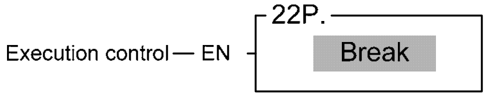
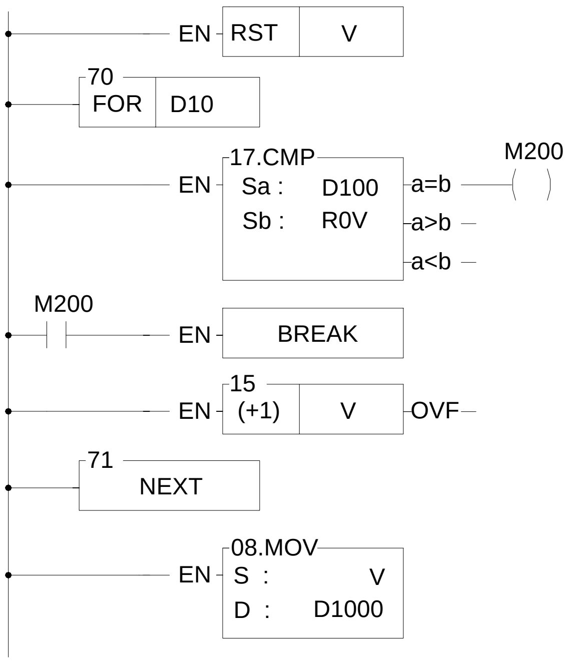
Chapter 7 Advanced Function Instructions

FUN22 P BREAK	BREAK FROM FOR AND NEXT LOOP （BREAK）	FUN22 P BREAK

Ladder symbol

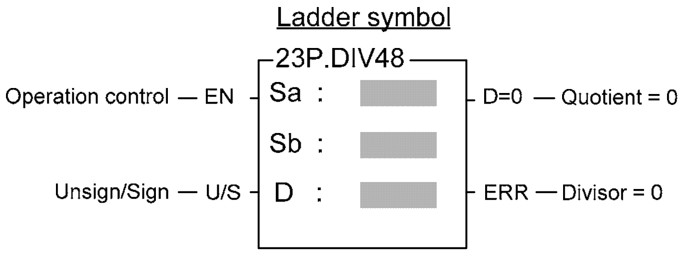


- When execution control〝EN〞=1 or changes from 0→1（ P instruction），it will terminate the FOR and NEXT program loop。
- The program within the FOR and NEXT loop will be executed N times (N is assigned by FOR instruction) successively，but if it is necessary to terminate the execution loop less than N times，the BREAK instruction is necessary to apply。
- The BREAK instruction must be located within the FOR and NEXT program loop。



Description：The loop count used to execute the FOR and NEXT program loop is assigned by register D10；the program within the FOR and NEXT loop is designed to search the same data storing in D100 from the register table starting at R0。If it finds，the searching loop will be terminated and then it goes to execute the program after the NEXT instruction；If it doesn't find，the searching loop will be executed N times (N is the content of D10) and then it goes to execute the program after the NEXT instruction。
M200 tells the status and D100 is the pointer of searching。

FUN 23 P DIV48	48-BIT DIVISION	FUN 23 P DIV48



Sa：Starting register of dividend

Sb：Starting register of divisor

D ：Starting register for storing the division result (quotient)

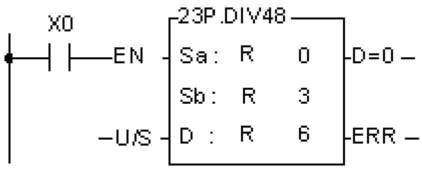
Sa，Sb，can combine V, Z, P0~P9 for index addressing.

Range / Operand	HR	OR	SR	ROR	DR	XR
	R0 ~ R3839	R3904 ~ R3967	R3968 ~ R4167	R5000 ~ R8071	D0 ~ D4095	V、Z P0~P9
Sa	○	○	○	○	○	○
Sb	○	○	○	○	○	○
D	○	○	○*	○*	○	○

- When operation control “EN”=1 or changes from 0→1 (P instruction), will perform the 48 bits division operation. Dividend and divisor are each formed by three consecutive registers starting by Sa and Sb respectively. If the result is zero, ‘D=0’ output will be set to 1. If divisor is zero then the ‘ERR’ will be set to 1 and the resultant register will keep unchanged.
- All operands involved in this function are all 48 bits, so Sa, Sb and D are all comprised by 3 consecutive registers.

Example: 48-bit division

In this example dividend formed by register R2, R1, R0 will be divided by divisor formed by register R5, R4, R3. The quotient will store in R8, R7, and R6.



Sa

R2	R1	R0
2147483647		

÷ Sb

R5	R4	R3
1234567		

R8	R7	R6
1739		

Quotient

FUN 24 D P SUM	SUM (Summation of block data)	FUN 24 D P SUM

Ladder symbol

24DP.SUM
Operation control — EN
S :
N :
D :

S : Starting number of source register

N : Number of registers to be summed
(successive N data units starting from S)

D : The register which stored the result (summation)

S, N, D, can associate with V, Z, P0~P9 index register to serve the indirect addressing application.

Range / Operand	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		○	○	○	○	○	○		○	○*	○*	○		○

- When operation control "EN"=1 or changes from 0→1 (P instruction), it puts the successive N units of 16bit or 32 bit (D instruction) registers for addition calculation to get the summation, and stores the result into the register which is designated by D.
- When the value of N is 0 or greater than 511, the operation will not be performed.
- Communication port1~4 can be used to serve as a general purpose ASCII communication interface. If the data error detecting method is Checksum, this instruction can be used to generate the sum value for sending data or ot use this instruction to check if the received data is error or not.

〈Example 1〉 When M1 changes from OFF→ON, following instruction will calculates the summation for 16-bit data.

M1 — EN
24P.SUM
S : R0
N : 6
D : R100

- The left illustrates that 6 16-bit registers starting from R0 is calculated for summation, and the result is stored into the R100 register.

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

→ R100=012FH

〈Example 2〉 When M1 is ON, it calculates the summation for 32-bit data.

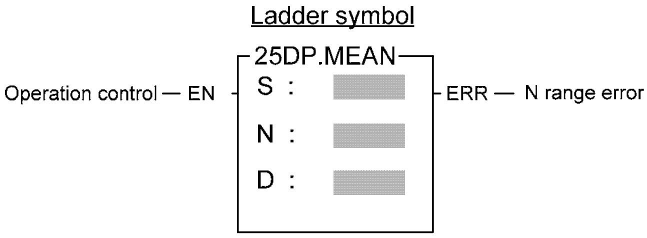
M1 — EN
24D.SUM
S : R0
N : 3
D : R100

- The left illustrates that three 32-bit registers starting from DR0, is calculated for their summation, and the result is stored into the DR100 register.

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

→ R101，R100=00A5009BH

FUN 25 D P MEAN	MEAN (Average of the block data)	FUN 25 D P MEAN



S : Source register number

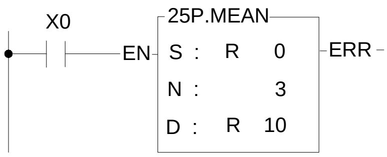
N : Number of registers to be averaged
(N units of successive registers starting from S)

D : Register number for storing result (mean value)

The S, N, D may combine with V, Z, P0~P9 to serve indirect address application

Range / Operand	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		○	○	○	○	○	○		○	○*	○*	○		○

- When operation control "EN" = 1 or from 0 to 1 (P instruction), add the N successive 16-bit or 32-bit (D instruction) numerical values starting from S, and then divided by N. Store this mean value (rounding off numbers after the decimal point) in the register specified by D.
- While the N value is derived from the content of the register, if the N value is not between 2 and 256, then the N range error "ERR" will be set to 1, and do not execute the operation.



- At left, the example program gets the mean value of the 3 successive 16-bit registers starting from R0, and stores the results into the 16-bit register R10

S (N=3)

R0	123
R1	9
R2	788

⇩ X0=↑

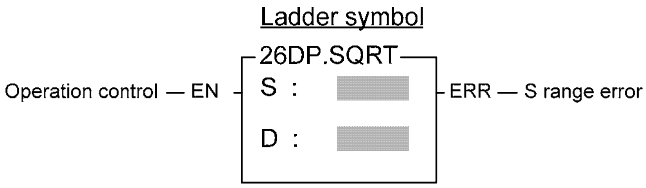
$$\frac{123+9+788}{3}$$

=306 （Rouding off the remainder）

D

R10	306

FUN 26 D P SQRT	SQUARE ROOT	FUN 26 D P SQRT



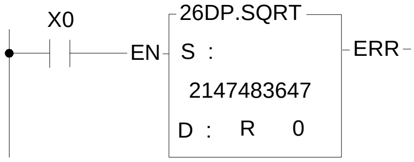
S : Source register to be taken square root

D : Register for storing result (square root value)

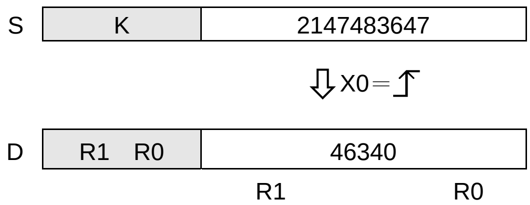
S, D may combine with V, Z, P0~P9 to serve indirect address application

Range / Operand	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	16/32-bit	V、Z P0~P9
S	○	○	○	○	○	○	○	○	○	○	○	○	○	○
D		○	○	○	○	○	○		○	○*	○*	○		○

- When operation control "EN" = 1 or from 0 to 1 (P instruction), take the square root (rounding off numbers after the decimal point) of the data specified by the S field, and store the result into the register specified by D.
- While the S value is derived from the content of the register, if the value is negative, then the S value error flag "ERR" will be set to 1, and do not execute the operation.



- The instruction at left calculates the square root of the constant 2147483647, and stores the result in R0.



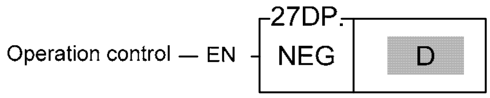
$$\sqrt{2147483647} = 46340.\underline{\underline{95}}$$

↑

rounding off

FUN 27 D P NEG	NEGATION (Take the negative value)	FUN 27 D P NEG

Ladder symbol

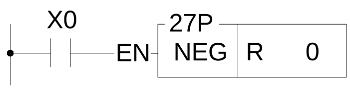


D : Register to be negated

D may combine with V, Z, P0~P9 to serve indirect address application

Range / Operand	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	○	○	○	○	○	○	○	○*	○*	○	○

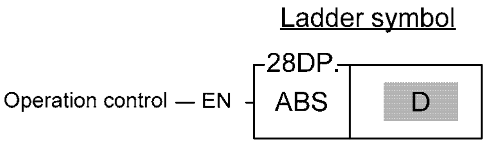
- When operation control "EN" = 1 or from 0 to 1 (P instruction), negate (ie. calculate 2's complement) the value of the content of the register specified by D, and store it back in the original D register.
- If the value of the content of D is negative, then the negation operation will make it positive.



- The instruction at left negates the value of the R0 register, and stores it back to R0.

D	R0	12345	☞3039H
		⇩ X0＝↑	
D	R0	−12345	☞CFC7H

FUN 28 D P ABS	ABSOLUTE (Take the absolute value)	FUN 28 D P ABS



D : Register to be taken absolute value

D may combine with V, Z, P0~P9 to serve indirect address application

Range / Operand	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	○	○	○	○	○	○	○	○*	○*	○	○

- When operation control "EN" = 1 or from 0 to 1 (P instruction), calculate the absolute value of the content of the register specified by D, and write it back into the original D register.

- The instruction at left calculates the absolute value of the R0 register, and stores it back in R0.

D | R1 R0 | –12345 | ☞CFC7H

⇩ X0 = ↑

D | R1 R0 | 12345 | ☞3039H

FUN 29 D P EXT	SIGN EXTENSION	FUN 29 D P EXT

Ladder symbol

Operation control — EN — 29P. EXT | D

D : Register to be taken sign extension

D may combine with V, Z, P0~P9 to serve indirect address application

Range / Ope-rand	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	○	○	○	○	○	○	○	○*	○*	○	○

- When operation control "EN" = 1 or from 0 to 1 (P instruction), this instruction will sign extent the 16 bit numerical value specified by D to 32-bit value and store it into the 32-bit register comprised by the two successive words, D + 1 and D. (Both values are the same, only it was originally formated as a 16 bit numerical value, and was then extended to be formated as a 32 bit numerical value.)

- This instruction extent the numerical value of a 16-bit register into an equivalent numerical value in a 32-bit register (for example 33FFH converts to 000033FFH), Its main function is for numerical operations (+,-,*,/,CMP......) which can take the 16 bit or 32 bit numerical values as operand. Before operation all the operand should be adjusted to the same length for proper operation.

X0 —EN— 29P EXT | R 0

- The instruction at left takes a 16 bit numerical value R0, and extends it to an equivalent value in 32 bits, then stores it into a 32 bit register (DR0=R1R0) comprised R0 and R1

		R1	B15								R0							B0	
D	R1 R0	Ignore the value of R1 before extension	1	1	0	0	1	1	1	1	1	1	0	0	0	1	1	1	－12345

⇩ X0＝↑

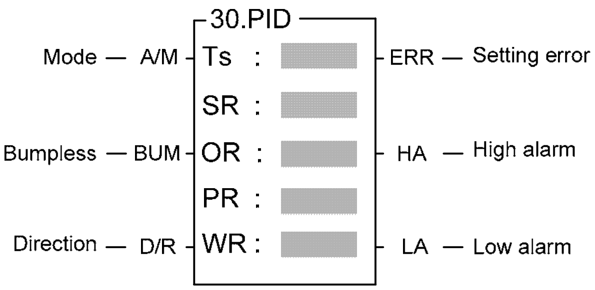
		B31							R1								B16	B15								R 0							B0	
D	R1 R0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	1	1	1	1	1	1	0	0	0	1	1	1	－12345

Fill B15 value into B31-B16,(if B15 is 0, then B31-B16 are all 0)

Before extension（16 bits） R0= CFC7H=－12345
After extension（32 bits）R1R0=FFFFCFC7H=－12345
} The two numerical values are actually the same

FUN 30 PID	GENERAL PURPOSE PID OPERATION (Brief description)	FUN 30 PID

Ladder symbol



Range / Operand	HR R0 – R3839	ROR R5000 – R8071	DR D0 – D4095	K
Ts	○	○	○	1~3000
SR	○	○*	○	
OR	○	○*	○	
PR	○	○*	○	
WR	○	○*	○	

Ts : PID Operation time interval

SR : Starting register of process control parameter table comprised by 8 consecutive registers.

OR : PID output register

PR : Starting register of the process parameter table comprised by 7 consecutive registers.

WR : Starting register of working variable for PID internal operation. It requires 7 registers and can't be re-used in other part of the ladder program.

- PID function (FUN 30) according to the current value of process variable (PV) derived from the external analog signal and the Set Point (SP) of process performs the calculation, which base on the PID formula. The result of calculation is the control output for the controlled process, which can feed directly to the AO module or other output interface or leaved for further process. The usage of PID control for process if properly can achieve a fast and smooth result of PV tracking toward SP change or immune to the disturbance of process.

- The PID formula in digital form:

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

M_n : Control output at time "n"

D4005 : The gain constant, the default is 1000, which can be set between 1~5000.

Pb : Proportional band (range : 2~5000, unit 0.1%. Kc (gain) =1000/ Pb)

T_i : Intergal time constant (range : 0~9999 corresponds to 0.00~99.99 Repeats/Minute)

T_d : Differential time constant (range : 0~9999 corresponds to 0.00~99.99 Minutes)

PV_n : Process value at time "n"

PV_{n-1} : Process value at time "n"

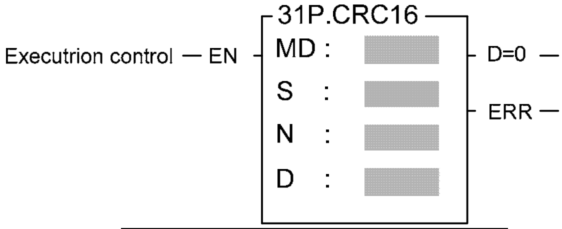
En :Error at time "n" =set value (SP) − process value at time "n" (PV_n)

T_s : Interval time of PID calculation (range: 1~3000, unit : 0.01 S)

Bias : Control output offset (range: 0~16380)

FUN31 P CRC16	CRC16 CALCULATION (CRC16)	FUN31 P CRC16

Ladder symbol



Range / Ope-rand	HR	ROR	DR	K
	R0 ∣ R3839	R5000 ∣ R8071	D0 ∣ D4095	
MD				0~1
S	○	○	○	
N	○	○	○	1~256
D	○	○*	○	

MD ：0, Lower byte of registers to be calculated the CRC16

：1, Reserved

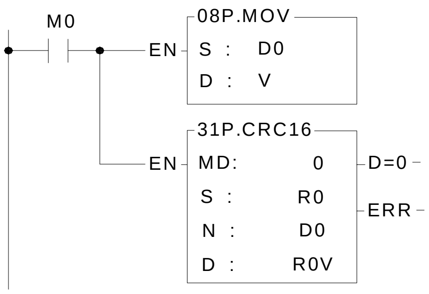
S：Starting address of CRC16 calculation

N：Length of CRC16 calculation (In Byte)

D：The destination register to store the calculation of CRC16,
Register D stores the Upper Byte of CRC16
Register D＋1 stores the Lower Byte of CRC16

S, N, D may associate with V、Z、P0~P9 index register to serve the indirect addressing application

- When execution control "EN"=1 or changes from 0→1（P instruction, it will start the CRC16 calculation from the lower byte of S and by the length of N, the result of calculation will be stored into register D and D+1.
- The output indication "D=0" will be ON if the value of calculation is 0.
- It will not execute the calculation and the output indication "ERR" will be ON if the length is invalid.
- When communicating with the intelligent peripheral in binary data format, the CRC16 error detection is used very often; the well known Modbus RTU communication protocol uses this method for error detection of message frame.
- CRC16 is the check value of a Cyclical Redundancy Check calculation performed on the message contents.
- Perform the CRC16 calculation on the received message data and error check value, the result of the calculation value must be 0, it means no error within this message frame.



Description：When M0 changes from 0→1, it will execute the CRC16 calculation starting from lower byte of R0, the length is assigned by D0, and then stores the CRC value into register R0+V and R0+V+1.

It is supposed D0=10, the registers R10 and R11 will store the CRC16 value.

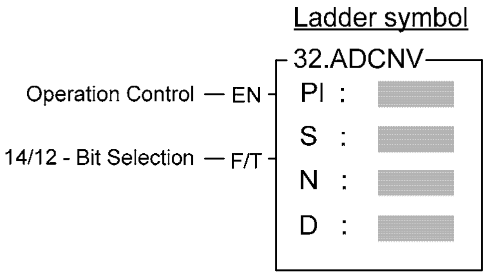
S

	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	CONVERTING THE RAW VALUE OF 4～20MA ANALOG INPUT （ADCNV）	FUN32 ADCNV



Range / Operand	HR	IR	ROR	DR	K
	R0 ∣ R3839	R3840 ∣ R3903	R5000 ∣ R8071	D0 ∣ D4095	
PI					0~1
S	○		○	○	
N	○	○	○	○	1~64
D	○		○*	○	

PI：0, the polarity setting of analog input module is at unipolar position

：1, the polarity setting of analog input module is at bipolar position

S：Starting address of source registers

N：Quantity of conversion (In Word)

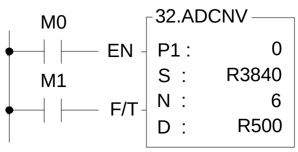
D：Starting address of destination registers

S, N, D may associate with V、Z、P0~P9 index register to serve the indirect addressing application.

- When the analog input is one of 2～10mA/ 4～20mA/1～5V/2～10V, the analog input module is the solution to get the value of this kind of signal, but the input span of the analog input module is 0～10mA/0～5V (Setting at 5V、Unipolar) or 0～20mA/0～10V(Setting at 10V、Unipolar), however there will exist the offset of the raw reading value; this instruction is applied to eliminate the offset and convert the raw reading value into the range of 0～4095(12-bit) or 0～16383(14-bit), it is more convenient for following operation.
- When execution control "EN"=1, it will execute the conversion starting from S, length by N, and then store the results into the D registers.
- When the input 〝F/T〞=0, it assigns the 12-bit analog input module; while 〝F/T〞=1, it assigns the 14-bit operation.
- This instruction will not act if invalid length of N.
- The reading value of the analog input must be in -2048~2047 or -8192~8191 format that the conversion will have the correct correspondence. Otherwise, if the reading value is in 0~4095 or 0~16383 format that the conversion will have the wrong correspondence.

FUN32 ADCNV	CONVERTING THE RAW VALUE OF 4～20MA ANALOG INPUT （ADCNV）	FUN32 ADCNV

Example：



Description：When M0 is ON and M1 is OFF, it will perform 6 points of conversion starting from R3840, where the offset of 4～20mA raw reading value will be eliminated, and the corresponding value 0～4095 will be stored into R500～R505.

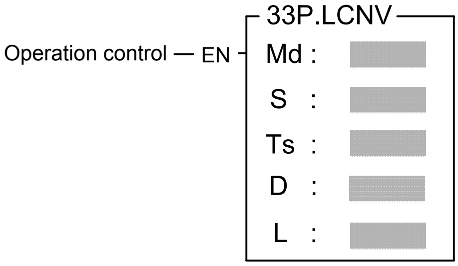
S		⇨	D		
R3840	－1229		R500	0	(4 mA)
R3841	409		R501	2047	(12 mA)
R3842	2047		R502	4095	(20 mA)
R3843	－2048		R503	0	(0 mA)
R3844	－2048		R504	0	(0 mA)
R3845	－2048		R505	0	(0 mA)

When M0 is ON and M1 is ON, it will perform 6 points of conversion starting from R3840, where the offset of 4~20mA raw reading value will be eliminated, and the corresponding value 0~16383 will be stored into 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	Linear Conversion （LCNV）	FUN33 P LCNV

Ladder symbol



Md：Operation mode，0～3

S ：Starting address of the source data

Ts：Starting address of the parameter table for conversion

D ：Starting address to store the result

L ：Quantity of conversion entry，1～64

Operand \ Range	HR R0 ∣ R3839	IR R3840 ∣ R3903	ROR R5000 ∣ R8071	DR D0 ∣ D3999	K
Md					0～3
S	○	○	○	○	
Ts	○		○	○	
D	○		○*	○	
L	○		○	○	1～64

- When the analog input module being used for the analog measurement, the raw reading value of the analog input can be converted into the engineering range through this instruction for display or for proceeding control operation.
- For process measurement calibration, making the linear conversion for the engineering process variable, which the measurement value from the PLC's can be corrected by the value from the standard meter's through this instruction.
- When execution control "EN"=1or from 0→1(P instruction), this instruction will perform the linear conversion operation according to the mode selection, where S is the starting address of the source data, Ts is the starting address of the conversion parameter table, D is the starting address to store the converted result, and L is the quantity of conversion entry.
- There are two expressions to meet the suitable application:

Expression 1 : Two points calibration method

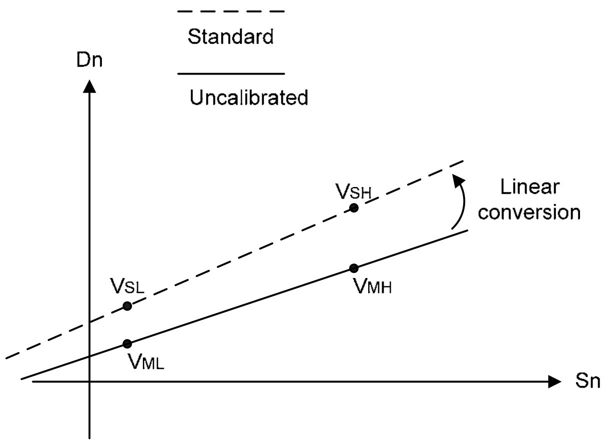
Fill the conversion parameter table with the low value of measurement(V_{ML}), high value of measurement(V_{MH}), and the corresponding low value of standard (V_{SL}), high value of standard(V_{SH}); the converted result(Dn) will be generated from the source data(Sn) through the formula shown below:

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

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

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

- The range of operands V_{SL},V_{SH}, V_{ML},V_{MH},Sn and Dn are between -32768 ~ 32767
- For analog input scaling, where
  V_{ML}=Minmum of analog input
  V_{MH}=Maximum of analog input
  V_{SL}=Minmum of engineering range
  V_{SH}=Maximum of engineering range



FUN33 P LCNV	Linear Conversion （LCNV）	FUN33 P LCNV

Expression 2 : Multiplicator＋Offset method

Fill the conversion parameter table with the values of multiplier(A), divisor(B) and offset(C);
The converted result(Dn) will be generated from the source data(Sn) through the formula shown below:

Dn =[(Sn×A)／B]＋C

The range of each operand as below:

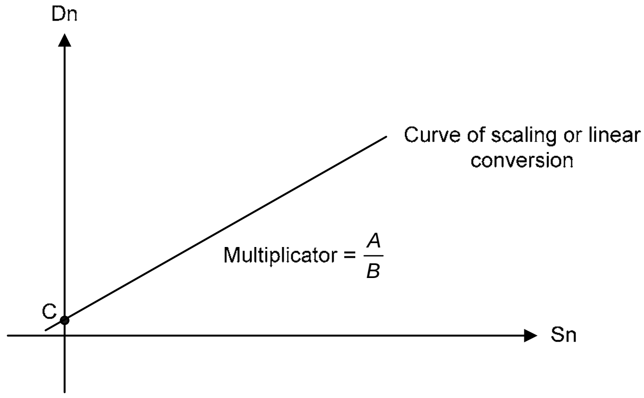
A = 1 ～ 65535

B = 1 ～ 65535

C = -32768 ～ 32767

Sn = 0 ～ 65535

Dn = -32768 ～ 32767

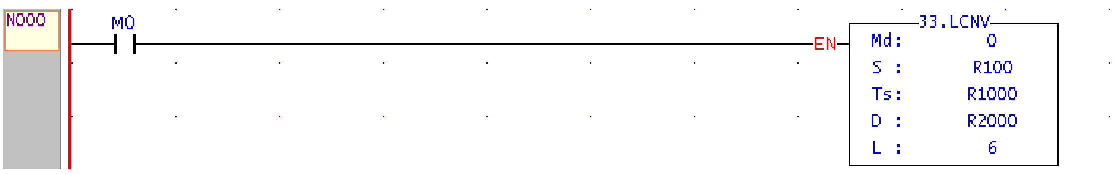


Description of operation mode :

1. When Md = 0, the linear conversion works by expression 1, and all source data share the same parameters VML、VMH、VSL and VSH for conversion.
2. When Md = 1, the linear conversion works by expression 1, and each source data has the independent corresponding parameters VML、VMH、VSL、VSH for conversion; if there are N entries of source data, the conversion parameter table should have N groups of VML、VMH、VSL、VSH for working, there are N×4 registers in the conversion parameter table.
3. When Md = 2, the linear conversion works by expression 2, and all source data share the same parameters A、B and C for conversion.
4. When Md = 3, the linear conversion works by expression 2, and each source data has the independent corresponding parameters A、B、C for conversion; if there are N entries of source data, the conversion parameter table should have N groups of A、B、C for working, there are N×3 registers in the conversion parameter table.

FUN33 P LCNV	Linear Conversion (LCNV)	FUN33 P LCNV

Example program 1 : Mode 0 of linear conversion



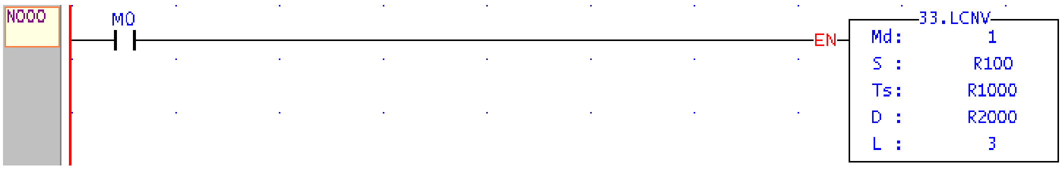
Description : When M0 = 1, it will perform the mode 0 operation of linear conversion, where R100 is the starting address of the source data, R1000 is the starting address of the table of the conversion parameters VML、VMH、VSL、VSH, the quantity is 6, and R2000～R2005 will store the converted results.

	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

FUN33 P LCNV	Linear Conversion (LCNV)	FUN33 P LCNV

Example program 2 : Mode 1 of linear conversion



Description : When M0 = 1, it will perform the mode 1 operation of linear conversion, where R100 is the starting address of the source data, R1000 is the starting address of the table of the conversion parameters VML、VMH、VSL、VSH, the quantity is 3, and R2000～R2002 will store the converted results.

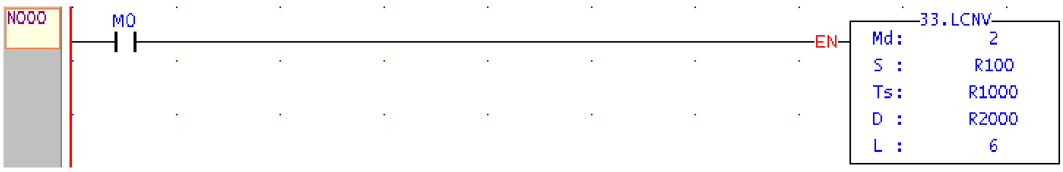
Ts

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	1101
R102	2399		R2002	2426

FUN33 P LCNV	Linear Conversion (LCNV)	FUN33 P LCNV

Example program 3 : Mode 2 of linear conversion



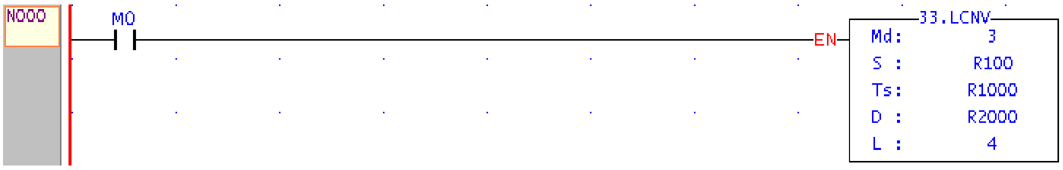
Description : When M0 = 1, it will perform the mode 2 operation of linear conversion, where R100 is the starting address of the source data, R1000 is the starting address of the table of the conversion parameters A、B、C, the quantity is 6, and R2000～R2005 will store the converted results.

	Ts	
R1000	985	A
R1001	1000	B
R1002	20	C

	S			D
R100	1000		R2000	1005
R101	2345		R2001	2330
R102	3560	⇨	R2002	3527
R103	401		R2003	415
R104	568		R2004	579
R105	2680		R2005	2660

FUN33 P LCNV	Linear Conversion （LCNV）	FUN33 P LCNV

Example program 4 : Mode 3 of linear conversion



Description : When M0 = 1, it will perform the mode 3 operation of linear conversion, where R100 is the starting address of the source data, R1000 is the starting address of the table of the conversion parameters A、B、C, the quantity is 4, and R2000～R2003 will store the converted results.

	Ts
R1000	5000
R1001	16380
R1002	0
R1003	10000
R1004	16383
R1005	0
R1006	2200
R1007	16380
R1008	-200
R1009	1600
R1010	16383
R1011	-100

	S			D
R100	8192		R2000	2501
R101	16383	⇨	R2001	10000
R102	8190		R2002	900
R103	0		R2003	-100

FUN34 P MLC	Multiple Linear Conversion (MLC)	FUN34 P MLC

34P. MLC

Execution Control EN ─ Rs : ─ OVR
SI :
Selection X/Y ─ Tx :
Ty :
Tl :
D :

Rs : Starting address of the source data
SI : Quantity of source data, 1~64
Tx : Starting address of X table
Ty : Starting address of Y table
Tl : Quantity of table, 2~255
D : Starting address to store the result

Operand \ Range	HR R0 – R3839	IR R3840 – R3903	ROR R5000 – R8071	DR D0 – D3999	K
Rs	○	○	○	○	
SI	○		○	○	1~64
Tx	○		○	○	
Ty	○		○*	○	
Tl	○		○	○	2～255
D	○		○	○	

- When the analog input module being used for the analog measurement, the raw reading value of the analog input can be converted into the engineering range through this instruction for display or for proceeding control operation.
- For process measurement calibration, making the linear conversion for the engineering process variable, which the measurement value from the PLC's can be corrected by the value from the standard meter's through this instruction.
- When execution control "EN"=1or from 0→1(P instruction), this instruction will perform the multiple linear conversion operation according to the selection of X/Y input; where Rs is the starting address of the source data, SI is the quantity of source data for conversion, Tx is the starting address of X conversion parameter table, Ty is the starting address of Y conversion parameter table, Tl is the quantity of X/Y table, D is the starting address to store the converted result.
- When executing and selection X/Y=0, it will compare the source data with the entities of Tx table to find the corresponding location in Tx table (The entities in Tx table must be in ascending sequence), and then calculate the linear conversion according to the located section of Tx and Ty table;
  When executing and selection X/Y=1, it will compare the source data with the entities of Ty table to find the corresponding location in Ty table (The entities in Ty table can either be in ascending or descending sequence), and then calculate the linear conversion according to the located section of Ty and Tx table.
- When the source data is out of all entities of table, OVR=1.
- It wouldn’t execute this instruction if illegal SI or Tl.

FUN34 P MLC	Multiple Linear Conversion (MLC)	FUN34 P MLC

Expression:

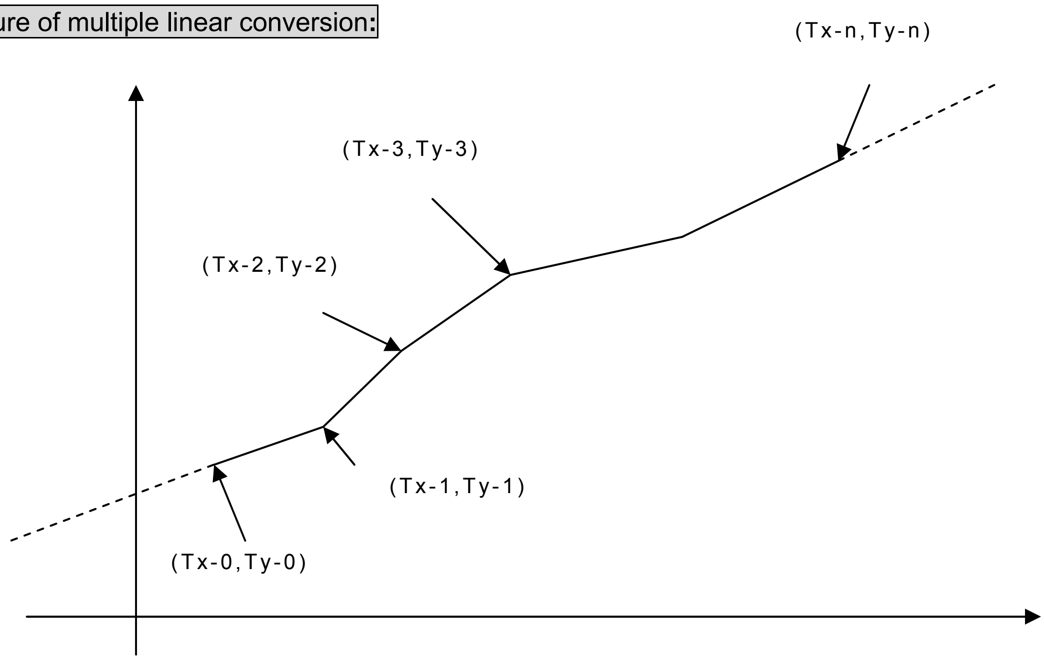
. The entities of Tx conversion parameter table must be in ascending sequence to have correct linear conversion; the entities of Ty conversion parameter table can either be in ascending or descending sequence. When executing this instruction, it will search the located section by comparing entities of the table with source data, and then calculate the linear conversion according to the following expression:

Vy = (Vx－Tx_n) × (Ty_n+1－Ty_n／Tx_n+1－Tx_n) ＋ Ty_n if X/Y=0

Vx = (Vy－Ty_n) × (Tx_n+1－Tx_n／Ty_n+1－Ty_n) ＋ Tx_n if X/Y=1

.Value of Vy、Vx、Tx_n、Tx_n+1、Ty_n、Ty_n+1 must be -32768～32767

Figure of multiple linear conversion:



FUN34 P MLC	Multiple Linear Conversion (MLC)	FUN34 P MLC

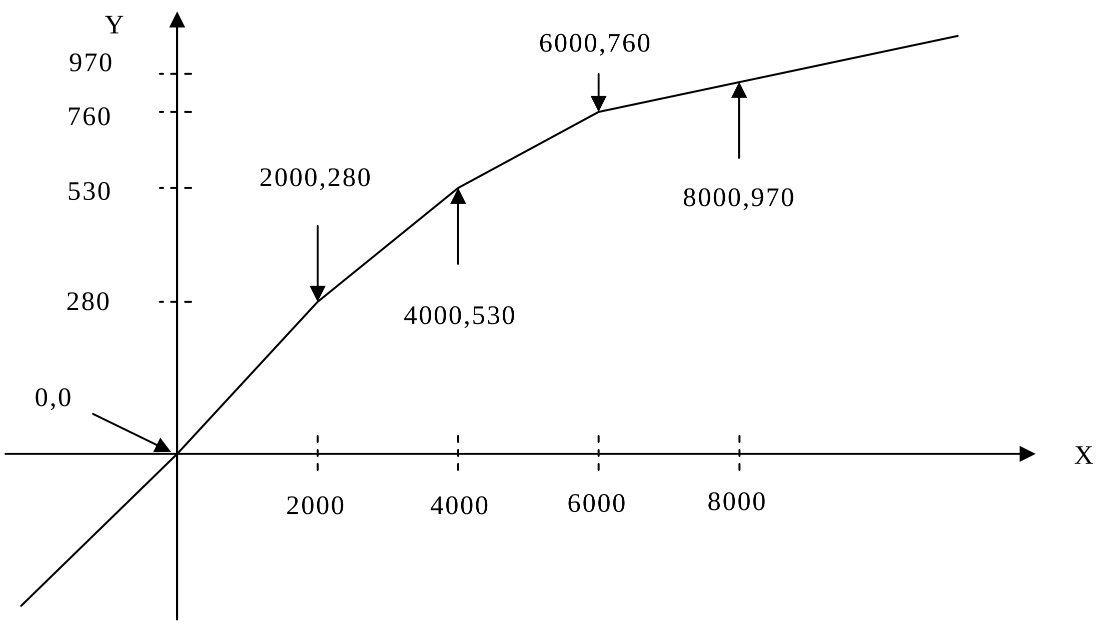
Example 1：

Description：When M10=1、M11=0, R0 is the starting address of source data、R99 is the quantity of source data, R1000 is the starting address of Tx conversion parameter table, R2000 is the starting address of Ty conversion parameter table、R199 is the quantity of table; the source data R0~R5 will be calculated the linear conversion according to Tx and Ty table between four sections, then store the results into D0～D5.

Status Monitoring

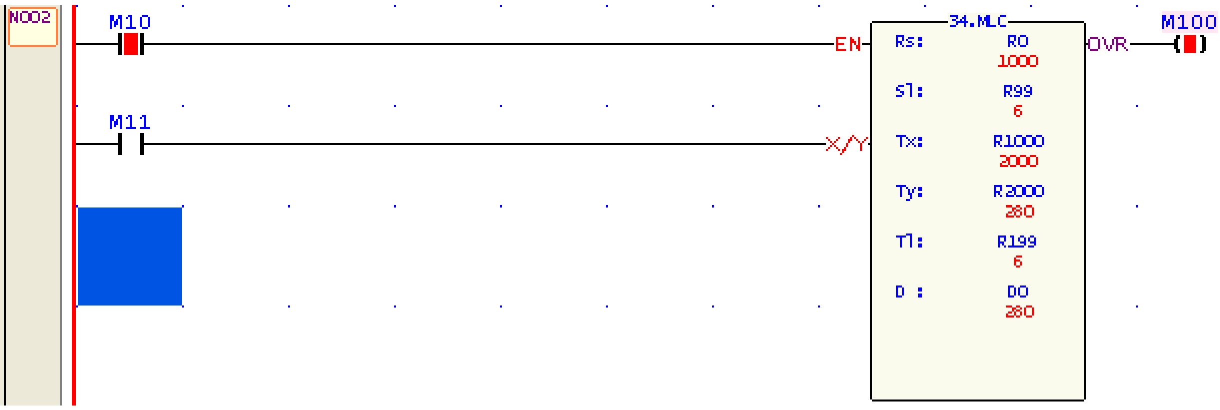
Ref. No.	Status	Data	Ref. No.	Status	Data	Ref. No	Status	Data	Ref. No	Status	Data
R1000	Decimal	0	R2000	Decimal	0	R0	Decimal	1000	D0	Decimal	140
R1001	Decimal	2000	R2001	Decimal	280	R1	Decimal	2500	D1	Decimal	342
R1002	Decimal	4000	R2002	Decimal	530	R2	Decimal	5600	D2	Decimal	714
R1003	Decimal	6000	R2003	Decimal	760	R3	Decimal	7500	D3	Decimal	917
R1004	Decimal	8000	R2004	Decimal	970	R4	Decimal	8000	D4	Decimal	970
R199	Decimal	5				R5	Decimal	10000	D5	Decimal	1180
M10	Enable	ON	M11	Enable	OFF	R99	Decimal	6			

StatusPage0 / StatusPage01 / StatusPage2

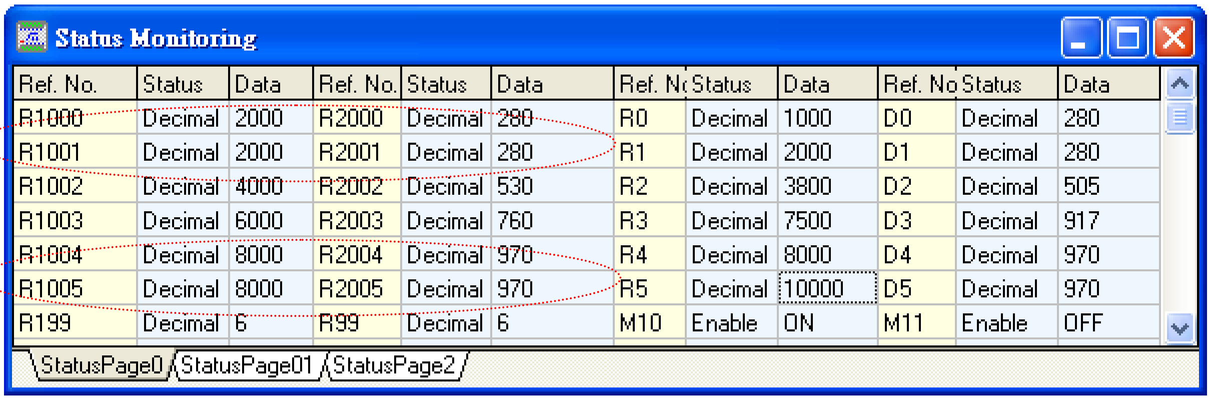


FUN34 P MLC	Multiple Linear Conversion (MLC)	FUN34 P MLC

Example 2：

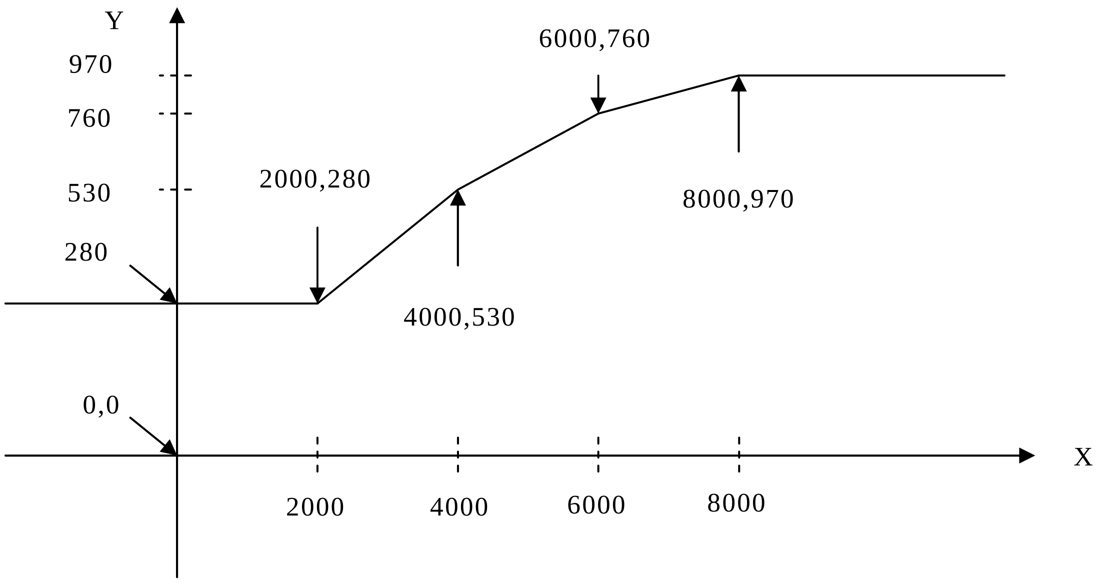


Description：When M10=1、M11=0, R0 is the starting address of source data、R99 is the quantity of source data, R1000 is the starting address of Tx conversion parameter table, R2000 is the starting address of Ty conversion parameter table、R199 is the quantity of table; the source data R0~R5 will be calculated the linear conversion according to Tx and Ty table between five sections, then store the results into D0～D5.The result value is 280 if source data ≦ 2000; the result value is 970 if source data ≧ 8000.


Status Monitoring

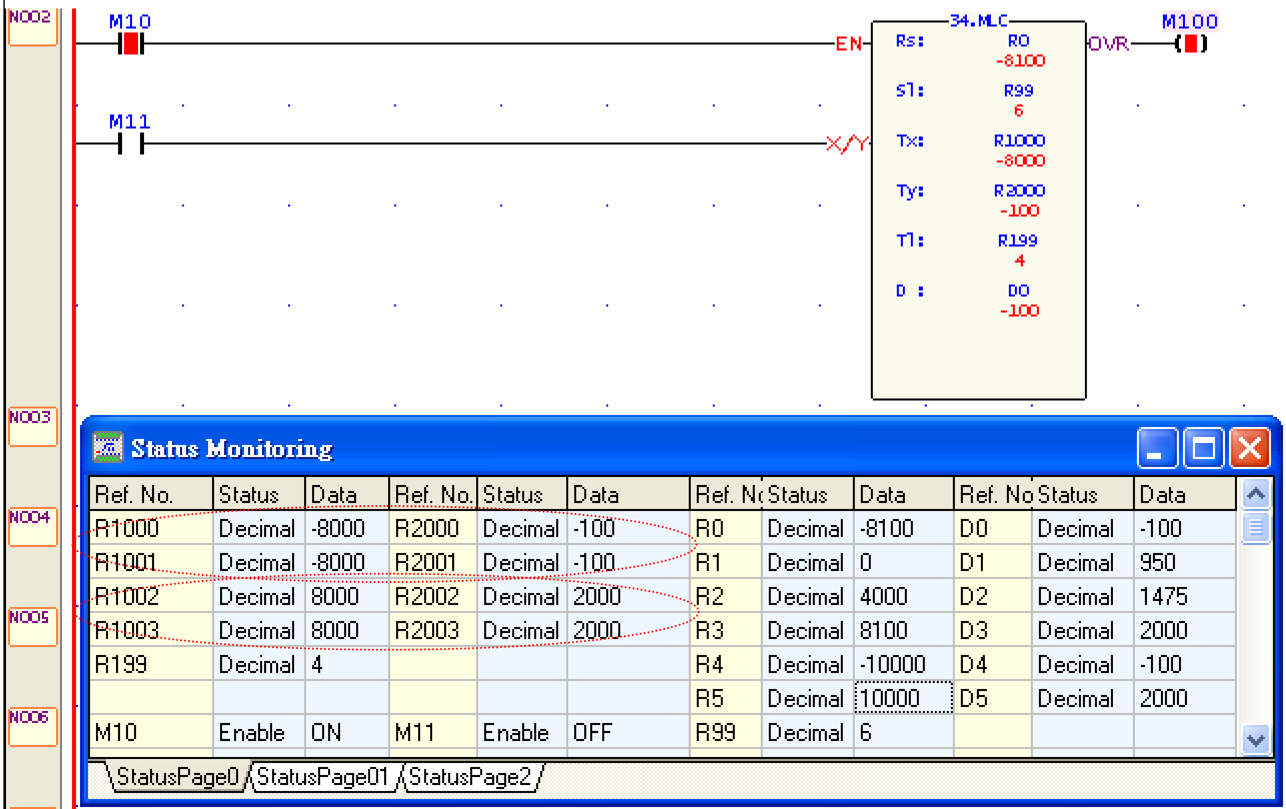
Ref. No.	Status	Data	Ref. No.	Status	Data	Ref. No	Status	Data	Ref. No	Status	Data
R1000	Decimal	2000	R2000	Decimal	280	R0	Decimal	1000	D0	Decimal	280
R1001	Decimal	2000	R2001	Decimal	280	R1	Decimal	2000	D1	Decimal	280
R1002	Decimal	4000	R2002	Decimal	530	R2	Decimal	3800	D2	Decimal	505
R1003	Decimal	6000	R2003	Decimal	760	R3	Decimal	7500	D3	Decimal	917
R1004	Decimal	8000	R2004	Decimal	970	R4	Decimal	8000	D4	Decimal	970
R1005	Decimal	8000	R2005	Decimal	970	R5	Decimal	10000	D5	Decimal	970
R199	Decimal	6	R99	Decimal	6	M10	Enable	ON	M11	Enable	OFF

StatusPage0 StatusPage01 StatusPage2



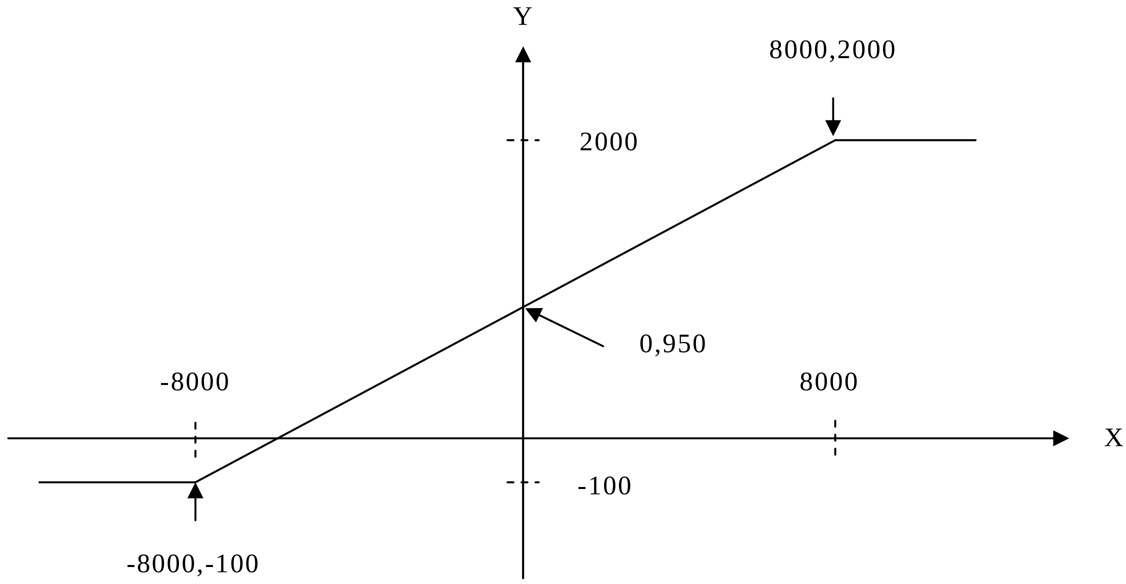
FUN34 P MLC	Multiple Linear Conversion (MLC)	FUN34 P MLC

Example 3：



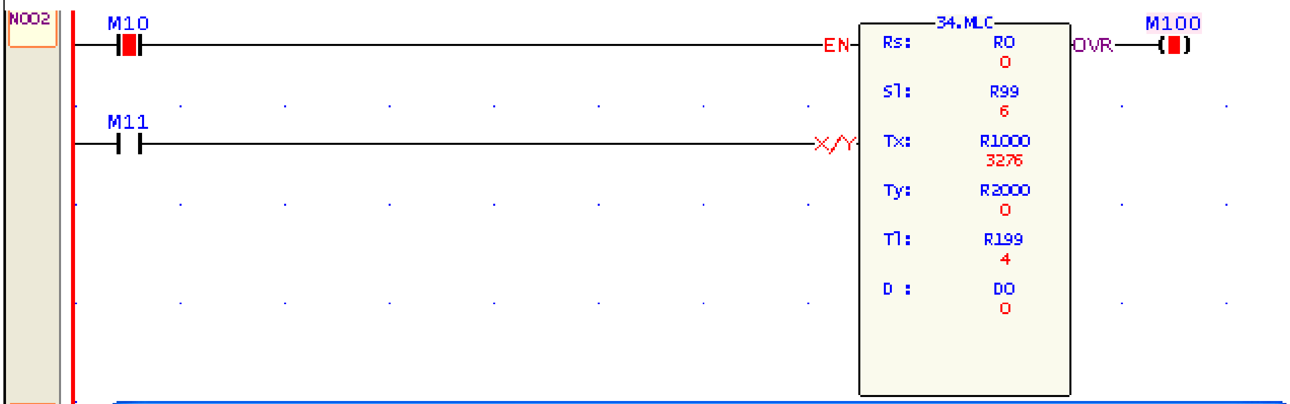
Ref. No.	Status	Data	Ref. No.	Status	Data	Ref. No	Status	Data	Ref. No	Status	Data
R1000	Decimal	-8000	R2000	Decimal	-100	R0	Decimal	-8100	D0	Decimal	-100
R1001	Decimal	-8000	R2001	Decimal	-100	R1	Decimal	0	D1	Decimal	950
R1002	Decimal	8000	R2002	Decimal	2000	R2	Decimal	4000	D2	Decimal	1475
R1003	Decimal	8000	R2003	Decimal	2000	R3	Decimal	8100	D3	Decimal	2000
R199	Decimal	4				R4	Decimal	-10000	D4	Decimal	-100
						R5	Decimal	10000	D5	Decimal	2000
M10	Enable	ON	M11	Enable	OFF	R99	Decimal	6			

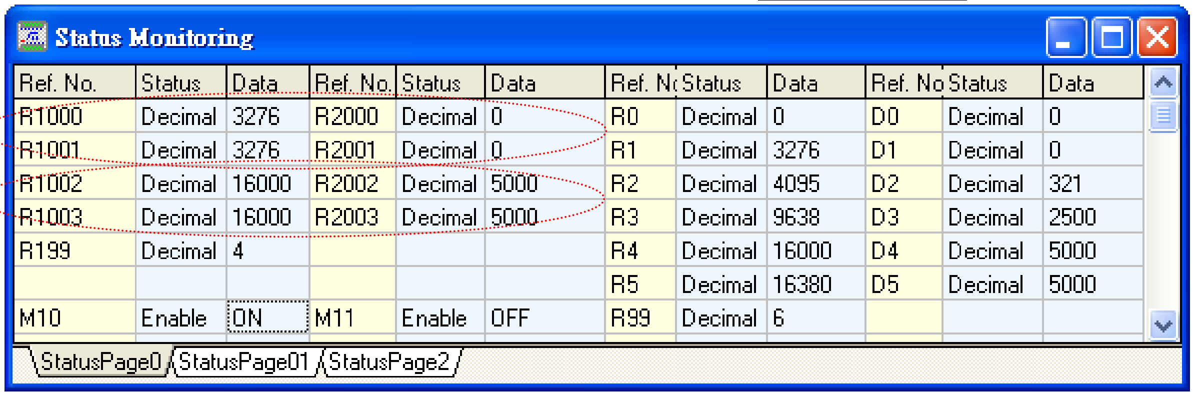
Description：When M10=1、M11=0, R0 is the starting address of source data、R99 is the quantity of source data, R1000 is the starting address of Tx conversion parameter table, R2000 is the starting address of Ty conversion parameter table、R199 is the quantity of table; the source data R0~R5 will be calculated the linear conversion according to Tx and Ty table between three sections, then store the results into D0～D5.The result value is -100 if source data ≦ -8000; the result value is 2000 if source data ≧ 8000.



FUN34 P MLC	Multiple Linear Conversion (MLC)	FUN34 P MLC

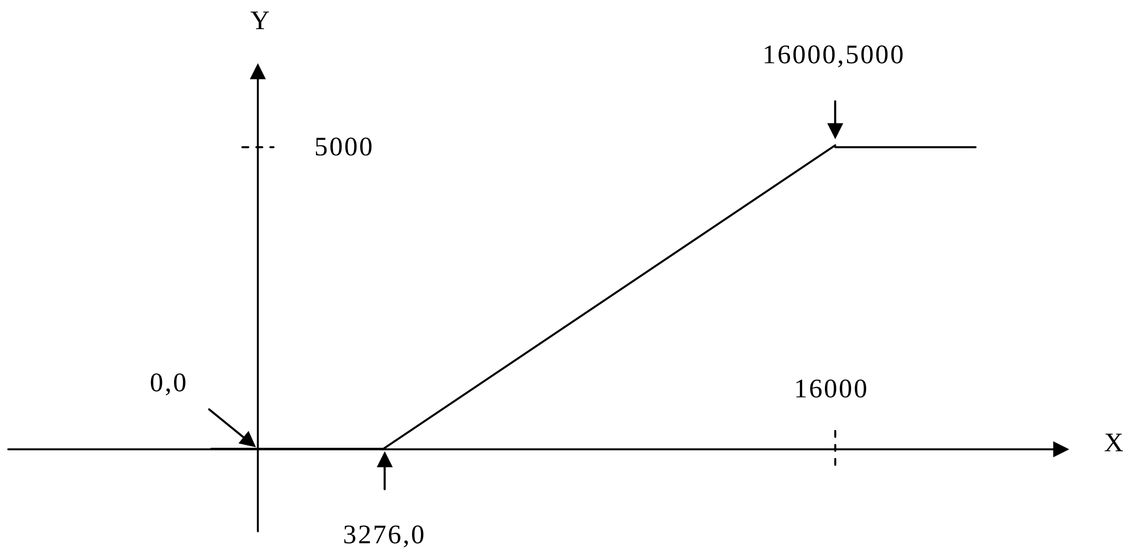
Example 4：





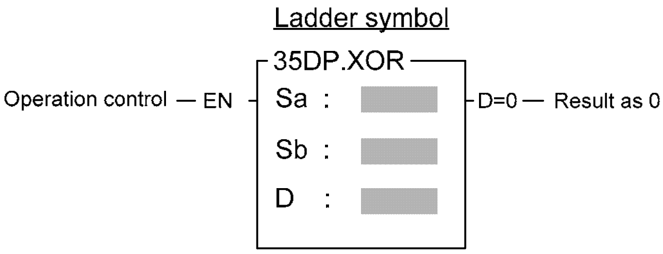
Ref. No.	Status	Data	Ref. No.	Status	Data	Ref. No	Status	Data	Ref. No	Status	Data
R1000	Decimal	3276	R2000	Decimal	0	R0	Decimal	0	D0	Decimal	0
R1001	Decimal	3276	R2001	Decimal	0	R1	Decimal	3276	D1	Decimal	0
R1002	Decimal	16000	R2002	Decimal	5000	R2	Decimal	4095	D2	Decimal	321
R1003	Decimal	16000	R2003	Decimal	5000	R3	Decimal	9638	D3	Decimal	2500
R199	Decimal	4				R4	Decimal	16000	D4	Decimal	5000
						R5	Decimal	16380	D5	Decimal	5000
M10	Enable	ON	M11	Enable	OFF	R99	Decimal	6			

Description：When M10=1、M11=0, R0 is the starting address of source data、R99 is the quantity of source data, R1000 is the starting address of Tx conversion parameter table, R2000 is the starting address of Ty conversion parameter table、R199 is the quantity of table; the source data R0~R5 will be calculated the linear conversion according to Tx and Ty table between three sections, then store the results into D0～D5.The result value is 0 if source data ≦ 3276; the result value is 5000 if source data ≧ 16000.



FUN 35 D P XOR	EXCLUSIVE OR	FUN 35 D P XOR

Ladder symbol



Sa : Source data a for exclusive or operation

Sb : Source data b for exclusive or operation

D : Register storing XOR results

Sa, Sb, D may combine with V, Z, P0~P9 to serve indirect address application

Range / Operand	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	16/32bit +/- number	V、Z P0~P9
Sa	○	○	○	○	○	○	○	○	○	○	○	○	○	○
Sb	○	○	○	○	○	○	○	○	○	○	○	○	○	○
D		○	○	○	○	○	○		○	○*	○*	○		○

- When operation control "EN" = 1 or changes from 0 to 1 (P instruction), will perform the logical XOR (exclusive or) operation of data Sa and Sb. The operation of this function is to compare the corresponding bits of Sa and Sb (B0~B15 or B0~B31), and if bits at the same position have different status, then set the corresponding bit within D as 1, otherwise as 0.
- After the operation, if all the bits in D are all 0, then set the 0 flag "D = 0" to 1.

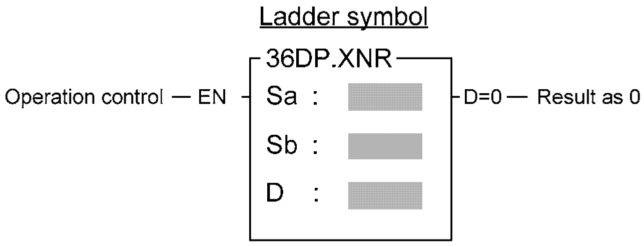
- The instruction at left makes a logical XOR operation using the R0 and R1 registers, and stores the result in R2.

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

FUN 36 D P XNR	EXCLUSIVE NOR	FUN 36 D P XNR



Sa : Data a for XNR operation

Sb : Data b for XNR operation

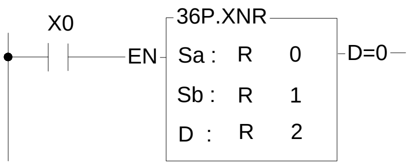
D : Register storing XNR results

Sa, Sb, D may combine with V, Z, P0~P9 to serve indirect address application

Range / Operand	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	16/32-bit ± number	V、Z P0~P9
Sa	○	○	○	○	○	○	○	○	○	○	○	○	○	○
Sb	○	○	○	○	○	○	○	○	○	○	○	○	○	○
D		○	○	○	○	○	○		○	○*	○*	○		○

- When operation control "EN" = 1 or changes from 0 to 1 (P instruction), will perform the logical XNR (inclusive or) operation of data Sa and Sb. The operation of this function is to compare the corresponding bits of Sa and Sb (B0~B15 or B1~B31), and if the bit has the same value, then set the corresponding bit within D as 1. If not then set it to 0.

- After the operation, if the bits in D are all 0, then set the 0 flag "D=0" to 1.



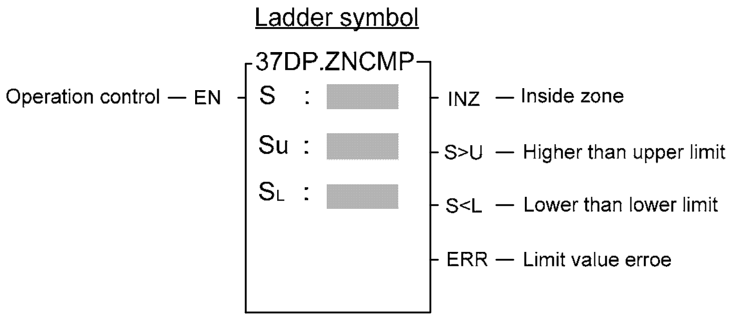
- The instruction at left makes a logical XNR operation of the R0 and R1 registers, and the results are stored in the R2 register.

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

FUN 37 D P ZNCMP	ZONE COMPARE	FUN 37 D P ZNCMP



S : Register for zone comparison

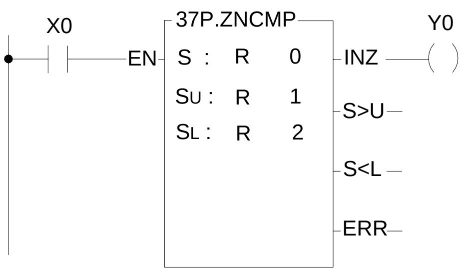
SU : The upper limit value

SL : The lower limit value

S, SU, SL may combine with V, Z, P0~P9 to serve indirect address application

Range / Ope-rand	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	16/32-bit +/- number	V、Z P0~P9
S	○	○	○	○	○	○	○	○	○	○	○	○		○
SU	○	○	○	○	○	○	○	○	○	○	○	○	○	○
SL	○	○	○	○	○	○	○	○	○	○	○	○	○	○

- When operation control "EN" = 1 or changes from 0 to 1 (P instruction), compares S with upper limit SU and lower limit SL. If S is between the upper limit and the lower limit ($S_L \leqq S \leqq S_U$), then set the inside zone flag "INZ" to 1. If the value of S is greater than the upper limit S_U, then set the higher than upper limit flag "S>U" to 1. If the value of S is smaller then the lower limit S_L, then set the lower than lower limit flag "S<L" as 1.
- The upper limit S_U should be greater than the lower limit S_L. If $S_U<S_L$, then the limit value error flag "ERR" will set to 1, and this instruction will not carry out.



- The instruction at left compares the value of R0 with the upper and lower limit zones formed by R1 and R2. If the values of R0~R2 are as shown in the diagram at bottom left, then the result can then be obtained as at the right of this diagram.
- If want to get the status of out side the zone, then OUT NOT Y0 may be used, or an OR operation between the two outputs S>U and S<L may be carried out, and move the result to Y0.

S	R0	200	
SU	R1	300	(Upper limit value)
SL	R2	100	(Lower limit value)

Before-execution

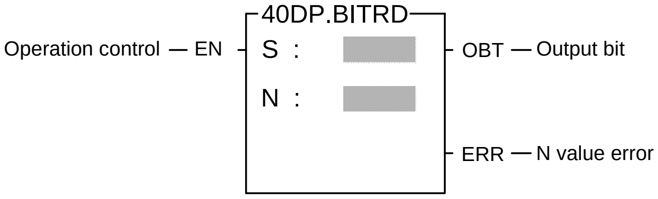
X0=↑ ☞

Y0 1

Results of execution

FUN 40 D P BITRD	BIT READ	FUN 40 D P BITRD

Ladder symbol



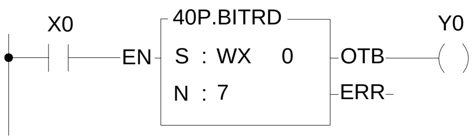
S : Source data to be read

N : The bit number of the S data to be read out.

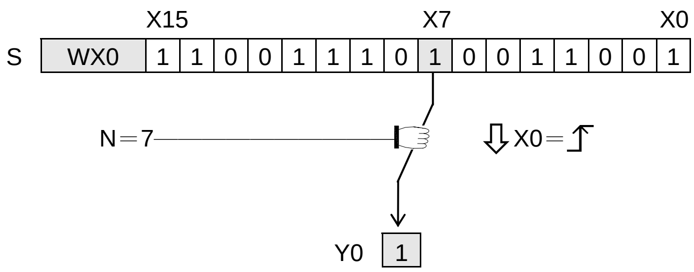
S, N may combine with V, Z, P0~P9 to serve indirect address application

Range / Operand	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	16/32-bit +/- number	V、Z P0~P9
S	○	○	○	○	○	○	○	○	○	○	○	○	○	○
N	○	○	○	○	○	○	○	○	○	○	○	○	0～31	○

- When read control "EN" = 1 or changes from 0 to 1 (P instruction), take the Nth bit of the S data out , and put it to the output bit "OTB".
- When read control "EN" = 0, the output "OTB" can be selected to keep at the last state (if M1919=0) or set to zero (if M1919=1).
- When the operand is 16 bit, the effective range for N is 0~15. For 32 bit operand (D instruction) it is 0~31. N beyond this range will set the N value error flag "ERR" to 1, and do not carry out this instruction.



- The instruction at left reads the 7th bit (X7) status from WX0 (X0~X15) and output to Y0. The results are as follows:



FUN 41 D P BITWR	BIT WRITE	FUN 41 D P BITWR

Ladder symbol

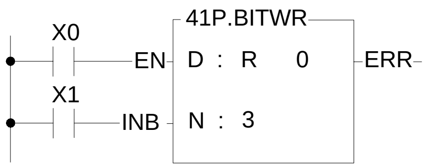
D : Register for bit write

N : The bit number of the D register to be written.

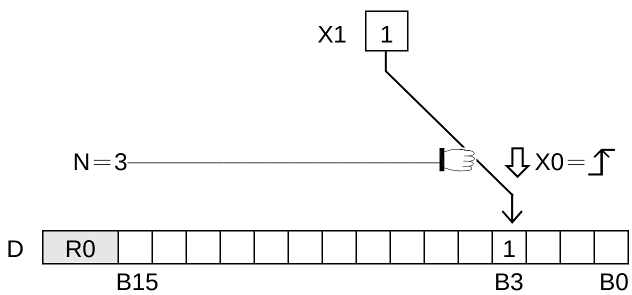
D, N may combine with V, Z , P0~P9 to serve indirect address application

Range / Operand	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 ∣ 15 or 0 ∣ 31	V、Z P0~P9
D		○	○	○	○	○	○		○	○*	○*	○		○
N	○	○	○	○	○	○	○	○	○	○	○	○	○	○

- When write control "EN" = 1 or changes from 0 to 1 (P instruction), will write the write bit (INB) into the Nth bit of register D.
- When the operand is 16 bit, the effective range of N is 0~15. For 32 bit (D instruction) operand it is 0~31. N beyond this range, will set the N value error flag "ERR" to 1, and do not carry out this instruction.

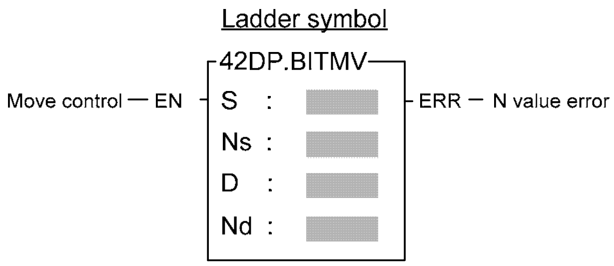


- The instruction at left writes the status of the write bit INB into B3 of R0. Assuming X1 = 1, the result will be as follows:



Bits other than B3 remain unchanged

FUN 42 D P BITMV	BIT MOVE	FUN 42 D P BITMV



S : Source data to be moved

Ns: Assign Ns bit within S as source bit

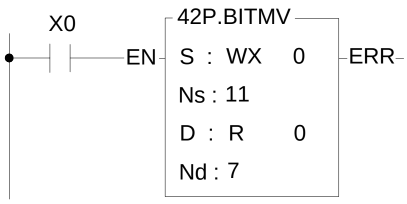
D : Destination register to be moved

Nd: Assign Nd bit within D as target bit

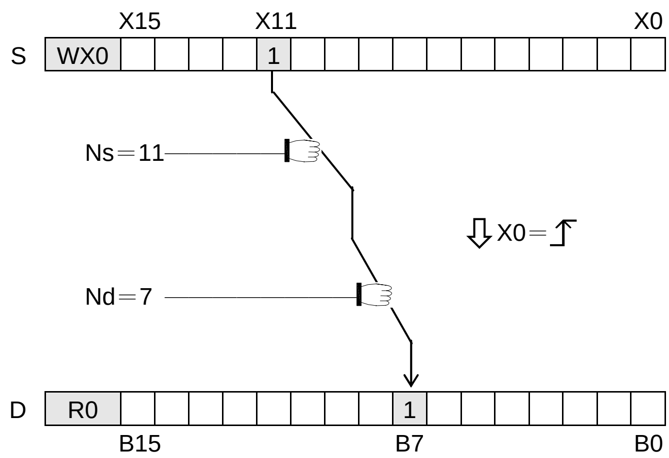
S, Ns, D, Nd may combine with V, Z, P0~P9 to serve indirect address application

Range / Ope-rand	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	16/32-bit +/- number	V、Z P0~P9
S	○	○	○	○	○	○	○	○	○	○	○	○	○	○
Ns	○	○	○	○	○	○	○	○	○	○	○	○	0～31	○
D		○	○	○	○	○	○		○	○*	○*	○		○
Nd	○	○	○	○	○	○	○	○	○	○	○	○	0～31	○

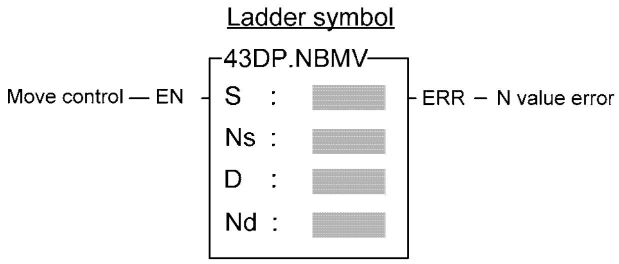
- When move control "EN" = 1 or changes from 0 to 1 (P instruction), will move the bit status specified by Ns within S into the bit specified by Nd within D.
- When the operand is 16 bit, the effective range of N is 0~15. For 32 bit (D instruction) operand the effective range is 0~31. N beyond this range will set the N value error flag "ERR" to 1, and do not carry out this instruction.



- The instruction at left moves the status of B11 (X11) within S into the B7 position within D. Except bit B7, other bits within D does not change.



FUN 43 D P NBMV	NIBBLE MOVE	FUN 43 D P NBMV



S : Source data to be moved

Ns: Assign Ns nibble within S as source nibble

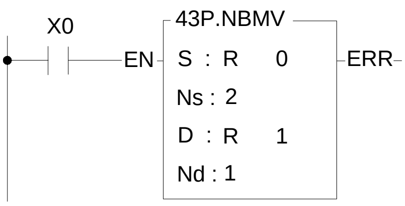
D : Destination register to be moved

Nd: Assign Nd nibble within D as target nibble

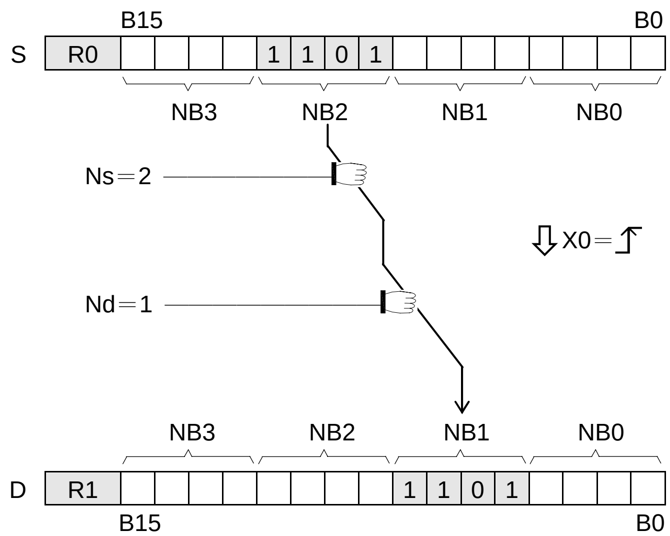
S, Ns, D, Nd may combine with V, Z, P0~P9 to serve indirect address application

Range / Ope-rand	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	16/32-bit +/- number	V、Z P0~P9
S	○	○	○	○	○	○	○	○	○	○	○	○	○	○
Ns	○	○	○	○	○	○	○	○	○	○	○	○	0～7	○
D		○	○	○	○	○	○		○	○*	○*	○		○
Nd	○	○	○	○	○	○	○	○	○	○	○	○	0～7	○

- When move control "EN" = 1 or has a transition from 0 to 1 (P instruction), will move the Ns'th nibble from within S to the nibble specified by Nd within D. (A nibble is comprised by 4 bits. Starting from the lowest bit of the register, B0, each successive 4 bits form a nibble, so B0~B3 form nibble 0, B4~B7 form nibble 1, etc...)
- When the operand is 16 bit, the effective range of Ns or Nd is 0~3. For 32 bit (D instruction) operand the range is 0~7. Beyond this range, will set the N value error flag "ERR" to 1 , and do not carry out this instruction.

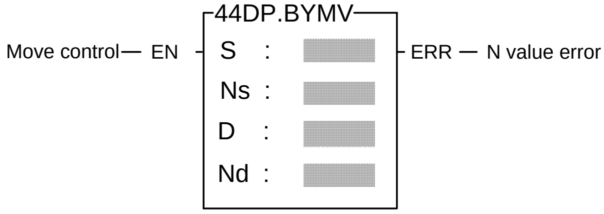


- The instruction at left moves the third nibble NB2 (B8~B11) within S to the first nibble NB1 (B4~B7) within D. Other nibbles within D remain unchanged.



FUN 44 D P BYMV	BYTE MOVE	FUN 44 D P BYMV

Ladder symbol



S : Source data to be moved

Ns : Assign Ns byte within S as source byte

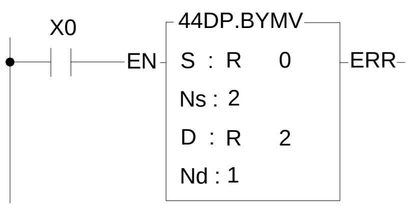
D : Destination register to be moved

Nd : Assign Nd byte within D as target byte

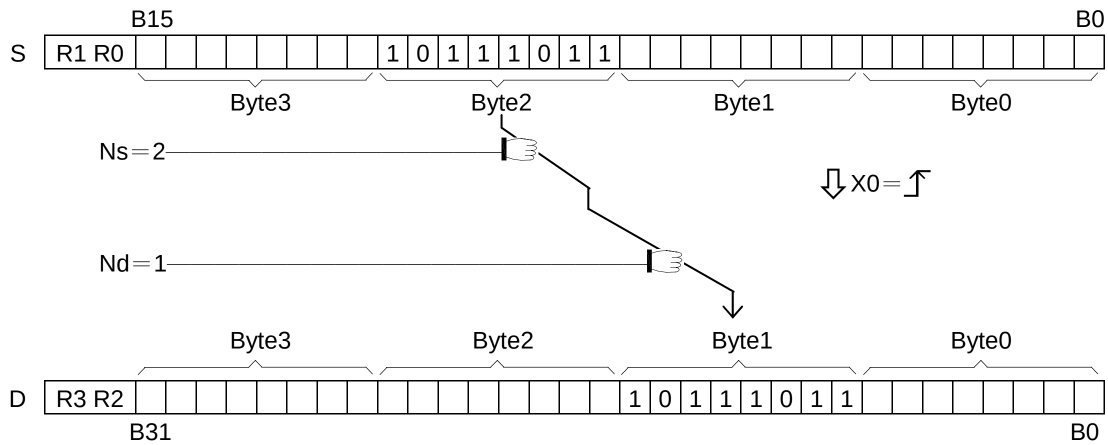
S, Ns, D, Nd may combine with V, Z, P0~P9 to serve indirect address application

Range / Ope-rand	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	16/32-bit +/- number	V、Z P0~P9
S	○	○	○	○	○	○	○	○	○	○	○	○	○	○
Ns	○	○	○	○	○	○	○	○	○	○	○	○	0～3	○
D		○	○	○	○	○	○		○	○*	○*	○		○
Nd	○	○	○	○	○	○	○	○	○	○	○	○	0～3	○

- When move control "EN" = 1 or has a transition from 0 to 1 (P instruction), move Nsth byte within S to Ndth byte position within D. (A byte is comprised of 8 bits. Starting from the lowest bit of the register, B0, each successive eight bits form a byte, so B0~B7 form byte 0, B8~B15 form byte 1, etc...)
- When the operand is 16 bit, the effective range of Ns or Nd is 0~1. For 32 bit (D instruction) operand, the range is 0~3. Beyond this range, will set the N value error flag "ERR" to 1, and do not carry out this instruction.

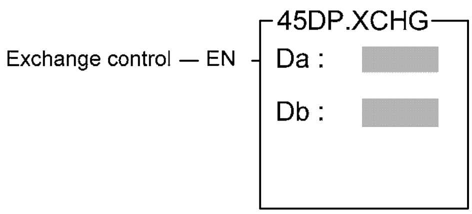


- The instruction at left moves the third byte (B16~B23) within S (32 bit register composed of R1R0), to the first byte within D (32 bit register composed of R3R2). Other bytes within D remain unchanged.



FUN 45 D P XCHG	EXCHANGE	FUN 45 D P XCHG

Ladder symbol



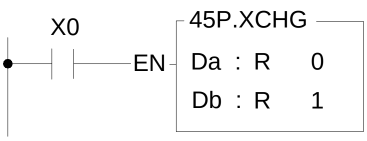
Da : Register a to be exchanged

Db : Register b to be exchanged

Da, Db may combine with V, Z, P0~P9 to serve indirect address application

Range / Operand	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	○	○	○	○	○	○	○	○*	○*	○	○

- When exchange control "EN" = 1 or has a transition from 0 to 1 (P instruction), will exchanges the contents of register Da and register Db in 16 bits or 32 bits (D instruction) format.



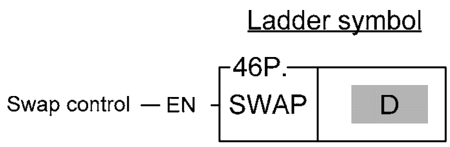
- The instruction at left exchanges the contents of the 16-bit R0 and R1 registers.

		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

FUN 46 P SWAP	BYTE SWAP	FUN 46 P SWAP

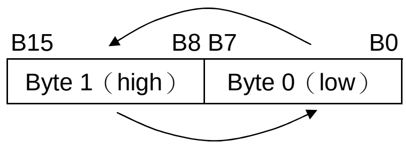


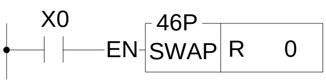
D : Register for byte data swap

D may combine with V, Z, P0~P9 to serve indirect address application

Range / Operand	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	○	○	○	○	○	○	○	○*	○*	○	○

- When swap control "EN" = 1 or has a transition from 0 to 1 (P instruction), swap the data of the low byte, Byte 0 (B0~B7), and the high byte, Byte 1 (B8~B15), in the 16 bit register specified by D.





- The instruction at left swaps the data of the low byte (B0~B7) and the high byte (B8~B15) in R0. The results are as follows:

Byte1 (B15~B8), Byte0 (B7~B0)

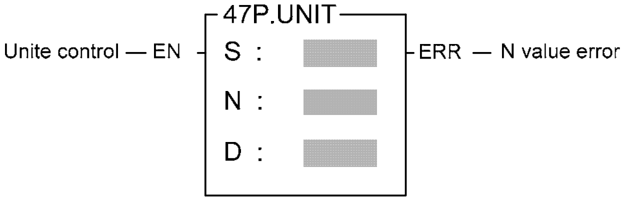
		B15							B8	B7							B0
D	R0	0	0	1	1	0	0	1	1	1	1	1	1	0	0	0	0

⇩ X0 = ↑

		B15															B0
D	R0	1	1	1	1	0	0	0	0	0	0	1	1	0	0	1	1

FUN 47 P UNIT	NIBBLE UNITE	FUN 47 P UNIT

Ladder symbol



S : Starting source register to be united

N : Number of nibbles to be united

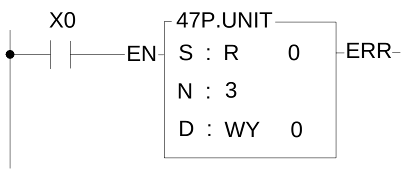
D : Registers storing united data

S, N, D may combine with V, Z, P0~P9 to serve indirect address application

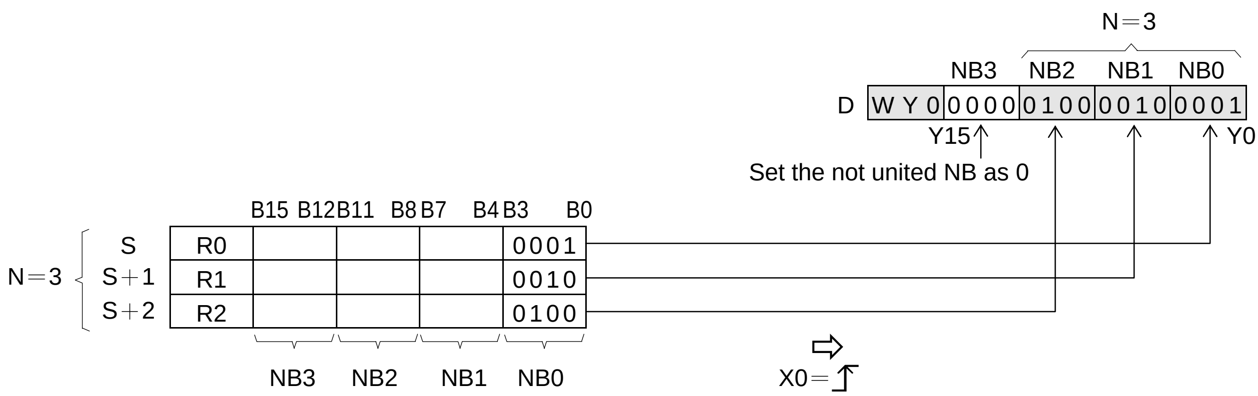
Range Ope-rand	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR
	WX0 | WX240	WY0 | WY240	WM0 | WM1896	WS0 | WS984	T0 | T255	C0 | C255	R0 | R3839	R3840 | R3903	R3904 | R3967	R3968 | R4167	R5000 | R8071	D0 | D4095	1 | 4	V、Z P0~P9
S	○	○	○	○	○	○	○	○	○	○	○	○		○
N	○	○	○	○	○	○	○	○	○	○	○	○	○	○
D		○	○	○	○	○	○		○	○*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

- When unite control "EN" = 1 or has a transition from 0 to 1 (P instruction), take out the lowest nibbles NB0, of N successive registers starting from S, and fill them into NB0, NB1, ...NBn-1 of D in ascending order. Nibbles not yet filled in D (when N is odd) are filled with 0. (A nibble is comprised by 4 bits. Starting from the lowest bit in the register, B0, each successive four bits form a nibble, so B0~B3 form nibble 0, B4~B7 form nibble 1, etc...).

- This instruction only provides WORD (16 bit) operand. Because of this, there are usually only 4 nibbles can be involved. Therefore the effective range of N is 1~4. Beyond this range, will set the N value error flag "ERR" to 1, and do not carry out this instruction.

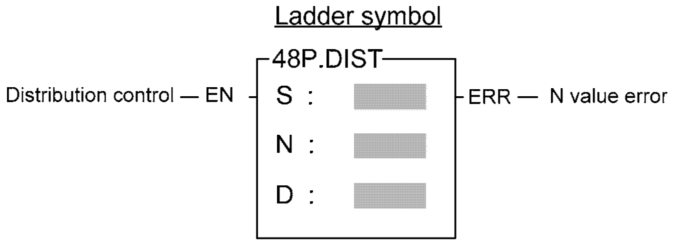


- The instruction at left takes out NB0 from 3 registers, R0, R1 and R2, and fills them into NB0~NB2 within WY0 register.



FUN 48 P DIST	NIBBLE DISTRIBUTE	FUN 48 P DIST

Ladder symbol



S : Source data to be distributed

N : Number of nibbles to be distributed

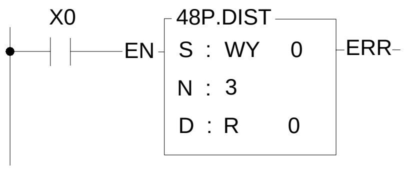
D : Starting register storing distribution data

S, N, D may combine with V, Z, P0~P9 to serve indirect address application

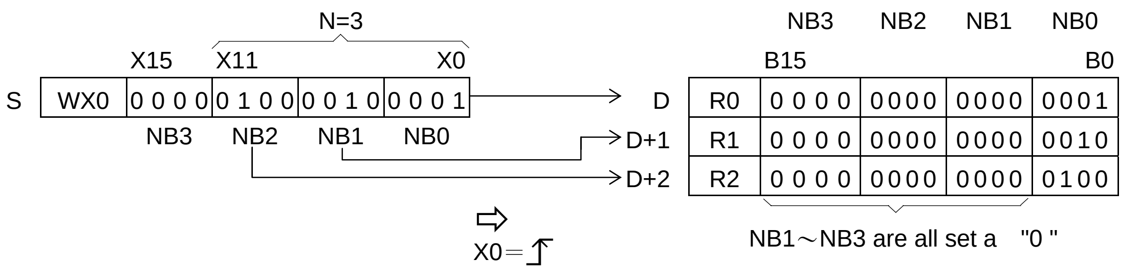
Range / Operand	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR
	WX0 | WX240	WY0 | WY240	WM0 | WM1896	WS0 | WS984	T0 | T255	C0 | C255	R0 | R3839	R3840 | R3903	R3904 | R3967	R3968 | R4167	R5000 | R8071	D0 | D4095	16-bit +/- number	V、Z P0~P9
S	○	○	○	○	○	○	○	○	○	○	○	○	○	○
N	○	○	○	○	○	○	○	○	○	○	○	○	1~4	○
D		○	○	○	○	○	○		○	○*	○*	○		○

- When distribution control "EN" = 1 or has a transition from 0 to 1(P instruction), will take N successive nibbles starting from the lowest nibble NB0 within S, and distribute them in ascending order into the 0 nibbles of N registers starting from D. The nibbles other than NB0 in each of the registers within D are all set to zero. (A nibble is comprised by 4 bits. Starting from the lowest bit in a register, B0, each successive 4 bits form a nibble, so B0~B3 form nibble 0, B4~B7 form nibble 1, etc...)

- This instruction only provides WORD (16 bit) operand. Therefore there are usually only 4 nibbles can be involved, so the effective value of N is 1~4. Beyond this range, will set the N value error flag "ERR" to 1, and do not carry out this instruction.

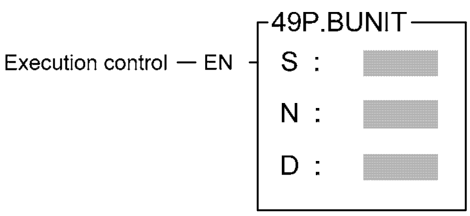


- The instruction at left writes NB0~NB2 from the WX0 register into the NB0 of the 3 consecutive registers R0~R2.



FUN49 P BUNIT	BYTE UNITE	FUN49 P BUNIT

Ladder symbol



S ：Starting address of source register to be united

N ：Number of bytes to be united

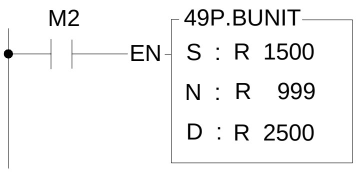
D ：Registers to store the united data

S, N, D may associate with V、Z、P0~P9 index register to serve the indirect addressing application

Range / Operand	HR R0 ∣ R3839	ROR R5000 ∣ R8071	DR D0 ∣ D4095	K
S	○	○	○	
N	○	○	○	1~256
D	○	○*	○	

- When execution control "EN"=1 or changes from 0→1 P instruction, it will perform the byte combination starting from S, length by N, and then store the results into D registers.
- This instruction will not act if invalid range of length.
- When communicating with intelligent peripheral in binary data format, this instruction may be applied to do byte combination for following word data processing.

Example：



Description：When M2 changes from 0→1, it will perform the byte combination starting from R1500, the length is assigned by R999, and then store the results into registers starting from R2500.
It is supposed R999=10, the results of combination will store into R2500～R2504.

S

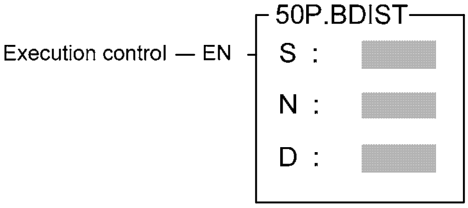
	High Byte	Low Byte
R1500	Don't care	Byte-0
R1501	Don't care	Byte-1
R1502	Don't care	Byte-2
R1503	Don't care	Byte-3
R1504	Don't care	Byte-4
R1505	Don't care	Byte-5
R1506	Don't care	Byte-6
R1507	Don't care	Byte-7
R1508	Don't care	Byte-8
R1509	Don't care	Byte-9

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	BYTE DISTRIBUTE	FUN50 P BDIST

Ladder symbol



S : Starting address of source register to be distributed

N : Number of bytes to be distributed

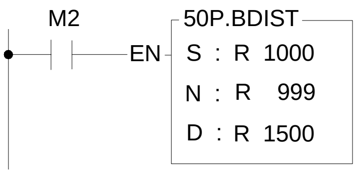
D : Registers to store the distributed data

S, N, D may associate with V、Z、P0~P9 index register to serve the indirect addressing application.

Range / Operand	HR R0 ∣ R3839	ROR R5000 ∣ R8071	DR D0 ∣ D4095	K
S	○	○	○	
N	○	○	○	1~256
D	○	○*	○	

- When execution control "EN" =1 or changes from 0→1 (P instruction) , it will perform the byte distribution starting from S, length by N, and then store the results into D registers.
- This instruction will not act if invalid range of length.
- When communicating with intelligent peripheral in binary data format, this instruction may be applied to do byte distribution for data transmission。

Example：



Description：When M2 changes from 0→1, it will perform the byte distribution starting from R1000, the length is assigned by R999, and then store the results into registers starting from R1500.
It is supposed R999=9, the results of distribution will store into R1500～R1508.

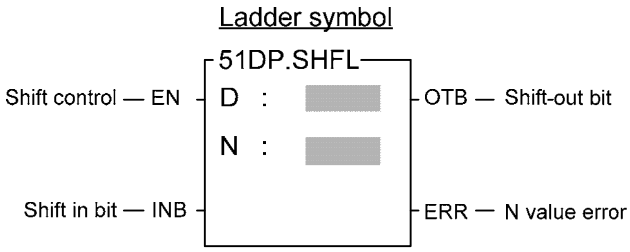
S

	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

FUN 51 D P SHFL	SHIFT LEFT	FUN 51 D P SHFL



D : Register to be shifted

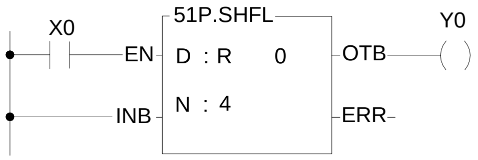
N : Number of bits to be shifted

N, D may combine with V, Z, P0~P9 to serve indirect address application

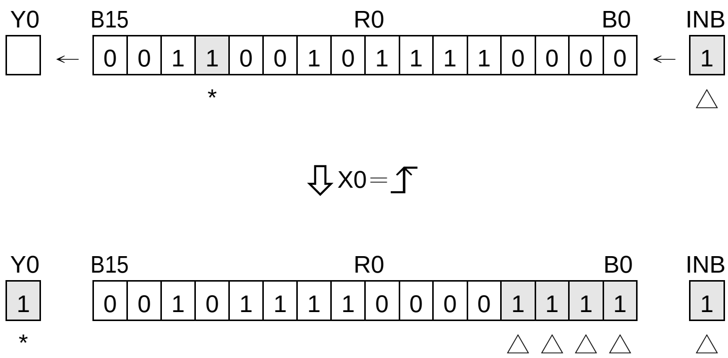
Range / Ope-rand	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR
	WX0 | WX240	WY0 | WY240	WM0 | WM1896	WS0 | WS984	T0 | T255	C0 | C255	R0 | R3839	R3840 | R3903	R3904 | R3967	R3968 | R4167	R5000 | R8071	D0 | D4095	1 | 16 or 1 | 32	V、Z P0~P9
D		○	○	○	○	○	○		○	○*	○*	○		○
N	○	○	○	○	○	○	○	○	○	○	○	○	○	○

- When shift control "EN" = 1 or has a transition from 0 to 1(P instruction), will shift the data of the D register towards the left by N successive bits (in ascending order). After the lowest bit B0 has been shifted left, its position will be replaced by shift-in bit INB, while the status of shift-out bits B15 or B31 (D instruction) will appear at shift-out bit "OTB".

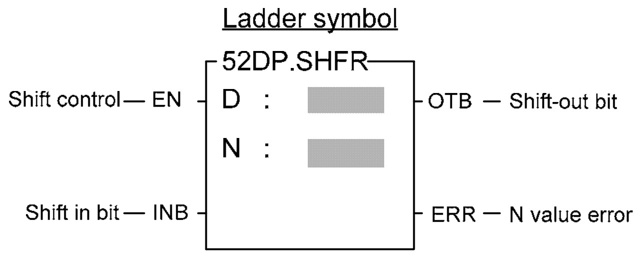
- If the operand is 16 bit, the effective range of N is 1~16. For 32 bits (D instruction) operand, it is 1~32. Beyond this range, will set the N value error flag "ERR" to 1, and do not carry out this instruction.



- The instruction at left shifts the data in register R0 towards the left by 4 successive bits. The results are shown below.



FUN 52 D P SHFR	SHIFT RIGHT	FUN 52 D P SHFR



D : Register to be shifted

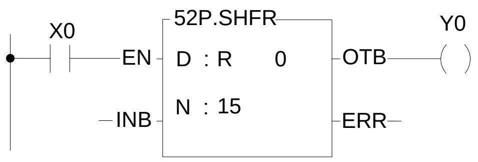
N : Number of bits to be shifted

D, N may combine with V, Z, P0~P9 to serve indirect address application

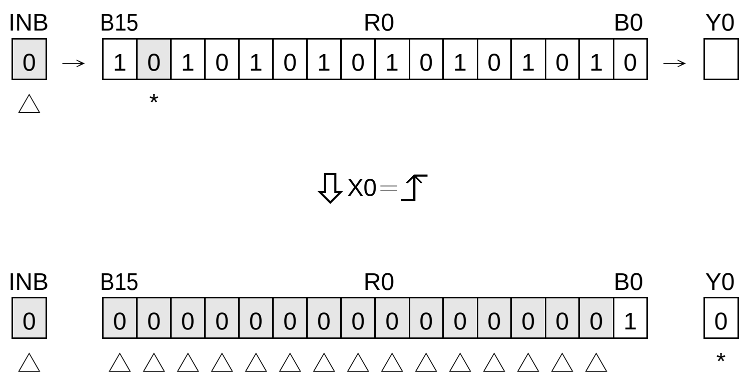
Range / Operand	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR
	WX0 ∣ WX240	WY0 ∣ WY240	WM0 ∣ WM1896	WS0 ∣ WS984	T0 ∣ T255	C0 ∣ C255	R0 ∣ R3839	R3840 ∣ R3903	R3904 ∣ R3967	R3968 ∣ R4167	R5000 ∣ R8071	D0 ∣ D4095	1 ∣ 16 or 1 ∣ 32	V、Z P0~P9
D		○	○	○	○	○	○		○	○*	○*	○		○
N	○	○	○	○	○	○	○	○	○	○	○	○	○	○

- When shift control "EN" = 1 or has a transition from 0 to 1 (P instruction), will shift the data of D register towards the right by N successive bits (in descending order). After the highest bits, B15 or B31 (D instruction) have been shifted right, their positions will be replaced by the shift-in bit INB, while shift-out bit B0 will appear at shift-out bit "OTB".

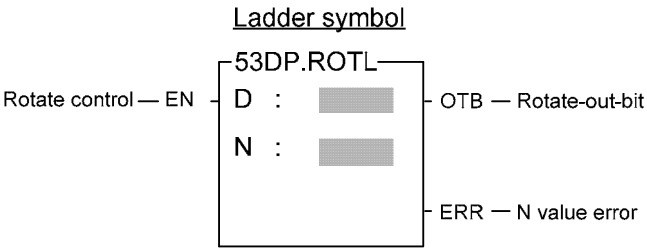
- If the operand is 16 bit, the effective range of N is 1~16. For 32 bits (D instruction) operand, it is 1~32. Beyond this range, will set the N value error flag "ERR" to 1, and do not carry out this instruction.



- The instruction at left shifts the data in R0 register towards the right by 15 successive bits. The results are shown below.



FUN 53 D P ROTL	ROTATE LEFT	FUN 53 D P ROTL



D : Register to be rotated

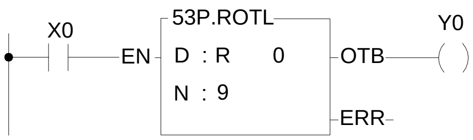
N : Number of bits to be rotated

D, N may combine with V, Z , P0~P9 to serve indirect address application

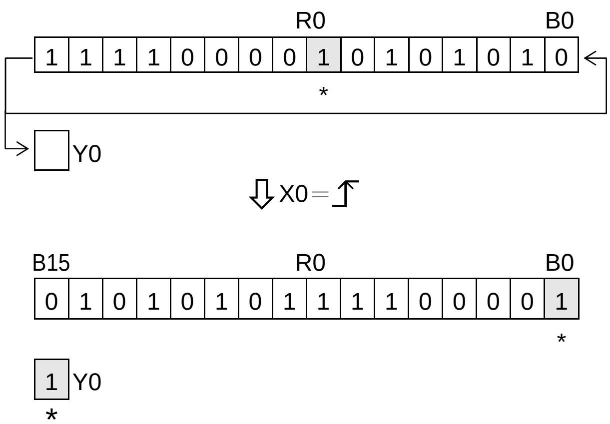
Operand \ Range	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR
	WX0 | WX240	WY0 | WY240	WM0 | WM1896	WS0 | WS984	T0 | T255	C0 | C255	R0 | R3839	R3840 | R3903	R3904 | R3967	R3968 | R4167	R5000 | R8071	D0 | D4095	1 | 16 or 1 | 32	V、Z P0~P9
D		○	○	○	○	○	○		○	○*	○*	○		○
N	○	○	○	○	○	○	○	○	○	○	○	○	○	○

- When rotate control "EN" = 1 or has a transition from 0 to 1 (P instruction), will rotate the data of D register towards the left by N successive bits (in ascending order, ie. in a 16-bit instruction, B0→B1, B1→B2, ... , B14→B15, B15→B0. In a 32-bit instruction, B0→B1, B1→B2, ... , B30→B31, B31→B0). At the same time, the status of the rotated out bits B15 or B31 (D instruction) will appear at rotate-out bit "OTB".

- If the operand is 16 bit, the effective range of N is 1~16. For 32 bits (D instruction) operand, it is 1~32. Beyond this range, will set the N value error flag "ERR" to 1, and do not carry out this instruction.

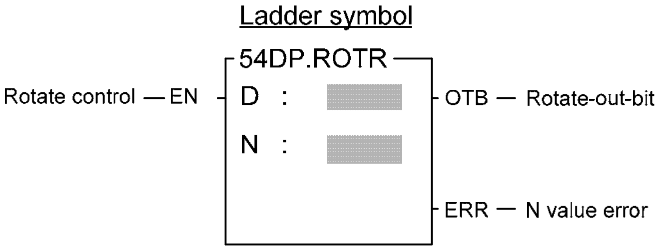


- The instruction at left rotates data from the R0 register towards the left 9 successive bits. The results are shown below.



FUN 54 D P ROTR	ROTATE RIGHT	FUN 54 D P ROTR

Ladder symbol



D : Register to be rotated

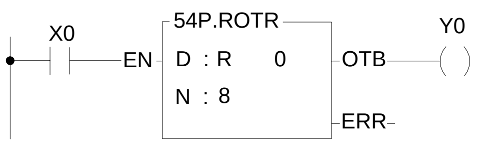
N : Number of bits to be rotated

D, N may combine with V, Z, P0~P9 to serve indirect address application

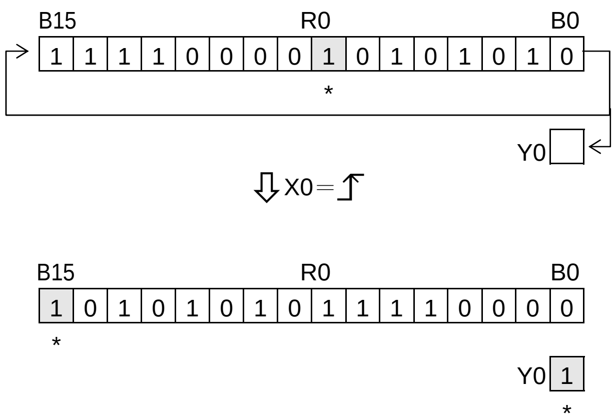
Range / Operand	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR
	WX0 ~ WX240	WY0 ~ WY240	WM0 ~ WM1896	WS0 ~ WS984	T0 ~ T255	C0 ~ C255	R0 ~ R3839	R3840 ~ R3903	R3904 ~ R3967	R3968 ~ R4167	R5000 ~ R8071	D0 ~ D4095	1 ~ 16 or 1 ~ 32	V、Z P0~P9
D		○	○	○	○	○	○		○	○*	○*	○		○
N	○	○	○	○	○	○	○	○	○	○	○	○	○	○

- When rotate control "EN" = 1 or has a transition from 0 to 1(P instruction), will rotate the bit data of D register towards the right by N successive bits (in descending order, ie. in a 16-bit instruction, B15→B14, B14→B13, ... , B1→B0, B0→B15. In a 32-bit instruction, B31→B30, B30→B29, ... , B1→B0, B0→B31). At the same time, the status of the rotated out B0 bits will appear at the rotate-out bit "OTB".

- If the operand is 16 bit, the effective range of N is 1~16. For 32 bits (D instruction) operand, it is 1~32. Beyond this range, will set the N value error flag "ERR" to 1, and do not carry out this instruction.

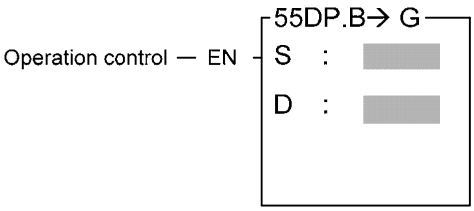


- The instruction at left rotates data from R0 register towards the right 8 successive bits. The results are shown below.



FUN55 D P B→G	BINARY-CODE TO GRAY-CODE CONVERSION	FUN55 D P B→G

Ladder symbol



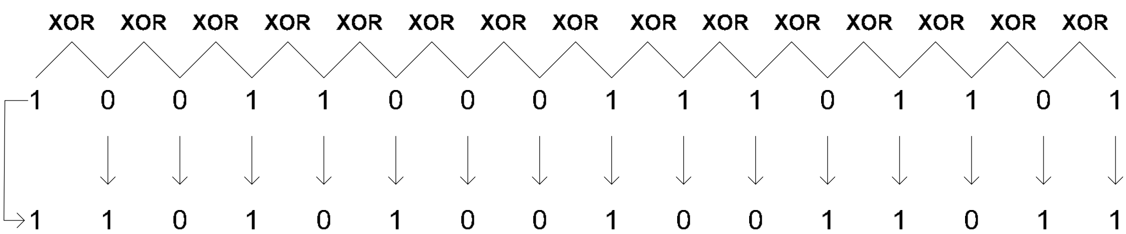
S ：Starting of source

D ：Starting address of destination

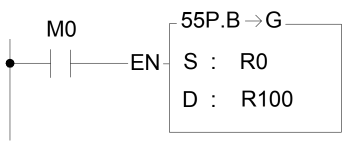
S，D operand can combine V、Z、P0~P9 for index addressing

Range / Ope-rand	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	16/32-bit +/- number	V、Z P0~P9
S	○	○	○	○	○	○	○	○	○	○	○	○		○
D		○	○	○	○	○	○		○	○*	○*	○		○

- When operation control "EN"=1 or changes from 0→1(P instruction), it will perform the code conversion; where S is the source (Binary code), and D is the destination (Gray code) for storing the result.
- The conversion method shown as below



Example 1: When M0 changes from 0→1, it will perform the 16-bit code conversion

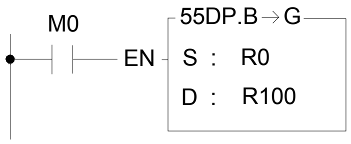


- Converting the 16-bit Binary-code in R0 into Gray-code, and then storing the result into R100.

R0＝1001010101010011B ➔ R100＝1101111111111010B

FUN55 D P B→G	BINARY-CODE TO GRAY-CODE CONVERSION	FUN55 D P B→G

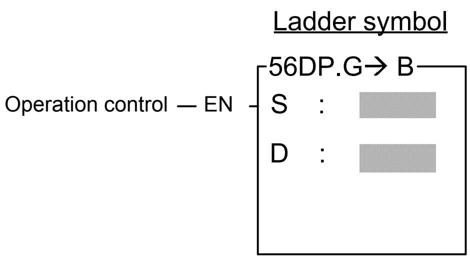
Example 2: When M0 =1, it will perform the 32-bit code conversion



- Converting the 32-bit Binary-code in DR0 into Gray-code, and then storing the result into DR100.

DR0＝00110111001001000010111100010100B ➔ DR100＝00101100101101100011100010011110B

FUN56 D P G→B	GRAY-CODE TO BINARY-CODE CONVERSION	FUN56 D P G→B



S ：Starting of source

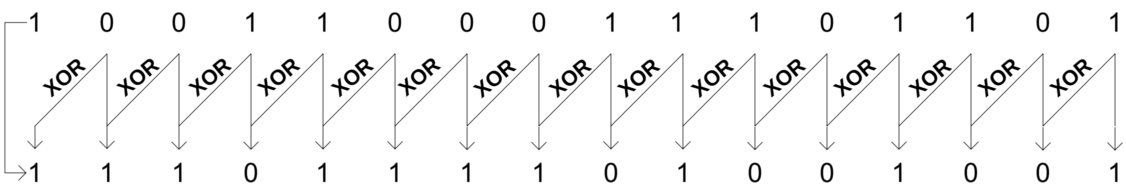
D ：Starting address of destination

S，D operand can combine V、Z、P0~P9 for index addressing

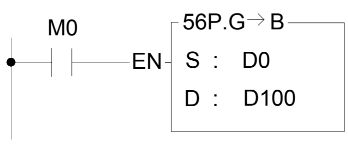
Range / Ope-rand	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR
	WX0 ∣ WX240	WY0 ∣ WY240	WM0 ∣ WM1896	WS0 ∣ WS984	T0 ∣ T255	C0 ∣ C255	R0 ∣ R3839	R3840 ∣ R3903	R3904 ∣ R3967	R3968 ∣ R4167	R5000 ∣ R8071	D0 ∣ D4095	16/32-bit +/- number	V、Z P0~P9
S	○	○	○	○	○	○	○	○	○	○	○	○		○
D		○	○	○	○	○	○		○	○*	○*	○		○

- When operation control "EN"=1 or changes from 0→1 (P instruction), it will perform the code conversion; where S is the source (Gray code), and D is the destination (Binary code) for storing the result.

- The conversion method shown as below :



Example 1: When M0 changes from 0→1, it will perform the 16-bit code conversion

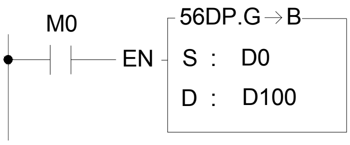


˙Converting the 16-bit Gray-code in D0 into Binary-code, and then storing the result into D100.

D0＝1001010101010011B ➔ D100＝1110011001100010B

FUN56 D P G→B	GRAY-CODE TO BINARY-CODE CONVERSION	FUN56 D P G→B

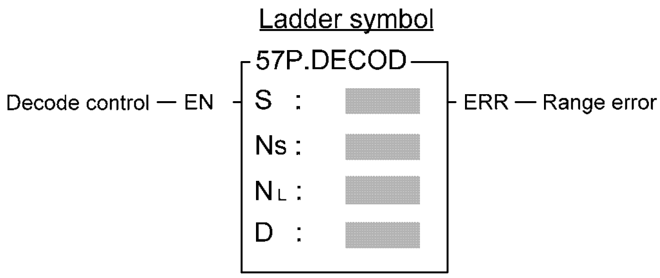
Example 2: When M0 =1, it will perform the 32-bit code conversion



˙Converting the 32-bit Gray-code in DD0 into Binary-code, and then storing the result into DD100.

DD0＝00110111001001000010111100010100B ➔ DD100＝00100101110001111100101000011000B

FUN 57 P DECOD	DECODE	FUN 57 P DECOD



S : Source data register to be decoded (16 bits)

N_S : Starting bits to be decoded within S

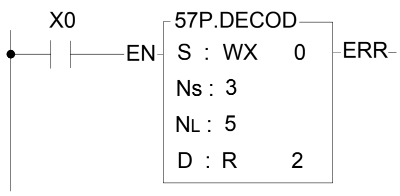
N_L : Length of decoded value (1~8 bits)

D : Starting register storing decoded results (2~256 points = 1~16 words)

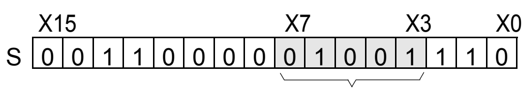
S, N_S, N_L, D may combine with V, Z, P0~P9 to serve indirect address application

Range / Operand	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	16/32-bit +/- number	V、Z P0~P9
S	○	○	○	○	○	○	○	○	○	○	○	○		○
N_S	○	○	○	○	○	○	○	○	○	○	○	○	0～15	○
N_L	○	○	○	○	○	○	○	○	○	○	○	○	2～256	○
D		○	○	○	○	○	○		○	○*	○*	○		○

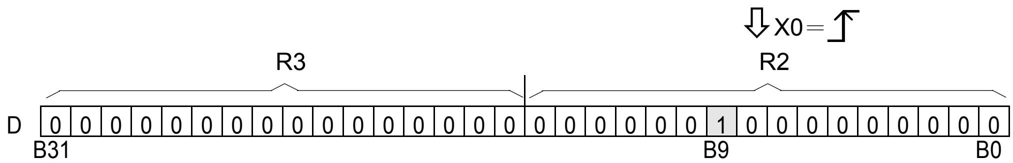
- This instruction, will set a single bit among the total of 2^{NL} discrete points (D) to 1 and the others bit are set to 0. The bit number to be set to 1 is specified by the value comprised by BN_S～BN_S+N_{L-1} of S（which is called the decode value, BN_S is the starting bit of the decode value, and BN_S+N_{L-1} is the end value）.
- When decode control "EN" = 1 or has a transition from 0 to 1 (P instruction), will take out the value BN_S～BN_S+N_{L-1} from S. And with this value to locate the bit position and set D accordingly, and set all the other bit to zero
- This instruction only provides 16 bit operand, which means S only has B0~B15. Therefore the effective range of Ns is 0~15, and the N_L length of the decode value is limited to 1~8 bits. Therefore the width of the decoded result D is $2^{1\sim8}$ points = 2~256 points = 1~16 words (if 16 points are not sufficient, 1 word is still occupied). If the value of N_S or N_L is beyond the above range, will set the range-error flag "ERR" to 1, and do not carry out this instruction.
- If the end bit value exceeds the B15 of S, then will extend toward B0 of S + 1. However if this occurs then S+1 can't exceed the range of specific type of operand (ie. If S is of D type register then S+1 can't be D3072). If violate this, then this instruction only takes out the bits from starting bit BNs to its highest limit as the decode value.



- The instruction at left takes out the data of five successive bits from X3 to X7 within the WX0 register and decodes it. The results are then stored in the 32-bit register starting at R2.

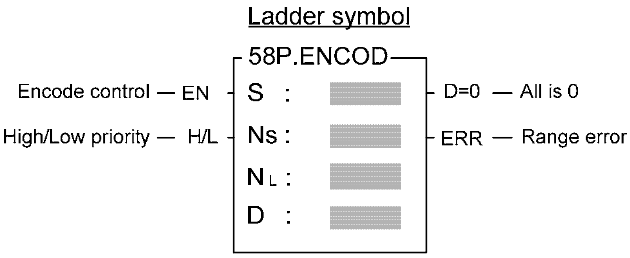


Length of decode value N_L=5,so bit value is formed by X7~X3 (equal 9)



Because N_L=5,the width of D is 2^5= 32 point = 2 word. That is, D is formed by R3R2, and the decoded value is 01001=9, therefore B9 (the 10th point) within D is set to 1, and all other points are 0.

FUN 58 P ENCOD	ENCODE	FUN 58 P ENCOD



S : Starting register to be encoded

N_S : Bit position within S as the encoding start point

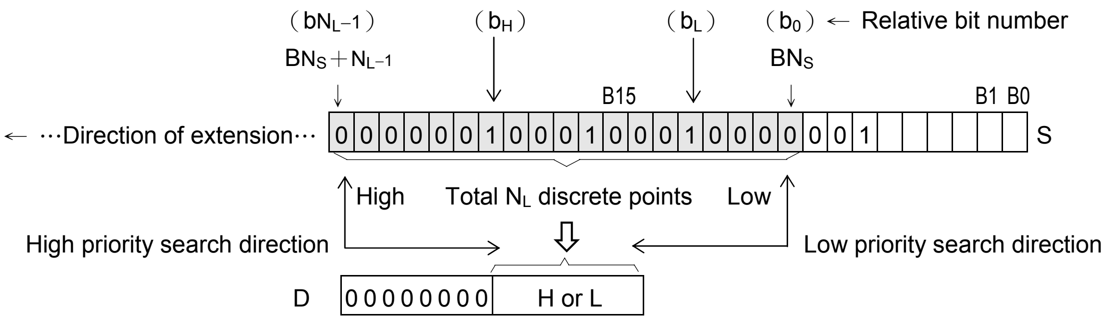
N_L : Number of encoding discrete points (2~256)

D : Number of register storing encoding results (1 word)

S, N_S, N_L, D may combine with V, Z, P0~P9 to serve indirect address application

Range / Operand	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-bit +/- number	V、Z P0~P9
S	○	○	○	○	○	○	○	○	○	○	○	○		○
N_S	○	○	○	○	○	○	○	○	○	○	○	○	0～15	○
N_L	○	○	○	○	○	○	○	○	○	○	○	○	2～256	○
D		○	○	○	○	○	○		○	○*	○*	○		○

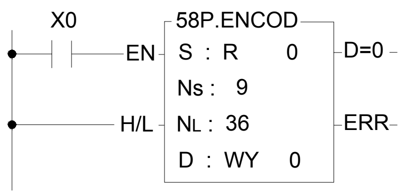
- When encode control "EN" = 1 or has a transition from 0 to 1 (P instruction), will starting from the points specified by Ns within S, take out towards the left (high position direction) N_L number of successive bits BN_S～ BN_S+N_{L-1} (BN_S is called the encoding start point, and its relative bit number is b0;BN_S+N_{L-1} is called the encoding end point, and its relative bit number is BN_{L-1}). From left to right do higher priority (when H/L=1) encoding or from right to left do lower priority (when H/L=0) encoding (i.e. seek the first bit with the value of 1, and the relative bit number of this point will be stored into the low byte (B0~B7) of encoded resultant register D, and the high byte of D will be filled with 0.



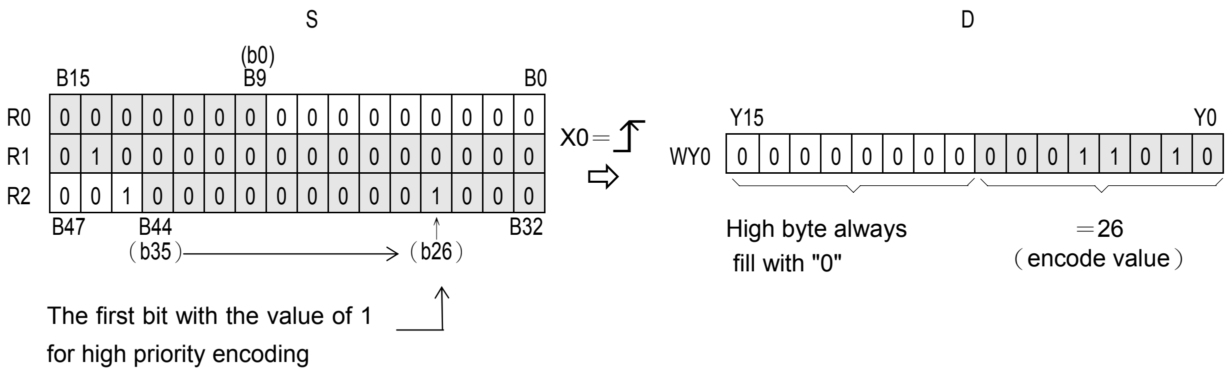
- As shown in the diagram above, for high priority encoding, the bit first to find is b_H (with a value of 12), and for low priority encoding, the bit first to find b_L (with a value of 4). Among the N_L discrete points there must be at least one bit with value of 1. If all bits are 0, will not to carry out this instruction, and the all zero flag "D=0" will set to 1.
- Because S is a 16-bit register, Ns can be 0~15, and is used to assign a point of B0~B15 within S as the encoding start point (b0). The value of N_L can be 2~256, and it is used to identify the encoding end point, i.e. it assigns N_L successive single points starting from the start point (b0) towards the left (high position direction) as the encoding zone (i.e. b0～bN_{L-1}). If the value of Ns or NL exceeds the above value, then do not carry out this instruction, and set the range-error flag "ERR" as 1.

FUN 58 P ENCOD	ENCODE	FUN 58 P ENCOD

- If the encoding end point (bN_{L-1}) beyond the B15 of S, then continue extending towards S+1, S+2, but it must not exceed the range of specific type of operand. If it goes beyond this, then this instruction can only take the discrete points between b0 and the highest limit into account for encoding.

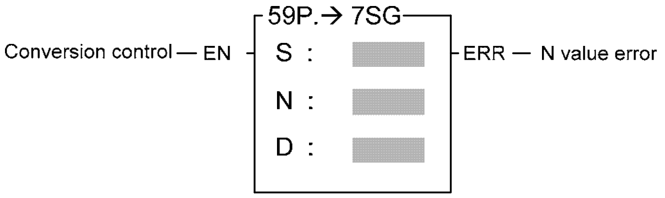


- The instruction at left is a high priority encode example. When X0 goes from 0 to 1, will take out toward left 36 successive bits starting from B9 (b0) specified by Ns within S, and perform high priority encoding (because H/L = 1). That is, starting from b35 (encoding end point), move right to find the first bit with the value of 1. The resultant value of this example is b26, so the value of D is $001A_H$=26, as shown in the diagram below.



FUN 59 P →7SG	7-SEGMENT CONVERSION	FUN 59 P →7SG

Ladder symbol



S : Source data to be converted

N : The nibble number within S for conversion

D : Register storing 7-segment result

S, N, D may combine with V, Z,P0~P9 to serve indirect address application

Range / Operand	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-bit +/- number	V、Z P0~P9
S	○	○	○	○	○	○	○	○	○	○	○	○	○	○
N	○	○	○	○	○	○	○	○	○	○	○	○	0～3	○
D		○	○	○	○	○	○		○	○*	○*	○		○

- When conversion control "EN" = 1 or has a transition from 0 to 1 (P instruction), will convert N+1 number of nibbles (A nibble is comprised by 4 successive bits, so B0~B3 of S form nibble 0, B4~B7 form nibble 1, etc...)within S to 7-segment code, and store the code into a low byte of D (High bytes does not change). The 7 segment within D are put in sequence, with "a" segment placed at B6, "b" segment at B5, ... ,"g" segment at B0. B7 is not used and is fixed as 0. For details please refer the "7-segment code and display pattern table".

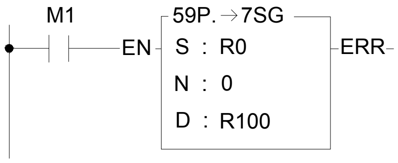
- Because this instruction is limited to 16 bits, and S only has 4 nibbles (NB0~NB3), the effective range of N is 0~3. Beyond this range, will set the N value flag error "ERR" to 1, and does not carry out this instruction.

- Care should be taken on total nibbles to be converted is N+1. N=0 means one digit to convert, N=1 means two digits to convert etc…

- When using the FATEK 7-segment expansion module(FBs-7SGxx) and the FUN84 (7SEG) handy instruction for mixing decoding and non-decoding application, FUN59 and FUN84 can be combined to simplify the program design.

FUN 59 P →7SG	7-SEGMENT CONVERSION	FUN 59 P →7SG

〈Example 1〉 When M1 OFF→ON, convert hexadecimal to 7-Segment

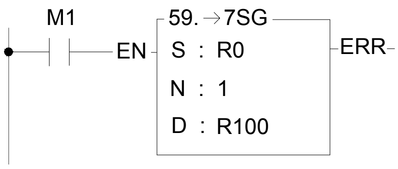


• Figure left shown the conversion of first digit(nibble) of R0 to 7-segment and store in low byte of R100, the high byte of R100 remain unchanged.

Original R100＝0000H

R0＝0001H ➔ R100＝0030H（1）

〈Example 2〉 When M1 ON, convert the hexadecimal to 7-Segment

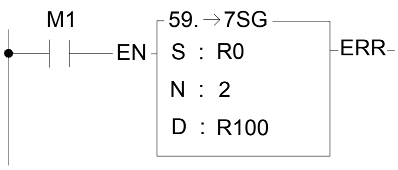


• Instruction at left will convert the first and the second digit of R0 to 7-segment and store in R100.
• The low byte of R100 stores first digit.
• The high byte of R100 stores second digit.

R0＝0056H ➔ R100＝5B5FH（56）

〈Example 3〉 When M1 ON, converting hexadecimal to 7-Segment

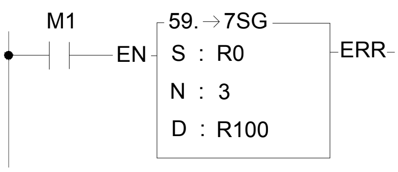
• Instruction at left will convert the first, second and third digit of R0 to 7-segment and store in R100 and R101.
• The low byte of R100 stores first digit.
• The high byte of R100 stores second digit.
• The low byte of R101 stores third digit.
• The high byte of R10 remain unchanged.



Original R101=0000H

R0=0A48H ➔ R100=337FH（48）
R101=0077H（A）

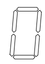
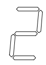
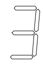
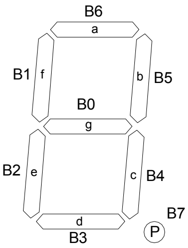
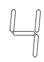
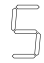
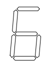
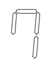
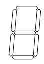
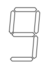
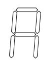
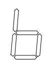
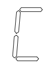
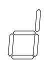
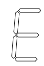
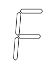
〈Example 4〉 When M1 ON, convert hexadecimal to 7-Segment



• Instruction at left will convert 1~4 digit of R0 to 7-segment and store in R100 and R101.
• The low byte of R100 stores first digit.
• The high byte of R100 stores second digit.
• The low byte of R101 stores third digit.
• The high byte of R10 stores 4^{th} digit.

R0=2790H ➔ R100=7B7EH（90）
R101=6D72H（27）

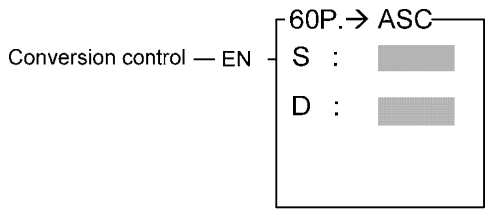
FUN 59 P →7SG	7-SEGMENT CONVERSION	FUN 59 P →7SG

Nibble data of S		7-segment display format	Low byte of D								Display pattern
Hexadecimal number	Binary number		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-segment display pattern table

FUN 60 P →ASC	ASCII CONVERSION	FUN 60 P →ASC

Ladder symbol

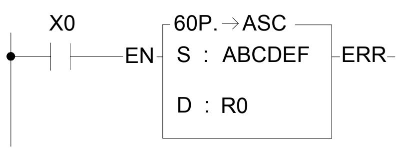


S : Alphanumerics to be converted into ASCII code

D : Starting register storing ASCII results

Range / Operand	WY	WM	WS	TMR	CTR	HR	OR	SR	ROR	DR	Alphanumeric
	WY0 | WY240	WM0 | WM1896	WS0 | WS984	T0 | T255	C0 | C255	R0 | R3839	R3904 | R3967	R3968 | R4167	R5000 | R8071	D0 | D4095	1～12 alphanumeric
S											○
D	○	○	○	○	○	○	○	○*	○*	○	

- When conversion control "EN" = 1 or has a transition from 0 to 1 (P instruction), will convert alphabets and numbers stored in S (S has a maximum of 12 alphanumeric character) into ASCII and store it into registers starting from D. Each 2 alphanumeric characters occupy one 16-bit register.
- The application of this instruction, most often, stores alphanumeric information within a program, and waits until certain conditions occur, then converts this alphanumeric information into ASCII and conveys it to external display devices which can accept ASCII code.



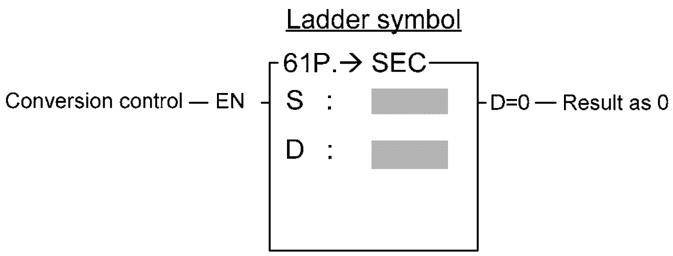
- The instruction at left converts the 6 alphabets -ABCDEF into ASCII then stores it into 3 successive registers starting from R0.

S: Alphabet ABCDEF X0=↑ ⇨ D:

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

FUN 61 P →SEC	HOUR:MINUTE:SECOND TO SECONDS CONVERSION	FUN 61 P →SEC

Ladder symbol

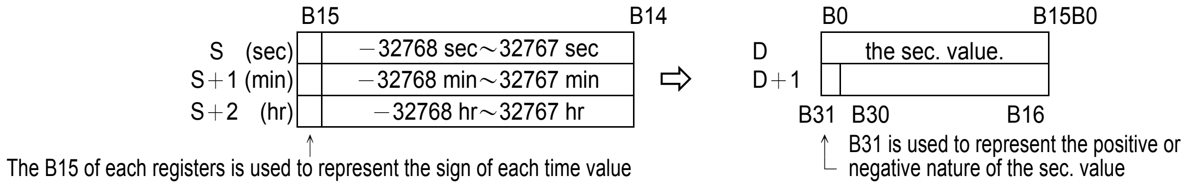


S : Starting calendar data register to be converted

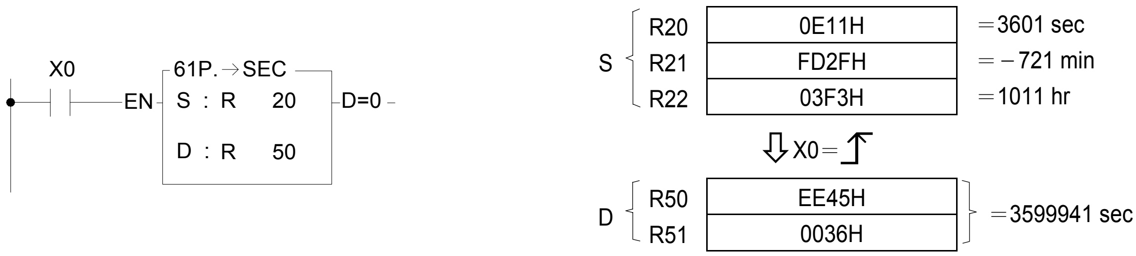
D : Starting register storing results

Range / Operand	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		○	○	○	○	○	○		○	○*	○*	○	

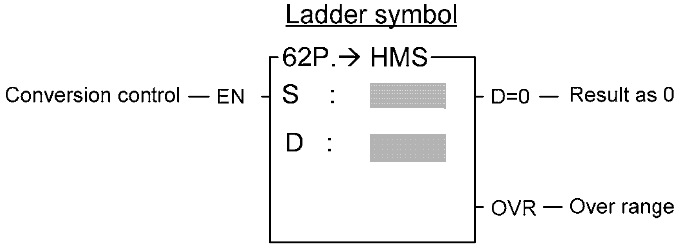
- When conversion control "EN" = 1 or has a transition from 0 to 1 (P instruction), will convert the hour: minute: second data of S~S+2 into an equivalent value in seconds and store it into the 32-bit register formed by combining D and D+1. If the result = 0, then set the "D = 0" flag as 1.
- Among the FBs-PLC instructions, the hour: minute: second time related instructions (FUN61 and 62) use 3 words of register to store the time data, as shown in the diagram below. The first word is the second register, the second word is the minute register, and finally the third word is the hour register, and in the 16 bits of each register, only B14~B0 are used to represent the time value. While bit B15 is used to express whether the time values are positive or negative. When B15 is 0, it represents a positive time value, and when B15 is 1 it represents a negative time value. The B14~B0 time value is represented in binary, and when the time value is negative, B14~B0 is represented with the 2's complement. The number of seconds that results from this operation is the result of summation of seconds from the three registers representing hours: minutes: seconds.



- Besides FUN61 or 62 instruction which treat hour: minute: second registers as an integral data, other instructions treat it as individual registers.
- The example program at below converts the hour: minute: second data formed by R20~R22 into their equivalent value in seconds then stored in the 32-bit register formed by R50~R51. The results are shown below.



FUN 62 P →HMS	SECOND→HOUR : MINUTE : SECOND	FUN 62 P →HMS

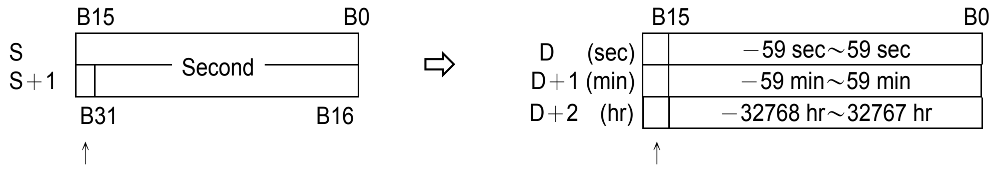


S :Starting register of second to be converted

D :Starting register storing result of conversion (hour : minute : second)

Range / Operand	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		○	○	○	○	○	○		○	○*	○*	○	

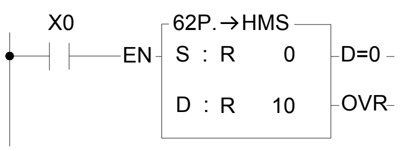
- When conversion control "EN" = 1 or has a transition from 0 to 1 (P instruction), will convert the second data from the S~S+1 32-bit register into the equivalent hour : minute : second time value and store it in the three successive registers D~D+2. All the data in this instruction is represented in binary (if there is a negative value it is represented using the 2's complement.)

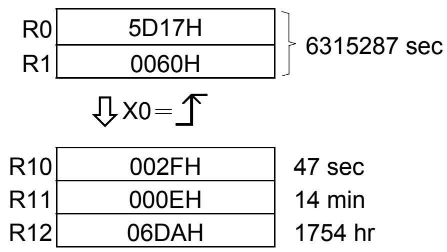


The bit B31 of the second register is used as the sign bit of the second value.

The bits B15 of each register are used as the sign bit of the hour : minute : second value.

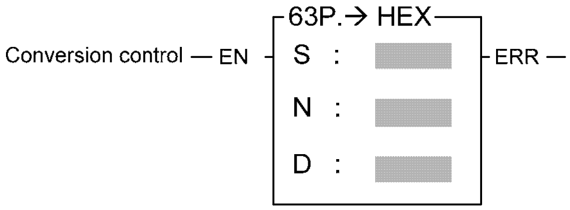
- As shown in the diagram above, after convert to hour : minute : second value, the minute : second value can only be in the range of -59 to 59, and the hour number can be in the range of -32768 to 32767 hours. Because of this, the maximum limit of D is -32768 hours, -59 minutes, -59 seconds to 32767 hours, 59 minutes, 59 seconds, the corresponding second value of S which is in the range of -117968399 to 117964799 seconds. If the S value exceeds this range, this instruction cannot be carried out, and will set the over range flag "OVR" to 1. If S = 0 then result is 0 flag "D = 0" will be set to 1.
- The program in the diagram below is an example of this instruction. Please note that the content of the registers are denoted by hexadecimal, and on the right is its equivalent value in decimal notation.





FUN 63 P →HEX	CONVERSION OF ASCII CODE TO HEXADECIMAL VALUE	FUN 63 P →HEX

Ladder symbol



S : Starting source register.

N : Number of ASCII codes to be converted to hexadecimal values.

D : The starting register that stores the result (hexadecimal value).

S, N, D, can associate with V, Z, P0~P9 to do the indirect addressing application.

Range / Operand	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-bit +number	V、Z P0~P9
S	○	○	○	○	○	○	○	○	○	○	○	○		○
N	○	○	○	○	○	○	○	○	○	○	○	○	1～511	○
D		○	○	○	○	○	○		○	○*	○*	○		○

- When conversion control "EN" =1 or changes from 0→1(P instruction), it will convert the N successive hexadecimal ASCII character('0'~'9','A'~'F') convey by 16 bit registers (Low Byte is effective) into hexadecimal value, and store the result into the register starting with D. Every 4 ASCII code is stored in one register. The nibbles of register, which does not involve in the conversion of ASCII code will remain unchanged.

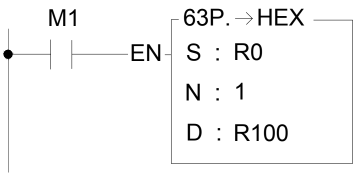
- The conversion will not be performed when N is 0 or greater than 511.

- When there is ASCII error (neither 30H～39H nor 41H～46H), the output "ERR" is ON.

- The main purpose of this instruction is to convert the hexadecimal ASCII character ('0'~'9','A'~'F'), which is received by communication port1 or communication port2 from the external ASCII peripherals, to the hexadecimal values that the CPU can process directly.

FUN 63 P →HEX	CONVERSION OF ASCII CODE TO HEXADECIMAL VALUE	FUN 63 P →HEX

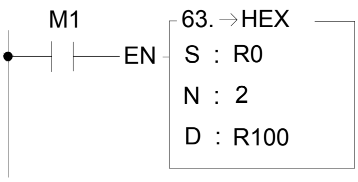
〈Example 1〉When M1 from OFF→ON, ASCII code converted to hexadecimal value.



- Converts the ASCII code of R0 into hexadecimal value and store to nibble0 (nibble1~nibble3 remain unchanged) of R100

Originally R100＝0000H
R0＝0039H（9）➔ R100＝0009H

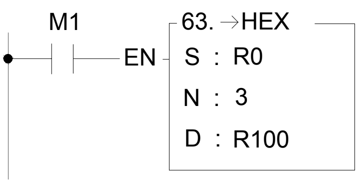
〈Example 2〉When M1 is ON, ASCII code converted to hexadecimal value.



- Converts the ASCII code of R0 and R1 into hexadecimal value and store to low byte (high byte remain unchanged) of R100

R0＝0039H（9） Originally R100＝0000H
R1＝0041H（A）➔ R100＝009AH

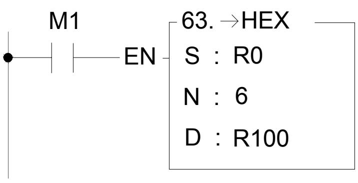
〈Example 3〉When M1 is ON, ASCII code converted to hexadecimal value.



- Converts the ASCII code of R0 and R1 into hexadecimal value and store result into R100 (nibble 3 remain unchanged)

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

〈Example 4〉When M1 is ON, ASCII code converted to hexadecimal value.

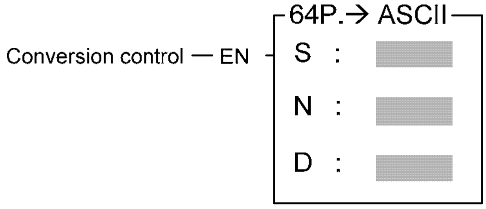


- Converts the ASCII code of R0~R5 into hexadecimal value and store it to R100~R101

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

FUN 64 P →ASCII	CONVERSION OF HEXADECIMAL VALUE TO ASCII CODE	FUN 64 P →ASCII

Ladder symbol



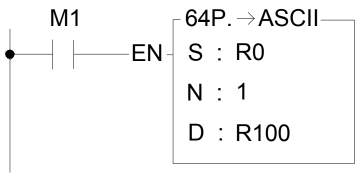
S : Starting source register
N : Number of hexadecimal digit to be converted to ASCII code.
D : The starting register storing result.
S, N, D, can associate with V, Z, P0~P9 to do the indirect addressing application.

Range / Operand	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-bit + number	V、Z P0~P9
S	○	○	○	○	○	○	○	○	○	○	○	○		○
N	○	○	○	○	○	○	○	○	○	○	○	○	1~511	○
D		○	○	○	○	○	○		○	○*	○*	○		○

- When conversion control "EN" =1 or changes from 0→1(P instruction), will convert the N successive nibbles of hexadecimal value in registers start from S into ASCII code, and store the result to low byte (high byte remain unchanged) of the registers which start from D.
- The conversion will not be performed when the value of N is 0 or greater than 511.
- The main purpose of this instruction is to convert the numerical value data, which PLC has processed, to ASCII code and transmit to ASCII peripherals by communication port1 or communication port 4.

FUN 64 P →ASCII	CONVERSION OF HEXADECIMAL VALUE TO ASCII CODE	FUN 64 P →ASCII

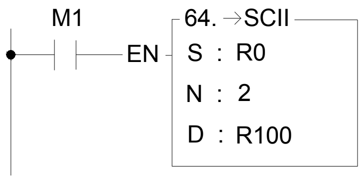
〈Example 1〉 When M1 changes from OFF→ON, it converts hexadecimal value to ASCII code.



• Converts the Nibble 0 of R0 to ASCII code and stores it into R100 (High byte does not change).

R0＝0009H ➔ R100＝0039H（9）

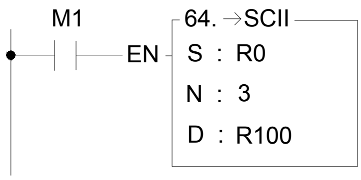
〈Example 2〉 When M1 is ON, it converts hexadecimal value to ASCII code.



• Converts the NB0～NB1 of R0 to ASCII code and stores it into R100 ～ R101 (high bytes remain unchanged).

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

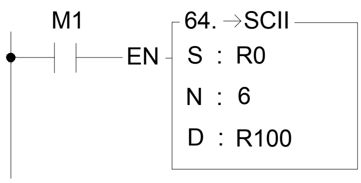
〈Example 3〉 When M1 is ON, it converts hexadecimal value to ASCII code.



• Converts the NB0～NB2 of R0 to ASCII code and stores it into R100～R102

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

〈Example 4〉 When M1 is ON, it converts hexadecimal value to ASCII code.

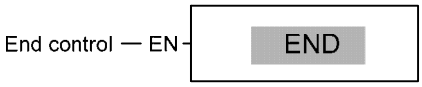


• Converts the NB0～NB5 of R0～R1 to ASCII code and stores it into R100～R105

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

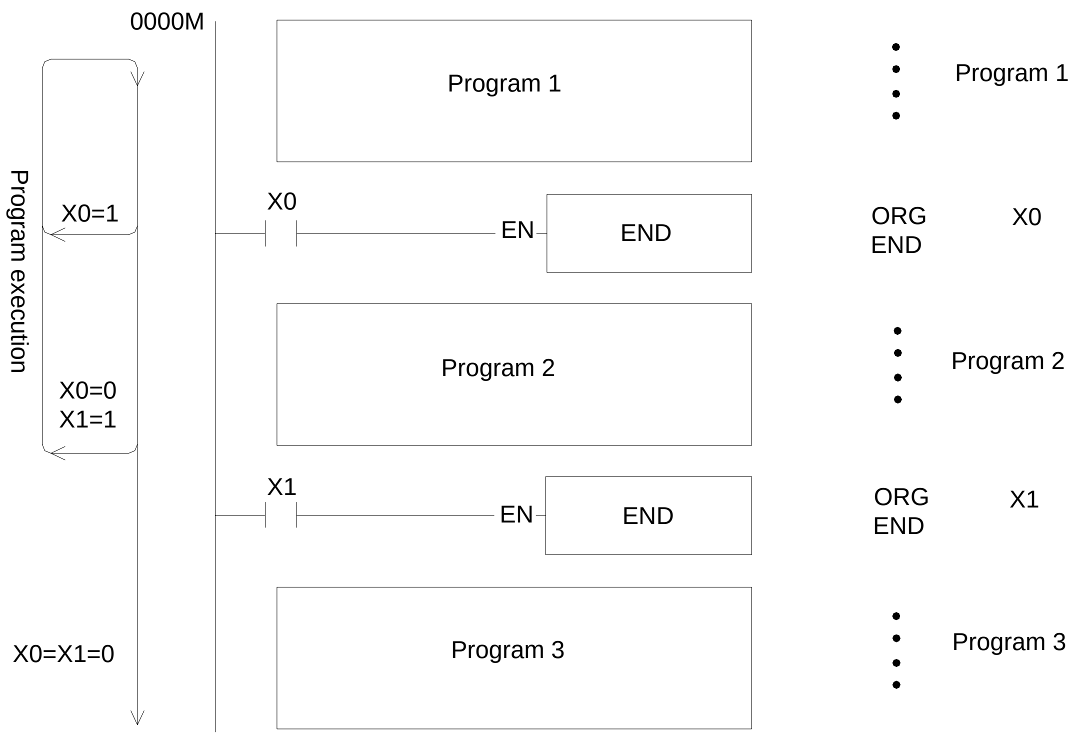
END	PROGRAM END	END

Ladder symbol



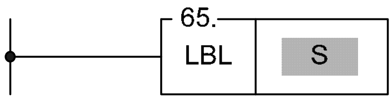
No operand

- When end control "EN" = 1, this instruction is activated. Upon executing the END instruction and "EN" = 1, the program flow will immediately returns to the starting point (0000M) to restart the next scan – i.e. all the programs after the END instruction will not be executed. When "EN" = 0, this instruction is ignored, and programs after the END instruction will continue to be executed as the END instruction is not exist.
- This instruction may be placed more than one point within a program, and its input (end control "EN") controls the end point of program execution. It is especially useful for debugging and for testing.
- It's not necessary to put any END instructions in the main program, CPU will automatic restart to start point when reach the end of main program.



FUN 65 LBL	LABEL	FUN 65 LBL

Ladder symbol



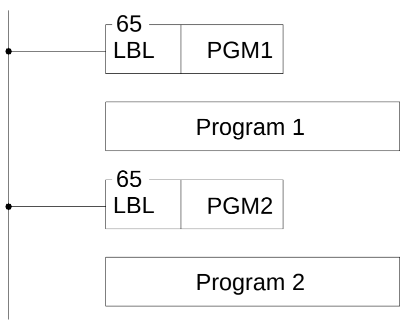
S : Alphanumeric, 1~6 characters

- This instruction is used to make a tag on certain address within a program, to provide a target address for execution of JUMP, CALL instruction and interrupt service. It also can be used for document purpose to improve the readability and interpretability of the program.
- This instruction serves only as the program address marking to provide the control of procedure flow or for remark. The instruction itself will not perform any actions; whether the program contains this instruction or not, the result of program execution will not be influenced by this instruction.
- The label name can be formed by any 1～6 alphanumeric characters and can't be duplicate in the same program. The following label names are reserved for interrupt function usage. These "reserved words", can't be used for normal program labels.

Reserved words	Description
X0+I～X15+I（INT0～INT15） X0–I～X15–I（INT0–～INT15–）	labels for external input (X0～X15) interrupt service routine.
HSC0I～HSC7I	labels for high speed counter HSC0～HSC7 interrupt service routine.
1MSI（1MS）、2MSI（2MS），3MSI（3MS），4MSI（4MS），5MSI（5MS），10MSI（10MS），50MSI（50MS），100MSI（100MS）	Labels for 8 kinds of internal timer interrupt service routine.
HSTAI（ATMRI），HST0I~HST3I	Label for High speed fixed timer interrupt service routine.
PSO0I～PSO3I	Labels for the pulse output command finished interrupt service routine.

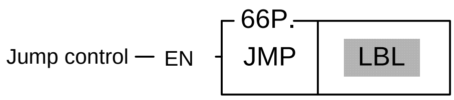
Only the interrupt service routine can use the label names listed on above table, if mistaken on using the reserved label on the normal subroutine can cause the CPU fail or unpredictable operation.

The label of following diagram illustration served only as program remarks (it is not treated as a label for call or jump target). For the application of labeling in jump control, please refer to JMP instruction for explanation. As to the labeling serves as subroutine names, please refer to CALL instruction for details.



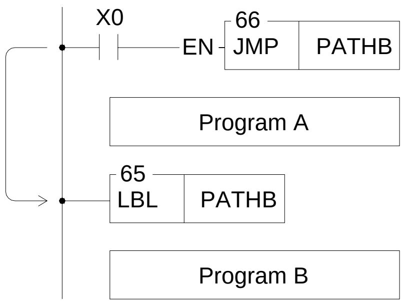
FUN 66 P JMP	JUMP	FUN 66 P JMP

Ladder symbol



LBL : The program label to be jumped

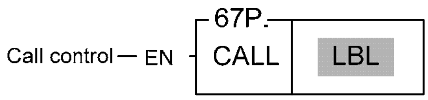
- When jump control “EN”=1 or changes from 0→1 (P instruction), PLC will jump to the location behind the marked label and continuous to execute the program.
- This instruction is especially suit for the applications where some part of the program will be executed only under certain condition. This can shorter the scan time while not executes the whole program.
- This instruction allows jump backward (i.e. the address of LBL is comes before the address of JMP instruction). However, care should be taken if the jump action cause the scan time exceed the limit set by the watchdog timer, the WDT interrupt will be occurred and stop executing.
- The jump instruction allows only for jumping among main program or jumping among subroutine area, it can’t jump across main/subroutine area.



• In the left diagram, when X0=1, the program will jump directly to the LBL position named PATHB and continuing to execute program B. Therefore it will skip the program A and none of the instructions of program A will be executed. The status of registers and the coils associated with program A will keep unchanged (as if there is no program section A).

FUN 67 P CALL	CALL	FUN 67 P CALL

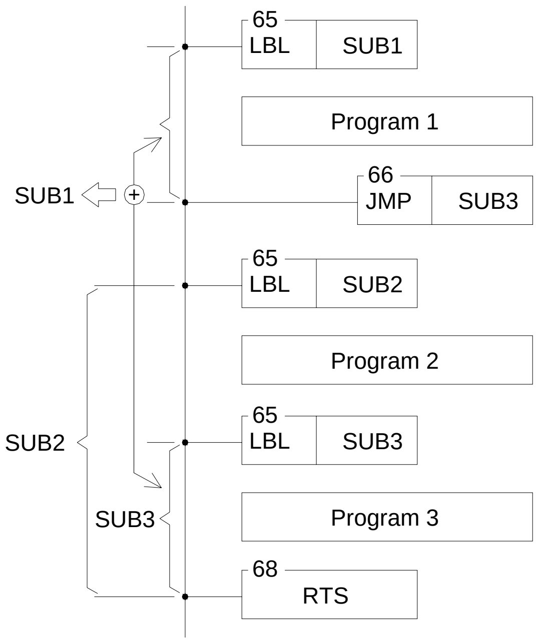
Ladder symbol



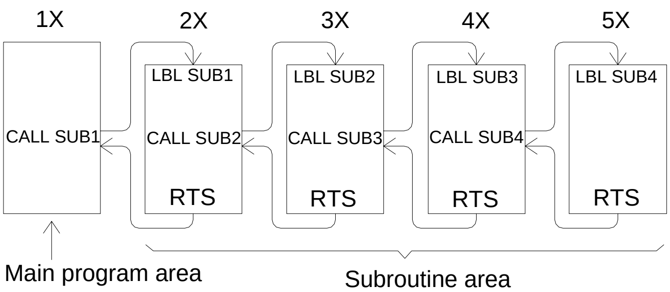
LBL : The subroutine label name to be called.

- When call control “EN”=1 or changes from 0→1 (P instruction), PLC will call (perform) the subroutine bear the same label name as the one being called. When execute the subroutine, the program will execute continuous as normal program does but when the program encounter the RTS instruction then the flow of the program will return back to the address immediately after the CALL instruction.

- All the subroutines must end with one “return from subroutine instruction RTS” instruction; otherwise it will cause executing error or CPU shut down. Nevertheless, an RTS instruction can be shared by subroutines (so called as multiple entering subroutines; even though the entry points are different, they have a same returning path) as illustrated in the right diagram subroutine SUB1～3.



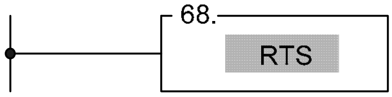
- When main program called a subroutine, the subroutine also can call the other subroutines (so called the nested subroutines) for up to 5 levels at the most (include the interrupt routine).



- Interrupt service programs （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） are also a kind of subroutine. It is also placed in sub program area. However, the calling of interrupt service program is triggered off by the signaling of hardware to make the CPU perform the corresponding interrupt service program (which we called as the calling of the interrupt service program). The interrupt service program can also call subroutine or interrupted by other interrupts with higher priority. Since it is also a subroutine (which occupied one level), it can only call or interrupted by 4 levels of subroutine or interrupt service program. Please refer to RTI instruction for explanation.

FUN 68 RTS	RETURN FROM SUBROUTINE	FUN 68 RTS

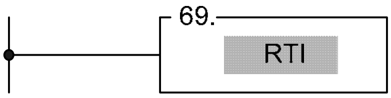
Ladder symbol



- This instruction is used to represent the end of a subroutine. Therefore it can only appear within the subroutine area. Its input side has no control signal, so there is no way to serially connect any contacts. This instruction is self sustain, and is directly connected to the power line.
- When PLC encounter this instruction, it means that the execution of a subroutine is finished. Therefore it will return to the address immediately after the CALL instruction, which were previously executed and will continue to execute the program.
- If this instruction encounters any of the three flow control instructions MC, SKP, or JMP, then this instruction may not be executed (it will be regarded as not exist). If the above instructions are used in the subroutine and causing the subroutine not to execute the RTS instruction, then PLC will halt the operation and set the M1933(flow error flag) to 1. Therefore, no matter what the flow is going, it must always ensure that any subroutine must be able to execute a matched RTS instruction.
- For the usage of the RTS instruction please refer to instructions for the CALL instruction.

FUN 69 RTI	RETURN FROM INTERRUPT	FUN 69 RTI

Ladder symbol



- The function of this instruction is similar to RTS. Nevertheless, RTS is used to end the execution of sub program, and RTI is used to end the execution of interrupt service program. Please refer to the explanation of RTS instruction.

- A RTI instruction can be shared by more than one interrupt service program. The usage is the same as the sharing of an RTS by many subroutines. Please refer to the explanation of CALL instruction.

- The difference between interrupts and call is that the sub program name (LBL) of a call is defined by user, and the label name and its call instruction are included in the main program or other sub program. Therefore, when PLC performs the CALL instruction and the input “EN”=1 or changes from 0→1 (P instruction), the PLC will call (execute) this sub program. For the execution of interrupt service program, it is directly used with hardware signals to interrupt CPU to pause the other less important works, and then to perform the interrupt service program corresponding to the hardware signal (we call it the calling of interrupt service program). In comparing to the call instruction that need to be scanned to execute, the interrupt is a more real time in response to the event of the outside world. In addition, the interrupt service program cannot be called by label name; therefore we preserve the special “reserved words” label name to correspond to the various interrupts offered by PLC (check FUN65 explanation for details). For example, the reserved word X0+I is assigned to the interrupt occurred at input point X0; as long as the sub program contains the label of X0+I, when input point X0 interrupt is occurred (X0: ↑), the PLC will pause the other lower priority program and jump to the subroutine address which labeled as X0+I to execute the program immediately.

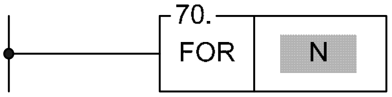
- If there is a interrupt occurred while CPU is handling the higher priority (such as hardware high speed counter interrupt) or same priority interrupt program (please refer to Chapter 10 for priority levels), the PLC will not execute the interrupt program for this interrupt until all the higher priority programs were finished.

- If the RTI instruction cannot be reached and performed in the interrupt service routine, may cause a serious CPU shut down. Consequently, no matter how you control the flow of program, it must be assured that the RTI instruction will be executed in any interrupt service program.

- For the detailed explanation and example for the usage of interrupts, please refer to Chapter 9 for explanation.

FUN 70 FOR	FOR	FUN 70 FOR

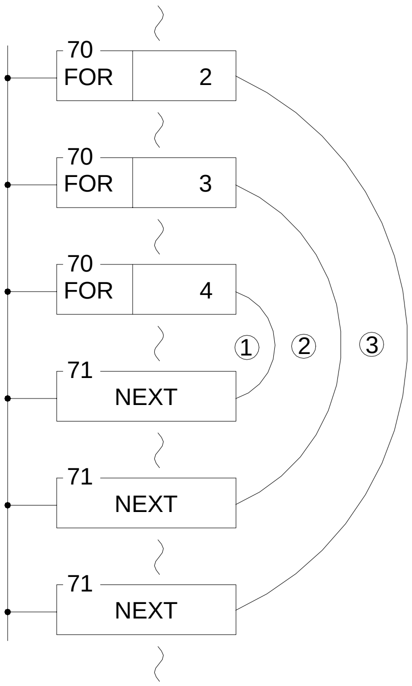
Ladder symbol



N : Number of times of loop execution

Range / Operand	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	1 | 16383
N	○	○	○	○	○	○	○	○	○	○	○	○	○

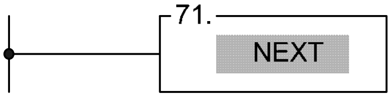
- This instruction has no input control, is connected directly to the power line, and cannot be in series with any conditions.
- The programs within the FOR and NEXT instructions form a program loop (the start of the loop program is the next instruction after FOR, and the last is the instruction before NEXT). When PLC executes the FOR instruction, it first records the N value after that instruction (loop execution number), then for N times successively execution from start to last of the programs in the loop. Then it jumps out of the loop, and continues executes the instruction immediately after the NEXT instruction.
- The loop can have a nested structure, i.e. the loop includes other loops, like an onion. 1 loop is called a level, and there can be a maximum of 5 levels. The FOR and NEXT instructions must be used in pairs. The first FOR instruction and the last NEXT instruction are the outermost (first) level of a nested loop. The second FOR instruction and the second last NEXT instruction are the second level, the last FOR instruction and the first NEXT instruction form the loop's innermost level.



- In the example in the diagram at left, loop ① will be executed 4 ×3 ×2 = 24 times, loop ② will be executed 3 ×2 = 6 times, and loop ③ will be executed 2 times.
- If there is a FOR instruction and no corresponding NEXT instruction, or the FOR and NEXT instructions in the nested loop have not been used in pairs, or the sequence of FOR and NEXT has been misplaced, then a syntax error will be generated and this program may not be executed.
- In the loop, the JMP instruction may be used to jump out of the loop. However, care must be taken that once the loop has been entered (and executed to the FOR instruction), no matter how the program flow jumps, it must be able to reach the NEXT instruction before reaching the END instruction or the bottom of the program. Otherwise FBs-PLC will halt the operation and show an error message.
- The effective range of N is 1~16383 times. Beyond this range FBs-PLC will treat it as 1. Care should be taken , if the amount of N is too large and the loop program is too big, a WDT may occur.

FUN 71 NEXT	LOOP END	FUN 71 NEXT

Ladder symbol



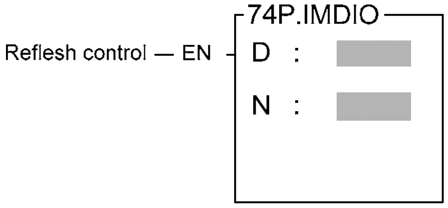
- This instruction and the FOR instruction together form a program loop. The instruction itself has no input control, is connected directly to the power line, and cannot be in series with any conditions.

- When PLC has not yet entered the loop (has not yet executed to the FOR instruction, or has executed but then jumped out), but the NEXT instruction is reached, then PLC will not take any action, just as if this instruction did not exist.

- For the usage of this instruction please refer to the explanations for the FOR instruction on the preceding page.

FUN 74 P IMDIO	IMMEDIATE I/O	FUN 74 P IMDIO

Ladder symbol



D : Starting number of I/O points to be refreshed
N : Number of I/O points to be refreshed

Range / Operand	X	Y	K
	Xn of Main Unit.	Yn of Main Unit.	1 ∣ 36
D	○	○	
N			○

- For normal PLC scan cycle, the CPU gets the entire input signals before the program is executed, and then perform the executing of program based on the fresh input signals. After finished the program execution the CPU will update all the output signals according to the result of program execution. Only after the complete scan has been finished will all the output results be transferred all at once to the output. Thus for the input event to output responses, there will be a delay of at least 1 scan time (maximum of 2 scan time). With this instruction, the input signals or output signals specified by this instruction can be immediately refresh to get the faster input to output response without the limitation imposed by the scan method.
- When refresh control "EN" = 1 or has a transition from 1 to 0(P instruction), then the status of N input points or output points (D~D+N-1) will be refreshed.
- The I/O points for FBs-PLC's immediate I/O are only limited to I/O points on the main unit. The table below shows permissible I/O numbers for 20, 32, 40 and 60 point main units:

Main-unit type / Permissible numbers	10 points	14 points	20 points	24 points	32 points	40 points	60 points
Input signals	X0～X5	X0～X7	X0～X11	X0～X13	X0～X19	X0～X23	X0～X35
Output signals	Y0～Y3	Y0～Y5	Y0～Y7	Y0～Y9	Y0～Y11	Y0～Y15	Y0～Y23

- If the intended refresh I/O signals of this instruction is beyond the range of I/O points specified on above table then PLC will be unable to operate and the M1931 error flag will be set to 1. (for example, if in a program, D=X11, N=10, which means X11 to X20 are to be immediately retrieved. Supposing the main unit is FBs-32MA, then its biggest input point is X19, and clearly X20 has already exceeded the main unit's input point number so under such case M1931 error flag will be set to 1).
- With this instruction, PLC can immediately refresh input/output signals. However, the delay of the hardware or the software filter impose on the I/O signals still exist. Please pay attention on this.

FUN 76 D TKEY	DECIMAL- KEY INPUT	FUN 76 D TKEY

Ladder symbol

IN : Key input point

D : register storing key-in numerals

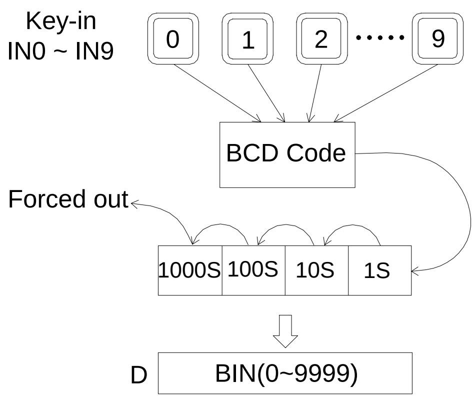
KL: starting coil to reflect the input status

D may combine with V, Z, P0~P9 to serve indirect address application

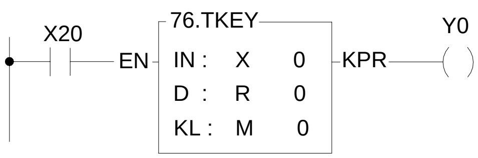
Range / Ope-rand	X	Y	M	S	WY	WM	WS	TMR	CTR	HR	OR	SR	ROR	DR	XR
	X0 | X240	Y0 | Y240	M0 | M1896	S0 | S984	WY0 | WY240	WM0 | WM1896	WS0 | WS984	T0 | T255	C0 | C255	R0 | R3839	R3904 | R3967	R3968 | R4167	R5000 | R8071	D0 | D4095	V、Z P0~P9
IN	○														
D					○	○	○	○	○	○	○	○*	○*	○	○
KL		○	○	○											

- This instruction has designated 10 input points IN~IN+9 (IN0~IN9) to one decimal number entry (IN->0, IN+1->1…). According to the key-in sequence (ON) of these input points, it is possible to enter 4 or 8 decimal numbers into the registers specified by D.

- When input control "EN" = 1, this instruction will monitor the 10 input points starting from IN and put the corresponding number into D register while the key were depressed. It will wait until the input point has released, then monitor the next "ON" input point, and shift in the new number into D register (high digit is older than low digit) . For the 16-bit operand, D register can store up to 4 digits, and for the 32-bit operand 8 digits may be stored. When the key numbers full fill the D register, new key-in number will kick out the oldest key number of the D register. The key-in status of the 10 input points starting from IN will be recorded on the 10 corresponding coil starting from KL. These coils will set to 1 while the corresponding key is depressed and remain unchanged even if the corresponding key is released. Until other key is depressed then it will return to zero. As long as any input point is depressed (ON), then the key-in flag KPR will set to 1. Only one of IN0~IN9 key can be depressed at the same time. If more than one is pressed, then the first one is the only one taken. Below is a schematic diagram of the function with 16-bit operand.



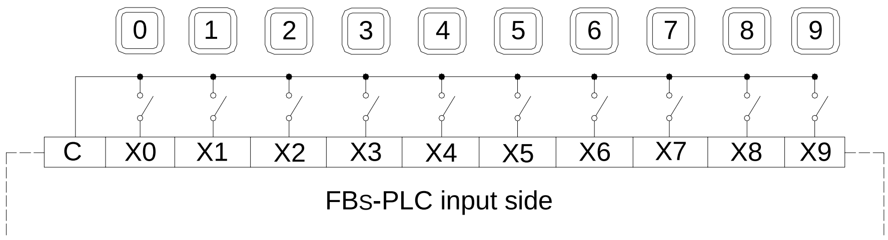
- When input control "EN" = 0, this instruction will not be executed. KPR output and KL coil status will be 0. However, the numerical values of D register will remain unchanged.



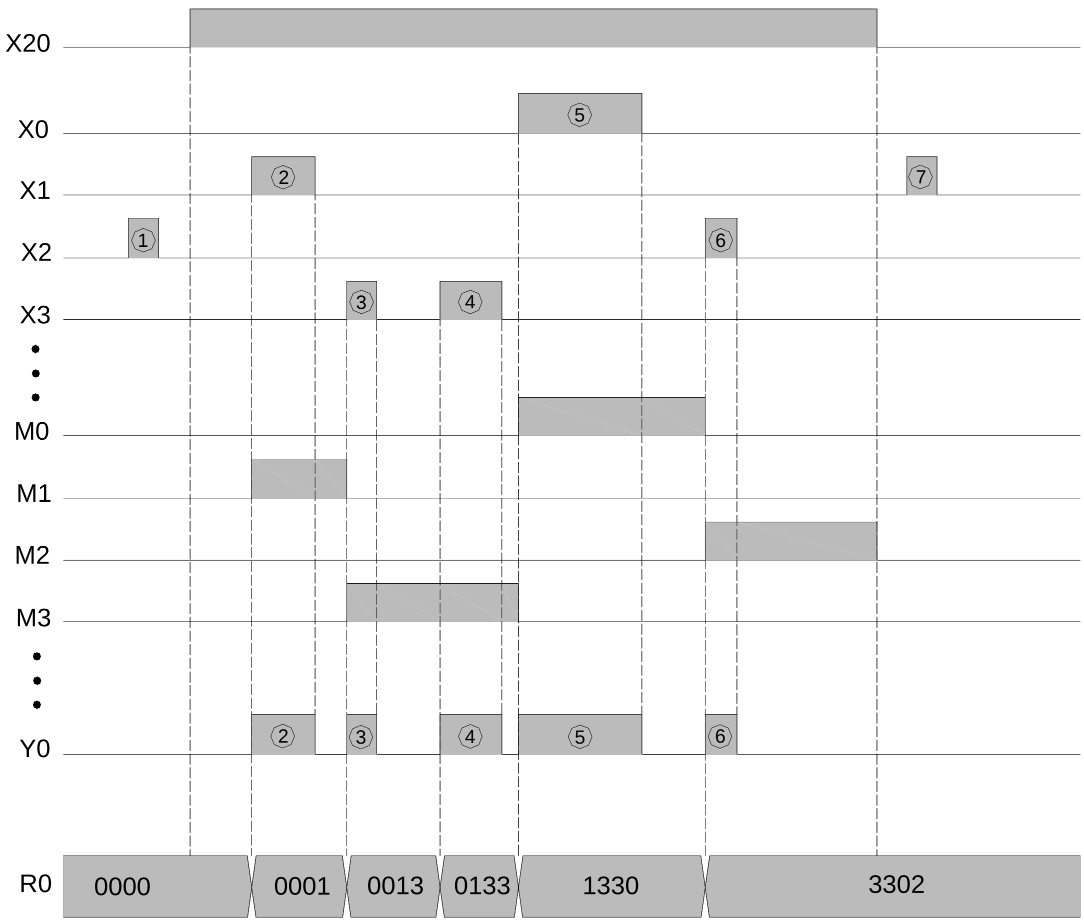
- The instruction at left represents the input point X0 with the number "0", X1 is represented by 1, ... , M0 records the action of X0, M1 records the action of X1 ... , and the input numerical values are stored in the R0 register.

FUN 76 D TKEY	DECIMAL- KEY INPUT	FUN 76 D TKEY

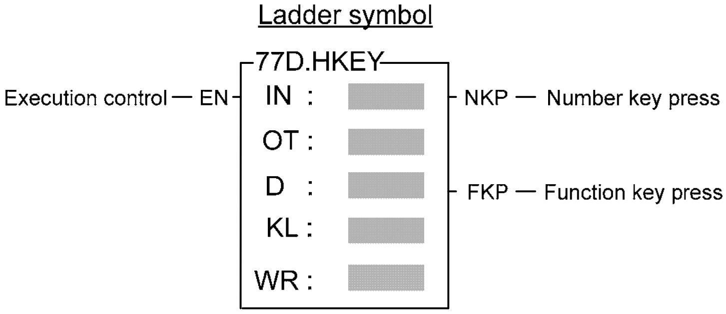
The following diagram is the input wiring schematic for this example:



- If the X0~X3 key-in sequence follow the ①②③④⑤⑥⑦ sequence in the following diagram. At step ① and ⑦ the X20 is 0, so there was no key generated, only steps ②③④⑤⑥ are effective. Because the register can only hold 4 key numbers, Of these 5 steps the first key was kick out. The key strokes 3302 of the steps ③④⑤⑥ are entered in the R0 register.



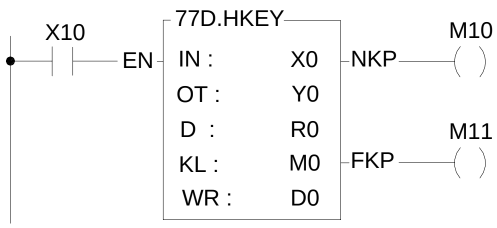
FUN 77 D HKEY	HEX-KEY INPUT	FUN 77 D HKEY

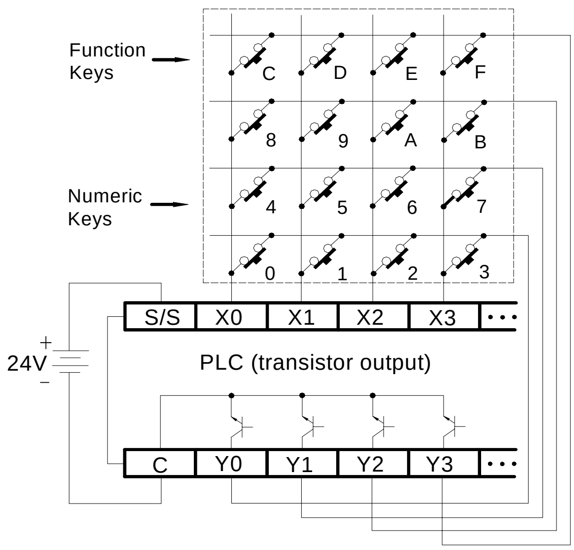


IN : Starting of digital input for key scan
OT: Starting of digital output for multiplexing key scan (4 points)
D : Register to store key-in numbers
KL : Starting relay for key status
WR: Working register, it can't repeat in use
D may combine with V、Z、P0~P9 to serve indirect addressing application

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

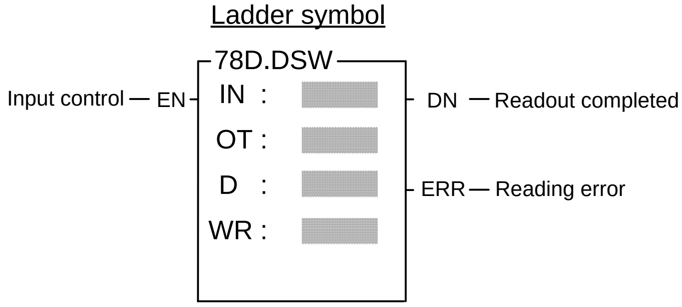
- The numeric (0~9) key function of this instruction is similar as for the TKEY instruction. The hardware connection for TKEY and HKEY is different. For TKEY instruction each key have one input point to connect, while HKEY use 4 input points and 4 output points to form a 4x4 multiplex 16 key input. 4x4 means that there can be 16 input keys, so in addition to the 10 numeric keys, the other 6 keys can be used as function keys (just like the usual discrete input). The actions of the numeric keys and the function keys are independent and have no effect on each other.
- When execution control "EN" = 1, this instruction will scan the numeric keys and function keys in the matrix formed by the 4 input points starting from IN and the 4 output points starting from OT. For the function of the numeric keys and "NKP" output please refer to the TKEY instruction. The function keys maintain the key-in status of the A~F keys in the last 6 relays specified by KL (the first 10 store the key-in status of the numeric keys). If any one of the A~F keys is depressed, FKP (FO1) will set to 1. The OT output points for this instruction must be transistor outputs.
- The biggest number for a 16-bit operand is 4 digits (9999), and for 32-bit operand is 8 digits (99999999). However, there are only 6 function keys (A~F), no matter whether it is a 16-bit or 32-bit operand.





- The instruction in the diagram above uses X0~X3 and Y0~Y3 to form a multiplex key input. It can input numeric values of 8 digits and stores the results in R1R0. The input status of the function keys is stored in M10(A)~M15(F).

FUN 78 D DSW	DIGITAL SWITCH INPUT	FUN 78 D DSW



IN : Starting of input for thumb wheel switch

OT : Starting of output for multiplexing scan (4 points)

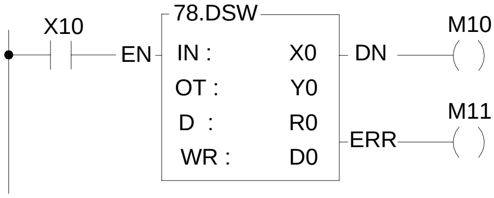
D : Register to store readout value

WR: Working register, it can't repeat in use (WR & WR+1 for 16-bit operation; WR, WR+1 & WR+2 for 32-bit operation)

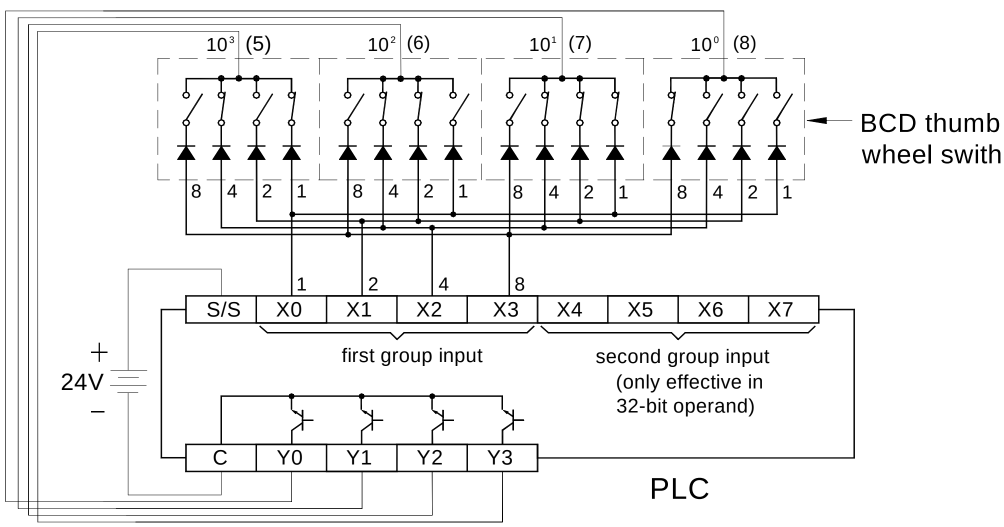
D may combine with V、Z、P0~P9 to serve indirect addressing application

Range / Operand	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			○	○	○	○	○	○	○	○*	○*	○	○

- When input control "EN" = 1, this instruction will readout one digit data from the 4 input points starting from IN (IN0~IN3). It takes 4 scans to read out a group of 4-digit BCD values (0000~9999) and store them into D register. With a 32-bit operand, each scan can get 2 digits of data by reading the additional digit from IN4-IN7 and store it in the D+1 register. Each bit of OT0~OT3 will sequentially set to 1 and get the digit data respectively into 10^0(ones), 10^1(tens), 10^2(hundreds), and 10^3(thousands). As long as EN is 1, PLC will scan and read out in continuous cycles. When each complete cycle is finished (i.e. the 4 digit readout of 10^0~10^3 is completed), the readout completed flag "DN" is set to 1. However, it is only kept for one scan. If any digital readout value is not within the range of 0~9 (BCD), then reading error "ERR" will be set to 1 and the value of that group of digits will be set to 0000.
- The output points must be transistor outputs.



- In this example, when X10 is 1, then the numeric value of the thumb wheel switch (5678 in this example) will be read out and stored into the R0 register.
- The bits (8,4,2,1) with same digit should be connect together and series with a diode (as shown in diagram below).
- With 32-bit operand a set of similar thumb wheel switch may be added to X4~X7 (Y0~Y3 are shared with another group).



FUN 79 D 7SGDL	7-SEGMENT OUTPUT WITH LATCH	FUN 79 D 7SGDL

Ladder symbol

S : Register storing the data (BCD) to be displayed

OT : Starting number of scanning output

N : Specify signal output and polarity of latch

WR : Working register, it can't repeat in use

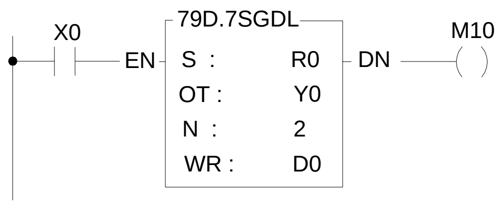
S may combine with V、Z、P0~P9 to serve indirect addressing application

Range / Operand	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-bit number	V、Z P0~P9
S			○	○	○	○	○	○	○	○	○	○	○	○	○
OT	○														
N														0～3	

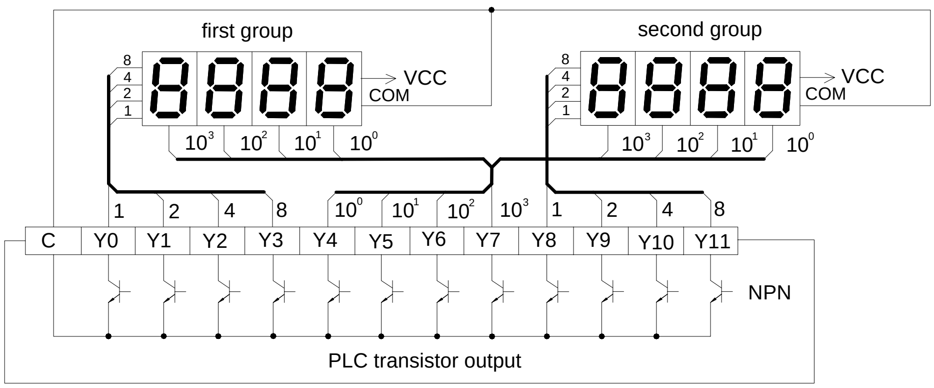
- When input control "EN" = 1, the 4 nibbles of the S register, from digit 0 to digit 3, are sequentially sent out to the 4 output points, OT0~OT3. While output the digit data, the latch signal of that digit (OT4 corresponds to digit 0, OT5 corresponds to digit 1, etc...) at the same time is also sent out so that the digital value will be loaded and latched into the 7-segment display respectively.

- When in D (32-bit) instruction, nibbles 0~3 from the S register, and nibbles 0~3 from the S+1 register are transferred separately to OT0~OT3 and OT8~OT11. Because they are transferred at the same time, they can use the same latch signal. 16-bit instructions do not use OT8~OT11.

- As long as "EN" remains 1, PLC will execute the transfer cyclically. After each transfer of a complete group of numerical values (nibbles 0~3 or 0~7), the output completed flag "DN" will set to 1. However, it will only be kept for 1 scan.



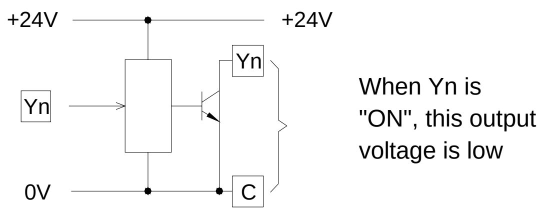
- In this example, when X0=1, the 4 nibbles of R0 will be transferred to the first group 7-segment display in the diagram below. The 4 nibbles of R1 will be transferred to the second group 7-segment display.



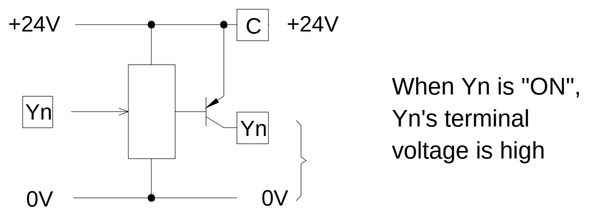
FUN 79 D 7SGDL	7-SEGMENT OUTPUT WITH LATCH	FUN 79 D 7SGDL

- FATEK PLC's transistor output has both a negative logic transistor output (NPN transistor - when the output status is ON, the terminal voltage of the transistor output is low), and a positive logic transistor output (PNP - when the output status is ON, the terminal voltage of the transistor output is high). Their structure is as follows:

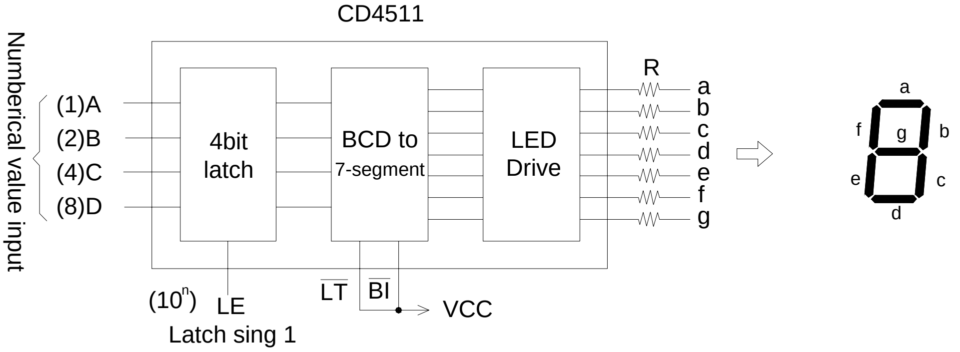
FBs-PLC negative logic output (NPN transistor)



FBs-PLC positive logic output (PNP transistor)



- The data inputs (8,4,2,1) and latch signals of the 7-segment displays on the shelf for positive and negative logic are all available. For example, for numerical value "8", the positive logic input should be 1000, and the negative logic input 0111. Similarly, when the latch signal is 0, the positive logic latch permits the display numerical values to enter through the latch (i.e. be loaded). When the latch signal is 1, the numerical values in the latch are latched (maintained), and with negative logic they are not. The following diagram of a CD-4511 7-segment display IC is an example of a positive logic numerical value input with latch.



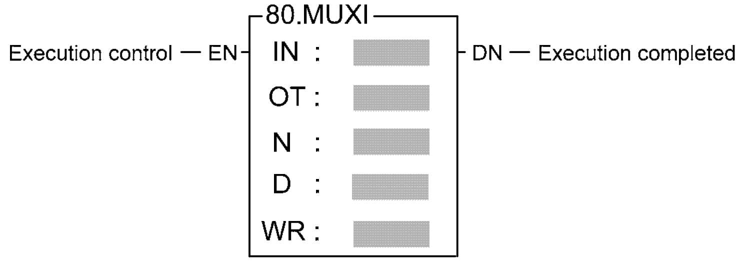
- Because the PLC output and the 7-segment display input polarity can be positive and negative logic. Therefore, the polarities between output and input must be coordinated to get the correct result. This instruction uses N to specify the polarity relation between the PLC transistor output, and the 7-segment display. The table below shows all the possibility.

Numerical value input (8~1)	Latch signal (10^0-10^3)	Value of N
Same	Same	0
	Different	1
Different	Same	2
	Different	3

- In the diagram above, CD4511 is used as an example. If use NPN output, the data input polarity is different to PLC, and its latch input polarity is the same as PLC, so N value should chosen as 2.

FUN 80 MUXI	MULTIPLEX INPUT	FUN 80 MUXI

Ladder symbol



IN : Multiplex input point number

OT: Multiplex output point number (must be transistor output point)

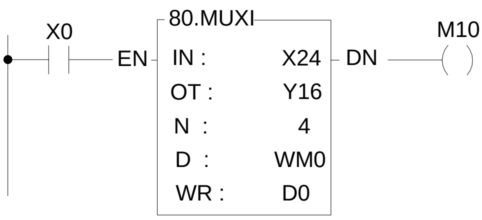
N : Multiplex input lines (2~8)

D : Register for storing results

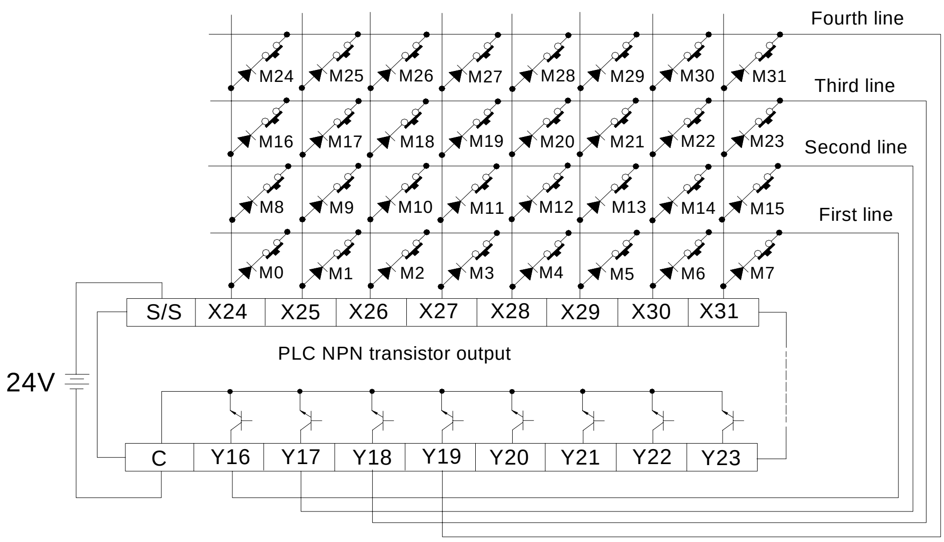
D may combine with V, Z, P0~P9 to serve indirect address application

Range / Operand	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~P0
IN	○													
OT		○												
N													○	
D			○	○	○	○	○	○	○	○*	○*	○		○

- This instruction uses the multiplex method to read out N lines of input status from 8 consecutive input points (IN0~IN7) starting from the input point specified by IN. With this method we can obtain 8xN input status, but only need to use 8 input points and N output points.
- The multiplex scanning method goes through N output points starting from the OT output point. Each scan one of the N bits will set to 1 and the corresponding line will be selected. OT0 responsible for first line, while OT1 responsible for second line, etc. Until it read all the N lines the 8xN status that has been read out is then stored into the register starting at D, and the execution completed flag "DN" is set as 1 (but is only kept for one scanning period).
- With every scan, this instruction retrieves a line for 8 input status, so N lines require N scan cycles before they can be completed.

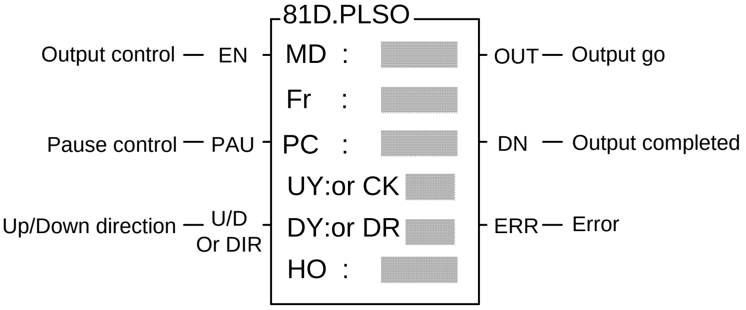


- This example retrieves 4 linesx8 points of input, 32 point status in all. They are stored into the 32-bit register of DWM0 (M0~M31).



FUN 81 D PLSO	PULSE OUTPUT	FUN 81 D PLSO

Ladder symbol



MD : Output mode selection
Fr : Pulse frequency
PC : Output pulse count
UY : Up pulse output point (MD=0).
DY : Down pulse output point (MD=0).
HO : Cumulative output pulse register. (Can be not assigned).
CK : Pulse output point (MD=1).
DR : Up/Down output point (MD=1).
DIR: 1- up; 0- down.

Range / Operand	Y	WX	WY	WM	WS	TMR	CTR	HR	OR	SR	ROR	DR	K
	Yn of Main Unit	WX0 ∣ WX240	WY0 ∣ WY240	WM0 ∣ WM1896	WS0 ∣ WS984	T0 ∣ T255	C0 ∣ C255	R0 ∣ R3839	R3904 ∣ R3967	R3968 ∣ R4167	R5000 ∣ R8071	D0 ∣ D4095	16/32-bit +/- number
MD													0~1
Fr		○	○	○	○	○	○	○	○	○	○	○	8~2000
PC		○	○	○	○	○	○	○	○	○	○	○	○
UY，CK	○												
DY，DR	○												
HO			○	○	○	○	○	○	○	○*	○*	○	

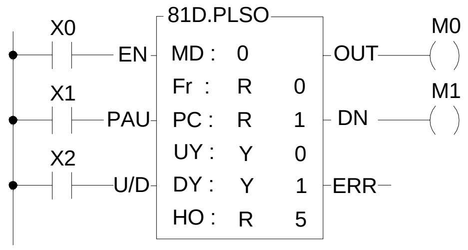
- When MD=0, this instruction performs the pulse output control as following:
- Whenever the output control "EN" changes from 0→1, it first performs the reset action, which is to clear the output flag "OUT" and "DN" as well as the pulse out register HO to be 0. It gets the pulse frequency and output pulse count values, and reads status of up and down direction "U/D", so as to determine the direction to be upward or downward. As the reset finished, this instruction will check the input status of pause output "PAU". No action will be taken if the pause output is 1 (output pause). If the PAU is 0, it will start to output the ON/OFF pulse with 50% duty at the frequency Fr to the UY(U/D=1) or DY(U/D=0) point. It will increment the value of HO register each time when a pulse is output, and will stop the output when HO register's pulse count is equal to or greater than the cumulative pulse count of PC register and set the output complete flag "DN" to 1. During the time when output pulse is transmitting the output transmitting flag "OUT" will be set to 1, otherwise it will be 0.
- Once it starts to transmit pulse, the output control "EN" should kept to 1. If it is changed to 0, it will stop the pulse sending (output point become OFF) and the flag "OUT" changes back to 0, but the other status or data will keep unchanged. However, when its "EN" changes again from 0 to 1, it will lead to a reset action and treat as a new start; the entire procedure will be restarted again.
- If you want to pause the pulse output and not to restart the entire procedure, the 'pause output' "PAU" input can be used to pause it. When "PAU" =1, this instruction will pause the pulse transmitting (output point is OFF, flag "OUT" change back to 0 and the other status or data keeps unchanged). As it waits until the "PAU" changes back from 1 to 0, this instruction will return to the status before it is paused and continues the pulse transmitting output.
- During the pulse transmission, this instruction will keep monitoring the value of pulse frequency Fr and output pulse count PC. Therefore, as long as the pulse output is not finished, it may allow the changing of the pulse frequency and pulse count. However, the up/down direction "U/D" status will be got only once when it takes the reset action ("EN" changes from 0→1), and will keep the status until the pulse output completed or another reset occur. That is to say, except that at the very moment of reset, the change of "U/D" does not influence the operation of this instruction.
- The main purpose of this instruction is to drive the stepping motor with the UY (upward) and DY (downward) two directional pulses control, so as to help you control the forward or reverse rotating of stepping motor. Nevertheless, if you need only single direction revolving, you can assign just one of the UY or DY (which will save one output point), and leaving the other output blank. In such case, the instruction will ignore the up/down input status of "U/D", and the output pulse will send to the output point you assigned.

FUN 81 D PLSO	PULSE OUTPUT	FUN 81 D PLSO

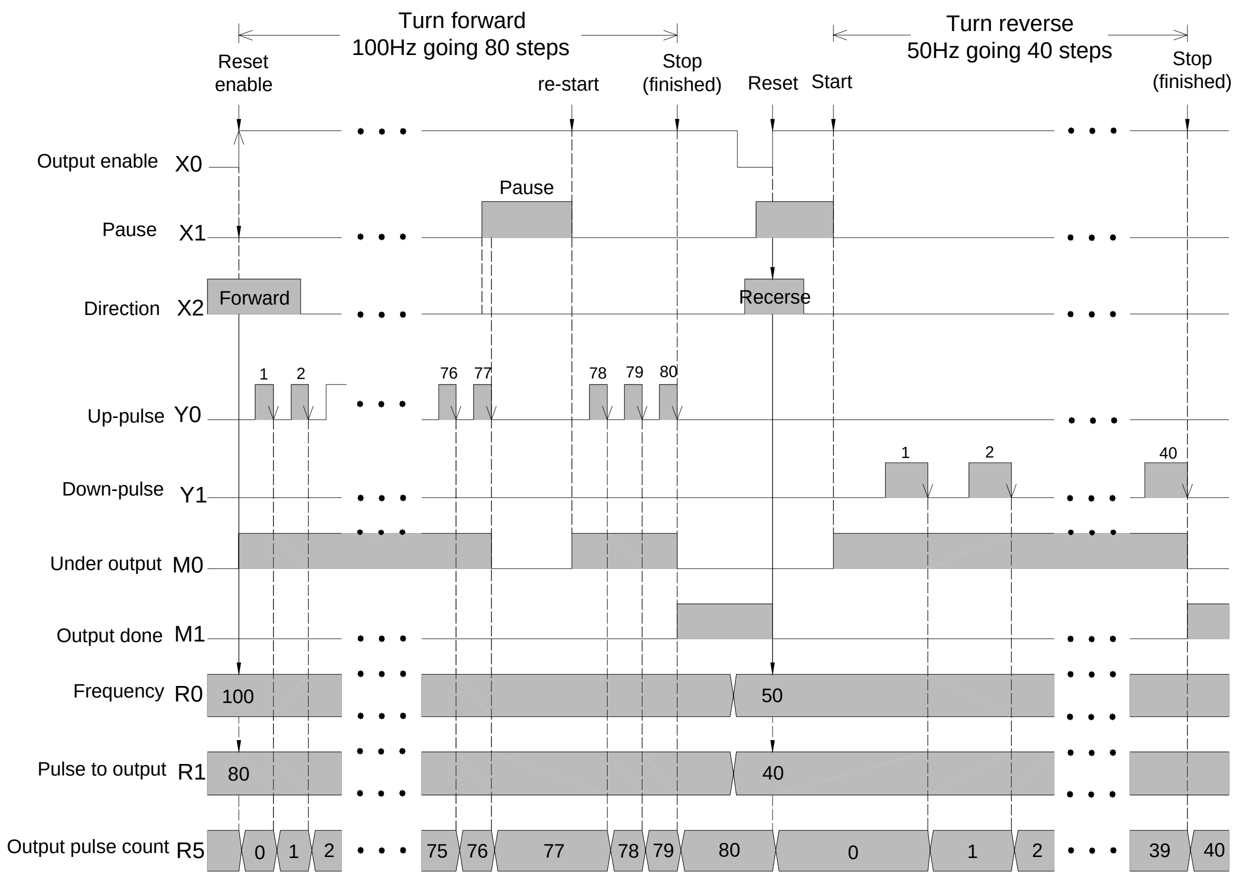
- When MD=1, the pulse output will reflect on the control output DIR (pulse direction. DIR=1, up; DIR=0, down) and CK (pulse output).

- This instruction can only be used once, and UY (CK) and DY (DR) must be transistor output point on the PLC main unit.

- The effective range of output pulse count PC for 16 bit operand is 0～32767. For the 32 bit operand(instruction), it is 0～2147483647. If the PC value = 0, it is treated as infinite pulse count, and this instruction will transmit pulses without end with HO value and "DN" flag set at 0 all the time. The effective range of pulse frequency (Fr) is 8～2000. If the value PC or Fr exceeds the range, this instruction will not be carried out and the error flag "ERR" will set to 1.

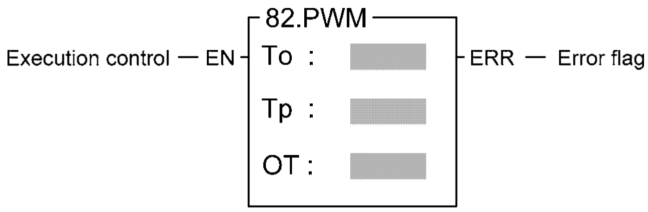


- In this example, the program controls the stepping motor to drive forward for 80 pulses (steps) at the speed of 100Hz first, and then makes it turn reverse for 40 pulses the speed of 50Hz. Make sure that the up/down direction, frequency Fr and the pulse count PC must be set before the reset take action("EN" changes from 0→1).



FUN 82 PWM	PULSE WIDTH MODULATION	FUN 82 PWM

Ladder symbol



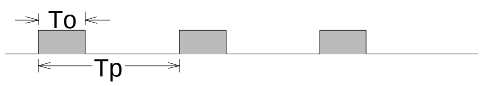
To : Pulse ON width (0~32767mS)

Tp : Pulse period (1~32676mS)

OT: Pulse output point

Range / Ope-rand	Y	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K
	Yn of main unit	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	○													

- When execution control "EN" = 1, will send the pulse to output point OT with the "ON" state for To ms and period as Tp. OT must be a transistor output point on the main unit. When "EN" is 0, the output point will be OFF.

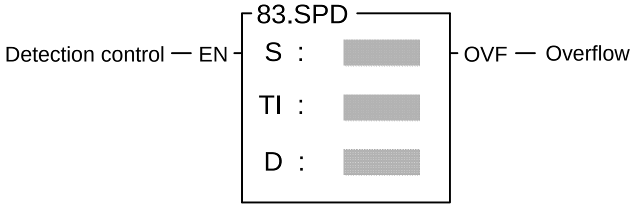


- The units for Tp and To are mS, resolution is 1 mS. The minimum value for To is 0 (under such case the output point OT will always be OFF), and its maximum value is the same as Tp (under such case the output point OT will always be on). If To > Tp there will be an error, this instruction will not be carried out, and the error flag "ERR" will set to 1.

- This instruction can only be used once.

FUN 83 SPD	SPEED DETECTION	FUN 83 SPD

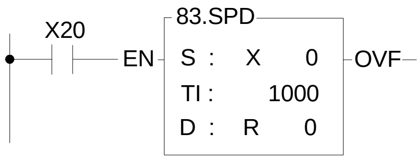
Ladder symbol

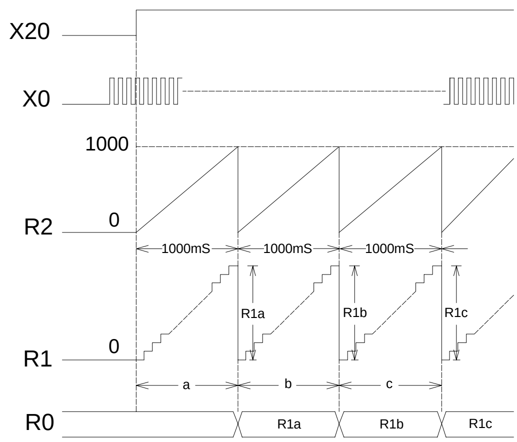


S : Pulse input point for speed detection
TI : Sampling duration (units in mS)
D : Register storing results

Range / Operand	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			○	○	○	○	○	○	○	○	○*	○*	○	

- This instruction uses the interrupt feature of the 8 high speed input points (X0~X7) on the PLC main unit to detect the frequency of the input signal. Within a specific sampling time (TI), it will calculate the input pulse count for S input point, and indirectly find the revolution speed of rotating devices (such as motors).
- While use this instruction to detect the rotating speed of devices, The application should design to generate more pulse per revolution in order to get better result, but the sum of input frequency of all detected signals should under 5KHz, otherwise the WDT may occur.
- The D register for storing results uses 3 successive 16-bit registers starting from D (D0~D2). Besides D0 which is used to store counting results, D1 and D2 are used to store current counting values and sampling duration.
- When detection control "EN" = 1, it starts to calculate the pulse count for the S input point, which can be shown in D1 register. Meanwhile the sampling timer (D2) is switched on and keeps counting until the value of D2 is reach to the sampling period (TI). The final counted value is stored into the D0 register, and then a new counting cycle is started again. The sampling counting will go on repeating until "EN" = 0.
- Because D0 only has 16 bits, so the maximum count is 32767. If the sampling period is too long or the input pulse is too fast then the counted value may exceed 32767, under that case the overflow flag will set to 1, and the counting action will stop.
- Because the sampling period TI is already known and if every revolution of attached rotating device produces "n" pulses, then the following equation can be used to get the revolution speed : $N = \frac{(D0) \times 60}{n \times TI} \times 10^3$ (rpm)

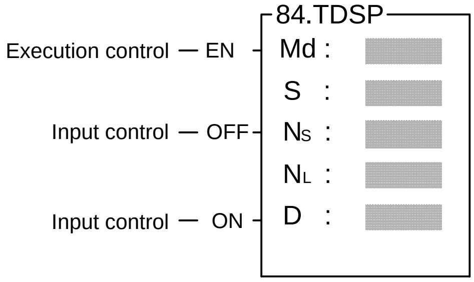




- In the above example, if every revolution of the rotating device produces 20 pulses (n = 20), and the R0 value is 200, then the revolution per minute speed "N" is as follows : $N = \frac{(200) \times 60}{60 \times 1000} \times 10^3 = 200$ rpm

FUN 84 TDSP	PATTERN CONVERSION FOR 16/7-SEGMENT DISPLAY	FUN 84 TDSP

Ladder symbol



Md : Mode selection
S : Starting address of begin converted characters
Ns : Start of character
Nl : Length of character
D : Starting address to store the converted pattern
S operand can be combined with V, Z, P0~P9 index registers for indirect addressing

Range / Operand	HR R0 ∣ R3839	OR R3904 ∣ R3967	ROR R5000 ∣ R8071	DR D0 ∣ D4095	K 16/32 bit	XR V、Z P0~P9
Md					0~1	
S	○	○	○	○	○	○
Ns	○	○	○	○	○	
Nl	○	○	○	○	○	
D	○	○	○*	○		

- This instruction is used for FBs-7SG1/FBs-7SG2 module's application. It can convert the source alphanumeric characters into display patterns suited for 16 segment encoded mode display or perform the leading zero substitution of the packed BCD number for non-decoded mode 7 segment display.

- When execution control "EN" =1, and input "OFF" = 0, input "ON"=0, if Md=0, this instruction will perform the display pattern conversion, where S is the starting address storing the being converted characters, Ns is the pointer to locate the starting address character, NI tells the length of being converted characters, and D is the starting address to store the converted result.

  Byte 0 of S is the "1st" displaying character, byte 1 of S is the 2nd displaying character,……..

  Ns is the pointer to tell where the start character is.

  After execution, each 8-bit character of the source will be converted into the corresponding 16-bit display pattern.

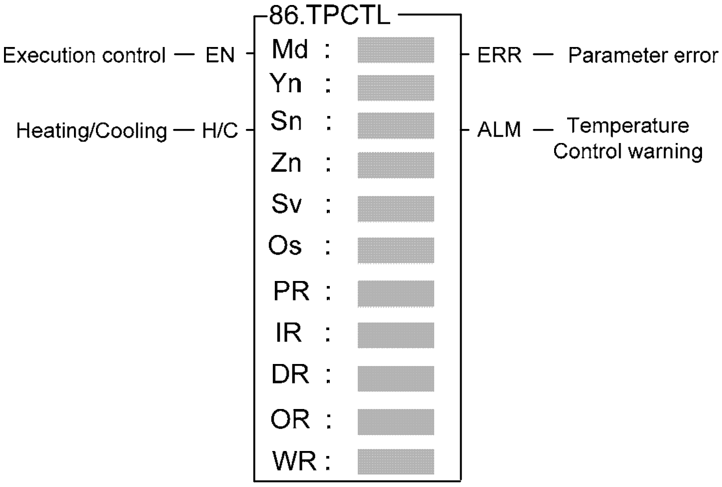
- When input "OFF" = 1, all bits of display pattern will be 'off' if Md = 0, if Md=1, all BCD codes will be substituted by blank code (0F)

- When input "ON" = 1, all bits of display pattern will be 'on' if Md=0. If Md=1, all BCD codes will be substituted by code 8(all light).

- Please refer Chapter 16 "FBs-7SG display module" for more detail description.

FUN 86 TPCTL	PID TEMPERATURE CONTROL INSTRUCTION	FUN 86 TPCTL

Ladder symbol



Range / Operand	Y Y0 ∣ Y255	HR R0 ∣ R3839	ROR R5000 ∣ R8071	DR D0 ∣ D3999	K
Md					0～1
Yn	○				
Sn					0～31
Zn					1～32
Sv		○	○*	○	
Os		○	○*	○	
PR		○	○*	○	
IR		○	○*	○	
DR		○	○*	○	
OR		○	○*	○	
WR		○	○*	○	

Md: Selection of PID method
=0, Modified minimum overshoot method
=1, Universal PID method

Yn: Starting address of PID ON/OFF output; it takes Zn points.

Sn: Starting point of PID control of this instruction; Sn = 0～31.

Zn: Number of the PID control of this instruction; 1≤Sn+Zn≤32

Sv: Starting register of the set point: it takes Zn registers.

Os: Starting register of the in-zone offset; it takes Zn registers.

PR: Starting register of the gain (Kc): it takes Zn registers.

IR: Starting register of integral tuning constant (Ti);it takes Zn registers..

DR: Starting register of derivative tuning constant (Td); it takes Zn registers.

OR: Starting register of the PID analog output. it takes Zn registers.

WR: Starting of working register for this instruction.
It takes 9 registers and can't be repeated in using.

Function guide and notifications

- By employing the temperature module and table editing method to get the current value of temperature and let it be as so called Process Variable (PV); after the calculation of software PID expression, it will respond the error with an output signal according to the setting of Set Point (SP),the error's integral and the rate of change of the process variable. Through the closed loop operation, the steady state of the process may be expected.
- Convert the output of PID calculation to be the time proportional on/off (PWM) output, and via transistor output to control the SSR for heating or cooling process; this is a good performance and very low cost solution.
- Through the analog output module (D/A module), the output of PID calculation may control the SCR or proportional valve to get more precise process control.
- Digitized PID expression is as follows:

$$Mn = [Kc \times En] + \sum_{0}^{n} [Kc \times Ti \times Ts \times En] - [Kc \times Td \times (PVn - PVn\text{-}1)/Ts]$$

Where,
Mn: Output at time "n".
Kc: Gain (Range: 1～9999；Pb=100(%) / Kc)
Ti: Integral tuning constant (Range:0～9999, equivalent to 0.00～99.99 Repeat/Minute)
Td: Derivative tuning constant (Range:0～9999, equivalent to 0.00～99.99 Minute)

FUN 86 TPCTL	PID TEMPERATURE CONTROL INSTRUCTION	FUN 86 TPCTL

PVn : Process variable at time "n"
PVn_1: Process variable when loop was last solved
En: Error at time "n" ; E= SP – PVn
Ts: Solution interval for PID calculation (Valid value are 10, 20, 40, 80,160, 320; the unit is in 0.1Sec)

PID Parameter Adjustment Guide

- As the gain (Kc) adjustment getting larger, the larger the proportional contribution to the output. This can obtain a sensitive and rapid control reaction. However, when the gain is too large, it may cause oscillation. Do the best to adjust "Kc" larger (but not to the extent of making oscillation), which could increase the process reaction and reduce the steady state error.
- Integral item may be used to eliminate the steady state error. The larger the number (Ti, integral tuning constant), the larger the integral contribution to the output. When there is steady state error, adjust the "Ti" larger to decrease the error.
  When the "Ti" = 0, the integral item makes no contribution to the output.
  For exam. , if the reset time is 6 minutes, Ti=100/6=17；if the integral time is 5 minutes, Ti=100/5=20.
- Derivative item may be used to make the process smoother and not too over shoot. The larger the number (Td, derivative tuning constant), the larger the derivative contribution to the output. When there is too over shoot, adjust the "Td" larger to decrease the amount of over shoot.
  When the "Td" = 0, the derivative item makes no contribution to the output.
  For exa, if the rate time is 1 minute, then the Td = 100; if the differential time is 2 minute, then the Td = 200.
- Properly adjust the PID parameters can obtain an excellent result for temperature control.
- The default solution interval for PID calculation is 4 seconds (Ts=40)
- The default of gain value (Kc) is 110, where Pb=1000/110×0.1%≒0.91%; the system full range is 1638°, it means 1638×0.91≒14.8°to enter proportional band control.
- The default of integral tuning constant is 17, it means the reset time is 6 minutes (Ti=100/6=17).
- The default of derivative tuning constant is 50, it means the rate time is 0.5 minutes (Td=50).
- When changing the PID solution interval, it may tune the parameters Kc, Ti, Td again.

Instruction guide

- FUN86 will be enabled after reading all temperature channels.
- When execution control "EN" = 1, it depends on the input status of H/C for PID operation to make heating (H/C=1) or cooling (H/C=0) control. The current values of measured temperature are through the multiplexing temperature module ; the set points of desired temperature are stored in the registers starting from Sv. With the calculation of software PID expression, it will respond the error with an output signal according to the setting of set point, the error's integral and the rate of change of the process variable. Convert the output of PID calculation to be the time proportional on/off (PWM) output, and via transistor output to control the SSR for heating or cooling process; where there is a good performance and very low cost solution. It may also apply the output of PID calculation (stored in registers starting from OR), by way of D/A analog output module, to control SCR or proportional valve, so as to get more precise process control.
- When the setting of Sn, Zn ($0 \leqslant Sn \leqslant 31$ and $1 \leqslant Zn \leqslant 32$, as well as $1 \leqslant Sn + Zn \leqslant 32$) comes error, this instruction will not be executed and the instruction output "ERR" will be ON.

This instruction compares the current value with the set point to check whether the current temperature falls within deviation range (stored in register starting from Os). If it falls in the deviation range, it will set the in-zone bit of that point to be ON; if not, clear the in-zone bit of that point to be OFF, and make instruction output "ALM" to be ON.

FUN 86 TPCTL	PID TEMPERATURE CONTROL INSTRUCTION	FUN 86 TPCTL

- In the mean time, this instruction will also check whether highest temperature warning (the register for the set point of highest temperature warning is R4008). When successively scanning for ten times the current values of measured temperature are all higher than or equal to the highest warning set point, the warning bit will set to be ON and instruction output "ALM" will be on. This can avoid the safety problem aroused from temperature out of control, in case the SSR or heating circuit becomes short.
- This instruction can also detect the unable to heat problem resulting from the SSR or heating circuit runs open, or the obsolete heating band. When output of temperature control turns to be large power (set in R4006 register) successively in a certain time (set in R4007 register), and can not make current temperature fall in desired range, the warning bit will set to be ON and instruction output "ALM" will be ON.
- WR: Starting of working register for this instruction. It takes 9 registers and can't be repeated in using.

  The content of the two registers WR+0 and WR+1 indicating that whether the current temperature falls within the deviation range (stored in registers starting from Os). If it falls in the deviation range, the in-zone bit of that point will be set ON; if not, the in-zone bit of that point will be cleared OFF.

  Bit definition of WR+0 explained as follows:
  Bit0=1, it represents that the temperature of the Sn+0 point is in-zone…
  Bit15=1, it represents that the temperature of the Sn+15 point is in-zone.

  Bit definition of WR+1 explained as follows:
  Bit0=1, it represents that the temperature of the Sn+16 point is in-zone…
  Bit15=1, it represents that the temperature of Sn+31 point is in-zone.

  The content of the two registers WR+2 and WR+3 are the warning bit registers, they indicate that whether there exists the highest temperature warning or heating circuit opened.

  Bit definition of WR+2 explained as follows:
  Bit0=1, it means that there exists the highest warning or heating circuit opened at the Sn+0 point...
  Bit15=1, it means that there exists the highest warning or heating circuit opened at the Sn+15 point.

  Bit definition of WR+11 explained as follows:
  Bit0=1, it means that there exists the highest warning or heating circuit opened at the Sn+16 point...
  Bit15=1 , it means that there exists the highest warning or heating circuit opened at the Sn+31 point.

  Registers of WR+4 ～ WR+8 are used by this instruction.
- It needs separate instructions to perform the heating or cooling control.

Specific registers related to FUN86

- R4005 : The content of Low Byte to define the solution interval between PID calculation
  =0, perform the PID calculation every 1 seconds.
  =1, perform the PID calculation every 2 seconds.
  =2, perform the PID calculation every 4 seconds. (System default)
  =3, perform the PID calculation every 8 seconds.
  =4, perform the PID calculation every 16 seconds.
  ≥5, perform the PID calculation every 32 second.

  : The content of High Byte to define the cycle time of PID ON/OFF（PWM）output.
  =0，PWM cycle time is 1 seconds.
  =1，PWM cycle time is 2 seconds. (System default)
  =2，PWM cycle time is 4 seconds.
  =3，PWM cycle time is 8 seconds.
  =4，PWM cycle time is 16 seconds.
  ≥5，PWM cycle time is 32 second.

Note 1: When changing the value of R4005, the execution control "EN" of FUN86 must be set at 0. The next time when execution control "EN" =1, it will base on the latest set point to perform the PID calculation.

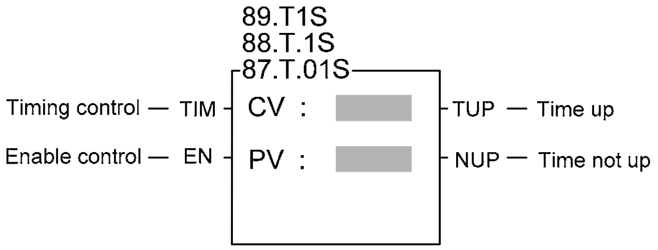
Note 2: The smaller the cycle time of PWM, the more even can it perform the heating. However, the error caused by the PLC scan time will also become greater. For the best control, it can base on the scan time of PLC to adjust the solution interval of PID calculation and the PWM cycle time.

FUN 86 TPCTL	PID TEMPERATURE CONTROL INSTRUCTION	FUN 86 TPCTL

- R4006: The setting point of large power output detection for SSR or heating circuit opened, or heating band obsolete. The unit is in % and the setting range falls in 80～100(%); system default is 90(%).
- R4007: The setting time to detect the continuing duration of large power output while SSR or heating circuit opened, or heating band obsolete. The unit is in second and the setting range falls in 60～65535 (seconds); system default is 600 (seconds).
- R4008: The setting point of highest temperature warning for SSR, or heating circuit short detection. The unit is in 0.1 degree and the setting range falls in 100～65535; system default is 3500 (Unit in 0.1˚).
- R4012: Each bit of R4012 to tell the need of PID temperature control.
  Bit0=1 means that 1st point needs PID temperature control.
  Bit1=1 means that 2nd point needs PID temperature control.
  •
  •
  Bit15=1 means that 16th point needs PID temperature control.
  (The default of R4012 is FFFFH)
- R4013: Each bit of R4013 to tell the need of PID temperature control.
  Bit0=1 means that 17th point needs PID temperature control.
  Bit1=1 means that 18th point needs PID temperature control.
  •
  •
  Bit15=1 means that 32th point needs PID temperature control.
  (The default of R4013 is FFFFH)
- While execution control “EN”=1 and the corresponding bit of PID control of that point is ON (corresponding bit of R4012 or R4013 must be 1), the FUN86 instruction will perform the PID operation and respond to the calculation with the output signal.
- While execution control “EN”=1 and the corresponding bit of PID control of that point is OFF (corresponding bit of R4012 or R4013 must be 0), the FUN86 will not perform the PID operation and the output of that point will be OFF.
- The ladder program may control the corresponding bit of R4012 and R4013 to tell the FUN86 to perform or not to perform the PID control, and it needs only one FUN86 instruction.

FUN87 T.01S FUN88 T.1S FUN89 T1S	ACCUMULATIVE TIMER	FUN87 T.01S FUN88 T.1S FUN89 T1S

Ladder symbol



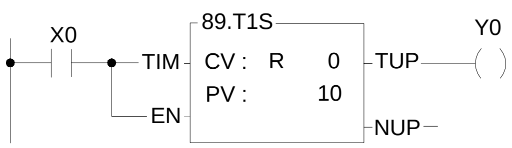
CV : Register storing elapse time (current value)

PV : Preset value of timer

Range / Operand	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	○	○	○	○	○	○	○	○	○	○	○	○	○

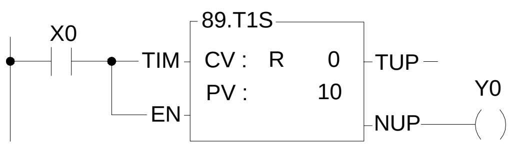
- The operation for this instruction is the same as that for the basic timer (T0~T255), except that the basic timer only has a "timing control" input - when its input is 1 it starts timing, and when input is 0 it get clear. Every time the input changes, it starts timing again and is unable to accumulate. Timing with this instruction is only permissible when enable control "EN" = 1. With this instruction, when timing control "TIM" is 1, it is the same as a basic timer, but when "TIM" is 0, it does not clear, but keeps the current value. If the timer need to clear, then change enable control "EN" to 0. When timing control "TIM" is once again to be 1, it will continue to accumulate from the previous value when the timer last paused. In addition, this instruction also has two outputs, "Time up TUP" (when time up it is 1, usually it is 0) and "Time not up" (usually it is 1, when time is up it is 0). Users can utilize input and output combinations to produce timers with various different functions. For example:

- On delay energizing timer:



- This timer's output (Y0 in this example) is normally not energized. When this timer's input control (X0 in this example) is activated (ON), only after delay by 10 sec will output Y0 become energized (ON).

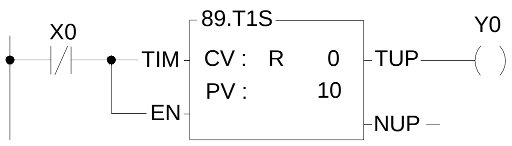
- On delay de-energizing timer:



- The output Y0 of this timer is usually energized. When this timer's input control X0 is on, only after delay by 10 sec will the output become de-energized (OFF).

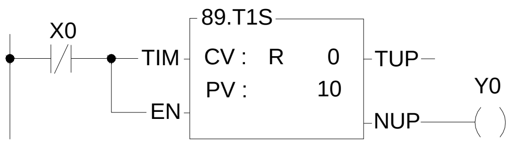
FUN87 T.01S FUN88 T.1S FUN89 T1S	ACCUMULATIVE TIMER	FUN87 T.01S FUN88 T.1S FUN89 T1S

- Off delay energizing timer:



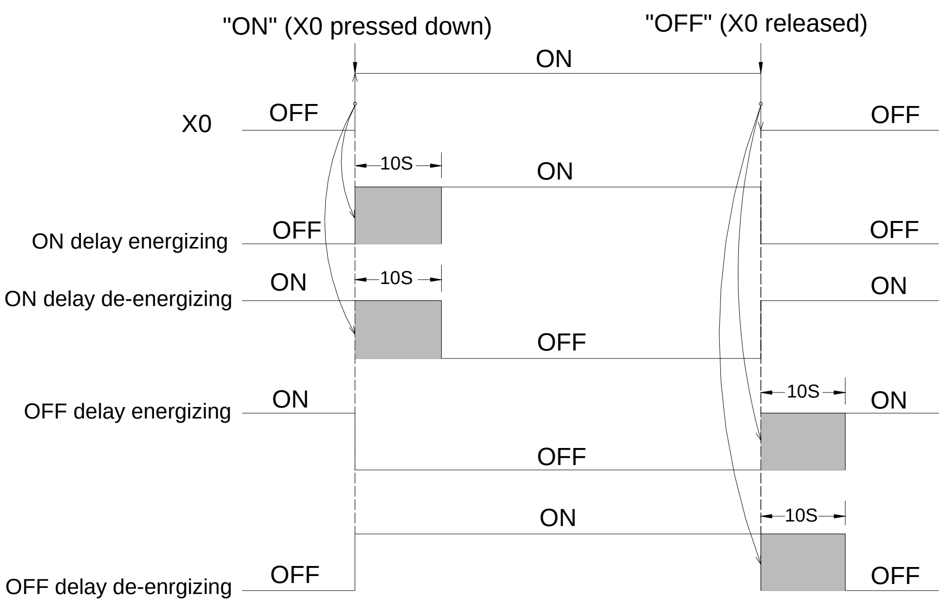
- This timer's output Y0 is usually de-energized. When this timer's input control X0 is off, only after delay by 10 sec will output Y0 become energized (ON).

- Off delay de-energizing timer:



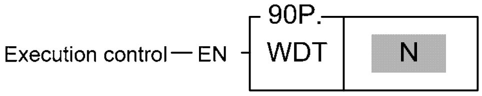
- This timer's output Y0 is usually energized. When this timer's timing control X0 is off, only after delay by 10 sec will output Y0 become de-energized (OFF).

- The diagram below shows the relation on input and output for the above 4 kinds of timers.



FUN 90 P WDT	WATCHDOG TIMER	FUN 90 P WDT

Ladder symbol

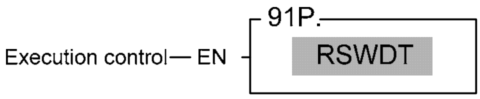


N : The watchdog time. The range of N is 5~120, unit in 10mS (i.e. 50ms~1.2 sec)

- When execution control "EN" = 1 or transition from 0 to 1(P instruction), will set the watchdog time to Nx10ms. If the scan time exceeds this preset time, PLC will shut down and not execute the application program.
- The WDT feature is designed mainly as a safety consideration from the system view for the application. For example, if the CPU of PLC is suddenly damaged, and there is no way to execute the program or refresh I/O, then after the WDT time expired, the WDT will automatically switch off all the I/Os, so as to ensure safety. In certain applications, if the scan time is too long, it may cause safety problems or problems of non-conformance with control requirements. This instruction can used to establish the limitation of the scan time that you require.
- Once the WDT time has been set it will always be kept, and there is no need to set it again on each scan. Therefore, in practice this instruction should use the P instruction.
- Default WDT time is 0.25 sec.
- For the operation principles of WDT please refer to the RSWDT(FUN 91) instruction.

FUN 91 P RSWDT	RESET WATCHDOG TIMER	FUN 91 P RSWDT

Ladder symbol



This instruction has no operand.

- When execution control "EN" = 1 or from 0 to 1 (P instruction), the WDT timer will be reset (i.e. WDT will start timing again from 0).

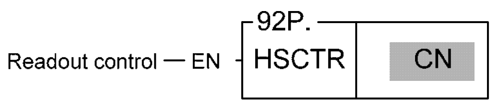
- The functions of WDT have already been described in FUN90 (WDT instruction).
  The operation principles of watch dog timer are as follows:

  The watchdog timer is normally implemented by a hardware one-shot timer (it can not be software, otherwise if CPU fail, the timer becomes ineffective, and safeguards are quite impossible). "One-shot" means that after triggered the timer once, the timing value will immediately be reset to 0 and timing will restart. If WDT has begun timing, and never triggered it again, then the WDT timing value will continue accumulating until it reach the preset value of N, at that time WDT will be activated, and PLC will be shut down. If trigger the WDT once every time before the WDT time N has been reached, then WDT will never be activated. PLC can use this feature to ensure the safety of the system. Each time when PLC enters into system housekeeping after finished the program scanning and I/O refresh, it will usually trigger WDT once, so if the system functions normally and scan time does not exceed WDT time then WDT is never activated. However, if CPU is damaged and unable to trigger WDT, or the scan time is too long, then there will not be enough time to trigger WDT within the period N, WDT will be activated and will shut off PLC.

- In some applications, when you set the WDT time (FUN90) to desire, the scan time of your program in certain situations may temporarily exceed the preset time of WDT. This situation can be anticipated and allowed for, and you naturally do not wish PLC to shut down for this reason. You can use this instruction to trigger WDT once and avoid the activation of WDT. This is the main purpose of this instruction.

FUN 92 D P HSCTR	HARDWARE HIGH SPEED COUNTER CURRENT VALUE (CV) ACCESS	FUN 92 D P HSCTR

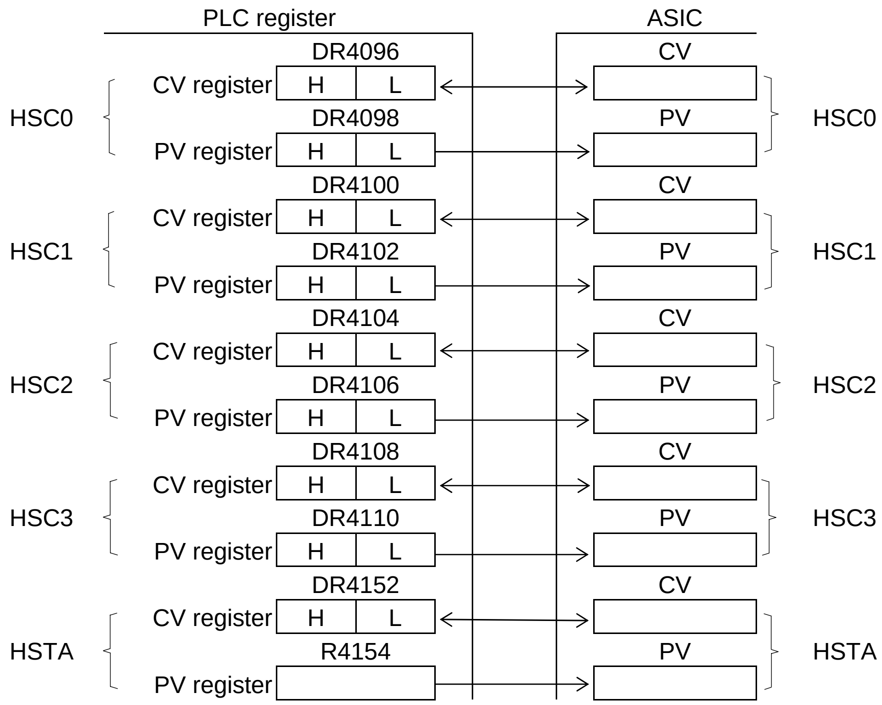
Ladder symbol



CN : Hardware high speed counter number

0: HSC0 or HST0
1: HSC1 or HST1
2: HSC2 or HST2
3: HSC3 or HST3
4: HSTA

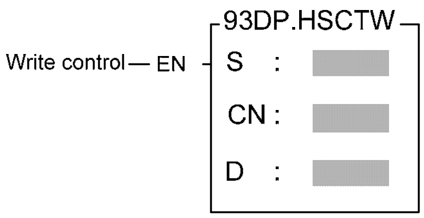
- The HSC0～HSC3 counters of FBs-PLC are 4 sets of 32bit high speed counter with the variety counting modes such as up/down pulse, pulse-direction, AB-phase. All the 4 high speed counters are built in the ASIC hardware and could perform count, compare, and send interrupt independently without the intervention of the CPU. In contrast to the software high speed counters HSC4～HSC7, which employ interrupt method to request for CPU processing, hence if there are many counting signals or the counting frequency is high, the PLC performance (scanning speed) will be degraded dramatically. Since the current values CV of HSC0～HSC3 are built in the internal hardware circuits of ASIC, the user control program (ladder diagram) cannot retrieve them directly from ASIC. Therefore, it must employ this instruction to get the CV value from hardware HSC and put it into the register which control program can access. The following is the arrangement of CV, PV in ASIC and their corresponding CV, PV registers of PLC for HSC0~HSC3.



- When access control "EN" =1 or changes from 0→1(P instruction), will gets the CV value of HSC designated by CN from ASIC and puts into the HSC corresponding CV register (i.e. the CV of HSC0 will be read and put into DR4096 or the CV of HSC1 will be read and put into DR4100).
- Although the PV within ASIC has a corresponding PV register in CPU, but it is not necessary to access it (actually it can't be) for that the PV value within ASIC comes from the PV register in CPU.
- HSTA is a timer, which use 0.1ms as its time base. The content of CV represents elapse time counting at 0.1mS tick.
- For detailed applications, please refer to Chapter 10 "The high speed counter and high speed timer of FBs-PLC".

FUN 93 D P HSCTW	HARDWARE HIGH SPEED COUNTER CURRENT VALUE AND PRESET VALUE WRITING	FUN 93 D P HSCTW

Ladder symbol



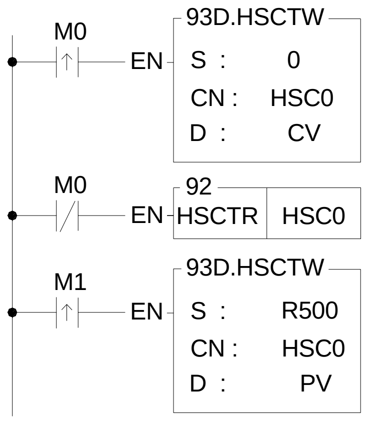
S : The source data for writing

CN : Hardware high speed counter to be written

- 0: HSC0 or HST1
- 1: HSC1 or HST2
- 2: HSC2 or HST3
- 3: HSC3 or HST4
- 4: HSTA

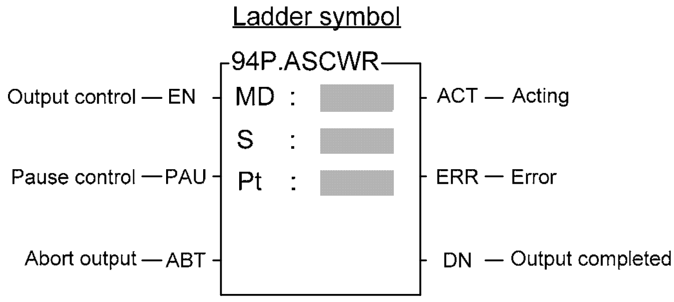
D : Write target (0 represents CV, 1 represents PV)

- Please refer first to FUN92 for the relation between the CV or PV value of HSC0～HSC3 and HSTA within ASIC and their corresponding CV and PV registers in CPU.
- When write control "EN"=1 or changes from 0→1 (P instruction), it writes the content of CV or PV register of high speed counter designed by CN of CPU, to the corresponding CV or PV of HSC within ASIC.
- It is quit often to set the PV value for most application program, When the count value reaches the preset value, the counter will send out interrupt signal immediately. By way of the interrupt service program, you can implement different kinds of precision counting or positioning control.
- When there is an interrupt of power supply for FBs-PLC, the values of current value registers CV of HSC0～HSC3 within ASIC will be read out and wrote into the HSC0～HSC3 CV registers (with power retentive function) of CPU automatically. When power comes up, these CV values will be restored to ASIC. However, if your application demands that when power is on, the values should be cleared to 0 or begin counting from a certain value, then you have to use this instruction to write in the CV value for HSC in ASIC.
- When write a non-zero value into the PV register of HSTA will cause the HSTAI interrupt subroutine to be executed for every PV×0.1ms.
- For detailed applications, please refer Chapter 10 "The high speed counter and high speed timer of FBs-PLC".



- As the program in the left diagram, when M0 changes from 0→1, it clears the current value of HSC0 to 0, and writes into ASIC hardware through FUN93.
- When M0 is 0, it reads out the current counting value.
- When M1 changes from 0→1, it moves DR500 to DR4098, and writes the preset value into ASIC hardware through FUN93.
- Whenever the current value equals to the DR500, The HSC0I interrupt sub program will be executed.

FUN 94 P ASCWR	ASCII WRITE	FUN 94 P ASCWR



MD: Output mode
=0, output to communication port1.
others, reserved for future usage.

S : Starting register of file data.

Pt : Starting working register for this instruction instance. It taken up 8 registers and can't be reused in other part of program.

Range / Operand	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		○	○	○	○	○	○		○	○*	○*	○	

- When MD=0 and output control "ENU" changes from 0→1, it transmits the ASCII data which starting from S to the communication port 1 (Port1), until reach end of file.

- S file data can be edited with the programming software PROLADDER or WinProladder (please refer to the explanation of Chapter 14 "ASCII function application".). If necessary the user can also edit the ASCII file directly by change the value of data registers. However, the edited data must be follow the ASCII file format (the details described in chapter 14), otherwise, this instruction will halt the transmission and set the error flag "ERR" to 1. If the entire file is correctly and successfully transmitted, then the output is completed and "DN" is set to 1.

- The control input of this instruction is of positive edge triggered. Once "ENU" changes from 0→1 then this instruction starts the execution, until finished the transmission of the entire file then the execution is completed. During the transmission, the action flag "ACT" will be kept at 1 all the time. Only when output pause, error, or abort occurs, will it change back to 0.

- This instruction can be repeatedly used, but only one will be executed (transmit data) at any certain time. It is the obligation of user to make sure the right execution sequence.

- While this instruction is in execution, if the pause "PAU" is 1, this instruction will pause the transmission of file data. It will resume transmission when the pause "PAU" backs to 0.

- While this instruction is in execution, if the abort "ABT" is 1, this instruction will abandon the transmission of file data, and then it is able to take next instruction for execution.

- or detail applications, please refer to Chapter 14 "The Application of ASCII file output function".

FUN 94 ASCWR	ASCII WRITE	FUN 94 ASCWR

- Interface signals:

  M1927: This signal is control by CPU, it is applied in ASCWR MD:0

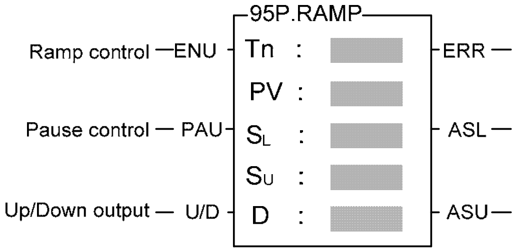
  : ON, it represents that the RTS (connect to the CTS of PLC) of the printer is “False”.
  I.e. the printer is not ready or abnormal.

  : OFF, it represents that the RTS of the Printer is “True”; Printer is Ready.

Note: Using the M1927 associates with timer can detect if the printer is abnormal or not.

FUN 95 P RAMP	RAMP FUNCTION FOR D/A OUTPUT	FUN 95 P RAMP

Ladder symbol



Tn : Timer for ramp function
PV : Preset value of ramp timer (the unit is 0.01 second) or the increment value of every 0.01 second
S_L : Lower limit value (ramp floor value).
S_U : Upper limit value (ramp ceiling value).
D : Register storing current ramping value.
D+1 : Working register
S_U, S_L could be positive or negative value when incorporate with AO module application.

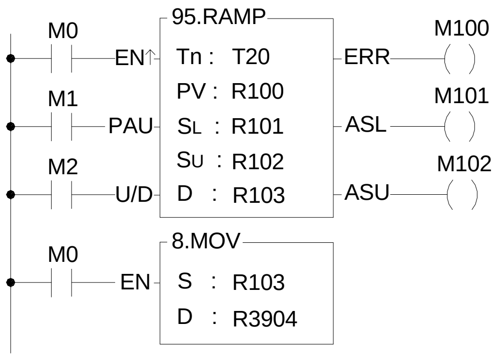
Range / Operand	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K
	WX0 | WX240	WY0 | WY240	WM0 | WM1896	WS0 | WS984	T0 | T255	C0 | C255	R0 | R3839	R3840 | R3903	R3904 | R3967	R3968 | R4167	R5000 | R8071	D0 | D4095	16-bit +/- number
Tn					○								
PV	○	○	○	○	○	○	○	○	○	○	○	○	○
S_L	○	○	○	○	○	○	○	○	○	○	○	○	○
S_U	○	○	○	○	○	○	○	○	○	○	○	○	○
D		○	○	○	○	○	○		○	○	○*	○	

Description

- Tn must be a 0.01 sec time base timer and never used in other part of program.
- PV is the preset value of ramp timer. Its unit is 10ms (0.01 second).
- When input control "ENU" changes from 0→1, it first reset the timer Tn to 0.
  When "U/D"=1 it will load the value of SL to register D. And when M1974 = 0 it will be increased by S_U–S_L / PV every 0.01 sec or when M1974 = 1 it will increase by PV every 0.01 sec. When the D value reaches the S_U value the output "ASU" =1.
  When "U/D"=0 it will load the value of S_U to register D. When M1974 = 0 it will be decreased by S_U–S_L / PV every 0.01 sec or when M1974 = 1 it will be decreased by PV every 0.01 sec. When the D value reaches the S_L value the output "ASL" =1.
- The ramping direction(U/D) is determined at the time when input control "ENU" changes from 0→1. After the output D start to ramp, the change of U/D is no effect.
- If it is required to pause the ramping action, it must let the input control "PAU" = 1; when "PAU"=0, and the ramping action is not completed, it will continue to complete the ramping action.
- The value of S_U must be larger than S_L, otherwise the ramp function will not be performed, and the output "ERR" will set to 1.
- This instruction use the register D to store the output ramping value; if the application use the D/A module to send the speed command, then speed command can be derived from the RAMP function to get a more smooth movement.
- In addition to use register D to store the ramping value, this instruction also used the register D+1 to act as internal working register; therefore the other part of program can not use the register D+1.

FUN 95 P RAMP	RAMP FUNCTION FOR D/A OUTPUT	FUN 95 P RAMP

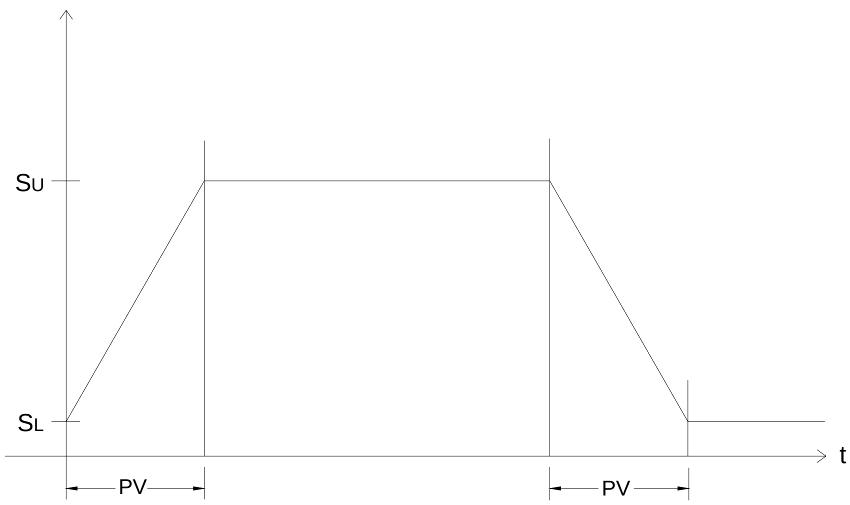
Program example



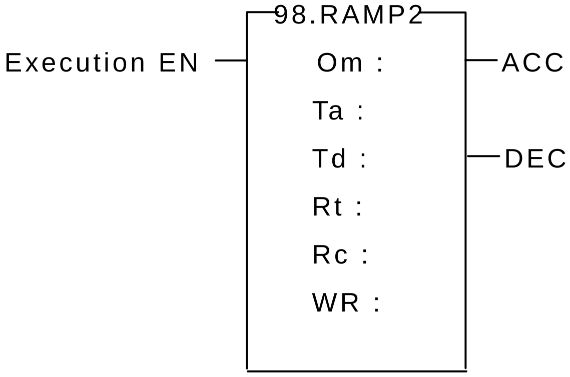
Move the ramping value to AO output register R3904

T20: Ramp timer (timer with 0.01 second time base)
R100: preset value of ramp timer (the unit is 0.01 second, 100 for a second).
R101: Lower limit value.
R102: Upper limit value.
R103: Register storing current ramp value.
R104: Working register

- If M1974=0, When input control M0 changes from 0→1, it first reset the timer T20 to 0. If M2=1, it will load the R101 (lower limit) value into the R103, and it will increase the output with fixed value (R102-R101 / R100) for every 0.01 second and stores it to register R103. When the T2 timer going up to the preset value R100, the output value equals to R102, and the output M102 will set to 1. If M2=0, will load the R102 (upper limit) value into the R103, and it will decrease the output amount with fixed ratio (R102-R101 / R100) for every 0.01 second and store it to register R103. The T2 timer going up to the preset value R100, the output value equals to R102, and the output M101 will set to 1.
- M1=1, pause the ramping action.
- The value of R102 must be greater than R101, otherwise the ramp action will not be performed, and the output M100 will set to 1.



FUN98 RAMP2	TRACKING TYPE RAMP FUNCTION FOR D/A OUTPUT	FUN98 RAMP2



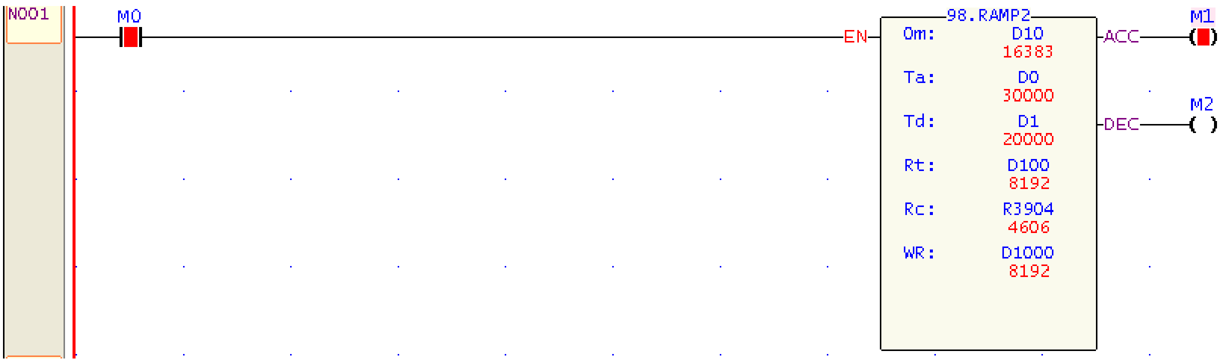
Om：Maximum output; range from 0~65535
Ta ：The acceleration time for the output from 0 up to maximum;
Range from 0~65000, unit is in mS
Td ：The deceleration time for the output from maximum down to 0;
Range from 0~65000, unit is in mS
Rt ：Register of target output;
Range from 0~65535
Rc ：Register of current output, it is used for analog output
WR：Starting address of working registers, it needs 4 registers
＊This instruction can be supported in PLC OS firmware V4.60 or late

Operand \ Range	HR	OR	ROR	DR	K
	R0 ∣ R3839	R3904 ∣ R3967	R5000 ∣ R8071	D0 ∣ D3999	16bit
Om	○	○	○	○	0~65535
Ta	○	○	○	○	0~65000
Td	○	○	○	○	0~65000
Rt	○	○	○	○	
Rc	○	○	○	○	
WR	○	○	○*	○	

- When execution 〝EN〞=0, current output value (Rc) will be 0 immediately; the output indicators ACC=0 and DEC=0.
- When execution 〝EN〞=1, this instruction being executed; it will output current value (Rc) first, and then compare the target output value (Rt) with current output value (Rc) every scan; if the target output value is greater than current output value, the current output will be increased according to the rate, which is decided by the settings of acceleration time (Ta) and maximum output (Om), till current output value is equal to the target output value (ACC=1 during this time); if the target output value is less than current output value, the current output will be decreased according to the rate, which is decided by the settings of deceleration time (Td) and maximum output (Om), till current output value is equal to the target output value (DEC=1 during this time).
- If the setting value of target output (Rt) is greater than maximum output(Om), the output value will be clamped by the maximum value.
- It can have smooth activity for acceleration and deceleration control via the execution of this instruction by using current output value (Rc) for analog output (R39044~R3967).
- The setting value of target output (Rt) needs to stay two scan times at least for proper operation.
- It needs 4 registers for working, they can not be repeated in use。
- This instruction is for positive value operation, but it also can have negative output by short and easy application program for help. Please see example 2.

FUN98 RAMP2	TRACKING TYPE RAMP FUNCTION FOR D/A OUTPUT	FUN98 RAMP2

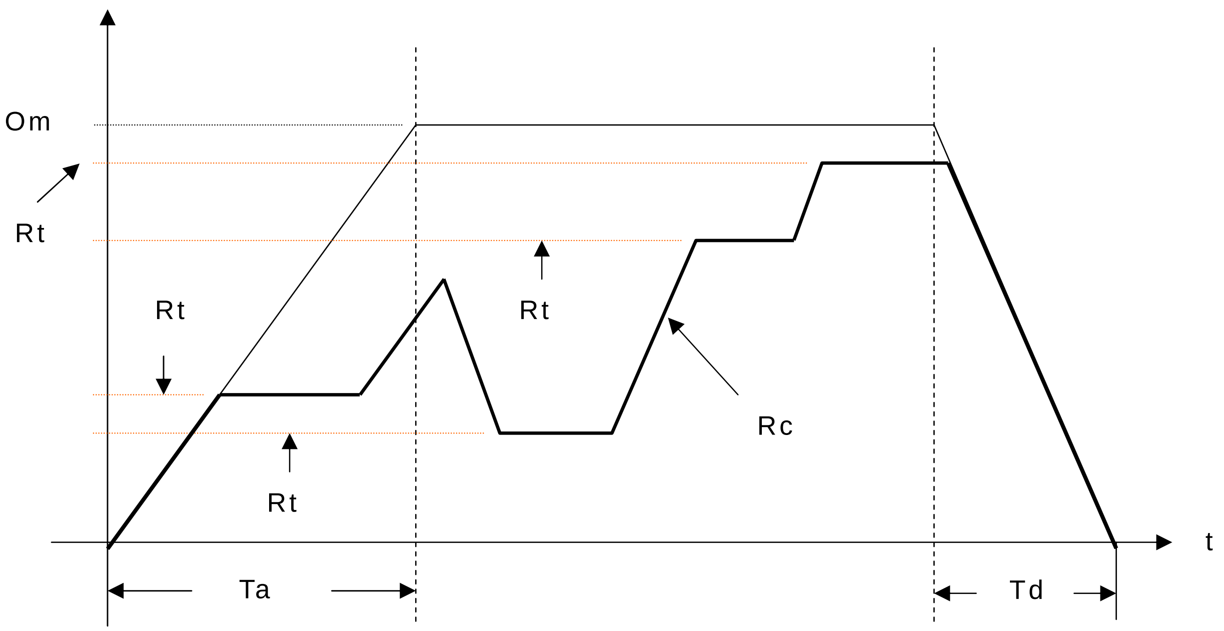
Example 1：Positive output for ACC/DEC control



D10 ：Setting of maximum output, it is 16383
D0 ：The acceleration time for the output from 0 up to maximum, it is 30000mS
D1 ：The deceleration time for the output from maximum down to 0, it is 20000mS
D100：Setting of target output value, it is 8192
R3904：Register of current output, it is used for analog output
D1000~D1003：Working registers

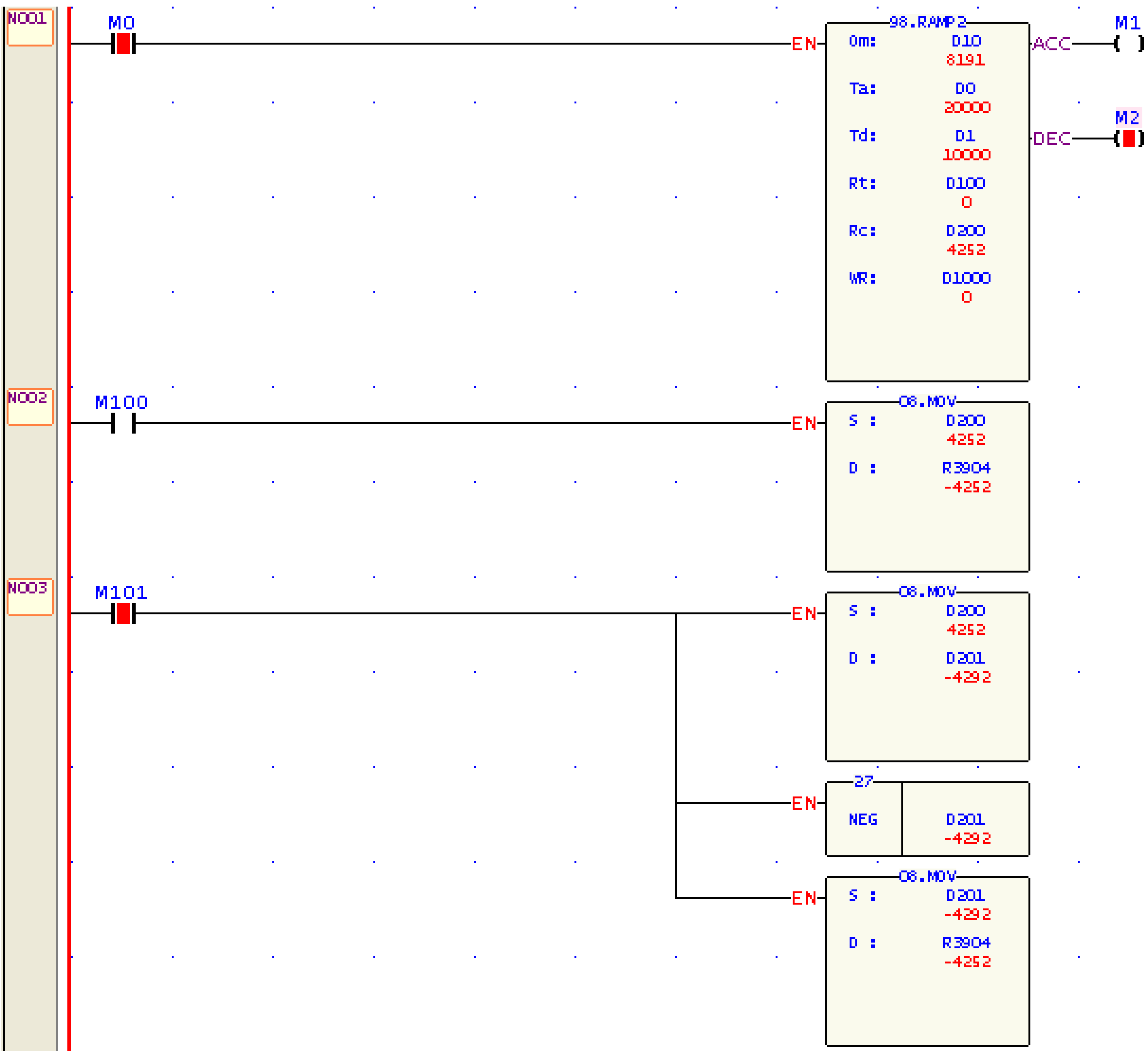
Description: When M0=0, current output value is 0 immediately (No ramp).

When M0=1, it will output the value of R3904 first; and then compare the target output value (D100) with current output value (R3904) every scan; if D100＞R3904, the current output value of R3904 will be increased according to the rate of 16383/30000 (Om=16383, Ta=30000), till R3904=D100 (ACC=1 during this time); if D100＜R3904, the current output value of R3904 will be decreased according to the rate of 16383/20000 (Om=16383, Td=20000), till R3904=D100 (DEC=1 during this time).



FUN98 RAMP2	TRACKING TYPE RAMP FUNCTION FOR D/A OUTPUT	FUN98 RAMP2

Example 2：Both positive and negative output for ACC/DEC control



D10 ：Setting of maximum output, it is 8191
D0 ：The acceleration time for the output from 0 up to maximum, it is 20000mS
D1 ：The deceleration time for the output from maximum down to 0, it is 10000mS
D100：Setting of target output value, it is 0
D200：Register of current output, it is used for analog output
D1000~D1003：Working registers

Description: When M0=0, current output value is 0 immediately (No ramp).

When M0=1, it will output the value of D200 first; and then compare the target output value (D100) with current output value (D200) every scan; if D100＞D200, the current output value of D200 will be increased according to the rate of 8191/20000 (Om=8191, Ta=20000), till D200=D100 (ACC=1 during this time); if D100＜D200, the current output value of D200 will be decreased according to the rate of 8191/10000 (Om=8191, Td=10000), till D200=D100 (DEC=1 during this time).

M100=1, positive output control; M101=1, negative output control.

The target output (D100) is always positive value from 0~65535.

FUN98 RAMP2	TRACKING TYPE RAMP FUNCTION FOR D/A OUTPUT	FUN98 RAMP2

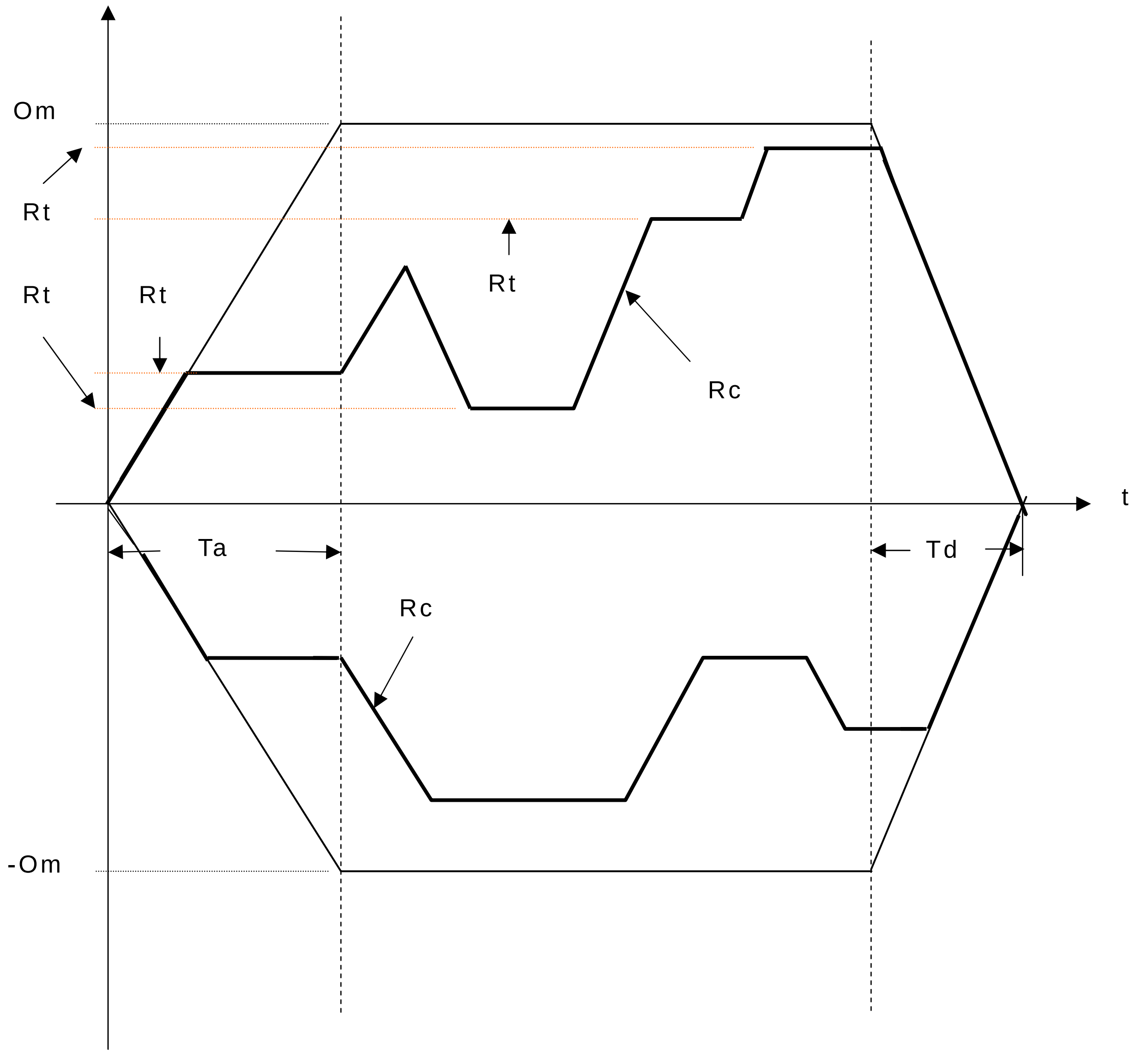


Table Instructions

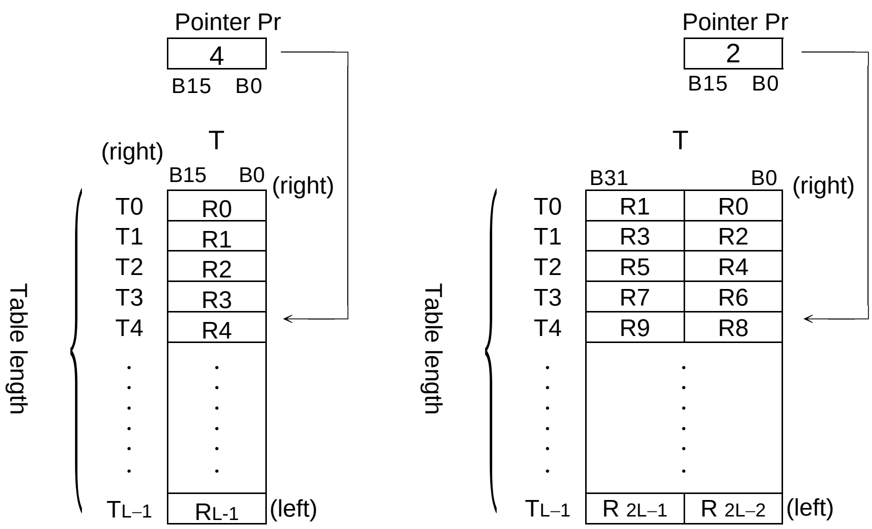
Fun No.	Mnemonic	Functionality	Fun No.	Mnemonic	Functionality
100	R→T	Register to table data move	107	T_FIL	Table fill
101	T→R	Table to register data move	108	T_SHF	Table shift
102	T→T	Table to table data move	109	T_ROT	Table rotate
103	BT_M	Block table move	110	QUEUE	Queue
104	T_SWP	Block table swap	111	STACK	Stack
105	R-T_S	Register to table search	112	BKCMP	Block compare
106	T-T_C	Table to table compare	113	SORT	Data Sort

- A table consists of 2 or more consecutive registers (16 or 32 bits). The number of registers that comprise the table is called the table length (L). The operation object of the table instructions always takes the register as unit (i.e. 16 or 32 bit data).

- The operation of table instructions are used mostly for data processing such as move, copy, compare, search etc, between tables and registers, or between tables. These instructions are convenient for application.

- Among the table instructions, most instructions use a pointer to specify which register within a table will be the target of operation. The pointer for both 16 and 32-bit table instructions will always be a 16-bit register. The effective range of the pointer is 0 to L-1, which corresponds to registers T_0 to T_{L-1} (a total of L registers). The table shown below is a schematic diagram for 16-bit and 32-bit tables.

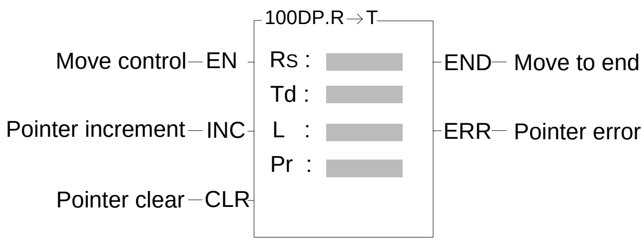
- Among the table operations, shift left/right, rotate left/right operations include a movement direction. The direction toward the higher register is called left, while the direction toward the lower register is called right, as shown in the diagram below.



16bit table　　　32bit table

FUN100 D P R→T	REGISTER TO TABLE MOVE	FUN100 D P R→T

Ladder symbol



Rs : Source data , can be constant or register

Td : Source register for destination table

L : Length of destination table

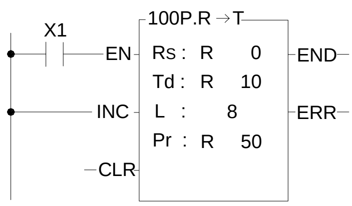
Pr : Pointer register

Rs, Td can associate with V, Z, P0~P9 index register as indirect addressing

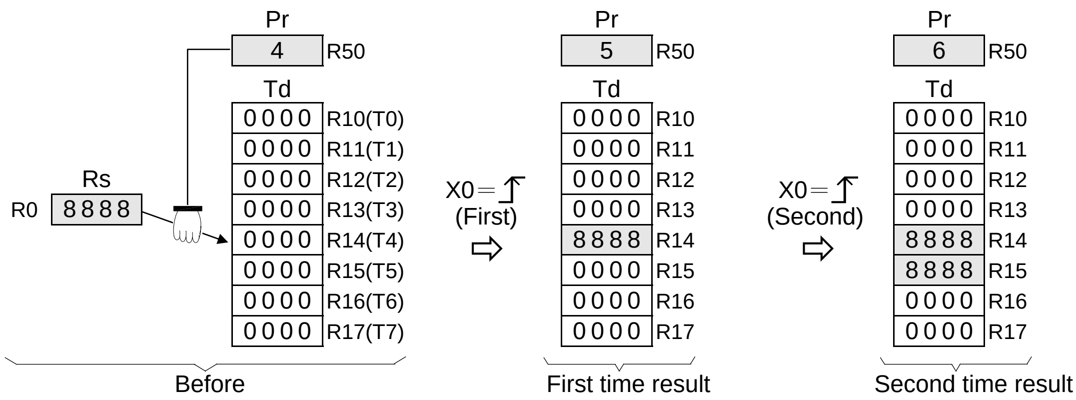
Range / Ope-rand	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR
	WX0 │ WX240	WY0 │ WY240	WM0 │ WM1896	WS0 │ WS984	T0 │ T255	C0 │ C255	R0 │ R3839	R3840 │ R3903	R3904 │ R3967	R3968 │ R4167	R5000 │ R8071	D0 │ D4095	16/32bit +/- number	V、Z P0~P9
Rs	○	○	○	○	○	○	○	○	○	○	○	○	○	○
Td		○	○	○	○	○	○		○	○*	○*	○		○
L							○				○*	○	2~2048	
Pr		○	○	○	○	○	○		○	○*	○*	○		

- When move control "EN" = 1 or transition from 0 to 1(P instruction), the contents of the source register Rs will be written onto the register Tdpr indicated by the pointer Pr within the destination table Td (length is L). Before executing, this instruction will first check the pointer clear "CLR" input signal. If "CLR" is 1, it will first clear the pointer Pr, and then carry out the move operation. After the move has been completed, it will then check the Pr value. If the Pr value has already reached L-1 (point to the last register in the table) then it will only set the move-to-end flag "END" to 1, and finish execution of this instruction. If the Pr value is less than L-1, then it must again check the pointer increment "INC" input signal. If "INC" is 1, then Pr value will be also increased. Besides, pointer clear "CLR" is able to operate independently, without being influenced by other input.

- The effective range of the pointer is 0 to L-1. Beyond this range, the pointer error "ERR" will be set to 1, and this instruction will not be performed.

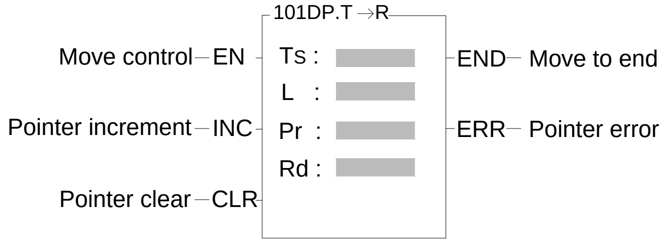


- The example at left at the very beginning pointer Pr = 4, the entire content of table Td is 0, and the Rs value is 8888. The diagram below shows the operation results when X1 have the transition of 0→1 twice.
- Because INC is 1, Pr will increase by 1 each time the instruction is executed.



FUN101 D P T→R	TABLE TO REGISTER MOVE	FUN101 D P T→R

Ladder symbol

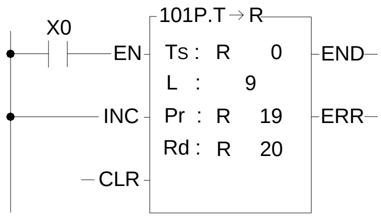


Ts : Source table starting register
L : Length of source table
Pr : Pointer register
Rd : Destination register
Ts, Rd may combine with V, Z, P0~P9 to serve indirect address application

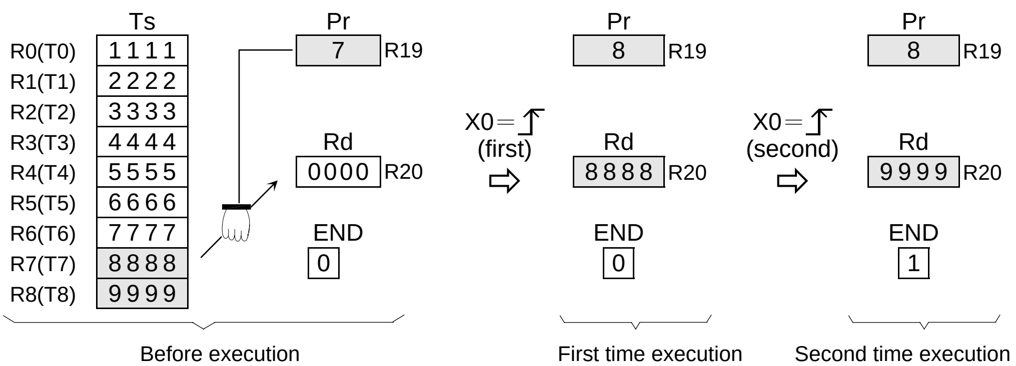
Range / Operand	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR
	WX0 │ WX240	WY0 │ WY240	WM0 │ WM1896	WS0 │ WS984	T0 │ T255	C0 │ C255	R0 │ R3839	R3840 │ R3903	R3904 │ R3967	R3968 │ R4167	R5000 │ R8071	D0 │ D4095	16/32bit +/- number	V、Z P0~P9
Ts	○	○	○	○	○	○	○	○	○	○	○	○		○
L							○				○*	○		
Pr		○	○	○	○	○	○		○	○*	○*	○	2~2048	
Rd		○	○	○	○	○	○		○	○*	○*	○		○

- When move control "EN" = 1 or transition from 0 to 1 (P instruction), the value of the register Tspr specified by pointer Pr within source table Ts (length is L) will be written into the destination register Rd. Before executing, this instruction will first check the input signal of pointer clear "CLR". If "CLR" is 1, it will first clear Pr and then carry out the move operation. After completing the move operation, it will then check the value of Pr. If the Pr value has already reached L-1 (point to the last register in the table), then it sets the move-to-end flag to 1, and finishes executing of this instruction. If Pr is less than L-1, it check the status of "INC". If "INC" is 1, then it will increase Pr and finish the execution of this instruction. Besides, pointer clear "CLR" can execute independently and is not influenced by other inputs.

- The effective range of the pointer is 0 to L-1. Beyond this range the pointer error "ERR" will be set to 1 and this instruction will not be carried out.

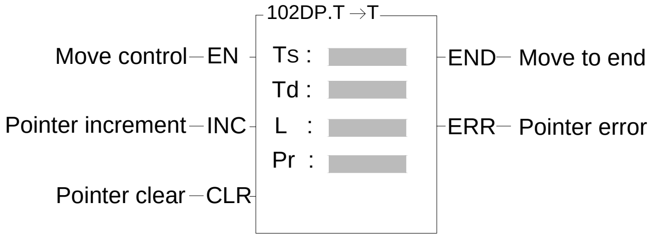


- In the example at left, at the very beginning Pr = 7 and Ts and Rd are as shown at left in the diagram below. When X0 have a transition from 0→1 twice, the results are shown at right in the diagram below.
- At the second time execution, the pointer has already reached to the end so there will be no increment.



FUN102 D P T→T	TABLE TO TABLE MOVE	FUN102 D P T→T

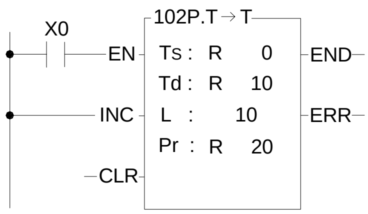
Ladder symbol



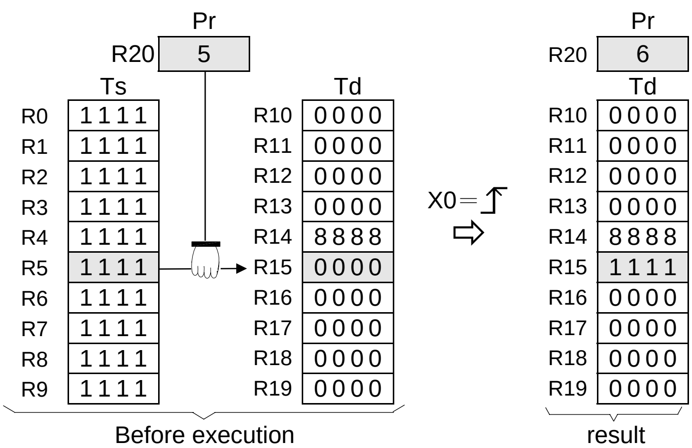
Ts : Starting number of source table register
Td : Starting number of destination table register
L : Table (Ts and Td) length
Pr : Pointer register
Ts, Td may combine with V, Z, P0~P9 to serve indirect address application

Range / Operand	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		○	○	○	○	○	○		○	○*	○*	○		

- When move control "EN" = 1 or have a transition from 0 to 1(P instruction), the register Tspr pointed by pointer Pr within the source table will be moved to a register Tdpr, which also pointed by the pointer Pr in the destination table. Before execution, it will first check the input signal of pointer clear "CLR". If "CLR" is 1, it will first clear Pr to 0 and then do the move (in this case Ts0→Td0). After the move action has been completed it will then check the value of pointer Pr. If the Pr value has already reached L-1 (point to the last register on the table), then it will set the move-to-end flag "END" to 1 and finish executing of this instruction. If the Pr value is less than L-1, it will check the status of "INC". If "INC" is 1, then the Pr value will be increased by 1 before execution. Besides, pointer clear "CLR" can execute independently, and will not be influenced by other input.
- The effective range of the pointer is 0 to L-1. Beyond this range, the pointer error flag "ERR" will be set to 1, and this instruction will not be carried out.

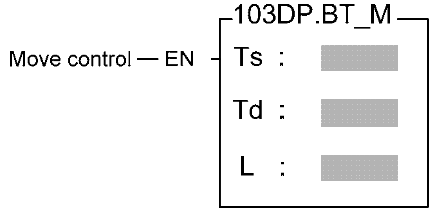


- The diagram at left below is the status before execution. When X0 from 0→1, the content of R5 in Ts table will copy to R15 and pointer R20 will be increased by 1.



FUN103 D P BT_M	BLOCK TABLE MOVE	FUN103 D P BT_M

Ladder symbol



Ts :Starting register for source table

Td : Starting register for destination table

L: Lengths of source and destination tables

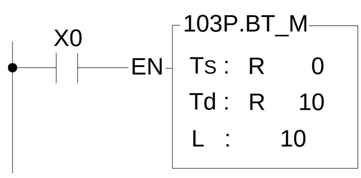
Ts, Td may combine with V, Z, P0~P9 to serve indirect

Range / Operand	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							○				○*	○	○	

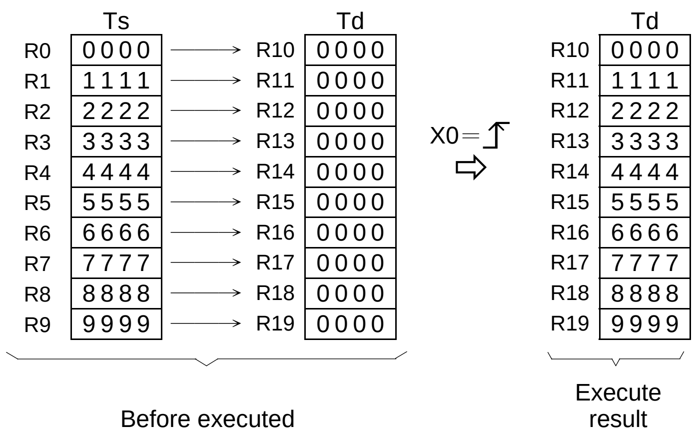
- In this instruction the source table and destination table are the same length. When this instruction was executed all the data in the Ts table is completely copied to Td. No pointer is involved in this instruction.

- When move control "EN" = 1 or have a transition from 0 to 1 (P instruction), all the data from source table Ts (length L) is copied to the destination table Td, which is the same length.

- One table is completely copied every time this instruction is executed, so if the table length is long, it will be very time consuming. In practice, P modifier should be used to avoid time waste caused by each scan repeating the same movement action.

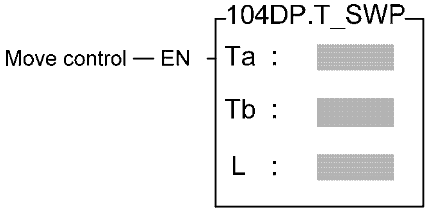


- The diagram at left below is the status before execution. When X0 from 0→1, the content of R0~R9 in Ts table will copy to R10~R19.



FUN104 D P T_SWP	BLOCK TABLE SWAP	FUN104 D P T_SWP

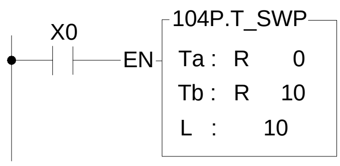
Ladder symbol



Ta : Starting register of Table a
Tb : Starting register of Table b
L : Lengths of Table a and b
Ts, Td may combine with V, Z, P0~P9 to serve indirect address application

Range / Ope-rand	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						○			○*	○	○	

- This instruction swaps the contents of Tables a and b, so the table must be the same length, and the registers in the table must of write able. Since a complete swap is done with each time the instruction is executed, no pointer is needed.
- When move control "EN" = 1 or have a transition from 0 to 1 (P instruction), the contents of Table a and Table b will be completely swapped.
- This instruction will swap all the registers specified in L each time the instruction is executed, so if the table length is big, it will be very time consuming, therefore P instruction should be used.



- The diagram at left below is the status before execution. When X0 from 0→1, the contents of R0~R9 in Ts table will swap with R10~R19.

Before executed:

	Ta		Tb
R0	0000	R10	1111
R1	0000	R11	1111
R2	0000	R12	1111
R3	0000	R13	1111
R4	0000	R14	1111
R5	0000	R15	1111
R6	0000	R16	1111
R7	0000	R17	1111
R8	0000	R18	1111
R9	0000	R19	1111

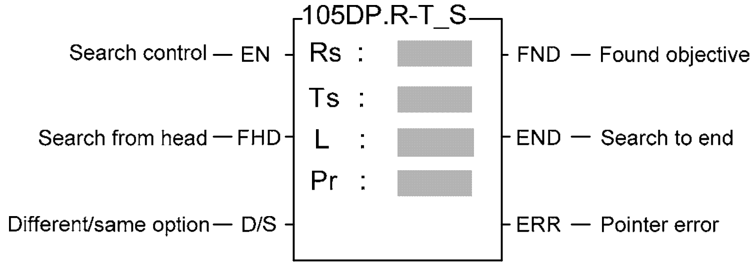
X0 = ↑ ⇨

After executed:

	Ta		Tb
R0	1111	R10	0000
R1	1111	R11	0000
R2	1111	R12	0000
R3	1111	R13	0000
R4	1111	R14	0000
R5	1111	R15	0000
R6	1111	R16	0000
R7	1111	R17	0000
R8	1111	R18	0000
R9	1111	R19	0000

FUN105 D P R-T_S	REGISTER TO TABLE SEARCH	FUN105 D P R-T_S

Ladder symbol

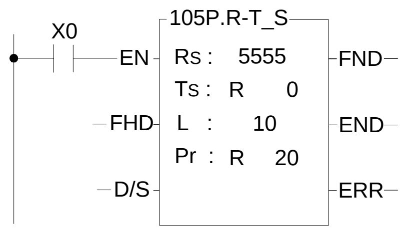


Rs : Data to search, It can be a constant or a register
Ts : Starting register of table being searched
L : Label length
Pr : Pointer of table
Rs, Ts may combine with V, Z, P0~P9 to serve indirect address application

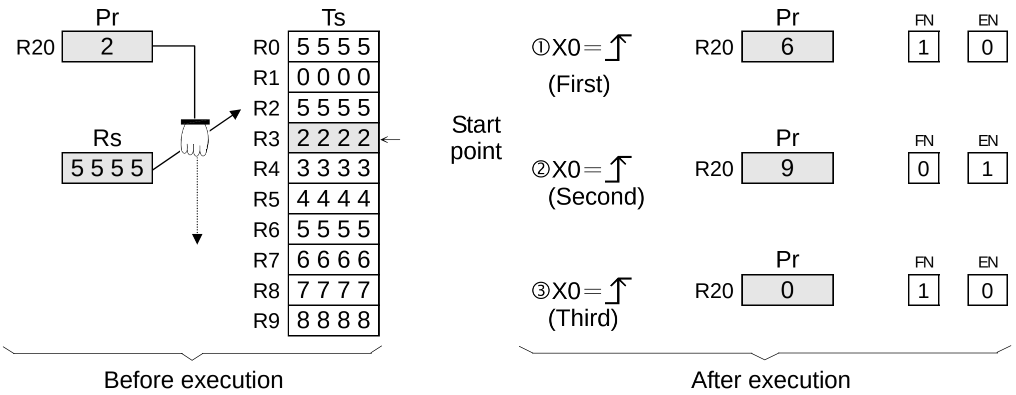
Range / Operand	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR
	WX0 | WX240	WY0 | WY240	WM0 | WM1896	WS0 | WS984	T0 | T255	C0 | C255	R0 | R3839	R3840 | R3903	R3904 | R3967	R3968 | R4167	R5000 | R8071	D0 | D4095	16/32-bit +/- number	V、Z P0~P9
Rs	○	○	○	○	○	○	○	○	○	○	○	○	○	○
Ts	○	○	○	○	○	○	○	○	○	○	○	○		○
L							○				○*	○	2~256	
Pr		○	○	○	○	○	○		○	○*	○*	○		

- When search control "EN" = 1 or has a transition from 0 to 1 (P instruction), will search from the first register of Table Ts (when "FHD" = 1 or Pr value has reached L-1), or from the next register (Tspr + 1) pointed by the pointer within the table ("FHD" = 0, while Pr value is less than L-1) to find the first data different with Rs(when D/S = 1) or find the first data the same with Rs (when D/S = 0). If it find a data match the condition it will immediately stop the search action, and the pointer Pr will point to that data and found objective flag "FND" will set to 1. When the searching has searched to the last register of the table, the execution of the instruction will stop, whether it was found or not. In that case the search-to-end flag "END" will be set to 1 and the Pr value will stop at L-1. When this instruction next time is executed, Pr will automatically return to the head of the table (Pr = 0) before the search begin.

- The effective range of Pr is 0 to L-1. If the value exceeds this range then the pointer error flag "ERR" will change to 1, and this instruction will not be carried out.

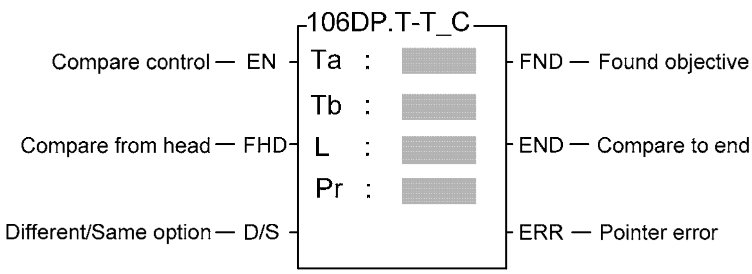


- The instruction at left is searching the table for a register with the value 5555 (because D/S = 0, it is searching for same value). Before execution, the pointer point to R2, but the starting point of the search is Pr + 1 (i.e. it starts from R3). After X0 has transition from 0→1 3 times, the results of each search may be obtained as shown in the diagram below.



FUN106 D P T-T_C	TABLE TO TABLE COMPARE	FUN106 D P T-T_C

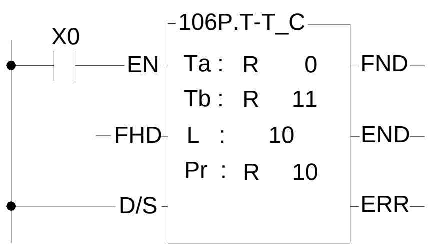
Ladder symbol



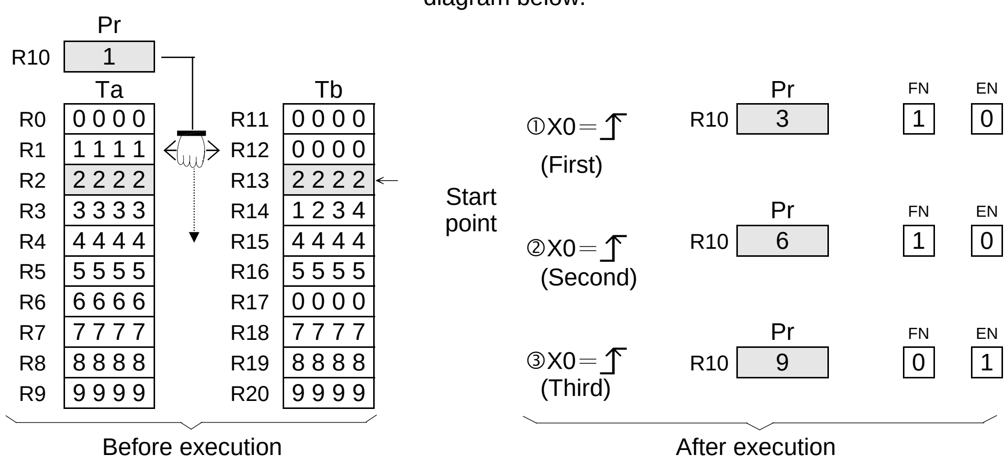
Ta : Starting register of Table a
Tb : Starting register of Table b
L : Lengths of Table
Pr : Pointer
Ta, Tb may combine with V, Z, P0~P9 to serve indirect address application

Range / Operand	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		○	○	○	○	○	○		○	○*	○*	○		

- When comparison control "EN" = 1 or has a transition from 0 to 1(P instruction), then starting from the first register in the tables Ta and Tb (when "FHD" = 1 or Pr value has reached L-1) or starting from the next pair of registers (Tapr+1 and Tbpr+1) pointed by Pr ("FHD" = 0, while Pr is less than L-1), this instruction will search for pairs of registers with different values (when "D/S" = 1) or the same value (when "D/S" = 0). When search found (either different or the same), it will immediately stop the search and the pointer Pr will point to the register pairs met the search criteria. The found flag "FND" will be set to 1. When it has searched to the last register of the table, the instruction will stop executing. whether it found or not. The compare-to-end flag "END" will be set to 1, and the pointer value will stop at L-1. When this instruction is executed next time, Pr will automatically return to the head of the table to begin the search. The effective range of Pr is 0 to L-1. The Pr value should not changed by other programs during the operation. As this will affect the result of the search. If the Pr value not in the effective range, the pointer error flag "ERR" will be set to 1, and this instruction will not be carried out.

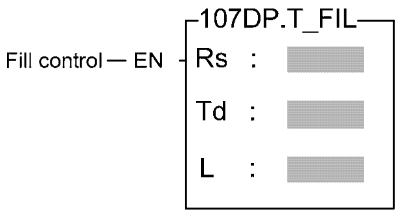


- The instruction at left starts from the register next to the register pointed by the pointer (because "FHD" is 0) to search for register pairs with different data (because "D/S" is 1) within the 2 tables. At the very beginning, Pr points to Ta1 and Tb1. There are 3 different pairs of data at the position 1,3,6 of the table. However, it does not compare from the beginning, and this instruction will start searching from position 3 downwards. After X0 has changed 3 times from 0 to 1, the results are shown in the diagram below.



FUN107 D P T_FIL	TABLE FILL	FUN107 D P T_FIL

Ladder symbol



Rs : Source data to fill, can be a constant or a register

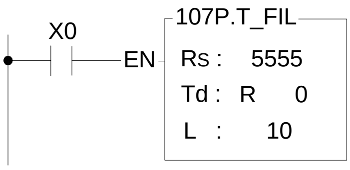
Td : Starting register of destination table

L :Table length

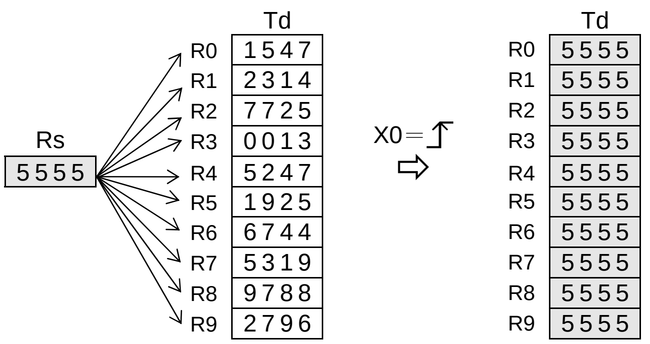
Rs, Td may combine with V, Z, P0~P9 to serve indirect address application

Range / Operand	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	16/32-bit +/- number	V、Z P0~P9
Ts	○	○	○	○	○	○	○	○	○	○	○	○	○	○
Td		○	○	○	○	○	○		○	○*	○*	○		○
L							○				○*	○	2~256	

- When fill control "EN" = 1 or has a transition from 0 to 1 (P instruction), the Rs data will be filled into all the registers of the table Td.
- This instruction is mainly used for clearing the table (fill 0) or unifying the table (filling in the same values). It should be used with the P instruction.

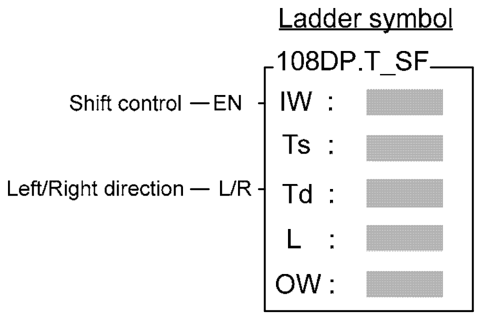


- The instruction at left will fill 5555 into the whole table Td. The results are as shown in the diagram below.



Before execution　　　　After execution

FUN108 D P T_SHF	TABLE SHIFT	FUN108 D P T_SHF



IW : Data to fill the room after shift operation, can be a constant or a register

Ts : Source table

Td : Destination table storing shift results

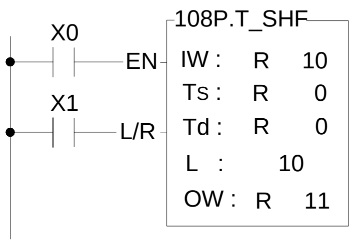
L : Lengths of tables Ts and Td

OW: Register to accept the shifted out data

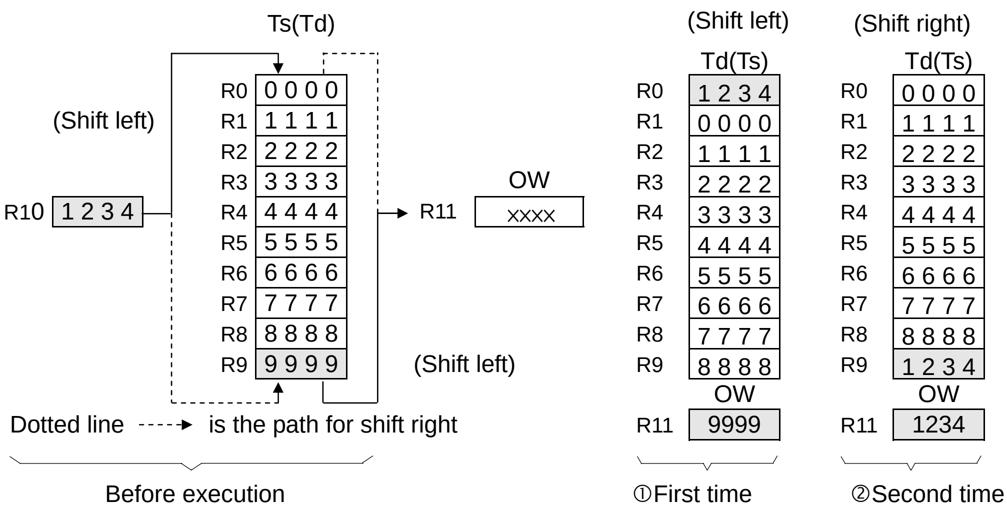
Ts, Td may combine with V, Z, P0~P9 to serve indirect address application

Range / Ope-rand	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	16/32-bit +/- number	V、Z P0~P0
IW	○	○	○	○	○	○	○	○	○	○	○	○	○	
Ts	○	○	○	○	○	○	○	○	○	○	○	○		○
Td		○	○	○	○	○	○		○	○*	○*	○		○
L							○				○*	○	2~256	
OW		○	○	○	○	○	○		○	○*	○*	○		

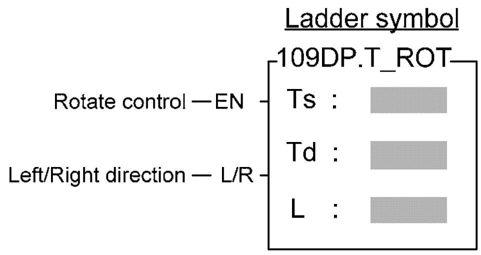
- When shift control "EN" = 1 or has a transition from 0 to 1(P instruction), all the data from table Ts will be taken out and shifted one position to the left (when "L/R" = 1) or to the right (when "L/R" = 0). The room created by the shift operation will be filled by IW and the results will be written into table Td. The data shifted out will be written into OW.



- In the program at left, Ts and Td is the same table. Therefore, the table shifts itself and then writes back to itself (the table must be writ able). It first perform a shift left operation (let X1 = 1, and X0 go from 0→1) then perform a shift to right operation (let X1 = 0, and makes X0 go from 0 →1). The result are shown at right in the diagram below.



FUN109 D P T_ROT	TABLE ROTATE	FUN109 D P T_ROT



Ts : Source table for rotate

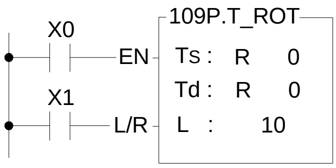
Td : Destination table storing results of rotation

L : Lengths of table

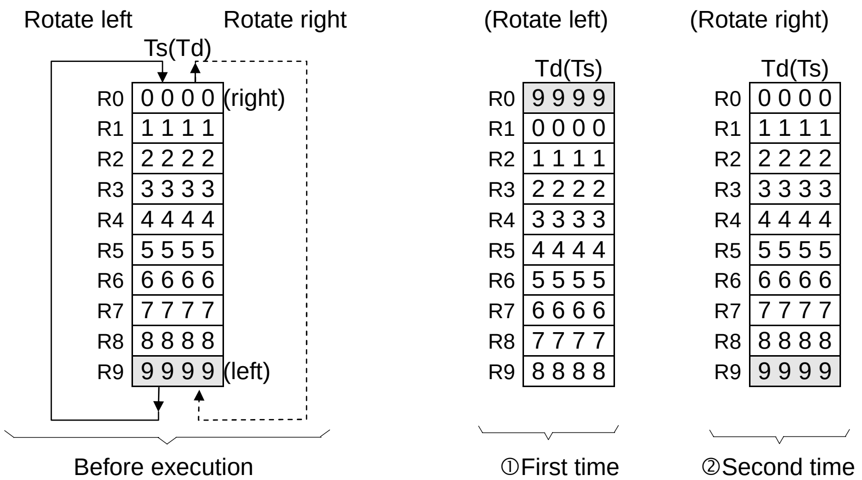
Ts, Td may combine with V, Z, P0~P9 to serve indirect address application

Range / Ope-rand	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							○				○*	○	○	

- When rotation control "EN" = 1 or has a transition from 0 to 1(P instruction), the data from the table of Ts will be rotated 1 position to the left (when "L/R" = 1)or 1 position to the right (when "L/R" = 0). The results of the rotation will then be written onto table Td.

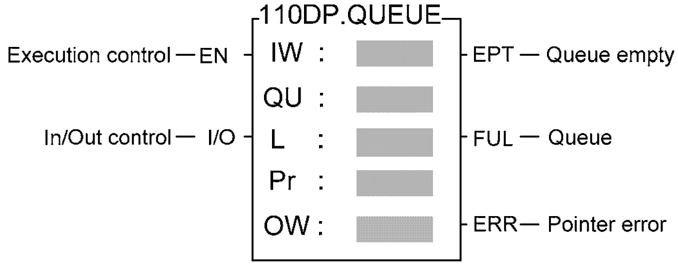


- In the program at left, Ts and Td is the same table. The table after rotation will write back to itself. It first perform one left rotation (let X1 = 1, and X0 go from 0→1), and then performs one right rotation (let X1 = 0, and X0 go from 0→1). The results are shown at right in the diagram below.



FUN110 D P QUEUE	QUEUE	FUN110 D P QUEUE

Ladder symbol



IW : Data pushed into queue, can be a constant or a register

QU : Starting register of queue

L : Size of queue

Pr : Pointer register

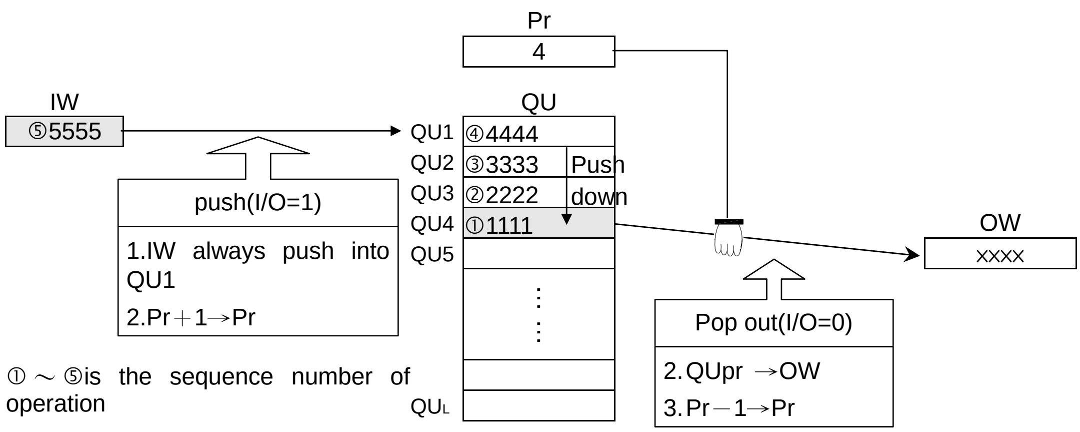
OW : Register accepting data popped out from queue

QU may combine with V, Z, P0~P9 to serve indirect address application

Range / Operand	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	16/32-bit +/- number	V、Z P0~P9
IW	○	○	○	○	○	○	○	○	○	○	○	○	○	
QU		○	○	○	○	○	○		○	○	○*	○		○
L							○				○*	○	2~256	
Pr		○	○	○	○	○	○		○	○*	○*	○		
OW		○	○	○	○	○	○		○	○*	○*	○		

- Queue is also a kind of table. It is different from ordinary table in that its queue register numbers go from 1 to L and not from 0 to L-1. In other words QU_1~QU_L respectively correspond to pointers Pr = 1 to L, and Pr = 0 is used to show that the queue is empty.

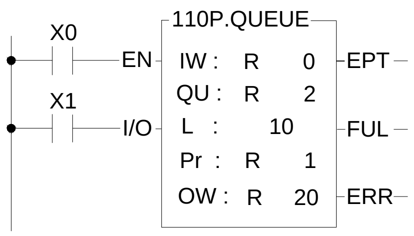
- Queue is a first in first out (FIFO) device, i.e. - the data that first pushed into the queue will be the first to pop out from the queue. A queue is comprised of L consecutive 16 or 32 bit registers (D instruction) starting from the QU register, as in the diagram below:



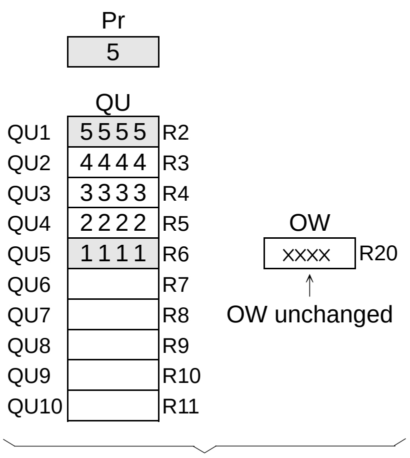
- When execution control "EN" = 1 or has a transition from 0 to 1 (P instruction), the status of in/out control "I/O" determines whether the IW data will be pushed into the queue (when "I/O" = 1) or be popped out and transferred to OW (when "I/O" = 0). As shown in the diagram above, the IW data will always be pushed into the first (QU1) register of the queue. After it has been pushed in, Pr will immediately be increased by 1, so that the pointer can always point to the first data that was pushed into the queue. When it is popped out, the data pointed by Pr will be transferred directly to OW. Pr will be reduced by 1, so that it still point to the first data remained in the queue.

FUN110 D P QUEUE	QUEUE	FUN110 D P QUEUE

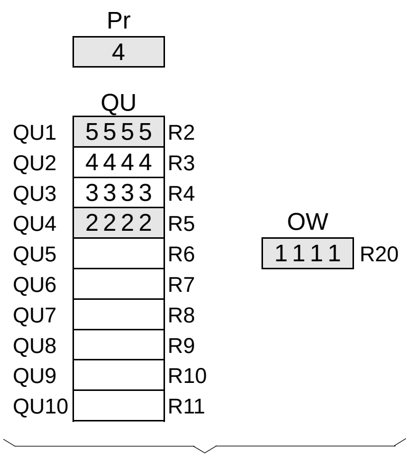
- If no data has yet been pushed into the queue or the pushed in data has already been popped out (Pr = 0), then the queue empty flag will be set to 1. In this case, even if there is further popping out action, this instruction will not be executed. If data is only pushed in and not popped out, or pushed in is more than that popped out, then the queue finally becomes full (pointer Pr indicates the QU_L position), and the queue full flag is changed to 1. In this case, if there is more pushing in action, this instruction will not execute. The pointer for this instruction is used during access of the queue, to indicate the data that was pushed in the earliest. Other programs should not be allowed to change it, or else an operation error will be created. If there is a specific application, which requires the setting of a Pr value, then its permissible range is 0 to L (0 means empty, and 1 to L respectively correspond to QU1 to QUL). Beyond this range, the pointer error flag "ERR" will be set as 1, and this instruction will not be carried out.



- The program at left assumes the queue content is the same with the queue at preceding page. It will first perform queue push operation, and then perform pop out action. The results are shown below. Under any circumstance, Pr always point to the first (oldest) data that was remained in queue.

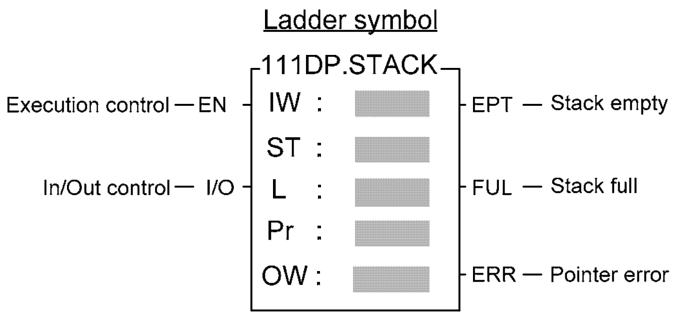


After push in (X1=1，X0 from 0→1)



After pop off (X1=0，X0 from 0→1)

FUN111 D P STACK	STACK	FUN111 D P STACK



IW : Data pushed into stack, can be a constant or a register

ST : Starting register of stack

L : Size of stack

Pr : Pointer register

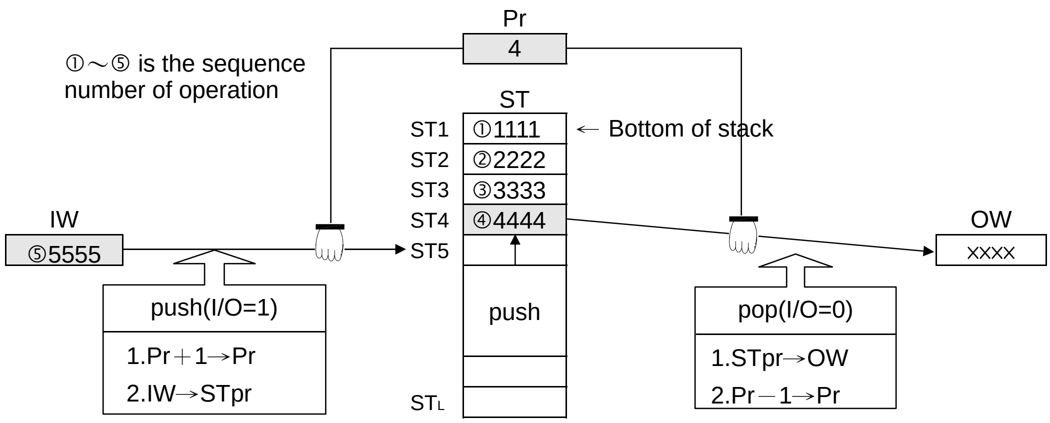
OW : Register accepting data popped out from stack

ST may combine with V, Z, P0~P9 to serve indirect address application

Range / Operand	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	16/32-bit +/- number	V、Z P0~P9
IW	○	○	○	○	○	○	○	○	○	○	○	○	○	
ST		○	○	○	○	○	○		○	○*	○*	○		○
L							○				○*	○	2~256	
Pr		○	○	○	○	○	○		○	○*	○*	○		
OW		○	○	○	○	○	○		○	○*	○*	○		

- Like queue, stack is also a kind of table. The nature of its pointer is exactly the same as with queue, i.e. Pr = 1 to L, which corresponds to ST_1 to ST_L, and when Pr = 0 the stack is empty.

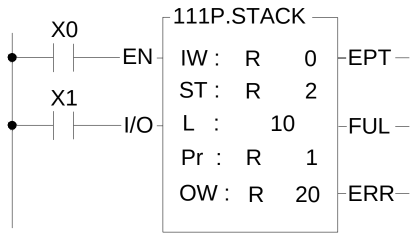
- Stack is the opposite of queue, being a last in first out (LIFO) device. This means that the data that was most recently pushed into the stack will be the first to be popped out of the stack. The stack is comprised of L consecutive 16 or 32-bit (D instruction) registers starting from ST, as shown in the following diagram:



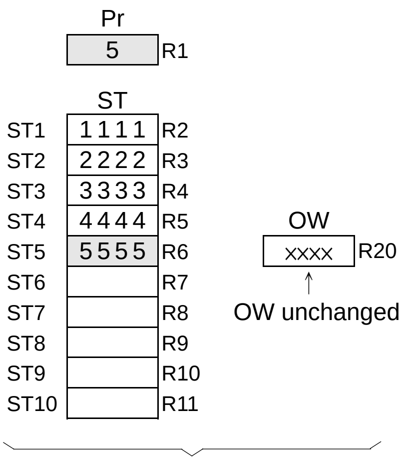
- When execution control "EN" = 1 or has a transition from 0 to 1(P instruction), the status of in/out control "I/O" determines whether the IW data will be pushed into the stack (when "I/O" = 1), or the data pointed by Pr within the stack (the data most recently pushed into the stack) will be moved out and transferred to OW (when "I/O" = 0). Note that the data pushed in is stacking, so before pushed in, Pr will increased by 1 to point to the top of the stack then the data will be pushed in. When it is popped out, the data pointed by pointer Pr (the most recently pushed in data) will be transferred to OW. After then Pr will decreased by 1. Under any circumstances, the pointer Pr will always point to the data that was pushed into the stack most recently.

FUN111 D P STACK	STACK	FUN111 D P STACK

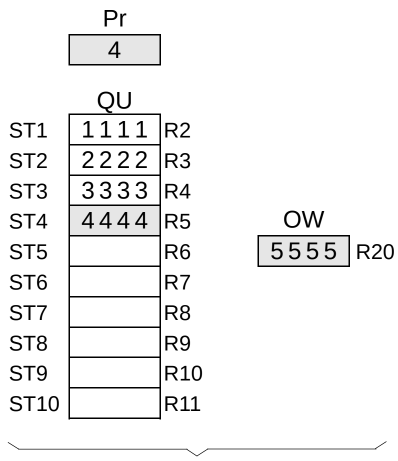
- When no data has yet been pushed into the stack or the pushed in data has already been popped out (Pr = 0), the stack empty flag "EPT" will set to 1. In this case any further pop up actions, will be ignored. If more data is pushed than popped out, sooner or latter the stack will be full (pointer Pr points to ST_L position), and the stack full flag "FUL" will set to 1. In this case any further push actions, will be ignored. As with queue, the stack pointer in normal case should not be changed by other instructions. If there is a special application which requires to set the Pr value, then its effective range is 0 to L (0 means empty, 1 to L respectively correspond to ST_1 to ST_L). Beyond this range, the pointer error flag "ERR" will set to 1, and the instruction will not be carried out.



- The program at left assumes that the initial content of the stack is just as in the diagram of a stack on the preceding page. The operation illustrated in this example is to push a data and than pop it from stack. The results are shown below. Under any circumstances, Pr always point to the data that was most recently pushed into the stack.

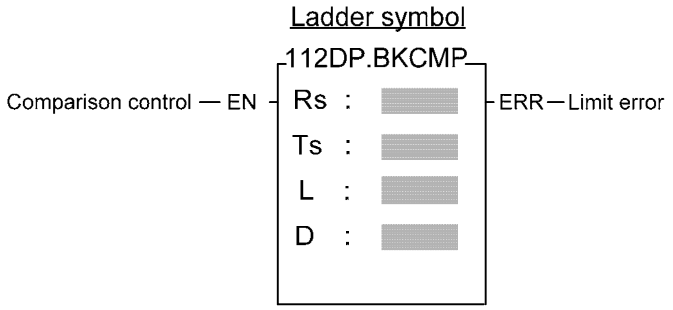


After push(X1=1，X0 from 0→1)



After pop up(X1=0，X0 from 0→1)

FUN112 D P BKCMP	BLOCK COMPARE（DRUM）	FUN112 D P BKCMP



Rs : Data for compare, can be a constant or a register

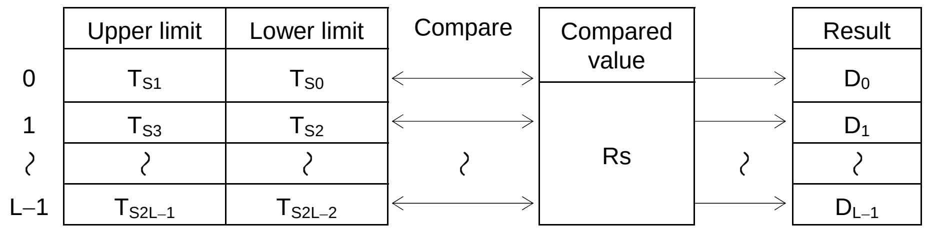
Ts : Starting register block storing upper and lower limit

L : Number of pairs of upper and lower limits

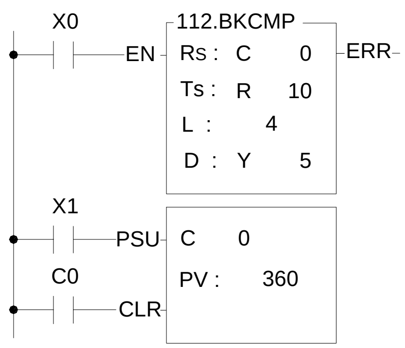
D : Starting relay storing results of comparison

Range / Operand	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	16/32-bit +/- number
Rs				○	○	○	○	○	○	○	○	○	○	○	○	○
Ts				○	○	○	○	○	○	○	○	○	○	○	○	
L										○				○*	○	1~256
D	○	○	○													

- When comparison control "EN" = 1 or has a transition from 0 to 1(P instruction), comparisons will be perform one by one between the contents of Rs and the upper and lower limits form by L pairs of 16 or 32-bit (D modifier) registers starting from the Ts register (starting from T0 each adjoining 2 register units form a pair of upper and lower limits). If the value of Rs falls within the range of the pair, then the bit within the comparison results relay D which corresponds to that pair will be set to 1. Otherwise it will be set as 0 until comparison of all the L pairs of upper and lower limits is completed.
- When M1975=0, if there is any pair where the upper limit value is less than the lower limit value, then the limit error flag "ERR" will be set to 1, and the comparison output for that pair will be 0.
- When M1975=1, there is no restriction on the relation of upper limit and lower limit, this can apply for 360°rotary electronic drum switch application.



- Actually this instruction is a drum switch, which can be used in interrupt program and when incorporate with immediate I/O instruction (IMDIO) can achieve an accurate electronic drum.

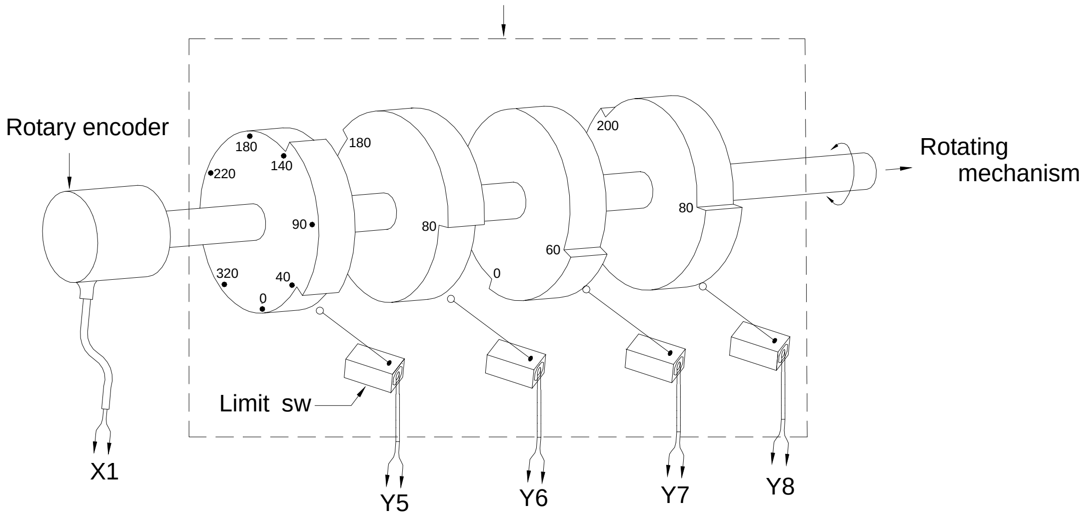


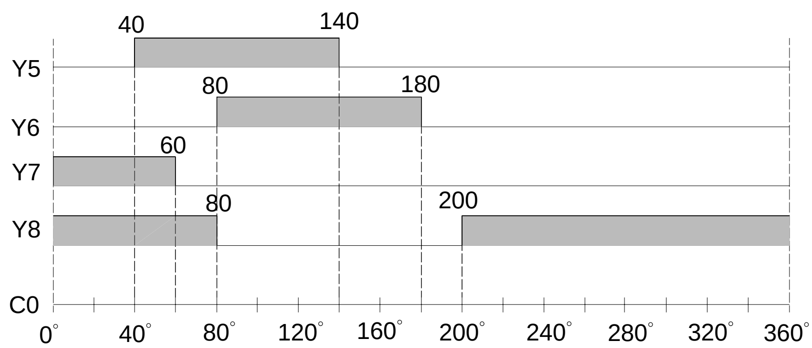
- In this program, C0 represents the rotation angle (Rs) of a drum shaft. The block compare instruction performs a comparison between Rs and the 4 pairs (L = 4) of upper and lower limits, R10,R11, R12,R13, R14,R15 and R16,R17. The comparison results can be obtained from the four drum output points Y5 to Y8.
- The input point X1 is a rotation angle detector mounted on the drum shaft. With each one degree rotation of the drum shaft angle, X1 produces a pulse. When the drum shaft rotates a full cycle, X1 produces 360 pulses.

FUN112 D P BKCMP	BLOCK COMPARE（DRUM）	FUN112 D P BKCMP

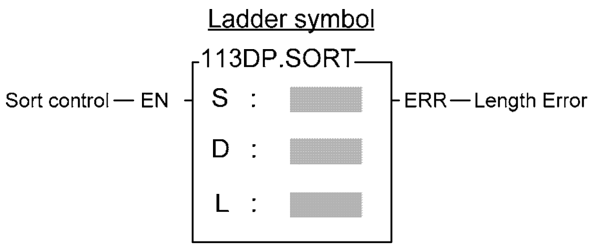
- The program in the diagram above coordinates a rotary encoder or other rotating angle detection device (directly connect to a rotating mechanism), which can form a mechanical device equivalent to the mechanical structure of an actual drum (see mechanism shown within dotted line in diagram below). While the upper and lower limits are being adjusted, you can change at will the range of the activated angle of the drum. This cannot be done with the traditional drum mechanism.

Equivalent mechanical drum emulated by above program





FUN113 D P SORT	DATA SORTING	FUN113 D P SORT



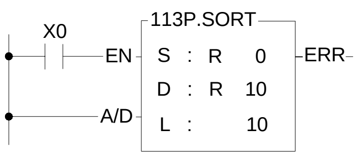
S : Starting register of source registers to sort

D : Starting register of destination registers to store the data after sorted

L : Total register for sorting

Range / Operand	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			○				○	○	○

- When sort control "EN" = 1 or has a transition from 0 to 1(P instruction), will sort the registers with ascending order (if A/D = 1) or descending order (if A/D = 0) and put the sorted result to the registers starting by D register.
- The valid data length of sort operation is between 2 and 127, other length will set the “ERR” to 1 and the sort operation will not perform.

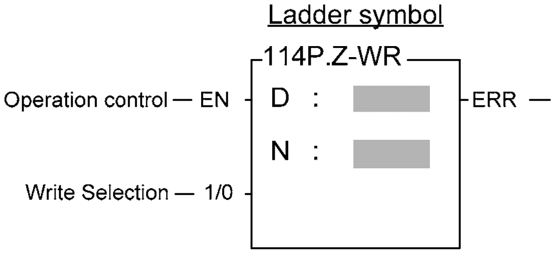


- The example at left sorts the table comprised of R0~R9 and stores the sorted data to the table locate at R10~R19.

S			D	
R0	1547		R10	0013
R1	2314		R11	1547
R2	7725		R12	1925
R3	0013	X0 = ↑	R13	2314
R4	5247	⇨	R14	2796
R5	1925		R15	5247
R6	6744		R16	5319
R7	5319		R17	6744
R8	9788		R18	7725
R9	2796		R19	9788

Before　　　　　After

FUN114DP Z-WR	ZONE WRITE	FUN114DP Z-WR



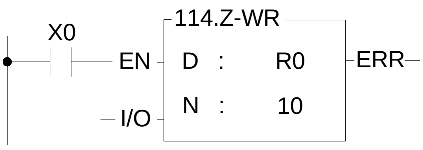
D : Starting address of being set or reset

N : Quantity of being set oe reset, 1~511

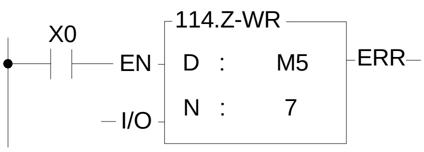
D、N operand can combine V、Z、P0~P9 for index addressing while word operation

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	○

- When operation control "EN"=1 or changes from 0→1（P instruction）, it will perform the write operation according to the input status of write selection, the specified area of registers or bits will all be reset to 0 ("1/0"=0) or set to 1("1/0"=1).



• Above example, registers R0~R9 will be reset to 0 while X0=1.

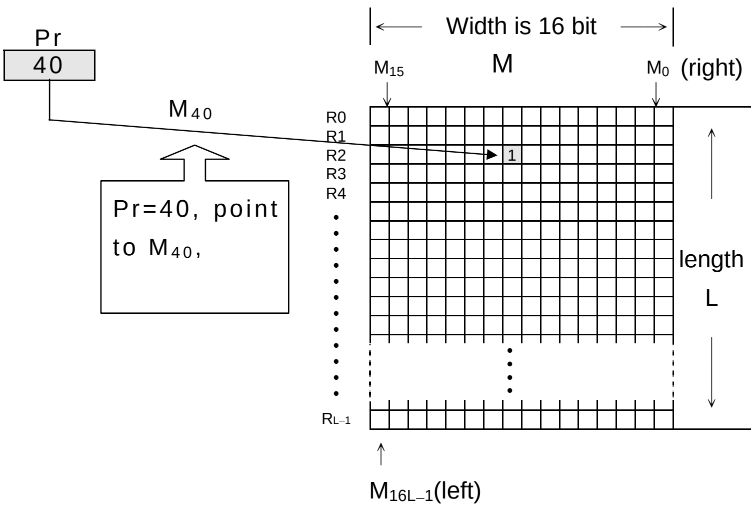


• Above example, bits M5~M11 will be reset to 0 while X0=1.

Matrix Instructions

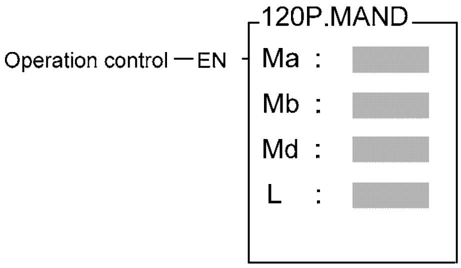
Fun No.	Mnemonic	Functionality	Fun No.	Mnemonic	Functionality
120	MAND	Matrix AND	126	MBRD	Matrix Bit Read
121	MOR	Matrix OR	127	MBWR	Matrix Bit Write
122	MXOR	Matrix XOR	128	MBSHF	Matrix Bit Shift
123	MXNR	Matrix XNOR	129	MBROT	Matrix Bit Rotate
124	MINV	Matrix Inverse	130	MBCNT	Matrix Bit Count
125	MCMP	Matrix Compare			

- A matrix is comprised of 2 or more consecutive 16-bit registers. The number of registers comprising the matrix is called the matrix length (L). One matrix altogether has L×16 bits (points), and the basic unit of the object for each operation is bit.
- The matrix instructions treats the 16×L matrix bits as a set of series points(denoted by M_0 to M_{16L-1}). Whether the matrix is formed by register or not, the operation object is the bit not numerical value.
- Matrix instructions are used mostly for discrete status processing such as moving, copying, comparing, searching, etc, of single point to multipoint (matrix), or multipoint-to-multipoint. These instructions are convenient, important for application.
- Among the matrix instructions, most instruction need to use a 16-bit register as a pointer to points a specific point within the matrix. This register is known as the matrix pointer (Pr). Its effective range is 0 to 16L-1, which corresponds respectively to the bits M_0 to M_{16L-1} within the matrix.
- Among the matrix operations, there are shift left/right, rotate left/right operations. We define the movement toward higher bit is left direction, while the movement toward lower bit is right direction, as shown in the diagram below.



FUN120 P MAND	MATRIX AND	FUN120 P MAND

Ladder symbol



Ma: Starting register of source matrix a

Mb: Starting register of source matrix b

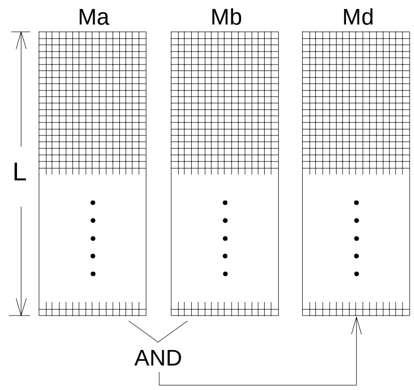
Md : Starting register of destination matrix

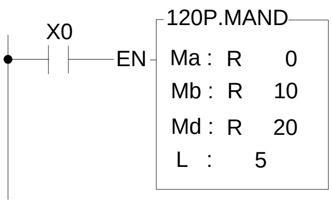
L : Length of matrix (Ma, Mb and Md)

Ma, Mb, Md may combine with V, Z, P0~P9 to serve indirect address application

Range / Operand	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							○				○*	○	○	

- When operation control "EN" = 1 or has a transition from 0 to 1(P instruction), this instruction will perform a logic AND (only if 2 bits are 1 will the result be 1, otherwise it will be 0)operation between two source matrixes with a length of L, Ma and Mb. The result will then be stored in the destination matrix Md, which is also the same length (the AND operation is done by bits with the same bit numbers). For example, if Ma_0 = 0, Mb_0 = 1, then Md_0 = 0; if Ma_1 = 1, Mb_1 = 1, then Md_1 = 1; etc, right up until AND reaches Ma_{16L-1} and Mb_{16L-1}.





- In the program at left, when X0 goes from 0→1, then matrix Ma, comprised by R0 to R4, and matrix Mb, comprised by R10 to R14, will do an AND operation. The results will be stored back in matrix Md, comprised by R20 to R24. The result is shown at right in the diagram below.

Ma (Ma_{15} … Ma_0 / Ma_{79} … Ma_{64})

R0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0
R2	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
R3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R4	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Mb (Mb_{15} … Mb_0 / Mb_{79} … Mb_{64})

R10	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
R11	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
R12	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
R13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R14	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Before execution

Md (Md_{15} … Md_0 / Md_{79} … Md_{64})

R20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R21	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R22	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
R23	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R24	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

After execution

FUN121 P MOR	MATRIX OR	FUN121 P MOR

Ladder symbol

121P.MOR

Operation control — EN — Ma :
Mb :
Md :
L :

Ma : Starting register of source matrix a

Mb : Starting register of source matrix b

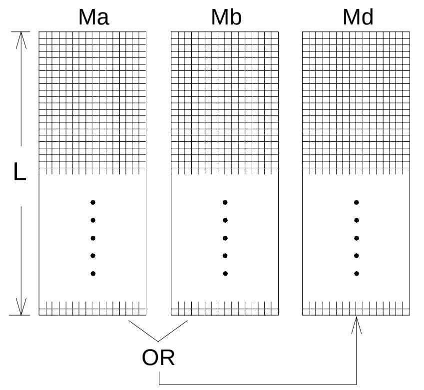
Md : Starting register of destination matrix

L : Length of matrix (Ma, Mb and Md)

Ma, Mb, Md may combine with V, Z, P0~P9 to serve indirect address application

Range / Ope-rand	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							○				○*	○	○	

- When operation control "EN" = 1 or has a transition from 0 to 1 (P instruction), this instruction will perform a logic OR(If any 2 of the bits are 1, then the result will be 1, and only if both are 0 will the result be 0) operation between 2 source matrixes with a length of L, Ma and Mb. The result will then be stored in the destination matrix Md, which is also the same length (the OR operation is done by bits with the same bit numbers). For example, if Ma_0 = 0, Mb_0 = 1, then Md_0 = 1; if Ma_1 = 0, Mb_1 = 0, then Md_1 = 0; etc, right up until OR reaches Ma_{16L-1} and Mb_{16L-1}.



121P.MOR

X0 — EN — Ma : R 0
Mb : R 10
Md : R 10
L : 5

- In the program at left, when X0 goes from 0→1, then matrix Ma, comprised by R0 to R4, and matrix Mb, comprised by R10 to R14, will do an OR operation. The results will then be stored into the destination matrix Md, comprised by R10 to R14. In this example, Mb and Md is the same matrix, so after operation the source matrix Mb will replaced by the new value. The result is shown at right in the diagram below.

Before execution

Ma (Ma_{15} … Ma_0 ; Ma_{79} … Ma_{64})

R0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0
R2	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
R3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R4	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Mb (Mb_{15} … Mb_0 ; Mb_{79} … Mb_{64})

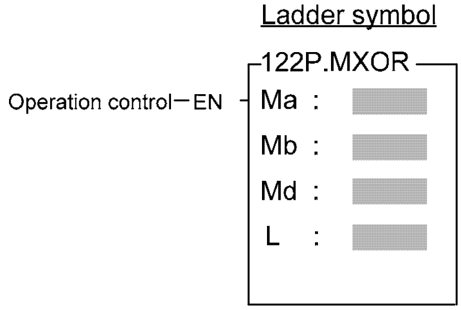
R10	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
R11	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
R12	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
R13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R14	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

After execution

Md (Md_{15} … Md_0 ; Md_{79} … Md_{64})

R20	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
R21	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
R22	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
R23	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R24	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

FUN122 P MXOR	MATRIX EXCLUSIVE OR（XOR）	FUN122 P MXOR



Ma: Starting register of source matrix a

Mb: Starting register of source matrix b

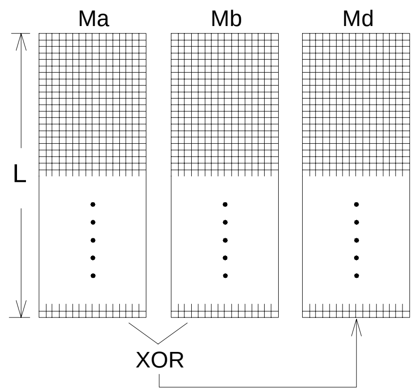
Md: Starting register of destination matrix

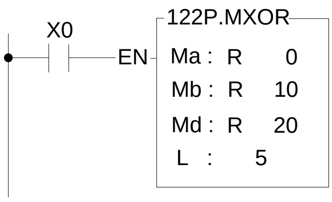
L : Length of matrix (Ma, Mb and Md)

Ma, Mb, Md may combine with V, Z, P0~P9 to serve indirect address application

Range / Operand	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							○				○*	○	○	

- When operation control "EN" = 1 or has a transition from 0 to 1 (P instruction), this instruction will performs a logic XOR (if the 2 bits are different, then the result will be 1, otherwise it will be 0)between 2 source matrixes with a length of L, Ma and Mb. The result will then be stored back into the destination matrix Md, which also has a length of L. For example the XOR operation is done by bits with the same bit numbers - for example, if Ma_0 = 0, Mb_0 = 1, then Md_0 = 1; if Ma_1 = 1, Mb_1 = 1, then Md_1 = 0; etc, right up until XOR reaches Ma_{16L-1} and Mb_{16L-1}.





- In the program at left, when X0 goes from 0→1, will perform a XOR operation between matrix Ma, comprised by R0 to R4, and matrix Mb, comprised by R10 to R14. The results will then be stored in destination matrix Md, comprised by R20 to R24. The results are shown at right in the diagram below.

Ma (Ma_{15} … Ma_0 ; Ma_{79} … Ma_{64})

R0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0
R2	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
R3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R4	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Mb (Mb_{15} … Mb_0 ; Mb_{79} … Mb_{64})

R10	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
R11	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
R12	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
R13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R14	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Before execution

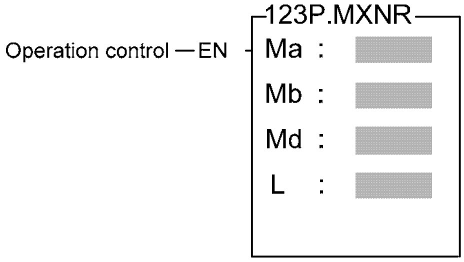
Md (Md_{15} … Md_0 ; Md_{79} … Md_{64})

R20	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
R21	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
R22	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R23	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R24	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

After execution

FUN123 P MXNR	MATRIX EXCLUSIVE NOR（XNR）	FUN123 P MXNR

Ladder symbol



Ma : Starting register of source matrix a

Mb : Starting register of source matrix b

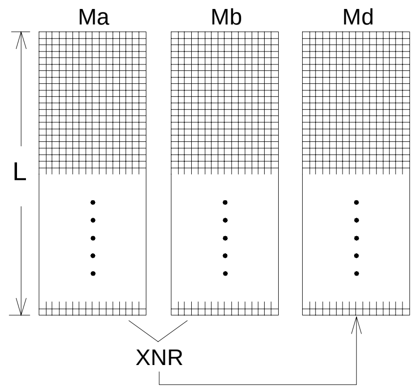
Md : Starting register of destination matrix

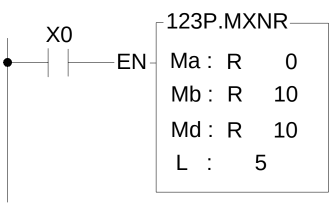
L : Length of matrix (Ma, Mb and Md)

Ma, Mb, Md may combine with V, Z,P0~P9 to serve indirect address application

Range / Ope-rand	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							○				○*	○	○	

- When operation control "EN" = 1 or has a transition from 0 to 1 (P instruction), will perform a logic XNR operation (if the 2 bits are the same, then the result will be 1, otherwise it will be 0)between 2 source matrixes with a length of L, Ma and Mb. The results will then be stored into the destination matrix Md, which also has the same length (the XNR operation is done by bits with the same bit numbers). For example, if Ma_0 = 0, Mb_0 = 1, then Md_0 = 0; Ma_1 = 0, Mb_1 = 0, then Md_1 = 1; etc, right up until XNR reaches Ma_{16L-1} and Mb_{16L-1}.





- When operation control "EN" = 1 or goes from 0 to 1 (P instruction), will perform a XNR operation between Ma matrix comprised by R0~R9 and Mb matrix comprised by R10~R19. The results will then be stored into the destination matrix Md comprised by R10~R19. The results are shown at right in the diagram below.

Ma (Ma_{15} … Ma_0 ; Ma_{79} … Ma_{64})

R0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0
R2	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
R3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R4	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Mb (Mb_{15} … Mb_0 ; Mb_{79} … Mb_{64})

R10	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
R11	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
R12	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
R13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R14	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Before execution

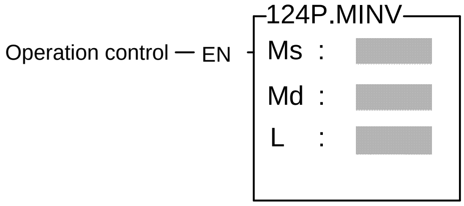
Md (Md_{15} … Md_0 ; Md_{79} … Md_{64})

R20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R21	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R22	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
R23	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
R24	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

After execution

FUN124 P MINV	MATRIX INVERSE	FUN124 P MINV

<u>Ladder symbol</u>



Ms : Starting register of source matrix

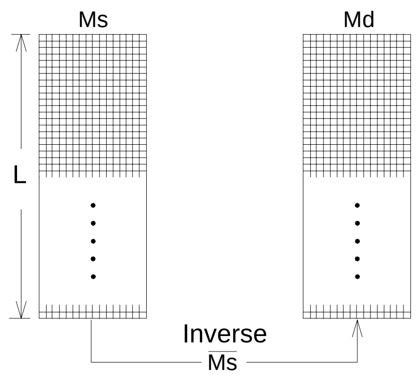
Md : Starting register of destination

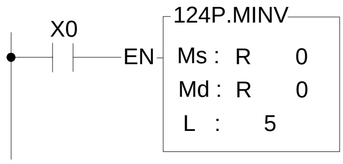
L : Length of matrix (Ms and Md)

Ma, Md may combine with V, Z, P0~P9 to serve indirect address application

Range / Operand	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							○				○*	○	○	

- When operation control "EN" = 1 or has a transition from 0 to 1 (P instruction), source register Ms, which has a length of L, will be completely inverted (all the bits with a value of 1 will change to 0, and all those with a value of 0 will change to 1). The results will then be stored into destination matrix Md.





- In the program at left, when X0 goes from 0→1, the matrix comprised by R0 to R4 will be inverted, and then store back into itself (because in this example Ms and Md are the same matrix). The results obtained are shown at right in the diagram below.

Ms (Ms_{15} … Ms_{0}; bottom row Ms_{79} … Ms_{64})

R0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0
R2	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
R3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R4	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Before execution

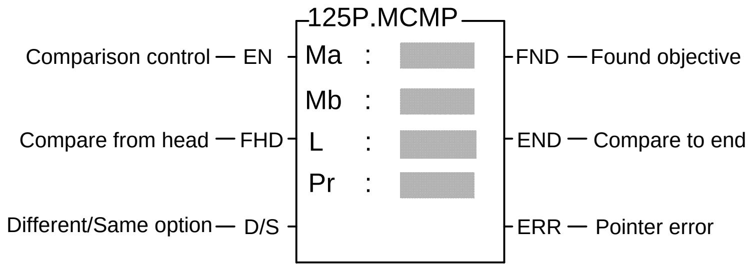
Md (Md_{15} … Md_{0}; bottom row Md_{79} … Md_{64})

R0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
R1	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
R2	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0
R3	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
R4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

After execution

FUN125 P MCMP	MATRIX COMPARE	FUN125 P MCMP

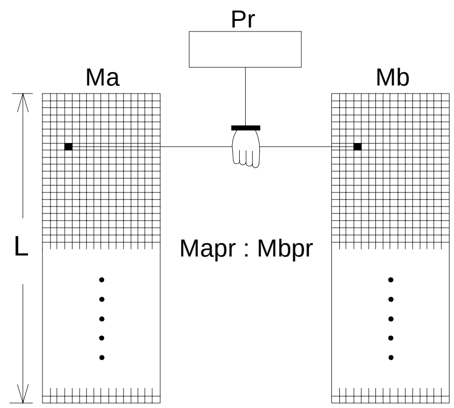
Ladder symbol



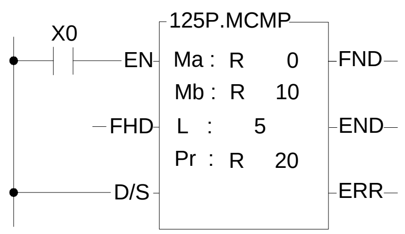
Md: Starting register of matrix a
Mb: Starting register of matrix b
L : Length of matrix (Ma, Mb)
Pr : Pointer register
Ma, Mb may combine with V, Z, P0~P9 to serve indirect address application

Range / Ope-rand	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		○	○	○	○	○	○		○	○*	○*	○		

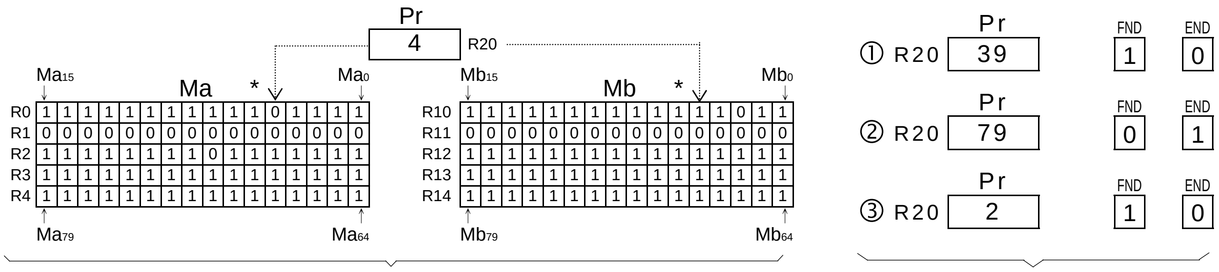
- When comparison control "EN" = 1 or has a transition from 0 to 1 (P instruction), then beginning from the top pair of bits (Ma_0 and Mb_0) within the 2 matrixes Ma and Mb (when "FHD" = 1 or Pr value is equal to 16L-1), or beginning from the next pair of bits (Mapr + 1 and Mbpr + 1) pointed by pointer Pr (when "FHD" = 0 and Pr value is less than L-1), this instruction will compare and search for pairs of bits with different value (when D/S = 1) or the same value (when D/S = 0). Once match found, pointer Pr will point to the bit number in the matrix met the search condition. The found objective flag "FND" will be set to 1. When it has searched to the final pair of bits in the matrix (Ma_{16L-1}, Mb_{16L-1}), this execution of the instruction will finish, no matter it has found or not. If this happen then The compare-to-end flag "END" will be set as 1, and the Pr value will set to 16L-1 and the next time that this instruction is executed, Pr will automatically return to the starting point of the matrix (Pr = 0) to begin the comparison search.



- The range for the pointer value is 0 to 16L-1. The Pr value should not be changed by other instructions, as this will affect the result of search. If the Pr value exceeds its range, then the pointer error flag "ERR" will be set to 1, and this instruction will not be carried out.



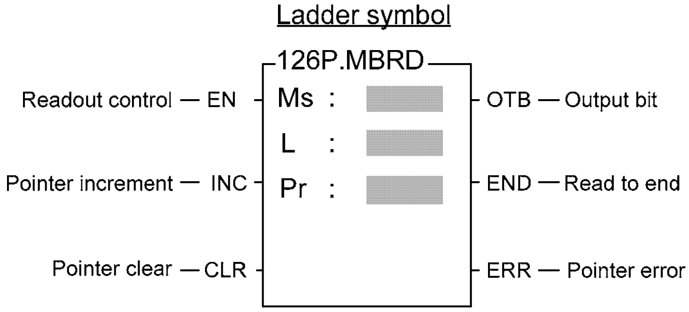
- In the program at left, the "FHD" input is 0, so starting from a position 1 greater than the pointer value at that time (marked by *), the instruction will do a search for bits with different status (because D/S = 1). When X0 has a transition from 0→1 three times, the results are shown at right in the diagram below.



Before execution   Execution result

FUN126 P MBRD	MATRIX BIT READ	FUN126 P MBRD

Ladder symbol



Ms : Starting register of matrix

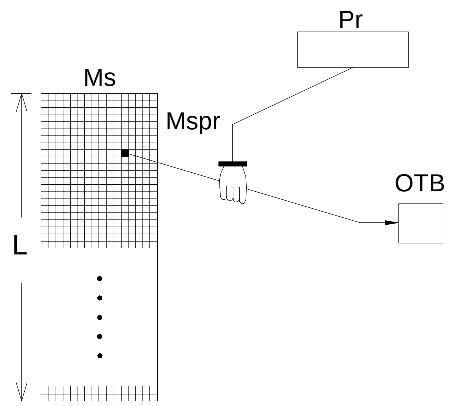
L : Matrix length

Pr : Pointer register

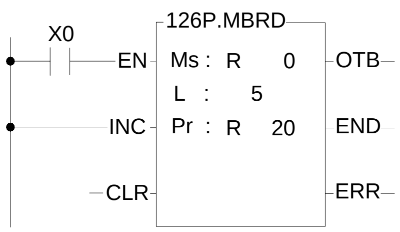
Ms may combine with V, Z, P0~P9 to serve indirect address application

Range / Operand	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		○	○	○	○	○	○		○	○*	○*	○		

- When readout control "EN" = 1 or has a transition from 0 to 1 (P instruction), the status of the bit Ms_{pr} pointed by pointer Pr within matrix Ms will be read out and appear at the output bit "OTB". Before the readout, this instruction will first check the input -pointer clear "CLR". If "CLR" is 1, then the Pr value will be cleared to 0 first before the readout action is carried out. After the readout is completed, If the Pr value has already reached 16L-1 (the final bit), then the read-to-end flag "END" will be set to 1. If Pr is less than 16L-1, then the status of pointer increment "INC" will be checked. If "INC" is 1, then Pr will be increased by 1. Besides this, pointer clear "CLR" can execute independently, and is not affected by other input.



- The effective range of the pointer is 0 to 16L-1. Beyond this range the pointer error flag "ERR" will be set to 1, and this instruction will not be carried out.



- In the program at left, INC = 1, so every time there is one readout the pointer will be increased by 1. With this way each bit in Ms may be read out successively, as shown at left in the diagram below. When X0 goes 3 times from 0→1, the results are shown at right in the diagram below .

Ms (Ms_{15} … Ms_0)

R0	0	0	0	0	0	1	1	1	1	1	0	0	0	0	0	1
R1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
R2	0	0	0	1	1	1	0	0	0	1	1	1	0	0	0	1
R3	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
R4	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0

Ms_{79} Ms_{77} … Ms_{64}

Pr: R20 = 77

OTB: 0

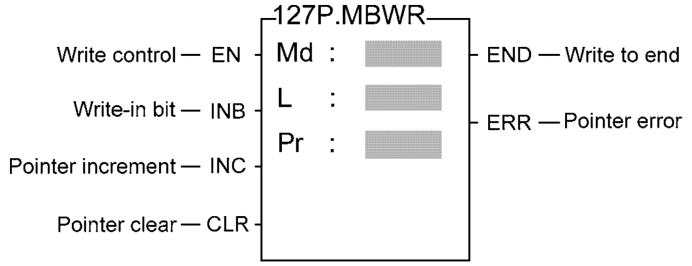
Before execution

	Pr	OTB	END
① R20	78	1	0
② R20	79	0	0
③ R20	79	1	1

Execution result

FUN127 P MBWR	MATRIX BIT WRITE	FUN127 P MBWR

Ladder symbol



Md : Starting register of matrix

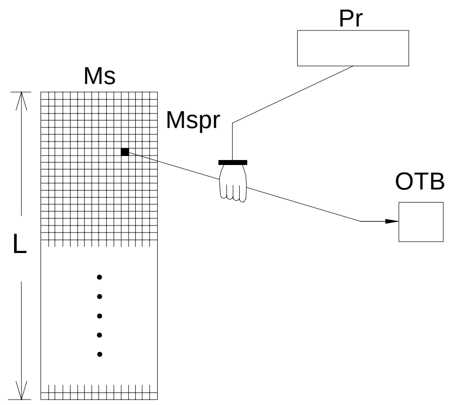
L : Matrix length

Pr : Pointer register

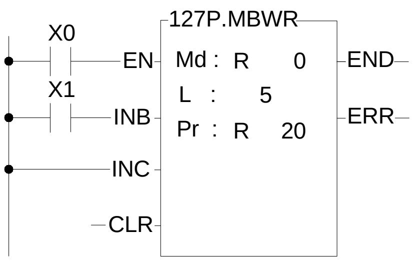
Md may combine with V, Z, P0~P9 to serve indirect address application

Range / Operand	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	○	○	○	○	○	○	○	○*	○*	○		

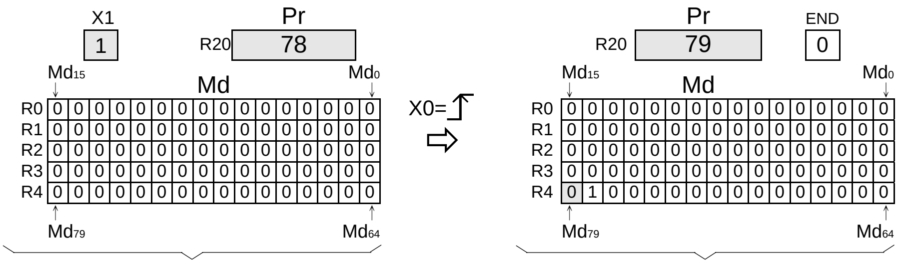
- When write control "EN" = 1 or has a transition from 0 to 1 (P instruction), the status of the write-in bit "INB" will be written into the bit Md_{pr} pointed by pointer Pr within matrix Md. Before the write-in takes place, the status of pointer clear "CLR" will be checked. If "CLR" is 1, then Pr will be cleared to 0 before the write-in action. After the write-in action has been completed, the Pr value will be checked again. If the Pr value has already reached 16L-1 (last bit), then the write-to-end flag will be set to 1. If the Pr value is less than 16L-1 and "INC" is 1, then the pointer will increased by 1. Besides this, pointer clear "CLR" can execute independently, and is not affected by other input.



- The effective range of Pr is 0 to 16L-1. Beyond this range, the pointer error flag "ERR" will be set to 1, and this instruction will not be carried out.



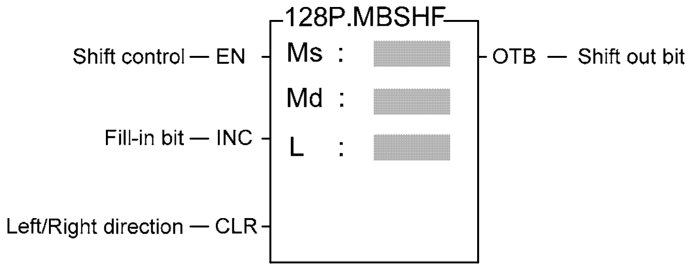
- In the program at left, pointer will be increased each time execution (because "INC" is 1). As shown in the diagram below, when X0 has a transition from 0→1, the status of INB (X1) will be written into the Md_{pr} (Md_{78}) position, and pointer Pr will increased by 1 (changing to 79). In this case, although Pr is pointing to the end, it has not yet been written into Md_{79}, so "END" flag is still 0. Only the next attempt to write to Md_{79} will set "END" to 1.



Before execution After execution

FUN128 P MBSHF	MATRIX BIT SHIFT	FUN128 P MBSHF

<u>Ladder symbol</u>



Ms : Starting register of source matrix

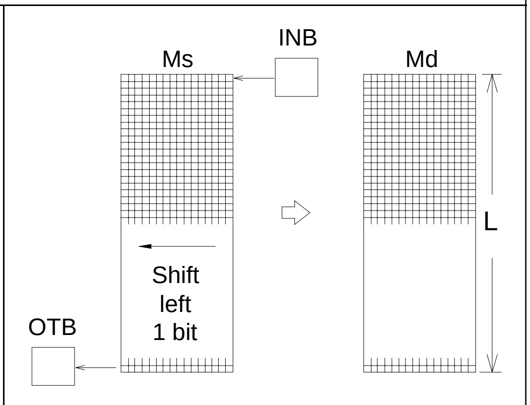
Md : Starting register of destination matrix

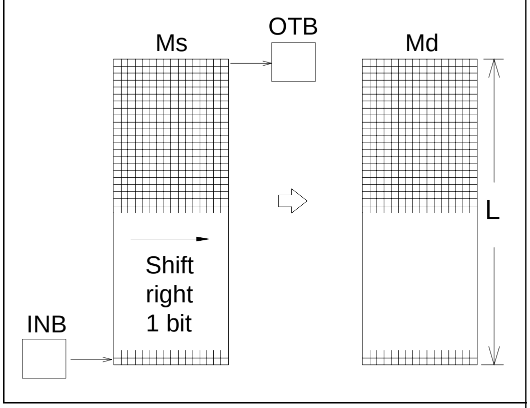
L : Length of matrix (Ms and Md)

Ms, Md may combine with V, Z, P0~P9 to serve indirect address application

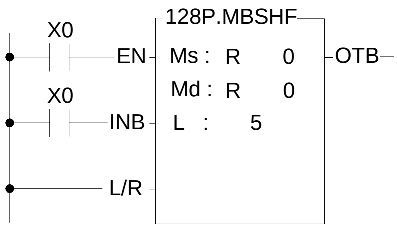
Ope-rand \ Range	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR
	WX0 ∣ WX240	WY0 ∣ WY240	WM0 ∣ WM1896	WS0 ∣ WS984	T0 ∣ T255	C0 ∣ C255	R0 ∣ R3839	R3840 ∣ R3903	R3904 ∣ R3967	R3968 ∣ R4167	R5000 ∣ R8071	D0 ∣ D4095	2 ∣ 256	V、Z P0~P9
Ms	○	○	○	○	○	○	○	○	○	○	○	○		○
Md		○	○	○	○	○	○		○	○*	○*	○		○
L											○*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

- When shift control "EN" = 1 or has a transition from 0 to 1 (P instruction), source matrix Ms will be retrieved and completely shifted one position to the left (when L/R = 1) or one position to the right (when L/R = 0). The space caused by the shift (with a left shift it will be M_0, and with a right shift it will be M_{16L-1}), is replaced by the status of fill-in bit "INB". The status of the bits popped out (with a left shift it will be M_{16L-1}, and with a right shift it will be M_0) will appear at the output bit "OTB". Then the results of this shifted matrix will be filled into the destination matrix Md.





- The program at left is an example where Ms and Md are the same matrix. When X0 goes from 0→1, Ms will be completely retrieved and moved to the left (because L/R = 1) by 1 bit. It will then be stored back to Md, and the results are shown at right in the diagram below.



Ms (Ms_{15} … Ms_0; Ms_{79} … Ms_{64})

R0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
R2	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0
R3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R4	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Before execution

X0=↑ ⇨

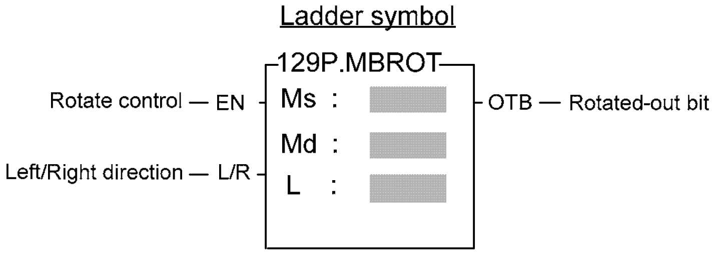
Md (Md_{15} … Md_0; Md_{79} … Md_{64})

R0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
R1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0
R2	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	1
R3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
R4	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0

After execution

FUN129 P MBROT	MATRIX BIT ROTATE	FUN129 P MBROT

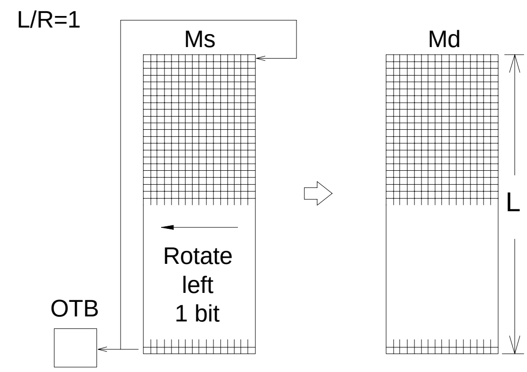
Ladder symbol

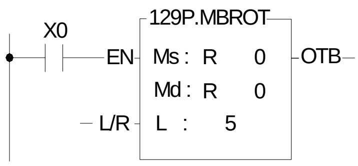


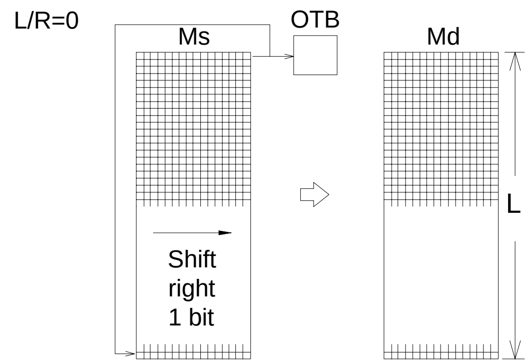
Ms : Starting register of source matrix
Md : Starting register of destination matrix
L : Length of matrix (Ms and Md)
Ms, Md may combine with V, Z, P0~P9 to serve indirect address application

Range / Operand	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											○*	○	○	

- When rotate control "EN" = 1 or has a transition from 0 to 1 (P instruction), matrix Ms will be completely retrieved and rotated by one bit towards the left (when L/R = 1) or to the right (when L/R = 0). The space created by the rotation (with a left rotation it will be M0, and with a right rotation it will be M_{16L-1}) will be replaced by the status of the rotated-out bit (with a left rotation it will be M_{16L-1}, and with a right rotation it will be M0). The rotated-out bit will not only be used to fill the above-mentioned space, it will also be transferred to rotated-out bit "OTB".







- In the program at left, Ms and Md are the same matrix. When X0 goes from 0→1, then the whole of Ms is retrieved and rotated right (because L/R = 0) by 1 bit. It is then stored back into Ms itself (because in this example Ms and Md are the same matrix). The results are shown at right in the diagram below.

Ms (Ms_{15} … Ms_0 top row; Ms_{79} … Ms_{64} bottom row)

R0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
R2	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0
R3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R4	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Before execution

X0=↑ ⇨

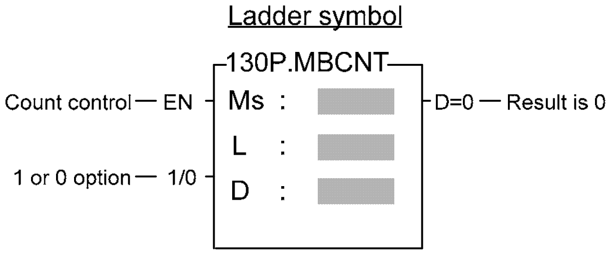
Md (Md_{15} … Md_0 top row; Md_{79} … Md_{64} bottom row)

R0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
R2	0	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0
R3	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R4	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

After execution

OTB: 0

FUN130 P MBCNT	MATRIX BIT STATUS COUNT	FUN130 P MBCNT



Ms : Starting register of matrix

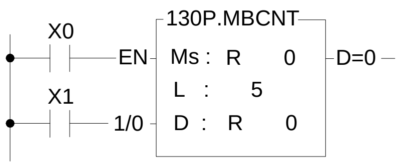
L : Matrix length

D : Register storing count results

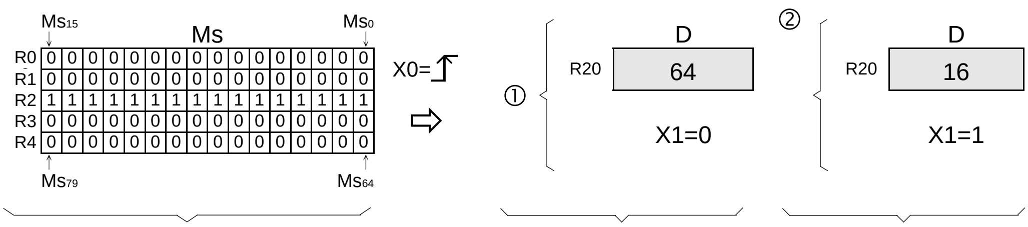
Ms may combine with V, Z, P0~P9 to serve indirect address application

Range / Operand	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		○	○	○	○	○	○		○	○*	○*	○		

- When count control "EN" = 1 or has a transition from 0 to 1(P instruction), then among the 16L bits of the Ms matrix, this instruction will count the total amount of bits with a status of 1 (when input "1/0" = 1) or the total amount of bits with a status of 0 (when input "1/0" = 0). The results of the counting will be stored into the register specified by D. If the value of these amounts is 0, then the Result-is-0 flag "D = 0" will be set to 1.



- The program at left sets X1 first as 0 (to count bits with status of 0) and then as 1 (to count bits with status of 1) and let the signal X0 has a transition from 0→1 for both case, the execution results are shown at right in the diagram below .



Source matrix　　Count of ‘0’ bit　　Count of ‘1’ bit

FUN 139 HSPWM	HIGH SPEED PULSE WIDTH MODULATION OUTPUT	FUN 139 HSPWM

Ladder symbol

```
                        139.HSPWM
Operation control — EN-  Pw  :        - ACT —
                         Op  :
                         Rs  :
                         Pn  :
                         OR  :
                         WR  :
```

PW : PWM output (0 = Y0、1 = Y2、2 = Y4、3 = Y6)

Op : Output polarity ; 0 = Normal
1 = Inverse of output

Rs : Resolution ; 0 = 1/100 (1%)
1 = 1/1000 (0.1%)

Pn : Setting of output frequency(0~255)

OR : Setting register of output pulse width (0~100 or 0~1000)

WR : Working register

Range / Operand	Y	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K
	Yn of main unit	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			○	○	○	○	○	○		○	○	○	○	

Description

- The setting of resolution(RS) must be same between output0(Y0) and output1(Y2) also the setting of output frequency(Pn). It means both output0 and output1 have the same output frequency and the same output resolution, only the pulse width can be different. Same principle for output2(Y4) and output3(Y6).
- When operation control "EN" = 1, the specified digital output will perform the PWM output, the expression for output frequency as shown bellow:

1. $f_{pwm} = \dfrac{184320}{(P_n + 1)}$ while Rs(Resolution)=1/100

2. $f_{pwm} = \dfrac{18432}{(P_n + 1)}$ while Rs(Resolution)=1/1000

Example 1 : If Pn (Setting of output frequency) = 50, Rs = 0(1/100), then

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

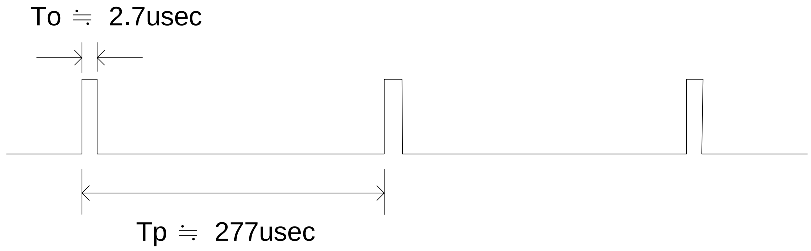
$$\text{T(Period)} = \frac{1}{f_{pwm}} \fallingdotseq 277\text{uS}$$

For Rs = 1/100, if OR(Setting of output pulse width) = 1, then T0 ≒ 2.7uS; if OR(Setting of output pulse width) = 50, then To ≒ 140uS.

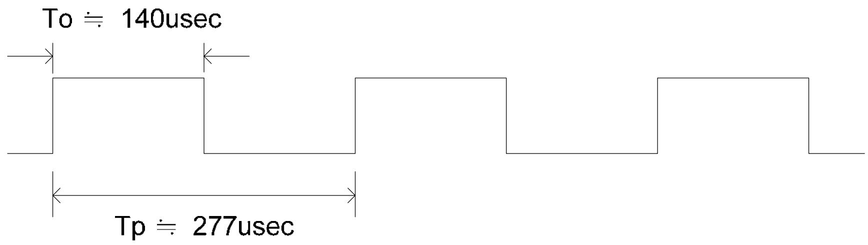
.Output waveform :

(1).Pn (Output frequency) = 50, Rs = 0 (1/100), OR (Output pulse width) = 1 :

FUN 139 HSPWM	HIGH SPEED PULSE WIDTH MODULATION OUTPUT	FUN 139 HSPWM



(2).Pn (Output frequency) = 50, Rs = 0 (1/100), OR (Output pulse width) = 50 :



Example 2 : If Pn (Setting of output frequency) = 200, Rs = 1(1/1000), then

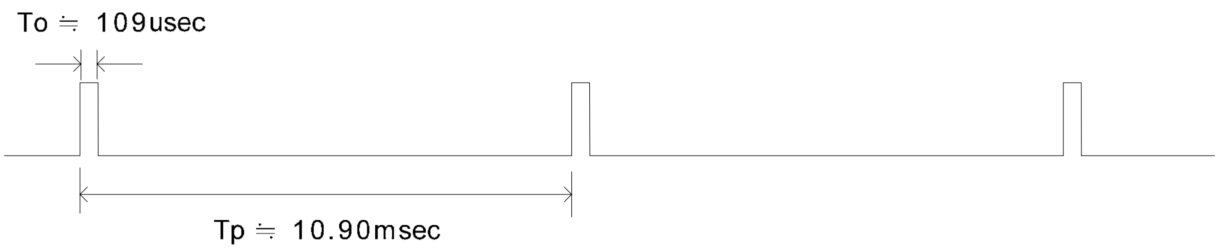
$$f_{pwm} = \frac{18432}{(200+1)} \fallingdotseq 91.7\text{Hz}$$

$$T(\text{Period}) = \frac{1}{f_{pwm}} \fallingdotseq 10.9\text{mS}$$

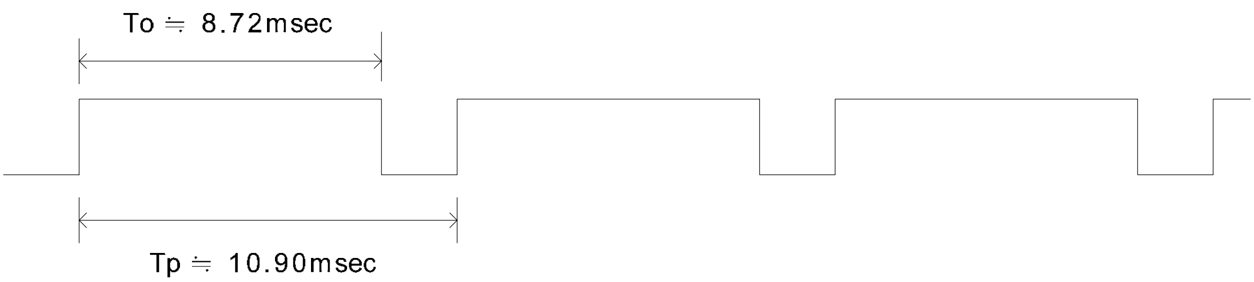
For Rs = 1/1000, if OR(Setting of output pulse width) = 10, then T0 ≒ 109uS; if OR(Setting of output pulse width) = 800, then To ≒ 8.72mS

.Output waveform :

(1).Pn (Output frequency) = 200, Rs = 1 (1/1000), OR (Output pulse width) = 10 :

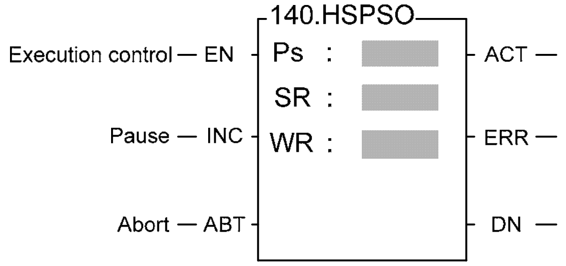


(2).Pn (Output frequency) = 200, Rs = 1 (1/1000), OR (Output pulse width) = 800 :



FUN140 HSPSO	HIGH SPEED PULSE OUTPUT INSTRUCTION (Brief description on function)	FUN140 HSPSO

Ladder symbol



Ps : The Pulse Output (0～3) selection
0:Y0 & Y1
1:Y2 & Y3
2:Y4 & Y5
3:Y6 & Y7
SR : Positioning program starting register.
WR : Starting working register of instruction operation, total 7 registers, can not used in any other part of program.

Range / Ope-rand	HR R0 ∣ R3839	DR D0 ∣ D4095	ROR R5000 ∣ R8071	K 2 ∣ 256
Ps				0～3
SR	○	○	○	
WR	○	○	○*	

Command descriptions

- The NC positioning program of HSPSO (FUN140) instruction is a program written and edited with text. The executing unit of program is divided by step (which includes output frequency, traveling distance, and transferring conditions). For one FUN140 instruction, can program 250 steps of positioning points at the most. Each step of positioning program requires 9 registers. For detailed application, please refer to chapter 13 “the NC positioning control of FBs-PLC”.
- The benefits of storing the positioning program in the register is that, while in application which use the MMI (man machine interface) as the operation console can save the positioning programs to MMI. Whenever the change of the positioning programs is requested, the download of positioning program can be simply done by a series of write register commands.
- The NC positioning of this instruction doesn’t provide the linear interpolation function.
- When execution control “EN”=1, if Ps0～3 is not controlled by other FUN140 instruction (the status of Ps0=M1992, Ps1=M1993, Ps2=M1994, and Ps3=M1995 is ON respectively), it will start to execute from the next step of positioning point (when goes to the last step, it will be restarted from the first step); if Ps0～3 is controlled by other FUN140 instruction (the status of Ps0=M1992, Ps1=M1993, Ps2=M1994, and Ps3=M1995 are OFF), this instruction will wait and acquires the control right of output point immediately right after other FUN140 release the output.
- When execution control input “EN” =0, it stops the pulse output immediately.
- When output pause “PAU” =1 and execution control was 1, it will pause the pulse output. When output pause “PAU” =0 and execution control is still 1, it will continue the unfinished pulse output.
- When output abort “ABT”=1, it will halt and stop pulse output immediately. (When the execution control input “EN” becomes 1 next time, it will restart from the first step of positioning point to execute.)
- While send the output pulse, the output indication “ACT” is ON.
- When there is an execution error, the output indication “ERR” will be ON. (The error code is stored in the error code register.)
- When the execution of each step of positioning program is completed, the output indication “DN” will be ON.

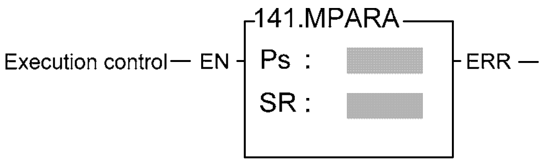
*** The working mode of Pulse Output must be configured (without setting, Y0～Y7 will be treated as normal output) to any one of following modes, before the HSPSO instruction can be worked.

U/D Mode: Y0 (Y2, Y4, Y6), as up pulse.
Y1 (Y3, Y5, Y7), as down pulse.
K/R Mode: Y0 (Y2, Y4, Y6), as the pulse out..
Y1 (Y3, Y5, Y7), as the direction.
A/B Mode: Y0 (Y2, Y4, Y6), as A phase pulse.
Y1 (Y3, Y5, Y7), as B phase pulse.

- The output polarity for Pulse Output can select to be Normally ON or Normally OFF.
- The working mode of Pulse Output can be configured by WINPROLADDER in “Output Setup” setting page.

FUN141 MPARA	NC POSITIONING PARAMETER VALUE SETTING (Brief description on function)	FUN141 MPARA

Ladder symbol



Ps : The pulse output (0～3) selection

SR : Starting register for parameter table; it has 18 parameters totally, and occupy 24 registers.

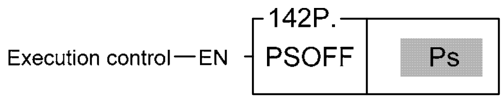
Range / Ope-rand	HR	DR	ROR	K
	R0 ｜ R3839	D0 ｜ D4095	R5000 ｜ R8071	2 ｜ 256
Ps				0～3
SR	○	○	○	

Operation descriptions

- It is not necessary to use this instruction. if the system default for parameter values is matching what user demanded, then this instruction is not needed. However, if it needs to change the parameter value dynamically, this instruction is required.
- This instruction incorporates with FUN140 or FUN147 for positioning control purpose.
- Whether the execution control input “EN” = 0 or 1, this instruction will be performed.
- When there are any errors in parameter value, the output indication “ERR” will be ON. (The error code is stored in the error code register.)
- For detailed functional description and usage, please refer to Chapter 11 “The NC positioning control of FBs-PLC” for explanation.

FUN142 P PSOFF	STOP THE HSPSO PULSE OUTPUT (Brief description on function)	FUN142 P PSOFF

Ladder symbol



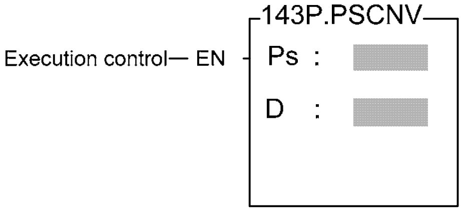
Ps : 0～3
Enforce the Pulse Output PSOn (n= Ps) to stop.

Command descriptions

- When execution control “EN” =1 or changes from 0→1(P instruction), this instruction will enforce the assigned number set of HSPSO (High Speed Pulse Output) to stop pulse output.
- While in the application for mechanical original point reset, as soon as reach the original point can use this instruction to stop the pulse output immediately, so as to make the original point stop at the same position every time when performing mechanical original point resetting.
- For detailed functional description and usage, please refer to Chapter 11 “The NC positioning control of FBs-PLC” for explanation.

FUN143 P PSCNV	CONVERT THE CURRENT PULSE VALUE TO DISPLAY VALUE (mm, Deg, Inch, PS) (Brief description on function)	FUN143 P PSCNV

Ladder symbol



Ps : 0～3; it converts the number of the pulse position to be the mm (Deg, Inch, PS) that has same unit as the set value, so as to make current position displayed.

D : Register that stores the current position after conversion. It uses 2 registers, e.g. if D = D10, which means D10 is Low Word and D11 is High Word.

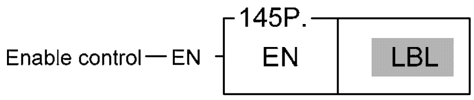
Range / Operand	HR	DR	ROR	K
	R0 | R3839	D0 | D4095	R5000 | R8071	2 | 256
Ps				0 ～3
D	○	○	○	

Command descriptions

- When execution control “En” =1 or changes from 0→1(P instruction), this instruction will convert the assigned current pulse position (PS) to be the mm (or Deg, Inch, or PS) that has same unit as the set value, so as to make current position displaying.
- Only when the FUN140 instruction is executed, then it can get the correct conversion value by executing this instruction.
- For detailed functional description and usage, please refer to Chapter 11 “The NC positioning control of FBs-PLC” for explanation.

FUN145 P EN	ENABLE CONTROL OF THE INTERRUPT AND PERIPHERAL	FUN145 P EN

Ladder symbol



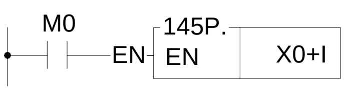
LBL : External input or peripheral label name that to be enabled.

- When enable control "EN" =1 or changes from 0→1 (P instruction), it allows the external input or peripheral interrupt action which is assigned by LBL.
- The enabled interrupt label name is as follows:(Please refer the section 9.3 for details)

LBL name	Description	LBL name	Description	LBL name	Description
HSTAI	HSTA High speed counter interrupt	X4+I	X4 positive edge interrupt	X10+I	X10 positive edge interrupt
HSC0I	HSC0 High speed counter interrupt	X4−I	X5 negative edge interrupt	X10−I	X10 negative edge interrupt
HSC1I	HSC1 High speed counter interrupt	X5+I	X5 positive edge interrupt	X11+I	X11 positive edge interrupt
HSC2I	HSC2 High speed counter interrupt	X5−I	X5 negative edge interrupt	X11−I	X11 negative edge interrupt
HSC3I	HSC3 High speed counter interrupt	X6+I	X6 positive edge interrupt	X12+I	X12 positive edge interrupt
X0+I	X0 positive edge interrupt	X6−I	X6 negative edge interrupt	X12−I	X12 negative edge interrupt
X0−I	X0 negative edge interrupt	X7+I	X7 positive edge interrupt	X13+I	X13 positive edge interrupt
X1+I	X1 positive edge interrupt	X7−I	X7 negative edge interrupt	X13−I	X13 negative edge interrupt
X1−I	X1 negative edge interrupt	X8+I	X8 positive edge interrupt	X14+I	X14 positive edge interrupt
X2+I	X2 positive edge interrupt	X8−I	X8 negative edge interrupt	X14−I	X14 negative edge interrupt
X2−I	X2 negative edge interrupt	X9+I	X9 positive edge interrupt	X15+I	X15 positive edge interrupt
X3+I	X3 positive edge interrupt	X9−I	X9 negative edge interrupt	X15−I	X15 negative edge interrupt
X3−I	X3 negative edge interrupt				

- In practical application, some interrupt signals should not be allowed to work at sometimes, however, it should be allowed to work at some other times. Employing FUN146 (DIS) and FUN145 (EN) instructions could attain the above mentioned demand.

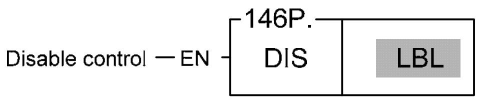
Program example



- When M0 changes from 0→1, it allows X0 to send interrupt when X0 changes from 0→1. CPU can rapidly process the interrupt service program of X0+I.

FUN146 P DIS	DISABLE CONTROL OF THE INTERRUPT AND PERIPHERAL	FUN146 P DIS

Ladder symbol



LBL : Interrupt label intended to disable or peripheral name to be disabled.

- When prohibit control “EN” =1 or changes from 0→1 (P instruction), it disable the interrupt or peripheral operation designated by LBL.
- The interrupt label name is as follows:

LBL name	Description	LBL name	Description	LBL name	Description
HSTAI	HSTA High speed counter interrupt	X4+I	X4 positive edge interrupt	X10+I	X10 positive edge interrupt
HSC0I	HSC0 High speed counter interrupt	X4−I	X5 negative edge interrupt	X10−I	X10 negative edge interrupt
HSC1I	HSC1 High speed counter interrupt	X5+I	X5 positive edge interrupt	X11+I	X11 positive edge interrupt
HSC2I	HSC2 High speed counter interrupt	X5−I	X5 negative edge interrupt	X11−I	X11 negative edge interrupt
HSC3I	HSC3 High speed counter interrupt	X6+I	X6 positive edge interrupt	X12+I	X12 positive edge interrupt
X0+I	X0 positive edge interrupt	X6−I	X6 negative edge interrupt	X12−I	X12 negative edge interrupt
X0−I	X0 negative edge interrupt	X7+I	X7 positive edge interrupt	X13+I	X13 positive edge interrupt
X1+I	X1 positive edge interrupt	X7−I	X7 negative edge interrupt	X13−I	X13 negative edge interrupt
X1−I	X1 negative edge interrupt	X8+I	X8 positive edge interrupt	X14+I	X14 positive edge interrupt
X2+I	X2 positive edge interrupt	X8−I	X8 negative edge interrupt	X14−I	X14 negative edge interrupt
X2−I	X2 negative edge interrupt	X9+I	X9 positive edge interrupt	X15+I	X15 positive edge interrupt
X3+I	X3 positive edge interrupt	X9−I	X9 negative edge interrupt	X15−I	X15 negative edge interrupt
X3−I	X3 negative edge interrupt				

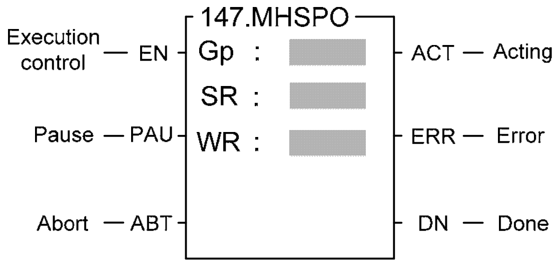
- In practical application, some interrupt signals should not be allowed to work at certain situation. To achieve this, this instruction may be used to disable the interrupt signal.

Program example

- When M0 changes from 0→1, it prohibits X2 from sending interrupt when X2 changes from 0→1.

FUN 147 MHSPO	Multi-Axis High Speed Pulse Output	FUN 147 MHSPO

Ladder symbol



Gp : Group number (0～1)

SR : Starting register for positioning program (example explanation)

WR : Starting register for instruction operation (example explanation). It controls 9 registers, which the other program cannot repeat in using.

Range / Operand	HR R0 – R3839	DR D0 – D3999	ROR R5000 – R8071	K
Gp				0～1
SR	○	○	○	
WR	○	○	○*	

Instruction Explanation

1. The FUN147 (MHSPO) instruction is used to support the linear interpolation for multi-axis motion control, it consists of the motion program written and edited with text programming. We named every position point as a step (which includes output frequency, traveling distance, and transfer conditions). Every step of positioning point owns 15 registers for coding.
2. The FUN147 (MHSPO) instruction can support up to 4 axes for simultaneous linear interpolation; or 2 sets of 2-axis linear interpolation (i.e. Gp0 = Axes Ps0 & Ps1 **;** Gp1 = Axes Ps2 & Ps3)
3. The best benefit to store the positioning program into the registers is that in the case of association with MMI (Man Machine Interface) to operate settings, it may save and reload the positioning program via MMI when replacing the molds.
4. When execution control “EN”=1, if the other FUN147/FUN140 instructions to control Ps0～3 are not active (corresponding status of Ps0=M1992, Ps1=M1993, Ps2=M1994, and Ps3=M1995 will be ON), it will start to execute from the next step of positioning point (when goes to the last step, it will be restarted from the first step to perform); if Ps0～3 is controlled by other FUN147/FUN140 instruction (corresponding status of Ps0=M1992, Ps1=M1993, Ps2=M1994, and Ps3=M1995 would be OFF), this instruction will acquire the pulse output right of positioning control once the controlling FUN147/FUN140 has released the control right.
5. When execution control input “EN” =0, it stops the pulse output immediately.
6. When output pause “PAU” =1 and execution control “EN” was 1 beforehand, it will pause the pulse output. When output pause “PAU” =0 and execution control is still 1, it will continue the unfinished pulse output.
7. When output abort “ABT”=1, it stops pulse output immediately. (When the execution control input “EN” becomes 1 next time, it will restart from the first step of positioning point to execute.)
8. While the pulse is in output transmitting, the output indication “ACT” is ON.
9. When there is execution error, the output indication “ERR” will be ON.
(The error code is stored in the error code register.)
10. When each step of positioning point is complete, the output indication “DN” will be ON.
11. Please refer to Chapter 11 “The NC Positioning Control of FBs-PLC” for further details.

FUN148 MPG	MANUAL PULSE GENERATOR FOR POSITIONING	FUN148 MPG

```
                 ┌148. MPG──┐
Execution   EN ──┤  Sc :    ├──ACT
                 │  Ps :    │
                 │  Fo :    │
                 │  Mr :    │
                 │  WR :    │
                 └──────────┘
```

Sc : Source of high speed counter; 0~7
Ps : Axis of pulse output; 0~3
Fo : Setting of output speed (2 registers)
Mr : Setting of multiplier (2 registers)
  Mr+0 : Multiplicand (Fa)
  Mr+1 : Dividend (Fb)
WR : Starting address of working registers, it needs 4 registers
∗This instruction can be supported in PLC OS firmware V4.60 or late

Operand \ Range	HR R0–R3839	ROR R5000–R8071	DR D0–D3999	K 16 bit
Sc	○	○	○	0~7
Ps	○	○	○	0~3
Fo	○	○	○	
Mr	○	○	○	
WR	○	○*	○	

- Let this instruction be executed in 50mS fixed time interrupt service routine (50MSI)、or by using the 0.1mS high speed timer to generate 50mS fixed time interrupt service to have accurate repeat time to sample the pulse input from manual pulse generator. If it comes the input pulses, it will calculate the number of pulses needing to output according to the setting of multiplier (Mr+0 and Mr+1), and then outputs the pulse stream in the speed of setting (Fo) during this time interval.

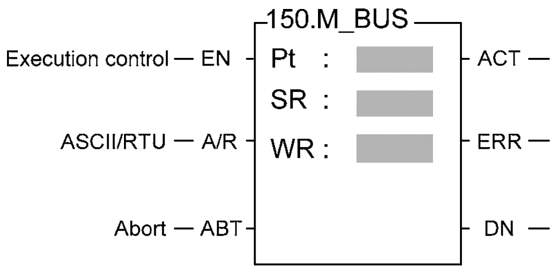
  The setting of output speed (Fo) must be fast enough, and the acceleration / deceleration rate (Parameter 4 and parameter 8 of FUN141 instruction) must be sharp to guarantee it can complete the sending of pulse stream during the time interval if it is under high multiplier (100 or 200 times) situation.
- When execution 〝EN〞=1, this instruction will sample the pulse input from manual pulse generator by reading the current value of assigned high speed counter every time interval; it doesn't have any output if it doesn't have any input pulse; but If it senses the input pulses, it will calculate the number of pulses needing to output according to the setting of multiplier (Mr+0 and Mr+1), and then outputs the pulse stream in the speed of setting (Fo) during this time interval.

  Number of output pulses = (Number of input pulses × Fa) / Fb
- This instruction also under the control of hardware resource management; it wouldn't be executed if the hardware is occupied.
- The output indicator ACT=1 if it outputs the pulses; otherwise ACT=0.
- Please refer to Chapter 11 "The NC Positioning Control of FBs-PLC" for further details.

```
        50mS                                    50mS
 ←──              ──→                    ←──              ──→
 │  · Sample pulse input  │     ...      │  · Sample pulse input  │
 │  · Output pulse stream │              │  · Output pulse stream │
 │    in the speed of Fo  │              │    in the speed of Fo  │
```

FUN150 M-BUS	MODBUS MASTER INSTRUCTION （WHICH MAKES PLC AS THE MODBUS MASTER THROUGH PORT 1~4）	FUN150 M-BUS

Ladder symbol



Pt ：1~4, specify the communication port being acted as the Modbus master

SR：Starting register of communication program

WR **:**Starting register for instruction operation. It controls 8 registers, the other programs can not repeat in using.

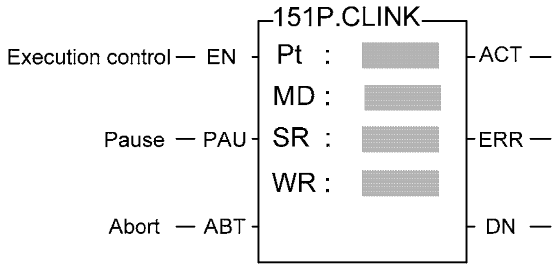
Range / Operand	HR	ROR	DR	K
	R0 ∣ R3839	R5000 ∣ R8071	D0 ∣ D4095	
Pt				1～4
SR	○	○	○	
WR	○	○*	○	

Description

1. FUN150 (M-BUS) instruction makes PLC act as Modbus master through Port 1～4, thus it is very easy to communicate with the intelligent peripheral with Modbus RTU/ASCII protocol.
2. The master PLC may connect with 247 slave stations through the RS-485 interface.
3. Only the master PLC needs to use Modbus RTU/ASCII instruction.
4. It employs the program coding method or table filling method to plan for the data flow controls; i.e. from which one of the slave station to get which type of data and save them to the master PLC, or from the master PLC to write which type of data to the assigned slave station. It needs only seven registries to make definition; every seven registers define one packet of data transaction.
5. When execution control 〝EN〞 changes from 0→1 and both inputs Pause “PAU” and Abort “ABT” are 0, and if Port 1/2/3/4 hasn’t been controlled by other communication instructions [i.e. M1960 (Port1) / M1962 (Port2) / M1936 (Port3) / M1938 (Port4) = 1], this instruction will control the Port 1/2/3/4 immediately and set the M1960/M1962/M1936/M1938 to be 0 (which means it is being occupied), then going on a packet of data transaction immediately. If Port 1/2/3/4 has been controlled (M1960/M1962/M1936/M1938 = 0), then this instruction will enter into the standby status until the controlling communication instruction completes its transaction or pause/abort its operation to release the control right (M1960/M1962/M1936/M1938 =1), and then this instruction will become enactive, set M1960/M1962/M1936/M1938 to be 0, and going on the data transaction immediately.
6. While in transaction processing, if operation control “ABT” becomes 1, this instruction will abort this transaction immediately and release the control right (M1960/M1962/M1936/M1938 = 1). Next time, when this instruction takes over the transmission right again, it will restart from the first packet of data transaction.
7. While 〝A/R〞 =0，Modbus RTU protocol； 〝A/R〞 =1，Modbus ASCII protocol。
8. While it is in the data transaction, the output indication “ACT” will be ON.
9. If there is error occurred when it finishes a packet of data transaction, the output indication “DN” & “ERR” will be ON.
10. If there is no error occurred when it finishes a packet of data transaction, the output indication “DN” will be ON.

FUN 151 CLINK	COMMUNICATION LINK INSTRUCTION (WHICH MAKES PLC ACT AS THE MASTER STATION IN CPU LINK NETWORK THROUGH PORT 1~4)	FUN 151 CLINK

Ladder symbol



Pt : Assign the port, 1～4

MD : Communication mode, MD0~MD3

SR : Starting register of communication table (see example for its explanation)

WR : Starting register for instruction operation (see example for its explanation). It controls 8 registers, the other programs can not repeat in using.

Range / Operand	HR R0 ~ R3839	ROR R5000 ~ R8071	DR D0 ~ D4095	K
Pt				1～4
MD				0～3
SR	○	○	○	
WR	○	○*	○	

Description

● This instruction provides 4 instruction modes MD0～MD3. Of which, three instruction modes MD0～MD2, are "regular link network", and the MD3 is the "high speed link network". The following are the function description of respective modes.

- MD0 : Master station mode for FATEK CPU LINK.
  For any PLC, whose ladder program contains the FUN151:MD0 instruction, will become master station of FATEK CPU LINK network. The master station PLC will base on the communication program stored in data registers in which the target station, data type, data length, etc, were specified to read or write slave station via "FATEK FB-PLC Communication Protocol" command. With this approach up to 254 PLC stations can share the data each other

- MD1 : Active ASCII data transmission mode.
  With this mode, the FUN151 instruction will parse the communication program stored in data registers and base on the parsing result send the data from port2 to ASCII peripherals (such as computer, other brand PLC, inverter, moving sign, etc, this kind of device can command by ASCII message). The operation can set to be (1) transmit only, which ignores the response from peripherals, (2) transmit and then to receive the response from peripherals. When operate with mode (2) then the user must base on the communication protocol of peripheral to parsing and prepare the response message by writing the ladder instructions.

- MD2 : Passive ASCII data receiving mode.
  With this mode, the FUN151 will first wait to receive ASCII messages sent by external ASCII peripherals (such as computer, other brand PLC, card reader, bar code reader, electronic weight, etc. this kind of device can send ASCII message). Upon receiving the message, the user can base on the communication protocol of peripheral to parsing and react accordingly. The operation can set to (1) receive only without responding, or (2) receive then responding. For operation mode (2) the user can use the table driver method to write a communication program and after received a message this instruction can base on this communication program automatically reply the message to peripheral.

- MD3 : Master station mode of FATEK high speed CPU LINK.
  The most distinguished difference between this mode and MD0 is that the communication response of MD3 is much faster than MD0. With The introduction of MD3 mode CPU LINK, The FATEK PLC can easily to implement the application of distributed control and real time data monitoring.

FUN160 D P RWFR	READ/WRITE FILE REGISTER	FUN160 D P RWFR

Ladder symbol

Sa: Starting address of data register

Sb: Starting address of file register

Pr : Record pointer register

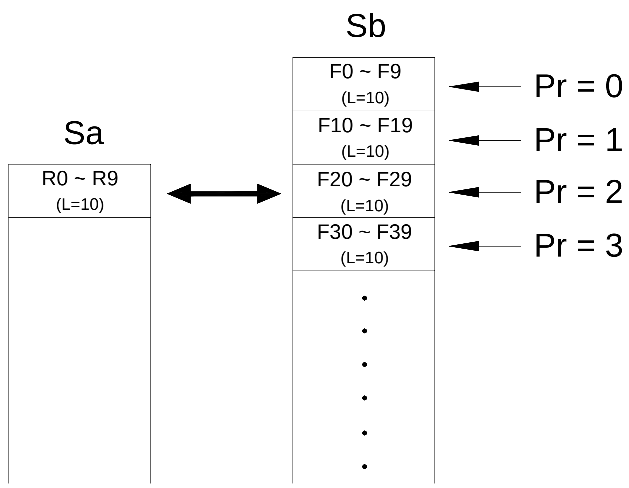
L : Quantity of register to form a record, 1~511

Sa operand can combine V、Z、P0~P9 for index addressing.

Range / Operand	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		

Description

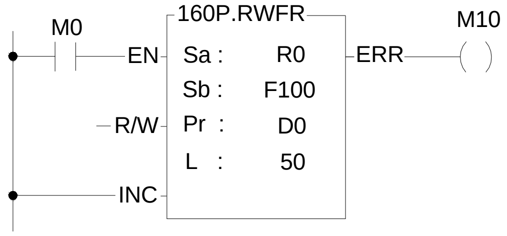
- When operation control "EN"=1 or changes from 0→1(P instruction), it will perform the read ("R/W"=1) or write ("R/W"=0) file register operation. While reading, the content of data registers starting from Sa will be overwritten by the content of file registers addressed by the base file register Sb and record pointer Pr; while writing, the content of file registers addressed by the base file register Sb and record pointer Pr will be overwritten by the content of data registers starting from Sa; L is the operation quantity or record size. The access of file register adopts the concept of RECORD data structure to implement. For example, Sa=R0, Sb=F0, L=10, the read/write details shown as below



FUN160 D P RWFR	READ/WRITE FILE REGISTER	FUN160 D P RWFR

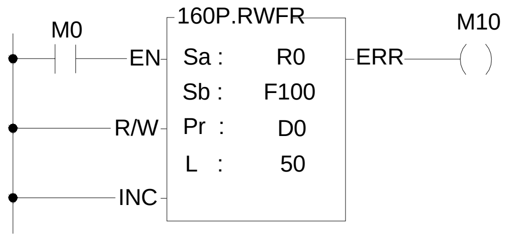
- For ladder program application, only this instruction can access the file registers.
- The record pointer will be increased by 1 after execution while pointer control input "INC"=1.
- This instruction will not be executed and error indicator "ERR" will be 1 while incorrect record size (L=0 or > 511) or the operation out of the file register's range (F0～F8191).

.



When M0 changes from 0→1, if D0 =2, the contents of file registers F200~F249 will be overwritten by the content of data registers R0~R49. the record length is 50.

.Pointer will be increased by 1 after operation.

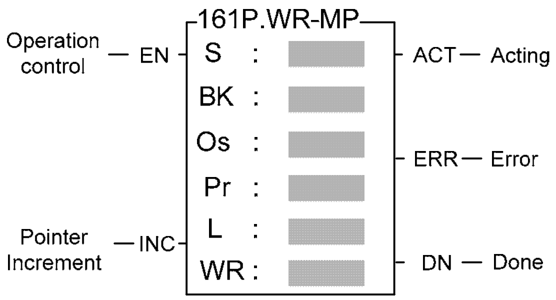


.When M0 changes from 0→1, if D0 = 1, the content of data registers R0~R49 will be overwritten by the file registers F150~F199.

.The record pointer will be increased by 1 after operation.

FUN161P WR-MP	Write Data Record into the MEMORY_PACK （Write memory pack）	FUN161P WR-MP

Ladder symbol



S ：Starting address of the source data

BK：Block number of the MEMORY_PACK，0～1

Os：Offset of the block

Pr：Address of the pointer

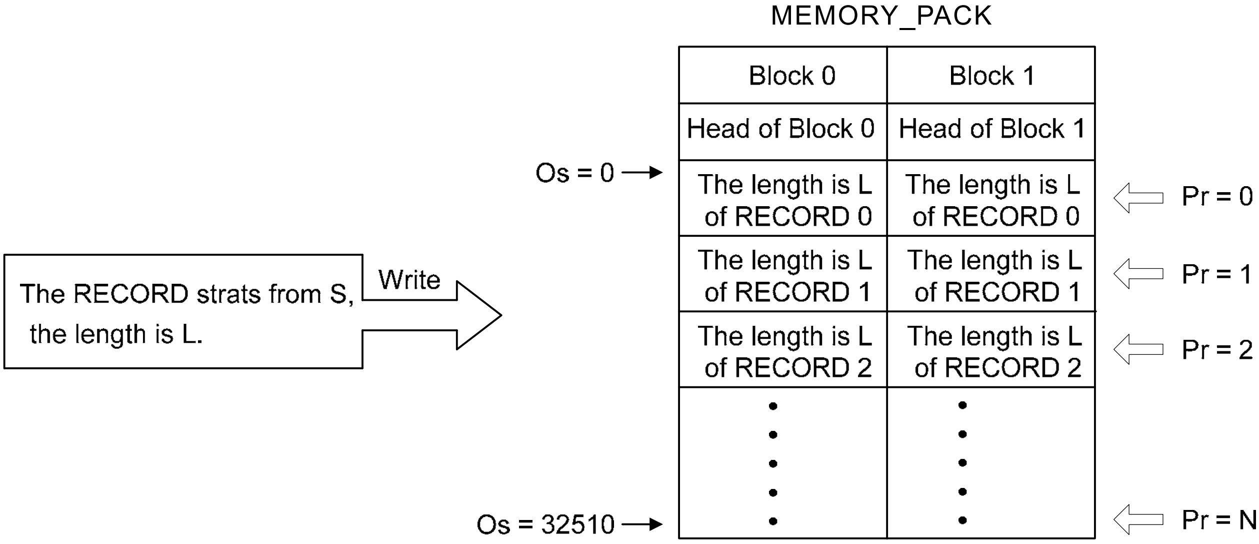
L：Quantity of writing，1～128

WR：Starting address of working registers, it takes 2 registers

S may combine with V、Z、P0～P9 for indirect addressing application

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

- The main purpose of the MEMORY_PACK of FBs series's is used for long term storing of the user's ladder program, except this, through the FUN161/FUN162 instructions, the MEMORY_PACK can be worked as the portable MEMORY_PACK for machine working parameters's saving and loading.
  When execution control〝EN〞changes from 0→1, it will perform the data writing, where S is the starting address of the source data, BK is the block number of the MEMORY_PACK to store this writing, Os is the offset of specified block, Pr is the pointer to point to corresponding data area, L is the quantity of this writing. The access of MEMORY_PACK manipulation adopts the concept of RECORD data structure to implement with. The working diagram as shown below :



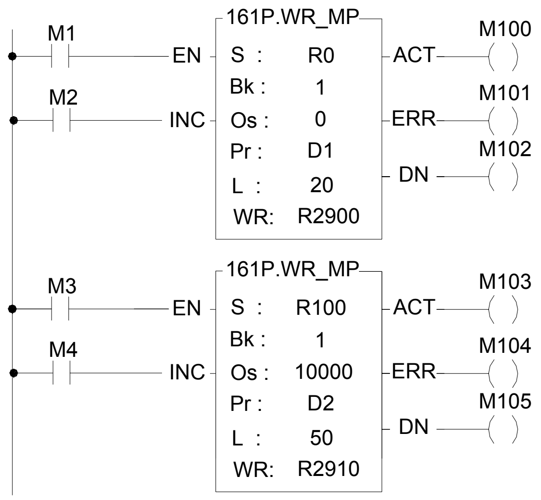
- When input "INC" = 1, the content of the pointer will be increased by one after the execution of writing, it points to next record.

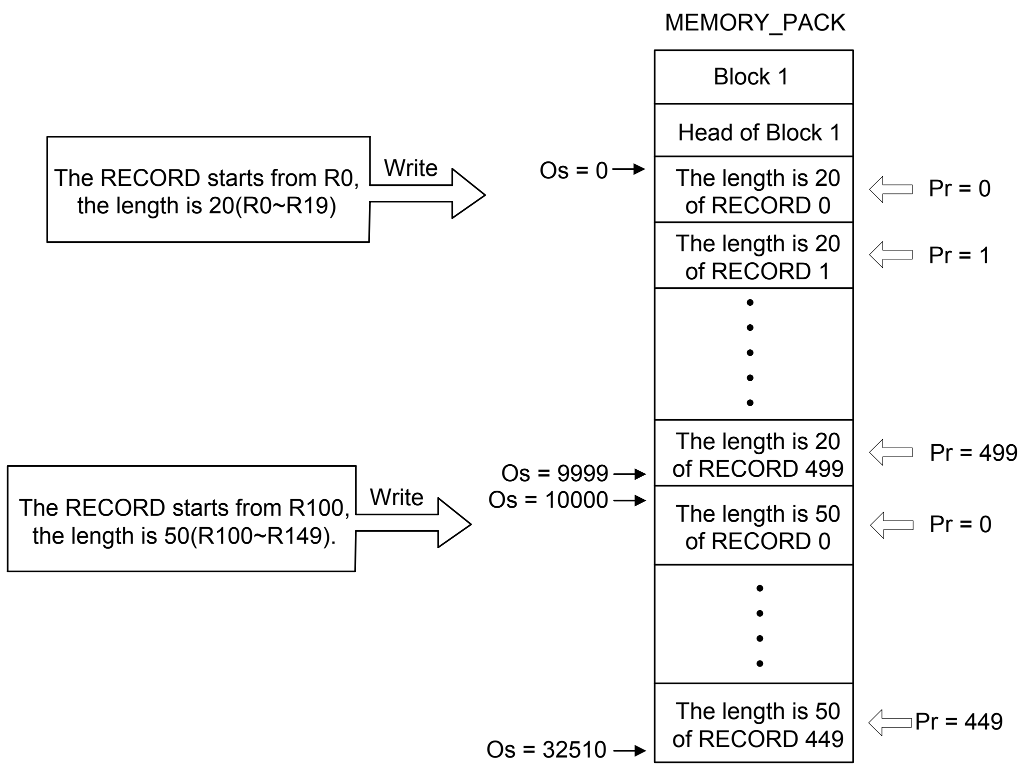
FUN161P WR-MP	Write Data Record into the MEMORY_PACK （Write memory pack）	FUN161P WR-MP

- If the value of L is equal to 0 or greater than 128, or the pointed data area over the range, the output "ERR" will be 1, it will not perform the writing operation.
- It needs couple of PLC solving scans for data writing and verification; during the execution, the output "ACT" will be 1; when completing the execution and verification without the error, the output "DN" will be 1; when completing the execution and verification with the error, the output "ERR" will be 1.

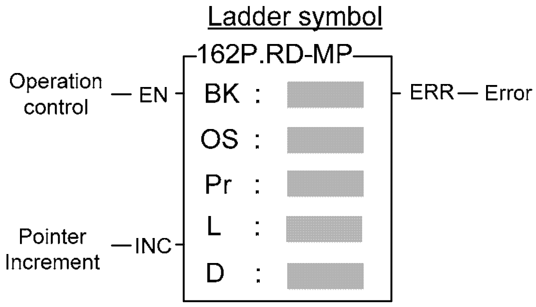
  The MEMORY_PACK can be configured to store the user's ladder program or machine's working parameters, or both. The ladder program can be stored into the block 0 only, but the machine's working parameters can be stored into block 0 or 1; the memory capacity of each block has 32K Word in total.

Example program : Writing the record into block 1 of MEMORY_PACK with the different length





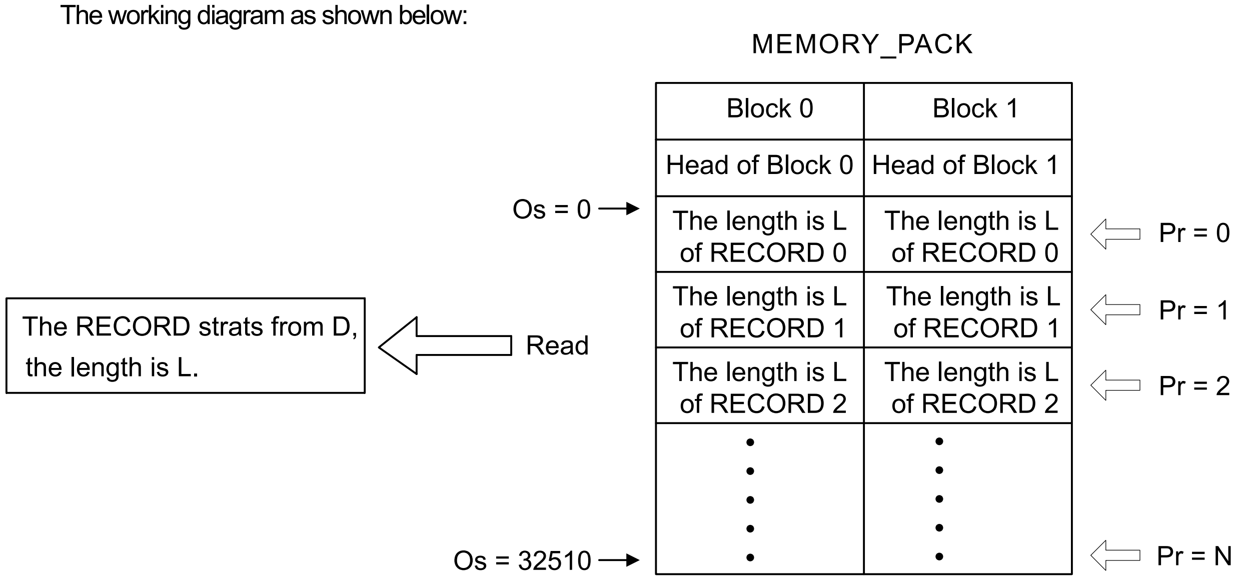
FUN162 P RD-MP	Read Data Record from the MEMORY_PACK （Read memory pack）	FUN162 P RD-MP



BK：Block number of the MEMORY_PACK，0～1
Os：Offset of the block
Pr：Address of the pointer
L：Quantity of reading，1～128
D：Starting address to store the reading record

Operand \ Range	HR R0–R3839	ROR R5000–R8071	DR D0–D3999	K
BK				0～1
Os	○	○	○	0～32510
Pr	○	○*	○	
L	○	○*	○	1～128
D	○	○*	○	

- If the MEMORY_PACK of the FBs series's has stored the data record written by the FUN161 instruction, they can be read out for machine's working through this instruction, it will reduce the tuning time for machine operation.
- When execution control "EN" = 1 or from 0→1(P instruction), it will perform the data reading, where BK is the block number of the MEMORY_PACK storing the record, Os is the offset of specified block, Pr is the pointer to point to corresponding data area, L is the quantity of this record, and D is the starting address to stor this reading of record. The access of MEMORY_PACK manipulation adopts the concept of RECORD data structure to implement with.
  The working diagram as shown below:



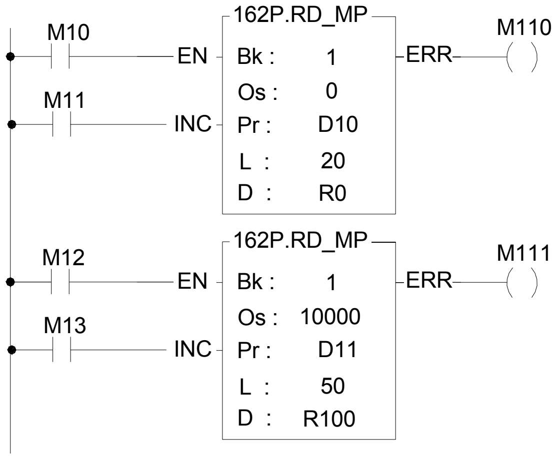
- When input "INC"=1, the content of the pointer will be increased by one after the execution of reading, it points to next record.

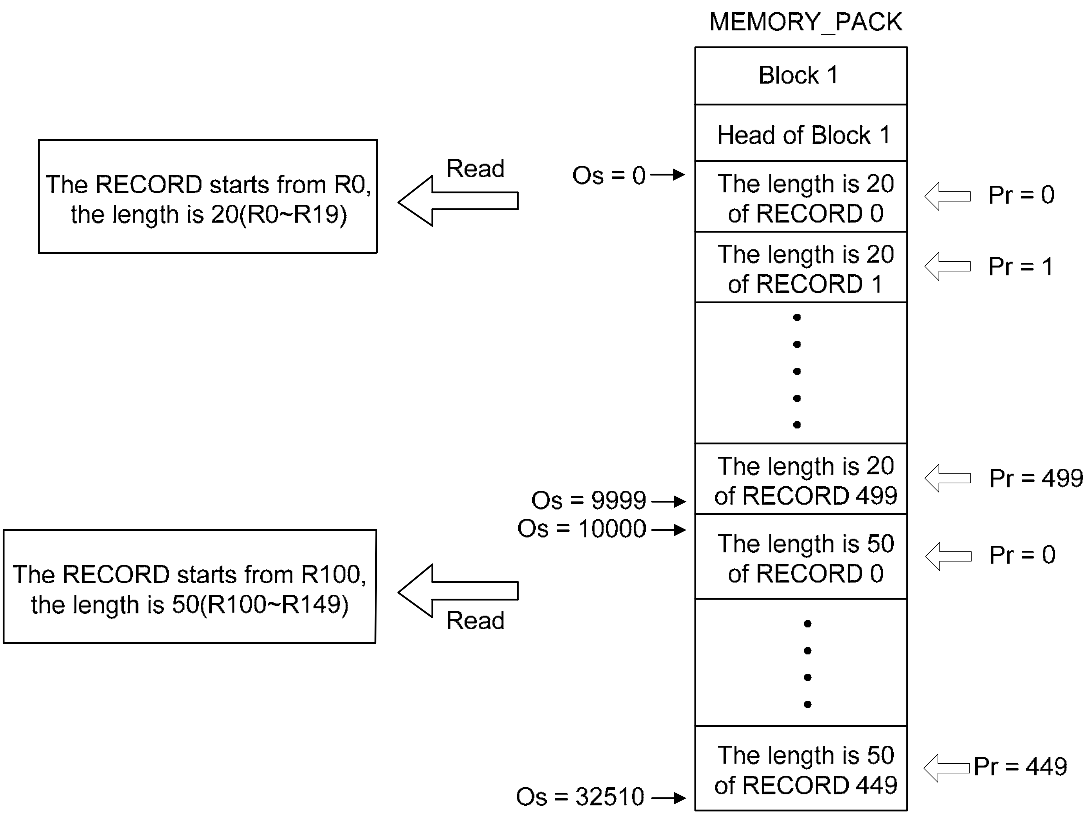
FUN162 P RD-MP	Read Data Record from the MEMORY_PACK （Read memory pack）	FUN162 P RD-MP

- If the value of L is equal to 0 or greater than 128, or the pointed data area over the range, the output "ERR" will be 1, it will not perform the reading operation.
- Output will be “ERR” if MEMORY_PACK is empty or data format not correct, and user used FUN162 to read data from MEMORY_PACK.

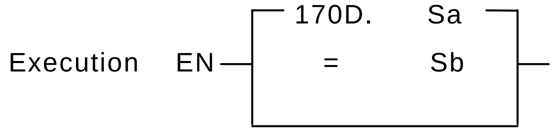
Example program : Reading the record from block 1 of MEMORY_PACK with the different length

※ It is necessary that correct data in MEMORY_PACK or this example can’t execute correctly.





FUN170 D =	EQUAL TO COMPARE （Compare whether Sa is equal to Sb）	FUN170 D =

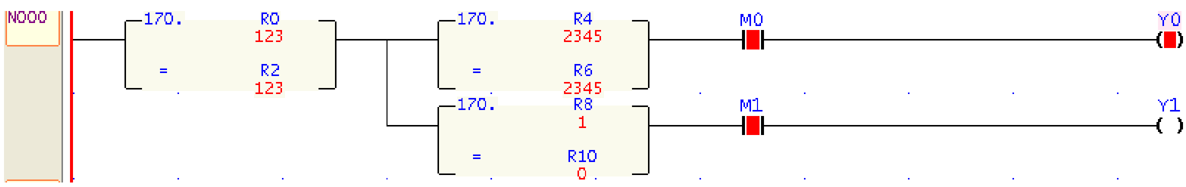


Sa：Operand A or the starting address of Sa
Sb：Operand B or the starting address of Sb
Sa、Sb may combine with V、Z、P0～P9 for indirect addressing application
∗This instruction can be supported in PLC OS firmware V4.60 or later

Range / Operand	WX	WY	WM	WS	TMR	CTR	HR	SR	ROR	DR	K	XR
	WX0 | WX240	WY0 | WY240	WM0 | WM1896	WS0 | WS984	T0 | T255	C0 | C255	R0 | R3839	R3804 | R4167	R5000 | R8071	D0 | D3999	16/ 32 bit +/- number	V、Z P0~P9
Sa	○	○	○	○	○	○	○	○	○	○	○	○
Sb	○	○	○	○	○	○	○	○	○	○	○	○

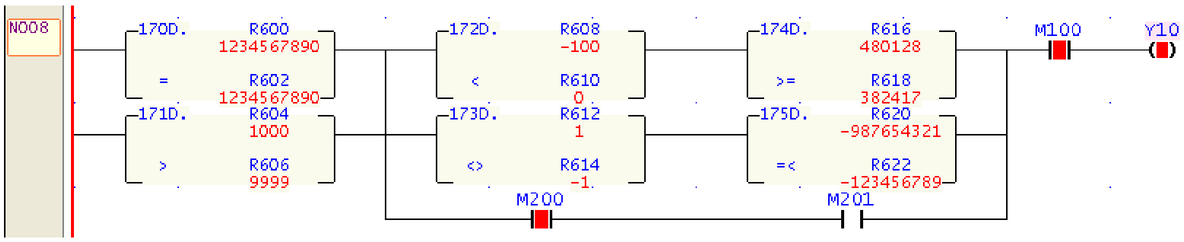
- When execution input 〝EN〞=1, this instruction will be executed in signed number to compare Sa with Sb. If Sa=Sb, the output is 1; otherwise the output is 0.

Example 1：



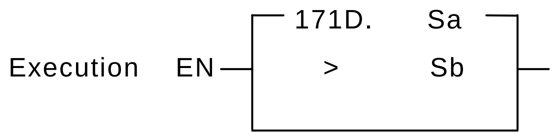
Description: When R0=R2、R4=R6 and M0=1, the output status of Y0 is 1; otherwise it is 0
R0=R2、R8=R10 and M1=1, the output status of Y1 is 1; otherwise it is 0

Example 2：



Description: When DR600=DR602 or DR604>DR606, after them DR608<DR610 and DR616≧DR618, or DR612≠DR614 and DR620≦DR622, or M200=1and M201=1, and then M100=1, the output status of Y10 is 1; otherwise it is 0.

FUN171 D >	GREATER THAN COMPARE （Compare whether Sa is greater than Sb）	FUN171 D >

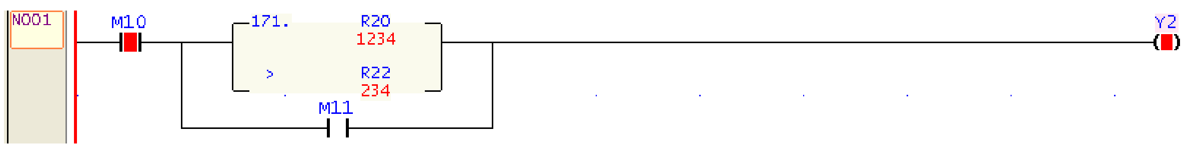


Sa：Operand A or the starting address of Sa
Sb：Operand B or the starting address of Sb
Sa、Sb may combine with V、Z、P0～P9 for indirect addressing application
∗This instruction can be supported in PLC OS firmware V4.60 or later

Range / Operand	WX	WY	WM	WS	TMR	CTR	HR	SR	ROR	DR	K	XR
	WX0 ｜ WX240	WY0 ｜ WY240	WM0 ｜ WM1896	WS0 ｜ WS984	T0 ｜ T255	C0 ｜ C255	R0 ｜ R3839	R3804 ｜ R4167	R5000 ｜ R8071	D0 ｜ D3999	16/ 32 bit +/- number	V、Z P0~P9
Sa	○	○	○	○	○	○	○	○	○	○	○	○
Sb	○	○	○	○	○	○	○	○	○	○	○	○

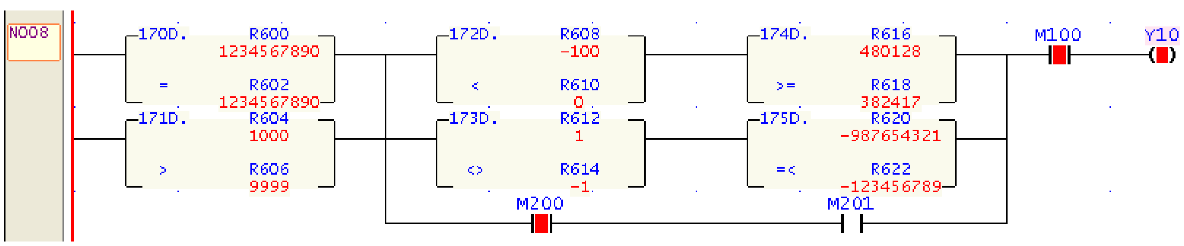
● When execution input〝EN〞=1, this instruction will be executed in signed number to compare Sa with Sb. If Sa>Sb, the output is 1; otherwise the output is 0.

Example 1：



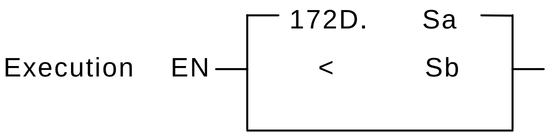
Description: When M10=1、R20 > R22 or M11=1, the output status of Y2 is 1; otherwise it is 0.

Example 2：



Description: When DR600=DR602 or DR604>DR606, after them DR608<DR610 and DR616≧DR618, or DR612≠DR614 and DR620≦DR622, or M200=1and M201=1, and then M100=1, the output status of Y10 is 1; otherwise it is 0.

FUN172 D <	LESS THAN COMPARE （Compare whether Sa is less than Sb）	FUN172 D <

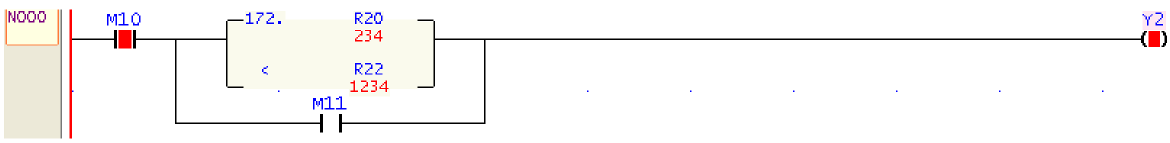


Sa：Operand A or the starting address of Sa
Sb：Operand B or the starting address of Sb
Sa、Sb may combine with V、Z、P0～P9 for indirect addressing application
∗This instruction can be supported in PLC OS firmware V4.60 or later

Range / Operand	WX	WY	WM	WS	TMR	CTR	HR	SR	ROR	DR	K	XR
	WX0 ｜ WX240	WY0 ｜ WY240	WM0 ｜ WM1896	WS0 ｜ WS984	T0 ｜ T255	C0 ｜ C255	R0 ｜ R3839	R3804 ｜ R4167	R5000 ｜ R8071	D0 ｜ D3999	16/ 32 bit +/- number	V、Z P0~P9
Sa	○	○	○	○	○	○	○	○	○	○	○	○
Sb	○	○	○	○	○	○	○	○	○	○	○	○

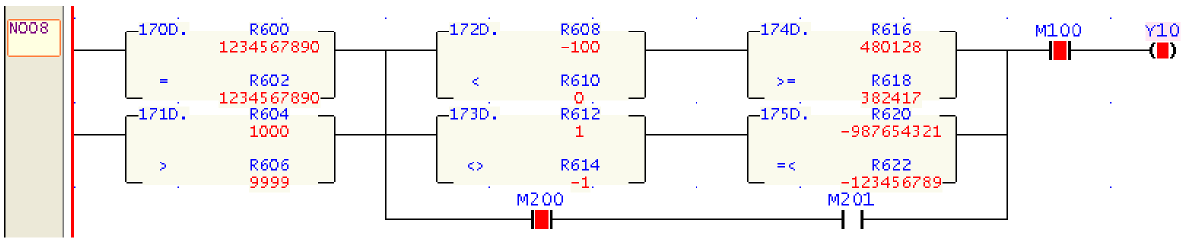
● When execution input〝EN〞=1, this instruction will be executed in signed number to compare Sa with Sb. If Sa<Sb, the output is 1; otherwise the output is 0.

Example 1：



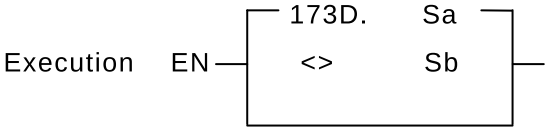
Description: When M10=1、R20＜R22 or M11=1, the output status of Y2 is 1; otherwise it is 0.

Example 2：



Description: When DR600=DR602 or DR604>DR606, after them DR608<DR610 and DR616≧DR618, or DR612≠DR614 and DR620≦DR622, or M200=1and M201=1, and then M100=1, the output status of Y10 is 1; otherwise it is 0.

FUN173 D <>	NOT EQUAL TO COMPARE (Compare whether Sa is not equal to Sb)	FUN173 D <>

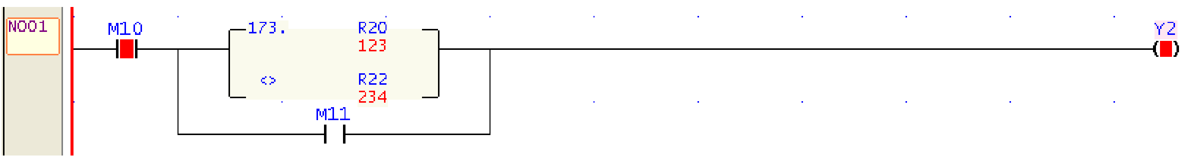


Sa：Operand A or the starting address of Sa
Sb：Operand B or the starting address of Sb
Sa、Sb may combine with V、Z、P0～P9 for indirect addressing application
∗This instruction can be supported in PLC OS firmware V4.60 or later

Range / Operand	WX	WY	WM	WS	TMR	CTR	HR	SR	ROR	DR	K	XR
	WX0 ∣ WX240	WY0 ∣ WY240	WM0 ∣ WM1896	WS0 ∣ WS984	T0 ∣ T255	C0 ∣ C255	R0 ∣ R3839	R3804 ∣ R4167	R5000 ∣ R8071	D0 ∣ D3999	16/ 32 bit +/- number	V、Z P0~P9
Sa	○	○	○	○	○	○	○	○	○	○	○	○
Sb	○	○	○	○	○	○	○	○	○	○	○	○

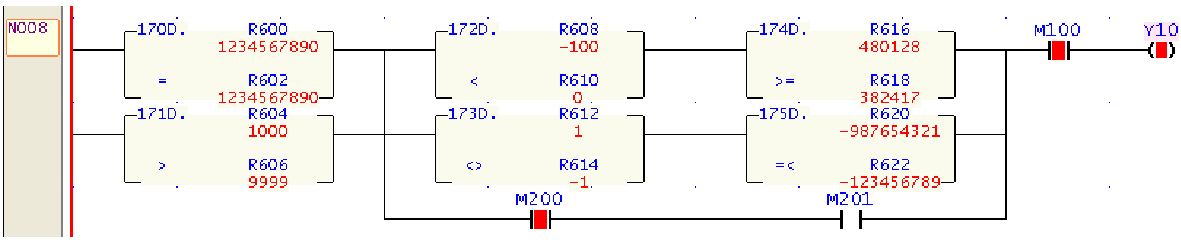
- When execution input 〝EN〞=1, this instruction will be executed in signed number to compare Sa with Sb. If Sa≠Sb, the output is 1; otherwise the output is 0.

Example 1：



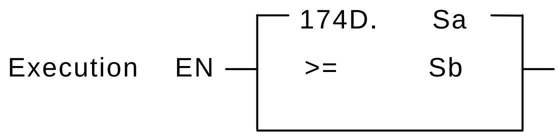
Description: When M10=1、R20≠R22 or M11=1, the output status of Y2 is 1; otherwise it is 0.

Example 2：



Description: When DR600=DR602 or DR604>DR606, after them DR608<DR610 and DR616≧DR618, or DR612≠DR614 and DR620≦DR622, or M200=1and M201=1, and then M100=1, the output status of Y10 is 1; otherwise it is 0.

FUN174 D >=	GREATER THAN OR EQUAL TO COMPARE (Compare whether Sa is greater than or equal to Sb)	FUN174 D >=

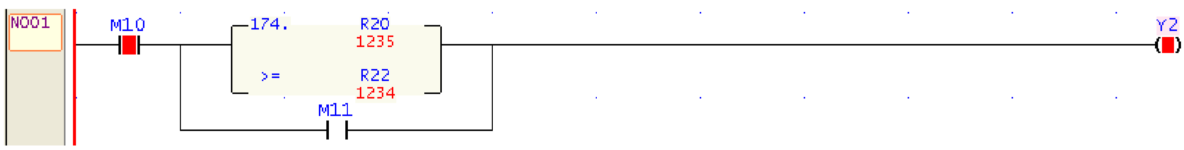


Sa：Operand A or the starting address of Sa
Sb：Operand B or the starting address of Sb
Sa、Sb may combine with V、Z、P0～P9 for indirect addressing application
∗This instruction can be supported in PLC OS firmware V4.60 or later

Range / Operand	WX	WY	WM	WS	TMR	CTR	HR	SR	ROR	DR	K	XR
	WX0 | WX240	WY0 | WY240	WM0 | WM1896	WS0 | WS984	T0 | T255	C0 | C255	R0 | R3839	R3804 | R4167	R5000 | R8071	D0 | D3999	16/ 32 bit +/- number	V、Z P0~P9
Sa	○	○	○	○	○	○	○	○	○	○	○	○
Sb	○	○	○	○	○	○	○	○	○	○	○	○

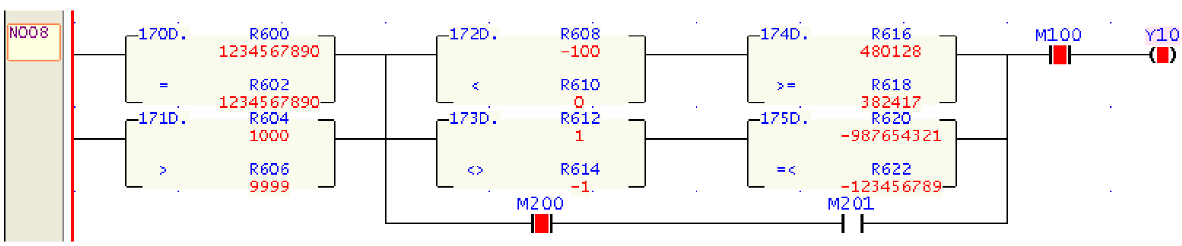
- When execution input 〝EN〞=1, this instruction will be executed in signed number to compare Sa with Sb. If Sa≧Sb, the output is 1; otherwise the output is 0.

Example 1：



Description: When M10=1、R20≧R22 or M11=1, the output status of Y2 is 1; otherwise it is 0.

Example 2：



Description: When DR600=DR602 or DR604>DR606, after them DR608<DR610 and DR616≧DR618, or DR612 ≠DR614 and DR620≦DR622, or M200=1and M201=1, and then M100=1, the output status of Y10 is 1; otherwise it is 0.

FUN175 D =<	LESS THAN OR EQUAL TO COMPARE （Compare whether Sa is less than or equal to Sb）	FUN175 D =<

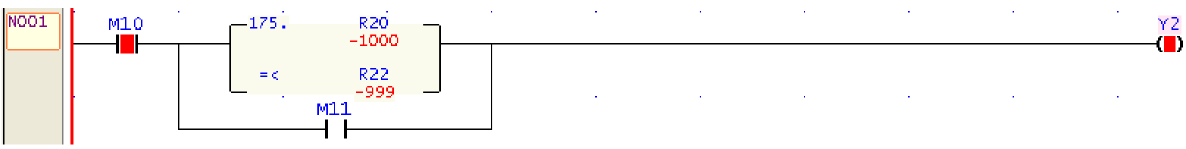
Execution EN ─ [175D. Sa / =< Sb] ─

Sa：Operand A or the starting address of Sa
Sb：Operand B or the starting address of Sb
Sa、Sb may combine with V、Z、P0～P9 for indirect addressing application
∗This instruction can be supported in PLC OS firmware V4.60 or later

Operand \ Range	WX	WY	WM	WS	TMR	CTR	HR	SR	ROR	DR	K	XR
	WX0 │ WX240	WY0 │ WY240	WM0 │ WM1896	WS0 │ WS984	T0 │ T255	C0 │ C255	R0 │ R3839	R3804 │ R4167	R5000 │ R8071	D0 │ D3999	16/ 32 bit +/- number	V、Z P0~P9
Sa	○	○	○	○	○	○	○	○	○	○	○	○
Sb	○	○	○	○	○	○	○	○	○	○	○	○

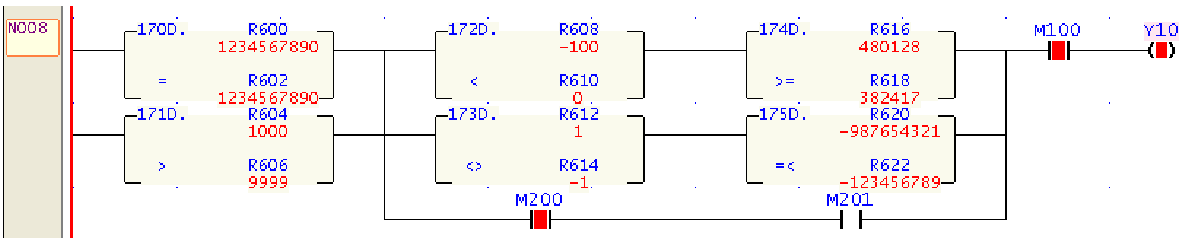
- When execution input〝EN〞=1, this instruction will be executed in signed number to compare Sa with Sb. If Sa≦Sb, the output is 1; otherwise the output is 0.

Example 1：



Description: When M10=1、R20≦R22 or M11=1, the output status of Y2 is 1; otherwise it is 0.

Example 2：



Description: When DR600=DR602 or DR604>DR606, after them DR608<DR610 and DR616≧DR618, or DR612 ≠DR614 and DR620≦DR622, or M200=1and M201=1, and then M100=1, the output status of Y10 is 1; otherwise it is 0.

FUN190 STAT	READ SYSTEM STATUS	FUN190 STAT

Execution EN — 190.STAT
Gp :
D :

Gp : Specified status group
0 : Get information of I/O expansion
1~3 : Reserved
D : Starting address of register to store the system status
D+0 : Quantity of status
D+1 : Status 1
…
D+N: Status N

∗ This instruction can be supported in PLC OS firmware V4.62 or later

Operand \ Range	HR	ROR	DR	K
	R0 ∣ R3839	R5000 ∣ R8071	D0 ∣ D3999	
Gp				0~3
D	○	○*	○	

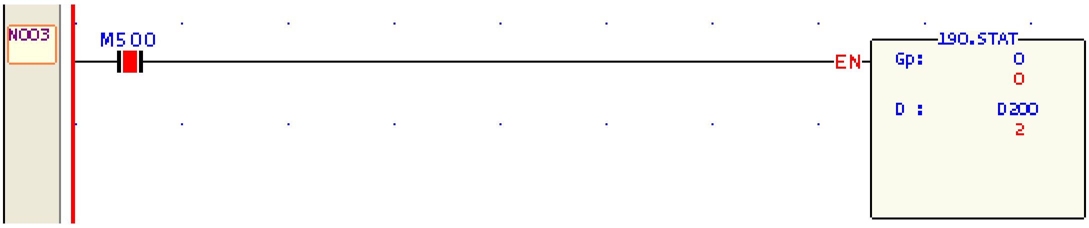
- When execution 〝EN〞=1, this instruction being executed, and if Gp=0, it means to get the information of I/O expansion modules; total quantity of I/O expansion modules will be stored in D register, code of I/O expansion module will be stored in D+1~D+N registers in order. Gp=1~3, reserved for future.

Code of I/O Expansion Module	Name of I/O Expansion Module
1	FBs-8XYR
2	FBs-8X
3	FBs-8YR
4	FBs-16XYR
5	FBs-20X
6	FBs-16YR
7	FBs-24X
8	FBs-24Y
9	FBs-24XYR
10	FBs-40XYR
11	FBs-60XYR
12	FBs-7SG1S (Decode)
13	FBs-7SG1H (Non-decode)
14	FBs-7SG2S (Decode)
15	FBs-7SG2H (Non-decode)
16	FBs-6AD
17	FBs-2DA
18	FBs-4DA
19	FBs-4PT
20	FBs-4A2D

Code of I/O Expansion Module	Name of I/O Expansion Module
21	FBs-6TC
22	FBs-6RTD
23	FBs-16TC
24	FBs-16RTD
25	FBs-2TC
26	FBs-2A4TC
27	FBs-2A4RTD
28	FBs-6NTC
29	FBs-16NTC (Reserved)
30	FBs-32DGI
31	FBs-VOM
32	FBs-1LC

FUN190 STAT	READ SYSTEM STATUS	FUN190 STAT

Example：There are two I/O expansion modules FBs-2DA + FBs-6AD installed in one system



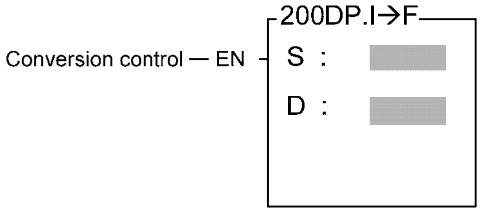
Status Monitoring

Ref. No.	Status	Data	Ref. No.	Status	Data
M500	Enable	ON			
D200	Decimal	2			
D201	Decimal	17			
D202	Decimal	16			

Description: When M500=1, this instruction being executed, register D200 is used to store the total quantity of I/O expansion modules, register D201 is used to store the code (17=FBs-2DA) of first I/O expansion module, register D202 is used to store the code (16=FBs-6AD) of second I/O expansion module.

FUN200 D P I→F	CONVERSION OF INTEGER TO FLOATING POINT NUMBER	FUN200 D P I→F

Ladder symbol



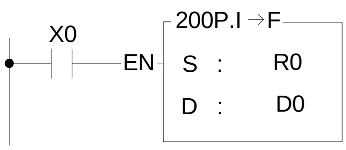
S : Starting register of Integer to be converted

D : Starting register to store the result of conversion

Range / Operand	HR R0 | R3839	ROR R5000 | R8071	DR D0 | D4095	K 16/32 bit Integer	XR V、Z P0～P9
S	○	○	○	○	○
D	○	○*	○		○

Description

- The format of floating point number of Fatek-PLC follows the IEEE-754 standard. For detail explanation of the format please refer to 5.3 (Numbering System)…page 5-9.
- When conversion control "EN" = 1 or has a transition from 0 to 1 (P instruction), will convert the integer data from S register into D~D+1 32-bits register(floating point number data)



※ R0 = 200 (0000000011001000)

Integer To Floating

DD0 = 43480000H

I→F

R0 :

0	0	0	0	0	0	0	0	1	1	0	0	1	0	0	0
b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0

DD0 :

0	1	0	0	0	0	1	1	0	1	0	0	1	0	0	0	0	00⋯0	0
b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16	b15	b14 ~ b1	b0
s	e	e	e	e	e	e	e	e	m	m	m	m	m	m	m	m	mm⋯m	m

FUN201 D P F→I	CONVERSION OF FLOATING POINT NUMBER TO INTEGER	FUN201 D P F→I

Ladder symbol

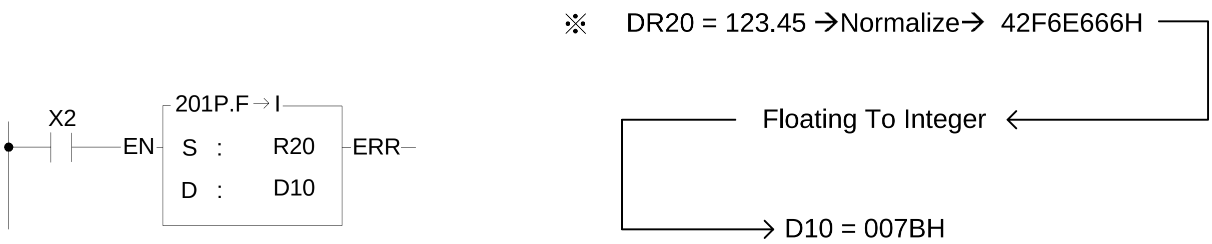
S : Starting register of Integer to be converted

D : Starting register to store the result of conversion

Range / Operand	HR R0 ∣ R3839	ROR R5000 ∣ R8071	DR D0 ∣ D4095	XR V、Z P0～P9
S	○	○	○	○
D	○	○*	○	○

Description

- The format of floating point number of Fatek-PLC follows the IEEE-754 standard. For detail explanation of the format please refer to 5.3 (Numbering System)…page 5-9.
- When conversion control "EN" = 1 or has a transition from 0 to 1 (P instruction), will convert the floating point data from S~S+1 32bits register into D register(integer data).
- If the value exceeds the valid range of destination, then do not carry out this instruction, and set the range-error flag “ERR” as 1 and the D register will be intact.



DR20:

0	1	0	0	0	0	1	0	1	1	1	1	0	1	1	0	1	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0
b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16	b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
s	e	e	e	e	e	e	e	e	m	m	m	m	m	m	m	m	m	m	m	m	m	m	m	m	m	m	m	m	m	m	m

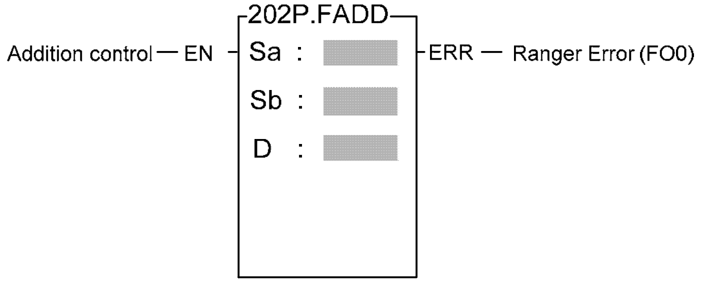
F → I

D10:

0	0	0	0	0	0	0	0	0	1	1	1	1	0	1	1
b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0

FUN202 P FADD	FLOATING POINT NUMBER ADDITION	FUN202 P FADD

Ladder symbol



Sa: Augend

Sb: Addend

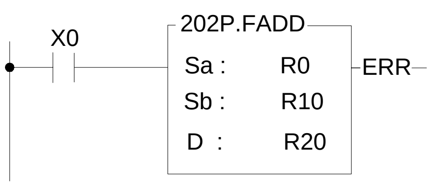
D : Destination register to store the results of the addition

Sa, Sb, D may combine with V, Z, P0~P9 to serve indirect addressing

Operand \ Range	HR R0 ∣ R3839	ROR R5000 ∣ R8071	DR D0 ∣ D4095	K Floating point number	XR V、Z P0～P9
Sa	○	○	○	○	○
Sb	○	○	○	○	○
D	○	○*	○		○

Description

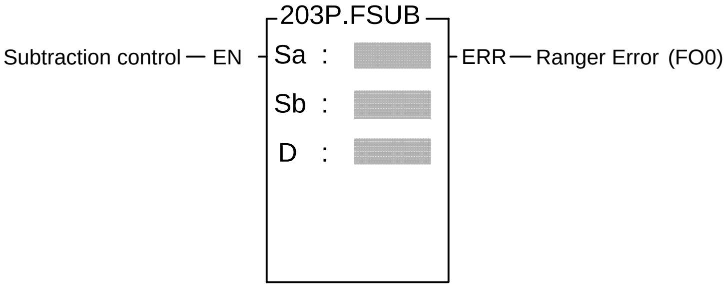
- The format of floating point number of Fatek-PLC follows the IEEE-754 standard. For detail explanation of the format please refer to 5.3 (Numbering System)…page 5-9.
- Performs the addition of the data specified at Sa and Sb and writes the results to a specified register D when the add control input "EN" =1 or from 0 to 1 (P instruction). If the result exceed the range that the floating point number can be expressed($\pm 3.4*10^{38}$) then the error flag FO0 will be set to 1 and the D register will be intact.



DR0	2 0 0	⇨ Floating Point Number :	DR0	4 3 4 8 0 0 0 0 H
DR10	1 5 0	⇨ Floating Point Number :	DR10	4 3 1 6 0 0 0 0 H
		+		
			DR20	4 3 A F 0 0 0 0 H

FUN 203 P FSUB	FLOATING POINT NUMBER SUBTRACTION	FUN 203 P FSUB

Ladder symbol



Sa: Minuend

Sb: Subtrahend

D : Destination register to store the results of the subtraction

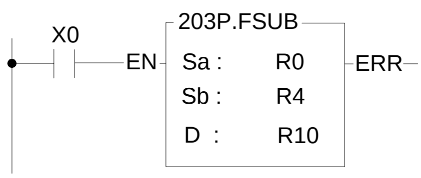
Sa, Sb, D may combine with V, Z, P0~P9 to serve indirect addressing

Range / Operand	HR	ROR	DR	K	XR
	R0 ∣ R3839	R5000 ∣ R8071	D0 ∣ D4095	Floating point number	V、Z P0～P9
Sa	○	○	○	○	○
Sb	○	○	○	○	○
D	○	○*	○		○

Description

- The format of floating point number of Fatek-PLC follows the IEEE-754 standard. For detail explanation of the format please refer to 5.3 (Numbering System)…page 5-9.

- Performs the subtraction of the data specified at Sa and Sb and writes the results to a specified register D when the subtract control input "EN" =1 or from 0 to 1 (P instruction). If the result exceed the range that the floating point number can be expressed($\pm 3.4*10^{38}$) then the error flag FO0 will be set to 1 and the D register will be intact.



DR0	200	⇨ Floating Point Number :	DR0	43480000H
DR4	500	⇨ Floating Point Number :	DR4	43FA0000H
		—		
			DR10	C3960000H

FUN 204 P FMUL	FLOATING POINT NUMBER MULTIPLICATION	FUN 204 P FMUL

Ladder symbol

Multiplication control — EN — 204P.FMUL — ERR — Ranger Error (FO0)

Sa :
Sb :
D :

Sa: Multiplicand

Sb: Multiplier

D : Destination register to store the results of the multiplication

Sa, Sb, D may combine with V, Z, P0~P9 to serve indirect addressing

Range / Operand	HR	ROR	DR	K	XR
	R0 ∣ R3839	R5000 ∣ R8071	D0 ∣ D4095	Floating point number	V、Z P0～P9
Sa	○	○	○	○	○
Sb	○	○	○	○	○
D	○	○*	○		○

Description

- The format of floating point number of Fatek-PLC follows the IEEE-754 standard. For detail explanation of the format please refer to 5.3 (Numbering System)…page 5-9.
- Performs the multiplication of the data specified at Sa and Sb and writes the results to a specified register D when the multiplication control input "EN" =1 or from 0 to 1 (P instruction). If the result exceed the range that the floating point number can be expressed($\pm 3.4*10^{38}$) then the error flag FO0 will be set to 1 and the D register will be intact.

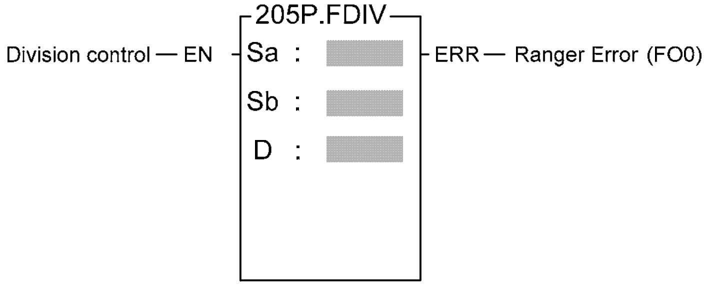
M10 — EN — 204P.FMUL — ERR

Sa : R10
Sb : R12
D : R14

DR10	1 2 3 . 4 5	⇨ Floating Point Number :	DR10	4 2 F 6 E 6 6 6 H
DR12	6 7 8 . 5 4	⇨ Floating Point Number :	DR12	4 4 2 9 A 2 8 F H
		×		
			DR14	4 7 A 3 9 A E 2 H

FUN 205 P FDIV	FLOATING POINT NUMBER DIVISION	FUN 205 P FDIV

Ladder symbol



Sa: Dividend

Sb: Divisor

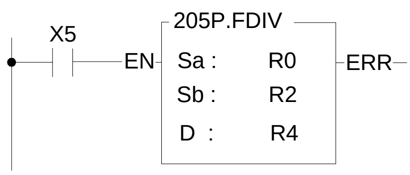
D : Destination register to store the results of the division

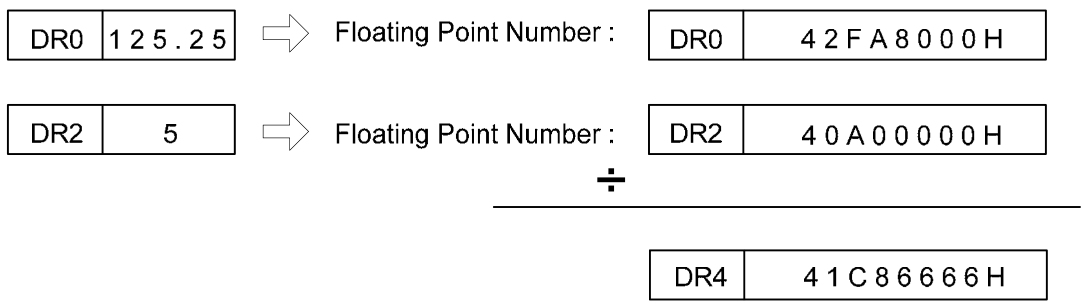
Sa, Sb, D may combine with V, Z, P0~P9 to serve indirect addressing

Operand \ Range	HR R0 | R3839	ROR R5000 | R8071	DR D0 | D4095	K Floating point number	XR V、Z P0～P9
Sa	○	○	○	○	○
Sb	○	○	○	○	○
D	○	○*	○		○

Description

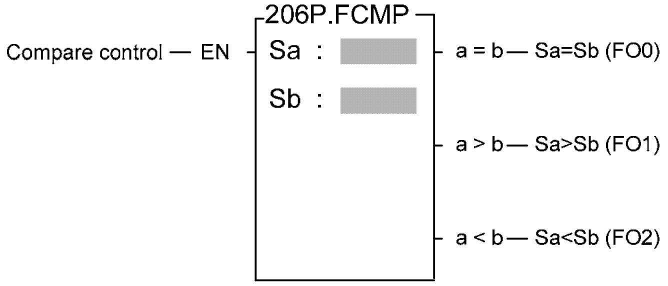
- The format of floating point number of Fatek-PLC follows the IEEE-754 standard. For detail explanation of the format please refer to 5.3 (Numbering System) page 5-9.
- Performs the division of the data specified at Sa and Sb and writes the result to the registers specified by register D when the division control input "EN" =1 or from 0 to 1 (P instruction). If the result exceed the range that the floating point number can be expressed($\pm 3.4*10^{38}$) then the error flag FO0 will be set to 1 and the D register will be intact.





FUN 206 P FCMP	FLOATING POINT NUMBER COMPARE	FUN 206 P FCMP

Ladder symbol



Sa: The register to be compared

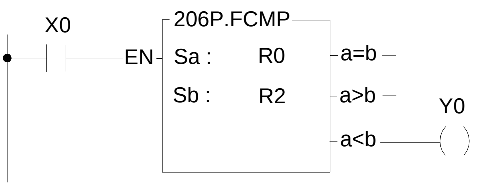
Sb: The register to be compared

Sa, Sb may combine with V, Z, P0~P9 to serve indirect addressing.

Range / Operand	HR	ROR	DR	K	XR
	R0 ∣ R3839	R5000 ∣ R8071	D0 ∣ D4095	Floating point number	V、Z P0～P9
Sa	○	○	○	○	○
Sb	○	○	○	○	○

Description

- The format of floating point number of Fatek-PLC follows the IEEE-754 standard. For detail explanation of the format please refer to 5.3 (Numbering System) page 5-9.
- Compares the data of Sa and Sb when the compare control input "EN" =1 or from 0 to 1 (P instruction). If the data of Sa is equal to Sb, then set FO0 to 1. If the data of Sa>Sb, then set FO1 to 1. If the data of Sa<Sb, then set FO2 to 1. If the data of Sa < Sb, then set the FO2 to 1.

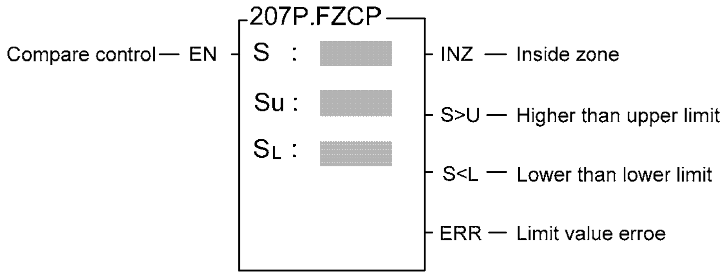


DR0	200.1	⇨ Floating Point Number :	DR0	4348199AH
DR2	200.2	⇨ Floating Point Number :	DR2	43483333H

- From the above example, we first assume the data of DR0 is 200.1 and DR2 is 200.2, and then compare the data by executing the CMP instruction. The FO0 and FO1 are set to 0 and FO2 (a<b) is set to 1 since a<b.
- If you want to have the compound results, such as≧、≦、< > etc., please send =、< and > results to relay first and then combine the result from the relays.

FUN 207 P FZCP	FLOATING POINT NUMBER ZONE COMPARE	FUN 207 P FZCP

Ladder Symbol



S : Register for zone comparison

S_U: The upper limit value

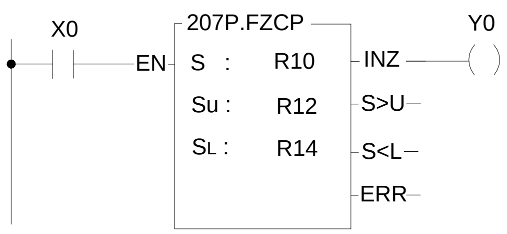
S_L: The lower limit value

S, S_U, S_L may combine with V, Z, P0~P9 to serve indirect address application

Range / Operand	HR R0 | R3839	ROR R5000 | R8071	DR D0 | D4095	K Floating point number	XR V、Z P0～P9
S	○	○	○	○	○
Su	○	○	○	○	○
SL	○	○	○	○	○

Description

- The format of floating point number of Fatek-PLC follows the IEEE-754 standard. For detail explanation of the format please refer to 5.3 (Numbering System) page 5-9.
- When compare control "EN" = 1 or changes from 0 to 1 (P instruction), compares S with upper limit SU and lower limit SL. If S is between the upper limit and the lower limit ($S_L \leqq S \leqq S_U$), then set the inside zone flag "INZ" to 1. If the value of S is greater than the upper limit S_U, then set the higher than upper limit flag "S>U" to 1. If the value of S is smaller then the lower limit S_L, then set the lower than lower limit flag "S<L" as 1.
- The upper limit S_U should be greater than the lower limit S_L. If S_U<S_L, then the limit value error flag "ERR" will set to 1, and this instruction will not carry out.



- The instruction at left compares the value of DR10 with the upper and lower limit zones formed by DR12 and DR14. If the values of DR10~DR14 are as shown in the diagram at bottom left, then the result can then be obtained as at the right of this diagram.
- If want to get the status of out side the zone, then OUT NOT Y0 may be used, or an OR operation between the two outputs S>U and S<L may be carried out, and move the result to Y0.

FUN 207 P FZCP	FLOATING POINT NUMBER ZONE COMPARE	FUN 207 P FZCP

S	DR10	2 0 0 0 . 2	Floating Point Number :	DR10	4 4 F A 0 6 6 6 H	
Su	DR12	3 0 0 0 . 3	Floating Point Number :	DR12	4 5 3 B 8 4 C D H	(Upper limit value)
SL	DR14	1 0 0 0 . 1	Floating Point Number :	DR14	4 4 7 A 0 6 6 6 H	(Lower limit value)

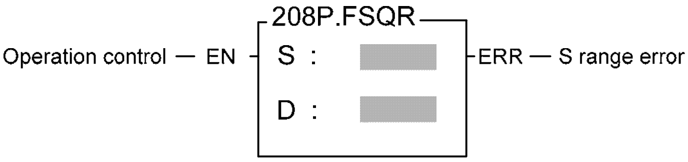
Before-execution

X0 = ↑ → FLOATING ZONE COMPARE → Y0 = $\boxed{1}$

Results of execution

FUN 208 P FSQR	FLOATING POINT NUMBER SQUARE ROOT	FUN 208 P FSQR

Ladder symbol



S : Source register to be taken square root

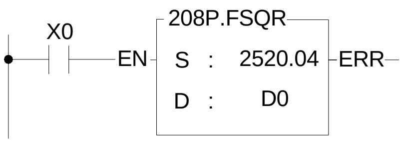
D : Register for storing result (square root value)

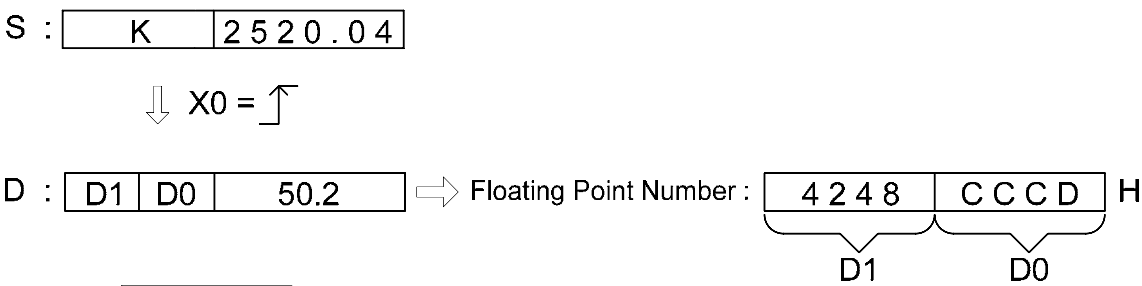
S, D may combine with V, Z, P0~P9 to serve indirect address application

Range / Operand	HR	ROR	DR	K	XR
	R0 | R3839	R5000 | R8071	D0 | D4095	Floating point number	V、Z P0～P9
S	○	○	○	○	○
D	○	○*	○		○

Description

- The format of floating point number of Fatek-PLC follows the IEEE-754 standard. For detail explanation of the format please refer to 5.3 (Numbering System) page 5-9.
- When operation control "EN" = 1 or from 0 to 1(P instruction), take the square root of the data specified by the S value or S~S+1 register, and store the result into the register specified by D~D+1.
- If the value of S is negative, then the error flag "ERR" will be set to 1, and do not execute the operation.

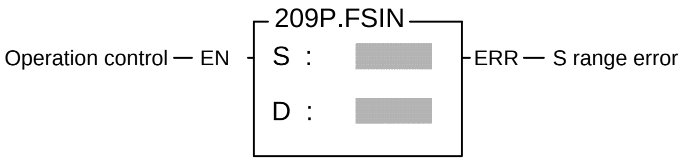




$$\sqrt{2520.04} = 50.2$$

FUN 209 P FSIN	SIN TRIGONOMETRIC INSTRUCTION	FUN 209 P FSIN

Ladder symbol



S : Source register to be taken SIN

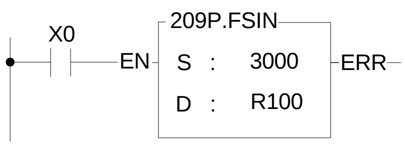
D : Register for storing result (SIN value)

S, D may combine with V, Z, P0~P9 to serve indirect address application.

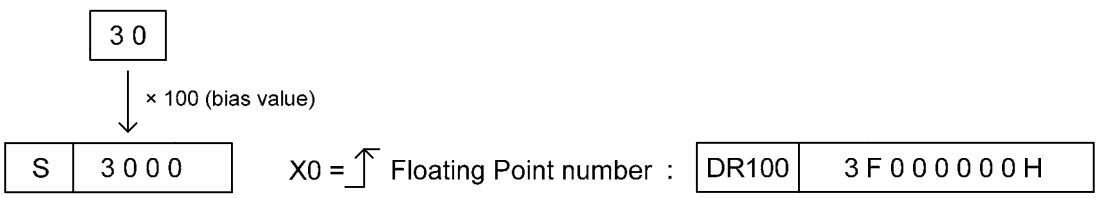
Range / Operand	HR R0 ∣ R3839	ROR R5000 ∣ R8071	DR D0 ∣ D4095	K Integer 16 Bit number	XR V、Z P0～P9
S	○	○	○	○	○
D	○	○*	○		○

Description

- The format of floating point number of Fatek-PLC follows the IEEE-754 standard. For detail explanation of the format please refer to 5.3 (Numbering System) page 5-9.
- When operation control "EN" = 1 or from 0 to 1 (P instruction), take the SIN value of the angle data specified by the S register and store the result into the register D~D+1 in floating point number format. The valid range of the angle is from –18000 to +18000, unit in 0.01 degree.
- If the S value is not within the valid range, then the S value error flag "ERR" will be set to 1, and do not execute the operation.



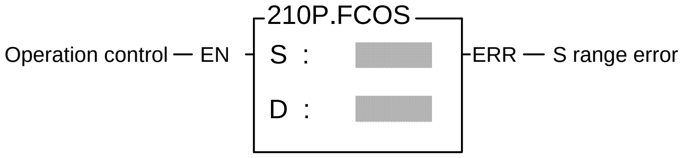
- At left, the example program gets the SIN value of 30, and stores the results the register DR100.



SIN(30) = 0.5

FUN 210 P FCOS	COS TRIGONOMETRIC INSTRUCTION	FUN 210 P FCOS

Ladder symbol



S : Source register to be taken COS

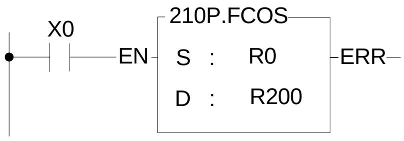
D : Register for storing result (COS value)

S, D may combine with V, Z, P0~P9 to serve indirect address application

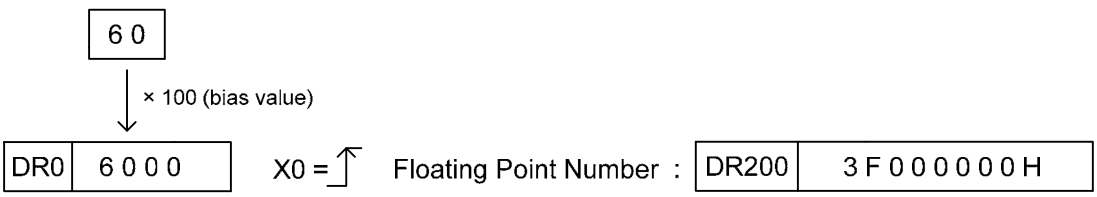
Range / Ope-rand	HR	ROR	DR	K	XR
	R0 | R3839	R5000 | R8071	D0 | D4095	Integer 16 Bit number	V、Z P0～P9
S	○	○	○	○	○
D	○	○*	○		○

Description

- The format of floating point number of Fatek-PLC follows the IEEE-754 standard. For detail explanation of the format please refer to 5.3 (Numbering System) page 5-9.
- When operation control "EN" = 1 or from 0 to 1 (P instruction), take the COS value of the angle data specified by the S register and store the result into the register D~D+1 in floating point number format. The valid range of the angle is from –18000 to +18000, unit in 0.01 degree.
- If the S value is not within the valid range, then the S value error flag "ERR" will be set to 1, and do not execute the operation.



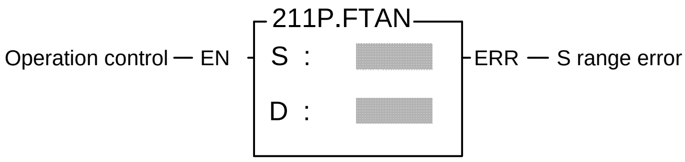
- At left, the example program gets the COS value of 60, and stores the results the register DR200.



COS(60) = 0.5

FUN 211 P FTAN	TAN TRIGONOMETRIC INSTRUCTION	FUN 211 P FTAN

Ladder symbol



S : Source register to be taken TAN

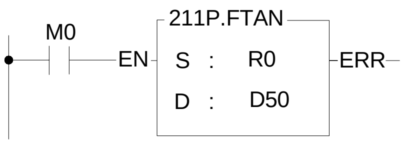
D : Register for storing result
(TAN value)

S, D may combine with V, Z, P0~P9 to serve indirect address application

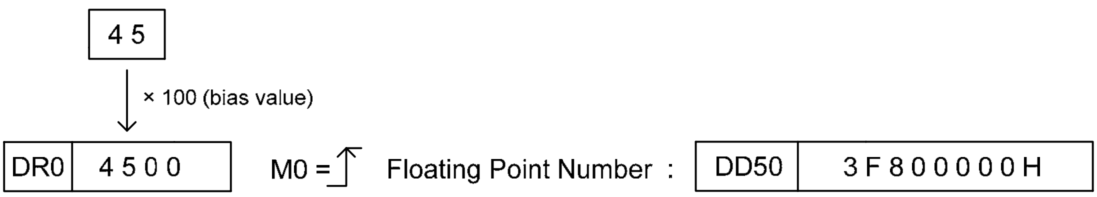
Range / Ope-rand	HR R0 ∣ R3839	ROR R5000 ∣ R8071	DR D0 ∣ D4095	K Integer 16 Bit number	XR V、Z P0～P9
S	○	○	○	○	○
D	○	○*	○		○

Description

- The format of floating point number of Fatek-PLC follows the IEEE-754 standard. For detail explanation of the format please refer to 5.3 (Numbering System) page 5-9.
- When operation control "EN" = 1 or from 0 to 1 (P instruction), take the COS value of the angle data specified by the S register and store the result into the register D~D+1 in floating point number format. The valid range of the angle is from –18000 to +18000, unit in 0.01 degree.
- If the S value is not within the valid range, then the S value error flag "ERR" will be set to 1, and do not execute the operation.



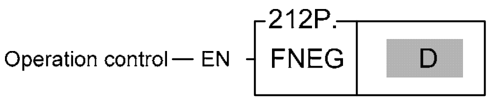
- At left, the example program gets the TAN value of 45, and stores the results the register DD50.



TAN(45) = 1

FUN 212 P FNEG	CHANGE SIGN OF THE FLOATING POINT NUMBER	FUN 212 P FNEG

Ladder symbol



D : Register to be changed sign

D may combine with V, Z, P0~P9 to serve indirect address application

Range / Ope-rand	HR	ROR	DR	XR
	R0 ∣ R3839	R5000 ∣ R8071	D0 ∣ D4095	V、Z P0～P9
D	○	○*	○	○

Description

- The format of floating point number of Fatek-PLC follows the IEEE-754 standard. For detail explanation of the format please refer to 5.3 (Numbering System) page 5-9.
- When operation control "EN" = 1 or from 0 to 1 (P instruction), the sign of the floating point number register specified by D will be toogled.

Programming Example

- The instruction at left negates the value of the DR0 register, and stores it back to DR0.

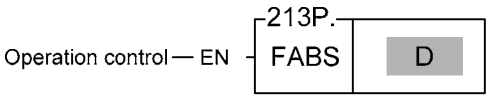
DR0	1 2 3 . 4 5	⇨ Floating Point Number :	DR0	4 2 F 6 E 6 6 6 H

⇩ (NEGATION) ⇩ X0 = ↑

DR0	- 1 2 3 . 4 5		DR0	C 2 F 6 E 6 6 6 H

FUN 213 P FABS	FLOATING POINT NUMBER ABSOLUTE VALUE	FUN 213 P FABS

Ladder symbol



D : Register to be taken absolute value

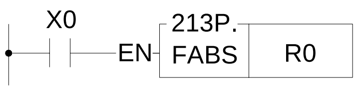
D may combine with V, Z, P0~P9 to serve indirect address application

Range / Ope-rand	HR	ROR	DR	XR
	R0 | R3839	R5000 | R8071	D0 | D4095	V、Z P0～P9
D	○	○*	○	○

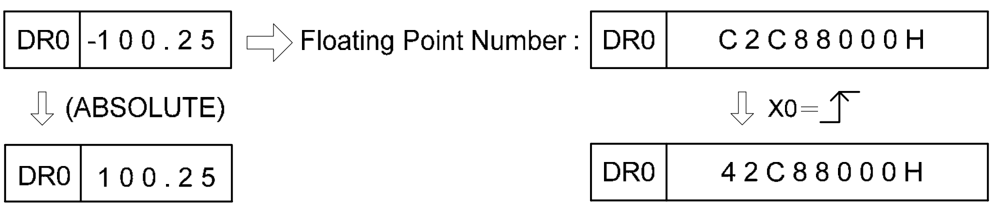
Description

- The format of floating point number of Fatek-PLC follows the IEEE-754 standard. For detail explanation of the format please refer to 5.3 (Numbering System) page 5-9.
- When operation control "EN" = 1 or from 0 to 1 (P instruction), calculate the absolute value of the floating point number register specified by D, and write it back into the original D register.

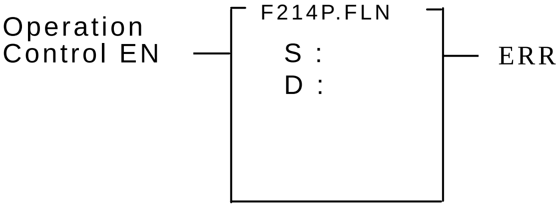
Programming Example



- The instruction at left calculates the absolute value of the DR0 register, and stores it back in DR0.



FUN 214 P FLN	FLOATING POINT NAPIERIAN LOGARITHM, $\log_e x$ or ln(x)	FUN 214 P FLN



S : Source data or register to be calculated Napierian logarithm value

D : Register for storing the result

S, D may combine with V, Z, P0~P9 to serve indirect address application

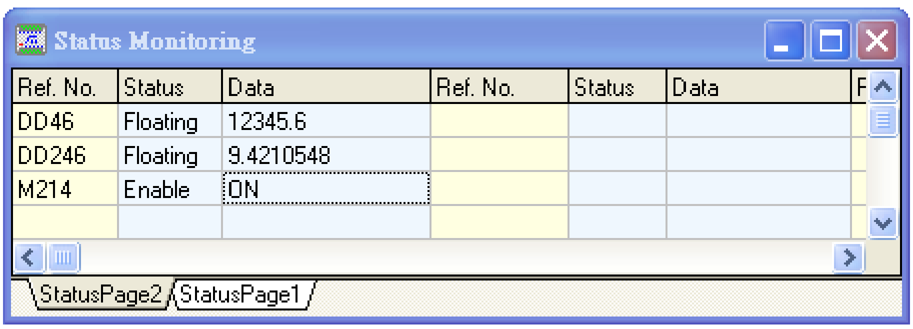
Range / Operand	HR R0 | R3839	ROR R5000 | R8071	DR D0 | D3999	K Floating number	XR V、Z P0～P9
S	○	○	○	○	○
D	○	○*	○		○

Description

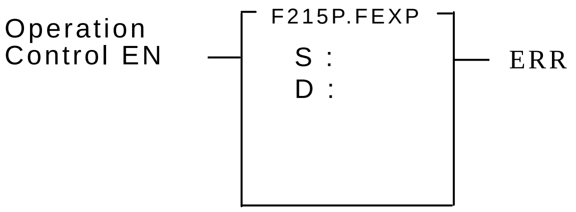
- The format of floating point number of Fatek-PLC follows the IEEE-754 standard of 32-bit.
- When operation control "EN" = 1 or from 0 to 1 (P instruction), take the Napierian logarithm of the data specified by the S value or S~S+1 register, and store the result into the register specified by D~D+1.
- If the value of S is negative or equal to 0、 invalid indirect addressing、 or over range of the result , the error flag "ERR" will be set to 1, and not update the value of D~D+1.
- All floating point instructions can't be executed in interrupt service routine.

Example

- When M214=1, calculate the Napierian logarithm value, it is DD246 = ln (DD46)



FUN 215 P FEXP	FLOATING POINT NATURE POWER FUNCTION, e^{x}	FUN 215 P FEXP



S : Source data or register to be calculated power function of nature number

D : Register for storing the result

S, D may combine with V, Z, P0~P9 to serve indirect address application

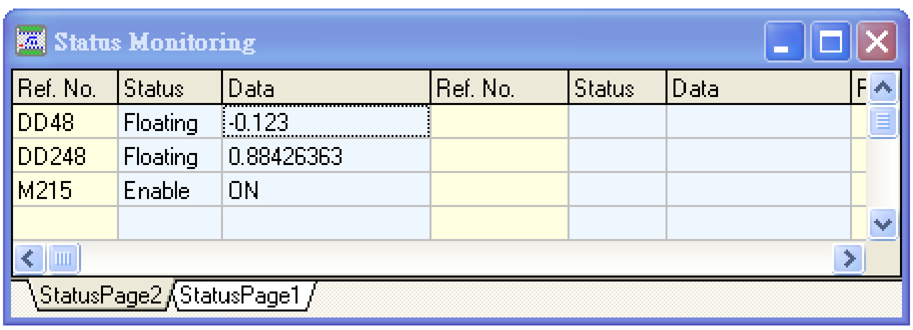
Range / Operand	HR R0 | R3839	ROR R5000 | R8071	DR D0 | D3999	K Floating number	XR V、Z P0～P9
S	○	○	○	○	○
D	○	○*	○		○

Description

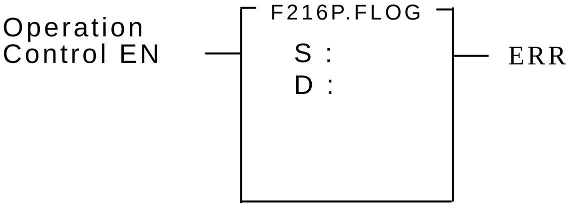
- The format of floating point number of Fatek-PLC follows the IEEE-754 standard of 32-bit.
- When operation control "EN" = 1 or from 0 to 1 (P instruction), calaulate the nature power function of the data specified by the S value or S~S+1 register, and store the result into the register specified by D~D+1.
- If the value of S is out of range、 invalid indirect addressing、 or over range of the result , the error flag "ERR" will be set to 1, and not update the value of D~D+1.
- All floating point instructions can't be executed in interrupt service routine.

Example

- When M215=1, calculate the nature power function, it is DD248 = e^{DD48}



FUN 216 P FLOG	FLOATING POINT LOGARITHM, $\log_{10}x$ or log(x)	FUN 216 P FLOG



S : Source data or register to be calculated logarithm value

D : Register for storing the result

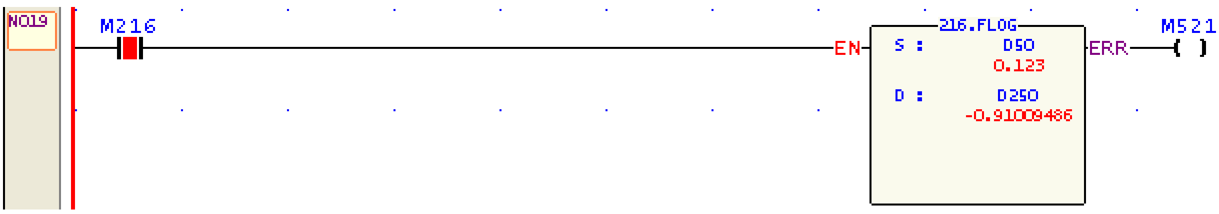
S, D may combine with V, Z, P0~P9 to serve indirect address application

Range / Operand	HR	ROR	DR	K	XR
	R0 ∣ R3839	R5000 ∣ R8071	D0 ∣ D3999	Floating number	V、Z P0～P9
S	○	○	○	○	○
D	○	○*	○		○

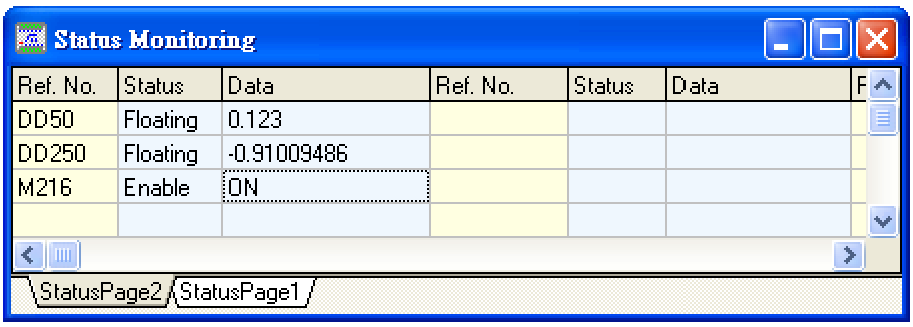
Description

- The format of floating point number of Fatek-PLC follows the IEEE-754 standard of 32-bit.
- When operation control "EN" = 1 or from 0 to 1 (P instruction), calculate the logarithm value of the data specified by the S value or S~S+1 register, and store the result into the register specified by D~D+1.
- If the value of S is negative or equal to 0、 invalid indirect addressing、 or over range of the result , the error flag "ERR" will be set to 1, and not update the value of D~D+1.
- All floating point instructions can't be executed in interrupt service routine.

Example



- When M216=1, calculate the logarithm value, it is DD250 = log (DD50)



Ref. No.	Status	Data	Ref. No.	Status	Data
DD50	Floating	0.123			
DD250	Floating	-0.91009486			
M216	Enable	ON			

FUN 217 P FPOW	FLOATING POINT POWER FUNCTION, x^y	FUN 217 P FPOW

Operation Control EN — F217P.FPOW — ERR

Sy :
Sx :
D :

Sy: Source data or register of exponential

SX: Source data or register of base。

D : Register for storing the result

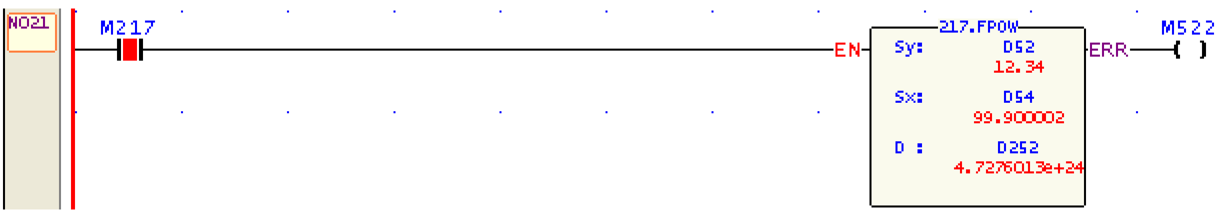
Sy, Sx, D may combine with V, Z, P0~P9 to serve indirect address application

Range / Operand	HR	ROR	DR	K	XR
	R0 | R3839	R5000 | R8071	D0 | D3999	Floating number	V、Z P0～P9
Sy	○	○	○	○	○
Sx	○	○*	○	○	○
D	○	○	○		○

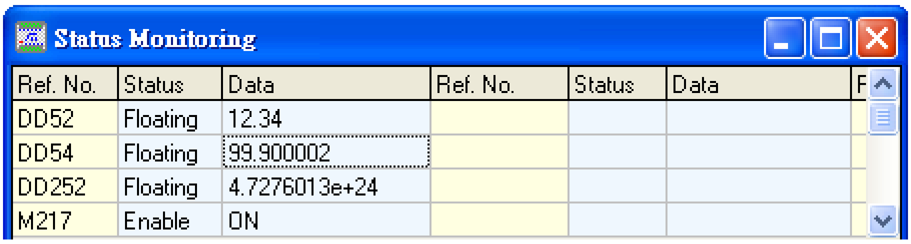
Description

- The format of floating point number of Fatek-PLC follows the IEEE-754 standard of 32-bit.
- When operation control "EN" = 1 or from 0 to 1 (P instruction), calculate the power function of the exponential data specified by the Sy、base data specified by the Sx, and store the result into the register specified by D~D+1.
- If it exists invalid indirect addressing、or over range of the result , the error flag "ERR" will be set to 1, and not update the value of D~D+1.
- All floating point instructions can't be executed in interrupt service routine.

Example



- When M217=1, calculate the power function, it is DD252 = $DD54^{DD52}$


Status Monitoring

Ref. No.	Status	Data	Ref. No.	Status	Data
DD52	Floating	12.34			
DD54	Floating	99.900002			
DD252	Floating	4.7276013e+24			
M217	Enable	ON			

FUN 218 P FASIN	FLOATING POINT ARC SINE FUNCTION, $\sin^{-1}$	FUN 218 P FASIN

Operation Control EN — F218P.FASIN (S : , D :) — ERR

S : Source data or register to be calculated the arc sine value

D : Register for storing the result

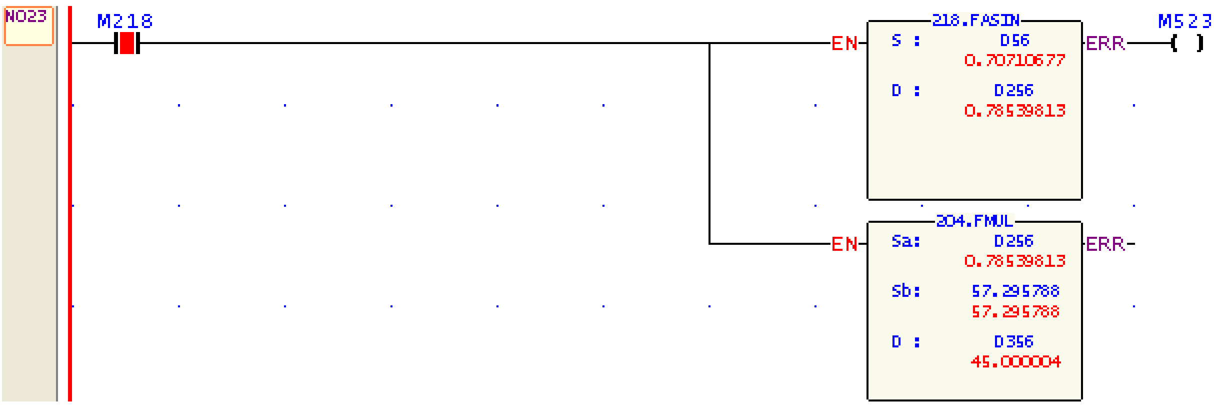
S, D may combine with V, Z, P0~P9 to serve indirect address application

Range / Operand	HR	ROR	DR	K	XR
	R0 ∣ R3839	R5000 ∣ R8071	D0 ∣ D3999	Floating number	V、Z P0～P9
S	○	○	○	○	○
D	○	○*	○		○

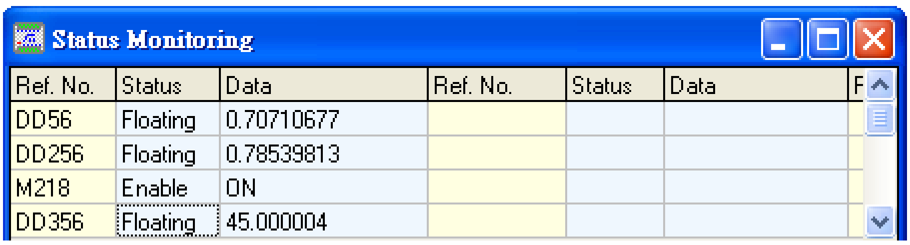
Description

- The format of floating point number of Fatek-PLC follows the IEEE-754 standard of 32-bit.
- When operation control "EN" = 1 or from 0 to 1 (P instruction), calculate the arc sine value of the data specified by the S value or S~S+1 register, and store the result into the register specified by D~D+1.
- Range of S data : -1~ +1 ; range of D value : $-\pi/2$ ~ $\pi/2$ (Unit in radian)
- If the value of S is out of range、or invalid indirect addressing, the error flag "ERR" will be set to 1, and not update the value of D~D+1.
- All floating point instructions can't be executed in interrupt service routine.

Example



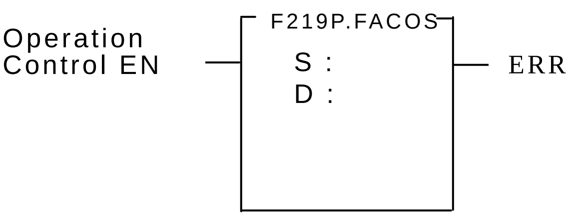
- When M218=1, calculate the arc sine value, it is DD256 = $\sin^{-1}$ DD56; DD256(Unit in radian) × 57.295788(180/π) to acquire the degree value



Status Monitoring

Ref. No.	Status	Data	Ref. No.	Status	Data
DD56	Floating	0.70710677			
DD256	Floating	0.78539813			
M218	Enable	ON			
DD356	Floating	45.000004			

FUN 219 P FACOS	FLOATING POINT ARC COSINE FUNCTION, $\cos^{-1}$	FUN 219 P FACOS



S : Source data or register to be calculated the arc cosine value

D : Register for storing the result

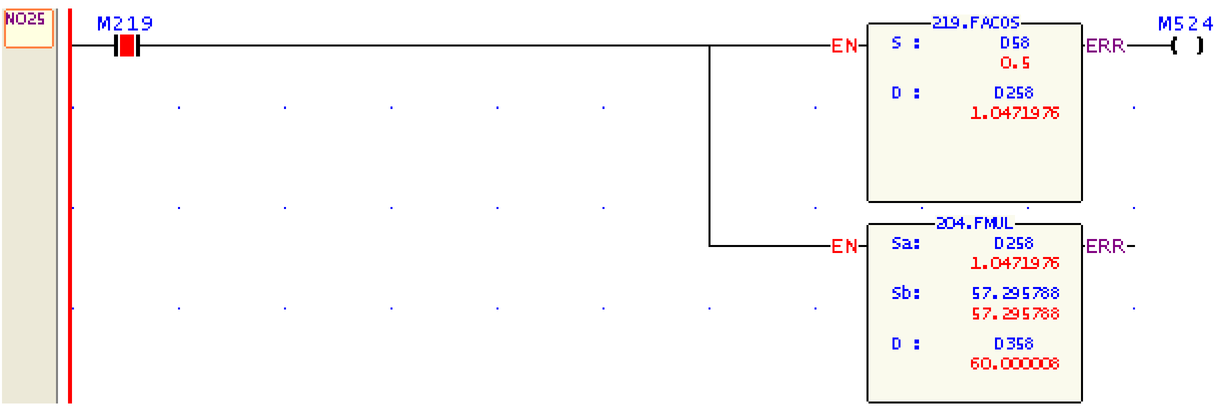
S, D may combine with V, Z, P0~P9 to serve indirect address application

Range / Operand	HR R0 ∣ R3839	ROR R5000 ∣ R8071	DR D0 ∣ D3999	K Floating number	XR V、Z P0～P9
S	○	○	○	○	○
D	○	○*	○		○

Description

- The format of floating point number of Fatek-PLC follows the IEEE-754 standard of 32-bit.
- When operation control "EN" = 1 or from 0 to 1(P instruction), calculate the arc cosine value of the data specified by the S value or S~S+1 register, and store the result into the register specified by D~D+1.
- Range of S data : -1~ +1 ; range of D value : 0 ~ π (Unit in radian)
- If the value of S is out of range、or invalid indirect addressing, the error flag "ERR" will be set to 1, and not update the value of D~D+1.
- All floating point instructions can't be executed in interrupt service routine.

Example

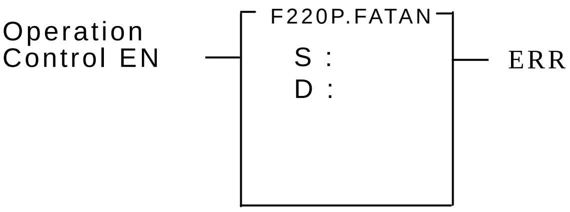


- When M219=1, calculate the arc cosine value, it is DD258 = $\cos^{-1}$ DD58; DD258(Unit in radian) × 57.295788(180/π) to acquire the degree value

Status Monitoring

Ref. No.	Status	Data	Ref. No.	Status	Data
DD58	Floating	0.5			
DD258	Floating	1.0471976			
M219	Enable	ON			
DD358	Floating	60.000008			

FUN 220 P FATAN	FLOATING POINT ARC TANGENT FUNCTION, $\tan^{-1}$	FUN 220 P FATAN



S : Source data or register to be calculated the arc tangent value

D : Register for storing the result

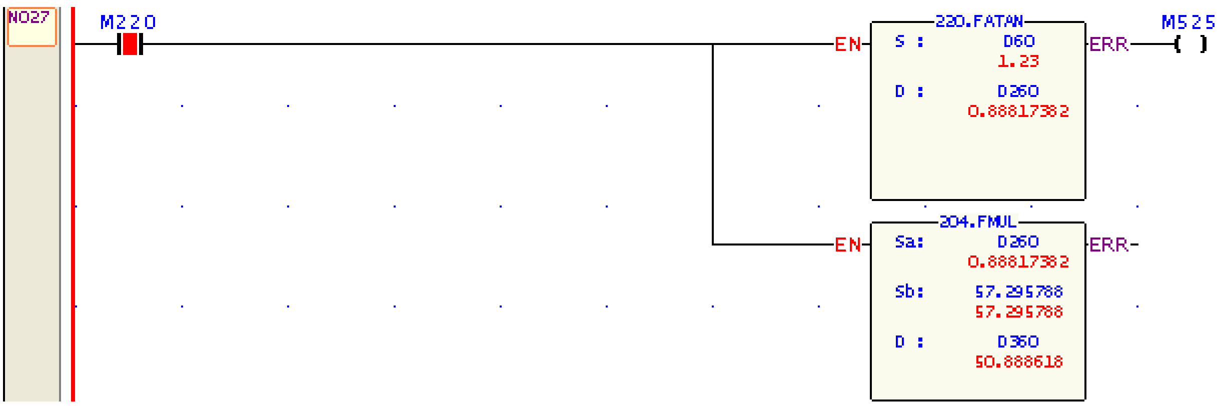
S, D may combine with V, Z, P0~P9 to serve indirect address application

Range / Ope-rand	HR	ROR	DR	K	XR
	R0 ∣ R3839	R5000 ∣ R8071	D0 ∣ D3999	Floating number	V、Z P0～P9
S	○	○	○	○	○
D	○	○*	○		○

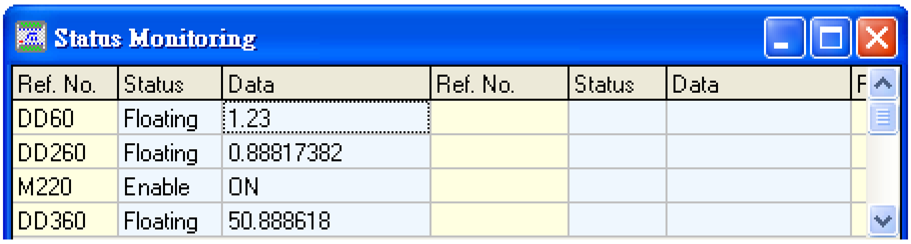
Description

- The format of floating point number of Fatek-PLC follows the IEEE-754 standard of 32-bit.
- When operation control "EN" = 1 or from 0 to 1 (P instruction), calculate the arc tangent value of the data specified by the S value or S~S+1 register, and store the result into the register specified by D~D+1.
- S data is any number ; range of D value : $-\pi/2 \sim \pi/2$ (Unit in radian)
- If it exists invalid indirect addressing, the error flag "ERR" will be set to 1, and not update the value of D~D+1.
- All floating point instructions can't be executed in interrupt service routine.

Example



- When M220=1, calculate the arc tangent value, it is DD260 = $\tan^{-1}$ DD60; DD260(Unit in radian) × 57.295788($180/\pi$) to acquire the degree value


Status Monitoring

Ref. No.	Status	Data	Ref. No.	Status	Data
DD60	Floating	1.23			
DD260	Floating	0.88817382			
M220	Enable	ON			
DD360	Floating	50.888618			