）、SM1.1（溢出和非法值）、SM1.2（负）。

使能流输出 ENO 断开的出错条件：SM1.1（溢出）、SM4.3（运行时间）、0006（间接寻址）。

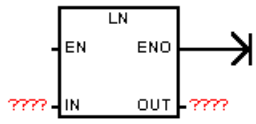
指令格式： SQRT IN,OUT

例： SQRT VD0,AC0

2．自然对数

LN，自然对数指令。将一个双字长（32 位）的实数 IN 取自然对数，得到 32 位的实数结果。

当求解以 10 为底的常用对数时，可以用（/R）DIV_R 指令将自然对数除以 2.302585 即可（LN10 的值约为 2.302585）。



在 LAD 和 FBD 中，以指令盒形式编程，执行结果：LN(IN)=OUT。

在 STL 中，执行结果：LN(IN)=OUT。

本指令影响的特殊存储器位：SM1.0（零）、SM1.1（溢出和非法值）、SM1.2（负）、SM4.3（运行时间）。

使能流输出 ENO 断开的出错条件：SM1.1（溢出）、0006（间接寻址）。

指令格式： LN IN,OUT

例： LN VD0,AC0

应用实例：求以 10 为底的 50（存于 VD0）的常用对数，结果放到 AC0。

本运算程序如图 4.24 所示。

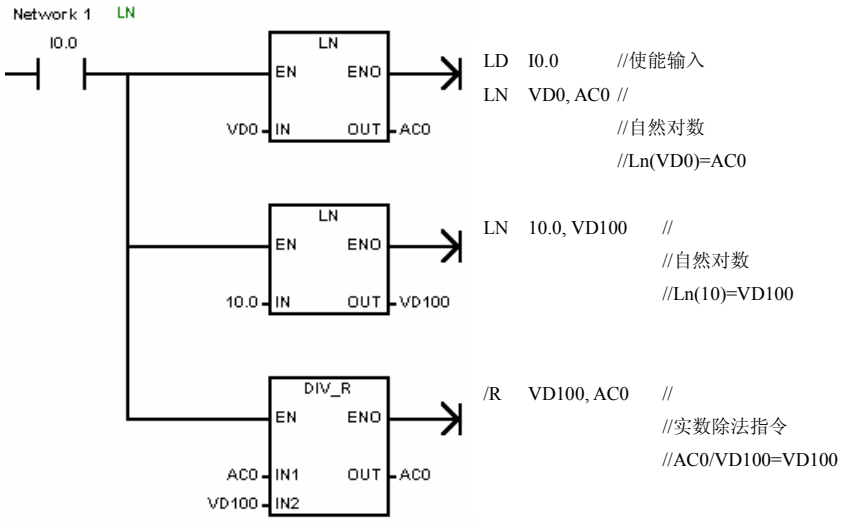
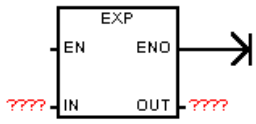


图 4.24 自然对数的应用

3．指数

EXP，指数指令。将一个双字长（32 位）的实数 IN 取以 e 为底的指数，得到 32 位的实数结果 OUT。

在 LAD 和 FBD 中，以指令盒形式编程，执行结果：EXP(IN)=OUT。



在 STL 中，执行结果：EXP(IN)=OUT。

本指令影响的特殊存储器位：SM1.0（零）、SM1.1（溢出和非法值）、SM1.2（负）、SM4.3（运行时间）。

使能流输出 ENO 断开的出错条件：SM1.1（溢出）、0006（间接寻址）。

指令格式：　EXP　IN,　OUT

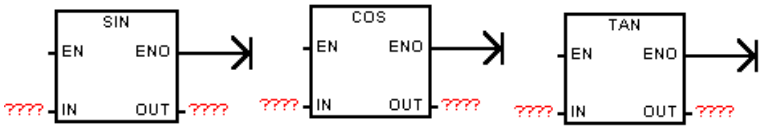
例：　　EXP　VD0, AC0

若求解以任意常数为底的指数，可以用指数指令和自然对数指令相配合来完成。例如：18 的 6 次方=18 ^ 6=EXP(6*LN(18))

4．正弦、余弦、正切

SIN、COS、TAN，即正弦、余弦、正切指令。将一个双字长（32 位）的实数弧度值 IN 分别取正弦、余弦、正切，各得到 32 位的实数结果。

如果已知输入值为角度，要先将角度值转化为弧度值，方法：使用（*R）MUL_R 指令用角度值乘以 π/180° 即可。



在 LAD 和 FBD 中，以指令盒形式编程，执行结果如：SIN(IN)=OUT。

在 STL 中，执行结果如：SIN(IN)=OUT。

这三条指令影响的特殊存储器位：SM1.0（零）、SM1.1（溢出和非法值）、SM1.2（负）；SM4.3（运行时间）。

使能流输出 ENO 断开的出错条件：SM1.1（溢出）、0006（间接寻址）。

指令格式：　SIN　IN,　OUT　（正弦）

　　COS　IN,　OUT　（余弦）

　　TAN　IN,　OUT　（正切）

例：　　TAN　VD0, AC0

应用实例：求 COS160° 的值，如图 4.25 所示。

4.2.6　增减

增、减指令，又称自增和自减，是对无符号或有符号表整数进行自动增加或减小一个单位的操作，数据长度可以是字节、字或双字。

1．字节增和字节减

INCB，字节增指令。使能输入有效时，把一字节长的无符号输入数（IN）加 1，得到一字节的无符号输出结果 OUT。

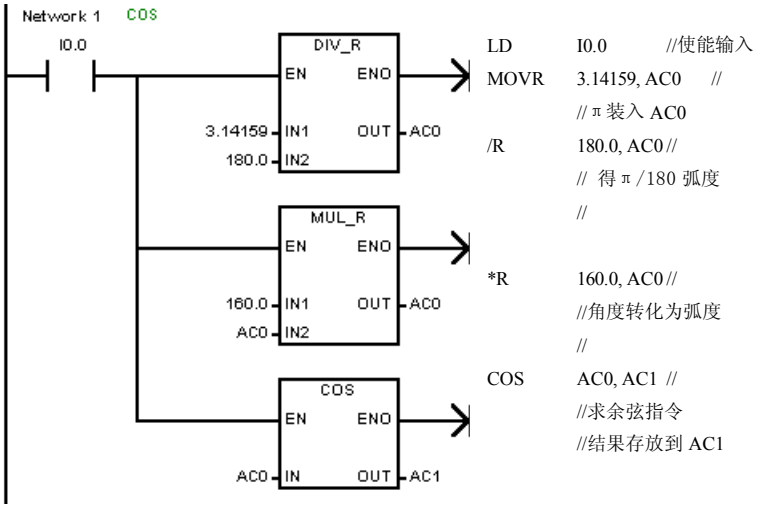


图 4.25　三角函数应用例

DECB，字节减指令。使能输入有效时，把一字节长的无符号输入数（IN）减 1，得到一字节的无符号输出结果 OUT。

IN 的寻址范围：VB、IB、QB、MB、SB、SMB、LB、AC、*VD、*AC、*LD 和常数。

OUT 的寻址范围：VB、IB、QB、MB、SB、SMB、LB、AC、*VD、*AC 和*LD。

在 LAD 和 FBD 中，以指令盒形式编程，执行结果：IN+1=OUT 和 IN-1=OUT。

在 STL 中，执行结果：OUT+1=OUT 和 OUT-1=OUT。

本指令影响的特殊存储器位：SM1.0（零）、SM1.1（溢出）。

使能流输出 ENO 断开的出错条件：SM1.1（溢出）、SM4.3（运行时间）、0006（间接寻址）。

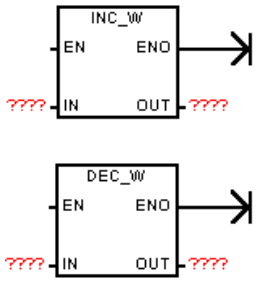
指令格式：　INCB　OUT　（字节增指令）

　　DECB　OUT　（字节减指令）

例：　　INCB　VB40

　　DECB　AC0

2．字增和字减



INCW，字增指令。使能输入有效时，把一字长（16 位）的有符号输入数（IN）加 1，得到 32 位的有符号输出结果 OUT。

DECW，字减指令。使能输入有效时，把一字长的有符号输入数（IN）减 1，得到一字长的有符号输出结果 OUT。

IN 的寻址范围：VW、IW、QW、MW、SW、SMW、LW、AIW、T、C、AC、*VD、*AC、*LD 和常数。

OUT 的寻址范围：VW、IW、QW、MW、SW、SMW、LW、

T、C、AC、*VD、*AC 和*LD。

在 LAD 和 FBD 中，以指令盒形式编程，执行结果：IN+1=OUT 和 IN−1=OUT。

在 STL 中，执行结果：OUT+1=OUT 和 OUT−1=OUT。

本指令影响的特殊存储器位：SM1.0（零）、SM1.1（溢出）、SM1.2（负）。

使能流输出 ENO 断开的出错条件：SM1.1（溢出）、SM4.3（运行时间）、0006（间接寻址）。

指令格式：　INCW　OUT　（字增指令）

　　DECW　OUT　（字减指令）

例：　　INCW　VW40

　　DEC　AC0

此处字增指令的执行情况如下：

操作数	地址单元	单元长度（字节）	运算前值	运算结果值
IN	VW40	2	2000	2001
OUT	AC0	2	2000	2001

3．双字增和双字减

INCD，双字增指令。使能输入有效时，把双字长（32 位）的有符号输入数（IN）加 1，得到双字长的有符号输出结果 OUT。

DECD，双字减指令。使能输入有效时，把双字长的有符号输入数（IN）减 1，得到双字长的有符号输出结果 OUT。

IN 的寻址范围：VD、ID、QD、MD、SD、SMD、LD、HC、AC、*VD、*AC、*LD 和常数。

OUT 的寻址范围：VD、ID、QD、MD、SD、SMD、LD、AC、*VD、*AC 和*LD。

在 LAD 和 FBD 中，以指令盒形式编程，执行结果：IN+1=OUT 和 IN−1=OUT。

在 STL 中，执行结果：OUT+1=OUT 和 OUT-1=OUT。

本指令影响的特殊存储器位：SM1.0（零）、SM1.1（溢出）、SM1.2（负）。

使能流输出 ENO 断开的出错条件：SM1.1（溢出）、SM4.3（运行时间）、0006（间接寻址）。

指令格式：　INCD　OUT　（双字增指令）

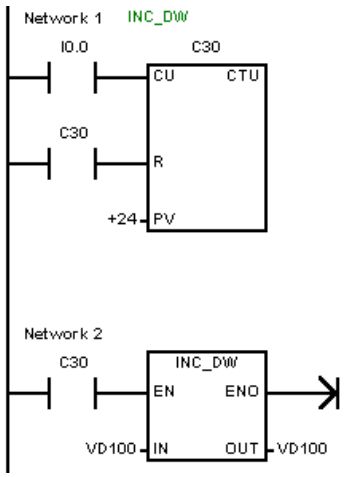
　　DECD　OUT　（双字减指令）

例：　　INCD　VB40

　　DECD　AC0

4．应用实例

控制要求：食品加工厂对饮料生产线上的盒装饮料进行计数，每 24 盒为一箱，要求能记录生产的箱数。程序如图 4.26 所示。




```
LD      I0.0      //增脉冲输入端
LD      C30       //复位输入端
                  //循环计数
CTU C30, +24      //增计数指令
                  //设定脉冲数为 24

LD      C30       //装入计数器触点
                  //作为双字增的
                  //脉冲输入
INCD    VD100     //双字增指令
                  //
```

图 4.26　增减指令的应用

说明：程序中利用 VD100 存储箱数，其初始值需预设为 0。用 I0.0 检测脉冲，用增计数器 C30 的常开触点作为计数器的复位输入，从而构成循环计数器，计数器 C30 每检测到 24 个脉冲输入双字增指令执行一次，VD100 中的数据加 1，同时计数器 C30 自动复位进入下一计数过程。

4.2.7　逻辑运算

逻辑运算对逻辑数（无符号数）进行处理，按运算性质包括逻辑与、逻辑或、逻辑异或、取反等。按参与运算的操作数的长度可分为字节、字和双字逻辑运算操作。

在 LAD 和 FBD 中，以指令盒形式编程，执行结果为：对两数 IN1 与 IN2 逻辑运算，或对单数 OUT 逻辑处理（取反），结果由 OUT 输出。

在 STL 中，执行结果：对两数 IN 与 OUT 逻辑运算，或对单数 OUT 逻辑处理（取反），结果由 OUT 输出。

所有逻辑运算指令影响的特殊存储器位：SM1.0（零）。

使能流输出 ENO 断开的出错条件：SM4.3（运行时间）、0006（间接寻址）。

1．字节逻辑运算

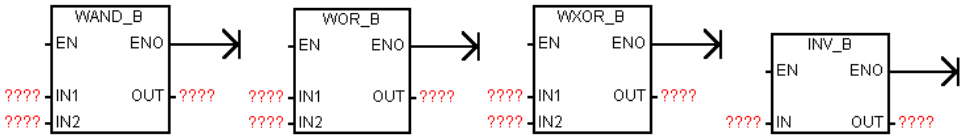
字节逻辑运算包括字节与、字节或、字节异或、字节取反。

ANDB，字节与指令。使能输入有效时，把两个 1 字节长的输入逻辑数按位相与，得到 1 字节的逻辑数输出结果 OUT。

ORB，字节或指令。使能输入有效时，把两个 1 字节长的输入逻辑数按位求或，得到 1 字节的逻辑数输出结果 OUT。

XORB，字节异或指令。使能输入有效时，把两个 1 字节长的输入逻辑数按位求异或，得到 1 字节的逻辑数输出结果 OUT。

INVB，字节取反指令。使能输入有效时，把一个 1 字节长的输入逻辑数 IN 按位求反，得到 1 字节的逻辑数输出结果 OUT。



IN1、IN2、IN 的寻址范围：VB、IB、QB、MB、SB、SMB、LB、AC、*VD、*AC、*LD 和常数。

OUT 的寻址范围：VB、IB、QB、MB、SB、SMB、LB、AC、*VD、*AC 和*LD。

指令格式：　ANDB　IN1,　OUT　　（字节与指令）

　　ORB　IN1,　OUT　　（字节或指令）

　　XORB　IN1,　OUT　　（字节异或指令）

　　INVB　OUT　　（字节取反指令）

例：（1）　ANDB　VB0,　AC1

（2）　ORB　VB0,　AC0

（3）　XORB　VB0,　AC2

（4）　INVB　VB10

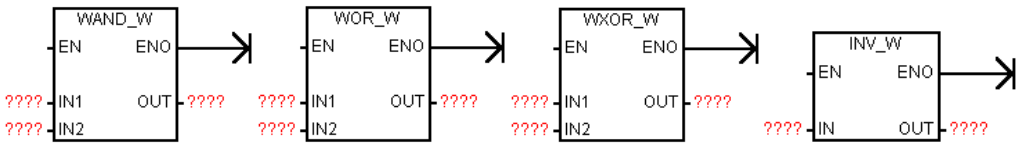
这 4 条指令的执行情况分别如表 4.15 所示（各单元内容都用二进制表示）。

表 4.15　指令执行情况表

指令	操作数	地址单元	单元长度（字节）	运算前值	运算结果值
（1）	IN1	VB0	1	01010011	01010011
	IN2(OUT)	AC1	1	11110001	01010001
（2）	IN1	VB0	1	01010011	01010011
	IN2(OUT)	AC0	1	00110110	01110111
（3）	IN1	VB0	1	01010011	01010011
	IN2(OUT)	AC2	1	11011010	10001001
（4）	IN(OUT)	VB10	1	01010011	10101100

2．字逻辑运算

字逻辑运算包括字与、字或、字异或、字取反。



ANDW，字与指令。使能输入有效时，把两个 1 字长（16 位）的输入逻辑数按位相与，得到 1 字长的逻辑数输出结果 OUT。

ORW，字或指令。使能输入有效时，把两个 1 字长（16 位）的输入逻辑数按位求或，得到 1 字长的逻辑数输出结果 OUT。

XORW，字异或指令。使能输入有效时，把两个 1 字长的输入逻辑数按位求异或，得到 1 字长的逻辑数输出结果 OUT。

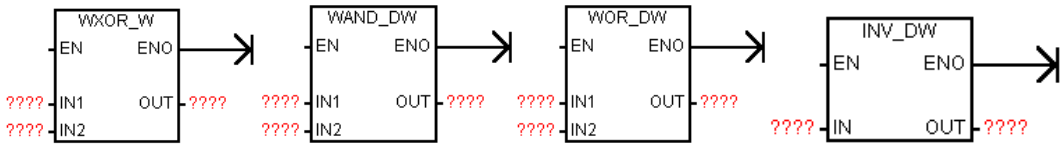
INVW，字取反指令。使能输入有效时，把一个 1 字长的输入逻辑数 IN 按位求反，得到 1 字长的逻辑数输出结果 OUT。

IN1、IN2、IN 的寻址范围：VW、IW、QW、MW、SW、SMW、LW、AIW、T、C、AC、*VD、*AC、*LD 和常数。

OUT 的寻址范围：VW、IW、QW、MW、SW、SMW、LW、T、C、AC、*VD、*AC 和*LD。

指令格式：
```
ANDW   IN1,  OUT      （字与指令）
ORW    IN1,  OUT      （字或指令）
XORW   IN1,  OUT      （字异或指令）
INVW   OUT      （字取反指令）
```
例：
```
ANDW   VB0,  AC1
ORW    VB0,  AC0
XORW   AC1,  AC2
INVW   LB10
```
3．双字逻辑运算

字逻辑运算包括双字与、双字或、双字异或、双字取反。



ANDD，双字与指令。使能输入有效时，把两个双字长（32 位）的输入逻辑数按位相与，得到双字长的逻辑数输出结果 OUT。

ORD，双字或指令。使能输入有效时，把两个双字长（32 位）的输入逻辑数按位求或，得到双字长的逻辑数输出结果 OUT。

XORD，双字异或指令。使能输入有效时，把两个双字长的输入逻辑数按位求异或，得到双字长的逻辑数输出结果 OUT。

INVD，双字取反指令。使能输入有效时，把一个双字长的输入逻辑数 IN 按位求反，得到双字长的逻辑数输出结果 OUT。

IN1、IN2、IN 的寻址范围：VD、ID、QD、MD、SMD、LD、AC、HC、*VD、*AC、*LD 和常数。

OUT 的寻址范围：VD、ID、QD、MD、SMD、LD、AC、*VD、*AC 和*LD。

指令格式：
```
ANDD   IN1,  OUT      （双字与指令）
ORD    IN1,  OUT      （双字或指令）
XORD   IN1,  OUT      （双字异或指令）
INVD   OUT            （双字取反指令）
```
例：
```
ANDD   VB0,  AC1
ORD    VB0,  AC0
XORD   AC1,  AC2
```

INVD　　LB10

4.3　其他数据处理指令

此类指令主要涉及对数据的非数值运算类操作，主要包括传送、移位、字节交换、循环移位和填充。

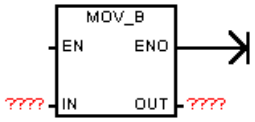
4.3.1　传送类指令

传送类指令可用来在各存储单元之间进行一个或多个数据的传送。按指令一次所传送数据的个数可分为单一传送指令和块传送指令。

1．单一传送

指令可用来进行一个数据的传送，数据类型可以是字节、字、双字和实数。本类指令使能流输出 ENO 断开的出错条件：SM4.3（运行时间）、0006（间接寻址）。

（1）MOVB，字节传送指令。



使能输入有效时，把一个单字节无符号数据由 IN 传送到 OUT 所指的字节存储单元。

IN 的寻址范围：VB、IB、QB、MB、SB、SMB、LB、AC、*VD、*AC、*LD 和常数。

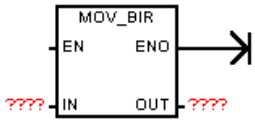
OUT 的寻址范围：VB、IB、QB、MB、SB、SMB、LB、AC、*VD、*AC 和*LD。

指令格式：　MOVB　IN1,　OUT

例：　　MOVB　VB0,　QB0

（2）BIR，传送字节立即读指令。

使能输入有效时，立即读取单字节物理输入区数据 IN，并传送到 OUT 所指的字节存储单元。



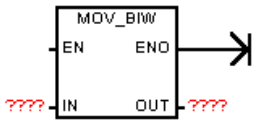
IN 的寻址范围：IB

OUT 的寻址范围：VB、IB、QB、MB、SB、SMB、LB、AC、*VD、*AC 和*LD。

指令格式：　BIR　　IN1,　OUT

例：　　BIR　　IB0,　VB10

（3）BIW，传送字节立即写指令。



使能输入有效时，立即将 IN 单元的字节数据写到 OUT 所指的物理输出区。

IN 的寻址范围：VB、IB、QB、MB、SB、SMB、LB、AC、*VD、*AC、*LD 和常数。

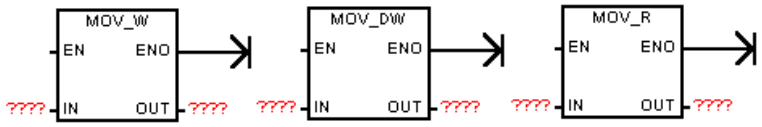
OUT 的寻址范围：QB

指令格式：　BIW　　IN1,　OUT

例：　　BIW　　VB0,　QB0

（4）MOVW，字传送指令。

使能输入有效时，把一个 1 字长有符号整数由 IN 传送到 OUT 所指的字存储单元。



（5）MOVD，双字传送指令。

使能输入有效时，把一个双字长有符号数据由 IN 传送到 OUT 所指的双字存储单元。

（6）MOVR，实数传送指令。

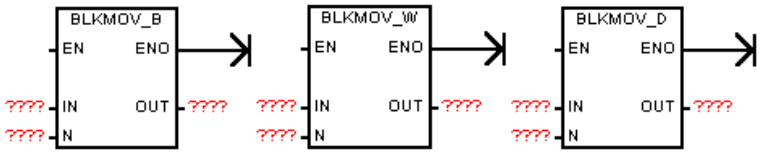
使能输入有效时，把一个 32 位实数由 IN 传送到 OUT 所指的双字长存储单元。

2．块传送

指令可用来进行一次多个（最多 255 个）数据的传送，数据块类型可以是字节块、字块、双字块。

3 条指令中 N 的寻址范围都是：VB、IB、QB、MB、SB、SMB、LB、AC、*VD、*AC、*LD 和常数。

使 ENO 断开的出错条件：SM4.3（运行时间）、0006（间接寻址）、0091（数超界）。



（1）BMB，字节块传送指令。

使能输入有效时，把从输入字节 IN 开始的 N 个字节型数据传送到从 OUT 开始的 N 个字节存储单元。

IN、OUT 的寻址范围：VB、IB、QB、MB、SB、SMB、LB、*VD、*AC、*LD。

指令格式： BMB IN1, OUT, N

例： BMB VB0, LB0, 30

（2）BMW，字块传送指令。

使能输入有效时，把从输入字 IN 开始的 N 个字型数据传送到从 OUT 开始的 N 个字存储单元。

IN 的寻址范围：VW、IW、QW、MW、SW、SMW、LW、AIW、T、C、*VD、*AC、*LD。

OUT 的寻址范围：VW、IW、QW、MW、SW、SMW、LW、AIW、T、C、AQW、*VD、*AC、*LD。

指令格式： BMW IN1, OUT, N

例： BMW IW0, QW0, 30

（3）BMD，双字块传送指令。

使能输入有效时，把从输入双字 IN 开始的 N 个双字型数据传送到从 OUT 开始的 N 个双字存储单元。

IN、OUT 的寻址范围：VD、ID、QD、MD、SD、SMD、LD、*VD、*AC、*LD。

指令格式： BMD IN1, OUT, N

例： BMD VD0, LD0, 10

4.3.2 移位指令

移位指令都是对无符号数进行的处理，执行时只考虑要移位的存储单元的每一位数字状态，而不管数据的值的大小。本类指令在一个数字量输出点对应多个相对固定状态的情况下有广泛的应用。

1．左移和右移

左移和右移根据所移位的数的长度分别又可分为字节型、字型、双字型。

移位操作特点：

- 移位数据存储单元的移出端与 SM1.1（溢出）相连，所以最后被移出的位被放到 SM1.1 位存储单元。
- 移位时，移出位进入 SM1.1，另一端自动补 0。例如在右移时，移位数据的最右端位移入 SM1.1，左端每次补 0。SM1.1 始终存放最后一次被移出的位。
- 移位次数与移位数据的长度有关，如果所需移位次数大于移位数据的位数，则超出的次数无效。如字左移时，若移位次数设定为 20，则指令实际执行结果是只能移位 16 次，而不是设定值 20 次。
- 如果移位操作使数据变为 0，则零存储器位（SM1.0）自动置位。
- 移位次数 N 为字节型数据。

移位指令影响的特殊存储器位：SM1.0（零）、SM1.1（溢出）。

使能流输出 ENO 断开的出错条件：SM4.3（运行时间）、0006（间接寻址）。

（1）字节左移和字节右移

SLB 和 SRB，字节左移和字节右移。使能输入有效时，把字节型输入数据 IN 左移或右移 N 位后，再将结果输出到 OUT 所指的字节存储单元。最大实际可移位次数为 8。

指令格式：　SLB　　OUT，　N　（字节左移）

　　SRB　　OUT，　N　（字节右移）

例：　　SLB　　MB0，　2

　　SRB　　LB0，　3

以第一条指令为例，指令执行情况如表 4.16 所示。

表 4.16　指令 SLB 执行结果

移位次数	地址	单元内容	位 SM1.1	说明
0	MB0	10110101	x	移位前
1	MB0	01101010	1	数左移，移出位 1 进入 SM1.1，右端补 0
2	MB0	11010100	0	数左移，移出位 0 进入 SM1.1，右端补 0

（2）字左移和字右移。

SLW 和 SRW，字左移和字右移。指令盒与字节移位比较，只有名称变为 SHR_W 和 SHR_W。使能输入有效时，把字型输入数据 IN 左移或右移 N 位后，再将结果输出到 OUT 所指的字存储单元。最大实际可移位次数为 16。

指令格式： SLW OUT, N （字左移）
SRW OUT, N （字右移）
例： SLW MW0, 2
SRW LW0, 3

以第二条指令为例，指令执行情况如表 4.17 所示。

表 4.17 指令 SRW 执行结果

移位次数	地址	单元内容	位 SM1.1	说明
0	LW0	1011010100110011	x	移位前
1	LW0	0101101010011001	1	右移，1 进入 SM1.1，左端补 0
2	LW0	0010110101001100	1	右移，1 进入 SM1.1，左端补 0
3	LW0	0001011010100110	0	右移，0 进入 SM1.1，左端补 0

（3）双字左移和双字右移。

SLD 和 SRD，双字左移和双字右移。指令盒与字节移位比较，只有名称变为 SHL_DW 和 SHR_DW，其他部分完全相同。使能输入有效时，把双字型输入数据 IN 左移或右移 N 位后，再将结果输出到 OUT 所指的双字存储单元。最大实际可移位次数为 32。

指令格式： SLD OUT, N （双字左移）
SRD OUT, N （双字右移）
例： SLD MD0, 2
SRD LD0, 3

2．循环左移、循环右移

循环左移和循环右移根据所循环移位的数的长度分别又可分为字节型、字型、双字型。

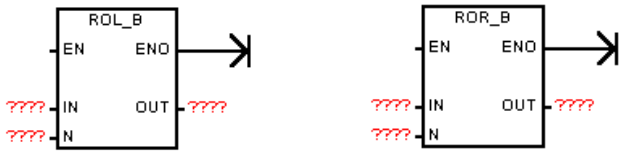
循环移位特点：

- 移位数据存储单元的移出端与另一端相连，同时又与 SM1.1（溢出）相连，所以最后被移出的位被移到另一端的同时，也被放到 SM1.1 位存储单元。例如在循环右移时，移位数据的最右端位移入最左端，同时又进入 SM1.1。SM1.1 始终存放最后一次被移出的位。
- 移位次数与移位数据的长度有关，如果移位次数设定值大于移位数据的位数，则执行循环移位之前，系统先对设定值取以数据长度为底的模，用小于数据长度的结果作为实际循环移位的次数。如字左移时，若移位次数设定为 36，则先对 36 取以 16 为底的模，得到小于 16 的结果 4，故指令实际循环移位 4 次。
- 移位次数 N 为字节型数据。

如果移位操作使数据变为 0，则零存储器位（SM1.0）自动置位。

移位指令影响的特殊存储器位：SM1.0（零）、SM1.1（溢出）。

使能流输出 ENO 断开的出错条件：SM4.3（运行时间）、0006（间接寻址）。



（1）字节循环左移和字节循环右移。

RLB 和 RRB，字节循环左移和字节循环右移。使能输入有效时，把字节型输入数据 IN 循环左移或循环右移 N 位后，再将结果输出到 OUT 所指的字节存储单元。实际移位次数为设定值取以 8 为底的模所得的结果。

指令格式：　RLB　　OUT,　N （字节循环左移）

　　RRB　　OUT,　N （字节循环右移）

例：　　RLB　　MB0,　2

　　RRB　　LB0,　3

（2）字循环左移和字循环右移。

RLW 和 RRW，字循环左移和字循环右移。指令盒与字节循环移位只有名称变为 ROL_W 和 ROR_W，其他部分完全相同。使能输入有效时，把字型输入数据 IN 循环左移或循环右移 N 位后，再将结果输出到 OUT 所指的字存储单元。实际移位次数为设定值取以 16 为底的模所得的结果。

指令格式：　RLW　　OUT,　N （字循环左移）

　　RRW　　OUT,　N （字循环右移）

例：　　RLW　　MW0,　2

　　RRW　　LW0,　3

（3）双字循环左移和双字循环右移。

RLD 和 RRD，双字循环左移和双字循环右移。指令盒与字节循环移位只有名称变为 ROL_DW 和 ROR_DW，其他部分完全相同。使能输入有效时，把双字型输入数据 IN 循环左移或循环右移 N 位后，再将结果输出到 OUT 所指的双字存储单元。实际移位次数为设定值取以 32 为底的模所得的结果。

指令格式：　RLD　　OUT,　N （双字循环左移）

　　RRD　　OUT,　N （双字循环右移）

例：　　RLD　　MD0,　2

　　RRD　　LD0,　3

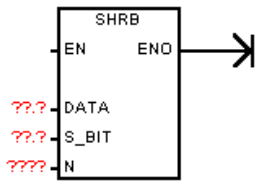
以指令 RRW　LW0,　3 为例，指令执行情况如表 4.18　所示。

表 4.18　指令 RRW 执行结果

移位次数	地址	单元内容	位 SM1.1	说明
0	LW0	1011010100110011	x	移位前
1	LW0	1101101010011001	1	右端 1 移入 SM1.1 和 LW0 左端
2	LW0	1110110101001100	1	右端 1 移入 SM1.1 和 LW0 左端
3	LW0	0111011010100110	0	右端 0 移入 SM1.1 和 LW0 左端

3．寄存器移位

SHRB，寄存器移位指令。



该指令在梯形图中有 3 个数据输入端：DATA 为数值输入， 将该位的值移入移位寄存器；S_BIT 为移位寄存器的最低位端；N 指定移位寄存器的长度。每次使能输入有效时，整个移位寄存器移动 1 位。

移位特点：

移位寄存器长度在指令中指定，没有字节型、字型、双字型之分。可指定的最大长度为 64 位，可正也可负。

移位数据存储单元的移出端与 SM1.1（溢出）相连，所以最后被移出的位被放到 SM1.1 位存储单元。

移位时，移出位进入 SM1.1，另一端自动补以 DATA 移入位的值。

移位方向分为正向移位和反向移位。正向移位时长度 N 为正值，移位是从最低字节的最低位 S_BIT 移入，从最高字节的最高位移出；反向移位时，长度为 N 为负值，移位是从最高字节的最高位移入，从最低字节的最低位 S_BIT 移出。

最高位的计算方法：(N 的绝对值−1+(S_BIT 的位号))/8，相除结果中，余数即是最高位的位号，商与 S_BIT 的字节号之和即是最高位的字节号。

移位指令影响的特殊存储器位：SM1.1（溢出）。

使能流输出 ENO 断开的出错条件：SM4.3（运行时间）、0006（间接寻址）、0091（操作数超界）、0092（计数区错误）。

指令格式： SHRB DATA, S_BIT, N

例： SHRB I0.5, V20.0, 5

以本条指令为例，指令执行情况如表 4.19 所示。

表 4.19 指令 SHRB 执行结果

脉冲数	I0.5 值	VB20 内容	位 SM1.1	说明
0	1	101 10101	x	移位前。移位时，从 VB20.4 移出
1	1	101 01011	1	1 移入 SM1.1，I0.5 的脉冲前值进入右端
2	0	101 10111	0	0 移入 SM1.1，I0.5 的脉冲前值进入右端
3	0	101 01110	1	1 移入 SM1.1，I0.5 的脉冲前值进入右端

4.3.3 字节交换指令

SWAP，字节交换指令。使能输入有效时，将字型输入数据 IN 的高字节和低字节进行交换。

本指令只对字型数据进行处理，指令的执行不影响特殊存储器位。

使能流输出 ENO 断开的出错条件：SM4.3（运行时间）、0006（间接寻址）。

指令格式： SWAP IN （字节交换）

例： SWAP VW10

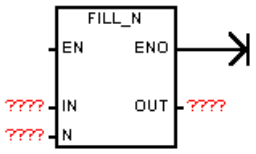
以第本指令为例，指令执行情况如表 4.20 所示。

表 4.20　指令 SWAP 执行结果

时间	单元地址	单元内容	说明
执行前	VW10	10110101 00000001	交换指令执行前
执行后	VW10	00000001 10110101	执行交换指令，将高、低字节的内容交换

4.3.4　填充指令

FILL，存储器填充指令。使能输入有效时，用字型输入数据 IN 填充从输出 OUT 所指的单元开始的 N 个字存储单元。



填充指令只对字型数据进行处理，N 值为字节型，可取从 1～255 的整数。指令的执行不影响特殊存储器位。

使能流输出 ENO 断开的出错条件：SM4.3（运行时间）、0006（间接寻址）、0091（操作数超界）。

指令格式：　FILL　　IN，　OUT，　N　（填充指令）

例：　　FILL　　10，　VW100，　12

本条指令的执行结果是：将数据 10 填充到从 VW100 到 VW122 共 12 个字存储单元中。

4.4　表功能指令

系统的表格存储格式：一个表由表地址（表的首地址）指明。 表地址和第二个字地址所对应的单元分别存放两个表参数（最大填表数 TL 和实际填表数 EC ）。之后是最多 100 个存表数据。

表只对字型数据存储，表的格式例如表 4.21 所示。

表 4.21　数据表格式

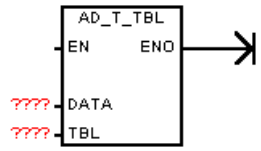
单元地址	单元内容	说明
VW100	0006	TL=6，最多可填 6 个数，VW100 为表地址
VW102	0004	EC=4，实际在表中存有 4 个数据
VW104	1203	数据 0
VW106	4467	数据 1
VW108	9086	数据 2
VW110	3592	数据 3
VW112	****	无效数据
VW114	****	无效数据

4.4.1　表存数指令

ATT，表存数指令。

该指令在梯形图中有 2 个数据输入端：DATA 为数值输入，指出将被存储的字型数据或其地址；TBL 表格的首地址，用以指明被访问的表格。当使能输入有效时，将输入字型数据

添加到指定的表格中。



表存数特点：

表存数时，新存的数据添加在表中最后一个数据的后面。每向表中存一个数据，实际填表数 EC 会自动加 1。

表存数指令影响的特殊存储器位：SM1.4（溢出）。

使能流输出 ENO 断开的出错条件：SM4.3（运行时间）、0006（间接寻址）、0091（操作数超界）。

指令格式： ATT DATA, TBL

例： ATT VW200, VW100

如果仍是对表 4.21 存取，指令执行前 VW200 中的内容为 2222，则指令执行情况如表 4.22 所示。

表 4.22 指令 ATT 执行结果

操作数	单元地址	执行前内容	执行后内容	说明
DATA	VW200	2222	2222	被填表数据及地址
TBL	VW100	0006	0006	TL=6，最大填表数为 6，不变化
	VW102	0004	0005	EC 实际存表数由 4 加 1 变为 5
	VW104	1203	1203	数据 0
	VW106	4467	4467	数据 1
	VW108	9086	9086	数据 2
	VW110	3592	3592	数据 3
	VW112	****	2222	将 VW200 中的数据填入表中
	VW114	****	****	无效数据

4.4.2 表取数指令

从表中移出一个字型数据可有两种方式：先进先出式和后进先出式。一个数据从表中取出之后，表的实际表数 EC 值减 1。两种方式指令在梯形图中都有两个数据端：输入端 TBL 表格的首地址，用以指明被访问的表格；输出端 DATA 指明数值取出后要存放的目标单元。

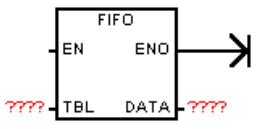
如果指令试图从空表中取走一个数值，则特殊标志寄存器位 SM1.5 置位。

表取数指令影响的特殊存储器位：SM1.5（表空）。

使能流输出 ENO 断开的出错条件：SM4.3（运行时间）、0006（间接寻址）、0091（操作数超界）。

1．FIFO，先进先出指令

当使能输入有效时，从 TBL 指明的表中移出第一个字型数据并将其输出到 DATA 所指定的字单元。



FIFO 表取数特点：

取数时，移出的数据总是最先进入表中的数据。每次从表中移出一个数据，剩余数据依次上移一个字单元位置，同时实际填表数

EC 会自动减 1。

指令格式：　FIFO　　TBL,　DATA

例：　　FIFO　　VW100,　AC0

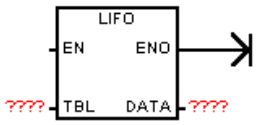
如果仍是对表 4.21 存取，则指令执行情况如表 4.23 所示。

表 4.23　指令 FIFO 执行结果

操作数	单元地址	执行前内容	执行后内容	说明
DATA	AC0	空	1203	从表中取走的数据及输出
TBL	VW100	0006	0006	TL=6，最大填表数为 6，不变化
	VW102	0004	0003	EC 实际存表数由 4 减 1 变为 3
	VW104	1203	4467	数据 0，剩余数据依次上移一格
	VW106	4467	9086	数据 1
	VW108	9086	3592	数据 2
	VW110	3592	****	无效数据
	VW112	****	****	无效数据
	VW114	****	****	无效数据

2．LIFO，后进先出指令

当使能输入有效时，从 TBL 指明的表中移出最后一个字型数据并将其输出到 DATA 所指定的字单元。



LIFO 表取数特点：

取数时，移出的数据是最后进入表中的数据。每次从表中取出一个数据，剩余数据位置保持不变，实际填表数 EC 会自动减 1。

指令格式：　LIFO　　TBL,　DATA

例：　　LIFO　　VW100,　AC0

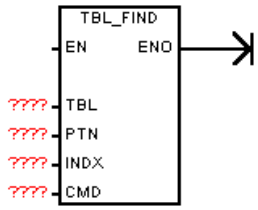
如果仍是对表 4.21 存取，则指令执行情况如表 4.24 所示。

表 4.24　指令 LIFO 执行结果

操作数	单元地址	执行前内容	执行后内容	说明
DATA	AC0	空	1203	从表中取走的数据输出到 AC0
TBL	VW100	0006	0006	TL=6，最大填表数为 6，不变化
	VW102	0004	0003	EC 实际存表数由 4 减 1 变为 3
	VW104	1203	1203	数据 0，剩余数据不移动
	VW106	4467	4467	数据 1
	VW108	9086	9086	数据 2
	VW110	3592	****	无效数据
	VW112	****	****	无效数据
	VW114	****	****	无效数据

4.4.3 表查指令

FND?，表查指令。通过表查指令可以从字型数表中找出符合条件的数据所在的表中数据编号，编号范围为 0～99。



在梯形图中有 4 个数据输入端：TBL 表格的首地址，用以指明被访问的表格；PTN 是用来描述查表条件时进行比较的数据；CMD 是比较运算符“？”的编码，它是一个 1～4 的数值，分别代表=、<>、<和>运算符；INDX 用来指定表中符合查找条件的数据的地址。

由 PTN 和 CMD 就可以决定对表的查找条件。例如，PTN 为 16#2555，CMD 为 3，则查找条件为“<2555（16 进制）”。

表查指令执行之前，应先对 INDX 的内容清 0。当使能输入有效时，从 INDX 开始搜索表 TBL，寻找符合由 PTN 和 CMD 所决定的条件的数据，如果没有发现符合条件的数据，则 INDX 的值等于 EC。如果找到一个符合条件的数据，则将该数据的表中地址装入 INDX 中。

表查指令执行完成，找到一个符合条件的数据，如果想继续向下查找，必须先对 INDX 加 1，以重新激活表查找指令。

查表指令不影响特殊存储器位。使能流输出 ENO 断开的出错条件：SM4.3（运行时间）；0006（间接寻址）；0091（操作数超界）。

在语句表中运算符直接表示，而不用各自的编码。

指令格式：　FND=　TBL，PTN，INDX　（查找条件：=PTN）

　　FND<>　TBL，PTN，INDX　（查找条件：<>PTN）

　　FND<　TBL，PTN，INDX　（查找条件：<PTN）

　　FND>　TBL，PTN，INDX　（查找条件：>PTN）

例：　　FND>　VW100，VW300，AC0

如果仍是对表 4.21 进行操作，指令的执行结果如表 4.25 所示。

表 4.25　表查指令执行结果

操作数	单元地址	执行前内容	执行后内容	说明
PTN	VW300	2000	5000	用来比较的数据
INDX	AC0	0	2	符合查表条件的单元地址
CMD	无	4	4	4 表示为>
TBL	VW100	0006	0006	TL=6，最大填表数，不需要
	VW102	0004	0004	EC 实际存表数
	VW104	1203	1203	数据 0
	VW106	4467	4467	数据 1
TBL	VW108	9086	9086	数据 2
	VW110	3592	3592	数据 3
	VW112	****	****	无效数据
	VW114	****	****	无效数据

4.5 转换指令

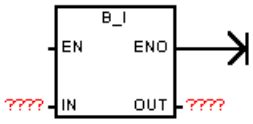
转换指令是指对操作数的类型进行转换，包括数据的类型转换、码的类型转换以及数据和码之间的类型转换。

4.5.1 数据类型转换

数据类型主要包括字节、整数、双整数、实数，现在的可编程序控制器对 BCD 码十进制数据和 ASCII 字符型数据的处理能力也大大增强。不同性质的指令对操作数的类型要求不同，类型转换指令可将固定的一个数值用到不同类型要求的指令，而不必对数据进行针对类型的重新装载。

1．字节与整数

（1）字节到整数。



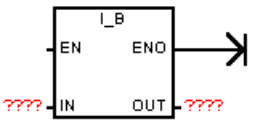
BTI，字节转换为整数指令。使能输入有效时，将字节输入数据 IN 转换成整数类型，并将结果送到 OUT 输出。字节型是无符号的，所以没有符号扩展。

使能流输出 ENO 断开的出错条件：SM4.3（运行时间）、0006（间接寻址）。

指令格式：　BTI　　IN，OUT

例：　　BTI　　VB0，AC0

（2）整数到字节。



ITB，整数转换为字节指令。使能输入有效时，将整数输入数据 IN 转换成字节类型，并将结果送到 OUT 输出。输入数据超出字节范围（0～255）则产生溢出。

转换指令影响的特殊存储器位：SM1.1（溢出）。

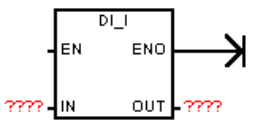
使能流输出 ENO 断开的出错条件：SM1.1（溢出）、SM4.3（运行时间）、0006（间接寻址）。

指令格式：　ITB　　IN，OUT

例：　　ITB　　AC0，VB10

2．整数与双整数

（1）双整数到整数。



DTI，双整数转换为整数指令。使能输入有效时，将双整数输入数据 IN 转换成整数类型，并将结果送到 OUT 输出。输入数据超出整数范围则产生溢出。

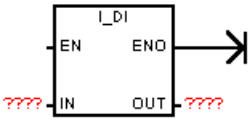
转换指令影响的特殊存储器位：SM1.1（溢出）。

使能流输出 ENO 断开的出错条件：SM1.1（溢出）、SM4.3（运行时间）、0006（间接寻址）。

指令格式：　DTI　　IN，OUT

例：　　DTI　　AC0，VW20

（2）整数到双整数。



ITD，整数转换为双整数指令。使能输入有效时，将整数输入数据 IN 转换成双整数类型（符号进行扩展），并将结果送到 OUT 输出。

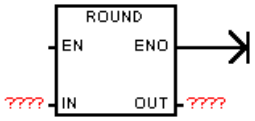
使能流输出 ENO 断开的出错条件：SM4.3（运行时间）、0006（间接寻址）。

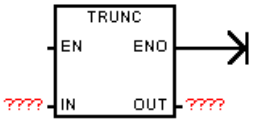
指令格式： ITD IN, OUT

例： ITD VW0, AC0

3．双整数与实数

（1）实数到双整数。





ROUND 和 TRUNC，实数转换为双整数指令。使能输入有效时，将实型输入数据 IN 转换成双整数类型，并将结果送到 OUT 输出。两条指令的区别是：前者小数部分 4 舍 5 入，而后者小数部分直接舍去。

转换指令影响的特殊存储器位：SM1.1（溢出）。

使能流输出 ENO 断开的出错条件：SM1.1（溢出）、SM4.3（运行时间）、0006（间接寻址）。

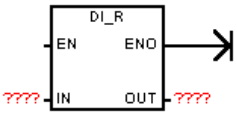
指令格式： ROUND IN, OUT

TRUNC IN, OUT

例： ROUND VD0, AC0

（2）双整数到实数。

DTR，双整数转换为实数指令。使能输入有效时，将双整数输入数据 IN 转换成实型，并将结果送到 OUT 输出。



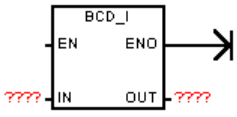
使能流输出 ENO 断开的出错条件：SM4.3（运行时间）、0006（间接寻址）。

指令格式： DTR IN, OUT

例： DTR AC0, VD100

4．整数与 BCD 码

（1）BCD 码到整数。



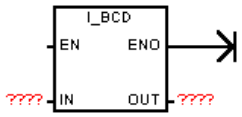
BCDI，BCD 码转换为整数指令。使能输入有效时，将 BCD 码输入数据 IN 转换成整数类型，并将结果送到 OUT 输出。输入数据 IN 的范围为 0～9999。

指令格式： BCDI OUT

例： BCDI AC0

（2）整数到 BCD 码

IBCD，整数转换为 BCD 码指令。使能输入有效时，将整数输入数据 IN 转换成 BCD 码



类型，并将结果送到 OUT 输出。输入数据 IN 的范围为 0～9999。

指令格式：　IBCD　　OUT

例：　　IBCD　　AC0

5．程序实例

功能：模拟量控制程序中的数据类型转换。将模拟量输入端采样值由整数转换为双整数，然后由双整数转换为实数，再除以一个比例因子得到 PLC 可以处理范围内的值。

本程序如图 4.27 所示。

4.5.2　编码和译码

1．编码

ENCO，编码指令。使能输入有效时，将字型输入数据 IN 的最低有效位（值为 1 的位）的位号输出到 OUT 所指定的字节单元的低 4 位。即用半个字节来对一个字型数据 16 位中的 1 位有效位进行编码。

使能流输出 ENO 断开的出错条件：SM4.3（运行时间）、0006（间接寻址）。

指令格式：　ENCO　　IN,　OUT

例：　　ENCO　　AC0,　VB0

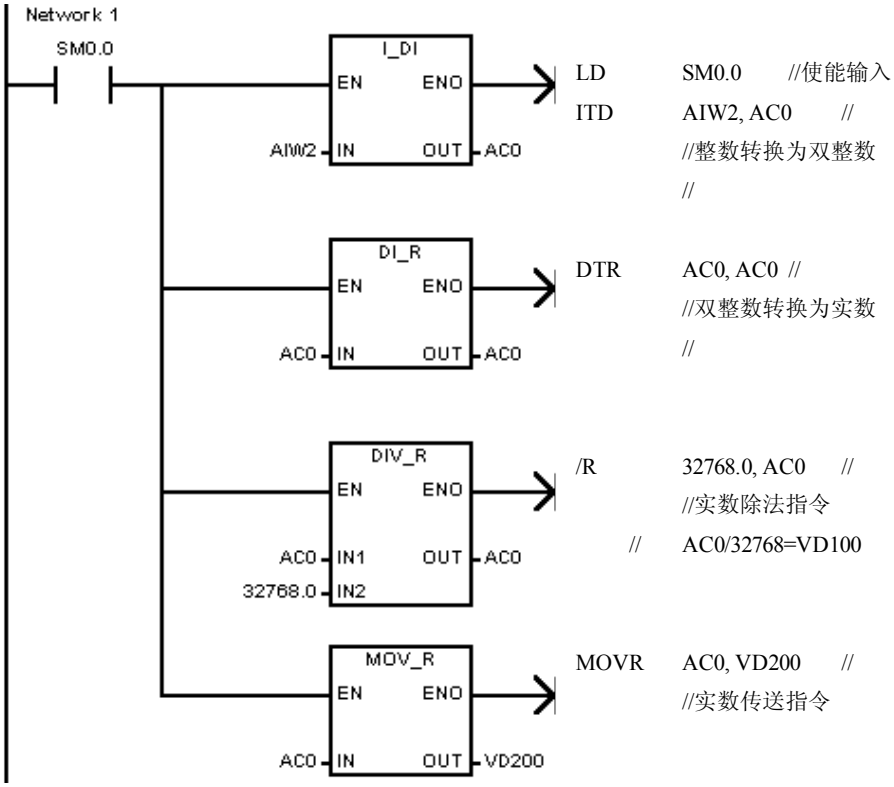


图 4.27　数据类型转换程序例

以本指令为例，指令执行情况如表 4.26 所示。

表 4.26　编码指令执行结果

时间	单元地址	单元内容	说明
执行前	AC0	00000000 01000000	要编码的为 AC0 中的第 6 位（始于 0 位）
	VB0	xxxxxxxx	任意值
执行后	AC0	00000000 01000000	数据未变
	VB0	00000110	将位号 6 写入 VB0 的低 4 位

2．译码

DECO，译码指令。使能输入有效时，将字节型输入数据 IN 的低 4 位所表示的位号对 OUT 所指定的字单元的对应位置 1，其他位置 0。即对半个字节的编码进行译码来选择一个字型数据 16 位中的 1 位。

使能流输出 ENO 断开的出错条件：SM4.3（运行时间）、0006（间接寻址）。

指令格式：　DECO　IN，OUT

例：　　DECO　VB0，AC0

本指令执行情况如表 4.27 所示。

表 4.27　译码指令执行结果

执行情况	单元地址	单元内容	说明
执行前	VB0	00001000	要编码的位的位号为 8，存于 VB0 的低 4 位
	AC0	xxxxxxxxxxxxxxxx	任意值
执行后	VB0	00001000	数据未变
	AC0	00000001 00000000	将位号 8 对应的第 8 位置 1，其他位为 0

4.5.3　七段码

SEG，七段码指令。使能输入有效时，将字节型输入数据 IN 的低 4 位有效数字产生相应的七段码，并将其输出到 OUT 所指定的字节单元。

该指令在数码显示时直接应用，非常方便。

使能流输出 ENO 断开的出错条件：SM4.3（运行时间）、0006（间接寻址）

指令格式：　SEG　IN，OUT

例：　　SEG　VB10，AC0

4.5.4　字符串转换

字符串转换是将标准字符编码 ASCII 码字符串与 16 进制值、整数、双整数及实数之间进行的转换。

可进行转换的 ASCII 码为 0～9 及 A～F 的编码。

1．指令种类

（1）ASCII 码转换为十六进制指令。

（2）十六进制转换为 ASCII 码指令。

（3）整数转换为 ASCII 码指令。

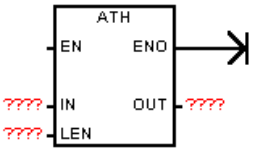
（4）双整数转换为 ASCII 码指令。

（5）实数转换为 ASCII 码指令。

2．指令介绍

下面仅以 ASCII 码转换为十六进制指令为例说明字符串与其他数据类型之间的转换。

ATH，ASCII 码转换为十六进制指令。指令盒中有 3 个操作数：IN，开始字符的字节地址，字节类型；LEN，字符串的长度，字节类型，最大长度为 255；OUT，输出目的开始字节地址，字节类型。使能输入有效时，把从 IN 开始的长度为 LEN 的 ASCII 码转换为十六进制数，并将结果送到 OUT 开始的字节进行输出。



指令影响的特殊标志位：SM1.7 非法（ASCII 码）。

使能流输出 ENO 断开的出错条件：SM4.3（运行时间）、0006（间接寻址）、0091（操作数超界）。

指令格式：　ATH　　IN,　　OUT, LEN

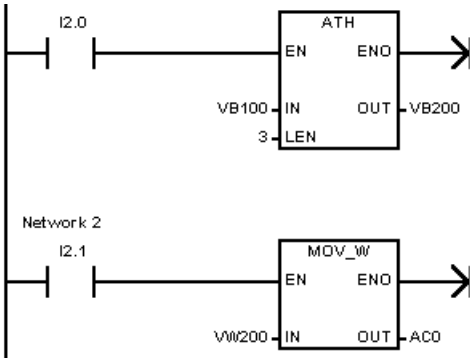
例：　　ATH　　VB100,　VB200,　　3

3．程序实例

以上面的指令为例，该条指令的执行结果如表 4.28 所示，程序如图 4.28 所示。

表 4.28　指令 ATH 执行结果

位置	首地址	含义	字节 1	字节 2	字节 3	说明
ASCII 码区	VB100	二进制	0011 0010	0011 0100	0100 0101	原信息的存储形式及对应的 ASCII 编码
		含义	2	4	E	
16 进制区	VB200	二进制	0010 0100	1110 xxxx	xxxx xxxx	转化结果信息编码及含义
		含义	2 4	E X	X X	




```
LD      I2.0        //使能输入
ATH     VB100, VB200, 3      //
        //从 VB100 开始的
        //3 个字符转换为
        //16 进制数值
LD      I2.1        //
MOVW    VW200, AC0           //
        //字传送指令
        //结果输出到 AC0
```

图 4.28　字符串转换

本章介绍 SIMATIC 指令集所包含的基本指令及使用方法，涉及的是位操作类指令和数

据运算及处理类指令。重点是掌握基本指令中常用指令的应用，并熟练掌握使用梯形图编程的方法。

（1）可编程序控制器的编程以指令系统为基础，指令又是以机器硬件为依据，编程时必须考虑到存储器中各编程元件的地址分配及操作数范围。

常用的构成梯形图的元素有触点、线圈、功能指令及相应的操作数等。

（2）通过位操作类指令的学习，基本上就可以用 PLC 实现原来继电接触控制系统所能实现的控制任务。这个类通常与位操作和位运算有关，位运算在机器内部通过逻辑堆栈完成。位操作类指令包括基本逻辑指令、复杂逻辑指令、定时器指令、计数器指令和比较指令等。

（3）各种运算指令包括算术运算指令和逻辑运算指令，使当前的 PLC 对数据处理的能力大大增强，开阔了 PLC 的应用领域。

算术运算是对有符号和大小含义的算术数进行处理，算术运算指令包括加法、减法、乘法、增减指令和一些常用的数学函数指令，如平方根、自然对数、三角函数等。逻辑运算是对无符号和大小含义的逻辑数进行处理，逻辑运算类指令包括逻辑与、逻辑或、逻辑异、逻辑取反等指令。

学会使用这些指令的同时，还应学会结合数学方法灵活运用这些指令，以完成较为复杂的运算任务。

（4）其他数据处理指令主要涉及非数值运算类的数据操作，包括传送类指令、移位指令、循环移位指令、字节交换指令、填充指令和数据类型转换指令等。

（5）表功能指令可以用来方便地建立和存取字类型的数据表，一个数据表由表的首地址来标识，首地址和第二个地址单元分别存放最大填表数和实际填表数，其后是最多 100 个存表字型数据。表功能指令包括表存数指令、表取数指令和表查找指令等。

一、选择题

1．请从下列语句表指令中选择错误的一个（　　）。

A．BMB VB20,MB20　　B．BCDI　AC0

C．DTR　AC0,VD100　　D．LPS

2．请从下列语句表指令中选择错误的一个（　　）。

A．IBCD AC0　　B．ROUND VW20,AC0

C．FND> VW100,VW300,AC0　　D．LDI I0.0

3．请从下列语句表指令中选择错误的一个（　　）。

A．SWAP VD20　　B．RRW AC0,2

C．RLB VB10,2　　D．INVW　VW10

4．请从下列语句表指令中选择错误的一个（　　）。

A．SQRT VD0,AC1　　B．EXP VD200,AC0

C．/I　VD20, AC0　　D．AB<> MB10,MB20

5．请从下列语句表指令中选择错误的一个（　　）。

A．TON T36,100　　B．TOF T0

C．LDS 5　　D．LRD

6．请从下列语句表指令中选择错误的一个（　　）。

A．LPS　　B．RLW 10,VB10

C．BIW VB10,QB0　　D．INVD VD100

7．请从下列语句表指令中选择错误的一个（　　）。

A．MOV 12,VB20　　B．LDNI SM5.4

C．SLW VW200,3　　D．SQRT VD10,AC0

8．请从下列语句表指令中选择错误的一个（　　）。

A．LN 16.0,VD100　　B．BIR IB0, VB10

C．BTI　VW0,AC0　　D．DTI AC0,VW20

9．请从下列语句表指令中选择错误的一个（　　）。

A．ROUND VD0,AC1　　B．ITD VW0,AC0

C．ITB AC0,VB20　　D．TRUNC VD30,80

10．请从下列语句表选项中选择错误的一个（　　）。

A．OI　I3.2　　B．MOV VW1,SMW0

C．RI　Q0.5,3　　D．CALL　SBR_0

11．下列（　　）属于双字寻址。

A．QW1　　B．V10　　C．IB0　　D．MD28

12．只能使用字寻址方式来存取信息的寄存器是（　　）。

A．S　　B．I　　C．H　　C．　　D．AI

13．SM 是（　　）存储器的标识符。

A．高速计数器　　B．累加器

C．内部辅助寄存器　　D．特殊辅助寄存器

14．字传送指令的操作数 IN 和 OUT 可寻址的寄存器不包括下列（　　）。

A．T　　B．M　　C．AQ　　D．AC

15．字节传送指令的操作数 IN 和 OUT 可寻址的寄存器不包括下列（　　）。

A．V　　B．I　　C．Q　　D．AI

16．若整数的加减法指令的执行结果发生溢出则影响（　　）位。

A．SM1.0　　B．SM1.1　　C．SM1.2　　D．SM1.3

17．字取反指令梯形图的操作码为（　　）。

A．INV-　　B．INV-W　　C．INV-　　D．INV-X

18．双字整数的加减法指令的操作数都采用（　　）寻址方式。

A．字　　B．双字　　C．字节　　D．位

19．若整数的乘/除法指令的执行结果是零则影响（　　）位。

A．SM1.0　　B．SM1.1　　C．SM1.2　　D．SM1.3

20．实数开方指令的梯形图操作码是（　　）。

A．EXP　　B．LN　　C．SQRT　　D．TIN

21．设 VW10 中存有数据 123.9，执行 TRUNC 指令后的的结果是（　）。

A．123.5　　B．124　　C．120　　D．123

22．取整指令的梯形图指令的操作码是（　）。

A．TRUNC　　B．ROUND　　C．EXP　　D．LN

23．指令 MOVB AC0,VB12 中累加器用的寻址方式是（　）。

A．位寻址　　B．字节寻址　　C．字寻址　　D．双字寻址

24．整数的加减法指令的操作数都采用（　）寻址方式。

A．字　　B．双字　　C．字节　　D．位

25．把一个 BCD 码转换为一个整数值的梯形图指令的操作码是（　）。

A．B-I　　B．I-BC　　C．BCD-I　　D．I-R

26．段译码指令的梯形图指令的操作码是（　）。

A．DECO　　B．ENCO　　C．SEG　　D．TRUNC

27．设 AC1 中的低 16 位存有十六进制数 16#8200，现执行 ENCO AC1, VB100 指令，则指令的执行结果 VB40 中的内容是（　）。

A．0009H　　B．09H　　C．08H　　D．04H

28．填表指令的功能是向表中增加一个数值，表中第一个数是（　）数。

A．要填进表中的数　　B．最大填表数

C．实际填表数　　D．表中已有的数值

29．在查表指令中，若被查数据与参考数据之间的关系是不等于，则查表指令的语句表的操作码是（　）。

A．FIFO　　B．FILO　　C．FIND=　　D．FIND<>

30．设 VW10 中的数据是 6543H，VW20 中的数据是 0897H，则执行 ANDW VW10,VW20 指令后，VW20 的内容是（　）。

A．4DD7H　　B．5489H　　C．0003H　　D．9ABCH

二、填空题

1．定时器包括_____、_____和_____3 种类型。

2．把一个实数转换为一个双字整数值的 ROUND 指令，它的小数部分采用_____原则处理。

3．段译码指令的操作码是_____，它的源操作数的寻址方式是_____寻址，目的操作数的寻址方式是_____寻址。

4．填表指令可以往表格里最多填充_____个数据。

5．字移位指令的最大移位位数为_____。

6．正跳变指令的梯形图格式为_____。

7．只有_____和_____可以使用立即指令。

8．计数器指令包括_____、_____和_____3 种。

9．定时器的状态位在当前值_____预置值时为 ON。

10．被置位的点一旦置位后，在执行_____指令前不会变为 OFF，具有锁存功能。

11．定时器预设值 PT 采用的寻址方式为_____。

12．定时器的两个变量是_____和_____。

13．如果加计数器 CTU 的复位输入电路（R）_____，计数输入电路（CU）由断开变为接通，计数器的当前值加 1。当前值达到设定值（PV）时，其常开触点_____，常闭触点_____。复位输入电路（R）_____时计数器被复位，复位后其常闭触点_____，当前值为_____。

三、语句表和梯形图的互换

1．将下列语句表转换为梯形图。

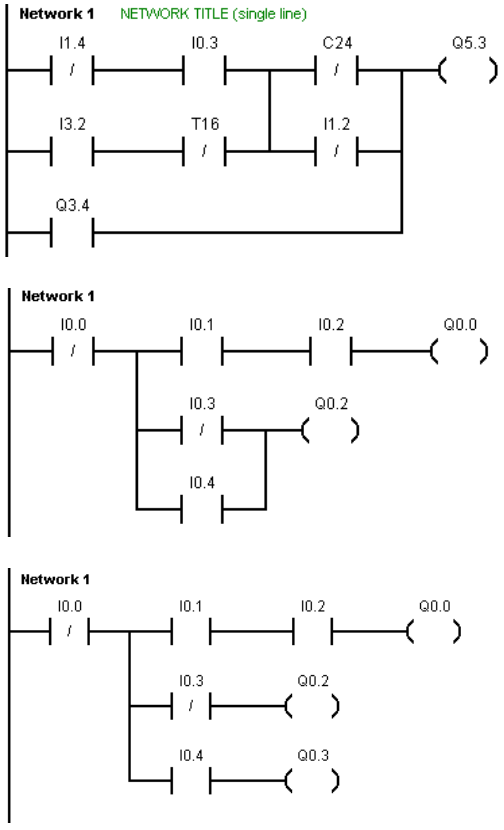
```
LD    I0.1
A     I0.0
LD    I0.2
ON    I0.4
O     I0.3
ALD
=     Q0.0
```

```
LD    I0.1
A     I0.0
LD    I0.2
A     I0.4
OLD
AN    I0.3
=I    Q0.1
ALD
=     Q0.0
```

```
LD    I0.0
A     I0.1
LD    I0.2
ONI   I0.4
AN    I0.3
OLD
=     Q0.0
```

```
LD   I0.0
AN   T37
TON  T37,1000
LD   T37
LD   Q0.0
CTU  C10,360
LD   C10
O    Q0.0
=    Q0.0
```

2．将下列梯形图转换为语句表。



四、设计题

1．编写一段程序计算以下函数的值。

（1）[(100＋200)×10]/3。

（2）6 的 78 次方。

（3）求 SIN65^{o} 的函数值。

2．编写一段梯形图程序，实现将 VB20 开始的 100 个字型数据移到 VB400 开始的存储区，这 100 个数据的相对位置在移动前后不发生变化。

3．编写一段输出控制程序，假设有 8 个指示灯，从左到右以 0.5s 速度依次点亮，到达最右端后，再从左到右依次点亮，如此循环显示。