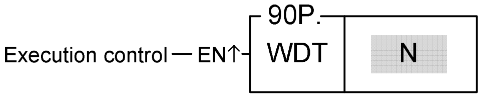
FUN 90 P WDT	WATCHDOG TIMER	FUN 90 P WDT

Ladder symbol



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

- When execution control "EN" = 1 or "EN ↑ " (P instruction) transition from 0 to 1, will set the watchdog time to Nx10ms. If the scan time exceeds this preset time, PLC will shut down and not execute the application program.

- The WDT feature is designed mainly as a safety consideration from the system view for the application. For example, if the CPU of PLC is suddenly damaged, and there is no way to execute the program or refresh I/O, then after the WDT time expired, the WDT will automatically switch off all the I/Os, so as to ensure safety. In certain applications, if the scan time is too long, it may cause safety problems or problems of non-conformance with control requirements. This instruction can used to establish the limitation of the scan time that you require.

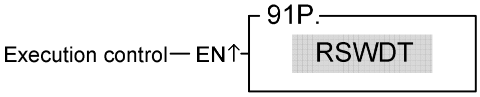
- Once the WDT time has been set it will always be kept, and there is no need to set it again on each scan. Therefore, in practice this instruction should use the P instruction.

- Default WDT time is 0.25 sec.

- For the operation principles of WDT please refer to the RSWDT(FUN 91) instruction.

FUN 91 P RSWDT	RESET WATCHDOG TIMER	FUN 91 P RSWDT

Ladder symbol



This instruction has no operand.

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

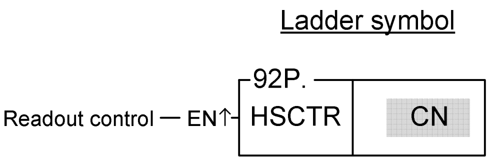
- The functions of WDT have already been described in FUN90 (WDT instruction).
  The operation principles of watch dog timer are as follows:

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

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

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

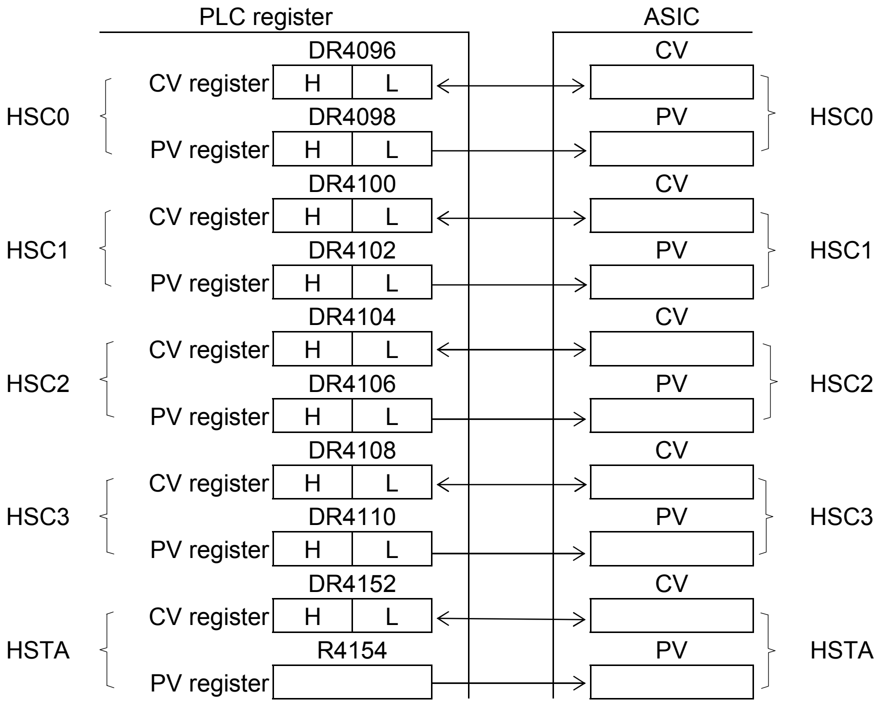
Ladder symbol



CN : Hardware high speed counter number

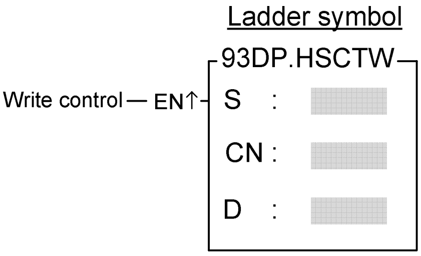
0: SC0 or HST0
1: SC1 or HST1
2: SC2 or HST2
3: SC3 or HST3
4: STA

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



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

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



S : The source data for writing

CN : Hardware high speed counter to be written

0: HSC0 or HST1

1: HSC1 or HST2

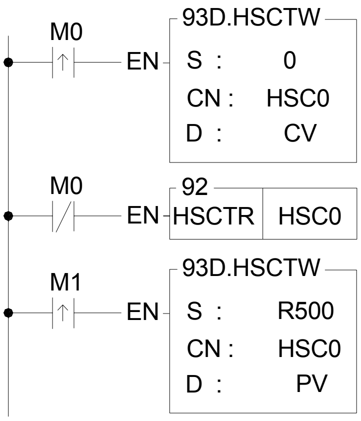
2: HSC2 or HST3

3: HSC3 or HST4

4: HSTA

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

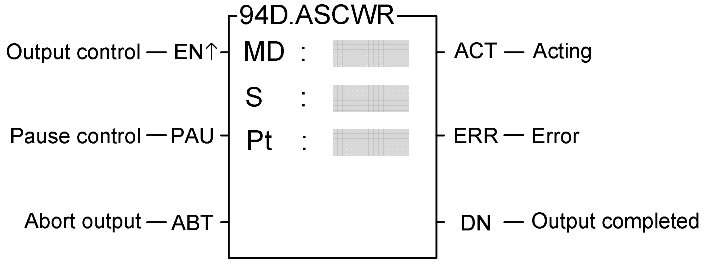
- Please refer first to FUN92 for the relation between the CV or PV value of HSC0～HSC3 and HSTA within ASIC and their corresponding CV and PV registers in CPU.
- When write control "EN"=1 or "EN ↑ " (P instruction) changes from 0→1, it writes the content of CV or PV register of high speed counter designed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



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

FUN 94 ASCWR	ASCII WRITE	FUN 94 ASCWR

Ladder symbol



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

S : Starting register of file data.

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

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

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

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

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

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

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

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

- or detail applications, please refer to chapter 14 “The Application of ASCII file output function”.

FUN 94 ASCWR	ASCII WRITE	FUN 94 ASCWR

- Interface signals:

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

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

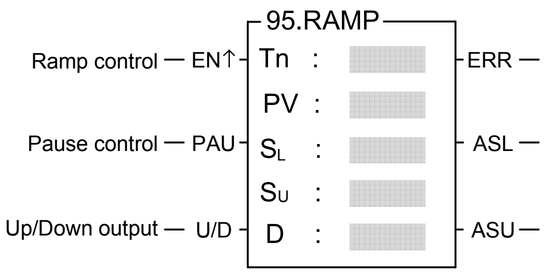
: OFF, it represents that the RTS of the Printer is “True”; Printer is Ready.

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

R4158: The setting of communication parameters (refer to section 11.7.2)

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

Ladder symbol



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

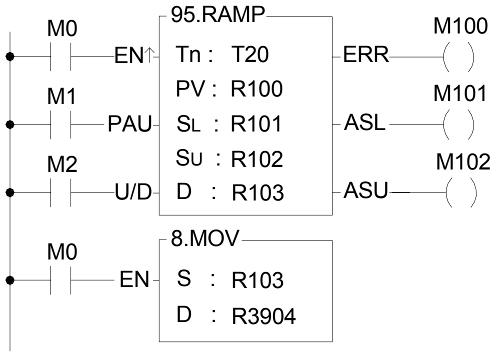
Range Ope- rand	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K
	WX0 | WX240	WY0 | WY240	WM0 | WM1896	WS0 | WS984	T0 | T255	C0 | C255	R0 | R3839	R3840 | R3903	R3904 | R3967	R3968 | R4167	R5000 | R8071	D0 | D4095	16-bit +/- number
Tn					○								
PV	○	○	○	○	○	○	○	○	○	○	○	○	○
S_L	○	○	○	○	○	○	○	○	○	○	○	○	○
S_U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

Description

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

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

Program example



Move the ramping value to AO output register R3904

T20: Ramp timer (timer with 0.01 second time base)

R100: preset value of ramp timer (the unit is 0.01 second, 100 for a second).

R101: Lower limit value.

R102: Upper limit value.

R103: Register storing current ramp value.

R104: Working register

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

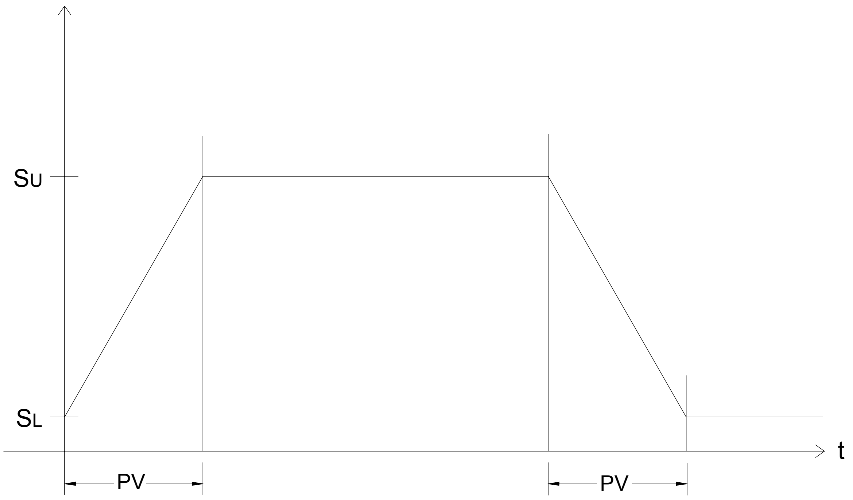


Table Instructions

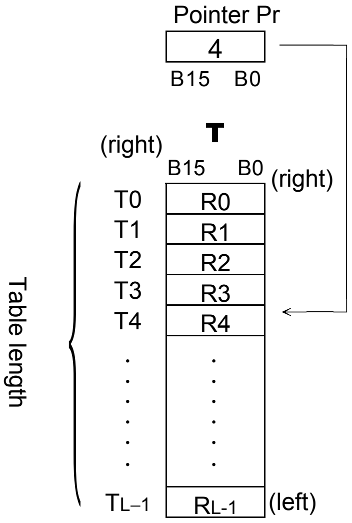
Fun No.	Mnemonic	Functionality	Fun No.	Mnemonic	Functionality
100	R→T	Register to table data move	107	T_FIL	Table fill
101	T→R	Table to register data move	108	T_SHF	Table shift
102	T→T	Table to table data move	109	T_ROT	Table rotate
103	BT_M	Block table move	110	QUEUE	Queue
104	T_SWP	Block table swap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

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

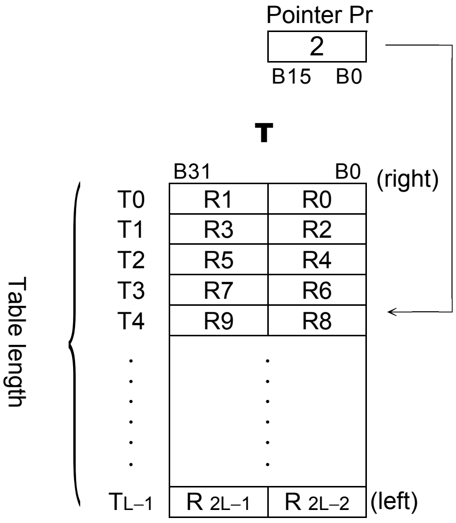
- The operation of table instructions are used mostly for data processing such as move, copy, compare, search etc, between tables and registers, or between tables. These instructions are convenient for application.

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

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



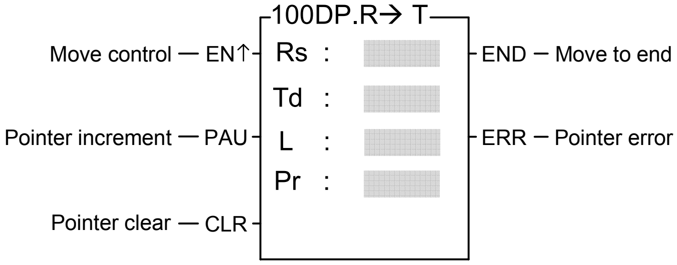
16bit table



32bit table

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

Ladder symbol



Rs : Source data , can be constant or register

Td : Source register for destination table

L : Length of destination table

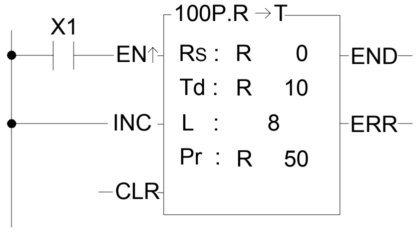
Pr : Pointer register

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

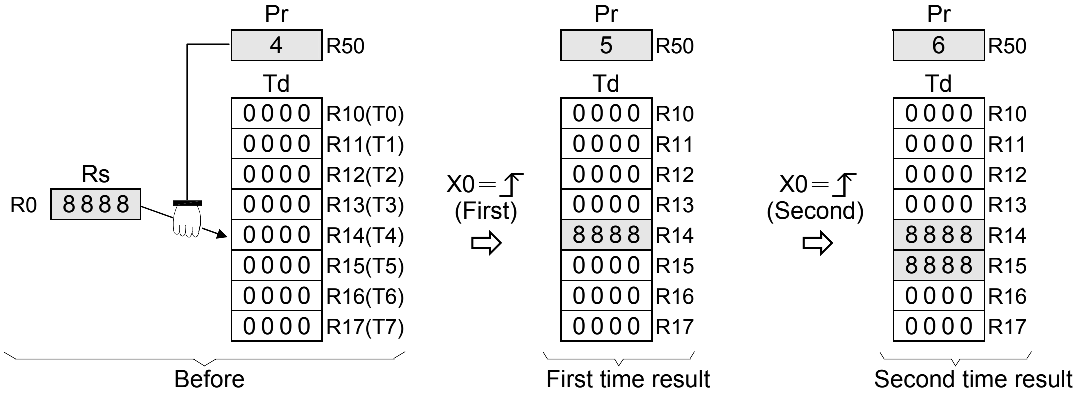
Range / Ope-rand	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR
	WX0 | WX240	WY0 | WY240	WM0 | WM1896	WS0 | WS984	T0 | T255	C0 | C255	R0 | R3839	R3840 | R3903	R3904 | R3967	R3968 | R4167	R5000 | R8071	D0 | D4095	16/32bit +/- number	V、Z P0~P9
Rs	○	○	○	○	○	○	○	○	○	○	○	○	○	○
Td		○	○	○	○	○	○		○	○*	○*	○		○
L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

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

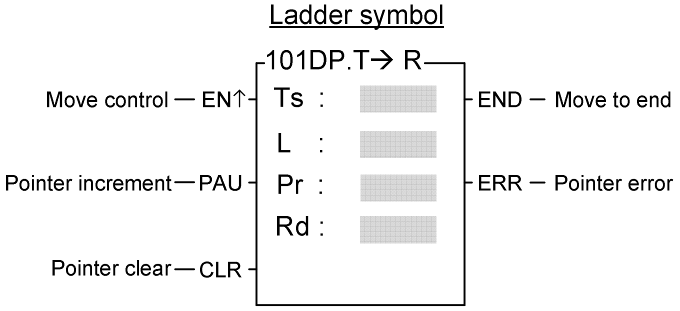
- The effective range of the pointer is 0 to L-1. Beyond this range, the pointer error "ERR" will be set to 1, and this instruction will not be performed.



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



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

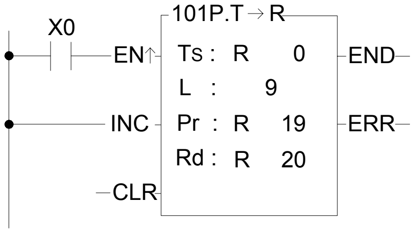


Ts : Source table starting register
L : Length of source table
Pr : Pointer register
Rd : Destination register
Ts, Rd may combine with V, Z, P0~P9 to serve indirect address application

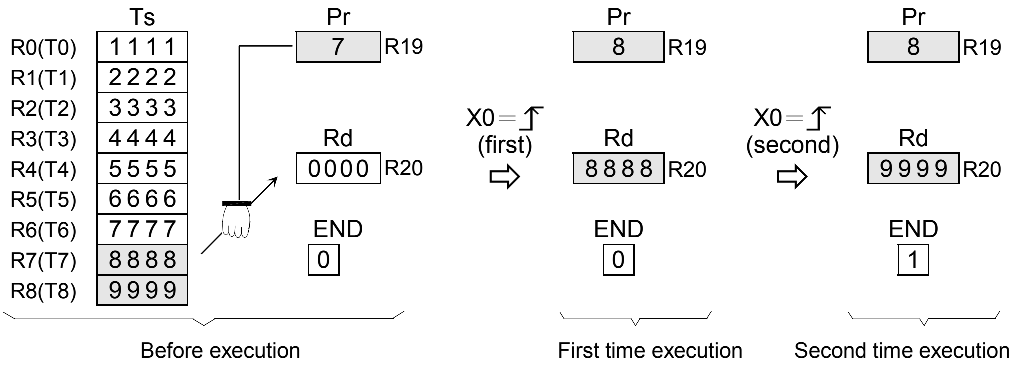
Range Ope-rand	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR
	WX0 | WX240	WY0 | WY240	WM0 | WM1896	WS0 | WS984	T0 | T255	C0 | C255	R0 | R3839	R3840 | R3903	R3904 | R3967	R3968 | R4167	R5000 | R8071	D0 | D4095	16/32bit +/- number	V、Z P0~P9
Ts	○	○	○	○	○	○	○	○	○	○	○	○		○
L							○				○*	○		○
Pr		○	○	○	○	○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

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

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

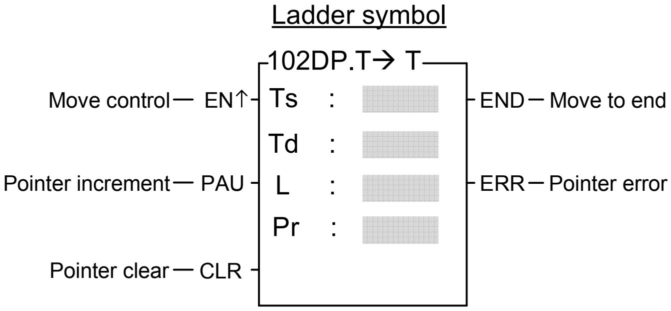


- In the example at left, at the very beginning Pr = 7 and Ts and Rd are as shown at left in the diagram below. When X0 have a transition from 0→1 twice, the results are shown at right in the diagram below.
- At the second time execution, the pointer has already reached to the end so there will be no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



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

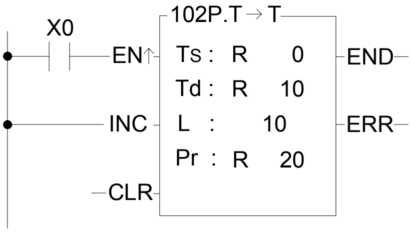
Ladder symbol



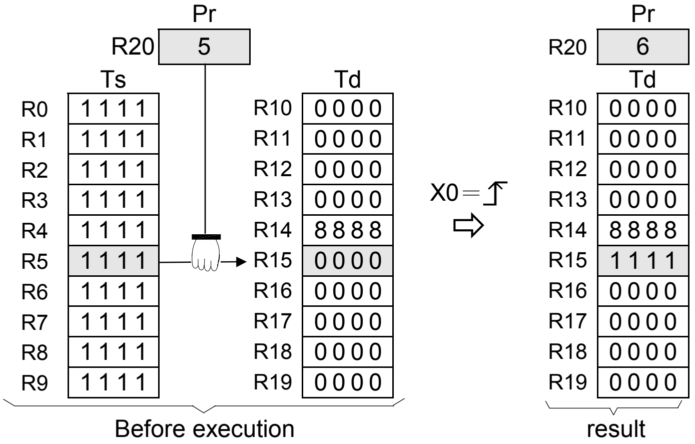
Ts : Starting number of source table register
Td : Starting number of destination table register
L : Table (Ts and Td) length
Pr : Pointer register
Ts, Rd may combine with V, Z, P0~P9 to serve indirect address application

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

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



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

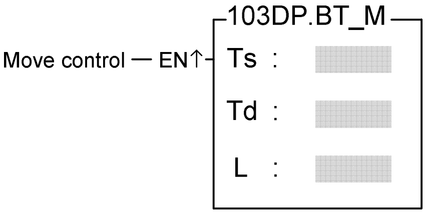


Before execution

result

FUN103 D P BT_M	BLOCK TABLE MOVE	FUN103 D P BT_M

Ladder symbol



Ts :Starting register for source table

Td : Starting register for destination table

L: Lengths of source and destination tables

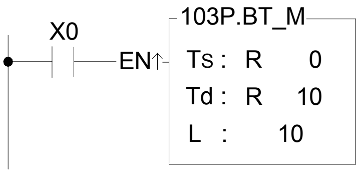
Ts, Rd may combine with V, Z, P0~P9 to serve indire

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

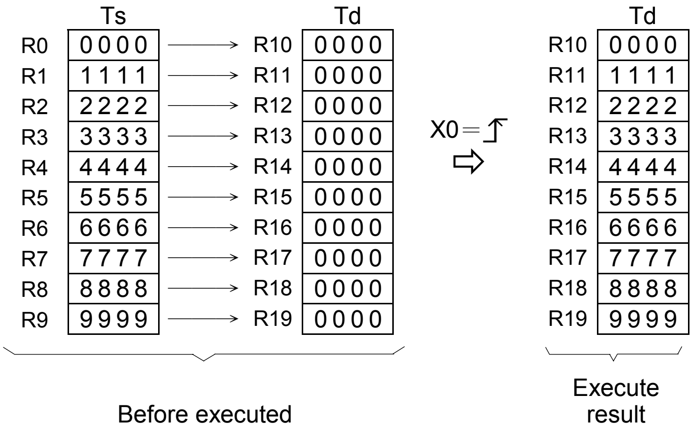
- In this instruction the source table and destination table are the same length. When this instruction was executed all the data in the Ts table is completely copied to Td. No pointer is involved in this instruction.

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

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



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



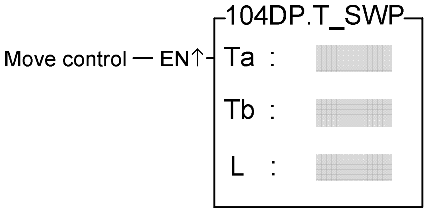
	Ts		Td		Td
R0	0000	R10	0000	R10	0000
R1	1111	R11	0000	R11	1111
R2	2222	R12	0000	R12	2222
R3	3333	R13	0000	R13	3333
R4	4444	R14	0000	R14	4444
R5	5555	R15	0000	R15	5555
R6	6666	R16	0000	R16	6666
R7	7777	R17	0000	R17	7777
R8	8888	R18	0000	R18	8888
R9	9999	R19	0000	R19	9999

Before executed

Execute result

FUN104 D P T_SWP	BLOCK TABLE SWAP	FUN104 D P T_SWP

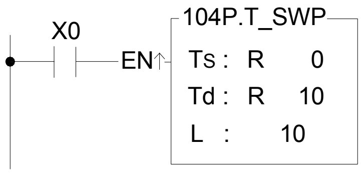
Ladder symbol



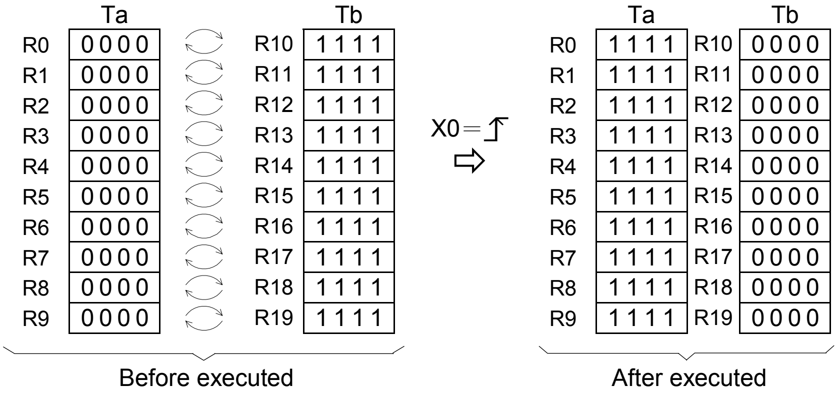
Ta : Starting register of Table a
Tb : Starting register of Table b
L : Lengths of Table a and b
Ts, Rd may combine with V, Z, P0~P9 to serve indirect address application

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

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



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



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

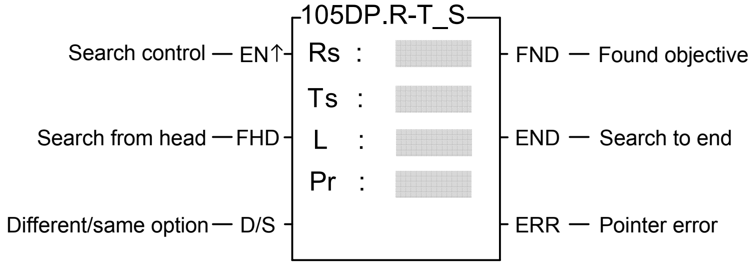
Before executed

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

After executed

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

Ladder symbol

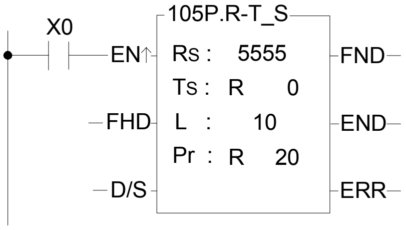


Rs : Data to search, It can be a constant or a register
Ts : Starting register of table being searched
L : Label length
Pr : Pointer of table
Rs, Ts may combine with V, Z, P0~P9 to serve indirect address application

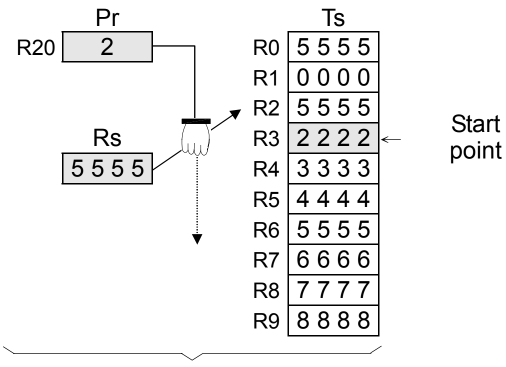
Range Ope- rand	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR
	WX0 ｜ WX240	WY0 ｜ WY240	WM0 ｜ WM1896	WS0 ｜ WS984	T0 ｜ T255	C0 ｜ C255	R0 ｜ R3839	R3840 ｜ R3903	R3904 ｜ R3967	R3968 ｜ R4167	R5000 ｜ R8071	D0 ｜ D4095	16/32-bit +/- number	V、Z P0~P9
Rs	○	○	○	○	○	○	○	○	○	○	○	○	○	○
Ts	○	○	○	○	○	○	○	○	○	○	○	○		○
L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

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

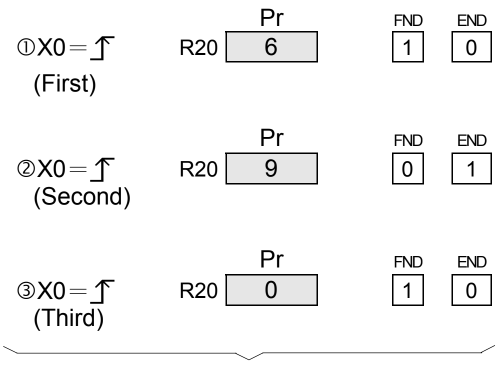
- The effective range of Pr is 0 to L-1. If the value exceeds this range then the pointer error flag "ERR" will change to 1, and this instruction will not be carried out.



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



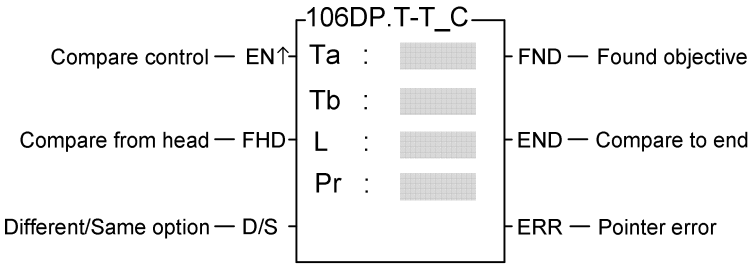
Before execution



After execution

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

Ladder symbol

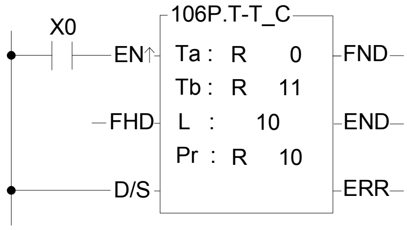


Ta : Starting register of Table a
Tb : Starting register of Table b
L : Lengths of Table
Pr : Pointer
Ta, Tb may combine with V, Z, P0~P9 to serve indirect address application

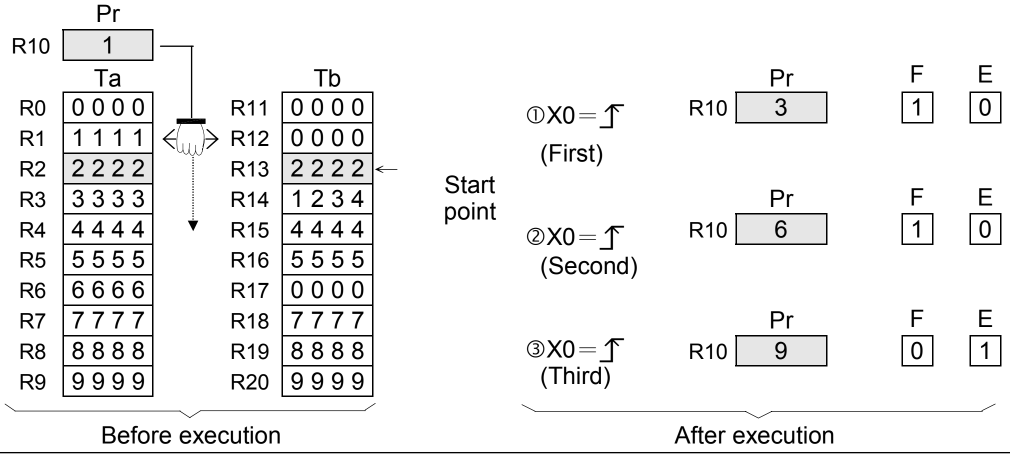
Range / Operand	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR
	WX0 | WX240	WY0 | WY240	WM0 | WM1896	WS0 | WS984	T0 | T255	C0 | C255	R0 | R3839	R3840 | R3903	R3904 | R3967	R3968 | R4167	R5000 | R8071	D0 | D4095	2 | 256	V、Z P0~P9
Ta	○	○	○	○	○	○	○	○	○	○	○	○		○
Tb	○	○	○	○	○	○	○	○	○	○	○	○		○
L							○				○*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

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

- The effective range of Pr is 0 to L-1. The Pr value should not changed by other programs during the operation. As this will affect the result of the search. If the Pr value not in the effective range, the pointer error flag "ERR" will be set to 1, and this instruction will not be carried out.



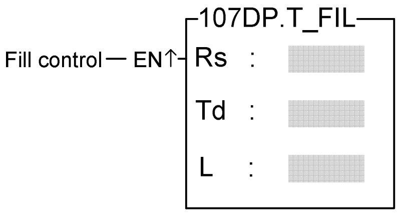
- The instruction at left starts from the register next to the register pointed by the pointer (because "FHD" is 0) to search for register pairs with different data (because "D/S" is 1) within the 2 tables. At the very beginning, Pr points to Ta1 and Tb1. There are 3 different pairs of data at the position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



	Ta		Tb
R0	0000	R11	0000
R1	1111	R12	0000
R2	2222	R13	2222
R3	3333	R14	1234
R4	4444	R15	4444
R5	5555	R16	5555
R6	6666	R17	0000
R7	7777	R18	7777
R8	8888	R19	8888
R9	9999	R20	9999

FUN107 D P T_FIL	TABLE FILL	FUN107 D P T_FIL

Ladder symbol



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

Td : Starting register of destination table

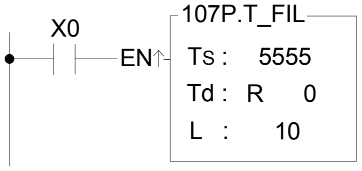
L :Table length

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

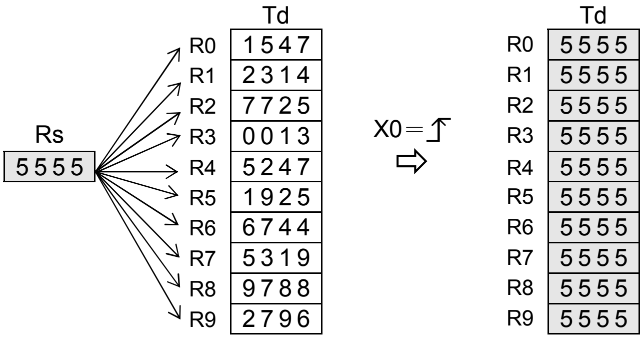
Range / Ope-rand	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K	XR
	WX0 | WX240	WY0 | WY240	WM0 | WM1896	WS0 | WS984	T0 | T255	C0 | C255	R0 | R3839	R3840 | R3903	R3904 | R3967	R3968 | R4167	R5000 | R8071	D0 | D4095	16/32-bit +/- number	V、Z P0~P9
Ts	○	○	○	○	○	○	○	○	○	○	○	○	○	○
Td		○	○	○	○	○	○		○	○*	○*	○		○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

- When fill control "EN" = 1 or "EN ↑ " (P instruction) has a transition from 0 to 1, the Rs data will be filled into all the registers of the table Td.

- This instruction is mainly used for clearing the table (fill 0) or unifying the table (filling in the same values). It should be used with the P instruction.



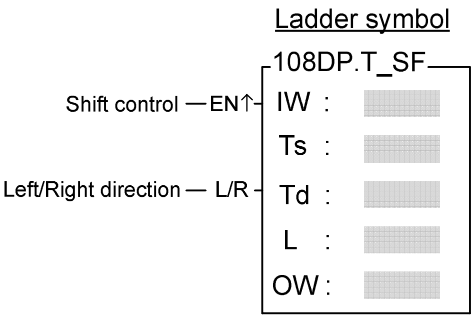
- The instruction at left will fill 5555 into the whole table Td. The results are as shown in the diagram below.



Before execution After execution

FUN108 D P T_SHF	TABLE SHIFT	FUN108 D P T_SHF

Ladder symbol



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

Ts : Source table

Td : Destination table storing shift results

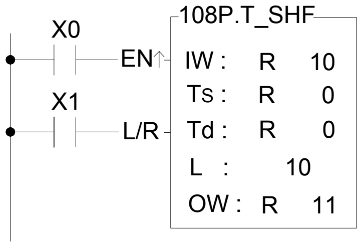
L : Lengths of tables Ts and Td

OW : Register to accept the shifted out data

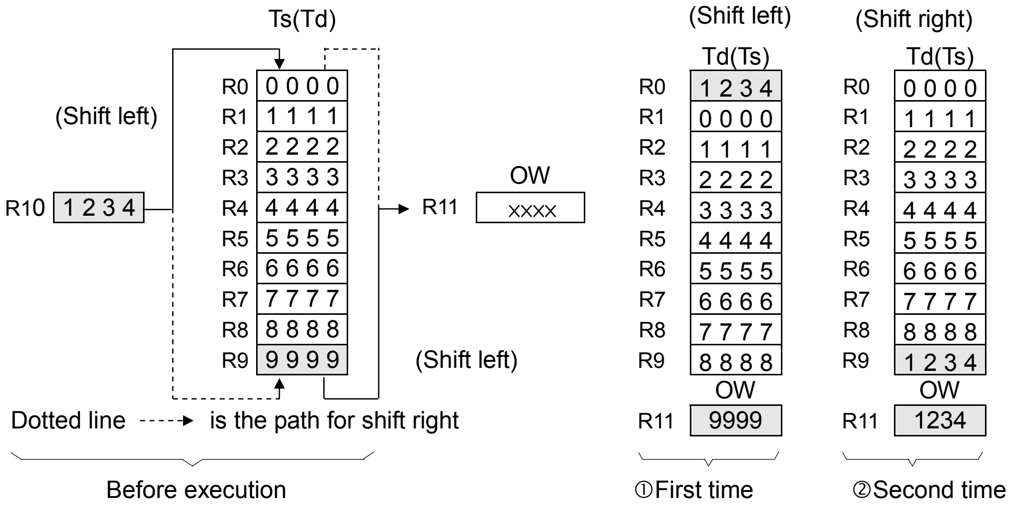
Ts, Td may combine with V, Z, P0~P9 to serve indirect address application

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

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

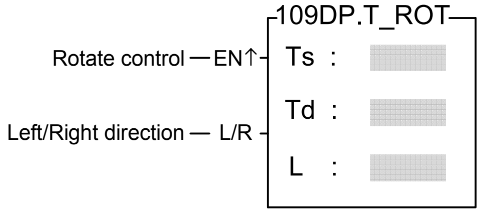


- In the program at left, Ts and Td is the same table. Therefore, the table shifts itself and then writes back to itself (the table must be writ able). It first perform a shift left operation (let X1 = 1, and X0 go from 0→1) then perform a shift to right operation (let X1 = 0, and makes X0 go from 0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



FUN109 D P T_ROT	TABLE ROTATE	FUN109 D P T_ROT

Ladder symbol



Ts : Source table for rotate

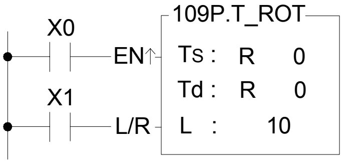
Td : Destination table storing results of rotation

L : Lengths of table

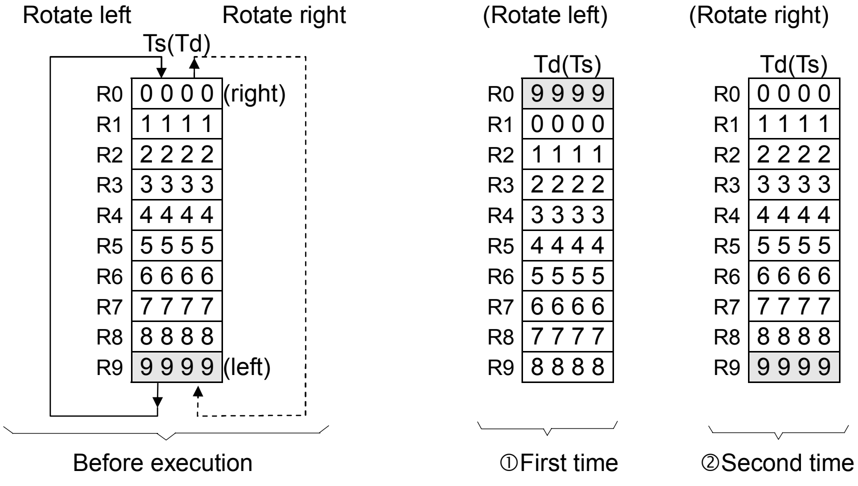
Ts, Td may combine with V, Z, P0~P9 to serve indirect address application

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

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



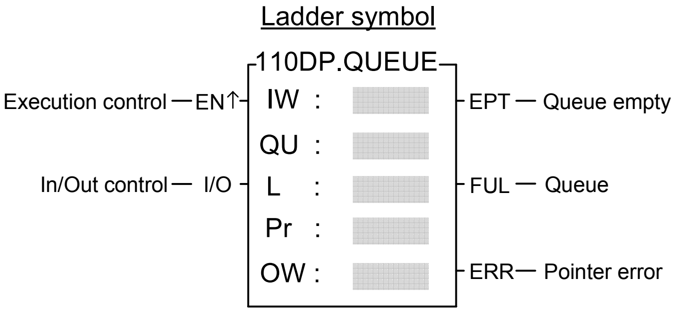
- In the program at left, Ts and Td is the same table. The table after rotation will write back to itself. It first perform one left rotation (let X1 = 1, and X0 go from 0→1), and then performs one right rotation (let X1 = 0, and X0 go from 0→1). The results are shown at right in the diagram below.



	Before execution	①First time	②Second time
R0	0 0 0 0	9 9 9 9	0 0 0 0
R1	1 1 1 1	0 0 0 0	1 1 1 1
R2	2 2 2 2	1 1 1 1	2 2 2 2
R3	3 3 3 3	2 2 2 2	3 3 3 3
R4	4 4 4 4	3 3 3 3	4 4 4 4
R5	5 5 5 5	4 4 4 4	5 5 5 5
R6	6 6 6 6	5 5 5 5	6 6 6 6
R7	7 7 7 7	6 6 6 6	7 7 7 7
R8	8 8 8 8	7 7 7 7	8 8 8 8
R9	9 9 9 9	8 8 8 8	9 9 9 9

FUN110 D P QUEUE	QUEUE	FUN110 D P QUEUE

Ladder symbol



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

QU : Starting register of queue

L : Size of queue

Pr : Pointer register

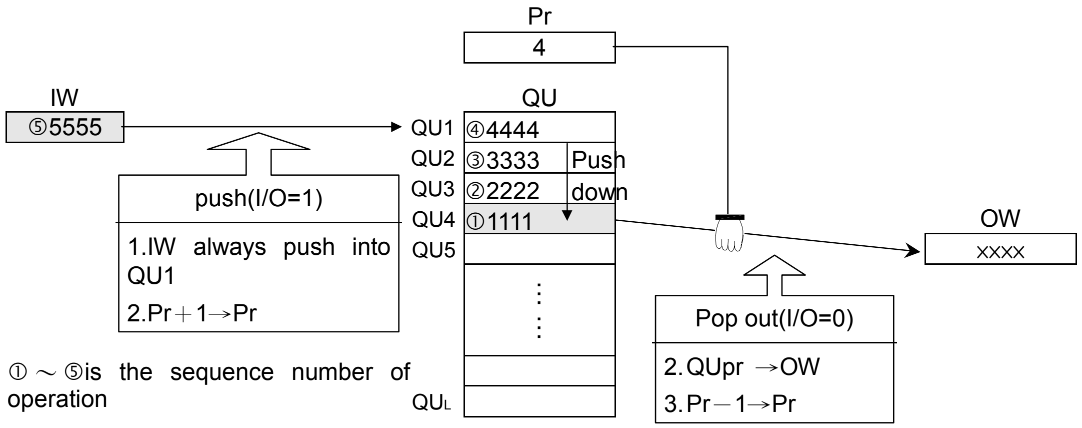
OW : Register accepting data popped out from queue

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

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

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

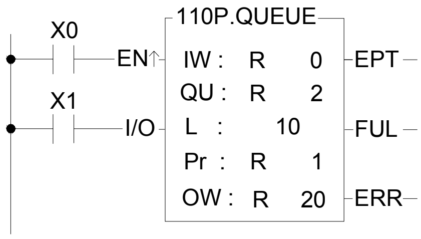
- Queue is a first in first out (FIFO) device, i.e. - the data that first pushed into the queue will be the first to pop out from the queue. A queue is comprised of L consecutive 16 or 32 bit registers (D instruction) starting from the QU register, as in the diagram below:



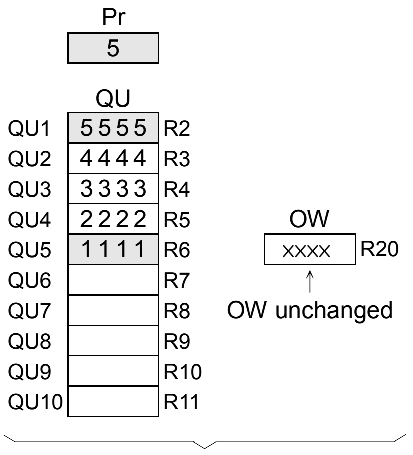
- When execution control "EN" = 1 or "EN↑" (P instruction) has a transition from 0 to 1, the status of in/out control "I/O" determines whether the IW data will be pushed into the queue (when "I/O" = 1) or be popped out and transferred to OW (when "I/O" = 0). As shown in the diagram above, the IW data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

FUN110 D P QUEUE	QUEUE	FUN110 D P QUEUE

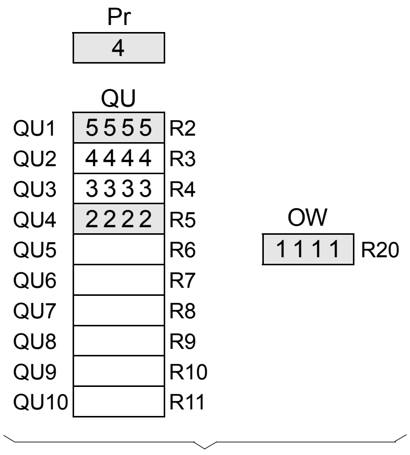
- If no data has yet been pushed into the queue or the pushed in data has already been popped out (Pr = 0), then the queue empty flag will be set to 1. In this case, even if there is further popping out action, this instruction will not be executed. If data is only pushed in and not popped out, or pushed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



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



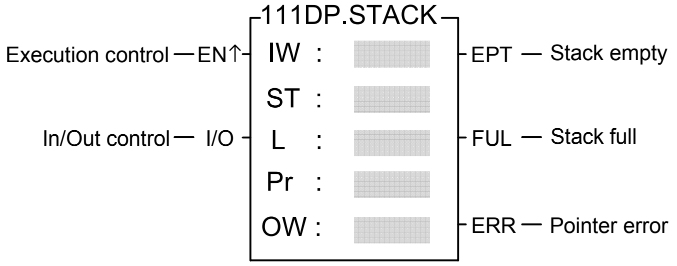
After push in (X1=1，X0 from 0→1)



After pop off (X1=0，X0 from 0→1)

FUN111 D P STACK	STACK	FUN111 D P STACK

Ladder symbol



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

ST : Starting register of stack

L : Size of stack

Pr : Pointer register

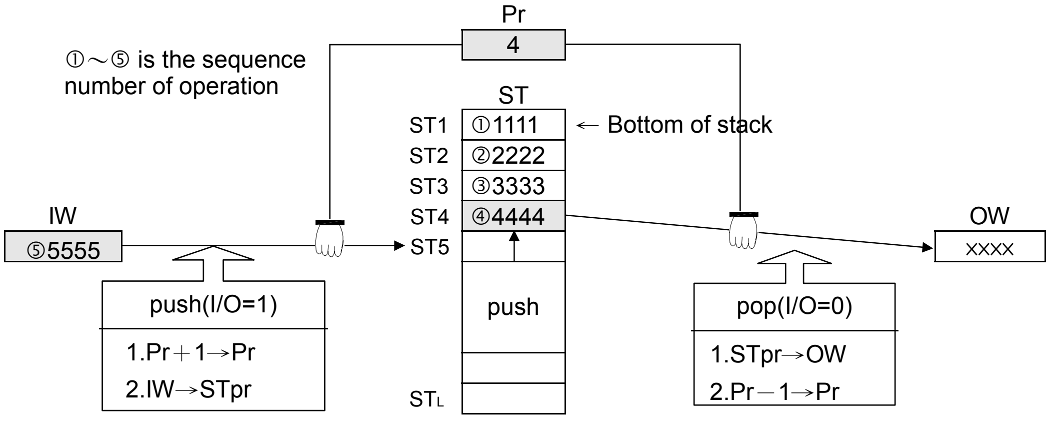
OW : Register accepting data popped out from stack

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

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

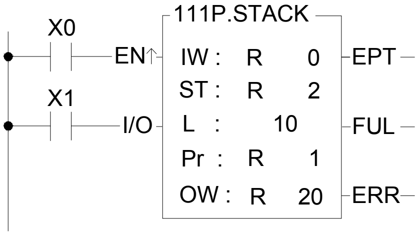
- Like queue, stack is also a kind of table. The nature of its pointer is exactly the same as with queue, i.e. Pr = 1 to L, which corresponds to ST_1 to ST_L, and when Pr = 0 the stack is empty.

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

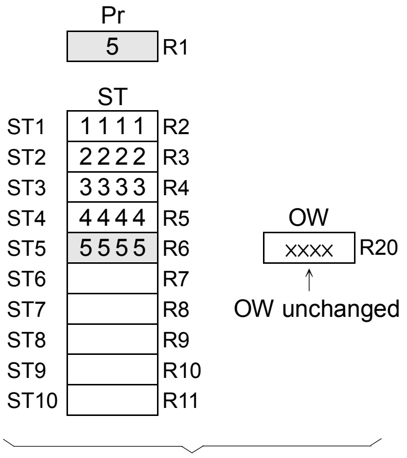


- When execution control "EN" = 1 or "EN ↑ " (P instruction) has a transition from 0 to 1, the status of in/out control "I/O" determines whether the IW data will be pushed into the stack (when "I/O" = 1), or the data pointed by Pr within the stack (the data most recently pushed into the stack) will be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

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

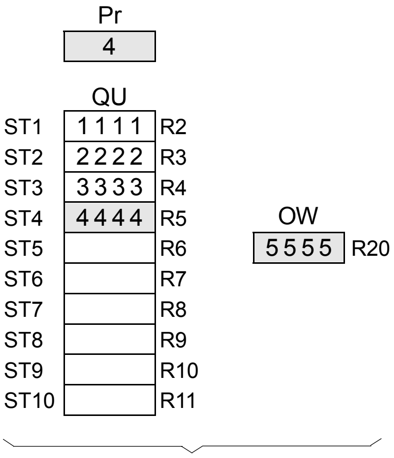


- The program at left assumes that the initial content of the stack is just as in the diagram of a stack on the preceding page. The operation illustrated in this example is to push a data and than pop it from stack. The results are shown below. Under any circumstances, Pr always point to the data that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



	ST		OW	
Pr	5	R1		
ST1	1111	R2		
ST2	2222	R3		
ST3	3333	R4		
ST4	4444	R5		
ST5	5555	R6	xxxx	R20
ST6		R7	↑	
ST7		R8	OW unchanged	
ST8		R9		
ST9		R10		
ST10		R11		

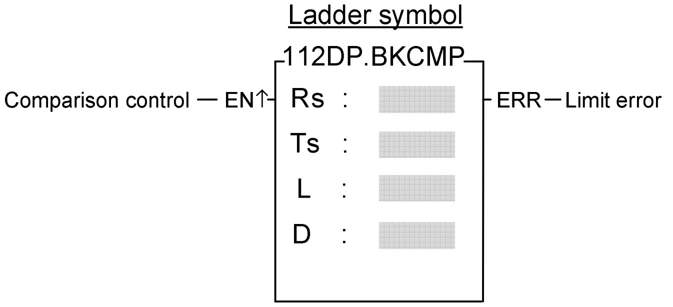
After push(X1=1，X0 from 0→1)



	QU		OW	
Pr	4			
ST1	1111	R2		
ST2	2222	R3		
ST3	3333	R4		
ST4	4444	R5		
ST5		R6	5555	R20
ST6		R7		
ST7		R8		
ST8		R9		
ST9		R10		
ST10		R11		

After pop up(X1=0，X0 from 0→1)

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



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

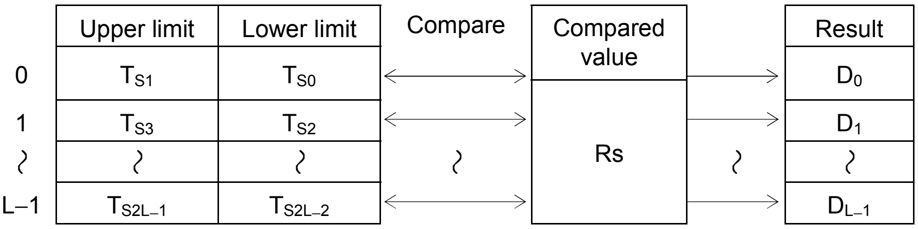
Ts : Starting register block storing upper and lower limit

L : Number of pairs of upper and lower limits

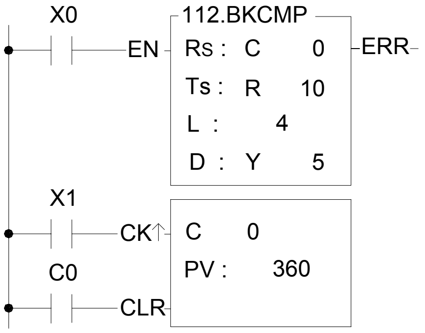
D : Starting relay storing results of comparison

Range / Ope-rand	Y	M	S	WX	WY	WM	WS	TMR	CTR	HR	IR	OR	SR	ROR	DR	K
	Y0 ∣ Y255	M0 ∣ M999	S0 ∣ S999	WX0 ∣ WX240	WY0 ∣ WY240	WM0 ∣ WM1896	WS0 ∣ WS984	T0 ∣ T255	C0 ∣ C255	R0 ∣ R3839	R3840 ∣ R3903	R3904 ∣ R3967	R3968 ∣ R4167	R5000 ∣ R8071	D0 ∣ D4095	16/32-bit +/- number
Rs				○	○	○	○	○	○	○	○	○	○	○	○	○
Ts				○	○	○	○	○	○	○	○	○	○	○	○	
L										○				○*	○	1~256
D	○	○	○													

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



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

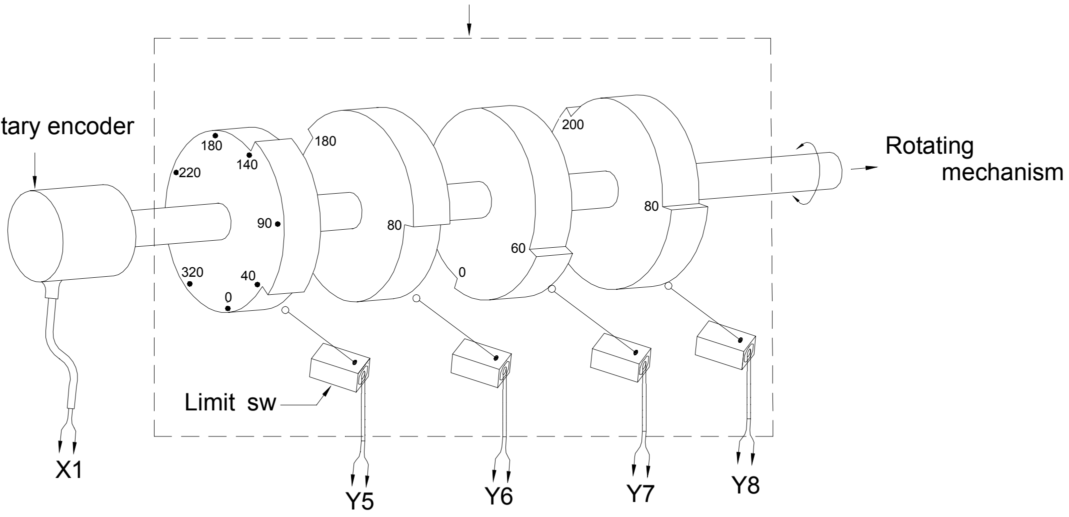


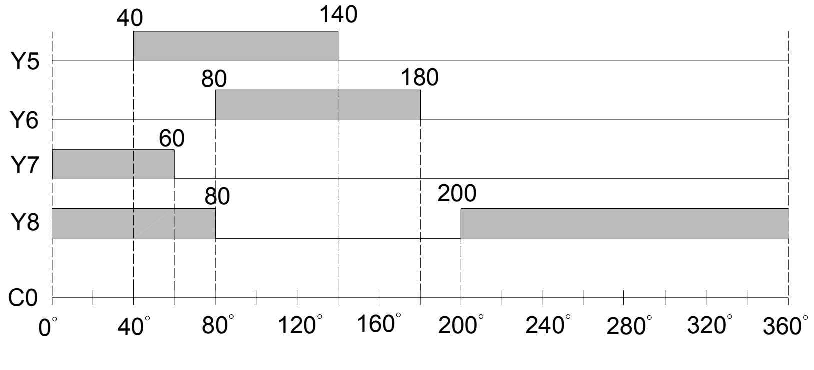
- In this program, C0 represents the rotation angle (Rs) of a drum shaft. The block compare instruction performs a comparison between Rs and the 4 pairs (L = 4) of upper and lower limits, R10,R11, R12,R13, R14,R15 and R16,R17. The comparison results can be obtained from the four drum output points Y5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

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

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

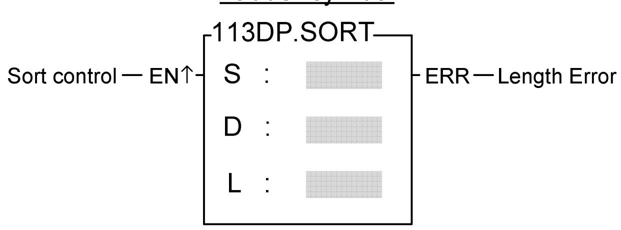
Equivalent mechanical drum emulated by above program





FUN113 D P SORT	DATA SORTING	FUN113 D P SORT

Ladder symbol



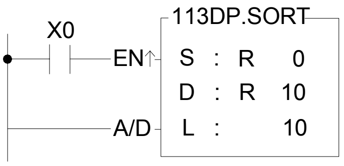
S : Starting register of source registers to sort

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

L : Total register for sorting

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

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



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

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

Before

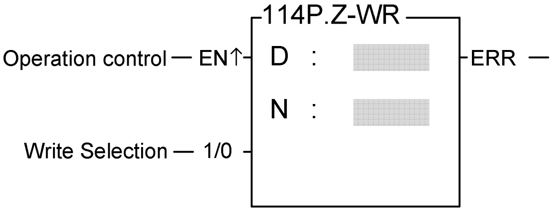
X0 = ↑ ⇨

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

After

FUN114 P Z-WR	ZONE WRITE	FUN114 P Z-WR

Ladder symbol



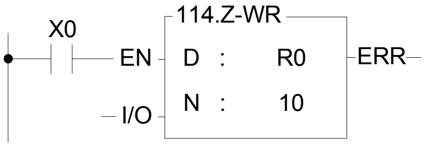
D : Starting address of being set or reset

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

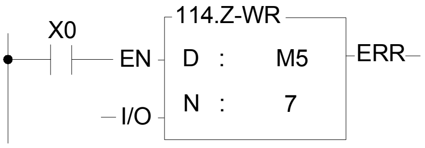
D 、N operand can combine V 、Z 、P0~P9 for index addressing while word operation

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

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



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

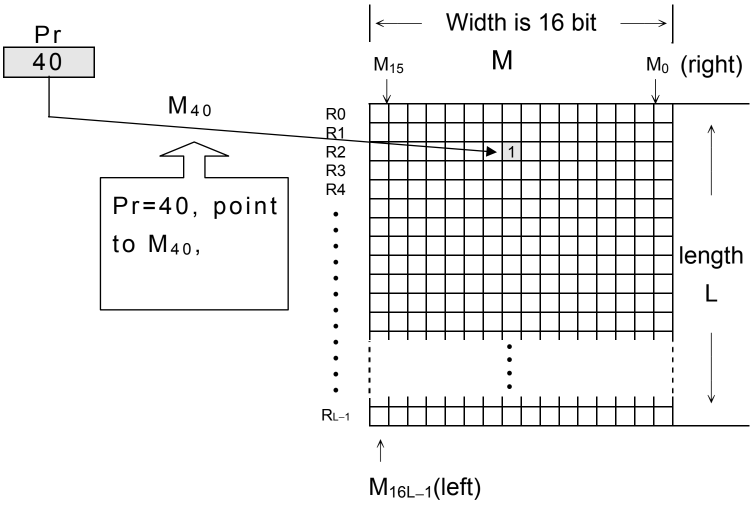


• Above example, bits M5~M11 will be reset to 0 while X0=1.

Matrix Instructions

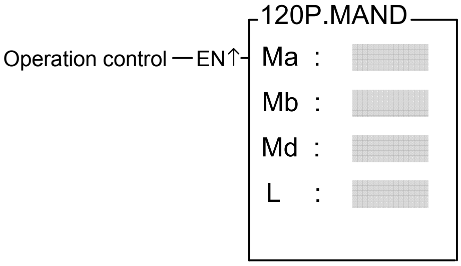
Fun No.	Mnemonic	Functionality	Fun No.	Mnemonic	Functionality
120	MAND	Matrix AND	126	MBRD	Matrix Bit Read
121	MOR	Matrix OR	127	MBWR	Matrix Bit Write
122	MXOR	Matrix XOR	128	MBSHF	Matrix Bit Shift
123	MXNR	Matrix XNOR	129	MBROT	Matrix Bit Rotate
124	MINV	Matrix Inverse	130	MBCNT	Matrix Bit Count
125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

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



FUN120 P MAND	MATRIX AND	FUN120 P MAND

Ladder symbol



Ma : Starting register of source matrix a

Mb : Starting register of source matrix b

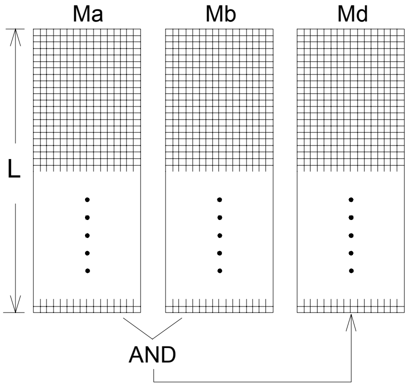
Md : Starting register of destination matrix

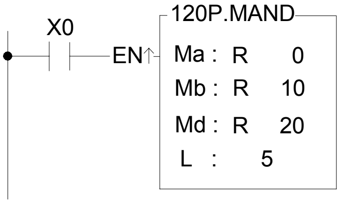
L : Length of matrix (Ma, Mb and Md)

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

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

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





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

Ma

Ma_{15} ↓ … ↓ Ma_0

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

Ma_{79} ↑ … ↑ Ma_{64}

Mb

Mb_{15} ↓ … ↓ Mb_0

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

Mb_{79} ↑ … ↑ Mb_{64}

Before execution

Md

Md_{15} ↓ … ↓ Md_0

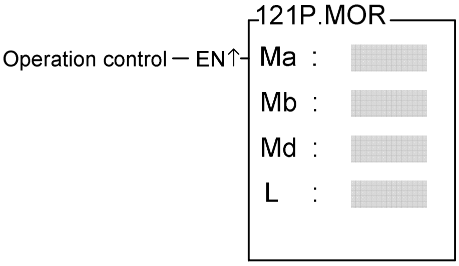
R20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R21	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R22	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
R23	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R24	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Md_{79} ↑ … ↑ Md_{64}

After execution

FUN121 P MOR	MATRIX OR	FUN121 P MOR

Ladder symbol



Ma : Starting register of source matrix a

Mb : Starting register of source matrix b

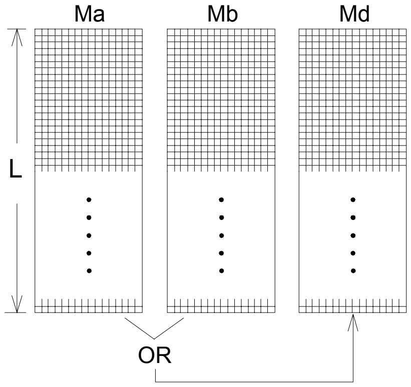
Md : Starting register of destination matrix

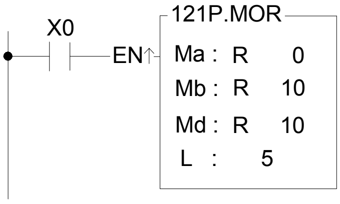
L : Length of matrix (Ma, Mb and Md)

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

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

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





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

Ma

	Ma_{15}															Ma_0
R0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0
R2	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
R3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R4	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	Ma_{79}															Ma_{64}

Mb

	Mb_{15}															Mb_0
R10	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
R11	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
R12	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
R13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R14	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	Mb_{79}															Mb_{64}

Before execution

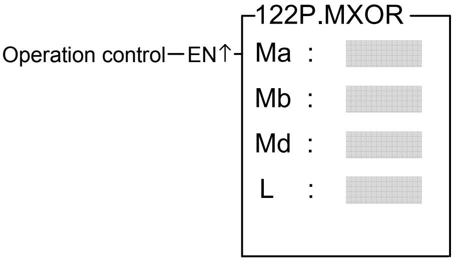
Md

	Md_{15}															Md_0
R20	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
R21	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
R22	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
R23	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R24	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	Md_{79}															Md_{64}

After execution

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

Ladder symbol



Ma: Starting register of source matrix a

Mb: Starting register of source matrix b

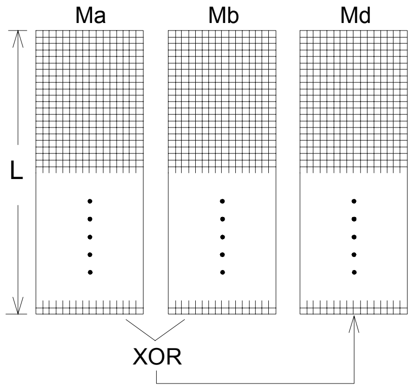
Md: Starting register of destination matrix

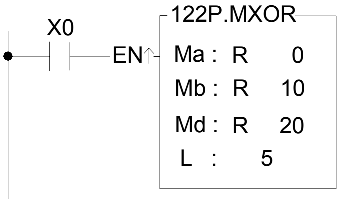
L : Length of matrix (Ma, Mb and Md)

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

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

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





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

Ma

	Ma_{15}															Ma_0
R0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0
R2	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
R3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R4	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	Ma_{79}															Ma_{64}

Mb

	Mb_{15}															Mb_0
R10	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
R11	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
R12	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
R13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R14	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	Mb_{79}															Mb_{64}

Before execution

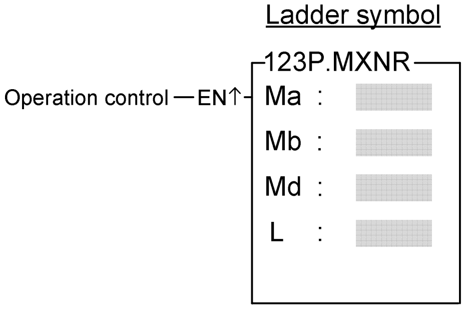
Md

	Md_{15}															Md_0
R20	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
R21	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
R22	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R23	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R24	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	Md_{79}															Md_{64}

After execution

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

Ladder symbol



Ma : Starting register of source matrix a

Mb : Starting register of source matrix b

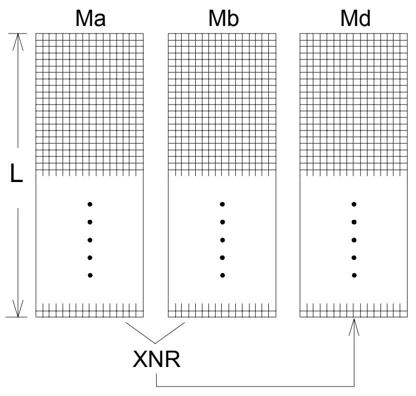
Md : Starting register of destination matrix

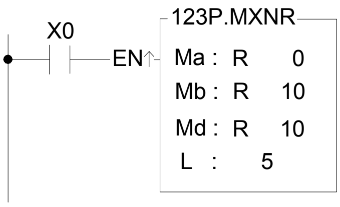
L : Length of matrix (Ma, Mb and Md)

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

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

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





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

Ma

	Ma_{15}															Ma_0
R0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0
R2	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
R3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R4	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	Ma_{79}															Ma_{64}

Mb

	Mb_{15}															Mb_0
R10	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
R11	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
R12	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
R13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R14	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	Mb_{79}															Mb_{64}

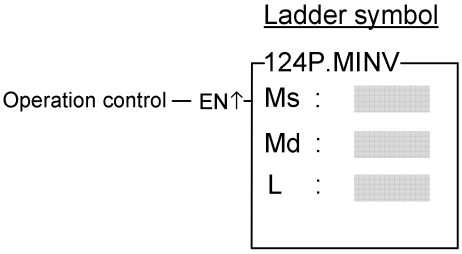
Before execution

Md

	Md_{15}															Md_0
R20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R21	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R22	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
R23	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
R24	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	Md_{79}															Md_{64}

After execution

FUN124 P MINV	MATRIX INVERSE	FUN124 P MINV



Ms : Starting register of source matrix

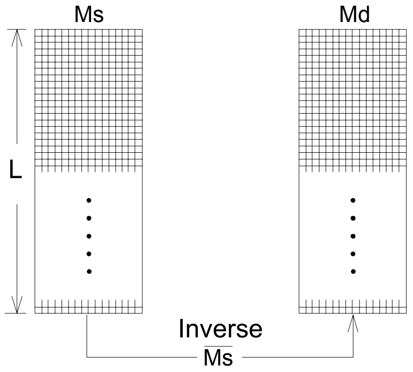
Md : Starting register of destination

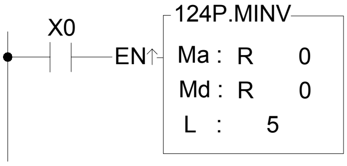
L : Length of matrix (Ms and Md)

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

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

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





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

Ms

	Ms_{15}															Ms_0
R0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0
R2	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
R3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R4	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	Ms_{79}															Ms_{64}

Before execution

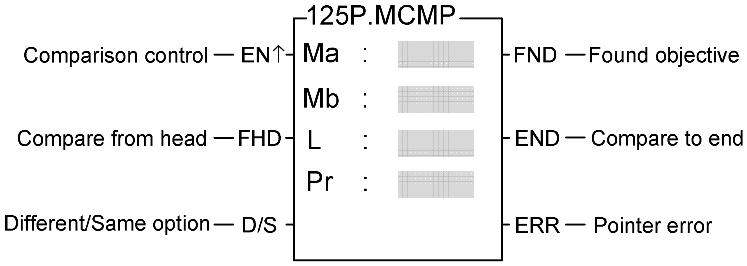
Md

	Md_{15}															Md_0
R0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
R1	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
R2	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0
R3	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
R4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	Md_{79}															Md_{64}

After execution

FUN125 P MCMP	MATRIX COMPARE	FUN125 P MCMP

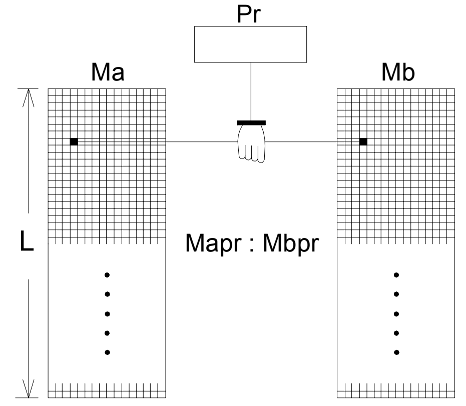
Ladder symbol



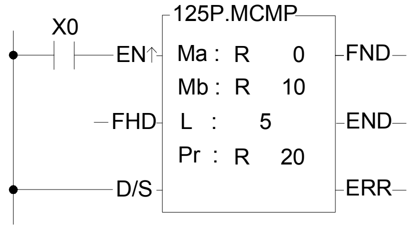
Md: Starting register of matrix a
Mb: Starting register of matrix b
L : Length of matrix (Ma, Mb)
Pr : Pointer register
Ma, Mb may combine with V, Z, P0~P9 to serve indirect address application

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

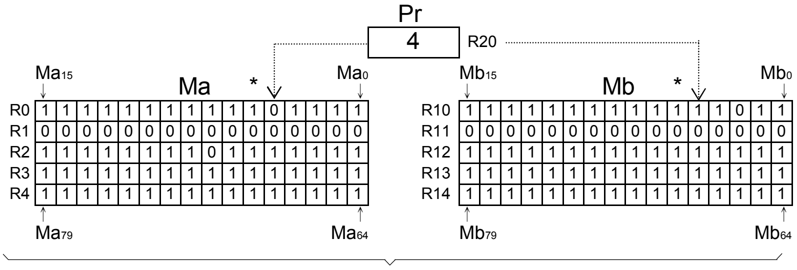
- When comparison control "EN" = 1 or "EN ↑ " (P instruction) has a transition from 0 to 1, then beginning from the top pair of bits (Ma_0 and Mb_0) within the 2 matrixes Ma and Mb (when "FHD" = 1 or Pr value is equal to 16L-1), or beginning from the next pair of bits (Mapr + 1 and Mbpr + 1) pointed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



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



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



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

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

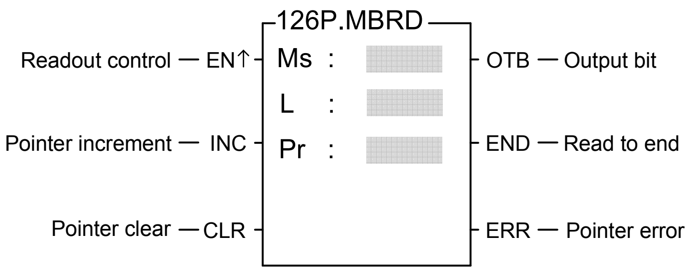
Before execution

	Pr	FND	END
① R20	39	1	0
② R20	79	0	1
③ R20	2	1	0

Execution result

FUN126 P MBRD	MATRIX BIT READ	FUN126 P MBRD

Ladder symbol



Ms : Starting register of matrix

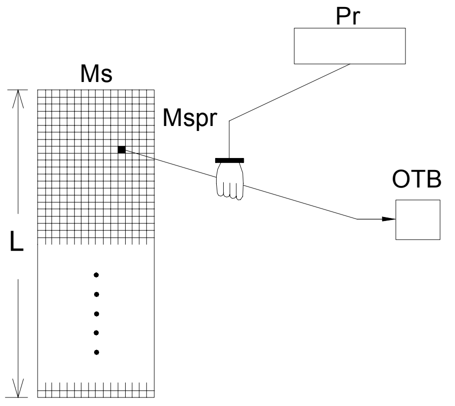
L : Matrix length

Pr : Pointer register

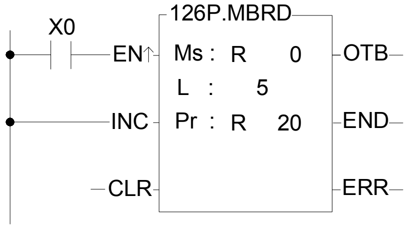
Ms may combine with V, Z, P0~P9 to serve indirect address application

Range 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 | C199	R0 | R3839	R3840 | R3903	R3904 | R3967	R3968 | R4167	R5000 | R8071	D0 | D4095	2 | 256	V、Z P0~P9
Ms	○	○	○	○	○	○	○	○	○	○	○	○		○
L							○				○*	○	○	
Pr		○	○	○	○	○	○		○	○*	○*	○		

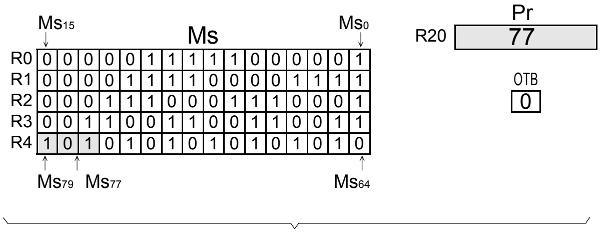
- When readout control "EN" = 1 or "EN ↑ " (P instruction) has a transition from 0 to 1, the status of the bit Mspr pointed by pointer Pr within matrix Ms will be read out and appear at the output bit "OTB". Before the readout, this instruction will first check the input -pointer clear "CLR". If "CLR"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



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

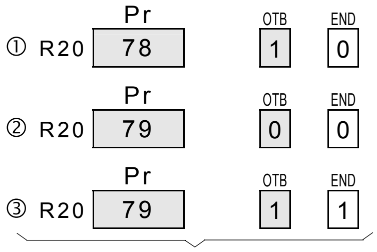


- In the program at left, INC = 1, so every time there is one readout the pointer will be increased by 1. With this way each bit in Ms may be read out successively, as shown at left in the diagram below. When X0 goes 3 times from 0→1, the results are shown at right in the diagram below .



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

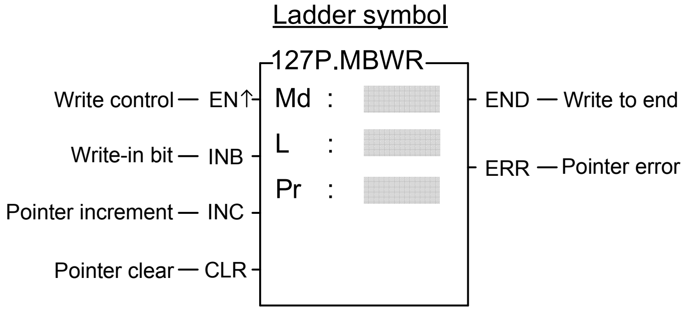
Before execution



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

Execution result

FUN127 P MBWR	MATRIX BIT WRITE	FUN127 P MBWR



Md : Starting register of matrix

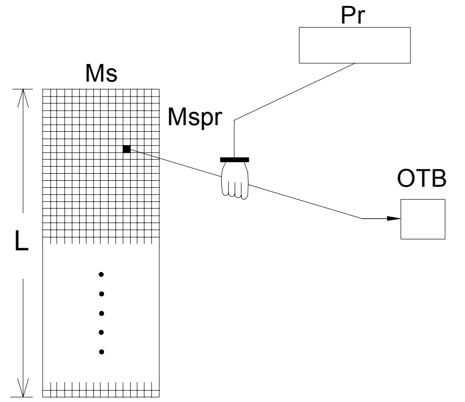
L : Matrix length

Pr : Pointer register

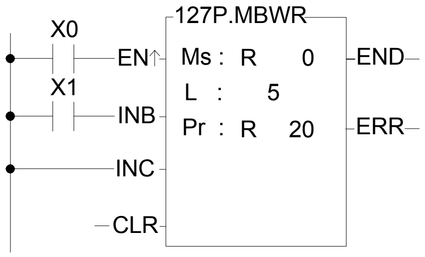
Md may combine with V, Z, P0~P9 to serve indirect address application

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

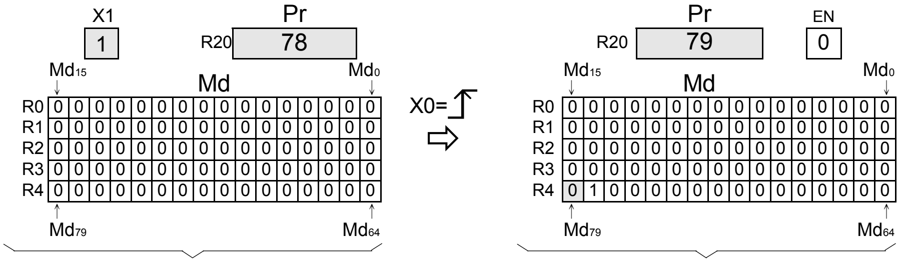
- When write control "EN" = 1 or "EN↑" (P instruction) has a transition from 0 to 1, the status of the write-in bit "INB" will be written into the bit Mdpr pointed by pointer Pr within matrix Md. Before the write-in takes place, the status of pointer clear "CLR" will be checked. If "CLR" is 1, then Pr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



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



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



Before execution

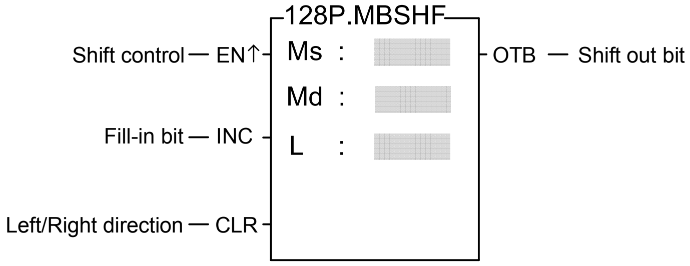
Md																
R0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

After execution

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

FUN128 P MBSHF	MATRIX BIT SHIFT	FUN128 P MBSHF

Ladder symbol



Ms : Starting register of source matrix

Md : Starting register of destination matrix

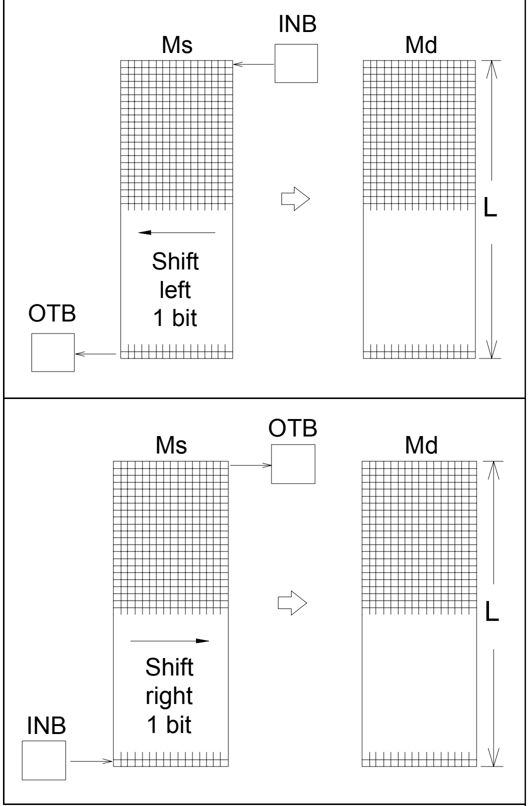
L : Length of matrix (Ms and Md)

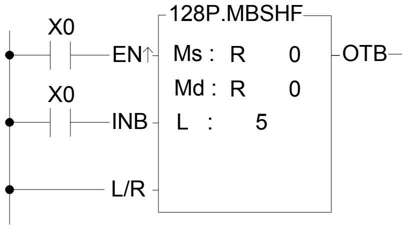
Ms, Md may combine with V, Z, P0~P9 to serve indirect address application

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

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

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





Ms

Ms_{15} ↓ … ↓ Ms_0

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

Ms_{79} ↑ … ↑ Ms_{64}

Before execution

X0=↑ ⇨

Md

Md_{15} ↓ … ↓ Md_0

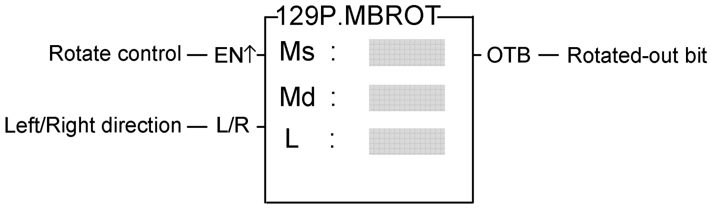
R0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
R1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0
R2	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	1
R3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
R4	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0

Md_{79} ↑ … ↑ Md_{64}

After execution

FUN129 P MBROT	MATRIX BIT ROTATE	FUN129 P MBROT

Ladder symbol



Ms : Starting register of source matrix

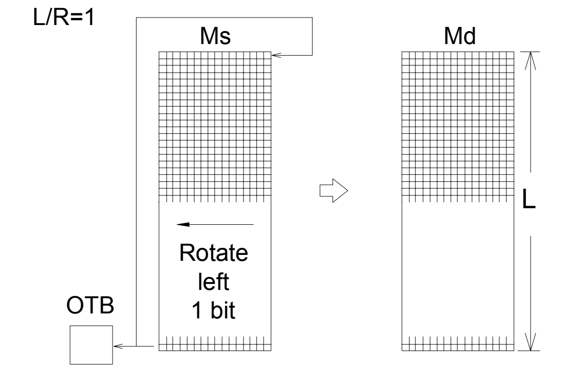
Md : Starting register of destination matrix

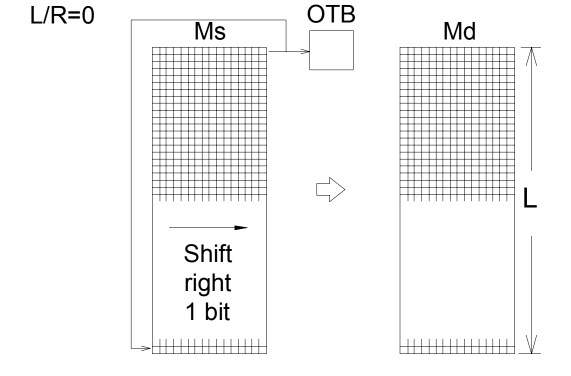
L : Length of matrix (Ms and Md)

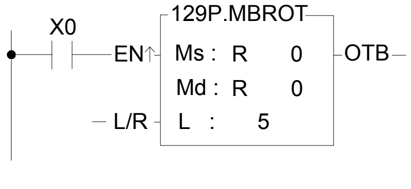
Ms, Md may combine with V, Z, P0~P9 to serve indirect address application

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

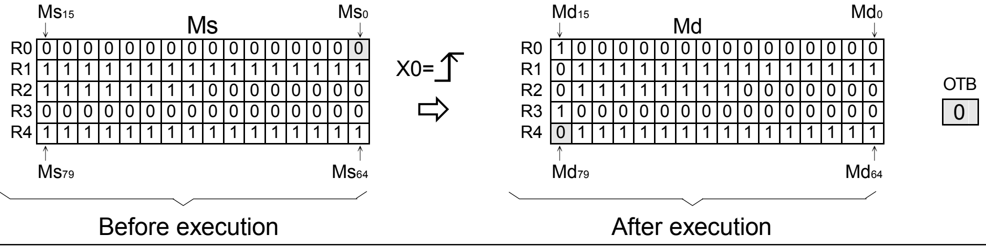
- When rotate control "EN" = 1 or "EN ↑ " (P instruction) has a transition from 0 to 1, matrix Ms will be completely retrieved and rotated by one bit towards the left (when L/R = 1) or to the right (when L/R = 0). The space created by the rotation (with a left rotation it will be M0, and with a right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







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



Ms

	Ms15															Ms0
R0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
R2	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0
R3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R4	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	Ms79															Ms64

Before execution

Md

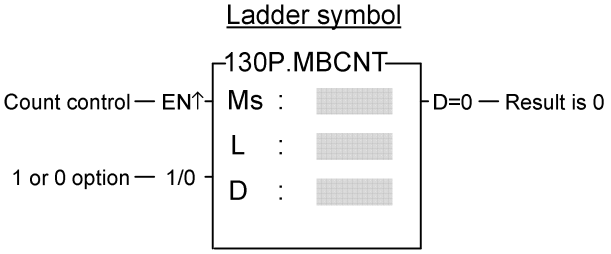
	Md15															Md0
R0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
R2	0	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0
R3	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R4	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	Md79															Md64

After execution

OTB 0

FUN130 P MBCNT	MATRIX BIT STATUS COUNT	FUN130 P MBCNT

Ladder symbol



Ms : Starting register of matrix

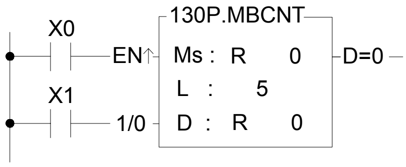
L : Matrix length

D : Register storing count results

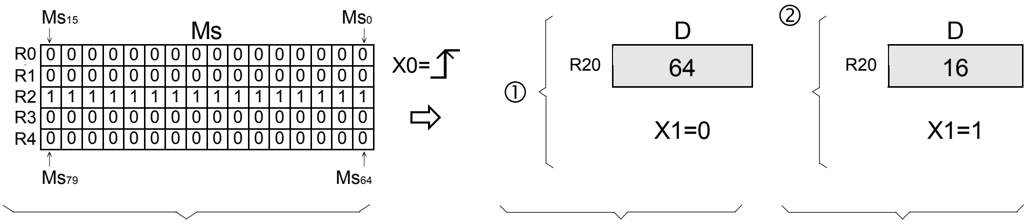
Ms may combine with V, Z, P0~P9 to serve indirect address application

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

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



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

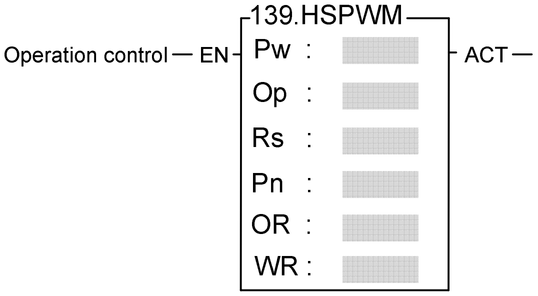


R0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R2	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
R3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

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

Ladder symbol



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

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

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

Pn : Setting of output frequency(0~255)

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

WR : Working register

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

Description

- When operation control “EN” = 1, the specified digital output will perform the PWM output, the expression for output frequency as shown bellow:

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

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

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

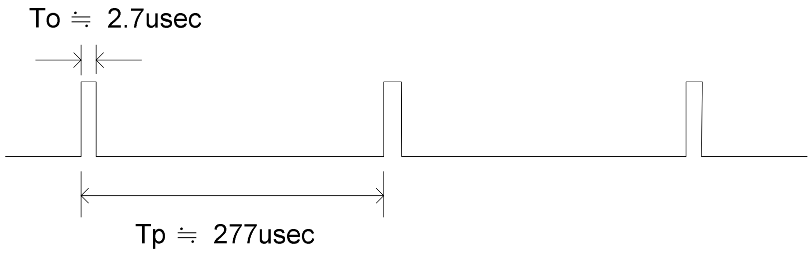
$$f_{pwm} = \frac{184320}{(50 + 1)} = 3614.117 \cdots ≒ 3.6\text{KHz}$$

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

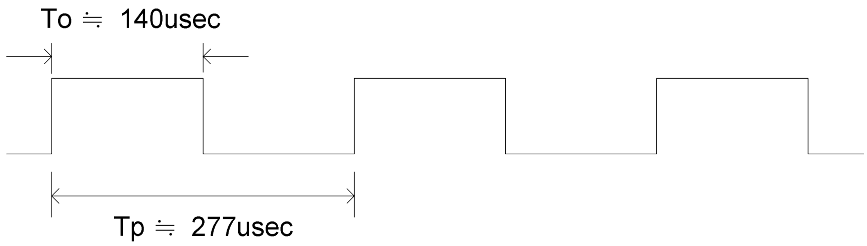
For Rs = 1/100, if OR(Setting of output pulse width) = 1, then T0 ≒ 2.7uS; if OR(Setting of output pulse width) = 50, then To ≒ 140uS.

.Output waveform :

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



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



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

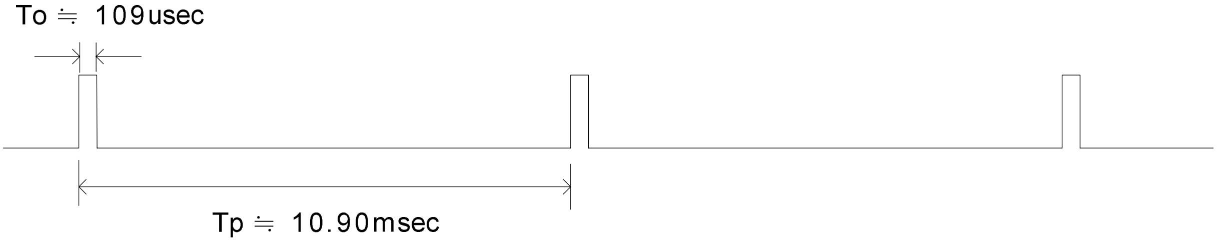
$$f_{pwm} = \frac{18432}{(200+1)} \fallingdotseq 91.7\text{Hz}$$

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

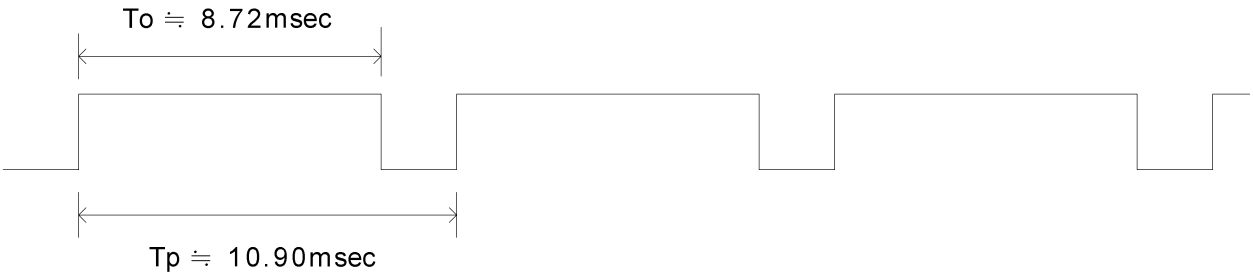
For Rs = 1/1000, if OR(Setting of output pulse width) = 10, then T0 ≒ 109uS; if OR(Setting of output pulse width) = 800, then To ≒ 8.72mS

.Output waveform :

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

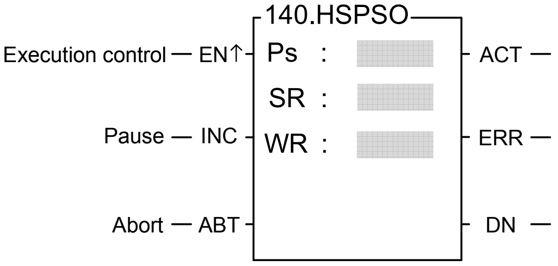


(2).Pn (Output frequency) = 200, Rs = 1 (1/1000), OR (Output pulse width) = 800 :



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

Ladder symbol



Ps : The Pulse Output (0～3) selection
0:Y0 & Y1
1:Y2 & Y3
2:Y4 & Y5
3:Y6 & Y7

SR : Positioning program starting register.

WR : Starting working register of instruction operation, total 7 registers, can not used in any other part of program.

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

Command descriptions

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

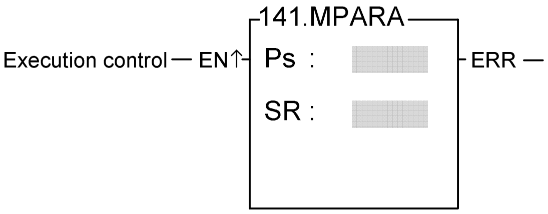
*** The working mode of Pulse Output must be configured (without setting, Y0～Y7 will be treated as normal output) to any one of following modes, before the HSPSO instruction can be worked.

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

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

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

Ladder symbol



Ps : The pulse output (0～3) selection

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

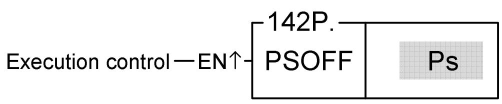
Range Ope- rand	HR R0 ｜ R3839	DR D0 ｜ D4095	ROR R5000 ｜ R8071	K 2 ｜ 256
Ps				0～3
SR	○	○	○	

Operation descriptions

- It is not necessary to use this instruction. if the system default for parameter values is matching what user demanded, then this instruction is not needed. However, if it needs to change the parameter value dynamically, this instruction is required.
- This instruction incorporates with FUN140 for positioning control purpose.
- Whether the execution control input “EN” = 0 or 1, this instruction will be performed.
- When there are any errors in parameter value, the output indication “ERR” will be ON. (The error code is stored in the error code register.)
- For detailed functional description and usage, please refer to chapter 13 “The NC positioning control of FBs-PLC” for explanation.

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

Ladder symbol



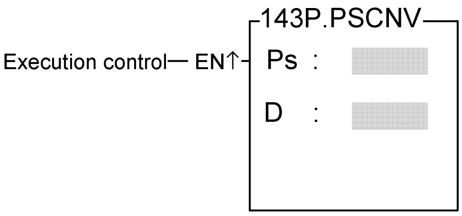
Ps : 0～3
Enforce the Pulse Output PSOn (n= Ps) to stop.

Command descriptions

- When execution control "EN" =1 or "EN ↑ " (P instruction) changes from 0→1, this instruction will enforce the assigned number set of HSPSO (High Speed Pulse Output) to stop pulse output.
- While in the application for mechanical original point reset, as soon as reach the original point can use this instruction to stop the pulse output immediately, so as to make the original point stop at the same position every time when performing mechanical original point resetting.
- For detailed functional description and usage, please refer to chapter 13 "The NC positioning control of FBs-PLC" for explanation.

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

Ladder symbol



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

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

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

Command descriptions

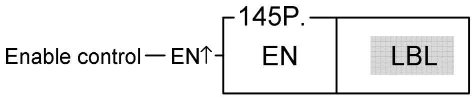
- When execution control "En" =1 or "EN ↑ "(P instruction) changes from 0→1, this instruction will convert the assigned current pulse position (PS) to be the mm (or Deg, Inch, or PS) that has same unit as the set value, so as to make current position displaying.

- Only when the FUN140 instruction is executed, then it can get the correct conversion value by executing this instruction.

- For detailed functional description and usage, please refer to chapter 13 "The NC positioning control of FBs-PLC" for explanation.

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

Ladder symbol



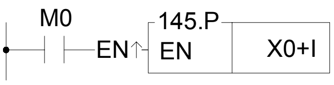
LBL : External input or peripheral label name that to be enabled.

- When enable control "EN" =1 or "EN↑" (P instruction) changes from 0→1, it allows the external input or peripheral interrupt action which is assigned by LBL.
- The enabled interrupt label name is as follows:(Please refer the section 10.3 for details)

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

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

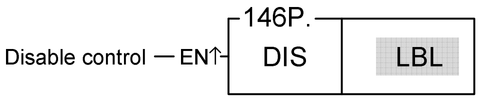
Program example



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

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

Ladder symbol



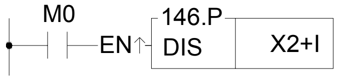
LBL : Interrupt label intended to disable or peripheral name to be disabled.

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

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

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

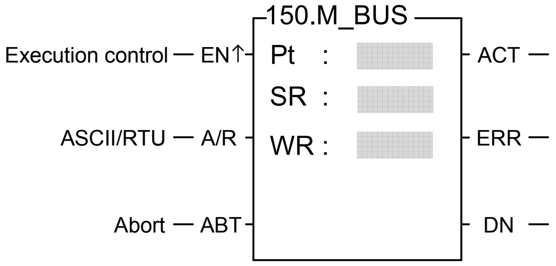
Program example



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

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

Ladder symbol



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

SR：Starting register of communication program

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

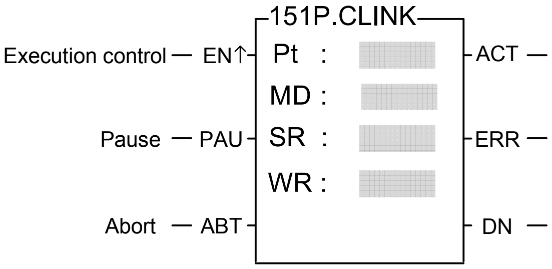
Range Ope- rand	HR	ROR	DR	K
	R0 | R3839	R5000 | R8071	D0 | D4095	
Pt				1～4
SR	○	○	○	
WR	○	○*	○	

Description

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

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

Ladder symbol



Pt : Assign the port, 1～4

MD : Communication mode, MD0~MD3

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

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

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

Description

● This instruction provides 4 instruction modes MD0～MD3. Of which, three instruction modes MD0～MD2, are “regular link network”, and the MD3 is the “high speed link network”. The following are the function description of respective modes. For the details, please refer to section 12.1.2 for explanation.

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

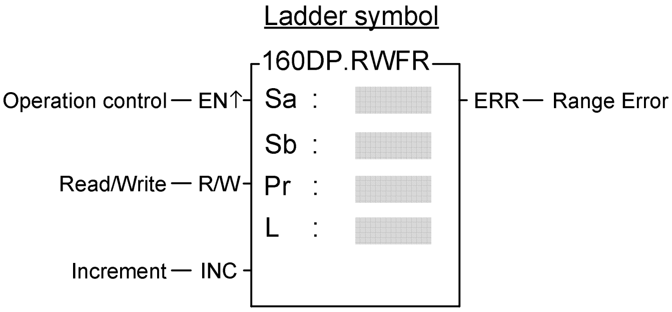
- MD1 : Active ASCII data transmission mode.
  With this mode, the FUN151 instruction will parse the communication program stored in data registers and base on the parsing result send the data from port2 to ASCII peripherals (such as computer, other brand PLC, inverter, moving sign, etc, this kind of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

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

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

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

Ladder symbol



Sa: Starting address of data register

Sb: Starting address of file register

Pr : Record pointer register

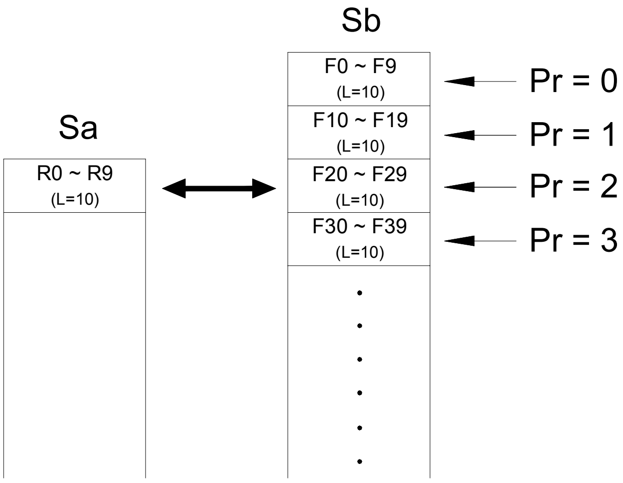
L : Quantity of register to form a record, 1~511

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

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

Description

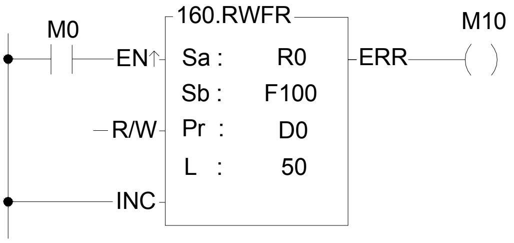
- When operation control "EN"=1 or "EN ↑ "(P instruction) changes from 0→1,it will perform the read ("R/W"=1) or write ("R/W"=0) file register operation. While reading, the content of data registers starting from Sa will be overwritten by the content of file registers addressed by the base file register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



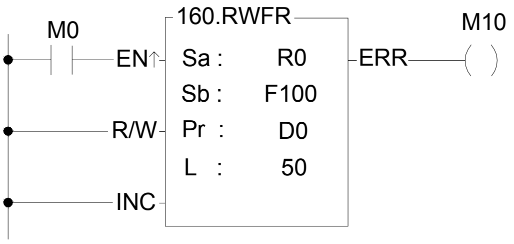
FUN160 D P RWFR	READ/WRITE FILE REGISTER	FUN160 D P RWFR

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

.



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



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