北京发那科培训讲义·设计篇（2）

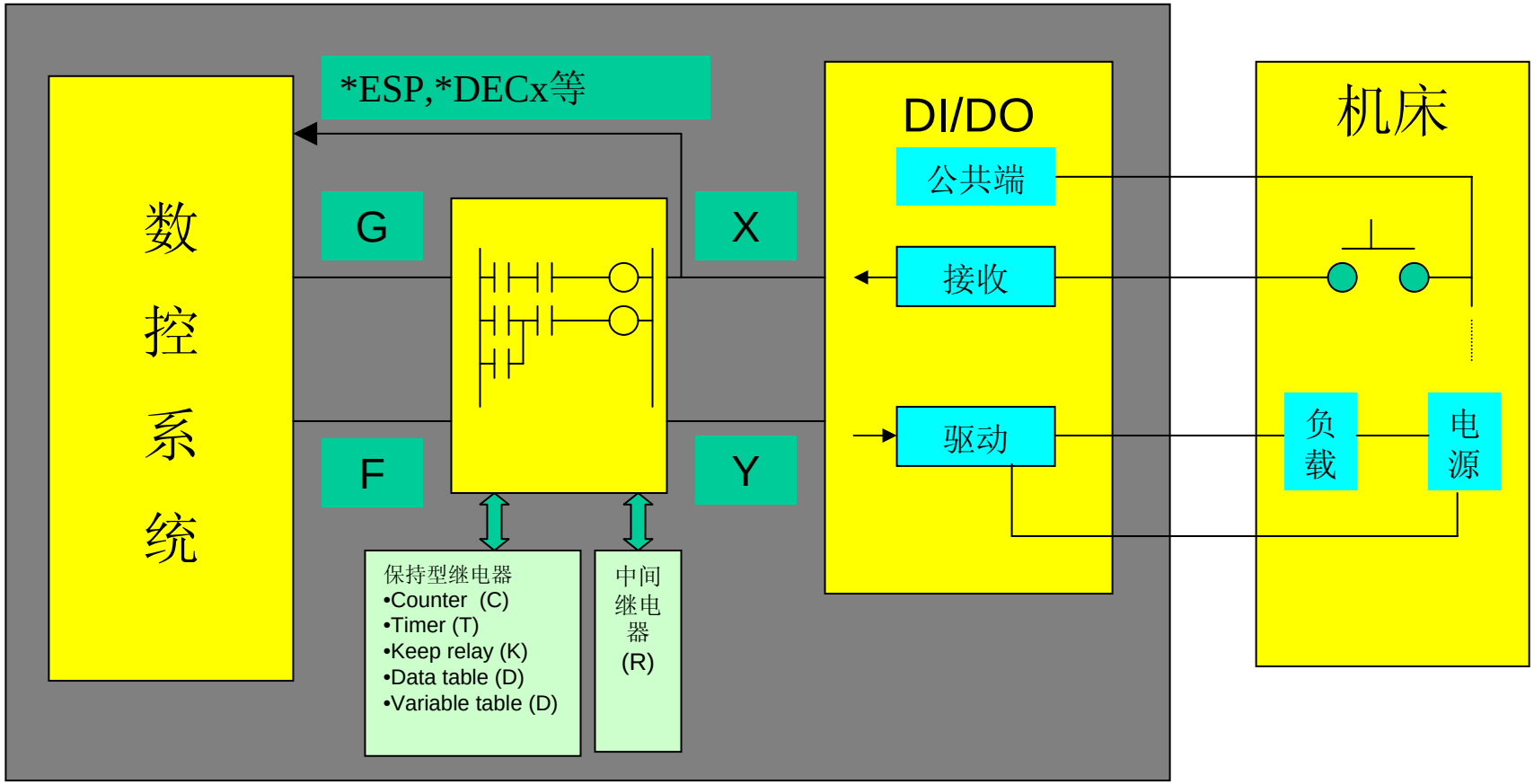
PMC编程

数控机床做为自动化控制设备，是在自动控制下进行工作的，数控机床所受控制可分为两类：

一类是最终实现对各坐标轴运动进行的“数字控制”。如：对CNC车床X轴和Z轴，CNC铣床X轴，Y轴，Z 轴的移动距离，各轴运行的插补，补偿等的控制即为“数字控制”。

另一类为“顺序控制”。对数控机床来说，“顺序控制”是在数控机床运行过程中，以CNC内部和机床各行程开关，传感器，按钮，继电器等的开关量信号状态为条件，并按照预先规定的逻辑顺序对诸如主轴的起停，换向，刀具的更换，工件的夹紧，松开，液压，冷却，润滑系统的运行等进行的控制。与“数字控制”比较，“顺序控制”的信息主要是开关量信号。

常把数控机床分为“NC侧”和“MT侧”（即机床侧）两大部分。“NC侧”包括CNC系统的硬件和软件，与CNC系统连接的外围设备如显示器，MDI面板等。“MT侧”则包括机床机械部分及其液压、气压、冷却、润滑、排屑等辅助装置、机床操作面板、继电器线路、机床强电线路等。PMC处于NC与MT之间，对NC和MT的输入、输出信号进行处理。 MT侧顺序控制的最终对象随数控机床的类型、结构、辅助装置等的不同而有很大的差别。机床结构越复杂，辅助装置越多，最终受控对象也越多。


数控系统
*ESP,*DECx等
G
F
X
Y
保持型继电器
•Counter (C)
•Timer (T)
•Keep relay (K)
•Data table (D)
•Variable table (D)
中间继电器 (R)
DI/DO
公共端
接收
驱动
机床
负载
电源

● 地址G和F的信号，由CNC控制软件决定其地址。

例如，自动运转起动信号ST的地址是G0007.2。

● 急停信号(*ESP)和跳转信号(SKIP)等，由于受PMC扫描时间的影响使处理缓慢，故而由CNC直接进行读取。
这些直接输入信号的X地址是确定的。
对于直接输入信号，请看下项。

📖 其他地址X和Y的信号地址，可根据实际情况任意定义。

📖 跳转信号SKIP使用如下。

❶ 指令G31 Xx Yy Zz Ff；(x,y,z是假想的终点位置)

❷ 在轴移动过程中输入跳转信号SKIP时，立即结束移动。

❸ 如果不输入跳转信号SKIP进给轴将一直移动到终点。

Model		Series 21*i*–B	Series 16*i*/18*i*/21*i*–B	
		PMC–SA1	PMC–SA1 (Loader control)	PMC–SB7
Programmingmethod		Ladder	Ladder	LadderC–language
Number of ladder levels		2	2	3
First–level execution period		8 ms	8 ms	8 ms
Basic instruction processing time		5.0 μ sec/step	5.0 μ sec/step	0.033 μ sec/step
Program capacity				
• Ladder (step)	(Note 1,2)	About 3,000 About 5,000	About 3,000 About 5,000 About 8,000 About 12,000	About 3,000 About 5,000 About 8,000 About 12,000 About 16,000 About 24,000 About 32,000 About 40,000 About 48,000 About 64,000
• Symbol & comment	(Note 2)	1 to 128KB	1 to 128KB	1KB or more
• Message	(Note 2)	0.1 to 64KB	0.1 to 64KB	8KB or more
Instruction	(Basic instruction)	12	12	14
	(Functionalinstruction)	48	48	69
Internal Relay	(R)	1100 bytes	1,100 byte	8,500 byte
Extra Relay	(E)	–	–	8,000 byte
Message Request	(A)	200 points (25 byte)	200 points (25 byte)	2,000 points (500 byte, 2 bit/point)
Nonvolatile Memory				
• Data Tables	(D)	1,860 byte	1,860 byte	10,000 byte
• Variable Timers	(T)	80 bytes (20 each)	40 points (80 byte)	250 points (1,000 byte, 4 byte/point)
• Fixed Timers		20 bytes	100 points	500 points (Timer number specify)
• Counters	(C)	1860 bytes	20 points (80 byte)	100 points (400 byte, 4 byte/point)
Fixed Counters	(C)	–	–	100 points (200 byte, 2 byte/point)
• Keep Relays	(K)	–	20 byte	120 byte
Subprograms	(P)	100 each	–	2000
Labels	(L)		–	9999
I/O I/O link				
• Input		1024 points maximum	1024 points maximum	2048 points maximum (Note 3)
• Output		1024 points maximum	1024 points maximum	2048 points maximum (Note 3)
Sequence program storage media		Flash ROM 128KB	Flash ROM 128KB	Flash ROM 128KB (16,000steps option or less) 256KB (24,000steps option) 384KB(32,000/40,000steps option) 512KB (48,000steps option) 768KB (64,000steps option)

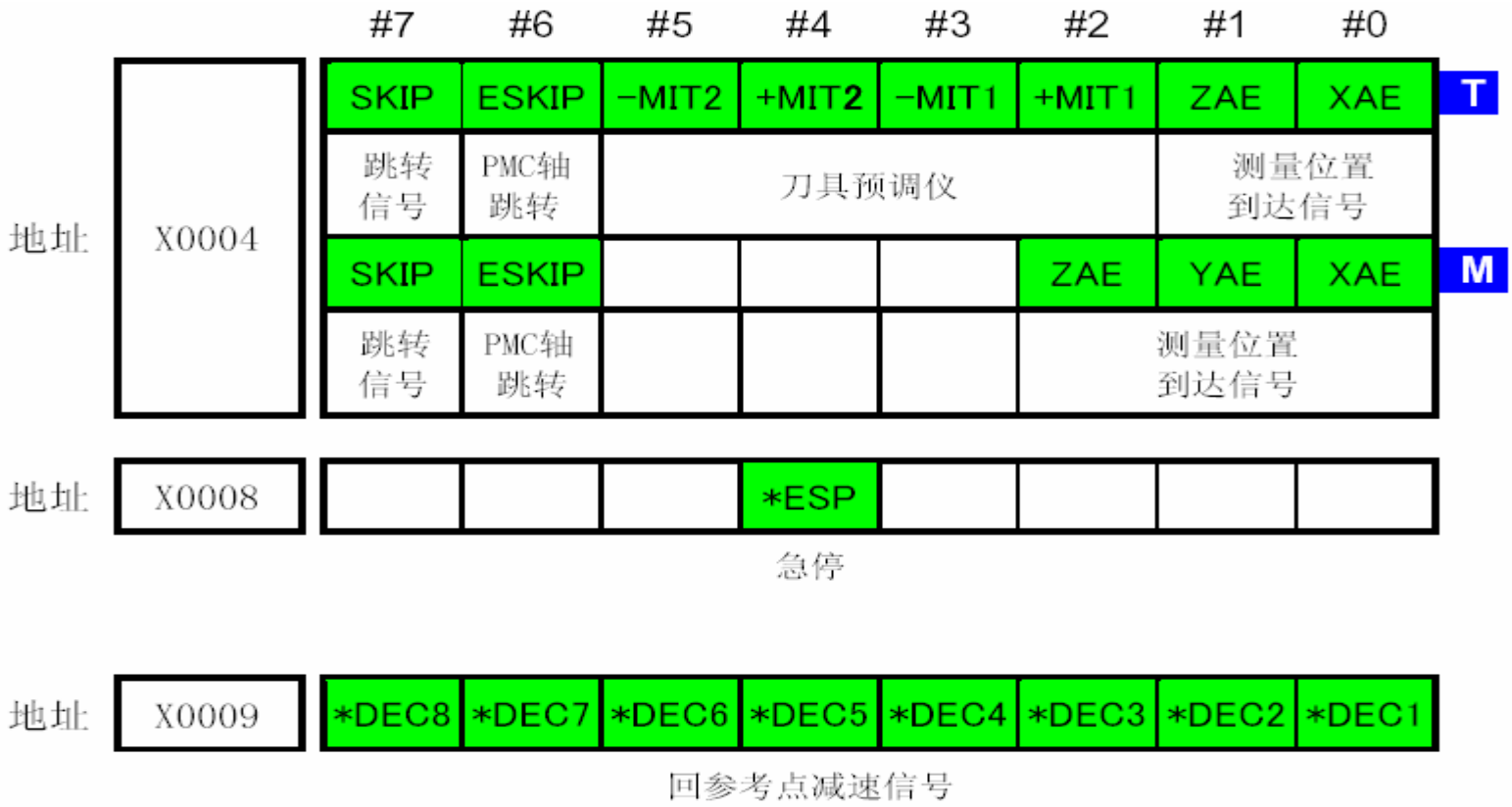
Model		Series 0*i*–A	
		PMC–SA1	PMC–SA3
Programmingmethod		Ladder	Ladder
Number of ladder levels		2	2
Level–1 cycle time		8 ms	8 ms
Basic instruction execution time		5.0 μ sec/step	0.15 μ sec/step
Program capacity			
• Ladder (step)	(Note 1)	Approx. 3,000 Approx. 5,000	Approx. 3,000 Approx. 5,000 Approx. 8,000 Approx. 12,000 Approx. 16,000
• Symbol/Comment	(Note 2,3)	1 to 128KB	1 to 128KB
• Message	(Note 2,3)	0.1 to 64KB	0.1 to 64KB
Instruction	(Basic)	12 kinds	14 kinds
	(Functional)	49 kinds	66 kinds
Internal relay	(R)	1100 bytes	1118 bytes
Message request	(A)	25 bytes (200 points)	25 bytes (200 points)
Non–volatile			
• Var.timer	(T)	80 bytes (40 each)	80 bytes (40 each)
• Counter	(C)	80 bytes (20 each)	80 bytes (20 each)
• Keep replay	(K)	20 bytes	20 bytes
• Data table	(D)	1860 bytes	1860 bytes
Subprogram	(P)	–	512 programs
Label	(L)	–	9999 labels
Fixed timer		Timer No.100 devices specified	Timer No.100 devices specified
Input/Output			
• I/O Link	(I) Max.	1024 points maximum	1024 points maximum
(master)	(O) Max.	1024 points maximum	1024 points maximum
• Built–in I/O	(I) Max.	96 points max.	96 points max.
	(O) Max.	64 points max.	64 points max.
		Flash ROM 128KB	Flash ROM 128KB

机床与PMC间的信号(X, Y)

● 机床→PMC间的信号(X)

- 从机床送到PMC的信号用X地址表示。
- 下表所列的信号是由CNC直接读取的，所以不需要PMC进行处理。

  另外，根据地址的分配决定连接线的端子号。
- 请一定要使用急停信号(*ESP)。
  SKIP等其他信号不使用时，其地址可由其它通用信号使用。
- 前头带“*”的信号是负逻辑信号。
  例如，急停信号(*ESP)通常为1，处于急停状态时*ESP为0。



- 把参数3006#0设为1时，回参考点减速信号(*DEC)变为地址G196。
  但此时需要编入顺序程序。
- 在FS16 *i/oi-mate*上没有内置I/O。

● PMC →机床的信号(Y)

从PMC送到机床的信号地址用Y表示，这些信号的地址可任意指定。

CNC与PMC间的信号

●在双系统或3系统控制时，所有系统也只公用1个PMC功能（包括顺序程序、PMC参数）。

[redacted]
急停等刀台间的公用信号输入到第1系统侧。
（在CNC内部通知第2系统）
在装料器控制板上，CNC还有另外的PMC。

●CNC和PMC间的输入输出信号如下。

系 统	PMC→CNC	CNC→PMC	备 注
第1系统	G0000～	F0000～	第2系统、第3系统的输入输出信号的地址，是在第1系统的地址上加了1000或2000后的地址。
第2系统	G1000～	F1000～	
第3系统	G2000～	F2000～	

单系统控制的自动运转起动信号ST的地址是G0007.2。
双系统控制时，第1系统的ST信号的地址与单系统控制的相同。
第2系统的地址是在第1系统的地址上加了1000后的G1007.2。

在急停等的刀架间公用信号输入到第1系统侧。
（在CNC内部通知第2系统）

字符	信号说明	型号		
		PMC-SA1	PMC-SA3	PMC-SB7
X	输入信号(MT→PMC)	X0 ~ X127 X1000 ~ X1011		X0 ~ X127 X200~X327 X1000 ~ X1127
Y	输出信号(MT←PMC)	Y0 ~ Y127 Y1000~Y1008		Y0~Y127 Y200~Y237 Y1000~Y1127
F	输入信号(NC→PMC)	F0 ~ F225 F1000 ~ F1255		F0 ~ F767 F1000 ~ F1767 F2000 ~ F2767 F3000 ~ F3767
G	输出信号(NC←PMC)	G0 ~ G255 G1000 ~ F1255		G0 ~G767 G1000 ~ G1767 G2000 ~ G2767 G3000 ~ G3767
R	内部继电器	R0 ~ R1999 R9000 ~ R9099	R0 ~ R1499 R000 ~ R9117	R0 ~ R7999 R000 ~ R9499
A	信息请求信号	A0 ~ A24		A0 ~ A249
C	计数器	C0 ~ C79		C0 ~ C399 C5000 ~ C5199
K	保持继电器	K0 ~ K19		K0 ~ K99 K900 ~ K919
D	数据表	D0 ~ D1859		D0 ~ D9999
T	可变定时器	T0 ~ T79		T0 ~ T499 T9000 ~ T9499
L	标号	-	L1 ~ L9999	
P	子程序号	-	P1 ~ P512	P1 ~ P2000

序号	名称	模块名称(实际模块名称)	占用地址	说明
1	FANUC CNC SYSTEM FANUC Power Mate series	FS 04A	Input: 4 byte Output: 4 byte	
		FS 08A	Input: 4 byte Output: 4 byte	
		OC02I	Input: 4 byte Output: 4 byte	FANUC Power Mate-Model D/H
		OC02O	Input: 4 byte Output: 4 byte	
		OC03I	Input: 4 byte Output: 4 byte	
		OC03O	Input: 4 byte Output: 4 byte	
2	0I 用机床操作面板	OC01I	Input: 12 byte	0I 用机床操作面板
		OC01O	Output: 8 byte	
3	机床操作面板连接单元	/4	Input: 12 byte	Ordering drawing No. A16B-2200-0661 (Sink type) A16B-2200-0661 (Sink) A16B-2201-0731 (Source type)
		/8	Output: 8 byte	
4	机床操作面板接口单元	OC02I	Input: 16 bytes	
		OC02O	Output: 16 bytes	
		OC03I	Input: 32 bytes	
		OC03O	Output: 32 bytes	
5	I/O Link 连接单元	/_	Input: _ bytes Output: bytes	Specify 1 to 8 in _.
		OC02I	Input: 16 bytes	
		OC02O	Output: 16 bytes	
		OC03I	Input: 32 bytes	
		OC03O	Output: 32 bytes	

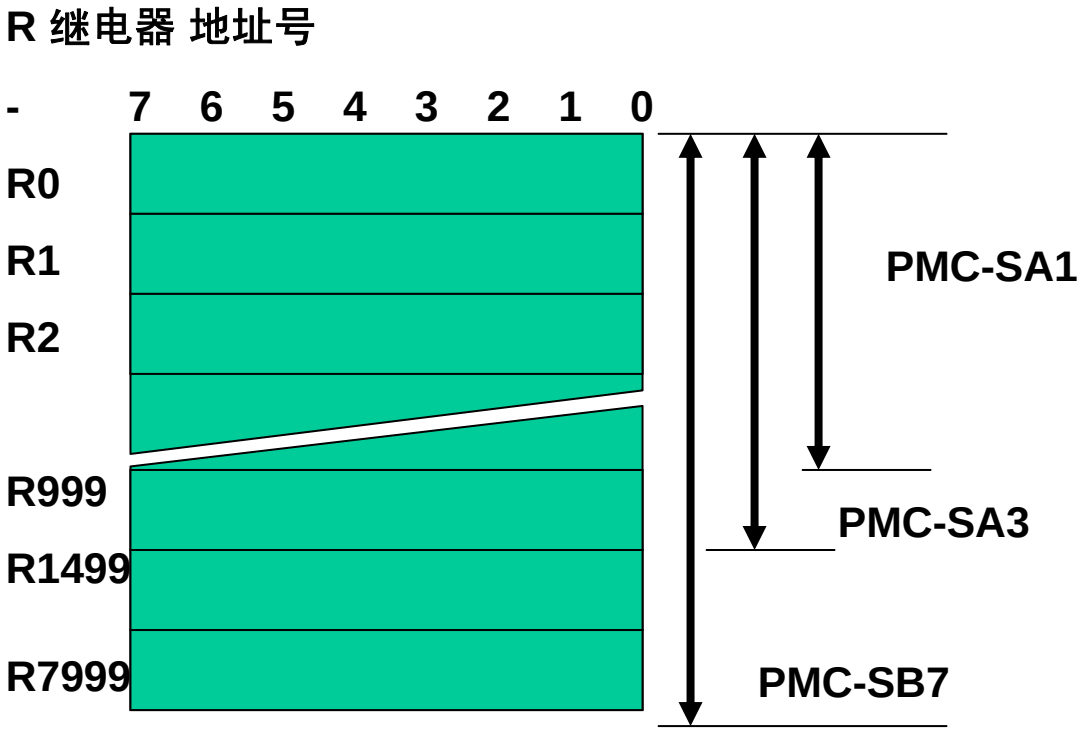
序号	名称	模块名称 (实际模块名称)	占用地址	说明
6	I/O 联接面板，I/O 模块(注 3，4)	CM03I(/3)	输入 3 字节	基本单元用
		CM06I(/6)	输入 6 字节	扩展单元 1 使用
		CM09I	输入 9 字节	扩展单元 2 使用
		CM12I(OC01I)	输入 12 字节	扩展单元 3 使用
		CM13I	输入 13 字节	首个 MPG 单元使用
		CM14I	输入 14 字节	第二个 MPG 单元使用
		CM15I	输入 15 字节	第三个 MPG 单元使用
		CM16I(OC02I)	输入 16 字节	DO 报警检测使用
		CM02O(/2)	输出 2 字节	基本单元用
		CM04O(/4)	输出 4 字节	扩展单元 1 使用
		CM06O(/6)	输出 6 字节	扩展单元 2 使用
		CM08O(/8)	输出 8 字节	扩展单元 3 使用
7	I/O 联接面板，I/O 模块	CM06I(/6)	输入 6 字节	
		CM13I	输入 13 字节	首个 MPG 单元使用
		CM14I	输入 14 字节	第二个 MPG 单元使用
		CM15I	输入 15 字节	第三个 MPG 单元使用
		CM16I(OC02I)	输入 16 字节	DO 报警检测使用
		CM04O(/4)	输出 4 字节	
		CM08O(/8)	输出 8 字节	

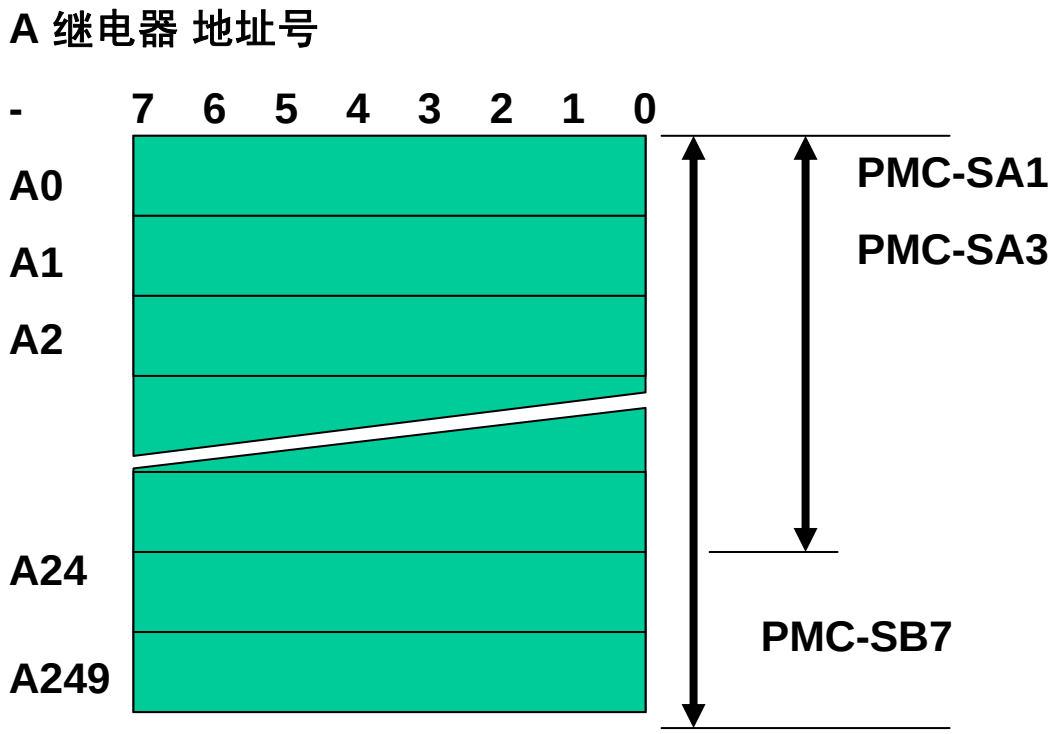
I/O Llink 模块设定例1

ADDRESS	GROUP	BASE	SLOT	NAME
X000				
X001				
X002				
X003				
X004				
X005	0	0	5	ID32A
X006	0	0	5	ID32A
X007	0	0	5	ID32A
X008	0	0	5	ID32A
X009				

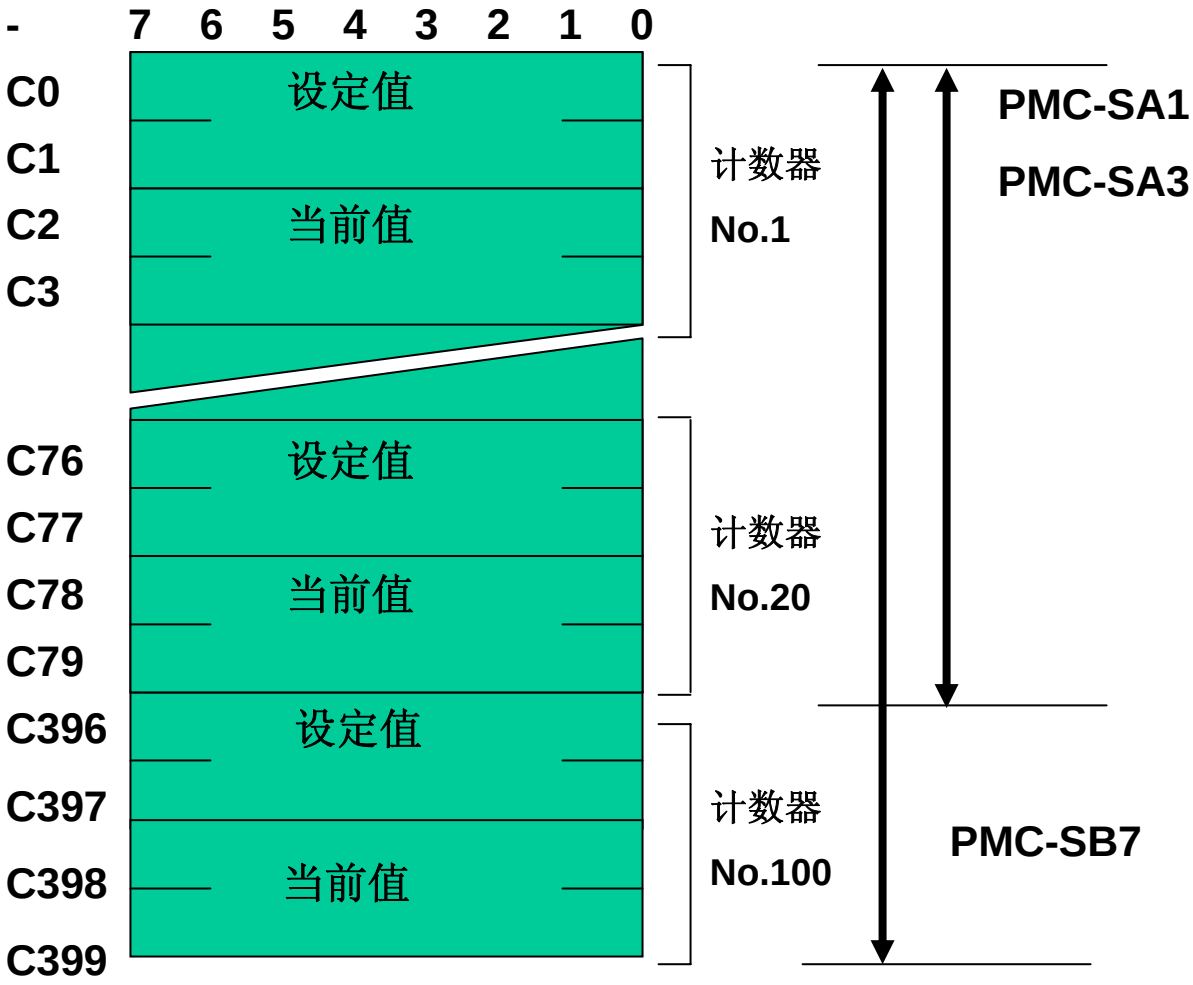
I/O Llink 模块设定例2

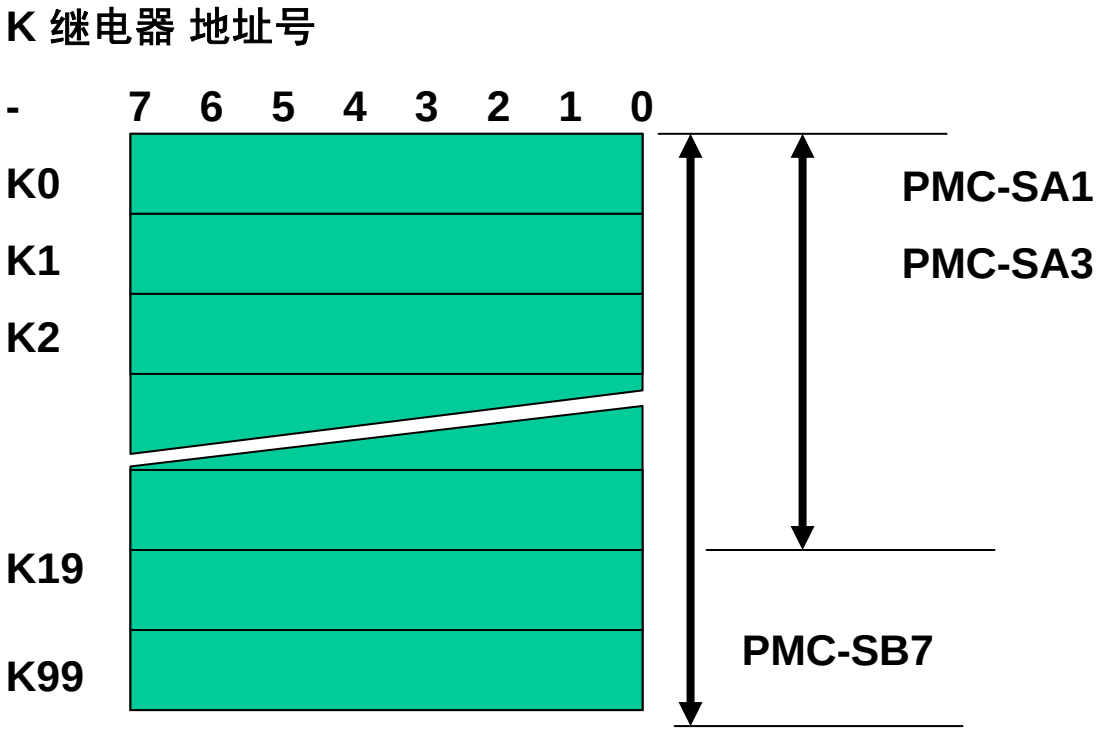
ADDRESS	GROUP	BASE	SLOT	NAME
X000	0	0	1	ID16C
X001	0	0	1	ID16C
X002	0	0	2	ID16D
X003	0	0	2	ID16D
X004	1	0	1	IA16G
X005	1	0	1	IA16G
X006	1	0	2	IA16G
X007	1	0	2	IA16G
X008	2	0	1	ID16D
X009	2	a 0	1	ID16D

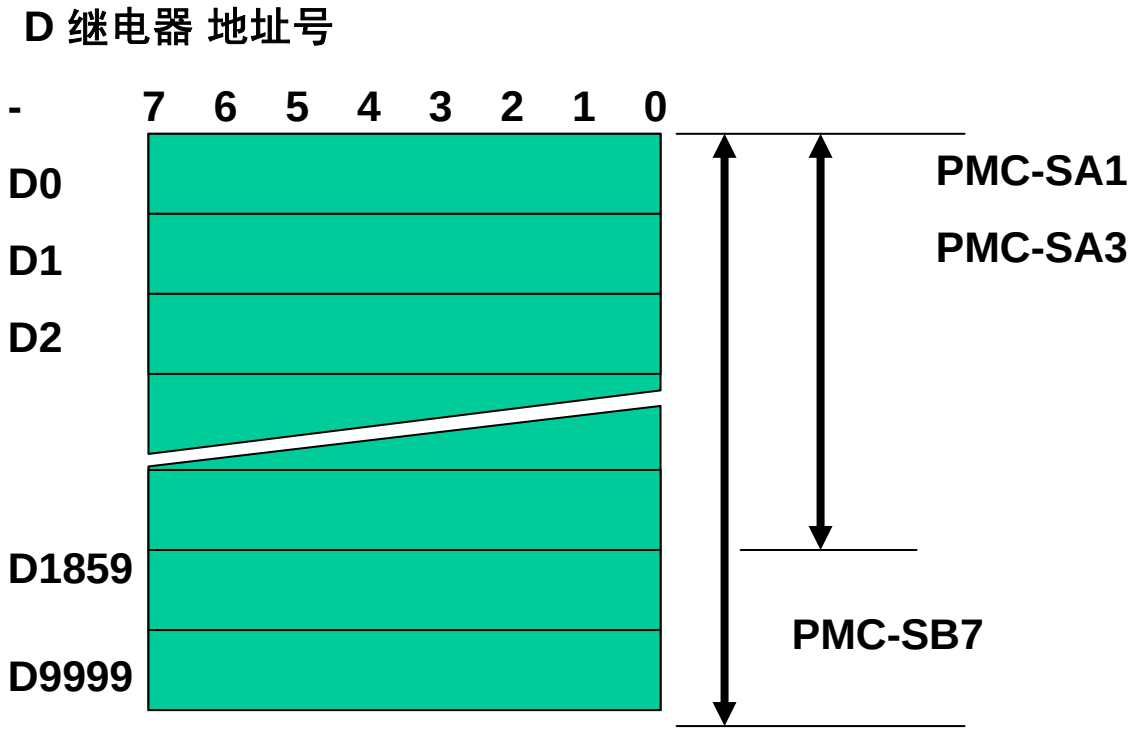

R 继电器 地址号
- 7 6 5 4 3 2 1 0
R0
R1
R2
R999
R1499
R7999
PMC-SA1
PMC-SA3
PMC-SB7

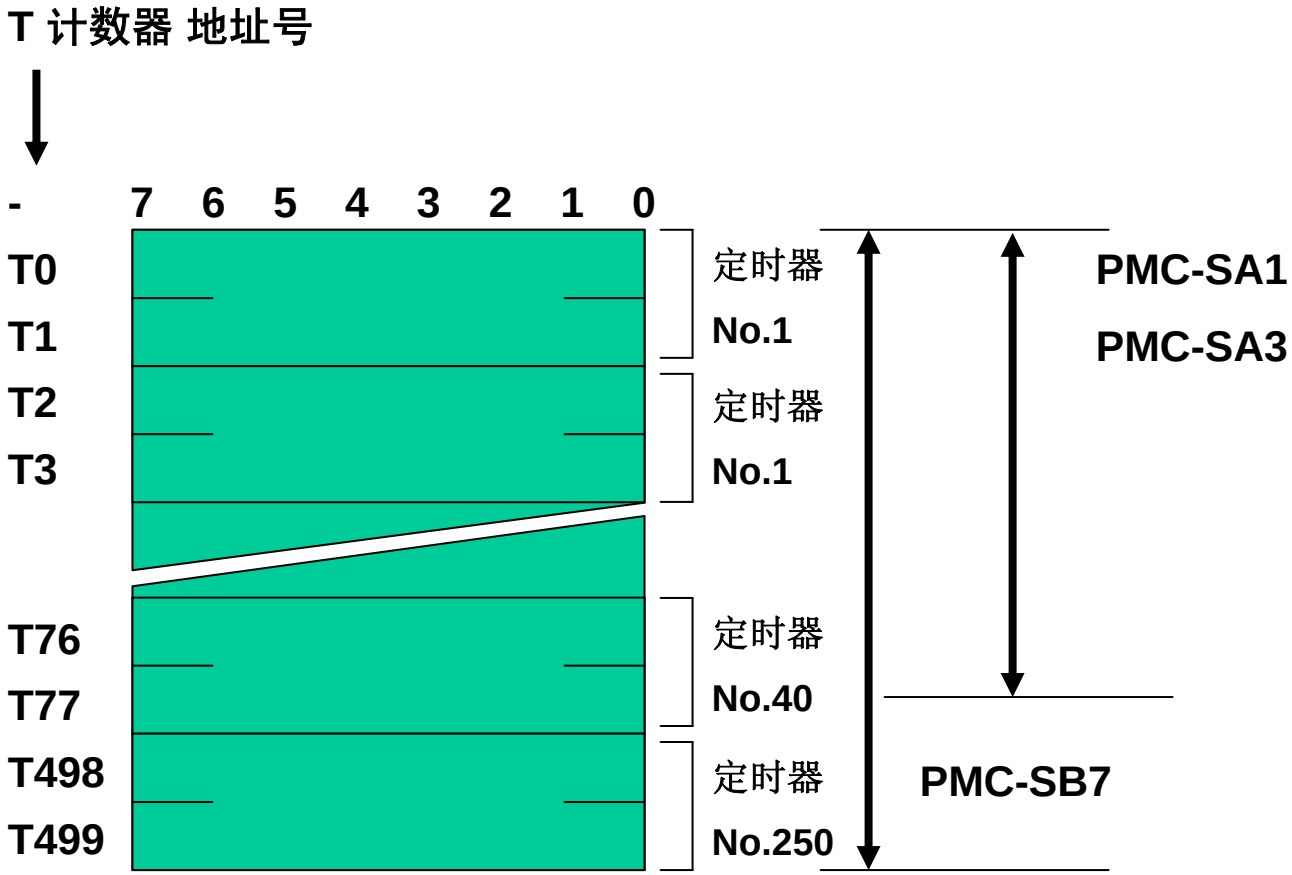

A 继电器 地址号
-
7
6
5
4
3
2
1
0
A0
A1
A2
A24
A249
PMC-SA1
PMC-SA3
PMC-SB7

C 计数器 地址号


- 7 6 5 4 3 2 1 0
C0
C1
C2
C3
设定值
当前值
计数器
No.1
C76
C77
C78
C79
设定值
当前值
计数器
No.20
C396
C397
C398
C399
设定值
当前值
计数器
No.100
PMC-SA1
PMC-SA3
PMC-SB7


K 继电器 地址号
- 7 6 5 4 3 2 1 0
K0
K1
K2
K19
K99
PMC-SA1
PMC-SA3
PMC-SB7


D 继电器 地址号
-
7 6 5 4 3 2 1 0
D0
D1
D2
D1859
D9999
PMC-SA1
PMC-SA3
PMC-SB7


T 计数器 地址号
- 7 6 5 4 3 2 1 0
T0
T1
T2
T3
T76
T77
T498
T499
定时器
No.1
定时器
No.1
定时器
No.40
定时器
No.250
PMC-SA1
PMC-SA3
PMC-SB7

规格	SA1	SA3	SB7
标记数	-	9999	9999

规格	SA1	SA3	SB7
子程序数量	-	512	2000

顺序程序的执行过程

在一般的继电器控制电路中，各继电器在时间上完全可以同时动作，在下图所举例中，当继电器 A 动作时，继电器 D 和 E 可同时动作(当触点 B 和 C 都闭合时)。在 PMC 顺序控制中，各个继电器依次动作。当继电器 A 动作时，继电器 D 首先动作，然后继电器 E 才动作(见图 2.1(a))。即各个继电器按梯形图中的顺序(编辑次序)动作。

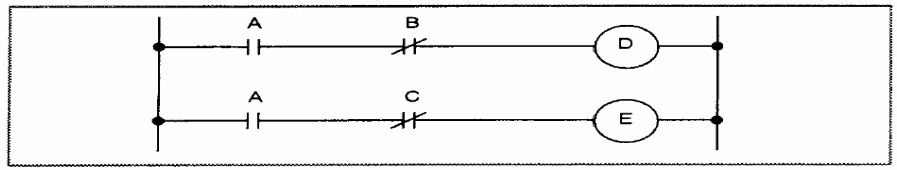


图 2.1(a)　电路举例

图 2.1(b)(A)和(B)图示了继电器电路和 PMC 程序动作之间的区别。

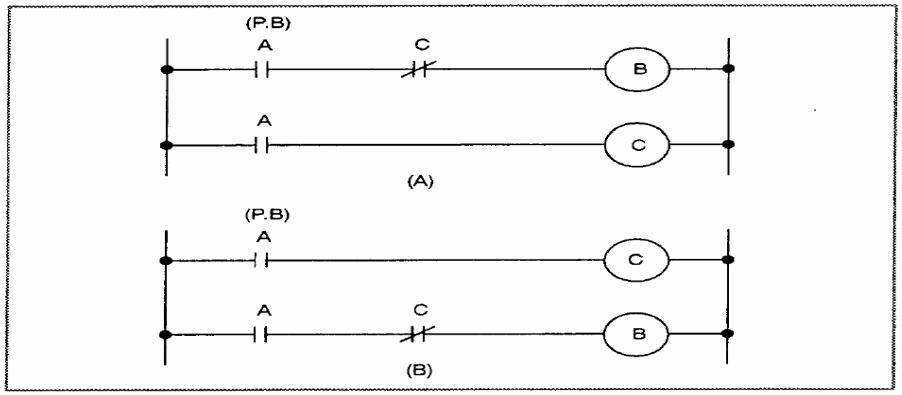


图 2.1(b)　电路举例

(1)继电器电路

图 2.1(b)(A)和(B)中的动作相同。接通 A(按钮开关)后线圈 B 和 C 中有电流通过，B 和 C 接通。C 接通之后 B 断开。

(2)PMC 程序

图 2.1(b)(A)中，同继电器电路一样，接通 A(按扭开关)后 B、C 接通，经过 PMC 程序的一个循环后 B 关断。但在图 2.1(b)(B)中，接通 A(按扭开关)后 C 接通，但 B 并不接通。

PMC上处理的数据形式

●带符号二进制形式(Binary)

●可进行1字节、2字节、4字节长的2进数(二进制)处理。

●可使用的数值范围如下。

数据长度	数据范围(10进数换算)	备 注
1字节数据	−128～+127	2的补码表示
2字节数据	−32768～+32767	
4字节数据	−2147483648～+2147483647	

●在顺序程序中指定数据的初始地址和数据长度。

●在用诊断画面(PMCDGN)确认2字节、4字节长的数据时，地址号大的为高位地址。

📖 由R100指定4字节长的区域时的地址和位的关系如下。

		#7	#6	#5	#4	#3	#2	#1	#0
地址	R0100	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
地址	R0101	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^9	2^8
地址	R0102	2^{23}	2^{22}	2^{21}	2^{20}	2^{19}	2^{18}	2^{17}	2^{16}
地址	R0103	±	2^{30}	2^{29}	2^{28}	2^{27}	2^{26}	2^{25}	2^{24}

●例：用2字节表示100和−100时：

10 进 数		100	−100	
2进数	＋0	01100100	10011100	最高位位是1时为负
	＋1	00000000	11111111	

●BCD形式（Binary Coded Decimal）（二—十进制）

- ●在10进数的二-十进制中，用4位的2进数表示10进数的各位。
- ●可处理2位或4位的10进数。符号用其他信号进行处理。

		#7	#6	#5	#4	#3	#2	#1	#0
地址	+0	10 位				个 位			
		80	40	20	10	8	4	2	1
	+1	1000 位				100 位			
		8000	4000	2000	1000	800	400	200	100

- ●例：

10 进数		63	1234
BCD码	+0	01100011	00110100
	+1	—	00010010

📖 有BCD码和2进数（二进制）的变换命令DCNV、DCNVB。

●位型（Bit）

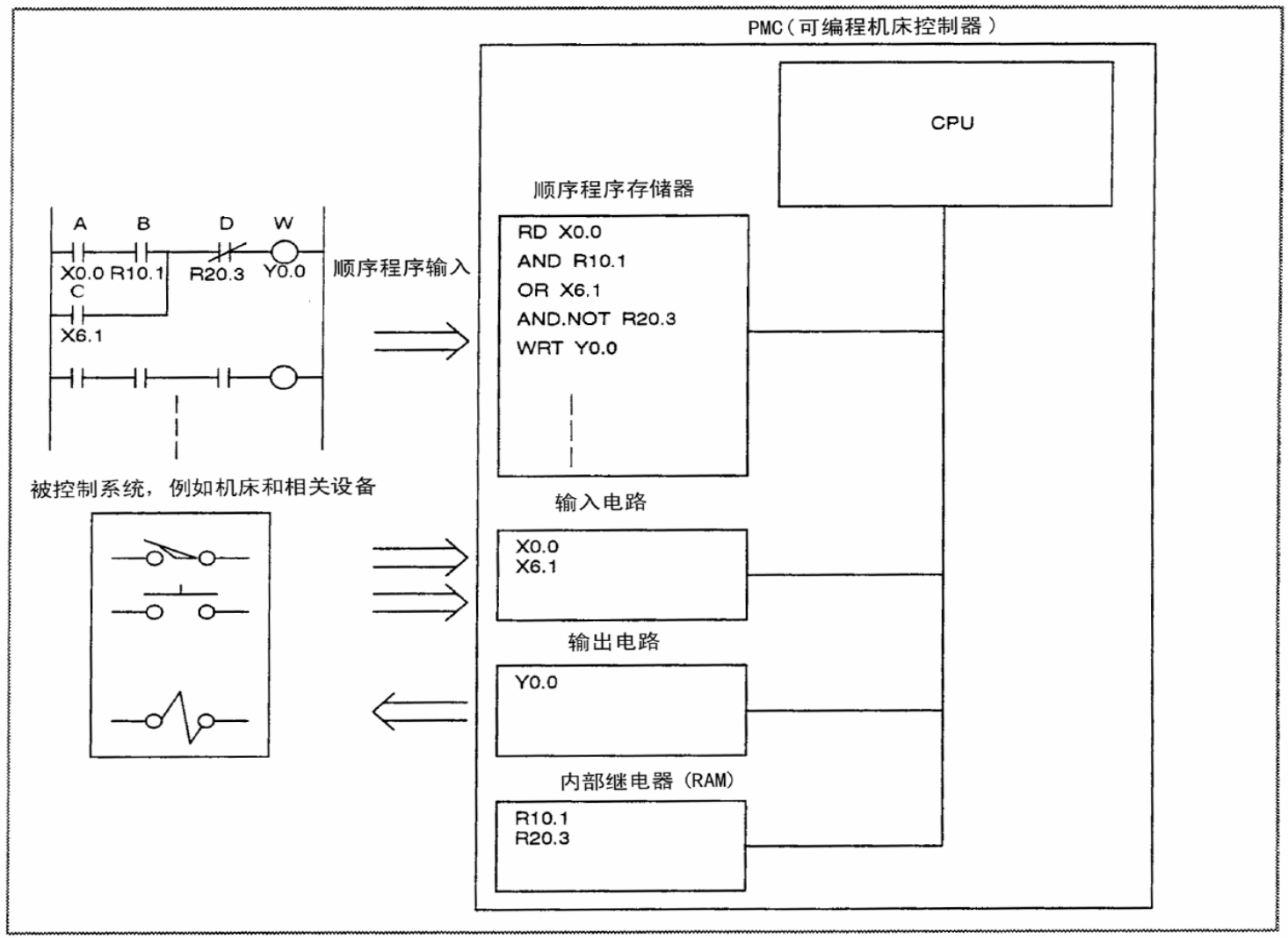
- ●处理1位信号和数据时，在地址之后指定小数点的位号。

		#7	#6	#5	#4	#3	#2	#1	#0
地址	xxxxx								

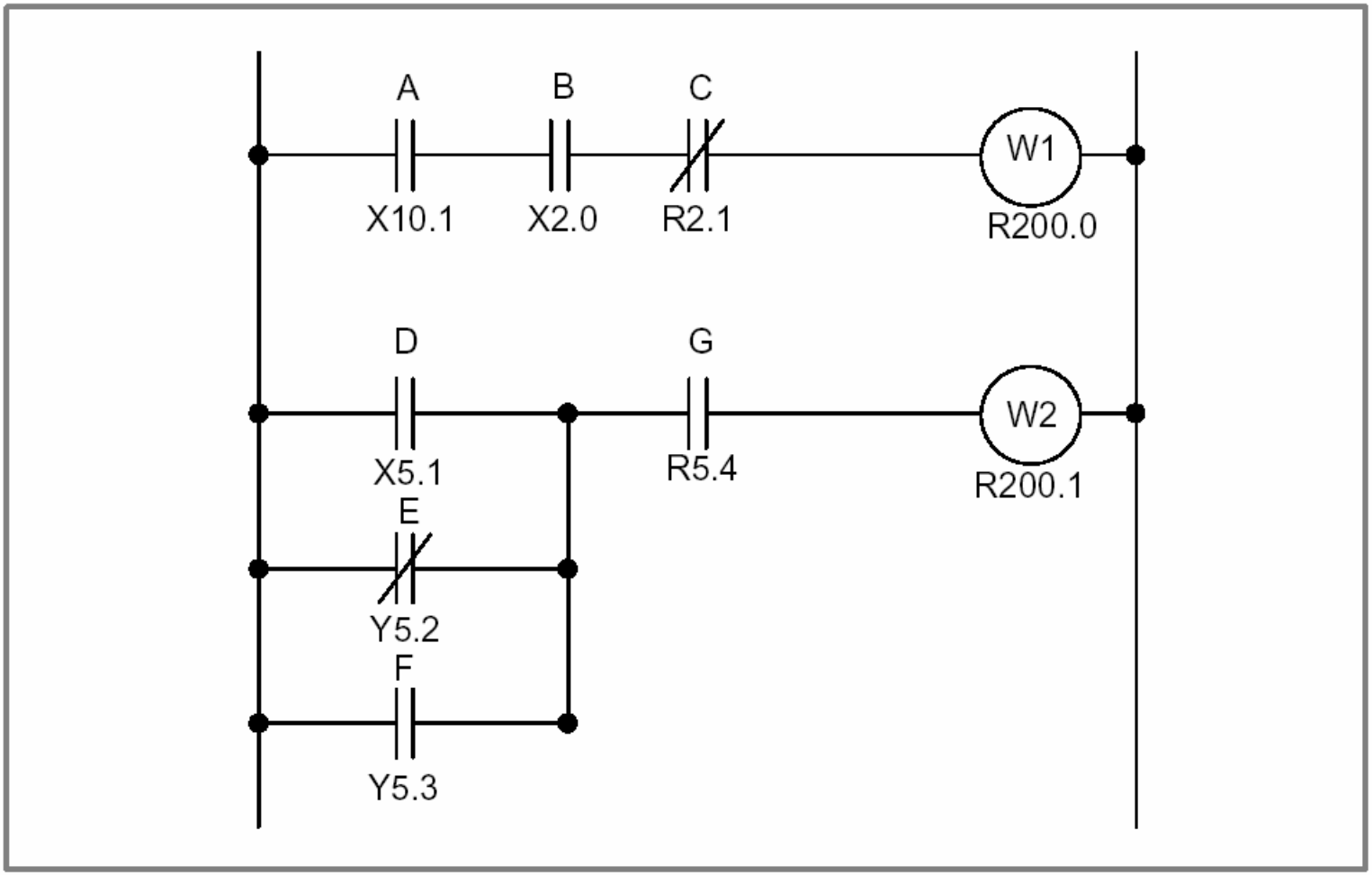
- ●例 ： X0001.2（地址X0001的位2）

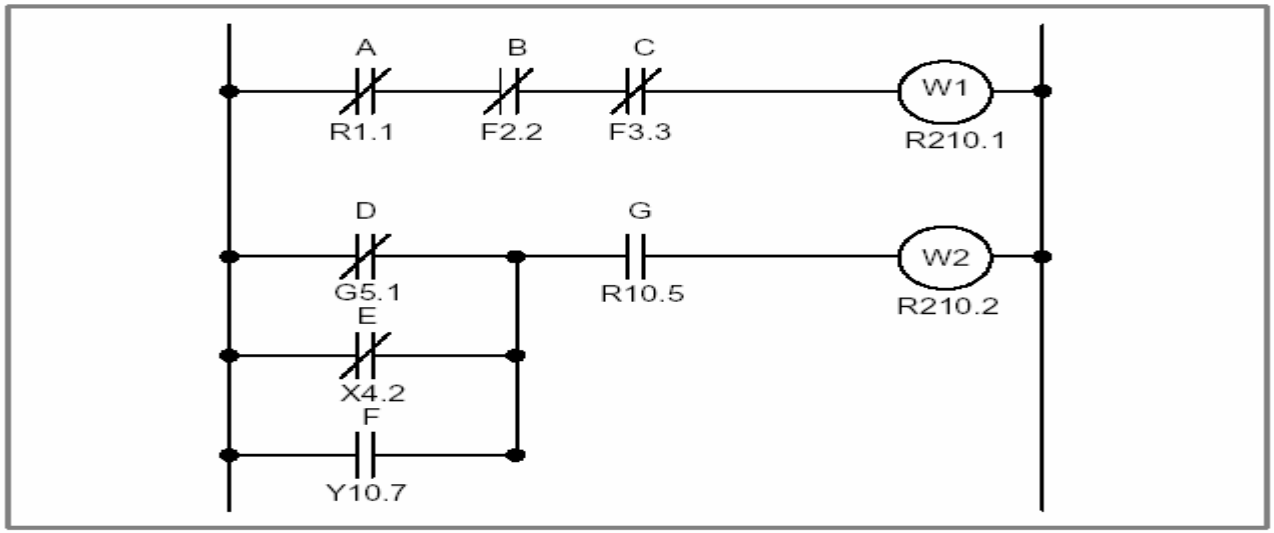
📖 可以以位为单位读写数据表等的部分数值数据。

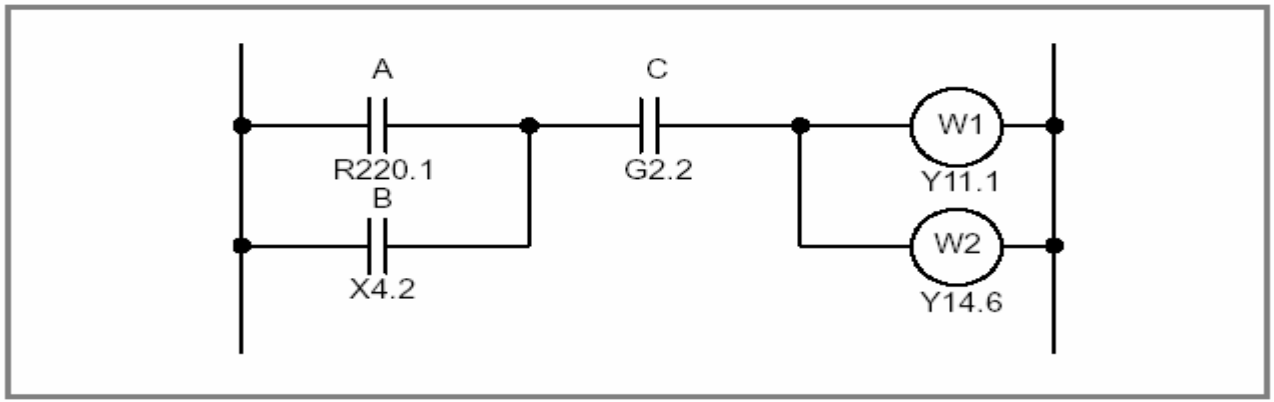
PMC 执行顺序程序

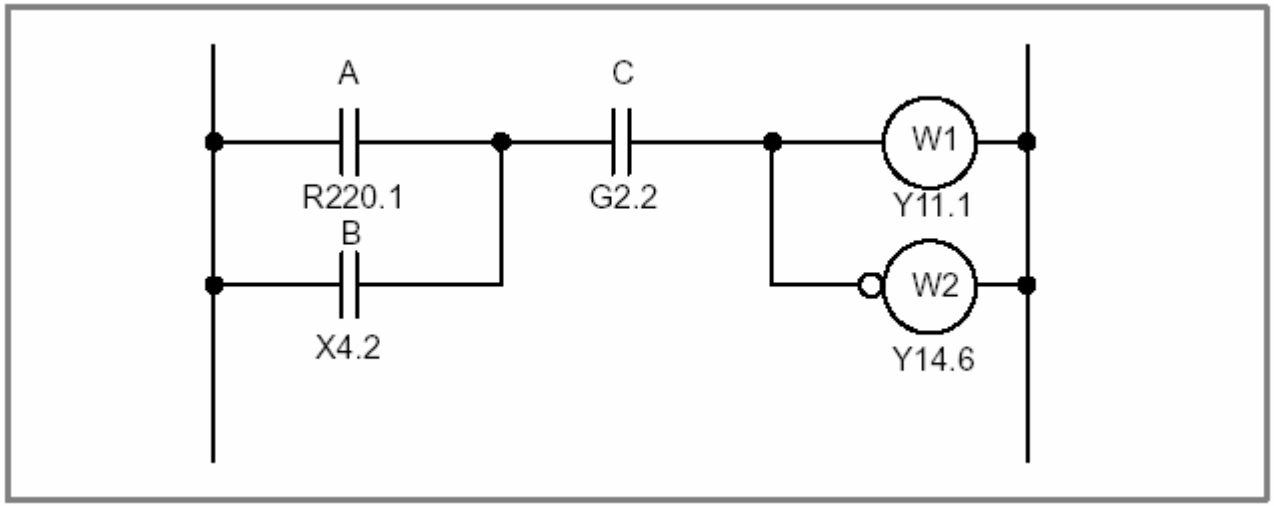

PMC（可编程机床控制器）
CPU
顺序程序存储器
RD X0.0
AND R10.1
OR X6.1
AND.NOT R20.3
WRT Y0.0
A
B
D
W
X0.0
R10.1
R20.3
Y0.0
C
X6.1
顺序程序输入
被控制系统，例如机床和相关设备
输入电路
X0.0
X6.1
输出电路
Y0.0
内部继电器（RAM）
R10.1
R20.3

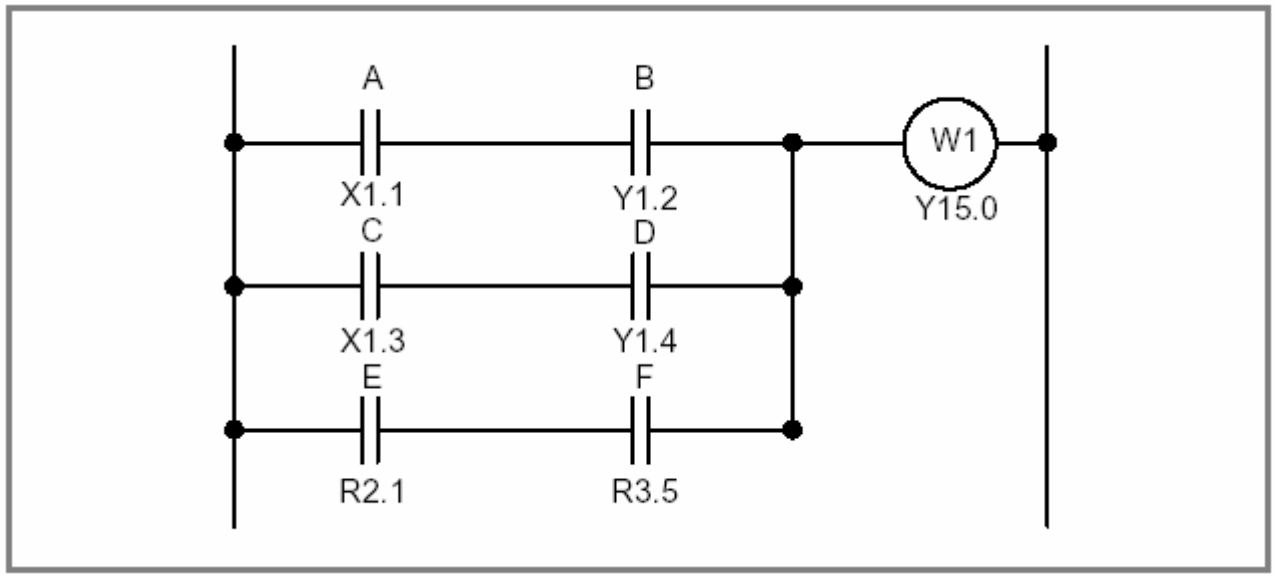
基本指令

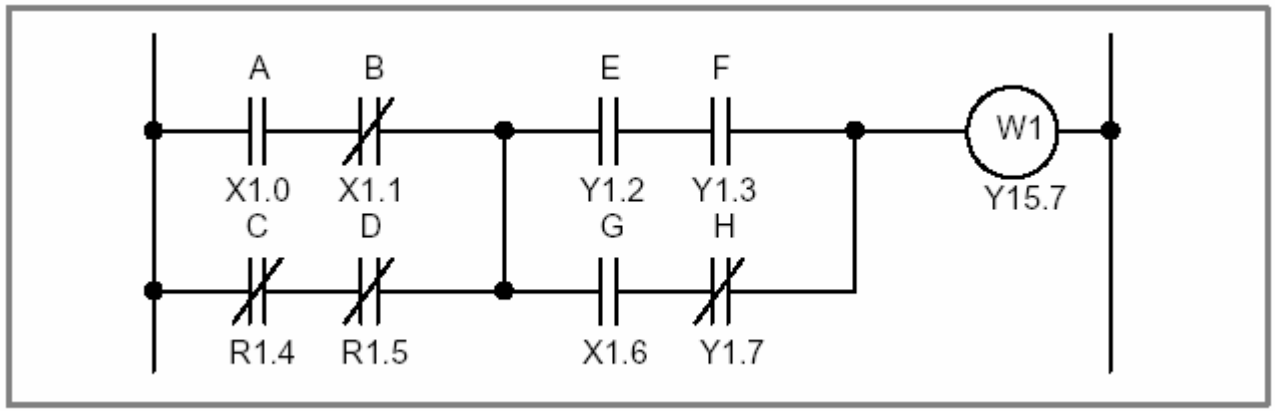


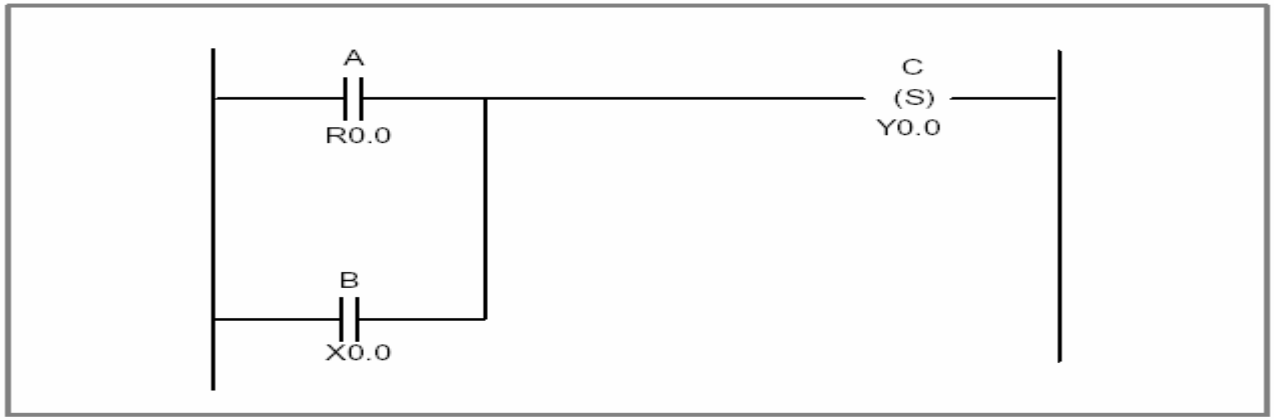

A
B
C
R1.1
F2.2
F3.3
W1
R210.1
D
G
G5.1
R10.5
W2
R210.2
E
X4.2
F
Y10.7

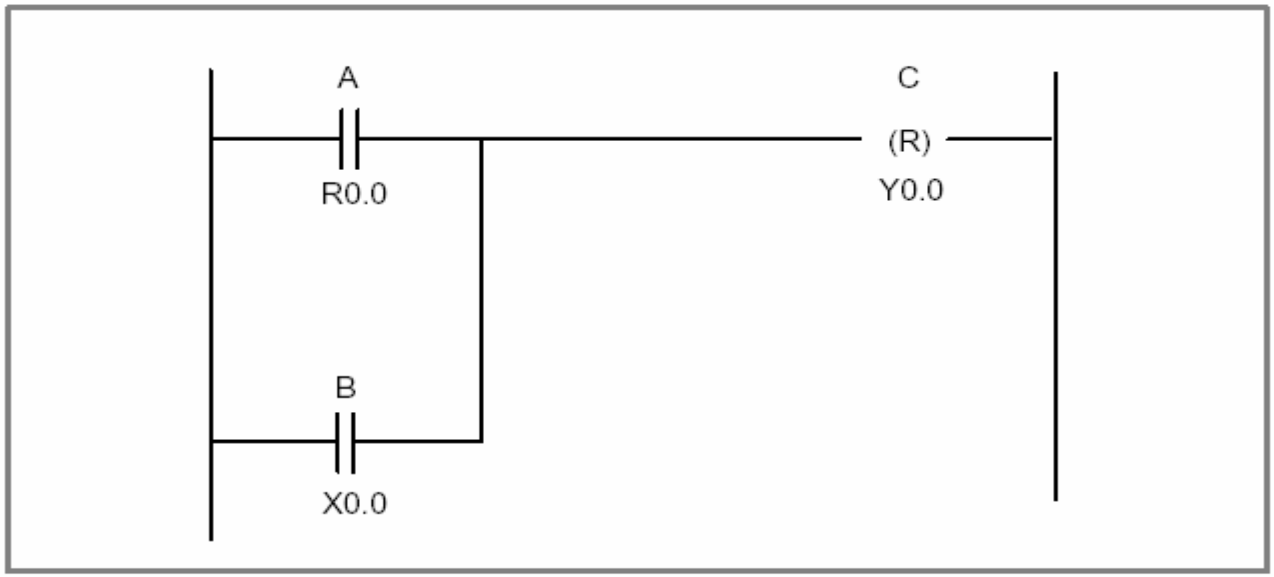

A
C
R220.1
G2.2
W1
Y11.1
B
X4.2
W2
Y14.6


A
R220.1
B
X4.2
C
G2.2
W1
Y11.1
W2
Y14.6

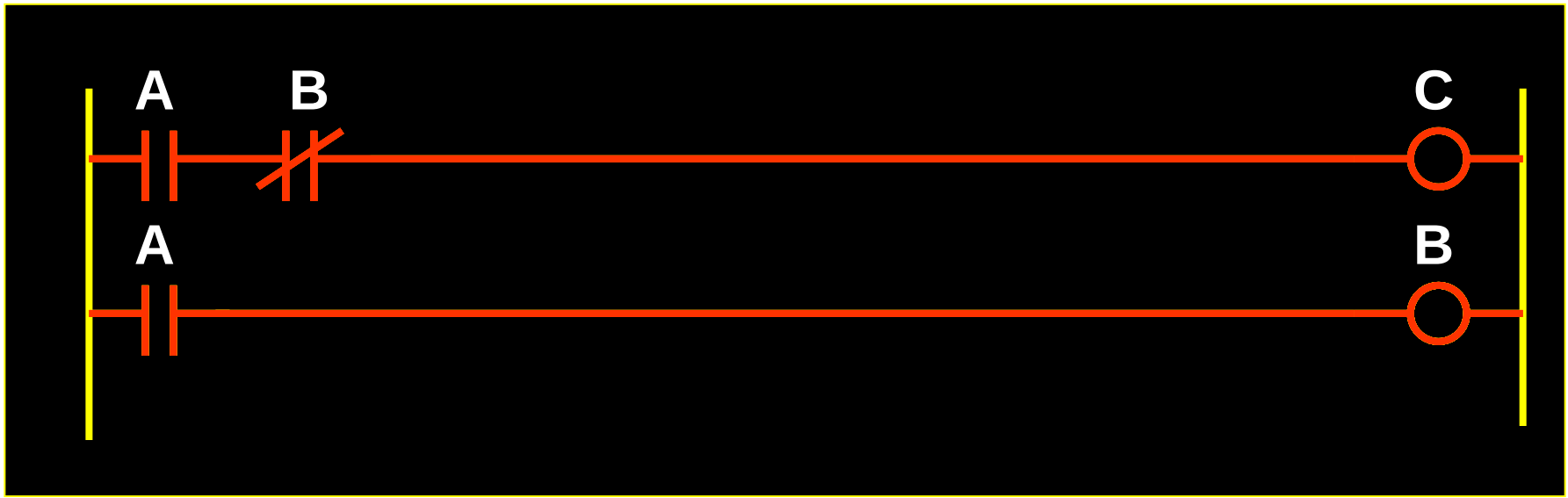

A
X1.1
C
X1.3
E
R2.1
B
Y1.2
D
Y1.4
F
R3.5
W1
Y15.0

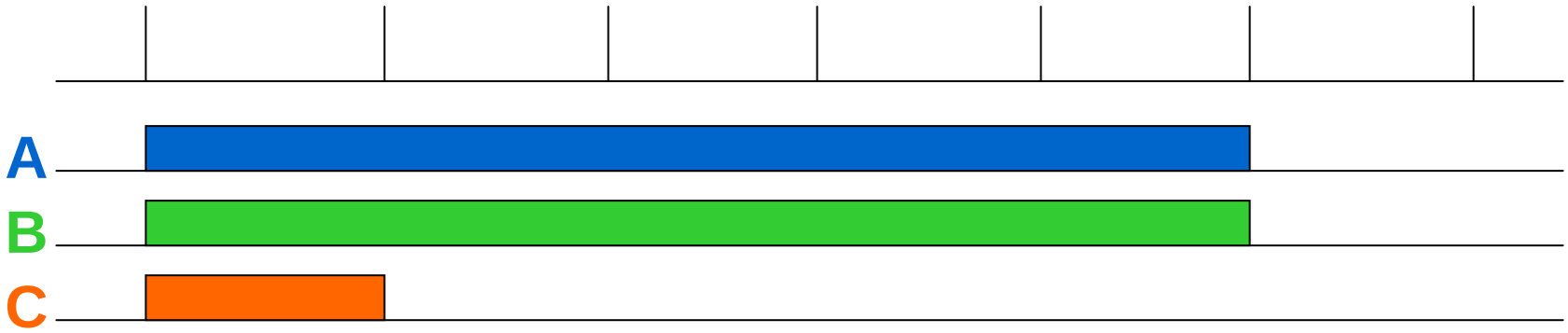

A
B
E
F
W1
X1.0
C
X1.1
D
Y1.2
G
Y1.3
H
Y15.7
R1.4
R1.5
X1.6
Y1.7


A
R0.0
B
X0.0
C
(S)
Y0.0

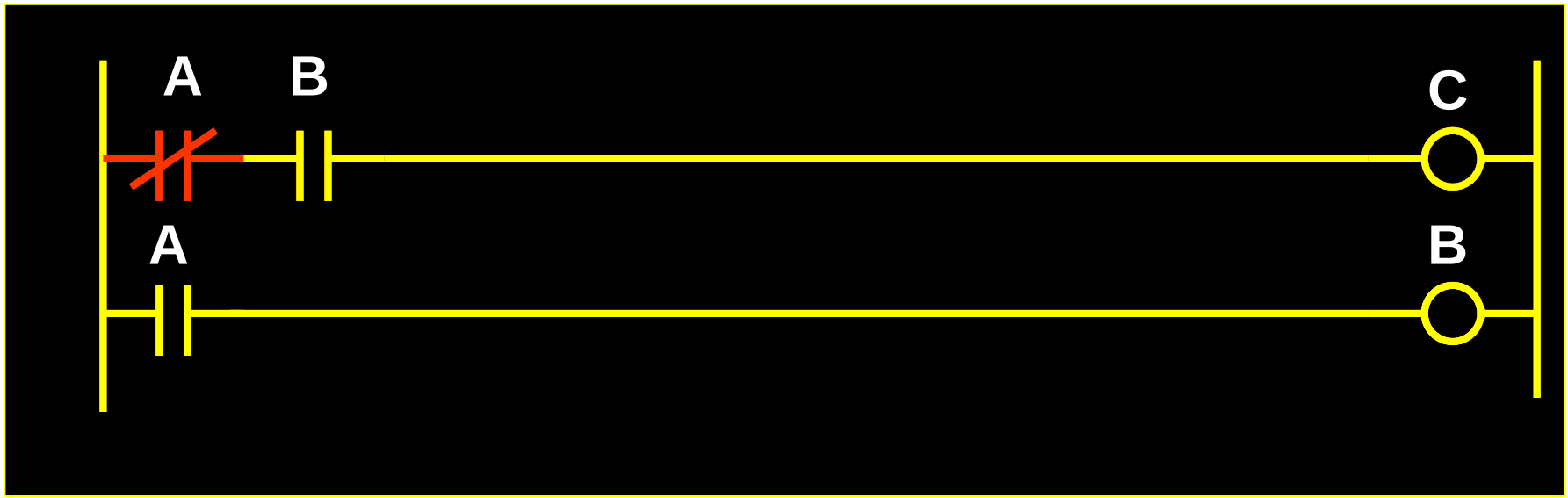

A
R0.0
B
X0.0
C
(R)
Y0.0

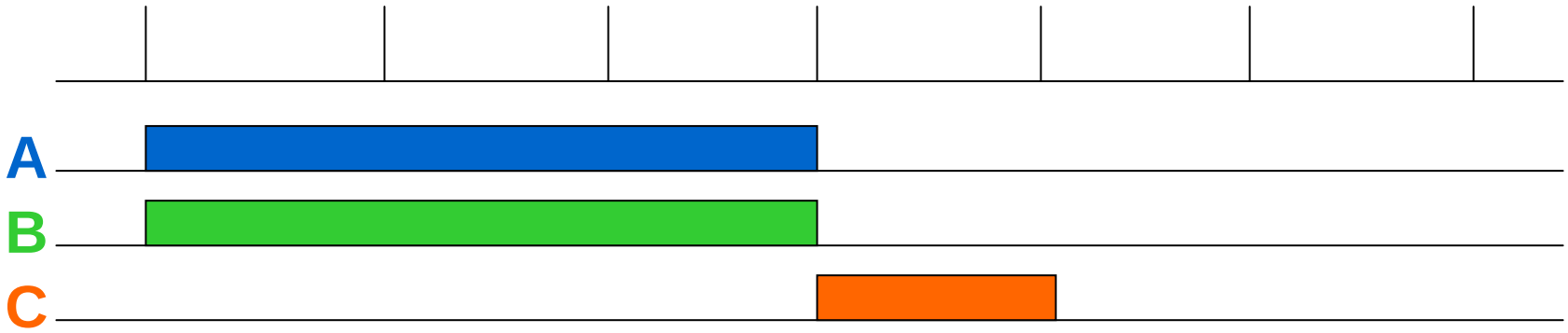
上升沿产生单脉冲的时序图



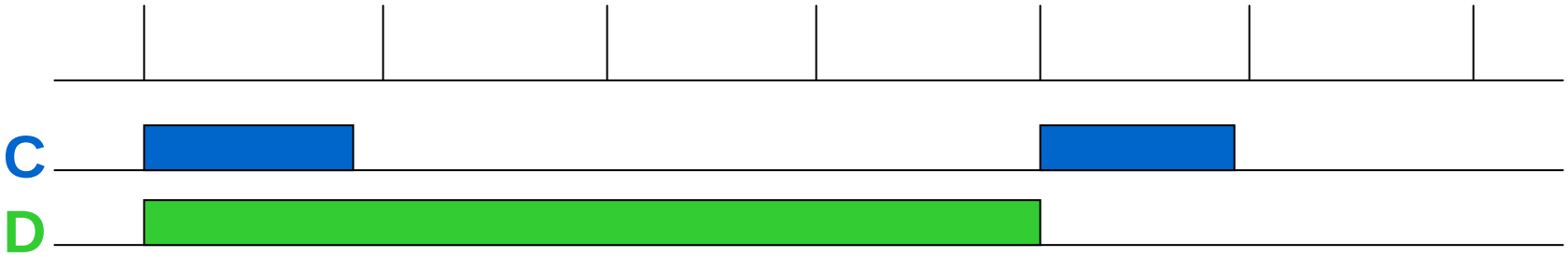


下降沿产生单脉冲的时序图





单信号接通断开时序图



数控机床用PMC的指令必须满足数控机床信息处理和动作控制的特殊要求。例如，由NC输出的M，S，T二进制代码信号的译码，机械部件动作状态或液压系统动作状态的延时确认，加工零件记数，刀库，分度台沿最短路径旋转和现在位置至目标位置步数的计算等。

在为数控机床编辑顺序程序时，对于上述译码、定时、记数、最短路径选择，以及比较、检索、代码转换、数据四则运算、信息显示等控制功能，仅用执行一位操作的基本指令编程，实现起来将会十分困难。因此，就需要增加一些具有专门控制功能的指令来解决基本指令无法处理的那些控制问题。这些专门指令就是“功能指令”

功能命令

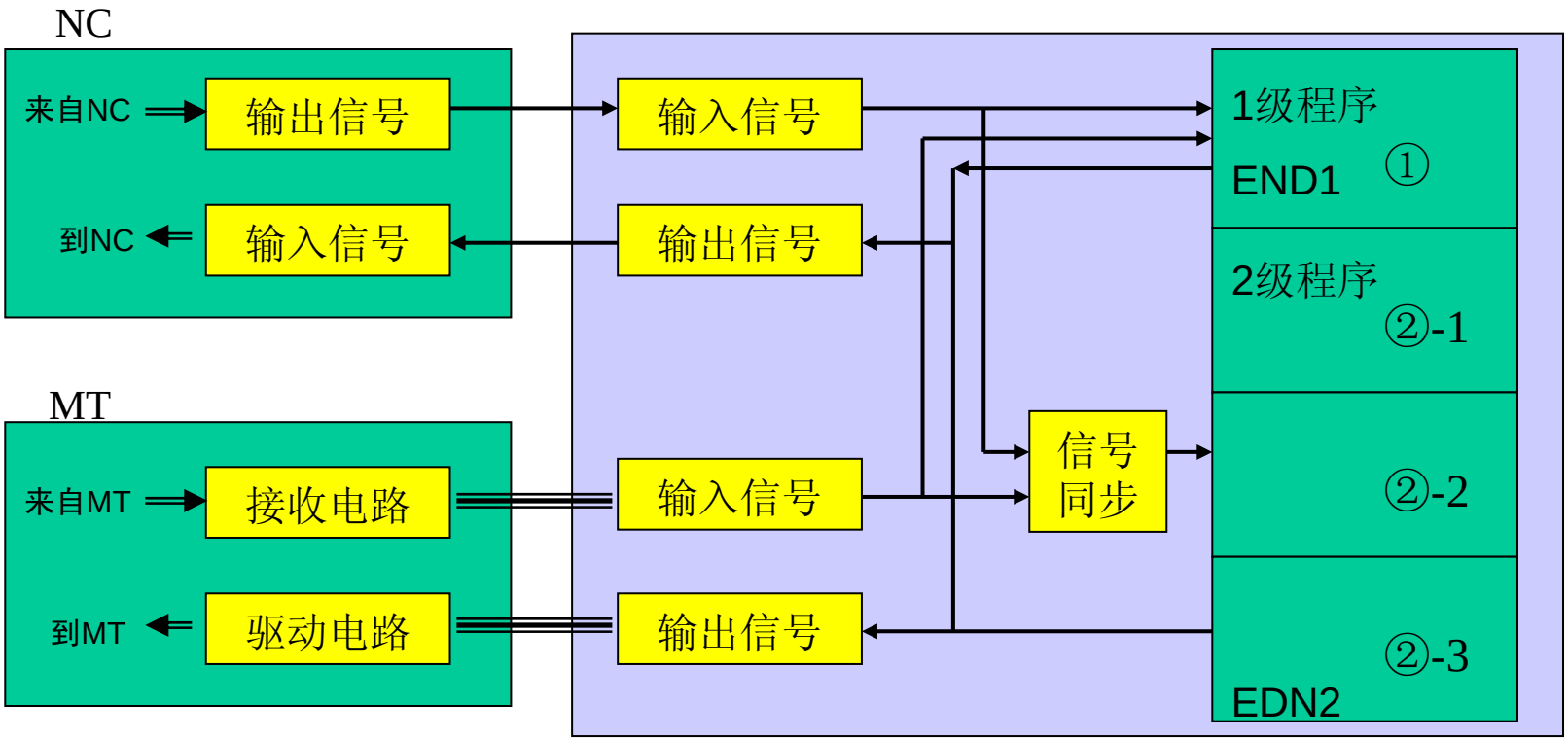
编号	功能名	命令号	处理内容	SA1	SA5 SB5 SB6 SB7
1	END1	SUB 1	第1级程序结束	○	○
2	END2	SUB 2	第2级程序结束	○	○
3	END3	SUB 48	第3级程序结束	×	×
4	TMR	SUB 3	定时器处理	○	○
5	TMRB	SUB 24	固定定时器处理	○	○
6	TMRC	SUB 5 4	追加定时器处理	○	○
7	DEC	SUB 4	BCD译码处理	○	○
8	DECB	SUB 25	二进制译码处理	○	○
9	CTR	SUB 5	计数器处理	○	○
10	CTRC	SUB 55	追加计数器处理	○	○
11	ROT	SUB 6	BCD回转控制	○	○
12	ROTB	SUB 26	二进制回转控制	○	○
13	COD	SUB 7	BCD码变换	○	○
14	CODB	SUB 27	二进制码变换	○	○

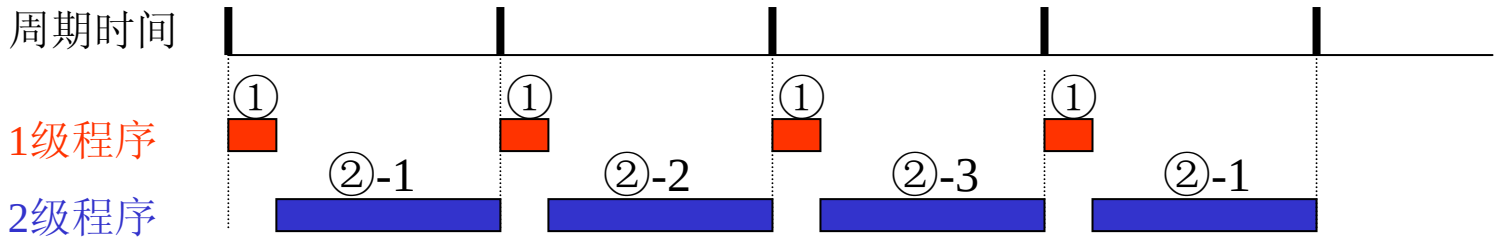
编号	功能名	命令号	处理内容	SA1	SA5 SB5 SB6 SB7
15	MOVE	SUB 8	逻辑乘后数据转送	○	○
16	MOVOR	SUB 28	逻辑加后数据转送	○	○
17	MOVB	SUB 43	1字节数据转送	×	○
18	MOVW	SUB 44	2字节数据转送	×	○
19	MOVN	SUB 45	任意字节数据转送	×	○
20	COM	SUB 9	公用线控制开始	○	○
21	COME	SUB 29	公用线控制结束	○	○
22	JMP	SUB 10	跳转	○	○
23	JMPE	SUB 30	跳转结束	○	○
24	JMPB	SUB 68	标号跳转1	×	○
25	JMPC	SUB 73	标号跳转2	×	○
26	LBL	SUB 69	标号	×	○
27	PARI	SUB 11	奇偶校验	○	○
28	DCNV	SUB 14	数据变换	○	○
29	DCNVB	SUB 31	扩展数据变换	○	○
30	COMP	SUB 15	BCD大小比较	○	○
31	COMPB	SUB 32	二进制大小比较	○	○
32	COIN	SUB 16	BCD一致判断	○	○
33	SFT	SUB 33	移位寄存器	○	○

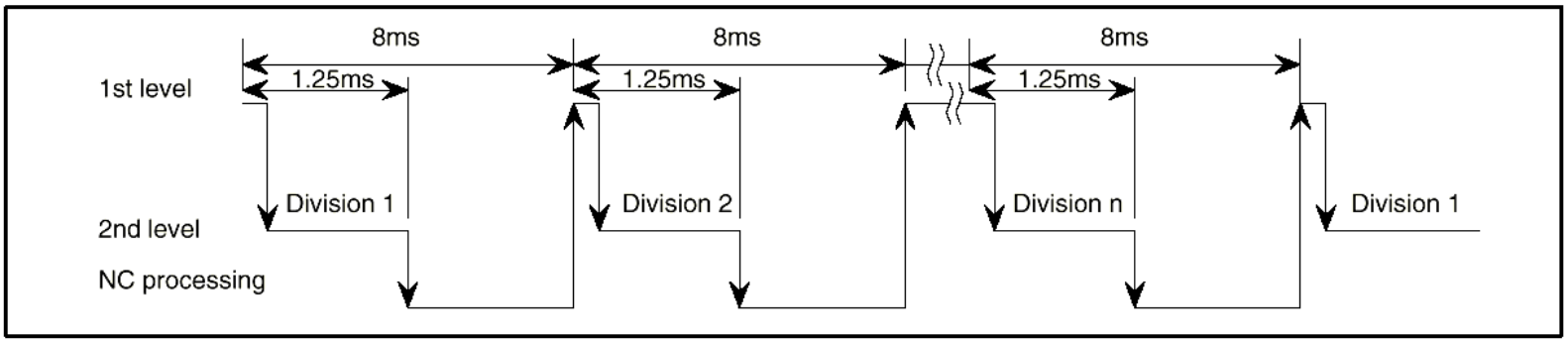
编号	功能名	命令号	处理内容	SA1	SA5 SB5 SB6 SB7
34	DSCH	SUB 17	BCD数据检索	○	○
35	DSCHB	SUB 34	二进制数据检索	○	○
36	XMOV	SUB 18	BCD变址修改 数据转送	○	○
37	XMOVB	SUB 35	二进制变址修改 数据转送	○	○
38	ADD	SUB 19	BCD加法运算	○	○
39	ADDB	SUB 36	二进制加法运算	○	○
40	SUB	SUB 20	BCD减法运算	○	○
41	SUBB	SUB 37	二进制减法运算	○	○
42	MUL	SUB 21	BCD乘法运算	○	○
43	MULB	SUB 38	二进制乘法运算	○	○
44	DIV	SUB 22	BCD除法运算	○	○
45	DIVB	SUB 39	二进制除法运算	○	○
46	NUME	SUB 23	BCD常数赋值	○	○
47	NUMEB	SUB 40	二进制常数赋值	○	○

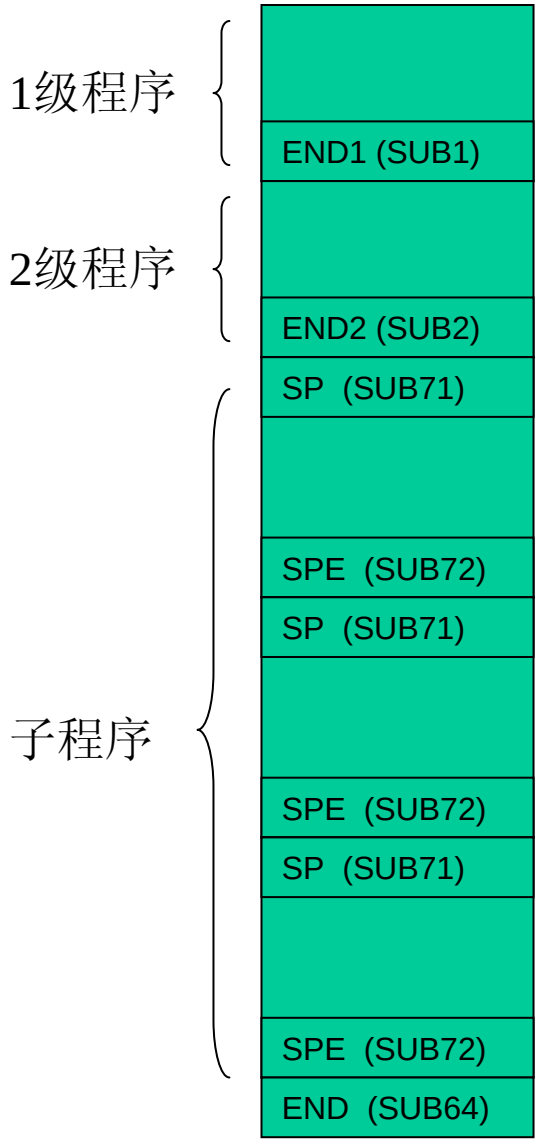
编号	功能名	命令号	处理内容	SA1	SA5 SB5 SB6 SB7
48	DISPB	SUB 41	信息显示	○	○
49	EXIN	SUB 42	外部数据输入	○	○
50	WINDR	SUB 51	CNC数据读取	○	○
51	WINDW	SUB 52	CNC数据写入	○	○
52	DIFU	SUB 57	前沿检测	×	○
53	DIFD	SUB 58	后沿检测	×	○
54	EOR	SUB 59	异或	×	○
55	AND	SUB 60	逻辑乘	×	○
56	OR	SUB 61	逻辑和	×	○
57	NOT	SUB 62	逻辑非	×	○
58	END	SUB 64	程序结束	×	○
59	CALL	SUB 65	有条件子程序调出	×	○
60	CALLU	SUB 66	子程序调出	×	○
61	SP	SUB 71	子程序开始	×	○
62	SPE	SUB 72	子程序结束	×	○

(1) 第1级结束	SUB 1 END1	第1级顺序结束。
(2) 第2级结束	SUB 2 END2	第2级顺序结束。
(3) 第3级结束	SUB48 END3	第3级顺序结束。 ☞ 可在PMC-RC系列上使用


NC
来自NC
输出信号
到NC
输入信号
MT
来自MT
接收电路
到MT
驱动电路
输入信号
输出信号
输入信号
输出信号
信号
同步
1级程序
END1
①
2级程序
②-1
②-2
②-3
EDN2


周期时间
1级程序
2级程序
①
②-1
②-2
②-3


8ms
8ms
8ms
1st level
1.25ms
1.25ms
1.25ms
Division 1
Division 2
Division n
Division 1
2nd level
NC processing


1级程序
END1 (SUB1)
2级程序
END2 (SUB2)
SP (SUB71)
SPE (SUB72)
SP (SUB71)
子程序
SPE (SUB72)
SP (SUB71)
SPE (SUB72)
END (SUB64)

● 程序的级别和输入输出的信号处理

●2级程序

●第1级是每隔8ms进行读取的程序。

- 主要是急停、跳转、超程等紧急动作的处理。
- 不使用第1级时，只编写END1 命令。

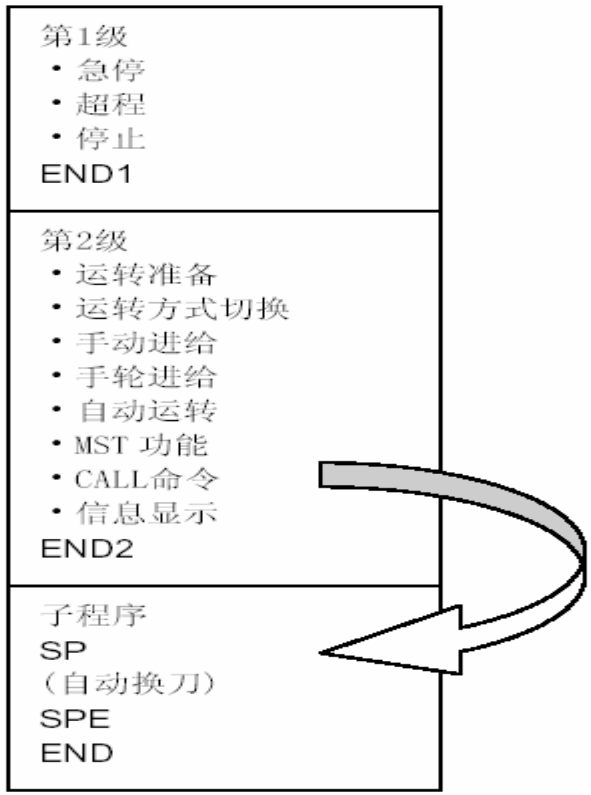
●在第2级上编写普通的顺序程序。

- Seguence是“顺序”的意思。
  ATC(自动换刀装置)在第2级上编写。
- 在第2级上因为有同步输入信号存储器，所以输入脉冲信号时，其信号宽度应大于扫描时间。
- 扫描时间显示在PMC诊断(PMCDGN)的标题栏(TITLE)上。

● 子程序

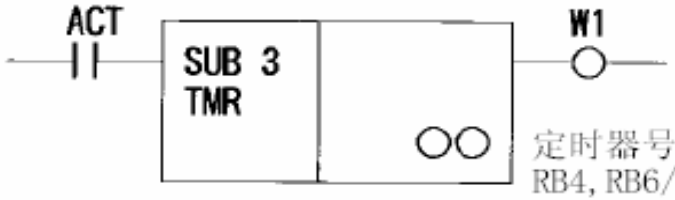
将重复执行的处理和模块化的程序作为子程序登录，然后用CALL或CALLU命令由第2级调用。

第1级不能调用子程序。
另外，在PMC-SA1上不能使用子程序功能。



(4)
定时器

是延时定时器。ACT=1后经过设定的时间时，输出W1即接通。

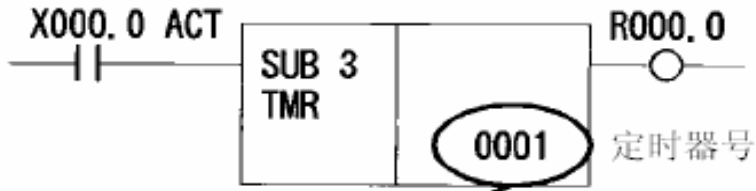


1～8号为48ms单位
（最大1572.8秒）
9号以后为8ms单位
（最大262.1秒）

定时器号
RB4,RB6/RC4为1～150
其它为1～40

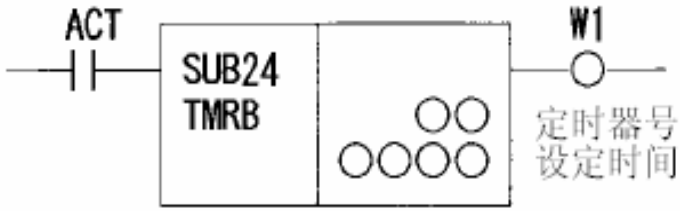
ACT=0:断开时间继电器。
=1:起动定时器。
W1 =1:ATC接通后经过设定的时间时，
输出即接通。

◀‖使用例‖▶



• 在X000.0接通后再经过4800ms，R000.0就接通。

(5)
固定定时器



定时器设定时间是固定的延时时间。

用功能命令的参数指定时间。

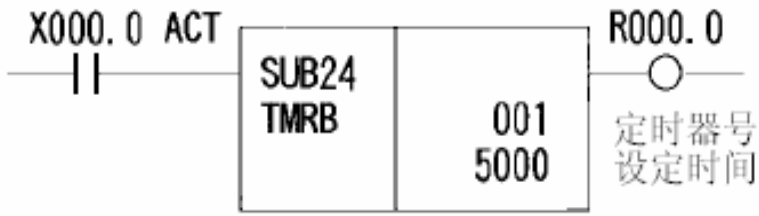
ACT=0:断开时间继电器。
=1:起动定时器。

W1 =1:在ACT接通后经过设定的时间时，输出即接通。

〔定时器号〕1～100

〔设定时间〕用ms单位的10进数设定时间。(最大262136)

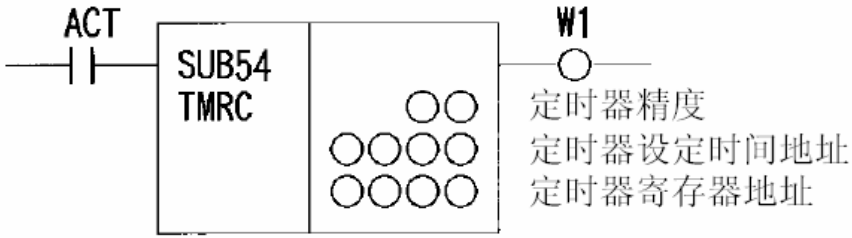
◀‖使用例‖▶



• 在X000.0接通后经过5秒，R000.0即接通的定时器。

(6)
可变定时器

利用数据表等设定定时器时间。



〔定时器精度〕　0:8ms　　1:48ms

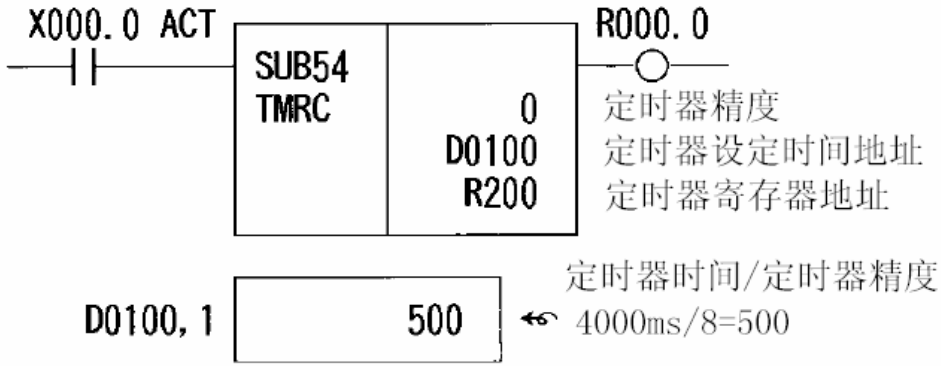
〔定时器设定时间地址〕

需要2字节的存储器，并且以定时器精度为单位用二进制形式进行设定。(定时器设定画面显示为10进数显示)

〔定时器寄存器地址〕

系统使用的作业区域需要4字节。

◀使用例▶

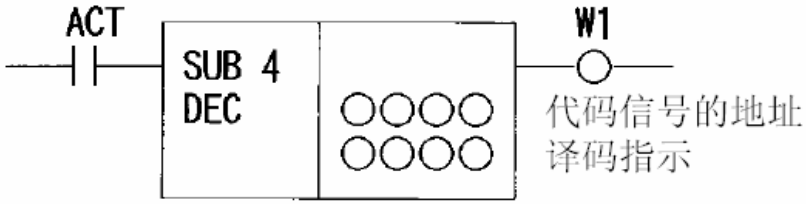


• 在X000.0接通后经过4秒，R000.0即接通的定时器。

(7)

译码处理

对2位的BCD码进行译码，当与指示的值相同时，W1接通，如不一致，则W1断开。

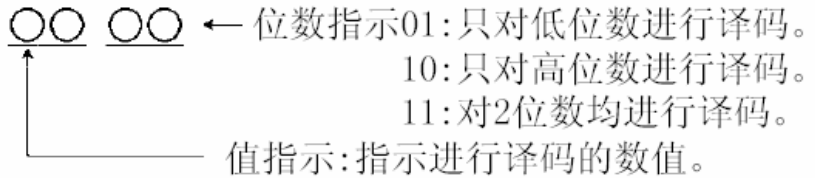


ACT=1:进行译码。

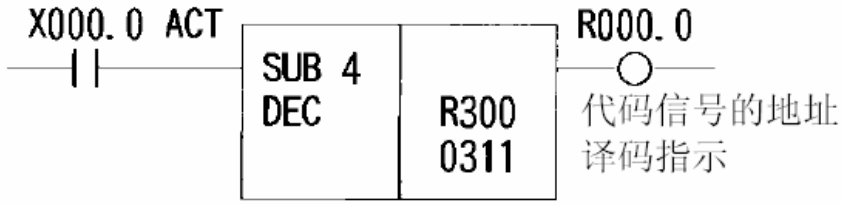
W1 =1:译码的结果已一致。

〔代码信号地址〕指定译码对象的地址。

〔译码指示〕



◀使用例▶

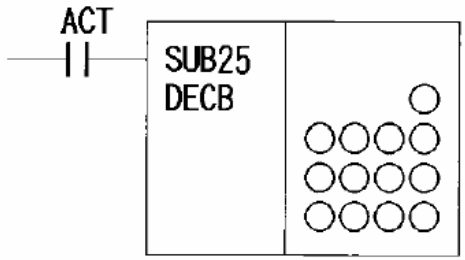


• 在X000.0接通时，如R0300为00000011(3)则R000.0接通。

(8)

二进制译码

对1、2、4字节长的二进制形式的代码数据进行译码。代码数据一致时，对应的位即为“1”，如不一致则为“0”。



〔形式指定〕代码数据的形式为

1:1字节长　2:2字节长　4:4字节长

〔代码信号的地址〕

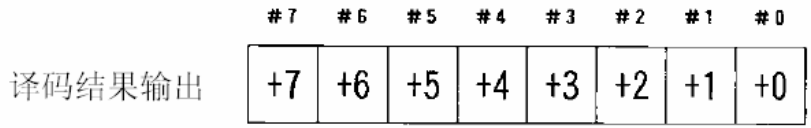
指定进行译码的数据的起始地址。

〔译码指示〕

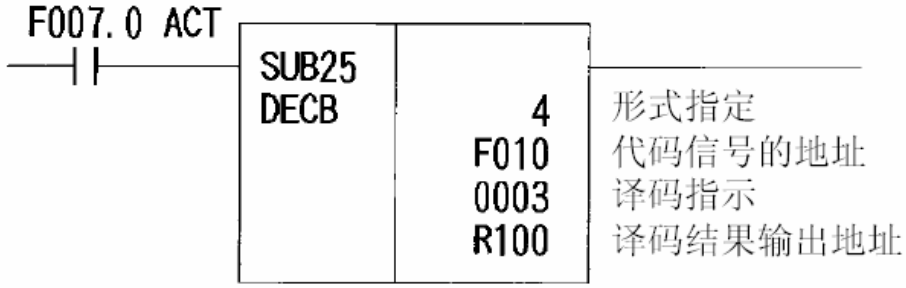
8个译出代码号的第一个号。

〔译码结果输出地址〕

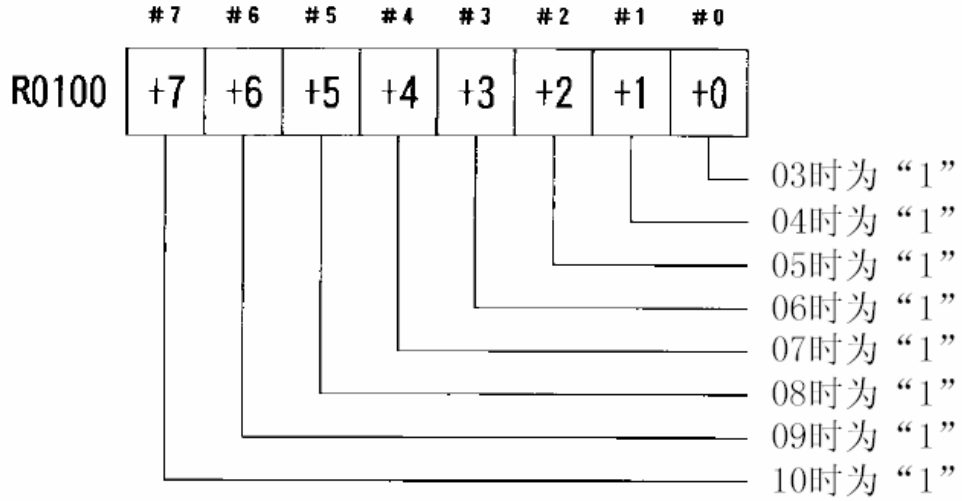
由译码指示指定号的译码结果被输到位0，号+1的译码结果被输到位1，号+7的译码结果被输到位7。



◀使用例▶

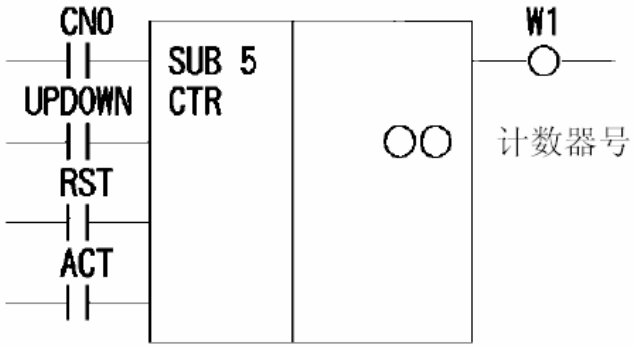


• F007.0接通后，对F0010～F0013的4字节进行译码，当译出结果在3～10的范围内时，与R0100对应的位变为“1”。



(9)
计数器

是进行加/减计数的环形计数器。
计数器的形式(二进制/BCD)用系统参数(SYSPRM)进行设定。



CNO =0: 计数器的初始值为0。
=1: 计数器的初始值为1。

UPDOWN =0: 是加计数。(初始值为CNO地设定)
=1: 是减计数。(初始值为计数器预置值)

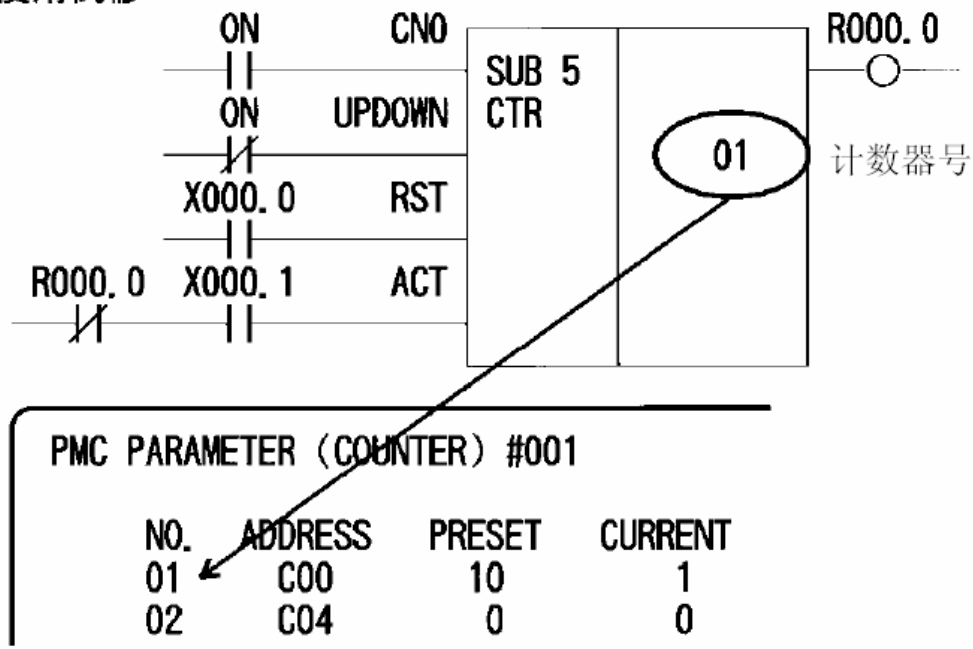
RST =1: 将计数器复位。
累计值被复位，加计数时，根据CNO的设定变为0或1，减计数时变为计数器预置值。

ACT =1: 取0到1的前沿进行计数。

W1 =1: 是计数结束输出。加计数时为最大值，减计数最小值时为1。

〔计数器号〕RB4, RB6/RC4为1～50
其它为1～20

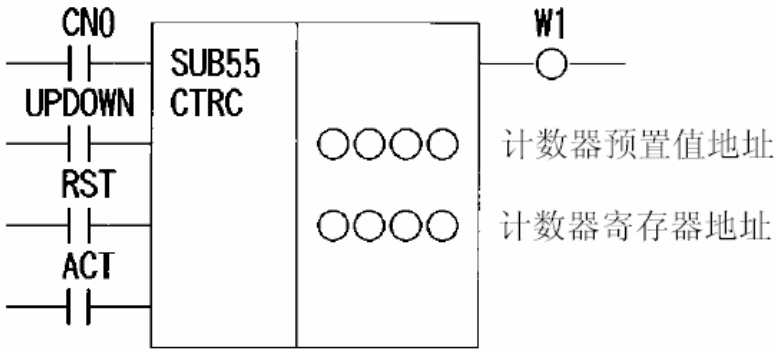
◀使用例▶



- “ON”是常“1”信号。
- 从1～10进行计数的计数器。
- 当X000.0为“1”时，计数器即复位到“1”。
- 在X000.1的上升沿，计数器加1。
- 计数器的值达最大值10时，R000.0变为“1”。
- 若控制条件ACT的 R000.0 断开时，其后X000.1即使从“0”变为“1”，计数器也不动作。

(10)
计数器

是进行加/减计数的二进制形式的环形计数器。



CN0 =0：计数器的初始值为0。
=1：计数器的初始值为1。

UPDOWN =0：是加计数。（初始值为CN0的设定）
=1：是减计数。（初始值为计数器预置值）

RST =1：将计数器复位
累计值被复位，加计数时，根据CN0的设定变为0或1，减计数时变为计数器预置值。

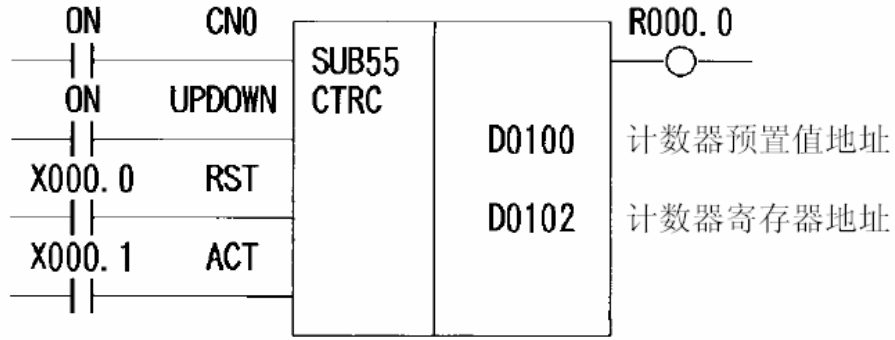
ACT =1：取0～1上升沿进行计数。

W1 =1：是计数结束输出。加计数时为最大值，减计数最小值时为1。

〔计数器预置值地址〕指定2字节的存储器的起始地址。

〔计数器寄存器地址〕指定连续的4字节的存储器的起始地址。头2字节为累计值，后2字节为系统的工作区。

◀‖使用例‖▶

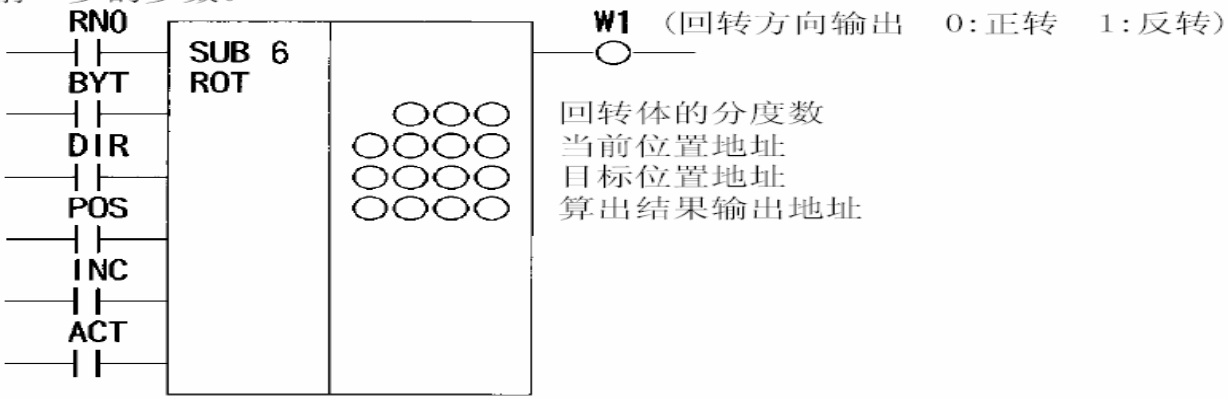


D0100,1	10
D0102,3	1
D0104,5	不定

- “ON”为常“1”信号。
- 从1～10进行计数的计数器。
- 当X000.0为1时，计数器(D0102.3)即变为“1”。
- 在X000.1的信号的前沿，计数器为+1。
- 计数器的值达最大值时，R000.0变为“1”。
- 在R000.0=1时，输入X000.1，即回到最小值1。

(11)
回转控制

判别回转体的下一步回转方向，计算出进行回转的步数，或计算到达目标位置前一步的步数。



RNO=0: 回转体的位置号是从0开始的连续号。
=1: 回转体的位置号是从1开始的连续号。

BYT=0: 回转体的位置号是BCD2位(1字节)的数据。
=1: 回转体的位置号是BCD4位(2字节)的数据。

DIR=0: 不判别下一步回转方向。(始终正转)
=1: 判别下一步回转方向。(方向输出到W1)

POS=0: 计算到达目标位置的步数。
=1: 计算到达目标位置前一步的步数。

INC=0: 计算目标位置的号。
=1: 计算到达目标位置的步数。

ACT=1: 执行ROT命令。

W1 =0: 回转方向为正转。
=1: 回转方向为反转。

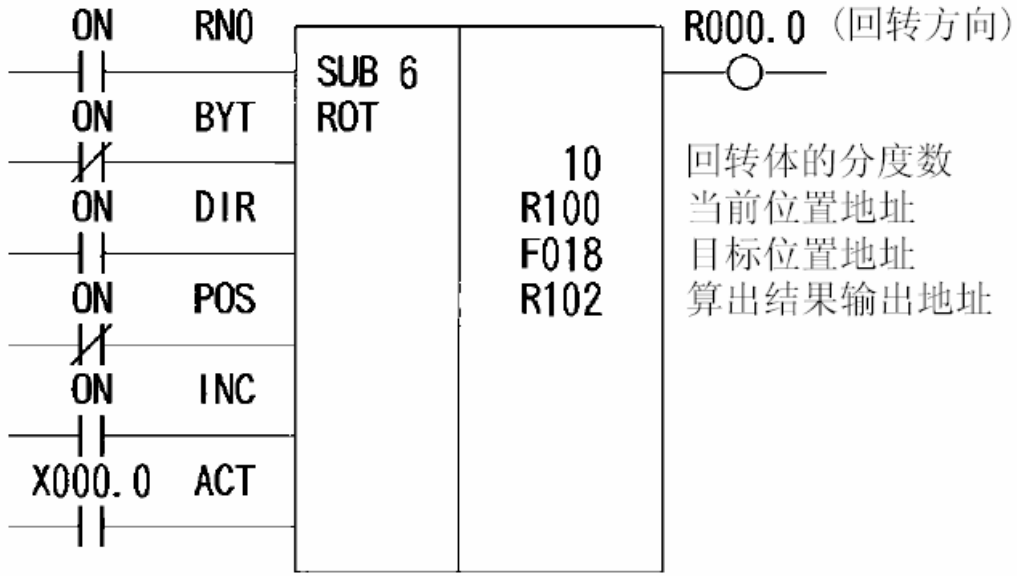
〔回转体分度数〕 设定回转体转位的数目。

〔当前位置地址〕 存储回转体当前步号的起始地址。

〔目标位置地址〕 存储目标位置的起始地址。

〔算出结果输出地址〕 算出的步数的输出地址。

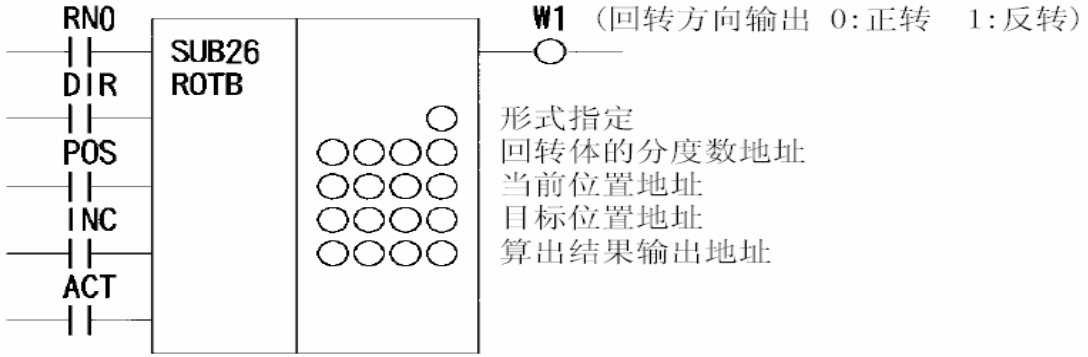
《使用例》



- “ON”始终为“1”的信号。
- X000.0接通时，计算分度数为10的回转体从R100的当前位置到F018的步数，并把结果写入R102。
- 此时的回转方向被输出到R000.0。

(12)
二进制
回转控制

可用地址指定回转体的分度数。另外，进行处理的数值都为二进制形式。其他功能与ROT命令相同。



RNO=0: 回转体的位置号是从0开始的连续号。
=1: 回转体的位置号是从1开始的连续号。

DIR=0: 不判别下一步回转方向。(始终正转)
=1: 判别下一步回转方向。(方向输出到W1)

POS=0: 计算到达目标位置的步数。
=1: 计算到达目标位置前一步的步数。

INC=0: 计算目标位置的号。
=1: 计算到达目标位置的步数。

ACT=1: 执行ROT命令。

W1 =0: 回转方向为正转。
=1: 回转方向为反转。

〔形式指定〕1:1字节长 2:2字节长 4:4字节长

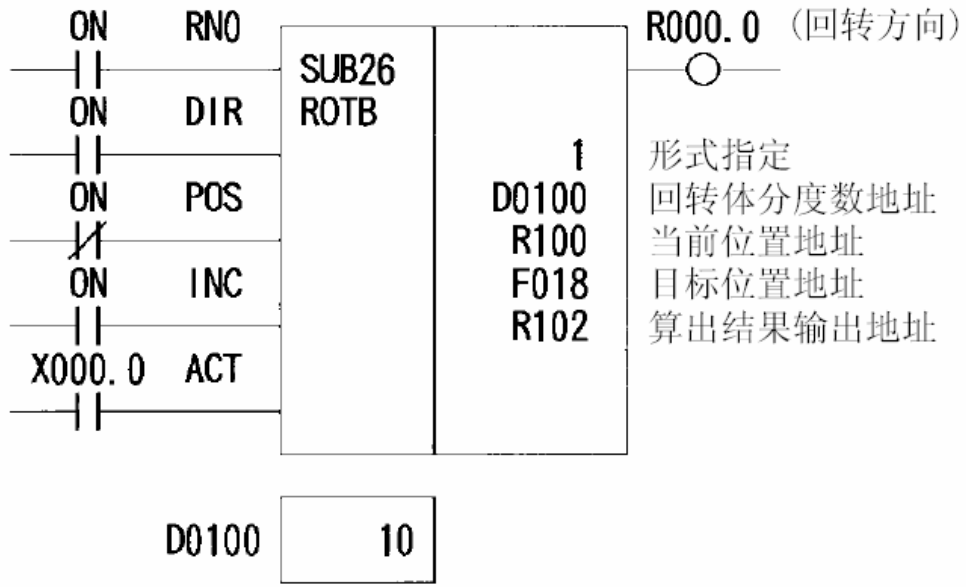
〔回转体分度数〕设定回转体转位的数目。

〔当前位置地址〕存储回转体当前步号的起始地址。

〔目标位置地址〕存储目标位置的起始地址。

〔算出结果输出地址〕算出的步数的输出地址。

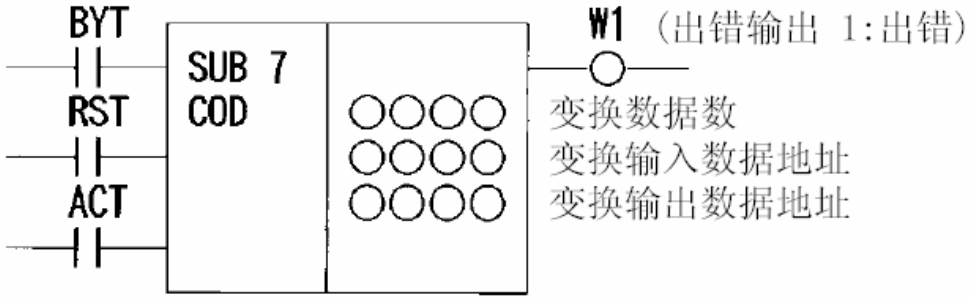
◀使用例▶



- “ON”始终为“1”的信号。
- X000.0接通时，计算分度数目10的回转体从R100的当前位置到F018的步数，并把结果写入R102。
- 此时的回转方向被输出到R000.0。

(13)
代码变换

用2位的BCD码指定变换数据表内的号，将与输出的表内号对应的2位或4位BCD码输出。



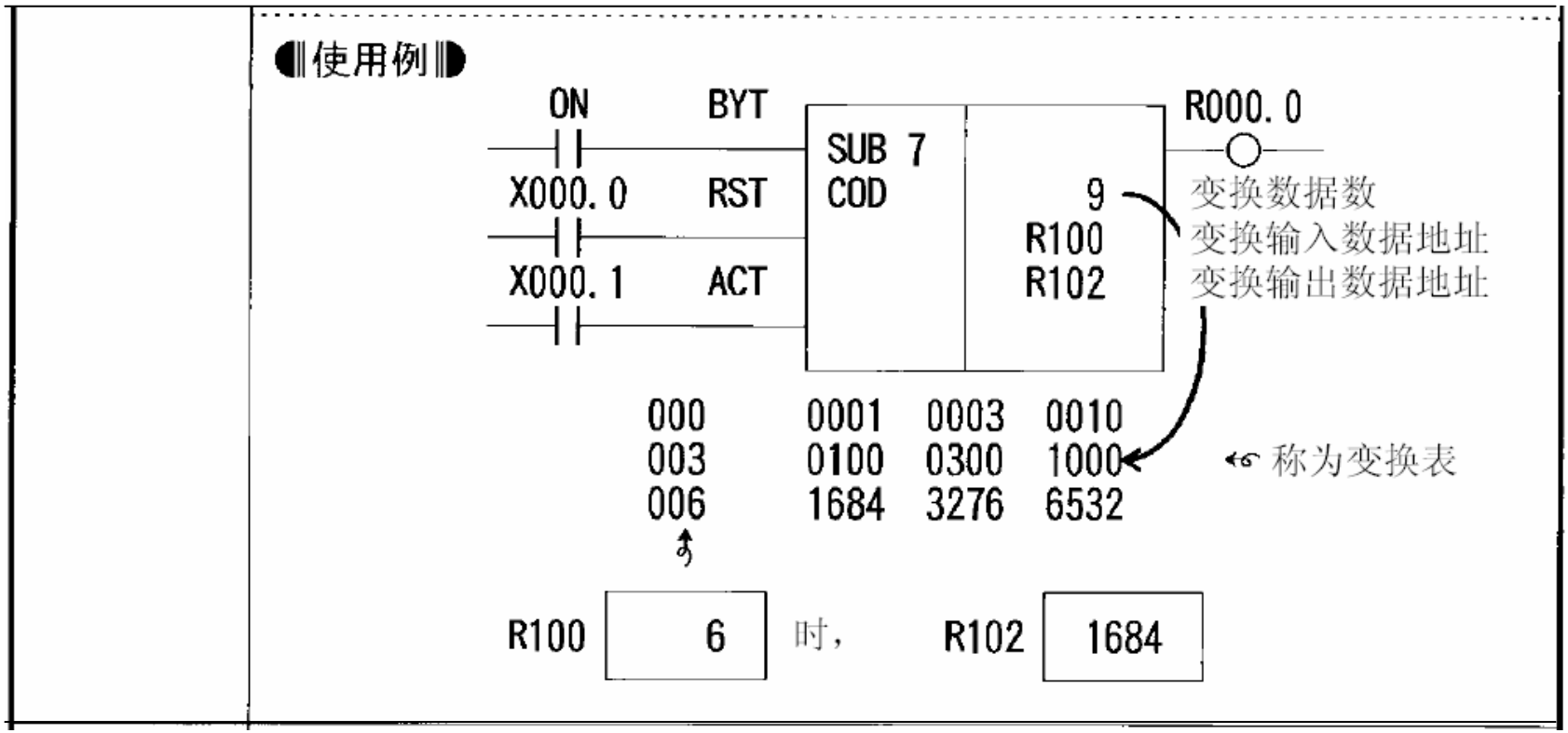
BYT=0: 变换数据表的数据为BCD2 位。
=1: 变换数据表的数据为BCD4 位。

RST=1: 把错误输出W1复位。

ACT=1: 执行COD命令。

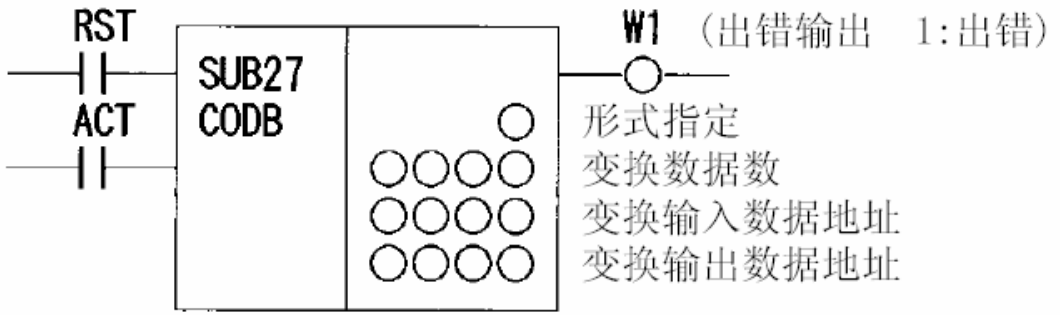
W1 =1: 变换输入号超过了变换数据数，指令出错。

〔变换输入数据地址〕指定表内号的地址(1字节)。
〔变换输出数据地址〕变换结果的存储地址。


使用例
ON
BYT
X000.0
RST
X000.1
ACT
SUB 7
COD
9
R100
R102
R000.0
变换数据数
变换输入数据地址
变换输出数据地址
000
003
006
0001
0100
1684
0003
0300
3276
0010
1000
6532
称为变换表
R100
6
时，
R102
1684

(14)
二进制码
变换

用2位的二进制码指定变换数据表内的号，将与输入的表内号对应的1、2、4字节的数值输出。



RST=1: 把错误输出W1复位。

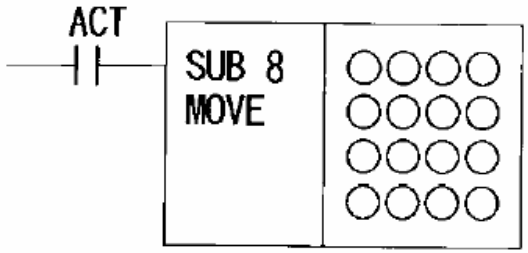
ACT=1: 执行COD命令。

W1 =1: 变换输入号超过了变换数据数，指令出错。

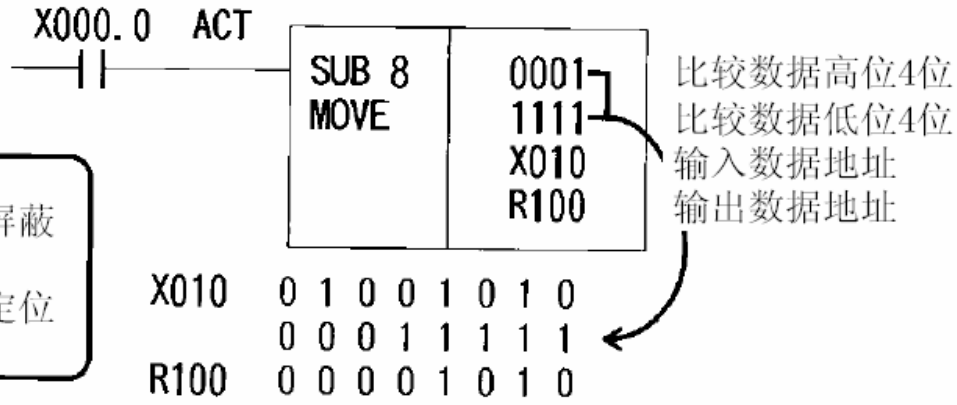
〔形式指定〕1:1字节长　2:2字节长　4:4字节长

(15)
逻辑乘后
数据传送

数据传送据地址指定的1字节的数据与比较数据进行逻辑乘(AND)，并把结果写入输出数据地址。

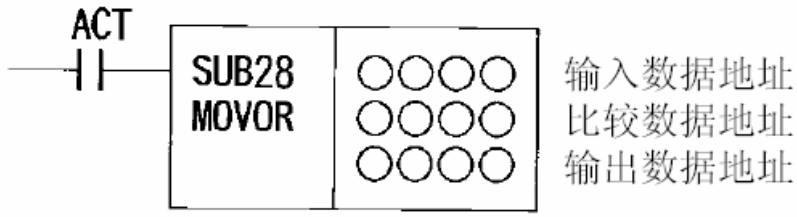


◀使用例▶

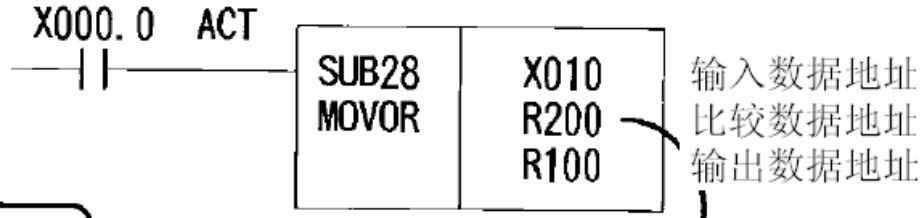


(16)
逻辑和后
数据传送

用输入数据地址指定的1字节的数据与比较数据进行逻辑和(OR)，并把结果写入输出数据地址。



◀使用例▶

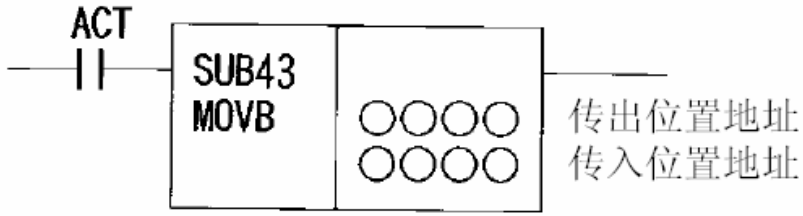


· 接通指定位

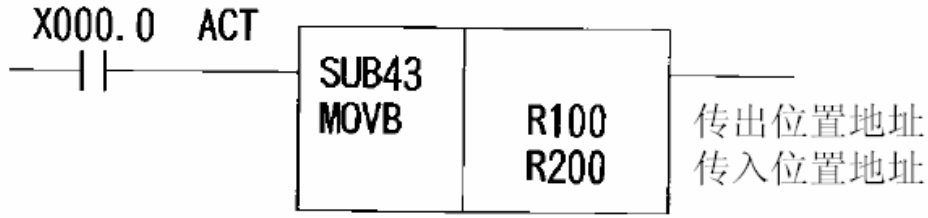
X010	0	1	0	0	1	0	1	0
R200	0	0	0	1	1	1	1	1
R100	0	1	0	1	1	1	1	1

(17)
传送1字节数据

把1字节的数据从被指令的传出位置地址传送到传入位置地址。



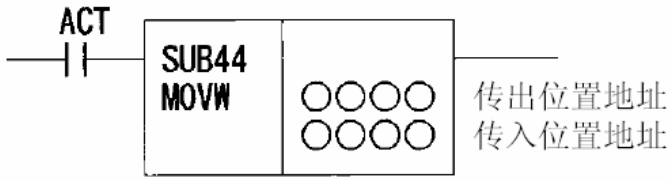
◀使用例▶



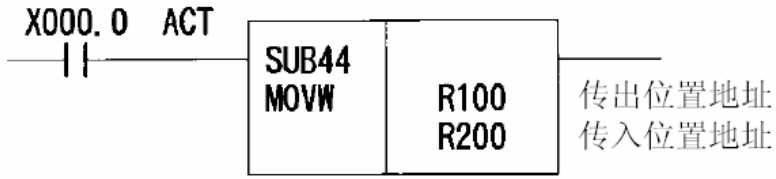
• 把R100的值传送到R200。

(18) 传送2字节数据

把2字节的数据从被指令的传出位置地址传送到传入位置地址。



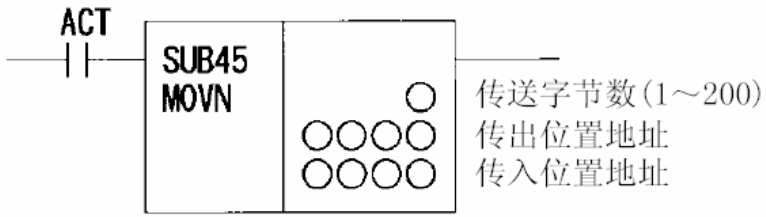
使用例



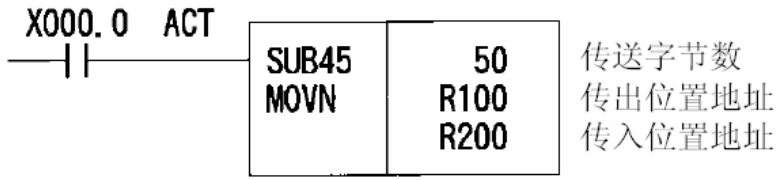
• 把R100和R101的值传送到R200和R201。

(19) 传送任意字节数据

把任意字节的数据从被指定的传出位置地址传送到传入位置地址。



使用例

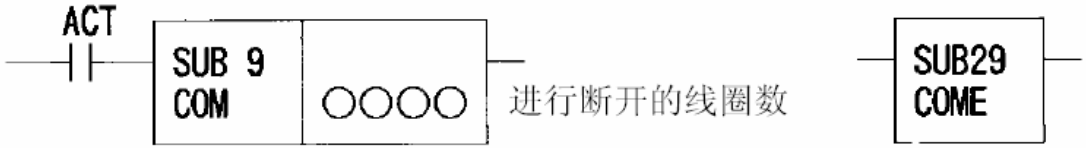


• 把R100～R149的值传送到R200～R249。

(20, 21)

公用线控制

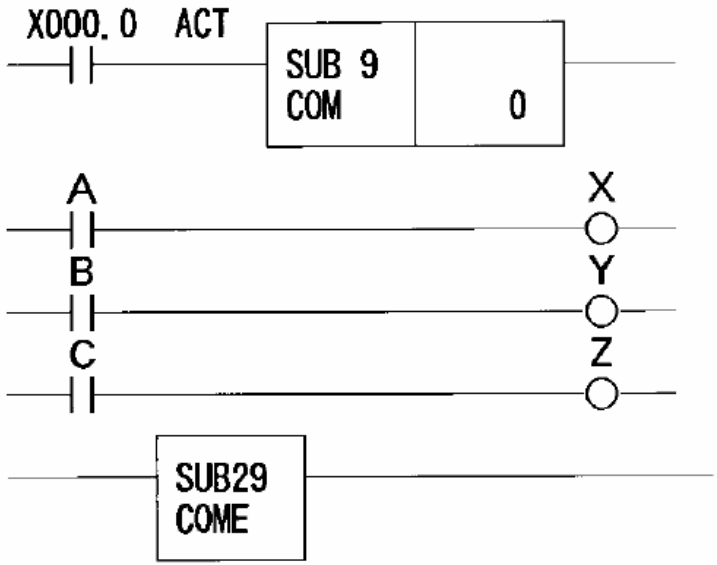
断开COME命令前的区间线圈。



ACT=0: 无条件断开COME之前的线圈。
=1: 什么都不动。

〔进行断开的线圈数〕RB/RC以外的PMC型号必须指定“0”。

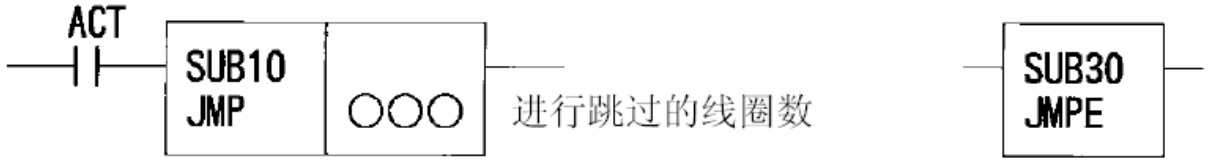
使用例



• X000.0为“0”时，信号X, Y, Z将无条件地变为“0”。

(22, 23)
跳转

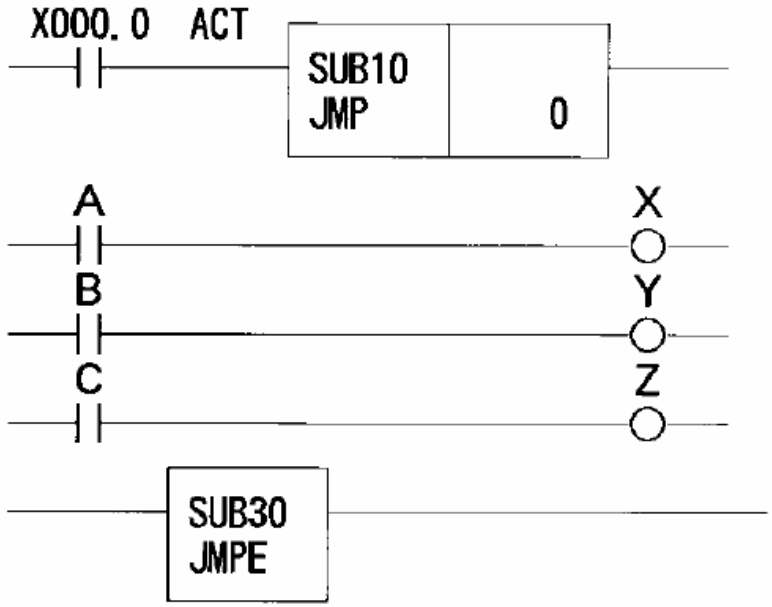
跳过JMPE命令前的区间。



ACT=0: 不跳转。执行下面的命令。
=1: 跳过指定区间。

〔进行跳过的线圈数〕RB/RC以外的PMC型号必须指定“0”。

◀使用例▶

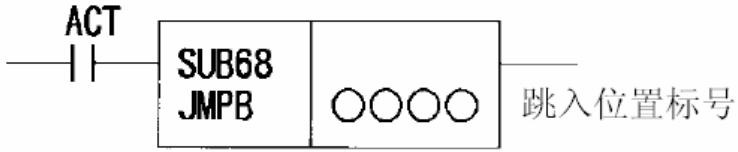


• X000.0为“1”时，信号X,Y,Z不变化。

(24)

标号跳转1

转移到被指定标号。



ACT=0: 不跳转。执行下面的命令。
=1: 跳转到被指定的标号。

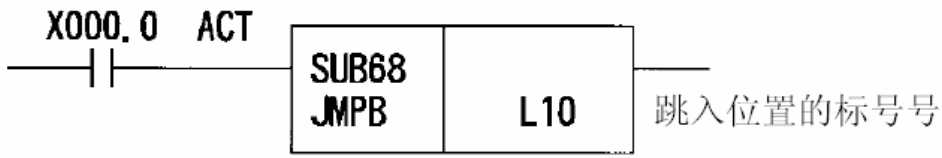
〔跳入位置的标号号〕L1～L9999

指定不跳过主程序、子程序等的程序单位的范围的标号。

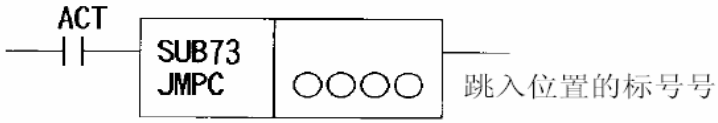
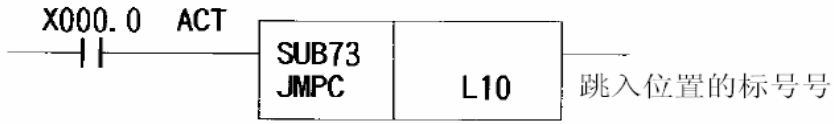
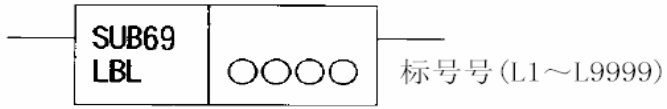
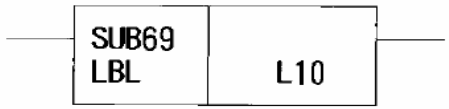
对1个标号也可指定多个JMPB命令。
另外，也可把跳转命令作成嵌套。

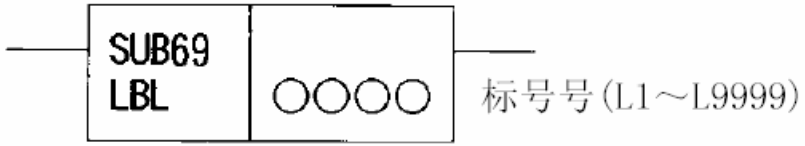
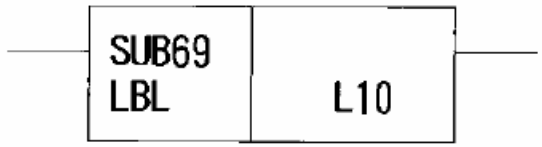
☞ 使用该命令向前跳转时，请注意不要造成无限循环。

◀使用例▶



• X000.0为“1”时，转移到标号10。

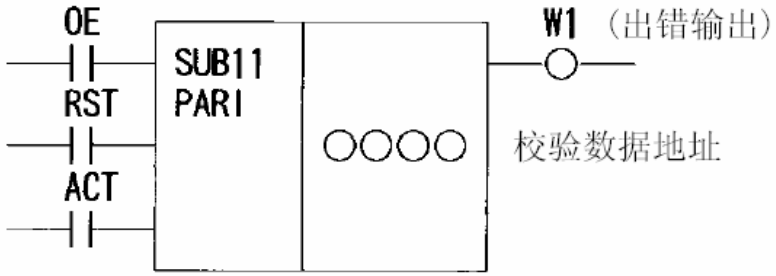
(25) 标号跳转2	转移到被主程序指定的标号。  **ACT=0**：不跳转。执行下面的命令。 **=1**：跳转到被指定的标号。 〔跳入位置的标号号〕L1～L9999 指定主程序上的标号号。 对1个标号也可指定多个JMPC命令。 ☞ 使用该命令进行跳转时，请注意不要造成无限循环。 **◀使用例▶**  • X000.0为“1”时，转移到主程序的标号10。
(26) 标号	定义标号。  〔标号号〕L1～L9999 每个主程序、子程序可使用相同的标号号。 **◀使用例▶** 

(26) 标号	定义标号。  〔标号号〕L1～L9999 每个主程序、子程序可使用相同的标号号。 **使用例** 

(27)

奇偶校验

对被指定的地址进行奇偶校验，如不正常时，输出出错报警。



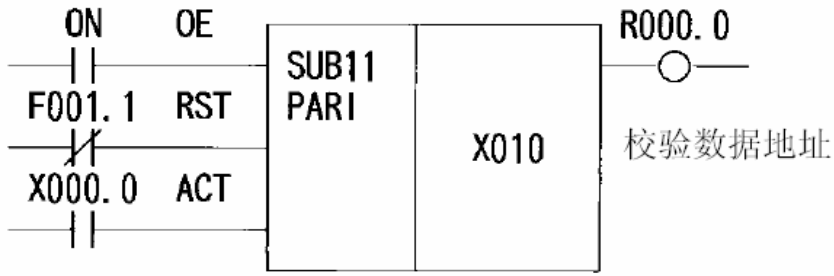
OE =0: 进行偶数校验。
=1: 进行奇数校验。

RST=1: 把出错输出W1复位。

ACT=1: 执行奇偶校验命令。

W1 =1: 在奇偶校验中发生错误时变为接通。

◀‖使用例‖▶



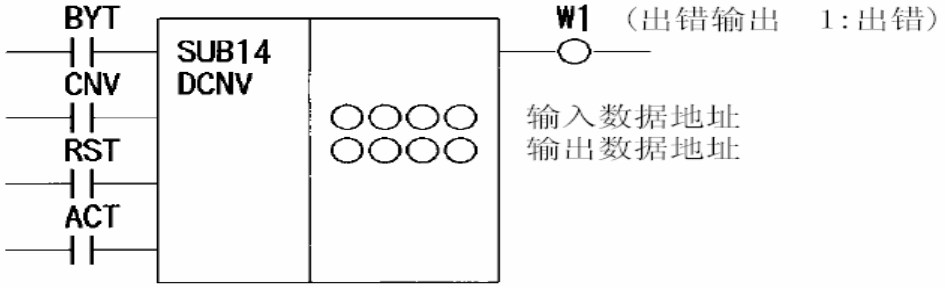
• X000.0接通时，在X010的位型数据中
“1”的数不是奇数时，出错输出R000.0即变为“1”。

X010 0 1 0 0 1 1 0 0 ⇨ R000.0=0
X010 0 1 0 1 0 0 0 0 ⇨ R000.0=1

• 信号F001.1闭合时出错输出R000.0即被复位。

(28)
数据变换

把1或2字节的数据从二进制码变换成BCD码，或从BCD码变换成二进制码。



BYT=0：变换1字节的数据。
　　=1：变换2字节的数据。

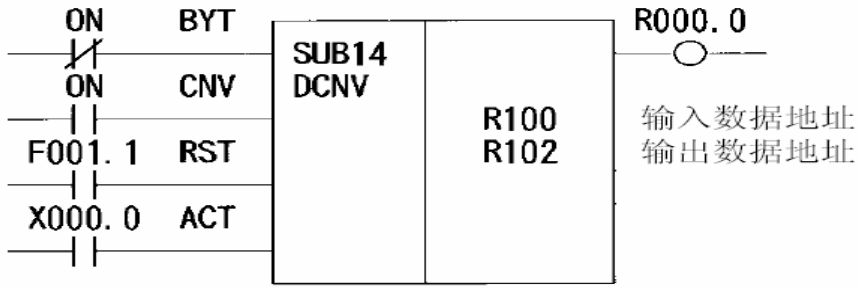
CNV=0：从二进制码变换成BCD码。
　　=1：从BCD码变换成二进制码。

RST=1：把出错输出W1复位。

ACT=1：执行数据变换命令。

W1　=1：输入数据应为BCD码的地方，如果已是二进制码，或从二进制码变换成BCD码时超过指定字节长即进行出错报警。

◀使用例▶

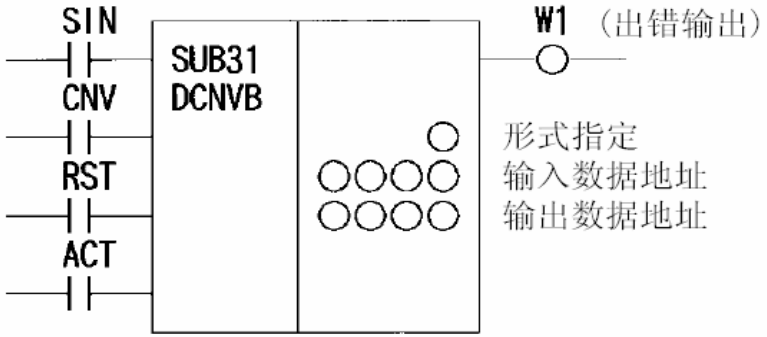


• 把设定在R100中的1字节的BCD码变换成二进制码后输出到R102。

R100 | 00010010(12) | 时，为 R102 | 00001100

(29)
扩展
数据变换

把1、2、4字节的二进制码变换成BCD码，或将BCD码变换成二进制码。



SIN=0：输入的BCD码的符号为正。
=1：输入的BCD码的符号为负。

CNV=0：从二进制码变换成BCD码。
=1：从BCD码变换成二进制码。

RST=1：把出错输出W1复位。

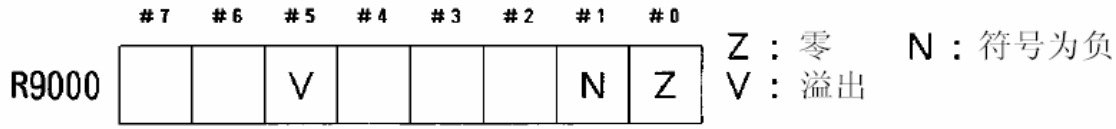
ACT=1：执行数据变换命令。

W1 =1：输入数据应为BCD码的地方，如果已是二进制码，或从二进制码变换成BCD码时，超过指定字节长即进行出错报警。

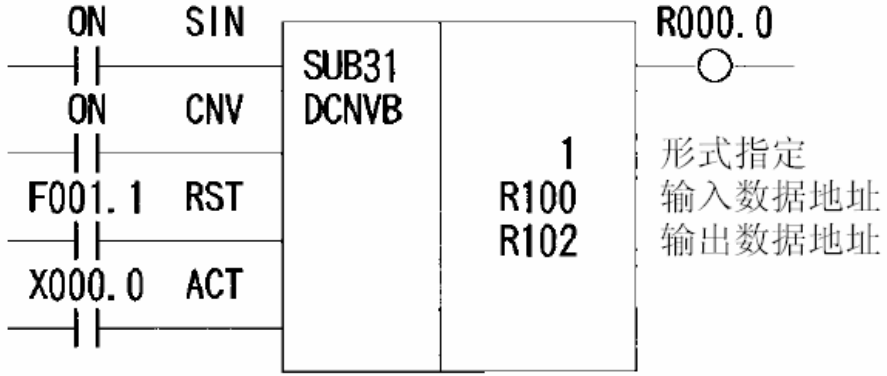
〔形式指定〕1:1字节　2:2字节　4:4字节

〔运算输出寄存器〕

从二进制码变换成BCD码后的符号，把结果存在运算输出寄存器中。



◀使用例▶

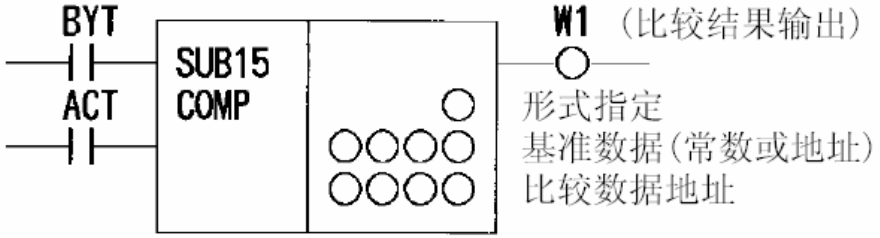


・把设定在R100中的1字节的BCD码变换成二进制码后，输出到R102。

R100 | 00010010(12) | 时，为 R102 | 11110100(-12)

(30)
大小比较

比较2位或4位BCD的数值，把比较结果输出到W1。

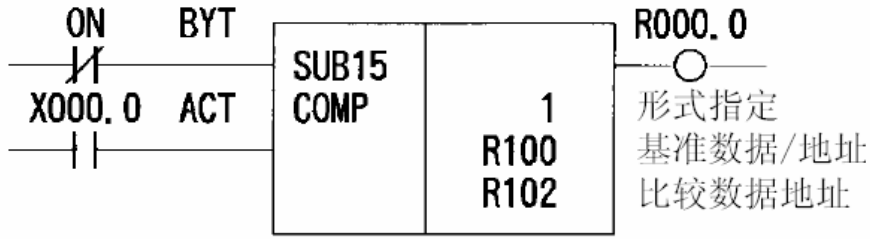


BYT=0: 比较BCD码2位。
=1: 比较BCD码4位。

W1 =0: 基准数据＞比较数据
=1: 基准数据≤比较数据

〔基准数据形式指定〕
0:基准数据为常数　　1:基准数据为指定地址

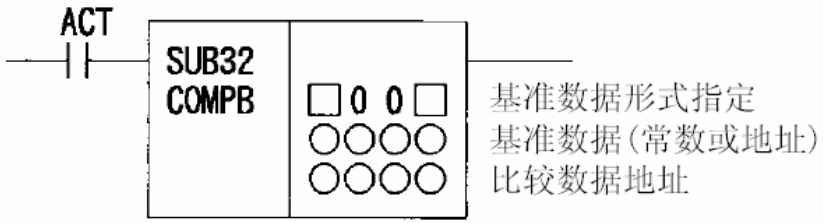
使用例



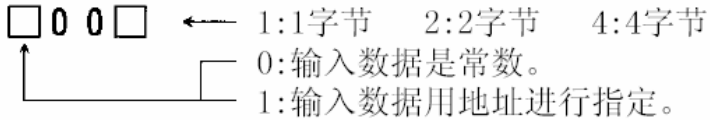
• X000.0接通时，比较R100和R102的值，R100≤R102时，R000.0就接通。

(31)
二进制
大小比较

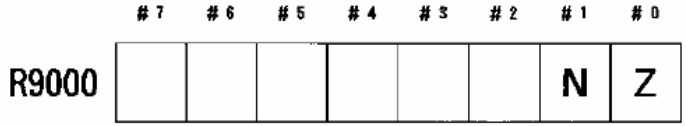
对1、2、4字节的二进制形式数据进行比较。
比较结果输出到运算输出寄存器(R9000)。



〔基准数据形式指定〕

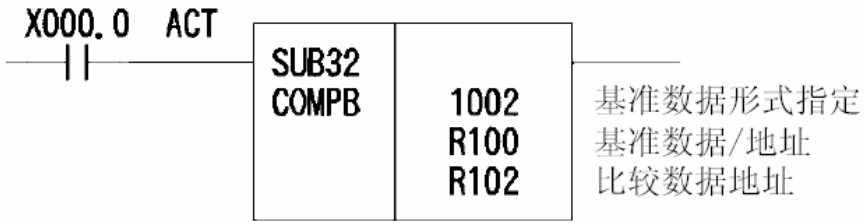


〔比较输出寄存器〕



Z：基准数据＝比较数据
N：基准数据＜比较数据

◀使用例▶

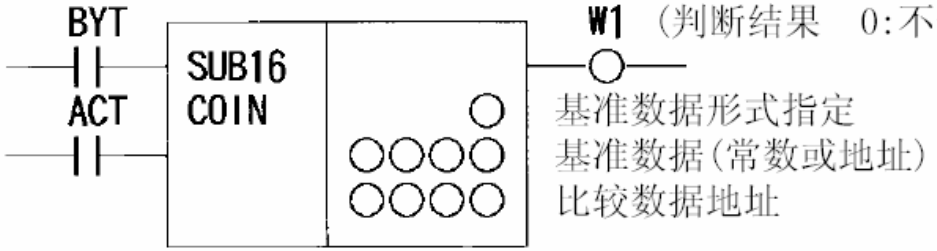


• X000.0接通时，对R100、R101和R102、R103的2字节的值进行比较。

值一致时，R9000.0=1；R100、R101比R102、R103小时，R9000.1=1。

(32)
一致性判断

比较BCD形式的数据，判断是否相同。

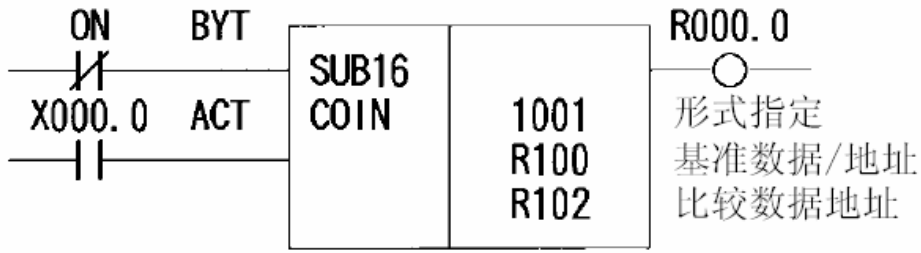


BYT=0：比较BCD码2位。
=1：比较BCD码4位。

W1 =0：基准数据≠比较数据
=1：基准数据＝比较数据

〔基准数据形式设定〕
0:基准数据为常数　1:基准数据为指定地址

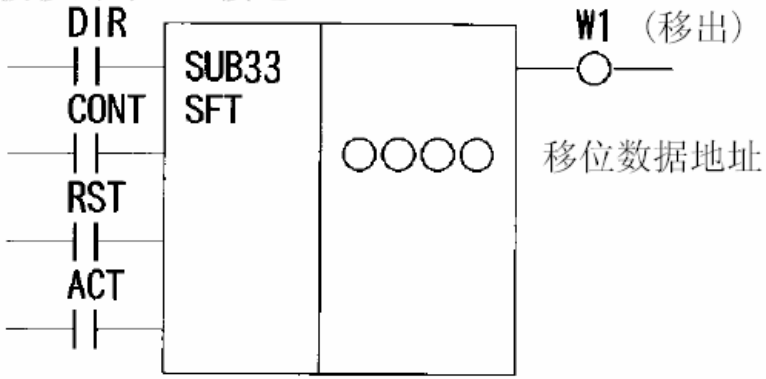
◀‖使用例‖▶



• X000.0接通时，比较R100和R102的值，R100＝R102时，R000.0即接通。

(33)
移位
寄存器

把连续的2字节的数据向右或左移动1位。
被移出时，W1接通。

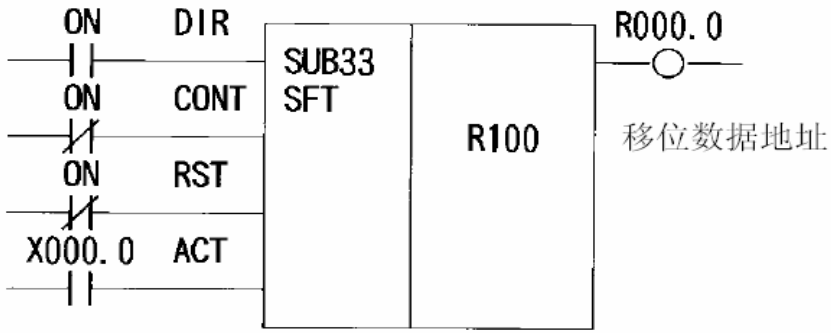


DIR =0：把数据向左移位。
=1：把数据向右移位。

CONT =0：移入0。
=1：原来的位为1时，保留原来的1。

RST =1：断开移出W1。

◀使用例▶



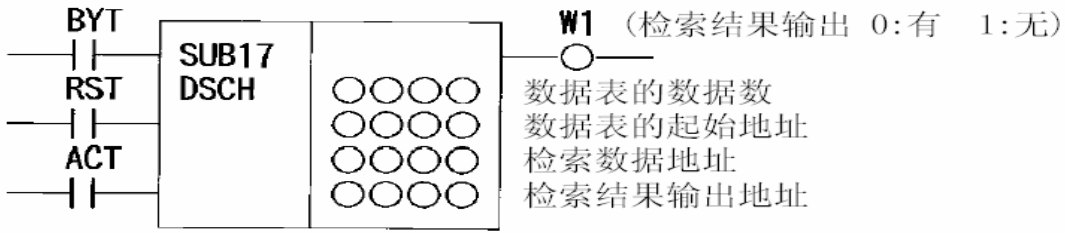
- X000.0为“1”时，把R100.1的值向右移1位。
- 进行移位前如果R100.0＝1，移出的R000.0即变为“1”。

R101	R100	W1
10100000	11001000	0
01010000	01100100	0
00101000	00110010	0
00010100	00011001	0
00001010	00001100	1
00000101	00000110	0
00000010	10000011	0
00000001	01000001	1
00000000	10100000	1
00000000	01010000	0
00000000	00101000	0
00000000	00010100	0
00000000	00001010	0
00000000	00000101	0
00000000	00000010	1
00000000	00000001	0
00000000	00000000	1
00000000	00000000	0

移位的顺序

(34)
数据检索

检索指定的数据是否存在于数据表内，并输出表内号数。

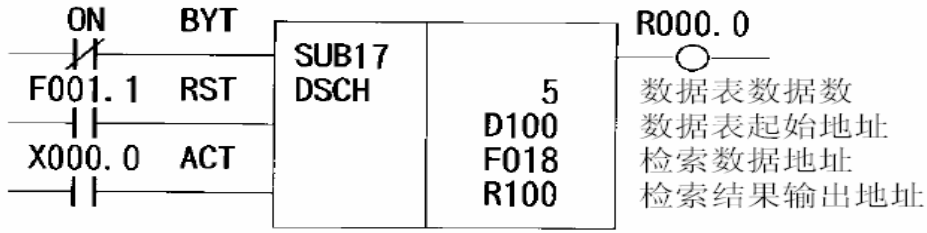


BYT=0：检索BCD码2位。
=1：检索BCD码4位。

RST=1：断开无检索数据的输出W1。

W1 =1：无检索的数据时，输出即接通。

使用例



- X000.0接通时，从D100开始在长度为5个单元的数据表中，依次检索F018中存储的值，并把检索到的数据的表内号写入R100。

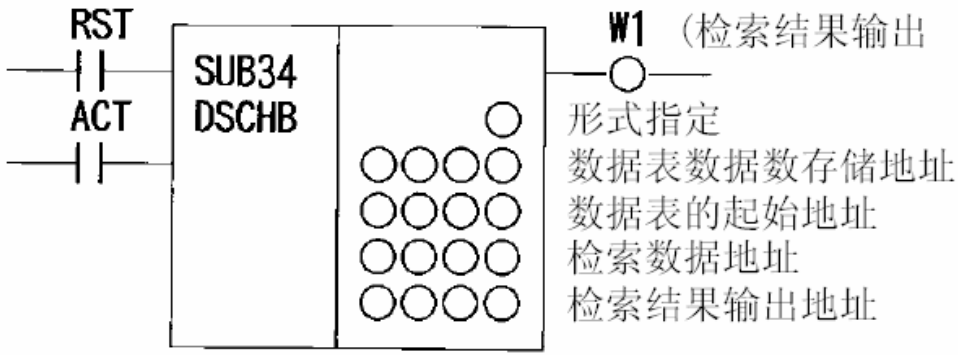
D100(+0)	10
D101(+1)	15
D102(+2)	17
D103(+3)	1
D104(+4)	8

F018 [17] 时，为 R100 [2]

- 没有检索的值时，出错输出R000.0将接通。
- 接通F001.1时，出错输出R000.0即被断开。

(35)
二进制
数据检索

与DSCH命令的不同点是进行处理的数值必须是二进制形式，而且为了能使用地址指定数据表的数据个数，即使在ROM制作完成后，仍可调整表的容量。



RST=1: 断开无检索数据输出W1。

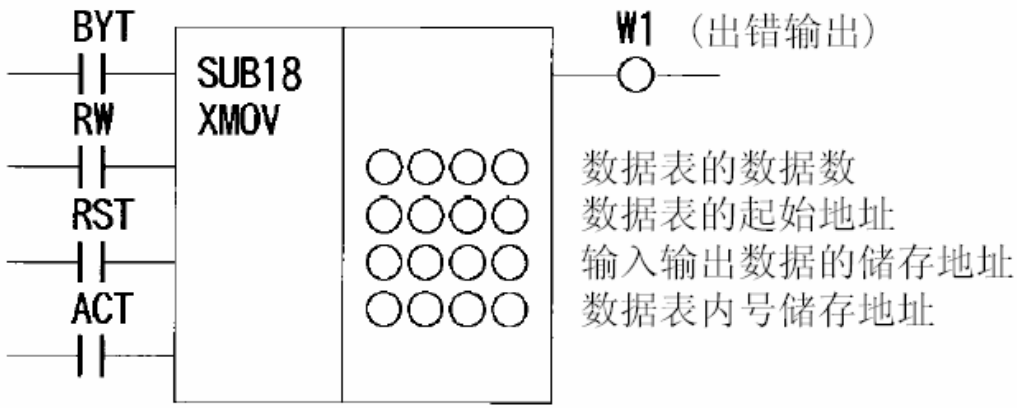
W1 =1: 没有检索数据时，接通输出。

〔形式指定〕1:1字节　2:2字节　4:4字节

使用例 请看DSCH命令。

(36)
变址修改
数据传送

读取或写入数据表内指定号的数据。进行处理的数值为BCD2位或BCD4位。



BYT=0：数据表的数据为BCD 2位。
=1：数据表的数据为BCD 4位。

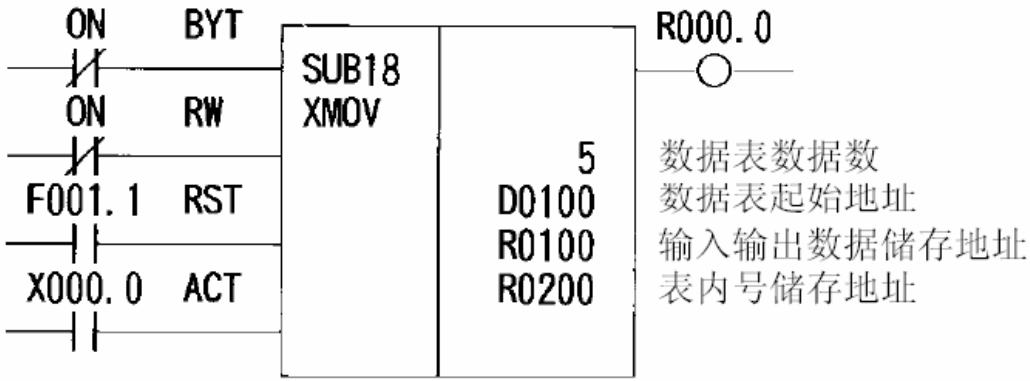
RW =0：从数据表读取数据。
=1：把数据写入数据表。

RST=1：断开出错输出W1。

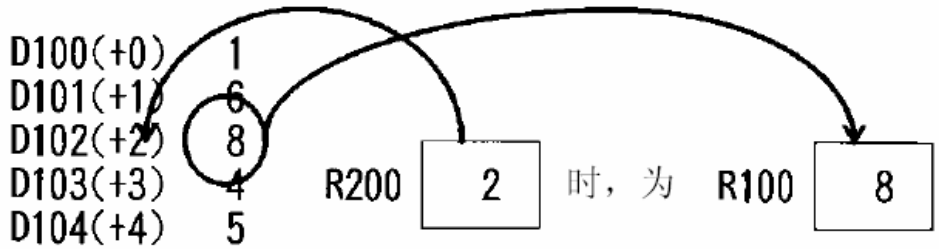
ACT=1：执行XMOV命令。

W1 =1：被指定的表内号超过数据表的数据数时，即出错报警。

◀使用例▶



• 数据表为D100开始的5个数据，读取由R200指定的表内号的数据，并写入R100。



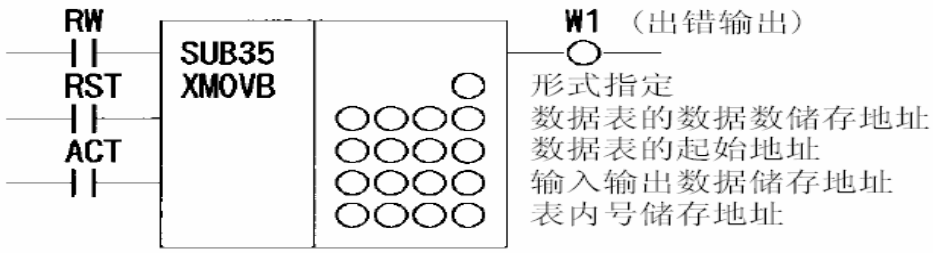
• 用R200指定的表号不正确时，出错输出的R000.0变为“1”。

• 使出错复位的F001.1为“1”时，出错输出的R000.0即变为“0”。

(37)
二进制
变址修改
数据传送

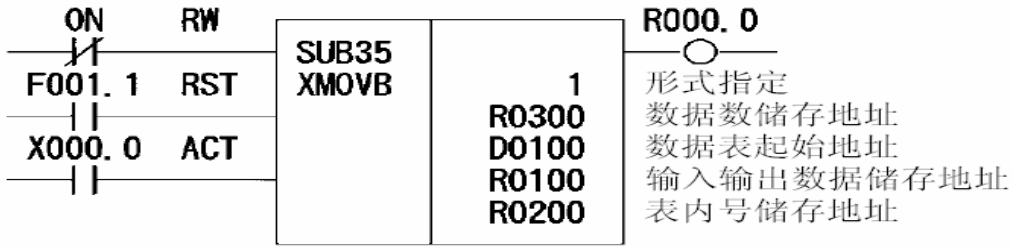
读取或写入数据表内指定号的数据。进行处理的数据为二进制形式。

另外，因为表容量是用地址指定的，所以在写入ROM后，还能修改表容量。

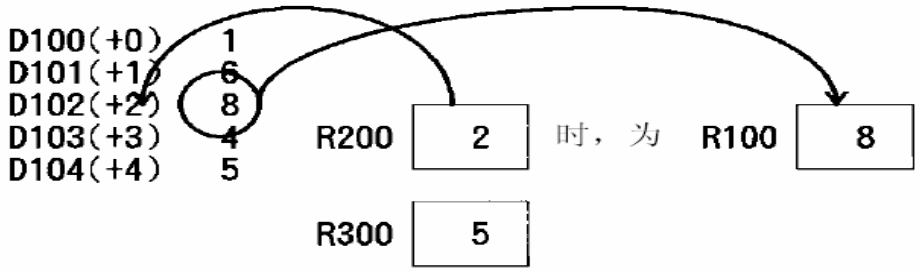


〔形式指定〕1:1字节　2:2字节　4:4字节

◀‖使用例‖▶



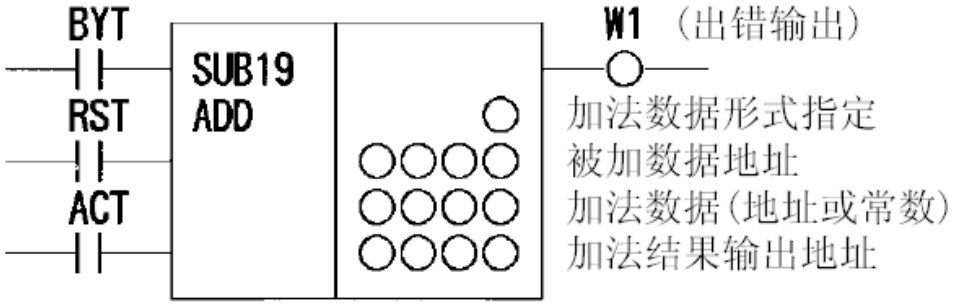
- 数据表为D200起始的5个数据，读取由R200指定的表内号的数据，并写入R100。



- 用R200指定的表号不正确时，出错输出的R000.0变为“1”。
- 出错复位的F001.1为“1”时，出错输出的R000.0即变为“0”。

(38)
加法

进行BCD 2位或4位的加法运算。



BYT=0: 进行处理的数值为BCD 2位。
=1: 进行处理的数值为BCD 4位。

RST=1: 把出错输出W1复位。

ACT=1: 执行ADD命令。

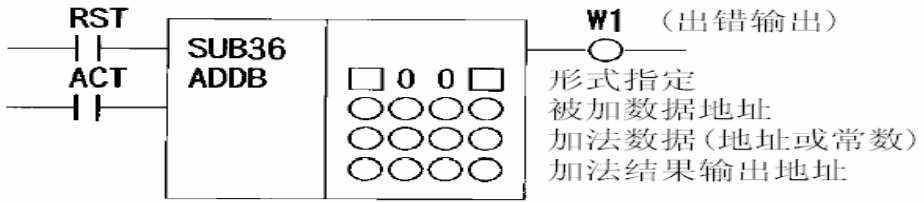
W1 =1: 加法结果超出指定的字节数时即接通。

〔加法数据形式指定〕 0:用常数指定加法数据。
1:用地址指定加法数据。

使用例 请看ADDB。

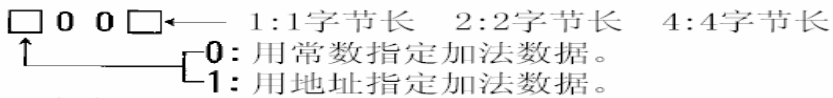
(39)
二进制
加法

进行1、2、4字节长的二进制形式的加法运算。



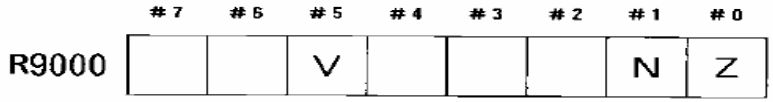
RST=1： 断开出错输出W1。
ACT=1： 执行ADDB命令。
W1 =1： 加法结果超出用形式指定的字节数时，即接通。

〔形式指定〕



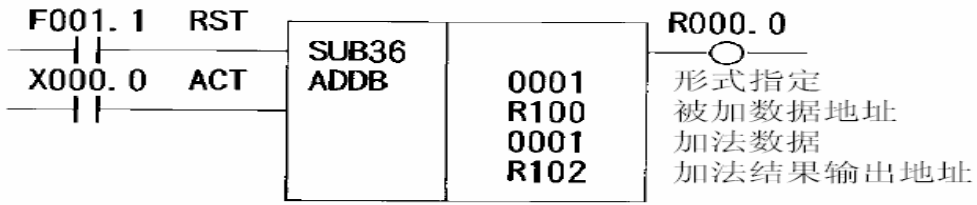
〔运算输出寄存器〕

二进制加法结果的状态输到运算输出寄存器。



V ： 溢出
Z ： 零
N ： 符号为负

使用例

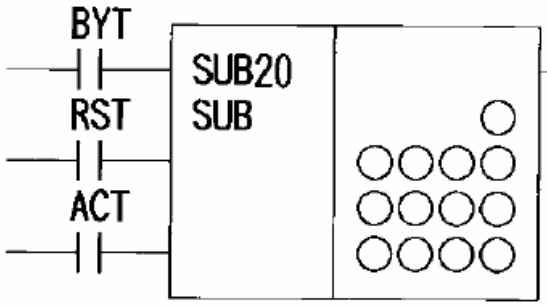


• 在R100上加1，结果写入R102。

R100 | 5 | 时，为 R102 | 6 |

(40)
减法

进行BCD 2位或4位的减法运算。



W1 （出错输出）

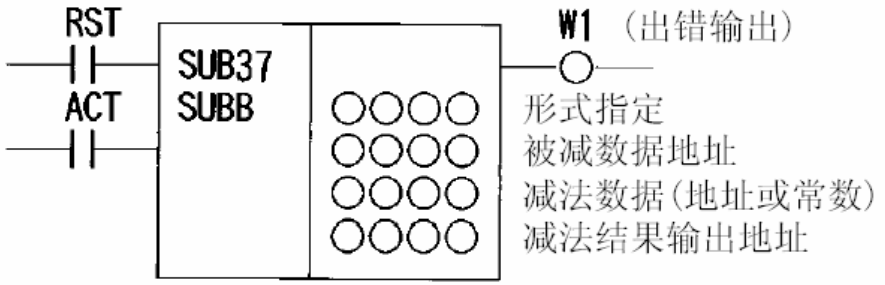
减法数据形式指定
减法数据地址
减法数据(地址或常数)
减法结果输出地址

W1 =1:减法结果为负时，接通输出。

◀使用例▶ 请看SUBB。

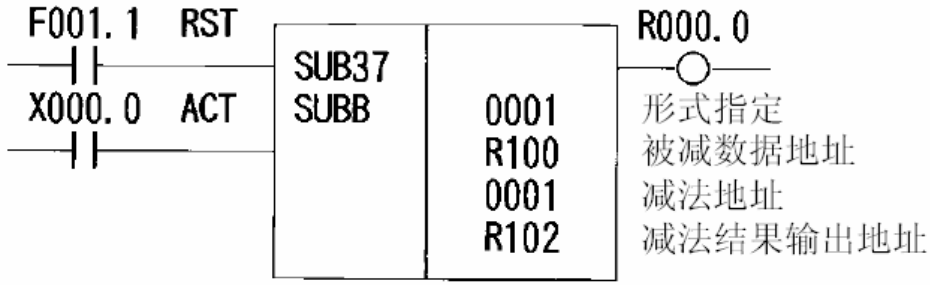
(41)
二进制
减法

进行1、2、4字节长的二进制形式的减法运算。



☞ 控制参数请看ADDB命令。

◀使用例▶

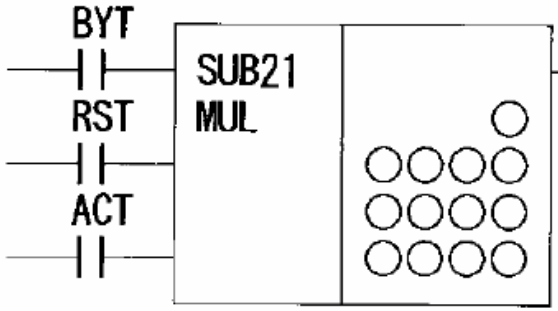


• 由R100减去1，结果写入R102。

R100 | 5 | 时，为 R102 | 4 |

(42)
乘法

进行BCD 2位或4位的乘法运算。



W1 （出错输出）

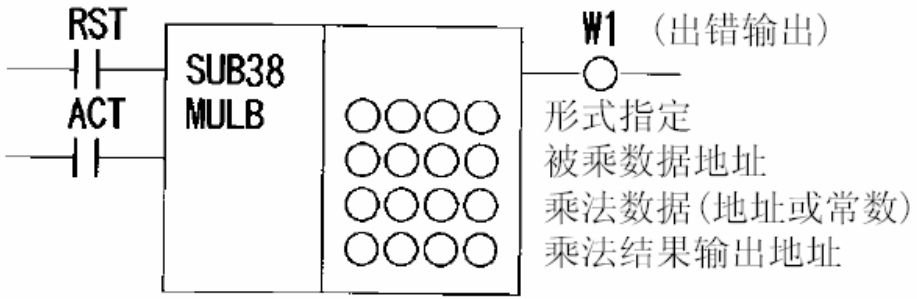
乘法数据形式指定
被乘数据地址
乘法数据(地址或常数)
乘法结果输出地址

W1 =1：乘法结果超出指定的字节数时即接通。

使用例 请看MULB。

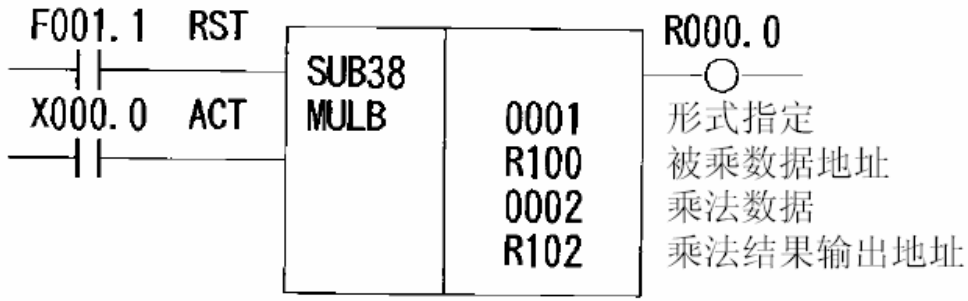
(43)
二进制
乘法

进行1、2、4字节长的二进制形式的乘法运算。



☞ 控制参数请看ADDB命令。

《使用例》

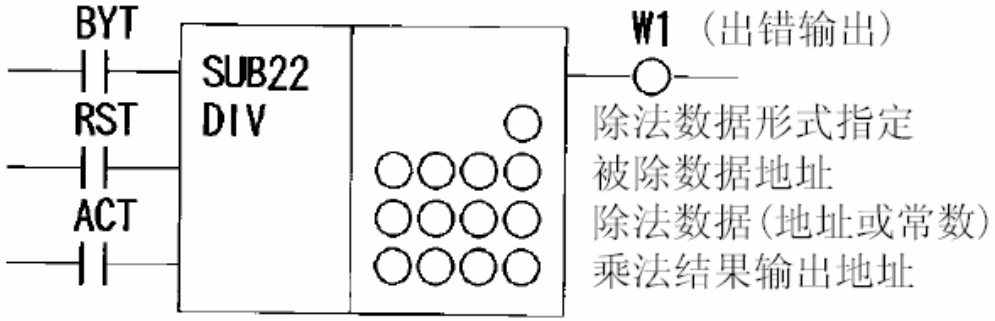


• 用R100乘2，结果写入R102。

R100 | 5 | 时，为 R102 | 10 |

(44)
除法

进行BCD 2位或4位的除法运算。

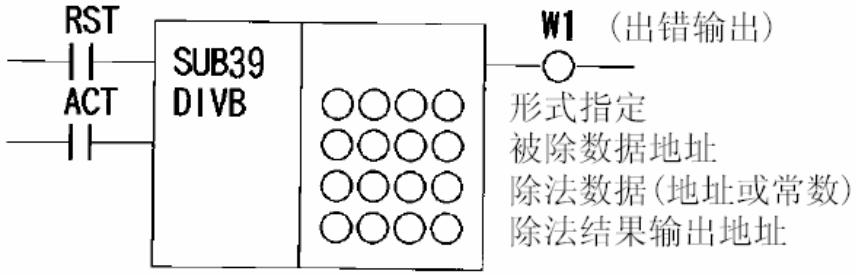


W1 =1: 除法结果超出指定的字节数时即接通。

使用例 请看DIVB。

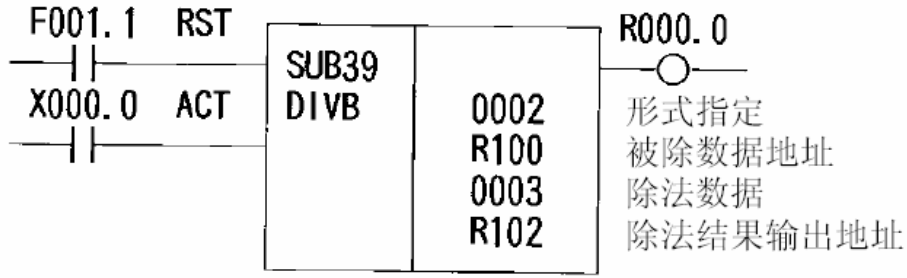
(45)
二进制
除法

进行1、2、4字节长的二进制形式的除法运算。



☞ 控制参数请看ADDB命令。

◀使用例▶



• R100除3，结果写入R102。

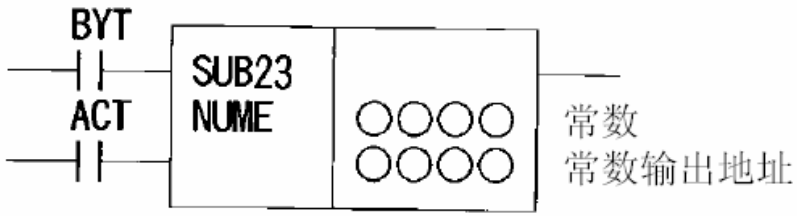
R100,1 | 10 | 时，为 R102,3 | 3

• 余数写入R9002、R9003。
（因为是2字节的运算，所以余数也是2字节）

R9002、R9003 | 1

(46)
常数定义

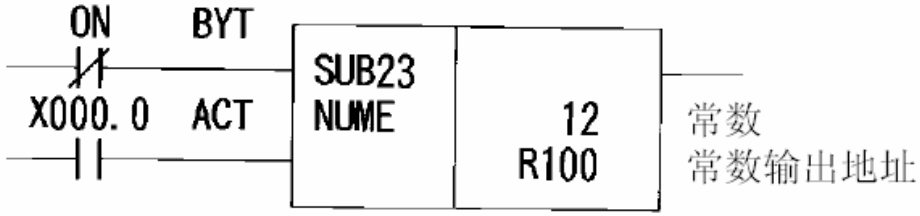
定义BCD 2位或4位的常数。



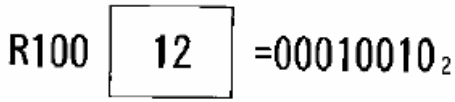
BYT=0：进行处理的数值为BCD 2位。
=1：进行处理的数值为BCD 4位。

ACT=1：执行NUME命令。

◀使用例▶

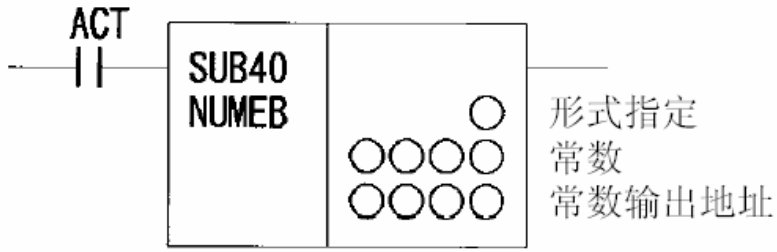


• 用BCD码把12写入R100。



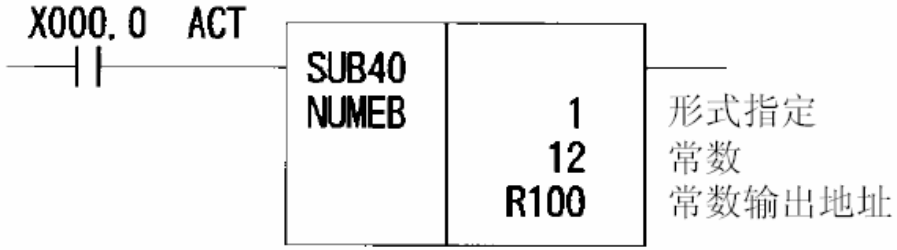
(47)
二进制
常数定义

定义1、2、4字节长的二进制形式的常数。



〔形式指定〕1:1字节长　2:2字节长　4:4字节长

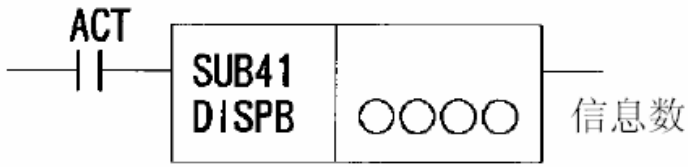
使用例



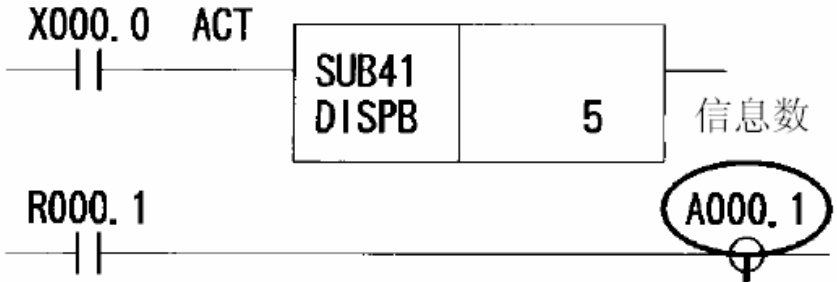
• 用2进数把12写入R100。

R100 | 12 | =00001100$_2$

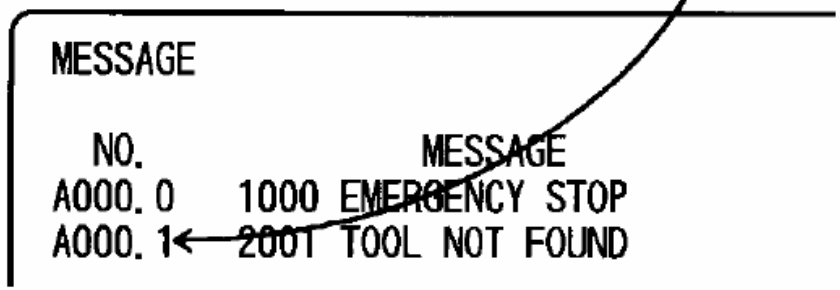
(48)
信息显示



◀使用例▶



• 当X000.0为“1”且A000.1为“1”时，在信息画面上定义的字符串即显示在画面上。

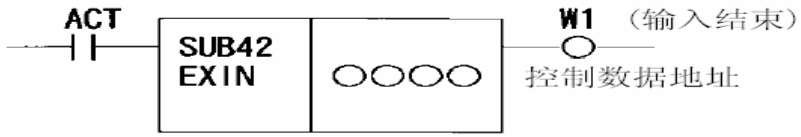


Message number	CNC screen	Display contents
1000 to 1999	Alarm message screen	Alarm message ● CNC is turned to alarm state.
2000 to 2099	Operator message screen	Operator message
2100 to 2999		Operator message (without message number) ● Only message data, no message number, is displayed.

(49)
外部
数据输入

进行外部数据输入(外部刀具补偿、外部信息功能、外部程序号检索、外部工件坐标系偏移、外部机械原点偏移)。

● 与信息显示功能(DISPB)并用时，一定要使用该命令。

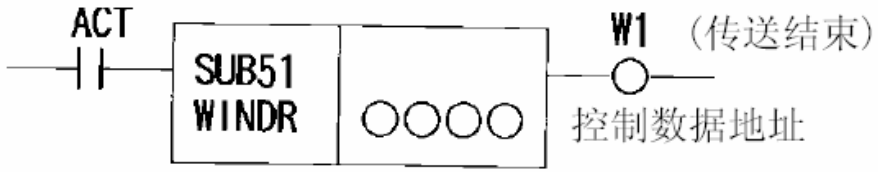
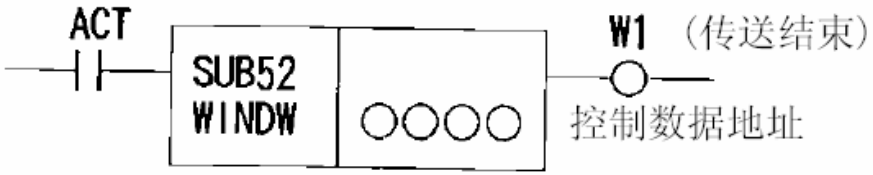


☞ 把ATC=1保持到外部数据输入处理结束。
处理一结束(W1=1)马上ACT=0。

控制数据地址		
+0	(HEAD)	0:主　2:子
+1	ED7~ED0	外部输入数据
+2	ED15~ED8	外部输入数据
+3	STB, EAx	外部数据输入地址

功　能	STB, EAx		ED15~ED0
外部程序号检索	1000	xxxx	程序号(BCD 4位)
外部刀具补偿	1001	xxxx	补偿量(带符号BCD 4位)
外部工件坐标系偏移	1010	轴	偏移量(带符号BCD 4位)
外部机床坐标系偏移	1011	轴	偏移量(BIN 0～±9999)
置入所要零件数	1110	0000	所要数量(BCD 4位)
置入加工零件数	1110	0001	加工数量(BCD 4位)

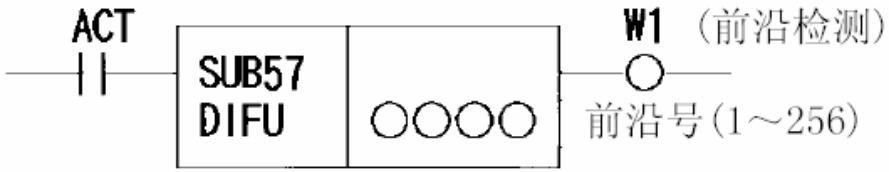
轴		
	第1轴	0000
	第2轴	0001
	第3轴	0010
	第4轴	0011
	⋮	⋮
	第8轴	0111

(50) 窗口数据 读取	可读取机床位置、报警状态、刀具寿命数据等。 
(51) 窗口数据 写入	可写入用户宏变量、参数等。 

(52)

前沿检测

读取输入信号的前沿，扫到1后，输出即为“1”。

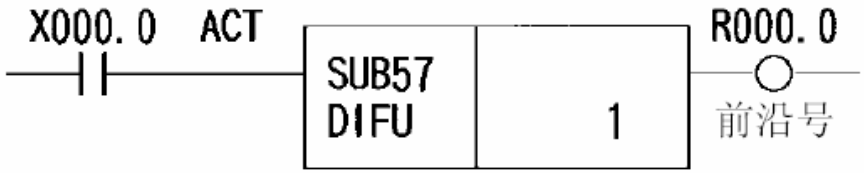


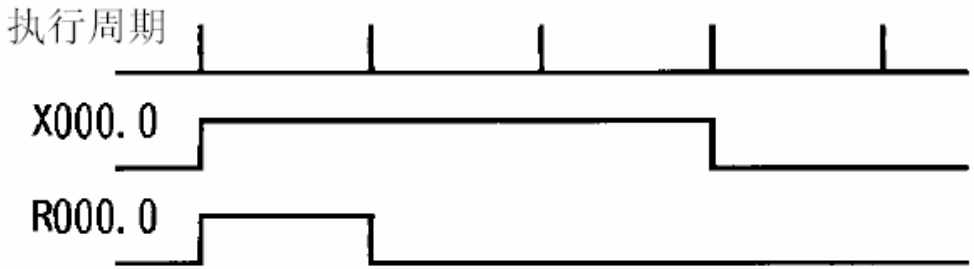
〔前沿号〕1～256

指定进行前沿检测的作业区号。

其他前沿/后沿检测和号重复时，就不能进行正确检测。

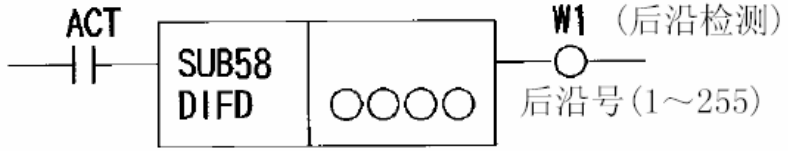
◀使用例▶





(53)
后沿检测

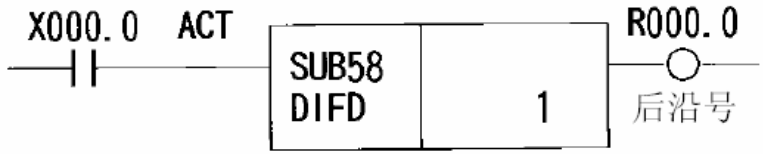
读取输入信号的后沿，扫到1后输出即为“1”。

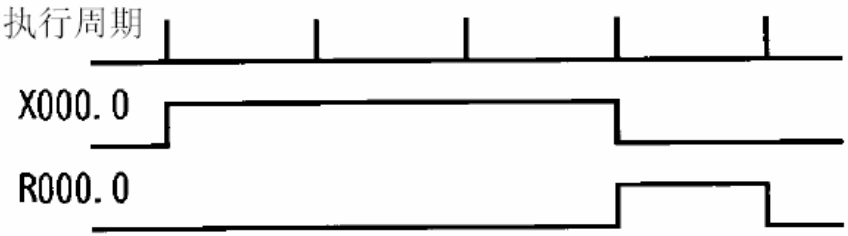


〔后沿号〕1～256
指定进行后沿检测的作业区号。

其他前沿/后沿检测和号重复时，就不能进行正确检测。

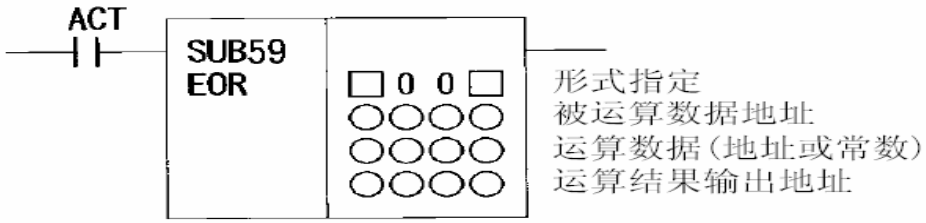
◀使用例▶





（54）
异或

对地址与地址或地址与常数进行异或运算，并把运算结果写入输出地址。



ACT=1：执行EOR命令。

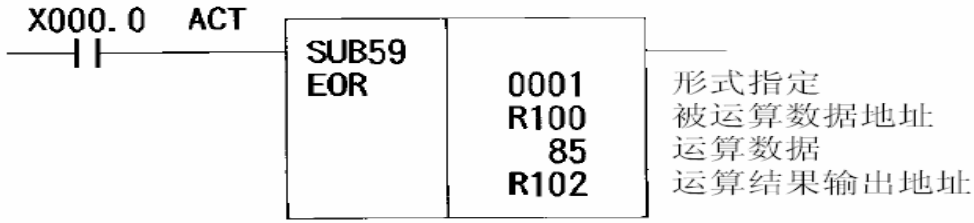
〔形式指定

□ 0 0 □ ← 1：1字节长　2：2字节长　4：4字节长

0：用常数指定运算数据。

1：用地址指定运算数据。

◀使用例▶



• 真值表

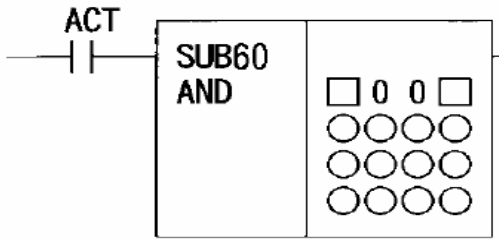
A	B	EOR
0	0	0
0	1	1
1	0	1
1	1	0

• 计算例（1字节）

R100	1	1	1	0	0	0	1	1
运算数据	0	1	0	1	0	1	0	1
R102	1	0	1	1	0	1	1	0

(55)
逻辑乘

对地址与地址或地址与常数进行逻辑乘运算，并把运算结果写入输出地址。



形式指定
被运算数据地址
运算数据(地址或常数)
运算结果输出地址

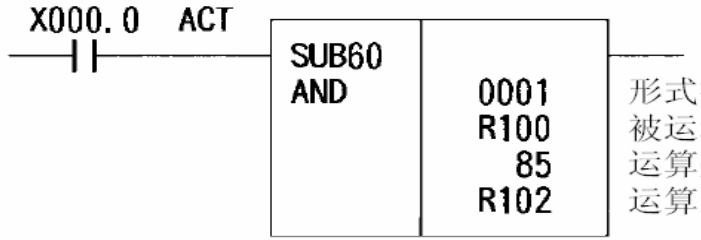
ACT=1：执行AND命令。

〔形式指定〕

□ 0 0 □ ← 1:1字节长　2:2字节长　4:4字节长

0：用常数指定运算数据。
1：用地址指定运算数据。

◀使用例▶



形式指定
被运算数据地址
运算数据
运算结果输出地址

• 真值表

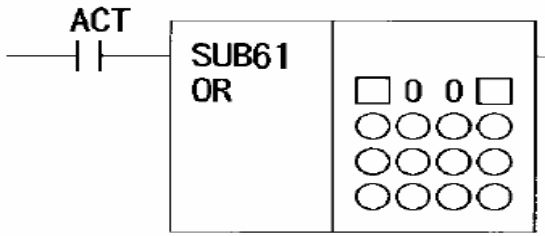
A	B	AND
0	0	0
0	1	0
1	0	0
1	1	1

• 计算例(1字节)

R100	1	1	1	0	0	0	1	1
运算数据	0	1	0	1	0	1	0	1
R102	0	1	0	0	0	0	0	1

(56) 逻辑和

对地址与地址或地址与常数进行逻辑和运算，并把运算结果写入输出地址。



ACT=1：执行OR命令。

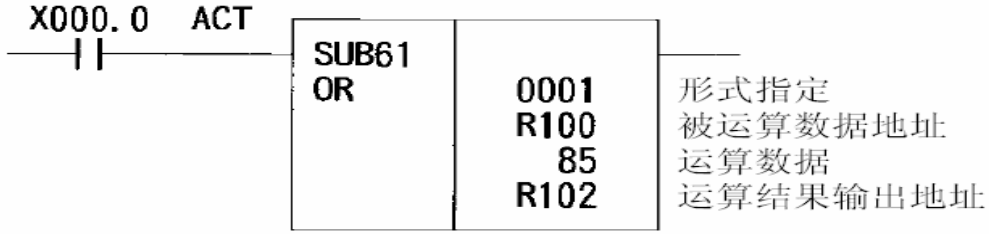
〔形式指定〕

□ 0 0 □ ← 1:1字节长　2:2字节长　4:4字节长

↑ 0：用常数指定运算数据。

↑ 1：用地址指定运算数据。

使用例



• 真值表

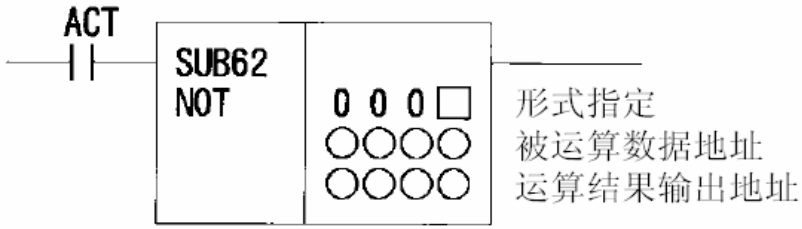
A	B	OR
0	0	0
0	1	1
1	0	1
1	1	1

• 计算例(1字节)

R100	1	1	1	0	0	0	1	1
运算数据	0	1	0	1	0	1	0	1
R102	1	1	1	1	0	1	1	1

(57)
逻辑非

对被指定地址的二进制数据进行逻辑非运算(把0和1翻转)，并把运算结果写入输出地址。

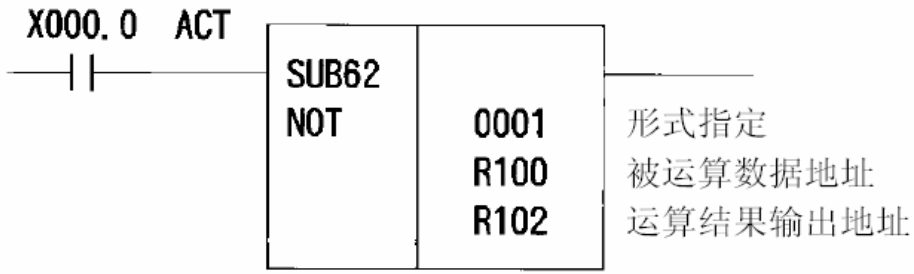


ACT=1: 执行OR命令。

〔形式指定〕

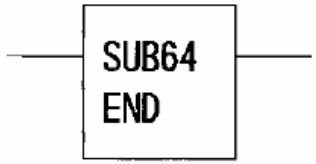
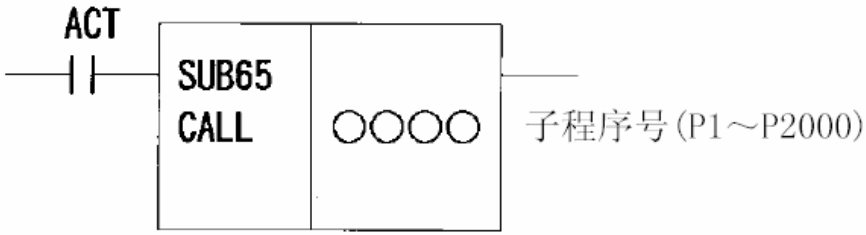
0 0 0 □ ← 1:1字节长　2:2字节长　4:4字节长

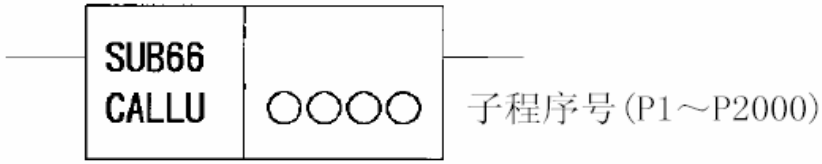
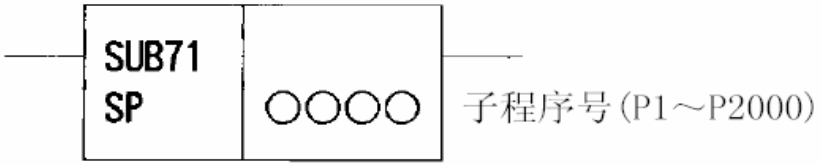
使用例



・计算例(1字节)

R100	1	1	1	0	0	0	1	1
R102	0	0	0	1	1	1	0	0

(58) 程序结束	编写子程序时，在子程序的最后写入该命令。 
(59) 子程序 有条件 调出	ACT=1 时，转移到被指定的子程序号。  ACT=1:调出被指定的子程序。 〔子程序号〕 指定被调出的子程序号。 ☞PMC-RA3, RB3/RC3 为P1~P512 PMC-RB4, RB6/RC4 为P1~P2000

(60) 子程序 无条件 调出	无条件转移到指定的子程序号。  〔子程序号〕 指定被调出的子程序号。 ☞PMC-RA3, RB3/RC3 为P1~P512 PMC-RB4, RB6/RC4 为P1~P2000
(61) 子程序	指示子程序的开始。  ☞PMC-RA3, RB3/RC3 为P1~P512 PMC-RB4, RB6/RC4 为P1~P2000
(62) 子程序 结束	指示子程序的结束。