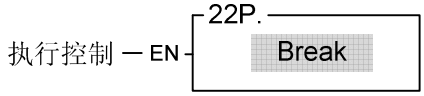
第 7 章：高级篇应用指令

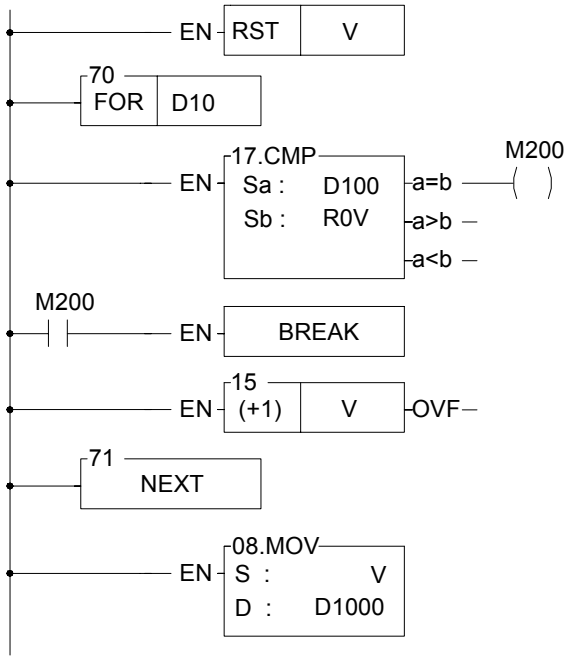
- 流程控制指令一 （FUN22） ……7- 2
- 数学运算指令 （FUN23～33）……7- 3 ～ 7- 18
- 逻辑运算指令 （FUN35～36）……7- 19 ～ 7- 20
- 比较指令 （FUN37） ……7- 21
- 搬移指令一 （FUN40～50）……7- 22 ～ 7- 32
- 位移 / 旋转指令 （FUN51～54）……7- 33 ～ 7- 36
- 数码变换指令 （FUN55～64）……7- 37 ～ 7- 51
- 流程控制指令二 （FUN65～71）……7- 52 ～ 7- 59
- I / O 指令一 （FUN74～86）……7- 60 ～ 7- 73
- 积算型定时器指令 （FUN87～89）……7- 74 ～ 7- 75
- 监控定时器指令 （FUN90～91）……7- 76 ～ 7- 77
- 高速计数器 / 定时器指令 （FUN92～93）……7- 78 ～ 7- 79
- 报表打印指令 （FUN94） ……7- 80
- 缓升 / 缓降指令 （FUN95） ……7- 81 ～ 7- 82
- 列表指令 （FUN100～114）……7- 83 ～ 7- 101
- 矩阵指令 （FUN120～130） ……7- 102 ～7- 113
- I / O 指令二 （FUN139） ……7- 114 ～7- 115
- NC 定位控制指令 （FUN140～143） ……7- 116 ～7- 119
- 中断控制指令 （FUN145～146） ……7- 120 ～7- 121
- 通讯指令 （FUN150～151） ……7- 122 ～7- 123
- 搬移指令二 （FUN160～162） ……7- 124 ～ 7- 129
- 浮点运算指令 （FUN200～213） ……7- 130 ～7- 143

FUN22 P BREAK	FOR 与 NEXT 循环的跳出指令 (BREAK)	FUN22 P BREAK

阶梯图符号



- 当执行控制〝EN〞=1 或〝EN↑〞（P 指令）由 0→1 时，将跳出该 FOR 与 NEXT 构成的循环程序。
- 在 FOR 与 NEXT 指令所构成的循环程序内，如果需要提前跳出该循环，则可使用本指令而不必等到指定的循环次数执行完毕才能跳出该循环。
- 本指令必须使用在 FOR 与 NEXT 指令所构成的循环内。



范例说明：缓存器 D10 的值决定 FOR 与 NEXT 指令所构成的循环程序应被执行次数；循环内程序用来找寻由 R0 为起始的表格内是否有与 D100 相同内容，如有，则停止找寻并跳出循环程序;如没有，则该循环程序会被执行 D10 所指定的次数。M200 的状态反应找寻结果，缓存器 D100 为找寻结果的指标缓存器。

FUN23 P DIV48	48 位除法运算（48-BIT DIVISION） （将 Sa 除以 Sb 所得的商存到 D 去）	FUN23 P DIV48

阶梯图符号

Sa：被除数的起头缓存器号码。
Sb：除数的起头缓存器号码。
D：存放结果(商)的起头缓存器号码。
Sa，Sb，D 可结合 V、Z、P0~P9 作间接寻址应用。

操作数 \ 范围	HR	OR	SR	ROR	DR	XR
	R0 ~ R3839	R3904 ~ R3967	R3968 ~ R4167	R5000 ~ R8071	D0 ~ D4095	V、Z P0~P9
Sa	○	○	○	○	○	○
Sb	○	○	○	○	○	○
D	○	○	○*	○*	○	○

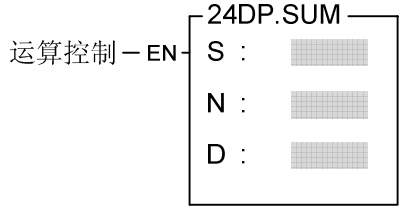
- 当除算控制〝EN〞=1 或〝EN↑〞（P 指令）由 0→1 而〝U/S〞=0 时，本指令将以正负数（Sign）运算法则将 Sa 除以 Sb 所得的商存到 D 去，同时如果商为 0，则 FO0 设为 1，如果除数 Sb=0 则错误旗号 FO1 设为 1 且本指令不执行。
- 当除算控制〝EN〞=1 或〝EN↑〞（P 指令）由 0→1 而〝U/S〞=1 时，本指令将以正整数（Unsign）运算法则将 Sa 除以 Sb 所得的商存到 D 去，同时若商为 0，则 FO0 设为 1，若除数 Sb=0 则错误旗号 FO1 设为 1 且本指令不执行。
- 本指令为 48 位运算，所以 Sa，Sb，D 都占用连续三个缓存器。

- 左图程序范例将 R0 开始到 R2 组成的 48 位被除数除以 R3～R5 组成的除数所获得的商存入 R6～R8 的 48 位缓存器中。

		R2	R1	R0
被除数 Sa		2147483647		
÷	除数 Sb	R5	R4	R3
		1234567		
商 D		R8	R7	R6
		1739		

FUN24 D P SUM	总和计算 （SUM）	FUN24 D P SUM

阶梯图符号



S：来源缓存器的起头号码

N：欲总和的缓存器个数
（由 S 开始连续 N 个）

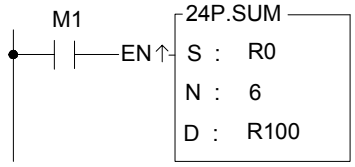
D：存放结果（总和）的缓存器号码

S，N，D 可结合 V、Z、P0~P9 作间接寻址应用

操作数 \ 范围	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR
	WX0 | WX240	WY0 | WY240	WM0 | WM1896	WS0 | WS984	T0 | T255	C0 | C255	R0 | R3839	R3840 | R3903	R3904 | R3967	R3968 | R4167	R5000 | R8071	D0 | D4095	1 | 511	V、Z P0~P9
S	○	○	○	○	○	○	○	○	○	○	○	○		○
N	○	○	○	○	○	○	○	○	○	○	○	○	○	○
D		○	○	○	○	○	○		○	○*	○*	○		○

- 当运算控制〝EN〞=1 或〝EN↑〞（P 指令）由 0→1 时，将 S 开始的连续 N 个 16 位或 32 位（P 指令）缓存器作加法运算，得出总和，并将结果存入 D 所指定的缓存器。
- 当 N 的值为 0 或大于 511 时，运算不执行。
- 通讯端口 1 或通讯端口 2 用来当作泛用 ASCII 通讯接口，如要通讯对象的数据错误检验方式为总和（Check-Sum）检验，则可使用此指令来产生总和值或利用此指令计算总和值并比对看是否数据有误。

〈范例 1〉M1 由 OFF→ON 时，计算 16 位总和

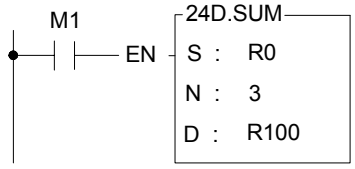


- 左图范例是将 R0 开始的 6 个缓存器以 16 位方式计算总和值，并将结果存入 R100 缓存器。

R0=0030H
R1=0031H
R2=0032H
R3=0033H
R4=0034H
R5=0035H

➔ R100=012FH

〈范例 2〉M1 ON 时，计算 32 位总和



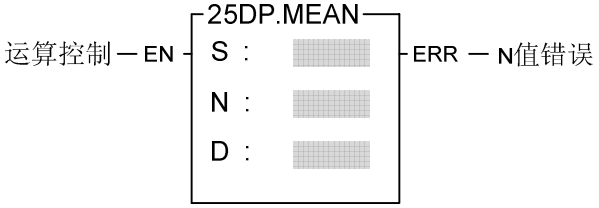
- 左图范例是将 DR0 开始，以 32 位方式计算总和值，并将结果存入 DR100（32 位）缓存器内。

R1～R0=00310030H
R3～R2=00330032H
R5～R4=00410039H

➔ R101～R100=00A5009BH

FUN25 D P MEAN	取平均值 （MEAN）	FUN25 D P MEAN

阶梯图符号



S：来源缓存器的起头号码

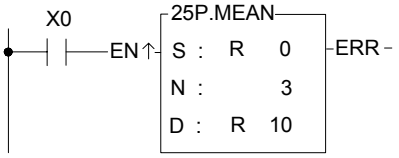
N：要平均的缓存器个数
（由 S 开始连续 N 个）

D：存放结果（平均值）的缓存器号码

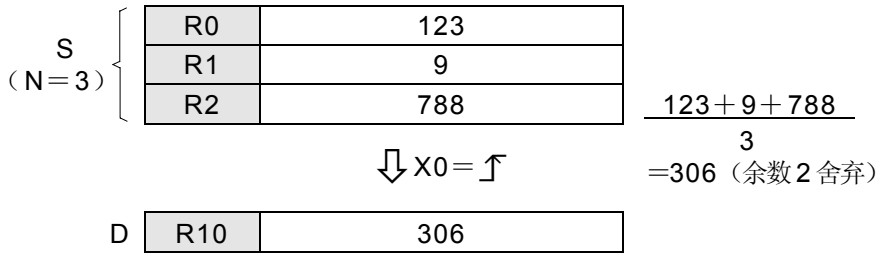
S，N，D 可结合 V、Z、P0~P9 作间接寻址应用

操作数 \ 范围	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR
	WX0 | WX240	WY0 | WY240	WM0 | WM1896	WS0 | WS984	T0 | T255	C0 | C255	R0 | R3839	R3840 | R3903	R3904 | R3967	R3968 | R4167	R5000 | R8071	D0 | D4095	2 | 256	V、Z P0~P9
S	○	○	○	○	○	○	○	○	○	○	○	○		○
N	○	○	○	○	○	○	○	○	○	○	○	○	○	○
D		○	○	○	○	○	○		○	○*	○*	○		○

- 当运算控制〝EN〞＝1 或〝EN↑〞（P 指令）由 0→1 时，将 S 开始的连续 N 个 16 位或 32 位（D 指令）的数值相加再除以 N，所得的平均值（舍弃余数）存入 D 所指定的缓存器。
- 以缓存器内容当 N 值时，若缓存器内容值不是 2～256，则 N 值错误〝ERR〞设为 1，且本指令不执行。



- 左图范例为求从 R0 开始连续 3 个 16 位缓存器的平均值，再将结果存在 16 位缓存器 R10 中。



FUN26 D P SQRT	取平方根值 （SQUARE ROOT）	FUN26 D P SQRT

阶梯图符号

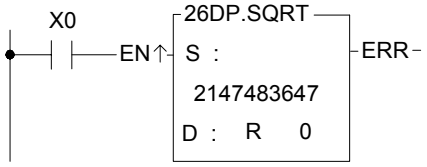
S：求平方根的来源数值或其缓存器号码

D：存放结果（平方根值）的缓存器号码

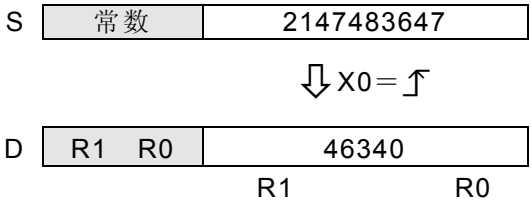
S，D 可结合 V、Z、P0~P9 作间接寻址应用

操作数 \ 范围	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR
	WX0 | WX240	WY0 | WY240	WM0 | WM1896	WS0 | WS984	T0 | T255	C0 | C255	R0 | R3839	R3840 | R3903	R3904 | R3967	R3968 | R4167	R5000 | R8071	D0 | D4095	16 或 32 位正数	V、Z P0~P9
S	○	○	○	○	○	○	○	○	○	○	○	○	○	○
D		○	○	○	○	○	○		○	○*	○*	○		○

- 当运算控制〝EN〞=1 或〝EN↑〞（P 指令）由 0→1 时，将 S 值或 S 所指定的缓存器内容值取平方根值（舍弃小数点以后的位数）后存入 D 所指定的缓存器内。
- 当 S 值为缓存器内容值，而值为负数，则 S 值错误旗号〝ERR〞设为 1，且本指令不执行。



- 左图程序范例是将常数值 2147483647 取其平方根值，再将结果存到 DR0（R1,R0）去。



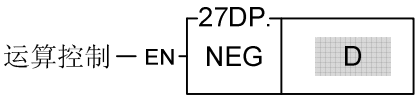
$$\sqrt{2147483647} = 46340.\underline{95}$$

↑

小数点以后舍弃

FUN27 D P NEG	取负数 （NEGATION）	FUN27 D P NEG

阶梯图符号



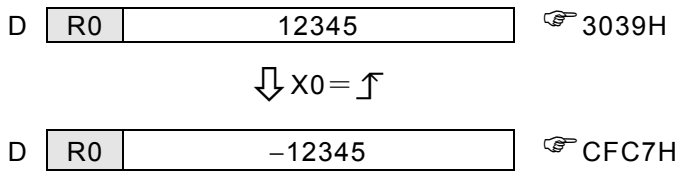
D：取负数的缓存器号码

D 可结合 V、Z、P0~P9 作间接寻址应用

操作数 \ 范围	WY	WM	WS	TMR	CTR	HR	OR	SR	ROR	DR	XR
	WY0 | WY240	WM0 | WM1896	WS0 | WS984	T0 | T255	C0 | C255	R0 | R3839	R3904 | R3967	R3968 | R4167	R5000 | R8071	D0 | D4095	V、Z P0~P9
D	○	○	○	○	○	○	○	○*	○*	○	○

- 当运算控制〝EN〞=1 或〝EN↑〞（P 指令）由 0→1 时，将 D 所指定的缓存器内容值取其负数（亦即取其 2 的补码）后存回原缓存器 D。
- 若 D 的内容值原为负数，取负数的结果将变为正数。

- 左图程序是将缓存器 R0 的值取负数后再存回 R0 去。



FUN28 D P ABS	取绝对值 （ABSOLUTE）	FUN28 D P ABS

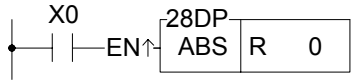
阶梯图符号

D：取绝对值的缓存器号码

D 可结合 V、Z、P0~P9 作间接寻址应用

操作数 \ 范围	WY	WM	WS	TMR	CTR	HR	OR	SR	ROR	DR	XR
	WY0 | WY240	WM0 | WM1896	WS0 | WS984	T0 | T255	C0 | C255	R0 | R3839	R3904 | R3967	R3968 | R4167	R5000 | R8071	D0 | D4095	V、Z P0~P9
D	○	○	○	○	○	○	○	○*	○*	○	○

● 当运算控制〝EN〞=1 或〝EN↑〞（P 指令）由 0→1 时，将 D 指定的缓存器内容值取绝对值后写回原缓存器 D。



- 左图程序例是将缓存器 DR0 的值取其绝对值后再存回 DR0（R1,R0）去。

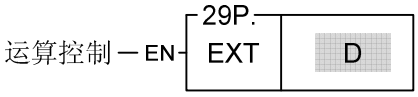
D R1 R0 −12345 ☞CFC7H

⇩X0＝⤒

D R1 R0 12345 ☞3039H

FUN29 D P EXT	缓存器正负符号扩展 （SIGN EXTENTION）	FUN29 D P EXT

阶梯图符号

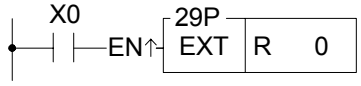


D：要扩展正负符号的缓存器号码

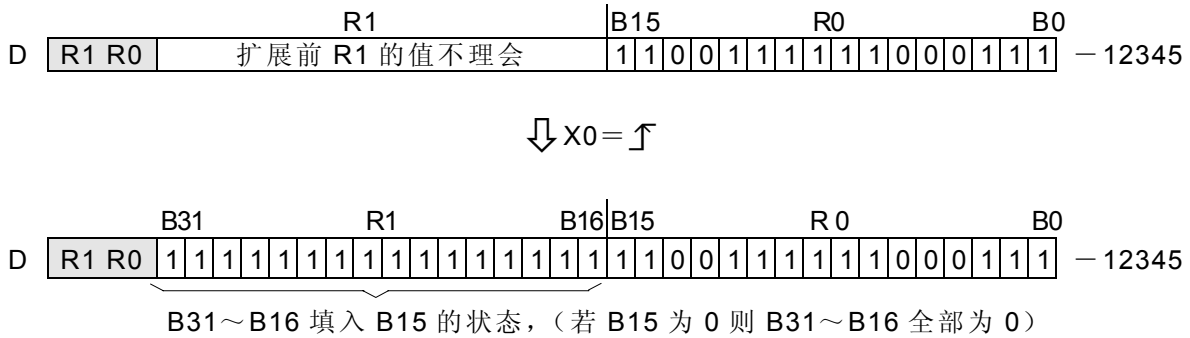
D 可结合 V、Z、P0~P9 作间接寻址应用

操作数 \ 范围	WY	WM	WS	TMR	CTR	HR	OR	SR	ROR	DR	XR
	WY0 ｜ WY240	WM0 ｜ WM1896	WS0 ｜ WS984	T0 ｜ T255	C0 ｜ C255	R0 ｜ R3839	R3904 ｜ R3967	R3968 ｜ R4167	R5000 ｜ R8071	D0 ｜ D4095	V、Z P0~P9
D	○	○	○	○	○	○	○	○*	○*	○	○

- 当运算控制〝EN〞=1 或〝EN↑〞（P 指令）由 0→1 时，将 D 所指定的数值，存入由 D+1 和 D 两个连续 Word 组成的 32 位缓存器。（两者值相同只是原来为 16 位所表示的数值，而扩展后变成由 32 位所表示的数值）。
- 本指令是将 16 位的缓存器数值扩展为等值的 32 位缓存器数值（例如将 33FFH 变成 000033FFH），其功用主要在于将 16 位数值和 32 位数值作各种运算（＋，－，＊，／，CMP……）时，用户数据的长度（表示位）一致，才能进行上述的各种运算。



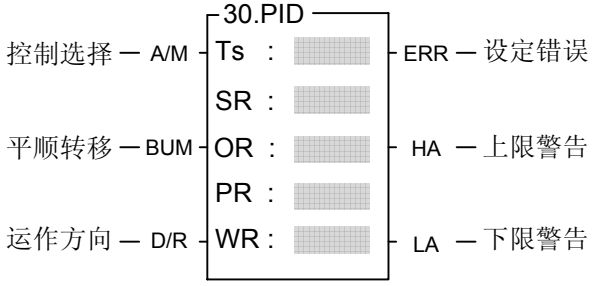
- 左图程序例是将 16 位的数值 R0 扩展为等值的 32 位数值后存到由 R0 本身和其左边（高位）相邻缓存器（R1）所构成的 32 位缓存器（DR0=R1R0）去。



扩展前（16 位） R0= CFC7H=－12345
扩展后（32 位）R1R0=FFFFCFC7H=－12345
} 两者实际数值相同

FUN30 PID	泛用 PID 运算指令 （功能简述）	FUN30 PID

阶梯图符号



操作数 \ 范围	HR R0 ｜ R3839	ROR R5000 ｜ R8071	DR D0 ｜ D4095	K
Ts	○	○	○	1～3000
SR	○	○*	○	
OR	○	○*	○	
PR	○	○*	○	
WR	○	○*	○	

Ts ：PID 运算间隔时间

SR ：程控设定值起始缓存器号码，共占用 8 个缓存器

OR ：PID 输出缓存器号码

PR ：参数设定值起始缓存器号码，共占用 7 个缓存器

WR：本指令所需使用的工作缓存器起始号码，共占用 5 个缓存器，其它地方不可重复使用。

- 泛用 PID 指令(FUN30)是将目前所测量的外界模拟量输入值当作程控变量（Process Variable，简称 PV），将用户所设定的设定值（Setpoint，简称 SP）与程控变量经由软件 PID 数学式运算后，得到适宜的输出控制值经由 D/A 模拟量输出模块或再处理经由其它界面来控制受控程序在用户所期望的设定范围内。
- 数字化 PID 表达式如下：

$$M_n = [(D4005/Pb)\times E_n] + \sum_{0}^{n} [(D4005/Pb)\times T_i \times T_s \times E_n] - [(D4005/Pb)\times T_d \times (PV_n - PV_{n-1})/T_s] + Bias$$

M_n = ： 〝n〞时的控制输出量

D4005 ：增益常数，默认值为 1000；可设定范围为 1～5000

Pb ： 比例带（范围：2~5000，单位为 0.1%；K_c（增益）=1000/ Pb）

T_i ： 积分时间常数（范围：0~9999，相当于 0.00~99.99 Repeats/Minute）

T_d ： 微分时间常数（范围：0~9999，相当于 0.00~99.99 Minutes）

PV_n ： 〝n〞时的程控变数值

PV_{n-1} ： 〝n〞的上一次的程控变数值

E_n ： 〝n〞时的误差=设定值（SP）–〝n〞时的程控变数值（PV_n）

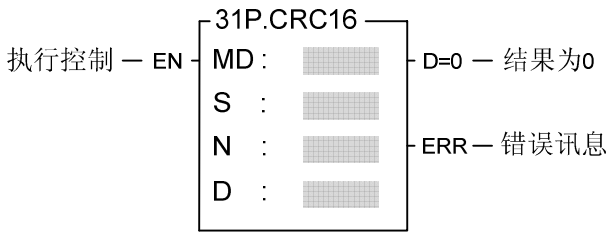
T_s ： PID 运算的间隔时间（范围：1~3000，单位：0.01S）

Bias ： 偏置输出量（范围：0~16380）

- 详细的功能说明与程序范例，请参考第 20 章〝FBs-PLC 的泛用 PID 控制〞的叙述。

FUN31 P CRC16	CRC16 计算指令 （CRC16）	FUN31 P CRC16

阶梯图符号



操作数 \ 范围	HR	ROR	DR	K
	R0 | R3839	R5000 | R8071	D0 | D4095	
MD				0~1
S	○	○	○	
N	○	○	○	1~256
D	○	○*	○	

MD：0，计算 CRC 时，只计算缓存器的低字节，缓存器的高字节不计算

：1，保留

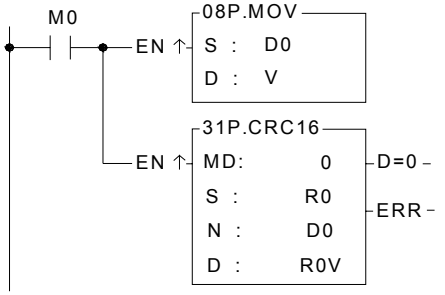
S：需计算 CRC 的起始缓存器号码

N：需计算 CRC 的数据长度，单位为 Byte

D：存放 CRC 计算结果的缓存器号码，
缓存器 D 存放 CRC 运算结果的 Upper Byte
缓存器 D＋1 存放 CRC 运算结果的 Lower Byte

S，N，D 操作数可结合 V、Z、P0~P9 指标作间接寻址应用。

- 当执行控制〝EN〞=1 或〝EN↑〞（P 指令）由 0→1 时，将以 S 为起始的 N 个数据缓存器的低字节作 CRC16 运算，并将运算结果存放到 D 与 D+1 缓存器中。
- 当 CRC16 运算结果为 0 时，输出指示〝D=0〞为 ON。
- 当运算数据长度不正确时，本指令不执行，输出指示〝ERR〞为 ON。
- PLC 与智能型外围通过通讯端口来作连结整合时，如果通讯间的数据类型为二进制而非 ASCII 码方式时，采用 CRC16 表达式来作整笔数据的校验计算是相当普遍的做法；在工业界使用相当普遍的 ModBus 通讯协议 RTU 模式即采用本表达式来作整笔数据的校验计算。
- 要核算 CRC16 运算结果的值是否正确，只要将用来计算 CRC16 的原始数据与其所产生 CRC16 的运算结果值再做一次 CRC16 运算，则新的 CRC16 的值必定为 0；当 PLC 与智能型外围通过通讯端口来作联机整合时，如果采用 CRC16 表达式来作整笔数据的校验计算，只要将所收到的整笔数据(其必含数据本身及 CRC16 侦误值)作 CRC16 运算，则 CRC16 的运算值必须为 0，才代表该笔数据无误。



范例说明：当 M0=1 时，以暂存器 R0 为起始，暂存 D0 的值为长度，执行 CRC16 运算，并将运算结果存放到暂存器 R0+V 和暂存器 R0+V+1 中。
本范例假设 D0=10，则存放 CRC16 运算结果的暂存器为 R10 和 R11。

S

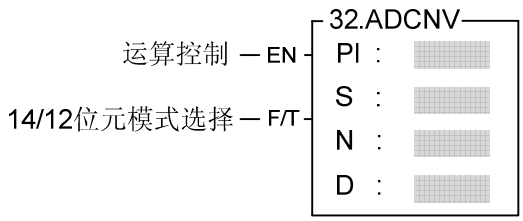
	High Byte	Low Byte
R0	Don't care	Byte-0
R1	Don't care	Byte-1
R2	Don't care	Byte-2
R3	Don't care	Byte-3
R4	Don't care	Byte-4
R5	Don't care	Byte-5
R6	Don't care	Byte-6
R7	Don't care	Byte-7
R8	Don't care	Byte-8
R9	Don't care	Byte-9

D

	High Byte	Low Byte
R10	00	CRC-Hi
R11	00	CRC-Lo

FUN32 ADCNV	4~20mA 模拟量输入读值转换指令 （ADCNV）	FUN32 ADCNV

阶梯图符号



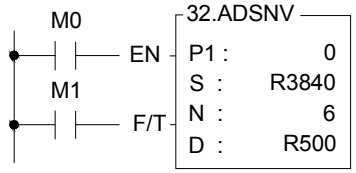
PI：0，模拟量输入模块设定在单极性信号。
：1，模拟量输入模块设定在双极性信号。
S：要转换的来源缓存器号码。
N：要转换的长度，单位为 Word。
D：存放转换结果的起始缓存器号码。
S，N，D 操作数可结合 V、Z、P0~P9 指标作间接寻址应用。

操作数 \ 范围	HR	IR	ROR	DR	K
	R0 ｜ R3839	R3840 ｜ R3903	R5000 ｜ R8071	D0 ｜ D4095	
PI					0~1
S	○		○	○	
N	○	○	○	○	1~64
D	○		○*	○	

- 当外界的模拟量信号为 2～10mA/4～20mA/1～5V/2～10V 时,可选择 FBs 模拟量输入模块来读取外界信号；但 FBs 模拟量模块输入范围为 0～10mA/0～5V(选择 5V、单极性工作模式)或 0～20mA/0～10V (选择 10V、单极性工作模式)，很明显的原始模拟量输入读值将会有一偏差值存在,本指令可用来将有偏差值的模拟量输入读值转换为 0～4095 (12 位格式)或 0～16383 (14 位格式)以利以后程序对此类模拟量信号作处理。
- 当执行控制〝EN〞=1 时，将以 S 为起始的 N 个数据缓存器的 2～10mA/4～20mA/1～5V/2～10V 模拟量输入原始读值转换为 0～4095 (12 位格式)或 0～16383 (14 位格式)，并将运算结果存放到 D 缓存器群中。
- 当〝F/T〞输入=0 时，模拟量输入原始读值为 12 位格式；〝F/T〞输入=1 时，模拟量输入原始读值为 14 位格式。
- 当运算数据长度不正确时，本指令不执行。
- 要使用本指令必须配合 FBs 模拟量输入模块设定为双极性读值模式,也就是模拟量输入原始读值为－8192～8191 模式才可得到正常转换值；如果模拟量输入模块设定为单极性读值模式，也就是模拟量输入原始读值为 0～16383 模式，则无法产生正确的转换值。

FUN32 ADCNV	4~20mA 模拟量输入读值转换指令 （ADCNV）	FUN32 ADCNV

程序范例:



范例说明：当 M0=1, M1=0 时，以缓存器 R3840 为起始，将 6 点有偏差值的模拟量输入读值转换为 0～4095，并将转换结果存放到缓存器 R500～R505。

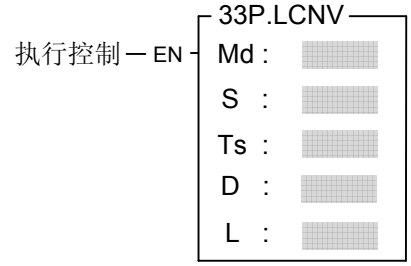
	S		D	
R3840	－1229	R500	0	(4 mA)
R3841	409	R501	2047	(12 mA)
R3842	2047	R502	4095	(20 mA)
R3843	－2048	R503	0	(0 mA)
R3844	－2048	R504	0	(0 mA)
R3845	－2048	R505	0	(0 mA)

而若 M0=1, M1=1 时，以缓存器 R3840 为起始，将 6 点有偏差值的模拟量输入读值转换为 0～16383，并将转换结果存放到缓存器 R500～R505。

	S		D	
R3840	－4916	R500	0	(4 mA)
R3841	1637	R501	8191	(12 mA)
R3842	8191	R502	16383	(20 mA)
R3843	－8192	R503	0	(0 mA)
R3844	－8192	R504	0	(0 mA)
R3845	－8192	R505	0	(0 mA)

FUN33 P LCNV	线性转换指令 （LCNV）	FUN33 P LCNV

阶梯图符号



Md：运算模式选择，0～3
S ：要转换的来源缓存器起始号码
Ts ：转换表格起始缓存器起始号码
D ：存放转换结果的起始缓存器号码
L ：要转换的长度，1～64

操作数＼范围	HR	IR	ROR	DR	K
	R0 ｜ R3839	R3840 ｜ R3903	R5000 ｜ R8071	D0 ｜ D3999	
Md					0～3
S	○	○	○	○	
Ts	○		○	○	
D	○		○*	○	
L	○		○	○	1～64

- 当使用模拟量输入模块读取外界模拟量信号时，可以利用本指令将原始模拟量读值转换为相对应的工程读值来作为实际工程值的显示或作为控制的比较、运算等应用。
- 当使用温度或模拟量模块来作温度或模拟量测量应用时，如果 PLC 所测量的温度或工程读值与标准温度计或相关标准仪表所测量的结果有偏差时，可以利用本指令来作线性修正以作为实际测量值的校正。
- 当执行控制〝EN〞=1 或由 0→1(P 指令)时，将以 S 为起始的 L 个数据缓存器根据运算模式选择，以转换表格内容所设定的参数值执行线性转换运算，并将运算结果存放到以 D 为起始的缓存器群中。
- 本指令共提供两种线性转换公式如下所示以适合各种不同的应用:

(公式一) 两点校正法:

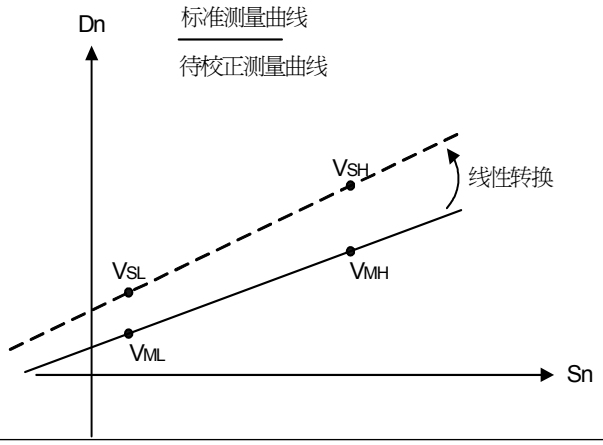
在转换表格内填入低点测量值(V_{ML})、高点测量值(V_{MH})及对应的低点标准值(V_{SL})与高点标准值(V_{SH})；执行线性转换时，来源数据(Sn)经由下列运算产生对应的目标值(Dn)。

$A = (V_{SL} - V_{SH} / V_{ML} - V_{MH}) \times 10000$

$B = V_{SL} - (V_{ML} \times A / 10000)$

$Dn = (Sn \times A / 10000) + B$

- 本表达式所有操作数(V_{SL}、V_{SH}、V_{ML}、V_{MH}、Sn、Dn)之值范围为 -32768～32767。
- 本表达式也可用来将原始量模拟输入读值量化为实际工程值；在此应用时，V_{ML}=模拟量输入最小值、V_{MH}=模拟量输入最大值，V_{SL}=工程最小值，V_{SH}=工程最大值。



FUN33 P LCNV	线性转换指令 （LCNV）	FUN33 P LCNV

(公式二) 倍率＋偏置量:

在转换表格内填入倍率分子值(A)、倍率分母值(B)及偏置值(C)；执行线性转换时，来源数据(Sn)经由下列运算产生对应的目标值(Dn)。

Dn =[(Sn×A) / B]＋C

本表达式各操作数的值范围如下:

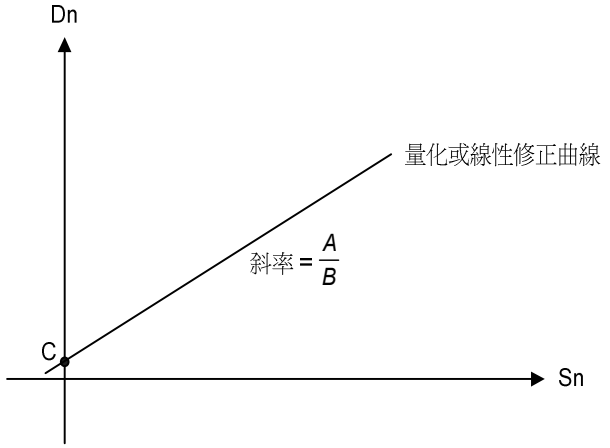
A = 1～65535

B = 1～65535

C = -32768～32767

Sn = 0～65535

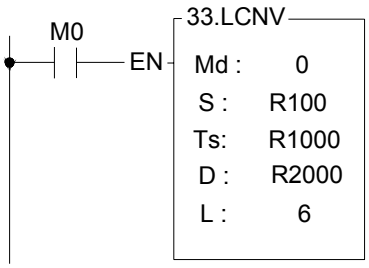
Dn = -32768～32767



运算模式说明:

1. 运算模式为 0 时，使用公式一表达式；所有来源数据共享转换表格内同一组 VML、VMH、VSL 与 VSH 参数值作线性转换运算。
2. 运算模式为 1 时，使用公式一表达式；每个来源数据独立使用转换表格内相对应的一组 VML、VMH、VSL 与 VSH 参数值作线性转换运算。如果有 N 个来源数据需转换，则转换表格内需有 N 组 VML、VMH、VSL 与 VSH 参数值，共占用 N×4 个缓存器。
3. 运算模式为 2 时，使用公式二表达式；所有来源数据共享转换表格内同一组 A、B 与 C 参数值作线性转换运算。
4. 运算模式为 3 时，使用公式二表达式；每个来源数据独立使用转换表格内相对应的一组 A、B 与 C 参数值作线性转换运算。如果有 N 个来源数据需转换，则转换表格内需有 N 组 A、B 与 C 参数值，共占用 N×3 个缓存器。

程序范例 1：运算模式为 0 的线性转换运算



范例说明：当 M0=1 时，以缓存器 R100 为起始来源数据，根据 R1000 为起始的转换表格内 VML、VMH、VSL、VSH 参数值作 6 点线性转换运算，并将转换结果存放至缓存器 R2000～R2005。

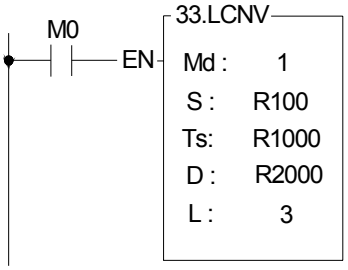
FUN33 P LCNV	线性转换指令（LCNV）	FUN33 P LCNV

Ts

R1000	282	VML
R1001	3530	VMH
R1002	260	VSL
R1003	3650	VSH

S			D	
R100	282		R2000	260
R101	3530		R2001	3650
R102	1906	⇨	R2002	1955
R103	0		R2003	-34
R104	5000		R2004	5184
R105	-115		R2005	-154

程序范例 2：运算模式为 1 的线性转换运算



范例说明：当 M0=1 时，以缓存器 R100 为起始来源数据，根据 R1000 为起始的转换表格内各组 VML、VMH、VSL、VSH 参数值作 3 点线性转换运算，并将转换结果存放至缓存器 R2000～R2002。

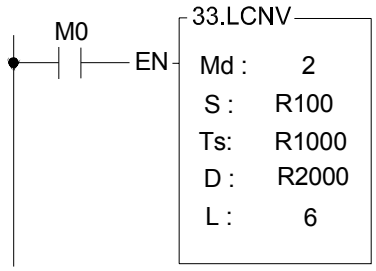
Ts

R1000	282	VML_0
R1001	3530	VMH_0
R1002	260	VSL_0
R1003	3650	VSH_0
R1004	-52	VML_1
R1005	1208	VMH_1
R1006	-38	VSL_1
R1007	1101	VSH_1
R1008	235	VML_2
R1009	4563	VMH_2
R1010	264	VSL_2
R1011	4588	VSH_2

S			D	
R100	282		R2000	260
R101	1208	⇨	R2001	1100
R102	2399		R2002	2426

FUN33 P LCNV	线性转换指令（LCNV）	FUN33 P LCNV

程序范例 3：运算模式为 2 的线性转换运算



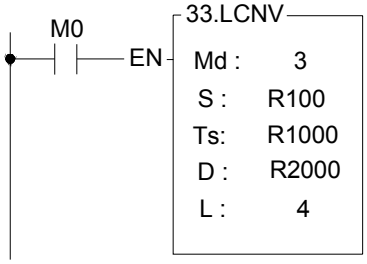
范例说明：当 M0=1 时，以缓存器 R100 为起始来源数据，根据 R1000 为起始的转换表格内 A、B、C 参数值作 6 点线性转换运算，并将转换结果存放到缓存器 R2000～R2005。

Ts

R1000	985	A
R1001	1000	B
R1002	22	C

S			D	
R100	1000	⇨	R2000	1005
R101	2345		R2001	2329
R102	3560		R2002	3526
R103	401		R2003	414
R104	568		R2004	579
R105	2680		R2005	2659

程序范例 4：运算模式为 3 的线性转换运算

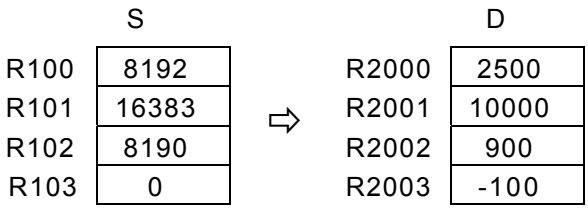


范例说明：当 M0=1 时，以缓存器 R100 为起始来源数据，根据 R1000 为起始的转换表格内各组 A、B、C 参数值作 4 点线性转换运算，并将转换结果存放到缓存器 R2000～R2003。

FUN33 P LCNV	线性转换指令 （LCNV）	FUN33 P LCNV

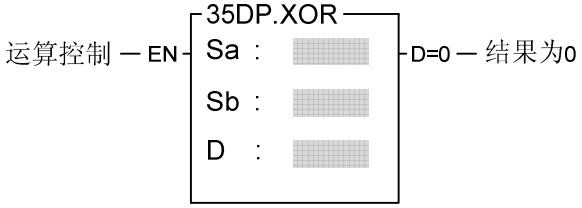
Ts

R1000	5000	A_0
R1001	16380	B_0
R1002	0	C_0
R1003	10000	A_1
R1004	16383	B_1
R1005	0	C_1
R1006	2200	A_2
R1007	16380	B_2
R1008	-200	C_2
R1009	1600	A_3
R1010	16383	B_3
R1011	-100	C_3



FUN35 D P XOR	逻辑互斥或（XOR）运算 （EXCLUSIVE OR）	FUN35 D P XOR

阶梯图符号



Sa：XOR 数据 a 或其缓存器号码

Sb：XOR 数据 b 或其缓存器号码

D ：存放 XOR 结果的缓存器号码

Sa，Sb，D 可结合 V、Z、P0~P9 作间接寻址应用

操作数 \ 范围	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR
	WX0 | WX240	WY0 | WY240	WM0 | WM1896	WS0 | WS984	T0 | T255	C0 | C255	R0 | R3839	R3840 | R3903	R3904 | R3967	R3968 | R4167	R5000 | R8071	D0 | D4095	16 或 32 位 正、负数	V、Z P0~P9
Sa	○	○	○	○	○	○	○	○	○	○	○	○	○	○
Sb	○	○	○	○	○	○	○	○	○	○	○	○	○	○
D		○	○	○	○	○	○		○	○*	○*	○		○

- 当运算控制〝EN〞=1 或〝EN↑〞（P 指令）由 0→1 时，将 Sa 和 Sb 数据作逻辑 XOR（Exclusive OR）运算，也就是将 Sa 和 Sb 的各对应位（B0～B15 或 B0～B31）作比较，任一个对应位的状态如果不相同，则在 D 的该对应位设为 1，相同则为 0。
- 若运算结果 D 的所有位都为 0，则结果为 0 旗号〝D=0〞设为 1。

• 左图程序例是将缓存器 R0 和 R1 作逻辑互斥或运算后将结果存到 R2 去

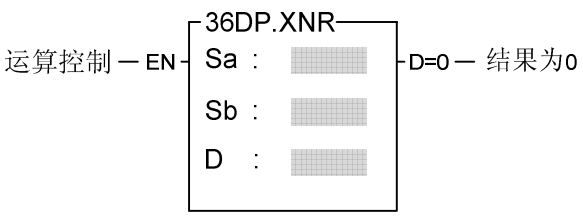
Sa	R0	1	0	1	1	1	0	1	1	0	1	1	0	1	1	0	1
Sb	R1	1	1	1	0	1	1	1	0	1	0	1	0	0	1	1	0

⇩X0＝⇂

D	R2	0	1	0	1	0	1	0	1	1	1	0	0	1	0	1	1

FUN36 D P XNR	逻辑互斥容（XNR）运算 （ENCLUSIVE OR）	FUN36 D P XNR

阶梯图符号



Sa ： XNR 数据 a 或其缓存器号码

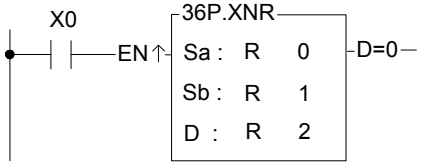
Sb ： XNR 数据 b 或其缓存器号码

D ： 存放 XNR 结果的缓存器号码

Sa，Sb，D 可结合 V、Z、P0~P9 作间接寻址应用

操作数 \ 范围	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR
	WX0 | WX240	WY0 | WY240	WM0 | WM1896	WS0 | WS984	T0 | T255	C0 | C255	R0 | R3839	R3840 | R3903	R3904 | R3967	R3968 | R4167	R5000 | R8071	D0 | D4095	16 或 32 位 正、负数	V、Z P0~P9
Sa	○	○	○	○	○	○	○	○	○	○	○	○	○	○
Sb	○	○	○	○	○	○	○	○	○	○	○	○	○	○
D		○	○	○	○	○	○		○	○*	○*	○		○

- 当运算控制〝EN〞=1 或〝EN↑〞（P 指令）由 0→1 时，将 Sa 和 Sb 数据作逻辑 XNR（Enclusive OR，即 Exclusive OR 的相反），也就是将 Sa 和 Sb 的各对应位（B0～B15 或 B0～B31）作比较，任一对应位的状态相同，则 D 的该对应位设为 1，如果不同则设为 0。
- 若运算结果 D 的所有位都为 0，则结果为 0 旗号〝D=0〞设为 1。



- 左图程序范例是将缓存器 R0 和 R1 作逻辑互斥容运算后，再将所得结果存到缓存器 R2 去

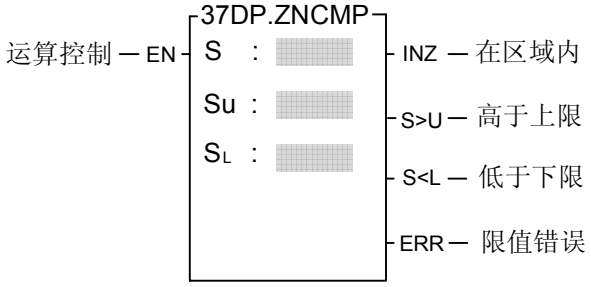
Sa	R0	1	0	1	1	1	0	1	1	0	1	1	0	1	1	0	1
Sb	R1	1	1	1	0	1	1	1	0	1	0	1	0	0	1	1	0

⇩X0＝⇂

D	R2	1	0	1	0	1	0	1	0	0	0	1	1	0	1	0	0

FUN37 D P ZNCMP	区域比较 （ZONE COMPARE）	FUN37 D P ZNCMP

阶梯图符号



S ：存放比较数据的缓存器号码

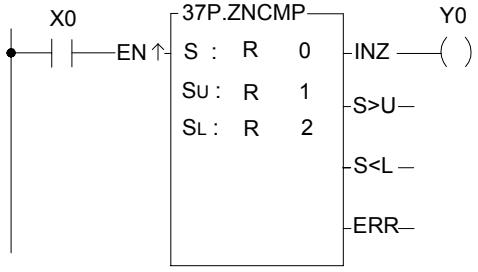
Su：区域上限值或上限值缓存器号码

SL：区域下限值或下限值缓存器号码

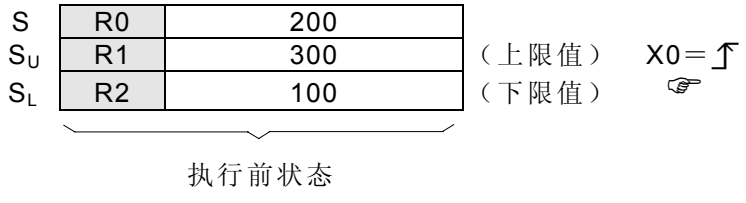
S，Su，SL 可结合 V、Z、P0~P9 作间接寻址应用

操作数 \ 范围	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR
	WX0 | WX240	WY0 | WY240	WM0 | WM1896	WS0 | WS984	T0 | T255	C0 | C255	R0 | R3839	R3840 | R3903	R3904 | R3967	R3968 | R4167	R5000 | R8071	D0 | D4095	16 或 32 位正、负数	V、Z P0~P9
S	○	○	○	○	○	○	○	○	○	○	○	○		○
Su	○	○	○	○	○	○	○	○	○	○	○	○	○	○
SL	○	○	○	○	○	○	○	○	○	○	○	○	○	○

- 当比较控制〝EN〞=1 或〝EN↑〞（P 指令）由 0→1 时，执行 S 与上限 Su 及下限 SL 的比较，如果 S 介于上限值与下限值之间（SL≦S≦Su），则在区域内旗号〝INZ〞设为 1，如果 S 的值大于上限 Su，则高于上限旗号〝S＞U〞设为 1，如果 S 的值小于下限 SL，则低于下限旗号〝S＜L〞设为 1。
- 上限 Su 应大于下限 SL，若 Su＜SL，则限值错误旗号〝ERR〞设为 1，且本指令不执行。

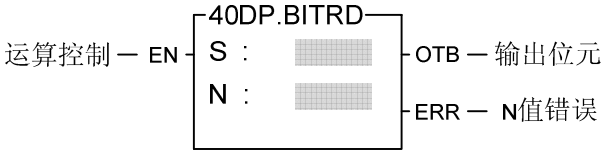


- 左图程序范例是将 R0 的值和由 R1 和 R2 所构成的上、下限区域作比较，设 R0～R2 的数值如下图左，则可获得如下图右的执行结果。
- 如果输出结果需要为不在区域内，则可用 OUT NOT Y0 即可。



FUN40 D P BITRD	位数据读取 （BIT READ）	FUN40 D P BITRD

阶梯图符号



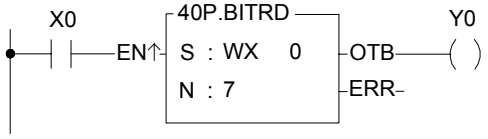
S：要读取位的数据或其缓存器号码

N：指定 S 数据中第 N 个位数据被读出

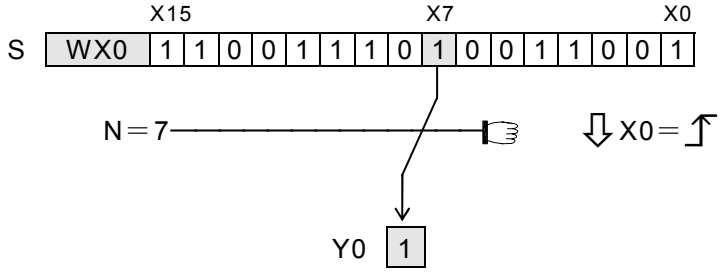
S，N 可结合 V、Z、P0~P9 作间接寻址应用

操作数 \ 范围	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR
	WX0 | WX240	WY0 | WY240	WM0 | WM1896	WS0 | WS984	T0 | T255	C0 | C255	R0 | R3839	R3840 | R3903	R3904 | R3967	R3968 | R4167	R5000 | R8071	D0 | D4095	16 或 32 位正、负数	V、Z P0~P9
S	○	○	○	○	○	○	○	○	○	○	○	○	○	○
N	○	○	○	○	○	○	○	○	○	○	○	○	0～31	○

- 当读取控制〝EN〞=1 或〝EN↑〞（P 指令）由 0→1 时，将 S 数据中的第 N 个位（BIT）取出送到输出位〝OTB〞去。
- 当读取控制〝EN〞或〝EN↑〞（P 指令）=0 时，输出位〝OTB〞的状态保持上次执行的结果（M1919=0）或清除为 0（M1919=1）。
- N 的值在 16 位指令时有效范围为 0～15，在 32 位（D 指令）时则为 0～31，超出此范围则 N 值错误旗号〝ERR〞设为 1，且本指令不执行。

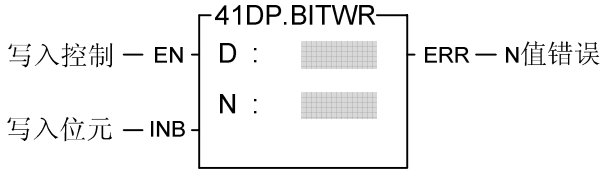


- 左图程序范例是自 WX0（X0～X15）中读取第 7 个位（X7）的状态，再将它送到 Y0 去，其结果如下：



FUN41 D P BITWR	位数据写入 （BIT WRITE）	FUN41 D P BITWR

阶梯图符号



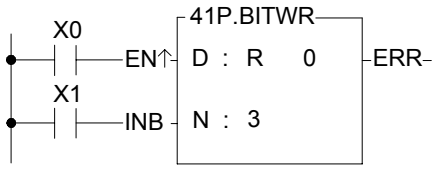
D：要写入位的缓存器号码

N：指定将写入位 INB 的状态写入 D 中第 N 个位处

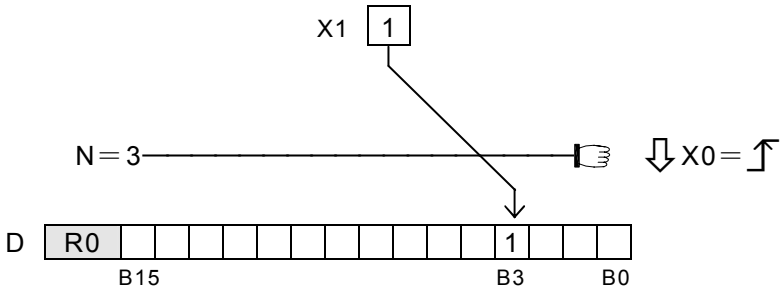
D，N 可结合 V、Z、P0~P9 作间接寻址应用

操作数 \ 范围	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR
	WX0 | WX240	WY0 | WY240	WM0 | WM1896	WS0 | WS984	T0 | T255	C0 | C255	R0 | R3839	R3840 | R3903	R3904 | R3967	R3968 | R4167	R5000 | R8071	D0 | D4095	0 0 | 或 | 15 31	V，Z P0~P9
D		○	○	○	○	○	○		○	○*	○*	○		○
N	○	○	○	○	○	○	○	○	○	○	○	○	○	○

- 当写入控制〝EN〞=1 或〝EN↑〞（P 指令）由 0→1 时，将写入位（INB）的状态写入 D 中 N 所指定的位去。
- N 的值在 16 位指令时有效范围为 0～15，在 32 位（D 指令）时则为 0～31，如果超出该范围则 N 值错误旗号〝ERR〞设为 1，且本指令不执行。



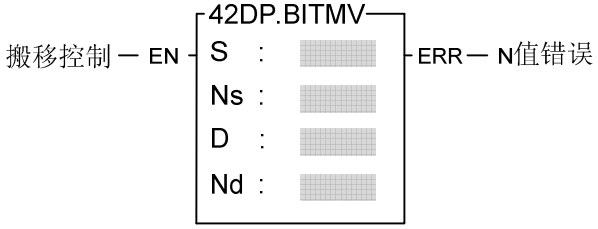
• 左图程序范例是将写入位 INB 的状态写到 R0 中的 B3 去，假设 X=1，其执行结果如下图。



B3 以外的位保持不变

FUN42 D P BITMV	位数据搬移 （BIT MOVE）	FUN42 D P BITMV

阶梯图符号



S ：搬移的来源数据或其缓存器号码

Ns：指定 S 中的 Ns 位为来源位

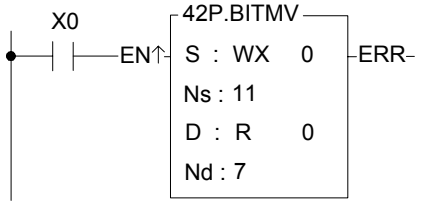
D ：搬移的目的缓存器号码

Nd：指定 D 中的 Nd 位为目的位

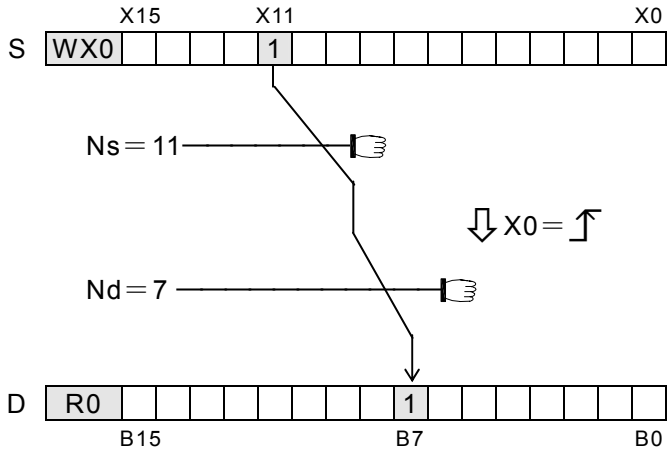
S，Ns，D，Nd 可结合 V、Z、P0~P9 作间接寻址应用

操作数 \ 范围	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR
	WX0 | WX240	WY0 | WY240	WM0 | WM1896	WS0 | WS984	T0 | T255	C0 | C255	R0 | R3839	R3840 | R3903	R3904 | R3967	R3968 | R4167	R5000 | R8071	D0 | D4095	16 或 32 位 正、负数	V、Z P0~P9
S	○	○	○	○	○	○	○	○	○	○	○	○	○	○
Ns	○	○	○	○	○	○	○	○	○	○	○	○	0～31	○
D		○	○	○	○	○	○		○	○*	○*	○		○
Nd	○	○	○	○	○	○	○	○	○	○	○	○	0～31	○

- 当搬移控制〝EN〞=1 或〝EN↑〞（P 指令）由 0→1 时，将 S 中 Ns 所指定的位状态搬移到 D 中 Nd 所指定的位去。
- Ns 或 Nd 的值在 16 位指令时有效范围为 0～15，在 32 位（D 指令）时则为 0～31，超出此范围，则 N 值错误旗号〝ERR〞设为 1，且本指令不执行。

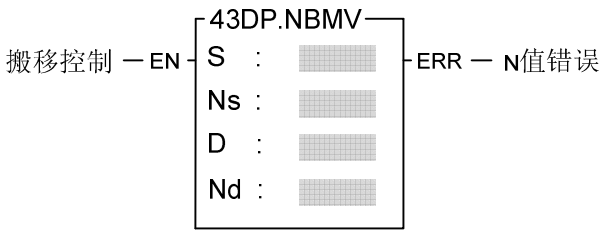


- 左图程序范例是将 S 中的 B11（即 X11）的状态搬移到 D 中 B7 的位置去，D 中除被写入的位 B7 外，其它位状态不变。



FUN43 D P NBMV	位数（NIBBLE）搬移 （NIBBLE MOVE）	FUN43 D P NBMV

阶梯图符号



S ：搬移的来源数据或其缓存器号码

Ns：指定 S 中的第 Ns 个位数为来源位数

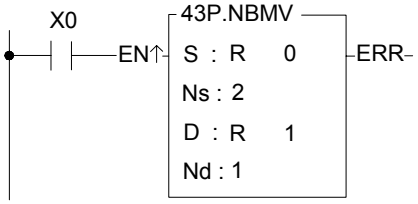
D ：搬移的目的缓存器号码

Nd：指定 D 中的第 Nd 个位数为目的位数

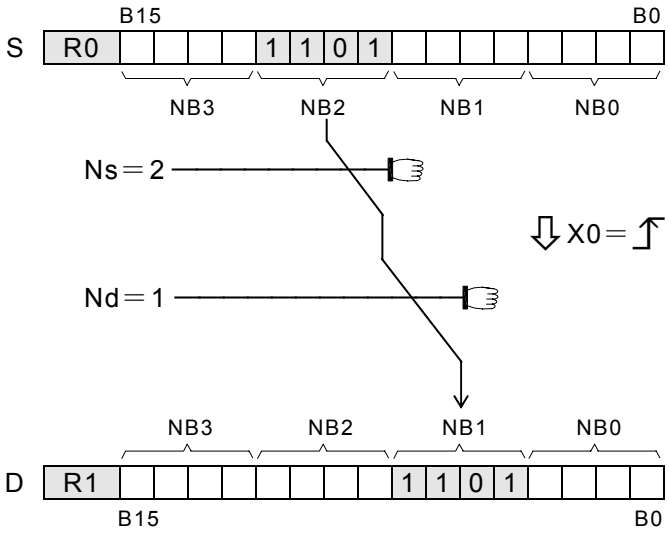
S，Ns，D，Nd 可结合 V、Z、P0~P9 作间接寻址应用

操作数 \ 范围	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR
	WX0 | WX240	WY0 | WY240	WM0 | WM1896	WS0 | WS984	T0 | T255	C0 | C255	R0 | R3839	R3840 | R3903	R3904 | R3967	R3968 | R4167	R5000 | R8071	D0 | D4095	16 或 32 位正、负数	V、Z P0~P9
S	○	○	○	○	○	○	○	○	○	○	○	○	○	○
Ns	○	○	○	○	○	○	○	○	○	○	○	○	0～7	○
D		○	○	○	○	○	○		○	○*	○*	○		○
Nd	○	○	○	○	○	○	○	○	○	○	○	○	0～7	○

- 当搬移控制〝EN〞=1 或〝EN↑〞（P 指令）由 0→1 时，将 S 中第 Ns 个位数（Nibble：为 4 个位所组合。由缓存器的最低位 B0 起每连续 4 个位形成一个 Nibble，即 B0～B3 为第 0 个位数，B4～B7 为第 1 个位数，....）搬移到 D 中 Nd 所指定的那个位数去。
- Ns 或 Nd 在 16 位指令时，有效范围为 0～3，在 32 位（D 指令）时则为 0～7，超出此范围则 N 值错误旗号〝ERR〞设为 1，且本指令不执行。

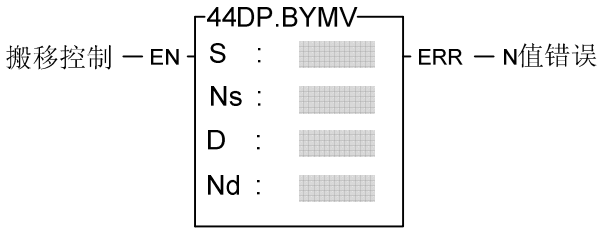


- 左图程序范例是将 S 中的第 2 个位数 NB2（即 B8～B11）搬到 D 中的第 1 个位数 NB1（即 B4～B7）去，D 中的其它位数则保持不变。



FUN44 D P BYMV	字节（BYTE）搬移 （BYTE MOVE）	FUN44 D P BYMV

阶梯图符号



S ：搬移的来源数据或其缓存器号码

Ns：指定 S 中的第 Ns 个字节为来源字节

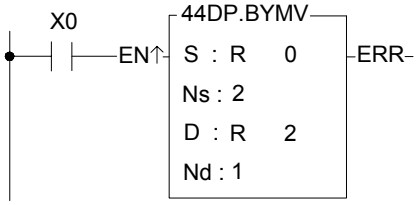
D ：搬移的目的缓存器号码

Nd：指定 D 中的第 Nd 个字节为目的字节

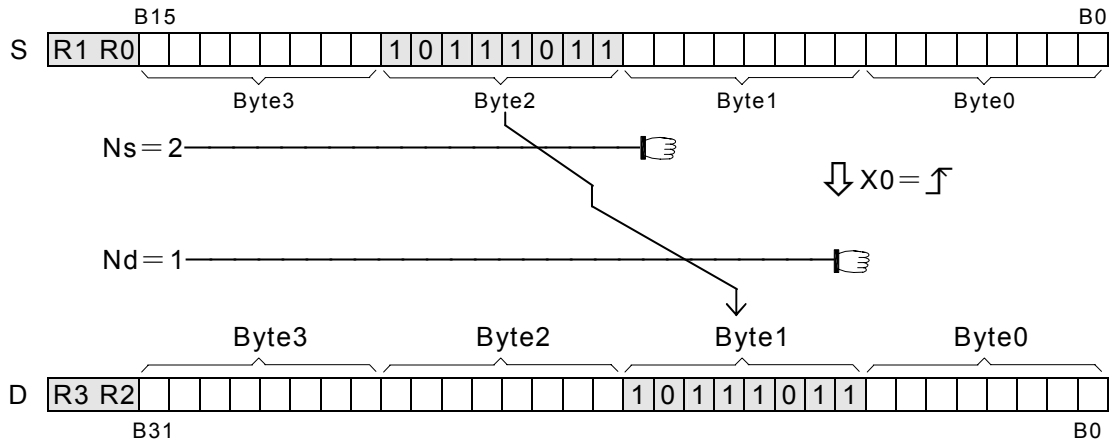
S，Ns，D，Nd 可结合 V、Z、P0~P9 作间接寻址应用

操作数 \ 范围	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR
	WX0 | WX240	WY0 | WY240	WM0 | WM1896	WS0 | WS984	T0 | T255	C0 | C255	R0 | R3839	R3840 | R3903	R3904 | R3967	R3968 | R4167	R5000 | R8071	D0 | D4095	16 或 32 位正、负数	V、Z P0~P9
S	○	○	○	○	○	○	○	○	○	○	○	○	○	○
Ns	○	○	○	○	○	○	○	○	○	○	○	○	0～3	○
D		○	○	○	○	○	○		○	○*	○*	○		○
Nd	○	○	○	○	○	○	○	○	○	○	○	○	0～3	○

- 当搬移控制〝EN〞=1 或〝EN↑〞（P 指令）由 0→1 时，将 S 中第 Ns 个字节（Byte：为 8 个字节成，由缓存器的最低位 B0 起每连续 8 个位形成一个 Byte，即 B0～B7 为第 0 个字节，B8～B15 为第 1 个字节……）搬移到 D 中第 Nd 个字节处。
- Ns 或 Nd 在 16 位指令时有效范围为 0～1，在 32 位（D 指令）时则为 0～3，超出该范围则 N 值错误旗号〝ERR〞设为 1，且本指令不执行。

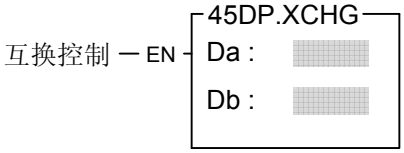


- 左图程序范例是将 S（由 R1R0 所构成的 32 位缓存器）中的第 2 个字节（即 B16～B23）搬移到 D（由 R3R2 构成的 32 位缓存器）中的第 1 个字节去，D 中的其它字节则保持不变。



FUN45 D P XCHG	资料互换 （EXCHANGE）	FUN45 D P XCHG

阶梯图符号



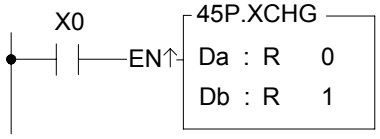
Da：互换的缓存器 A 的号码

Db：互换的缓存器 B 的号码

Da，Db 可结合 V、Z、P0~P9 作间接寻址应用

操作数 \ 范围	WY	WM	WS	TMR	CTR	HR	OR	SR	ROR	DR	XR
	WY0 | WY240	WM0 | WM1896	WS0 | WS984	T0 | T255	C0 | C255	R0 | R3839	R3904 | R3967	R3968 | R4167	R5000 | R8071	D0 | D4095	V、Z P0~P9
Da	○	○	○	○	○	○	○	○*	○*	○	○
Db	○	○	○	○	○	○	○	○*	○*	○	○

● 当互换控制〝EN〞=1 或〝EN↑〞（P 指令）由 0→1 时，将 16 位或 32 位（D 指令）的缓存器 Da 和缓存器 Db 的内容互换。



- 左图程序范例是将 16 位缓存器 R0 和 R1 的数据内容互换。

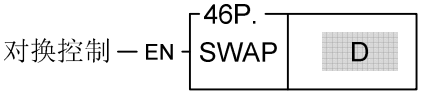
		B15															B0
Da	R0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Db	R1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

⇩X0＝⤒

		B15															B0
Da	R0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Db	R1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FUN46 P SWAP	字节（BYTE）数据对换 （BYTE SWAP）	FUN46 P SWAP

阶梯图符号

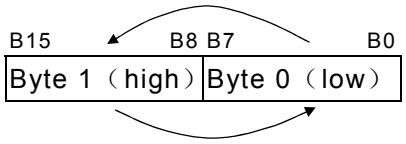


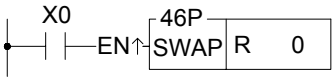
D：执行字节数据对换的缓存器号码

D 可结合 V、Z、P0~P9 作间接寻址应用

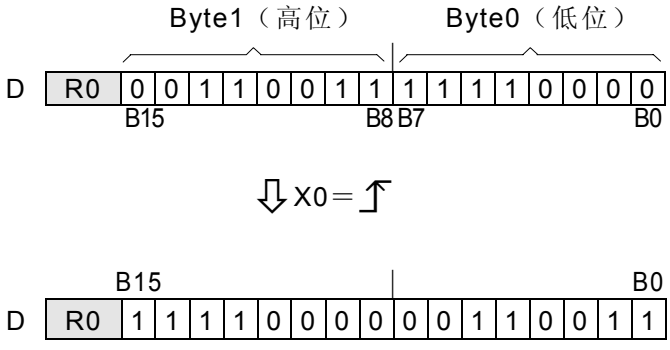
操作数 \ 范围	WY	WM	WS	TMR	CTR	HR	OR	SR	ROR	DR	XR
	WY0 ｜ WY240	WM0 ｜ WM1896	WS0 ｜ WS984	T0 ｜ T255	C0 ｜ C255	R0 ｜ R3839	R3904 ｜ R3967	R3968 ｜ R4167	R5000 ｜ R8071	D0 ｜ D4095	V、Z P0~P9
D	○	○	○	○	○	○	○	○*	○*	○	○

● 当对换控制〝EN〞=1 或〝EN↑〞（P 指令）由 0→1 时，将 D 所指定的 16 位缓存器的低字节 Byte 0（B0～B7）和高字节 Byte 1（B8～B15）的数据对换。



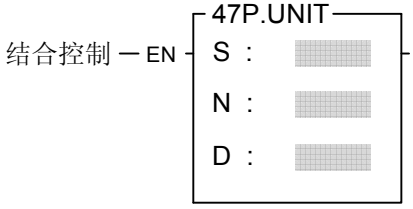


- 左图程序是将 R0 的低字节（B0～B7）和高字节（B8～B15）的资料互换，其结果如下。



FUN47 P UNIT	位数（NIBBLE）数据结合 （NIBBLE UNITE）	FUN47 P UNIT

阶梯图符号



S：要被结合的来源缓存器起头号码

N：要结合的位数

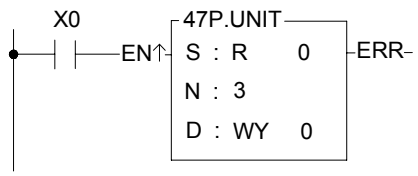
D：存放结合数据的缓存器号码

S，N，D 可结合 V、Z、P0~P9 作间接寻址应用

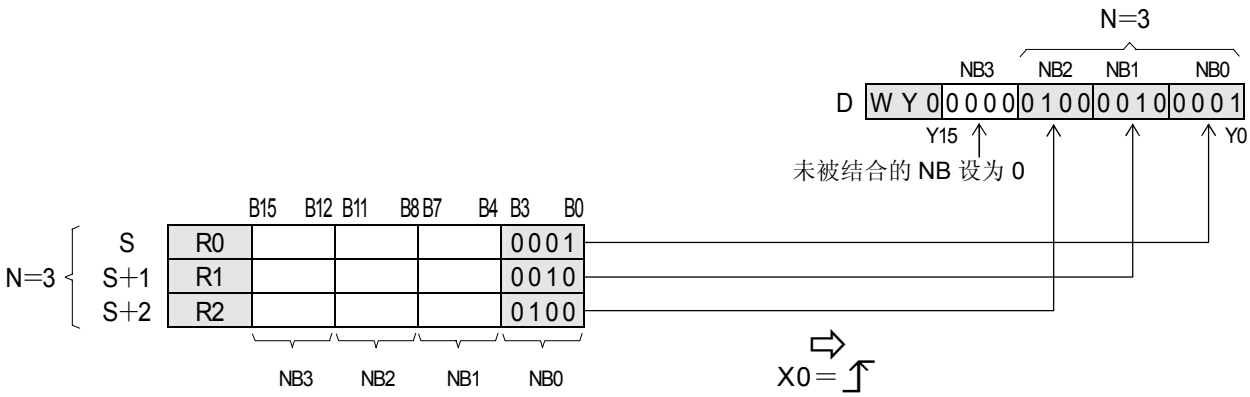
操作数 \ 范围	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR
	WX0 ｜ WX240	WY0 ｜ WY240	WM0 ｜ WM1896	WS0 ｜ WS984	T0 ｜ T255	C0 ｜ C255	R0 ｜ R3839	R3840 ｜ R3903	R3904 ｜ R3967	R3968 ｜ R4167	R5000 ｜ R8071	D0 ｜ D4095	1 ｜ 4	V、Z P0~P9
S	○	○	○	○	○	○	○	○	○	○	○	○		○
N	○	○	○	○	○	○	○	○	○	○	○	○	○	○
D		○	○	○	○	○	○		○	○*	○*	○		○

- 当结合控制〝EN〞=1 或〝EN↑〞（P 指令）由 0→1 时，取出由 S 开始的连续 N 个缓存器的最低位数 NB0（Nibble：为 4 位所组成，由缓存器的最低位 B0 起往左每连续 4 个位构成一位数，即 B0～B3 为第 0 个位数 NB0，B4～B7 为第 1 个位数 NB1，.....）并将他 3 由低位往高位的顺序依序填到 D 中的 NB0，NB1，..... NBn-1，D 中未被填入的位数则填入 0。

- 本指令只提供 WORD（16 位）指令，因此最多只有 4 个 Nibble，故 N 的有效范围为 1～4，超出此范围则 N 值错误旗号〝ERR〞设为 1，且本指令不执行。

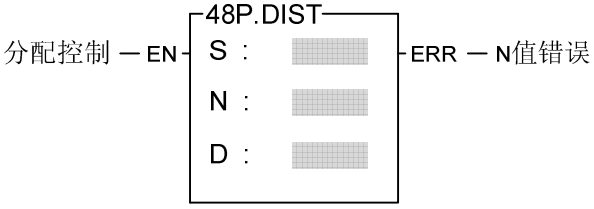


- 左图程序范例是将 R0，R1 和 R2 三个缓存器的 NB0 取出填到 WY0 缓存器中的 NB0～NB2 去。



FUN48 P DIST	位数（NIBBLE）数据分配 （NIBBLE DISTRIBUTE）	FUN48 P DIST

阶梯图符号



S：分配的来源数据或缓存器号码

N：要分配的位数的数目

D：存放分配数据的缓存器起头号码

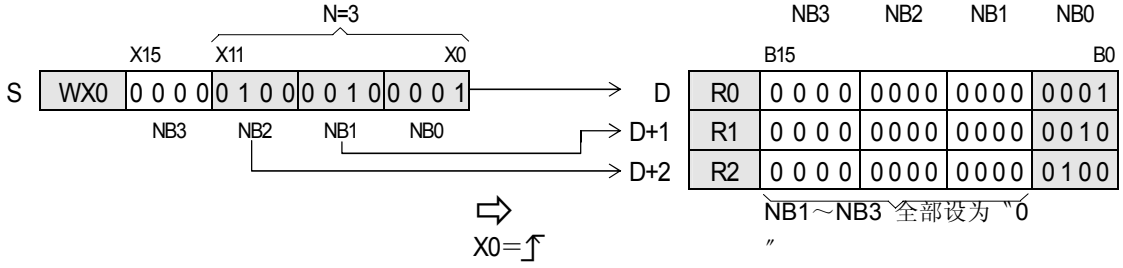
S，N，D 可结合 V、Z、P0~P9 作间接寻址应用

操作数 \ 范围	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR
	WX0 | WX240	WY0 | WY240	WM0 | WM1896	WS0 | WS984	T0 | T255	C0 | C255	R0 | R3839	R3840 | R3903	R3904 | R3967	R3968 | R4167	R5000 | R8071	D0 | D4095	16 位 正、负数	V、Z P0~P9
S	○	○	○	○	○	○	○	○	○	○	○	○	○	○
N	○	○	○	○	○	○	○	○	○	○	○	○	1~4	○
D		○	○	○	○	○	○		○	○*	○*	○		○

- 当分配控制〝EN〞=1 或〝EN↑〞（P 指令）由 0→1 时，将 S 中自最低位数 NB0 开始的连续 N 个位数（位数：Nibble，是由 4 个位所组成，由一缓存器的最低位 B0 开始往左，每连续 4 个位构成一位数，即 B0～B3 为第 0 个位数 NB0，B4～B7 为第 1 个位数，……），由低至高依序分配填到由 D 开始的 N 个缓存器的第 0 个位数 NB0 去。D 中各缓存器的 NB0 以外的位数则都填入 0。

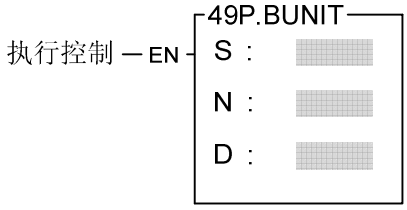
- 本指令只提供 WORD（16 位）指令，故最多只有 4 个 Nibble，所以 N 的有效值为 1～4，超出此范围，则 N 值错误旗号〝ERR〞设为 1，且本指令不执行。

- 左图程序范例是将 WX0 缓存器的 NB0～NB2 填写到 R0～R2 三个连续缓存器的 NB0 去。



FUN49 P BUNIT	字节数据结合 （BYTE UNITE）	FUN49 P BUNIT

阶梯图符号



S：要作字节(Byte)结合的来源缓存器起始号码

N：要结合的数据个数，单位为 Byte

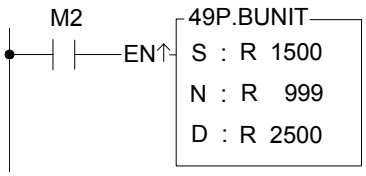
D：存放结合数据的起始缓存器号码

S，N，D 操作数可结合 V、Z、P0~P9 指标作间接寻址应用。

操作数 \ 范围	HR	ROR	DR	K
	R0 | R3839	R5000 | R8071	D0 | D4095	
S	○	○	○	
N	○	○	○	1~256
D	○	○*	○	

- 当执行控制〝EN〞=1 或〝EN↑〞（P 指令）由 0→1 时，将以 S 为起始的 N 个数据缓存器的低字节作数据结合，并将数据结合结果存放到以 D 为起始的缓存器群。
- 当结合的数据个数不正确时，本指令不执行。
- PLC 与智能型外围通过通讯端口来做连结整合时，如果通讯间的数据类型为二进制而非 ASCII 码方式时，有时需将所收到的 8 位(Byte)数据结合成 16 位(Word) 数据才能作后续处理，本指令即可有效作此应用。

程序范例：



范例说明：当 M2=1 时，以缓存器 R1500 为起始，缓存器 R999 的值为长度，作字节结合，并将结果存放到缓存器 R2500 为起始的缓存器群。

本范例假设 R999=10，则存放字节结合结果的缓存器为 R2500～R2504。

S

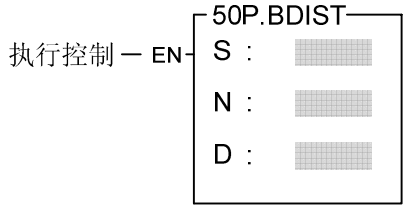
	High Byte	Low Byte
R1500	Don't care	Byte-0
R1501	Don't care	Byte-1
R1502	Don't care	Byte-2
R1503	Don't care	Byte-3
R1504	Don't care	Byte-4
R1505	Don't care	Byte-5
R1506	Don't care	Byte-6
R1507	Don't care	Byte-7
R1508	Don't care	Byte-8
R1509	Don't care	Byte-9

D

	High Byte	Low Byte
R2500	Byte-0	Byte-1
R2501	Byte-2	Byte-3
R2502	Byte-4	Byte-5
R2503	Byte-6	Byte-7
R2504	Byte-8	Byte-9

FUN50 P BDIST	字节数据分配 （BYTE DISTRIBUTE）	FUN50 P BDIST

阶梯图符号



S：要作字节(Byte)分配的来源缓存器起始号码

N：要分配的数据个数，单位为 Byte

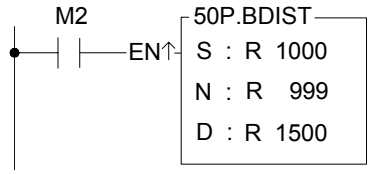
D：存放分配数据的起始缓存器号码

S，N，D 操作数可结合 V、Z、P0~P9 指标作间接寻址应用。

操作数 \ 范围	HR	ROR	DR	K
	R0 | R3839	R5000 | R8071	D0 | D4095	
S	○	○	○	
N	○	○	○	1~256
D	○	○*	○	

- 当执行控制〝EN〞=1 或〝EN↑〞（P 指令）由 0→1 时，将以 S 为起始的 N 个数据缓存器作字节数据分配，并将数据分配结果存放到以 D 为起始的缓存器群。
- 当分配的数据个数不正确时，本指令不执行。
- PLC 与智能型外围通过通讯端口来做连结整合时，如果通讯间的数据类型为二进制而非 ASCII 码方式时，需将 16 位(Word)数据分配成 8 位(Byte)数据后才能正确传送出数据，本指令即可有效作该应用。

程序范例：



范例说明：当 M2=1 时，以缓存器 R1000 为起始，缓存器 R999 的值为长度，作字节分配，并将结果存放到缓存器 R1500 为起始的缓存器群。

本范例假设 R999=9，则存放字节分配结果的缓存器为 R1500～R1508。

S

	High Byte	Low Byte
R1000	Byte-0	Byte-1
R1001	Byte-2	Byte-3
R1002	Byte-4	Byte-5
R1003	Byte-6	Byte-7
R1004	Byte-8	Don't care

D

	High Byte	Low Byte
R1500	00	Byte-0
R1501	00	Byte-1
R1502	00	Byte-2
R1503	00	Byte-3
R1504	00	Byte-4
R1505	00	Byte-5
R1506	00	Byte-6
R1507	00	Byte-7
R1508	00	Byte-8

FUN51 D P SHFL	向左位移 （SHIFT LEFT）	FUN51 D P SHFL

阶梯图符号

D：被位移的缓存器号码

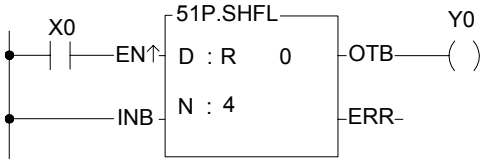
N：位移的位数

D，N 可结合 V、Z、P0~P9 作间接寻址应用

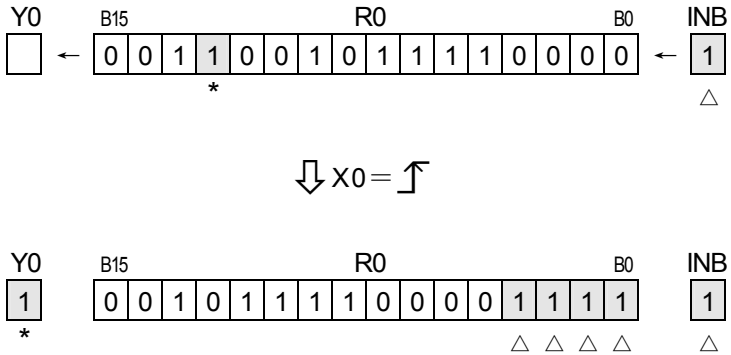
操作数 \ 范围	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR
	WX0 | WX240	WY0 | WY240	WM0 | WM1896	WS0 | WS984	T0 | T255	C0 | C255	R0 | R3839	R3840 | R3903	R3904 | R3967	R3968 | R4167	R5000 | R8071	D0 | D4095	1 | 或 | 16 32	V、Z P0~P9
D		○	○	○	○	○	○		○	○*	○*	○		○
N	○	○	○	○	○	○	○	○	○	○	○	○	○	○

● 当位移控制〝EN〞=1 或〝EN↑〞（P 指令）由 0→1 时，将 D 缓存器的数据向左（由低位往高位）连续移动 N 个位，在最低位 B0 左移后，其空位用移入位 INB 填补，同时将移出位 B15 或 B31（D 指令）的状态送到移出位〝OTB〞去。

● N 的有效范围在 16 位指令为 1～16，在 32 位（D 指令）则为 1～32，如果超出该范围则 N 值错误旗号〝ERR〞设为 1，且本指令不执行。



• 左图程序范例是将缓存器 R0 的数据连续向左位移 4 个位（4 次），下图为其执行结果。



FUN52 D P SHFR	向右位移 （SHIFT RIGHT）	FUN52 D P SHFR

阶梯图符号

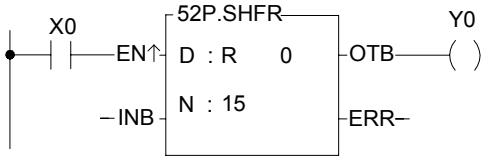
D：被位移的缓存器号码

N：位移的位数

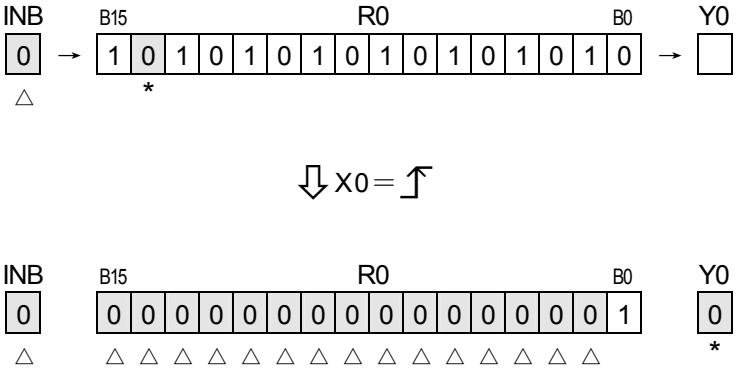
D，N 可结合 V、Z、P0~P9 作间接寻址应用

操作数 \ 范围	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR
	WX0 | WX240	WY0 | WY240	WM0 | WM1896	WS0 | WS984	T0 | T255	C0 | C255	R0 | R3839	R3840 | R3903	R3904 | R3967	R3968 | R4167	R5000 | R8071	D0 | D4095	1 | 或 | 16 32	V、Z P0~P9
D		○	○	○	○	○	○		○	○*	○*	○		○
N	○	○	○	○	○	○	○	○	○	○	○	○	○	○

- 当位移控制〞EN〞=1 或〝EN↑〞（P 指令）由 0→1 时，将 D 缓存器的数据向右（由高位往低位）连续移动 N 个位，在最高位 B15 或 B31（D 指令）右移后，其空位由移入位 INB 填补，同时将移出位 B0 的状态送到移出位〝OTB〞去。
- N 的有效范围在 16 位指令为 1～16，在 32 位（D 指令）则为 1～32，超出此范围则 N 值错误旗号〝ERR〞设为 1，且本指令不执行。



- 左图程序范例是将缓存器 R0 的数据连续向右位移 15 个位（15 次），下图为其执行结果。



FUN53 D P ROTL	向左旋转 （ROTATE LEFT）	FUN53 D P ROTL

阶梯图符号

D：被旋转的缓存器号码

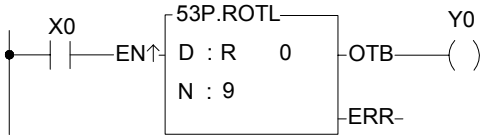
N：旋转的位数

D，N 可结合 V、Z、P0~P9 作间接寻址应用

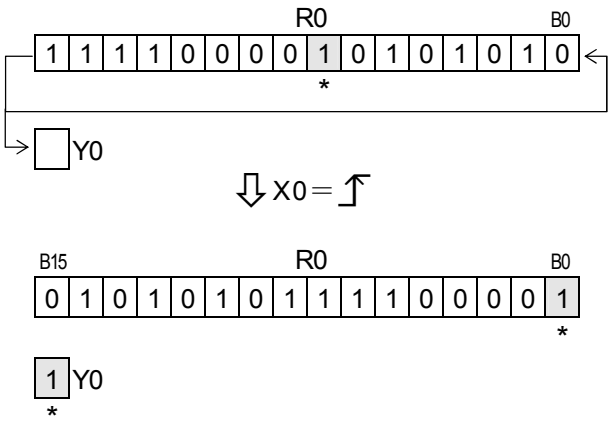
操作数 \ 范围	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR
	WX0 | WX240	WY0 | WY240	WM0 | WM1896	WS0 | WS984	T0 | T255	C0 | C255	R0 | R3839	R3840 | R3903	R3904 | R3967	R3968 | R4167	R5000 | R8071	D0 | D4095	1 1 | 或 | 16 32	V、Z P0~P9
D		○	○	○	○	○	○		○	○*	○*	○		○
N	○	○	○	○	○	○	○	○	○	○	○	○	○	○

● 当旋转控制〝EN〞=1 或〝EN↑〞（P 指令）由 0→1 时，将 D 缓存器的数据向左（低位往高位，即 16 位指令 B0→B1，B1→B2,.....，B14→B15，B15→B0。32 位指令则为 B0→B1，B1→B2,.....，B30→B31，B31→B0）连续旋转 N 位，同时并将旋出的 B15 或 B31（D 指令）位状态送到旋出位〝OTB〞去。

● N 的有效值在 16 位指令为 1～16，在 32 位（D 指令）则为 1～32，超出此范围则 N 值错误旗号〝ERR〞设为 1，且本指令不执行。

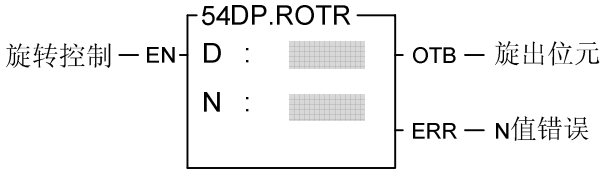


• 左图程序范例是将缓存器 R0 的数据连续向左旋转 9 次，下图为其执行结果。



FUN54 D P ROTR	向右旋转 （ROTATE RIGHT）	FUN54 D P ROTR

阶梯图符号



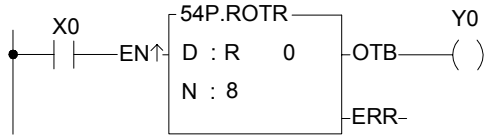
D：被旋转的缓存器号码

N：旋转的位数

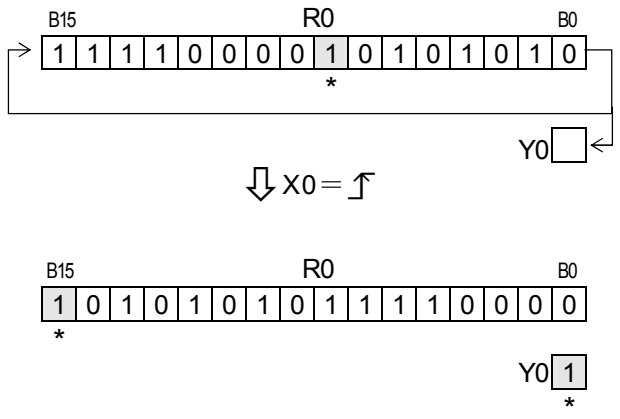
D，N 可结合 V、Z、P0~P9 作间接寻址应用

操作数 \ 范围	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR
	WX0 | WX240	WY0 | WY240	WM0 | WM1896	WS0 | WS984	T0 | T255	C0 | C255	R0 | R3839	R3840 | R3903	R3904 | R3967	R3968 | R4167	R5000 | R8071	D0 | D4095	1 | 或 | 16 32	V、Z P0~P9
D		○	○	○	○	○	○		○	○*	○*	○		○
N	○	○	○	○	○	○	○	○	○	○	○	○	○	○

- 当旋转控制〝EN〞=1 或〝EN↑〞（P 指令）由 0→1 时，将 D 缓存器的位数据向右（高位往低位，即 16 位指令 B15→B14，B14→B13,.....，B1→B0，B0→B15。32 位指令则为 B31→B30，B30→B29,.....，B1→B0，B0→B31）连续旋转 N 位，同时并将旋出的 B0 位状态送到旋出位〝OTB〞去。
- N 的有效值在 16 位指令为 1～16，在 32 位（D 指令）则为 1～32，超出该范围则 N 值错误旗号〝ERR〞设为 1，且本指令不执行。



- 左图程序范例是将缓存器 R0 的数据连续向右旋转 8 次，下图为其执行结果。



FUN55 D P B→G	二进制码转换格雷码 (BINARY-CODE TO GRAY-CODE CONVERSION)	FUN55 D P B→G

阶梯图符号

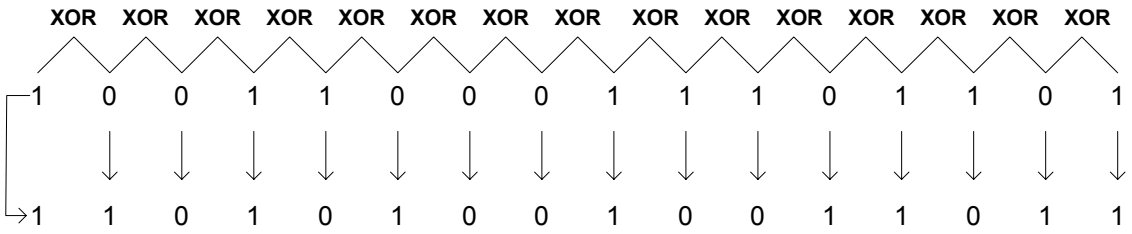
S：来源缓存器的起始号码

D：存放结果（格雷码）的缓存器起始号码

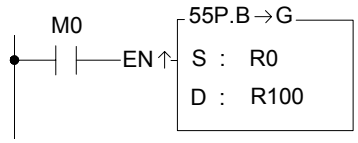
S，D 操作数可结合 V、Z、P0~P9 指标作间接寻址应用

操作数 \ 范围	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR
	WX0 | WX240	WY0 | WY240	WM0 | WM1896	WS0 | WS984	T0 | T255	C0 | C255	R0 | R3839	R3840 | R3903	R3904 | R3967	R3968 | R4167	R5000 | R8071	D0 | D4095	16 或 32 位正、负数	V、Z P0~P9
S	○	○	○	○	○	○	○	○	○	○	○	○		○
D		○	○	○	○	○	○		○	○*	○*	○		○

- 当执行控制〝EN〞=1 或〝EN↑〞（P 指令）由 0→1 时，将 S 缓存器的二进制码转换为格雷码。
- 当转换位小于 16 位时，需一个缓存器存放转换结果。大于或等于 16 位时需两个缓存器（D 指令）。
- 转换范例如下所示:



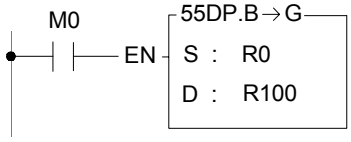
程序范例一 :



- 当 M0 由 OFF→ON 时，将 R0(二进制码)转换为格雷码，然后存入 R100。

R0＝1001010101010011B ➔ R100＝1101111111111010B

程序范例二 :



- 当 M0 ON 时，将 DR0(二进制码)转换为格雷码，然后存入 DR100。

DR0＝00110111001001000010111100010100B ➔ DR100＝00101100101101100011100010011110B

FUN56 D P G→B	格雷码转换二进制码 (GRAY-CODE TO BINARY-CODE CONVERSION)	FUN56 D P G→B

阶梯图符号

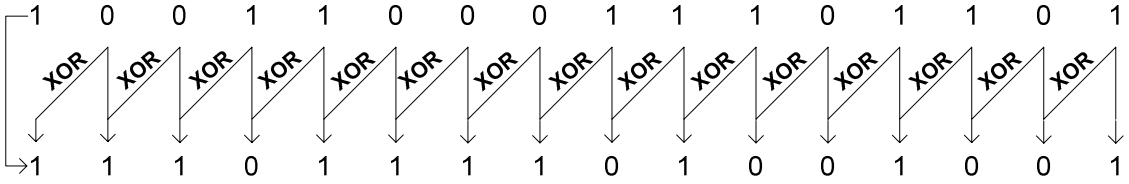
S：来源缓存器的起始号码

D：存放结果（格雷码）的缓存器起始号码

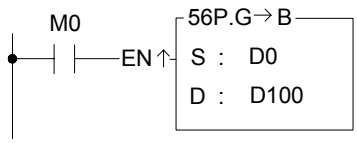
S，D 操作数可结合 V、Z、P0~P9 指标作间接寻址应用

操作数 \ 范围	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR
	WX0 | WX240	WY0 | WY240	WM0 | WM1896	WS0 | WS984	T0 | T255	C0 | C255	R0 | R3839	R3840 | R3903	R3904 | R3967	R3968 | R4167	R5000 | R8071	D0 | D4095	16 或 32 位正、负数	V、Z P0~P9
S	○	○	○	○	○	○	○	○	○	○	○	○		○
D		○	○	○	○	○	○		○	○*	○*	○		○

- 当执行控制〝EN〞=1 或〝EN↑〞（P 指令）由 0→1 时，将 S 缓存器的格雷码转换为二进制码。
- 当转换位小于 16 位时，需一个缓存器存放转换结果。大于或等于 16 位时需两个缓存器（D 指令）。
- 转换范例如下所示:



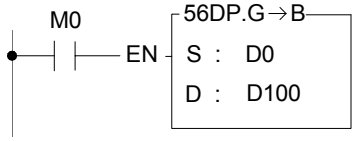
程序范例一 :



- 当 M0 由 OFF→ON 时，将 D0(格雷码)转换为二进制码格雷码，然后存入 D100。

D0＝1001010101010011B ➔ D100＝1110011001100010B

程序范例二 :

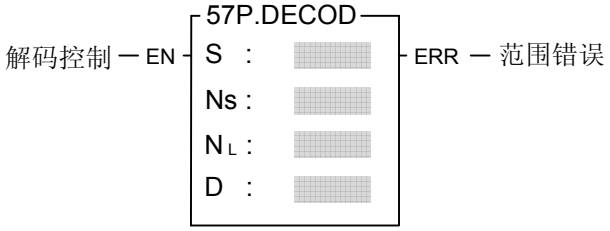


- 当 M0 ON 时，将 DD0(格雷码)转换为二进制码，然后存入 DD100。

DD0＝00110111001001000010111100010100B ➔ DD100＝00100101110001111100101000011000B

FUN57 P DECOD	解码 （DECODE）	FUN57 P DECOD

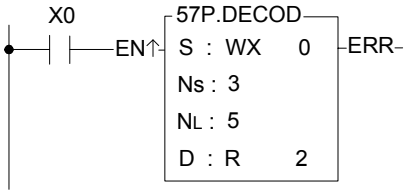
阶梯图符号



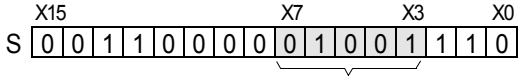
S ：译码的来源数据缓存器号码（16 位）
Ns：S 中要被译码的起始位
NL：译码值的长度（1～8 位）
D ：存放译码结果的缓存器起头号码
（2～256 点=1～16 Words）
S，Ns，NL、D 可结合 V、Z、P0~P9 作间接寻址应用

操作数 \ 范围	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR
	WX0 | WX240	WY0 | WY240	WM0 | WM1896	WS0 | WS984	T0 | T255	C0 | C255	R0 | R3839	R3840 | R3903	R3904 | R3967	R3968 | R4167	R5000 | R8071	D0 | D4095	16 或 32 位正、负数	V、Z P0~P9
S	○	○	○	○	○	○	○	○	○	○	○	○		○
Ns	○	○	○	○	○	○	○	○	○	○	○	○	0～15	○
NL	○	○	○	○	○	○	○	○	○	○	○	○	2～256	○
D		○	○	○	○	○	○		○	○*	○*	○		○

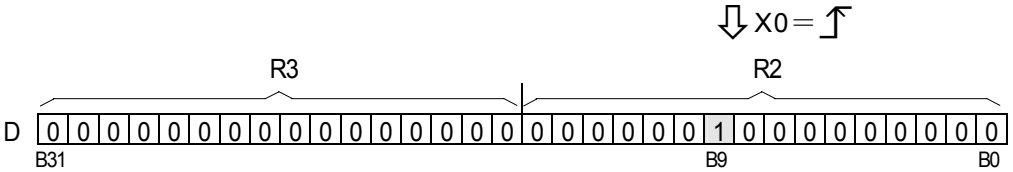
- 本指令所谓的译码是在宽度为 2^{NL} 个单点（D）中，将 S 中位 BNs～BNs+NL-1（称为解码值，而 BNs 为译码值的起始位，BNs+NL-1 则为其终止位）所指定的那个单点设为 1，其它设为 0。
- 当译码控制〝EN〞=1 或〝EN↑〞（P 指令）由 0→1 时，将 S 中 Ns 所指定的位开始，往左（高位方向）连续 NL 个位数据（即 BNs～BNs+BNL-1）取出当作译码值，并将译码结果 D 的 2^{NL} 个单点中，解码值所指定的那个单点设为 1，而其它单点全部设为 0。
- 本指令只有 16 位指令，S 只有 B0～B15，故 Ns 有效范围为 0～15，而解码值长度 NL 限制为 1～8 位。故解码结果 D 的宽度为 $2^{1\sim8}$ 个点=2～256 点=1～16 Words（未满 16 点仍占 1 个 Word），如果 Ns 或 NL 值超出上述范围则范围错误“ERR”设为 1，且本指令不执行。
- 如果终止位超出 S 的 B15，则往 S+1 的 B0 延伸。但终止位不得超过该种类操作数的最高极限（各单点操作数的最后一点或各缓存器操作数的最后一个 Word 的 B15），如果超出，则本指令只取起始位 BNs 到其最高极限间的位当译码值。



- 左图程序范例是自缓存器 WX0 中 X3 至 X7 连续 5 个位的数据取出译码后，将结果存到 R2 开始的 32 位缓存器中。



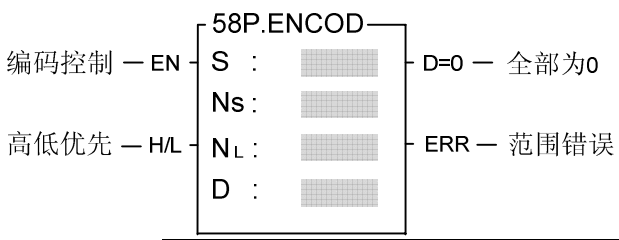
解码值长度 NL＝5，故为 X3～X7（其值为 9）



因 NL=5，故 D 的宽度为 2^5=32 点=2 个 WORD，即 D 为 R3R2 合成的 32 点宽度，而解码值为 01001=9，故 D 中的 B9（第 10 点）为 1，其它点都为 0。

FUN58 P ENCOD	编码 （ENCODE）	FUN58 P ENCOD

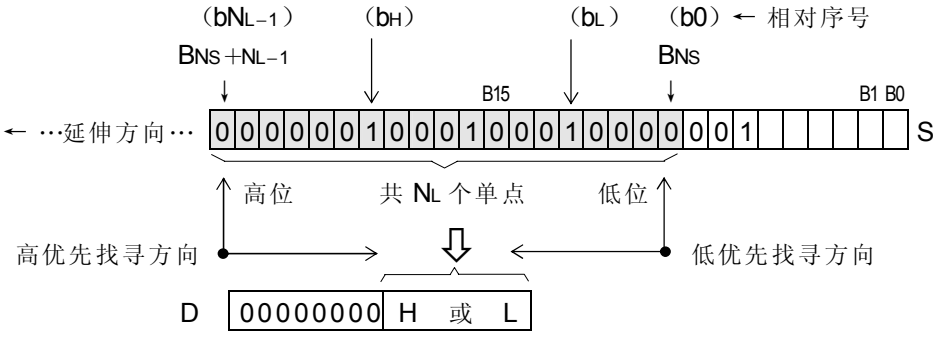
阶梯图符号



S ：被编码的缓存器起头号码
Ns：指定 S 中的一点为编码起始点
NL：编码的单点数目（2～256 点）
D ：存放编码结果的缓存器号码
（1 个 Word）
S，Ns，NL，D 可结合 V、Z、P0~P9 作间接寻址应用

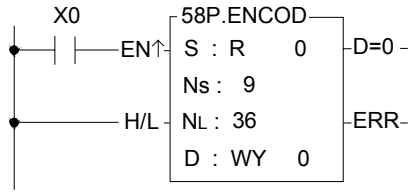
操作数 \ 范围	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR
	WX0 | WX240	WY0 | WY240	WM0 | WM1896	WS0 | WS98	T0 | T255	C0 | C255	R0 | R3839	R3840 | R3903	R3904 | R3967	R3968 | R4167	R5000 | R8071	D0 | D4095	16 位 正、负数	V、Z | P0~P9
S	○	○	○	○	○	○	○	○	○	○	○	○		○
Ns	○	○	○	○	○	○	○	○	○	○	○	○	0～15	○
NL	○	○	○	○	○	○	○	○	○	○	○	○	2～256	○
D		○	○	○	○	○	○		○	○*	○*	○		○

- 当编码控制〝EN〞=1 或〝EN↑〞（P 指令）由 0→1 时，将 S 中 Ns 所指定的单点开始往左（高位方向）的连续 NL 个单点 BNs～BNs+NL-1（BNs 称为编码起始点，其相对序号为 b0，BNs+NL-1 则称为编码终止点，相对序号为 bNL-1）取出，由左往右作高优先（H / L=1 时）或由右往左作低优先（H / L=0 时）编码（也就是找出第一个状态为 1 的单点，该单点的相对序号值即为编码值），再将编码值存到编码结果缓存器 D 的低字节（B0～B7），而 D 的高字节则填 0。

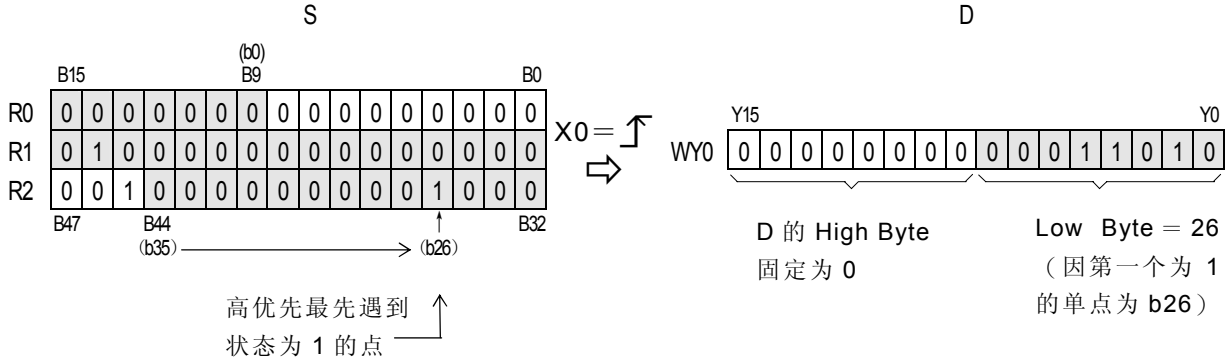


- 如上示意图范例，若为高优先编码，将先找到 bH（值为 12）；若为低优先编码则会先找到 bL（值为 4）。在 NL 个单点中至少要有一个状态为 1。若全为 0 则本指令不执行，同时将全部为 0 旗号〝D=0〞设为 1。
- 因 S 为一 16 位缓存器，故 Ns 可为 0～15，用以指定 S 中 B0～B15 的一点为编码起始点（b0）。而 NL 值可为 2～256，是用来界定编码终止点，即指定自起始点（b0）开始往左（高位方向）连 NL 个单点为编码区域（即 b0～bNL-1）。Ns 或 NL 值若超出上述范围则本指令不执行，并将范围错误旗号〝ERR〞设为 1。
- 若编码终止点（bNL-1）超出 S 的 B15，则继续往 S+1，S+2，..... 延伸，但最大不能超过该种类操作数的最高极限（各单点操作数的最后一点或各缓存器操作数的最后一个 Word 的 B15），若超出则本指令只取 b0 至其最高极限间的单点当作编码范围。

FUN58 P ENCOD	编码 （ENCODE）	FUN58 P ENCOD

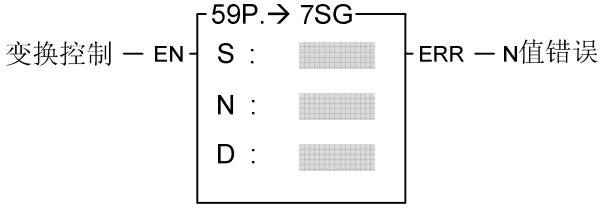


- 左图程序例为高优先编码的范例，当 X0 由 0→1 时，将 S（R0）中 Ns 所指的点 B9（b0）开始往左连续 36 个单点取出作高优先编码（因 H / L=1），也就是自 b35（编码终止点）开始往右找寻第一个状态为 1 的单点。本例的结果其相对序号为 b26，故 D 的值为 001AH=26，如下图所示。



FUN59 P →7SG	7 段显示码变换 （7-SEGMENT CONVERSION）	FUN59 P →7SG

阶梯图符号



S：变换的来源数据或其缓存器号码

N：指定 S 数据中连续 N+1 个位数（Nibble）

D：存放 7 段码结果的起始缓存器号码

S，N，D 可结合 V、Z、P0~P9 作间接寻址应用

操作数 \ 范围	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR
	WX0 | WX240	WY0 | WY240	WM0 | WM1896	WS0 | WS984	T0 | T255	C0 | C255	R0 | R3839	R3840 | R3903	R3904 | R3967	R3968 | R4167	R5000 | R8071	D0 | D4095	16 位 正、负数	V，Z P0~P9
S	○	○	○	○	○	○	○	○	○	○	○	○	○	○
N	○	○	○	○	○	○	○	○	○	○	○	○	0～3	○
D		○	○	○	○	○	○		○	○*	○*	○		○

- 当变换控制〝EN〞=1 或〝EN↑〞（P 指令）由 0→1 时，将 S 中连续 N+ 1 个位数（Nibble：由连续 4 个位所组成，即 S 的 B0～B3 为位数 0，B4～B7 为位数 1，……）转换成 7 段显示码后，将它存入 D。D 中 7 段码的摆放顺序为 a 段置于 B6，b 段置于 B5，……g 段置于 B0，B7 不用而固定为 0。请参考〝7 段码与显示字型表〞。
- 因本指令只限 16 位，因 S 只有 4 个 Nibble（NB0～NB3），故 N 的有效范围为 0～3，超出此范围则 N 值错误旗号〝ERR〞设为 1，且本指令不执行。
- N=0，代表一位数；N=1，代表二位数；N=2，代表三位数；N=3，代表四位数。
- 当使用永宏 7 段显示器扩展模块（FBs-7SGxx）且利用 FUN84 便利指令作译码与非译码的综合使用时，可结合 FUN 59 与 FUN 84 两个指令而简化程序的设计。

FUN59 P →7SG	7 段显示码变换 （7-SEGMENT CONVERSION）	FUN59 P →7SG

〈范例 1〉M1 由 OFF→ON 时，转十六进制值为 7 段显示码

- 左图范例是将 R0 的第 1 个位数（Nibble）转换为 7 段显示码并存放到 R100 的低字节（Low Byte），而 R100 的高字节（High Byte）保持不变。

原 R100＝0000H

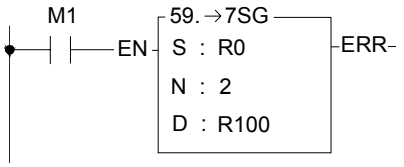
R0＝0001H ➔ R100＝0030H（1）

〈范例 2〉M1 ON 时，转十六进制值为 7 段显示码

- 左图范例是将 R0 的第 1 和第 2 个位数转换为 7 段显示码并存放到 R100。
- R100 的低字节存放第 1 位数。
- R100 的高字节存放第 2 位数。

R0＝0056H ➔ R100＝5B5FH（56）

〈范例 3〉M1 ON 时，转十六进制值为 7 段显示码



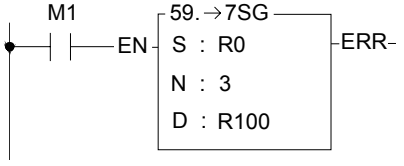
- 左图范例是将 R0 的第 1 和第 2 和第 3 个位数转换为 7 段显示码，并存放到 R100 与 R101。
- R100 的低字节存放第 1 位数。
- R100 的高字节存放第 2 位数。
- R101 的低字节存放第 3 位数。
- R101 的高字节保持不变。

原 R101=0000H

R0=0A48H ➔ R100=337FH（48）

R101=0077H（A）

〈范例 4〉M1 ON 时，转十六进制值为 7 段显示码



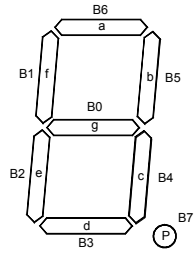
- 左图范例是将 R0 的第 1～4 位数转换为 7 段显示码，并存放到 R100 与 R101。
- R100 的低字节存放第 1 位数。
- R100 的高字节存放第 2 位数。
- R101 的低字节存放第 3 位数。
- R101 的高字节存放第 4 位数。

R0=2790H ➔ R100=7B7EH（90）

R101=6D72H（27）

数码变换指令

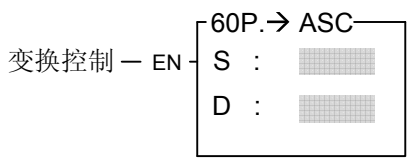
FUN59 P →7SG	7 段显示码变换 （7-SEGMENT CONVERSION）	FUN59 P →7SG

S 的位数（4 位）		7 段显示器结构	D 的字节（7 段显示码）								显示字形
十六进制	二进制		B7 ●	B6 a	B5 b	B4 c	B3 d	B2 e	B1 f	B0 g	
0	0000		0	1	1	1	1	1	1	0	
1	0001		0	0	1	1	0	0	0	0	
2	0010		0	1	1	0	1	1	0	1	
3	0011		0	1	1	1	1	0	0	1	
4	0100		0	0	1	1	0	0	1	1	
5	0101		0	1	0	1	1	0	1	1	
6	0110		0	1	0	1	1	1	1	1	
7	0111		0	1	1	1	0	0	1	0	
8	1000		0	1	1	1	1	1	1	1	
9	1001		0	1	1	1	1	0	1	1	
A	1010		0	1	1	1	0	1	1	1	
B	1011		0	0	0	1	1	1	1	1	
C	1100		0	1	0	0	1	1	1	0	
D	1101		0	0	1	1	1	1	0	1	
E	1110		0	1	0	0	1	1	1	1	
F	1111		0	1	0	0	0	1	1	1	

7 段码与显示字型表

FUN60 P →ASC	ASCII 码变换 （ASCII CONVERSION）	FUN60 P →ASC

阶梯图符号

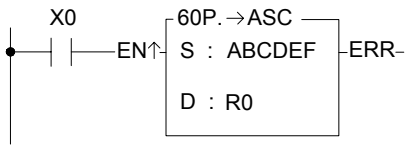


S：要变换成 ASCII 码的文 / 数字

D：存放 ASCII 码结果的缓存器起头号码

操作数 \ 范围	WY	WM	WS	TMR	CTR	HR	OR	SR	ROR	DR	文 / 数字
	WY0 | WY240	WM0 | WM1896	WS0 | WS984	T0 | T255	C0 | C255	R0 | R3839	R3904 | R3967	R3968 | R4167	R5000 | R8071	D0 | D4095	1～12 个英文或数字
S											○
D	○	○	○	○	○	○	○	○*	○*	○	

- 当变换控制〝EN〞=1 或〝EN↑〞（P 指令）由 0→1 时，将 S 的文 / 数字（最多可达 12 个字符）变换为 ASCII 码再存入由 D 起头的缓存器内，每两个字符将占用一个 16 位缓存器。
- 本指令的应用是将文 / 数字信息先存于程序中，等某些条件发生时，再将此文 / 数字信息变成 ASCII 码送出给外界能接受 ASCII 码的显示装置显示。



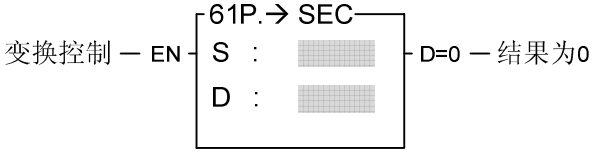
- 左图程序将 ABCDEF 6 个英文字母转换成 ASCII 码，再将的存到 R0 开始的连续 3 个缓存器去。

D

	高位 Byte	低位 Byte
R0	42（B）	41（A）
R1	44（D）	43（C）
R2	46（F）	45（E）

FUN61 P →SEC	时：分：秒→秒 （HOUR：MINUTE：SECOND→SECOND）	FUN61 P →SEC

阶梯图符号



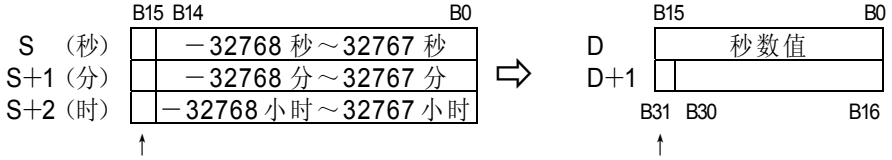
S ：要变换的时间数据缓存器起头号码

D ：存放结果的缓存器起头号码

操作数 \ 范围	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K
	WX0 | WX240	WY0 | WY240	WM0 | WM1896	WS0 | WS984	T0 | T255	C0 | C255	R0 | R3839	R3840 | R3903	R3904 | R3967	R3968 | R4167	R5000 | R8071	D0 | D4095	−117968399 | 117964799
S	○	○	○	○	○	○	○	○	○	○	○	○	○
D		○	○	○	○	○	○		○	○*	○*	○	

- 当变换控制〝EN〞=1 或〝EN↑〞（P 指令）由 0→1 时，将 S～S+2 的（时：分：秒）数据转换为等值的秒数值后存入由 D 和 D+1 合并而成的 32 位缓存器中。若结果=0，则〝D=0〞旗号设为 1。

- 永宏 PLC 指令中，和（时：分：秒）时间相关的指令（FUN61 和 62），其时间数据的格式如下图所示将自动合并连续 3 个缓存器（Word）来当时间值使用，其起头第一个为秒数（Second）缓存器，下一个为分数（Minute）缓存器，最后一个则为时数（Hour）缓存器。每个缓存器的 16 个位中只有 B14～B0 用以表示时间值，而其最高位 B15 则用以表示各该时间值的正、负。B15 为 0 表示该时间为正，B15 若为 1 则表示该时间值为负，B14～B0 的时间值是以二进制表示，当时间值为负时，B14～B0 则以 2 的补码表示。运算的秒数结果为（时：分：秒）三个缓存器的秒数加减结果。

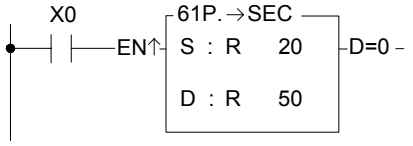


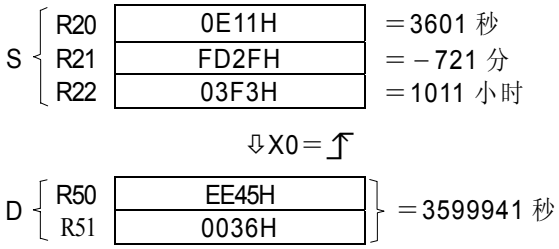
各缓存器的 B15 用以表示各时间值的正负

B31 用以表示秒数值的正负

- 任一（时：分：秒）时间数据，除了用 FUN61 或者 FUN62 两个指令去存取时才会自动合并使用，其它指都会将它视为个别的一般缓存器，不会自动合并使用，3 个缓存器之间没有任何关系，因此可个别对时、分、秒的任一数据运算，结果互不影响。

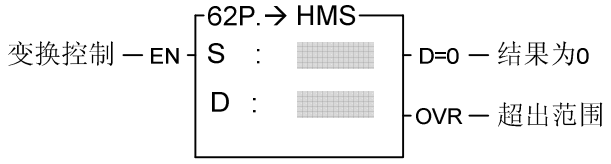
- 下图程序例，本指令会将 R20 开始的 3 个数据视为（时：分：秒）数据而将它转换成等值的秒数值后再存入 R50～R51 所组成的 32 位缓存器中，其结果如下图右。





FUN62 P →HMS	秒数→时：分：秒 （SECOND→HOUR：MINUTE：SECOND）	FUN62 P →HMS

阶梯图符号

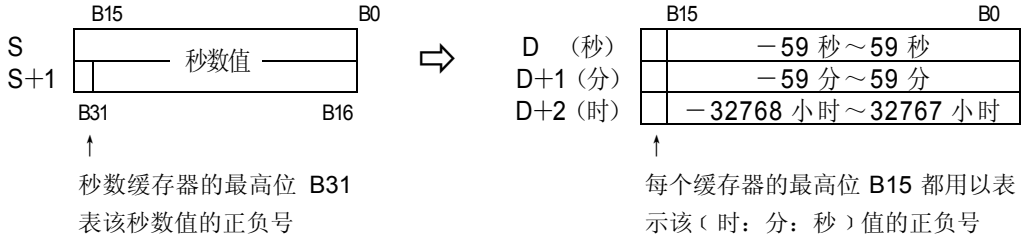


S：要变换的秒数数据缓存器起头号码

D：变换结果（时：分：秒）存放的缓存器起头号码

操作数 \ 范围	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K
	WX0 ｜ WX240	WY0 ｜ WY240	WM0 ｜ WM1896	WS0 ｜ WS984	T0 ｜ T255	C0 ｜ C255	R0 ｜ R3839	R3840 ｜ R3903	R3904 ｜ R3967	R3968 ｜ R4167	R5000 ｜ R8071	D0 ｜ D4095	−117968399 ｜ 117964799
S	○	○	○	○	○	○	○	○	○	○	○	○	○
D		○	○	○	○	○	○		○	○*	○*	○	

● 当变换控制〝EN〞=1 或〝EN↑〞（P 指令）由 0→1 时，将 S～S+1 的 32 位秒数数据转换为等值的（时：分：秒）时间值存入 D～D+2 三个连续的缓存器中，本指令所有数据都以二进制码表示（若为负值则以 2 的补码表示）。

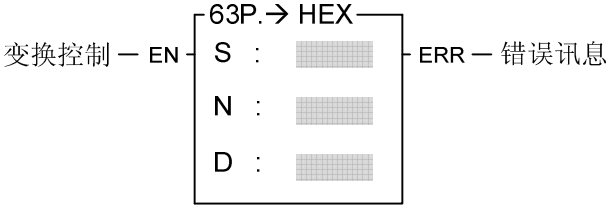


● 如上图所示本指令转成（时：分：秒）时间后，其（分：秒）值只可能为−59～59，而时数则可为−32768～32767 小时，因此 D 的最大极限是−32768 小时−59 分−59 秒至 32767 小时 59 分 59 秒，分别对应到 S 的秒数为−117968399 秒～117964799 秒。若 S 值超出此范围 D 将放不下，此时本指令便不执行，并将超出范围旗号〝OVR〞设为 1。若 S 为 0 则结果为零旗号〝D=0〞会设为 1。

● 下图程序为本指令执行的结果范例，注意缓存器内容值都为二进制值，其右边为其等效的 10 进制表示值。

FUN63 P →HEX	ASCII 码转换为十六进制值	FUN63 P →HEX

阶梯图符号



S：来源缓存器的起始号码

N：要转 ASCII 码为十六进制值的个数

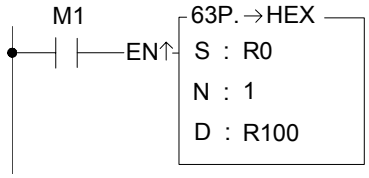
D：存放结果（十六进制值）的缓存器起始号码

S，N，D 可结合 V、Z、P0~P9 作间接寻址应用

操作数 \ 范围	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR
	WX0 | WX240	WY0 | WY240	WM0 | WM1896	WS0 | WS984	T0 | T255	C0 | C255	R0 | R3839	R3840 | R3903	R3904 | R3967	R3968 | R4167	R5000 | R8071	D0 | D4095	正数 16 位	V、Z P0~P9
S	○	○	○	○	○	○	○	○	○	○	○	○		○
N	○	○	○	○	○	○	○	○	○	○	○	○	1～511	○
D		○	○	○	○	○	○		○	○*	○*	○		○

- 当换变控制〝EN〞=1 或〝EN↑〞（P 指令）由 0→1 时，将 S 开始的连续 N 个 16 位缓存器（Low Byte 有效）的 ASCII 码转换为十六进制值，并将结果存入由 D 所指定开始的缓存器。每 4 个 ASCII 码由一个缓存器储存，未对应到 ASCII 码的缓存器内容维持原值不变。
- 当 N 的值为 0 或大于 511 时，运算不执行。
- 当 ASCII 码错误时（非 30H～39H 或 41H～46H），输出〝ERR〞ON。
- 此指令最大用途是将通讯端口 1 或通讯端口 2 所接收到外界 ASCII 外围（以 ASCII 码传送数值给 PLC）的 ASCII 数码转换为 CPU 能够直接处理的十六进制值。

〈范例 1〉M1 由 OFF→ON 时，转 ASCII 码为十六进制值

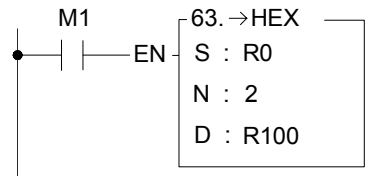


- 将 R0 的 ASCII 码转换为十六进制值并存入 R100 的 Nibble 0（Nibble1～Nibble3 不变）

原 R100＝0000H

R0＝0039H（9）➔ R100＝0009H

〈范例 2〉M1 ON 时，转 ASCII 码为十六进制值



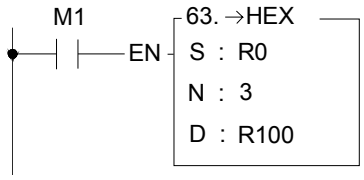
- 将 R0 与 R1 的 ASCII 码转换为十六进制值并存入 R100 的低字节（高字节不变）

R0＝0039H（9） 原 R100＝0000H

R1＝0041H（A）➔ R100＝009AH

FUN63 P →HEX	ASCII 码转换为十六进制值	FUN63 P →HEX

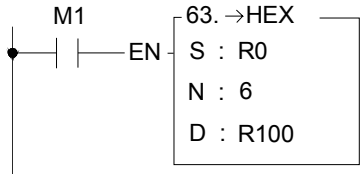
〈范例 3〉M1 ON 时，转 ASCII 码为十六进制值



- 将 R0～R2 的 ASCII 码转换为十六进制值并存入 R100（Nibble 3 不变）

R0＝0039H （9） 原 R100＝0000H
R1＝0041H （A）
R2＝0045H （E）➔ R100＝09AEH

〈范例 4〉M1 ON 时，转 ASCII 码为十六进制值

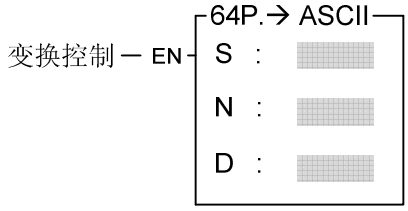


- 将 R0～R5 的 ASCII 码转换为十六进制值并存入 R100～R101

R0＝0031H（1） 原 R100＝0000H
R1＝0032H（2） R101＝0000H
R2＝0033H（3）
R3＝0034H（4）
R4＝0035H（5）➔ R100＝3456H
R5＝0036H（6） R101＝0012H

FUN64 P →ASCII	十六进制值转换为 ASCII 码	FUN64 P →ASCII

阶梯图符号



S：来源缓存器的起始号码

N：要转十六进制值为 ASCII 码的个数

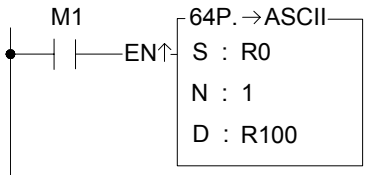
D：存放结果（ASCII 码）的缓存器起始号码

S，N，D 可结合 V、Z、P0~P9 作间接寻址应用

操作数 \ 范围	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR
	WX0 | WX240	WY0 | WY240	WM0 | WM1896	WS0 | WS984	T0 | T255	C0 | C255	R0 | R3839	R3840 | R3903	R3904 | R3967	R3968 | R4167	R5000 | R8071	D0 | D4095	正数 16 位	V、Z P0~P9
S	○	○	○	○	○	○	○	○	○	○	○	○		○
N	○	○	○	○	○	○	○	○	○	○	○	○	1~511	○
D		○	○	○	○	○	○		○	○*	○*	○		○

- 当变换控制〝EN〞=1 或〝EN↑〞（P 指令）由 0→1 时，将 S 开始的缓存器连续 N 个位数（4 位）的十六进制值转换为 ASCII 码，并将结果存入 D 所指定开始缓存器的低字节。(高字节维持原值不变)。
- 当 N 的值为 0 或大于 511 时，运算不执行。
- 该指令最大用途是将 PLC 处理完的数值数据转换为 ASCII 码通过通讯端口 1 或通讯端口 2 传送给 ASCII 外围设备。

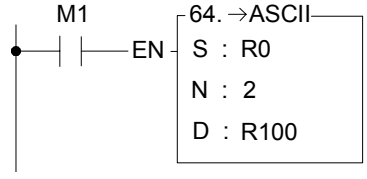
〈范例 1〉M1 由 OFF→ON 时，转十六进制值为 ASCII 码



- 将 R0 的 Nibble 0 转换为 ASCII 码并存入 R100 (高字节不变)

R0＝0009H ➔ R100＝0039H（9）

〈范例 2〉M1 ON 时，转十六进制值为 ASCII 码

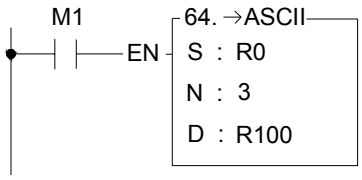


- 将 R0 的 NB0～NB1 转换为 ASCII 码并存入 R100～R101（高字节都维持原值不变）

R0＝009AH ➔ R100＝0039H（9）
R101＝0041H（A）

FUN64 P →ASCII	十六进制值转换为 ASCII 码	FUN64 P →ASCII

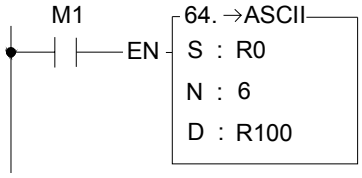
〈范例 3〉M1 ON 时，转十六进制值为 ASCII 码



- 将 R0 的 NB0～NB2 转换为 ASCII 码并存入 R100～R102

R0＝0123H ➔ R100＝0031H（1）
R101＝0032H（2）
R102＝0033H（3）

〈范例 4〉M1 ON 时，转十六进制值为 ASCII 码

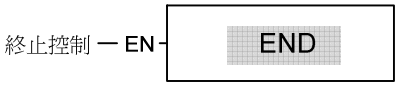


- 将 R0～R1 的 NB0～NB5 转换为 ASCII 码并存入 R100～R105

R0＝3456H ➔ R100＝0031H（1）
R1＝0012H R101＝0032H（2）
R102＝0033H（3）
R103＝0034H（4）
R104＝0035H（5）
R105＝0036H（6）

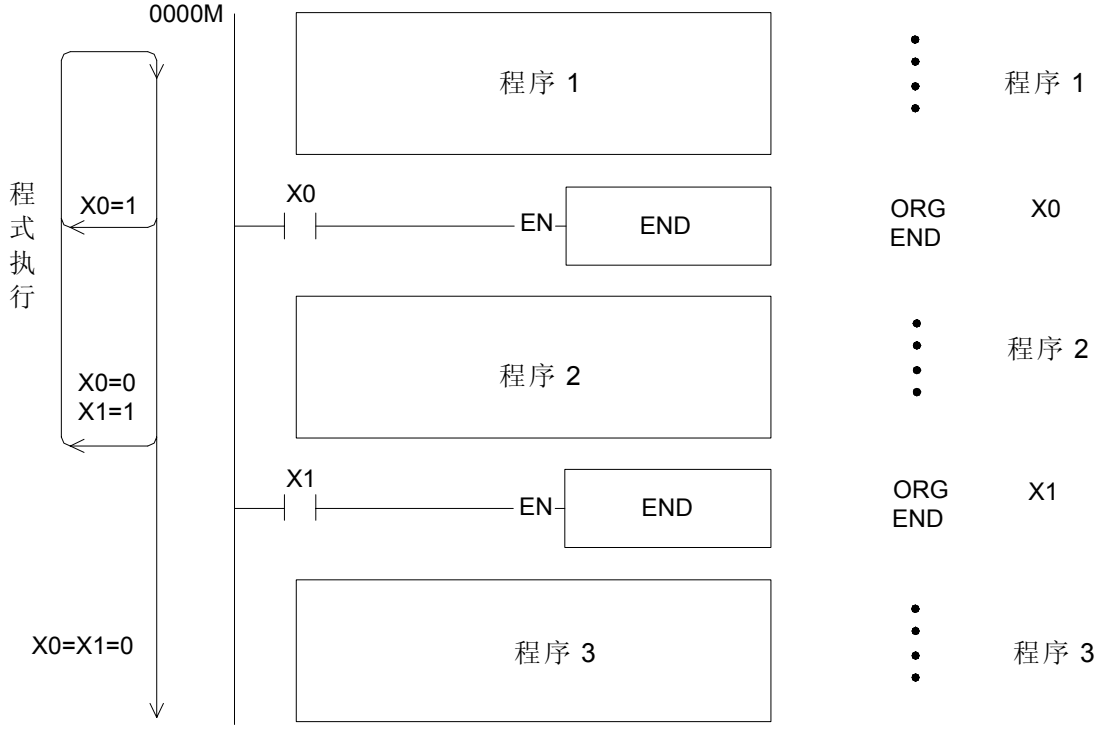
END	程序终止 （PROGRAM END）	END

階梯圖符號



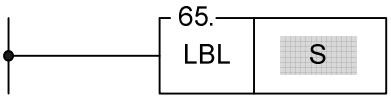
无操作数

- 当终止控制〝EN〞=1 时，本指令动作，立即结束本次的程序扫描，也就是在 END 指令后的程序虽然存在，但却不会被执行。当〝EN〞=0 时本指令不执行（当作无此指令），在 END 指令后的程序会继续被执行。
- 本指令可在程序中多点放置，而以其输入（终止控制〝EN〞）来控制程序执行的终止处，特别有利于除错或测试。
- 程序的最后并不一定要有 END 指令，CPU 会自动检查程序的结束。



FUN65 LBL	标记 （LABEL）	FUN65 LBL

阶梯图符号



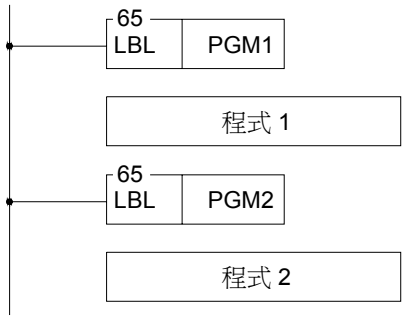
S：英文 / 数字 1～6 字

- 本指令用于标示程序中某一特定地址，以供程序跳跃（JUMP）到此标记所在的地址来执行，或当作中断服务程序或子程序的名称，以供中断或呼叫（CALL）用。若不需作跳跃或呼叫等的流程控制，也可作标记来对程序作批注，以利程序的辨识或提高可读性。
- 本指令只当程序地址标记来供流程控制或批注用，指令本身不会执行任何动作，程序中有没有本指令，程序执行结果都不受其影响。
- 标记名称可以 1～6 个任意英文字母或数字组成，但不得重复，且下列标记名称是保留给中断功能使用，称为〝保留字〞，一般程序标记不得使用：

保　　留　　字	中　断　服　务　程　序　名　称
X0+I～X15+I（INT0～INT15） X0–I～X15–I（INT0–～INT15–）	外部 X0～X15 的中断服务程序名称
HSC0I～HSC7I	HSC0～HSC7 的中断服务程序名称
1MSI（1MS）、2MSI（2MS），3MSI（3MS），4MSI（4MS），5MSI（5MS），10MSI（10MS），50MSI（50MS），100MSI（100MS）	PLC 内部 1mS，2mS……100mS 等 8 种定时中断的服务程序名称
HSTAI（ATMRI）	高速定时中断服务程序名称
PSO0I～PSO3I	脉冲输出结束的中断服务程序名称

除非所标注的程序确实是上述中断所对应的服务程序才可用上述的名称，其它地方不能使用，否则当中断发生时，PLC 会把标记的一般程序当作中断程序执行，而造成错误或当机。

下图例为标记只当作程序批注（未被呼叫或跳跃至此标记）的范例，至于标记在跳跃控制的应用请参考跳跃（JMP）指令的说明，标记当子程序名称则请参考呼叫（CALL）指令的说明。

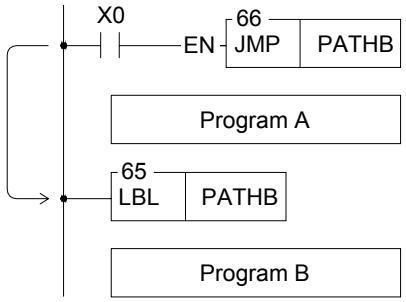


FUN66 P JMP	跳跃 （JUMP）	FUN66 P JMP

阶梯图符号

LBL：要跳跃的程序标记

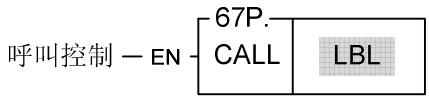
- 当跳跃控制〝EN〞=1 或〝EN↑〞（P 指令）由 0→1 时，PLC 直接跳到其后标记 LBL 所在位置，继续往下执行程序。
- 本指令的应用特别适合在特定状况发生才需执行某部分程序的应用，平常不执行以节省时间。以及在线圈多重输出的应用场合，再以输入控制选择执行某一段程序的应用。
- 本指令程序跳跃可往回跳（即跳回的 LBL 地址比该 JMP 指令所在的地址要小），但需注意如往回跳致使扫描时间延长超过 Watchdog Timer 所设定的时间，则 PLC 会发生 WDT 中断，而停止运转，并发出错误信号。
- 跳跃指令只限于主程序区跳主程序区，或子程序区跳子程序区，不能跨越主 / 子程序区作跳跃。



- 左图中当 X0=1，则程序执行将由 JMP 指令所在处直接跳到 LBL 名称为 PATHB 的地方往下执行，故程序 A 被跳过，A 中所有指令都不执行，和程序 A 相关的单点或缓存器状态都保持不变（如同无 A 这段程序）。

FUN67 P CALL	呼叫 （CALL）	FUN67 P CALL

阶梯图符号

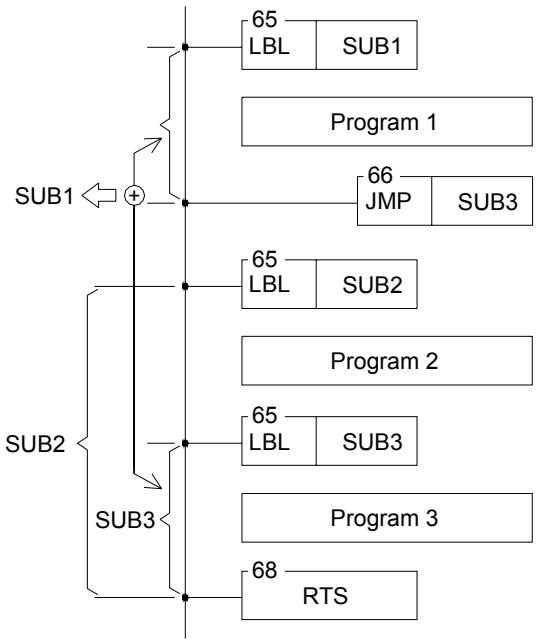


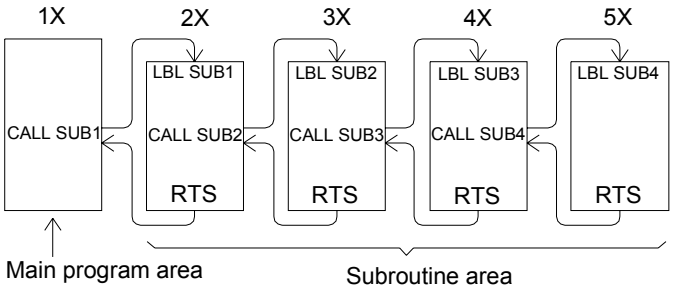
LBL：被呼叫的子程序标记名称

- 当呼叫控制〝EN〞=1 或〝EN↑〞（P 指令）由 0→1 时，PLC 将呼叫（执行）标记名称与被呼叫的标记名称相同的子程序，在子程序执行前 PLC 会先将子程序执行完后所需返回的地址（该 CALL 指令的下一个地址）存入 CPU 内部的堆栈（STACK）内，然后再去执行呼叫的子程序，直到遇到子程序中的〝子程序返回指令 RTS〞后才将先前存入堆栈的返回地址取回，而从返回地址处的指令往下继续执行程序。

- 子程序的最后都要有〝子程序返回指令 RTS〞，否则将造成执行错误或死机，但多个子程序可共享一个 RTS 指令(此即所谓的多进入点子程序，这种子程序的进入点不同，返回点却一致)，如右图例的子程序 SUB1～3。

- 主过程调用子程序后子程序尚可呼叫其它子程序(即所谓巢式子程序),最多可达 5 层(中断＋呼叫)。

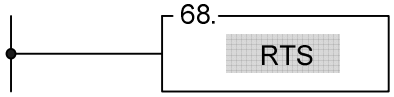




- 中断服务程序（HSC0I～HSC7I、PSO0I～PSO3I、X0+I～X15+I / INT0～INT15、X0–I～X15–I / INT0–～INT15–、HSTAI / ATMRI、1MSI / 1MS、2MSI / 2MS、3MSI / 3MS、4MSI / 4MS、5MSI / 5MS、10MSI / 10MS、50MSI / 50MS、100MSI / 100MS）也算是子程序的一种，也存放在子程序区内，但中断服务程序的呼叫，是利用硬件触发信号促使 CPU 去执行对应的中断服务程序（我们称为中断服务程序的召用）。中断服务程序也能再呼叫子程序或再召用中断服务程序，但因其本身就是子程序（已占一层），因此最多只能再呼叫或召用四层子程序或中断服务程序，请参考 RTI 指令的说明。

FUN68 RTS	子程序返回 （RETURN FROM SUBROUTINE）	FUN68 RTS

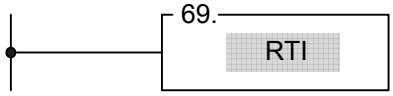
阶梯图符号



- 本指令用于表示一个子程序的终了，因此只能出现在子程序区内，其输入侧无控制信号，故无法串联任何组件，本指令单独就是一个完整指令，是直接接到母线上。
- 当 PLC 执行到本指令时，表示子程序已执行完毕，因此会将先前存入堆栈中的返回地址取回，以便 PLC 回到先前呼叫子程序的下一个指令，继续往下执行程序。
- 如果在子程序中执行不到 RTS 指令，则程序流程将不再正确，系统堆栈也会被破坏（M1933 ON），并造成系统失控。因此，无论流程如何控制，都需要确保所有子程序都会执行到 RTS 指令。
- RTS 指令的应用请参考 CALL 指令的说明。

FUN69 RTI	中断返回 （RETURN FROM INTERRUPT）	FUN69 RTI

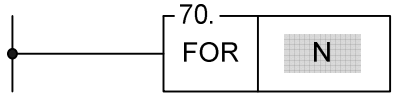
阶梯图符号



- 本指令的功能和 RTS 类似，只是 RTS 是用于子程序的最后，而 RTI 则用于中断服务程序的最后，请参考 RTS 指令的说明。
- 多个中断服务程序可共享一个 RTI 指令，其用法和多个子程序可共享一个 RTS 指令一样，请参考呼叫（CALL）指令的说明。
- 中断和呼叫的差异只有在呼叫是由用户自行定义子程序的名称（标记 LBL），然后在主程序或其它子程序中有呼叫指令并指名该子程序的标记，这样当 PLC 执行到该呼叫指令（CALL），且其输入〝EN〞=1 或〝EN↑〞（P 指令）由 0→1 时，PLC 即会去呼叫（执行）该子程序。而中断服务程序的执行则是直接以硬件信号来中断 CPU，要 CPU 暂停其它较次要的工作，而来执行该硬件信号所对应的中断服务程序（我们称为中断服务程序召用）。因此与呼叫必须扫描到该呼叫指令才会执行的作法相比，中断则为更实时（Real Time）的作法。此外因中断服务程序无法指名呼叫，因此我们以特定的〝保留字〞标记名称来对应 PLC 所提供的各种中断（详见 FUN65 说明），例如保留字 X0+I 指定给输入点 X0 所发生的中断，只要子程序中有标记为 X0+I 的程序，当输入点 X0 中断允许发生（X0：↑），PLC 就会立即暂停其它低优先级的程序扫描工作，而马上跳到子程序中标记为 X0+I 的地址去执行程序。
- 如果中断发生时，CPU 正在处理比该中断优先级更高（如硬件高速计数器中断）或优先级一样的中断（请参考 9-3 节中断的优先级），则 PLC 会等执行完上述所有中断服务程序后才会处理此中断。
- 如果在中断服务程序中执行不到 RTI 指令，则 PLC 的系统堆栈会被破坏、程序流程错乱，而有可能引起严重死机。因此，无论流程如何控制，都需要确保任一个中断服务程序都会执行到 RTI 指令。
- 关于中断的详细说明与使用方法范例请参考第 9 章的说明。

FUN70 FOR	循环开始 （FOR）	FUN70 FOR

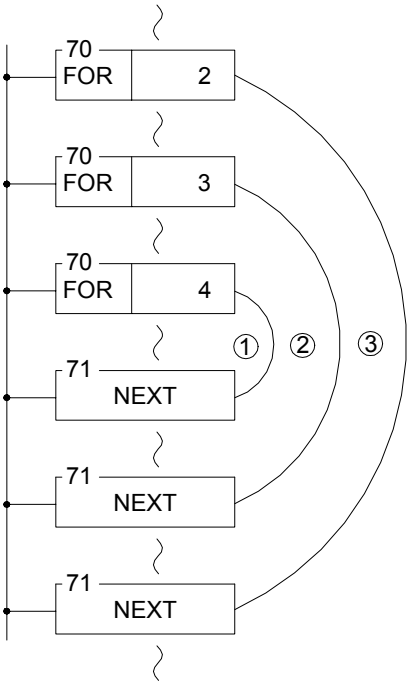
阶梯图符号



N：循环执行次数

操作数 \ 范围	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K
N	WX0 | WX240	WY0 | WY240	WM0 | WM1896	WS0 | WS984	T0 | T255	C0 | C255	R0 | R3839	R3840 | R3903	R3904 | R3967	R3968 | R4167	R5000 | R8071	D0 | D4095	1 | 16383
	○	○	○	○	○	○	○	○	○	○	○	○	○

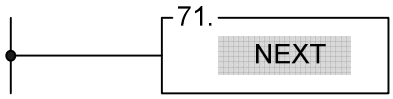
- 本指令无输入控制，是直接接在母线上的，不能串接任何组件。
- FOR 指令和 NEXT 指令所包夹的程序形成一程序循环（循环程序的开头为 FOR 的次一个指令，结尾为 NEXT 之前一个指令），当 PLC 执行到 FOR 指令时，首先记下该指令后的 N（循环执行次数），然后将该循环内的程序从头到尾连续执行 N 次后，跳离该循环，继续往下（NEXT 的下一指令开始）执行。
- 循环可为巢式结构，即循环内包含着循环，犹如洋葱一般，一个循环称为一层最多可达 5 层。FOR 和 NEXT 指令必须成对使用，第一个 FOR 指令和最后一个 NEXT 为巢式循环的最外（第一）层。而第二个 FOR 指令和倒数第二个 NEXT 指令为第二层，……最后一个 FOR 指令和第一个 NEXT 指令形成最内层的循环。



- 左图例循环①将被执行 4×3×2=24 次，循环②将被执行 3×2=6 次，而循环③则会执行 2 次。
- 如果有 FOR 指令而无 NEXT 指令与的对应，或巢式循环的 FOR 和 NEXT 指令未配对使用，或 FOR、NEXT 顺序颠倒，都将造成语法错误，程序无法执行。
- 循环中不可使用 JMP 指令跳出循环，否则 PLC 的系统堆栈会被破坏、程序流程错乱，而有可能引起严重当机。
- N 的有效范围为 1～16383 次，超出此范围，PLC 都将视为 1 次。N 的次数若太大，或循环程序太长，可能造成 Watchdog 发生，请注意。

FUN71 NEXT	循环结束	FUN71 NEXT

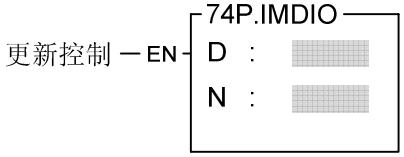
阶梯图符号



- 本指令和 FOR 指令配合形成一个程序循环。指令本身无输入控制，是直接接在母线上的，不能串接任何组件。
- 未执行到 FOR 指令，绝不可以执行到 NEXT 指令，否则有可能造成 PLC 当机。
- 其应用请参考前页 FOR 指令的说明。

FUN74 P IMDIO	实时 I / O 更新 （IMMIDIATE I/O REFRESH）	FUN74 P IMDIO

阶梯图符号



D：要更新的 I/O 点开头号码
N：要更新的 I/O 点数

操作数 \ 范围	X	Y	K
	主机上的 Xn	主机上的 Yn	1 | 36
D	○	○	
N			○

- PLC 系统的输入 / 输出信号更新通常在程序执行前先一次抓取全部的输入信号，然后开始扫描程序，等全部扫描结束才将所有输出结果一次送到输出点，这样的输入动作到输出反应至少会有一个扫描时间的延时（最大为 2 个扫描时间）。本指令的作法则为遇到本指令便立即去抓取或送出指令所指定的输入信号或输出信号，这样可获得最实时快速的输入 / 输出反应。

- 当更新控制〝EN〞=1 或〝EN↑〞（P 指令）由 0→1 时，将 D 所指定的 I（输入点）或 O（输出点）开始的 N 个输入点或输出点（即 D～D+N–1）状态更新。

- PLC 的实时 I/O 更新的 I/O 点仅限于在主机上的 I/O 点。下表为 10、14、20、24、32、40、60 点主机允许的实时 I/O 号码：

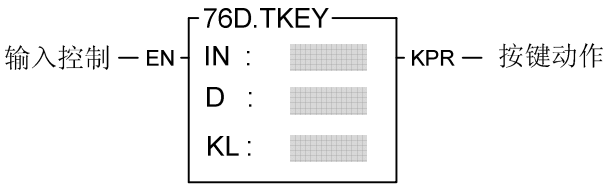
允许号码 \ 主机 I/O 点数	10 点	14 点	20 点	24 点	32 点	40 点	60 点
输入点	X0～X5	X0～X7	X0～X11	X0～X13	X0～X19	X0～X23	X0～X35
输出点	Y0～Y3	Y0～Y5	Y0～Y7	Y0～Y9	Y0～Y11	Y0～Y15	Y0～Y23

- 如果程序中的实时 I/O 点的范围超出主机的输入点或输出点号码（例如程序中 D=X7，N=10，则表示要实时抓取 X7～X16 等 10 个输入点信号，而假设主机为 32 点 I/O 机种，其输入点最大为 X19，明显地 X20 已经超出该主机的输入点号码）则 PLC 将无法运转（STOP，ERR 灯亮）同时 M1931 错误旗号设定为 1。

- 本指令执行时，虽然 PLC 会立即去抓取或送出实时输入 / 输出信号，但在输入点上的硬件或软件积分的延时或输出点的动作延时（如继电器或晶体管等输出组件的动作反应时间）仍然存在，请特别注意。

FUN76 D TKEY	10 进位数字按键 （DECIMAL KEY-IN）	FUN76 D TKEY

阶梯图符号



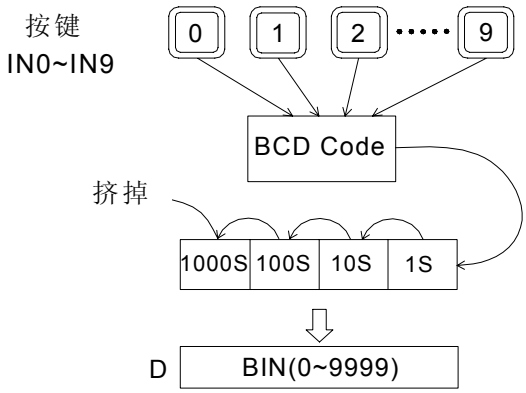
IN ： 按键输入点
D ： 存放按键数字的缓存器号码
KL： 输入按键的对应继电器起头号码
D 可结合 V、Z、P0~P9 作间接寻址应用

操作数 \ 范围	X	Y	M	S	WY	WM	WS	TMR	CTR	HR	OR	SR	ROR	DR	XR
	X0 | X240	Y0 | Y240	M0 | M1896	S0 | S984	WY0 | WY240	WM0 | WM1896	WS0 | WS984	T0 | T255	C0 | C255	R0 | R3839	R3904 | R3967	R3968 | R4167	R5000 | R8071	D0 | D4095	V、Z | P0~P9
IN	○														
D					○	○	○	○	○	○	○	○*	○*	○	○
KL		○	○	○											

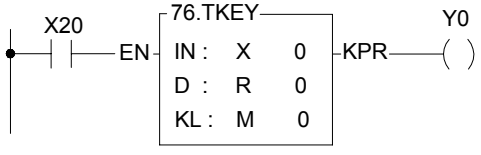
- 本指令指定由 IN 开始的连续 10 个输入点（IN0～IN9）依序代表十进制数的 0～9（BCD 码为 0000～1001），根据这些输入点被压下（ON）的先后顺序可输入 4 个或 8 个十进制数到 D 所指定的缓存器去。

- 当输入控制〝EN〞=1，本指令会去检查 IN 开始的 10 个输入点并将〝ON〞输入点所代表的十进制数存入 D 中，等该输入点放开后，再检查下一个〝ON〞的输入点，再将其所代表的数字挤进 D 中（先存入的为高位数，后存入的为低位数）。在 16 位指令中 D 可存放 4 位数，而 32 位指令可存放 8 位数，超出时则挤掉先存入的（即高位数的数字）。IN 开始的 10 个输入点按键状况将会被记录在由 KL 开始的 10 个对应的继电器上，同时只要有任何一个输入点被按下（ON），则按键动作旗号〝KPR〞即变为 1。在同一时间内 IN0～IN9 中只能有一个被按下，若超过一个只取最先按下的，下图为 16 位指令的功能示意图（32 位也一样，但数值可达〝千万〞）。

- 当输入控制〝EN〞=0 时，本指令不执行，同时清除〝KPR〞输出及 KL 继电器的状态为 0，但缓存器 D 的数值则保持不变。

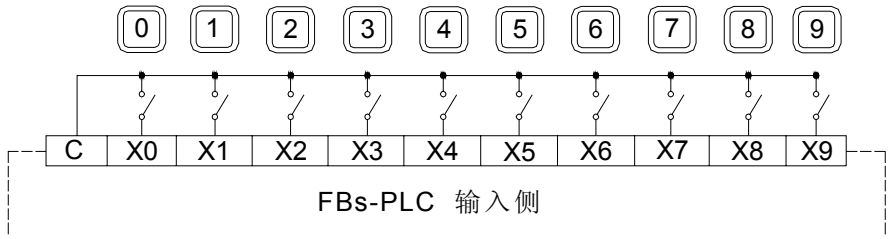


• 下图程序指定输入点 X0 代表数字〝0〞，X1 代表〝1〞，……，按键状况则以 M0 记录 X0 的动作，M1 记录 X1 的动作……，输入的数值存在 R0 缓存器中。

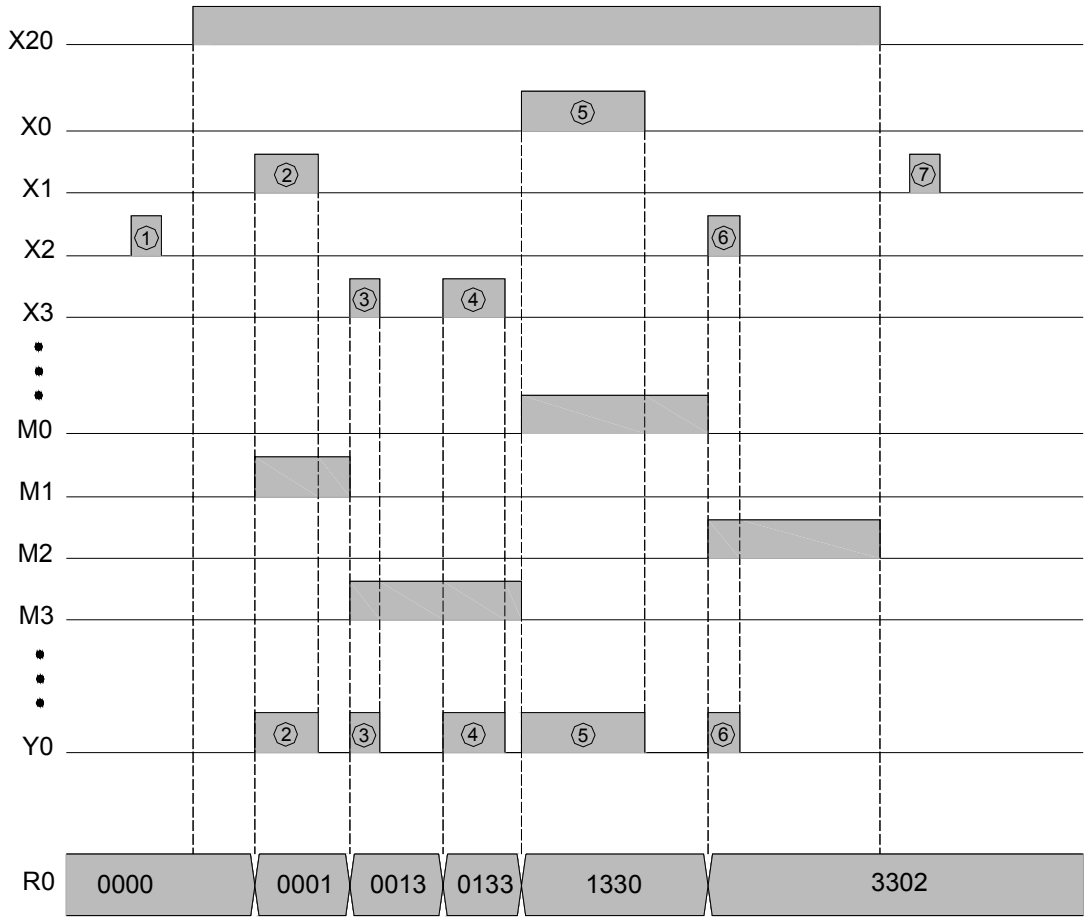


FUN76 D TKEY	10 进位数字按键 （DECIMAL KEY-IN）	FUN76 D TKEY

下图为本范例的实际输入配线图：

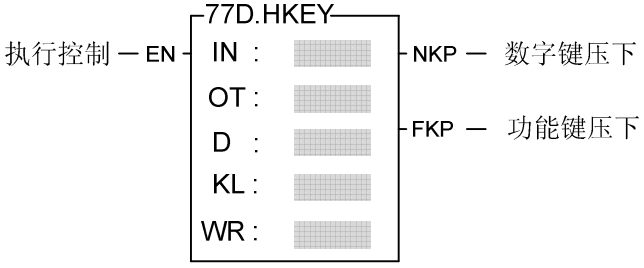


- 假设 X0～X3 的按键顺序如下图的①②③④⑤⑥⑦的顺序，因①和⑦按下时 X20 为 0，而不发生作用，有效的仅为②③④⑤⑥，而此 5 个按键的第一个按键②因已超出 4 位而被挤掉，只剩下③④⑤⑥的按键数字 3302 存在缓存器 R0 中。



FUN77 D HKEY	16 个键多任务输入 （HEX-KEY INPUT）	FUN77 D HKEY

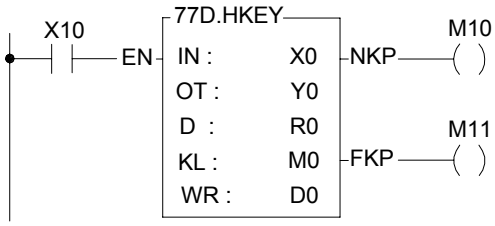
阶梯图符号



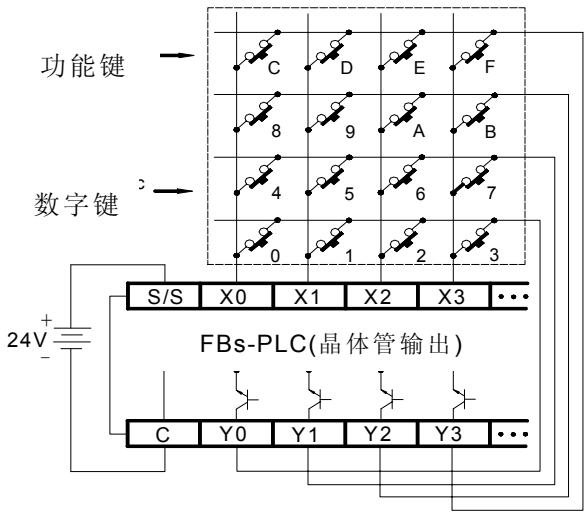
IN ：按键扫描输入点号码
OT：多任务扫描输出点号码
D ：存放〝按键数字〞的缓存器号码
KL：记录〝动作键〞的继电器起头号码
WR：工作缓存器，其它地方不可重复使用
D 可结合 V、Z、P0~P9 作间接寻址应用

操作数 \ 范围	X	Y	M	S	WY	WM	WS	TMR	CTR	HR	OR	SR	ROR	DR	XR
	X0 | X240	Y0 | Y240	M0 | M1896	S0 | S984	WY0 | WY240	WM0 | WM1896	WS0 | WS984	T0 | T255	C0 | C255	R0 | R3839	R3904 | R3967	R3968 | R4167	R5000 | R8071	D0 | D4095	V、Z | P0~P9
IN	○														
OT		○													
D					○	○	○	○	○	○	○	○*	○*	○	○
KL		○	○	○											

- 本指令的数字键（0～9）功能和 TKEY 指令非常类似，只是硬件输入接线在 TKEY 指令是一个按键占一输入点，而本指令则以 4 个输入点配合 4 个输出点组成多任务扫描输入方式，因 4×4 可有 16 个输入键，除 10 个数字键外，还剩余的 6 个则当作功能键使用（和一般单点输入相同），数字键和功能键的动作是独立而互不影响。
- 当执行控制〝EN〞=1 时，本指令会扫描由 IN 开始的 4 个输入点和由 OT 开始的 4 个输出点组成的矩阵回路中的数字键和功能键两部分，数字键部份请参考 TKEY 指令，而功能键则将 A～F 键的按键状态保持在 KL 所指 16 个继电器的后 6 个（前 10 个存数字键的按键状态），同时 A～F 有任一个键压下，〝FKP〞（FO1）为 1。本指令的 OT 输出点必须为晶体管输出。
- 16 位指令最大可输入 4 位数（9999），D 指令最大则为 8 位数（99999999），但功能键无论 16 或 32 位指令都只有 A～F 6 个。



- 上图程序范例以 X0～X3 和 Y0～Y3 组成多任务按键输入，可以输入 8 位数的数值而将结果存放在 R1R0 中，功能键的输入状态则存放在 M10（A）～M15（F）中。



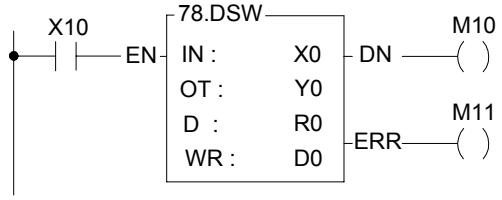
FUN78 D DSW	指拨开关输入 （DIGITAL SWITCH）	FUN78 D DSW

阶梯图符号

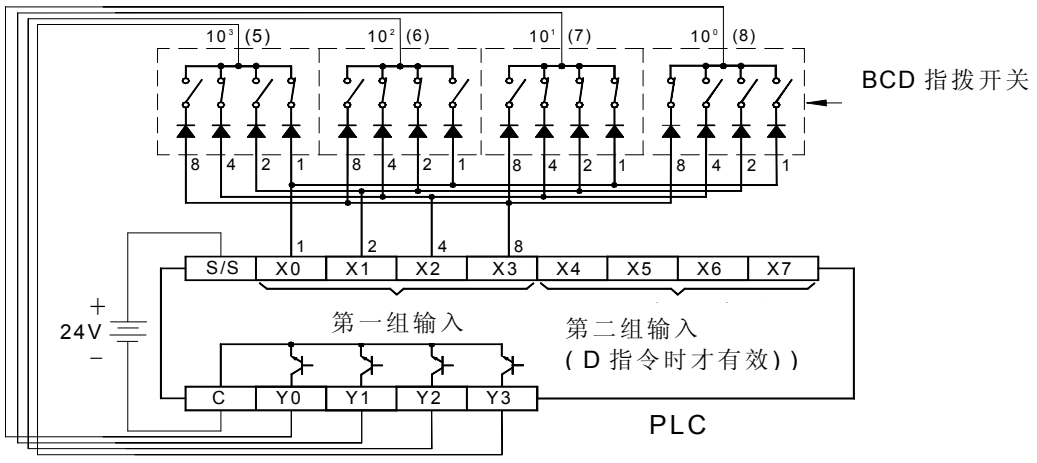
IN ：开关输入点（4 点，D 指令为 8 点）
OT：多任务扫描输出点（4 点）
D ：存放读值的缓存器号码
WR：工作缓存器，其它地方不可重复使用
D 可结合 V、Z、P0~P9 作间接寻址应用

操作数 \ 范围	X	Y	WY	WM	WS	TMR	CTR	HR	OR	SR	ROR	DR	XR
	X0 | X240	Y0 | Y240	WY0 | WY240	WM0 | WM1896	WS0 | WS984	T0 | T255	C0 | C255	R0 | R3839	R3904 | R3967	R3968 | R4167	R5000 | R8071	D0 | D4095	V、Z P0~P9
IN	○												
OT		○											
D			○	○	○	○	○	○	○	○*	○*	○	○

- 当输入控制〝EN〞=1 时，本指令会以 IN 开始的 4 个输入点（IN0～IN3）当作一个位数（Nibble），从低（个）位数开始分四次扫描读取一组 4 个位数的 BCD 数值（0000～9999）再将它存入 D 中，如果为 32 位（D 指令）则一次扫描同时读取两组的位数（即 IN0～IN3 和 IN4～IN7），而将由 IN4～IN7 读到的那组 4 个位数值存入 D+1 缓存器中，扫描的顺序是将 OT0～OT3 位按照顺序设为 1，而分别读到 10^0（个）、10^1（十）、10^2（百）、10^3（千）4 位数。只要〝EN〞为 1，则 PLC 会循环不停的扫描读取，每一个循环（10^0～10^3 4 个位数读取完毕）结束，读取结束旗号〝DN〞会设为 1，但只维持一个扫描时间 t。若有任一个数读值非 0～9（BCD），则读值错误〝ERR〞设为 1，该组数值设为 0000。
- 本指令只能使用一次，且其输出点必须为晶体管输出。

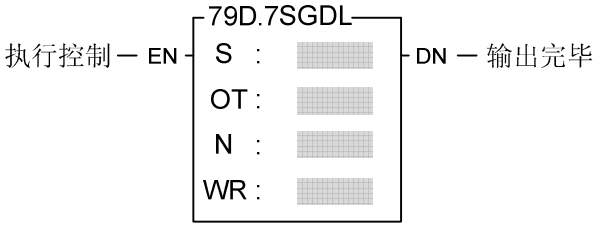


- 本范例当 X10 为 1 则指拨开关的数字量（本例为 5678）值会被读取存入 R0 中。
- 各位数同值的 Bit（8，4，2，1）要并联在一起且需串联二极管，如下图所示。（市场上销售的指拨开关通常已串加二极管）
- D 指令时再加装一组同样的指拨开关到 X4～X7 即可（Y0～Y3 共享）。



FUN79 D 7SGDL	7 段显示器扫描输出 （7 SEGMENT OUTPUT WITH LATCH）	FUN79 D 7SGDL

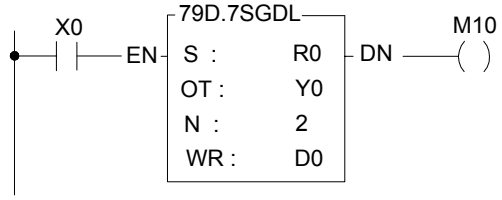
阶梯图符号



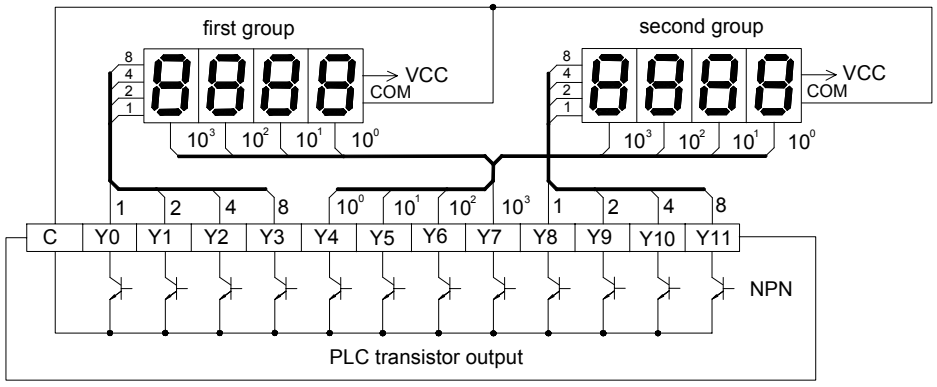
S ：显示数据（BCD）存放的缓存器号码
OT：扫描输出点起头号码
N ：指定信号输出和闩锁信号的极性
WR：工作缓存器，其它地方不可重复使用
S 可结合 V、Z、P0~P9 作间接寻址应用

操作数＼范围	Y	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR
	Y0 | Y240	WX0 | WX240	WY0 | WY240	WM0 | WM1896	WS0 | WS984	T0 | T255	C0 | C255	R0 | R3839	R3840 | R3903	R3904 | R3967	R3968 | R4167	R5000 | R8071	D0 | D4095	16 或 32 位正负数	V、Z P0~P9
S			○	○	○	○	○	○	○	○	○	○	○	○	○
OT	○														
N														0～3	

- 当执行控制〝EN〞=1 时，将缓存器 S 的 4 个位数（Nibble）即位数 0～位数 3 按照顺序分四次送到 OT0～OT3 的 4 个输出点，同时每送出一位数，即送出该位数的闩锁信号，（OT4 对应到位数 0，OT5 对应到位数 1，……），以便将这些送出的位数值载入并闩锁在 7 段显示器内。
- D 指令时则将 S 缓存器的位数 0～3 和 S+1 缓存器的位数 0～3，同时分别送到 OT0～OT3 和 OT8～OT11，因为都是同时送出，故共享闩锁信号。16 位指令没有使用到 OT8～OT11。
- 只要〝EN〞维持 1，PLC 会循环的执行送出动作，在每次送完整组数值（位数 0～3）后，输出结束旗号〝DN〞会变为 1，但只维持一个扫描时间 t。



- 本程序范例当 X0=1 时，R0 的 4 位数字将被送到下图第一组 7 段显示器上，R1 的 4 位数将被送到第 2 组 7 段显示器上

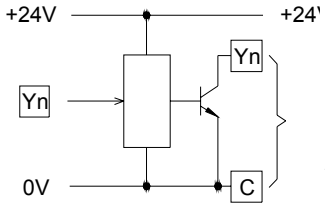


FUN79 D 7SGDL	7 段显示器扫描输出 （7 SEGMENT OUTPUT WITH LATCH）	FUN79 D 7SGDL

● 永宏 PLC 的晶体管输出有负逻辑晶体管输出（NPN 晶体管，当该点状态为 ON 时，该晶体管输出端电压为 Low）及正逻辑晶体管输出（PNP 晶体管，当该点状态为 ON 时，该晶体管输出端电压为 High）两种，其结构如下：

SINK

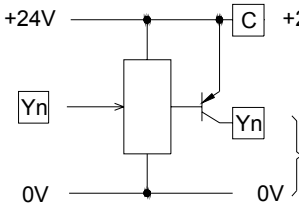
PLC 负逻辑输出（NPN 晶体管）



当 Yn 为〝ON〞时此输出电压为 Low

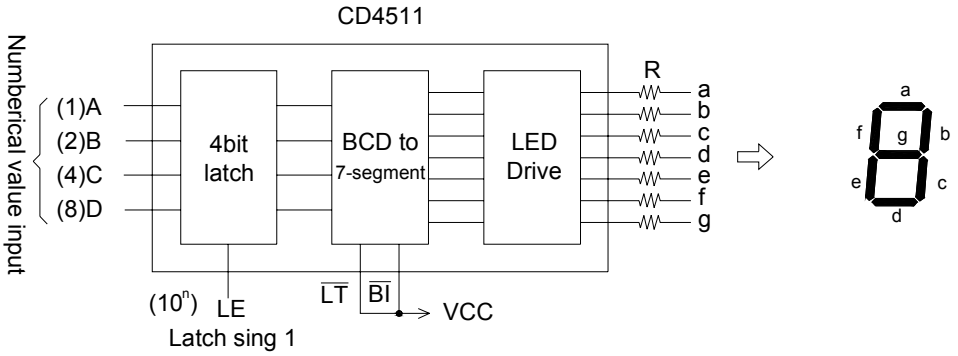
SOURCE

PLC 正逻辑输出（PNP 晶体管）



当 Yn 为〝ON〞时 Yn 输出端电压为 High

● 市场上销售的 7 段显示器的数值输入（8、4、2、1）和闩锁信号也有正、负逻辑之分，例如某一位数值为〝8〞，正逻辑输入应为 1000，但负逻辑输入则为 0111。相同地，正逻辑闩锁在该闩锁信号为 0 时允许显示数值进入闩锁（即载入），而当其为 1 时将当时闩锁内的数值锁住（保持），而负逻辑则反之。下图 CD-4511 七段显示 IC 为正逻辑数值输入及闩锁的一个例子。



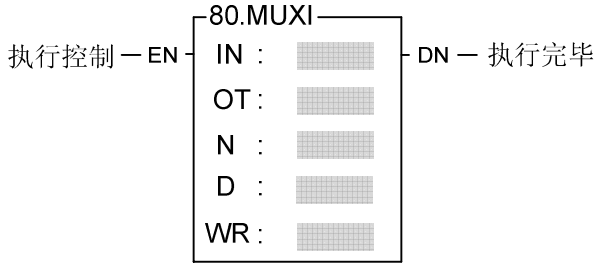
● 因有 PLC 正、负逻辑输出极性和 7 段显示器极性的区分，如果要将它们连结并得到正确显示，必须两者极性要能配合，本指令利用 N 来指定 PLC 晶体管输出的极性，以配合 7 段显示器的极性而实现一致，下表为 PLC 输出和 7 段显示极性组合所必须指定 N 值。

数值输入（8～1）	闩锁信号（10^0～10^3）	正确的 N 值
一致	一致	0
	不一致	1
不一致	一致	2
	不一致	3

● 以上图 7 段显示器 CD4511 为例，其数值输入和 PLC 不一致，而闩锁信号则一致，故 N 值应设定为 2。

FUN80 MUXI	多任务接点输入 （MULTIPLEX INPUT）	FUN80 MUXI

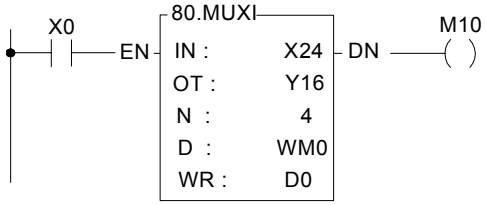
阶梯图符号



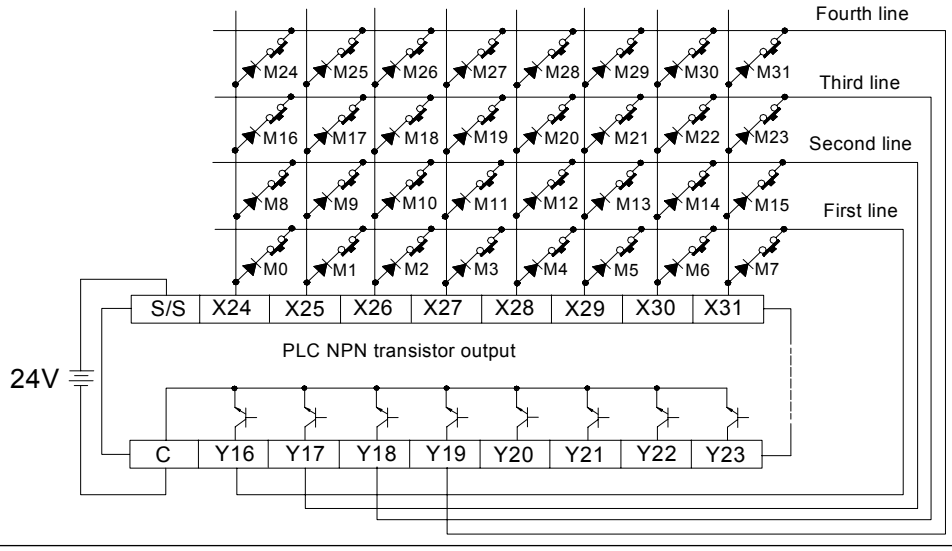
IN：多任务输入点号码
OT：多任务输出点号码
（必须为晶体管输出点）
N ：多任务输入的列数（2～8）
D ：存放结果的缓存器号码
WR：工作缓存器，其它地方不可重复使用
D 可结合 V、Z、P0~P9 作间接寻址应用

操作数 \ 范围	X	Y	WY	WM	WS	TMR	CTR	HR	OR	SR	ROR	DR	K	XR
	X0 | X240	Y0 | Y240	WY0 | WY240	WM0 | WM1896	WS0 | WS984	T0 | T255	C0 | C255	R0 | R3839	R3904 | R3967	R3968 | R4167	R5000 | R8071	D0 | D4095	2 | 8	V，Z | P0~P9
IN	○													
OT		○												
N													○	
D			○	○	○	○	○	○	○	○*	○*	○		○

- 本指令以多任务方式从 IN 所指定的输入点开始的连续 8 个输入点（IN0～IN7），读取 N 列输入状态，而获得 8×N 个输入状态，但却只须用到 8 个输入点和 N 个输出点而已。
- 多任务扫描方式是在 OT 输出点开始的 N 个输出点中，由 OT0 开始设为 1，读取第一列状态，接着把 OT1 设为 1，读取第 2 列状态，……直到读完 N 列为止。再将所读取到的 8×N 个状态存入从 D 开始的缓存器中，并将执行结束旗号〝DN〞设为 1（但只维持一个扫描时间）。
- 本指令每一次扫描抓取一列 8 个输入点状态，故 N 列要 N 个扫描时间才能抓完。

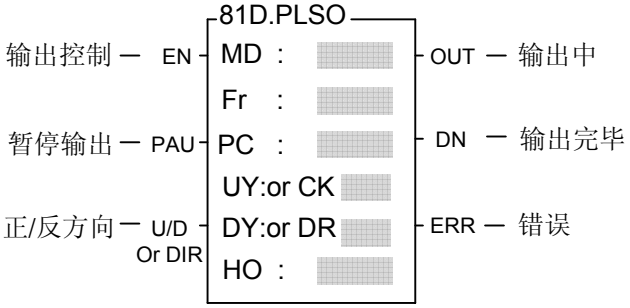


- 本范例抓取 4 列×8 点输入共 32 点状态，并将它存放在 DWM0（M0～M31）的 32 位缓存器中。



FUN81 D PLSO	脉冲输出指令 （PULSE OUTPUT）	FUN81 D PLSO

阶梯图符号



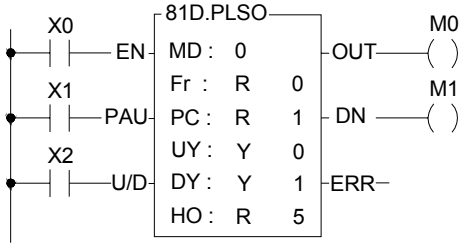
MD：运转模式选择
Fr ：脉冲频率
PC：输出脉冲数
UY：正转脉冲的输出点（MD=0）
DY：反转脉冲的输出点（MD=0）
HO：已送出脉冲的缓存器（可不指定）
CK：脉冲输出点（MD=1）
DR：正／反转输出点（MD=1）
DIR：1，正转；0，反转

操作数＼范围	Y	WX	WY	WM	WS	TMR	CTR	HR	OR	SR	ROR	DR	K
	主机上的 Yn	WX0 | WX240	WY0 | WY240	WM0 | WM1896	WS0 | WS984	T0 | T255	C0 | C255	R0 | R3839	R3904 | R3967	R3968 | R4167	R5000 | R8071	D0 | D4095	16 或 32 位正数
MD													0～1
Fr		○	○	○	○	○	○	○	○	○	○	○	8～2000
PC		○	○	○	○	○	○	○	○	○	○	○	○
UY，CK	○												
DY，DR	○												
HO			○	○	○	○	○	○	○	○*	○*	○	

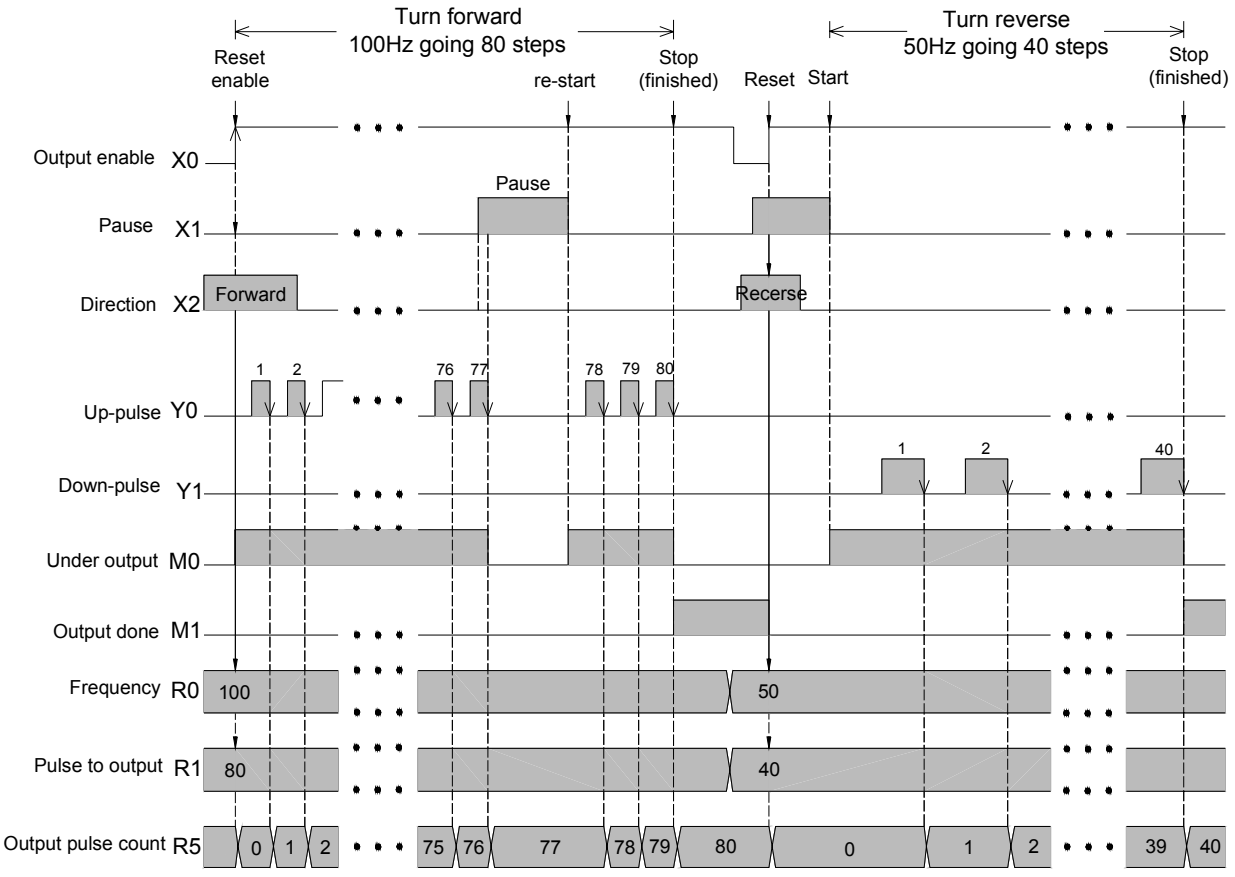
- 当 MD=0 时，本指令如下述方式作脉冲输出控制：
- 当输出控制〝EN〞由 0→1 瞬间，首先执行重置（RESET）动作，也就是将输出旗号〝OUT〞和〝DN〞以及已送出脉冲缓存器 HO 都清为 0，并抓取脉冲频率 Fr 及脉冲数 PC 的值，再读取正反方向〝U/D〞的状态来决定正、反转方向。完成重置工作后本指令已完成输出准备，紧接着检查暂停输出〝PAU〞的状态，如果它为 1（暂停输出）则不作任何动作，若为 0 则开始以脉冲频率 Fr 所指定的频率自 UY（U/D=1 时）或 DY（U/D= 0 时）输出点送出 ON/OFF 脉宽各为 50％的方形脉冲，每送出一个脉冲即将 HO 缓存器加 1，一直到 HO 缓存器内的脉冲数等于或大于 PC 缓存器的脉冲数才停止脉冲的送出，并将输出结束旗号〝DN〞设为 1。任何时刻只要本指令是在脉冲输出当中则输出中旗号〝OUT〞即设为 1，否则为 0。
- 在开始输出脉冲后输出控制〝EN〞仍应保持为 1，如果它变为 0，则立刻停止脉冲的送出（输出点变为 OFF），〝OUT〞旗号回到 0，其它状态或数据则保持不变，但当其〝EN〞再度由 0 回到 1 时，却会造成重置动作而当作一个新的开始，整个程序将重新来过。
- 如果要暂停脉冲输出而又不被整个重新来执行，则可利用暂停输出〝PAU〞来暂停脉冲的输出。当〝PAU〞=1 时本指令会暂停脉冲的送出（输出点为 OFF，〝OUT〞旗号回到 0，而其它状态或数据都保持不变），等〝PAU〞由 1 变回 0 后，本指令会回到暂停前的状态并由此开始继续脉冲的输出动作。
- 在脉冲输出当中，本指令每次扫描到时仍会去抓取脉冲频率 Fr 及脉冲数 PC 的数值，因此只要脉冲尚未送完，都可更改脉冲频率或输出脉冲数。但正反转方向〝U/D〞的状态只有在重置动作（〝EN〞由 0→1）时抓取一次便一直保持到送完或下一次重置为止，也就是除了重置瞬间外，〝U/D〞的变化对本指令无任何影响。
- 本指令主要在推动步进电机，UY（正转）和 DY（反转）两种方向的脉冲来方便控制步进电机的正反转动作。如果只需要单方向运转，可单独指定 UY 或 DY 的其中之一（可省下一个输出点），另一个空白不指定。此时本指令将不理会正反方向〝U/D〞的输入状态，输出脉冲将固定送往用户所指定的那个输出点。

FUN81 D PLSO	脉冲输出指令 （PULSE OUTPUT）	FUN81 D PLSO

- 当 MD=1 时，控制方式为正／反转（DIR=1，正转；DIR=0，反转）结合脉冲信号（CK）输出方式控制。控制说明与上述相同。
- 本指令只能使用一次，且 UY（CK）和 DY（DR）必须为 PLC 主机上的晶体管输出点。
- 本指令的输出脉冲数 PC 在 16 位指令时的有效范围为 0～32767，在 32 位（指令）时则为 0～2147483647。如果 PC 值=0 时认为是无限脉冲数，本指令将无限制地送出脉冲而 HO 值和〝DN〞旗号则永远为 0。而脉冲频率 Fr 的有效范围则为 8～2000。无论 PC 或 Fr，如果其值超出上述范围即为错误，本指令将不执行且将错误旗号〝ERR〞设为 1。

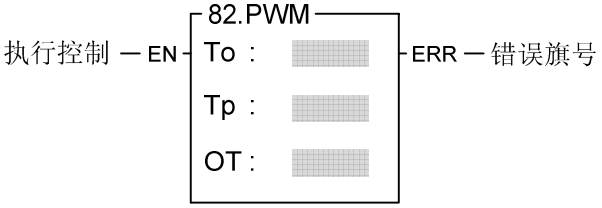


- 本范例控制步进电机先以 100Hz 的速度向前（正转）走 80 个脉冲（步），然后再以 50Hz 的速度反方向退后 40 个脉冲。注意正反方向、频率 Fr 及脉冲数 PC 要在重置（〝EN〞由 0→1）前就要先准备好。



FUN82 PWM	脉冲宽度调变 （PULSE WIDTH MODULATION）	FUN82 PWM

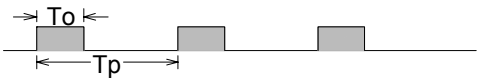
阶梯图符号



To：指定脉冲 ON 的宽度（0～32767mS）
Tp：指定脉冲的周期（1～32676mS）
OT：指定调变脉冲的输出点

操作数＼范围	Y	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K
	主机上之 Yn	WX0 | WX240	WY0 | WY240	WM0 | WM1896	WS0 | WS984	T0 | T255	C0 | C255	R0 | R3839	R3840 | R3903	R3904 | R3967	R3968 | R4167	R5000 | R8071	D0 | D4095	0 | 32767
To		○	○	○	○	○	○	○	○	○	○	○	○	○
Tp		○	○	○	○	○	○	○	○	○	○	○	○	○
OT	○													

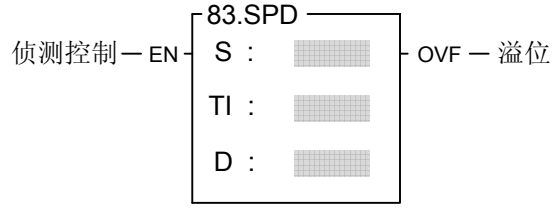
- 当执行控制〝EN〞=1 时，将周期为 Tp，〝ON 脉宽为 To 的方波送到输出点 OT 去，而 OT 必须为主机上的晶体管输出点。当〝EN〞=0 时，输出点为不动作（OFF）。



- To 和 Tp 的单位都为 mS，分辨率为 1mS，To 的数值最小可为 0（此时输出点 OT 永远 OFF）最大可等于 Tp（此时输出点 OT 永远 ON），若 To＞Tp 则为错误，本指令即不执行，且错误旗号〝ERR〞变成 1。
- 本指令只能使用一次。

FUN83 SPD	速度检测 (SPEED DETECTION)	FUN83 SPD

阶梯图符号

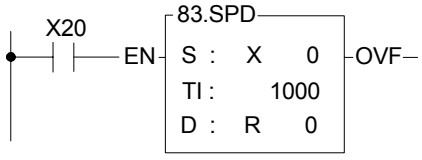


S ：要检测速度的脉冲输入点
TI：检测的取样时间（单位为 mS）
D ：存放结果的缓存器号码

操作数＼范围	X	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K
	X0 ｜ X7	WX0 ｜ WX240	WY0 ｜ WY240	WM0 ｜ WM1896	WS0 ｜ WS984	T0 ｜ T255	C0 ｜ C255	R0 ｜ R3839	R3840 ｜ R3903	R3904 ｜ R3967	R3968 ｜ R4167	R5000 ｜ R8071	D0 ｜ D4095	1 ｜ 32767
S	○													
TI		○	○	○	○	○	○	○	○	○	○	○	○	○
D			○	○	○	○	○	○	○	○	○*	○*	○	

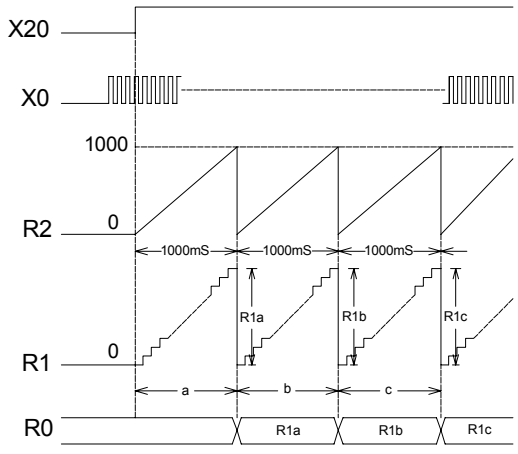
- 本指令是利用 PLC 主机上的 X0～X7，8 个高速输入点的中断功能，而在一个特定的取样时间 TI 内计算某一个输入点的输入脉冲数，而间接求出接在该输入点的运转装置（如电机）的转速。
- 应用本指令时，齿轮盘齿数必须大于 60 齿；且输入频率总和必须小于 5KHz。
- 本指令的结果缓存器 D 总共使用了由 D 开始的连续 3 个 16 位缓存器（D0～D2），除 D0 为存放计数结果外，D1 和 D2 用以存放计数经过值和累计取样时间。
- 当执行检测〝EN〞=1 时，开始计算 S 输入点的脉冲数，并先将的暂存在缓存器 D1 中，同时启动取样定时器（D2），等到 D2 值等于检测取样时间 TI 后停止计数，并将最后的计数值 D1 存入缓存器 D0 中，然后再重新开始另一次计数，时间到后再将新得到的计数值存入（盖过）到缓存器 D0 中，这样周而复始地取样计数，直到〝EN〞=0 方才停止。
- 因 D0 只有 16 位最多只能计数到 32767，如果取样时间过长，而且输入脉冲过快将导致计数值超过 32767，则溢位旗号发生，计数动作停止。
- 因取样时间 TI 已知，若运转装置每转一圈产生 n 个脉冲，则可利用下式求出其转速。

$$N=\frac{(D0)\times 60}{n\times TI}\times 10^3 \quad (rpm)$$



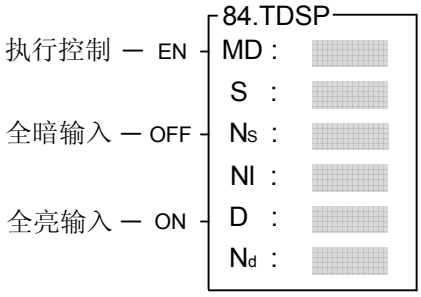
- 如上图范例，若运转装置每转一圈产生 60 个脉冲（n=60），而 R0 读值为 200，则该回转装置每分钟转速 N 如下：

$$N=\frac{(200)\times 60}{60\times 1000}\times 10^3 = 200\,rpm$$



FUN84 P TDSP	文、数字显示字型转换便利指令 (功能简述)	FUN84 P TDSP

阶梯图符号



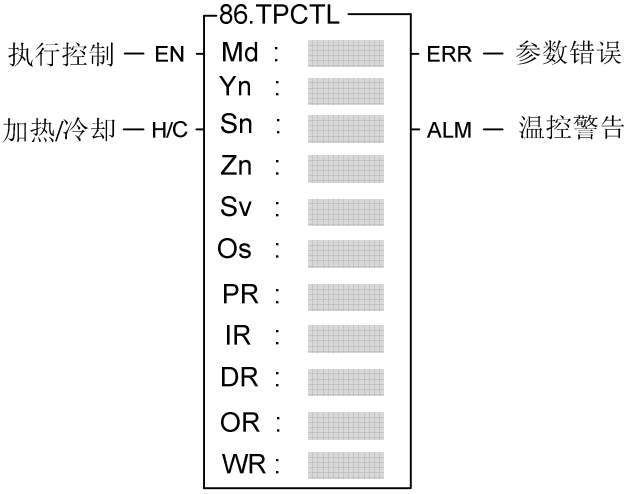
Md：模式选择
S ：要做文、数字显示的起始缓存器号码
Ns：要做文、数字显示的起始字符
Nl ：要做文、数字显示的字符长度
D ：存放显示字型的起始缓存器号码
S 可结合 V、Z、P0～P9 作间接寻址应用

操作数＼范围	HR	OR	ROR	DR	K	XR
	R0 | R3839	R3904 | R3967	R5000 | R8071	D0 | D4095	正数 16/32 位	V、Z P0~P9
Md					0～1	
S	○	○	○	○	○	○
Ns	○	○	○	○	○	
Nl	○	○	○	○	○	
D	○	○	○*	○		

- 本指令为 7 段显示器模块（FBs-7SG-X ）配合米字型显示器的专用输出指令，本指令采用填表方式用来指定要显示的内容地址、显示的字数、及是否零前导等指示，可大幅缩减程序设计时间及简化程序。
- 本指令的详细说明及范例请参考第 16 章〝FBs-7SG-X 七段显示器模块〞的叙述。

FUN86 TPCTL	PID 温控便利指令 (PID TEMPERATURE CONTROL INSTRUCTION)	FUN86 TPCTL

阶梯图符号



操作数＼范围	Y	HR	ROR	DR	K
	Y0 | Y255	R0 | R3839	R5000 | R8071	D0 | D4095	
Md					0～1
Yn	○				
Sn					0～31
Zn					1～32
Sv		○	○*	○	
Os		○	○*	○	
PR		○	○*	○	
IR		○	○*	○	
DR		○	○*	○	
OR		○	○*	○	
WR		○	○*	○	

Md：PID 运算模式选择
=0，改良型最小超越法
=1，泛用 PID 法则

Yn：ON/OFF 温控输出起始号码，共占用 Zn 点

Sn：本指令从第几点温度开始执行 PID 温控，
Sn=0～31

Zn：本指令所控制之 PID 温控点数；
1 ≤ Zn ≤ 32 且 1 ≤ Sn+Zn ≤ 32

Sv：温度设定值起始缓存器号码，共占用 Zn 个缓存器

Os：温度偏差值起始缓存器号码，共占用 Zn 个缓存器

PR：增益设定值起始缓存器号码，共占用 Zn 个缓存器

IR：积分时间常数设定值起始缓存器号码，共占用 Zn 个缓存器

DR：微分时间常数设定值起始缓存器号码，共占用 Zn 个缓存器

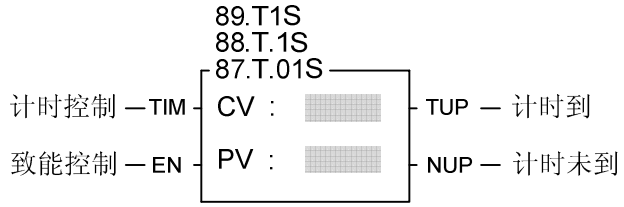
OR：温控数值输出起始缓存器号码，共占用 Zn 个缓存器

WR：本指令所需使用的工作缓存器起始号码，共占用 9 个缓存器，其它地方不可重复使用

- PID 温控（FUN86）是利用温度模块配合温度规划表格将外界目前的温度值测量进来当作程控变量（Process Variable，简称 PV），并将用户所设定的温度设定值（Set Point，简称 SP）与程控变量经由软件 PID 数学式运算后，得到适宜的输出控制值以控制温度在用户所期望的温度范围内。
- 将 PID 运算后的数值结果转换为时间比例 ON/OFF（PWM）输出，经由晶体管式接点输出控制 SSR 所串接的加热或冷却回路，即可得到相当精确且价廉的控制结果。
- 也可将 PID 运算后的数值结果经由 D/A 模拟量输出模块，控制 SCR 导通角度或比例阀来作温度精确控制。
- 详细的功能与说明及其用法与范例请参考〝FBs-PLC 温度测量及温度 PID 控制〞章节详述。

FUN87 T.01S FUN88 T.1S FUN89 T1S	积算型定时器（0.01 秒，0.1 秒，1 秒） （ACCUMULATIVE TIMER）	FUN87 T.01S FUN88 T.1S FUN89 T1S

阶梯图符号



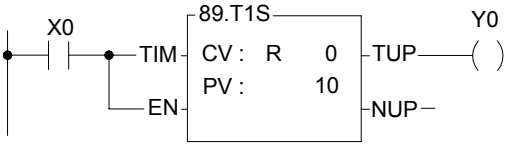
CV：存放计时时间（即现在值）的缓存器号码

PV：定时器的设定或存放设定值的缓存器号码

操作数 \ 范围	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K
	WX0 | WX240	WY0 | WY240	WM0 | WM1896	WS0 | WS984	T0 | T255	C0 | C199	R0 | R3839	R3840 | R3903	R3904 | R3967	R3968 | R4167	R5000 | R8071	D0 | D4095	0 | 32767
CV		○	○	○	○	○	○	○	○	○*	○*	○	
PV	○	○	○	○	○	○	○	○	○	○	○	○	○

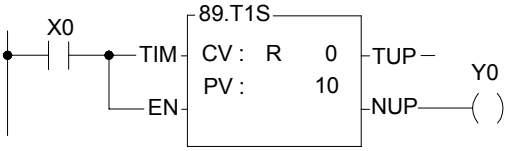
● 本指令的工作原理和一般定时器（T0～T255）相同，只是一般定时器只有一个计时控制〝EN〞输入，在其输入为 1 时计时，为 0 时则清除，每次输入变动都会重新计时，无法累计。而本指令的计时需在致能控制〝EN〞=1 的条件下才允许，此时其计时控制〝TIM〞为 1 时和一般定时器一样，但为 0 时，则不清除而保持现值。若需清除则使致能控制〝EN〞变为 0 即可，如果不清除，等计时控制〝TIM〞再度为 1 时，定时器将从上次暂停时的计时值继续累加。此外本指令还有计时到〝TUP〞（计时到时为 1，平时为 0），及计时未到〝NUP〞（平时为 1，计时到为 0）两个输出，用户可利用输入和输出组合出各种不同功能需求的定时器，如以下范例：

● ON DELAY ENERGIZING（ON 延时供电）定时器：



- 是指该定时器输出（本例为 Y0）平常不供电，而在该定时器输入控制（本例为 X0）动作（ON）后延时 10 秒后，输出 Y0 才供应电能（ON）。

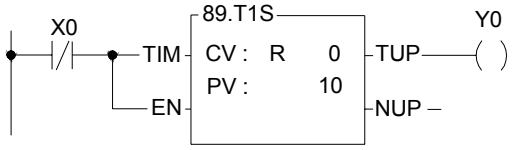
● ON DELAY DE-ENERGIZING（ON 延迟断电）定时器：



- 是指该定时器输出 Y0 平常就在供电状态下，而在该定时器的输入控制 X0 ON 后延时 10 秒后，输出才断电（OFF）。

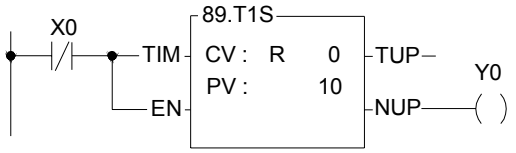
FUN87 T.01S FUN88 T.1S FUN89 T1S	积算型定时器（0.01 秒，0.1 秒，1 秒） （ACCUMULATIVE TIMER）	FUN87 T.01S FUN88 T.1S FUN89 T1S

● OFF DELAY ENERGIZING（OFF 延时供电）定时器：



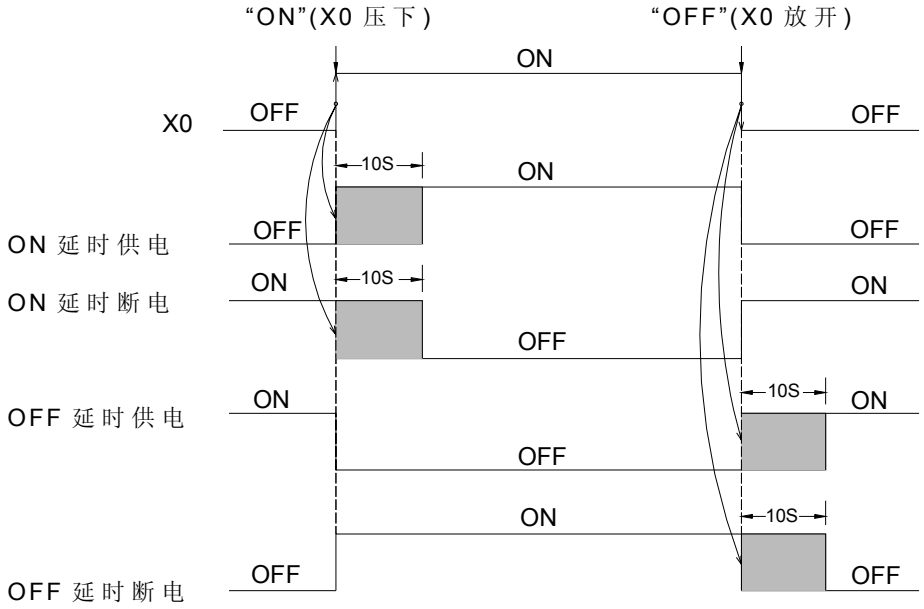
- 是指该定时器的输出 Y0 平常在断电状态下，在该定时器的输入控制 X0 OFF 后延时 10 秒后，输出 Y0 才供电（ON）。

● OFF DELAY DE-ENERGIZING（OFF 延时断电）定时器：



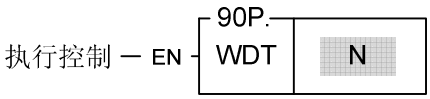
- 是指该定时器的输出 Y0 平常在供电状态下，在该定时器的计时控制 X0 OFF 后延时 10 秒后输出 Y0 才断电（OFF）。

● 下图为以上四种定时器的输入与输出的对应结果。



FUN90 P WDT	监控定时器（WATCHDOG TIMER）设定时间设定	FUN90 P WDT

阶梯图符号

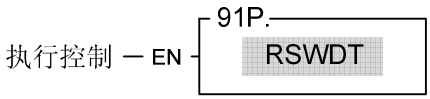


N：监控定时器的设定时间。
其值只能为 50、60、70……120，单位为 10mS，即设定的时间范围为(50～120)×10mS，即 0.5 秒～1.2 秒。

- 当执行控制〝EN〞=1 或〝EN↑〞（P 指令）由 0→1 时，将监控定时器的设定时间改为 Nx10ms。一旦设定后，Watchdog Timer (WDT) 即以此为计时时间，如果扫描时间超过该设定时间，PLC 将会停机不执行。
- WDT 设定时间主要是以系统应用上的安全考虑而特别设计的，例如 PLC CPU 如果突然损坏，无法执行程序或 I/O 更新时，经过 WDT 所设定的时间后，WDT 会自动从硬件上将 I/O 完全关闭以确保安全。在某些应用上如果扫描时间太长也可能造成某些安全上或不符控制要求的问题，也可利用本指令设定所要求的扫描时间极限值，超过设定值，PLC 会立刻停机，以确保安全。
- 设定值一旦设定后，即永久保存，无需每次扫描都设定一次，因此本指令实用上应使用 P 指令。
- WDT 时间默认为 0.25 秒。
- WDT 的工作原理请参考 FUN91（RSWDT）指令。

FUN91 P RSWDT	清除监控定时器 （RESET WATCHDOG TIMER）	FUN91 P RSWDT

阶梯图符号



本指令无操作数

- 当执行控制〝EN〞=1 或〝EN↑〞（P 指令）由 0→1 时，将 WDT 定时器清除（也就是使 WDT 重新由 0 开始计时）。

- Watchdog Timer 的功能已在 FUN90（WDT 指令）中叙述，其原理如下：
  监控定时器一般都为硬件单击（One-Shot）定时器（不能用软件作，否则 CPU 若当机，该定时器便失效，当然谈不上能保护了），所谓单击，意思就是只要触发定时器一下，该定时器的计时值便立刻清为 0 再重新计时。如果在 WDT 开始计时后都未去触发它，则 WDT 计时时间继续累增至设定值 N 后 WDT 即动作，而将 PLC 停机。若您每次在 WDT 计时时间 N 尚未到达前就触发 WDT 一次，则 WDT 永远都不会发生，PLC 即利用此原理来确保系统安全，因为 PLC 一般会在程序扫描和 I/O 更新后，进入系统服务（Housekeeping）时触发 WDT 一次，如果系统正常且扫描时间未超出 WD 的设定时间 N,就一定来得及在 WDT 未动作前清掉 WDT 而使它不动作，但如果 CPU 损坏而无法触发 WDT 或扫描时间过长导致来不及在 N 时间内触发 WDT，WDT 就会动作而关掉 PLC。

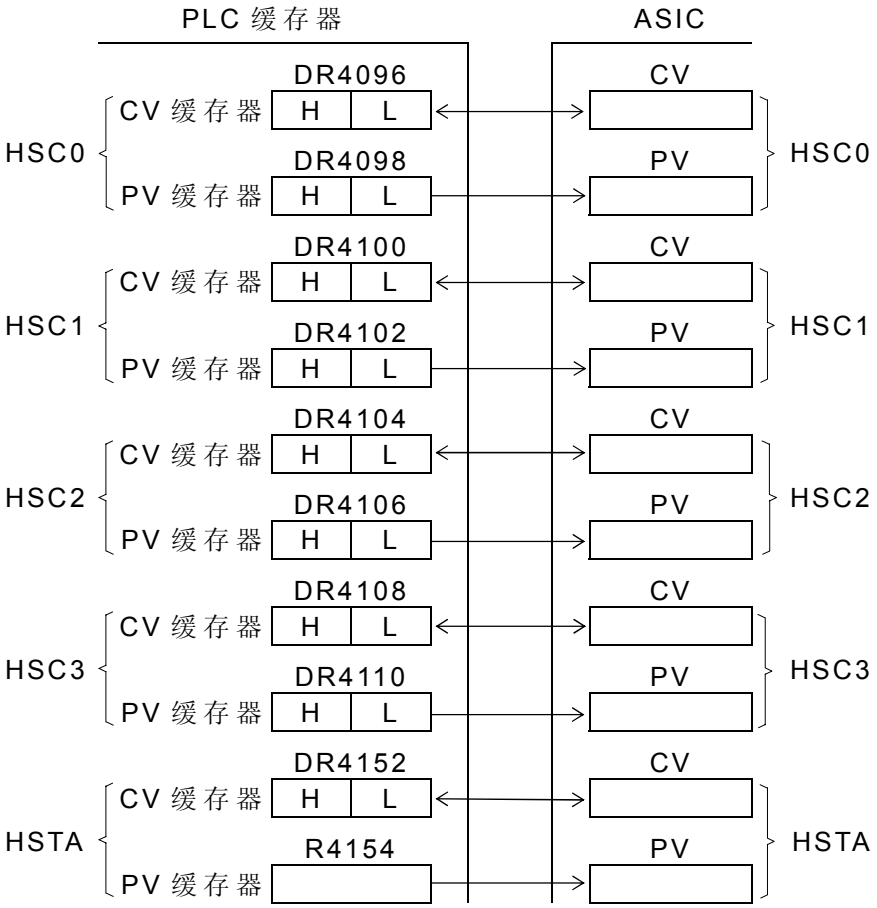
- 在某些应用下，已经设定好了 WDT 时间（FUN90），而程序在某些情况下扫描时间可能会暂时超出 WDT 的设定时间，这情况是用户所能预期且允许的，当然不希望因此而 PLC 停机，此时用户可用本指令触发 WDT 一下即可避免 WDT 发生，这就是本指令的主要目的。

FUN92 P HSCTR	硬件高速计数器现在值（CV）读取	FUN92 P HSCTR

阶梯图符号

CN：硬件高速计数器号码
0：HSC0 或 HST0
1：HSC1 或 HST1
2：HSC2 或 HST2
3：HSC3 或 HST3
4：HSTA

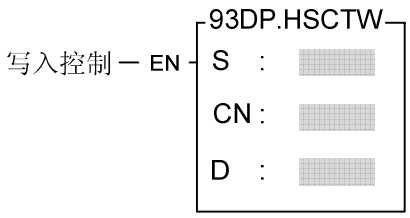
- FBs-PLC 的硬件高速计数器 HSC0～HSC3 是四组 32 位可作双相或单相上下数的高速计数器，是由 ASIC 内部硬件电路所组成，能独立进行计数、比较，发出中断，不会占用 CPU 的时间。（软件高速计数器 HSC4～HSC7 是用中断方式由 CPU 来处理，因此当使用个数多或计数频率高时，整个 PLC 的性能（扫描速度）将严重恶化）。因为 HSC0～HSC3 的现在值 CV 是在 ASIC 内部的硬件电路中，用户的控制（梯形图）程序无法直接到 ASIC 去抓取，因此需要利用本指令将硬件 HSC 的 CV 值读出并放到控制程序能抓取到的缓存器去，以下为 ASIC 中 CV、PV 和 PLC 内部相对应的 CV、PV 缓存器的安排。



- 当读取控制〝EN〞=1 或〝EN↑〞（P 指令）由 0→1 时，将 ASIC 中，CN 所指定的 HSC 的 CV 值读到 PLC 内部该 HSC 所对应的 CV 缓存器去。（即将 HSC0 的 CV 读到 DR4096 或将 HSC1 的 CV 读到 DR4100）。
- 虽然 ASIC 中的 PV 在 PLC 内部也有 PV 缓存器与它对应，但却不需读取，因 ASIC 中的 PV 值是来自 PLC 内部的 PV 缓存器。
- HSTA 是以 0.1mS 为时基的定时器，CV 的内容，代表经过多少个 0.1mS 时间。
- 详细的应用请参考第 10 章〝FBs-PLC 的高速计数器与高速定时器〞。

FUN93 P HSCTW	硬件高速计数器 CV 或 PV 值写入	FUN93 P HSCTW

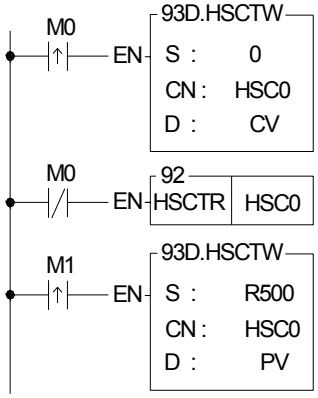
阶梯图符号



CN：硬件高速计数器号码
- 0：HSC0 或 HST1
- 1：HSC1 或 HST2
- 2：HSC2 或 HST3
- 3：HSC3 或 HST4
- 4：HSTA

D ：写入对象（0：表示 CV，1：表示 PV）

- 请先参考 FUN92 有关 ASIC 中 HSC0～HSC3 与 HSTA 的 CV 或 PV 值和 PLC 内部相对应的 CV 缓存器和 PV 缓存器的关系。
- 当写入控制〝EN〞=1 或〝EN↑〞（P 指令）由 0→1 时，将 PLC 内部 CN 所指定的那个高速计数器的 CV 缓存器或 PV 缓存器的内容值写到 ASIC 内部相对应 HSC 的 CV 或 PV 去。
- 一般应用常需写入 PV，也就是将预先设定的设定值写到 ASIC 中的 PV 去，当计数值到达所要求的设定值时，该计数器立即发出中断，通过中断服务程序可作各种精密的计数或定位控制。
- FBs-PLC 在电源断电时会自动将当时 ASIC 内部 HSC0～HSC3 的现在值缓存器 CV 的值读出再将它写入 PLC 内部 HSC0～HSC3 的 CV 缓存器（具有断电保持功能）中，而在 PLC 恢复电源时则会反向地将 PLC 内部的 CV 缓存器写回 ASIC 内部的 CV 缓存器，因此每次 PLC 断电再恢复电源，ASIC 内部 HSC0～HSC3 的 CV 缓存器内容值将会自动回复到上次断电前的数值，但如果控制应用在复电时需清为 0 或从某一个特定值开始计数，就必须利用本指令来作 ASIC 内部 HSC 的 CV 值写入。
- HSTA 写入不为 0 的 PV 值，代表每 PV×0.1ms 会定时发出中断；HSTAI 中断子程序即为定时中断处理程序。
- 详细的应用请参考第 10 章〝FBs-PLC 的高速计数器与高速定时器〞。

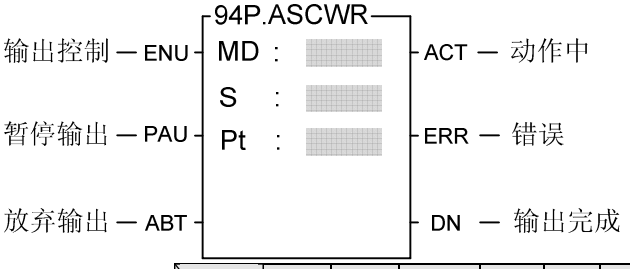


- 左图程序，M0 由 0→1 时，将 HSC0 目前值清除为 0，并通过 FUN93 写入硬件 ASIC 中
- M0 为 0 时，随时读出目前的计数值
- M1 由 0→1 时，将 DR500 的计数设定值搬到 DR4098，并通过 FUN93 写入硬件 ASIC 中
- 当计数值等于 DR500 中的设定值时，立即执行 HSC0I 中断处理子程序

列表打印指令

FUN94 ASCWR	ASCII 档案数据输出 （ASCII FILE WRITE）	FUN94 ASCWR

阶梯图符号



MD：输出模式选择
=0，外围为 Printer 时，利用 ASCWR 将 ASCII Editer 所编的信息经由通讯端口 1 传送给 Printer。
=1，保留特定用途。
S ：档案数据的起始缓存器号码
Pt ：指令运作起始缓存器共占用 8 个缓存器，其它地方不可重复使用

操作数 \ 范围	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K
	WX0 | WX240	WY0 | WY240	WM0 | WM1896	WS0 | WS984	T0 | T255	C0 | C255	R0 | R3839	R3840 | R3903	R3904 | R3967	R3967 | R4167	R5000 | R8071	D0 | D4095	0 | 1
MD													○
S	○	○	○	○	○	○	○	○	○	○	○	○	
Pt		○	○	○	○	○	○		○	○*	○*	○	

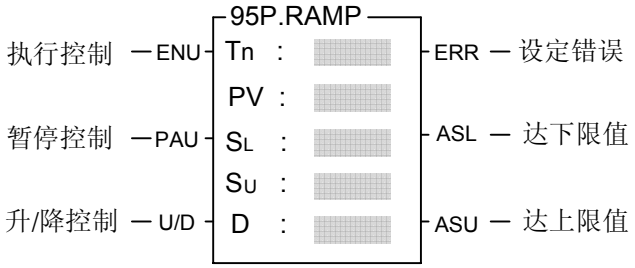
- 当 MD＝0 时，输出控制〝EN↑〞由 0→1 时，将 S 开始的 ASCII 档案数据送到通讯端口 1（Port1）去，直到送完整个档案（遇到档案结尾字符 END）为止。
- S 档案数据可通过梯形图大师（WINPROLADDER）软件包上的 ASCII 编辑器来编辑（请参考第 15 章〝ASCII 功能应用〞的说明），用户也可自行按照接在 PortI 的 ASCII 外围的特点，自己编辑所希望的报表或画面的档案数据，但档案数据必须符合本指令规定的格式（详述于第 15 章），否则本指令将中止输出动作，并将错误旗号〝ERR〞设为 1。如果整个档案都正确且成功送出则输出完成〝DN〞设为 1。
- 本指令的输入为正缘触发，一旦〝EN↑〞由 0→1 时，本指令开始执行后，就一直要等到整个档案送完才算执行完成，这期间动作中旗号〝ACT〞都将一直维持 1，除非遇到暂停输出、错误或放弃输出才会变回 0。
- 本指令可重复使用，但任何一个时间内只能有一个被执行（作输出），用户必须自行控制其执行的先后顺序。
- 如果本指令执行中，暂停输出〝PAU〞若变为 1，则本指令暂停档案数据的输出，等待暂停输出〝PAU〞变为 0，本指令才又继续先前档案数据的输出。
- 如果本指令执行中，放弃输出〝ABT〞若变为 1，则本指令中断该输出中档案的执行，此时可接受下一个指令的执行。
- 使用 FUN94 (ASCWR)指令，必须将 CPU 主机上的 DIP 开关设定成 SW−1 OFF & SW−2 ON。
- 详细的应用请参考第 14 章〝ASCII 档案功能的应用〞饿说明
- 接口处理信号：

M1927：此信号由 CPU 产生，适用于 ASCWR MD：0
：ON，代表 Printer 的 RTS (接至 PLC 的 CTS)〝False〞，也就是 Printer Not Ready 或异常。
：OFF，代表 Printer 的 RTS〝True〞，Printer Ready。
※利用 M1927 结合定时器可计时检测 Printer 是否异常。

R4158：通讯参数设定。

FUN95 RAMP	D／A 输出缓升／缓降指令	FUN95 RAMP

阶梯图符号



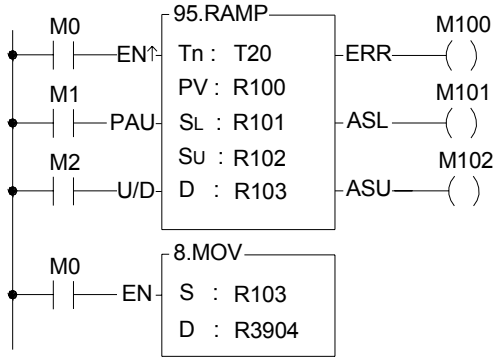
Tn：缓升／缓降定时器号码
PV：缓升／缓降定时器设定值（单位为 0.01 秒）或每 10mS 的增／减量设定值
SL：下限值（缓升初始值或缓降最终值）
SU：上限值（缓降初始值或缓升最终值）
D ：缓升／缓降值存放缓存器
D+1：工作缓存器
SU，SL 配合 AO 模块应用可为正、负值

操作数＼范围	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K
	WX0 ｜ WX240	WY0 ｜ WY240	WM0 ｜ WM1896	WS0 ｜ WS984	T0 ｜ T255	C0 ｜ C255	R0 ｜ R3839	R3840 ｜ R3903	R3904 ｜ R3967	R3968 ｜ R4167	R5000 ｜ R8071	D0 ｜ D4095	正负数 16 位
Tn					○								
PV	○	○	○	○	○	○	○	○	○	○	○	○	○
SL	○	○	○	○	○	○	○	○	○	○	○	○	○
SU	○	○	○	○	○	○	○	○	○	○	○	○	○
D		○	○	○	○	○	○		○	○	○*	○	

- Tn 务必使用时基为 0.01 秒的定时器，而且在程序里不得重复使用。
- 当 M1974=0 时，PV 为缓升／缓降定时器设定值，单位为 10mS（0.01 秒）。
- 当 M1974=1 时，PV 为每 10mS 的增／减量设定值。
- 当输入控制〝EN↑〞由 0→1 时，首先将定时器 Tn 复归为 0；
  如此时〝U/D〞=1，则表示缓升而将 SL 的值载入缓升／缓降值存放缓存器 D，以后每 0.01 秒等比例（SU−SL / PV，M1974=0）或以 PV 为设定值（M1974=1）增加输出量，并存放到缓存器 D，达定时器设定值时（M1974=0）或达上限设定值时（M1974=1），输出值等于 SU，输出〝ASU〞=1（达上限值）；
  如此时〝U/D〞=0，则表示缓降而将 SU 的值载入缓升／缓降值存放缓存器 D，以后每 0.01 秒等比例（SU−SL / PV，M1974=0）或以 PV 为设定值（M1974=1））减少输出量，并存放到缓存器 D，达到定时器设定值时（M1974=0）或达到下限设定值时，输出值等于 SL，输出〝ASL〞=1（达下限值）。
- 缓升／缓降（U/D）的决定是在输入控制〝EN↑〞由 0→1 时，其它时间无效；只要输入控制〝EN↑〞由 0→1 即自动完成一次缓升／缓降控制。
- 如需暂停缓升／缓降动作，则必须使输入控制〝PAU〞=1；当〝PAU〞=0 时，如果缓升／缓降动作未完成，则继续完成未完成的动作。
- SU 的值必须大于 SL，否则缓升／缓降动作不执行，输出〝ERR〞=1。
- 本指令使用缓升／缓降值存放缓存器 D 来存放输出变化值；如使用 AO 模块来做速度控制时，可将缓升／缓降值存放缓存器 D 的值搬到 AO 输出缓存器（R3904～R3967），而使启动／结束的控制较为平稳。
- 本指令除了使用缓升／缓降值存放缓存器 D 来存放输出变化值外，还使用缓存器 D+1 来作为工作缓存器，所以程序里不得再使用 D+1 这个缓存器。

FUN95 RAMP	D / A 输出缓升 / 缓降指令	FUN95 RAMP

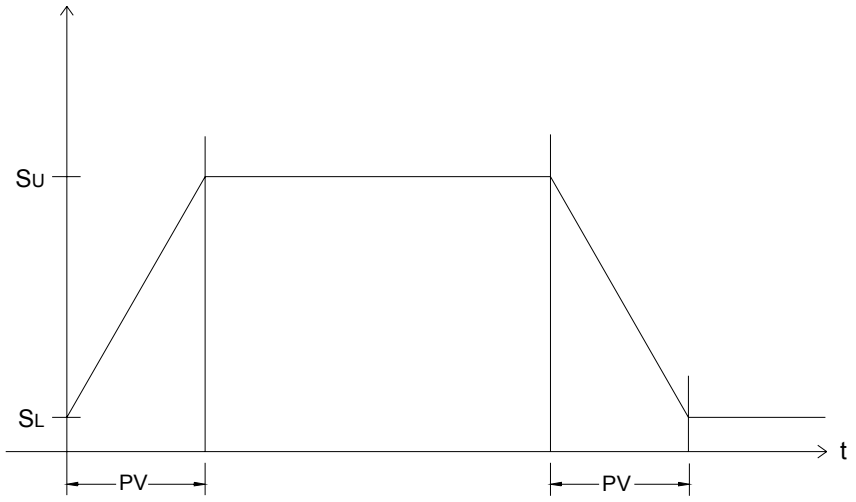
程序范例



将缓升 / 缓降值搬到 AO 输出缓存器 R3904 输出

T20 ：缓升 / 缓降定时器号码（0.01 秒时基定时器）
R100：缓升 / 缓降定时器设定值（M1974=0 时，单位为 0.01 秒）
每 10mS 的增 / 减量设定值（M1974=1 时，无单位）
R101：下限值（缓升初始值或缓降最终值）
R102：上限值（缓降初始值或缓升最终值）
R103：缓升 / 缓降值存放缓存器
R104：工作缓存器

- 若 M1974=0，当输入控制 M0 由 0→1 时，首先将定时器 T20 复归为 0，如此时 M2=1，则表示缓升而将 R101（下限）的值载入 R103，以后每 0.01 秒等比例（R102–R101 / R100）增加输出量，并存放到缓存器 R103，达到定时器设定值 R100 时，输出值等于 R102，输出 M102=1（达上限值）；如此时 M2=0，则表示缓降而将 R102（上限的值载入 R103，以后每 0.01 秒等比例（R102–R101 / R100）减少输出量，并存放到缓存器 R103，达定时器设定值 R100 时，输出值等于 R101，输出 M101=1（达下限值）。
- M1=1，暂停缓升 / 缓降动作。
- R102 的值必须大于 R101，否则缓升 / 缓降动作不执行，输出 M100=1。



列　表　(TABLE)　指　令

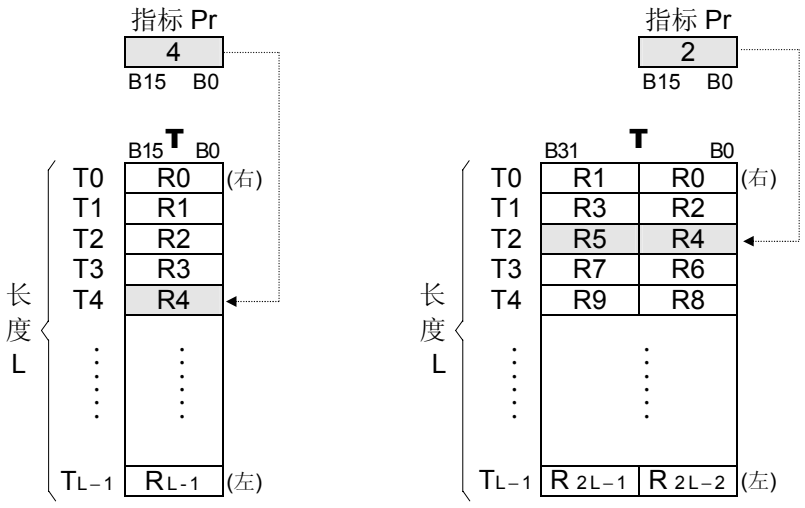
100. R→T
101. T→R
102. T→T
103. BT_M
104. T_SWP
105. R-T_S
106. T-T_C
107. T_FIL
108. T_SHF
109. T_ROT
110. QUEUE
111. STACK
112. BKCMP

- 列表是 2 个以上连续的缓存器（16 或 32 位）所组成，组成列表的缓存器个数称为列表的长度 L（Length），列表指令运算每次都以列表的一个缓存器为单位（即 16 或 32 位数据）。

- 列表指令主要在处理列表和缓存器或列表和列表间的数据处理，诸如搬移、拷贝、比较、搜寻等，是极为方便和重要的应用指令。

- 在列表指令运作中通常需要有一指引器来指定列表中的某一缓存器当作运算对象，此指引器我们称为指标 Pr（Pointer）。无论 16 或 32 位列表指令其指针都只是一个 16 位的缓存器。指标的有效范围为 0～L–1，分别用以对应（指引）至列表的缓存器 T0～T_{L-1}（共 L 个），以下为 16 位及 32 位列表的示意图。

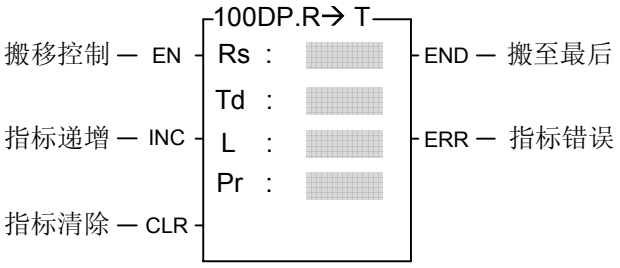
- 在列表运算中有左、右移或旋转，我们定义高序号的为左，低序号的为右，如下图示。



16 位列表　　32 位列表

FUN100 D P R→T	缓存器→列表搬移 (REGISTER TO TABLE MOVE)	FUN100 D P R→T

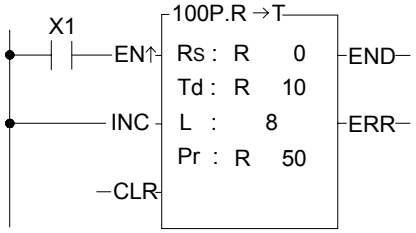
阶梯图符号



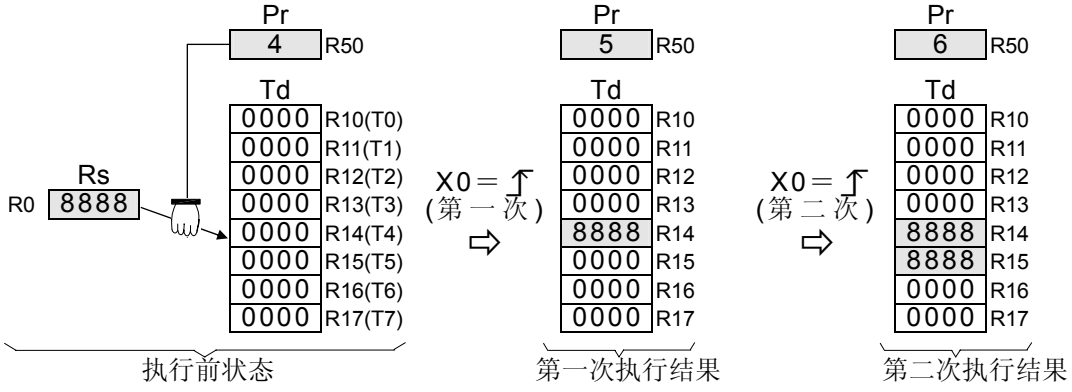
Rs：来源数据或其缓存器号码
Td：目的列表的起头缓存器号码
L ：目的列表的长度
Pr：指针缓存器号码
Rs，Td 可结合 V、Z、P0~P9 作间接寻址应用

操作数＼范围	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR
	WX0 | WX240	WY0 | WY240	WM0 | WM1896	WS0 | WS984	T0 | T255	C0 | C255	R0 | R3839	R3840 | R3903	R3904 | R3967	R3968 | R4167	R5000 | R8071	D0 | D4095	16 或 32 位 正、负数	V、Z P0~P9
Rs	○	○	○	○	○	○	○	○	○	○	○	○	○	○
Td		○	○	○	○	○	○		○	○*	○*	○		○
L							○				○*	○	2~2048	
Pr		○	○	○	○	○	○		○	○*	○*	○		

- 当搬移控制〝EN〞=1 或〝EN↑〞（P 指令）由 0→1 时，将来源缓存器 Rs 的内容写到目的列表 Td（长度为 L）中指针 Pr 所指的缓存器 Tdpr 去。在执行搬移前本指令会先检视指针清除〝CLR〞的输入信号，若〝CLR〞为 1，则会先将指标 Pr 的内容清除为 0 后再做搬移。在做完搬移动作后接着检视 Pr 值，若 Pr 值已达 L–1（已指在列表的最后一个缓存器）则将搬到最后旗号〝END〞设为 1后结束本指令的执行；若 Pr 小于 L–1，则再检视指标递增〝INC〞的状态，若〝INC〞为 1，则再将 Pr 加 1 后才结束执行。此外，指标清除〝CLR〞可单独执行，不受其它输入影响。
- 指标的有效范围为 0～L–1，超出此范围则指标错误〝ERR〞设为 1，且本指令不执行。

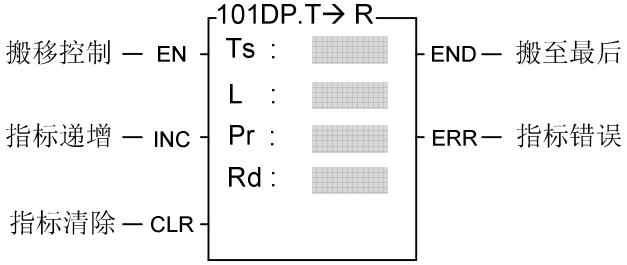


- 左图程序例假设刚开始指针 Pr=4，列表 Td 内容全部为 0，而 Rs 值为 8888，下图为当 X0 连续由 0→1 变换 2 次所得的运算结果。
- 因 INC 为 1，故每执行一次 Pr 即加 1 一次。



FUN101 D P T→R	列表→缓存器搬移 （TABLE TO REGISTER MOVE）	FUN101 D P T→R

阶梯图符号

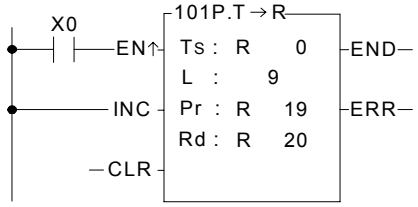


Ts：来源列表的起头缓存器号码
L ：来源列表的长度
Pr：指针缓存器号码
Rd：目的缓存器的起头号码
Ts，Rd 可结合 V、Z、P0~P9 作间接寻址应用

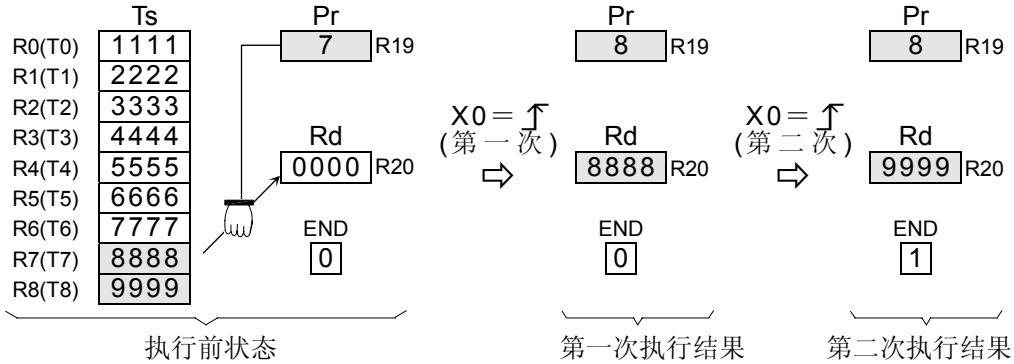
操作数＼范围	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR
	WX0 | WX240	WY0 | WY240	WM0 | WM1896	WS0 | WS984	T0 | T255	C0 | C255	R0 | R3839	R3840 | R3903	R3904 | R3967	R3968 | R4167	R5000 | R8071	D0 | D4095	16 或 32 位 正、负数	V、Z | P0~P9
Ts	○	○	○	○	○	○	○	○	○	○	○	○		○
L							○				○*	○		
Pr		○	○	○	○	○	○		○	○*	○*	○	2~2048	
Rd		○	○	○	○	○	○		○	○*	○*	○		○

● 当搬移控制〝EN〞=1 或〝EN↑〞（P 指令）由 0→1 时，将来源列表 Ts（长度为 L）中指针 Pr 所指的缓存器 Tspr 的内容值写到目的缓存器 Rd 去。在执行搬移前本指令会先检视指针清除〝CLR〞的输入信号，若〝CLR〞为 1，则会先将指标 Pr 的内容清除为 0 后再做搬移。在做完搬移动作后接着检视 Pr 值，若 Pr 值已达 L–1（已指在列表的最后一个缓存器）则将搬到最后旗号〝END〞设为 1 后结束本指令的执行；若 Pr 小于 L–1，则再检视指标递增〝INC〞的状态，若〝INC〞为 1，则再将 Pr 加 1 后才结束执行。此外，指标清除〝CLR〞可单独执行，不受其它输入影响。

● 指标的有效范围为 0～L–1，超出此范围则指标错误〝ERR〞设为 1，且本指令不执行。

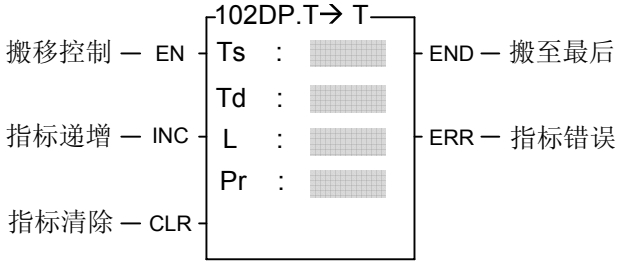


- 左图程序例假设刚开始指针 Pr=7，而 Ts 和 Rd 内容如下图左所示，当 X0 连续由 0→1 变换 2 次后，可得到下图右方的两个结果。
- 第二次执行时指标已到最后，故不再增加。



FUN102 D P T→T	列表→列表搬移 （TABLE TO TABLE MOVE）	FUN102 D P T→T

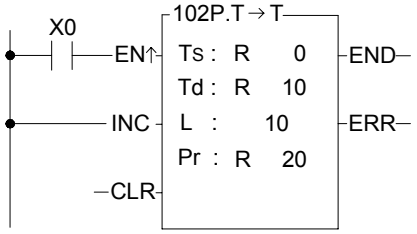
阶梯图符号



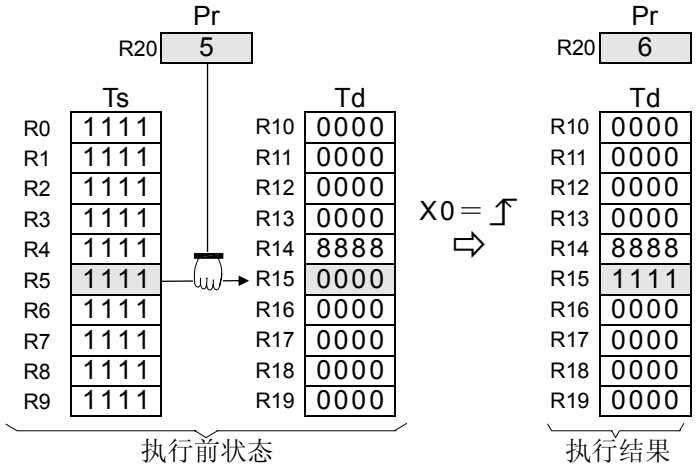
Ts：来源列表的缓存器起头号码
Td：目的列表的缓存器起头号码
L ：列表（Ts 和 Td）的长度
Pr：指针缓存器号码
Ts，Td 可结合 V、Z、P0~P9 作间接寻址应用

操作数 \ 范围	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR
	WX0 | WX240	WY0 | WY240	WM0 | WM1896	WS0 | WS984	T0 | T255	C0 | C255	R0 | R3839	R3840 | R3903	R3904 | R3967	R3968 | R4167	R5000 | R8071	D0 | D4095	2 | 2048	V、Z P0~P9
Ts	○	○	○	○	○	○	○	○	○	○	○	○		○
Td		○	○	○	○	○	○		○	○*	○*	○		○
L							○				○*	○	○	
Pr		○	○	○	○	○	○		○	○*	○*	○		

- 当搬移控制〝EN〞=1 或〝EN↑〞（P 指令）由 0→1 时，将来源列表中指标 Pr 所指的那一个缓存器数据 Tspr 搬到目的列表中同样是 Pr 所指的缓存器 Tdpr 去。在执行搬移前本指令会先检视指针清除输入〝CLR〞信号，如果为 1，会先清除 Pr 为 0 后再搬移（此时为 Ts0→Td0）。在作完搬移动作后接着检视指标 Pr 的值，如果 Pr 值已达 L–1（已指到列表的最后一对缓存器），则将搬到最后旗号〝END〞设为 1 后结束指令的执行；若 Pr 值小于 L–1，将再检视指标递增〝INC〞的状态，若〝INC〞为 1，则再将 Pr 值加 1 后才结束指令的执行。此外，指标清除〝CLR〞可单独执行，不受其它输入影响。
- 指标的有效范围为 0～L–1，超出此范围则指标错误〝ERR〞设为 1，且本指令不执行。

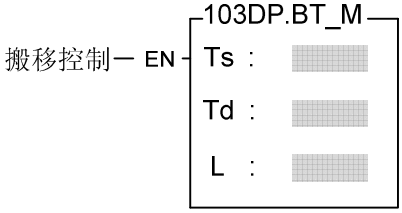


- 左图程序范例起始状态如下图左的执行前状态，当 X0 由 0→1 时，将得到下图右的结果，Ts 为来源列表不受指令执行的影响。



FUN103 D P BT_M	整个列表搬移 （BLOCK TABLE MOVE）	FUN103 D P BT_M

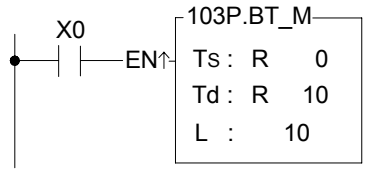
阶梯图符号



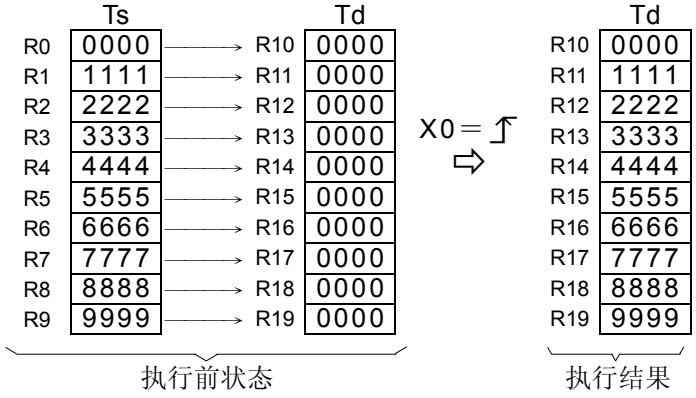
Ts：来源列表的起头缓存器号码
Td：目的列表的起头缓存器号码
L ：来源和目的列表的长度
Ts，Td 可结合 V、Z、P0~P9 作间接寻址应用

操作数 \ 范围	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR
	WX0 | WX240	WY0 | WY240	WM0 | WM1896	WS0 | WS984	T0 | T255	C0 | C255	R0 | R3839	R3840 | R3903	R3904 | R3967	R3968 | R4167	R5000 | R8071	D0 | D4095	2 | 256	V、Z P0~P9
Ts	○	○	○	○	○	○	○	○	○	○	○	○		○
Td		○	○	○	○	○	○		○	○*	○*	○		○
L							○				○*	○	○	

- 本指令来源和目的列表长度相同，且在一次指令执行中即将整个 Ts 列表数据全部搬到 Td 中，故无需指标来指定列表中的某一缓存器。
- 当搬移控制〝EN〞=1 或〝EN↑〞（P 指令）由 0→1 时，将来源列表 Ts（长度为 L）的所有数据整个一次搬到相同长度的目的列表 Td 去。
- 因本指令在每次执行中是整个列表一次搬完，如果列表长度较长时，将会耗费较多的时间，实用上应使用指令，以免每次扫描都重复同样的搬移动作而浪费时间。

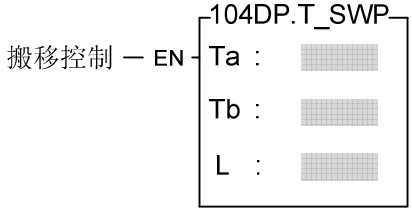


- 左图程序范例，假设 Ts 和 Td 列表的状态如下图左的执行前状态，当 X0 由 0→1 时，可得到下图右的执行结果。



FUN104 D P T_SWP	整个列表互换 （BLOCK TABLE SWAP）	FUN104 D P T_SWP

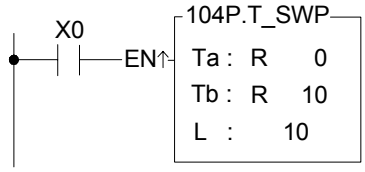
阶梯图符号



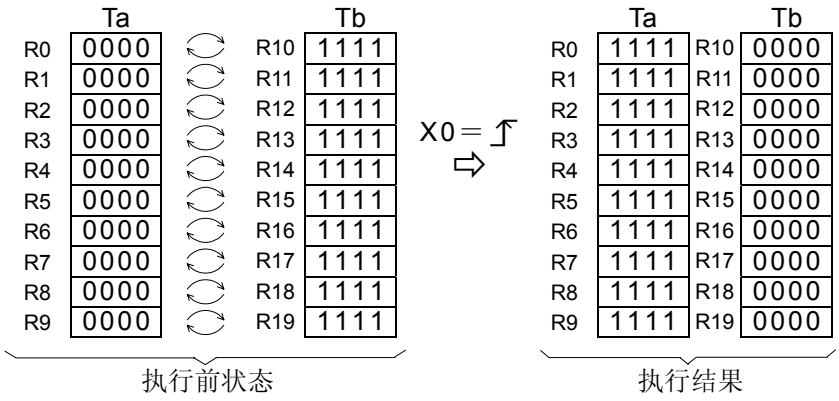
Ta：列表 a 的缓存器起头号码
Tb：列表 b 的缓存器起头号码
L ：列表 a 和 b 的长度
Ta，Tb 可结合 V、Z、P0~P9 作间接寻址应用

操作数 \ 范围	WY	WM	WS	TMR	CTR	HR	OR	SR	ROR	DR	K	XR
	WY0 | WY240	WM0 | WM1896	WS0 | WS984	T0 | T255	C0 | C255	R0 | R3839	R3904 | R3967	R3968 | R4167	R5000 | R8071	D0 | D4095	2 | 256	V、Z P0~P9
Ta	○	○	○	○	○	○	○	○*	○*	○		○
Tb	○	○	○	○	○	○	○	○*	○*	○		○
L						○			○*	○	○	

- 本指令列表 a，b 内容要对换（SWAP），故列表长度必须相同，且列表本身必须是可写入的缓存器。因其为一次执行即全部互换完毕，故无需指标。
- 当搬移控制〝EN〞=1 或〝EN↑〞（P 指令）由 0→1 时，将列表 Ta 和列表 Tb 的内容整个互换。
- 本指令因一次互换完毕，若列表长度过长，将会耗费较多的时间，实用上应使用指令。

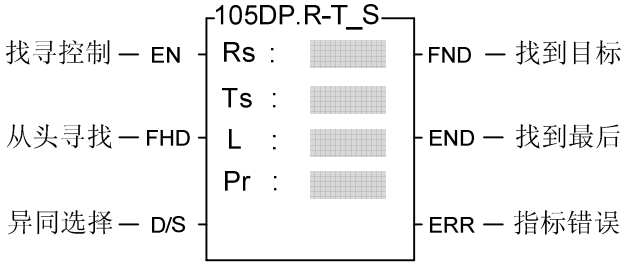


- 左图程序范例假设 Ta 和 Tb 的起始状态如下图左执行前的情况，当 X0 由 0→1 时，可得到下图右执行的结果。



FUN105 D P R-T_S	缓存器对列表找寻异同 （REGISTER TO TABLE SEARCH）	FUN105 D P R-T_S

阶梯图符号

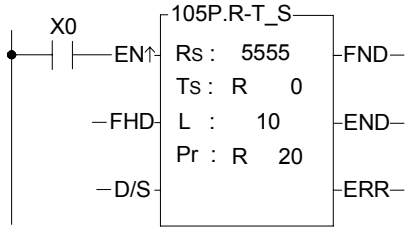


Rs ：找寻的来源（样本）数据或其缓存器号码
Ts：被找寻的列表起头缓存器号码
L ：列表长度
Pr：指针，用来记录目标所在的位置值
Rs，Ts 可结合 V、Z、P0~P9 作间接寻址应用

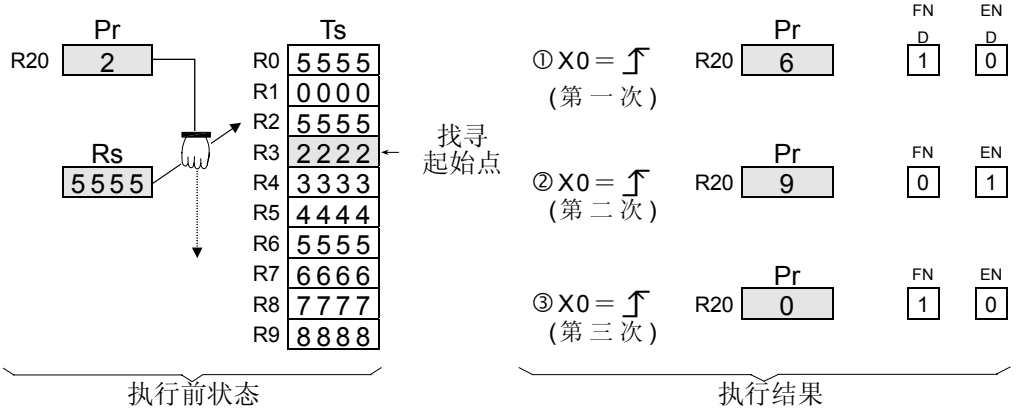
操作数 \ 范围	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR
	WX0 | WX240	WY0 | WY240	WM0 | WM1896	WS0 | WS984	T0 | T255	C0 | C255	R0 | R3839	R3840 | R3903	R3904 | R3967	R3968 | R4167	R5000 | R8071	D0 | D4095	16 或 32 位 正、负数	V、Z P0~P9
Rs	○	○	○	○	○	○	○	○	○	○	○	○	○	○
Ts	○	○	○	○	○	○	○	○	○	○	○	○		○
L							○				○*	○	2~256	
Pr		○	○	○	○	○	○		○	○*	○*	○		

● 当找寻控制〝EN〞=1 或〝EN↑〞（P 指令）由 0→1 时，自列表 Ts 的开头第一个缓存器开始（〝FHD〞=1 或 Pr 值已达 L–1 时）或自列表中当时指针所指那个缓存器的下一个缓存器 Tspr+1 开始（〝FHD〞=0 同时 Pr 值小于 L–1）往下找寻和样本数据 Rs 不同（D/S=1 时）或相同（D/S=0 时）的缓存器。若找到目标（不同或相同的），则立即停止找寻动作，并将该目标在列表的位置序号值存放到指标 Pr 去，同时将找到目标旗号〝FND〞设为 1 后结束本指令的执行。当找到列表的最后一个缓存器时，无论是否找到目标都将结束该次指令执行，并将找到最后旗号〝END〞设为 1，而 Pr 值则停在 L–1。当本指令下次再度被执行时，Pr 将会自动循环至列表的最开头（Pr=0）开始往下找寻。

● 指标值的有效范围为 0～L–1，如果值超出该范围，则指标错误旗号〝ERR〞变为 1，且本指令不执行。

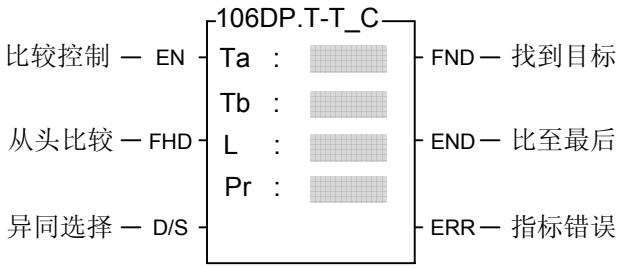


• 左图程序范例在列表中找寻数值同为 5555 的缓存器（因 D/S=0，为找相同），执行前指标指在 R2，但开始找寻点是 Pr+1（即 R3 开始），在 X0 连续 3 次由 0→1 动作后，可得到如下图①、②、③三个结果。



FUN106 D P T-T_C	列表对列表比较异同 （TABLE TO TABLE COMPARE）	FUN106 D P T-T_C

阶梯图符号

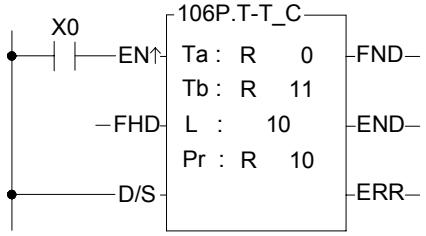


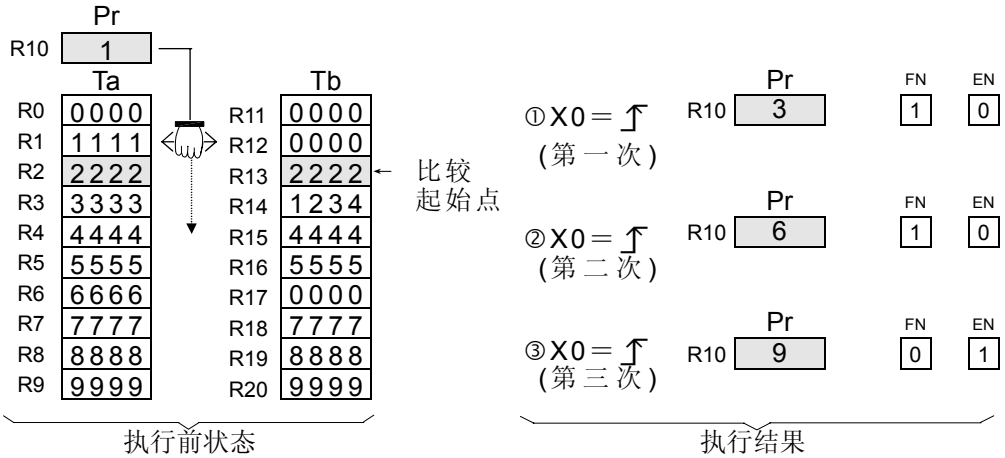
Ta：列表 a 的缓存器起头号码
Tb：列表 b 的缓存器起头号码
L ：列表 a 和 b 的长度
Pr：指针，用以记录目标所在的位置值
Ta，Tb 可结合 V、Z、P0~P9 作间接寻址应用

操作数 \ 范围	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR
	WX0 | WX240	WY0 | WY240	WM0 | WM1896	WS0 | WS984	T0 | T255	C0 | C255	R0 | R3839	R3840 | R3903	R3904 | R3967	R3968 | R4167	R5000 | R8071	D0 | D4095	2 | 256	V、Z P0~P9
Ta	○	○	○	○	○	○	○	○	○	○	○	○		○
Tb	○	○	○	○	○	○	○	○	○	○	○	○		○
L							○				○*	○	○	
Pr		○	○	○	○	○	○		○	○*	○*	○		

● 当比较控制〝EN〞=1 或〝EN↑〞（P 指令）由 0→1 时，从 Ta 和 Tb 两列表中的最开头那对缓存器（Ta0 和 Tb0）开始（〝FHD〞=1 或 Pr 值已达 L–1 时）或从当时 Pr 所指那对缓存器的下一对缓存器（Tapr+1 和 Tbpr+1）开始（〝FHD〞=0 同时 Pr 值小于 L–1）往下双双成对比较寻找内容值不同（〝D/S〞=1 时）或相同（〝D/S〞=0 时）的缓存器对（Pair），当找到目标（不同或相同者）后立即停止比较寻找，同时将该对目标在列表中的位置序号值存放到指标 Pr 去，并将找到目标旗号〝FND〞设为 1，然后结束本指令的执行。当找到列表的最后一对缓存器，无论其是否为所要寻找的目标，都将结束本指令的执行，同时将比到最后旗号〝END〞设为 1，而指标值则停在 L–1。而当本指令下一次再度被执行时，Pr 将会自动循环到列表的最开头开始往下找起。指标 Pr 的范围为 0～L–1，在执行中应避免更动到 Pr 值，以免影响其正确的比较寻找，若 Pr 值超出此范围，则指标错误旗号〝ERR〞变为 1，且本指令不执行。

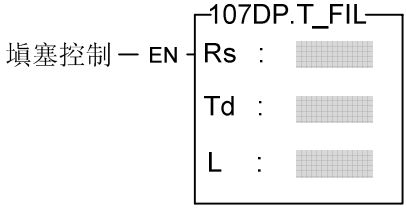
• 右图程序范例为自指针所指的下一个缓存器开始往下比较寻找（因〝FHD〞为 0）两列表中数据不相同的缓存器对（因〝D/S〞=1）。刚开始 Pr 指在 Ta1 和 Tb1，虽然两列表中分别在列表位置 1、3、6 三处有数据不同，但因其非从头比较，故本指令将先找到位置 3，下图右显示 X0 由 0→1 变换 3 次的结果。





FUN107 D P T_FIL	列表填塞 （TABLE FILL）	FUN107 D P T_FIL

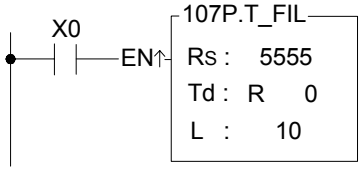
阶梯图符号



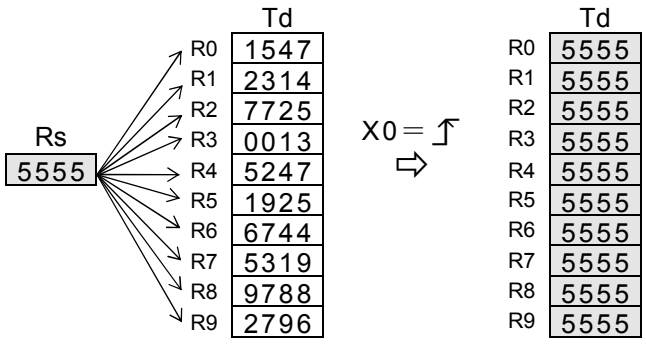
Rs ：要填入列表的来源数据或其缓存器号码
Td：列表的起头缓存器号码
L ：列表的长度
Rs，Td 可结合 V、Z、P0~P9 作间接寻址应用

操作数 \ 范围	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR
	WX0 | WX240	WY0 | WY240	WM0 | WM1896	WS0 | WS984	T0 | T255	C0 | C255	R0 | R3839	R3840 | R3903	R3904 | R3967	R3968 | R4167	R5000 | R8071	D0 | D4095	16 或 32 位 正、负数	V、Z P0~P9
Rs	○	○	○	○	○	○	○	○	○	○	○	○	○	○
Td		○	○	○	○	○	○		○	○*	○*	○		○
L							○				○*	○	2~256	

- 当填塞控制〝EN〞=1 或〝EN↑〞（P 指令）由 0→1 时，将 Rs 的数据填写到列表 Td 中所有的缓存器中。
- 本指令主要用于列表的清除（填 0）或一致化（全部填相同值）用，实用上应使用指令。



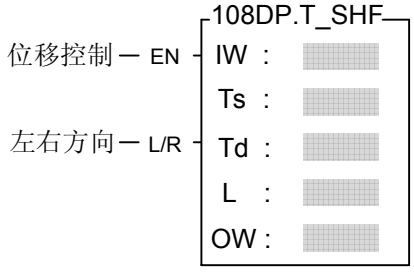
- 左图程序范例，将列表 Td 全部填入 5555，如下图结果。



执行前状态　　执行结果

FUN108 D P T_SHF	列表位移 （TABLE SHIFT）	FUN108 D P T_SHF

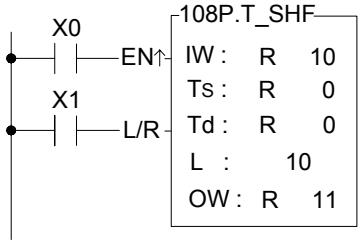
阶梯图符号



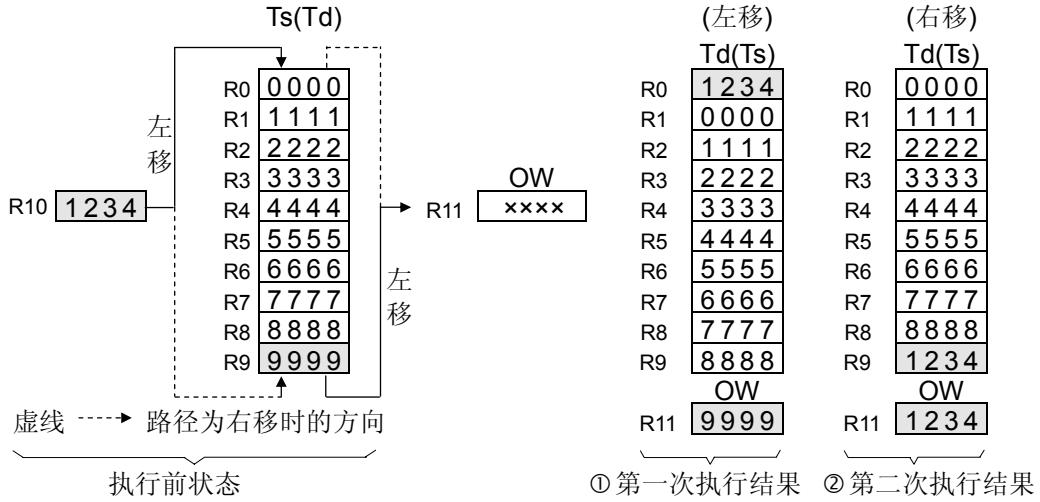
IW ：填补位移空位的数据或其缓存器号码
Ts ：被位移的来源列表
Td ：存放结果饿目的列表
L ：列表 Ts 和 Td 长度
OW：存放自列表移出数据的缓存器号码
Ts，Td 可结合 V、Z、P0~P9 作间接寻址应用

操作数 \ 范围	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR
	WX0 | WX240	WY0 | WY240	WM0 | WM1896	WS0 | WS984	T0 | T255	C0 | C255	R0 | R3839	R3840 | R3903	R3904 | R3967	R3968 | R4167	R5000 | R8071	D0 | D4095	16 或 32 位 正、负数	V、Z | P0~P9
IW	○	○	○	○	○	○	○	○	○	○	○	○	○	
Ts	○	○	○	○	○	○	○	○	○	○	○	○		○
Td		○	○	○	○	○	○		○	○*	○*	○		○
L							○				○*	○	2~256	
OW		○	○	○	○	○	○		○	○*	○*	○		

● 当位移控制〝EN〞=1 或〝EN↑〞（P 指令）由 0→1 时，将列表 Ts 的数据全部取出，整个向左（〝L/R〞=1 时）或向右（〝L/R〞=0 时）位移一个位置，因位移造成的空位，用 IW 填补，再将该位移过并填补的结果，写到列表 Td 去，而因位移而挤出的资料则写到 OW 去。

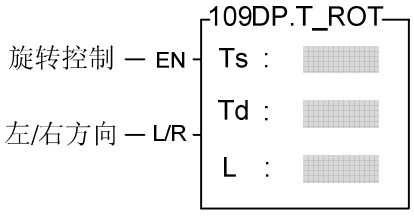


- 左图程序范例 Ts 和 Td 相同，因此就是把列表自己位移后再写回自己（列表必须为可写入的缓存器），如下图执行前的状态，先执行一次左移（使 X1=1，再使 X0 由 0→1），然后再作一次右移（使 X1=0，再使 X0 由 0→1），将可得到下图右方的两个结果。



FUN109 D P T_ROT	列表旋转 （TABLE ROTATE）	FUN109 D P T_ROT

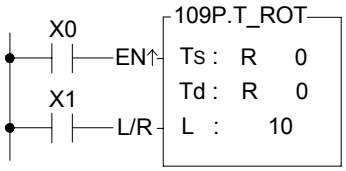
阶梯图符号



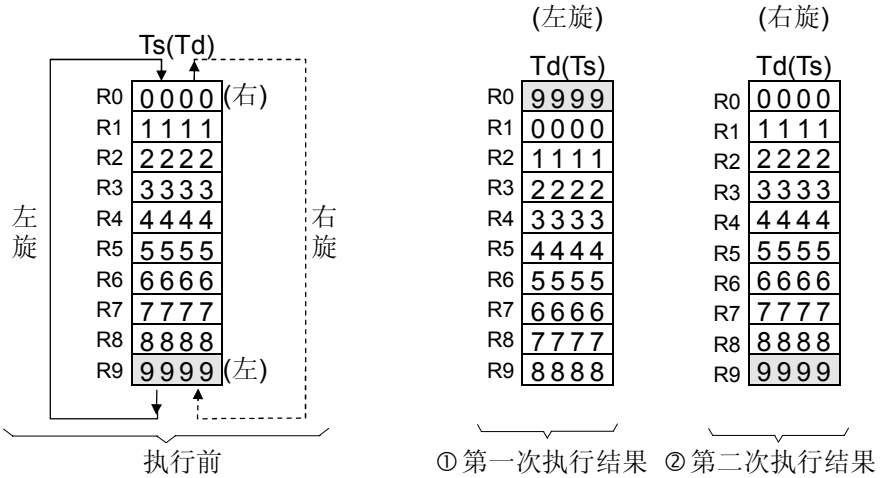
Ts：被旋转的来源列表
Td：存放旋转结果的列表
L ：列表 Ts 和 Td 的长度
Ts，Td 可结合 V、Z、P0~P9 作间接寻址应用

操作数 \ 范围	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR
	WX0 | WX240	WY0 | WY240	WM0 | WM1896	WS0 | WS984	T0 | T255	C0 | C255	R0 | R3839	R3840 | R3903	R3904 | R3967	R3968 | R4167	R5000 | R8071	D0 | D4095	2 | 256	V、Z P0~P9
Ts	○	○	○	○	○	○	○	○	○	○	○	○		○
Td		○	○	○	○	○	○		○	○*	○*	○		○
L							○				○*	○	○	

● 当旋转控制〝EN〞=1 或〝EN↑〞（P 指令）由 0→1 时，将列表 Ts 的数据取出后向左（〝L/R〞=1 时）或向右（〝L/R〞=0 时）旋转一个位置后，将此旋转过的结果写到 Td 去。

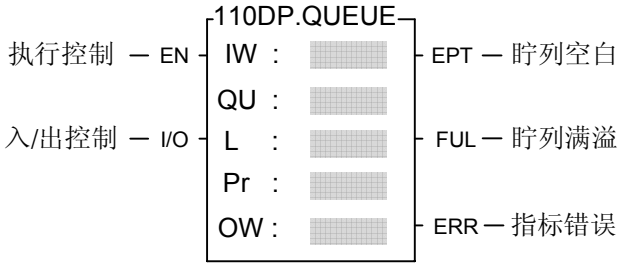


• 左图程序例，Ts 和 Td 相同，故是将列表自己取出旋转后再写回自己，如下图执行前的状态，先执行一次左旋（使 X1=1，再使 X0 由 0→1），然后再作一次右旋（使 X1=0，再使 X0 由 0→1）后，可得到下图右方的①、②两个结果。



FUN110 D P QUEUE	贮列 （QUEUE）	FUN110 D P QUEUE

阶梯图符号

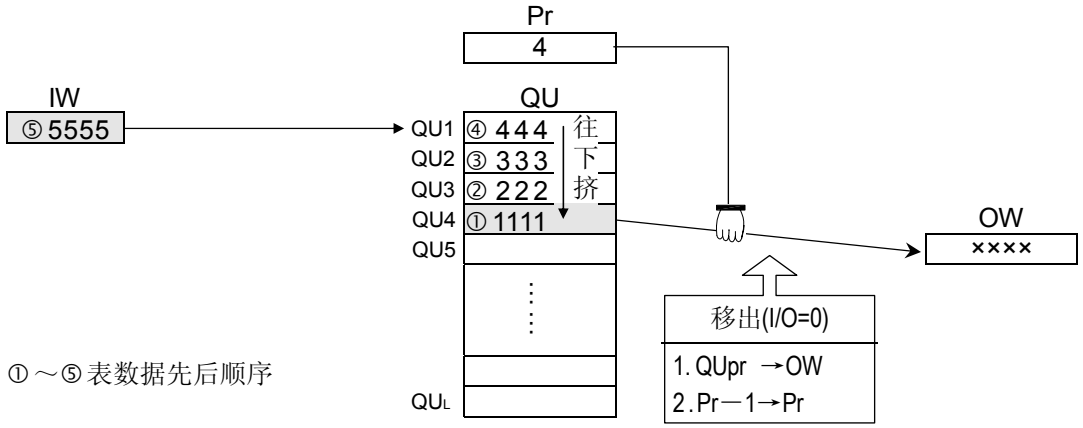


IW：挤入贮列的数据或其缓存器号码
QU：贮列的起头缓存器号码
L ：贮列的长度
Pr ：指针缓存器号码
OW：接收自贮存器移出数据的缓存器号码
QU 可结合 V、Z、P0~P9 作间接寻址应用

操作数＼范围	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR
	WX0 | WX240	WY0 | WY240	WM0 | WM1896	WS0 | WS984	T0 | T255	C0 | C255	R0 | R3839	R3840 | R3903	R3904 | R3967	R3968 | R4167	R5000 | R8071	D0 | D4095	16 或 32 位 正、负数	V、Z P0~P9
IW	○	○	○	○	○	○	○	○	○	○	○	○	○	
QU		○	○	○	○	○	○		○	○	○*	○		○
L							○				○*	○	2~256	
Pr		○	○	○	○	○	○		○	○*	○*	○		
OW		○	○	○	○	○	○		○	○*	○*	○		

● 贮列（QUEUE）也属列表的一种，其有别于一般列表的是其贮列缓存器序号是由 1～L 而非 0～L–1，也就是 QU_1～QU_L，分别以指标 Pr=1～L 来对应，而指标 Pr=0 则用以表示该贮列为空白。

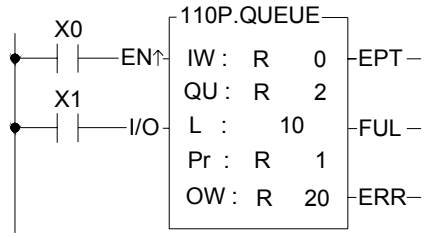
● 贮列（QUEUE）是一种先进先出装置，即最先挤入（PUSH）贮列的资料，在移出（POP）时要最先移出。本指令的贮列是由 QU 缓存器开始的连续 L 个 16 位或 32 位（D 指令）缓存器所组成。



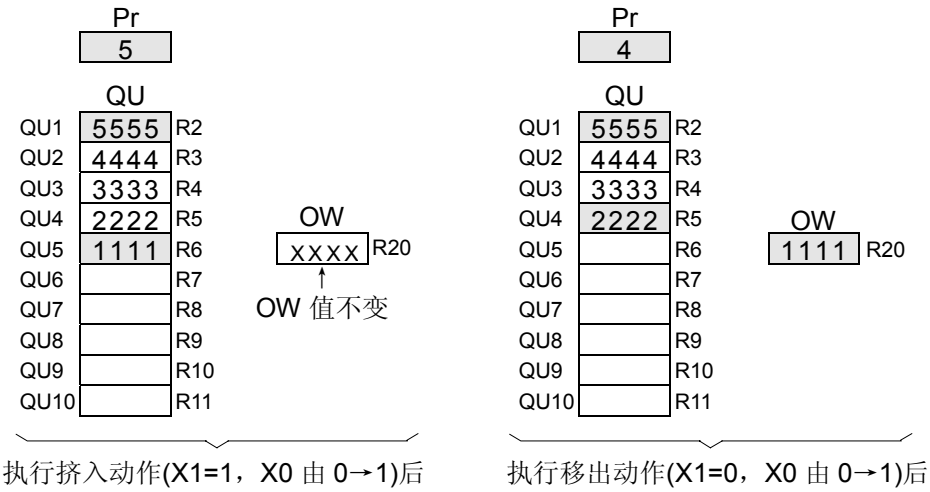
● 贮列指令的动作是当执行控制〝EN〞=1 或〝EN↑〞（P 指令）由 0→1 时，由入出控制〝I/O〞的状态判断是将挤入数据 IW 挤入贮列（〝I/O〞=1 时）或将贮列中最早挤入的那数据移出送到 OW 去（〝I/O〞=0 时），如上示意图所示，挤入数据 IW 永远往贮列的第一个缓存器 QU_1 挤，挤入后 Pr 立刻加 1，使指标能永远指在贮列中最早挤入的资料。在移出时则直接将 Pr 所指的数据送到 OW，再将 Pr 减 1，使它仍然保持指在剩余数据中最先挤入的那个资料。

FUN110 D P QUEUE	贮列 （QUEUE）	FUN110 D P QUEUE

● 在贮列未挤入任何资料或填入的都已被移出时（Pr=0），贮列空白旗号〝EPT〞将变为 1，此时即使再有移出动作，本指令也不执行。而如果数据仅挤入不移出或挤入多移出少，最终造成贮列已被挤满（指标 Pr 已指在 QU_L 处），则贮列满溢旗号〝FUL〞变为 1，此时若再有挤入动作本指令也不再执行。本指令的指针为供贮列于存取时永远保持指在最先挤入的数据，应避免其它程序去更动到它，否则将造成运作错误。若有特定的应用需强制设定指标值，则其容许范围为 0～L（0 表空白，1～L 则分别对应到 QU_1～QU_L），超出此范围，指标错误旗号〞ERR〞设为 1，且本指令不执行。



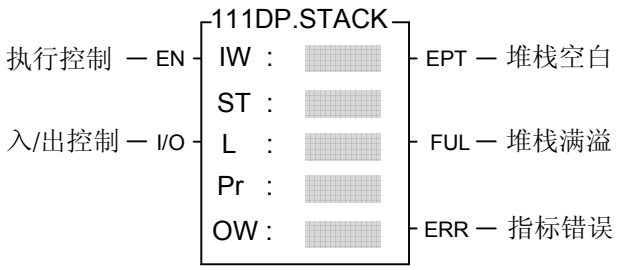
• 左图范例程序，假设其起始状态如上页的贮列示意图范例所示，先将它作一次贮列挤入动作，再作一次移出动作，将可得到如下两个执行结果。无论如何 Pr 总是指在 QUEUE 中最先挤入的资料。



执行挤入动作(X1=1，X0 由 0→1)后　　执行移出动作(X1=0，X0 由 0→1)后

FUN111 D P STACK	堆栈 （STACK）	FUN111 D P STACK

阶梯图符号

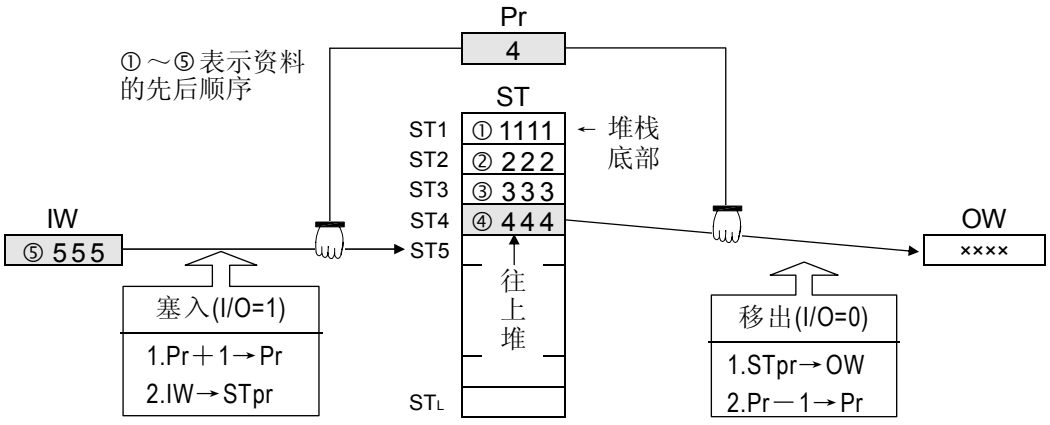


IW ：塞入堆栈的数据或其缓存器号码
ST ：堆栈饿起头缓存器号码
L ：堆栈的长度
Pr ：指针缓存器号码
OW：接收堆栈移出数据的缓存器号码
ST 可结合 V、Z、P0~P9 作间接寻址应用

操作数 \ 范围	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR
	WX0 | WX240	WY0 | WY240	WM0 | WM1896	WS0 | WS984	T0 | T255	C0 | C255	R0 | R3839	R3840 | R3903	R3904 | R3967	R3968 | R4167	R5000 | R8071	D0 | D4095	16 或 32 位正、负数	V、Z P0~P9
IW	○	○	○	○	○	○	○	○	○	○	○	○	○	
ST		○	○	○	○	○	○		○	○*	○*	○		○
L							○				○*	○	2~256	
Pr		○	○	○	○	○	○		○	○*	○*	○		
OW		○	○	○	○	○	○		○	○*	○*	○		

- 堆栈和贮列一样同属于列表的一种，其指标序号性质和贮列完全相同，以 Pr=1～L 来对应 ST_1～ST_L，而 Pr=0 则用以表示该堆栈为空白。

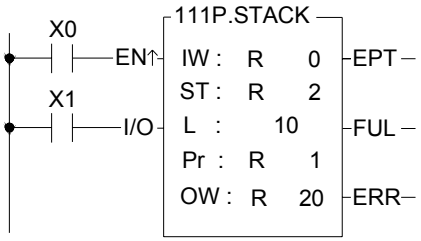
- 堆栈和贮列正好相反，是一种后进先出的装置，即最后塞入（PUSH）堆栈的数据，在移出（POP）时要最先移出，堆栈是由 ST 开始的连续 L 个 16 位或 32 位（D 指令）缓存器所组成，如下的示意图所示：



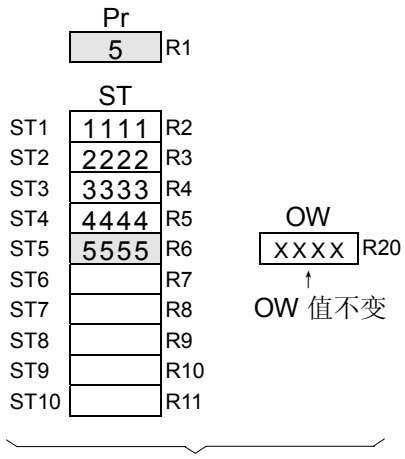
- 堆栈指令的动作是当执行控制〝EN〞=1 或〝EN↑〞（P 指令）由 0→1 时，由入出控制〝I/O〞的状态判断是将塞入数据 IW 塞入堆栈（〝I/O〞=1 时）或将堆栈中指针 Pr 所指的数据（现存数据中最后塞入的）搬出送到 OW 去（〝I/O〞=0 时），注意塞入的数据是堆栈（STACKING）上去的，故在塞入前要先将 Pr 加 1 使它指到堆栈的最上面（倒着看），再将数据塞入，而在移出时只须将指针 Pr 所指的数据（最后塞入的数据）送到 OW，然后再将 Pr 减 1，使它无论如何动作指标 Pr 都能永远指在堆栈中最后塞入的那个数据。

FUN111 D P STACK	堆栈 （STACK）	FUN111 D P STACK

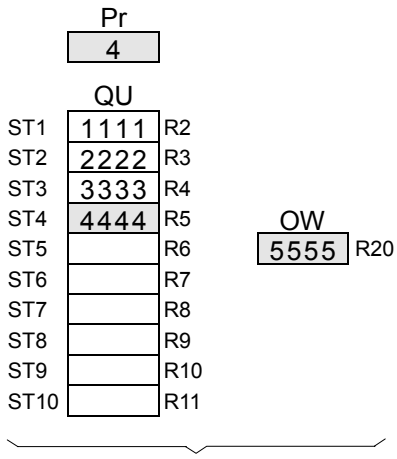
● 在堆栈未塞入任何资料或塞入的都已被移出时（Pr=0），堆栈空白旗号〝EPT〞变为 1，此时若再有移出动作本指令也不执行，而若数据仅塞入不移出或塞入多移出少，最终造成整个堆栈被塞满（指针 Pr 已指在 ST 处），则堆栈满溢旗号〝FUL〞变为 1，此时若再有塞入动作本指令也不再执行。同贮列一样，堆栈的指针 Pr 应避免去更动它，若有特殊应用需强制设定 Pr 值，其有效范围为 0～L（0 表空白，1～L 则分别对应到 ST_1～ST_L），超出此范围，指标错误〝ERR〞设为 1，且本指令不执行。



• 左图范例程序，假设堆栈状况正好如上页的堆栈示意图所示，先将它作一次塞入动作，再执行一次移出动作，可得到如下两个结果，无论如何动作 Pr 始终指在堆栈中最后塞入的那个数据。



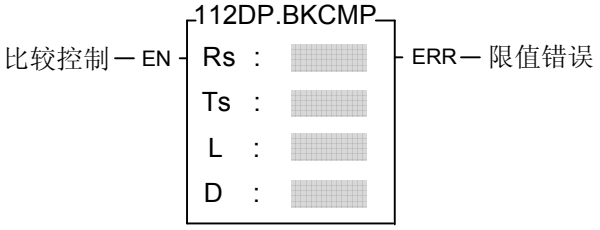
执行塞入动作(X1=1，X0 由 0→1)后



执行移出动作(X1=0，X0 由 0→1)后

FUN112 D P BKCMP	区块比较（凸轮开关 DRUM）（BLOCK COMPARE）	FUN112 D P BKCMP

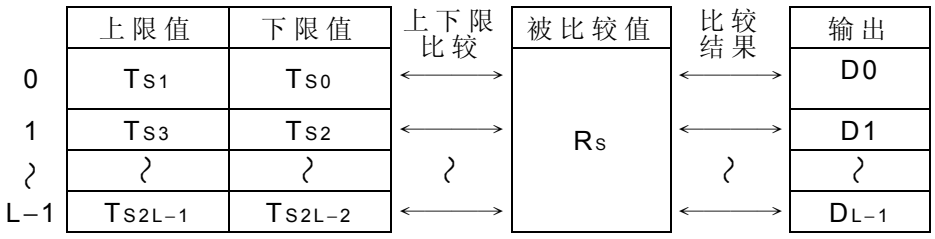
阶梯图符号



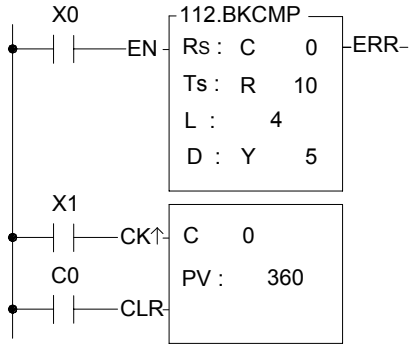
Rs：被比较的数据或其缓存器号码
Ts：上下限值缓存器区块的起头号码
L ：上下限值的组数
D ：存放比较结果的继电器起头号码

操作数 \ 范围	Y	M	S	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K
	Y0 | Y255	M0 | M999	S0 | S999	WX0 | WX240	WY0 | WY240	WM0 | WM1896	WS0 | WS984	T0 | T255	C0 | C255	R0 | R3839	R3840 | R3903	R3904 | R3967	R3968 | R4167	R5000 | R8071	D0 | D4095	16 或 32 位正、负数
Rs				○	○	○	○	○	○	○	○	○	○	○	○	○
Ts				○	○	○	○	○	○	○	○	○	○	○	○	
L										○				○*	○	1~256
D	○	○	○													

- 当比较控制〝EN〞=1 或〝EN↑〞（P 指令）由 0→1 时，将 Rs 的内容值逐一和由缓存器 Ts 开始的 L 组 16 或 32（D 指令）位的上下限值（由 T0 开始每相邻的两缓存器形成一组上下限设定值）作比较，若 Rs 值落在该组设定值范围内，则将比较结果的继电器 D 中对应于该组的位设为 1，否则为 0，直到比完所有 L 组上下限设定值为止。
- 当 M1975=0 时，若上下限设定值中有任何一组的上限值小于下限值，则限值错误旗号〝ERR〞设为 1，而且该组比较输出为 0。
- 当 M1975=1 时，下限值可大于上限值的设定，而适用于 360°圆周运动，当有跨 0°的电子凸轮角度的应用。



- 本指令实质上为一个机械式的绝对式凸轮开关（DRUM）指令；其也可置放在定时中断程序里，配合 FUN74(IMDIO)可得到较准确的电子凸轮角度输出。

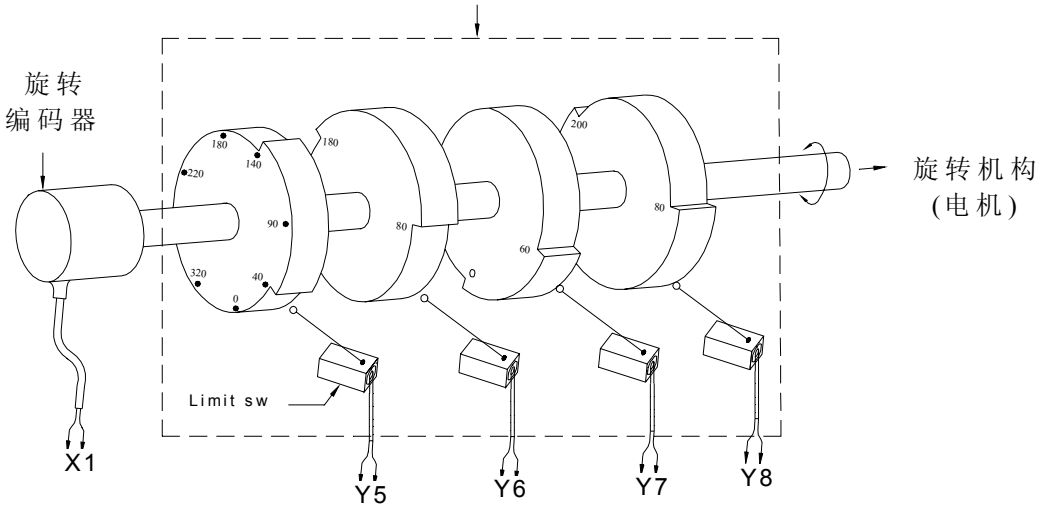


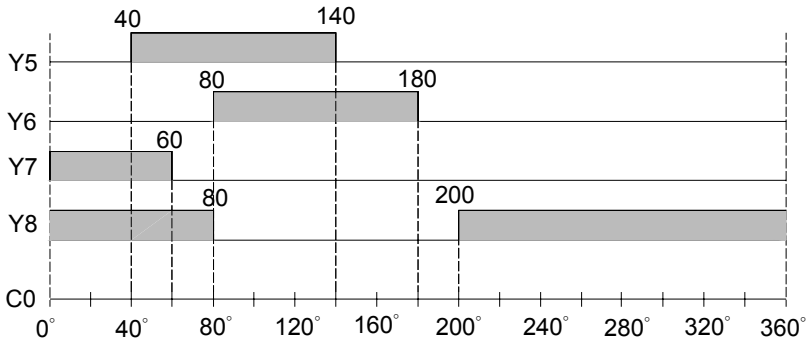
- 本程序范例指定 C0 值作为凸轮轴的旋转角度（Rs），并借由本区块比较指令，将 Rs 和 R10R11、R12R13、R14R15 和 R16R17 等 4 组（因 L=4）上下限设定值作比较，而得到 Y5～Y8 四个凸轮输出点的输出结果。
- 程序中输入接点 X1 为一旋转角度检测器，是安装在凸轮轴上，凸轮轴角度每转动 1 度，X1 就产生一个脉冲，凸轮轴转一周，X1 会产 360 个脉冲。

FUN112 D P BKCMP	区块比较（凸轮开关 DRUM） （BLOCK COMPARE）	FUN112 D P BKCMP

● 上图程序配合一个旋转编码器或其它转动角度检测装置（直接连结到旋转机构）即可组成和实际凸轮机械结构等效的机构装置（如下图虚线标示的机构）。同时通过上下限值的修改，便可随时改变凸轮触动的角度范围，是传统凸轮机构所无法达成。

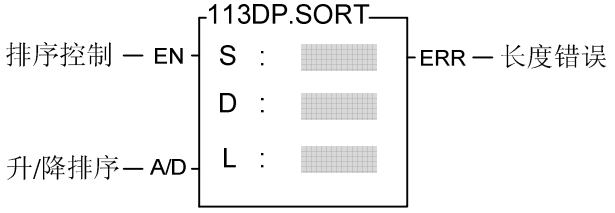
本指令范例所取代的
实际凸轮机械结构





FUN113 D P SORT	大小排序便利指令 （SORTING）	FUN113 D P SORT

阶梯图符号



S：要排序的来源数据起头缓存器号码
D：排序后的数据起头缓存器号码
L：排序的数据长度

操作数 \ 范围	TMR	CTR	HR	IR	OR	SR	ROR	DR	K
	T0 | T255	C0 | C255	R0 | R3839	R3840 | R3903	R3904 | R3967	R3968 | R4167	R5000 | R8071	D0 | D4095	2 | 127
S	○	○	○	○	○	○	○	○	
D			○				○*	○	
L			○				○	○	○

- 当排序控制〝EN〞=1 或〝EN↑〞（P 指令）由 0→1 时，将以 S 为起始的 L 个数据由小而大排序（A/D=1）或由大而小排序（A/D=0），并将排序结果存放到以 D 为起始的缓存器中。
- 当排序的数据长度错误（127 < L 或 L < 2）时，本指令不执行，输出〝ERR〞=1。

- 左图程序范例，将 R0 为起始的缓存器列表由小而大排序，并将排序结果存放到以 R10 为起始的缓存器列表中，如下图结果。

	S
R0	1547
R1	2314
R2	7725
R3	0013
R4	5247
R5	1925
R6	6744
R7	5319
R8	9788
R9	2796

执行前状态

X0 = ↑

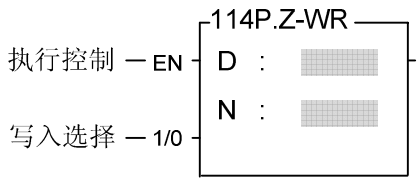
⇨

	D
R10	0013
R11	1547
R12	1925
R13	2314
R14	2796
R15	5247
R16	5319
R17	6744
R18	7725
R19	9788

执行结果

FUN114 D P Z-WR	区域写入 （ZONE WRITE）	FUN114 D P Z-WR

阶梯图符号



D：要写入或清除区域的起始地址

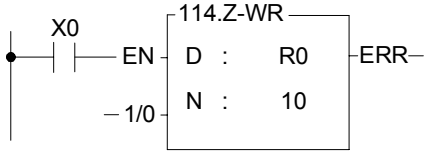
N要写入或清除区域的长度:1~511

D、N可结合V、Z、P0~P9作间接寻址应用

Range / Operand	Y	M	S	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR
	Y0 | Y255	M0 | M1911	S0 | S99	WY0 | WY240	WM0 | WM1896	WS0 | WS984	T0 | T255	C0 | C255	R0 | R3839	R3840 | R3903	R3904 | R3967	R3968 | R4167	R5000 | R8071	D0 | D4095		V、Z P0～P9
D	○	○	○	○	○	○	○	○	○	○	○	○	○	○		○
N									○				○	○	1-511	○

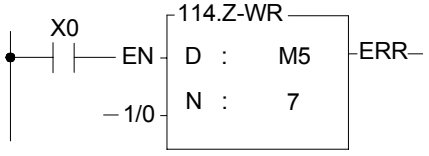
- 当执行控制〝EN〞=1或〝EN↑〞（P指令）由0→1时，将以D为起始的N个缓存器，依据写入选择(1或0)，将其区域数据覆盖。
- 当N的长度设定不正确时(N=0或N>511)，则错误旗标"ERR"设定为1。

程序范例一:



• 上图程序范例当X0 "ON"时，将R0~R9写入为0。

程序范例二:

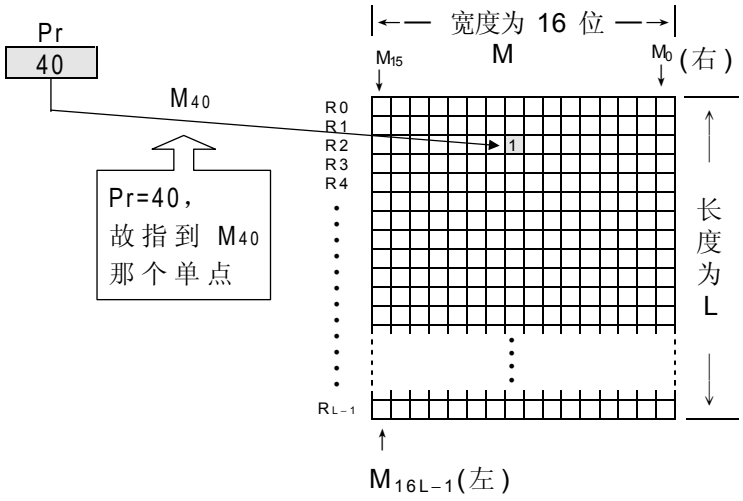


• 上图程序范例当X0 "ON"时，将M5~M11清除为0。

矩　阵　（MATRIX）　指　令

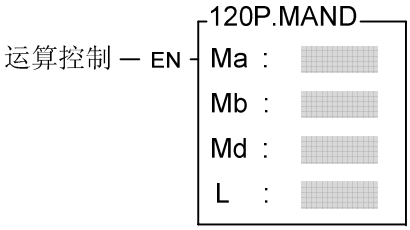
120. MAND
121. MOR
122. MXOR
123. MXNR
124. MINV
125. MCMP
126. MBRD
127. MBWR
128. MBSHF
129. MBROT
130. MBCNT

- 矩阵是 2 个以上连续的 16 位缓存器所组成，组成矩阵的缓存器个数称为矩阵的长度 L，一个矩阵共有 L×16 个位（点），其运算单位一次只有一个位（点）。
- 矩阵指令是将 16×L 个矩阵位（序号由 M_0～M_{16L-1}）当作一连串单点的集合，而从该集合中指定某一个单点作运作，而不将它当作数值看待。
- 矩阵指令主要在处理单点对多点（矩阵）或多点对多点的状态处理，如搬移、拷贝、比较、搜寻等，是极为方便和重要的应用指令。
- 在矩阵指令运作中，通常需要有一个 16 位缓存器来指定矩阵中 16L 个单点的某个单点当作运算对象，此缓存器称为矩阵的指针 Pr（Pointer），它的有效范围为 0～16L–1，分别对应到矩阵中的位 M_0～M_{16L-1}。
- 矩阵运作中有左、右位移或旋转，我们定义高序号的为左，低序号的为右，如下图示。



FUN120 P MAND	矩阵逻辑及（AND）运算 （MATRIX AND）	FUN120 P MAND

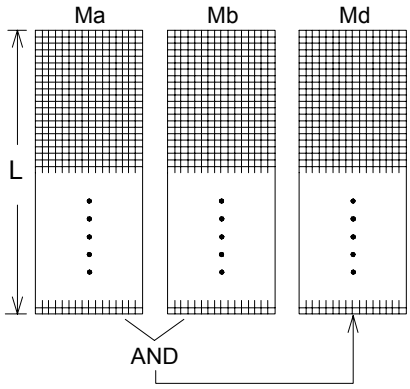
阶梯图符号

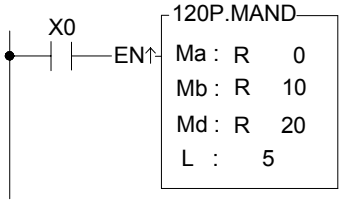


Ma：来源矩阵 a 的起头缓存器号码
Mb：来源矩阵 b 的起头缓存器号码
Md：存放结果的目的矩阵起头缓存器号码
L ：矩阵（Ma、Mb 和 Md）的长度
Ma，Mb，Md 可结合 V、Z、P0~P9 作间接寻址应用

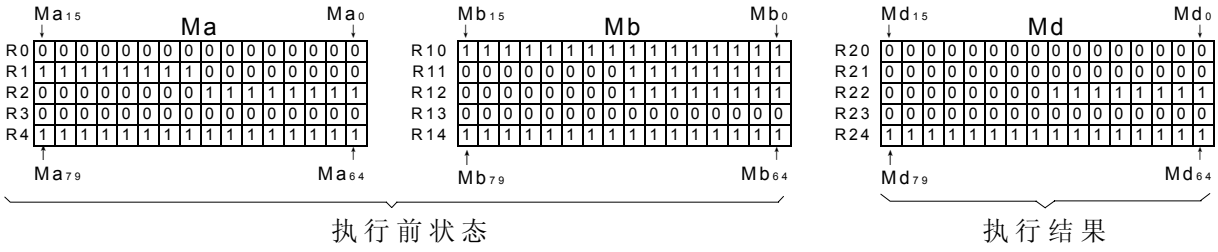
操作数 \ 范围	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR
	WX0 | WX240	WY0 | WY240	WM0 | WM1896	WS0 | WS984	T0 | T255	C0 | C255	R0 | R3839	R3840 | R3903	R3904 | R3967	R3968 | R4167	R5000 | R8071	D0 | D4095	2 | 256	V、Z P0~P9
Ma	○	○	○	○	○	○	○	○	○	○	○	○		○
Mb	○	○	○	○	○	○	○	○	○	○	○	○		○
Md		○	○	○	○	○	○		○	○*	○*	○		○
L							○				○*	○	○	

● 当运算控制〝EN〞=1 或〝EN↑〞（P 指令）由 0→1 时，将长度为 L 的两来源矩阵 Ma 和 Mb 整个作逻辑 AND 运算（两位都为 1 结果始为 1，否则为 0）后，再将结果存到长度同为 L 的目的矩阵 Md 去（相同序号的位作 AND，例如 Ma_0=0，Mb_0=1，则 Md_0=0；Ma_1=1，Mb_1=1 则 Md_1=1；……一直 AND 到 Ma_{16L-1} 和 Mb_{16L-1} 止）。



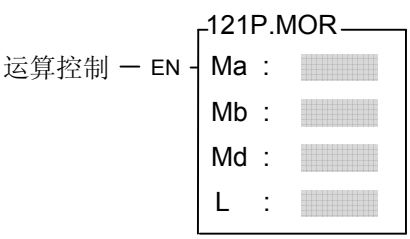


• 左图程序范例，当 X0 由 0→1 时将 R0～R4 构成的 Ma 和 R10～R14 构成的 Mb 作 AND 后，将结果存到由 R20～R24 所构成的 Md 去，其执行结果如下图右。



FUN121 P MOR	矩阵逻辑或（OR）运算 （MATRIX OR）	FUN121 P MOR

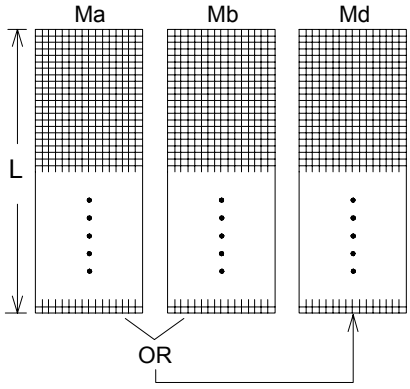
阶梯图符号



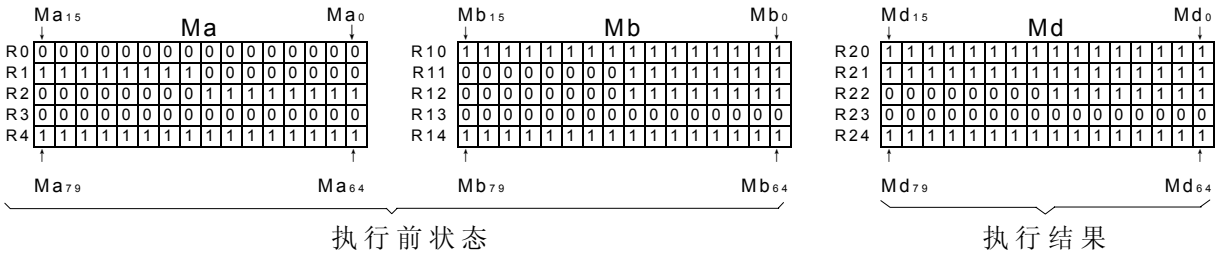
Ma：来源矩阵 a 的起头缓存器号码
Mb：来源矩阵 b 的起头缓存器号码
Md：存放结果的目的矩阵起头缓存器号码
L ：矩阵（Ma、Mb 和 Md）的长度
Ma，Mb，Md 可结合 V、Z、P0~P9 作间接寻址应用

操作数 \ 范围	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR
	WX0 | WX240	WY0 | WY240	WM0 | WM1896	WS0 | WS984	T0 | T255	C0 | C255	R0 | R3839	R3840 | R3903	R3904 | R3967	R3968 | R4167	R5000 | R8071	D0 | D4095	2 | 256	V、Z P0~P9
Ma	○	○	○	○	○	○	○	○	○	○	○	○		○
Mb	○	○	○	○	○	○	○	○	○	○	○	○		○
Md		○	○	○	○	○	○		○	○*	○*	○		○
L							○				○*	○	○	

● 当运算控制〝EN〞=1 或〝EN↑〞（P 指令）由 0→1 时，将长度为 L 的两来源矩阵 Ma 和 Mb 整个作逻辑 OR 运算（两位有任一个为 1 则结果为 1，两者都为 0 结果才为 0）后，再将结果存回长度同为 L 的目的矩阵 Md 去。（相同序号的位作 OR，例如 M---$_{0}$$_{-}$=0，$Mb_0$=1，则 Md_0=1；Ma_1=0，Mb_1=0 则 Md_1=0；……一直 OR 到 Ma_{16L-1} 和 Mb_{16L-1} 止）。

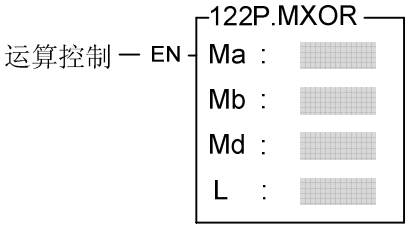


• 左图程序范例，当 X0 由 0→1 时，将 R0～R4 构成的 Ma 和由 R10～R14 构成的 Mb 作 OR 运算后，将结果存回由 R10～R14 构成的目的矩阵 Md 去，本例因 Mb 和 Md 为同一个矩阵，故运算后来源矩阵 Mb 已被新值覆盖，如下右图的结果。



FUN122 P MXOR	矩阵逻辑互斥或（XOR）运算 （MATRIX EXCLUSIVE OR）	FUN122 P MXOR

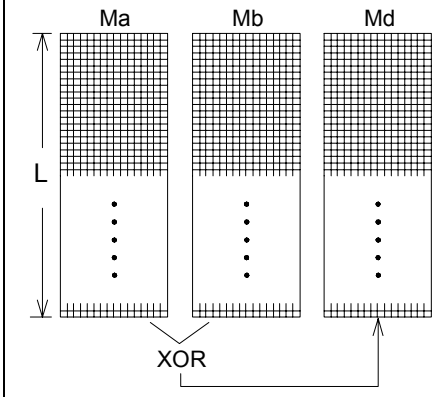
阶梯图符号



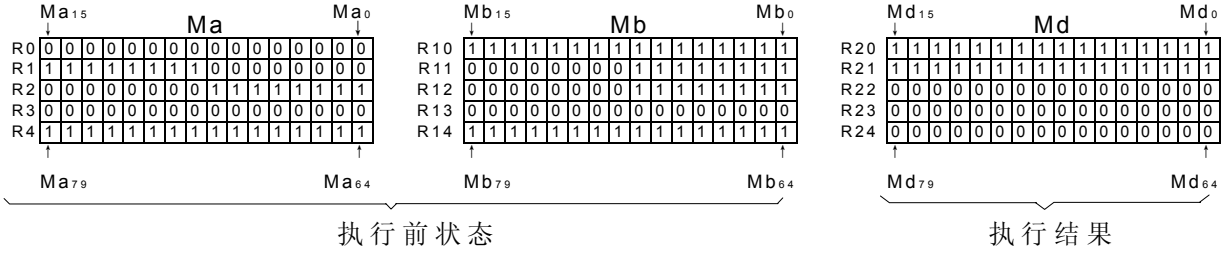
Ma：来源矩阵 a 的起头缓存器号码
Mb：来源矩阵 b 的起头缓存器号码
Md：存放结果的目的矩阵起头缓存器号码
L ：矩阵（Ma，Mb，Md）的长度
Ma，Mb，Md 可结合 V、Z、P0~P9 作间接寻址应用

操作数 \ 范围	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR
	WX0 | WX240	WY0 | WY240	WM0 | WM1896	WS0 | WS984	T0 | T255	C0 | C255	R0 | R3839	R3840 | R3903	R3904 | R3967	R3968 | R4167	R5000 | R8071	D0 | D4095	2 | 256	V、Z P0~P9
Ma	○	○	○	○	○	○	○	○	○	○	○	○		○
Mb	○	○	○	○	○	○	○	○	○	○	○	○		○
Md		○	○	○	○	○	○		○	○*	○*	○		○
L							○				○*	○	○	

● 当运算控制〝EN〞=1 或〝EN↑〞（P 指令）由 0→1 时，将长度为 L 的两来源矩阵 Ma 和 Mb 整个作逻辑 XOR 运算（两位不同结果为 1，否则为 0）后，再将结果存回长度同为 L 的目的矩阵 Md 去。（相同序号的位作 XOR，例如 Ma_0=0，Mb_0=1，则 Md_0=1；Ma_1=1，Mb_1=1 则 Md_1=0；……一直 XOR 到 Ma_{16L-1} 和 Mb_{16L-1} 止）。

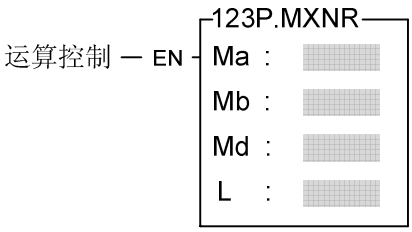


• 左图程序范例，当 X0 由 0→1 时，将 R0～R4 构成的 Ma 和由 R10～R14 构成的 Mb 作 XOR 运算后，将结果存到由 R20～R24 构成的目的矩阵 Md 去，其运算结果如下图右。



FUN123 P MXNR	矩阵互容或（XNR）运算 （MATRIX ENCLUSIVE OR）	FUN123 P MXNR

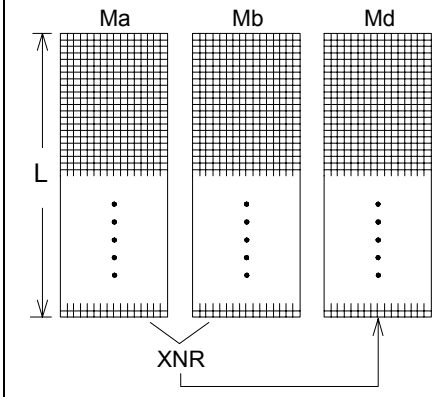
阶梯图符号

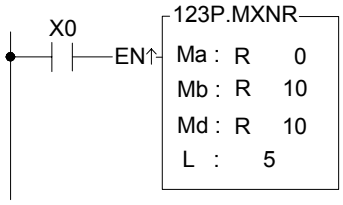


Ma：来源矩阵 a 的起头缓存器号码
Mb：来源矩阵 b 的起头缓存器号码
Md：存放结果的目的矩阵起头缓存器号码
L ：矩阵（Ma，Mb，Md）的长度
Ma，Mb，Md 可结合 V、Z、P0~P9 作间接寻址应用

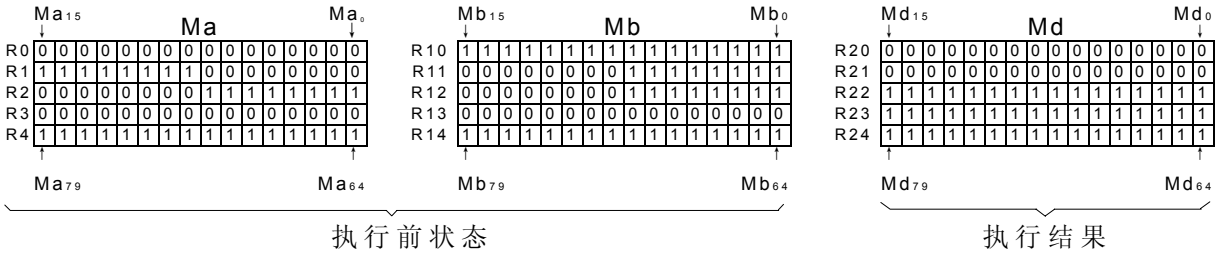
操作数 \ 范围	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR
	WX0 | WX240	WY0 | WY240	WM0 | WM1896	WS0 | WS984	T0 | T255	C0 | C255	R0 | R3839	R3840 | R3903	R3904 | R3967	R3968 | R4167	R5000 | R8071	D0 | D4095	2 | 256	V、Z P0~P9
Ma	○	○	○	○	○	○	○	○	○	○	○	○		○
Mb	○	○	○	○	○	○	○	○	○	○	○	○		○
Md		○	○	○	○	○	○		○	○*	○*	○		○
L							○				○*	○	○	

● 当运算控制〝EN〞=1 或〝EN↑〞（P 指令）由 0→1 时，将长度为 L 的两来源矩阵 Ma 与 Mb 整个作逻辑 XNR 运算（两位相同则结果为 1，否则为 0）后，再将结果存到长度同为 L 的目的矩阵 Md 去（相同序号的位作 XNR，例如 Ma_0=0，Mb_0=1，则 Md_0=0；Ma_1=0，Mb_1=0 则 Md_1=1；……一直 XNR 到 Ma_{16L-1} 和 Mb_{16L-1} 止）。



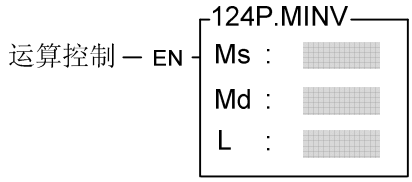


- 左图程序范例，当 X0 由 0→1 时，将 R0～R4 所构成的来源矩阵 Ma 和由 R10～R14 构成的来源矩阵 Mb 作 XNR 运算后，将结果存回由 R10～R14 所构成的 Md 去，本例因 Mb 和 Md 为同一个矩阵，故运算后来源矩阵 Mb 已被新值所覆盖，如下右图的结果。



FUN124 P MINV	矩阵倒相 （MATRIX INVERSE）	FUN124 P MINV

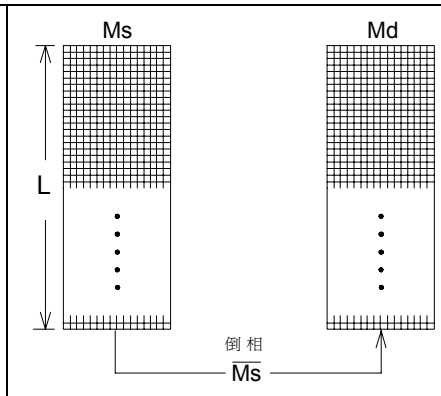
阶梯图符号

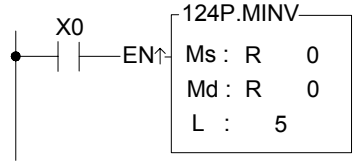


Ms：来源矩阵的起头缓存器号码
Md：存放结果的目的矩阵起头缓存器号码
L ：矩阵（Ms 和 Md）的长度
Ms，Md 可结合 V、Z、P0~P9 作间接寻址应用

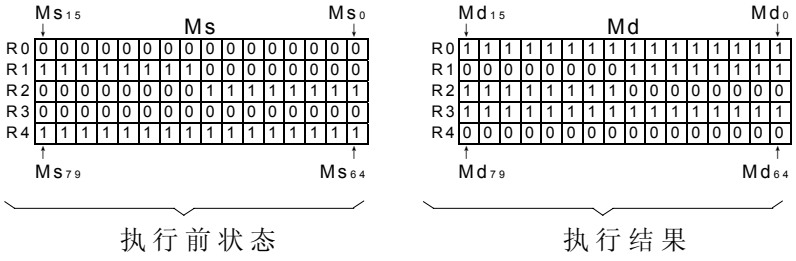
操作数 \ 范围	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR
	WX0 | WX240	WY0 | WY240	WM0 | WM1896	WS0 | WS984	T0 | T255	C0 | C255	R0 | R3839	R3840 | R3903	R3904 | R3967	R3968 | R4167	R5000 | R8071	D0 | D4095	2 | 256	V、Z | P0~P9
Ms	○	○	○	○	○	○	○	○	○	○	○	○		○
Md		○	○	○	○	○	○		○	○*	○*	○		○
L							○				○*	○	○	

● 当运算控制〝EN〞=1 或〝EN↑〞（P 指令）由 0→1 时，将长度为 L 的来源矩阵 Ms 整个反相（所有状态为 1 的位变成 0，而状态为 0 的则变为 1）后，再存到目的矩阵 Md 中去。





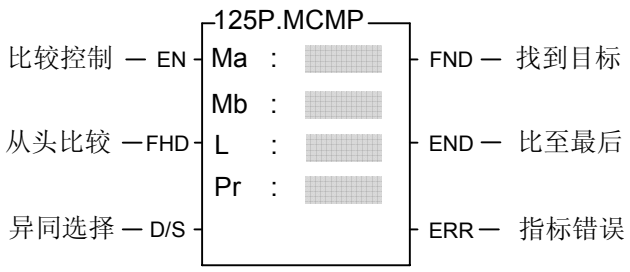
• 左图程序范例，当 X0 由 0→1 时，将 R0～R4 所组成的矩阵反相后存回自己（因本例 Ms 和 Md 为同一个矩阵）。所获得的结果如下图右。



执行前状态　　执行结果

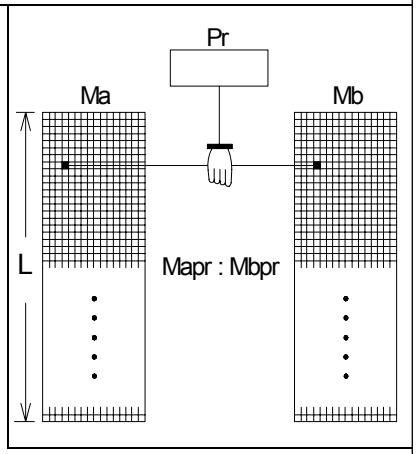
FUN125 P MCMP	矩阵对矩阵比较异同 （MATRIX COMPARE）	FUN125 P MCMP

阶梯图符号

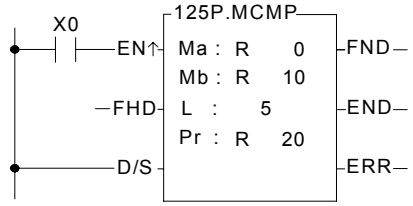


Ma：矩阵 a 的起头缓存器号码
Mb：矩阵 b 的起头缓存器号码
L ：矩阵（Ma，Mb）的长度
Pr ：指针，用来存放目标的位置值
Ma，Mb 可结合 V、Z、P0~P9 作间接定址应用

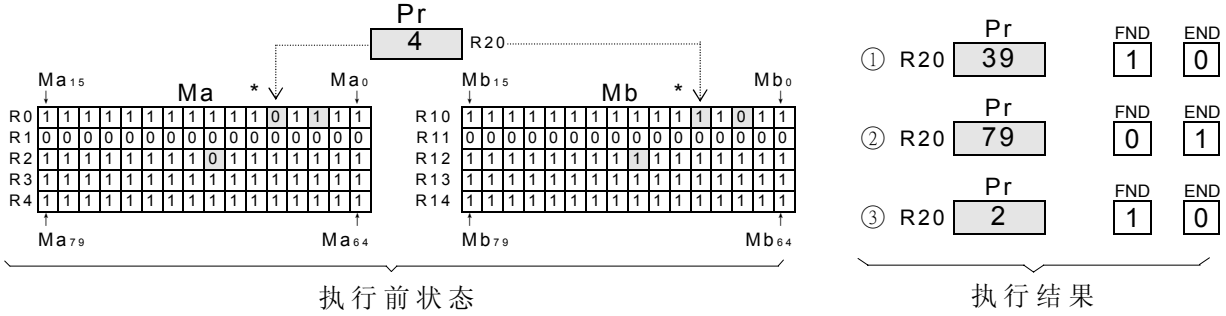
操作数 \ 范围	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR
	WX0 | WX240	WY0 | WY240	WM0 | WM1896	WS0 | WS984	T0 | T255	C0 | C255	R0 | R3839	R3840 | R3903	R3904 | R3967	R3968 | R4167	R5000 | R8071	D0 | D4095	2 | 256	V、Z P0~P9
Ma	○	○	○	○	○	○	○	○	○	○	○	○		○
Mb	○	○	○	○	○	○	○	○	○	○	○	○		○
L							○				○*	○	○	
Pr		○	○	○	○	○	○		○	○*	○*	○		



● 当比较控制〝EN〞=1 或〝EN↑〞（P 指令）由 0→1 时，从 Ma 和 Mb 两矩阵中的最开头那对位（Ma_0 和 Mb_0）开始（〝FHD〞=1 或 Pr 值已等于 16L-1 时）或从当时指标 Pr 所指那对位的下一对位（Ma_{pr+1} 和 Mb_{pr+1}）开始（〝FHD〞=0 同时 Pr 值小于 16L–1）往下双双成对比较寻找状态不同（D/S=1 时）或相同（D/S=0 时）的位对（Pair），当找到目标（状态不同或相同的位对）后立即停止比较动作，同时将该对目标在矩阵中的位置序号值存到指标 Pr 去，并将找到目标旗号〝FND〞设为 1 后结束本指令的执行。当找到矩阵的最后一对位（Ma_{16L-1}，Mb_{16L-1}）时，无论它是否为要找寻的目标都将结束该次的比较寻找动作，并将比到最后旗号〝END〞设为 1，而 Pr 值则停在 16L–1。当下次本指令再度被执行时，Pr 将会自动循环到矩阵的最开头（Pr=0）处开始往下比较。指标值的范围为 0～16L-1，在运作中应避免变动到 Pr 值，以免影响其正确的比较寻找，如果 Pr 值超出此范围则指标错误旗号〝ERR〞设为 1，而且本指令不执行。

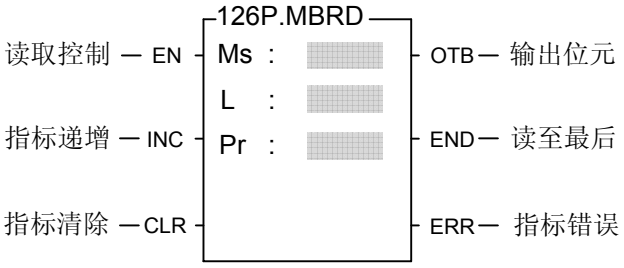


● 左图程序范例，因〝FHD〞输入为 0，故由指标当时值加 1 处（标注*处）开始往下比较寻找位状态不同（因 D/S=1 为找不同）的。当 X0 由 0→1 动作 3 次，可得到如下图右①，②，③三个执行结果。



FUN126 P MBRD	矩阵位读取 （MATRIX BIT READ）	FUN126 P MBRD

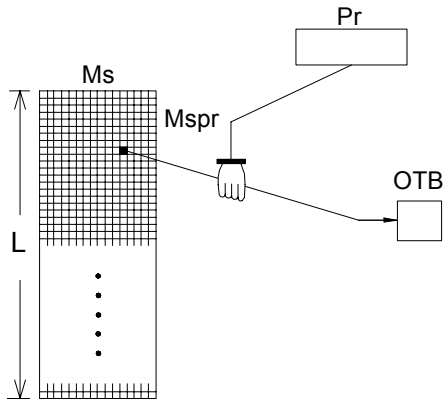
阶梯图符号



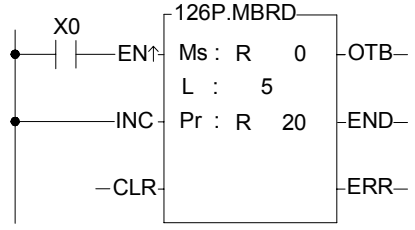
Ms：矩阵的起头缓存器号码
L ：矩阵的长度
Pr ：指针缓存器号码
Ms 可结合 V、Z、P0~P9 作间接寻址应用

操作数 \ 范围	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR
	WX0 | WX240	WY0 | WY240	WM0 | WM1896	WS0 | WS984	T0 | T255	C0 | C199	R0 | R3839	R3840 | R3903	R3904 | R3967	R3968 | R4167	R5000 | R8071	D0 | D4095	2 | 256	V、Z P0~P9
Ms	○	○	○	○	○	○	○	○	○	○	○	○		○
L							○				○*	○	○	
Pr		○	○	○	○	○	○		○	○*	○*	○		

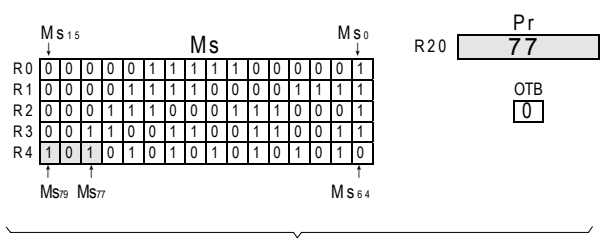
● 当读取控制〝EN〞=1 或〝EN↑〞（P 指令）由 0→1 时，读取矩阵 Ms 中指标 Pr 所指的那个位 Ms_{pr} 的状态并将它送到输出位〝OTB〞去。在读取前会先去检视指标清除〝CLR〞的状态，如果〝CLR〞为 1，则会先将 Pr 清为 0 后再作读取动作。在读取完毕后接着检视 Pr 值，如果 Pr 值已达 16L–1（指到最后一个位），则将读到最后旗号〞END〞设为 1，然后结束本指令的执行，如果 Pr 小于 16L–1，则再查看指标递增〝INC〞的状态，如果〝INC〞为 1 则将指标 Pr 值加 1 后才结束本指令的执行。此外，指标清除〝CLR〞可单独执行，不受其它输入影响。



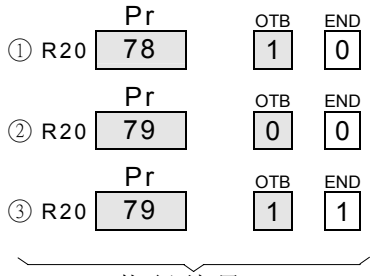
● 指标的有效范围为 0～16L–1，超出此范围则将指标错误旗号〝ERR〞设为 1，且本指令不执行。



· 左图程序范例，因 INC=1，故每读取一次，指标即加 1，如此可连续读取 Ms 中的每一位，如下图左的状态，当 X0 由 0→1 动作 3 次后，可得到下图右①，②，③三个执行结果。



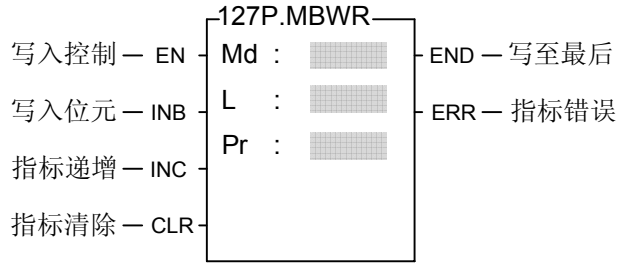
執行前状态



執行结果

FUN127 P MBWR	矩阵位写入 （MATRIX BIT WRITE）	FUN127 P MBWR

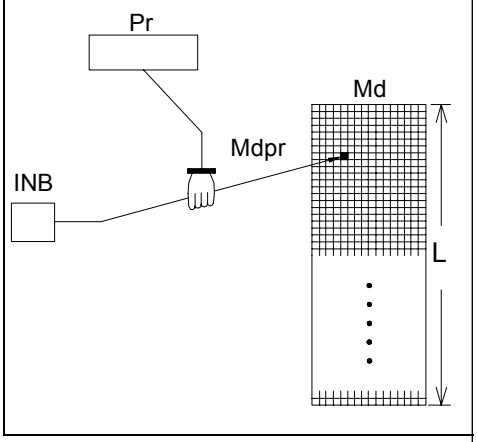
阶梯图符号



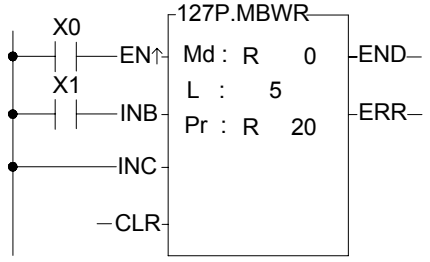
Md：矩阵的起头缓存器号码
L ：矩阵的长度
Pr：指针缓存器的号码
Md 可结合 V、Z、P0~P9 作间接寻址应用

操作数 \ 范围	WY	WM	WS	TMR	CTR	HR	OR	SR	ROR	DR	K	XR
	WY0 | WY240	WM0 | WM1896	WS0 | WS984	T0 | T255	C0 | C255	R0 | R3839	R3904 | R3967	R3968 | R4167	R5000 | R8071	D0 | D4095	2 | 256	V、Z P0~P9
Md	○	○	○	○	○	○	○	○	○	○		○
L						○			○*	○	○	
Pr	○	○	○	○	○	○	○	○*	○*	○		

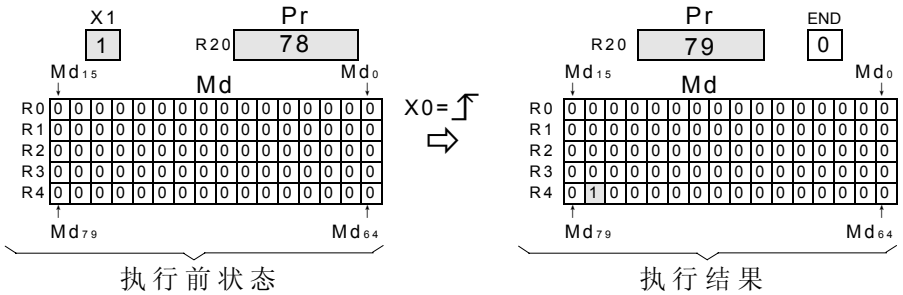
● 当写入控制〝EN〞=1 或〝EN↑〞（P 指令）由 0→1 时，将写入位〝INB〞的状态写到矩阵 Md 中指标 Pr 所指的那个位 Md_{pr} 去。在写入之前会先去检视指标清除〝CLR〞的状态，如果〝CLR〞为 1，则会先将 Pr 清为 0 后再作写入动作。在执行完写入动作后接着检视指标 Pr 的值，如果 Pr 值已达 16L–1（指到最后位），则将写到最后旗号〝END〞设为 1 后结束该次执行。如果 Pr 值小于 16L–1，则再检视指标递增输入〝INC〞的状态，如果〝INC〞为 1 则将 Pr 值加 1 后才结束执行。此外，指标清除〝CLR〞能单独执行，不受其它输入影响。



● Pr 的有效范围为 0～16L–1，超出该范围则指标错误旗号〝ERR〞设为 1，且本指令不执行。



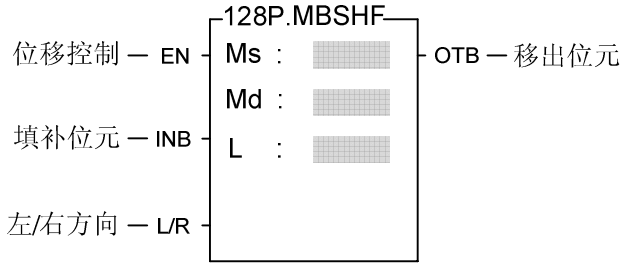
• 左图程序范例，指针每次执行后都会递增（因〝INC〞为 1）。如下图示，当 X0 由 0→1 动作一次后，INB 的状态（X1）即被写到 Md_{pr}（即 Md_{78}）处，且指标 Pr 加 1（变为 79）。此时虽 Pr 已指到最后但尚未写入 Md_{79}，故〝END〞仍为 0，须等到下次动作真正写入 Md_{79} 后，〝END〞才会变为 1。



执行前状态　　执行结果

FUN128 P MBSHF	矩阵位位移 （MATRIX BIT SHIFT）	FUN128 P MBSHF

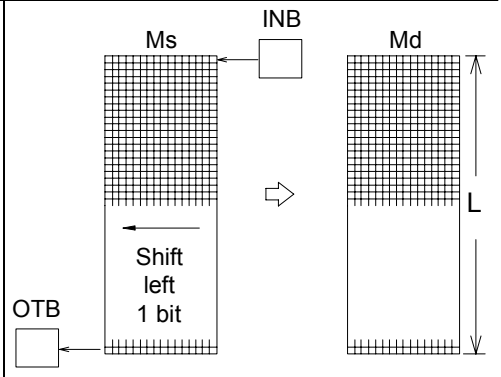
阶梯图符号

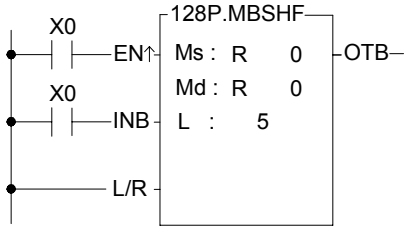


Ms：来源矩阵的起头缓存器号码
Md：目的矩阵的起头缓存器号码
L ：矩阵（Ms 和 Md）长度
Ms，Md 可结合 V、Z、P0~P9 作间接寻址应用

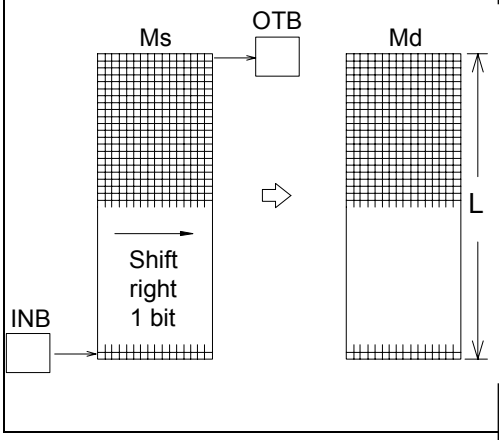
操作数 \ 范围	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR
	WX0 | WX240	WY0 | WY240	WM0 | WM1896	WS0 | WS984	T0 | T255	C0 | C255	R0 | R3839	R3840 | R3903	R3904 | R3967	R3968 | R4167	R5000 | R8071	D0 | D4095	2 | 256	V、Z P0~P9
Ms	○	○	○	○	○	○	○	○	○	○	○	○		○
Md		○	○	○	○	○	○		○	○*	○*	○		○
L											○*	○	○	

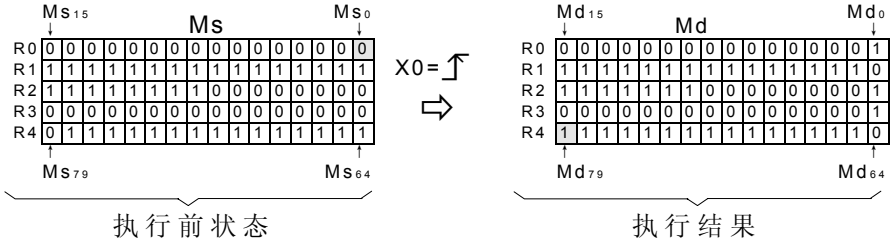
● 当位移控制〝EN〞=1 或〝EN↑〞（P 指令）由 0→1 时，将矩阵 Ms 整个取出向左（L/R=1 时）或向右（L/R=0 时）位移一个位置，因位移而腾出的空位（左移时为 M_0，右移时为 M_{16L-1}）则以填补位〝INB〞的状态填补。而因位移而挤出的位（左移时为 M_{16L-1}，右移时为 M_0）状态则送到移出位〝OTB〞去，然后再将此位移过的矩阵结果填入目的矩阵 Md 中去。





• 上图程序范例的 Ms 和 Md 为同一个矩阵的范例，当 X0 由 0→1 动作时，将 Ms 整个取出作左移（因 L/R=1）一位后，再存回 Md 而得到如下图右的结果。

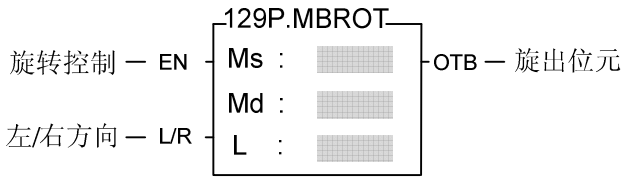




执行前状态　　执行结果

FUN129 P MBROT	矩阵位旋转 （MATRIX BIT ROTATE）	FUN129 P MBROT

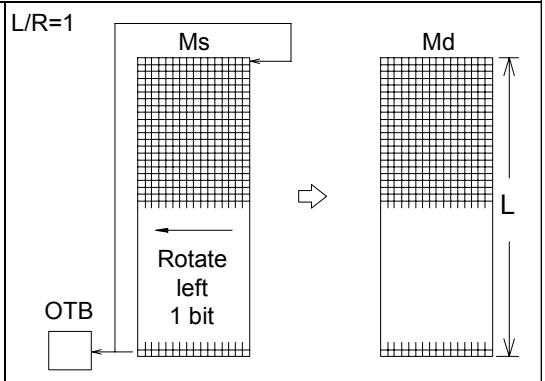
阶梯图符号

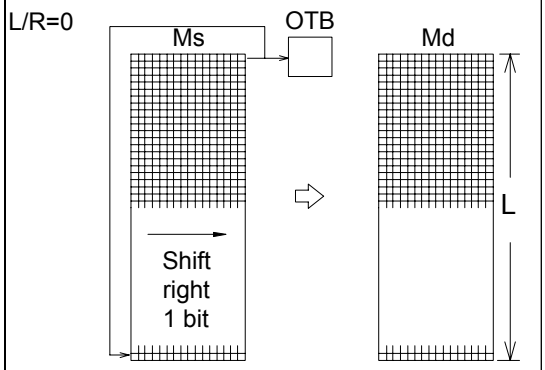


Ms：来源矩阵的起头缓存器号码
Md：目的矩阵的起头缓存器号码
L ：矩阵（Ms 和 Md）长度
MS，Md 可结合 V、Z、P0~P9 作间接寻址应用

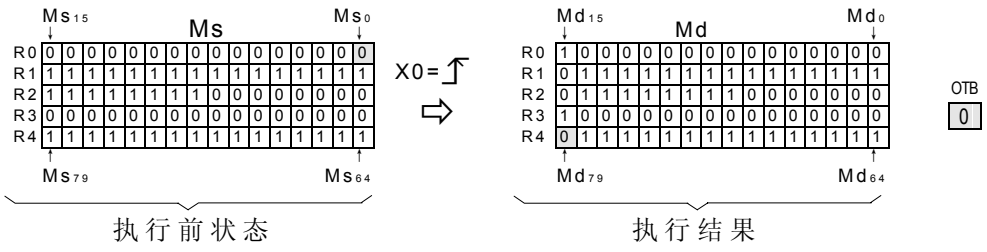
操作数 \ 范围	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR
	WX0 | WX240	WY0 | WY240	WM0 | WM1896	WS0 | WS984	T0 | T255	C0 | C255	R0 | R3839	R3840 | R3903	R3904 | R3967	R3968 | R4167	R5000 | R8071	D0 | D4095	2 | 256	V、Z | P0~P9
Ms	○	○	○	○	○	○	○	○	○	○	○	○		○
Md		○	○	○	○	○	○		○	○*	○*	○		○
L											○*	○	○	

● 当旋转控制〝EN〞=1 或〝EN↑〞（P 指令）由 0→1 时，将矩阵 Ms 整个取出向左（L/R=1 时）或向右（L/R=0 时）旋转一位，因旋转造成的空位（左旋时为 M_0，右旋时为 M_{16L-1}）由旋出位（左旋时为 M_{16L-1} 右旋时为 M_0）状态填补。再将该经旋转后的结果填入 Md 中去。旋出位不但用来填补前述的空位，同时并将它送到旋出位〝OTB〞去。



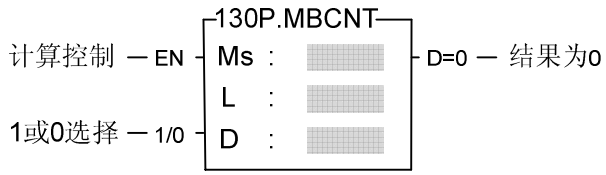


• 上图程序范例的 Ms 和 Md 为同一个矩阵，当 X0 由 0→1 动作时，将整个 Ms 取出向右旋转（因 L/R=0）一位后再存回到 Ms 自己（因本例 Ms 和 Md 为同一个矩阵），而得到如下图右的结果。



FUN130 P MBCNT	矩阵位状态数量计算 （MATRIX BIT STATUS COUNT）	FUN130 P MBCNT

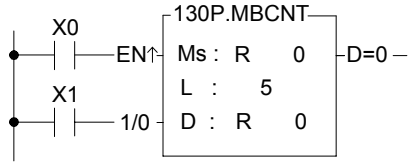
阶梯图符号



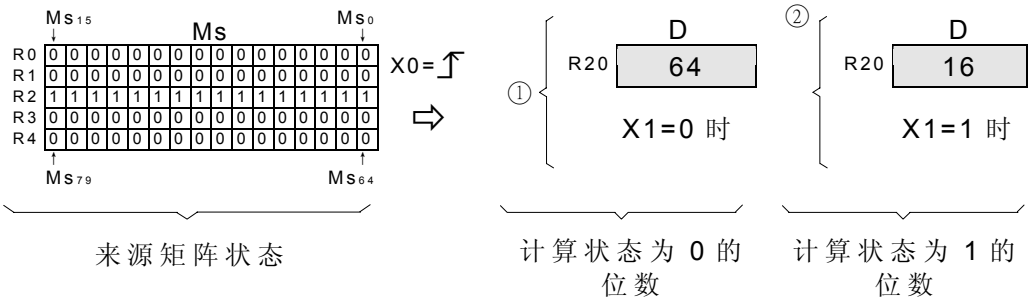
Ms：矩阵的起头缓存器号码
L ：矩阵的长度
D ：存放数量结果的缓存器号码
Ms 可结合 V、Z、P0~P9 作间接寻址应用

操作数 \ 范围	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR
	WX0 | WX240	WY0 | WY240	WM0 | WM1896	WS0 | WS984	T0 | T255	C0 | C255	R0 | R3839	R3840 | R3903	R3904 | R3967	R3968 | R4167	R5000 | R8071	D0 | D4095	2 | 256	V、Z | P0~P9
Ms	○	○	○	○	○	○	○	○	○	○	○	○		○
L							○				○*	○	○	
D		○	○	○	○	○	○		○	○*	○*	○		

● 当计算控制〝EN〞=1 或〝EN↑〞（P 指令）由 0→1 时，统计矩阵 Ms 的 16L 个位中，所有状态为〝1〞（ON）的位总数目（1 或 0 选择输入〝1/0〞=1 时）或所有状态为〝0〞（OFF）的位总数（1 或 0 选择输入〝1/0〞=0 时）。再将统计所得的数目结果存到 D 所指定的缓存器去，如果该数目值为 0，则将结果为 0 旗号〝D=0〞设为 1。

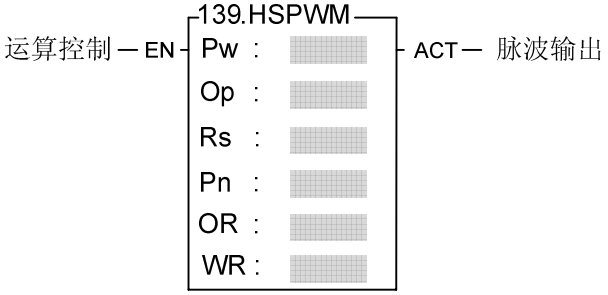


• 左图程序范例分别将 X1 设为 0（统计状态为 0 的位数）及设为 1（统计状态为 1 的）两种条件，再分别在这两条件下使 X0 由 0→1 各一次，可获得如下图右①，②两个执行结果。



FUN139 HSPWM	高速脉冲宽度调变 （HIGH SPEED PULSE WIDTH MODULATION）	FUN139 HSPWM

阶梯图符号



Pw：高速脉冲宽度调变输出点
(0=Y0，1=Y2，2=Y4，3=Y6)
Op：输出极性；0=输出不倒相
1=输出倒相
Rs：分辨率；0=1/100 (1%)
1=1/1000 (0.1%)
Pn：输出频率参数设定(0~255)
OR：PWM 输出宽度设定缓存器 0~100 或 0~1000

WR：指令运作工作缓存器，其它程序不可重复使用

操作数 \ 范围	Y	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K
	主机上之 Yn	WX0 | WX240	WY0 | WY240	WM0 | WM1896	WS0 | WS984	T0 | T255	C0 | C255	R0 | R3839	R3840 | R3903	R3904 | R3967	R3968 | R4167	R5000 | R8071	D0 | D4095	
Pw	○													0～3
Op														0～1
Rs														0～1
Pn		○	○	○	○	○	○	○	○	○	○	○	○	0～255
OR								○				○	○	0～1000
WR			○	○	○	○	○	○		○	○	○	○	

- 第 0 点(Y0)与第 1 点(Y2)PWM 输出，其分辨率必须相同，同为 1/100 或 1/1000；而输出频率参数设定也必须相同，只有输出宽度(PLUSE WIDTH)可以不一样；同理第 2 点(Y4)与第 3 点(Y6)PWM 输出也如上述规定。
- 当执行控制〝EN〞=1 时，本指令所指定的输出点将按照下列公式所决定的频率以指定的脉冲宽度输出。

1. $f_{pwm} = \frac{184320}{(P_n + 1)}$ 当 Rs(分辨率)设定为 1/100 时

2. $f_{pwm} = \frac{18432}{(P_n + 1)}$ 当 Rs(分辨率)设定为 1/1000 时

程序范例一：假设 Pn(输出频率参数)设为 50，Rs(分辨率)=0 则

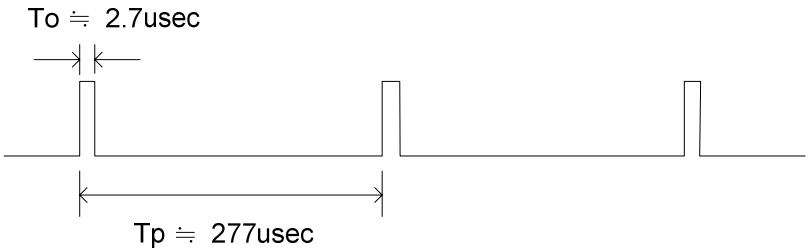
$$f_{pwm} = \frac{184320}{(50 + 1)} = 3614.117\cdots\cdots \fallingdotseq 3.6KHz$$

$$T(周期) = \frac{1}{f_{pwm}} \fallingdotseq 277uS$$

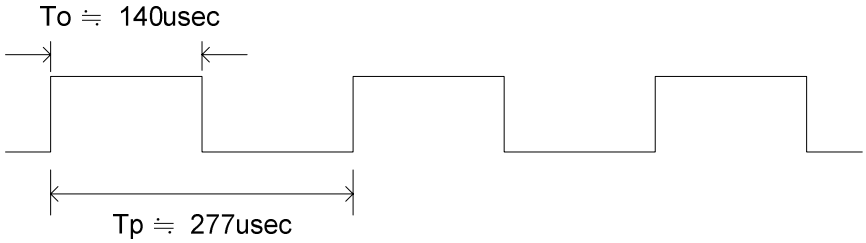
因为分辨率为 1/100，所以 OR(输出宽度)若为 1 则 To≒2.7uS，OR(输出宽度)若为 50 则 To≒140uS。图形如下:

(1).Pn(输出频率参数)=50，Rs=0(分辨率=1/100)，OR(输出宽度)=1 的输出波形：

FUN139 HSPWM	高速脉冲宽度调变 （HIGH SPEED PULSE WIDTH MODULATION）	FUN139 HSPWM



(2). Pn(输出频率参数)=50，Rs=0(分辨率=1/100)，OR(输出宽度)=50 的输出波形：

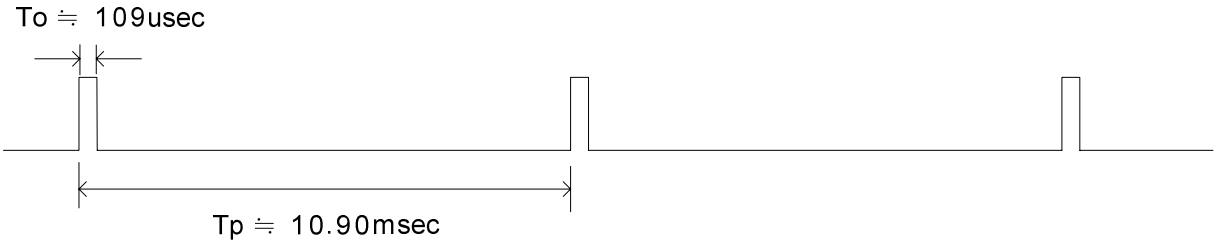


程序范例二：假设 Pn(输出频率参数)设为 200，Rs(分辨率)=1 则，

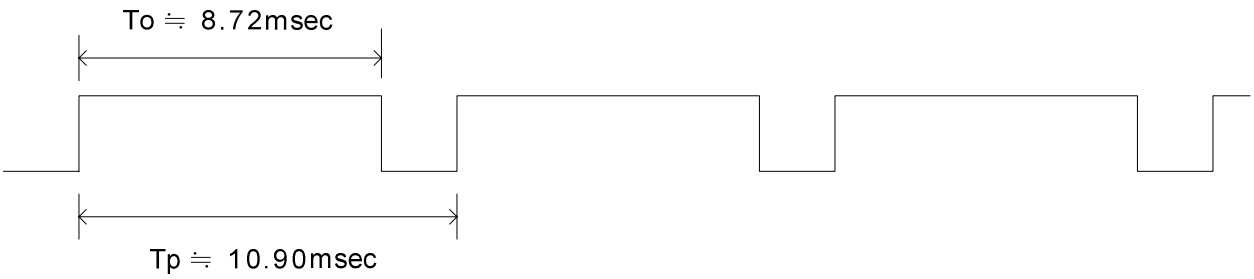
$$f_{pwm} = \frac{18432}{(200+1)} \fallingdotseq 91.7Hz \quad ; \quad T(周期) = \frac{1}{f_{pwm}} \fallingdotseq 10.9mS$$

因为 Rs(分辨率)为 1/1000，所以 OR(输出宽度)如果为 10 则 To≒109uS，OR(输出宽度)如果为 800 则 To≒8.72mS。图形如下:

(1).Pn(输出频率参数)=200，Rs=1(分辨率=1/1000) ，OR(输出宽度)=10 的输出波形：

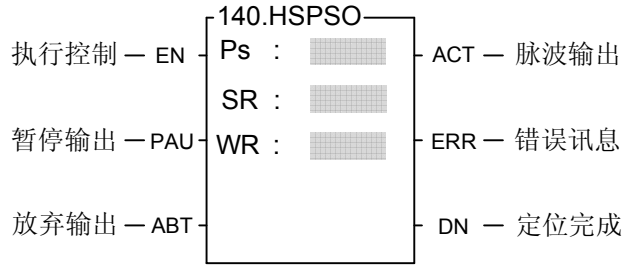


(2).Pn(输出频率参数)=200，Rs=1(分辨率=1/1000) ，OR(输出宽度)=800 的输出波形：



FUN140 HSPSO	高速脉冲输出（HSPSO）指令 （功能简述）	FUN140 HSPSO

阶梯图符号



Ps ： 第几组 Pulse Output （0～3）
0：Y0 & Y1
1：Y2 & Y3
2：Y4 & Y5
3：Y6 & Y7
SR ： 定位程序起始缓存器
WR： 指令运作起始缓存器，共占用 7 个缓存器，其它程序不可重复使用

操作数 \ 范围	HR	DR	ROR	K
	R0 ～ R3839	D0 ～ D4095	R5000 ～ R8071	2 ～ 256
Ps				0～3
SR	○	○	○	
WR	○	○	○*	

指令功能简述

- HSPSO（FUN140）指令的 NC 定位程序是以文字的程序书写方式来编辑；每一定位点我们称为一步（含输出频率、动作行程、转移条件），一个 FUN140 最多可编 250 步定位点，每一步定位点需占用 9 个缓存器。(详细的应用请参考第 13 章〝FBs-PLC 的 NC 定位控制)。
- 将定位程序存在缓存器最大好处是，如果结合人机作机台操作设定，则可将定位程序存入人机，更换模具时，可直接由人机操作存取当前模具的定位程序。
- 当执行控制输入〝EN〞=1 时，如 Ps0～3 没有被其它 FUN140 指令占用（Ps0=M1992，Ps1=M1993，Ps2=M1994，Ps3=M1995 的状态为 ON），则由下一步定位点开始执行（如果已到最后一步，则重新由第 1 步开始执行）；如果 Ps0～3 被其它 FUN140 指令占用（Ps0=M1992，Ps1=M1993，Ps2=M1994，Ps3=M1995 的状态为 OFF），则等占用的 FUN140 释出控制权，本指令取得定位控制的脉冲（Pulse）输出权。
- 当执行控制输入〝EN〞= 0 时，马上停止脉冲输出。
- 当暂停输出〝PAU〞= 1 ，而且执行控制〝EN〞原先为 1 时，则暂停脉冲输出；当暂停输出 〝PAU〞= 0，而执行控制〝EN〞仍为 1 时，继续输出未完成的脉冲数。
- 当放弃输出〝ABT〞= 1 时，马上停止脉冲输出。（下一次当执行控制输入〝EN〞= 1 时，重新由第一步定位点开始执行）。
- 当脉冲输出中，输出指示〝ACT〞ON。.
- 当指令执行错误时，输出指示〝ERR〞ON 。（错误代码存放在错误码缓存器）
- 当每一步定位点完成时，输出指示〝DN〞ON。

*** 务必设定 Pulse Output 的工作模式（不设定时，Y0～Y7 当作一般输出）为 U/D，K/R 或 A/B 等三种模式之一，Pulse Output 才能正常输出。

U/D 模式：Y0（Y2，Y4，Y6）送出上数脉冲
Y1（Y3，Y5，Y7）送出下数脉冲
K/R 模式：Y0（Y2，Y4，Y6）送出脉冲
Y1（Y3，Y5，Y7）送出方向信号；ON=上数，OFF=下数
A/B 模式：Y0（Y2，Y4，Y6）送出 A 向脉冲
Y1（Y3，Y5，Y7）送出 B 向脉冲

- Pulse Output 输出极性可选择 Normal ON 或 Normal OFF
- 在 WinProladder〝HSC〞设定页可设定 Pulse Output 的工作模式。

FUN141 MPARA	NC 定位参数值设定指令 （功能简述）	FUN141 MPARA

阶梯图符号

Ps：第几组 Pulse Output （0～3）

SR：参数表起始缓存器，共 18 个参数，占用 24 个缓存器

操作数 \ 范围	HR	DR	ROR	K
	R0 ｜ R3839	D0 ｜ D4095	R5000 ｜ R8071	2 ｜ 256
Ps				0 ～3
SR	○	○	○	

指令功能简述

- 本指令并不一定要使用；如果系统默认的参数值已符合用户需求，则可不必有此指令；如果需开放参数值作动态修改，则需要有此指令。
- 本指令配合 FUN140 作定位控制使用。
- 不管执行控制输入〝EN〞=0 或 1 时，本指令都会被执行。
- 当参数值有错误时，输出指示〝ERR〞ON。（错误代码存放在错误码缓存器）
- 详细功能叙述与使用方法请参考高级应用篇第 13 章〝FBs-PLC 的 NC 定位控制〞的说明。

FUN142 P PSOFF	强制停止 HSPSO 脉冲输出指令 （功能简述）	FUN142 P PSOFF

阶梯图符号

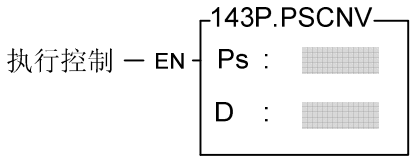
Ps：0～3
强制第几组 Pulse Output 停止输出

指令功能简述

- 当执行控制〝EN〞=1 或〝EN↑〞（P 指令）由 0→1 时，本指令将强制所指定的第几组 HSPSO（High Speed Pulse Output）停止脉冲输出。
- 在执行机械原点复归的应用时，当原点条件满足时，利用本指令可快速停止脉冲输出，让每次作机械原点复归时，都停在同一个位置。
- 详细功能叙述与使用方法请参考高级应用篇第 13 章〝FBs-PLC 的 NC 定位控制〞的说明。

FUN143 P PSCNV	目前脉冲值转换为显示值（mm，Deg，Inch，PS）指令（功能简述）	FUN143 P PSCNV

阶梯图符号



Ps：0～3；将第几组脉冲位置（PS）转换为与设定值同单位的 mm（Deg，Inch，PS），来作为目前位置显示.

D ：储存转换后目前位置的缓存器,共需要使用两个缓存器；例如 D10，即代表 D10（Low Word）与 D11（High Word）两个缓存器。

操作数 \ 范围	HR	DR	ROR	K
	R0 ～ R3839	D0 ～ D4095	R5000 ～ R8071	2 ～ 256
Ps				0 ～3
D	○	○	○	

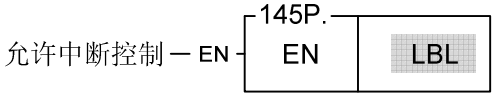
指令功能简述

- 当执行控制〝EN〞=1 或〝EN↑〞（P 指令）由 0→1 时，本指令将所指定的目前脉冲位置（PS）转换为与设定值同单位的 mm（或 Deg 或 Inch 或 PS），来作为目前位置显示。
- FUN140 指令执行后，本指令执行时，才会得到正确的转换值。
- 详细功能叙述与使用方法请参考第 13 章〝FBs-PLC 的 NC 定位控制〞的说明。

中断控制指令

FUN145 P EN	允许外界输入或外围中断作动指令	FUN145 P EN

阶梯图符号



LBL：允许中断作动的外界输入或外围标记名称。

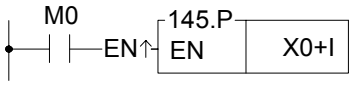
- 当允许控制〝EN〞=1 或〝EN↑〞（P 指令）由 0→1 时，允许 LBL 所指定的外界输入或外围中断作动。

- 可允许的中断标记名称如下：（详细请参考高级应用篇第 9.3 节的说明）

LBL 名称	叙　述	LBL 名称	中断控制点	LBL 名称	中断控制点
HSTAI	HSTA 高速定时器中断	X4+I	X4 正缘中断	X10+I	X10 正缘中断
HSC0I	HSC0 高速计数器中断	X4–I	X4 负缘中断	X10–I	X10 负缘中断
HSC1I	HSC1 高速计数器中断	X5+I	X5 正缘中断	X11+I	X11 正缘中断
HSC2I	HSC2 高速计数器中断	X5–I	X5 负缘中断	X11–I	X11 负缘中断
HSC3I	HSC3 高速计数器中断	X6+I	X6 正缘中断	X12+I	X12 正缘中断
X0+I	X0 正缘中断	X6–I	X6 负缘中断	X12–I	X12 负缘中断
X0–I	X0 负缘中断	X7+I	X7 正缘中断	X13+I	X13 正缘中断
X1+I	X1 正缘中断	X7–I	X7 负缘中断	X13–I	X13 负缘中断
X1–I	X1 负缘中断	X8+I	X8 正缘中断	X14+I	X14 正缘中断
X2+I	X2 正缘中断	X8–I	X8 负缘中断	X14–I	X14 负缘中断
X2–I	X2 负缘中断	X9+I	X9 正缘中断	X15+I	X15 正缘中断
X3+I	X3 正缘中断	X9–I	X9 负缘中断	X15–I	X15 负缘中断
X3–I	X3 负缘中断				

- 在实际应用上，有些中断信号在有些时候不可以让它发生作用，而在有些时候却必须让它发生作用；利用 FUN146（DIS）和 FUN 145（EN）指令就可以实现上述需求。

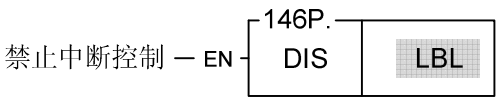
程序范例



- 当 M0 由 0→1 时，允许 X0 由 0→1 时发出中断；CPU 可立即快速处理 X0+I 的中断服务程序。

FUN146 P DIS	禁止外界输入或外围中断作动指令	FUN146 P DIS

阶梯图符号



LBL：禁止作动的外界输入或外围的中断标记名称。

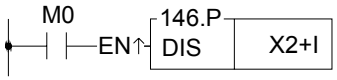
- 当禁止控制〝EN〞=1 或〝EN↑〞（P 指令）由 0→1 时，禁止 LBL 所指定的外界输入或外围中断或外围功能作动。

- 可以禁止的中断标记名称如下：（和可允许的一样）

LBL 名称	叙　述	LBL 名称	中断控制点	LBL 名称	中断控制点
HSTAI	HSTA 高速定时器中断	X4+I	X4 正缘中断	X10+I	X10 正缘中断
HSC0I	HSC0 高速计数器中断	X4–I	X4 负缘中断	X10–I	X10 负缘中断
HSC1I	HSC1 高速计数器中断	X5+I	X5 正缘中断	X11+I	X11 正缘中断
HSC2I	HSC2 高速计数器中断	X5–I	X5 负缘中断	X11–I	X11 负缘中断
HSC3I	HSC3 高速计数器中断	X6+I	X6 正缘中断	X12+I	X12 正缘中断
X0+I	X0 正缘中断	X6–I	X6 负缘中断	X12–I	X12 负缘中断
X0–I	X0 负缘中断	X7+I	X7 正缘中断	X13+I	X13 正缘中断
X1+I	X1 正缘中断	X7–I	X7 负缘中断	X13–I	X13 负缘中断
X1–I	X1 负缘中断	X8+I	X8 正缘中断	X14+I	X14 正缘中断
X2+I	X2 正缘中断	X8–I	X8 负缘中断	X14–I	X14 负缘中断
X2–I	X2 负缘中断	X9+I	X9 正缘中断	X15+I	X15 正缘中断
X3+I	X3 正缘中断	X9–I	X9 负缘中断	X15–I	X15 负缘中断
X3–I	X3 负缘中断				

- 在实际应用上，有些中断信号在有些时候不可以让它发生作用，即可利用此指令将该中断信号禁止而不会产生中断处理。

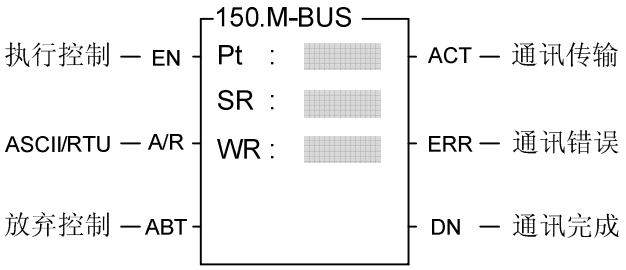
程序范例



- 当 M0 由 0→1 时，禁止 X2 由 0→1 时发出中断处理。

FUN150 M-BUS	Modbus RTU 通讯协议(主站)通讯联机便利指令 (使 PLC 经由 Port 1,2,3 或 4 当作 Modbus RTU 通讯协议的主站)	FUN150 M-BUS

阶梯图符号



Pt ：1～4，通过该通讯端口,以 Modbus RTU 通讯协议作数据传输

SR：通讯程序起始缓存器（见范例说明）

WR ：指令运作起始缓存器（见范例说明），共占用 8 个缓存器，其它程序不可重复使用。

操作数 \ 范围	HR	ROR	DR	K
	R0 | R3839	R5000 | R8071	D0 | D4095	
Pt				1～4
SR	○	○	○	
WR	○	○*	○	

- FUN150（M-BUS）指令提供永宏 PLC(主站)通过 Port 1,2,3 或 4 以 Modbus RTU 通讯协议与具有该通讯协议的智能型外围(仆站)联机。
- 一个通讯端口可经由 RS-485 接口最多与 247 台仆站联机互享数据。
- 被 FUN150 指定使用的通讯端口即为该 Modbus RTU 网络的主站。
- 利用程序书写方式或填表格方式来规划数据流控制；也就是要从哪一台仆站读取何种数据存放到主站(PLC)，或从主站(PLC)要写何种数据到仆站，只需要利用七个缓存器来定义，每七个缓存器定义一次传输交易。
- 当执行控制〝EN↑〞由 0→1 且放弃运作〝ABT〞为 0 时，若 Port 1,2,3 或 4 未被其它通讯指令占用[M1960(Port1),M1962(Port2),M1936(Port3)或 M1938(Port4)= 1]，则本指令立即控制 Port 1,2,3 或 4，并将 M1960,M1962,M1936 或 M1938 设为 0（表示占用），然后立即进行一笔数据传输交易。若 Port 1,2,3 或 4 已被占用（M1960,M1962,M1936 或 M1938 =0），则本指令进入等待状态，一直等到占用的通讯指令传送完毕或放弃运作，放出控制权（M1960,M1962,M1936 或 M1938=1）后，本指令立即脱离等待状态，将 M1960,M1962,M1936 或 M1938 设为 0 并立即进行传输交易。
- 在传输交易进行中，若放弃运作〝ABT〞变为 1，则本指令将立即停止传输，并放出控制权（将 M1960,M1962,M1936 或 M1938 设为 1）。当本指令回复执行，并再次控制 Port 1,2,3 或 4 时，会从头由第一笔数据开始传输。
- 〝A/R〞=0，Modbus RTU 通讯协议；〝A/R〞=1，Modbus ASCII 通讯协议（保留）。
- 当数据交易传输中，输出指示〝ACT〞ON。
- 当一笔数据交易传输完，如有错误发生，则输出指示〝DN〞与〝ERR〞同时 ON。
- 当一笔数据交易传输完，如无错误发生，则输出指示〝DN〞ON。
- 详细应用范例请参考高级应用篇第 12 章〝FBs-PLC LINK 功能的应用〞。

FUN151 CLINK	FUN151（CLINK）：通讯联机便利指令 （使 PLC 经由 Port 1,2,3 或 4 当作永宏通讯协议的主站）	FUN151 CLINK

阶梯图符号

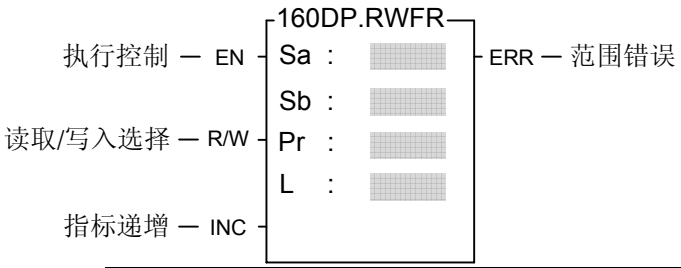
Pt ：指定通讯端口，1～4
MD：通讯模式选择(MD0~MD3)
SR：存放通讯程序起始缓存器
WR：指令运作起始缓存器，共占用 8 个缓存器，其它程序不可重复使用。

操作数 \ 范围	HR	ROR	DR	K
	R0 | R3839	R5000 | R8071	D0 | D4095	
Pt				1～4
MD				0～3
SR	○	○	○	
WR	○	○*	○	

- 本指令为 MD0~MD3 通用通讯联机便利指令，客户可以根据自己需求，指定通讯模式 (MD0~MD3)。
- FUN151（CLINK）：MD 0 模式提供永宏 PLC 与 PLC 之间的数据互享。
- 一台主 PLC 可经由 RS-485 接口最多与 254 台仆 PLC 联机互享数据。
- 仅主 PLC 需使用 CLINK 指令（设为梯形图指令控制界面），其它所有仆 PLC 都不必（设在标准界面）。
- 利用程序书写方式或填表格方式来规划数据流控制；也就是要从那一台仆 PLC 读取何种类型的数据存放到主 PLC，或从主 PLC 要写何种数据到仆 PLC，只需要利用七个缓存器来定义，每七个缓存器定义一笔传输交易。
- 当执行控制〝EN↑〞由 0→1 且暂停运作〝PAU〞与放弃运作〝ABT〞都为 0 时，若指定的通讯端口未被其它通讯指令所占用 [M1960(Port1),M1962(Port2),M1936(Port3)或 M1938(Port4)= 1]，则本指令立即控制该通讯端口，并将 M1960,M1962,M1936 或 M1938 设为 0（表示占用），然后立即进行一笔数据传输交易。若指定的通讯端口已被占用（M1960,M1962,M1936 或 M1938 =0），则本指令进入等待状态，一直等到占用的通讯指令传送完毕或暂停 / 放弃运作，放出控制权后（M1960,M1962,M1936 或 M1938=1），本指令立即脱离等待状态，将 M1960,M1962,M1936 或 M1938 设为 0，并立即进行传输交易。
- 在传输交易进行中，若暂停运作〝PAU〞变为 1，则本指令将在当时正在传输的那笔交易数据传输完毕后，暂停运作并释出控制权。而等到本指令恢复执行并再次控制传输权时，将会接续上次暂停传输的下一笔数据开始传输（也就是暂停是以一笔完整的交易数据为单位）。
- 传输交易进行中，若放弃运作〝ABT〞变为 1，则本指令将立即停止传输，并放出控制权。当本指令恢复执行，并再次控制通讯端口时，会重头由第一笔数据开始传输。
- 当数据交易传输中，输出指示〝ACT〞ON。
- 当一笔数据交易传输完，如有错误发生，则输出指示〝DN〞与〝ERR〞同时 ON。
- 当一笔数据交易传输完，如无错误发生，则输出指示〝DN〞ON。
- 详细应用范例请参考高级应用篇第 12 章〝FBs-PLC LINK 功能的应用〞。

FUN160D P RWFR	读/写档案缓存器 （Read/Write File Register）	FUN160D P RWFR

阶梯图符号



Sa：缓存器列表的起始缓存器号码

Sb：档案缓存器的起始号码

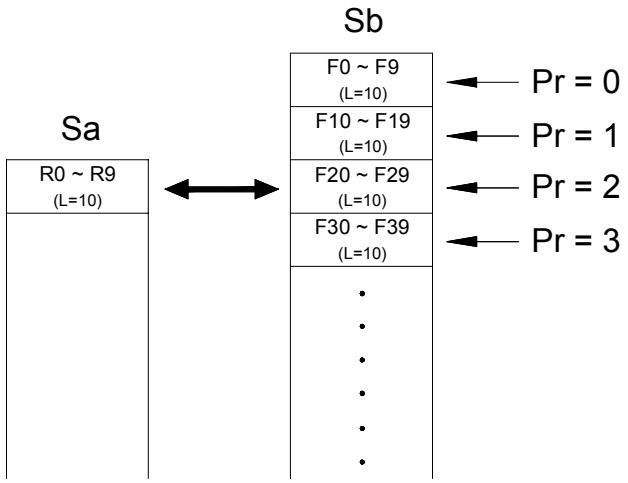
Pr：指针缓存器号码

L ：列表的长度 1~511

Sa 可结合 V、Z、P0~P9 作间接寻址应用

操作数 \ 范围	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR	FR
	WX0 | WX240	WY0 | WY240	WM0 | WM1896	WS0 | WS984	T0 | T255	C0 | C255	R0 | R3839	R3840 | R3903	R3904 | R3967	R3968 | R4167	R5000 | R8071	D0 | D4095		V、Z P0～P9	F0 | F8191
Sa	○	○	○	○	○	○	○	○	○	○	○	○		○	
Sb															○
Pr		○	○	○	○	○	○		○	○*	○*	○			
L							○				○*	○	1~511		

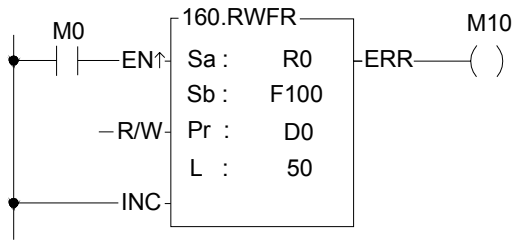
- 当执行控制〝EN〞=1 或〝EN↑〞（P 指令）由 0→1 时，从缓存器 Sa 开始，将长度 L 的数据按照读写控制〝R/W〞决定对档案缓存器进行读出或写入的动作；〝R/W〞=1 为读出档案缓存器，〝R/W〞=0 为写入档案缓存器。本指令以数据结构的 Record 观念执行的，也就是 Pr 指标所指的是每笔长度为 L 的区块，举例来说若 Sa=R0，Sb=F0，Pr=2，L=10，那么执行读写的区域就是 F20~F29，示意图如下：



- 本指令读写的缓存器，为系统内部的"档案缓存器"，这些缓存器无法由其它的功能指令存取，只有本指令才可对其读写。
- 如果指标递增〝INC〞=1，则每次执行完本指令之后，指针缓存器 Pr 的内容值加 1，也就是说指向下一个长度为 L 的内存区块。
- 若长度为 0 或大于 511 或指针长度超出档案缓存器范围 F0～F8191，则"指标错误"ERR 设为 1，本指令不执行。

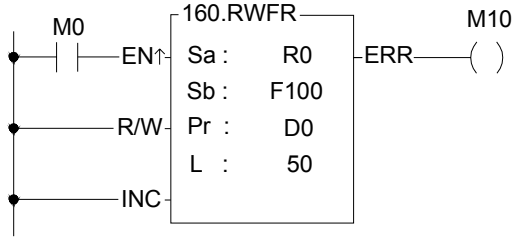
FUN160D P RWFR	读/写档案缓存器 （Read/Write File Register）	FUN160D P RWFR

程序范例一：



- 当 M0 由 0→1 时，若 D0=2，则将 R0～R49 内的数据，搬移到 F200～F249 内覆盖它。
- 执行完本指令之后，指针缓存器自动加 1。

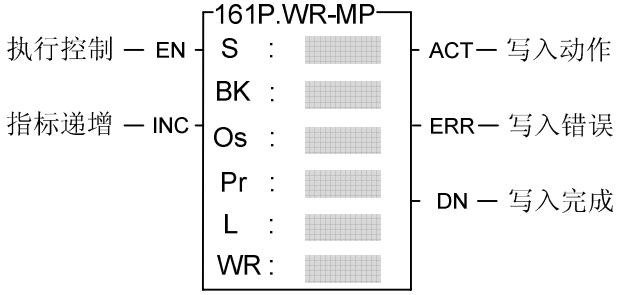
程序范例二：



- 当 M0 由 0→1 时，若 D0=1，则将 F150～F199 内的资料。搬移到 R0～R49 内覆盖它。
- 执行完本指令之后，指针缓存器自动加 1。

FUN161 P WR-DP	写入数据至数据记忆匣 （Write Data Pack）	FUN161 P WR-DP

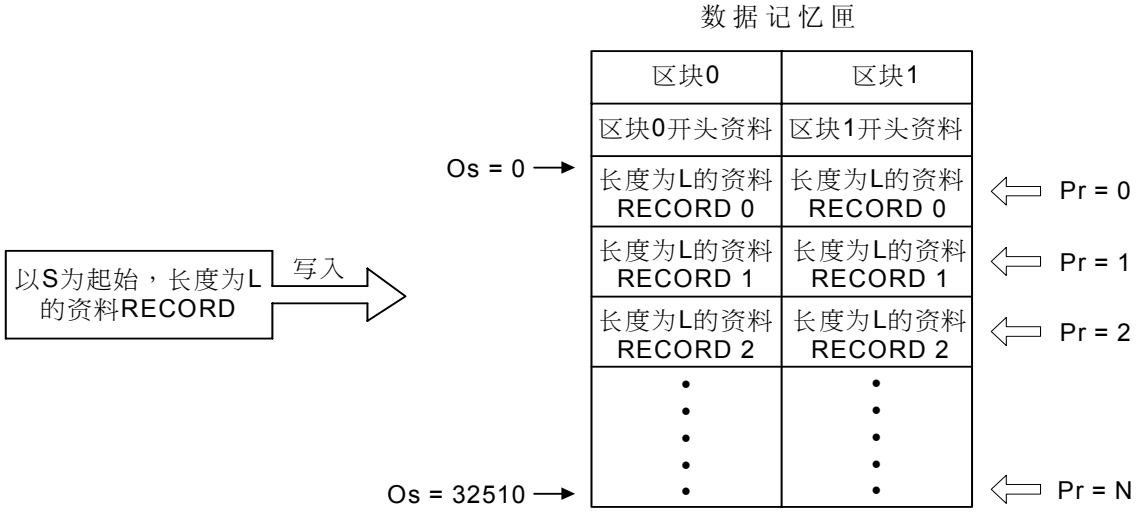
阶梯图符号



S：写入数据的来源起始缓存器号码
BK：Data Pack 的区块号码，0～1
Os：分区数据起始位置
Pr：指针缓存器号码
L：写入数据长度 1～128
WR：工作缓存器起始号码，共占用 2 个缓存器
S 可结合 V、Z、P0～P9 作间接寻址应用

操作数 \ 范围	HR	ROR	DR	K	XR
	R0 | R3839	R5000 | R8071	D0 | D4095		V、Z P0～P9
S	○	○	○		○
BK				0～1	
Os	○	○	○	0～32510	
Pr	○	○*	○		
L	○	○*	○	1～128	
WR	○	○*	○		

● FBs 的 ROM PACK 除了可用来储存梯形图控制程序外，还可以通过本指令用来当作数据记忆匣(Data Pack)用来作为可携式(Portable)机台生产成型数据的存取装置。当执行控制〝ENP〞由 0→1 时，自缓存器 S 开始，将长度 L 的数据写入所指定数据记忆匣的区块(BK)内，由分区数据起始位置(Os)加指针所指地址开始写入。本指令以数据结构的 RECORD 观念执行，也就是 Pr 指标所指的是每笔长度为 L 的 RECORD，通过本指令将其储存到数据记忆匣内。本指令执行示意图如下:



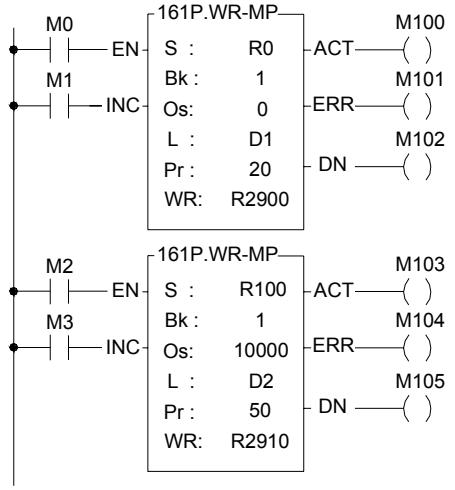
● 若指标递增〝INC〞=1，则每次执行完本指令之后，指针缓存器 Pr 的内容值加 1，也就是说指向下一个长度为 L 的 RECORD。

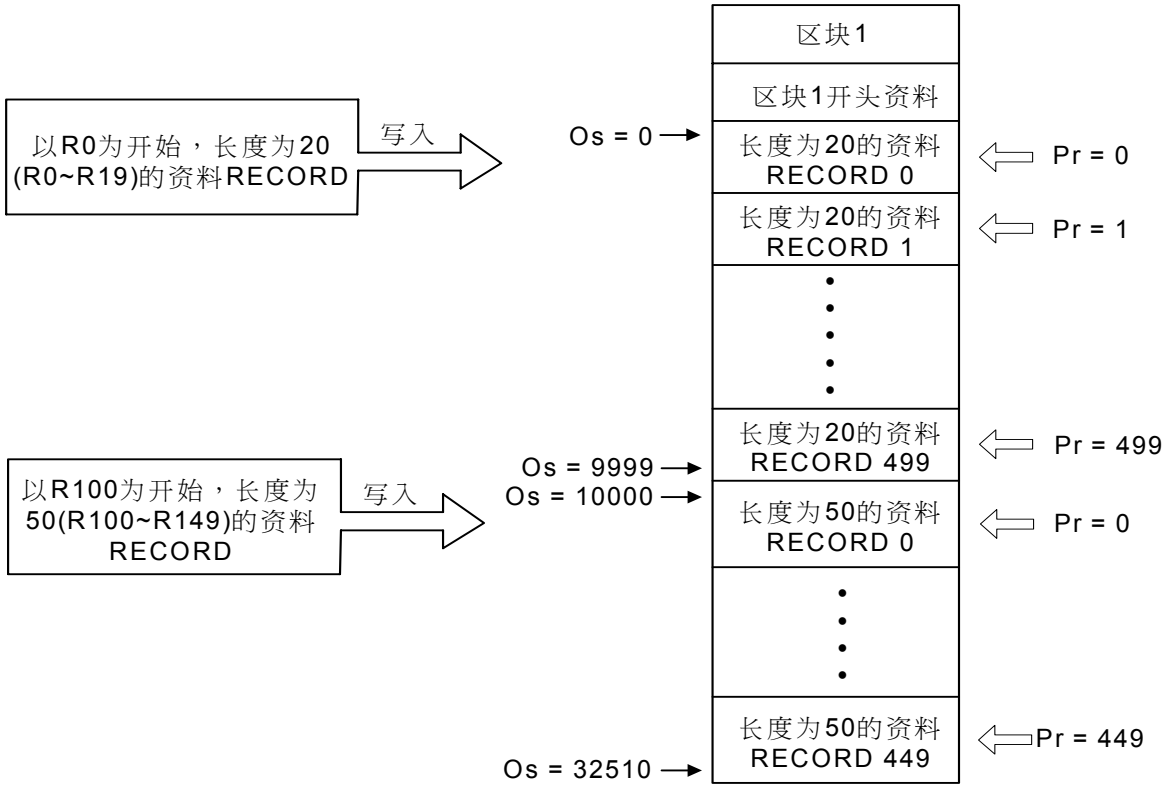
● 若长度为 0 或大于 128 或指标所指超出范围，则错误指示"ERR"设为 1，本指令不执行。

FUN161 P WR-DP	写入数据至数据记忆匣 （Write Data Pack）	FUN161 P WR-DP

- 本指令在执行数据写入与写入数据比对过程中有可能会需要多次扫描时间才能完成；在写入执行过程中时，输出指示"ACT"为 1；当写入完成且写入数据比对无误时，输出指示"DN"为 1；当写入完成但写入数据比对有误时，输出指示"ERR"为 1。
- FBs 的 ROM PACK 可规划为程序储存装置或当作机台生产成型数据记忆装置，或两者兼具；梯形图控制程序固定储存在区块 0，而生产成型数据则可选择储存在区块 0 或区块 1；每个区块的内存容量为 32K Word。

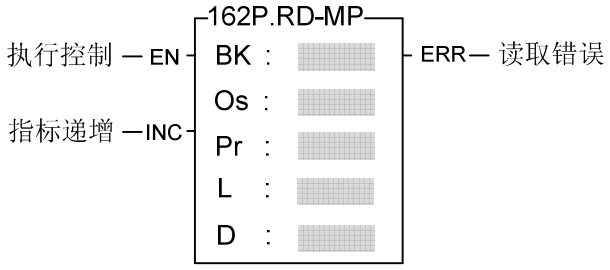
程序范例一：写入两种不同长度的 RECORD 到数据记忆匣区块 1





FUN162 P RD-DP	由数据记忆匣读取数据 （Read Data Pack）	FUN162 P RD-DP

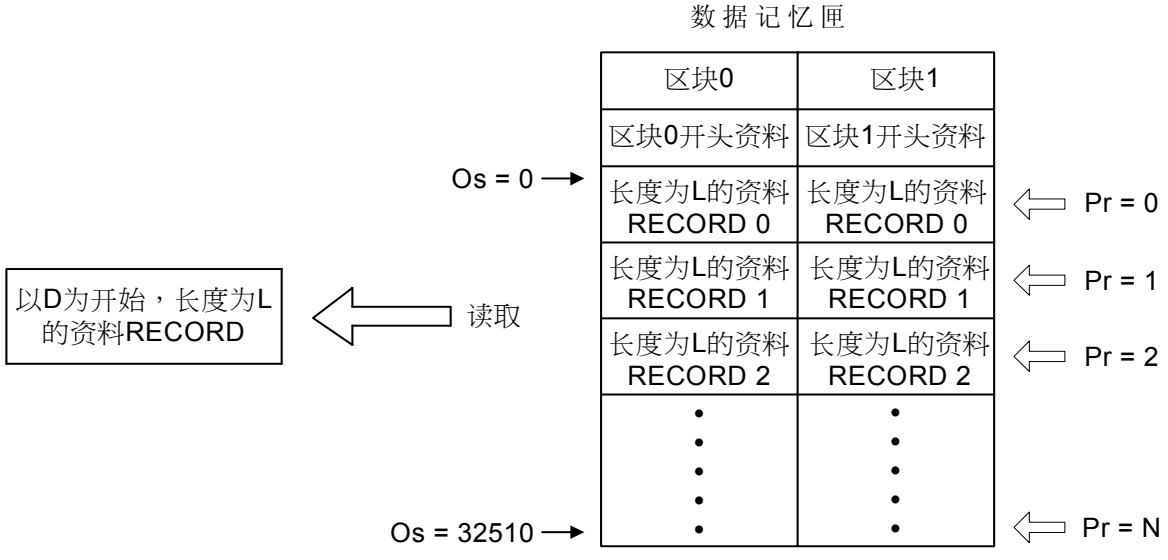
阶梯图符号



BK：Data Pack 的区块号码，0～1
Os：分区数据起始位置
Pr：指针缓存器号码
L：读取数据长度 1～128
D：存放读取数据的缓存器起始号码

操作数 \ 范围	HR	ROR	DR	K
	R0 | R3839	R5000 | R8071	D0 | D3999	
BK				0～1
Os	○	○	○	0～32510
Pr	○	○*	○	
L	○	○*	○	1～128
D	○	○*	○	

● FBs 的 ROM PACK 如果储存有 FUN161 指令所写入的机台生产成型数据，则可通过本指令将储存的数据读出再用，以减少生产调机时间。
当执行控制〝EN〞=1 或由 0→1(P 指令)时，将所指定数据记忆匣的区块(BK)内，由分区数据起始位置(Os)加指针所指地址开始，长度为 L 的数据 RECORD 读出。本指令以数据结构的 RECORD 观念执行，也就是 Pr 指标所指的是每笔长度为 L 的 RECORD，通过本指令将其由数据记忆匣内读出。本指令执行示意图如下:



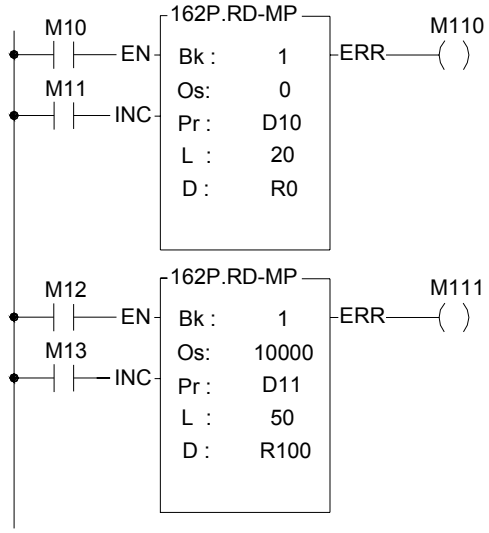
● 如果指标递增〝INC〞=1，则每次执行完本指令之后，指针缓存器 Pr 的内容值加 1，也就是说指向下一个长度为 L 的 RECORD。

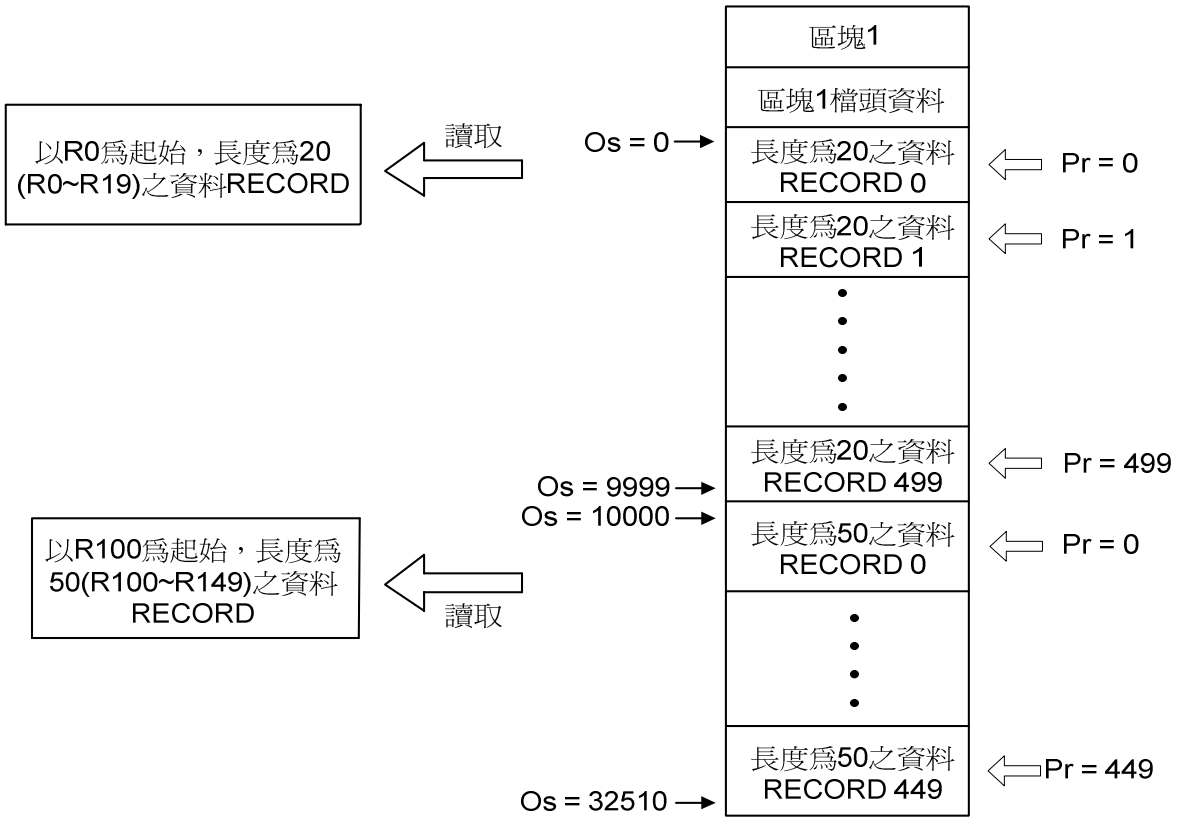
● 如果长度为 0 或大于 128 或指标所指超出范围，则错误指示"ERR"设为 1，本指令不执行。当 ROM PACK 内无数据或数据格式不正确导致 FUN162 无法读取数据时，错误指示"ERR"亦设为 1，且本指令不执行。

FUN162 P RD-DP	由数据记忆匣读取数据 （Read Data Pack）	FUN162 P RD-DP

程序范例一：由数据记忆匣区块 1 读取两种不同长度的 RECORD

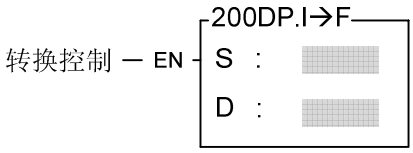
※ ROM PACK 内需要有相符数据格式的数据，否则本范例无法执行。





FUN200D P I→F	整数转换浮点数 (CONVERSION OF INTEGER TO FLOATING POINT NUMBER)	FUN200D P I→F

阶梯图符号



S：来源缓存器的起始号码

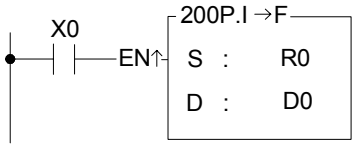
D：存放结果(浮点数)的缓存器起始号码

S、D 操作数可结合 V、Z、P0~P9 指标作间接定只应用

操作数 \ 范围	HR	ROR	DR	K	XR
	R0 | R3839	R5000 | R8071	D0 | D4095	16/32 位 正、负数	V、Z P0~P9
S	○	○	○	○	○
D	○	○*	○*		○

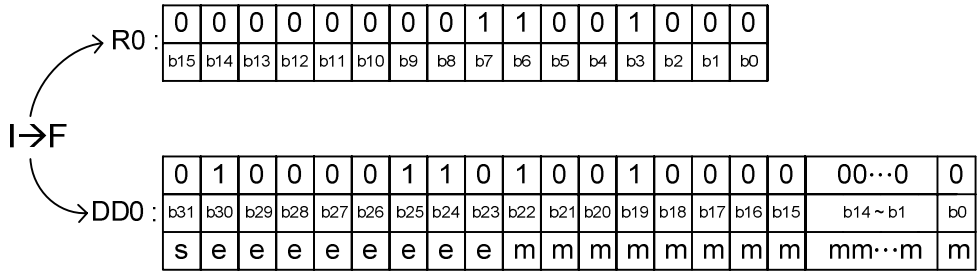
- 永宏 PLC 的浮点数格式符合 IEEE-754 所制定的标准，有关浮点数格式的详细说明请参阅 5-3 章(数目系统)..... 5-9 页。
- 当执行控制〝EN〞=1 或〝EN↑〞（P 指令）由 0→1 时，将 S 缓存器内的整数数值数据，转换成浮点数格式数据之后，存放于 D 缓存器中。

程序范例



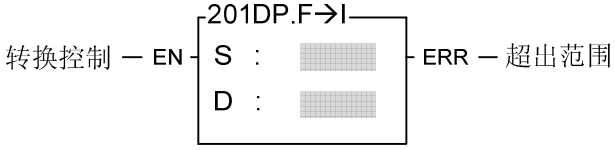
- R0 = 200 → X0=↑

(经浮点数转换之后) → DD0 = 4348000H



FUN201D P F→I	浮点数转换整数 (CONVERSION OF FLOATING POINT NUMBER TO INTEGER)	FUN201D P F→I

阶梯图符号



S：来源缓存器的起始号码

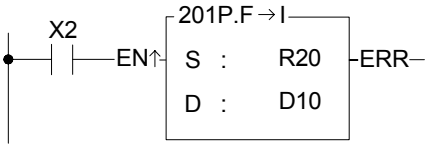
D：存放结果(整数)的缓存器起始号码

S、D 操作数可结合 V、Z、P0~P9 指标作间接定址应用

操作数 \ 范围	HR	ROR	DR	XR
	R0 | R3839	R5000 | R8071	D0 | D4095	V、Z P0~P9
S	○	○	○	○
D	○	○*	○*	○

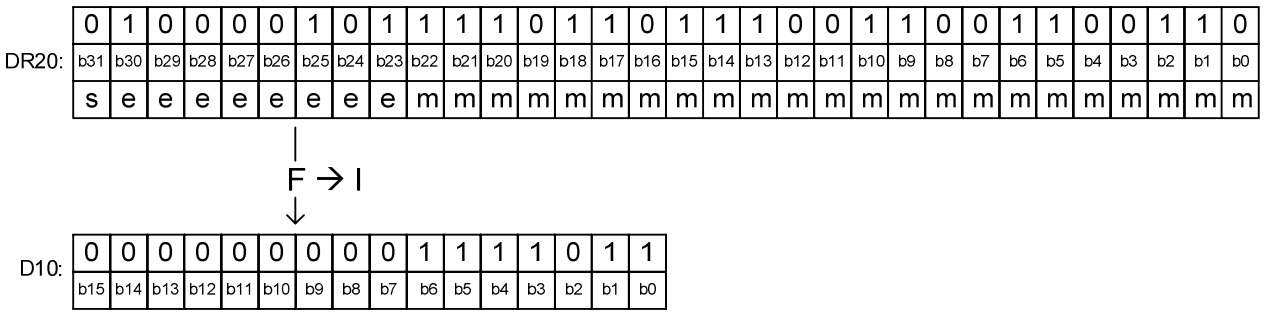
- 永宏 PLC 的浮点数格式符合 IEEE-754 所制定的标准，有关浮点数格式的详细说明请参考 5-3 章(数目系统)..... 5-9 页。
- 当执行控制〝EN〞=1 或〝EN↑〞（P 指令）由 0→1 时，将 S 缓存器内的浮点数数值数据，转换成整数格式数据之后，存放在 D 缓存器中。
- 若 D 缓存器(目的缓存器)存放的转换结果，超出有效范围，则错误旗号〝ERR〞设为 1，且本指令不执行，而 D 缓存器的内容维持不变。

程序范例



- DR20 = 123.45 → X2=↑

(经整数转换之后) → D10 = 007BH



FUN202 P FADD	浮点数加法运算 (FLOATING POINT NUMBER ADDITION)	FUN202 P FADD

阶梯图符号

Sa：被加数或其缓存器号码。

Sb：加数或其缓存器号码。

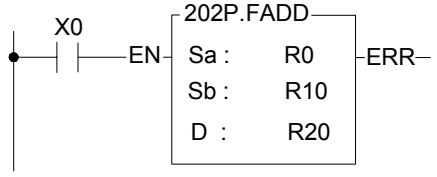
D：存放结果(和)的缓存器起始号码

Sa、Sb、D 操作数可结合 V、Z、P0~P9 指标作间接定址应用

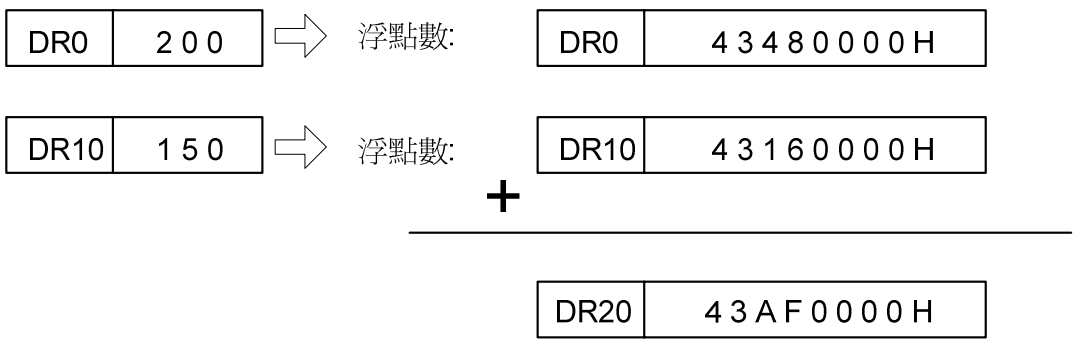
操作数 \ 范围	HR	ROR	DR	K	XR
	R0 | R3839	R5000 | R8071	D0 | D4095	浮点数	V、Z P0~P9
Sa	○	○	○	○	○
Sb	○	○	○	○	○
D	○	○*	○*		○

- 永宏 PLC 的浮点数格式符合 IEEE-754 所制的标准，有关浮点数格式的详细说明请参考 5-3 章(数目系统)..... 5-9 页。
- 当加算控制〝EN〞=1 或〝EN↑〞（P 指令）由 0→1 时，将 Sa 与 Sb 作浮点数加法运算并将结果写入 D 去。假若执行结果超出浮点数可表示的范围($\pm 3.4*10^{38}$)，则错误旗号〝ERR〞设为 1，且本指令不执行，而 D 缓存器的内容维持不变。

程序范例



• 当 X0=1，将 Sa 与 Sb 作浮点数加法运算:



FUN 203 P FSUB	浮点数减法运算 (FLOATING POINT NUMBER SUBTRACTION)	FUN 203 P FSUB

阶梯图符号

Sa：被减数或其缓存器号码。

Sb：减数或其缓存器号码。

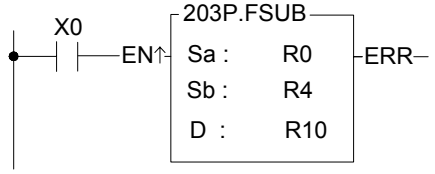
D：存放结果(差)的缓存器起始号码

Sa、Sb、D 操作数可结合 V、Z、P0~P9 指标作间接定址应用

操作数 \ 范围	HR	ROR	DR	K	XR
	R0 | R3839	R5000 | R8071	D0 | D4095	浮点数	V、Z P0~P9
Sa	○	○	○	○	○
Sb	○	○	○	○	○
D	○	○*	○*		○

- 永宏 PLC 的浮点数格式符合 IEEE-754 所制定的标准，有关浮点数格式的详细说明请参考 5-3 章(数目系统)..... 5-9 页。
- 当加算控制〝EN〞=1 或〝EN↑〞（P 指令）由 0→1 时，将 Sa 与 Sb 作浮点数减法运算并将结果写入 D 去。假若执行结果超出浮点数可表示的范围($\pm 3.4*10^{38}$)，则错误旗号〝ERR〞设为 1，且本指令不执行，而 D 缓存器的内容维持不变。

程序范例



• 当 X0=↑，将 Sa 与 Sb 作浮点数减法运算:

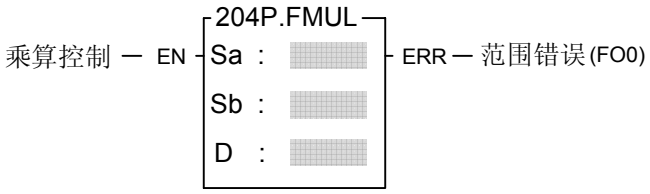
DR0	200	⇨ 浮点数：	DR0	43480000H
DR4	500	⇨ 浮点数：	DR4	43FA0000H

—

DR10	C3960000H

FUN 204 P FMUL	浮点数乘法运算 (FLOATING POINT NUMBER MULTIPLICATION)	FUN 204 P FMUL

阶梯图符号



Sa：被乘数或其缓存器号码。

Sb：乘数或其缓存器号码。

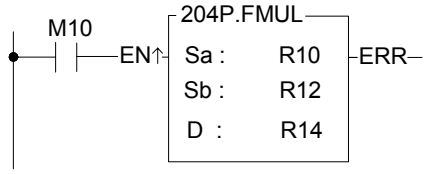
D：存放结果(积)的缓存器起始号码

Sa、Sb、D 操作数可结合 V、Z、P0~P9 指标作间接定址应用

操作数 \ 范围	HR	ROR	DR	K	XR
	R0 | R3839	R5000 | R8071	D0 | D4095	浮点数	V、Z P0~P9
Sa	○	○	○	○	○
Sb	○	○	○	○	○
D	○	○*	○*		○

- 永宏 PLC 的浮点数格式符合 IEEE-754 所制定的标准，有关浮点数格式的详细说明请参考 5-3 章(数目系统)..... 5-9 页。
- 当加算控制〝EN〞=1 或〝EN↑〞（P 指令）由 0→1 时，将 Sa 与 Sb 作浮点数乘法运算并将结果写入 D 去。假如执行结果超出浮点数可表示的范围($\pm 3.4*10^{38}$)，则错误旗号〝ERR〞设为 1，且本指令不执行，而 D 缓存器的内容维持不变。

程序范例



• 当 M10=↑，将 Sa 与 Sb 作浮点数乘法运算:

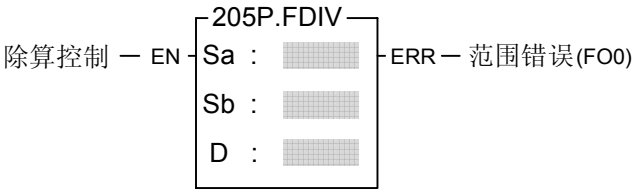
DR10	123.45	⇨ 浮点数：	DR10	42F6E666H
DR12	678.54	⇨ 浮点数：	DR12	4429A28FH

×

DR14	47A39AE2H

FUN 205 P FDIV	浮点数除法运算 (FLOATING POINT NUMBER DIVIDION)	FUN 205 P FDIV

阶梯图符号



Sa：被除数或其缓存器号码。

Sb：除数或其缓存器号码。

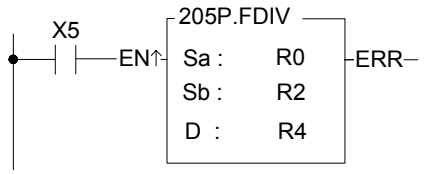
D：存放结果(商)的缓存器起始号码

Sa、Sb、D 操作数可结合 V、Z、P0~P9 指标作间接定址应用

操作数 \ 范围	HR	ROR	DR	K	XR
	R0 | R3839	R5000 | R8071	D0 | D4095	浮点数	V、Z P0~P9
Sa	○	○	○	○	○
Sb	○	○	○	○	○
D	○	○*	○*		○

- 永宏 PLC 的浮点数格式符合 IEEE-754 所制定的标准，有关浮点数格式的详细说明请参考 5-3 章(数目系统)..... 5-9 页。
- 当加算控制〝EN〞=1 或〝EN↑〞（P 指令）由 0→1 时，将 Sa 与 Sb 作浮点数除法运算并将结果写入 D 去。假若执行结果超出浮点数可表示的范围($\pm 3.4*10^{38}$)，则错误旗号〝ERR〞设为 1，且本指令不执行，而 D 缓存器的内容维持不变。

程序范例



• 当 X5=↑，将 Sa 与 Sb 作浮点数除法运算:

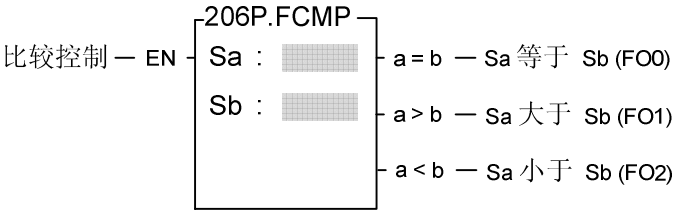
DR0	125.25	⇨ 浮点数：	DR0	42FA8000H
DR2	5	⇨ 浮点数：	DR2	40A00000H

÷

DR4	41C86666H

FUN 206 P FCMP	浮点数比较运算 (FLOATING POINT NUMBER COMPARE)	FUN 206 P FCMP

阶梯图符号



Sa：比较值 a 或其缓存器号码。

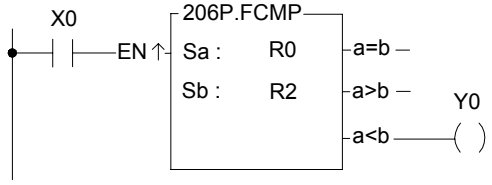
Sb：比较值 b 或其缓存器号码。

Sa、Sb 操作数可结合 V、Z、P0~P9 指标作间接定址应用

操作数 \ 范围	HR	ROR	DR	K	XR
	R0 ~ R3839	R5000 ~ R8071	D0 ~ D4095	浮点数	V、Z P0~P9
Sa	○	○	○	○	○
Sb	○	○	○	○	○

- 永宏 PLC 的浮点数格式符合 IEEE-754 所制定的标准，有关浮点数格式的详细说明请参考 5-3 章(数目系统)..... 5-9 页。
- 当比较控制〝EN〞=1 或〝EN↑〞（P 指令）由 0→1 时，将 Sa 与 Sb 作浮点数比较运算，若 Sa=Sb 则 FO0 设为 1，若 Sa＞Sb 则 FO1 设为 1，若 Sa＜Sb 则 FO2 设为 1。

程序范例



• 当 X0=↑，将 Sa 与 Sb 作浮点数比较运算:

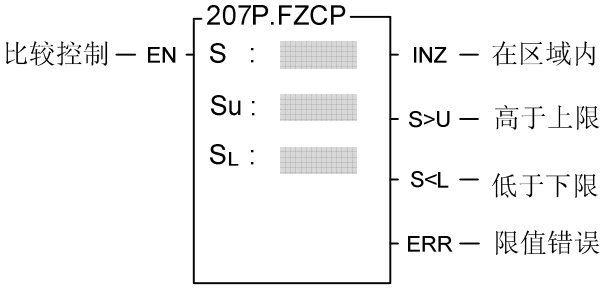
DR0	200.1	⇨ 浮点数：	DR0	43481 99AH

DR2	200.2	⇨ 浮点数：	DR2	43483333H

- 上例假若 DR0 的值为 200.1，DR2 的值为 200.2，则当 X0=↑时，CMP 指令执行比较工作，并得出 a＜b 的结果，故会将 FO0 及 FO1 设为 0，FO2（a＜b）设为 1。
- 若需要复合结果，如≧、≦、＜＞等，请先将＝、＞、＜等结果送到继电器再由继电器取出 OR 起来即可。

FUN 207 P FZCP	浮点数区域比较运算 (FLOATING POINT NUMBER ZONE COMPARE)	FUN 207 P FZCP

阶梯图符号



S ：存放比较数据(浮点数)的缓存器号码。

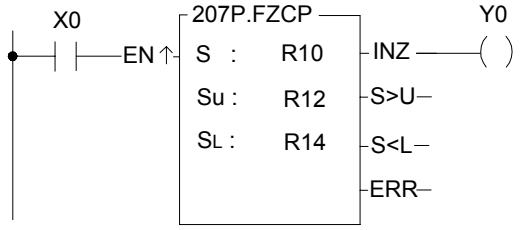
Su：区域上限值或上限值缓存器号码。

SL：区域下限值或下限值缓存器号码。

S，Su，SL 可结合 V、Z、P0~P9 作间接寻址应用。

操作数 \ 范围	HR	ROR	DR	K	XR
	R0 ~ R3839	R5000 ~ R8071	D0 ~ D4095	浮点数	V、Z P0~P9
S	○	○	○		○
Su	○	○	○	○	○
SL	○	○	○	○	○

- 永宏 PLC 的浮点数格式符合 IEEE-754 所制定的标准，有关浮点数格式的详细说明请参考 5-3 章(数目系统)..... 5-9 页。
- 当比较控制〝EN〞=1 或〝EN↑〞（P 指令）由 0→1 时，执行 S 与上限 Su 及下限 SL 的比较，若 S 介于上限值与下限值之间（SL≦S≦Su），则在区域内旗号〝INZ〞设为 1，若 S 的值大于上限 Su，则高于上限旗号〝S＞U〞设为 1，若 S 的值小于下限 SL，则低于下限旗号〝S＜L〞设为 1。
- 上限 Su 应大于下限 SL，若 Su＜SL，则限值错误旗号〝ERR〞设为 1，且本指令不执行。



- 左图程序范例是将 DR10 的值和由 DR12 和 DR14 所构成的上、下限区域作比较，假设 DR10～DR14 的数值如下图左，则可获得如下图右的执行结果。
- 若输出结果需要不在区域内，则可用 OUT NOT Y0 即可。

S	DR10	2000.2	⇨ 浮点数：	DR10	44FA0666H
Su	DR12	3000.3	⇨ 浮点数：	DR12	453B84CDH （上限值）
SL	DR14	1000.1	⇨ 浮点数：	DR14	447A0666H （下限值）

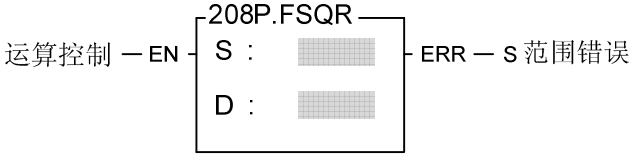
执行前

X0=↑ → 浮点数区域比较 → Y0 = 1

执行后结果

FUN 208 P FSQR	浮点数开根号运算 (FLOATING POINT NUMBER SQUARE ROOT)	FUN 208 P FSQR

阶梯图符号



S：求平方根的来源数值或缓存器号码。

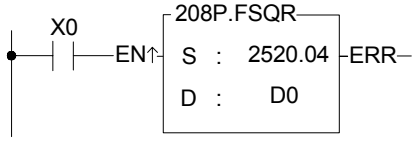
D：存放结果(平方根值)的缓存器号码。

S、D 可结合 V、Z、P0~P9 作间接寻址应用

操作数 \ 范围	HR	ROR	DR	K	XR
	R0 | R3839	R5000 | R8071	D0 | D4095	浮点数	V、Z P0~P9
S	○	○	○	○	○
D	○	○*	○*		○

- 永宏 PLC 的浮点数格式符合 IEEE-754 所制定的标准，有关浮点数格式的详细说明请参考 5-3 章(数目系统)..... 5-9 页。
- 当运算控制〝EN〞=1 或〝EN↑〞（P 指令）由 0→1 时，将 S 值或 S 所指定的缓存器内容值取平方根值后存入 D 所指定的缓存器内。
- 当 S 值为缓存器内容值，而值为负数时，则 S 值错误旗号〝ERR〞设为 1，且本指令不执行。

程序范例



• 当 X0=↑，将 S 作浮点数开根号运算:

S : K | 2520.04

⇩ X0 = ↑

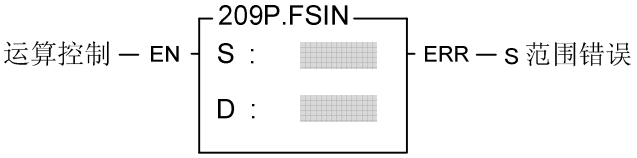
D : D1 | D0 | 50.2 ⇨ 浮点数 4248 | CCCD H

D1 D0

$\sqrt{2520.04} = 50.2$

FUN 209 P FSIN	浮点数表示法取三角函数(SIN)运算 (SIN TRIGONOMETRIC INSTRUCTION)	FUN 209 P FSIN

阶梯图符号



S：求 SIN 值的来源数值或缓存器号码。

D：存放结果的缓存器号码。

S、D 可结合 V、Z、P0~P9 作间接寻址应用

操作数 \ 范围	HR	ROR	DR	K	XR
	R0 | R3839	R5000 | R8071	D0 | D4095	整数 16 位	V、Z P0~P9
S	○	○	○	○	○
D	○	○*	○*		○

- 永宏 PLC 的浮点数格式符合 IEEE-754 所制定的标准，有关浮点数格式的详细说明请参考 5-3 章(数目系统)..... 5-9 页。
- 当运算控制〝EN〞=1 或〝EN↑〞（P 指令）由 0→1 时，将 S 值或 S 所指定的缓存器内容值取 SIN 函数后存入 D 所指定的缓存器内。S 的有效范围为-18000～+18000，单位为 0.01 度。
- 若 S 值或 S 所指定的缓存器内容值超过其有效范围(-18000～+18000)，则错误旗号〝ERR〞设为 1，且本指令不执行。

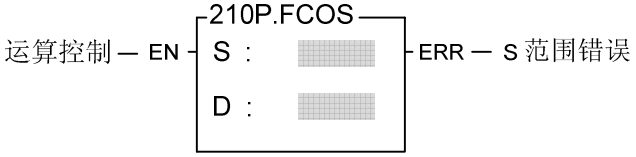
程序范例

- 左图范例，当 X0=↑将 SIN∠30的值存入 DR100 之中。

SIN(30) = 0.5

FUN 210 P FCOS	浮点数表示法取三角函数(COS)运算 (COS TRIGONOMETRIC INSTRUCTION)	FUN 210 P FCOS

阶梯图符号



S：求 COS 值的来源数值或缓存器号码。

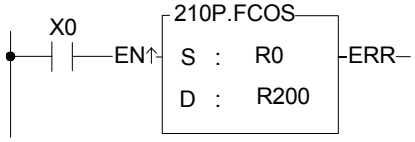
D：存放结果的缓存器号码。

S、D 可结合 V、Z、P0~P9 作间接寻址应用

操作数 \ 范围	HR	ROR	DR	K	XR
	R0 | R3839	R5000 | R8071	D0 | D4095	整数 16 位	V、Z P0~P9
S	○	○	○	○	○
D	○	○*	○*		○

- 永宏 PLC 的浮点数格式符合 IEEE-754 所制定的标准，有关浮点数格式的详细说明请参考 5-3 章(数目系统)..... 5-9 页。
- 当运算控制〝EN〞=1 或〝EN↑〞（P 指令）由 0→1 时，将 S 值或 S 所指定的缓存器内容值取 COS 函数后存入 D 所指定的缓存器内。S 的有效范围为-18000～+18000，单位为 0.01 度。
- 若 S 值或 S 所指定的缓存器内容值超过其有效范围(-18000～+18000)，则错误旗号〝ERR〞设为 1，且本指令不执行。

程序范例

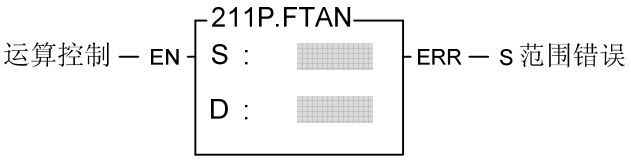


- 左图范例，当 X0=↑将 $\cos \angle 60$ 的值存入 DR200 之中。

COS(60) = 0.5

FUN 211 P FTAN	浮点数表示法取三角函数(TAN)运算 (TAN TRIGONOMETRIC INSTRUCTION)	FUN 211 P FTAN

阶梯图符号



S：求 TAN 值的来源数值或缓存器号码。

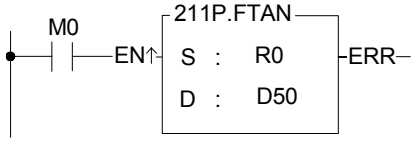
D：存放结果的缓存器号码。

S、D 可结合 V、Z、P0~P9 作间接寻址应用

操作数 \ 范围	HR	ROR	DR	K	XR
	R0 | R3839	R5000 | R8071	D0 | D4095	整数 16 位	V、Z P0~P9
S	○	○	○	○	○
D	○	○*	○*		○

- 永宏 PLC 的浮点数格式符合 IEEE-754 所制定的标准，有关浮点数格式的详细说明请参考 5-3 章(数目系统)..... 5-9 页。
- 当运算控制〝EN〞=1 或〝EN↑〞（P 指令）由 0→1 时，将 S 值或 S 所指定的缓存器内容值取 TAN 函数后存入 D 所指定的缓存器内。S 的有效范围为-18000～+18000，单位为 0.01 度。
- 若 S 值或 S 所指定的缓存器内容值超过其有效范围(-18000～+18000)，则错误旗号〝ERR〞设为 1，且本指令不执行。

程序范例

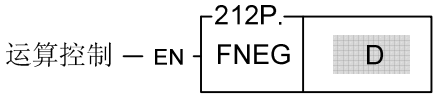


- 左图范例，当 M0=↑将 TAN∠45的值存入 DD50 之中。

TAN(45) = 1

FUN 212 P FNEG	浮点数取负值运算 (CHANGE SIGN OF THE FLOATING POINT NUMBER)	FUN 212 P FNEG

阶梯图符号



D：存放取负值运算结果的缓存器号码。

D 可结合 V、Z、P0~P9 作间接寻址应用

操作数 \ 范围	HR	ROR	DR	XR
	R0 ~ R3839	R5000 ~ R8071	D0 ~ D4095	V、Z P0~P9
D	○	○	○	○

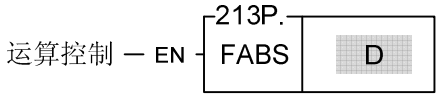
- 永宏 PLC 的浮点数格式符合 IEEE-754 所制定的标准，有关浮点数格式的详细说明请参考 5-3 章(数目系统)..... 5-9 页。
- 当运算控制〝EN〞=1 或〝EN↑〞（P 指令）由 0→1 时，将 D 所指定的缓存器内容取其负数后存回原缓存器 D(浮点数)。
- 若 D 的内容值原为负数，取负数的结果将变为正数。

程序范例

- 左图范例，将 DR0 的内容值取负值运算之后，存回到 DR0 内(浮点数)。

FUN 213 P FABS	浮点数取绝对值运算 (FLOATING POINT NUMBER ABSOLUTE VALUE)	FUN 213 P FABS

阶梯图符号



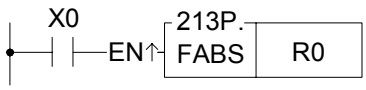
D：存放取绝对值运算结果的缓存器号码。

D 可结合 V、Z、P0~P9 作间接寻址应用

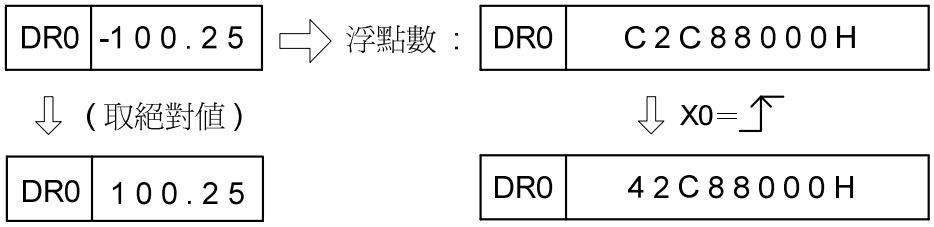
操作数 \ 范围	HR	ROR	DR	XR
	R0 | R3839	R5000 | R8071	D0 | D4095	V、Z P0~P9
D	○	○	○	○

- 永宏 PLC 的浮点数格式符合 IEEE-754 所制定的标准，有关浮点数格式的详细说明请参靠 5-3 章(数目系统)..... 5-9 页。
- 当运算控制〝EN〞=1 或〝EN↑〞（P 指令）由 0→1 时，将 D 所指定的缓存器内容取其绝对值后存回到原缓存器 D(浮点数)。

程序范例



- 左图范例，将 DR0 的内容值取绝对值运算之后，存回到 DR0 内(浮点数)。



MEMO