目录

◎ 应用命令通则

♦应用命令被设计成以功能数字表示(FNC 00-99)每一应用命令被冠以一个”符号”。
例 : FNC 12 被冠以“MOV”
♦有些应用命令的格式仅须指定 FNC 数字的部份，有些须再加入操作数。

◎ COMPARE

FNC(10)			16 bits: CMP & CMP(P) - - - - - - - - - - - - - - - - - - -7 steps
D	CMP	P	32 bits:(D)CMP & (D)CMP(P) - - - - - - - - - - - - - -13 steps

Operands: [S1.][S2.]

K.H.	KnX	KnY	KnM	KnS	T	C	D	V,Z

Operands: [D.]

X	Y	M	S

[S.] :代表一个操作数，其被称为来源”Source”操作数，如果”Source”超过一个时，则表示成[S1.] [S2.]等。
[D.] :代表一个操作数，其被称为目的”Destination”操作数，如果”Destination”超过一个时，则表示成[D1.] [D2.] 等。

♦ 指令部份占用一个程序步序，操作数则占用 1-2 程序步序，占用程序步序的多寡完全由指令是 16 位元或 32 位元指令
来决定。

♦ X,Y,M,S,等位要素(bit device)亦可当作操作数处理，经组合后，以字符要素(word device)的形态表示。

♦ 数据缓存器(D)皆为 16 位，当处理 32 位数据时将自动成对使用。
例:指定 D0 执行 32 位运算时，将会取(D1,D0)的数据来做处理(D1 为上 16 位，D0 为下 16 位)。

♦ 32 位的计数器则不能当作 16 位命令的操作数。

◎ 数据长度及命令执行格式

♦ 32 位元的指令被表示成在指令前加入(D)。例:(D) MOV

♦ 不管指令的要素号码是偶数还是奇数皆可使用，但为避免混乱尽量使用偶数要素号码来配合 32 位使用。

◎ 位要素与字符要素

♦ 诸如 X,Y,M 及 S 等要素仅能以 ON/OFF 状态表示，因此被称为”位要素”，其它要素如 T,C 及 D 等要素能表示数值数据，因此被称为”字符要素”，但对于 X,Y,M,S 组成的位要素亦可以字符的状态来表现，只要在前面加上指定位数的 Kn 符号即可。

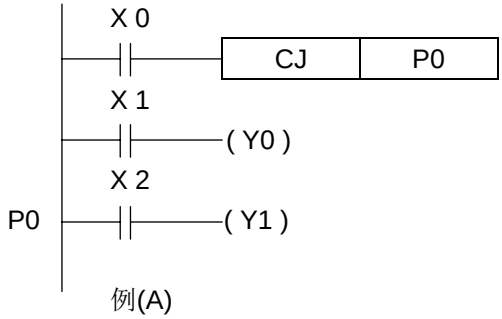
♦ 位要素能组合以 4 个位为单位的要素，K1~K4 允许 16 位数据运算，K1~K8 允许 32 位数据运算。
例:K2M0 表示由 M0 开始 2 个 4 位为单位的要素，即 M0~M7 组合成的字符。

◎ 条件跳跃 CONDITION JUMP

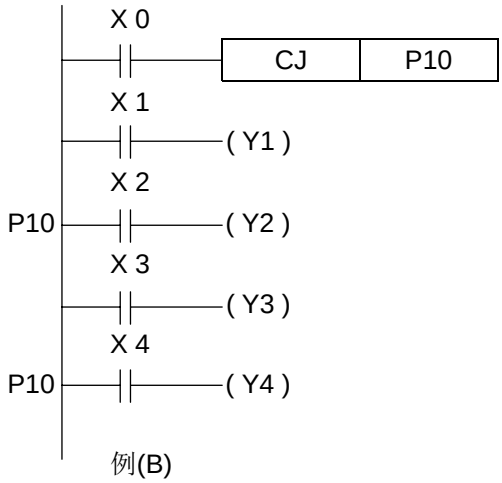
FNC(00)			16 bits: CJ & CJ(P) - 3 Steps
	CJ	P	

		J1n	J2n--

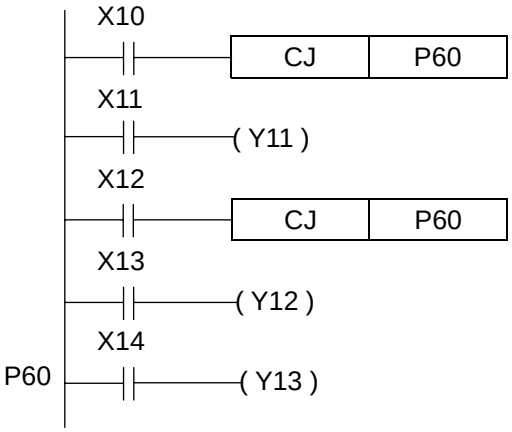
Operand: P00 ~ P63



例(A)



例(B)



例(C)

- 使用 CJ(P)隔开部分程序，能减少演算周期且允许双重线圈线路。
- 例 (A):假如 X0 ON 强迫程序跳至 LAB P0，被隔开的指令将不被执行，即使输入条件动作，输出状态亦不会改变。
- 例 (A):假如未写 LAB P0，X0 ON 则直接跳至 END。
- 当使用逆向跳跃时，须注意 watchdog timer overrun。
- 假如 P 标记号码重复时，仅最后的标记号码有效。
- 例 (B):X0 ON 则程序跳至第二 LAB P10。
- 例 (C):多个 CJ(P)指令可指定跳至同一个标记号码。

◎ 呼叫子程序 Subroutine Call

FNC(01)			16 bits: CALL & CALL(P) - - - - - - - - - - - - - - - - - - -3 Steps
	CALL	P	

		J1n	J2n--

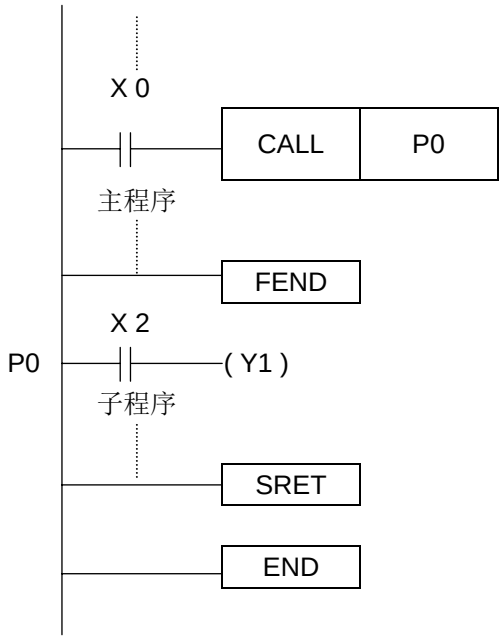
Operand: P00~P63

◎ 子程序返回 Subroutine Return

FNC(02)			16 bits: SRET - 1 Steps
	SRET		

		J1n	J2n--

Operand: None



- 当 X0 ON，程序会跳至 LAB P0 执行子程序，直到 SRET 被执行，则程序返回至主程序继续执行。
- 子程序必须写在 FEND 之后。
- 多个 CALL 指令可指定至同一个标记号码。
- 在同一个 CALL 指令下最多四层的子程序被使用。
- 子程序之后须写 SRET 指令。

◎ 中断返回 Interrupt Return

FNC(03)	16 bits: IRET - 1 Steps			J1n	J2n--
IRET					

Operand: None

◎ 中断致能 Enable Interrupt

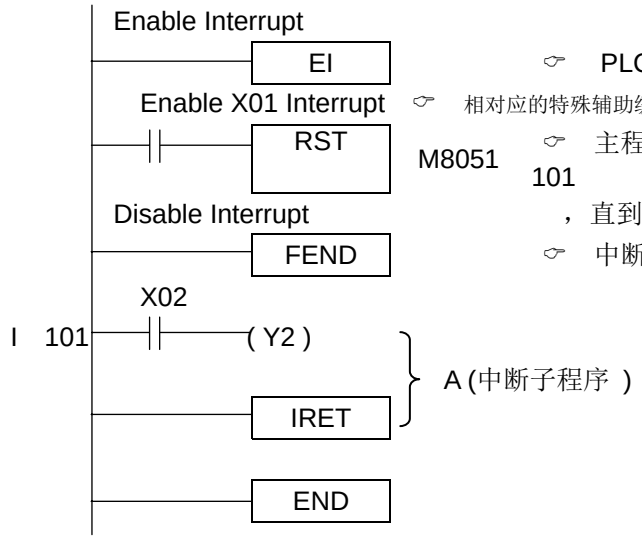
FNC(04)	16 bits: EI - 1 Steps			J1n	J2n--
EI					

Operand: None

◎ 中断禁止 Disable Interrupt

FNC(05)	16 bits: DI - 1 Steps			J1n	J2n--
DI					

Operand: None



☞ PLC 常处于中断禁止状态，当欲使用中断时须写入 EI 指令。

☞ 相对应的特殊辅助缓存器动作时(M8051 ON)，则中断子程序将不执行。

☞ 主程序下当中断产生时(X01 ON)，则程序跳往中断子程序 I 101 ，直到 IRET 指令再返回主程序继续执行。

☞ 中断子程序必须写在 FEND 指令之后。

中断指针号码

I □ 0 □ 输入中断 6 点

- 0：下缘中断 1：上缘中断
- 输入 X0 ~ X5 的号码 0 ~ 5 (不可重复使用)

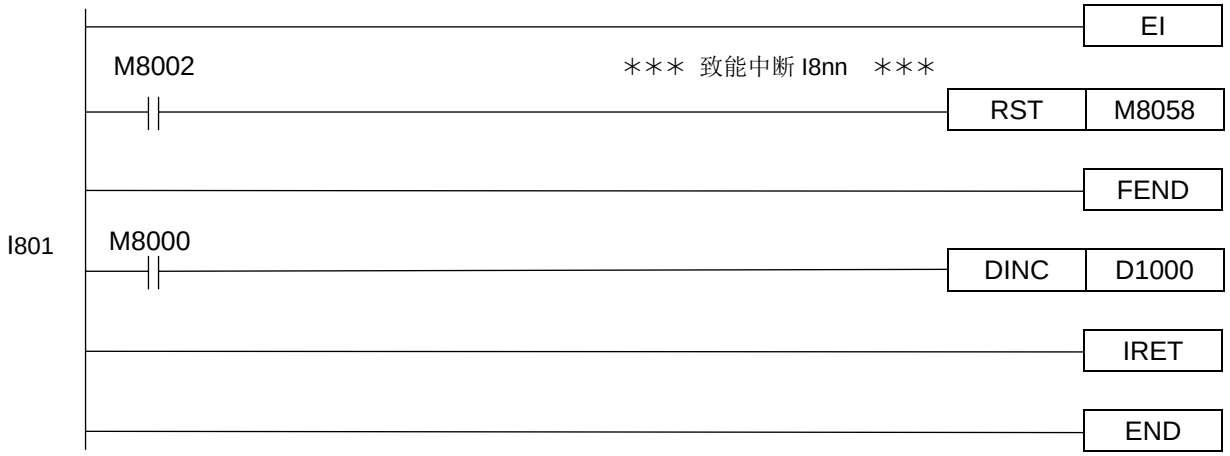
I □ □ □ 定时器中断 1 点

- 01 ~ 99 ms
- 8 (不可重复使用)

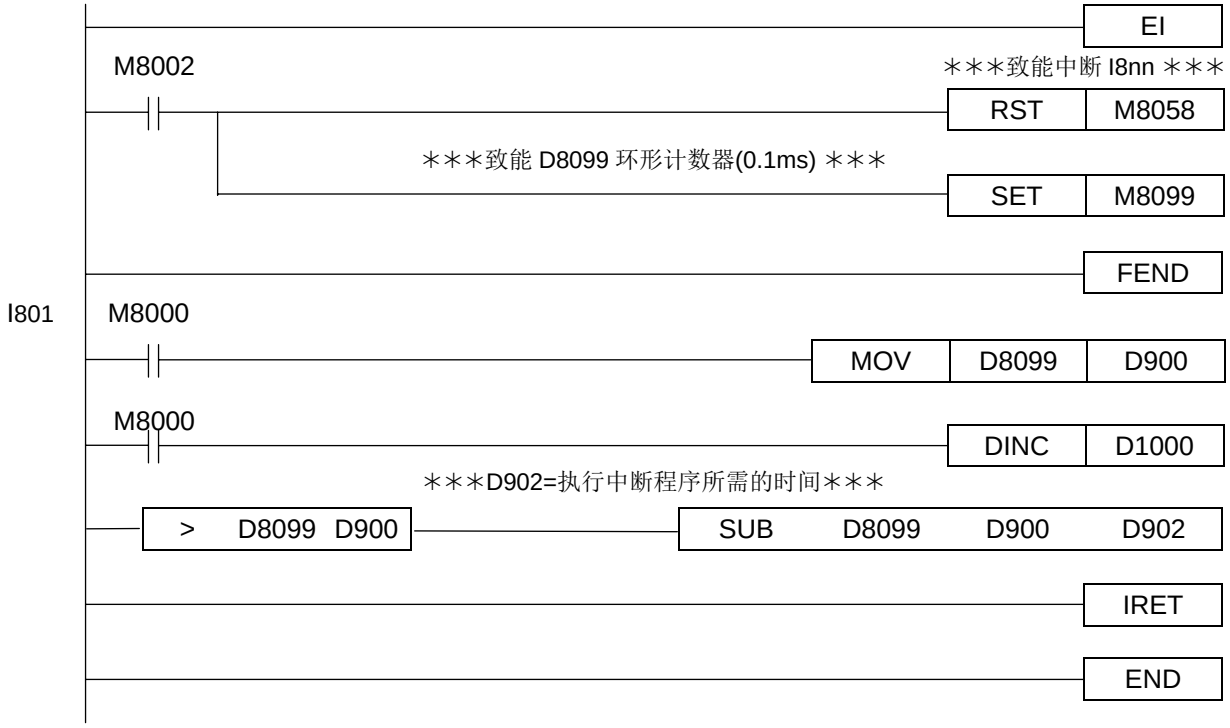
<<注意事项>>

- 当一中断程序执行时，其它中断呼叫视为无效。
- 假如中断发生在中断禁止范围内(DI ~ EI)时，这中断要求信号暂时被储存，待中断致能范围内(EI ~ DI)再执行。
- 中断禁止旗标 M805n 动作时(n=0 – 5)，相对应的中断输入将不被执行。
- 中断程序内不可使用 FNC(50) REF 指令。(如上述范例程式中的 A 区段)

◎ 时间中断范例



◎ 计算时间中断程序执行时间范例

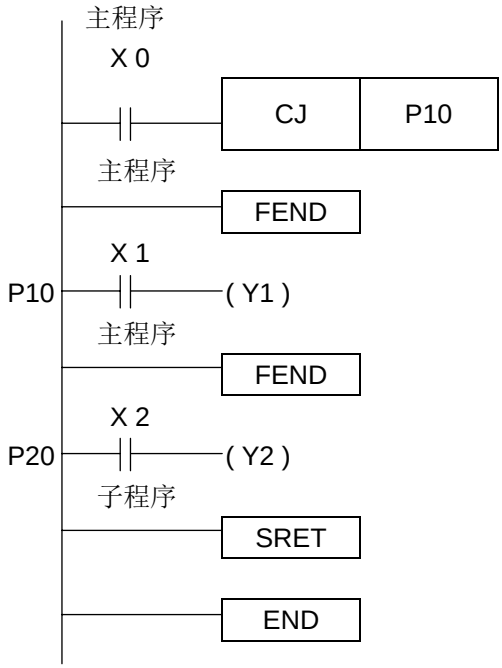


◎ 主程序结束 First End

FNC(06)	16 bits: FEND - 1 Steps
FEND	

		J1n	J2n--

Operand: None



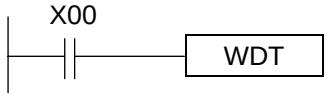
- FEND 指令表示主程序结束，子程序及中断程序开始书写的区域。
- FEND 指令被执行则程序返回 STEP 0。
- 多个 FEND 指令用来分离不同的主程序。
- FEND 不能写在 END 指令之后。

◎ 定时器 Watch Dog Timer

FNC(07)	16 bits: WDT - 1 Steps
WDT P	

		J1n	J2n--

Operand: None



- 此命令主要在检查扫瞄周期的时间是否大于特殊数据缓存器 D8000 的内容值。
- 若大于 D8000 的内容值，则会产生错误，错误码 6309。
- 特殊数据缓存器 D8000 的内容值可利用 MOV 命令来变更。
- 未写入此命令则不执行时间的比对。

◎ 重复起始点 FOR

FNC(08)	16 bits: FOR - 3 Steps
FOR	

		J1n	**J2n--**

Operands: ←——— [S.] ———→

K.H.	KnX	KnY	KnM	KnS	T	C	D	V,Z

影响旗号:

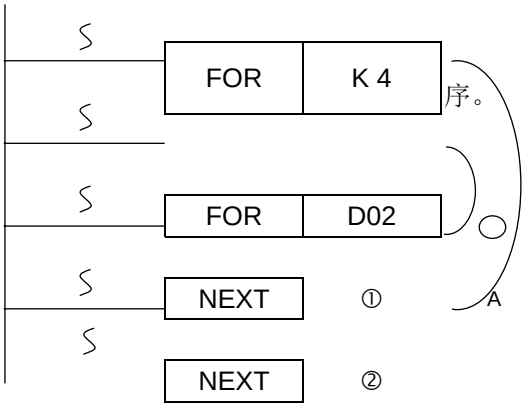
◎ 重复结束点 NEXT

FNC(09)	16 bits: NEXT - 1 Steps
NEXT	

		J1n	**J2n--**

Operand: None

时，



序。

- 程序 B 被执行 4 次后，再执行第②个 NEXT 以下的程
- 若数据缓存器 D0Z 中的值为 5，则程序 B 每一次执行

程序 A 将执行 5 次，也就是说程序 A 总共要执行 20 次。

- FOR–NEXT 回路至多 5 层。

◎ 比较 COMPARE

FNC(10)			16 bits: CMP & CMP(P) - - - - - - - - - - - - - - - - - - - 7 Steps
D	CMP	P	32 bits:(D)CMP&(D)CMP(P) - - - - - - - - - - - - - - - 13 Steps

		J1n	J2n--

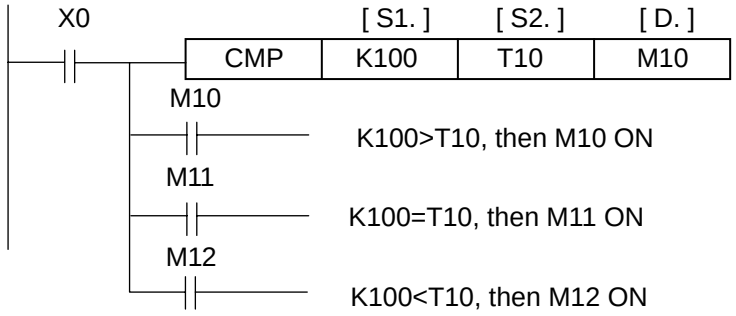
Operands: [S1.][S2.]

K.H.	KnX	KnY	KnM	KnS	T	C	D	V,Z

Operands: [D.]

X	Y	M	S

影响旗号:



- 来源[S1.]与[S2.]内的数据互相比较，[D.]则依据比较的结果产生变化。本指令目的位元将自动占据 3bits，本例为(M10-M12)。
- 来源数据以代数方式做比较。例: -10 < 2。
- 当 X0 OFF，则[D.]位状态不改变。

◎ 区域比较 ZONE COMPARE

FNC(11)			16 bits: ZCP & ZCP(P) - - - - - - - - - - - - - - - - - - - 9 Steps
D	ZCP	P	32 bits: (D)ZCP&(D)ZCP(P) - - - - - - - - - - - - - - - -17 Steps

		J1n	J2n--

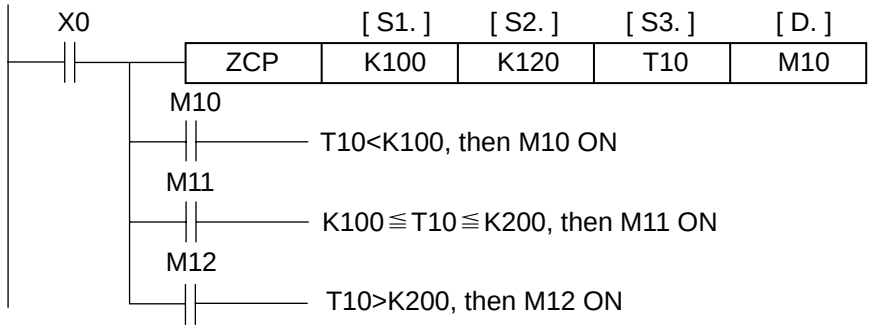
Operands: [S1.] [S2.] [S3.]

K.H.	KnX	KnY	KnM	KnS	T	C	D	V,Z

Operands: [D.]

X	Y	M	S

影响旗号:



- 来源[S3.]内的资料与来源[S1.]及[S2.]内的数据范围做比较，[D.]则依据比较的结果产生变化。本指令目的位元将自动占据 3bits，本例为(M10-M12) 。
- 请设定 [S1.] ≦ [S2.]。若 [S1.] > [S2.] 时，则 [S2.] 的值视为与 [S1.] 的数值相同。
- 来源数据以代数方式做比较。例: -10 < 2 。
- 当 X0 OFF，则[D.]位状态不改变。

◎ 搬移 MOVE

FNC(12)			16 bits: MOV & MOV(P) - - - - - - - - - - - - - - - - - - - 5 Steps
D	MOV	P	32 bits:(D)MOV&(D)MOV(P) - - - - - - - - - - - - - - - - 9 Steps

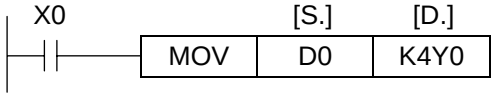
		J1n	J2n--

Operands: [S.]

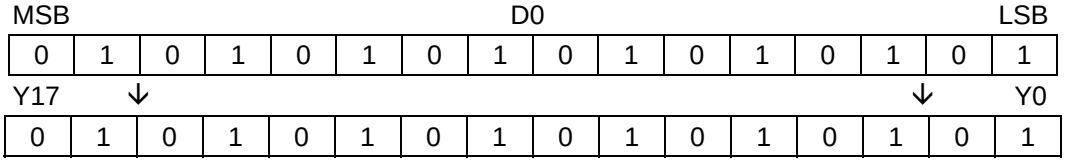
K.H.	KnX	KnY	KnM	KnS	T	C	D	V,Z

[D.]

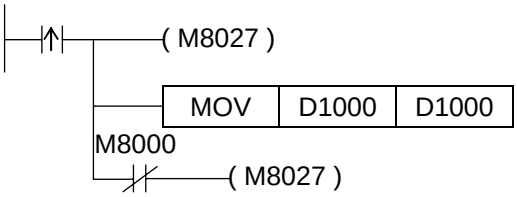
影响旗号:



♦ 当 X0 ON，[S.]内的数据搬移至[D.]中。



♦ 当 M8027 ON 时，系统同时将 [S.] 之数据写入 EEPROM 相对应之 [D.] 中。



注意事项：当 M8027 ON 时，为避免破坏 EEPROM，请使用脉波指令且目的字符仅 D 缓存器有效。

◎ 移位搬移 Shift Move

FNC(13)			16 bits: SMOV & SMOV(P) - - - - - - - - - - - - - - - - - 7 Steps
	SMOV	P	

Operands: [S.]

K.H.	KnX	KnY	KnM	KnS	T	C	D	V,Z

[D.]

♦ Reserved

◎ 互补 COMPLEMENT

FNC(14)			16 bits: CML & CML(P) - - - - - - - - - - - - - - - - - - - 5 Steps
D	CML	P	32 bits: (D)CML & (D)CML(P) - - - - - - - - - - - - - - -9 Steps

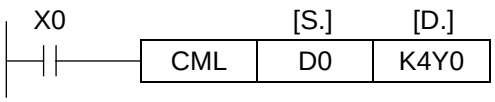
		J1n	J2n--

Operands: [S.]

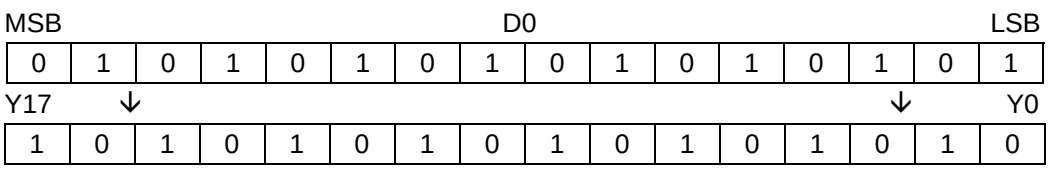
K.H.	KnX	KnY	KnM	KnS	T	C	D	V,Z

[D.]

影响旗号:



♦ [S.] 的每一位数值被反相 (0→1,1→0) 搬移至 [D.] 中。



◎ 区块搬移 BLOCK MOVE

FNC(15)			16 bits: BMOV & BMOV(P) - - - - - - - - - - - - - - - - 7 Steps
	BMOV	P	

		J1n	J2n--

Operands: ←——— [S.] ———→

K.H.	KnX	KnY	KnM	KnS	T	C	D	V,Z

←n→ ←——— [D.] ———→

n ≤ 128

影响旗号: 无

[S.] [D.] [n.]

BMOV	D10	D20	K10

♦当 X0 ON，搬移顺序如下

X0

RST	M8160

BMOV	D10	D8	K10

D19	D18	D17	D16	D15	D14	D13	D12	D11	D10
↓⑩	↓⑨	↓⑧	↓⑦	↓⑥	↓⑤	↓④	↓③	↓②	↓①
D17	D16	D15	D14	D13	D12	D11	D10	D9	D8

♦当转送编号重复时，搬移顺序如下

X0

RST	M8160

BMOV	D10	D12	K10

D10	D11	D12	D13	D14	D15	D16	D17	D18	D19
↓⑩	↓⑨	↓⑧	↓⑦	↓⑥	↓⑤	↓④	↓③	↓②	↓①
D12	D13	D14	D15	D16	D17	D18	D19	D20	D21

♦当 M8027 ON 时，系统同时将[S.]区块数据写入 EEPROM 相对应之[D.]中，仅 D 缓存器有效。

SET	M8027

BMOVP	D10	D1000	K8

RST	M8027

注意事项：当 M8027 ON 时，为避免破坏 EEPROM，请使用脉波指令且目的字符仅 D 缓存器有效。

♦当 M8160 为 ON 时，搬移情况如下(M8027 不可为 ON) (V2.85 后有效)，

X0

SET	M8160

BMOV	D10	D20	K4

D14 下	D13 上	D13 下	D12 上	D12 下	D11 上	D11 下	D10 上	D10 下
↓	↓	↓	↓	↓	↓	↓	↓	↓
D24 上	D24 下	D23 上	D23 下	D22 上	D22 下	D21 上	D21 下	D20 上	D20 下

◎ 多点搬移 FILL MOVE

FNC(16)			16 bits: FMOV & FMOV(P) - - - - - - - - - - - - - - - - - 7 Steps
D	FMOV	P	32 bits: (D)FMOV & (D)FMOV(P) - - - - - - - - - - - -13 Steps

		J1n	J2n--

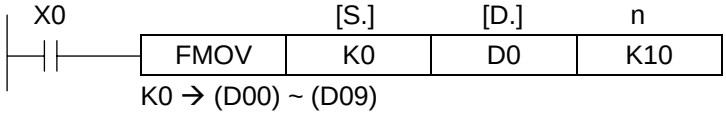
Operands: [S.]

K.H.	KnX	KnY	KnM	KnS	T	C	D	V,Z

← n → [D.]

n ≤ 128

影响旗号:



K0 → (D00) ~ (D09)

◎ 互换 EXCHANGE

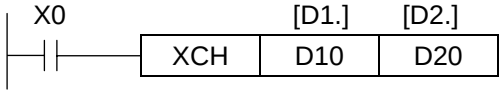
FNC(17)			16 bits: XCH & XCH(P) - - - - - - - - - - - - - - - - - - - 5 Steps
D	XCH	P	32 bits: (D)XCH & (D)XCH(P) - - - - - - - - - - - - - -9 Steps

		J1n	J2n--

Operands: [D1.] [D2.]

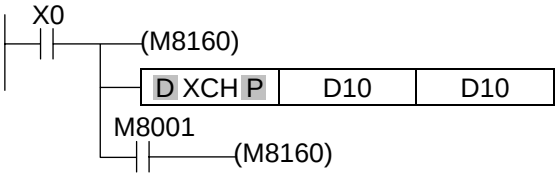
K.H.	KnX	KnY	KnM	KnS	T	C	D	V,Z

影响旗号:



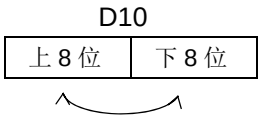
执行前 : (D10)=100　　执行后 : (D10)=200
(D20)=200　　(D20)=100

<< 扩充机能 >> SWAP



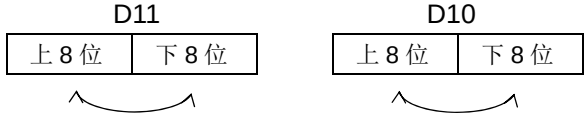
- M8160 为 ON，[D1.]与[D2.]又为同一要素时，将会执行上 8 位与下 8 位数据的交换。
- 若[D1.]与[D2.]不为同一要素时，错误旗标 M8067 ON，错误码 6705，错误步序存入 D8069，且命令不执行。

16 位:



执行前(D10)=0050H=80，执行后(D10)=5000H=20480

32 位:



执行前(D11,D10)=87654321H=80，执行后 65872143H

◎ BCD 变换 (BINARY CODE TO DECIMAL)

FNC(18)			16 bits: BCD & BCD(P) - - - - - - - - - - - - - - - - - - - 5 Steps
D	BCD	P	32 bits: (D)BCD & (D)BCD(P) - - - - - - - - - - - - - - -9 Steps

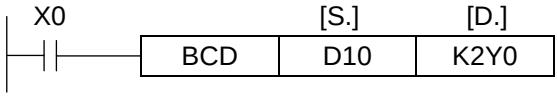
		J1n	**J2n--**

Operands: [S.]

K.H.	KnX	KnY	KnM	KnS	T	C	D	V,Z

[D.]

影响旗号:



- ♦ 来源 [S.] 二进制 (BIN) 的数据转换成十进制 (BCD) 存入 [D.] 。
- ♦ 假如十进制 BCD 的数值超过 0 - 9999 (16 bits operation)或 0 - 99999999 (32 bits operation)则错误旗标 M8067ON，

  错误代码 6705，错误步序存入 D8069，程序继续执行，但演算结果不存入[D.]中。
- ♦ 此指令可用来驱动七段显示器。

◎ BIN 变换 (DECIMAL CODE TO BINARY)

FNC(19)			16 bits: BIN & BIN(P) - 5 Steps
D	BIN	P	32 bits: (D)BIN & (D)BIN(P) - - - - - - - - - - - - - - - - -9 Steps

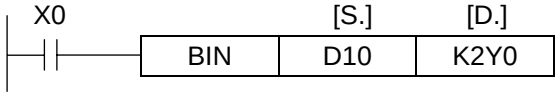
		J1n	**J2n--**

Operands: [S.]

K.H.	KnX	KnY	KnM	KnS	T	C	D	V,Z

[D.]

影响旗号:



- ♦ 来源[S.]十进制(BCD)的数据转换成二进制(BIN)存入[D.]。
- ♦ 若来源[S.]数据不符合 BCD 的格式，则错误旗标 M8067 ON，错误代码 6705，错误步序存入 D8069，程序继续执行，但不执行演算。
- ♦ 来源[S.]不可指定为常数 K/H。

◎ 加法 ADDITION

FNC(20)			16 bits: ADD & ADD(P) - - - - - - - - - - - - - - - - - - - 7 Steps
D	ADD	P	32 bits: (D)ADD & (D)ADD(P) - - - - - - - - - - - - - -13 Steps

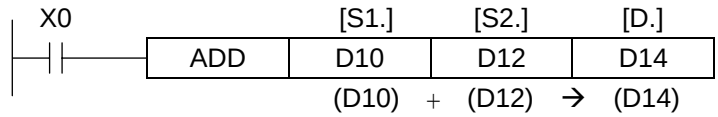
		J1n	**J2n--**

Operands: [S1.][S2.]

K.H.	KnX	KnY	KnM	KnS	T	C	D	V,Z

[D.]

影响旗号: M8020, M8021, M8022



- 来源[S1.]与[S2.] BIN 资料相加结果存入[D.]中。
- 所有运算均以代数进行，i.e. 5+(-8) = -3.
- 最高位代表正负号 (0:正，1:负)。
- 运算结果等于“0”，则零位旗标 M8020 ON。
- 运算结果大于 32,767 (16 bit operation) 或 3,147,483,647 (32 bit operation)，则进位旗标 M8022 ON。
- 运算结果小于 -32,767 (16 bit)或 -2,147,483,647 (32 bit)，则借位旗标 M8021 ON。

◎ 减法 SUBTRACTION

FNC(21)			16 bits: SUB & SUB(P) - - - - - - - - - - - - - - - - - - - 7 Steps
D	SUB	P	32 bits: (D)SUB & (D)SUB(P) - - - - - - - - - - - - - -13 Steps

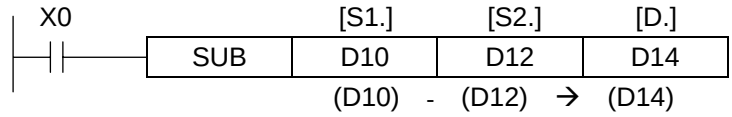
		J1n	**J2n--**

Operands: [S1.][S2.]

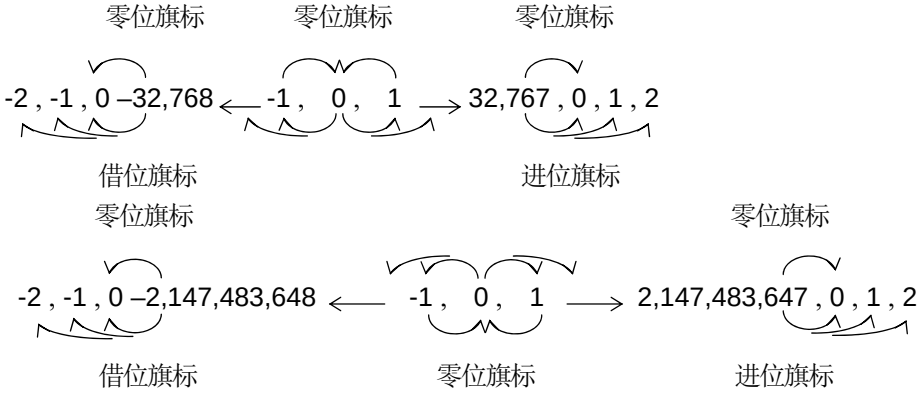
K.H.	KnX	KnY	KnM	KnS	T	C	D	V,Z

[D.]

影响旗号: M8020, M8021, M8022



- 来源 [S1.] 与 [S2.] 资料内容相减结果存入 [D.]。
- 所有运算均以代数进行，i.e. 5+(-8) = -3。
- 最高位代表正负号 (0:正，1:负) 。
- 旗号设定及正负数间的关系。



◎ 乘法 MULTIPLICATION

FNC(22)			16 bits: MUL & MUL(P) - 7 Steps
D	MUL	P	32 bits: (D)MUL & (D)MUL(P) - - - - - - - - - - - - - - - - - - -13 Steps

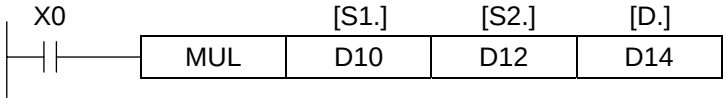
		J1n	J2n--

Operands: [S1.][S2.]

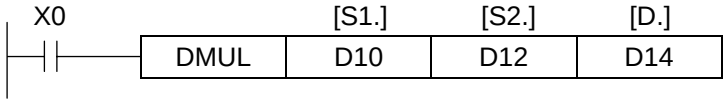
K.H.	KnX	KnY	KnM	KnS	T	C	D	V,Z

[D.]

影响旗号:



16 bits: (D10) × (D12) → (D15,D14)



32 bits: (D11,D10) × (D13,D12) → (D17,D16,D15,D14)

♦ 来源 [S1.] 乘以 [S2.] 结果存入 [D.] 中。

◎ 除法 DIVISION

FNC(23)			16 bits: DIV & DIV(P) - 7 Steps
D	DIV	P	32 bits: (D)DIV & (D)DIV(P) - - - - - - - - - - - - - - - - -13 Steps

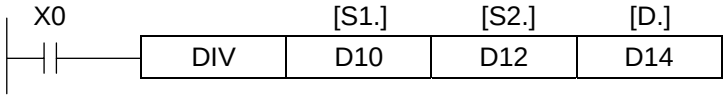
		J1n	J2n--

Operands: [S1.][S2.]

K.H.	KnX	KnY	KnM	KnS	T	C	D	V,Z

[D.]

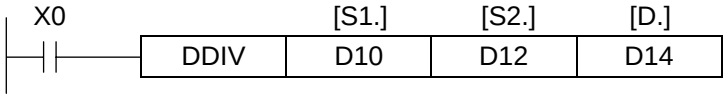
影响旗号:



Dividend divisor quotient remainder

(D10) ÷ (D12) → (D14) . . . . . (D15)

16 bits 16 bits 16 bits 16 bits



Dividend divisor quotient remainder

(D11,D10) ÷ (D13,D12) → (D15,D14) ..... (D17,D16)

32 bits 32 bits 32 bits 32 bits

♦ 来源 [S1.] 除以 [S2.] 结果存入 [D.] 中。

♦ 若来源 [S2.] 等于 “0” (zero)，则程序停止运转，错误旗标 M8067 ON，错误代码 6706。

♦ V1.17 版：若来源 [S2.] = ”0”，则不予处理，直接跳至下一个指令。

◎ 递增 INCREMENT

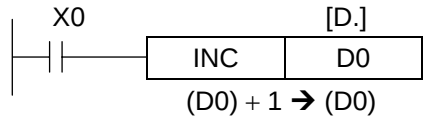
FNC(24)			16 bits: INC & INC(P) - 3 Steps
D	INC	P	32 bits: (D)INC & (D)INC(P) - - - - - - - - - - - - - - - - -5 Steps

		J1n	**J2n--**

K.H.	KnX	KnY	KnM	KnS	T	C	D	V,Z

Operands: ←——— [D.] ———→

影响旗号:



- 每次执行目的字符 [D.] 的内容加 ”1”。
- 16 bits 演算，当到达 +32,767，则下次执行会将 -32,768 写入 [D.] 中。
- 32 bits 演算，当到达 +2,147,483,647，则下次执行会将 -2,147,483,648 写入 [D.] 中。
- 演算过程进位旗标、零位旗标、借位旗标均不变。

◎ 递减 DECREMENT

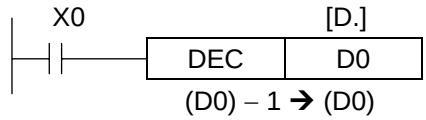
FNC(25)			16 bits: DEC & DEC(P) - 3 Steps
D	DEC	P	32 bits: (D)DEC & (D)DEC(P) - - - - - - - - - - - - - - - -5 Steps

		J1n	**J2n--**

K.H.	KnX	KnY	KnM	KnS	T	C	D	V,Z

Operands: ←——— [D.] ———→

影响旗号:



- 每次执行目的字符 [D.] 的内容减”1”。
- 16 bits 演算，当到达 -32,767，则下次执行会将 +32,768 写入 [D.] 中。
- 32 bits 演算，当到达 -2,147,483,647，则下次执行会将+2,147,483,648 写入 [D.] 中。
- 演算过程进位旗标、零位旗标、借位旗标均不变。

◎ 逻辑及 LOGICAL AND

FNC(26)			16 bits: WAND & WAND(P) - - - - - - - - - - - - - - - - 7 Steps
D	WAND	P	32 bits: (D)WAND & (D)WAND(P) - - - - - - - - - - -13 Steps

		J1n	J2n--

◎ 逻辑或 LOGICAL OR

FNC(27)			16 bits: WOR & WOR(P) - - - - - - - - - - - - - - - - - - 7 Steps
D	WOR	P	32 bits: (D)WOR & (D)WOR(P) - - - - - - - - - - - - -13 Steps

		J1n	J2n--

◎ 互斥或 XOR

FNC(28)			16 bits: WXOR & WXOR(P) - - - - - - - - - - - - - - - - 7 Steps
D	WXOR	P	32 bits: (D)WXOR &(D)WXOR(P) - - - - - - - - - - - 13 Steps

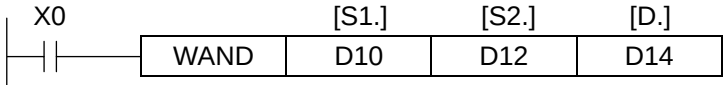
		J1n	J2n--

Operands: ←— [S1.][S2.] —→

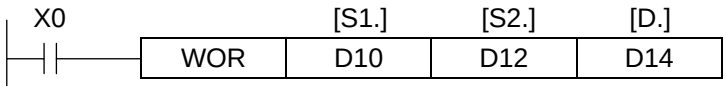
K.H.	KnX	KnY	KnM	KnS	T	C	D	V,Z

←— [D.] —→

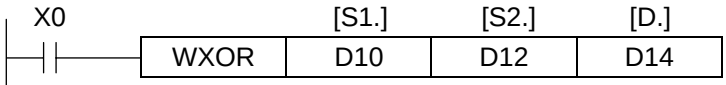
影响旗号:



(D10) ∧ (D12)→(D14)，1∧1=1，1∧0=0，0∧1=0，0∧0=0。



(D10) ∨(D12) →(D14)，1∨1=1，1∨0=1，0∨1=1，0∨0=0。



(D10)⩛(D12)→(D14)，1⩛1=0，1⩛0=1，0⩛1=1，0⩛0=0。

◎ 补码 NEGATION

FNC(29)			16 bits: NEG & NEG(P) - - - - - - - - - - - - - - - - - - - 3 Steps
D	NEG	P	32 bits: (D)NEG & (D)NEG(P) - - - - - - - - - - - - - - -5 Steps

		J1n	J2n--

K.H.	KnX	KnY	KnM	KnS	T	C	D	V,Z

Operands: ←——— [D.] ———→

影响旗号 ：

X0 [D.]

NEG D10

/(D10)+1➔(D10)

♦ [D.]所指定的内容值每一位被反相(“1”→”0”，“0”→”1”)后，再加”1”存入[D.]。

♦ 此指令可对负数取绝对值。

<<应用回路>> 取负数的绝对值

M8000

BON D10 M0 K15

M0

NEG P D10

<<负数及绝对值的表示法>>

(D 10)=2

0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0

(D 10)=1

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

(D 10)=0

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

(D 10)= -1 → ($\overline{\text{D 10}}$)+1=1

1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

→

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

(D 10)= -2 → ($\overline{\text{D 10}}$)+1=2

1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0

→

0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0

(D 10)= -32,765 → ($\overline{\text{D 10}}$)+1= 32,765

1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1

→

0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

(D 10)= -32,766 → ($\overline{\text{D 10}}$)+1= 32,766

1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0

→

0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0

(D 10)= -32,767 → ($\overline{\text{D 10}}$)+1= 32,767

1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

→

0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

(D 10)= -32,768 → ($\overline{\text{D 10}}$)+1= -32,768

1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

→

1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

◎ 右回旋 ROTATION RIGHT

FNC(30)			16 bits: ROR & ROR(P) - - - - - - - - - - - - - - - - - - - 5 Steps
D	ROR	P	32 bits: (D)ROR & (D)ROR(P) - - - - - - - - - - - - - - -9 Steps

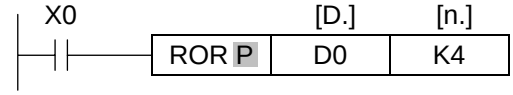
		J1n	J2n--

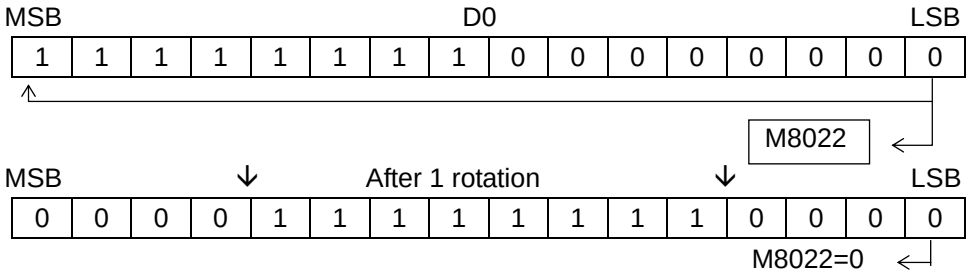
	K.H.	KnX	KnY	KnM	KnS	T	C	D	V,Z
Operands:	← n →		←			[D.]			→

16bit : n ≤ 16

32bit : n ≤ 32

影响旗号: M8022





♦ 右旋后的最右位被存入进位旗标。

◎ 左回旋 ROTATION LEFT

FNC(31)			16 bits: ROL & ROL(P) - - - - - - - - - - - - - - - - - - - 5 Steps
D	ROL	P	32 bits: (D)ROL & (D)ROL(P) - - - - - - - - - - - - - - -9 Steps

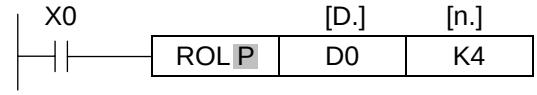
		J1n	J2n--

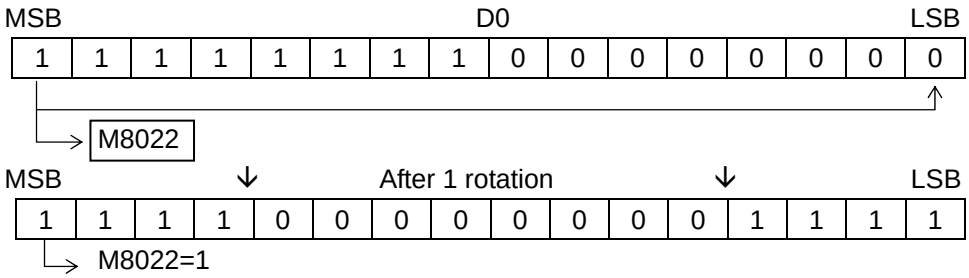
	K.H.	KnX	KnY	KnM	KnS	T	C	D	V,Z
Operands:	← n →		←			[D.]			→

16bit : n ≤ 16

32bit : n ≤ 32

影响旗号:





♦ 左旋后的最左位被存入进位旗标。

◎ 右回旋含进位旗号 ROTATION RIGHT WITH CARRY

FNC(32)			16 bits: RCR & RCR(P) - - - - - - - - - - - - - - - - - - - 5 Steps
D	RCR	P	32 bits: (D)RCR & (D)RCR(P) - - - - - - - - - - - - - 9 Steps

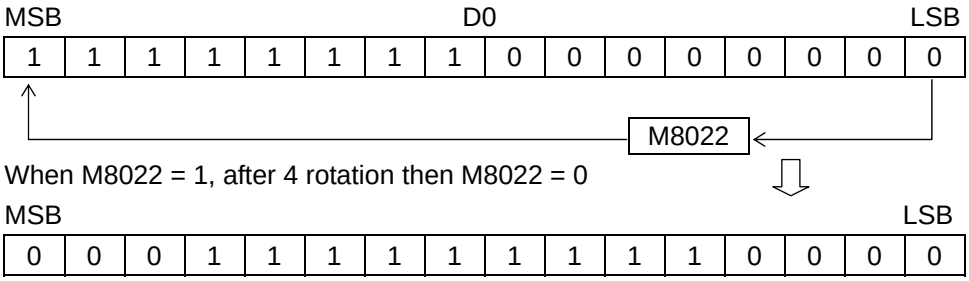
		J1n	**J2n--**

	K.H.	KnX	KnY	KnM	KnS	T	C	D	V,Z
Operands:	← n →		←			[D.]			→

16bit : n ≤ 16

32bit : n ≤ 32

影响旗号:



◎ 左回旋含进位旗号 ROTATION LEFT WITH CARRY

FNC(33)			16 bits: RCL & RCL(P) - - - - - - - - - - - - - - - - - - - 5 Steps
D	RCL	P	32 bits: (D)RCL & (D)RCL(P) - - - - - - - - - - - - - - 9 Steps

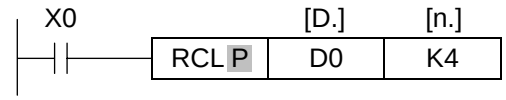
		J1n	**J2n--**

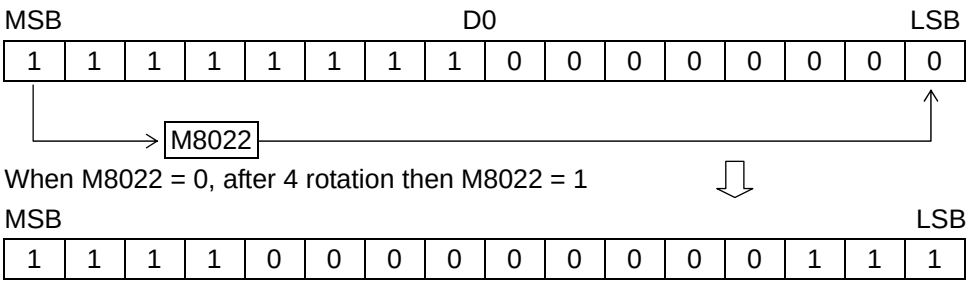
	K.H.	KnX	KnY	KnM	KnS	T	C	D	V,Z
Operands:	← n →		←			[D.]			→

16bit : n ≤ 16

32bit : n ≤ 32

影响旗号:





◎ 位右位移 SHIFT RIGHT

FNC(34)			16 bits: SFTR & SFTR(P) - - - - - - - - - - - - - - - - - - 9 steps
	SFTR	P	

		J1n	J2n--

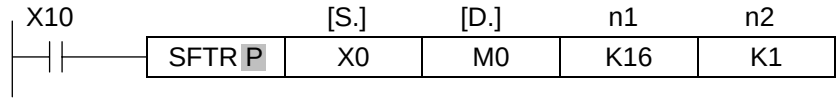
◎ 位左位移 SHIFT LEFT

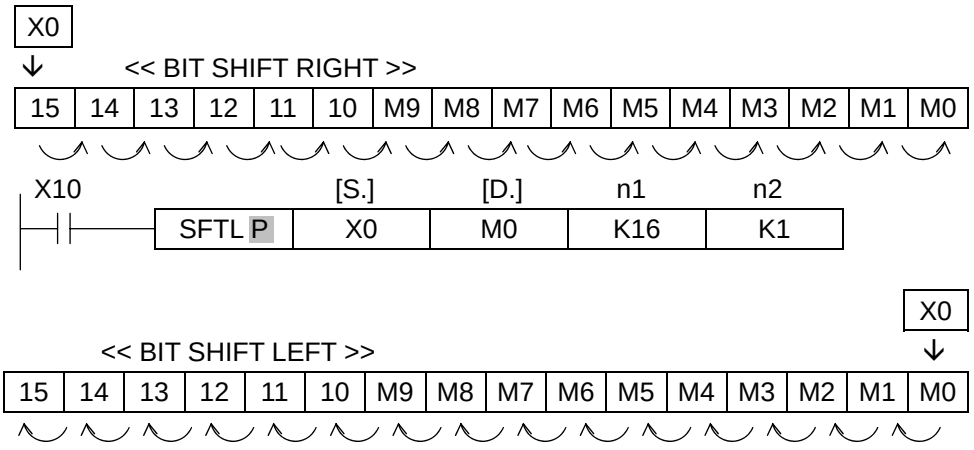
FNC(35)			16 bits: SFTL & SFTL(P) - - - - - - - - - - - - - - - - - - 9 steps
	SFTL	P	

		J1n	J2n--

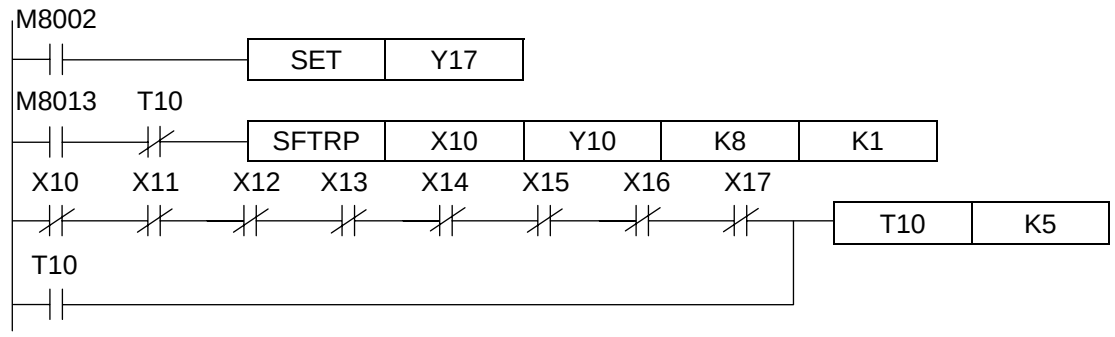
	K.H.		X	Y	M	S
Operands:	← n →	n2 ≤ n1 ≤ 256		← [D.] →		
			← [S.] →			

影响旗号:





范例 I/O 测试: 接线 X10 ↔Y10 … X17 ↔Y17



◎ 字符右位移 WORD SHIFT RIGHT

FNC(36)			16 bits: WSFR & WSFR(P) - - - - - - - - - - - - - - - - - 9 steps
	WSFR	P	

		J1n	J2n--

◎ 字符左位移 WORD SHIFT LEFT

FNC(37)			16 bits: WSFL & WSFL(P) - - - - - - - - - - - - - - - - - -9 steps
	WSFL	P	

		J1n	J2n--

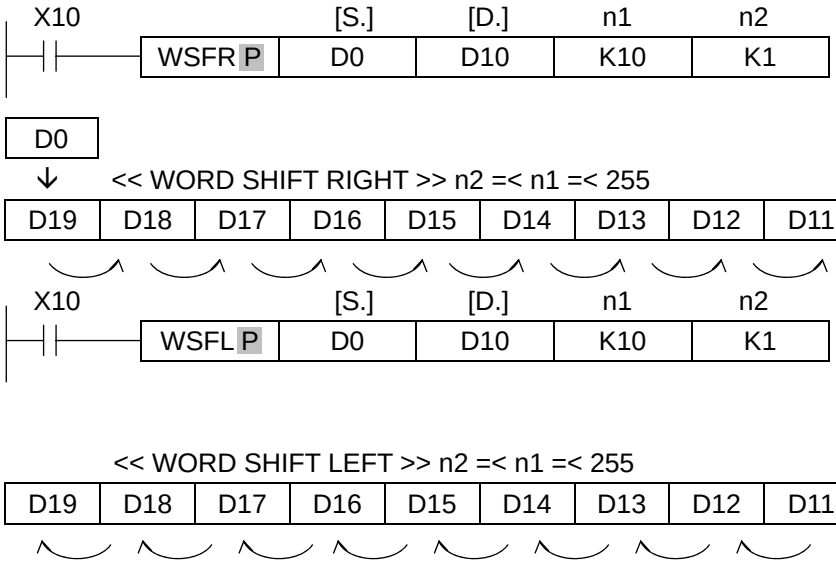
Operands: [S.]

K.H.	KnX	KnY	KnM	KnS	T	C	D	V,Z

n [D.]

n2 ≤ n1 ≤ 256

影响旗号:



◎ 移位缓存器写入 SHIFT REGISTER WRITE

FNC(38)			16 bits: SFWR & SFWR(P) - - - - - - - - - - - - - - - - - 7 Steps
	SFWR	P	

		J1n	**J2n--**

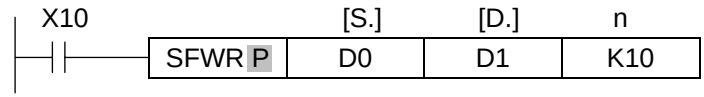
Operands: [S.]

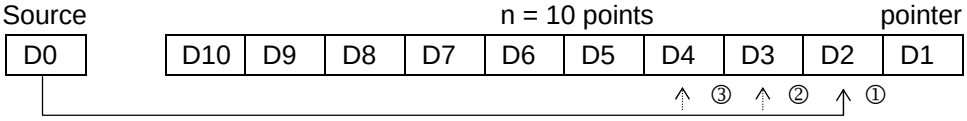
K.H.	KnX	KnY	KnM	KnS	T	C	D	V,Z

n [D.]

$2 \le n \le 256$

影响旗号:





- 当 X10 OFF → ON，来源 D0 的内容存入 D2 且堆栈指针 D1="1"，当下一个 OFF → ON 的脉波，D0 的内容存入 D3 且堆栈指针 D1="2"，[S.] 的内容依序存入目的缓存器且堆栈指针 [D.] 的内容自动加"1"。
- 假如 [D.] 的内容超过"n-1" (n 是 FIFO 堆栈长度)时，则无法处理且进位旗标 M8022 ON。

◎ 移位缓存器读取 SHIFT REGISTER READ

FNC(39)			16 bits: SFRD & SFRD(P) - - - - - - - - - - - - - - - - - 7 Steps
	SFRD	P	

		J1n	**J2n--**

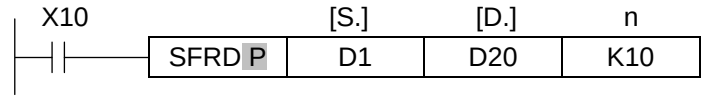
Operands: [S.]

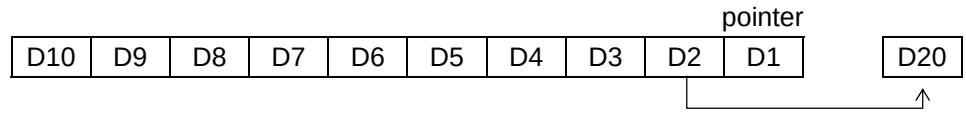
K.H.	KnX	KnY	KnM	KnS	T	C	D	V,Z

n [D.]

$2 \le n \le 256$

影响旗号:





- 当 X10 OFF → ON，D2 的内容存入 D20 且堆栈指针 D1 的内容自动减"1"。
- 当 [S.] = "0"， i.e. FIFO 堆栈无数据，零位旗标 M8020 ON。
- 数据一律由 [S.]+1 处读出，且 D10 的内容保持不变。

◎ 区域复置 ZONE RESET

FNC(40)			16 bits: ZRST(P) - 5 steps
	ZRST	P	

		J1n	J2n--

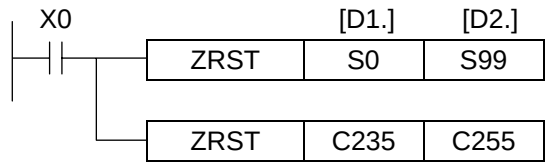
Operands: ←— [D1.][D2.] —→

K.H.	KnX	KnY	KnM	KnS	T	C	D	V,Z

Operands: ←— [D1.] [D2.] —→

X	Y	M	S

影响旗号:



♦ 清除 [D1.] 与 [D2.] 所指定范围的数据，对字符要素而言内容值=”0”，对位要素而言位状态 OFF。

♦ [D1.] 与 [D2.]须指定相同的要素且 [D1.] ≤ [D2.] 。

♦ 假如 [D1.] > [D2.]，则仅 [D1.] 被清除。

◎ 解碼 DECODE

FNC(41)			16 bits: DECO(P) - 7 steps
	DECO	P	

		J1n	J2n--

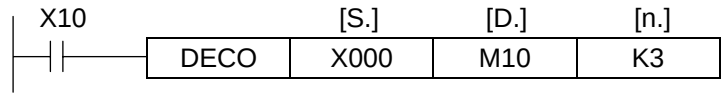
Operands: ←———— [S.] ————→

X	Y	M	S	K.H	T	C	D	V,Z

←— [D.] —→⇩ ←— [D.] —→

n = 1 – 8

影响旗号:



♦ [D.] 所指定的要素若为 T, C 或 D 时，则 n ≤ 4。

♦ 若来源全部为“0”，则 M10 被设为“1”。

◎ 编码 ENCODE

FNC(42)			16 bits: ENCO(P) - 7 steps
	ENCO	P	

		J1n	J2n--

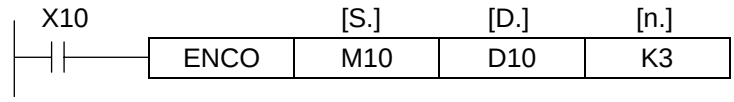
Operands:

X	Y	M	S	K.H	T	C	D	V,Z

[S.] [D.]

n = 1 – 8

影响旗号:



- ♦ [S.] 所指定的要素若为 T,C 或 D 时，则 n ≤ 4。
- ♦ 若来源要素中“1”的位超过 1 个时，仅最低位的“1”视为有效。
- ♦ 若来源要素的每一位都为“0”，则将发生错误。

◎ 和 SUM

FNC(43)			16 bits: SUM(P) - 5 steps
D	SUM	P	32 bits: (D)SUM(P) - 9 steps

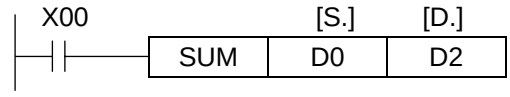
		J1n	J2n--

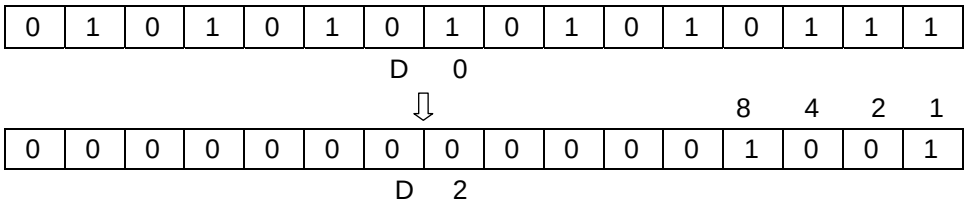
Operands:

K.H.	KnX	KnY	KnM	KnS	T	C	D	V,Z

[S.] [D.]

影响旗号:





- ♦ [S.] 资料中”1”的个数存入[D.]中。
- ♦ 若[S.]中任一位都不为“0”，则零位旗标 M8020 ON。

◎ 位检查 BIT ON CHECK

FNC(44)			16 bits: BON(P) - 7 steps
D	BON	P	32 bits: (D)BON(P) - 13 steps

		J1n	J2n--

Operands: ←——— [S.] ———→

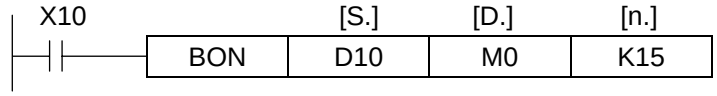
K.H.	KnX	KnY	KnM	KnS	T	C	D	V,Z

← n → [n.] = 0 -15 或 0 - 31

Operands: ←——— [D.] ———→

X	Y	M	S

影响旗号:



0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

Bit15, D10=0, then M0 = OFF LSB

1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0

Bit15, D10=1, then M0 = ON LSB

◎ 平均值 MEAN

FNC(45)			16 bits: MEAN(P) - 7 steps
	MEAN	P	

		J1n	J2n--

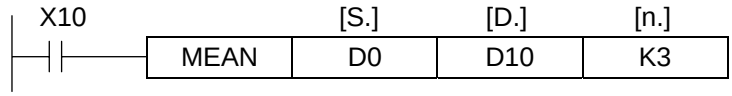
Operands: ←——— [S.] ———→

K.H.	KnX	KnY	KnM	KnS	T	C	D	V,Z

Operands: ← n → ←——— [D.] ———→

[n]=1-64

影响旗号:



[(D0)+(D1)+(D2)] / 3 → (D10)

◎ 故障指示器设定 ANNUNCIATOR SET

FNC(46)			16 bits: ANS - 7 steps
	ANS		

Reserved

◎ 故障指示器复置 ANNUNCIATOR RESET

FNC(47)			16 bits: ANR(P) - 1 steps
	ANR		

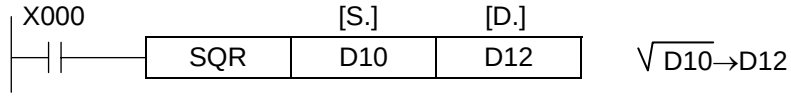
Reserved

◎ 开平方根 SQUARE ROOT

FNC(48)			16 bits: SQR(P) - 5 steps
D	SQR	P	32 bits: (D)SQR(P) -9 steps

Operands: ←→	[S.]						[S.] ←→		
	K.H.	KnX	KnY	KnM	KnS	T	C	D	V,Z
Operands:							[D.] ←→		

影响旗号: M8020, M8021, M8022



♦[S.]之内容必须为正数，若为负数时，错误旗号 M8067 将会动作，且命令不会执行。

♦演算结果若有小数点将被舍去，若结果未满 1 而被舍去时，借位旗号 M8021 将会动作。

♦演算结果正好为 0 时，零旗号 M8020 将会动作。

◎ 浮动小数点 FLOAT

FNC(49)			16 bits: FLT(P) -5 steps
D	FLT	P	32 bits: (D)FLT(P) -9 steps

			J2n--

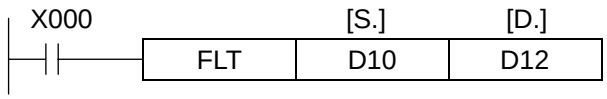
Operands: [S.] ⟷

K.H.	KnX	KnY	KnM	KnS	T	C	D	V,Z

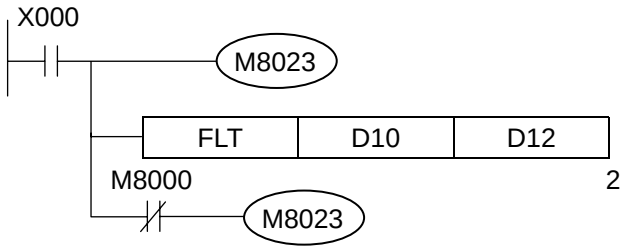
Operands: [D.] ⟷

影响旗号: M8020, M8021, M8022

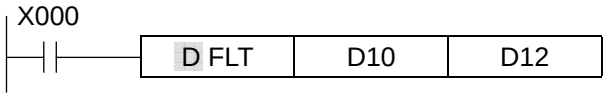
♦ FLT 指令为 BIN 之整数值与 2 进位浮动小数点值间的转换命令，由于常数 K,H 值于浮动小数点运算时会自动转换，因此不适用于此命令。



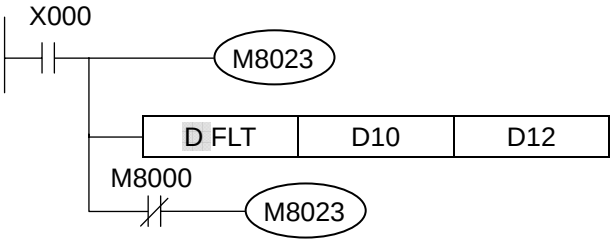
(D10) → (D13、D12)
BIN 整数　　2 进位浮动小数点值



(D11、D10) → (D12)
2 进位浮动小数点值　　BIN 整数值
小数点以下舍去



(D11、D10) → (D13、D12)
BIN 整数值　　2 进位浮动小数点值



(D11、D10) → (D13、D12)
2 进位浮动小数点值　　BIN 整数值
小数点以下舍去

♦ M8023 = ON 时，执行 2 进位浮动小数点值→BIN 整数值之转换。
M8023 = OFF 时，则执行反方向之转换。

♦ 2 进位浮动小数点值→BIN 整数值之转换结果若未满 1，而造成被舍去为 0 或产生溢位时，M8021 / M8022 将分别 ON，结果若正好为 0 时，M8020 将会 ON。

◎ 输出入更新 REFRESH

FNC(50)			16 bits: REF(P) - 5 steps
	REF	P	

		J1n	J2n--

Operands:

K.H.	KnX	KnY	KnM	KnS	T	C	D	V,Z

←n→

Operands:

X	Y	M	S

←— [D.] —→

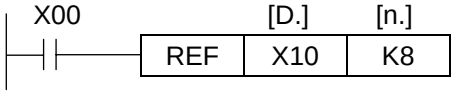
[D.] X/Y 的号码应指定为 10 的倍数，X00, X10,……

[n.] K/H 的号码应指定为 8 的倍数，i.e. 8,16,24,….

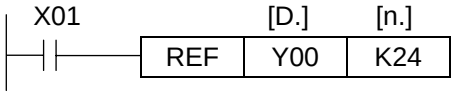
影响旗号:

♦ PLC 输入全部更新均在程序 STEP 0 执行前，而输出均在 END 或 FEND 命令之后执行，在演算过程中不变更。若演算过程中需实时的输入数据或输出演算的结果，则须用输出入更新命令。

<< 输入更新 >> 仅 X10 – X17 被更新



<< 输出更新 >> Y00-Y07，Y10-Y17，Y20-Y27 被更新。



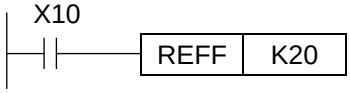
♦ 中断程序内不可使用 FNC(50) REF 指令。

◎ 更新及时间滤波器 REFRESH AND FILTER ADJUSTMENT

FNC(51)			16 bits: REFF(P) - 3 steps
	REFF	P	

			J2n--

Operand: [n.] = 0 - 60



♦ 为防止不要噪声干扰，一般 PLC 输入继电器都有硬件 RC 滤波器的设计，及可用来调整软件滤波器的时间。

♦ 此命令只变更 X00-X07 软件滤波时间，即 D8020 的内容。若须变更其它输入点的滤波时间，请利用 MOV 指令。

◎ 数组 MATRIX

FNC(52)			16 bits: MTR - 9 Steps
	MTR		

		J1n	J2n--

Operands:

K.H.	KnX	KnY	KnM	KnS	T	C	D	V,Z

|← n →|

Operands |←[S.]→|<[D1.]>|

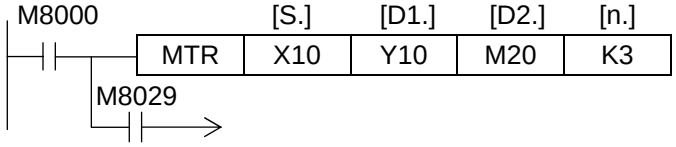
X	Y	M	S

|←—— [D2.] ——→|

(S.): X00, X10, X20, X30 - - - - - - -X160, X170。

(D1.): Y00, Y10, Y20, Y30 - - - - - -Y160, Y170。 (D2.): Y, M, S multiple of 10, i.e. 00, 10, 20 etc。

(n.): K, H n=2~8。

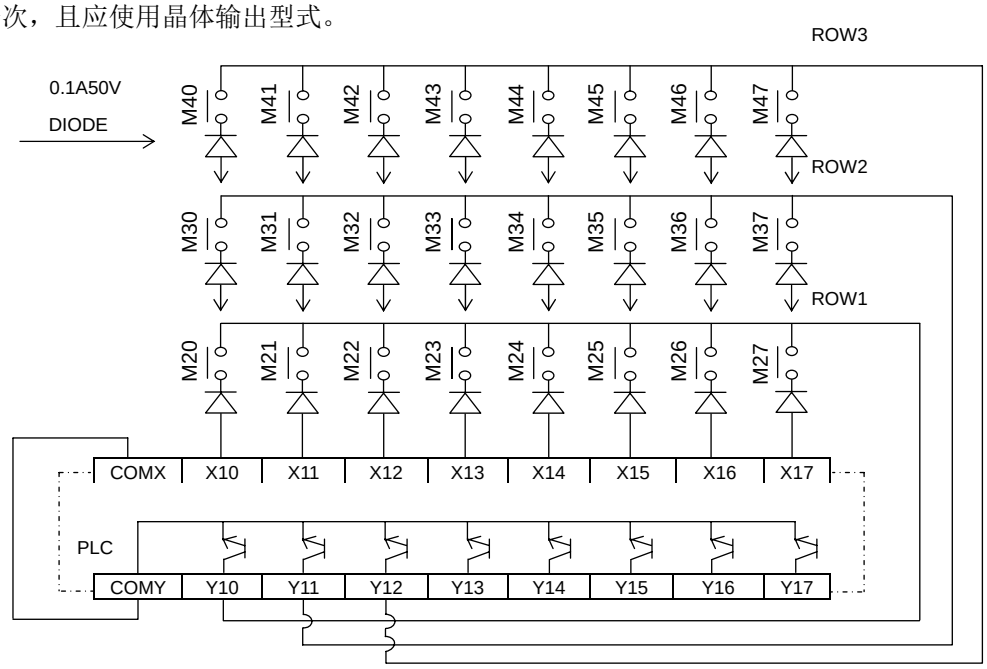


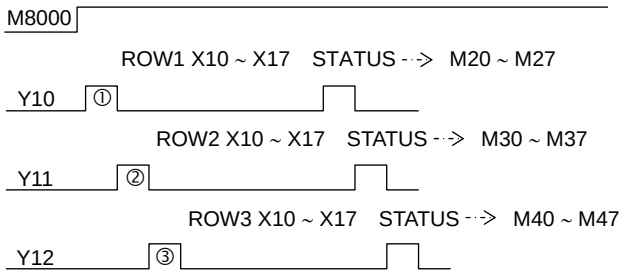
♦ MTR 指令用来读取 8 点 n 列的输入信号(8 个输入点，n 个输出点)。8 个输入点的起始由(S.)指定，n 个输出点的起

始由(D1.)指定，如例所示，输出点 Y10, Y11, Y12 依序且重复为 ON，则第一列、第二列、第三列的输入点被依序且重复的读入，并存放在 M10~M17，M20~M27，M30~M37 中。 ♦ (D2.)为数组的起始地址。

♦ 当命令执行完毕，完成旗号 M8029 设为 ON。当命令再度被执行时，完成旗号 M8029 自动复置。

♦ 此命令只能使用一次，且应使用晶体输出型式。





◎ 高速计数器设定 SET BY HIGH SPEED COUNTER

FNC(53)			32 bits: HSCS - 13 Steps
D	HSCS		

		J1n	J2n--

Operands: ←——— [S1.] ———→

K.H.	KnX	KnY	KnM	KnS	T	C	D	V,Z

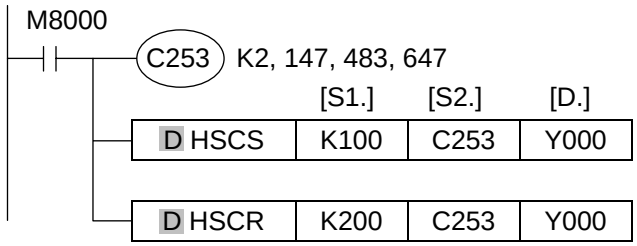
←——→ [S2.] : C235~C255

Operands:

X	Y	M	S

←——— [D.] ———→ [D.]时，I010~I060 中断可使用索引来指定。

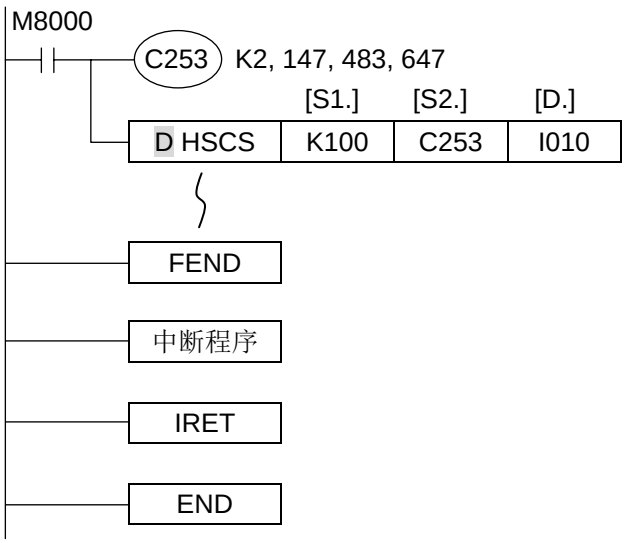
影响旗号:



♦ 使用 FNC53 命令时，外部输出动作以中断来处理，C253 的现在值 99→100 和 101→100 变化时，Y000 被 SET。当 C253 的现在值 199→200 和 201→200 变化时，Y000 OFF。

♦ 此命令为 32 位之专用命令，请输入 D HSCS 命令。

♦ FNC53, FNC54, FNC55 只可以使用一次。



♦ D HSCS 命令的 [D.] 可指定 I0 □ 0 = (□=1~6)(□=1~6 不可重复使用编号)

♦因此 [S2.] 所指定的高速计数器的现在值与 [S1.] 所指定的值相同时，中断主程序，立即跳往指针 I0 □ 0 中断程序执行。

♦特殊辅助继电器 M8059 ON 时，I010~I060 中断全部禁止。

◎ 高速计数器复置 RESET BY HSC

FNC(54)		
D	HSCR	

32 bits: HSCR - 13 Steps

		J1n	J2n--

Operands: [S1.]

K.H.	KnX	KnY	KnM	KnS	T	C	D	V,Z

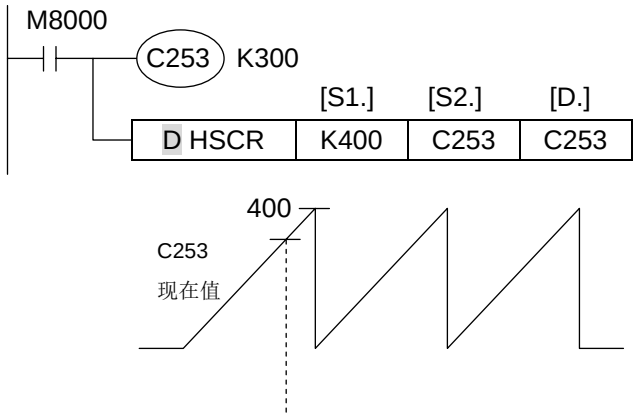
[S2.] : C235~C255

Operands:

X	Y	M	S

[D.] 可指定 [D.] 与 [S2.] 为同一高速计数器。

影响旗号:



♦ C253 的现在值为 400 时，C253 立即清除，现在值变成 0，输出接点不动作。

♦ 此命令为 32 位专用命令，请务必用 D HSCR 命令。

◎ 高速计数器区域比较 ZONE COMPARE FOR HSC

FNC(55)		
D	HSZ	

32 bits: HSZ- 17 Steps

Operands: [S1.][S2.]

K.H.	KnX	KnY	KnM	KnS	T	C	D	V,Z

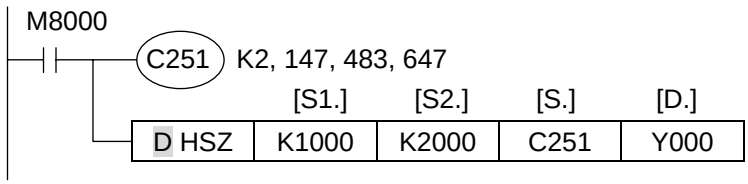
[S.] : C235~C255

Operands:

X	Y	M	S

[D.]

影响旗号:



<比较输入的动作>

K1000>C251 现在值	Y000	ON
K1000≦C251 现在值≦K2000	Y001	ON
K2000<C251 现在值	Y002	ON

♦ 此命令为 32 位命令，请务必用 D HSZ 命令。

♦ [S1.], [S2.]的内容，请依照 [S1.] ≦ [S2.] 规则。

♦ 使用 FNC55 时，外部输出是以中断来处理。输出不受扫描周期的影响而动作。

◎ 速度侦测 SPEED DETECT

FNC(56)			16 bits: SPD - 7 Steps
	SPD		

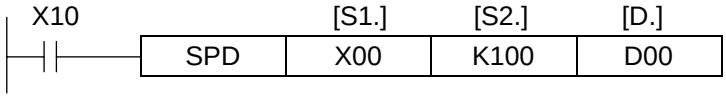
		J1n	J2n--

Operands: (S1.): X000~X005

Operands: ←——— [S2.] ———→

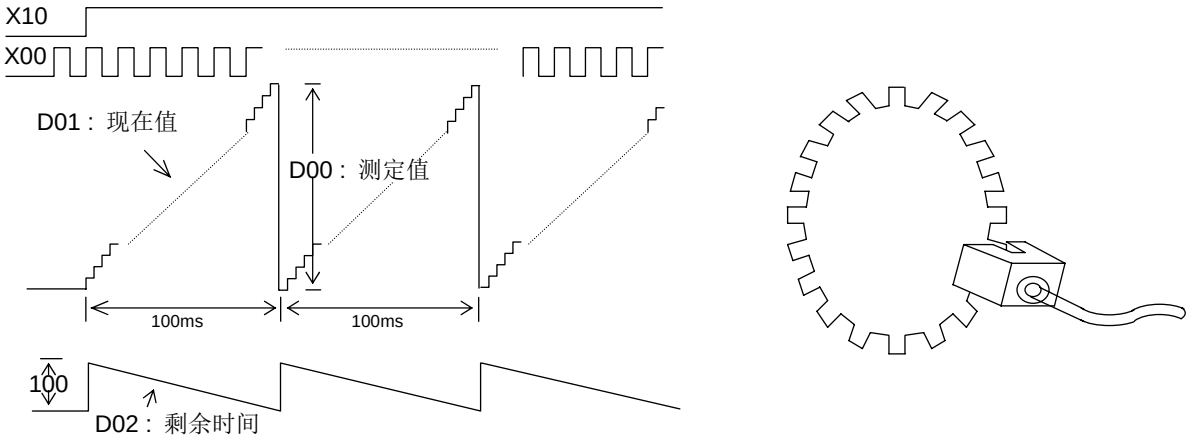
K.H.	KnX	KnY	KnM	KnS	T	C	D	V,Z

影响旗号:M8029 ←——— [D.] ———→



- 输入脉波由 [S1.] 指定，[S2.] 指定计数时间，随时把结果存放在 [D.] 中。
- 本指令[D.]会占用 3 个目的要素。(本例为 D00-D02)
- 本例 D01 计算 X00 (OFF→ON)的次数，100msec 后，把计数结果存入 D00 中，然后 D01 被复置再重新开始计数。
- D02 用来测量剩余时间。
- 指定的时间所计数的脉波量不可超过 65535。
- 脉波密度与 RPM 成比例，下列公式可求得回转数。
  RPM : N = (D00×60) × 1000 / n × t　n =每转脉波数，t =测量时间。
- 输入(X00-X05) ON/OFF 最大频率与单相高速计数器相同。
- SPD 指令所使用的输入点(X00-X05)，不得再作为其它高速处理或是插断信号。
  当 C251 使用时，X00,X01 不可再做为速度侦测点。
- 利用完成旗号 M8029，轻易达到连续测得多笔数据，再求平均值。

(i) 量频率模式



(ii) 量脉波宽度模式

- [S2.] 的内容值为"0"时，只需 1 脉波宽度即可测出速度 pps(pulse/second) 。
- 本例速度值存放在 D01,D00 中。

◎ 脉波输出 PULSE OUTPUT

FNC(57)			32 bits: (D)PLSY- 13 steps
D	PLSY		

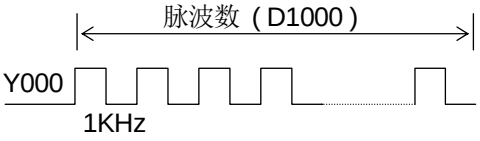
		J1n	J2n--

Operands: [S1.]

K.H.	KnX	KnY	KnM	KnS	T	C	D	V,Z

[D.] : Y00 – Y03　　[S2.]

影响旗号: M8029



- ♦ 此指令为无加减速的脉波输出。
- ♦ [D.] 指定脉波输出点
  [S1.] 指定输出频率(10~200,000Hz).
  [S2.] 从指定的[S2.]开始会连续占用 100 个 words。在本例中，占用 D1000~D1099。
  [S2.]+1, [S2.]+0 : 输出脉波数　　[S2.]+3, [S2.]+2 : 系统保留
  [S2.]+5, [S2.]+4 : 启始地址　　[S2.]+7, [S2.]+6 : 绝对地址(监视用)
  [S2.]+9, [S2.]+8 : 相对地址(监视用)
- ♦ DPLSY 用以产生一指定脉波数，32 位: 1 ~ 2,147,483,647 个脉波。
- ♦ 若[S2.]+1, [S2.]+0 指定为“0”，则无限制的产生脉波。
- ♦ 固定为 32 位运算。若指定 16 位元运转模式，则产生 error 6509。
- ♦ 脉波导通周期 (duty cycle) 50% ON 50% OFF。
- ♦ 执行过程中，若更动 [S2.]+1, [S2.]+0 的数值不予考虑，须待下一次输出才有效。指令执行完毕 M8029 ON。
- ♦ 此命令只能使用一次，且应使用晶体输出型式。

◎ 脉波宽度调变 PULSE WIDTH MODULATION

FNC(58)			16 bits: PWM - 7 steps
	PWM		

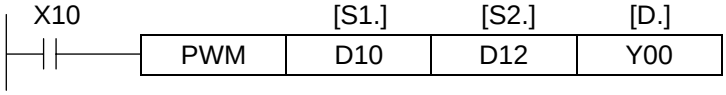
		J1n	J2n--

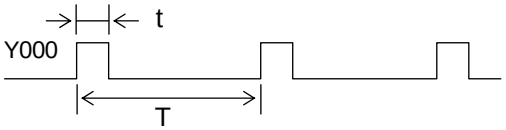
Operands: [S1.][S2.]

K.H.	KnX	KnY	KnM	KnS	T	C	D	V,Z

[D.] : Y00 – Y07

影响旗号: 无





- ♦ [S1.] 指定(t)导通脉波宽度，Y00 - Y01 值范围(0 - 32,767) x 0.01ms ; Y02 - Y07 值范围 (0 - 32,767 msec)
- ♦ [S2.] 指定(T)周期，Y00 - Y01 值范围(0 - 32,767) x 0.01ms ; Y02 - Y07 值范围 (0 – 32,767 msec)
- ♦ [D.] 指定输出点。(以中断方式输出)
- ♦ 若 [S1.] 的数值 > [S2.] 的数值，CPU 判定错误。
- ♦ 此指令外加滤波回路即可仿真为模拟输出。
- ♦ 此命令应使用晶体输出型式。

◎ 付加减速脉波输出 PULSE OUTPUT WITH SLOPE

FNC(59)			
D	PLSR		32 bits: (D)PLSR- 17 steps

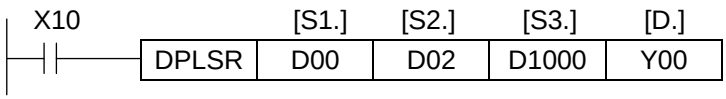
		J1n	J2n--

Operands: ← [S1.] →

K.H.	KnX	KnY	KnM	KnS	T	C	D	V,Z

[D.] : Y00 – Y03 ←→ [S2.][S3.]

影响旗号: M8029



♦ [D.] 指定脉波输出点，若指定 Y04 则为虚拟轴 无实际脉波输出。

[S1.] 指定输出频率。(10 ~ 200,000 pps)

[S2.] 指定输出脉波数。从指定的[S2.]开始会连续占用 8 个 words。本例中，占用 D02~D09。

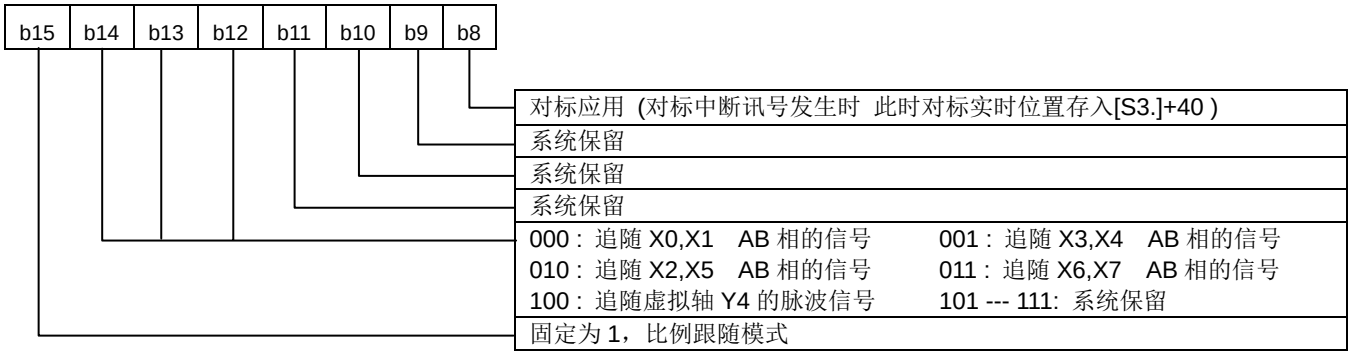
[S3.] 从指定的[S3.]开始会连续占用 100 个 words。在本例中，占用 D1000~D1099。

[S3.]+0 : 运动模式: 命令码 0~99 相当于 G00~G99

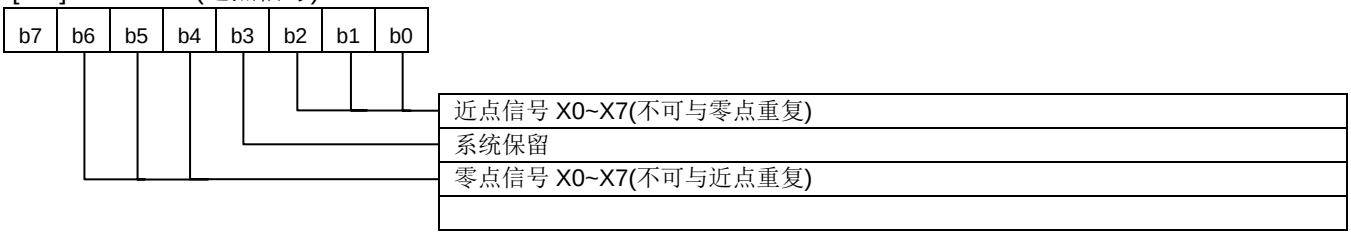
命令码	内 容
00	一段位置运动
01	直线补间(J2nB only) n=2,4
02	圆弧补间 CW (J2nB only) n=2,4
03	圆弧补间 CCW (J2nB only) n=2,4
06	比例跟随(硬件追随):齿轮比必须为分数，分子<分母
28	原点复归

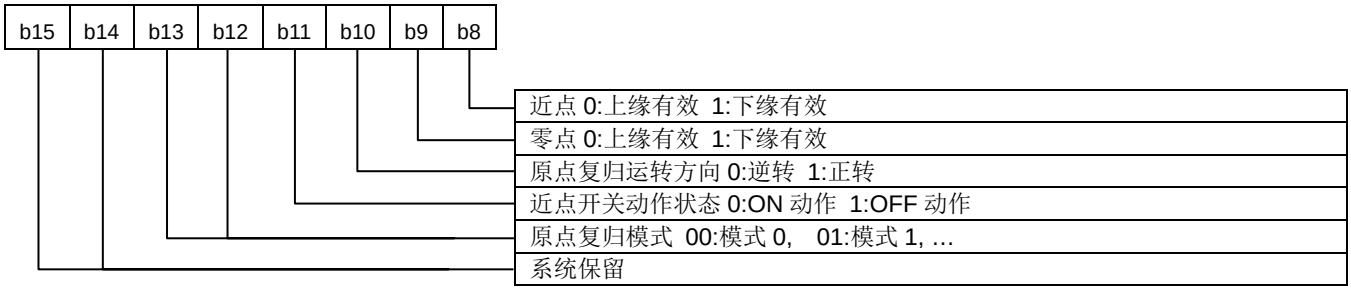
[S3.]+1 : 运转方向控制点: Y02~Y07

[S3.]+2 : 参数设定



[S3.]+3 : 系统保留
[S3.]+5, [S3.]+4 : 启始地址(监视用)
[S3.]+7, [S3.]+6 : 绝对地址(监视用)
[S3.]+9, [S3.]+8 : 相对地址(监视用)
[S3.]+11, [S3.]+10 : 剩余脉波(监视用)
[S3.]+13, [S3.]+12 : 目标地址(监视用)
[S3.]+15, [S3.]+14 : 现在速度(监视用)
[S3.]+17, [S3.]+16 : 最高速度
[S3.]+19, [S3.]+18 : 系统保留
[S3.]+20 : 启始速度(pps)
[S3.]+21 : 系统保留
[S3.]+22 : 加速时间(1ms - 50,000ms)
[S3.]+23 : 减速时间(1ms - 50,000ms)
[S3.]+24 : DOG(近点信号)



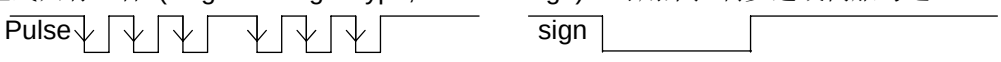


[S3.]+25 : 零点信号设定值。归原点时，若无零点信号(步进马达时)，则将找寻零点次数设为"0"即可。
[S3.]+26 : 零点信号计数值(监视用)
[S3.]+27 : 系统保留
[S3.]+28 : 电子齿轮比(分子)
[S3.]+29 : 电子齿轮比(分母)
[S3.]+30 : 系统保留
[S3.]+32 : 系统保留

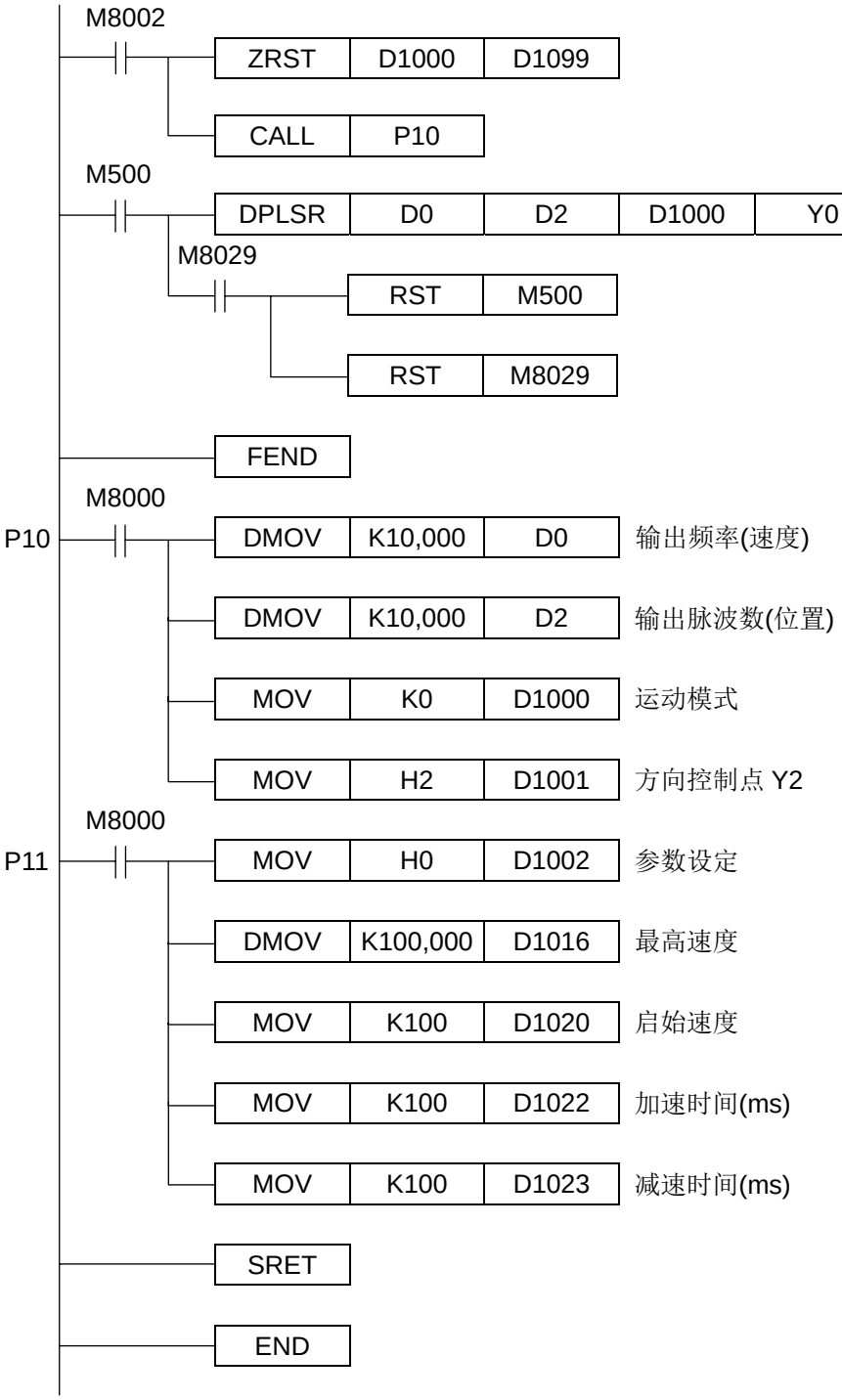
[S3.]+41, [S3.]+40 : PLSR-G00 对标实时位置(输出脉波数)缓存器。
[S3.]+41, [S3.]+40 : PLSV 输出脉波数。数值等于 0 为无目标运转。

♦多轴同动 : 先驱动虚拟轴 其它轴设为[G06]比例跟随模式且追随虚拟轴 Y4 的脉波信号
♦JOG+ JOG- 设定无目标运转旗号，相对位置模式 利用位置正负值 控制运转方向

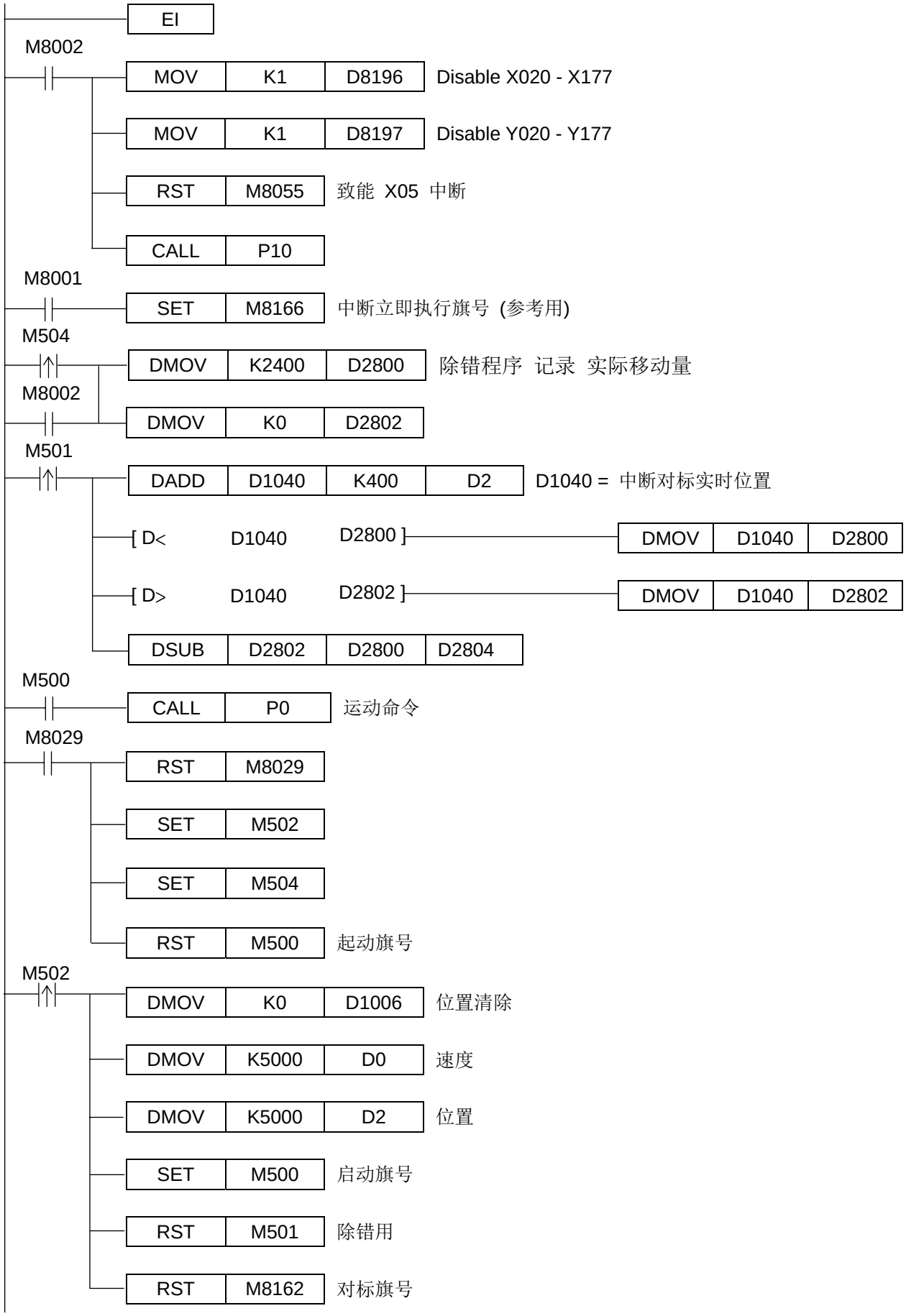
♦ 使用此命令，须先将相对距离或绝对地址换算为脉波数再存入 [S2.] 中。
♦ 脉波输出中，X10 OFF，脉波依停止旗号[S3]+2,b1 的设定状态停止输出。
♦ 脉波导通周期(duty cycle) 50% ON, 50% OFF。
♦ G06 有目标有斜率模式的命令 运转中，变更 [S2.] 的内容无效。
♦ 此命令针对 Y00 或 Y01 只能使用一次(共二次)，且应选择为晶体输出型式。
♦ 固定为 32 位运算。若指定 16 位元运转模式，则产生 error 6509
♦ 此命令脉波输出型式只有一种 (Negative Logic Type, Pulse & Sign)，可用来控制步进或伺服马达。

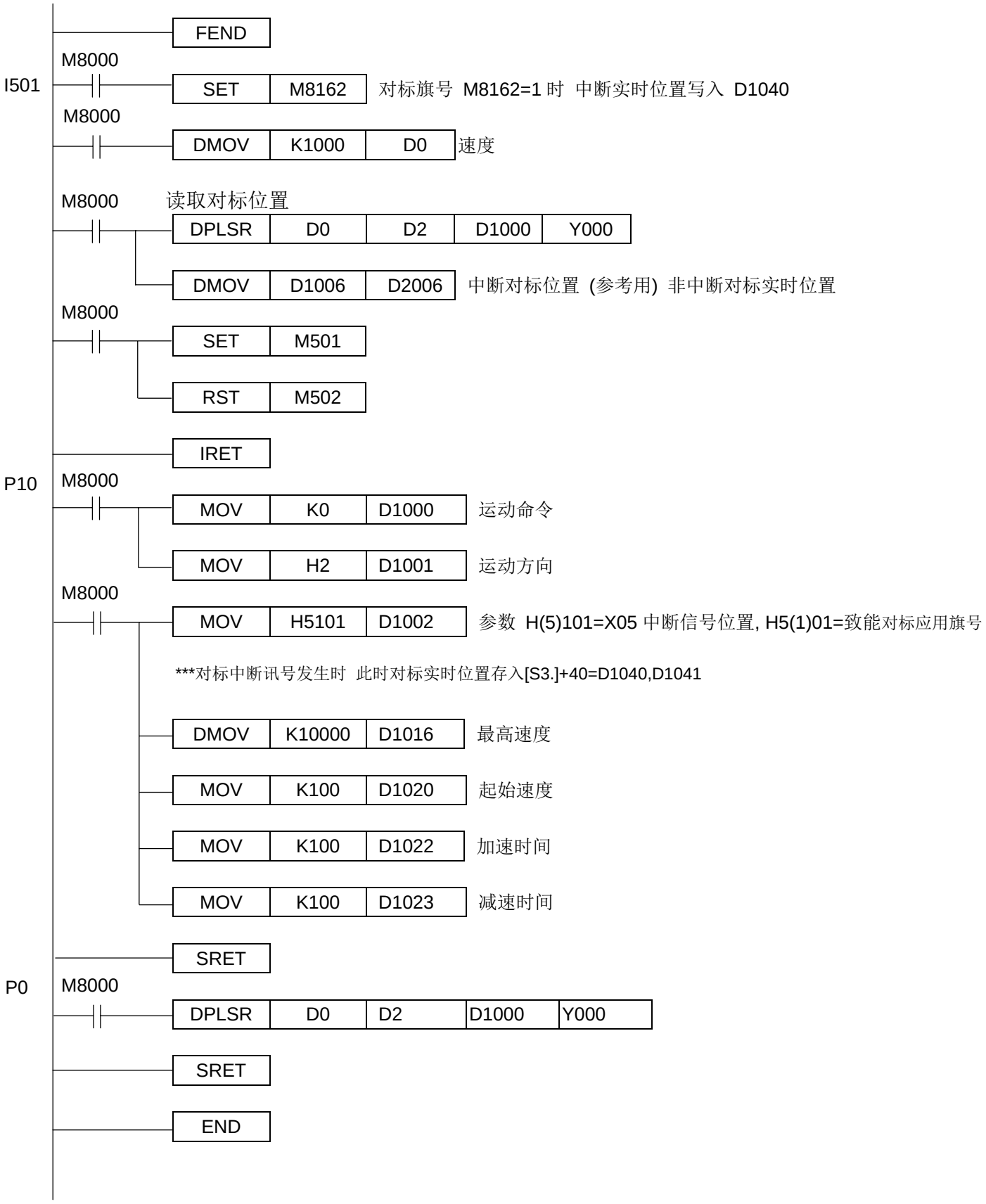


※ 命令码 00 [G00] 一段位置

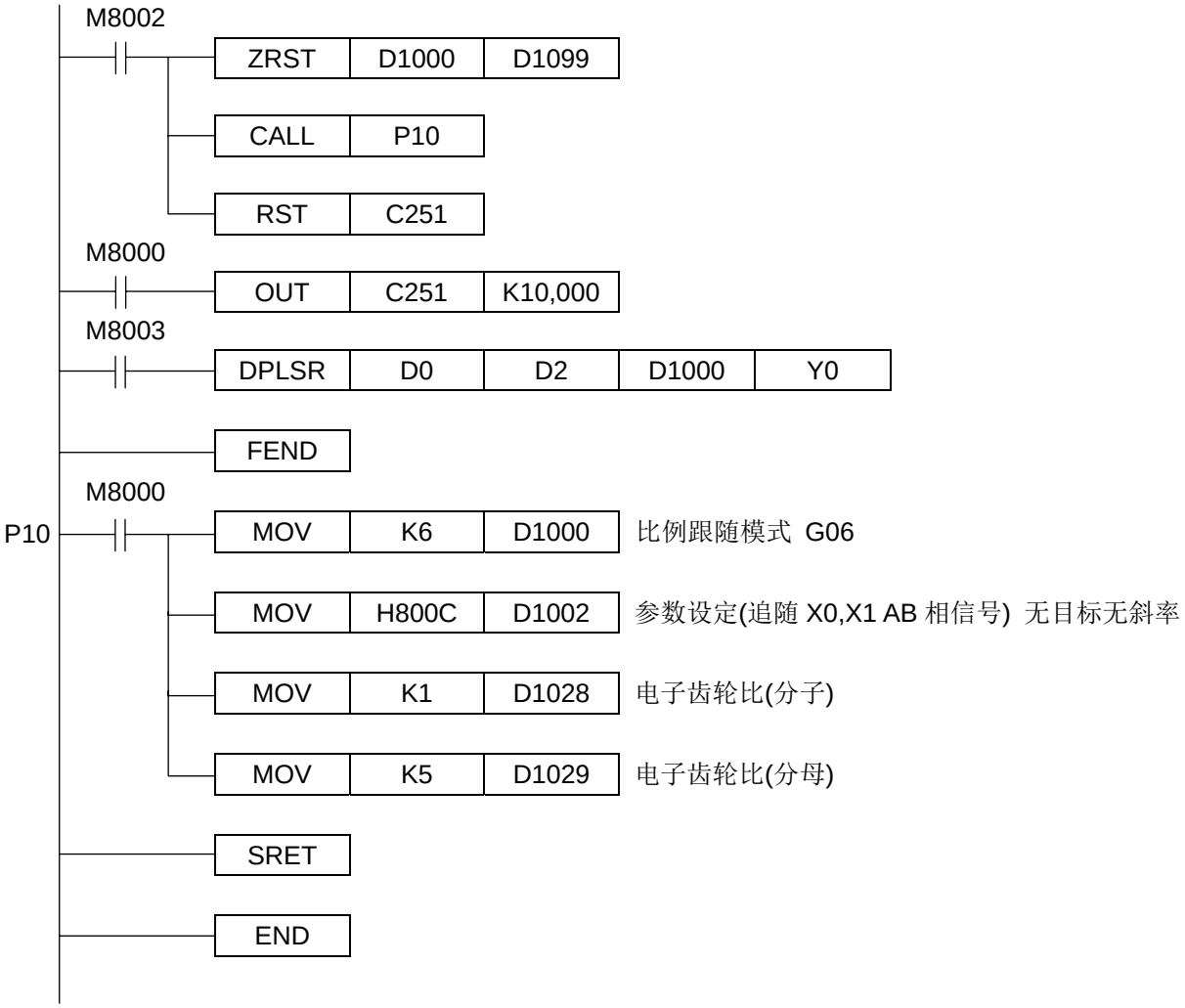


※ 命令码 00 [G00] 对标 变速度变位置

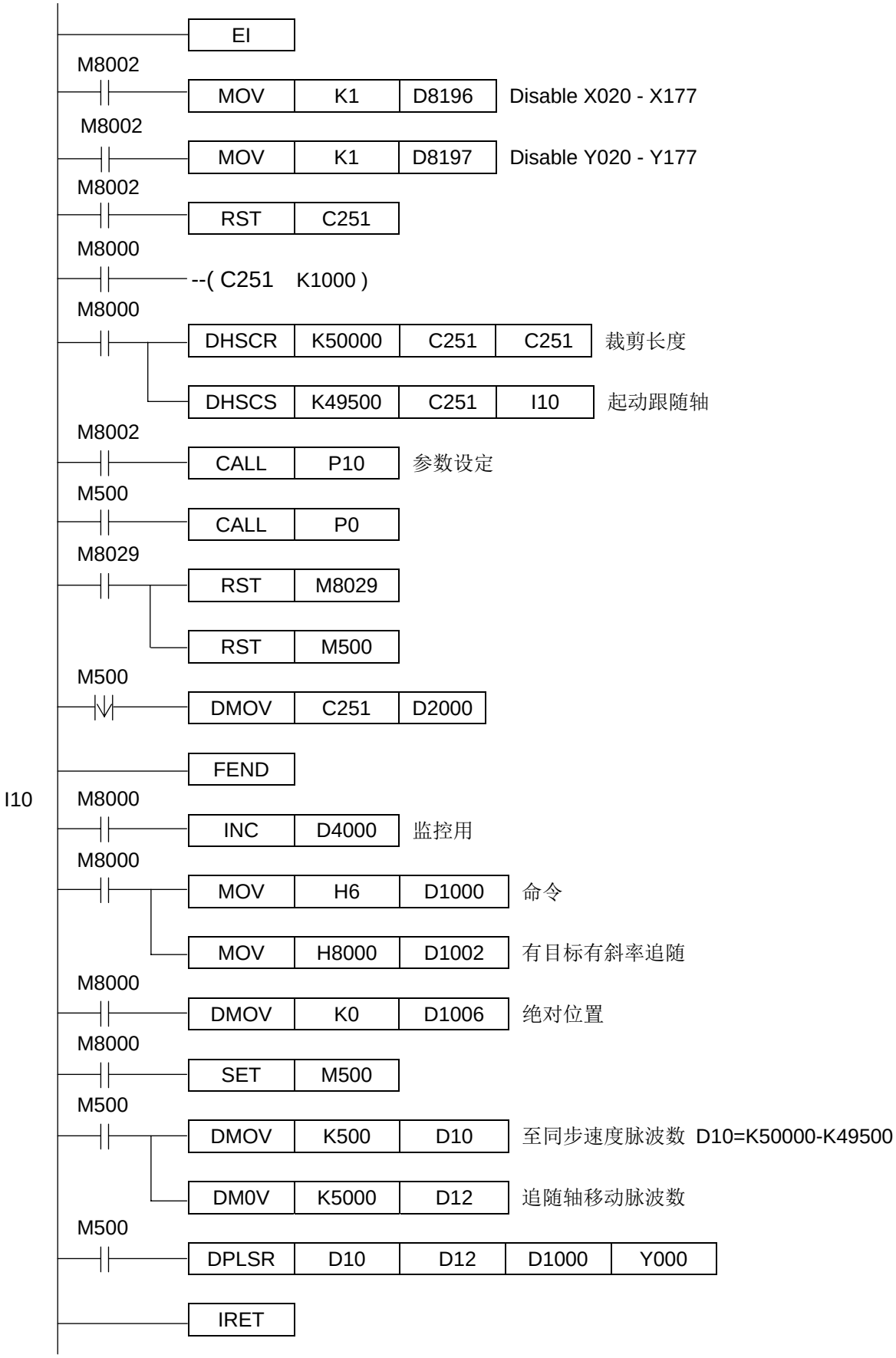


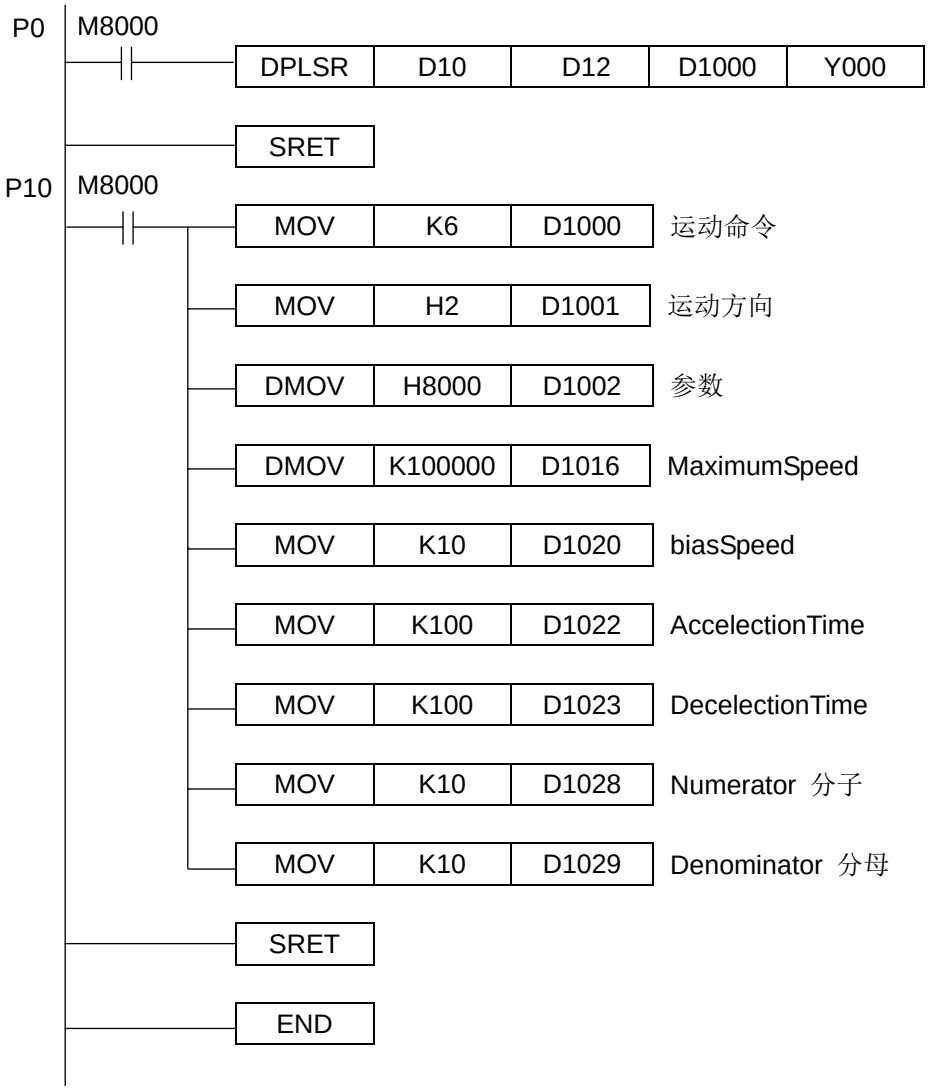

FEND
M8000
I501
SET M8162
对标旗号 M8162=1 时 中断实时位置写入 D1040
M8000
DMOV K1000 D0
速度
M8000
读取对标位置
DPLSR D0 D2 D1000 Y000
DMOV D1006 D2006
中断对标位置 (参考用) 非中断对标实时位置
M8000
SET M501
RST M502
IRET
P10
M8000
MOV K0 D1000
运动命令
MOV H2 D1001
运动方向
M8000
MOV H5101 D1002
参数 H(5)101=X05 中断信号位置, H5(1)01=致能对标应用旗号
***对标中断讯号发生时 此时对标实时位置存入[S3.]+40=D1040,D1041
DMOV K10000 D1016
最高速度
MOV K100 D1020
起始速度
MOV K100 D1022
加速时间
MOV K100 D1023
减速时间
SRET
P0
M8000
DPLSR D0 D2 D1000 Y000
SRET
END

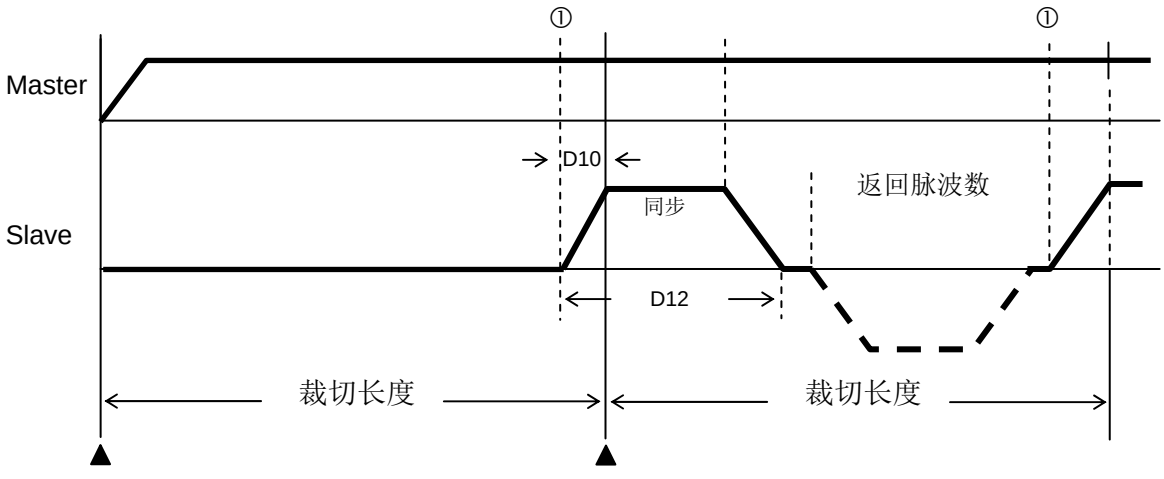
※ 命令码 06 [G06] 比例跟随(Y0 轴方向固定为 Y2，Y1 轴方向固定为 Y3)



※ 命令码 06 [G06] 追剪控制

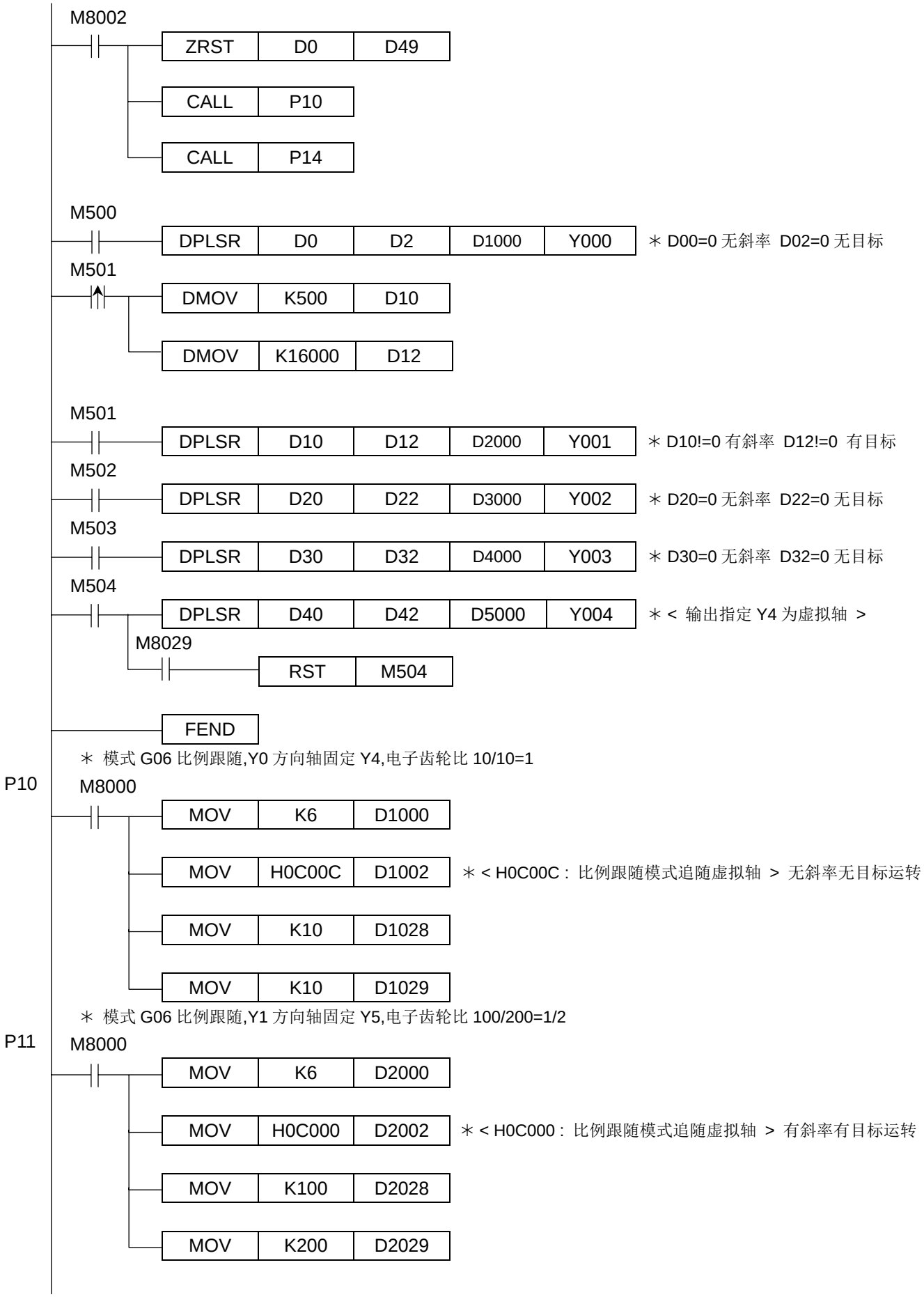




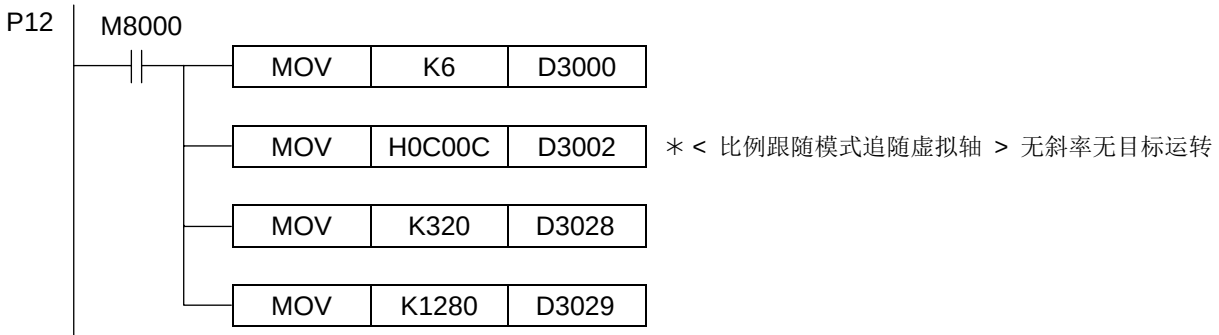


① 启动跟随轴 (当 c251=裁切长度 - 至同步速度脉波数 D10)
②

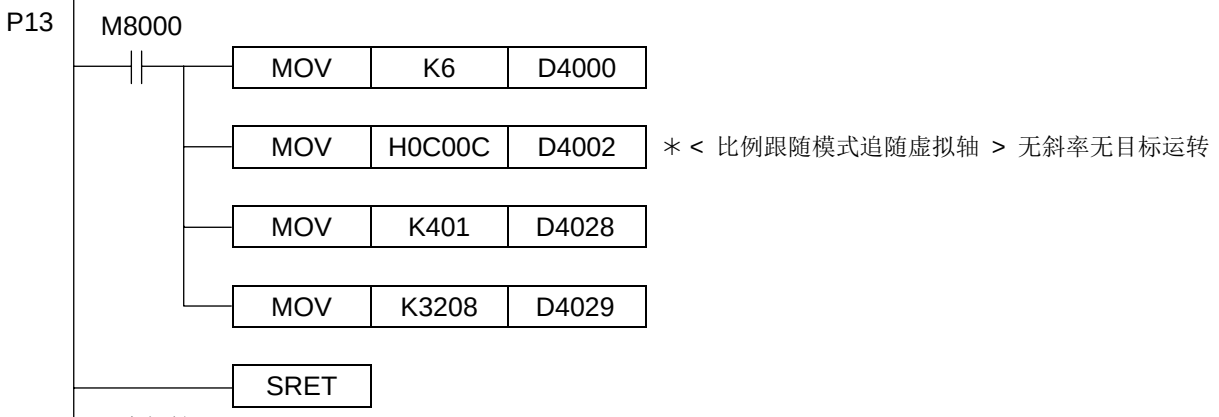
※ 多轴同动 (若虚拟轴参数 bit2,D5002=1 则为无目标运转) 实体轴须先驱动



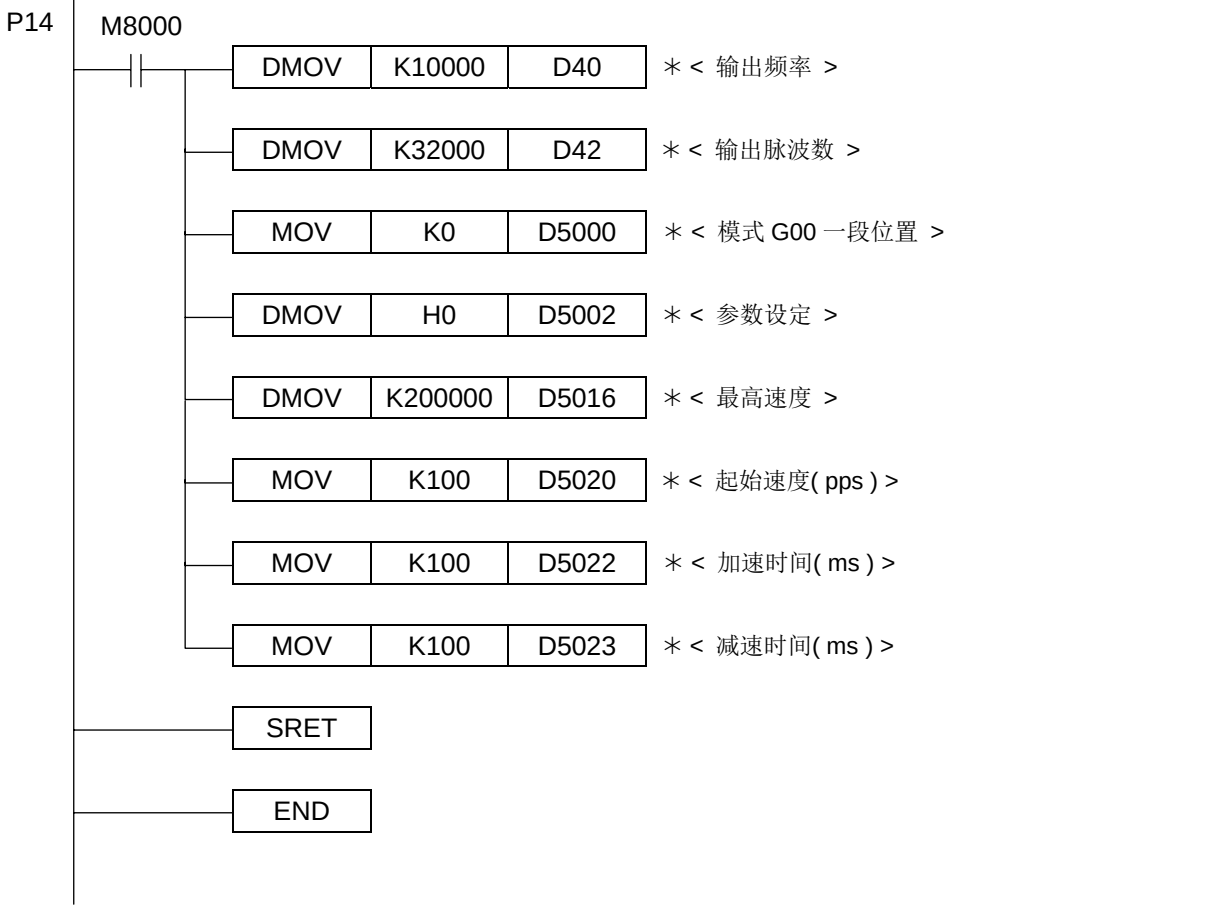
＊ 模式 G06 比例跟随,Y2 方向轴固定 Y6,电子齿轮比 320/1280=1/4



＊ 模式 G06 比例跟随,Y3 方向轴固定 Y7,电子齿轮比 401/3208=1/8



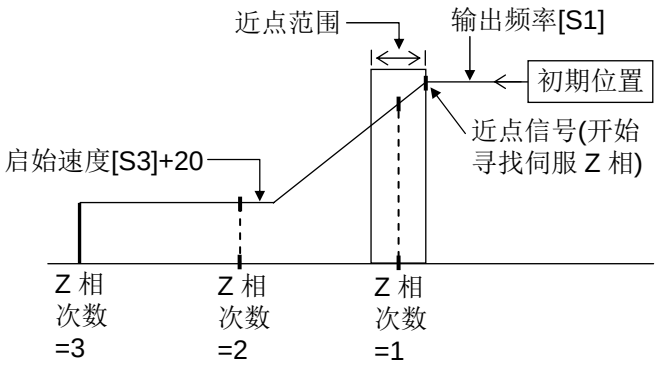
＊ 虚拟轴



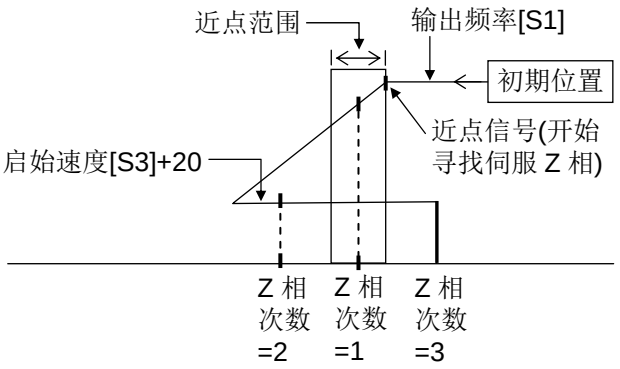
※ 命令码 28 [G28] 零点复归(寻找 Z 相次数不为 0)

```
M8002
 ||  ZRST  D1000  D1099
 ||  CALL  P10
M500
 |↑|  RST   M510    清除零点复归完毕旗号
M500
 ||  DPLSR  D0  D2  D1000  Y0
M8029
 ||  RST   M500
 ||  SET   M510    设定零点复归完毕旗号，本范例使用 M510(使用者可自行指定 M 缓存器)
 ||  RST   M8029
 FEND
M8000
P10 ||  DMOV  K10,000  D0    输出频率(速度)
 ||  DMOV  K10,000  D2    输出脉波数(位置)
 ||  MOV   K28      D1000  原点复归模式
 ||  MOV   H0002    D1001  方向控制点 Y2
 ||  MOV   H0043    D1024  近点信号 X3 上缘有效，零点信号 X4 上缘有效，初期运转方向逆转
 ||  MOV   K3       D1025  设定寻找 Z 相次数，寻找 Z 相速度与启始速度相同
P11 ||  [ ]   与 DPLSR 一段位置范例相同
 SRET
 END
```

< 模式 0 > D1024=H0043 相同方向寻找零点信号



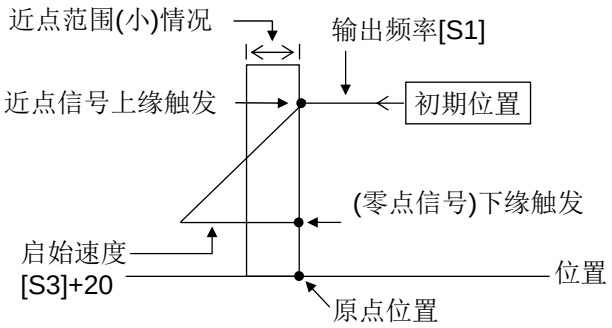
< 模式 1> D1024=H1043 相反方向寻找零点信号

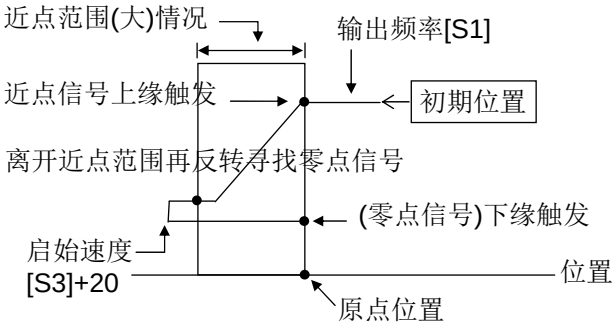


命令码 28 [G28] 零点复归(寻找 Z 相次数为 0，近点信号与零点信号需设定成同一点)
程序代码与零点复归(寻找 Z 相次数不为 0 的模式)相同，只有 D1024,D1025 设定值不同

<< 模式 0 >> 近点确认 减速至起始速度 ***且需离开近点范围*** **立即反转开始寻找零点**

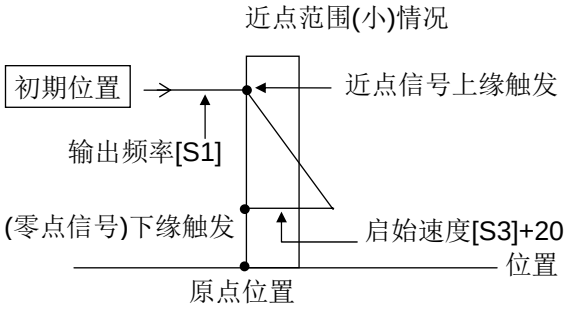
D1024 = H0233 (近点信号 X3 上缘有效，零点信号 X3 下缘有效，初期运转方向逆转)
D1025 = K0 (Z 相次数 = 0)

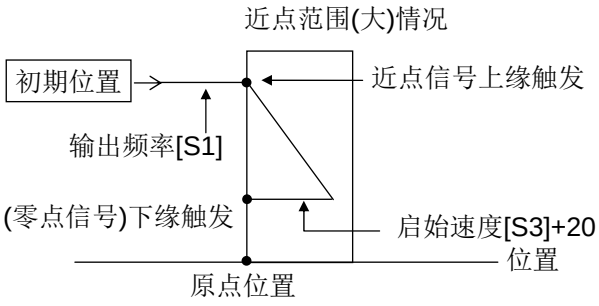




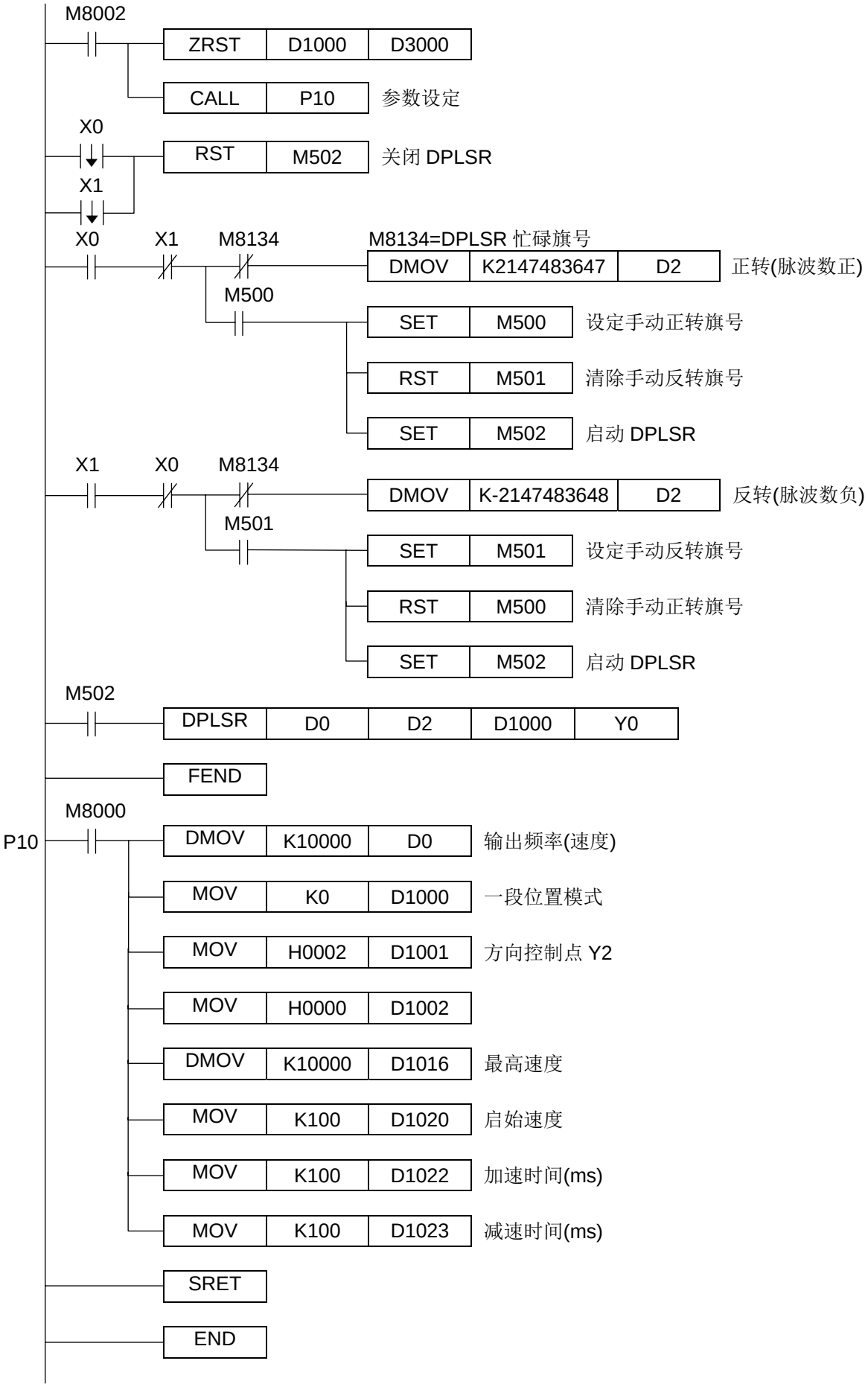
<< 模式 1 >> 近点确认 减速至起始速度 ***不需离开近点范围*** **立即反转开始寻找零点**

D1024 = H1633 (近点信号 X3 上缘有效，零点信号 X3 下缘有效，初期运转方向正转)
D1025 = K0 (Z 相次数 = 0)

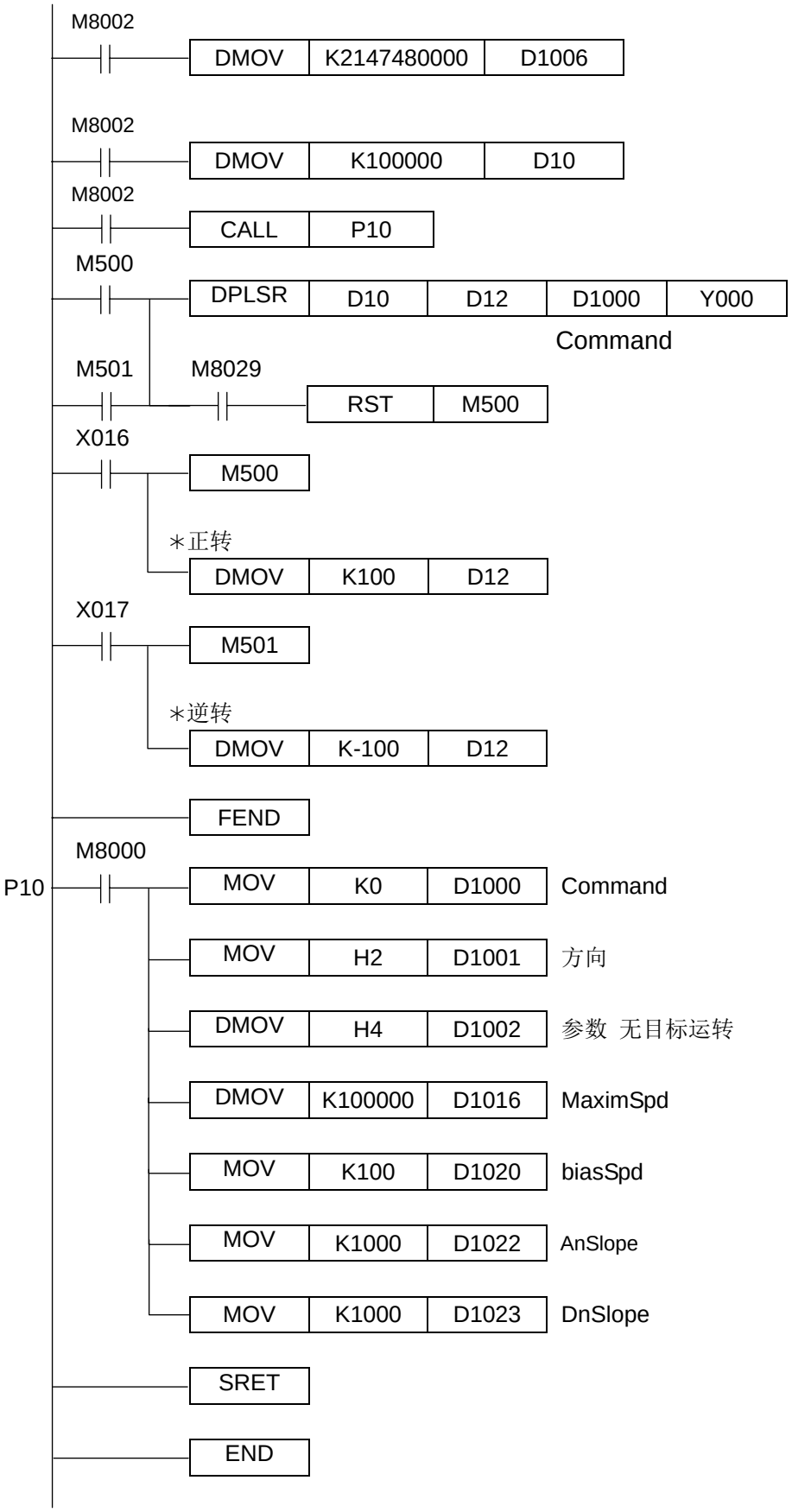




※ DPLSR 范例程式:手动正反转



※ DPLSR 范例程式:手动正反转 (无 Rollover 问题)



初始状态 INITIAL STATE

FNC(60)			16 bits: IST - 7 steps
	IST		

Reserved

◎ 数据搜寻 DATA SEARCH

FNC(61)			16 bits: SER(P)- 9 steps
D	SER	P	32 bits: (D)SER(P) -17 steps

Reserved

◎ 鼓轮控制 ABSOLUTE DRUM SEQUENCE

FNC(62)			16 bits: ABSD - 9 steps
D	ABSD		32 bits: (D)ABSD -17 steps

		J1n	J2n--

Operands: ←——— [S1.] ———→

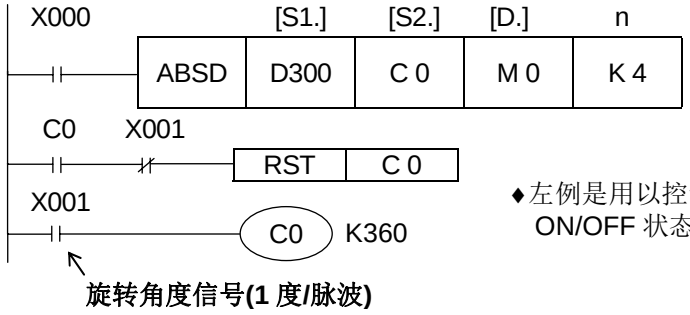
K.H.	KnX	KnY	KnM	KnS	T	C	D	V,Z

← n → n ≤ 64 ←——→ [S2.]

Operands:

X	Y	M	S

←——— [D.] ———→



本命令用以对计数器值产生一多变的输出型式。可做角度检出凸轮控制动作。

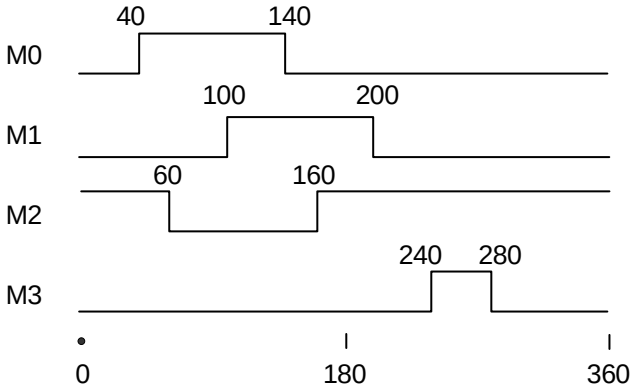
♦左例是用以控制旋转台，在旋转一圈内补助继电器 M0~M3 的 ON/OFF 状态。

♦ 使用 MOVE 命令把下列的数值写入 D300~D307

ON 设定数据	OFF 设定数据	输出点
D300= 40	D301= 140	M0
D302= 100	D303= 200	M1
D304= 160	D305= 60	M2
D306= 240	D307= 280	M3

把 Turn ON 点的数值放在编号为偶数的 D 要素中，并且把 Turn OFF 点的数值放在编号为奇数的 D 要素中。

♦ X0 为 ON 时，M0~M3 的变化为如下所描述。Turn ON 点及 Turn OFF 点的数值能够重新更改写入到 D300~D307。



♦输出点的号码由[D.]的设定值决定。

♦当 X0 变为 OFF 时，输出保持不变。

♦ ABSD 命令在一程序中仅能使用一次。

♦ 若在[S.]中指定高速计数器时，亦可使用(D)ABSD 命令。
但此时对于计数器之现在值，其输出型态将因其扫描周期而产生延迟现象，建议使用 HSZ 命令中之 Table 高速比较模式。

◎ 鼓轮控制 INCREMENTAL DRUM SEQUENCE

FNC(63)	16 bits: INCD - 9 steps
INCD	

		J1n	J2n--

Operands: [S1.]

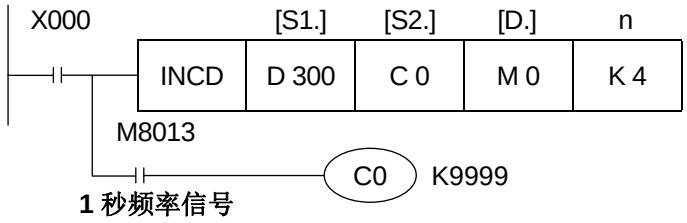
K.H.	KnX	KnY	KnM	KnS	T	C	D	V,Z

n ≤ 64 [S2.]

Operands:

X	Y	M	S

[D.]



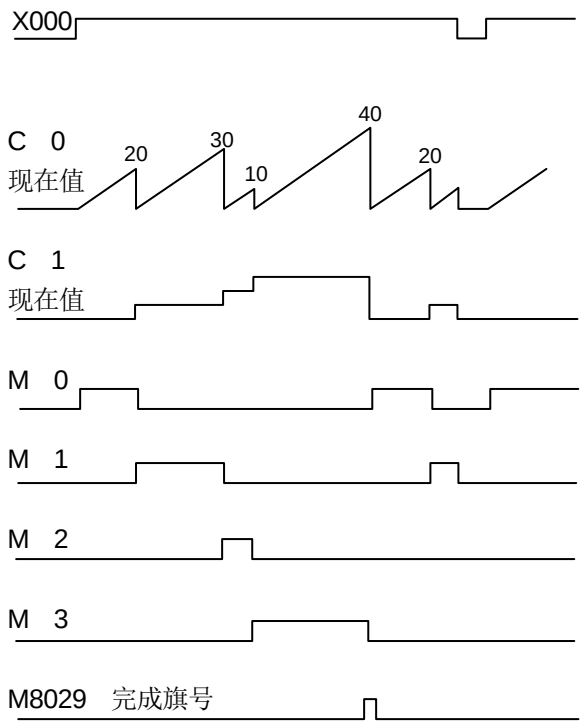
本命令在使用一对计数器情况下用以产生一多变的输出。

如下为四点 (M0~M3) 的控制范围。

♦使用 MOVE 命令把下列的数值预先写入[S1.]中。

D300 = 20　　D302 = 10

D301 = 30　　D303 = 40



♦当 C0 的计数值达到 D300~D303 的设定值时，C0 自动地依序被复置。

♦C1 计算 C0 复置发生的次数。

♦M0~M3 依据 C1 的计数值依序动作。

♦在完成由"n"所设定次数的最后一次处理后，旗号 M8029 变为 ON。如此相定的循环将一再重复。

♦在 X0 为 OFF 时 C0 及 C1 被清除，M0~M3 变为 OFF，然后在 X0 又变为 ON 时重新运作。

♦INCD 命令在一程序中仅能使用一次。

◎ 教导式定时器 TECHING TIMER

FNC(64)			16 bits: TTMR - 5 steps			
	TTMR					

Reserved

◎ 特殊定时器 SPECIAL TIMER

FNC(65)			16 bits: STMR - 7 steps			
	STMR					

Reserved

◎ 交替式输出 ALTERNATE OUTPUT

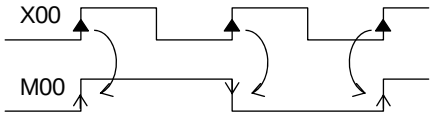
FNC(66)			16 bits: ALT(P) - 3 steps
	ALT	P	

		J1n	J2n--

Operands: [D.]

X	Y	M	S

影响旗号:



♦连续使用此指令，可得到多段除频

◎ 斜率信号 RAMP

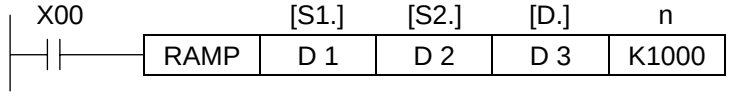
FNC(67)			16 bits: RAMP - 9 steps
	RAMP		

		J1n	J2n--

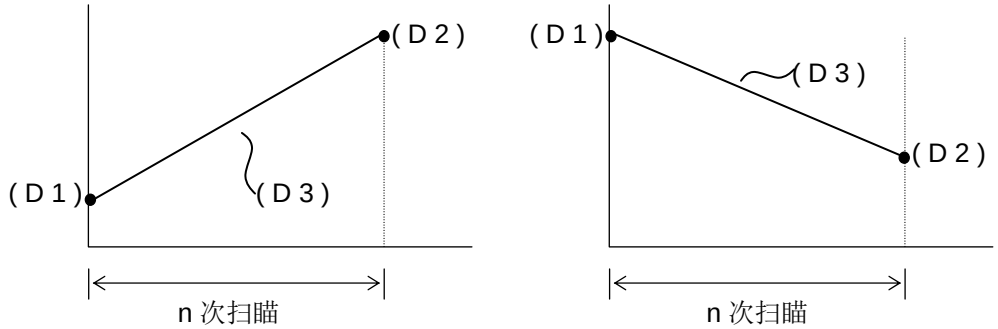
Operands: [S1.][S2.][D.] : D

n : K,H n = 1 to 32,767

影响旗号: M8029



♦ 在 X0 为 ON 时分别把 D1 及 D2 写入初值及终值。D3 内的数值则由 D1 的设定值逐渐向 D2 的设定值递增，而所需的时间(扫瞄次数)，则由"n"指定。



扫瞄次数放在 D4 内。

♦ 在 M8029 被驱动后把 1 次扫瞄时间值(比实际扫瞄时间稍长)写入 D8039 中，则 PLC 将进入定扫瞄模式。
例如，上例中 n = K1000，若扫瞄周期设定为 20msec，则 D3 内的数值将在 20 秒内由 D1 的设定值变为 D2 的设定值。

♦ 假如 X0 在动作途中变为 OFF，则倾斜信号的动作也半途停止，若 X0 又再次为 ON 时，D4 被清除，D3 由 D1 设定值重新开始。

♦ 在执行结束后，旗号 M8029 动作，然后 D3 的值也复归为 D1 的值。

♦ 在起始 / 结束点的控制可以由结合 RAMP 命令及模拟输出加以执行。

♦ 在 X0 为 ON 时进入 RUN 状态，若 D4 具停电保持作用，则应先将 D4 清除。

◎ 旋转控制 ROTARY CONTROL

FNC(68)			16 bits: ROTC - 9 steps				
	ROTC						

Reserved

◎ 资料排列 SORT

FNC(69)			16 bits: SORT - 11 steps				
	SORT						

Reserved

◎ 十键输入 TENKEY INPUT

FNC(70)			16 bits: TKY - 7 Steps
D	TKY		32 bits: (D)TKY - 13 Steps

		J1n	J2n--

Operands: [D1.]

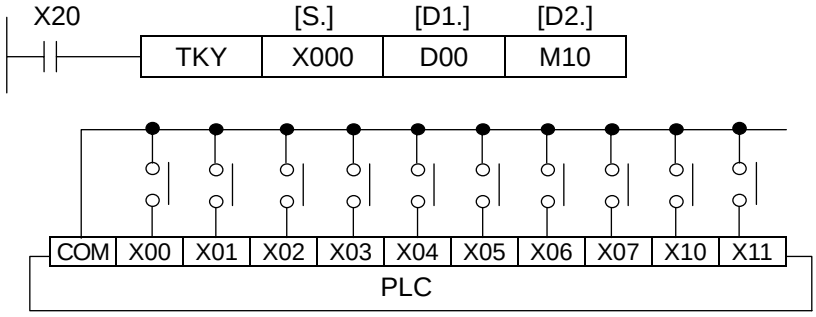
K.H.	KnX	KnY	KnM	KnS	T	C	D	V,Z

Operands: [S.]

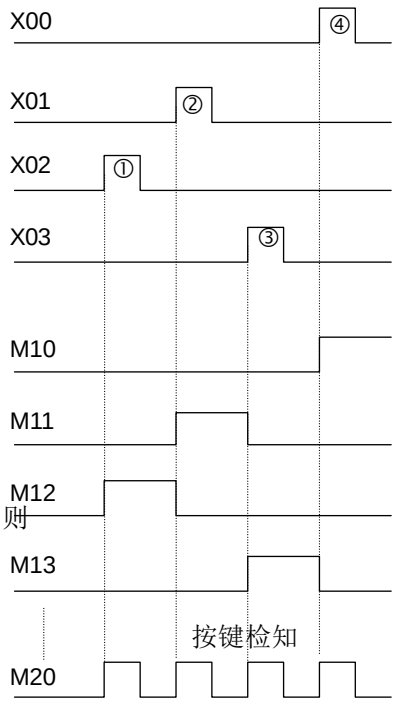
X	Y	M	S

[D2.]

影响旗号:



- 此命令的用途为利用按钮开关将输入数值写入数据缓存器。
- 号码键 (0~9) 须连续连接到以 X00 为开头的输入端。
- 16 位元模式下，[D1]可储存 0000 ~ 9999 (最大 4 位数)。
  32 位元模式下，[D1]可储存 00000000 ~ 99999999 (最大 8 位数)。两模式下，假如输入的数值超过允许范围。则最高位数溢位并且被忽略。
- 当 X20 OFF 时，[D2.]中的位要素被清除，但[D1.]中的数值保持不变。
- 本指令在一程序中只能使用一次。



- 此范例为依序按号码键 2,1,3, 0，亦即将数值 2130 存入 D00 中。
- 如果一次按了 2 个以上的按键，则只有第一个被按下的按键有效。
- 此范例 M10~M19 根据 X00~X11 的输入而变化。
- 在任何按键被按下且未放开前，按键辨认旗号 M20 保持为 ON，放开按键则 M20 立即变为 OFF。

◎ 十六键输入 HEXADECIMAL KEY

FNC(71)			16 bits: HKY - 9 Steps
D	HKY		32 bits: (D)HKY - 17 Steps

		J1n	J2n--

Operands: [D2.]

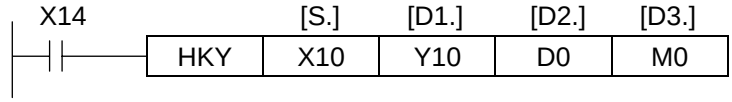
K.H.	KnX	KnY	KnM	KnS	T	C	D	V,Z

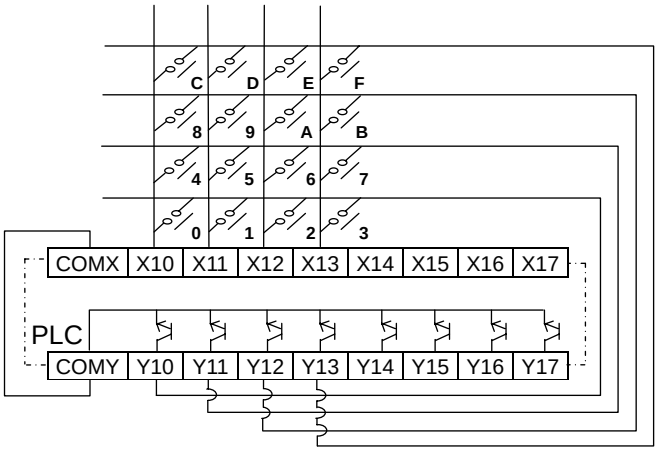
Operands: [D3.]

X	Y	M	S

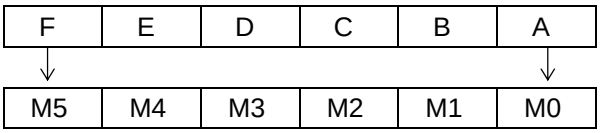
[S.] [D1.]

影响旗号:





- 此命令为利用十六进制键盘读取数值或功能键。
- 当数位键(0 ~ 9)被按下且未放开时，则位要素 [D3.]+7 维持在 ON 状态。
- 当功能键(A ~ F)被按下且未放开时，则位要素 [D3.]+6 维持在 ON 状态。
- 当功能键(A ~ F)被按下，则相对应的位要素 [D3.]+0 ~ [D3.]+5 亦被设为 ON，直到下个功能键被按下才被清除。



- 16 位元模式下，[D2.] 可储存 0000 ~ 9999 (最大 4 位数)。32 位元模式下，[D2.]可储存 0000000 ~ 99999999(最大 8 位数)。两模式下，假如输入的数值超过允许范围，则最高位数溢位并且被忽略。
- 如果同时按 2 个以上的按键，则只有第一个被按下的有效。
- 当 X14 OFF 时，[D3.]中的位要素被清除，但[D2.]中的数值保持不变。
- 此命令须 8 个扫瞄周期来读取按键，当察觉按键被按下后，旗号 M8029 被设定为 ON。
- 此命令只能使用一次，且应使用晶体输出型式。

◎ 指拨开关 DIGITAL SWITCH

FNC(72)			16 bits: DSW - 9 Steps
	DSW		

		J1n	J2n--

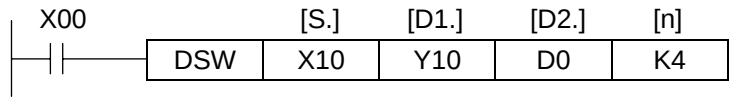
Operands: ←[n]→ =1~8 ←——— [D2.] ———→

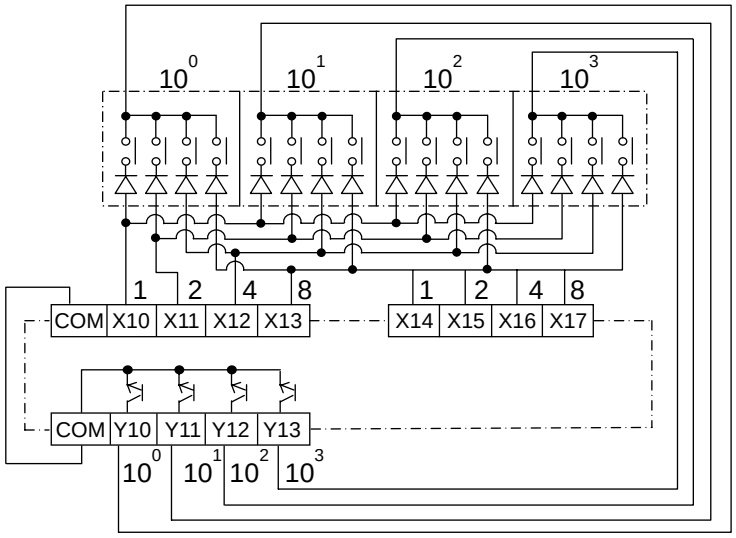
K.H.	KnX	KnY	KnM	KnS	T	C	D	V,Z

Operands: <[S.]> <[D1.]>

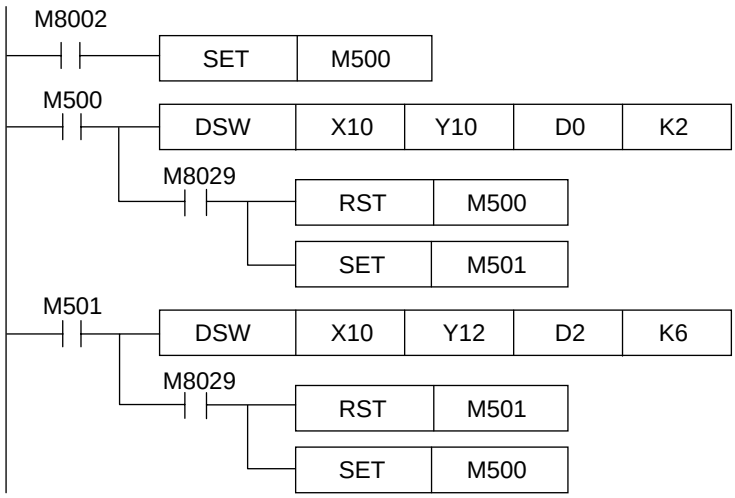
X	Y	M	S

影响旗号:M8029





- 本命令可用 n=1~8 点输出点，以及 4 点输入点读取 n 个(n=1~8)指拨开关的数值，若读入数大于 32 位(n≥5)则[D2.]自动占用下一组缓存器。
- BCD 4 位数指拨开关 1,2,4,8 接脚连接到 X10~X13 或 X14~X17，[S.]须以 X10,X14,X20,X24…为起始点，[D1.]则可任意指定，但不可超出 1 个 CHANNEL 值，如 n=2 [D1.]不可指定为 Y17，Y10~Y16 都可被指定。
- 一旦命令被执行则 M8029 清除为"0"，Y10~Y13 依序动作，待执行完毕(数个演算周期后)，M8029 被设定为"1"。
- 指拨开关上之每 1,2,4,8 接脚须再外接二极管(0.1A/50V)
- 允许多个 DSW 命令，但一次仅允许一个 DSW 命令及动作，如下图利用 M8029 来控制两组 DSW 的命令。



◎ 七段显示器译码器 SEVEN SEGMENT DECODER

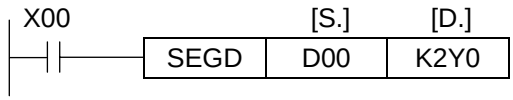
FNC(73)			16 bits: SEGD(P) - 5 steps
	SEGD	P	

		J1n	J2n--

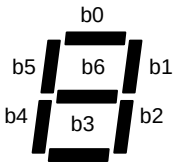
Operands:

[S.]								
K.H.	KnX	KnY	KnM	KnS	T	C	D	V,Z
[D.]								

影响旗号:



♦ [S.] 中下 4 位所指定的数值 0~F (十六进制)，被译码为 7 段显示用数值后放入[D.]中。[D.]的上 8 位不变。

(S.)		7 段显示器之构成	(D.)								显示数据
16 进制	Bit 组合		b7	b6	b5	b4	b3	b2	b1	b0	
0	0000		0	0	1	1	1	1	1	1	0
1	0001		0	0	0	0	0	1	1	0	1
2	0010		0	1	0	1	1	0	1	1	2
3	0011		0	1	0	0	1	1	1	1	3
4	0100		0	1	1	0	0	1	1	0	4
5	0101		0	1	1	0	1	1	0	1	5
6	0110		0	1	1	1	1	1	0	1	6
7	0111		0	0	1	0	0	1	1	1	7
8	1000		0	1	1	1	1	1	1	1	8
9	1001		0	1	1	0	1	1	1	1	9
A	1010		0	1	1	1	0	1	1	1	A
B	1011		0	1	1	1	1	1	0	0	b
C	1100		0	0	1	1	1	0	0	1	C
D	1101		0	1	0	1	1	1	1	0	d
E	1110		0	1	1	1	1	0	0	1	E
F	1111		0		1	1	0	0	0	1	F

♦ 位要素的开头(本例为 Y0)或字符要素的最低位(LSB)均对称到 b0，且依此类推。

◎ 栓锁式七段显示器 SEVEN SEGMENT WITH LATCH

FNC(74)			16 bits: SEGL(P) - 5 steps
	SEGL	P	

Reserved

◎ 箭号开关 ARROW SWITCH

FNC(75)			16 bits: ARWS(P) - 9 steps
	ARWS		

Reserved

◎ ASCII 转换 ASCII CODE CONVERSION

FNC(76)			11 steps
	ASC		

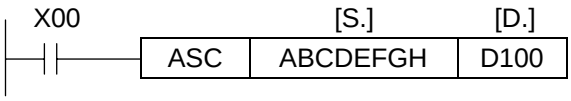
		J1n	J2n--

Operands: [S.]: 8 个英文字或阿拉伯数字。

Operands: ←— [D.] —→

	T	C	D

影响旗号:



♦ 字符"A" ~ "H"被转换成 ASCII 码并放在 D100~D103 中。

M8161 OFF 时

M8161=OFF	上 8 位	下 8 位
D100	"B"	"A"
D101	"D"	"C"
D102	"F"	"E"
D103	"H"	"G"

M8161 ON 时

	上 8 位	下 8 位		上 8 位	下 8 位
D100	0	"A"	D104	0	"E"
D101	0	"B"	D105	0	"F"
D102	0	"C"	D106	0	"G"
D103	0	"D"	D107	0	"H"

◎ 打印 PRINT

FNC(77)			16 bits: PR 5 steps
	PR		

Operands: ←— [S.] —→

	T	C	D

Operands: [D.]: Y

Reserved

◎ FROM 命令

FNC(78)			16 bits: FROM(P) - 9 steps
D	FROM	P	32 bits: (D)FROM(P) - 17 steps

		J1n	J2n--

Operands: ←——— [D.] ———→

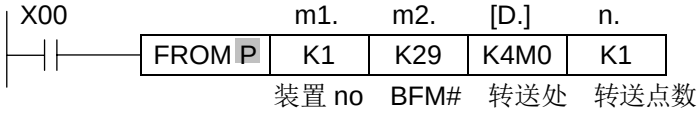
K.H.	KnX	KnY	KnM	KnS	T	C	D	V,Z

Operands: ← → m1 = 0 ~ 7 特殊模块号码

m2.= 0 ~ 31 缓冲存储器 (BFM) 号码

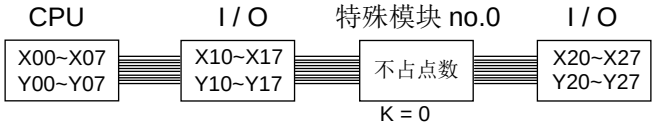
n.= 1 ~ 31 转送点数 (D 命令时=1 ~ 15)

影响旗号:



♦ 当 X00 ON 时，将特殊模块 NO.1 之缓冲存储器 BFM#29 读出，转送到可程控器之 M00~M15。

<<特殊装置 模块号码 m1>>



♦ 特殊模块号码的排列依靠近主机的顺序分别为 NO.0~NO.7

♦ 特殊模块不占 I/O 点数且最多可扩充 8 台。

♦ 所谓缓冲存储器 BFM 即为特殊模块与可程控器沟通之数据缓存器。

◎ TO 命令

FNC(79)			16 bits: TO(P) - 9 steps
D	TO	P	32 bits: (D)TO(P) - 17 steps

		J1n	J2n--

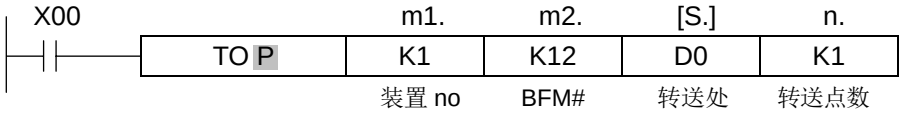
Operands: ←——— [S.] ———→

K.H.	KnX	KnY	KnM	KnS	T	C	D	V,Z

Operands: ← → m1 = 0 ~ 7 特殊模块号码

m2.= 0 ~ 31 缓冲存储器(BFM) 号码 n.= 1 ~ 31 转送点数(D 命令时=1 ~ 15)

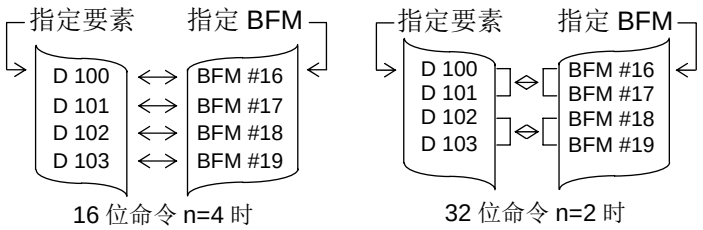
影响旗号:



♦ 当 X00 ON 时，将 D0 的 16 位数据写入特殊模块 NO.1 之缓冲存储器 BFM#12。

♦ 此命令尽量使用脉波命令，低扫瞄周期时间。

<< 转送点数 n >>



◎ 数据通讯 COMMUNICATION

FNC(80)			16 bits: RS - 9 steps
	RS		

		J1n	J2n--

Operands:

							←S, →	
K.H.	KnX	KnY	KnM	KnS	T	C	D	V,Z
←——→	m,n=1~128						← D.→	m,n

影响旗号:

<< 通讯格式 >> D8120

	内容	0	1
Bit0	数据长度	7 bit	8 bit
Bit1	同位	(00):none, (01):odd, (11):even	
Bit2			
Bit3	停止位	1 bit	2 bit
Bit4	传送速率 (bps)	(0011):300, (0100):600	
Bit5		(0101):1200, (0110):2400	
Bit6		(0111):4800, (1000):9600	
Bit7		(1001):19200	
Bit8	前端 1	无	D8124
Bit9	终端 1	无	D8125
Bit10	Reserved	-	-
Bit11	Reserved	-	-
Bit12	终端 2	无	D8126
Bit13	通讯模式	自定模式	ModBus
Bit14	ModBus Mode	Ascii Mode	RTU Mode 或 Computer Link
Bit15	Protocol	Format 1	Format 4

♦此命令乃利用主机第二通讯端口连接 EXADP232/422/485 通讯板来执行数据的传送与接收，通讯协议由 D8120 来设定。

♦由于通讯格式与数据框架(FRAME)均由使用者自行规划，且可选用不同的通讯接口板，所以此通讯 Port 可连接多种不同通讯格式的机器。

♦主机运转开始，先自行检查是否有书写 RS 指令。若有，则 Computer Link 模式即无效。通讯协议模式变更为 user define mode 或 Modbus mode。

♦Computer link mode:此模式程序不可书写 RS 指令，亦即均为被动站，仅设定 D8120 及 D8121 的内容值(D8120 的 bit14 须设为 1)，即可架构多站连结的系统。

例:

```
M8002
─┤├─[ MOV    HE086    D8120 ] ; 通讯协议格式 4; 9600; 7E1
  └─[ MOV    K1       D8121 ] ; station no. (1~32)
```

♦Modbus mode:此模式程序须利用 RS 指令来变更通讯协议(D8120 的 bit13 须设为 1)。由于有 RS 指令所以可当为主动站，亦可为被动站。利用 M8122 及 M8123 控制数据传送与接收。

例:

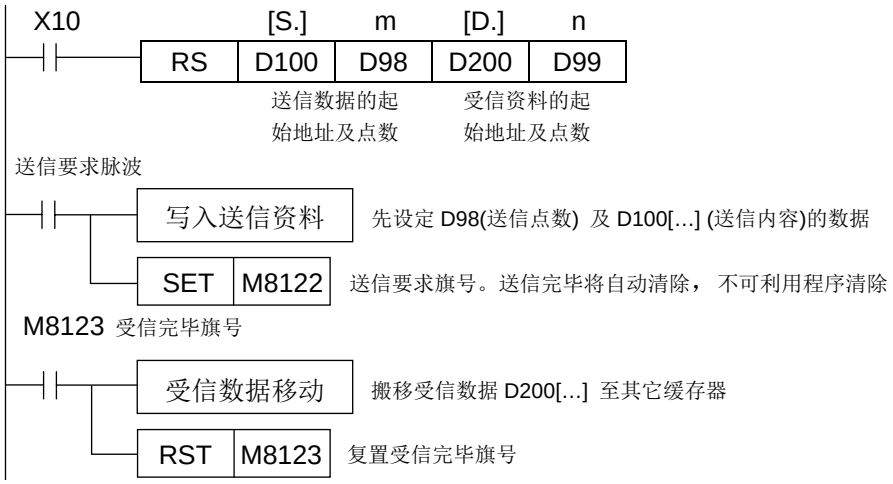
```
M8002
─┤├─[ MOV    H3386    D8120 ] ; Modbus Ascii mode; 9600; 7E1
  └─[ MOV    H00      D8121 ] ; Master station
─┤├─[ RS     D100     D98     D200     D99 ]
```

♦RS 命令被驱动后，若修改 D8120 之设定值，将不被接受。

♦此命令在程序中不限使用次数，但一扫瞄周期仅可用一驱动命令，且于切换时须设计 1 个扫瞄周期以上之 OFF 时间。

♦此通讯 Port 可为主动模式亦可为被动模式。所以 RS 指令一旦被驱动，则 PLC 即处于等待送信与受信的状态。

♦若 RS 指令已使用，则不可再使用 PRUN 指令。



<< 送信要求 >> M8122

♦ 无论在等待受信或受信完毕的状态，若以脉波指令驱动 M8122 送信要求旗号，则 PLC 将从 D100 起始之 D98 点的

数据传送出，且于送信完毕后 M8122 将自动 Reset。

♦ PLC 于数据受信结束后才会执行数据送信，在此期间，若要求送信则等待送信旗号 M8121 将被设定。

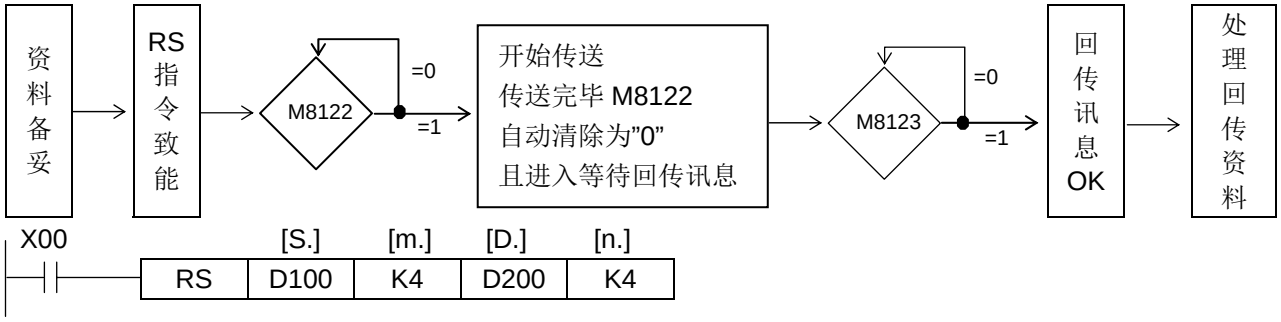
<< 受信完毕 >> M8123

♦ PLC 接收数据完毕，则受信完毕旗号 M8123 被设定。请利用程序将 M8123 复置，此时 PLC 将在处于等待受信状态。

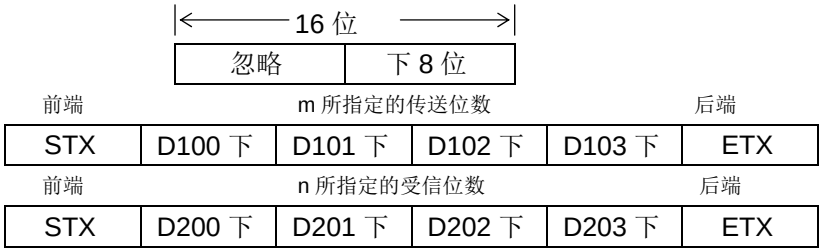
♦ M8123 动作中，若接获送信要求，M8123 不会被清除，但仍会执行数据送信。

<< 载波检出 >> M8124

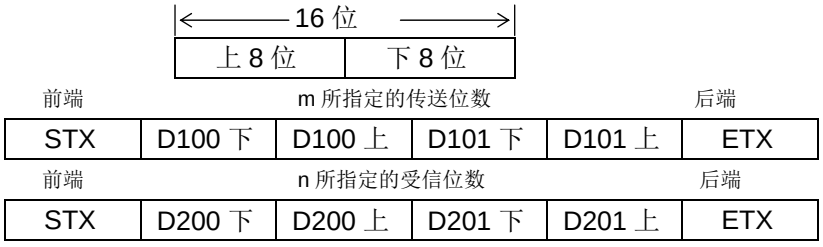
<< RS 指令通则 >>



< 8 位元数据处理模式 > M8161=ON 为 8 位元模式



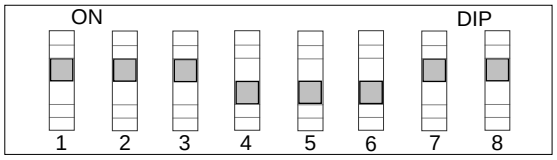
< 16 位元数据处理模式 > M8161=OFF 为 16 位元模式



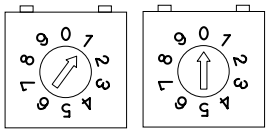
♦ 送信/受信中若有错误发生，M8063 将被设定，且错误内容将会写入 D8063 中。

<< MODBUS RTU 模式的应用 >> CRC 侦误方式

♦ EXRM0808R/T 开关



SW1



RSW1 RSW2

例: 主站与 Remote I/O 模块联机范例程式

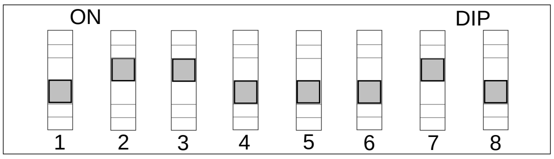
```
M8002
 |--[ZRST  D0    D255]
 |--[SET   M8161]    8 位元模式
 |--[MOV   H6087 D8120]  ModbusRTU, 9600, 8, E, 1
 |--[MOV   K0    D8121]  PLC 站号 0(主站)
 |--[MOV   K200  D8129]  逾时监控
X007
 |/-[MOV   K1    D10 ]   站号 1(副站)
 |--[MOV   H0006 D11 ]   写入资料
 |--[MOV   H0000 D12 ]   地址 Upper byte
 |--[MOV   H0004 D13 ]   地址 Lower byte
 |--[MOV   H0000 D14 ]   资料 Upper byte
 |--[MOV   H0084 D15 ]   资料 Lower byte
 |--[MOV   K6    D98 ]   送信笔数
 |--[MOV   K64   D99 ]   受信笔数
X007
 |--[MOV   K1    D10 ]   站号 1(副站)
 |--[MOV   H0003 D11 ]   读取数据
 |--[MOV   H0000 D12 ]   地址 Upper byte
 |--[MOV   H0000 D13 ]   地址 Lower byte
 |--[MOV   H0000 D14 ]   数据数 Upper (以 word 计算)
 |--[MOV   H0002 D15 ]   数据数 Lower (以 word 计算)
 |--[MOV   K6    D98 ]   送信笔数
 |--[MOV   K64   D99 ]   受信笔数
M8000
 |--[BMOV  D10   D100  D98 ]   移动数据至缓冲存储器
 |--[RS    D100  D98   D200  D99 ]   指定送信/受信起始地址
M500
 |--[SET   M8122]    开始送信旗标
M8123
 |--[RST   M8123]    受信完毕旗标
 |--[INC   D998 ]
 |--[PLS   M500 ]
M8128
 |--[RST   M8128]CRC 错误旗标
 |--[INC   D999 ]
 |--[END        ]
```

♦ 在 ModBus RTU 模式下，请将 D8120 的通讯格式设为无前/终端(无 STX/ETX)，且送信数据点数必须正确。

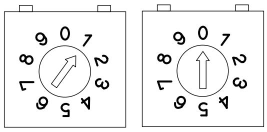
♦ 侦误值不列入送信笔数，且由 PLC 自动计算，并将结果存入下 2 个缓存器。

<< MODBUS ASCII 模式的应用 >> LRC 侦误方式

♦ EXRM0808R/T 开关



SW1



RSW1 RSW2

例: 主站与 Remote I/O 模块联机范例程式

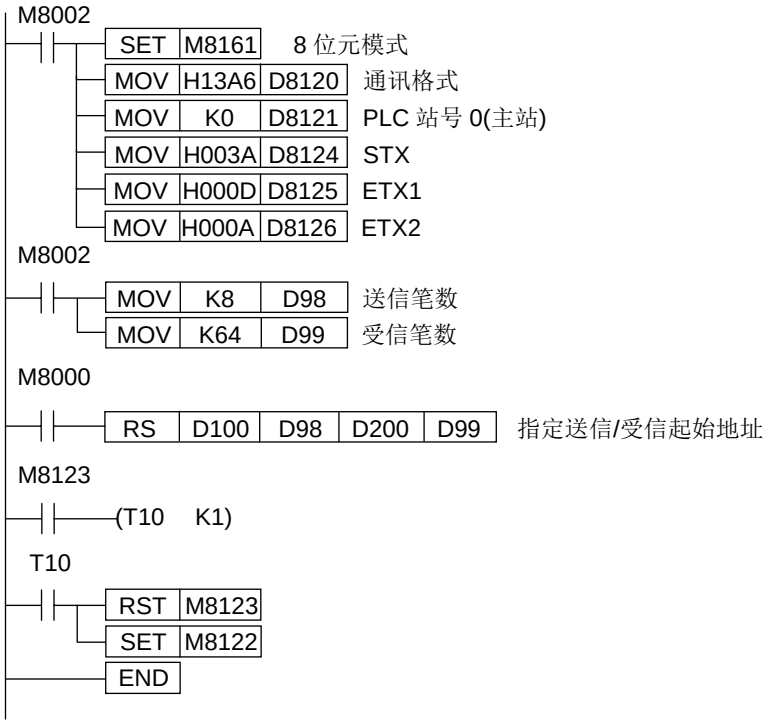
```
M8002
─┤├─┬─[ZRST  D0     D255 ]
    ├─[SET   M8161       ]   8 位元模式
    ├─[MOV   H3386  D8120]   ModbusAscii, 9600, 7, E, 1
    ├─[MOV   K0     D8121]   PLC 站号 0(主站)
    ├─[MOV   H003A  D8124]   STX
    ├─[MOV   H000D  D8125]   ETX1
    ├─[MOV   H000A  D8126]   ETX2
    └─[MOV   K200   D8129]   逾时监控
X007
─┤/├─┬─[MOV  K1     D10  ]   站号 1(副站)
     ├─[MOV  H0006  D11  ]   写入资料
     ├─[MOV  H0000  D12  ]   地址 Upper byte
     ├─[MOV  H0004  D13  ]   地址 Lower byte
     ├─[MOV  H0000  D14  ]   资料 Upper byte
     ├─[MOV  H0084  D15  ]   资料 Lower byte
     ├─[MOV  K6     D98  ]   送信笔数
     └─[MOV  K64    D99  ]   受信笔数
X007
─┤├─┬─[MOV   K1     D10  ]   站号 1(副站)
    ├─[MOV   H0003  D11  ]   读取数据
    ├─[MOV   H0000  D12  ]   地址 Upper byte
    ├─[MOV   H0000  D13  ]   地址 Lower byte
    ├─[MOV   H0000  D14  ]   数据数 Upper (以 word  计算)
    ├─[MOV   H0002  D15  ]   数据数 Lower (以 word 计算)
    ├─[MOV   K6     D98  ]   送信笔数
    └─[MOV   K64    D99  ]   受信笔数
M8000
─┤├─┬─[BMOV  D10    D100  D98      ]   移动数据至缓冲存储器
    └─[RS    D100   D98   D200  D99]   指定送信/受信起始地址
M500
─┤↑├──[SET   M8122]   开始送信旗标
M8123
─┤├─┬─[RST   M8123]   受信完毕旗标
    ├─[INC   D998 ]
    └─[SET   M8122]
M8129
─┤├─┬─[RST   M8129]   LRC 错误旗标
    └─[INC   D999 ]
──────[END]
```

♦ 在 ModBus Ascii 模式下，请将 D8120 的通讯格式设为有前/终端(STX/ETX1,2) ，且送信数据点数必须正确。

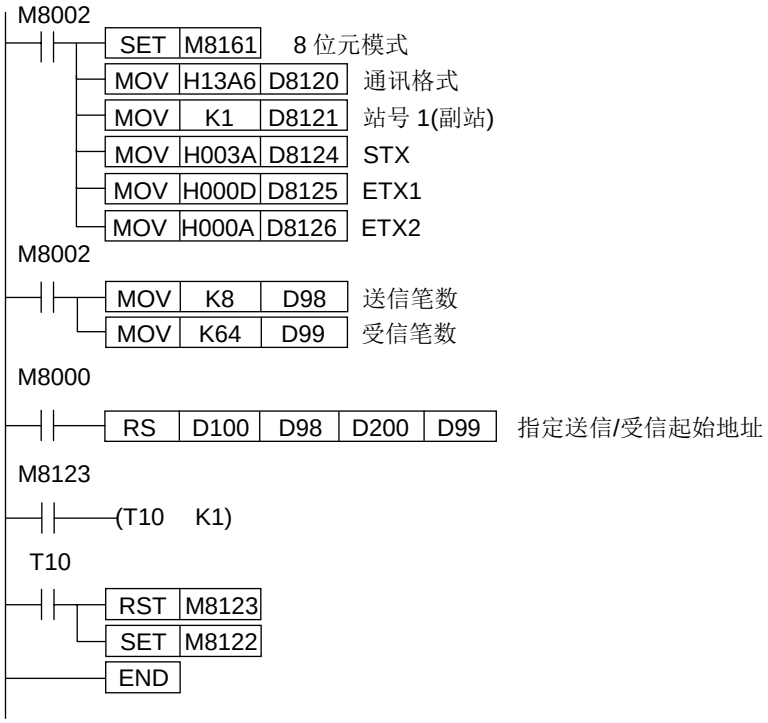
♦ 侦误值不列入送信笔数，且由 PLC 自动计算，并将结果存入下 2 个缓存器。

<< 使用者自定模式的应用 >> 自定侦误方式

例 1: Ascii 模式主站范例程式

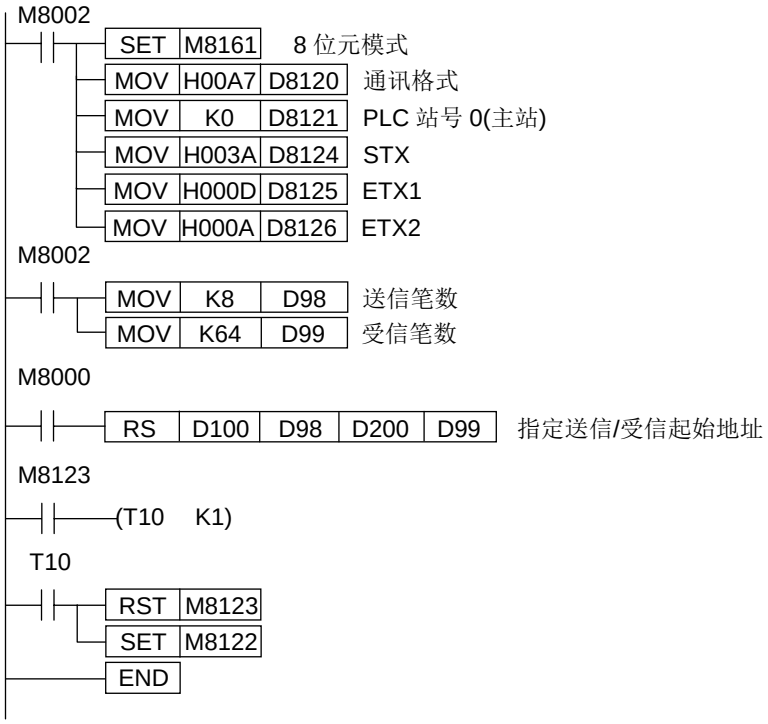


副站范例程式

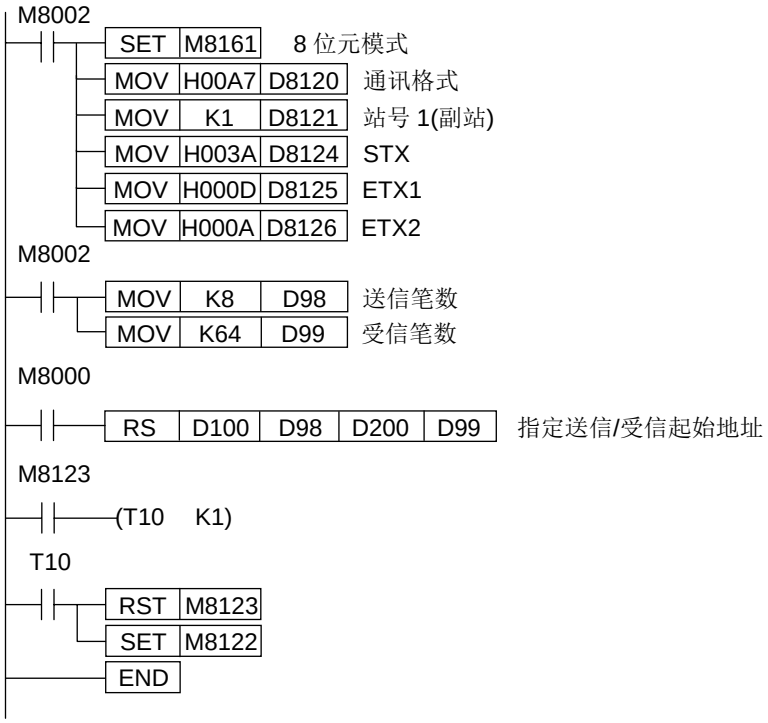


- 侦误值须由程序设计人员计算，系统不自动演算。
- 传送数据须转换为 Ascii 存入传送区。

例 2: HEX 模式主站范例程式



副站范例程式



♦ 侦误值须由设计人员计算，系统不自动演算。

◎ 双机并联运转 PARALLEL RUNNING

FNC(81)			16 bits: PRUN(P) - 5 Steps
D	PRUN	P	32 bits: (D)PRUN(P) -9 Steps

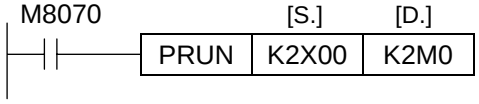
		J1n	J2n--

Operands: [S.]: KnX, KnM 最低位数字须为”0”

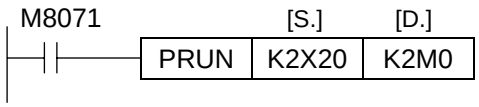
[D.]: KnM, KnY 最低位数字须为”0”

影响旗号: M8073, M8129

亲局程序 M8070=1，[S.] [D.]为虚操作数



子局程序 M8071=1，[S.] [D.]为虚操作数



- ♦ 亲局会将 D490~D497 传入子局的 D490~D497(M8070=1)。
- ♦ 子局会将 D500~D507 传入亲局的 D500~D507(M8070=0)。
- ♦ 此命令只须设定 M8070 及 M8071，数据缓存器(D)不须指定，两者即会自动通讯。
- ♦ 由于辅助缓存器(M)不对传，必要时利用 MOV 命令转换。

♦相关参数

M8122: 启动通讯传输旗号
M8123: 接收完毕旗号
M8070: 亲局旗号
M8071: 子局旗号
M8129: 和检查错误旗号
M8073: 逾时旗号
D8070: 逾时缓存器(ms)
D8072: 通讯所费时间(ms)

♦范例程式请参阅力扬应用范例 F081。

♦若 PRUN 指令已使用，则不可再使用 RS 指令。

◎ HEX TO ASCII 转换

FNC(82)			16 bits: ASCI(P) - 7 Steps
	ASCI	P	

		J1n	J2n--

Operands: [S.]

K.H.	KnX	KnY	KnM	KnS	T	C	D	V,Z

[D.]

n = (1 ~ 64)

影响旗号:

	[S.]	[D.]	n
ASCI	D100	D200	K4

♦ [S.]内的 16 进制数据被转换为 ASCII 码传送到 [D.]的上/下 8 位中。转换的文字数由 n 指定。

♦ 当 M8161=OFF 时，为 16 位元模式。

例: (D100)=0ABCH，(D101)=1234H

	K1	K2	K3	K4	K5	K6	K7	K8
D200 下	“C”	“B”	“A”	“0”	“4”	“3”	“2”	“1”
D200 上		“C”	“B”	“A”	“0”	“4”	“3”	“2”
D201 下			“C”	“B”	“A”	“0”	“4”	“3”
D201 上				“C”	“B”	“A”	“0”	“4”
D202 下					“C”	“B”	“A”	“0”
D202 上						“C”	“B”	“A”
D203 下							“C”	“B”
D203 上								“C”

X10

(M8161) 8 位元模式

	[S.]	[D.]	n
ASCI	D100	D200	K4

M8000

(M8161) M8161 = ON 时，为 8 位元模式

16 位	
0	下 8 位

目的数据

♦ [S.]内的 16 进制数据被转换为 ASCII 码传送到[D.]的下 8 位中。转换的文字数由 n 指定。

♦ 当 M8161=ON 时，为 8 位元模式。

例: (D100)=0ABCH, (D101)=1234H

	K1	K2	K3	K4	K5	K6	K7	K8
D200 下	“C”	“B”	“A”	“0”	“4”	“3”	“2”	“1”
D201 下		“C”	“B”	“A”	“0”	“4”	“3”	“2”
D202 下			“C”	“B”	“A”	“0”	“4”	“3”
D203 下				“C”	“B”	“A”	“0”	“4”
D204 下					“C”	“B”	“A”	“0”
D205 下						“C”	“B”	“A”
D206 下							“C”	“B”
D207 下								“C”

◎ ASCII TO HEX 转换

FNC(83)			16 bits: HEX(P) - 7 Steps
	HEX	P	

		J1n	J2n--

Operands: [S.]

K.H.	KnX	KnY	KnM	KnS	T	C	D	V,Z

[D.]

n = (1 ~ 64)

影响旗号:

	[S.]	[D.]	n
HEX	D200	D100	K4

♦ [S.]内上/下 8 位的 ASCII 码被转换为 16 进制数据，以每 4 笔为单位传送到[D.]中。转换的文字数由 n 指定。

♦ 当 M8161=OFF 时，为 16 位元模式。

例: D200 下=”0”，D200 上=”A”，D201 下=”B”，D201 上=”C”

D202 下=”1”，D202 上=”2”，D203 下=”3”，D203 上=”4”

	D102	D101	D100
K1			0H
K2			0AH
K3			0ABH
K4			0ABCH
K5		0H	ABC1H
K6		0AH	BC12H
K7		0ABH	C123H
K8		0ABCH	1234H

X10

(M8161) 8 位元模式

	[S.]	[D.]	n
HEX	D200	D100	K4

M8000

(M8161) M8161 = ON 时，为 8 位元模式

16 位

忽略	下 8 位

来源资料

♦ [S.]内下 8 位的 ASCII 码被转换为 16 进制数据，以每 4 笔为单位传送到[D.]中。转换的文字数由 n 指定。

♦ 当 M8161=ON 时，为 8 位元模式。

例: D200=”0”，D201=”A”，D202=”B”，D203=”C”

D204=”1”，D205=”2”，D206=”3”，D207=”4”

	D102	D101	D100
K1			0H
K2			0AH
K3			0ABH
K4			0ABCH
K5		0H	ABC1H
K6		0AH	BC12H
K7		0ABH	C123H
K8		0ABCH	1234H

◎ 检查码 CHECK CODE

FNC(84)			16 bits: CCD(P) - 7 Steps
	CCD	P	

		J1n	J2n--

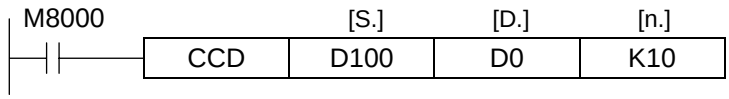
Operands: [S.]

K.H.	KnX	KnY	KnM	KnS	T	C	D	V,Z

n = 1~64 (n 适用于 [S.] 与 D)

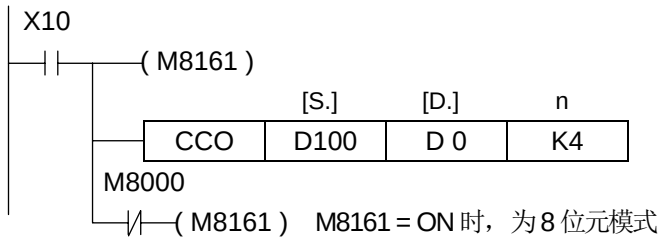
[D.]

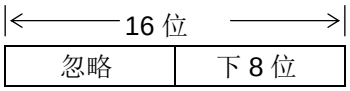
影响旗号:



♦ 将所指定之起始地址之 n 笔数据(8 位)，其总和→D00，水平极性 Vertical Parity→D01([D.]+1)。

M8161=OFF 16 位元模式									
(S.)		Bit Pattern							
D100 L	K100	0	1	1	0	0	1	0	0
D100 H	K111	0	1	1	0	1	1	1	1
D101 L	K100	0	1	1	0	0	1	0	0
D101 H	K98	0	1	1	0	0	0	1	0
D102 L	K123	0	1	1	1	1	0	1	1
D102 H	K66	0	1	0	0	0	0	1	0
D103 L	K100	0	1	1	0	0	1	0	0
D103 H	K95	0	1	0	1	1	1	1	1
D104 L	K210	1	1	0	1	0	0	1	0
D104 H	K88	0	1	0	1	1	0	0	0
Vertical parity		1	0	0	0	0	1	0	1
Sum	K1091								





来源资料

♦ 将所指定之起始地址之 n 笔数据(仅 8 位)，其总和→D00，水平极性 Vertical Parity→D01([D.]+1).

M8161=ON 8 位元模式									
(S.)		Bit Pattern							
D100	K100	0	1	1	0	0	1	0	0
D101	K111	0	1	1	0	1	1	1	1
D102	K100	0	1	1	0	0	1	0	0
D103	K98	0	1	1	0	0	0	1	0
D104	K123	0	1	1	1	1	0	1	1
D105	K66	0	1	0	0	0	0	1	0
D106	K100	0	1	1	0	0	1	0	0
D107	K95	0	1	0	1	1	1	1	1
D108	K210	1	1	0	1	0	0	1	0
D109	K88	0	1	0	1	1	0	0	0
Vertical parity		1	0	0	0	0	1	0	1
SUM	K1091								

◎ 旋钮读取 VOLUME READ

FNC(85)			16 bits: VRRD(P) - 5 Steps
	VRRD	P	

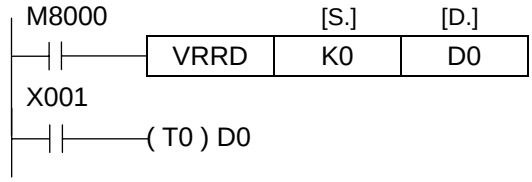
		J1n	J2n--

Operands: [D.]

K.H.	KnX	KnY	KnM	KnS	T	C	D	V,Z

[S.] = (0 – 3)

影响旗号:



- 读取被指定的 VR 模拟输入且转换为 8 位二进制代码(0 ~ 255)再存入目的字符要素中。(K=0 时读取 VR1，K=1 时读取 VR2，K=2 时读取 VR3，K=3 时读取 VR4)
- 目的要素[D.]的内容可作为 Timer 或 Counter 的设定值。

◎ 旋钮刻度 VOLUME SCALE

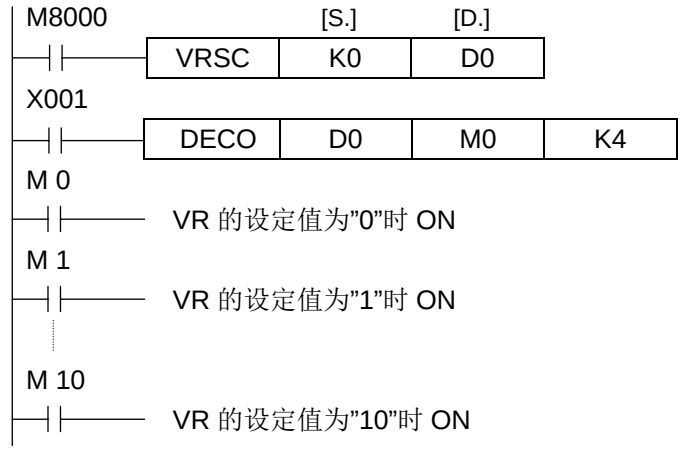
FNC(86)			16 bits: VRSC(P) - 5 Steps
	VRSC	P	

Operands: [D.]

K.H.	KnX	KnY	KnM	KnS	T	C	D	V,Z

[S.] = (0 – 3)

影响旗号:



- 读取被指定的 VR 模拟输入且转换为 8 位二进制代码(0 ~ 255)除以 16 取四舍五入再存入目的字符要素中(0~15)。
- 此指令可将 VR 当作为 16 段的旋转开关。

◎ PID 演算

FNC(88)	16 bits: PID - 9 Steps
PID	

		J1n	J2n--

Operands: [S1.][S2.][S3.]

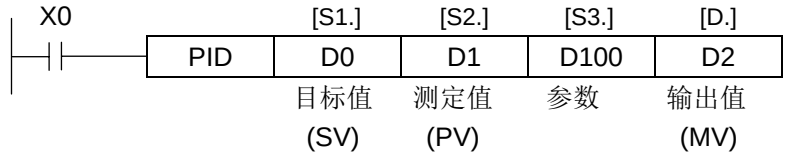
K.H.	KnX	KnY	KnM	KnS	T	C	D	V,Z

[D.]

Operands:

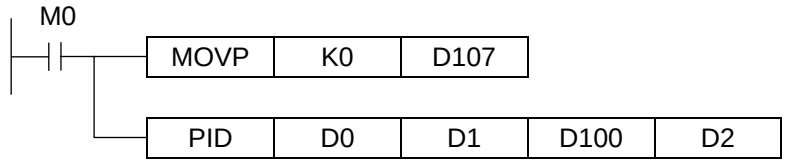
X	Y	M	S

影响旗号:



[S1.]：为目标值
[S2.]：为测定值
[S3.] ~ [S3.]+6：为控制参数
} 以左边的设定执行程序，并将演算结果(MV)存入[D.]中
[D.]：输出值数据缓存器

♦ 从指定的[S3.]开始会占用连续 25 点的组件。在本例中，则占用 D100~D124。
♦ 初次执行时，须先清除[S3.]+7 的内容为 0



♦ 在开始执行 PID 演算前，必须先以 MOV 等命令写入 PID 控制用的参数设定值。

[S3.]	取样时间(Ts)	1~32767 (ms) (不可设定比扫描时间短)
[S3.] + 1	动作方向(ACT)	BIT0 : 0：正动作 ；1：逆动作
		BIT1 : 0：无输入变动量警报 ；1：具输入变动量警报
		BIT2 : 0：无输出变动量警报 ；1：具输出变动量警报
		BIT3：保留
		BIT4：保留
		BIT5 : 0：无输出限制 ；1：具输出限制
		BIT6 ~ BIT15：保留
[S3.] + 2	输入滤波常数(α)	0 ~ 99 (%)
[S3.] + 3	比例增益(Kp)	1 ~ 32767 (%)
[S3.] + 4	积分时间(Ti)	1 ~ 32767 (x 100ms)，0 时为无穷大(无积分动作)
[S3.] + 5	微分增益(Kd)	0 ~ 100 (%)
[S3.] + 6	微分时间(Td)	1 ~ 32767 (x 10ms)，0 为无微分动作
[S3.] + 7 ~ [S3.] + 19		执行 PID 演算时，内部处理用
[S3.] + 20	系统保留	
[S3.] + 21	系统保留	
[S3.] + 22	输出最大值限制，[S3.]+1 的 BIT5=1 时有效	
[S3.] + 23	输出最小值限制，[S3.]+1 的 BIT5=1 时有效	
[S3.] + 24	系统保留	

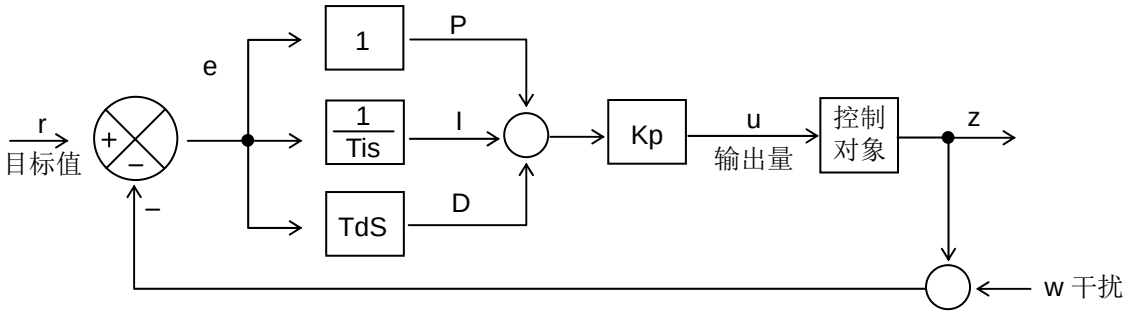
♦ PID 指令的基本演算 ：

本指令依照速度形，测定微分形演算式，执行 PID 演算。

在 PID 的控制，依[S3.]所指定”动作方向”的内容，执行正动作或逆动作的演算式。

PID 基本型式:

输出 $u(t) = Kp \{ e(t) + \frac{1}{Ti} \int_0^t e(t)dt + Td \frac{de(t)}{dt} \}$　　e(t) = 偏差值



◎

FNC(89)							

◎

FNC(90)							

◎

FNC(91)							

◎

FNC(92)							

◎

FNC(93)							

◎

FNC(94)							

◎

FNC(95)							

◎

FNC(96)							

◎

FNC(97)							

◎

FNC(98)							

◎

FNC(99)							

◎ 2 进制浮动小数点比较

FNC(110)			
D	ECMP	P	32 bits:(D)ECMP & (D)ECMP(P) - - - - - - - - - - - 13 steps

			J2n--

Operands: |<[S1.]>|

K,H	KnX	KnY	KnM	KnS	T	C	D	V,Z

|<[S2.]>|

Operands: |<——— [D.] ———>|

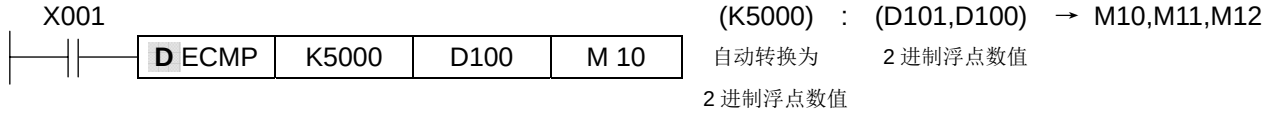
X	Y	M	S

占用[D.]开始连续 3 点

影响旗号: None

即使将 X001 OFF，不执行 ECMP 命令，但是 M0~M2 仍会保持 X001 OFF 前的状态。

- 将 2 个来源组件[S1.]及[S2.]的 2 进制浮点数值做比较，依比较大小的结果，[D.]开始连续 3 点自动 ON/OFF。
- 来源操作数，若以常数 K 或 H 指定时，则自动地转换为 2 进制浮点数值作比较处理。



◎ 2 进制浮动小数点区域比较

FNC(111)			
D	EZCP	P	32 bits:(D)EZCP & (D)EZCP(P) - - - - - - - - - - - - 17 steps

			J2n--

Operands: [S1.] [S2.] [S.] [S1.] [S2.] [S.]

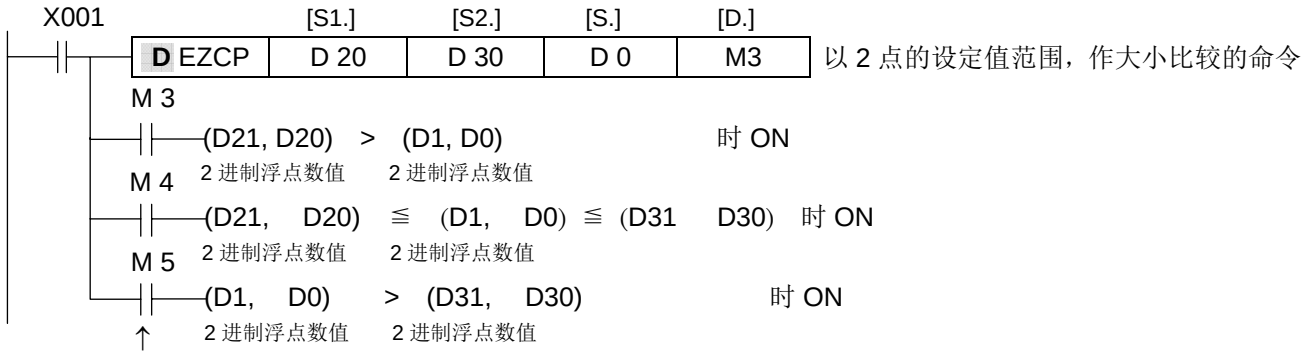
K,H	KnX	KnY	KnM	KnS	T	C	D	V,Z

Operands: [D.]

X	Y	M	S

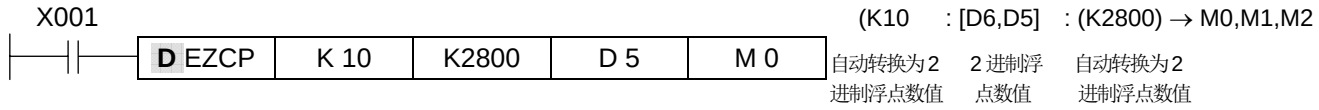
[D.]占用开始连续 3 点，[S1.], [S2.]请设定为[S1.]≦[S2.]

影响旗号: None



即使将 X001 OFF，不执行 ECMP 命令，但是 M3~M5 仍会保持 X001 OFF 前的状态。

- ♦以 2 进制浮点数值所指定的上下 2 点的范围和 [S1.] , [S2.]+1 的内容作比较，依比较大小的结果 [D.] 开始连续 3 点自动 ON/OFF。
- ♦来源的操作数若以常数 K 或 H 指定时，则自动地将其转换成 2 进制浮点数值作比较处理。



- ♦ 请设定 [S1.] ≦ [S2.]。若 [S1.] > [S2.] 时，则 [S2.] 的值视为与 [S1.] 的数值相同。

◎ 2 进制浮动小数点 → 10 进制浮动小数点转换

FNC(118)			
D	EBCD	P	32 bits:(D)EBCD & (D)EBCD(P) - - - - - - - - - - - - - 9 steps

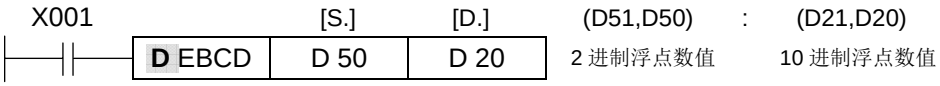
Operands:

K,H	KnX	KnY	KnM	KnS	T	C	D [S.]	V,Z
							D [D.]	

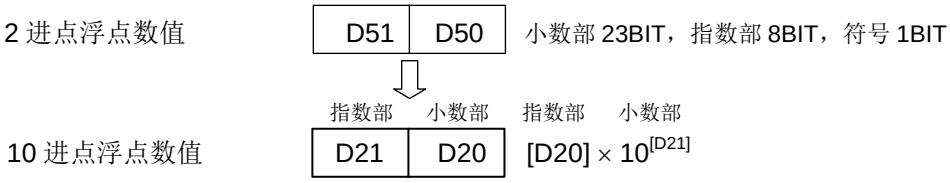
Operands:

X	Y	M	S

影响旗号:



♦ 将来源组件所指定的 2 进制浮点数值，转换成 10 进制浮点数值，存入目的地组件.



♦Reserved

◎ 10 进制浮动小数点 → 2 进制浮动小数点转换

FNC(119)			
D	EBIN	P	32 bits:(D)EBIN & (D)EBIN(P) - - - - - - - - - - - - - 9 steps

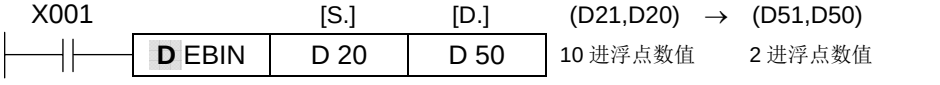
Operands:

K,H	KnX	KnY	KnM	KnS	T	C	D [S.]	V,Z
							D [D.]	

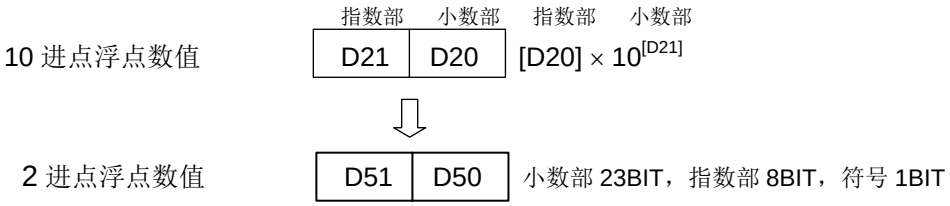
Operands:

X	Y	M	S

影响旗号:



♦ 将来源组件所指定的 10 进制浮点数值，转换成 2 进制浮点数值，存入目的地组件.



♦ Reserved

◎ 2 进制浮动小数点加算

FNC(120)			
D	EADD	P	32 bits:(D)EADD & (D)EADD(P) - - - - - - - - - - - - 13 steps

			J2n--

Operands: [S1.] [S2.] [S1.] [S2.]

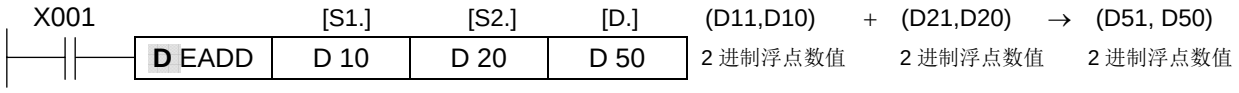
K,H	KnX	KnY	KnM	KnS	T	C	D	V,Z

[D.]

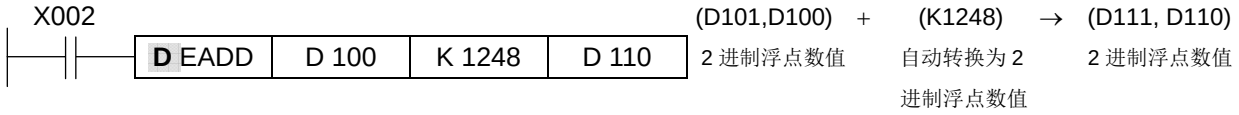
Operands:

X	Y	M	S

影响旗号: 无



- 将 2 个储存 2 进制浮点数值的组件作加算后，以 2 进制浮点数值的形式存入目的地组件中。
- 来源操作数，若以常数 K 或 H 指定时，则自动地将其转换为 2 进制浮点数值作处理。



- 可以指定来源操作数[S.]和目的地操作数[D.]为相同的组件编号。此时，若使用连续执行型命令，则因每个扫瞄周期均会执行加算。

◎ 2 进制浮动小数点减算

FNC(121)			
D	ESUB	P	32 bits:(D)ESUB & (D)ESUB(P) - - - - - - - - - - - - 13 steps

			J2n--

Operands: [S1.] [S2.] [S1.] [S2.]

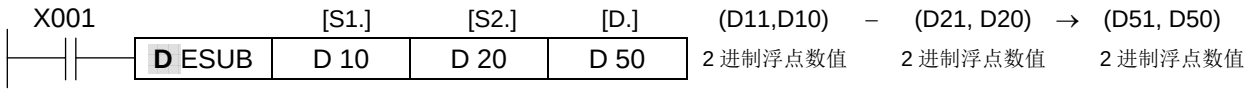
K,H	KnX	KnY	KnM	KnS	T	C	D	V,Z

[D.]

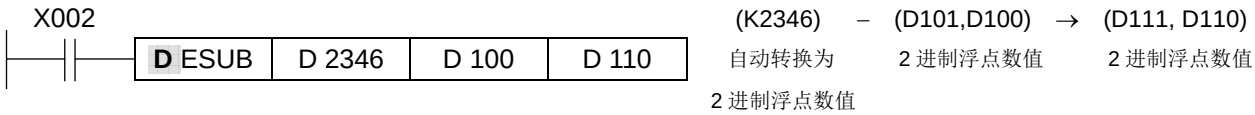
Operands:

X	Y	M	S

影响旗号: 无



- [S1.]的 2 进制浮点数值减去[S2.]的 2 进制浮点数值，将其结果以 2 进制浮点数值形式存放在目的地组件[D.]。
- 来源操作数，若以常数 K 或 H 指定时，则自动地将其转换为 2 进制浮点数值来作处理。



- 可以指定来源操作数[S.]和目的地操作数[D.]为相同的组件编号。此时，若使用连续执行型命令，则因每个扫瞄周期均会执行减算。

◎ 2 进制浮动小数点乘算

FNC(122)			
D	EMUL	P	32 bits:(D)EMUL & (D)EMUL(P) - - - - - - - - - - - - 13 steps

			J2n--

Operands: |←→| [S1.] [S2.] [S1.] [S2.] |←→|

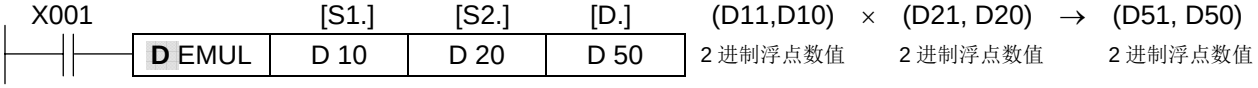
K,H	KnX	KnY	KnM	KnS	T	C	D	V,Z

|←[D.]→|

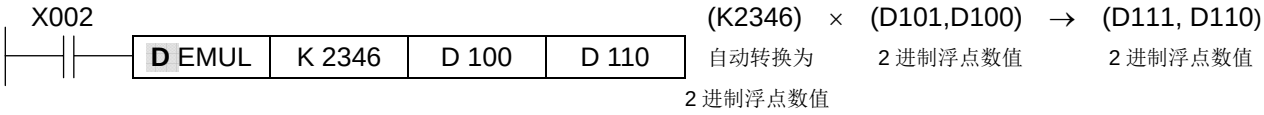
Operands:

X	Y	M	S

影响旗号: 无



- ♦ 将 2 个来源组件[S1.]和[S2.]的 2 进制浮点数值相乘后，将其结果以 2 进制浮点数值的形式存放在目的地组件[D.]。
- ♦ 若来源操作数以常数 K 或 H 指定时，则自动地将其转换为 2 进制浮点数值来作处理。



◎ 2 进制浮动小数点除算

FNC(123)			
D	EDIV	P	32 bits:(D)EDIV & (D)EDIV(P) - - - - - - - - - - - - 13 steps

			J2n--

Operands: |←→| [S1.] [S2.] [S1.] [S2.] |←→|

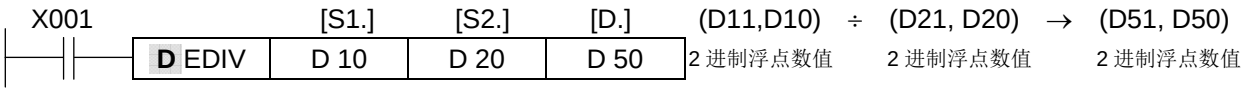
K,H	KnX	KnY	KnM	KnS	T	C	D	V,Z

|←[D.]→|

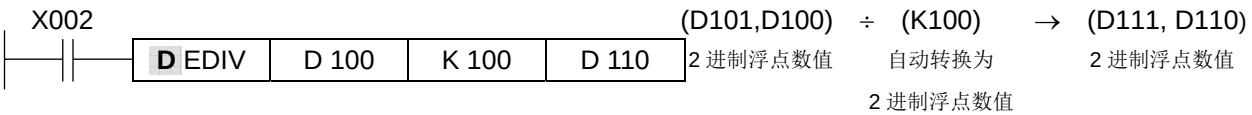
Operands:

X	Y	M	S

影响旗号: 无



- ♦ 将所指定组件[S1.]的 2 进制浮点数值，除以[S2.]指定组件的 2 进制浮点数值，并将其结果以 2 进制浮点数值的形式存放在目的地组件[D.]。
- ♦ 来源操作数，若以常数 K 或 H 指定时，则自动地将其转换为 2 进制浮点数值来作处理。



◎ 2 进制浮动小数点开平方根

FNC(127)			
D	ESQR	P	32 bits:(D)ESQR & (D)ESQR(P) - - - - - - - - - - - 13 steps

			J2n--

Operands: |←[S.]→| |←[S.]→|

K,H	KnX	KnY	KnM	KnS	T	C	D	Z

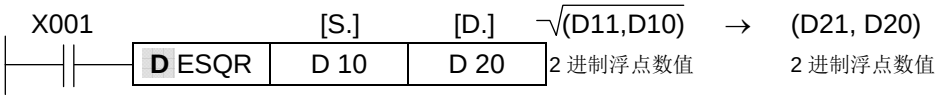
|←[D.]→|

Operands:

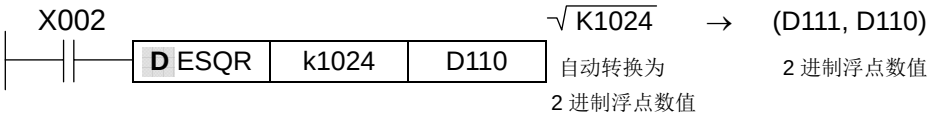
X	Y	M	S

[S.]的内容只有为正数才有效

影响旗号: M8020



- 将来源[S.]的 2 进制浮点数值的组件做开平方根演算，并将其结果以 2 进制浮点数值储放在[D.]。
- 若来源操作数，以常数 K 或 H 指定时，则自动地将其转换为 2 进制浮点数值来作处理。



- 演算结果若确实为 0，则零旗标 M8020 会 ON。
- 来源操作数的内容，仅在为正数时才有效。若为负数时，则演算异常 M8067 会 ON，并停止执行命令。

◎ 2 进制浮动小数点→BIN 整数变换

FNC(129)			16 bits:INT & INT - 5 steps
D	INT	P	32 bits:(D)INT & (D)INT(P) - - - - - - - - - - - - - - - - -9 steps

			J2n--

Operands: |←[S.]→|

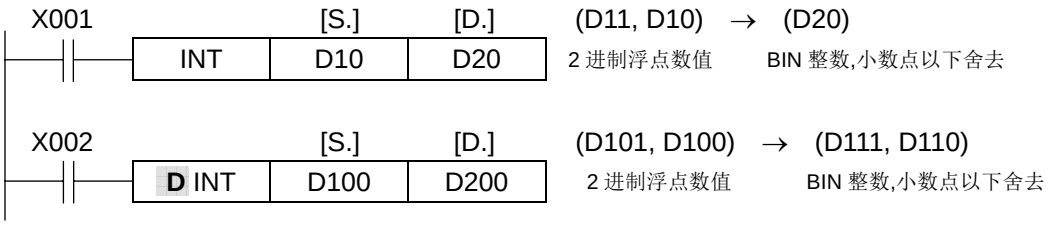
K,H	KnX	KnY	KnM	KnS	T	C	D	V,Z

|←[D.]→|

Operands:

X	Y	M	S

影响旗号:



- 将所指定组件[S.]组件的 2 进制浮点数值转换成 BIN 整数并存放在目的地组件[D.]。此时小数点以下的数值被舍去。
- 本命令为 FNC49 (FLT) 命令的相反转换命令。
- 演算结果为 0 时，则零旗标 M8020 为 ON。
  转换时因不足 1 而舍去成为 0 时，则负号旗标 M8021 会 ON。
  演算结果若超出下记范围会发生溢位，且进位旗标 M8022 会 ON。
  16 位演算时 **：**-32,768~32,767
  32 位演算时 **：**-2,147,483,648~2,147,483,647

◎ 浮动小数点 SIN 演算

FNC(130)			
D	SIN	P	32 bits:(D)SIN & (D)SIN(P) - - - - - - - - - - - - - - - - 9 steps

			J2n--

Operands:

K,H	KnX	KnY	KnM	KnS	T	C	D ←[S.]→ ←[D.]→	V,Z

Operands:

X	Y	M	S

0°≦角度<360°

影响旗号:

X000 [S.] [D.] (D51, D50)RAD → (D61, D60)SIN

D SIN	D50	D60

2 进制浮点数值　　2 进制浮点数值

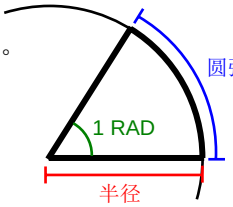
♦ 将来源 [S.] 所指定的弧度 (RAD) ，求取其 SIN 值，并将结果存放在目的地组件 [D.]。

[S.]	D 51	D 50
[D.]	D 61	D 60

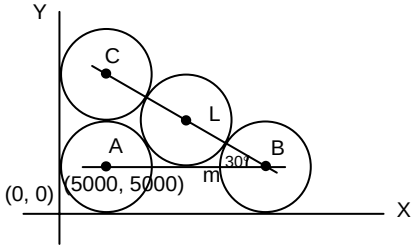
RAD 值 (角度×π/180)
指定为 2 进制浮点数值

SIN 值
2 进制浮点数值

♦单位弧度定义: 圆弧长度=半径时的圆心角。记为 RAD。
♦1 rad = 180/π° ; 1°=π/180 rad



♦求 m 的长度



M8002

D MOV P	K 60	D 0

∠C=60° → (D0) 2 进制整数值

M8000

D FLT	D 0	D 4

将∠C 转换为 2 进制浮点数值→(D5,D4)

D EDIV	K31415926	K1800000000	D 20

(π/180) → (D21,D20)

D EMUL	D 4	D 20	D 30

(D5,D4)角度×(π/180) → (D31,D30)RAD
2 进制浮点数值

D SIN	D 30	D 32

(D31,D30)RAD → (D33,D32)SIN 值
2 进制浮点数值

D MUL	K 10000	K 2	D 40

直线 L 的长度为 2 倍直径

D FLT	D 40	D 42

将直线 L 整数数值转换为 2 进制浮动小数点格式

D EMUL	D 42	D 32	D 100

D100 为直线 m 的 2 进制浮动小数点数值

D INT	D 100	D 200

D200 为直线 m 的 2 进制整数值

◎ 浮动小数点 COS 演算

FNC(131)			
D	COS	P	32 bits:(D)COS & (D)COS(P) - - - - - - - - - - - - - 9 steps

			J2n--

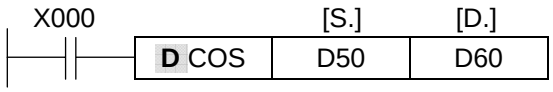
Operands:

K,H	KnX	KnY	KnM	KnS	T	C	D ←[S.]→	V,Z ←[D.]→

Operands:

X	Y	M	S	0°≦角度<360°

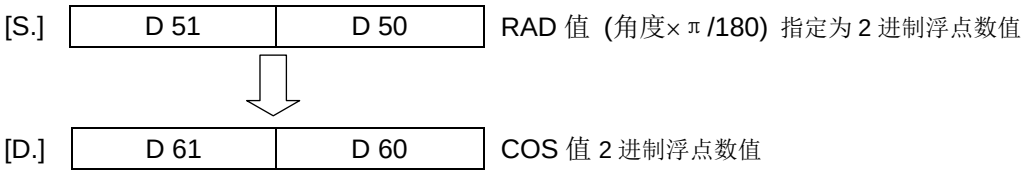
影响旗号:



(D51, D50)RAD → (D61, D60)COS

2 进制浮点数值　　2 进制浮点数值

♦ 将来源组件[S.]所指定的角度(RAD)，求取其 COS 值，并将结果存放在目的地组件[D.]。



◎ 浮动小数点 TAN 演算

FNC(132)			
D	TAN	P	32 bits:(D)TAN & (D)TAN(P) - - - - - - - - - - - - - - 9 steps

			J2n--

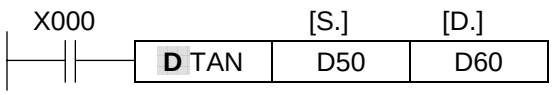
Operands:

K,H	KnX	KnY	KnM	KnS	T	C	D ←[S.]→	V,Z ←[D.]→

Operands:

X	Y	M	S	0°≦角度<360°

影响旗号:



(D51, D50)RAD → (D61, D60)TAN

2 进制浮点数值　　2 进制浮点数值

♦ 将来源组件[S.]所指定的角度(RAD) ，求取其 TAN 值，并将结果存放在目的地组件[D.]。

◎ 上下字节(Byte)交换

FNC(147)			16 bits:SWAP & SWAP(P) - - - - - - - - - - - - - - - - - 5 steps
D	SWAP	P	32 bits:(D)SWAP & (D)SWAP(P) - - - - - - - - - - - -9 steps

		J1n	J2n--

Operands:

[S.]

K,H	KnX	KnY	KnM	KnS	T	C	D	Z

Operands:

X	Y	M	S

影响旗号:

X000 [S.]

SWAP **P** D10

♦ 在 16 位命令的场合，将下 8 位和上 8 位互相交换。

D10

上 8 位	下 8 位

X000 [S.]

D SWAP **P** D10

♦ 在 32 位命令的场合，将各别的上 8 位和下 8 位互相交换。

D11

上 8 位	下 8 位

，

D10

上 8 位	下 8 位

♦ 若使用连续执行命令，因每个扫瞄周期均会执行交换处理，请特别注意。

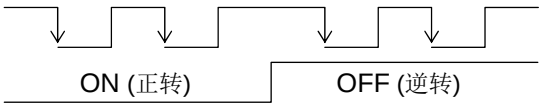
♦ 本命令和 FNC17 (XCH)命令的扩张机能动作相同。

◎ FNC150 – 159 定位控制概述

♦此系列控制器脉波输出信号是以”脉波列(负逻辑)+符号”的形态，如下图

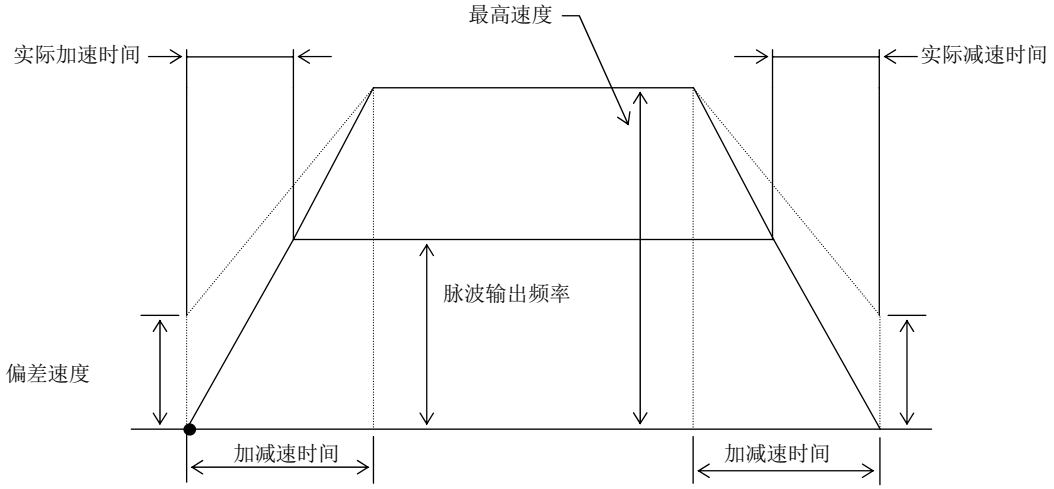
固定以 Y00, Y01 为脉波输出点

固定 Y02,Y03 为方向输出点



♦ 脉波导通周期(duty cycle) 50% ON 50% OFF。

♦ 1 段位置驱动曲线情形(定斜率模式)及相关组件



◎ ABS 现在值读出

FNC(155)			16 bits:ABS - 7 steps
D	ABS		32 bits:(D)ABS - 11 steps

Operands:

[S.]

K,H	KnX	KnY	KnM	KnS	T	C	D	Z

Operands:

X	Y	M	S

影响旗号: M8029

Reserved

◎ 原点复归 Zero Return

FNC(156)		
D	ZRN	

32 bits:(D)ZRN - 17 steps

		J1n	J2n--

Operands: [S1.]

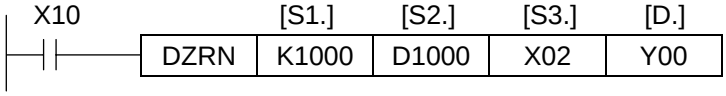
K,H	KnX	KnY	KnM	KnS	T	C	D	Z

[S2.]

Operands: [S3.] [D.]

X	Y	M	S

影响旗号: M8029



♦ [D.] 指定脉波输出点。

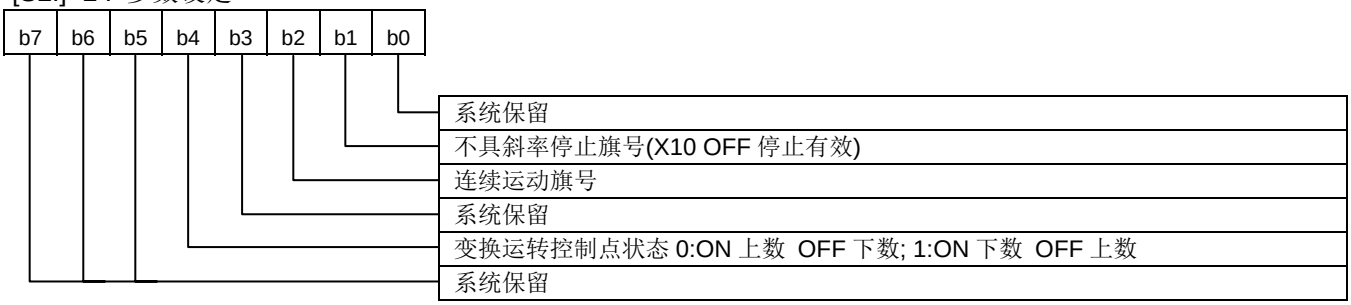
[S1.] 指定原点复归找寻近点速度(Home Speed) 10 ~ 200,000 pps。

[S2.] 从指定的[S2.]开始会连续占用 100 个 words。在本例中，占用 D1000~D1099

[S2.]+0 : 寻找零点速度 10~32,767 pps

[S2.]+1 : 运转方向控制点 Y2~Y7

[S2.]+2 : 参数设定



[S2.]+3 ~ [S2.]+99 : 与 FNC(59) PLSR 的[S3.]+3 ~ [S3.]+99 相同

[S3.] 指定近点输入信号 ，有效范围 X00~X07 (脉波截取信号 M8170~M8177)。

零点信号由[S2.]+24 设定。

♦ 当执行 ZRN 指令时，归零点忙碌旗标 M8154~M8157 将被自动设定，避免同时驱动 DRVI,DRVA。

♦ 这个指令 Y00 到 Y03 只能使用一次而且必须选择晶体管输出模块。

♦ 固定为 32 位运算。若指定 16 位元运转模式，则产生 error 6509。

可调变速脉波输出

FNC(157)			
D	PLSV		32 bits:(D)PLSV - 13 steps

		J1n	J2n--

Operands: |< [S.] →|

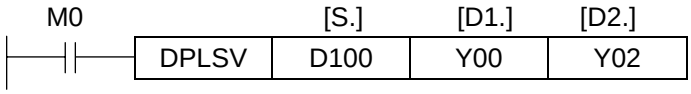
K,H	KnX	KnY	KnM	KnS	T	C	D	Z

Operands: |<[D2.]→|

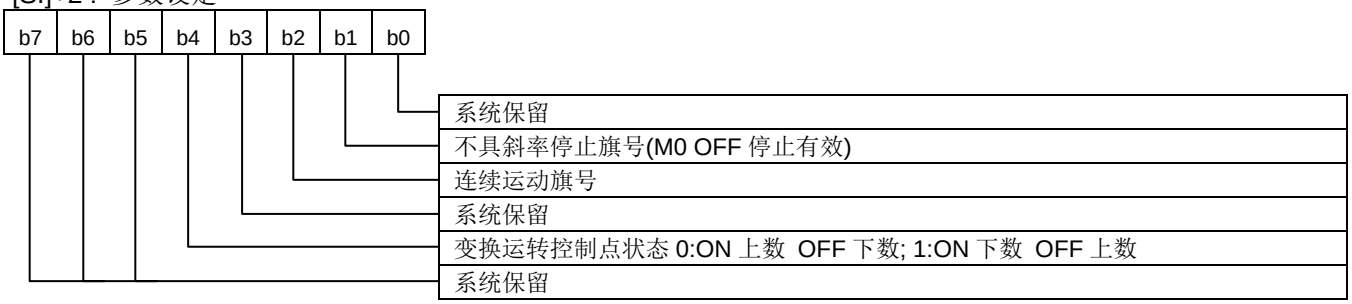
X	Y	M	S

|<[D1.]→|

影响旗号: M8029



♦ [D1.] 指定运转脉波输出点。(固定以 Y00~Y03 为输出点)。
[D2.] 指定运转方向输出点。(固定以 Y02~Y07 为输出点)。
[S.] 从指定的[S.]开始会占用连续 100 个 words。在本例中，占用 D1000~D1099。
[S.]+1, [S]+0 : 指定输出频率。[32bits]:10 ~ 200,000 Hz。
[S.]+2 : 参数设定



[S.]+41, [S.]+40 : PLSV 输出脉波数。数值等于 0 为无目标运转。
[S.]+3 ~ [S.]+99 : 与 FNC(59) PLSR 的[S3.]+3 ~ [S3.]+99 相同

♦ 当执行 PLSV 指令时，则忙碌旗标 M8142~M8145 将会被自动设定。
♦ 脉波输出中可任意变更[S.]的内容值，但符号(+,-)不可变更，若驱动接点 OFF 直接减速至启动速度停止。
♦ 固定为 32 位运算。若指定 16 位元运转模式，则产生 error 6509。
♦ 下列模式均可达成

◎ 相对地址定位控制

FNC(158)			
D	DRVI		32 bits:(D)DRVI - 17 steps

		J1n	J2n--

Operands: [S1.]

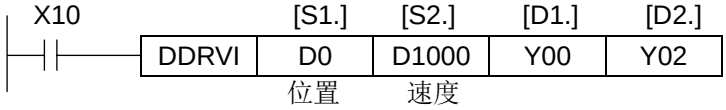
K,H	KnX	KnY	KnM	KnS	T	C	D	Z

[S2.]

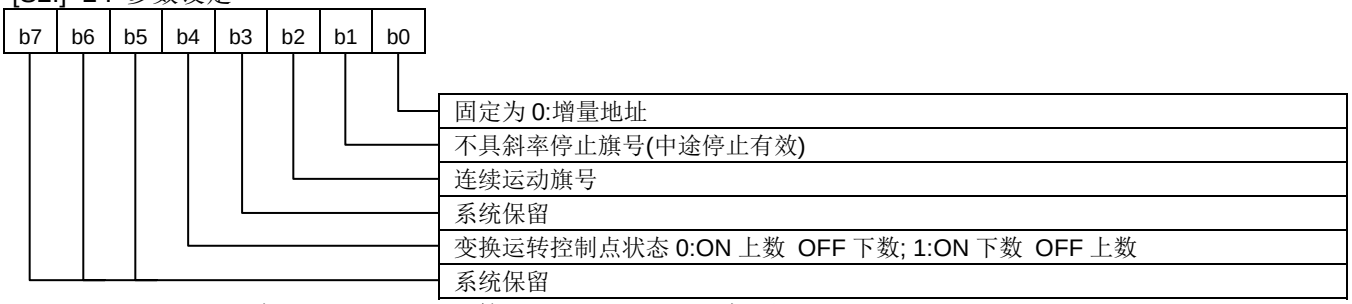
Operands: [D1.] [D2.]

X	Y	M	S

影响旗号: M8029

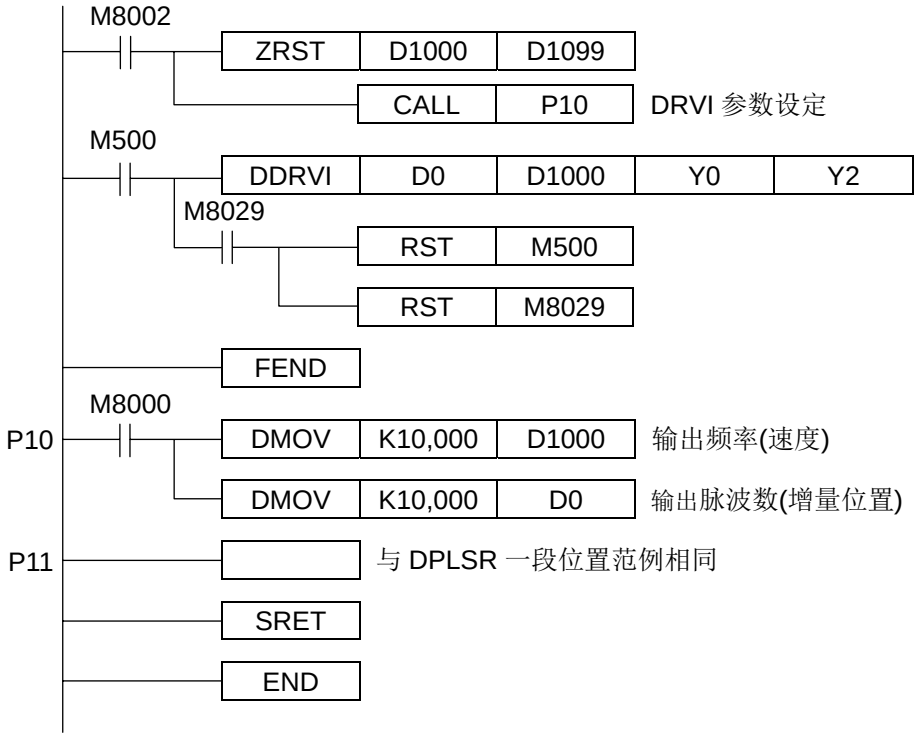


♦ [D1.] 指定运转脉波输出点。(固定以 Y00~Y03 为输出点)。
[D2.] 指定运转方向输出点。(固定以 Y02~Y07 为输出点)。
[S1.] 指定相对位置输出脉波数。(正值:正转，负值:逆转)
[S2.] 从指定的[S2.]开始会占用连续 100 个 words。在本例中，占用 D1000~D1099。
[S2.]+1, [S2]+0：指定输出频率。[32bits]:10 ~ 200,000 Hz。
[S2.]+2：参数设定



[S2.]+3 ~ [S2.]+99：与 FNC(59) PLSR 的[S3.]+3 ~ [S3.]+99 相同

♦ 这个指令 Y00 到 Y03 只能使用一次，而且必须选择晶体管输出模块。
♦ 当执行 DDRVI 指令时，则忙碌旗标 M8146~M8149 将会被自动设定。
♦ 输出脉波中，修改 [S1], [S2]+1, [S2]+0 的内容值无效。
♦ 固定为 32 位运算。若指定 16 位元运转模式，则产生 error 6509。



◎ 绝对地址定位控制

FNC(159)			
D	DRVA		32 bits:(D)DRVA - 17 steps

		J1n	J2n--

Operands: [S1.]

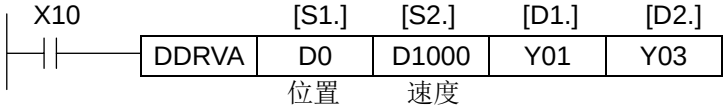
K,H	KnX	KnY	KnM	KnS	T	C	D	Z

[S2.]

Operands: [D1.] [D2.]

X	Y	M	S

影响旗号: M8029

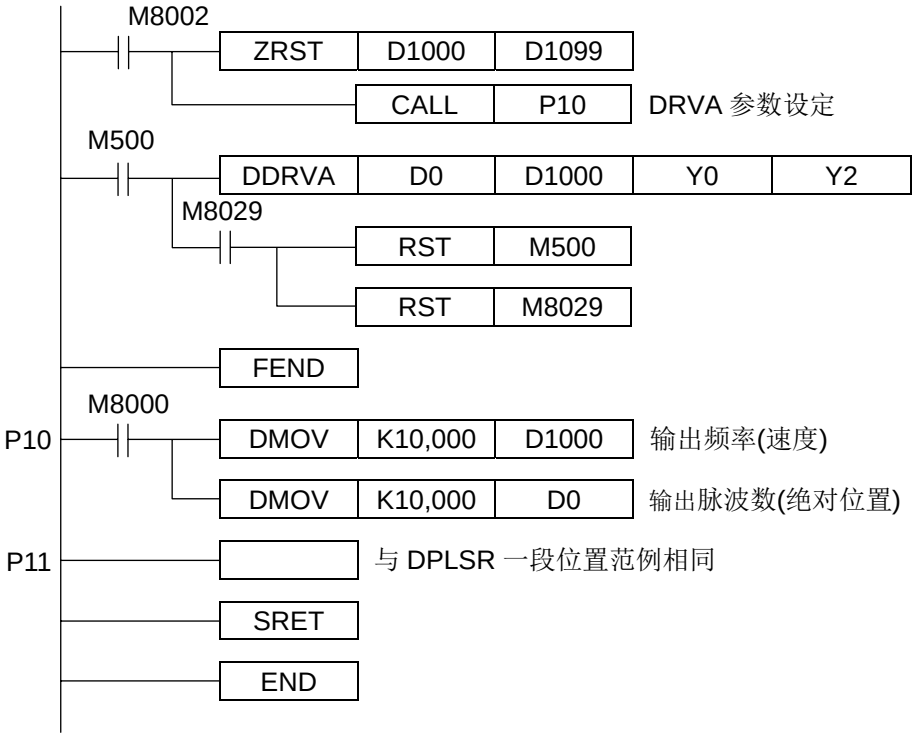


♦ [D1.] 指定运转脉波输出点。(固定以 Y00~Y03 为输出点)。
[D2.] 指定运转方向输出点。(固定以 Y02~Y07 为输出点)。
[S1.] 指定绝对位置输出脉波数。(与启始地址比较，决定正、逆转)
[S2.] 从指定的[S2.]开始会占用连续 100 个 words。在本例中，占用 D1000~D1099。
[S2.]+1, [S2]+0 : 指定输出频率。[32bits]:10 ~ 200,000 Hz。
[S2.]+2 : 参数设定

b7	b6	b5	b4	b3	b2	b1	b0	
								固定为 1: 绝对地址
								不具斜率停止旗号(中途停止有效)
								连续运动旗号
								系统保留
								变换运转控制点状态 0:ON 上数 OFF 下数; 1:ON 下数 OFF 上数
								系统保留

[S2.]+3 ~ [S2.]+99 : 与 FNC(59) PLSR 的[S3.]+3 ~ [S3.]+99 相同

♦ 这个指令 Y00 到 Y03 只能使用一次，而且必须选择晶体管输出模块。
♦ 当执行 DDRVA 指令时，则忙碌旗标 M8150~M8153 将会被自动设定。
♦ 输出脉波中，修改 [S1], [S2]+1, [S2]+0 的内容值无效。
♦ 固定为 32 位运算。若指定 16 位元运转模式，则产生 error 6509。



◎ 时钟数据比较

FNC(160)			16 bits:TCMP & TCMP(P) - - - - - - - - - - - - - - - - - 5 steps
	TCMP	P	

		J1n	J2n--

Operands: [S1.][S2.][S3.]

K,H	KnX	KnY	KnM	KnS	T	C	D	V,Z

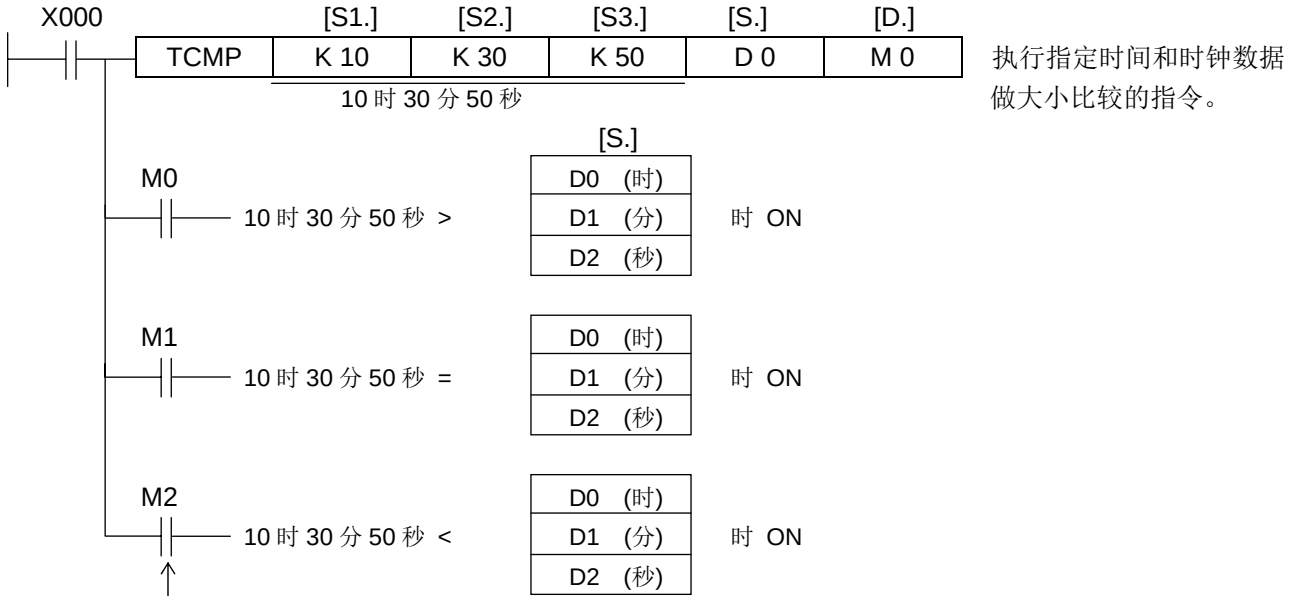
[S.]

Operands: [D.]

X	Y	M	S

[D.]占用连续 3 点

影响旗号: M8020, M8021, M8022



执行指定时间和时钟数据做大小比较的指令。

若将 X000 OFF 时，则不执行 TCMP 命令，但 M0~M2 仍保持 OFF 前的状态。

- 来源组件「[S1.],[S2.],[S3.]」的时间和储存在[S.]开始连续 3 点的时间数据做比较，依比较结果，[D.]开始连续 3 点的组件，自动地 ON/OFF。

[S1.] ：比较时间的 “时” 指定为「0~23」时。
[S2.] ：比较时间的 “分” 指定为「0~59」分。
[S3.] ：比较时间的 “秒” 指定为「0~59」秒。

[S.] ：比较时间的 “时” 指定为「0~23」时。
[S.] + 1：比较时间的 “分” 指定为「0~59」分。
[S.] + 2：比较时间的 “秒” 指定为「0~59」秒。

[D.], [D.] +1, [D.] +2：依比较结果 [D.] 开始连续 3 点的组件自动 ON/OFF。

- 时钟数据可以利用控制器的内部时钟(Real Time Clock)数据，时钟数据储存在下记特殊缓存器 D8015(时)，D8014(分)，D8013(秒)。

◎ 时钟数据比较

FNC(161)			16 bits:TZCP & TZCP(P) - - - - - - - - - - - - - - - - - - 9 steps
	TZCP	P	

		J1n	J2n--

Operands: [S1.][S2.][S3.]

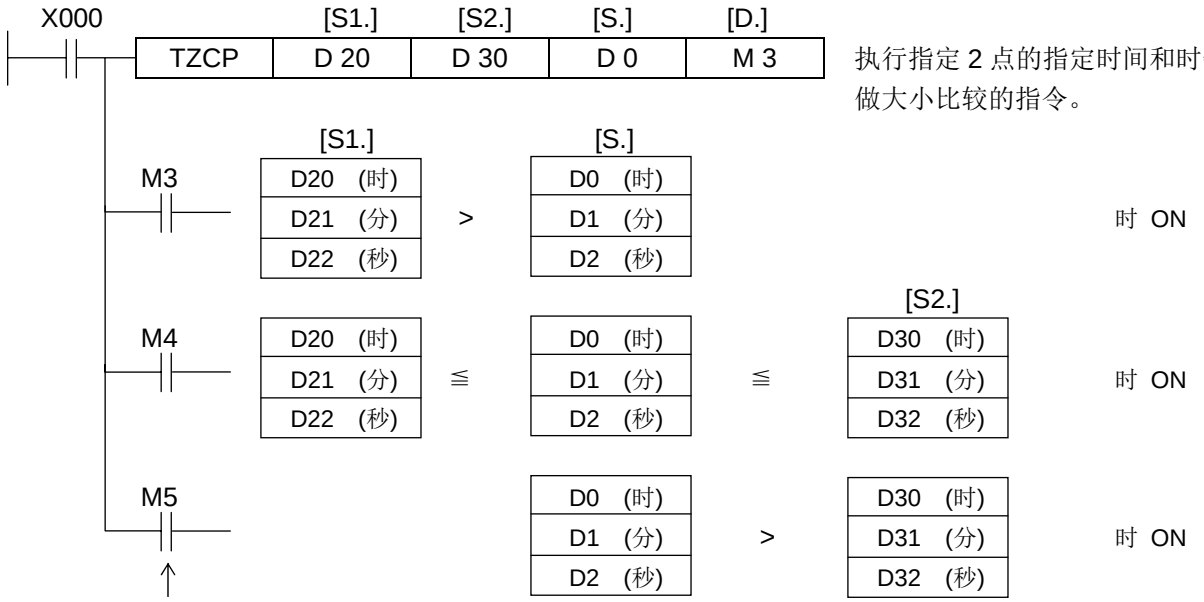
K,H	KnX	KnY	KnM	KnS	T	C	D	V,Z

Operands: [D.]

X	Y	M	S

占用[D.]开始连续 3 点，请设定为[S1.]≦[S2.]

影响旗号: M8020, M8021, M8022



若将 X000 OFF 时，则不执行 TZCP 命令，但 M3~M5 仍保持 X000 OFF 前的状态。

♦ 上下点的比较范围和[S.]开始连续 3 点的时间数据区域做比较，对应比较大小的结果，[D.]开始连续 3 点自动地 ON/OFF。

[S1.], [S.] +1, [S.] +2 **：** 比较范围的下限，指定“时”，“分”，“秒”。
[S2.], [S2.] +1, [S2.] +2 **：** 比较范围的上限，指定“时”，“分”，“秒”。

[S.], [S.] +1, [S.] +2 **：** 时钟数据，指定“时”，“分”，“秒”。
[D.], [D.] +1, [D.] +2 **：** 依区域比较结果，[D.]开始连续 3 点的字节件自动 ON/OFF。

“时”，“分”，“秒”的设定范围和控制器的内部时钟的处理，请参考 FNC160 (TCMP)命令。

◎ 时钟资料的加算

FNC(162)			16 bits: TADD & TADD(P) - - - - - - - - - - - - - - - - - 7 steps
	TADD	P	

		J1n	J2n--

Operands: |←——[S1.][S2.]——→|

K,H	KnX	KnY	KnM	KnS	T	C	D	V,Z

|←—— [D.] ——→|

Operands:

X	Y	M	S

影响旗号: M8020, M8021, M8022

X000 [S1.] [S2.] [D.]

TADD	D 10	D 20	D 30

(D10,D11,D12)+(D20,D21,D22) →(D30,D31,32)

[S1.]		[S2.]		[D.]
D10 10(时)		D20 3(时)		D30 13(时)
D11 30(分)	+	D21 10(分)	→	D31 40(分)
D12 10(秒)		D22 5(秒)		D32 15(秒)
10时30分10秒		3时10分5秒		13时40分15秒

♦ 将储存在[S1.]开始连续 3 点的时间数据加上[S2.]开始连续 3 点的时间数据，并将加算结果存放在[D.]开始连续 3 点的组件。

♦ 演算结果若超过(24)时，则进位旗标 M8022 ON。并将加总数值减去 24 后的数值存放在[D.]。

[S1.]		[S2.]		[D.]
18 (时)		10 (时)		4 (时)
10 (分)	+	20 (分)	→	30 (分)
30 (秒)		5 (秒)		35 (秒)
18时10分10秒		10时20分5秒		4时30分35秒

♦ 演算结果为 0 (0 时 0 分 0 秒) ，则零旗标 M8020 会 ON。

♦ “时”，“分”，“秒”的设定范围和控制器的内部时钟的处理，请参考 FNC160 (TCMP)命令。

◎ 时钟资料的减算

FNC(163)			16 bits: TSUB & TSUB(P) - - - - - - - - - - - - - - - - - 7 steps
	TSUB	P	

		J1n	J2n--

Operands:

|←——[S1.][S2.]——→|

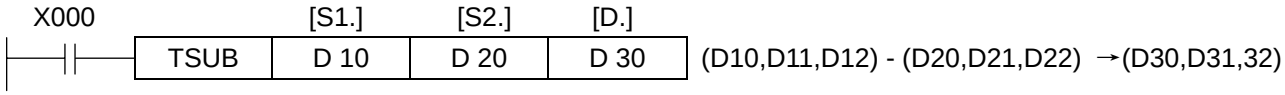
K,H	KnX	KnY	KnM	KnS	T	C	D	V,Z

|←—— [D.] ——→|

Operands:

X	Y	M	S

影响旗号: M8020, M8021, M8022



[S1.]		[S2.]		[D.]
D10 10(时)		D20 3(时)		D30 7(时)
D11 30(分)	–	D21 10(分)	→	D31 20(分)
D12 10(秒)		D22 5(秒)		D32 5(秒)
10时30分10秒		3时10分5秒		7时20分5秒

♦ 将储存在[S1.]开始连续 3 点的时间数据，减去[S2.]开始连续 3 点的时间数据，并将结果存放在[D.]开始连续 3 点的组件。

♦ 演算结果为 0 时以下负位旗标 ON，并将减算结果加上 24 后的数值存入[D.]中。

[S1.]		[S2.]		[D.]
5 (时)		18 (时)		11 (时)
20 (分)	–	10 (分)	→	10 (分)
40 (秒)		5 (秒)		35 (秒)
5时20分40秒		18时10分5秒		11时10分35秒

♦ 演算结果为 0 (0 时 0 分 0 秒) ，则零旗标 M8020 会 ON。

♦ “时”，“分”，“秒”的设定范围和控制器的内部时钟的处理，请参考 FNC160(TCMP)命令。

◎ 时钟资料的读出

FNC(166)			16 bits: TRD & TRD(P) - - - - - - - - - - - - - - - - - - - 5 steps
	TRD	P	

		J1n	J2n--

Operands:

K,H	KnX	KnY	KnM	KnS	T	C	D	V,Z

|← [D.] →|

Operands:

X	Y	M	S

占用[D.]开始连续 7 点

影响旗号:

读出 PLC 内部时钟的数据，存放在 [D.] 开始连续 7 点的目的地组件

- 控制器内部时钟的时间数据，以如下格式读出。
- 读出的时间数据系储存在特殊缓存器 D8013~D8019。

	元 件	项 目	时 锺 资 料		元 件	项 目
特殊内部暂时存锺器用	D8018	年(公历)	0~99(公历下 2 位)	→	D0	年(公历)
	D8017	月	1~12	→	D1	月
	D8016	日	1~31	→	D2	日
	D8015	时	0~23	→	D3	时
	D8014	分	0~59	→	D4	分
	D8013	秒	0~59	→	D5	秒
	D8019	星期	0(日)~6(六)	→	D6	星期

◎ 时钟资料的写入

FNC(167)			16 bits: TWR & TWR(P) - - - - - - - - - - - - - - - - - - 5 steps
	TWR	P	

		J1n	J2n--

Operands: |←——— [S.] ———→|

K,H	KnX	KnY	KnM	KnS	T	C	D	V,Z

Operands:

X	Y	M	S

影响旗号:

将时间数据写入控制器的内部时钟的命令。

- 将时间数据写入控制器的内部时钟。
- 将欲写入的时间数据先存放在所指定 [S.] 的连续 7 点组件中。

	元 件	项 目	时 锺 资 料		元 件	项 目	
时间设定用资料	D10	年(公历)	0~99(公历下 2 位)	→	D8018	年(公历)	特殊资料暂存器 内部时锺用
	D11	月	1~12	→	D8017	月	
	D12	日	1~31	→	D8016	日	
	D13	时	0~23	→	D8015	时	
	D14	分	0~59	→	D8014	分	
	D15	秒	0~59	→	D8013	秒	
	D16	星期	0(日)~6(六)	→	D8019	星期	

执行 FNC167 (TWR)命令后，内部时钟的时间数据，立即变更为所写入的新设时间值。因此，请先将数分钟后数值写入[S.]组件，待实际时间到达时再执行 TWR 命令，而且利用本命令来校正时间，则不需再使用特殊继电器 M8015(时间停止和时间校正)。

◎ 格莱码(GRAY CODE)转换

FNC(170)			16 bits:GRY & GRY(P) - - - - - - - - - - - - - - - - - - - 5 steps
D	GRY	P	32 bits:(D)GRY & (D)GRY(P) - - - - - - - - - - - - - - -9 steps

		J1n	J2n--

Operands: [S.]

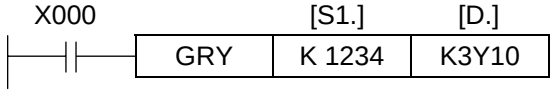
K,H	KnX	KnY	KnM	KnS	T	C	D	V,Z

[D.]

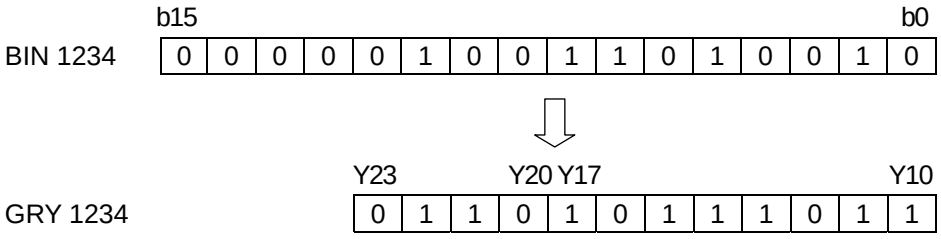
Operands:

X	Y	M	S

影响旗号:



执行来源操作数[S.](BIN)→目的地操作数 [D.] (GRY) 转换传送的命令。



- 将 BIN 数值转换为 GRAY CODE 的转换传送命令。
  数据的转换传送速度，由控制器的扫瞄时间而定。
- 使用 **D** GRY 命令时，可执行最多 32 位的 GRAY CODE 转换。
- 对于[S.]有效的数值范围如下，
  16 位演算时 **：**0~32,767
  32 位演算时 **：**0~2,147,483,647

◎ 格莱码(GRAY CODE)相反转换

FNC(171)			16 bits:GBIN & GBIN(P) - - - - - - - - - - - - - - - - - - - 5 steps
D	GBIN	P	32 bits:(D)GBIN & (D)GBIN(P) - - - - - - - - - - - - - -9 steps

		J1n	J2n--

Operands:

[S.]

K,H	KnX	KnY	KnM	KnS	T	C	D	V,Z

[D.]

Operands:

X	Y	M	S

影响旗号:

X000 [S1.] [D.]

GIBN	K3X0	D 10

执行 X000~X013(GRY)→D10(BIN) 转换传送命令。

	X13			X10	X7						X0
GRY 1234	0	1	1	0	1	0	1	1	1	0	1	1

	b15															b0
BIN 1234	0	0	0	0	0	1	0	0	1	1	0	1	0	0	1	0

- 将 GRAY CODE 转换成 BIN 数值的转换传送命令，可利用 GRAY CODE 方式的编码器，作绝对位置的检出。
- 指定 [S.] 为输入(X)时，会有「扫瞄时间+输入滤波器常数」的反应延迟时间。
- 使用 FNC51 (REFF)命令或改变输入滤波器常数值 D8020 (滤波器调整)，则可消除滤波器常数部分的反应延迟时间。
- 使用 **D** GBIN 命令时，可执行最多 32 位的 GRAY CODE 相反转换。
- 对于 [S.] 有效的数值范围如下，

  16 位演算时 **：**0~32,767

  32 位演算时 **：**0~2,147,483,647

◎ 接点型比较命令演算开始 LD ※

FNC(224~230)			16 bits: - 5 steps
D	LD ※		32 bits: - 9 steps

		J1n	J2n--

※ : = , > , < , <> , ≦ , ≧

Operands: [S1.][S2.]

K,H	KnX	KnY	KnM	KnS	T	C	D	V,Z

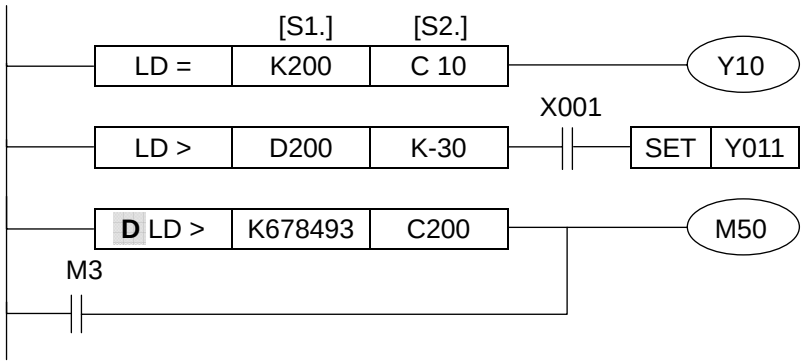
Operands:

X	Y	M	S

影响旗号:

♦ 2 个来源操作数的内容做 BIN 比较，对应比较的结果，执行后段的顺序程序。LD ※ 为连接母线的接点型比较命令。

FNC No.	16 位命令	32 位命令	导通条件	非导通条件
224	LD =	**D** LD =	[S1.] = [S2.]	[S1.] ≠ [S2.]
225	LD >	**D** LD >	[S1.] > [S2.]	[S1.] ≦ [S2.]
226	LD <	**D** LD <	[S1.] < [S2.]	[S1.] ≧ [S2.]
228	LD < >	**D** LD < >	[S1.] ≠ [S2.]	[S1.] = [S2.]
229	LD ≦	**D** LD ≦	[S1.] ≦ [S2.]	[S1.] > [S2.]
230	LD ≧	**D** LD ≧	[S1.] ≧ [S2.]	[S1.] < [S2.]



♦ 当来源资料 [S1.][S2.] 的最上位位(16 位命令**:** b15，32 位命令**:** b31)为 1 时，视为负值来执行比较。

♦ 使用 32 位的计数器 (C200~) 做比较时，必须使用 32 位命令来执行。若使用 16 位命令做比较，则会发生程序异常或演算异常。

◎ 接点型比较命令串联接续 AND=, AND>, AND<, AND<>, AND<=, AND>=

FNC(232~238)			16 bits: - 5 steps
D	AND ※		32 bits: - 9 steps

		J1n	**J2n--**

※ : = , > , <, <> , ≦ , ≧

Operands: |←— [S1.][S2.] —→|

K,H	KnX	KnY	KnM	KnS	T	C	D	V,Z

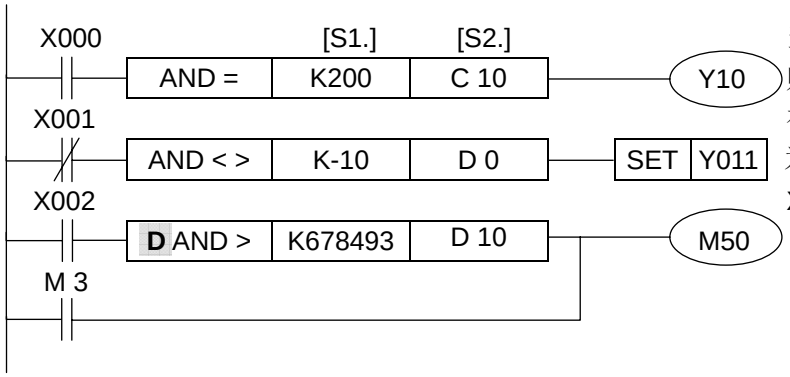
Operands:

X	Y	M	S

影响旗号:

♦ 2 个来源操作数的内容做 BIN 比较，对应比较的结果，执行后段的顺序程序。AND ※ 为和其它接点串联连接的接点型比较命令。

FNC No.	16 位命令	32 位命令	导通条件	非导通条件
232	AND =	**D** AND =	[S1.] = [S2.]	[S1.] ≠ [S2.]
233	AND >	**D** AND >	[S1.] > [S2.]	[S1.] ≦ [S2.]
234	AND <	**D** AND <	[S1.] < [S2.]	[S1.] ≧ [S2.]
236	AND < >	**D** AND < >	[S1.] ≠ [S2.]	[S1.] = [S2.]
237	AND ≦	**D** AND ≦	[S1.] ≦ [S2.]	[S1.] > [S2.]
238	AND ≧	**D** AND ≧	[S1.] ≧ [S2.]	[S1.] < [S2.]



当 X000 ON，计数器 C10 的现在值为 200 时，则 Y010 ON。

在 X001 为 OFF，且数据缓存器的 D0 的内容为-10 时，Y001 SET ON。

X002 为 ON 且数据缓存器 D11, D10 的值比 678,439 小或 M3 为 ON 时，则 M50 会 ON。

♦ 当来源资料 [S1.][S2.] 的最上位位(16 位命令**:**b15，32 位命令**:**b31)为 1 时，视为负值来执行比较。

♦ 使用 32 位的计数器(C200~)做比较时，必须使用 32 位命令来执行。若使用 16 位命令做比较，则会发生程序异常或演算异常。

◎ 接点型比较命令并联接续 OR=, OR>, OR<, OR<>, OR<=, OR>=

FNC(240~246)			16 bits: - 5 steps
D	OR ※		32 bits: - 9 steps

		J1n	J2n--

※ : = , > , < , <> , ≦ , ≧

Operands: |←— [S1.][S2.] —→|

K,H	KnX	KnY	KnM	KnS	T	C	D	V,Z

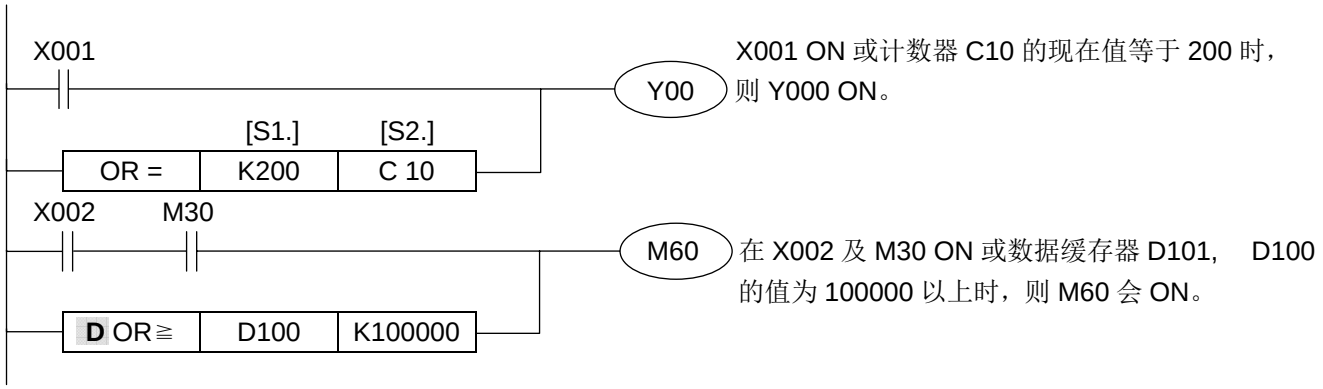
Operands:

X	Y	M	S

影响旗号:

- 2 个来源操作数的内容做 BIN 比较，对应比较的结果，执行后段的顺序程序。OR ※ 为连接母线的接点型比较命令。

FNC No.	16 位命令	32 位命令	导通条件	非导通条件
240	OR =	**D** OR =	[S1.] = [S2.]	[S1.] ≠ [S2.]
241	OR >	**D** OR >	[S1.] > [S2.]	[S1.] ≦ [S2.]
242	OR <	**D** OR <	[S1.] < [S2.]	[S1.] ≧ [S2.]
244	OR < >	**D** OR < >	[S1.] ≠ [S2.]	[S1.] = [S2.]
245	OR ≦	**D** OR ≦	[S1.] ≦ [S2.]	[S1.] > [S2.]
246	OR ≧	**D** OR ≧	[S1.] ≧ [S2.]	[S1.] < [S2.]



X001 ON 或计数器 C10 的现在值等于 200 时，则 Y000 ON。

在 X002 及 M30 ON 或数据缓存器 D101, D100 的值为 100000 以上时，则 M60 会 ON。

- 当来源资料 [S1.][S2.] 的最上位位 (16 位命令**:**b15，32 位命令**:**b31) 为 1 时，视为负值来执行比较。
- 使用 32 位的计数器 (C200~) 做比较时，必须使用 32 位命令来执行。若使用 16 位命令做比较，则会发生程序异常或演算异常。

Note