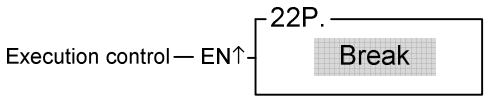
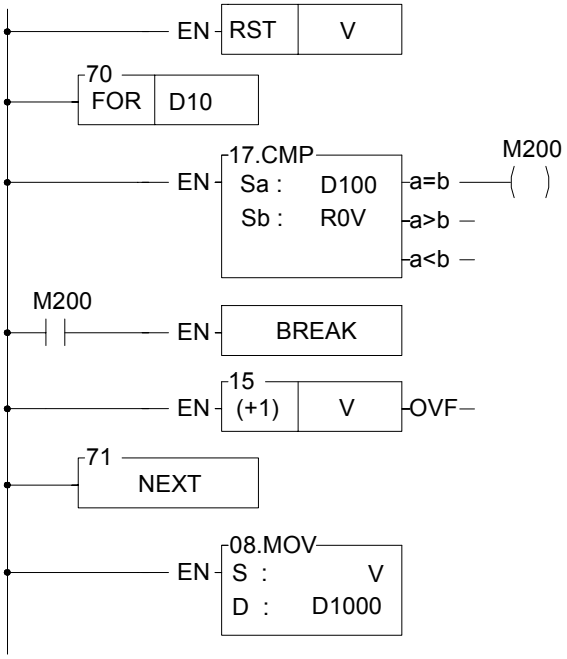
Chapter 7 Advanced Function Instructions

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

Ladder symbol



- When execution control〝EN〞=1 or〝EN↑〞（ P instruction）changes from 0→1，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。

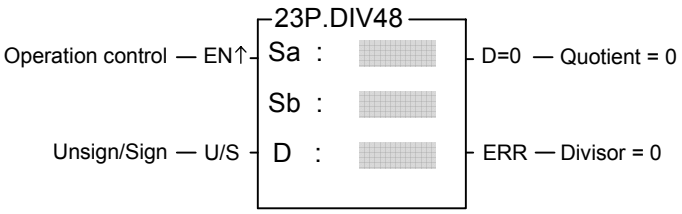


Description：The loop count used to execute the FOR and NEXT program loop is assigned by register D10；the program within the FOR and NEXT loop is designed to search the same data storing in D100 from the register table starting at R0。If it finds，the searching loop will be terminated and then it goes to 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 D1000 is the pointer of searching。

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

Ladder symbol



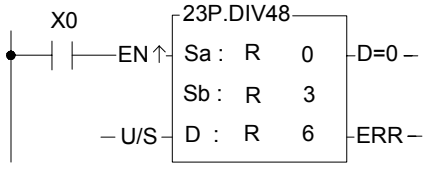
Sa：Starting register of dividend
Sb：Starting register of divisor
D : Starting register for storing the division result (quotient)
Sa、Sb、can combine V, Z, P0~P9 for index addressing.

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

- When operation control “EN”=1 or “EN ↑ ” (P instruction) changes from 0→1, 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

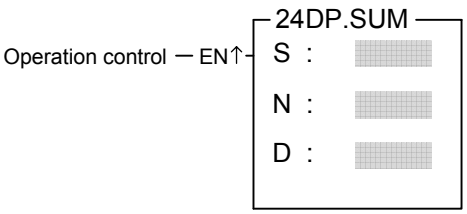
R5	R4	R3
1234567		

R8	R7	R6
1739		

Quotient

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

Ladder symbol



S : Starting number of source register

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

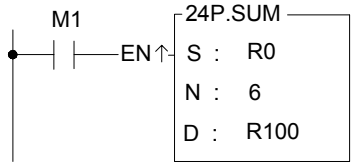
D : The register which stored the result (summation)

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 “EN ↑ ” (P instruction) changes from 0→1, 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 or port2 can be used to serve as a general purpose ASCII communication interface. If the data error detecting method is Check-Sum, 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.

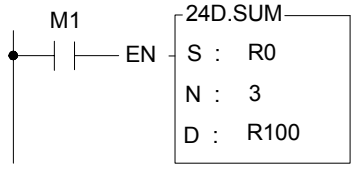


- The left illustrates that 6 16-bit registers starting from R0 is calculated for summation, and the result is stored into the R100 register.

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.



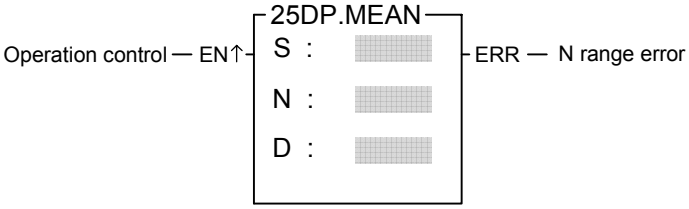
- The left illustrates that three 32-bit registers starting from DR0, is calculated for their summation, and the result is stored into the DR100 register.

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

Ladder symbol



S : Source register number

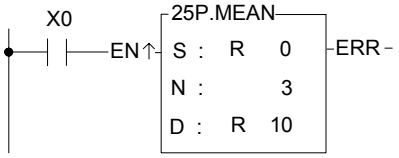
N : Number of registers to be averaged
(N units of successive registers starting from S)

D : Register number for storing result (mean value)

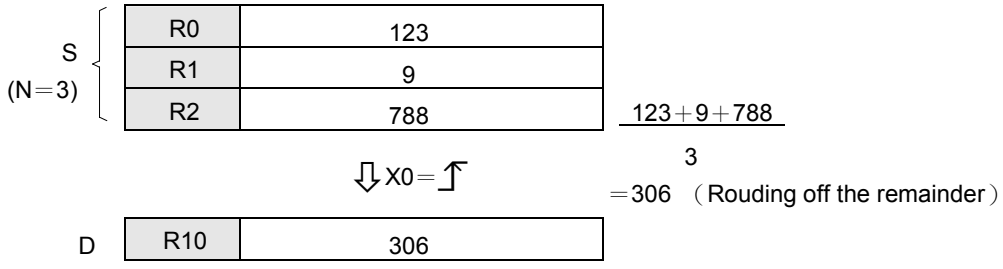
The S, N, D may combine with V, Z, P0~P9 to serve indirect address application

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 "EN ↑" (P instruction) from 0 to 1, 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



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

Ladder symbol

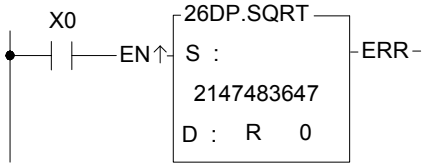
S : Source register to be taken square root

D : Register for storing result (square root value)

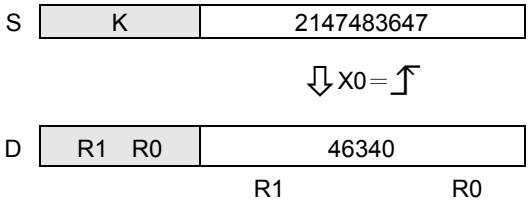
S, D may combine with V, Z, P0~P9 to serve indirect address application

Range / Operand	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 "EN ↑ " (P instruction) from 0 to 1, 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.



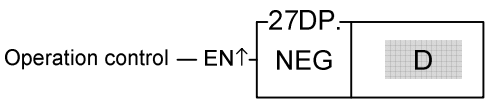
$$\sqrt{2147483647} = 46340.\underline{\underline{95}}$$

↑

rounding off

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

Ladder symbol

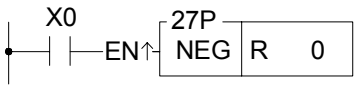


D : Register to be negated

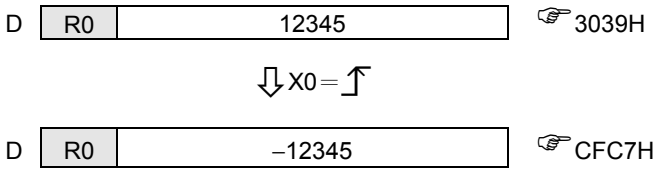
D may combine with V, Z, P0~P9 to serve indirect address application

Operand \ Range	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 "EN ↑ " (P instruction) from 0 to 1, 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.

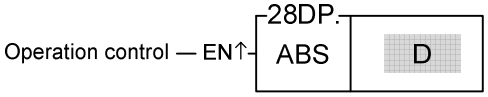


- The instruction at left negates the value of the R0 register, and stores it back to R0.



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

Ladder symbol

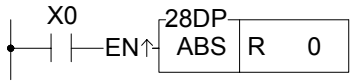


D : Register to be taken absolute value

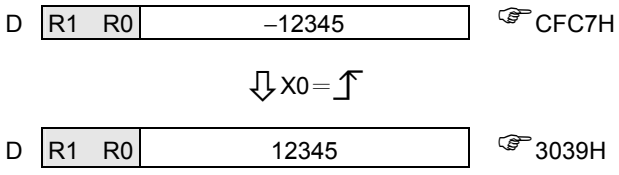
D may combine with V, Z, P0~P9 to serve indirect address application

Operand \ Range	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 "EN ↑ " (P instruction) from 0 to 1, calculate the absolute value of the content of the register specified by D, and write it back into the original D register.

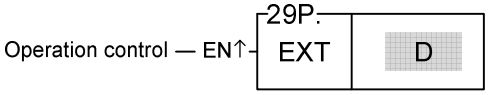


- The instruction at left calculates the absolute value of the R0 register, and stores it back in R0.



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

Ladder symbol



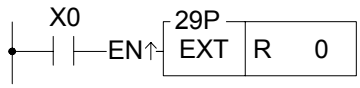
D : Register to be taken sign extension

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

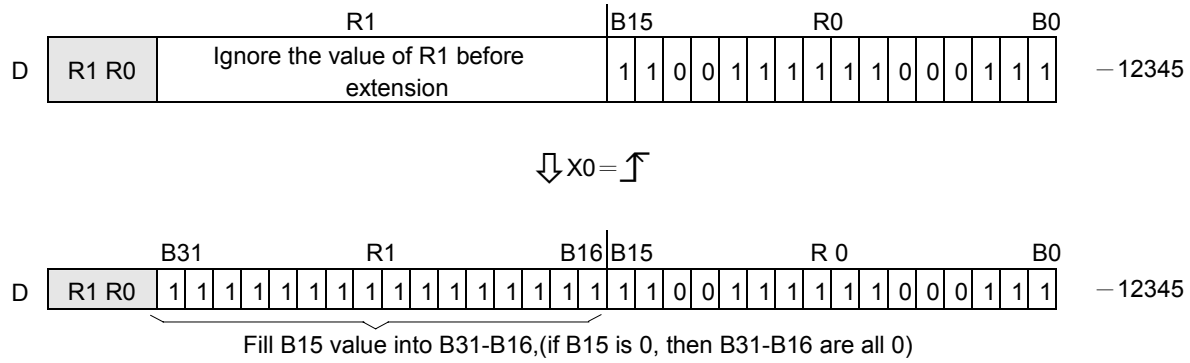
Range / Operand	WY	WM	WS	TMR	CTR	HR	OR	SR	ROR	DR	XR
	WY0 | WY240	WM0 | WM1896	WS0 | WS984	T0 | T255	C0 | C255	R0 | R3839	R3904 | R3967	R3968 | R4167	R5000 | R8071	D0 | D4095	V、Z P0~P9
D	○	○	○	○	○	○	○	○*	○*	○	○

- When operation control "EN" = 1 or "EN ↑ " (P instruction) from 0 to 1, 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.



- 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

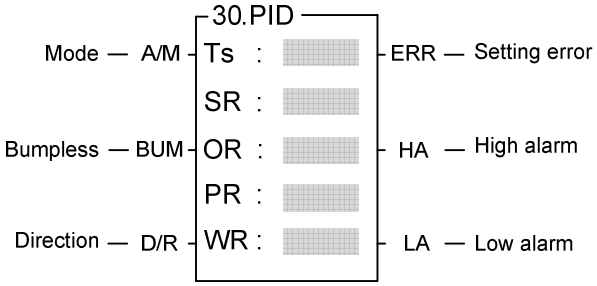


Before extension（16 bits） R0= CFC7H=－12345
After extension（32 bits）R1R0=FFFFCFC7H=－12345

} 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 according to the current value of process variable (PV) derived from the external analog signal and the setting value (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:

$$Mn = [(D4005/Pb) \times En] + \sum_{0}^{n} [(D4005/Pb) \times Ti \times Ts \times En] - [(D4005/Pb) \times Td \times (PVn - PVn\text{-}1)/Ts] + Bias$$

Mn : Control output at time "n"

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

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

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

PVn : 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" (PVn)

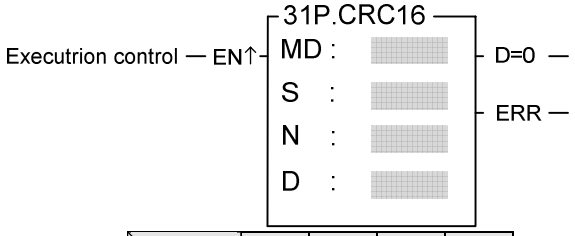
Ts : Interval time of PID calculation (range: 1~3000, unit : 0.01 S)

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

- For detail description of this function, please refer chapter 20.

FUN31 P CRC16	CRC16 CALCULATION （CRC16）	FUN31 P CRC16

Ladder symbol



Operand \ Range	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

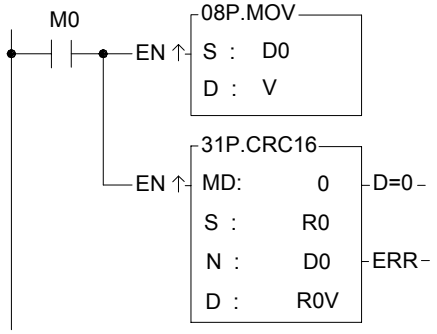
S：Starting address of CRC16 calculation

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 "EN↑" （ P instruction）changes from 0→1, 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 fromat, 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

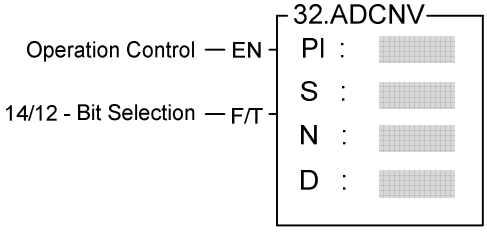
	High Byte	Low Byte
R0	Don't care	Byte-0
R1	Don't care	Byte-1
R2	Don't care	Byte-2
R3	Don't care	Byte-3
R4	Don't care	Byte-4
R5	Don't care	Byte-5
R6	Don't care	Byte-6
R7	Don't care	Byte-7
R8	Don't care	Byte-8
R9	Don't care	Byte-9

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

Ladder symbol



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

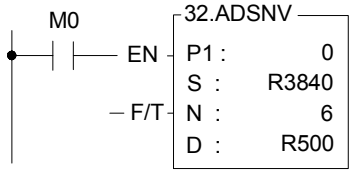
N：Quantity of conversion (In Word)

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 4～20mA, the analog input module is one of the solution to get this kind of signal, but the input span of the analog input module is 0～20mA (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.
- This instruction will not act if invalid length of N.
- When the input〝F/T〞=0, it assigns the 12-bit analog input module; while〝F/T〞=1, it assigns the 14-bit analog input module.

Example：

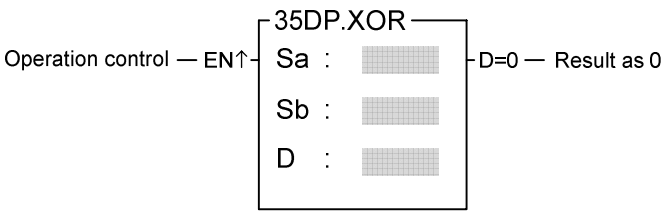


Description：When M0 is ON, it will perfom 6 points of conversion starting from R3840, where the offset of 4～20mA raw reading value will be eliminated, and the corresponding value 0～4095 will be stored into R500～R505.

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)

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 "EN ↑ " (P instruction) changes from 0 to 1, 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.

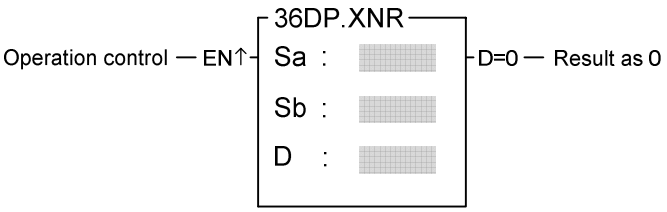
Sa	R0	1	0	1	1	1	0	1	1	0	1	1	0	1	1	0	1
Sb	R1	1	1	1	0	1	1	1	0	1	0	1	0	0	1	1	0

⇩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

Ladder symbol



Sa : Data a for XNR operation

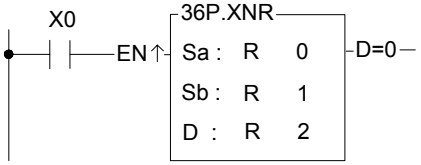
Sb : Data b for XNR operation

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 "EN ↑ " (P instruction) changes from 0 to 1, 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.

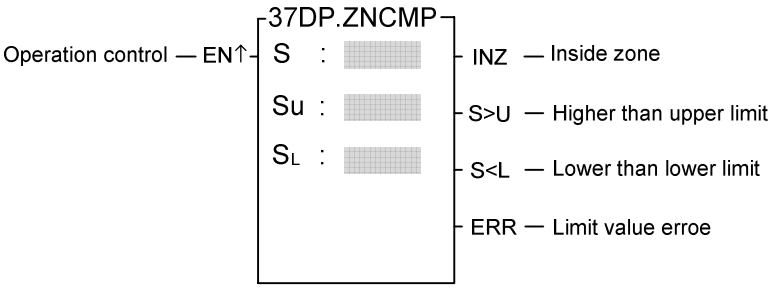
Sa	R0	1	0	1	1	1	0	1	1	0	1	1	0	1	1	0	1
Sb	R1	1	1	1	0	1	1	1	0	1	0	1	0	0	1	1	0

⇩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

Ladder symbol



S : Register for zone comparison

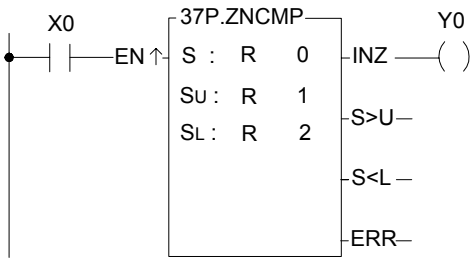
SU : The upper limit value

SL : The lower limit value

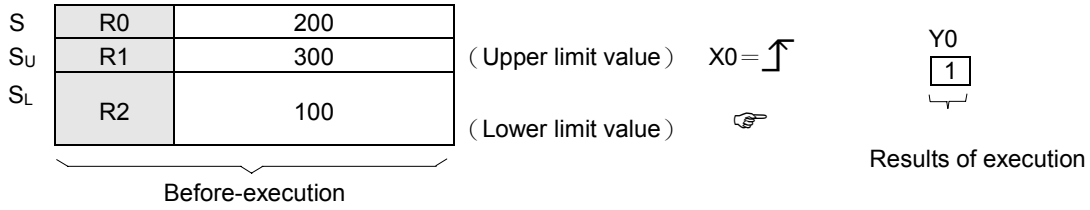
S, SU, SL may combine with V, Z, P0~P9 to serve indirect address application

Range / 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	○	○	○	○	○	○	○	○	○	○	○	○		○
SU	○	○	○	○	○	○	○	○	○	○	○	○	○	○
SL	○	○	○	○	○	○	○	○	○	○	○	○	○	○

- When operation control "EN" = 1 or "EN ↑ " (P instruction) changes from 0 to 1, 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.

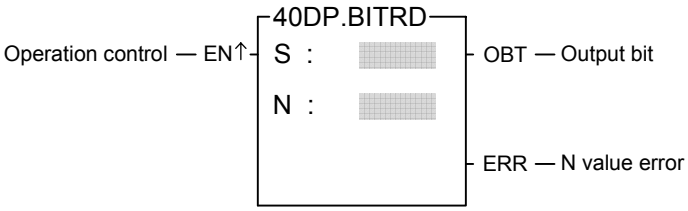


- The instruction at left compares the value of R0 with the upper and lower limit zones formed by R1 and R2. If the values of R0~R2 are as shown in the diagram at bottom left, then the result can then be obtained as at the right of this diagram.
- If want to get the status of out side the zone, then 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.



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

Ladder symbol



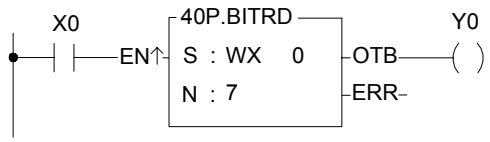
S : Source data to be read

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

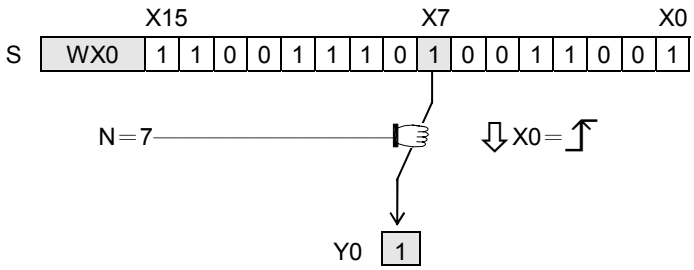
S, N may combine with V, Z, P0~P9 to serve indirect address application

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 "EN ↑ " (P instruction) changes from 0 to 1, take the Nth bit of the S data out , and put it to the output bit "OTB".
- When read control "EN" = 0 or "EN ↑ " (P instruction) is not change from 0 to 1, 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.

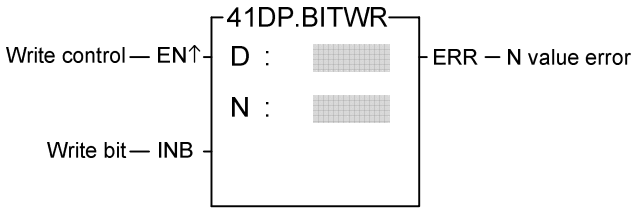


- The instruction at left reads the 7th bit (X7) status from WX0 (X0~X15) and output to Y0. The results are as follows:



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

Ladder symbol



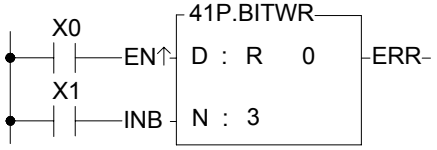
D : Register for bit write

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

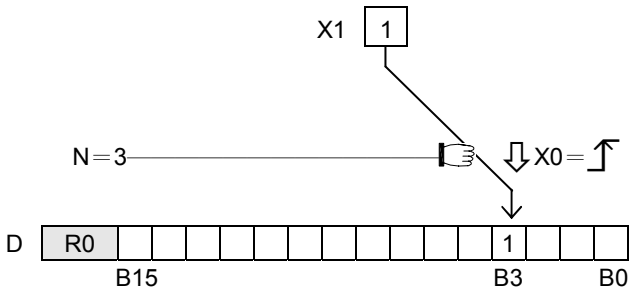
D, N may combine with V, Z , P0~P9 to serve indirect address application

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 or 15 | 0 or 31	V、Z P0~P9
D		○	○	○	○	○	○		○	○*	○*	○		○
N	○	○	○	○	○	○	○	○	○	○	○	○	○	○

- When write control "EN" = 1 or "EN ↑ " (P instruction) changes from 0 to 1, 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.



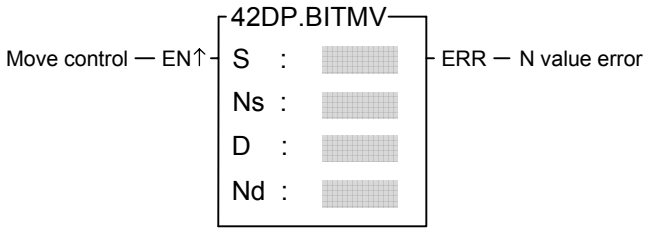
- The instruction at left writes the status of the write bit INB into B3 of R0. Assuming X1 = 1, the result will be as follows:



Bits other than B3 remain unchanged

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

Ladder symbol



S : Source data to be moved

Ns : Assign Ns bit within S as source bit

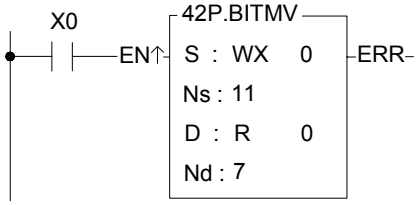
D : Destination register to be moved

Nd : Assign Nd bit within D as target bit

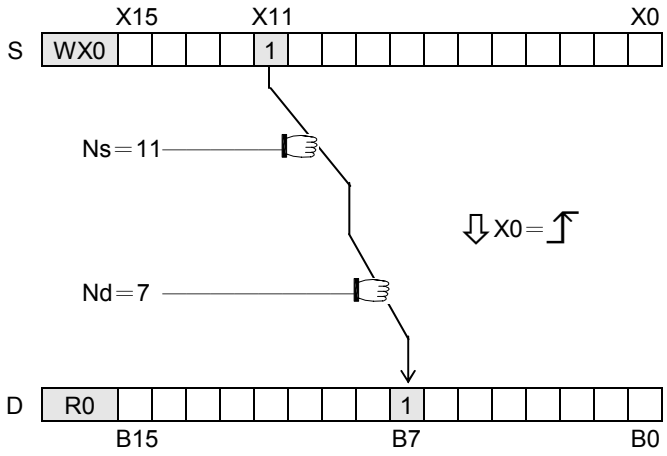
S, Ns, D, Nd may combine with V, Z, P0~P9 to serve indirect address application

Range / 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	○	○	○	○	○	○	○	○	○	○	○	○	○	○
Ns	○	○	○	○	○	○	○	○	○	○	○	○	0～31	○
D		○	○	○	○	○	○		○	○*	○*	○		○
Nd	○	○	○	○	○	○	○	○	○	○	○	○	0～31	○

- When move control "EN" = 1 or "EN ↑ " (P instruction) changes from 0 to 1, 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.

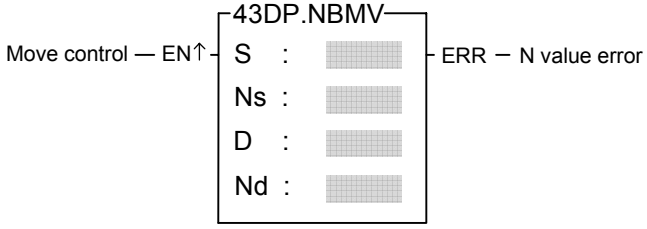


- The instruction at left moves the status of B11 (X11) within S into the B7 position within D. Except bit B7, other bits within D does not change.



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

Ladder symbol



S : Source data to be moved

Ns: Assign Ns nibble within S as source nibble

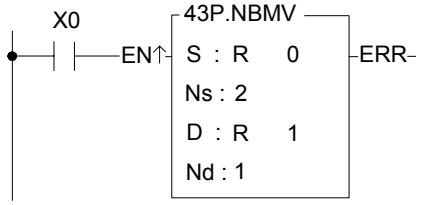
D : Destination register to be moved

Nd: Assign Nd nibble within D as target nibble

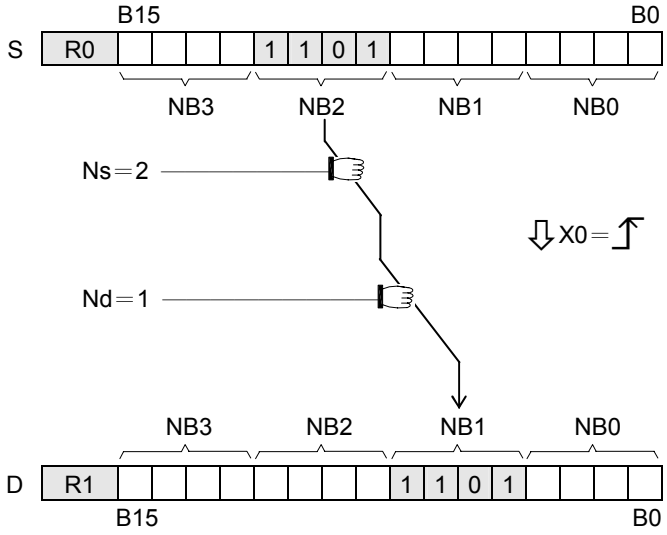
S, Ns, D, Nd may combine with V, Z, P0~P9 to serve indirect address application

Range / 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	○	○	○	○	○	○	○	○	○	○	○	○	○	○
Ns	○	○	○	○	○	○	○	○	○	○	○	○	0～7	○
D		○	○	○	○	○	○		○	○*	○*	○		○
Nd	○	○	○	○	○	○	○	○	○	○	○	○	0～7	○

- When move control "EN" = 1 or "EN ↑ " (P instruction)has a transition from 0 to 1, 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.

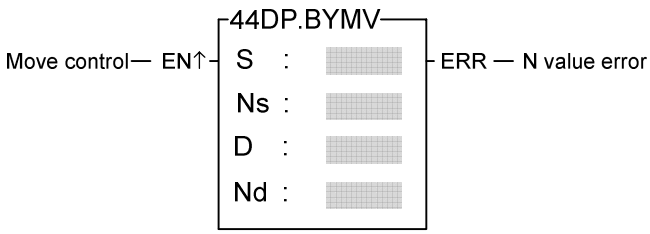


- The instruction at left moves the third nibble NB2 (B8~B11) within S to the first nibble NB1 (B4~B7) within D. Other nibbles within D remain unchanged.



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

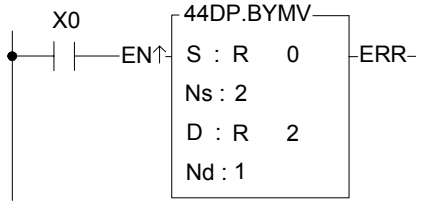
D : Destination register to be moved

Nd : Assign Nd byte within D as target byte

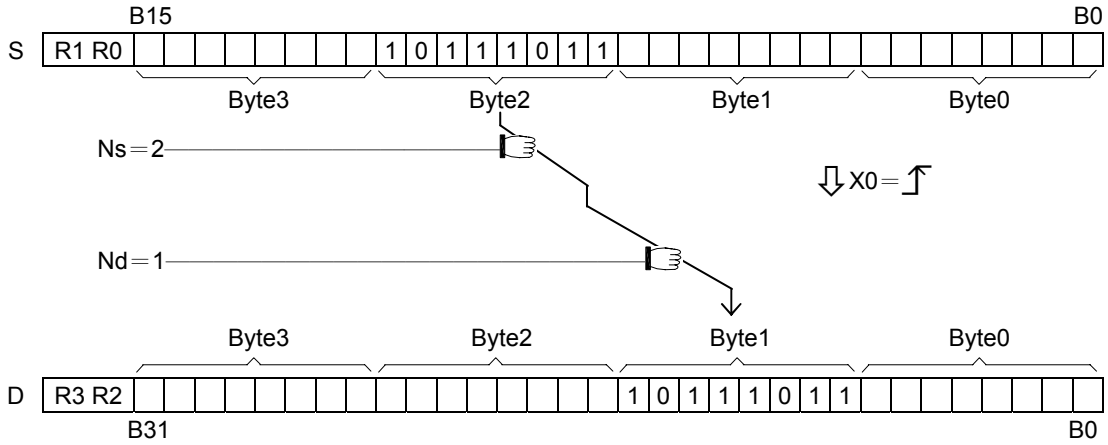
S, Ns, D, Nd may combine with V, Z, P0~P9 to serve indirect address application

Range / 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	○	○	○	○	○	○	○	○	○	○	○	○	○	○
Ns	○	○	○	○	○	○	○	○	○	○	○	○	0～3	○
D		○	○	○	○	○	○		○	○*	○*	○		○
Nd	○	○	○	○	○	○	○	○	○	○	○	○	0～3	○

- When move control "EN" = 1 or "EN ↑ " (P instruction) has a transition from 0 to 1, 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.

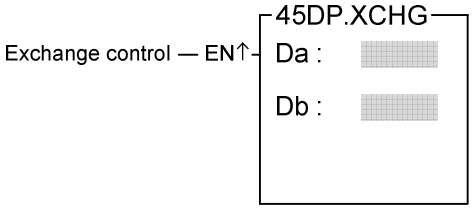


- The instruction at left moves the third byte (B16~B23) within S (32 bit register composed of R1R0), to the first byte within D (32 bit register composed of R3R2). Other bytes within D remain unchanged.



FUN 45 D P XCHG	EXCHANGE	FUN 45 D P XCHG

Ladder symbol



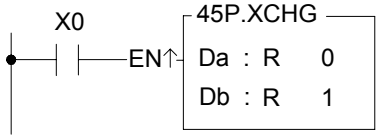
Da : Register a to be exchanged

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 "EN ↑ " (P instruction) has a transition from 0 to 1, 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.

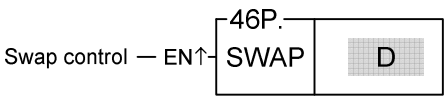
		B15															B0
Da	R0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Db	R1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

⇩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

Ladder symbol

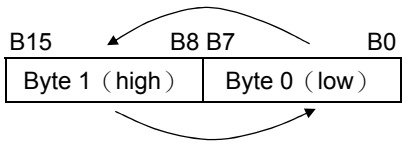


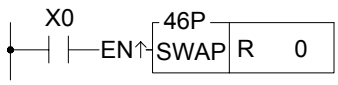
D : Register for byte data swap

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

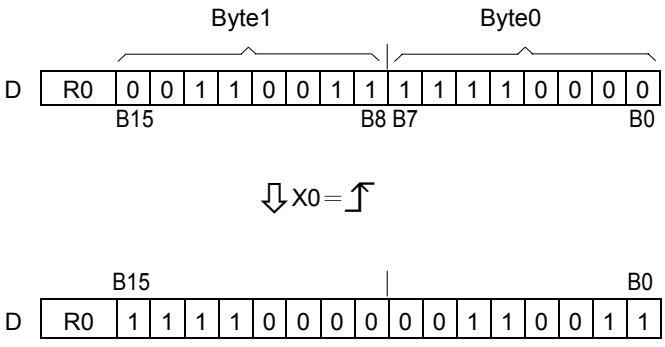
Range / Operand	WY	WM	WS	TMR	CTR	HR	OR	SR	ROR	DR	XR
	WY0 | WY240	WM0 | WM1896	WS0 | WS984	T0 | T255	C0 | C255	R0 | R3839	R3904 | R3967	R3968 | R4167	R5000 | R8071	D0 | D4095	V、Z P0~P9
D	○	○	○	○	○	○	○	○*	○*	○	○

- When swap control "EN" = 1 or "EN ↑ " (P instruction) has a transition from 0 to 1, 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.



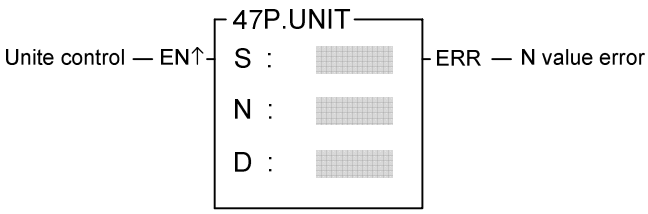


- The instruction at left swaps the data of the low byte (B0~B7) and the high byte (B8~B15) in R0. The results are as follows:



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

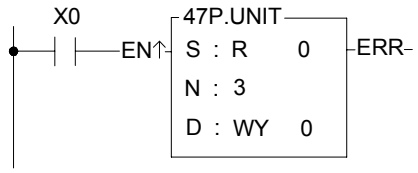
D : Registers storing united data

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

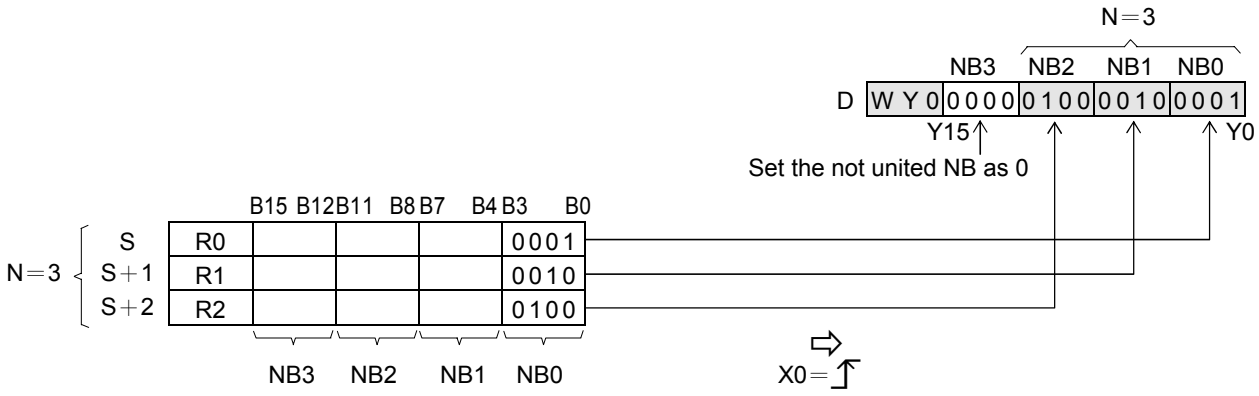
Range / Operand	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR
	WX0 | WX240	WY0 | WY240	WM0 | WM1896	WS0 | WS984	T0 | T255	C0 | C255	R0 | R3839	R3840 | R3903	R3904 | R3967	R3968 | R4167	R5000 | R8071	D0 | D4095	1 | 4	V、Z P0~P9
S	○	○	○	○	○	○	○	○	○	○	○	○		○
N	○	○	○	○	○	○	○	○	○	○	○	○	○	○
D		○	○	○	○	○	○		○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

- When unite control "EN" = 1 or "EN ↑ " (P instruction) has a transition from 0 to 1, 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.

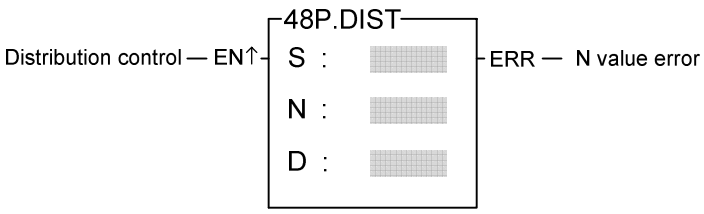


- The instruction at left takes out NB0 from 3 registers, R0, R1 and R2, and fills them into NB0~NB2 within WY0 register.



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

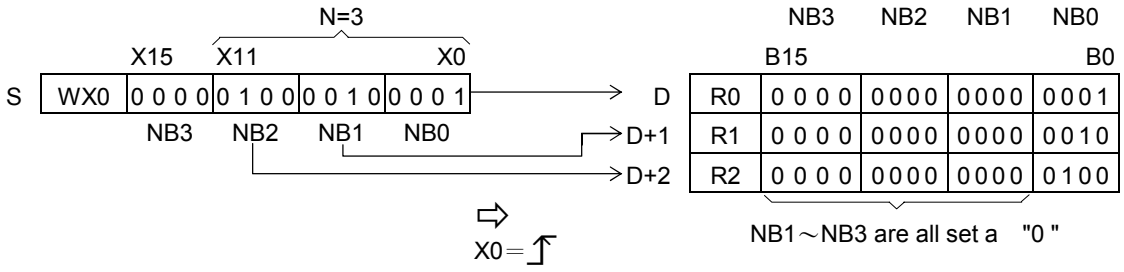
S, N, D may combine with V, Z, P0~P9 to serve indirect address application

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 "EN ↑" (P instruction) has a transition from 0 to 1, 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...)

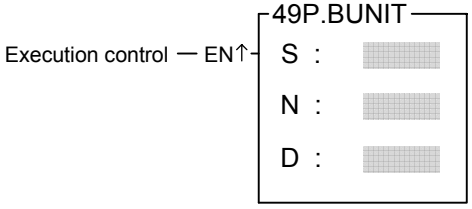
- This instruction only provides WORD (16 bit) operand. Therefore there are usually only 4 nibbles can be involved, so the effective value of N is 1~4. Beyond this range, will set the N value error flag "ERR" to 1, and do not carry out this instruction.

- 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

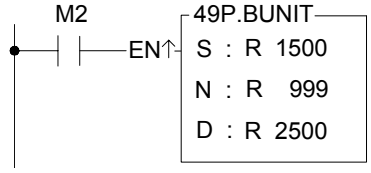
D ：Registers to store the united data

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

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

- When execution control "EN"=1 or "EN↑" (P instruction) changes from 0→1, 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

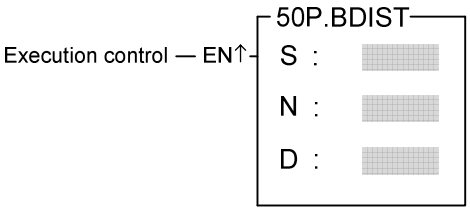
	High Byte	Low Byte
R1500	Don't care	Byte-0
R1501	Don't care	Byte-1
R1502	Don't care	Byte-2
R1503	Don't care	Byte-3
R1504	Don't care	Byte-4
R1505	Don't care	Byte-5
R1506	Don't care	Byte-6
R1507	Don't care	Byte-7
R1508	Don't care	Byte-8
R1509	Don't care	Byte-9

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

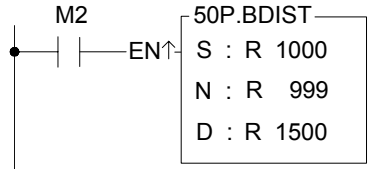
D ：Registers to store the distributed data

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 "EN ↑" （ P instruction）changes from 0→1, 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

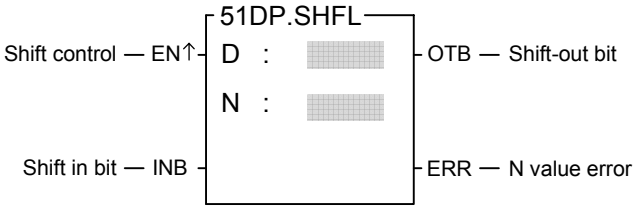
	High Byte	Low Byte
R1000	Byte-0	Byte-1
R1001	Byte-2	Byte-3
R1002	Byte-4	Byte-5
R1003	Byte-6	Byte-7
R1004	Byte-8	Don't care

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

Ladder symbol



D : Register to be shifted

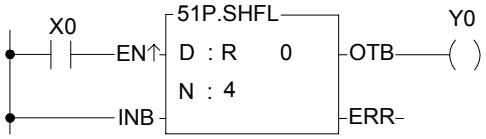
N : Number of bits to be shifted

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

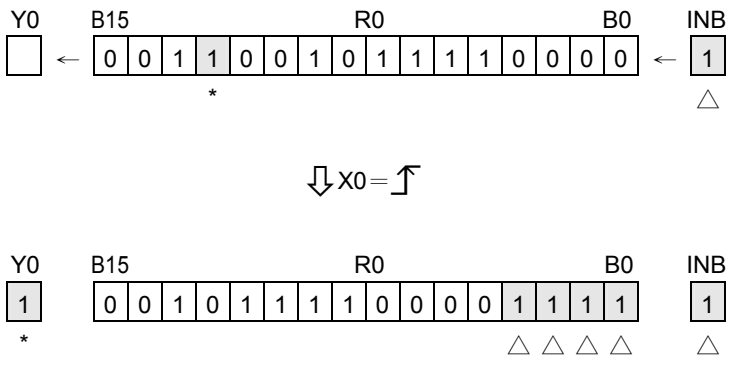
Operand \ Range	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR
	WX0 | WX240	WY0 | WY240	WM0 | WM1896	WS0 | WS984	T0 | T255	C0 | C255	R0 | R3839	R3840 | R3903	R3904 | R3967	R3968 | R4167	R5000 | R8071	D0 | D4095	1 | or | 16 32	V、Z P0~P9
D		○	○	○	○	○	○		○	○*	○*	○		○
N	○	○	○	○	○	○	○	○	○	○	○	○	○	○

- When shift control "EN" = 1 or "EN ↑ " (P instruction) has a transition from 0 to 1, 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.

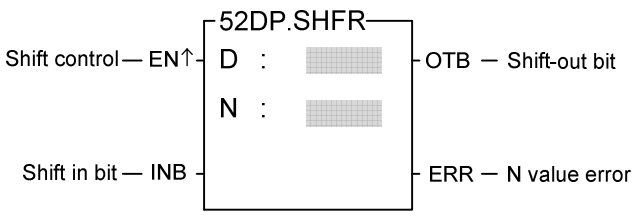


- The instruction at left shifts the data in register R0 towards the left by 4 successive bits. The results are shown below.



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

Ladder symbol



D : Register to be shifted

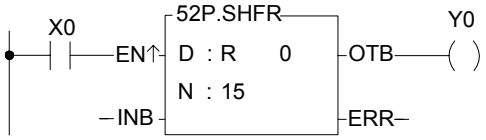
N : Number of bits to be shifted

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

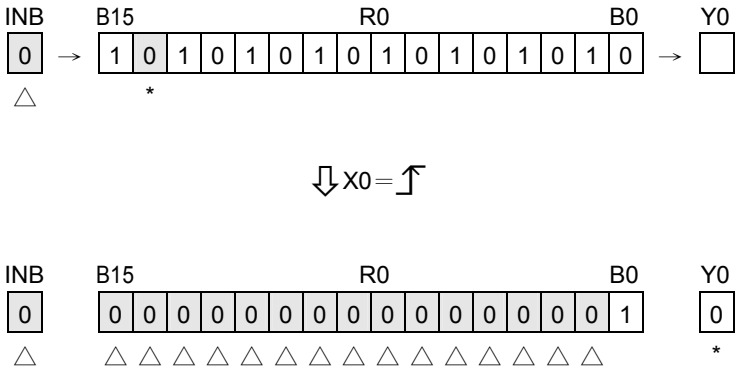
Range / Operand	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR
	WX0 | WX240	WY0 | WY240	WM0 | WM1896	WS0 | WS984	T0 | T255	C0 | C255	R0 | R3839	R3840 | R3903	R3904 | R3967	R3968 | R4167	R5000 | R8071	D0 | D4095	1 | or | 16 32	V、Z P0~P9
D		○	○	○	○	○	○		○	○*	○*	○		○
N	○	○	○	○	○	○	○	○	○	○	○	○	○	○

- When shift control "EN" = 1 or "EN ↑ " (P instruction) has a transition from 0 to 1, 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.

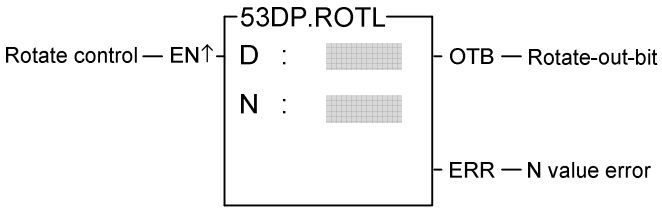


- The instruction at left shifts the data in R0 register towards the right by 15 successive bits. The results are shown below.



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

Ladder symbol



D : Register to be rotated

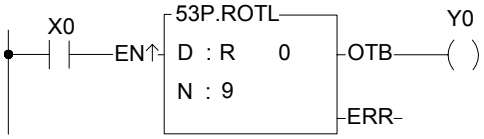
N : Number of bits to be rotated

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

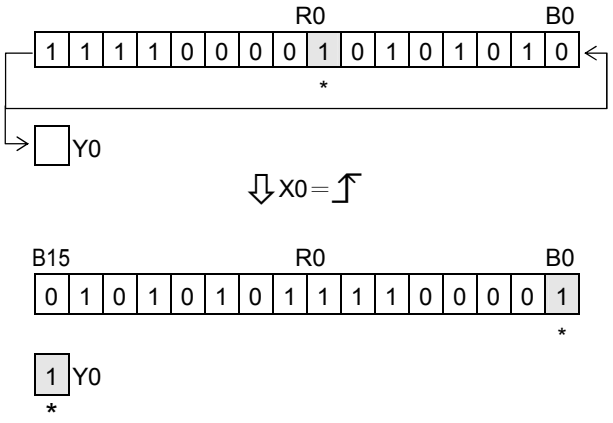
Operand \ Range	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR
	WX0 | WX240	WY0 | WY240	WM0 | WM1896	WS0 | WS984	T0 | T255	C0 | C255	R0 | R3839	R3840 | R3903	R3904 | R3967	R3968 | R4167	R5000 | R8071	D0 | D4095	1 | or | 16 32	V、Z P0~P9
D		○	○	○	○	○	○		○	○*	○*	○		○
N	○	○	○	○	○	○	○	○	○	○	○	○	○	○

- When rotate control "EN" = 1 or "EN ↑ " (P instruction) has a transition from 0 to 1, 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.

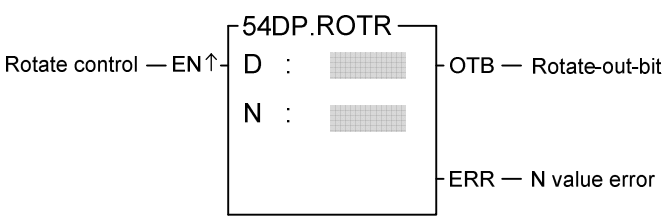


- The instruction at left rotates data from the R0 register towards the left 9 successive bits. The results are shown below.



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

Ladder symbol



D : Register to be rotated

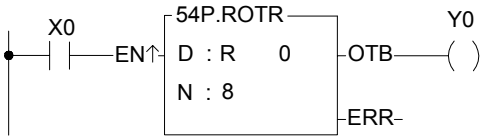
N : Number of bits to be rotated

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

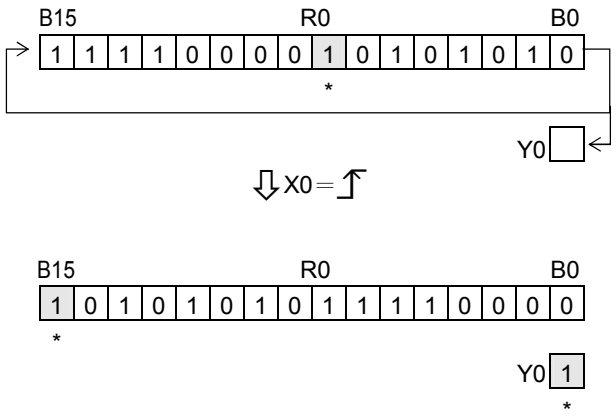
Operand \ Range	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR
	WX0 | WX240	WY0 | WY240	WM0 | WM1896	WS0 | WS984	T0 | T255	C0 | C255	R0 | R3839	R3840 | R3903	R3904 | R3967	R3968 | R4167	R5000 | R8071	D0 | D4095	1 | or | 16 32	V、Z P0~P9
D		○	○	○	○	○	○		○	○*	○*	○		○
N	○	○	○	○	○	○	○	○	○	○	○	○	○	○

- When rotate control "EN" = 1 or "EN ↑ " (P instruction) has a transition from 0 to 1, 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.

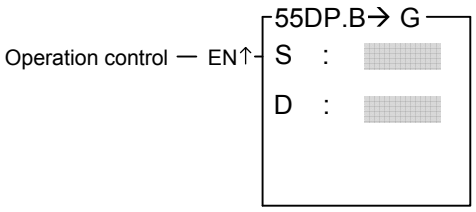


- The instruction at left rotates data from R0 register towards the right 8 successive bits. The results are shown below.



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

Ladder symbol



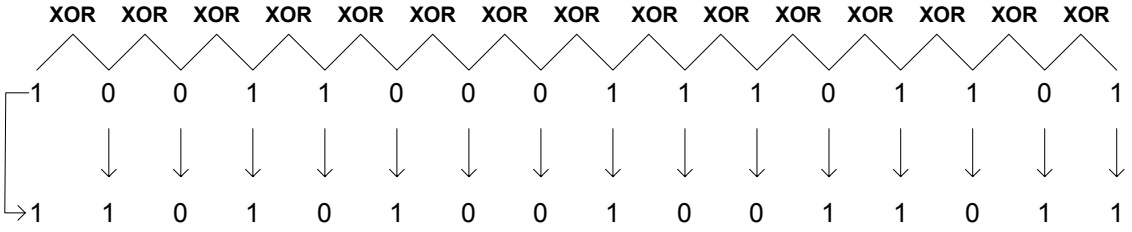
S ：Starting of source

D ：Starting address of destination

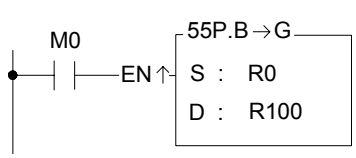
S，D operand can combine V、Z、P0~P9 for index addressing

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	0~FFFFH 0~FFFFFFFFH	V、Z P0~P9
S	○	○	○	○	○	○	○	○	○	○	○	○	○	○
D		○	○	○			○				○*	○		○

- When operation control "EN"=1 or "EN ↑"(P instruction) changes from 0→1, 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

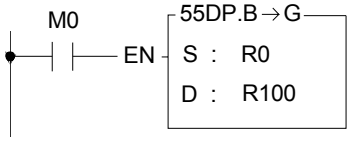


- Converting the 16-bit Binary-code in R0 into Gray-code, and then storing the result into R100.

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

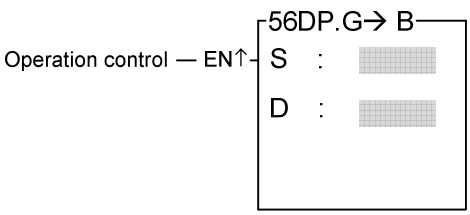


- Converting the 32-bit Binary-code in DR0 into Gray-code, and then storing the result into DR100.

DR0＝00110111001001000010111100010100B ➔ DR100＝00101100101101100011100010011110B

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

Ladder symbol



S ：Starting of source

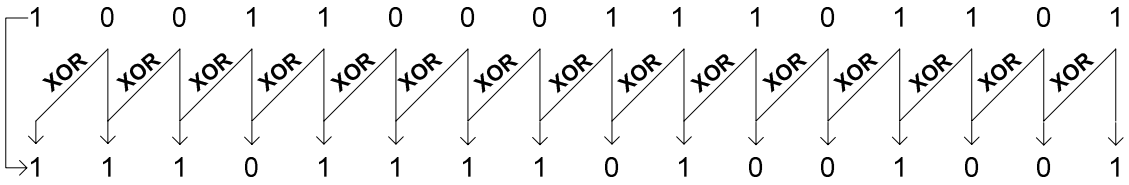
D ：Starting address of destination

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

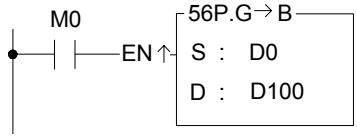
Range / Operand	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR
	WX0 | WX240	WY0 | WY240	WM0 | WM1896	WS0 | WS984	T0 | T255	C0 | C255	R0 | R3839	R3840 | R3903	R3904 | R3967	R3968 | R4167	R5000 | R8071	D0 | D4095	0~FFFFH 0~FFFFFFFFH	V、Z P0~P9
S	○	○	○	○	○	○	○	○	○	○	○	○	○	○
D		○	○	○			○				○*	○		○

- When operation control "EN"=1 or "EN↑"(P instruction) changes from 0→1, 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

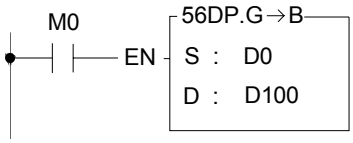


·Converting the 16-bit Gray-code in D0 into Binary-code, and then storing the result into D100.

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

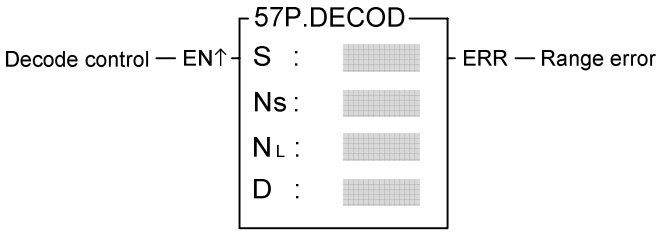


˙Converting the 32-bit Gray-code in DD0 into Binary-code, and then storing the result into DD100.

DD0＝00110111001001000010111100010100B ➔ DD100＝00100101110001111100101000011000B

FUN 57 P DECOD	DECODE	FUN 57 P DECOD

Ladder symbol



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

N_S : Starting bits to be decoded within S

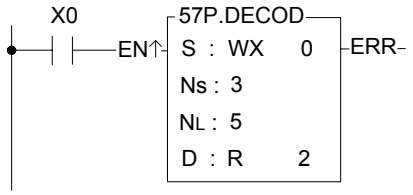
N_L : Length of decoded value (1~8 bits)

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

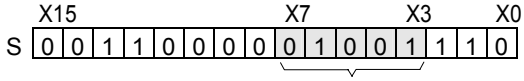
S, N_S, N_L, D may combine with V, Z, P0~P9 to serve indirect address application

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	16-bit +/- number	V、Z P0~P9
S	○	○	○	○	○	○	○	○	○	○	○	○	○	○
N_S	○	○	○	○	○	○	○	○	○	○	○	○	0～15	○
N_L	○	○	○	○	○	○	○	○	○	○	○	○	1～8	○
D		○	○	○	○	○	○		○	○*	○*	○		○

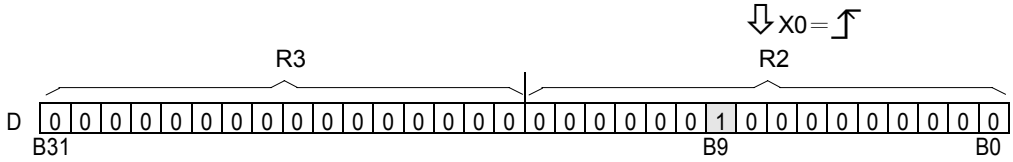
- This instruction, will set a single bit among the total of 2^{NL} discrete points (D) to 1 and the others bit are set to 0. The bit number to be set to 1 is specified by the value comprised by BN_S～BN_S+N_L-1 of S（which is called the decode value, BN_S is the starting bit of the decode value, and BN_S+N_L-1 is the end value）,.
- When decode control "EN" = 1 or "EN ↑ " (P instruction) has a transition from 0 to 1, 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.



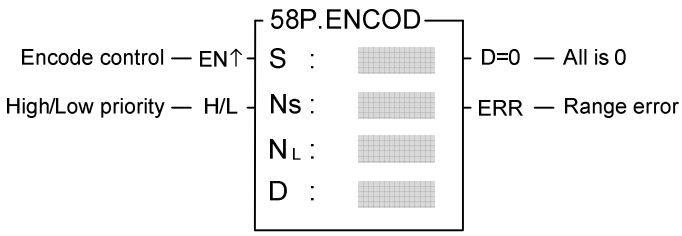
Length of decode value N_L=5,so bit value is formed by X7~X3 (equal 9)



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

Ladder symbol



S : Starting register to be encoded

N_S : Bit position within S as the encoding start point

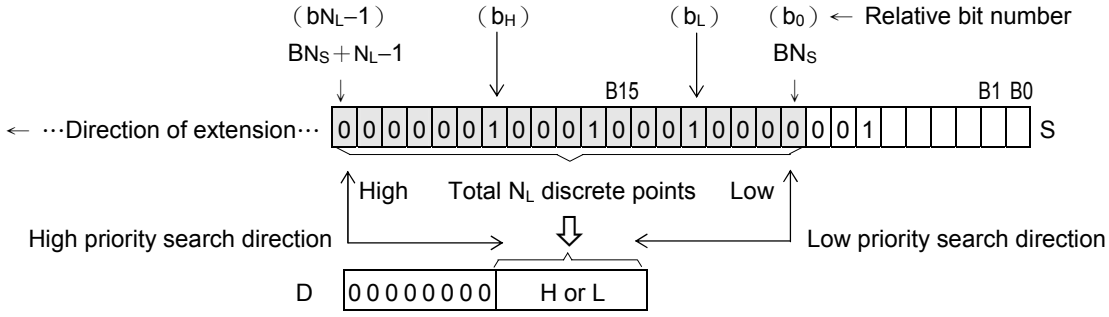
N_L : Number of encoding discrete points (2~256)

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		○	○	○	○	○	○		○	○*	○*	○		○

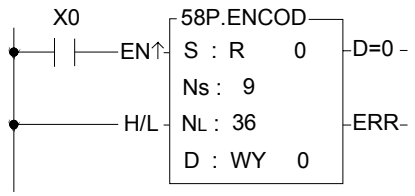
- When encode control "EN" = 1 or "EN ↑ " (P instruction) has a transition from 0 to 1, will starting from the points specified by Ns within S, take out towards the left (high position direction) N_L number of successive bits BN_S～BN_S+N_L-1 (BN_S is called the encoding start point, and its relative 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.



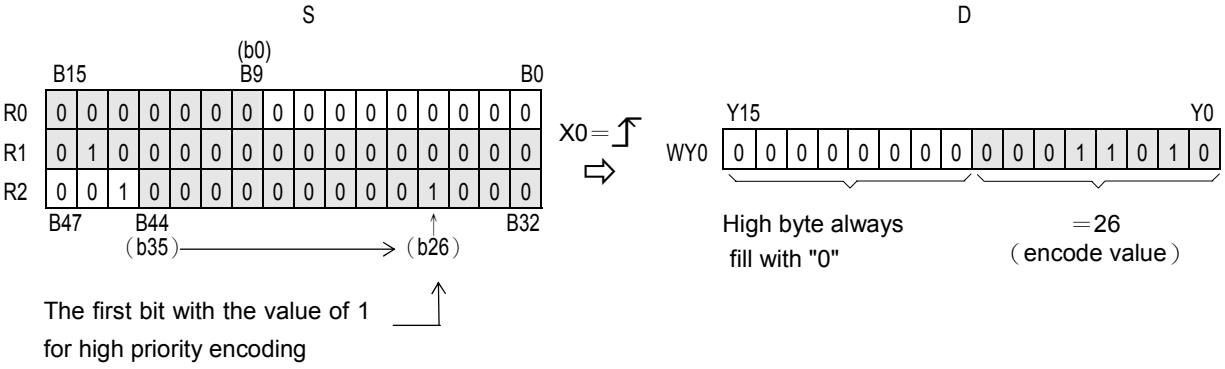
- As shown in the diagram above, for high priority encoding, the bit first to find is b_H (with a value of 12), and for low priority encoding, the bit first to find b_L (with a value of 4). Among the N_L discrete points there must be at least one bit with value of 1. If all bits are 0, will not to carry 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.

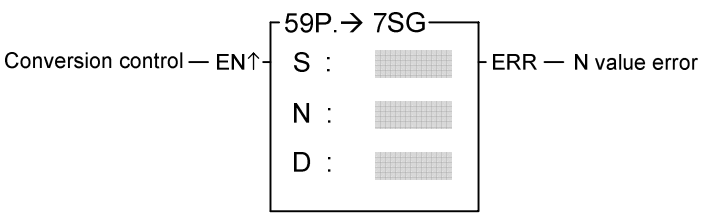


- The instruction at left is a high priority encode example. When X0 goes from 0 to 1, will take out toward left 36 successive bits starting from B9 (b0) specified by Ns within S, and perform high priority encoding (because H/L = 1). That is, starting from b35 (encoding end point), move right to find the first bit with the value of 1. The resultant value of this example is b26, so the value of D is 001AH=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 / 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-bit +/- number	V、Z P0~P9
S	○	○	○	○	○	○	○	○	○	○	○	○	○	○
N	○	○	○	○	○	○	○	○	○	○	○	○	0～3	○
D		○	○	○	○	○	○		○	○*	○*	○		○

- When conversion control "EN" = 1 or "EN ↑ " (P instruction) has a transition from 0 to 1, 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" shown in page 9-31.
- 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-7SG) and the FUN84 (7SEG) handy instruction for mixing decoding and non-decoding application, FUN59 and FUN84 can be combined to simplify the program design.(Please refer the example in chapter 16)

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

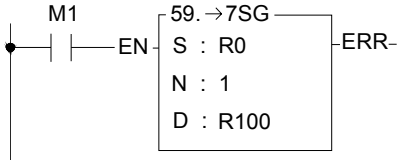
〈Example 1〉When M1 OFF→ON, convert hexadecimal to 7-Segment

- 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）

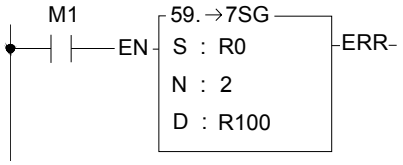
〈Example 2〉When M1 ON, convert the hexadecimal to 7-Segment



- 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



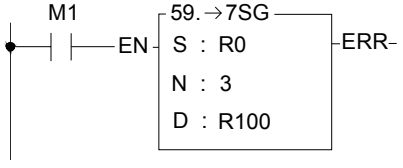
- Instruction at left will convert the first, second and third digit of R0 to 7-segment and store in R100 and R101.
- The low byte of R100 stores first digit.
- The high byte of R100 stores second digit.
- The low byte of R101 stores third digit.
- The high byte of R10 remain unchanged.

Original R101=0000H

R0=0A48H ➔ R100=337FH（48）

R101=0077H（A）

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

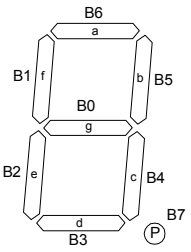


- Instruction at left will convert 1~4 digit of R0 to 7-segment and store in R100 and R101.
- The low byte of R100 stores first digit.
- The high byte of R100 stores second digit.
- The low byte of R101 stores third digit.
- The high byte of R10 stores 4th digit.

R0=2790H ➔ R100=7B7EH（90）

R101=6D72H（27）

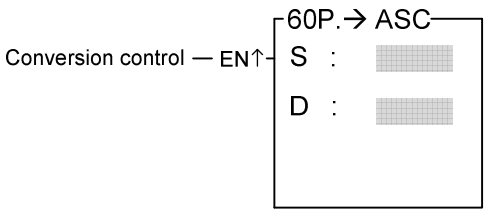
FUN 59 P →7SG	7-SEGMENT CONVERSION	FUN 59 P →7SG

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



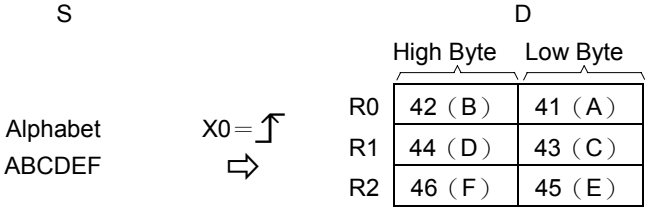
S : Alphanumerics to be converted into ASCII code

D : Starting register storing ASCII results

Operand \ Range	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	○	○	○	○	○	○	○	○*	○*	○	

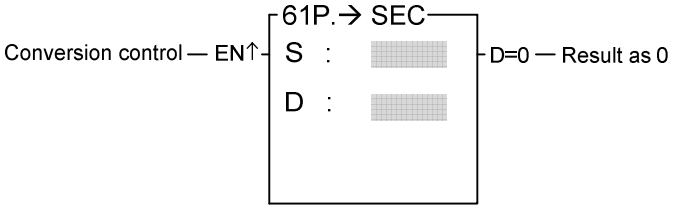
- When conversion control "EN" = 1 or "EN ↑ " (P instruction) has a transition from 0 to 1, will convert alphabets and numbers stored in S (S has a maximum of 12 alphanumeric character) into ASCII and store it into registers starting from D. Each 2 alphanumeric characters occupy one 16-bit register.
- 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.



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

Ladder symbol

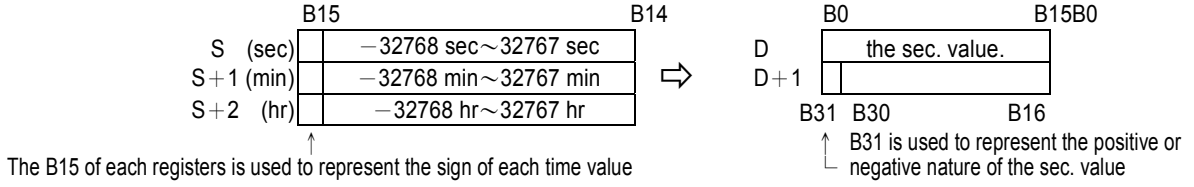


S : Starting calendar data register to be converted

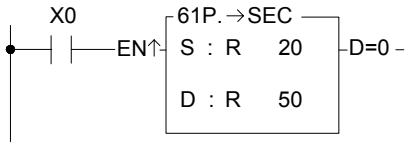
D : Starting register storing results

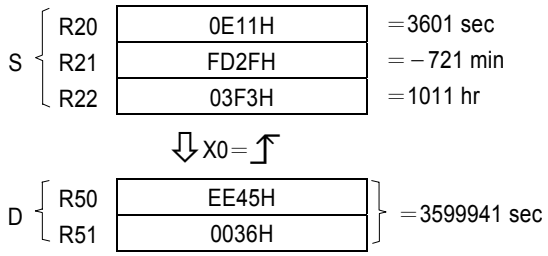
Range / Operand	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K
	WX0 | WX240	WY0 | WY240	WM0 | WM1896	WS0 | WS984	T0 | T255	C0 | C255	R0 | R3839	R3840 | R3903	R3904 | R3967	R3968 | R4167	R5000 | R8071	D0 | D4095	−117968399 | 117964799
S	○	○	○	○	○	○	○	○	○	○	○	○	○
D		○	○	○	○	○	○		○	○*	○*	○	

- When conversion control "EN" = 1 or "EN ↑ " (P instruction) has a transition from 0 to 1, 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.



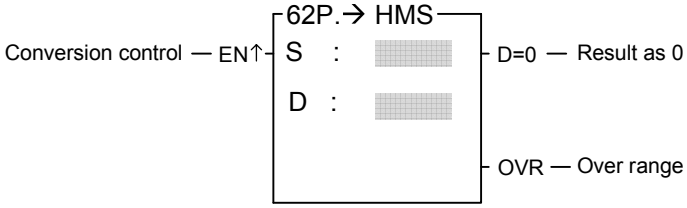
- Besides FUN61 or 62 instruction which treat hour: minute: second registers as an integral data, other instructions treat it as individual registers.
- The example program at below converts the hour: minute: second data formed by R20~R22 into their equivalent value in seconds then stored in the 32-bit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





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

Ladder symbol

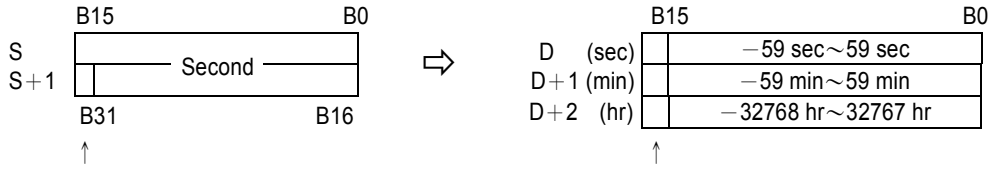


S :Starting register of second to be converted

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		○	○	○	○	○	○		○	○*	○*	○	

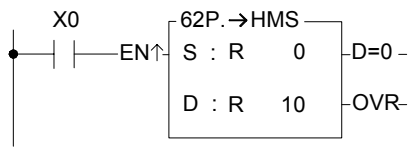
- When conversion control "EN" = 1 or "EN ↑ " (P instruction) has a transition from 0 to 1, will convert the second data from the S~S+1 32-bit register into the equivalent hour : minute : second time value and store it in the three successive registers D~D+2. All the data in this instruction is represented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

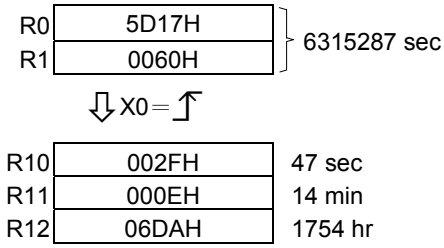


The bit B31 of the second register is used as the sign bit of the second value.

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

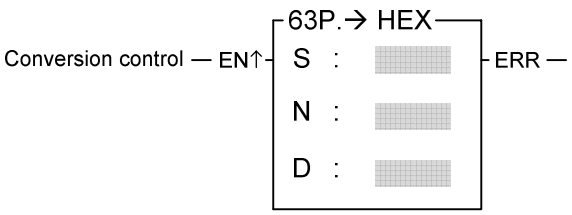
- As shown in the diagram above, after convert to hour : minute : second value, the minute : second value can only be in the range of -59 to 59, and the hour number can be in the range of -32768 to 32767 hours. Because of this, the maximum limit of D is -32768 hours, -59 minutes, -59 seconds to 32767 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 “EN ↑ ” (P instruction) changes from 0→1, 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.

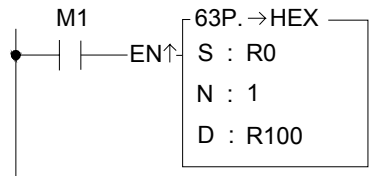
- The conversion will not be performed when N is 0 or greater than 511.

- 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

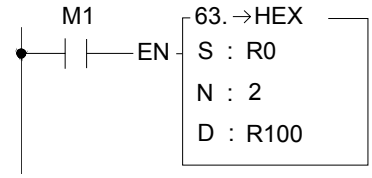
〈Example 1〉When M1 from OFF→ON, ASCII code converted to hexadecimal value.



- 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

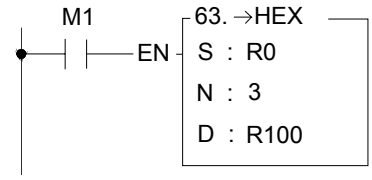
〈Example 2〉When M1 is ON, ASCII code converted to hexadecimal value.



- 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

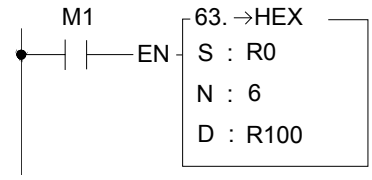
〈Example 3〉When M1 is ON, ASCII code converted to hexadecimal value.



- 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.

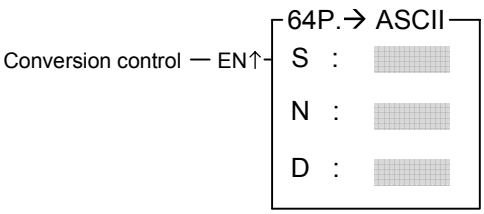


- Converts the ASCII code of R0~R5 into hexadecimal value and store it to R100~R101

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



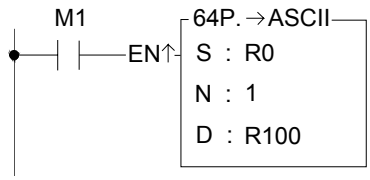
S : Starting source register
N : Number of hexadecimal digit to be converted to ASCII code.
D : The starting register storing result.
S, N, D, can associate with V, Z, P0~P9 to do the indirect addressing application.

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 “EN ↑ ” (P instruction) changes from 0→1, 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 2.

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

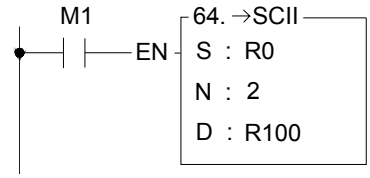
〈Example 1〉 When M1 changes from OFF→ON, it converts hexadecimal value to ASCII code.



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

R0＝0009H ➔ R100＝0039H（9）

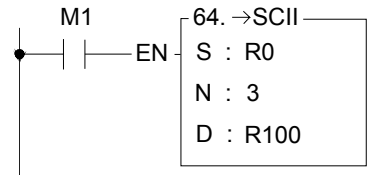
〈Example 2〉 When M1 is ON, it converts hexadecimal value to ASCII code.



- 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）

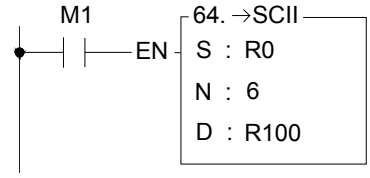
〈Example 3〉 When M1 is ON, it converts hexadecimal value to ASCII code.



- 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.

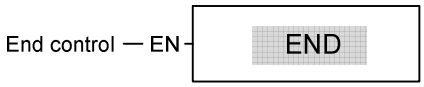


- Converts the NB0～NB5 of R0～R1 to ASCII code and stores it into R100～R105

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

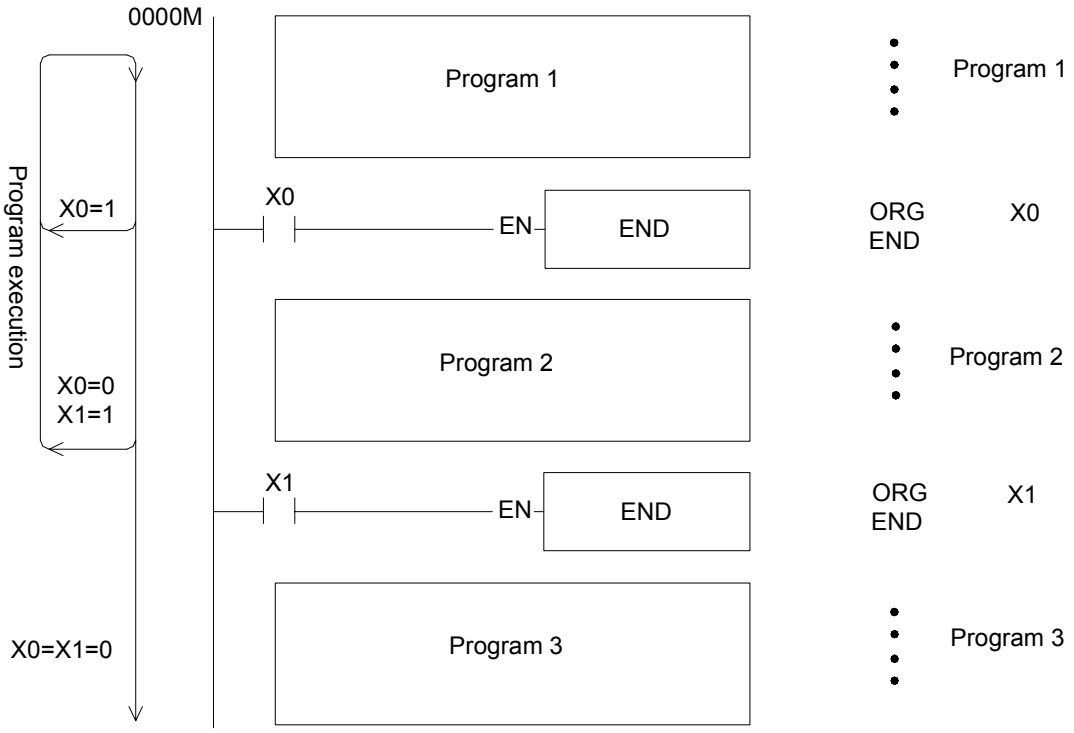
END	PROGRAM END	END

Ladder symbol



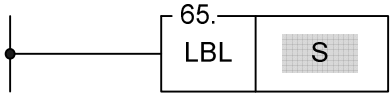
No operand

- 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



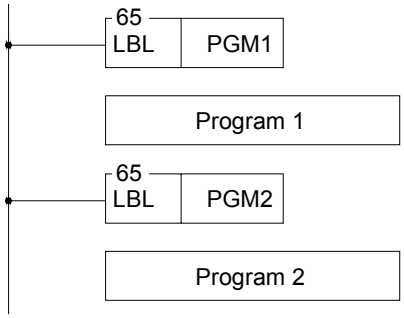
S : Alphanumeric, 1~6 characters

- 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）	Label for High speed fixed timer interrupt service routine.
PSO0I～PSO3I	Labels for the pulse output command finished interrupt service routine.

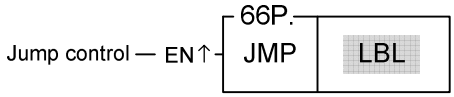
Only the interrupt service routine can use the label names listed on above table, if mistaken on using the reserved label on the normal subroutine can cause the CPU fail or unpredictable operation.

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.



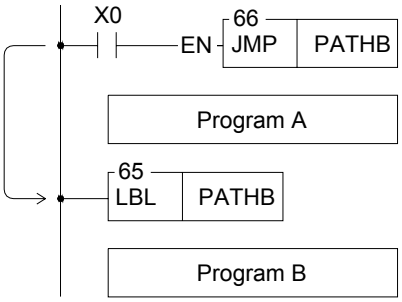
FUN 66 P JMP	JUMP	FUN 66 P JMP

Ladder symbol



LBL : The program label to be jumped

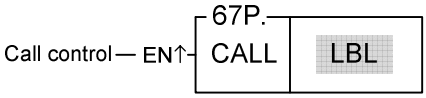
- When jump control "EN"=1 or "EN ↑ " (P instruction) changes from 0→1, PLC will jump to the location behind the marked label and continuous to execute the program.
- This instruction is especially suit for the applications where some part of the program will be executed only under certain condition. 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

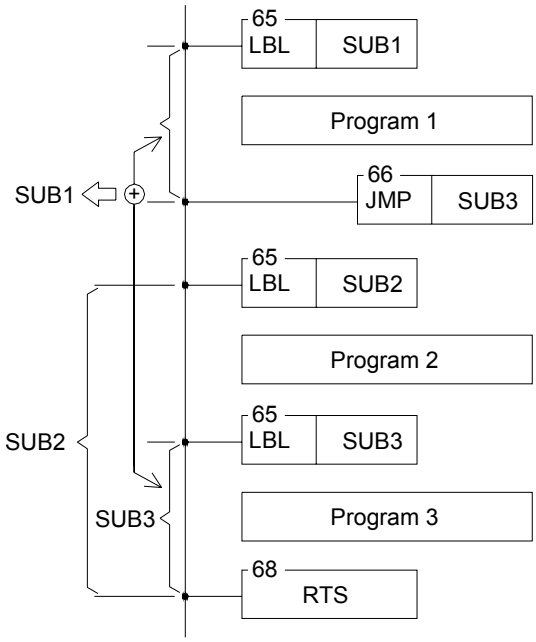


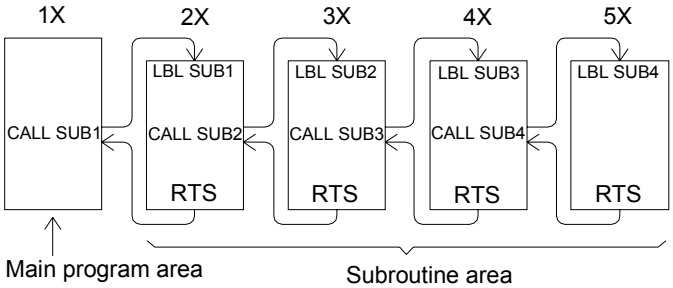
LBL : The subroutine label name to be called.

- When call control "EN"=1 or "EN ↑ " (P instruction) changes from 0→1, 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).

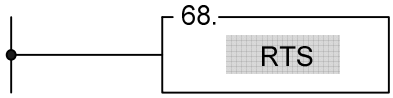




- Interrupt service programs （HSC0I～HSC7I、PSO0I～PSO3I、X0+I～X15+I／INT0～INT15、X0–I～X15–I／INT0–～INT15–、HSTAI／ATMRI、1MSI／1MS、2MSI／2MS、3MSI／3MS、4MSI／4MS、5MSI／5MS、10MSI／10MS、50MSI／50MS、100MSI／100MS） are also a kind of subroutine. It is also placed in sub program area. However, the calling of interrupt service 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

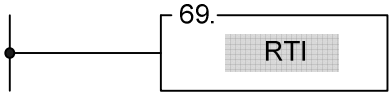
Ladder symbol



- 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

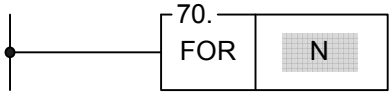
Ladder symbol



- 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 "EN ↑ " (instruction) changes from 0→1, 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 10 for explanation.

FUN 70 FOR	FOR	FUN 70 FOR

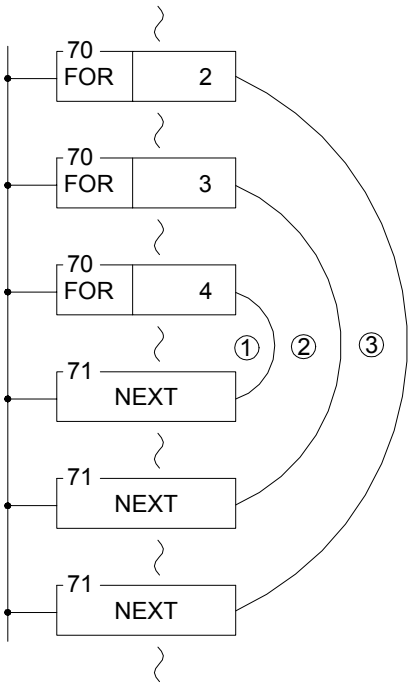
Ladder symbol



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	○	○	○	○	○	○	○	○	○	○	○	○	○

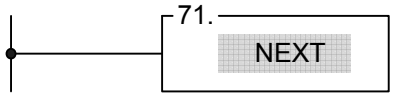
- This instruction has no input control, is connected directly to the power line, and cannot be in series with any conditions.
- The programs within the FOR and NEXT instructions form a program loop (the start of the loop program is the next instruction after FOR, and the last is the instruction before 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



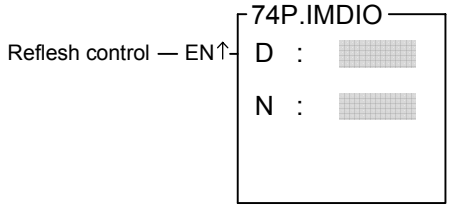
- This instruction and the FOR instruction together form a program loop. The instruction itself has no input control, is connected directly to the power line, and cannot be in series with any conditions.

- 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	IMMIDIATE 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 / Ope-rand	X	Y	K
	Xn of Main Unit.	Yn of Main Unit.	1 | 36
D	○	○	
N			○

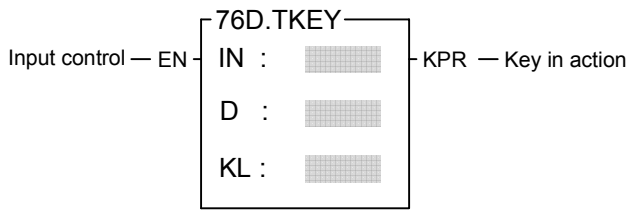
- For normal PLC scan cycle, the CPU gets the entire input signals before the program is executed, and then perform the executing of program based on the fresh input signals. After finished the program execution the CPU will update all the output signals according to the result of program execution. 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 "EN ↑ " (P instruction) has a transition from 1 to 0, 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	20 points	32 points	40 points	60 points
Input signals	X0～X11	X0～X19	X0～X23	X0～X35
Output signals	Y0～Y7	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

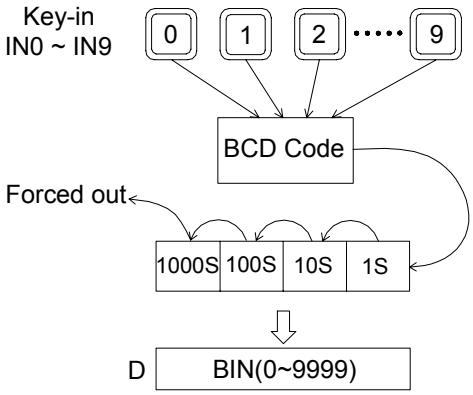
KL: starting coil to reflect the input status

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

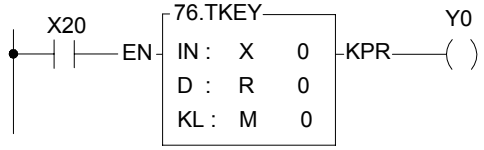
Range / Operand	X	Y	M	S	WY	WM	WS	TMR	CTR	HR	OR	SR	ROR	DR	XR
	X0 | X240	Y0 | Y240	M0 | M1896	S0 | S984	WY0 | WY240	WM0 | WM1896	WS0 | WS984	T0 | T255	C0 | C255	R0 | R3839	R3904 | R3967	R3968 | R4167	R5000 | R8071	D0 | D4095	V、Z | P0~P9
IN	○														
D					○	○	○	○	○	○	○	○*	○*	○	○
KL		○	○	○											

- 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.



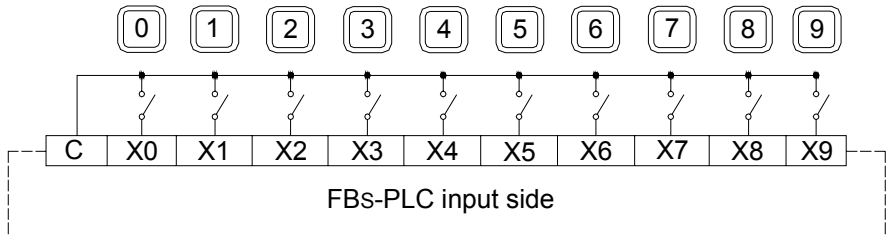
- When input control "EN" = 0, this instruction will not be executed. KPR output and KL coil status will be 0. However, the numerical values of D register will remain unchanged.



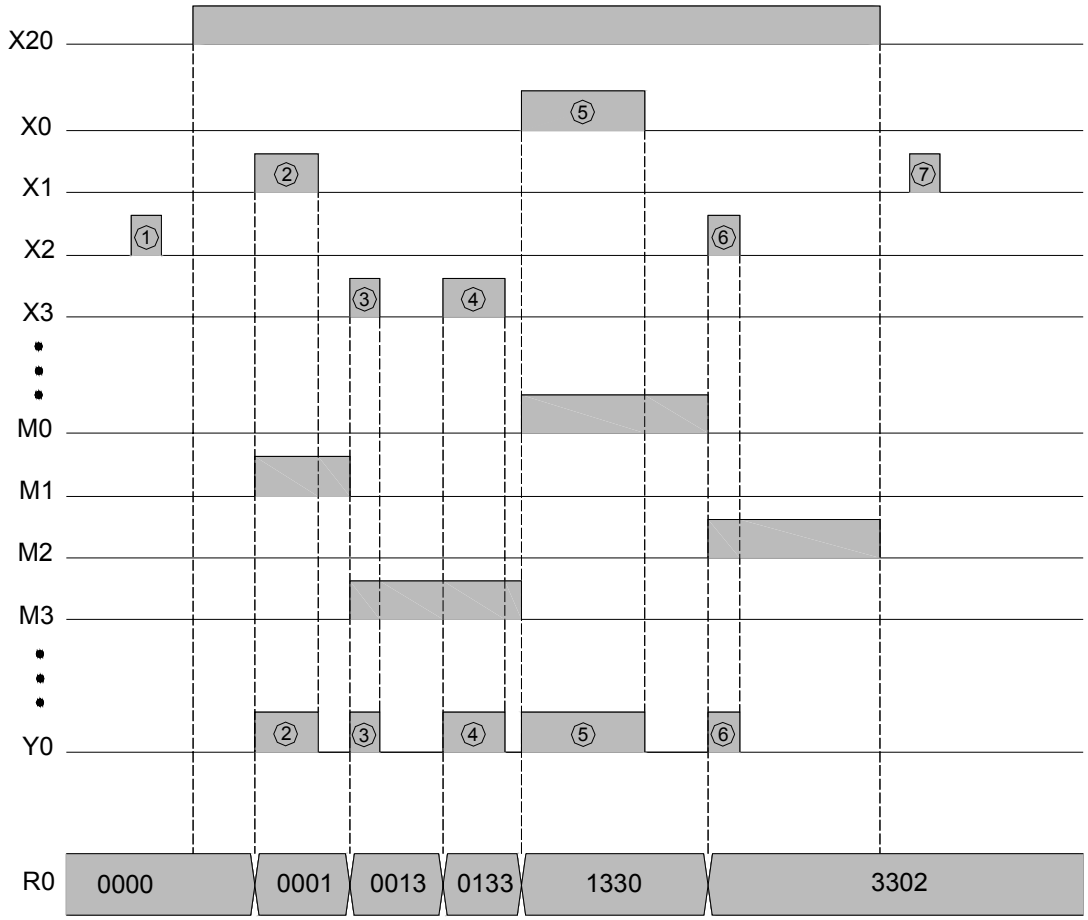
• The instruction at left represents the input point X0 with the number "0", X1 is represented by 1, ... , M0 records the action of X0, M1 records the action of X1 ... , and the input numerical values are stored in the R0 register.

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

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

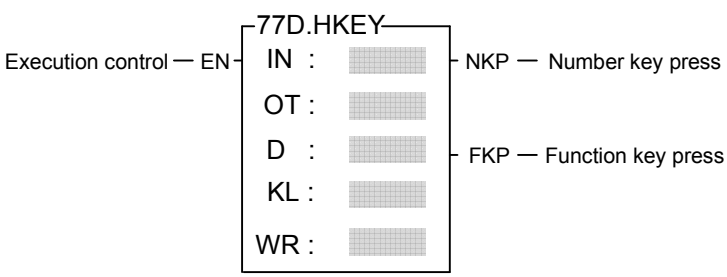


- If the X0~X3 key-in sequence follow the ①②③④⑤⑥⑦ sequence in the following diagram. At step ① and ⑦ the X20 is 0, so there was no key generated, only steps ②③④⑤⑥ are effective. Because the register can only hold 4 key numbers, Of these 5 steps the first key was kick out. The key strokes 3302 of the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



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

Ladder symbol



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

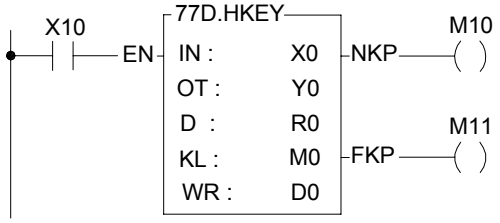
KL : Starting relay for key status

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

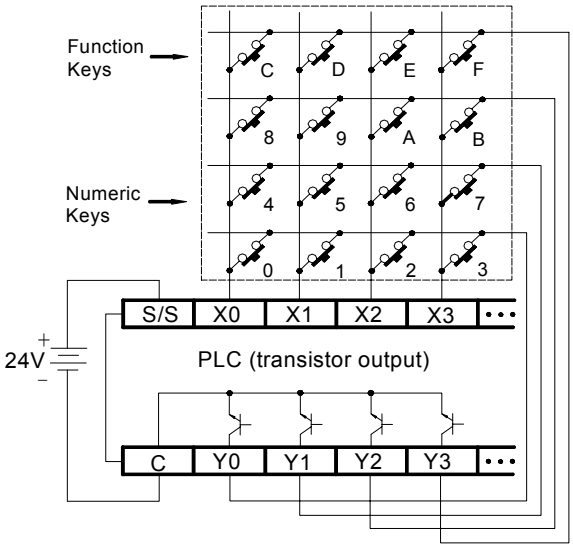
D may combine with V、Z、P0~P9 to serve indirect addressing application

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

- 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. 4×4 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.

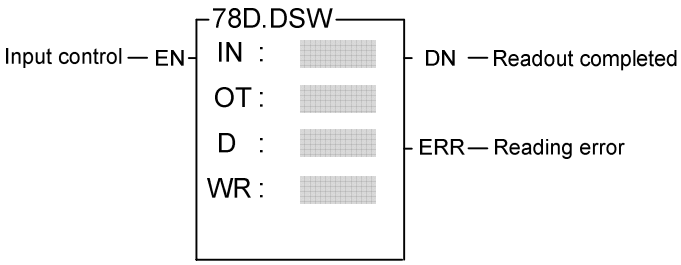


- The instruction in the diagram above uses X0~X3 and Y0~Y3 to form a multiplex key input. It can input numeric values of 8 digits and stores the results in R1R0. The input status of the function keys is stored in M10(A)~M15(F).



FUN 78 D DSW	DIGITAL SWITCH INPUT	FUN 78 D DSW

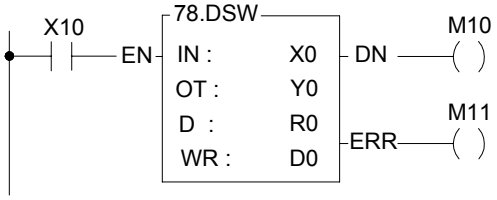
Ladder symbol



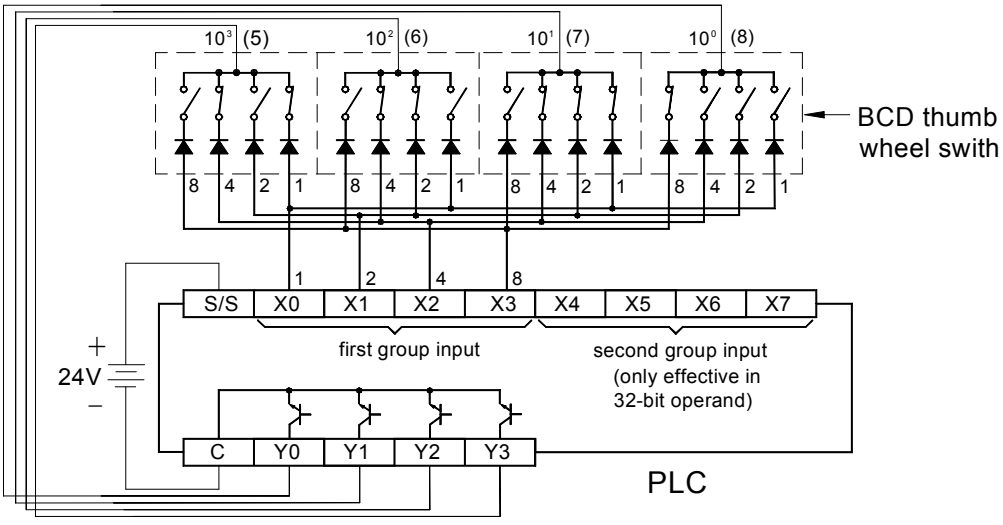
IN : Starting of input for thumb wheel switch
OT: Starting of output for multiplexing scan (4 points)
D : Register to store readout value
WR: Working register, it can't repeat in use (WR & WR+1 for 16-bit operation; WR, WR+1 & WR+2 for 32-bit operation)
D may combine with V、Z、P0~P9 to serve indirect addressing application

Operand \ Range	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			○	○	○	○	○	○	○	○*	○*	○	○
WR								○			○*	○	

- 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.

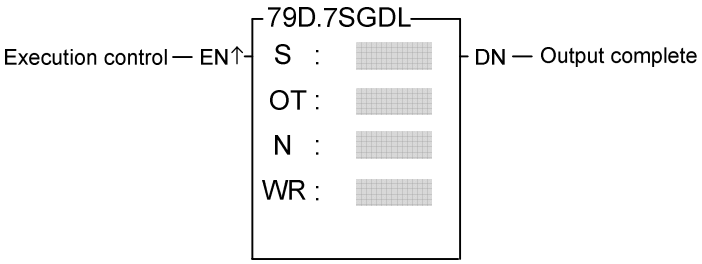


- In this example, when X10 is 1, then the numeric value of the thumb wheel switch (5678 in this example) will be read out and stored into the R0 register.
- The bits (8,4,2,1) with same digit should be connect together and series with a diode (as shown in diagram below).
- With 32-bit operand a set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

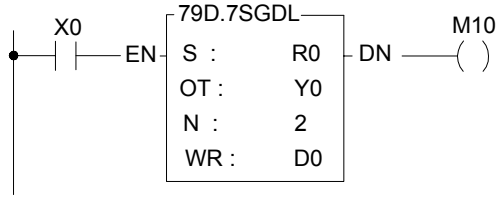
S may combine with V、Z、P0~P9 to serve indirect addressing application

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	

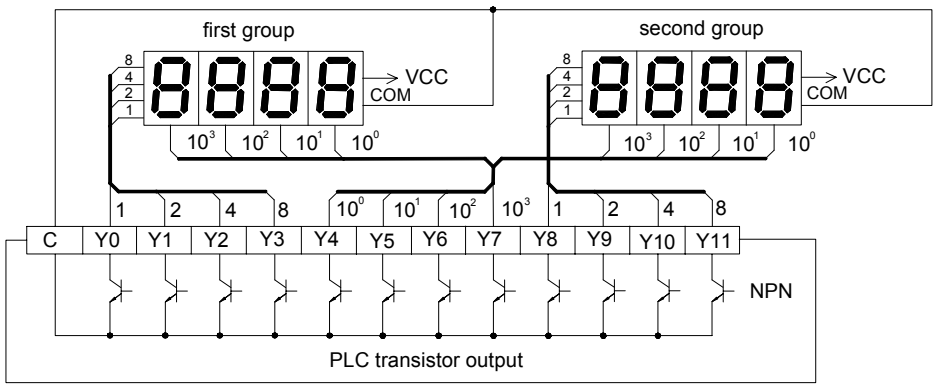
- When input control "EN" = 1, the 4 nibbles of the S register, from digit 0 to digit 3, are sequentially sent out to the 4 output points, OT0~OT3. While output the digit data, the latch signal of that digit (OT4 corresponds to digit 0, OT5 corresponds to digit 1, etc...) at the same time is also sent 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.



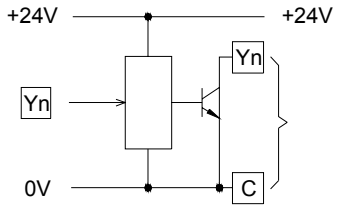
- In this example, when X0=1, the 4 nibbles of R0 will be transferred to the first group 7-segment display in the diagram below. The 4 nibbles of R1 will be transferred to the second group 7-segment display.



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

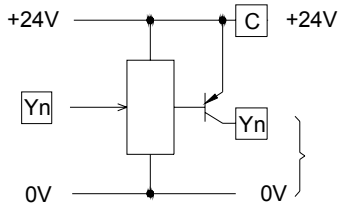
- FACON PLC's transistor output has both a negative logic transistor output (NPN transistor - when the output status is ON, the terminal voltage of the transistor output is low), and a positive logic transistor output (PNP - when the output status is ON, the terminal voltage of the transistor output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

FBs-PLC negative logic output (NPN transistor)



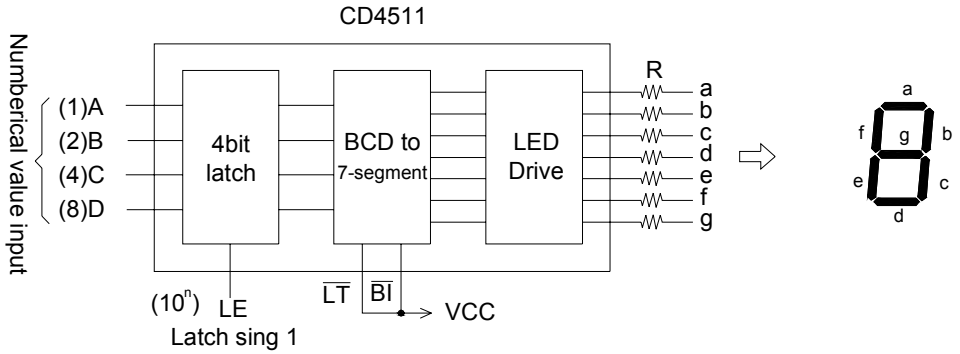
When Yn is "ON", this output voltage is low

FBs-PLC positive logic output (PNP transistor)



When Yn is "ON", Yn's terminal voltage is high

- 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.



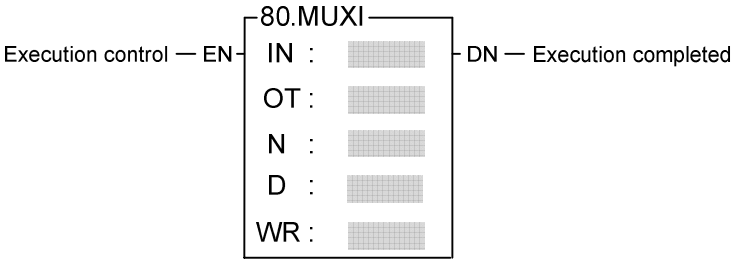
- Because the PLC output and the 7-segment display input polarity can be positive and negative logic. Therefore, the polarities between output and input must be coordinated to get the correct result. This instruction uses N to specify the polarity relation between the PLC transistor output, and the 7-segment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

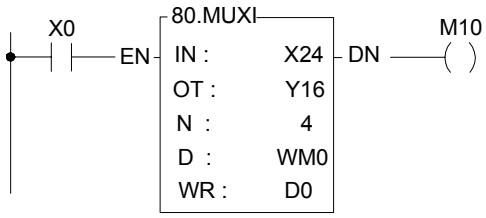
N : Multiplex input lines (2~8)

D : Register for storing results

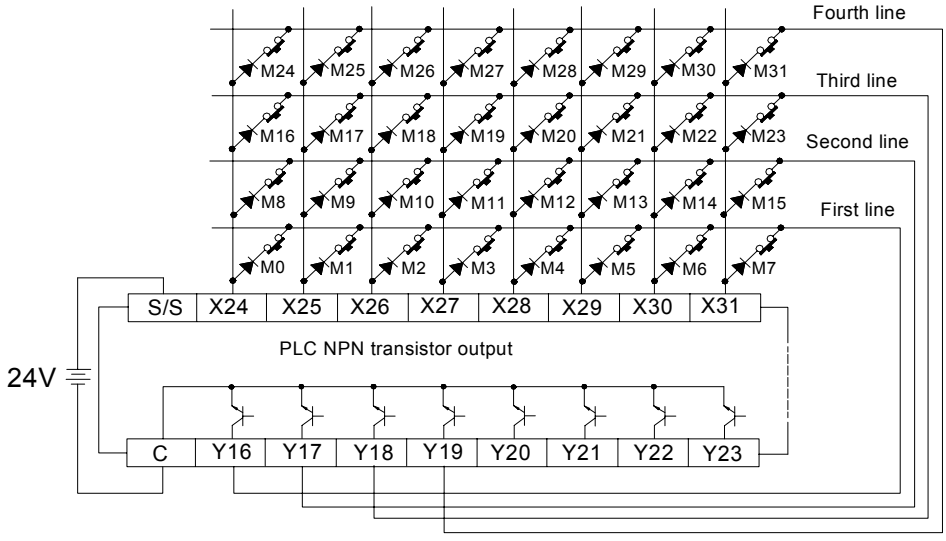
D may combine with V, Z, P0~P9 to serve indirect address application

Operand \ Range	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 8×N 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 8×N 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.

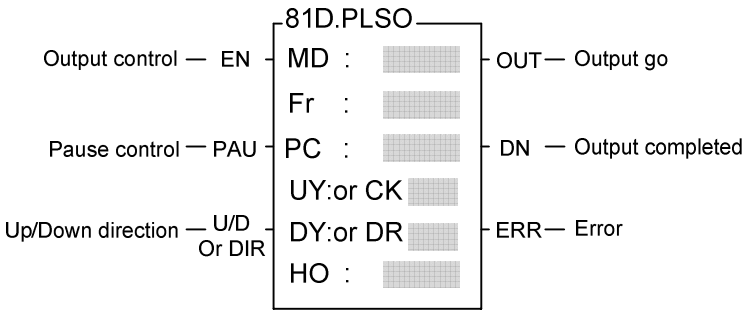


- This example retrieves 4 lines×8 points of input, 32 point status in all. They are stored into the 32-bit register of DWM0 (M0~M31).



FUN 81 D PLSO	PULSE OUTPUT	FUN 81 D PLSO

Ladder symbol



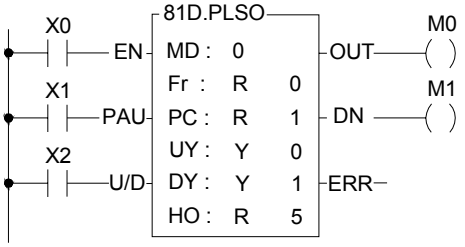
MD : Output mode selection
Fr : Pulse frequency
PC : Output pulse count
UY : Up pulse output point (MD=0).
DY : Down pulse output point (MD=0).
HO : Cumulative output pulse register. (Can be not assigned).
CK : Pulse output point (MD=1).
DR : Up/Down output point (MD=1).
DIR: 1- up; 0- down.

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			○	○	○	○	○	○	○	○*	○*	○	

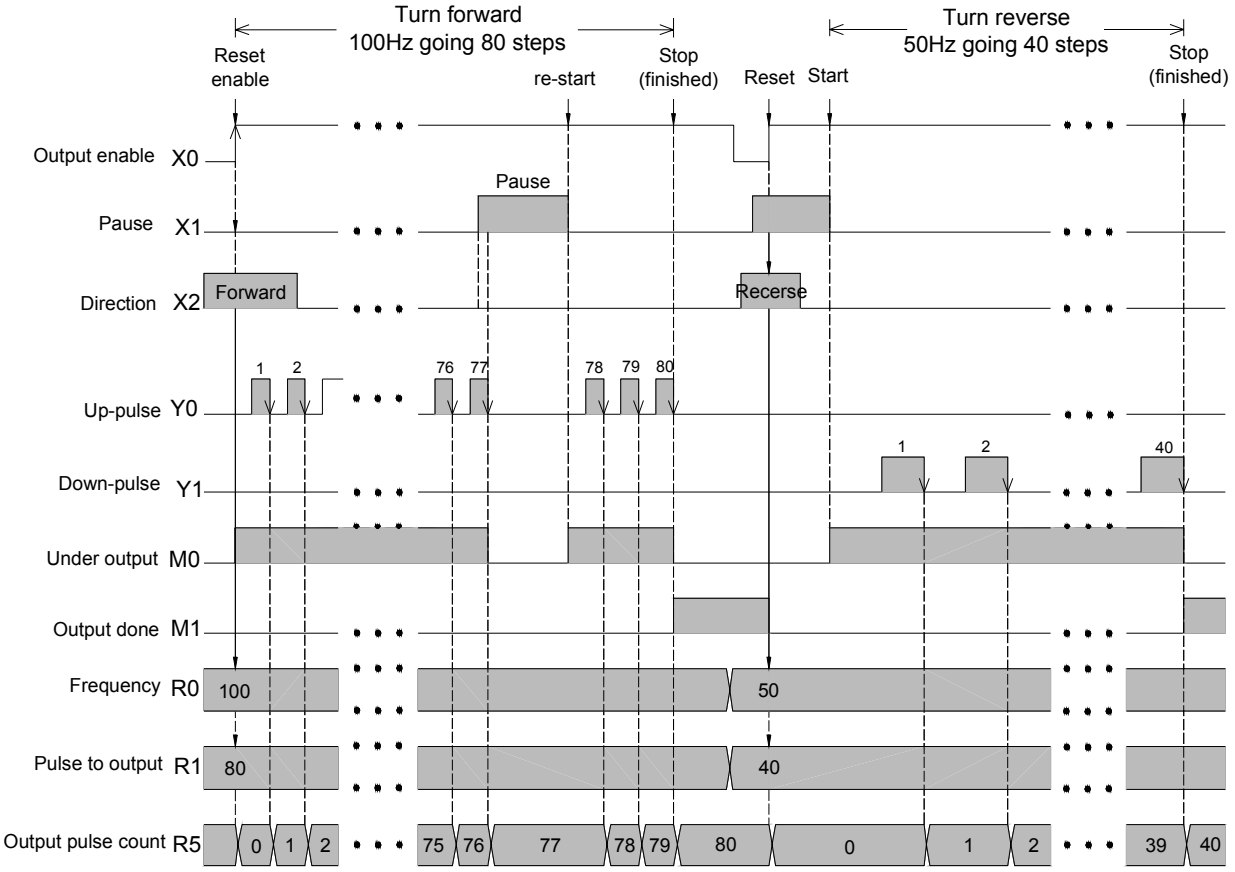
- When MD=0, this instruction performs the pulse output control as following:
- Whenever the output control “EN” changes from 0→1, it first performs the reset action, which is to clear the output flag “OUT” and “DN” as well as the pulse out register HO to be 0. It gets the pulse frequency and output 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 PLSO	PULSE OUTPUT	FUN 81 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.

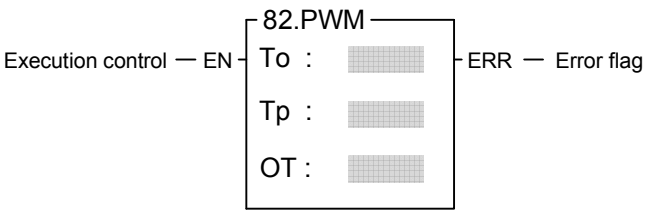


- In this example, the program controls the stepping motor to drive forward for 80 pulses (steps) at the speed of 100Hz first, and then makes it turn reverse for 40 pulses the speed of 50Hz. Make sure that the up/down direction, frequency Fr and the pulse count PC must be set before the reset take action(“EN”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



FUN 82 PWM	PULSE WIDTH MODULATION	FUN 82 PWM

Ladder symbol



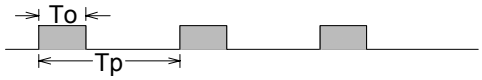
To : Pulse ON width
(0~32767mS)

Tp : Pulse period
(1~32676mS)

OT: Pulse output point

Operand \ Range	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	○													

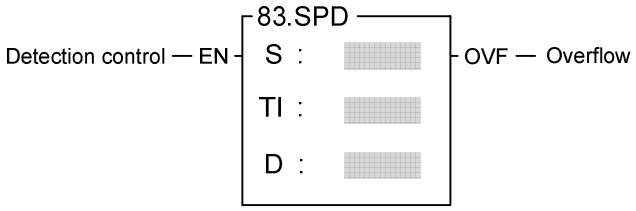
- When execution control "EN" = 1, will send the pulse to output point OT with the "ON" state for To ms and period as Tp. OT must be a transistor output point on the main unit. When "EN" is 0, the output point will be OFF.



- 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

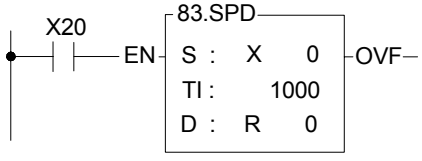


S : Pulse input point for speed detection
TI : Sampling duration
(units in mS)
D : Register storing results

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			○	○	○	○	○	○	○	○	○*	○*	○	

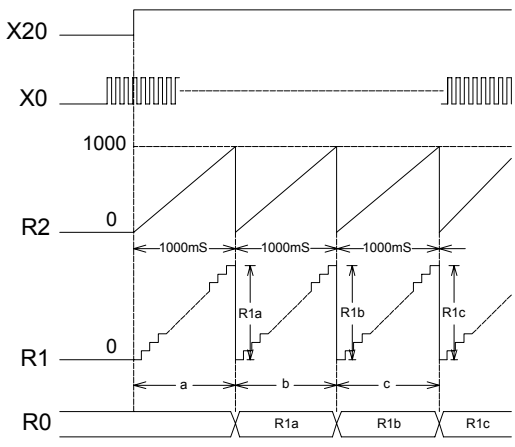
- This instruction uses the interrupt feature of the 8 high speed input points (X0~X7) on the PLC main unit to detect the frequency of the input signal. Within a specific sampling time (TI), it will calculate the input pulse count for S input point, and indirectly find the revolution speed of rotating 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

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



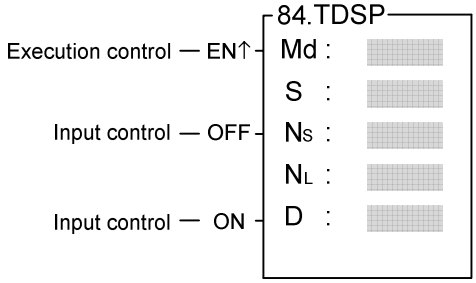
- In the above example, if every revolution of the rotating device produces 20 pulses (n = 20), and the R0 value is 200, then the revolution per minute speed "N" is as follows :

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



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	OR	ROR	DR	K	XR
	R0 | R3839	R3904 | R3967	R5000 | R8071	D0 | D4095	16/32 bit	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 begin converted characters, Ns is the pointer to locate the starting address character, Nl tells the length of begin 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.

  Nl is the character quantity for conversion.

  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.

  16-Segment display patterns shown as below :

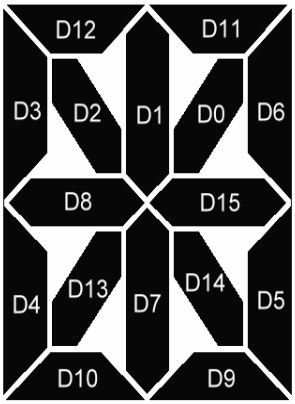
FUN 84 TDSP	PATTERN CONVERSION FOR 16/7-SEGMENT DISPLAY	FUN 84 TDSP

MSB / LSB	x000	x001	x010	x011	x100	x101
0000	0					
0001	1					
0010	2					
0011	3					
0100	4					
0101	5					
0110	6					
0111	7					
1000	8					
1001	9					
1010	A					
1011	B					
1100	C					
1101	D					
1110	E					
1111	F					

- If you don't find the pattern that you want in left table, you can create the pattern by yourself just reference below table.

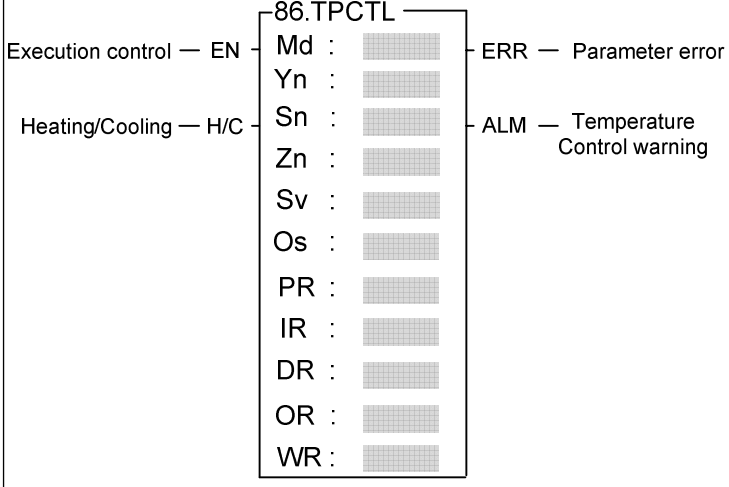
D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0

1 Word



FUN 86 TPCTL	PID TEMPERATURE CONTROL INSTRUCTION	FUN 86 TPCTL

Ladder symbol



Range / Operand	Y	HR	ROR	DR	K
	Y0 | Y255	R0 | R3839	R5000 | R8071	D0 | D3999	
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.

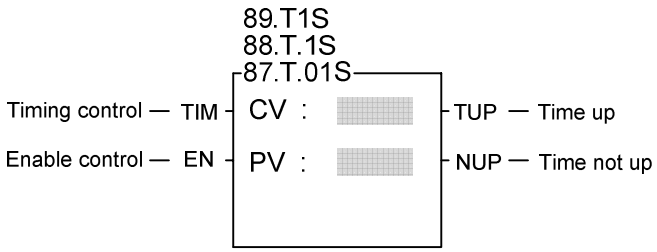
Note 2: The smaller the cycle time of PWM, the more even can it perform the heating. However, the error caused by the PLC scan time will also become greater. For the best control, it can base on the scan time of PLC to adjust the solution interval of PID calculation and the PWM cycle time.

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.

FUN89/FUN89D (T1S) FUN88/FUN88D (T.1S) FUN87/FUN87D (T.01S)	CUMULATIVE TIMER	FUN89/FUN89D (T1S) FUN88/FUN88D (T.1S) FUN87/FUN87D (T.01S)

Ladder symbol



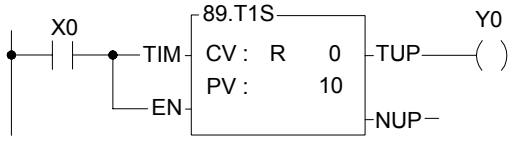
CV : Register storing elapse time (current value)

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 or 0～2147483647
CV		○	○	○	○	○	○	○	○	○*	○*	○	
PV	○	○	○	○	○	○	○	○	○	○	○	○	○

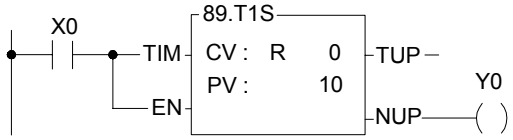
- The operation for this instruction is the same as that for the basic timer (T0~T255), except that the basic timer only has a "timing control" input - when its input is 1 it starts timing, and when input is 0 it get clear. Every time the input changes, it starts timing again and is unable to accumulate. 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).

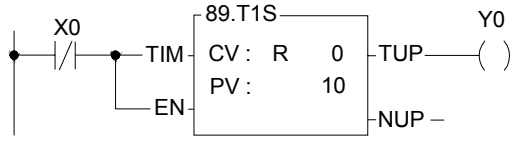
- On delay de-energizing timer:



- 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).

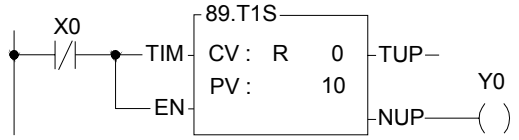
FUN89/FUN89D (T1S) FUN88/FUN88D (T.1S) FUN87/FUN87D (T.01S)	CUMULATIVE TIMER	FUN89/FUN89D (T1S) FUN88/FUN88D (T.1S) FUN87/FUN87D (T.01S)

- 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.